

**Ace the Exam Series®**

**2022**

**FIRST EDITION**

# **AWS**

**CERTIFIED DATA ANALYTICS - SPECIALTY**

**STUDY GUIDE WITH PRACTICE QUESTIONS & LABS**



**UP-TO-DATE EXAM BLUEPRINT**

LATEST EXAM QUESTIONS AND REGULARLY  
REFRESHED CONTENT FOR YOU TO MASTER  
YOUR EXAM CERTIFICATION



**ACE EXAMS WITH CONFIDENCE**

OUR PROVEN AND COMPREHENSIVE STUDY  
MATERIAL ENSURES PASSING GRADES



**IPSpecialist**

Empower Your Skills for the Digital Future

# **DAS-C01: AWS Certified Data Analytics - Specialty**

---

Study Guide with Practice Questions & Labs

First Edition

---

[www.ipsspecialist.net](http://www.ipsspecialist.net)

**Document Control**

Proposal Name : AWS Certified Data Analytics  
Specialty

Document Edition : First Edition

Document Release Date : 9<sup>th</sup> February 2022

Reference : DAS-Co1

Copyright © 2022 IPSpecialist LTD.

Registered in England and Wales

Company Registration No: 10883539

Registration Office at: Office 32, 19-21 Crawford Street, London W1H 1PJ,  
United Kingdom

[www.ipspecialist.net](http://www.ipspecialist.net)

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without the written permission from IPSpecialist LTD, except for the inclusion of brief quotations in a review.

Feedback:

If you have any comments regarding the quality of this book, or otherwise

alter it to better suit your needs, you can contact us through email at [info@ipspecialist.net](mailto:info@ipspecialist.net)

Please make sure to include the book's title and ISBN in your message.

## About IPSpecialist

[IPSPECIALIST](#) LTD. IS COMMITTED TO EXCELLENCE AND DEDICATED TO YOUR SUCCESS.

Our philosophy is to treat our customers like family. We want you to succeed, and we are willing to do everything possible to help you make it happen. We have the proof to back up our claims. We strive to accelerate billions of careers with great courses, accessibility, and affordability. We believe that continuous learning and knowledge evolution are the most important things to keep re-skilling and up-skilling the world.

Planning and creating a specific goal is where IPSpecialist helps. We can create a career track that suits your visions as well as develop the competencies you need to become a professional Network Engineer. Based on the career track you choose, we can also assist you with the execution and evaluation of your proficiency level, as they are customized to fit your specific goals.

We help you STAND OUT from the crowd through our detailed IP training content packages.

### Course Features:

- Self-Paced Learning
  - Learn at your own pace and in your own time
- Covers Complete Exam Blueprint
  - Prep-up for the exam with confidence
- Case Study Based Learning
  - Relate the content with real-life scenarios
- Subscriptions that Suits You

- Get more and pay less with IPS subscriptions
- Career Advisory Services
  - Let the industry experts plan your career journey
- Virtual Labs to test your skills
  - With IPS vRacks, you can evaluate your exam preparations
- Practice Questions
  - Practice questions to measure your preparation standards
- On Request Digital Certification
  - On request digital certification from IPSpecialist LTD.

### **About the Authors:**

This book has been compiled with the help of multiple professional engineers. These engineers specialize in different fields e.g. Networking, Security, Cloud, Big Data, IoT, etc. Each engineer develops content in its specialized field that is compiled to form a comprehensive certification guide.

### **About the Technical Reviewers:**

#### **Nouman Ahmed Khan**

AWS-Architect, CCDE, CCIEX5 (RandS, SP, Security, DC, Wireless), CISSP, CISA, CISM is a Solution Architect working with a major telecommunication provider in Qatar. He works with enterprises, mega-projects, and service providers to help them select the best-fit technology solutions. He also works closely as a consultant to understand customer business processes and helps select an appropriate technology strategy to support business goals. He has more

than 14 years of experience working in Pakistan/Middle-East and UK. He holds a Bachelor of Engineering Degree from NED University, Pakistan, and M.Sc. in Computer Networks from the UK.

### **Abubakar Saeed**

Abubakar Saeed has more than twenty-five years of experience, Managing, Consulting, Designing, and implementing large-scale technology projects, extensive experience heading ISP operations, solutions integration, heading Product Development, Presales, and Solution Design. Emphasizing adhering to Project timelines and delivering as per customer expectations, he always leads the project in the right direction with his innovative ideas and excellent management.

### **Dr. Fahad Abdali**

Dr. Fahad Abdali is a seasoned leader with extensive experience managing and growing software development teams in high-growth start-ups. He is a business entrepreneur with more than 18 years of experience in management and marketing. He holds a Bachelor's Degree from NED University of Engineering and Technology and a Doctor of Philosophy (Ph.D.) from the University of Karachi.

### **Mehwish Jawed**

Mehwish Jawed is working as a Senior Research Analyst. She holds a Master's and Bachelors of Engineering degree in Telecommunication Engineering from NED University of Engineering and Technology. She

also worked under the supervision of HEC Approved supervisor. She has more than three published papers, including both conference and journal papers. She has a great knowledge of TWDM Passive Optical Network (PON). She also worked as a Project Engineer, Robotic Trainer in a private institute and has research skills in the field of communication networks. She has both technical knowledge and industry-sounding information, which she utilizes effectively when needed. She also has expertise in cloud platforms, such as AWS, GCP, Oracle, and Microsoft Azure.

### **Rafia Muzaffar**

Rafia Muzaffar is working as a Technical Content Developer. She holds a Bachelor's of Engineering degree in Telecommunication Engineering from NED University of Engineering and Technology. She possesses exceptional research and writing skills. She is enthusiastic and passionate in her academic pursuit. She has a sound knowledge of Networking, IoT, and Cloud and also knows multiple programming languages, including MATLAB, HTML, CSS, React. Js, C# and Java.

### **Mohammad Usman Khan**

Muhammad Usman Khan is a Technical Content Developer. He holds a Bachelor's Degree in Telecommunication Engineering from Sir Syed University of Engineering and Technology. He holds the First Position in Telecommunication Engineering and received two Gold Medals, the first from Sir Syed University of Engineering and Technology. He worked on many Deep Learning projects. He is certified by the

National Center of Artificial Intelligence (NCAI), a research institute of the Government of Pakistan in Artificial Intelligence. He is also certified by the Nvidia Deep Learning Institute in Deep Learning with Computer Vision.

### **Syeda Fariha Ashrafi**

Syeda Fariha Ashrafi is working as a technical content developer. She has completed bachelor's degree in telecommunication engineering from NED University of Engineering and Technology. She has also completed the CCNA (Routing and Switching) course. During her bachelor's program, she has worked on the project "Smart metering using PLC (Power Line Communication).

### **Free Resources**

For Free Resources: Please visit our website and register to access your desired Resources Or contact us at: [helpdesk@ipspecialist.net](mailto:helpdesk@ipspecialist.net)

**Career Report:** This report is a step-by-step guide for a novice who wants to develop his/her career in the field of computer networks. It answers the following queries:

- What are the current scenarios and future prospects?
- Is this industry moving towards saturation, or are new opportunities knocking at the door?
- What will the monetary benefits be?
- Why get certified?
- How to plan, and when will I complete the certifications if I start today?
- Is there any career track that I can follow to accomplish the specialization level?

Furthermore, this guide provides a comprehensive career path towards being a specialist in networking and highlights the tracks needed to obtain certification.

**IPS Personalized Technical Support for Customers:** Good customer service means helping customers efficiently, in a friendly manner. It is essential to be able to handle issues for customers and do your best to ensure they are satisfied. Providing good service is one of the most important things that can set our business apart from the others of its kind.

Excellent customer service will result in attracting more customers and attain maximum customer retention.

IPS offers personalized TECH support to its customers to provide better value for money. If you have any queries related to technology and labs, you can simply ask our technical team for assistance via Live Chat or Email.

## **Our Products**

### **Study Guides**

IPSpecialist Study Guides are the ideal guides to developing the hands-on skills necessary to pass the exam. Our study guides cover the official exam blueprint and explain the technology with real-life case study-based labs. The content covered in each study guide consists of individually focused technology topics presented in an easy-to-follow, goal-oriented, step-by-step approach. Every scenario features detailed breakdowns and thorough verifications to help you completely understand the task and associated technology.

We extensively used mind maps in our study guides to visually explain the technology. Our study guides have become a widely used tool to learn and remember information effectively.

### **vRacks**

Our highly scalable and innovative virtualized lab platforms let you practice the IPSpecialist Study guide at your own time and your own place as per your convenience.

## **Exam Cram**

Our Exam cram notes are a concise bundling of condensed notes of the complete exam blueprint. It is an ideal and handy document to help you remember the most important technology concepts related to the certification exam.

## **Practice Questions**

IP Specialists' Practice Questions are dedicatedly designed from a certification exam perspective. The collection of these questions from our Study Guides is prepared to keep the exam blueprint in mind, covering not only important but necessary topics as well. It is an ideal document to practice and revise your certification.

# **Content at a glance**

**Chapter 01: Introduction**

**Chapter 02: Amazon Simple Storage Service**

**Chapter 03: Databases in AWS**

**Chapter 04: Collecting Streaming Data**

**Chapter 05: Data Collection and Getting Data into AWS**

**Chapter 06: Amazon Elastic Map Reduce**

**Chapter 07: Using Redshift**

**Chapter 08: Redshift Maintenance and Operations**

**Chapter 09: AWS Glue, Athena, and QuickSight**

**Chapter 10: ElasticSearch**

**Chapter 11: AWS Security Services**

**Answers**

**Acronyms**

**References**

**About Our Products**

# Table of Contents

## **Chapter 01: Introduction**

[Course Introduction](#)

[Recommended AWS Knowledge](#)

[What is Data Analytics?](#)

[Steps for Success](#)

[Digital Use Cases](#)

## **Chapter 02: Amazon Simple Storage Service**

[Introduction to S3](#)

[Getting Data Into S3 - Concepts, AWS Management Console, AWS CLI](#)

[Upload Interfaces](#)

[Transfer Acceleration](#)

[Demo 2-01: AWS Management Console](#)

[Demo 2-02: AWS CLI](#)

[Getting Data Into S3 - Boto3](#)

[Demo: Python Boto3 SDK](#)

[S3 Multipart Upload](#)

[What do we do when we have a lot of data?](#)

[Three Multipart Upload API Calls](#)

[Considerations](#)

[Best Practices and Limitations](#)

[Demo 2-03: Exploring Multipart Upload Script](#)

[Demo 2-04: Uploading a File with Multipart Upload Script](#)

[S3 Storage Classes](#)

[What are Storage Classes?](#)

[Availability and Durability](#)

[S3 Standard](#)

[S3 Infrequent Access](#)

[S3 Standard-IA](#)

[S3 Intelligent Tiering](#)

[S3 Glacier](#)

[S3 Glacier Deep Archive](#)

[S3 Lifecycle Policies](#)

[Life Of Data](#)

[Data Lifecycle in S3](#)

[How To Make It Happen](#)

[Demo 2-05: Applying S3 Lifecycle Policy](#)

[S3 Security and Encryption](#)

[S3 Security Overview](#)

[In-Flight Security](#)

[At Rest Security](#)

[Client-Side Encryption](#)

[Server-Side Encryption](#)

[The S3 Access Security Waterfall](#)

[Access Logging, Alerting, and Auditing](#)

[Object Protection and Replication](#)

[Lab 2-01: Programmatically Utilizing Data from S3](#)

[Introduction](#)

[Problem](#)

[Solution](#)

[Mind Map](#)

[Practice Questions](#)

## **[Chapter 03: Databases in AWS](#)**

[Introduction](#)

[Organizing Data](#)

[Services](#)

[\*\*Relational Database\*\*](#)

[\*\*Non-Relational Database\*\*](#)

[Relational Database Service](#)

[Introduction](#)

[Managed Service](#)

[Second Level Service](#)

[RDS Instance](#)

[Operation System Access](#)

[Disaster Recovery](#)

[Neptune](#)

[Introduction](#)

[Graph Structure](#)

[Interface Languages](#)

[Comparison to Relation Database Service](#)

[Neptune Use Cases](#)

[DocumenDB](#)

[Introduction](#)

[DocumentDB Features](#)

[DocumentDB Use Cases](#)

[Serverless Options](#)

[Introduction](#)

[S3 Select](#)

[Athena](#)

[DynamoDB](#)

[Aurora](#)

[Aurora Serverless](#)

[Lab 3-01: Programmatically Utilizing S3 Select](#)

[Introduction](#)

[Problem](#)

[Solution](#)

**[Mind Map](#)**

**[Practice Questions](#)**

**[Chapter 04: Collecting Streaming Data](#)**

[Introduction to Collecting Streaming Data](#)

[Kinesis Family](#)

[Data Collection](#)

[Data Collection Methods](#)

[Big Data Collection](#)

[Streaming Data](#)

[AWS Kinesis Family](#)

[Kinesis Data Streams](#)

[Introduction](#)

[Working with Kinesis Data Streams](#)

[Benefits of Using Kinesis Data Streams](#)

[Shard](#)

[Processing & Storage](#)

[Interacting with Kinesis Data Stream](#)

[KPL vs. Kinesis API](#)

[Kinesis Data Stream Use Cases](#)

[Lab 4-01: AWS Kinesis Data Stream](#)

[Kinesis Data Firehose](#)

[Introduction](#)

[AWS Kinesis Firehose Key Concepts](#)

[Kinesis Firehose Data Flow](#)

[S3 Destination](#)

[Redshift Destination](#)

[Elasticsearch Destination](#)

[Splunk Destination](#)

[Buffer Size & Buffer Interval](#)

[Kinesis Firehose Use Cases](#)

[Demo: Kinesis Data Firehose](#)

[Kinesis Video Streams](#)

[Introduction](#)

[Producer & Consumer Applications](#)

[Real-time vs. Batch-oriented](#)

[Kinesis Video Stream Benefits](#)

[Kinesis Video Streams Working](#)

[AWS Kinesis Video Stream Use Cases](#)

[Kinesis Data Analytics](#)

[Introduction](#)

[AWS Kinesis Data Analytics Benefits](#)

[Kinesis Data Analytics Working](#)

[AWS Kinesis Data Analytics Use Cases](#)

[Demo: Kinesis Data Analytics](#)

[Amazon Managed Streaming for Kafka](#)

[Apache Kafka](#)

[Apache Kafka Publish/Subscribe Conceptually](#)

[Apache Kafka Management](#)

[Amazon MSK](#)

[MSK Architecture](#)

[Compare MSK & Kinesis Data Stream](#)

[Streaming Services Uses Cases](#)

[Lab 4-02: Joining, Enriching, & Transforming Streaming Data with Amazon Kinesis](#)

[Introduction](#)

[Problem](#)

[Solution](#)

[Mind Map](#)

[Practice Questions](#)

## **[Chapter 05: Data Collection and Getting Data into AWS](#)**

[Introduction](#)

[Data Loses Value Quickly Over Time](#)

[Direct Connect, Snowball, Snowball Edge, Snowmobile](#)

[AWS General Rule Of Thumb](#)

[Data Migration Service \(Managed Services To Move Your Data To AWS\)](#)

[Which Solution Should you Use?](#)

[Database Migration Service](#)

[DMS Use Cases](#)

[Supported Migrations](#)

[Migrations](#)

[Mass Amount Of Data](#)

[Replication](#)

[Data Pipeline](#)

[Creating Data Pipeline](#)

[Some Overlap With Lambda](#)

[Key Concepts](#)

[Data Pipeline for On-premises](#)

[Lambda, API Gateway, and CloudFront](#)

[Definitions](#)

[Lambda Events and Integration](#)

[Lambda – Use Cases](#)

[Lambda Limits](#)

[Serverless Architectures](#)

[Kinesis and Lambda Integrations](#)

[Kinesis and Lambda Scaling](#)

[Comparing our Options](#)

[Time Required To Move Data Into AWS](#)

[Choosing A Service](#)

[Mind Map](#)

[Practice Questions](#)

## **[Chapter 06: Amazon Elastic Map Reduce \(EMR\)](#)**

[Introduction](#)

[Apache Hadoop and EMR Software Collection](#)

[Map Reduce](#)

[Distributed File Systems](#)

[Hadoop Distributed File System \(HDFS\)](#)

[EMR](#)

[EMR Architecture](#)

[Introduction](#)

[Primary Node Features](#)

[Core Node Features](#)

[Task Node Features](#)

[Single Availability Zone Concept](#)

[EMR Storage Options](#)

[EMR Operations - Transient vs. Long-Running](#)

[Transient Clusters](#)

[Long-Running Clusters](#)

[Considerations](#)

[EMR Operations - Choosing an Instance Type](#)

[Choosing an Instance Type](#)

[EMR Operations - Choosing the Right Number of Instances](#)

[Choosing the Right Number of Instances](#)

[HDFS Capacity Guidelines](#)

[EMR Operations - On-Demand and Spot Instances](#)

[Quick Reference for Application Scenarios](#)

[EMR Operations - Monitoring and Resizing Clusters](#)

[CloudWatch Events](#)

[CloudWatch Metrics](#)

[Monitor a Cluster with UI](#)

[Resizing a Cluster – Manually](#)

[Resizing a Cluster – Auto Scaling](#)

[EMR File Storage and Compression](#)

[How Hadoop Splits Files?](#)

[Different Compression Algorithms](#)

[The Benefits of File Compression](#)

[Different EMR File Formats](#)

[File Sizes Best Practices](#)

[S3DistCp Command](#)

[Lab 6-01: Data Analytics with Spark and EMR](#)

[Introduction](#)

[Problem](#)

[Solution](#)

[Mind Map](#)

[Practice Questions](#)

## **[Chapter 07: Using Redshift](#)**

[Introduction](#)

[Redshift Architecture](#)

[Cluster](#)

[Node](#)

[Slice](#)

[Redshift Query Process](#)

[Redshift in the AWS Service Ecosystem](#)

[Redshift Use Cases](#)

[Data Warehouse VS Data Lake](#)

[What makes Redshift Different?](#)

[Redshift Table Design](#)

[Data Types](#)

[Compression](#)

[Sort Keys](#)

[Distribution Styles](#)

[Constraints](#)

[Redshift Spectrum](#)

[How do you query flow?](#)

[Demo 7-01: Redshift](#)

[Lab 7-01: Querying Data from Multiple Redshift Spectrum Tables](#)

[Introduction](#)

[Problem](#)

[Solution](#)

[Mind Map](#)

[Practice Questions](#)

## **[Chapter 08: Redshift Maintenance and Operations](#)**

[Launching a Redshift Cluster](#)

[Interfaces](#)

[Required Parameters](#)

[Considerations](#)

[Demo 8-01: Launching A Cluster In A Web Console](#)

[Resizing a Redshift Cluster](#)

[Classic Resize](#)

[Elastic Resize](#)

[Utilizing Vacuum and Deep Copy](#)

[The Vacuum Process](#)

[Vacuum Options](#)

[Automatic Vacuuming](#)

[Deep Copy Methods](#)

[Backup and Restore](#)

[Snapshots](#)

[Restoring from Snapshot](#)

[Loading Data From S3](#)

[Unloading Data To S3](#)

[Monitoring](#)

[Redshift Console](#)

[CloudWatch](#)

[Demo 8-02: Monitoring An Active Cluster](#)

## Lab 8-01: Manually Migrating Data Between Redshift Clusters

Introduction

Problem

Solution

Mind Map

Practice Questions

## Chapter 09: AWS Glue, Athena, and QuickSight

Introduction

Glue Data Catalog

What is AWS?

AWS Glue - Use cases

AWS Glue Components

AWS Glue Data Catalog

Demo 9-01: Populating the AWS Glue Data Catalog

Converting Semi-Structured Schemas to Rational Schemas

Glue Jobs

AWS Glue Jobs

Workflow Overview

Lab 9-01: AWS Glue Jobs

Output File Formats

Data Processing Units (DPUs)

Glue Jobs Run In Isolated

AWS Glue Jobs – Glue jobs are the business logic that performs ETLs work in AWS Glue.

Workflow Overview – Various parts are needed for an AWS Glue job.

Output Data Formats – The different output formats Glue Jobs can perform.

[Data Processing Units \(DPUs\) – The units used for processing your Glue Jobs.](#)

[Glue Jobs Run In Isolated – Glue Jobs run on virtual resources, Glue jobs needs, and how traffic is governed.](#)

[Job Bookmarks](#)

[Explanation](#)

[Job Bookmarks Defined](#)

[Options For Job Bookmarks](#)

[Demo 9-01: How To Set Up Job Bookmarks](#)

[Getting Started with Athena](#)

[What Is Athena?](#)

[Athena Federated Queries](#)

[Athena Data Formats And Integrations](#)

[Connecting to Data Sources](#)

[More On Integrations](#)

[Athena Use Cases](#)

[\*\*Demo 9-02: Amazon Athena\*\*](#)

[\*\*Introduction\*\*](#)

[\*\*Problem\*\*](#)

[\*\*Solution\*\*](#)

[When To Use Athena](#)

[S3 Select And Glacier Select](#)

[Overview Of Similar Services](#)

[Comparing Athena To Other Services](#)

[QuickSight Visualizations and Dashboards](#)

[Amazon QuickSight](#)

[How QuickSight Works?](#)

[Visualization types](#)

[QuickSight Dashboards](#)

[QuickSight Security and Authentication](#)

[QuickSight Data Encryption](#)

[Connecting To AWS Resources](#)

[Identity And Access Management In Quicksight](#)

[Best Practices For Security](#)

[Mind Map](#)

[Practice Questions](#)

## **[Chapter 10: Elasticsearch](#)**

[Introduction to Elasticsearch](#)

[Elasticsearch Service](#)

[Searching](#)

[Logs Into Data](#)

[Using Elasticsearch](#)

[JSON All the Way Down](#)

[The Interface](#)

[Loading Data](#)

[Demonstration Pipeline](#)

[Service Integration](#)

[Demo 10-01: Querying Elasticsearch](#)

[Visualizing Elasticsearch Data](#)

[Visualization Tools Examples](#)

[What can We Visualize?](#)

## **[Lab 10-01: Implementing an Elasticsearch \(OpenSearch\) Backed Search Microservice](#)**

[Introduction](#)

[Problem](#)

[Solution](#)

[Mind Map](#)

[Practice Questions](#)

## [Chapter 11: AWS Security Services](#)

[Introduction](#)

[Overview](#)

[Identity Access Management](#)

[Introduction](#)

[IAM Concept](#)

[IAM Permission Objects](#)

[IAM Features](#)

[IAM Secured Services](#)

[External Identity Federation](#)

[Key Management System](#)

[Introduction](#)

[KMS Concept](#)

[AWS KMS Keys](#)

[Customer Managed Keys](#)

[Symmetric KMS Keys](#)

[Asymmetric KMS Keys](#)

[Data Keys](#)

[Customer Master Key](#)

[Envelope Encryption](#)

[KMS Encryption Flows](#)

[Secrets Manager](#)

[Secrets Manager Concept](#)

## Secrets Manager Features

### Secret Storage

### Different Secrets Types Store in AWS Secrets Manager

### Encrypt Secret Data

### Secret Rotation

### Automatically Rotate Secrets

### Rotation Strategies

## VPC Network Security Features

### VPC Concept

### Components of VPC

### Network Access Control List

### Security Groups

### Traffic Monitoring

## Lab 11-01: Advanced S3 Security Configuration

### Introduction

### Problem

### Solution

## Mind Map

## Practice Questions

## Answers

### Chapter 02: Amazon Simple Storage Service

### Chapter 03: Databases in AWS

### Chapter 04: Collecting Streaming Data

### Chapter 05: Data Collection and Getting Data into AWS

### Chapter 06: Amazon Elastic Map Reduce

### Chapter 07: Using Redshift

### Chapter 08: Redshift Maintenance and Operations

[Chapter 09: AWS Glue, Athena, and QuickSight](#)

[Chapter 10: ElasticSearch](#)

[Chapter 11: AWS Security Services](#)

[Acronyms](#)

[References](#)

[About Our Products](#)

## **AWS Cloud Certifications**

AWS Certifications are industry-recognized credentials that validate your technical cloud skills and expertise while assisting in your career growth. These are one of the most valuable IT certifications right now since AWS has established an overwhelming lead in the public cloud market. Even with the presence of several tough competitors such as Microsoft Azure, Google Cloud Engine, and Rackspace, AWS is by far the dominant public cloud platform today, with an astounding collection of proprietary services that continues to grow.

The two key reasons as to why AWS certifications are prevailing in the current cloud-oriented job market:

- There's a dire need for skilled cloud engineers, developers, and architects – and the current shortage of experts is expected to continue into the foreseeable future.
- AWS certifications stand out for their thoroughness, rigor, consistency, and appropriateness for critical cloud engineering positions.

## **Value of AWS Certifications**

AWS places equal emphasis on sound conceptual knowledge of its entire platform, as well as on hands-on experience with the AWS infrastructure and its many unique and complex components and services.

### **For Individuals**

- Demonstrates your expertise to design, deploy, and operate highly available, cost-effective, and secure applications on AWS.
- Gain recognition and visibility for your proven skills and

proficiency with AWS.

- Earn tangible benefits such as access to the AWS Certified LinkedIn Community, invite to AWS Certification Appreciation Receptions and Lounges, AWS Certification Practice Exam Voucher, Digital Badge for certification validation, AWS Certified Logo usage, access to AWS Certified Store.
- Foster credibility with your employer and peers.

### **For Employers**

- Identify skilled professionals to lead IT initiatives with AWS technologies.
- Reduce risks and costs to implement your workloads and projects on the AWS platform.
- Increase customer satisfaction.

## **Types of Certification**

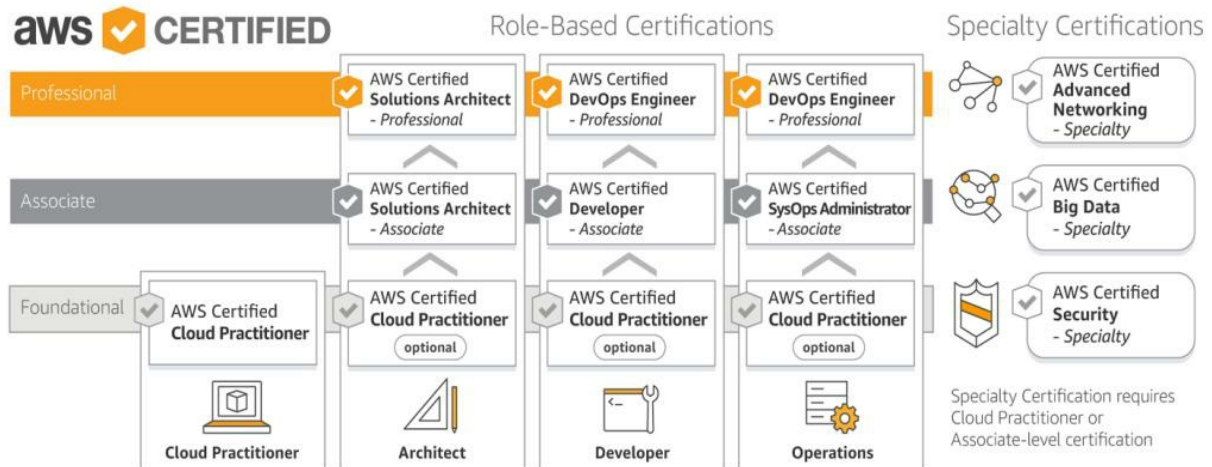
### **Role-Based Certifications:**

- ***Foundational*** - Validates overall understanding of the AWS Cloud. Prerequisite to achieving Specialty certification or an optional start towards Associate certification.
- ***Associate*** - Technical role-based certifications. No prerequisite.
- ***Professional*** - Highest level technical role-based certification. Relevant Associate certification required.

### **Specialty Certifications:**

- Validate advanced skills in specific technical areas.
- Require one active role-based certification.

## **Certification Roadmap**



## About AWS – Certified Data Analytics – Specialty Exam

|                     |   |
|---------------------|---|
| Exam Questions      | Case study, short answer, repeated answer, MCQs |
| Number of Questions | 250-270   |
| Time to Complete    | 180 minutes                                     |
| Exam Fee            | 300 USD   |

### Overview of AWS Data Analytics – Specialty Certification

Individuals with experience and competence working with AWS services to design, construct, protect, and maintain analytics systems can pursue the AWS Certified Data Analytics - Specialty. We recommend that you have the following items before taking this exam:

- Experience with typical data analytics technologies for five years
- Two years of hands-on experience and competence designing, building, securing, and maintaining analytics applications using AWS services.
- Ability to define AWS data analytics services and how they work together Ability

to explain how AWS data analytics services fit into the data lifecycle of collection, storage, processing, and visualizations

The basic knowledge and skills required at this level should include all of the areas, and objective components are given below:

### ***AWS Knowledge***

A minimum of 5 years of experience with popular data analytics tools is required for the target applicant. The ideal applicant will also have at least 2 years of hands-on experience and skill designing, building, securing, and maintaining analytics solutions using AWS services.

### ***General IT Knowledge***

- 1-2 years' experience as a system's administrator in a systems operations role
- Experience in understanding virtualization technology
- Monitoring and auditing system's experience
- Knowledge of networking concepts (DNS, TCP/IP, and Firewalls)
- Ability to collaborate with developers

### ***Intended Audience***

Eligible candidates for this exam must have:

- One or more years of hands-on experience in operating AWS-based applications
- Experience in provisioning, operating and maintaining systems running on AWS
- Ability to identify and gather requirements to define a solution to be built and operated on AWS
- Capabilities to provide AWS operations and deployment guidance and best practices throughout the life cycle of a

project

## **Recommended AWS Knowledge**

- Determine the operational characteristics of the collection system
- Select a collection system that handles the frequency, volume, and the source of data
- Select a collection system that addresses the key properties of data, such as order, format, and compression
- Determine the operational characteristics of the storage solution for analytics
- Determine data access and retrieval patterns
- Select appropriate data layout, schema, structure, and format
- Define data lifecycle based on usage patterns and business requirements
- Determine the appropriate system for cataloging data and managing metadata
- Determine appropriate data processing solution requirements
- Design a solution for transforming and preparing data for analysis
- Automate and operationalize data processing solutions
- Determine the operational characteristics of the analysis and visualization solution
- Select the appropriate data analysis solution for a given scenario
- Select the appropriate data visualization solution for a given scenario
- Select appropriate authentication and authorization

mechanisms

- Apply data protection and encryption techniques
- Apply data governance and compliance controls

The table below lists the main content domains and their weightings on the exam.

|          | Domain                      | Percentage |
|----------|-----------------------------|------------|
| Domain 1 | Collection                  | 18%        |
| Domain 2 | Storage and Data Management | 22%        |
| Domain 3 | Processing                  | 24%        |
| Domain 4 | Analysis and Virtualization | 18%        |
| Domain 5 | Security                    | 18%        |
| Total    |                             | 100%       |

# CHAPTER 01: INTRODUCTION

## Course Introduction

Individuals with experience and skills working with AWS services to develop, create, protect, and maintain analytics systems can pursue the AWS Certified Data Analytics - Specialty. This course walks you through the different concepts, including how to design analytical questions, then gather, analyze, and prepare data, while evaluating the data and uncovering insights from it.

## Recommended AWS Knowledge

The ideal applicant should have:

- 1) A minimum of five years of expertise with popular data analytics tools
- 2) Two years of hands-on experience and competence in designing, building, securing, and maintaining analytics systems using AWS services
- 3) The ability to create AWS data analytics services and comprehend how they interact
- 4) An understanding of how AWS data analytics services fit within the data lifecycle of collection, storage, processing, and visualization

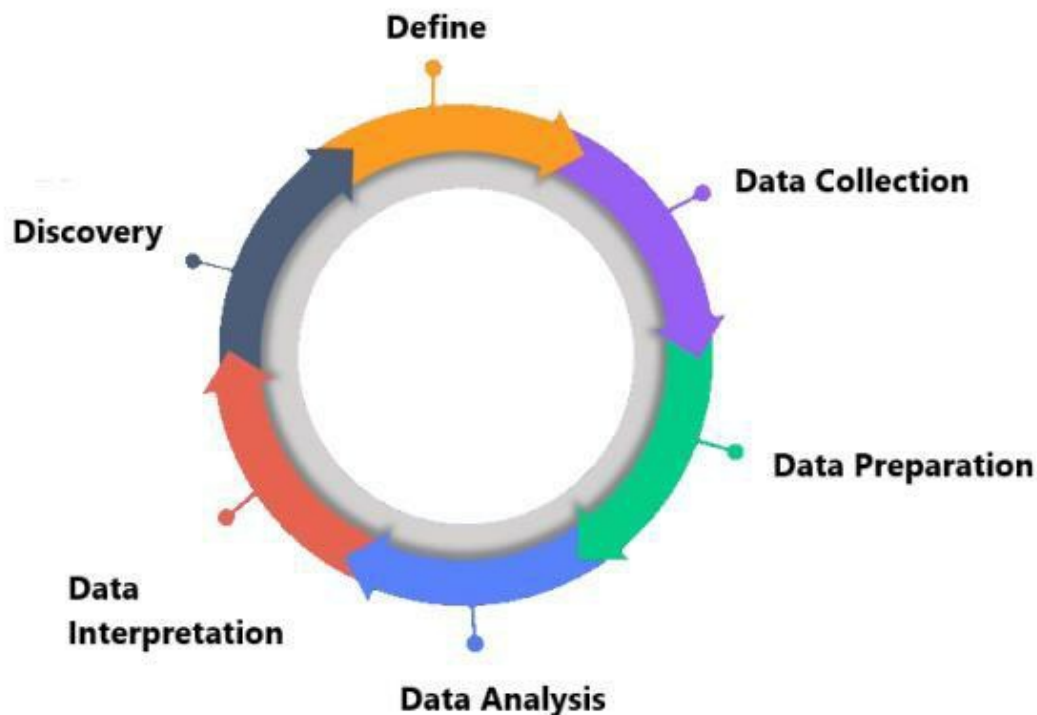
## What is Data Analytics?

Inspection, cleaning, converting, and modeling data to uncover usable information, informing conclusions, and assisting decision-making are what data analytics is all about. Simply said, data analytics implies that we have some data, and we need to answer questions with it. We have a

process that needs to be improved, and we can use the data we collect from our users in different locations and on different devices to answer questions and improve the processes that occur within our data collection, data processing, data analytics, and our understanding of what that data represents.

## Steps for Success

To understand the different steps that go into a data analytics pipeline, we need to understand the idea of the steps for a successful data analytics pipeline.



*Figure 1-01: Steps for Success*

We first need to define the questions we need to answer and the types of problems we are trying to solve. Then, we must define the types of insights we are looking for. Once we define those questions that need to be answered, we can then start the data collection process. Data can

come from a single source or many different sources. We need to find ways to build applications around collecting mass amounts of data from various sources and assembling it all in either a single source or a way to interact with a single source of data. The data that we collect can be streaming data, real-time data, historical data, or data that gets produced every so often. As we collect that data, we need to think about what type of preparations need to be done on that data before we store it off. After this, we will analyze that data, run queries on that data, build charts and graphs, and metrics from that data. Once we understand how to prepare our data, we can then start the data analysis process. This means running queries on the data, taking subsets, merges, or joins of the data, and starting the initial analysis to answer and improve the questions we defined at the beginning of our cycle. As we analyze that data, we start to have interpretations. Hence, we start to answer some of those questions. We might get some questions wrong; we might not have enough data; we might have to go back and collect more data, or we might have to prepare the data differently. But once we start to interpret the data, we can start to discover and figure out if our questions are being answered. If our process improves, we can refine and define more questions that need to be answered from that data. Therefore, it is a cycle that happens continually.

## **Digital Use Cases**

1. **Application Monitoring:** We can monitor our application performance, stability, and resource usage with data analytics to make our applications more efficient and cost-effective.
2. **Financial Analysis:** Financial analysis is another huge area for data analytics. We can gain useful insights from huge amounts of financial data.
3. **Machine Learning:** We can manage our large data sets, train

models, feed information from our machine learning application back into that same data analytics system to help refine those models and generally improve our machine learning systems through the use of data analytics tools.

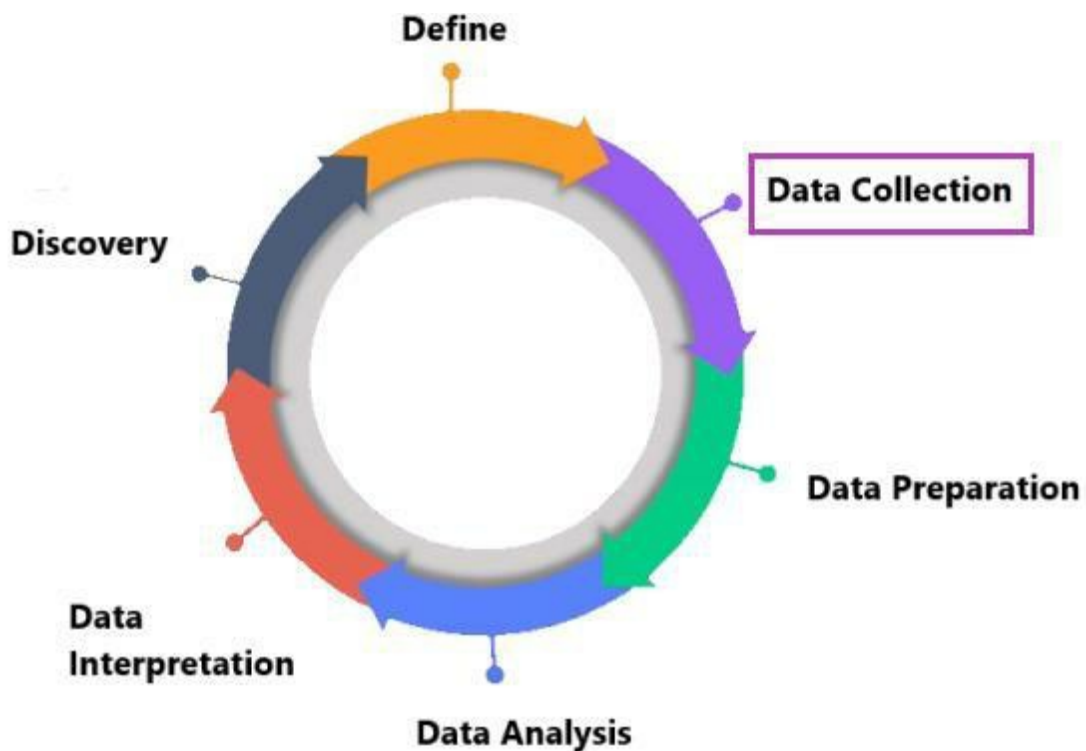
4. **IOT Management:** We can track, manage, and refine distributed networks of discrete devices using data analytics tools.



# CHAPTER 02: AMAZON SIMPLE STORAGE SERVICE

## Introduction to S3

In our data analytics steps for success, S3 will fall in data collection primarily. It has features that will use in data preparation, analysis, and data interpretation.



*Figure 2-01: Steps for Success*

Amazon S3 is a type of object storage that allows you to store and recover any quantity of data from any location. It is a low-cost storage solution with business resilience, reliability, efficiency, privacy, and infinite expansion. S3 uses an object called the bucket. The bucket is the atomic unit for S3.

Amazon S3 offers a straightforward web service interface for storing and retrieving any amount of data from any location at any time. You may quickly create projects that integrate cloud-native storage using this service. Because Amazon S3 is easily customizable and you only pay for what you use, you can start small and scale up as needed without sacrificing performance or dependability.

## Getting Data Into S3 - Concepts, AWS Management Console, AWS CLI

### Upload Interfaces

When we upload data at S3, we have several interfaces to work with. The AWS management console, the AWS CLI, and several AWS SDKs.



AWS SDKs

```
1 import boto3
2
3 someData = b'Hello Pinehead Gurus!'
4
5 s3Client = boto3.client("s3")
6
7 response = s3Client.put_object(
8     Bucket = "das-c01-files",
9     Body = someData,
10    Key = "Hello.txt"
11 )
12
13 print(response)
```

Figure 2-02: Interfaces

## **AWS Management Console**

When we use the management console, we use a graphical user interface. We can add files and folders. We can set most of the options to upload with this interface.

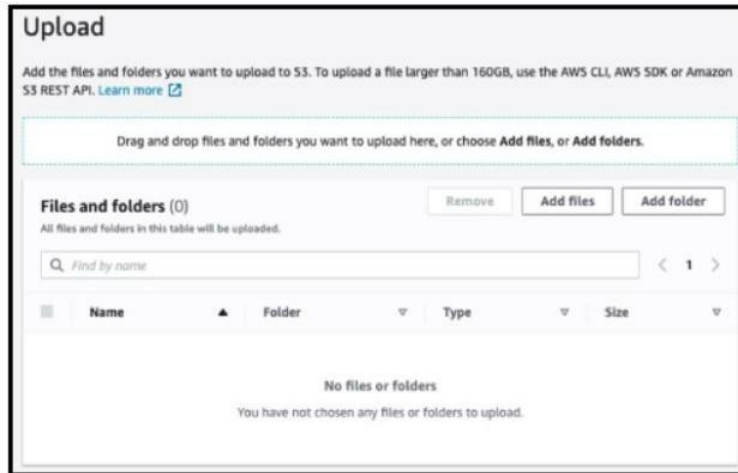


Figure 2-03: AWS Management Console

## **AWS CLI**

When using the AWS CLI, we enter commands in our terminal that will allow us to move data into the S3 bucket. We can use these commands to retrieve data from S3 buckets. Shown in the screenshot is the copy command. We also have the move command and a synchronized command.

```
[ec2-user@ip-172-31-17-217 ~]$ aws s3 cp album.csv s3://tatourney
upload: ./album.csv to s3://tatourney/album.csv
[ec2-user@ip-172-31-17-217 ~]$
```

| Commands           |
|--------------------|
| cp - Copy          |
| mv - Move          |
| sync - Synchronize |

*Figure 2-04: AWS CLI*

## **AWS SDKs**

Then comes the last interface, i.e., AWS SDKs. Shown in the screenshot is the Python SDK, which is called Boto3. When we use these SDKs, we will write code that will make API calls, which the other two interfaces do. But these are a little more direct. They are closer to the actual API calls. In the screenshot, we are looking at the put object API call. There is also a copy and copy object API call. These are now analogous commands in the Boto3 SDK. They do essentially the same thing, but they use different interfaces. Hence, the syntax is a little different. There is also the upload file and upload file object actions. These are very similar; they just use different interfaces. The code in this screenshot is using Boto3, which is the Python SDK. There are also official AWS SDKs for C ++, Go, Java, Javascript, .net, Node JS, PHP, and Ruby.

```

1  import boto3
2
3  someData = b'Hello Pinehead Gurus!'
4
5  s3Client = boto3.client("s3")
6
7  response = s3Client.put_object(
8      Bucket = "das-c01-files",
9      Body = someData,
10     Key = "Hello.txt"
11 )
12
13 print(response)

```

| API Calls      |
|----------------|
| put_object     |
| copy           |
| copy_object    |
| upload_file    |
| upload_fileobj |

*Figure 2-05: AWS SDKs*

## Transfer Acceleration

Another feature of S3 that we should be aware of when talking about getting data into S3 is Transfer Acceleration. To understand why we use Transfer Acceleration and how it works, let's look at a scenario. Assume we have an application. Our application will store data in a bucket in the us-east-1 region. And that works very well to send and receive data. Our users will add items to the bucket and read articles. As long as our users are properly close to the us-east-1 region, they will not have any issues. We will see some latency once we get out to the other geographic locations. As users get further and further away, that latency will increase. So, they may complain about how long it takes to get things from our S3 data store.

If our application user base keeps growing, we will run into issues as we have users worldwide.



Figure 2-06: Transfer Acceleration Scenario (a)

Therefore, to deal with it, the first thing we will likely implement is CloudFront, which will allow us to get data to users more quickly with a network of caching nodes. These caching nodes have an optimized network path between them. Hence, data goes from our Bucket into Cloud Front and our users. If an object is not cached in CloudFront, it can take a little while to get to our end user. But it is still going to use those optimized network paths. It significantly increases the usability of our application for global users.



Figure 2-07: Implementing CloudFront

But we still have an issue. That is, we need our users to get data into our

bucket. Hence, we have the same problem as before. We have solved the download side of the communication. However, we will suffer from increased latency as we increase geographic distance. The users across the ocean are going to have failed uploads likely and will be unhappy. Hence, this is where Transfer Acceleration comes in.



*Figure 2-08: Increased Latency*

Transfer Acceleration is enabled per bucket. It uses distinct endpoints. If you are hard writing endpoints, you will use **bucketname.S3-accelerate.amazonaws.com**. Or, if you are using IPv6, you would add a dual-stack between accelerate and Amazonaws.com, i.e., **bucketname.S3-accelerate.dualstack.amazonaws.com**. There are additional costs for using Transfer Acceleration. You will be charged 4 cents per gigabyte of data transferred from the United States to Europe and Japan edge locations and 8 cents per gigabyte for all other AWS edge locations.

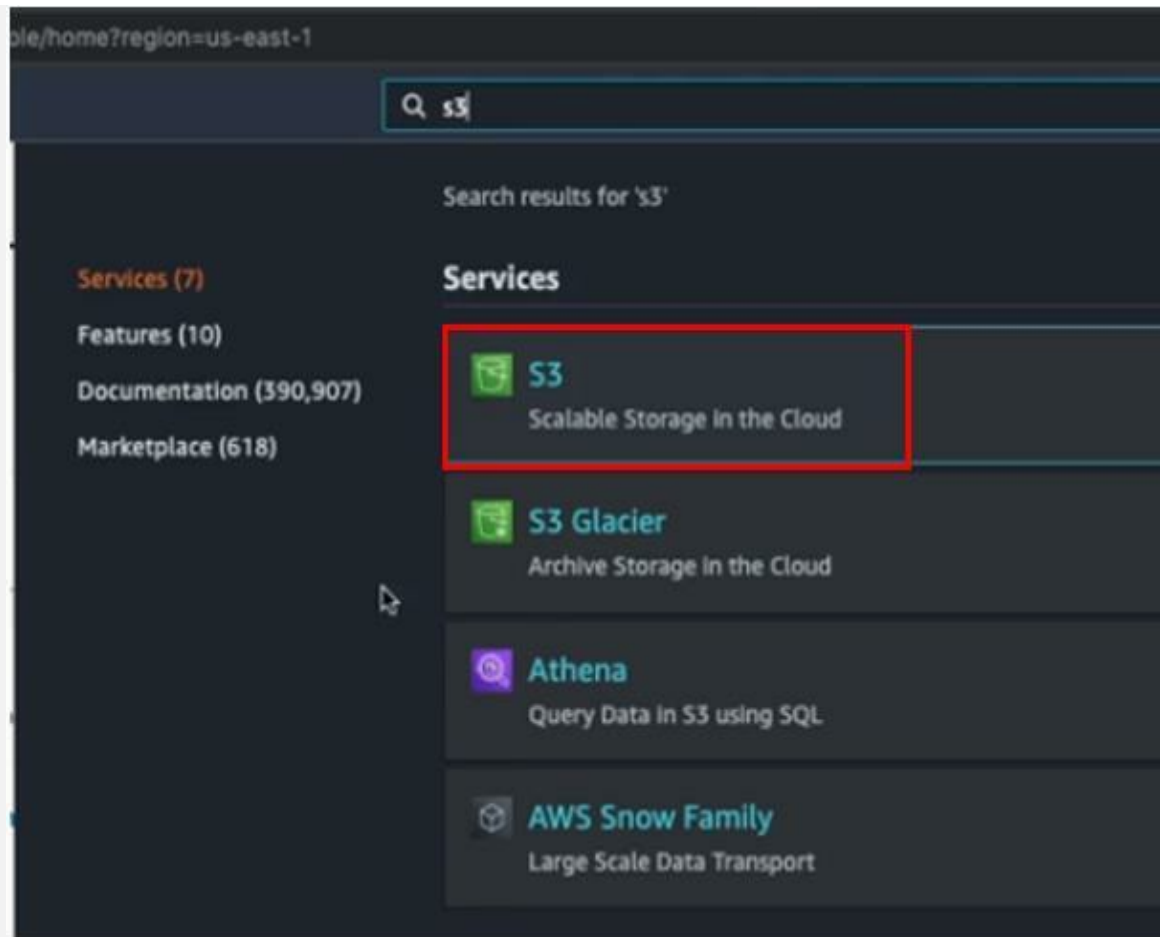
Transfer Acceleration leverages the edge locations to send data back to S3. Hence, this becomes a content ingestion network and not a distribution network. It means that our users use optimized network links to get our data or send data to us.



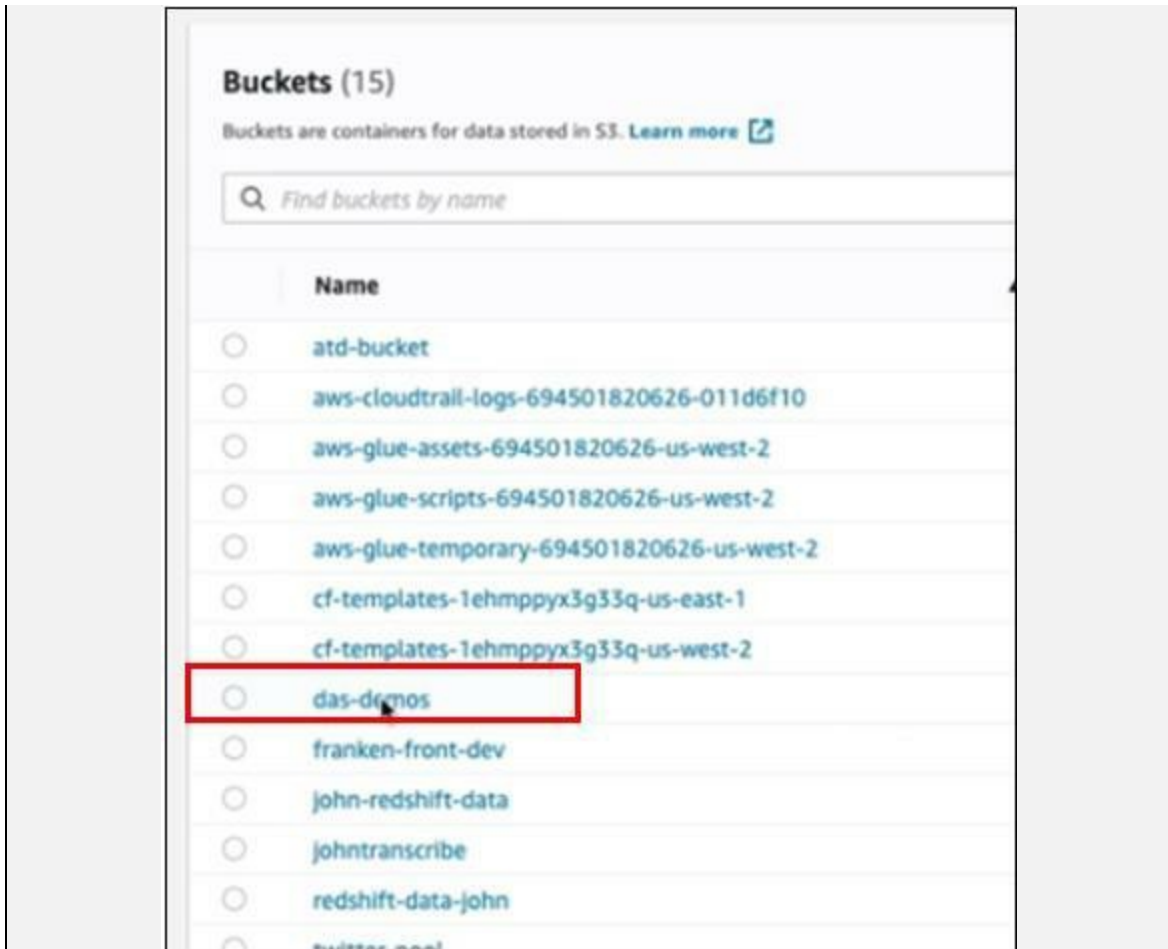
*Figure 2-09: Transfer Acceleration*

## Demo 2-01: AWS Management Console

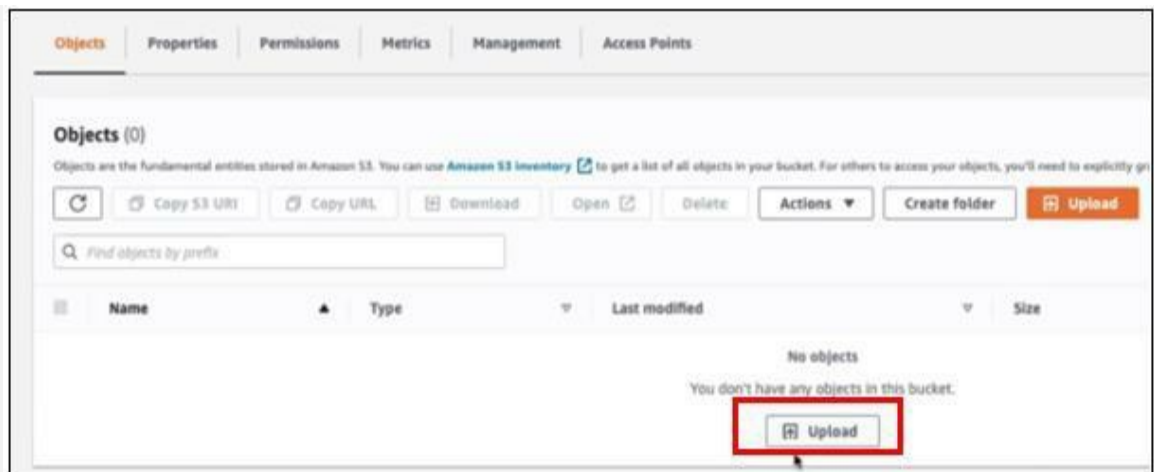
1. Log into AWS console.
2. Go to services and click on **S3**.



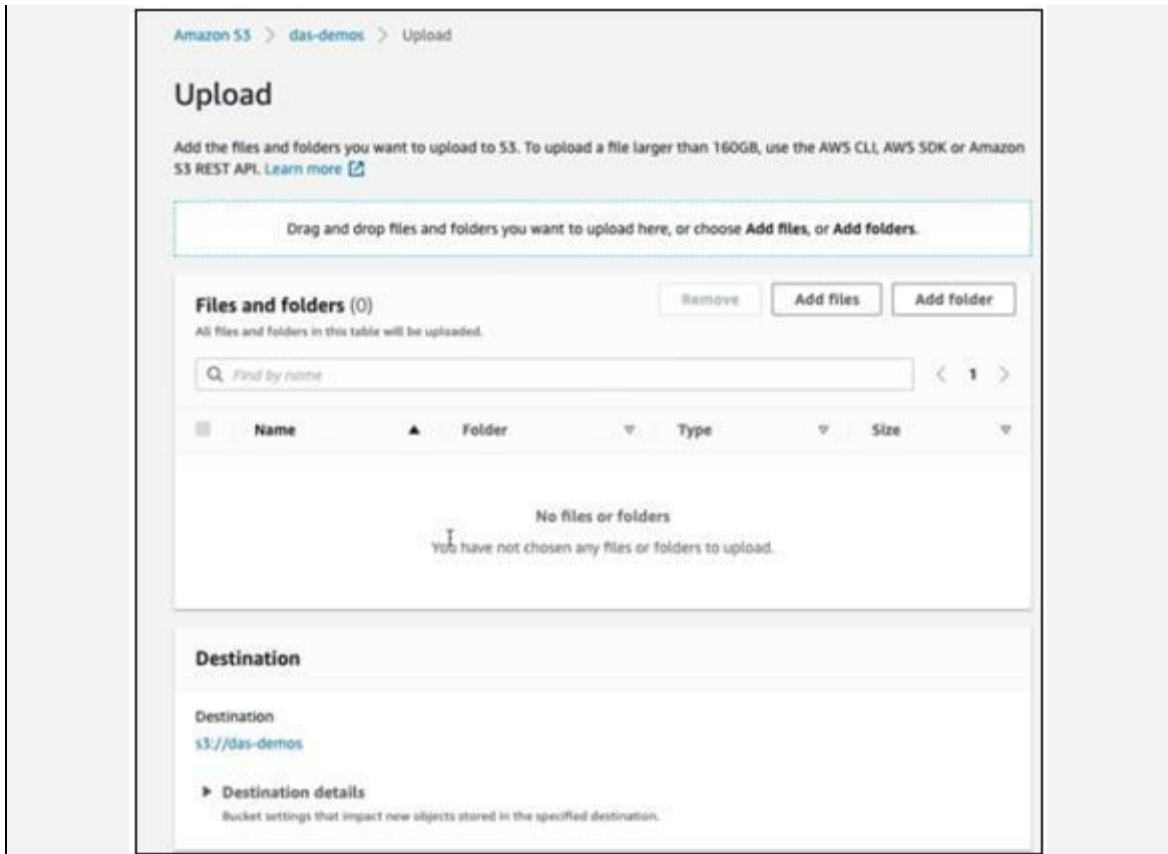
3. There are several buckets here. Click on **das-demo**.



4. Click on **Upload**.



5. The **graphical user interface** will be shown.



6. Set the **destination**.



7. Turn on **bucket versioning**.

**Destination**

Destination  
s3://das-demos

▼ Destination details  
Bucket settings that impact new objects stored in the specified destination.

**Bucket Versioning**  
When enabled, multiple variants of an object can be stored in the bucket to easily recover from unintended user actions and application failures. [Learn more](#)

Disabled

**Default encryption**  
When enabled, new objects stored in this bucket are automatically encrypted. [Learn more](#)

Disabled

**Object Lock**  
When enabled, objects in this bucket might be prevented from being deleted or overwritten for a fixed amount of time or indefinitely. [Learn more](#)

Disabled

⚠ We recommend that you enable Bucket Versioning to help protect against unintentionally overwriting or deleting objects. [Learn more](#)

**Enable Bucket Versioning**

- Now, set the **permissions**. You can use an access control list. A few predefined access controls lists are available, or you can write your access control list for this upload.

▼ **Permissions**  
Grant public access and access to other AWS accounts.

**Access control list (ACL)**  
Grant basic read/write permissions to other AWS accounts. [Learn more](#)

ⓘ AWS recommends using S3 bucket policies or IAM policies for access control. [Learn more](#)

**Access control list (ACL)**

☒ Choose from predefined ACLs

☐ Specify individual ACL permissions

**Predefined ACLs**

☒ **Private (recommended)**  
Only the object owner will have read and write access.

☐ Grant public-read access  
Anyone in the world will be able to access the specified objects. The object owner will have read and write access. [Learn more](#)

- Set the **properties**.



10. Set the **storage class**.

**Storage class**

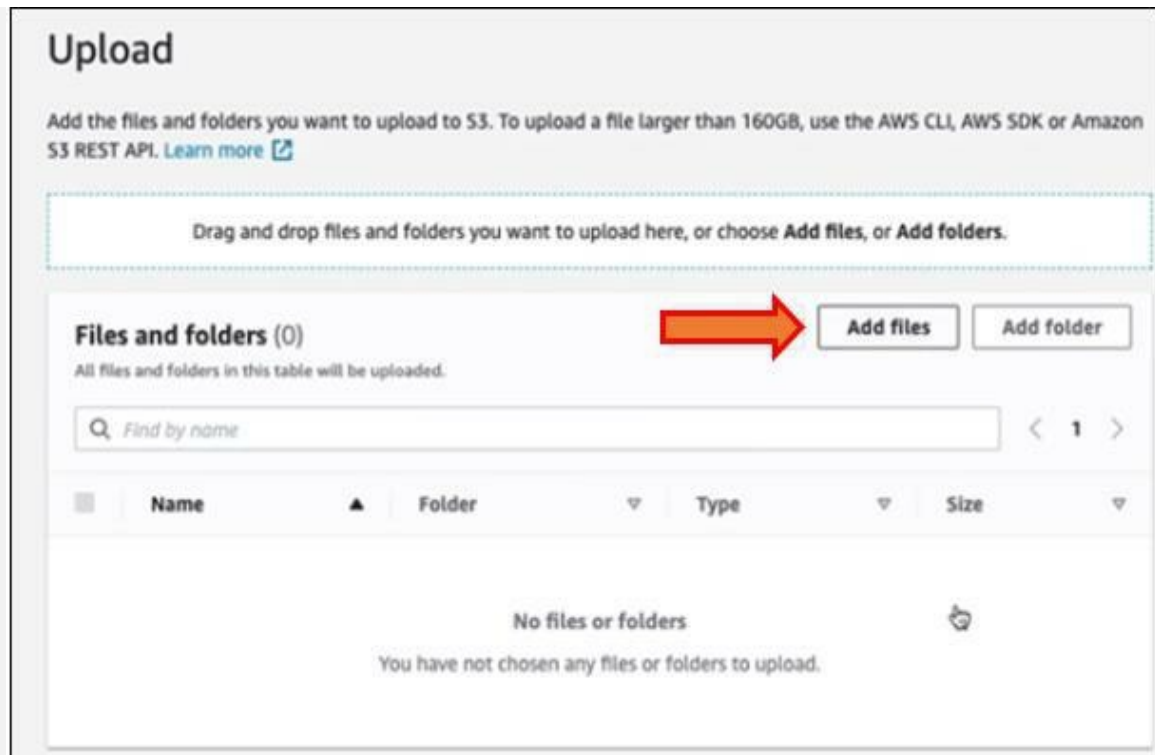
Amazon S3 offers a range of storage classes designed for different use cases. [Learn more](#) or see [Amazon S3 pricing](#)

|                                  | Storage class        | Designed for  | Availability Zones | Min storage duration |   |
|----------------------------------|----------------------|---|--------------------|----------------------|---|
| <input checked="" type="radio"/> | Standard             | Frequently accessed data  | ≥ 3                | -                    | - |
| <input type="radio"/>            | Intelligent-Tiering  | Long-lived data with changing or unknown access patterns                    | ≥ 3                | 30 days              | - |
| <input type="radio"/>            | Standard-IA          | Long-lived, infrequently accessed data                                      | ≥ 3                | 30 days              | 1 |
| <input type="radio"/>            | One Zone-IA          | Long-lived, infrequently accessed, non-critical data                        | 1                  | 30 days              | 1 |
| <input type="radio"/>            | Glacier              | Long-term data archiving with retrieval times ranging from minutes to hours | ≥ 3                | 90 days              | - |
| <input type="radio"/>            | Glacier Deep Archive | Long-term data archiving with retrieval times within 12 hours               | ≥ 3                | 180 days             | - |
| <input type="radio"/>            | Reduced redundancy   | Frequently accessed, non-critical data                                      | ≥ 3                | -                    | - |

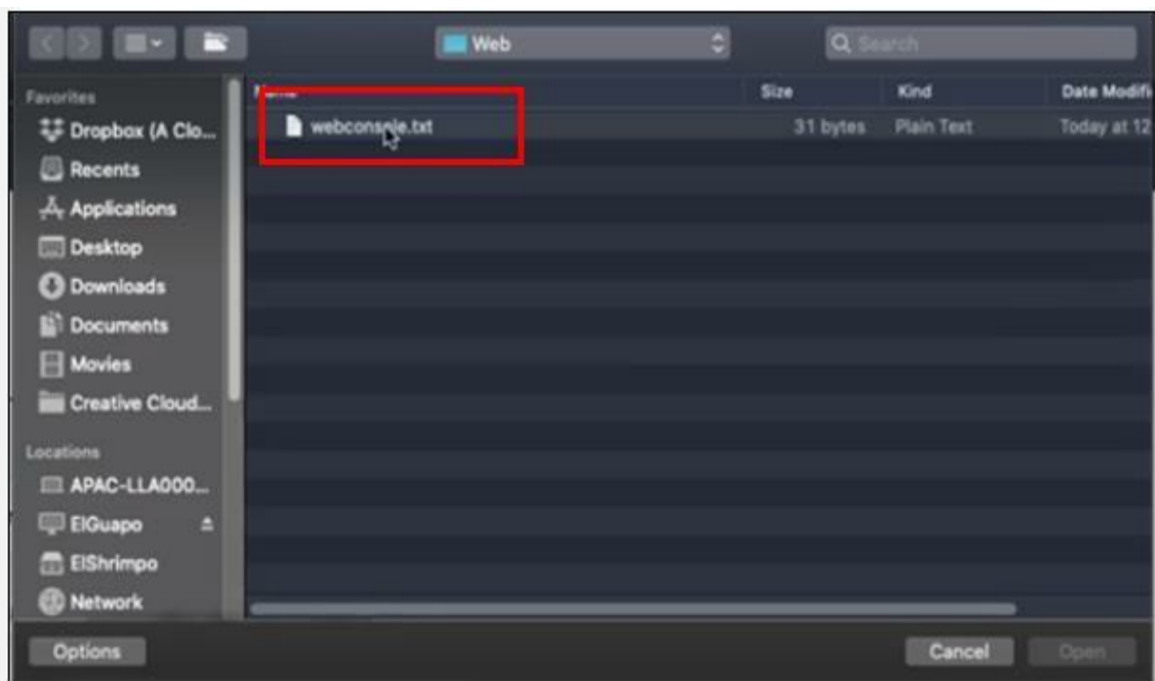
11. Here, we can decide if we want encryption turned on and which key we are going to use.

12. We can **add tags**, and we can add **additional metadata**.

13. To **upload** a file, click **Add Files**.



14. We have `web console.txt`.



15. The file is ready to go.

## Upload

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose **Add files**, or **Add folders**.

**Files and folders (1 Total, 31.0 B)**  
All files and folders in this table will be uploaded.

Remove

Add files

Add folder

< 1 >

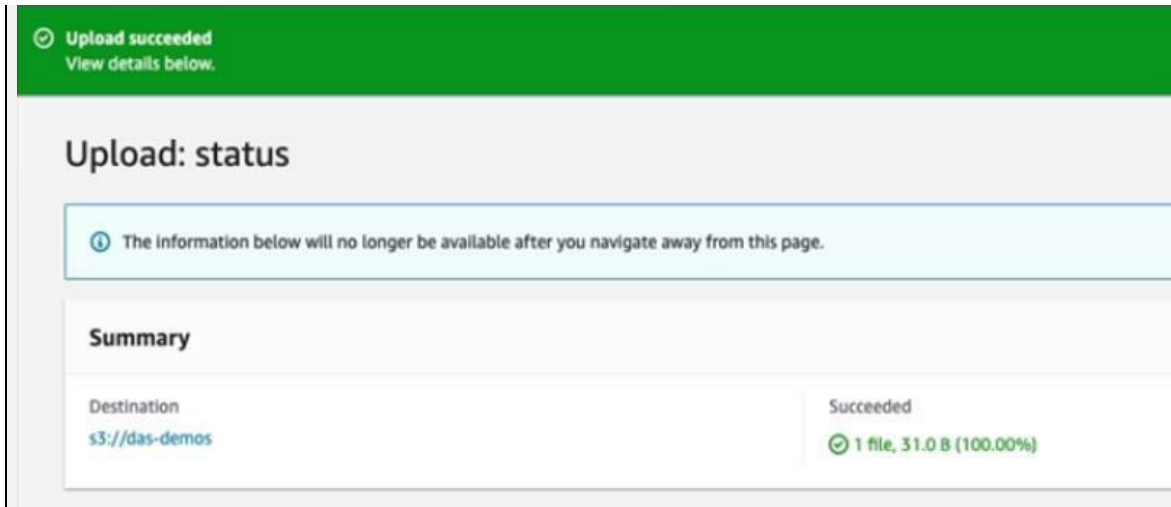
| <input type="checkbox"/> | Name           | Folder | Type       | Size   |
|--------------------------|----------------|--------|------------|--------|
| <input type="checkbox"/> | webconsole.txt |        | text/plain | 31.0 B |

16. Click **Upload**.

Cancel

**Upload**

17. You will see that the file has been uploaded.



## Demo 2-02: AWS CLI

1. Open the route of the folder.

```
> ls
boto3 cli  web
> cd ..
```

2. All of these files are available on the GitHub link given below.

[https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/tree/master/Amazon Simple Storage Service/Getting I](https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/tree/master/Amazon%20Simple%20Storage%20Service/Getting%20Started)

3. Move into the CLI directory and ls this folder. You can see a CLI.txt, a movemeCLI.txt, and a syncmeCLI.txt.

```
> ls
boto3 cli  web
> cd cli
> ls
cli.txt      movemeCLI.txt  syncmeCLI.txt
>
```

4. If you need to look at the content of these files, you can **cat** one out and take a look. It just says **hi, I came from the CLI** or **hi, I was moved from the CLI**, so forth.

```
> ls
boto3 cli  web
> cd cli
> ls
cli.txt      movemeCLI.txt syncmeCLI.txt
> cat cli.txt
Hi! I came from the CLI!%
> █
```

5. We will start with our copy command. We are first going to get the version. We are using **version 2**. We can enter AWS S3 because we are going to use an S3 command. We are going to **copy cli.txt to S3://das-demos**.

```
> aws --version
aws-cli/2.0.16 Python/3.7.4 Darwin/19.4.0 botocore/2.0.0dev20
> aws s3 cp cli.txt s3://das-demos █
```

6. The copy command is going to leave the file in the source and copy it to the destination. We can see that we have uploaded **cli.txt** to our **S3 bucket das-demos**. And it has the key **cli.txt**.

```
> aws --version
aws-cli/2.0.16 Python/3.7.4 Darwin/19.4.0 botocore/2.0.0dev20
> aws s3 cp cli.txt s3://das-demos
upload: ./cli.txt to s3://das-demos/cli.txt
> █
```

7. You also have the move command. It is beneficial to use spot instances as we know that our data will not necessarily be with

that instance. You cannot retain it with the move command on the source and only have it in the destination. You have the **movemeCLI.txt**.

```
> ls
cli.txt      movemeCLI.txt syncmeCLI.txt
> █
```

8. Once you run this command, you should see that the **movemeCLI.txt** is only in your S3 bucket. Confirm that your **cli.txt** made it to your bucket once you have moved the other files. Again, type **aws s3 mv, movemeCLI.txt** to **S3://das-demos**.

```
> ls
cli.txt      movemeCLI.txt syncmeCLI.txt
> aws s3 mv movemeCLI.txt s3://das-demos █
```

9. The files have been moved.

```
> ls
cli.txt      movemeCLI.txt syncmeCLI.txt
> aws s3 mv movemeCLI.txt s3://das-demos
move: ./movemeCLI.txt to s3://das-demos/movemeCLI.txt
> █
```

10. To confirm this, use **ls** the directory that you are in.

```
> ls
cli.txt          movemeCLI.txt  syncmeCLI.txt
> aws s3 mv movemeCLI.txt s3://das-demos
move: ./movemeCLI.txt to s3://das-demos/movemeCLI.txt
> ls
cli.txt          syncmeCLI.txt
> █
```

11. Now, we will use another S3 command, **AWS s3 ls das-demos**. S3 ls only functions for buckets and bucket prefixes. Therefore, we do not need to specify that it is an S3 bucket with the s3:// because it will not do anything for local directories.

```
> ls
cli.txt          movemeCLI.txt  syncmeCLI.txt
> aws s3 mv movemeCLI.txt s3://das-demos
move: ./movemeCLI.txt to s3://das-demos/movemeCLI.txt
> ls
cli.txt          syncmeCLI.txt
> aws s3 ls das-demos █
```

12. We will see the contents of our bucket. We have **cli.txt**, **movemeCLI.txt**, and the **web console.txt** that we added with the web console.

```
> ls
cli.txt          movemeCLI.txt  syncmeCLI.txt
> aws s3 mv movemeCLI.txt s3://das-demos
move: ./movemeCLI.txt to s3://das-demos/movemeCLI.txt
> ls
cli.txt          syncmeCLI.txt
> aws s3 ls das-demos
2020-05-27 12:44:26      24 cli.txt
2020-05-27 12:45:46      37 movemeCLI.txt
2020-05-27 12:40:42      31 webconsole.txt
> █
```

13. Lastly, the sync command will only move the files in our bucket. It also uses the last modified flag to detect if that file is changed

versus the one in the bucket. If the file has changed, it will upload it again so that the newest version is in our bucket.

14. To do this, type **aws s3 sync. s3://das-demo**.

15. We want to sync the current folder. Hence, we should only see our syncmeCLI.txt be uploaded from this command because the cli.txt is already in our bucket.

16. Hit **Enter**.

17. As you can see, we have moved our **DS\_store** as well.

```
> ls
cli.txt          syncmeCLI.txt
> aws s3 sync . s3://das-demos
upload: ./syncmeCLI.txt to s3://das-demos/syncmeCLI.txt
upload: ./DS_Store to s3://das-demos/DS_Store
> █
```

18. It will also move hidden files by default. Therefore, you need to be aware of that because the DAS store in this directory was moved with the sync.

19. Again, we can **ls** our bucket and confirm that our file has made it to our bucket.

```
> ls
cli.txt          syncmeCLI.txt
> aws s3 sync . s3://das-demos
upload: ./syncmeCLI.txt to s3://das-demos/syncmeCLI.txt
upload: ./DS_Store to s3://das-demos/DS_Store
> aws s3 ls das-demos
2020-05-27 12:47:50      6148 DS_Store
2020-05-27 12:44:26         24 cli.txt
2020-05-27 12:45:46         37 movemeCLI.txt
2020-05-27 12:47:50         49 syncmeCLI.txt
2020-05-27 12:40:42         31 webconsole.txt
> █
```

20. You can see the **syncmeCLI.txt** and the **DS\_store** have been moved to your bucket.

21. You can see several files that are not available in the current directory. To accomplish that, we are going to type **aws s3 sync S3://das-demos**.

```

> aws s3 ls das-demos
2020-05-27 12:47:50      6148 .DS_Store
2020-05-27 12:44:26       24 cli.txt
2020-05-27 12:45:46       37 movemeCLI.txt
2020-05-27 12:47:50       49 syncmeCLI.txt
2020-05-27 12:40:42       31 webconsole.txt
> ls
cli.txt      syncmeCLI.txt
> aws s3 sync s3://das-demos .

```

22. In addition, you can see the two files that you did not **download**. It is useful because you are not moving data that we already have.

```

> aws s3 ls das-demos
2020-05-27 12:47:50      6148 .DS_Store
2020-05-27 12:44:26       24 cli.txt
2020-05-27 12:45:46       37 movemeCLI.txt
2020-05-27 12:47:50       49 syncmeCLI.txt
2020-05-27 12:40:42       31 webconsole.txt
> ls
cli.txt      syncmeCLI.txt
> aws s3 sync s3://das-demos .
download: s3://das-demos/movemeCLI.txt to ./movemeCLI.txt
download: s3://das-demos/webconsole.txt to ./webconsole.txt
>

```

23. You can see that you have all your files in your bucket now.

```

> aws s3 ls das-demos
2020-05-27 12:47:50      6148 .DS_Store
2020-05-27 12:44:26       24 cli.txt
2020-05-27 12:45:46       37 movemeCLI.txt
2020-05-27 12:47:50       49 syncmeCLI.txt
2020-05-27 12:40:42       31 webconsole.txt
> ls
cli.txt      movemeCLI.txt  syncmeCLI.txt  webconsole.txt
> aws s3 sync s3://das-demos .
download: s3://das-demos/movemeCLI.txt to ./movemeCLI.txt
download: s3://das-demos/webconsole.txt to ./webconsole.txt
> ls
cli.txt      movemeCLI.txt  syncmeCLI.txt  webconsole.txt
>

```

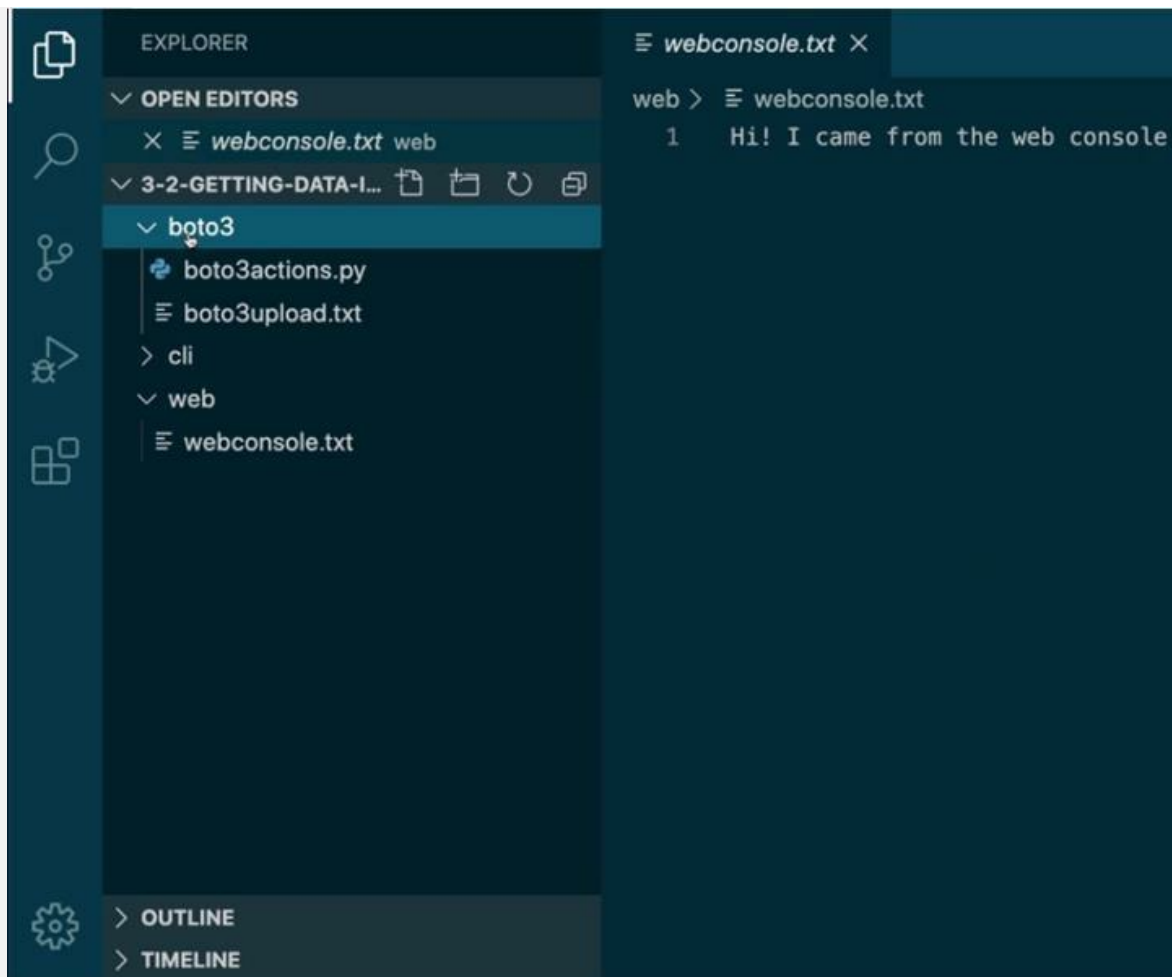
# Getting Data Into S3 - Boto3

## Demo: Python Boto3 SDK

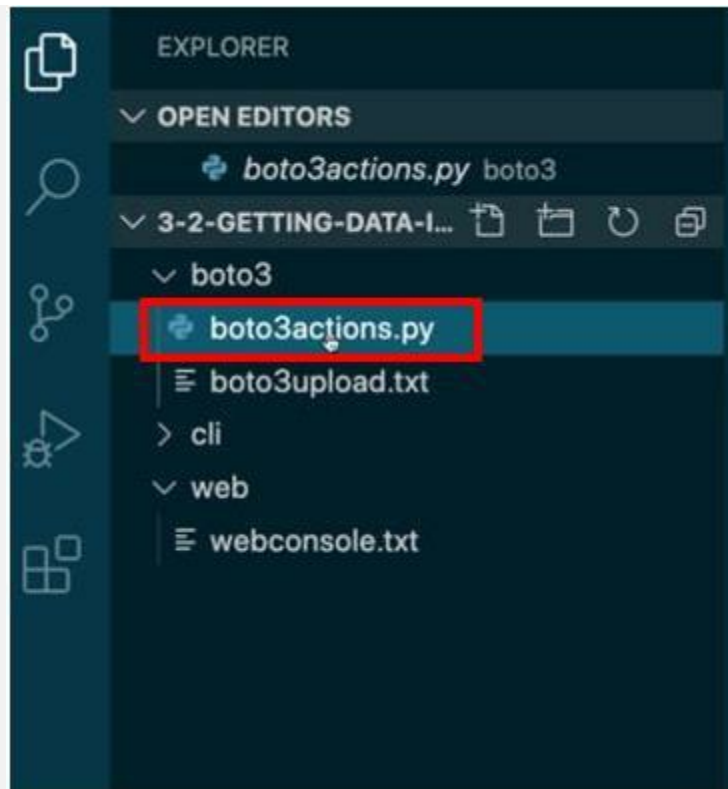
**Note:** The scripts used in the demo are available in the following GitHub link:

[https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/tree/master/Amazon Simple Storage Service/Getting Data](https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/tree/master/Amazon%20Simple%20Storage%20Service/Getting%20Data%20into%20S3)

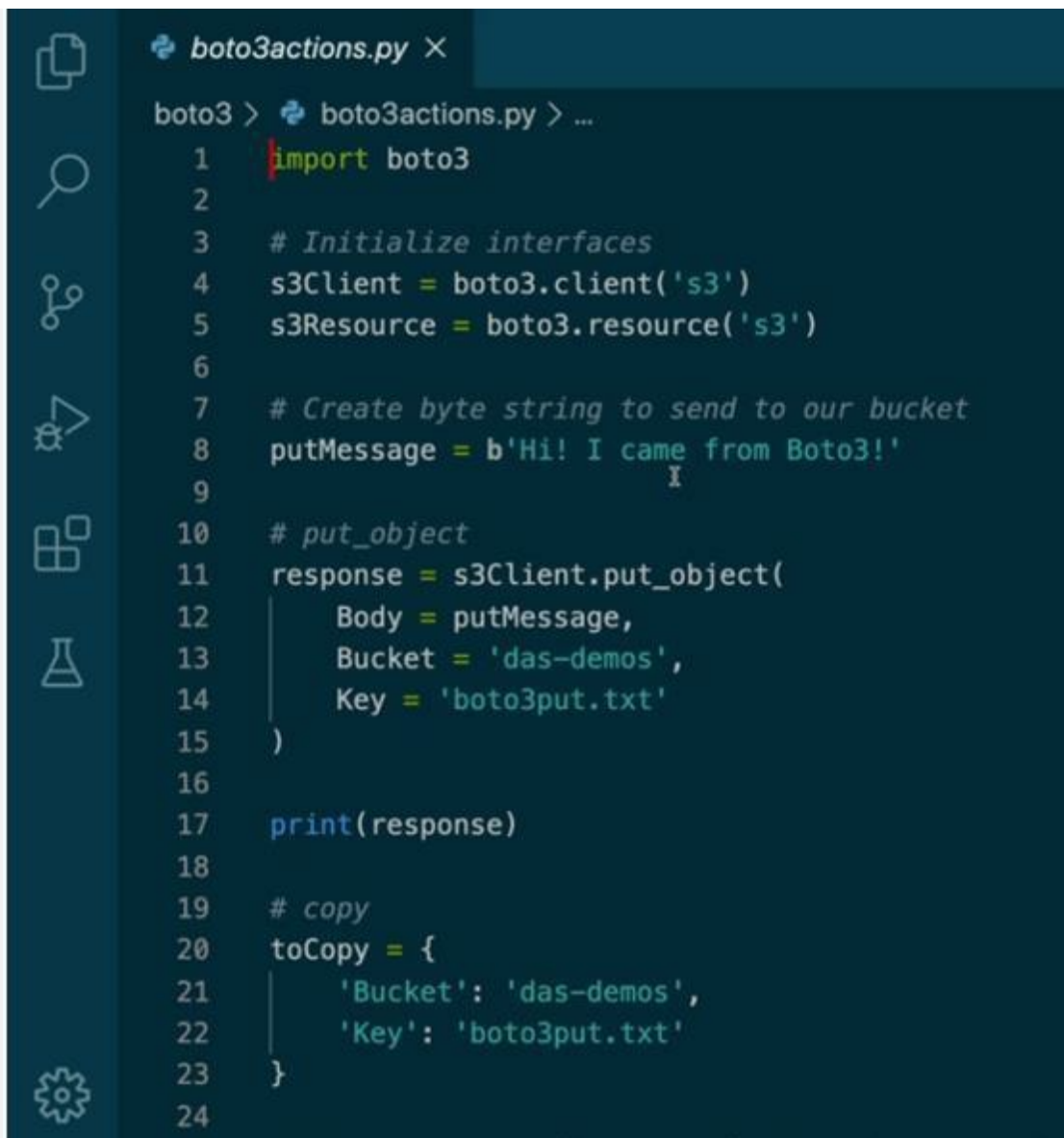
1. Click on the **Boto3** directory. Here, we have a **boto3upload.txt** and **boto3actions.py**.



2. Click on **boto3actions.py**.

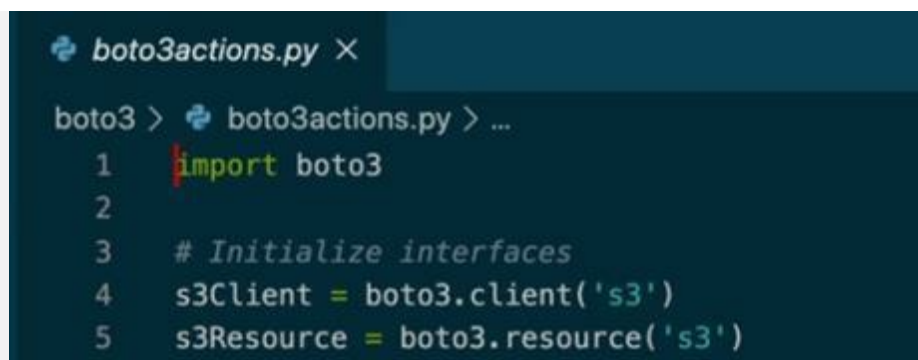


3. Here, we have a script prepared to perform our actions.



```
boto3actions.py X
boto3 > boto3actions.py > ...
1  import boto3
2
3  # Initialize interfaces
4  s3Client = boto3.client('s3')
5  s3Resource = boto3.resource('s3')
6
7  # Create byte string to send to our bucket
8  putMessage = b'Hi! I came from Boto3!'
9
10 # put_object
11 response = s3Client.put_object(
12     Body = putMessage,
13     Bucket = 'das-demos',
14     Key = 'boto3put.txt'
15 )
16
17 print(response)
18
19 # copy
20 toCopy = {
21     'Bucket': 'das-demos',
22     'Key': 'boto3put.txt'
23 }
24
```

4. First, we will **import Boto3** to our Python script, making it available to use in the script.



```
boto3actions.py X
boto3 > boto3actions.py > ...
1  import boto3
2
3  # Initialize interfaces
4  s3Client = boto3.client('s3')
5  s3Resource = boto3.resource('s3')
```

5. We are going to initialize our two interfaces. We have an **S3 client** variable that will be the reference to the **Boto3 client S3**. We also have an **S3 resource** variable, which refers to the **Boto3 resource for S3**.

```
3  # Initialize interfaces
4  s3Client = boto3.client('s3')
5  s3Resource = boto3.resource('s3')
```

6. We will use the **put object** to upload a binary object from our script to our S3 bucket.

7. We will capture the response of our API call in the **response** variable. We are going to use the **S3 client**. And we are going to **put\_object**. We need to provide a body of our object, which will be our byte string. And we need to specify a bucket, which is our **das-demos bucket**. And also, a **key** for our object, and then we will print the response to verify that there were no errors.

```
7  # Create byte string to send to our bucket
8  putMessage = b'Hi! I came from Boto3!'
9
10 # put_object
11 response = s3Client.put_object(
12     Body = putMessage,
13     Bucket = 'das-demos',
14     Key = 'boto3put.txt'
15 )
16
17 print(response)
```

8. Next, we have the **copy command**. We need to specify the source for our copy. Hence, we will set a bucket as the das-demos, and our Key will be **boto3put.txt**. It is the source, so we will copy the object we just uploaded with the **put\_object** call and send it to a

destination.

```
18
19 # copy
20 toCopy = {
21     'Bucket': 'das-demos',
22     'Key': 'boto3put.txt'
23 }
24
25 s3Resource.meta.client.copy(toCopy, 'das-demos', 'boto3copy.txt')
```

9. Next, there is the **copy\_object** operation, which uses the **S3 client**.

```
27 # copy_object
28 response = s3Client.copy_object(
29     Bucket = 'das-demos',
30     CopySource = '/das-demos/boto3put.txt',
31     Key = 'boto3copyobject.txt'
32 )
33
34 print(response)
35
```

10. Now, we are going to do the **upload\_file** operation. Here, we can see again; it is a resource operation, not a client operation. It is a meta operation because it will interact with the operating system and not anything else in our script.

```
35
36 # upload_file
37 boto3Upload = 'boto3upload.txt'
38
39 s3Resource.meta.client.upload_file(boto3Upload, 'das-demos', boto3Upload)
40
```

11. The next call is the client version of the **upload operation**.

```
41 # upload_fileobj
42 with open(boto3Upload, 'rb') as fileObj:
43     response = s3Client.upload_fileobj(fileObj, 'das-demos', 'boto3uploadobj.txt')
44     print(response)
45
```

12. Next part is commented out because we do not want to perform

this operation on our bucket. It can take 30 minutes to take effect, but this shows how you would turn on Transfer Acceleration for the bucket with the SDK.

We are going to use our S3 client. We are going to **put\_bucket\_accelerate\_configuration** to the **Bucket das-demos**. Our **AccelerateConfiguration** is Status **Enabled**. It can also be suspended to disable Transfer Acceleration.

```
'''
response = s3Client.put_bucket_accelerate_configuration(
    Bucket='das-demos',
    AccelerateConfiguration={
        'Status': 'Enabled'
    }
)

print(response)
'''
```

13. Go back to the terminal, and we will look at our operation. We need to make sure that we are in the correct folder.

```
> cd ../boto3
> █
```

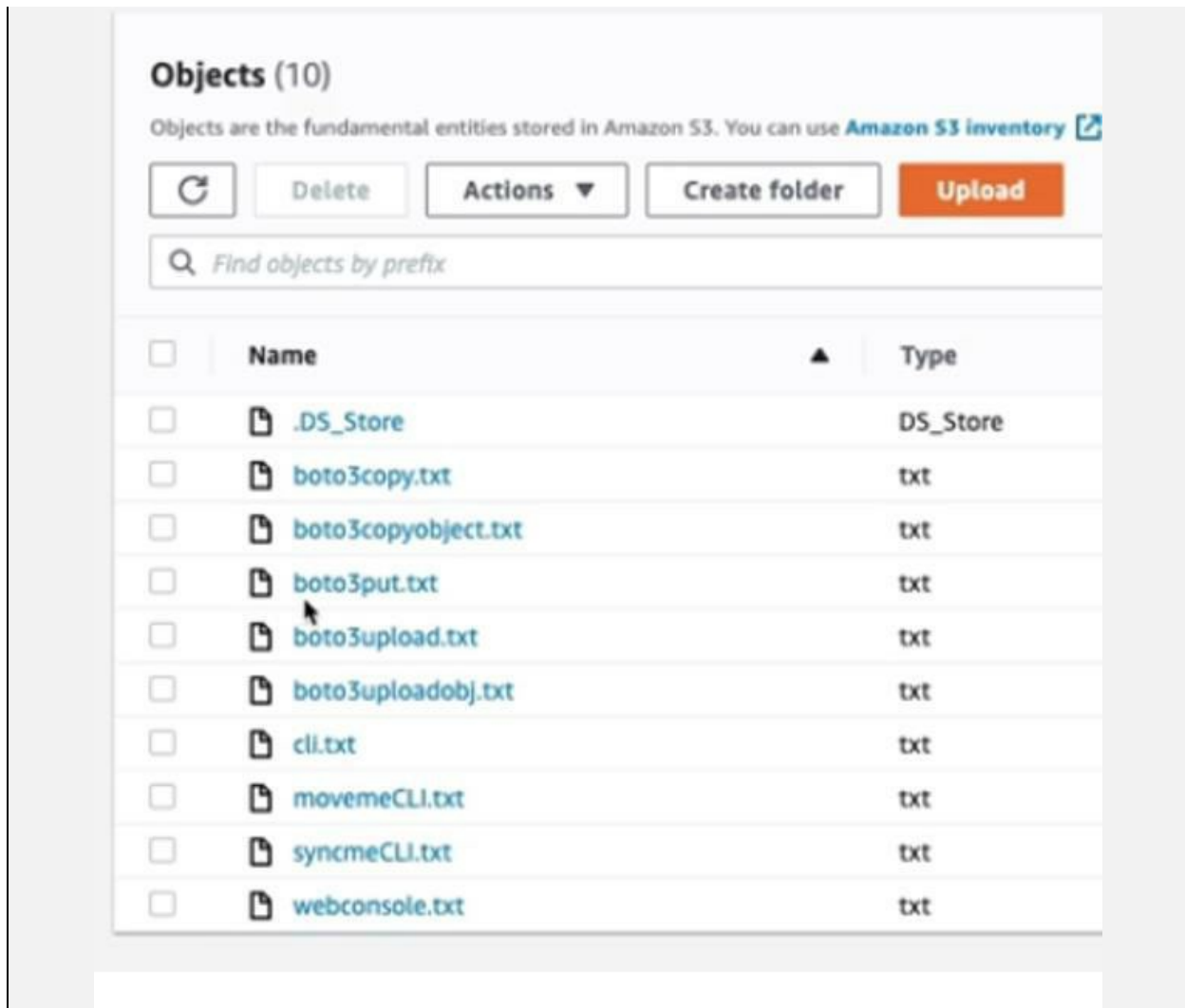
14. We will check our Python version. We are using Python 3.7.7.

```
> ls
boto3actions.py boto3upload.txt
> python --version
Python 3.7.7
> █
```

15. To run the script, type **python boto3actions.py** and hit enter. You will see the output of your operations here. We are looking for the 200's. Some of them do not return anything so that we will get a **None**.

```
> python boto3actions.py
{'ResponseMetadata': {'RequestId': 'BEBE4B1F2B030740', 'HostId': '9yuJIeuJgqazyT49YphrCjImqu/QmpC1+MEdvDYs+6qYopkvQEEqqAcB4xN32DfzZ3SIPYelMfU=', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amz-id-2': '9yuJIeuJgqazyT49YphrCjImqu/QmpC1+MEdvDYs+6qYopkvQEEqqAcB4xN32DfzZ3SIPYelMfU=', 'x-amz-request-id': 'BEBE4B1F2B030740', 'date': 'Wed, 27 May 2020 20:03:06 GMT', 'etag': '"a5e53aeb28dbd0ca45caa33cc25c1565"', 'content-length': '0', 'server': 'AmazonS3'}, 'RetryAttempts': 0}, 'ETag': '"a5e53aeb28dbd0ca45caa33cc25c1565"'}}
{'ResponseMetadata': {'RequestId': 'DFE967C13A552F6D', 'HostId': 'knEZmDqdpFd1LCgrA27p/mxYuGHu04IiekBUfMhjYvm42L5YMBAA9dlq8DF1n0m1EMSG38FCNkCg=', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amz-id-2': 'knEZmDqdpFd1LCgrA27p/mxYuGHu04IiekBUfMhjYvm42L5YMBAA9dlq8DF1n0m1EMSG38FCNkCg=', 'x-amz-request-id': 'DFE967C13A552F6D', 'date': 'Wed, 27 May 2020 20:03:07 GMT', 'content-type': 'application/xml', 'content-length': '234', 'server': 'AmazonS3'}, 'RetryAttempts': 0}, 'CopyObjectResult': {'ETag': '"a5e53aeb28dbd0ca45caa33cc25c1565"', 'LastModified': datetime.datetime(2020, 5, 27, 20, 3, 7, tzinfo=tzutc())}}
None
> █
```

16. Go back to the web console to confirm that the files are there. Refresh the page. We should see our **CLI** files, **DS\_Store** file, and **Boto3** files. We see **cli.txt**, **movemeCLI.txt**, **syncmeCLI.txt**, as well as the **webconsole.txt**. We also have our **boto3put.txt**, the **boto3copy.txt**, **boto3copyobject.txt**, and our **boto3upload.txt** files.



#### **KAM TIP:**

- When we upload data at S3, we have several interfaces to work with. The AWS management console, the AWS CLI, and several AWS SDKs.
- When we use the management console, we use a graphical user interface. We can add files and folders. We can set most of the options for upload with this interface.
- When using the AWS CLI, we enter commands in our terminal that will allow us to move data into our S3 bucket.
- Transfer Acceleration is enabled per bucket.

## S3 Multipart Upload

### What do we do when we have a lot of data?

If we think about a large file or object in the context of S3, we can think about it as a large building. Moving it all at once is impractical if we are moving a building. It can be time-consuming if we have to carry a completely big building all by itself, in one piece, it will be moved very slowly, and we are probably going to damage the structure, and it is potentially expensive. Hence, when we talk about moving a large object, we run into some same issues. It is kind of impractical. Depending on the compute resources involved, it will be time-consuming might be expensive. We might have a lot of idle CPU cycles because we are just moving a large file all at once. To get around that, multipart upload comes in. If we look at some of the limitations of the standard ways to get data into S3, a single S3 Put is only going to let us upload five gigabytes of data in a single Put, but an S3 object can be up to five tebibytes.

To get a five-tebibyte object into S3, we will use a multipart upload. It has three steps, like if we were going to move a building, we would probably break it down into components and have a plan and use that plan to reassemble those components on the other side of the move.

- **Prepare Data:** We will break the data into reasonably sized pieces.
- **Move Pieces:** We will perform the multipart upload steps to move all data to your S3 bucket.
- **S3 puts it together:** We will let S3 know the upload is complete, and S3 puts the data back together in the bucket.

## Three Multipart Upload API Calls

There are three API calls that we use to perform this process.

- 1) **Create Multipart Upload:** First, we have the **CreateMultipartUpload** API call. It returns Bucket, Key, and UploadID. The main thing we need from returning this API call is the UploadID. The multipart upload acts as a meta object that stores all of the information about our upload while it is happening. It is going to hold the information about all the parts.
- 2) **Upload Parts:** Next, we have the **UploadPart** API call. We need to provide Bucket, Key, Part Number, and Upload ID. It returns an ETag. It is very important because we need to deliver that when we do our final API call.
- 3) **Complete Multipart Upload:** Finally, we have the **CompleteMultipartUpload** API call. It returns Bucket, ETags, and Key. We need to provide Bucket, Key, Part Number, Upload ID, all Part Numbers, and ETags.

## Considerations

We have some considerations for this process.

- 1) **Parts:** Multipart uploads can be made up of up to 10,000 fragments.
- 2) **Overwrite:** Specifying the same part number as a previously uploaded part can be utilized to overwrite that part. We can overwrite the parts while the multipart upload is still in progress. Suppose something in your log file changes, or there is an update to a section of it. In that case, you can overwrite that part, that that section is in and have the latest version of the log file in your object when it uploads, or if one of the parts fails or has some corrupted data in it, you can write it again into the multipart upload. It will use whatever the latest data is for that piece of the upload when it is reassembled.
- 3) **Auto-Abort:** A bucket lifecycle policy that can be utilized to abort

multipart uploads after a specified time automatically. It prevents situations where a multipart upload gets started, something goes wrong with it, and the closing piece of it never goes through, or there is an error when the close is requested that upload will not work until the upload is completed or aborted. Generally, in a production system, it is good to have an auto abort configured for your bucket if you know that there will be a lot of multipart uploads going to that bucket.

## Best Practices and Limitations

There are also a few best practices and limitations that we need to think about.

- AWS recommends considering multipart upload for files more significant than a hundred megabytes. It means that you might want to use multipart upload for anything larger than a hundred megabytes (100 MiB).
- We need to consider the limitation that all parts must be at least five megabytes (5 MiB), except for the final part.
- When we put these together, parts should be between five and a hundred megabytes.

## Demo 2-03: Exploring Multipart Upload Script

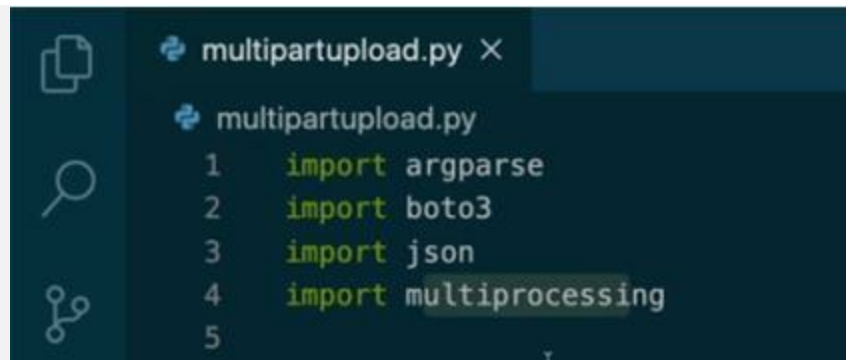
We will take a look at a script that is provided in the GitHub link given below.

[https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/tree/master/Amazon\\_Simple\\_Storage\\_Service/Multipart\\_Upl](https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/tree/master/Amazon_Simple_Storage_Service/Multipart_Upl)

It performs multipart uploads, and we will go through it, run it, and verify that our upload worked.

1. This script is written in Python; we use the Python SDK for AWS,

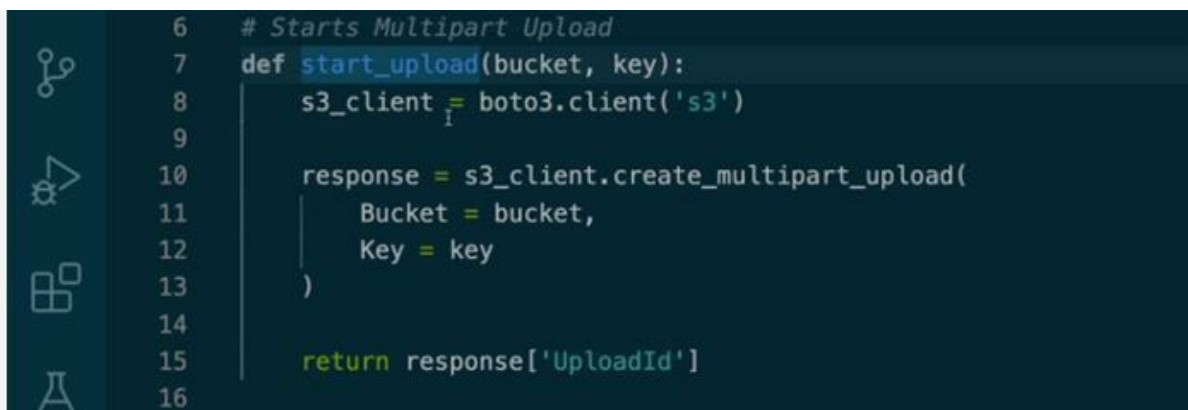
**Boto3**. It is imported here.



```
multipartupload.py ×
multipartupload.py
1  import argparse
2  import boto3
3  import json
4  import multiprocessing
5
```

2. We have three functions defined in our script that represent the three steps in our multipart upload. We have our **start multipart upload** piece. We have to **add the upload** or the add part and then the **end upload**. All three of these will initialize the Boto3 client for S3 and make a single call. Each one of those calls needs some different parameters passed into it.

3. To **create the multipart upload**, we must pass in a **bucket** and a **key** for our object. It will return, amongst other things, in the response and **Upload ID**. We need this for the rest of our calls.



```
6  # Starts Multipart Upload
7  def start_upload(bucket, key):
8      s3_client = boto3.client('s3')
9
10     response = s3_client.create_multipart_upload(
11         Bucket = bucket,
12         Key = key
13     )
14
15     return response['UploadId']
16
```

4. We then have the **add part** function. It uses a message queue to pass information between the parent process, which will be the foremost process for our script, and the child process, which will be the code within this function.

```

17 # Add upload part
18 def add_part(proc_queue, body, bucket, key, part_number, upload_id):
19     s3_client = boto3.client('s3')
20
21     response = s3_client.upload_part(
22         Body = body,
23         Bucket = bucket,
24         Key = key,
25         PartNumber = part_number,
26         UploadId = upload_id
27     )

```

5. Just so we know when this is finished, we will print a little message that says **Finished Part**, the **part number**, and then the **ETag** for that part.

```

28
29     print(f"Finished Part: {part_number}, ETag: {response['ETag']}")
30     proc_queue.put({'PartNumber': part_number, 'ETag': response['ETag']})
31     return
32

```

6. Finally, we have a function to **end our upload**. We provide that with a **bucket**, a **key**, the **upload ID**, and the **finished parts**, which will be the dictionaries that we will gather. Again, we have a fresh client. We will use the **complete multipart upload call**.

```

33 # End Multipart Upload
34 def end_upload(bucket, key, upload_id, finished_parts):
35     s3_client = boto3.client('s3')
36
37     response = s3_client.complete_multipart_upload(
38         Bucket = bucket,
39         Key = key,
40         MultipartUpload={
41             'Parts': finished_parts
42         },
43         UploadId = upload_id
44     )
45
46     return response

```

7. We have an **ArgumentParser**. We need to provide these, and a few of them are optional. We need to deliver a **file**. We do not have to provide a **key**. We need to provide a **bucket** and a **chunk size**.

```
# Primary logic
def main():
    ap = argparse.ArgumentParser()
    ap.add_argument('-f', '--file', required = True, help = "File to be chunked and uploaded")
    ap.add_argument('-k', '--key', help = "Key for destination object")
    ap.add_argument('-b', '--bucket', required = True, help = "Destination bucket")
    ap.add_argument('-cs', '--chunk_size', required = True, type = int, choices = range(5,101),
        metavar = '[5-100]', help = "Chunk size in MB, must be > 5MiB")
    ap.add_argument('-p', '--processes', type = int, choices = range(1,256), metavar = '[1-256]',
        default = 10, help = "Number of upload processes to run simultaneously")
    args = vars(ap.parse_args())
```

8. This ends up looking like a dictionary. We will create some variables that are more convenient than using these full objects and key accesses. The **if statement** says that if no key is provided, use the file name instead of the Key.

```
if args['key'] in [None, '']:
    args['key'] = args['file']

file = args['file']
key = args['key']
bucket = args['bucket']
sim_proc = args['processes']
upload_id = start_upload(bucket, key)
print(f'Starting upload: {upload_id}')
```

9. Then, we need to **open** our file. We have a list called **part\_procs** that is going to hold that. We are going to initialize a message queue, and then we need the messages that are returned. We will store those in a list called **queue\_returns**.

```
file_upload = open(file, 'rb')
part_procs = []
proc_queue = multiprocessing.Queue()
queue_returns = []
chunk_size = (args['chunk_size'] * 1024) * 1024
part_num = 1
chunk = file_upload.read(chunk_size)
```

10. Eventually, how this read operation works removes that data from the file when it is read. Hence, we can initialize a loop that says **while** the chunk size is **more significant than zero**, we will create a new process.

```
while len(chunk) > 0:
    proc = multiprocessing.Process(target=add_part, args=(proc_queue, chunk, bucket, key,
    part_num, upload_id))
    part_procs.append(proc)
    part_num += 1
    chunk = file_upload.read(chunk_size)
```

11. We will then use list comprehension to break our extensive list into a list of smaller lists that are the length of our simultaneous process variable.

```
part_procs = [part_procs[i * sim_proc:(i + 1) * sim_proc] for i in range((len(part_procs) +
(sim_proc - 1)) // sim_proc)]
```

12. This part of the script uploads all the parts.

```
for i in range(len(part_procs)):
    for p in part_procs[i]:
        p.start()

    for p in part_procs[i]:
        p.join()

    for p in part_procs[i]:
        queue_returns.append(proc_queue.get())
```

13. When we end our upload, the parts need to be in the correct order.. Hence, we need to **sort** the queue return list. We will use the **Key**, and the **Lambda II** lets us iterate through the values in our inventory.

These final couple of lines in the script say if the name of the process

is the primary, then run the main function. It prevents our child processes from running any of this code, so we do not end up with cascading spawning processes.

```
queue_returns = sorted(queue_returns, key = lambda i: i['PartNumber'])
response = end_upload(bucket, key, upload_id, queue_returns)
print(json.dumps(response, sort_keys=True, indent=4))

if __name__ == '__main__':
    main()
```

## Demo 2-04: Uploading a File with Multipart Upload Script

1. We will now go to the terminal in the folder where the script is and the file we will upload. The file we are uploading is a 1938 cowboy movie called **Where the Buffalo Roam**.

```
> ls
WheretheBuffaloRoam.mp4 multipartupload.py
> █
```

2. We will run our script here.

Type **python multipartupload.py -f WheretheBuffaloRoam.mp4 -b das-demos -cs 20**.

```
> ls
WheretheBuffaloRoam.mp4 multipartupload.py
> python multipartupload.py -f WheretheBuffaloRoam.mp4 -b das-demos -cs 20 █
```

3. Our upload has started.

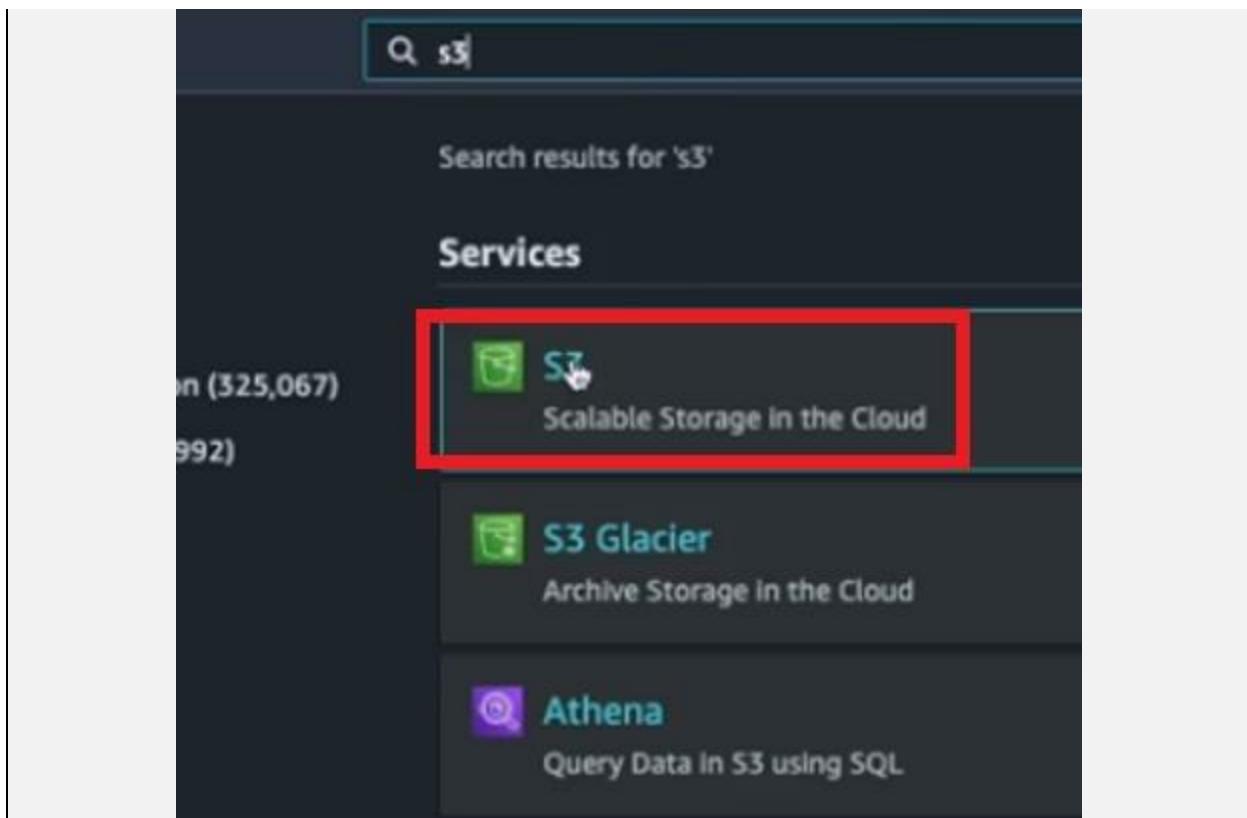
```
> ls
WheretheBuffaloRoam.mp4 multipartupload.py
> python multipartupload.py -f WheretheBuffaloRoam.mp4 -b das-demos -cs 20
Starting upload: ntEBzm648TtDUjdlTto_MEu598Z8Ggg7p5.0vGcVfq1auDl_f8.4.JqEqmhaJ77FskNZRDxmefseLl3mnoCrwvmv4nLmc1
bxu4rhdq.xQZg2wxWQvboMT8uBM_7aAs0
```

4. We can see that it took **two minutes and 50 seconds** to run our upload. We can see in our return that we have uploaded to the das-demos. We can see the **ETag** for our final object, **the Key** it ended up with, **the location**, and some metadata about the response. We can see in this that if we were logging this, we might want to keep track of, and capturing some metrics about the activity in our S3 buckets is the **transfer-encoding**. It tells us that it is uploaded with a multipart upload.

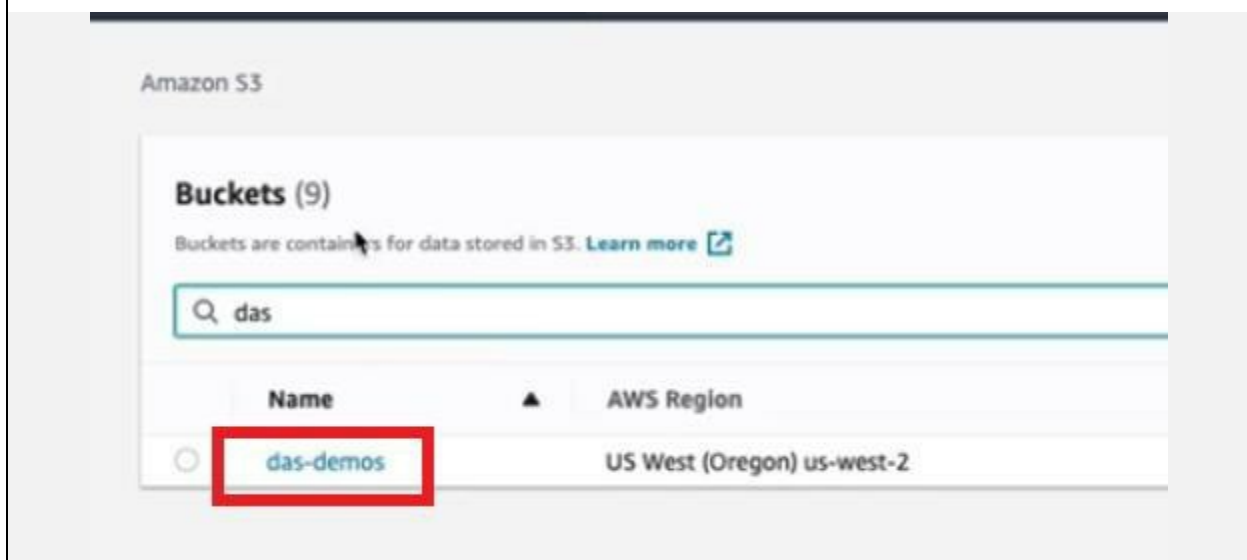
```
Finished Part: 17, ETag: "d281c9a26118038d1c4578a23e501806"
Finished Part: 18, ETag: "16abbc864459c4885afee3f83e76cc4d"
Finished Part: 12, ETag: "199d78ed76e2034f42cd946584c5cdaa"
Finished Part: 13, ETag: "0c87b4ffa93b6e754a4ff7508ebb55d5"
Finished Part: 15, ETag: "daae700858c83fe4f611535d307579a4"
Finished Part: 11, ETag: "d20054606f646ca9bb32369c7fcec0d3"
Finished Part: 14, ETag: "2ba06024365610e0a7c9b84f88906de2"
Finished Part: 16, ETag: "bb9d795d60e5049b827b6c25afb6f494"
{
  "Bucket": "das-demos",
  "ETag": "\"50684efb3cc689b39c4bbd86f85b8ba2-18\"",
  "Key": "WheretheBuffaloRoam.mp4",
  "Location": "https://das-demos.s3.us-west-2.amazonaws.com/WheretheBuffaloRoam.mp4",
  "ResponseMetadata": {
    "HTTPHeaders": {
      "content-type": "application/xml",
      "date": "Wed, 29 Jul 2020 22:13:53 GMT",
      "server": "AmazonS3",
      "transfer-encoding": "chunked",
      "x-amz-id-2": "3k6R5ywQjXhl+t4L5qNKqpTTT56ewmWiEGQ83Pb+7MtdLzH1ywS9vYFdLwWft00cAW9IRCZ7U5I=",
      "x-amz-request-id": "992850CA28C67040"
    },
    "HTTPStatusCode": 200,
    "HostId": "3k6R5ywQjXhl+t4L5qNKqpTTT56ewmWiEGQ83Pb+7MtdLzH1ywS9vYFdLwWft00cAW9IRCZ7U5I=",
    "RequestId": "992850CA28C67040",
    "RetryAttempts": 0
  }
}
```

2m 50s

5. Now, log into the AWS console. Click on services and click on **S3**.



6. We have a **das-demos** bucket in our account. Click on the **das-demos** bucket.



7. We have our file **WheretheBuffaloRoam.mp4**.

## das-demos

**Objects**

Properties

Permissions

Metrics

Management

Access Points

**Objects (1)**

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all of



Delete

Actions ▼

Create folder

Upload



Find objects by prefix



Name



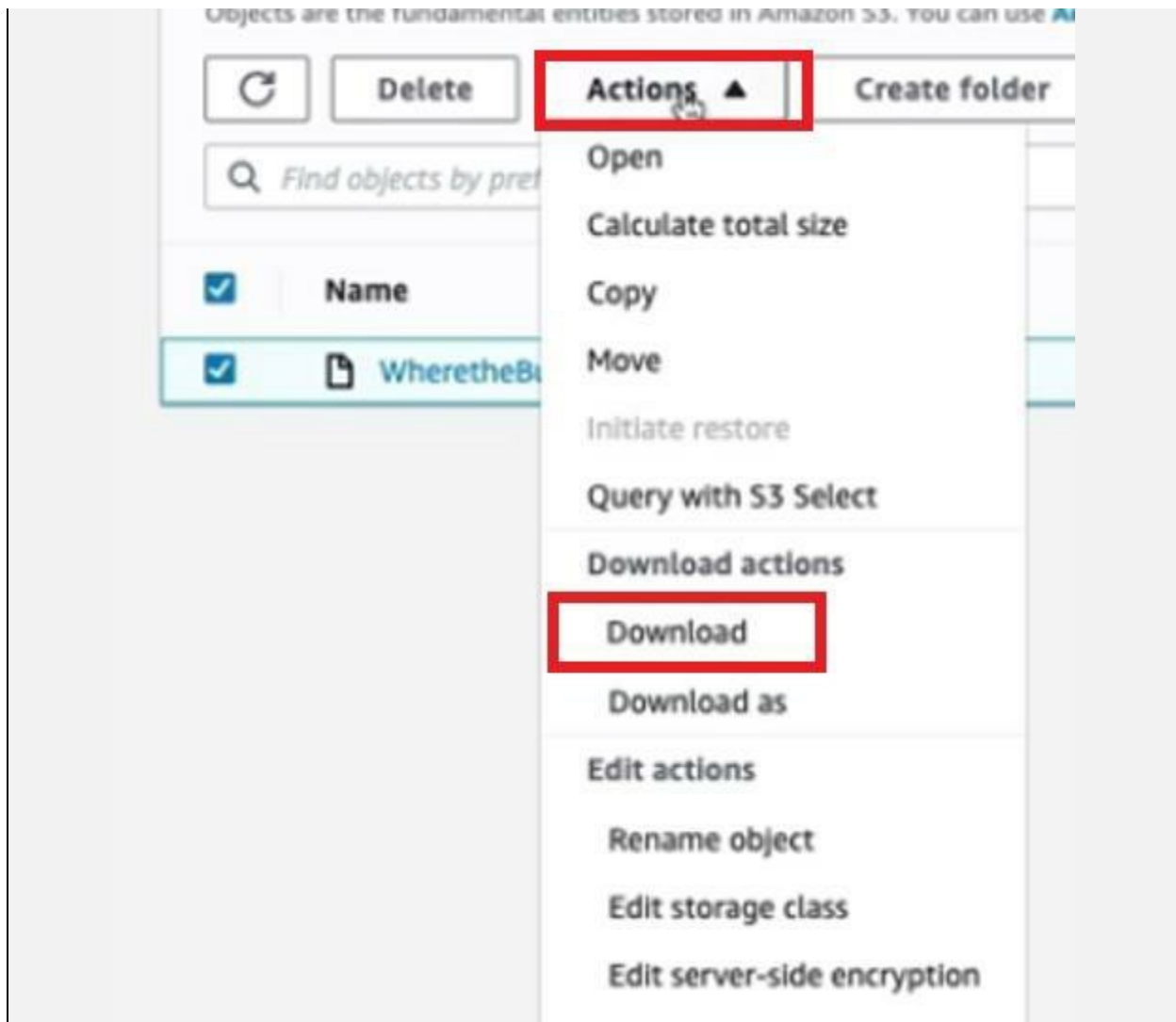
Type



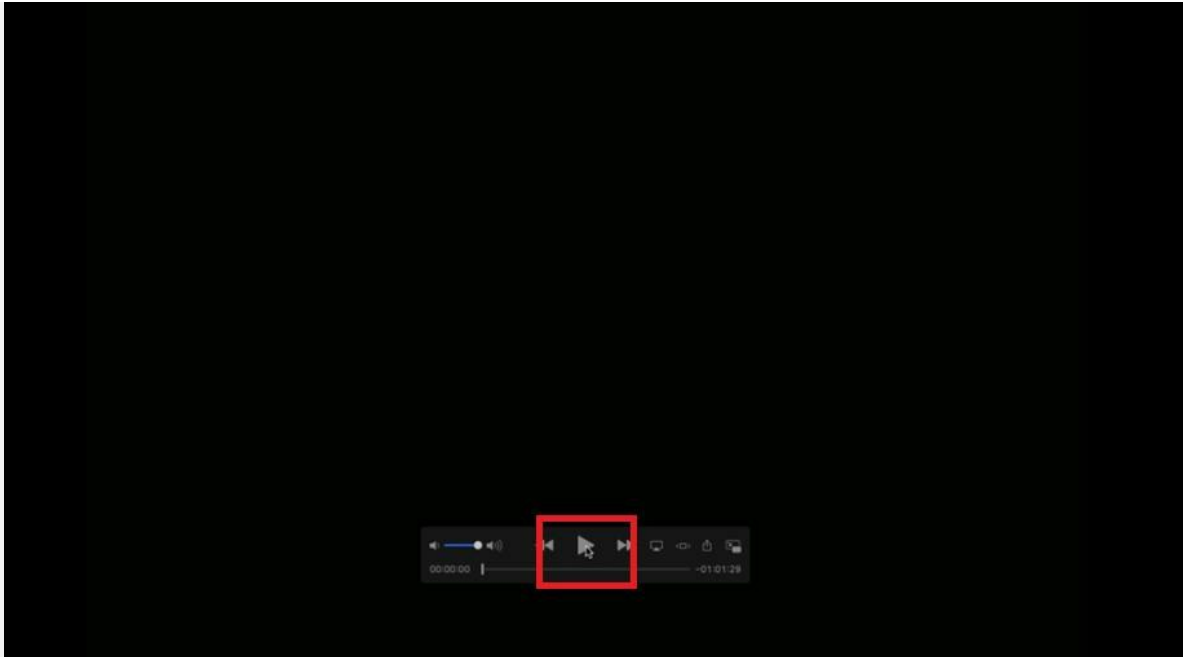
WheretheBuffaloRoam.mp4

mp4

8. Click on **Actions** and then click on **Download**.



9. Open the downloaded file and click on the **play** button.



10. The file has started playing.



**EXAM TIP:** AWS recommends multipart uploading files more

significant than a hundred megabytes. To get a five-tebibyte object into S3, we will use multipart upload. It has three steps: Prepare data, Move Pieces, and lastly, S3 puts it together.

## S3 Storage Classes

### What are Storage Classes?

Amazon S3 provides a variety of storage classes to satisfy a variety of use cases. Storage classes can be assigned to individual objects, or the bucket can be configured to use a specific storage class by default for anything added to it. These include:

- **S3 Standard** is a storage type for general-purpose storage of commonly accessed data.
- **S3 Intelligent-Tiering** is utilized for data with uncertain or changing access patterns. It is more of a management engine than a storage class itself.
- **Standard Infrequent Access** is used for infrequently accessed data.
- **One Zone Infrequent Access** is very similar to Standard Infrequent Access, except it can easily replace data. We will use this storage class for that data if we have an on-premise data store that we want to keep a copy of in the cloud for easier access.
- **The Glacier storage** class is used for data archives that we may need faster than the Glacier Deep Archive.
- **The Glacier Deep Archive** is also an archival storage class. It is utilized for digital and long-term archive preservation.

| Class               | Use Cases              |
|---------------------|------------------------|
| Standard            | General-purpose        |
| Intelligent-Tiering | Unknown access pattern |

|                            |  |
|----------------------------|--|
| Standard Infrequent Access | Weekly to monthly access                       |
| One Zone Infrequent Access | Easily replaced data                           |
| Glacier                    | Archive, acceptable minutes to hours retrieval |
| Glacier Deep Archive       | Archive, acceptable hours retrieval            |

*Table 2-01: Storage Classes and their Use Cases*

If we look at the storage cost for each of these storage classes, it is cheaper at the bottom of the table and more expensive as we move our way upward.

| Class                      | Use Cases                                      |
|----------------------------|--|
| Standard                   | General purpose                                |
| Intelligent- Tiering       | Unknown access pattern                         |
| Standard Infrequent Access | Weekly to monthly access                       |
| One Zone Infrequent Access | Easily replaced data                           |
| Glacier                    | Archive, acceptable minutes to hours retrieval |
| Glacier Deep Archive       | Archive, acceptable hours retrieval            |

*Figure 2-10: General Purpose Classes*

We have a general-purpose category. If we break these classes down into use cases and if we know our access pattern and will frequently access our data and store a bit of it, we will use either Standard or Intelligent-Tiering.

| Class                      | Use Cases                                      |
|----------------------------|--|
| Standard                   | General purpose                                |
| Intelligent- Tiering       | Unknown access pattern                         |
| Standard Infrequent Access | Weekly to monthly access                       |
| One Zone Infrequent Access | Easily replaced data                           |
| Glacier                    | Archive, acceptable minutes to hours retrieval |
| Glacier Deep Archive       | Archive, acceptable hours retrieval            |

*Figure 2-11: Infrequent Access Storage Classes*

Then, we have our Infrequent Access Storage Classes. It is the data that maybe we will access in a longer timeframe. Intelligent-Tiering is included in both of these because it bridges the gap between these 2.

| Class                      | Use Cases                                      |
|----------------------------|--|
| Standard                   | General purpose                                |
| Intelligent- Tiering       | Unknown access pattern                         |
| Standard Infrequent Access | Weekly to monthly access                       |
| One Zone Infrequent Access | Easily replaced data                           |
| Glacier                    | Archive, acceptable minutes to hours retrieval |
| Glacier Deep Archive       | Archive, acceptable hours retrieval            |

*Figure 2-12: Archival Storage Classes*

Then finally, we have Archival Storage Classes. It is for data that we are keeping or need to keep an original backup.

## Availability and Durability

S3 Standard, S3 Standard-IA, S3 Intelligent-Tiering, S3 One Zone-IA, S3 Glacier, and S3 Glacier Deep Archive are all meant to offer 99.999999999 percent (11 9's) data durability over a year. This level of

durability corresponds to a projected yearly loss of 0.000000001 percent of items. For example, if you store 10,000,000 objects on Amazon S3, you may anticipate a single object to be lost once every 10,000 years on average. On Outposts, S3 is meant to store data reliably and redundantly across several devices and servers. Furthermore, Amazon S3 Standard, S3 Standard-IA, S3 Glacier, and S3 Glacier Deep Archive are all built to keep data alive in the case of a complete S3 Availability Zone failure.

The S3 Standard storage class is designed for 99.99 percent availability, the S3 Standard-IA storage class and the S3 Intelligent-Tiering storage class for 99.9% availability, the S3 One Zone-IA storage class for 99.5 percent availability, and the S3 Glacier and S3 Glacier Deep Archive storage classes for 99.99 percent availability and a 99.9% Service Level Agreement (SLA).

## **S3 Standard**

S3 Standard provides excellent durability, availability, and performance object storage for frequently accessed data. S3 Standard is suitable for a wide range of use cases, including cloud services, dynamic websites, content distribution, mobile and gaming apps, and big data analytics, due to its low latency and high throughput. A single bucket can contain objects stored across S3 Standard, S3 Intelligent-Tiering, S3 Standard-IA, and S3 One Zone-IA; S3 Storage Classes can be defined at the object level. You may also utilize S3 Lifecycle policies to migrate items across storage classes without having to make any modifications to your application.

### ***Key Features***

- It is built for 99.999999999 percent object durability across

various Availability Zones.

- It is meant to have year-round availability of 99.99 percent.
- For availability, it is supported by the Amazon S3 Service Level Agreement.

## **S3 Infrequent Access**

**S3 Standard-IA** is for data accessed infrequently but has to be available quickly when needed. S3 Standard-IA combines S3 Standard's strong durability, speed, and low latency with a cheap per-GB storage and retrieval charge. S3 Standard-IA is appropriate for long-term storage, backups, and data storage for disaster recovery files because of its low cost and significant performance. A single bucket can contain objects stored across S3 Standard, S3 Intelligent-Tiering, S3 Standard-IA, S3 One Zone-IA, and S3 Storage Classes can be defined at the object level. You may also utilize S3 Lifecycle policies to migrate items across storage classes without having to make any modifications to your application.

### ***Key Features***

- It is built to keep 99.999999999 percent of items alive across various Availability Zones
- It is designed to be available 99.9% of the time throughout the year
- It is also backed by the Amazon S3 Service Level Agreement, which ensures that it is available when needed
- Additional Data Write Charge - \$0.01 per 1000 Request
- Additional Data Retrieval Charge - \$0.01 per GB
- Objects smaller than 128KB are billed as 128KB objects

**S3 One Zone-IA** is for data accessed infrequently but available quickly when needed. S3 One Zone-IA, unlike other S3 Storage Classes that store data in at least three Availability Zones (AZs), stores data in just

one AZ and costs 20% less than S3 Standard-IA. Customers that want a lower-cost alternative for seldom accessed data but don't need the availability and resilience of S3 Standard or S3 Standard-IA should use S3 One Zone-IA. It is a suitable option for storing off-site backup copies of on-premises data or data that can be created again. You may also use it to store data that has been replicated from another AWS Region using S3 Cross-Region Replication for a lower cost.

S3 One Zone-IA provides the same high durability, throughput, and latency as S3 Standard, but at a lower cost per GB storage and retrieval. A single bucket can contain objects stored across S3 Standard, S3 Intelligent-Tiering, S3 Standard-IA, S3 One Zone-IA, and S3 Storage Classes can be defined at the object level. You may also utilize S3 Lifecycle policies to migrate items across storage classes without having to make any modifications to your application.

### ***Key Features***

- It is built to last 99.99999999 percent of the time in a single Availability Zone.
- It is planned to be available 99.5 percent of the time throughout the year.
- For availability, it is backed by the Amazon S3 Service Level Agreement.

### **S3 Intelligent Tiering**

Independent of object size or retention duration, S3 Intelligent-Tiering is the best storage class for data with unknown, changing, or unexpected access patterns. S3 Intelligent-Tiering may be the default storage class for data lakes, analytics, and new applications.

S3 Intelligent-Tiering monitors our objects for access assigned to the Intelligent-Tiering storage class. If the object is not accessed for 30

days, it will move it to the configured Infrequent Access storage class. And once that object is accessed, it will be transferred back to the Standard storage class. It is where the unknown access pattern comes in. We will pay to monitor our objects, but we trade off the access cost of Infrequent Access for this monitoring cost. The monitoring is 1/4 of a penny per 1,000 objects, and the storage cost will depend on the storage class the object is in, in the underlying storage class. If it is in Standard or Infrequent Access, that will determine what we are paying to store those objects.

### ***Key Features***

- It is built to last 99.999999999 percent of the time in a single Availability Zone
- It is planned to be available 99.9 percent of the time throughout the year
- For availability, it is backed by the Amazon S3 Service Level Agreement

## **S3 Glacier**

S3 Glacier is a safe, long-lasting, low-cost data archiving storage type. You can store any quantity of data reliably at prices comparable to or lower than on-premises alternatives. S3 Glacier offers three retrieval options, ranging from a few minutes to hours, to keep costs reasonable while meeting various demands. You may utilize S3 Lifecycle policies to move data between the S3 Storage Classes for active data (S3 Standard, S3 Intelligent-Tiering, S3 Standard-IA, and S3 One Zone-IA) S3 Glacier, or you can upload objects directly to S3 Glacier.

### ***Key Features***

- It is built to keep 99.999999999 percent of items alive across various Availability Zones.

- If a whole Availability Zone is lost, data is unaffected.

## **S3 Glacier Deep Archive**

S3 Glacier Deep Archive is Amazon S3's cheapest storage tier, and it allows for long-term data retention and digital preservation for material that is only viewed once or twice a year. Clients need to keep data sets for 7-10 years or more to fulfill regulatory compliance requirements, such as those in highly regulated industries like financial services, healthcare, and government. S3 Glacier Deep Archive is a cost-effective and easy-to-manage replacement to magnetic tape systems, whether on-premises libraries or off-premises services, and may be used for backup and disaster recovery. S3 Glacier Deep Archive is a companion to Amazon S3 Glacier, appropriate for archives where data is often retrieved, and part of it is needed in minutes. All items in S3 Glacier Deep Archive are duplicated and stored across at least three geographically scattered Availability Zones are guaranteed to be 99.999999999 percent durable and may be recovered in less than 12 hours.

### ***Key Features***

- It is built to keep 99.999999999 percent of items alive across various Availability Zones
- The most affordable storage class, designed for long-term data retention of 7 to 10 years

**KAM TIP:** Storage classes provide ways to manage our performance, durability, cost, and data retrieval time.

- The S3 Standard storage class is a General-Purpose storage class.
- The S3 Infrequent Access is used for data that we will only access on the week-to-month scale for the most part.

- The S3 Intelligent-Tiering is a way to get the benefits of the Standard storage class and the Infrequent Access storage class for a small fee.
- The S3 Glacier is an Archival storage class minutes to hours of data retrieval time.
- In addition, the S3 Glacier Deep Archive is also an Archival storage class, but it has an hour's data retrieval time. It also has the lowest storage cost of all of our storage classes.

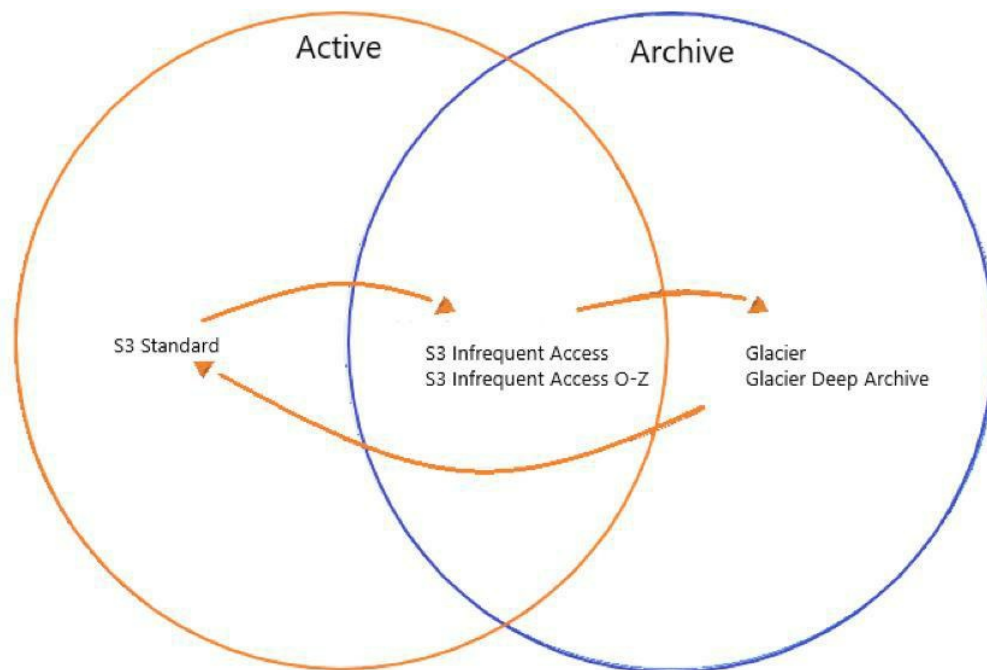
## **S3 Lifecycle Policies**

### **Life Of Data**

When we talk about the life of data, we mean that we will create our data, that data will be actively utilized, and then eventually, we will likely archive or delete that data. Once the data has been formed, it will be active in either the standard or infrequent access S3 storage class. And then, it will move over to the archive storage class. We may then push that data back into active utilization at some point. We can manage most of this with our lifecycle policy. We do not need to work potentially millions of objects stored in S3, and all of this will happen for us automatically.

### **Data Lifecycle in S3**

When we look at the storage classes, we overlap our Venn diagram of active and archive. Typically, we will bounce from the S3 standard to the infrequent access storage classes and then move to the archive. We might move back to standard again and repeat this process in the lifecycle of our data. It may have a loop around and around, or it may eventually be deleted after a certain amount of time.



*Figure 2-13: Data Lifecycle in S3*

## How To Make It Happen

It is the skeleton of a lifecycle policy. You can see that there are rules. We have a filter, a status, the transitions, the expiration for the procedure, and an ID for each one of our rules. We can write our lifecycle policies in XML or JSON.

```
{
  "Rules": [
    {
      "Filter": {},
      "Status": "Enabled",
      "Transitions": [],
      "Expiration": {},
      "ID": ""
    }
  ]
}
```

*Figure 2-14: Lifecycle Policy Format*

Let's assume a scenario to understand how to write a policy. We want only to manage our logs prefixes in our bucket. We want our data to be an S3 standard for one month, then move to S3 infrequent access for three months after that, and then move the data into Glacier Deep Archive for six years, and in eight months, it should be deleted.

To write this policy, we will filter on the prefix of logs. Filters can be a combination of tags and prefixes. If we wanted to tag data that moves through our bucket lifecycle at a different pace, we could use tags to accomplish that and have multiple policies.

```
{
  "Rules": [
    {
      "Filter": {
        "Prefix": "logs/"
      },
      "Status": "Enabled",
      "Transitions": [],
      "Expiration": {},
      "ID": ""
    }
  ]
}
```

*Figure 2-15: Writing Policy (i)*

After a month, we need to create a transition to move our data from S3 standard to S3 infrequent access. Hence, we will keep this data in the S3 standard for 30 days after it is created and then move it to standard rare access after those 30 days. We have various storage classes that we can select. We have normally reduced redundancy, which is no longer used because we have the infrequent access storage classes, standard infrequent access, one zone infrequent access, intelligent tiering,

glacier, and Glacier Deep Archive (referred to as deep archive in these policies).

```
{
  "Rules": [
    {
      "Filter": {
        "Prefix": "logs/"
      },
      "Status": "Enabled",
      "Transitions": [
        {
          "Days": 30,
          "StorageClass": "STANDARD_IA"
        }
      ],
      "Expiration": {},
      "ID": ""
    }
  ]
}
```

*Figure 2-16: Writing Policy (ii)*

We then need to keep the data in S3 infrequent access for three months. Hence, we will have days 120 and storage class deep archive because data will go after these 120 days. You will notice that 120 days is four months, and that is because these rules are not additive, and we need to include the time for the previous transitions in any subsequent changes.

```
{
  "Rules": [
    {
      "Filter": {
        "Prefix": "logs/"
      },
      "Status": "Enabled",
      "Transitions": [
        {
          "Days": 30,
          "StorageClass": "STANDARD_IA"
        },
        {
          "Days": 120,
          "StorageClass": "DEEP_ARCHIVE"
        }
      ],
      "Expiration": {},
      "ID": ""
    }
  ]
}
```

*Figure 2-17: Writing Policy (iii)*

We then set the expiration for the data with the expiration attribute. Here, it says that delete it after the data is 2,555 days old.

```

{
  "Rules": [
    {
      "Filter": {
        "Prefix": "logs/"
      },
      "Status": "Enabled",
      "Transitions": [
        {
          "Days": 30,
          "StorageClass": "STANDARD_IA"
        },
        {
          "Days": 120,
          "StorageClass": "DEEP_ARCHIVE"
        }
      ],
      "Expiration": {
        "Days": 2555
      },
      "ID": "ApplicationLogArchiving"
    }
  ]
}

```

*Figure 2-18: Writing Policy (iv)*

You need to provide an ID. It can be any alphanumeric string. We are going to call this ApplicationLogArchiving.

```
{
  "Rules": [
    {
      "Filter": {
        "Prefix": "logs/"
      },
      "Status": "Enabled",
      "Transitions": [
        {
          "Days": 30,
          "StorageClass": "STANDARD_IA"
        },
        {
          "Days": 120,
          "StorageClass": "DEEP_ARCHIVE"
        }
      ],
      "Expiration": {
        "Days": 2555
      },
      "ID": "ApplicationLogArchiving"
    }
  ]
}
```

*Figure 2-19: Writing Policy (v)*

## Demo 2-05: Applying S3 Lifecycle Policy

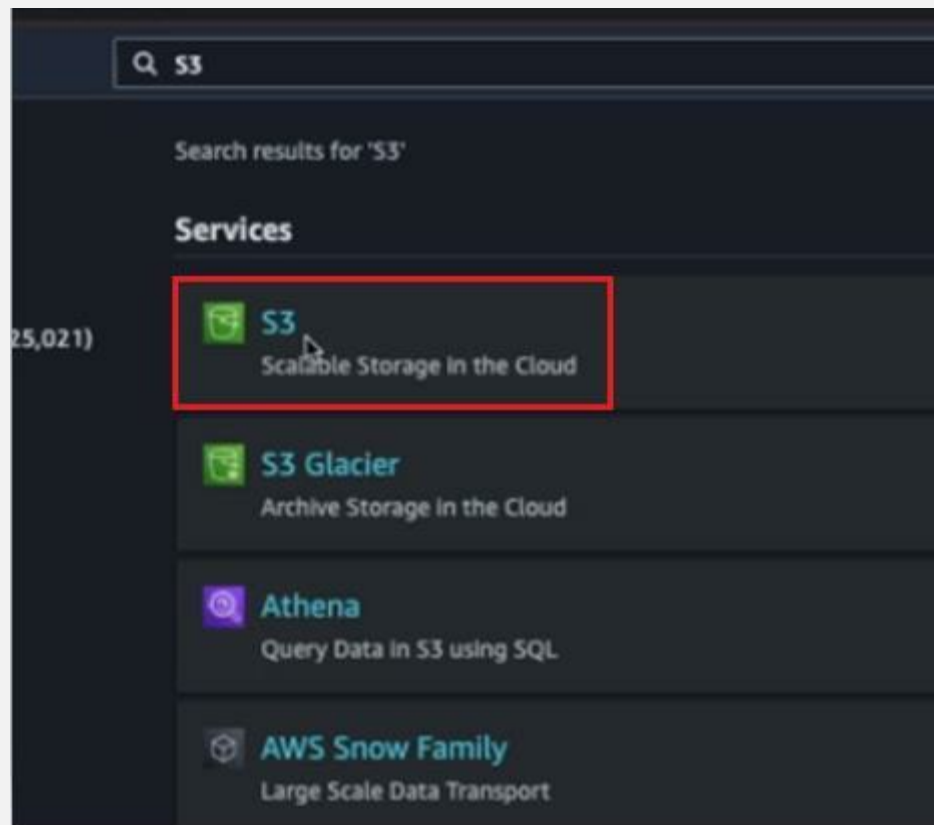
1) We will apply the following S3 lifecycle policy to our bucket. It is called the **ArchivePolicy.json**.

```
{ } ArchivePolicy.json ×
{ } ArchivePolicy.json > [ ] Rules > { } 0 > [ ] Transitions > { } 0 > StorageClass
1 {
2   "Rules": [
3     {
4       "Filter": {
5         "Prefix": "logs/"
6       },
7       "Status": "Enabled",
8       "Transitions": [
9         {
10          "Days": 30,
11          "StorageClass": "STANDARD_IA"
12        },
13        {
14          "Days": 120,
15          "StorageClass": "DEEP_ARCHIVE"
16        }
17      ],
18      "Expiration": {
19        "Days": 2555
20      },
21      "ID": "ApplicationLogArchiving"
22    }
23  ]
24 }
25
```

2) The easiest way to apply this policy to the bucket is with the **AWS CLI**.

```
> aws s3api put-bucket-lifecycle-configuration --bucket das-demos --lifecycle-configuration file:///ArchivePolicy.json
```

3) Log into the AWS console. Click on services and click on **S3**.



4) Here, we can see the **das-demos bucket**. Click on it.

## Amazon S3

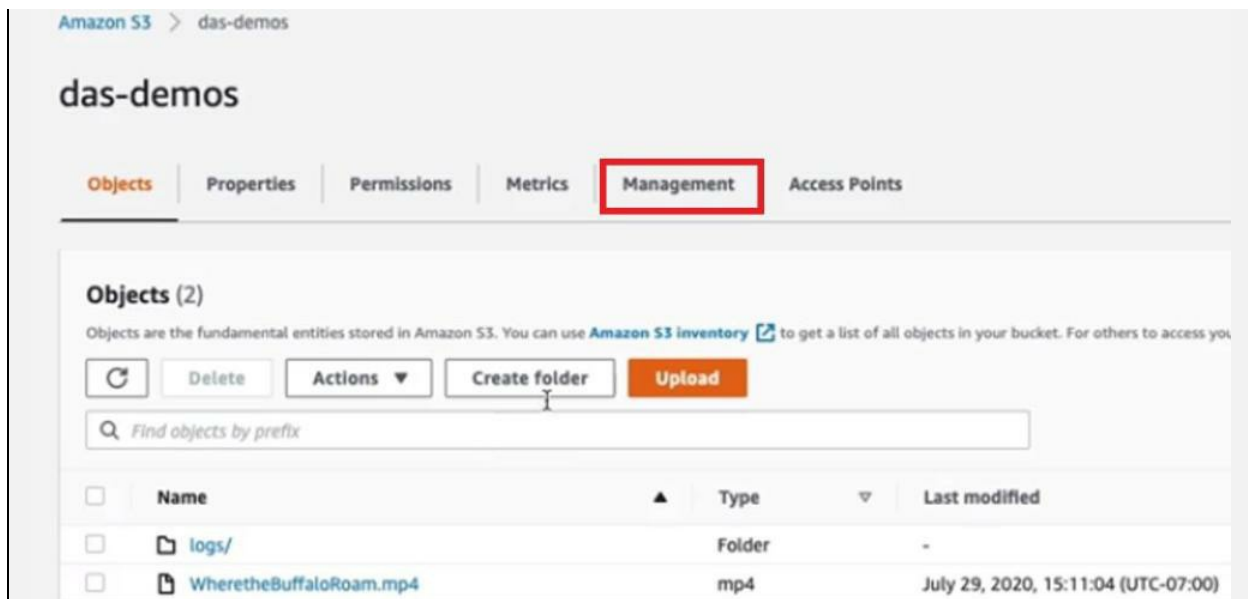
### Buckets (9)

Buckets are containers for data stored in S3. [Learn more](#) 

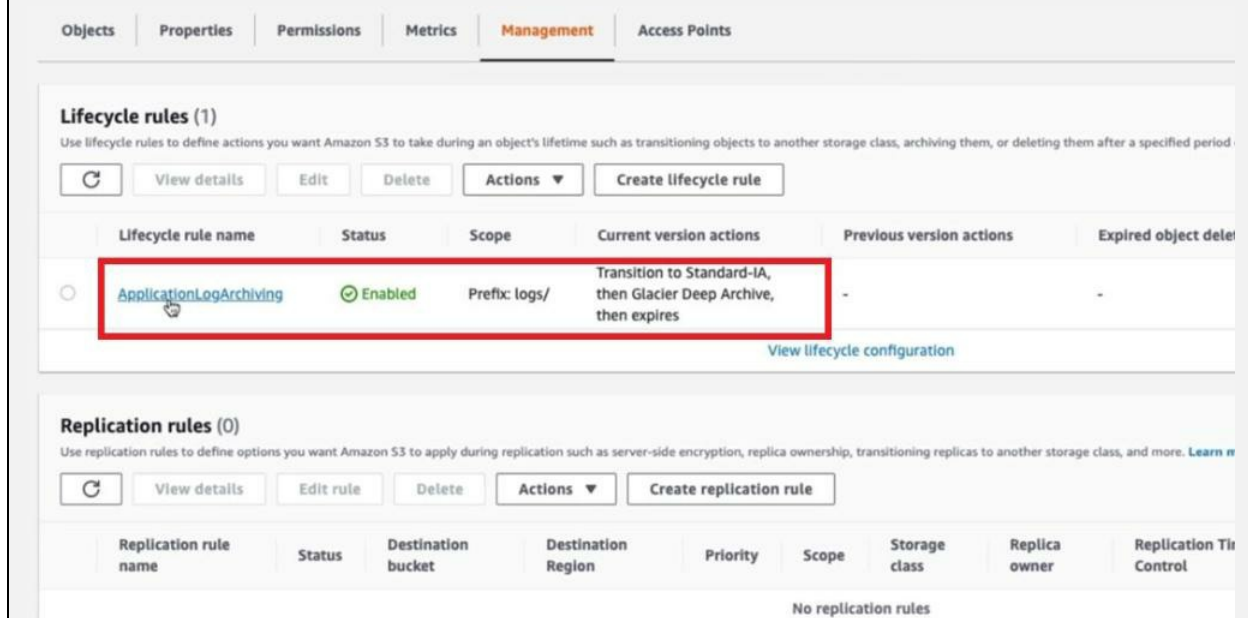
 Find buckets by name

|                       | Name                                   |
|-----------------------|--|
| <input type="radio"/> | aws-glue-assets-694501820626-us-west-2 |
| <input type="radio"/> | cf-templates-1ehmppyx3g33q-us-west-2   |
| <input type="radio"/> | das-demos                              |
| <input type="radio"/> | frank-front-dev                        |
| <input type="radio"/> | john-redshift-data                     |
| <input type="radio"/> | johntranscribe                         |
| <input type="radio"/> | redshift-data-john                     |
| <input type="radio"/> | users-test-john                        |
| <input type="radio"/> | users-test-john-frontend               |

5) Click on **Management**.



6) Here, we will find the lifecycle rule name. We can see our lifecycle rule, which is **ApplicationLogArchiving** supplied to the prefix logs. It moves data into standard infrequent access, Glacier Deep Archive, and then expires the data. It does not have actions for previous versions.





7) We can click **Edit** and get some more information.

## Lifecycle rules (1)

Use lifecycle rules to define actions you want Amazon S3 to take during an object's lifetime such as transitioning objects to another storage class, archiving them, or deleting them.

[Refresh](#) [View details](#) **Edit** [Delete](#) [Actions](#) [Create lifecycle rule](#)

| Lifecycle rule name   | Status  | Scope         | Current version actions  | Previous version actions |
|---|---|---------------|--|--------------------------|
|  ApplicationLogArchiving |  Enabled | Prefix: logs/ | Transition to Standard-IA, then Glacier Deep Archive, then expires | -                        |

[View lifecycle configuration](#)

8) When you create these policies in the console, you use a wizard. It is what our policy looks like in the wizard. We are going to limit the scope to specific prefixes or tags. We have the prefix of the log.

## Edit lifecycle rule

### Lifecycle rule configuration

#### Lifecycle rule name

ApplicationLogArchiving

Up to 255 characters.

#### Choose a rule scope

- ☒ Limit the scope of this rule using one or more filters
- ☐ This rule applies to *all* objects in the bucket

#### Filter type

You can filter objects by prefix, object tags, or a combination of both.

#### Prefix

Add filter to limit the scope of this rule to a single prefix.

logs/

Don't include the bucket name in the prefix. Using certain characters in key names can cause problems with some applications and protocols.

#### Object tags

You can limit the scope of this rule to the key/value pairs added below.

[Add tag](#)

9) Then, we have the **Lifecycle rules actions**.

**Lifecycle rule actions**  
Choose the actions you want this rule to perform. Per-request fees apply. [Learn more](#) or see [Amazon S3 pricing](#)

- ☒ Transition *current* versions of objects between storage classes
- ☐ Transition *previous* versions of objects between storage classes
- ☒ Expire *current* versions of objects
- ☐ Permanently delete *previous* versions of objects
- ☐ Delete expired delete markers or incomplete multipart uploads  
When a lifecycle rule is scoped with tags, these actions are unavailable.

**Transition current versions of objects between storage classes**


| Storage class transitions | Days after object creation |                   |
|---------------------------|----------------------------|-------------------|
| Standard-IA ▼             | 30                         | Remove transition |

10) We can see our roles here, and it gives us a warning that this lifecycle role could potentially be expensive because we will transition small objects through a glacier, Glacier Deep Archive.

**Transition current versions of objects between storage classes**


| Storage class transitions | Days after object creation |                   |
|---------------------------|----------------------------|-------------------|
| Standard-IA ▼             | 30                         | Remove transition |
| Glacier Deep Archive ▼    | 120                        | Remove transition |

[Add transition](#)

 **Transitioning small objects to Glacier or Glacier Deep Archive will increase costs**  
Lifecycle transitions include a per-object transition cost. For example, if you were to transition all objects currently in this bucket to Glacier or Glacier Deep Archive, the transition cost would be approximately 0.01 (USD). You can reduce this cost by limiting the number of objects to transition (by prefix, tag, or version) or by aggregating objects before transitioning them. [Learn more](#) or see [Amazon S3 pricing](#)

☐ I acknowledge that this lifecycle rule will increase the one-time lifecycle request cost if it transitions small objects.

11) We are going to **acknowledge** that in this wizard.

**Transitioning small objects to Glacier or Glacier Deep Archive will increase costs**

Lifecycle transitions include a per-object transition cost. For example, if you were to transition all objects currently in this bucket to Glacier or Glacier Deep Archive, the transition cost would be approximately 0.01 (USD). You can reduce this cost by limiting the number of objects to transition (by prefix, tag, or version) or by aggregating objects before transitioning them. [Learn more](#) or see [Amazon S3 pricing](#)

☒ I acknowledge that this lifecycle rule will increase the one-time lifecycle request cost if it transitions small objects.

12) We can see that after the **Number of days after object creation** is 2,555 days.

**Expire current versions of objects**

For version-enabled buckets, Amazon S3 adds a delete marker and the current version of an object is retained as a previous version. For non-versioned buckets, Amazon S3 permanently removes the object. [Learn more](#)

Number of days after object creation

**Timeline summary**

| Current version actions                          | Previous version actions           |
|--|------------------------------------|
| Day 0<br>Objects uploaded<br>↓                   | Day 0<br>Objects become noncurrent |
| Day 30<br>Objects transition to Standard-IA<br>↓ |                                    |
| Day 120  |                                    |

13) And then, we can get a review of our policy here.

14) Click **Save**.

### Timeline summary

#### Current version actions

Day 0

Objects uploaded

↓

Day 30

Objects transition to Standard-IA

↓

Day 120

Objects transition to Glacier Deep Archive

↓

Day 2555

Objects expire

#### Previous version actions

Day 0

Objects become noncurrent

Cancel

Save

The lifecycle configuration was updated. Lifecycle rule "ApplicationLogArchiving" was successfully updated. It may take some time for the configuration to be updated. Press the refresh button if changes to the rule are not displayed.

[Amazon S3](#) > [das-demos](#) > Lifecycle configuration

## Lifecycle configuration

To manage your objects so that they are stored cost effectively throughout their lifecycle, configure their lifecycle. A lifecycle configuration is a set of rules that define a

### Lifecycle rules (1)

Use lifecycle rules to define actions you want Amazon S3 to take during an object's lifetime such as transitioning objects to another storage class, archiving them, or deleting them after a specifi

View details

Edit

Delete

Actions ▼

Create lifecycle rule

| Lifecycle rule name     | Status  | Scope         | Current version actions  | Previous version actions |
|-------------------------|---------|---------------|--|--------------------------|
| ApplicationLogArchiving | Enabled | Prefix: logs/ | Transition to Standard-IA, then Glacier Deep Archive, then expires | -                        |

### KAM TIP:

- The data life is when we create data; we actively use and archive our data. That archive data may go back into active use or may reach the

end of its life and be deleted.

- Data lifecycles in S3 tend to use the S3 standard, the S3 infrequent access, and then glacier for archiving our data.
- When writing a lifecycle policy, we have to provide a filter for our policy. We will write transitions r when the data moves between the different storage classes. Then, we will give an expiration for our data, which is when the data should be automatically deleted.

## **S3 Security and Encryption**

### **S3 Security Overview**

At AWS, cloud security is a top priority. As an AWS client, you have access to a data center and network architecture designed to fulfill the needs of the most security-conscious businesses. When we look at S3 security and encryption, there are many S3 features and integrated services that provide various functions to maintain the security of our S3 buckets.

### **S3 Features**

- Access Analyzer for S3
- Amazon S3 Server Access Logging
- Bucket Policy
- Bucket Access Control List (ACL)
- Cross-Region Replication
- Multi-factor authentication (MFA) Delete
- Object Access Control List (ACL)
- Object Locking
- Versioning

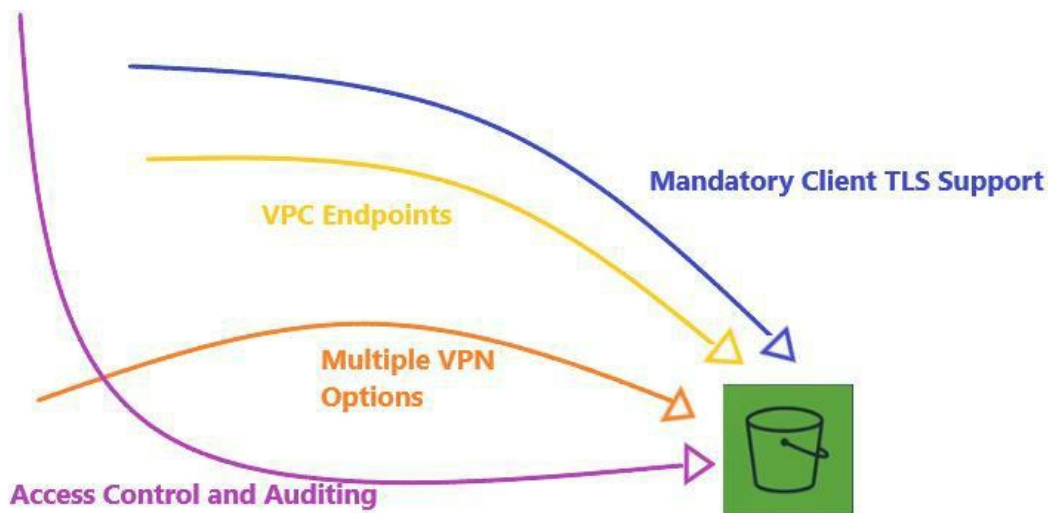
### **Integrated Services**

- Amazon CloudWatch Alarms
- AWS CloudTrail Logs
- Identity Access Management (IAM)

- VPC Endpoints
- Service Control Policies
- Key Management Service (KMS)

## In-Flight Security

Our S3 Bucket in In-Flight security requires TLS support from the clients that connect to the S3 buckets. We also have VPC endpoints, making it so that we can only access our bucket through our VPC. We can combine that VPC with VPN options to access S3 buckets from the outside. Then some access control and auditing feature that we can use to ensure that our above security features are operating in the way we expect them to.



*Figure 2-20: In-Flight Security*

1) For mandatory TLS support, the recommendation is that TLS 1.2 or above is used, but clients must at least support TLS version 1.0. Additionally, clients must support Perfect Forward Secrecy (PFS) cipher suites:

- Ephemeral Diffie-Hellman (DHE)
- Elliptic Curve Diffie Hellman Ephemeral (ECDHE)

2) VPC Endpoints create an endpoint within our VPC that we can

use to communicate with the S3 service via the private subnet of our VPC. So, we don't need to send any data out to the internet. It does not even need to traverse Amazon's public network space. All the traffic stays within our VPC. It is very useful because we can lock our buckets down.

3) If we have locked our bucket down, we may still need to access our buckets via external clients from our VPC. Therefore, we can use VPNs to connect to our VPC and then communicate with our buckets from the desired clients. There are multiple VPN options. We can build our VPN. There are AWS site-to-site VPN options, and there are direct connect options.

4) When we talk about access control and auditing, S3 is one of the oldest services. Hence, it has many layers of security on top, making it very flexible in how we control access to our buckets. We can use service control policies at the organizational level. We can use IAM to control access to buckets. We can use bucket policies. There are bucket and object ACLs and also object policies. Therefore, there are few options for setting how users in other services can access our buckets.

5) For auditing, we have CloudTrail logs, which will log the API calls. There are also S3 server access logs and the access analyzer for S3. Hence, we would need to filter those for S3.

## **At Rest Security**

At-rest encryption and security involve client-side encryption and server-side encryption.

- **Client-side encryption**

Client-side encryption has a few options. We can either store a master

key in the key fundamental management service and use that or encrypt our objects with an application stored Key. If we use the customer master key option, we need to use one of the available AWS SDKs, which are not available in any other client for S3. It is supported in the following SDKs.

- .NET
- Go
- Java
- PHP
- Ruby
- C++

If we perform client-side application encryption, the application stack is entirely responsible for encrypting and decrypting the objects stored and retrieved from S3.

## **Server-side encryption**

For server-side encryption, we have:

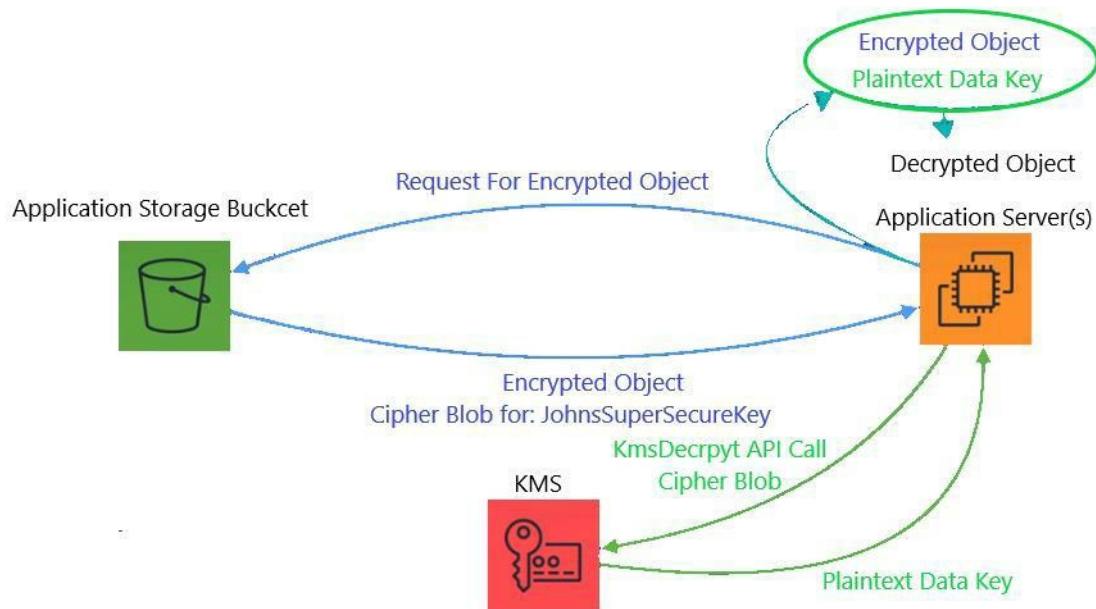
- S3 Managed Encryption keys (SSE-S3).
- We can provide our encryption keys for the encryption, i.e., Customer-provided Encryption Keys (SSE-C).
- We can use the crucial management service to store keys (SSE-KMS).

S3 will then use those for server-side encryption.

## **Client-Side Encryption**

For client-side encryption, our application server will request an encrypted object if we are using the KMS option. Our bucket is going to return that encrypted object and a cipher blob. The cipher blob identifies the Key that that object was encrypted with. Our application server then needs to call KMS for that Key. Hence, it will request the

Key associated with the cipher blob that is returned to the object, and then a data key is returned for that object. We can then combine our encrypted object and our plain text data key into a decrypted object. The SDKs take care of most of this for us, but this is the following process.



*Figure 2-21: Client-Side Encryption*

## Server-Side Encryption

For server-side encryption, if S3 manages our encryption keys, our application server will request an object, and S3 will decrypt it and send it back.



Figure 2-22: S3 Managed Encryption Keys

Using the KMS option, our Key is stored in KMS. Therefore, we request S3; S3 get the key information from KMS and return the decrypted object. It will be transparent other than rotating our keys in KMS, and the important exchange happens automatically.

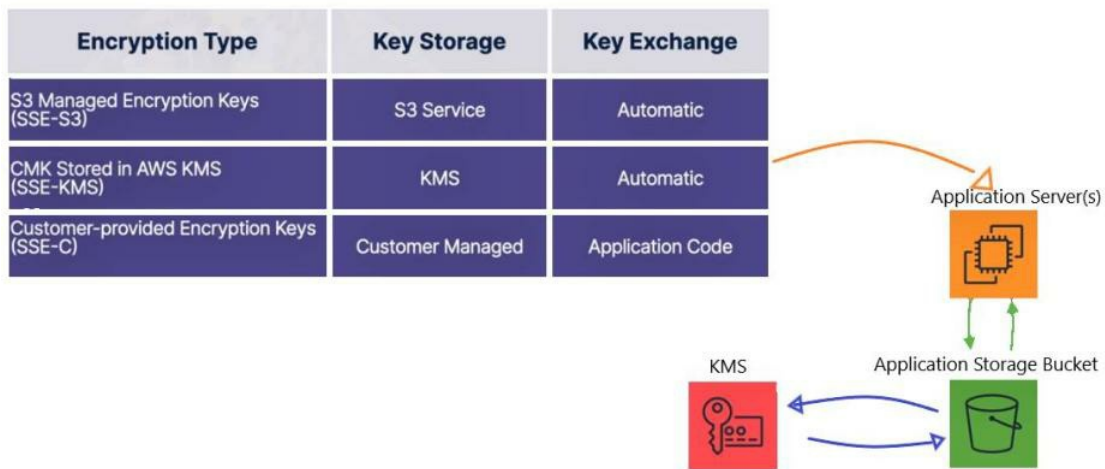
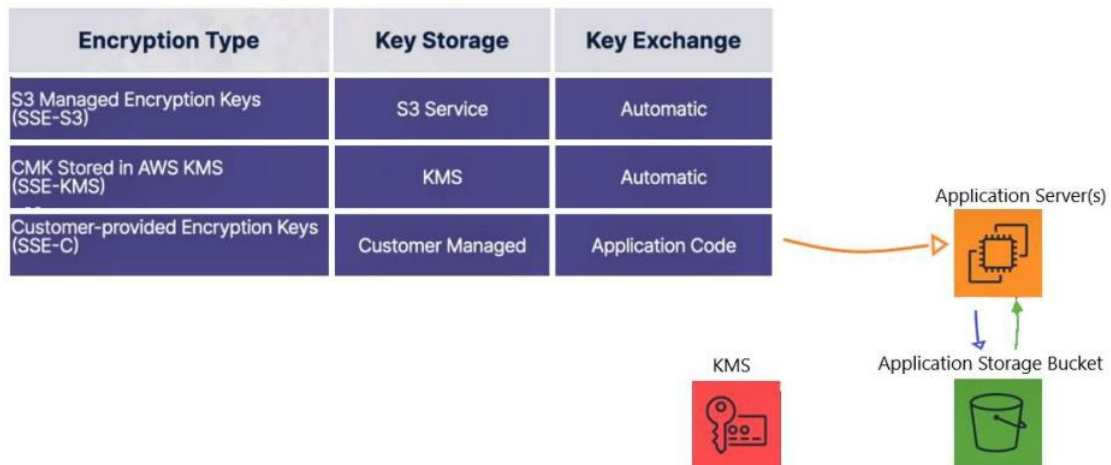


Figure 2-23: CMK stored in AWS KMS

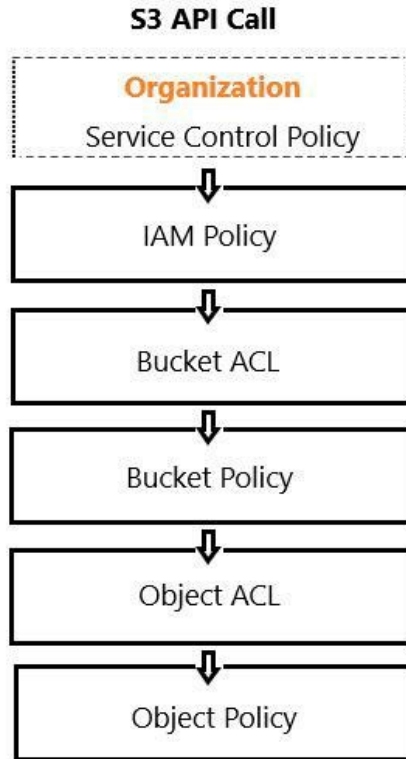
We can also provide our encryption keys, which tells S3 on the server-side, encrypt with this key and return the object, and then it is the application's responsibility to decrypt that object.



*Figure 2-24: Customer provided Encryption Keys*

## The S3 Access Security Waterfall

There is a permission waterfall when we talk about the API and our various ways to access objects in S3 and control that access. Each one of the services in the waterfall adds to the flow, and at the bottom line of the waterfall, they are all combined to create a single policy that determines whether or not we can access an object or a bucket in S3.



*Figure 2-25: S3 Access Security Waterfall*

If we look at the waterfall, there are a lot of opportunities for access to be denied. If we use an organization, the service control policy could say that this account cannot use S3. We can have an IAM policy, which can also deny; there is a bucket ACL that may also deny. Therefore, there are many restricted possibilities here as we move through to our final pool, representing a combined policy. If there is no decline in any of these steps that are flattened into a final approach, we will be provided access for API calls, whatever resources are requested.

It can get quite complicated; it is best to leave the options in this waterfall alone that do not fit our use case. From the organization's perspective, we do not want an account in our organization to use S3 at all. We are just going to turn it off via the service control policy. When we talk about IAM, maybe we have a group that we want to provide

access to, and the rest of our groups are not allowed to use S3. We can control that at the bucket policy, ACL, and object ACL and object policy. These are the original security controls that S3 launched with, and they permit controlling access from the bucket perspective.

An everyday use case is that if we are hosting a website out of our S3 bucket, we will use a bucket policy that says, this is a public bucket, and anything in this bucket can be served. We might want to be more specific about that, in which case would you use the object policy. But generally, it is better to use multiple buckets if we are storing data; we do not want it accessible from our public website.

## **Waterfall Rules**

- Implicit Deny is the default.
- Explicit Allow beats Implicit Deny.
- Explicit Deny beats Explicit Allow.

## **Access Logging, Alerting, and Auditing**

We have the S3 server access log and cloud trail logs, and with these two combined, we can get various granularities of access to our buckets. It can be used to feed data into CloudWatch, or CloudWatch on its own can be used for alerting to perform various actions. If a bucket suddenly receives a considerable amount of requests, we could set up a CloudWatch alarm that will trigger a Lambda function that will turn off access to that bucket. Maybe a timer waits a certain amount of time and then automatically re-enables access to that bucket to resume operation.

Understanding all the layers of access authorization can get quite complicated, and that is where the access analyzer for S3 comes in. It will analyze the various policies and ACLs involved in providing access,

and it will generate a report around what is available. It is useful if there is a hole that we missed in our access policies that maybe we do not want to provide access to a bucket through some specific avenue, which will reveal that for us.



*Figure 2-26: Access Logging, Alerting, and Auditing*

## Object Protection and Replication

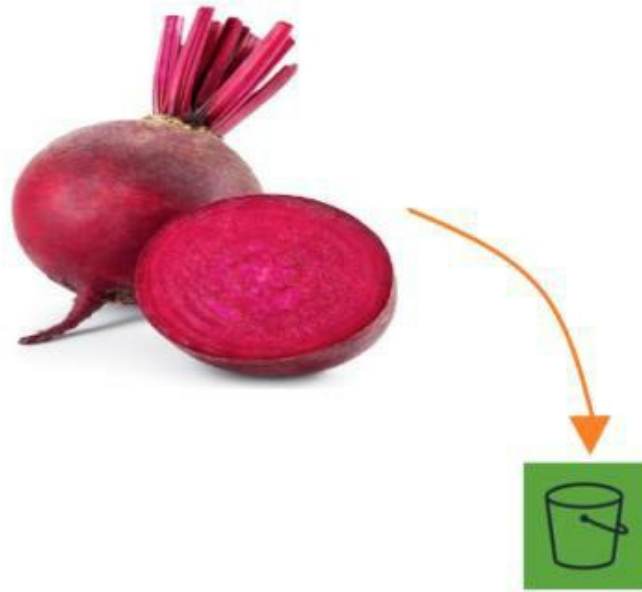
- **Protection**

We have an object that is in our S3 bucket. This object happens to be a beat. We want to protect this object so that it cannot be deleted, or we are going to put it into a Write Once Read Many modes or WORM. We can turn on object locking, which prevents the deletion of this object without disabling object locking.

Alternatively, we can enable multi-factor authentication to delete our bucket, requiring an MFA token to delete objects in the bucket. It is useful because we can control this feature with some granularity, and these users with MFA tokens are allowed to delete objects from a bucket. All other users are not, which gives our administrators the ability to remove objects if needed.

We can also turn on versioning for our buckets. It stores the specified numbers of versions of our objects in the bucket. It can be very useful if

a file becomes corrupted and accidentally stored in a bucket. We can roll back to an older version.

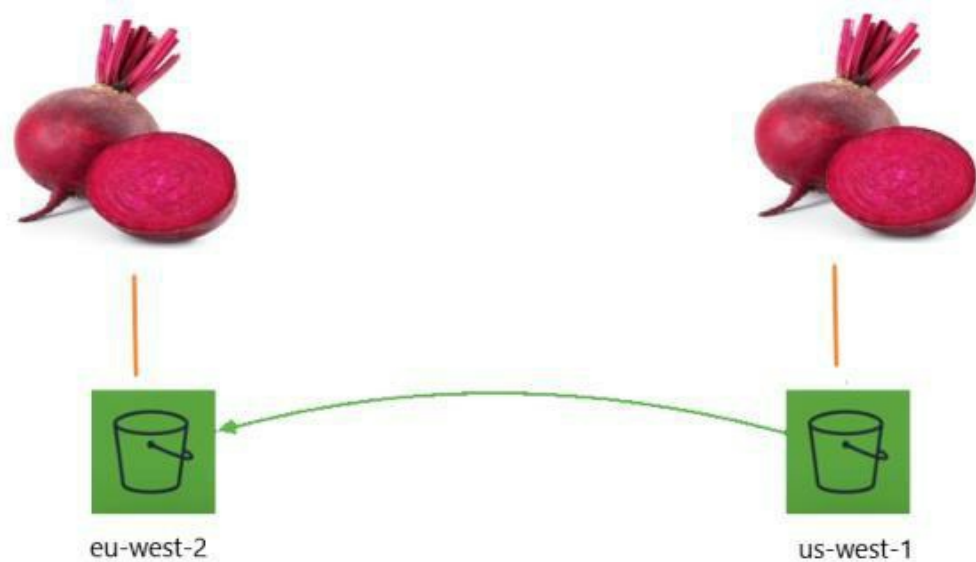


*Figure 2-27: Object Protection*

- **Replication**

We can also replicate our bucket across regions. Hence, the objects in our bucket will be copied to a bucket in a second region bucket. We will need to have a different name, but this provides disaster recovery in case there is a loss of the entire AWS region for whatever reason. We can turn on cross-region replication, and we will still access our buckets. We may need to update our code to point our applications to the correct bucket or set up some automatic failover in our application code. To turn on cross-region replication, you need to enable versioning. This is because if an object is placed in a bucket and the replication engine starts, and it is a huge object, that object is replaced. Suddenly you have an invalid replication occurring. Therefore, the service can replicate a specific version. Then if that object is overwritten

while replication is still happening, it will complete the reproduction of the performance that it is copying and then replicate the newer version. It means that it is possible for there to be some delay in replicating our objects into our second region. Still, generally, this works out very well for disaster recovery.



*Figure 2-28: Replication*

**KAM TIP:**

- For the S3 security overview, there are lots of S3 security features and support services.
- In-flight security: We have mandatory TLS or VPC endpoints, VPN options, access control, and auditing.
- For at rest encryption, we can perform client-side encryption. We can use AWS SDKs and perform server-side encryption with the key management service.
- For client-side encryption, we can use a key management service to hold our keys, and when we do that, the key identifiers stored with the object are returned where the object and the request can be made to retrieve the Key. That will allow us to decrypt that object.
- There are several server-side encryption options, and which one we choose will depend on our security requirements.

- We have a lot of steps in our waterfall of security that are going to produce our final policy that dictates whether or not your particular client gets access to a bucket or objects within that bucket.
- For access logging and alerting, there are server access logs in S3. We also have CloudTrail logs. We can use CloudWatch alarms and the S3 access analyzer.
- For object protection and replication, we can lock our objects. We can require multi-factor authentication for deleting objects. We can enable versioning to keep multiple versions of our substances.
- Once we have versioning enabled, we can turn on cross-region replication.

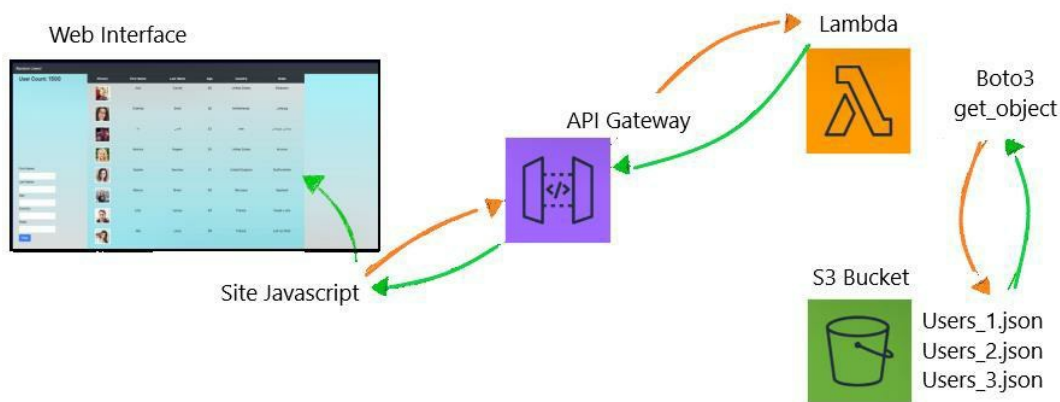
## Lab 2-01: Programmatically Utilizing Data from S3

### Introduction

In this lab, we will create a Lambda function that gathers data from S3, applies some basic formatting, and sends it to API Gateway to be put into a simple web interface.

### Problem

You have been assigned to a team to develop a proof of concept for your company's low-cost employee directory. The team built a simple web interface, arranged some placeholder employee data, and uploaded it to S3. They are experiencing problems moving data from S3 into Lambda, and they are looking for your help. All 1500 placeholder records from JSON files stored in an S3 bucket must be collected and returned in a single return from the Lambda function.

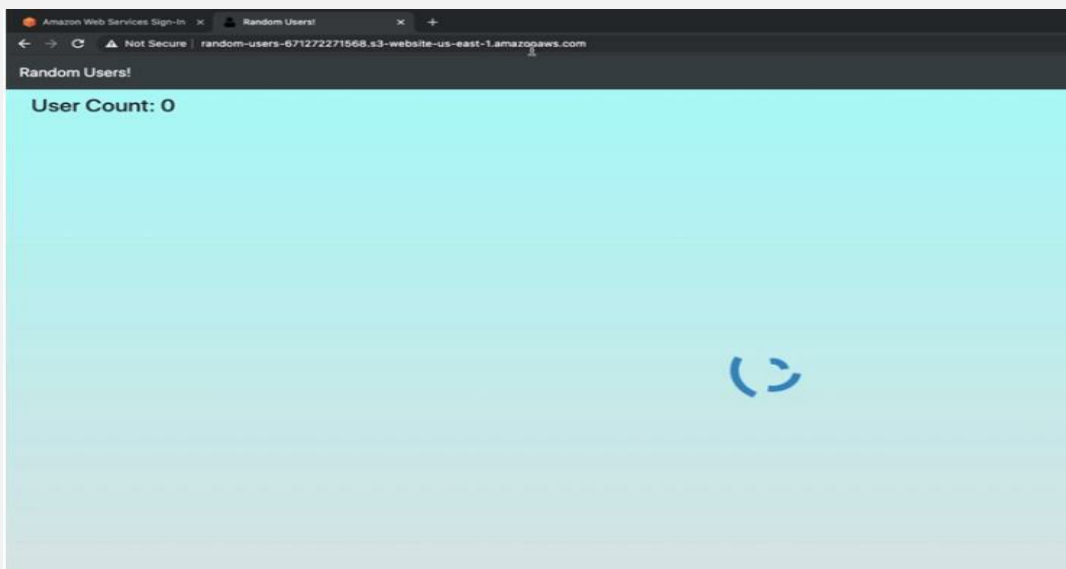


*Figure 2-29: Lab Diagram*

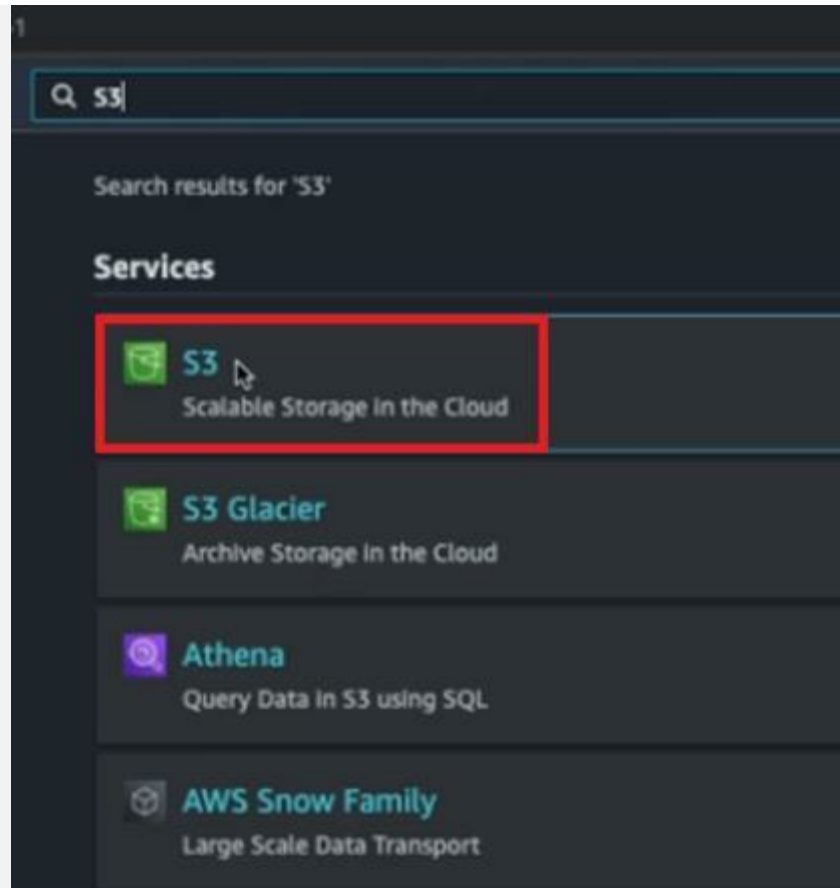
## Solution

### Step 1: Investigate the Lab Environment

1) Open the **random-userswebsite** provided for the lab. The site will not load yet because you have not assigned the Lambda function an action.



2) From the AWS Management Console, navigate to **S3** using the Services menu.



3) You should see two buckets in your account: **random-users-  
<ACCOUNT\_NUMBER>**, and **random-users-data-  
<ACCOUNT\_NUMBER>**.

**Buckets (2)** [Info](#)

Buckets are containers for data stored in S3. [Learn more](#) [↗](#)

[↻](#) [📄 Copy ARN](#) [Empty](#) [Delete](#) [Create bucket](#)






[🔍](#) Find buckets by name



|                       | Name ▲                         | AWS Region ▼                       | Access ▼                  |
|-----------------------|--------------------------------|------------------------------------|---------------------------|
| <input type="radio"/> | random-users-649397770088      | US East (N. Virginia)<br>us-east-1 | <a href="#">⚠️ Public</a> |
| <input type="radio"/> | random-users-data-649397770088 | US East (N. Virginia)<br>us-east-1 | Objects can be public     |


4) Click on **random-users-data-<ACCOUNT\_NUMBER>**, which is the data that will populate the website.





**Buckets (2)** [Info](#)



Buckets are containers for data stored in S3. [Learn more](#)





  Copy ARN  Empty  Delete 

 Find buckets by name < 1 > 

|                       | Name ▲                           | AWS Region ▼                       | Access ▼   | Creation date ▼                         |
|-----------------------|----------------------------------|------------------------------------|--|---|
| <input type="radio"/> | <b>random-users-745044865554</b> | US East (N. Virginia)<br>us-east-1 |  Public | November 14, 2021, 16:32:33 (UTC+05:00) |
| <input type="radio"/> | random-users-data-745044865554   | US East (N. Virginia)<br>us-east-1 | Objects can be public  | November 14, 2021, 16:32:33 (UTC+05:00) |

 Services ▼    More ▼

| <input type="checkbox"/> | Name ▲  | Type ▼ | Last modified ▼                         | Size ▼  | Storage class ▲ |
|--------------------------|---|--------|---|---------|-----------------|
| <input type="checkbox"/> |  <a href="#">icon.png</a>        | png    | November 14, 2021, 16:33:04 (UTC+05:00) | 10.8 KB | Standard        |
| <input type="checkbox"/> |  <a href="#">index.html</a>      | html   | November 14, 2021, 16:33:03 (UTC+05:00) | 3.7 KB  | Standard        |
| <input type="checkbox"/> |  <a href="#">randomusers.css</a> | css    | November 14, 2021, 16:33:04 (UTC+05:00) | 1.4 KB  | Standard        |
| <input type="checkbox"/> |  <a href="#">randomusers.js</a>  | js     | November 14, 2021, 16:33:04 (UTC+05:00) | 2.2 KB  | Standard        |






[Feedback](#) [English \(US\) ▼](#) [Privacy Policy](#) [Terms of Use](#) [Cookie preferences](#)


© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.


5) Click on **random-users-<ACCOUNT\_NUMBER>**, which is your static website.




**Buckets (2)** [Info](#)



Buckets are containers for data stored in S3. [Learn more](#)


  Copy ARN  Empty  Delete  Create bucket




< 1 > 

|                       | Name ▲                         | AWS Region ▼                       | Access ▼   | Creation date ▼                         |
|-----------------------|--------------------------------|------------------------------------|--|---|
| <input type="radio"/> | random-users-745044865554      | US East (N. Virginia)<br>us-east-1 |  Public | November 14, 2021, 16:32:33 (UTC+05:00) |
| <input type="radio"/> | random-users-data-745044865554 | US East (N. Virginia)<br>us-east-1 | Objects can be public  | November 14, 2021, 16:32:33 (UTC+05:00) |

**aws** Services ▼    More ▼

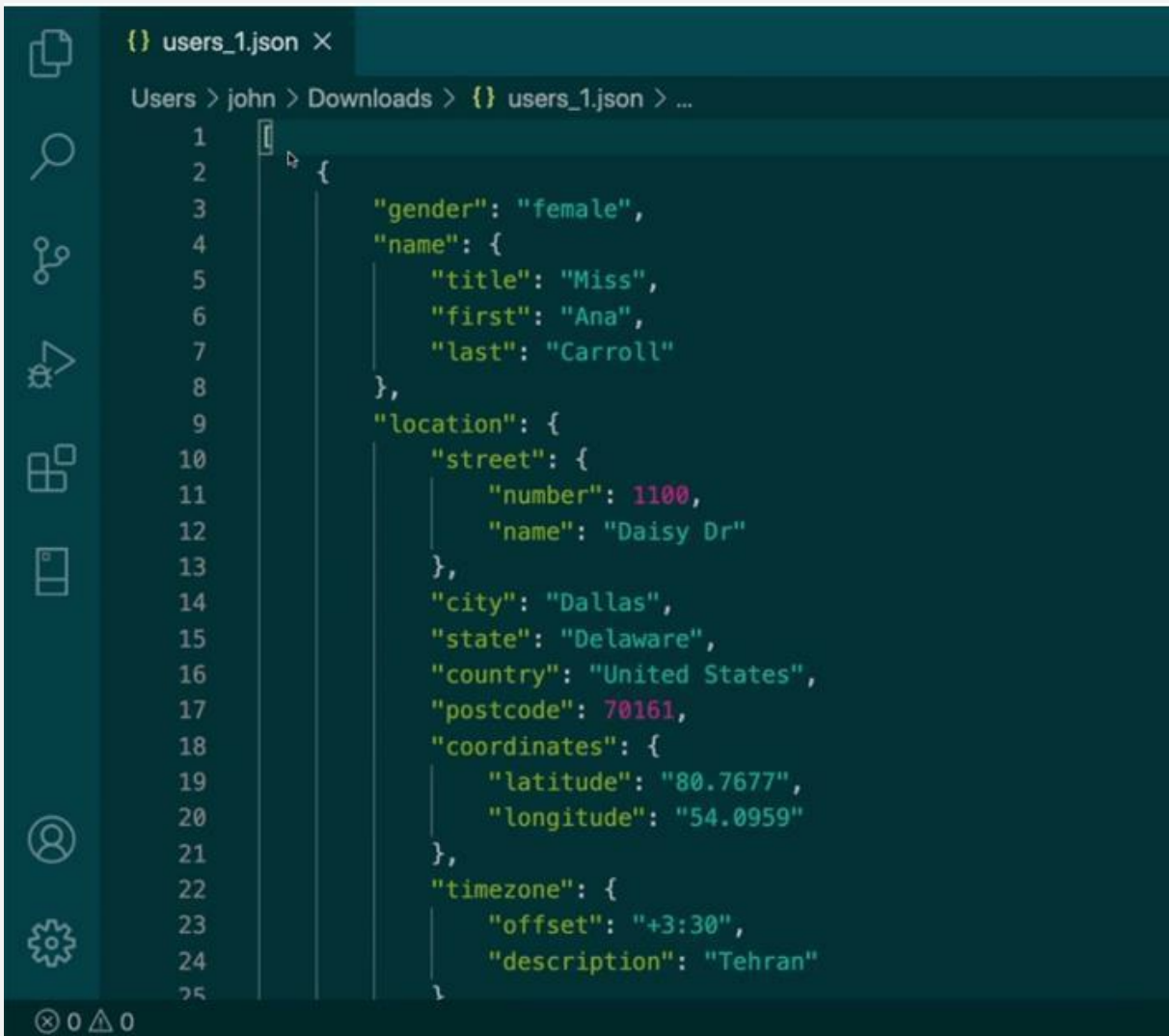
< 1 > 

| <input type="checkbox"/> | Name ▲   | Type ▼ | Last modified ▼                         | Size ▼   | Storage class ▼ |
|--------------------------|--|--------|---|----------|-----------------|
| <input type="checkbox"/> |  <a href="#">users_1.json</a> | json   | November 14, 2021, 16:33:03 (UTC+05:00) | 678.6 KB | Standard        |
| <input type="checkbox"/> |  <a href="#">users_2.json</a> | json   | November 14, 2021, 16:33:03 (UTC+05:00) | 676.4 KB | Standard        |
| <input type="checkbox"/> |  <a href="#">users_3.json</a> | json   | November 14, 2021, 16:33:03 (UTC+05:00) | 676.8 KB | Standard        |

Feedback English (US) ▼ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

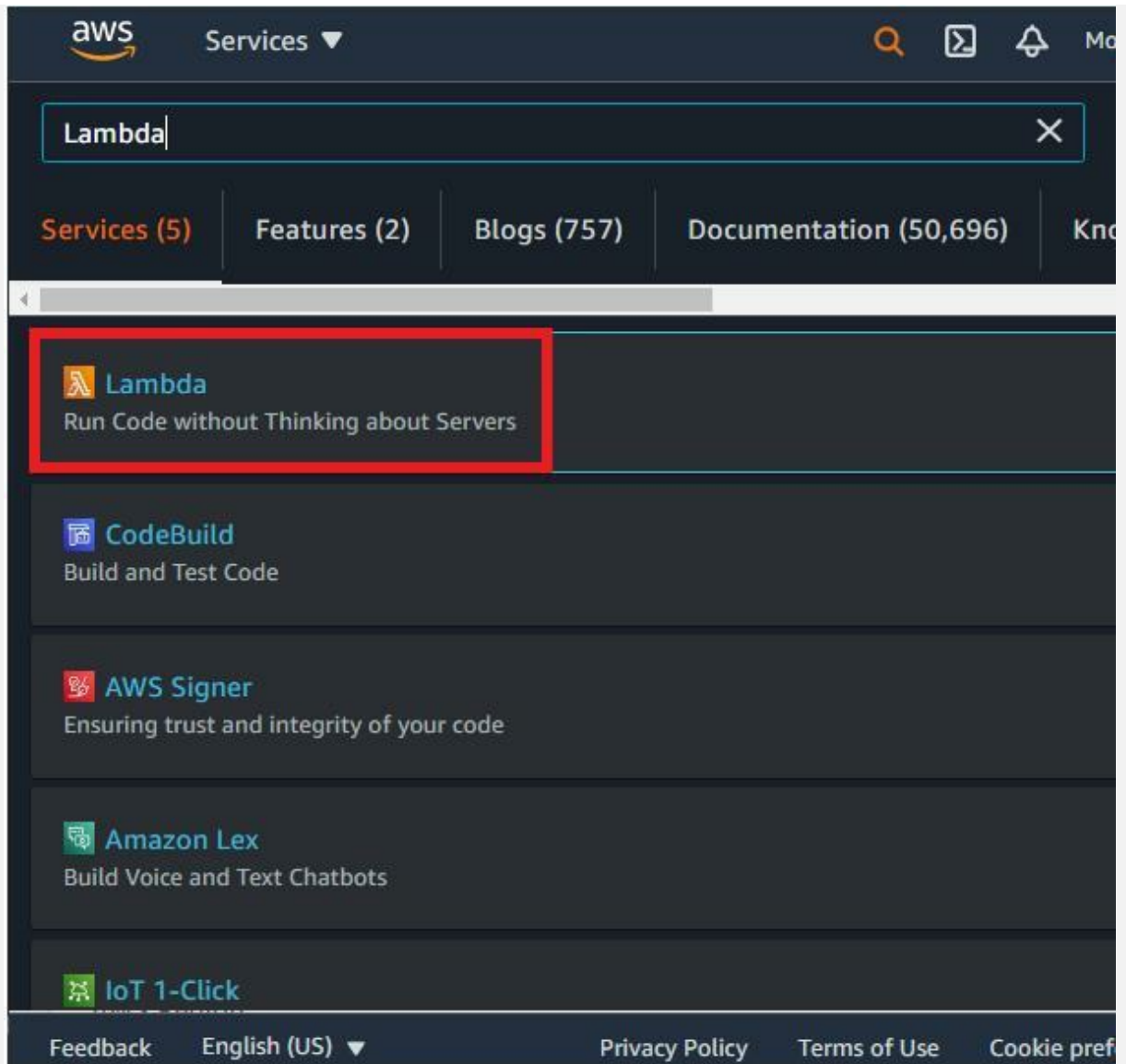
6) We can open the **user\_1.json** file and review the user data. You will collect the data from all three objects and organize it into a single entity returned in the web interface.

A screenshot of a code editor with a dark theme. The editor shows a file named 'users\_1.json' with a breadcrumb path 'Users > john > Downloads > {} users\_1.json > ...'. The code is a JSON object representing a user. The left sidebar contains icons for file explorer, search, source control, run and debug, extensions, testing, user profile, and settings. The status bar at the bottom shows '0' errors, '0' warnings, and '0' info messages.

```
1  {
2    {
3      "gender": "female",
4      "name": {
5        "title": "Miss",
6        "first": "Ana",
7        "last": "Carroll"
8      },
9      "location": {
10       "street": {
11         "number": 1100,
12         "name": "Daisy Dr"
13       },
14       "city": "Dallas",
15       "state": "Delaware",
16       "country": "United States",
17       "postcode": 70161,
18       "coordinates": {
19         "latitude": "80.7677",
20         "longitude": "54.0959"
21       },
22       "timezone": {
23         "offset": "+3:30",
24         "description": "Tehran"
25       }
26     }
27   }
```



## Step 2: Create the Employee Directory Using Objects Keys and Data from S3


1) Navigate to **Lambda** using the Services menu.







- 2) You should see 2 Lambda functions in your account.
- 3) Select the **Users\_primary** function.


**Functions (2)**

Last fetched 12 seconds ago  **Actions**  **Create function**

 *Filter by tags and attributes or search by keyword*

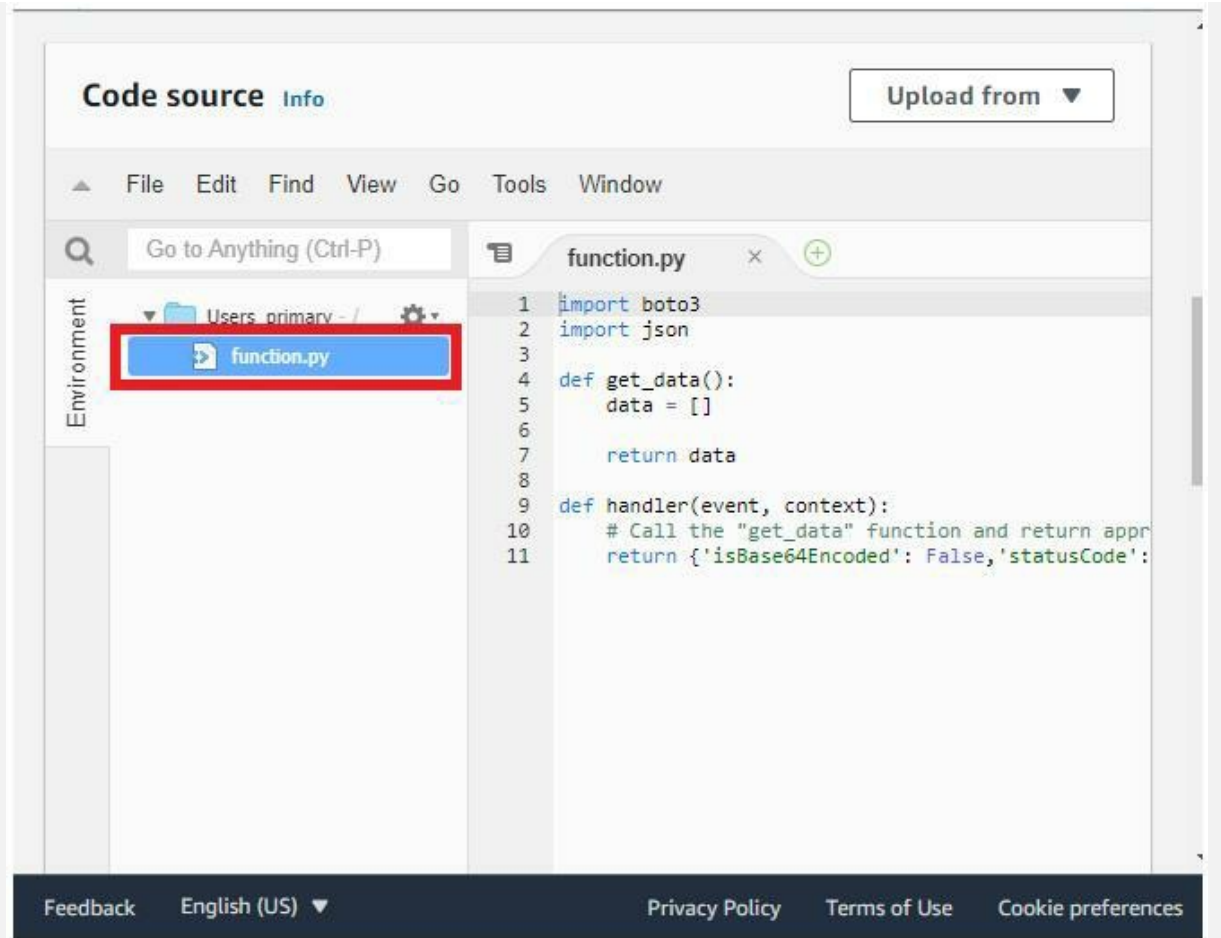
< 1 > 

| <input type="checkbox"/> | Function name  | Description  | Package type  | Runtime  | Code size  |
|--------------------------|---|--|---|---|------------|
| <input type="checkbox"/> | <b>Users_primary</b>  | -  | Zip   | Python 3.8  | 426.0 byte |
| <input type="checkbox"/> | users-index-provisioner   | Copies objects from source bucket, modifies API Gateway Placeholder to URL | Zip   | Python 3.7  | 2.7 k      |

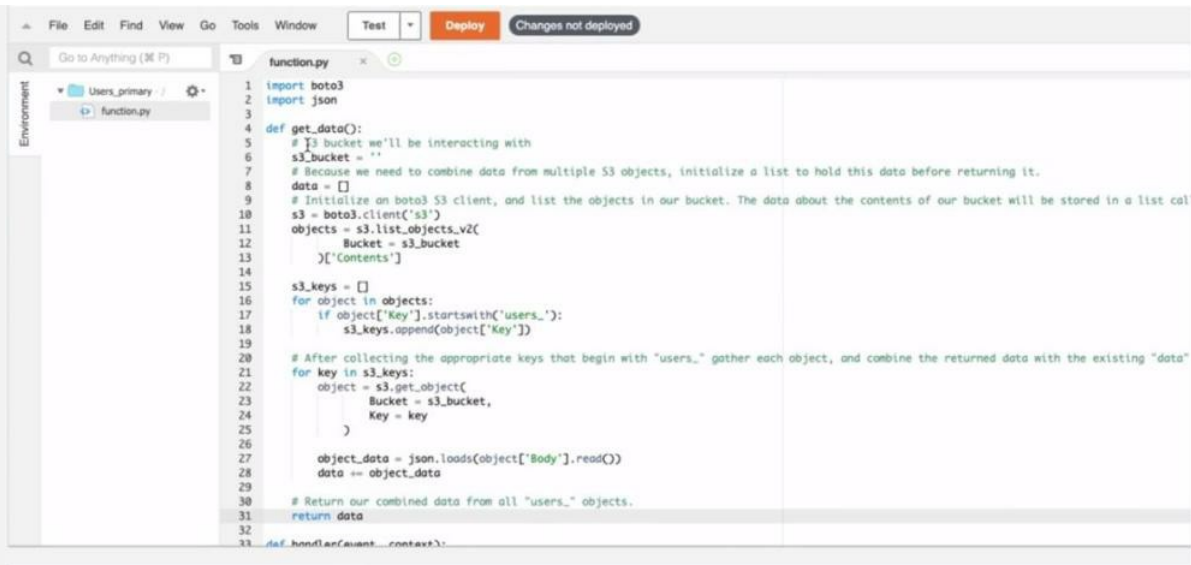
Feedback English (US)  Privacy Policy Terms of Use Cookie preferences

© 2008 – 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

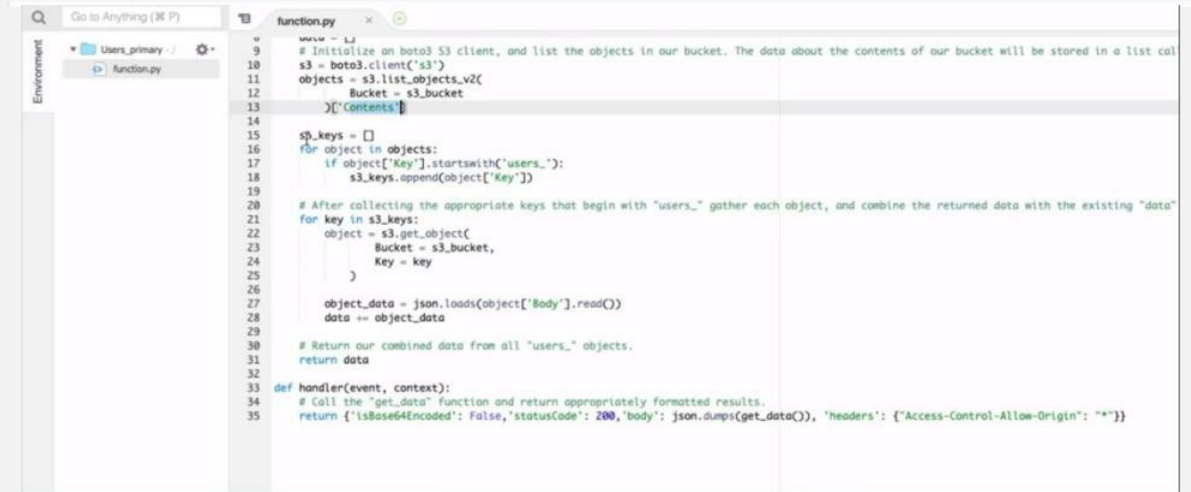
4) In the Code Source section, select **function.py** and review the code.



5) Replace the existing code with the **function\_solved.py** code provided in the GitHub link. [https://github.com/linuxacademy/Content-AWS-Certification-Specialty/tree/master/Lab\\_Assets/programmatically\\_utilizing\\_data\\_from\\_Amazon\\_S3](https://github.com/linuxacademy/Content-AWS-Certification-Specialty/tree/master/Lab_Assets/programmatically_utilizing_data_from_Amazon_S3)



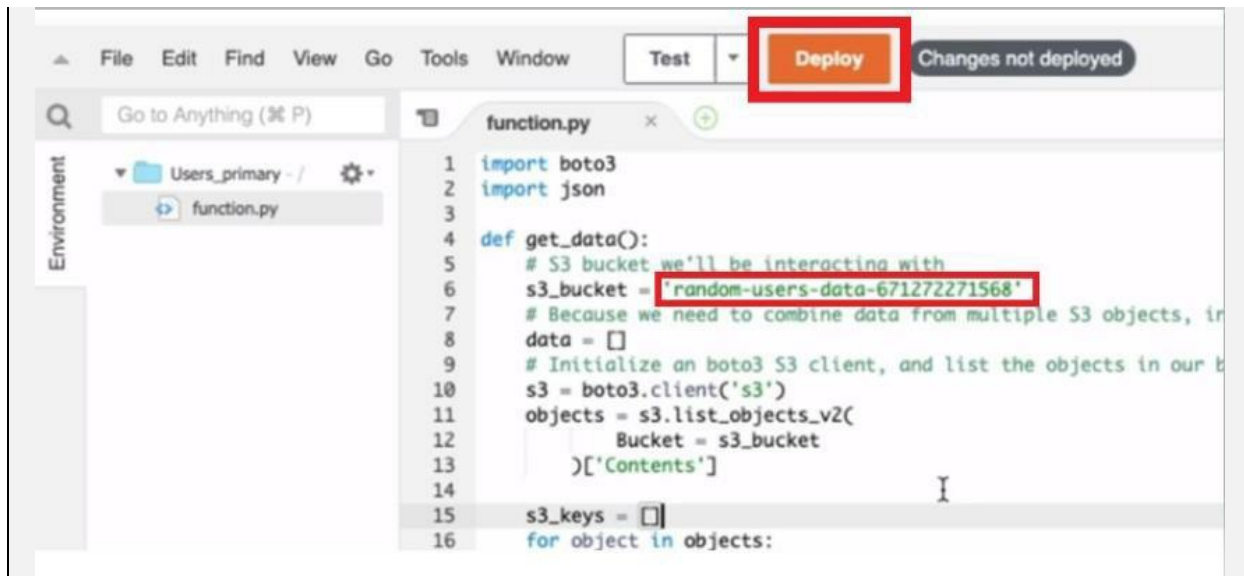
```
1 import boto3
2 import json
3
4 def get_data():
5     # S3 bucket we'll be interacting with
6     s3_bucket = ''
7     # Because we need to combine data from multiple S3 objects, initialize a list to hold this data before returning it.
8     data = []
9     # Initialize an boto3 S3 client, and list the objects in our bucket. The data about the contents of our bucket will be stored in a list called
10    s3 = boto3.client('s3')
11    objects = s3.list_objects_v2(
12        Bucket = s3_bucket
13    )['Contents']
14
15    s3_keys = []
16    for object in objects:
17        if object['Key'].startswith('users_'):
18            s3_keys.append(object['Key'])
19
20    # After collecting the appropriate keys that begin with "users_" gather each object, and combine the returned data with the existing "data"
21    for key in s3_keys:
22        object = s3.get_object(
23            Bucket = s3_bucket,
24            Key = key
25        )
26
27        object_data = json.loads(object['Body'].read())
28        data += object_data
29
30    # Return our combined data from all "users_" objects.
31    return data
32
33 def handler(event, context):
```



```
9
10 # Initialize an boto3 S3 client, and list the objects in our bucket. The data about the contents of our bucket will be stored in a list called
11 s3 = boto3.client('s3')
12 objects = s3.list_objects_v2(
13     Bucket = s3_bucket
14 )['Contents']
15
16 s3_keys = []
17 for object in objects:
18     if object['Key'].startswith('users_'):
19         s3_keys.append(object['Key'])
20
21 # After collecting the appropriate keys that begin with "users_" gather each object, and combine the returned data with the existing "data"
22 for key in s3_keys:
23     object = s3.get_object(
24         Bucket = s3_bucket,
25         Key = key
26     )
27
28     object_data = json.loads(object['Body'].read())
29     data += object_data
30
31 # Return our combined data from all "users_" objects.
32 return data
33
34 def handler(event, context):
35     # Call the "get_data" function and return appropriately formatted results.
36     return {'statusCode': 200, 'body': json.dumps(get_data()), 'headers': {'Access-Control-Allow-Origin': '*'}}
```

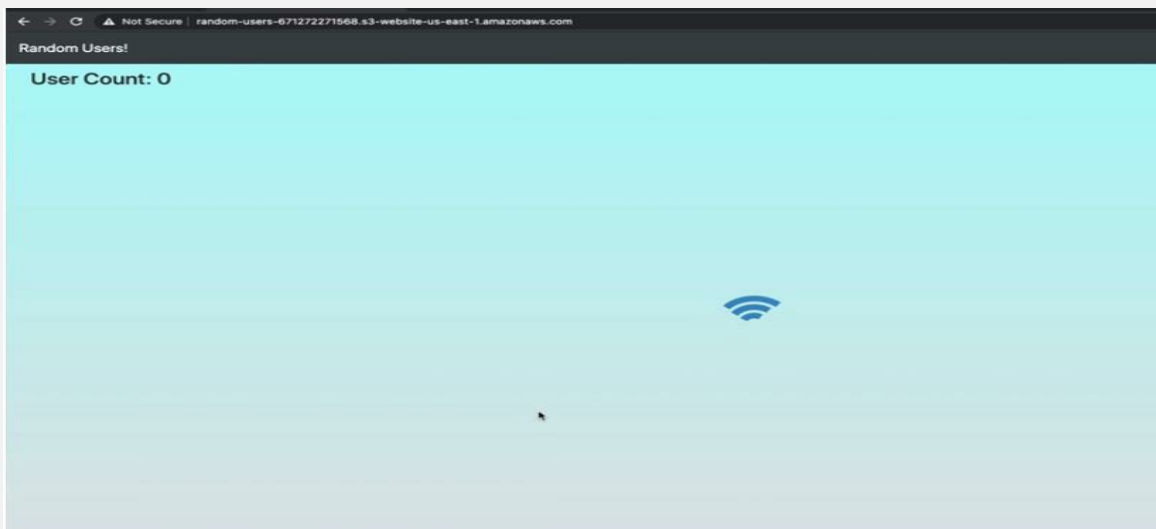
6) Copy the **random-users-data-<ACCOUNT\_NUMBER>** bucket name and paste the bucket name on the **s3\_bucket =** line.

7) Click **Deploy**.



### Step 3: Observe the Results on the Web Interface










1) After the changes are successfully deployed, navigate to the **random-users website**. You may need to refresh the page and wait a few moments for the data to load.



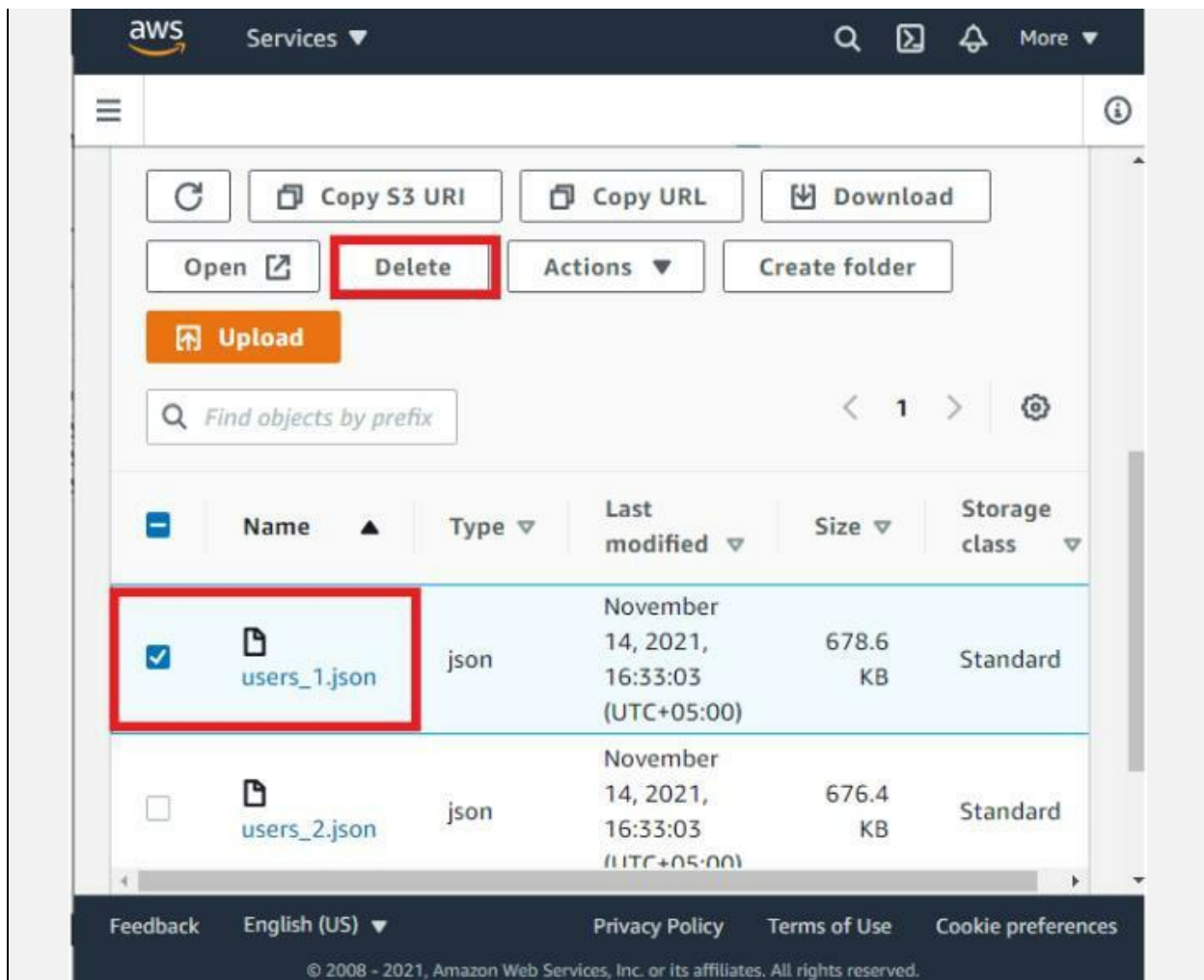
2) All **1500 users** should load correctly.

Random Users!

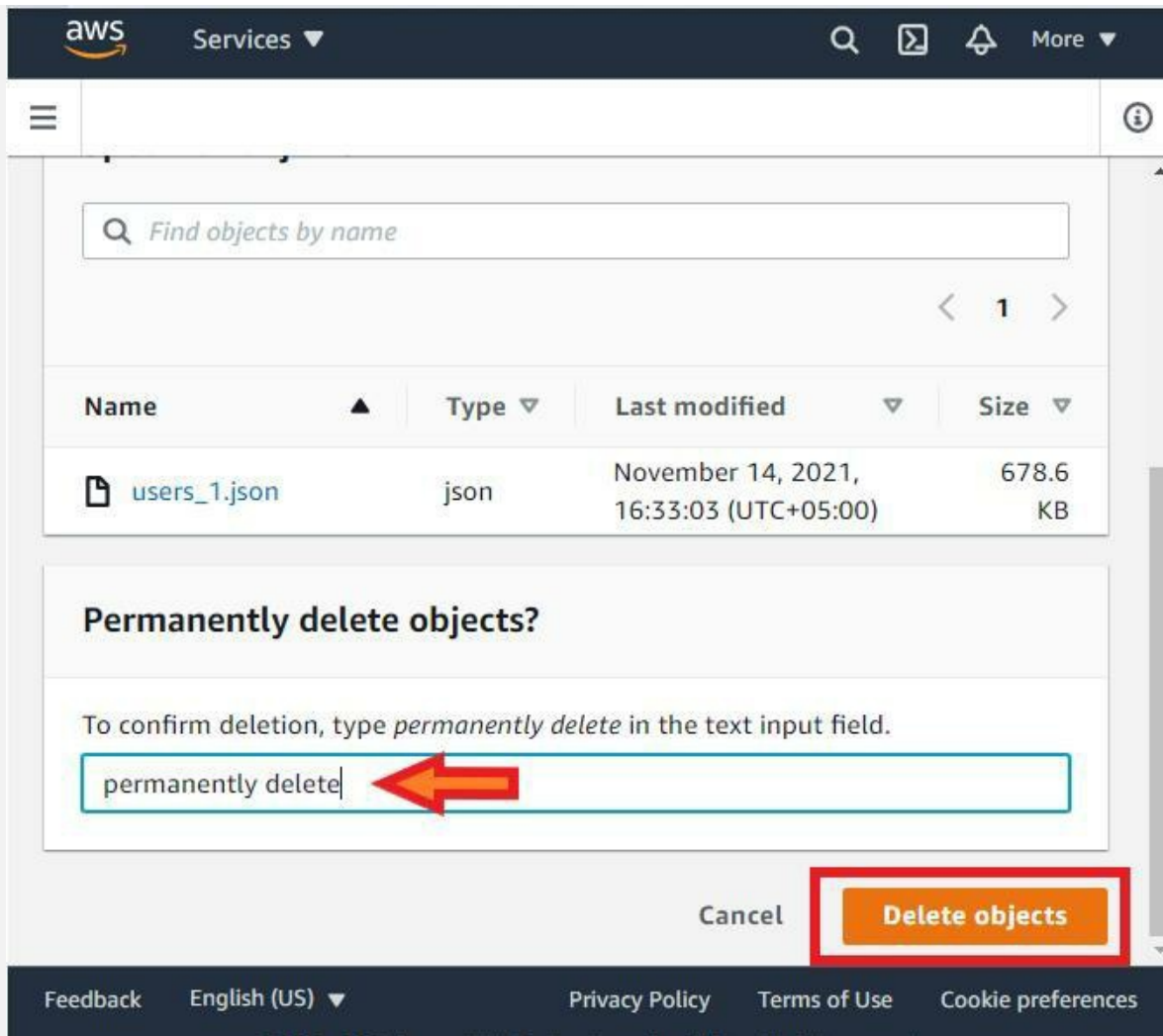
User Count: 1500

| Picture   | First Name | Last Name | Age | Country        | State             |
|---|------------|-----------|-----|----------------|-------------------|
|  | Ana        | Carroll   | 65  | United States  | Delaware          |
|  | Chahida    | Smid      | 26  | Netherlands    | Limburg           |
|  | فاطمه      | قاسمی     | 32  | Iran           | سیستان و بلوچستان |
|  | Monica     | Rogers    | 30  | United States  | Arizona           |
|  | Sophie     | Sanchez   | 41  | United Kingdom | Staffordshire     |
|  | Blanca     | Breier    | 60  | Germany        | Saarland          |
|  | Loïc       | Leroux    | 54  | France         | Haute-Loire       |
|  | Mia        | Leroy     | 38  | France         | Loir-et-Cher      |
|  | Julia      | Maki      | 46  | Finland        | North Karelia     |

3) Navigate to S3 select the **users\_1.json** object. Click **Delete**.











4) Type: **permanently delete** into the text field and click **Delete objects**.



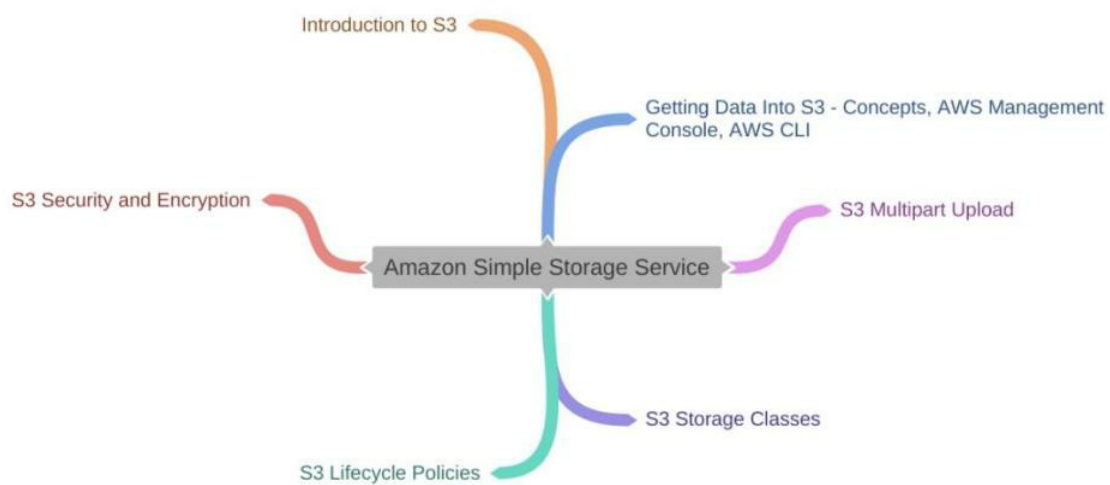
5) Navigate back to the random-users website and refresh the page. You should now have **1000 employee** records instead of 1500.

Random Users!

User Count: 1000

| Picture   | First Name | Last Name | Age | Country       | State                        |
|---|------------|-----------|-----|---------------|------------------------------|
|  | Arlo       | Roberts   | 67  | New Zealand   | Wellington                   |
|  | Caroline   | Petersen  | 33  | Denmark       | Sjælland                     |
|  | Daryl      | McDonald  | 69  | United States | New Hampshire                |
|  | Iida       | Wirkkala  | 33  | Finland       | Ostrobothnia                 |
|  | Kirk       | Simmons   | 39  | Australia     | Australian Capital Territory |
|  | Earl       | Carr      | 52  | Australia     | New South Wales              |
|  | Ece        | Sadiklar  | 26  | Turkey        | Erzincan                     |
|  | Mille      | Madsen    | 73  | Denmark       | Hovedstaden                  |

## Mind Map



*Figure 2-30: Mind Map*

## Practice Questions

1. Amazon S3 is a type of object storage that allows you to store and recover any quantity of data from any location. True or false?

- True
- False

2. When we go to upload data at S3, we have ----- interfaces to work with.

- 4
- 3
- 2

3. When we use the -----, we use a graphical user interface.

- The AWS CLI
- AWS SDKs
- AWS management console

4. If we are using -----, we enter commands in our terminal that will allow us to move data into our S3 bucket.

- The AWS CLI
- AWS SDKs
- AWS management console

5. Transfer Acceleration is enabled per bucket. True or false?

- True
- False

6. There are no additional costs for using Transfer Acceleration. True or false?

- True
- False

7. Transfer Acceleration leverages the edge locations to send data back to S3. True or false?

- True
- False

8. To get a five-tebibyte object into S3, we will use multipart upload. True or false?

- True
- False

9. There are ----- API calls that we use to perform multipart upload process.

- Five
- Four

A. Three

10. We cannot overwrite the parts while the multipart upload is still in progress. True or false?

B. True

C. False

11. ----- is a storage type for general-purpose storage of commonly accessed data.

D. S3 Standard

E. S3 Intelligent-Tiering

F. Standard Infrequent Access

12. ----- is utilized for data with uncertain or changing access patterns.

G. S3 Standard

H. S3 Intelligent-Tiering

I. Standard Infrequent Access

13. ----- offers three retrieval options, ranging from a few minutes to hours, to keep prices reasonable while meeting a variety of demands.

J. S3 Glacier

K. S3 Standard

L. S3 Intelligent-Tiering

14. S3 Glacier Deep Archive is Amazon S3's cheapest storage tier, and it allows for long-term data retention and digital preservation for material that is only viewed once or twice a year. True or false?

M. True

N. False

15. ----- is an Archival storage class that has the minutes to hours data retrieval time.

O. S3 Glacier

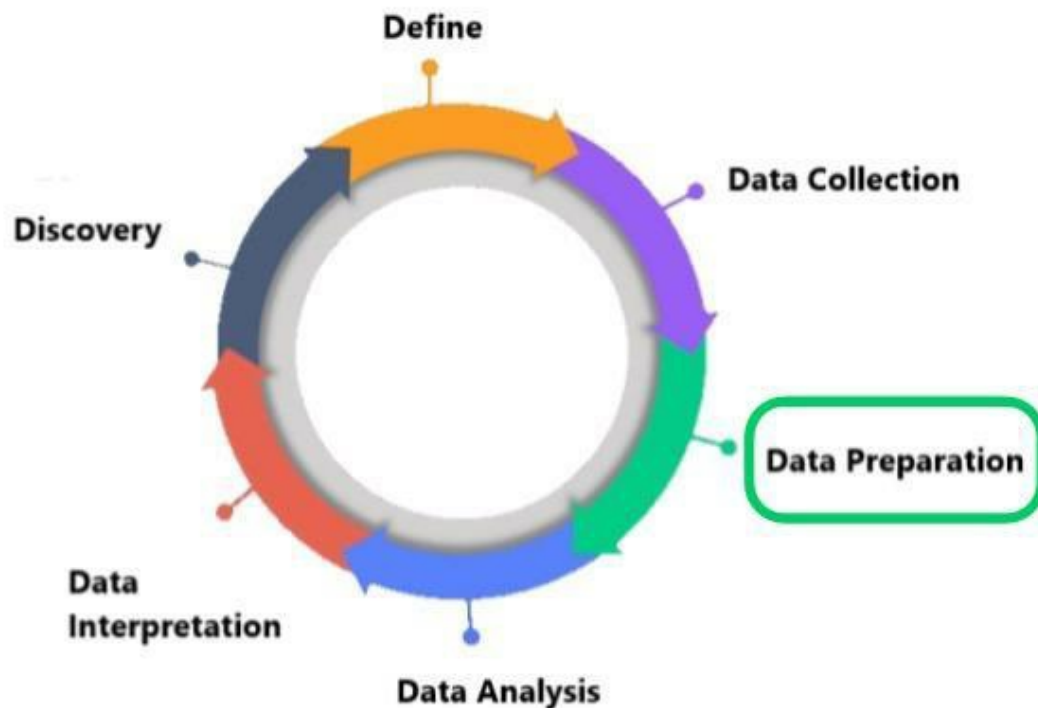
P. S3 Standard

Q. S3 Intelligent-Tiering

# CHAPTER 03: DATABASES IN AWS

## Introduction

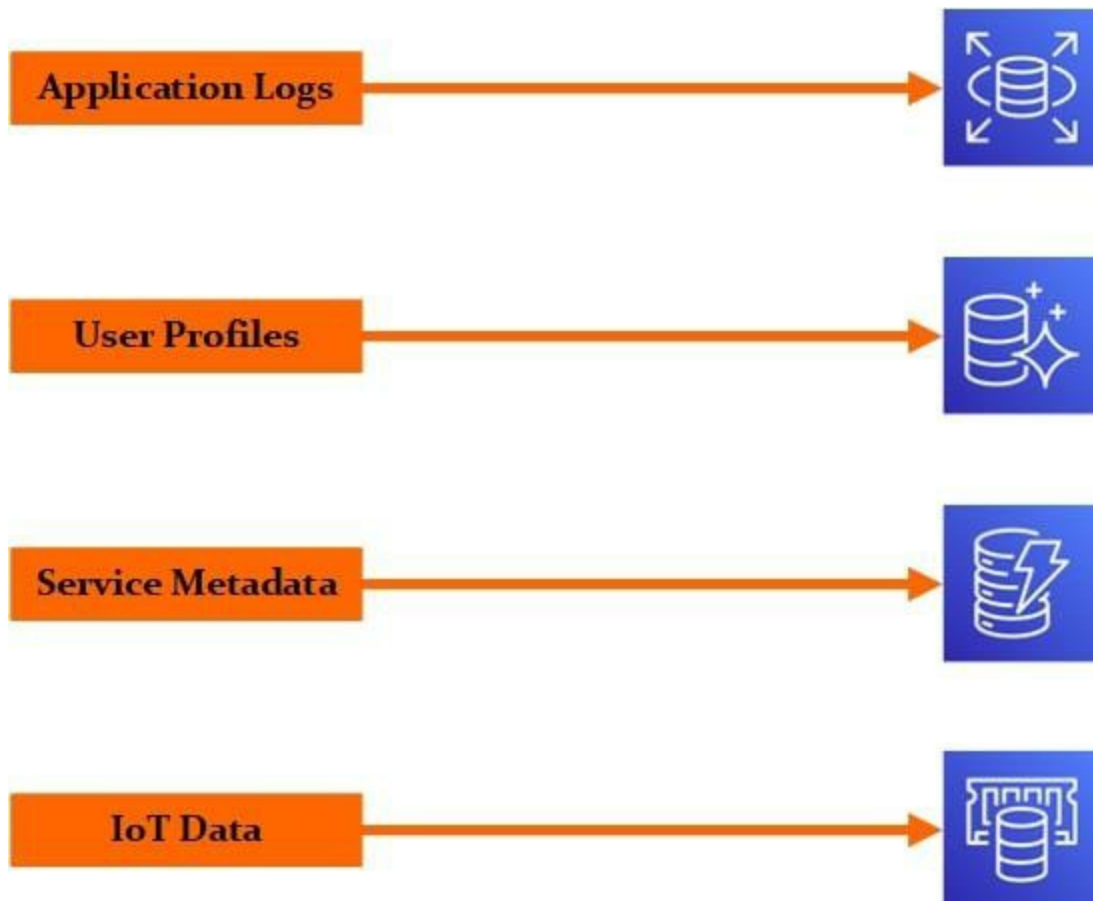
In the analytics process, how do you use databases? We will discuss about databases services in AWS in this chapter. In the analytics process, how do you use databases? For example, databases. Databases will, for the most part, be in our data preparation section. They will be a starting point for aggregating data or a source of data that you will input into your pipeline, perhaps as a secondary dimension, or collect and send into our data analysis. It might be moved to a data warehouse or data lake, or it could be read directly from the database in our pipeline.



*Figure 3-01: Introduction to Databases in AWS*

## Organizing Data

We might have user profile service metadata application logs and IoT data. We will take that data and put it in our databases, then gather it all together, get it organized, and clean up our little pile of data.



*Figure 3-02: Organizing Data in the Database*

## Services

We have the relational database service Aurora, an engine that runs in the relational database service, DynamoDB, and ElastiCache. All of these have different use cases.



*Figure 3-03: AWS Database Services*

## **Database Engines Types**

A database engine (sometimes known as a storage engine) is the software component that enables a Database Management System (DBMS) to generate, read, update, and delete (GRUD) data from a database. Most database management systems have an Application Programming Interface (API) that allows users to communicate with the underlying engine without going via the DBMS's user interface.

The terms "database engine" and "database server" or "database management system" are commonly interchanged. The processes and memory structures of the running database engine are referred to as a 'database instance.'

## **Relational Database**

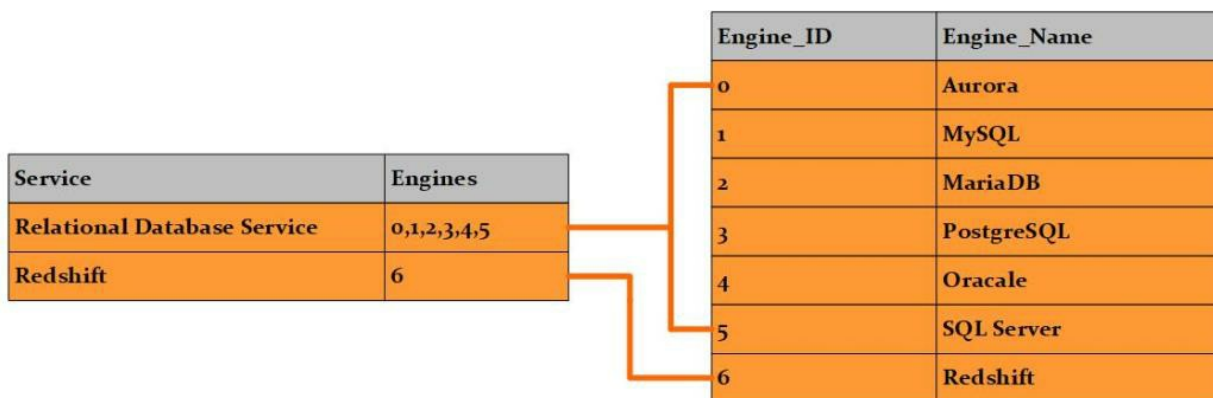
A relational database is a collection of data objects with specified relationships and can be easily retrieved. In the relational database paradigm, data structures such as data tables, indexes, and views are maintained distinct from physical storage structures, allowing database managers to alter the physical data storage without affecting the logical data structure.

Relational databases are used in the enterprise to organize data and find links between crucial data elements. They simplify managing and finding information, allowing businesses to make better-informed

decisions and save money. They work effectively with data that is structured.

When relational engines were in their heyday, data storage was expensive relative to computing power. Hence, you needed to find a way to store data a single time, which is what relational engines do. It is called data normalization. Therefore, in this example figure, you can see that you have replaced the values on the right with integers in the table on the left. This way, you only need to store this data once for each table that references it. It just needs to keep an integer to hold that relates to the other table. You can then use joins in our queries to join that data back together.

There are two types of relational engines. You have our row and column or columnar.



*Figure 3-04: Relational Database*

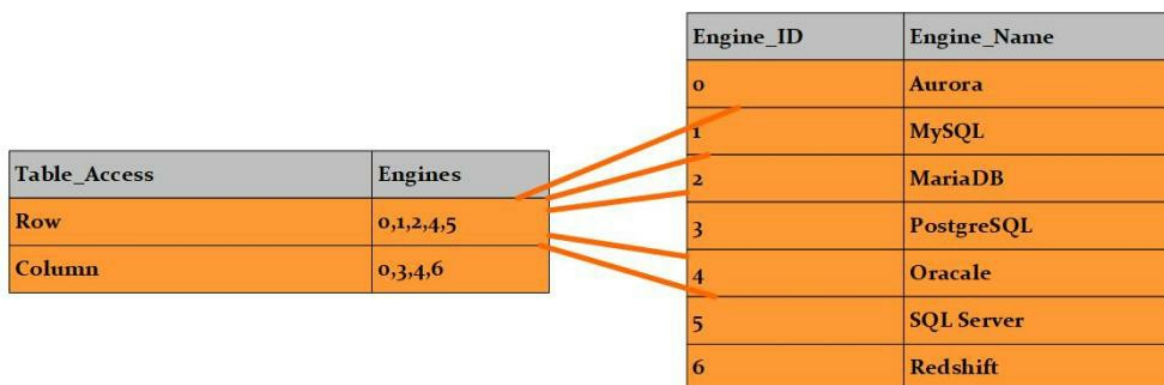
**EXAM TIP:** Relational databases are used in the enterprise to organize data and find links between crucial data elements.

## Row Databases

Row-oriented databases organize data by history and retain all of the

data associated with a record in memory adjacent to each other. Row-oriented databases are the conventional method of data organization, and they still offer some important advantages for storing data fast. They have been designed to read and write rows quickly.

The figure below shows that the row-based engines you will be working within AWS are Aurora, MySQL, MariaDB, Oracle, and SQL Server. These engines access data via rows. Hence, they will pick up a whole row at a time unless there are indexes, but you will not get into that and use those to perform our queries. It will read through the table shown below and select these rows.



*Figure 3-05: Row Databases*

## Row Database Use Cases

OLTP-Online Transaction Processing has the following features.

- Used for rapid transactions
- Protects data through transaction rollback
- Ideal for low latency applications

These engines are excellent for OLTP or online transaction processing. They are used for rapid transactions where you deal with relatively small pieces of data that will protect the data through transaction

rollback, which the columnar engines do. Typically, these are going to be used in places where those transactions are very important. If there is an error with a piece of the trade because you normalized the data, you will want to roll that transaction back and retry it. These are ideal for low latency applications like online storefronts or a simple social media site, but they are not good for data analysis. Running large queries is going to be fairly slow, and it will congest the engine. That indicates that it will conflict with the other shorter transactions if you do very large transactions. That is generally not a good thing for our application, and hence you have these columnar engines.

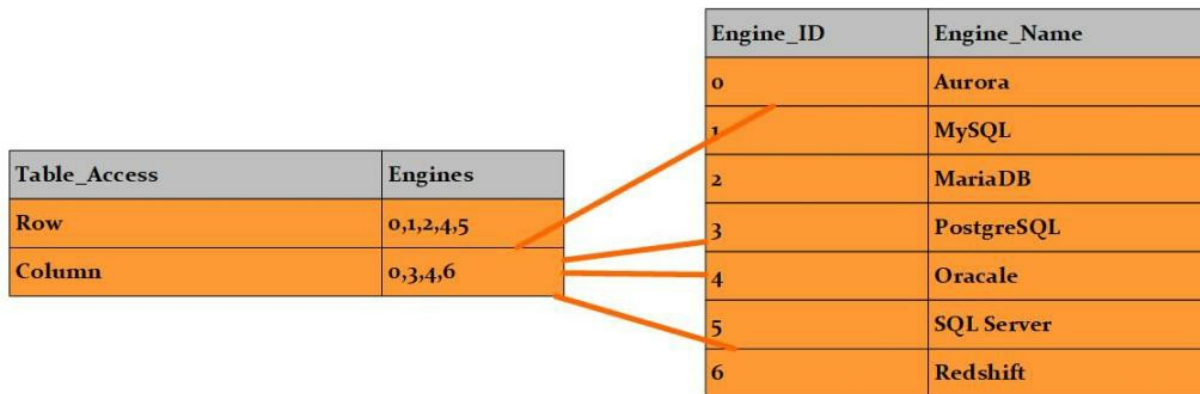
**EXAM TIP:** OLTP databases are often a data source in analytics.

## Columnar Databases

A columnar database is a Database Management System (DBMS) that stores data in columns rather than rows. A columnar database's goal is to reduce the time it takes to return a query by quickly writing and reading data to and from hard disc storage. Columnar databases store data so that disc I/O speed is considerably improved. They are very useful for data warehousing and analytics.

These engines will be Aurora Postgres, PostgreSQL, Oracle, and Redshift. Aurora has two different flavors of MySQL compatible or PostgreSQL compatible. Oracle has a row and columnar-based storage engine built into it; hence you can choose when you create your tables. Redshift is technically a data warehouse, but it does use a columnar type engine, and it is relational hence have included it here as well, and these are going to read full columns. You can use table partitioning to

shorten the data readout of the column and indexes.



*Figure 3-06: Columnar Databases*

## Columnar Database Use Cases

OLAP – Online Analytics Processing has the following features.

- Uses for analytics workloads
- Manages large amounts of data
- Handles complex long-running query operations

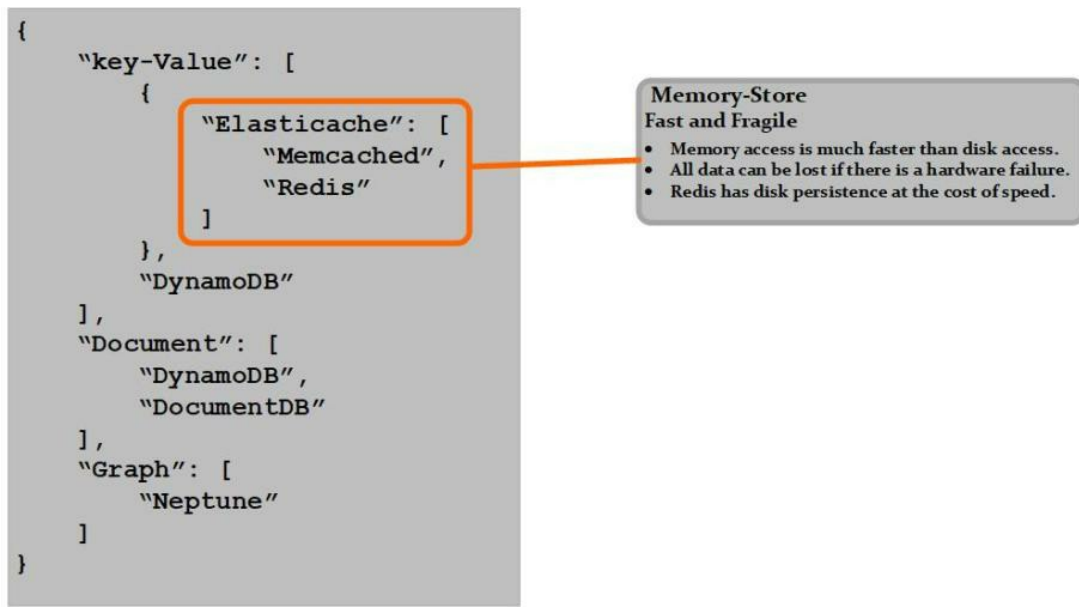
Columnar engines are good for OLAP or online analytics processing, and when you are talking about data analytics and AWS, they will be fairly important, particularly RedShift. The RedShift database is excellent at handling large amounts of data. They do well with those long-running queries that will interact with large amounts of data. You need to be aware of the row-based engines because there may be a source or even a destination for the data going into or coming out of our analytics workflow.

**EXAM TIP:** Columnar database's goal is to reduce the time it takes to return a query by quickly writing and reading data to and from hard disk storage. Columnar database engines are going to be Aurora PostgreSQL, Oracle, and Redshift.

## **Non-Relational Database**

Non-relational databases (often referred to as 'NoSQL' or 'JSON' or 'key: value' databases) differ from traditional relational databases because they are stored in a non-tabular format. Non-relational databases are substantially more adaptable than relational databases since they can digest and organize various information side-by-side. Non-relational databases, on the other hand, should be built on data formats such as documents. A document can be quite thorough while also including various data types in multiple forms.

The other variety of databases is going to be non-relational, and we will start with key-value. For the most part, our key-value databases in AWS will run on ElastiCache. You have Memcached and Redis, and these are a key and a value. They are very fast relative to other engines because all the data is stored in memory. The trade-off here is that if our ElastiCache instance goes down or is restarted, for whatever reason, you could lose all of the data that is stored there. Hence typically, this will be used as a caching point, but Redis does support disk persistence. Accordingly, you could use Redis as a primary data store in ElastiCache.

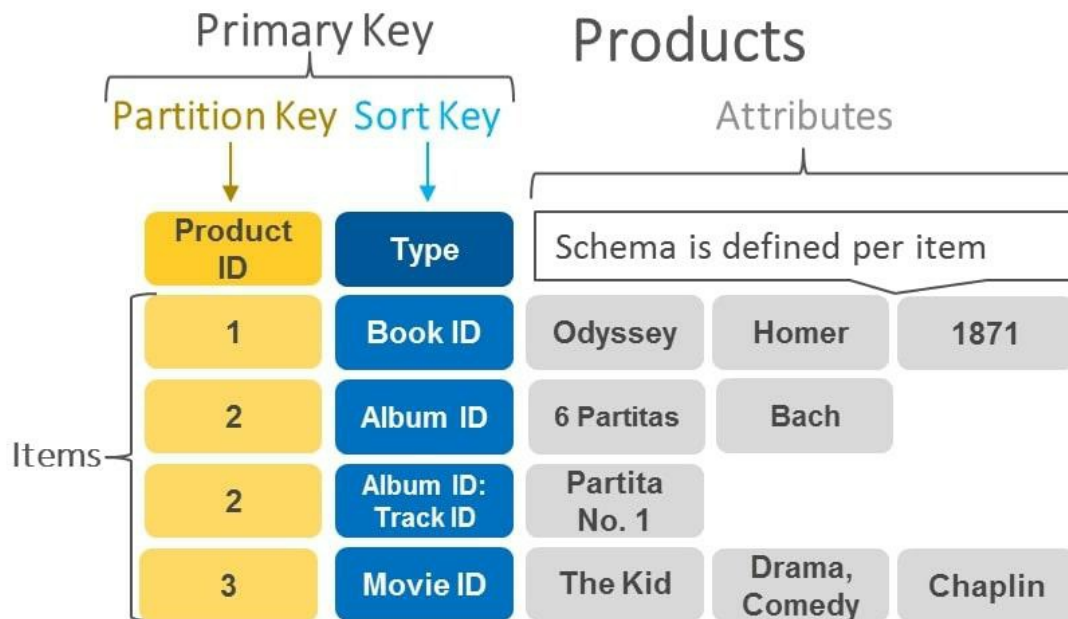


*Figure 3-07: Non-Relational Database*

## Key-Value Database

A key-value database is a non-relational database that uses a basic key-value method to store data. Data is stored in a key-value database as a collection of key-value pairs. A key serves as a unique identifier. Both keys and values can be any type of object, from basic to sophisticated compound objects. Key-value databases are extremely partitioned tables and can scale horizontally to scales that other databases cannot. Suppose a current section fills and extra storage space is necessary. In that case, Amazon DynamoDB assigns additional partitions to the database.

The graphic below displays an example of data saved in DynamoDB as key-value pairs.



*Figure 3-08: Key-Value Database*

**EXAM TIP:** A key-value database is a non-relational database that uses a basic key-value method to store data. Data is stored in a key-value database as a collection of key-value pairs.

## Key-Value Database Use Cases

- **Session Store**

When a user checks in to a session-oriented service, such as a web application, the session begins and continues until the user logs out or the session expires. The software saves all session-related data in the main memory or a database during this time. User profile information, messaging, tailored data and themes, suggestions, targeted promotions, and discounts are all examples of session data. Unique identification is assigned to each user session. A quick key-value store is better because session data is never queried by anything other than the main key. Key-

value databases, in general, may have a lower per-page overhead than relational databases.

- **Shopping Cart**

An e-commerce website may get billions of orders in seconds during the Christmas shopping season. Through distributed processing and storage, key-value databases can scale to accommodate massive quantities of data and extremely high rates of state changes while serving millions of concurrent users. Redundancy is incorporated into key-value databases to tolerate the loss of storage nodes.

## **Document Database**

A document database is a non-relational database that stores and queries data as JSON-like documents. Documents and document databases can adapt to the demands of applications due to their flexible, semi-structured, and hierarchical nature. By employing the same document-model format as their application code, document databases make it easier for developers to store and query data in a database. The document model is particularly suited to use cases where each document is unique and changes over time, such as catalogs, user profiles, and content management systems. Flexible indexing, strong ad hoc searches, and analytics over collections of documents are all possible with document databases.

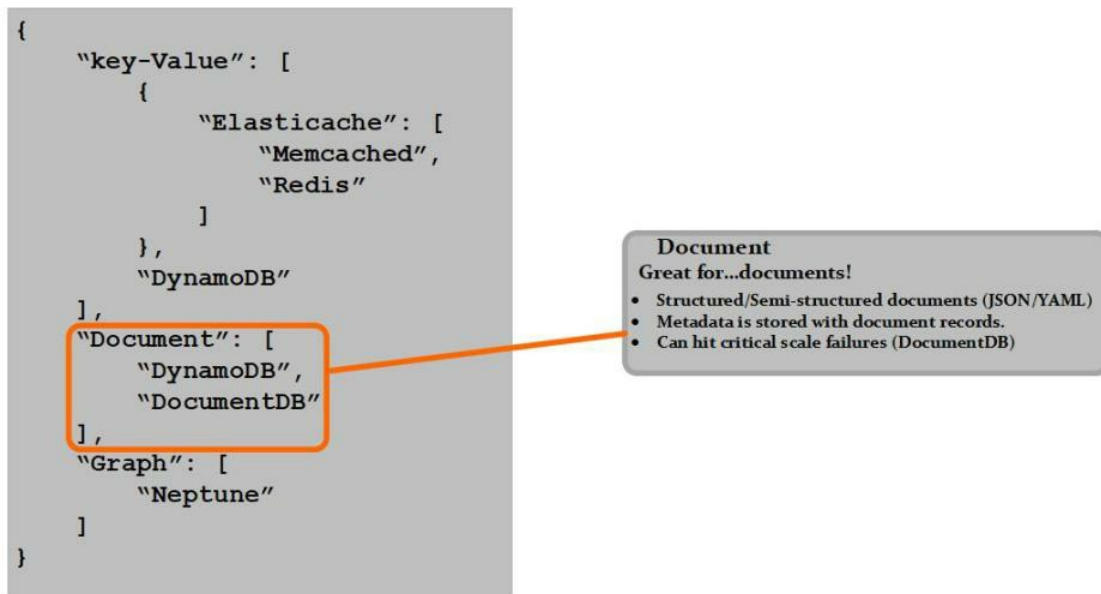
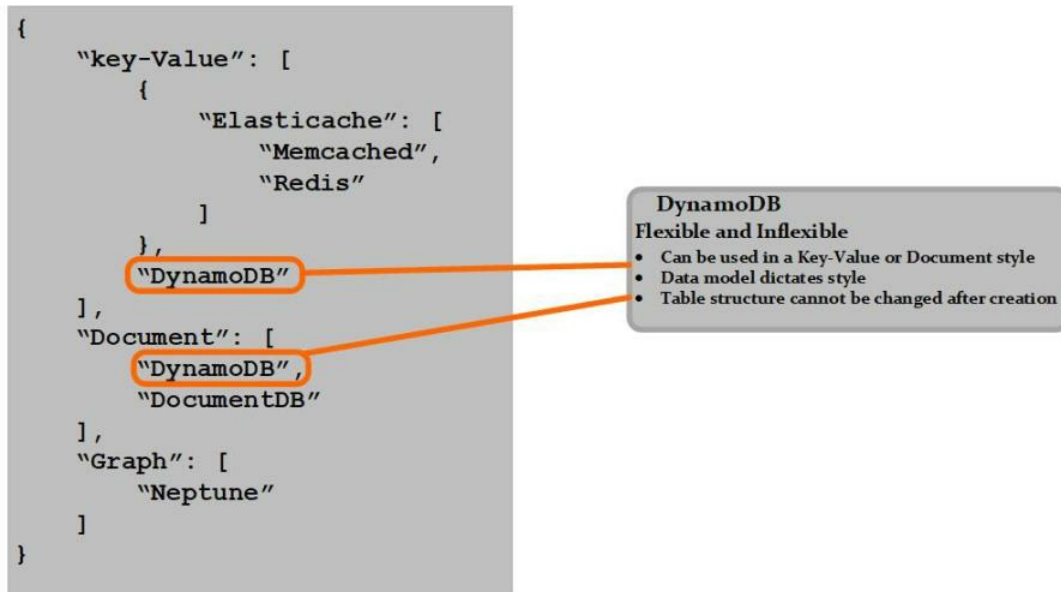


Figure 3-09: Document Database

DynamoDB bridges between our key value and the next type, the document in which DynamoDB can be used as a simple key-value. Still, you can also use it to store documents, and the data model you set up when you set up our DynamoDB table will dictate which type it is. You cannot change the table structure once the table is created. Hence, you do need to do planning when you are going to utilize DynamoDB. That planning may just be that you need to make our data model. Therefore, it can function flexibly. When we talk about the document category on its own, they are great for documents. Documents are going to be structured or semi-structured documents. Hence, it could be JSON or YAML or strings even. The metadata for those documents is going to be stored in the records. Document DB can be susceptible to scaling failures.



*Figure 3-10: DocumentDB as Key-Value*

**EXAM TIP:** A document database is a non-relational database that stores and queries data as JSON-like documents.

## Document Database Use Cases

- **Content Management**

A document database is an excellent solution for content management systems such as blogs and video platforms. The application records each entity that can be kept separate in a document database. Updating an application when it needs change is easier for a developer with a document database. Furthermore, the affected documents must be updated if the data model has to be modified. There is no need to alter the schema, and no database downtime is required to implement the modifications.

- **Catalog**

For storing catalog information, document databases are efficient and

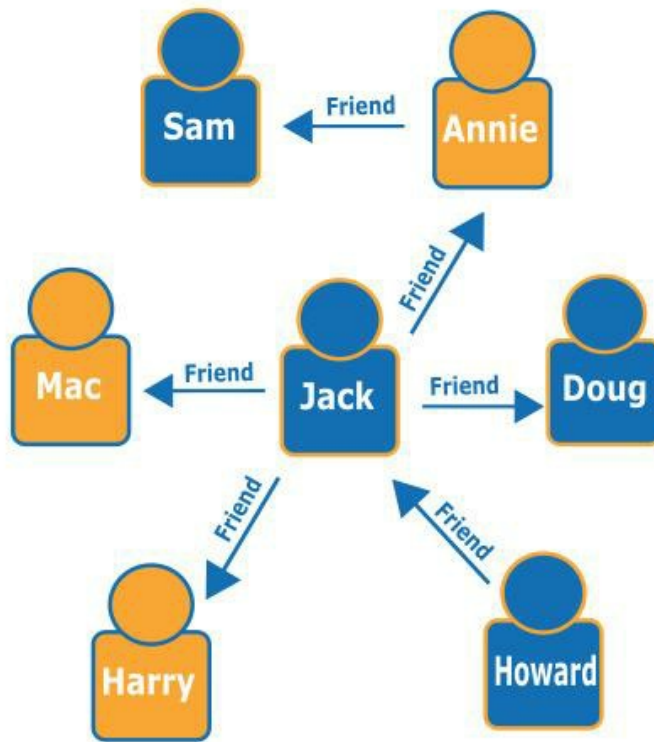
useful. For example, varied items in an e-commerce platform typically have different quantities of characteristics. In relational databases, managing hundreds of features is wasteful, and reading speed suffers. Each product's properties may be documented in a single document using a document database, making it easier to maintain and understand. Changing the characteristics of one product has no bearing on the others.

## **Graph Database**

Graph databases are especially suited for storing and traversing relationships. Relationships are treated as first-class citizens in graph databases, accounting for the vast bulk of the database's value. Nodes store data entities, while edges store relationships between things in graph databases. Edge contains a start node, an end node, a type, and a direction, and it may be used to define parent-child connections, actions, and ownership, among other things. A node can have an infinite number and variety of relationships.

You can navigate a graph along with specified edge types or over the whole graph in a graph database. The associations between nodes are not computed at query time but stored in the database, traversing the joins or relationships in graph databases. Graph databases are useful in applications like social networking, recommendation engines, and fraud detection, where it is required to construct linkages between data and query these relationships efficiently.

The graph below depicts a social network graph. Look at the people (nodes) and their relationships (edges) to see who a person's "friends of friends" are, for example, Howard's pals.

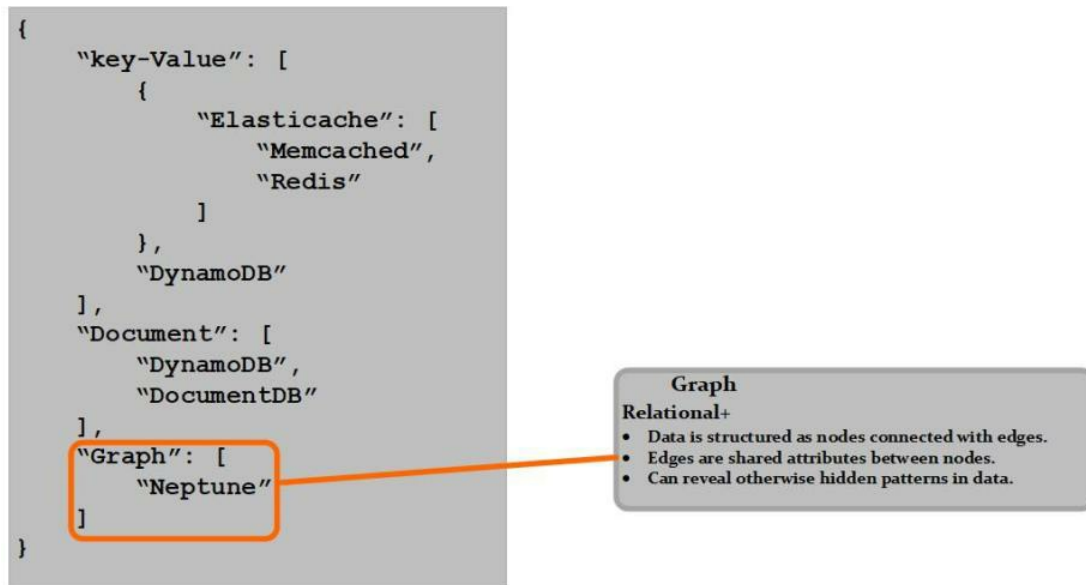


*Figure 3-11: Graph Database*

**EXAM TIP:** A graph can be navigated along with specified edge types or over the whole graph in a graph database because the associations between nodes are not computed at query time but stored in the database, traversing the joins or relationships in graph databases.

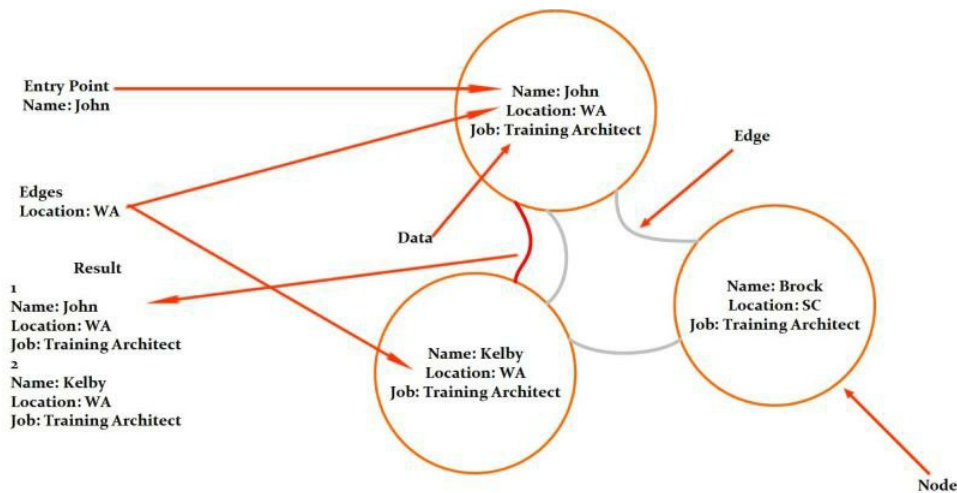
The graph database, which in AWS is Neptune graph databases are not exactly NoSQL, and they are not non-relational, but they tend to get lumped in with them. They are more relational plus or hyper relational, and the data is structured as nodes, which are connected with edges. The edges will be shared attributes between the nodes, and they are good at finding hidden patterns in data. Hence, if you are looking at the

structure of graph databases, you will have our nodes to contain data. You can label a key for the data.



*Figure 3-12: Neptune Graph Database*

Hence, for instance, as an edge, a job will create connections between our nodes. In this example, you have a location edge and a job edge, and you can see that our nodes are connected through those edges. When you query this data, you provide an entry point. You will want the node with the name value of John, and you want the location edge. Therefore, any nodes connected via location to the John node return those. If you request all of the attributes for the connected nodes, you will get in John and Kelby in return because they both live in Washington State.



*Figure 3-13: Graph Database Structure*

## Graph Database Use Cases

- **Fraud Detection**

Sophisticated fraud protection is possible with graph databases. You can leverage relationships in graph databases to conduct financial and buy transactions near-real-time. You may discover that, for example, a potential purchaser is using the same email address and payment card as a known fraud instance using quick graph queries. Numerous persons linked with a personal email account or multiple people with the same IP address but in various physical places may be readily detected using graph databases.

- **Recommendation Engines**

For recommendation applications, graph databases are an excellent solution. Relationships between information categories such as customer interests, friends, and purchase history can be recorded in graph databases. You may provide product suggestions to users based on items purchased by others who follow the same sport and have comparable purchase histories using a highly accessible graph database.

Alternatively, you may find folks who share a friend but have not met yet and offer a friendship referral.

## **Relational Database Service**

### **Introduction**

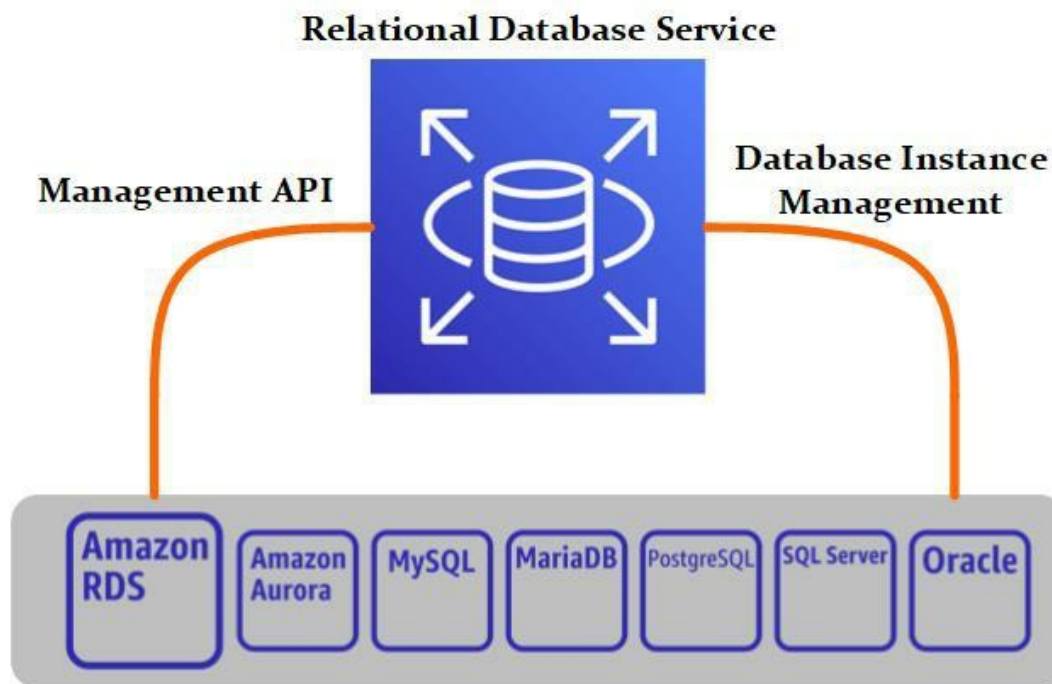
Amazon RDS (Amazon Relational Database Service) is a cloud-based managed service that makes it easier to set up, operate, and scale a relational database. It offers cost-effectiveness and scalability while handling time-consuming database management responsibilities, allowing you to focus on your applications and company.

Amazon RDS offers the functionality of a well-known MySQL, MariaDB, Oracle, SQL Server, or PostgreSQL database. It implies that the code, apps, and tools you are already using with your old databases should operate just fine with Amazon RDS. Amazon RDS can back up your database automatically and maintain your database software up to date with the newest version. You have the option of scaling the processing resources or storage space associated with your relational database instance. Furthermore, for read-heavy database workloads, Amazon RDS makes it simple to implement replication to increase database availability data durability or grow beyond the capacity restrictions of a single database instance. There are no upfront expenditures necessary, and you just pay for the resources you use, as with other Amazon Web Services.

Amazon Aurora, MySQL, MariaDB, Oracle, SQL Server, and PostgreSQL are all supported by Amazon RDS.

### **Managed Service**

RDS is a managed service. What does this mean? The basic explanation is that we do not have access to the operating system of the instances we run in these services. Hence, what we get is a management API that allows us to manage database instances. RDS runs several engines, and you have Aurora, MySQL, MariaDB, PostgreSQL, SQL Server, and Oracle and Aurora in two varieties MySQL and PostgreSQL.



*Figure 3-14: Managed Service*

Amazon RDS manages every element of establishing a relational database, from providing infrastructure capacity to installing database software. Once your database is up and running, Amazon RDS automates common administrative tasks such as backups and software upgrades. Amazon RDS also provides synchronous data replication between Availability Zones with automated failover in Multi-AZ installations.

Because Amazon RDS enables native database access, you interact with

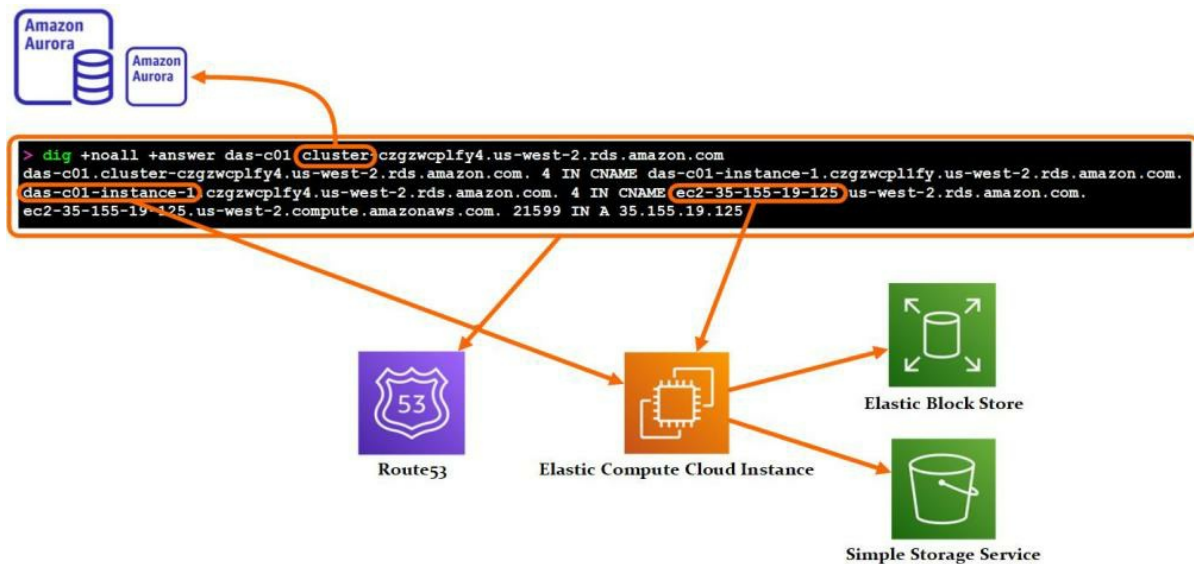
relational database applications in the same way you would normally. It means you are still in charge of managing the database settings unique to your application. You will be in control of creating the relational structure that best fits your use case and any performance tweaks to make your database more efficient workflow of your application.

**EXAM TIP:** Amazon RDS manages every element of establishing a relational database, from providing infrastructure capacity to installing database software.

## Second Level Service

RDS is considered the second level of service, and to understand that, we use the dig command. You create a database instance and run a dig against it. You can get a ton of information about RDS structure from the simple control. You can also tell that this is Aurora because Aurora is the only database engine in RDS that will provide us with a cluster endpoint. It means it connects to whatever the right node is at that time. Moving further into the command, you can see that the instance you are connecting to is an Elastic Compute Cloud or an EC2 example. You know from our EC2 knowledge that you probably have Elastic Block Store or EBS involved and that Simple Storage service is likely in the mix as well, and then the fact that Route 53 is probably involved, that is how RDS creates and manages our DNS endpoints. Hence, RDS is built from several different services. Amazon has achieved database engines on EC2 and created a service that simplifies many common use cases for databases and makes it much easier to do. The trade-off here is that you do have to pay more than you would for just an EC2

instance, but again, you get time back because you do not need to manage those databases ourselves on the EC2 examples.

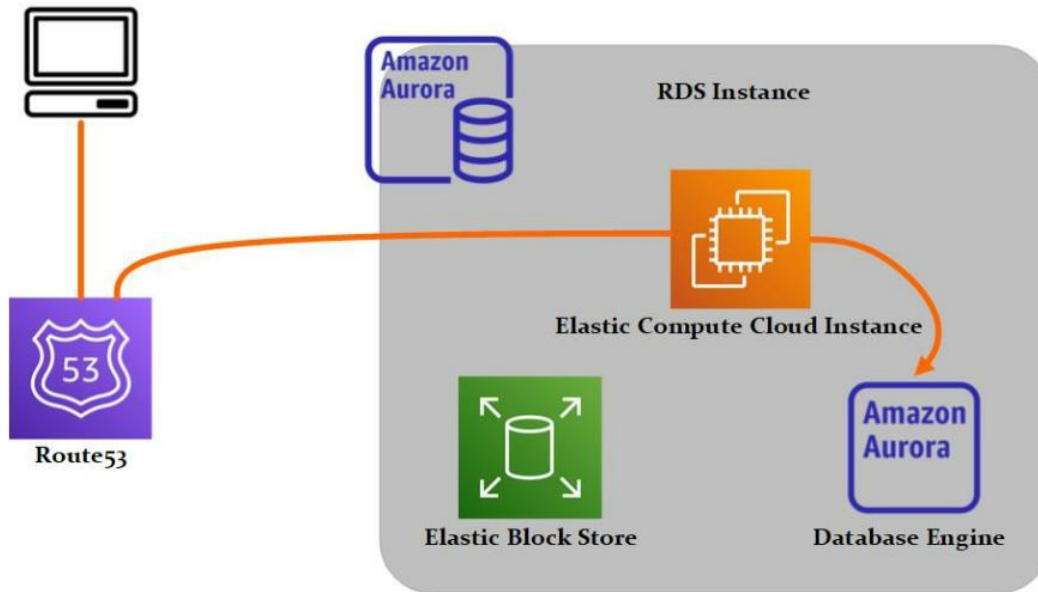


*Figure 3-15: Second Level Service*

## RDS Instance

A DB instance is a standalone database environment that runs on the cloud. It is the fundamental component of Amazon RDS. A DB instance can hold numerous user-created databases and be accessed using the same client tools and applications as a single database instance.

What is an RDS instance? You will have Route 53 hosted endpoints, which will point to our EC2 instance, which will allow us to connect to our database engine. You will likely have data stored in EBS, which provides block storage for our database engine. S3 is used for backups and snapshots, but it is not included in this graphic. It is used pretty extensively by EC2 and RDS or data storage.

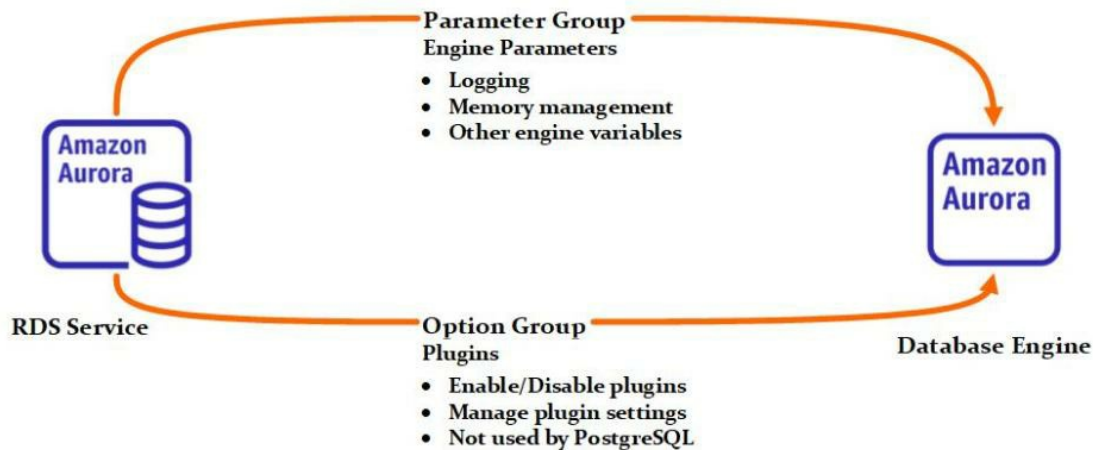


*Figure 3-16: RDS Instance*

**EXAM TIP:** A DB instance is a standalone database environment that runs on the cloud. It is the fundamental component of Amazon RDS.

## Operation System Access

How do you manage not having operating system access and still tune our database to function to our workload? You have parameter groups, which are controlled through the RDS API. You have option groups, which are also held through the RDS API parameter groups, which will allow us to change engine parameters, like how logging is configured and stored various memory management parameters and any other engine variables. Some operating system variables/option groups are used to manage plug-ins in some cases. Oracle uses option groups pretty heavily, whereas PostgreSQL does not use them.



*Figure 3-17: Operation System Access*

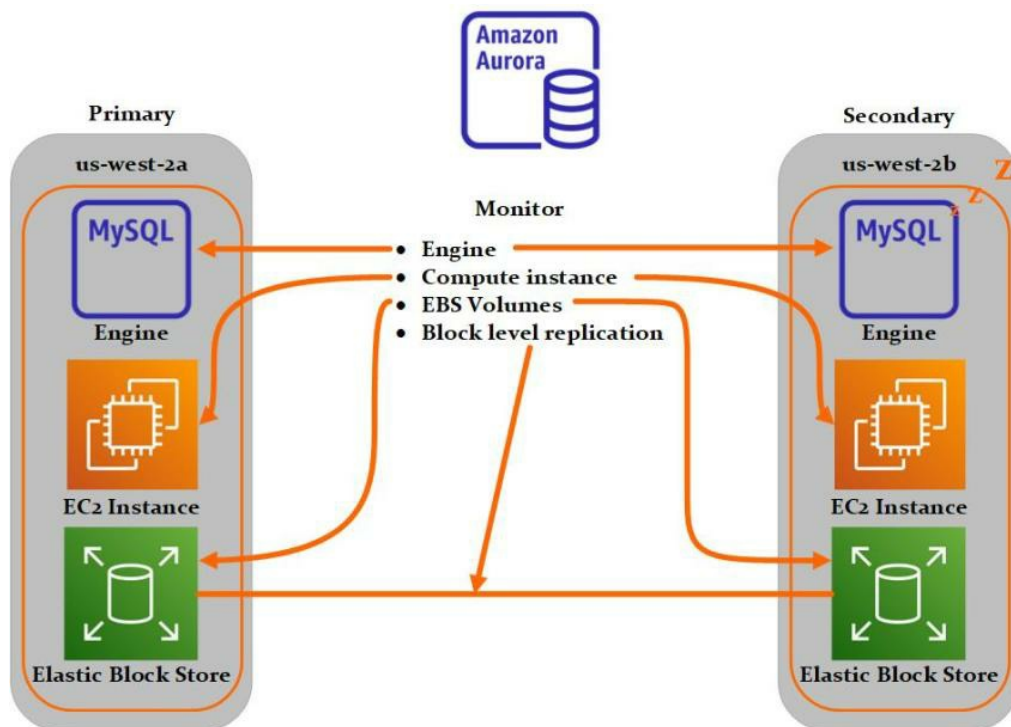
**EXAM TIP:** No operating system access in AWS RDS.

## Disaster Recovery

The big disaster recovery feature for RDS is multi-AZ deployments. The way that it works is that you are going to have a primary and secondary instance. In this example, our primary instances are in US West 2A or secondary ones in US West 2B. These instances are going to be replicating at the block level. The database engine is not involved at all; the database engine will not be running at all and on the secondary. It is going to be sleeping. RDS then monitors the engine, the compute instances, the EBS volumes, and the replication between those EBS volumes, and if the machine or the EC2 instance or the EBS volume fails, it will terminate that instance. That instance will no longer be the primary, and the secondary in US West 2B will become the primary. The RDS service will wake up the database engine, and then a new secondary in our old availability zone US West 2A will be created with an engine that is not running.

Block-level replication will then be enabled to that secondary, and you

will be in a healthy multi-AZ state. You can run RDS instances in a single AZ state, which is not recommended for production workflows for obvious reasons. This multi-AZ failover takes roughly 60 seconds most of the time, whereas a full instance replacement for a single AZ deployment can take up to five minutes or even more. Hence generally, for production workloads, multi-availability is worth it in terms of the downtime that you can experience if there is an issue with the primary instance in your deployment. You will also notice that I changed the engine to MySQL because Aurora handles this differently.



*Figure 3-18: Disaster Recovery*

**EXAM TIP:** AWS RDS works on block-level replication.

## Neptune

### Introduction

Amazon Neptune is a fast, trustworthy, and fully-managed graph database service that simplifies the design and operation of applications that deal with vast, linked datasets. Neptune is designed around a purpose-made, high-performance graph database engine. This engine is designed to store billions of relationships and query the graph in milliseconds. Neptune supports the popular graph query languages Apache TinkerPop Gremlin and W3C's SPARQL, allowing you to effectively create searches that explore densely linked datasets. Neptune drives graph application cases, including recommendation engines, fraud detection, knowledge graphs, medicine discovery, and network security.

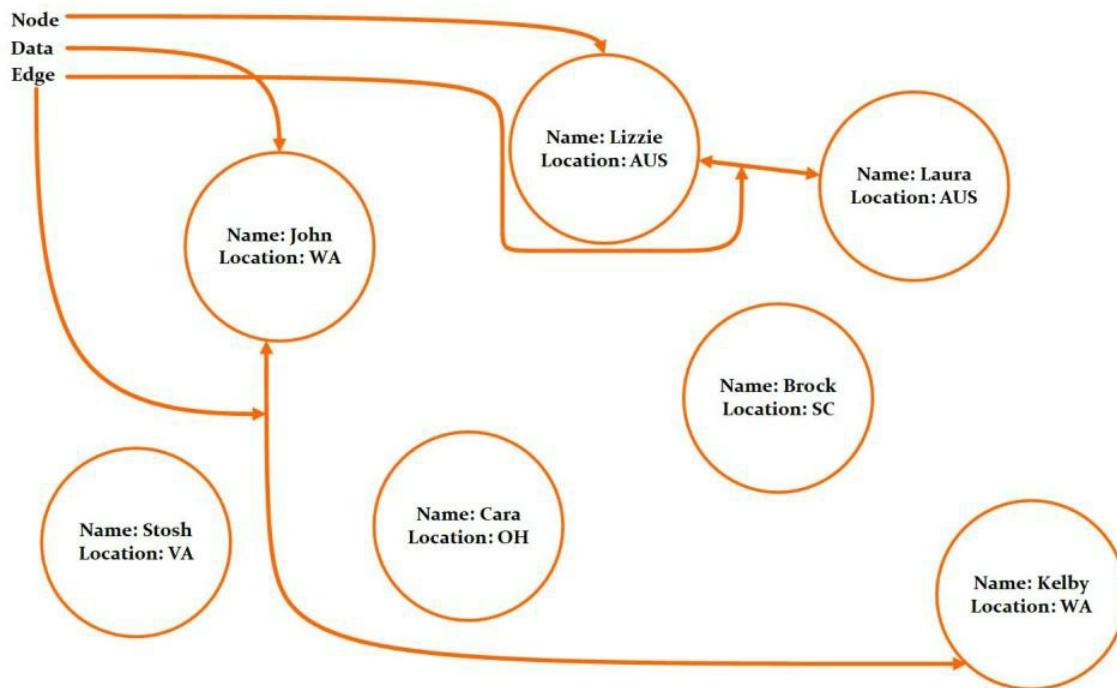
Neptune is extremely available with read replicas, point-in-time recovery, continuous backup to Amazon S3, and replication across Availability Zones. Neptune offers data security measures such as encryption at rest and in transit. Because Neptune is fully managed, you no longer need to worry about database administration activities such as hardware provisioning, software patching, setup, configuration, or backups.

## **Graph Structure**

A graph is a data structure consisting of a finite number of nodes (or vertices) and edges that link them. In the following examples, circles represent vertices, and lines indicate edges. An edge (x,y) signals that the x vertex relates to the y vertex.

We know that a graph has a node from our database engine types section. Those nodes will contain data, and then the nodes will be connected via edges. The edges are common data in those nodes. We

can traverse these graphs and collect data secured by the nodes.



*Figure 3-19: Graph Database Structure*

**EXAM TIP:** A graph is a data structure consisting of a finite number of nodes (or vertices) and edges that link them.

## Interface Languages

It is two different interface languages that Neptune supports. We have Apache's TinkerPop Gremlin, or simply Gremlin, and we have the W3C sparkle protocol, which is used with the resource description framework or RDF query language.

### 1. Apache – TinkerPop Germlin

- Graph structure property
- Interface: WebSocket
- Query Pattern: Traversal

Gremlin uses a property graph structure. The interface is a web socket, and the query pattern is traversal. Hence, Gremlin is structured very much like a namespace access pattern; G is for Gremlin. You are accessing a Vertex or node, and you want a node with Washington's location. You also care about the name, and you will traverse that place attribute to any other nodes that have the same location, and you want to return the name and location. Hence, you will get John and Kelby when you do that with the data.



Traversal

Elements: Entry, Property(s), Traversal path, Return values

```
g.V().has('name','location','WA').out('location').values('name','location')  
[name: John, location: WA]  
[name: Kelby, location: WA]
```

*Figure 3-20: Traversal Query Structure*

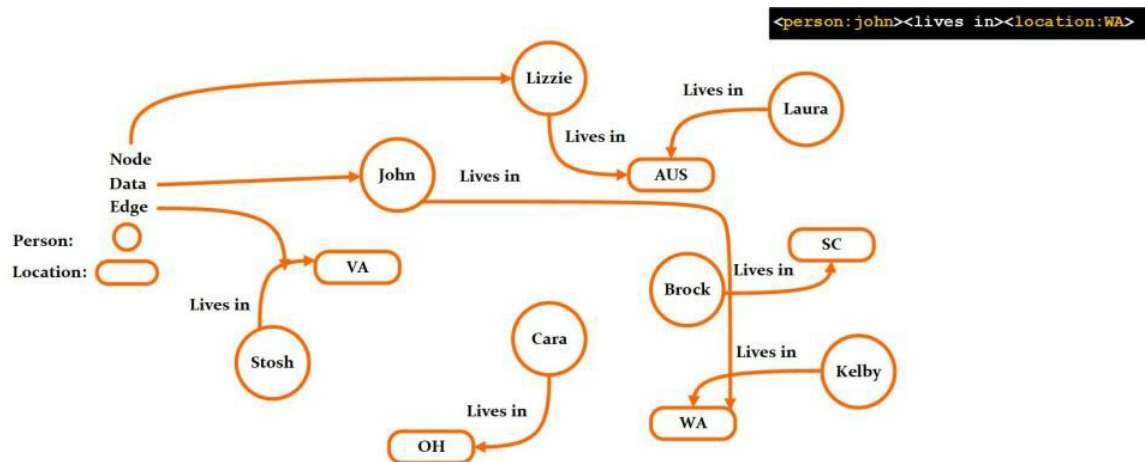
**EXAM TIP:** Gremlin uses a property graph structure. The interface is a web socket, and the query pattern is traversal.

## 2. W<sub>3</sub>C – SPARQL Protocol & RDF Query Language

- Graph structure: Resource Description Framework (RDF)
- Interface: HTTP Rest
- Query Pattern: SQL

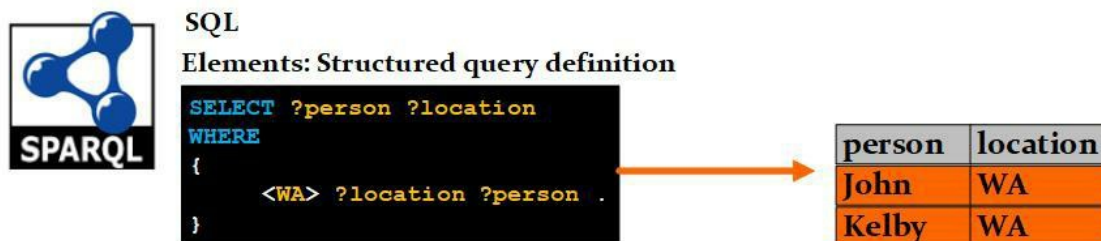
Sparkle uses the resource description framework to structure the data. The interface is a rest API, and it uses SQL to structure its queries. You have seen so far that the property style graphs resource description framework still has a node. It still has data within those nodes, and it also has edges. The difference is that you are going to describe the data as triples. Whereas you would have a node with several attributes instead, you will have lots of nodes connected by edges, and you define

those in a triple like, shown below, hence the person John lives in location, Washington. The prepended key on the data is not required, but it does help to organize that data and will assist when querying the information when you are talking about those nodes.



*Figure 3-21: RDF Graph Structure*

With sparkle, you are going to write a sequel structured query. It is going to return a table with John and Kelby. You want the person and location where the location is Washington, and you will return the person as well. You get the same results, and our traversal is entering the graph on the first node that it finds a location in Washington, then it traverses out. You see other areas that are Washington.



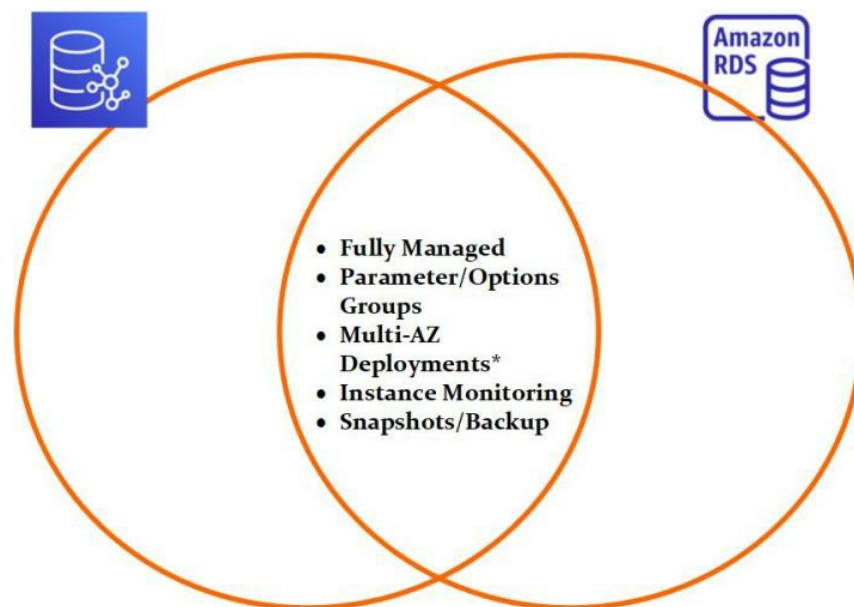
*Figure 3-22: SPARQL Query Structure*

**EXAM TIP:** Sparkle uses the resource description framework to structure the data.

---

## Comparison to Relation Database Service

When we compare Neptune to RDS, they are very similar. They are fully managed. They have parameter and option groups to change settings that you would have access to at the operating system level, and they support multi-AZ deployments. Neptune's deployments are more similar to what you would see with Aurora because there is no engine-to-engine replication. The replication happens at the storage level, and it behaves like a cluster with multiple compute nodes attached. There are similar instance monitoring tools, and you also have snapshots and backups.



*Figure 3-23: RDS VS Neptune*

**EXAM TIP:** Neptune is extremely available with read replicas, point-in-time recovery, continuous backup to Amazon S3, and replication across Availability Zones.

## Neptune Use Cases

Following are some use cases of Neptune.

- We use graphs, as they are very useful for security. They are used a lot by banks. They are used a lot by blue teams in cybersecurity. They are good at spotting patterns and analyzing patterns. Hence, you will see differences in financial transactions or our application architecture.
- They are also used quite a bit in social media because it helps us to identify connections between people and make suggestions. Hence maybe running a suggestion engine off of a graph database that can recognize patterns in profiles and fill in other pieces of those profiles that you might want to suggest to our users, or you will target advertising. You know that this person has an interest in outdoor activities. Hence you are going to target advertisements about outdoor activities to that person.
- Graph databases are also fairly heavily used in the scientific field because of their pattern recognition and analysis ability.

## DocumenDB

### Introduction

We are going to build a document about DocumentDB. DocumentDB is a document store engine. Document store engines typically are going to format their data as JSON. Hence, you are going to look at this as a JSON object.



*Figure 3-24: JSON Format Database*

Amazon DocumentDB (with MongoDB compatibility) is a highly scalable, dependable, and completely managed database service. You may run the same application code and utilize the same drivers and tools with MongoDB with Amazon DocumentDB. Amazon DocumentDB simplifies the setup, operation, and scaling of MongoDB-compatible databases in the cloud.

Some programmers may not consider their data model normalized rows and columns. Data is typically represented as a JSON document at the application tier because it is more intuitive for developers to think of their data model as a document.

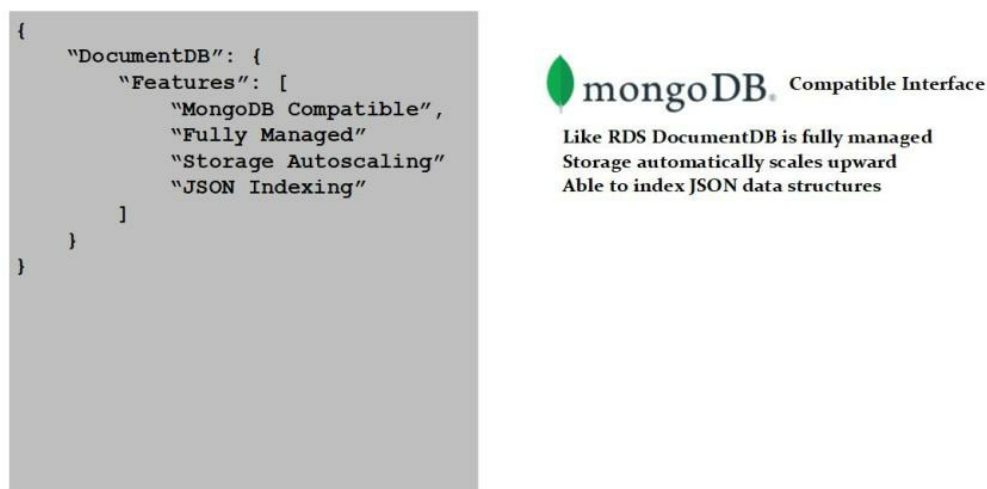
Document databases have increased in popularity because they allow you to persist data in a database using the same document model format in your application code. Document databases offer powerful and easy-to-use APIs for rapid and flexible development.

**EXAM TIP:** Amazon DocumentDB (with MongoDB compatibility) is

a highly scalable, dependable, and completely managed database service.

## DocumentDB Features

The features of DocumentDB are that it is MongoDB compatible. Hence this is a MongoDB drop-in solution for AWS for a managed MongoDB database. It is compatible. It is not MongoDB. There are a few small differences, but for the most part, you can interface with DocumentDB the way you would with a MongoDB database. It is a fully managed service like RDS. It has storage auto-scaling features. The storage will only scale upwards, which works like the Aurora storage engine. It can index JSON data structures. Hence, this is very powerful, but this is also one of the failing points of MongoDB in general in that if our data structures become too large then indexing can become very resource-intensive and can cause issues.



*Figure 3-25: AWS DocumentDB Features*

Amazon DocumentDB provides a flexible JSON document format, data types, and efficient indexing. It employs a scale-up, in-memory

optimized design to enable quick query evaluation over many documents.

Amazon DocumentDB provides a flexible JSON document format, data types, and efficient indexing, and it employs a scale-up, in-memory optimized design to enable quick query evaluation over huge amounts of documents.

When an instance fails, Amazon DocumentDB automatically fails over to one of up to 15 Amazon DocumentDB replicas you have set up in any of three Availability Zones. Suppose no Amazon DocumentDB replicas have been deployed. In that case, Amazon DocumentDB will attempt to construct a new instance for you automatically in the event of a failure.

**EXAM TIP:** It is a fully managed service like RDS. It has storage auto-scaling features. The storage will only scale upwards, which works very much like the Aurora storage engine.

## DocumentDB Use Cases

Following are some use cases of DocumentDB.

- Social media profiles are a common use place for this type of data store because they will hold all these semi-static attributes of our social media profiles.
- Object catalogs are often stored in document store databases; hence, if you are running, say, a record store and have a big database of all of the records you have ever had, this will be a great place to store those things.
- DocumentDB will work well for us. It is used in content management systems. Hence if you are writing a blog, all of our blog posts could fit neatly within documents in a MongoDB or a DocumentDB database. Anything, where we have the semi-

static documents like you, sees here will be a good use case, however, for this is not great for transactional systems where we need to track lots of small transactions; however, if you are storing data that you want to be more structured than a flat storage system like S3. It would be best to index it to return specific things based on the contents of that data.

```
{
  "DocumentDB": {
    "Features": [
      "MongoDB Compatible",
      "Fully Managed"
      "Storage Autoscaling"
      "JSON Indexing"
    ]
    "Use Cases": [
      "Social Media Profiles",
      "Object Catalogues",
      "Content Management Systems"
    ]
  }
}
```

*Figure 3-26: AWS DocumentDB Use Cases*

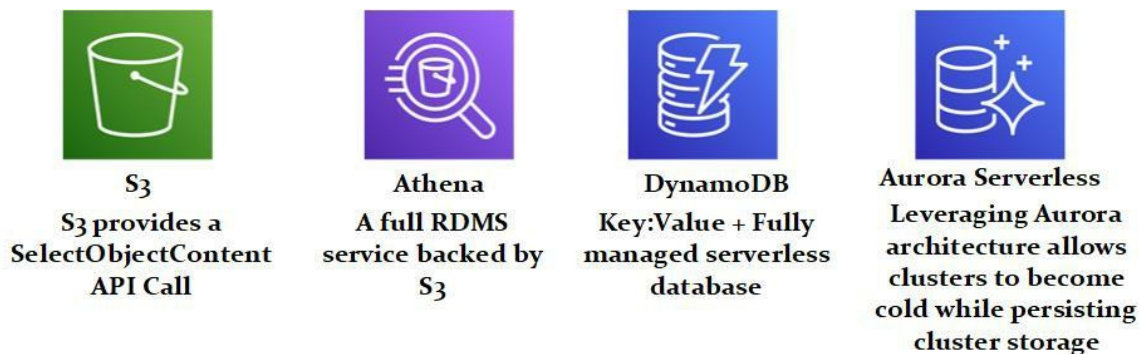
## Serverless Options

### Introduction

Serverless is a means of offering backend services as needed. Users may use a serverless provider to build and publish code without worrying about the underlying infrastructure. Because the service is auto-scaling, a firm that obtains backend services from a serverless vendor is charged depending on their calculations and does not have to reserve and pay for a predetermined amount of bandwidth or number of servers. It

should be noted that, despite the term, actual servers are still utilized, but developers are not required to be aware of them.

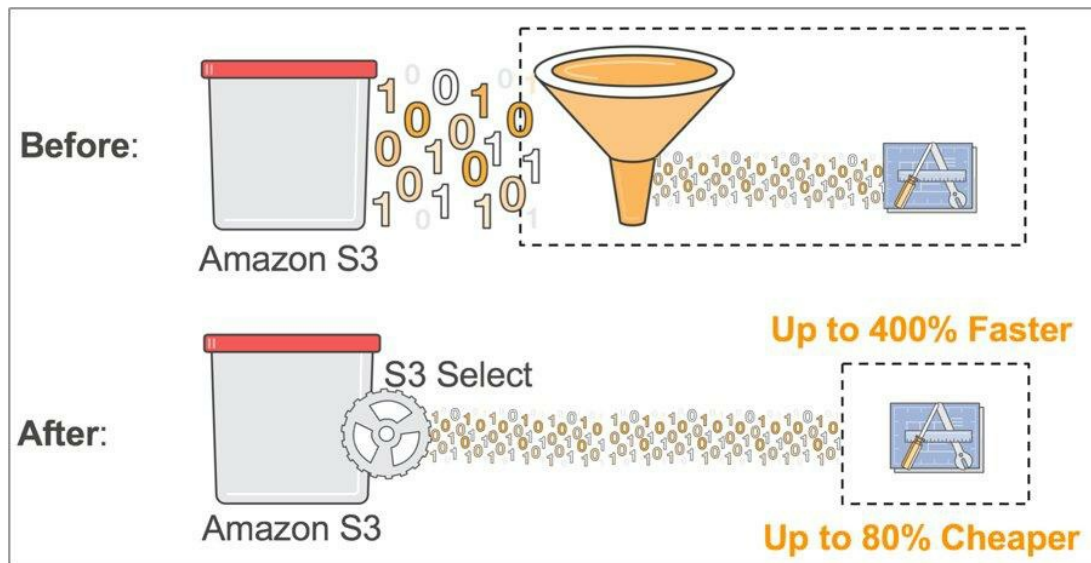
To start, we have S3 select, which is just an API call that lets us make selects against data in S3 buckets. We have Athena, which is more of a fully relational database management system that S3 backs. We have DynamoDB, a key-value plus a fully managed serverless database. We also have Aurora serverless, which leverages the architecture of Aurora to treat our database as though it is serverless. Serverless means that we do not need to administrate any servers. Hence there are still servers behind all of these services. We are just not going to be building them at all.



*Figure 3-27: Database Serverless Options*

## S3 Select

Using simple SQL expressions, S3 Select allows apps to get only a subset of data from an object. You may drastically improve speed by retrieving only the data required by your application using S3 Select. In many circumstances, you might expect to see a 400% improvement.

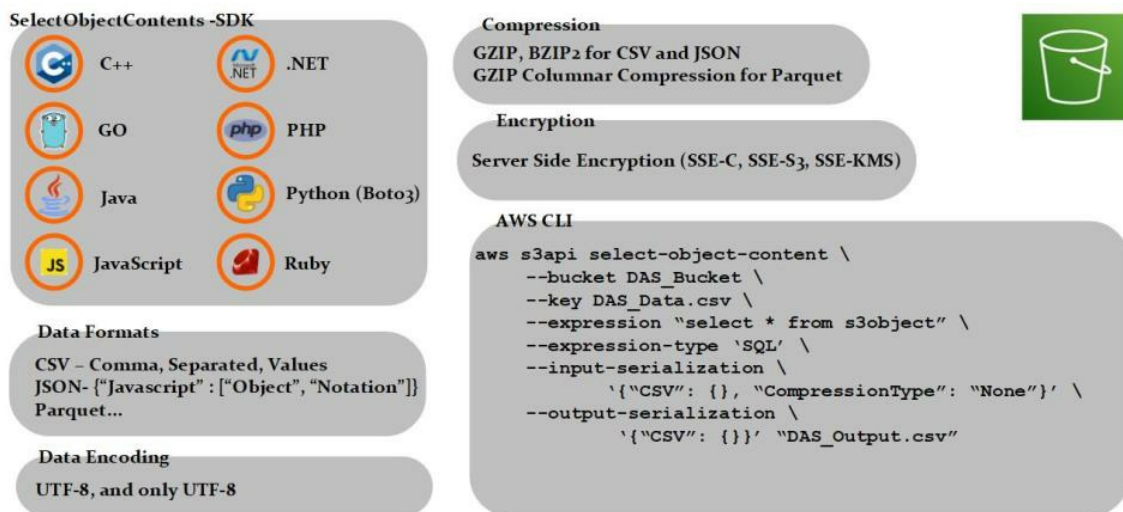


*Figure 3-28: S3 vs. S3 Select*

Assume you are a developer at a huge retailer that needs to evaluate weekly sales data from a single location, but the data for all 200 stores is stored in a fresh GZIP CSV every day. Instead of accessing the full object, S3 Select allows you to use a simple SQL phrase to return only the data from the store you are interested in. To acquire the data you wanted without S3 Select, you would have to download, decompress, and analyze the complete CSV. It implies that you are dealing with orders of magnitude fewer data, which increases the performance of your underlying applications.

S3 Select is available in the AWS CLI and SDKs C++, Go, Java, Javascript, .net, PHP, Python, and Ruby. It can read data from CSV or comma, separated value, JSON or JavaScript, object notation, formatted data, or Parquet. Hence Parquet formatted data is accepted by S3 select. The data must be encoded in UTF-8. Gzip and Bzip two compression is available for CSV and JSON, and Parquet is a fairly complex data format. Hence, it only supports columnar compression within that

format. It is fairly interesting if you are into data formats. Encryption is supported on the server-side. Client-side encryption would not work for S3 Select, and if you are doing a select from the AWS CLI, you use the S3 API command set because you are asking S3 to do something you are not moving or managing data. Hence this is a quick example of what that would look like if you had a `das_data` CSV. You wanted to select everything out of that object that needs to provide an expression and expression type, and then the inputs realization and the output serialization that you want. You can change the output serialization from one to the other if you wish to.



*Figure 3-29: S3 Select*

**EXAM TIP:** S3 Select is available in the AWS CLI and SDKs C++, GO, Java, Javascript, .net, PHP, Python, and Ruby. It can read data from CSV or comma, separated value, JSON or JavaScript, object notation, formatted data, or Parquet.

## Athena

Amazon Athena is an interactive query service that allows you to use conventional SQL to evaluate data directly in Amazon Simple Storage Service (Amazon S3). With a few clicks in the AWS Management Console, you can aim Athena at your Amazon S3 data and start running ad-hoc searches with conventional SQL to obtain results in seconds.

Because Athena is serverless, there is no infrastructure to install or administer. You just pay for the queries you conduct. Athena intelligently scales searches in parallel, resulting in rapid responses even with big datasets and sophisticated questions.

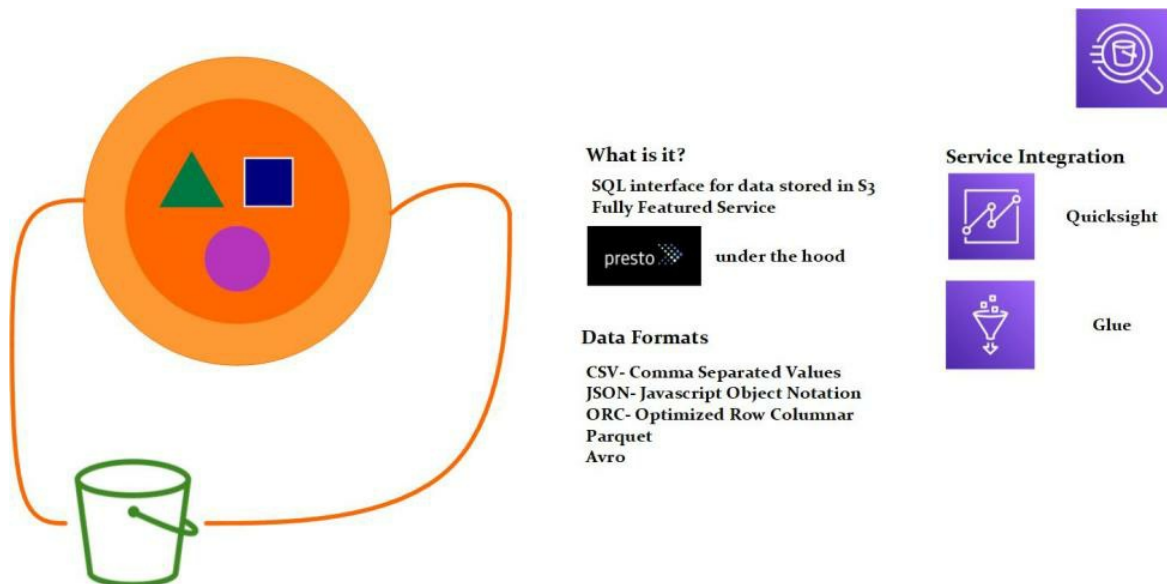
Athena aids you in analyzing unstructured, semi-structured, or structured data stored on Amazon S3. Examples are CSV, JSON, and columnar data formats such as Apache Parquet and Apache ORC. You can use Athena to conduct ad-hoc ANSI SQL queries without aggregating or loading the data into Athena.

For quick data visualization, Athena works with Amazon QuickSight. Athena may produce reports or study data using business intelligence tools or SQL clients linked through a JDBC or an ODBC driver.

Athena works with the AWS Glue Data Catalog, which provides a permanent metadata repository on Amazon S3 for your data. It lets you construct tables and query data in Athena based on a centralized metadata store available throughout your Amazon Web Services account and connected with AWS Glue's ETL and data discovery functionalities.

Athena is very much like S3 Select but with more features. Hence, it is a SQL interface for data stored in S3. It is a full service all into itself. You do not access it through S3 or the S3 API. It has its API running Presto

under the hood, and it accepts CSV, JSON ORC, or optimized row columnar Parquet. It is Parquet Avro, and you will notice that ORC and Avro are additional data formats that Athena can read that S3 select cannot, depending on the type and structure of your data. If you want data to make more complex queries against or format differently than you would use for S3 Select, Athena will fit into that space. Athena has some very useful service integrations. When you are looking at the back end of our data analytics workflow, it will integrate well with QuickSight, and it also can be cataloged with Glue, which lets us do some fairly complex ETL work with Athena of our data sources.



*Figure 2-30: AWS Athena*

**EXAM TIP:** Amazon Athena is an interactive query service that allows you to use conventional SQL to evaluate data directly in Amazon Simple Storage Service (Amazon S3).

## DynamoDB

Amazon DynamoDB is a fully managed NoSQL database service that delivers quick and predictable performance while seamless scaling. You can offload the administrative requirements of running and growing a distributed database using DynamoDB, so you do not have to worry about hardware provisioning, setup, configuration, replication, software patching, or cluster scalability. DynamoDB also supports encryption at rest, removing the operational load and complexity associated with securing sensitive data.

You may use DynamoDB to design database tables that store and retrieve any quantity of data while also serving any degree of request volume. You may increase or decrease the throughput capacity of your tables without experiencing downtime or performance reduction. The AWS Management Console may be used to track resource use and performance data.

DynamoDB allows for on-demand backups. It enables you to build comprehensive backups of your tables for long-term retention and archival to meet regulatory compliance requirements.

For your Amazon DynamoDB tables, you may make on-demand backups and allow point-in-time recovery. You may restore a table to any point in time during the previous 35 days using point-in-time recovery. Point-in-time recovery safeguards your tables against erroneous write or deletes operations.


DynamoDB allows you to automatically remove expired items from tables to save storage use and the expense of maintaining data that is no longer relevant.

Hence DynamoDB is a key-value and document store. The table looks

fairly similar to this in the console, where you will have attributes and keys for those attributes that are accessible via an API. If you store documents and want to access their values, you could just store flat JSON in it through the API. However, you would not be able to make queries against our JSON object's sub-attribute. Hence, you can use a map to see an M here in the metadata. That is for the map, not metadata that specifies the data format. Therefore, you are making a map. You then have a use cases key here, and then you have a list data format designator, which will contain strings. Hence our use cases are content management, metadata store, user profiles, transaction logging, and things like that. It is really good for OLTP use cases, and it is a bit more flexible in terms of the rapidness of transactions than, say, DocumentDB or MongoDB. DynamoDB has several data formats available for the values in its fields. Hence, it does have local secondary indexes, which adds a secondary key to the table. You can add a global secondary index, like a second key structure for the table. Hence you would query against that global secondary index the same way you would against the primary index. Another very useful feature stream, which is a stream of the data coming into the table, any changes, deletions, writes, subsequently that you can use to trigger Lambda functions and perform secondary functions based on the activity on our table.

DynamoDB has several other features, but these are the ones that are good to know about. It has a couple of closely integrated services, Dax, a DynamoDB accelerator, a read cache for DynamoDB, and fairly granular IAM integration. You can get down to the single record level for IAM access, and as mentioned with streams, Lambda is fairly closely

integrated with DynamoDB. Then you have a fun little record here that lets you decode that this is a byte data field, which for DynamoDB is going to be base 64 encoded.



| Partition        | Sort               | Access | Metadata  |
|------------------|--------------------|--------|---|
| Engine: DynamoDB | Key/Value&Document | API    | <pre> "M": {   "Use Cases": {     "L": [       {"s": "Content Management"},       {"s": "Metadata Store"},       {"s": "User Profile"},       {"s": "Transaction Logging"}     ]   },   "Features": {     "SS": [ "Local Secondary Index", "Global Secondary Index", "Stream" ],     "Integrated Services": {       "SS": ["DAX", "IAM", "Lambda"]     },     "Fun": {"B": "SGVsbG8g02xvdw0gR3VvdXMh"}   } } </pre> |

*Figure 3-31: AWS DynamoDB*

**EXAM TIP:** Amazon DynamoDB is a fully managed NoSQL database service that delivers quick and predictable performance while seamless scaling. DynamoDB allows you to automatically remove expired items from tables to save storage use and the expense of maintaining data that is no longer relevant.

## Aurora

Amazon Aurora (Aurora) is a relational database engine that is fully managed and compatible with MySQL and PostgreSQL. MySQL and PostgreSQL combine the performance and dependability of high-end commercial databases with the simplicity and low cost of open-source databases. The code, tools, and applications that you are now using with your existing MySQL and PostgreSQL databases may be utilized

with Aurora. Aurora may give up to five times the performance of MySQL and three times the throughput of PostgreSQL in specific workloads without needing modifications to the bulk of your existing applications.

Aurora is equipped with a high-performance storage subsystem. Its MySQL and PostgreSQL database engines have been tailored to use fast distributed storage. As needed, the underlying storage expands automatically. Aurora also automates and standardizes database clustering and replication, two of the most difficult parts of database deployment and maintenance. An Aurora cluster volume can be up to 128 terabytes in size (TiB).

Aurora is a managed database service provided by Amazon Relational Database Service (Amazon RDS). Amazon RDS is a cloud-based online service that simplifies the setup, administration, and scale the relational databases.

The following comparisons show how Aurora compares to the typical MySQL and PostgreSQL engines offered in Amazon RDS:

- When configuring new database servers with Amazon RDS, you select Aurora as a DB engine option
- Aurora uses the well-known Amazon Relational Database Service (Amazon RDS) functionalities for management and administration. Aurora manages typical database processes such as provisioning, patching, backup, recovery, failure detection, and repair using the Amazon RDS AWS Management Console interface, AWS CLI commands, and API operations
- Instead of individual database instances, Aurora administration activities often include whole clusters of database servers that are synchronized via replication. The automated clustering, replication, and storage allocation make setting up, operating,

and scaling your biggest MySQL and PostgreSQL systems straightforward and cost-effective

- You may import data from Amazon RDS for MySQL and Amazon RDS for PostgreSQL into Aurora by taking and restoring snapshots or using one-way duplication. You may convert your current Amazon RDS for MySQL and Amazon RDS for PostgreSQL apps to Aurora using push-button conversion tools

**EXAM TIP:** Aurora is equipped with a high-performance storage subsystem. It is MySQL, and PostgreSQL database engines have been tailored to use fast distributed storage.

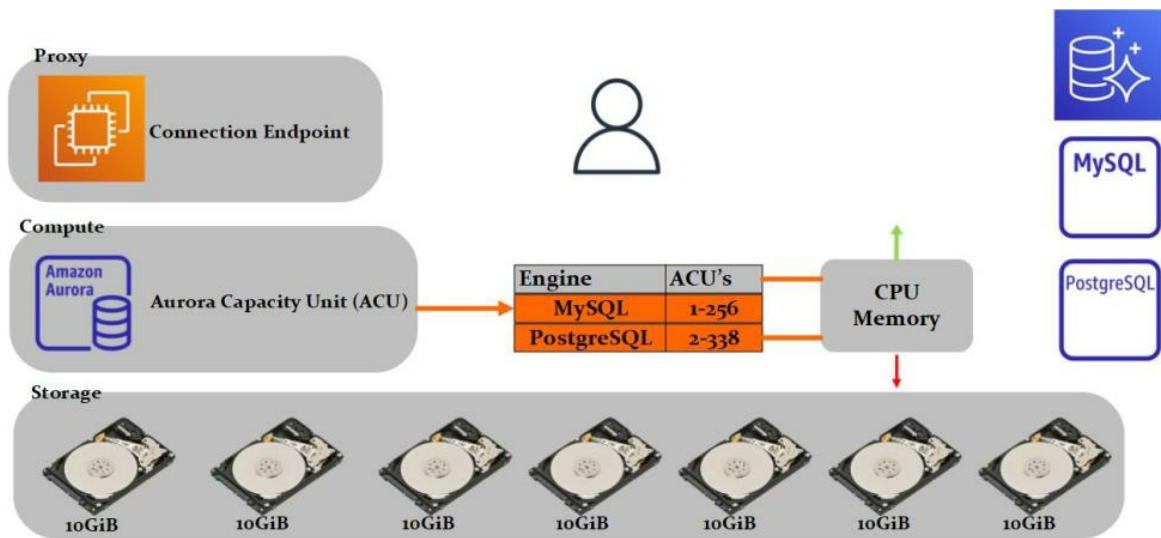
## Aurora Serverless

Amazon Aurora Serverless v1 (Amazon Aurora Serverless version 1) is a configuration for on-demand autoscaling in Amazon Aurora. An Aurora Serverless DB cluster is a database cluster that dynamically scales processing capacity based on the needs of your application. In contrast, Aurora supplied DB clusters require manual capacity management. Aurora Serverless v1 is a simple, low-cost choice for occasional, intermittent, or unexpected workloads. It saves money since it starts up automatically, boosts processing capacity to match the needs of your application, and shuts down when not in use.

The same high-capacity, distributed, Aurora Serverless v1 clusters use highly available storage volume by deployed DB clusters. An Aurora Serverless v1 cluster's cluster volume is always encrypted. You may select the encryption key, but you cannot turn off encryption. You can currently conduct the same activities on an Aurora Serverless v1 that you do on encrypted snapshots.

We have Aurora serverless. Aurora Serverless can run a MySQL compatible or PostgreSQL compatible version. A proxy in front of our database handles our incoming connections. There are compute instances between that proxy and a storage cluster. As we use that storage, it will add ten gibibyte chunks as you go along.

Hence, Aurora has no hard IOPS limit because the data is stored in these chunks, individual nodes within the storage cluster. Therefore, each one of those would be very difficult. It would not be easy to exceed the IOPS capability of these storage nodes. Hence, one of the best features of any Aurora flavor is that the storage engine provides this unlimited pool of IOPS. You pay for those IOPS, but you will not have production down events because someone provisioned a GP2 volume for a production system. As mentioned, the storage and compute are connected. Still, the compute node can go into a zero-allocation state, which is why it should be used for infrequently accessing data but say, our user comes along and requests a connection. It will go to that compute node and provision the compute node standing by it. You will load our configuration onto it, and now you can make queries against our database. Thus, you may have noticed the work capacity unit. It allows the instance to scale up and down to demand against the database. Hence MySQL can go from 1 to 256 Aurora capacity units, and PostgreSQL can go from 2 to 384 or our capacity units.



*Figure 3-32: AWS Aurora Serverless*

**EXAM TIP:** Aurora Serverless DB cluster is a database cluster that dynamically scales processing capacity based on the needs of your application. In contrast, Aurora supplied DB clusters require manual capacity management.

## Lab 3-01: Programmatically Utilizing S3 Select

### Introduction

#### AWS Simple Storage Service (S3)

Amazon S3 is a type of object storage that allows you to store and recover any quantity of data from any location. It is a low-cost storage solution with business resilience, reliability, efficiency, privacy, and infinite expansion.

Amazon S3 offers a straightforward web service interface for storing and retrieving any amount of data from any location at any time. You may quickly create projects that integrate cloud-native storage using this

service. Because Amazon S3 is easily customizable and you only pay for what you use, you can start small and scale up as needed without sacrificing performance or dependability.

Amazon S3 is also built to be highly adaptable. Instead of finding out how to store their data, Amazon S3 allows developers to focus on innovation. Build a simple FTP system or a complex web application like the Amazon.com retail website; read the same piece of data a million times or only for emergency disaster recovery; store whatever type and amount of data you desire.

### **AWS S3 Select**

S3 Select is an Amazon S3 feature that allows you to access a selection of S3 object content rather than the complete object by using simple SQL queries. SQL clauses such as SELECT and WHERE can get data from objects saved in CSV, JSON, or Apache Parquet formats. It also supports GZIP or BZIP2 compressed objects (CSV and JSON files) and server-side encrypted data.

### **AWS Lambda**

AWS Lambda allows you to run code without creating or managing servers. There is no charge when your code is not executing; you only pay for the compute time you use. You can run code for nearly any application or backend service with Lambda, and you do not have to worry about administration. Upload your code, and Lambda will handle everything necessary to run and grow it with high availability. You may configure your code to be automatically triggered by other AWS services, or you can access it directly from any computer or smartphone app.

## **AWS API Gateway**

Amazon API Gateway is a fully managed service that allows developers to publish easily, maintain, monitor, and protect APIs of any size. You can develop an API that serves as a "front door" for apps to access data, business logic, or functionality from your backend services, such as those operating on your server Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Container Service (Amazon ECS), or AWS Elastic Beanstalk, code running on AWS Lambda, or any web application, with a few clicks in the AWS Management Console. Accepting and processing hundreds of thousands of concurrent API calls and traffic management, authorization, access control, monitoring, and API version management are all supported by Amazon API Gateway. There are no minimum fees or initial charges with Amazon API Gateway. You only pay for the API requests you receive and the amount of data transmitted out when using HTTP APIs and REST APIs. You only pay for messages delivered and received, as well as the time a user/device is connected to the WebSocket API.

## **Problem**

Assume you are a Data Analyst in a company. Your company's development team is working on a proof of concept for a directory of all of the company's workers. They have a pretty simple web application running, but they want to add a filtering function. So how can you automate this task and add a data filtering feature to the website?

## **Solution**

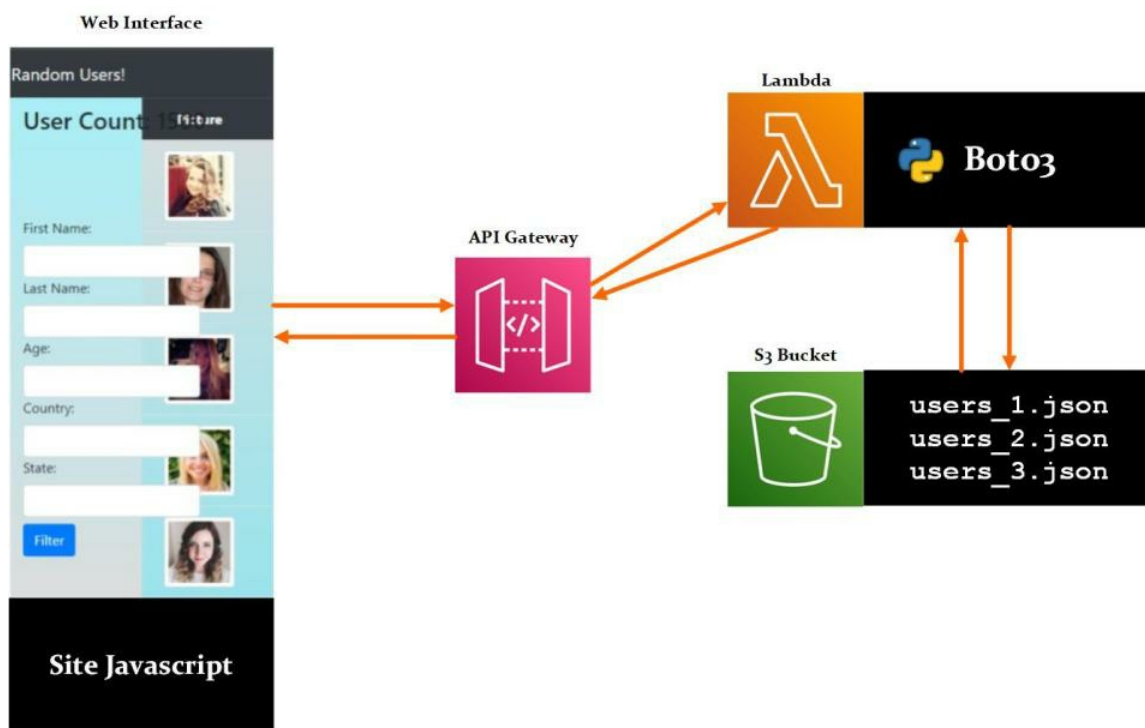
The solution is simple to design a filtering pipeline using AWS Lambda functions and API Gateway. The main service you use to create a

database is on S3. Then, you use S3 Select, an Amazon S3 feature that allows you to access a selection of S3 object content rather than the complete object by using simple SQL queries.

**Note:** Before starting the lab, make sure to create an S3 bucket and upload the related files used in this lab, as is provided in the following Github link:

[https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/tree/master/Lab\\_Assets/programmatically\\_utilizing\\_s3\\_select](https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/tree/master/Lab_Assets/programmatically_utilizing_s3_select)

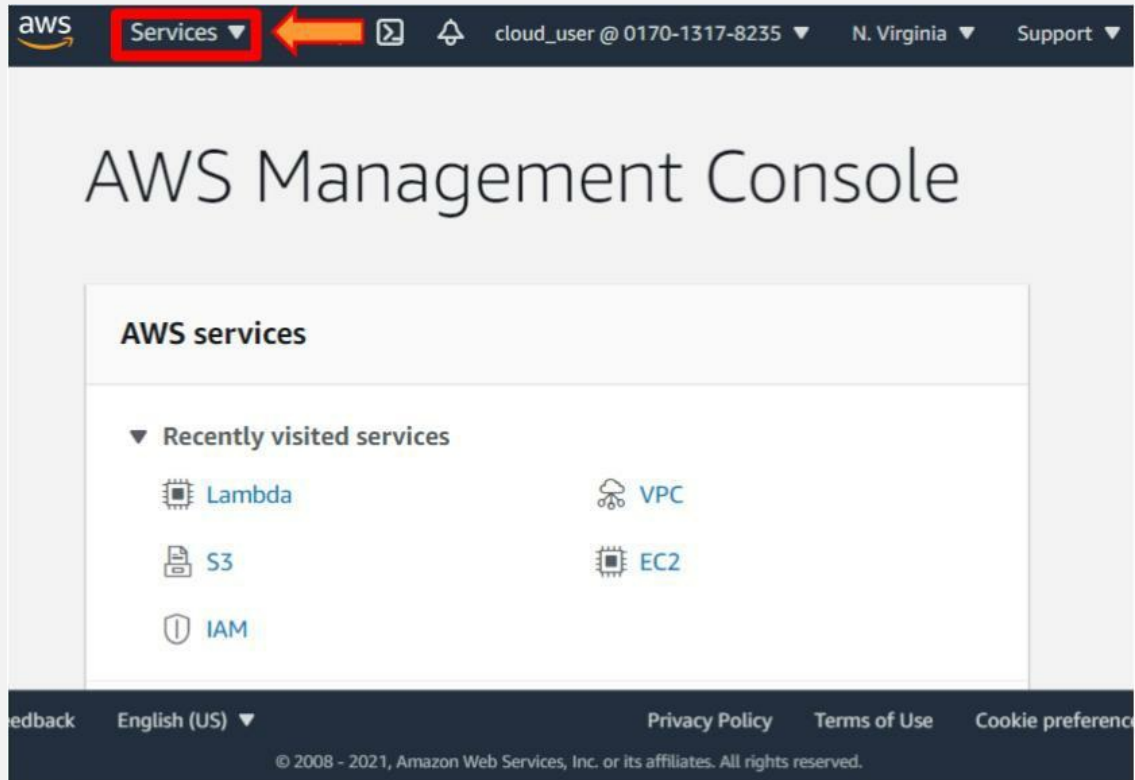
Also, create lambda functions as this is used in this lab environment. Making all these resources is mentioned in the IPSpecialist AWS Certified Solutions Architect – Associate (SAA-Co2) coursebook.



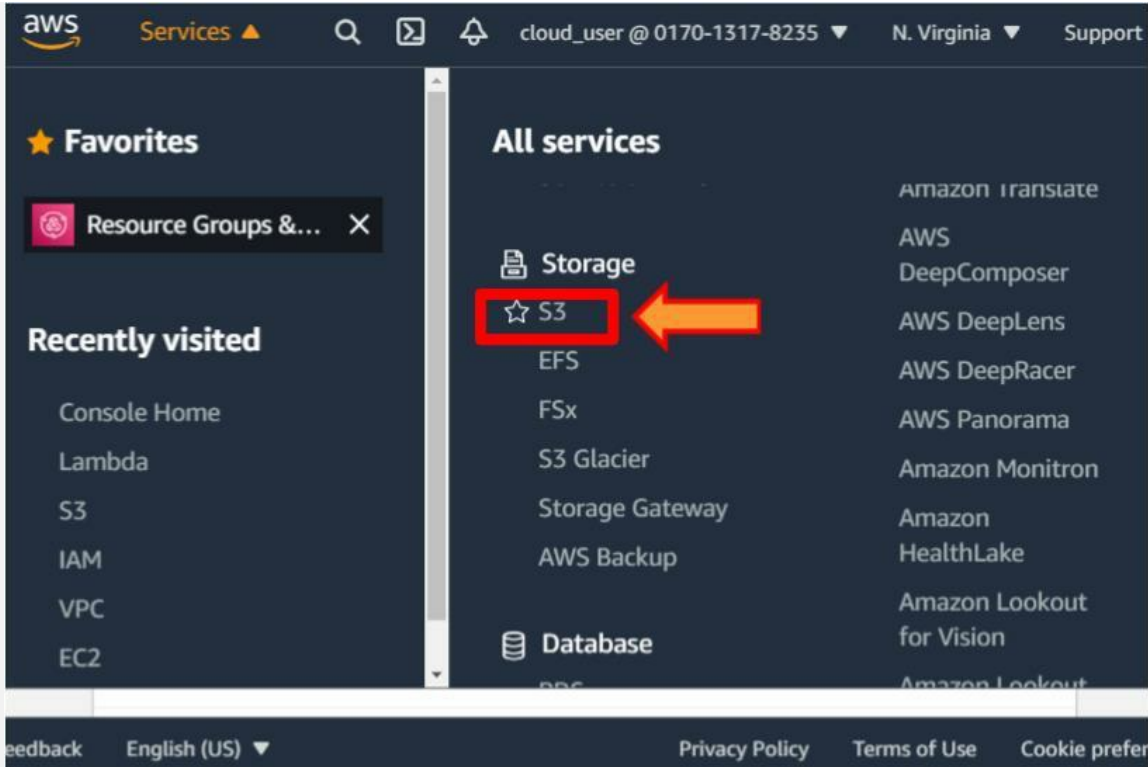
*Figure 3-33: Programmatically Utilizing S3 Select*

## Step 1: Reviewing S3 Bucket

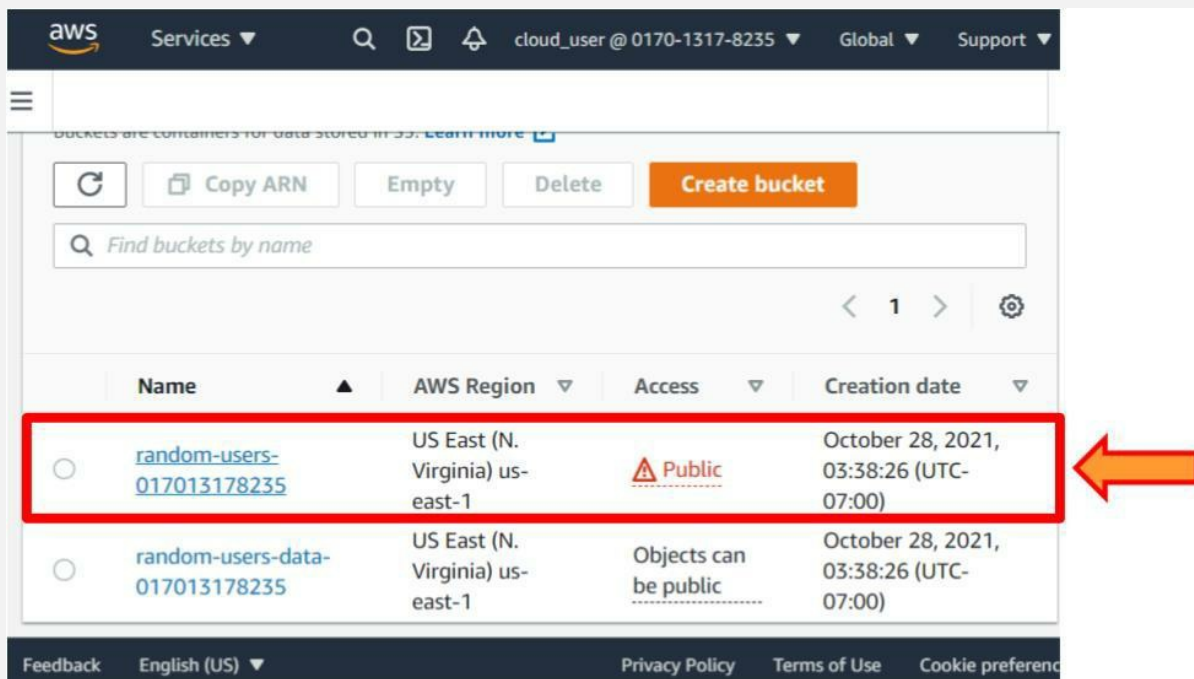
1. Log in to the **AWS Console**.
2. Click on the **Services**.







3. Select **S3** from the **Storage**.



4. You will create two buckets—the **random\_users** and **random\_users\_data**.
5. Click on the **random\_users** bucket.






6. Review the objects.

| aws Services <span>cloud_user @ 0170-1317-8235</span> <span>Global</span> <span>Support</span> |   |        |  |         |                 |  |
|--|---|--------|--|---------|-----------------|--|
| <input type="checkbox"/>   | Name ▲  | Type ▼ | Last modified ▼                        | Size ▼  | Storage class ▼ |  |
| <input type="checkbox"/>   |  <a href="#">icon.png</a>        | png    | October 28, 2021, 03:39:00 (UTC-07:00) | 10.8 KB | Standard        |  |
| <input type="checkbox"/>   |  <a href="#">index.html</a>      | html   | October 28, 2021, 03:38:58 (UTC-07:00) | 3.7 KB  | Standard        |  |
| <input type="checkbox"/>   |  <a href="#">randomusers.css</a> | css    | October 28, 2021, 03:38:59 (UTC-07:00) | 1.4 KB  | Standard        |  |
| <input type="checkbox"/>   |  <a href="#">randomusers.js</a>  | js     | October 28, 2021, 03:38:59 (UTC-07:00) | 2.2 KB  | Standard        |  |

Feedback English (US) Privacy Policy Terms of Use Cookie preference

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

7. Click on the **random\_users\_data** bucket.

| aws Services <span>cloud_user @ 0170-1317-8235</span> <span>Global</span> <span>Support</span> |  |                                 |  |  |
|--|--|---------------------------------|--|--|
| Buckets are containers for data stored in S3. <a href="#">Learn more</a>                       |  |                                 |  |  |
|             |  Copy ARN | Empty                           | Delete   | Create bucket                          |
| Find buckets by name   |  |                                 |  |  |
| < 1 > ⚙  |  |                                 |  |  |
| <input type="radio"/>  | Name ▲   | AWS Region ▼                    | Access ▼   | Creation date ▼                        |
| <input type="radio"/>  | <a href="#">random-users-017013178235</a>  | US East (N. Virginia) us-east-1 |  Public | October 28, 2021, 03:38:26 (UTC-07:00) |
| <input type="radio"/>  | <a href="#">random-users-data-017013178235</a>   | US East (N. Virginia) us-east-1 | Objects can be public  | October 28, 2021, 03:38:26 (UTC-07:00) |

Feedback English (US) Privacy Policy Terms of Use Cookie preference

8. Click on the **users\_1.json** file.

The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with the AWS logo, 'Services', a search icon, a document icon, a notification bell, the user 'cloud\_user @ 0170-1317-8235', 'Global', and 'Support'. Below this is a toolbar with 'Delete', 'Actions', 'Create folder', and 'Upload' buttons. A search bar says 'Find objects by prefix'. The main area is a table of objects:

| <input type="checkbox"/> | Name                         | Type | Last modified                          | Size     | Storage class |
|--------------------------|------------------------------|------|--|----------|---------------|
| <input type="checkbox"/> | <a href="#">users_1.json</a> | json | October 28, 2021, 03:38:58 (UTC-07:00) | 678.6 KB | Standard      |
| <input type="checkbox"/> | <a href="#">users_2.json</a> | json | October 28, 2021, 03:38:58 (UTC-07:00) | 676.4 KB | Standard      |
| <input type="checkbox"/> | <a href="#">users_3.json</a> | json | October 28, 2021, 03:38:58 (UTC-07:00) | 676.8 KB | Standard      |

At the bottom, there's a footer with 'Feedback', 'English (US)', 'Privacy Policy', 'Terms of Use', and 'Cookie preference'.

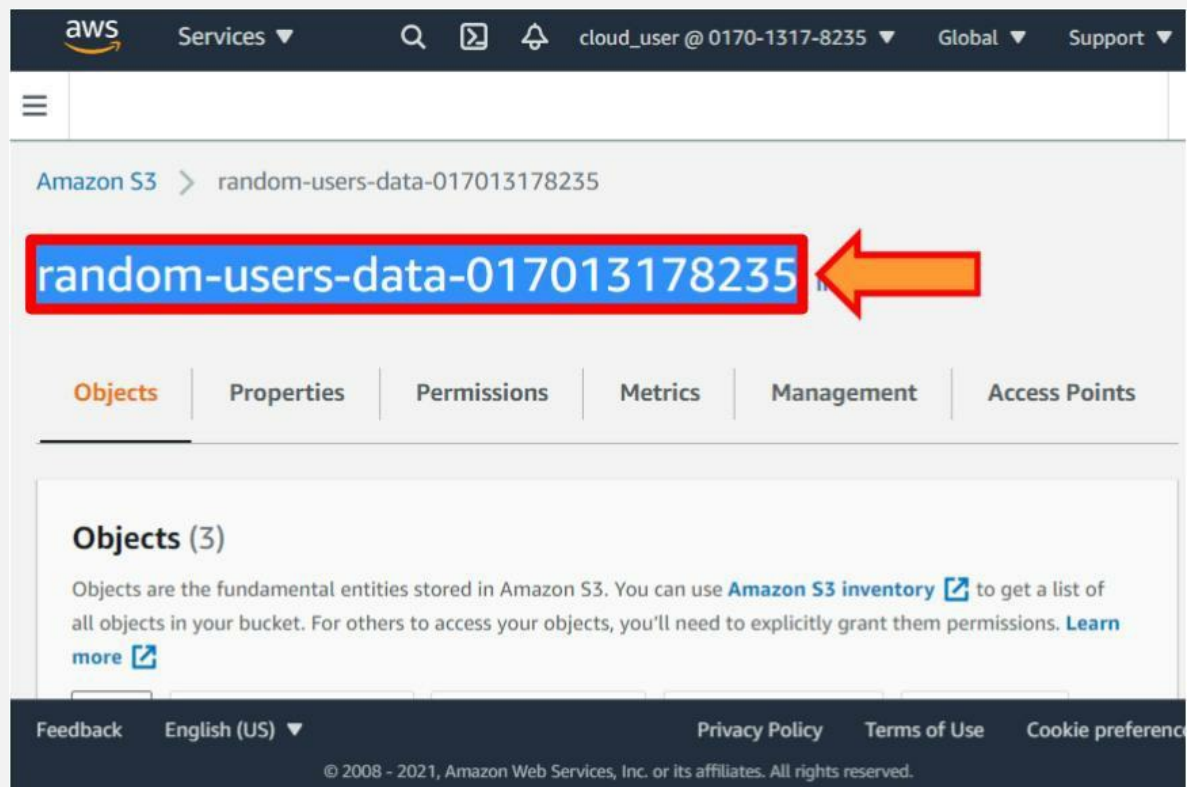
9. Click on the **Download** button.

The screenshot shows the details page for the 'users\_1.json' object in the AWS S3 console. The breadcrumb trail is 'Amazon S3 > ... > users\_1.json'. The object name 'users\_1.json' is displayed with an 'Info' link. Below this are three buttons: 'Copy S3 URI', 'Download' (highlighted with a red box and an orange arrow), and 'Object actions'. There are three tabs: 'Properties' (selected), 'Permissions', and 'Versions'. The 'Object overview' section is visible, showing the 'Owner' field. The footer is the same as the previous screenshot.

10. Open the **user\_1.json** file. Review the user data.

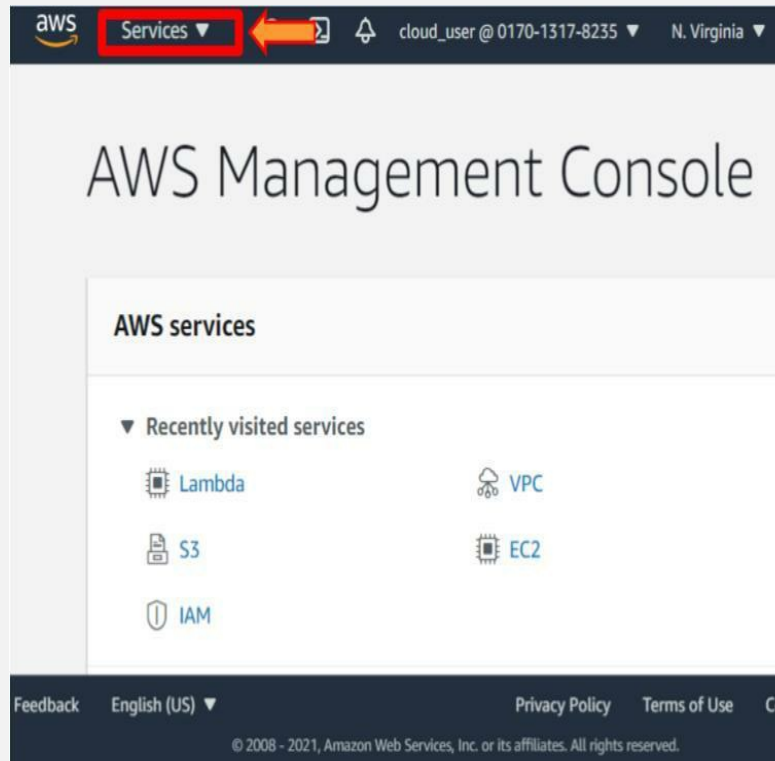
```
1  [
2    {
3      "gender": "female",
4      "name": {
5        "title": "Miss",
6        "first": "Ana",
7        "last": "Carroll"
8      },
9      "location": {
10       "street": {
11         "number": 1100,
12         "name": "Daisy Dr"
13       },
14       "city": "Dallas",
15       "state": "Delaware",
16       "country": "United States",
17       "postcode": 70161,
18       "coordinates": {
19         "latitude": "80.7677",
20         "longitude": "54.0959"
21       }
22     }
23   ]
```

11. Copy the **random-users\_data** bucket.

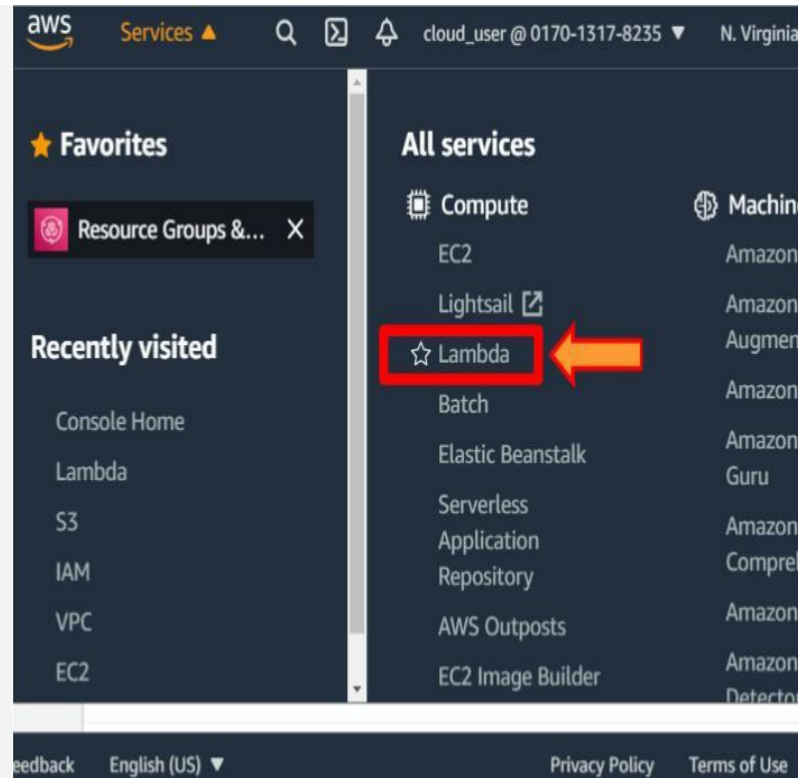


## Step 2: Modifying Lambda Function

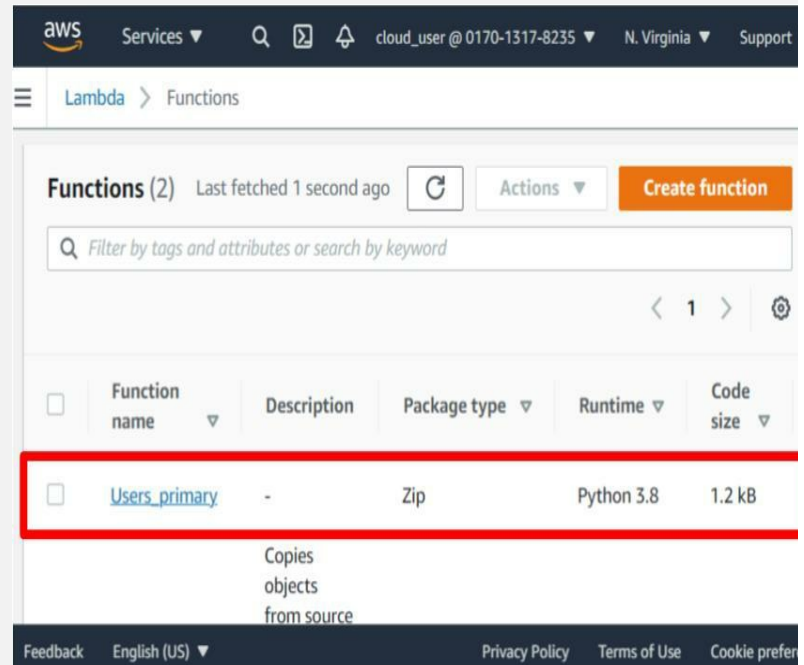
1. Click on the **Services**.



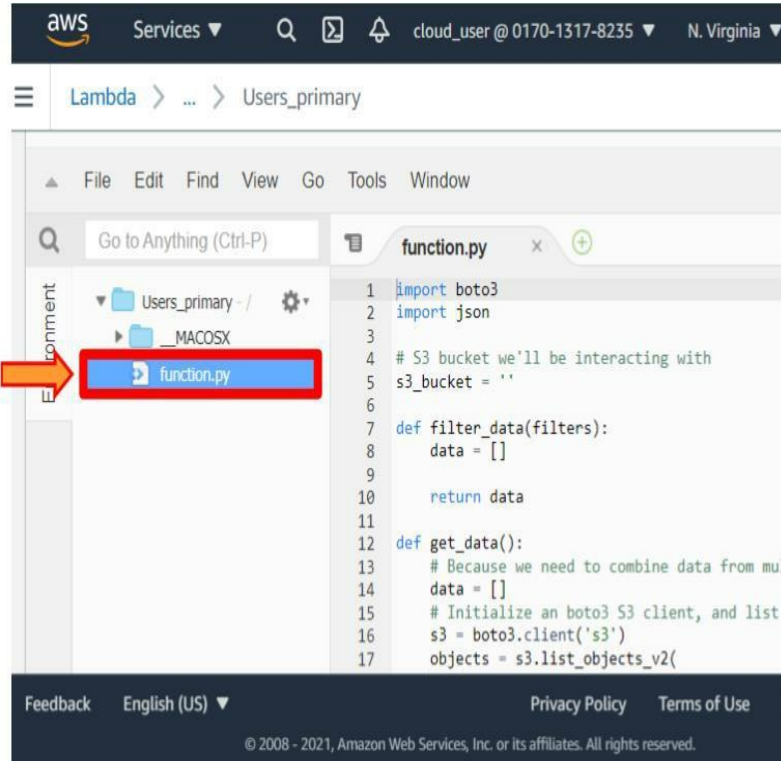
2. Select the **Lambda** from the **Compute**.



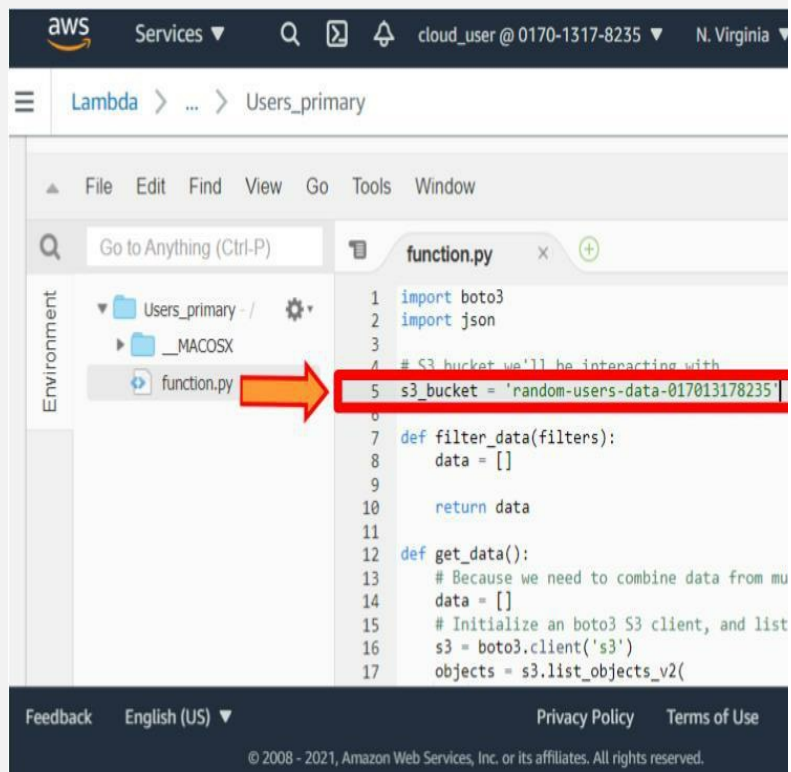
3. Click on the **Users\_Primary** Lambda function.



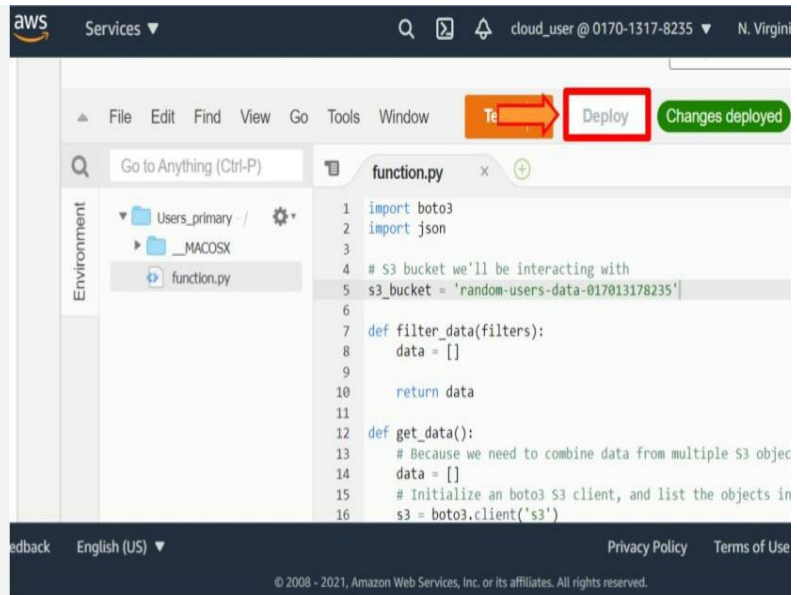
4. Click on the **function.py** file.



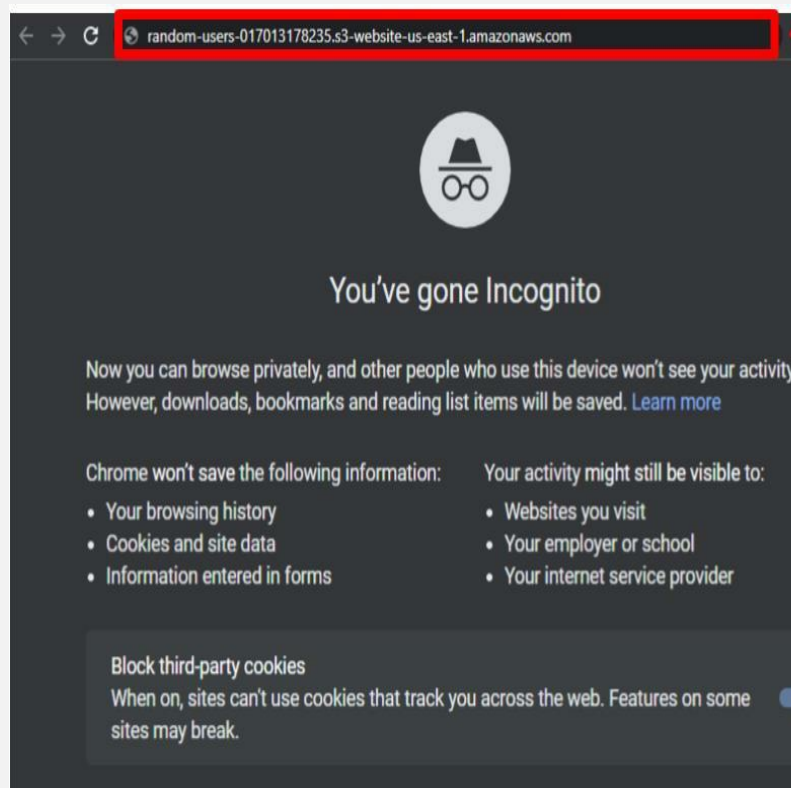
5. Paste the **random-users-data-Account\_Number** bucket name



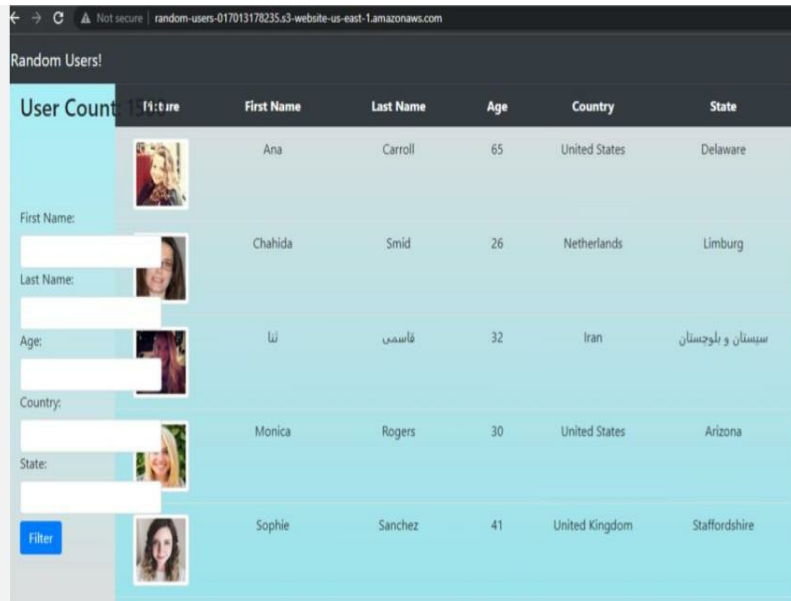
6. Click on the **Deploy** button to save the changes.



7. Open a new tab of the browser. Type URL **random-users-<Account ID>-us-east-1.amazonaws.com**. Then, press **Enter** on your keyboard.



8. After a few seconds, you will have the data of random people on



9. But you cannot filter out the data in this web app right **Users\_Primary** Lambda function dashboard.
10. Modify the **filter\_data** function in the code.

```

aws Services 🔽 🔍 📄 🔔 cloud_user @ 0170-1317-8235 🔽 N. Virginia 🔽

Lambda > ... > Users_primary

function.py

3
4 # S3 bucket we'll be interacting with
5 s3_bucket = 'random-users-data-01701
6
7 def filter_data(filters):
8     data = []
9
10    return data
11
12 def get_data():
13     # Because we need to combine data
14     data = []
15     # Initialize an boto3 S3 client,
16     s3 = boto3.client('s3')
17     objects = s3.list_objects_v2(
18         Bucket = s3_bucket
19     )['Contents']
20
21

```

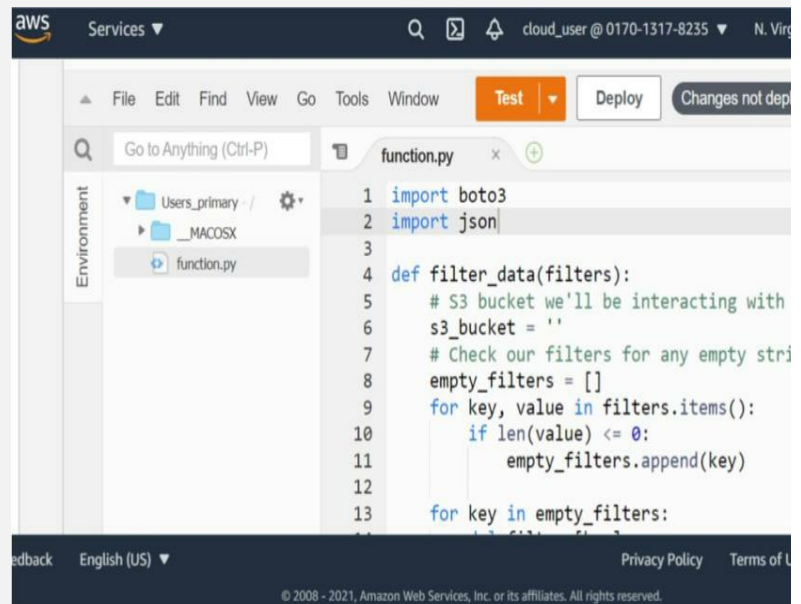
11. Copy the code from the provided Github link:  
<https://github.com/linuxacademy/Content-AWS-Certified-Data-Ana>

## [Speciality/blob/master/Lab Assets/programmatically utilizing s3 s](#)

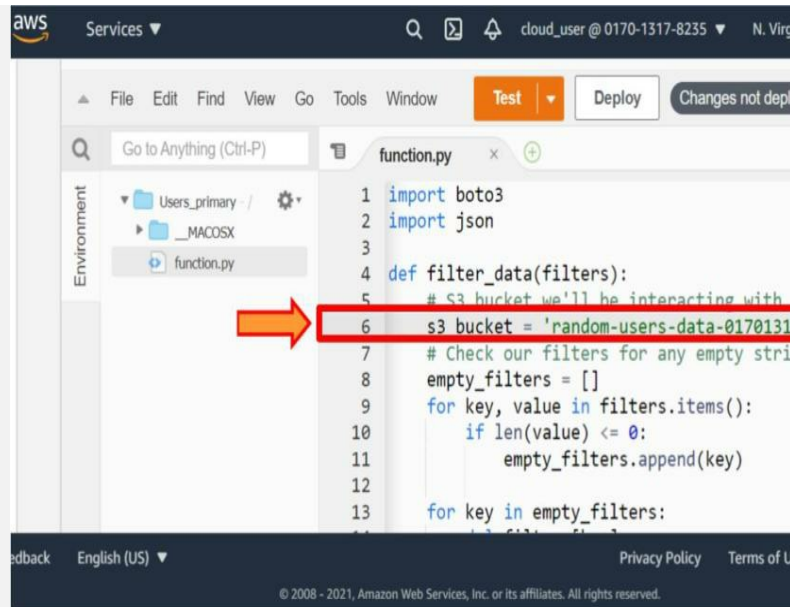
101 lines (87 sloc) 4.73 KB

```
1 import boto3
2 import json
3
4 def filter_data(filters):
5     # S3 bucket we'll be interacting with
6     s3_bucket = ''
7     # Check our filters for any empty strings to ensure our S3 Se
8     empty_filters = []
9     for key, value in filters.items():
10         if len(value) <= 0:
11             empty_filters.append(key)
12
13     for key in empty_filters:
14         del filters[key]
15
16     # If all filters are removed by the above, call the get_data(
17     if len(filters) <= 0:
```

12. Paste the code in the code editor.

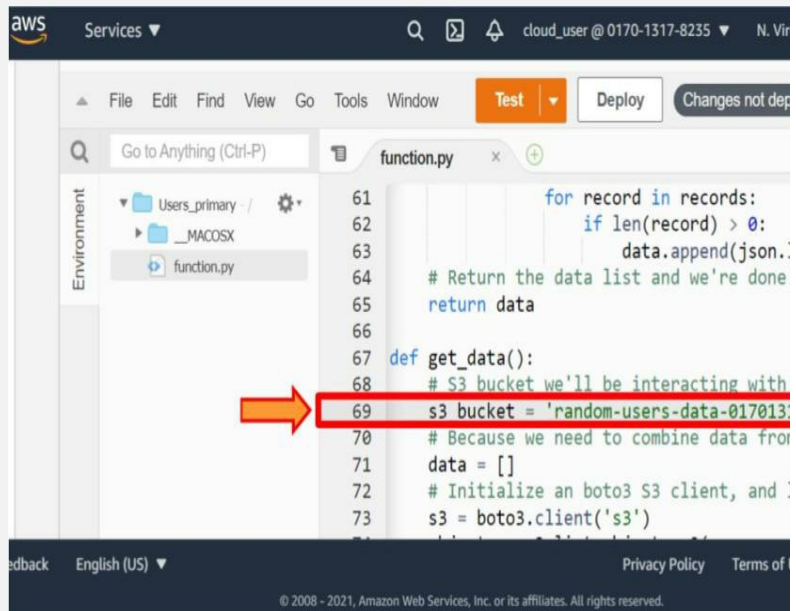


13. Online number 6, copy and paste the **random-users-data-<Acc**



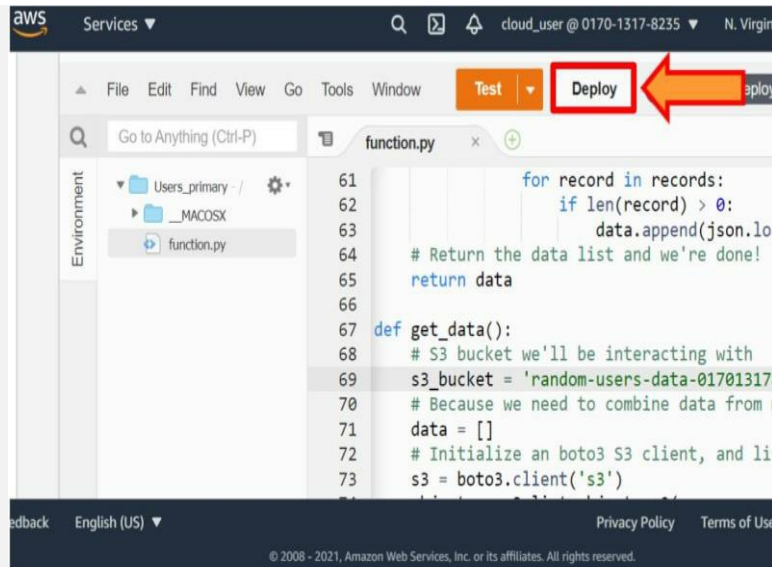
```
1 import boto3
2 import json
3
4 def filter_data(filters):
5     # S3 bucket we'll be interacting with
6     s3_bucket = 'random-users-data-0170131'
7     # Check our filters for any empty strings
8     empty_filters = []
9     for key, value in filters.items():
10         if len(value) <= 0:
11             empty_filters.append(key)
12
13     for key in empty_filters:
```

14. On line number 69, copy and paste the **random-users-data-0170131**

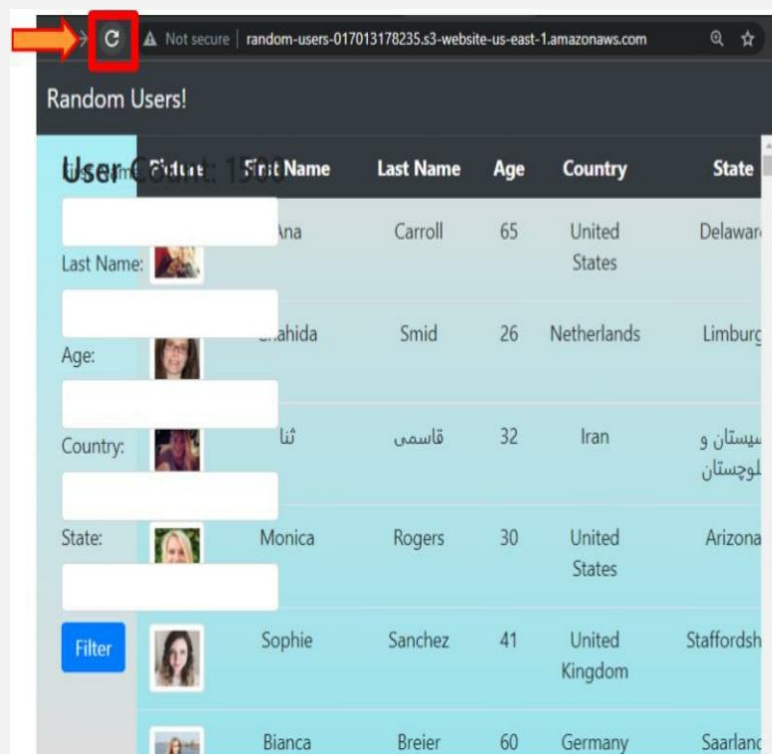


```
61     for record in records:
62         if len(record) > 0:
63             data.append(json.loads(record))
64     # Return the data list and we're done
65     return data
66
67 def get_data():
68     # S3 bucket we'll be interacting with
69     s3_bucket = 'random-users-data-0170131'
70     # Because we need to combine data from multiple buckets
71     data = []
72     # Initialize an boto3 S3 client, and get the bucket name
73     s3 = boto3.client('s3')
```

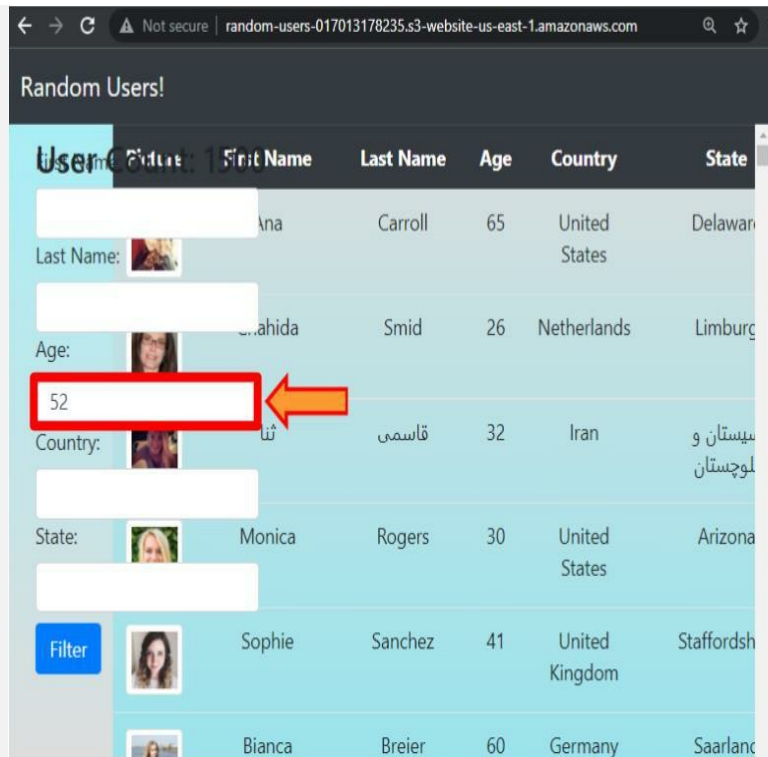
15. Click on the **Deploy** button to save the changes.



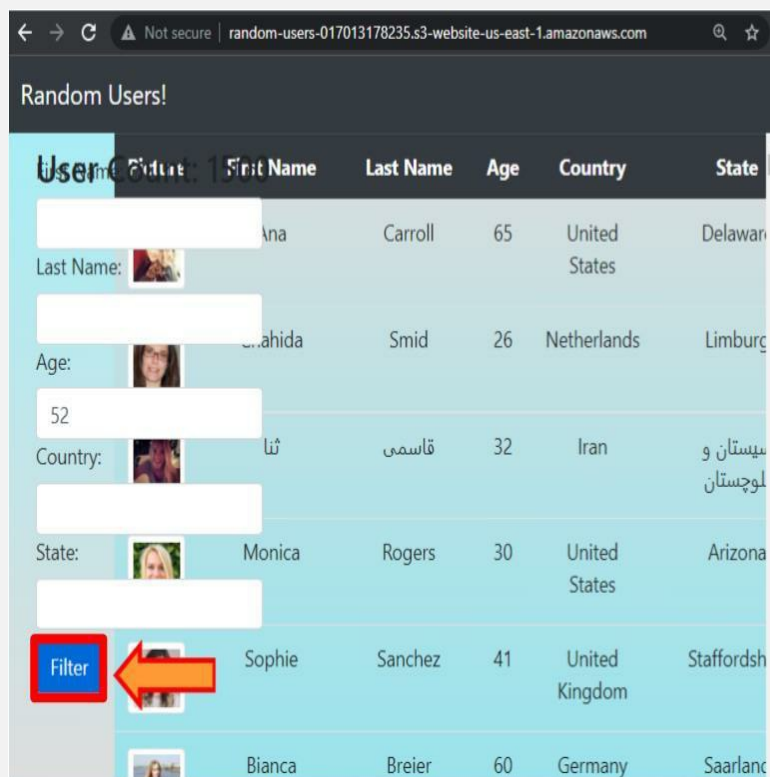
16. Go back to the web app tab. Click on the **Refresh** button.



17. In the **Age** text bar, type **52**.



18. Click on the **Filter** button.



19. You will see the web app is now showing the data of the **Age: 52**

random-users-017013178235.s3-website-us-east-1.amazonaws.com

### Random Users!

User Profile: 27

Last Name: [Redacted]

Age: [Redacted]

Country: [Redacted]

State: [Redacted]

Filter

| First Name | Last Name | Age | Country        | State                 |
|------------|-----------|-----|----------------|-----------------------|
| Cireneu    | Porto     | 52  | Brazil         | Sergipe               |
| Tim        | Steward   | 52  | Australia      | Tasmania              |
| [Redacted] | Douglas   | 52  | United Kingdom | County Armagh         |
| [Redacted] | Da Silva  | 52  | Switzerland    | Basel-Landschaft      |
| Daniel     | Polon     | 52  | Finland        | Northern Ostrobothnia |

20. In the **Country** text bar, type **Spain**.

random-users-017013178235.s3-website-us-east-1.amazonaws.com

### Random Users!

User Profile: 27

Last Name: [Redacted]

Age: [Redacted]

Country:

State: [Redacted]

Filter







| First Name | Last Name | Age | Country        | State                 |
|------------|-----------|-----|----------------|-----------------------|
| Cireneu    | Porto     | 52  | Brazil         | Sergipe               |
| Tim        | Steward   | 52  | Australia      | Tasmania              |
| [Redacted] | Douglas   | 52  | United Kingdom | County Armagh         |
| [Redacted] | Da Silva  | 52  | Switzerland    | Basel-Landschaft      |
| Daniel     | Polon     | 52  | Finland        | Northern Ostrobothnia |

21. Click on the **Filter** button.

← → ↻ ⚠ Not secure | random-users-017013178235.s3-website-us-east-1.amazonaws.com 🔍 ☆

Random Users!

User Count: 27






|   | Picture  | First Name | Last Name | Age | Country        | State                 |
|---|--|------------|-----------|-----|----------------|-----------------------|
| Last Name:  |   | Cireneu    | Porto     | 52  | Brazil         | Sergipe               |
| Age:  |   | Tim        | Steward   | 52  | Australia      | Tasmania              |
| Country:  |   | an         | Douglas   | 52  | United Kingdom | County Armagh         |
| State:  |   | lip        | Da Silva  | 52  | Switzerland    | Basel-Landschaft      |
|  |  | Daniel     | Polon     | 52  | Finland        | Northern Ostrobothnia |

22. You will see the web app is now showing the data of the **Country**

← → ↻ ⚠ Not secure | random-users-017013178235.s3-website-us-east-1.amazonaws.com

### Random Users!

User Count: 87

| Picture   | First Name | Last Name | Age | Country | State              |
|---|------------|-----------|-----|---------|--------------------|
|  | scar       | Ortega    | 55  | Spain   | País Vasco         |
|  | Clara      | Dominguez | 65  | Spain   | País Vasco         |
|  | Rodrigo    | Carrasco  | 46  | Spain   | La Rioja           |
|  | idad       | Serrano   | 26  | Spain   | Cantabria          |
|  | David      | Ferrer    | 30  | Spain   | Castilla la Mancha |

Filter

23. Hence, you can create data filtering web app using S3 Select.

**Note:** After completing the lab, delete all the AWS services used in the lab to get charged.

## Mind Map



***Figure 3-34: Mind Map Databases in AWS***

## Practice Questions

1. A database engine, sometimes known as \_\_\_\_\_.
  - A. Storage Device
  - B. Storage Software
  - C. Storage Engine
  - D. Storage Machine
2. Which of the following is a collection of data objects that have specified relationships and can be easily retrieved?

- A. Relational Database
  - B. Non-relational Database
  - C. Row Database
  - D. Columnar Database
3. Which database arranges data by the record and keeps all data related with a record in memory next to each other?
- A. Relational Database
  - B. Non-relational Database
  - C. Row Database
  - D. Columnar Database
4. Which of the following database management systems stores data in columns rather than rows?
- A. Relational Database
  - B. Non-relational Database
  - C. Row Database
  - D. Columnar Database
5. Which databases differ from traditional relational databases because they are stored in a non-tabular format?
- A. Relational Database
  - B. Non-relational Database
  - C. Row Database
  - D. Columnar Database
6. Which AWS resource-managed graph database service simplifies the design and operation of applications that deal with vast linked datasets?
- A. Amazon RDS

- B. Amazon Neptune
  - C. Amazon DocumentDB
  - D. Amazon Aurora
7. Which AWS database service is a document store engine?
- A. Amazon RDS
  - B. Amazon Neptune
  - C. Amazon DocumentDB
  - D. Amazon Aurora
8. Which of the following is a data structure consisting of a finite number of nodes and edges that link them?
- A. Graph Structure
  - B. JSON Structure
  - C. Document Structure
  - D. Array Structure
9. The row database is excellent for which purpose?
- A. Social Media
  - B. OLAP
  - C. Cyber Security
  - D. OLTP
10. The columnar database engines are good for which purpose?
- A. Social Media
  - B. OLAP
  - C. Cyber Security
  - D. OLTP
11. Which of the following allows apps to get only a subset of

data from an object?

- A. S3 Select
- B. Athena
- C. DynamoDB
- D. Aurora Serverless

12. Which interactive query service allows you to use conventional SQL to evaluate data directly in AWS S3?

- A. S3 Select
- B. Athena
- C. DynamoDB
- D. Aurora Serverless

13. Which fully managed NoSQL database engine service delivers quick and predictable performance while seamless scaling?

- A. S3 Select
- B. Athena
- C. DynamoDB
- D. Aurora Serverless

14. Which is a relational database engine fully managed and compatible with MySQL and PostgreSQL?

- A. Aurora
- B. Athena
- C. Auto-Scaling
- D. AppFlow

15. Which is a DB cluster that dynamically scales processing capacity based on the needs of your application?

- A. Aurora

- B. Athena
- C. Auto-Scaling
- D. AppFlow

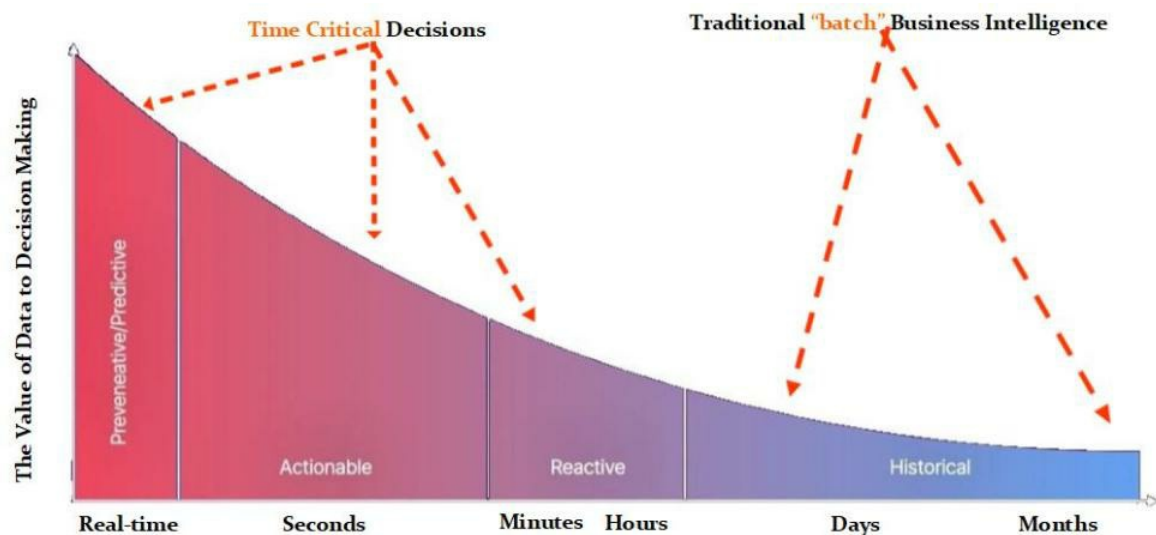
# CHAPTER STREAMING DATA

## 04: COLLECTING

### Introduction to Collecting Streaming Data

The streaming data makes up a mass amount of data collected, so it is important to understand what it is and what are the different ways you can collect, process, and store it within AWS. Streaming data is new data, and it plays a big role in actionable decisions that can be made with that data.

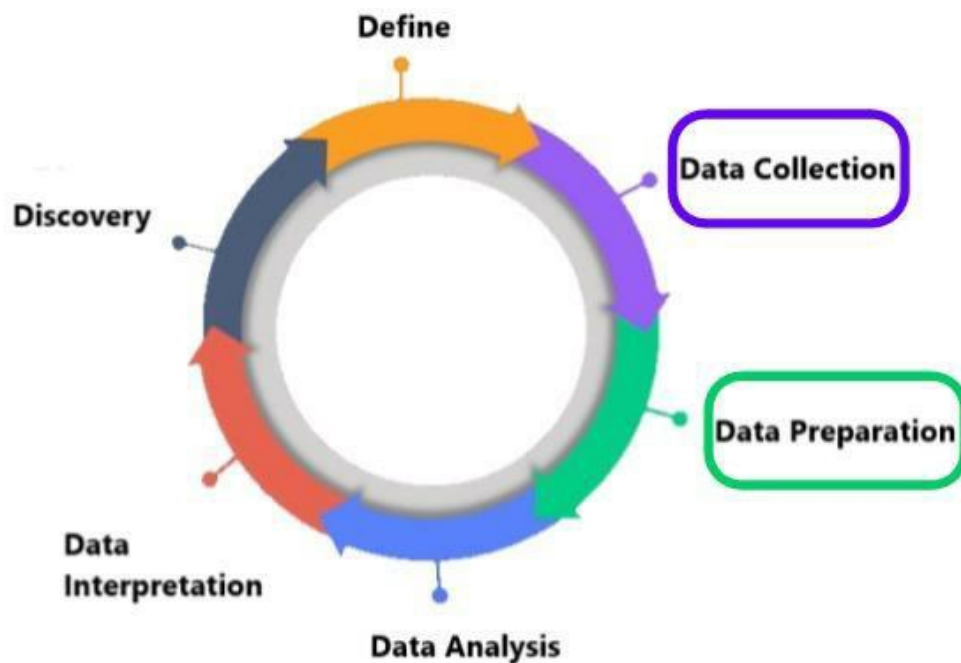
Let us look at this chart from Forrester's research; it describes how data loses value over time. On the right-hand side, you can see historical data that is normally stored in a data warehousing solution or a data lake. If you move further left along with this graph, you start to see more actionable data. This data is captured in real-time, within seconds or minutes. It is where all the interesting data exists.



*Figure 4-01: Data Losses Value Quickly Over Time*

## Steps for Success

If you look at our steps for success during this chapter, you will cover several AWS services that allow us to capture streaming and interesting data. You will learn about collecting, processing, transforming, and storing streaming data using services like Amazon Kinesis and Amazon Managed Streaming for Kafka.



*Figure 4-02: Introduction to Collecting Streaming Data*

## Kinesis Family

### Data Collection

The practice of gathering, measuring, and evaluating correct research insights using established procedures is data collection. Based on the evidence gathered, a researcher can assess their hypothesis. Depending on the information requested, the approach to data gathering differs for different topics of research. Regardless of the subject of study, data

gathering is the first and most significant stage in most situations.

The systematic process of gathering and measuring information from many sources to create a full and accurate picture of a subject is known as data collection. Data collecting allows a person or organization to answer pertinent questions, assess results, and forecast future probability and trends.

Data collection's most critical objective is to ensure that information-rich and reliable data is collected for statistical analysis so that data-driven decisions can be made.

Maintaining research integrity, making educated business choices, and guaranteeing quality assurance all need accurate data collection. Data from mobile applications, website visits, loyalty programs, and online surveys, for example, might be used to understand more about customers in retail transactions. In a server consolidation project, data collection would include a physical inventory of all servers and an accurate description of what is installed on each server, the operating system, middleware, and the server-supported application or database

## **Data Collection Methods**

Surveys, interviews, and focus groups are the most common methods for gathering information. Corporations may now collect data from mobile devices, website traffic, server activity, and other relevant sources using Web and analytics technologies depending on the project.

## **Big Data Collection**

It is based on collecting and then mining large amounts of data for information. Big data refers to massive volumes of organized, semi-

structured, and unstructured data acquired by businesses. New approaches for collecting and analyzing data have emerged because it takes a lot of time and money to load big data into a traditional relational database for analysis. In a data lake, raw data with extra information is aggregated. Machine learning and artificial intelligence systems then employ complicated algorithms to search for repeating patterns.

Hence, you know that data comes from many different places, and most of the time, we can capture our data as static or streaming data. Static data is the data that is not changing or that you download and then do something with. Most of the in-house datasets that your organization has collected and stored in a data warehouse or a data lake are considered static data. This data has been collected over many months or many years. As a Machine Learning Specialist, it is your job to do something with that data. Streaming data is a bit different because streaming data is constantly being updated or continuously being added to during the data collection process.

## **Streaming Data**

Streaming data is generated continuously by thousands of data sources. These are typically sent in small data records simultaneously, think in the order of kilobytes. For example, think about sensors in vehicles or industrial equipment that you might see in some factory or farming machinery. These sensors send data to streaming applications; these applications monitor performance, detect any potential defects in advance, or even place an order for spare parts. When defects or any type of anomaly is detected within the equipment it is running on; it prevents equipment downtime. Another example is a financial

institution that tracks changes in the stock market in real-time. It then computes value at risk and rebalances portfolios automatically depending on stock price fluctuations.

Another example is a solar power company that maintains power throughput for its customers or pays penalties. Hence, it implements a streaming data application that monitors all of its panels in the field. It schedules service in real-time, thereby minimizing the periods of low throughput for each panel and the associated penalty payouts. Another example is a media publisher that streams billions of clickstream records from online partners, social media websites, or e-commerce websites. It aggregates and enriches the data depending on the users' demographics using those applications. It can optimize the content placed on its site to deliver a more relevant and better experience for its audience or its users.

A final example is an online gaming company that collects streaming data about the players playing the game interactively or in real-time. It feeds that data into the gaming platform itself and then analyzes that data in real-time. It also offers incentives or another type of dynamic experience to help engage the players more or to help them further enjoy the video game.



Figure 4-03: Different Data Streaming Platforms

Collecting, transforming, processing, and making decisions about data in real-time, in seconds or minutes, is very important.

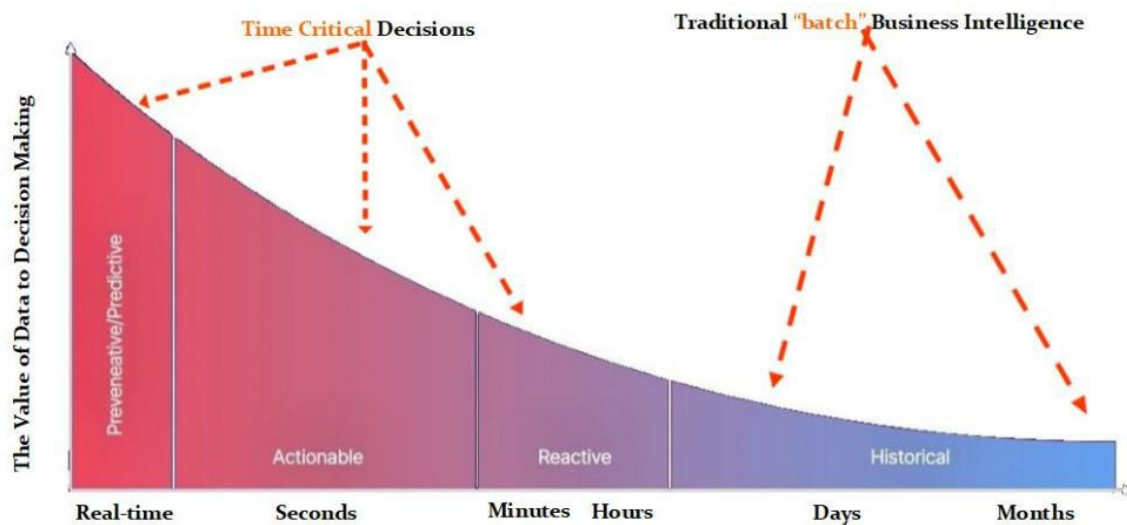
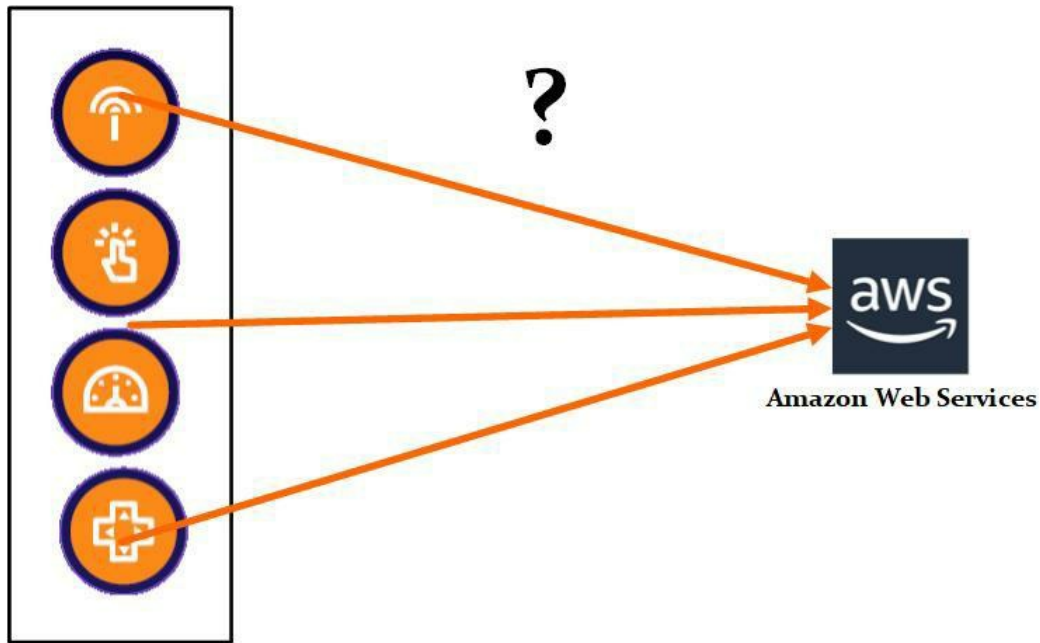


Figure 4-04: Data Losses Values Quickly Over Time

We have our streaming data on our left-hand side through these various icons:



*Figure 4-05: Which Service to Use for Data Streaming*

There are several ways to get streaming data into AWS. When getting streaming data into AWS, you need to think about the Kinesis service, which will be important to understand and utilize for our streaming data applications.

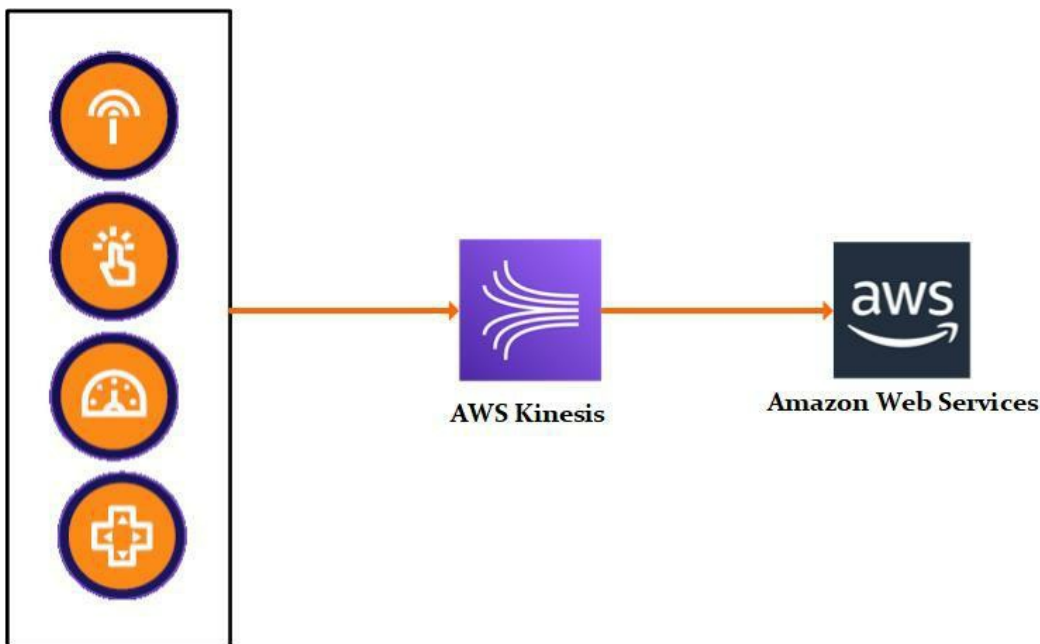


Figure 4-06: Data Streaming on AWS Kinesis

## AWS Kinesis Family

The AWS Kinesis family is not just one service but a family of services. Kinesis offers several different services that help you get your streaming data into AWS and build robust applications around streaming data. The first service is the Kinesis data stream.

Kinesis data stream allows us to collect and process large streams of data records in real-time. It is the most important service when it comes to Kinesis services. The next Kinesis service is Kinesis Data Firehose. Kinesis Data Firehose is our delivery stream, which allows us to deliver our streaming data to various data sources, such as Amazon S3, Redshift, Elasticsearch, or Splunk. The next Kinesis service is Kinesis video streams that allow us to stream live videos from devices to the AWS cloud, and you can build real-time applications around these video streams. Finally, we have Kinesis data analytics. It is what you can use to process and analyze streaming data using standard SQL queries. Hence, you can essentially run SQL queries in real-time on streaming data.

Thus, Kinesis Data Stream, Kinesis Data Firehose, Kinesis Video Streams, and Kinesis Data Analytics are the four services that make up the Kinesis family.

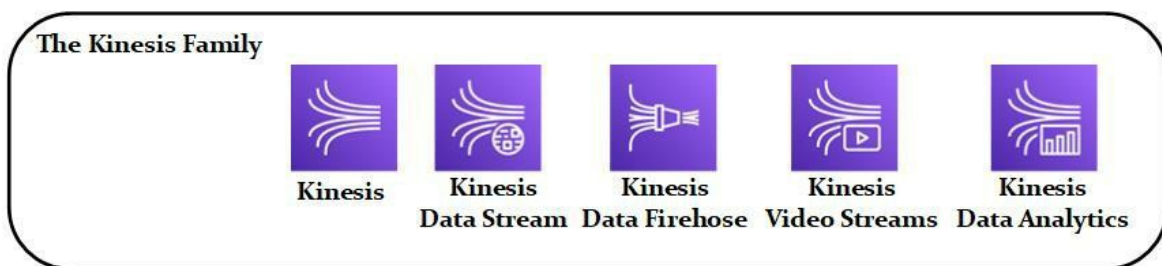


Figure 4-07: AWS Kinesis Family

# Kinesis Data Streams

## Introduction

One of the benefits of using Kinesis Data Stream is aggregating real-time data and then loading it into some data warehousing solution like Redshift or some MapReduce cluster like EMR. Kinesis Data Streams are also durable and elastic, meaning that you would not lose our data records. You would not lose our streaming data, and it scales up or down depending on the number of records coming into our stream. It means that you can get all the benefits of a managed streaming service rather than having to host it on EC2 instances ourselves. You want to take advantage of managed services wherever you can, which is an integral part of collecting streaming data within AWS. You can also have parallel applications reading from the stream. Hence, you can perform different functions on that data. It is one of the significant differences between Kinesis and other queuing services.

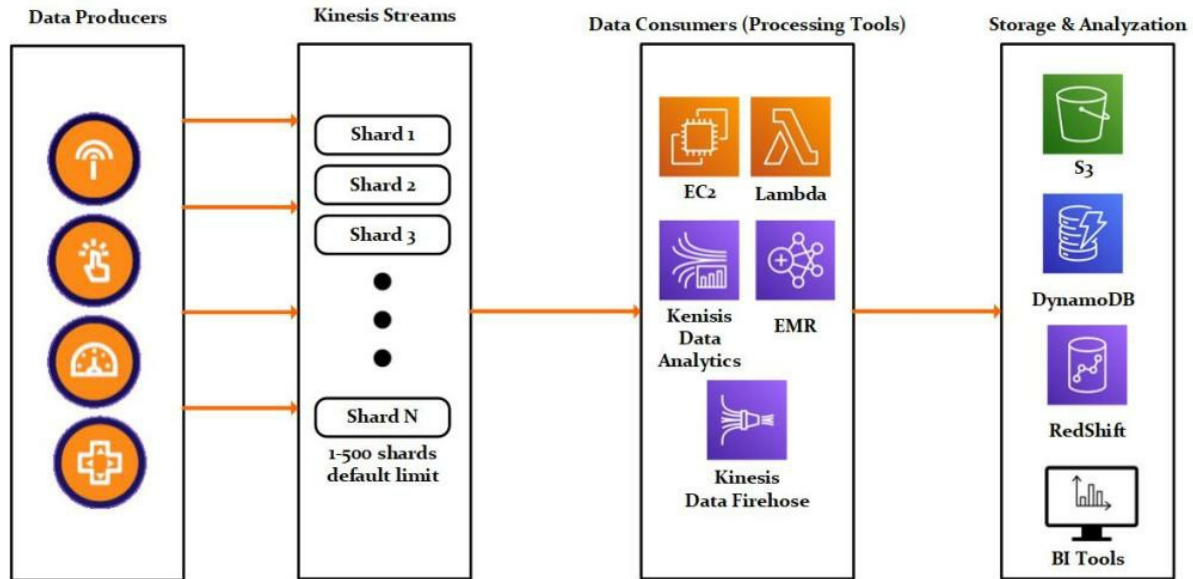
We will look at some of the critical components and the different parts that allow us to use Kinesis Data Streams. First, you have to have some data producers. These are the applications that are producing streaming data. For instance, they could be application log files, social media streams, real-time user interactions from video games, IoT devices, or they could even be EC2 instances or traditional servers on-premise. They could be mobile clients or clickstream data from users interacting with an online website or application. Hence, anything that produces streaming data or anything that makes data you can collect in real-time is going to be our data producer. You can set up a Kinesis stream and push that data to the Kinesis string. The way in which data is

aggregated and captured is through shards.

Shards are essentially the packaging mechanism that allows us to aggregate our data and send it into AWS. You can have one shard, two shards, three shards, or you can have up to n shards. The default limit is 500 shards, but you can submit a ticket and increase this limit if you need to.

These shards act as packaging mechanisms that take our data, package it, and ship it off to the consumers or to the applications consuming the data. Hence, once you start collecting our streaming data into our shards, some data consumers who process the data consume that data. You can think about data consumers as being any application or software that processes or ingests the data. Hence, we can think about EC2 instances. They can be the Lambda function, other Kinesis family services like Kinesis Data Analytics and Kinesis Data Firehose, or they can also be EMR. These data consumers can do a multitude of different things. They can process the data, analyze the data, create some real-time dashboards, store it off, or perform some type of analysis with that data.

With Kinesis Data Streams, you are not required to store that data anywhere. Hence, you could just run some analytics on it and then get rid of it. You do not have to store it, but you will need to store that off into a data lake or some data warehousing solution in most cases. You can use various methods to get that data and then store it off to process it later or just add it to our historical data sets. You could store the data into resources like S3 for data lakes, DynamoDB. Also, you can keep it into Redshift for data warehousing analysis or create dashboards using business intelligence tools.



*Figure 4-08: Kinesis Data Stream*

**EXAM TIP:** Know what each service is and how it processes/handles streaming data.

## Working with Kinesis Data Streams

Kinesis Data Streams may be used to collect and aggregate data in real-time. IT infrastructure log data, application logs, social media, market data feeds, and online clickstream data are some examples of the data types that may be employed. Because the data intake and processing are both done in real-time, the processing is generally minimal.

The following are some examples of how Kinesis Data Streams can be used:

- 1. Accelerated log and data feed intake and processing:**

Data can be immediately sent into a stream by producers. For example, push system and application logs will be ready for analysis in seconds. If

the front-end or application server dies, the log data will not be lost. Kinesis Data Streams provide quicker data feed intake because you do not batch the data on the servers before submitting it for input.

## **2. Real-time metrics and reporting:**

Data gathered via Kinesis Data Streams may be used for real-time data analysis and reporting. Instead of waiting for batches of data to arrive, your data-processing application might work on metrics and reporting for system and application logs as they come.

## **3. Real-time data analytics:**

The strength of parallel processing is combined with the value of real-time data in this way. For example, utilizing many Kinesis Data Streams applications operating in parallel, process website clickstreams in real-time, and then assesses site usability engagement.

## **4. Complex stream processing:**

Kinesis Data Stream applications and data streams may be turned into Directed Acyclic Graphs (DAGs). It usually entails combining data from many Kinesis Data Stream applications into a single stream for later processing by another Kinesis Data Stream application.

# **Benefits of Using Kinesis Data Streams**

Although Kinesis Data Streams may be used to tackle a wide range of streaming data issues, one frequent use is real-time data aggregation. It is loading the aggregate data into a data warehouse or map-reduce cluster.

Kinesis data streams are used to store data, ensuring its longevity and flexibility. The time it takes for a record to be put into the stream and

for it to be retrieved (put-to-get latency) is generally less than one second. To put it another way, a Kinesis Data Stream application may begin consuming data from the stream practically as soon as it is added. Kinesis Data Stream's managed service relieves you of the operational load of setting up and maintaining a data intake pipeline. You can make map-reduce applications that stream data. Kinesis Data Stream's elasticity allows you to scale the stream up or down as needed, ensuring that you never lose data records before they expire.

Multiple Kinesis Data Stream applications can ingest data from a stream, allowing for parallel and different activities such as archiving and processing. Two applications, for example, can read data from the same stream. The first application updates an Amazon DynamoDB table with ongoing aggregates, and the second compresses and archives data to a data repository like Amazon Simple Storage Service (Amazon S3). A dashboard reads the DynamoDB database with ongoing aggregates for up-to-the-minute reporting.

The Kinesis Client Library enables fault-tolerant data consumption from streams and offers scalability support for Kinesis Data Stream applications.

## **Shard**

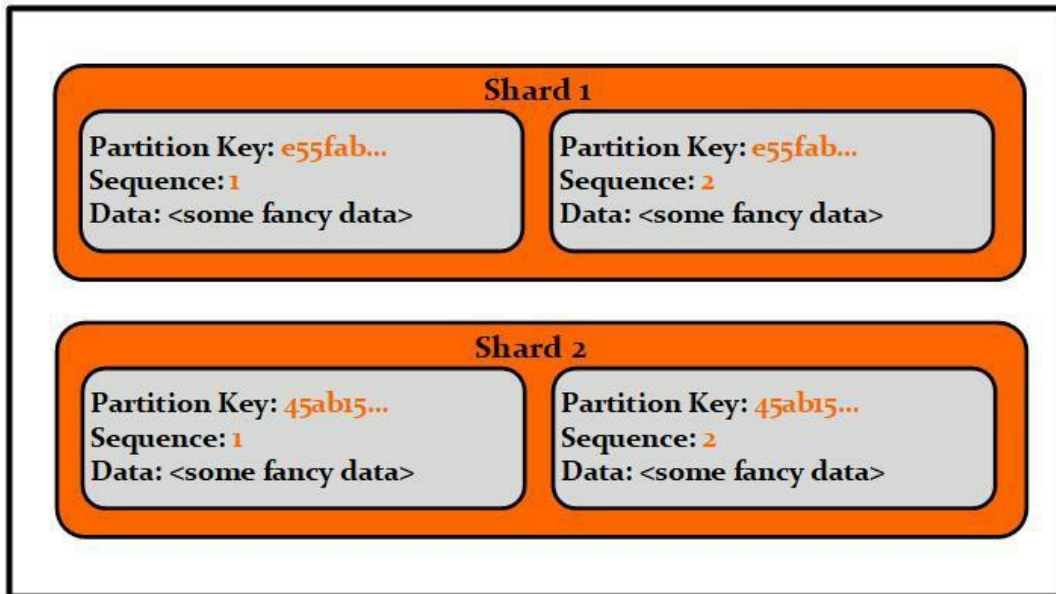
Shards are a container that holds our information shipped off to consumers. Let us assume that you have a single shard. You can see this shard has two data records; each one of these data records consists of a partition key, a sequence ID, the data, and the actual data you want to ship off to consumers. The partition key is going to be the same for all the records within a shard. The sequence number will be in the order in

which the shard received the record, and that will be our data. Each shard consists of a sequence of data records. Here, you have two, which can be ingested at 1000 records per second. The actual data payload per record can be up to one megabyte.

In most cases, you will only have a few kilobytes of information within your payload, but it can be up to one megabyte. One shard can process one megabyte per second of input data. If it can process two megabytes per second of output data, it means that a consumer can consume our data faster than you can put data into the shard. This would be an example of a single shard.

Let us take a look at what two shards would look like. Assume that you have two shards, and with two shards, you can ingest data up to 2000 records per second. Thus, you just double the number of records that can be consumed per second. This brings up an essential key feature; scaling up our shards is how you scale Kinesis streams. Hence, if you have more records input into our Kinesis streams and more user activity, you need to scale up the number of shards you are using. Like in our single shard example, the data payload per record can be up to one megabyte. So no matter what, you can have up to one megabyte of data per record. The gray squares represent our records. Thus, you double the amount of input data records you can read and the amount of output data records you can read with two shards. Hence, in this case, you can have up to two megabytes per second of input data and four megabytes per second of output data. If our input data exceeds some of these limits, maybe Kinesis Streams is not a good solution. However, in most cases, Kinesis Data Streams will be one of the go-to services for streaming data.

## Kenesis Streams



*Figure 4-09: Shards*

For example, compare a shard to objects in real-world trains, and think about the entire train. The train ID, or let us assume the train's name, will be the partition key that is going to be the same for all of the cars. Each train car will be associated with what you call a sequence number in a shard. Hence, each train car will have a different sequence number. The passengers will be the actual data you are shipping off to our consumers or our input data. Hence, those are the people that are on the train.



TrainID = Partition Key  
Train Car = Sequence Number  
Passenger = Data



*Figure 4-10: Sharding Train Example*

**EXAM TIP:** Understand what shards are, a data record, and how long a shard should be kept.

## Processing & Storage

A shard is temporary data storage. Data records are stored for 24 hours by default and can be extended up to 365 days. By default, the data retention period is 24 hours. You can raise the data retention period to seven days by enabling extended data retention. You can increase it even further by allowing long-term data retention to have the data persist in the shard for up to 365 days. You can do this by using the increased stream retention period operation. You can decrease it by using the reduced stream retention period operation. Hence, going back to our train example, passengers will be booted from the train every 24 hours, but some rules may differ for some trains. It is the

retention period, so some passengers may be allowed to stay for up to 365 days.

## **Interacting with Kinesis Data Stream**

There are a few different ways to interact with Kinesis Data Streams:

### **1. Kinesis Producer Library (KPL):**

An application that inserts user data into a Kinesis data stream is an Amazon Kinesis Data Streams producer (also called data ingestion). The Kinesis Producer Library (KPL) makes it easier for developers to create producer applications by achieving high write throughput to a Kinesis data stream.

### **2. Kinesis Client Library (KCL):**

KCL takes care of many complicated duties connected with distributed computing, allowing you to receive and process data from a Kinesis data stream. Load balancing across numerous consumer application instances, responding to consumer application instance failures, checkpointing processed records, and responding to re-sharding are examples of these. The KCL handles all of these subtasks, allowing you to concentrate on implementing your unique record-processing logic.

The KCL is not the same as the Kinesis Data Streams APIs offered in the AWS SDKs. The Kinesis Data Streams APIs assist you in managing many elements of Kinesis Data Streams, such as establishing streams, re-sharding, inserting and receiving information. The KCL adds a layer of abstraction around all of these subtasks, allowing you to focus on the particular data processing logic in your consumer application.

### **3. Kinesis Agent**

The Kinesis Agent is a ready-to-use Java application that can be

deployed on a Linux-based server. It is an agent that monitors specific files and continuously sends data to our Data Stream. Hence, you might want to install this on web servers, log servers, or database servers.

Kinesis Agent is a Java software application that allows you to quickly gather and transfer data to Kinesis Data Streams. The agent watches a group of files in real-time and feeds new data to your stream. The agent performs file rotation, checkpointing, and retries in the event of a failure. It distributes all of your data in a dependable, fast, and straightforward manner. It also emits Amazon CloudWatch metrics to assist you in monitoring and troubleshooting the streaming operation.

By default, entries from each file are processed based on the newline ('n') character. On the other hand, the agent may be set to parse multi-line entries.

The agent may be deployed on Linux-based web servers, log servers, and database servers. Configure the agent after installing it by providing the files to monitor and the data stream. Once set up, the agent takes data from files and consistently feeds it to the stream.

#### **4. Kinesis API (AWS SDK):**

It is used to process data from the Kinesis Data Stream. Once the data is in Kinesis Data Streams, you can use Kinesis Client Library, abbreviated as KCL, to directly interact with the Kinesis Producer Library to consume. These libraries are used to abstract some of the low-level commands you would have to use with the Kinesis API. However, it is used for more low-level API operations and more manual configurations. Hence, the interaction with Kinesis Data Stream is by using the Kinesis API. With the Kinesis API, you can perform the same

actions that you can achieve with the Kinesis Producer Library or the Kinesis Client Library. Hence, you can install the Kinesis Producer Library on two EC2 Instances or integrate it directly into your Java applications.

**EXAM TIP:** Know the difference between the KPL, KCL, and Kinesis API.

## **KPL vs. Kinesis API**

The Kinesis Producer Library provides a layer of abstraction when you are ingesting the data. Thus, you have to manage stream creation with the Kinesis API, things like resharding and getting records from the Kinesis stream. It is kind of all automatically handled for us with the Kinesis Producer Library. One benefit of using the API is that there are no delays in processing. Under the hood, that Kinesis Producer Library uses a Kinesis API. However, there might be some additional processing delays that might occur with the Kinesis Producer Library. Hence, with the Kinesis Producer Library, you have a higher packaging efficiency when sending your data to your stream for better performance. With the Kinesis Producer Library, you can only install it with the Java wrapper. With the Kinesis API, you are only bound by the SDKs that AWS offers, which provide a wide variety of SDKs to use.

| Kinesis Producer Library (KPL)   | Kinesis API  |
|--|--|
| Provides a layer of abstraction specifically for ingesting data.                               | API calls (PutRecords and GetRecords).   |
| Automatic and configurable retry mechanism.  | Stream creation, resharding, and putting and getting records are manually handled. |
| Additional processing delays can occur for higher packing efficiencies and better performance. | No delays in processing.   |
| Java Wrapper   | Any AWS SDK  |

*Figure 4-11: KPL VS Kinesis API*

Some key features between the Kinesis Producer Library and the Kinesis API are mentioned below:

### Features of KPL:

- Provides a layer of abstraction dedicated to data intake
- Retry system that is both automatic and adjustable
- To achieve higher packing efficiency and better performance, additional processing delays may occur
- Java wrapper

### Features of Kinesis API:

- Low-level API calls (PutRecords and GetRecords)
- Stream creations, re-sharding, and putting and getting records are manually handled
- No delays in processing
- Any AWS SDK

## Kinesis Data Stream Use Cases

Following are some use cases of Kinesis data streams where you could use Kinesis Data Streams, or where it would be a good option:

### 1. Process and evaluate logs immediately:

You can process and evaluate logs immediately. Imagine that you work for an organization with SLA and downtime agreements with other

companies and shareholders for the application you are building. You can continuously stream the logs from those applications, and you can monitor those for any errors and respond to them accordingly. Paired with other monitoring platforms, it could save your software from ever being down. Example: Continuously analyze system and application logs, and process them in seconds.

## 2. Real-time data analytics:

You could also run real-time analytics on things like clickstream data. Imagine that you work for a company that sends out automated emails and messages to the users who interact with the application more. You can analyze that clickstream data, see what products they are looking at, and see how they use the application or the e-commerce website. You can process that clickstream data, recommend various items, and send out automated coupons, emails, or deals to drive the customer to make a purchase. You probably see this all the time as you interact more with e-commerce style websites. The faster you can consume, process, and do something with the data, the more value there will be. Example: Run real-time analytics on clickstream data and process it within seconds.

**EXAM TIP:** For a given scenario, know which streaming kinesis service to use.

# Lab 4-01: AWS Kinesis Data Stream

## Introduction

### Kinesis Data Streams

You may use Amazon Kinesis Data Streams to create custom

applications that process or analyze streaming data for specific purposes. To an Amazon Kinesis data stream, you may constantly add data from hundreds of thousands of sources, such as clickstreams, application logs, and social media. Within seconds, your Amazon Kinesis Applications will be able to read and analyze data from the stream.

### **AWS CloudFormation**

AWS CloudFormation is a tool that makes it simple for developers and organizations to construct a collection of linked AWS Third-party resources and then provision and manage them logically and reasonably.

Developers may use a simple, declarative approach to deploy and change compute, database, and many other resources, abstracting away the complexities of individual resource APIs. AWS CloudFormation is meant to make resource lifecycle management repeatable, predictable, and safe, with features like automatic rollbacks, automated state management, and resource management across accounts and regions. Multiple ways to generate resources have recently been added, including leveraging AWS CDK for writing in higher-level languages, importing existing resources, and detecting configuration drift. A new registry makes it easier to create custom types that inherit many of CloudFormation's core functionalities.

### **Problem**

Assume you work in an organization as a Data Analytics engineer. Your organization has thousands of users interacting with the organization application. The organization gives you a task to capture real-time data about the users for a marketing campaign and monitor the captured

data in real-time. So, how can you automate this task?

## Solution

The solution is using AWS services to automate your work. You can use Kinesis Data Streams to generate, collect and monitor data. Another helping service that you can use is AWS CloudFormation to create a stack.

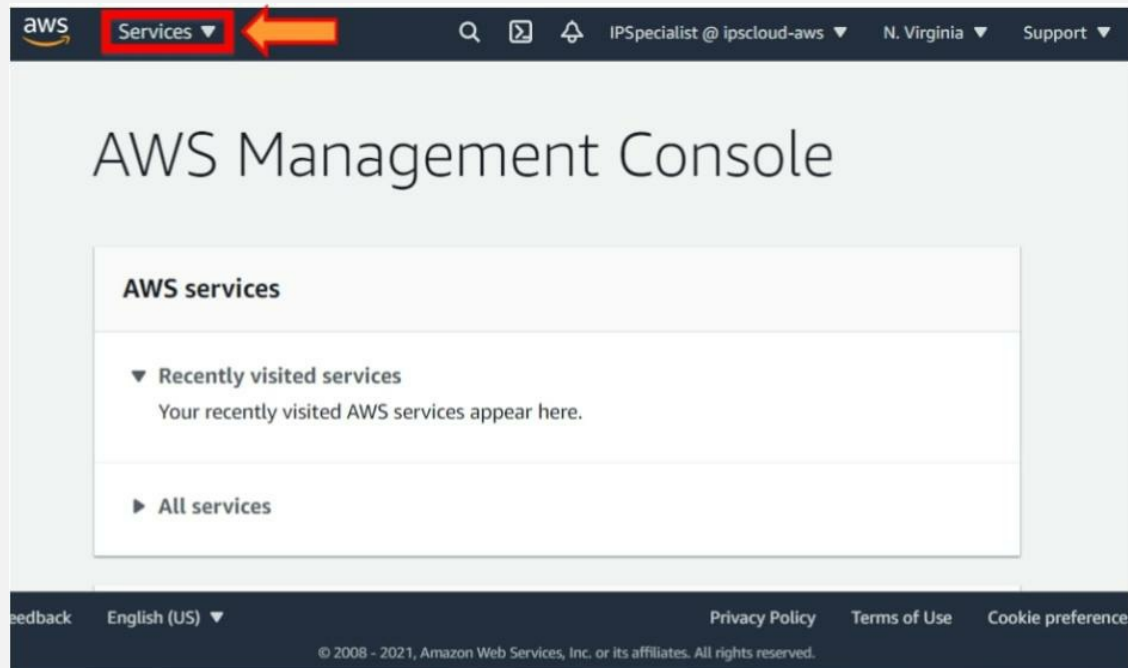
Before starting the lab, the Python script is used to collect the data. The Python code is provided in the following Github link: [https://github.com/ACloudGuru-Resources/Course\\_AWS\\_Certified\\_Machine\\_Learning/blob/master/Cha-record-python-program.py](https://github.com/ACloudGuru-Resources/Course_AWS_Certified_Machine_Learning/blob/master/Cha-record-python-program.py). This Python script is integrated on CloudFormation stack YAML file. In this lab, the only template is used to create a stack and collect the data.

```
1  import requests
2  import boto3
3  import uuid
4  import time
5  import random
6  import json
7
8  client = boto3.client('kinesis', region_name='<INSERT_YOUR_REGION>')
9  partition_key = str(uuid.uuid4())
10
11 # Added 08/2020 since randomuser.me is starting to throttle API calls
12 # The following code loads 500 random users into memory
13 number_of_results = 500
14 r = requests.get('https://randomuser.me/api/?exc=login&results=' + str(number_of_results))
15 data = r.json()["results"]
16
17 while True:
18     # The following chooses a random user from the 500 random users pulled from the API in a single API call.
19     random_user_index = int(random.uniform(0, (number_of_results - 1)))
20     random_user = data[random_user_index]
21     random_user = json.dumps(data[random_user_index])
22     client.put_record(
23         StreamName='<INSERT_YOUR_STREAM_NAME>',
24         Data=random_user,
25         PartitionKey=partition_key)
26     time.sleep(random.uniform(0, 1))
```

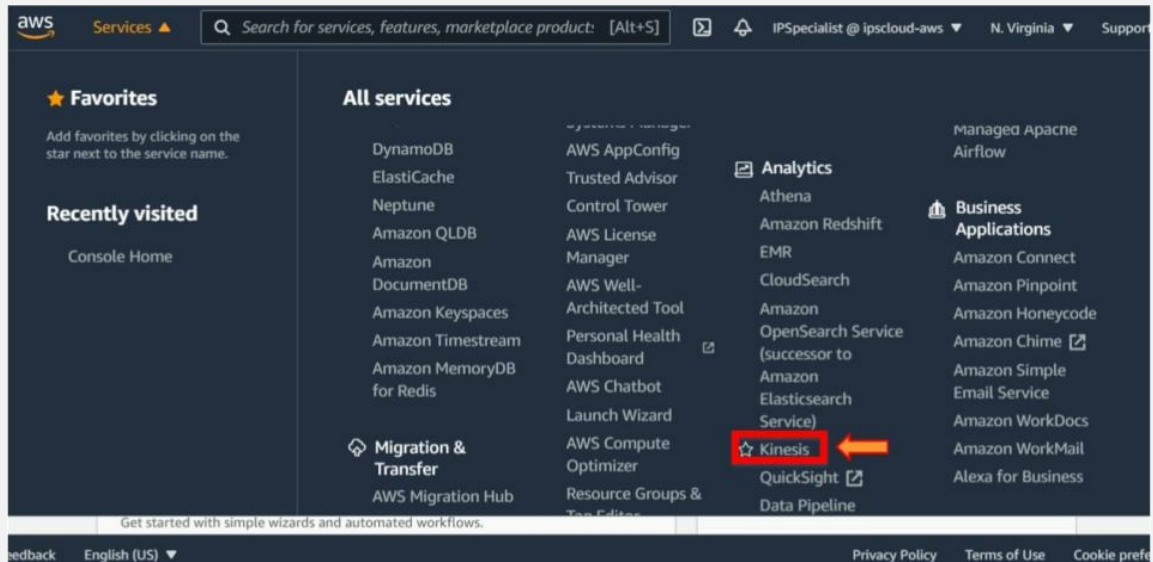
*Figure 4-12: Python Script*

## Step 1: Create Kinesis Data Stream

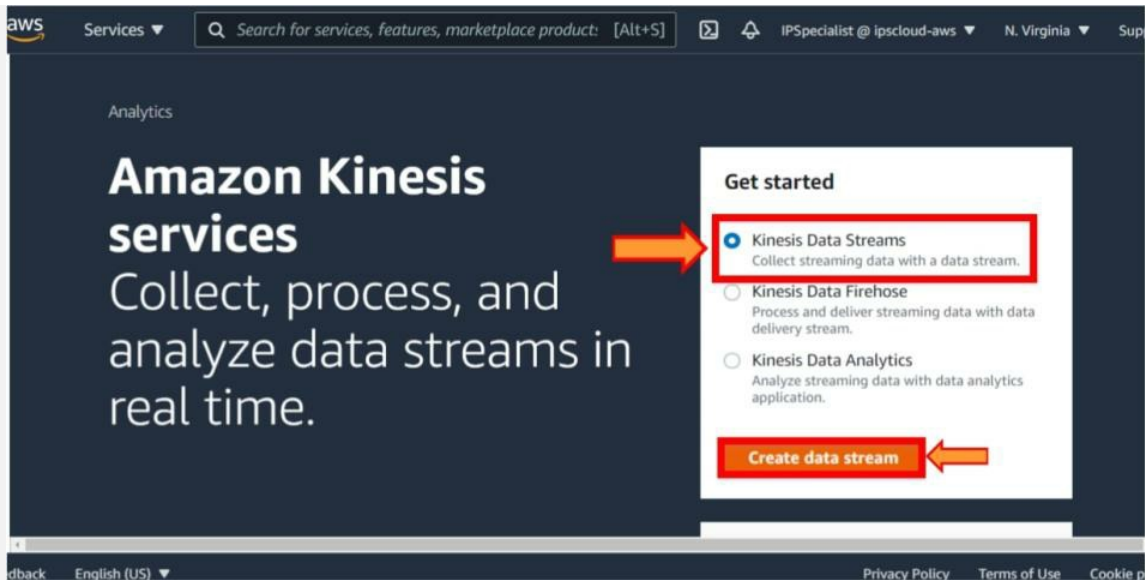
1. Log in to the **AWS Management Console**.
2. Click on the **Services**.



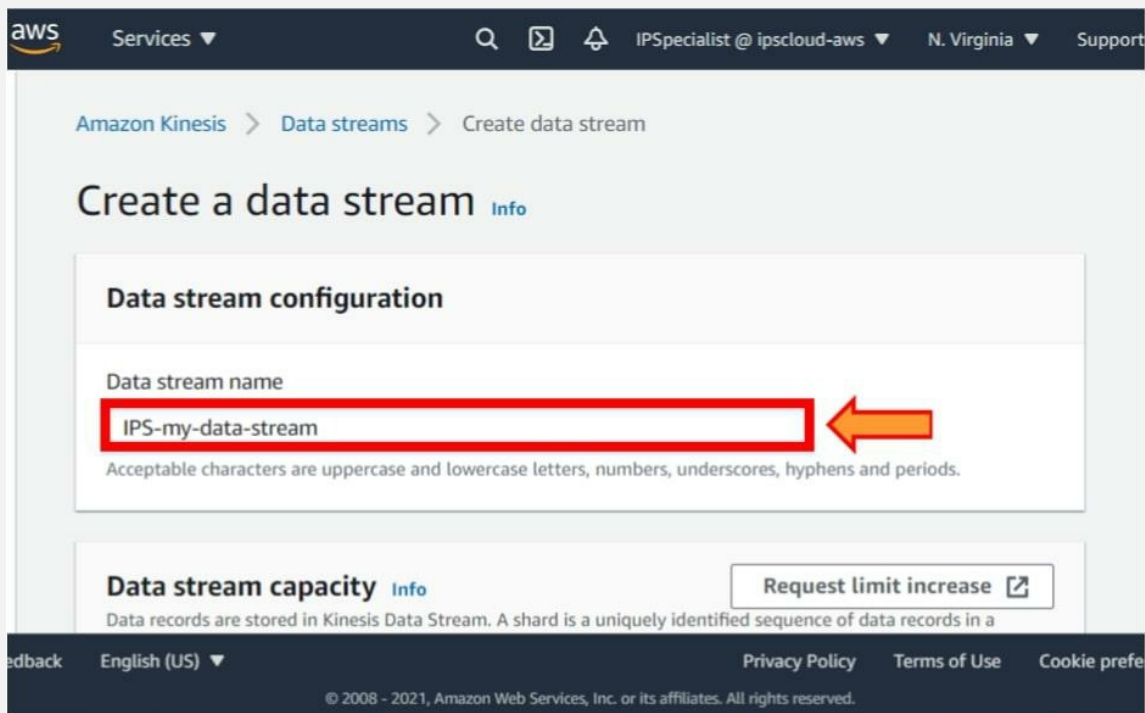
3. Select the **Kinesis** from the **Analytics**.






4. Select the **Kinesis Data Streams**.
5. Click on the **Create Data Stream** button.



6. Give a name **IPS-my-data-stream**.





7. Now to calculate the number of shards, click on the **Shard Estimate**.

aws Services ▾    IPSpecialist @ ipsccloud-aws ▾ N. Virginia ▾ Support

## Data stream capacity [Info](#)

Data records are stored in Kinesis Data Stream. A shard is a uniquely identified sequence of data records in a stream.

[Request limit increase](#) 

**▶ Shard estimator** 

**Number of open shards**  
Each shard ingests up to 1 MiB/second and 1000 records/second and emits up to 2 MiB/second.

Minimum: 1, Maximum: 500, Account limit: 500.

**Total data stream capacity**  
Total data stream capacity is calculated based on the number of shards entered above.

Write  
0 MiB/second, 0 Data records/second

[Feedback](#) English (US) ▾ [Privacy Policy](#) [Terms of Use](#) [Cookie preferences](#)

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

8. Give the following values in **Shard Calculator**:

Average record size: **1 KB**

Max records written: **5 per second**

Number of consumer applications: **1**

9. The result is 'Estimated shards: **1**

aws Services ▾    IPSpecialist @ ipsccloud-aws ▾ N. Virginia ▾ Support

Average record size (in KiB)  
   
Minimum: 1 KiB, maximum: 1024 KiB.

Maximum records written per second  
 

**Reading from the stream**  
Total number of consumers  
 

Estimated number of open shards  
1

Number of open shards

[Feedback](#) English (US) ▾ [Privacy Policy](#) [Terms of Use](#) [Cookie preferences](#)

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

## 10. Set number of shards: 1

aws Services Search for services, features, marketplace product: [Alt+S] IPSpecialist @ ipscloud-aws N. Virginia

Apply this value Cost calculator

**Number of open shards**  
Each shard ingests up to 1 MiB/second and 1000 records/second and emits up to 2 MiB/second.

1

Minimum: 1, Maximum: 500, Account limit: 500.

**Total data stream capacity**  
Total data stream capacity is calculated based on the number of shards entered above.

**Write**  
1 MiB/second, 1000 Data records/second

**Read**  
2 MiB/second

Cancel Create data stream

Feedback English (US) Privacy Policy Terms of Use

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

## 11. Click on the **Create Data Stream** button.

aws Services Search for services, features, marketplace product: [Alt+S] IPSpecialist @ ipscloud-aws N. Virginia

Apply this value Cost calculator

**Number of open shards**  
Each shard ingests up to 1 MiB/second and 1000 records/second and emits up to 2 MiB/second.

1

Minimum: 1, Maximum: 500, Account limit: 500.

**Total data stream capacity**  
Total data stream capacity is calculated based on the number of shards entered above.

**Write**  
1 MiB/second, 1000 Data records/second

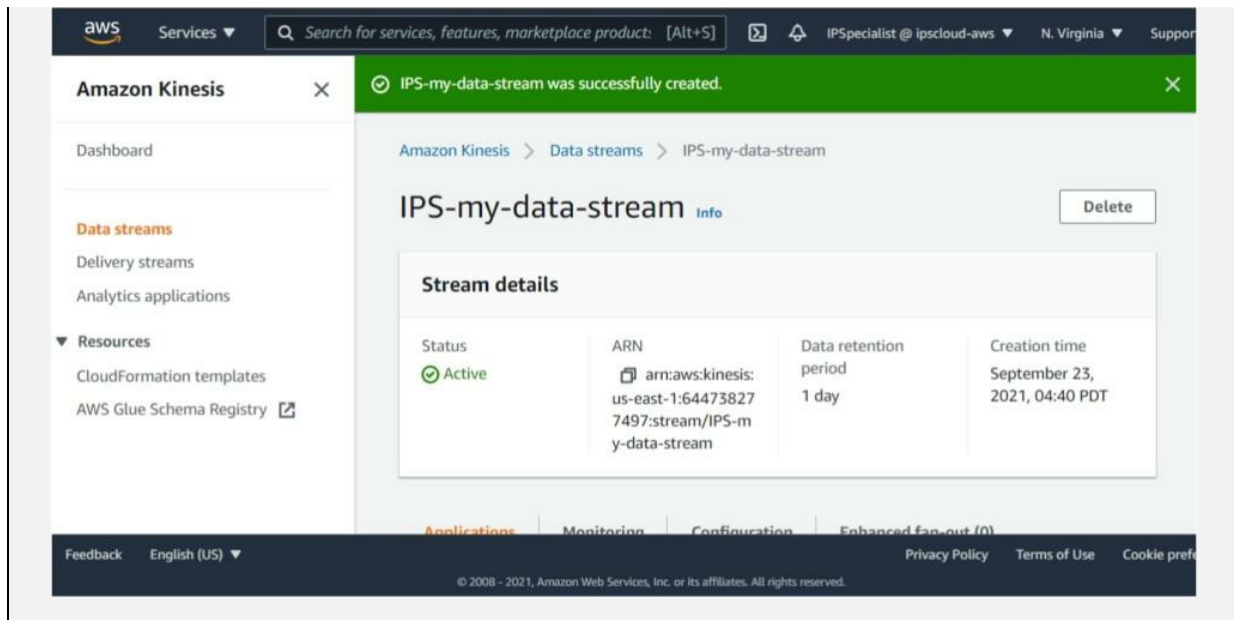
**Read**  
2 MiB/second

Cancel Create data stream

Feedback English (US) Privacy Policy Terms of Use

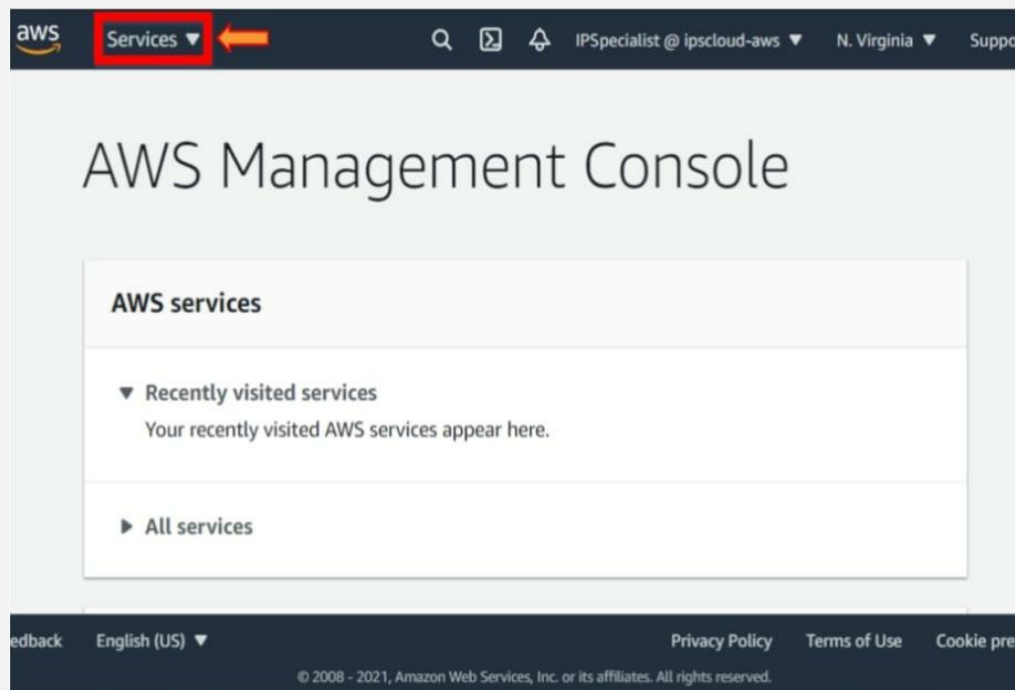
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

## 12. Hence, a Kinesis data stream is created.

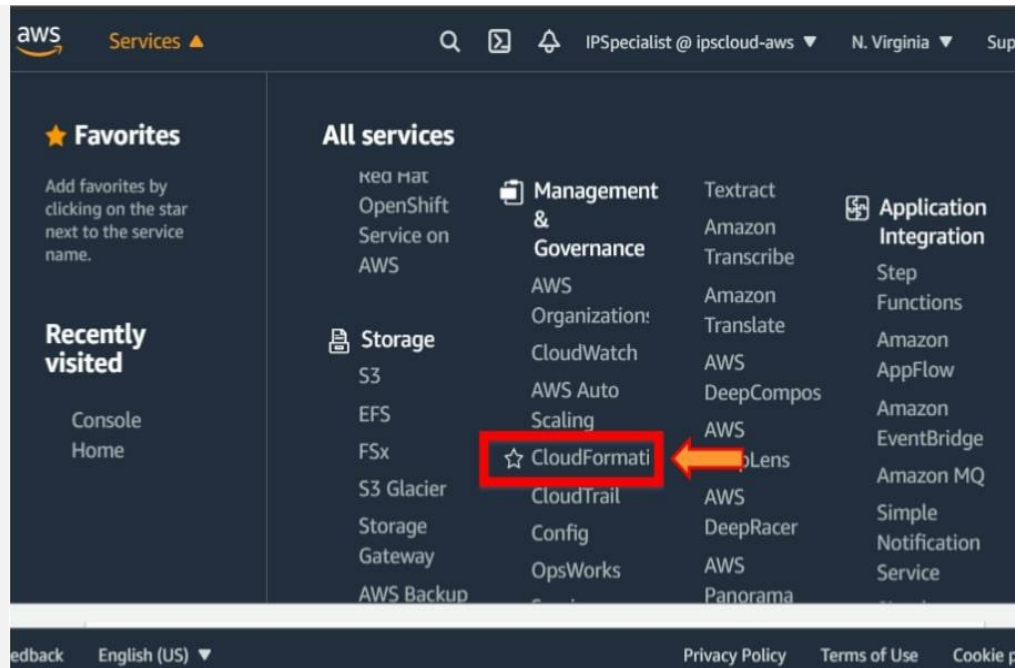


## Step 2: Create CloudFormation Stack

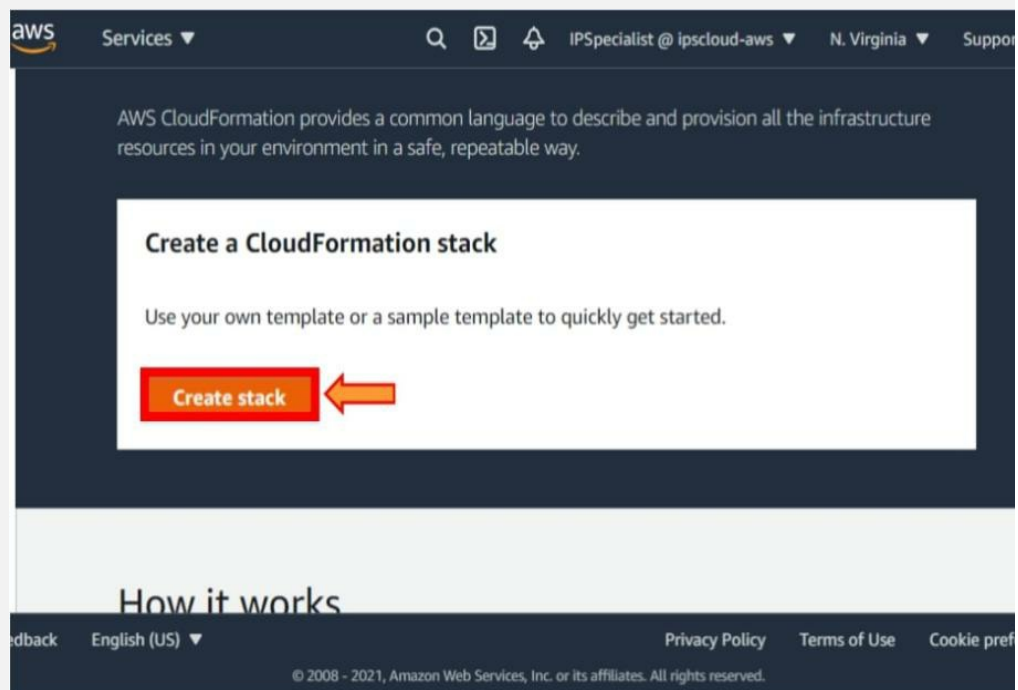
1. Click on the **Services**.



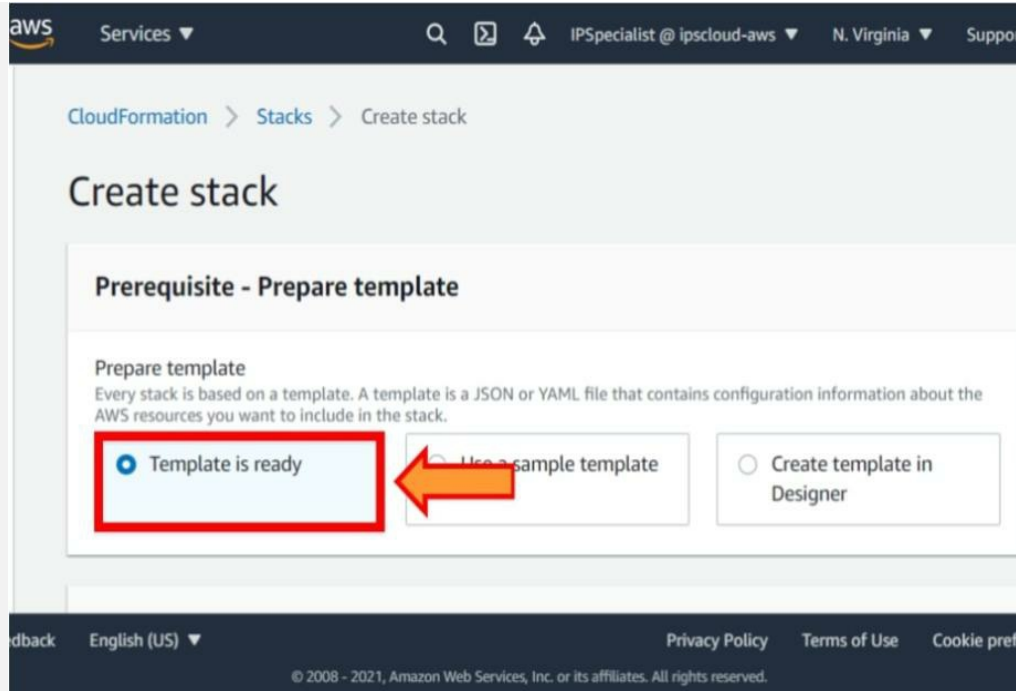
2. Select the **CloudFormation** from the **Management & Governance**.



3. Click on the **Create Stack** button.



4. Select **Template is ready**.



5. Download a template file provided in the following Github link:  
[https://raw.githubusercontent.com/ACloudGuru-Resources/Course\\_AWS\\_Certified\\_Machine\\_Learning/master/Cdata-producer.yml](https://raw.githubusercontent.com/ACloudGuru-Resources/Course_AWS_Certified_Machine_Learning/master/Cdata-producer.yml).

```
Parameters:
  KinesisDataStream:
    Description: The name of your Kinesis Data Stream.
    Type: String
Mappings:
  RegionMap:
    us-east-1:
      AMI: ami-0080e4c5bc078760e
    us-east-2:
      AMI: ami-0cd3dfa4e37921605
    us-west-1:
      AMI: ami-0ec6517f6edbf8044
    us-west-2:
      AMI: ami-01e24be29428c15b2
Resources:
  LnDataProducerInstance:
    Type: AWS::EC2::Instance
    Properties:
      InstanceType: t2.micro
      ImageId:
        Fn::FindInMap:
          - RegionMap
          - !Ref AWS::Region
          - AMI
      IamInstanceProfile: !Ref LnInstanceProfiler
      UserData:
        Fn::Base64:
          !Join [ "", [
            "#!/bin/bash -xe\n",
            "sudo /opt/aws/bin/cfn-init -v ", #use cfn-init to install packages in cloudformation :
            !Sub "--stack ${AWS::StackName} ",
            "--resource LnDataProducerInstance ",
            "--configsets InstallAndConfigure ",
            !Sub "--region ${AWS::Region}"
          ] ]
```

6. Select the **Upload a template file**.

aws

Services ▾

IPSpecialist @ ipscloud-aws ▾

N. Virginia ▾

Support ▾


## Specify template

A template is a JSON or YAML file that describes your stack's resources and properties.

### Template source


Selecting a template generates an Amazon S3 URL where it will be stored.

☐ Amazon S3 URL



☒ Upload a template file

### Upload a template file

Choose file 

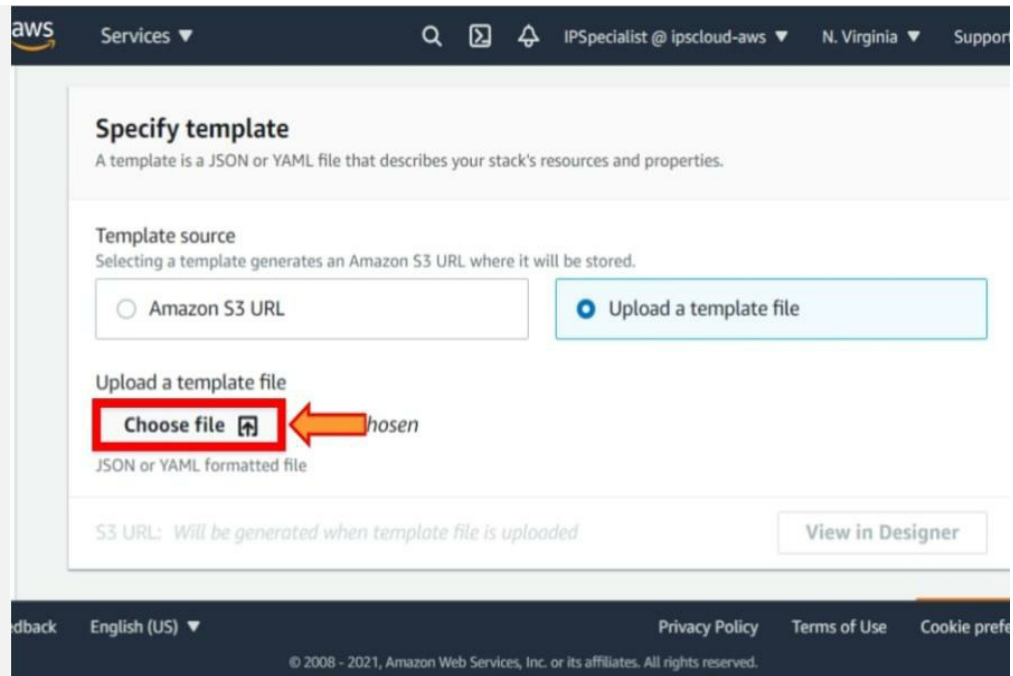
No file chosen

JSON or YAML formatted file

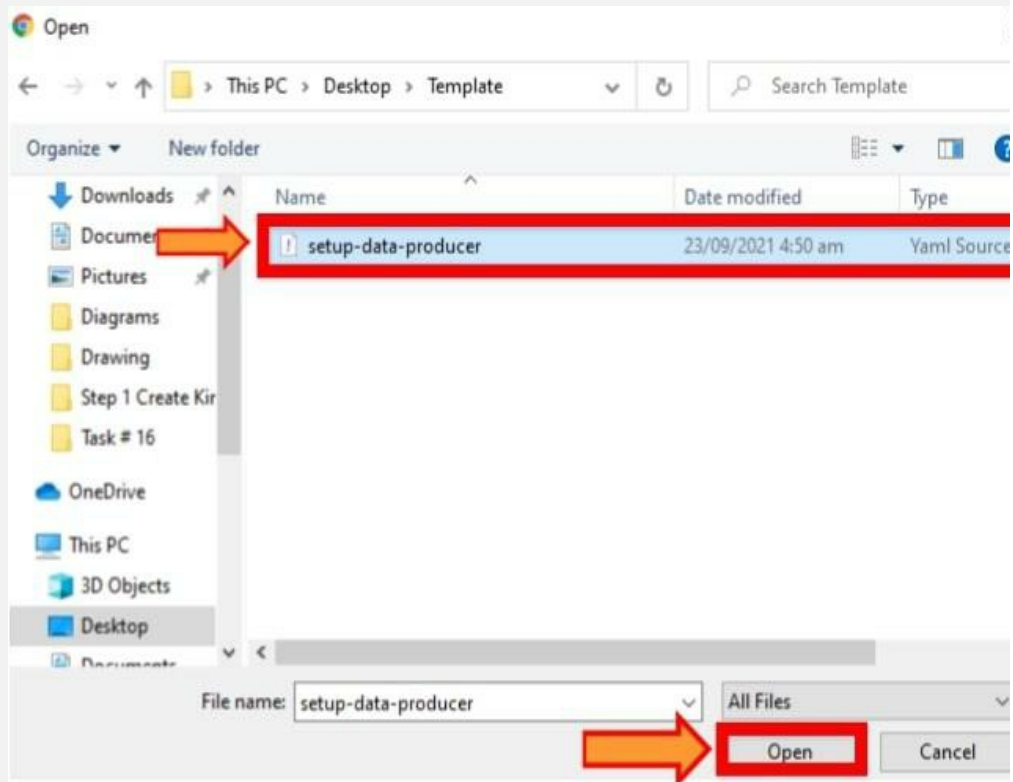
S3 URL: Will be generated when template file is uploaded

View in Designer

7. Click on the **Choose file** button.



8. Select the **setup-data-producer.yml** file. Then Click on the **Open**



9. Click on the **Next** button.

aws Services ▾

Template source  
Selecting a template generates an Amazon S3 URL where it will be stored.

☐ Amazon S3 URL ☒ Upload a template file

Upload a template file  
Choose file setup-data-producer.yml  
JSON or YAML formatted file

S3 URL: https://s3-external-1.amazonaws.com/cf-templates-1taywmeqps8cu-us-east-1/20212666s0-setup-data-producer.yml View in Designer

Next

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

10. Give a stack name **IPS-data-producer-stack**.

aws Services ▾

CloudFormation > Stacks > Create stack

Specify stack details

Stack name

Stack name  
Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters  
Parameters are defined in your template and allow you to input custom values when you create or update a stack.

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

11. Then give Kinesis data stream name: **IPS-my-data-stream**.

aws Services

CloudFormation > Stacks > Create stack

## Specify stack details

**Stack name**

Stack name

IPS-data-producer-stack

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

**Parameters**

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

12. Click on the **Next** button.

aws Services

Stack name

IPS-data-producer-stack

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

**Parameters**

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

**KinesisDataStream**

The name of your Kinesis Data Stream.

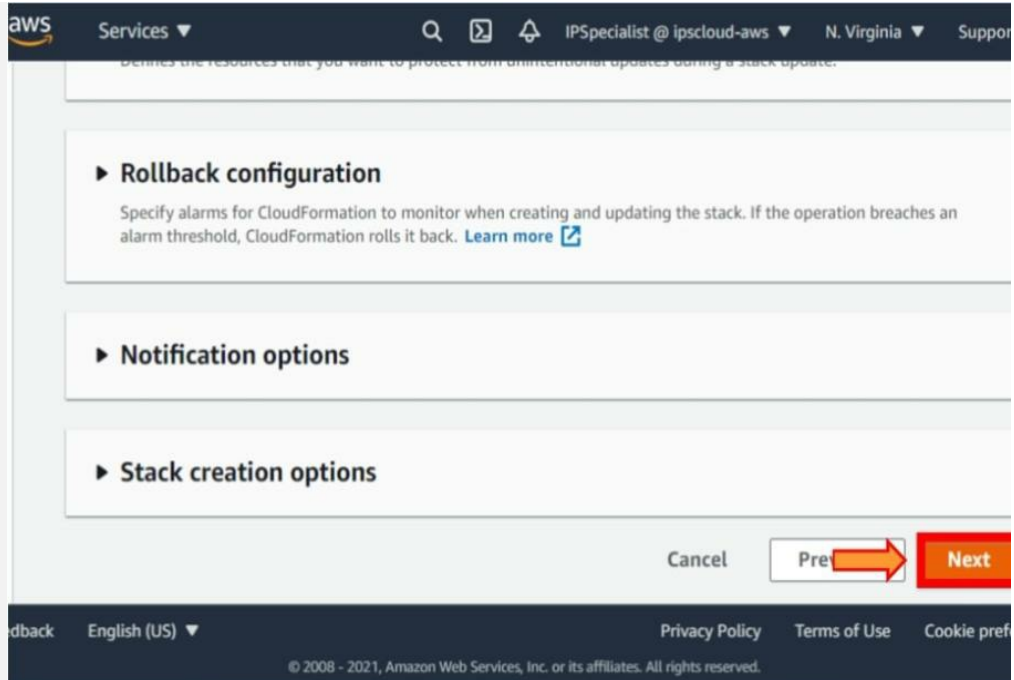
IPS-my-data-stream

Cancel Previous **Next**

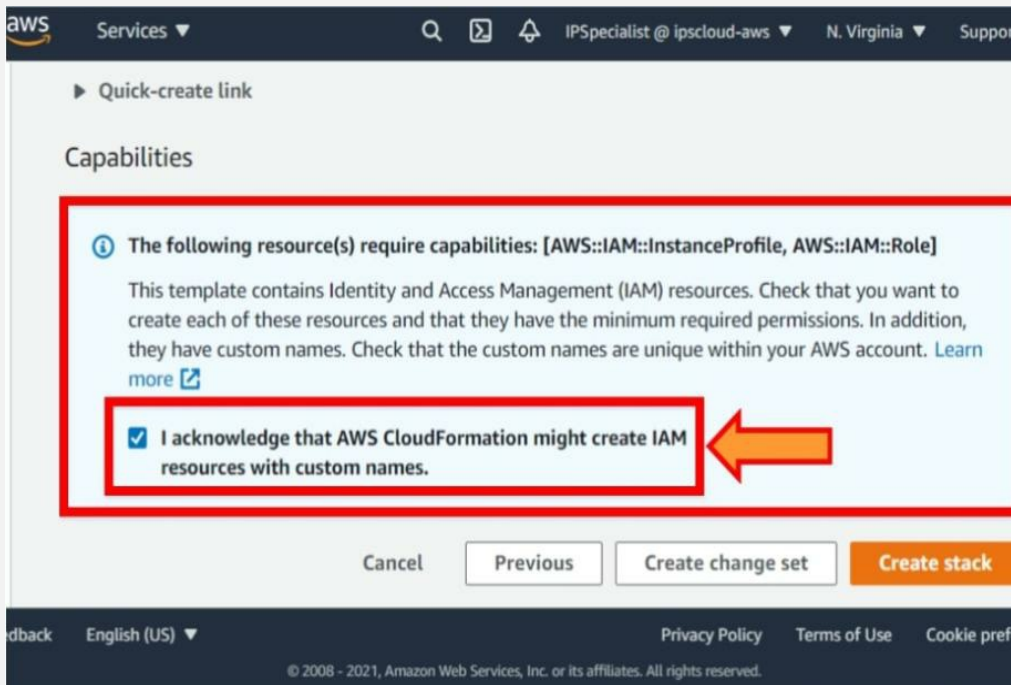
Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

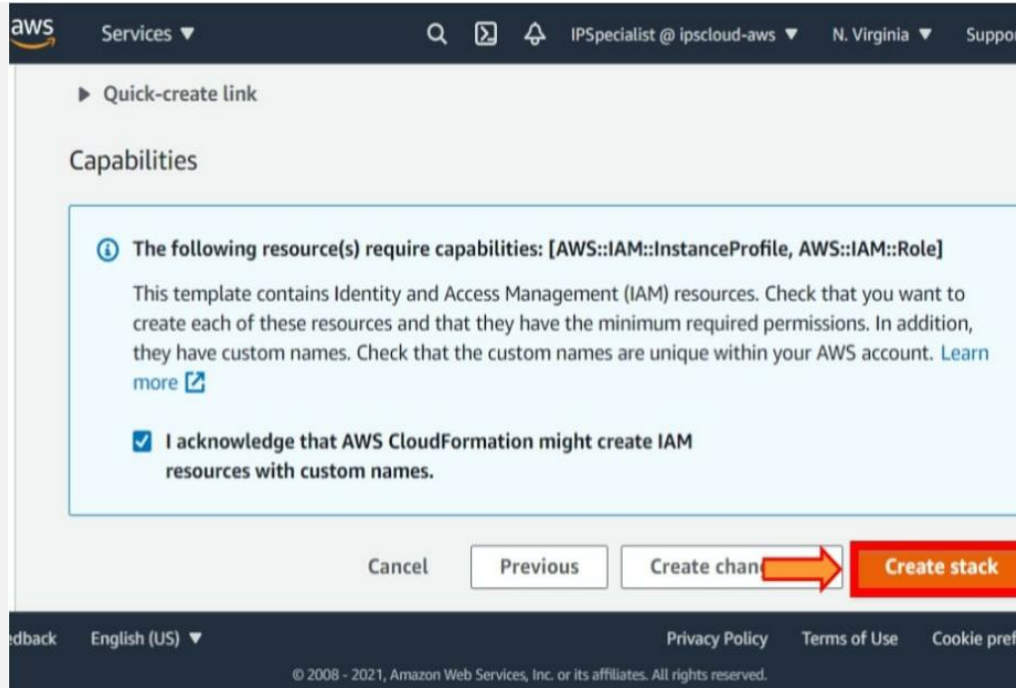
13. Scroll down. Click on the **Next** button.



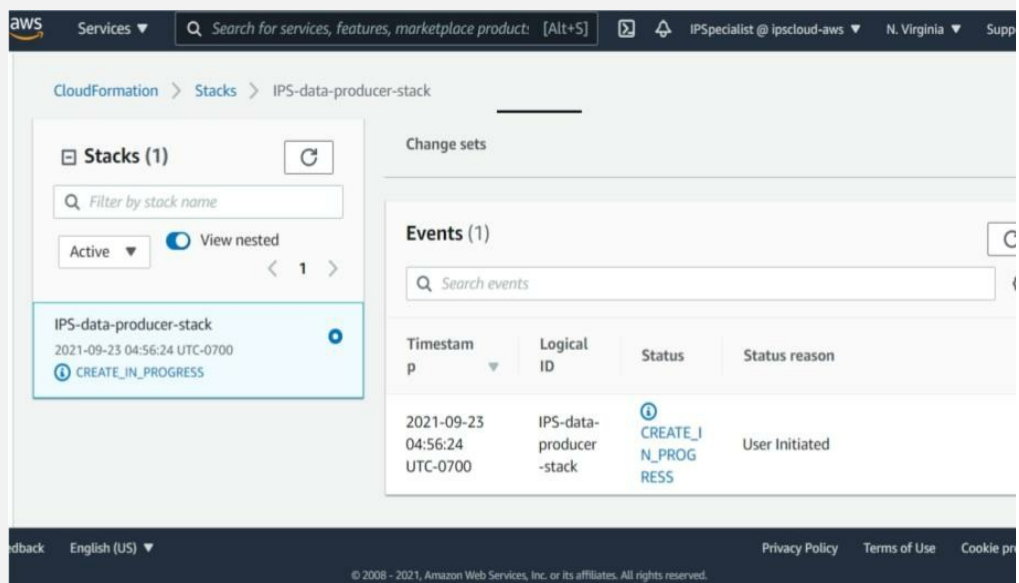
14. Scroll down. Click on the **check box**.



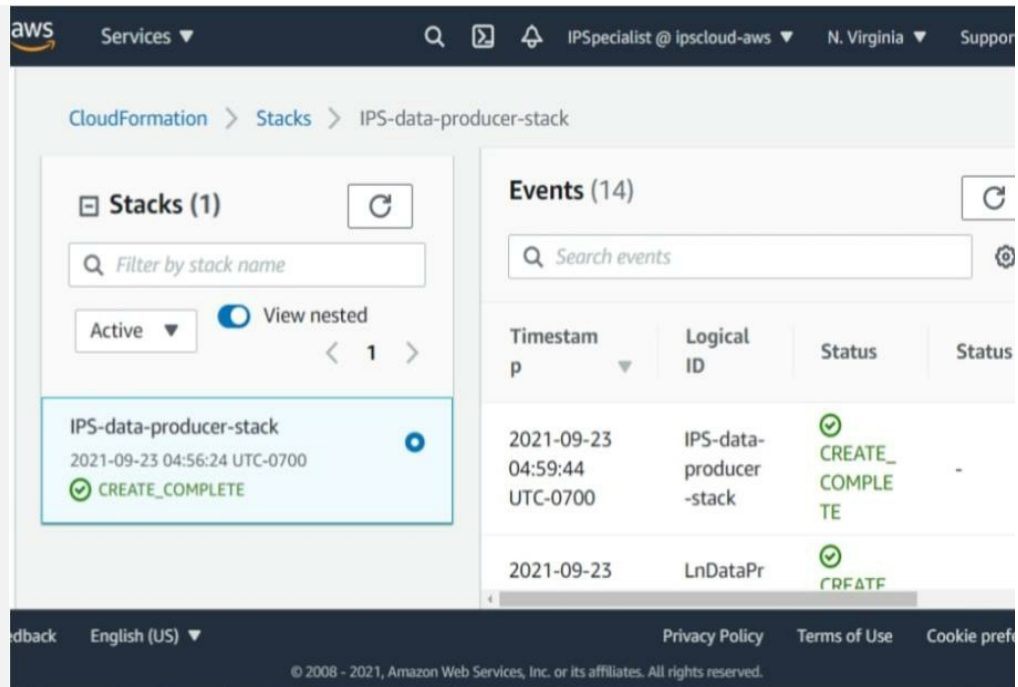
15. Then Click on the **Create Stack** button.



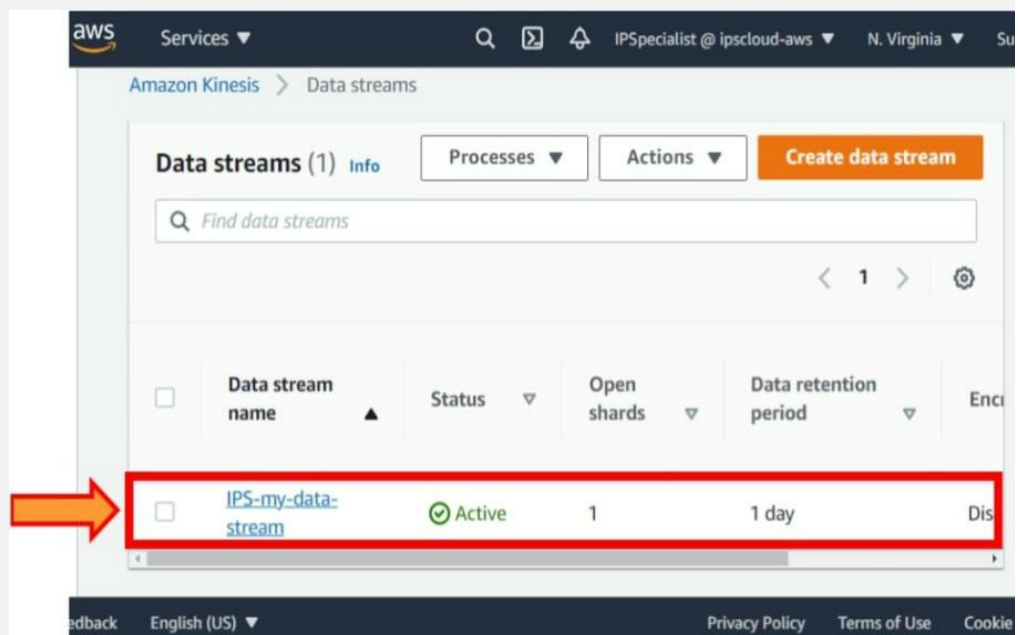
16. It will take a few minutes to create a stack. Once it CloudFormation will create all the services and automatically st random user data into the Kinesis stream.



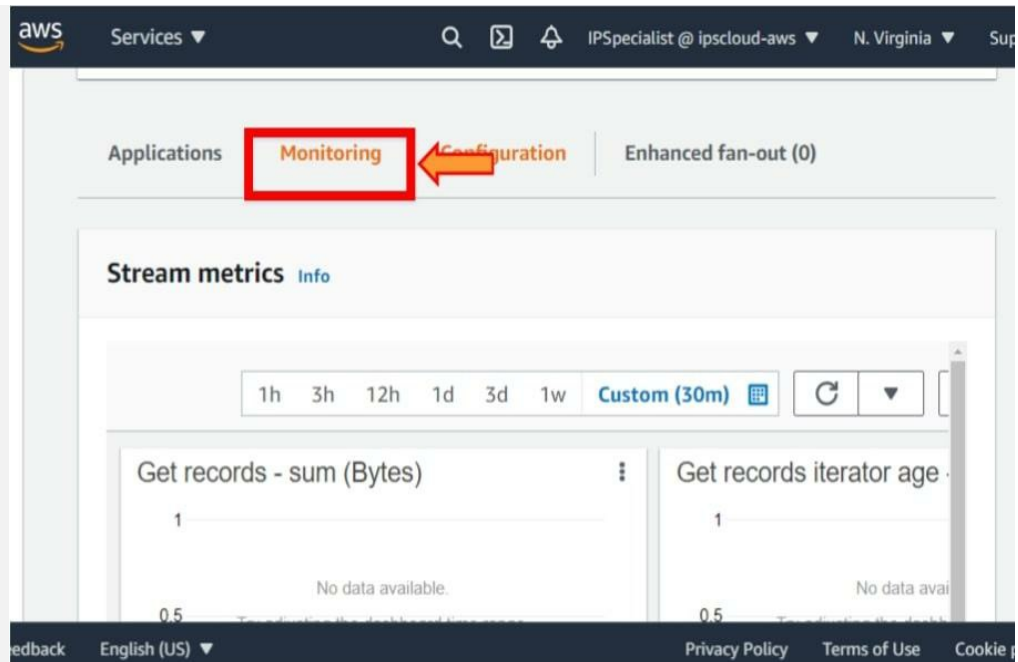
17. Hence, CloudFormation Stack work is completed.



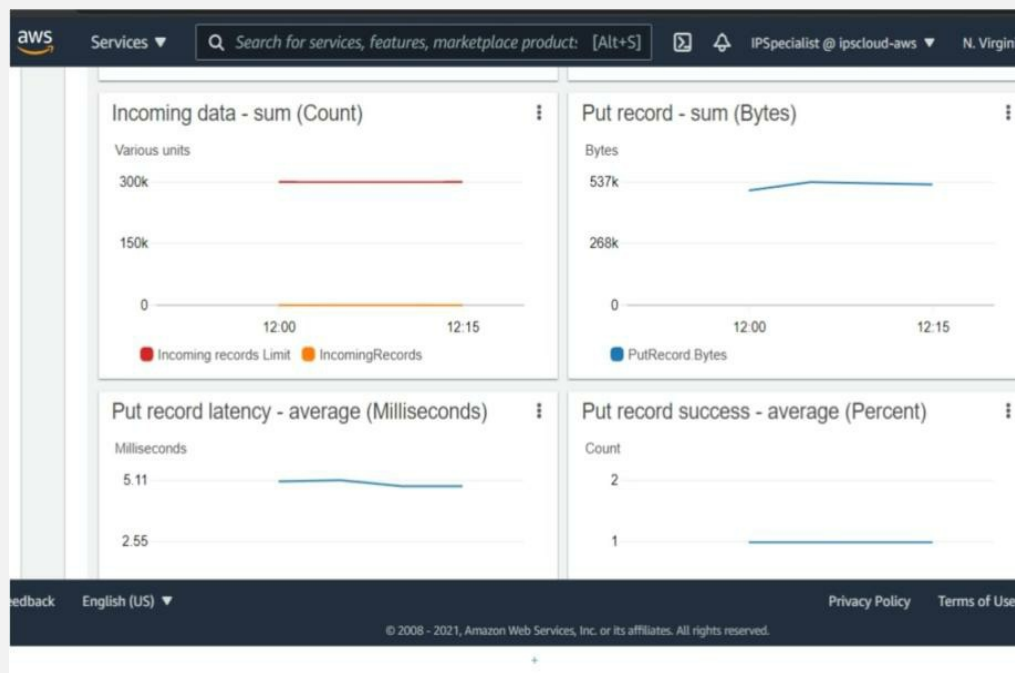
18. Now go back to the **Kinesis Data Stream** dashboard, and then **IPS-my-data-stream**.



19. Click on the **Monitoring** tab.



20. You will see different graphs.



# Kinesis Data Firehose

## Introduction

Amazon Kinesis Data Firehose is a fully managed service that delivers real-time streaming data to Amazon S3, Amazon Redshift, Amazon OpenSearch Service (Amazon ES), Splunk, and any custom HTTP endpoint or HTTP endpoints owned by supported third-party service providers like Datadog, Dynatrace, LogicMonitor, MongoDB, New Relic, and Sumo Log. Kinesis Data Firehose, Kinesis Data Streams, Kinesis Video Streams, and Amazon Kinesis Data Analytics are part of the Kinesis streaming data platform. You do not need to build applications or manage resources using Kinesis Data Firehose. You set up your data producers to transmit data to Kinesis Data Firehose, and the data is delivered automatically to the destination you specify. Kinesis Data Firehose may also be configured to alter data before sending it.

Kinesis Firehose is similar to Kinesis Data Streams. You get data from Data Producers, which can be things like application log files from EC2 instances or servers. Real-time clickstream data from an e-commerce website could be things like IoT devices or sensors on manufacturing devices. Any streaming data can be considered a data producer. With Kinesis Data Firehose, you no longer have to worry about shards. Thus, you can pre-process the data using AWS Lambda, which acts like an ETL service right before you land data on some data store. With Kinesis Data Firehose, you can land data onto RedShift, S3, Amazon Elastic Search, and Splunk. Pre-processing the data with Lambda is optional, so you do not have to use Lambda to pre-process the data, and you could stream the data directly from Data Firehose to storage.

Firehose is going to be your delivery service for streaming data. You can also take advantage of S3 Events, which allow us to call a Lambda

function when an event happens on an S3 bucket. For example, if you have some streaming data that lands on an S3 bucket, you can create an S3 event, which calls a Lambda function that pushes that data onto DynamoDB. The different places where Kinesis Data Firehose can output your data onto or deliver your data onto are RedShift, S3, Splunk, and Amazon Elastic Search. It allows you to easily stream data to a final data source or a final destination. Kinesis Data Streams support shards and data retention, so if your processing tools fail, Kinesis Data Streams will preserve the data and allow you to reprocess it. With Kinesis Data Firehose, you do not have to worry about shards since it is mainly used for streaming data directly into some storage or data repository like S3. With Kinesis Data Streams, storing the data is optional. You could just run some analytics on it and then get rid of the data and never keep it. But with Kinesis Data Firehose, our end goal is to store our data somewhere in AWS like S3.

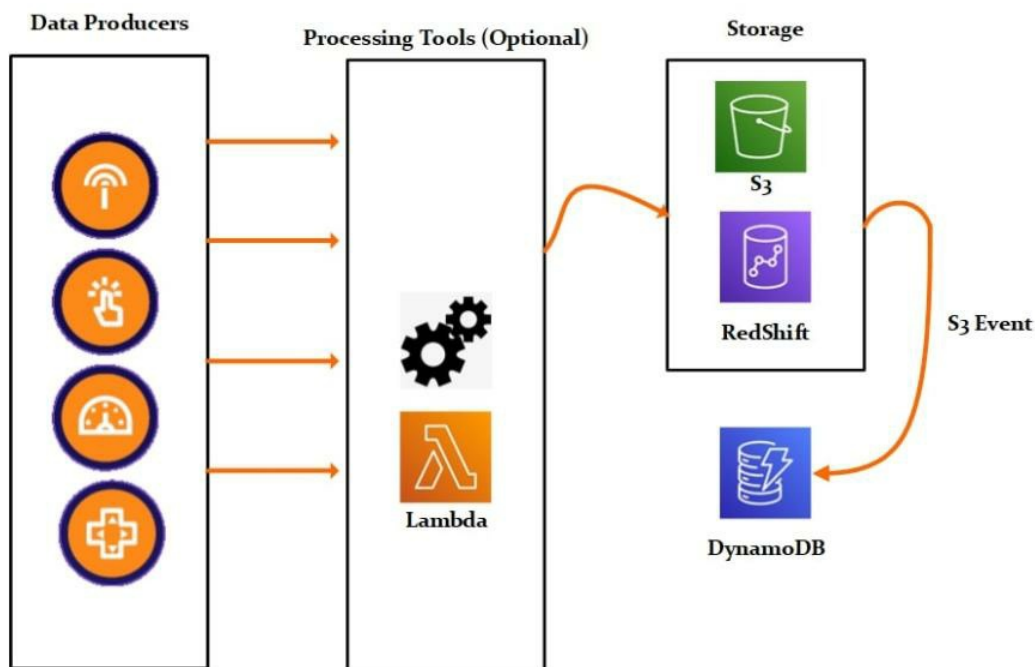


Figure 4-13: Kinesis Firehose

Kinesis Data Firehose takes data and delivers it to some destination. Using Kinesis Data Firehose, you can deliver data to S3, Redshift, Elasticsearch, or Splunk instances.

**EXAM TIP:** Kinesis Data Firehose is a solution for streaming ETL. It is the most convenient method for loading streaming data into data warehouses and analytics tools. It is capable of capturing, transforming, and loading streaming data into Amazon S3.

## AWS Kinesis Firehose Key Concepts

Understanding the following ideas will help you get started with Kinesis Data Firehose:

### 1. Kinesis Data Firehose Delivery Stream:

Kinesis Data Firehose is an underlying entity. You utilize Kinesis Data Firehose by first generating a delivery stream and then delivering data to it.

### 2. Record:

Your data producer provides the relevant data to a Kinesis Data Firehose delivery stream. A record can be up to 1,000 KB in size.

### 3. Data Producer:

Records are sent to Kinesis Data Firehose delivery streams by producers. A data producer is, for example, a web server that delivers log data to a delivery stream. You can also set up your Kinesis Data Firehose delivery stream to read data from an existing Kinesis data stream and put it into destinations automatically.

### 4. Buffer Size & Buffer Interval:

Before sending data from Kinesis to destinations, Firehose buffers incoming streaming data to a specific size or for a certain amount of time. Buffer size is measured in megabytes, while buffer interval is measured in seconds.

## Kinesis Firehose Data Flow

Streaming data is transmitted to your Amazon S3 bucket for Amazon S3 destinations. You can optionally backup source data to another Amazon S3 bucket if data transformation is enabled.

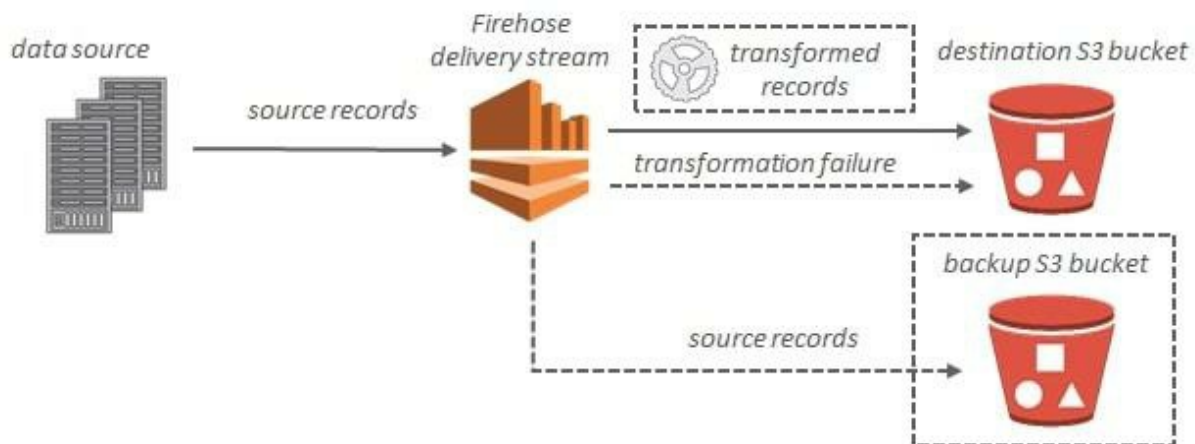


Figure 4-14: Kinesis Firehose Data Flow S3 Bucket

Streaming data is transmitted to your S3 bucket first for Amazon Redshift destinations. Kinesis Data Firehose then sends the Amazon Redshift COPY command to load data from your S3 bucket to your Amazon Redshift cluster. You can optionally backup source data to another Amazon S3 bucket if data transformation is enabled.

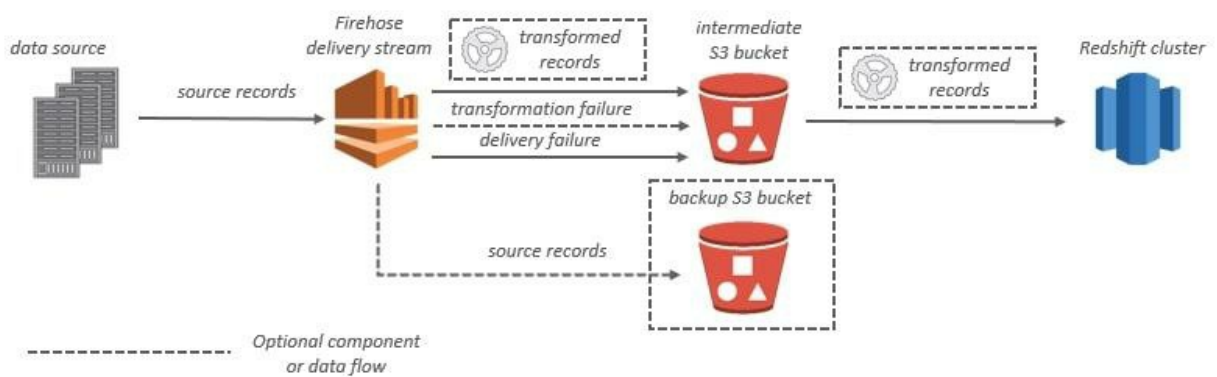


Figure 4-15: Kinesis Firehose Data Flow Redshift Cluster

Streaming data is sent to your Amazon ES cluster and may optionally be backed up to your S3 bucket simultaneously for Amazon ES destinations.

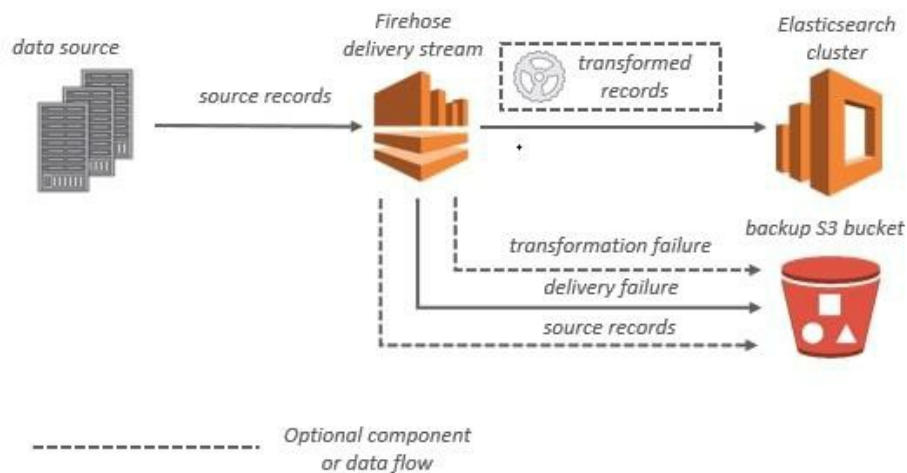
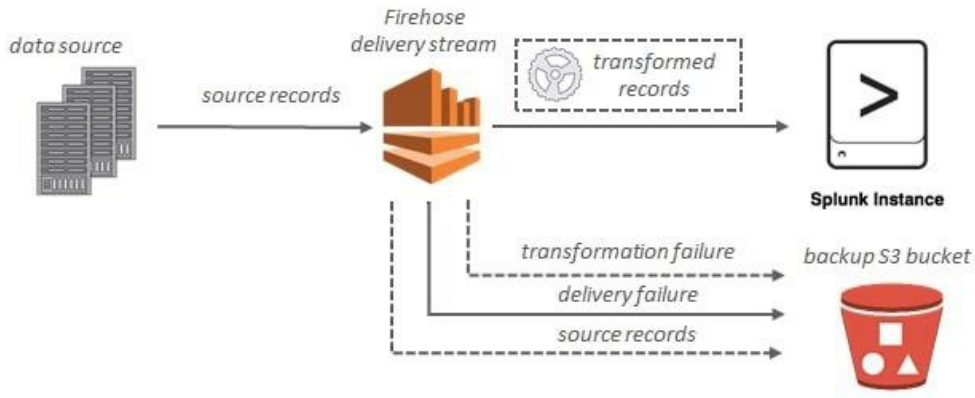


Figure 4-16: Kinesis Firehose Data Flow ElasticSearch Cluster

Streaming data is provided to Splunk for Splunk destinations, and it may optionally be backed up to your S3 bucket simultaneously.



*Figure 4-17: Kinesis Data Flow Splunk Instance*

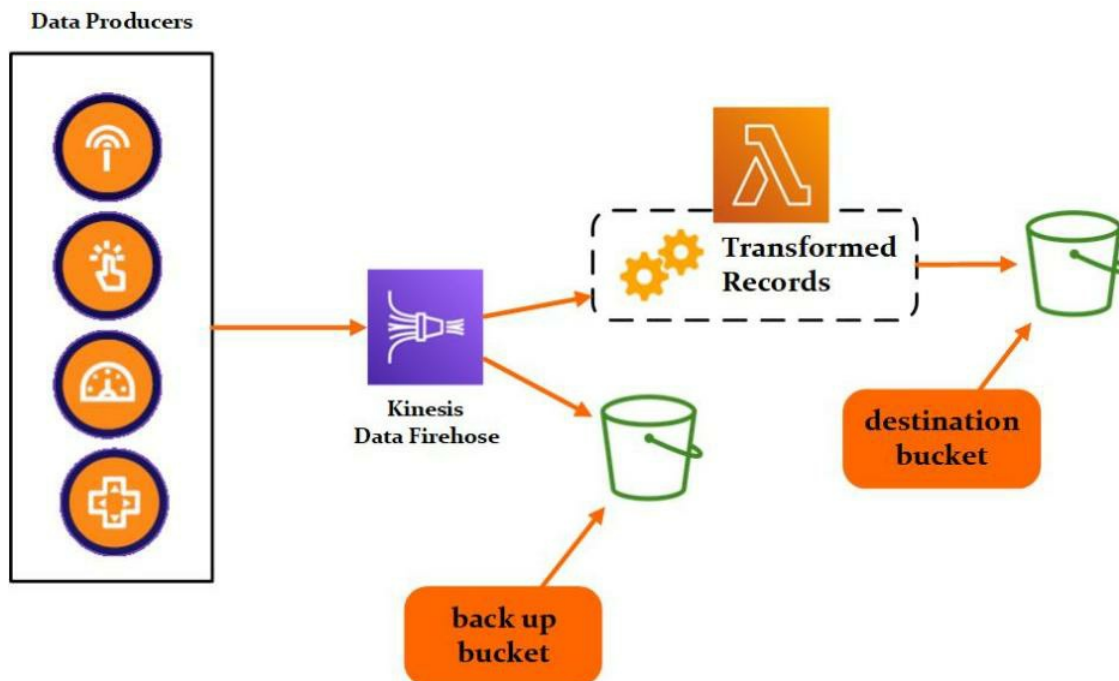
**EXAM TIP:** Understand how to get data from public or in-house data sets and load it into AWS.

## S3 Destination

The S3 as a destination is similar to Kinesis Data Streams. You have data producers, which are the applications that are producing streaming data. With Firehose, you can have Kinesis Data Streams, be a data producer and send that data to Kinesis Data Firehose. However, you have your data producers, and these data producers produce records. But data records can be as large as a thousand kilobytes, and once Data Firehose receives the records, it can then ship them off to S3.

You can just send a data producer via the Firehose to the bucket with no modifications, transformations, or anything else that can be readily accomplished with Kinesis Data Firehose. You can also intercept and transform that record or do some type of manipulation on the streaming data record, typically done with AWS Lambda. It integrates directly with Kinesis Data Firehose to take that record and then

transform it before loading it onto the S3 bucket. You can also introduce another step before you transform those records and store the original records into a backup bucket. Then you can have your other bucket with the transformed records be your destination bucket. These are various ways that you can take data producers. You can take our streaming data, feed it through Kinesis Data Firehose, transform it if you want, and then ship it off to S3.



*Figure 4-18: S3 Destination*

**EXAM TIP:** Know the different ways to upload into S3 by using the console, the S3 API, or the AWS CLI.

## Redshift Destination

The next destination is Redshift. With Redshift, you have our data producers, and you can send the data through Kinesis Data Firehose.

The data will always go to S3 first and then to Redshift. Hence, what happens is that the streaming data is delivered to an S3 bucket first, and then Firehose automatically issues the copy command to load the data from S3 to Redshift. Anytime you load data into Redshift, it first gets loaded into S3, and then it issues a copy command to load that data from S3 to Redshift. You can do the same thing by intercepting records, transforming them using AWS Lambda, loading them onto the S3 bucket, and then eventually using the copy command to load it onto Redshift. You can also intercept here and transform the records as well. Hence, you have a few different options to transform our records before you load them onto Redshift. You can always have a backup bucket if you want to and store off that data before you load onto Redshift or before you transform those records. As a result, Redshift just knows that it is the data warehousing solution that AWS offers.

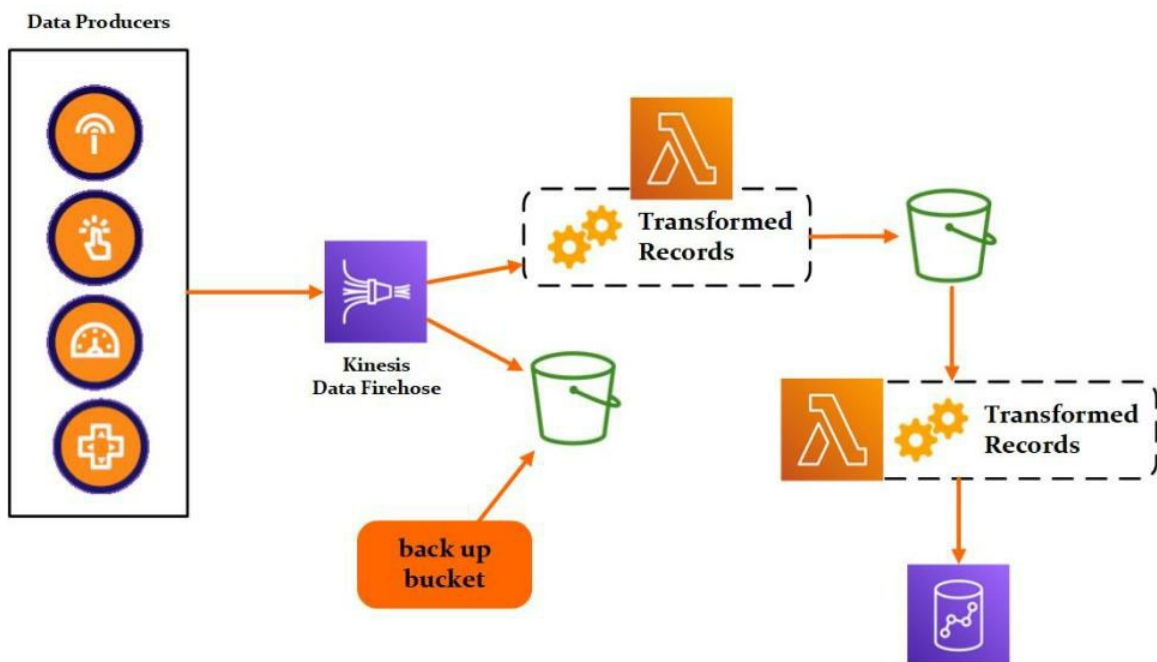
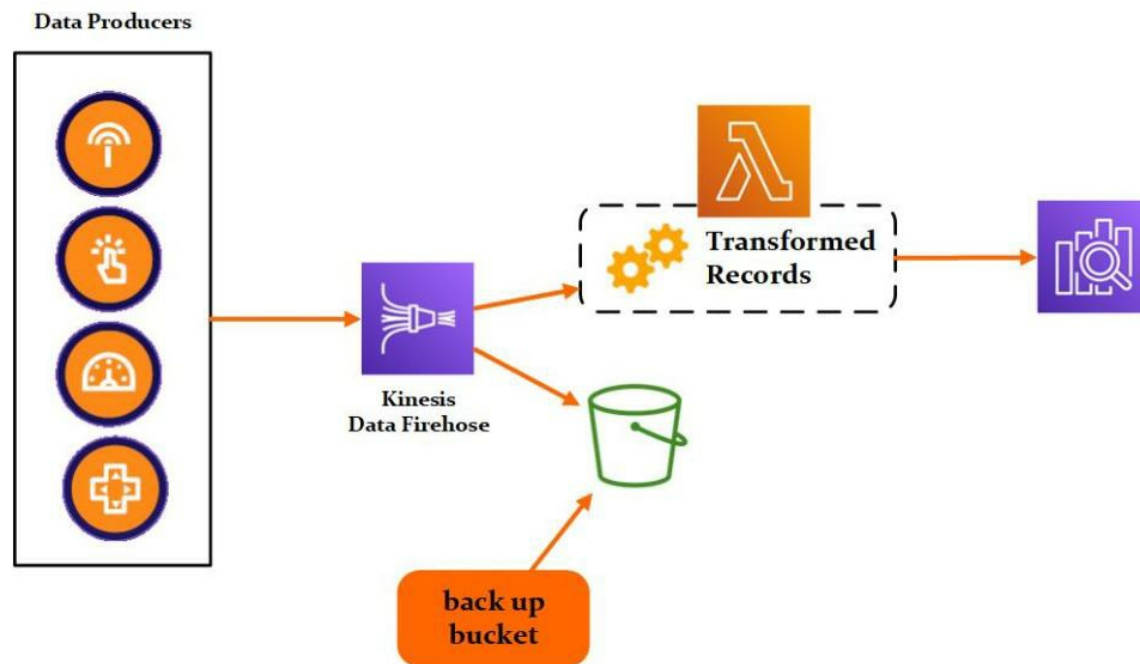


Figure 4-19: Redshift Destination

## Elasticsearch Destination

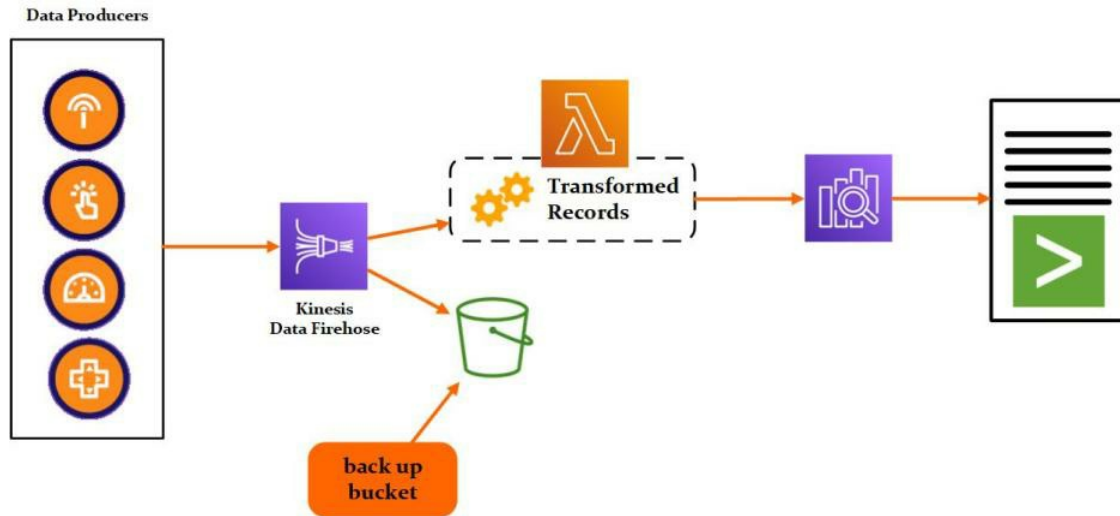
Similar to the other destinations, you have our data producers that go through Kinesis Data Firehose. You can then load it into Elasticsearch and transform the records before they load onto our Elasticsearch cluster. Similarly, you can load the data before converting it onto Elasticsearch into a backup bucket in the S3 bucket.



*Figure 4-20: Elasticsearch Destination*

## Splunk Destination

The last destination is Splunk instances. Splunk is a way to aggregate your log files from servers or applications to have a single place where you can have all your log files aggregated. It is important to know that it is a destination for Firehose. Hence, just like other destinations, you have our data producers. It goes through Kinesis Data Firehose, where you can load the data off into our Splunk instances and transform the records as well. Similarly, you can load that data into an S3 bucket as a backup before you transform it or load it into our Splunk instances.



*Figure 4-21: Splunk Destination*

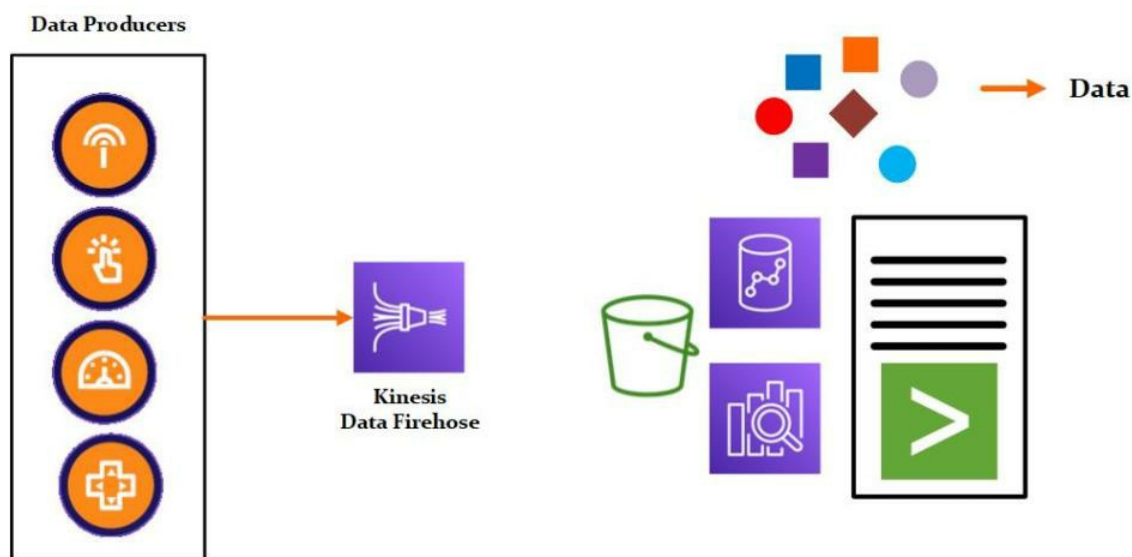
**Note:** AWS recently added a new destination. This can be a generic HTTP endpoint since this was added in mid to late 2020. Kinesis Data Firehose is buffered and aggregated before it is shipped off as the data is coming. It enables you to use a fully managed delivery HTTP endpoint rather than building a custom application for data delivery infrastructure. It also opens up the opportunity for key integrations to services like DynamoDB, SNS, RDS, and API Gateway.

## Buffer Size & Buffer Interval

You have all sorts of data producers. It might be clickstream, gaming, IoT, or any streaming data. You can feed that through Kinesis Data Firehose. You have data coming in aggregated, which then finally shipped off to our destination. Hence, Amazon Kinesis Firehose buffers the incoming streaming data to a certain size for a certain period before delivering it to a destination. The buffer size for S3 as a destination

spans from 1 to 128 megabytes. In contrast, Elasticsearch ranges from 1 megabyte to 100 megabytes.

The data will be shipped off whenever the buffer sizes or buffer interval is hit. Let us assume the buffer interval is hit. It will ship off the data to its final destination, where the buffer interval is in seconds, ranging from 60 seconds to 900 seconds. It is important to note that in circumstances where the data delivery to the destination is falling behind, the actual data is written to the stream. Firehose will raise the buffer size dynamically to catch up, just to make sure that all the data is delivered to its destination.



*Figure 4-22: Buffer Size & Buffer Interval*

## Kinesis Firehose Use Cases

Following are some use cases of AWS Kinesis Firehose that can be used in real life:

### 1. Stream & Store Data from Devices:

Kinesis Data Firehose allows us to stream and store data from all kinds of devices easily. Think of IoT devices, embedded systems, and

customer applications where you can store it all off into some destination, data lake, or other data warehousing solution. For example, let us imagine you work for a software team that builds smart IoT thermostats that go inside your home to help you control the temperature of your lovely home. Your team has set up a Kinesis Data Firehose to push important data into an S3 data Lake, such as temperature readings, the user's region, and the location of where they live. You can start to answer important questions like thermostat changes during a particular part of the morning or a specific part of the evening. For example, what is a typical household thermostat set to during the winter months for a user in the Northern European countries? Since Kinesis Data Firehose stores all of the important data into an S3 data Lake, you can query that data and answer important questions. It can help build better marketing strategies and suggestions on energy-saving techniques for your customers in a similar region. Example: Capturing important data from IoT devices, embedded systems, and consumer applications to store it into a data lake.

## **2. Creating ETL Jobs on Streaming Data:**

You can also create ETL streaming jobs on the data coming in through Kinesis Data Firehose. Take the same smart thermostat reading data, which you can transform it using an ETL job. You can store that data off into different tables into a data warehousing solution like Redshift. It means that various teams (let us assume the marketing team, the engineering team, and the executive team) can run specific queries they need to answer for their particular analytical questions. Hence, you no longer have to sift through unstructured data that is in our Data Lake. You can split out certain tables or permission and IAM policies to

different teams you have within your organization. Hence, they can run their specific queries on your data analytics or your data warehousing solution. Accordingly, Kinesis Data Firehose is a great tool to use if you want to deliver that data to a Data Lake, a data warehouse, or store off that data. For example, before data is stored in a data warehousing system, ETL tasks run on streaming data.

# **Demo: Kinesis Data Firehose**

## **Introduction**

### **Kinesis Data Firehose**

Kinesis Data Firehose is a solution for streaming ETL. It is the most convenient method for loading streaming data into data storage and analytics tools. It can collect, transform, and load streaming data into Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, and Splunk. It enables near-real-time analytics with your existing business intelligence tools and dashboards. It is a fully managed service that scales automatically to meet your data flow and does not require any ongoing management. It may also batch, compress, and encrypt data before loading it, reducing storage requirements at the destination and enhancing security.

### **Kinesis Data Streams**

You may use Amazon Kinesis Data Streams to create custom applications that process or analyze streaming data for specific purposes. To an Amazon Kinesis data stream, you may constantly add data from hundreds of thousands of sources, such as clickstreams, application logs, and social media. Within seconds, your Amazon Kinesis Applications will be able to read and analyze data from the stream.

### **Amazon Simple Storage Service S3**

Amazon S3 is a type of object storage that allows you to store and recover any quantity of data from any location. It is a low-cost storage solution with business resilience, reliability, efficiency, privacy, and infinite expansion.

Amazon S3 is a web service that allows you to store and retrieve an

infinite quantity of data from any place and at any time. You may quickly create projects that integrate cloud-native storage using this service. Because Amazon S3 is easily customizable and you only pay for what you use, you can start small and scale up as needed without sacrificing performance or dependability.

Amazon S3 is also built to be highly adaptable. Instead of finding out how to store their data, Amazon S3 allows developers to focus on innovation like building a simple FTP system or a complex web application like the Amazon.com retail website. You can read the same piece of data a million times or only for emergency disaster recovery and store whatever type and amount of data you desire.

#### **AWS CloudFormation**

AWS CloudFormation is a tool that makes it simple for developers and organizations to construct a collection of linked AWS and third-party resources and then provision and manage them logically and reasonably.

Developers may use a simple, declarative approach to deploy and change compute, database, and many other resources, abstracting away the complexities of individual resource APIs. AWS CloudFormation is meant to make resource lifecycle management repeatable, predictable, and safe, with features such as automatic rollbacks, automated state management, and resource management across accounts and regions. Multiple ways to generate resources have recently been added, including leveraging AWS CDK for writing in higher-level languages, importing existing resources, and detecting configuration drift. A new Registry makes it easier to create custom types that inherit some of CloudFormation's core functionalities.

## Problem

Assume you work in an organization as Data Analytics. Suppose your organization wanted to stream and store data directly from devices. You can capture essential data from IoT devices, embedded systems, or things like customer applications and store that into a storage destination. So, how can you automate this task?

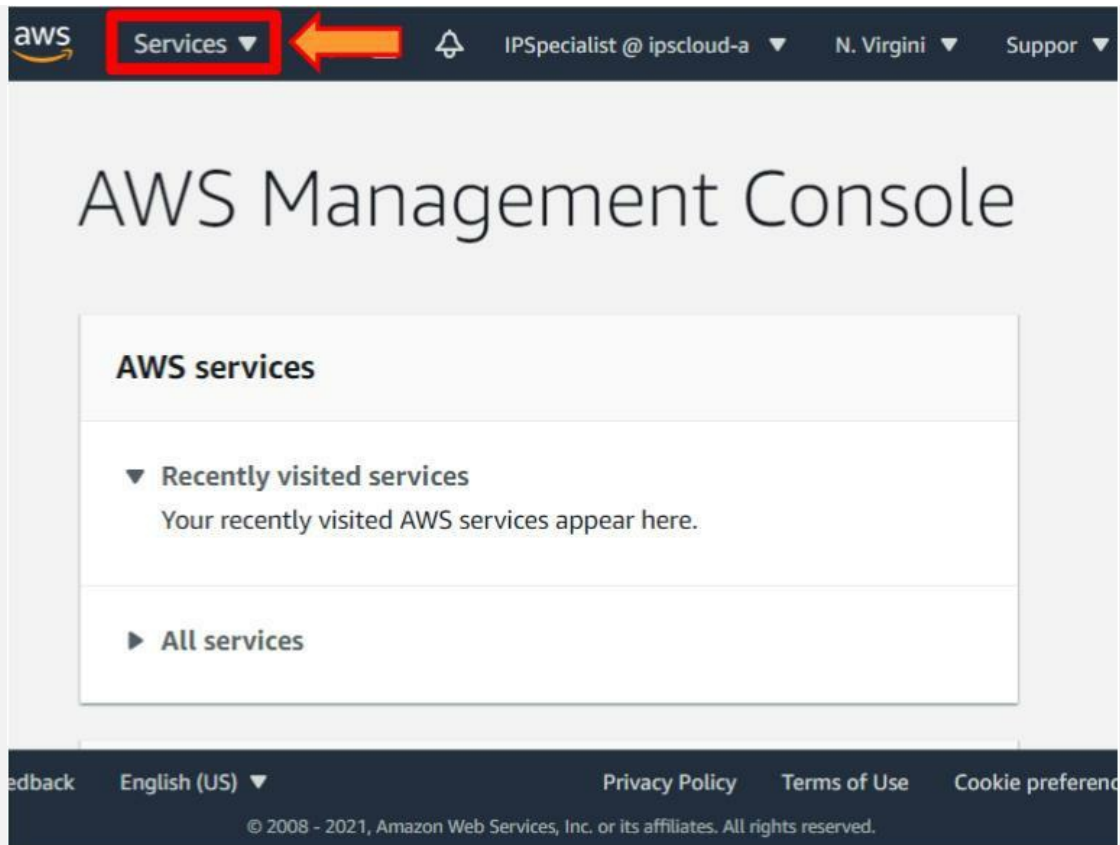
## Solution

The solution is using AWS services to automate your work. You can use Kinesis Data Streams to generate or collect data. You should use Kinesis Data Firehose to store data at any storage destination for storing real-time data. The other helping service that you can use is AWS CloudFormation to create a stack. An S3 bucket is used as a storage destination to store the streaming data.

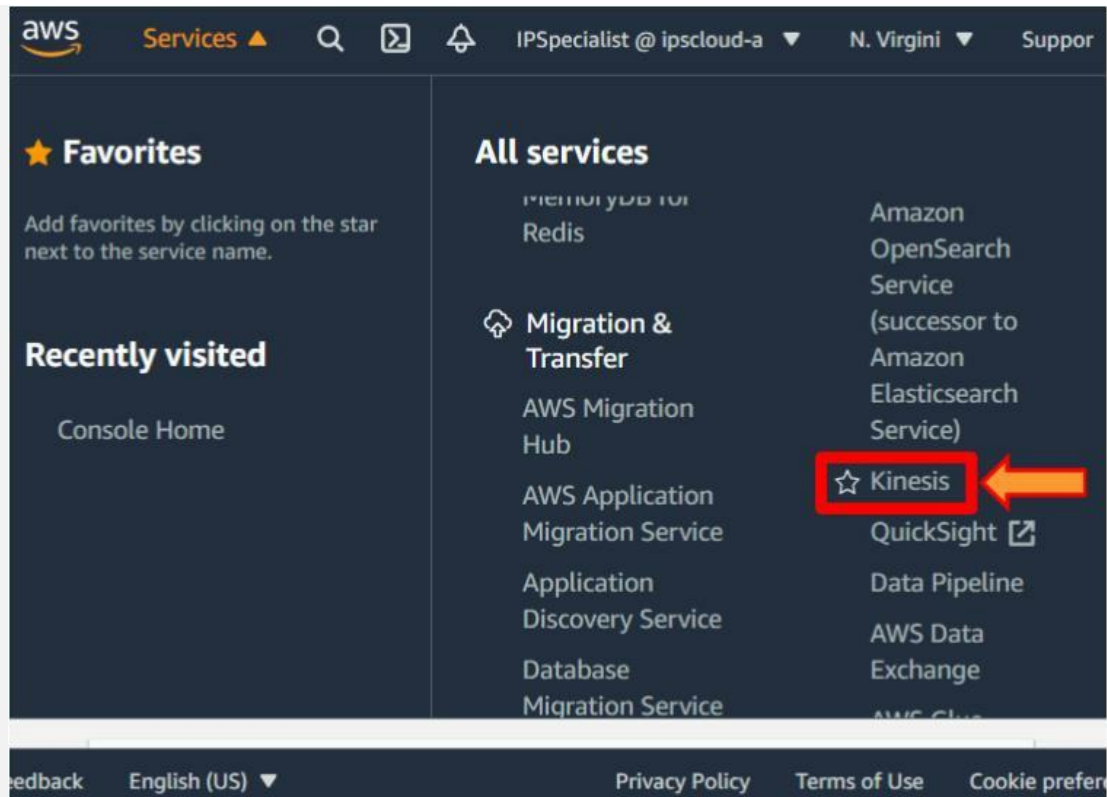
**Note:** Before starting this demo, make sure to create AWS Kinesis Data Streaming and AWS CloudFormation stack, as it is used in this lab and as mentioned in the above demo of Kinesis Data Stream.

### Step 1: Create Kinesis Data Firehose

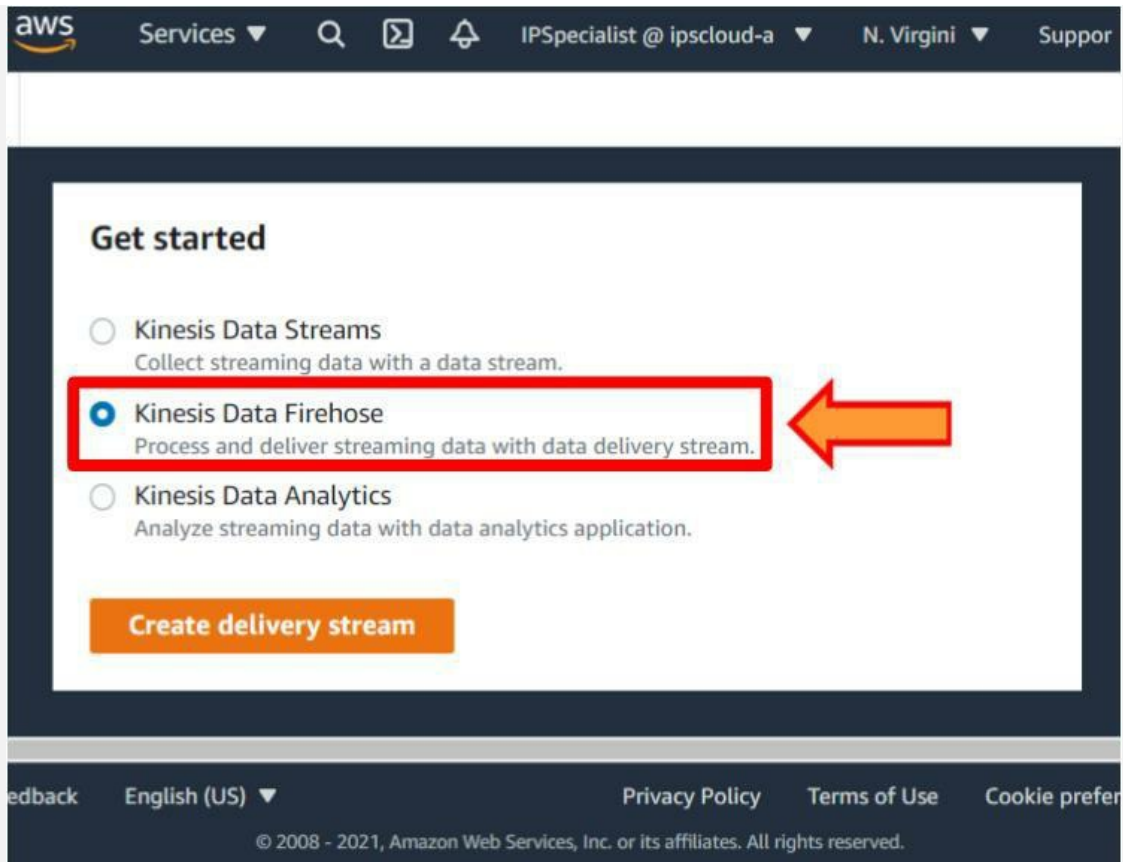
1. Log in to the AWS Console.
2. Click on the **Services**.



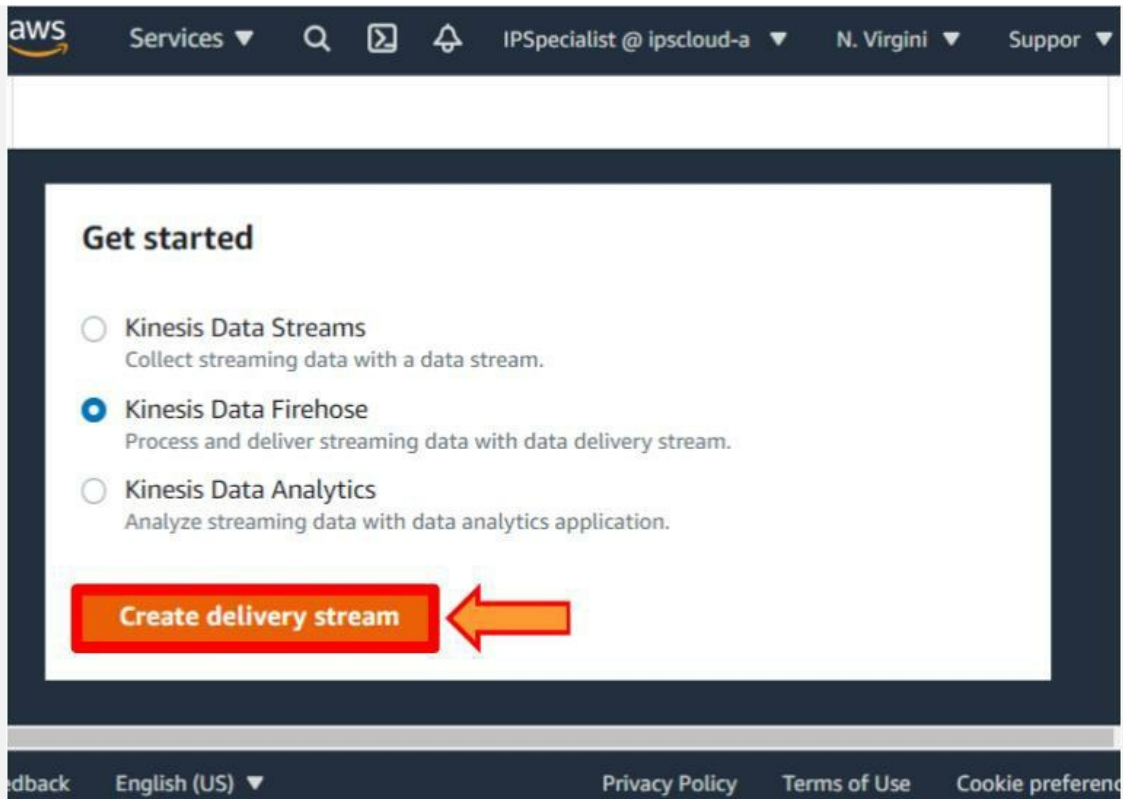
3. Select the **Kinesis** from the **Analytics**.



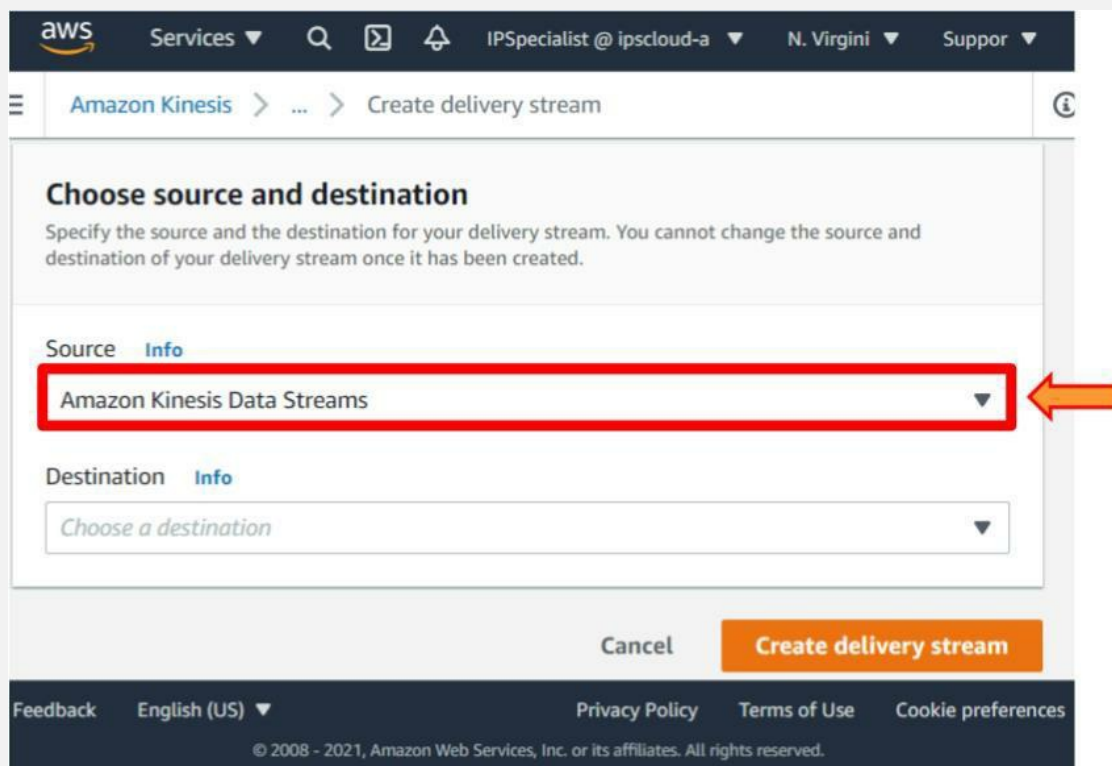
4. Select the **Kinesis Data Firehose**.



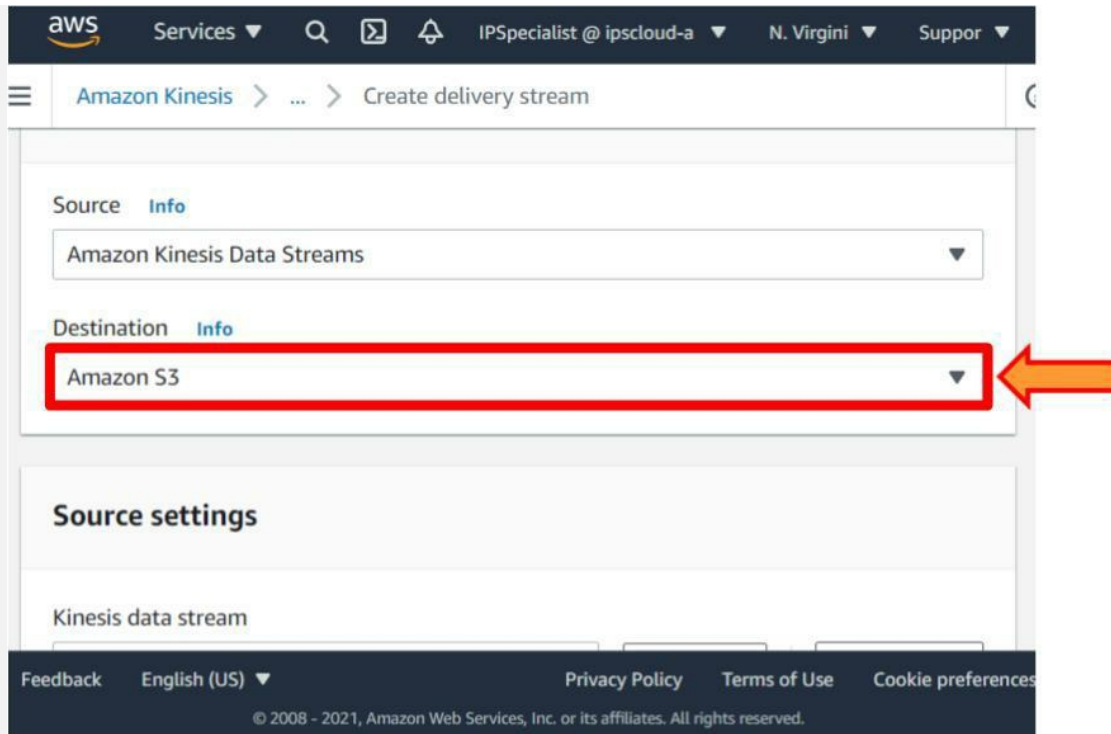
5. Click on the **Create delivery stream** button.



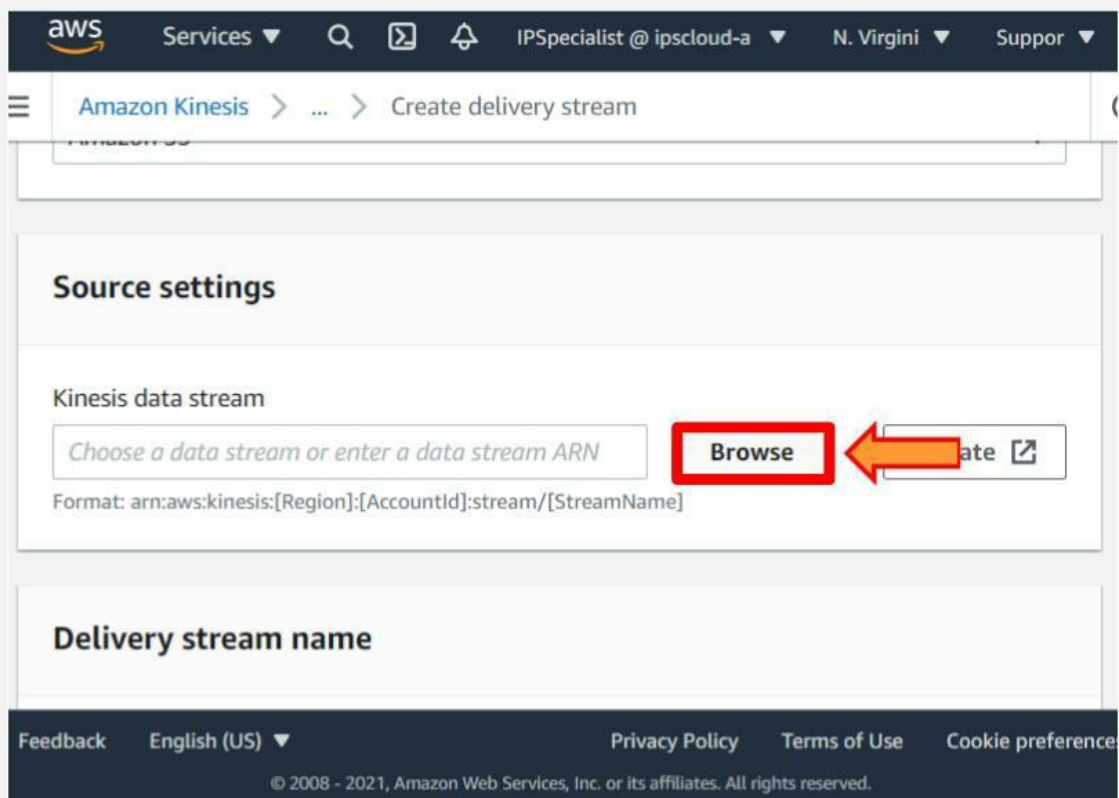
6. Select the source **Amazon Kinesis Data Stream**.



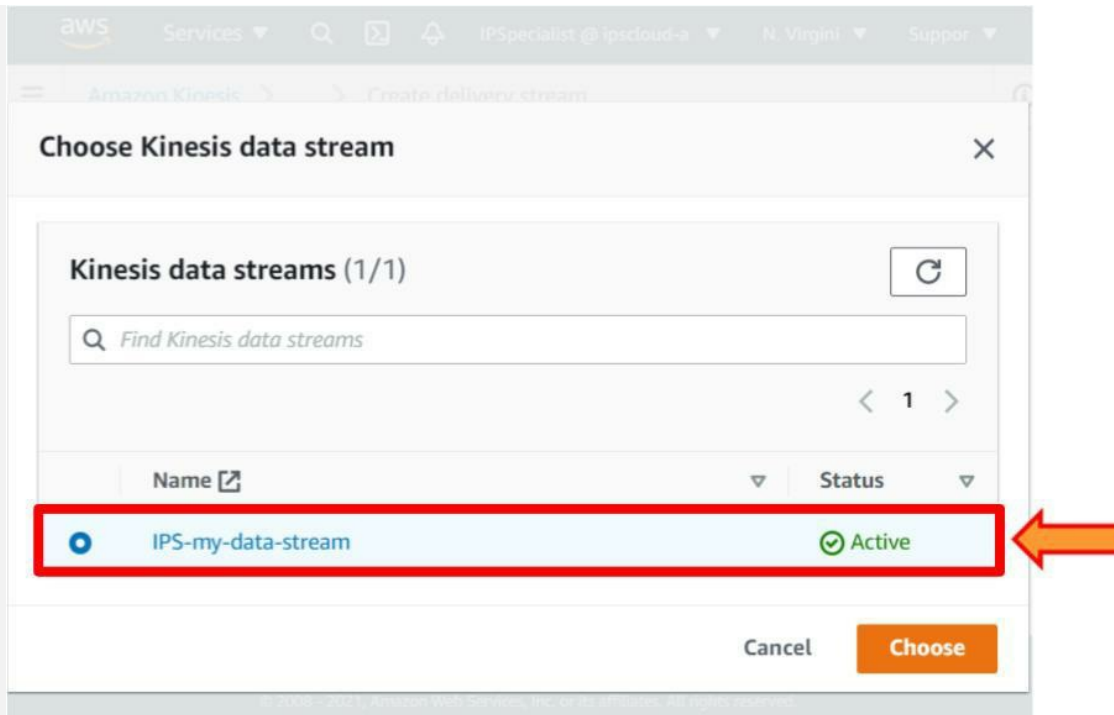
7. Select the Destination **Amazon S3**.



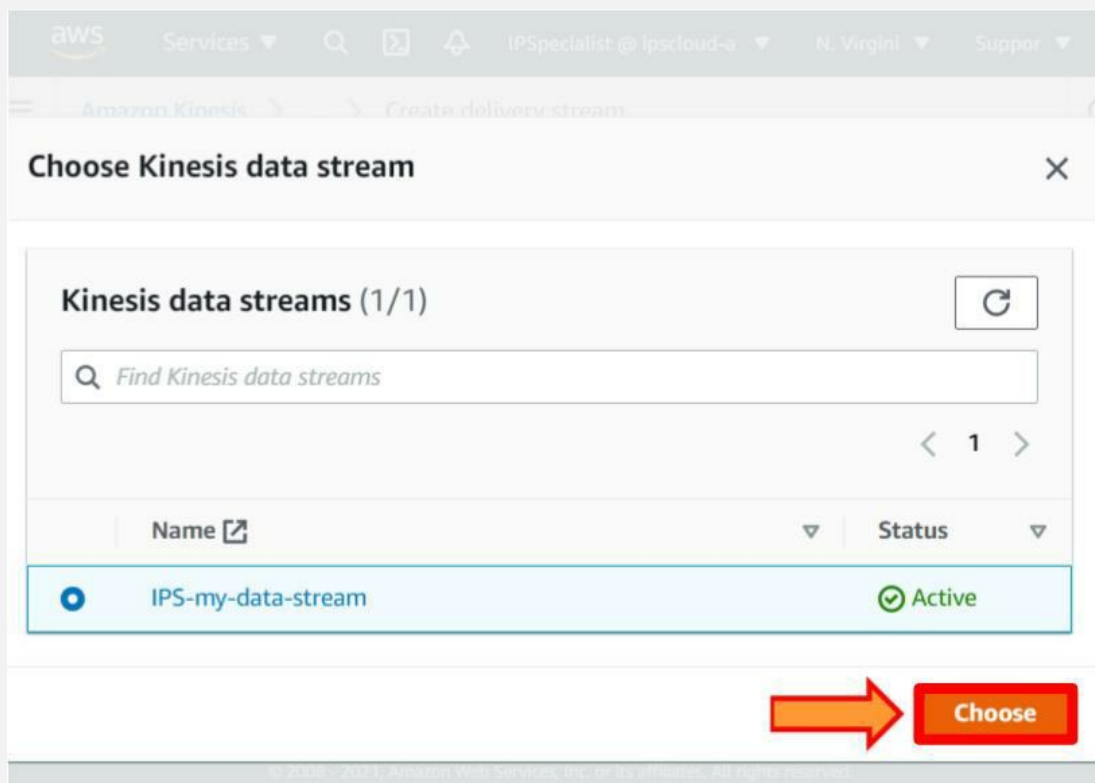
8. Click on the **Browse** button to select delivery stream.



9. Select the **ips-my-data-stream**.



10. Click on the **Choose** button.



11. Give a delivery stream name **ips-my-delivery- stream**.

aws Services Search IPSpecialist @ ipscloud-a N. Virgini Support

Amazon Kinesis > ... > Create delivery stream

arn:aws:kinesis:us-east-1:644738277497:stream/IP... Browse Create

Format: arn:aws:kinesis:[Region]:[AccountId]:stream/[StreamName]

**Delivery stream name**

Delivery stream name

ips-my-delivery-stream

Acceptable characters are uppercase and lowercase letters, numbers, underscores, hyphens, and periods.

**Transform and convert records - optional**

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

12. Select the **Disabled** Data Transformation.

aws Services Search IPSpecialist @ ipscloud-a N. Virgini Support

Amazon Kinesis > ... > Create delivery stream

Configure Kinesis Data Firehose to transform and convert your record data.

**Transform source records with AWS Lambda** Info

Kinesis Data Firehose can invoke an AWS Lambda function to transform, filter, un-compress, convert and process your source data records. The specified AWS Lambda function can also be used to provide dynamic partitioning keys for the incoming source data before its delivery to the specified destination.

**Data transformation**

☒ Disabled

☐ Enabled

**Convert record format** Info

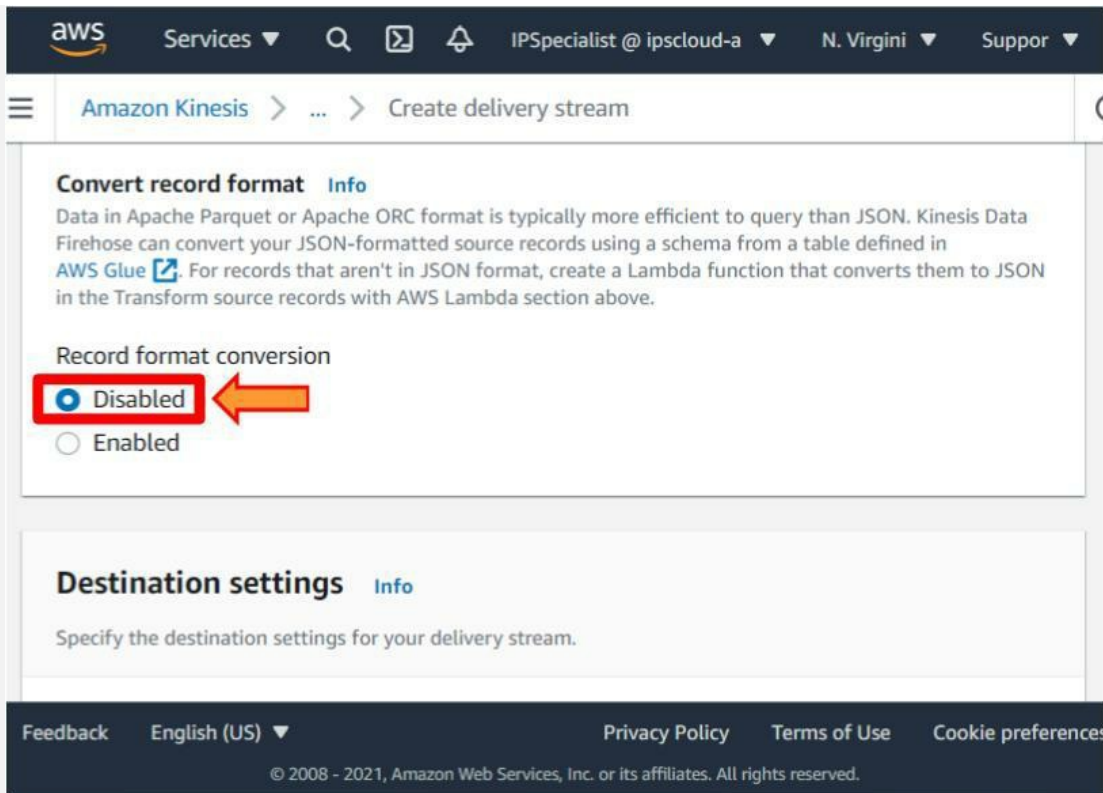
Data in Apache Parquet or Apache ORC format is typically more efficient to query than JSON. Kinesis Data Firehose can convert your JSON-formatted source records using a schema from a table defined in AWS Glue. For records that aren't in JSON format, create a Lambda function that converts them to JSON in the Transform source records with AWS Lambda section above.

**Record format conversion**

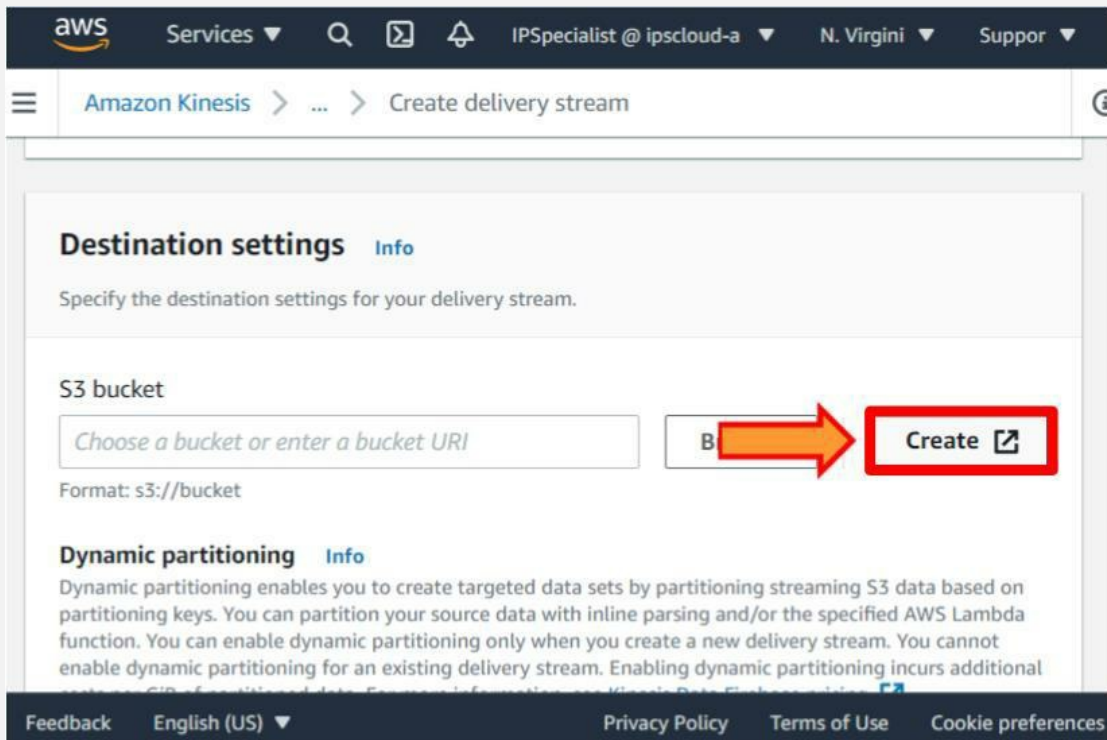
Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

13. Select the **Disabled** Record Format Conversion.



14. Click on the **Create** button to create an S<sub>3</sub> bucket.



15. Give an S<sub>3</sub> bucket name **ips-my-user-data-output-bucket**.

aws Services Search Icons IPSpecialist @ ipscloud-aws Global Support

Amazon S3 > Create bucket

## Create bucket [Info](#)

Buckets are containers for data stored in S3. [Learn more](#)

### General configuration

Bucket name

ips-my-user-data-output-bucket

Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

16. Click on the **Create bucket**.

aws Services Search Icons IPSpecialist @ ipscloud-aws Global Support

Server-side encryption

☒ Disable

☐ Enable

### ► Advanced settings

**i** After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.

**Create bucket**

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

17. Click on the **browse** button.

aws Services ▾ 🔍 ⓘ 🔔 IPSpecialist @ ipscloud-aws ▾ N. Virginia ▾ Support

### Destination settings [Info](#)

Specify the destination settings for your delivery stream.

**S3 bucket**

**Browse**

Format: S3://bucket

**Dynamic partitioning [Info](#)**

With dynamic partitioning, you create targeted data sets from the streaming S3 data by partitioning the data based on partitioning keys. You can partition your source data with inline parsing and/or the AWS Lambda function specified in the "Transform source records with AWS Lambda". Enabling dynamic partitioning incurs additional costs per GiB of partitioned data. For more information, see [Kinesis Data Firehose pricing](#).

☒ Disabled  
☐ Enabled

[Feedback](#) English (US) ▾ [Privacy Policy](#) [Terms of Use](#) [Cookie preferences](#)

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

18. Select the **ips-my-user-data-output-bucket**.

Choose a bucket in Amazon S3 ✕

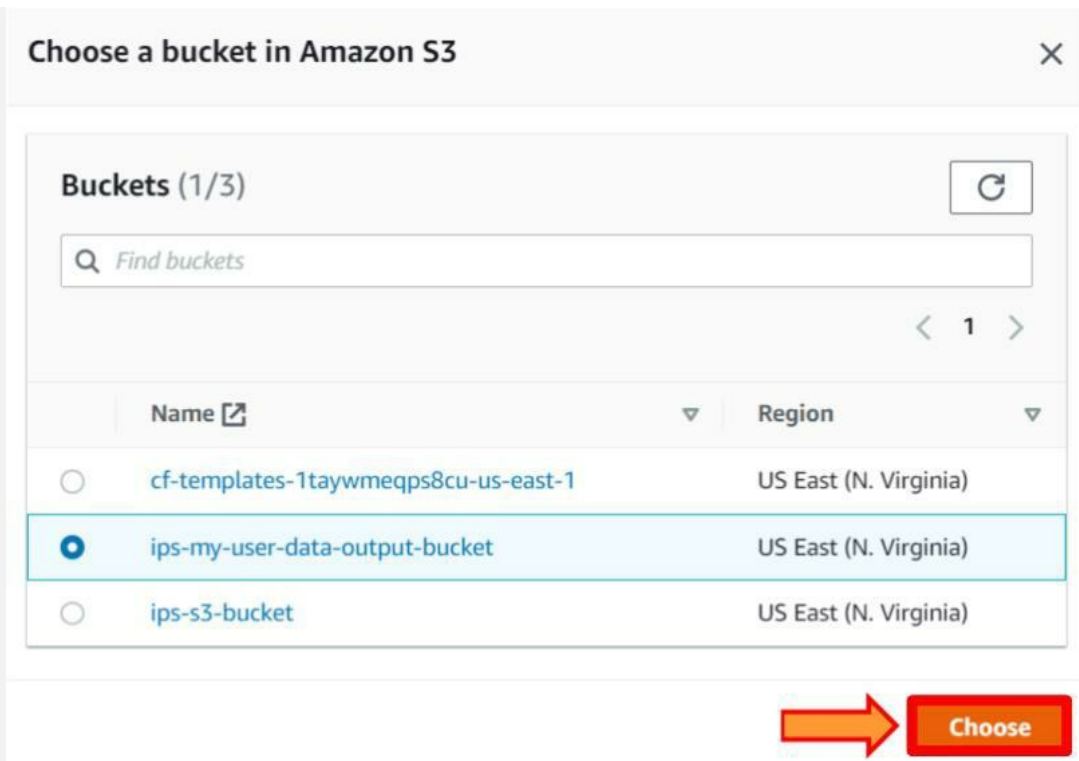
**Buckets (1/3)**

< 1 >

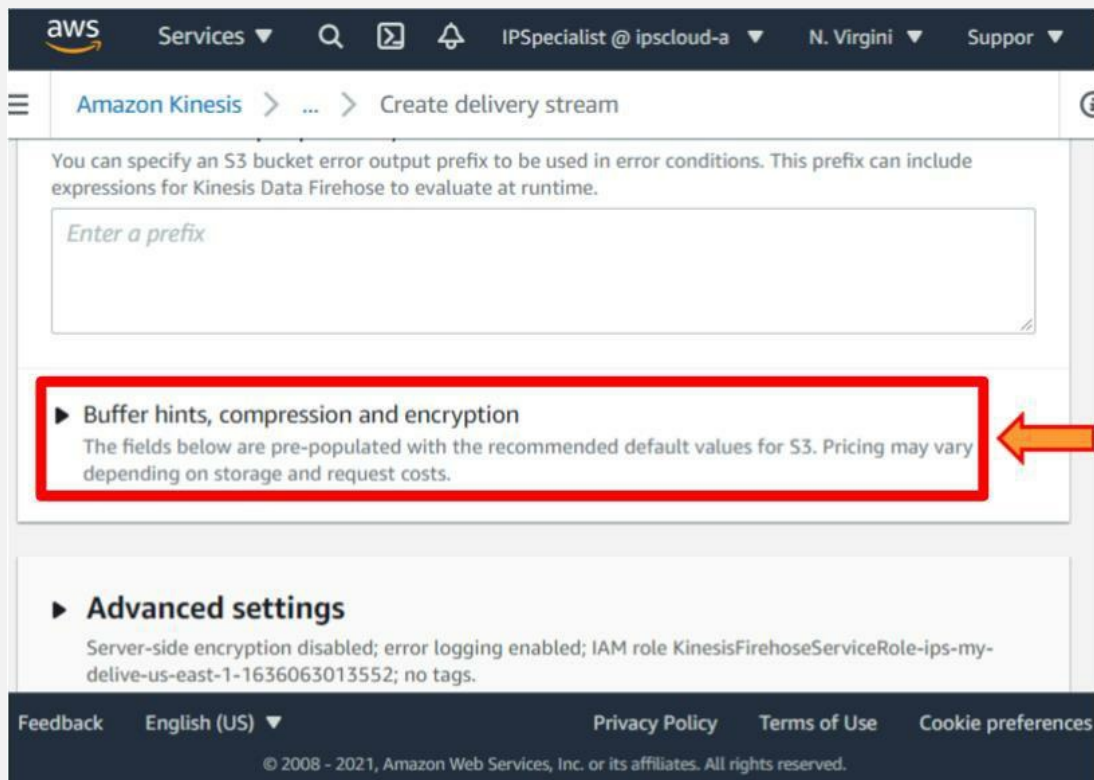
|                                  | Name                                 | Region                |
|----------------------------------|--------------------------------------|-----------------------|
| <input type="radio"/>            | cf-templates-1taywmeqps8cu-us-east-1 | US East (N. Virginia) |
| <input checked="" type="radio"/> | ips-my-user-data-output-bucket       | US East (N. Virginia) |
| <input type="radio"/>            | ips-s3-bucket                        | US East (N. Virginia) |

[Cancel](#) [Choose](#)

19. Click on the **Choose** button.



20. Click on the **Buffer hints, compression and encryption**.



21. Set the Buffer size to **1 MB**.

22. Set the Buffer interval to 60 sec.

aws Services 🔻 🔍 ⓘ 🔔 IPSpecialist @ ipsccloud-a 🔻 N. Virgini 🔻 Support 🔻

Amazon Kinesis > ... > Create delivery stream

Kinesis Data Firehose buffers incoming records before delivering them to your S3 bucket. Record delivery is triggered once the value of either of the specified buffering hints is reached.

**Buffer size**  
The higher buffer size may be lower in cost with higher latency. The lower buffer size will be faster in delivery with higher cost and less latency.

1 MiB  
Minimum: 1 MiB, maximum: 128 MiB. Recommended: 5 MiB.

**Buffer interval**  
The higher interval allows more time to collect data and the size of data may be bigger. The lower interval sends the data more frequently and may be more advantageous when looking at shorter cycles of data activity.

60 seconds  
Minimum: 60 seconds, maximum: 900 seconds. Recommended: 300 seconds.

Feedback English (US) 🔻 Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

23. Click on the **Create delivery stream** button.

aws Services 🔻 🔍 ⓘ 🔔 IPSpecialist @ ipsccloud-a 🔻 N. Virgini 🔻 Support 🔻

Amazon Kinesis > ... > Create delivery stream

☐ Hadoop-Compatible Snappy

**Encryption for data records**  
Compressed record gets encrypted in the S3 bucket using a KMS master key.

☒ Disabled  
☐ Enabled

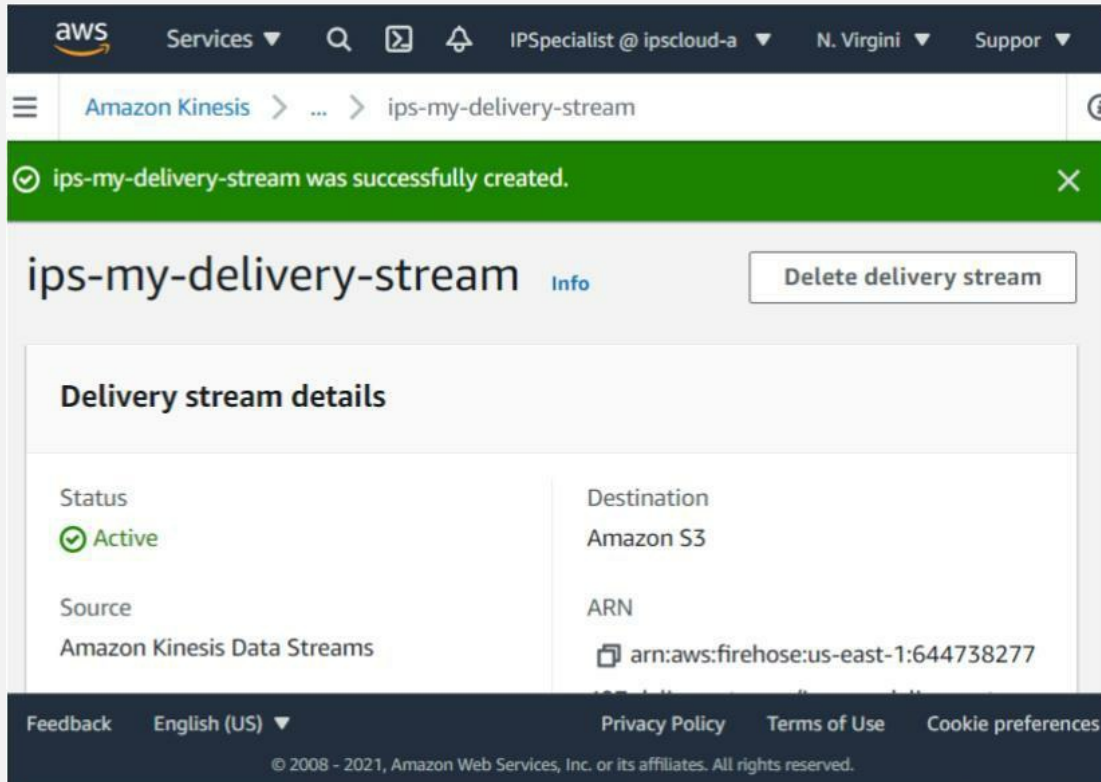
► **Advanced settings**  
Server-side encryption disabled; error logging enabled; IAM role KinesisFirehoseServiceRole-ips-my-delive-us-east-1-1636063013552; no tags.

Create delivery stream

Feedback English (US) 🔻 Privacy Policy Terms of Use Cookie preferences

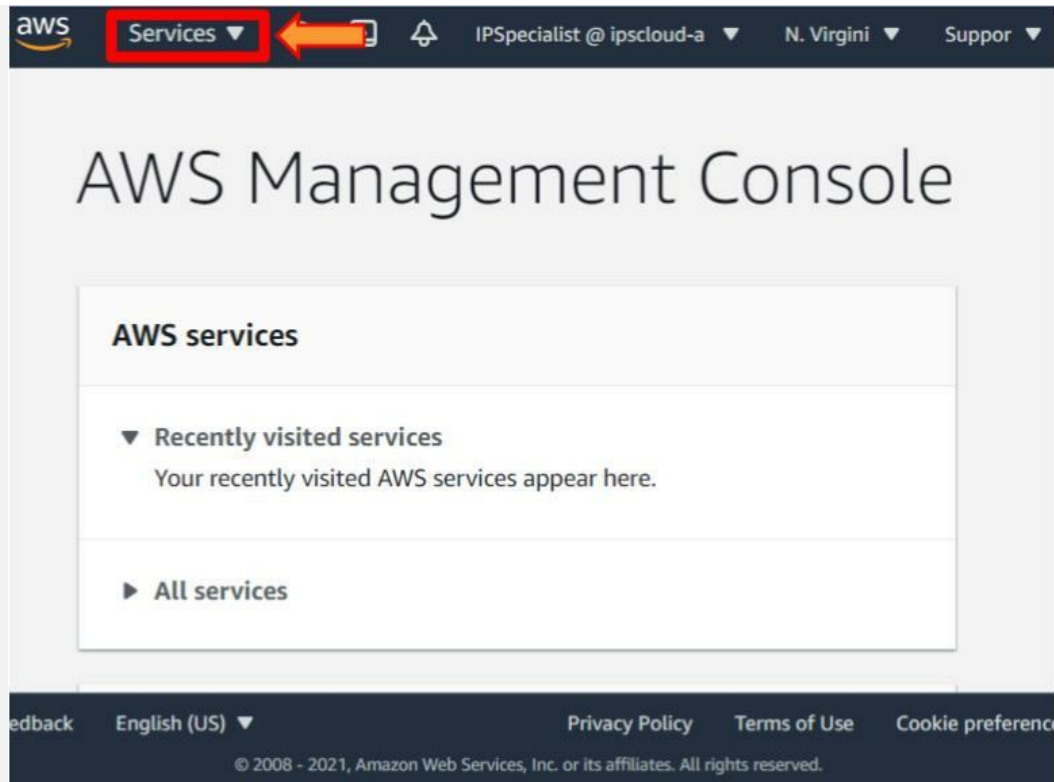
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

24. Hence, the Kinesis Data Firehose is created.

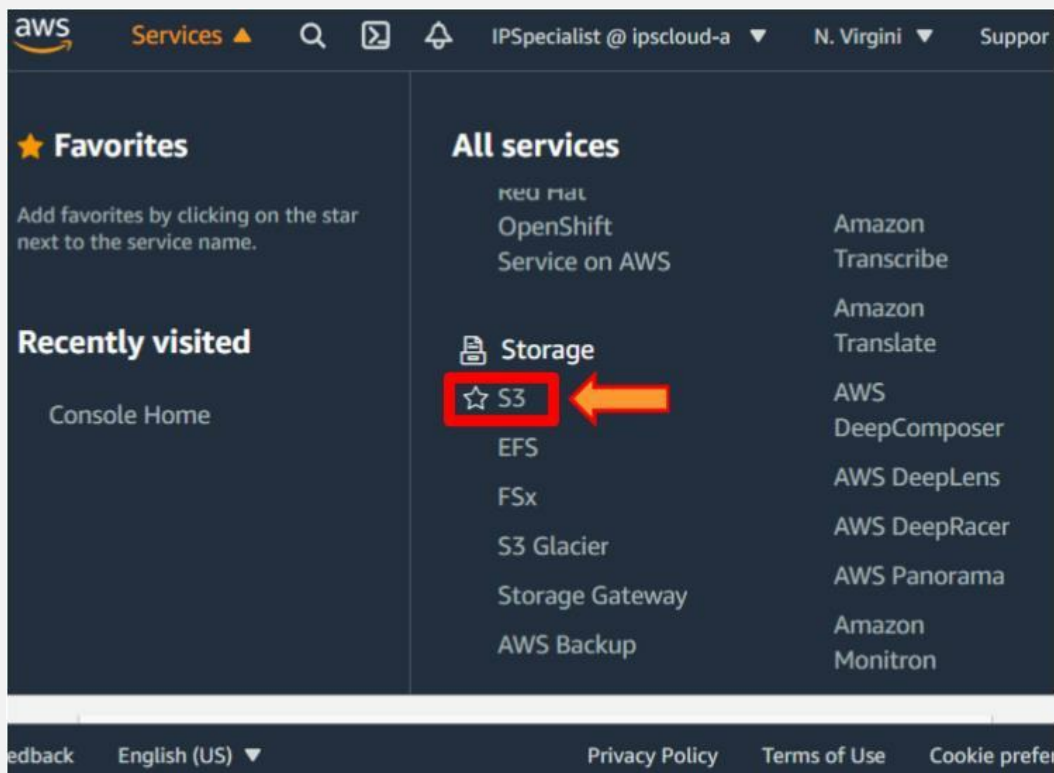


## Step 2: Viewing Collected Data

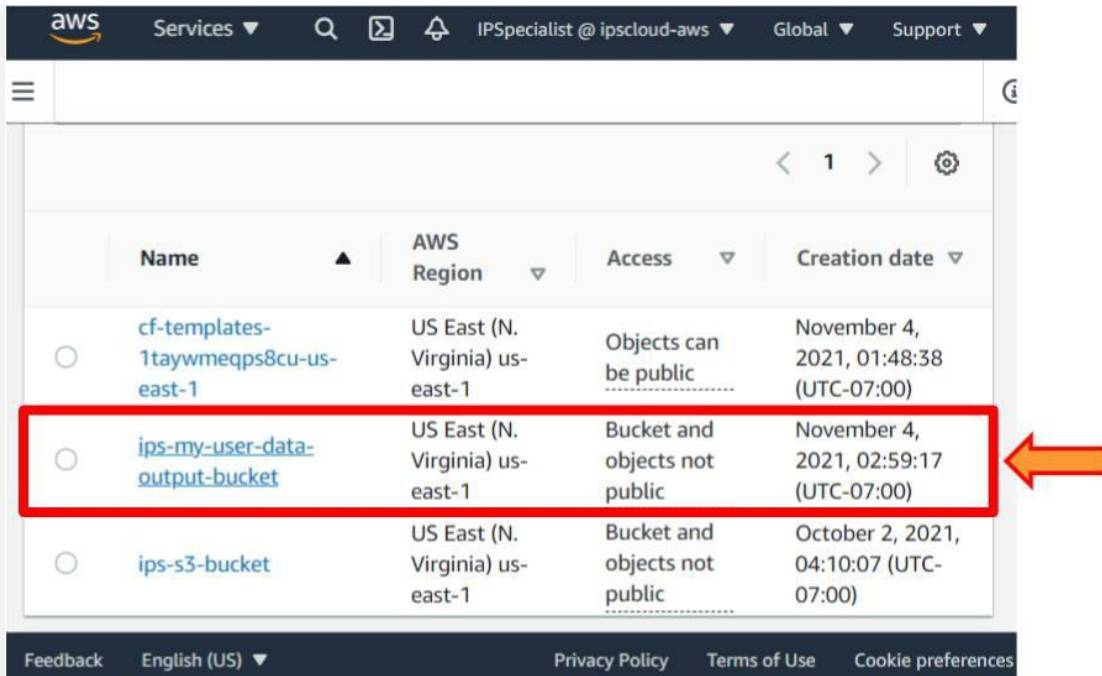
1. Click on the **Services**.



2. Select the **S3** from the **Storage**.

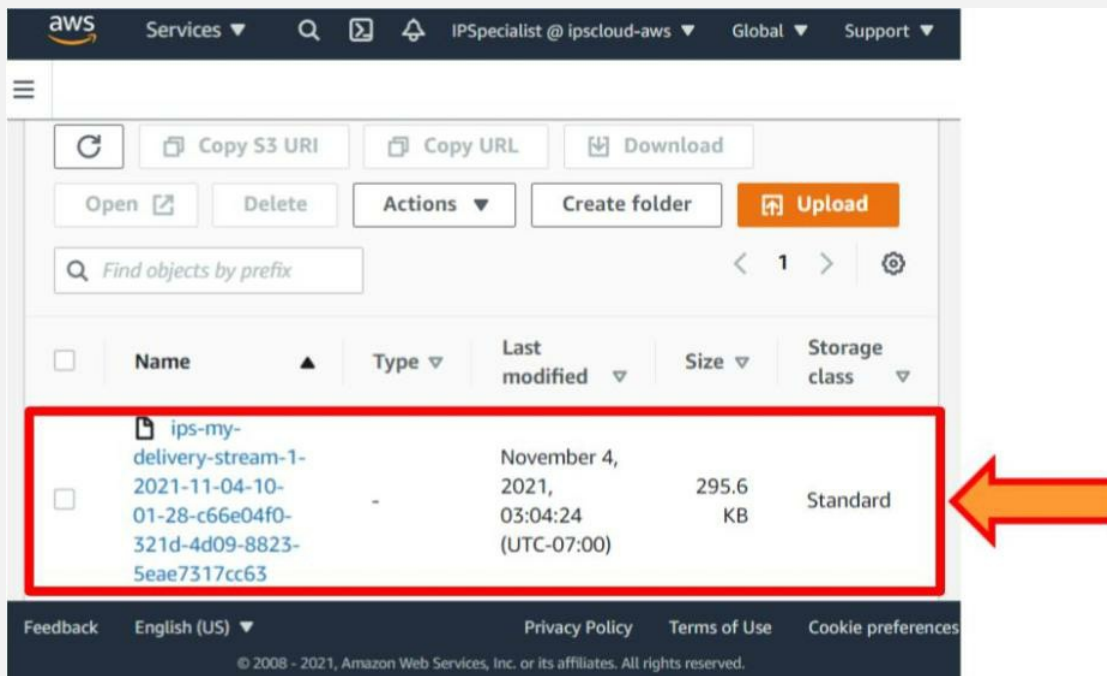


3. Click on the **ips-my-user-data-output-bucket**.



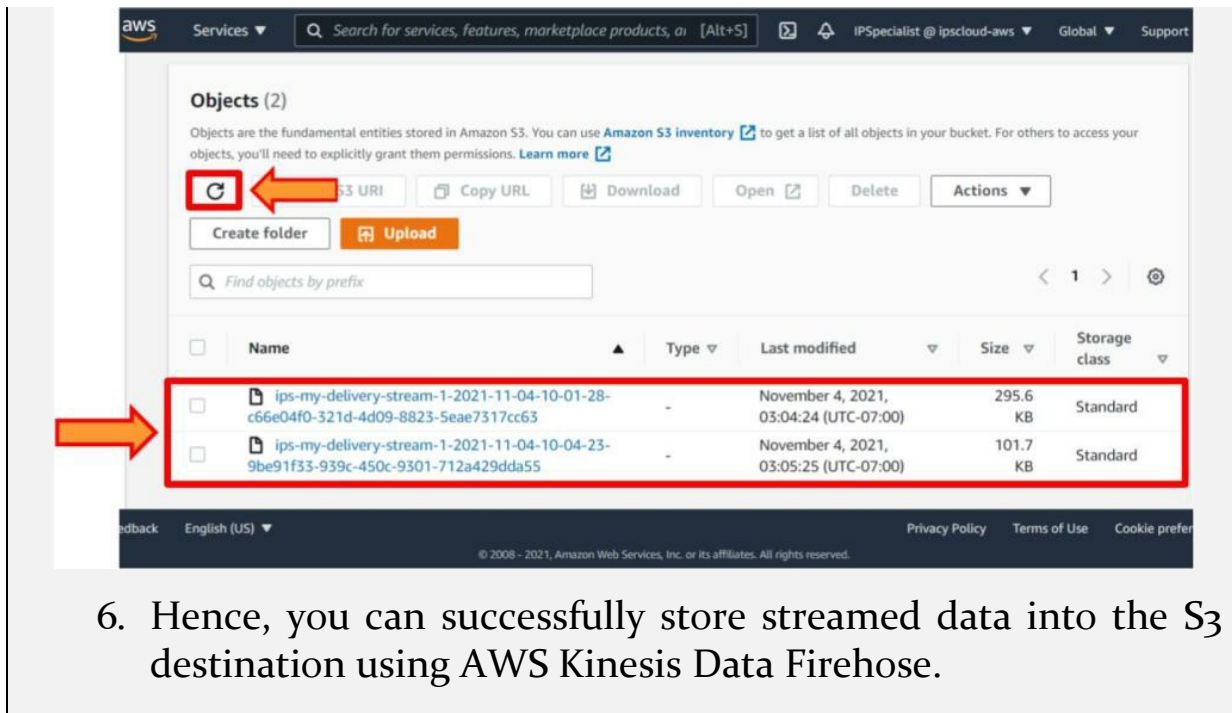
| Name   | AWS Region                      | Access                        | Creation date                          |
|--|---------------------------------|-------------------------------|--|
| <a href="#">cf-templates-1taywmeqps8cu-us-east-1</a> | US East (N. Virginia) us-east-1 | Objects can be public         | November 4, 2021, 01:48:38 (UTC-07:00) |
| <a href="#">ips-my-user-data-output-bucket</a>       | US East (N. Virginia) us-east-1 | Bucket and objects not public | November 4, 2021, 02:59:17 (UTC-07:00) |
| <a href="#">ips-s3-bucket</a>                        | US East (N. Virginia) us-east-1 | Bucket and objects not public | October 2, 2021, 04:10:07 (UTC-07:00)  |

- Click on the folder. Finally, you can see a file that is uploaded onto the S3 bucket.



| Name  | Type | Last modified                          | Size     | Storage class |
|---|------|--|----------|---------------|
| <a href="#">ips-my-delivery-stream-1-2021-11-04-10-01-28-c66e04f0-321d-4d09-8823-5eae7317cc63</a> | -    | November 4, 2021, 03:04:24 (UTC-07:00) | 295.6 KB | Standard      |

- Click on the **Refresh** button, and you will see more data is uploaded.



6. Hence, you can successfully store streamed data into the S3 destination using AWS Kinesis Data Firehose.

## Kinesis Video Streams

### Introduction

Amazon Kinesis Video Streams is a fully managed Amazon Web Services (AWS) solution for streaming live videos from devices to the AWS Cloud. Develop applications for real-time video processing and batch-oriented video analytics.

Kinesis Video Streams offers more than simply video data storage. You may use it to see your video feeds as they arrive in the cloud in real-time. You may either watch your live streams via the AWS Management Console or create your monitoring application that displays live video using the Kinesis Video Streams API library.

Kinesis Video Streams can gather enormous volumes of live video data from various sources, including cellphones, security cameras, webcams,

and cameras embedded in automobiles or drones. Audio, thermal imaging, depth data, RADAR data, and other non-video time-serialized data can also be transmitted. You can develop real-time applications that access the data frame-by-frame for low-latency processing. Kinesis Video Streams are source-independent. You may use the GStreamer library to stream video from a computer's webcam or RTSP to stream video from a camera on your network.

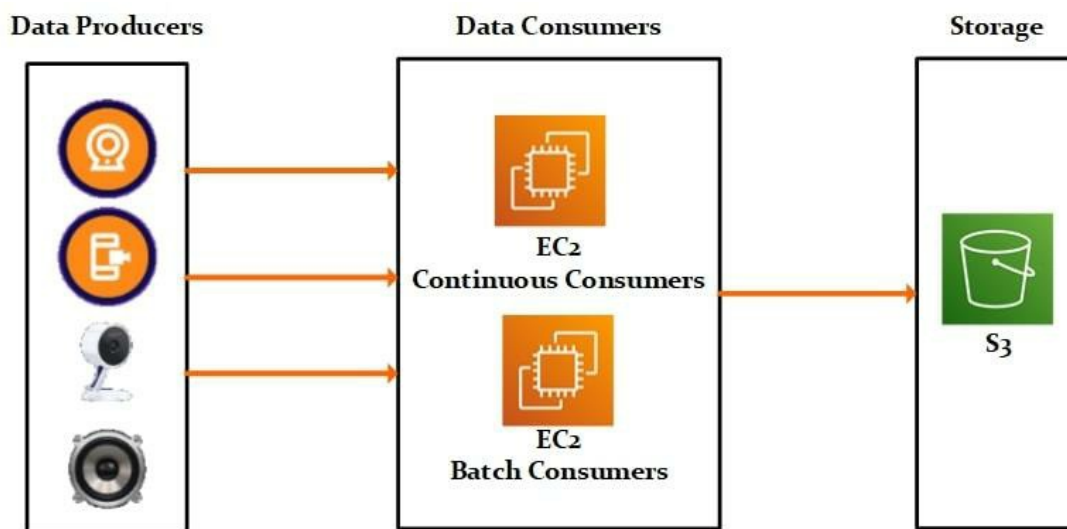
You may also set your Kinesis video stream to keep media data indefinitely for the duration of the retention period. Kinesis Video Streams automatically save and encrypt this data at rest. Kinesis Video Streams also time-index stored data based on both producer and ingestion time stamps. You may design applications that regularly batch-process video data or create applications that require ad hoc access to previous data for various use cases.

Amazon EC2 instances can execute your applications, whether they are real-time or batch-oriented. These applications may utilize open-source deep-learning algorithms to handle data or use third-party applications that interface with Kinesis Video Streams.

Kinesis video streams is a service that allows us to stream videos into the AWS cloud. You can build real-time video processing applications. Suppose you wanted to use videos within the machine learning process. In that case, you could use Kinesis Video Streams to stream those live feeds or stream those videos into AWS. You will also see how you can use Kinesis Video Streams to stream images in real-time or stream audio files in real-time. The way that Kinesis Video Streams works are that you have your data producers, which can be webcams or security cameras. These can also be things like non-video data, such as audio

feeds or images and radar data. These data producers send the data into AWS, where you have different types of continuous or batch consumers. These are the applications that are going to consume and process the video streaming data.

These users are commonly referred to as Kinesis video streams applications. You can write applications that consume and process the video streams in real-time or after that data has been stored onto S3. You could write the applications to process it once it has been held, and you know with confidence that it is durably backed up. You can write consumer applications on Amazon EC2 instances; these consumers get the data in fragments and frames from Kinesis Video Streams to view the process and analyze it. Once you process or consume that data, you can always store it off into S3, or you can store it off into S3. You can then consume that data, process it, and analyze it however you see fit, for instance, for machine learning models.



*Figure 4-23: Kinesis Video Stream*

**EXAM TIP:** Amazon Kinesis Video Streams is a fully managed

Amazon Web Services (AWS) solution for streaming live video from devices to the AWS Cloud. You can create programs for batch-oriented video analytics and real-time video processing.

## **Producer & Consumer Applications**

CCTV monitoring streams video into the cloud using an Amazon cloud cam, which is an opt-in service that you can subscribe to. This is an example where videos are being streamed every day. You have a camera that streams through your wifi and collects any videos you want, for either surveillance or in front of your own home. This camera has some software installed onto it that pushes Video Streams into the cloud. Since Amazon builds it, it is all captured by Amazon, and then it has to use Kinesis Video Streams.

Hence, you can imagine those are stored off into S3. It also sends different notifications if it recognizes certain motions or certain people, or even when dogs are barking. It means that it is using some of the machine learning and artificial intelligence services that AWS offers. It is also using some of the notification systems that AWS offers. Thus, this is just a simple example of Video Streams. You can imagine that this is being captured and analyzed through Amazon Kinesis Video Streams.

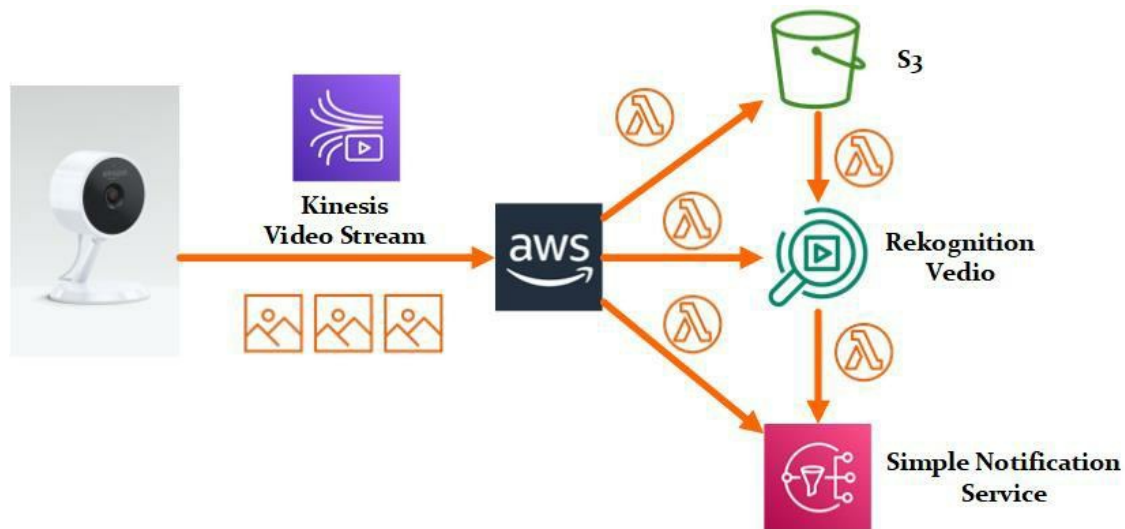
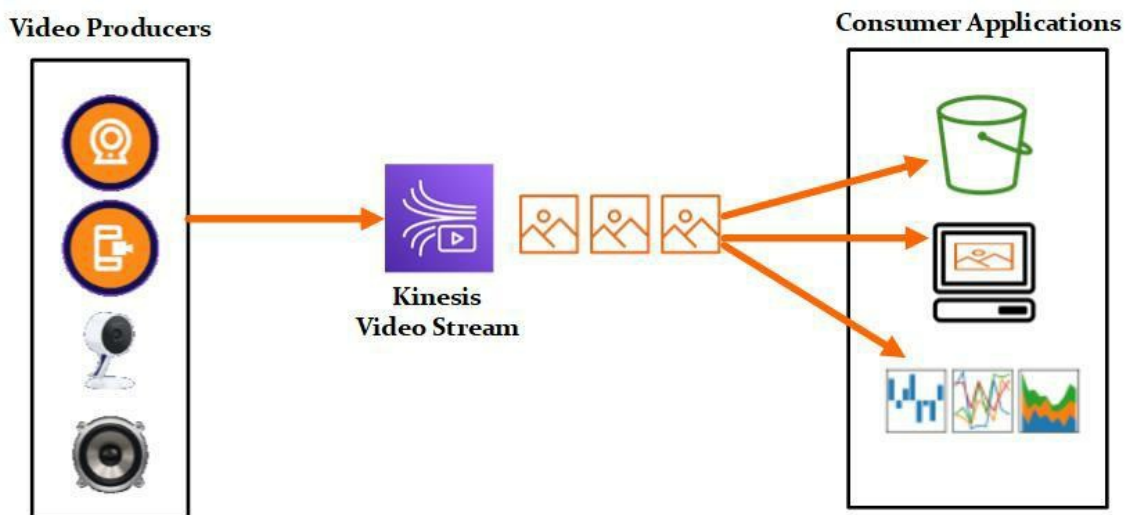


Figure 4-24: Data Coming from CCTV Camera through Kinesis Video Stream

A security camera, a camera from a cell phone or even radar, uses the Kinesis Video Streams libraries that are installed onto the devices for securely connecting to the Kinesis Video Streams application. These libraries also give us techniques for controlling media sources, receiving data from sources, and managing the stream lifecycle as data flows from these devices into Kinesis Video Streams. These video producers or media sources can be from anything, such as camera media or microphone media. They can even be things like infrared cameras, which could be radar or depth cameras, and even things like data logs from text files or application logs. Hence, you could use Kinesis Video Streams to capture that type of information as well. AWS offers us different producer clients and producer libraries to import or ship that data off into Kinesis Video Streams. There are a few different ways that media can be streamed into the Kinesis video stream service. They can be streamed in real-time or have a buffer for a few seconds or after the media uploads.

The software installed onto these devices can extract the video and the

data in frames and then uploads those to the Kinesis video stream service. You can use different consumers to get the data after you have those frames into the Kinesis video stream service. The fragments of the frames you can view, process, or analyze. You can use applications to analyze this information. You can stream it right back and view it in the console or store it all for later use. Hence, with these consumer applications, you can consume and process the data in real-time or do it after it has been stored off into S3.



*Figure 4-25: Producer & Consumer Applications*

**EXAM TIP:** Obtains data from a Kinesis video stream, such as fragments and frames, for viewing, manipulating, or analyzing. When low-latency processing is not necessary, you may build applications that ingest and analyze data from Kinesis video streams in real-time. These users are also known as Kinesis video streams applications.

## Real-time vs. Batch-oriented

There are many different services that you can use to analyze our data, either in real-time or in a batch-oriented process. You can use Amazon EC2 instances; you can hook it into Amazon Rekognition that allows us to connect in machine learning services for our video stream. You can also hook it into several other AWS and even connect it to other third-party services. You can use any of these services that you see below in the figure to analyze, process, and consume our video applications.



*Figure 4-26: Real-time VS Batch-oriented*

**EXAM TIP:** Kinesis Video Streams may be used to build custom real-time applications. That deals with live data streams and batch or ad hoc applications that work with durably stored data. There are no strict latency limits. To design, deploy, and maintain bespoke applications to process and analyze data streams. You can utilize open-source (Apache MXNet, OpenCV), homemade, or third-party AWS Marketplace solutions.

---

## **Kinesis Video Stream Benefits**

The following are some of the advantages of using Kinesis Video Streams:

- 1. Connect & Stream from Millions of Devices:**

Kinesis video streams link and stream video, audio, and other data from millions of devices, such as cellphones, drones, and dash cams. Using the Kinesis Video Streams producer libraries, you may set up your devices to broadcast in real-time or as after-the-fact media uploads.

- 2. Durably Store, Encrypt, & Index Data:**

You may set your Kinesis video stream to save media data indefinitely for specified retention periods. Kinesis Video Streams additionally create an index over the recorded data based on timestamps supplied by the service producer. Using the time index in your applications, you may obtain specified data in a stream.

- 3. Focus on Managing Applications Instead of Infrastructure:**

Because Kinesis Video Streams is server-less, no infrastructure is required to set up or administer. You do not have to worry about the underlying infrastructure's deployment, setup, or elastic scalability as your data streams and the number of consuming applications increase and decline. Kinesis Video Streams provide all of the administration and maintenance required to automatically maintain streams, allowing you to concentrate on the applications rather than the infrastructure.

- 4. Build Real-Time & Batch Applications on Data Streams:**

Kinesis Video Streams may be used to construct bespoke real-time

applications. That works on live data streams and batch or ad hoc applications that function on durably persistent data. It does not have tight latency constraints, so you may use open source (Apache MXNet, OpenCV), homemade, or third-party solutions from the AWS Marketplace to create, deploy, and manage bespoke applications to process and analyze data streams. Kinesis Video Streams Get APIs allow you to create several concurrent applications that can handle real-time or batch mode data.

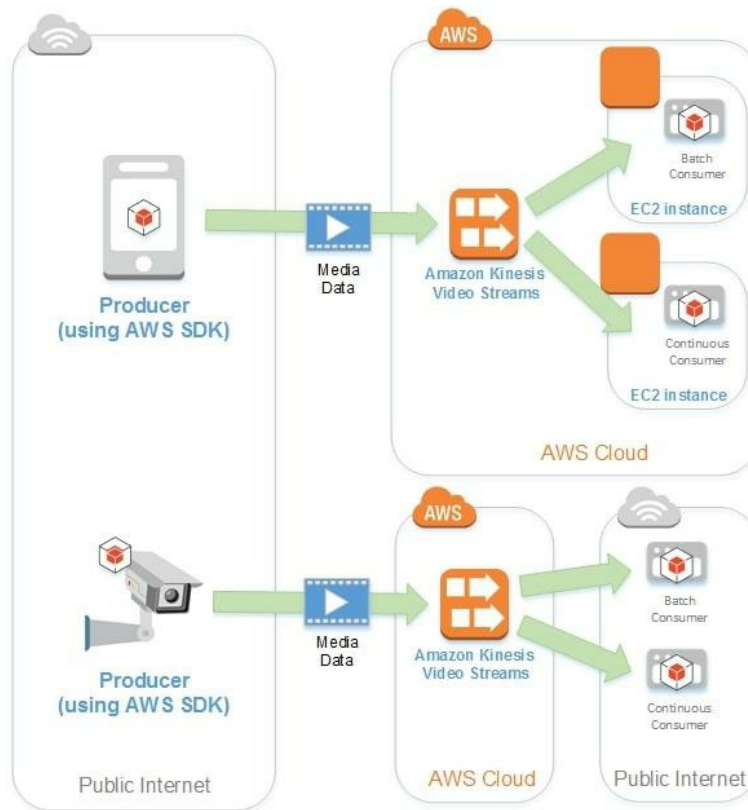
#### **5. Stream Data More Securely:**

Kinesis Video Streams encrypt all data as it passes through the service and as it is saved. Kinesis Video Streams use AWS Key Management Service to encrypt all data at rest and enforce Transport Layer Security (TLS) based on data streaming from devices (AWS KMS). AWS Identity and Access Management may also be used to manage data access (IAM).

### **Kinesis Video Streams Working**

Amazon Kinesis Video Streams is a fully managed AWS service that allows you to broadcast and store live video to the cloud from your devices. After that, you may create your applications for real-time video processing or batch-oriented video analytics.

The figure below depicts the operation of Kinesis Video Streams.



*Figure 4-27: Kinesis Video Streams Working*

The figure above depicts the interplay of the following elements:

- **Producer:**

Any source that generates data for the Kinesis video stream is a piece of video-generating equipment. Such a security camera, body-worn camera, smartphone, or dashboard camera might be considered a producer. Non-video data, such as audio feeds, pictures, or RADAR data, can also be sent by a producer.

- **Kinesis Video Streams Producer Libraries:**

A collection of simple applications and libraries can be installed and configured on your devices; these libraries make it simple to connect to securely. You can consistently stream video in various ways, including in real-time, after a brief buffering period, or as after-the-fact media

uploads.

- **Kinesis Video Stream:**

It is a resource that lets you send live video data, store it if you want to, and then make it available for consumption in real-time, batch, or ad hoc mode. A Kinesis video stream typically has only one producer releasing data into it.

Audio, video, and other time-encoded data streams, such as depth-sensing and RADAR feeds, can be sent via the stream. The AWS Management Console may be used to generate a Kinesis video stream, or the AWS SDKs can be used to build one programmatically.

A Kinesis video stream can be consumed in parallel by many separate applications.

- **Consumer:**

Gets data from a Kinesis video stream, such as fragments and frames, to watch, manipulate, or analyze. These users are commonly referred to as Kinesis video streams applications. When low-latency processing is not required, you may create applications that ingest and analyze data from Kinesis video streams in real-time or after it has been durably stored and time-indexed. These consumer applications may be built to run on Amazon EC2 instances.

- **Kinesis Video Stream Parser Library:**

It allows Kinesis video streaming applications to properly get material from Kinesis video streams with minimal latency. Furthermore, it parses the media frame boundaries, letting programs concentrate on processing and analyzing the frames themselves.

## **AWS Kinesis Video Stream Use Cases**

Following are some uses of Kinesis Video Stream:

### **1. Smart City with Amber Alert Systems:**

It helps capture and stream live video feeds from city cameras and ingest, store, and index video streams from cities cameras. For example, it helps match the license plate of the suspected vehicle and sends alerts to law enforcement officials.

### **2. Equipment Preventive Maintenance**

Another example of streaming thermal images from industrial manufacturing equipment and Amazon will ingest these thermal images. You can set up applications to check the different thermal profiles to predict if overheating is happening. Suppose there is some type of defect or overheating is happening. In that case, you can schedule some preventative maintenance or have someone else come out there and replace that given part. You can kind of predict that ahead of time.

## **Kinesis Data Analytics**

### **Introduction**

You may use standard SQL to handle and analyze streaming data using Amazon Kinesis Data Analytics for SQL Applications. The service lets you quickly develop run sophisticated SQL code against streaming sources to do time-series analytics, feed real-time dashboards, and generate real-time metrics.

To get started with Kinesis Data Analytics, develop a Kinesis data analytics application that constantly reads and analyses streaming data.

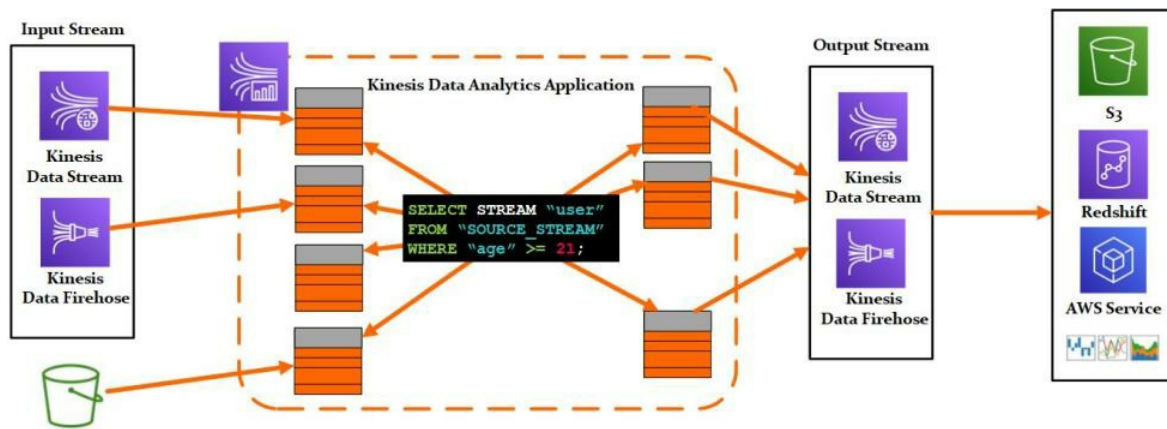
Data may be ingested via Amazon Kinesis Data Streams and Amazon Kinesis Data Firehose streaming sources. Then, using the interactive editor, you can write your SQL code and test it with live streaming data. You may also choose where you want Kinesis Data Analytics to deliver the results.

Amazon Kinesis Data Firehose (Amazon S3, Amazon Redshift, Amazon OpenSearch Service, and Splunk), AWS Lambda, and Amazon Kinesis Data Streams are supported as destinations by Kinesis Data Analytics.

Kinesis Data Analytics is a cool application that allows you to take streaming data and run SQL queries on them to get started using Kinesis Data Analytics. You have to have someplace from where the input stream begins. Based on where the data is coming from, you have two choices: you can either choose Kinesis Data Streams or Kinesis Data Firehose. You can create a Kinesis Data Analytics application that takes the data from the input stream and runs queries on that data. You can also use data that is in S3 to run queries to merge that data. These queries are going to produce successful records, and they might also produce any error records that happen.

The successful records will be a subset of the information you want, and any errors will be captured as well. You can handle these differently if you need to. Once you create one or more in-application streams, this will hold our intermediate results for the actual query you are running. Hence, you could store this information off, or you can keep that within our Kinesis Data Analytics application. Refer to it in external destinations whose examples are things like Kinesis Data Streams and Kinesis Data Firehose. Once the output stream has been forwarded to Kinesis Data Streams or Kinesis Data Firehose, you can store that data

off into S3, Redshift, or any other destinations that Kinesis Data Firehose allows us to store our data. You can use Kinesis Data Streams to forward the data along to AWS Lambda. Since it is in Lambda, you can then write custom applications around it and have business analysis tools analyze that data. Some custom real-time applications view different dashboards, as shown in the figure below:



*Figure 4-28: Kinesis Data Analytics*

**EXAM TIP:** Using Amazon Kinesis Data Analytics for SQL Applications, you may handle and analyze streaming data using conventional SQL. The service enables you to develop swiftly. Execute complex SQL code against streaming inputs to perform time-series analytics, feed real-time dashboards, and produce real-time metrics.

Within the console, Kinesis data analytics is easy to set up. You essentially tell it where the streaming input sources from either Firehose or data streams are. You can write sequel queries and output those results onto S3 or view them in real-time.

The screenshot shows the Amazon Kinesis Real-time analytics console. The left sidebar contains navigation links: Dashboard, Data Streams, Data Firehose, Data Analytics (highlighted), Video Streams, External resources, and What's new. The main content area is titled 'Real-time analytics' and includes buttons for 'Save and run SQL', 'Add SQL from templates', 'Download SQL', 'SQL reference guide', and 'Kinesis data generator tool'. A SQL query is entered in a text area: `SELECT STREAM ROWTIME FROM SOURCE_SQL_STREAM_001`. Below the query, the 'Application status' is 'RUNNING'. The 'Source data' tab is active, showing 'Streaming data' as 'SOURCE\_SQL\_STREAM\_001' and 'Reference data (optional)' as 'Connect reference data'. A table of streaming data is displayed with columns: ROWTIME, type, x, y, url, and APPROXIMATE\_ARRIVAL. The data shows mousemove events with coordinates and timestamps.

| ROWTIME                 | type        | x       | y       | url        | APPROXIMATE_ARRIVAL     |
|-------------------------|-------------|---------|---------|------------|-------------------------|
| TIMESTAMP               | VARCHAR(16) | INTEGER | INTEGER | VARCHAR(8) | TIMESTAMP               |
| 2019-04-01 10:38:45.768 | mousemove   | 806     | 8       | /about     | 2019-04-01 10:38:44.117 |
| 2019-04-01 10:38:45.768 | mousemove   | 806     | 8       | /about     | 2019-04-01 10:38:44.127 |
| 2019-04-01 10:38:45.768 | mousemove   | 790     | 31      | /about     | 2019-04-01 10:38:44.144 |
| 2019-04-01 10:38:45.768 | mousemove   | 747     | 90      | /about     | 2019-04-01 10:38:44.162 |
| 2019-04-01 10:38:45.768 | mousemove   | 733     | 115     | /about     | 2019-04-01 10:38:44.163 |

*Figure 4-29 Kinesis Analytics Console*

## AWS Kinesis Data Analytics Benefits

You can create SQL code that reads, analyzes, and stores data in real-time using Amazon Kinesis Data Analytics. You may build applications that convert and give insights into your data using conventional SQL queries on streaming data. The following are some use-case examples for Kinesis Data Analytics:

### 1. Generate Time-Series Analytics:

Metrics may be calculated over periods and then sent to Amazon S3 or Amazon Redshift through a Kinesis data delivery stream.

### 2. Feed Real-Time Dashboards:

You may feed real-time dashboards with aggregated and processed streaming data findings.

### 3. Create Real-Time Metrics:

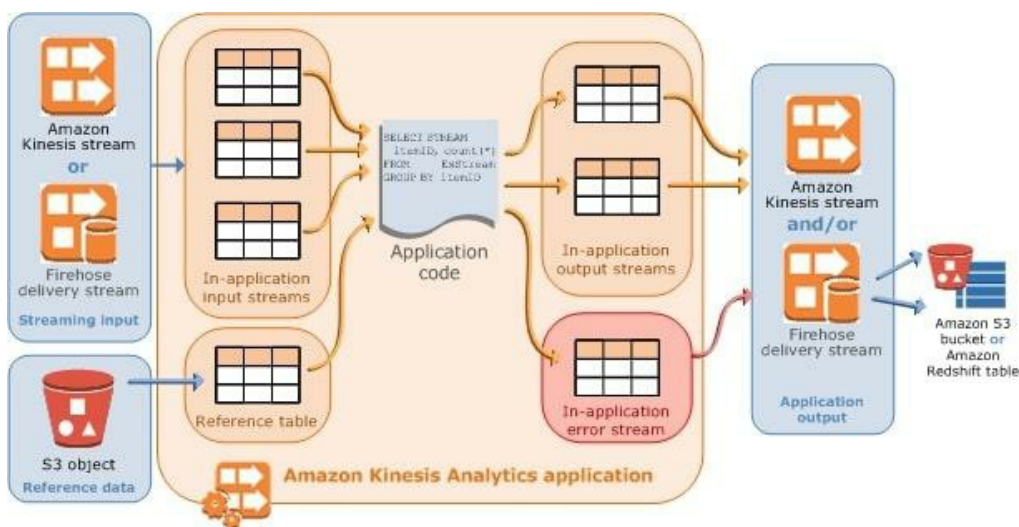
Custom metrics and triggers may be created for usage in real-time monitoring, alerts, and alarms.

**EXAM TIP:** You may input aggregated and processed streaming data discoveries into real-time dashboards.

## Kinesis Data Analytics Working

In Amazon Kinesis Data Analytics, the primary resource that you may build in your account is an application. The AWS Management Console or the Kinesis Data Analytics API may be used to develop and administer applications. Kinesis Data Analytics provides API functions for application management.

Kinesis Data Analytics applications constantly read and process real-time streaming data. To handle the incoming streaming data and provide output, you can create application code in SQL. The output is then written to a location specified by Kinesis Data Analytics. The figure below depicts a standard application architecture:



*Figure 4-30 Kinesis Data Analytics Working*

Each application is given a name, description, version ID, and status. When you initially construct an application, Amazon Kinesis Data Analytics provides it a version ID. When you alter any application configuration, this version ID is updated. Kinesis Data Analytics changes the current application version ID when you add an input configuration. You can add or remove a reference data source, add or delete an output configuration, or update the application code. Kinesis Data Analytics also keeps track of when an application was developed and when it was last updated.

In addition to these fundamental features, each application includes the following:

- **Input:**

Your application is a streaming source. You may choose between a Kinesis data stream and a Kinesis Data Firehose data delivery stream as the streaming source. The streaming source is mapped to an in-application input stream in the input settings. The in-application stream functions similarly to a continually updating table to execute SQL operations such as SELECT and INSERT. Additional in-application streams can be created in your application code to store intermediate query results. You can divide a single streaming source into numerous in-application input streams to enhance performance.

Each Amazon Kinesis Data Analytics application stream contains a timestamp column named Timestamps and the ROWTIME Column. This column can be used in time-based windowed queries.

A reference data source can be configured as an input data stream to supplement your application. As a result, an in-application reference

table is created. Your reference data must be saved as an object in your S3 bucket. Amazon Kinesis Data Analytics reads the Amazon S3 object and produces an in-app table when launching the application.

- **Application Code:**

A set of SQL queries that process input and output results. SQL statements can be written against in-application streams and reference tables. JOIN queries can also be used to integrate data from both of these sources.

Application code can be as simple as a single SQL statement that picks a streaming input and outputs the results. It may also be a sequence of SQL statements, with the output of one feeding into the input of the next. You may also create application code to divide an input stream into several streams. Additional queries can then be used to handle these streams.

- **Output:**

Query results are sent to in-application streams in application code. To retain intermediate results, you can construct one or more in-application streams in your application code. Next, you may set the application output to persist data to external destinations in the in-application streams. That contains your application output (also known as in-application output streams). A Kinesis Data Firehose delivery stream or a Kinesis data stream can be used as an external target. Take note of the following facts regarding these locations:

- A Kinesis Data Firehose delivery stream may be configured to write results to Amazon S3, Amazon Redshift, or Amazon OpenSearch Service (OpenSearch Service).
- Instead of Amazon S3 or Amazon Redshift, you may alternatively publish application output to a specific destination. To accomplish this, you must specify a Kinesis data stream as the destination in your output settings. Then, you tell AWS Lambda to poll

the stream and execute your Lambda function. Stream data is fed into your Lambda function code. You can write the incoming data to your destination in your Lambda function code.

**EXAM TIP:** Kinesis Data Analytics quires read and analyze real-time streaming data continuously. You write SQL application code to handle the incoming streaming data and deliver output.

## AWS Kinesis Data Analytics Use Cases

Following are some uses of the AWS Kinesis Data Analytics and where you should use Kinesis Data Analytics:

### 1. **Responsive Real-Time Analytics:**

Assume you work for an organization. It is your job to find the root cause of any errors or response time issues you are having with your application. You can build responsive and real-time analytic applications. Whenever you use Kinesis Data Analytics coupled with Data Firehose, you can use Data Firehose as the transport layer for particular logging data. You can uncover real-time monitoring metrics, such as response time errors or error rate spikes. Once all of the data goes through this pipeline, you can send it off to Amazon CloudWatch for additional metrics displayed in a standard dashboard across the business. Hence, anyone within the organization or anyone within your team could see these metrics within CloudWatch. You can include metrics like the overall traffic summary, response time errors, total requests, and any API metrics. You can look at response time percentiles, the number of successful requests, or the number of error requests. You can look at things like disk usage, network, and CPU

utilization. Using Kinesis Data Analytics makes it super simple to integrate the other Kinesis services like Data Firehose, Kinesis Data Streams, and Lambda to create things like responsive real-time analytics. Example: Uncover real-time monitoring metrics such as response time and error-rate spikes.

## **2. Managing Real-Time Gaming Metrics:**

Another example assumes you need to manage real-time gaming metrics. Imagine you work for a company that builds iPhone or Android applications for games. You can use Kinesis Data Streams combined with AWS Lambda and Kinesis Data Analytics to build sophisticated pipelines. You can capture important streaming data like the gamer's minimum and the average time they play the game. As a result, if there is a significant increase in user gaming traffic, you may add more shards to your Kinesis Data Stream. Utilize Kinesis Data Analytics to combine all data to guarantee that the final analytics output is right. Hence, you could store it into some data store or build real-time dashboards for the data coming from the stream. Example: Aggregate and join important gaming data like gamers account or min, max, and average time playing the game over a 10-minute sliding window.

# **Demo: Kinesis Data Analytics**

## **Introduction**

### **Kinesis Data Analytics**

Amazon Kinesis Data Analytics is the most straightforward method for converting and analyzing real-time streaming data using Apache Flink.

Apache Flink is an open-source data stream processing framework and engine. Apache Flink applications' creation, maintenance, and integration with other AWS services are made easier using Amazon Kinesis Data Analytics.

Amazon Kinesis Data Analytics handles everything needed to constantly operate streaming applications and grow automatically to meet the amount and throughput of your incoming data. There are no servers to manage with Amazon Kinesis Data Analytics and no minimum charge or setup cost. You pay for the resources your streaming applications utilize.

### **Kinesis Data Streams**

You may use Amazon Kinesis Data Streams to create custom applications that process or analyze streaming data for specific purposes. To an Amazon Kinesis data stream, you may constantly add data from hundreds of thousands of sources, such as clickstreams, application logs, and social media. Within seconds, your Amazon Kinesis Applications will be able to read and analyze data from the stream.

### **Amazon Simple Storage Service S3**

Amazon S3 is a type of object storage that allows you to store and recover any quantity of data from any location. It is a low-cost storage solution with business resilience, reliability, efficiency, privacy, and infinite expansion.

Amazon S3 is a web service that allows you to store and retrieve an infinite quantity of data from any place and at any time. You may quickly create projects that integrate cloud-native storage using this

service. Because Amazon S3 is easily customizable and you only pay for what you use, you can start small and scale up as needed without sacrificing performance or dependability.

Amazon S3 is also built to be highly adaptable. Instead of finding out how to store their data, Amazon S3 allows developers to focus on innovation. Build a simple FTP system or a complex web application like the Amazon.com retail website. Read the same piece of data a million times or only for emergency disaster recovery, and store whatever type and amount of data you desire.

### **AWS CloudFormation**

AWS CloudFormation is a tool that makes it simple for developers and organizations to construct a collection of linked AWS and third-party resources and then provision and manage them logically and reasonably.

Developers may use a simple, declarative approach to deploy and change compute, database, and many other resources, abstracting away the complexities of individual resource APIs. AWS CloudFormation is meant to make resource lifecycle management repeatable, predictable, and safe, with features such as automatic rollbacks, automated state management, and resource management across accounts and regions. Multiple ways to generate resources have recently been added, including leveraging AWS CDK for writing in higher-level languages, importing existing resources, and detecting configuration drift. A new Registry makes it easier to create custom types that inherit some of CloudFormation's core functionalities.

### **Problem**

Assume you work in an organization as a Data Analytics engineer. Your organization has thousands of users interacting with the organization application. The organization gives you a task to capture real-time data about the users for a marketing campaign. For users aged 21 and up, you must collect information such as name, age, gender, and location. So, how can you automate this task?

### **Solution**

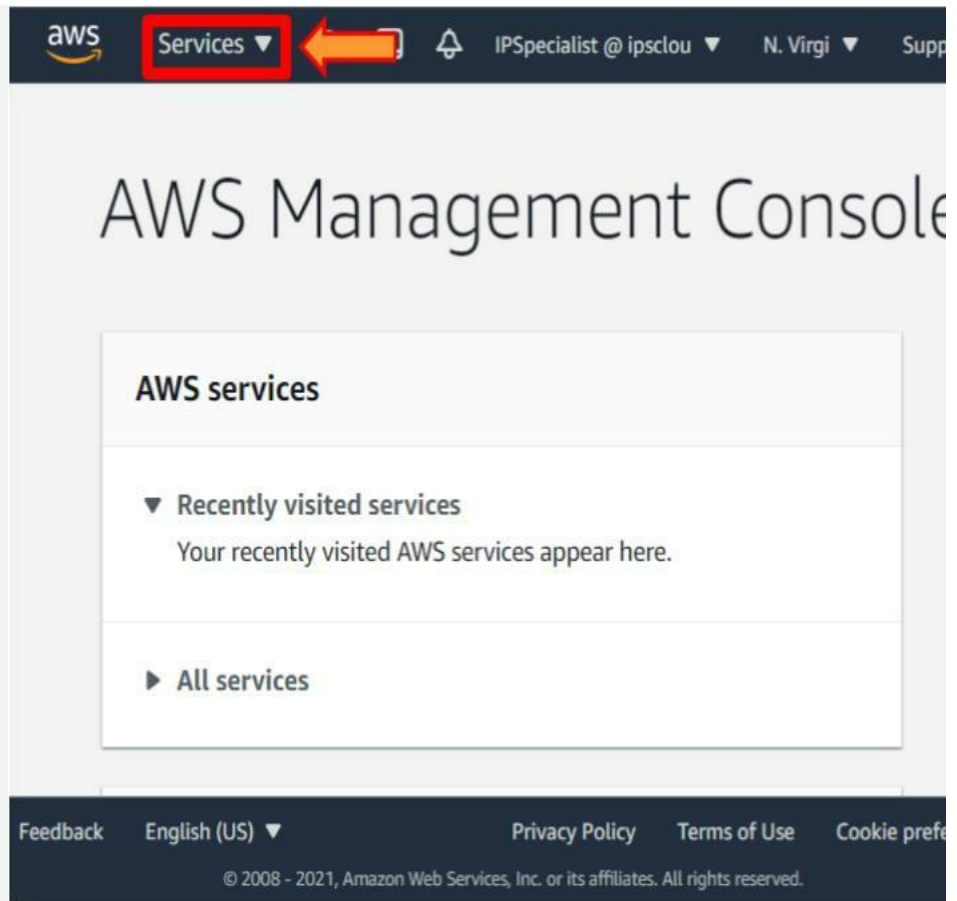
The solution is using AWS services to automate your work. You use Kinesis Data Streams to generate or collect data, and then you use Kinesis Data Analytics to analyze data by running SQL queries and for storing the real-time analyzing data. The other helping service that you can use is AWS CloudFormation to create a stack. An S3 bucket is used to store that transformed data.

**Note:** Before starting this demo, make sure to create AWS Kinesis Data Streaming and AWS CloudFormation stack, as used in this lab and as mentioned in the above demo of Kinesis Data Stream.

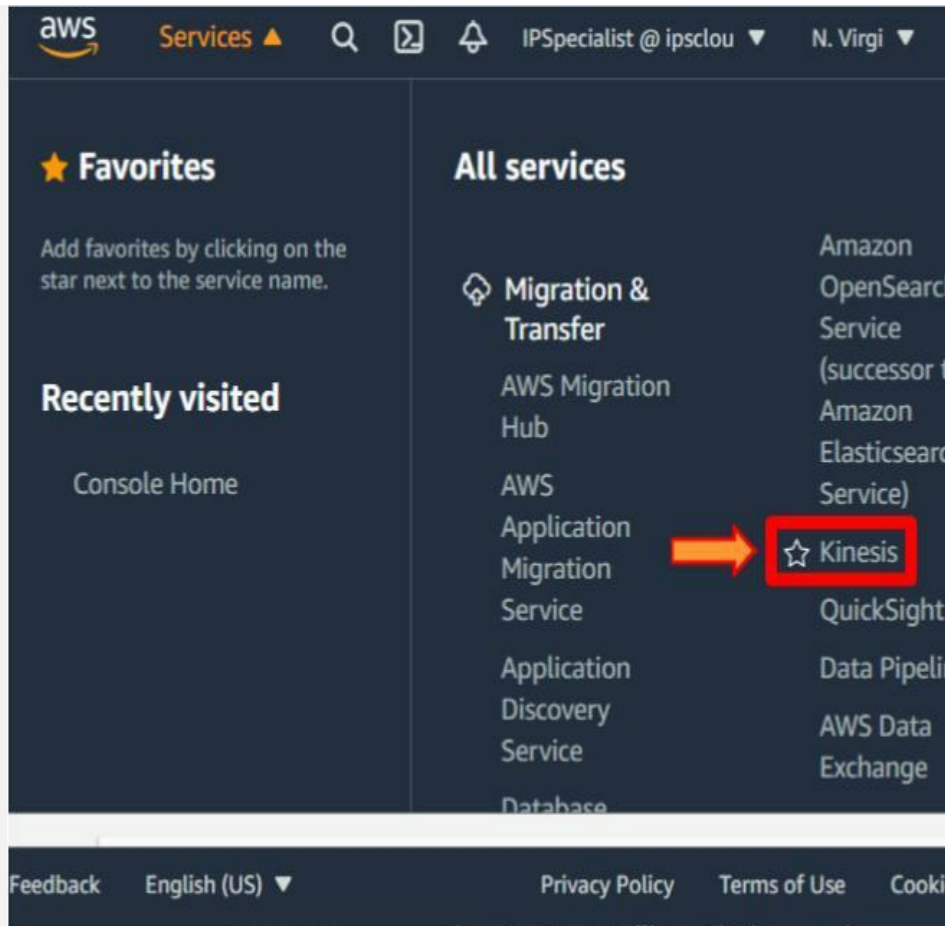
---

### **Step 1: Create Kinesis Data Analytics**

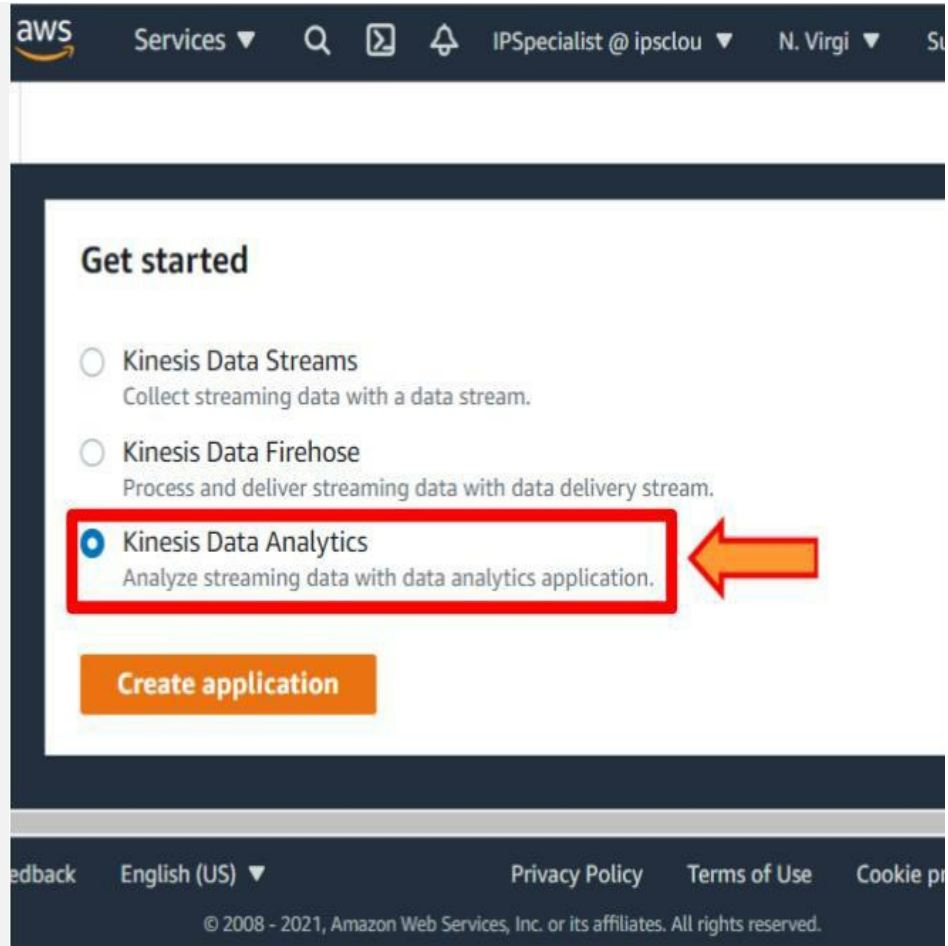
1. Log in to the **AWS Console**.
2. Click on the **Services**.



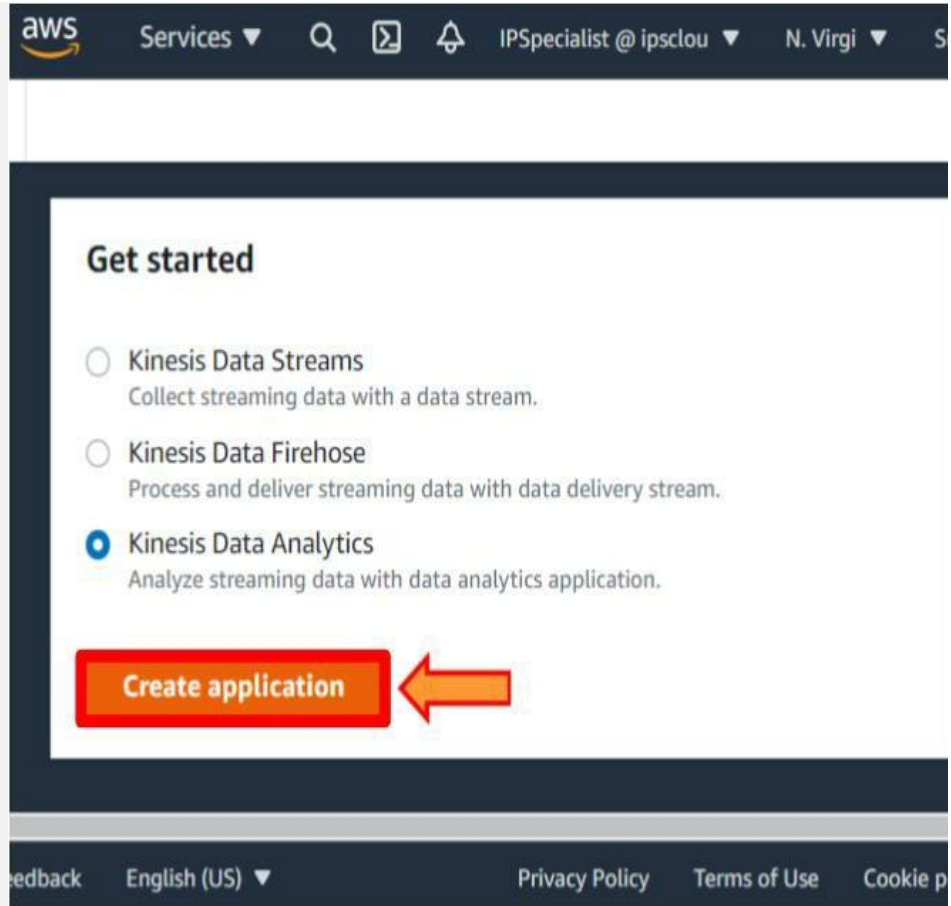
3. Select the **Kinesis** from the **Analytics**.



4. Select the **Kinesis Data Analytics**.



5. Click on the **Create Application** button.





6. Select the **Legacy SQL**.



aws Services ▾ 🔍 ⓘ 🔔 IPSpecialist @ ipsco ▾ N. Virg ▾ Sup

Amazon Kinesis > ... > Create application

Apache Flink is an open-source framework and distributed processing engine for stateful computations over unbounded and bounded data streams. Use this option to build streaming application using Apache Flink in Java, Scala, and Python. You can also build Java-based streaming applications using Apache Beam. Apache Beam is an open source, unified model and set of language-specific SDKs for defining and executing data processing workflows.

☒ Legacy SQL 

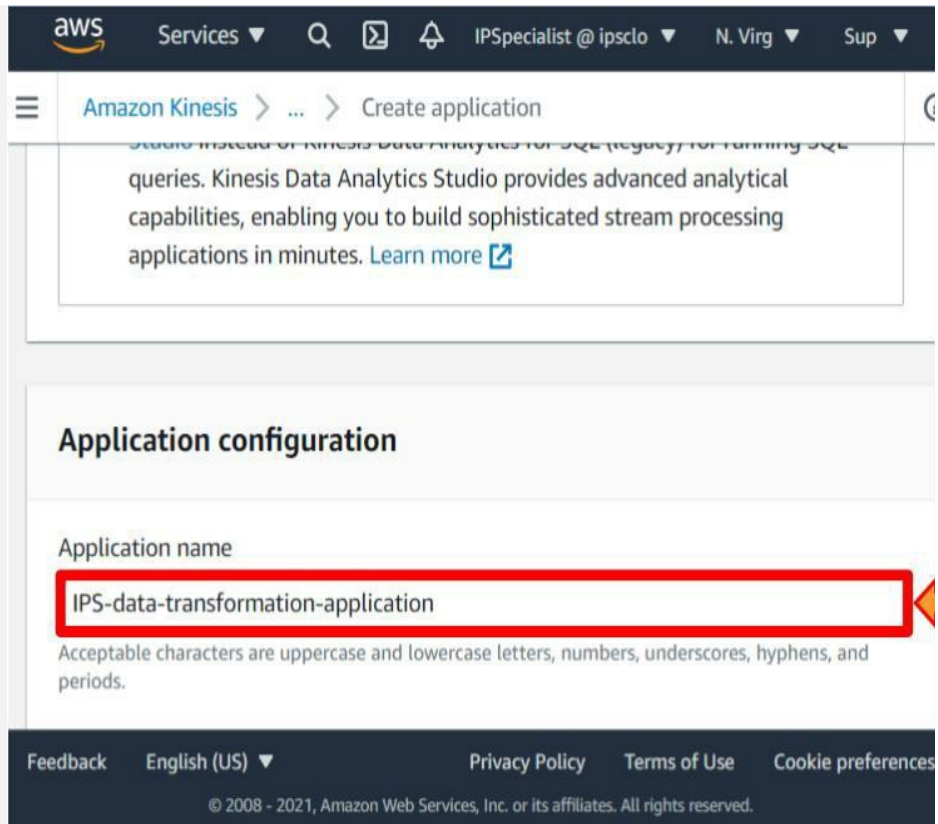
Process data in real-time using Kinesis Data Analytics legacy SQL engine, which provides an easy way to quickly query large volumes of streaming data. We do not recommend this option for new applications, instead use [Studio applications](#) .

 For new applications, we recommend that you use [Kinesis Data Analytics Studio](#) instead of Kinesis Data Analytics for SQL (legacy) for running SQL queries. Kinesis Data Analytics Studio provides advanced analytical capabilities, enabling you to build sophisticated stream processing applications in minutes. [Learn more](#) .

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie prefer

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

7. Give an application name **IPS-data-transformation-application**



8. In the **Description** box, type the following **IPS-data-transformation**

aws Services 🔻 🔍 📄 🔔 IPSpecialist @ ipsclo 🔻 N. Virg 🔻 Sup 🔻

☰ Amazon Kinesis > ... > Create application ⓘ

Acceptable characters are uppercase and lowercase letters, numbers, underscores, hyphens, and periods.

Description - *optional*

IPS-data-transformation-application

Tags - *optional*

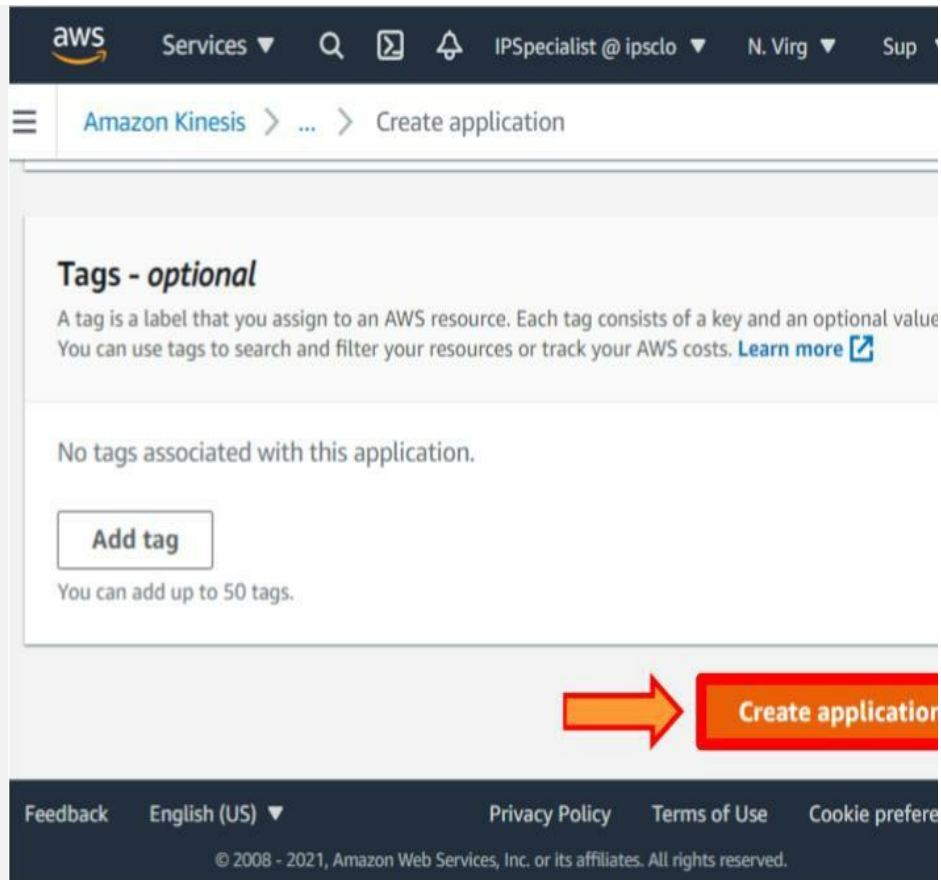
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs. [Learn more](#) 📄

No tags associated with this application.

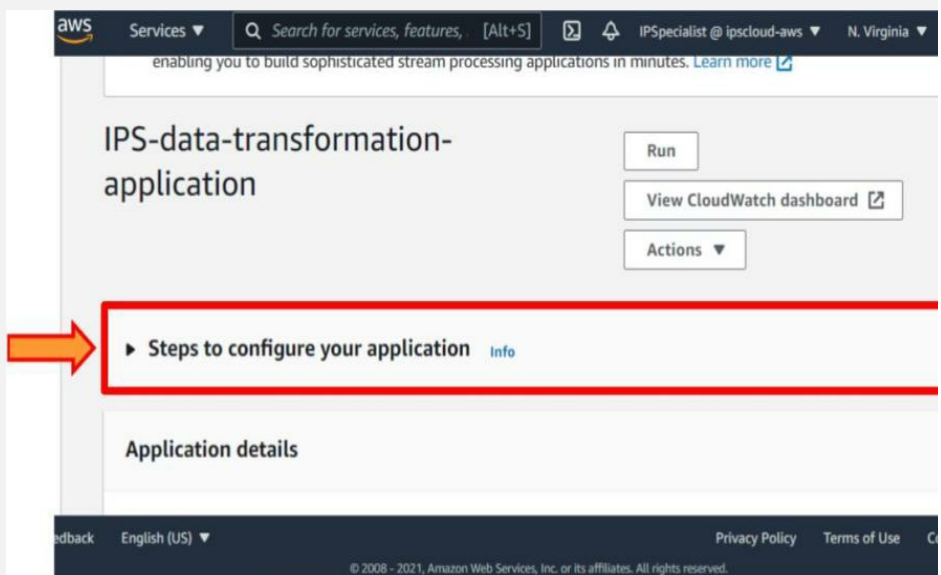
Feedback English (US) 🔻 Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

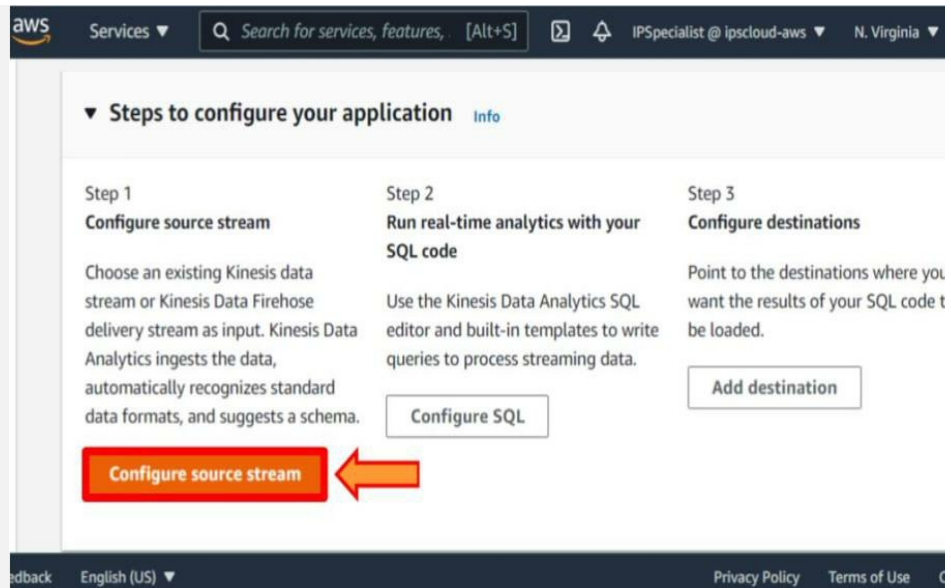
9. Click on the **Create application** button.



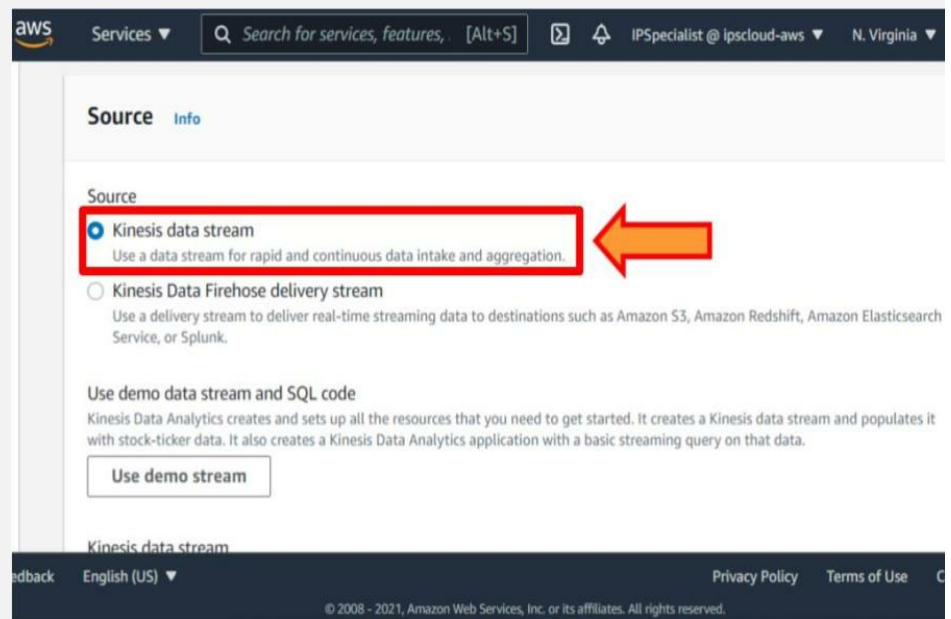
10. Click on the **Steps to configure your application**.



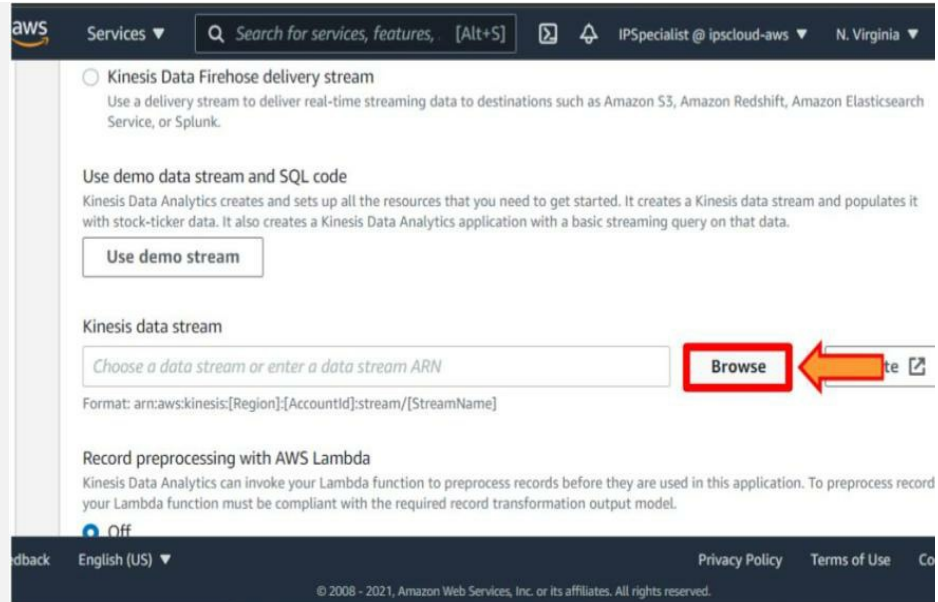
11. Click on the **Configure source stream** button.



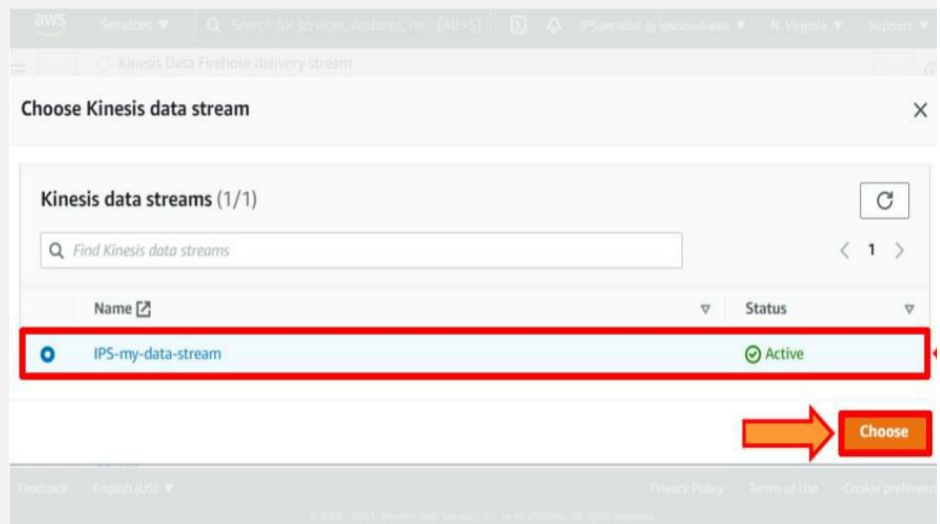
12. Select the **Kinesis Data Stream**.



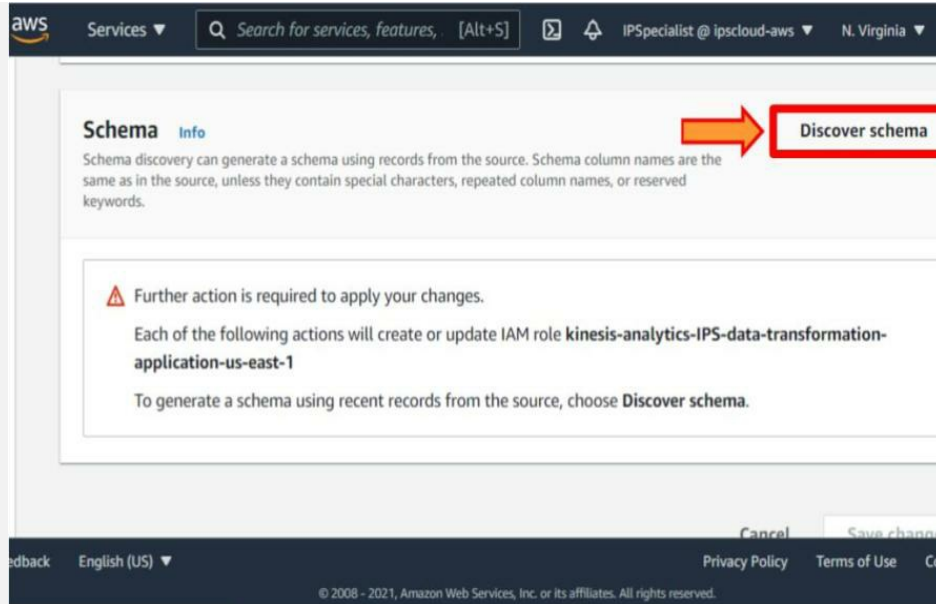
13. Click on the **Browse** button.



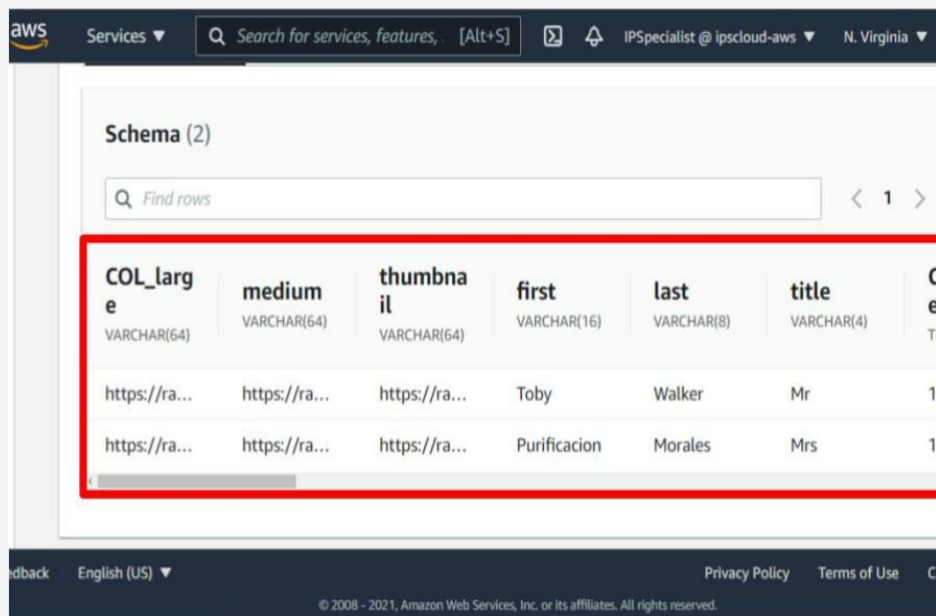
14. Select the **IPS-my-data-stream**, and then click on the **Choose**



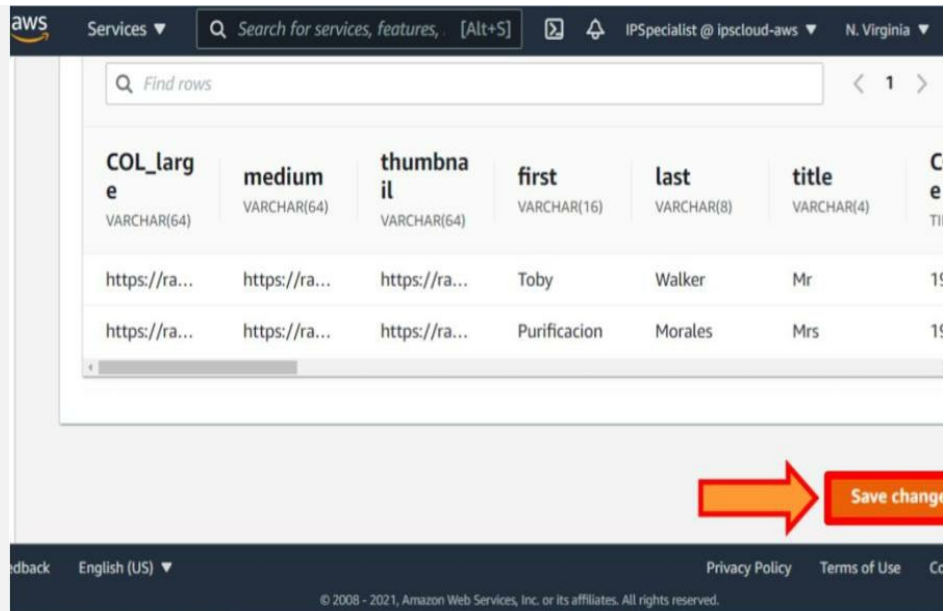
15. Scroll down, and click on the **Discover Schema** button.



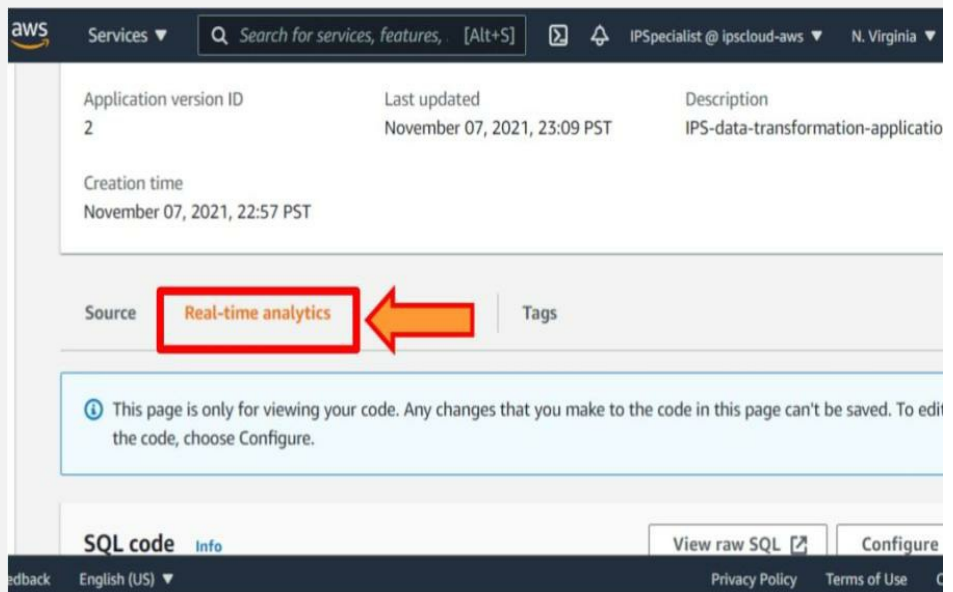
16. You will see the Raw and Formatted streaming data, which is co



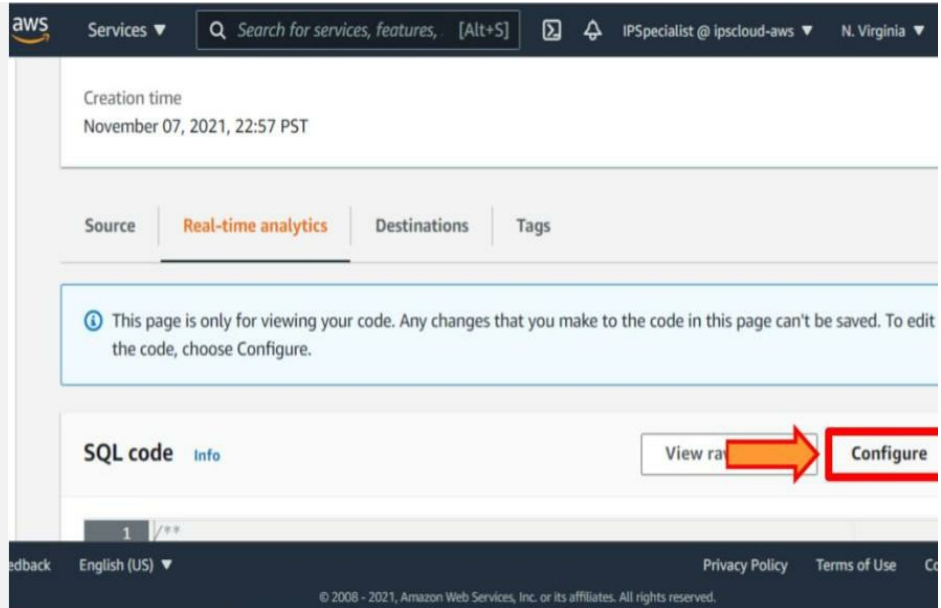
17. Click on the **Save changes** button.



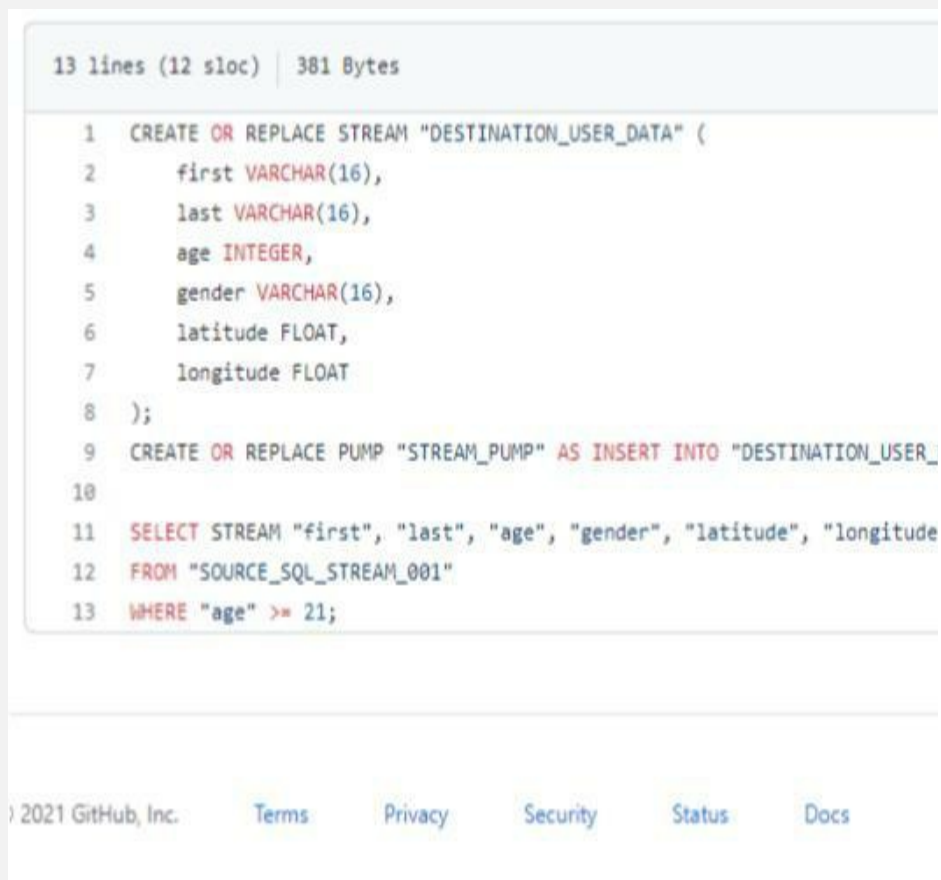
18. Click on the **Real-time analytics** tab.



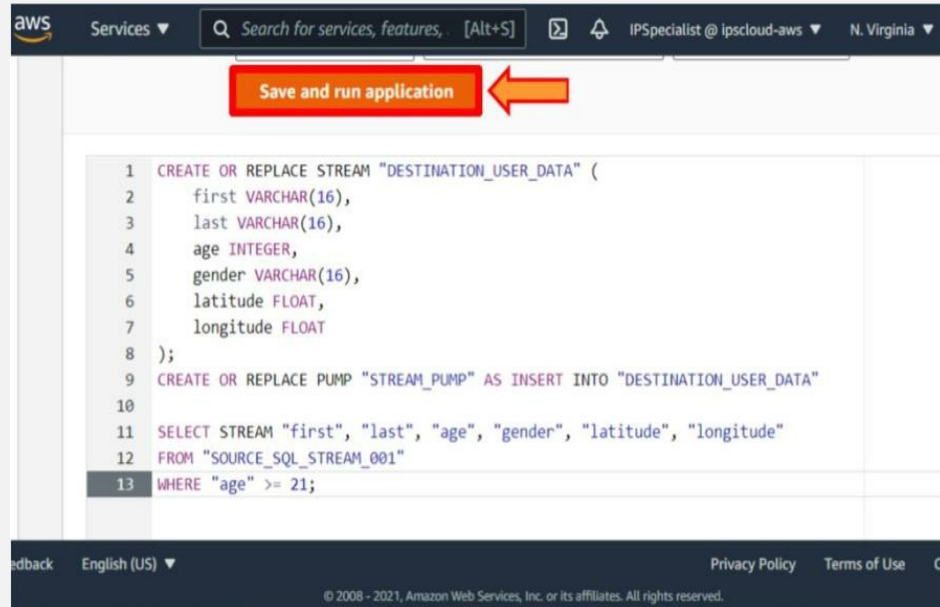
19. Then click on the **Configure** button.



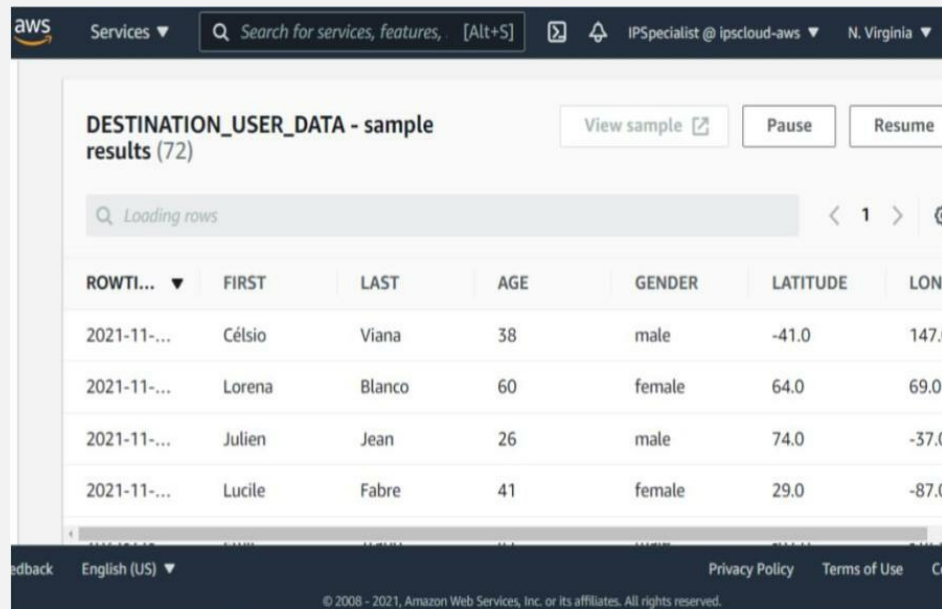
20. Copy and paste the SQL queries in the code editor from the file following Github link: [https://github.com/ACloudGuru-Resources/Course\\_AWS\\_Certified\\_Machine\\_Learning/blob/main/subset-transformation-query.sql](https://github.com/ACloudGuru-Resources/Course_AWS_Certified_Machine_Learning/blob/main/subset-transformation-query.sql)



21. Click on the **Save and run application** button.



22. It will take a few minutes. Once it is completed, you will see the



# Amazon Managed Streaming for Kafka

## Apache Kafka

Apache Kafka is a real-time data input and processing distributed data storage system. Streaming data is constantly created by hundreds of data sources, which often transmit data records simultaneously. A streaming platform must manage this continual input of data while still processing it sequentially and progressively. Apache Kafka was originally developed by LinkedIn and was made open source in 2011. The Apache community then took it over and a distributed streaming platform with three key capabilities.

Kafka offers its consumers the following three primary functions:

- Streams of recordings can be published and subscribed to.
- Streams of records can be effectively stored in the sequence in which they were created.
- Real-time processing of record streams.

Kafka is generally used to construct real-time streaming data pipelines and applications that react to data streams. It integrates communications, storage, and stream processing to enable the storage and analysis of both historical and real-time data.

A data pipeline consistently processes and transports data from one system to another, whereas a streaming application consumes data streams. For example, you use Kafka to ingest if you want to build a data pipeline. That uses user activity data to track how people use your website

in real-time—store streaming data while sending reads to the applications that run the data pipeline. Kafka is also frequently used as a message broker solution, a platform for processing and mediating messages between two applications.

Kafka combines two communications paradigms, queuing and publish-subscribe, to offer customers the core features of both. Queuing distributes data processing over numerous consumer instances, making it very scalable. Traditional queues, on the other hand, do not support multiple subscribers. Because every message is delivered to every subscriber, the publish-subscribe strategy cannot distribute work across several worker processes. Kafka uses a partitioned log design to connect these two systems. A log is an ordered sequence of data divided into segments or partitions that correspond to various subscribers. It implies that several subscribers can be assigned to the same topic, where each one is allocated a section to allow for greater scalability. Finally, Kafka's paradigm includes a replay ability, which enables many independent applications reading from data streams to operate independently and at their rate.

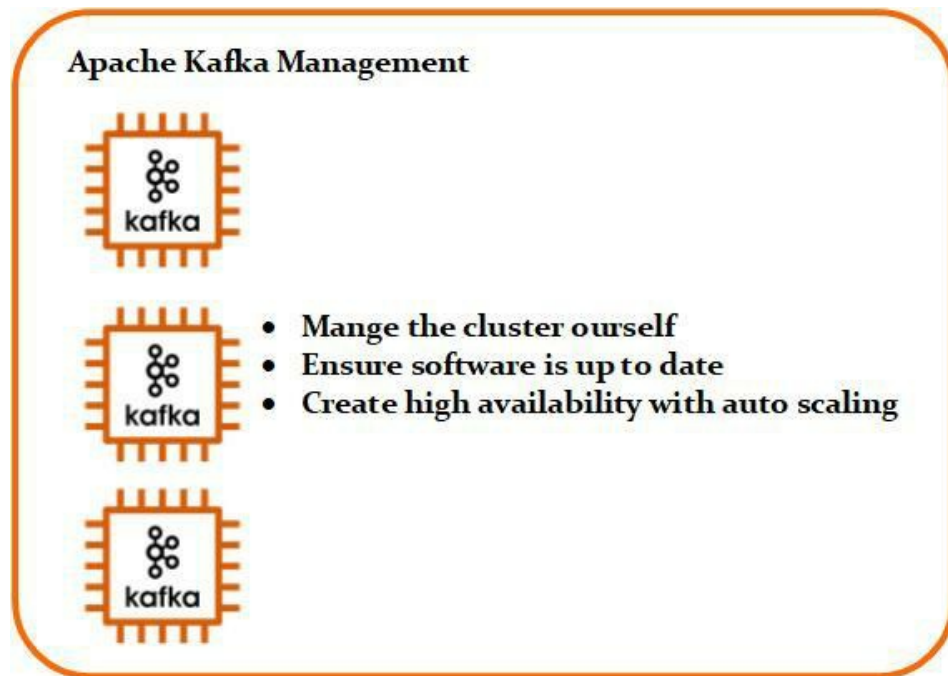
**EXAM TIP:** Apache Kafka is a distributed data storage system that accepts and processes data in real-time. Hundreds of data sources regularly generate streaming data, which frequently transmits data records at the same time. A streaming platform must manage this continuous data flow while still processing it sequentially and progressively.

## **Apache Kafka Publish/Subscribe Conceptually**

To understand how to publish and subscribe or pub/sub systems work, think about a coffee house. You know that there is always a bulletin board with advertisements inside a coffee house, for example, someone trying to give out guitar lessons or someone looking for a specific math tutor. What people do is post messages on the bulletin board. They can put information into a central place without knowing the actual receiver or the people reading the messages. The publisher is the person who puts the post-it note on the board, while the subscribers are those people who go to it and take down the post-it note or read the advertisement. In this scenario, the bulletin board is known as a broker. It allows publishers and subscribers to decouple from one another, which reduces complexity.

## **Apache Kafka Management**

The Apache Kafka is an open-source piece of software. You can download that and install it onto a server or an EC2 instance. However, if you decide to go down that path, you have to manage all of these Apache Kafka servers yourself. You have to ensure that the cluster is up to date and make sure that it is auto-scaling as well. It scales up to demand and scales back down whenever the streaming data spikes decrease. This is where MSK, Managed Streaming service for Kafka, comes to the rescue.



*Figure 4-31: Apache Kafka Management*

## **Amazon MSK**

Amazon Managed Streaming for Apache Kafka (Amazon MSK) is a completely managed service that allows you to design and run applications that handle streaming data using Apache Kafka. Amazon MSK takes control-plane actions, including building, updating, and removing clusters. It enables the usage of Apache Kafka data-plane tasks, such as data production and data consumption. It runs Apache Kafka open-source versions and implies that existing applications, tools, and plugins from partners and the Apache Kafka community are supported without application code modifications.

The figure below depicts the operation of Amazon MSK:

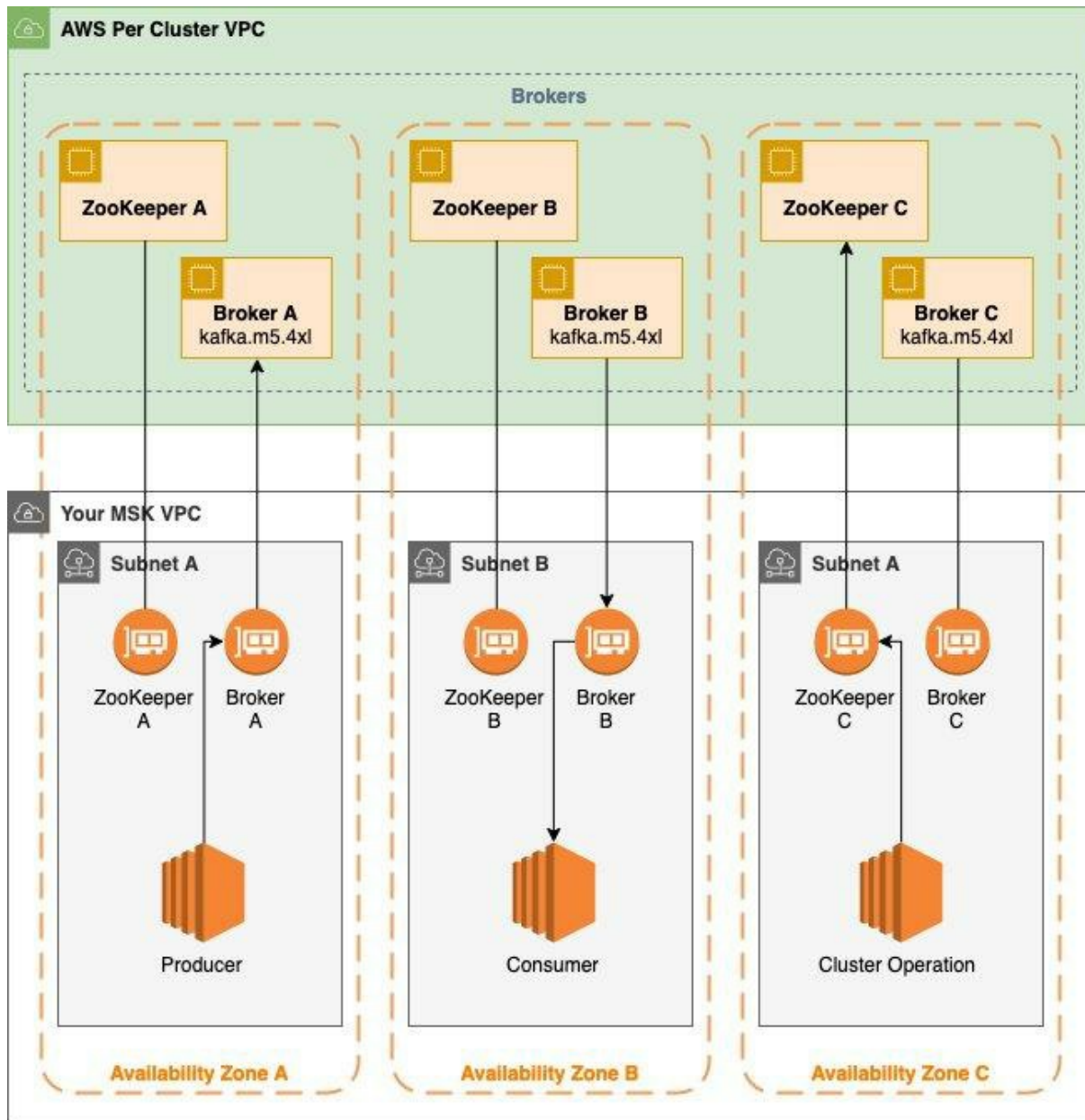


Figure 4-32: Amazon MSK

The figure depicts the interplay of the following elements:

- **Broker nodes:** You indicate how many broker nodes you want Amazon MSK to construct in each Availability Zone when building an Amazon MSK cluster. There is one broker per Availability Zone in the sample cluster depicted in this diagram.

Each Availability Zone has its subnet of virtual private cloud (VPC).

- **Zookeeper nodes:** Amazon MSK will also set up Apache Zookeeper nodes for you. Apache Zookeeper is a free and open-source server that allows for highly dependable distributed coordination.
- **Producers, consumers, and topic creators:** Amazon MSK allows you to establish topics, produce and consume data, and leverage Apache Kafka data-plane functions.
- **Cluster Operations:** To accomplish control-plane tasks, you can utilize the AWS Management Console, the AWS Command Line Interface (AWS CLI), or the SDK APIs. For example, you can establish or remove an Amazon MSK cluster, see the attributes of a cluster, and change the number and type of brokers in a cluster.

Amazon MSK identifies and recovers automatically from the most frequent cluster failure scenarios, allowing your producer-consumer applications to continue their write and read activities with minimum effect. When Amazon MSK detects a broker failure, it either mitigates it or replaces the unhealthy or unreachable broker with a new one. Furthermore, it reuses storage from the older broker wherever feasible to limit the amount of data that Apache Kafka needs to duplicate. The impact on your availability is limited to the time it takes Amazon MSK to perform the detection and recovery. Following a recovery, your producer and consumer applications will be able to interact using the same broker IP addresses that they used before the incident.

**EXAM TIP:** Amazon MSK (Amazon Managed Streaming for Apache Kafka) is a fully managed service. It enables you to create and execute applications that use Apache Kafka to handle

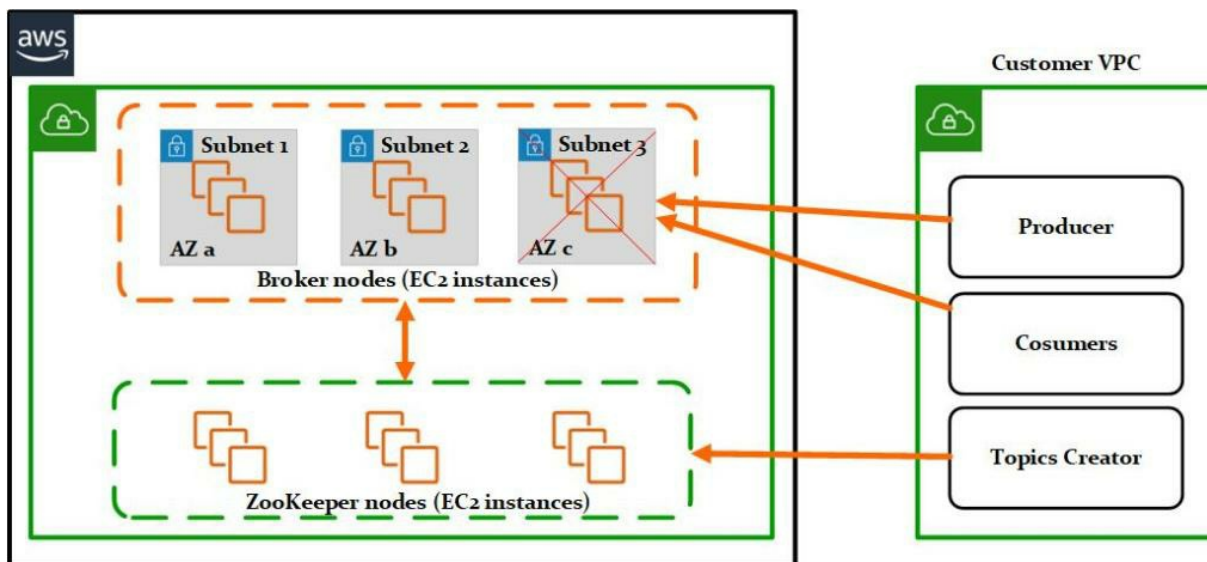
streaming data. Amazon MSK performs control-plane operations, such as cluster creation, updating, and removal.

## **MSK Architecture**

You can think about it as being the bulletin board or kind of the central place where the messages will live, or in our case, our streaming data. Hence, you have our AWS account, and within that, you have VPC. Inside of that VPC, you have our broker nodes. You have different broker nodes that live in different availability zones. In our case, you have three different availability zones, and each will have a single subnet in the availability zones. Thus, whenever you create an MSK cluster, you need to provide different subnets. These need to live in different availability zones for high availability. It will also create Apache zookeeper nodes, the coordinating device between our broker and zookeeper nodes. Messages are published and subscribed to through a customer VPC or a VPC in the same AWS account. You have producers that publish or produce information onto the broker nodes. Then you have consumers or subscribers who subscribed to those messages to get the published information back. The job of a zookeeper is to handle all of the coordination and to create topics. A topic is just a category or a feed name where the records are published to.

The zookeeper nodes are in charge of coordination and topic creators. However, they are also taking care of the different partitions similar to shards in Kinesis Data Streams. One of the key benefits of using MSK is that it automatically detects and recovers from our clusters' most

common failure scenarios. This means that our producers and consumer applications can continue writing and reading operations with the most minimal impact. That is one of the reasons you have it in several different availability zones for our broker nodes. As a result, MSK recognizes a problem and mitigates it by replacing the unhealthy or unavailable broker with a new broker. This is the power behind using MSK and Apache Kafka through AWS's managed service.



*Figure 4-33: MSK Architecture*

**EXAM TIP:** Amazon MSK will also configure Apache Zookeeper nodes on your behalf. Apache Zookeeper is a free and open-source server for highly reliable distributed coordination.

## Compare MSK & Kinesis Data Stream

The big question is, how does MSK differ from Kinesis Data Streams? One way they differ is by their provisioning model. In Kinesis Data Streams, you will increase the number of shards when you increase the

throughputs. However, with MSK, you would need to increase the types of instances or the number of instances within the MSK cluster. These are going to be limited by the instance type that you are using within your cluster. With Kinesis Data Streams, scaling is almost seamless, meaning that you do not have to do anything to scale up your applications.

All you have to do is add more shards. With MSK, it can be a little bit trickier. Scaling is not seamless to the client. One important key feature to remember is the retention time. For Kinesis Data Streams, it is one day, and the max can be for seven days. However, for MSK, the default retention time is seven days, but the maximum is unlimited. Hence, you can have data stored for an unlimited amount of time in MSK in your broker nodes. With Kinesis Data Streams, it has a deep AWS integration. It is almost seamlessly integrated with tons of different AWS services. It makes it easy to use and integrate within your other applications, but with MSK, it is more for third-party tooling. Hence, if there is some service that you are using that Kinesis Data Streams just does not interact with very well, then MSK might be a better option. You will have no problems choosing one over the other for your streaming service solutions.

| Kinesis Data Stream                  | MSK   |
|--------------------------------------|---|
| Throughput provisioning model        | Cluster provisioning model                  |
| Seamless scaling                     | Scaling is not seamless to the client       |
| Retention time is 1 day (max 7 days) | Retention time is 7 days (max is unlimited) |
| Deep AWS integration                 | Strong 3 <sup>rd</sup> party tooling        |

*Figure 4-34: Compare MSK & Kinesis Data Stream*

## Streaming Services Uses Cases

Here are some scenarios are given to see and decide which streaming service to use:

| Task at hand  | Which Kinesis service to use? | Why?  |
|---|-------------------------------|---|
| Need to stream Apache log files directly from (100) EC2 instances and store them into Redshift. | Kinesis Firehose              | Firehose can easily stream data directly to a final destination. First, the data is loaded into S3 and then copied into Redshift using the <b>COPY</b> command. |
| Need to stream live video coverage of a sporting event to                                       |                               | Kinesis Video Streams process real-time streaming video data  |

|   |   |   |
|---|---|---|
| distribute to customers in near real-time.  | Kinesis Video Streams                   | (audio, images, radar) and feed into other AWS services.  |
| Need to transform real-time streaming data and immediately feed it into a custom ML application.                      | Kinesis Streams                         | Kinesis Streams allow for streaming huge amounts of data, process/transforming it, and then storing or feeding it into custom applications or other AWS services. |
| Need to stream and store every newspaper article since 1850 for consumer applications, and the messages never expire. | Amazon Managed Streaming for Kafka(MSK) | MSK gives us the ability to tune for optimal throughput and latency, as well as an <b>unlimited retention period</b> .  |
| Real-time data must be queried, metric graphs must be created, and output stored in S3.                               | Kinesis Analytics                       | Kinesis Analytics gives you the ability to run SQL queries on streaming data, then store or feed the output into other AWS services.                              |

## **Lab 4-02: Joining, Enriching, & Transforming Streaming Data with Amazon Kinesis**

### **Introduction**

#### **Kinesis Data Streams**

You may use Amazon Kinesis Data Streams to create custom applications that process or analyze streaming data for specific purposes. To an Amazon Kinesis data stream, you may constantly add data from hundreds of thousands of sources, such as clickstreams, application logs, and social media. Within seconds, your Amazon Kinesis Applications will be able to read and analyze data from the stream.

#### **Kinesis Data Analytics**

Amazon Kinesis Data Analytics is the most straightforward method for converting and analyzing real-time streaming data using Apache Flink. Apache Flink is an open-source data stream processing framework and engine. Apache Flink applications' creation, maintenance, and integration with other AWS services are made easier using Amazon Kinesis Data Analytics.

Amazon Kinesis Data Analytics handles everything needed to constantly operate streaming applications and grow automatically to meet the amount and throughput of your incoming data. There are no servers to use with Amazon Kinesis Data Analytics and no minimum charge or setup cost. Thus, you pay for the resources your streaming applications utilize.

## **Kinesis Data Firehose**

Kinesis Data Firehose is a solution for streaming ETL. It is the most convenient method for loading streaming data into data storage and analytics tools. It can collect, transform, and load streaming data into Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, and Splunk. It enables near-real-time analytics with your existing business intelligence tools and dashboards. It is a fully managed service that scales automatically to meet your data flow and does not require any ongoing management. It may also batch, compress, and encrypt data before loading it, reducing storage requirements at the destination and enhancing security.

## **AWS Lambda**

AWS Lambda allows you to run code without the need to create or manage servers. There is no charge when your code is not executing; you only pay for the compute time you use. You can run code for nearly any application or backend service with Lambda, and you do not have to worry about administration. Upload your code, and Lambda will handle everything necessary to run and grow it with high availability. You may configure your code to be automatically triggered by other AWS services, or you can access it directly from any computer or smartphone app.

## **Amazon Simple Storage Service S3**

Amazon S3 is a type of object storage that allows you to store and recover any quantity of data from any location. It is a low-cost storage solution with business resilience, reliability, efficiency, privacy, and infinite expansion.

Amazon S3 is a web service that allows you to store and retrieve an

infinite quantity of data from any place and at any time. You may quickly create projects that integrate cloud-native storage using this service. Because Amazon S3 is easily customizable and you only pay for what you use, you can start small and scale up as needed without sacrificing performance or dependability.

Amazon S3 is also built to be highly adaptable. Instead of finding out how to store their data, Amazon S3 allows developers to focus on innovation. You can build a simple FTP system or a complex web application like the Amazon.com retail website. You can read the same piece of data a million times or only for emergency disaster recovery, as well as store whatever type and amount of data you desire.

### **AWS CloudWatch**

Amazon CloudWatch is a tracking service for Amazon Web Services (AWS) cloud services and software. Amazon CloudWatch may be used to collect and monitor data, monitor log files, and trigger alarms. Amazon CloudWatch may be used to monitor Amazon EC2 instances, DynamoDB tables, and RDS database instances, as well as custom metrics and log files generated by your applications and services. Your whole system may be monitored using Amazon CloudWatch, which allows you to keep track of resource use, application performance, and operational health. These insights might help you react and keep your app working smoothly.

### **AWS Identity and Access Management (IAM)**

Individuals and groups can be granted secure access to your AWS resources by using IAM. It allows you to create and manage IAM users and provide them access to your resources. Additionally, you have the option of granting access to users outside of AWS (federated users).

- **Managed Policy:** This contains the permission required to stop an EC2 instance.
- **Inline Policy:** This allows this role to be passed to another service.
- **Trust Policy:** Allows System Manager and EC2 to assume the role. It enables EC2 to register with the Systems Manager and Systems Manager to stop the EC2 instance.

## **AWS DynamoDB**

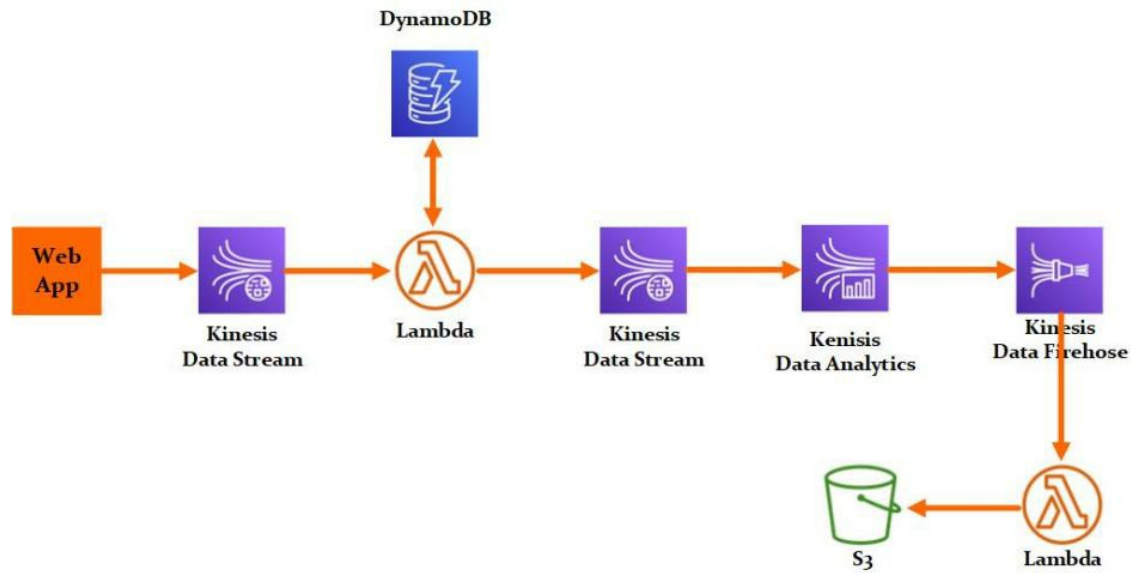
DynamoDB is a high-performance, scalable, and non-relational database service. Customers may use DynamoDB to transfer the administrative duties of running and managing distributed databases to AWS, allowing them to focus on other things, such as hardware provisioning, setup and configuration, throughput capacity planning, replication, software patching, and cluster scaling.

## **Problem**

You work as a Data Analytics engineer for an organization with a mobile application that allows users to place grocery store orders. The organization gives you a task to create a streaming application that helps rewards users who spend more than \$100 on grocery orders. How can you make this type of application?

## **Solution**

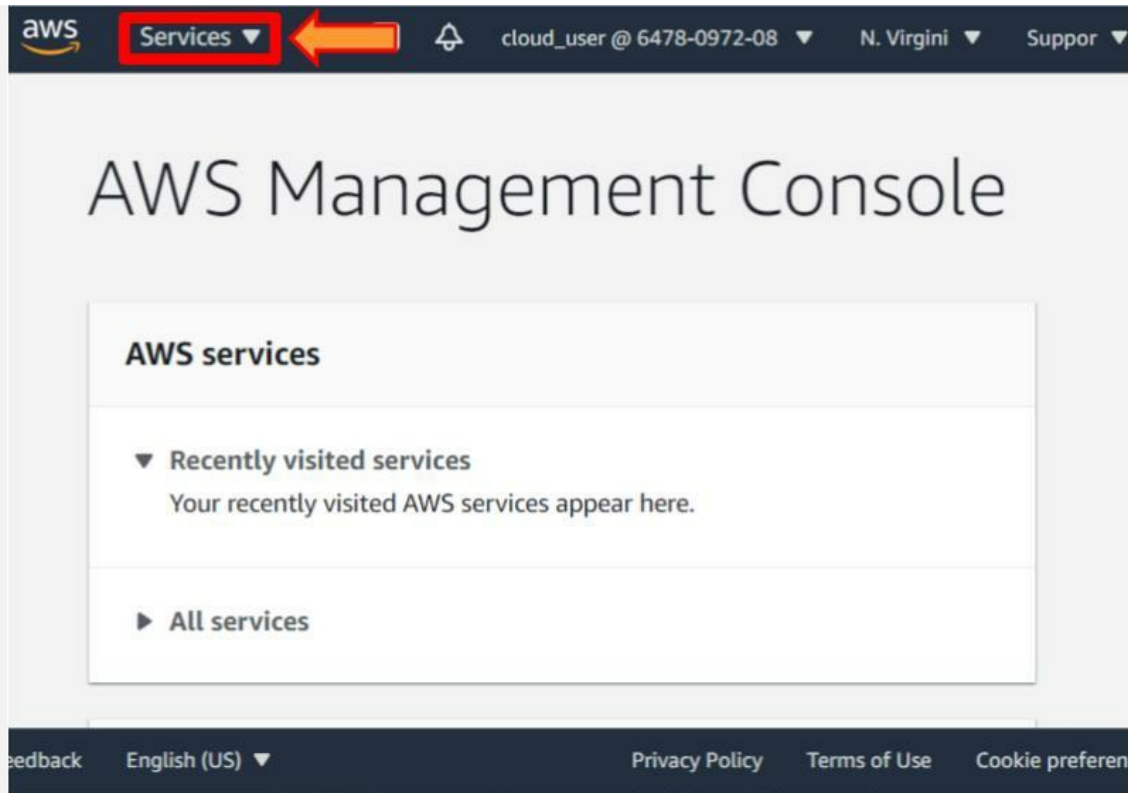
The solution to creating a streaming application is to use AWS services. You will create a streaming application that uses Kinesis Stream, Lambda, and Kinesis Data Analytics. Kinesis Firehose is used to ingest, enrich, filter, and store users' orders into an S3 data lake.



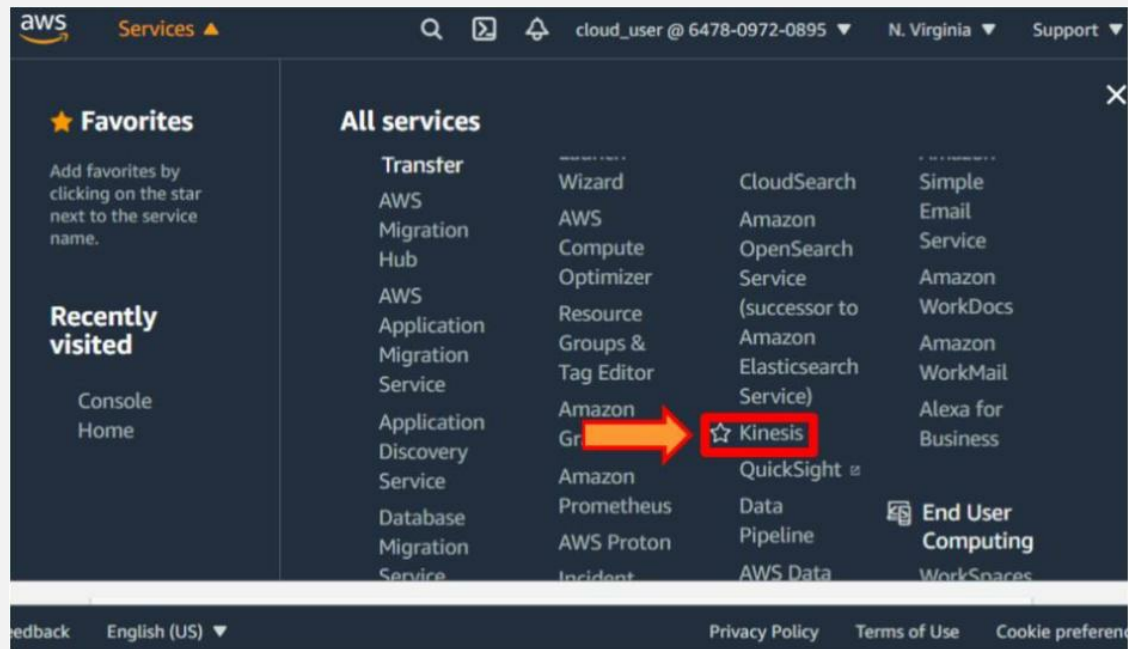
*Figure 4-35: Joining, Enriching & Transforming Streaming Data with Amazon Kinesis*

### Step 1: Create Kinesis Data Stream

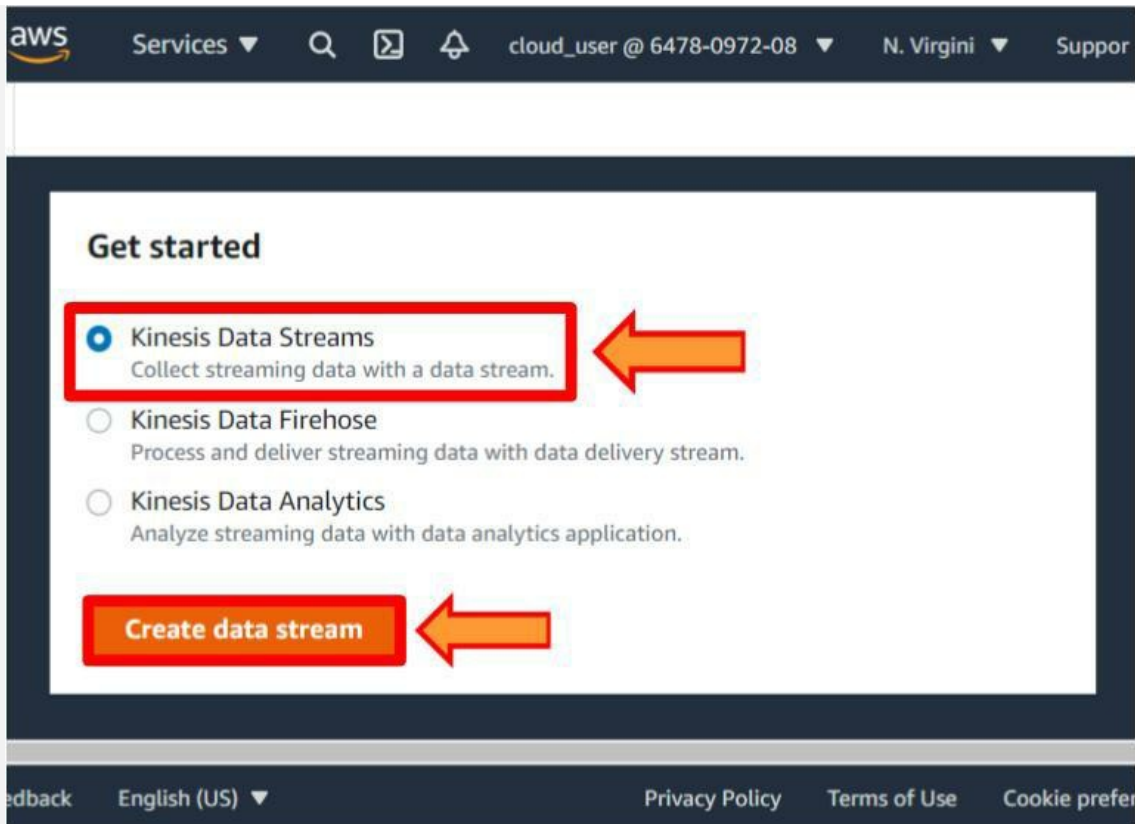
1. Log in to the **AWS Console**.
2. Click on the **Services**.



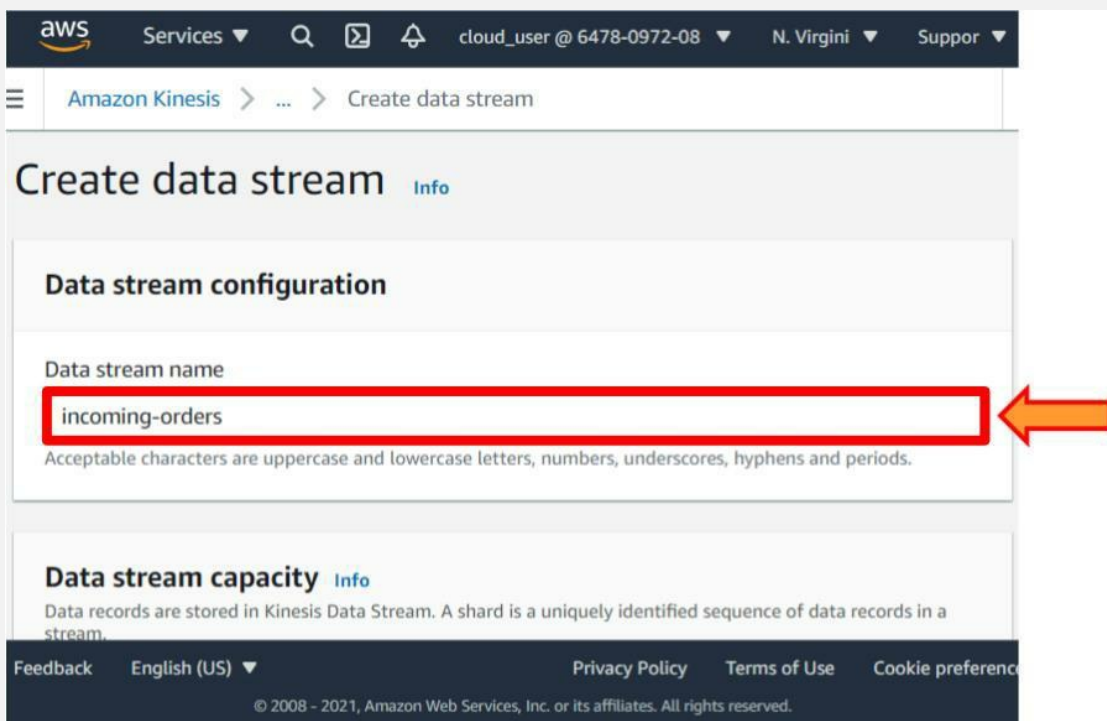
3. Select the **Kinesis** from the **Analytics**.



4. Create a data stream for incoming orders
5. Select the **Kinesis Data Streams**.
6. Click on the **Create Data Stream** button.



7. Give the name **incoming-orders**.



8. Enter the **Number of shards: 1**.

aws Services 🔻 🔍 📄 🔔 cloud\_user @ 6478-0972-08 🔻 N. Virgini 🔻 Support 🔻

Amazon Kinesis > ... > Create data stream

### Data stream capacity [Info](#)

Data records are stored in Kinesis Data Stream. A shard is a uniquely identified sequence of data records in a stream.

**Number of open shards**

The total capacity of a stream is the sum of the capacities of its shards. Enter number of provisioned shards to see total data stream capacity.

← estimator

Minimum: 1, Maximum available: 500, Account quota limit: 500.  
[Request shard quota increase](#) 🔗

**Total data stream capacity**

Shard capacity is determined by the number of open shards. Each open shard ingests up to 1 MiB/second and 1000 records/second and emits up to 2 MiB/second. If writes and reads exceed capacity, the application will receive throttles.

Feedback English (US) 🔻 Privacy Policy Terms of Use Cookie preference

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

9. Click on the **Create Data Stream** button.

aws Services 🔻 🔍 📄 🔔 cloud\_user @ 6478-0972-08 🔻 N. Virgini 🔻 Support 🔻

Amazon Kinesis > ... > Create data stream

Minimum: 1, Maximum available: 500, Account quota limit: 500.  
[Request shard quota increase](#) 🔗

**Total data stream capacity**

Shard capacity is determined by the number of open shards. Each open shard ingests up to 1 MiB/second and 1000 records/second and emits up to 2 MiB/second. If writes and reads exceed capacity, the application will receive throttles.

**Write capacity**  
1 MiB/second and 1000 records/second

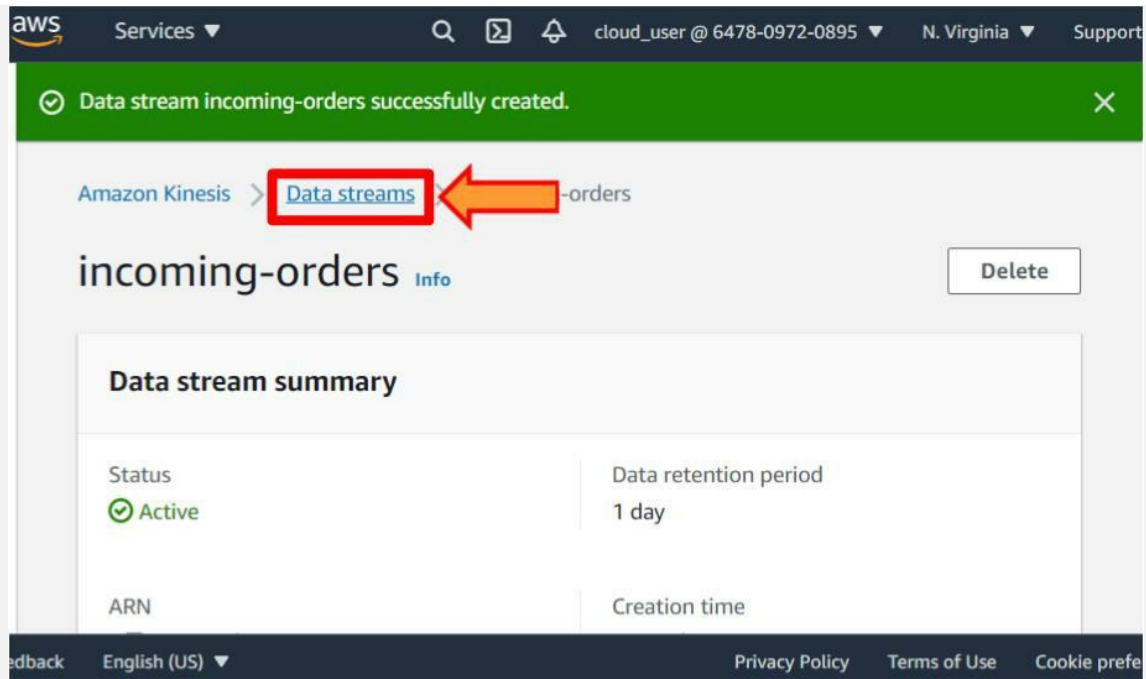
**Read capacity**  
2 MiB/second

→ **Create data stream**

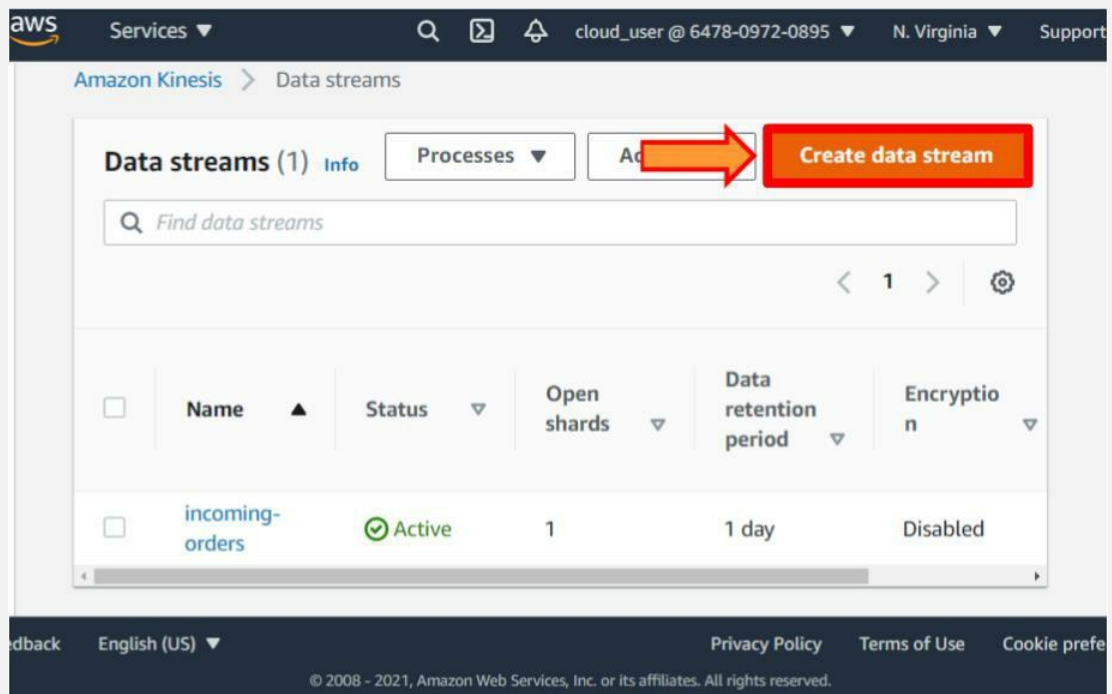
Feedback English (US) 🔻 Privacy Policy Terms of Use Cookie preference

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

10. Click on the **Data Streams**.



11. Create a second data stream for enriched orders
12. Click on the **Create Data Stream** button.



13. Give a name **enriched-orders**.

aws Services ▾ 🔍 ⓘ 🔔 cloud\_user @ 6478-0972-0895 ▾ N. Virginia ▾ Support

Amazon Kinesis > Data streams > Create data stream

## Create data stream [Info](#)

### Data stream configuration

Data stream name

enriched-orders

←

Acceptable characters are uppercase and lowercase letters, numbers, underscores, hyphens and periods.

### Data stream capacity [Info](#)

Data records are stored in Kinesis Data Stream. A shard is a uniquely identified sequence of data records in a stream.

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

14. Enter the **Number of shards: 1**.

aws Services ▾ 🔍 ⓘ 🔔 cloud\_user @ 6478-0972-0895 ▾ N. Virginia ▾ Support

### Data stream capacity [Info](#)

Data records are stored in Kinesis Data Stream. A shard is a uniquely identified sequence of data records in a stream.

Number of open shards

The total capacity of a stream is the sum of the capacities of its shards. Enter number of provisioned shards to see total data stream capacity.

1

← estimator

Minimum: 1, Maximum available: 499, Account quota limit: 500.  
[Request shard quota increase](#) ⓘ

### Total data stream capacity

Shard capacity is determined by the number of open shards. Each open shard ingests up to 1 MiB/second and 1000 records/second and emits up to 2 MiB/second. If writes and reads exceed capacity, the application will receive throttles.

Write capacity

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

15. Click on the **Create Data Stream** button.

aws Services ▾ 🔍 ⓘ 🔔 cloud\_user @ 6478-0972-0895 ▾ N. Virginia ▾ Support


1 Shard estimator

Minimum: 1, Maximum available: 499, Account quota limit: 500.  
[Request shard quota increase](#)

**Total data stream capacity**  
Shard capacity is determined by the number of open shards. Each open shard ingests up to 1 MiB/second and 1000 records/second and emits up to 2 MiB/second. If writes and reads exceed capacity, the application will receive throttles.

Write capacity  
1 MiB/second and 1000 records/second

Read capacity  
2 MiB/second

 **Create data stream**

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

16. Hence, both Kinesis data streams are created.

aws Services ▾ 🔍 ⓘ 🔔 cloud\_user @ 6478-0972-0895 ▾ N. Virginia ▾ Support

**Data streams (2)** Info Processes ▾ Actions ▾ **Create data stream**

Find data streams

< 1 > ⚙

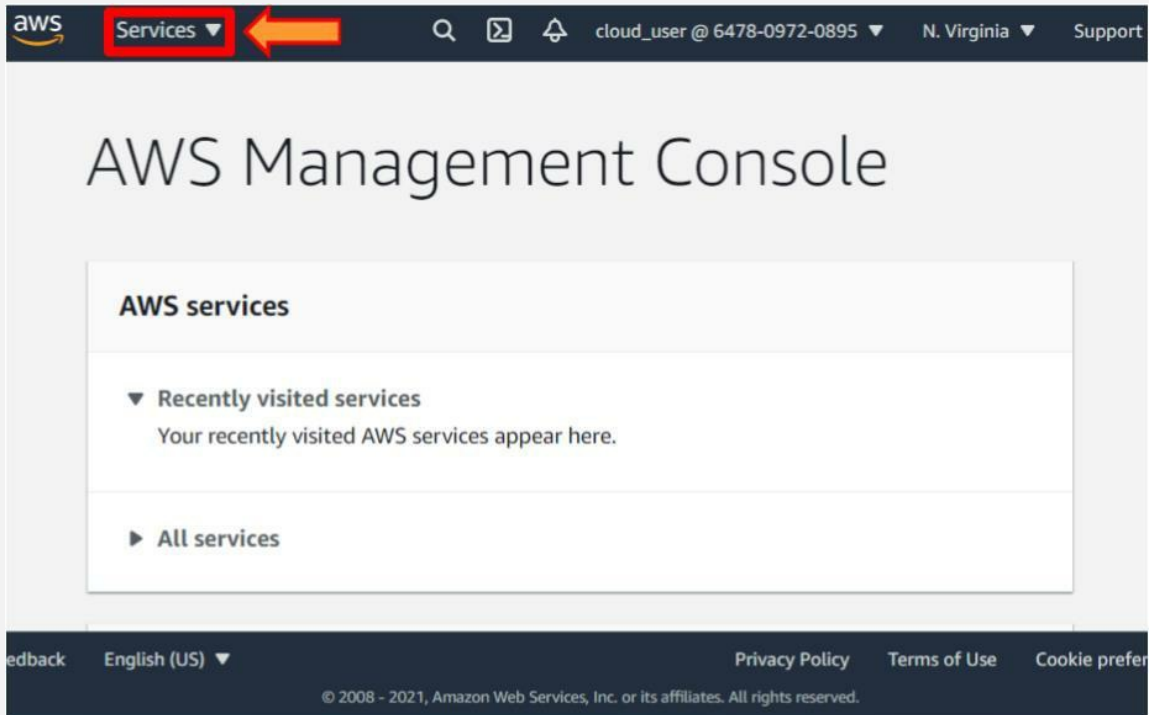
| <input type="checkbox"/> | Name ▲          | Status ▾ | Open shards ▾ | Data retention period ▾ | Encryption ▾ |
|--------------------------|-----------------|----------|---------------|-------------------------|--------------|
| <input type="checkbox"/> | enriched-orders | ✓ Active | 1             | 1 day                   | Disabled     |
| <input type="checkbox"/> | incoming-orders | ✓ Active | 1             | 1 day                   | Disabled     |

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

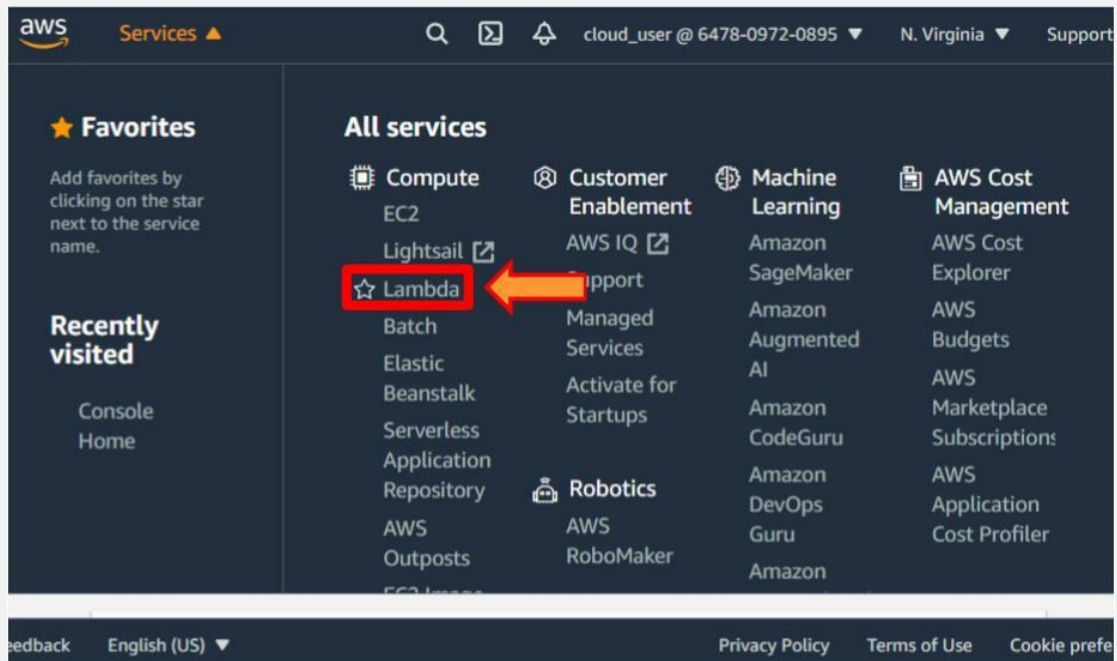
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

## Step 2: Create Lambda Function to Enrich Records

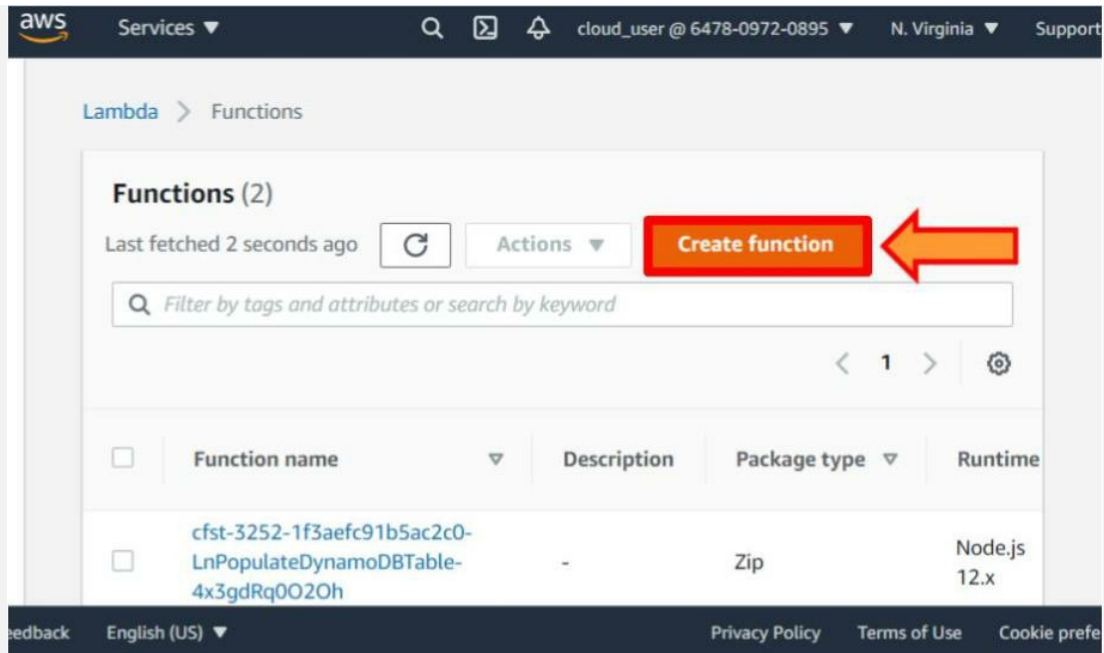
1. Click on the **Services**.



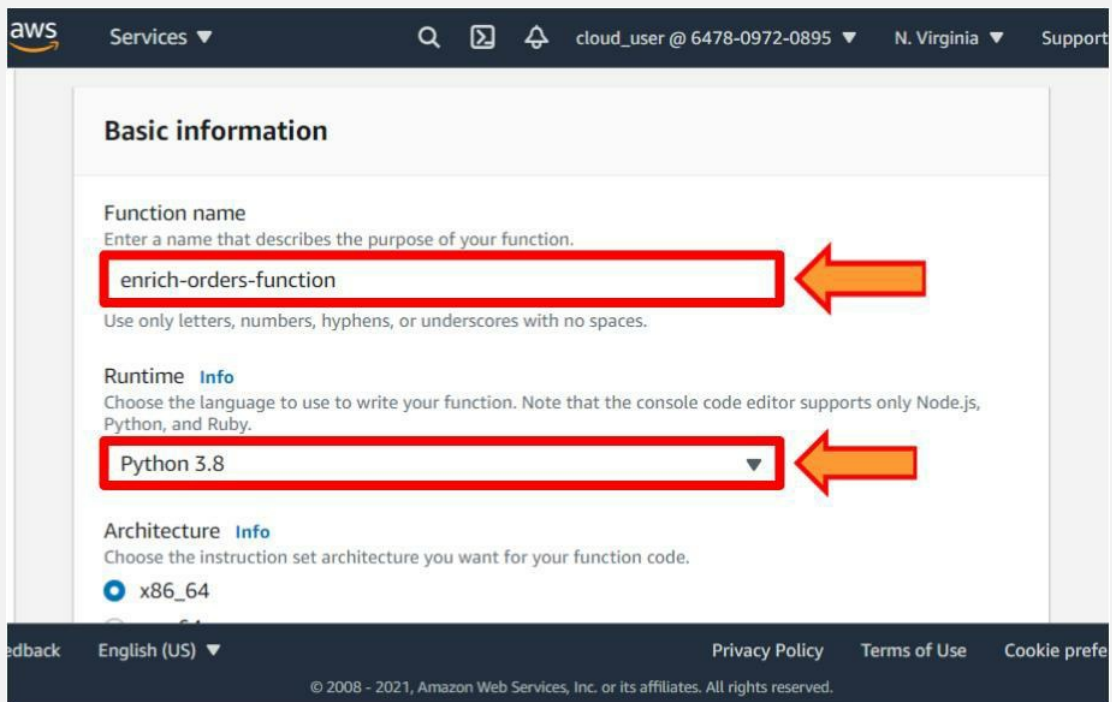
2. Select the **Lambda** from the **Compute**.



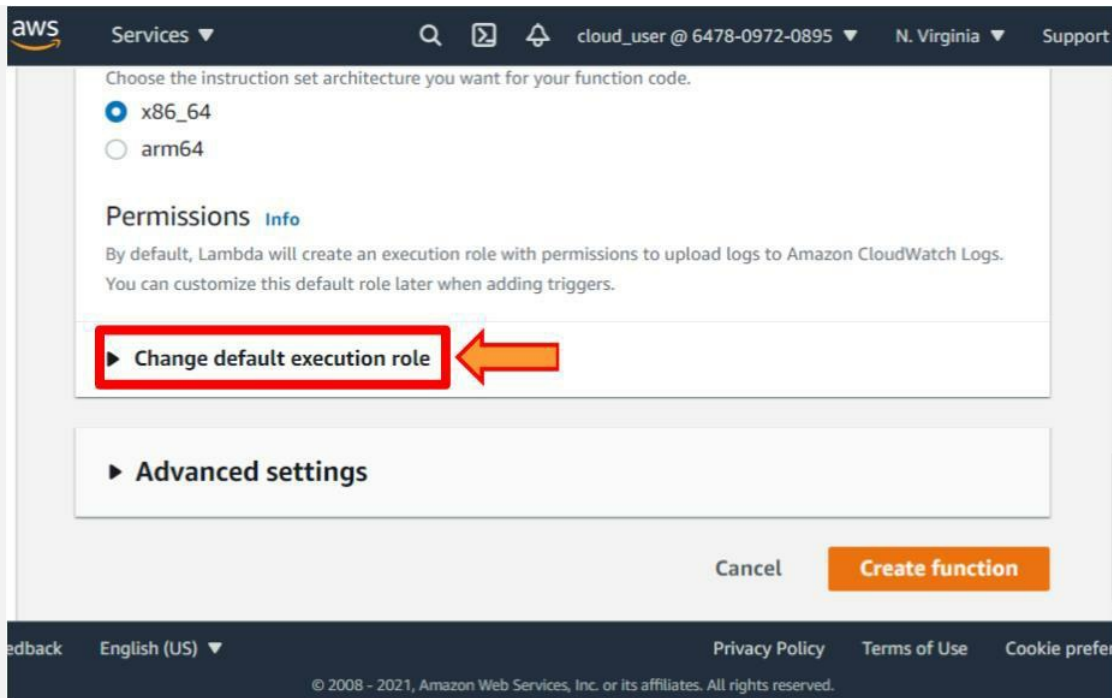
3. Click on the **Create function** button.



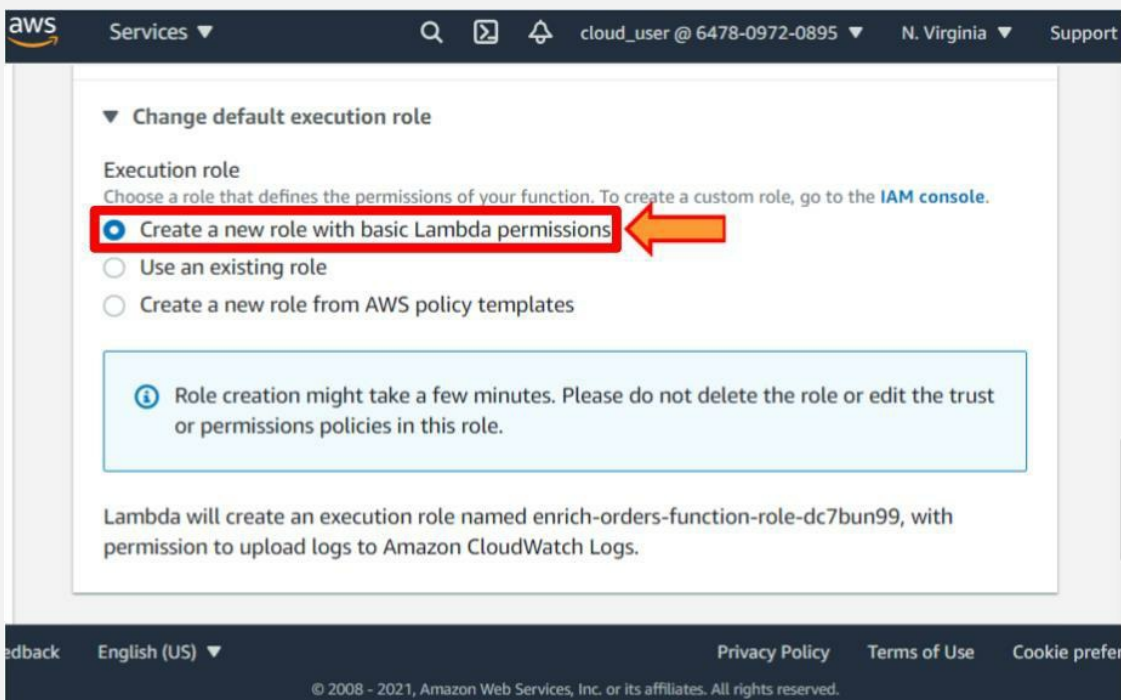
4. Enter the name for the function **enrich-orders-function**.
5. Select the **Python 3.8** Runtime.



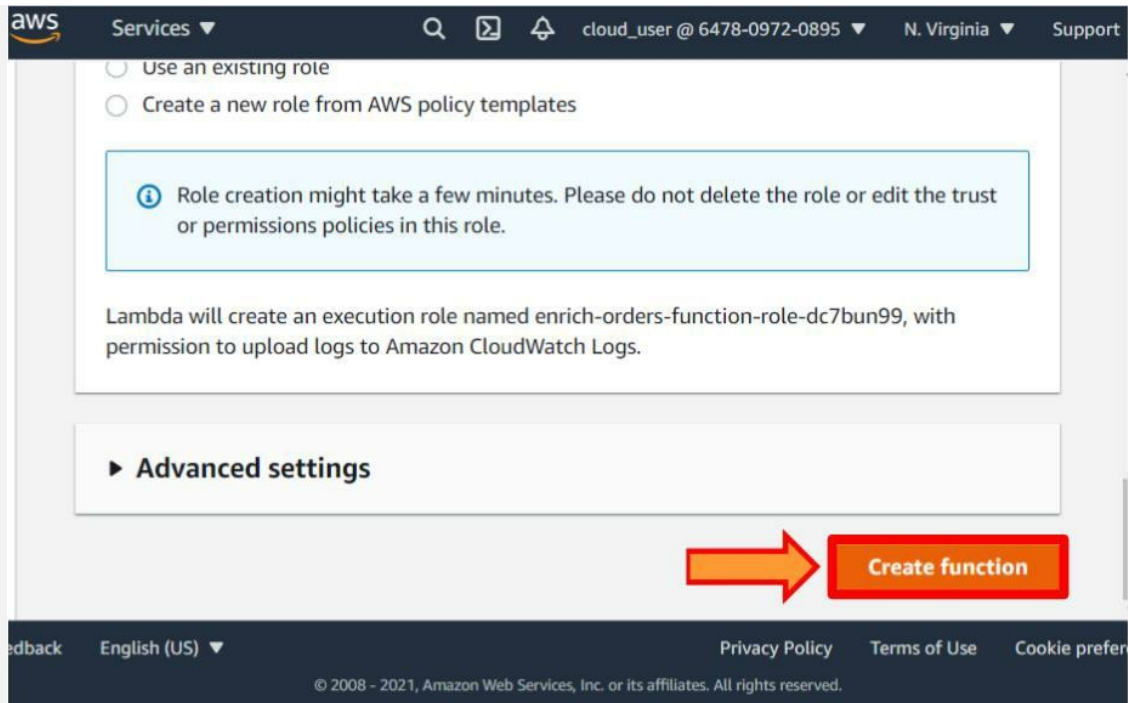
6. Click on the **Change default execution role**.



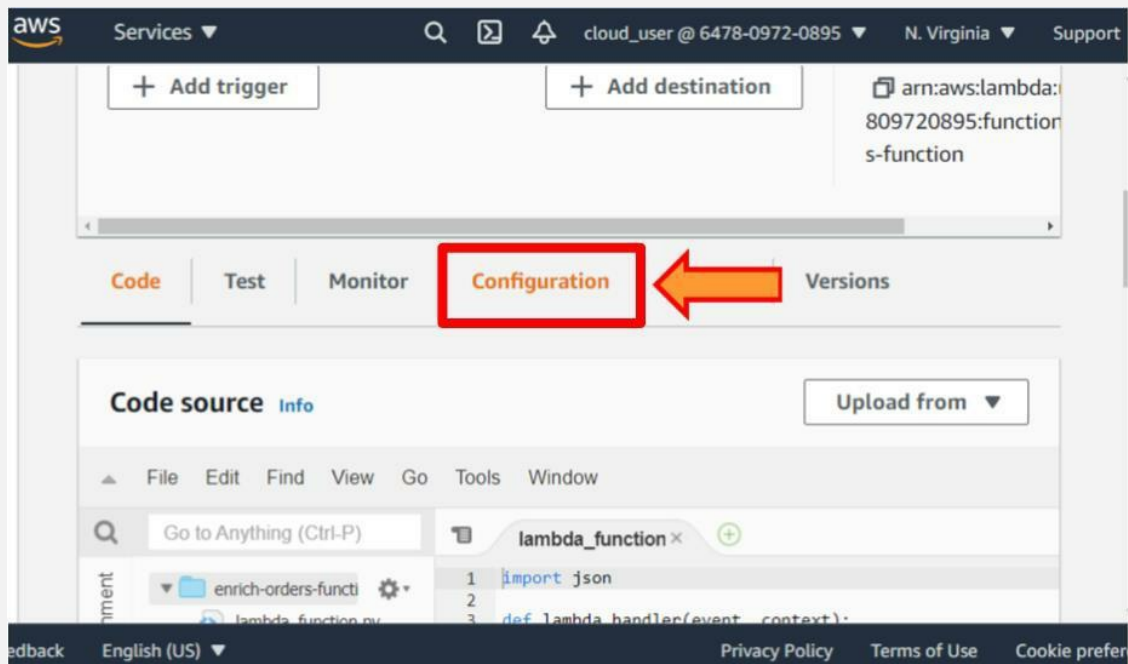
7. Select the **Create a new role with basic Lambda permissions**.



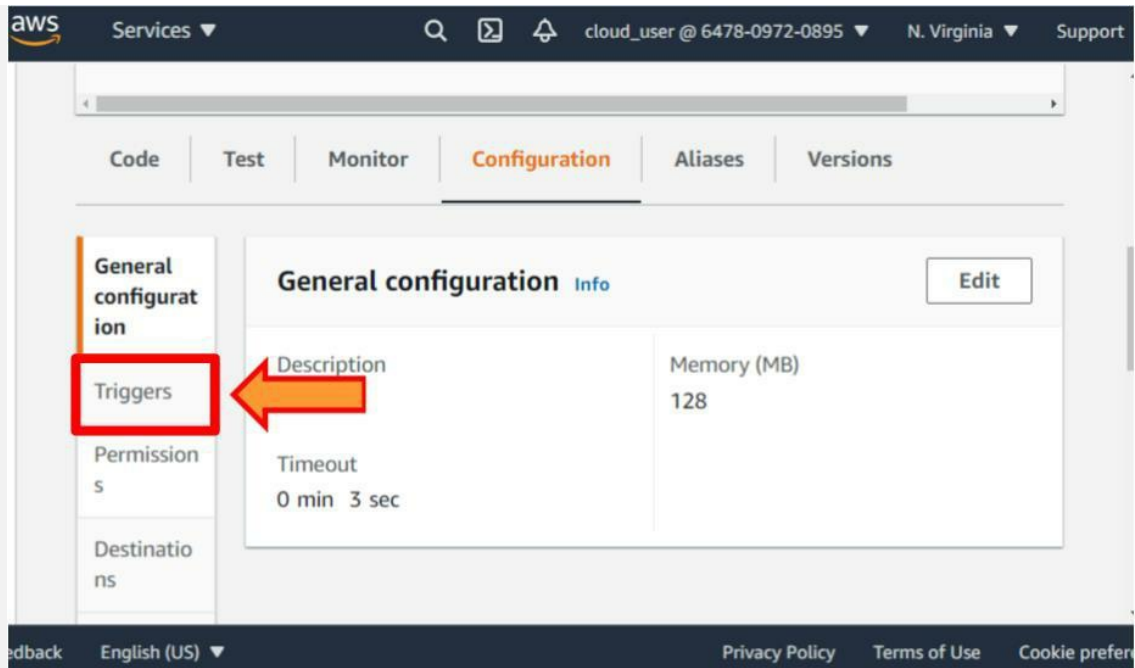
8. Click on the **Create function** button.



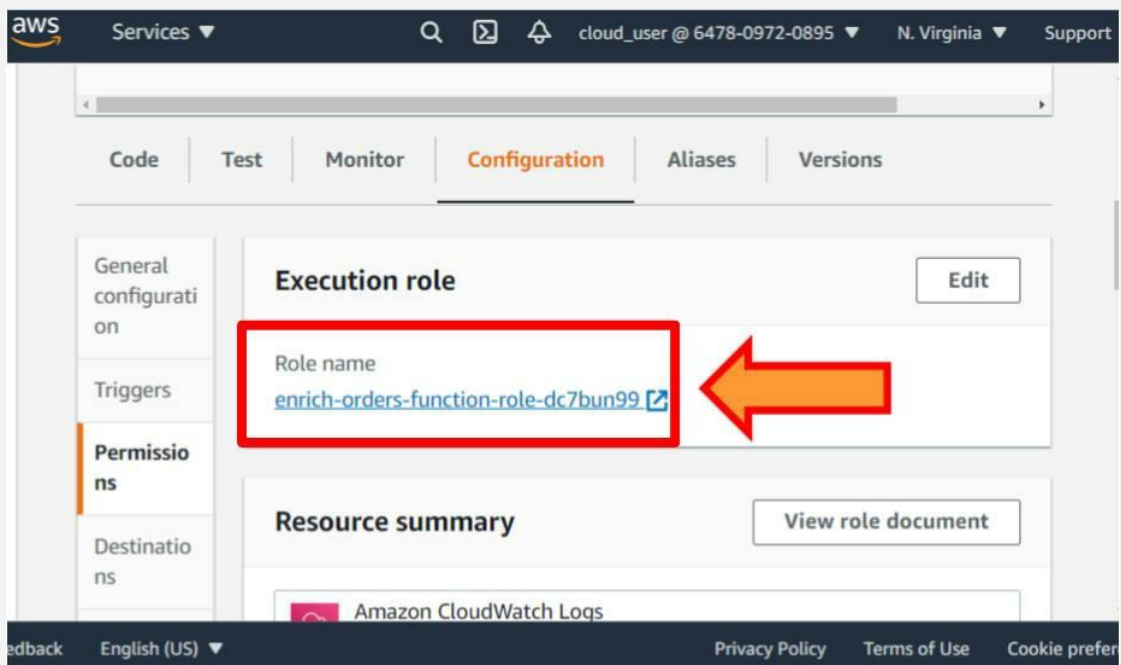
9. Click on the **Configuration** tab.



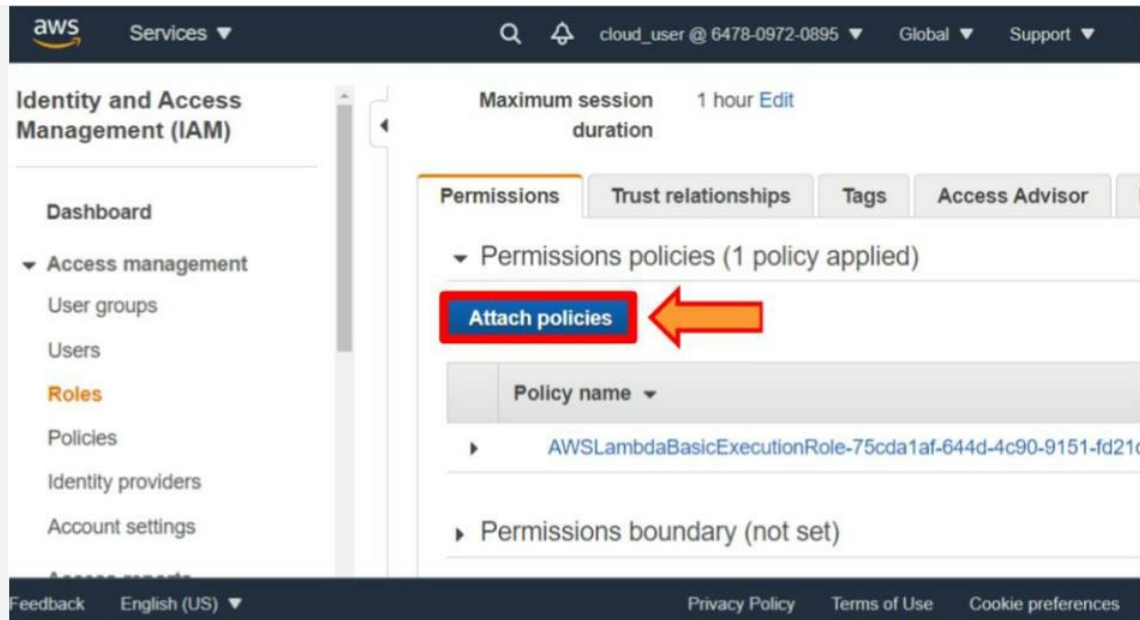
10. Click on the **Triggers**.



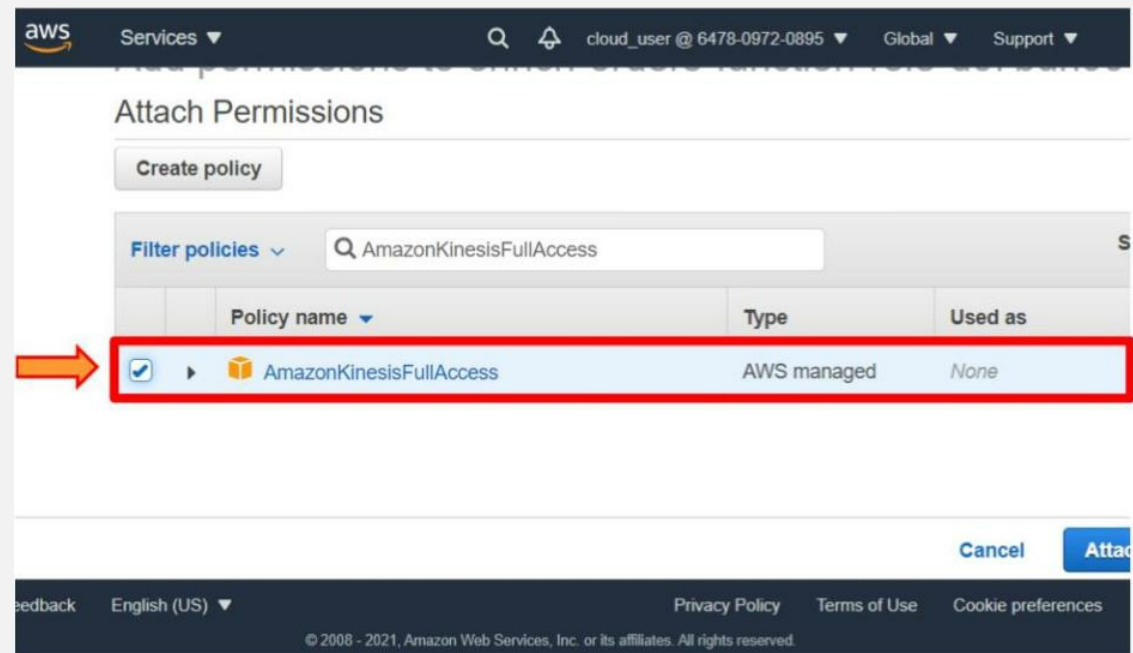
11. Click on the **Role name URL**.



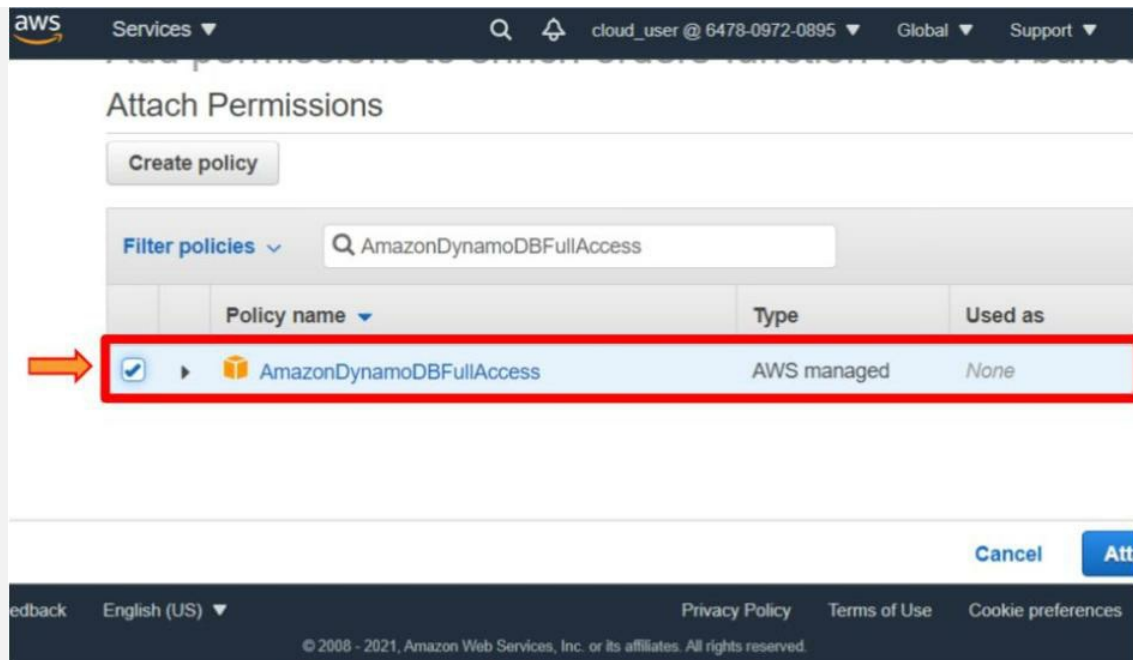
12. Click on the **Attach policies** button.



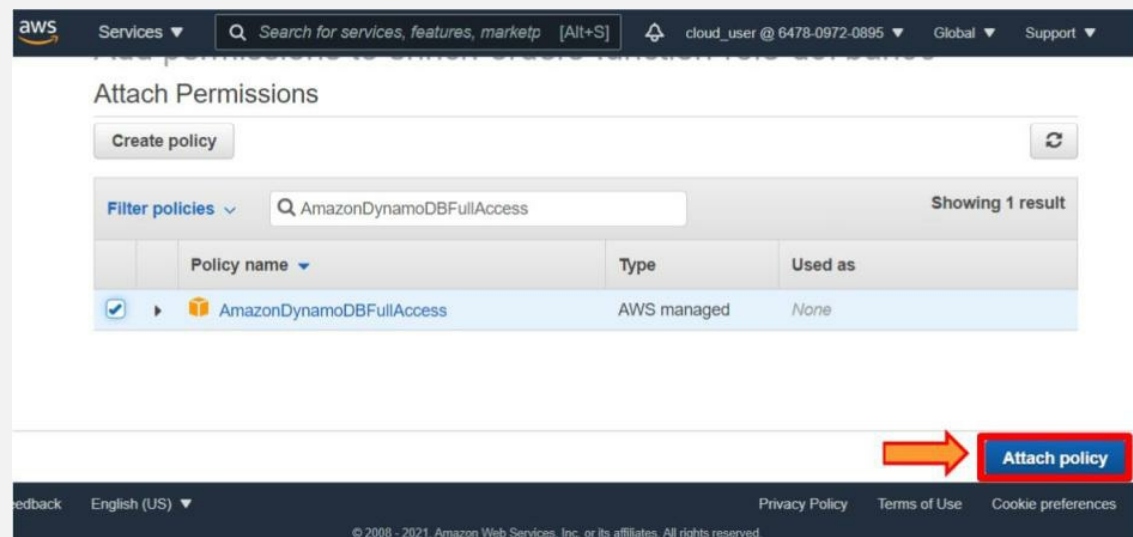
13. Select the **AmazonKinesisFullAccess**.



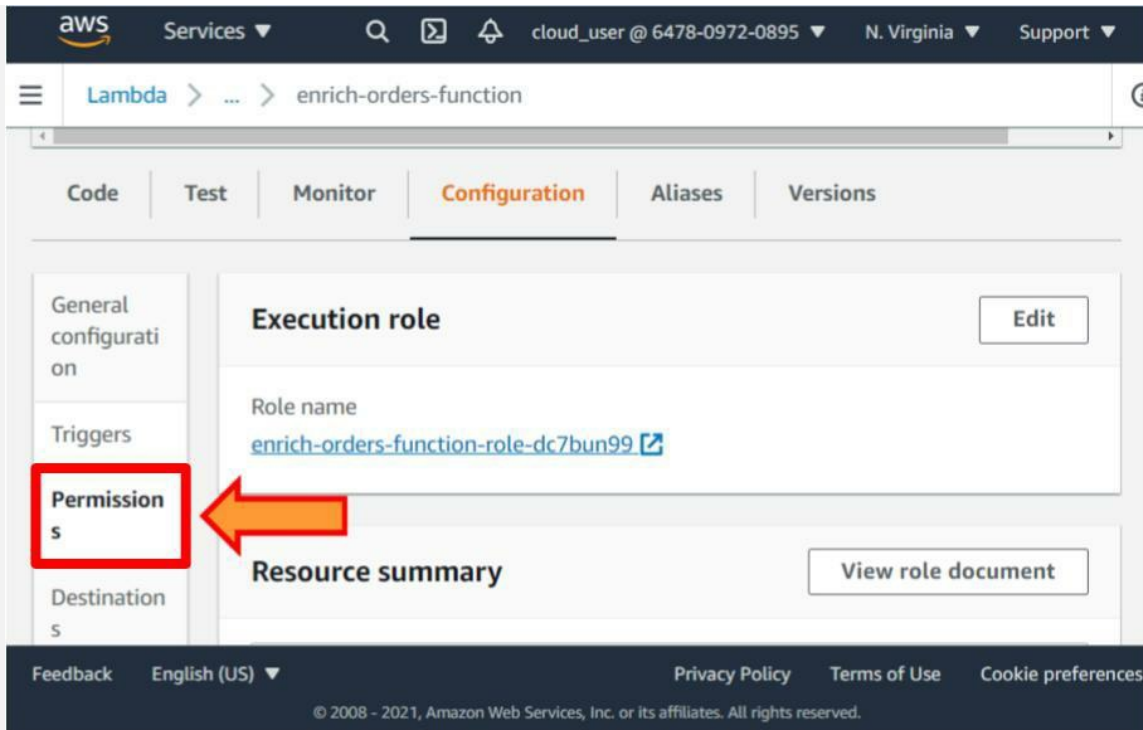
14. Select the **AmazonDynamoDBFullAccess**.



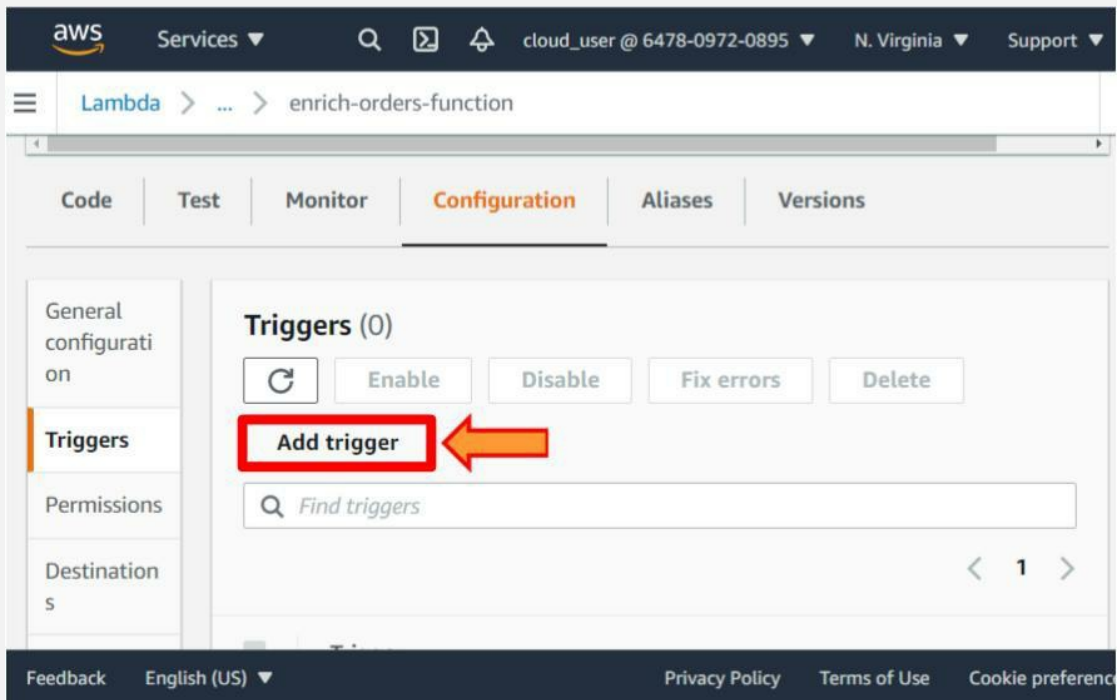
15. Click on the **Attach policy** button.



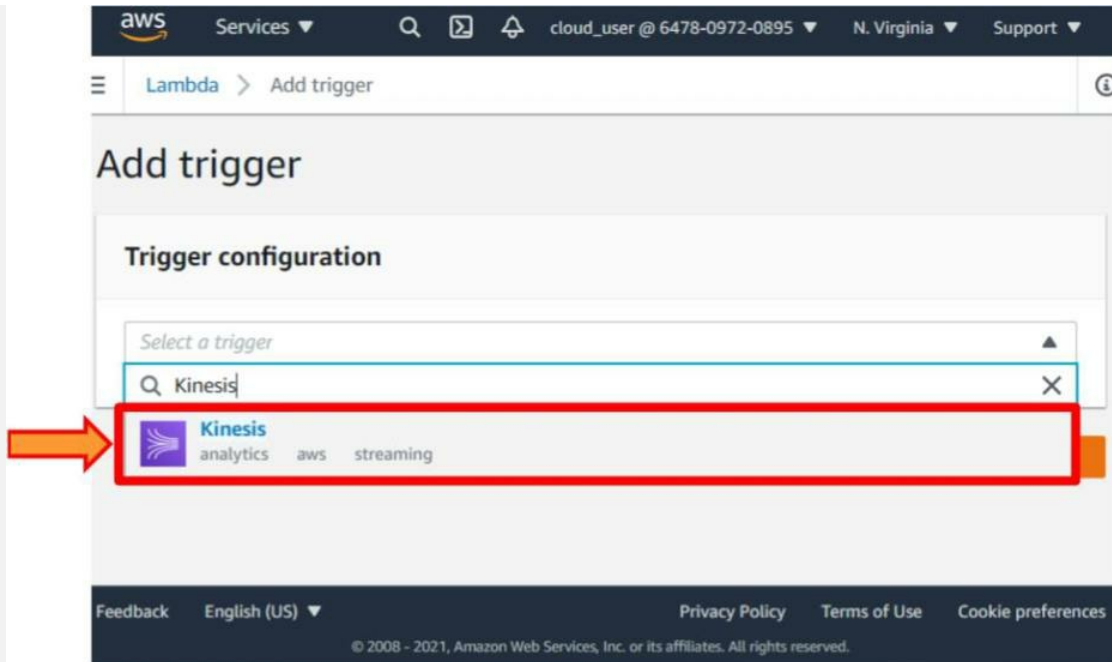
16. Go back to the Lambda function dashboard, and click on the **Permissions**.



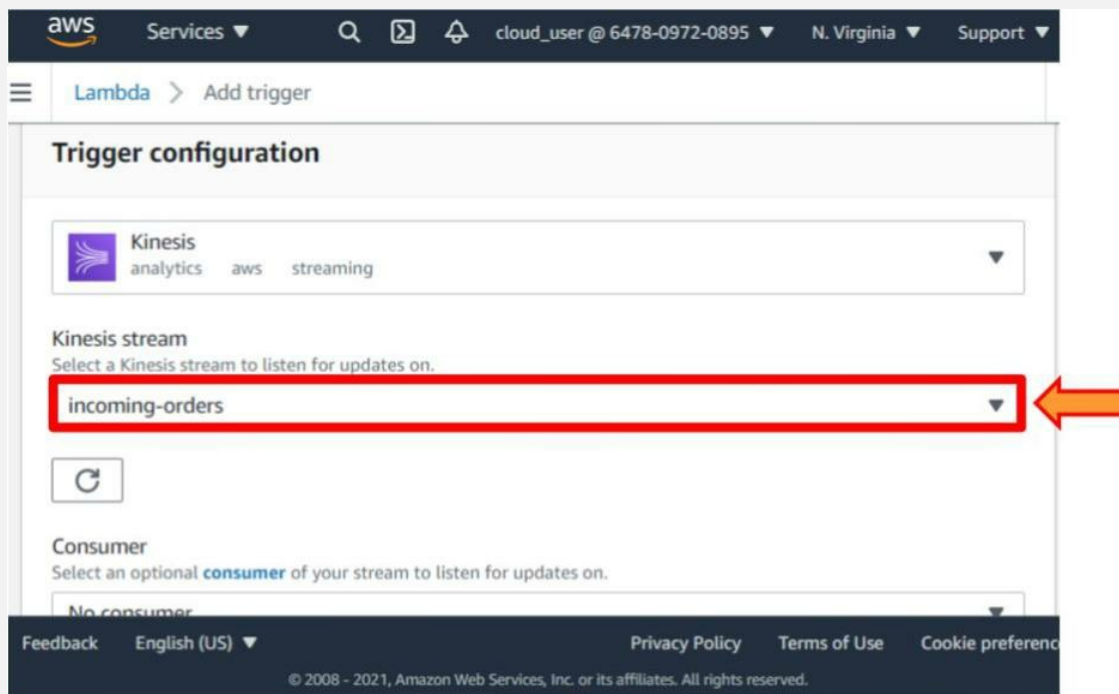
17. Click on the **Add trigger** button.



18. Select a trigger **Kinesis**.



19. Select the **incoming-orders** Kinesis stream.



20. Enter the batch size as 10.

aws Services cloud\_user @ 6478-0972-0895 N. Virginia Support

Lambda > Add trigger

No consumer

Batch size  
The largest number of records that will be read from your stream at once.

10

Batch window - optional  
The maximum amount of time to gather records before invoking the function, in seconds.

Starting position  
The position in the stream to start reading from. For more information, see [ShardIteratorType](#) in the Amazon Kinesis API Reference.

Feedback English (US) Privacy Policy Terms of Use Cookie preference

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

21. Click on the **Add** button.

aws Services cloud\_user @ 6478-0972-0895 N. Virginia Support

Lambda > Add trigger

Starting position  
The position in the stream to start reading from. For more information, see [ShardIteratorType](#) in the Amazon Kinesis API Reference.

Latest

Additional settings

In order to read from the Kinesis trigger, your execution role must have proper permissions.

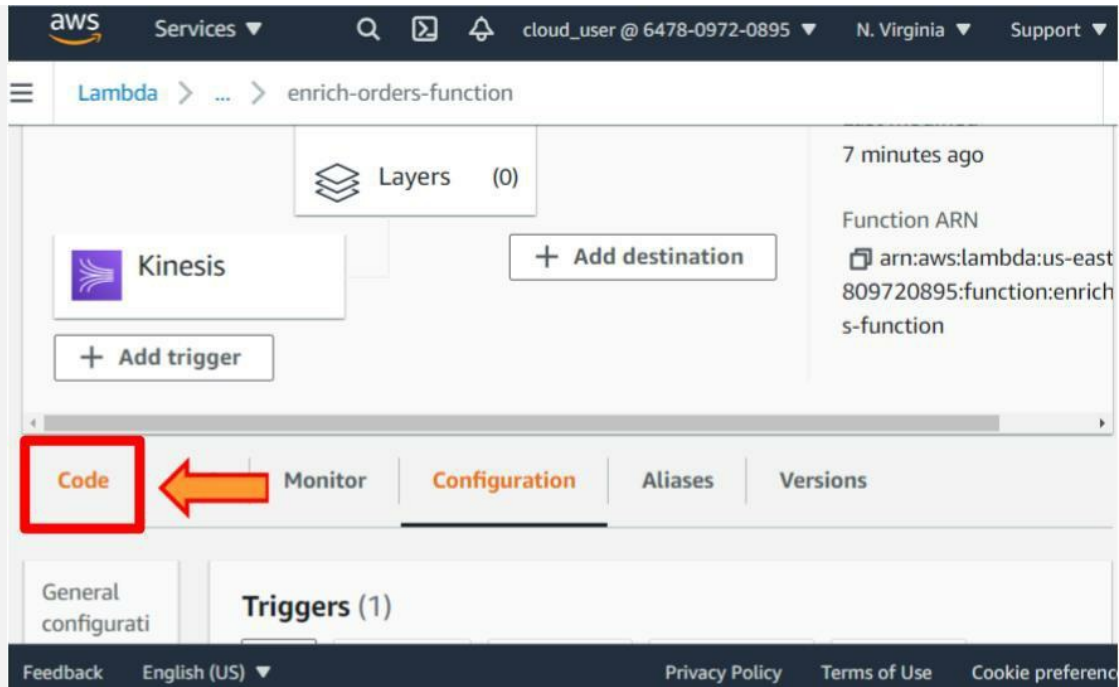
☒ Enable trigger  
Enable the trigger now, or create it in a disabled state for testing (recommended).

Add

Feedback English (US) Privacy Policy Terms of Use Cookie preference

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

22. Click on the **Code** tab.



23. Copy the code from the following provided link: <https://das-c01-data-analytics-specialty.s3.amazonaws.com/Labs/enrich-data-lambda-function.py>.

```
import boto3, json, base64

TABLE_NAME = 'users-information'
OUTPUT_STREAM_NAME = '<OUTPUT_STREAM_NAME>'

def lambda_handler(event, context):

    incoming_orders_list = [] # Used to hold the incoming records in JSON format.
    user_id_set = set() # Used to hold the ids of the users who placed an order. We'll use
    this in a mapping function later.

    # Let's loop through the batch records coming in to populate lists
    for record in event['Records']:
        payload = base64.b64decode(record['kinesis']['data'])
        incoming_order = json.loads(payload)
        user_id_set.add(incoming_order['user_id'])
        incoming_orders_list.append(incoming_order)

    # Let's get the matching batch records from DynamoDB
    id_dict = get_records(user_id_set)

    # Holds the records we are going to put onto the output enriched stream
    enriched_orders_list = []
    for incoming_order in incoming_orders_list:
        enriched_record = incoming_order.copy()
        if incoming_order['user_id'] in id_dict:
            user_id = incoming_order['user_id']
            enriched_record['first_name'] = id_dict[user_id]['first_name']['S']
            enriched_record['last_name'] = id_dict[user_id]['last_name']['S']
            enriched_record['email'] = id_dict[user_id]['email']['S']
```

24. Paste the code in the Lambda function code editor.

The screenshot shows the AWS Lambda console interface. At the top, the navigation bar includes the AWS logo, 'Services', a search icon, a document icon, a notification bell, the user 'cloud\_user @ 6478-0972-0895', the region 'N. Virginia', and a 'Support' link. The breadcrumb trail is 'Lambda > ... > enrich-orders-function'. On the left, the 'Environment' pane shows a folder 'enrich-orders-functi' with a file 'lambda\_function.py'. The main editor area, titled 'lambda\_function.py', contains the following Python code:

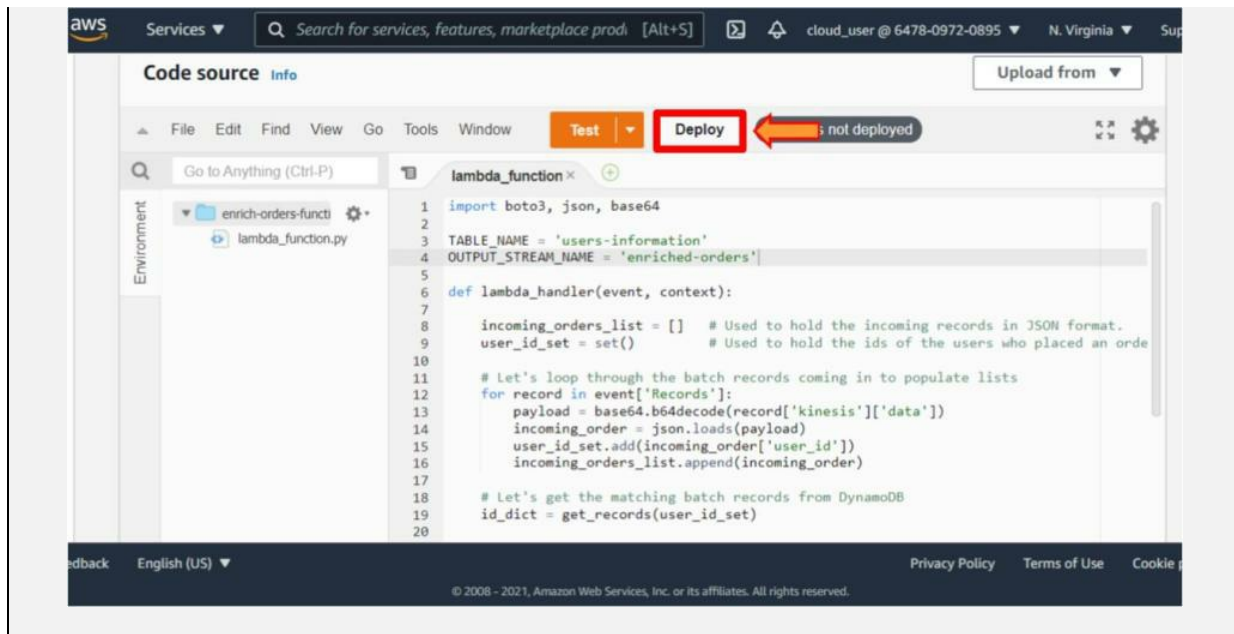
```
1 import boto3, json, base64
2
3 TABLE_NAME = 'users-information'
4 OUTPUT_STREAM_NAME = '<OUTPUT_STREAM_NAME>'
5
6 def lambda_handler(event, context):
7
8     incoming_orders_list = [] # Used to hold the incoming
9     user_id_set = set() # Used to hold the ids of t
10
11     # Let's loop through the batch records coming in to pop
12     for record in event['Records']:
13         payload = base64.b64decode(record['kinesis']['data']
14         incoming_order = json.loads(payload)
15         user_id_set.add(incoming_order['user_id'])
16         incoming_orders_list.append(incoming_order)
17
18     # Let's get the matching batch records from DynamoDB
19     id_dict = get_records(user_id_set)
20
```

At the bottom of the console, there are links for 'Feedback', 'English (US)', 'Privacy Policy', 'Terms of Use', and 'Cookie preference', along with a copyright notice: '© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.'

25. Change the <OUTPUT\_STREAM\_NAME> to **enriched-orders**.

This screenshot is similar to the previous one, but with a modification. The line `OUTPUT_STREAM_NAME = '<OUTPUT_STREAM_NAME>'` on line 4 is now `OUTPUT_STREAM_NAME = 'enriched-orders'`. This line is highlighted with a red rectangular box. A red arrow points from the 'Environment' pane, specifically from the 'lambda\_function.py' file, towards this line in the code editor. The rest of the interface, including the navigation bar, breadcrumb trail, and footer, remains the same as in the previous screenshot.

26. Click on the **Deploy** button to save the changes.



## Step 3: Start Streaming Data

1. We are using the Kinesis Live web app to stream the live data.

Kinesis Stream Name

AWS Region

Access Key

Secret Access Key

Start Streaming Data

2. Fill in the streaming details.
3. In the Kinesis Stream Name field, enter **incoming-orders**.
4. In the AWS Region field, **enter us-east-1**.
5. Fill in the **Access Key** and **Secret Access Key**.

incoming-orders

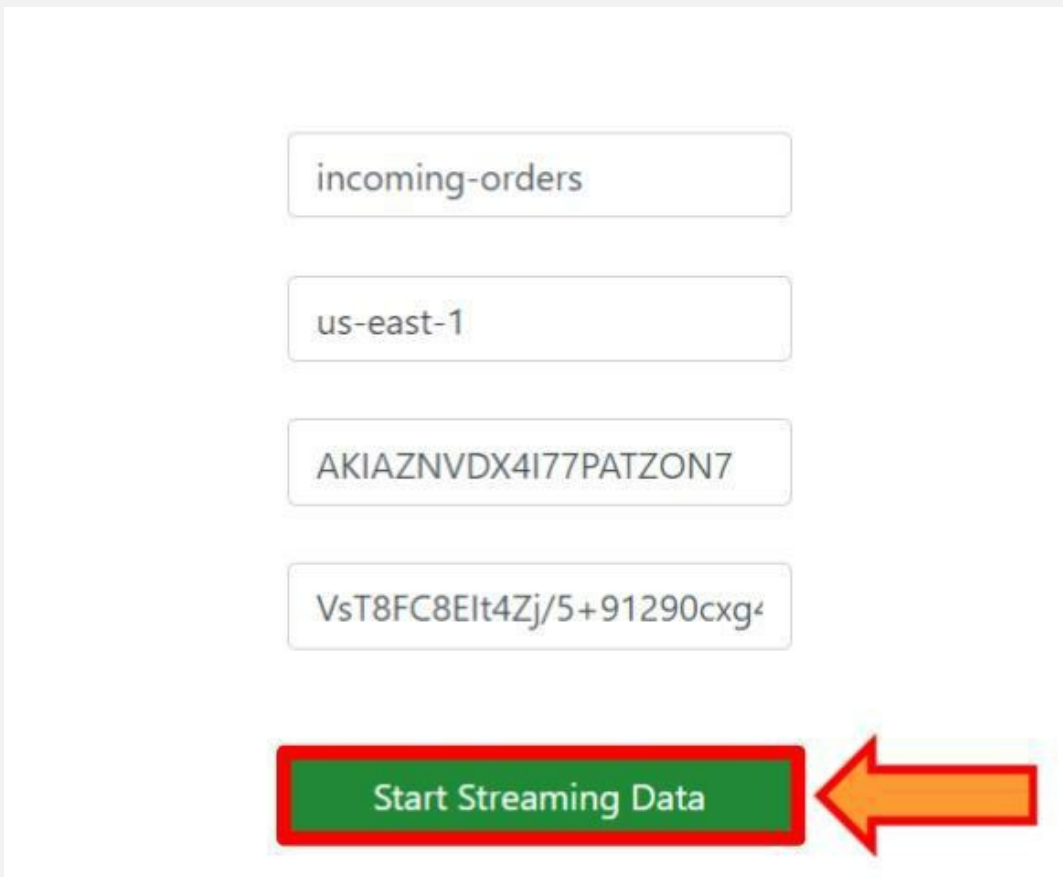
us-east-1

AKIAZNVDX4I77PATZON7

VsT8FC8Elt4Zj/5+91290cxg4

Start Streaming Data

6. Click on the **Start Streaming Data** button to stream the data.



incoming-orders

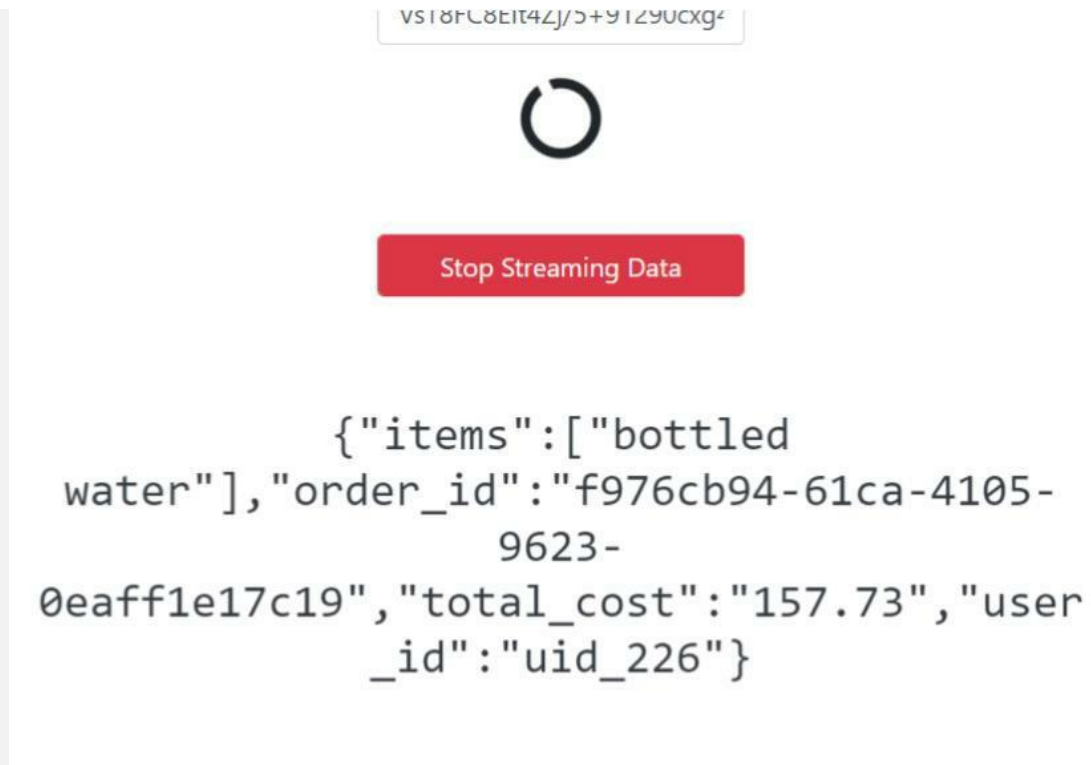
us-east-1

AKIAZNVDX4I77PATZON7

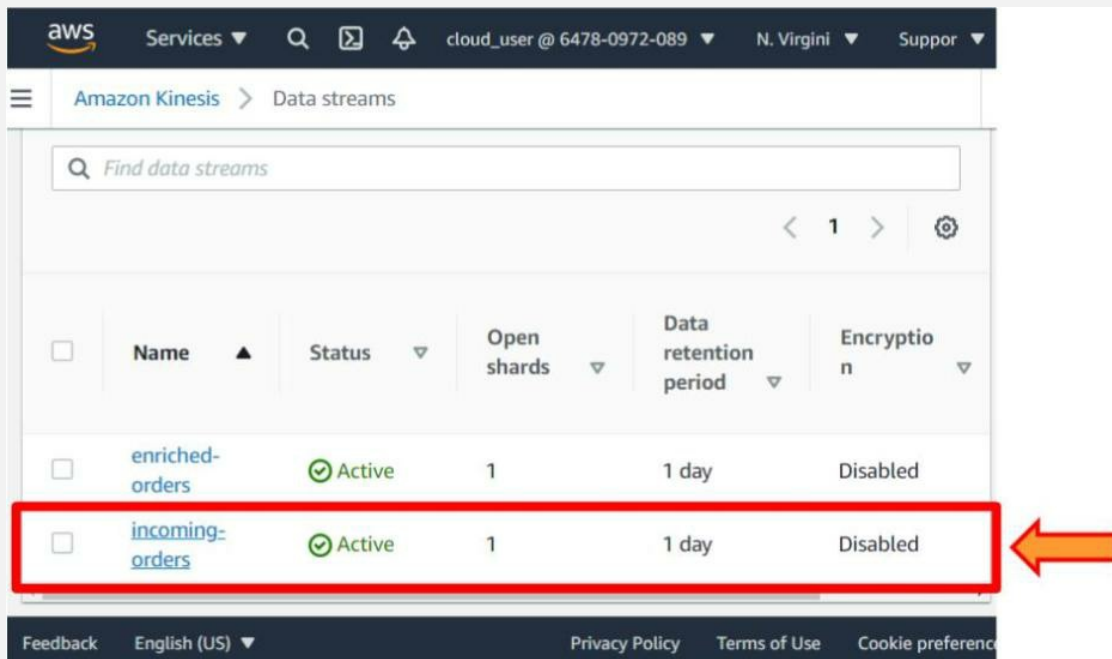
VsT8FC8Elt4Zj/5+91290cxg4

**Start Streaming Data**

7. As you can see, the data starts streaming into the Kinesis Data Stream.



8. Go back to the Kinesis Data Stream Dashboard. Click on the **incoming-orders**.



9. Click on the **Monitoring** tab.

aws Services cloud\_user @ 6478-0972-089 N. Virginia Support

Amazon Kinesis > ... > incoming-orders

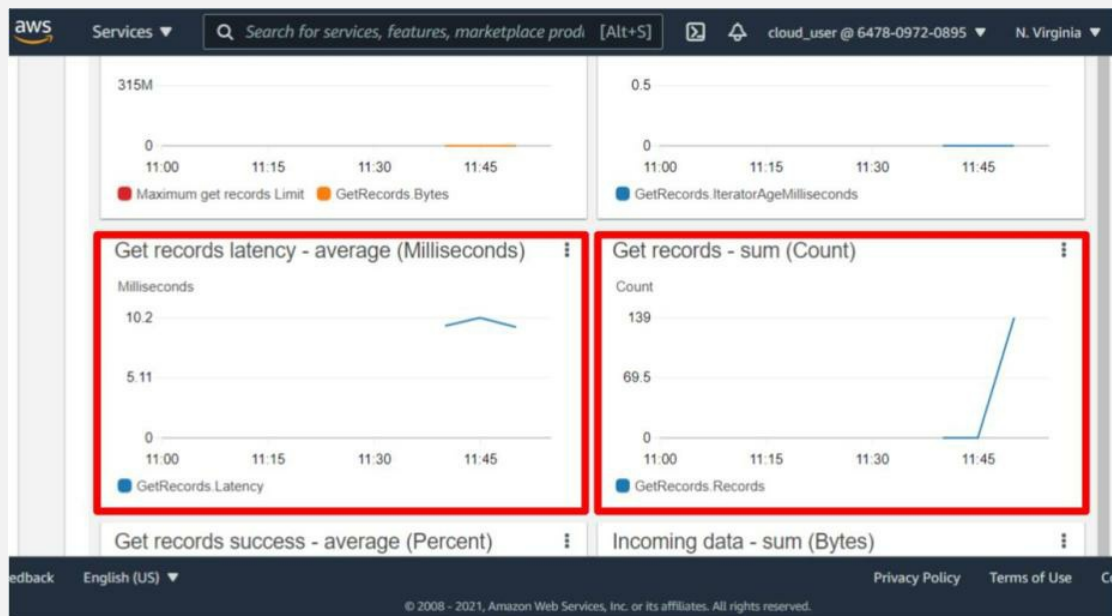
|   |   |
|---|---|
| Status<br>Active  | Data retention period<br>1 day                  |
| ARN<br>arn:aws:kinesis:us-east-1:64780972089:stream/incoming-orders | Creation time<br>November 09, 2021, 11:29 GMT+5 |

Applications **Monitoring** ation Enhanced fan-out (0)

Stream metrics [Info](#)

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

10. You will see different graphs.



11. Click on the **Data Streams**.

aws Services ▾ 🔍 📄 🔔 cloud\_user @ 6478-0972-0895 ▾ N. Virginia ▾ Support

Amazon Kinesis > **Data streams** ← incoming-orders

## incoming-orders [Info](#) Delete

### Data stream summary

|  |   |
|--|---|
| Status<br>✔ Active   | Data retention period<br>1 day                  |
| ARN<br>arn:aws:kinesis:us-east-1:6478097208:stream/incoming-orders | Creation time<br>November 09, 2021, 11:29 GMT+5 |

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

12. Click on the **enriched-orders**.

aws Services ▾ 🔍 📄 🔔 cloud\_user @ 6478-0972-0895 ▾ N. Virginia ▾ Support

Processes ▾ Actions ▾ Create data stream

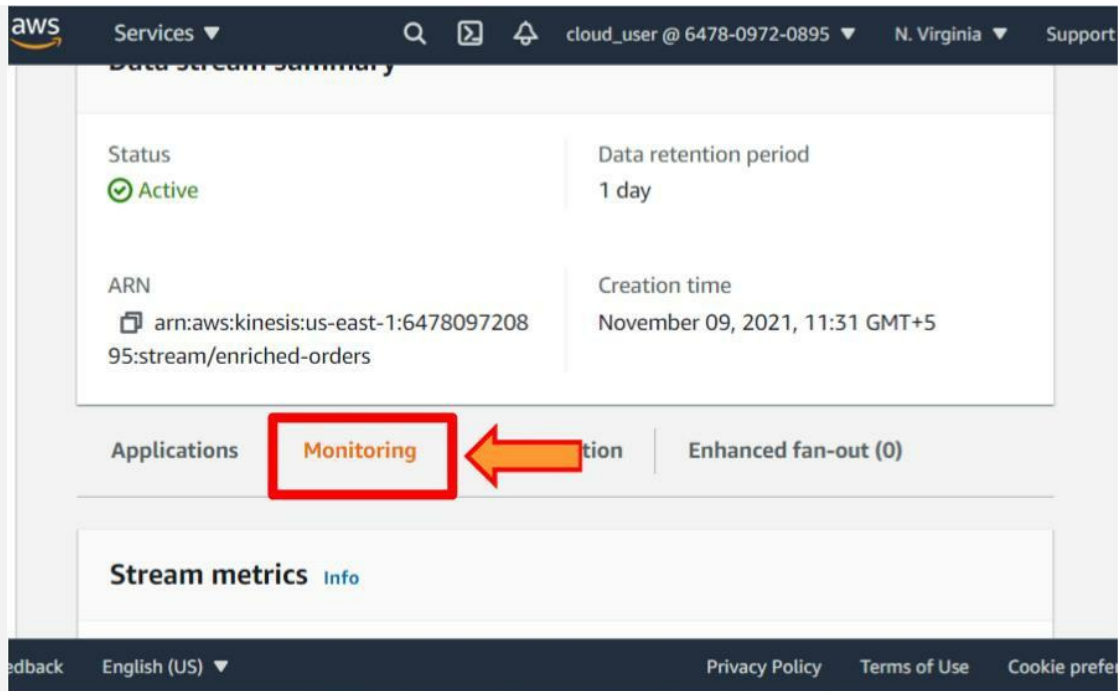
Find data streams

< 1 > ⚙️

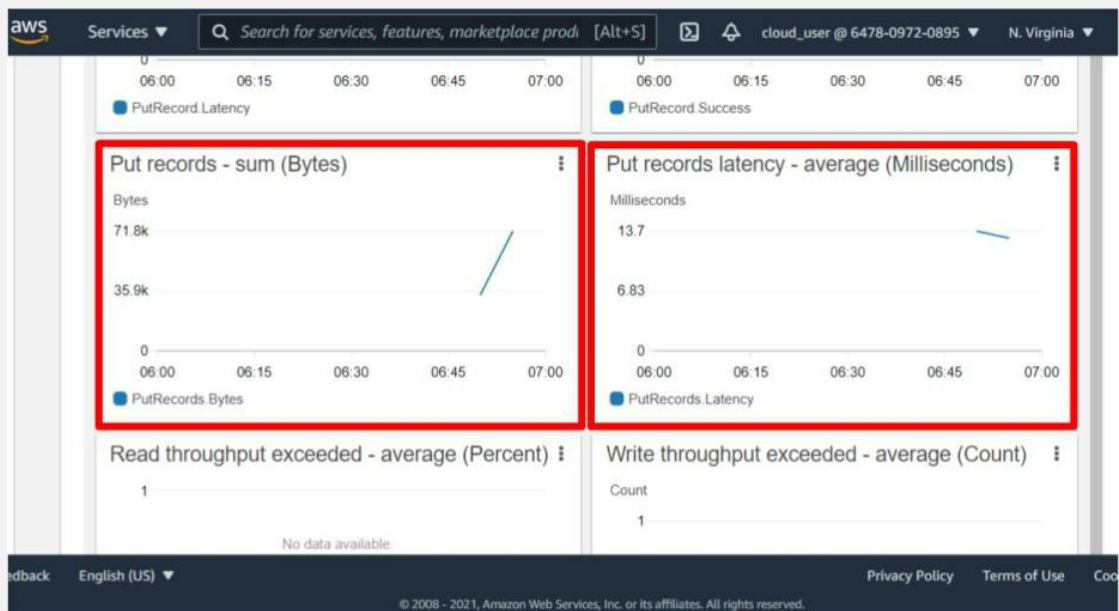
| <input type="checkbox"/> | Name ▲                          | Status ▾ | Open shards ▾ | Data retention period ▾ | Encryption |
|--------------------------|---------------------------------|----------|---------------|-------------------------|------------|
| <input type="checkbox"/> | <a href="#">enriched-orders</a> | ✔ Active | 1             | 1 day                   | Disabled   |
| <input type="checkbox"/> | <a href="#">incoming-orders</a> | ✔ Active | 1             | 1 day                   | Disabled   |

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

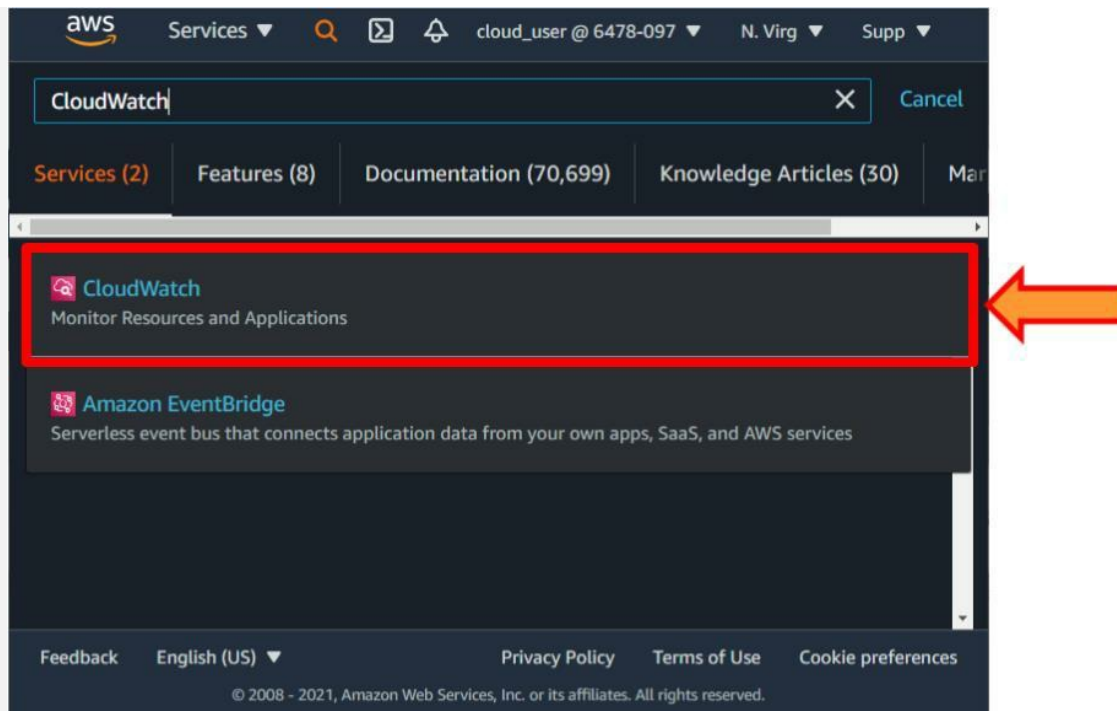
13. Click on the **Monitoring** tab.



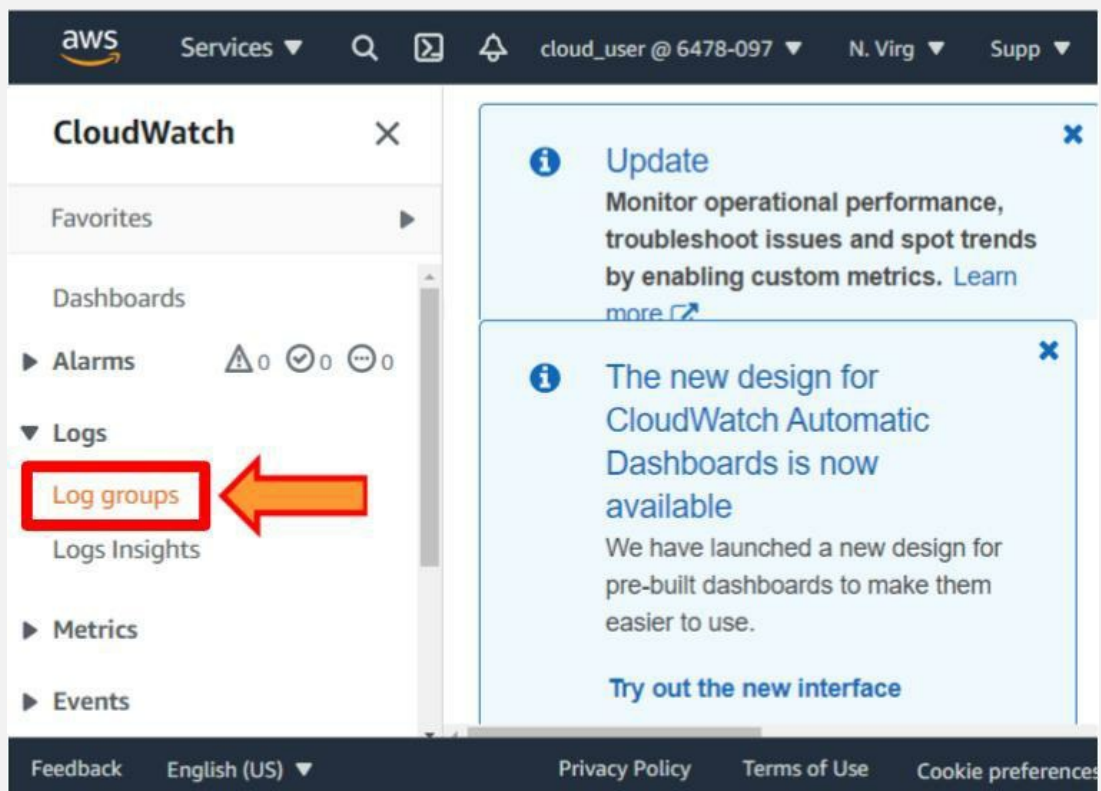
14. You will see different graphs.



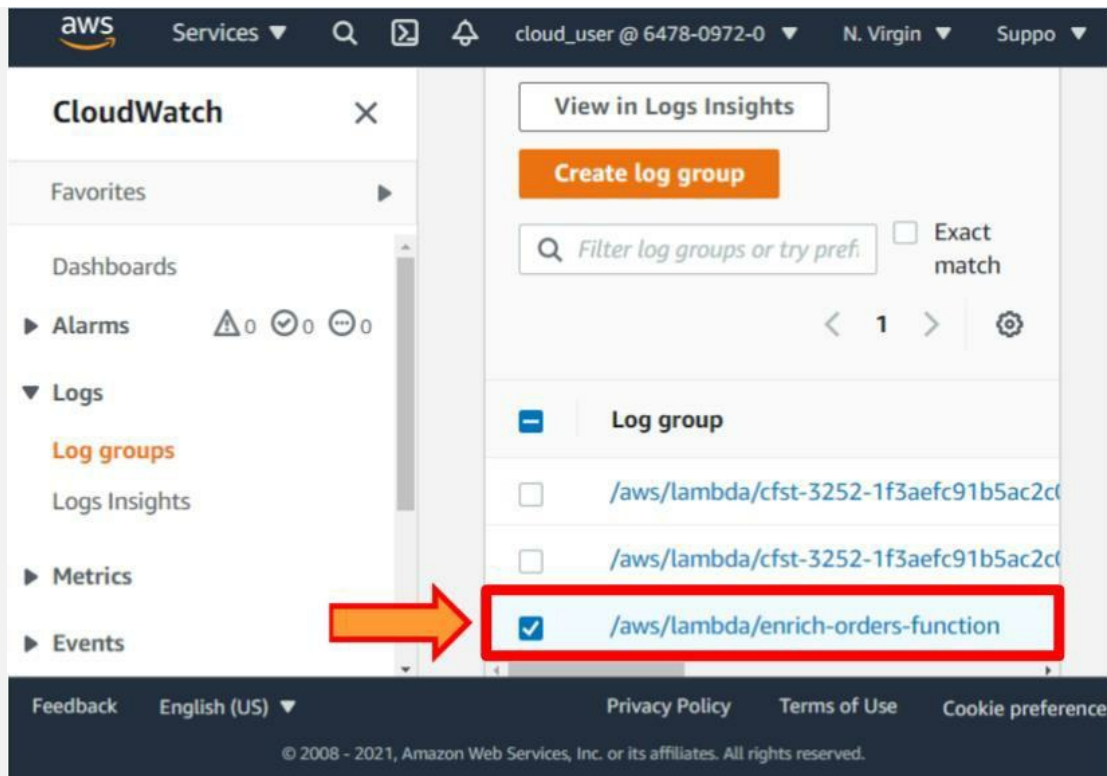
15. Open the **CloudWatch** in the new tab.



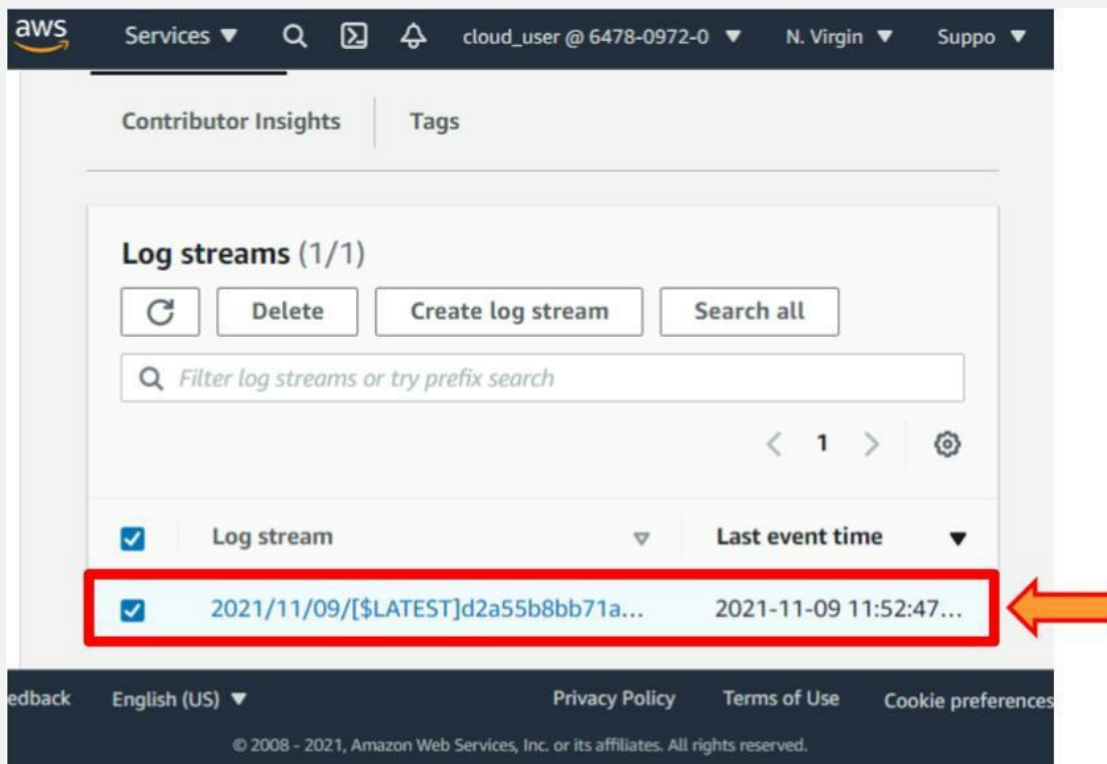
16. Click on the **Log groups**.



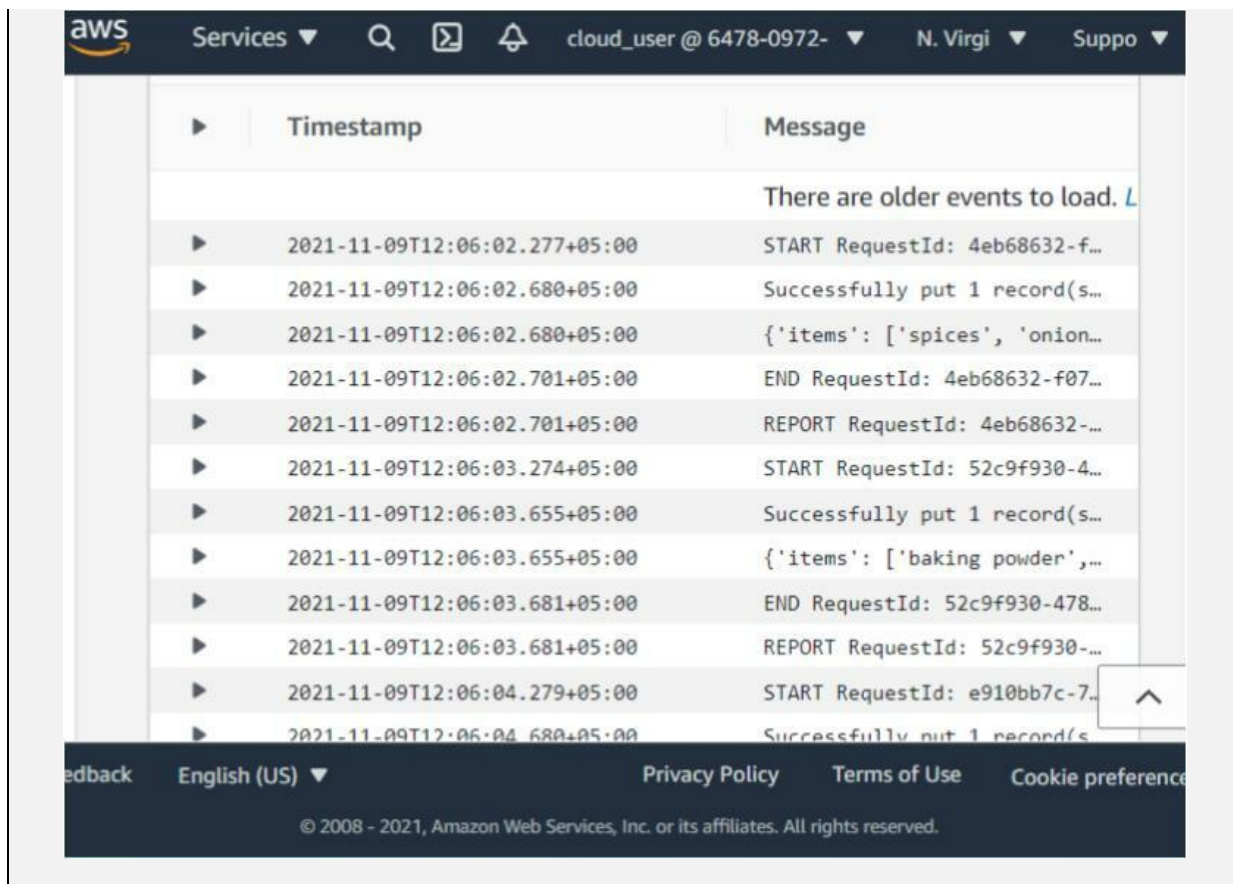
17. Click on the `/aws/lambda/enrich-orders-function`.



18. Click on the **Log Stream**.

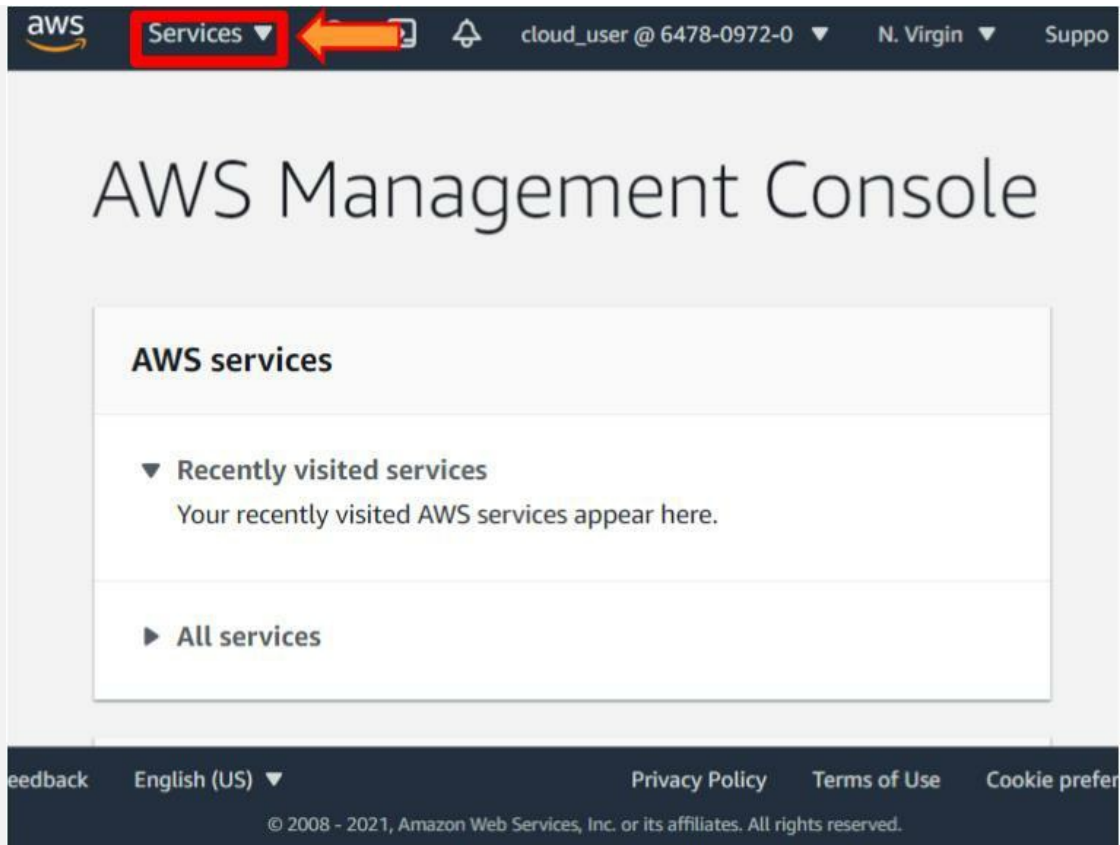


19. You will see the logs of streaming data.

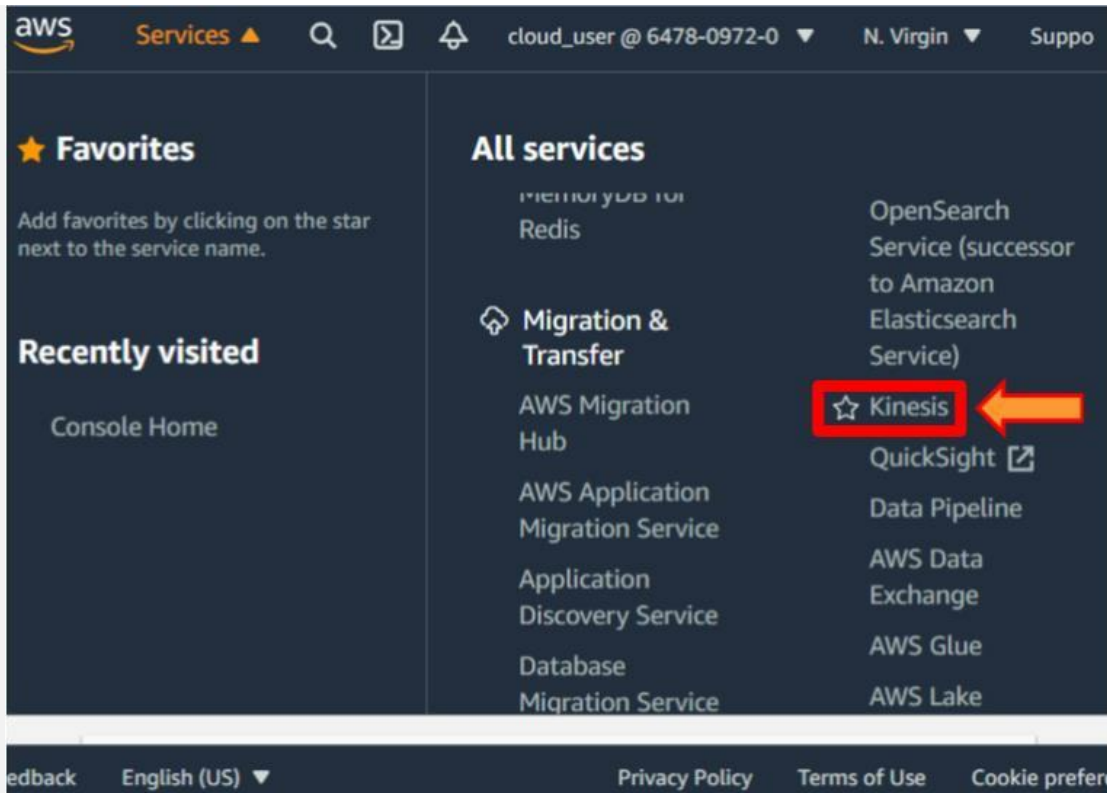


## Step 4: Filter Streaming Data with Kinesis Data Analytics

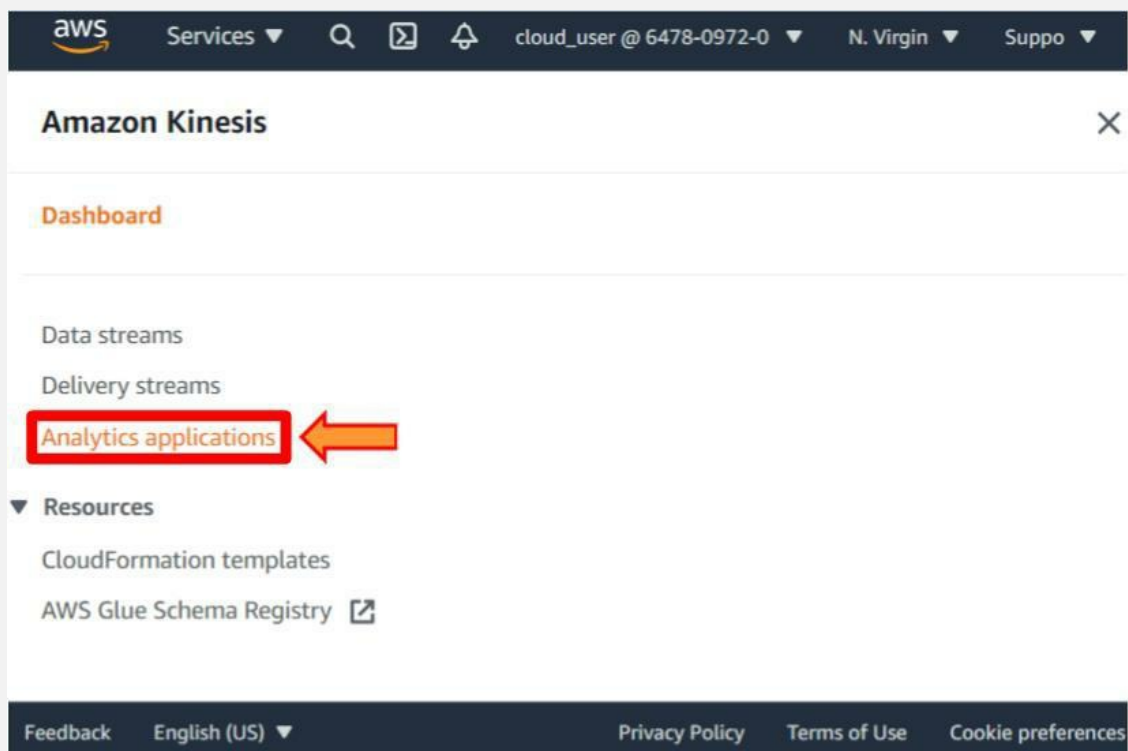
1. Click on the **Services**.



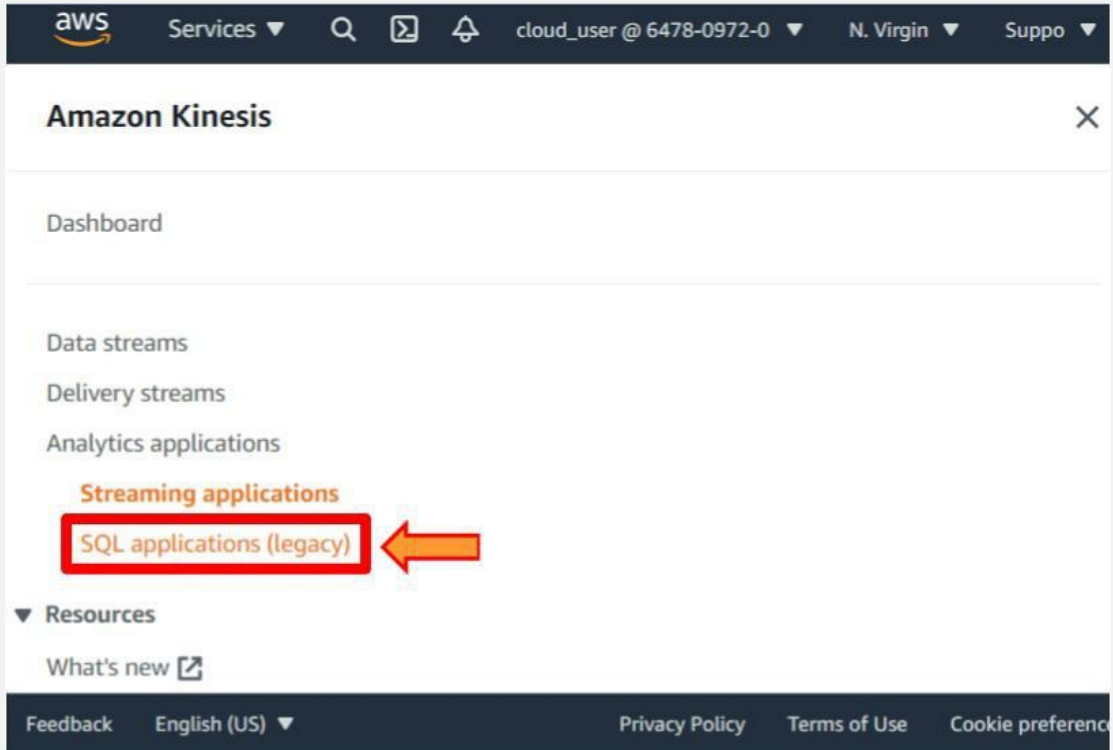
2. Select the **Kinesis** from the **Analytics**.



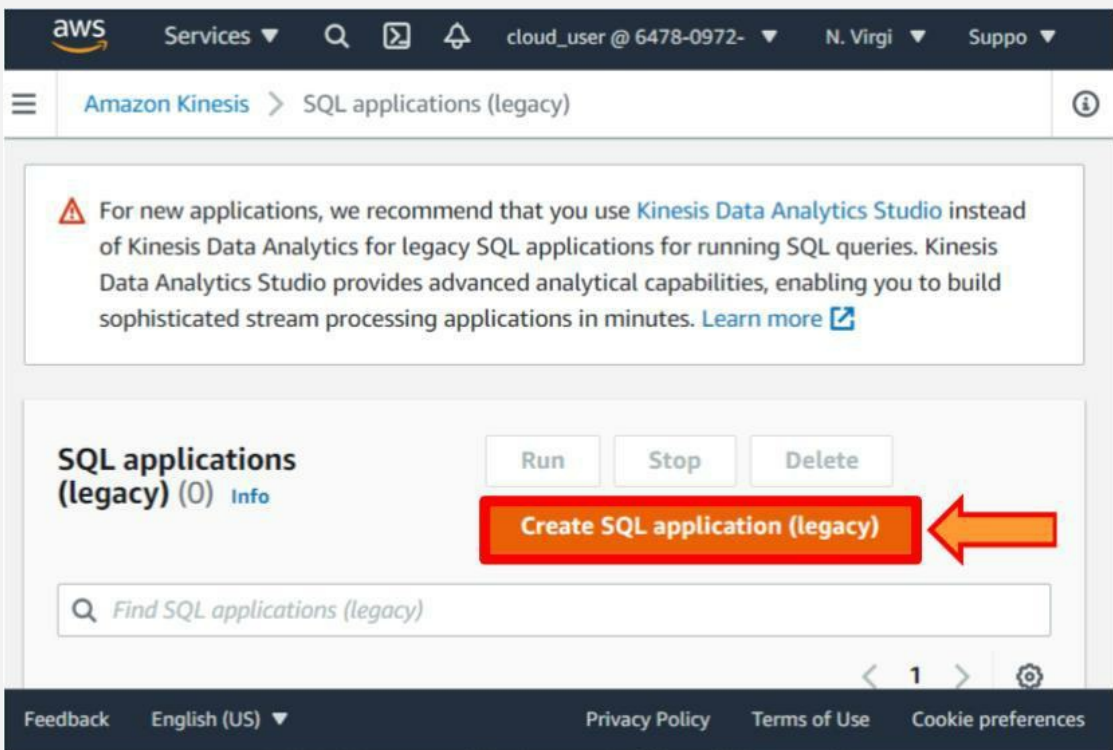
3. Click on the **Analytics Application** from the left-hand side menu.



4. Click on the **SQL Applications (legacy)**.



5. Click on the **Create SQL application (legacy)** button.



6. Give an application name **filter-top-orders**.
7. In the 'Description' box, paste the following **filter-top-orders**.

aws Services 🔻 🔍 📄 🔔 cloud\_user @ 6478-0972- 🔻 N. Virgi 🔻 Suppo 🔻

Amazon Kinesis > ... > Create legacy SQL application ⓘ

### Application configuration

Application name

filter-top-orders

Acceptable characters are uppercase and lowercase letters, numbers, underscores, hyphens, and periods.

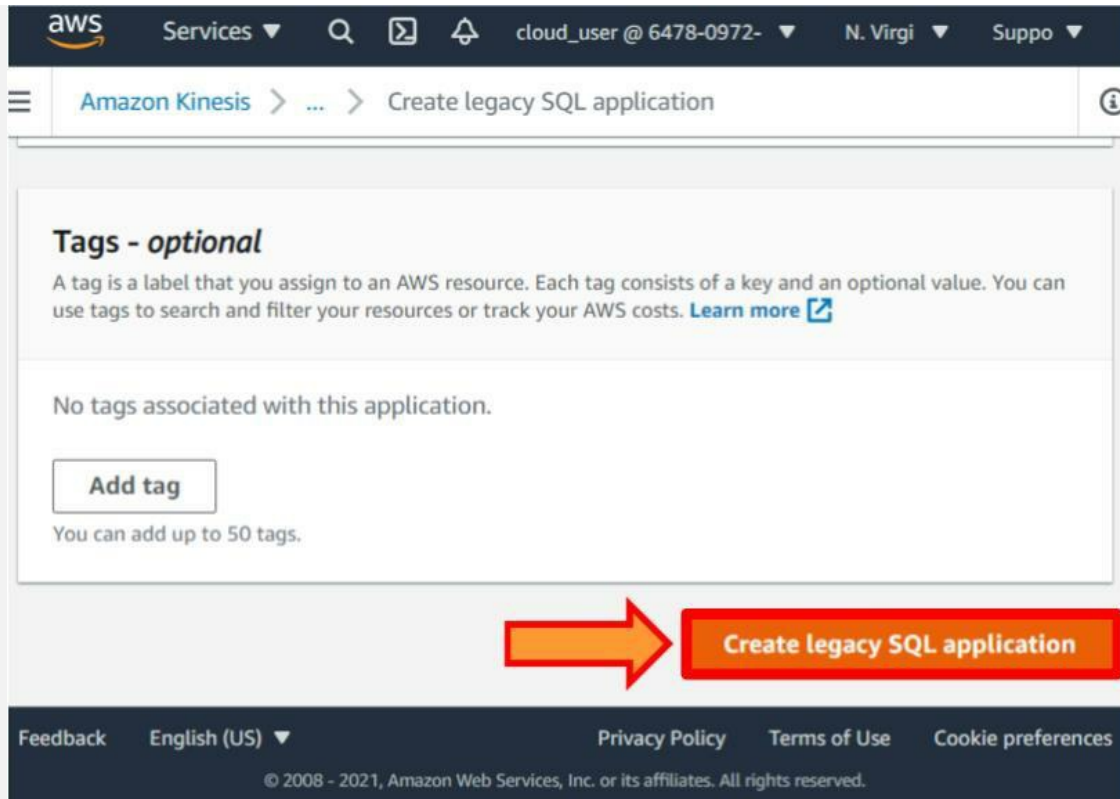
Description - *optional*

filter-top-orders

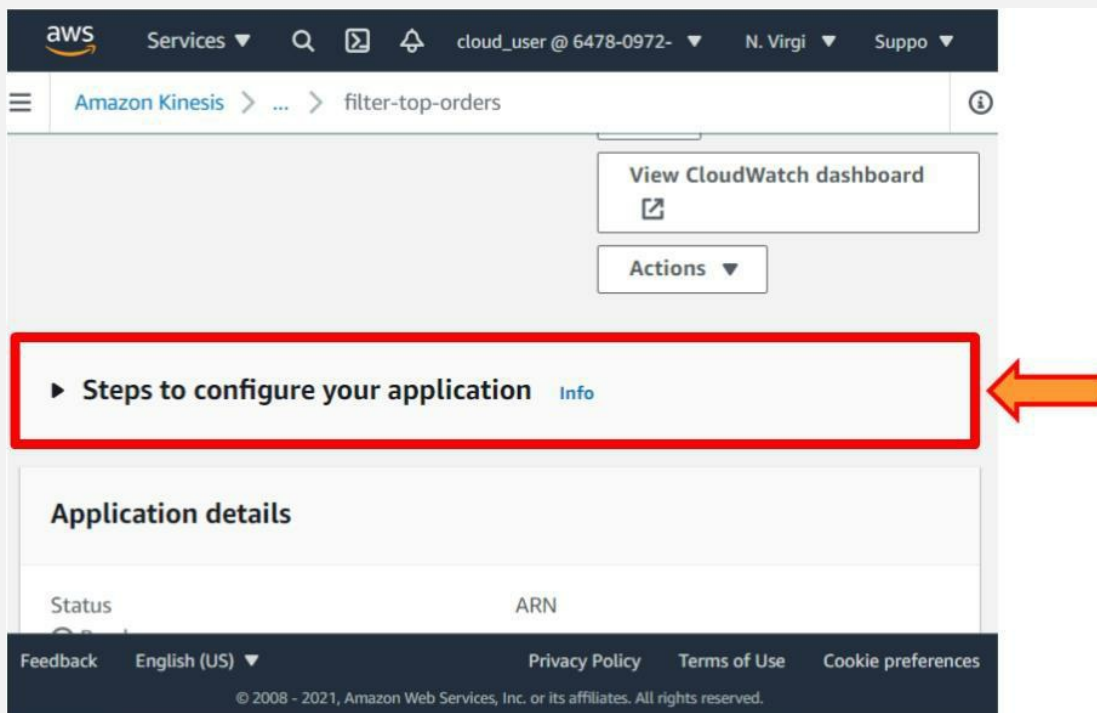
Feedback English (US) 🔻 Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

8. Click on the **Create SQL application (legacy)** button.



9. Click on the **Steps to configure your application**.



10. Click on the **Configure source stream** button.

aws Services 🔍 ⓘ cloud\_user @ 6478-0972- N. Virgi Suppo

Amazon Kinesis > ... > filter-top-orders ⓘ

### ▼ Steps to configure your application Info

Step 1

#### Configure source stream

Choose an existing Kinesis data stream or Kinesis Data Firehose delivery stream as input. Kinesis Data Analytics ingests the data, automatically recognizes standard data formats, and suggests a schema.

**Configure source stream**

Step 2

#### Run real-time analytics with your SQL code

Use the Kinesis Data Analytics SQL editor and built-in templates to write queries to process streaming data.

Configure SQL

Step 3

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

## 11. Select the **Kinesis Data Stream**.

aws Services 🔍 ⓘ cloud\_user @ 6478-0972- N. Virgi Suppo

Amazon Kinesis > ... > Configure source ⓘ

### Source Info

☒ **Kinesis data stream**  
Use a data stream for rapid and continuous data intake and aggregation.

☐ Kinesis Data Firehose delivery stream  
Use a delivery stream to deliver real-time streaming data to destinations such as Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, or Splunk.

Use demo data stream and SQL code

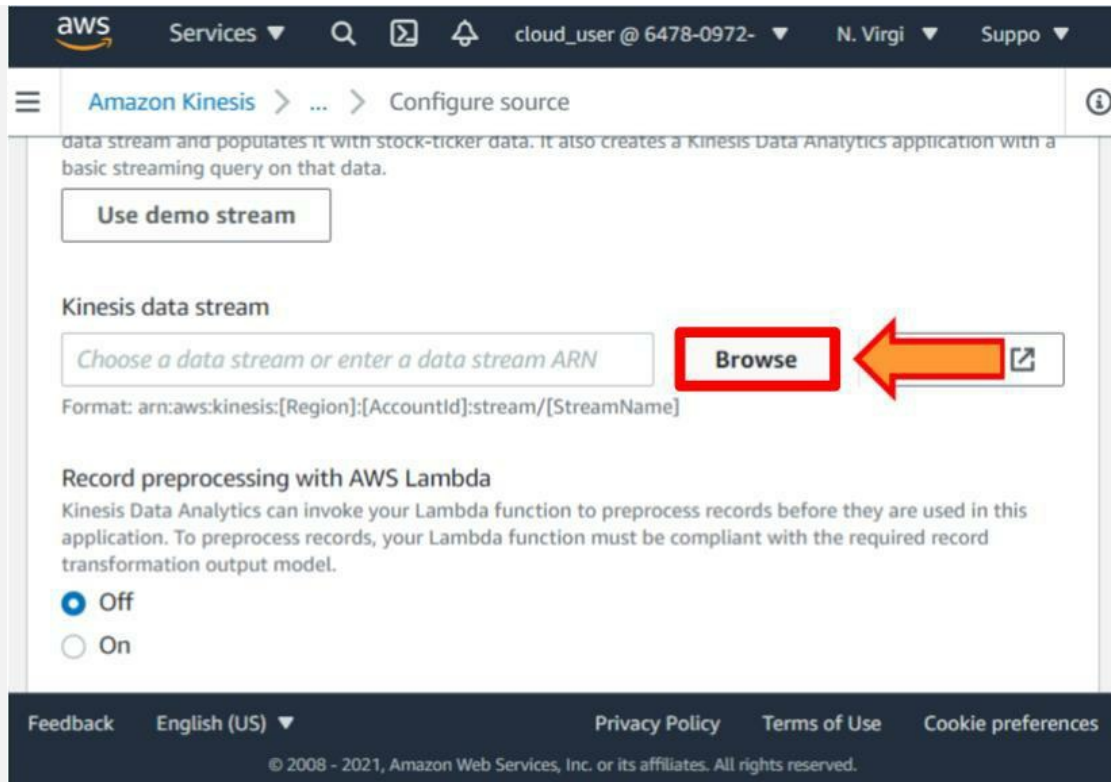
Kinesis Data Analytics creates and sets up all the resources that you need to get started. It creates a Kinesis data stream and populates it with stock-ticker data. It also creates a Kinesis Data Analytics application with a basic streaming query on that data.

Use demo stream

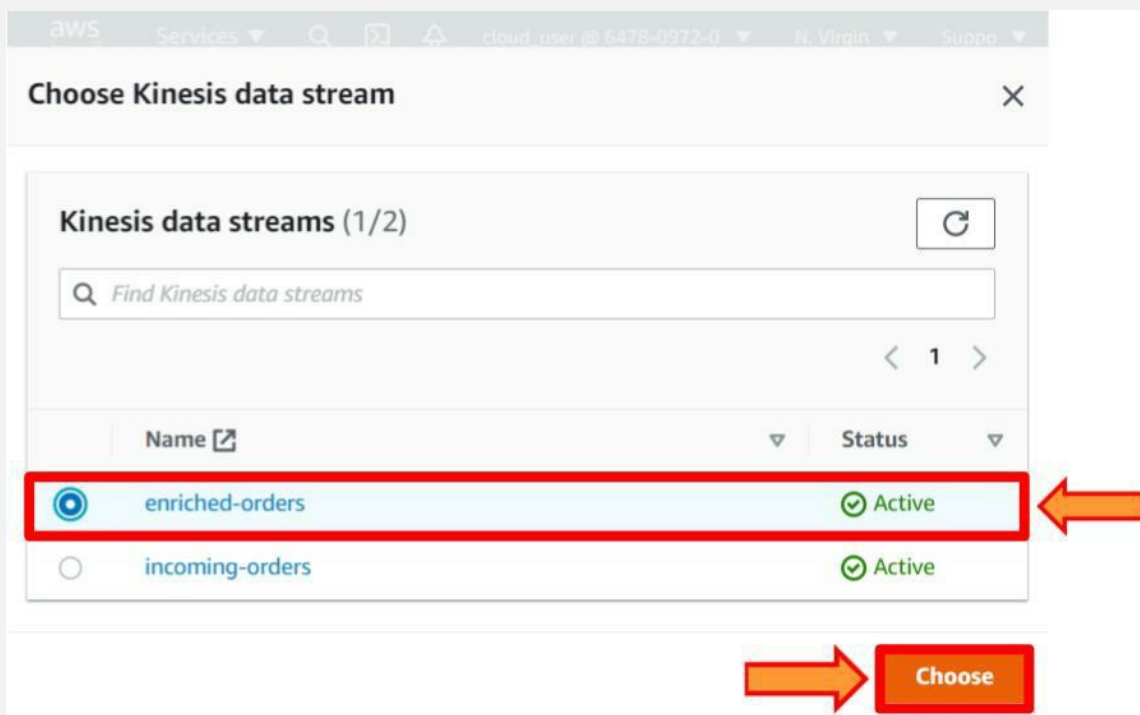
Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

## 12. Click on the **Browse** button.







13. Select the **enriched-orders**. Then, click on the **Choose** button.



14. Scroll down. Click on the **Discover Schema** button.

The screenshot shows the AWS console interface for configuring a Kinesis source. The top navigation bar includes the AWS logo, 'Services', a search icon, a document icon, a bell icon, the user 'cloud\_user @ 6478-0972-', the region 'N. Virgi', and 'Suppo'. The breadcrumb trail shows 'Amazon Kinesis > ... > Configure source'. The main content area is titled 'Schema' with an 'Info' link. Below the title, a paragraph explains that schema discovery generates a schema from source records, with column names matching the source unless they contain special characters, repeated names, or reserved keywords. An orange arrow points from this text to a red-bordered button labeled 'Discover schema'. Below this, a message box with a warning icon states: 'Further action is required to apply your changes. Each of the following actions will create or update IAM role **kinesis-analytics-filter-top-orders-us-east-1**. To generate a schema using recent records from the source, choose **Discover schema**.' The footer contains links for 'Feedback', 'English (US)', 'Privacy Policy', 'Terms of Use', and 'Cookie preferences', along with a copyright notice for 2008-2021 Amazon Web Services, Inc.

15. You will see the Raw and Formatted streaming data that were collected.

| <div>  <div> Services ▼ </div> <div>    <div>cloud_user @ 6478-0972-</div> <div>N. Virgi</div> <div>Suppo</div> </div> </div> |              |              |            |             |
|---|--------------|--------------|------------|-------------|
| <div> <div>Amazon Kinesis</div> <div>Configure source</div> </div>  |              |              |            |             |
| items   | order_id     | total_cost   | user_id    | first_name  |
| VARCHAR(128)  | VARCHAR(64)  | DECIMAL(5,2) | VARCHAR(8) | VARCHAR(16) |
| ["cleaner", ...]  | 1e5b9cae...  | 176.07       | uid_466    | Samantha    |
| ["soups", "...]   | 3d112da9...  | 186.20       | uid_121    | Thea        |
| ["yogurt", "...]  | 565e156c...  | 68.99        | uid_474    | Becky       |
| ["frozen ...]   | a4c6b79c...  | 120.26       | uid_322    | Villads     |
| ["ketchup"...   | 4870f89b...  | 157.91       | uid_127    | Owen        |
| ["meat enr...   | 330ef7330... | 108.80       | uid_130    | Billie      |

Feedback
English (US)
Privacy Policy
Terms of Use
Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

16. Click on the **Save changes** button.

aws Services 🔻 🔍 📄 🔔 cloud\_user @ 6478-0972- 🔻 N. Virgi 🔻 Suppo 🔻

Amazon Kinesis > ... > Configure source ⓘ

|                |              |        |         |         |
|----------------|--------------|--------|---------|---------|
| ["meat spr...  | aaef7330-... | 108.80 | uid_130 | Billie  |
| ["salty sna... | 998568ae...  | 139.23 | uid_399 | Gina    |
| ["ham","d...   | 11376e66...  | 46.26  | uid_396 | Sara    |
| ["kitchen t... | 9d4ce5e6...  | 147.91 | uid_403 | Brennan |
| ["beverag...   | 01c40fc3-... | 138.99 | uid_223 | Liam    |

Save changes

Feedback English (US) 🔻 Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

17. Click on the **Real-time analytics** tab.

aws Services 🔻 🔍 📄 🔔 cloud\_user @ 6478-0972- 🔻 N. Virgi 🔻 Suppo 🔻

Amazon Kinesis > ... > filter-top-orders ⓘ

Source **Real-time analytics** Tags

**Source stream** Info

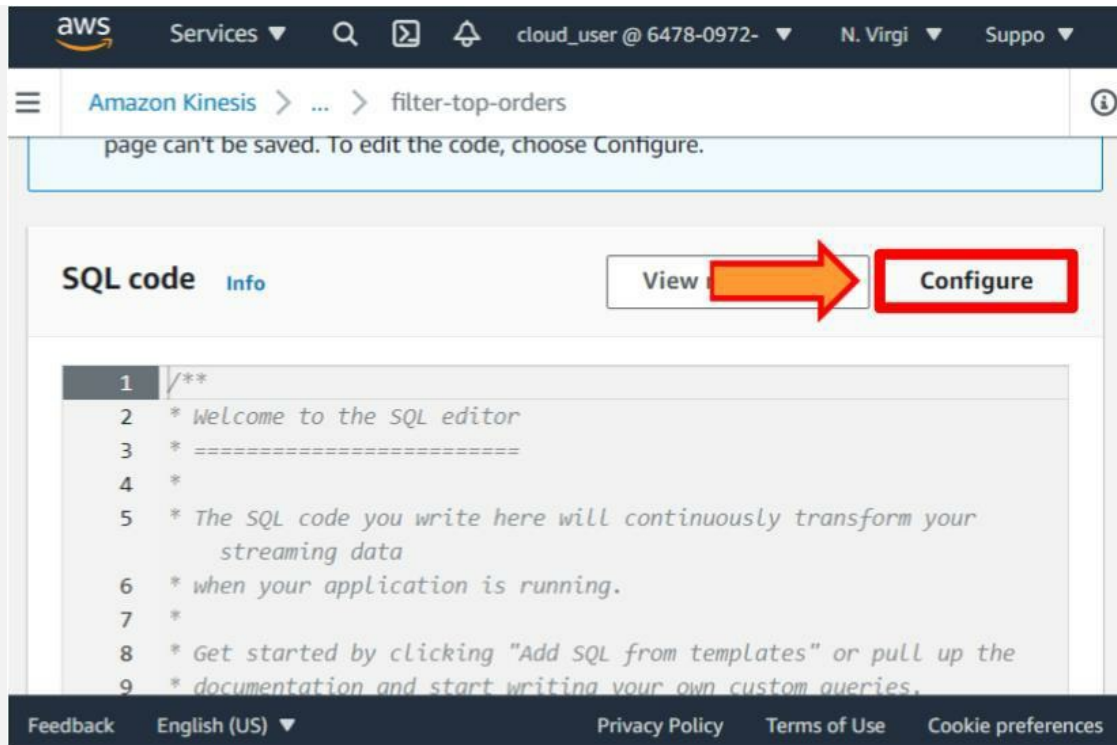
Connect to an existing Kinesis data stream or Firehose delivery stream, or easily create and connect to a new demo Kinesis data stream. Each application can connect to one streaming data source.

Configure

|                            |                   |
|----------------------------|-------------------|
| Source                     | Stream            |
| Kinesis Data Stream        | enriched-orders 🔗 |
| In-application stream name | ID                |

Feedback English (US) 🔻 Privacy Policy Terms of Use Cookie preferences

18. Then click on the **Configure** button.



19. Copy and paste the SQL queries from the below code snippet.

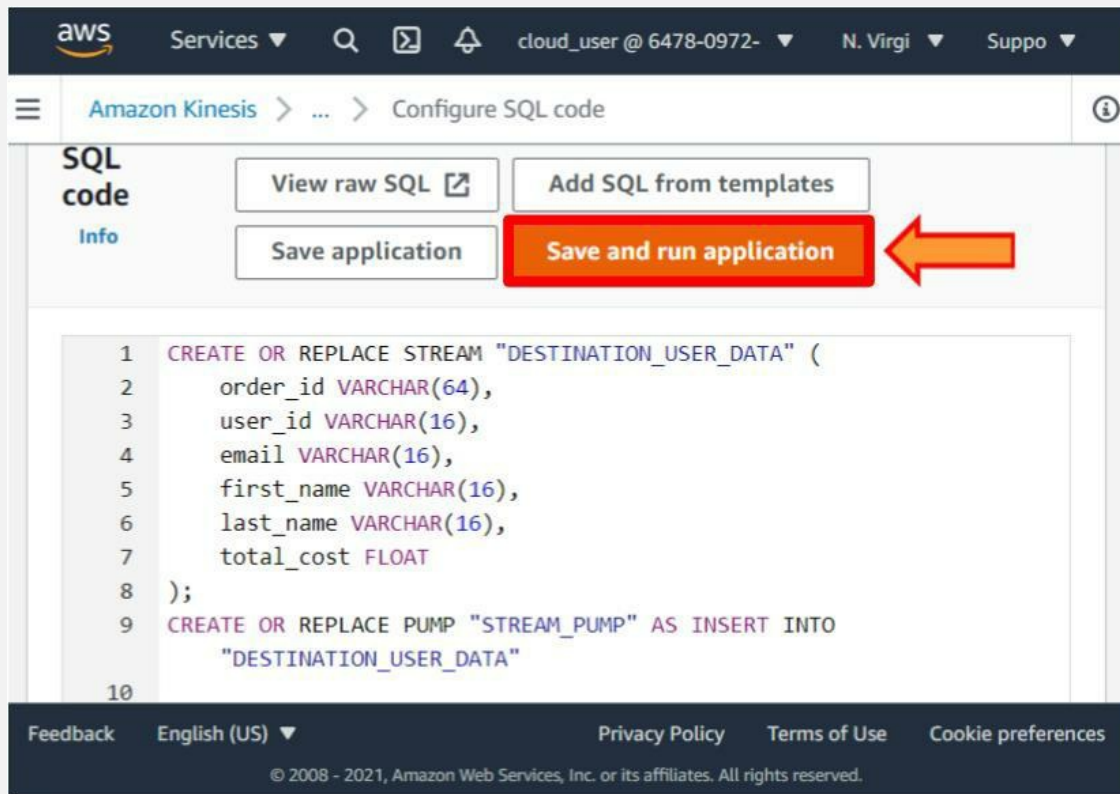
```
CREATE OR REPLACE STREAM "DESTINATION_USER_DATA"
(
    order_id VARCHAR(64),
    user_id VARCHAR(16),
    email VARCHAR(16),
    first_name VARCHAR(16),
    last_name VARCHAR(16),
    total_cost FLOAT
);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT
INTO "DESTINATION_USER_DATA"

SELECT STREAM "order_id", "user_id", "email",
"first_name", "last_name", "total_cost"
```

```
FROM "SOURCE_SQL_STREAM_001"  
WHERE "total_cost" >= 100;
```

20. Click on the **Save and run application** button.

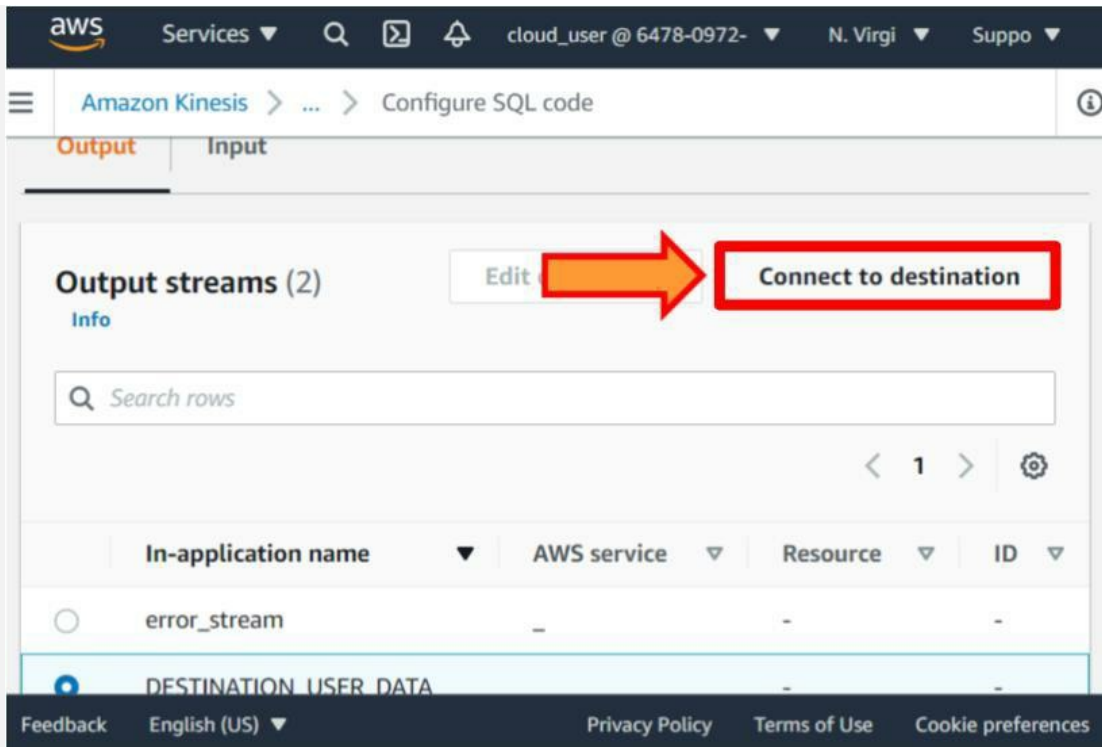


21. It will take a few minutes. Once it is completed, you will see the subset of data.

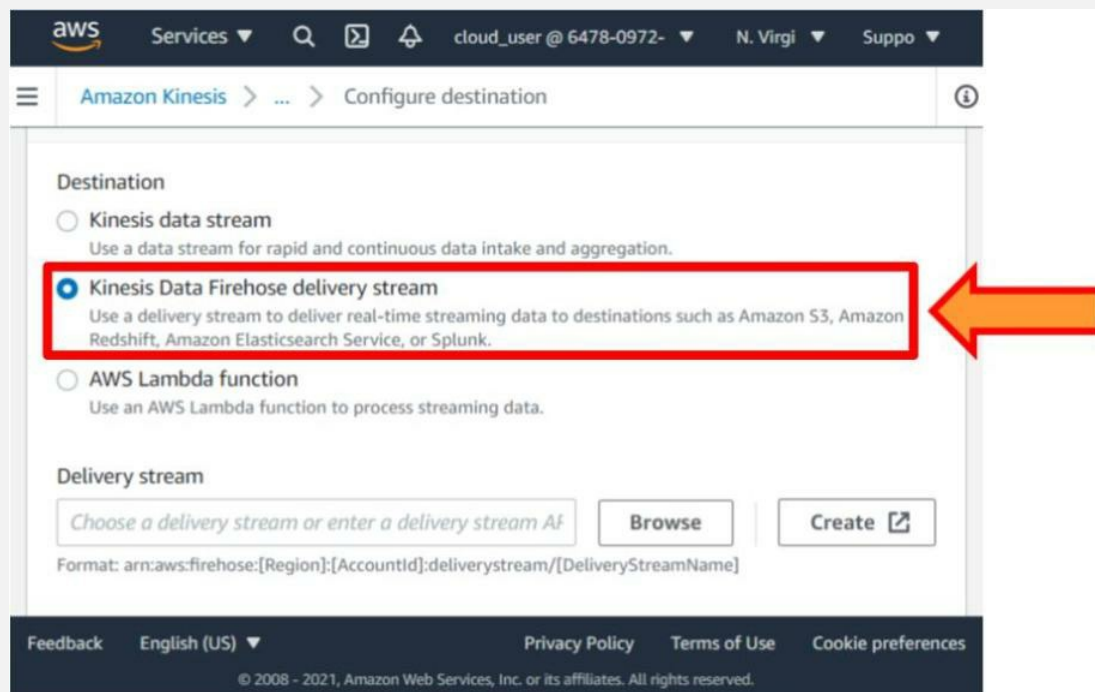
| ROWID       | ORDER_ID     | USER_ID | EMAIL        | FIRST_NAME | LAST_NAME |
|-------------|--------------|---------|--------------|------------|-----------|
| 2021-11-... | 4abe08d1...  | uid_156 | debbie.wo... | Debbie     | W...      |
| 2021-11-... | af7c3cd9-... | uid_253 | ambre.pic... | Ambre      | Pi...     |
| 2021-11-... | 9f2ad226...  | uid_55  | david.bro... | David      | Br...     |
| 2021-11-... | e64a94c3...  | uid_39  | brandon.g... | Brandon    | Gi...     |
| 2021-11-... | ad6b433b...  | uid_244 | gaspard....  | Gaspard    | Ma...     |

## Step 5: Create Kinesis Data Firehose to transform & Deliver the Final Result

1. Click on the **Connect to destination** button.



2. Select the **Kinesis Data Firehose delivery stream**.



3. Click on the **Create** button.

aws Services 🔍 ⓘ 🔔 cloud\_user @ 6478-0972- N. Virgi Suppo

Amazon Kinesis > ... > Configure destination ⓘ

Use a data stream for rapid and continuous data intake and aggregation.

- ☒ **Kinesis Data Firehose delivery stream**  
Use a delivery stream to deliver real-time streaming data to destinations such as Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, or Splunk.
- ☐ **AWS Lambda function**  
Use an AWS Lambda function to process streaming data.

**Delivery stream**

Choose a delivery stream or enter a delivery stream ARN **Create** ⓘ

Format: arn:aws:firehose:[Region]:[AccountId]:deliverystream/[DeliveryStreamName]

**Access permissions for writing output stream**  
Create or choose IAM role with the required permissions. [Learn more](#) ⓘ

- ☒ **Create / update IAM role kinesis-analytics-filter-top-orders-us-east-1 with required policies**

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

4. Select the source **Direct PUT**. Then select the destination **Amazon S3**.

aws Services 🔍 ⓘ 🔔 cloud\_user @ 6478-0972-0 N. Virgin Suppo

Amazon Kinesis > ... > Create delivery stream

**Choose source and destination**  
Specify the source and the destination for your delivery stream. You cannot change the source and destination of your delivery stream once it has been created.

**Source** ⓘ

Direct PUT

**Destination** ⓘ

Amazon S3

**Delivery stream name**

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

5. Give a delivery stream name **top-orders-delivery-stream**.

aws Services 🔽 🔍 📄 🔔 cloud\_user @ 6478-0972-0 🔽 N. Virgin 🔽 Suppo 🔽

Amazon Kinesis > ... > Create delivery stream

### Delivery stream name

Delivery stream name

**top-orders-delivery-stream**

Acceptable characters are uppercase and lowercase letters, numbers, underscores, hyphens, and periods.

### Transform and convert records - *optional*

Configure Kinesis Data Firehose to transform and convert your record data.

Feedback English (US) 🔽 Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

6. Scroll down. Select the **Enabled**.

aws Services 🔽 🔍 📄 🔔 cloud\_user @ 6478-0972-0 🔽 N. Virgin 🔽 Suppo 🔽

Amazon Kinesis > ... > Create delivery stream

### Transform source records with AWS Lambda [Info](#)

Kinesis Data Firehose can invoke an AWS Lambda function to transform, filter, un-compress, convert and process your source data records. The specified AWS Lambda function can also be used to provide dynamic partitioning keys for the incoming source data before its delivery to the specified destination.

#### Data transformation

☐ Disabled

**☒ Enabled**

AWS Lambda function Version or alias

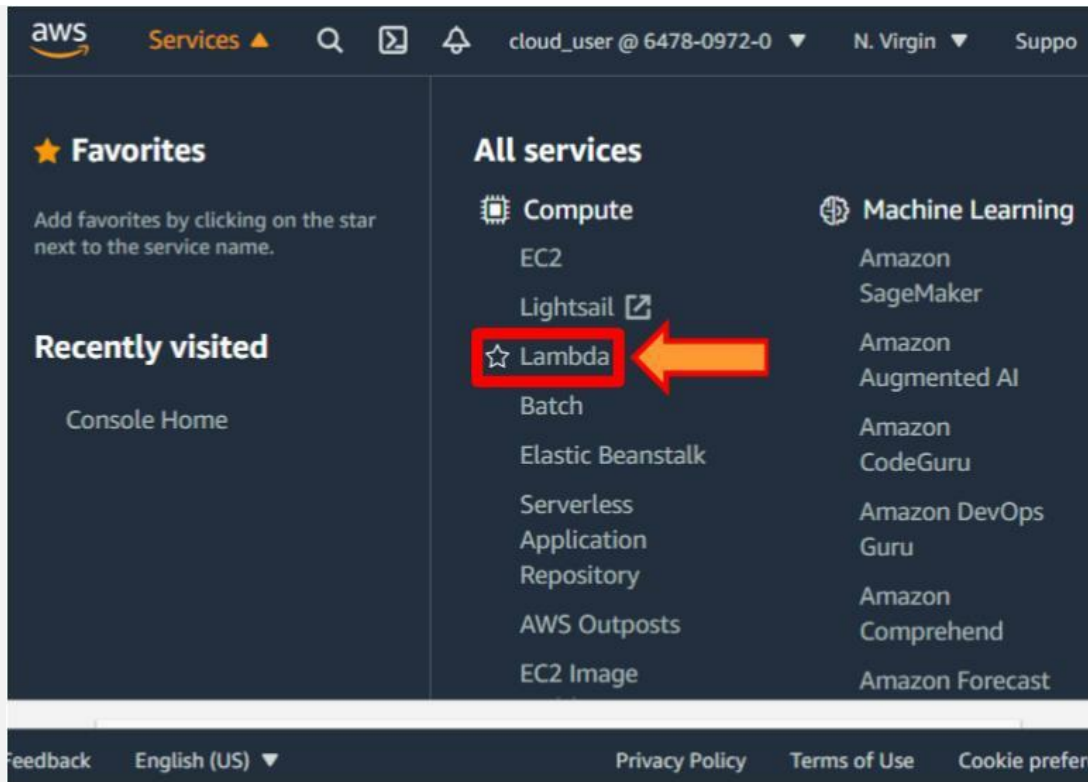
Format: arn:aws:lambda:[Region]:[AccountId]:function:[FunctionName]

Buffer size

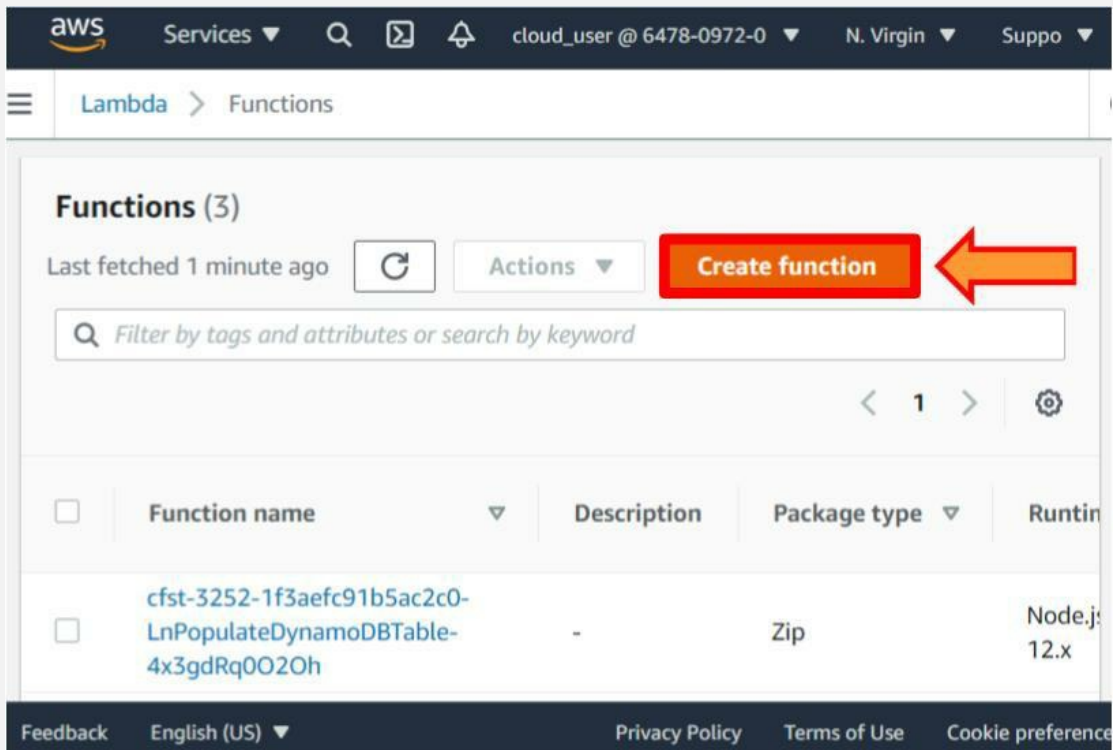
Feedback English (US) 🔽 Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

7. Open the **Lambda** on the new tab.



8. Click on the **Create function** button.



9. Enter the name for the function **add-new-line**.

10. Select the **Python 3.8** Runtime.

aws Services 🔽 🔍 📄 🔔 cloud\_user @ 6478-0972-0 🔽 N. Virgin 🔽 Suppo 🔽

☰ Lambda > ... > Create function

**Function name**  
Enter a name that describes the purpose of your function.  
**add-new-line**  
Use only letters, numbers, hyphens, or underscores with no spaces.

**Runtime** [Info](#)  
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.  
**Python 3.8** ▼

**Architecture** [Info](#)  
Choose the instruction set architecture you want for your function code.  
☒ x86\_64  
☐ arm64

Feedback English (US) 🔽 Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

11. Click on the **Change default execution role**.

aws Services 🔽 🔍 📄 🔔 cloud\_user @ 6478-0972-0 🔽 N. Virgin 🔽 Suppo 🔽

☰ Lambda > ... > Create function ⓘ

Choose the instruction set architecture you want for your function code.  
☒ x86\_64  
☐ arm64

**Permissions** [Info](#)  
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▶ **Change default execution role**

▶ **Advanced settings**

Feedback English (US) 🔽 Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

12. Select **Use an existing role**.

aws Services 🔻 🔍 📄 🔔 cloud\_user @ 6478-0972-0 🔻 N. Virgin 🔻 Suppo 🔻

☰ Lambda > ... > Create function

### Permissions [Info](#)

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

#### ▼ Change default execution role

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☐ Create a new role with basic Lambda permissions

☒ Use an existing role

☐ Create a new role from AWS policy templates

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

Feedback English (US) 🔻 Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

13. Use the Existing role dropdown to select the **enrich-orders-function** role you created earlier.

aws Services 🔻 🔍 📄 🔔 cloud\_user @ 6478-0972-0 🔻 N. Virgin 🔻 Suppo 🔻


☰ Lambda > ... > Create function

☐ Create a new role from AWS policy templates

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

service-role/enrich-orders-function-role-dc7bun99 ▼



[View the enrich-orders-function-role-dc7bun99 role](#) on the IAM console.

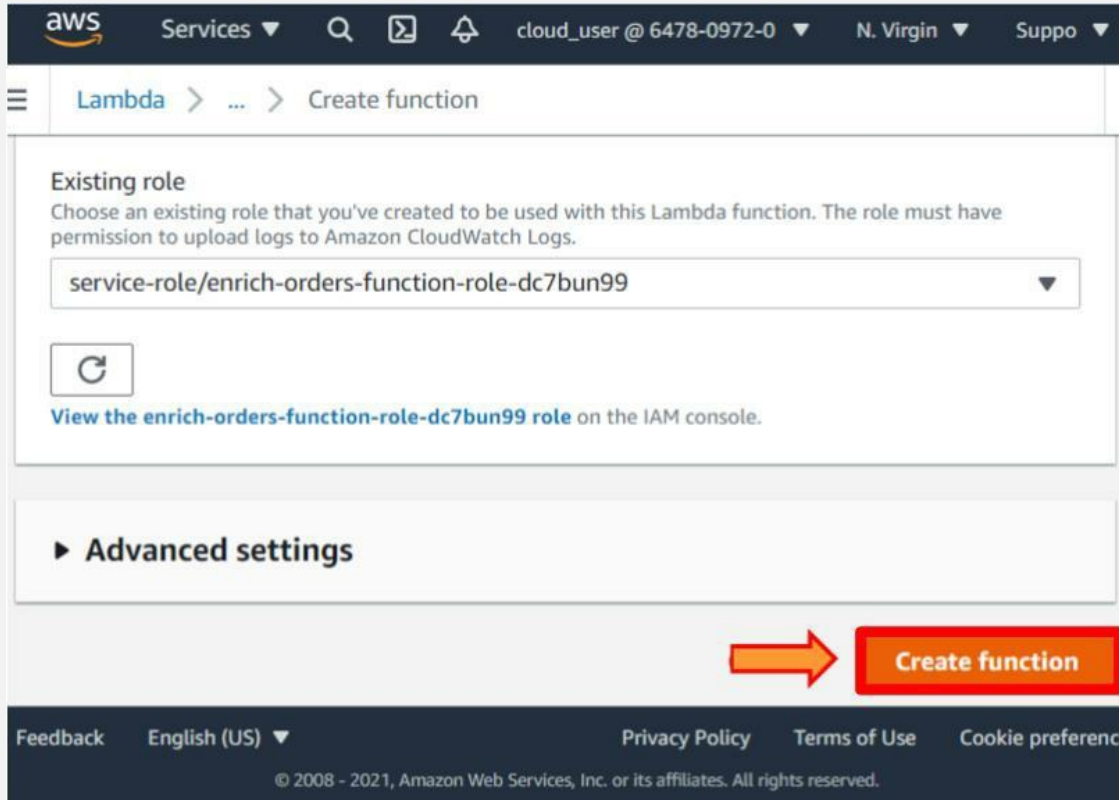
► Advanced settings

Cancel Create function

Feedback English (US) 🔻 Privacy Policy Terms of Use Cookie preference

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

14. Click on the **Create function** button.



15. Copy and paste code in the lambda function code editor from the following provided link: <https://das-co1-data-analytics-specialty.s3.amazonaws.com/Labs/new-line-function.py>.

```
from __future__ import print_function

import base64

print('Loading function')

def lambda_handler(event, context):
    output = []

    for record in event['records']:
        print(record['recordId'])
        payload = base64.b64decode(record['data'])

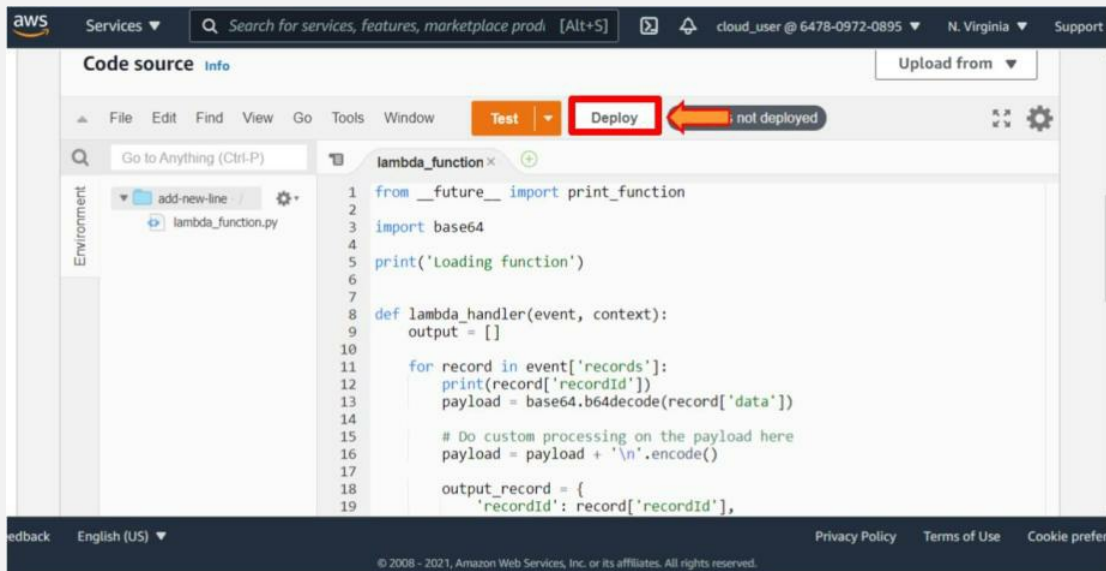
        # Do custom processing on the payload here
        payload = payload + '\n'.encode()

        output_record = {
            'recordId': record['recordId'],
            'result': 'Ok',
            'data': base64.b64encode(payload)
        }
        output.append(output_record)

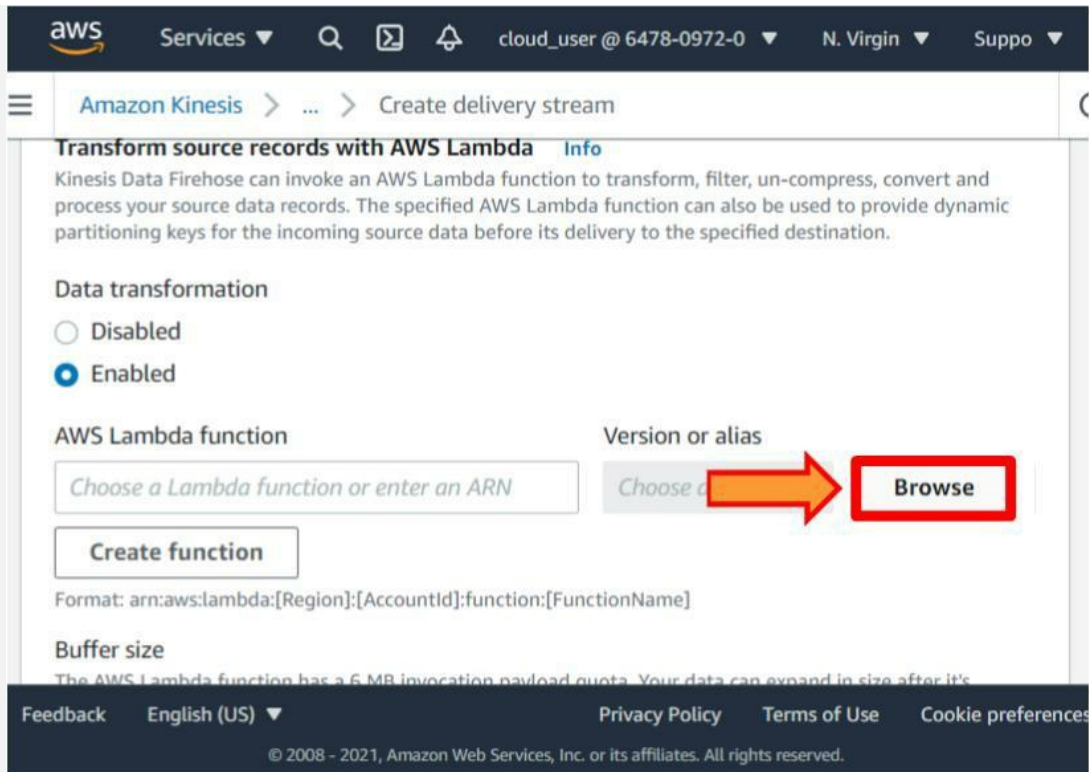
    print('Successfully processed {} records.'.format(len(event['records'])))

    return {'records': output}
```

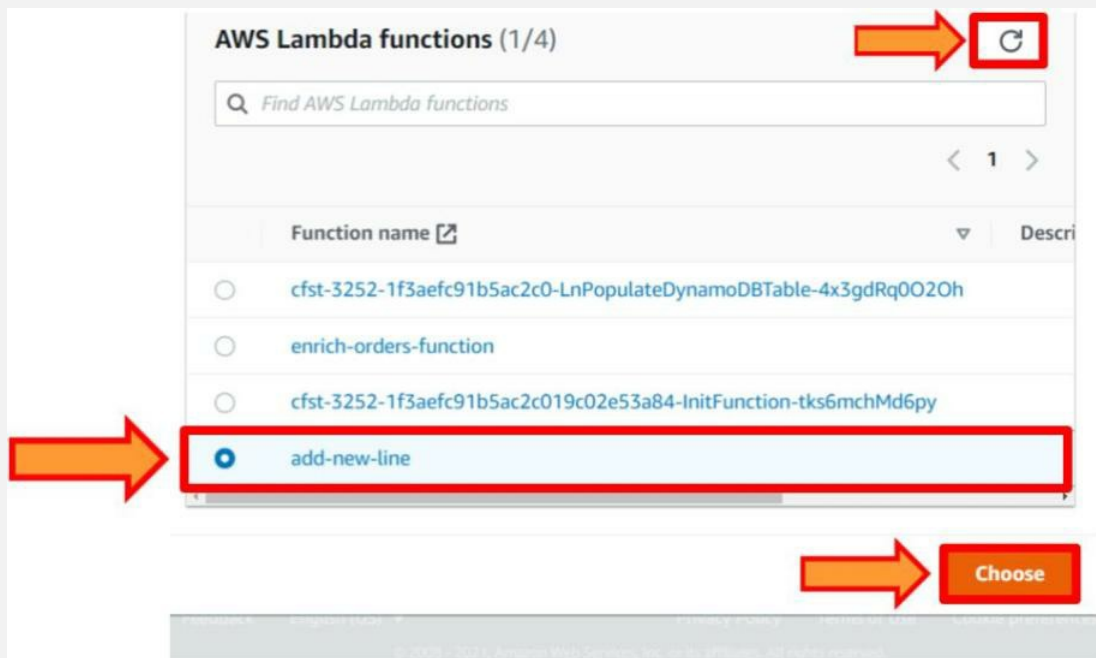
16. Click on the **Deploy** button to save the changes.



17. Go back to the Kinesis Data Firehose dashboard. Click on the **Browse** button.



18. Click on the **Refresh** button.
19. Select on the **add-new-line** Lambda function.
20. Click on the **Choose** button.



21. Set the Buffer size to **1 MB**.
22. Set the Buffer interval to **60 sec**.

aws Services 🔻 🔍 📄 🔔 cloud\_user @ 6478-0972-0 🔻 N. Virgin 🔻 Suppo 🔻

Amazon Kinesis > ... > Create delivery stream ⓘ

**Buffer size**

The AWS Lambda function has a 6 MB invocation payload quota. Your data can expand in size after it's processed by the AWS Lambda function. A smaller buffer size allows for more room should the data expand after processing.

MB ←

Minimum: 1 MB, maximum: 3 MB.

**Buffer interval**

The period of time during which Kinesis Data Firehose buffers incoming data before invoking the AWS Lambda function. The AWS Lambda function is invoked once the value of the buffer size or the buffer interval is reached.

seconds ←

Minimum: 60 seconds, maximum: 900 seconds.

**Convert record format** Info

Data in Apache Parquet or Apache ORC format is typically more efficient to query than JSON. Kinesis Data

Feedback English (US) 🔻 Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

23. Click on the **Create** button to create an S3 bucket.

aws Services 🔻 🔍 📄 🔔 cloud\_user @ 6478-0972-0 🔻 N. Virgin 🔻 Suppo 🔻

Amazon Kinesis > ... > Create delivery stream ⓘ

**Destination settings** Info

Specify the destination settings for your delivery stream.

**S3 bucket**

Br →

Format: s3://bucket

**Dynamic partitioning** Info

Dynamic partitioning enables you to create targeted data sets by partitioning streaming S3 data based on partitioning keys. You can partition your source data with inline parsing and/or the specified AWS Lambda function. You can enable dynamic partitioning only when you create a new delivery stream. You cannot enable dynamic partitioning for an existing delivery stream. Enabling dynamic partitioning incurs additional costs per GiB of partitioned data. For more information, see [Kinesis Data Firehose pricing](#). 🔗

☒ Disabled

Feedback English (US) 🔻 Privacy Policy Terms of Use Cookie preference

24. Give an S3 bucket name **ips-s3-example- bucket**.

The screenshot shows the 'General configuration' section of the AWS S3 'Create bucket' page. The 'Bucket name' field contains 'ips-s3-example-bucket' and is highlighted with a red rectangle. An orange arrow points to this field from the right. Below the field, a note states: 'Bucket name must be unique and must not contain spaces or uppercase letters. See rules for bucket naming'. The 'AWS Region' dropdown is set to 'US East (N. Virginia) us-east-1'. There is a section for 'Copy settings from existing bucket - optional' with a 'Choose bucket' button. The footer includes 'Feedback', 'English (US)', 'Privacy Policy', 'Terms of Use', 'Cookie preference', and copyright information.

aws Services 🔍 ⓘ 🔔 cloud\_user @ 6478-0972-089 ▼ Globa ▼ Support ▼

### General configuration

Bucket name

ips-s3-example-bucket

Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

AWS Region

US East (N. Virginia) us-east-1 ▼

Copy settings from existing bucket - *optional*  
Only the bucket settings in the following configuration are copied.

Choose bucket

Feedback English (US) ▼ Privacy Policy Terms of Use Cookie preference

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

25. Click on the **Create bucket**.

The screenshot shows the 'Advanced settings' section of the AWS S3 'Create bucket' page. The 'Server-side encryption' section has 'Disable' selected. Below this is the 'Advanced settings' section header. A light blue box contains an information icon and the text: 'After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.' At the bottom right, the 'Create bucket' button is highlighted with a red rectangle, and an orange arrow points to it from the left. The footer is identical to the previous screenshot.

aws Services 🔍 ⓘ 🔔 cloud\_user @ 6478-0972-089 ▼ Globa ▼ Support ▼

### Server-side encryption

☒ Disable  
☐ Enable

### Advanced settings

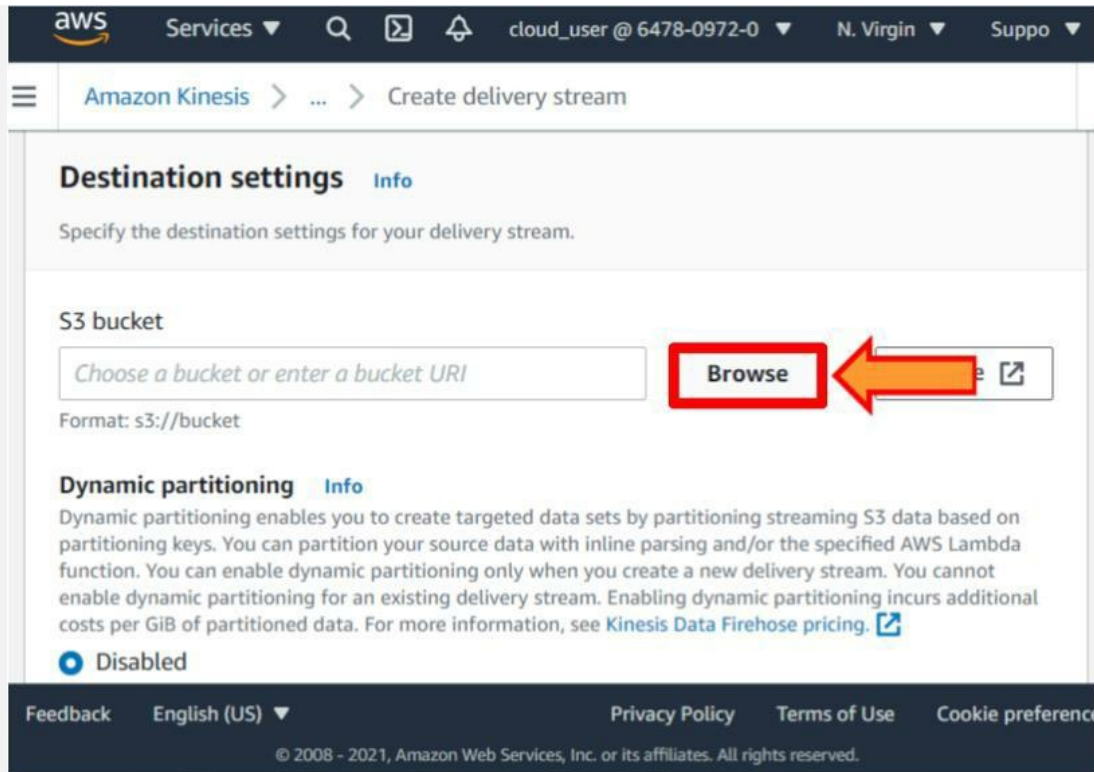
After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.

Create bucket

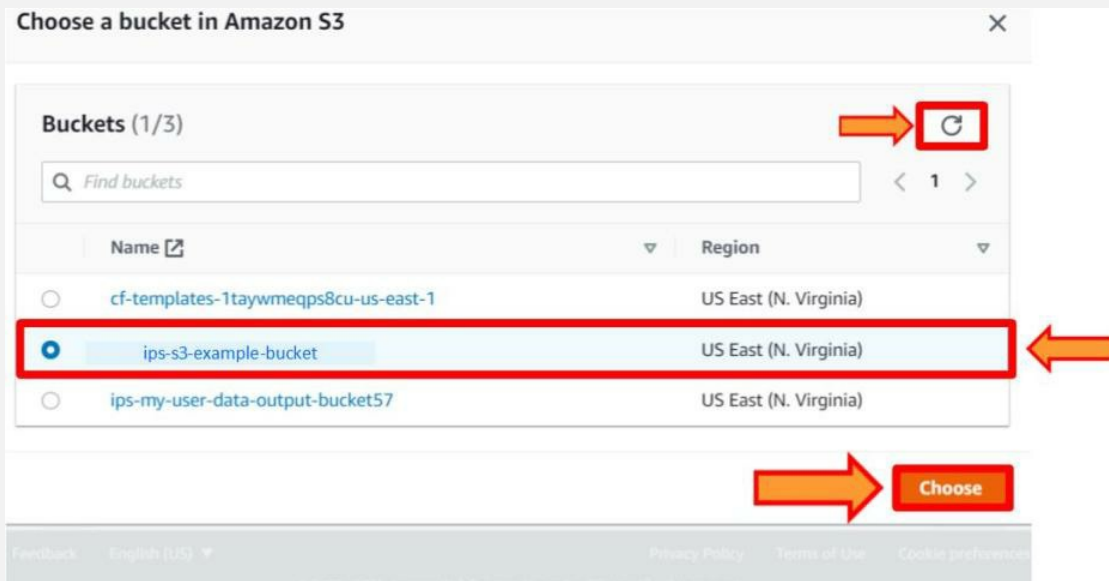
Feedback English (US) ▼ Privacy Policy Terms of Use Cookie preference

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

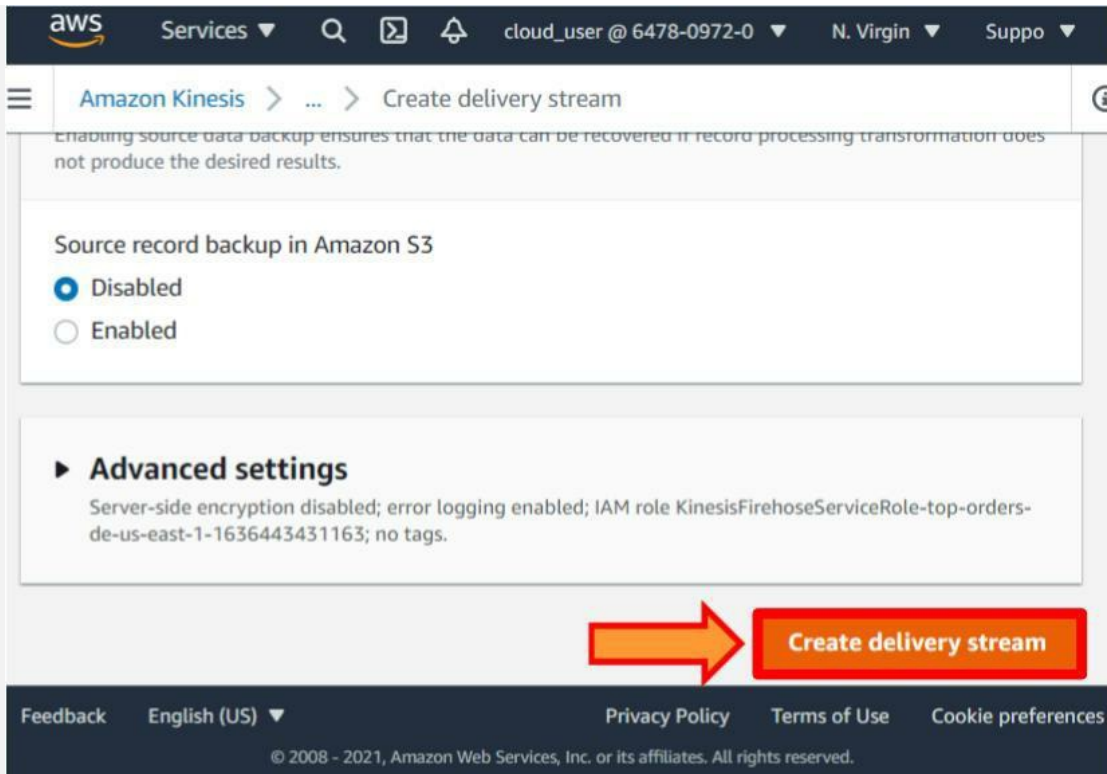
26. Click on the **Browse** button.



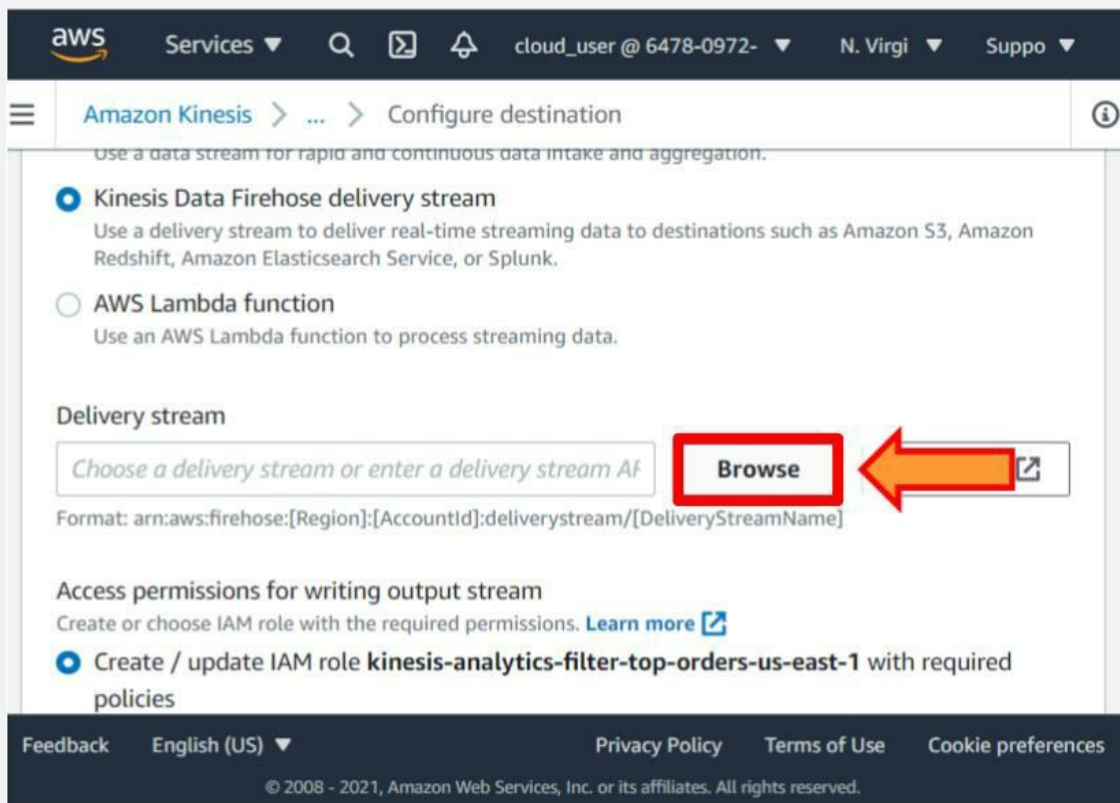
27. Click on the **Refresh** button.
28. Select the **ips-s3-example-bucket**.
29. Click on the **Choose** button.



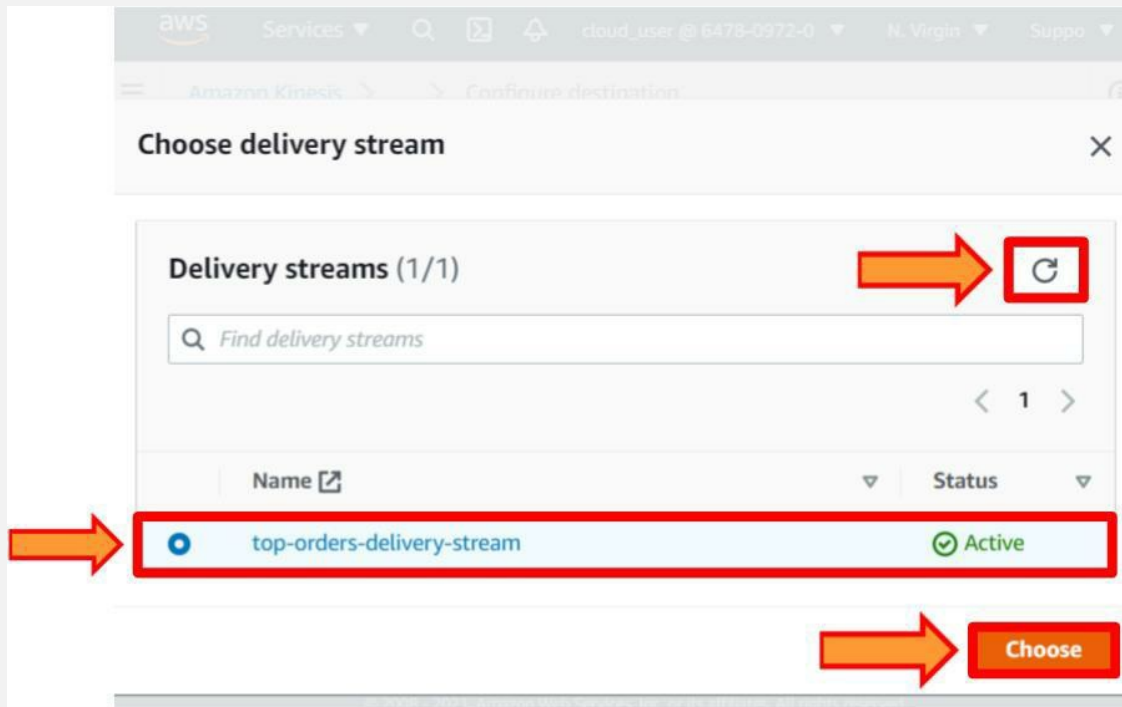
30. Click on the **Create delivery stream** button.



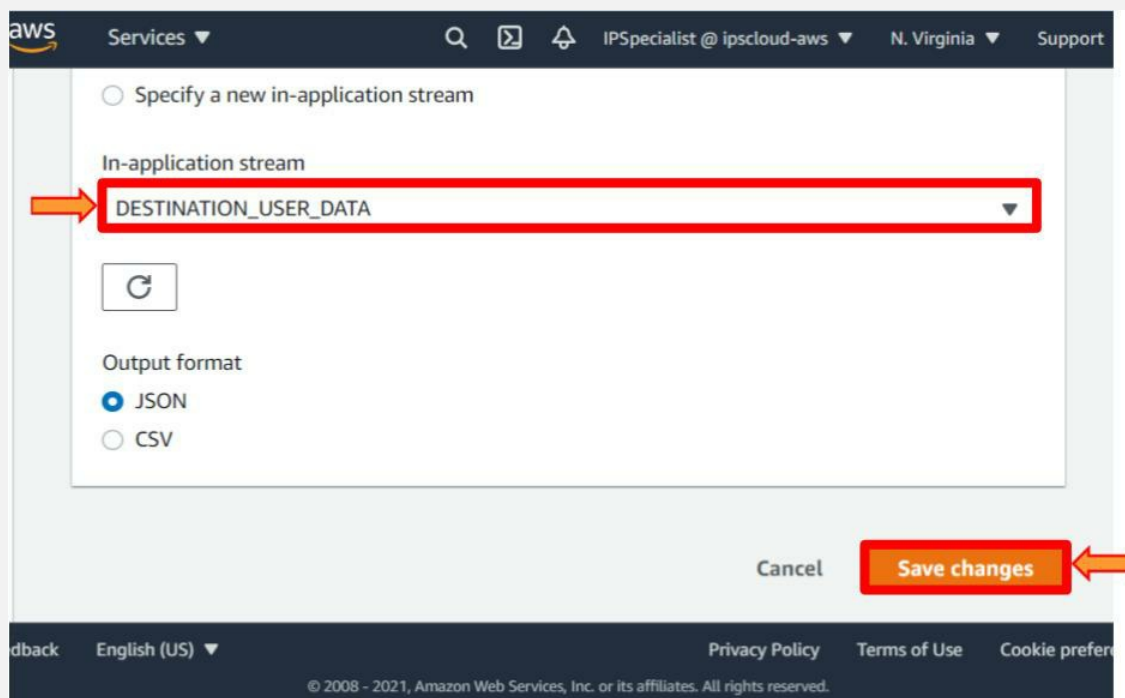
31. Go back to the Kinesis Firehose Delivery Stream dashboard. Click the **Browse** button.



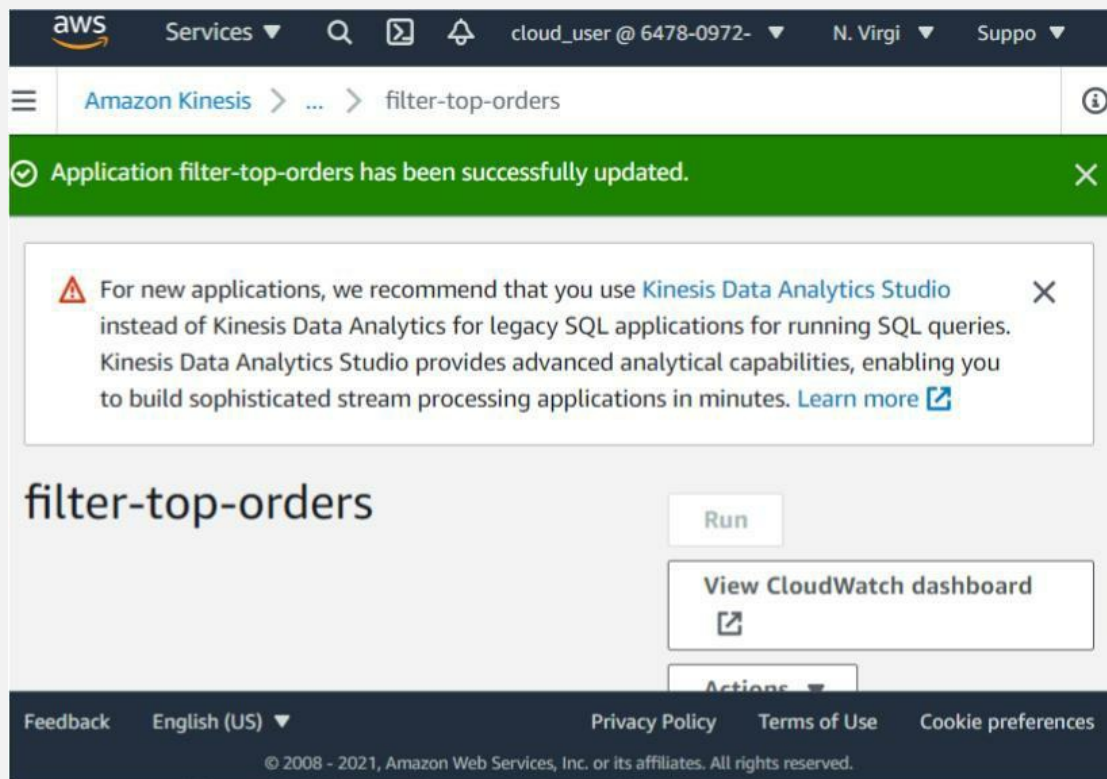
32. Click on the **Refresh** button.
33. Select the **top-orders-delivery-stream**.
34. Click on the **Choose** button.



35. Scroll down. Select the **DESTINATION\_USER\_DATA**.
36. Then, click on the **Save changes** button.

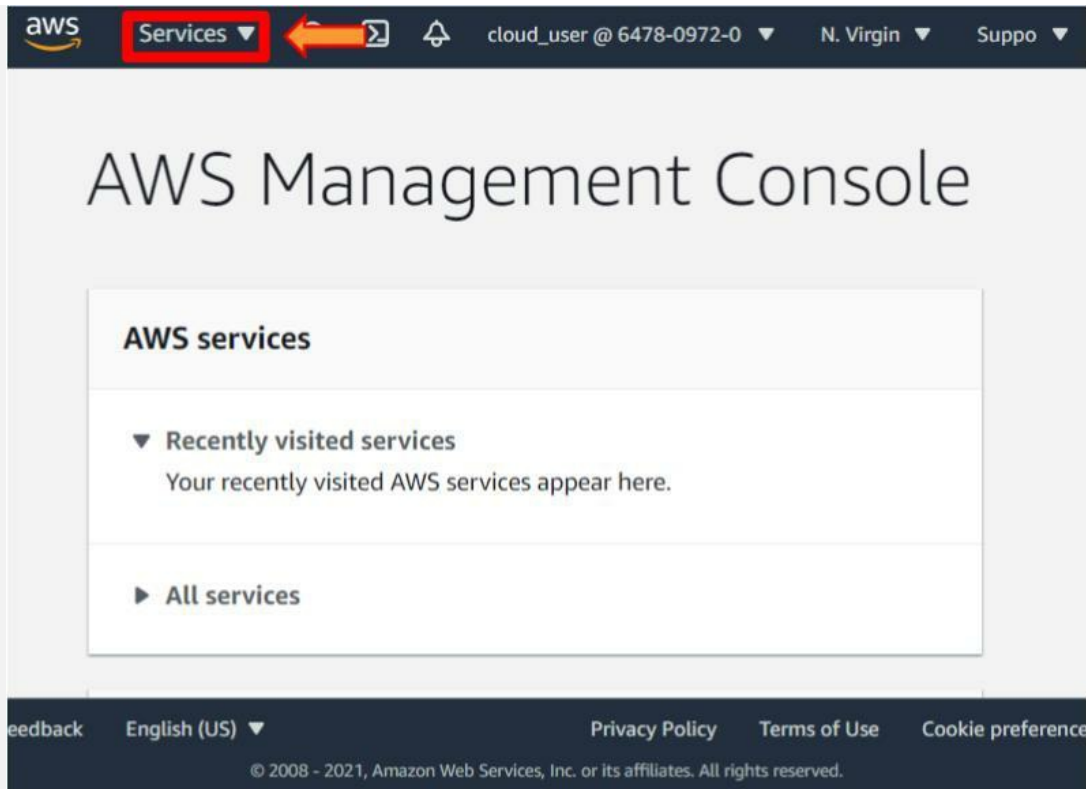


37. Hence, the Kinesis Data Firehose is created.

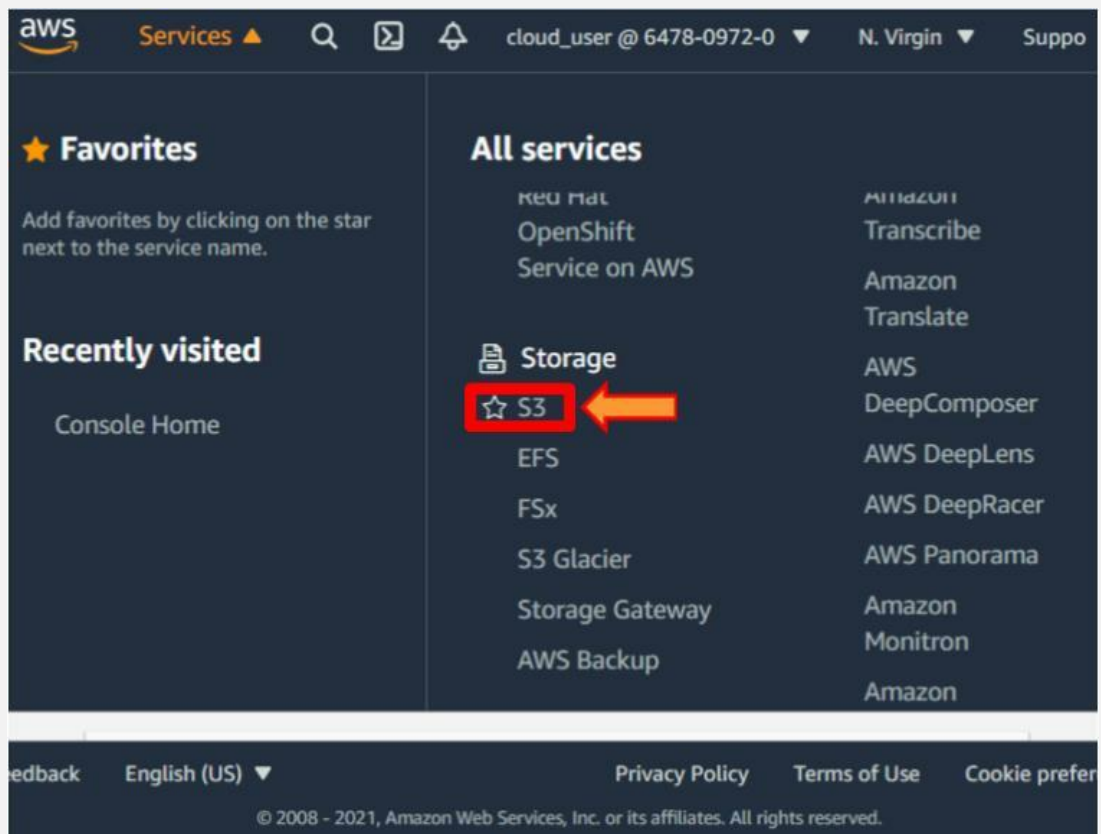


## Step 6: Review the Results in the S3 bucket

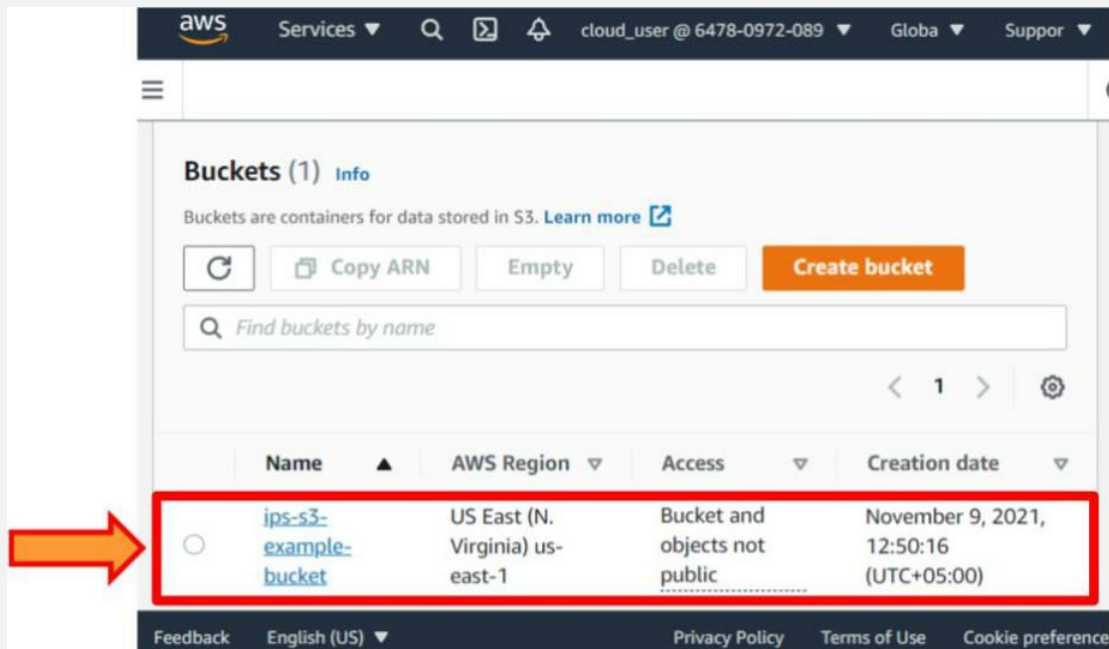
1. Click on the **Services**.



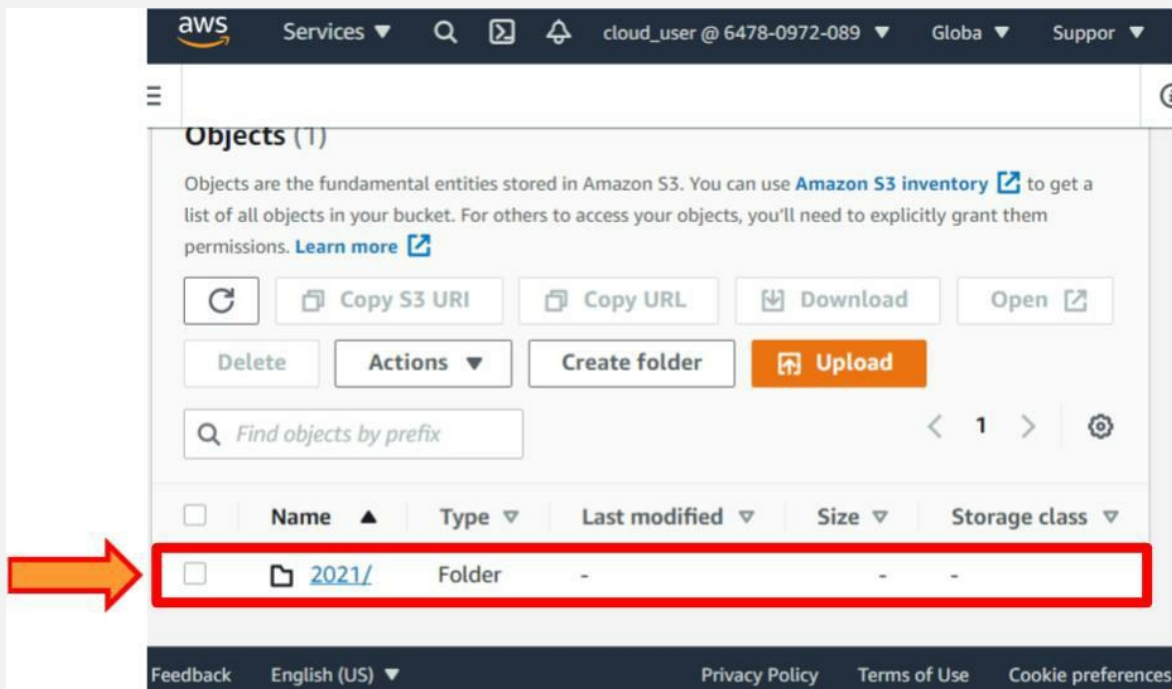
2. Select the **S3** from the **Storage**.



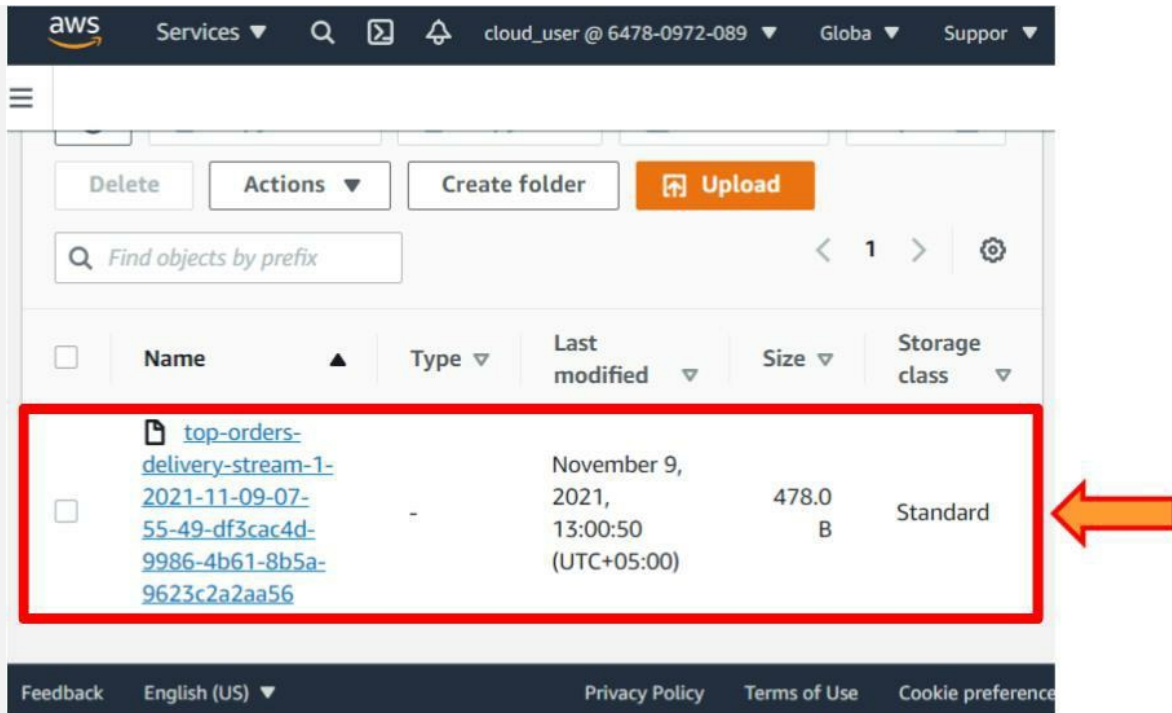
3. Click on the **ips-s3-example-bucket**.



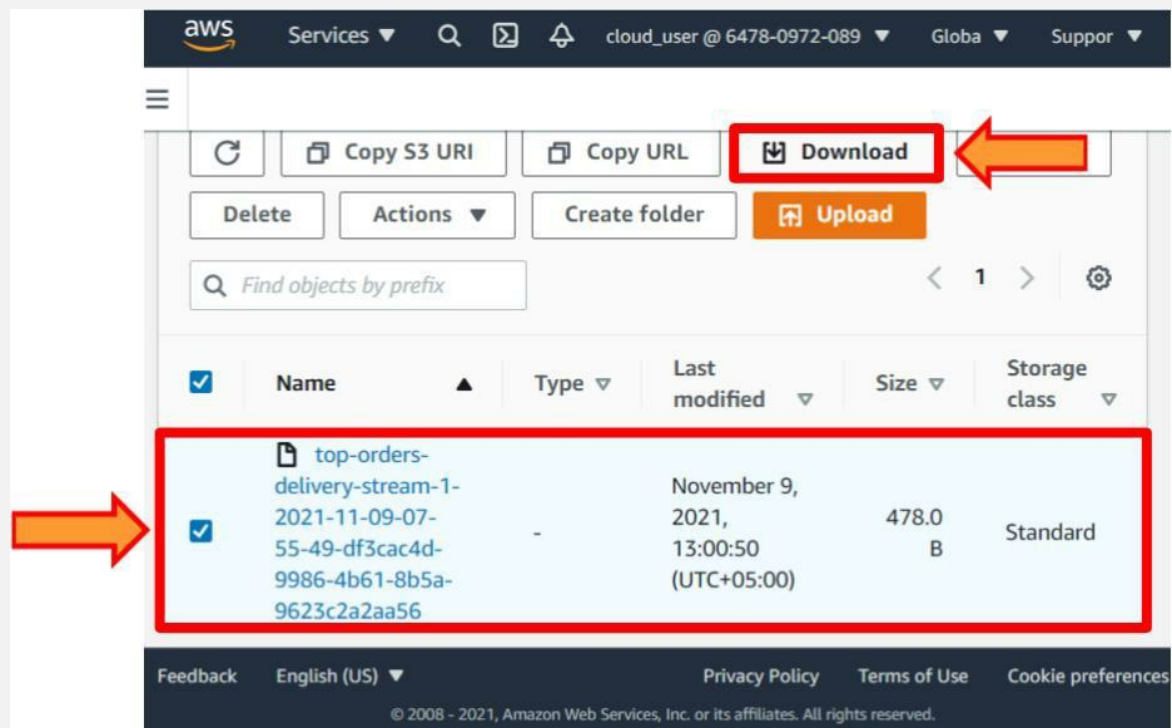
4. Click on the folders.



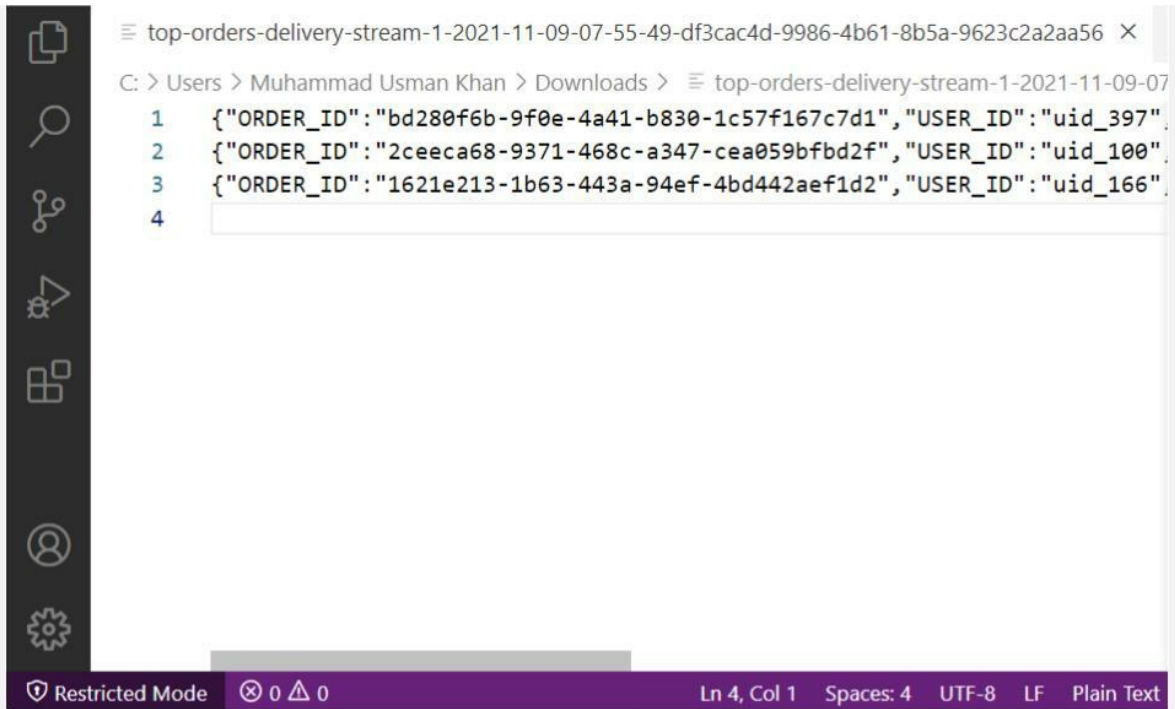
5. Finally, you can see a file that is uploaded onto the S3 bucket.



6. Select the file. Click on the **Download** button.



7. Open the download file in any code editor you will see, like JSON.

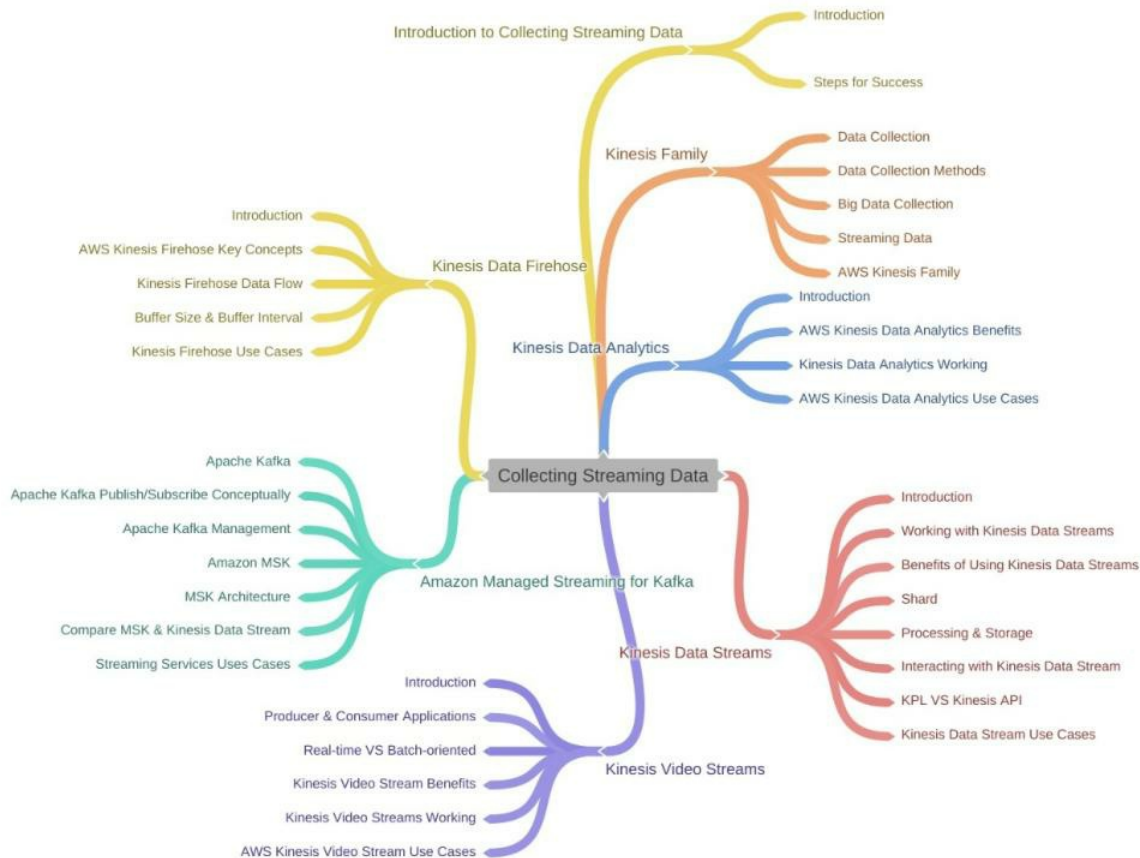


The screenshot shows a Notepad++ window with a file named "top-orders-delivery-stream-1-2021-11-09-07-55-49-df3cac4d-9986-4b61-8b5a-9623c2a2aa56". The file path is "C:\Users\Muhammad Usman Khan\Downloads\top-orders-delivery-stream-1-2021-11-09-07-55-49-df3cac4d-9986-4b61-8b5a-9623c2a2aa56". The file contains three lines of JSON data, each representing an order record. The status bar at the bottom indicates "Restricted Mode", "0 errors, 0 warnings", "Ln 4, Col 1", "Spaces: 4", "UTF-8", "LF", and "Plain Text".

```
1 {"ORDER_ID": "bd280f6b-9f0e-4a41-b830-1c57f167c7d1", "USER_ID": "uid_397",
2 {"ORDER_ID": "2ceeca68-9371-468c-a347-cea059bfd2f", "USER_ID": "uid_100",
3 {"ORDER_ID": "1621e213-1b63-443a-94ef-4bd442aef1d2", "USER_ID": "uid_166",
4
```

8. Hence, you can successfully join, enrich, and transform streaming data using Amazon Kinesis.

# Mind Map



*Figure 4-36: Collecting Streaming Data Mind Map*

## Practice Questions

1. Which AWS resource helps you gather and handle streams of data records in real-time?
  - A. Kinesis Data Streams
  - B. Kinesis Data Firehose
  - C. Kinesis Data Video Streams
  - D. Kinesis Data Analytics

2. Which AWS service delivers real-time streaming data to a destination such as Amazon S3?
  - A. Kinesis Data Streams
  - B. Kinesis Data Firehose
  - C. Kinesis Data Video Streams
  - D. Kinesis Data Analytics
3. Which AWS service allows you to stream live video from devices to develop applications for real-time video processing and batch-oriented video analytics?
  - A. Kinesis Data Streams
  - B. Kinesis Data Firehose
  - C. Kinesis Data Video Streams
  - D. Kinesis Data Analytics
4. Which AWS service do you use for standard SQL to handle and analyze streaming data?
  - A. Kinesis Data Streams
  - B. Kinesis Data Firehose
  - C. Kinesis Data Video Streams
  - D. Kinesis Data Analytics
5. Which are the rappers or containers containing all of the streaming data you want to load into AWS?
  - A. Box
  - B. Queue
  - C. Shard
  - D. Pipeline
6. Which of the following assists you in consuming data from a Kinesis data stream by handling sophisticated distributed

computing tasks?

- A. Kinesis Client Library
- B. Kinesis API
- C. Kinesis Producer Library
- D. Kinesis Data Stream

7. Which of the following is characterized as the provider of correct insights for research utilizing established validation techniques?

- A. Data visualizing
- B. Data collection
- C. Data handling
- D. Data cleansing

8. What are the intelligent data collection methods?

- A. Surveys
- B. Interviews
- C. Mobile devices & Websites
- D. Focus Groups

9. What devices are used by enterprises to capture large volumes of structured, semi-structured, and unstructured data?

- A. Deep Learning
- B. Big Data
- C. Machine Learning
- D. Data Analytics

10. Which of the following is a real-time data input and processing data storage system?

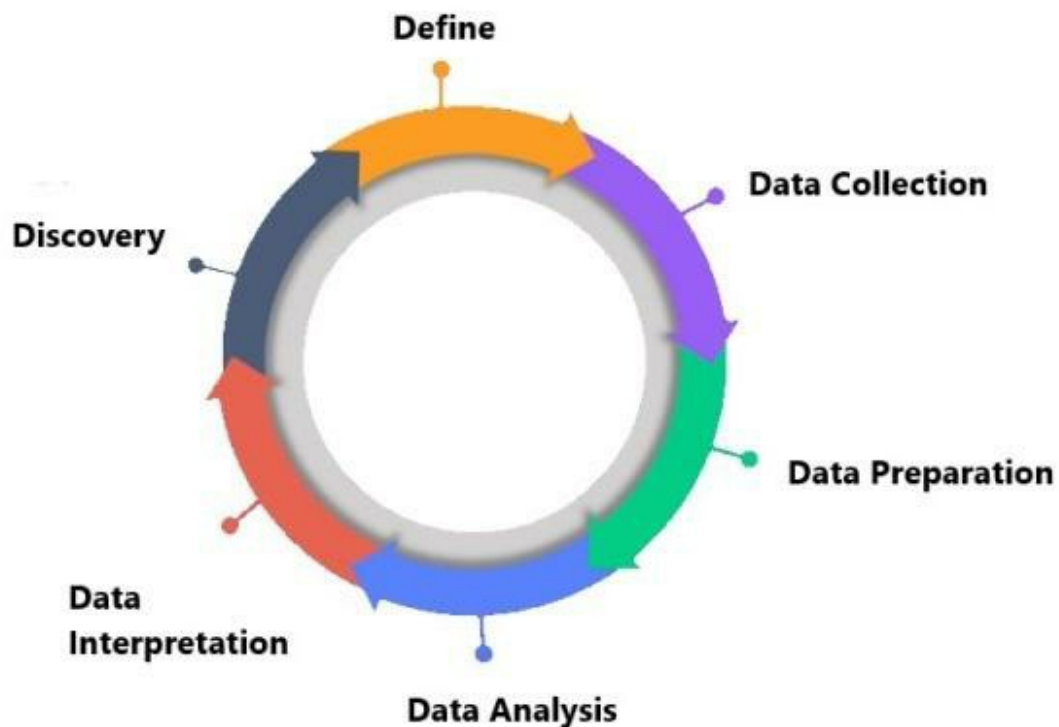
- A. Kinesis Data Stream
  - B. Kinesis Data Firehose
  - C. Kinesis Data Analytics
  - D. Apache Kafka
11. Which of the following is continually created by hundreds of data sources that generally send in data records in tiny batches?
- A. Data Cleansing
  - B. Data Handling
  - C. Data Analyzing
  - D. Data Streaming
12. Which AWS product simplifies the real-time streaming collection, processing, and analytics?
- A. Amazon S3
  - B. Amazon SageMaker
  - C. Amazon Kinesis
  - D. Amazon EC2
13. Your organization has a standalone Javascript (Node.js) application that streams data into AWS using Kinesis Data Streams. You can see that they are utilizing the Kinesis API (AWS SDK) rather than the Kinesis Producer Library (KPL). What might be the reasoning behind this?
- A. The Kinesis API (AWS SDK) runs faster in Javascript applications over the Kinesis Producer Library.
  - B. The Kinesis Producer Library must be installed as a Java application to use with Kinesis Data Streams.
  - C. The Kinesis API (AWS SDK) provides greater functionality over the Kinesis Producer Library.

- D. The Kinesis Producer Library cannot be integrated with a Javascript application because of its asynchronous architecture.
14. You have been tasked with capturing data from an online gaming platform to run analytics on and process through a machine learning pipeline. The data you are ingesting is the player's controller inputs in JSON format every 1 second (up to 10 players in a game). The data must be ingested via Kinesis Data Streams, and the JSON data blob has a size of 100 KB. What minimum number of shards can be used to ingest this data successfully?
- A. 1 shard
  - B. 10 shards
  - C. 100 shards
  - D. Greater than 500 shards, hence you will need to request more shards from AWS.
15. You are collecting clickstream data from an e-commerce website using Kinesis Data Firehose. To send data to the stream, you use the PutRecord API from the AWS SDK. What arguments are required when using the API PutRecord call to deliver data to Kinesis Data Firehose?
- A. Data, PartitionKey, StreamName
  - B. Data, PartitionKey, StreamName, ShardId
  - C. DataStreamName, PartitionKey, and Record (containing the data)
  - D. DeliveryStreamName and Record (containing the data)

# CHAPTER 05: DATA COLLECTION AND GETTING DATA INTO AWS

## Introduction

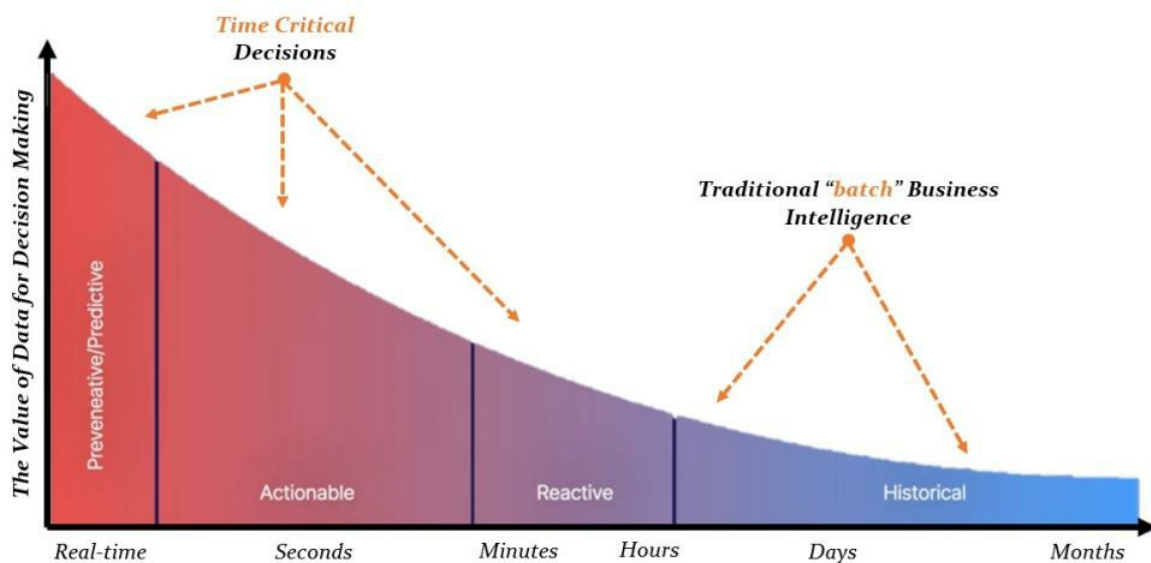
In this chapter, you will learn how to collect data on AWS and how to decide the best ways to ingest your data into AWS. You will also learn how to collect data in AWS via a dedicated network or using hardware appliances and transfer databases using the Database Migration Service (DMS). You will also learn how to use the Amazon Kinesis family of services and when each one is most useful. For data interpretation and the discovery of our data, we can use QuickSight.



*Figure 5-01: Steps to Success*

## Data Loses Value Quickly Over Time

From the graph, data loses value quickly over time. On the left-hand side, our streaming data or our preventative, predictive, and actionable data are where our time-critical decisions can be made. If we want fast, actionable data, then that is where our streaming data lies. We want to build tools around batch-oriented processes, build out ETL pipelines, or possibly build business intelligence tools around our data.



*Figure 5-02: Data Loses Value Quickly Over Time*

## Direct Connect, Snowball, Snowball Edge, Snowmobile

### AWS General Rule Of Thumb

AWS has a general rule of thumb. This graph lays out your network connection speed, the amount of data, and whether to use a managed or unmanaged service from AWS. The difference between an unmanaged and managed is that unmanaged uses the CLI, the AWS command line, or the AWS console to transfer data into AWS. A managed service is the Direct Connect option, the Snowball family, and

other tools that you can use to move data from one location to another or from on-premise into the AWS network.

| Network Connection   | Amount of Data      | Method    |
|----------------------|---------------------|-----------|
| Less than 10 Mbps    | Less than 500 GB    | Unmanaged |
| Greater than 10 Mbps | Greater than 500 GB | Managed   |

*Table 5-01: AWS General Rule of Thumb*

## **Data Migration Service (Managed Services To Move Your Data To AWS)**

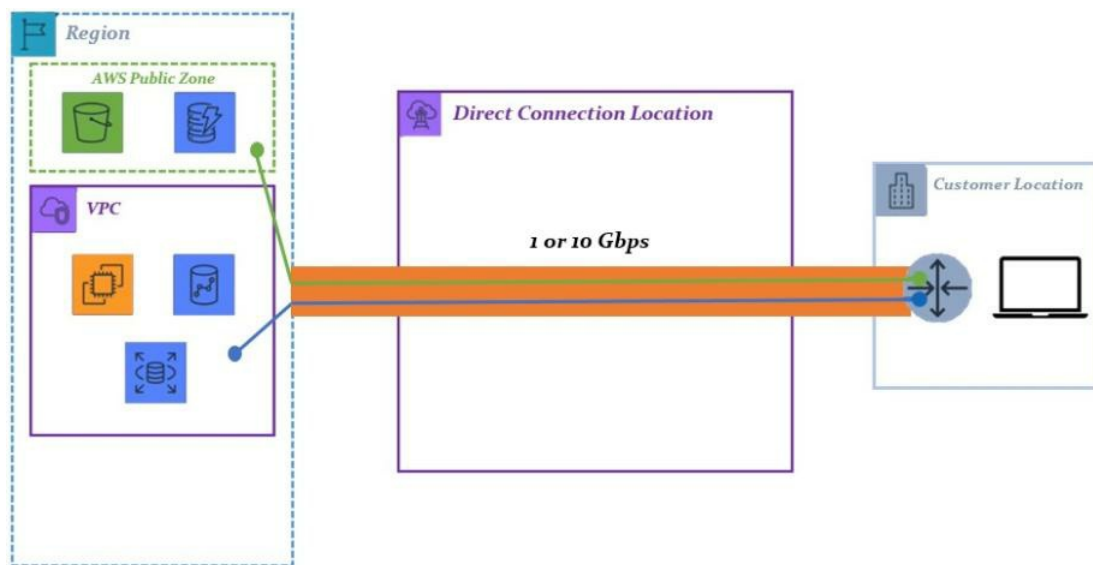
### ***Hybrid Cloud Storage***

The hybrid cloud storage connects your on-premises applications that require low-latency access or need rapid data transfer to cloud storage.

#### *Hybrid Cloud Storage Using Direct Connect*

We have an AWS region and a Direct Connect location. The AWS cages are live, and all networking routers and connectivity capabilities lie within the Direct Connect location. This can be a single physical location or can be a partner location. The dedicated lines are established from an AWS region into the AWS physical networking hardware. These dedicated lines can be either one or 10 gigabits per second. You as a customer have your customer on-premises location, your hardware, and your router that you want to set up to connect into the AWS Direct Connect location; AWS can own this, or it can be a partner location, and you would make sure that you have all of the hardware set up to connect into the AWS cage. You have a Direct Connection from your customer location into the AWS region through

the Direct Connect location. It is a dedicated hard line into AWS. Once you have established a Direct Connect connection, you can either connect into the AWS public zone, this is public services like S3 and DynamoDB or connect into services hosted within a VPC like EC2 instances Redshift, or RDS. This allows you to connect directly into an AWS region and not have to traverse the internet. You can maximize your latency and how fast you can move data from your location, from your data center, from your on-premises infrastructure into AWS.



*Figure 5-03: Hybrid Cloud Storage Using Direct Connect*

### ***Online Data Transfer***

It makes it simple and easy to transfer your data into and out of AWS via online methods. AWS Data Sync allows you to automate moving data between your on-premises storage into AWS, into S3, EFS, or FSX, which is a windows file server. You can use AWS Data Sync to transfer data at speeds up to 10 times faster than some of the other open-source tools that you can use. You can use Data Sync for one-time data migrations or have recurring data processing workflows. You can also

automate the replication and the data protection and recovery of your data. It is a great tool to set up automation for moving data between various locations into AWS. Several different transfer families allow you to transfer directly into and out of S3. You can use FTP, SFTP, or FTPS. S3 Transfer Acceleration allows you to maximize your available bandwidth regardless of the distance from your customers into S3. Kinesis Data Firehose is a simple way to load streaming data into AWS.

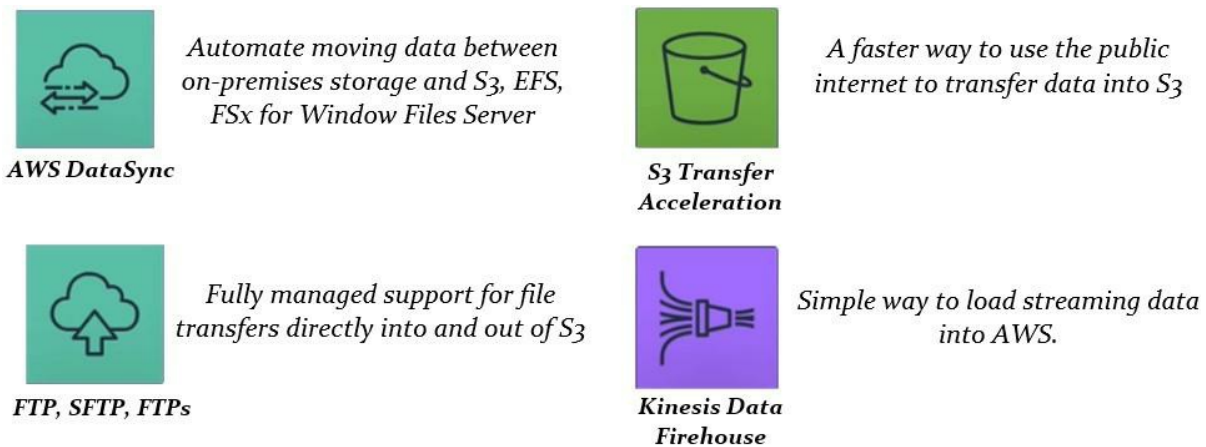


Figure 5-04: Online Data Transfer

Snowcone is the newest and the smallest member of the AWS snow family that allows you to collect, process, and move data to AWS online with AWS Data Sync.

**EXAM TIP:** Use Snowcone to collect, process, and move data to AWS online with AWS DataSync.

### **Offline Data Transfer – The Snow Family**

This introduces the snow family, which makes it simple to get your data into and out of AWS via offline methods.

*Snowcone*

You can load data to Snowcone through Wi-fi wired 10 GbE networking.

You can then ship the device with data to AWS for offline data transfer.

*Snowball*

Used for petabyte-scale data transport with import and export to S3.

*Snowmobile*

An exabyte-scale data transport solution uses a secure semi 40-foot shipping container to transfer large amounts of data into and out of AWS.

*Snowball Edge*

It is local storage and large-scale data transfer. Also, local Lambda and EC2 instances compute, and AWS IoT Greengrass.



Figure 5-05: The Snow Family

## Which Solution Should you Use?

| Scenario | Data Migration | Reason |
|----------|----------------|--------|
|----------|----------------|--------|

|   | Plan                      |  |
|---|---------------------------|--|
| You need a consistent, high throughput connection to transfer data in and out of AWS from your on-premise databases. The transfer speed can never fall between 1 Gbps.                                      | Direct Connect Connection | Direct connect gives customer locations a dedicated fiber channel with high throughput and low latency onto the AWS backbone. You can choose from 1 Gbps to 10 Gbps connections. |
| Your team manages a fleet of IoT devices that monitors engine parts on a remote deep-sea fishing rig with intermittent internet connectivity. These IoT devices need to send and process data using Lambda. | Snowball Edge             | You can use these devices for data collection and processing in environments with intermittent connectivity.   |

*Table 5-02: Which Solution You Should Use*

**EXAM TIP:**

- Moving Mass Amounts of Data – Unmanaged and managed services
- Data Migration Services – Hybrid Cloud Storage, Online Data Transfer, and Offline Data Transfer
- Direct Connect – Use 1 Gbps to 10 Gbps dedicated networking
- The Snow Family – Snowball, Snowball Edge, and Snowmobile

## Database Migration Service

### 1. *Data Migration*

Easily and securely migrate widely used commercial and open-source databases and data warehouses into the cloud.

### 2. *Replication*

Easily replicate your databases and data warehouse between two locations.

### 3. *Fully Operational*

Databases stay fully operational during the migration, minimizing downtime for the applications using them.

## DMS Use Cases

### 1. *Migrate Applications*

Migrate business-critical databases, migrate from Classic to VPC, costly and license-driven data warehouse to Redshift.

### 2. *Upgrade*

With DMS, you can upgrade versions of your database software easily with no downtime.

### 3. *Achieve Old Data*

You can migrate historical data to a more cost-efficient storage solution while still retaining the Solution.

### 4. *Migrate Datastores*

You can migrate from NoSQL to SQL, SQL to NoSQL, and SQL to SQL.

## Supported Migrations

The supported migrations that DMS offers are we can transfer data from one of the database engines on the left-hand side to any of the

database engines on the right-hand side, as shown in the following figure. We can migrate all our data into S3. Whenever we use S3 as the target, we can store the data in a CSV format, or for more compact and faster query options, we can use the Apache Parquet format. We can also use DMS to migrate our data from an existing database onto S3.



*Figure 5-06: Supported Migrations*

## Migrations

We define the source endpoint, where the data comes from, and we have a target endpoint. DMS supports heterogeneous and homogenous migrations.



Figure 5-07: Migrations

## Mass Amount Of Data

Use a Snowball device to:

- Store 80 TB storage, 10 GB network
- User interface similar to S3
- All data is encrypted end-to-end

## Replication

Leverages “change data Capture,” which pulls just the changes from the source and delivers them to the destination.

### 1. *Cross-Region Replication*

It gives you the ability to create cross-region replications of your database for applications running in other regions.

### 2. *Offload Analytics*

You can replicate data to the cloud and run analytics on your cloud databases rather than the original database that users interact with.

### 3. *Keeping Data in Sync*

Sometimes you need to keep your data in sync between testing, staging, and production environments.

**EXAM TIP:**

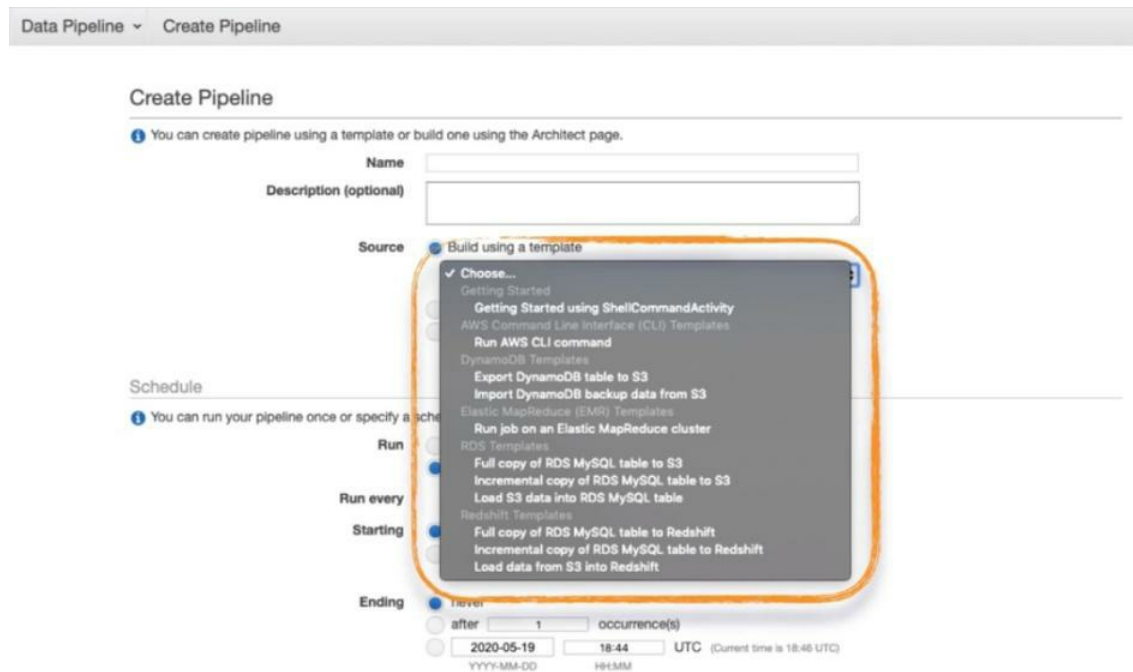
- Database Migration Service – managed service for migrating databases
- When to use DMS – migrations, upgrade, achieving data and replications
- Supported Migrations – heterogenous and homogenous migrations
- Replications - Cross-Region Replication, Offloading Analytics, and Keeping Data in Sync

## Data Pipeline

Data Pipeline automates the movement and transformation of your data. It helps you process and move data between AWS compute storage services and on-premises data sources. You can create an ETL workflow to automate the processing and movement of data at scheduled intervals.

### Creating Data Pipeline

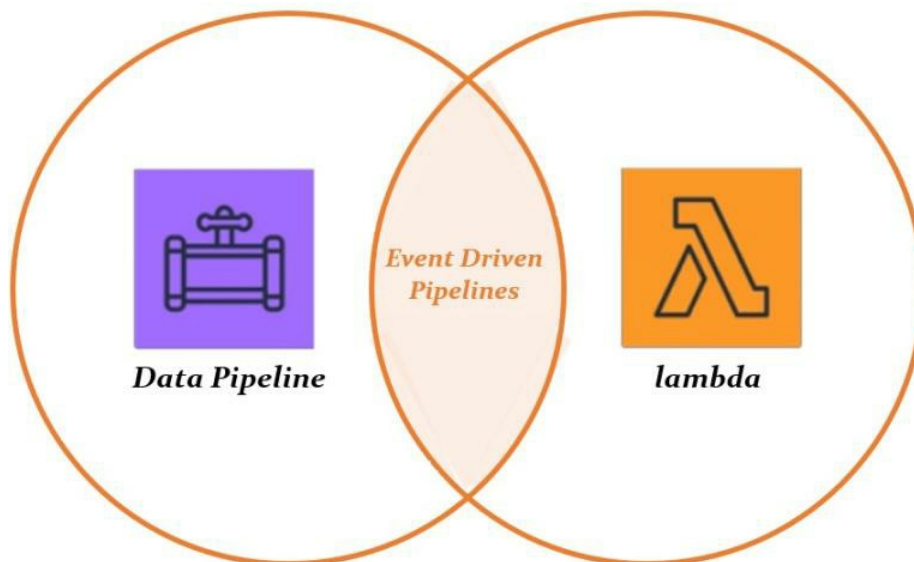
Whenever we create a Data Pipeline, we have several templates that can be chosen. We can create our custom templates and move data from any input source to any output source and transform it before it lands on its final destination.



*Figure 5-08: Creating Data Pipeline*

## Some Overlap With Lambda

There are several areas where the event-driven pipelines are overlapped between Data Pipeline and Lambda.



*Figure 5-09: Some Overlap With Lambda*

## Key Concepts

Data Pipeline is a container that consists of four parts that include data nodes, activities, Precondition, and schedules. Data nodes define where the data is coming from; this can be DynamoDB, S3, RDS instance or some on-premises database, or from Redshift. Activities are a way for the components of the Pipeline to define work to perform. Preconditions are conditional statements that must be true before the activity is run.

In schedules, we set up when we want the Data Pipeline to run AWS provisions and terminates the EC2 instances or the EMR clusters that do the transformation and the processing and moving the data from one source to another. When it is finished, it terminates automatically.

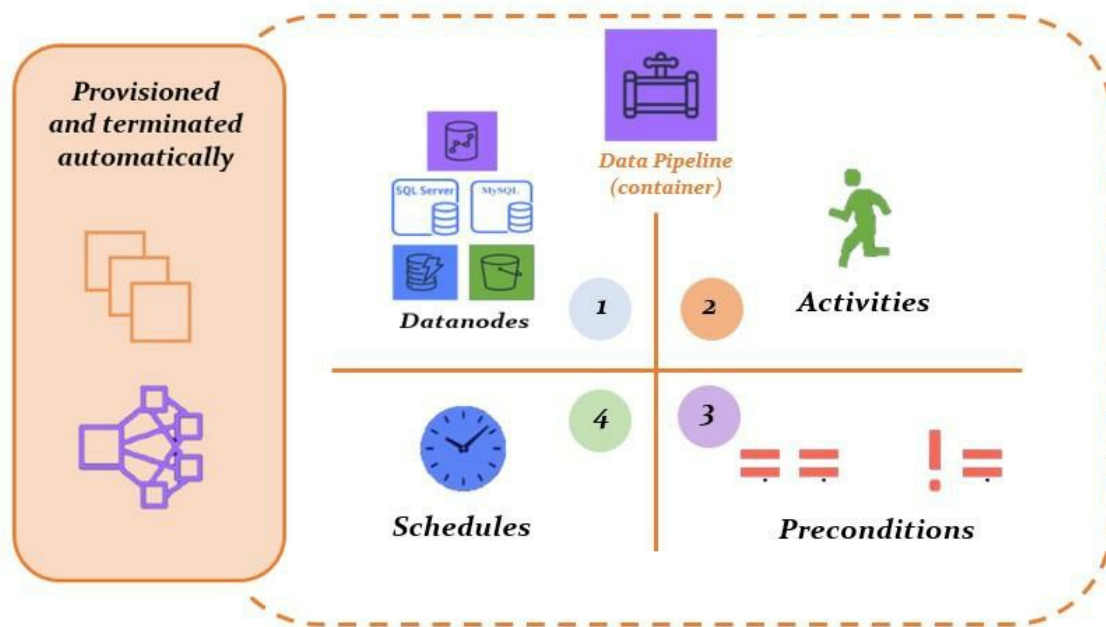


Figure 5-10: Key Concepts

1. The end destination of your data.
2. An action that Data Pipeline initiates on your behalf as part of the Pipeline.
3. The preconditions that must be met before the activity is run:
  - DynamoDBDdataExists

- DynamoDBTableExists
  - S3KeyExists
  - S3PrefixExists
  - ShellCommandPrecondition
4. Defines when your activity run and the frequency with which the services expect your data to be available.

## Data Pipeline for On-premises

We can run Data Pipeline for on-premises by installing the task runner on a server in our local network. We can access the local database securely and pull the Data Pipeline for the next task to run. When it recognizes a task to run, it runs that task on the task runner installed on-premises.

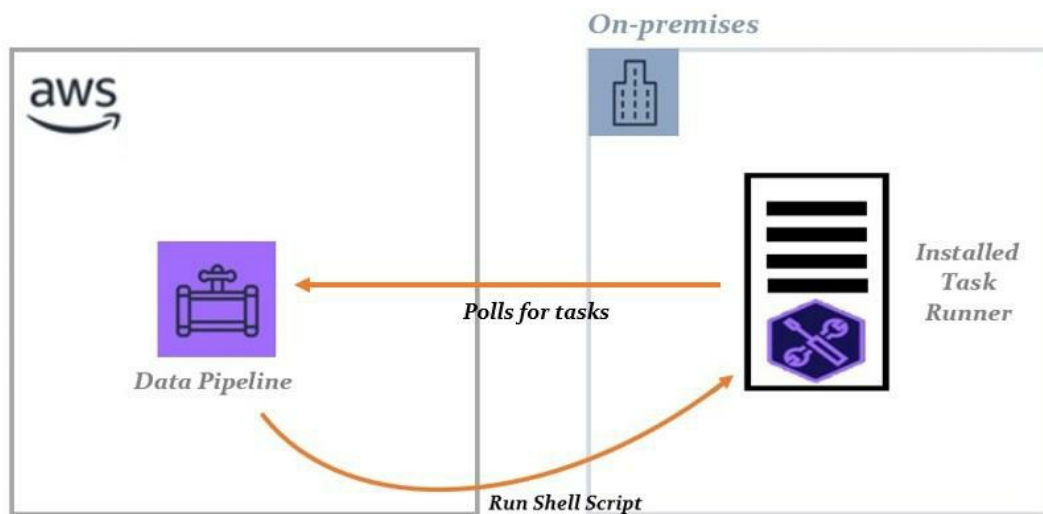


Figure 5-11: Data Pipeline for On-premise

## Data Pipeline for On-premise - Use Cases

*On-premise Database for Analytical and BI Tools*

Example: Move tables between the production database running on RDS and a non-production MySQL database running on-premises for BI purposes.

*Remotely executing stored procedures*

Example: Utilize stored procedures and run them on your scheduled tasks for your on-premises databases.

**EXAM TIP:**

- Data Pipeline - Automate the movement and transformation of your data
- Key Concepts – Datanodes, activities, preconditions, schedules
- Using Pipeline for On-premise – Installing a task runner that polls data Pipeline
- Use Cases – Moving data in databases and remotely executing stored procedures

## **Lambda, API Gateway, and CloudFront**

### **Definitions**

#### ***Lambda***

An event-driven service that allows you to run your code in AWS without managing infrastructure.

#### ***API Gateway***

A serverless API that can be used to create RESTful, HTTP, and WebSocket APIs.

#### ***CloudFront***

A content delivery network that allows you to deliver data, videos, applications, and APIs with low latency.

### **Lambda Events and Integration**

We have event data that comes out of the actual event that triggers the Lambda function. The event data is passed into the Lambda function, and then the Lambda does some type of processing. Then results come out of that Lambda function, and these results either land on some destination, or it can trigger another event. After each of these events,

either a new event is triggered or lands on some final destination. Many different things can trigger an event to talk to Lambda, like S3 events. When a new file is added or when a new file is removed, things like DynamoDB. When a new row in the DynamoDB table is added or deleted, any of those events occur. When streaming data is coming through Kinesis, we can take that data and do some processing. Before the data lands into S3 or on our final destination, we can do some type of processing. You can also trigger events through Redshift whether a new table is created or new data comes into Redshift and AWS IoT. AWS IoT allows us to manage IoT devices that are spread out throughout the edge. We can also have Elasticsearch. Elasticsearch also triggers an event, and the same with a data pipeline.

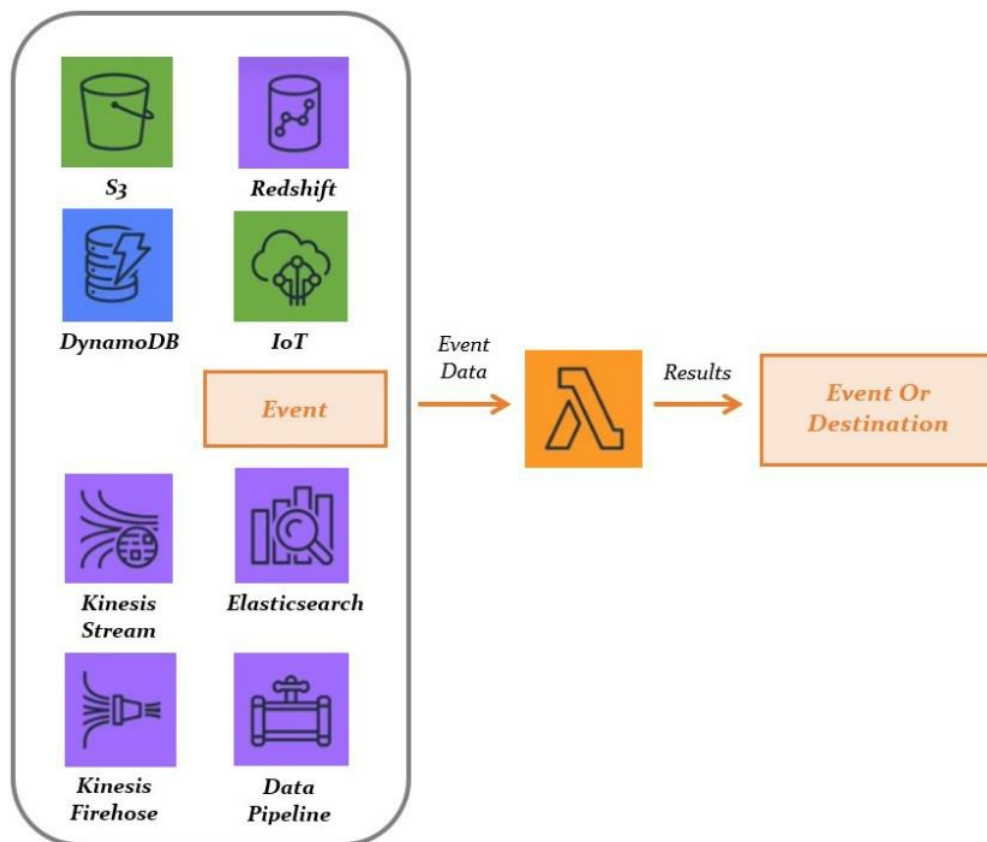


Figure 5-12: Lambda Events and Integration

## Lambda – Use Cases

### *Real-time Log Processing*

Example: You have a fleet of servers that are continuously creating logs. Use Lambda and Kinesis to capture, process, transform, and store those logs for real-time applications and notifications.

### *Creating ETL Pipelines*

Example: Trigger Lambda functions from various events and processes the incoming data. Once the data is transformed, you can use Lambda to load that data into the datastore.

### *Creating Cron Jobs*

Example: Set up Lambda functions to start daily batch jobs that are currently being done manually.

## Lambda Limits

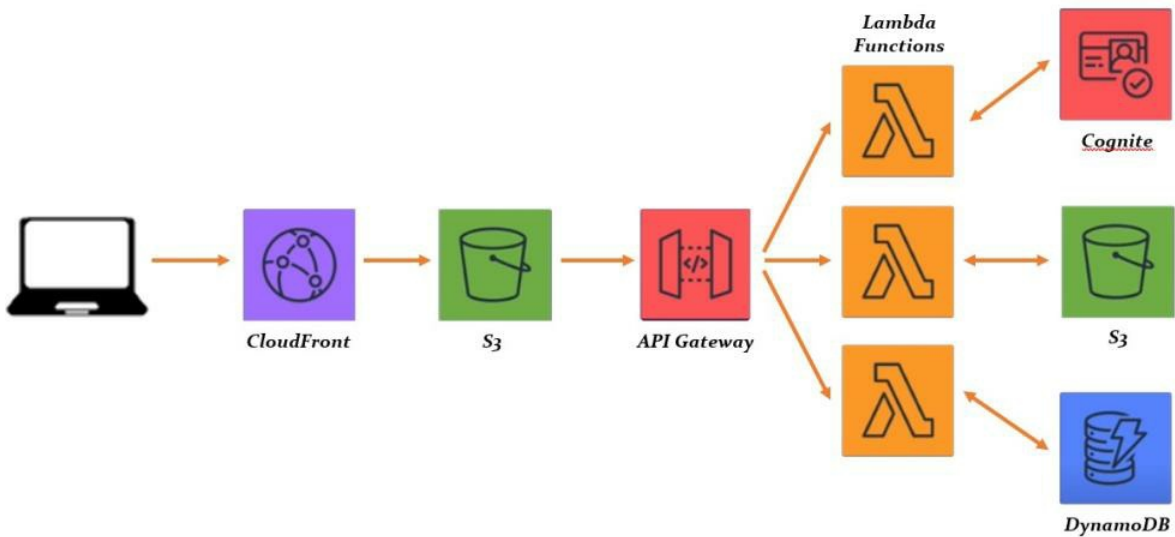
| Resource  | Limit   |
|---|---|
| Function timeout                                      | 900 seconds (19 minutes)  |
| Invocation payload (request and response)             | 6 MB (Synchronous)<br>256 KB (Asynchronous)   |
| Invocation frequency per region (requests per second) | 10 x concurrent executions<br>(Synchronous – all sources)<br>(Asynchronous – non-AWS sources)<br><br>Unlimited (Asynchronous – AWS) |

|  |                  |
|--|------------------|
|  | service sources) |
|--|------------------|

*Table 5-03: Lambda Limits*

# Serverless Architectures

These services in action can trigger things like Lambda functions. You can have an application that runs on CloudFront that gets the data from S3 or is served out through S3 via edge locations. It can trigger API Gateway endpoints that, in turn, trigger Lambda functions. These Lambda functions can communicate with Cognito. You can use Cognito for the login and authentication services. In addition, you can communicate back and forth to S3 and the same with DynamoDB. These architectures are pretty much endless to trigger endpoints through API Gateway, communicate with Lambda functions, do things like login users, get data back and forth from S3, and create data stores within DynamoDB.

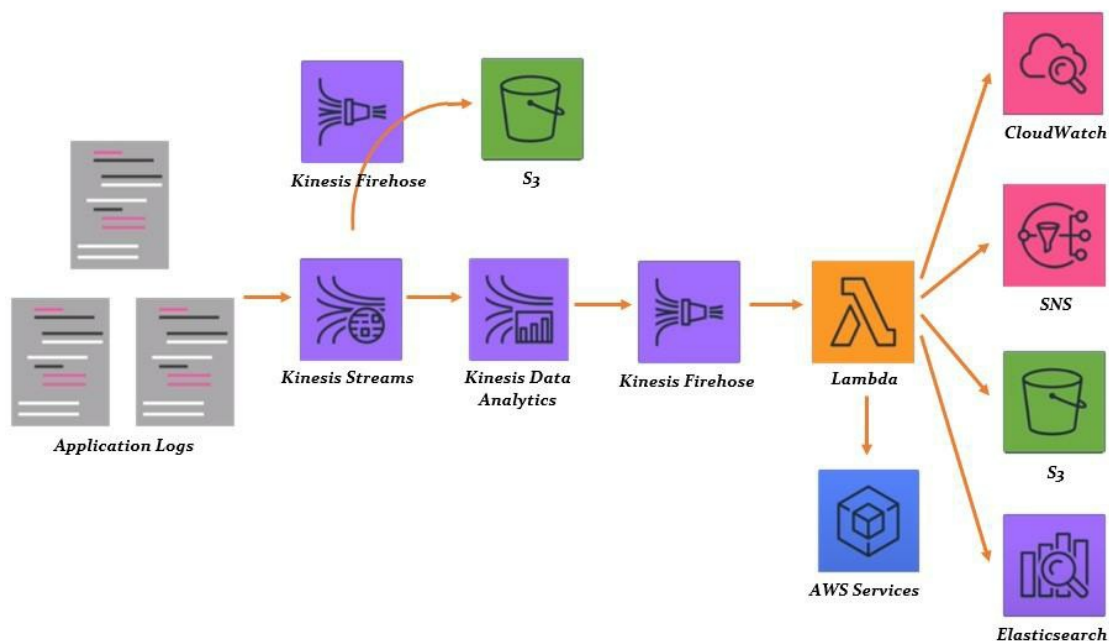


*Figure 5-13: Serverless Architecture*

Assume you have some application logs, and you want to stream those

into Kinesis streams. You can stream that Data into Kinesis streams that use Firehose to store it off into S3. Then you can assemble all your logs into a single location. You can use things like Kinesis data analytics to run SQL queries on those log files and set up different triggers or notifications if certain thresholds are hit.

Then, you can use Kinesis Data Firehose to trigger a Lambda function that communicates with CloudWatch or a simple notification service to trigger notifications. You can store that data off into S3 and perform the transformation, and set up ETL pipelines to finally store that data off into S3. Also, you can store that data on Elasticsearch, Redshift, and DynamoDB. In addition, you can trigger more events and spin up more AWS services. Furthermore, you can pass that data through or have that data land in some final destination.



*Figure 5-14: Serverless Architecture*

## Kinesis and Lambda Integrations

If we have some data producers and we are streaming that Data into Kinesis streams, we can trigger an event with Lambda. Consider some data transformation, or consider we want to have it delivered to DynamoDB. We can set up a Lambda function that does that for us. Lambda will trigger a function to fire for that particular shard. We can have multiple Lambda functions associated with a Kinesis stream event to have up to five.

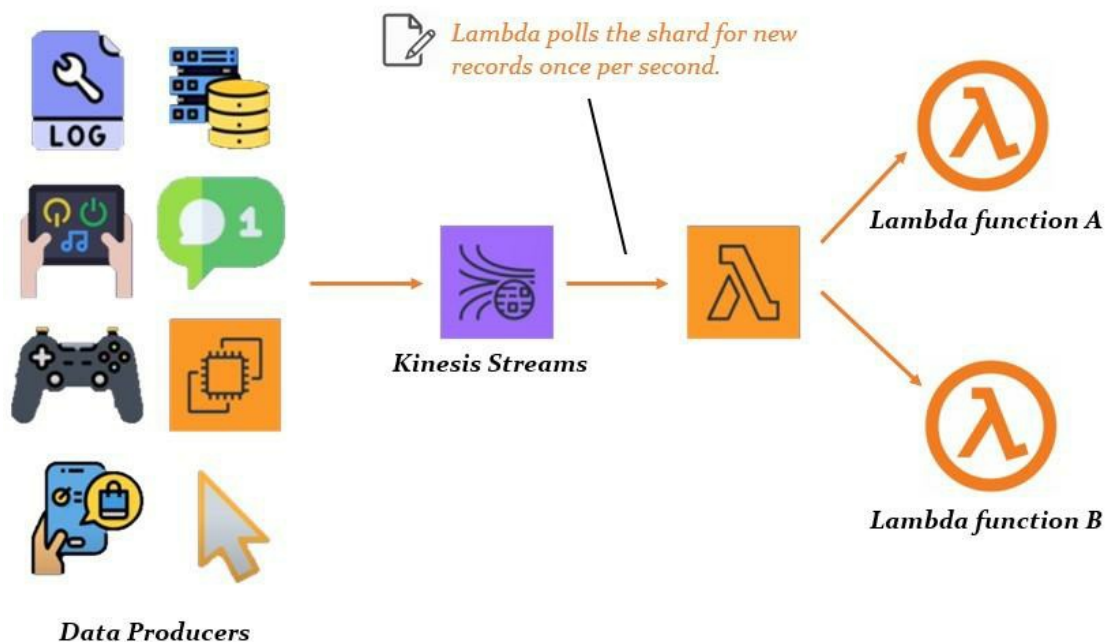


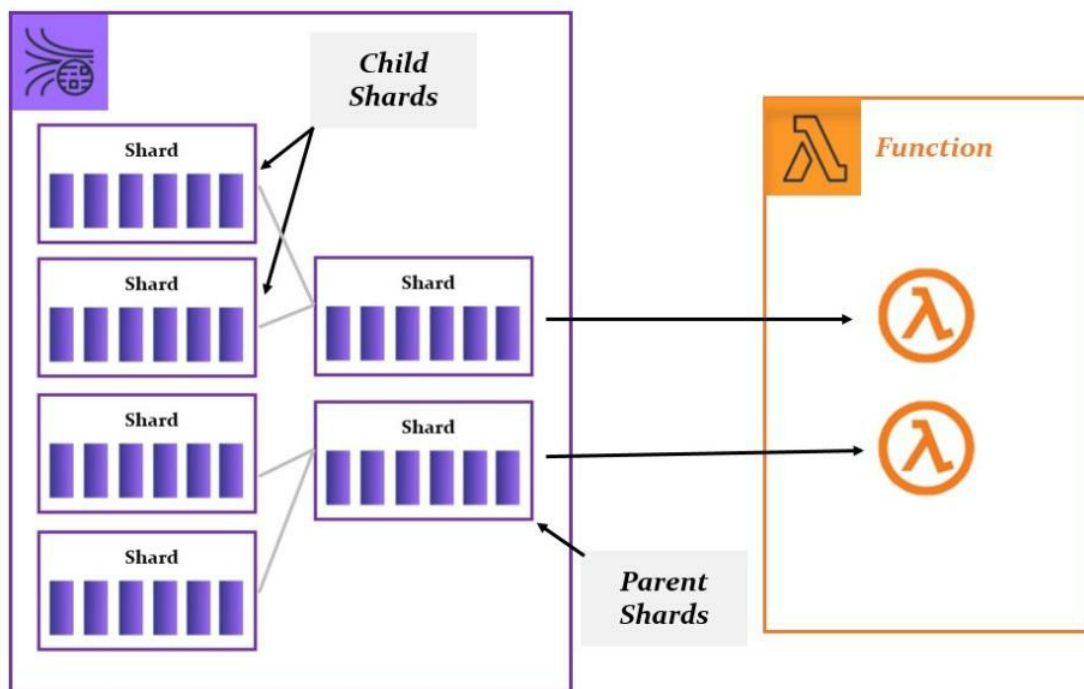
Figure 5-15: Kinesis and Lambda Integrations

## Kinesis and Lambda Scaling

When there is a high traffic volume, we want to process the records as fast as possible. Since only one Lambda function invocation happens at a time, we need to figure out a way to scale up our Kinesis stream to handle more data. We can do this by creating more shards. That is how we scale up our Kinesis data stream when more data starts to come in. For every shard, a single Lambda function is triggered. Therefore, we

will have multiple Lambda invocations that occur. The way that we scale up our shards is by using the update shard count API call. Another way to scale the Kinesis data stream is by splitting our shards. It creates two new child shards by splitting the partition keyspace of the parent shard.

We have two parent shards, and we have four child shards. Lambda will not start receiving any records from the child shards until all of the records have been processed from the parent shard. Therefore, we have our parents' shards, splitting them into separate child shards. Lambda invocations are not going to occur until all of the records are out of the parent shards. Then it will start processing the child shard records.



*Figure 5-16: Kinesis and Lambda Scaling*

#### EXAM TIP:

- Lambda Limits – Function timeout, invocation payload, invocation frequency
- Serverless Architectures – using these services together can create serverless architectures

- Lambda and Kinesis – Integrating Kinesis and Lambda and you can scale up Kinesis by using parent and child shards.

## Comparing our Options

### Time Required To Move Data Into AWS

If we have a small amount of data, consider 1 terabyte, 10 terabytes, 100 terabytes when moving petabytes of data; depending on your network bandwidth will determine how fast that data can be moved into AWS. As your data grows, you can start to see how long it would take to move an increasing amount of data depending on your network speed. A petabyte or 10 petabytes of data with a hundred megabits per second network bandwidth or one gigabit per second could take years to move.

| Network Bandwidth |        |          |          |            |
|-------------------|--------|----------|----------|------------|
| Amount of Data    |        | 100 Mbps | 1 Gbps   | 10 Gbps    |
|                   | 1 TB   | 30 hours | 3 hours  | 18 minutes |
|                   | 10 TB  | 12 days  | 30 hours | 3 hours    |
|                   | 100 TB | 124 days | 12 days  | 30 hours   |
|                   | 1 PB   | 3 years  | 124 days | 12 days    |
|                   | 10 PB  | 34 years | 3 years  | 124 days   |

Figure 5-17: How Long Will It Take

### Choosing A Service

We can use AWS Data Sync, which is great for the rapid transferring of data. We can schedule regular intervals for the data to be moved into AWS. If we want to transfer data over long distances into S3, we can utilize S3 Transfer Acceleration. This is the go-to service for heterogeneous migrations and homogenous migrations from one database platform to another if we need to migrate databases. Data Firehose is great at taking your streaming data and loading it into its

final destination, whether S3, Redshift, Splunk instances, or Elasticsearch. We can transform the data before it lands on its final destination using AWS Lambda. Offline data migrations are where Snowball, Snowball Edge, and Snowmobile come into play. We can load our data onto a physical device, ship it back to AWS, and then with their network capabilities; they just load that data directly into S3 for us.

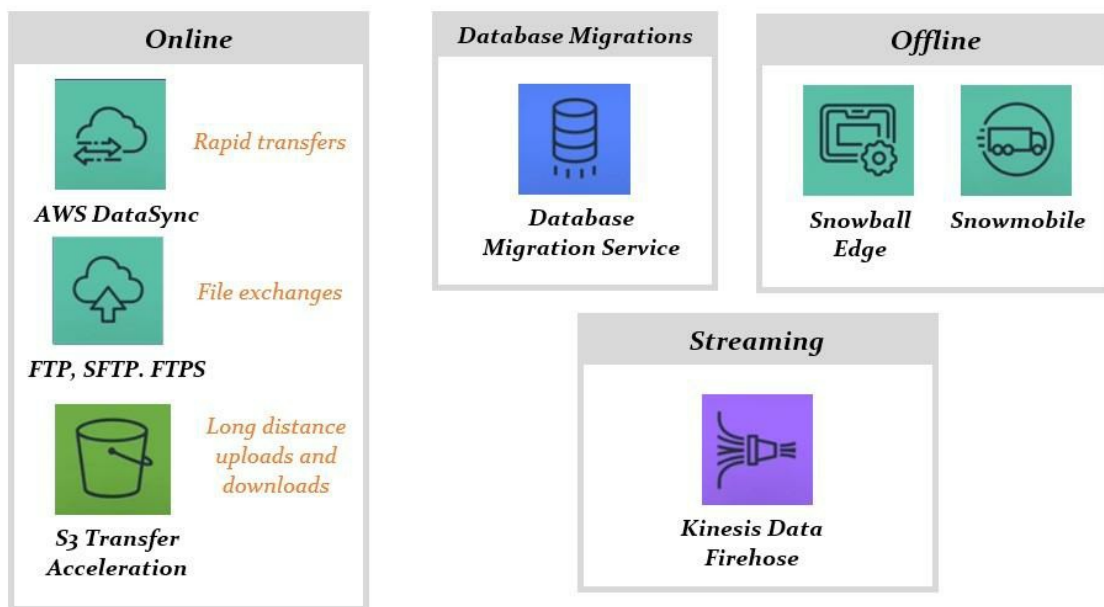
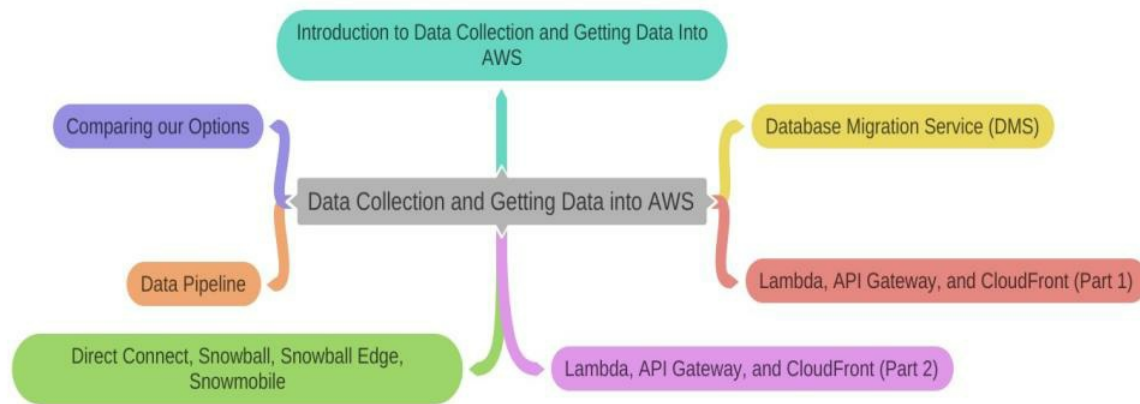


Figure 5-18: Choosing a Service

## Mind Map



*Figure 5-19: Mind Map*

## Practice Questions

1. Which of the following makes it simple and easy to transfer your data into and out of AWS via online methods?
  - A. Online Data Transfer
  - B. Offline Data Transfer
  - C. Both of them
  - D. None of them
2. Which of the following is used to collect, process, and move data to AWS online with AWS DataSync?
  - A. Snowmobile
  - B. Snowcone
  - C. Snowball
  - D. Snowball Edge
3. Which of the following is used for petabyte-scale data transport with import and export to S3?
  - A. Snowmobile
  - B. Snowcone
  - C. Snowball
  - D. Snowball Edge
4. Which of the following is an exabyte-scale data transport solution that uses a secure semi-40-foot shipping container to transfer large amounts of data into and out of AWS?
  - A. Snowcone

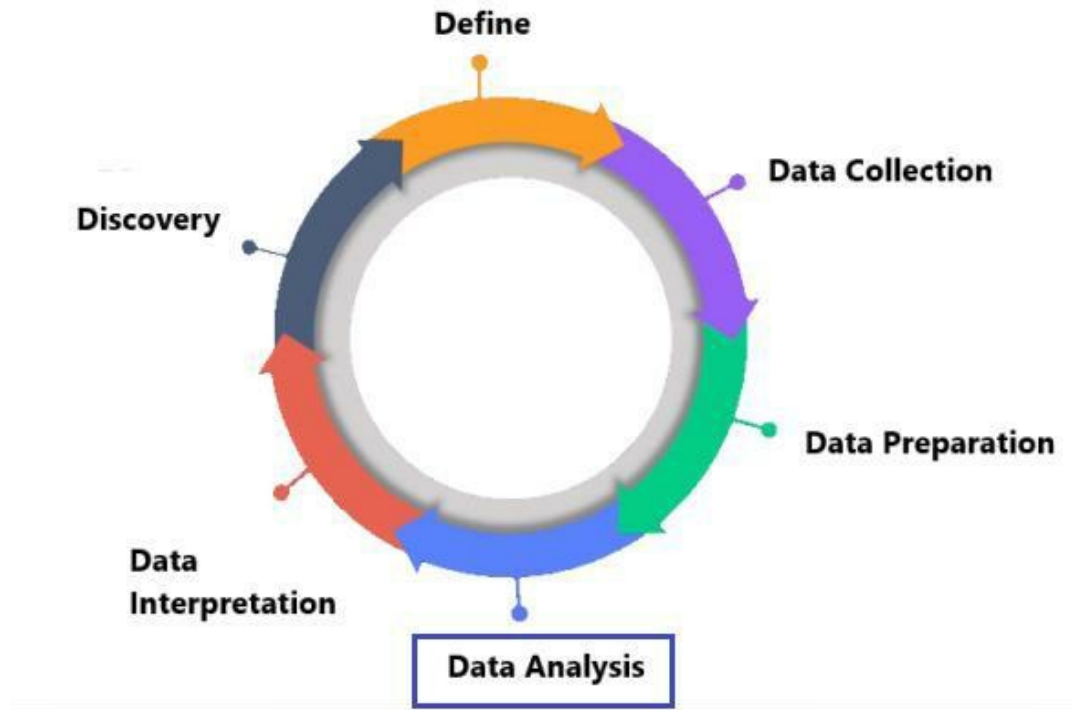
- B. Snowball
  - C. Snowball Edge
  - D. Snowmobile
5. Which of the following is a local storage and large-scale data transfer, local Lambda and EC2 instances compute, and AWS IoT Greengrass?
- A. Snowcone
  - B. Snowball Edge
  - C. Snowball
  - D. None of them
6. The Snow Family includes -----.
- A. Snowball
  - B. Snowball Edge
  - C. Snowmobile
  - D. All of them
7. AWS Glue Version 0.9 and 0.1 are billed in ----- increments with a 10-minute minimum.
- A. 1-second
  - B. 0.1-second
  - C. 0.01-second
  - D. None of them
8. Direct Connect Uses ----- Gbps dedicated networking.
- A. 1 Gbps to 10 Gbps
  - B. 0.1 Gbps to 10 Gbps
  - C. 1 Gbps to 100 Gbps
  - D. 0.1 Gbps to 100 Gbps
9. Which service easily and securely migrates widely used commercial and open-source databases and data warehouses into the cloud?
- A. CloudWatch
  - B. Database Migration Service
  - C. AWS Glue jobs
  - D. None of them
10. Replication easily replicates your databases and data warehouse between how many locations?
- A. 1
  - B. 2
  - C. 3
  - D. None of them

11. Which of the following gives you the ability to create cross-region replications of your database for applications running in other regions?
  - A. Offload Analytics
  - B. Keeping Data in Sync
  - C. Cross-Region Replication
  - D. None of them
  
12. With the help of -----, you can replicate data to the cloud and run analytics on your cloud databases rather than the original database that users interact with.
  - A. Offload Analytics
  - B. Keeping Data in Sync
  - C. Cross-Region Replication
  - D. None of them
  
13. When should you use DMS?
  - A. Migrations
  - B. Upgrade
  - C. Achieving data and replications
  - D. All of them
  
14. Snowball device is used to -----.
  - A. Store 80 TB storage, 10 GB network
  - B. User interface similar to S3
  - C. All data is encrypted end-to-end.
  - D. All of them
  
15. Supported Migrations include -----.
  - A. Heterogenous migrations
  - B. Homogenous migrations
  - C. Both A and B
  - D. None of them

# CHAPTER 06: AMAZON ELASTIC MAP REDUCE (EMR)

## Introduction

Elastic Map Reduce or EMR plays a huge role in data analytics, processing, and big data frameworks. We can use the EMR architecture and the Hadoop framework to process and analyze massive amounts of data. Log analysis, web indexing, data warehousing, machine learning (ML), financial analysis, scientific modeling, and bioinformatics all use Amazon EMR for data analysis. It also supports Apache Spark, Apache Hive, Presto, and Apache HBase workloads, connecting with Hive and Pig, free source Hadoop data warehousing technologies. Pig provides a high-level interface for scripting Map-Reduce tasks in Hadoop, whereas Hive utilizes queries and analyses. This service is expensive, and it uses a lot of computing power, but it gives you the ability to process huge amounts of data in a short amount of time.



*Figure 6-01: Data Analysis with EMR*

## Apache Hadoop and EMR Software Collection

### Map Reduce

Map-reduce is a technique that data scientists can use to distribute workloads across many different computing nodes to process other data and get the information back quicker than just on a single node.

Let us take a look at an example of Map reduce. Imagine if someone shows you the image given below and asks you the question, Can you tell me how many Google cloud and AWS icons there are? Disregard any of the Azure icons. If you take a few minutes and count them, you could tell the answer, but let us imagine that there were hundreds of thousands of these. Your eye would not be able to determine how many there are. Hence, we could create a Map reduce function.

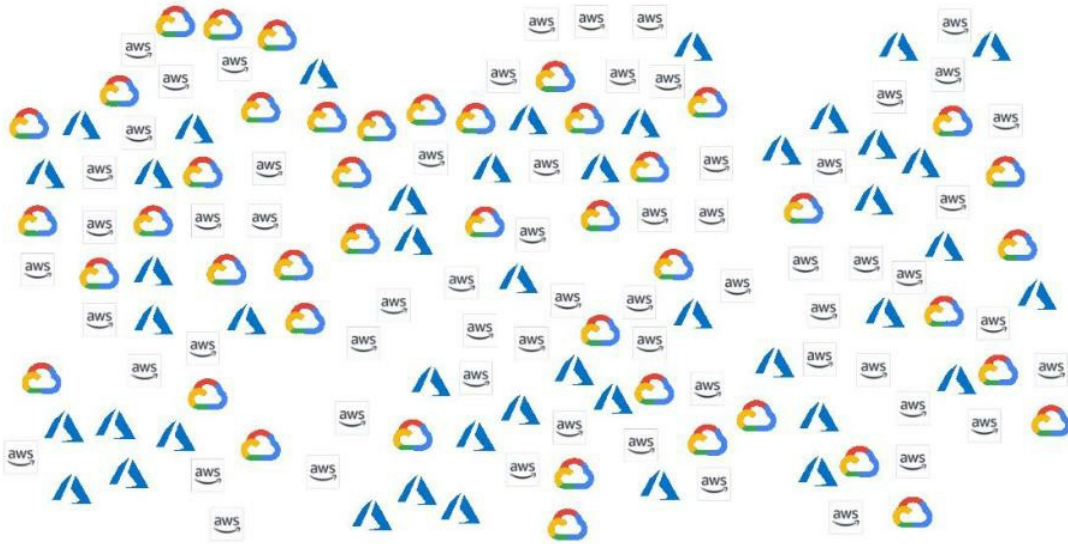


Figure 6-02: Counting AWS and GCP icons

Our Map reduce function will catch all AWS icons and the GCP icons but disregard any Azure icons.

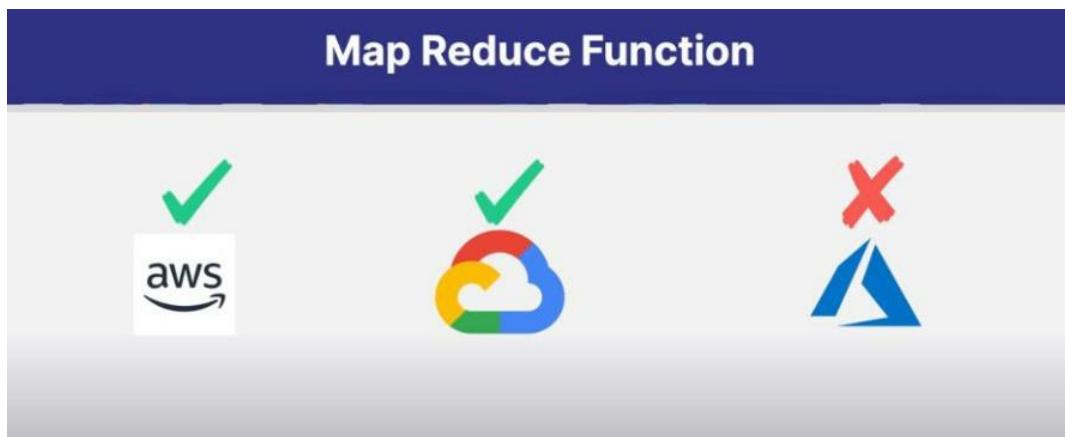
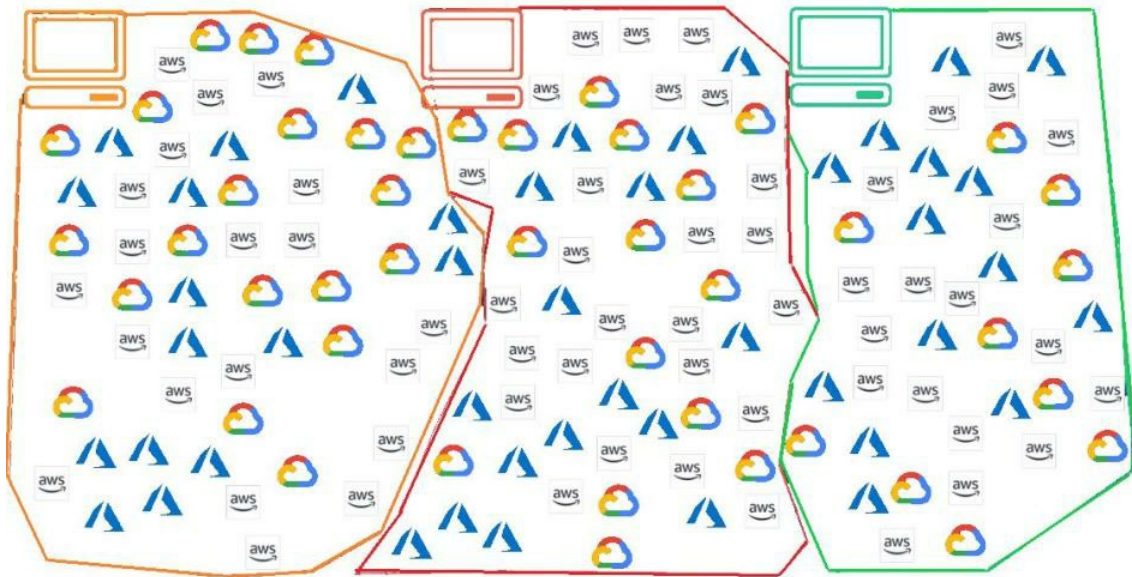


Figure 6-03: Map-Reduce Function

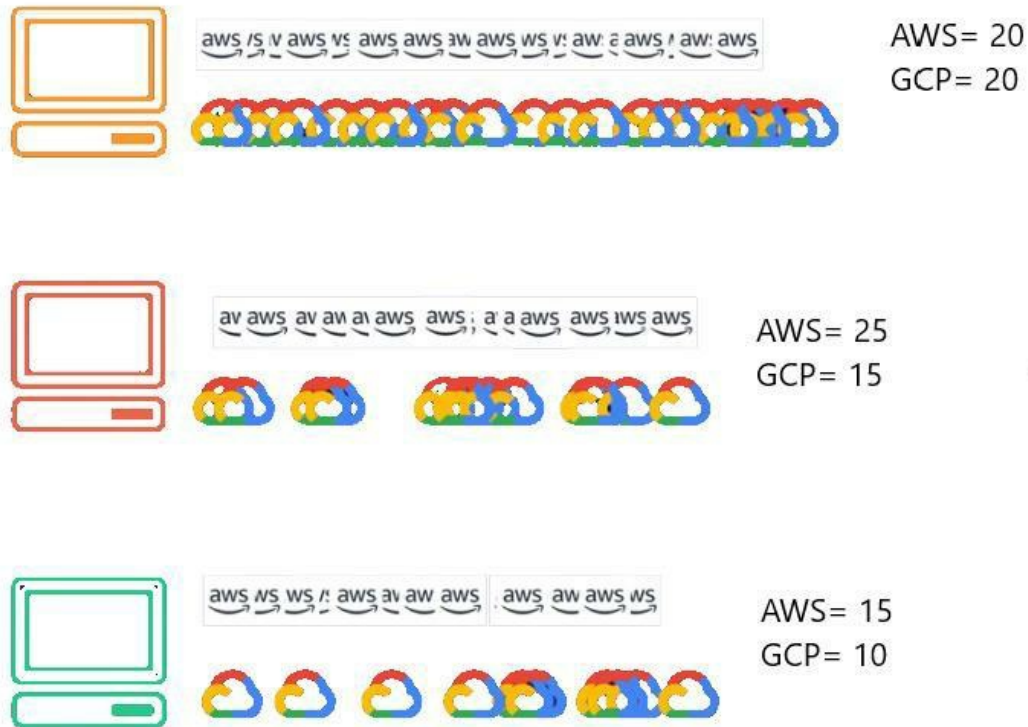
With a Map reduce function; the workload is distributed across many different nodes. The orange node is going to process some part of the information. The red node will process some other part of the information, and the green node will process the remaining data. In Map-reduce, the workload is split evenly across these nodes. Therefore, anyone can use the processing power individually. It can be scaled to

hundreds and thousands of nodes if needed.



*Figure 6-04: Nodes*

Once the orchestration of breaking the data apart has occurred, each of these nodes will apply their map phase, and each node counts the various icons. It is known as the map phase.



*Figure 6-05: Map Phase*

Once the map phase is complete, then the reduce phase happens. In this case, all of the data is assembled on a single node where reduce determines whatever needs to be processed. In this case, it counts the number of icons.



Figure 6-06: Reduce Phase

Now you can see automatically AWS has 50 and GCP has 45 icons. Hence, if you knew that picture of all the icons, you would not have been able to tell that just right off the bat. This is known as the reduce phase.

Map reduce means breaking apart a processing task over many different nodes, having a map phase where some processing, counting, filtering, or aggregation happens, and then the reduce phases where all of that data is assembled onto a single node. It can be multiple nodes, but in the case of EMR, it is built on a single node, and the output of our Map-reduce function is determined.

## Distributed File Systems

Imagine that you have two huge files with tons and tons of data. You can imagine these being gigabyte files or many megabyte files within a distributed file system. You have a primary node, also known as the master node.

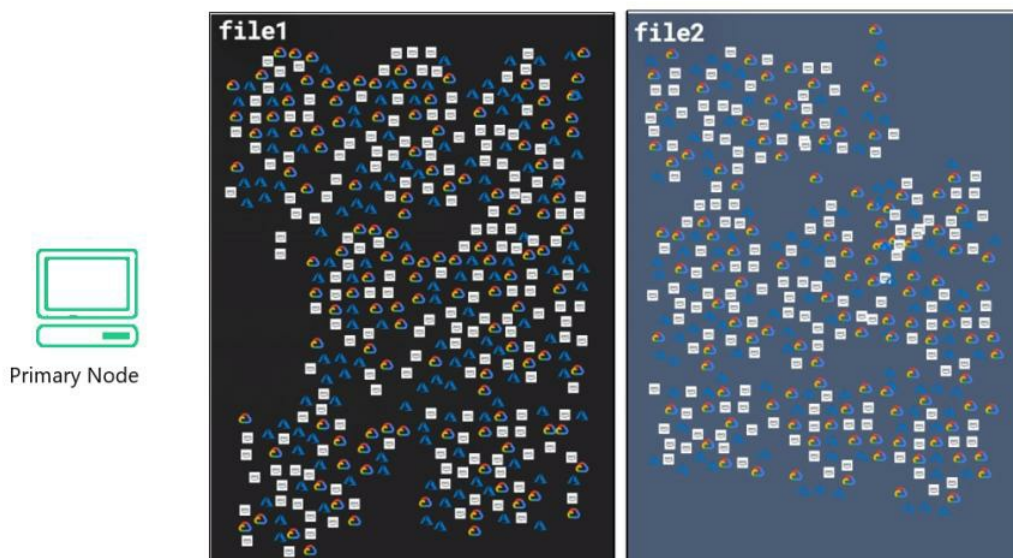


Figure 6-07: Distributed File System (i)

These files are broken up on a distributed file system, but they look like single files on the primary node. Hence, imagine that we have some secondary nodes or slave nodes. These files are split up across the secondary nodes, and not only are they broken up across the secondary nodes, but they are also replicated. That means that each secondary nodes have multiple replication or copies of these broken-up blocks or broken-up files. It allows us to lose one of the secondary nodes, but we still have a copy of the data. It is what makes distributed file systems so resilient and powerful.

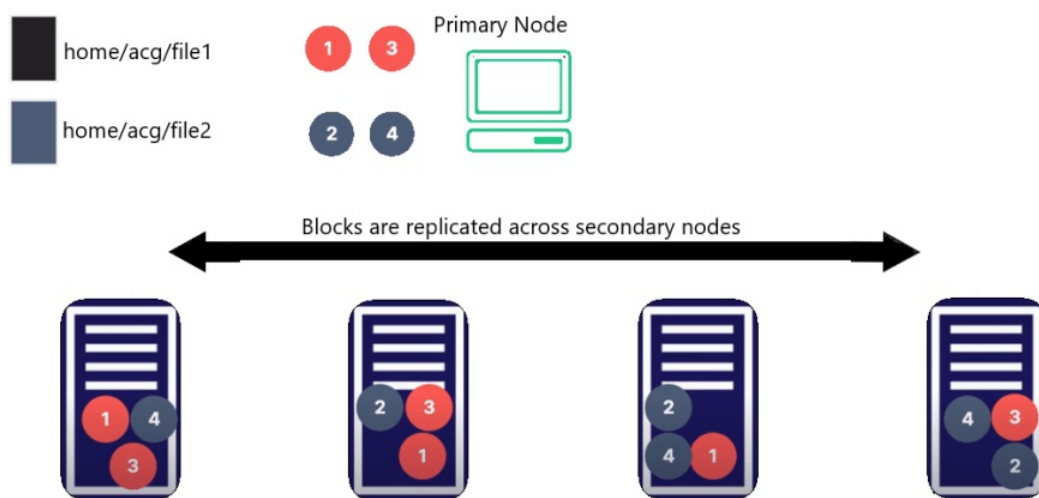


Figure 6-08: Distributed File System (ii)

## Hadoop Distributed File System (HDFS)

Hadoop Distributed File System is open-source software that allows you to operate a distributed file system over several computers to tackle challenges requiring large amounts of data. HDFS is meant to run on low-cost hardware and is extremely fault-tolerant. HDFS is a file system that allows high-throughput access to application data and is well

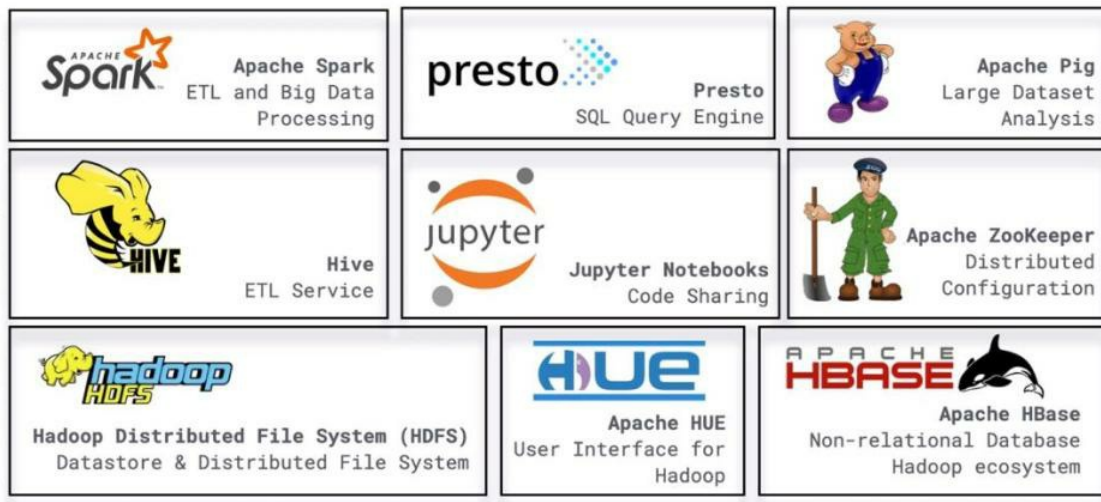
suited to applications with huge data collections. The problem with setting up an HDFS cluster so it requires a lot of maintenance and management. This is where Elastic Map reduce comes in.

## **EMR**

Elastic Map reduce is a fully managed AWS service that allows you to spin up Hadoop ecosystems. Not only can you store data on HDFS, but you have some other storage options as well. We also have the EMR file system or EMRFS. This means that the Elastic Map reduce cluster shares the data with S3. We also can store it on the local file system. This can be an instance store or on EBS volumes.

Hadoop is not the only piece of software that runs on EMR. There is a plethora of applications and open-source software tools that you can pre-install or install and configure yourself on EMR. You want Presto, Jupiter Notebooks, or Hue installed. When we say install and configure yourself, that means just checking a checkbox. You check a checkbox, and EMR takes care of the installation. They also take care of the maintenance. They make sure that it is up to date, and they ensure that all of the software systems can communicate with one another.

Each of these software systems does different things. Typically, Apache Spark is for ETL and big data processing. Presto is a SQL query engine that we can use. We also have Jupiter notebooks to share code or write Apache Spark programs right within the EMR cluster. Apache Pig is for large data analysis. We also have Zookeeper, HBase Hue, and the Hadoop distributed file system. These are not the only software that can be installed onto EMR. There are tons of them, and whenever you create your cluster, you choose what you want to install.



*Figure 6-09: EMR Software Collection*

**Note:** Hive is another ETL service.

## Quick Options

If we look at what it looks like in the console, we have to give our cluster name, and we point to an S3 folder where we want to have the logging.

aws Services Resource Groups

Create Cluster - Quick Options [Go to advanced options](#)

### General Configuration

Cluster name: My cluster

☒ Logging ⓘ

S3 folder: s3://aws-logs-174570359254-us-east-2/elasticmapreduce/ ⓘ

Launch mode: ☒ Cluster ⓘ ☐ Step execution ⓘ

### Software configuration

Release: emr-5.28.0 ⓘ

Applications:

- ☒ Core Hadoop: Hadoop 2.8.5 with Ganglia 3.7.2, Hive 2.3.6, Hue 4.4.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.9.2
- ☐ HBase: HBase 1.4.10 with Ganglia 3.7.2, Hadoop 2.8.5, Hive 2.3.6, Hue 4.4.0, Phoenix 4.14.3, and ZooKeeper 3.4.14
- ☐ Presto: Presto 0.227 with Hadoop 2.8.5 HDFS and Hive 2.3.6 Metastore
- ☐ Spark: Spark 2.4.4 on Hadoop 2.8.5 YARN with Ganglia 3.7.2 and Zeppelin 0.8.2

☐ Use AWS Glue Data Catalog for table metadata ⓘ

*Figure 6-10: General Configuration*

We then choose the software configuration. These configurations are constantly updating the versions of the software as well.

aws Services Resource Groups

Create Cluster - Quick Options [Go to advanced options](#)

### General Configuration

Cluster name: My cluster

☒ Logging ⓘ

S3 folder: s3://aws-logs-174570359254-us-east-2/elasticmapreduce/ ⓘ

Launch mode: ☒ Cluster ⓘ ☐ Step execution ⓘ

### Software configuration

Release: emr-5.28.0 ⓘ

Applications:

- ☒ Core Hadoop: Hadoop 2.8.5 with Ganglia 3.7.2, Hive 2.3.6, Hue 4.4.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.9.2
- ☐ HBase: HBase 1.4.10 with Ganglia 3.7.2, Hadoop 2.8.5, Hive 2.3.6, Hue 4.4.0, Phoenix 4.14.3, and ZooKeeper 3.4.14
- ☐ Presto: Presto 0.227 with Hadoop 2.8.5 HDFS and Hive 2.3.6 Metastore
- ☐ Spark: Spark 2.4.4 on Hadoop 2.8.5 YARN with Ganglia 3.7.2 and Zeppelin 0.8.2

☐ Use AWS Glue Data Catalog for table metadata ⓘ

*Figure 6-11: Software Configuration*

We then tell it how many nodes we want to have running the size. In addition, what type of EC2 instance that we want to have running. We can configure it to be specific for each primary node and each type of secondary node.

The screenshot displays the AWS EMR console configuration page. The 'Hardware configuration' section is highlighted with a red box and contains the following settings: 'Instance type' set to 'm3.xlarge' and 'Number of instances' set to '3' with a subtext '(1 master and 2 core nodes)'. Below this, the 'Security and access' section is visible, showing 'EC2 key pair' as 'aws\_key', 'Permissions' set to 'Default', 'EMR role' as 'EMR\_DefaultRole', and 'EC2 instance profile' as 'EMR\_EC2\_DefaultRole'. At the bottom right, there are 'Cancel' and 'Create cluster' buttons. The footer shows '(US)' and copyright information '© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.'

*Figure 6-12: Hardware Configuration*

We can configure the security for this EMR cluster to access it via SSH or some other type of way. Finally, we have the security access where we just set up an SSH key if we want to SSH into the primary node, and we have some roles that need to be set up in IAM. Therefore, we can interact with Redshift, S3, DynamoDB, and any other services we want to interact with.

Hardware configuration

Instance type: m3.xlarge

Number of instances: 3 (1 master and 2 core nodes)

**Security and access**

EC2 key pair: aws\_key

Permissions: ☒ Default ☐ Custom

Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.

EMR role: EMR\_DefaultRole

EC2 instance profile: EMR\_EC2\_DefaultRole

Cancel Create cluster

(US) © 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.

*Figure 6-13: Security and Access*

## Advanced Options

We can choose which software we want to have installed onto our EMR cluster in the advanced options.

Create Cluster - Advanced Options [Go to quick options](#)

Step 1: Software and Steps

Step 2: Hardware

Step 3: General Cluster Settings

Step 4: Security

**Software Configuration**

Release: emr-5.28.0

|  |   |  |
|--|---|--|
| <input checked="" type="checkbox"/> Hadoop 2.8.5 | <input type="checkbox"/> Zeppelin 0.8.2 | <input type="checkbox"/> Livy 0.6.0            |
| <input type="checkbox"/> JupyterHub 1.0.0        | <input type="checkbox"/> Tez 0.9.2      | <input type="checkbox"/> Flink 1.9.0           |
| <input type="checkbox"/> Ganglia 3.7.2           | <input type="checkbox"/> HBase 1.4.10   | <input checked="" type="checkbox"/> Pig 0.17.0 |
| <input checked="" type="checkbox"/> Hive 2.3.6   | <input type="checkbox"/> Presto 0.227   | <input type="checkbox"/> ZooKeeper 3.4.14      |
| <input type="checkbox"/> MXNet 1.5.1             | <input type="checkbox"/> Sqoop 1.4.7    | <input type="checkbox"/> Mahout 0.13.0         |
| <input checked="" type="checkbox"/> Hue 4.4.0    | <input type="checkbox"/> Phoenix 4.14.3 | <input type="checkbox"/> Oozie 5.1.0           |
| <input type="checkbox"/> Spark 2.4.4             | <input type="checkbox"/> HCatalog 2.3.6 | <input type="checkbox"/> TensorFlow 1.14.0     |

Multiple master nodes (optional)

☐ Use multiple master nodes to improve cluster availability. [Learn more](#)

AWS Glue Data Catalog settings (optional)

☐ Use for Hive table metadata

Edit software settings

☒ Enter configuration ☐ Load JSON from S3

classification=config-file-name,properties=[myKey1=myValue1,myKey2=myValue2]

Steps (optional)

A step is a unit of work you submit to the cluster. For instance, a step might contain one or more Hadoop or Spark jobs. You can also submit additional steps to a cluster after it is running. [Learn more](#)

Feedback English (US) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Private

*Figure 6-14: Software Configuration*

As we are setting up our EMR cluster, we can also set up steps, which is

a process or a Map-reduce function, or an Apache Spark program, a Hive query, any Java jar that we want to run on the EMR cluster, you can run it on an EMR cluster. We can set up these steps to submit a unit of work to that cluster. It needs to accomplish some tasks. It needs to go through and see how many AWS icons there are, how many GCP icons there are, it needs to go through a terabyte of log files, it needs to filter through billions of customer reviews, or it needs to do some machine learning analysis on terabytes of brainwave data.

The screenshot shows the AWS EMR console interface. The 'Steps (optional)' section is highlighted with a red box. It contains the following elements:

- Steps (optional)** header.
- A description: "A step is a unit of work you submit to the cluster. For instance, a step might contain one or more Hadoop or Spark jobs. You can also submit additional steps to a cluster after it is running. [Learn more](#)"
- Concurrency:** ☐ Run multiple steps at the same time to improve cluster utilization
- After last step completes:** ☒ Clusters enters waiting state, ☐ Cluster auto-terminates
- Step type:** Select a step,

At the bottom of the console, there are 'Cancel' and 'Next' buttons.

*Figure 6-15: Steps (optional)*

#### **KAM TIP:**

- Map Reduce is the splitting, mapping, shuffling, and reducing big data to produce the desired output.
- Distributed File Systems are how large files are split and replicated across many nodes.
- HDFS is an open-source Apache Hadoop software that makes it easy

to run distributed file systems.

- EMR is a fully managed Hadoop cluster in AWS to store, analyze and process big data.
- EMR storage options include HDFS, EMR file system (EMRFS), and local file system.

## EMR Architecture

### Introduction

The entire cluster is spun up in a single availability zone. Every single EMR cluster has a primary node, or it can have three primary nodes. Hence, it is either a single primary node or three primary nodes. This primary node manages all of the components in the distributed applications. The core nodes come into play when a job needs to be submitted or some processing or Map reduce tasks. The primary node manages these core nodes. The last part of the EMR architecture is our task nodes. Task nodes are optional. They add power to perform parallel, computational tasks on the data, and they help the core nodes.

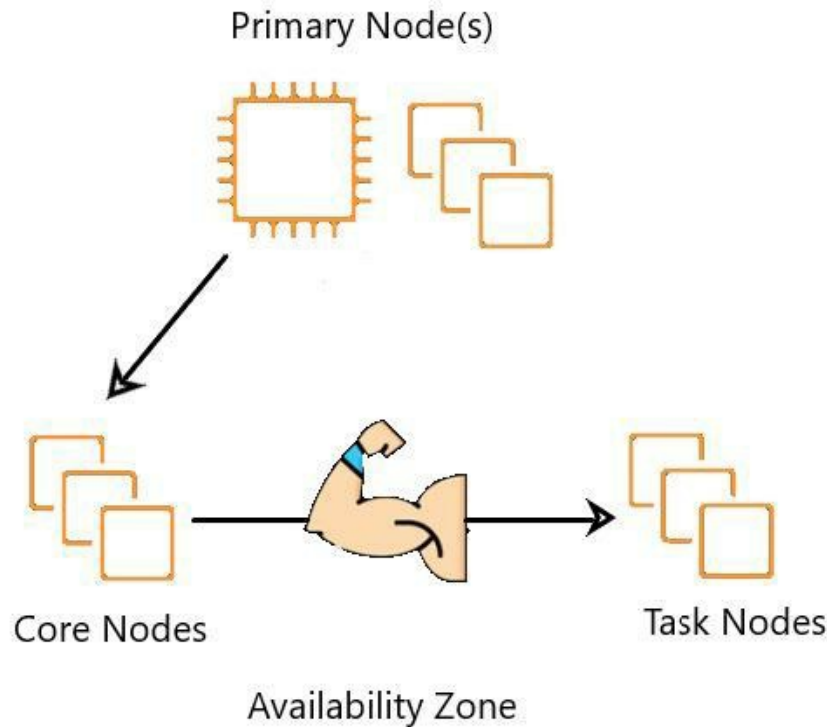


Figure 6-16: EMR Architecture

## Primary Node Features

- ***Single or Multi-Primary Nodes***

Whenever you launch a cluster, you will have the option to choose between one primary node and three primary nodes. You will only have a single primary node most of the time, but now you can also have multiple primary nodes. You would have numerous primary nodes because you do not have a single point of failure. Therefore, if one master node fails, the cluster uses the other two master nodes to run without interruptions. EMR automatically replaces the primary node and provisions it with any configurations or bootstrap actions that need to happen. Hence, all it does is remediate that single point of failure.

- ***Manages the Cluster Resources***

The primary node also manages the cluster resources. It coordinates the distribution of the parallel execution for the different Map reduce tasks.

- ***Tracks and Directs HDFS***

The primary node also tracks and directs the HDFS. The primary node knows how to lookup files and track data on the core nodes.

- ***YARN Resource Management***

The primary node is also responsible for the YARN resource management. EMR uses YARN (Yet Another Resource Negotiator) to manage cluster resources for multiple data-processing frameworks.

- ***Monitors Core and Task Nodes Health***

The primary node tracks the status of jobs submitted to the cluster and monitors the health of the core and task nodes.

## **Core Node Features**

- ***Run Tasks for the Primary Node***

The primary node manages core nodes and runs Hadoop Map reduce tasks, Hive Scripts, and Spark executors.

- ***Coordinates Data Storage***

The core node is also responsible for coordinating data storage. The core nodes know how and where to store the data. This data is stored on HDFS or EMRFS. The DataNode daemons run on the core node.

- ***Multiple Core Nodes, Only One Core Instance Group***

We can have multiple core nodes but only one core instance group. These multiple core nodes are made up of multiple EC2 instances. This makeup the instance in group or fleet from.

## **Task Node Features**

- ***Optional Helpers***

Task nodes are optional and can add power to perform parallel computation tasks on data like Map reduce tasks and Spark executor.

- ***No HDFS or DataNode Daemon***

Task nodes do not store data in HDFS. It is not used as a data store and does not run the Data Node daemon.

- ***Added and Removed from running clusters***

Task nodes can be added and removed from the core nodes to ramp up extra CPU or memory for compute-intensive tasks.

## **Single Availability Zone Concept**

The EMR clusters only reside in a single availability zone. The main reason behind the single availability zone concept so the nodes in the cluster can communicate faster. It means that they do not have to traverse as much internet or the AWS backbone. They are closer together, and they are in the same availability zone. It means block replication can happen more quickly. Hence, you can find your files faster when finding them on HDFS. In addition, the communication between nodes happens faster. Hence, whenever core nodes are processing back and forth, they can communicate faster. In addition, the access to metadata and the ease of launching a replacement cluster is faster if one of the nodes goes down or has some overload.

## **EMR Storage Options**

The storage layer contains the many file systems that your cluster use. These are the basic storage options you have with EMR, and you will choose one of these whenever you are setting up your EMR cluster, whenever you are trying to process and store your data onto either EMR or use S3 for your input and output of your data.

| <b>File System</b> | <b>How to Access</b> | <b>Description</b> | <b>Usage</b> |
|--------------------|----------------------|--------------------|--------------|
|--------------------|----------------------|--------------------|--------------|

|   |         |  |  |
|---|---------|--|--|
| <b>Local File System<br/>(Instance Storage)</b> | ~/      | It is located on disks that are attached to the host machine. Size/speed is determined by instance type. | They are used for very high I/O performance and high IOPS at low cost. Best used for temporary data (caches, buffers, scratch data). |
| <b>Local File System<br/>(EBS Volume)</b>       | ~/      | Used to extend your HDFS, but these EBS volumes are ephemeral.   | It is used to add more storage for HDFS.   |
| <b>Hadoop Distributed File System (HDFS)</b>    | hdfs:// | Fast but ephemeral, distributed storage for EMR.   | Best used for caching the results produced by intermediate job-flow steps.   |
| <b>Elastic Map Reduce File System (EMRFS)</b>   | s3://   | Feature-rich persistent storage. Read and write files from EMR   | Best used for persistent store and S3 features that are needed, like server-side encryption and                                      |

|  |  |                 |              |
|--|--|-----------------|--------------|
|  |  | directly to S3. | consistency. |
|--|--|-----------------|--------------|

*Table 6-01: EMR Storage Options*

#### **KAM TIP:**

- EMR architecture consists of the primary core and tasks nodes.
- Primary nodes manage resources, track and direct HDFS, and monitor core and task nodes' health.
- Core Nodes run jobs as well as stores data.
- Task Node is used for extra compute power to help core nodes.
- The single availability zone concept means EMR cluster instances are all in the same AZ.
- EMR storage options include HDFS, Local File System (instance store and EBS), and EMRFS (integrated with S3).

## **EMR Operations - Transient vs. Long-Running**

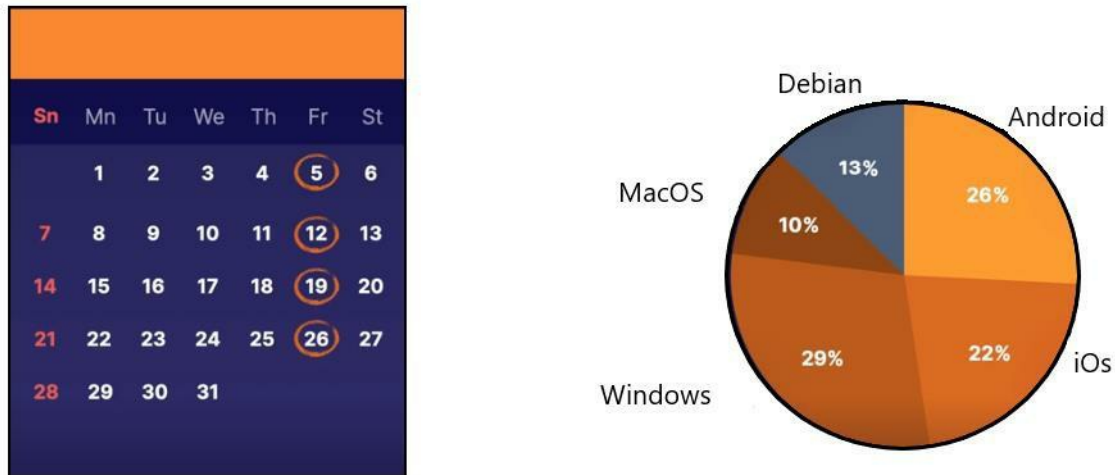
Transient and long-running EMR clusters are the two types of EMR clusters.

### **Transient Clusters**

If you set your cluster to terminate automatically, it will do so after completing all the steps. It is a Transient Cluster. Transient clusters are computed clusters that shut down and stop billing after completing the process.

Let us assume that we have some tasks that need to run every Friday of every week, and we need to produce some graph or metric to show our boss or the company's CEO. We have to run some EMR clusters to process the data. Let us assume that this analytic process needs to show the number or the percentage of users using your application and the type of operating system they are using. Therefore, the log files that

come in from requests from all of your users have this particular operating system that they are using to access your application, and all of that is stored into S<sub>3</sub>.



*Figure 6-17: Transient cluster*

We could create a transient EMR cluster.

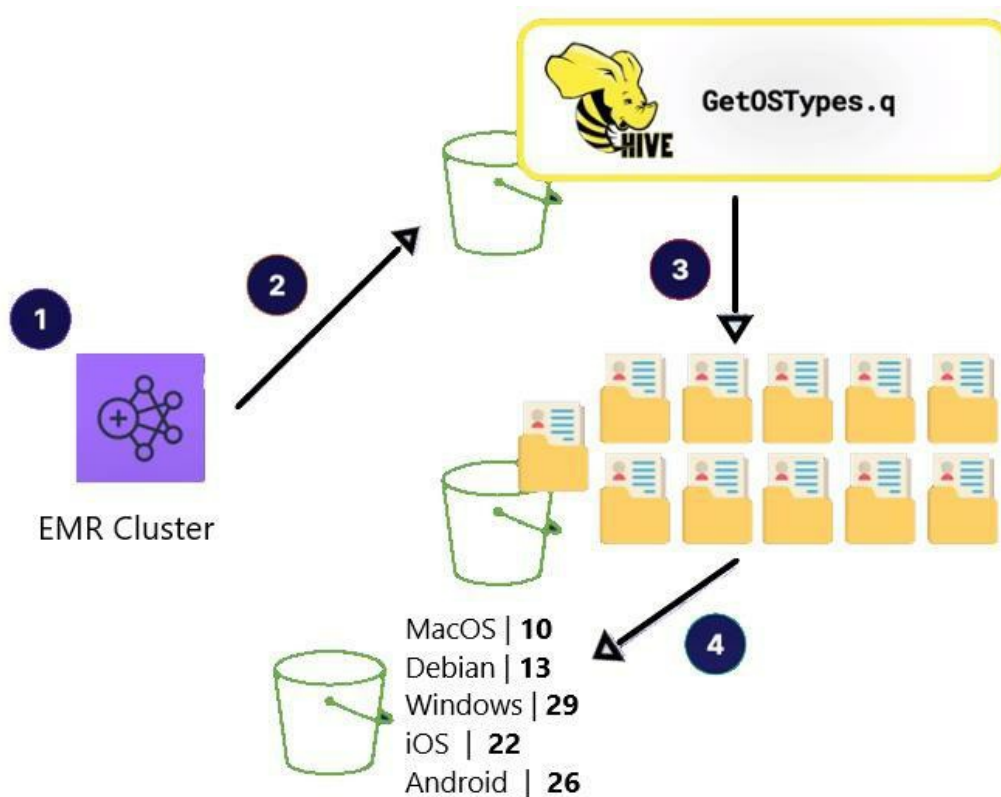


Figure 6-18: Transient EMR cluster

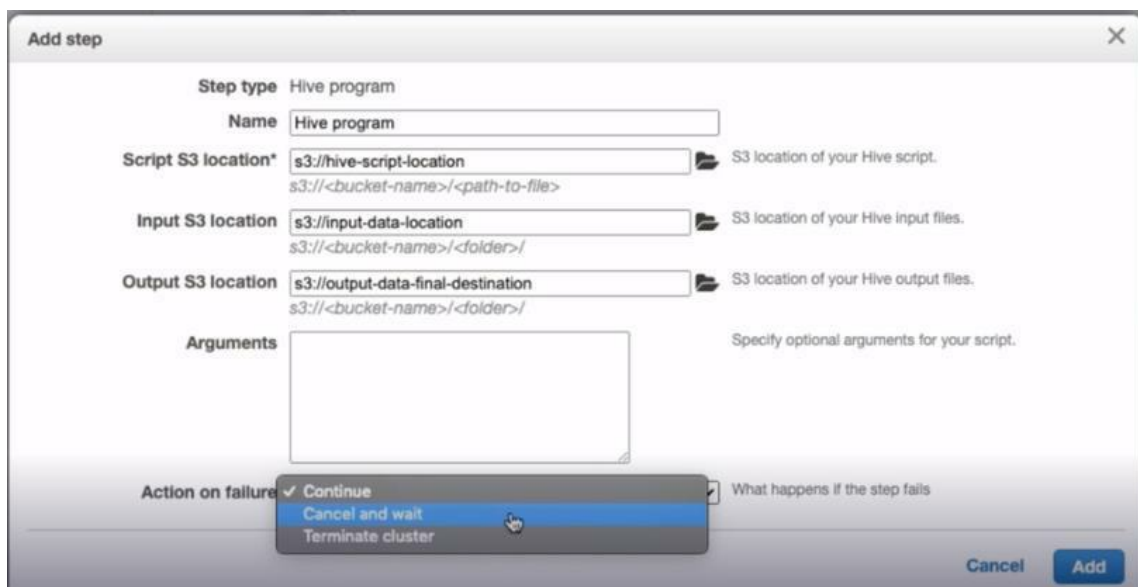
1. AWS CLI can create the EMR cluster, the API, or right within the console.
2. A degree is used within an EMR cluster for a unit of work. It needs to accomplish some tasks. We could then create a Hive script and run this as a step within our EMR cluster.
3. Hence, this Hive script will read the log data stored in S3. It is all of the access logs with information about all users and what OSs they were using.
4. We can then use the Hive script to assemble the data, determine what OSs they were using, and then return the percentage of users using that particular operating system.
5. We can output the results into an S3 bucket.
6. Once all of this is done, every Friday, this is going to, and then once it is complete, we will terminate the EMR cluster. It is what a transient cluster is. As soon as the process and job run, the cluster is removed.

If we look at what this looks like in the console, we can look at the steps. Steps are a unit of work that you can submit with your cluster. We can choose from these different settings shown in the image below. We can choose a Hive program, a Pig program, a Spark application, a custom jar, or any other streaming program. Let us assume we select the Hive program and add that step.



Figure 6-19: Select a Step

We can add the location for the Hive script. We can tell where the input data is what S3 bucket the input data is in. We can tell it where to store the output for the Hive query, and then we can also select any of the options for the action on failure, i.e., whether to terminate the instance if it fails, or go ahead and continue running the example if it fails.



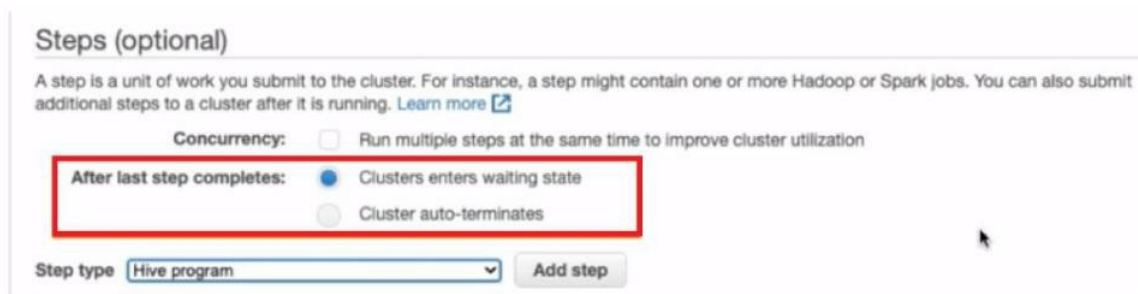
The 'Add step' dialog box is shown with the following fields and options:

- Step type:** Hive program
- Name:** Hive program
- Script S3 location\*:** s3://hive-script-location (with a placeholder s3://<bucket-name>/<path-to-file> and a description: S3 location of your Hive script.)
- Input S3 location:** s3://input-data-location (with a placeholder s3://<bucket-name>/<folder>/ and a description: S3 location of your Hive input files.)
- Output S3 location:** s3://output-data-final-destination (with a placeholder s3://<bucket-name>/<folder>/ and a description: S3 location of your Hive output files.)
- Arguments:** (An empty text area with a description: Specify optional arguments for your script.)
- Action on failure:** A dropdown menu is open showing three options:
  - ☒ Continue
  - ☐ Cancel and wait
  - ☐ Terminate cluster (The description is: What happens if the step fails)

Buttons at the bottom: Cancel, Add

Figure 6-20: Give details

Here, it says that put the cluster in a waiting state after the last step completes or auto terminates the cluster.



The 'Steps (optional)' section is shown with the following information:

A step is a unit of work you submit to the cluster. For instance, a step might contain one or more Hadoop or Spark jobs. You can also submit additional steps to a cluster after it is running. [Learn more](#)

**Concurrency:** ☐ Run multiple steps at the same time to improve cluster utilization

**After last step completes:**

- ☒ Clusters enters waiting state
- ☐ Cluster auto-terminates

**Step type:** Hive program (dropdown menu)

**Add step** (button)

Figure 6-21: After the last step completes

## Long-Running Clusters

If you set up the cluster to continue operating after processing is completed, the type of cluster is known as a long-running cluster. Long-running allows you to communicate with the cluster after it has completed its operations, but it requires manual shutdown.

## Considerations

### Transient Cluster

- The total number of EMR processing hours per day is less than 24, and you can benefit from shutting down your cluster when it is not being used.
- You are not using HDFS as your primary data storage (instead, you are using EMRFS with S3).
- Your job processing is intensive, iterative data processing.

### Long-running Cluster

- You frequently run processing jobs where it is beneficial to keep the cluster running after the previous position.
- Your processing jobs have an input-output dependency on one another.
- It is more cost-effective to store your data on HDFS instead of S3.
- You have a requirement of higher performance I/O HDFS provides.

**EXAM TIP:** If you set your cluster to terminate automatically, it will do so after completing all the steps. If you set up the cluster to continue operating after processing is completed, the type of cluster is known as a long-running cluster. This is known as Transient Cluster.

## EMR Operations - Choosing an Instance Type

Whenever we provision an EMR cluster, the instance size of our nodes is important because we might have workloads that are CPU intensive. We might have some that are input-output or memory intensive.

You can choose many different instances. Whenever we choose an instance type, we prefer it either for the primary node or for the core of task nodes. We can bunch core and task nodes together because the primary node will not be a super compute-intensive machine like the core and task nodes will be.

## Choosing an Instance Type

### Primary Nodes

- Primary Nodes does not have large computational requirements.
- For clusters with 50 or fewer nodes, you can use the M5 family.
- For clusters with greater than 50 nodes, you can use the M4 family.

### Core and Tasks Nodes

- Depends on the type of processing.
- For general purposes, the balance of CPU, disk space, and I/O, you can use the M5 family.
- For Batch Processing, HPC, or CPU-based machine learning, you can use C4, C5, and Z1d families.
- For Graphics processing or GPU-based machine learning, you can use G3, P2, and P3 families.
- For spark applications (in-memory caching), you can use R4 and R5 families.
- For Large HDFS and Map reduce jobs requiring high I/O performance and high IOPS, you can use H1, I3, and D2 families.

**EXAM TIP:** AWS limits you to 20 EC2 instances per region. You can

request a limit increase on that by contacting support.

## **EMR Operations - Choosing the Right Number of Instances**

Pairing the best instances with the right number of instances will help us in our EMR cluster to handle any workload and amount of data that needs to be processed. It is important to understand that a little cluster will be slow, and a big cluster will incur unnecessary costs. Therefore, it is important to get this number right as you are creating EMR clusters, tuning your EMR clusters, and processing and sorting data from EMR.

### **Choosing the Right Number of Instances**

#### **Primary Node**

- We can have just one primary node, or we can also have three primary nodes. It is a newer feature that helps customers maintain high availability for their primary node. AWS will detect that the traffic will flow through the new primary node if a primary node goes down. The primary node will be torn down, be terminated, and will spin it back up, a new primary node will come online.

#### **Core and Task Nodes**

- When choosing the right number of instances for your core and task nodes, you need to find out if you will be doing a lot of processing, i.e., running task, or are you going to be storing huge data in HDFS? If you are going to be holding huge data, which storage option will you be using? Will you be using EMRFS, HDFS instance store, EBS volumes for HDFS, or will you be using a combination of instance stores in EBS? Hence, the right number of nodes depends on your data.

We know that data is replicated across the nodes. It means that if you have very, very large files, then those files are copied across multiple instances. Hence, you need to consider large files and make sure that you have enough room on each of the dedicated models to store those replicated files.

It can adjust the replication factor depending on how many nodes you have. By default, within EMR, if the number of core nodes is ten plus, the replication factor is three. If you have four to nine nodes, then the replication factor will be two, and if you have one to three core nodes, then the replication factor will be one.

You can override the default replication factor in the software settings upon cluster launch or later in the `hdfs-site.xml` file.

## **HDFS Capacity Guidelines**

AWS gives us guidelines on calculating the HDFS capacity of a cluster:

- EMRFS
- HDFS (instance store or EBS) for high I/O requirements.

It depends on whether we are using EMRFS to store our data or using HDFS to store our data. If we are using HDFS, we are using instance storage or EBS volumes or some combination of both, and we would use HDFS over EMRFS if we have a high input-output requirement. Hence, HDFS is going to be a bit faster with higher input-output rather than EMRFS because that data is stored onto S3 rather than on the Hadoop cluster.

For example, we have high input-output requirements. That means we are going to use HDFS. We need to store three terabytes of data on the HDFS cluster. Whenever we are creating our EMR cluster within the

console, we can choose the master node sizes, the core node sizes, the task node sizes, and the number of instances that we want to run. You can also see that we can choose a purchasing option, either on-demand or spot.

Choose the instance type, number of instances, and a purchasing option. [Learn more about instance purchasing options](#)

Console options for automatic scaling have changed. [Learn more](#)

| Node type            | Instance type  | Instance count | Purchasing option  |
|----------------------|--|----------------|--|
| Master<br>Master - 1 | m5.xlarge<br>4 vCore, 16 GiB memory, EBS only storage<br>EBS Storage: 64 GiB<br>Add configuration settings | 1 Instances    | <input checked="" type="radio"/> On-demand<br><input type="radio"/> Spot<br>Use on-demand as max price |
| Core<br>Core - 2     | m5.xlarge<br>4 vCore, 16 GiB memory, EBS only storage<br>EBS Storage: 64 GiB<br>Add configuration settings | 2 Instances    | <input checked="" type="radio"/> On-demand<br><input type="radio"/> Spot<br>Use on-demand as max price |
| Task<br>Task - 3     | m5.xlarge<br>4 vCore, 16 GiB memory, EBS only storage<br>EBS Storage: 64 GiB<br>Add configuration settings | 0 Instances    | <input checked="" type="radio"/> On-demand<br><input type="radio"/> Spot<br>Use on-demand as max price |

+ Add task instance group

*Figure 6-22: Multiple options for the instance*

Assume we are using an on-demand instance. If we select the instance types for the core, we can see the various instance types that we can choose from.

|                       |             |    |      |           |
|-----------------------|-------------|----|------|-----------|
| <input type="radio"/> | h1.4xlarge  | 16 | 64   | 4000 SSD  |
| <input type="radio"/> | h1.8xlarge  | 32 | 128  | 8000 SSD  |
| <input type="radio"/> | h1.16xlarge | 64 | 256  | 16000 SSD |
| <input type="radio"/> | hs1.8xlarge | 64 | 117  | 49152 SSD |
| <input type="radio"/> | i2.xlarge   | 4  | 30.5 | 800 SSD   |
| <input type="radio"/> | i2.2xlarge  | 8  | 61   | 1600 SSD  |
| <input type="radio"/> | i2.4xlarge  | 16 | 122  | 3200 SSD  |
| <input type="radio"/> | i2.8xlarge  | 32 | 244  | 6400 SSD  |
| <input type="radio"/> | i3.xlarge   | 4  | 30.5 | 950 SSD   |
| <input type="radio"/> | i3.2xlarge  | 8  | 61   | 1900 SSD  |
| <input type="radio"/> | i3.4xlarge  | 16 | 122  | 3800 SSD  |
| <input type="radio"/> | i3.8xlarge  | 32 | 244  | 7600 SSD  |

*Figure 6-23: Instance types for core nodes*

Let us assume that we choose the I3 extra-large. It is going to have 950 gigabytes of SSD. If we multiply that by five, it will have a total of five times 950 gigabytes of storage to store all of the HDFS data.

Choose the instance type, number of instances, and a purchasing option. [Learn more about instance purchasing options](#)

Console options for automatic scaling have changed. [Learn more](#)

| Node type            | Instance type  | Instance count                           | Purchasing option   |
|----------------------|--|--|---|
| Master<br>Master - 1 | m5.xlarge<br>4 vCore, 16 GiB memory, EBS only storage<br>EBS Storage: 64 GiB<br>Add configuration settings | 1 Instances                              | <input checked="" type="radio"/> On-demand<br><input type="radio"/> Spot<br><input type="text" value="Use on-demand as max price"/> |
| Core<br>Core - 2     | i3.xlarge<br>4 vCore, 30.5 GiB memory<br>EBS Storage: none<br>Add configuration settings                   | <input type="text" value="5"/> Instances | <input checked="" type="radio"/> On-demand<br><input type="radio"/> Spot<br><input type="text" value="Use on-demand as max price"/> |
| Task<br>Task - 3     | m5.xlarge<br>4 vCore, 16 GiB memory, EBS only storage<br>EBS Storage: 64 GiB<br>Add configuration settings | <input type="text" value="0"/> Instances | <input checked="" type="radio"/> On-demand<br><input type="radio"/> Spot<br><input type="text" value="Use on-demand as max price"/> |

+ Add task instance group

*Figure 6-24: Choosing the instance i3x.large*

We know that core node are the only nodes that store data. Task nodes are used for extra processing power. We have 4.75 terabytes of data that we can store, and since we chose five nodes, the replication factor by default is two. If we calculate the HDFS capacity, it will be 2.375 terabytes, which is not large enough for the three terabytes of data. Hence, the HDFS capacity is not enough space. Therefore, it is important to consider the replication factor and how much data you need to store across your nodes.

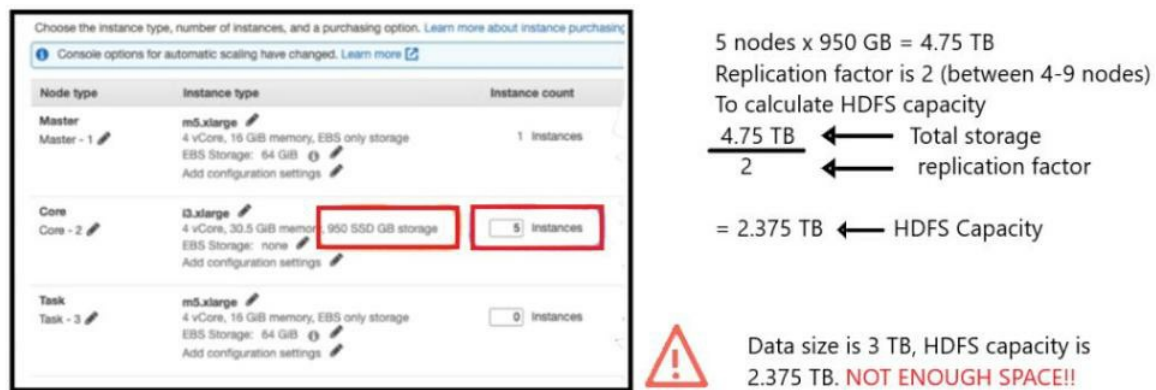


Figure 6-25: Calculating HDFS Capacity

To alleviate that issue, we go back into the console. To add more storage for our HDFS cluster, we have a couple of options. We can add more instances, or we also have the option of adding EBS storage. Hence, we can either make it ten instances or add extra EBS volumes to add more room onto our HDFS file system, and we can also increase the size of the EBS volumes. We also have the option of just changing the core instance types. We can change them to a specific instance type with more instant storage data. Therefore, those are our three options.



*Figure 6-26: Add EBS volume*

AWS suggests using a smaller cluster of larger nodes, and there are a few reasons behind it. It reduces the failure possibilities. If there are fewer moving parts or fewer nodes, there are fewer chances of nodes failing. It also reduces the amount of maintenance. Less moving parts means less maintenance to ensure that the cluster is working properly.

**EXAM TIP:** By default, within EMR, if the number of core nodes is ten plus, the replication factor is three. If you have 4 to 9 nodes, then the replication factor will be two, and if you have 1 to 3 core nodes, then the replication factor will be one. You can override the default replication factor in the software settings upon cluster launch or later in the `hdfs-site.xml` file.

## EMR Operations - On-Demand and Spot Instances

### On-Demand Instances

You pay for computing capacity by the second with On-Demand Instances, and there are no long-term obligations. You have complete control over its lifespan, deciding when to start, restart, hibernate, or

terminate it.

When you buy On-Demand Instances, you do not have to commit to anything long-term. You pay for time your On-Demand Instances are operating. A running On-Demand Instance's pricing per second is fixed.

## **Spot Instances**

A Spot Instance is a virtual machine that runs on spare EC2 capacity accessible at a lower price than the On-Demand price. Spot Instances allow you to request new EC2 instances at great discounts, allowing you to reduce your Amazon EC2 charges dramatically.

If you can be flexible about when your applications run and if your applications is interrupted, Spot Instances are a cost-effective option. Spot Instances are ideal for data analysis, batch processes, background processing, and optional activities.

We will look at some scenarios where we would want to use one over the other.

## **Quick Reference for Application Scenarios**

Most of the EMR scenarios will fall into one of these categories. Assume we run long-running clusters with predictable variations in compute capacity, such as a data warehouse. We might have requirements for a cost-driven structure, meaning that you need to run a transient cluster because lower costs are more important than the time is to complete. We also have data-critical scenarios, meaning that if you lose any part of the data, that is not acceptable. You need to ensure that all data is persisted on HDFS and protected from sudden terminations. Finally, you might run EMR clusters for application testing, which means that you are testing new applications and preparing them for production

environments.







*Figure 6-27: Application Scenarios*

We will look at what type of instances we would want to run for each of these scenarios, whether an on-demand or spot instance.

1. For long-running clusters and data warehouses, we need the data to persist. We also have an idea of what the computational capacity will be. For example, let us assume at the end of every week or the end of every day, all of the data scientists in the company submit an EMR job at a particular time, let us say 4:00 PM, 5:00 PM. The compute capacity needs to expand to handle all of the jobs. It means that we can add task nodes to add more computational power to the EMR cluster. Hence, we could use spot instances for our task nodes to save the most money or set up an instance fleet. Instance fleet allows us to use spot instances if the capacity is available and then switch to on-demand power if the spot is unavailable at the price we bid on. Depending on the importance of the workload instance, the fleet might be a better option. We would use the on-demand instances to handle the normal capacity and spot instances to run our peak load requirements.
2. If money is a huge factor and we want to save as much money as possible, we could use spot instances for our different nodes. Let us say that the spot instance price exceeds whatever you bid, then that means you will lose any data on the EMR cluster. Because we will be running transient clusters at this point, we are more focused on saving money than the actual time it takes to complete,

but losing partial work is acceptable.

3. It is where critical data scenarios come into play. We cannot lose any data if the instances within the cluster are terminated. We will also not lower costs, but losing partial data is not acceptable. We could run on-demand instances on the primary and the core nodes to ensure that any data persisted within HDFS. We could use spot instances for our task nodes to handle any spikes in workload.
4. The entire cluster can be run on spot instances with application testing to save on testing costs.

| Application Scenarios  | Primary Nodes | Core Nodes                  | Task Nodes             |
|--|---------------|-----------------------------|------------------------|
|  <b>Long-Running</b>          | On-Demand     | On-Demand or Instance-fleet | Spot or Instance-fleet |
|  <b>Cost-Driven</b>           | Spot          | Spot                        | Spot                   |
|  <b>Data Critical</b>        | On-Demand     | On-Demand                   | Spot or Instance-fleet |
|  <b>Application Testing</b> | Spot          | Spot                        | Spot                   |

*Figure 6-28: Choosing Instance type*

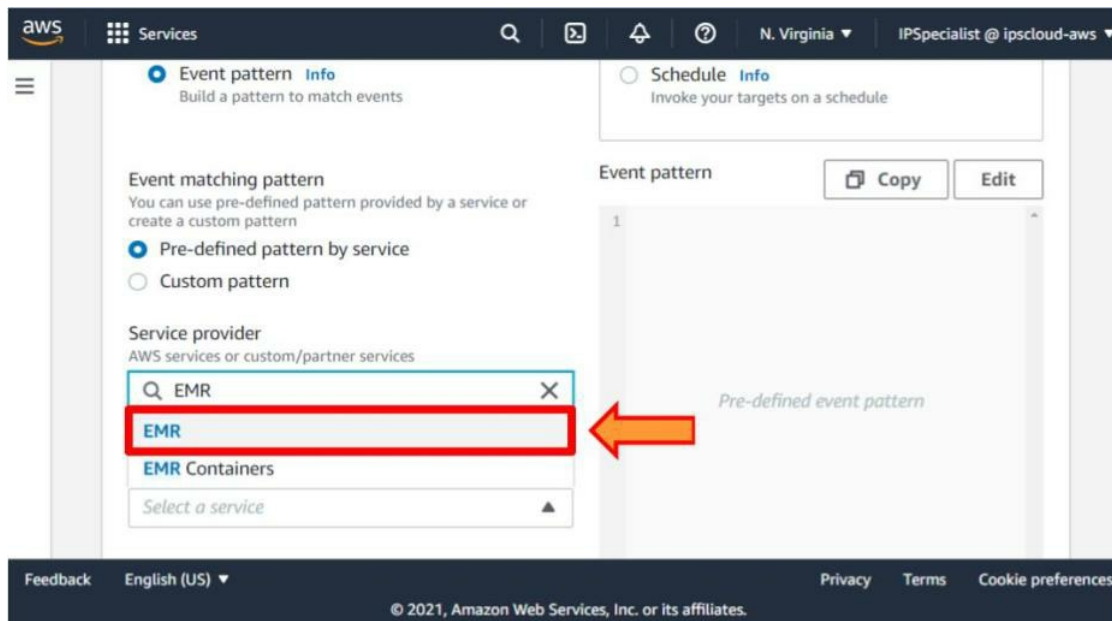
**EXAM TIP:** If you want to save money, then use spot instances. If your data is super important or your instances are running the entire time, make sure to use on-demand instances for your primary and core nodes, and for your task nodes, you can use spot instances.

## EMR Operations - Monitoring and Resizing Clusters

## CloudWatch Events

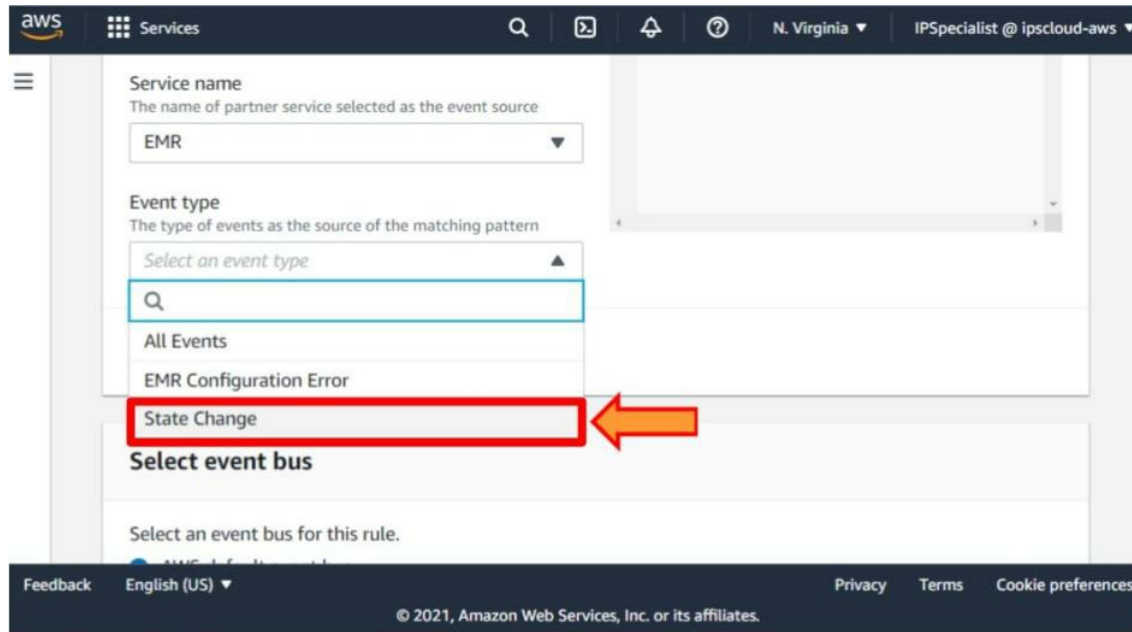
CloudWatch is used for monitoring your AWS resources, and there are several different CloudWatch events that we can use to monitor our EMR clusters.

If we look at this in the console, we can create a CloudWatch event rule. We can select the EMR as a service and select the event type.



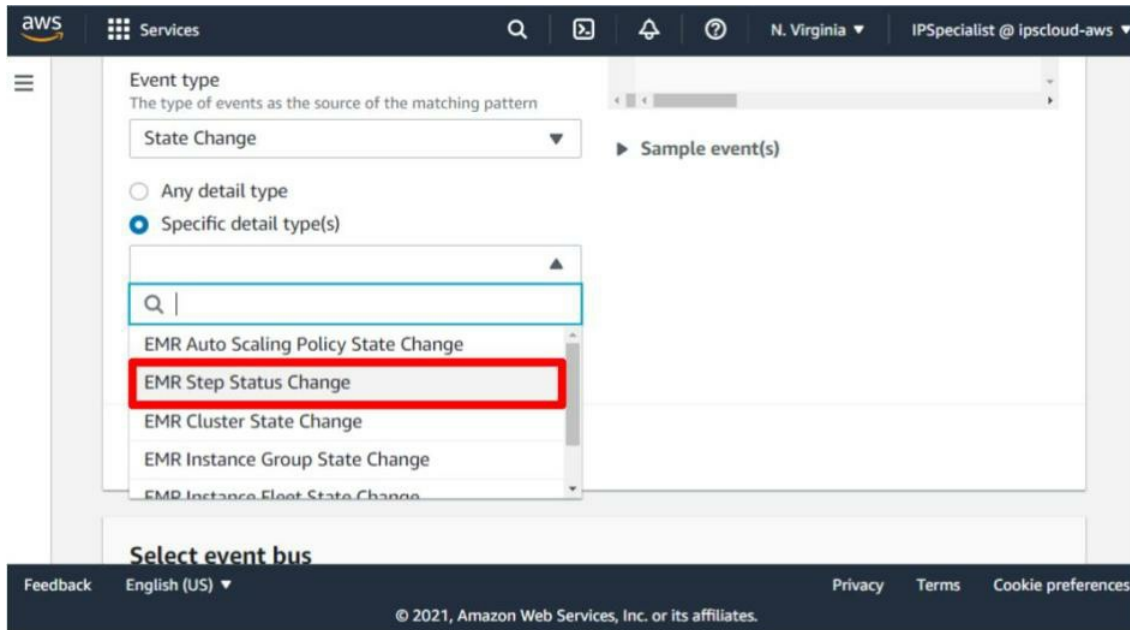
*Figure 6-29: Service Name*

We can select a **state change** either within the EMR cluster or within any EMR configuration error that happens. Let us assume we prefer a state change. We want to know if any of the EMR cluster state changes.



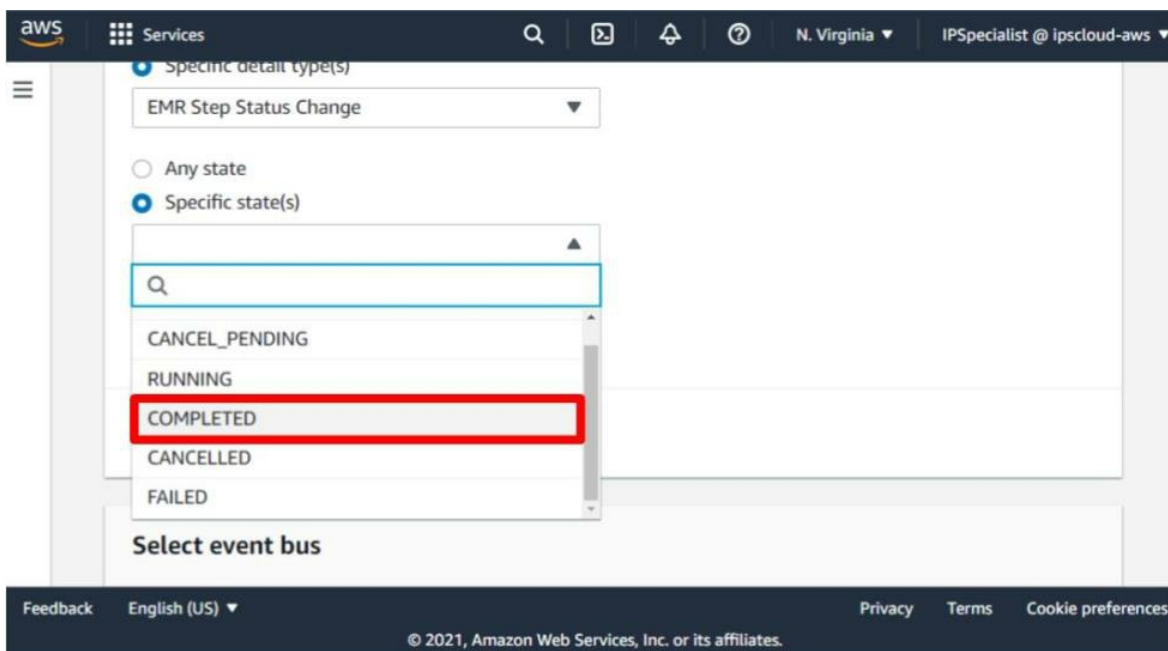
*Figure 6-30: Event Type*

If we select a **state change**, we can choose from many different state changes. For example, we can be notified if we create an auto-scaling policy that scales up our cluster during high demands. If the EMR cluster or the auto-scaling group adds more EC2 instances, we can also create steps or jobs within our EMR cluster. If the status changes on those, i.e., whether they are completed or terminated, we can be notified of that. We can be told for cluster state changes, i.e., is our cluster up and running? Is it terminated? Is there some type of error that happened? Hence, there are many different types of state changes that we can choose from.



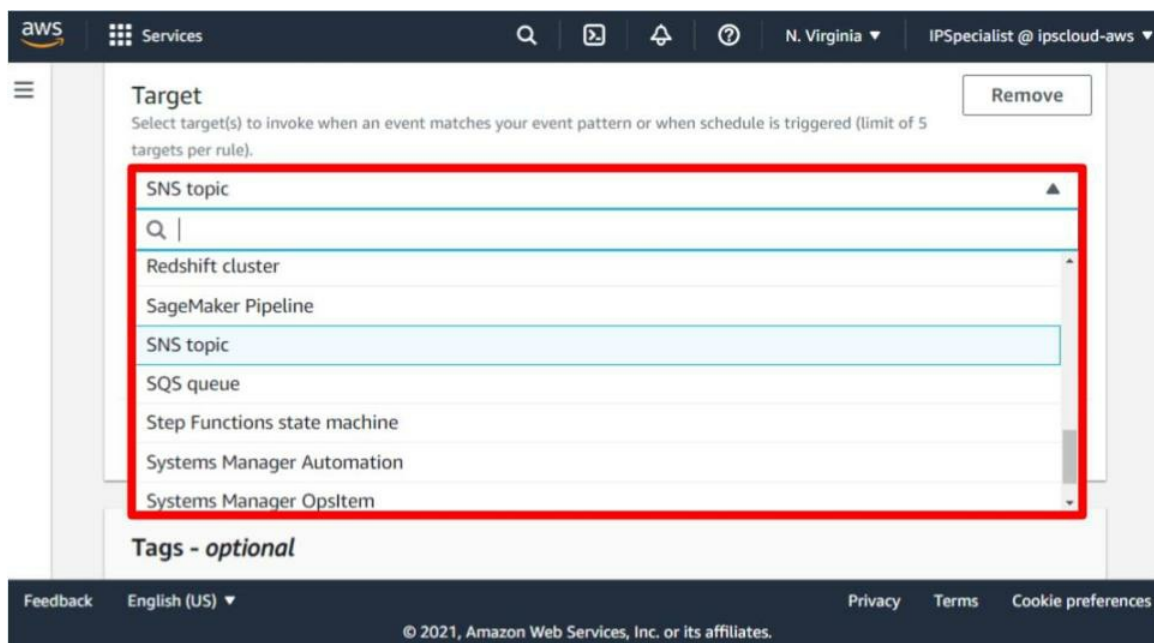
*Figure 6-31: Specific Detail Type*

If we select a **step status change**, we can determine if the job being run is canceled, pending, completed, failed, or currently running. It gives us a good understanding of what is being done on the cluster, the processing, and whether the job is completed.



*Figure 6-32: Specific State*

We can invoke many different targets whenever the Cloud Watch rule event occurs. Let us assume we select **when it completes**. Once it is complete, the CloudWatch event rule can then trigger another event. This event might be an SNS topic that sends an email or text message. We could also start a Lambda function that spins up another EMR cluster that uses the output data from the original EMR cluster.



*Figure 6-33: Targets*

## CloudWatch Metrics

We will discuss some metrics that we might want to follow to scale up our cluster or add more instances or needing to scale down our cluster.

- **Tracking Cluster Progress**

If we want to track the progress of the cluster, we can use these metrics:

1. RunningMapTasks
2. RemainingMapTasks

3. RunningReduceTasks
4. RemainingReduceTasks

It will help us determine the number of maps, reduce tasks currently running on the cluster, as well as the number of maps, and reduce tasks remaining for a particular job.

- **Detecting Idle Cluster**

We could also track a Cloud Watch metric that helps us determine if one of our clusters is idle. It means we are being charged, and the EMR cluster is not even doing any work. That means life is costing us money, but it is not running any task.

- **No More Storage for HDFS**

Another important Cloud Watch metric so if there is no more storage for HDFS, we can monitor the HDFS utilization metric, which is the percentage of disk space currently being used. We can trigger an event that fires once a high rate of power is used; let's say 80% of the capacity is used. We will fire a trigger that sends an email informing us that we are running out of space in HDFS. You need to add more instances, you need to add more EBS volumes, or you need to do something to make sure you have enough room for all of your HDFS data, as well as replications.

## **Monitor a Cluster with UI**

Not only can we monitor our cluster with CloudWatch metrics, but we can also monitor our cluster with an actual user interface. If you go into the EMR documentation, you can see all of the links for these various UI components that come installed with an EMR cluster.

| Name of interface                          | URI  |
|--|--|
| Ganglia                                    | http:// <i>master-public-dns-name</i> /ganglia/    |
| Hadoop HDFS NameNode (EMR version pre-6.x) | https:// <i>master-public-dns-name</i> :50470/     |
| Hadoop HDFS NameNode                       | https:// <i>master-public-dns-name</i> :50070/     |
| Hadoop HDFS DataNode                       | https:// <i>coretask-public-dns-name</i> :50075/   |
| Hadoop HDFS NameNode (EMR version 6.x)     | https:// <i>master-public-dns-name</i> :9871/      |
| Hadoop HDFS DataNode (EMR version pre-6.x) | https:// <i>coretask-public-dns-name</i> :50475/   |
| Hadoop HDFS DataNode (EMR version 6.x)     | https:// <i>coretask-public-dns-name</i> :9865/    |
| HBase                                      | http:// <i>master-public-dns-name</i> :16010/      |
| Hue  | http:// <i>master-public-dns-name</i> :8888/       |
| JupyterHub                                 | https:// <i>master-public-dns-name</i> :9443/      |
| Livy                                       | http:// <i>master-public-dns-name</i> :8998/       |
| Spark HistoryServer                        | http:// <i>master-public-dns-name</i> :18080/      |
| Tez  | http:// <i>master-public-dns-name</i> :8080/tez-ui |
| YARN NodeManager                           | http:// <i>coretask-public-dns-name</i> :8042/     |
| YARN ResourceManager                       | http:// <i>master-public-dns-name</i> :8088/       |
| Zeppelin                                   | http:// <i>master-public-dns-name</i> :8890/       |

*Figure 6-34: Interfaces*

We install Hadoop and other applications on our EMR cluster publish user interfaces as websites hosted on the primary node. Hence, a lot of the stuff that we can do using the command line by SSH into the primary node, we can also do through these user interfaces, and they are a bit easier to work with. To enable these user interfaces, we need to use SSH tunneling with local port forwarding or use SSH tunneling with port forwarding with SOCKS proxy settings.

## **Resizing a Cluster – Manually**

To resize the cluster, you cannot have fewer core nodes than the replication factor. We cannot resize less than the replication factor. For example, if we have ten or more core nodes, the replication factor is three, meaning we cannot resize our core nodes to have one or two core nodes. Therefore, we would first need to change the replication factor

in the HDFS-site.xml file and restart the NameNode daemon; then, we could manually adjust the number of core nodes.

## Resizing a Cluster – Auto Scaling

### *EMR-managed Scaling*

With EMR managed scaling, we can automatically increase and decrease the number of instances in the core and task nodes based on your workload. Master nodes do not scale, though. You set a minimum and maximum limit for the number of instances in your cluster nodes. You create a custom auto-scaling policy.

Whenever we create our EMR cluster within the console, we can select cluster scaling, and then we can use an EMR manager to scale or create a custom auto-scaling policy.

If we select EMR managed to scale, we set the minimum, the maximum, the on-demand limit, and the full core nodes we want. Depending on our workload, it will automatically decrease and increase the number of instances.

Cluster scaling

Adjust the number of Amazon EC2 instances available to an EMR cluster via EMR-managed scaling or a custom automatic scaling policy. [Learn more](#)

Cluster scaling ☒ Enable Cluster Scaling

☒ Use EMR-managed scaling

☐ Create a custom automatic scaling policy

EMR-managed scaling

EMR will automatically increase and decrease the number of instances in core and task nodes based on workload. Set a minimum and maximum limit of the number of instances for the cluster nodes. Master nodes do not scale.

Core and task units

Minimum: 2

Maximum: 4

On-demand limit: 4

Maximum Core Node: 4

***Figure 6-35: EMR Managed scaling***

## Custom Auto Scaling Policy

For a custom auto-scaling policy, we would select the minimum instances, the maximum instances, and what we want to scale out and scale back in, depending on the metrics within CloudWatch.

Adjust the number of Amazon EC2 instances available to an EMR cluster via EMR-managed scaling or a custom automatic scaling po

Cluster scaling ☒ Enable cluster scaling

Scaling method ☐ Use EMR-managed scaling

☒ Create a custom automatic scaling policy

### Custom automatic scaling policies

Create a custom automatic scaling policy to programmatically scale out and scale in core nodes and task nodes based on a Clou  
other parameters that you specify.

| Type   | Name       | Min | Max | Scale In    | Scale Out   |  |
|--------|------------|-----|-----|-------------|-------------|--|
| Master | Master - 1 | 0   | 0   | Not Enabled | Not Enabled | Not available for Master   |
| Core   | Core - 2   | 0   | 0   | Not Enabled | Not Enabled | No Rules Applied     |
| Task   | Task - 3   | 0   | 0   | Not Enabled | Not Enabled | No Rules Applied   |

Figure 6-36: Custom Automatic Scaling Policy

We can set the minimum and maximum instances for our core or task nodes if we go deeper. We can then select the scale-out policy. This scale-out policy is completely customizable to any of the CloudWatch metrics that you want to follow, whether it has to do with memory utilization or CPU utilization or if it has to do with the amount of space that is left in HDFS.

Finally, you can set a scale in the policy. If a certain threshold goes below or above a certain point, you can get rid of some core and task nodes.

Auto Scaling rules

Maximum instances:  ⓘ  
Minimum instances:  ⓘ

☒ Scale out

Default-scale-out-1: Add  instance if YARNMemoryAvailablePercentage is less than  for  five-minute period with a cooldown of  seconds

Default-scale-out-2: Add  instance if ContainerPendingRatio is greater than  for  five-minute period with a cooldown of  seconds

+ Add rule

☒ Scale in

Default-scale-in: Terminate  instance if YARNMemoryAvailablePercentage is greater than  for  five-minute period with a cooldown of  seconds

+ Add rule

*Figure 6-37: Auto Scaling Rules*

Let's assume we wanted to add a new rule for a scale-out policy. It would look like the image below.

Rule name

MyScalingRule

Scaling Adjustment

Add  Instances ⓘ

if YARNMemoryAvailablePercentage ⓘ

is greater than or equal to ⓘ

15 (percent) ⓘ

Comparison operator

Threshold

for  five-minute periods ⓘ

Evaluation Period

Cooldown period

Cooldown period  seconds ⓘ

*Figure 6-38: New Rule for Scale-out Policy*

We need to set the minimum and a maximum number of instances that

need to be spun up or have an individual capacity when this scale-out rule is hit. We give it a rule name, add the number of EC2 instances, set the CloudWatch metric that we want this rule to trigger on, and give the comparison operator a particular value. This rule will trigger as long as that value is met for an evaluation period or a specific amount of time. We can also set a cool-down period. A cool-down period determines the amount of time that must elapse between a scaling activity, starting by rule, and the start of the next scaling activity, regardless of the power that triggers. Once the scaling activity occurs, you want to make sure that you provide it with some cool-down period to ensure your EMR cluster is available and stable. At that point, the next rule can start to be evaluated.

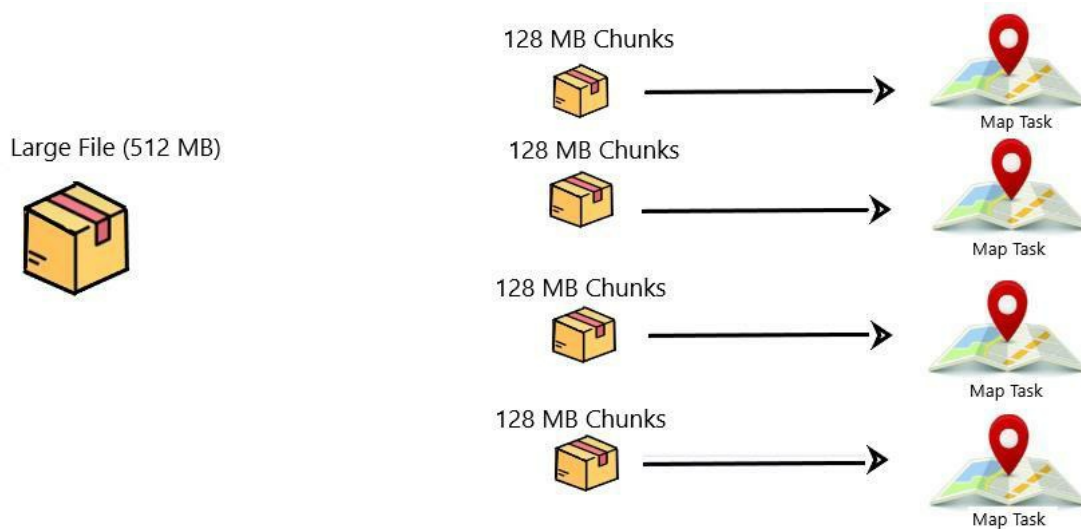
**EXAM TIP:** We can monitor our cluster by monitoring the CloudWatch metrics. We can manually resize our EMR clusters, or we can use auto-scaling. Either we can use a custom auto-scaling policy or the EMR managed to scale policy.

## EMR File Storage and Compression

### How Hadoop Splits Files?

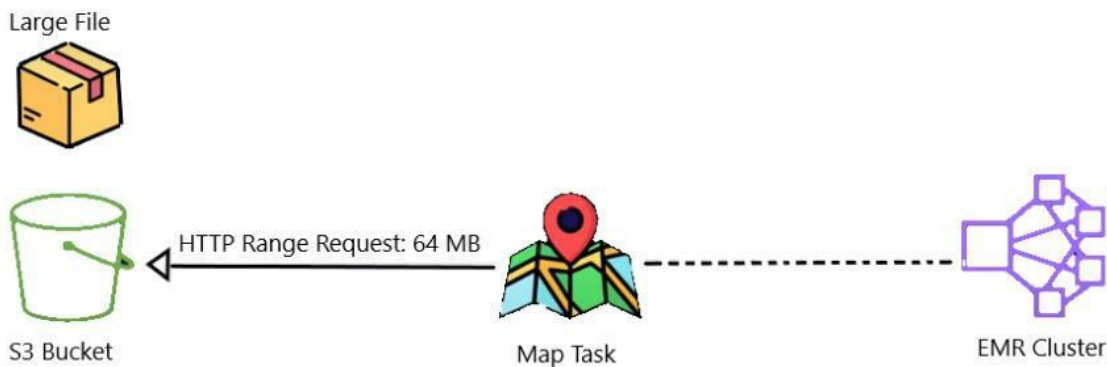
Hadoop splits large files into multiple chunks of smaller sizes. After breaking the files, a single map task processes each part. The HDFS framework has already separated the data files into various blocks using Hadoop as the underlying data storage. Since our data is already fragmented, Hadoop uses HDFS data blocks to assign a single map task to each of the HDFS blocks. Hence, whenever we use Hadoop on our EMR cluster, Hadoop either splits the files or stores the files in HDFS or has the file stored in S3. If stored in HDFS, the files are automatically

divided into chunks. If they are stored into S3, files are split into multiple HTTP range requests. Whether using HDFS or S3, the compression algorithm needs to have splitting available.



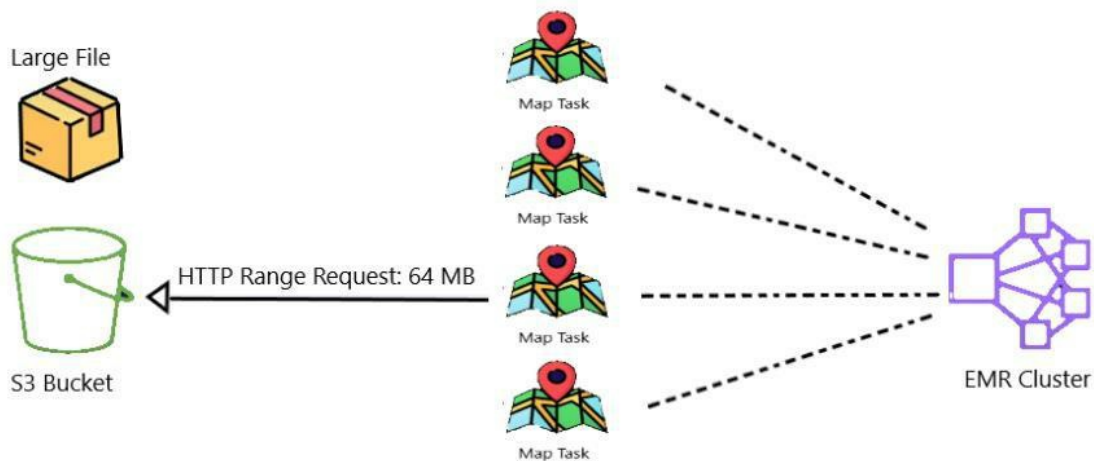
*Figure 6-39: Splitting of files*

Consider an example in which we have a large file on which we are using some compression algorithm. It does not allow for splitting.



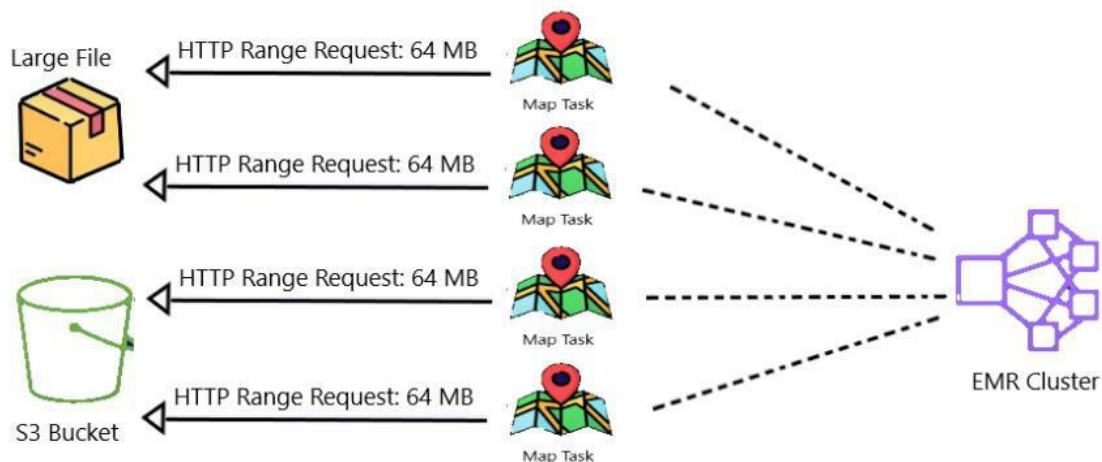
*Figure 6-40: Splitting example (i)*

That means that only a single map task will be mapped to that file, but if it allows for splitting, in this case, we can have multiple HTTP range requests when moving that data from S3 onto our EMR cluster.



*Figure 6-41: Splitting example (ii)*

If the compression algorithm we are using allows for the files to be split, then EMR will process the chunks in parallel.



*Figure 6-42: Splitting example (iii)*

## Different Compression Algorithms

The table below gives us a good understanding of which algorithms are splittable versus those not splittable. It lays out the compression ratio and can it speed to decompressing those files.

If we are trying to save space on our HDFS cluster, we might use the

Gzip or the Bzip2 compression algorithm.

| Algorithm | Splittable | Compression Ratio | Compress + Decompress Speed |
|-----------|------------|-------------------|-----------------------------|
| GZIP      | No         | High              | Medium                      |
| bzip2     | Yes        | Very High         | Slow                        |
| LZO       | Yes        | Low               | Fast                        |
| Snappy    | No         | Low               | Very Fast                   |

Space Savers

*Figure 6-43: Space Savers*

If we are trying to save time, we might use the LZO and Snappy algorithms.

| Algorithm | Splittable | Compression Ratio | Compress + Decompress Speed |
|-----------|------------|-------------------|-----------------------------|
| GZIP      | No         | High              | Medium                      |
| bzip2     | Yes        | Very High         | Slow                        |
| LZO       | Yes        | Low               | Fast                        |
| Snappy    | No         | Low               | Very Fast                   |

Time Savers

*Figure 6-44: Time Savers*

Bzip2 and LZO are splittable algorithms. These will be the best algorithms to use if we want to map multiple. We will map request tasks to a single file because these files can be split up when moving from S<sub>3</sub> onto our cluster.

| Algorithm | Splittable | Compression Ratio | Compress + Decompress Speed |
|-----------|------------|-------------------|-----------------------------|
| GZIP      | No         | High              | Medium                      |
| bzip2     | Yes        | Very High         | Slow                        |
| LZO       | Yes        | Low               | Fast                        |
| Snappy    | No         | Low               | Very Fast                   |

Splitability

*Figure 6-45: The split ability*

## The Benefits of File Compression

- **Better Performance:** You get better performance when less data is transferred between S3, mappers, and reducers.
- **Less Network Traffic:** It also gives us less network traffic between S3 and EMR since you share fewer data.
- **Reduced Storage Costs:** Smaller compressed files take up less storage, so you end up paying less for storage.

## Different EMR File Formats

EMR accepts many different file formats. We can use text files. This can be CSV files or tab-separated files, or JSON files. These are commonly used everywhere. Parquet is going to be a column-oriented file format. It is widely used in the Hadoop ecosystem and has HBase, Map reduce, Pig, Spark. It is a file format created by Apache specifically for the Hadoop ecosystem. We can also use ORC or ORC file formats. It is an optimized row-column file format, and this is a highly efficient way to store Hive data. EMR also accepts sequence files, and these are flat files that consist of binary key-value pairs. The last file format is Avro, and it is a row-oriented data serialization framework created by the Apache Hadoop project. It uses JSON to define the different data types and protocols. It serializes data into a compact binary format.

Understanding the file formats that we can use and the compression algorithms will better understand how large our files need to be.

| File Format  | Descriptions                                     | Commonly Used  |
|--------------|--|--|
| Text         | csv, tsv, json                                   | Everywhere   |
| Parquet      | Columnar-oriented file format                    | Common in the Hadoop ecosystem (Hive, HBase, Map reduce, Pig, Spark) |
| ORC          | Optimized Row Columnar file format               | The highly efficient way to store Hive data                          |
| SequenceFile | A flat file consisting of binary key/value pairs | Used extensively with Map reduce input/output formats                |
| Avro         | Row-oriented Data serialization framework        | Developed and used by the Hadoop project.                            |

*Table 6-02: EMR File Formats*

## File Sizes Best Practices

According to AWS and the EMR best practices, we can look at the algorithms we are using, whether they are splittable or not, and then determine the file sizes. Hadoop will assign a single mapper to process our data if the compression type does not allow for splitting. That

means that a single thread is responsible for fetching that data from S3. Since a single line is limited to how much information it can pull from S3 at any given time, this is the throughput. The process of reading the entire file from S3 into a mapper becomes a bottleneck for our data. Hence, the best practice is to have a file size of **one to two gigabytes** for algorithms that do not allow splitting. If splitting is available for our compression algorithm, we can have file sizes from **two to four gigabytes**.

We need to avoid smaller file sizes, hence, 100 megabytes or less. We need to plan for fewer larger files. Typically, you want your file sizes to map to the block size that you have set within Hadoop and EMR. Subsequently, it will feed those files into a mapper; your block size has a fixed cost; hence, you want to utilize that over large volumes of data to maximize the overall throughput of your cluster.

Block size and file size need to be about the same. If you have smaller files, you have two options.

- Your first option is you can reduce the HDFS block size. For example, you can set it to one megabyte (1MB) since the default is 128 megabytes (128 MB).
- Your other option is to use the S3DistCP command to combine smaller files. You can take a lot of smaller files and combine them into a single large file.

## **S3DistCp Command**

S3DistCP is an extension of DistCP. It is optimized to work with AWS S3 service. The Apache framework creates DistCP, while AWS makes the S3DistCP command. It has been optimized for your EMR and S3 workloads.

- It allows you to copy files within a cluster or from one cluster to

another or S3 into your HDFS cluster.

- It also allows you to combine smaller files into larger files. It can help you copy data between S3 buckets or S3 to HDFS or HDFS to S3.
- Either you can run S3DistCP by using a step within your EMR cluster, or you can run it on the primary node. It allows you to copy data and combine many small files into fewer larger files. We can either run this on the primary node's command line-shell or create a new step in the existing EMR clusters.

#### **KAM TIP:**

- **How Hadoop Splits Files** – Files are split into smaller chunks and a map task processes each chunk.
- **Different Compression Algorithm** – Gzip, Bzip2, LZO, and Snappy.
- **The benefits of File Compression** – Better performance, less network traffic, reduced storage cost.
- **Different EMR File Formats** – Text, Parquet, ORC, SequenceFile, and Avro.
- **File Sizes Best Practices** – Dependent on your compression algorithm and time constraints.
- **S3DistCP Command** – A tool used for copying files and combining many smaller files into fewer larger files.

## **Lab 6-01: Data Analytics with Spark and EMR**

### **Introduction**

#### **Amazon EMR**

Amazon EMR is the industry's most advanced cloud big data platform for data processing, interactive analysis, and machine learning. That leverages open-source frameworks such as Apache Spark, Apache Hive, and Presto. EMR allows you to execute petabyte-scale analysis for less than half the cost of typical on-premises solutions and more than 1.7x

quicker than ordinary Apache Spark.

Amazon EMR frees you up to focus on data transformation and analysis rather than maintaining computing resources or open-source apps, and it saves you money. You may rapidly deploy as much or as little capacity as you want on Amazon EC2 using EMR. You can build up scaling rules to handle changing compute demand. You may configure CloudWatch alerts to notify you of changes in your infrastructure and take fast action. You may use EMR to submit workloads to Amazon EKS clusters if you utilize Kubernetes. Whether you utilize EC2 or EKS, EMR's optimized runtimes speed up your analysis and save you time and money.

### **Amazon Simple Storage Service (S3)**

Amazon S3 is a type of object storage that allows you to store and recover any quantity of data from any location. It is a low-cost storage solution with business resilience, reliability, efficiency, privacy, and infinite expansion.

Amazon S3 is a web service that allows you to store and retrieve an infinite quantity of data from any place and at any time. You may quickly create projects that integrate cloud-native storage using this service. Because Amazon S3 is easily customizable and you only pay for what you use, you can start small and scale up as needed without sacrificing performance or dependability.

Amazon S3 is built to be highly adaptable. Instead of finding out how to store their data, Amazon S3 allows developers to focus on innovation. Build a simple FTP system or a complex web application like the Amazon.com retail website. Read the same piece of data a million times

or only for emergency disaster recovery; store whatever type and amount of data you desire.

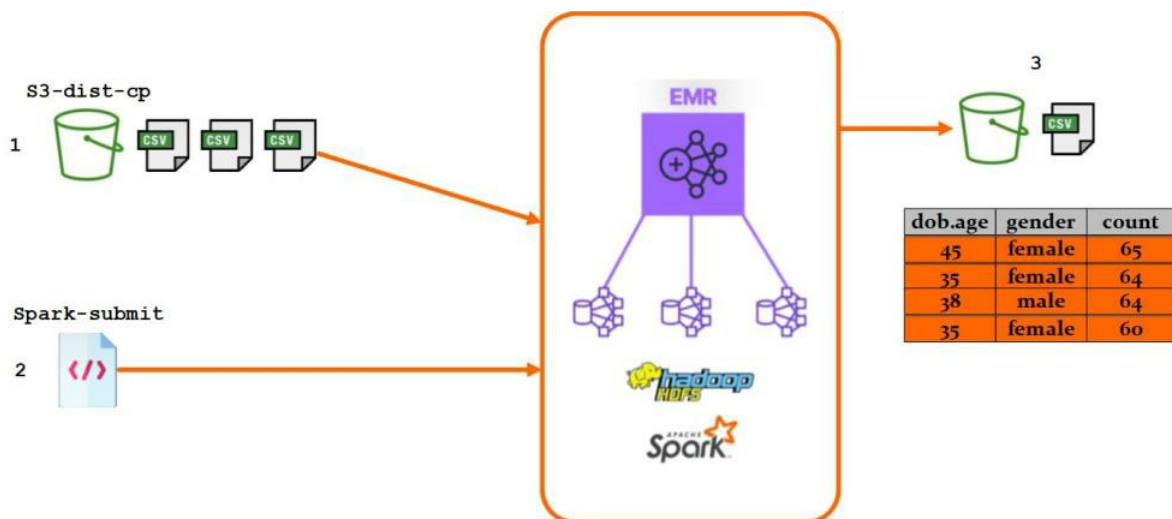
## Problem

Assume you are a data analytics in an organization. The organization that you work in has a wide variety of users. They give you the task of running some data analytics for an upcoming marketing campaign. They give you a target to determine the most common users, grouped by gender and age. Hence, how can you automate this task?

## Solution

The solution is you use the AWS service to automate this task. To accomplish this, you must first build an AWS EMR cluster and upload user data into HDFS. Following that, you will execute a PySpark Apache Spark script to count the number of users and categories them according to their age and gender. Finally, you will need to load the results into the AWS S3 bucket for further analysis.

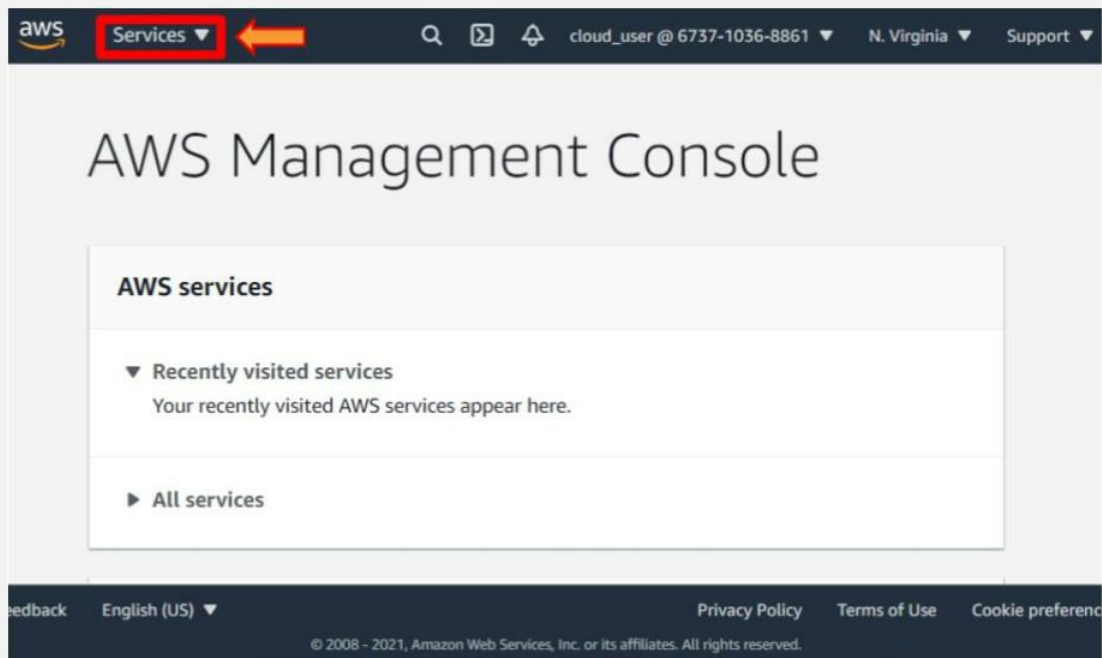
**Note:** Before starting the lab, create an S3 bucket as used in this lab.



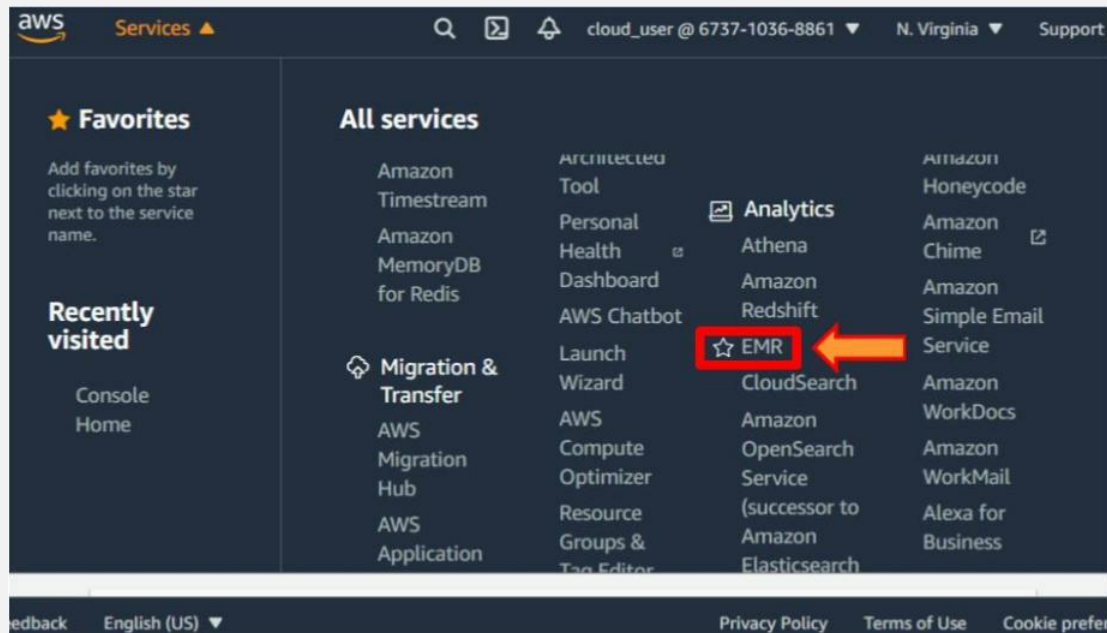
*Figure 6-46: Data Analysis with Spark and EMR*

### Step 1: Create AWS EMR Cluster

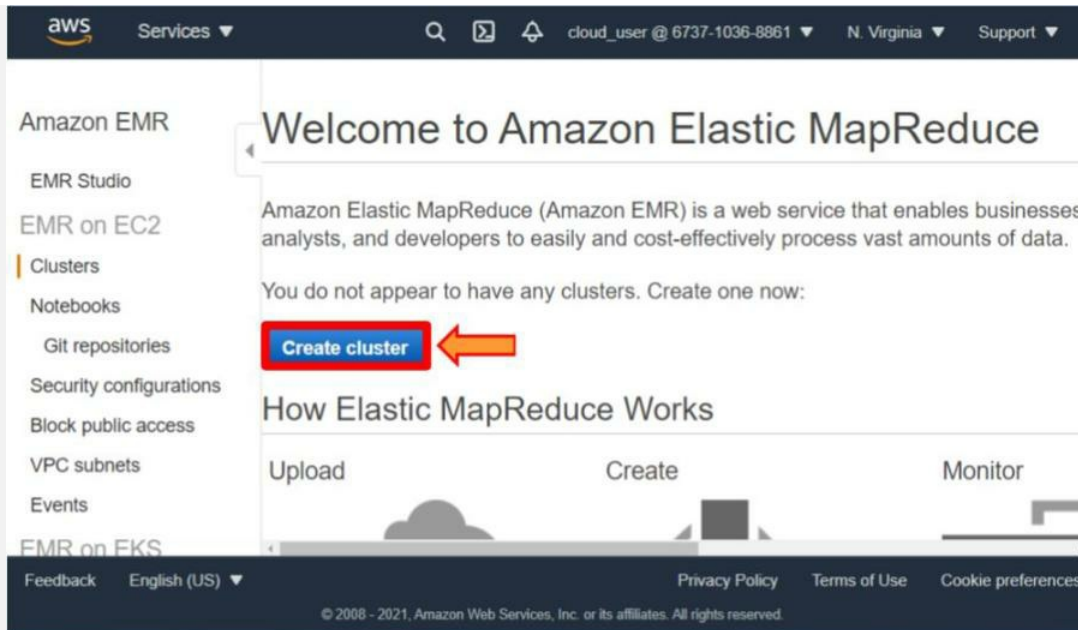
1. Log in to the **AWS Console**.
2. Click on the **Services**.



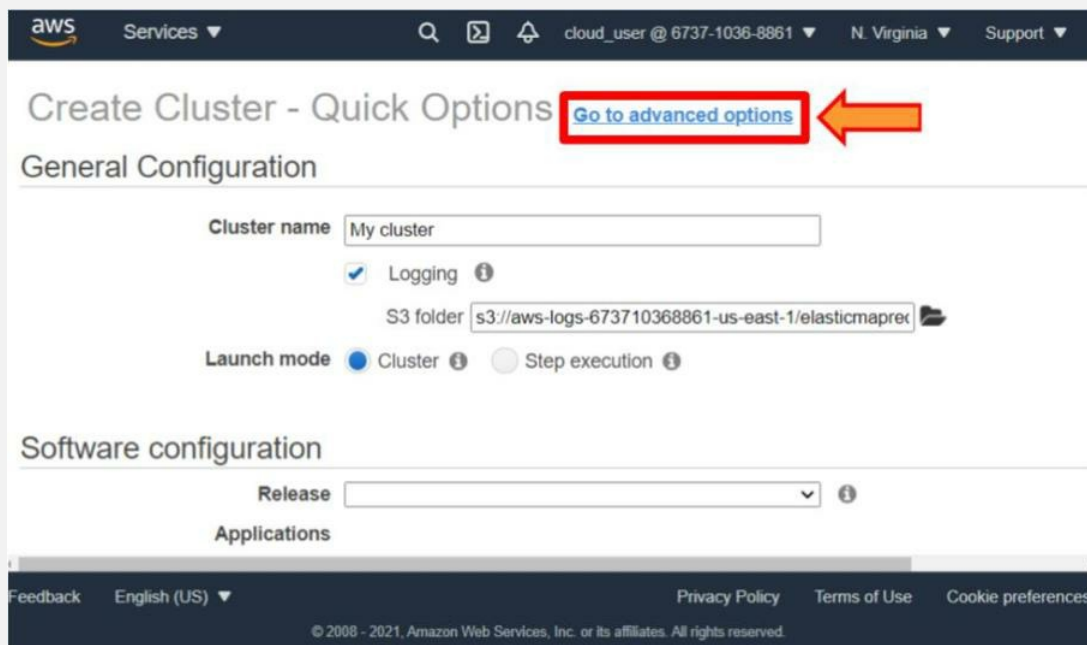
3. Select the **EMR** from the **Analytics**.



4. Click on the **Create Cluster** button.



5. Click on the **Go to advanced options**.



6. Select the **Hadoop 2.10.1** and **Spark 2.4.7**.

aws Services cloud\_user @ 6737-1036-8861 N. Virginia Support

## Create Cluster - Advanced Options [Go to quick options](#)

**Step 1: Software and Steps**

Step 2: Hardware

Step 3: General Cluster Settings

Step 4: Security

### Software Configuration

Release

|   |   |
|---|---|
| <input checked="" type="checkbox"/> Hadoop 2.10.1       | <input type="checkbox"/> Zeppelin 0.9.0         |
| <input type="checkbox"/> JupyterHub 1.1.0               | <input type="checkbox"/> Tez 0.9.2              |
| <input type="checkbox"/> Ganglia 3.7.2                  | <input type="checkbox"/> HBase 1.4.13           |
| <input checked="" type="checkbox"/> Hive 2.3.7          | <input type="checkbox"/> Presto 0.245.1         |
| <input type="checkbox"/> JupyterEnterpriseGateway 2.1.0 | <input type="checkbox"/> MXNet 1.7.0            |
| <input type="checkbox"/> Mahout 0.13.0                  | <input checked="" type="checkbox"/> Hue 4.9.0   |
| <input type="checkbox"/> Oozie 5.2.0                    | <input checked="" type="checkbox"/> Spark 2.4.7 |
| <input type="checkbox"/> TensorFlow 2.4.1               |   |

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

7. Scroll down. Click on the **Next** button.

aws Services cloud\_user @ 6737-1036-8861 N. Virginia Support

al)

you submit to the cluster. For instance, a step might contain one or more Hadoop or Spark jobs. You can also submit additional it is running. [Learn more](#)

**currency:** ☐ Run multiple steps at the same time to improve cluster utilization

**ompletes:** ☒ Clusters enters waiting state ☐ Cluster auto-terminates

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

8. Click on the **Pencil** icon button.

aws Services ▾ 🔍 ⓘ cloud\_user @ 6737-1036-8861 ▾ N. Virginia ▾ Support ▾

for automatic scaling have changed. [Learn more](#) ⓘ

| Instance type   | Instance count                           | Purchasing option  |
|---|--|--|
| <b>m5.xlarge</b> ⓘ<br>4 vCore, 16 GiB memory, EBS only storage<br>EBS Storage: 32 GiB ⓘ ⓘ<br>Add configuration settings ⓘ | 1 Instances                              | <input checked="" type="radio"/> On-demand ⓘ<br><input type="radio"/> Spot ⓘ<br>Use on-demand as max price |
| <b>m5.xlarge</b> ⓘ<br>4 vCore, 16 GiB memory, EBS only storage<br>EBS Storage: 32 GiB ⓘ ⓘ<br>Add configuration settings ⓘ | <input type="text" value="2"/> Instances | <input checked="" type="radio"/> On-demand ⓘ<br><input type="radio"/> Spot ⓘ<br>Use on-demand as max price |
| <b>m5.xlarge</b> ⓘ<br>4 vCore, 16 GiB memory, EBS only storage<br>EBS Storage: 32 GiB ⓘ ⓘ<br>Add configuration settings ⓘ | <input type="text" value="0"/> Instances | <input checked="" type="radio"/> On-demand ⓘ<br><input type="radio"/> Spot ⓘ<br>Use on-demand as max price |


Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

9. Select the **m4.large** instance type. Then click on the **Save** button.

Instance types ⓘ

|   |    |      |          |
|---|----|------|----------|
| <input type="radio"/> m2.2xlarge          | 4  | 34.2 | 850 SSD  |
| <input type="radio"/> m2.4xlarge          | 8  | 68.4 | 1690 SSD |
| <input type="radio"/> m3.xlarge           | 4  | 15   | 80 SSD   |
| <input type="radio"/> m3.2xlarge          | 8  | 30   | 160 SSD  |
| <input checked="" type="radio"/> m4.large | 2  | 8    | EBS only |
| <input type="radio"/> m4.xlarge           | 4  | 16   | EBS only |
| <input type="radio"/> m4.2xlarge          | 8  | 32   | EBS only |
| <input type="radio"/> m4.4xlarge          | 16 | 64   | EBS only |
| <input type="radio"/> m4.10xlarge         | 40 | 160  | EBS only |
| <input type="radio"/> m4.16xlarge         | 64 | 256  | EBS only |

 **Save**

10. Click on the **Pencil** icon button.

aws Services cloud\_user @ 6737-1036-8861 N. Virginia Support

for automatic scaling have changed. [Learn more](#)

| Instance type   | Instance count | Purchasing option  |
|---|----------------|--|
| <b>m4.large</b><br>2 vCore, 8 GiB memory, EBS only storage<br>EBS Storage: 32 GiB<br>Add configuration settings   | 1 Instances    | <input checked="" type="radio"/> On-demand<br><input type="radio"/> Spot<br>Use on-demand as max price |
| <b>m5.xlarge</b><br>4 vCore, 16 GiB memory, EBS only storage<br>EBS Storage: 32 GiB<br>Add configuration settings | 2 Instances    | <input checked="" type="radio"/> On-demand<br><input type="radio"/> Spot<br>Use on-demand as max price |
| <b>m5.xlarge</b><br>4 vCore, 16 GiB memory, EBS only storage<br>EBS Storage: 32 GiB                               | 0 Instances    | <input checked="" type="radio"/> On-demand<br><input type="radio"/> Spot<br>Use on-demand as max price |

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

11. Select the **m4.large** instance type. Then click on the **Save** button.

Instance types

|   |    |      |          |
|---|----|------|----------|
| <input type="radio"/> m2.2xlarge          | 4  | 34.2 | 850 SSD  |
| <input type="radio"/> m2.4xlarge          | 8  | 68.4 | 1690 SSD |
| <input type="radio"/> m3.xlarge           | 4  | 15   | 80 SSD   |
| <input type="radio"/> m3.2xlarge          | 8  | 30   | 160 SSD  |
| <input checked="" type="radio"/> m4.large | 2  | 8    | EBS only |
| <input type="radio"/> m4.xlarge           | 4  | 16   | EBS only |
| <input type="radio"/> m4.2xlarge          | 8  | 32   | EBS only |
| <input type="radio"/> m4.4xlarge          | 16 | 64   | EBS only |
| <input type="radio"/> m4.10xlarge         | 40 | 160  | EBS only |
| <input type="radio"/> m4.16xlarge         | 64 | 256  | EBS only |

**Save**

12. Update the instance count for the core node to **one**.

aws Services cloud\_user @ 6737-1036-8861 ▼ N. Virginia ▼ Support ▼

for automatic scaling have changed. [Learn more](#)

| Instance type   | Instance count                           | Purchasing option  |
|---|--|--|
| <b>m4.large</b><br>2 vCore, 8 GiB memory, EBS only storage<br>EBS Storage: 32 GiB<br>Add configuration settings | 1 Instances                              | <input checked="" type="radio"/> On-demand<br><input type="radio"/> Spot<br>Use on-demand as max price |
| <b>m4.large</b><br>2 vCore, 8 GiB memory, EBS only storage<br>EBS Storage: 32 GiB<br>Add configuration settings | <input type="text" value="1"/> Instances | <input checked="" type="radio"/> On-demand<br><input type="radio"/> Spot<br>Use on-demand as max price |
| <b>m5.xlarge</b><br>4 vCore, 16 GiB memory, EBS only storage<br>EBS Storage: 32 GiB                             | <input type="text" value="0"/> Instances | <input checked="" type="radio"/> On-demand<br><input type="radio"/> Spot<br>Use on-demand as max price |

Feedback English (US) ▼ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

13. Scroll down. Click on the Next button.

aws Services cloud\_user @ 6737-1036-8861 ▼ N. Virginia ▼ Support ▼

Cluster scaling ☐ Enable Cluster Scaling

on

the cluster terminate after the cluster becomes idle. Choose a minimum of 1 minute or a max of 24 hours. [Learn more](#)

Auto-termination ☐ Enable auto-termination

ume

e volume size up to 100 GiB. This sizing applies to all instances in the cluster. [Learn more](#) .

root device EBS volume size  GiB

Cancel **Next**

Feedback English (US) ▼ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

14. Give a name **ips-age-gender-analytics-cluster**.

aws Services cloud\_user @ 6737-1036-8861 ▼ N. Virginia ▼ Support ▼

## Create Cluster - Advanced Options [Go to quick options](#)

step 1: Software and Steps  
step 2: Hardware  
**step 3: General Cluster Settings**  
step 4: Security

### General Options

Cluster name

☒ Logging ⓘ  
S3 folder

☐ Log encryption ⓘ

☒ Debugging ⓘ

☒ Termination protection ⓘ

Tags

Feedback English (US) ▼ Privacy Policy Terms of Use Cookie preferences  
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

15. Click on the **Next** button.

aws Services cloud\_user @ 6737-1036-8861 ▼ N. Virginia ▼ Support ▼

Value (optional)

Options

Parent view ⓘ

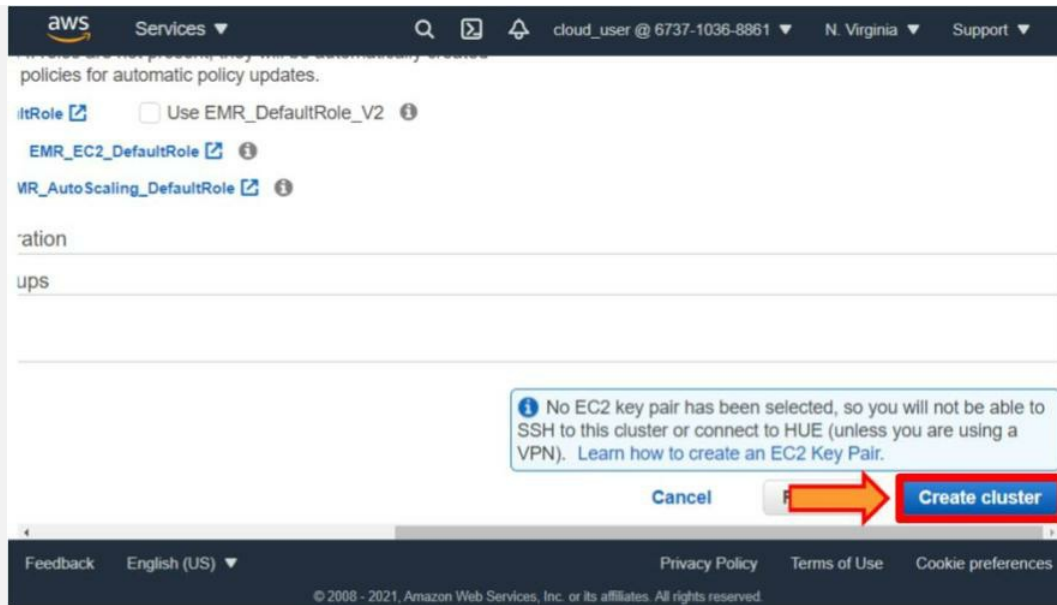
Role  ⓘ

3

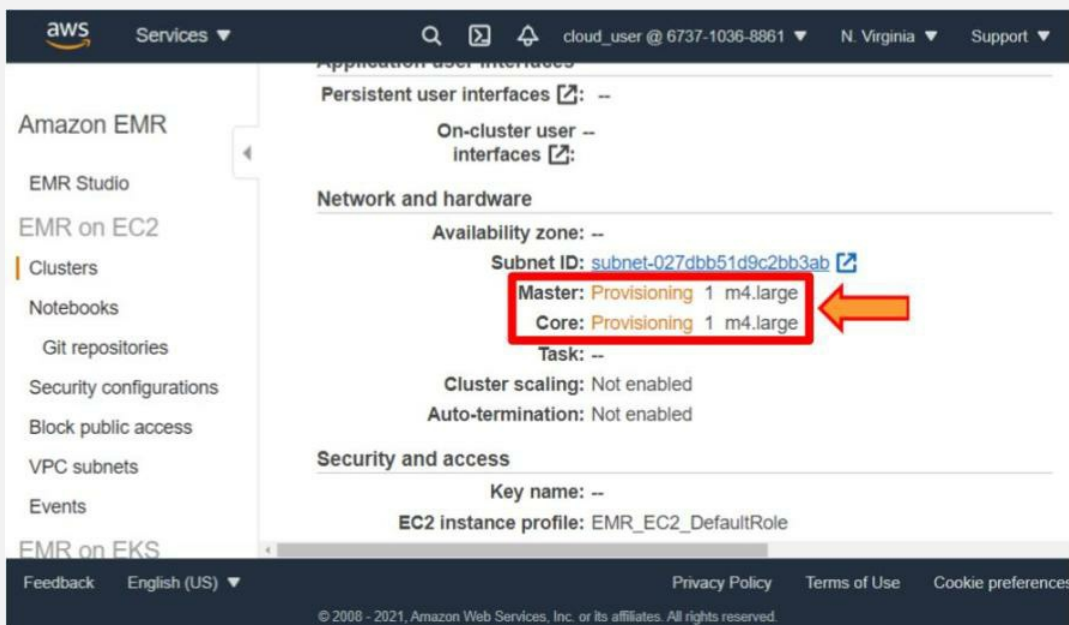
[Cancel](#) **Next**

Feedback English (US) ▼ Privacy Policy Terms of Use Cookie preferences  
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

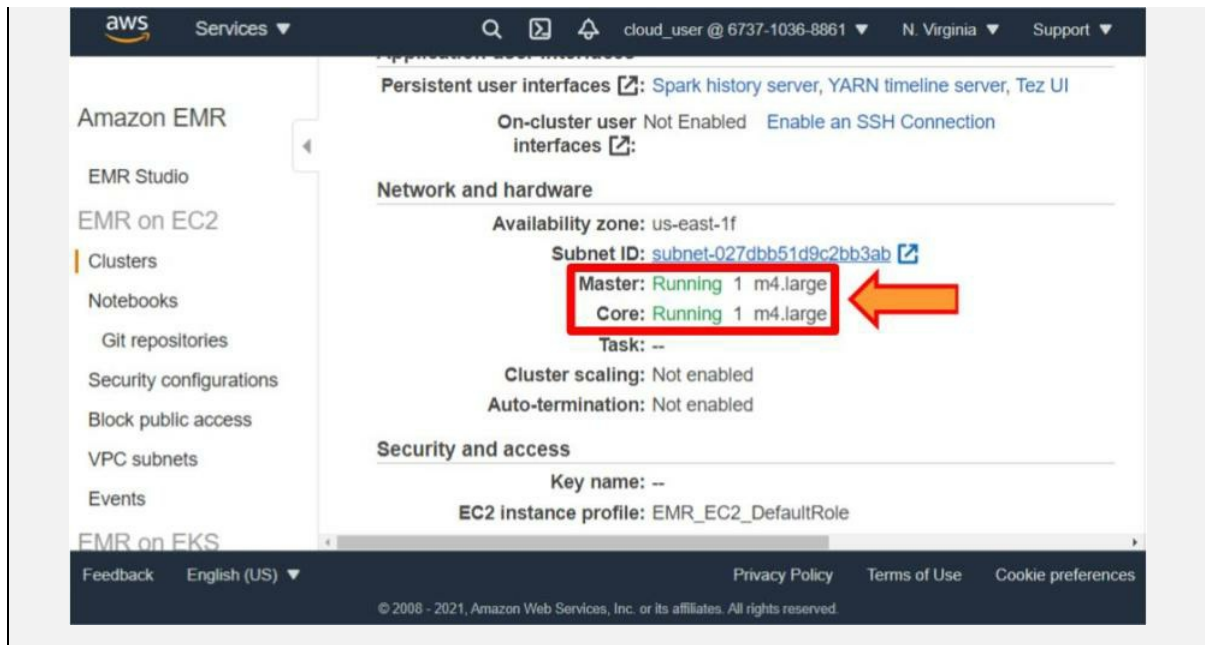
16. Click on the **Create Cluster** button.



17. It will take 10 to 15 minutes to create a cluster.

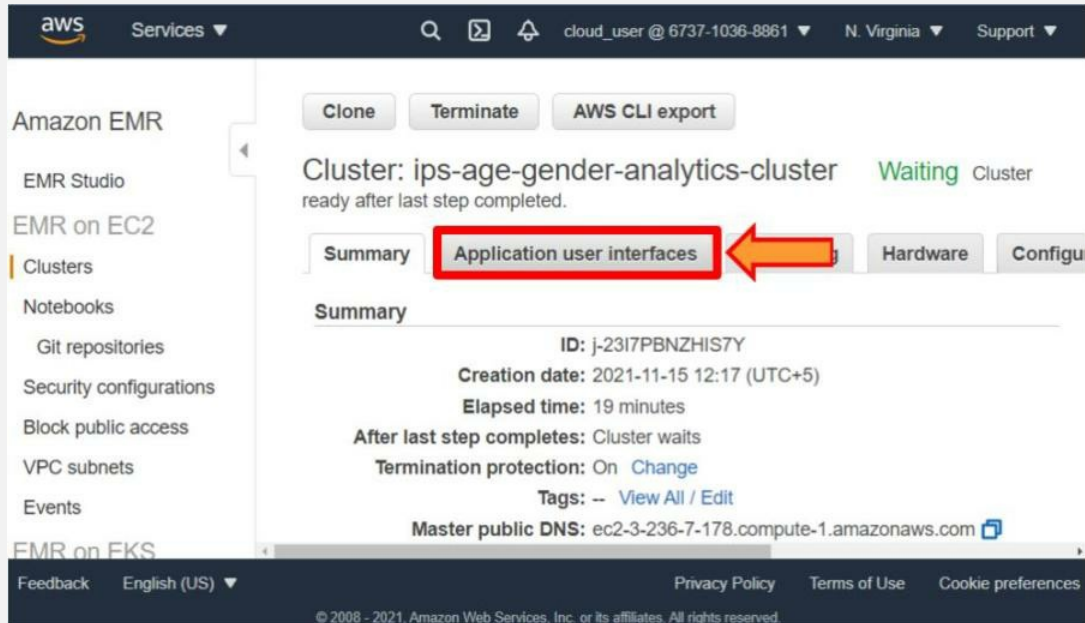


18. Hence, successfully created the AWS EMR cluster.



## Step 2: Open the HDFS Port

1. Click on the **Application user interfaces** tab.



2. Copy the **Port Number** from the HDFS Name Node URL.

Amazon EMR console showing the 'On-cluster application user interfaces' page. The page lists various applications and their user interface URLs. An orange arrow points to the 'HDFS Name Node' entry.

| Application          | User interface URL  | Status                 |
|----------------------|---|------------------------|
| HDFS Name Node       | <a href="http://ec2-3-236-7-178.compute-1.amazonaws.com:50070/">http://ec2-3-236-7-178.compute-1.amazonaws.com:50070/</a> | SSH tunnel not enabled |
| Hue                  | <a href="http://ec2-3-236-7-178.compute-1.amazonaws.com:8888/">http://ec2-3-236-7-178.compute-1.amazonaws.com:8888/</a>   | SSH tunnel not enabled |
| Spark History Server | <a href="http://ec2-3-236-7-178.compute-1.amazonaws.com:18080/">http://ec2-3-236-7-178.compute-1.amazonaws.com:18080/</a> | SSH tunnel not enabled |
| Resource Manager     | <a href="http://ec2-3-236-7-178.compute-1.amazonaws.com:8088/">http://ec2-3-236-7-178.compute-1.amazonaws.com:8088/</a>   | SSH tunnel not enabled |

The following table lists web interfaces you can view on the task nodes:

| Application    | User interface URL  |
|----------------|---|
| HDFS Data Node | <a href="http://ec2-000-000-000-000.compute-1.amazonaws.com:50075/">http://ec2-000-000-000-000.compute-1.amazonaws.com:50075/</a> |
| Node Manager   | <a href="http://ec2-000-000-000-000.compute-1.amazonaws.com:8042/">http://ec2-000-000-000-000.compute-1.amazonaws.com:8042/</a>   |

3. Click on the **Summary** tab.

Amazon EMR console showing the 'Summary' tab for a cluster. The cluster is named 'ips-age-gender-analytics-cluster' and is in a 'Waiting' state. An orange arrow points to the 'Summary' tab.

Cluster: ips-age-gender-analytics-cluster **Waiting** Cluster ready after last step completed.

Summary Application user interfaces Monitoring Hardware Configurati

Persistent application user interfaces

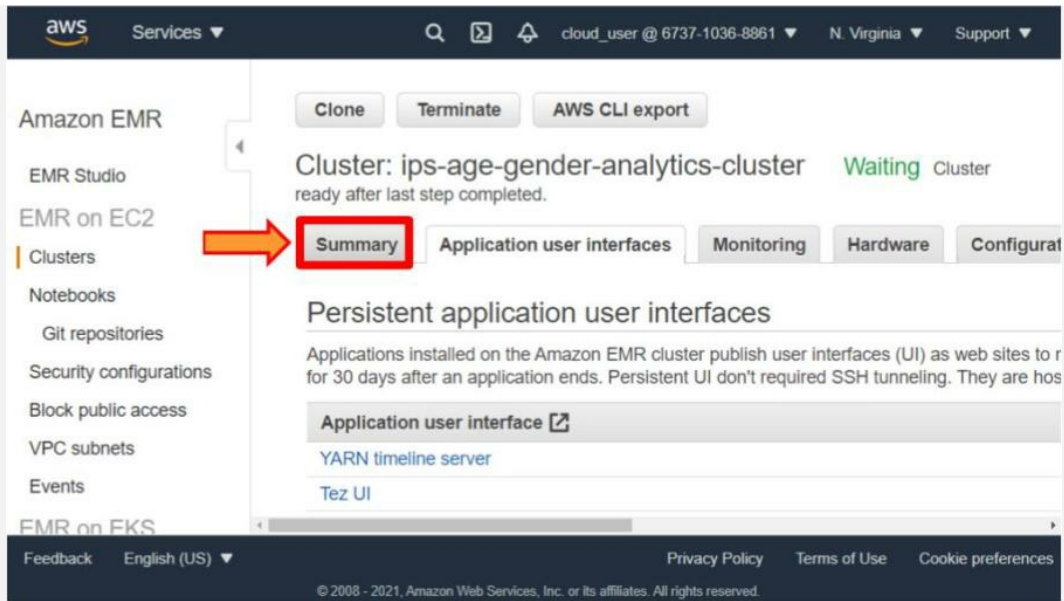
Applications installed on the Amazon EMR cluster publish user interfaces (UI) as web sites to r for 30 days after an application ends. Persistent UI don't required SSH tunneling. They are hos

Application user interface

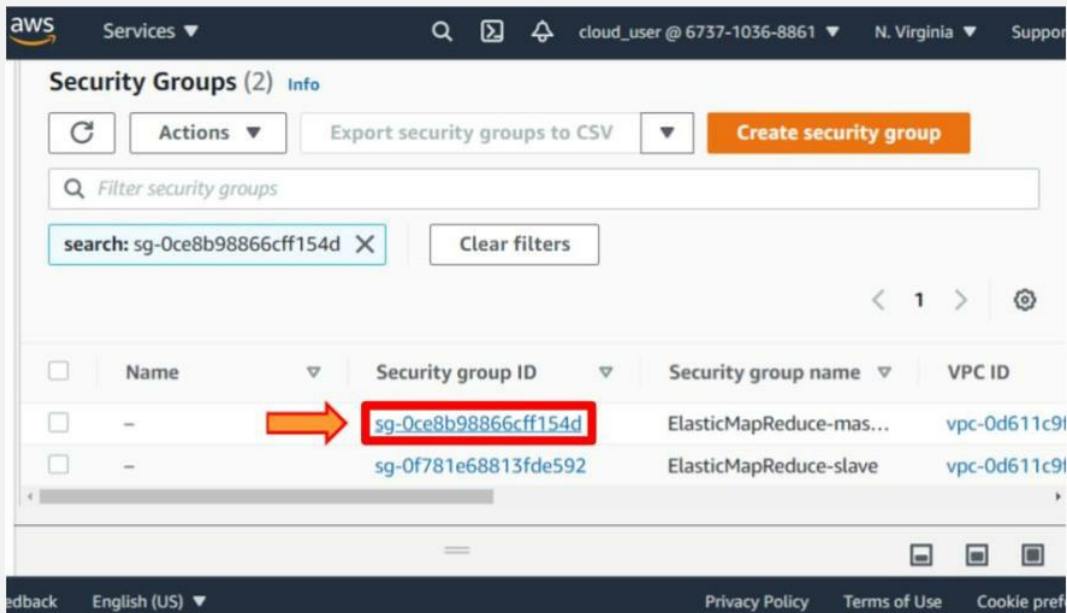
YARN timeline server

Tez UI

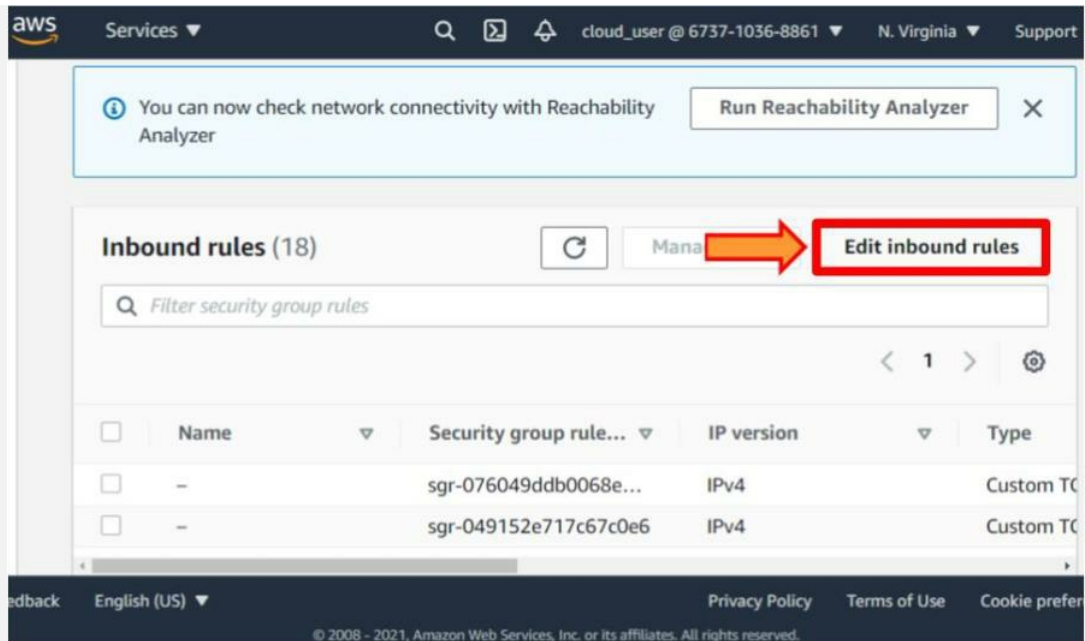
4. Click on the **Security Groups for Master URL** in the new browser tab.



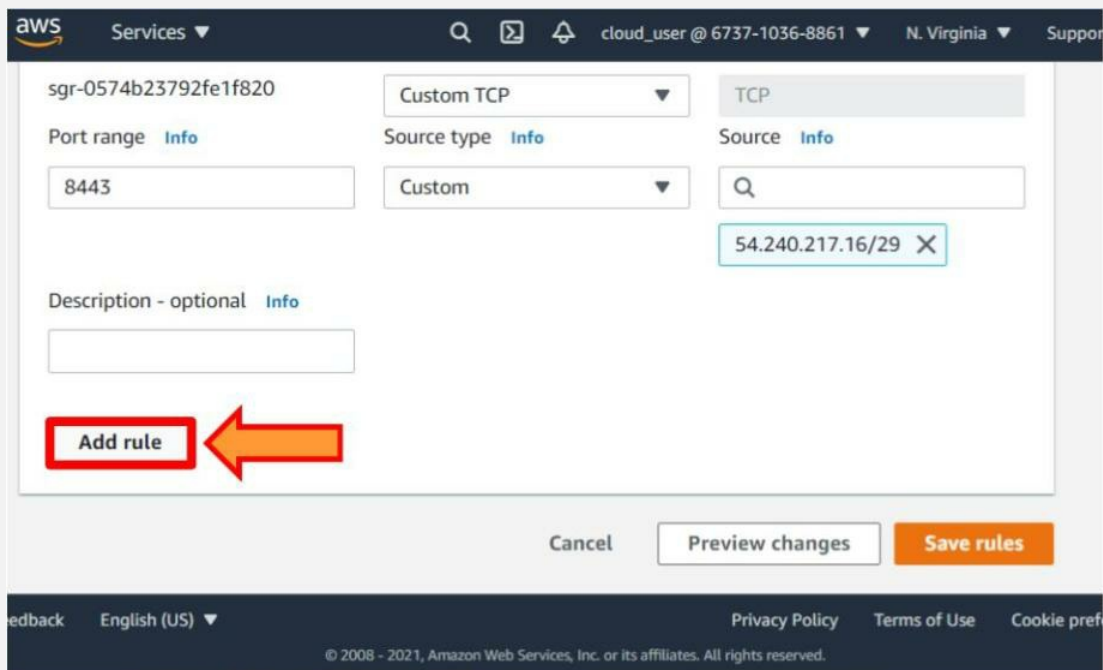
5. Select the **Elastic Map reduce-master** security group ID link.



6. Click on the **Edit inbound rules** button.



7. Scroll down. Click on the **Add rule** button.



8. Leave the type as the **Custom TCP**.
9. Paste the copied HDFS **Port Number**.
10. Select the **o.o.o.o/o** IP address.

aws Services cloud\_user @ 6737-1036-8861 N. Virginia Support

Inbound rule 19 Delete

Security group rule ID -

Type Info **Custom TCP**

Protocol Info TCP

Port range Info **50070**

Source type Info Anywhere-IPv4

Source Info 0.0.0.0/0 X

Description - optional Info

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

11. Click on the **Save rules** button.

aws Services cloud\_user @ 6737-1036-8861 N. Virginia Support

- Custom TCP TCP

Port range Info 50070

Source type Info Anywhere-IPv4

Source Info 0.0.0.0/0 X

Description - optional Info

Add rule

Cancel Preview **Save rules**

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

12. Close the Security Groups tab and go to the **AWS EMR dashboard**.

### Step 3: Copy Data from S3 to HDFS Using s3-dist-cp Command

1. Click on the **Application user interfaces** tab.

The screenshot shows the Amazon EMR console interface. On the left, a navigation menu lists various services like EMR Studio, EMR on EC2, Clusters, Notebooks, etc. The main content area displays details for a cluster named 'ips-age-gender-analytics-cluster'. The cluster is in a 'Waiting' state. A red box highlights the 'Application user interfaces' tab, with an orange arrow pointing to it from the right. Below the tabs, a 'Summary' section provides key information: ID (j-23I7PBNZHIS7Y), Creation date (2021-11-15 12:17 UTC+5), Elapsed time (19 minutes), and Master public DNS (ec2-3-236-7-178.compute-1.amazonaws.com).

2. Open the **HDFS Name Node URL** on the new tab.

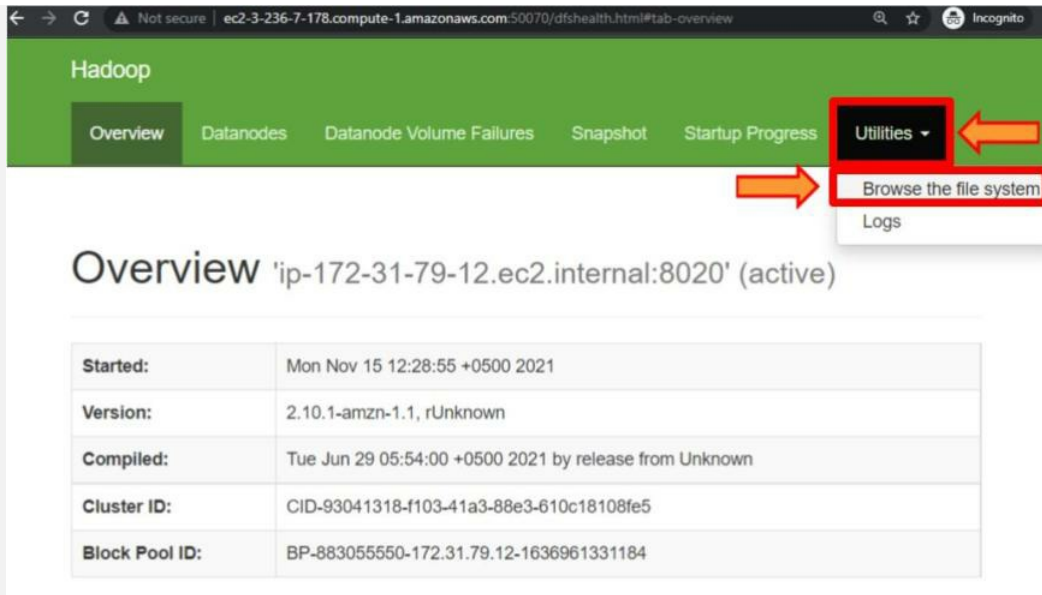
This screenshot shows the 'On-cluster application user interfaces' page. It explains that on-cluster UIs are only available while clusters are running and require SSH tunneling. A table lists several applications and their corresponding user interface URLs. The 'HDFS Name Node' row is highlighted with a red box, and an orange arrow points to it from the left. Below this table, another section lists web interfaces available on task nodes.

| Application          | User interface URL                                    | Status                 |
|----------------------|---|------------------------|
| HDFS Name Node       | http://ec2-3-236-7-178.compute-1.amazonaws.com:50070/ | SSH tunnel not enabled |
| Hue                  | http://ec2-3-236-7-178.compute-1.amazonaws.com:8888/  | SSH tunnel not enabled |
| Spark History Server | http://ec2-3-236-7-178.compute-1.amazonaws.com:18080/ | SSH tunnel not enabled |
| Resource Manager     | http://ec2-3-236-7-178.compute-1.amazonaws.com:8088/  | SSH tunnel not enabled |

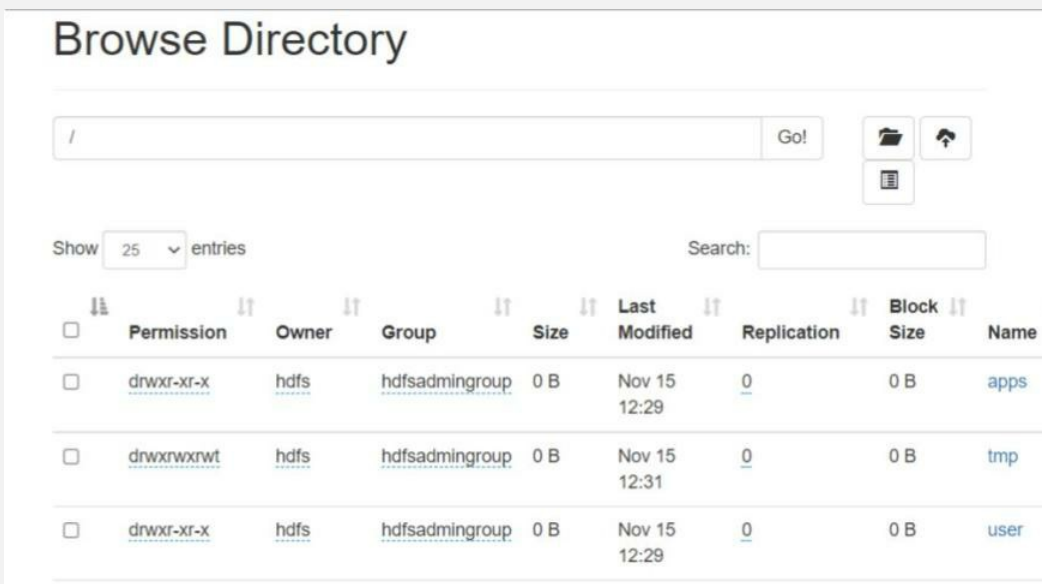
  

| Application    | User interface URL  |
|----------------|---|
| HDFS Data Node | http://ec2-000-000-000-000.compute-1.amazonaws.com:50075/ |
| Node Manager   | http://ec2-000-000-000-000.compute-1.amazonaws.com:8042/  |

3. Click on the **Utilities**. Then click on the **Browser the file system**.

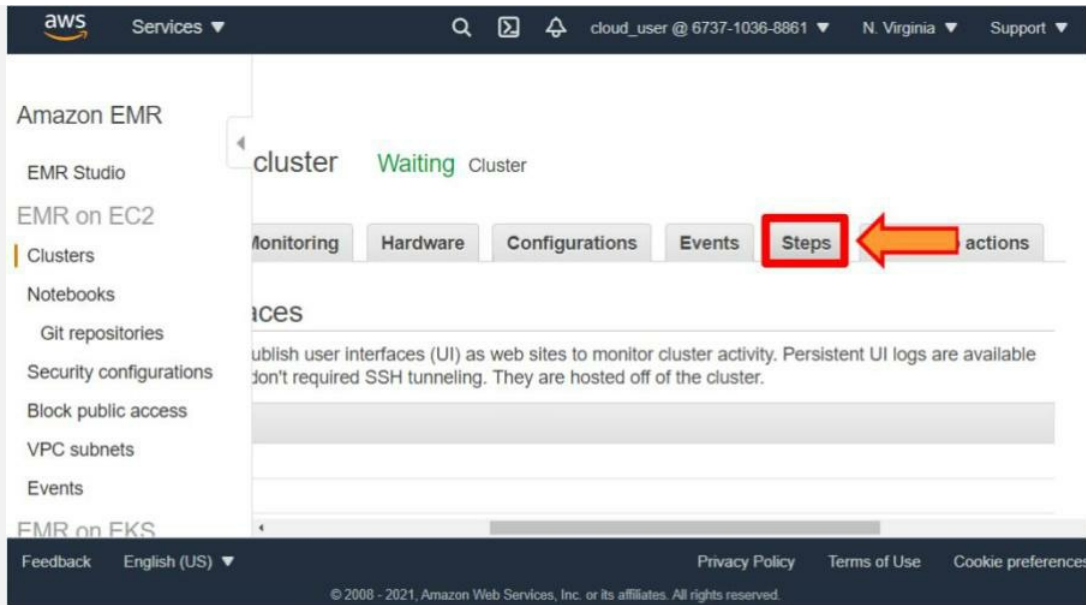


4. You will see the different folders.

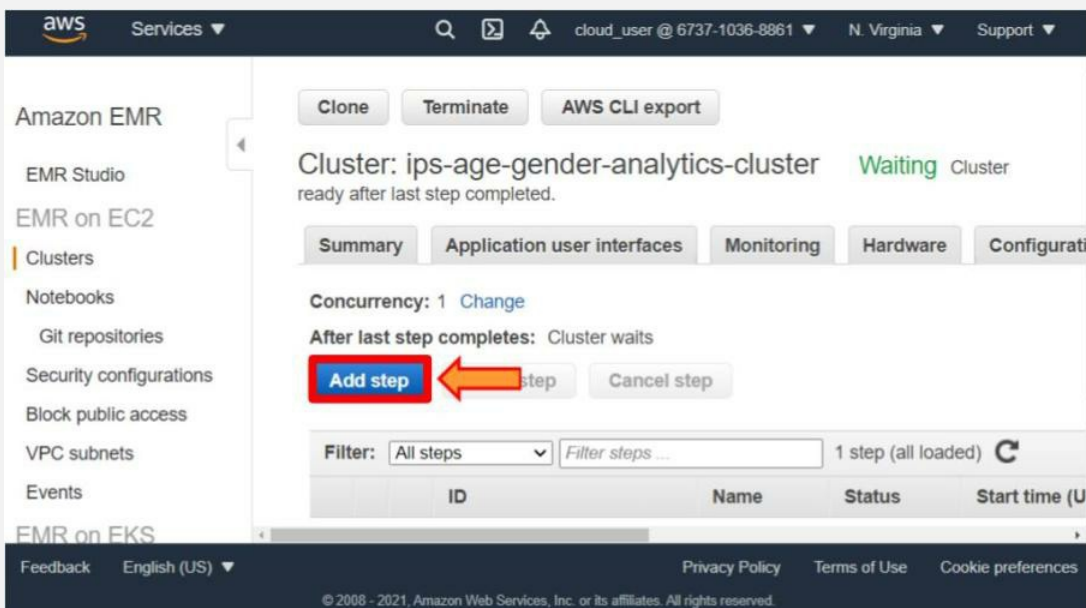


5. Go back to the AWS EMR Cluster dashboard.

6. Click on the **Steps** tab.




7. Click on the **Add Step** button.




8. Select the **Custom JAR** step type.


**Add step** ✕

**Step type** Custom JAR 


**Name\*** IPS Copy data and script to HDFS

**JAR location\*** s3:// 

JAR location maybe a path into S3 or a fully qualified java class in the classpath.

**Arguments** 

These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.

**Action on failure** Continue 


What happens if the step fails


Cancel Add

9. Give a name **ISP Copy data and script to HDFS.**


**Add step** ✕

**Step type** Custom JAR


**Name\*** IPS Copy data and script to HDFS 

**JAR location\*** s3:// 

JAR location maybe a path into S3 or a fully qualified java class in the classpath.

**Arguments** 

These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.

**Action on failure** Continue 

What happens if the step fails



Cancel Add

10. In the JAR location, copy and paste the following command **command-runner.jar**.

**Add step** [X]

Step type: Custom JAR

Name\*: IPS Copy data and script to HDFS

JAR location\*: command-runner.jar  

JAR location maybe a path into S3 or a fully qualified java class in the classpath.

Arguments:

These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.

Action on failure: Continue

What happens if the step fails


Cancel Add

11. In the argument field copy and paste the following command  
**s3-dist-cp --src=s3://das-c01-data-analytics-specialty/Data\_Analytics\_With\_Spark\_and\_EMR/ --dest=hdfs: ///**


**Add step** [X]

Step type: Custom JAR

Name\*: IPS Copy data and script to HDFS

JAR location\*: command-runner.jar 

JAR location maybe a path into S3 or a fully qualified java class in the classpath.

Arguments: s3-dist-cp --src=s3://das-c01-data-analytics-specialty/Data\_Analytics\_With\_Spark\_and\_EMR/ --dest=hdfs: /// 

These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.

Action on failure: Continue

What happens if the step fails

Cancel Add

12. Select the **Continue** in action on failure.

**Add step** [X]

Step type: Custom JAR

Name\*: IPS Copy data and script to HDFS

JAR location\*: command-runner.jar

JAR location maybe a path into S3 or a fully qualified java class in the classpath.

Arguments: s3-dist-cp --src=s3://das-c01-data-analytics-specialty/Data\_Analytics\_With\_Spark\_and\_EMR/ --dest=hdfs:///

These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.

Action on failure: Continue

What happens if the step fails

Cancel Add

13. Click on the **Add** button.

**Add step** [X]

Step type: Custom JAR

Name\*: IPS Copy data and script to HDFS

JAR location\*: command-runner.jar

JAR location maybe a path into S3 or a fully qualified java class in the classpath.

Arguments: s3-dist-cp --src=s3://das-c01-data-analytics-specialty/Data\_Analytics\_With\_Spark\_and\_EMR/ --dest=hdfs:///

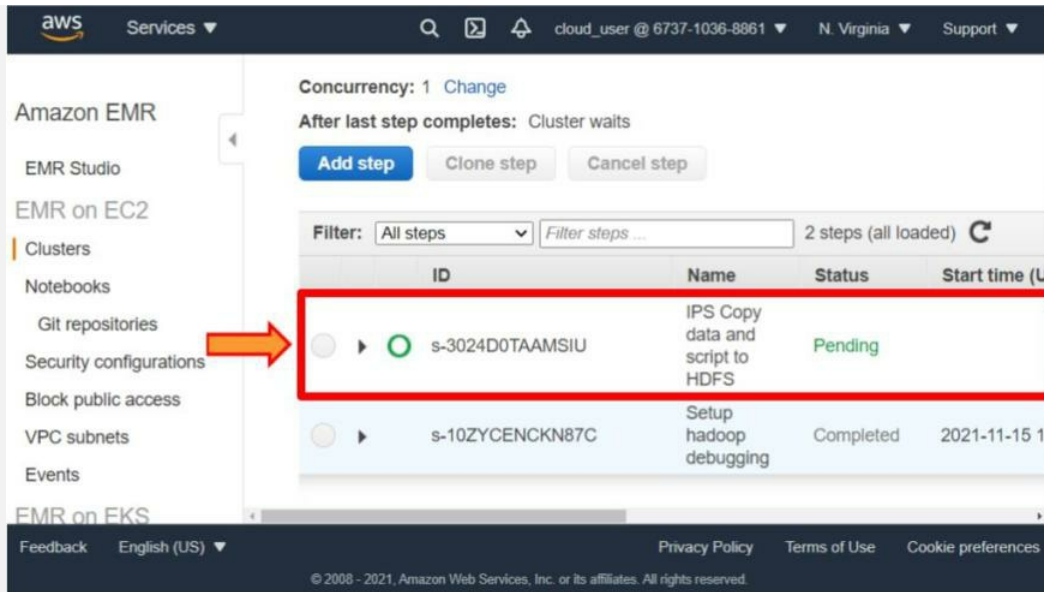
These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.

Action on failure: Continue

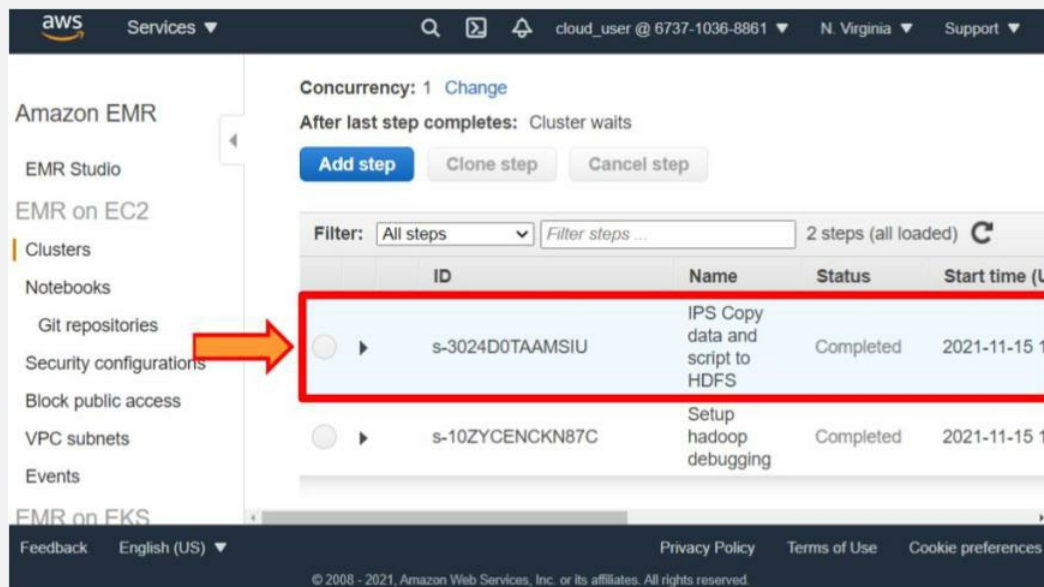
What happens if the step fails

Cancel Add

14. It will take a few minutes to complete. The step copies data from the S3 bucket onto the Hadoop cluster.

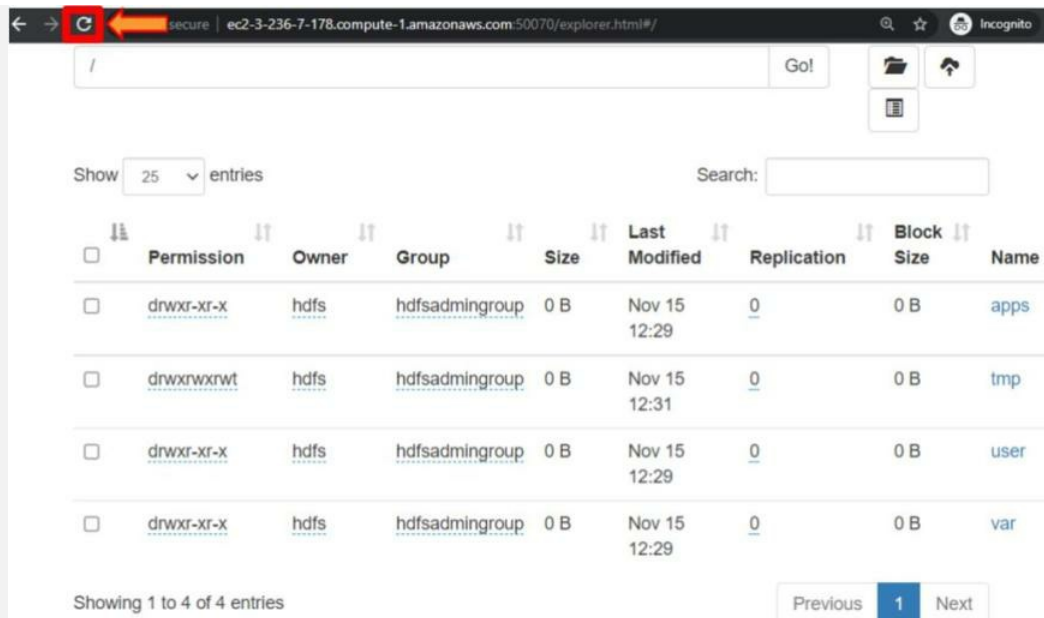


15. Hence, it completes the step.

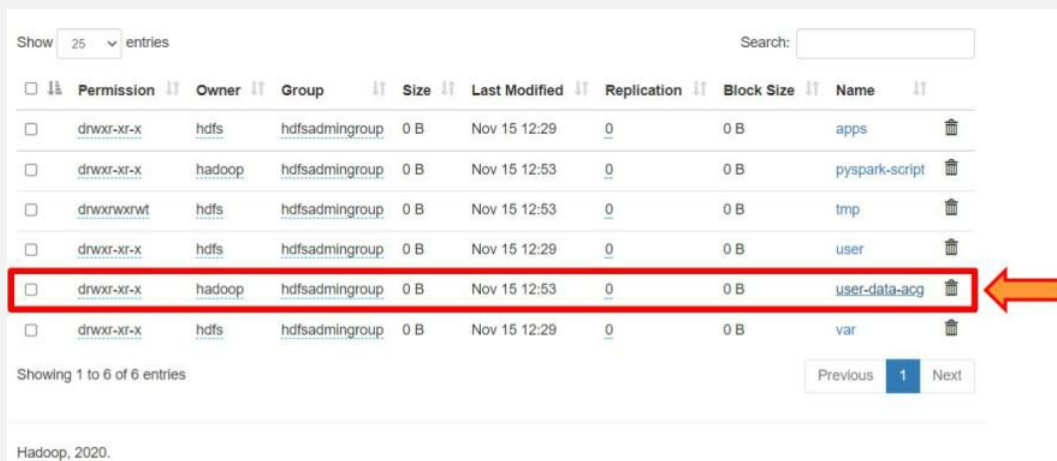


## Step 4: Run a PySpark Script Using spark-submit Command




1. Go back to the **Hadoop HDHS** tab. Click on the **Refresh** button.







2. Click on the **user-data-acg** folder.



3. The folder contains hundreds of CSV files.








/user-data-acg Go!   

Show 25 entries Search:

| <input type="checkbox"/> | Permission | Owner  | Group          | Size    | Last Modified | Replication | Block Size | Name   |  |
|--------------------------|------------|--------|----------------|---------|---------------|-------------|------------|--|---|
| <input type="checkbox"/> | -rw-r--r-- | hadoop | hdfsadmingroup | 5.86 KB | Nov 15 12:52  | 1           | 128 MB     | user-data-012188bc-a4e5-4d77-ae73-eb8082ea4e6d.csv |  |
| <input type="checkbox"/> | -rw-r--r-- | hadoop | hdfsadmingroup | 6.63 KB | Nov 15 12:53  | 1           | 128 MB     | user-data-0160ad6f-b4ba-4bba-a55e-2a5e305bd8c1.csv |  |
| <input type="checkbox"/> | -rw-r--r-- | hadoop | hdfsadmingroup | 7.05 KB | Nov 15 12:52  | 1           | 128 MB     | user-data-01de9ac9-89b5-4ebd-b498-f829ef1720f8.csv |  |

4. Go back and Click on the **pyspark-script** folder.

Show 25 entries Search:




| <input type="checkbox"/> | Permission | Owner  | Group          | Size | Last Modified | Replication | Block Size | Name           |    |
|--------------------------|------------|--------|----------------|------|---------------|-------------|------------|----------------|---|
| <input type="checkbox"/> | drwxr-xr-x | hdfs   | hdfsadmingroup | 0 B  | Nov 15 12:29  | 0           | 0 B        | apps           |    |
| <input type="checkbox"/> | drwxr-xr-x | hadoop | hdfsadmingroup | 0 B  | Nov 15 12:53  | 0           | 0 B        | pyspark-script |    |
| <input type="checkbox"/> | drwxrwxrwt | hdfs   | hdfsadmingroup | 0 B  | Nov 15 12:53  | 0           | 0 B        | tmp            |   |
| <input type="checkbox"/> | drwxr-xr-x | hdfs   | hdfsadmingroup | 0 B  | Nov 15 12:29  | 0           | 0 B        | user           |  |
| <input type="checkbox"/> | drwxr-xr-x | hadoop | hdfsadmingroup | 0 B  | Nov 15 12:53  | 0           | 0 B        | user-data-acg  |  |
| <input type="checkbox"/> | drwxr-xr-x | hdfs   | hdfsadmingroup | 0 B  | Nov 15 12:29  | 0           | 0 B        | var            |  |

Showing 1 to 6 of 6 entries Previous 1 Next



Hadoop, 2020.

5. The folder contains the PySpark script.

### Browse Directory

/pyspark-script Go!   

Show 25 entries Search:

| <input type="checkbox"/> | Permission | Owner  | Group          | Size  | Last Modified | Replication | Block Size | Name                |  |
|--------------------------|------------|--------|----------------|-------|---------------|-------------|------------|---------------------|---|
| <input type="checkbox"/> | -rw-r--r-- | hadoop | hdfsadmingroup | 480 B | Nov 15 12:53  | 1           | 128 MB     | emr-pyspark-code.py |  |

Showing 1 to 1 of 1 entries Previous 1 Next

Hadoop, 2020.

6. Below is the figure of PySpark python code.

```
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession

spark = SparkSession.builder.master("local")
    .config(conf=SparkConf()).getOrCreate()

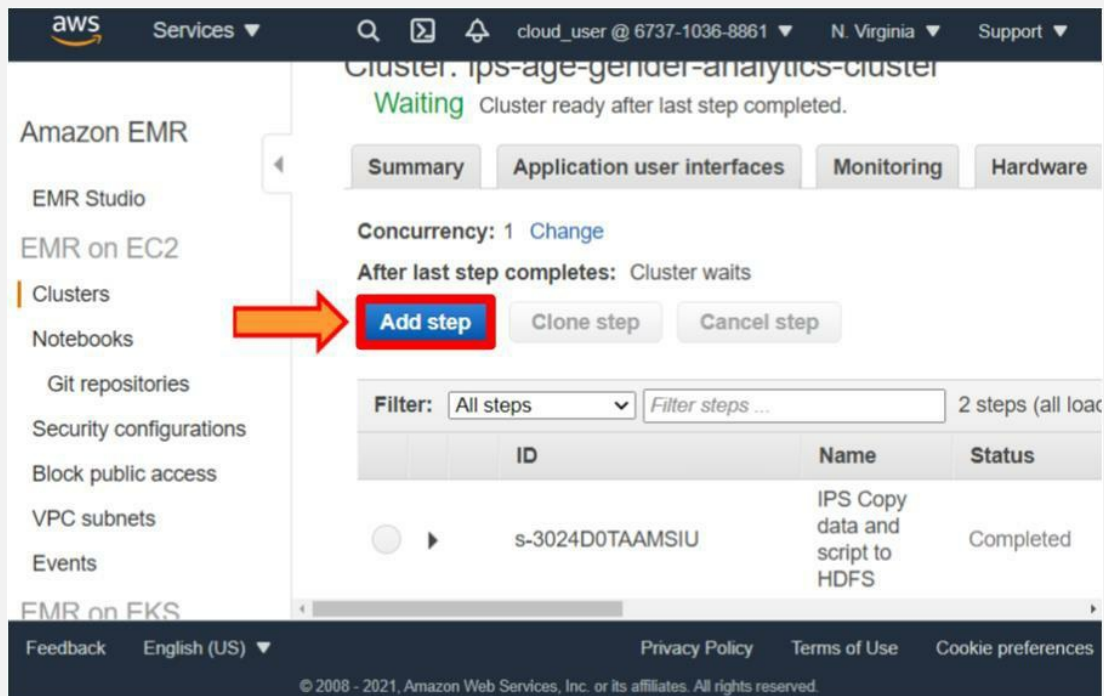
df = spark.read.format("csv")
    .option("header", "true")
    .load("hdfs:///user-data-acg/user-data-*.csv")

results = df.groupBy("`dob.age`", "`gender`")
    .count()
    .orderBy("count", ascending=False)

results.show()

results.coalesce(1).write.csv("hdfs:///results", sep=",", header="true")
```

7. Go back to the AWS EMR dashboard. Click on the **Add Step** button.



The screenshot shows the AWS EMR console interface. The top navigation bar includes the AWS logo, 'Services', a search icon, a user profile 'cloud\_user @ 6737-1036-8861', the region 'N. Virginia', and a 'Support' link. The main content area is titled 'Cluster: ips-age-gender-analytics-cluster' and shows a 'Waiting' status with the message 'Cluster ready after last step completed.' Below this are tabs for 'Summary', 'Application user interfaces', 'Monitoring', and 'Hardware'. The 'Summary' tab is active, displaying 'Concurrency: 1' with a 'Change' link, and 'After last step completes: Cluster waits'. A red box highlights the 'Add step' button, with an orange arrow pointing to it from the left. To the right of 'Add step' are 'Clone step' and 'Cancel step' buttons. Below these buttons is a table of steps. The table has columns for 'Filter', 'ID', 'Name', and 'Status'. The first step listed has ID 's-3024D0TAAMSIU', Name 'IPS Copy data and script to HDFS', and Status 'Completed'. The footer of the console includes 'Feedback', 'English (US)', 'Privacy Policy', 'Terms of Use', 'Cookie preferences', and a copyright notice '© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.'

| Filter    | ID              | Name                             | Status    |
|-----------|-----------------|----------------------------------|-----------|
| All steps | s-3024D0TAAMSIU | IPS Copy data and script to HDFS | Completed |

8. Select the **Custom JAR** step type.

Add step

Step type Custom JAR

Name\* Custom JAR

JAR location\* s3://

JAR location maybe a path into S3 or a fully qualified java class in the classpath.

Arguments

These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.

Action on failure Continue

What happens if the step fails

Cancel Add

9. Give a name IPS Run PySpark script.

Add step

Step type Custom JAR

Name\* IPS Run PySpark script

JAR location\* s3://

JAR location maybe a path into S3 or a fully qualified java class in the classpath.

Arguments

These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.

Action on failure Continue

What happens if the step fails


Cancel Add

10. In the JAR location, copy and paste the following command **command-runner.jar**.

**Add step** [X]

Step type: Custom JAR

Name\*: IPS Run PySpark script

JAR location\*: **command-runner.jar** 

JAR location maybe a path into S3 or a fully qualified java class in the classpath.

Arguments: 

These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.

Action on failure: Continue

What happens if the step fails


Cancel Add

11. In the argument field, copy and paste the following command **spark-submit hdfs:///pyspark-script/emr-pyspark-code.py**


**Add step** [X]

Step type: Custom JAR

Name\*: IPS Run PySpark script

JAR location\*: command-runner.jar 

JAR location maybe a path into S3 or a fully qualified java class in the classpath.

Arguments: **spark-submit hdfs:///pyspark-script/emr-pyspark-code.py** 

These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.

Action on failure: Continue

What happens if the step fails

Cancel Add

12. Select **Continue** in Action on failure.

**Add step** [X]

**Step type** Custom JAR

**Name\*** IPS Run PySpark script

**JAR location\*** command-runner.jar

JAR location maybe a path into S3 or a fully qualified java class in the classpath.

**Arguments** spark-submit hdfs:///pyspark-script/emr-pyspark-code.py

These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.

**Action on failure** Continue

What happens if the step fails

Cancel Add

13. Click on the **Add** button.

**Add step** [X]

**Step type** Custom JAR

**Name\*** IPS Run PySpark script

**JAR location\*** command-runner.jar

JAR location maybe a path into S3 or a fully qualified java class in the classpath.

**Arguments** spark-submit hdfs:///pyspark-script/emr-pyspark-code.py

These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.

**Action on failure** Continue

What happens if the step fails

Add

14. The step takes some time to run the PySpark script.

Amazon EMR

EMR Studio

EMR on EC2

Clusters

Notebooks

Git repositories

Security configurations

Block public access

VPC subnets

Events

EMR on FKS

Concurrency: 1 [Change](#)

After last step completes: Cluster waits

[Add step](#) [Clone step](#) [Cancel step](#)

Filter: All steps  3 steps (all lo

| ID              | Name                             | Status    |
|-----------------|----------------------------------|-----------|
| s-38XRBH28VISO3 | IPS Run PySpark script           | Pending   |
| s-3024D0TAAMSIU | IPS Copy data and script to HDFS | Completed |
|                 | Setup                            |           |

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

15. After the step is completed, click on the **View logs** of the **IPS Run PySpark script**.

Amazon EMR

EMR Studio

EMR on EC2

Clusters

Notebooks

Git repositories

Security configurations

Block public access

VPC subnets

Events

EMR on FKS

View Jobs in the Application History Tab

| TC+5)        | Elapsed time | Log files <a href="#">Log files</a> |
|--------------|--------------|-------------------------------------|
| 3:04 (UTC+5) | 38 seconds   | <a href="#">View logs</a>           |
| 2:51 (UTC+5) | 2 minutes    | <a href="#">View logs</a>           |
| 2:34 (UTC+5) | 8 seconds    | <a href="#">View logs</a>           |

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.


16. Click on the **stdout**.

aws Services 🔻 🔍 📄 🔔 cloud\_user @ 6737-1036-8861 🔻 N. Virginia 🔻 Support 🔻

Amazon EMR

- EMR Studio
- EMR on EC2
  - Clusters
  - Notebooks
  - Git repositories
  - Security configurations
  - Block public access
  - VPC subnets
  - Events
- EMR on FKS

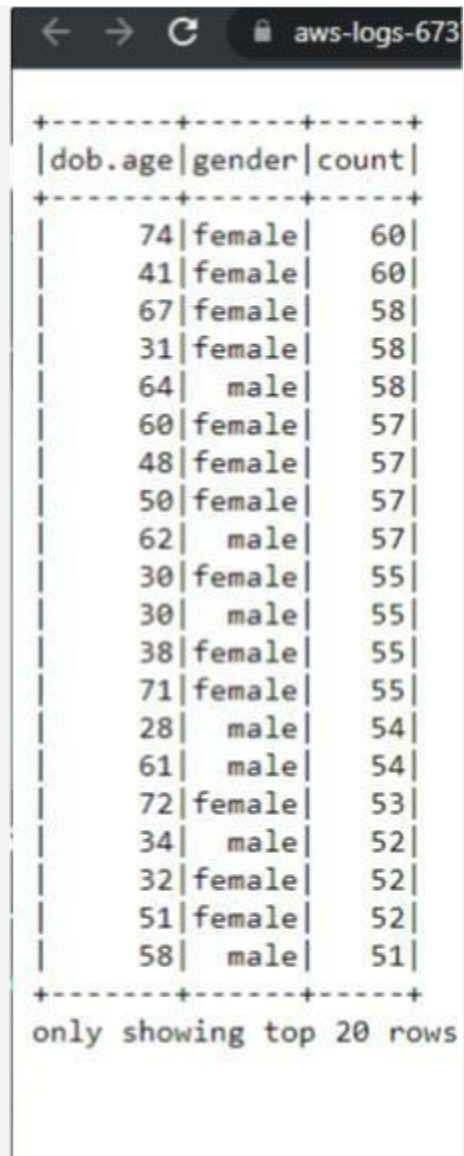
View Jobs in the Application History Tab

| TC+5) 🔻      | Elapsed time | Log files 📄  |
|--------------|--------------|--|
| 3:04 (UTC+5) | 38 seconds   | controller   sys  <span>stdout</span> 🔄 |
| 2:51 (UTC+5) | 2 minutes    | <a href="#">View logs</a>  |
| 2:34 (UTC+5) | 8 seconds    | <a href="#">View logs</a>  |

Feedback English (US) 🔻 Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

17. The list shows 20 results, and the top shows the most common age and gender.

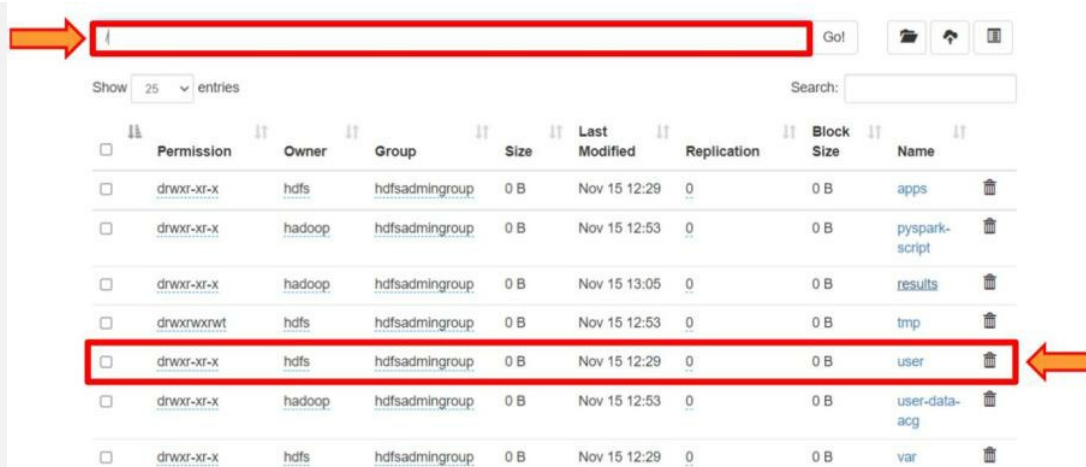


The screenshot shows a terminal window with a dark header bar containing navigation icons and the text 'aws-logs-673'. The main content is a table with three columns: 'dob.age', 'gender', and 'count'. The table is enclosed in a dashed border. Below the table, the text 'only showing top 20 rows' is displayed.

| dob.age | gender | count |
|---------|--------|-------|
| 74      | female | 60    |
| 41      | female | 60    |
| 67      | female | 58    |
| 31      | female | 58    |
| 64      | male   | 58    |
| 60      | female | 57    |
| 48      | female | 57    |
| 50      | female | 57    |
| 62      | male   | 57    |
| 30      | female | 55    |
| 30      | male   | 55    |
| 38      | female | 55    |
| 71      | female | 55    |
| 28      | male   | 54    |
| 61      | male   | 54    |
| 72      | female | 53    |
| 34      | male   | 52    |
| 32      | female | 52    |
| 51      | female | 52    |
| 58      | male   | 51    |

only showing top 20 rows

18. Go back to the **Hadoop HDFS** tab and view the root directory by entering / into the search bar.

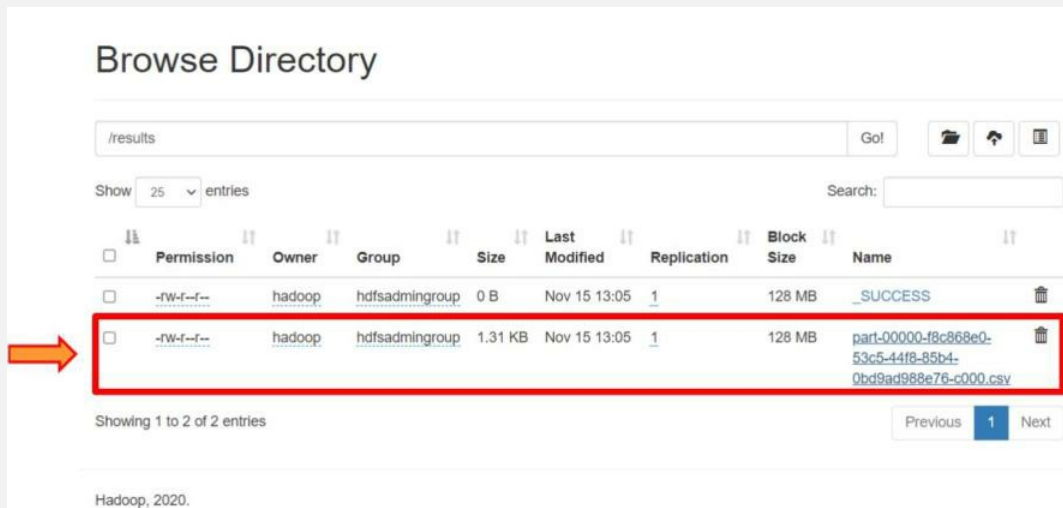


Go!

Show 25 entries Search:

| Permission | Owner  | Group     | Size | Last Modified | Replication | Block Size | Name           |
|------------|--------|-----------|------|---------------|-------------|------------|----------------|
| drwxr-xr-x | hdfs   | hdfsadmin | 0 B  | Nov 15 12:29  | 0           | 0 B        | apps           |
| drwxr-xr-x | hadoop | hdfsadmin | 0 B  | Nov 15 12:53  | 0           | 0 B        | pyspark-script |
| drwxr-xr-x | hadoop | hdfsadmin | 0 B  | Nov 15 13:05  | 0           | 0 B        | results        |
| drwxrwxrwt | hdfs   | hdfsadmin | 0 B  | Nov 15 12:53  | 0           | 0 B        | tmp            |
| drwxr-xr-x | hdfs   | hdfsadmin | 0 B  | Nov 15 12:29  | 0           | 0 B        | user           |
| drwxr-xr-x | hadoop | hdfsadmin | 0 B  | Nov 15 12:53  | 0           | 0 B        | user-data-acg  |
| drwxr-xr-x | hdfs   | hdfsadmin | 0 B  | Nov 15 12:29  | 0           | 0 B        | var            |

19. The folder contains the CSV file.



Browse Directory

/results Go!

Show 25 entries Search:

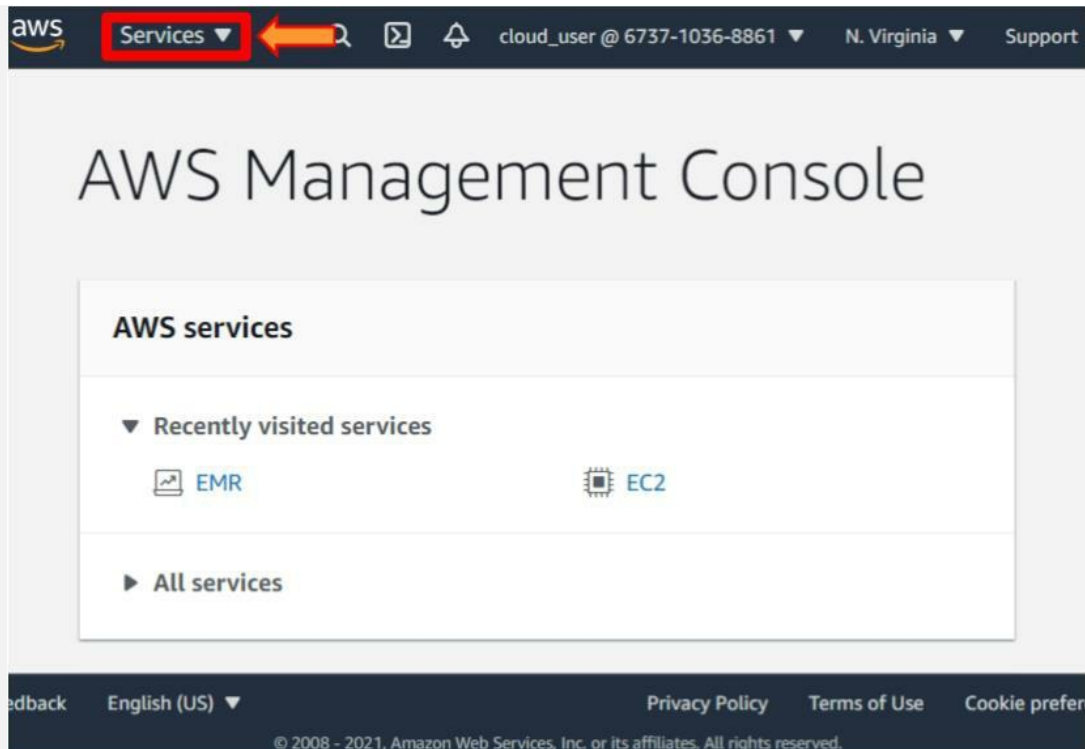
| Permission | Owner  | Group     | Size    | Last Modified | Replication | Block Size | Name   |
|------------|--------|-----------|---------|---------------|-------------|------------|--|
| -rw-r--r-- | hadoop | hdfsadmin | 0 B     | Nov 15 13:05  | 1           | 128 MB     | _SUCCESS   |
| -rw-r--r-- | hadoop | hdfsadmin | 1.31 KB | Nov 15 13:05  | 1           | 128 MB     | part-00000-f8c868e0-53c5-44f8-85b4-0bd9ad988e76-c000.csv |

Showing 1 to 2 of 2 entries Previous 1 Next

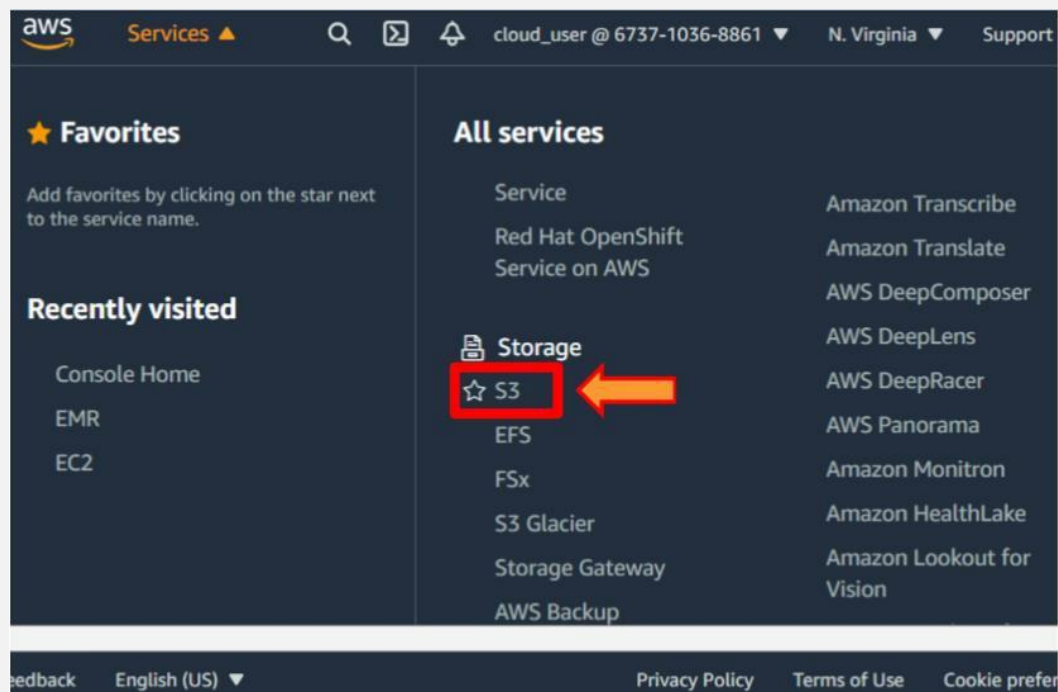
Hadoop, 2020.

## Step 5: Copy Data from HDFS to S3 Using s3-dist-cp Command

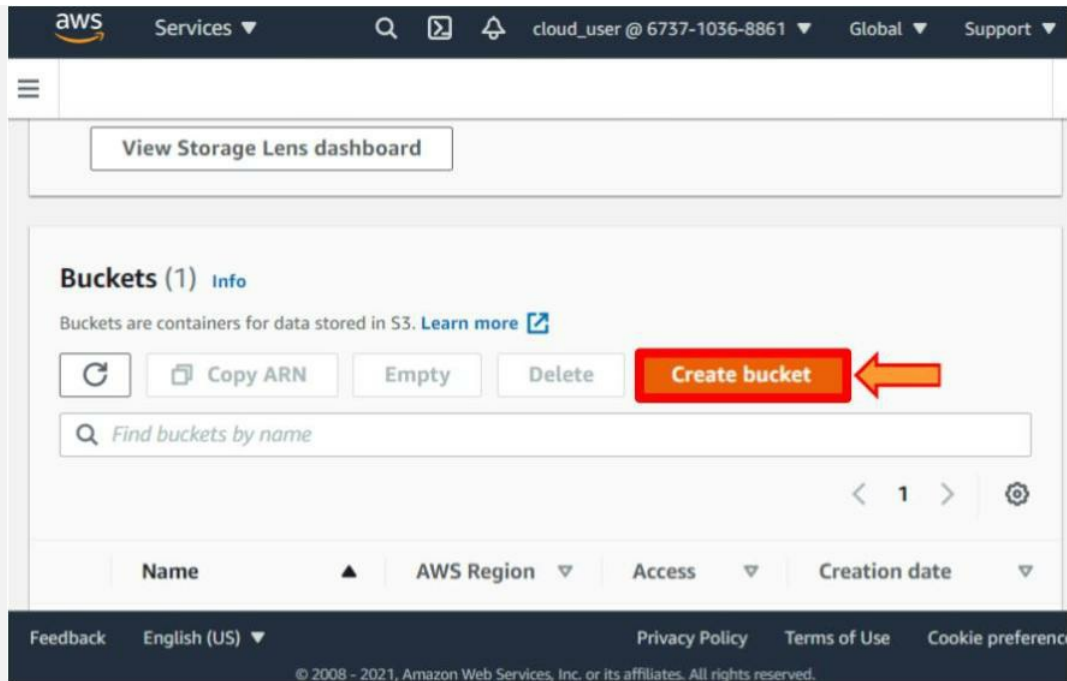
1. Click on the **Services**.



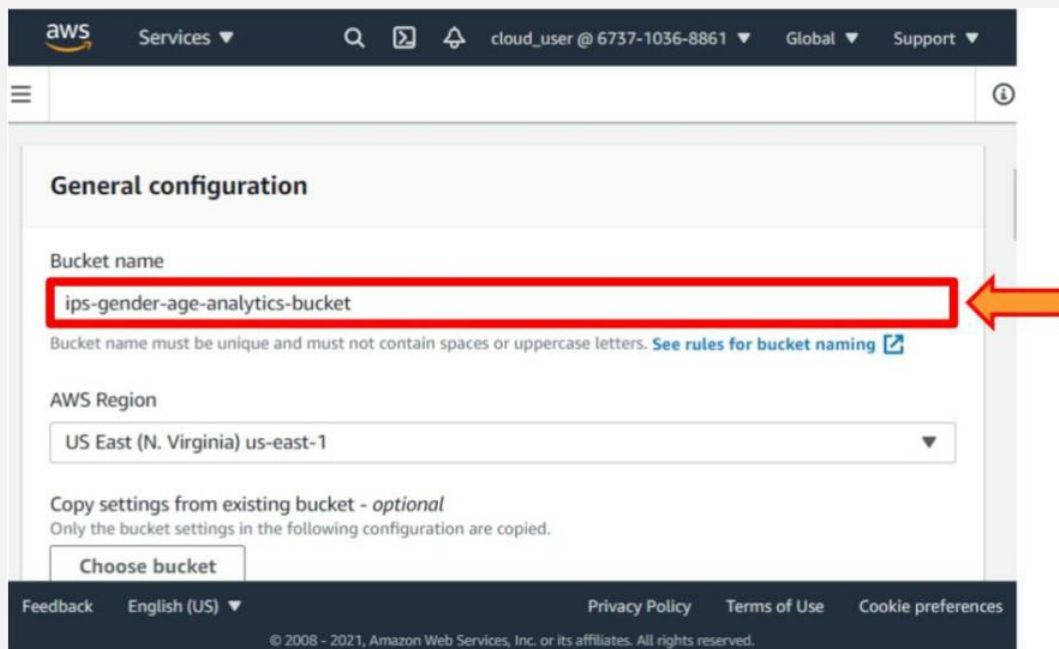
2. Select the **S3** from the **Storage**.



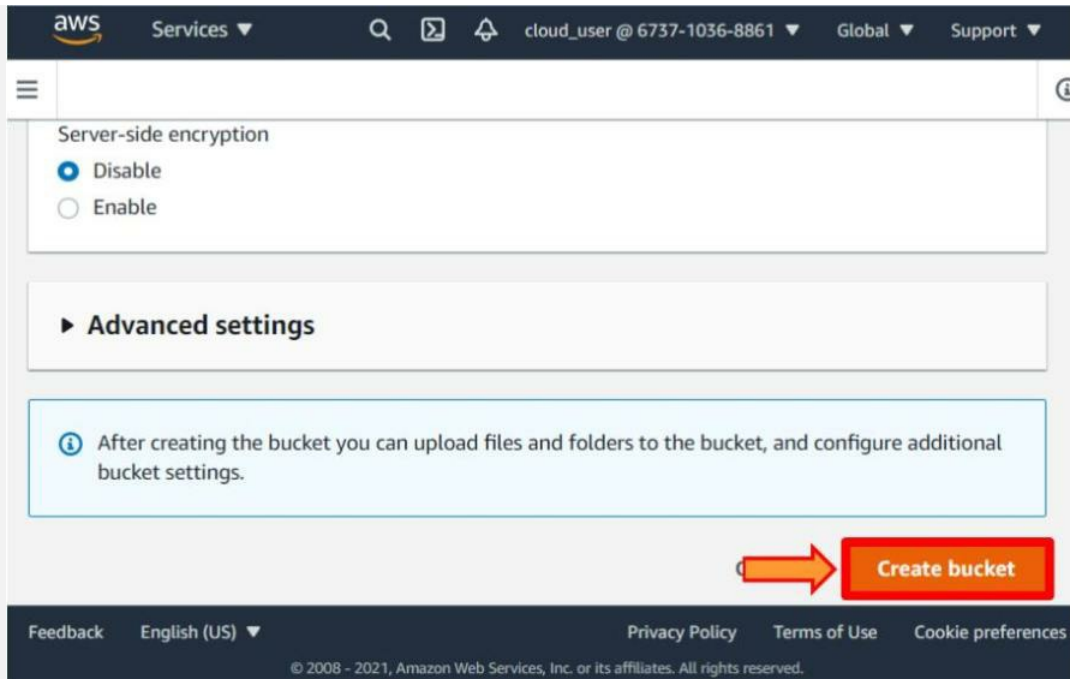
3. Click on the **Create bucket** button.



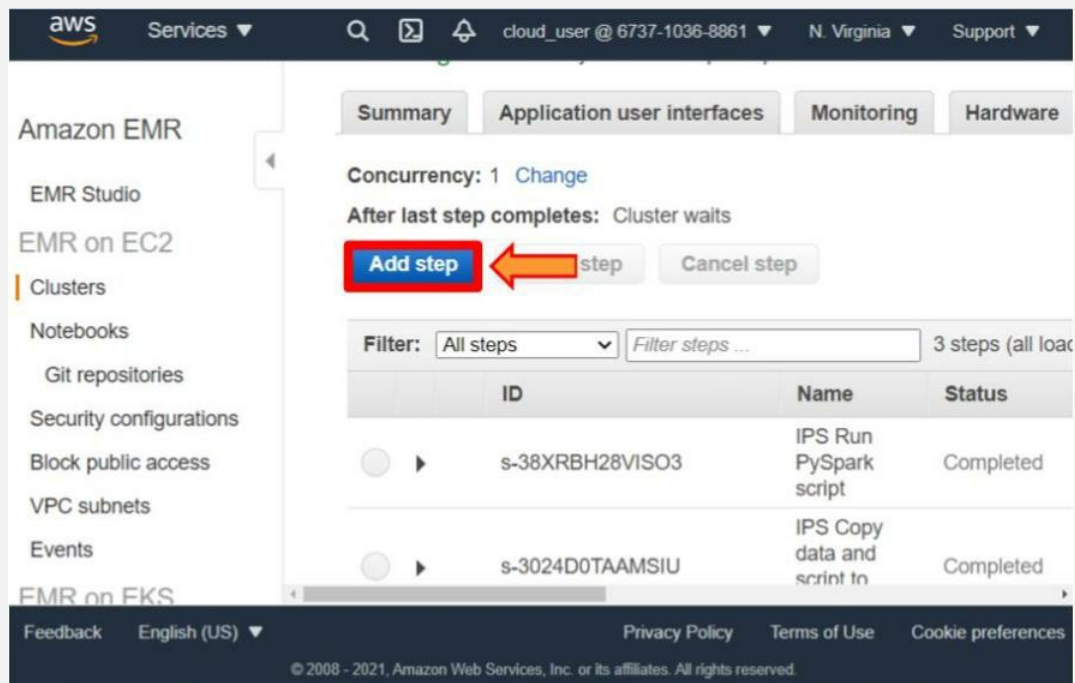
4. Give a name **ips-gender-age-analytics-bucket**.



5. Click on the **Create Bucket** button.



6. Click on the **Add Step** button.



7. Select the **Custom JAR** step type.

8. Give the name **IPS Load results to S3**.

9. In the JAR location, copy and paste the following command **command-runner.jar**.

**Add step** [X]

Step type: Custom JAR

Name\*: IPS Load results to S3

JAR location\*: command-runner.jar

JAR location maybe a path into S3 or a fully qualified java class in the classpath.

Arguments

These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.

Action on failure: Continue

What happens if the step fails

Cancel Add

10. In the argument field copy and paste the following command
- ```
s3-dist-cp --src=hdfs:///results --dest=s3://<YOUR_BUCKET_NAME>/
```

**Add step** [X]

Step type: Custom JAR

Name\*: IPS Load results to S3

JAR location\*: command-runner.jar

JAR location maybe a path into S3 or a fully qualified java class in the classpath.

Arguments

```
s3-dist-cp --src=hdfs:///results --dest=s3://ips-gender-age-analytics-bucket/
```

These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.

Action on failure: Continue

What happens if the step fails

Cancel Add

11. Select the **Continue** in action on failure.

**Add step** [X]

**Step type** Custom JAR

**Name\*** IPS Load results to S3

**JAR location\*** command-runner.jar  
JAR location maybe a path into S3 or a fully qualified java class in the classpath.

**Arguments**  
s3-dist-cp --src=hdfs:///results --dest=s3://ips-gender-age-analytics-bucket/

These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.

**Action on failure** Continue  
What happens if the step fails

Cancel Add

12. Click on the **Add** button.

**Add step** [X]

**Step type** Custom JAR

**Name\*** IPS Load results to S3

**JAR location\*** command-runner.jar  
JAR location maybe a path into S3 or a fully qualified java class in the classpath.

**Arguments**  
s3-dist-cp --src=hdfs:///results --dest=s3://ips-gender-age-analytics-bucket/

These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.

**Action on failure** Continue  
What happens if the step fails

Add

13. The step takes some time to copy the data from HDFS into the S3 bucket.

aws Services cloud\_user @ 6737-1036-8861 N. Virginia Support

Amazon EMR

EMR Studio

EMR on EC2

Clusters

Notebooks

Git repositories

Security configurations

Block public access

VPC subnets

Events

EMR on FKS

Summary Application user interfaces Monitoring Hardware

Concurrency: 1 [Change](#)

After last step completes: Cluster waits

[Add step](#) [Clone step](#) [Cancel step](#)

Filter: All steps Filter steps ... 4 steps (all lo

| ID              | Name                   | Status    |
|-----------------|------------------------|-----------|
| s-203ENFA1SW1XF | IPS Load results to S3 | Pending   |
| s-38XRBH28VISO3 | IPS Run PySpark script | Completed |

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

14. Hence, the step is completed.

aws Services cloud\_user @ 6737-1036-8861 N. Virginia Support

Amazon EMR

EMR Studio

EMR on EC2

Clusters

Notebooks

Git repositories

Security configurations

Block public access

VPC subnets

Events

EMR on FKS

Cluster: ips-age-gender-analytics-cluster

Waiting Cluster ready after last step completed.

Summary Application user interfaces Monitoring Hardware

Concurrency: 1 [Change](#)

After last step completes: Cluster waits

[Add step](#) [Clone step](#) [Cancel step](#)

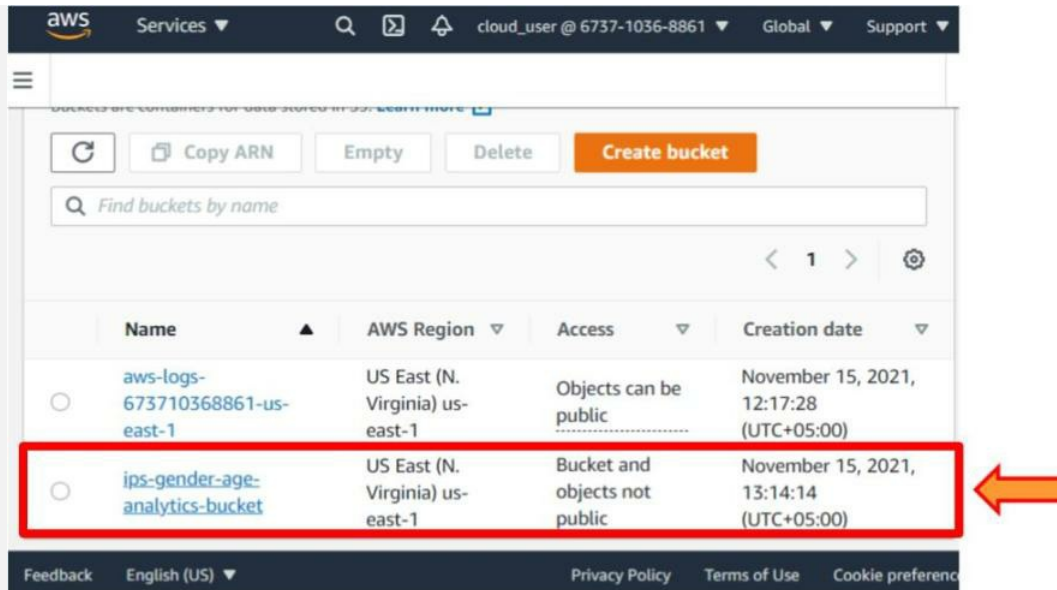
Filter: All steps Filter steps ... 4 steps (all lo

| ID              | Name                   | Status    |
|-----------------|------------------------|-----------|
| s-203ENFA1SW1XF | IPS Load results to S3 | Completed |
|                 | IPS Run                |           |

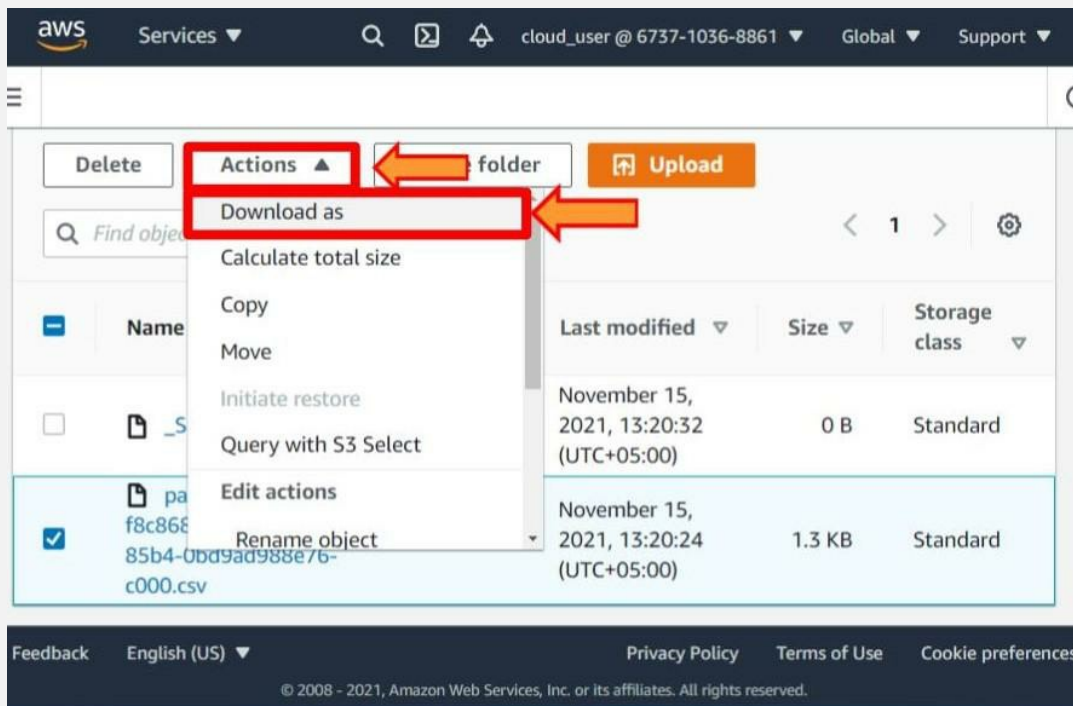
Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

15. Go to the S3 dashboard. Click on the **ips-gender-age-analytics-bucket**.



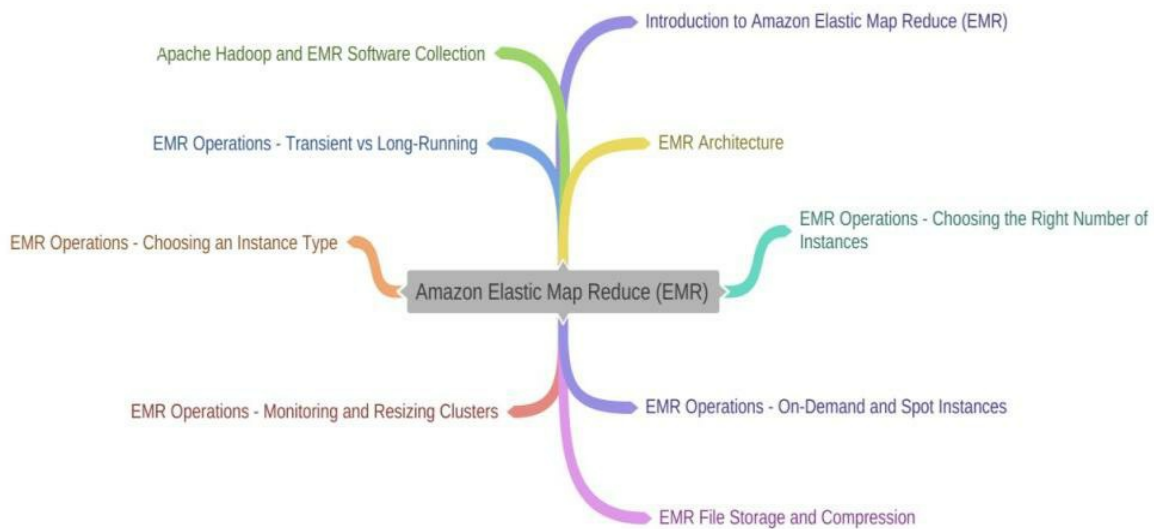
16. Select the file. Click on the **Actions**. Then click on the **Download as**.



17. Open the downloaded file to verify that it shows all the data, aggregated by age and gender.

```
File Edit Selection View Go ... part-00000-f8c868e0-53c5-44f8-85b4-0bd9ad988e76-...
part-00000-f8c868e0-53c5-44f8-85b4-0bd9ad988e76-c000.csv
C: > Users > Muhammad Usman Khan > Downloads > part-00000-f8c868e0-53c5-44f8-85b4-0bd9ad988e76-c000.csv
1 job,age,gender,count
2 74,female,60
3 41,female,60
4 31,female,58
5 67,female,58
6 64,male,58
7 62,male,57
8 60,female,57
9 48,female,57
10 50,female,57
11 30,male,55
12 38,female,55
13 30,female,55
14 71,female,55
15 61,male,54
16 28,male,54
17 72,female,53
18 51,female,52
19 34,male,53
Ln 1, Col 1 Spaces: 4 UTF-8 LF Plain Text
```

## Mind Map



*Figure 6-47: Mind Map*

## Practice Questions

1. Map reduce is a technique that data scientists can use to distribute workloads across many computing nodes.

A. True

B. False

2. ----- is open-source software that allows you to operate a distributed file system over several computers to tackle challenges requiring large amounts of data.

A. EMR

B. Hadoop Distributed File System

C. EMRFS

3. ----- is a fully managed AWS service that allows you to spin up Hadoop ecosystems.

A. Hadoop Distributed File System

B. CloudWatch

C. Elastic Map Reduce

4. ----- node tracks and directs the HDFS.

A. Task Nodes

B. Primary Nodes

C. Core Nodes

5. The ----- is responsible for the YARN resource management.

A. Primary Node

B. Core Node

• Task Node

6. ----- are managed by the primary node, and run tasks such as Hadoop Map reduce tasks, Hive Scripts, and Spark executors.

- Primary Node
- Core Node
- Task Node

7. ----- are optional and can be used to add power to perform parallel computation tasks on data like Map reduce tasks and Spark executor.

- Primary Node
- Core Node
- Task Node

8. ----- can be added and removed from the core nodes to ramp up extra CPU or memory for compute-intensive tasks.

A. Task Nodes

B. Primary Nodes

C. Core Nodes

9. The EMR clusters only reside in a single availability zone.

A. True

B. False

10. ----- is used for very high I/O performance and high IOPS at low cost. It is best used for temporary data (caches, buffers, and scratch data).

A. Local File System (Instance Storage)

B. Local File System (EBS Volume)

C. Hadoop Distributed File System (HDFS)

D. Elastic Map Reduce File System (EMRFS)

11. ----- is used to add more storage for HDFS.

A. Local File System (Instance Storage)

B. Local File System (EBS Volume)

C. Hadoop Distributed File System (HDFS)

D. Elastic Map Reduce File System (EMRFS)

12. ----- is best used for caching the results produced by intermediate job-flow steps.

A. Local File System (Instance Storage)

B. Local File System (EBS Volume)

C. Hadoop Distributed File System (HDFS)

D. Elastic Map Reduce File System (EMRFS)

13. ----- is best used for persistent store and S3 features that are needed, like server-side encryption and consistency.

A. Local File System (Instance Storage)

B. Local File System (EBS Volume)

C. Hadoop Distributed File System (HDFS)

D. Elastic Map Reduce File System (EMRFS)

14. If you set your cluster to terminate automatically, it will do so after completing all the steps. It is known as -----.

A. Transient Cluster

B. Long-Running Cluster

15. If you set up the cluster to continue operating after processing is completed, the type of cluster is known as -----.

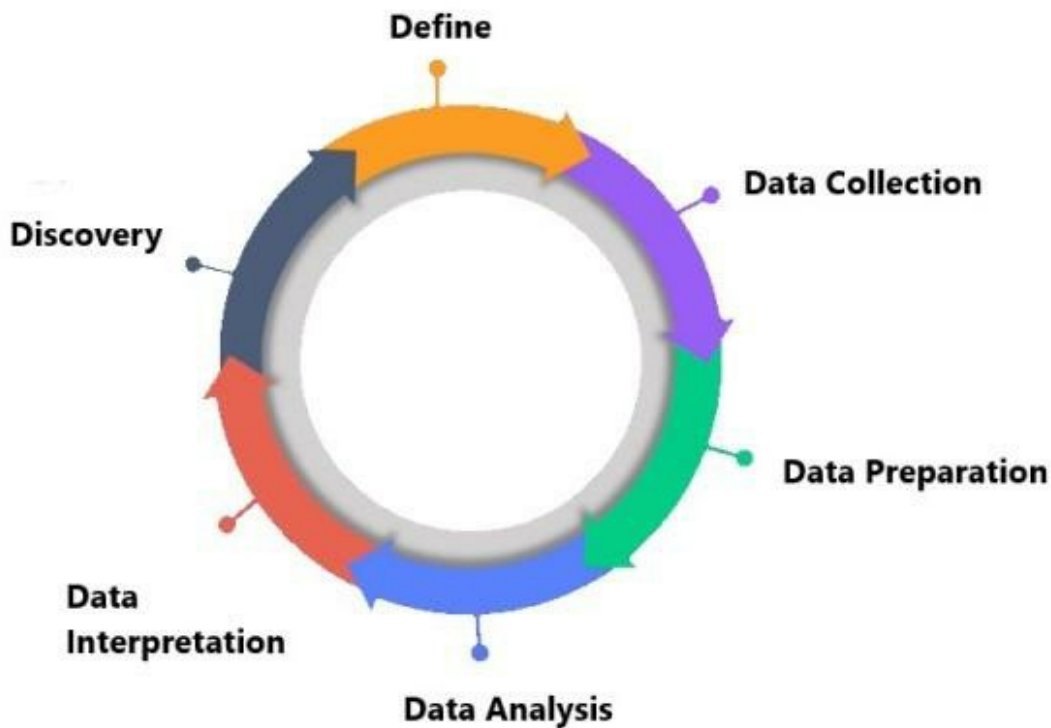
A. Transient Cluster

B. Long-Running Cluster

# CHAPTER 07: USING REDSHIFT

## Introduction

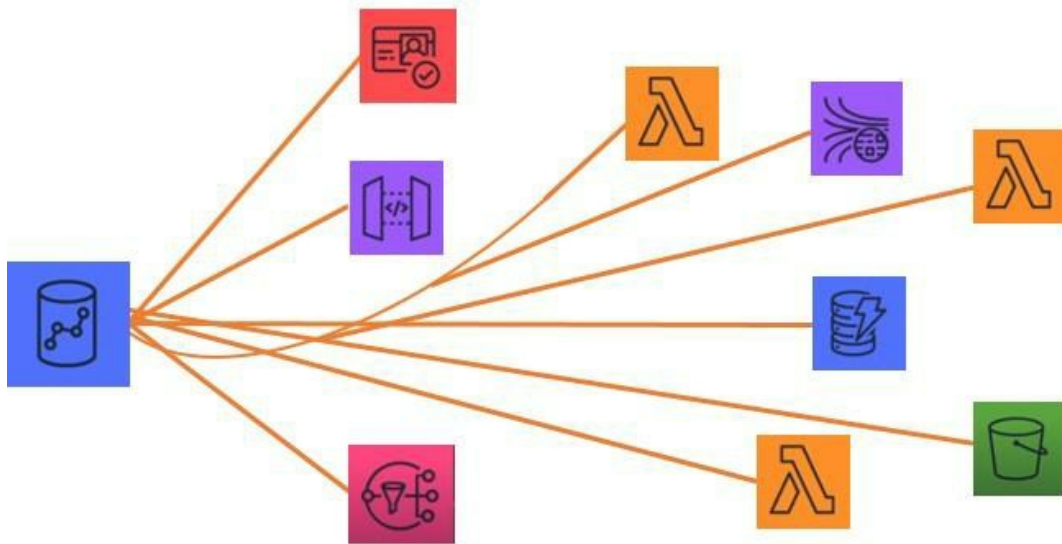
Redshift is a data warehousing service. It can warehouse the data at the petabyte scale, which means Redshift can store large data. It can also index and query data so that it remains usable. We can store petabytes and hexabytes of data in S3.



*Figure 7-01: Steps to Success*

Consider application architecture; all the services in architecture will emit data. In this example, we are collecting a huge amount of data, depending on how heavily our application is. We can manage all of the data artifacts that come from Redshift's services and perform analysis on our application. We can perform analysis on the operation, the

performance, the general usage, everything about our application we can collect in Redshift, and then would be able to write queries to perform analysis of that data.



*Figure 7-02: Data Warehouse*

**EXAM TIP:** Redshift is a data warehousing service. It can warehouse the data at the petabyte scale and index and query that data.

## Redshift Architecture

### Cluster

Within the cluster, we have either leader nodes or worker nodes. The leader node manages the schema. It contains the data warehouse

metadata and performs all the query planning and script generation. The worker nodes perform query execution and slice management before storing all the data within the Slices.

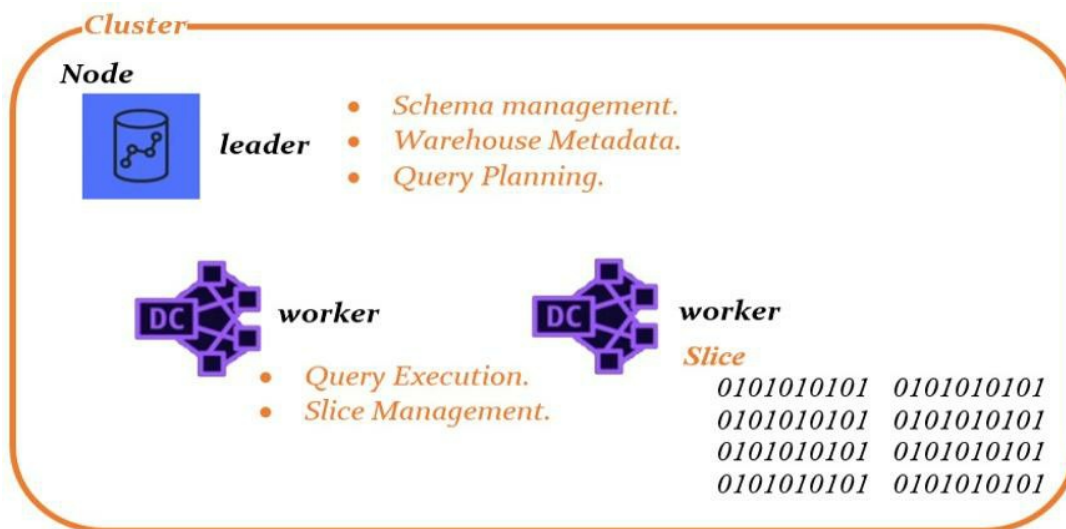


Figure 7-03: Cluster

## Node

These are EC2 instances; there are three types to choose from for the most part. Node is the individual compute resources with storage attached for the Redshift cluster. The storage attached to these nodes is faster. These are generally used in cases where we need to perform queries quickly. We want near real-time analytics.

We also have dense storage node types. These have slower storage. The full capacity in these tables is the total capacity for a cluster. If we have 128 DS2 X large nodes in our group, we will have two petabytes of storage. It is how we get the petabyte-scale data warehouse with Redshift, anything larger than 2 petabytes. We need to stand up a second cluster and manage that in our application layer, but that is unlikely. Two petabytes are a huge amount of data. We also have the

new RA3 nodes, and these have the same amount of storage associated with both types. The warehouse is backed by S3 and uses a local storage cache to provide hot data for our queries. This storage auto-scales with your usage; we get a total capacity of 4 to 8,000 tebibytes for these node types, with the maximum nodes per cluster, for each node type.

#### Dense Compute



| Node size   | vCPU | RAM (GiB) | Default Sizes | Storage          | Node Range | Total capacity |
|-------------|------|-----------|---------------|------------------|------------|----------------|
| dc2.large   | 2    | 15        | 2             | 160 GB NVMe-SSD  | 1-32       | 5.12 TB        |
| dc2.8xlarge | 32   | 244       | 16            | 2.56 TB NVMe-SSD | 2-128      | 326 TB         |

#### Dense Storage



| Node size   | vCPU | RAM (GiB) | Default Sizes | Storage   | Node Range | Total capacity |
|-------------|------|-----------|---------------|-----------|------------|----------------|
| ds2.large   | 4    | 13        | 2             | 2 TB HDD  | 1-32       | 16 TB          |
| ds2.8xlarge | 36   | 244       | 16            | 16 TB HDD | 2-128      | 2 PB           |

#### RA3

| Node size    | vCPU | RAM (GiB) | Default Sizes | Storage | Node Range | Total capacity |
|--------------|------|-----------|---------------|---------|------------|----------------|
| ra3.4large   | 12   | 96        | 4             | 64 TB   | 2-32       | 4096 TB        |
| Ra3.16xlarge | 48   | 384       | 16            | 64 TB   | 2-128      | 8192 TB        |

*Figure 7-04: SageMaker Hosting Services*

**EXAM TIP:** We get faster computing and more storage than dense compute. We can experience a little latency with our queries if the storage cache on the nodes themselves does not have the data. It has to be retrieved from S3, but generally, they perform faster than thick storage while having more storage than dense computers.

## Slice

A slice is a group of configurable entities kept in a reusable asset as a single unit. Slices are useful for grouping entities and other slices for reuse. Prefabs and slices are similar, but slices are part of the new component entity structure. Prefabs cannot contain component entities, although slices can.



*Figure 7-05: Slice*

**EXAM TIP:** Each Slice has its separate piece of computing, memory, and storage.

## Redshift Query Process

A cluster has a leader node and several worker nodes. If there is a single node cluster, the leader and worker will be the same node. It will separate the leader into a Slice, and we will have our user that sends a query to the leader node. The leader node generates a query plan, which lets it create execution scripts to complete that query because it knows where all of the data is stored in the cluster. Those execution scripts will go to worker nodes. The Slices themselves do not store any table data. They have data, and they know where that data is. The worker nodes will go to their Slices, and then they will each have a piece of data that they need to return, which will come back, and the leader will combine that data into our query response, and that query response will go back to our end user. That is the Redshift query

process at a very high level.

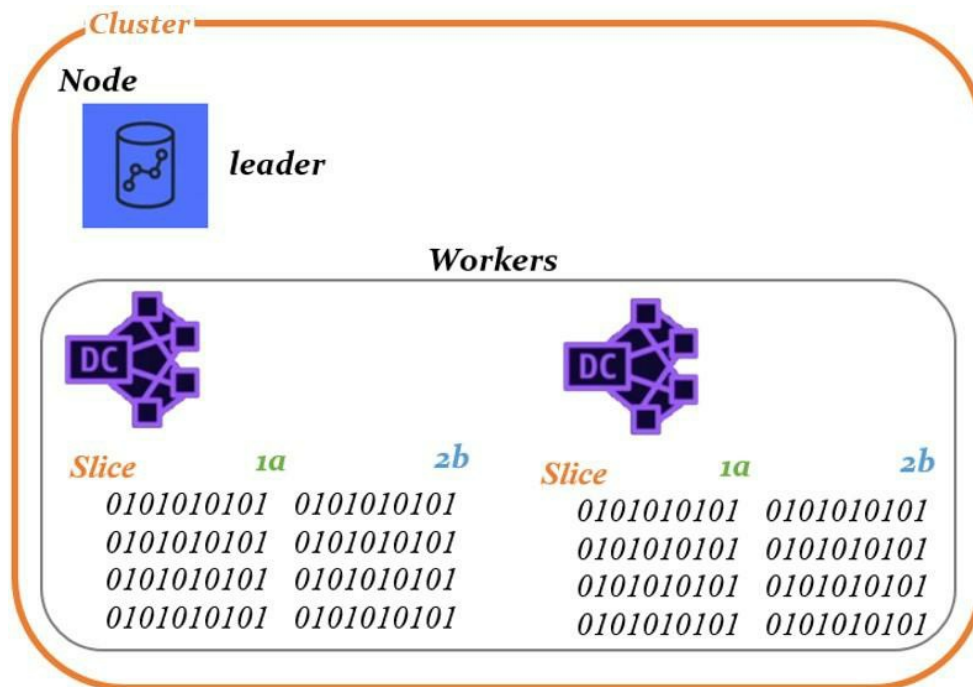


Figure 7-06: Redshift Query Process

#### EXAM TIP:

- Cluster – Organizational container for resources
- Node – Similar to an instance in RDS
- Slice – Logical subdivision of node resources
- Query Process – Query plan, Query, Execution Scripts

## Redshift in the AWS Service Ecosystem

With Redshift, we are in the active utilization area. We have end users on the left and a businessperson on the right that needs to perform some analytics on the application. End-users collect authentication data, and they will then log in to our service. An email goes back to the

end-user and logs something in a database. There might be some data that we want to make sure makes it through into our storage or otherwise into our application, so we will put it into a data stream, which will be processed out by another compute resource that computes resource may log something into Redshift. It may also store something in our database. It will generate a return that goes back to the end-user.

After a while, the businessperson wants to perform some analysis to hook Redshift up to a visualization tool. He will generate graphs of our data with that visualization tool.

As the application grows, we may want to track any emails are sent out. We will trigger a compute resource off any usage of our email service start the same event service from our database engine so that we can archive some more data in Redshift and our storage bucket; Redshift may be storing a processed version of the record where the bucket stores the raw form itself. We can use something in RedShift to query data directly out of buckets. This is very useful as the application grows and we get more data; we may need to store some of that data outside of RedShift to keep it in our storage bucket or infrequent queries.

The businessperson can also run queries directly from RedShift. He may want to generate some comprehensive data reports or other data to share with third parties, so he has two ways to get data from our analytics pipeline. We have DynamoDB Streams, which will let us trigger Lambda functions. The service that enables us to query S3 data from Redshift is called the Redshift spectrum.

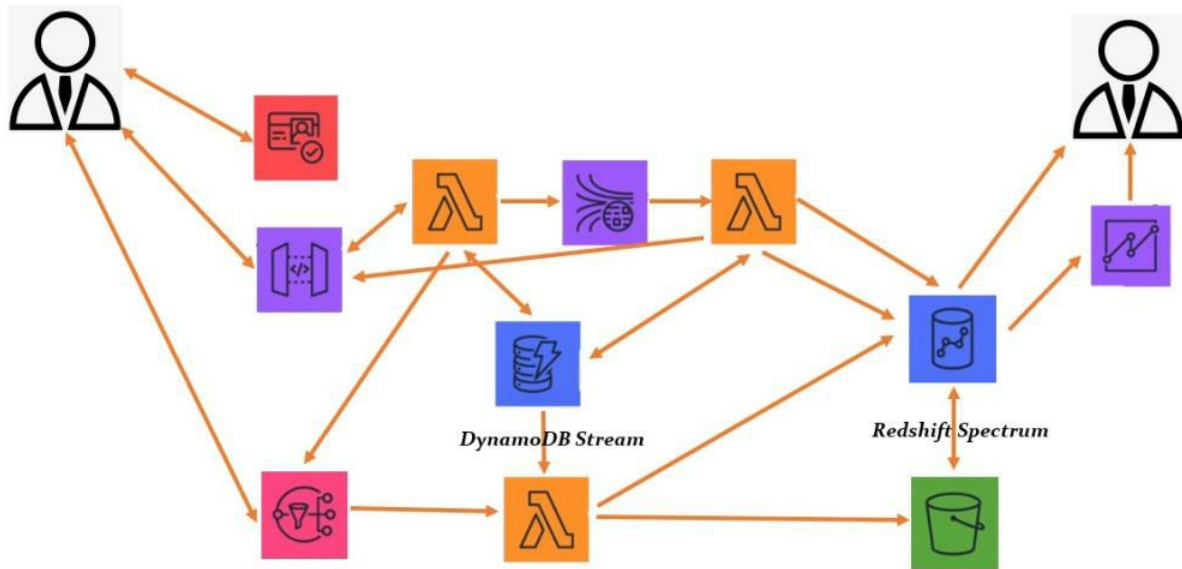


Figure 7-07: Application Service

Cognito performs user authentication and management. API Gateway provides us API services. Lambda is a compute resource. Simple notification service lets us send emails or text messages to end-users or the application administrators. Kinesis is a data streaming service. DynamoDB is used as a transactional database for our application. QuickSight is a visualization tool. The simple storage service provides archiving and raw data storage.

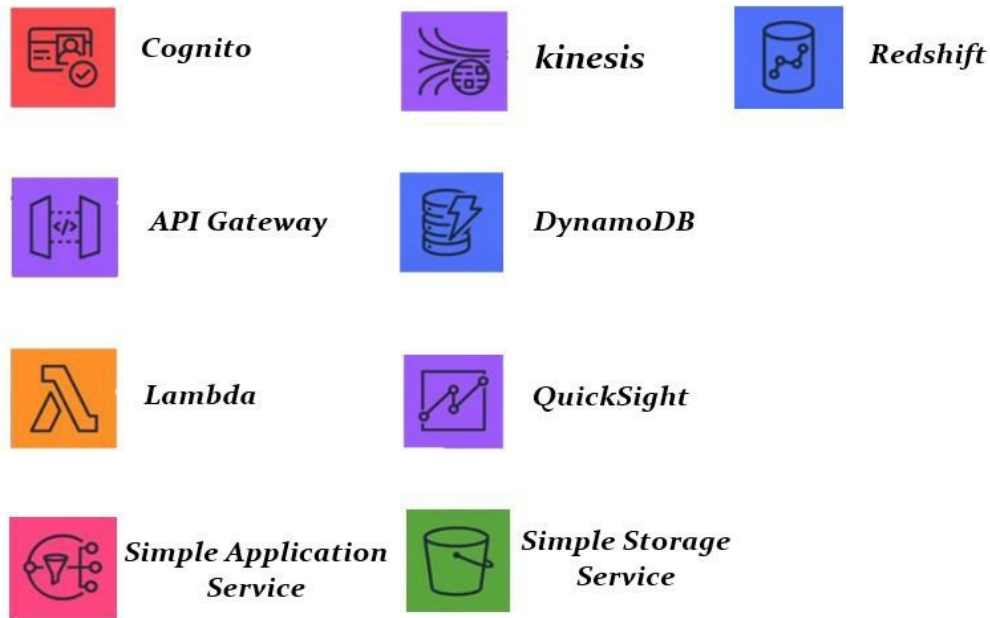


Figure 7-08: Application Services

#### EXAM TIP:

- Basic Analytics – Gathering points for application emitted data
- More data – Utilizing streams, triggers, and Lambda allows easy expansion
- Application Service – Services in our scenarios, Redshift aggregates data from them

## Redshift Use Cases

### Data Warehouse VS Data Lake

The dissimilarity between a data warehouse and a data lake is more structured. The data in it will be cataloged, and the access speed for retrieving data from a data warehouse is much faster than it would be from a data lake. It is primarily because of the lack of structure and cataloging that can be seen in a data lake.

## What makes Redshift Different?

It is, in most cases, much faster if optimizing to the same level. It is much more scalable, so we can have a small warehouse and easily scale it up to a gigantic warehouse. A few AWS service integrations make some of the paths that we would perform with the data warehouse considerably easier than they would be if we had to integrate with other systems in an ad hoc manner.

## Redshift Table Design

Columnar databases let us access columns of our data more efficiently. The column table is arranged to set a columnar data file. To get the node size from our row database, we access every row in our table, whereas we need to access a single row with the columnar database. It has significant implications for OLAP or OLAP transactions where we need to make comparisons or calculations from large amounts of single-column data.

### Row

```
Node_Size, vCPU, RAM, Default_Slices, Storage, Node_Range, Total Capacity
dc2.large, 2, 15, 2, 160 GB NVME, 1-32, 5.12 TB
dc2.8xlarge, 32, 244, 16, 2.56 TB NVME, 2-128, 326 TB
ds2.xlarge, 4, 13, 2, 2 TB HDD, 1-32, 64 TB
ds2.8xlarge, 32, 244, 16, 16 TB HDD, 2-128, 2 PB
ra3.4xlarge, 12, 96, 4, 64 TB, 2-32, 4096 TB
ra3.16xlarge, 48, 384, 16, 64 TB, 2-128, 8192 TB
```

### Column

```
Node_Size, dc2.large, dc2.8xlarge, ds2.xlarge, ds2.8xlarge, ra3.4xlarge, ra3.16xlarge
vCPU, 2, 32, 4, 32, 12, 48
RAM, 15, 244, 13, 244, 96, 384
Default_Slices, 2, 16, 2, 16, 4, 16
Storage, 160 GB NVME, 2.56 TB NVME, 2 TB HDD, 16 TB HDD, 64 TB, 64 TB
Node_Range, 1-32, 2-128, 1-32, 2-128, 2-32, 2-128
Total Capacity, 5.12TB, 326 TB, 64 TB, 2 PB, 4096 TB, 8192 TB
```

Figure 7-09: Columnar Database

## Data Types

There are several numeric data types, and they have aliases. You can maintain compatibility between Postgres and Redshift, for instance, because Redshift's interface is Postgres compatible. Redshift only has a reduced set, but it contains aliases to map them appropriately. There are few signed integer data types and few floating-point data types. Boolean is standard. It is only for true or false. Texts have character and variable characters. The character is fixed length, whereas varchar or variable character is a variable length. In time data types, the date is the calendar date, which is the year, month, day, and the timestamp is the date and time. The timestamp TZ includes the time zone. Time and time TZ is the daytime or zone daytime had; this gives us our set of data types. These will map incompatibility to the Postgres data types because Redshift is the primary compatible interface. Aliases are used to map to the actual Redshift data types to maintain compatibility.

### Numeric

| <i>Data Type</i> | <i>Alias</i>  |
|------------------|---------------|
| SMALLINT         | INT2          |
| INTEGER          | INT, INT4     |
| BIGINT           | INT8          |
| DECIMAL          | NUMERIC       |
| REAL             | FLOAT4        |
| DOUBLE PRECISION | FLOAT8, FLOAT |

### Boolean

| <i>Data Type</i> | <i>Alias</i> |
|------------------|--------------|
| BOOLEAN          | BOOL         |

### Text

| <i>Data Type</i> | <i>Alias</i>                      |
|------------------|-----------------------------------|
| CHAR             | CHARACTER, NCHAR, BPCHAR          |
| VARCHAR          | CHARACTER VARYING, NVARCHAR, TEXT |

### Time

| <i>Data Type</i> | <i>Alias</i>                |
|------------------|-----------------------------|
| DATE             |                             |
| TIMESTAMP        | TIMESTAMP WITHOUT TIME ZONE |
| TIMESTAMP TZ     | TIMESTAMP WITH TIME ZONE    |
| TIME             | TIME WITHOUT TIME ZONE      |
| TIME TZ          | TIME WITH TIME ZONE         |

Figure 7-10: Data Types

## Compression

In Redshift, Postgres can compress individual columns, which means different compression types are available depending on the data type. Most of our number and time data types will default to AZ64 compression, and our character and variable character data types will default to LZO compression and several other impression encodings available. The other data types are default to raw, Boolean, real, and double. If something is our sort key, it will need to be raw; it is uncompressed because the database engine frequently performs queries.

| Encoding        | Keyword                                   | Default                                                          |
|-----------------|-------------------------------------------|------------------------------------------------------------------|
| Raw             | RAW                                       | Sort keys, Boolean, Real, Double                                 |
| AZ64            | AZ64                                      | SMALLINT, INTEGER, BIGINT, DECIMAL, date, TIMESTAMP, TIMESTAMPTZ |
| Byte dictionary | BYTEDICT                                  |                                                                  |
| Delta           | DELTA, DELTA <sub>32K</sub>               |                                                                  |
| LZO             | LZO                                       | CHAR, VARCHAR                                                    |
| Mostlyn         | MOSTLY8, MOSTLY16, MOSTLY32               |                                                                  |
| Run-length      | RUNLENGTH                                 |                                                                  |
| Text            | TEXT <sub>255</sub> , TEXT <sub>32K</sub> |                                                                  |
| zstandard       | ZSTDD                                     |                                                                  |

Table 7-01: Compression

**EXAM TIP:** The inability to create a sort key because the column compression type is set to something different from basic; you would need to modify that table to remove the compression from that column to use it as a sort key.

## Sort Keys

Consider a data set; it can be represented as Y 1, 2, 3, and then A, B, C. There are two options for sorting our data. For example, the Chroma and number values for sort keys. The Redshift engine will split the data set into three blocks if using a compound sort key. Within each one of those blocks, the secondary sorting is by the numbers. An interleaved sort key breaks the data set into nine blocks.

|               |            |              |                                     |  |  |
|---------------|------------|--------------|-------------------------------------|--|--|
| <b>Data</b>   |            |              | <b>Dataset</b>                      |  |  |
| <b>Chroma</b> | <b>Num</b> | <b>Alpha</b> | Y1A Y1B Y1C Y2A Y2B Y2C Y3A Y3B Y3C |  |  |
| Yellow        | 1          | A            | P1A P1B P1C P2A P2B P2C P3A P3B P3C |  |  |
| Pink          | 2          | B            | C1A C1B C1C C2A C2B C2C C3A C3B C3C |  |  |
| Cyan          | 3          | C            |                                     |  |  |

|                                     |             |             |                                  |  |  |
|-------------------------------------|-------------|-------------|----------------------------------|--|--|
| <b>Compound - Chroma, Num</b>       |             |             | <b>Interleaved - Chroma, Num</b> |  |  |
| Y1A Y1B Y1C Y2A Y2B Y2C Y3A Y3B Y3C | Y1A Y1B Y1C | Y2A Y2B Y2C | Y3A Y3B Y3C                      |  |  |
| P1A P1B P1C P2A P2B P2C P3A P3B P3C | P1A P1B P1C | P2A P2B P2C | P3A P3B P3C                      |  |  |
| C1A C1B C1C C2A C2B C2C C3A C3B C3C | C1A C1B C1C | C2A C2B C2C | C3A C3B C3C                      |  |  |

Figure 7-11: Sort Keys

The compound is the default sort key type. If we do not specify complex or interleave, it will create compact compound keys that benefit operations. Compound sort keys work well for hierarchical data or multi-column sorting. Order keys when creating a compound key will increase our table maintenance. Interleave will improve table

maintenance. It is important to vacuum and analyze our tables. Interleave keys makes it easier to pull out those individual blocks that increase table maintenance. Analyzing interleave keys work much better when using a single column for a key.

## **Distribution Styles**

1. **Even** – Blocks are distributed evenly between cluster slices (default).
2. **Key** – identical key values are stored on the same Slice.
3. **All** – All slices store a copy of table data.

## **Constraints**

1. **Primary, Foreign Key:**

Used by the query planner as a hint about relations.

2. **Unique:**

Not enforced used by query planner as a hint.

3. **Not Null:**

Not enforced or respected.

## **Redshift Spectrum**

### **How do you query flow?**

Redshift Spectrum is an interface to create foreign tables in our Redshift cluster from stored data in S3. It uses the external keyword when creating schemas and tables in our collection. It can read and write to Spectrum tables. It does not support update and delete operations.

Access control can either IAM or use AWS Lake Formation in query flow, giving more granular control over tables. It is an external service

that sends data requests through access control and then goes to the data store. The data store comprises a Glue data catalog and Athena SQL interface connected to the underlying data in S3. Once that query has been answered in Athena, it will return the data to Spectrum, which then passes it back to the Redshift cluster and is combined with any other data involved in our query to provide a query response cluster.

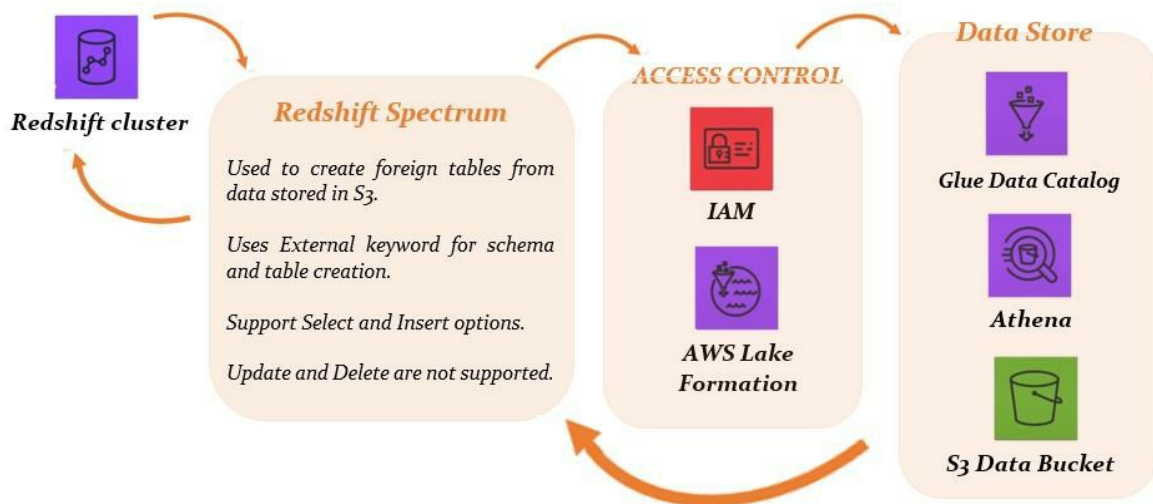
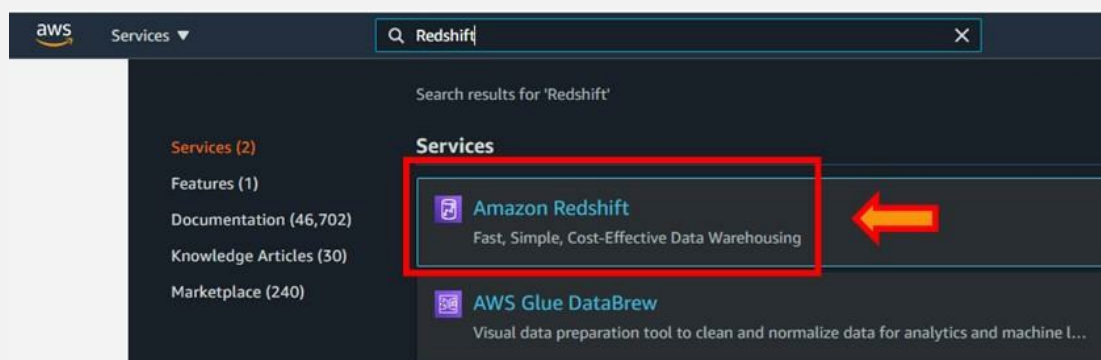


Figure 7-12: How Query Flows

## Demo 7-01: Redshift

### Step 1: Creating External Tables and Schemas

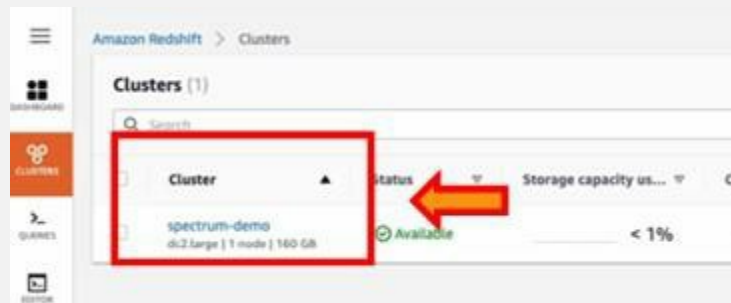
1. Log in to the **AWS management console**.
2. Search **Redshift**.



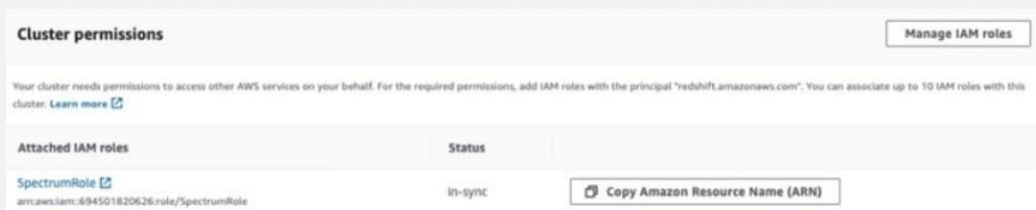
3. Select the **cluster**.



4. Click on **Cluster**.



5. Spectrum role is created.



6. Click on **Query Editor**.

7. Click on **Connect to Database**.



8. Define database name and user.

9. Click on **Connect to Database**.

### Connect to database

Connection  
Create a new database connection or select a recent connection.

Create new connection ▼

Cluster  
🕒 spectrum-demo ▼

Database name  
spectrum-demo

Database user  
The master user name for your database instance.  
john

Database password  
The master user password for your database instance.  
Strongpass1

☒ Show password

Connect with temporary password → **Connect to database**

10. Select **Public** schema.

### Data objects

Select schema  
Select a schema to view data tables

information\_schema ▲

🔍 |

information\_schema

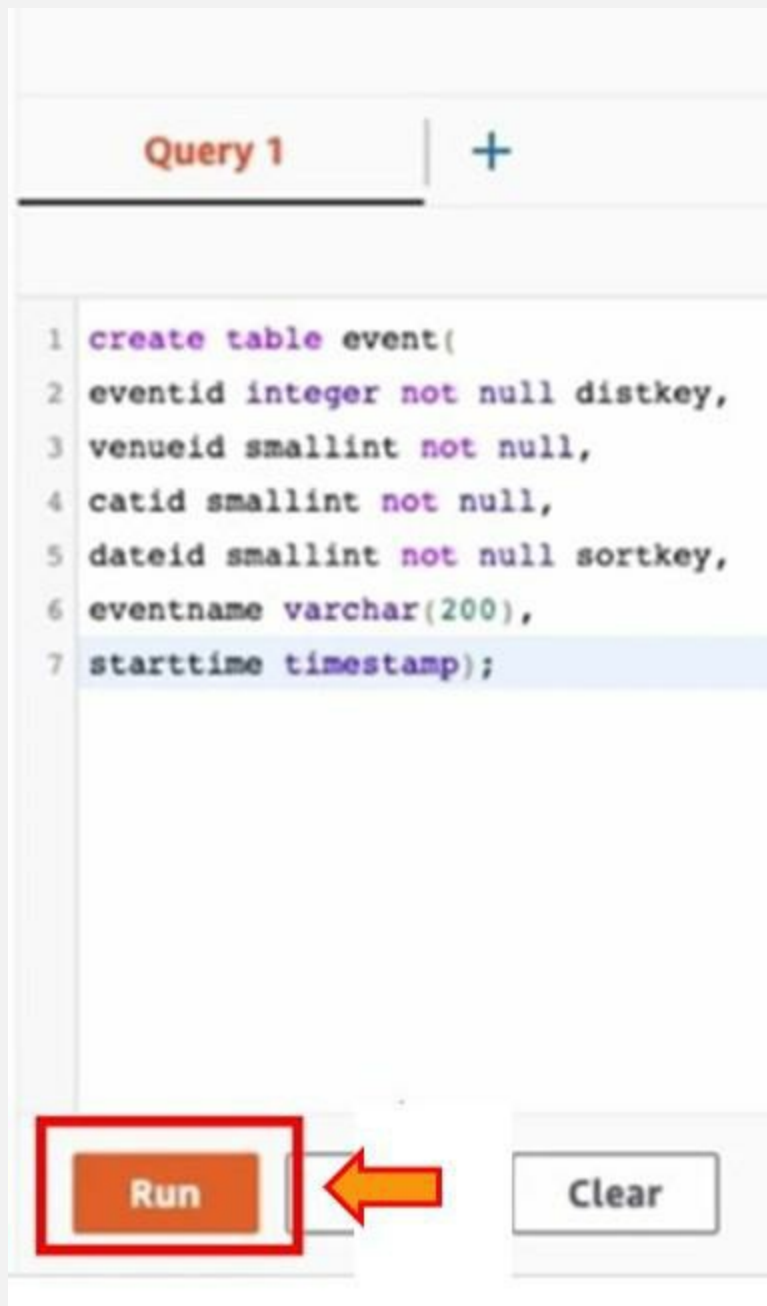
pg\_catalog

pg\_internal

**public** ←

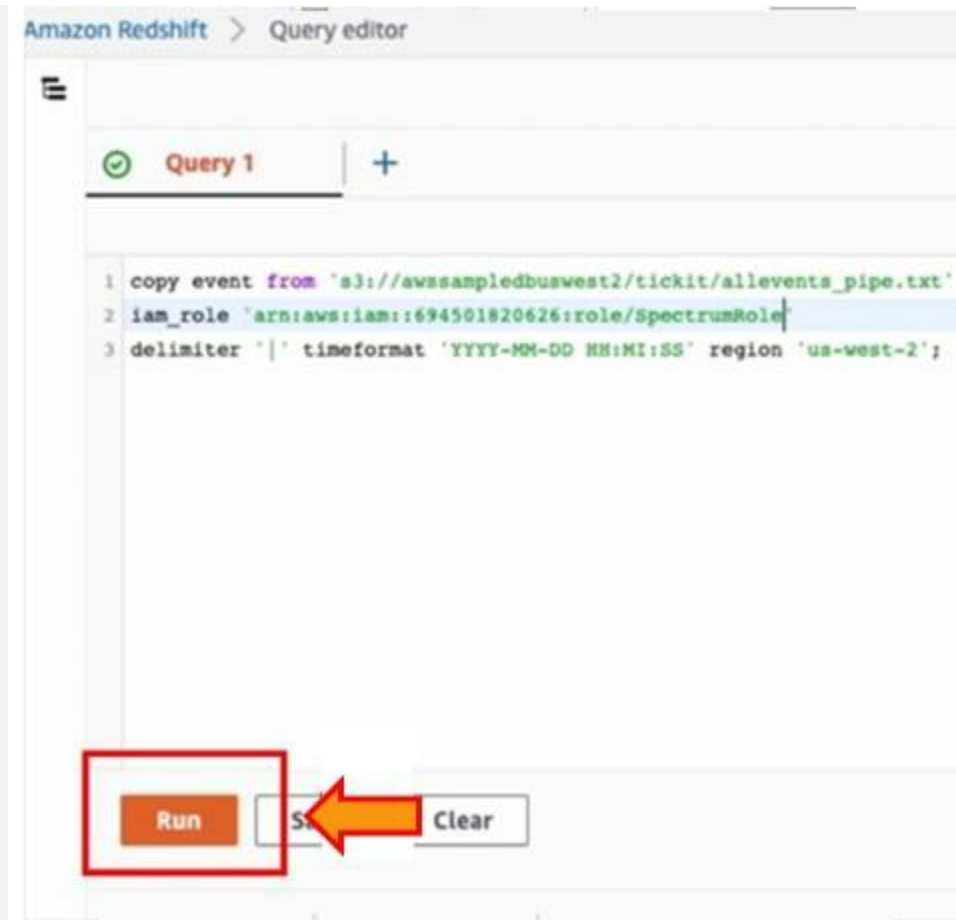
check\_constraints ...

11. Edit the Query. Click on **Run**.



12. Edit the Query.

13. Click on **Run**.



14. Edit the Query.
15. Click on **Run**.



Query 1



```
1 create external schema spectrum
2 from data catalog
3 database 'spectrumbd'
4 iam_role 'arn:aws:iam::694501820626:role/SpectrumRole'
5 create external database if not exists;
```

Run



Clear

16. Edit the Query and click on **Run**.

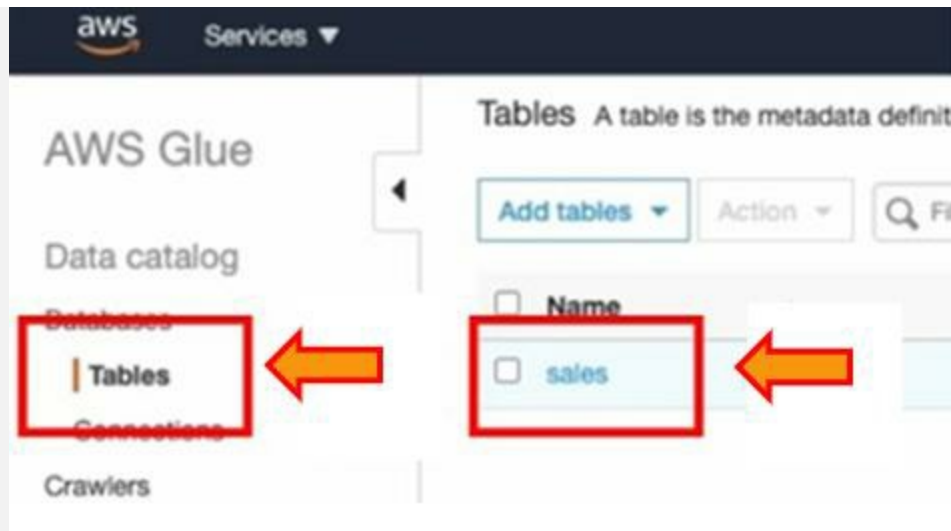


Query 1



```
1 create external table spectrum.sales(  
2   salesid integer,  
3   listid integer,  
4   sellerid integer,  
5   buyerid integer,  
6   eventid integer,  
7   dateid smallint,  
8   qtysold smallint,  
9   pricepaid decimal(8,2),  
10  commission decimal(8,2),  
11  saletime timestamp)  
12  row format delimited  
13  fields terminated by '\t'  
14  stored as textfile  
15  location 's3://awssampleduswest2/tickit/spectrum/sales/'  
16  table properties ('numRows'='172000');
```

17. Click on **Tables** under the dashboard.
18. Select the **table**.



19. See the column name and the data type for each one. It is how Spectrum works.

|                         |                                                            |      |                      |        |                                  |
|-------------------------|------------------------------------------------------------|------|----------------------|--------|----------------------------------|
| Name                    | sales                                                      |      |                      |        |                                  |
| Description             |                                                            |      |                      |        |                                  |
| Database                | spectrumdb                                                 |      |                      |        |                                  |
| Classification          | Unknown                                                    |      |                      |        |                                  |
| Location                | s3://awssampledbuswest2/ticket/spectrum/sales              |      |                      |        |                                  |
| Connection              |                                                            |      |                      |        |                                  |
| Deprecated              | No                                                         |      |                      |        |                                  |
| Last updated            | Wed Sep 30 15:35:45 GMT-700 2020                           |      |                      |        |                                  |
| Input format            | org.apache.hadoop.mapred.TextInputFormat                   |      |                      |        |                                  |
| Output format           | org.apache.hadoop.hive ql.io.HiveIgnoreKeyTextOutputFormat |      |                      |        |                                  |
| Serde serialization lib | org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe         |      |                      |        |                                  |
| Serde parameters        | field.delim                                                |      | serialization.format |        |                                  |
| Table properties        | EXTERNAL                                                   | TRUE | numRows              | 172000 | transient_lastDdlTime 1601505345 |

#### Schema

|    | Column name | Data type    | Partition key |
|----|-------------|--------------|---------------|
| 1  | salesid     | int          |               |
| 2  | listid      | int          |               |
| 3  | sellerid    | int          |               |
| 4  | buyerid     | int          |               |
| 5  | eventid     | int          |               |
| 6  | dateid      | smallint     |               |
| 7  | qtysold     | smallint     |               |
| 8  | pricepaid   | decimal(8,2) |               |
| 9  | commission  | decimal(8,2) |               |
| 10 | saletime    | timestamp    |               |

## Step 2: Querying External Tables

1. Edit the Query.
2. Click on **Run**.



Query 1



```
1 select count(*) from spectrum.sales;
```

Run



Clear

3. See the Query result (172,456 rows in our Spectrum table).

Query history

Query results

Table details

Query 1328 [↗](#)



Completed, started on September 30, 2020 at 15:39:30

ELAPSED TIME: 00 m 13 s

Rows returned (1)



Search rows

count

172456

4. To get the actual data, edit the Query.
5. Click on **Run**.



Query 1



```
1 select * from spectrum.sales limit 10;
```

Run



Clear

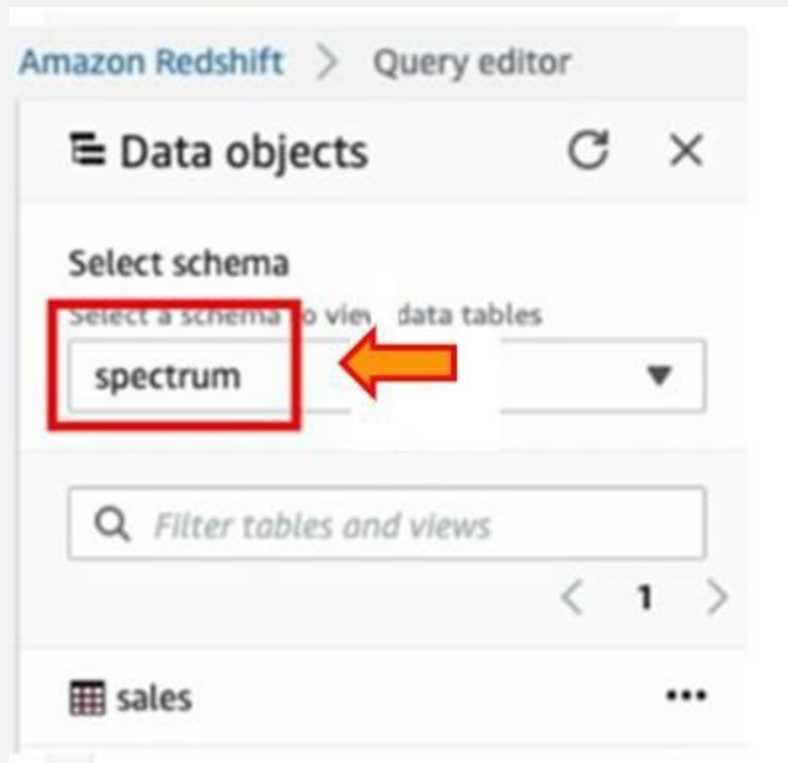
6. We have a sales ID, a list ID, the seller ID, so forth, a bunch of IDs, a many relational connection points.

Rows returned (10)

Search rows

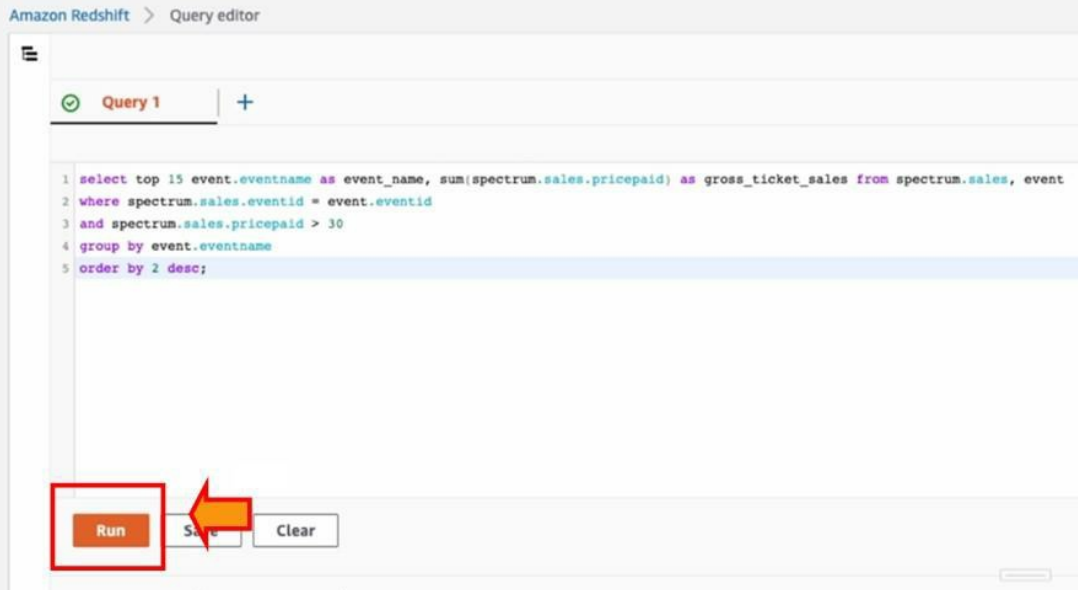
| salesid | listid | sellerid | buyerid | eventid | dateid | qtysold | pricepaid | cor  |
|---------|--------|----------|---------|---------|--------|---------|-----------|------|
| 2       | 4      | 8117     | 11498   | 4337    | 1983   | 2       | 76.00     | 11.  |
| 6       | 10     | 24858    | 24888   | 3375    | 2023   | 2       | 394.00    | 59.  |
| 7       | 10     | 24858    | 7952    | 3375    | 2003   | 4       | 788.00    | 111. |
| 8       | 10     | 24858    | 19715   | 3375    | 2017   | 1       | 197.00    | 29.  |
| 9       | 10     | 24858    | 29891   | 3375    | 2029   | 3       | 591.00    | 88.  |
| 28      | 29     | 34152    | 10978   | 7622    | 2133   | 2       | 58.00     | 8.7  |
| 29      | 29     | 34152    | 9876    | 7622    | 2131   | 1       | 29.00     | 4.3  |
| 45      | 52     | 27110    | 11635   | 2300    | 2129   | 3       | 378.00    | 56.  |
| 62      | 72     | 11258    | 10729   | 3398    | 2089   | 2       | 328.00    | 49.  |
| 63      | 72     | 11258    | 2671    | 3398    | 2080   | 1       | 164.00    | 24.  |

7. Select the "Spectrum" schema.



8. Edit the Query.

9. Click on **Run**.



10. Query result of page 1.

| Query history                                          |                    | Query results | Table details |
|--------------------------------------------------------|--------------------|---------------|---------------|
| Query 1408 <a href="#">🔗</a>                           |                    |               |               |
| ✓ Completed, started on September 30, 2020 at 15:43:07 |                    |               |               |
| ELAPSED TIME: 00 m 13 s                                |                    |               |               |
| Rows returned (15)                                     |                    |               |               |
| <input type="text" value="Search rows"/>               |                    |               |               |
| event_name                                             | gross_ticket_sales |               |               |
| Mamma Mia!                                             | 1135065.00         |               |               |
| Spring Awakening                                       | 972372.00          |               |               |
| The Country Girl                                       | 910237.00          |               |               |
| Macbeth                                                | 862302.00          |               |               |
| Jersey Boys                                            | 811511.00          |               |               |
| Legally Blonde                                         | 804272.00          |               |               |
| Chicago                                                | 790630.00          |               |               |
| Spamalot                                               | 713816.00          |               |               |
| Hedda Gabler                                           | 660843.00          |               |               |
| Thurgood                                               | 639728.00          |               |               |

11. Query result of page 2.

|                                                      |                    |                          |               |
|------------------------------------------------------|--------------------|--------------------------|---------------|
| Query history                                        |                    | Query results            | Table details |
| Query 1408                                           |                    | Execution Data Visualize |               |
| Completed, started on September 30, 2020 at 15:43:07 |                    |                          |               |
| ELAPSED TIME: 00:00:13                               |                    |                          |               |
| Rows returned (15)                                   |                    | Export                   |               |
| Search rows                                          |                    | < 1 2 > ⌕                |               |
| event_name                                           | gross_ticket_sales |                          |               |
| The Seagull                                          | 626594.00          |                          |               |
| Rhinoceros                                           | 593774.00          |                          |               |
| Mystere Cirque du Soleil                             | 583180.00          |                          |               |
| Blue Man Group                                       | 567080.00          |                          |               |
| Waiting for Godot                                    | 560857.00          |                          |               |

# Lab 7-01: Querying Data from Multiple Redshift Spectrum Tables

## Introduction

This lab will utilize Redshift Spectrum to create several tables from data stored in S3 and then test to ensure we can perform queries that include joins of the tables we have created.

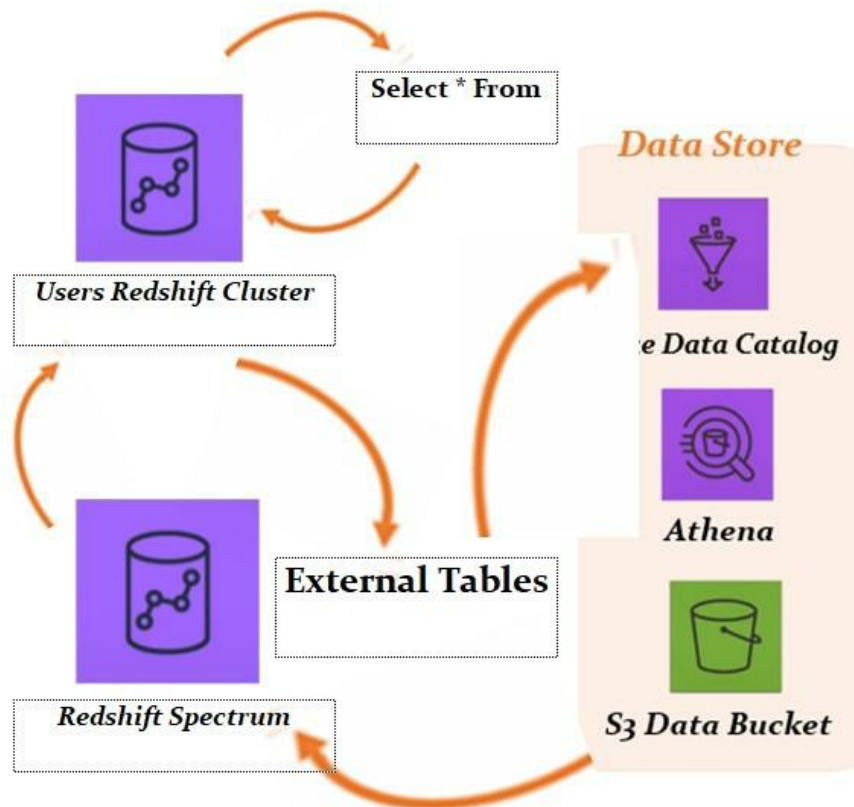


Figure 7-13: Lab Diagram

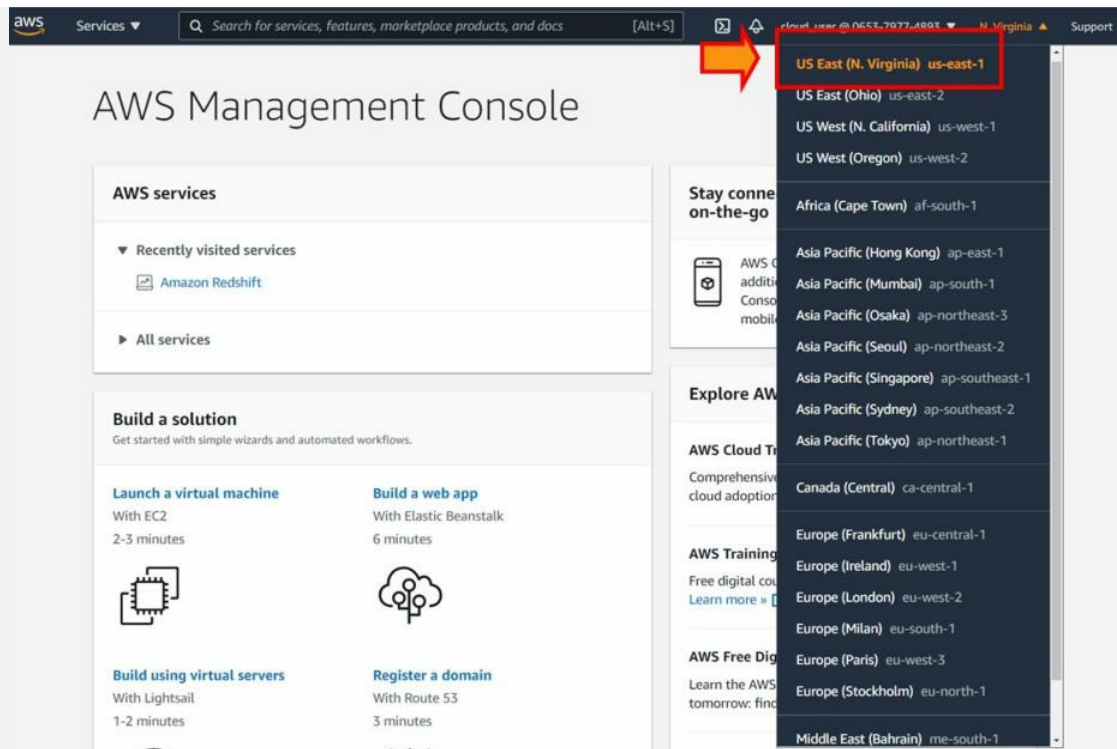
## Problem

You are working in an organization as an AWS architect. You have been tasked with testing accessing data stored in S3 from a Redshift cluster. Your team hopes to realize cost savings by moving infrequently accessed non-latency sensitive data outside your company's production Redshift cluster. They have provided the required tables below with the requisite column headers and test data in an S3 bucket. You will need to create Redshift Spectrum tables with the appropriate DDL and run the provided test query to ensure Redshift Spectrum will function for the planned use case.

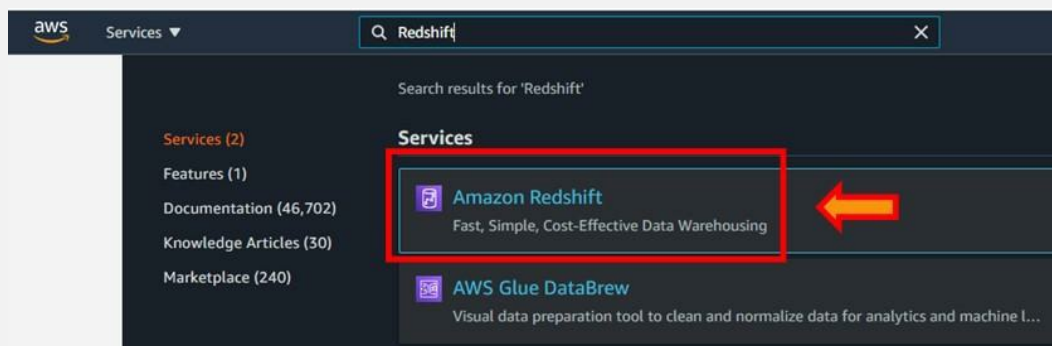
## Solution

### Step 1: Inspect the Lab Environment

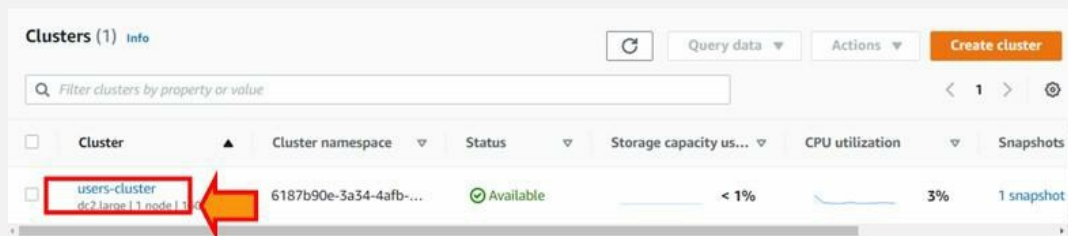
1. Log in to the AWS Management Console. Make sure you are in the **us-east-1 (N. Virginia)** region.



## 2. Search Redshift.



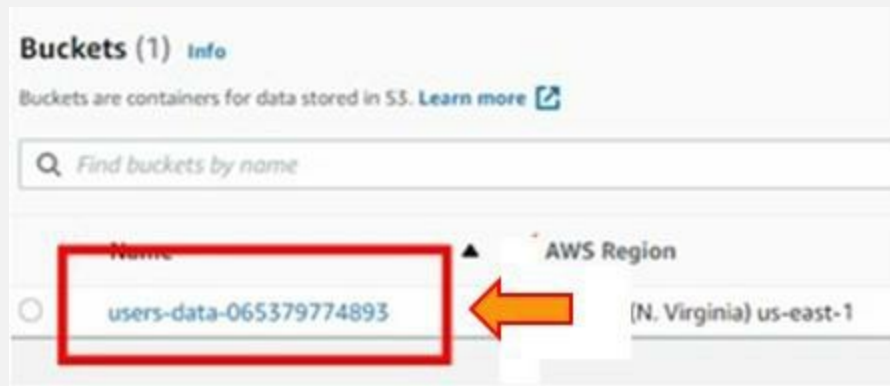
## 3. Select the Cluster.



## 4. Click on S3 under Services.

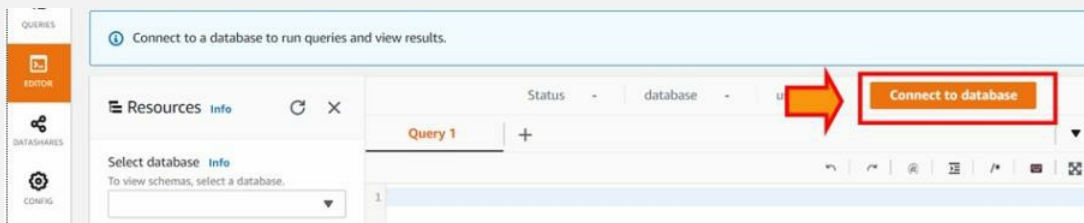


5. Click on the **bucket**.



## Step 2: Create the Redshift Spectrum Tables

1. Click **Connect to the database**.



2. Click **Connect**.

**Connect to database** [X]

**Connection**  
Select a recent database connection or create a new database connection.

☐ Use a recent connection

☒ Create a new connection

**Authentication**


☒ Temporary credentials  
Use the GetClusterCredentials IAM permission and your database user to generate temporary access credentials. [Learn more](#)

☐ AWS Secrets Manager  
Use a stored secret to authenticate access. [Learn more](#)

**Cluster**  
users-cluster (Available) ▼

**Database name**  
users

**Database user**  
User name authorized to access your database.  
users\_admini


 **Connect**

3. Select **public** schema.
4. Edit the Query.
5. Click on **Run**.

**Resources** Info [Refresh] [X]

Status Connected database users

**Select database** Info  
To view schemas, select a database.  
users ▼


**Select schema** In  
To view tables, select a schema.  
**public** ▼ 

Filter tables

No resources  
No resources to display

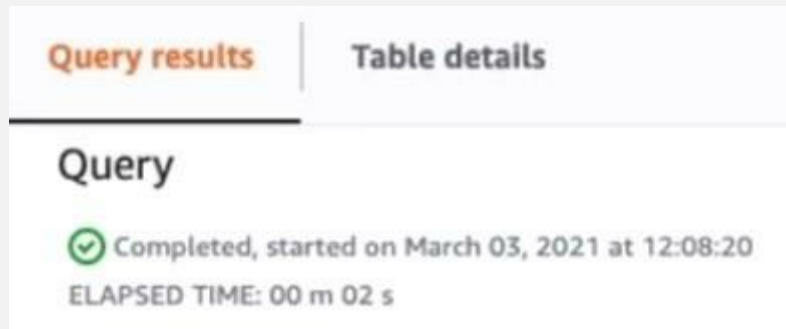
**Query 1** +

```
1 create external schema users_data
2 from data catalog
3 database 'users'
4 iam_role 'arn:aws:iam::065379774893:user/cloud_user'
5 create external database if not exists
```

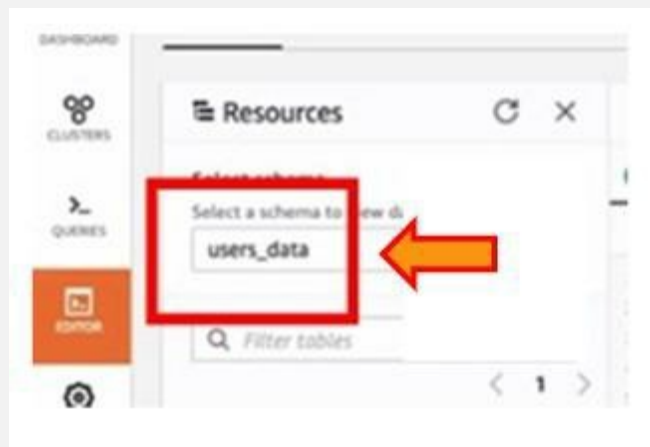
 **Run** Schedule Clear

6. There is nothing in this schema yet, so you need to create

your tables.



7. Use the Select schema dropdown on the left to select **users' data**.



8. Edit the Query.
9. Click on **Run**.

✓ Query 1

+

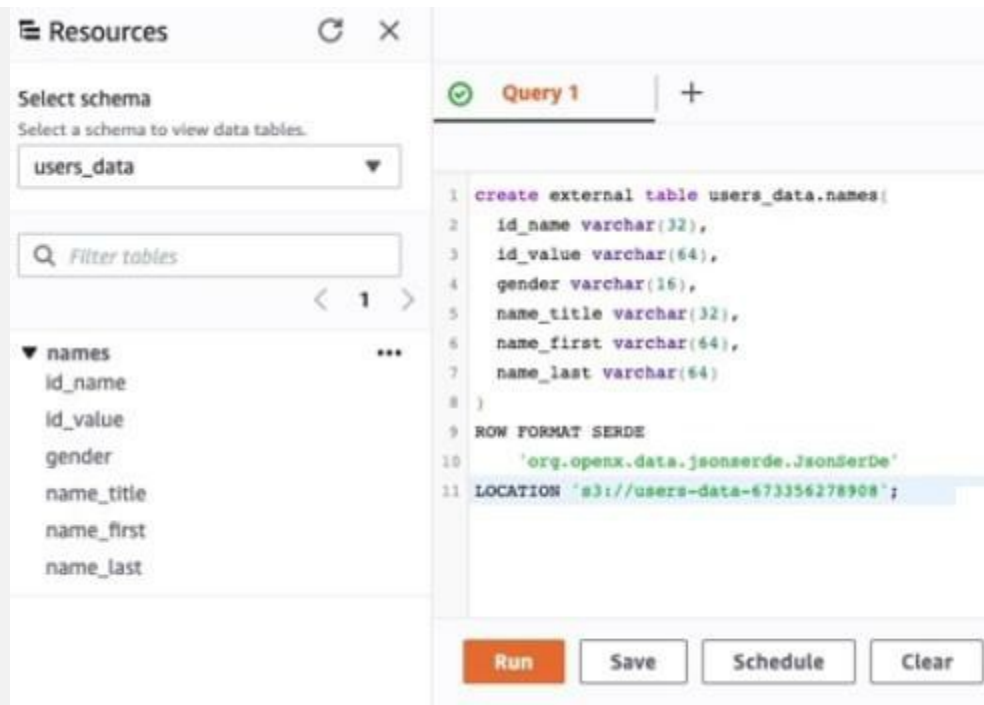
```
1 create external table users_data.names(  
2   id_name varchar(32),  
3   id_value varchar(64),  
4   gender varchar(16),  
5   name_title varchar(32),  
6   name_first varchar(64),  
7   name_last varchar(64)  
8 )  
9 ROW FORMAT SERDE  
10   'org.openx.data.jsonserde.JsonSerDe'  
11 LOCATION 's3://users-data-673356278908';
```

Run

Schedule

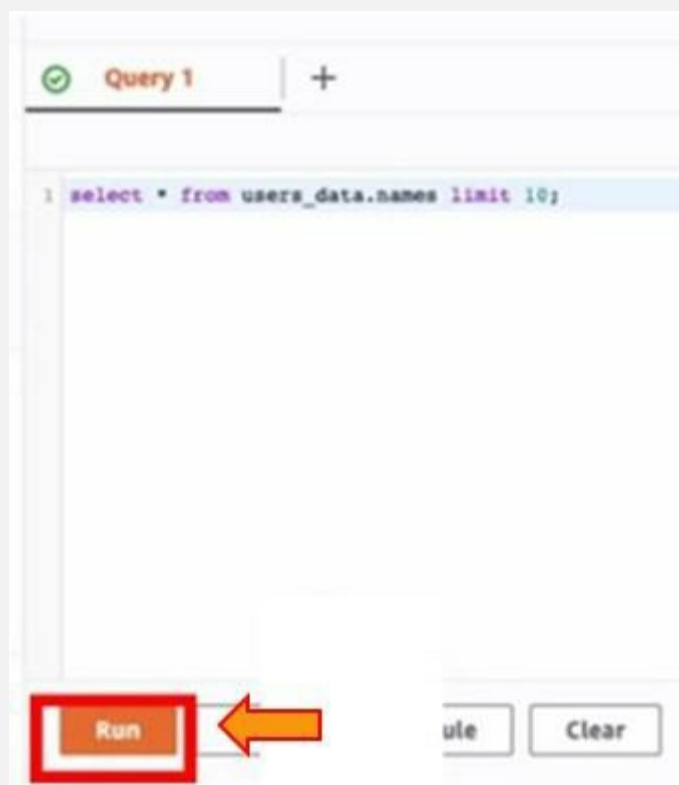
Clear

10. After the query is complete, you can see the users' data schema on the left now shows the names table.



11. Edit the Query.

12. Click on **Run**.



13. After the query is complete, you can see the user data

schema on the left now shows the location table.

Query results | Table details

Query 347 [🔗](#)

Completed, started on March 03, 2021 at 12:13:10  
ELAPSED TIME: 00 m 07 s

Execution | Data

Rows returned (10)

🔍 Search rows

| id_name | id_value          | gender | name_title | name_first | name_last |
|---------|-------------------|--------|------------|------------|-----------|
| SSN     | 069-19-8546       | female | Miss       | Ana        | Carroll   |
| BSN     | 56257887          | female | Ms         | Chahida    | Smid      |
|         |                   | female | Mrs        | لينا       | قاسمي     |
| SSN     | 085-33-6154       | female | Mrs        | Monica     | Rogers    |
| NINO    | YB 52 84 61 Y     | female | Mrs        | Sophie     | Sanchez   |
|         |                   | female | Miss       | Bianca     | Breier    |
| INSEE   | 1NNaN05031525 41  | male   | Mr         | Loic       | Leroux    |
| INSEE   | 2NNaN79689561 11  | female | Ms         | Mia        | Leroy     |
| HETU    | NaNNA652undefined | female | Ms         | Julia      | Maki      |
| INSEE   | 2NNaN36638103 74  | female | Miss       | Lilly      | Meyer     |

14. Edit the Query.
15. Click on **Run**.

✓ Query 1

+

```
6    location_city varchar(32),
7    location_state varchar(32),
8    location_country varchar(32),
9    location_postcode varchar(32),
10   location_coordinates_latitude varchar(64),
11   location_coordinates_longitude varchar(64),
12   location_timezone_offset varchar(32),
13   location_timezone_description varchar(32),
14   nat varchar(16)
15 )
16 ROW FORMAT SERDE
17     'org.openx.data.jsonserde.JsonSerDe'
18 LOCATION 's3://users-data-673356278908';
```

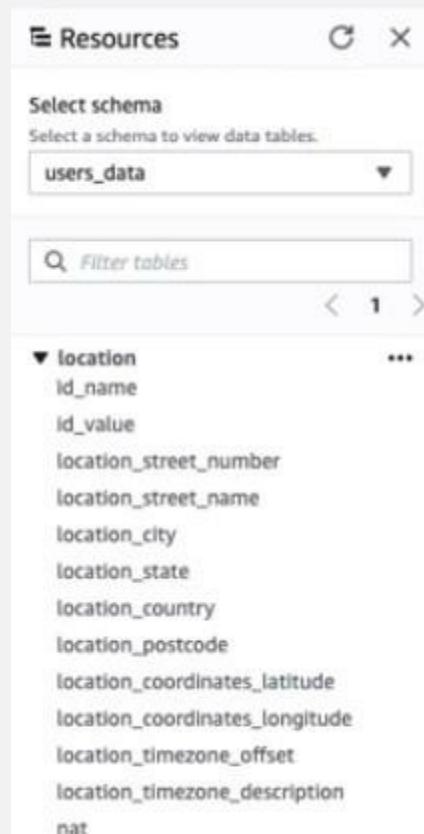
Run

←

Schedule

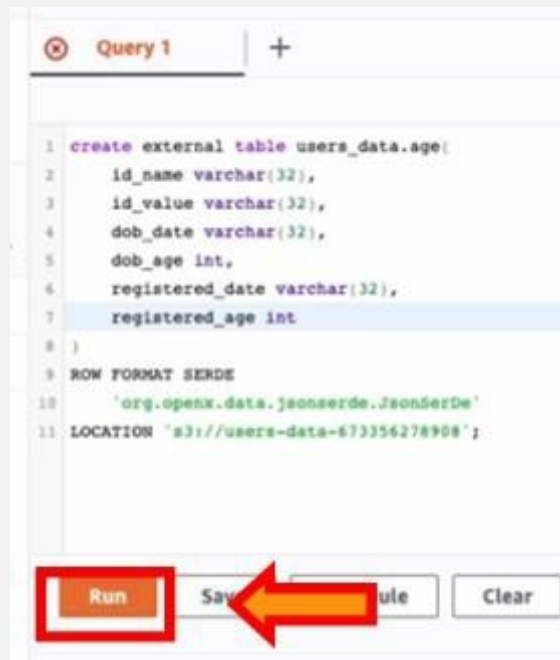
Clear

16. After the query is complete, you can see the **users\_data** schema on the left now shows the **location** table.



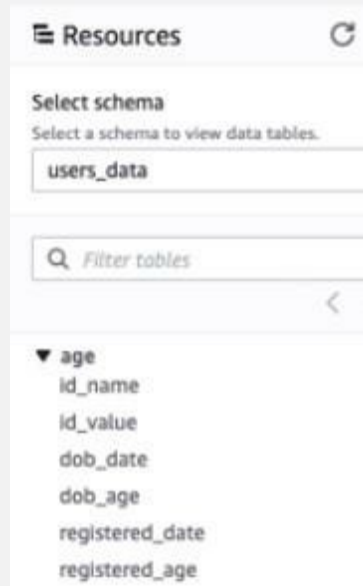
17. Edit the Query.

18. Click on **Run**.



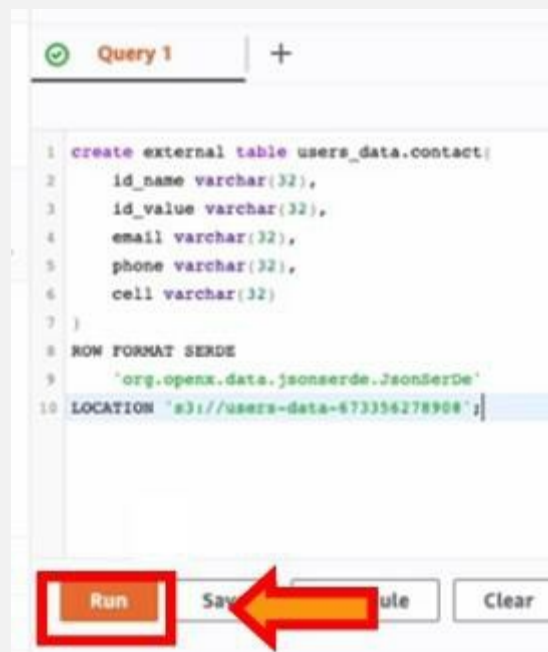
19. After the query is complete, you can see the **users' data**

schema on the left now shows the **age** table.

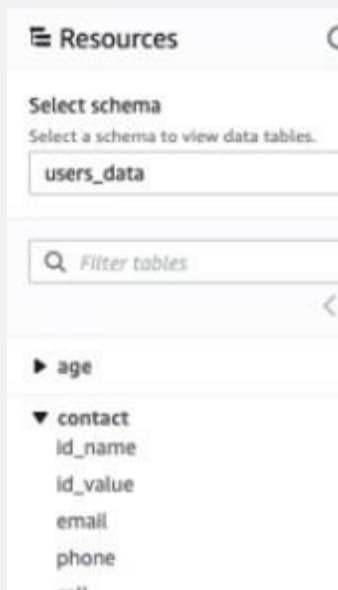


20. Edit the Query.

21. Click on **Run**.

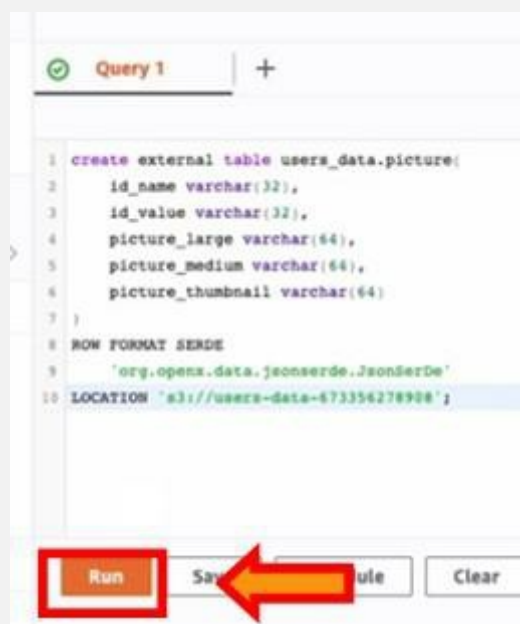


22. After the query is complete, you can see the users' data schema on the left now shows the **contact** table.



23. Edit the Query.

24. Click on **Run**.



### Step 3: Test Your Newly Created Redshift Spectrum Tables

25. Edit the Query.

26. Click on **Run**.

Query 1

```
1 select
2   names.name_first as first_name,
3   names.name_last as last_name,
4   location.location_state as state,
5   age.dob_age as age,
6   contact.cell as cell,
7   picture.picture_large as picture
8 from users_data.names
9   join users_data.location on users_data.names.id_value = users_data.location.id_value
10  join users_data.age on users_data.names.id_value = users_data.age.id_value
11  join users_data.contact on users_data.names.id_value = users_data.contact.id_value
12  join users_data.picture on users_data.names.id_value = users_data.picture.id_value
13 order by age
14 limit 10;
```

Run Schedule Clear

27. The query ensures your tables are correctly formed, and it is possible to perform joins between them.

Query results Table details

Query 495 [🔗](#) Execution Data

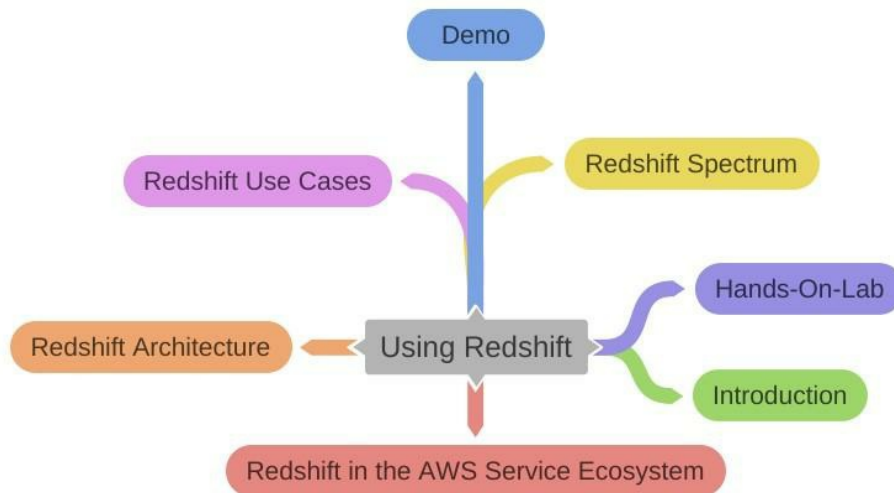
Completed, started on March 03, 2021 at 12:21:12  
ELAPSED TIME: 00 m 13 s

Rows returned (10)

Search rows

| first_name | last_name | state                        | age | cell           | picture                                                                                                         |
|------------|-----------|------------------------------|-----|----------------|-----------------------------------------------------------------------------------------------------------------|
| Harun      | Kvaale    | Hordaland                    | 23  | 42922415       | <a href="https://randomuser.me/api/portraits/men/91.jpg">https://randomuser.me/api/portraits/men/91.jpg</a>     |
| Laura      | Larsen    | Syddanmark                   | 23  | 44488502       | <a href="https://randomuser.me/api/portraits/women/91.jpg">https://randomuser.me/api/portraits/women/91.jpg</a> |
| Jeremy     | Carter    | Australian Capital Territory | 23  | 0428-040-776   | <a href="https://randomuser.me/api/portraits/men/15.jpg">https://randomuser.me/api/portraits/men/15.jpg</a>     |
| Chad       | Burke     | Laos                         | 23  | 081-695-9640   | <a href="https://randomuser.me/api/portraits/men/84.jpg">https://randomuser.me/api/portraits/men/84.jpg</a>     |
| Eric       | Armstrong | New Mexico                   | 23  | (346)-213-9189 | <a href="https://randomuser.me/api/portraits/men/56.jpg">https://randomuser.me/api/portraits/men/56.jpg</a>     |
| Daniel     | Johansen  | Midtjylland                  | 23  | 34367635       | <a href="https://randomuser.me/api/portraits/men/19.jpg">https://randomuser.me/api/portraits/men/19.jpg</a>     |
| Thomas     | Fletcher  | Waterford                    | 24  | 081-070-0212   | <a href="https://randomuser.me/api/portraits/men/82.jpg">https://randomuser.me/api/portraits/men/82.jpg</a>     |
| Anna       | Chevalier | Mayenne                      | 24  | 06-61-90-73-53 | <a href="https://randomuser.me/api/portraits/women/40.jpg">https://randomuser.me/api/portraits/women/40.jpg</a> |
| Irfan      | Korterink | Groningen                    | 25  | (714)-884-3628 | <a href="https://randomuser.me/api/portraits/men/26.jpg">https://randomuser.me/api/portraits/men/26.jpg</a>     |
| Guillaume  | Bourgeois | Neuchâtel                    | 25  | 078 413 17 88  | <a href="https://randomuser.me/api/portraits/men/68.jpg">https://randomuser.me/api/portraits/men/68.jpg</a>     |

# Mind Map



*Figure 7-14 Mind Map*

## Practice Questions

1. Which service is a data warehousing service?
  - A. Redshift
  - B. S<sub>3</sub>
  - C. IAM
  - D. None of the above
2. Which service can store petabytes and exabytes of data in S<sub>3</sub>?
  - A. S<sub>3</sub>
  - B. Redshift
  - C. IAM
  - D. None of the above

3. What tool has nodes that are either leader node or worker nodes?

A. Node

B. Slice

C. Cluster.

D. None of the above

4. What is the individual compute resources that have storage attached for the Redshift cluster?

A. Node

B. Slice

C. Cluster.

D. None of the above

5. Which of the following can compress columns in Redshift?

A. MySQL

B. Postgres

C. NoSQL

D. Aurora

6. What divides the compute, memory, and storage into separate pieces?

A. Node

B. Cluster

C. Slice

D. None of the above

7. What manages the schema?

- A. Leader Node
- B. Worker Nodes
- C. Both A and B
- D. None of the above

8. What performs query execution, Slice management, and store all of the data within the Slices?

- A. Leader Node
- B. Worker Nodes
- C. Both A and B
- D. None of the above

9. Organizational container for resources is \_\_\_\_\_.

- A. Cluster
- B. Node
- C. Slice
- D. None of the above

10. Similar to an instance in RDS?

- A. Cluster
- B. Node
- C. Slice
- D. None of them

11. Which of the following relate Query plan, Query, and Execution

Scripts?

- A. Cluster
- B. Node
- C. Slice
- D. Query Process

12. Which of the following is the Logical subdivision of node resources?

- A. Cluster
- B. Node
- C. Slice
- D. None of the above

13. What performs user authentication and management?

- A. Cognito
- B. API Gateway
- C. Simple notification service
- D. DynamoDB

14. Which service lets us send emails or text messages to end-users or the administrators of the application?

- A. Cognito
- B. API Gateway
- C. Simple notification service
- D. DynamoDB

15. Which service is used as a transactional database for applications?

- A. Simple notification service
- B. API Gateway
- C. Cognito

## D. DynamoDB

# CHAPTER 08: REDSHIFT MAINTENANCE AND OPERATIONS

## Launching a Redshift Cluster

### Interfaces

There are several interfaces for launching a Redshift cluster. Examples of such include the AWS management console, the AWS CLI, and various AWS SDKs.



*Figure 8-01: Interface*

### Required Parameters

To use the AWS CLI, the parameters that need to provide when we launch a Redshift cluster using the create cluster command for the CLI, is a node type for our cluster. The cluster will have the same node type for every node in our group. We cannot configure this per node. If our cluster type is multi-node, we do not need to provide the cluster type parameter but require several nodes. If our cluster type is a single node, which is the other option for cluster type, we do not need to give the number of nodes parameter. We will need to provide a user name and a master user password. A cluster identifier must be provided to identify our cluster.

## AWS CLI

```
aws redshift create-cluster \  
--node-type dc2.large \  
--cluster-type multi-node \  
--number-of-nodes 2 \  
--Master-username john \  
--Master-user-password Strongpass1 \  
--cluster-identifier demo-cli-cluster
```

*Figure 8-02: Required Parameters*

## Considerations

|                   |                                                                               |
|-------------------|-------------------------------------------------------------------------------|
| Workload          | Do we know our workload? Is it fixed?                                         |
| Nodes             | Do we know how much storage is needed? How compute-intensive is our workload? |
| Pre-Existing Data | Are we going to need to load existing data?                                   |

*Table 8-01: Considerations*

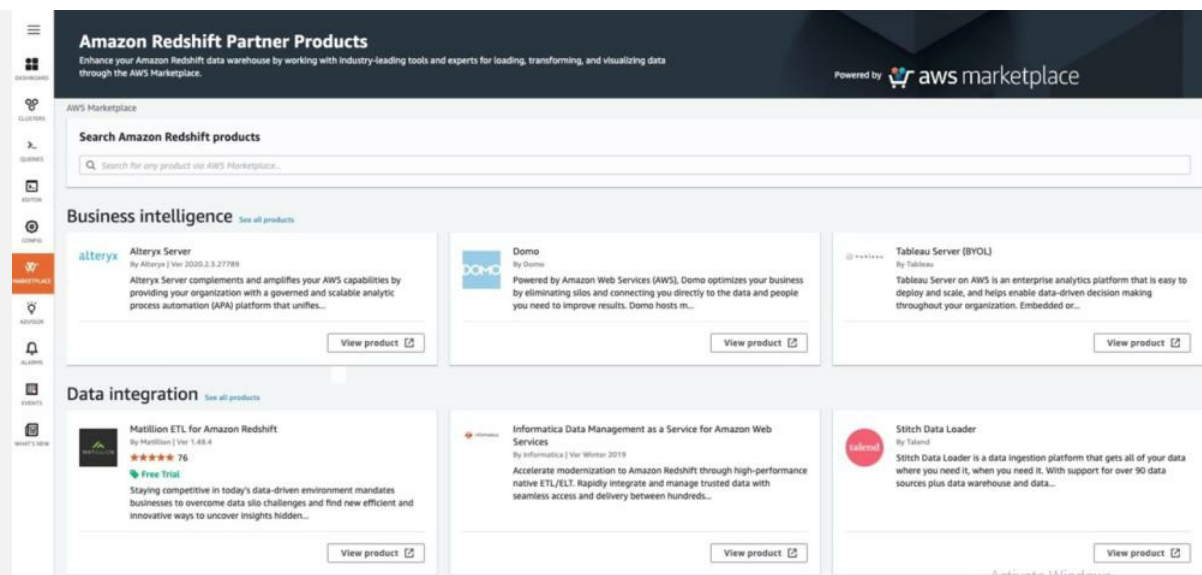
### EXAM TIP:

- Interfaces – Web console, AWS CLI, AWS SDKs.
- Required Parameters – Minimal parameters are required at launch.
- Considerations – Knowing the workload and use case helps define our cluster.

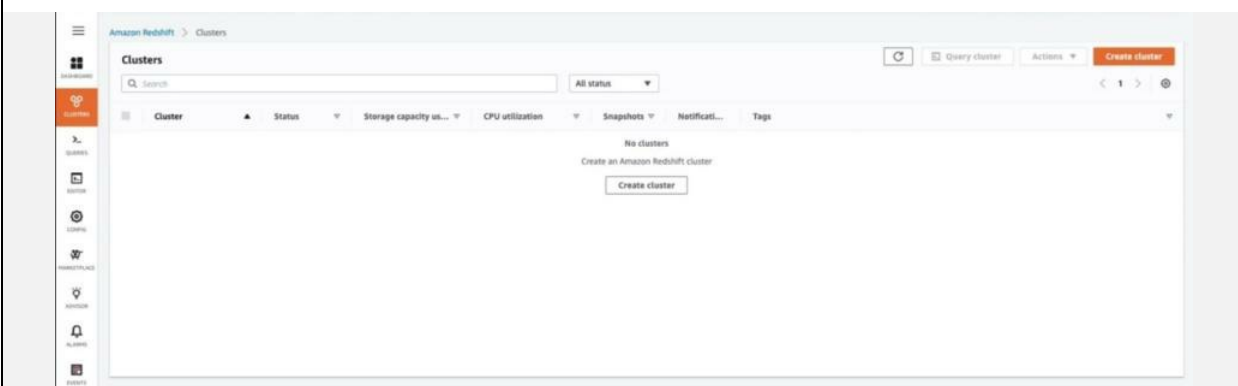
## Demo 8-01: Launching A Cluster In A Web Console

Redshift keeps track of queries. We can save SQL queries. If we run an inquiry a couple of times a year, it is always the same; we can keep

them and run them from a console or using the CLI or one of the SDKs. There are also several configuration options. We can configure some alarms about metrics. Also, a server marketplace has some services that can be plugged into Redshift. The advisor monitoring section gives us recommendations for our Redshift clusters and their workloads.



To create a new cluster. Click on Create cluster.



If you choose the production option in cluster identifier, you can configure your cluster much more minutely. If we click Free Trial, it will eliminate most of our choices. We are not going to use running because they are fairly expensive and it is also very nice that they show

the cost per hour. We do a single DC2 large node because it only costs a quarter an hour, and see that with the RA3 16X large, we are paying \$13 an hour. These are big instances.

Amazon Redshift > Clusters > Create cluster

## Create cluster

### Cluster configuration

**Cluster identifier**  
This is the unique key that identifies a cluster.

The identifier must be from 1-63 characters. Valid characters are a-z (lowercase only) and - (hyphen).

**What are you planning to use this cluster for?**

☒ **Production**  
Configure for fast and consistent performance at the best price.

☐ **Free trial**  
Configure for learning about Amazon Redshift. This configuration is free for a limited time if your organization has never created an Amazon Redshift cluster.

**Choose the size of the cluster**

**Node type**  
Choose a node type that meets your CPU, RAM, storage capacity, and drive type requirements.

| Recommended                                                                                 |                                       |
|---------------------------------------------------------------------------------------------|---------------------------------------|
| <b>RA3</b>                                                                                  |                                       |
| High performance with scalable managed storage                                              |                                       |
| <input checked="" type="radio"/> <b>ra3.4xlarge</b><br>Managed storage:<br>up to 64 TB/node | \$3.26/node/hour<br>\$0.024/GB/month  |
| <input type="radio"/> <b>ra3.16xlarge</b><br>Managed storage:<br>up to 64 TB/node           | \$13.04/node/hour<br>\$0.024/GB/month |

|                                                                  |                  |
|------------------------------------------------------------------|------------------|
| <b>DC2</b>                                                       |                  |
| High performance with fixed local SSD storage                    |                  |
| <input type="radio"/> <b>dc2.large</b><br>Storage: 160 GB/node   | \$0.25/node/hour |
| <input type="radio"/> <b>dc2.8xlarge</b><br>Storage: 2.6 TB/node | \$4.80/node/hour |

There are slower CPU options in dense storage that use a quieter storage volume to give more storage per node. If we want a big petabyte cluster, we need to use DS28X large nodes.

▼ Show legacy dense storage node types

DS2

Large workloads with fixed local HDD storage

☐

ds2.xlarge

Storage: 2 TB/node

\$0.85/node/hour

☐

ds2.8xlarge

Storage: 16 TB/node

\$6.80/node/hour

HDD




ds2.xlarge

4 vCPU (gen 2)

We have 32 nodes of dc2.large. We can use a slider to scale up or change the numbers, showing our configuration summary change. We end up with 5.1 terabytes of storage, and it would cost us almost \$6,000 a month. We need to be careful when launching Redshift resources because they get expensive quickly. Alternatively, we do an RA3 16X large scale-up; we pay \$1.2 million a month. It gives 8.2 petabytes of storage.

► Show legacy dense storage node types


**Nodes**  
Enter the number of nodes that you need.



Range (1-32)

---

**Configuration summary**  
dc2.large | 32 nodes

|                                                                                                                                                                                                                                                                   |                                                                                                                                                          |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>\$5,760.00/month</b></p> <p>Estimated on-demand compute price</p> <p>Save more than 60% of your costs by purchasing reserved nodes.</p> <p><a href="#">Learn more</a> </p> | <p><b>5.1 TB</b></p> <p>Total compressed storage</p> <p>The total storage capacity for the cluster if you deploy the number of nodes that you chose.</p> |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|

If you do not provide a database name, your default database will be named dev. We will put a strong password to satisfy most of AWS's requirements.

**Database configurations**

|                                                                                                                                                                                                                                                             |                                                                                                                                                                                                                                                             |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Database name (optional)</b><br/>Specify a database name to create an additional database.</p> <p><input type="text" value="dev"/></p> <p>The name must be 1-64 alphanumeric characters (lowercase only), and it can't be a <b>reserved word</b>.</p> | <p><b>Database port (optional)</b><br/>Port number where the database accepts inbound connections. You can't change the port after the cluster has been created.</p> <p><input type="text" value="5439"/></p> <p>The port must be numeric (1150-65535).</p> |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

---

**Master user name**  
Enter a login ID for the master user of your DB instance.

The name must be 1-128 alphanumeric characters, and it can't be a **reserved word**.

**Master user password**


☒ Show password

If you want to attach an IAM role, click on the available IAM part, select the role, and click the add IAM role.

### ▼ Cluster permissions (optional)

Your cluster needs permissions to access other AWS services on your behalf. For the required permissions, add IAM roles with the principal "redshift.amazonaws.com". You can associate up to 10 IAM roles with this cluster. [Learn more](#)

#### Available IAM roles



At

Enter ARN

Choose an existing IAM role

AWSServiceRoleForRedshift

arn:aws:iam::694501820626:role/aws-service-role/redshift.amazonaws.com/AWSServiceRoleForRedshift

SpectrumRole

arn:aws:iam::694501820626:role/SpectrumRole

Status

ed with this resource

If security groups need to be edited, we can either go to the VPC console or the EC2 console, but we will use the default, which will let us access it through the query editor in the console. We can change the subnet group. If no additional subnet groups are created, use the default.

### ▼ Network and security

#### Virtual private cloud (VPC)

This VPC defines the virtual networking environment for this cluster. Choose a VPC that has a subnet group. Only valid VPCs are enabled in the list.

Default VPC

vpc-86e25cfe

▼

 You can't change the VPC associated with this cluster after the cluster has been created. [Learn more](#) 

#### VPC security groups

This VPC security group defines which subnets and IP ranges the cluster can use in the VPC.

▼

default

sg-7c7cce32

×

#### Cluster subnet group

Choose the Amazon Redshift subnet group to launch the cluster in.

default

▼

Activate Windows  
Go to Settings to activate Windows.

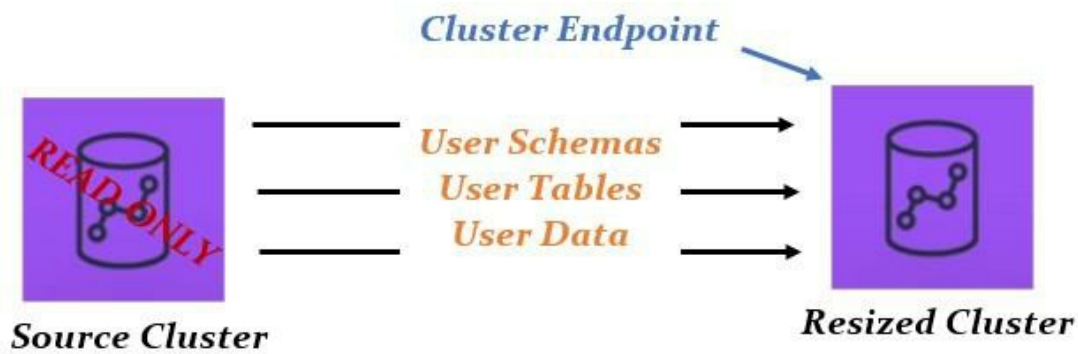
# Resizing a Redshift Cluster

## Classic Resize

- Hours To Days

Duration Impacting Factors:

- Source cluster activity.
- Size and number of tables.
- Uniformity of data distribution across nodes.
- Source and target node configuration.



*Figure 8-03: Classic Resize*

## Elastic Resize

- Minutes

Constraints:

- It cannot be used on single-node clusters.
- The cluster must be in a VPC.
- The new configuration must have sufficient storage.

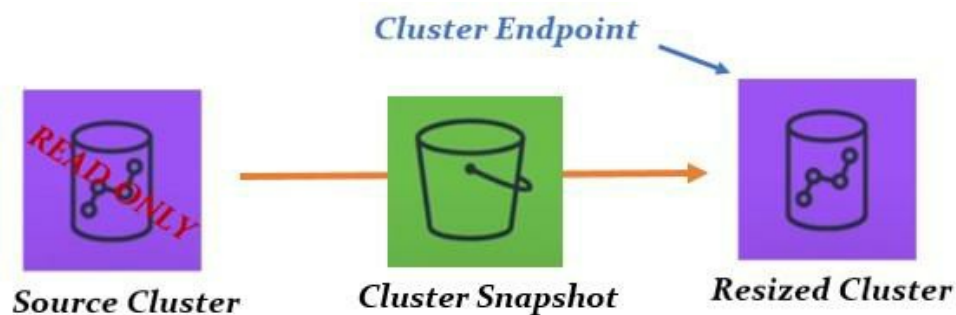


Figure 8-04: Elastic Resize

## Check Available Resize Configuration:

It is possible to use the Describe-node-configuration-options CLI command to see what compatible node configurations are available for our cluster.

### EXAM TIP:

- Classic Resize – Hours to days: only moves user objects.
- Elastic resize – Minutes: Migrate through the snapshot process.

### Using Elastic Resize

We had a single node cluster and could not do the elastic resize on a single node cluster. Go to the Options menu and select Resize.

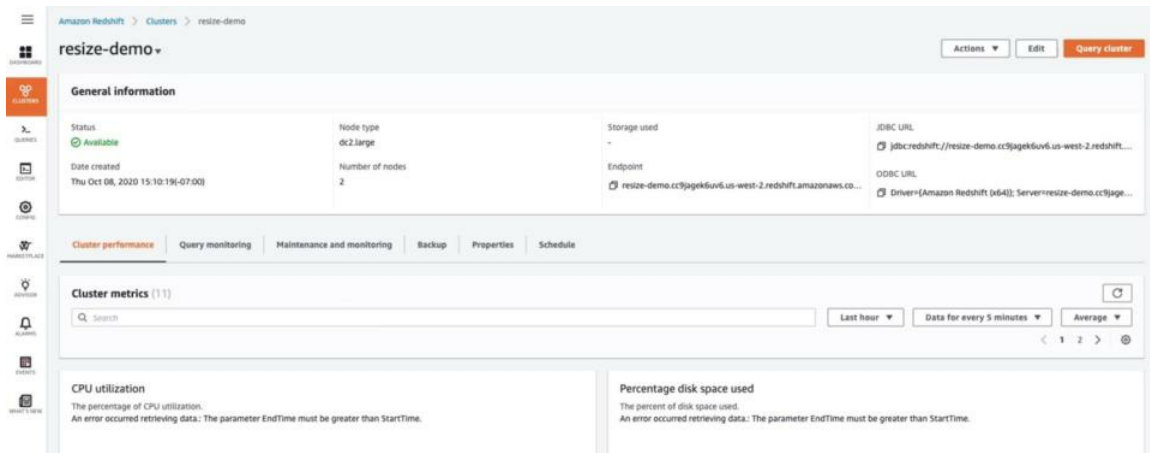


Figure 8-05: Resize-demo

There is a two-node cluster; we can do a classic resize. If we want, we also have the elastic resize option available, and then we have our resize options here.

☒ **Elastic resize (recommended)**  
 Use elastic resize to change the node type, number of nodes, or both. If you only change the number of nodes, then queries are temporarily paused and connections are held open if possible.  
  
 Cluster downtime: Read-only for 10 - 15 minutes

☐ **Classic resize**  
 Use classic resize to change the node type, number of nodes, or both. Choose this option when you are resizing to a configuration that isn't available through elastic resize. For example, to or from a single-node cluster.  
  
 Cluster downtime: Read-only for 2 hours - 2 days depending on your data's size

*Figure 8-06: Elastic Resize*

Our workload falls off at a certain time. If we want to shrink the cluster or increase the workload, we can schedule when that will occur. We can change our node type. We have dense storage and RA3 options available that have sufficient storage. We cannot go down to two nodes, so we cannot go from a multi-node cluster to a single node cluster. We cannot go from a DC2.large to the larger dense compute option, but we can change between node families.

Current cluster configuration

| Node type | Nodes | Total storage |
|-----------|-------|---------------|
| dc2.large | 2     | 320 GB        |

Estimated pricing: \$12.00/day

---

New cluster configuration

Node type

dc2.large  
 Storage type: NVMe-SSD      Storage: 160 GB/node      \$0.25/hour

Nodes

The number of compute nodes.  
 2 x 160 GB/node = 320 GB total storage  
 Choose an available resize option.

Estimated pricing: \$12.00/day

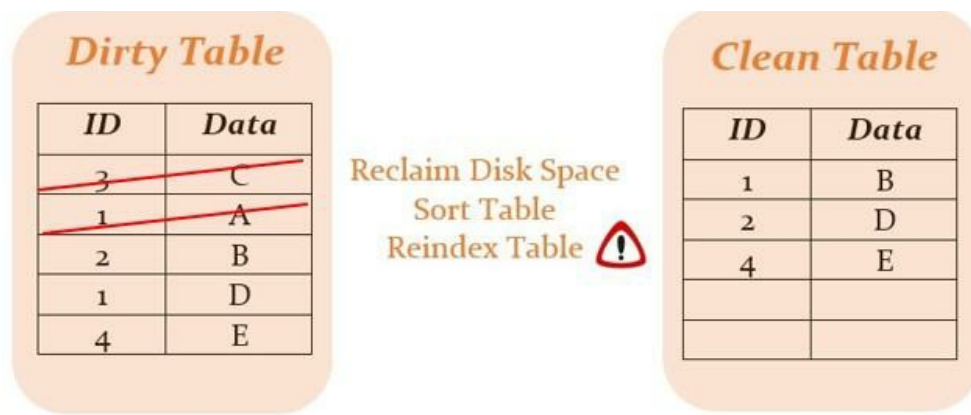
☒ Resize the cluster now  
☐ Schedule resize at a later time  
☐ Schedule recurring resize events

*Figure 8-07: Cluster configuration*

## Utilizing Vacuum and Deep Copy

## The Vacuum Process

The Vacuum process will first reclaim disk space to remove rows marked for deletion. Then it will sort the table, which will make queries more efficient, and then reindex the table to account for that new sorting so that our query planner knows exactly where all of our rows are on our table. There is an exception: tables using interleave sort keys need a special option.



*Figure 8-08: The Vacuum Process*

## Vacuum Options

|                                      |                                                                                                                            |
|--------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| <b>VACUUM FULL</b>                   | <ul style="list-style-type: none"><li>• Reclaims Disk Space.</li><li>• Sorts Rows.</li><li>• Reindexes Tables.</li></ul>   |
| <b>VACUUM SORT ONLY</b>              | <ul style="list-style-type: none"><li>• Sorts Rows.</li><li>• Default threshold 95% sorted.</li></ul>                      |
| <b>VACUUM DELETE ONLY</b>            | <ul style="list-style-type: none"><li>• Reclaims Disk Space.</li><li>• Default Threshold 5% marked for deletion.</li></ul> |
| <b>VACUUM REINDEX table<br/>name</b> | <ul style="list-style-type: none"><li>• Performs Vacuum FULL on.</li><li>• Interleaved Tables.</li></ul>                   |
| <b>To Threshold PERCENT</b>          | Sets threshold to perform SORT,                                                                                            |

|              |                                                                           |
|--------------|---------------------------------------------------------------------------|
|              | DELETE operations.                                                        |
| <b>BOOST</b> | Ignores cluster activity and contends for resources like other processes. |

*Table 8-02: Vacuum options*

## Automatic Vacuuming

### *Automatic Vacuum Delete*

- Automatically reclaim disk space. It is triggered by a high percentage of rows marked for deletion.
- Activity monitoring dictates the schedule.

### *Automatic Table Sort*

- A high percentage of unsorted Rows triggers it.
- Utilizes SCAN operations to identify unsorted tables.

### *Automatic Analysis*

- Automatically updates table statistics.
- Waits for low activity periods to analyze jobs
- Utilizes table statistics age for triggering.

*What Is Deep Copy?*

Consider starting with a dirty table. We want the same result as a full Vacuum, but we have a huge amount of unsorted data in this table. We cannot see this because it is a little table, but that is one of the use cases for Deep Copy. We will create a new table. That is the same as our existing table. We will copy the data, and then the process of copying that data will sort it and remove any rows marked

for deletion. We then truncate the source stable, take all the data out of it, and rename our target table, and we get a clean table with the same effect as a full Vacuum, but it is not as resource accelerated as a full Vacuum. It does require that table not be in active use when we do this because it can take a while, but it can take less time than a full Vacuum.



*Figure 8-09: Deep Copy*

## Deep Copy Methods

A deep copy uses a bulk insert to duplicate and repopulate a database, sorting it automatically. A deep copy is substantially faster than a vacuum if a table has a huge unsorted Region. The trade-off is that you should avoid making concurrent modifications during a deep copy unless you can track them and place the delta updates into the new table after the procedure is finished. A VACUUM operation automatically allows concurrent updating.

We can create a table and use the existing table that will use the same table definition to create our new table. We will insert a star from our existing table, drop the current table, and rename it to the original table name.

We can perform these operations through a temporary table, in which we create a temp table, **my table temp**, and select star from the existing table. We then truncate the current table, insert from our temporary table the data that has the rows marked for deletion removed, and it will sort the data in that select. We will insert that back into the original table and then drop the temporary table.

We use the first option if we need to change the DDL of the table in the process. The second option is the easiest, and the third option is the fastest.

#### *With Original Table DDL*

```
CREATE TABLE my_table-copy ( ... );  
INSERT INTO my_table-copy ( SELECT * FROM my_table );  
DROP TABLE my_table;  
ALTER TABLE my_table-copy RENAME TO my_table;
```

Create New  
Copy Data  
Truncate Source  
Rename Target

#### *Create Table Like*

```
CREATE TABLE my_table-copy ( LIKE my_table );  
INSERT INTO my_table-copy ( SELECT * FROM my_table );  
DROP TABLE my_table;  
ALTER TABLE my_table-copy RENAME TO my_table;
```

Create New  
Copy Data  
Truncate Source  
Rename Target

#### *Temp Table, Truncate*

```
CREATE TEMP TABLE my_table-temp as SELECT * FROM my_table;  
TRUNCATE my_table;  
INSERT INTO my_table ( SELECT * FROM my_table-temp );  
DROP TABLE my_table-temp;
```

Create New  
Truncate Source  
Copy Data  
Drop Temp Table

Figure 8-10: Deep Copy Methods

#### **EXAM TIP:**

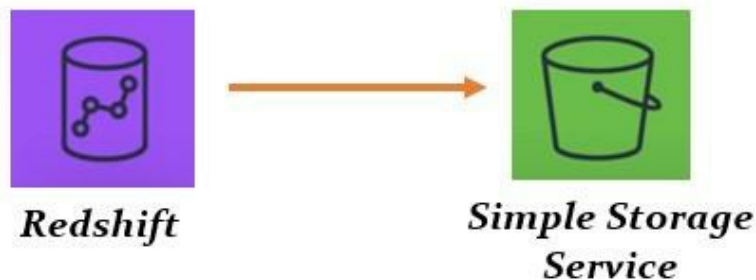
- Why vacuum? – Reclaim disk space maintain optimal query performance.

- The vacuum process - Reclaim disk space, sort Data, reindex table.
- Vacuum options – Full, sort only, delete only, and reindex, to threshold percent, boost.
- Automatic Vacuuming – Delete, sort, analyze.
- What is Deep Copy? – Process to sort large amounts of unsorted data quickly.
- Deep Copy Methods – Original table DDL, like table, temp table truncate.

## Backup and Restore

### Snapshots

- Point in time backup of the whole cluster.
- It can be manually triggered.
- Scheduled automated snapshots.



*Figure 8-11: Snapshots*

### Restoring from Snapshot

It creates a new cluster, and then that snapshot data is loaded as queries request it. Hence, as the cluster needs data to fulfill a query, it will bump that data up in the queue of loading from S3 to complete those queries quickly. RDS does this for most of its engines as well. It is not as noticeable for Redshift because of how the Redshift query process works.

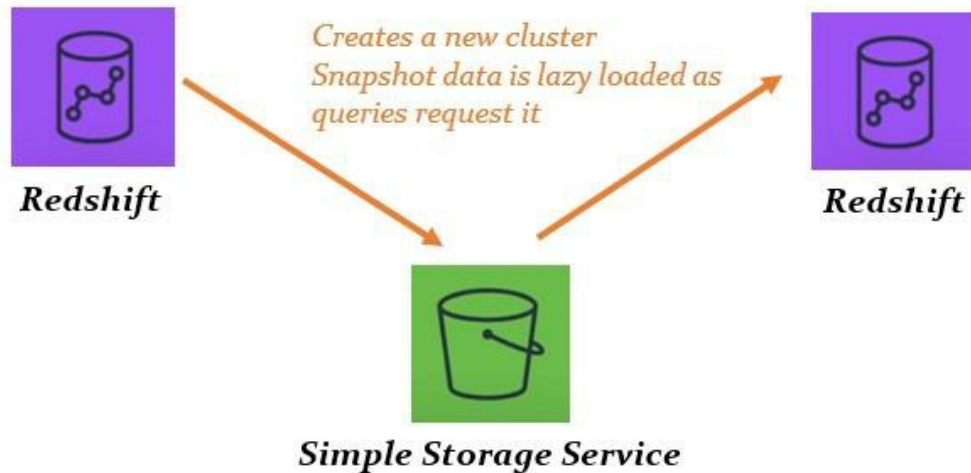


Figure 8-12: Restoring from Snapshot

## Loading Data From S3

If we want to copy single tables in and out of a backup, Redshift has very tight S3 integration. We can copy data out of S3. The IAM role permits us to read out of that S3 bucket we identified. Redshift will go and retrieve that data and load it into our table. We need to provide some information about that data that we are pulling into our bucket, but if it is formatted in a CSV or Parquet format, it can infer that information from that data. We can also use this command to copy Elastic MapReduce, DynamoDB, and SSH connections.

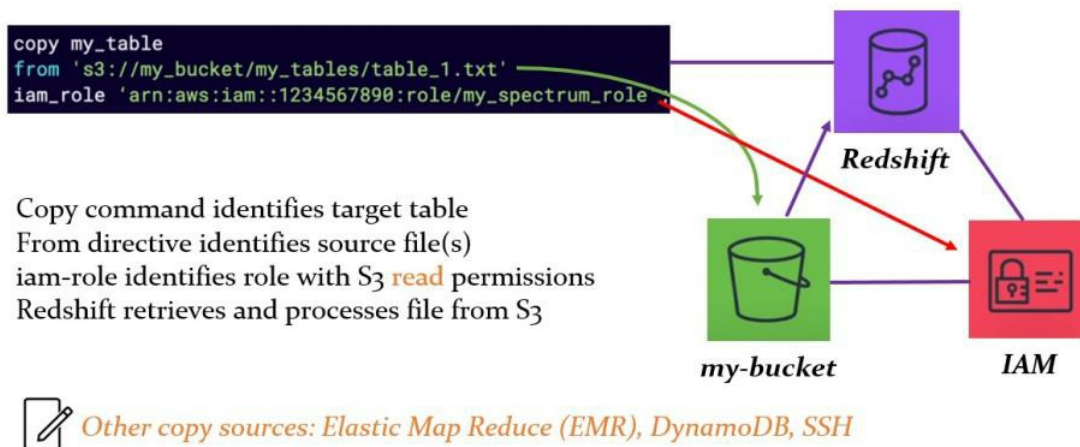
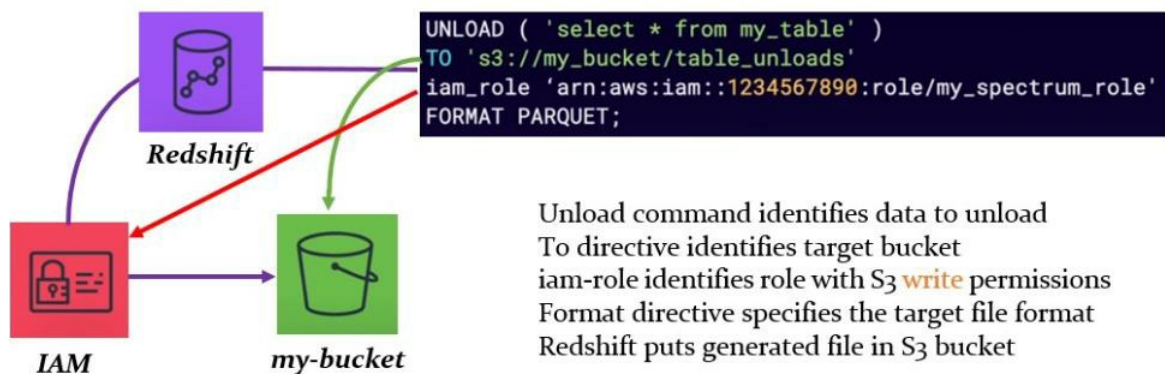


Figure 8-13: Loading Data from S3

## Unloading Data To S3

We can use the unload command and provide a piece of SQL that specifies the data we want to unload. When we give two directives, which tell the command where to save that data, we need to provide an IAM role with permissions to write into our target bucket and provide a format. If we format it as Parquet, AWS calls this lake to unload because Parquet is a common data lake format. Parquet is very efficient. We are offloading a table to S3 because we will switch part of our analytics workload to Athena. Athena works well with Parquet. We can dump out our entire Redshift cluster into S3 and then point Athena at that bucket, and we have access to the same data without running the Redshift cluster. Once all is specified, Redshift will generate a file and place it into our S3 bucket.



*Figure 8-14: Unloading Data to S3*

### EXAM TIP:

- Snapshots – Point in time backup of the entire cluster.
- Restoring from Snapshot – Creates a new cluster; cluster configuration can be modified.
- Loading Data From S3 – copy command can copy data from S3 into the existing table.
- Unloading Data To S3 – unload command can export data from query to S3 in CSV or Parquet format.

# Monitoring

## Redshift Console

Redshift console has the following services that we can monitor.

### *Cluster*

- CPU Utilization.
- Maintenance mode.

### *Storage*

- Percentage Disk Space used.
- Auto Vacuum Freed.
- Read throughput.
- Read latency.
- Write throughput.
- Write latency.

### *Database*

- Database connections.
- Total table count.
- Health status

### *Queries/Load*

- Query duration.
- Query throughput.
- Query duration per WLM queue.
- Concurrency scaling activity.
- Concurrency scaling usage.
- Average query time by priority.

### *Usage limits*

- Usage limit for concurrency scaling.
- Usage limit for Redshift Spectrum.

## CloudWatch

In the CloudWatch, we have:

## Cluster

- Commit queue length.
- Concurrency scaling sounds.
- Database connections.
- Health status.
- Maintenance mode etc.

## Node

- CPU utilization.
- Read IOPS.
- Write IOPS etc.

## Demo 8-02: Monitoring An Active Cluster

In the dashboard, we can see information about all clusters and Redshifts. It is useful if we want to monitor a fleet of Redshift clusters. This graph shows the number of queries over time and a running script that loads this cluster for about an hour and can see the result.



We use a few links that got up to 64 in database connections.



The disk space used will be a flat line because we are doing a read load against the cluster.



The CPU utilization only has one cluster. Hence, there is not a ton of information, but this gives, if we had more than one, those metrics would be available.



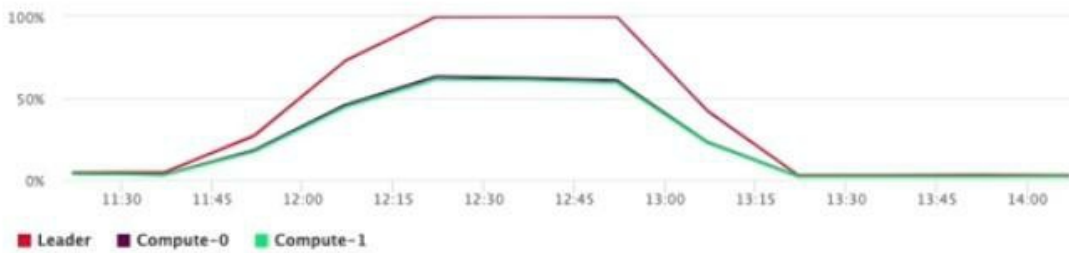
Redshift logs every query that comes to a cluster. We can change the timescale. We can put in a custom scale if we want. We can see our query overview, which gives us some throughput and the question's duration. It was running quick, short-read queries against the cluster, and we can see how long things sat in the queue, how long they took to run, and the average wait time for those queries. We could view all of those queries.



The percentage disk space used is not useful. Leader computes zero and one node CPU utilization. We can tell that our leader node was doing most of the work because it had to generate the query execution scripts to run those queries against our compute nodes.

### CPU utilization

The percentage of CPU utilization.



Query monitoring metrics are available to the query history and runtime. Therefore, we will look at the last three hours again and can see all of our various. These are all of the query run times for each one. We see which queries took longer to run than the actual queries. We can select these and terminate the queries if they are still running, which is very useful.

### Query monitoring

Parameter group: default.redshift-1.0

Top 100 queries by duration Last 3 hours Data for every 5 mins

Query history Database performance Workload concurrency

Time scale

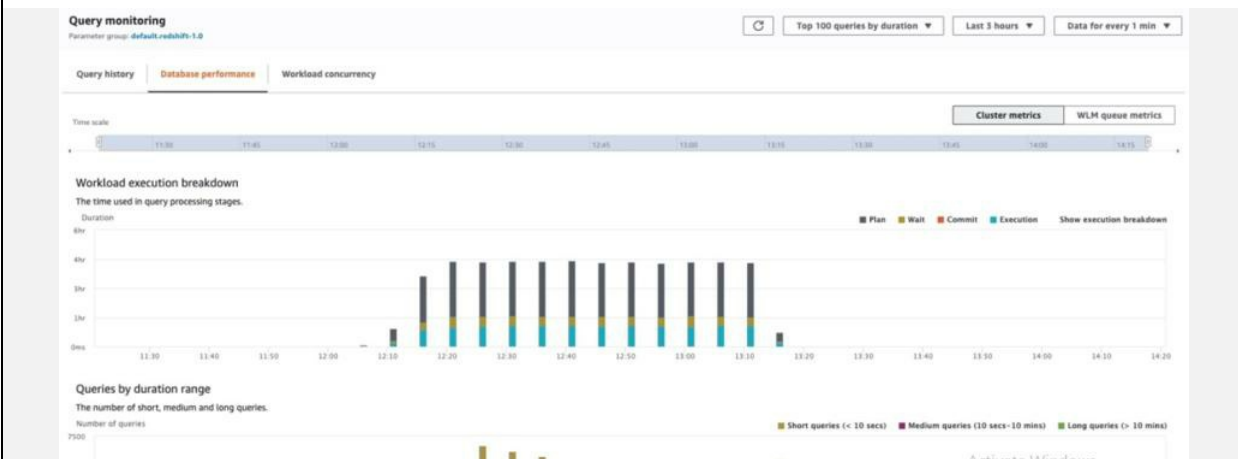
Query runtime

The query activity on a timeline. Use this graph to see which queries are running in the same timeframe. Choose a query to view more query execution details.

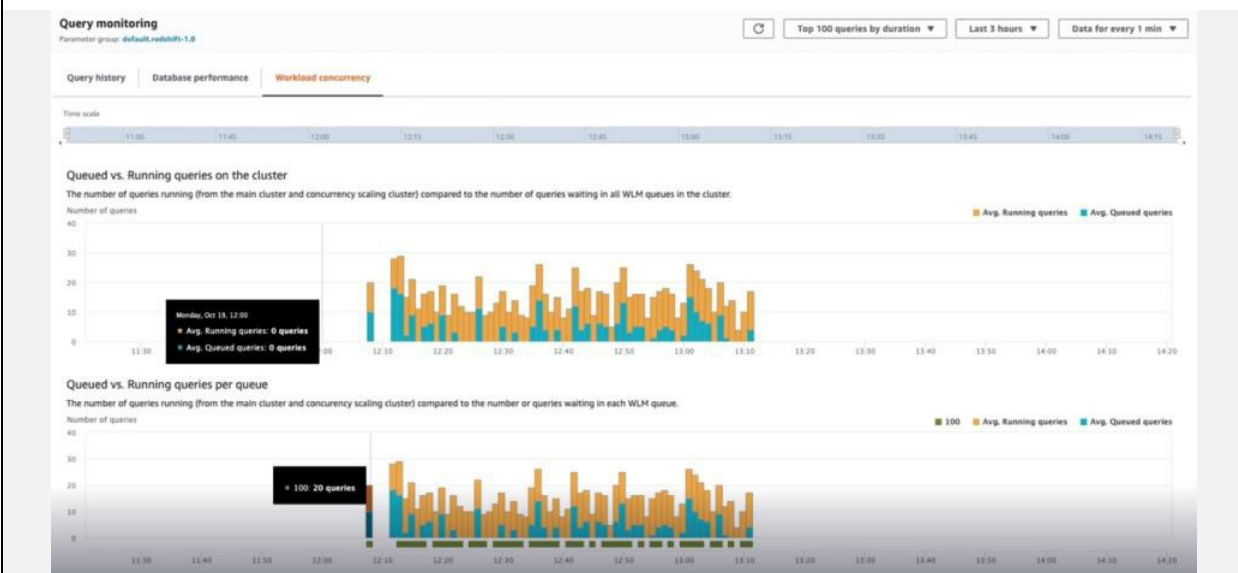


The database performance metrics will use the last three hours. We can break it down to minutes, which will change it slightly, but we can see the time used in the query processing stage. Most of our queries were short, and they took under 10 seconds. We can see the query throughput. We get an average of all of our queries, which most of our quarters are short. The standard will be low because the medium and long queries are not producing any metrics, then we have the query duration. Redshift has great monitoring tools for all of the database

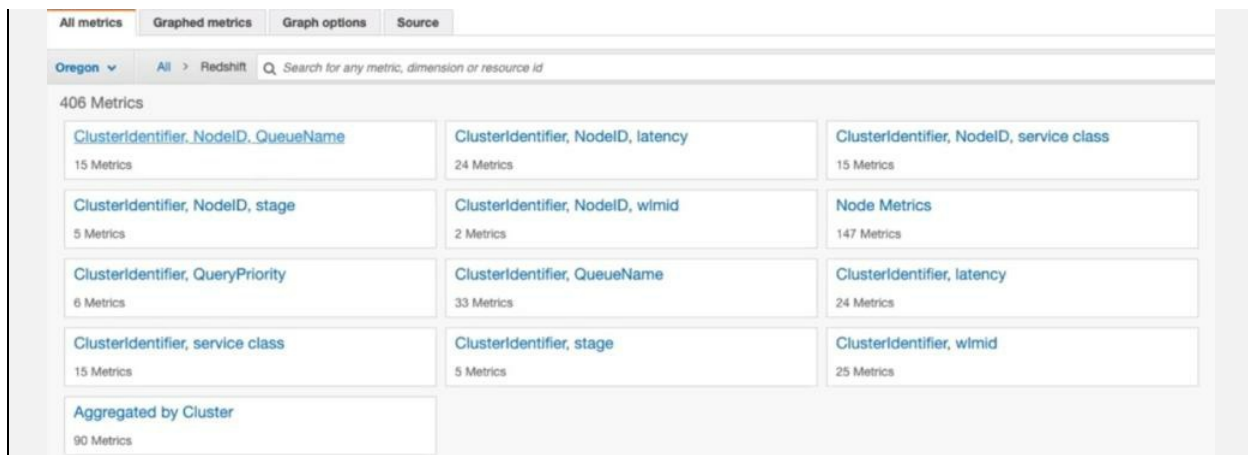
services, and it breaks a lot of this down, making it easier to look at the performance of our clusters. We have a lot of tools available for troubleshooting.



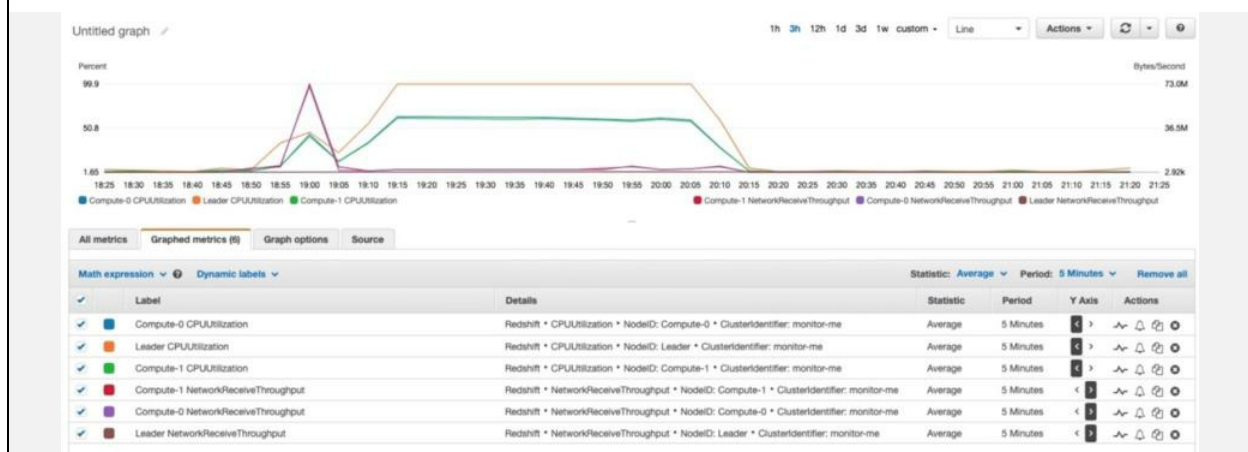
Workload concurrency gives the average running queries versus average queued queries.



In the CloudWatch console, under metrics, go to Redshift.



For example, we are going to go to node metrics. These are the nodes within our cluster. We will filter this down to the Monitor cluster. The leader node was up near 100%, and our two compute nodes were a little lower. Suppose we want to see the network received throughput. We can see that there was quite a bit of network receive on the compute nodes in our cluster. We loaded a bunch of data into our group; then we did a reading against it.



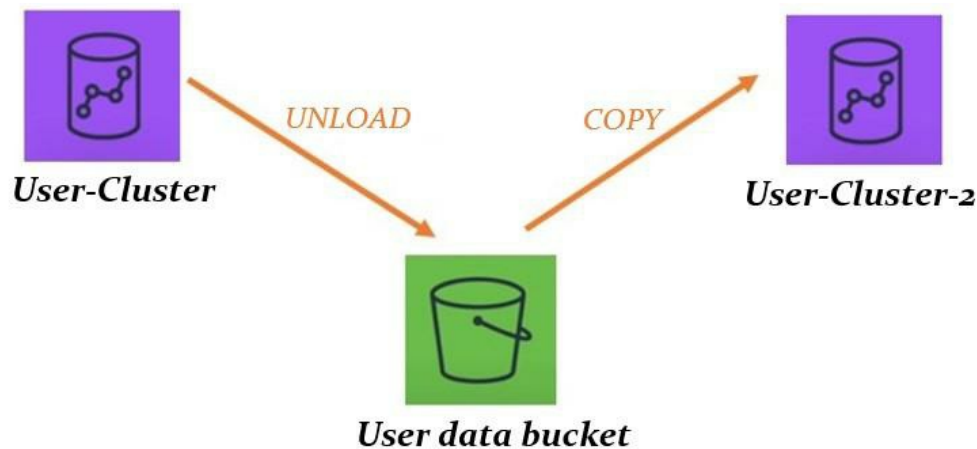
## EXAM TIP:

- Redshift Console – Cluster/node metrics, Engine specifies metrics.
- CloudWatch – provides a more granular view that can combine metric graphs.

# Lab 8-01: Manually Migrating Data Between Redshift Clusters

## Introduction

This will utilize the Redshift UNLOAD and COPY commands to emigrate data between existing Redshift clusters.



*Figure 8-15: Lab diagram*

## Problem

You have been handed over with a few pain points to clarify around your company's Redshift solution. Some groups wish to create incremental backups of certain tables to S3 in a format that can be plugged into data lake solutions. Other groups want to have select pieces of the main Redshift schema splintered to new department-specific clusters. The original Redshift cluster introduced for the company's analytics stack has become powerless over time.

You have come up with a plan to use the UNLOAD and COPY commands to facilitate all of the above and need to evaluate a proof of concept to make sure that all pain points above can be acquainted in this manner.

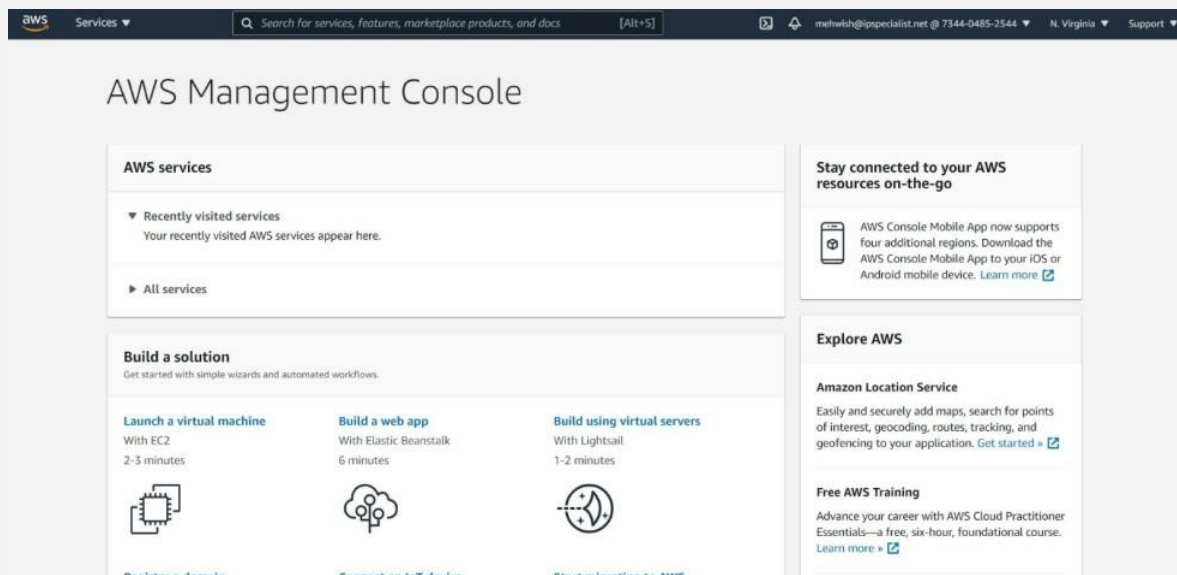
You have to reclaim an evaluated AWS account with a Redshift cluster

consisting of a relatively small test table. In the AWS account, there is an S3 bucket that will function as an intermediary storage point between Redshift clusters and an end to test backup/data lake solutions.

## Solution

### Step 01: Investigate the Lab Environment

1. Log in to **AWS Management Console**. Make sure AWS Management Console is in the **us-east-1** region.



2. Search **S3** in the search bar.

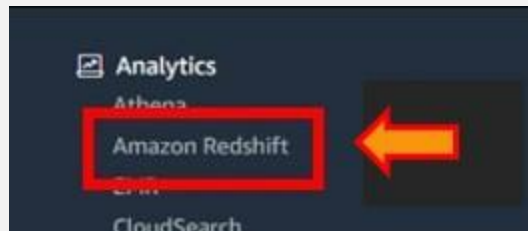


3. Select the **Bucket**.

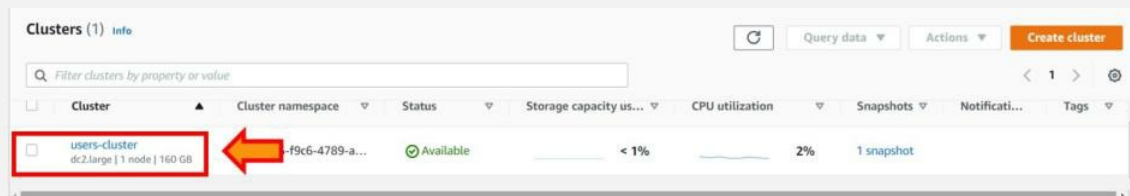


4. Search **Amazon Redshift** in the search bar.

5. Click on **Amazon Redshift**.

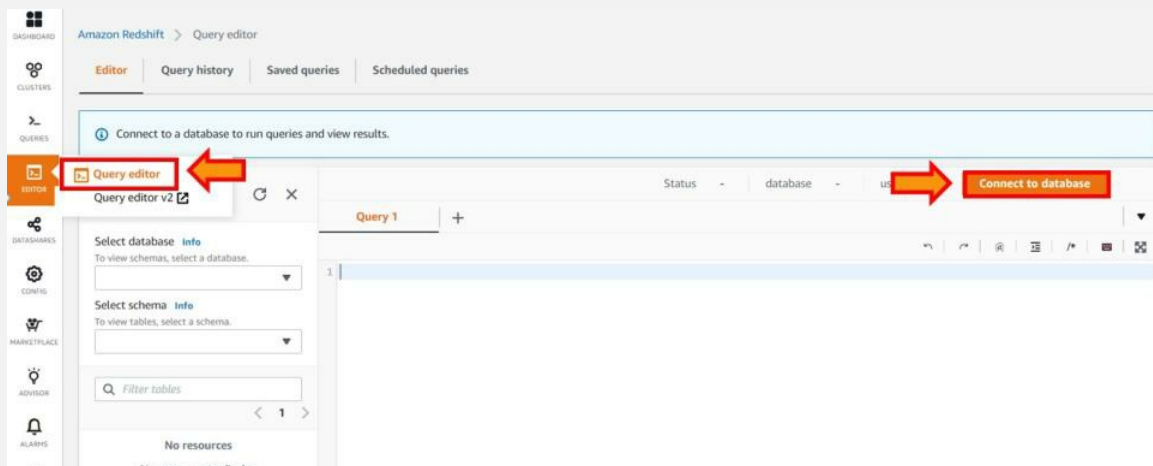


6. Click on the **Cluster**.



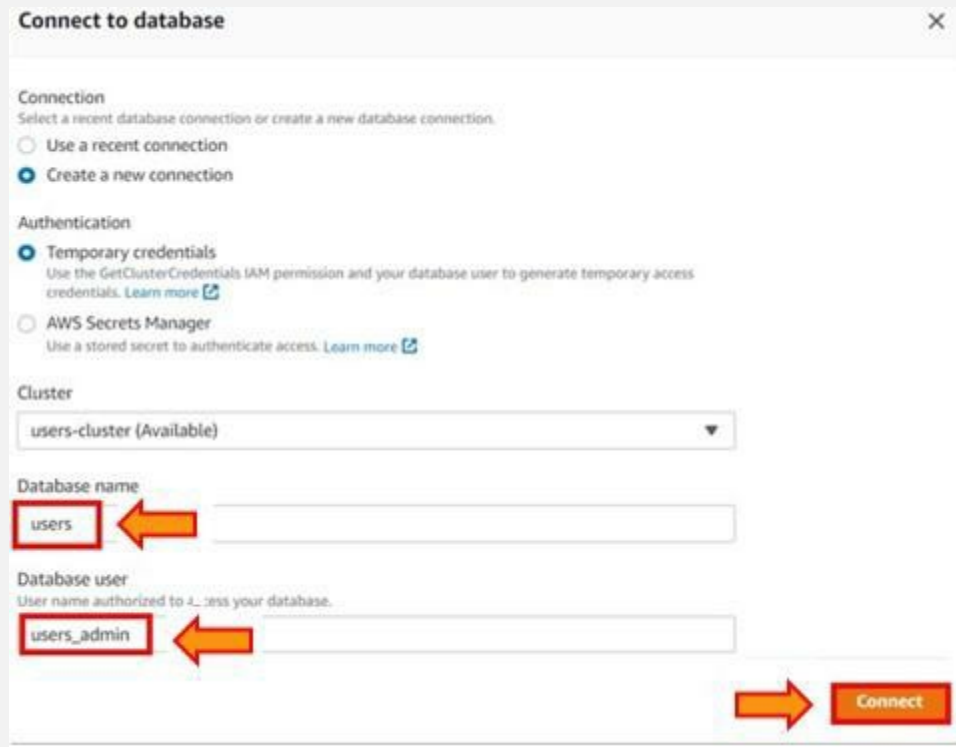
7. Click on **Quick editor**.

8. Click on **Connect to a database**.



9. Define **database name** and **user**.

10. Click on **Connect**.



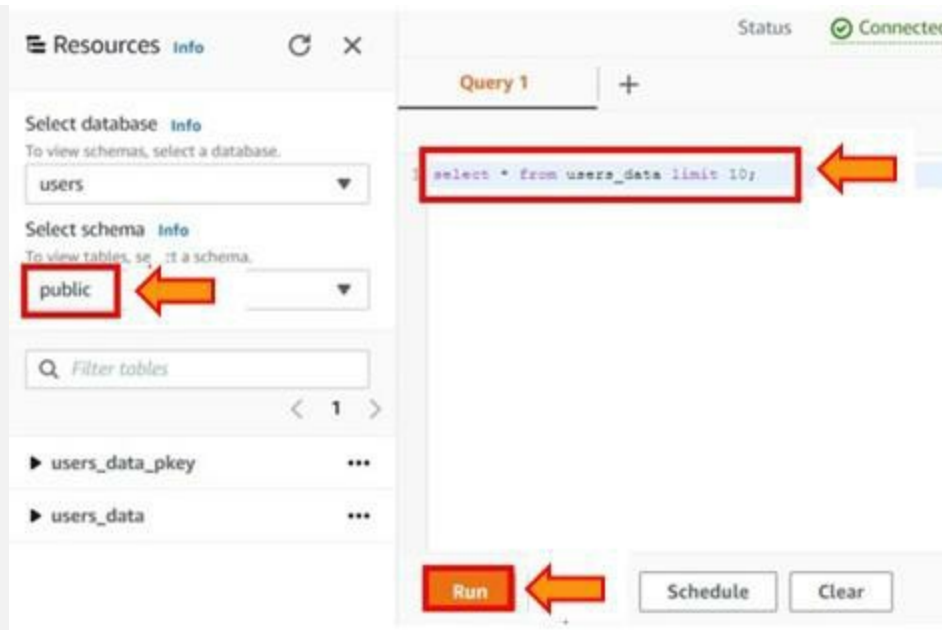
The screenshot shows a 'Connect to database' dialog box with the following sections and annotations:

- Connection:** Two radio buttons. 'Use a recent connection' is unselected. 'Create a new connection' is selected.
- Authentication:** Two radio buttons. 'Temporary credentials' is selected. Below it is a link 'Learn more'. 'AWS Secrets Manager' is unselected. Below it is a link 'Learn more'.
- Cluster:** A dropdown menu showing 'users-cluster (Available)'.
- Database name:** A text input field containing 'users'. A red box highlights the text, and a red arrow points to it from the right.
- Database user:** A text input field containing 'users\_admin'. A red box highlights the text, and a red arrow points to it from the right.
- Connect button:** A red button labeled 'Connect' with a red arrow pointing to it from the left.

11. Select the **Public** schema.

12. Edit the Query.

13. Click on **Run**.



#### 14. Review some sample data.

Query 522 [🔗](#) Execution Data Visualize

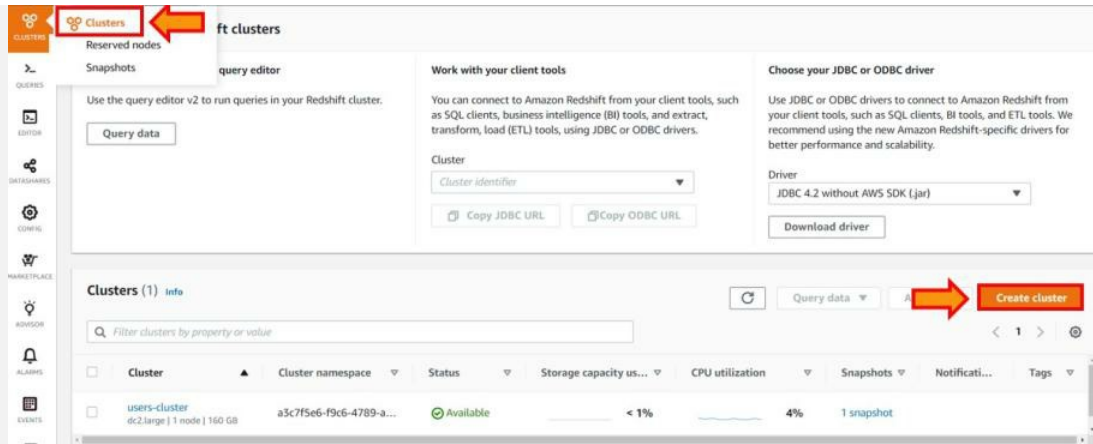
🟢 Completed, started on November 03, 2021 at 17:21:52  
ELAPSED TIME: 00 m 02 s

Rows returned (10) Export ▼

| id_value ▼ | name_first ▼ | name_last ▼ | location_country ▼ | dob_age ▼ | picture_large ▼                                                                                                 |
|------------|--------------|-------------|--------------------|-----------|-----------------------------------------------------------------------------------------------------------------|
| 00447727-Z | Jordi        | Soler       | Spain              | 70        | <a href="https://randomuser.me/api/portraits/men/42.jpg">https://randomuser.me/api/portraits/men/42.jpg</a>     |
| 0084505T   | Janet        | Boyd        | Ireland            | 72        | <a href="https://randomuser.me/api/portraits/women/17.jpg">https://randomuser.me/api/portraits/women/17.jpg</a> |
| 0243354T   | Louise       | Andrews     | Ireland            | 69        | <a href="https://randomuser.me/api/portraits/women/68.jpg">https://randomuser.me/api/portraits/women/68.jpg</a> |
| 0323582T   | Justin       | Johnson     | Ireland            | 32        | <a href="https://randomuser.me/api/portraits/men/26.jpg">https://randomuser.me/api/portraits/men/26.jpg</a>     |
| 04591322-K | Marco        | Peña        | Spain              | 65        | <a href="https://randomuser.me/api/portraits/men/4.jpg">https://randomuser.me/api/portraits/men/4.jpg</a>       |
| 0922974T   | Thomas       | Fletcher    | Ireland            | 24        | <a href="https://randomuser.me/api/portraits/men/82.jpg">https://randomuser.me/api/portraits/men/82.jpg</a>     |
| 10595294-A | Pilar        | Montero     | Spain              | 69        | <a href="https://randomuser.me/api/portraits/women/84.jpg">https://randomuser.me/api/portraits/women/84.jpg</a> |
| 1220937T   | Shane        | Fitzsimmons | Ireland            | 65        | <a href="https://randomuser.me/api/portraits/men/74.jpg">https://randomuser.me/api/portraits/men/74.jpg</a>     |
| 1408020T   | Maria        | Fields      | Ireland            | 30        | <a href="https://randomuser.me/api/portraits/women/29.jpg">https://randomuser.me/api/portraits/women/29.jpg</a> |
| 14221588-U | Vicente      | Garcia      | Spain              | 63        | <a href="https://randomuser.me/api/portraits/men/56.jpg">https://randomuser.me/api/portraits/men/56.jpg</a>     |

### Step 02: Launch the Target Redshift Cluster

1. Click on **Cluster**.
2. Click on **Create Cluster**.



3. Define the **Cluster Identifier**.

4. Select **Free Trial**.

Amazon Redshift > Clusters > Create cluster

## Create cluster Info

### Cluster configuration

#### Cluster identifier

This is the unique key that identifies a cluster.

users-cluster-2

The identifier must be from 1-63 characters. Valid characters are a-z (lowercase only) and - (hyphen).

What are you planning to use this cluster for?

☐ Production

Configure for fast and consistent performance at a lower price.

☒ Free trial

Configure for learning about Amazon Redshift. This configuration is free for a limited time if your organization has never created an Amazon Redshift cluster.

**i** When the free trial ends, delete your cluster to avoid incurring charges at **on-demand rate** [for compute and storage](#). If you want to take a final snapshot of your cluster and store the snapshot on an S3, our on-demand rate applies.

5. Define **User Name** and **Password**.

6. Click on **Create Cluster**.

## Database configurations

### Admin user name

Enter a login ID for the admin user of your DB instance.

users\_admin

The name must be 1-128 alphanumeric characters, and it can't be a [reserved word](#).

### ☐ Auto generate password

Amazon Redshift can generate a password for you, or you can specify your own password.

### Admin user password

Strongpass1

☒ Show password

Must be 8-64 characters long. Must contain at least one uppercase letter, one lowercase letter and one number. Can be any printable ASCII character except `/`, `""`, or `@`.

Create cluster

## Step 03: Copy the Existing Redshift Table to S3

1. Edit the query.
2. Click on **Connect and run**.

Query 1 +

```
1 UNLOAD ('select * from users_data')
2 TO 's3://users-data-066184308177'
3 IAM_ROLE 'arn:aws:iam::066184308177:role/RedshiftS3'
4 FORMAT AS PARQUET;
```

Connect and run ← Schedule Clear

3. Query completed.

Query results Table details

Query 675 [🔗](#)

✅ Completed, started on November 03, 2021 at 17:35:10  
ELAPSED TIME: 00 m 26 s

4. Select the **Bucket**.

Buckets (1) [Info](#)

Buckets are containers for data stored in S3. [Learn more](#) [🔗](#)

🔍 Find buckets by name

| Name                                                     | AWS Region              |
|----------------------------------------------------------|-------------------------|
| <input checked="" type="radio"/> users-data-066184308177 | us-east-1 (N. Virginia) |

←

5. Two objects were created.

**Objects (2)**

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[Refresh](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#) [Upload](#)

| <input type="checkbox"/> | Name                 | Type    | Last modified                          | Size   | Storage class |
|--------------------------|----------------------|---------|----------------------------------------|--------|---------------|
| <input type="checkbox"/> | 0000_part_00.parquet | parquet | November 3, 2021, 17:34:45 (UTC+05:00) | 8.6 KB | Standard      |
| <input type="checkbox"/> | 0001_part_00.parquet | parquet | November 3, 2021, 17:34:45 (UTC+05:00) | 6.1 KB | Standard      |

6. Select **users-cluster-2**.

**Clusters (2)** [Info](#)

| <input type="checkbox"/> | Cluster                                               | Cluster namespace       |
|--------------------------|-------------------------------------------------------|-------------------------|
| <input type="checkbox"/> | <b>users-cluster</b><br>dc2.large   1 node   160 GB   | a3c7f5e6-f9c6-4789-a... |
| <input type="checkbox"/> | <b>users-cluster-2</b><br>dc2.large   1 node   160 GB | 0-979c-42f1-b...        |

7. Go through the **cluster permission**.

**Cluster permissions (1)**

Your cluster needs permissions to access other AWS services on your behalf. For the required permissions, add IAM roles with the principal "redshift.amazonaws.com". You can also [learn more](#).

| Associated IAM roles                                                    | Status  | Amazon Resource Name (ARN)                |
|-------------------------------------------------------------------------|---------|-------------------------------------------|
| <a href="#">RedshiftS3</a><br>arn:aws:iam::740721034715:role/RedshiftS3 | in-sync | arn:aws:iam::740721034715:role/RedshiftS3 |

8. Click on **Query editor**.

9. Click on **Change connection**.

**Resources** [Info](#) [Refresh](#)

[Query editor](#) [Query editor v2](#)

**Query editor**

Select schema [Info](#)

To view tables, select a schema.

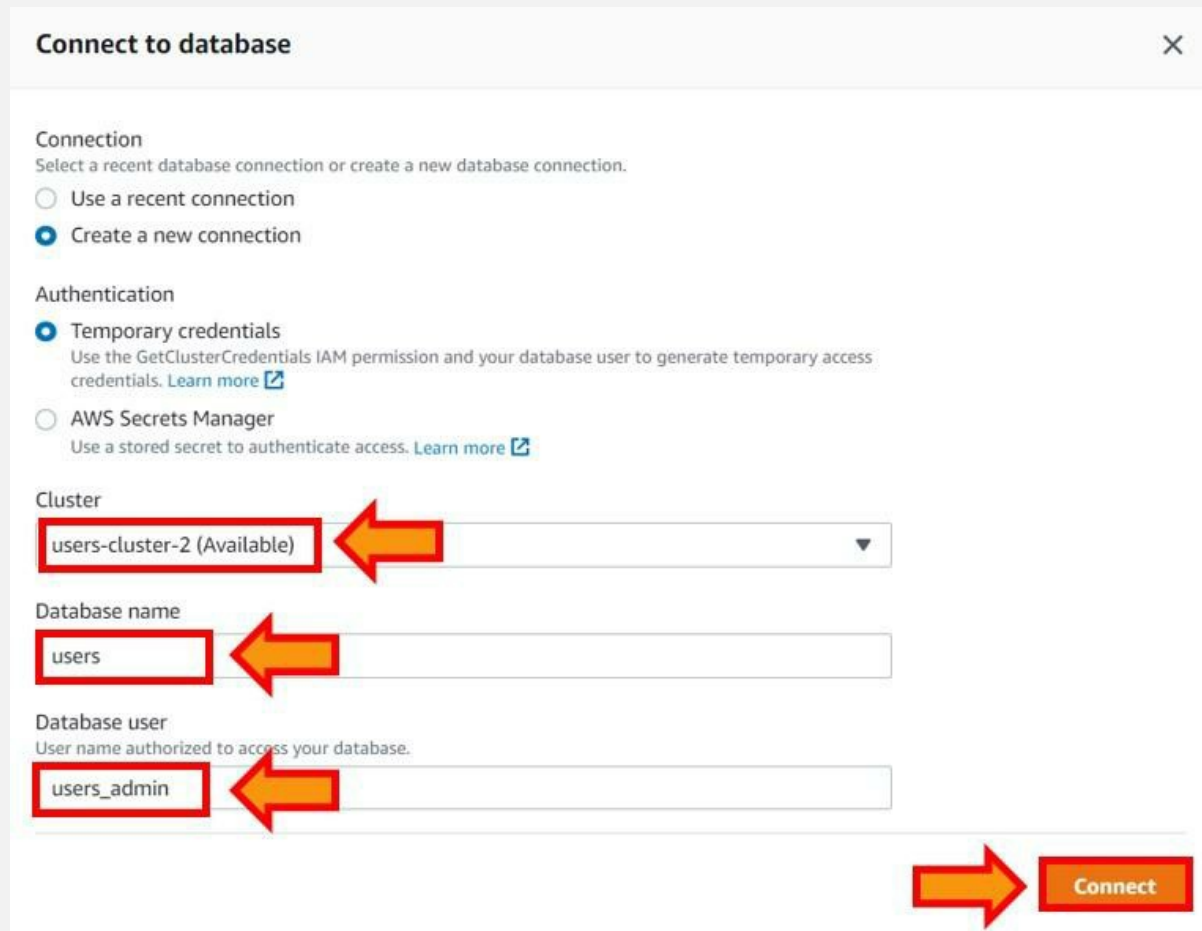
public

Status [Connected](#) database: users user: users [Change connection](#)

```
1 UNLOAD ('select * from users_data')
2 TO 's3://users-data-046184308177'
3 IAM_ROLE 'arn:aws:iam::046184308177:role/RedshiftS3'
4 FORMAT AS PARQUET;
```

10. Define **Cluster**, **Database name**, and **Database user**.

11. Click on **Connect**.



The screenshot shows a 'Connect to database' dialog box with a close button (X) in the top right corner. The dialog is divided into three sections: 'Connection', 'Authentication', and 'Cluster'. In the 'Connection' section, 'Create a new connection' is selected. In the 'Authentication' section, 'Temporary credentials' is selected. The 'Cluster' section contains three input fields: 'Cluster' (a dropdown menu showing 'users-cluster-2 (Available)'), 'Database name' (a text field with 'users'), and 'Database user' (a text field with 'users\_admin'). Each of these three fields is highlighted with a red rectangular box, and a large orange arrow points to each box from the left. A final large orange arrow points from the bottom right towards a red 'Connect' button.

**Connect to database** X

**Connection**  
Select a recent database connection or create a new database connection.

☐ Use a recent connection

☒ Create a new connection

**Authentication**

☒ Temporary credentials  
Use the GetClusterCredentials IAM permission and your database user to generate temporary access credentials. [Learn more](#)

☐ AWS Secrets Manager  
Use a stored secret to authenticate access. [Learn more](#)

**Cluster**

users-cluster-2 (Available) ▼

**Database name**

users

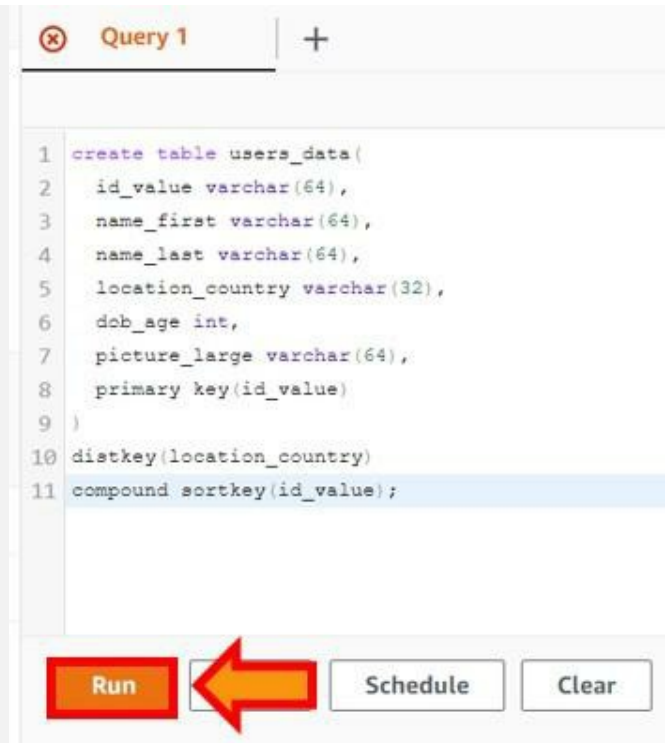
**Database user**  
User name authorized to access your database.

users\_admin

Connect

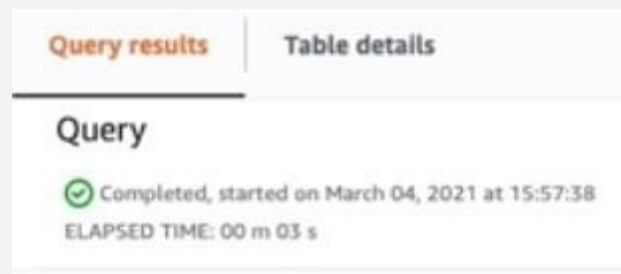
12. Edit the Query.

13. Click on **Run**.



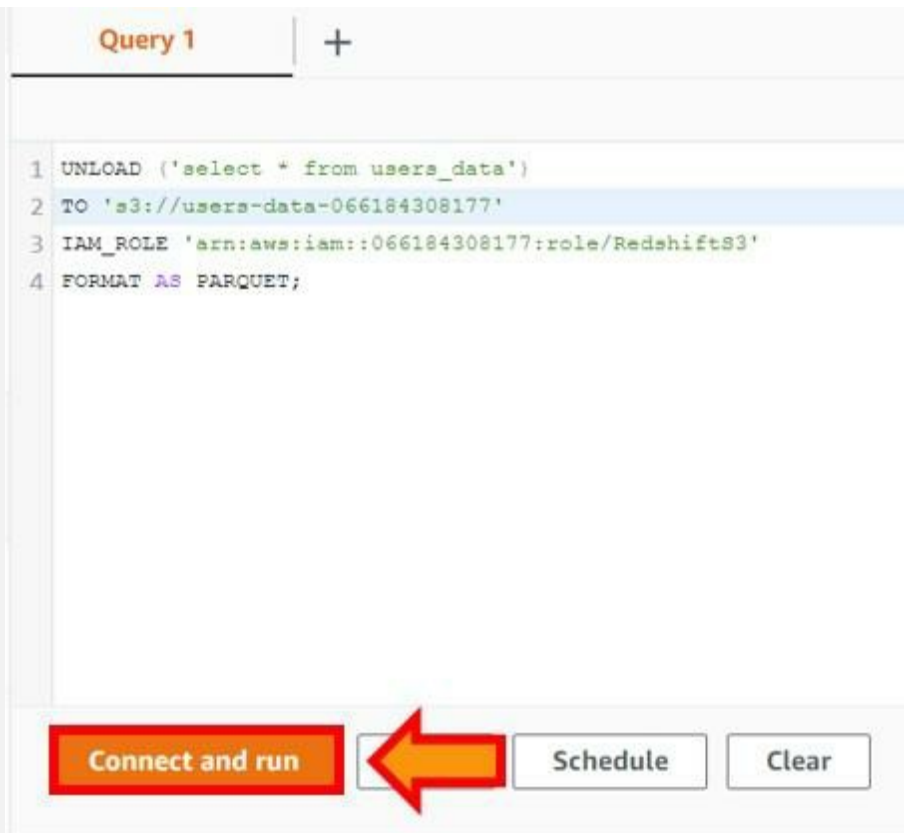
```
1 create table users_data(
2   id_value varchar(64),
3   name_first varchar(64),
4   name_last varchar(64),
5   location_country varchar(32),
6   dob_age int,
7   picture_large varchar(64),
8   primary key(id_value)
9 )
10 distkey(location_country)
11 compound sortkey(id_value);
```

14. Go through the Query result.

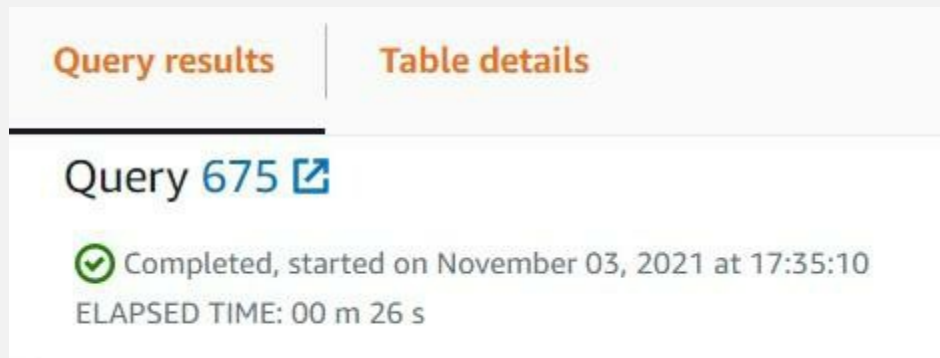


## Step 04: Copy Data from S3 to the Newly Launched Redshift Cluster

1. Edit the Query.
2. Click on **Connect and run**.



3. The data should be identical to what you originally saw in the user's cluster.



### Step 05: Check Your Data

1. Edit the Query.
2. Click on **Run**.



3. Your data should be the same as the data you noted from users-cluster-2.

Query 522 [🔗](#)

✅ Completed, started on November 03, 2021 at 17:21:52  
ELAPSED TIME: 00 m 02 s

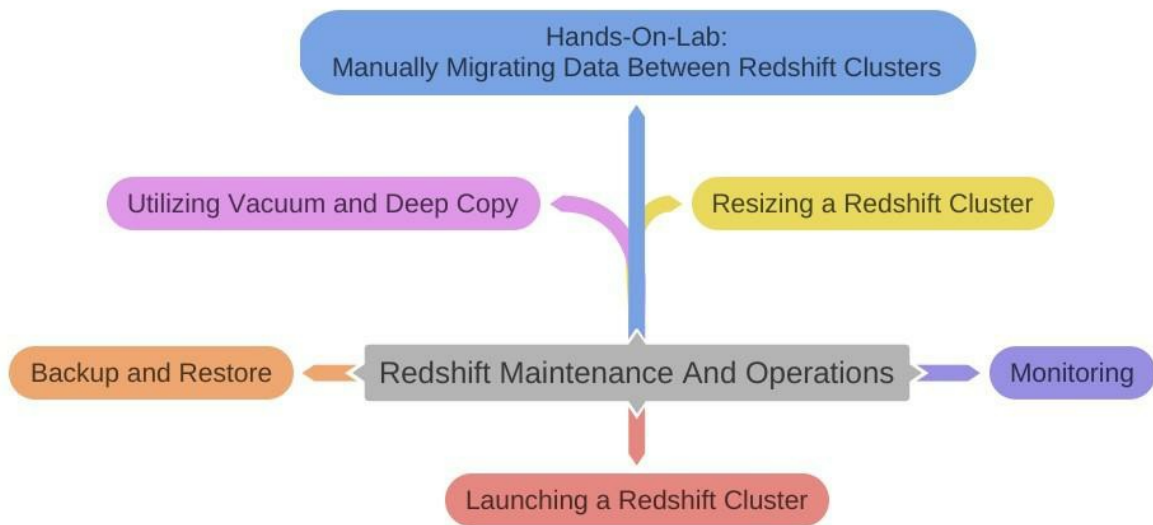
Execution Data Visualize

Rows returned (10) Export ▼

🔍 Search rows

| id_value ▼ | name_first ▼ | name_last ▼ | location_country ▼ | dob_age ▼ | picture_large ▼                                                                                                 |
|------------|--------------|-------------|--------------------|-----------|-----------------------------------------------------------------------------------------------------------------|
| 00447727-Z | Jordi        | Soler       | Spain              | 70        | <a href="https://randomuser.me/api/portraits/men/42.jpg">https://randomuser.me/api/portraits/men/42.jpg</a>     |
| 0084505T   | Janet        | Boyd        | Ireland            | 72        | <a href="https://randomuser.me/api/portraits/women/17.jpg">https://randomuser.me/api/portraits/women/17.jpg</a> |
| 0243354T   | Louise       | Andrews     | Ireland            | 69        | <a href="https://randomuser.me/api/portraits/women/68.jpg">https://randomuser.me/api/portraits/women/68.jpg</a> |
| 0323582T   | Justin       | Johnson     | Ireland            | 32        | <a href="https://randomuser.me/api/portraits/men/26.jpg">https://randomuser.me/api/portraits/men/26.jpg</a>     |
| 04591322-K | Marco        | Peña        | Spain              | 65        | <a href="https://randomuser.me/api/portraits/men/4.jpg">https://randomuser.me/api/portraits/men/4.jpg</a>       |
| 0922974T   | Thomas       | Fletcher    | Ireland            | 24        | <a href="https://randomuser.me/api/portraits/men/82.jpg">https://randomuser.me/api/portraits/men/82.jpg</a>     |
| 10595294-A | Pilar        | Montero     | Spain              | 69        | <a href="https://randomuser.me/api/portraits/women/84.jpg">https://randomuser.me/api/portraits/women/84.jpg</a> |
| 1220937T   | Shane        | Fitzsimmons | Ireland            | 65        | <a href="https://randomuser.me/api/portraits/men/74.jpg">https://randomuser.me/api/portraits/men/74.jpg</a>     |
| 1408020T   | Maria        | Fields      | Ireland            | 30        | <a href="https://randomuser.me/api/portraits/women/29.jpg">https://randomuser.me/api/portraits/women/29.jpg</a> |
| 14221588-U | Vicente      | Garcia      | Spain              | 63        | <a href="https://randomuser.me/api/portraits/men/56.jpg">https://randomuser.me/api/portraits/men/56.jpg</a>     |

## Mind Map



*Figure 8-16: Mind Map*

# Practice Questions

1. Which of the following defines several interfaces for launching a Redshift cluster?

- A. AWS Web console
- B. AWS CLI
- C. AWS SDKs
- D. All of the above

2. Which of the following is the minimal parameters required at launch?

- A. Interface
- B. Required Parameters
- C. Considerations
- D. None of the above

3. \_\_\_\_\_ means knowing the workload and use case helps define our cluster?

- A. Interface.
- B. Required Parameters
- C. Considerations
- D. None of the above

4. Which of the following defined the requirement to use vacuum?

- A. Reclaim disk space
- B. maintain optimal query performance
- C. Both A and B
- D. None of the above

5. The vacuum process is -----.

- A. Reclaim disk space
- B. Sort Data
- C. Reindex table.
- D. All of the above

6. Which of the following defines vacuum option?

- A. Full, the sort only
- B. Delete only
- C. Reindex
- D. To threshold per cent
- E. Boost
- F. All of the above

7. What is the purpose of Automatic Vacuuming?

- A. Delete
- B. Sort
- C. Analyze
- D. All of the Above

8. What is Deep Copy?

- A. The process to sort large amounts of unsorted data quickly.
- B. The process to sort small amounts of unsorted data rapidly.
- C. The process to sort large amounts of sorted data swiftly.
- D. None of the above

9. Which of the following defines the Deep Copy Methods?

- A. Original table DDL
- B. like table
- C. tempt able truncate
- D. None of the above

10. Classic Resize means -----.

- A. Hours to days: only moves user objects.
- B. Minutes: Migrate through the snapshot process
- C. Both A and B
- D. None of the above

11. Elastic resize means -----.

- A. Hours to days: only moves user objects.
- B. Minutes: Migrate through the snapshot process
- C. Both A and B
- D. None of them

12. Restoring from Snapshot is -----.

- A. Create a new cluster
- B. Cluster configuration can be modified
- C. Both A and B
- D. None of the above

13. Which of the following option is used for loading data from S3?

- A. Copy command can copy data from S3 into the existing table.
- B. Unload command can export data from query to S3 in CSV or Parquet format.
- C. Both A and B
- D. None of the above

14. Which of the following is used to unload data to S3?

- A. Copy command can copy data from S3 into the existing table.
- B. Unload command can export data from query to S3 in CSV or Parquet format.
- C. Both A and B
- D. None of them

15. Which of the following represents CloudWatch?

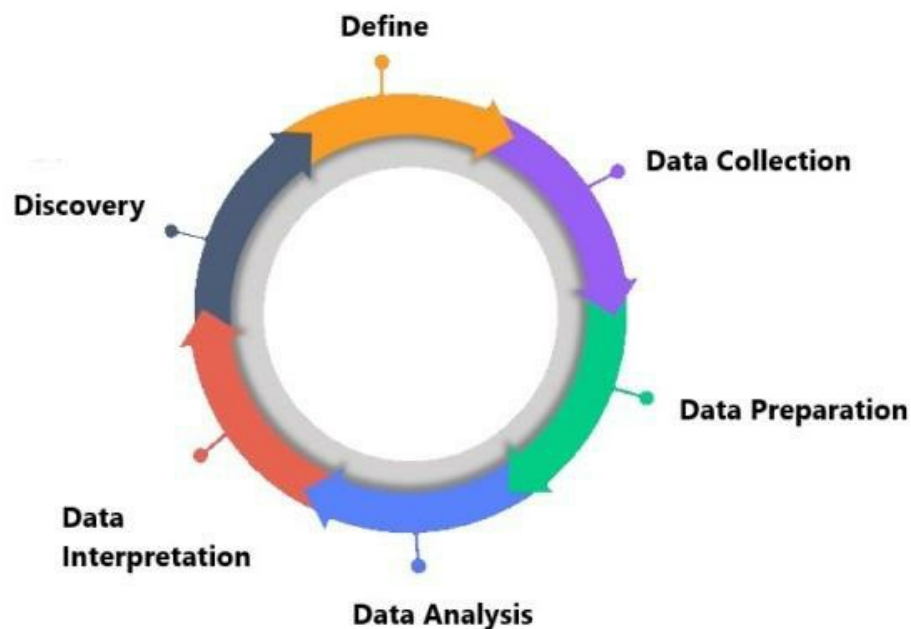
- A. Provides a more granular view
- B. Can be used to combine metric graphs
- C. Both A and B
- D. None of the above



# CHAPTER 09: AWS GLUE, ATHENA, AND QUICKSIGHT

## Introduction

Construct Glue Crawler, perform SQL queries in Athena, and create Visualization Charts with AWS QuickSight in this chapter.



*Figure 9-01: Steps to Success*

## Glue Data Catalog

### What is AWS?

#### *Serverless ETL Service*

Some important points about Serverless ETL services are:

- No server provisioning
- AWS fully manages them
- Extract Transform load

- Categorize, clean, and enrich your data
- Move data between various data stores

## **AWS Glue - Use cases**

- ***Query Data in S3***

Consider an example; you have a massive data lake in S3 with customer feedback data. You can use AWS Glue to crawl your S3 data lake to prepare tables that you can then query using Athena to see your customer feedback.

- ***Joining Data for A Data Warehouse***

Consider an example; you have a Clickstream data lake in RDS and customer data in S3. You can use Redshift Spectrum to Query your data or QuickSight to visualize the data. You can use AWS Glue to join and enrich your data and then load the results into Redshift.

- ***Creating a Centralized Data Catalog***

Consider an example; you have different types of stored in many other locations. You can use AWS Glue to manage the metadata and create a central repository. You can then access the Data Catalog for ETL and analytics with many other AWS services.

## **AWS Glue Components**

The data can be in a plethora of different locations; it can be in S3, DynamoDB, RDS, Redshift, or a database on EC2. You can also have databases outside of AWS, so as long as you can associate with them using a JDBC connection, you will access that data as a data source within AWS Glue.

Before AWS Glue knows anything about your data, you have to set up a crawler. Consider bug representing a crawler; it crawls the database and

finds important information about that data or metadata. Then, it stores that in the Data Catalog within AWS Glue. Data Catalog comprises databases that are made up of tables that you can then query or run ETL jobs on. ETL jobs are made up of either Scala code or Python code. AWS allows us to write this code ourselves manually, or it can be pre-generated. Once the job has been completed, we have some data output sources where the data will be stored onto. The data can be held onto places like S3, and then we can query it with Athena. We can store this data in a Hadoop cluster in EMR. We can keep it in S3 or hold it back into Redshift or any other JDBC connection. Not only can we store the output of these ETL jobs on services like EMR, S3, and Redshift, but we can also use services like Athena and EMR to query our data that is in our input data sources. This uses the Data Catalog the metadata about that data to be able to query it.

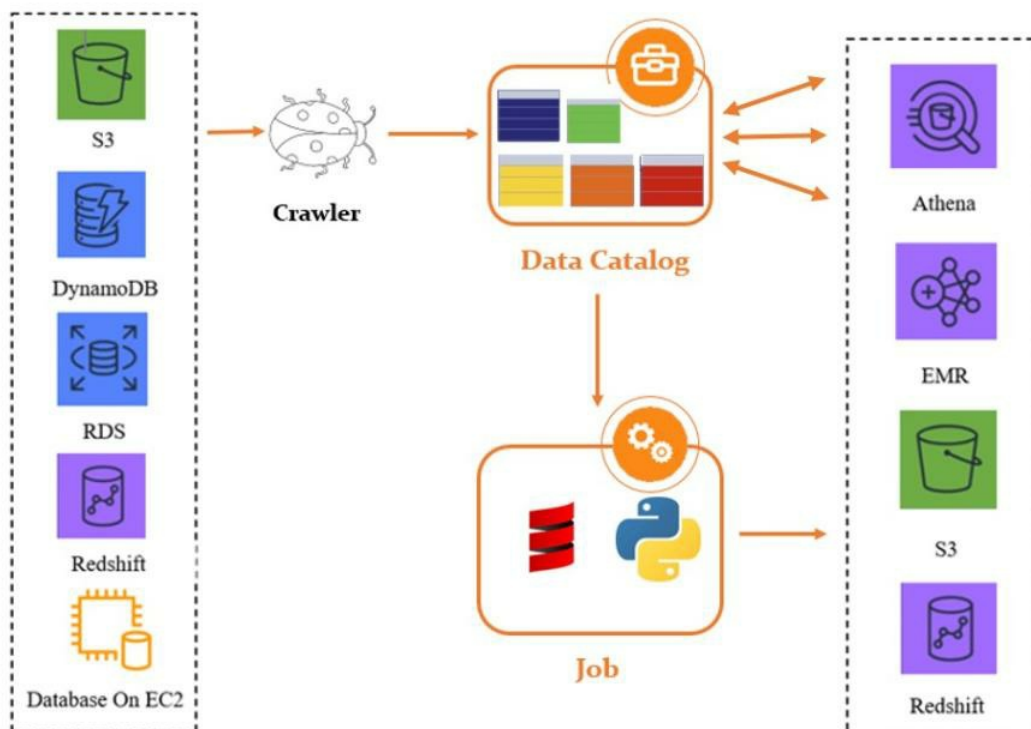
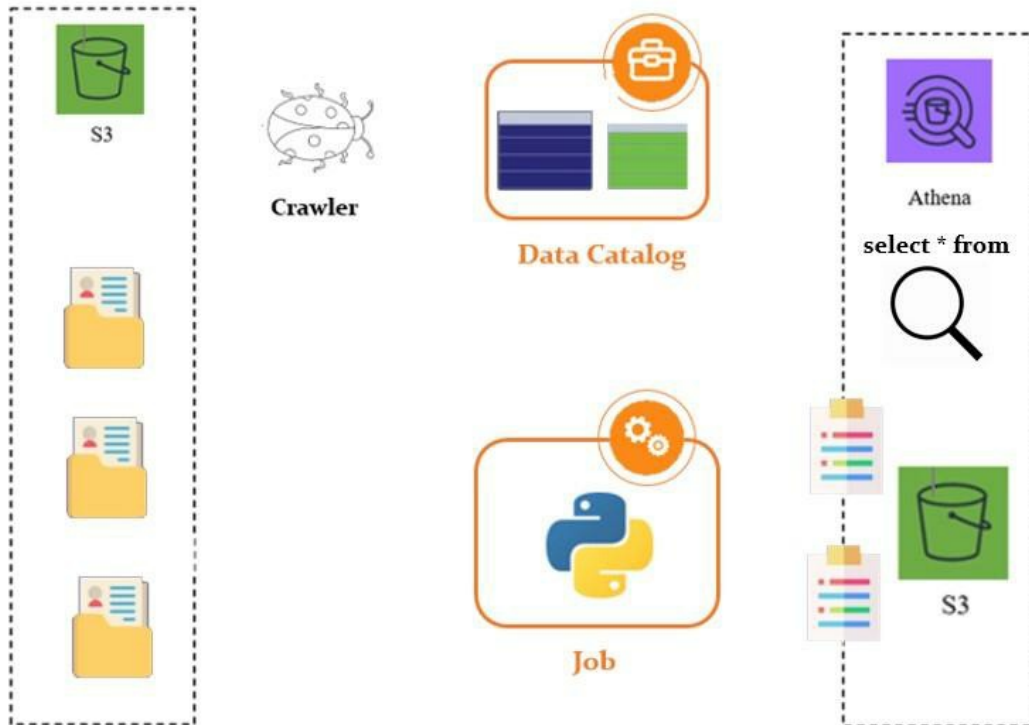


Figure 9-02: AWS Glue Components

Considering we have some data in S3, we can set up a crawler that crawls the data present in S3. It tries to find the important information, the metadata, the column names, the column types, and all of that important metadata information about the data stored in S3. Then, it creates tables within our Data Catalog. Once we have those tables within our Data Catalog, we can use Athena to query that data and use normal SQL querying syntax.

Consider we need to run some transformation job on that data. We need to drop some columns or possibly change some column names. We can set up a Glue job that runs either Python code or Scala code. In this case, we will run some Python code. When the job is started, AWS will spin up a group of servers that runs the principle that we have in our Python code, and then it will make the transformations that we specify. Once the job is complete, the servers are torn down by AWS, and we can have the output of our transformed data onto S3. We could set up another crawler to crawl that data to understand the metadata about the newly transformed data. It will create tables in our Data Catalog, and we can use Athena to query that new data or offload some other type of analytics to our transformed data.



*Figure 9-03: AWS Glue Components*

## **AWS Glue Data Catalog**

### ***President Metadata Store***

You can store, annotate, and share metadata between AWS services (similar to Apache Hive megastore).

### ***Centralized Repository***

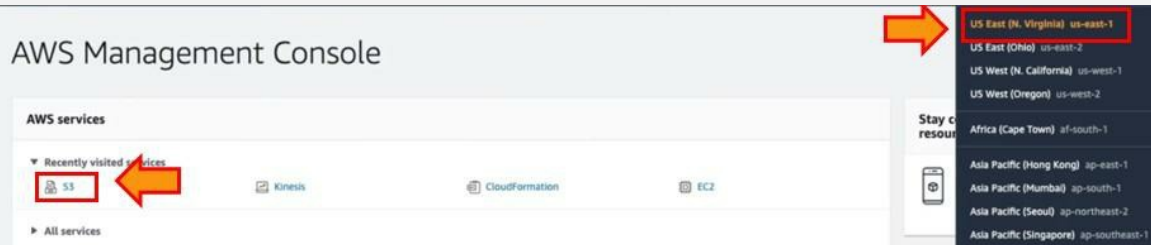
There is only one data catalog per AWS region, providing a uniform repository so different systems can store and find metadata to query and transform that data.

### ***Provided Comprehensive Audit***

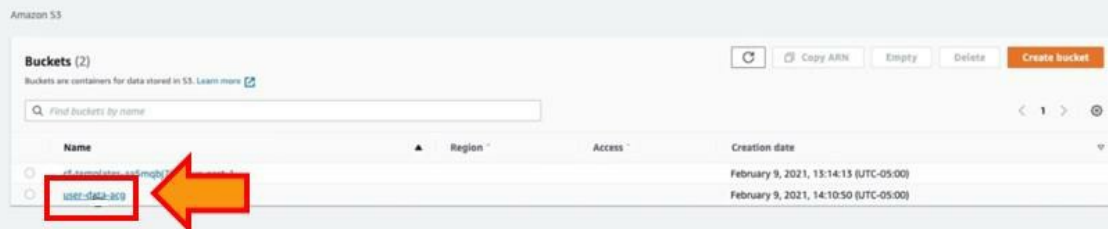
You can stack schema changes and data access control. This helps ensure that its data is not inappropriately modified or inadvertently shared.

# Demo 9-01: Populating the AWS Glue Data Catalog

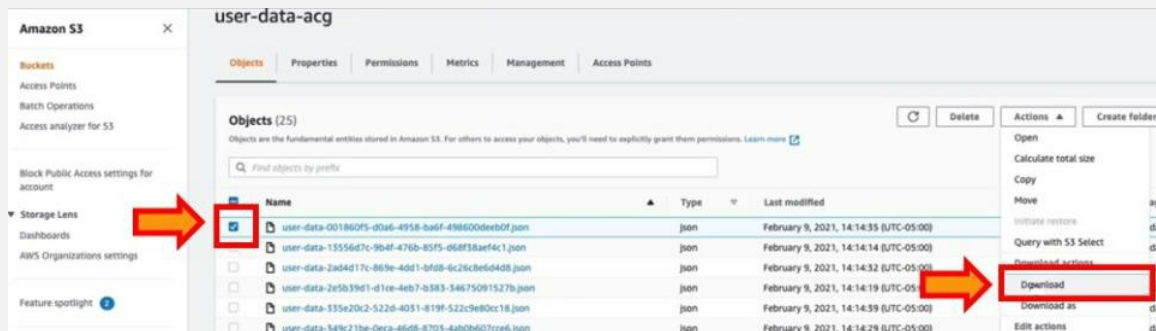
1. Login to AWS Management Console.



2. Click on the **Bucket**.



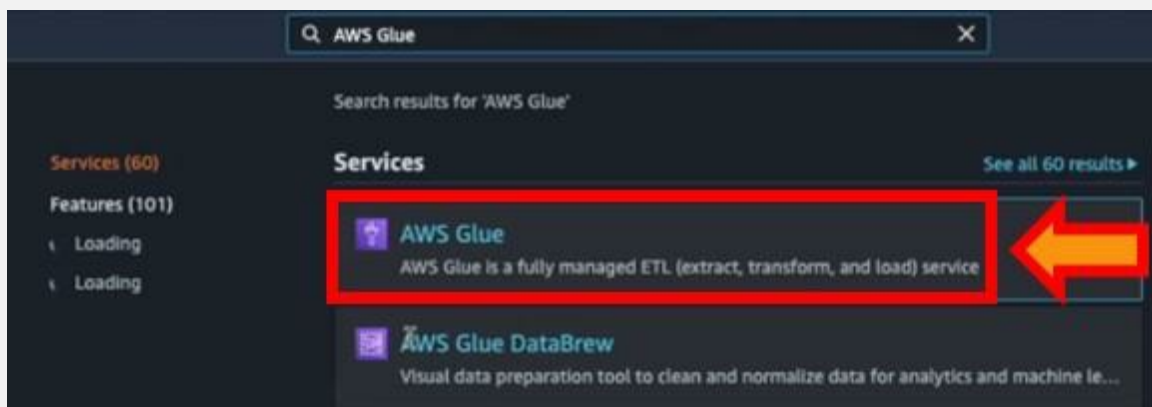
3. Select the object.
4. Click on **Download** under actions.



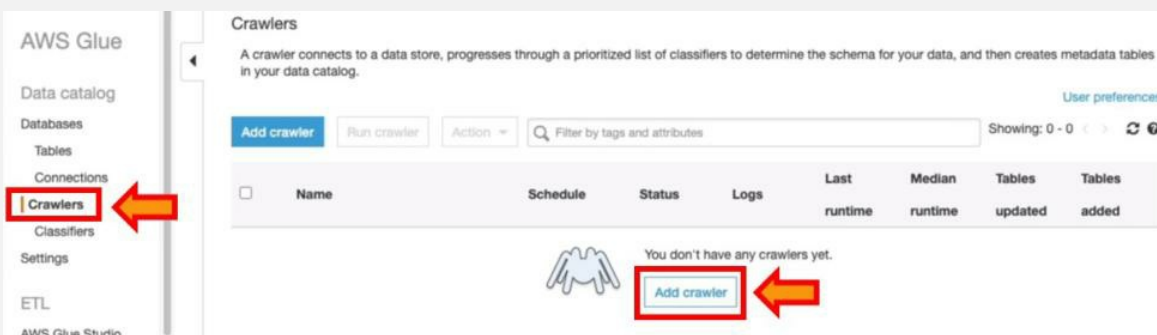
5. Open the downloaded file.

```
{ "picture": { "large": "https://randomuser.me/api/portraits/men/62.jpg", "medium": "https://randomuser.me/api/portraits/men/90.jpg", "small": "https://randomuser.me/api/portraits/men/84.jpg" }, "name": "Mr. John Doe" },
{ "picture": { "large": "https://randomuser.me/api/portraits/women/84.jpg", "medium": "https://randomuser.me/api/portraits/women/75.jpg", "small": "https://randomuser.me/api/portraits/women/50.jpg" }, "name": "Mrs. Jane Smith" },
{ "picture": { "large": "https://randomuser.me/api/portraits/women/50.jpg", "medium": "https://randomuser.me/api/portraits/women/39.jpg", "small": "https://randomuser.me/api/portraits/men/96.jpg" }, "name": "Mr. Robert Brown" },
{ "picture": { "large": "https://randomuser.me/api/portraits/women/39.jpg", "medium": "https://randomuser.me/api/portraits/men/96.jpg", "small": "https://randomuser.me/api/portraits/women/90.jpg" }, "name": "Mrs. Emily White" },
{ "picture": { "large": "https://randomuser.me/api/portraits/men/82.jpg", "medium": "https://randomuser.me/api/portraits/men/94.jpg", "small": "https://randomuser.me/api/portraits/men/8.jpg" }, "name": "Mr. David Green" },
{ "picture": { "large": "https://randomuser.me/api/portraits/men/94.jpg", "medium": "https://randomuser.me/api/portraits/men/12.jpg", "small": "https://randomuser.me/api/portraits/women/59.jpg" }, "name": "Mrs. Sarah Black" },
{ "picture": { "large": "https://randomuser.me/api/portraits/women/59.jpg", "medium": "https://randomuser.me/api/portraits/men/16.jpg", "small": "https://randomuser.me/api/portraits/women/75.jpg" }, "name": "Mr. Michael Blue" },
{ "picture": { "large": "https://randomuser.me/api/portraits/men/16.jpg", "medium": "https://randomuser.me/api/portraits/women/75.jpg", "small": "https://randomuser.me/api/portraits/women/44.jpg" }, "name": "Mrs. Lisa Yellow" },
{ "picture": { "large": "https://randomuser.me/api/portraits/women/44.jpg", "medium": "https://randomuser.me/api/portraits/women/88.jpg", "small": "https://randomuser.me/api/portraits/men/5.jpg" }, "name": "Mr. James Purple" },
{ "picture": { "large": "https://randomuser.me/api/portraits/men/5.jpg", "medium": "https://randomuser.me/api/portraits/men/75.jpg", "small": "https://randomuser.me/api/portraits/women/75.jpg" }, "name": "Mrs. Anna Pink" }
```

6. Search AWS Glue in the search bar.
7. Click on AWS Glue.



8. Click on Crawlers.
9. Click on Add Crawler.



10. Define Crawler name.
11. Click on Next.

### Add crawler

- ☒ Crawler info
- ☐ Crawler source type
- ☐ Data store
- ☐ IAM Role
- ☐ Schedule
- ☐ Output
- ☐ Review all steps

#### Add information about your crawler

Crawler name

user-data-crawler

► Tags, description, security configuration, and classifiers (optional)

Next

12. Specify crawler source type.
13. Click on **Next**.

### Specify crawler source type

Choose Existing catalog tables to specify catalog tables as the crawler source. The selected tables specify the data stores to crawl. This option doesn't support JDBC data stores.

**Crawler source type**

☒ Data stores

☐ Existing catalog tables

Back

Next

14. Include a path.
15. Choose the S3 path.
16. Click on **Select**.

## Choose S3 path



17. Click on **Next**.

[Add connection](#)

**Crawl data in**

☒ Specified path in my account  
☐ Specified path in another account


**Include path**

All folders and files contained in the include path are crawled. For example, type `s3://MyBucket/MyFolder/` to crawl all objects in MyFolder within MyBucket.

▼ **Exclude patterns (optional)**

**Exclude patterns**

The exclude pattern is relative to the include path. Objects that match the exclude pattern are not crawled. For example, with include path `s3://mybucket/` and exclude pattern, `mydir/**`, then all objects in the include path below the `mydir` directory are skipped. In this example, any object whose path matches `s3://mybucket/mydir/**` is not crawled. For more information about patterns, see [Cataloging Tables with a Crawler](#).

 [Next](#)

18. Click on **Next**.

**Add another data store**

☐ Yes  
☒ No

 [Next](#)

19. Define **IAM role**.

20. Click on **Next**.

## Choose an IAM role

The IAM role allows the crawler to run and access your Amazon S3 data stores. [Learn more](#)

☐ Update a policy in an IAM role  
☐ Choose an existing IAM role  
☒ Create an IAM role

**IAM role** ⓘ


AWSGlueServiceRole-  

To create an IAM role, you must have **CreateRole**, **CreatePolicy**, and **AttachRolePolicy** permissions.

Create an IAM role named "AWSGlueServiceRole-rolename" and attach the AWS managed policy, **AWSGlueServiceRole**, plus an inline policy that allows read access to:

- s3://user-data-acg


You can also create an IAM role on the [IAM console](#). ☐




21. Define **frequency**.
22. Click on **Next**.

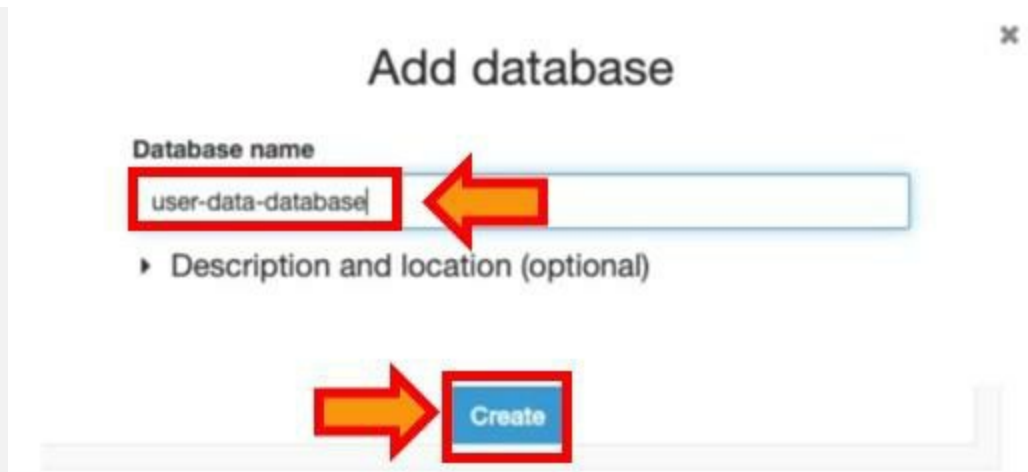
## Create a schedule for this crawler

**Frequency**





23. Define **Database name**.
24. Click on **Create**.



## Add database

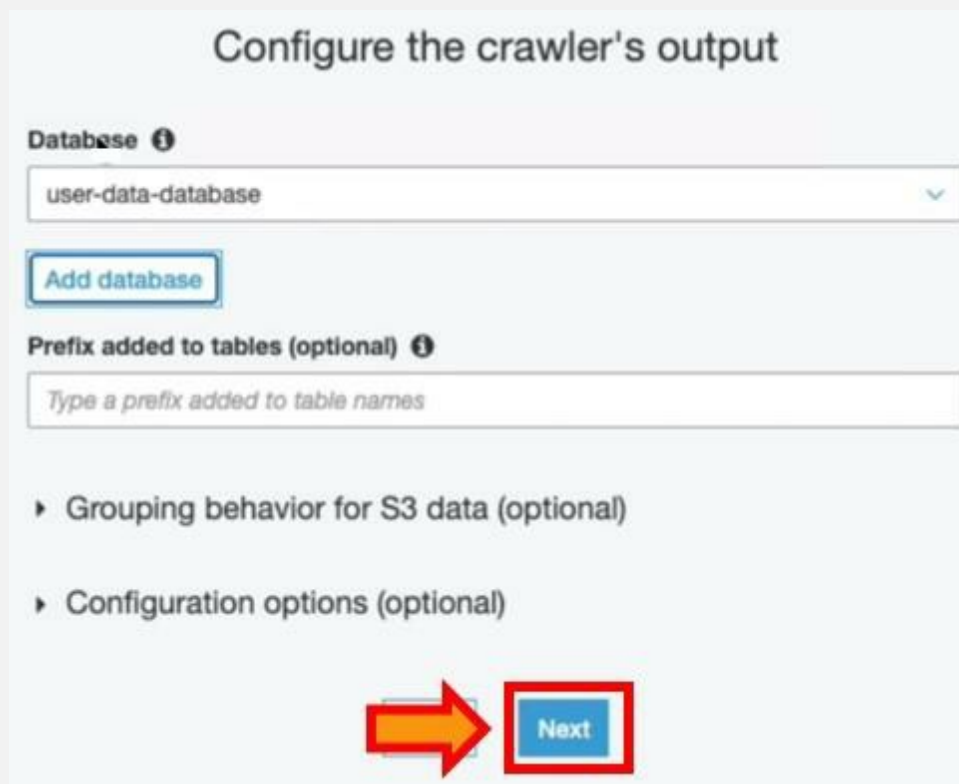
Database name

user-data-database

► Description and location (optional)

Create

25. Click on **Next**.



## Configure the crawler's output

Database ⓘ

user-data-database

Add database

Prefix added to tables (optional) ⓘ

Type a prefix added to table names

► Grouping behavior for S3 data (optional)

► Configuration options (optional)

Next

26. Click on **Finish**.

**IAM role**

**IAM role** `arn:aws:iam::839712734073:role/service-role/AWSGlueServiceRole-Master`

---

**Schedule**

**Schedule** `Run on demand`

---



**Output**

**Database** `user-data-database`

**Prefix added to tables (optional)**

**Create a single schema for each S3 path** `false`

► **Configuration options**





27. Select **Crawler**.
28. Click on **Run Crawler**.

**Crawlers**


A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

[User preferences](#)

**Add crawler** **Run crawler** 

Filter by tags and attributes Showing: 1 - 1


| <input checked="" type="checkbox"/> | Name              | Schedule | Status | Logs | Last runtime | Median runtime | Tables updated | Tables added |
|-------------------------------------|-------------------|----------|--------|------|--------------|----------------|----------------|--------------|
| <input checked="" type="checkbox"/> | user-data-crawler |          | Ready  |      | 0 secs       | 0 secs         | 0              | 0            |



29. Click on **Tables**.
30. Select **Crawler**.

**AWS Glue**

**Data catalog**

**Databases** 

**Tables**

**Connections**


**Crawlers**

**Classifiers**

**Settings**

**Crawlers**

A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

**Add crawler** **Run crawler** **Action** 

Filter by tags and attributes Showing: 1

| <input checked="" type="checkbox"/> | Name              | Schedule | Status   | Logs | Last runtime | Median runtime | Tables updated |
|-------------------------------------|-------------------|----------|----------|------|--------------|----------------|----------------|
| <input checked="" type="checkbox"/> | user-data-crawler |          | Stopping |      | 1 min        | 1 min          | 0              |

31. Click on **Table**.

**Tables** A table is the metadata definition that represents your data, including its schema. A table can be used as a s

[Add tables](#) [Action](#)  [Save view](#)

| <input type="checkbox"/> Name                 | Database           | Location            | Classifi |
|-----------------------------------------------|--------------------|---------------------|----------|
| <input type="checkbox"/> <b>user_data_acg</b> | user-data-database | s3://user-data-acg/ | json     |

32. See the details about the table.

[Edit table](#) [Delete table](#) [View properties](#) [Compare versions](#) [Edit schema](#)

**Name** user\_data\_acg

**Description**

**Database** user-data-database

**Classification** json

**Location** s3://user-data-acg/

**Connection**

**Deprecated** No

**Last updated** Thu Oct 15 11:21:46 GMT-400 2020

**Input format** org.apache.hadoop.mapred.TextInputFormat

**Output format** org.apache.hadoop.hive.q1.io.HiveIgnoreKeyTextOutputFormat

**Serde serialization lib** org.openx.data.jsonserde.JsonSerDe

**Serde parameters**

paths cell,dob,email,gender,id,location,name,nat,phone,picture,registered

sizeKey 23662873 objectCount 10 UPDATED\_BY\_CRAWLER user-data-crawler

**Table properties**

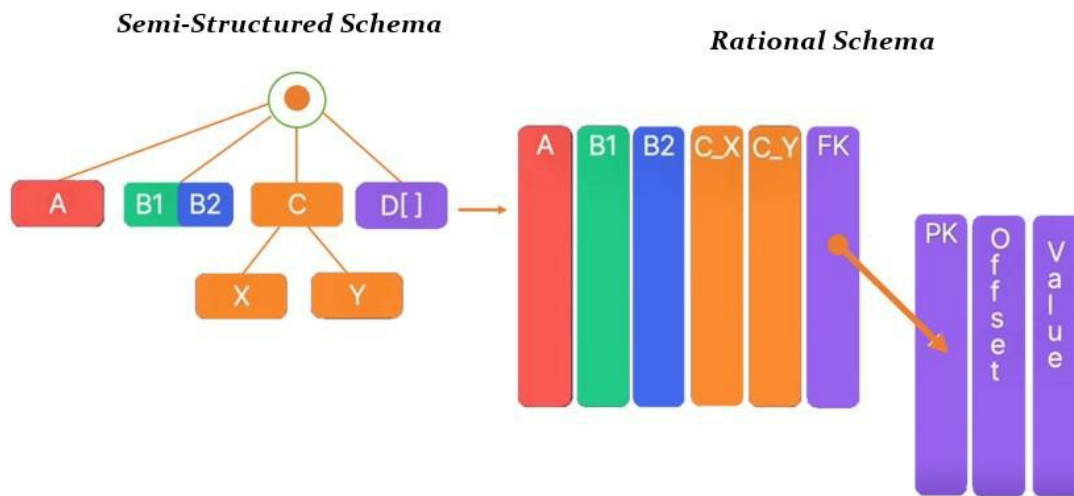
CrawlerSchemaSerializerVersion 1.0 recordCount 22845 averageRecordSize 1033

CrawlerSchemaDeserializerVersion 1.0 compressionType none typeOfData file

## Converting Semi-Structured Schemas to Rational Schemas

Consider we have semi-structured data. AWS will convert the data into a relational schema. The single value A converts directly to a relational column. The pair of values B<sub>1</sub> and B<sub>2</sub> convert to two different relational columns. The structure C that has children X and Y converts to two relational columns. The array D converts to a relational column with a foreign key that points to another relational table, along with a primary key. The second relational table has columns that contain offsets and values for the items within the array. We will have a foreign key that points to a primary key, and the other columns within that table contain an offset and the values for the items in the array. It allows you

to create that metadata catalog that you can query and start running ETL jobs on.



*Figure 9-04: Converting Semi-Structured Schemas to Rational Schemas*

#### EXAM TIP:

- **What is AWS?** – Fully managed ETL service to categorize, clean, and enrich your data.
- **AWS Glue - Use cases** – Query Data in S3, join data, and create a centralized metadata catalog.
- **AWS Glue Components** – Source data stores, crawlers, catalog, jobs, output data store, or services using the data catalog.
- **AWS Glue Data Catalog** – persistent Metabase store that is a centralized repository.
- **Demo** – Created a crawler that populated a table in our Glue Data Catalog.
- **Schema Conversions** – Glue flattens a hierarchical schema to a logical schema.

## Glue Jobs

### AWS Glue Jobs

AWS Glue job performs the extract, transform, and load (ETL) work in AWS Glue. You have some input data; consider the input data as the fabric. The Glue job is where the actual work is being done; this is the cleaning of your data, the transformation of your data, the enrichment

of your data, and the joining of your data. This will be the actual physical labor and the tools that you use to create the final output. Output data onto another data store or into some data lake or data warehousing solution.

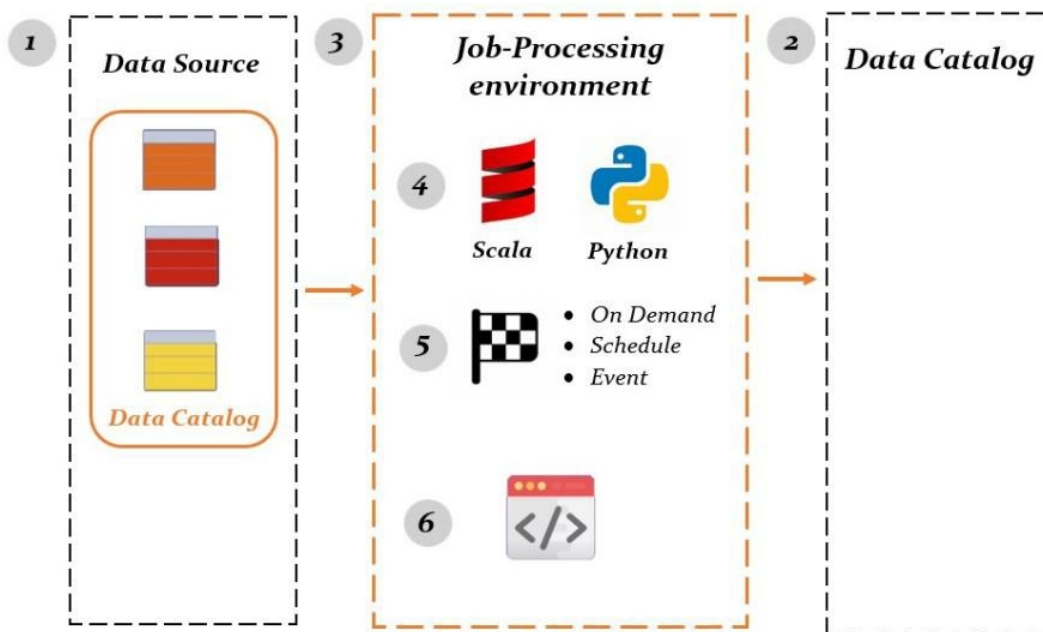


Figure 9-05: AWS Glue Jobs

## Workflow Overview

The main part of the job is AWS's actual processing environment. The data source can come from many different locations, but it has to be from the data. First, we have to crawl data, whether that is S3, DynamoDB, or JDBC connection, and create the tables within our data catalog. Then we have to have a data target, set up a scaler or a Python script, and then tell it to run on a particular interval, whether on-demand or some schedule. For example, every Monday of every week or the first Friday of every month. We can have the job run after some event occurs, whether that is data landing onto S3 or some Lambda function that triggers this event. Then, we will need to define the actual code that will perform the transformation or the data loading and processing that needs to be done where and when our data comes from our data source and lands in our data target. AWS Glue jobs can author

the scripts from scratch by ourselves or use the generated PI Spark or Scala script that AWS Glue creates for us. We can take this script and tailor it to our business needs.



*Figure 9-06: Workflow Overview*

## Lab 9-01: AWS Glue Jobs

### Introduction

#### AWS Glue

Amazon Glue is a serverless data integration service that simplifies data identification, preparation, and integration for analytics, machine learning, and application development. Amazon Glue has all the tools required for data integration, allowing you to begin analyzing and utilizing your data in minutes rather than months. AWS Glue offers both visual and code-based interfaces to help with data integration. The AMAZON Glue Data Catalog allows users to discover and retrieve data easily. ETL workflows may be created and executed by data engineers

and ETL (extract, transform, and load) developers. AMAZON Glue DataBrew allows data analysts and data scientists to enhance, clean visually, and standardize data without writing code.

### **AWS Simple Storage Service (S3)**

Amazon S3 is a type of object storage that allows you to store and recover any quantity of data from any location. It is a low-cost storage solution with business resilience, reliability, efficiency, privacy, and infinite expansion.

Amazon S3 is a web service that allows you to store and retrieve an infinite quantity of data from any place and at any time. You may quickly create projects that integrate cloud-native storage using this service. Because Amazon S3 is easily customizable and you only pay for what you use, you can start small and scale up as needed without sacrificing performance or dependability.

Amazon S3 is also built to be highly adaptable. Instead of finding out how to store their data, Amazon S3 allows developers to focus on innovation. You can build a simple FTP system or a complex web application like the Amazon.com retail website, read the same piece of data a million times or only for emergency disaster recovery, as well as store whatever type and amount of data you desire.

### **Problem**

Assume you are a Data Analytics in an organization. The organization you work in develops online games. The organization gives you a task to integrate, transform, and store real-time data coming from an online game that produces hourly GBs of user play data. Hence, how can you automate this task?

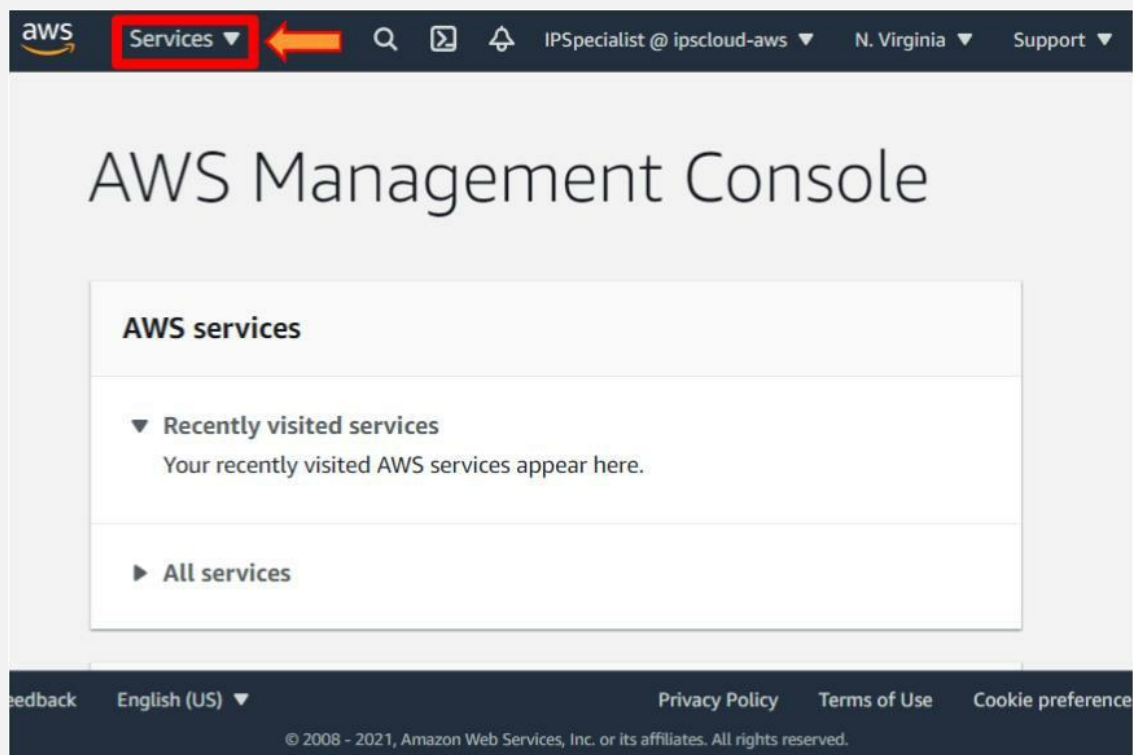
## Solution

The solution is to use AWS Glue Jobs for data integration operations, such as extraction, cleaning, normalizing, combining, loading, and performing scalable ETL procedures. It reduces the time it takes to examine and use your data from months to minutes. For storing the real-time data, you use the S3 bucket.

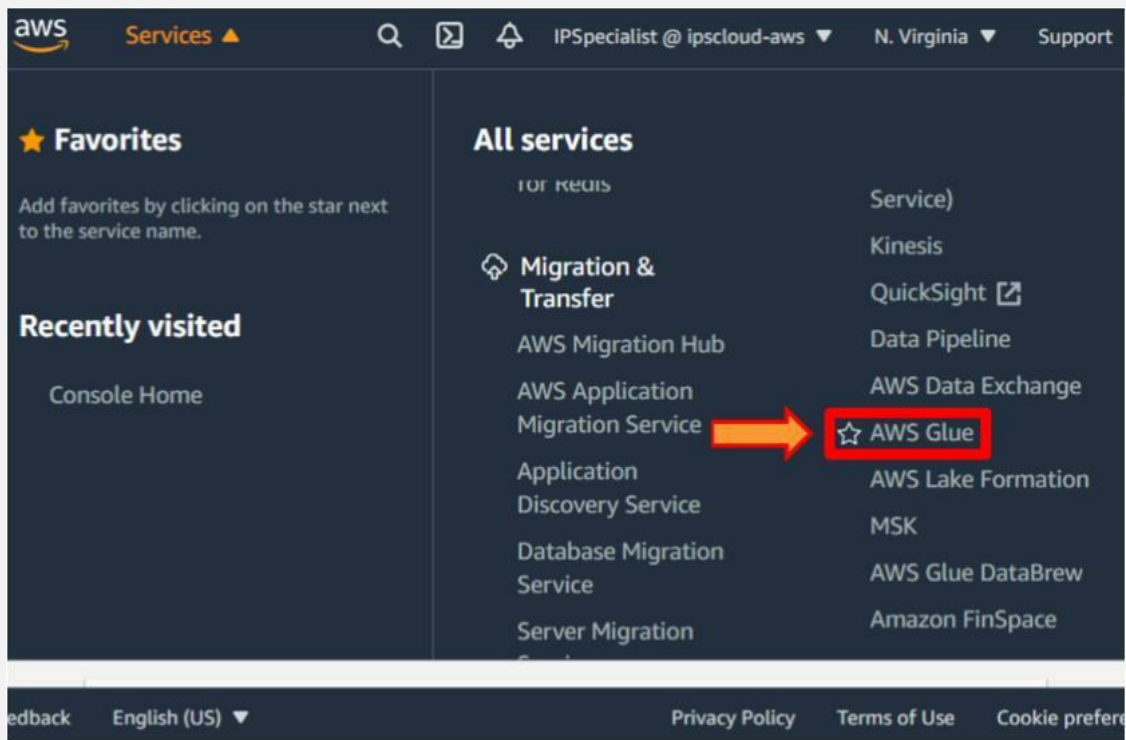
**Note:** Before starting the demo, create an s3 bucket and AWS Glue crawlers used in this demo. How to create AWS crawlers is mentioned in the above demo of AWS Glue Catalog.

### Step 1: Create AWS Glue Job

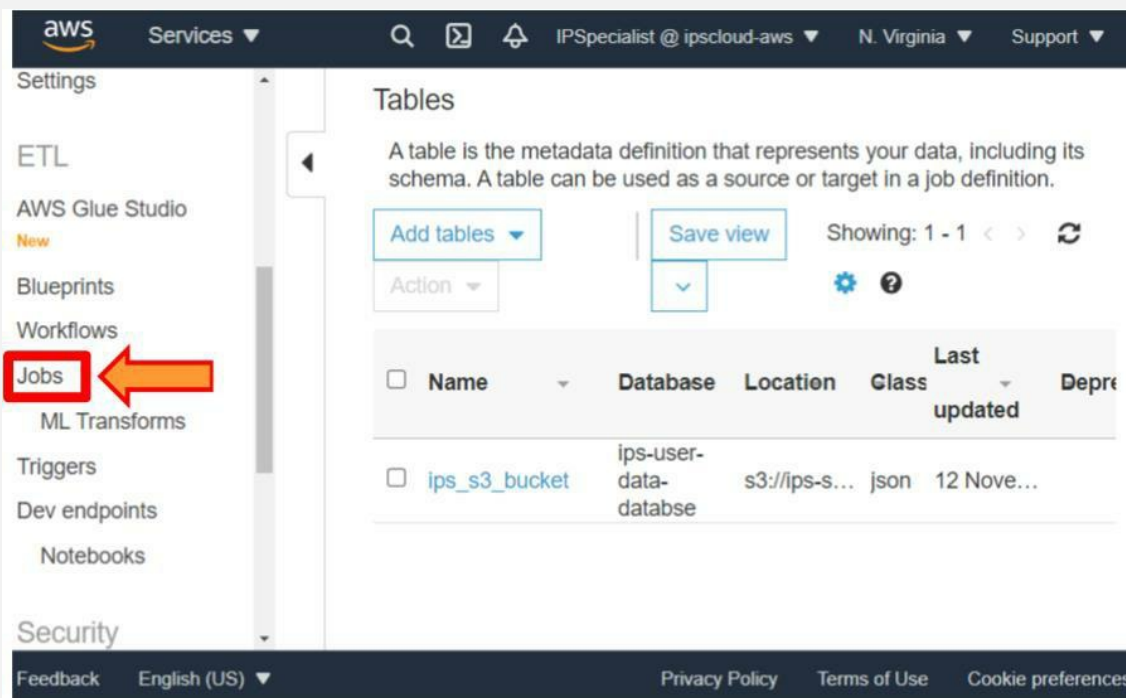
1. Log in to the **AWS Console**.
2. Click on the **Services**.



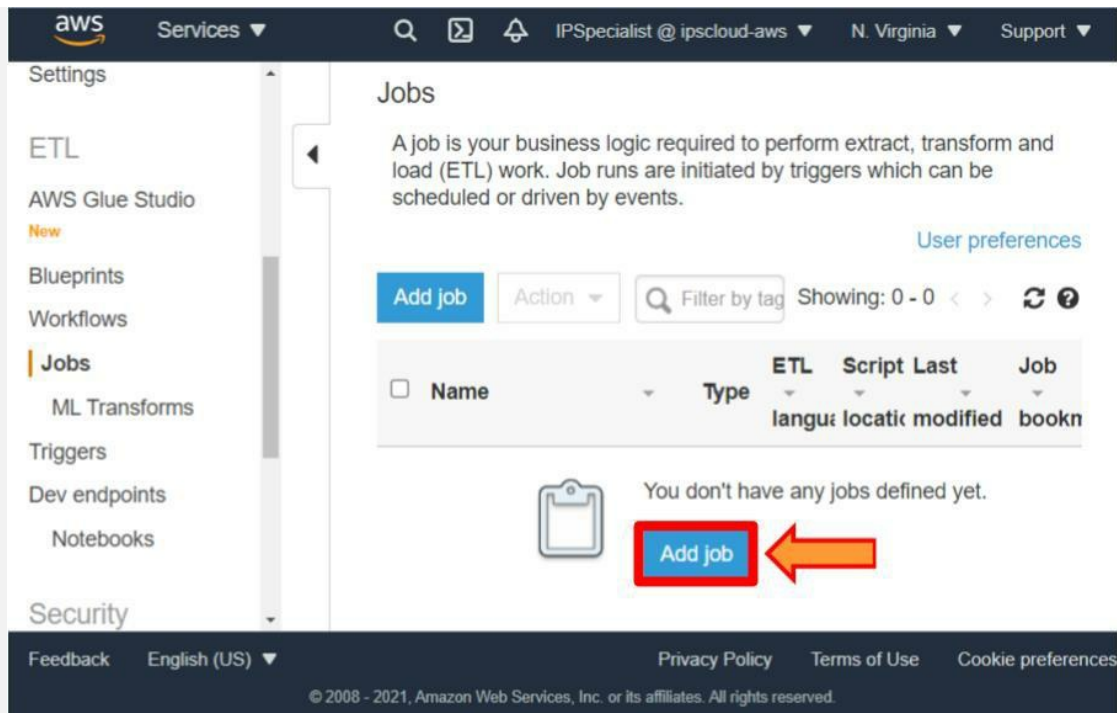
3. Select the **AWS Glue** from the **Analytics**.



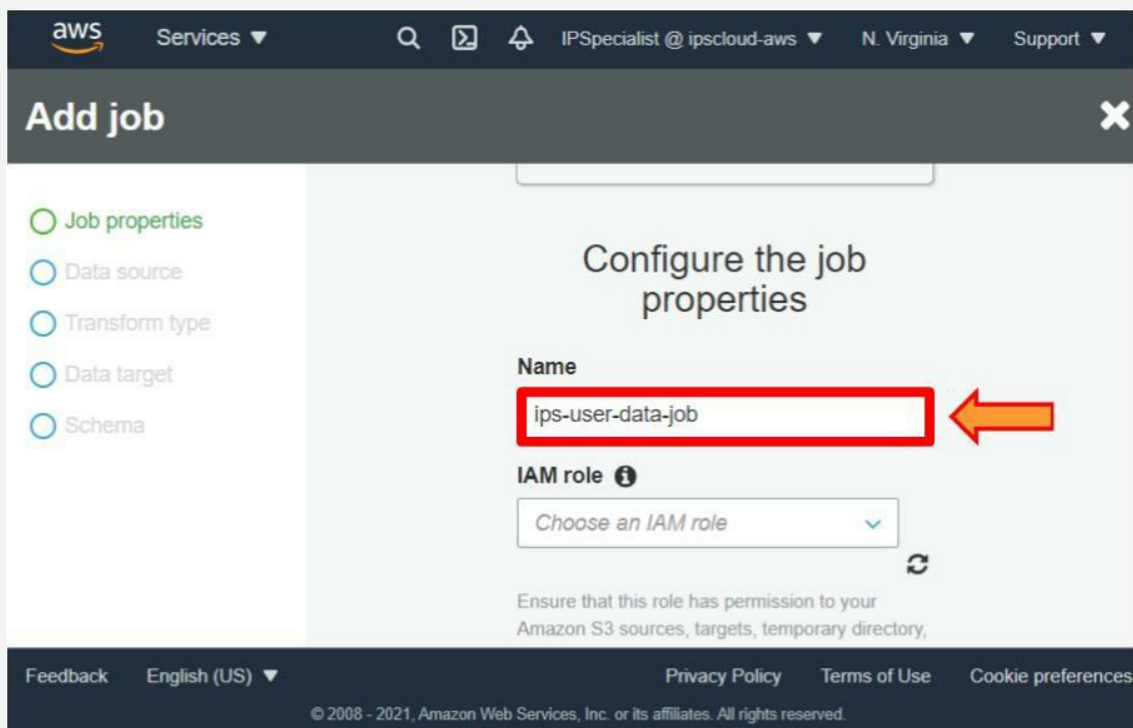
4. Click on the **Jobs** from the left-hand side menu.



5. Click on the **Add Job** button.



6. Give the job name **ips-user-data-job**.



7. Select the **IAM role** that you previously created in the crawlers demo.

aws Services ▾

IPSpecialist @ ipscloud-aws ▾ N. Virginia ▾ Support ▾

## Add job

Configure the job properties

**Name**

ips-user-data-job

**IAM role** ⓘ

**AWSGlueServiceRole-IPS-MasterRole** ▾ ↻

Ensure that this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job. [Create IAM role.](#)

**Type**

Spark ▾

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

8. Select the **Spark** type.

aws Services ▾

IPSpecialist @ ipscloud-aws ▾ N. Virginia ▾ Support ▾

## Add job

Job properties

Data source

Transform type

Data target

Schema

Ensure that this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job. [Create IAM role.](#)

**Type**

**Spark** ▾ ↻

**Glue version**

Spark 2.4, Python 3 (Glue Versio... ▾

**This job runs**

☒ A proposed script generated by AWS Glue

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

9. Select the **Glue Version 2.0.**

The screenshot shows the AWS Glue 'Add job' console. On the left, there is a sidebar with links: Job properties (selected), Data source, Transform type, Data target, and Schema. The main content area has the following fields:

- Type:** A dropdown menu with 'Spark' selected.
- Glue version:** A dropdown menu with 'Spark 2.4, Python 3 (Glue Version 2.0)' selected. An orange arrow points to this dropdown.
- This job runs:** Three radio button options:
  - ☒ A proposed script generated by AWS Glue ⓘ
  - ☐ An existing script that you provide
  - ☐ A new script to be authored by you
- Script file name:** A text input field containing 'ips-user-data-job'.

The footer contains links for Feedback, English (US), Privacy Policy, Terms of Use, and Cookie preferences, along with a copyright notice: © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

10. Select the **A proposed script generated by AWS Glue**.

This screenshot is similar to the previous one, but it focuses on the 'This job runs' section. The 'Glue version' dropdown remains 'Spark 2.4, Python 3 (Glue Version 2.0)'. In the 'This job runs' section, the radio button for 'A proposed script generated by AWS Glue ⓘ' is now selected and highlighted with a red box. An orange arrow points to this radio button. The 'Script file name' field still contains 'ips-user-data-job'. The 'S3 path where the script is stored' field is now visible and contains 's3://aws-glue-scripts-644738277497-us-ea:'. The footer is identical to the previous screenshot.


11. Click on the **Monitoring options**.

aws Services ▾

Q ⓘ 🔔 IPSpecialist @ ipscloud-aws ▾ N. Virginia ▾ Support ▾

## Add job ✕

- ☒ Job properties
- ☐ Data source
- ☐ Transform type
- ☐ Data target
- ☐ Schema

▸ Monitoring options 

▸ Tags (optional)

▸ Security configuration, script libraries, and job parameters (optional)

▸ Catalog options (optional)

Next

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

12. Select the **Job Metrics** and **Continuous logging**.

aws Services ▾


Q ⓘ 🔔 IPSpecialist @ ipscloud-aws ▾ N. Virginia ▾ Support ▾

## Add job ✕

- ☒ Job properties
- ☐ Data source
- ☐ Transform type
- ☐ Data target
- ☐ Schema

▼ Monitoring options

☒ Job metrics ⓘ

☒ Continuous logging 

Log filtering ⓘ

☒ Standard filter ☐ No filter

☐ Spark UI ⓘ

▸ Tags (optional)

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

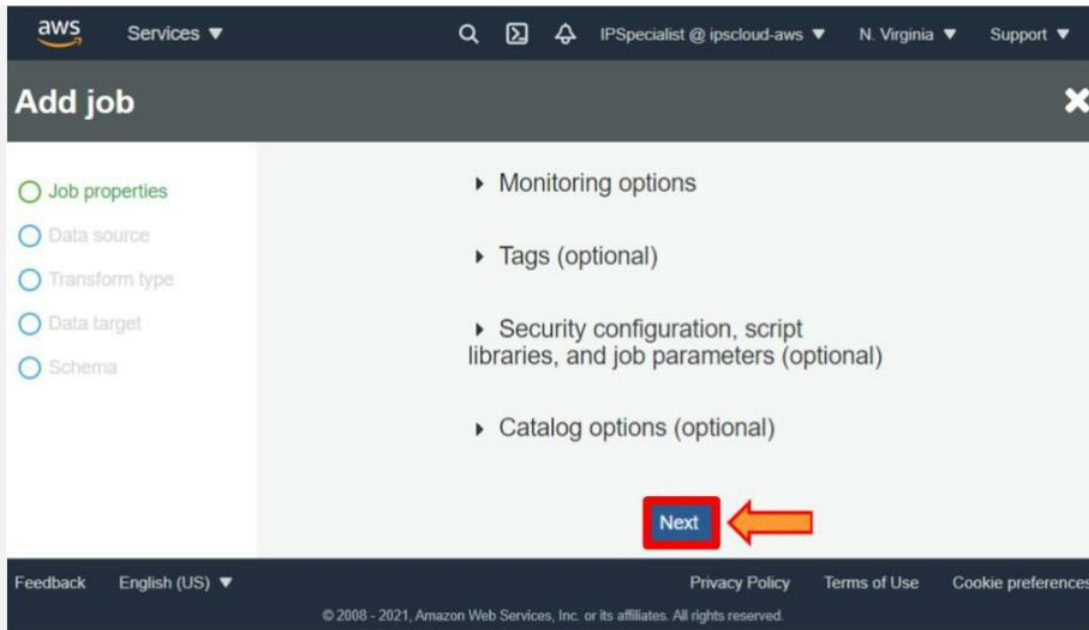
13. Click on the **Security Configuration**.

The screenshot shows the AWS Glue 'Add job' configuration page. On the left, there is a sidebar with a list of configuration sections: 'Job properties' (selected), 'Data source', 'Transform type', 'Data target', and 'Schema'. The main content area displays several expandable sections: 'Monitoring options', 'Tags (optional)', 'Security configuration, script libraries, and job parameters (optional)' (highlighted with a red box and an orange arrow), and 'Catalog options (optional)'. A blue 'Next' button is located at the bottom right of the main content area. The footer contains links for 'Feedback', 'English (US)', 'Privacy Policy', 'Terms of Use', and 'Cookie preferences', along with a copyright notice for Amazon Web Services, Inc.

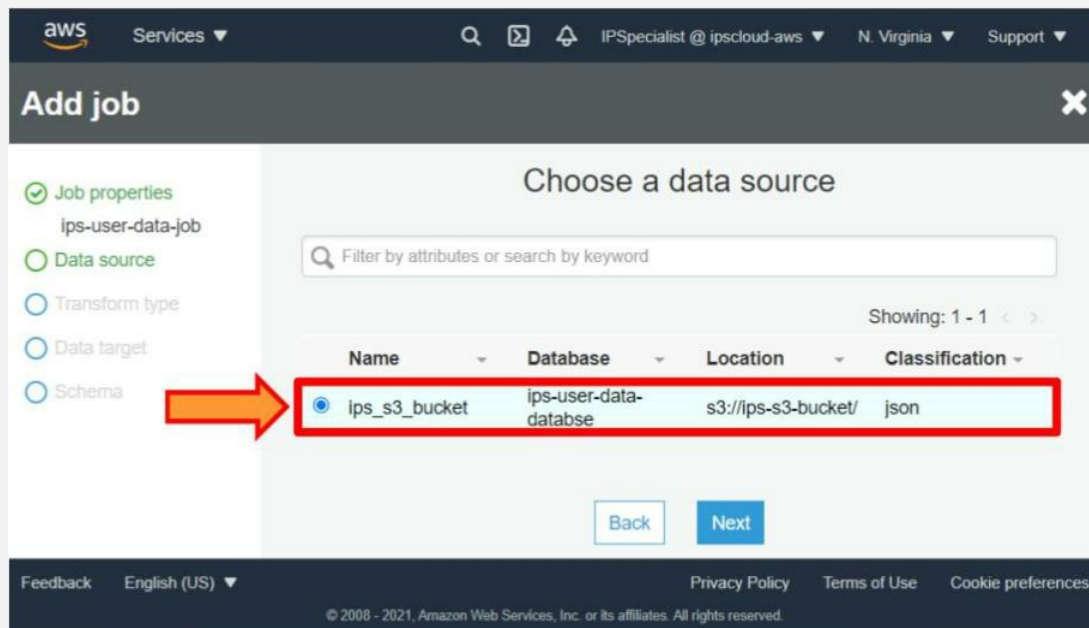
14. Scroll down. Select the **G.1x** worker type.

The screenshot shows the AWS Glue 'Add job' configuration page, scrolled down to the 'Worker type' section. The 'Referenced files path' field is set to 's3://bucket/prefix/object'. The 'Worker type' dropdown menu is highlighted with a red box and an orange arrow, showing 'G.1X (Recommended for memory-intensi...'. Below this, the 'Number of workers' is set to '10', with a note stating 'The maximum number of workers you can define are 299 for G.1X, and 149 for G.2X.'. The 'Max concurrency' section is partially visible at the bottom. The footer contains links for 'Feedback', 'English (US)', 'Privacy Policy', 'Terms of Use', and 'Cookie preferences', along with a copyright notice for Amazon Web Services, Inc.

15. Click on the **Next** button.



16. Select the **ips-s3-bucket**.



17. Click on the **Next** button.

aws Services ▾

IPSpecialist @ ipscloud-aws ▾ N. Virginia ▾ Support ▾

## Add job ✕

✓ Job properties  
ips-user-data-job

○ Data source

○ Transform type

○ Data target

○ Schema

### Choose a data source

Filter by attributes or search by keyword

Showing: 1 - 1 < >

| Name ▾                                         | Database ▾             | Location ▾          | Classification ▾ |
|------------------------------------------------|------------------------|---------------------|------------------|
| <input checked="" type="radio"/> ips_s3_bucket | ips-user-data-database | s3://ips-s3-bucket/ | json             |

Back

Next

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

18. Select the **Change Schema**.

aws Services ▾

IPSpecialist @ ipscloud-aws ▾ N. Virginia ▾ Support ▾

## Add job ✕

✓ Job properties  
ips-user-data-job

✓ Data source  
ips\_s3\_bucket

○ Transform type

○ Data target

○ Schema

### Choose a transform type

☒ **Change schema**  
Change schema of your source data and create a new target dataset

☐ Find matching records  
Use machine learning to find matching records within your source data

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

19. Click on the **Next** button.

aws Services ▾

IPSpecialist @ ipscloud-aws ▾ N. Virginia ▾ Support ▾

## Add job

- ✓ Job properties  
ips-user-data-job
- ✓ Data source  
ips\_s3\_bucket
- Transform type
- Data target
- Schema

☒ Change schema  
Change schema of your source data and create a new target dataset

☐ Find matching records  
Use machine learning to find matching records within your source data

Back Next

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

20. Select the **Create tables in your data**.

aws Services ▾

IPSpecialist @ ipscloud-aws ▾ N. Virginia ▾ Support ▾

## Add job

- ✓ Job properties  
ips-user-data-job
- ✓ Data source  
ips\_s3\_bucket
- ✓ Transform type  
Change schema
- Data target
- Schema

### Choose a data target

☒ Create tables in your data target

☐ Use tables in the data catalog and update your data target

Data store  
JDBC ▾

Connection  
- Select one - ▾

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

21. Select the **Amazon S3** data store.

aws Services ▾

IPSpecialist @ ipscloud-aws ▾ N. Virginia ▾ Support ▾

## Add job ✕

✓ Job properties  
ips-user-data-job

✓ Data source  
ips\_s3\_bucket

✓ Transform type  
Change schema

○ Data target

○ Schema

### Choose a data target

☒ Create tables in your data target  
☐ Use tables in the data catalog and update your data target

**Data store**  
Amazon S3

**Format**  
JSON

**Compression type**

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

## 22. Select the CSV format

aws Services ▾

IPSpecialist @ ipscloud-aws ▾ N. Virginia ▾ Support ▾

## Add job ✕

✓ Job properties  
ips-user-data-job

✓ Data source  
ips\_s3\_bucket

✓ Transform type  
Change schema

○ Data target

○ Schema

☒ Create tables in your data target  
☐ Use tables in the data catalog and update your data target

**Data store**  
Amazon S3

**Format**  
CSV

**Compression type**  
None

**Connection**

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

## 23. Select the gzip compression type.

aws Services IPSpecialist @ ipscloud-aws N. Virginia Support

### Add job

- Job properties
- ips-user-data-job
- Data source
- ips\_s3\_bucket
- Transform type
- Change schema
- Data target
- Schema

Data store: Amazon S3

Format: CSV

Compression type: **gzip**

Connection: - Select one -

Add connection

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

24. Click on the **Folder** button icon.

aws Services IPSpecialist @ ipscloud-aws N. Virginia Support

### Add job

- Job properties
- ips-user-data-job
- Data source
- ips\_s3\_bucket
- Transform type
- Change schema
- Data target
- Schema

Compression type: gzip

Connection: - Select one -

Add connection

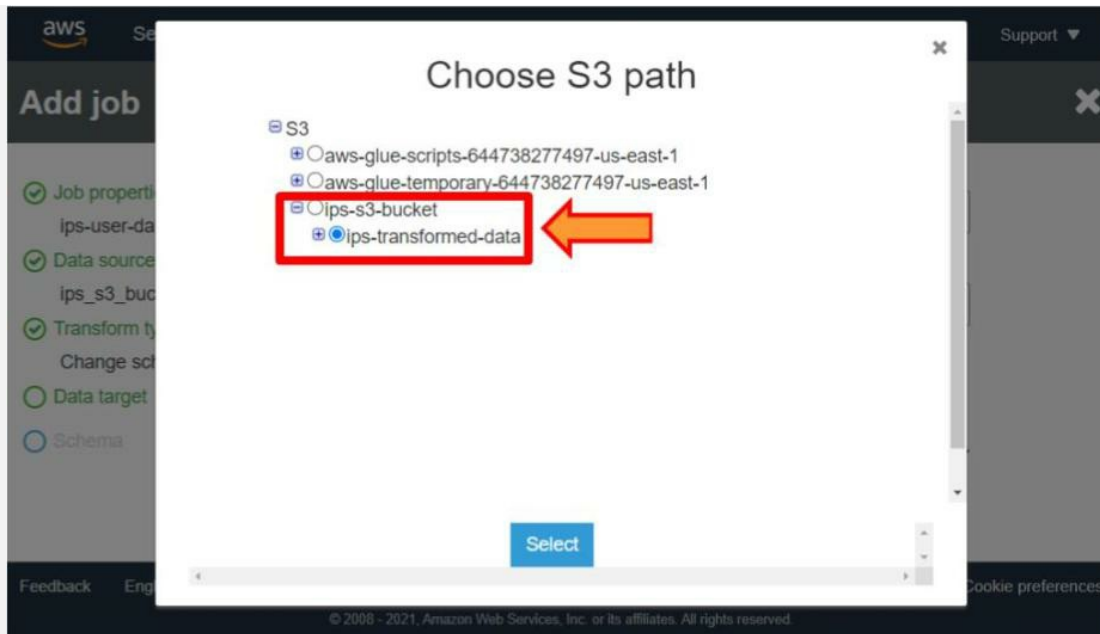
Target path: s3://bucket/prefix/object

Back Next

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

25. Select the **ips-transformed-data** folder in the **ips-s3-bucket**.



26. Click on the **Select** button.



27. Click on the **Next** button

aws Services IPSpecialist @ ipscloud-aws N. Virginia Support

## Add job

Job properties

ips-user-data-job

Data source

ips\_s3\_bucket

Transform type

Change schema

Data target

Schema

Compression type

gzip

Connection

- Select one -

Add connection

Target path

s3://ips-s3-bucket/ips-transformed-data

Back

Next

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

28. Delete the **Cell** and **Phone** data fields.

aws Services Search for services, features, blogs, docs, and r... [Alt+S] IPSpecialist @ ipscloud-aws N. Virginia Support

## Add job

Job properties

ips-user-data-job

Data source

ips\_s3\_bucket

Transform type

Change schema

Data target

s3://ips-s3-bucket/ip...

Schema

gender string gender

registerer struct -

date string 'registered.dat

age int 'registered.ag

id struct -

name string 'id.name'

value string 'id.value'

cell string cell

phone string phone

location struct -

'id.name' string x ↓ ↑

'id.value' string x ↓ ↑

cell string x ↓ ↑

phone string x ↓ ↑

'location.city' string x ↓ ↑

'location.coun string x ↓ ↑

'location.coor string x ↓ ↑

'location.coor string x ↓ ↑

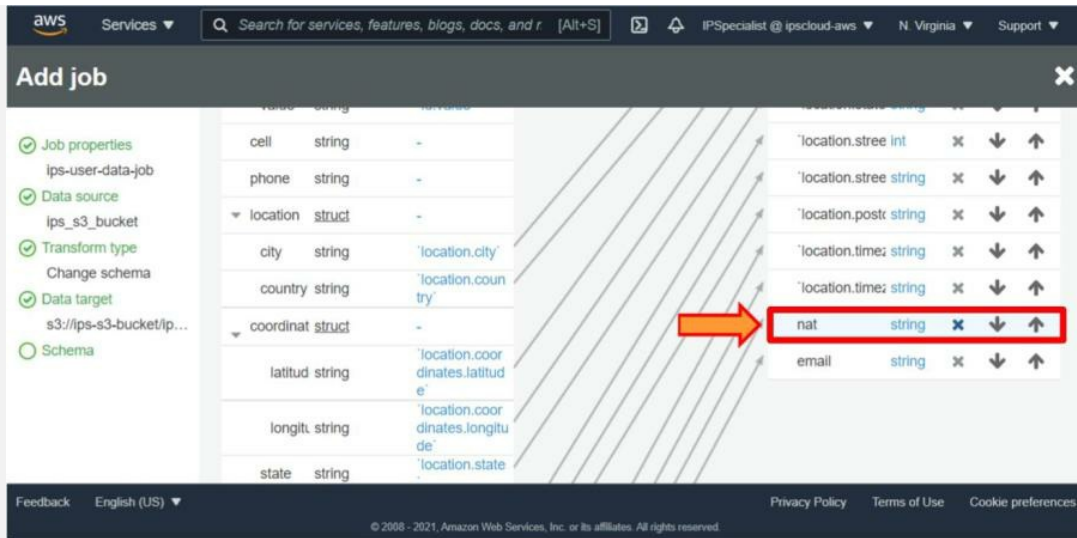
'location.state string x ↓ ↑

'location.stree int x ↓ ↑

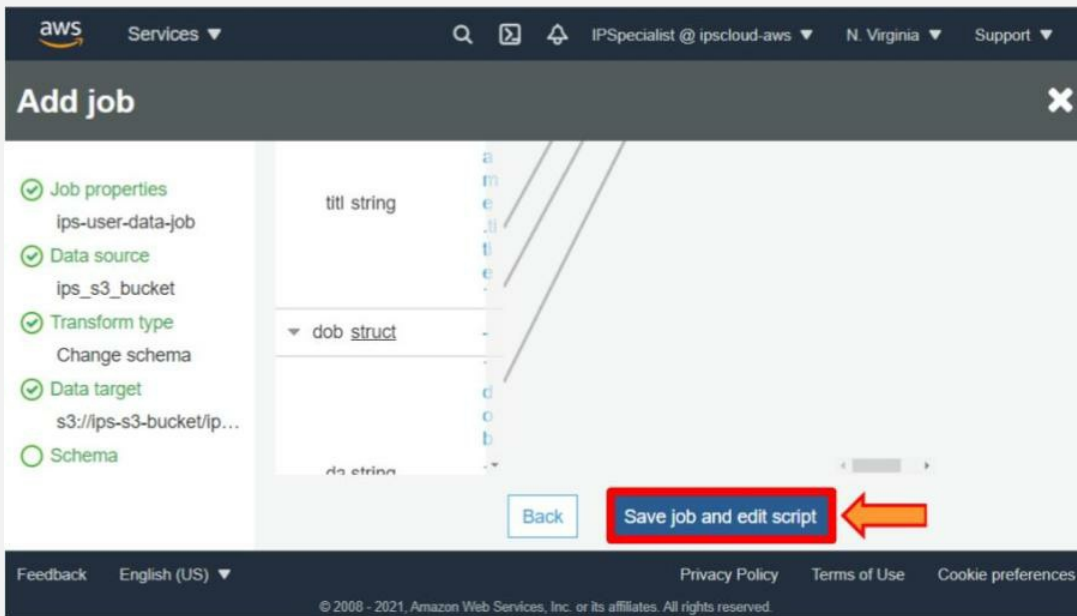
Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

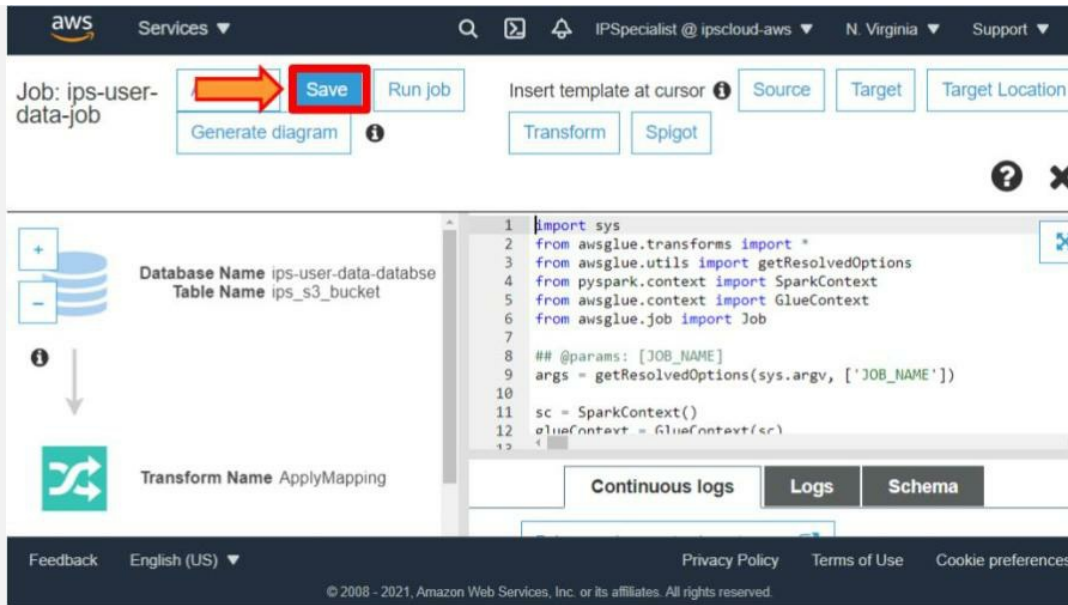
29. Delete the **Nat** data field.



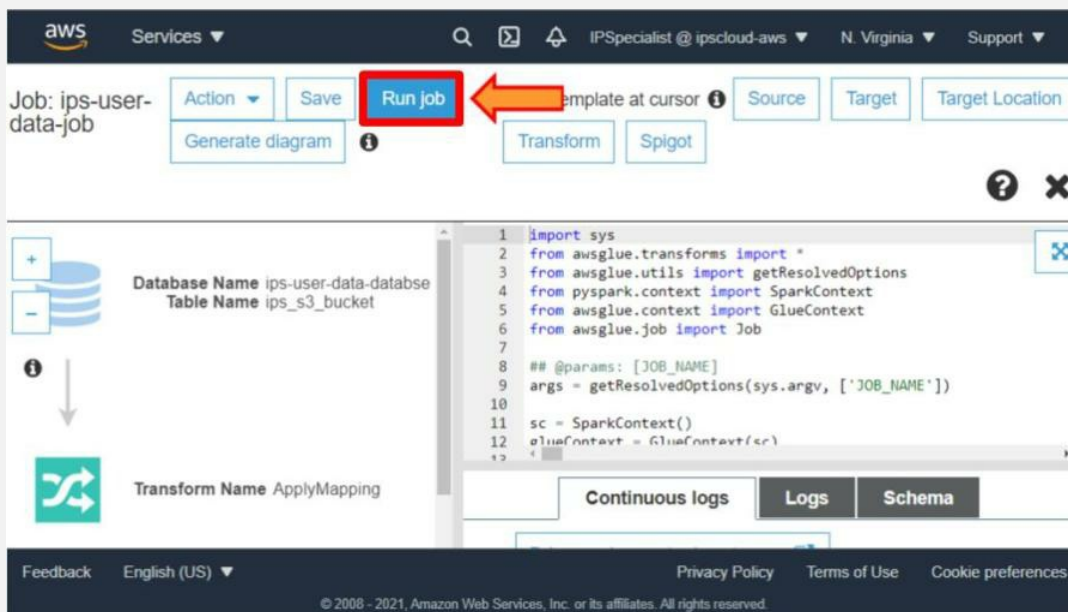
30. Click on the **Save job and edit script** button.



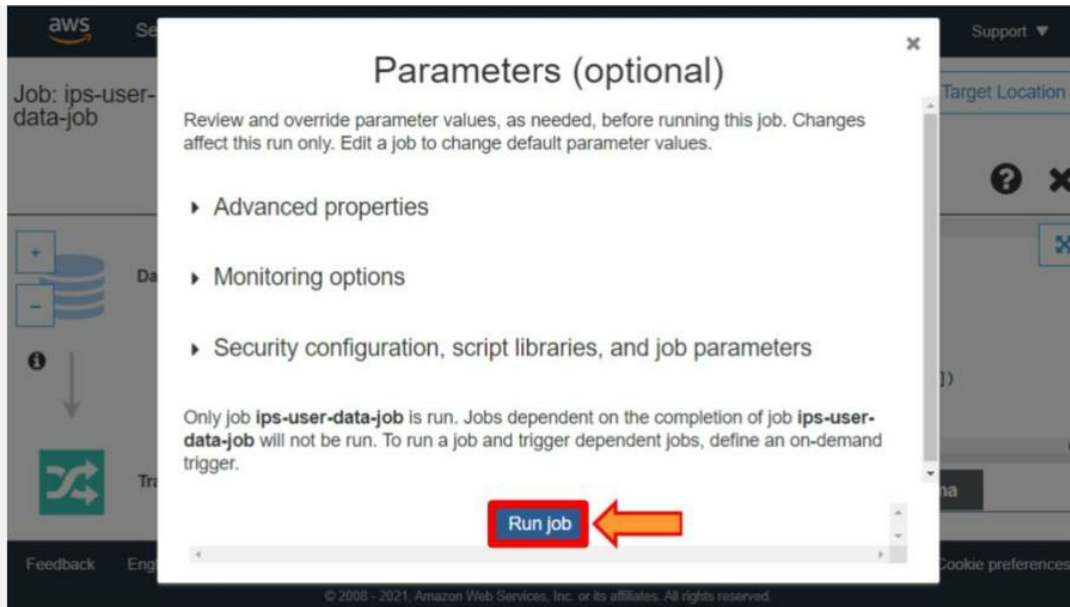
31. Click on the **Save** button.



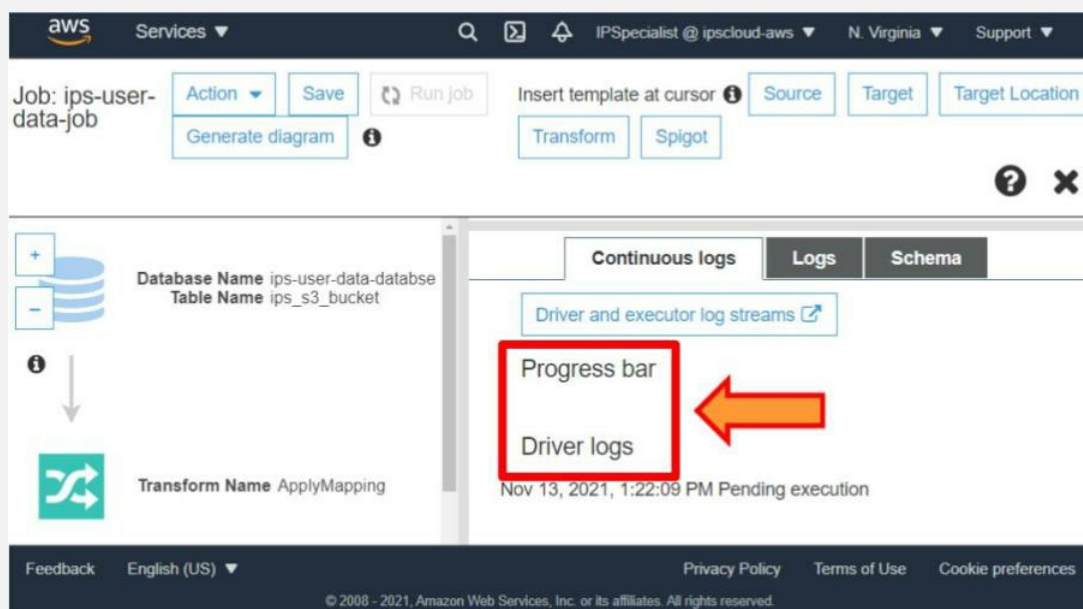
32. Click on the **Run Job** button.



33. Again, Click on the **Run Job** button.



34. It will take a few minutes as you see the **Progress bar** and **Driver logs**.



35. The **Progress bar** finishes. This means that successfully ran and created the AWS Glue job.

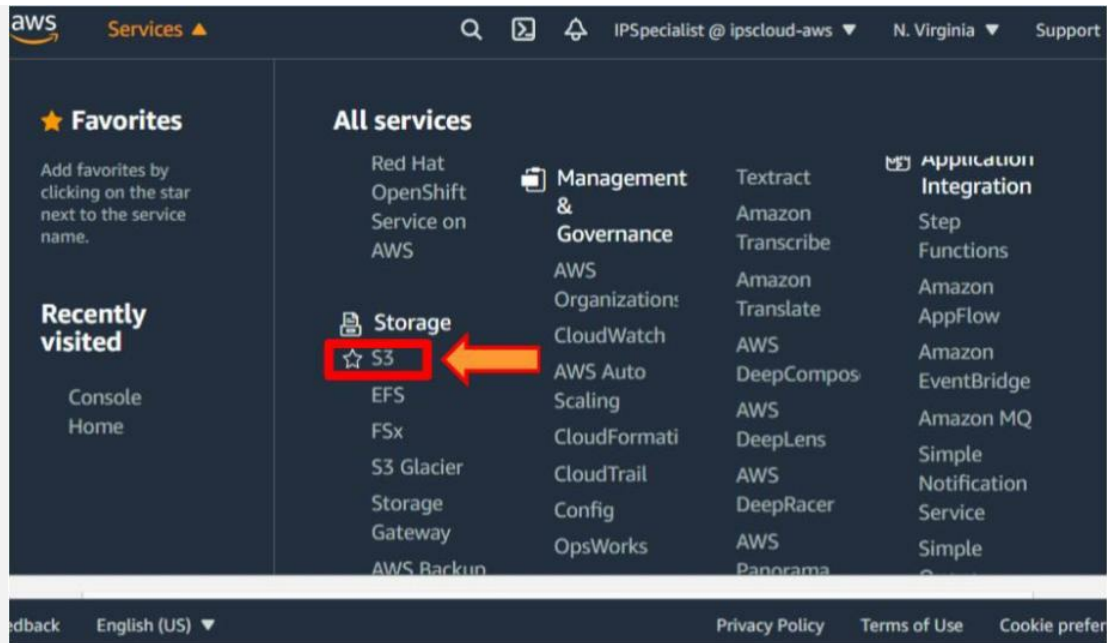
The screenshot shows the AWS Glue console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and user information. Below this, the job details for 'ips-user-data-job' are displayed. On the left, a diagram shows the job's structure: a database 'ips-user-data-database' with a table 'ips\_s3\_bucket' is transformed by 'ApplyMapping'. The right pane shows the 'Logs' tab, which includes a 'Progress bar' section. This section is highlighted with a red box and an orange arrow pointing to it. The progress bar shows the job is at Stage 0 (runJob at HadoopWriters.scala:129) with 7 output partitions. Below the progress bar, the 'Driver logs' are visible, showing logs from the DAGScheduler.

## Step 2: Reviewing Store Data

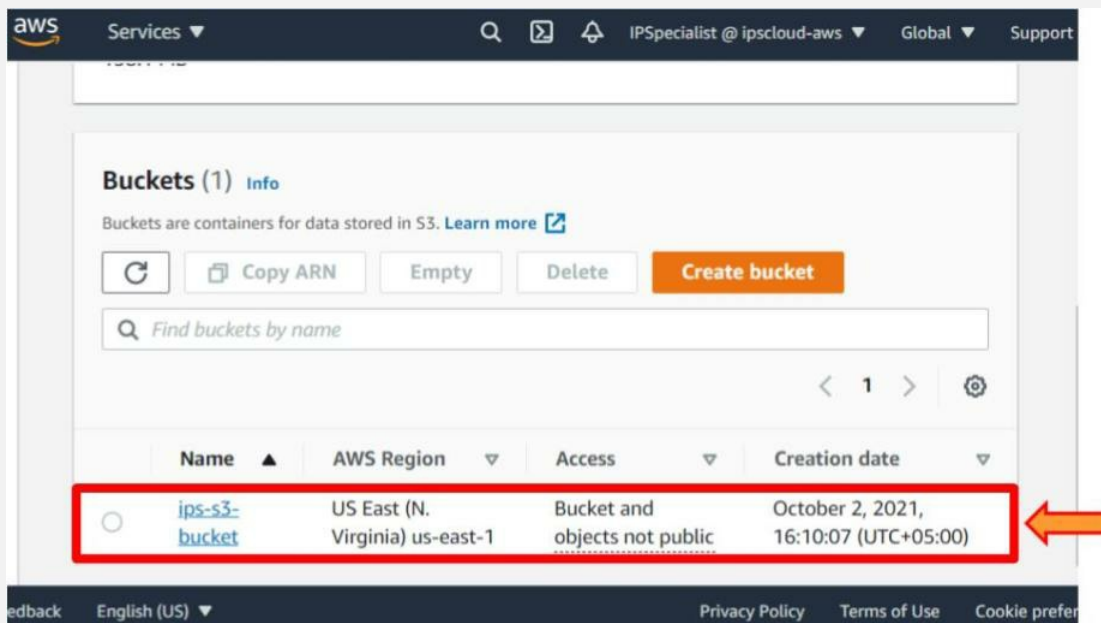
1. Click on the **Services**.

The screenshot shows the AWS Management Console. At the top, there's a navigation bar with the AWS logo, a search bar, and user information. Below this, the 'Services' dropdown menu is open, showing a list of AWS services. The 'Services' dropdown is highlighted with a red box and an orange arrow pointing to it. The main content area shows the 'AWS services' section with a 'Recently visited services' list and an 'All services' link.

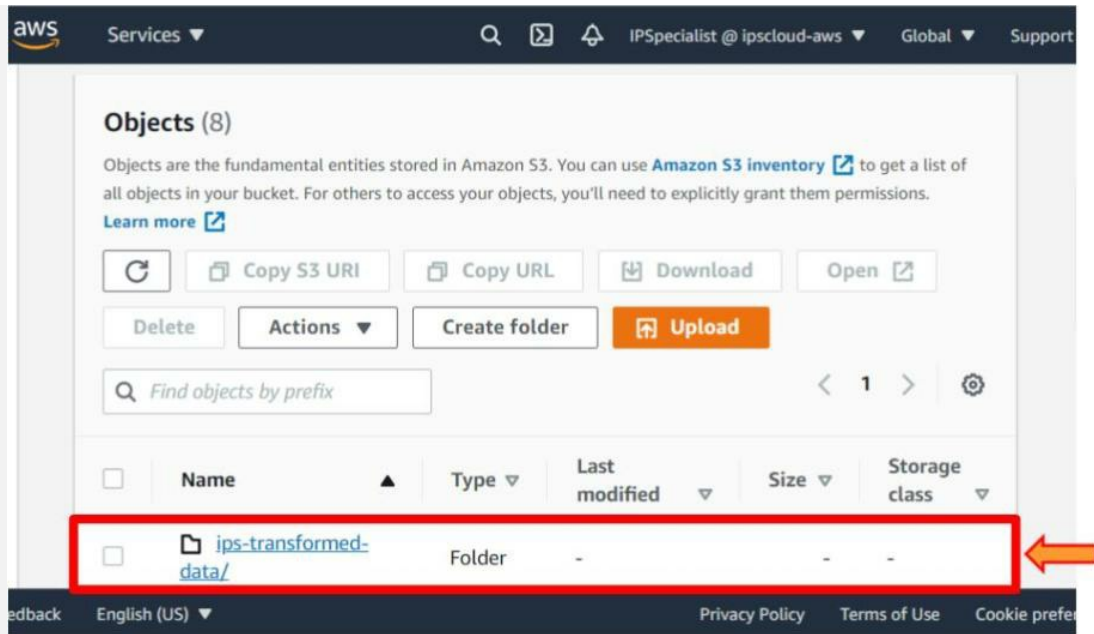
2. Select the **S3** from the **Storage**.



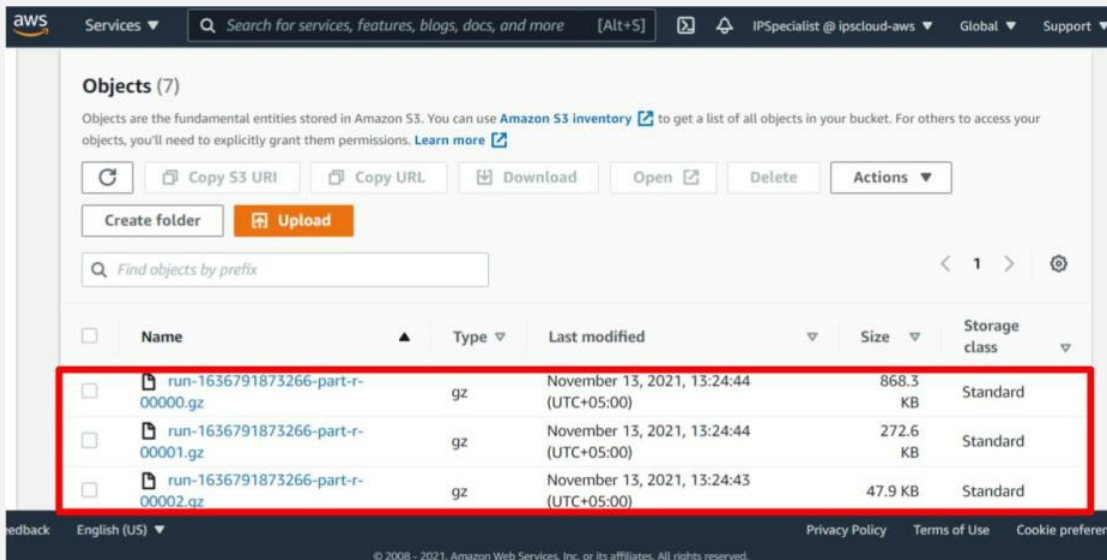
3. Click on the **ips-s3-bucket**.



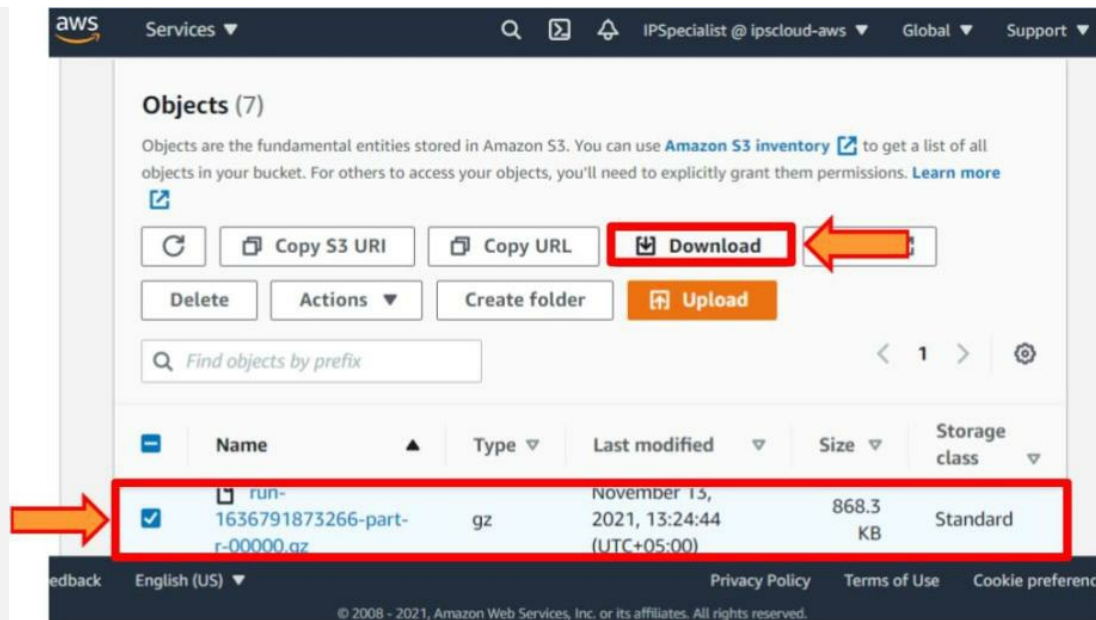
4. Click on the **ips-transformed-data** folder.



5. You should see the zip files.



6. Select one of the files. Then click on **Download**.



- Unzip the downloaded file. Open the file in any code editor you will file like it was done in the below image.

The screenshot shows a Visual Studio Code editor window. The file name is 'run-1636791873266-part-r-00000'. The content is a JSON array of objects, each containing a URL for a portrait image. The first few lines of the JSON are visible:

```

1  picture.large,picture.medium,picture.thumbnail,name.first,name.last,name.title,dob.date,dob.age,
2  "https://randomuser.me/api/portraits/men/62.jpg", "https://randomuser.me/api/portraits/med/men/62.jpg",
3  "https://randomuser.me/api/portraits/men/90.jpg", "https://randomuser.me/api/portraits/med/men/90.jpg",
4  "https://randomuser.me/api/portraits/women/84.jpg", "https://randomuser.me/api/portraits/med/women/84.jpg",
5  "https://randomuser.me/api/portraits/women/75.jpg", "https://randomuser.me/api/portraits/med/women/75.jpg",
6  "https://randomuser.me/api/portraits/women/50.jpg", "https://randomuser.me/api/portraits/med/women/50.jpg",
7  "https://randomuser.me/api/portraits/women/39.jpg", "https://randomuser.me/api/portraits/med/women/39.jpg",
8  "https://randomuser.me/api/portraits/men/96.jpg", "https://randomuser.me/api/portraits/med/men/96.jpg",
9  "https://randomuser.me/api/portraits/women/90.jpg", "https://randomuser.me/api/portraits/med/women/90.jpg",
10 "https://randomuser.me/api/portraits/men/82.jpg", "https://randomuser.me/api/portraits/med/men/82.jpg",
11 "https://randomuser.me/api/portraits/men/94.jpg", "https://randomuser.me/api/portraits/med/men/94.jpg",
12 "https://randomuser.me/api/portraits/men/8.jpg", "https://randomuser.me/api/portraits/med/men/8.jpg",
13 "https://randomuser.me/api/portraits/men/12.jpg", "https://randomuser.me/api/portraits/med/men/12.jpg",
14 "https://randomuser.me/api/portraits/women/59.jpg", "https://randomuser.me/api/portraits/med/women/59.jpg",
15 "https://randomuser.me/api/portraits/men/16.jpg", "https://randomuser.me/api/portraits/med/men/16.jpg",
16 "https://randomuser.me/api/portraits/women/75.jpg", "https://randomuser.me/api/portraits/med/women/75.jpg",
17 "https://randomuser.me/api/portraits/women/44.jpg", "https://randomuser.me/api/portraits/med/women/44.jpg",
18 "https://randomuser.me/api/portraits/women/88.jpg", "https://randomuser.me/api/portraits/med/women/88.jpg",
19 "https://randomuser.me/api/portraits/men/5.jpg", "https://randomuser.me/api/portraits/med/men/5.jpg",
20 "https://randomuser.me/api/portraits/men/75.jpg", "https://randomuser.me/api/portraits/med/men/75.jpg",
21 "https://randomuser.me/api/portraits/women/75.jpg", "https://randomuser.me/api/portraits/med/women/75.jpg"

```

## Output File Formats

If we are storing data in a relational database or a JDBC connection, the output file format would not matter. However, if we are storing it into a file server or S3, we can have various output file formats as follows;

### Output File Formats

|                                     |
|-------------------------------------|
| JSON*                               |
| CSV*                                |
| ORC                                 |
| Parquet                             |
| Avro                                |
| *optional compression (gzip, bzip2) |

*Table 9-01: Output File Formats*

**EXAM TIP:** The reason that JSON and CSV have an asterisk is that we have the option of compressing that data before it is stored off.

## Data Processing Units (DPUs)

| Job Types       | Minimum DPUs | Maximum DPUs | Default DPUs | Cost per DPU                                                                                           |
|-----------------|--------------|--------------|--------------|--------------------------------------------------------------------------------------------------------|
| Apache Spark    | 2            | 100          | 10           | Region<br>Dependent<br>\$0.44/hour<br><br>*\$0.44 is the<br>most common<br>(\$0.52, \$0.59,<br>\$0.69) |
| Spark Streaming | 2            | 100          | 5            |                                                                                                        |
| Python Shell    | 0.0625 or 1  | 1            | 0.0625       |                                                                                                        |

Table 9-02: Data Processing Units (DPUs)

**Example**

*First Run*

- Apache Spark
- Default 10 DPUs
- \$0.44/hour

30 minutes to run the job =  $\frac{1}{2}$  of an hour

$$\frac{1}{2} * \$0.44 = \$0.22 \text{ per DPU}$$

$$\$0.22 * 10 \text{ DPUs} = \$2.20 \text{ per job}$$

*Second Run*

- Apache Spark
- 25 DPUs
- \$0.44/hour

10 minutes to run the job =  $\frac{1}{6}$  of an hour

$$\frac{1}{6} * \$0.44 = \$0.073 \text{ per DPU}$$

$$\$0.073 * 25 \text{ DPUs} = \$1.82 \text{ per job}$$

*Third Run*

- Apache Spark
- 75 DPUs
- \$0.44/hour

6 minutes to run the job =  $\frac{1}{10}$  of an hour

$$\frac{1}{10} * \$0.44 = \$0.044 \text{ per DPU}$$

$$\$0.044 * 75 \text{ DPUs} = \$3.30 \text{ per job}$$

**Note:**

- AWS Glue Version 2.0 is billed in 1-second increments with a 1-minute minimum.
- AWS Glue Version 0.9 and 0.1 are billed in 1-second increments with a 10-minute minimum.

**EXAM TIP:** You can use CloudWatch metrics to determine under or over-provisioned DPUs in the cluster by monitoring the total number of actively running executors, the number of completed stages, and the number of maximum needed executors.

## Glue Jobs Run In Isolated

### *Glue Runs Jobs on Virtual Resources*

All the resources needed to run ETL jobs are provisioned and managed in its isolated service account.

### *What a Glue Job needs?*

You provide output data sources and input data targets in your VPC. In addition, you give the IAM role, VPC ID, subnet ID, and security group that is needed to access data sources and targets.

### *Traffic governed by Your VPC*

Traffic in, out, and within the spark environment is determined by your networking policies. The one exception by your networking policies is calls made to the AWS Glue API. However, these can be audited through CloudTrail.

### **EXAM TIP:**

**AWS Glue Jobs** – Glue jobs are the business logic that performs ETLs work in AWS Glue.

**Workflow Overview** – Various parts are needed for an AWS Glue job.

**Output Data Formats** – The different output formats Glue Jobs can perform.

**Data Processing Units (DPUs)** – The units used for processing your Glue Jobs.

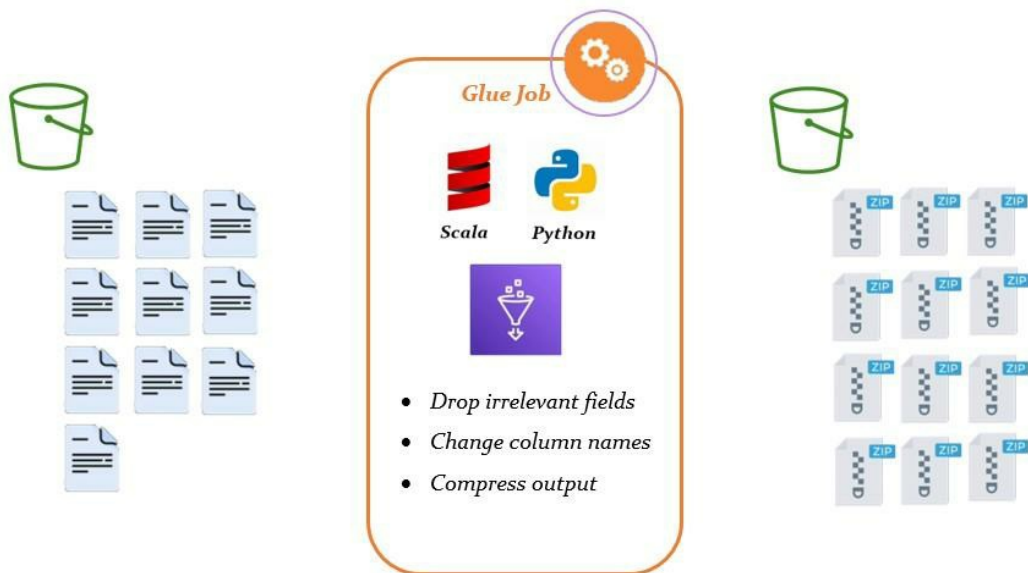
**Glue Jobs Run In Isolated** – Glue Jobs run on virtual resources, Glue jobs needs, and how traffic is governed.

---

# Job Bookmarks

## Explanation

Consider an S3 bucket that has some files in it. It will not only have a few files, but it may have a lot of files. Consider we create a Glue job with Scala or Python code that does simple transformations. The Glue job will scan the files and make the transformations. These transformations might be dropped to the relevant fields. It may change some column names or compress the output of the data. Once the Glue job finishes, it will store this data off into S3 and perform whatever type of transformation we want. In this case, we dropped the irrelevant fields, changed the column names, and compressed the output, so we zipped up the files.



*Figure 9-07: Job Bookmarks*

Consider we add more files to our S3 bucket. If we rerun this job, the only files that will be scanned are the new files added to the S3 bucket.

## Job Bookmarks Defined

AWS Glue is a serverless environment for extracting, transforming, and loading massive amounts of data from a variety of sources for analytics. When rerunning a task on a scheduled interval, it features a job bookmarks that process incremental data. The states for various task elements, such as inputs, transformations, and targets, make up a job bookmark. This is accomplished by storing state information from a job run, which aids AWS Glue in avoiding the reprocessing of outdated data.

## Options For Job Bookmarks

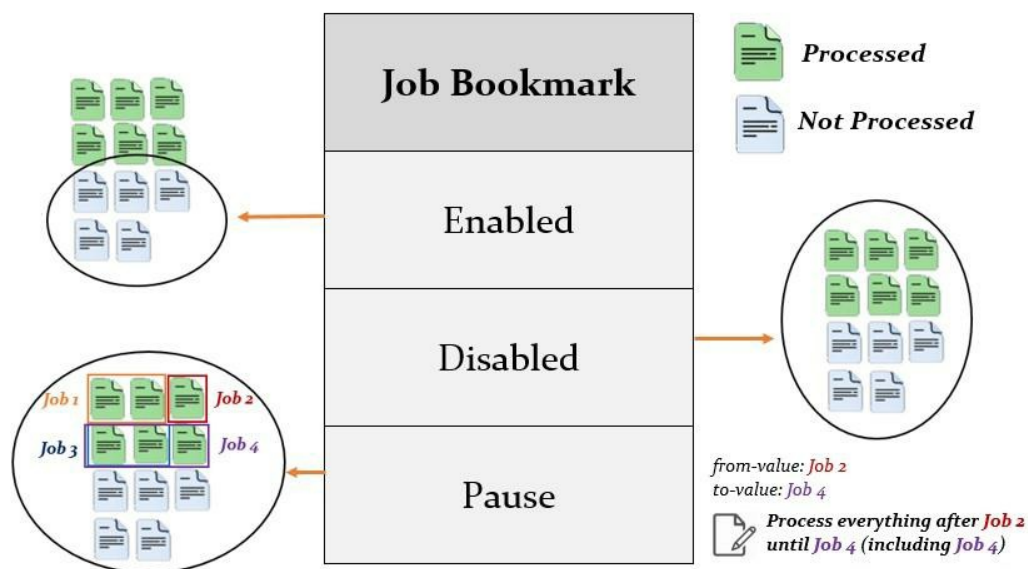
| Job Bookmark | Description                                                                                                                                                   |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Enabled      | Motives the job to update the state after a run to keep track of previously processed data. If new data is processed, it will start from the last checkpoint. |
| Disabled     | Job Bookmarks are not utilized and the job always processes the entire dataset. This is the default.                                                          |
| Pause        | Processes incremental data defined by a from-value and to-value range. The data is after the from-value job and before the to-value appointment (inclusive).  |

Table 9-03: Options for Job Bookmarks

The green files represent processed files, and the white files represent files that have not been processed yet.

Whenever the job bookmarks are set to enable, the only files that will be processed are the white files. When the job bookmarks are disabled; it will process the entire data set.

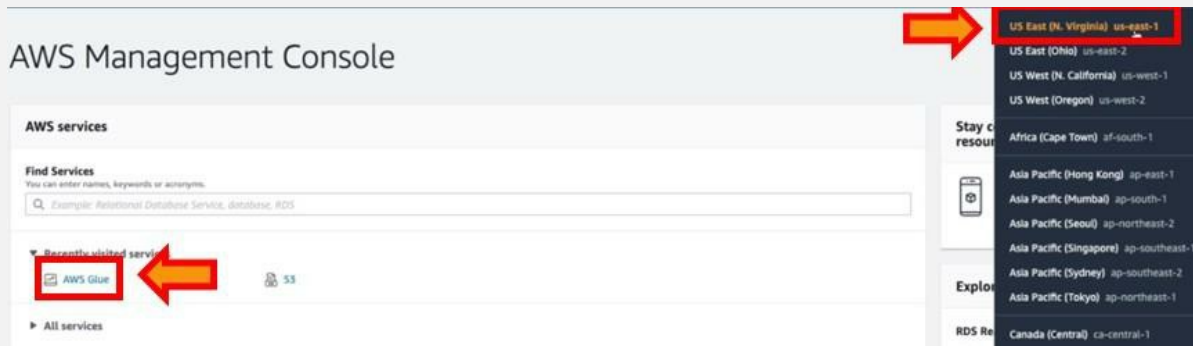
Whenever the job bookmarks are set to pause bookmark, consider we ran several different Glue jobs already. In the first job, we process two files; in the second job, we process a file; in the third job, we process two files; and in the 4th job, we process the single file, while some files have not been processed yet. Whenever we set up the pause job bookmark, we need to set up a from-value and a to-value. In this case, we will set the from-value to job number 2 and the to-value to job number 4. It will process everything after job 2 until job 4, including job 4. This case will process all the data in job 3 and job 4, so any new data after job 4 or after the two values will not be processed.



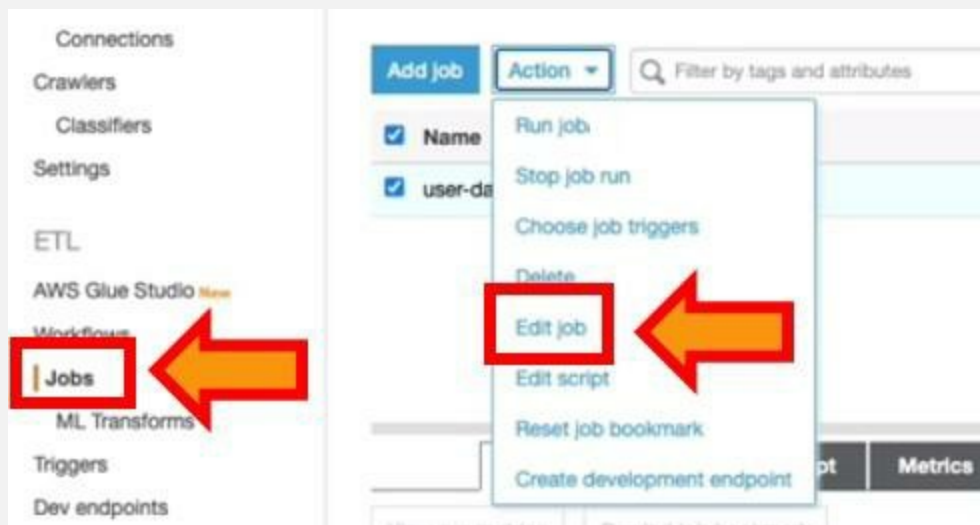
*Figure 9-08: Job Bookmarks*

## Demo 9-01: How To Set Up Job Bookmarks

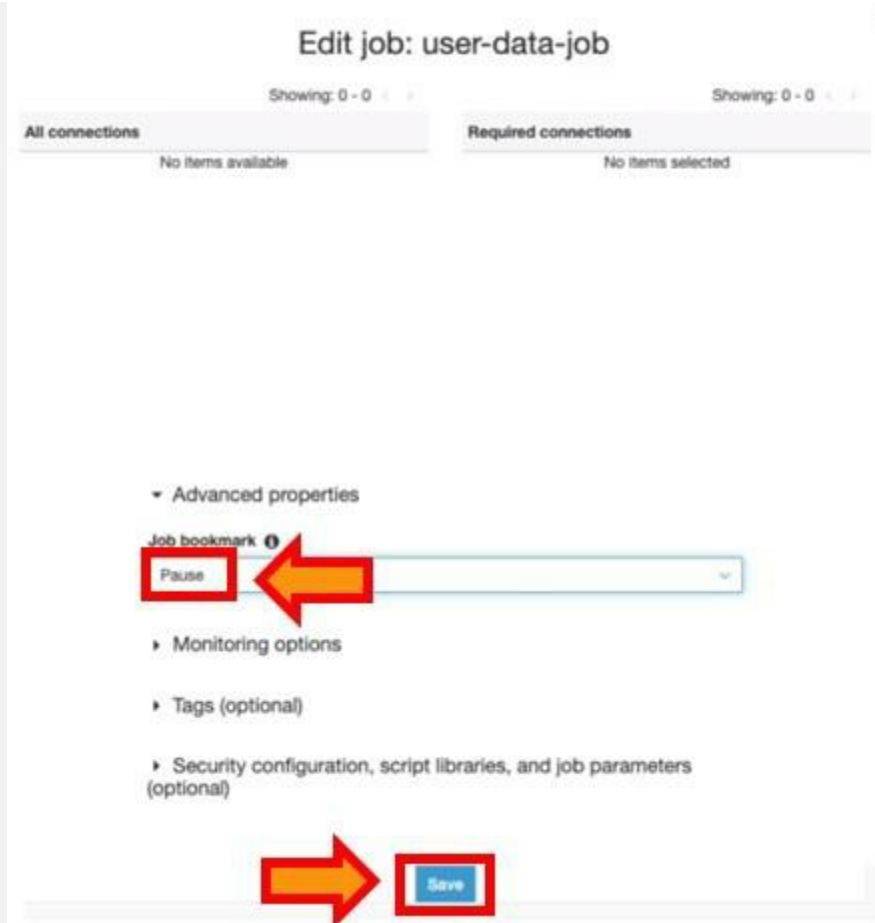
1. Login to **AWS Management Console**.
2. Click on **AWS Glue**.



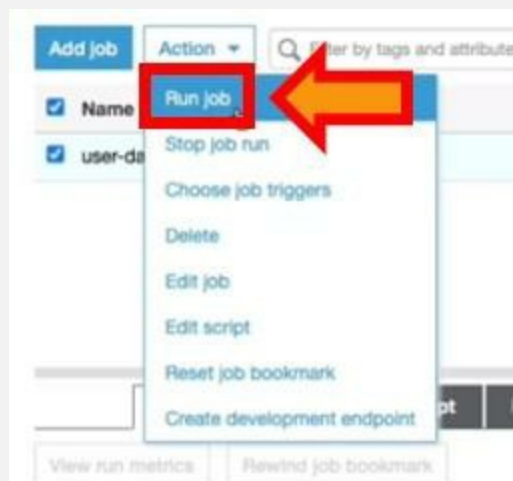
3. Click on **Jobs**.
4. Click on **Edit Job** under actions.



5. Define Job bookmark.
6. Click on **Save**.



7. Click on **Run Job** under action.



8. Define Job bookmark.

9. Click on **Run Job**.

### Parameters (optional)

Review and override parameter values, as needed, before running this job. Changes affect this run only. Edit a job to change default parameter values.

▼ Advanced properties

Job bookmark ⓘ

Enable

▶ Monitoring options

▶ Tags

▶ Security configuration, script libraries, and job parameters

Only job **user-data-job** is run. Jobs dependent on the completion of job **user-data-job** will not be run. To run a job and trigger dependent jobs, define an on-demand trigger.

Run job

10. Job is running.

Job "user-data-job" is now running.

Add job Action Filter by tags and attributes

| Name          | Type  | ETL language | Script location           |
|---------------|-------|--------------|---------------------------|
| user-data-job | Spark | python       | s3://aws-glue-scripts-... |

## Getting Started with Athena

### What Is Athena?

Athena helps to easily query your S3 data. You can use standard SQL queries to analyze data directly in S3. Athena is serverless, and you can only pay for the queries that you run. Athena automatically scales, so results are fast, even with large datasets and complex questions.

### Athena Federated Queries

Athena could only query data in S3, but since customers have data in other data sources, AWS created the ability to connect to external data sources using Athena federated queries. Athena uses data source connectors to run on AWS Lambda to run federated queries. A data source connector is a section of code that can translate between Athena and your target data source. With this new feature, you can query data in places or build pipelines to extract data from multiple data sources, such as CloudWatch Metrics, DynamoDB, Elasticsearch, JDBC compliant data sources like Redshift and RDS, and store the query results in S3.

The screenshot shows the AWS Athena console interface for setting up a federated query. It is divided into two main sections: 'Choose where your data is located' and 'Choose a data source (beta)'. The first section has two options: 'Query data in Amazon S3' (selected by default) and 'Query a data source (beta)'. The second section, 'Choose a data source (beta)', lists various data sources with radio buttons for selection. The selected source is 'Amazon CloudWatch Logs'. Other sources include Amazon CloudWatch Metrics, Amazon DocumentDB, Amazon DynamoDB, Amazon Redshift, Apache HBase, MySQL, PostgreSQL, Redis, and 'All other data sources' (which allows creating a custom connector). At the bottom right, there are 'Cancel' and 'Next' buttons.

Choose where your data is located

Athena queries data where it is. Data is not loaded or moved. [Learn more](#)

☐ Query data in Amazon S3

Choose an external data catalog.

☒ Query a data source (beta)

Configure a connector for common data sources.

Choose a data source (beta)

Choose the data source to query with Athena. After you choose a data source, you will configure a Lambda function to handle the connection. [Learn more](#)

☒ Amazon CloudWatch Logs

☐ Amazon CloudWatch Metrics

☐ Amazon DocumentDB

☐ Amazon DynamoDB

☐ Amazon Redshift

☐ Apache HBase

☐ MySQL

☐ PostgreSQL

☐ Redis

☐ All other data sources  
Create your own data connector

Cancel Next

*Figure 9-09: Athena federated Queries*

**EXAM TIP:** If you have data in sources other than S3, you can use

Federated Query (beta) to query the data in place or build pipelines that extract data from multiple data sources and store them in Amazon S3.

## **Athena Data Formats And Integrations**

### ***Data Formats***

Athena helps you analyze unstructured, semi-structured, and structured data stored in S3. Examples include CSV, TSV, JSON, Textfiles, Parquet, ORC, and Snappy, Zlib, LZO, and GZIP.

### ***Integrates With QuickSight***

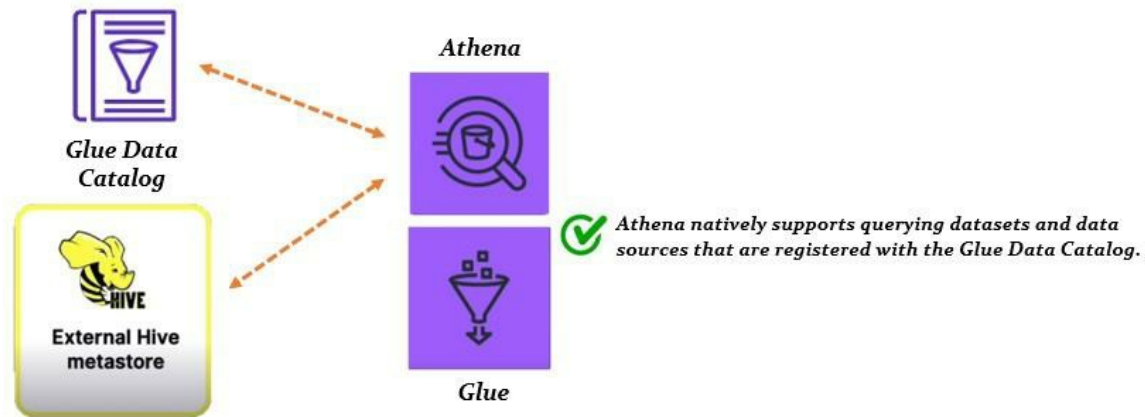
Create easy data visualizations by using Athena to generate reports to explore data with BI tools of SQL clients connected with JDBC and ODBC drivers.

### ***Integrates With AWS Glue***

Athena Integrates with AWS Glue Data Catalog, allowing you to create tables and query data in Athena as well as use the ETL and data discovery features of AWS Glue.

## **Connecting to Data Sources**

Athena natively supports querying datasets and data sources that are registered with the Glue Data Catalog. You can have a data connector using an external Hive metastore to query datasets in Amazon S3. You can also use a data connector for external Hive meta stores to query data in S3 that is using an Apache Hive metastore. You do not have to migrate your Hive metastore data to the AWS Glue Data Catalog.



*Figure 9-10: Connecting to Data Sources*

**EXAM TIP:** Athena natively supports querying datasets and data sources registered with the Glue Data Catalog.

## More On Integrations

| AWS Services           | Integrations                                                                                                                                 |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| CloudTrail             | Query CloudTrail logs to analyze AWS service activity                                                                                        |
| CloudFront             | You can query CloudFront logs to explore user's surfing patterns across your content served by CloudFront                                    |
| Elastic Load Balancing | You can query logs and see the traffic, latency, and bytes transferred to and from Elastic Load Balancing instances and backend applications |
|                        | Query VPC flow logs about the IP                                                                                                             |

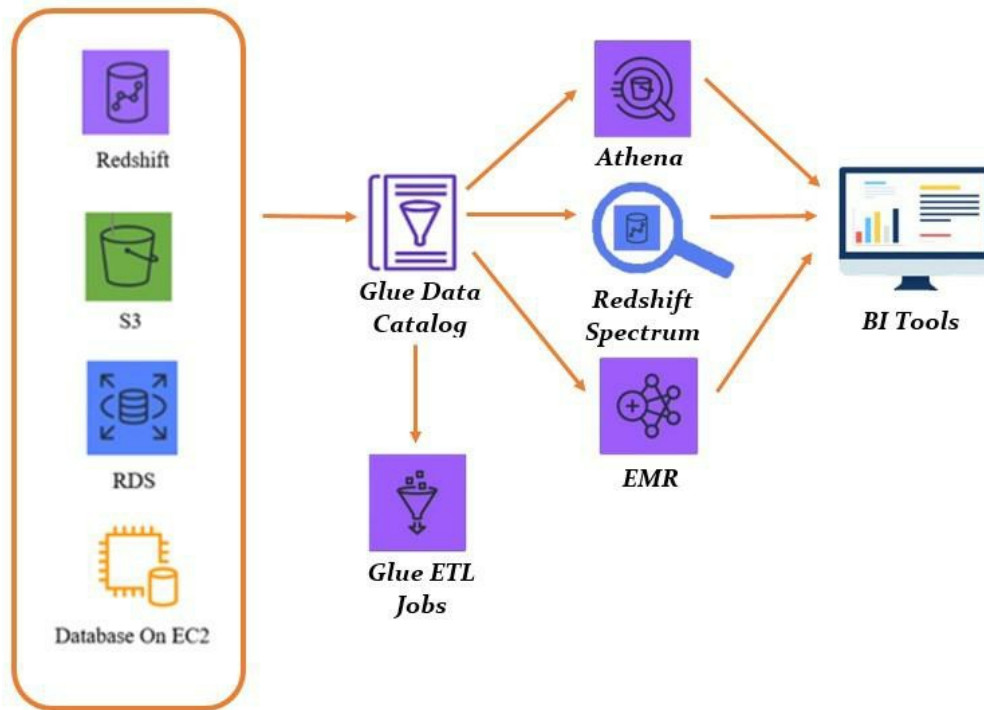
|                           |                                                                         |
|---------------------------|-------------------------------------------------------------------------|
| VPC (Flow Logs)           | traffic going to and from the network interface in a VPC                |
| CloudFormation            | Create Athena components using CloudFormation                           |
| IAM                       | Using IAM, you can control Athena API actions using permission policies |
| Systems Manager Inventory | You can query inventory from multiple AWS regions and accounts          |

*Table 9-04: AWS Services and Integration*

## Athena Use Cases

### *Ad-Hoc and BI Tools*

Athena is an easy-to-use tool for ad hoc queries and business intelligence tools integrations. We can do this by setting up a Glue Data Catalog from a variety of data sources. We can use Glue to perform ETL jobs if we clean, enrich, or transform our data. Once we have our metadata information in our Glue Data Catalog, we can use services like Athena, Redshift Spectrum, and EMR to further run ad hoc queries or other processes and analyze our data. We can use BI tools like QuickSight or Tableau to visualize our data, create some dashboards, share them, and create reports for management.



*Figure 9-11: Use Cases - Ad-hoc And BI Tools*

### ***Joining Data***

Assume we have some frequently accessed data stored in Aurora and some historical or cold data stored in S3. We can set up a Glue Data Catalog with metadata information about these datasets and schemas to perform joins on frequently accessed data in Aurora and our historical data stored in S3. When a user comes in and requests API Gateway, our data's querying and joining can occur and send. We can use Athena to join data from relational databases and data that sits in S3.

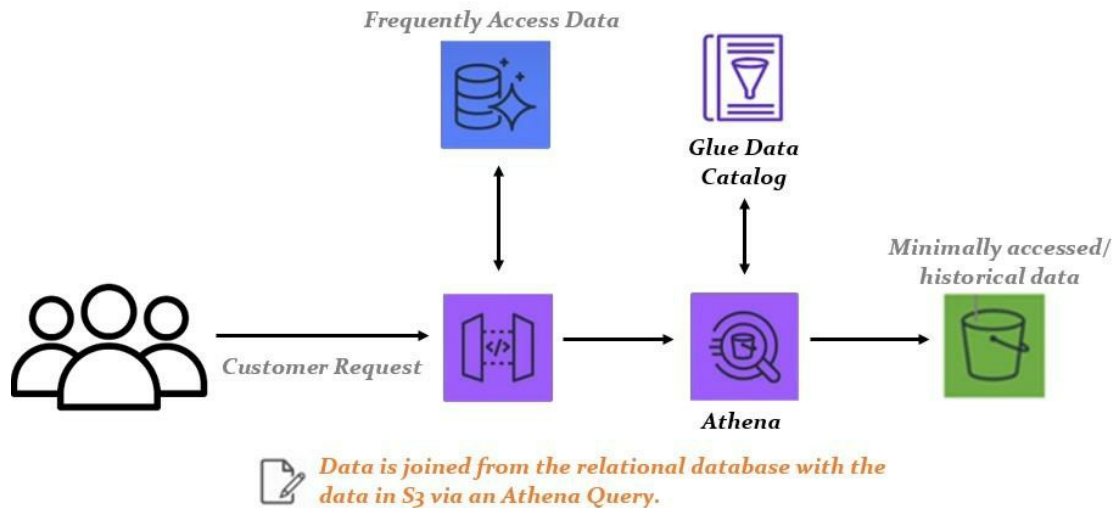
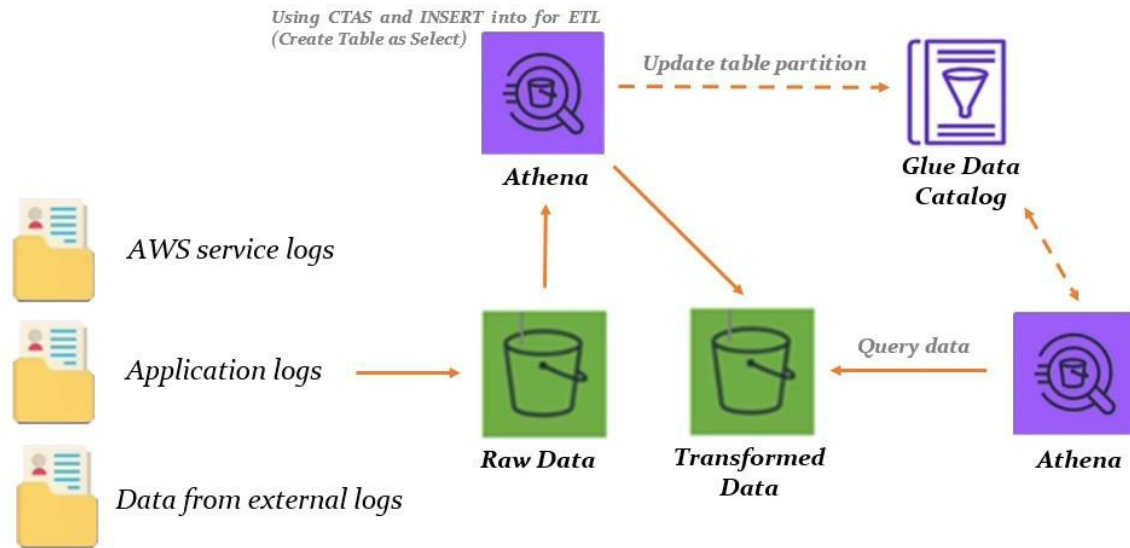


Figure 9-12: Use Cases - Joining Data

### ***ETL Pipelines and Integrations***

The last use case is creating ETL pipelines with data like AWS service logs or application logs as well as data from external vendors. This data is typically stored in S3 as raw data or as a data lake. We can utilize Athena and Glue Data Catalog to create a table SELECT AS and insert statements to perform ETL processes on that data. Then, we can store that transformed data back into S3, allowing us to partition and convert it into a columnar data format to optimize it for data analysis. Once the data is transformed, we can then use Athena to query the data and return the important information we seek.



**Figure 9-13: Use Cases - ETL Pipelines and Integrations**

### EXAM TIP:

- What Is Athena? – Serverless querying tool to easily query data in S3.
- Athena Data Formats And Integrations – Various file formats and compressions Athena can query, and many serve Athena can integrate with.
- Athena Use Cases – Ad-hoc queries, joining data from multiple data sources, creating ETL pipelines, and transforming your data

## Demo 9-02: Amazon Athena

### Introduction

#### 1. Amazon Athena

Amazon Athena is a query service that allows you to examine data in Amazon S3 using conventional SQL easily. Because Athena is serverless, there is no infrastructure to set up or operate. You can begin analyzing the data right away. You do not even need to import your data into Athena; it works with S3 data immediately. You can log into the Athena Management Console, specify your schema, and begin querying. Amazon Athena deals with several common data formats, including

CSV, JSON, ORC, Apache Parquet, and Avro. It also leverages Presto with full standard SQL support. At the same time, Amazon Athena is perfect for ad-hoc querying and connects with Amazon QuickSight for rapid visualization. It can also perform complicated analyses, such as massive joins, window functions, and arrays.

## **2. AWS Glue**

Amazon Glue is a serverless data integration service that simplifies data identification, preparation, and integration for analytics, machine learning, and application development. Amazon Glue has all the tools required for data integration, allowing you to begin analyzing and utilizing your data in minutes rather than months. AWS Glue offers both visual and code-based interfaces to help with data integration. The AMAZON Glue Data Catalog allows users to discover and retrieve data easily. ETL workflows may be created and executed by data engineers and ETL (extract, transform, and load) developers. AMAZON Glue DataBrew allows data analysts and data scientists to enhance, clean visually, and standardize data without writing code.

## **3. AWS Simple Storage Service (S3)**

Amazon S3 is a type of object storage that allows you to store and recover any quantity of data from any location. It is a low-cost storage solution with business resilience, reliability, efficiency, privacy, and infinite expansion.

Amazon S3 is a web service that allows you to store and retrieve an infinite quantity of data from any place and at any time. You may quickly create projects that integrate cloud-native storage using this service. Because Amazon S3 is easily customizable and you only pay for what you use, you can start small and scale up as needed without

sacrificing performance or dependability.

## **Problem**

Assume you are a Data Analytics engineer in an organization. The organization runs many e-commerce platforms such as clothing, smartphones, and grocery items. They want you to do customers' real-time streaming stored data analysis using SQL queries. So, how can you automate this task?

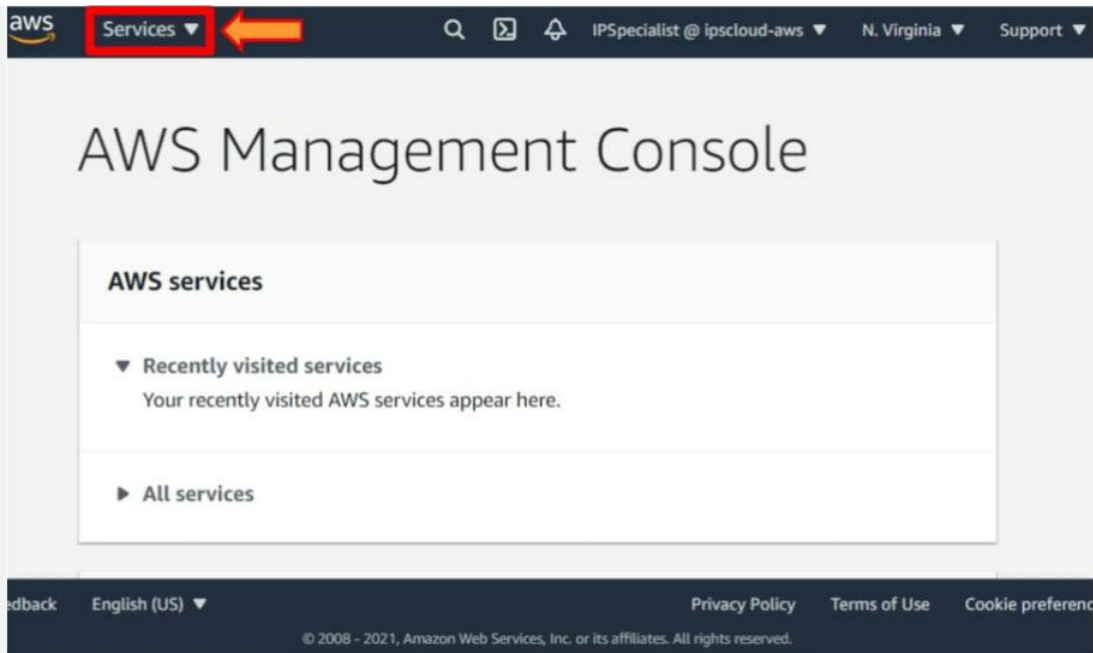
## **Solution**

The solution is to use the AWS Athena service. Athena does not have any servers. You do not need to set up or operate any servers or data warehouses to query your data rapidly. Point to your data on Amazon S3, set the schema, and use the built-in query editor to begin querying. Amazon Athena allows you to access all of your data in S3 without having to put up sophisticated data extraction, transformation, and loading processes (ETL). You also use AWS Glue to transform the store data into the S3 bucket.

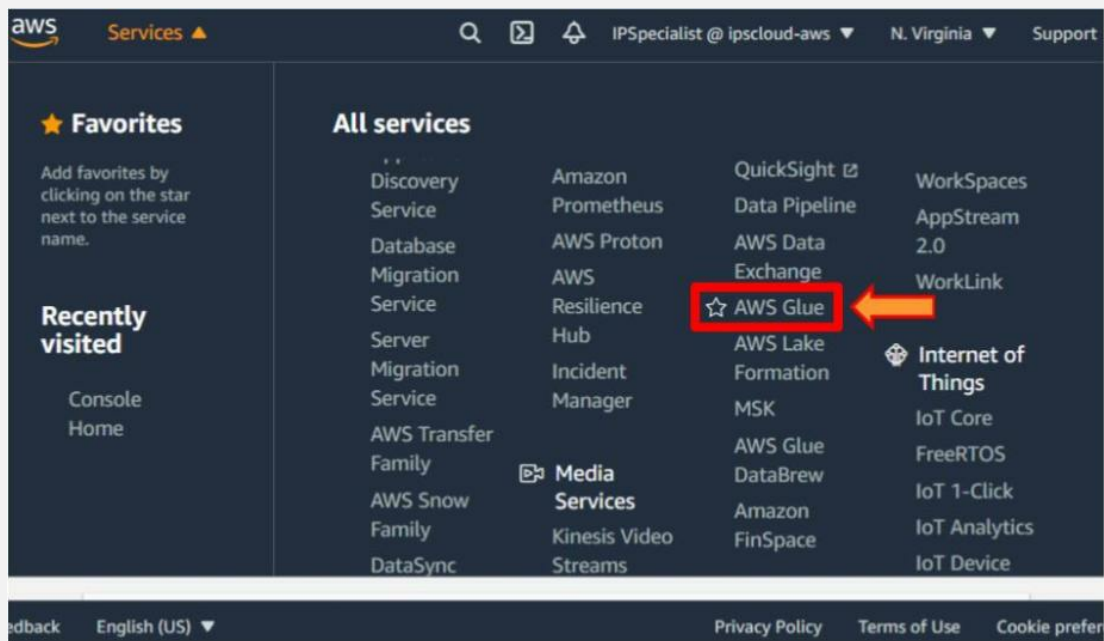
**Note:** Before starting the demo, create an S3 bucket.

### **Step 1: Create AWS Glue Job**

1. Log in to the **AWS Console**.
2. Click on the **Services**.



3. Select the **AWS Glue** from the **Analytics**.



4. Click on the **Jobs** from the left-hand side menu.

**Tables**

A table is the metadata definition that represents your data, including its schema. A table can be used as a source or target in a job definition.

[Add tables](#) [Save view](#) Showing: 1 - 1 [Refresh](#) [Settings](#)

| <input type="checkbox"/> | Name          | Database               | Location        | Classi | Last updated | Deprec |
|--------------------------|---------------|------------------------|-----------------|--------|--------------|--------|
| <input type="checkbox"/> | ips_s3_bucket | ips-user-data-database | s3://ips-s3-... | json   | 12 Novemb... |        |

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

5. Click on the **Add Job** button.

**Jobs**

A job is your business logic required to perform extract, transform and load (ETL) work. Job runs are initiated by triggers which can be scheduled or driven by events.

[User preferences](#)

[Add job](#) [Filter by tags and attribut](#) Showing: 1 - 1 [Refresh](#) [Help](#)

| <input type="checkbox"/> | Name              | Type  | ETL language | Script location | Last modified | Job bookmark |
|--------------------------|-------------------|-------|--------------|-----------------|---------------|--------------|
| <input type="checkbox"/> | ips-user-data-job | Spark | python       | s3://...        | 13 Nove...    | Disa...      |

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

6. Give the job name **ips-flatten-data-job**.

aws Services ▾

IPSpecialist @ ipscloud-aws ▾ N. Virginia ▾ Support ▾

### Add job

Configure the job properties

**Name**

ips-flatten-data-job

**IAM role** ⓘ

Choose an IAM role ▾ ↻

Ensure that this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job. [Create IAM role.](#)

**Type**

Spark ▾

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

7. Select the **IAM Role** which you previously created in the AWS Glue Job demo.

aws Services ▾

IPSpecialist @ ipscloud-aws ▾ N. Virginia ▾ Support ▾

### Add job

Configure the job properties

**Name**

ips-flatten-data-job

**IAM role** ⓘ

AWSGlueServiceRole-IPS-MasterRole ▾ ↻

Ensure that this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job. [Create IAM role.](#)

**Type**

Spark ▾

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

8. Click on the **Monitoring** option.

aws Services ▾

IPSpecialist @ ipscloud-aws ▾ N. Virginia ▾ Support ▾

## Add job

- Job properties
- Data source
- Transform type
- Data target
- Schema

- Monitoring options
- Tags (optional)
- Security configuration, script libraries, and job parameters (optional)
- Catalog options (optional)

Next

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

9. Select the **Job Metrics** and **Continuous logging**.

aws Services ▾

IPSpecialist @ ipscloud-aws ▾ N. Virginia ▾ Support ▾

## Add job

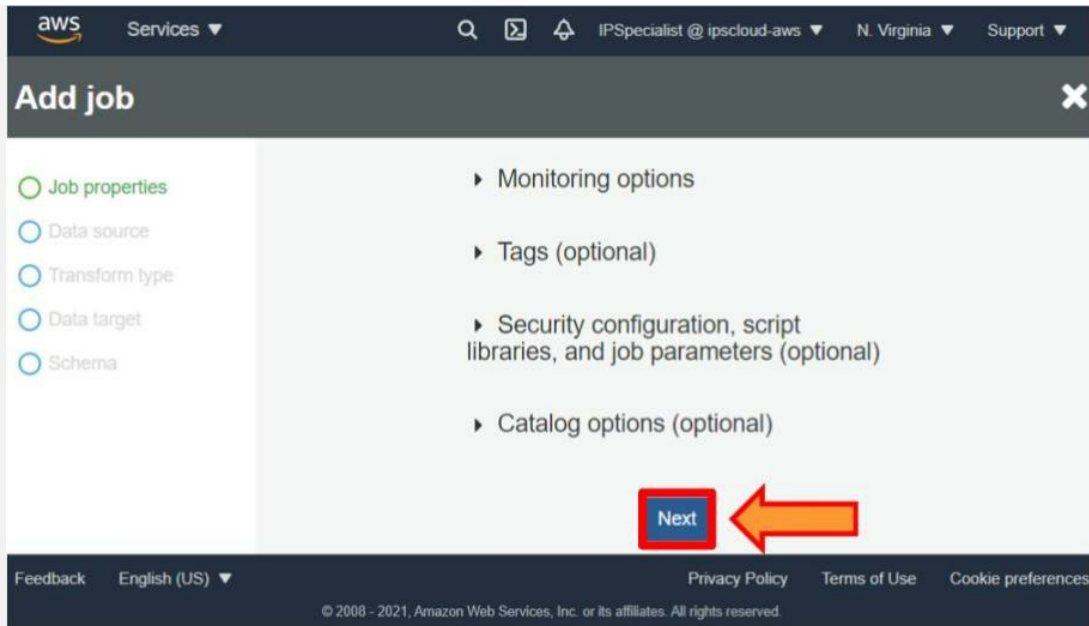
- Job properties
- Data source
- Transform type
- Data target
- Schema

- Monitoring options
  - Job metrics
  - Continuous logging
- Log filtering
  - Standard filter
  - No filter
- Spark UI
- Tags (optional)

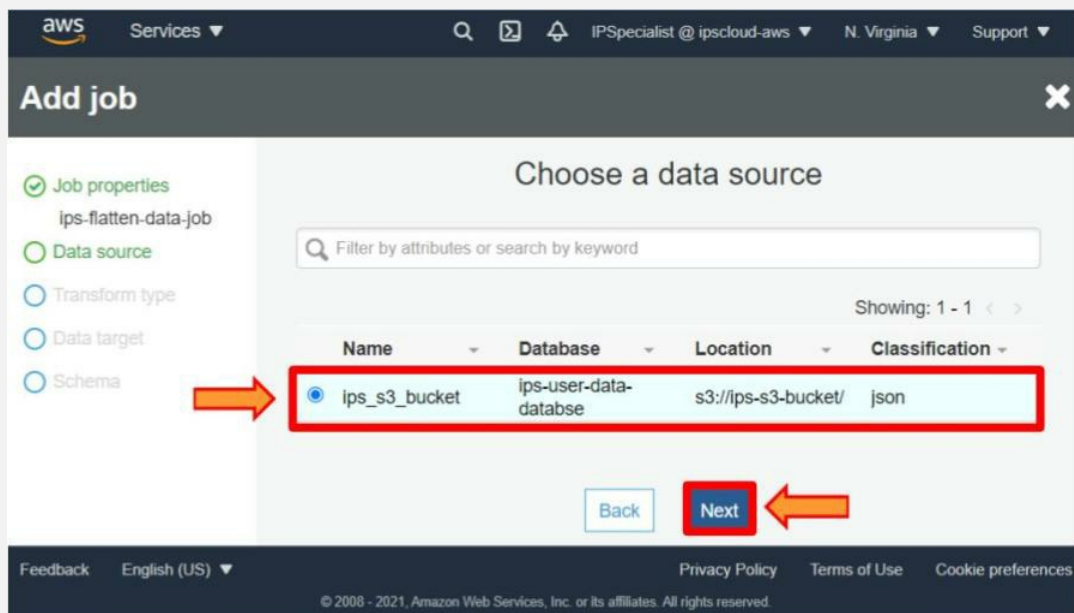
Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

10. Click on the **Next** button.



11. Select the **ips-s3-bucket**, and then click on the **Next** button.



12. Select the **Change Schema**, and then click on the **Next** button.

The screenshot shows the AWS Glue 'Add job' console. On the left, a sidebar lists the steps: 'Job properties' (completed), 'Data source' (completed), 'Transform type' (in progress), 'Data target' (pending), and 'Schema' (pending). The main area shows two radio button options: 'Change schema' (selected) and 'Find matching records'. The 'Change schema' option is highlighted with a red rectangle and an orange arrow. Below the options are 'Back' and 'Next' buttons, with the 'Next' button also highlighted by a red rectangle and an orange arrow. The footer contains links for Feedback, English (US), Privacy Policy, Terms of Use, and Cookie preferences, along with a copyright notice for 2008-2021 Amazon Web Services.

13. Select the **Create tables in your data target**.

This screenshot shows the 'Add job' console at the 'Data target' step. The sidebar now shows 'Data target' as completed. The main area has two radio button options: 'Create tables in your data target' (selected) and 'Use tables in the data catalog and update your data target'. The 'Create tables in your data target' option is highlighted with a red rectangle and an orange arrow. Below the options, there are dropdown menus for 'Data store' (set to 'JDBC') and 'Connection' (set to '- Select one -'). An 'Add connection' button is located below the connection dropdown. At the bottom, there is a 'Database name' field with an information icon. The footer is identical to the previous screenshot.

14. Select the **Amazon S3** data store.

aws Services IPSpecialist @ ipscloud-aws N. Virginia Support

## Add job

- Job properties
  - ips-flatten-data-job
- Data source
  - ips\_s3\_bucket
- Transform type
  - Change schema
- Data target
- Schema

☒ Create tables in your data target  
☐ Use tables in the data catalog and update your data target

**Data store**  
Amazon S3

**Format**  
JSON

**Compression type**  
None

Feedback English (US) Privacy Policy Terms of Use Cookie preferences  
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

15. Select the **CSV** format.

aws Services IPSpecialist @ ipscloud-aws N. Virginia Support

## Add job

- Job properties
  - ips-flatten-data-job
- Data source
  - ips\_s3\_bucket
- Transform type
  - Change schema
- Data target
- Schema

Amazon S3

**Format**  
CSV

**Compression type**  
None

**Connection**  
- Select one -  
[Add connection](#)

**Target path**

Feedback English (US) Privacy Policy Terms of Use Cookie preferences  
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

16. Select the **gzip** compression type.

The screenshot shows the 'Add job' configuration page in the AWS Glue console. On the left, a sidebar lists configuration steps: Job properties (checked), Data source (checked), Transform type (checked), Data target (unchecked), and Schema (unchecked). The main area shows the 'Compression type' dropdown set to 'gzip', which is highlighted with a red box and an orange arrow. Below it, the 'Connection' dropdown is set to '- Select one -'. The 'Target path' field contains the text 's3://bucket/prefix/object'. At the bottom, there are links for 'Feedback', 'English (US)', 'Privacy Policy', 'Terms of Use', and 'Cookie preferences', along with a copyright notice for 2008-2021 Amazon Web Services, Inc.

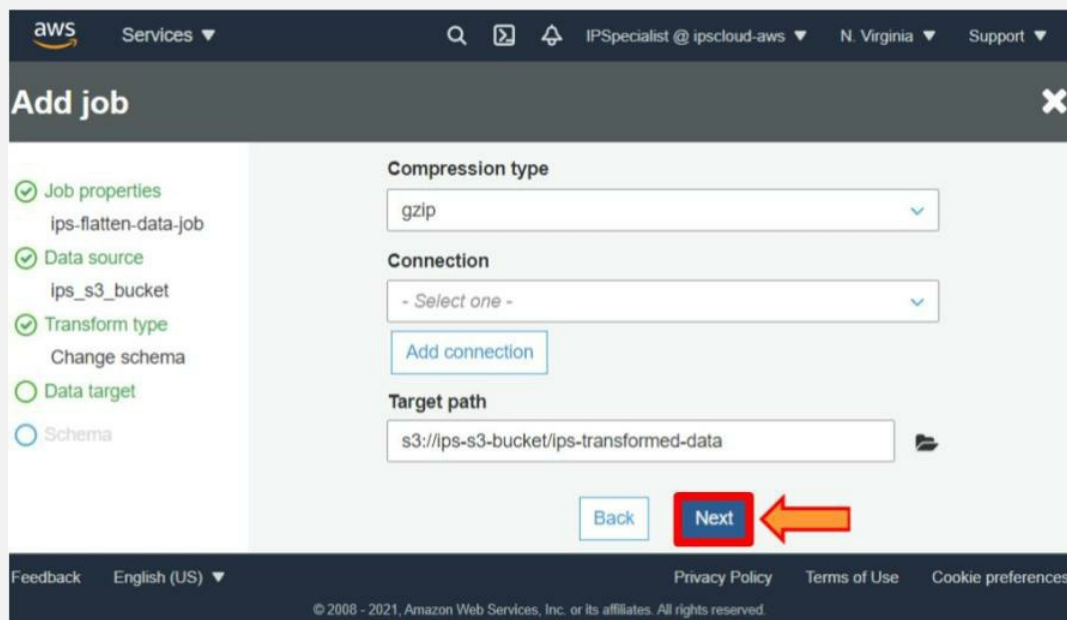
17. Click on the **Folder** button icon.

This screenshot shows the same 'Add job' configuration page. The 'Compression type' is still 'gzip'. The 'Target path' field now contains 's3://bucket/prefix/object'. A red box highlights the 'Folder' button icon (a blue folder icon) located to the right of the 'Target path' field, with an orange arrow pointing to it. Below the 'Target path' field, there are 'Back' and 'Next' buttons. The footer remains the same as in the previous screenshot.

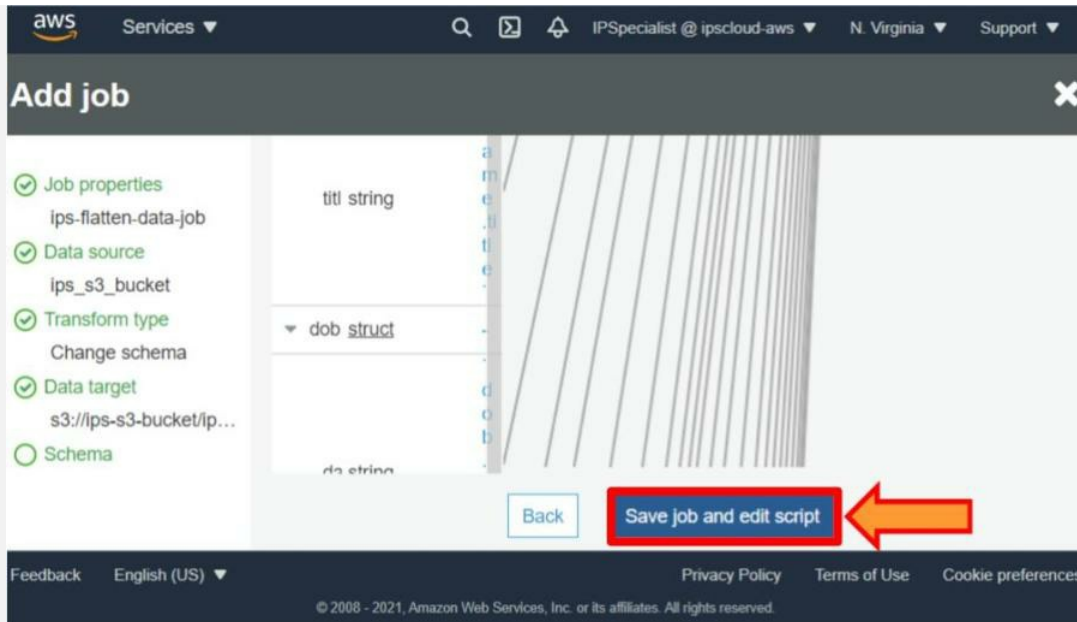
18. Select the **ips-transformed-data** folder in the **ips-s3-bucket**. Then, click on the **Select** button.



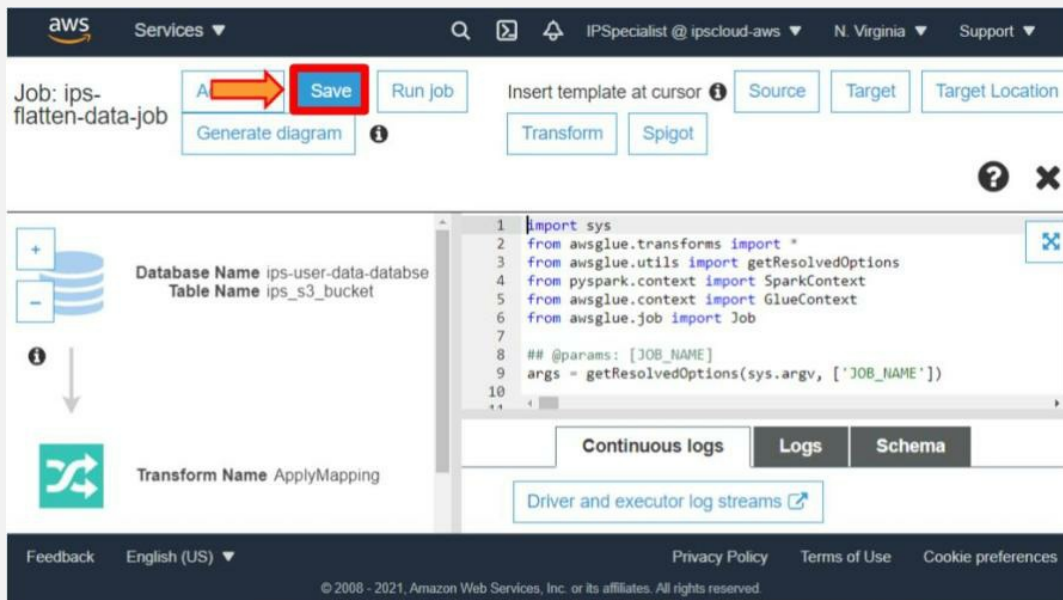
19. Click on the **Next** button.



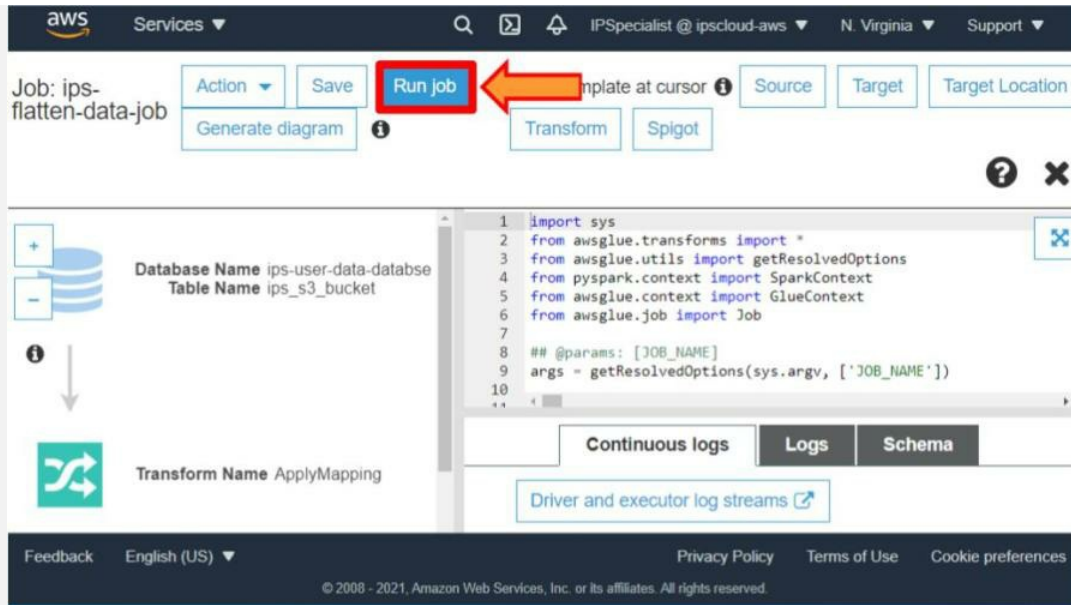
20. Scroll down, and click on the **Save job and edit script** button.



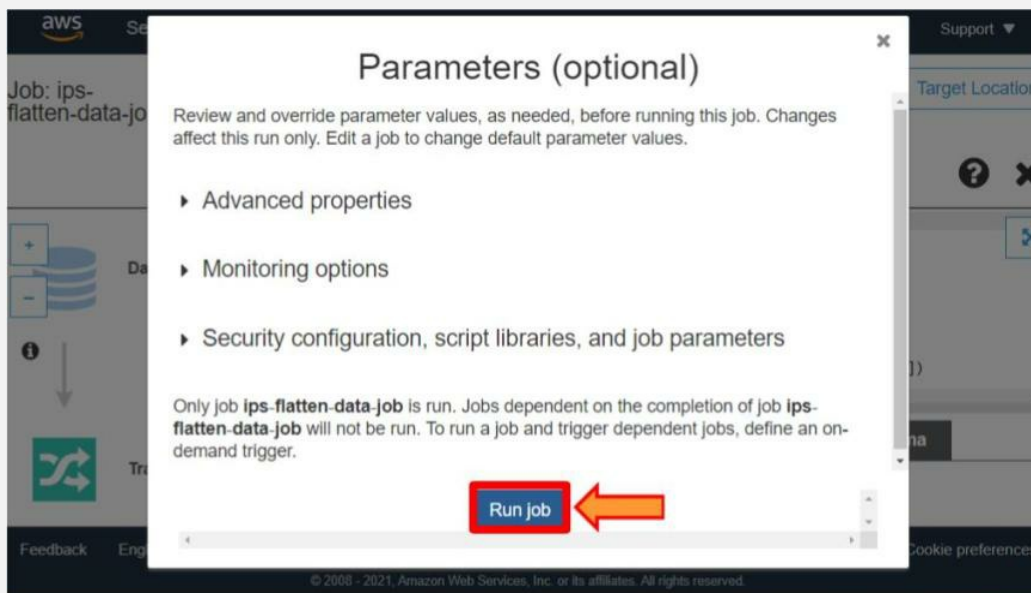
21. Click on the **Save** button.



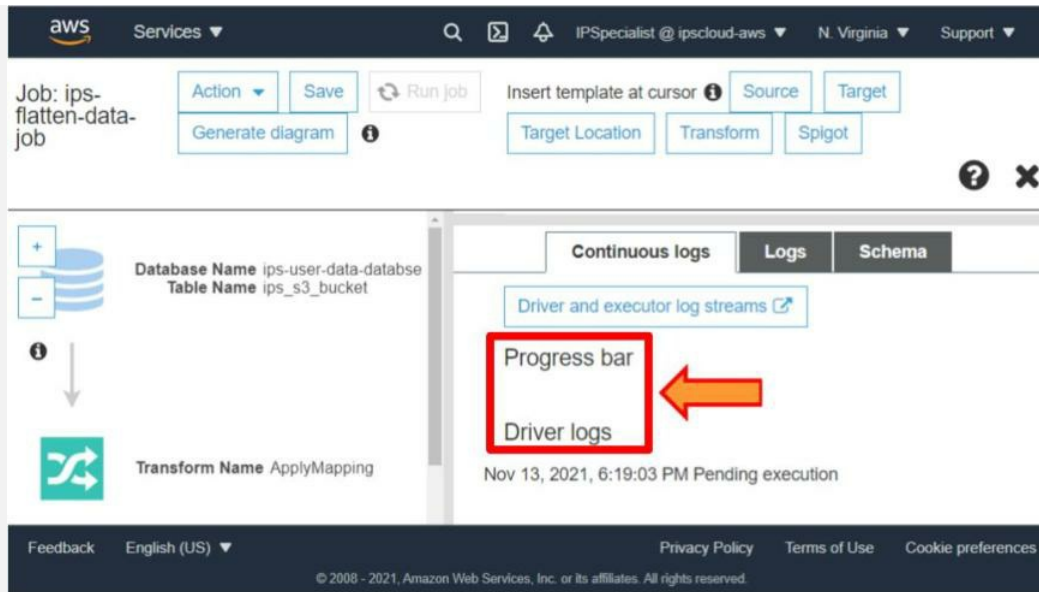
22. Click on the **Run Job** button.



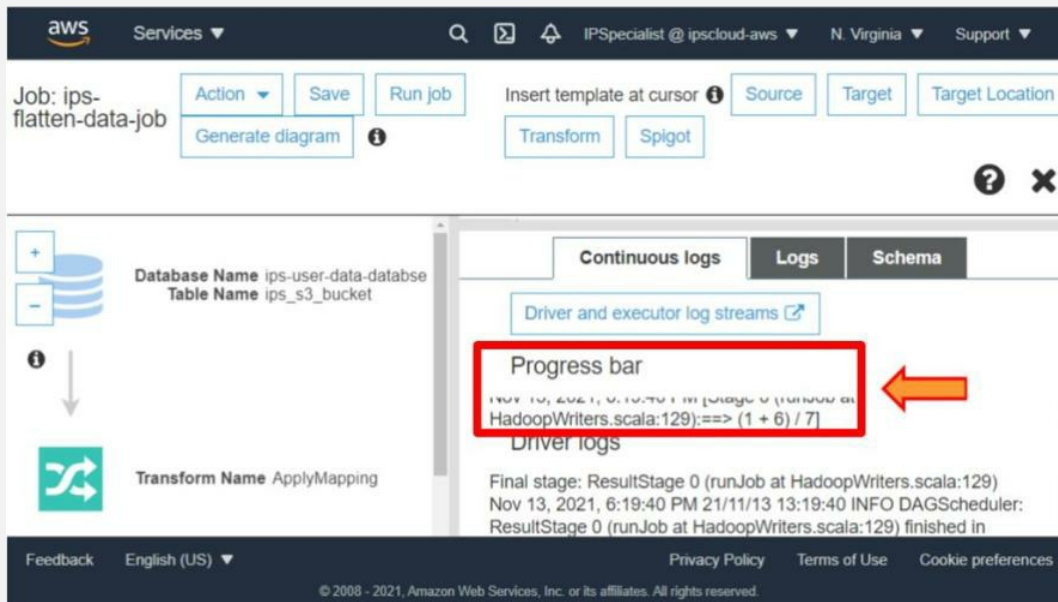
23. Again, click on the **Run Job** button.



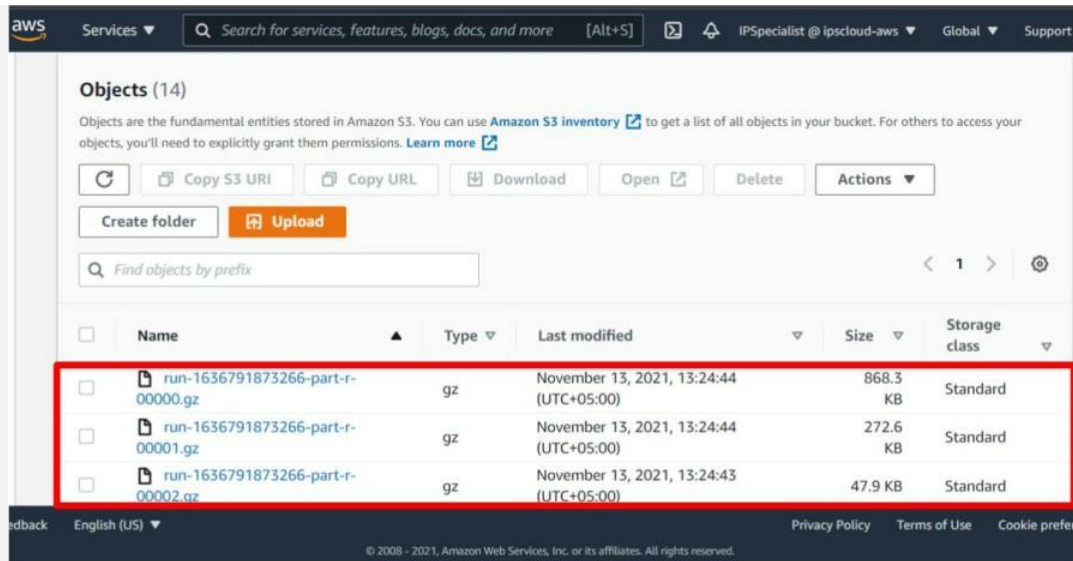
24. It will take a few minutes as you see the **Progress bar** and **Driver logs**.



25. The **Progress bar** finishes. This means that successfully run and created the AWS Glue job.

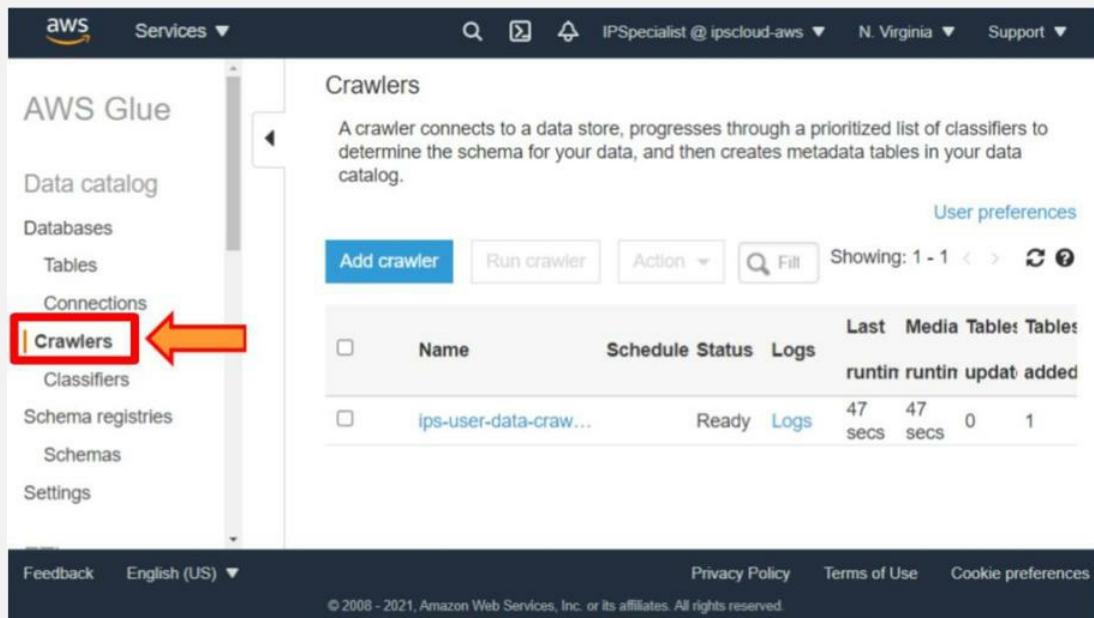


26. Go to the **S3 dashboard**; you will see the compressed files.

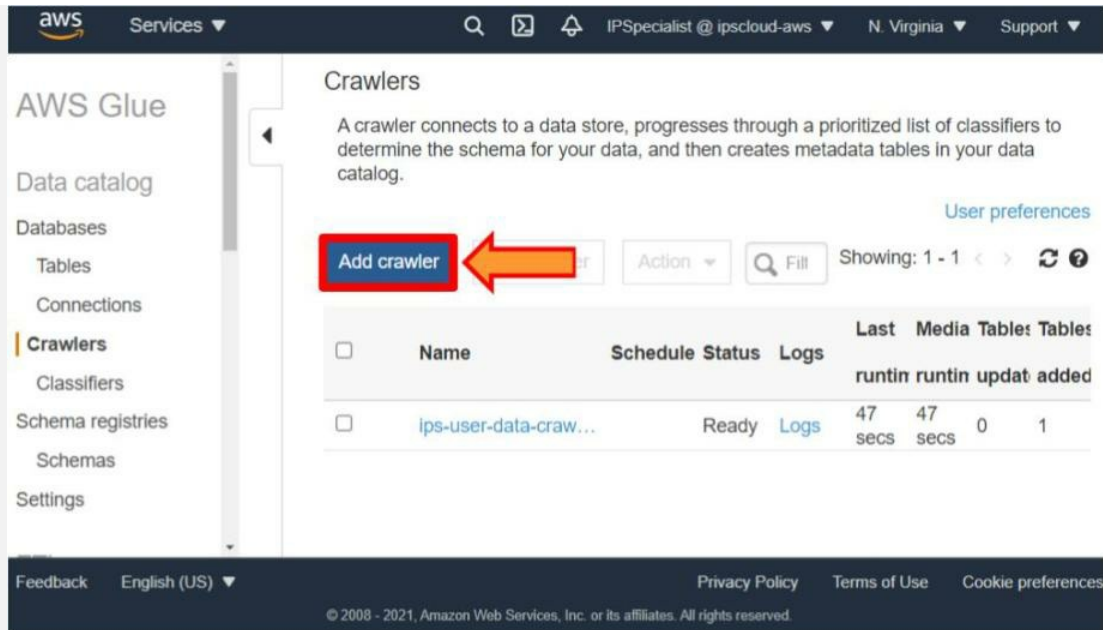


## Step 2: Create AWS Glue Crawler

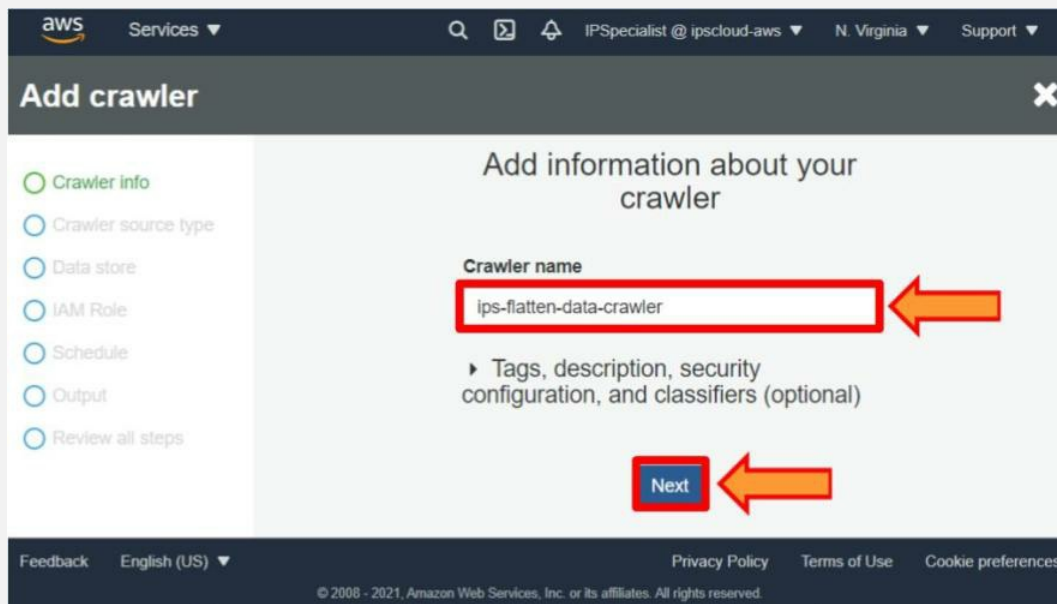
1. Navigate back to the **AWS Glue** tab.
2. Click on **Crawlers** from the left-hand side.



3. Click on the **Add Crawler** button.



4. Give a name **ips-flatten-data-crawler**. Then, click on the **Next** button.



5. Leave everything as default, scroll down and click on the **Next** button.

aws Services IPSpecialist @ ipscloud-aws N. Virginia Support

## Add crawler

- ✓ Crawler info  
ips-flatten-data-crawler
- Crawler source type
- Data store
- IAM Role
- Schedule
- Output
- Review all steps

☒ Crawl all folders  
Crawl all folders again with every subsequent crawl.

☐ Crawl new folders only  
Only Amazon S3 folders that were added since the last crawl will be crawled. If the schemas are compatible, new partitions will be added to existing tables.

☐ Crawl changed folders identified by Amazon S3 Event Notifications  
Rely on Amazon S3 events to control what folders to crawl.

Back Next

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

6. Select the **S3** data store.

aws Services IPSpecialist @ ipscloud-aws N. Virginia Support

## Add crawler

- ✓ Crawler info  
ips-flatten-data-crawler
- ✓ Crawler source type
- Data store
- IAM Role
- Schedule
- Output
- Review all steps

### Add a data store

Choose a data store

S3

Connection

Select a connection

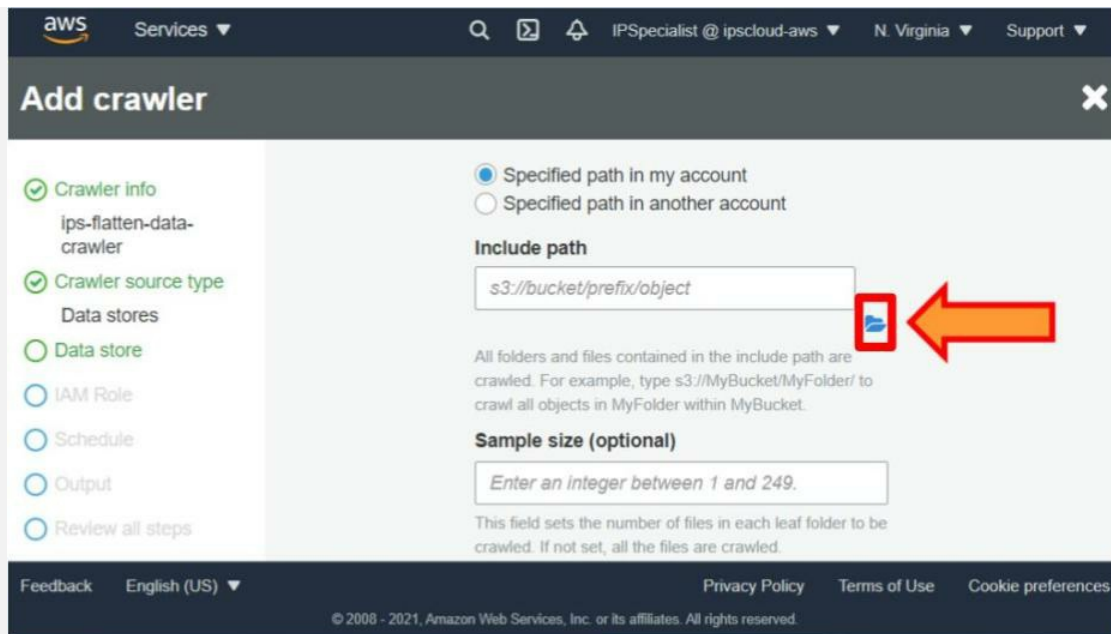
Optionally include a Network connection to use with this S3 target. Note that each crawler is limited to one Network connection so any future S3 targets will also use the same connection (or none, if left blank).

Add connection

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

7. Click on the **Folder** icon button.



aws Services IPSpecialist @ ipsccloud-aws N. Virginia Support

## Add crawler

☒ Specified path in my account  
☐ Specified path in another account

**Include path**

s3://bucket/prefix/object

All folders and files contained in the include path are crawled. For example, type s3://MyBucket/MyFolder/ to crawl all objects in MyFolder within MyBucket.

**Sample size (optional)**

Enter an integer between 1 and 249.

This field sets the number of files in each leaf folder to be crawled. If not set, all the files are crawled.

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

8. Select the **ips-transformed-data** folder in the **ips-s3-bucket**. Then, click on the **Select** button.



aws Services Support

## Add crawler

### Choose S3 path

☒ S3

- ☐ aws-glue-scripts-644738277497-us-east-1
- ☐ aws-glue-temporary-644738277497-us-east-1
- ☒ ips-s3-bucket
  - ☒ ips-transformed-data
  - ☐ User-data-001860f5-d0a6-4958-ba6f-498600deeb0f.json
  - ☐ User-data-0bf115ab-8c82-41fd-a961-6d71833ab35b.json
  - ☐ User-data-10b04868-6cd8-407f-9e1b-5997380bed7a.json
  - ☐ User-data-15078f3e-62d3-44cb-813f-0d339a487ccc.json
  - ☐ User-data-b3e7891f-36c9-4c15-96d7-613a6ebca61f.json
  - ☐ User-data-c5104e53-62f4-4973-b2c5-0f50a66ab474.json
  - ☐ User-data-e967772a-517b-4820-8852-cb3061a3aa1e.json

Select

Feedback Eng... Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

9. Click on the **Next** button.

aws Services

Search for services, features, blogs, docs, and r [Alt+S]

IPSpecialist @ ipscloud-aws N. Virginia Support

## Add crawler

- ✓ Crawler info  
ips-flatten-data-crawler
- ✓ Crawler source type  
Data stores
- Data store
- IAM Role
- Schedule
- Output
- Review all steps

All folders and files contained in the include path are crawled. For example, type `s3://MyBucket/MyFolder/` to crawl all objects in MyFolder within MyBucket.

**Sample size (optional)**

Enter an integer between 1 and 249.

This field sets the number of files in each leaf folder to be crawled. If not set, all the files are crawled.

▸ Exclude patterns (optional)

Back Next

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

10. Select **No** when prompted to add another data source. Then click on the **Next** button.

aws Services

Search for services, features, blogs, docs, and r [Alt+S]

IPSpecialist @ ipscloud-aws N. Virginia Support

## Add crawler

- ✓ Crawler info  
ips-flatten-data-crawler
- ✓ Crawler source type  
Data stores
- Data store
- IAM Role
- Schedule
- Output
- Review all steps

**Add another data store**

☐ Yes ☒ No

Back Next

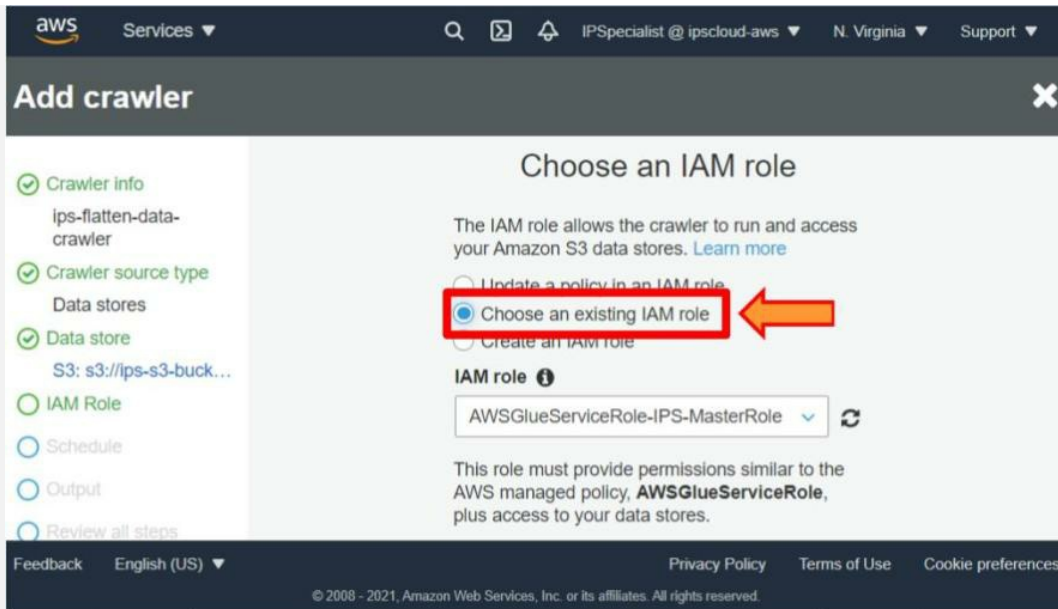
**Chosen data stores**

S3: s3://ips-s3-bucket...

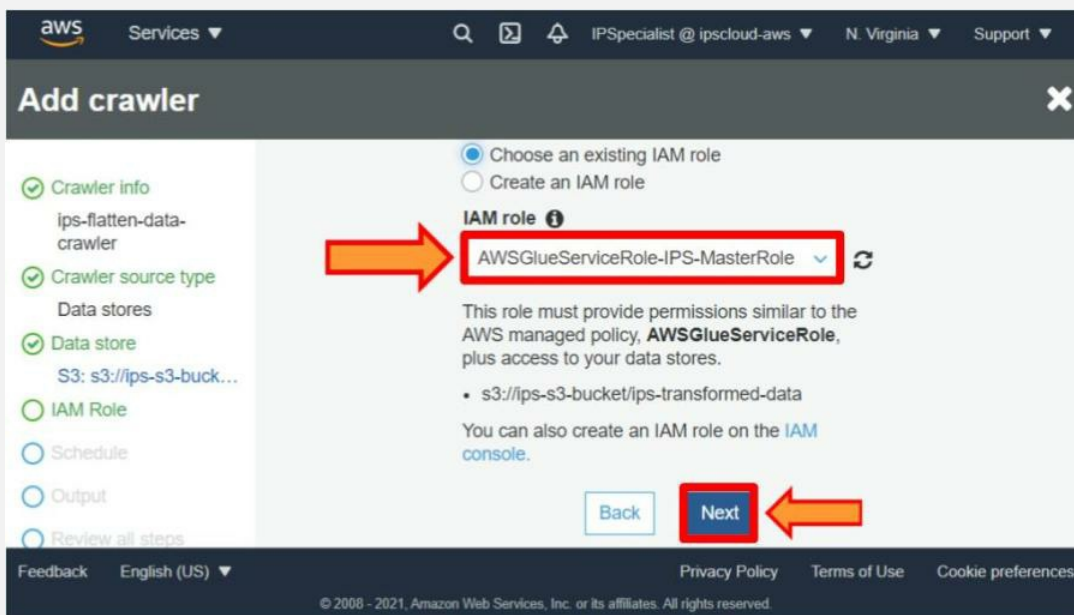
Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

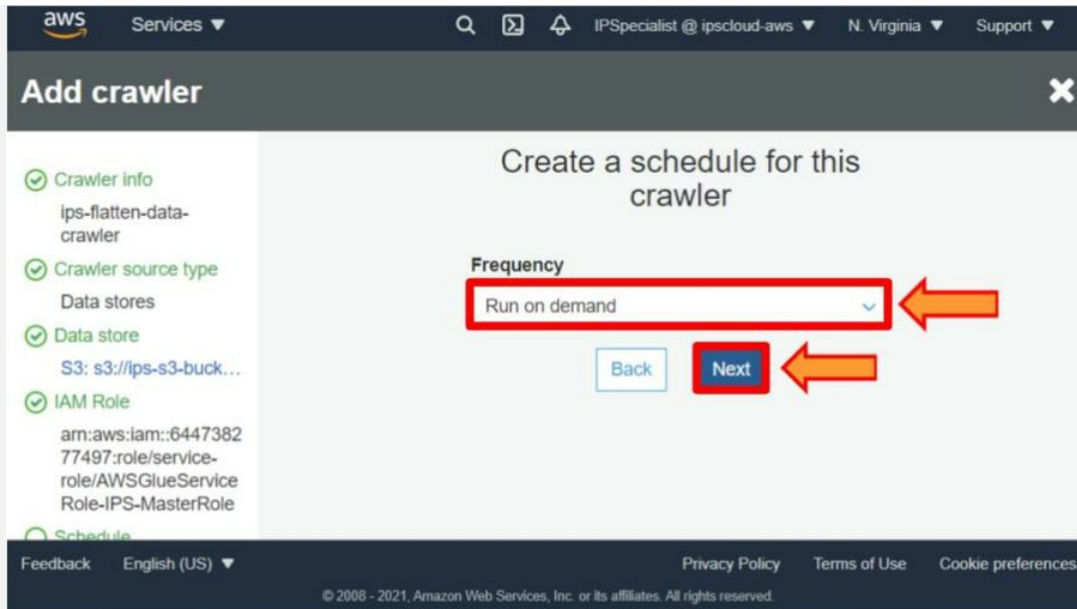
11. Select the **Choose an existing IAM role**.



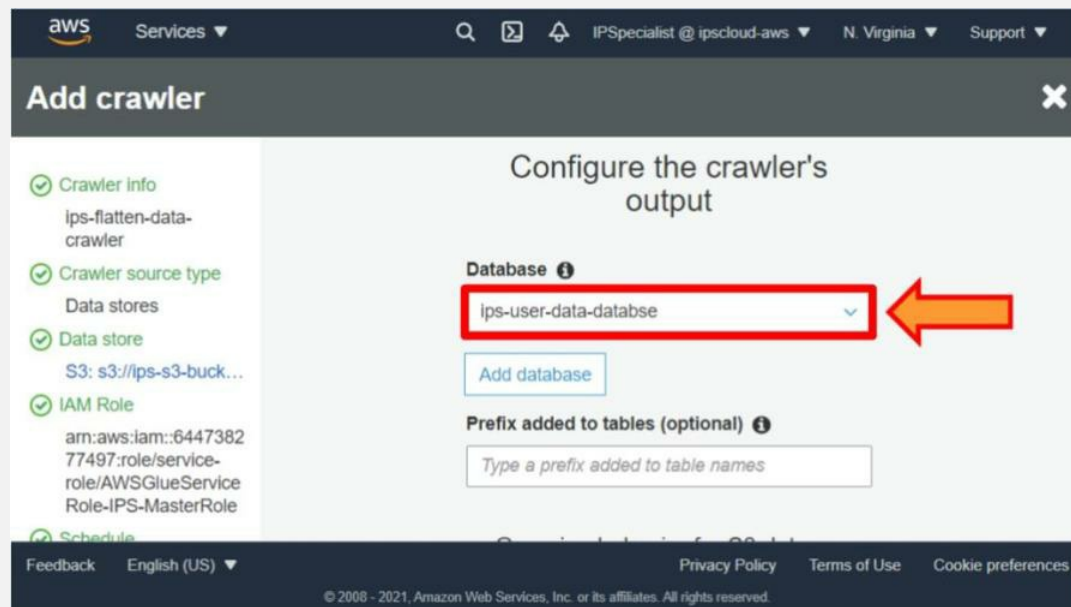
12. Select the **IAM Role** that you previously created in the demo. Then, click on the **Next** button.



13. Select the Frequency **Run on demand**. Then, click on the **Next** button.



14. Select the **ips-user-data-database**.



15. Click on the **Next** button.

aws Services IPSpecialist @ ipscloud-aws N. Virginia Support

## Add crawler

✓ Crawler info  
ips-flatten-data-crawler

✓ Crawler source type  
Data stores

✓ Data store  
S3: s3://ips-s3-buck...

✓ IAM Role  
arn:aws:iam::644738277497:role/service-role/AWSGlueServiceRole-IPS-MasterRole

✓ Schedule

Add database

Prefix added to tables (optional) ⓘ  
Type a prefix added to table names

► Grouping behavior for S3 data (optional)

► Configuration options (optional)

Back Next

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

16. Click on the **Finish** button.

aws Services IPSpecialist @ ipscloud-aws N. Virginia Support

## Add crawler

✓ Crawler info  
ips-flatten-data-crawler

✓ Crawler source type  
Data stores

✓ Data store  
S3: s3://ips-s3-buck...

✓ IAM Role  
arn:aws:iam::644738277497:role/service-role/AWSGlueServiceRole-IPS-MasterRole

✓ Schedule

Database  
ips-user-data-database

Prefix added to tables (optional)

Create a single schema for each S3 path false

Table level (optional)

► Configuration options

Back Finish

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

17. Select the **ips-flatten-data-crawler**.

**aws** Services ▾

Search [ ] [ ] IPSpecialist @ ipscloud-aws ▾ N. Virginia ▾ Support ▾

### AWS Glue

Data catalog

Databases

Tables

Connections

**Crawlers**

Classifiers

Schema registries

Schemas

Settings

#### Crawlers

A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

[Add crawler](#) [Run crawler](#) [Action ▾](#) [Filter](#) Showing: 1 - 2 < > ↺ ? [User preferences](#)

| <input type="checkbox"/>            | Name                    | Schedule | Status | Logs                 | Last run | Media run | Tables updated | Tables added |
|-------------------------------------|-------------------------|----------|--------|----------------------|----------|-----------|----------------|--------------|
| <input checked="" type="checkbox"/> | ips-flatten-data-cra... |          | Ready  |                      | 0 secs   | 0 secs    | 0              | 0            |
| <input type="checkbox"/>            | ips-user-data-craw...   |          | Ready  | <a href="#">Logs</a> | 47 secs  | 47 secs   | 0              | 1            |

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

18. Click on the **Run Crawler** button.

**aws** Services ▾

Search [ ] [ ] IPSpecialist @ ipscloud-aws ▾ N. Virginia ▾ Support ▾

### AWS Glue

Data catalog

Databases

Tables

Connections

**Crawlers**

Classifiers

Schema registries

Schemas

Settings

#### Crawlers

A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

[Add crawler](#) [Run crawler](#) [Action ▾](#) [Filter](#) Showing: 1 - 2 < > ↺ ? [User preferences](#)

| <input type="checkbox"/>            | Name                    | Schedule | Status | Logs                 | Last run | Media run | Tables updated | Tables added |
|-------------------------------------|-------------------------|----------|--------|----------------------|----------|-----------|----------------|--------------|
| <input checked="" type="checkbox"/> | ips-flatten-data-cra... |          | Ready  |                      | 0 secs   | 0 secs    | 0              | 0            |
| <input type="checkbox"/>            | ips-user-data-craw...   |          | Ready  | <a href="#">Logs</a> | 47 secs  | 47 secs   | 0              | 1            |

Feedback English (US) ▾ Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

19. After running finish, one table is added.

AWS Glue

Data catalog

Databases

Tables

Connections

Crawlers

Classifiers

Schema registries

Schemas

Settings

ETL

AWS Glue Studio

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Crawlers

A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

Crawler "ips-flatten-data-crawler" completed and made the following changes: 1 tables created, 0 tables updated. See the tables created in database [ips-user-data-database](#).

[User preferences](#)

[Add crawler](#) [Run crawler](#) [Action](#)  Showing: 1 - 2

| <input type="checkbox"/> | Name                                     | Schedule | Status | Logs                 | Last runtime | Median runtime | Tables updated | Tables added |
|--------------------------|------------------------------------------|----------|--------|----------------------|--------------|----------------|----------------|--------------|
| <input type="checkbox"/> | <a href="#">ips-flatten-data-crawler</a> |          | Ready  | <a href="#">Logs</a> | 1 min        | 1 min          | 0              | 1            |
| <input type="checkbox"/> | <a href="#">ips-user-data-crawler</a>    |          | Ready  | <a href="#">Logs</a> | 47 secs      | 47 secs        | 0              | 1            |

20. Click on **Tables** from the left-hand side menu.

AWS Glue

Data catalog

Databases

Tables

Connections

Crawlers

Classifiers

Schema registries

Schemas

Settings

Feedback English (US) Privacy Policy Terms of Use Cookie preferences

Tables

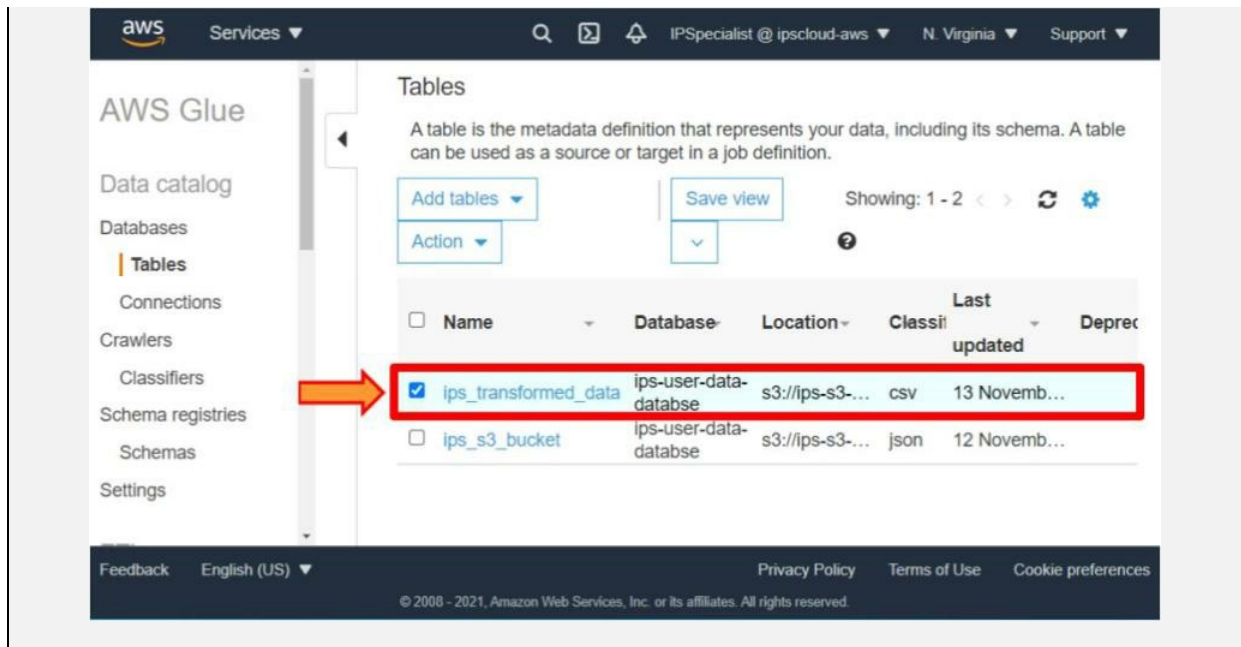
A table is the metadata definition that represents your data, including its schema. A table can be used as a source or target in a job definition.

[Add tables](#) [Save view](#) Showing: 1 - 2

[Action](#) [?](#)

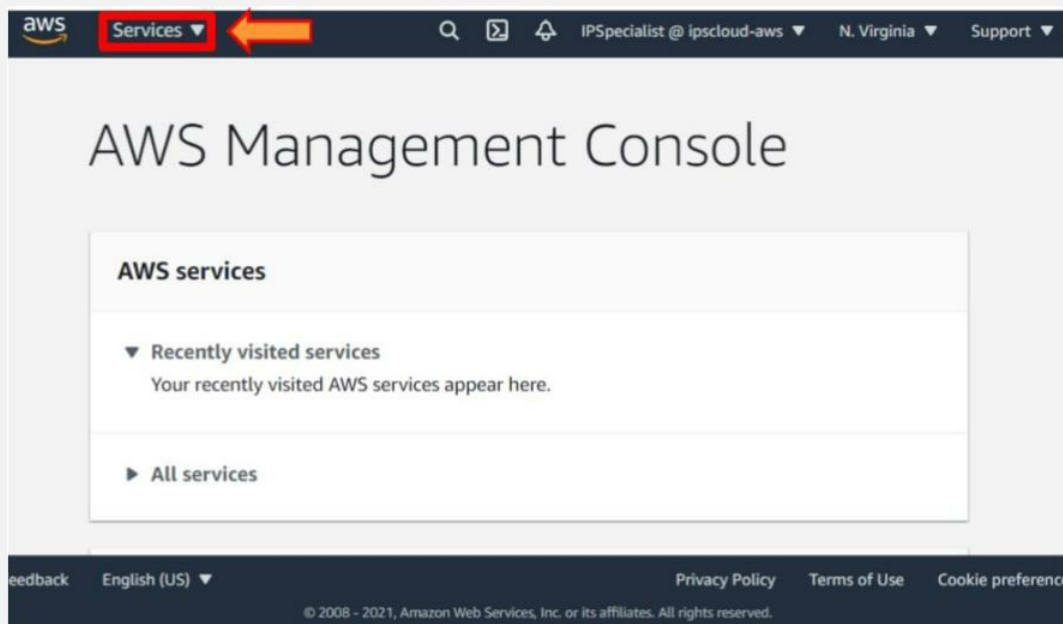
| <input type="checkbox"/> | Name                                 | Database               | Location        | Classi | Last updated | Deprec |
|--------------------------|--------------------------------------|------------------------|-----------------|--------|--------------|--------|
| <input type="checkbox"/> | <a href="#">ips_transformed_data</a> | ips-user-data-database | s3://ips-s3-... | csv    | 13 Novemb... |        |
| <input type="checkbox"/> | <a href="#">ips_s3_bucket</a>        | ips-user-data-database | s3://ips-s3-... | json   | 12 Novemb... |        |

21. You will see the **ips-transformed-data** table.

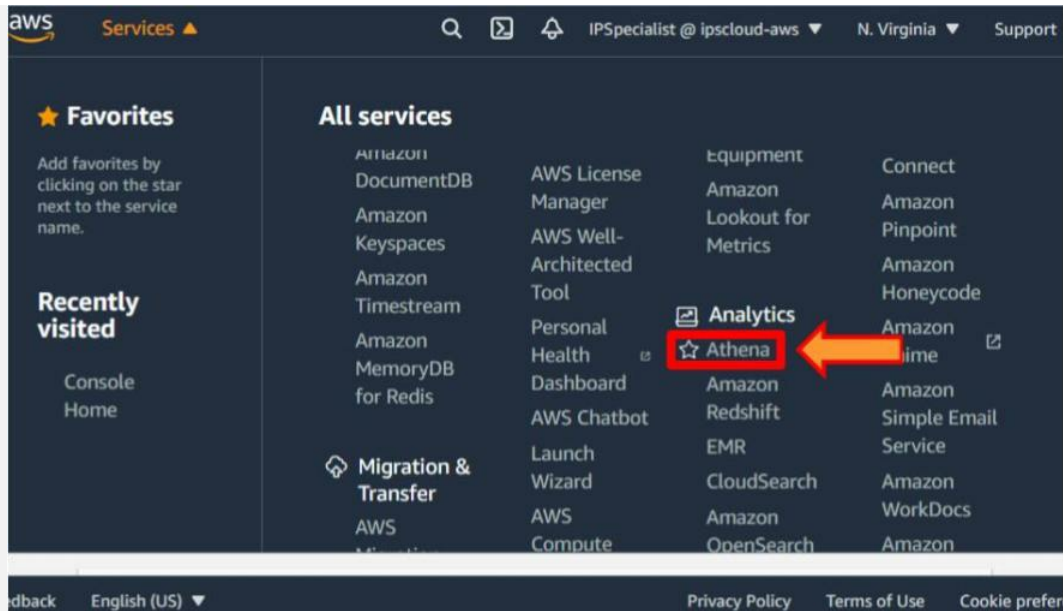


### Step 3: Execute SQL Quieres on AWS Athena

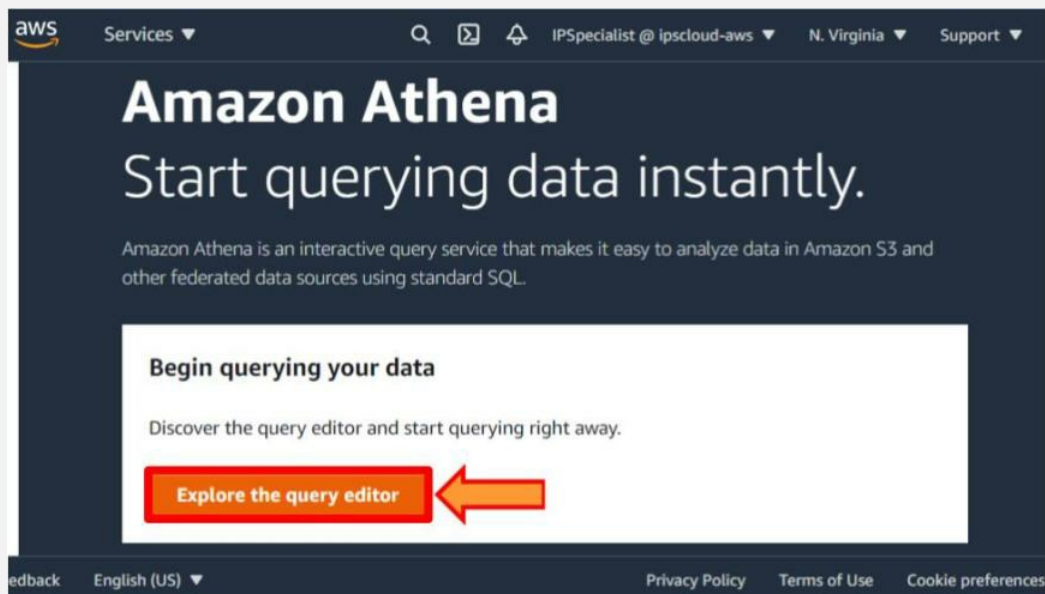
1. Click on **Services**.



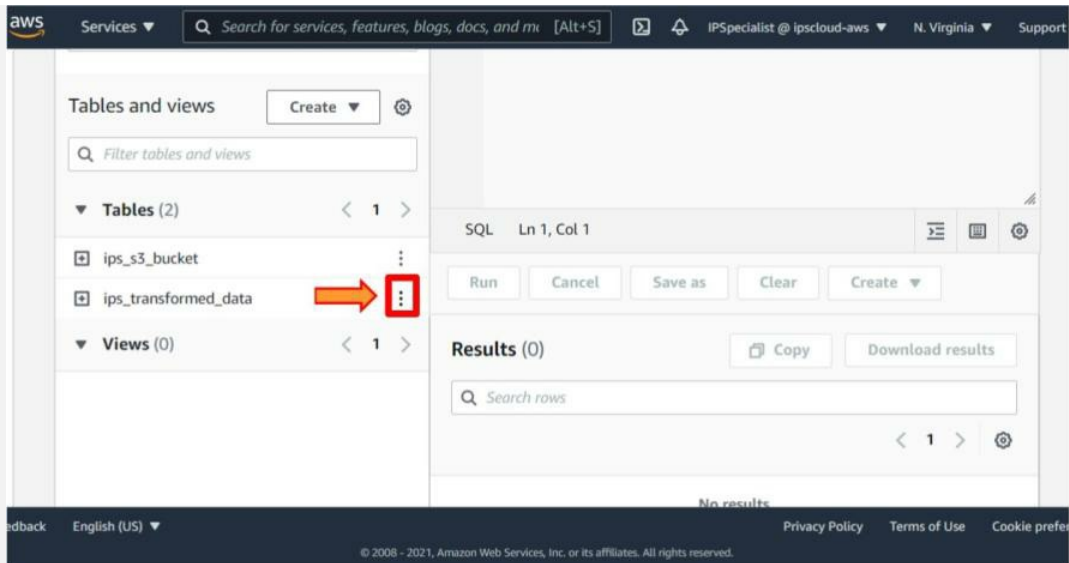
2. Select **Athena** from **Analytics**.



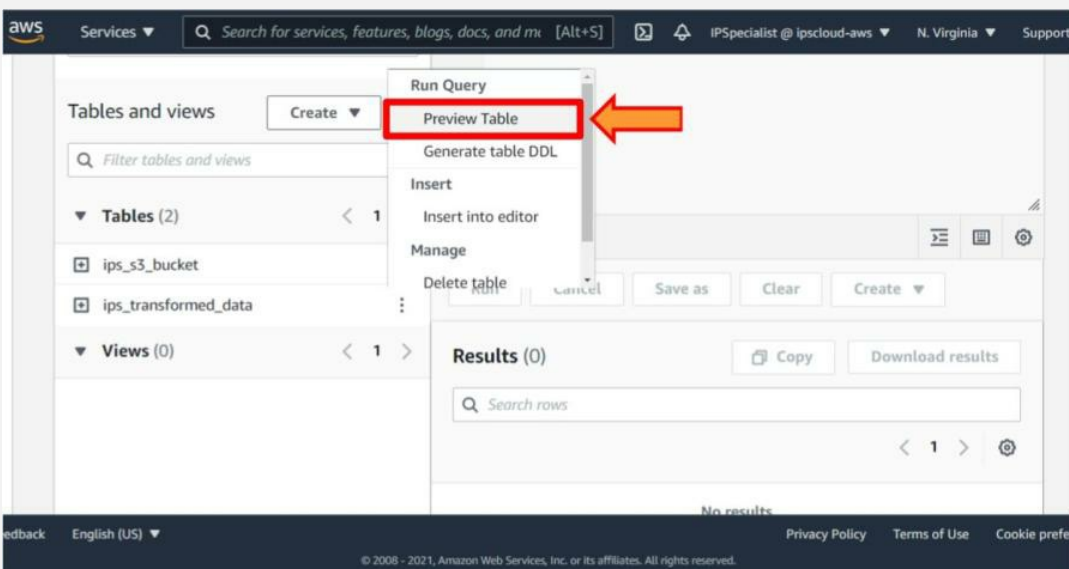
3. Click on the **Explore the query editor** button.



4. Click on the **Options** icon of **ips-transformed-data**.



5. Click on **Preview table**.



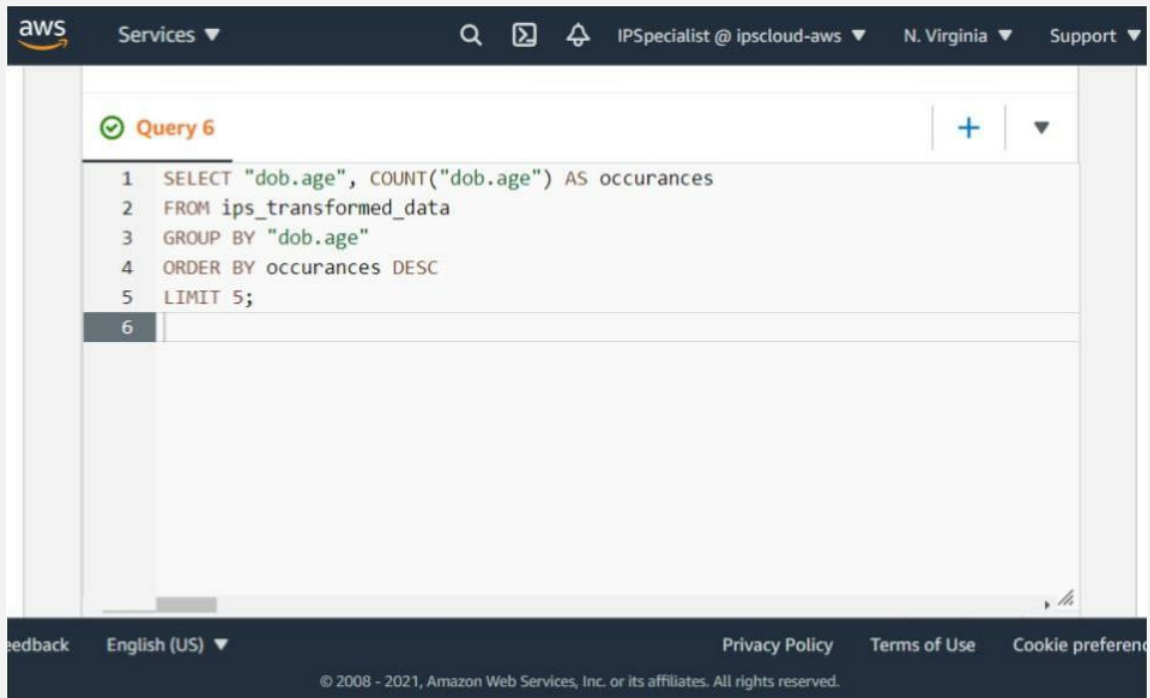
6. After that, you will run the real-time SQL queries and do real-time data analytics.

7. Copy and paste the below query in the query editor.

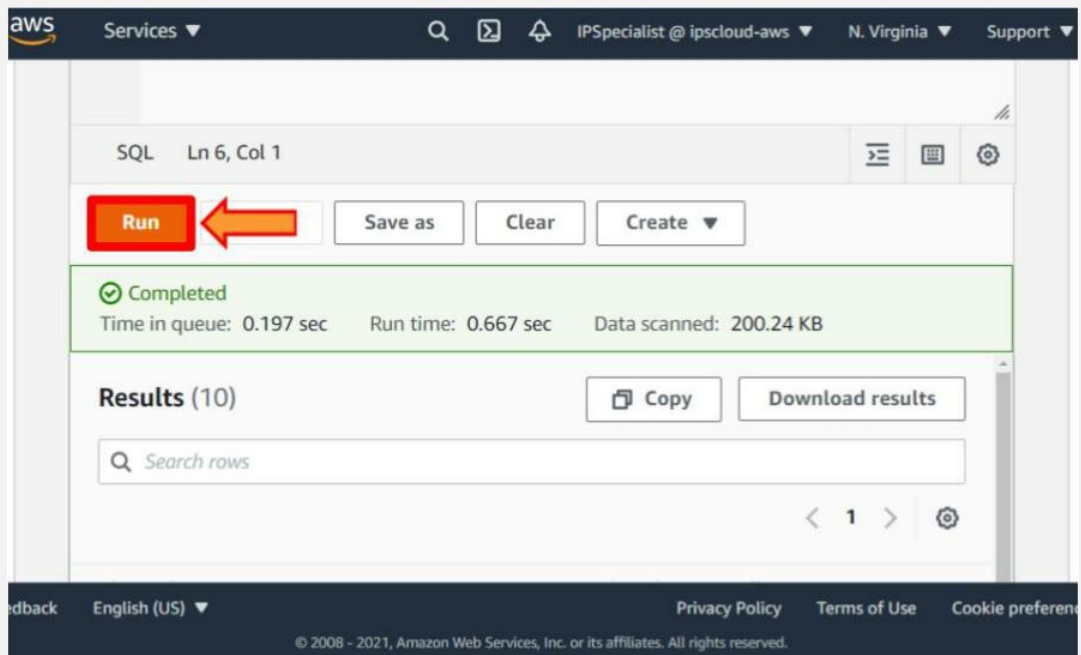
```
SELECT "dob.age", COUNT("dob.age") AS occurrences
FROM ips_transformed_data
GROUP BY "dob.age"
```

ORDER BY occurrences DESC

LIMIT 5;



8. Click on the **Run Query** button.



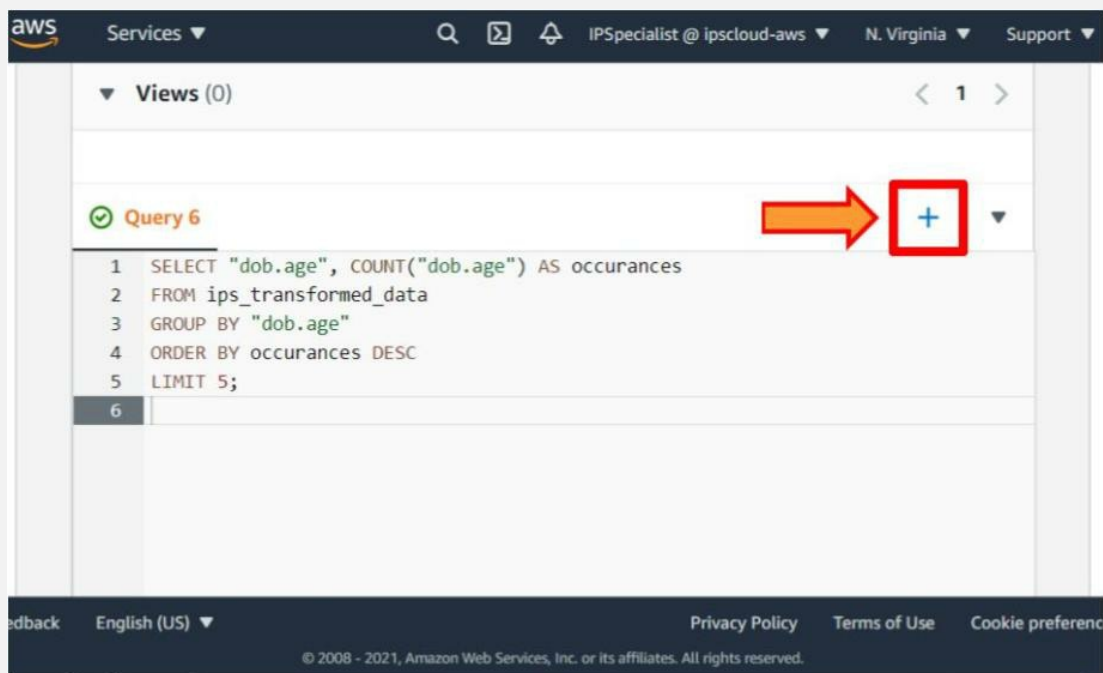
9. Hence, you can see the output.



The screenshot shows the AWS Glue console interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, a search bar, and user information 'IPSpecialist @ ipscloud-aws' in 'N. Virginia'. Below this, the 'Results (5)' section is visible, with 'Copy' and 'Download results' buttons. A search bar for rows is present. The results are displayed in a table with two columns: 'dob.age' and 'occurrences'. The table is highlighted with a red border. The footer includes 'Feedback', 'English (US)', 'Privacy Policy', 'Terms of Use', 'Cookie preferences', and a copyright notice '© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.'

| dob.age | occurrences |
|---------|-------------|
| 72      | 1554        |
| 67      | 1328        |
| 54      | 1278        |
| 50      | 1246        |
| 68      | 1240        |

10. Click on the + icon to open a new tab on the query editor.



The screenshot shows the AWS Glue console interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, a search bar, and user information 'IPSpecialist @ ipscloud-aws' in 'N. Virginia'. Below this, the 'Views (0)' section is visible. A red arrow points to a blue '+' icon in the top right corner of the query editor area, which is also highlighted with a red square. The query editor shows a SQL query for 'Query 6' with the following text:

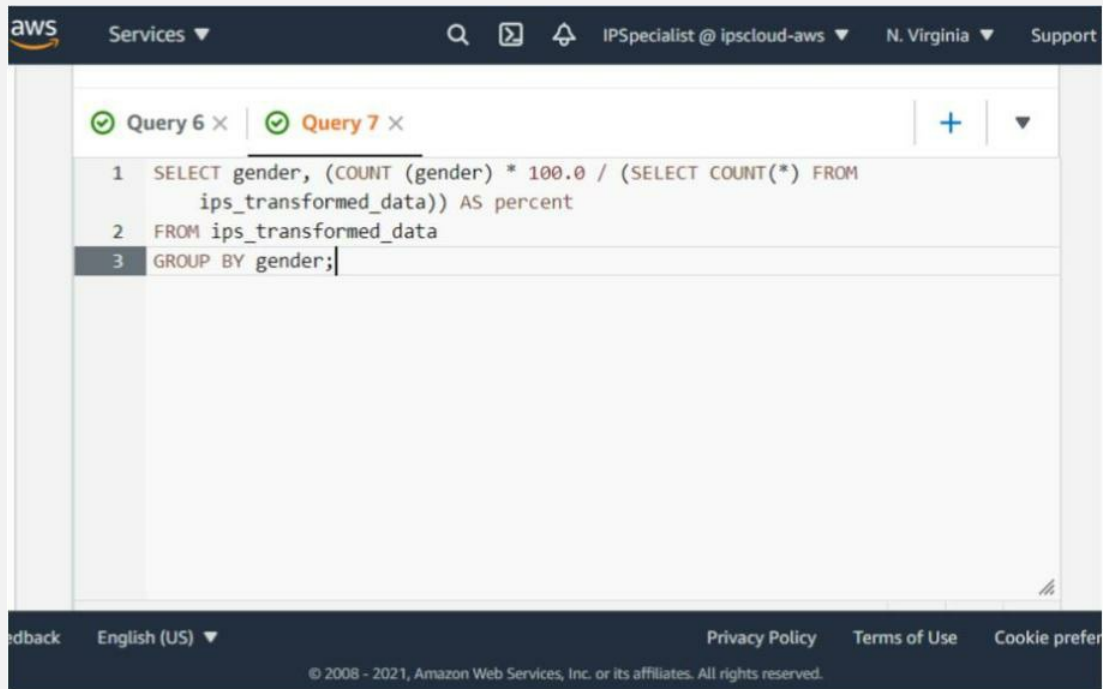
```
1 SELECT "dob.age", COUNT("dob.age") AS occurrences
2 FROM ips_transformed_data
3 GROUP BY "dob.age"
4 ORDER BY occurrences DESC
5 LIMIT 5;
6
```

The footer includes 'Feedback', 'English (US)', 'Privacy Policy', 'Terms of Use', 'Cookie preferences', and a copyright notice '© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.'

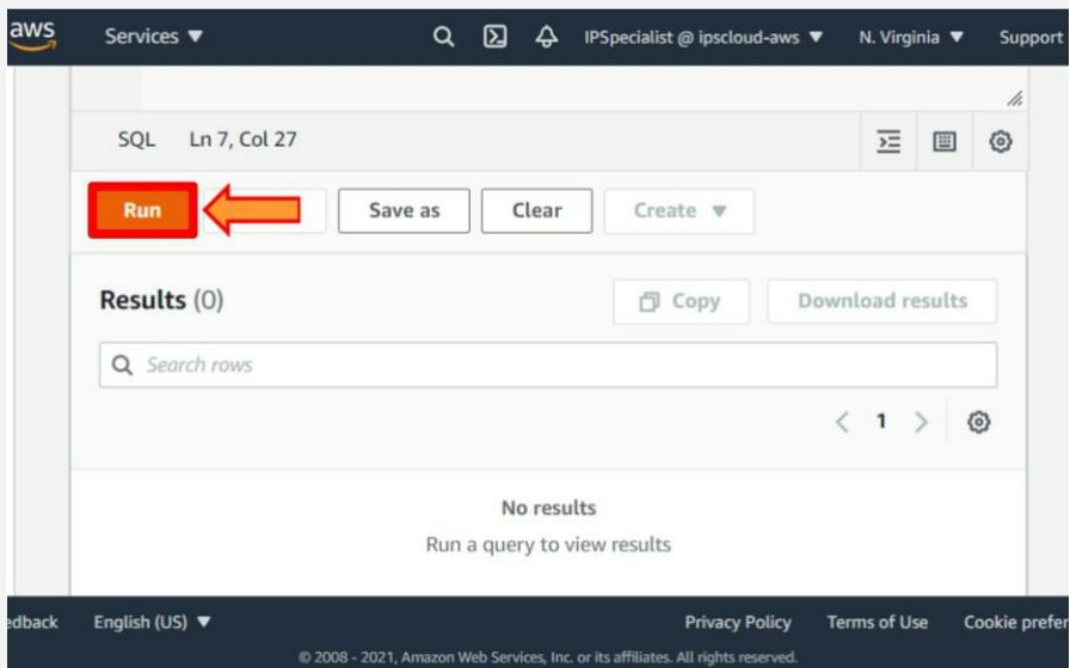
11. Copy and paste the below query in the query editor.

```
SELECT gender, (COUNT (gender) * 100.0 / (SELECT
COUNT(*) FROM ips_transformed_data)) AS percent
```

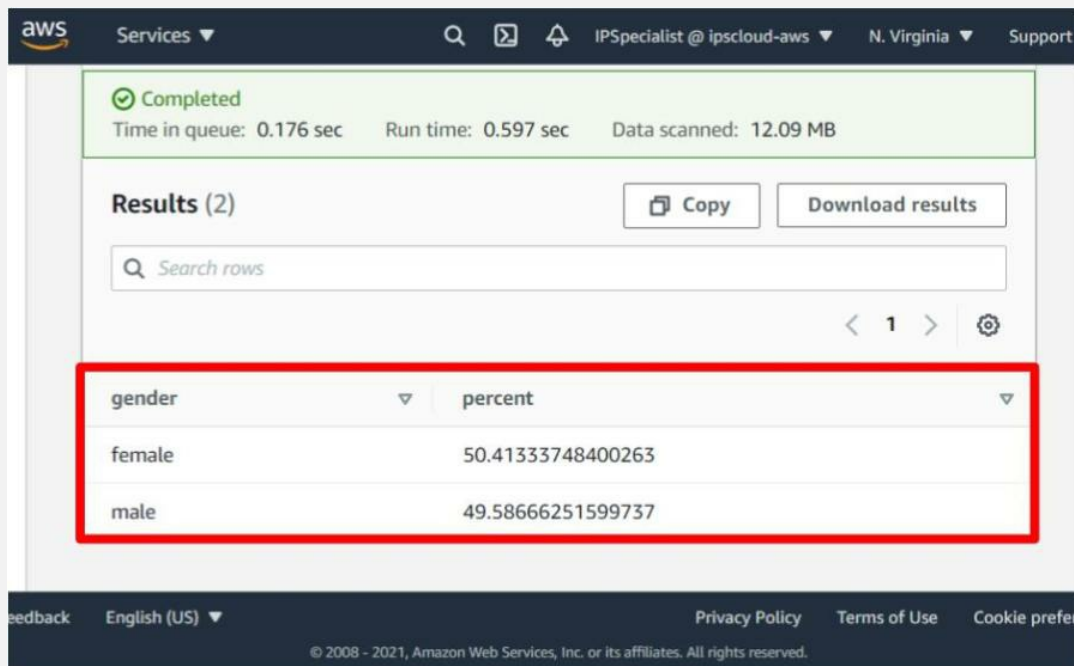
```
FROM ips_transformed_data  
  
GROUP BY gender;
```



12. Click on the **Run Query** button.



13. Hence, you can see the output.



## When To Use Athena

### S3 Select And Glacier Select

- **S3 Select** – Use SQL statements to filter the contents of S3 objects and retrieve just the subset of data you need.
- **Glacier Select** - Use SQL statements directly on your data in S3 Glacier without restoring data to a more frequently accessible tier.

### Overview Of Similar Services

| Redshift       | EMR           | Athena | S3 Select | Glacier Select |
|----------------|---------------|--------|-----------|----------------|
| Fats querying, | Simple to run |        |           | Queries        |

|                                                                     |                                                                   |                                                       |                                                             |                             |  |
|---------------------------------------------------------------------|-------------------------------------------------------------------|-------------------------------------------------------|-------------------------------------------------------------|-----------------------------|--|
| enterprise reporting, and BI workloads                              | distributed processing frameworks like Hadoop, Spark, and Presto  | Easily run ad-hoc queries for data in S3              | Retrieves a subset of data from an object                   | Glacier data within minutes |  |
| Very complex SQL (multiple joints and sub-queries)                  | Flexible enough to run custom applications and code               | No need to set up and manage servers                  | Offload filtering data in S3 instead of your application    | Use standard SQL statements |  |
| Bringing data from many different data sources into a common format | You define the compute, memory, and storage to optimize workloads | No need to format data                                | Simple SQL expressions                                      | No need to restore to S3    |  |
| Storing data from long periods                                      |                                                                   | It can be used with EMR and Redshift as an integrated | Limited data formats CSV, JSON, Parquet (GZIP and BZIP2 for |                             |  |
| Build Sophisticated business                                        |                                                                   |                                                       |                                                             |                             |  |

|                                    |  |                 |                  |  |
|------------------------------------|--|-----------------|------------------|--|
| reports from<br>historical<br>data |  | data<br>catalog | CSV and<br>JSON) |  |
|------------------------------------|--|-----------------|------------------|--|

*Table 9-05: Overview of Similar Services*

## Comparing Athena To Other Services

### *Redshift and Athena*

| Redshift                                                                                                                                                                 | Athena                                                                                                                                       |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>• Needs to pull data from various sources</li> <li>• We have built sophisticated business reports with complex queries</li> </ul> | <ul style="list-style-type: none"> <li>• Data must be stored in S3</li> <li>• Run quick queries for troubleshooting and analytics</li> </ul> |

*Table 9-06: Redshift and Athena*

### *EMR and Athena*

| EMR                                                                                                                                                                | Athena                                                                                                                                                                                              |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>• It goes far beyond just running queries</li> <li>• Custom code to process and analyze massively large datasets</li> </ul> | <ul style="list-style-type: none"> <li>• Less of a large-scale data processing solution</li> <li>• More of an ad-hoc querying tool</li> <li>• Serverless – no need to manage the cluster</li> </ul> |

Table 9-07: EMR and Athena

### ***S3 Select, Glacier Select, and Athena***

| <b>S3 Select And Glacier Select</b>                                                                                                                                                                | <b>Athena</b>                                                                                                                                                                                                          |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><li>• No need to query a single object in S3</li><li>• It can return just a few bytes in a large object</li><li>• Query in-place, then return the data</li></ul> | <ul style="list-style-type: none"><li>• Need to query all results in a bucket</li><li>• More powerful, fully interactive query service</li><li>• Complex analysis, large joins, window functions, and arrays</li></ul> |

Table 9-08 Select, Glacier Select, and Athena

## **QuickSight Visualizations and Dashboards**

### **Amazon QuickSight**

Amazon QuickSight helps to create visualization and dashboard with your data. Business intelligence tool makes it easy to create visualization and dashboard from your data. Simply point QuickSight at your input data source to start creating visualizations.

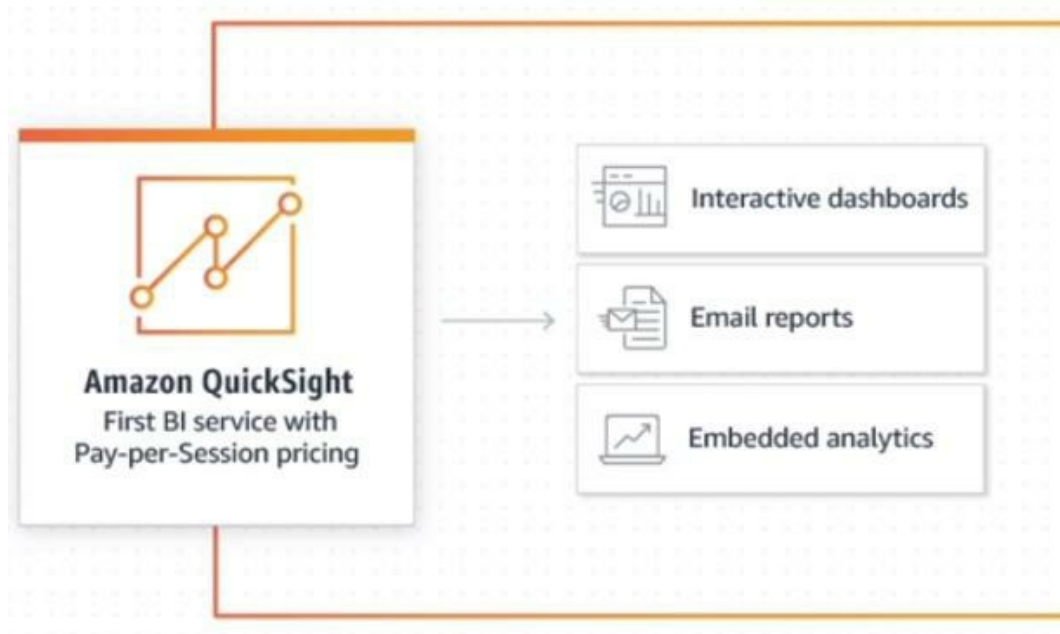


Figure 9-14: Amazon QuickSight

## **How QuickSight Works?**

Once we sign up for a QuickSight account, we need to start with connecting our data. This data can be AWS data. It can be applications either running in AWS or on some third-party websites or servers, it can be running on-premises, or we can use files, text files, CSV files, and JSON files. We can use relational data, where all of these services are included right out of the box whenever you start a QuickSight account to connect Athena, Aurora, Redshift, Redshift Spectrum, S3, S3 and AWS IOT analytics. We have running inside the AWS or outside of AWS that we can connect with QuickSight like Apache spark, MariaDB, Microsoft SQL Server, MySQL PostgreSQL, Presto, Snowflake, and Teradata. We can import files right into CSV and TSV files, ELF and CLF files, JSON, and even Excel spreadsheets. We can use zip and Gzip files that reside in S3. We also have software as a service data that we can connect into QuickSight, such as Jira data from ServiceNow, Adobe Analytics data from GitHub, Salesforce, and data from Twitter.

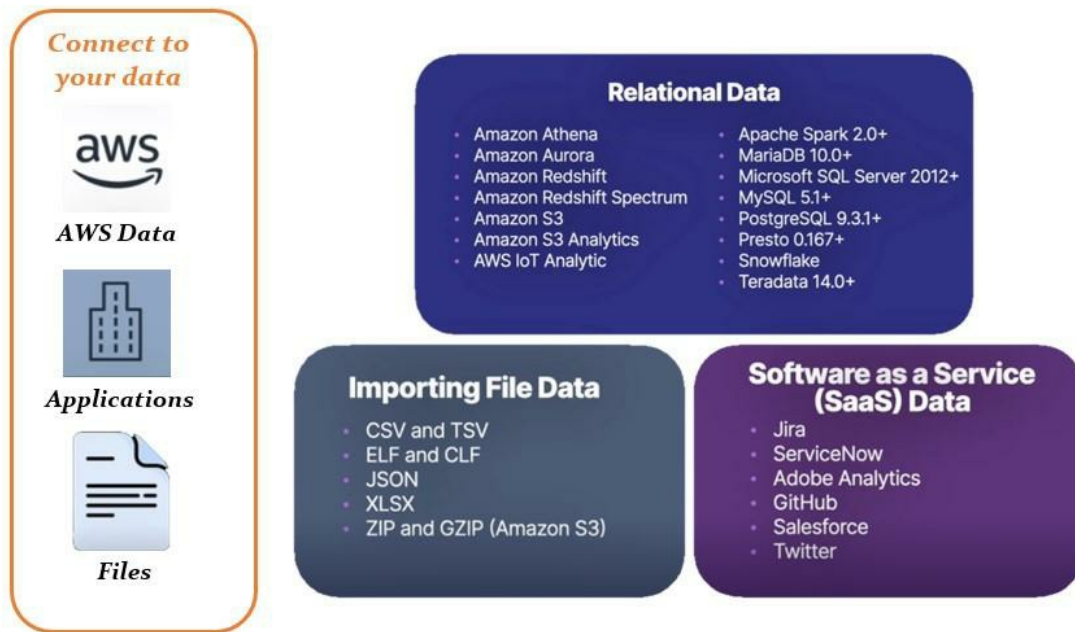


Figure 9-15: QuickSight Working

Once we have decided what data we want to load into QuickSight or start visualizing, we then load that data into a place called SPICE (Super-Fast Parallel In-Memory Calculation Engine). We want to use SPICE because it is engineered to perform advanced calculations and serve our data rapidly. We do not have to make API calls or query our data each time we create a visualization.

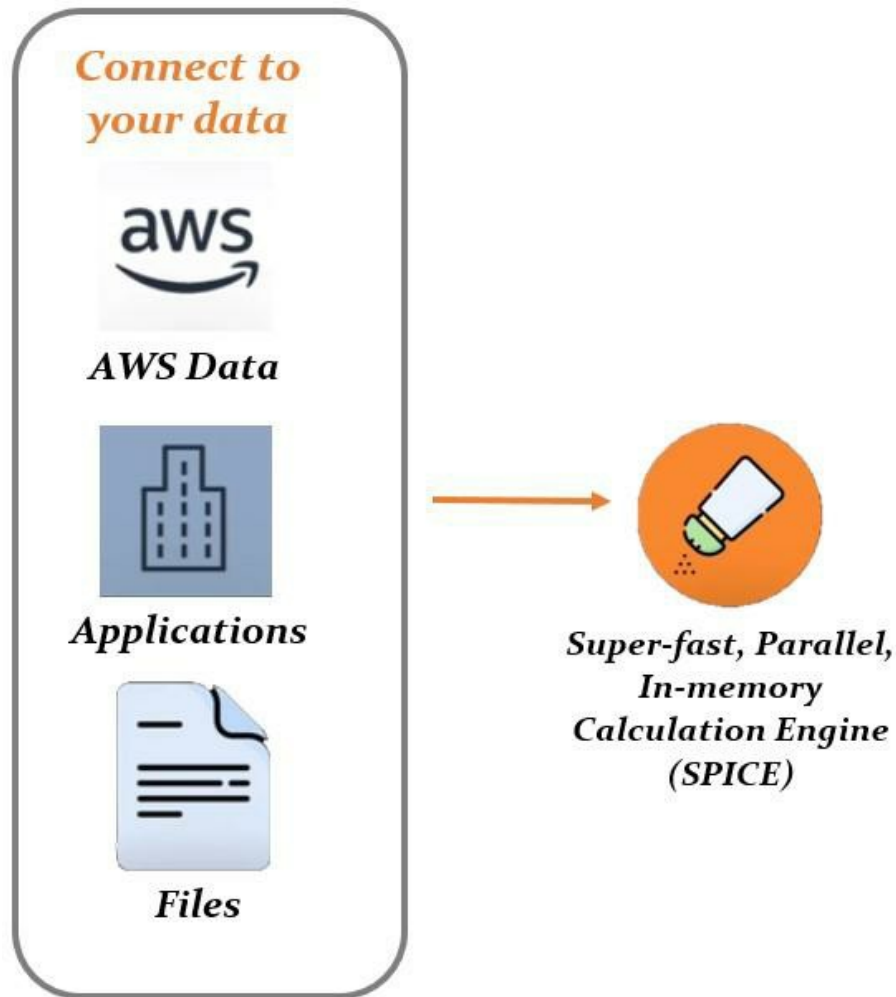


Figure 9-16: QuickSight Working

## **Spice**

- Analytical queries process faster
- You do not need to wait for a direct query to process
- Data stored in SPICE can be reused multiple times without incurring additional costs
- The SPICE capacity is allocated separately for each AWS region

You can also do direct queries, and once you run these direct queries or take your data from the SPICE engine, you can start examining your data. You can create visualizations and design different charts and

graphs. From there, you can make charts and graphs, create dashboards, and share those dashboards with either other developers with upper management or your end-users. These people need to see analytics about either specific data within your application or detailed data about their applications. A dashboard is like a collection of charts and graphs with insights.

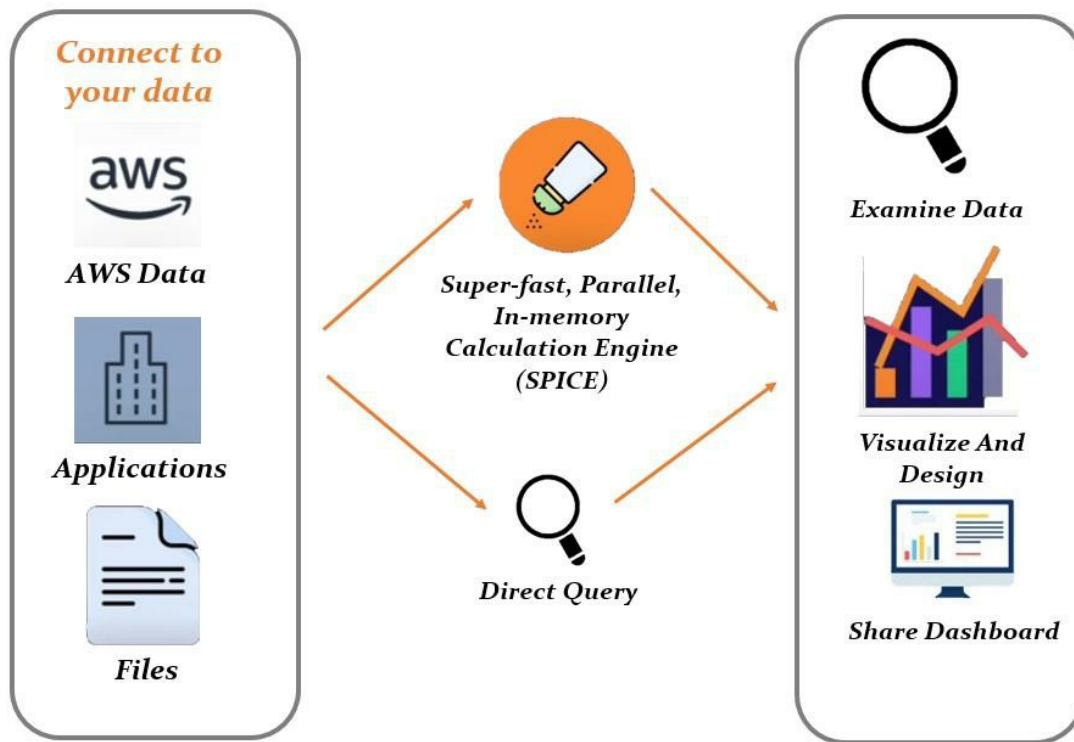


Figure 9-17: QuickSight Working

## Visualization types

- Bar Charts
- Combo Charts
- Donut Charts
- Gauge Charts (Maps)
- Heat Maps
- Histograms
- KPIs
- Line Charts

- Pie Charts
- Scatter Plots
- Tree Maps
- Word Clouds



Figure 9-18: Visualization Types

## QuickSight Dashboards

### *Specify User Access*

When sharing a dashboard, you specify which user has access to it.

### *Dashboard Viewers vs. Owners*

Dashboard viewers have limited access (viewing, filtering, sorting). Owners can share the dashboard and optionally edit and share the analysis.

### *Embedded In Website or Application*

A shared dashboard can be embedded only in a website or app if QuickSight is set up as an Enterprise edition.

#### **EXAM TIP:**

- Amazon QuickSight is a business intelligence (BI) tool that makes it easy to create a visualization and dashboard from your data.
- How QuickSight works – Connect to your data source, load the data into SPICE, and then start the analysis.

- QuickSight Dashboards – Specify which user has access to the dashboard, viewers vs. owners, and embed the dashboard into a website or other applications.

## QuickSight Security and Authentication

### QuickSight Data Encryption

#### *Encryption at Rest (Enterprise edition only)*

All of the metadata and data uploaded into SPICE is encrypted with AWS-managed keys.

#### *Encryption in Transit*

QuickSight supports encryption of all data transfers using SSL. This includes data to and from SPICE and from SPICE to the user interface.

#### *Key Management*

AWS manages all the keys associated with QuickSight. Database server certificates are the responsibility of the customer.

### Connecting To AWS Resources

We can connect to resources in AWS, for example, services like Redshift, RDS, Aurora, and databases on EC2. Make sure that all of these security mechanisms are set up and configured properly to allow QuickSight access to these resources. We can also connect QuickSight into S3. We will need to make sure that IAM roles, the bucket policy, and the manifest file are set up properly, which is essentially just a metadata file that allows QuickSight to connect the dots and better understand the data schema S3.

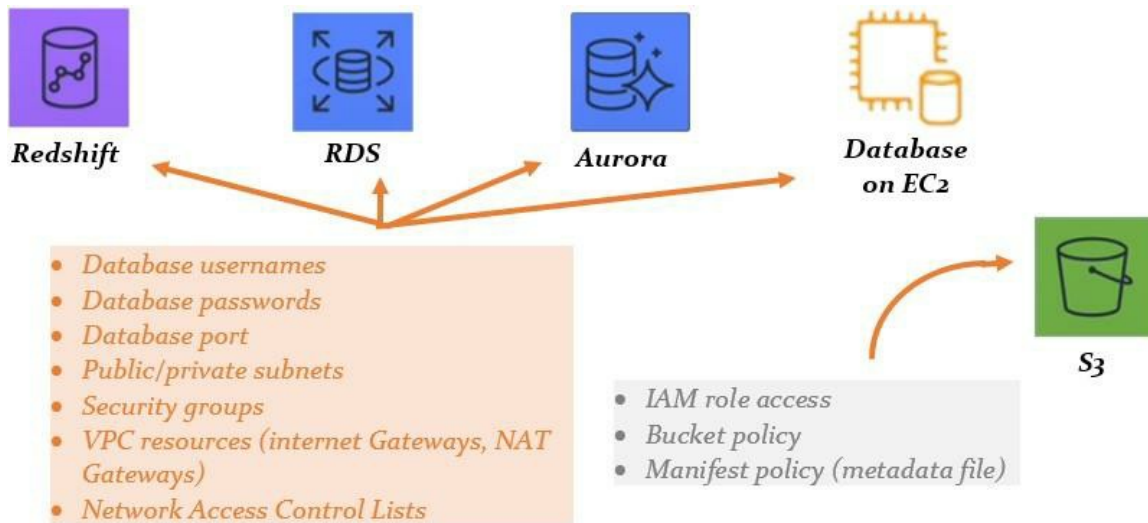


Figure 9-19: Connecting To AWS Resources

We have two different ways to connect; we can either connect manually or use auto-discovery. With manual connect, the username you are using to connect QuickSight to the database must have special permission on system tables. QuickSight can discover table schemas and estimate table size, and for auto-discovery, the resources must be in the same region as your QuickSight account. Whenever setting up a QuickSight account, it will be in a particular region. If you are trying to use auto-discovery, the resources you are trying to connect need to be in the same region.

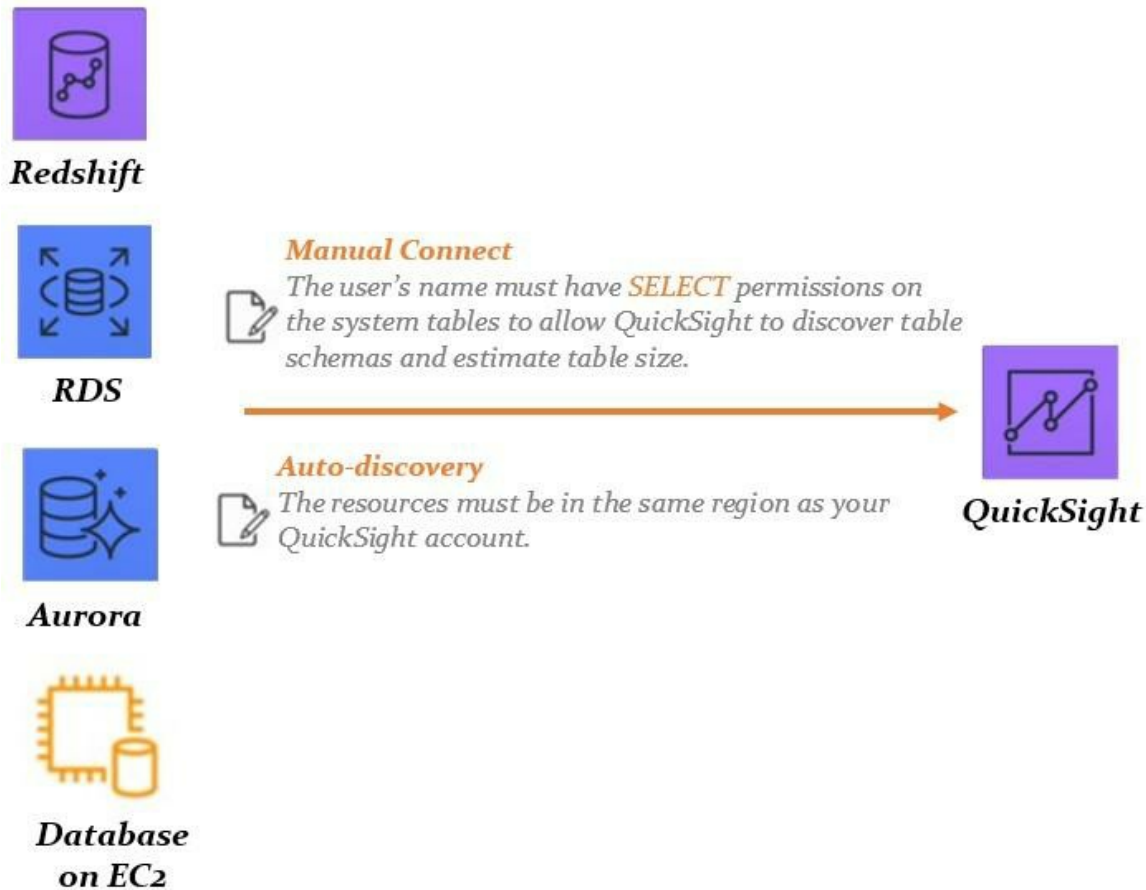
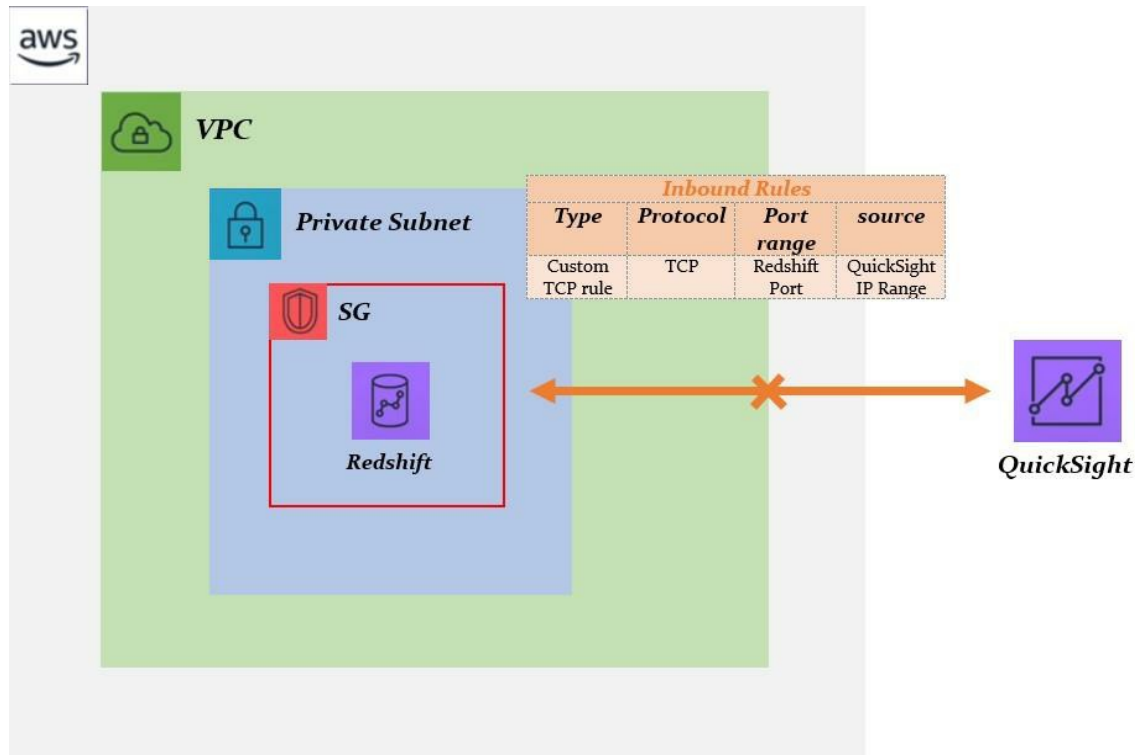


Figure 9-20: Ways to Connect

We have a Redshift cluster that will run in AWS, and it will be inside of a VPC. Most of the time, this Redshift cluster runs in a private subnet. We will have a security group associated with our Redshift cluster. By default, QuickSight is not allowed to access because the security group has not been set up properly or allows inbound rules from the QuickSight endpoint or the QuickSight IP address. Assume it has a bunch of more rules, but for this example, it has no rules. We need to specify the type as a TCP custom protocol and ensure the class is set up correctly as a TCP rule, where the protocol is TCP. The port range is the port that our Redshift cluster is running on, and in the source is the QuickSight IP range that we saw in the documentation. If we use auto-

discovery, they need to be in the same region but manually connecting; they could be in separate areas. Once we have set it up properly, we can communicate with QuickSight.



*Figure 9-21: Example*

#### EXAM TIP:

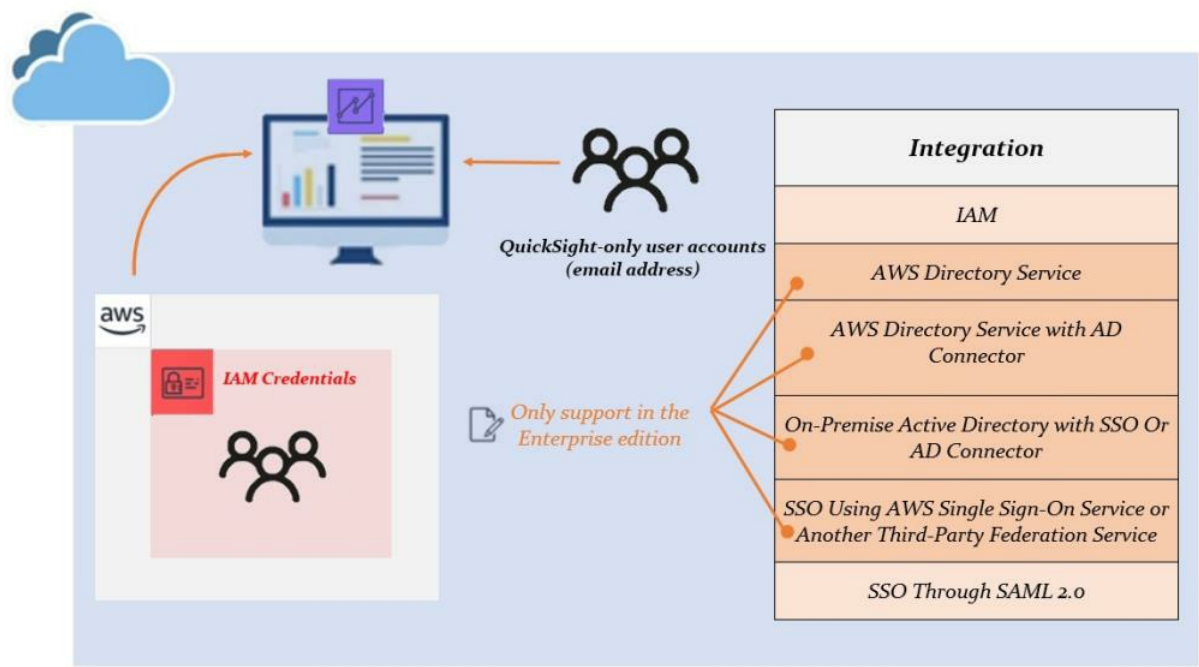
**Allowing Access to VPC Resources:** For Amazon QuickSight to connect to a private VPC resource, the security group must contain an inbound rule authorizing access from the appropriate IP address range for the QuickSight servers in the AWS Region:

- RDS Instance
- Redshift Clusters
- EC2 Instances

## Identity And Access Management In Quicksight

After creating dashboards and visualizations, you want to give users access to that data or those visualizations, so you need to find a way to make sure that you can set up your users within QuickSight. We have our QuickSight dashboards; they sit on the public internet. We can set up users either using IAM credentials or have QuickSight only users with just email addresses, but these are filtered through IAM.

There are many different ways to get users signed into your application. If you have your user directories, you can use IAM integration to give users access to dashboards. You can use third-party services like Octa and other SAML services and use single sign-on to ensure users have access. If you are using the standard edition and not the enterprise, you would not be able to take advantage of these integrations for your users.



*Figure 9-22: Identity and Access Management in Quicksight*

## Best Practices For Security

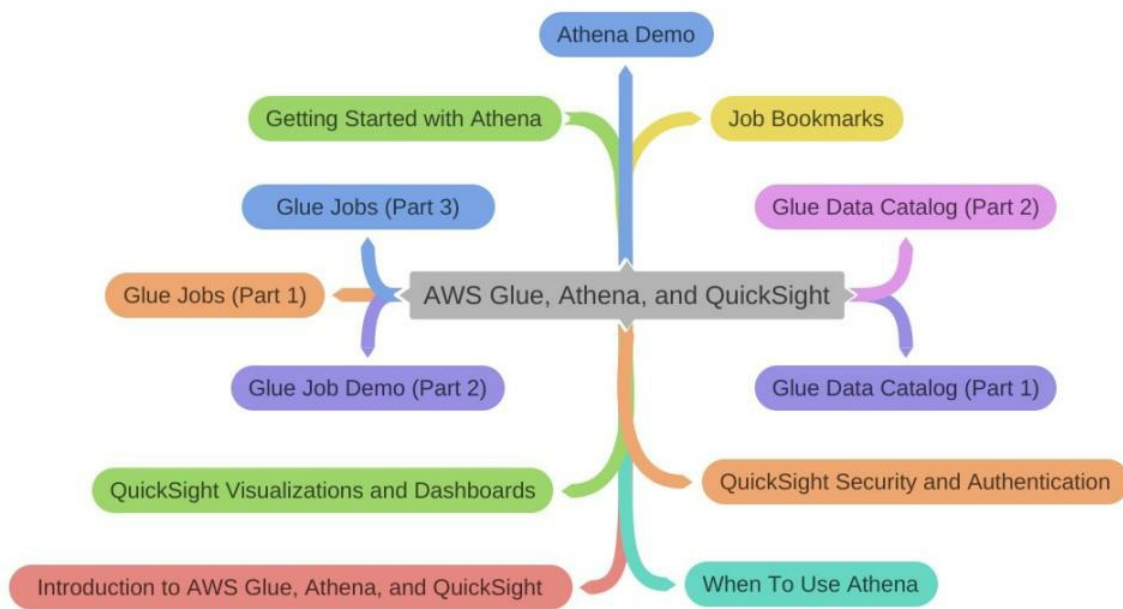
|  |  |
|--|--|
|  |  |
|--|--|

| Security Feature  | Best Practice                                                                                                                                                                                                                                                                     |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Firewall          | Allow access to HTTPS and WebSockets (wess: //) protocol. This allows QuickSight to reach a database that is not on AWS. Change the non-AWS server's firewall configuration to accept traffic from the appropriate QuickSight IP range.                                           |
| SSL               | Use SSL to connect to your database. Using SSL with QuickSight requires the use of publicity-recognized certificate authorities.                                                                                                                                                  |
| Enhanced Security | <ul style="list-style-type: none"> <li>• Encrypt data stored at rest in SPICE.</li> <li>• Integrate Active Directory and SSO authentication.</li> <li>• Securely access data in private VPCs and on-premises.</li> <li>• Limit access to data with low-level security.</li> </ul> |
| VPC               | Use a VPC for data in AWS data sources. Utilize AWS Direct Connect to create a secure, private connection to link your on-premise resources to your VPC and QuickSight.                                                                                                           |

*Table 9-09: Best Practices for Security*

**EXAM TIP:** Quicksight Data Encryption – How data is encrypted at rest, transit, and key management.

## Mind Map



*Figure 9-23: Mind Map*

## Practice Questions

1. What is AWS?

- A. Fully managed ETL service to categorize, clean, and enrich your data.
- B. Query Data in S3, join data and create a centralized metadata catalog.
- C. Source data stores, crawlers, catalog, jobs, output data store, or services using the data catalog.
- D. None of the above

2. AWS Glue use cases to -----.

- A. Query Data in S3

- B. Join data
- C. Create a centralized metadata catalog.
- D. All of the above

3. AWS Glue Components are -----.

- A. Source data stores, output data
- B. Crawlers, store
- C. Catalog, services using the data catalog.
- D. Jobs.
- E. All of the above

4. AWS Glue job -----.

- A. Extract
- B. Transform
- C. load (ETL) work in AWS Glue
- D. All of the above

5. The reason that JSON and CSV have an asterisk in output file format is that we have the option of ----- that data before it is stored off.

- A. Compressing
- B. Decompressing
- C. Both A and B

D. None of the above

6. AWS Glue Version 2.0 is billed in ----- increments with a 1-minute minimum.

A. 0.1-second

B. 1-second

C. 0.01-second

D. None of the above

7. AWS Glue Version 0.9 and 0.1 are billed in ----- increments with a 10-minute minimum.

A. 1-second

B. 0.1-second

C. 0.01-second

D. None of them

8. You can use ----- metrics to determine under or over-provisioned DPU's in the cluster by monitoring the total number of actively running executors, the number of completed stages, and the number of maximum needed executors

A. CloudWatch

B. Athena

C. S3

D. None of the above

9. ----- are the business logic that performs ETLs work in AWS Glue.

A. CloudWatch

B. Athena

C. AWS Glue jobs

D. None of the above

10. Data Processing Units (DPUs) are defined as -----.

A. The units used for processing your Glue

B. The units used for processing your database

C. The units used for processing your VPC

D. None of the above

11. Glue Jobs run on -----.

A. Virtual resources

B. Glue jobs needs

C. How traffic is governed

D. All of the above

12. If you have data in sources other than -----, you can use Federated Query (beta) to query the data in place or build pipelines that extract data from multiple data sources and store them in

Amazon S3.

- A. S3
- B. CloudWatch
- C. Athena
- D. None of the above

13. Athena natively supports -----.

- A. Querying datasets
- B. Data sources registered with the Glue Data Catalog
- C. Both A and B
- D. None of the above

14. What Is Athena?

- A. Serverless querying tool to easily query data in S3
- B. Serverless querying tool to easily query data in CloudWatch
- C. Serverless querying tool to easily query data in VPC
- D. None of the above

15. Athena Use Cases are -----.

- A. Ad-hoc queries
- B. Joining data from multiple data sources
- C. Creating ETL pipelines and transforming your data
- D. All of the above

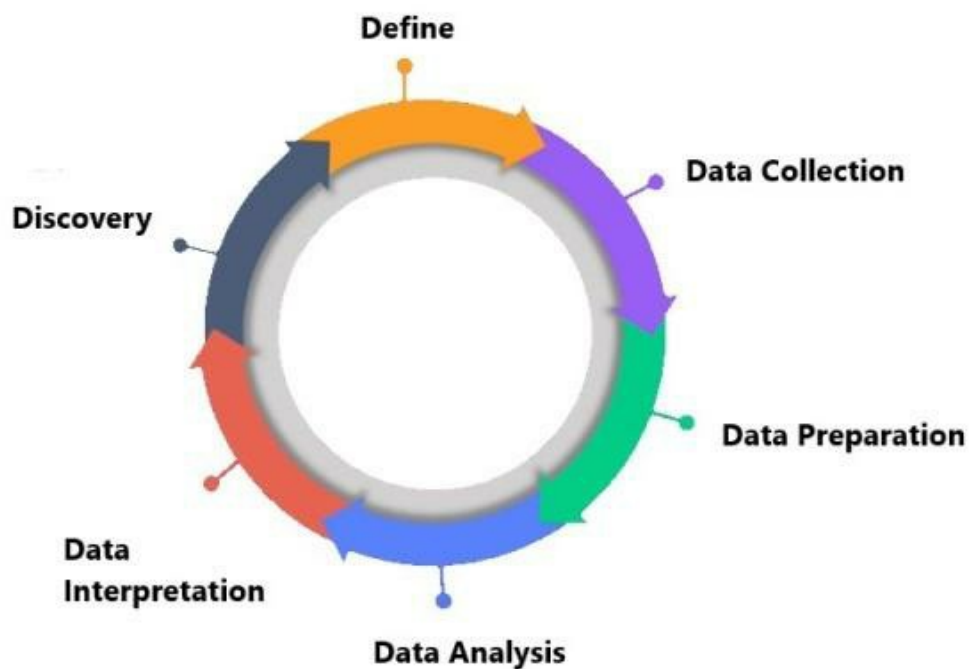


# CHAPTER 10: ELASTICSEARCH

## Introduction to Elasticsearch

The Amazon Elasticsearch Service is a managed service that makes deploying, operating, and scaling Elasticsearch in the AWS Cloud simple. Elasticsearch is a prominent open-source search and analytics engine for log analytics, real-time application monitoring, and clickstream analytics, among other applications.

With Amazon Elasticsearch Service, you have direct access to the Elasticsearch open-source API, allowing you to reuse code and applications from your existing Elasticsearch setups. Kibana is incorporated into the Amazon Elasticsearch Service, allowing you to easily display and analyze your data.



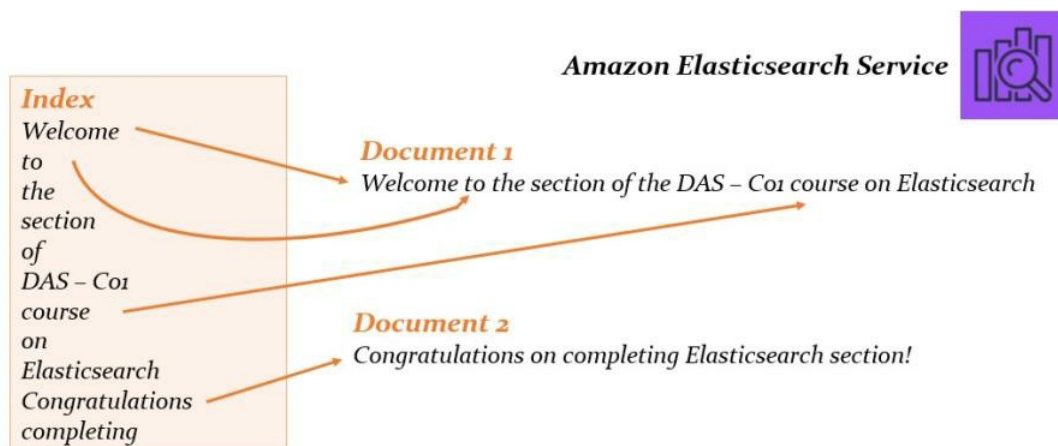
*Figure 10-01: Steps to Success*

## Elasticsearch Service

The Amazon Elasticsearch service is a search domain that runs most of the ELK (ElasticSearch Logstash and Kibana) stack. The service created with ElasticSearch is called search domains. They come pre-installed with ElasticSearch and Kibana Logstash; it is a fairly complex system all on its own, but it does have tight integration with ElasticSearch.

## Searching

ElasticSearch organizes data as indexes. The purpose of ElasticSearch is to enable word searching. It is a search engine utility. The way that searching works is called a reverse index. For example, we take two documents; ElasticSearch will index each word in two documents. Essentially, this service creates an index and can define some characteristics about it, and it puts every word in these two documents in all of the documents and stores it into that index. Part of the metadata in that index is which documents those words appear in and exactly where they are within that document. We get this mapping of all of the words in each document in the store. This is very powerful because it can return those documents very quickly, so we can search for words or phrases and find them very quickly.



*Figure 10-02: Searching*

## **Logs Into Data**

### ***Logitech***

Ingests, processes, and stores log data.

### ***Kibana***

Web utility interface for Elasticsearch and visualization engine.

**EXAM TIP:** The Amazon ElasticSearch service is a search domain that runs most of the ELK (ElasticSearch Logstash and Kibana) stack.

## **Using Elasticsearch**

### **JSON All the Way Down**

#### ***Index***

Top-level organizational unit.

#### ***Type***

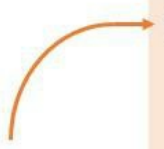
The second level is used to categorize data.

#### ***Document***

Elasticsearch data object.

### Document

Elasticsearch data object



```
{
  "r-aws-scrap": {
    "aliases": {},
    "mappings": {
      "properties": {
        "author": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        }
      }
    }
  },
  "body": {
    "type": "text",
    "fields": {
      "keyword": {
        "type": "keyword",
        "ignore_above": 256
      }
    }
  }
}
```

*Figure 10-03: JSON All the Way Down*

## The Interface

Elasticsearch uses a REST (Representational State Transfer) API for its interface. JSON is a common format for REST APIs. With the standard HTTP methods, we can interact with Elasticsearch; assuming that we have all the permissions open, we can send a GET to the base URL, then the index name, and provide a type and item ID. It will return that item.

We have a few other options if we break that down. If we put either the index or the item ID, it will add a document to either that index or item

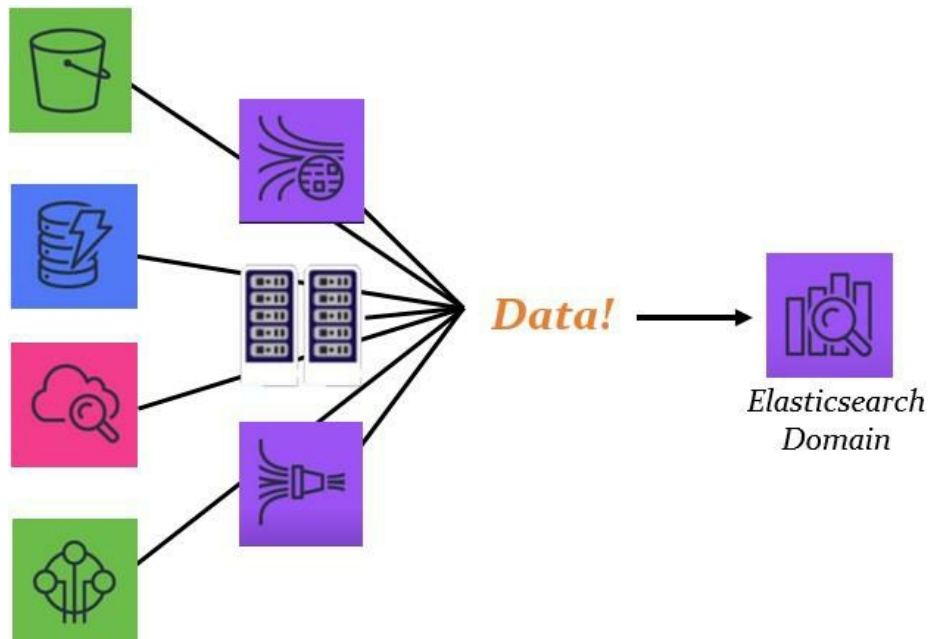
ID. If we do it to just the index, it will generate an ID. If we use a post, we can post a new type, which will let us set the definition for that type, but we cannot send something straight to the index. It would not generate an ID. The delete technique can also be used to delete individual items or an entire index. You can delete a type, although this is not particularly useful because it will invalidate any indexes that use it.



Figure 10-04: The Interface

## Loading Data

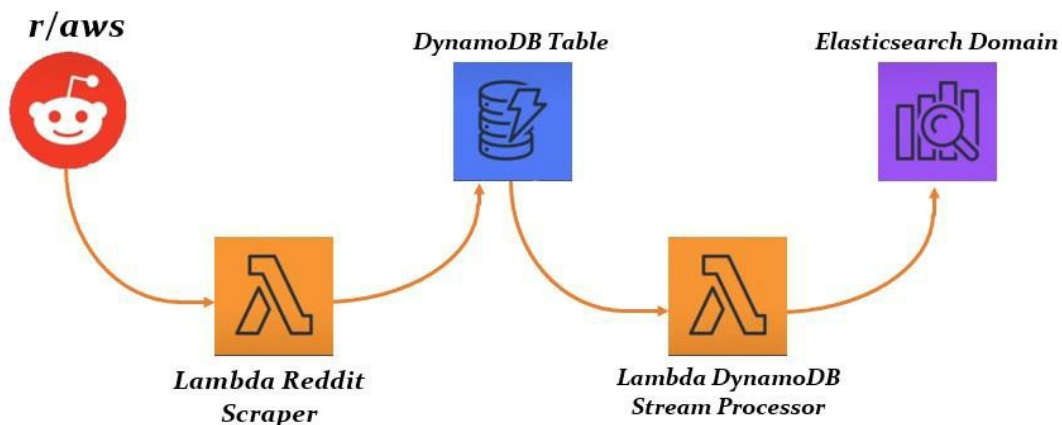
Anything that can interact with the API can send data into Elasticsearch. We will have to write the actual code to send that data into Elasticsearch, but it is just interacting with an API.



*Figure 10-05: Loading Data*

## Demonstration Pipeline

For example, you have set up a Lambda function that scrapes the AWS subreddit. It loads that data into a DynamoDB table, and then from the stream on that table, we are loading data into our Elasticsearch domain. The stream processor deserializes the data. It is not in the DynamoDB JSON format, and it loads it into Elasticsearch's documents.



*Figure 10-06: Demonstration Pipeline*

# Service Integration

## *Kinesis Data Firehose*

Elasticsearch is a configurable target.

## *Cloudwatch*

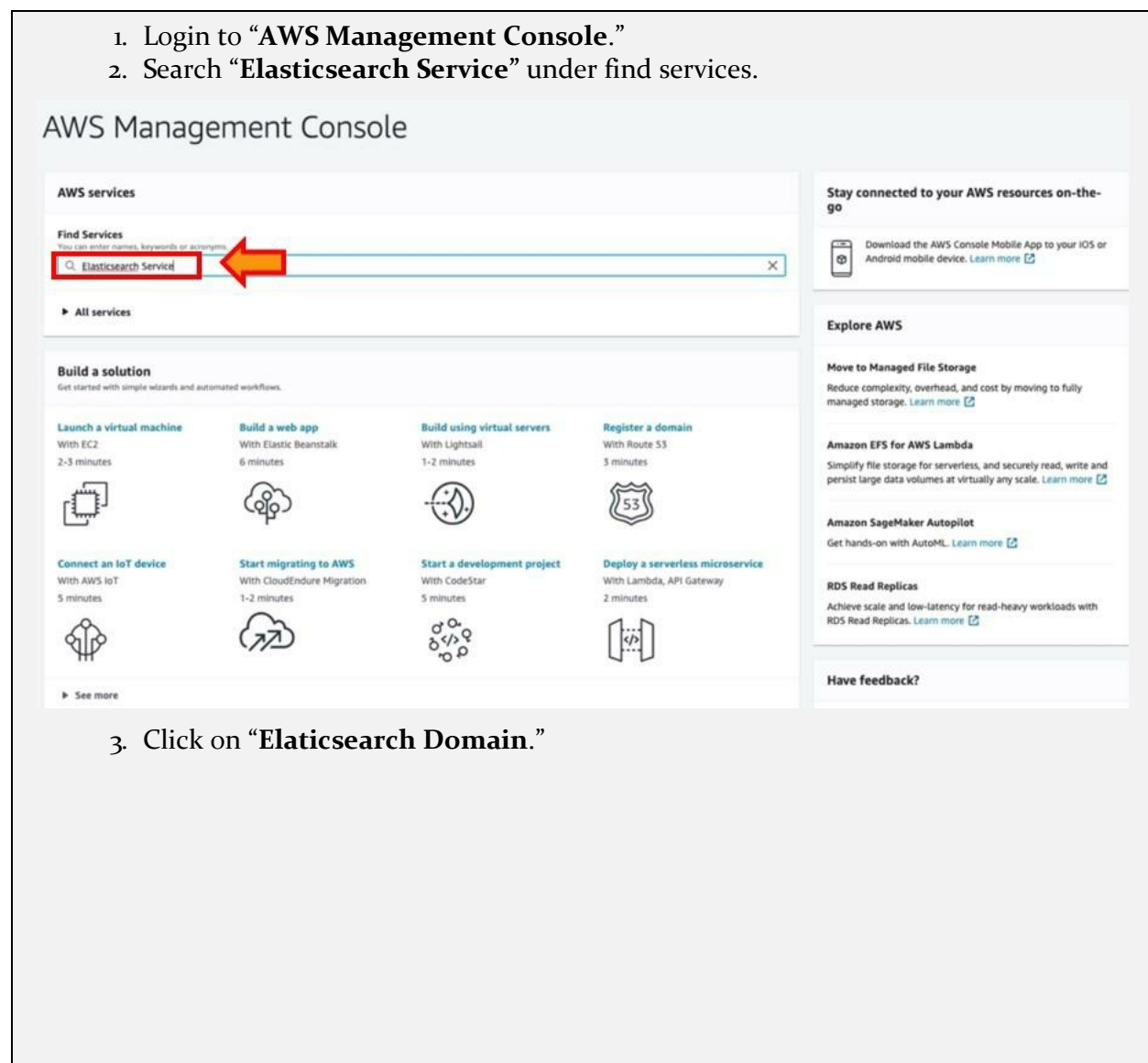
CloudWatch Logs subscription can deliver to Elasticsearch Service.

## *IoT*

IoT rules can send data to Elasticsearch Service.

## Demo 10-01: Querying Elasticsearch

1. Login to “AWS Management Console.”
2. Search “Elasticsearch Service” under find services.



The screenshot displays the AWS Management Console interface. At the top, the title "AWS Management Console" is visible. Below it, the "AWS services" section contains a "Find Services" search bar. A red box highlights the search bar, and a red arrow points to the text "Elasticsearch Service" entered in the search field. Below the search bar, there is a list of services, with "Elasticsearch Service" being the first result. To the right of the search bar, there is a sidebar with various links and information, including "Stay connected to your AWS resources on-the-go", "Explore AWS", and "Have feedback?".

3. Click on “Elasticsearch Domain.”

### Amazon Elasticsearch Service dashboard

Create a new domain

My Elasticsearch domains

| Domain       | Elasticsearch version | Endpoint | Searchable documents | Elasticsearch cluster health | Free storage space | Minimum free storage space | UltraWarm storage usage | Domain status |
|--------------|-----------------------|----------|----------------------|------------------------------|--------------------|----------------------------|-------------------------|---------------|
| r-aws-scrape |                       | Internet | 1,215                | Yellow                       | 78.67 GiB          | 78.67 GiB                  | Disabled                | Active        |

Learning content

- UltraWarm for Amazon Elasticsearch Service**  
Step-by-step tutorial to help you enable UltraWarm storage tier. [Learn more](#)
- Upload Data**  
Step-by-step tutorial to upload data to an Amazon Elasticsearch Service domain. [Learn more](#)
- Set Alerts in Amazon Elasticsearch Service**  
Learn how to monitor your log data and set thresholds and alerts. [Learn more](#)

Feature Spotlight

- k-Nearest Neighbor (k-NN) Search**  
Easily enable high scale, low latency nearest neighbor search on billions of documents across thousands of dimensions. [Learn more](#)
- UltraWarm for Amazon Elasticsearch Service**  
Store and interactively analyze petabytes of logs at 1/10th the cost of your existing storage tiers. [Learn more](#)
- Fine-Grained Data Access**  
Define granular permissions for indices, documents, or fields and extend Kibana with read-only views. [Learn more](#)

4. We can see an internet endpoint, and it has a little 1200 documents to search.

Dashboard

My domains

**r-aws-scrape**

Reserved instances

Packages

**Elasticsearch version** 7.7

**Endpoint** <https://search-r-aws-scrape-ipb6jt4hdfgwwbkdmg2gr2uy.us-west-2.es.amazonaws.com>

**Domain ARN** arn:aws:es:us-west-2:694501820626:domain/r-aws-scrape

**Kibana** [https://search-r-aws-scrape-ipb6jt4hdfgwwbkdmg2gr2uy.us-west-2.es.amazonaws.com/\\_plugin/kibana/](https://search-r-aws-scrape-ipb6jt4hdfgwwbkdmg2gr2uy.us-west-2.es.amazonaws.com/_plugin/kibana/)

**Availability zones** 1

**Instance type (data)** r5.large.elasticsearch

**Number of nodes** 1

**Data nodes storage type** EBS

**EBS volume type** General Purpose (SSD)

**EBS volume size** 100 GiB

**Upgrade status** -

**Start hour for the daily automated snapshot** 00:00 UTC (default)

**Fine-grained access control** Enabled

**Master user type** Internal user database

**Amazon Cognito for authentication** Disabled

**Require HTTPS** Enabled

**Encryption at rest** Enabled

**KMS master key** arn:aws:kms:us-west-2:694501820626:key/7831ad0e-b17d-45c9-a743-0f9157a20cf7

**Node-to-node encryption** Enabled

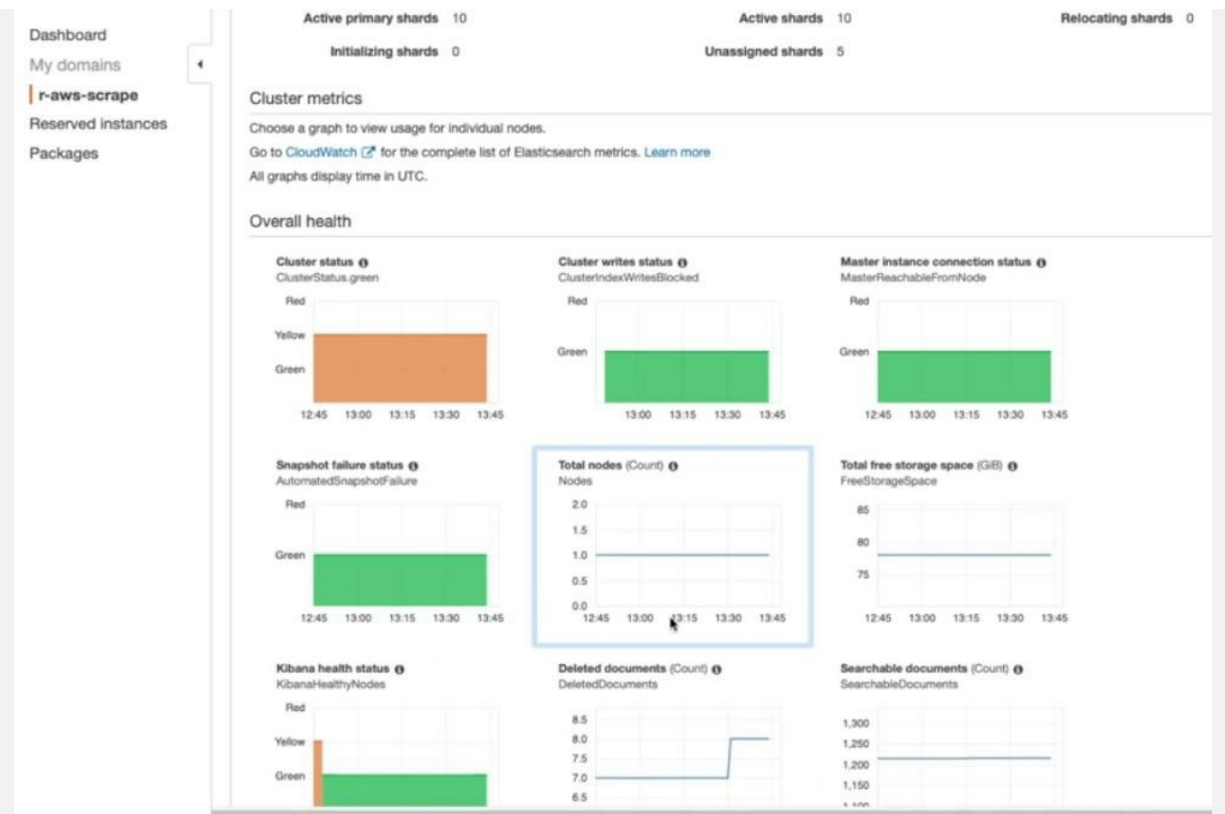
**Service software release** R20201019 [Update](#)

**Optional Elasticsearch cluster settings**

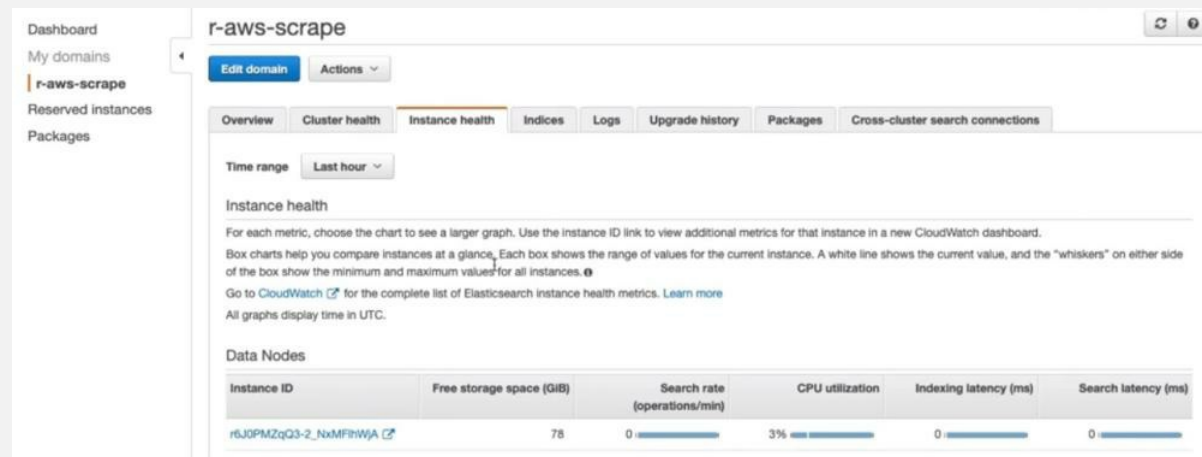
Allow APIs that can span multiple indices and bypass index-specific access policies: true

Fielddata cache allocation: unbounded (default)

5. We can see Cluster Metrics and Overall Health.



6. We can see Instance Health.



7. We can see Indices.

## r-aws-scrape

Edit domain

Actions ▾

Overview

Cluster health

Instance health

Indices

▸ .kibana\_2314539\_john\_2

▸ r-aws-scrape

▸ .kibana\_2314539\_john\_1

▸ .opendistro\_security

▸ .kibana\_1

▸ .tasks

8. There are several logs available.

[Edit domain](#)

Actions ▾

[Overview](#) [Cluster health](#) [Instance health](#) [Indices](#) [Logs](#) [Upgrade history](#) [Packages](#) [Cross-cluster search connections](#)**Set up Search slow logs**

Search slow logs provide insights into how fast or slow queries and fetches are performing. Once enabled, your logs are captured in CloudWatch Logs under the log group you specify. Standard CloudWatch charges will apply. [Learn more](#)

Status Disabled

CloudWatch Logs log group -

[Setup](#)**Set up Index slow logs**

Index slow logs provide insights into the indexing process. Once enabled, your logs are captured in CloudWatch Logs under the log group you specify. Standard CloudWatch charges will apply. [Learn more](#)

Status Disabled

CloudWatch Logs log group -

[Setup](#)**Set up Error logs**

Elasticsearch error logs provide access to information about errors and warnings raised during the operation of the service and can be useful for troubleshooting. Once enabled, your logs are captured in CloudWatch Logs under the log group you specify. Standard CloudWatch charges will apply. [Learn more](#)

Status Disabled

CloudWatch Logs log group -

[Setup](#)**Set up Audit logs**

Audit logs are records that provide documentary evidence of security related activities in a given system. Audit logs let you track user access to your Elasticsearch cluster and are useful for compliance purposes or in the aftermath of a security breach. [Learn more](#)

## 9. We can see Packages.

[Edit domain](#)

Actions ▾

[Overview](#) [Cluster health](#) [Instance health](#) [Indices](#) [Logs](#) [Upgrade history](#) [Packages](#) [Cross-cluster search connections](#)**Packages**

Packages allow you to add custom dictionary files, such as synonyms or stop words, to your domain. To upload packages to your AWS account, navigate to Packages from the left navigation bar. [Learn more](#)

[Associate](#)

Actions ▾

1

| Package name ▾ | Package ID ▾ | Reference path ▾ | Package associated ▾ | Package status ▾ | Message ▾ |
|----------------|--------------|------------------|----------------------|------------------|-----------|
|----------------|--------------|------------------|----------------------|------------------|-----------|

No packages have been associated with your domain. Packages must be uploaded to your AWS account from S3 prior to being associated with a domain, from the Packages menu in the left navigation bar.

[Associate](#)

## 10. You can create cross-cluster search connections.

r-aws-scrape

Edit domain Actions

Overview Cluster health Instance health Indices Logs Upgrade history Packages Cross-cluster search connections

### Outbound cluster connections

Connect destination clusters to enable cross-cluster search. You can send a search request to one cluster and receive results from several clusters. To add a destination cluster, specify its Amazon Elasticsearch Service domain ARN.

Connect Remove

| Domain name                             | Domain alias | Domain ARN | Request status |
|-----------------------------------------|--------------|------------|----------------|
| There are no remote clusters connected. |              |            |                |

Connect

### Inbound cluster connections

Allow another cluster to send a search request and receive results from this cluster. A coordinating cluster can send a search request to this cluster.

Accept Reject Remove

| Domain name                                   | Domain ARN | Request status |
|-----------------------------------------------|------------|----------------|
| There are no clusters requesting connections. |            |                |

11. You can query against the domain.

```
[root@ip-172-31-13-131 query-es]# python3 query_es.py -qt s3
```

12. The query will return all the documents that have S3 somewhere in the body.

```
{
  "subreddit": "aws",
  "title": "EC2 Instance - separate account bucket access"
},
{
  "_type": "_doc"
},
{
  "_id": "itrmrh",
  "_index": "r-aws-scrape",
  "_score": 3.2249413,
  "_source": {
    "author": "Lucas314159",
    "body": "Currently, we are using our rest-api backend to deliver files stored in S3 to our customers. This has the advantage that it's easy to authenticate users. However, we want to optimize those files for speed.\n\nThat's why we are thinking about serving files directly over S3 instead of going through our backend. The files have random file names, so we would only give authenticated users access to these filenames using our rest api. Should we just make our bucket public or use CloudFront in front of S3?",
    "created_unix": 1600244783000,
    "created_utc": "2020-09-16 08:26:23",
    "id": "itrmrh",
    "link": "https://www.reddit.com/r/aws/comments/itrmrh/deliver_files_stored_in_s3_to_user/",
    "num_comments": 3,
    "score": 2,
    "subreddit": "aws",
    "title": "Deliver files stored in S3 to user"
  },
  "_type": "_doc"
},
{
  "_id": "j9help",
```

13. To import some Elasticsearch functions, run content in the script.

```

query_es.py X
Users > john > Documents > Courses > Data Analytics DAS-C01 > repo > Content-AWS-Certified-Data-Analytics---Speciality > Elasticsearch > query_utility >
1 from elasticsearch import Elasticsearch, RequestsHttpConnection
2 from requests_aws4auth import AWS4Auth
3 import boto3
4 import json
5 import argparse
6
7
8 def query(term):
9     host = 'search-r-aws-scrape-ipb6jt4hdfgwbkexmg2gr2uy.us-west-2.es.amazonaws.com'
10    region = 'us-west-2'
11
12    service = 'es'
13    credentials = boto3.Session().get_credentials()
14    awsauth = AWS4Auth(credentials.access_key, credentials.secret_key, region, service, session_token=credentials.token)
15
16    es = Elasticsearch(
17        hosts = [{'host': host, 'port': 443}],
18        http_auth = awsauth,
19        use_ssl = True,
20        verify_certs = True,
21        connection_class = RequestsHttpConnection
22    )
23

```

14. Log in to “Open Distro for Elasticsearch.”

15. Put in the “Body: S3.”

16. Click on “Update.”



17. Run the query.

18. Edit the query.

19. Click the “Arrow” to send the request.

## EXAM TIP:

- JSON All the Way Down – Elasticsearch uses JSON for everything
- The Interface – REST interface, GET, PUSH, DELETE, POST
- Loading Data – Elasticsearch can ingest data from anything that can generate JSON
- Service Integration – Kinesis Firehose, CloudWatch, and IoT Have built-in ES integration

# Visualizing Elasticsearch Data

## Visualization Tools Examples

## ***Kibana***

- Part of ELK stack
- Pre-installed on Elasticsearch Service (ES) domains
- It does not require writing extra code

## ***D3.js***

- They are easily embedded in external applications
- Can visualize any JSON formatted data
- Third-party connectors are available

## **What can We Visualize?**

### ***Document Statistics***

Utilize various functions and filters to create statistical information about documents.

### ***Numerical Data***

Numerical data can be visualized or utilized to organize or filter document data.

## **Lab 10-01: Implementing an Elasticsearch (OpenSearch) Backed Search Microservice**

### **Introduction**

#### **1. Amazon Elasticsearch (OpenSearch)**

Interactive log analytics, real-time application monitoring, web search, and other duties are made easier with Amazon OpenSearch Service. OpenSearch is a distributed search and analytics package based on Elasticsearch that is open source. Amazon OpenSearch Service provides the most recent versions of OpenSearch, support for 19 Elasticsearch versions (1.5 to 7.10), and visualization features driven by OpenSearch Dashboards and Kibana (1.5 to 7.10 versions).

## **2. Amazon API Gateway**

Amazon API Gateway is a fully managed service that enables developers to effortlessly publish, administer, monitor, and secure APIs of any size. You can develop an API that serves as a "front door" for apps to access data, business logic, or functionality from your backend services, such as those operating on your server. Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Container Service (Amazon ECS), or AWS Elastic Beanstalkcode running on AWS Lambda, or any web application, with a few clicks in the AWS Management Console. Accepting and processing hundreds of thousands of concurrent API calls, including traffic management, authorization, access control, monitoring, and API version management, is supported by Amazon API Gateway. There are no minimum fees or initial charges with Amazon API Gateway. You only pay for the API requests you receive and the amount of data transmitted out when using HTTP APIs and REST APIs. You only pay for messages delivered and received, as well as the time a user/device is connected to the WebSocket API.

## **3. Amazon Lambda**

AWS Lambda allows you to run code without creating or managing servers. There is no charge when your code is not executing; you only pay for the compute time you use. You can run code for nearly any application or backend service with Lambda, and you do not have to worry about administration. Upload your code, and Lambda will handle everything necessary to run and grow it with high availability. You may configure your code to be automatically triggered by other AWS services, or you can access it directly from any computer or smartphone app.

#### **4. Amazon Simple Storage Service S3**

Amazon S3 is a type of object storage that allows you to store and recover any quantity of data from any location. It is a low-cost storage solution with business resilience, reliability, efficiency, privacy, and infinite expansion.

Amazon S3 is a web service that allows you to store and retrieve an infinite quantity of data from any place and at any time. You may quickly create projects that integrate cloud-native storage using this service. As Amazon S3 is easily customizable and you only pay for what you use, you can start small and scale up as needed without sacrificing performance or dependability.

Amazon S3 is also built to be highly adaptable. Instead of finding out how to store their data, Amazon S3 allows developers to focus on innovation. Build a simple FTP system or a complex web application like the Amazon.com retail website. Read the same piece of data a million times or only for emergency disaster recovery; store whatever type and amount of data you desire.

#### **Problem**

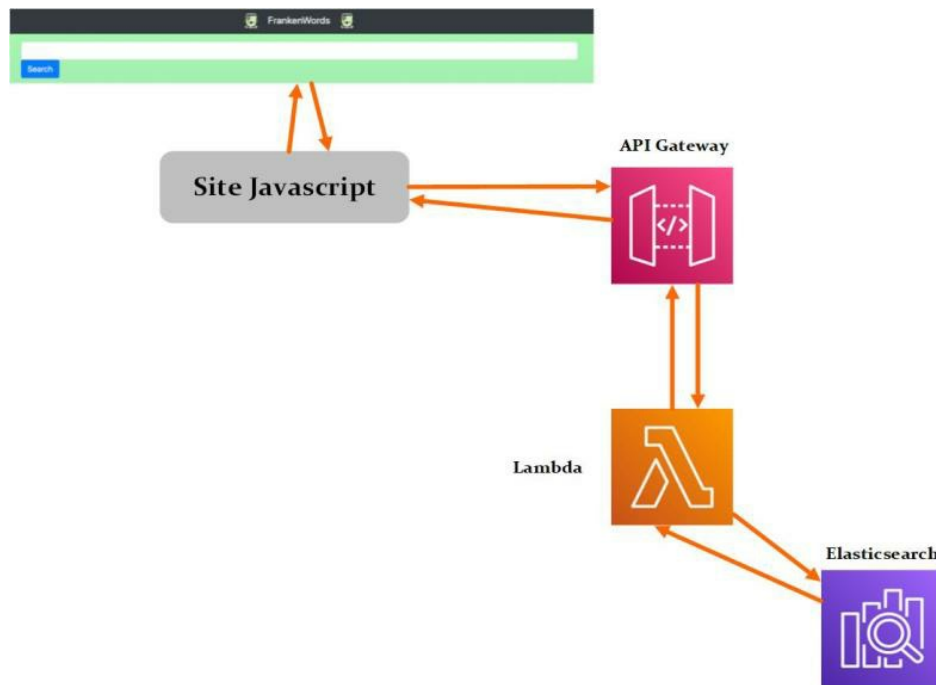
Assume you are a Data Analytics Engineer in an organization. The organization wants you to integrate a function in the website to provide statistical information about the appearance of words and phrases in any novel. How can you create this type of functionality in the website?

#### **Solution**

The solution is using AWS services to achieve this functionality in the website. You use AWS Elasticsearch, which name was changed to OpenSearch by AWS. You use the AWS Lambda and AWS API Gateway

as backend services in the website. For hosting the static website, you will use the AWS S3 hosting service.

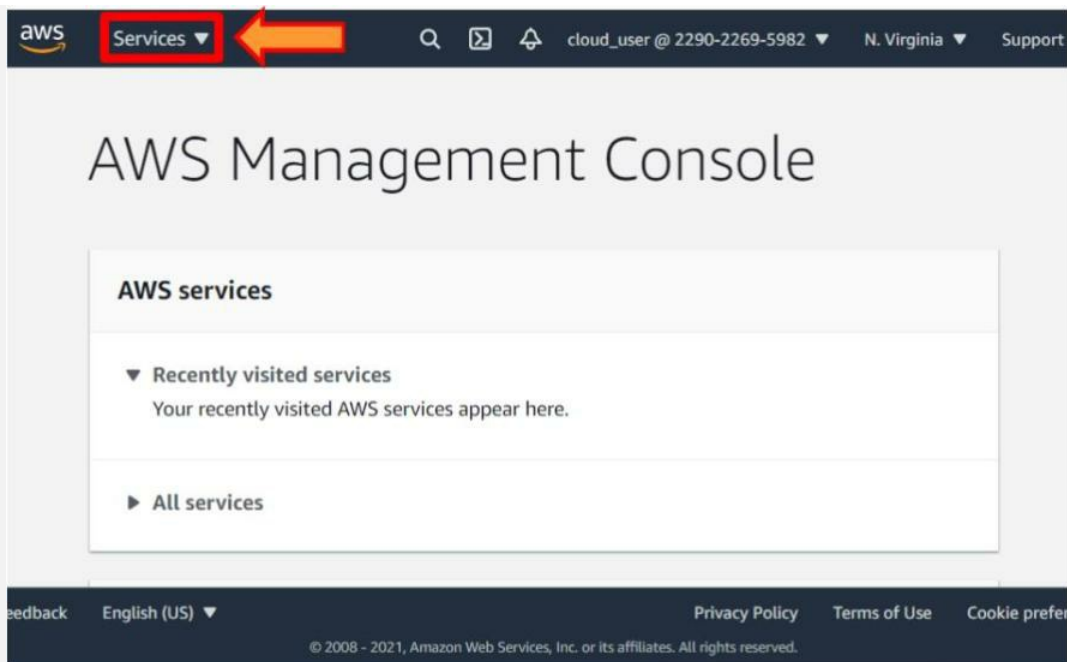
**Note:** Before starting the lab, create a Lambda function and Elasticsearch (OpenSearch) cluster.



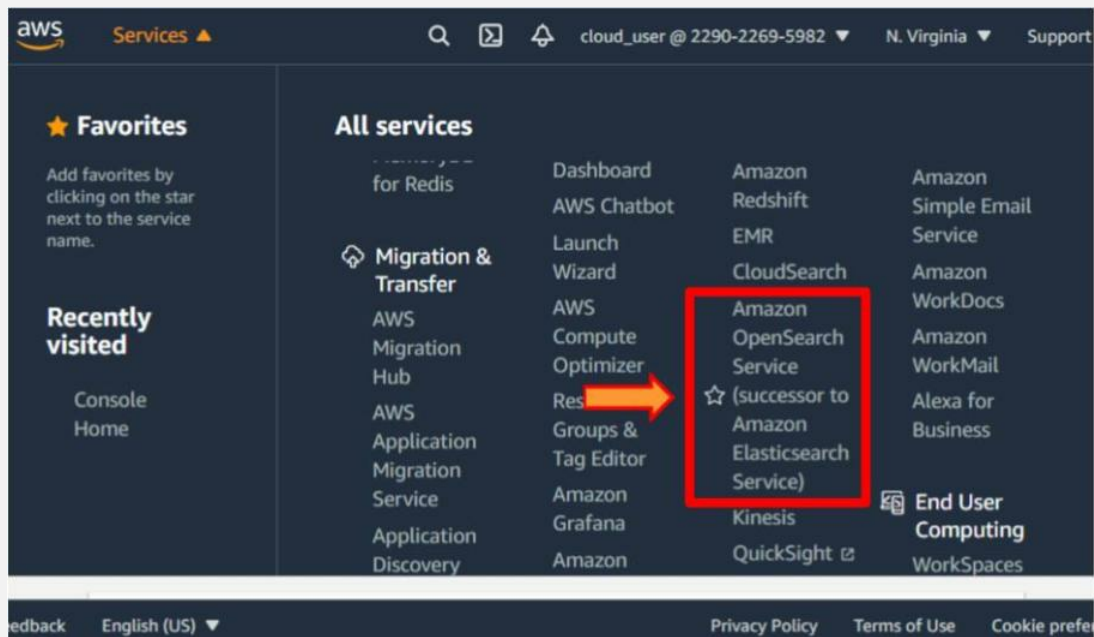
*Figure 10-07: Implementing an Elasticsearch (OpenSearch) Backed Search Microservice*

### Step 1: Review the Elasticsearch (OpenSearch) Query in Kibana

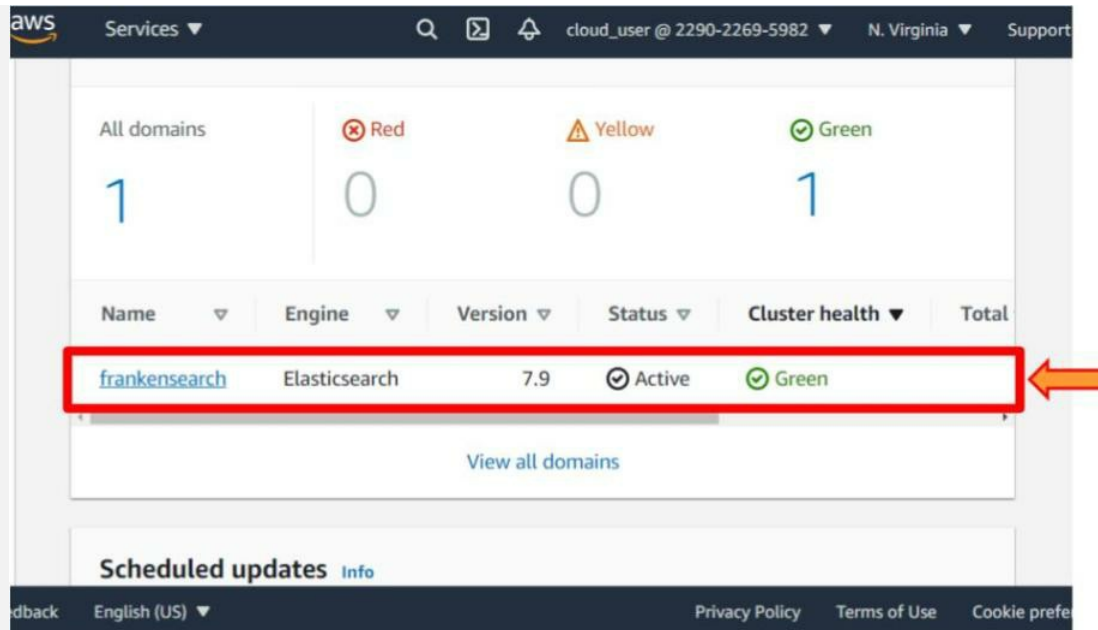
1. Log in to the “AWS Console.”
2. Click on the “Services.”



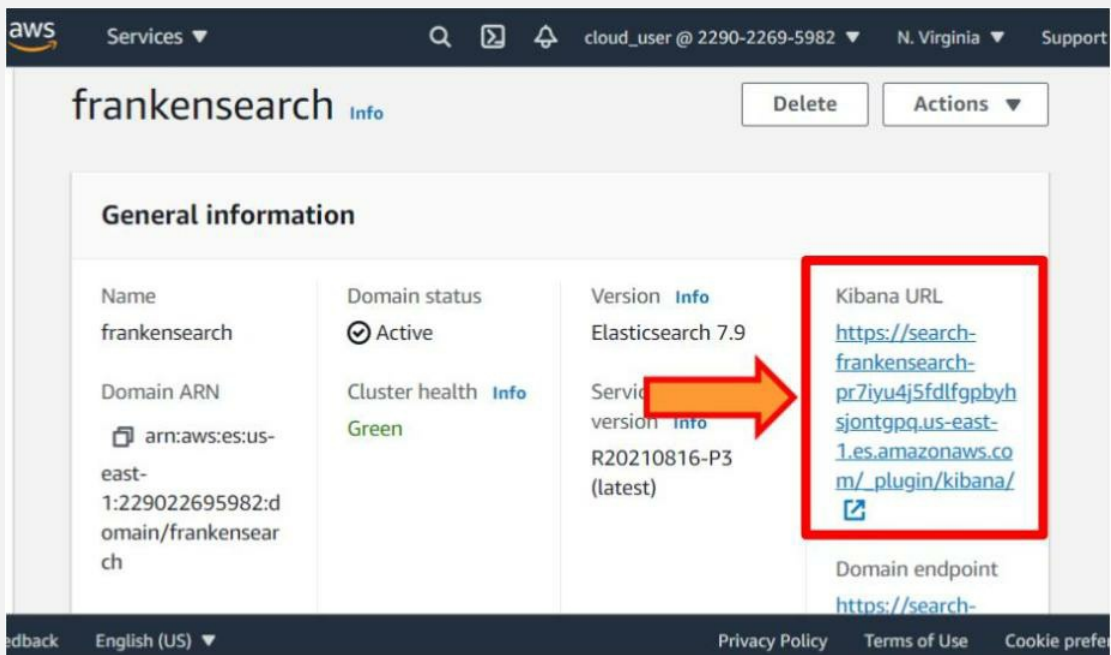
3. Select the “Amazon OpenSearch Service” from the “Analytics.”



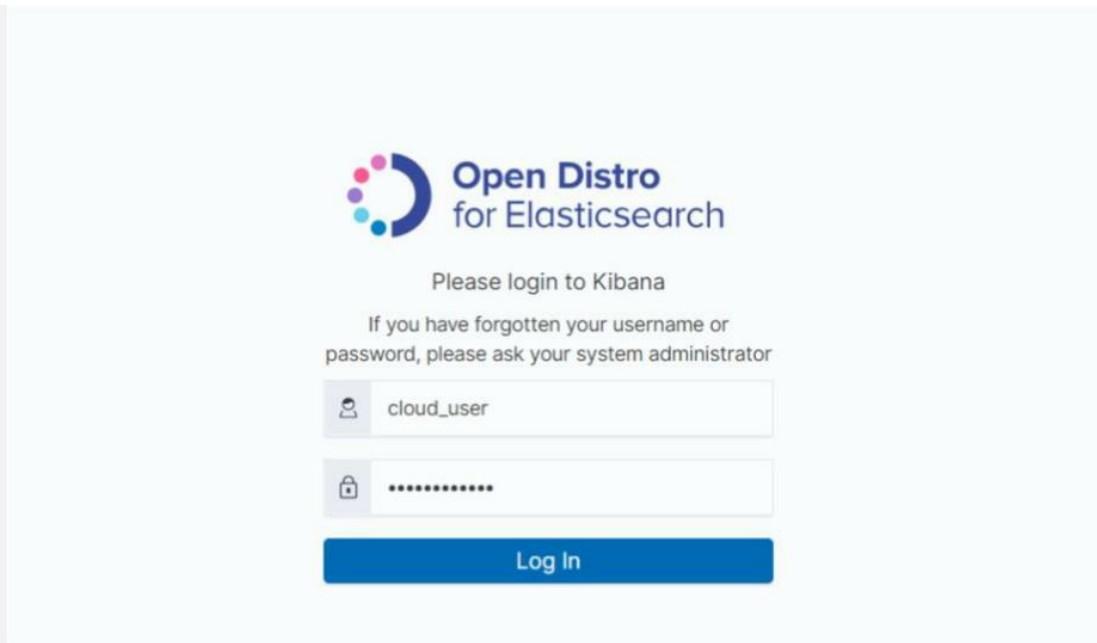
4. Click on the “frankensearch” domain.



5. Open the “Kibana” URL in a separate browser tab.

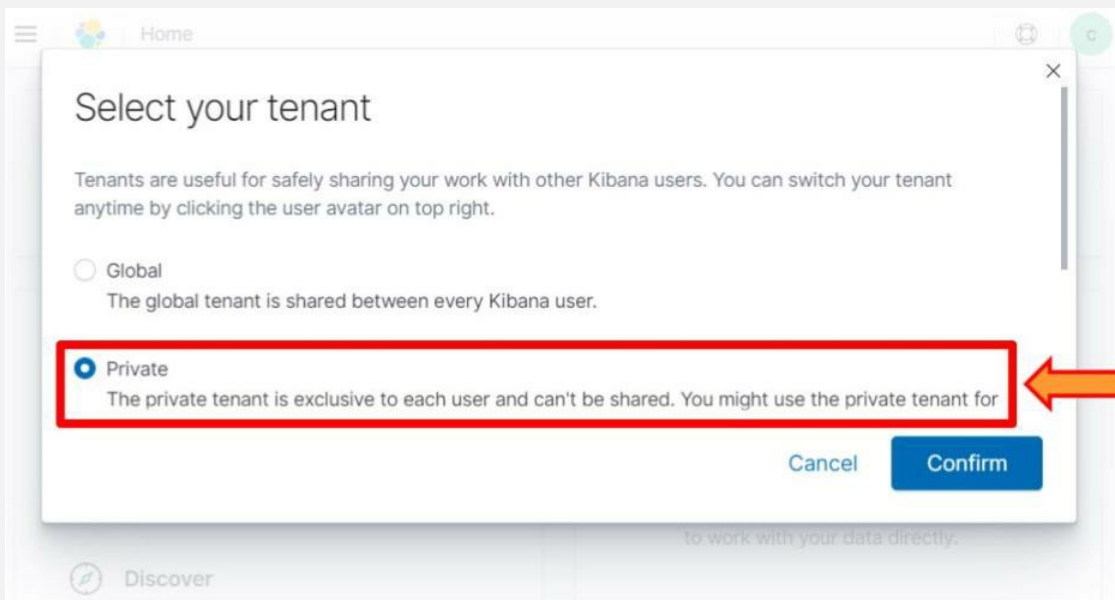


6. Log in to the “Kibana.”



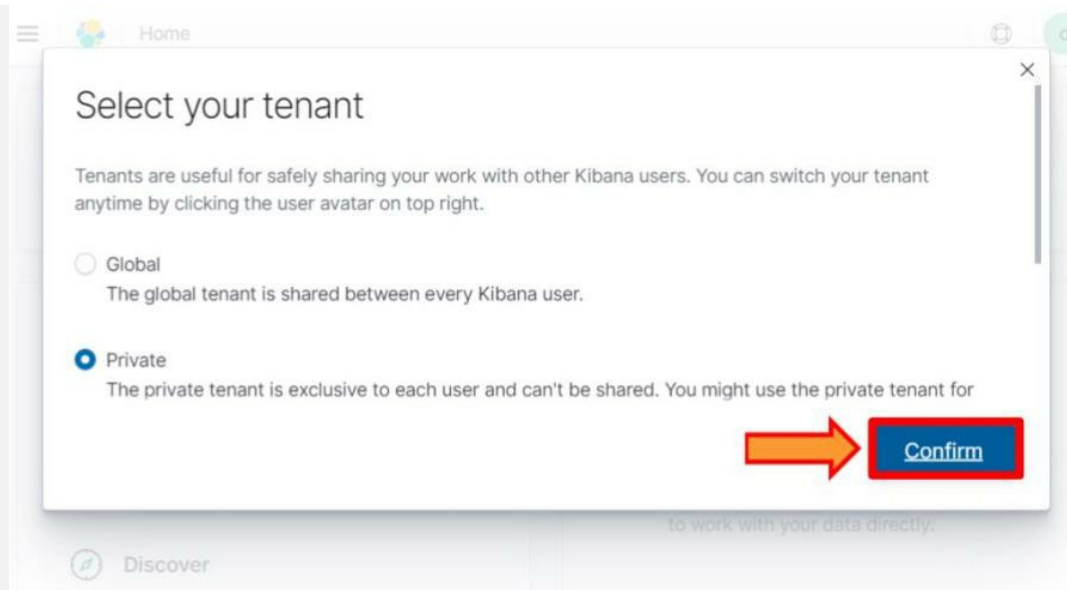
The image shows the login page for Open Distro for Elasticsearch. At the top is the logo, which consists of a circular arrangement of colored dots followed by the text "Open Distro for Elasticsearch". Below the logo, the text "Please login to Kibana" is displayed. A note states: "If you have forgotten your username or password, please ask your system administrator". There are two input fields: the first is for the username, containing the text "cloud\_user", and the second is for the password, represented by a series of dots. Below these fields is a blue button labeled "Log In".

7. Select the “**Private.**”

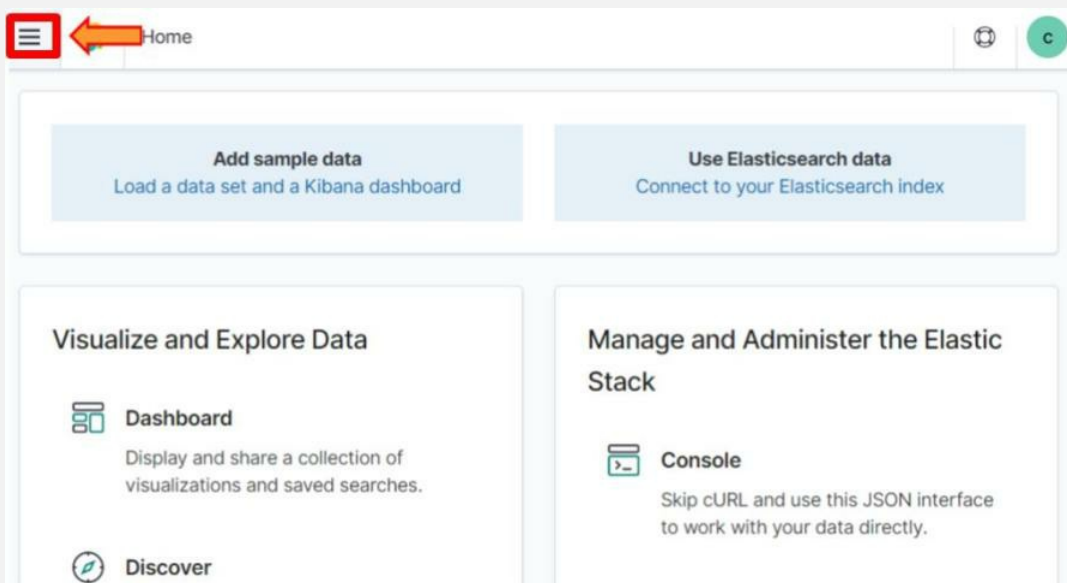


The image shows a "Select your tenant" dialog box. The title is "Select your tenant". Below the title, a paragraph explains: "Tenants are useful for safely sharing your work with other Kibana users. You can switch your tenant anytime by clicking the user avatar on top right." There are two radio button options. The first is "Global", with the description "The global tenant is shared between every Kibana user." The second is "Private", which is selected (indicated by a blue dot). The "Private" option is highlighted with a red rectangular box, and an orange arrow points to it from the right. The description for "Private" is: "The private tenant is exclusive to each user and can't be shared. You might use the private tenant for". At the bottom right of the dialog are two buttons: "Cancel" and "Confirm".

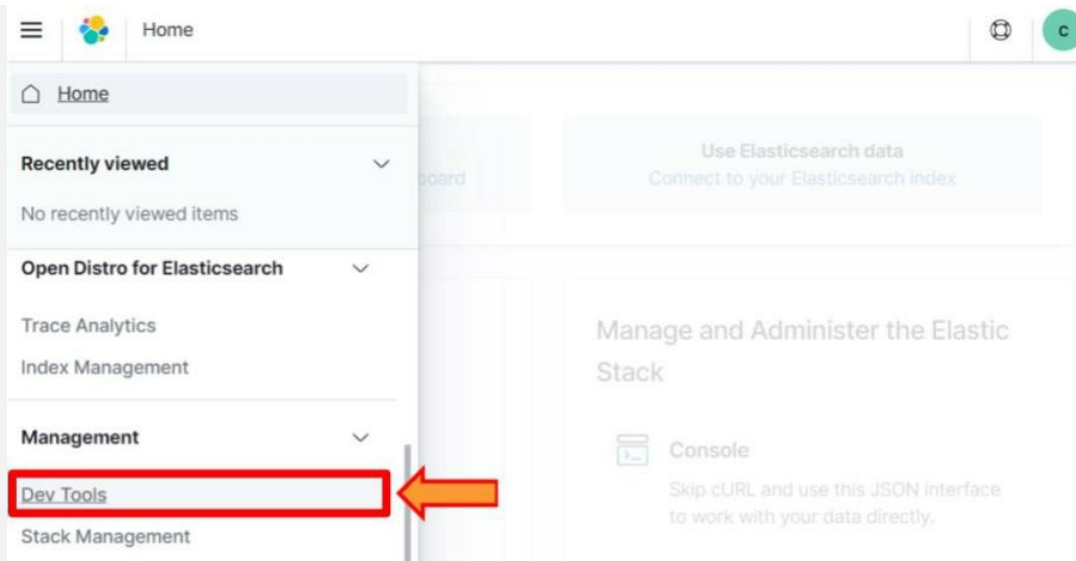
8. Click on the “**Confirm**” button.



9. Click on the “**Menu**” icon in the top left corner.

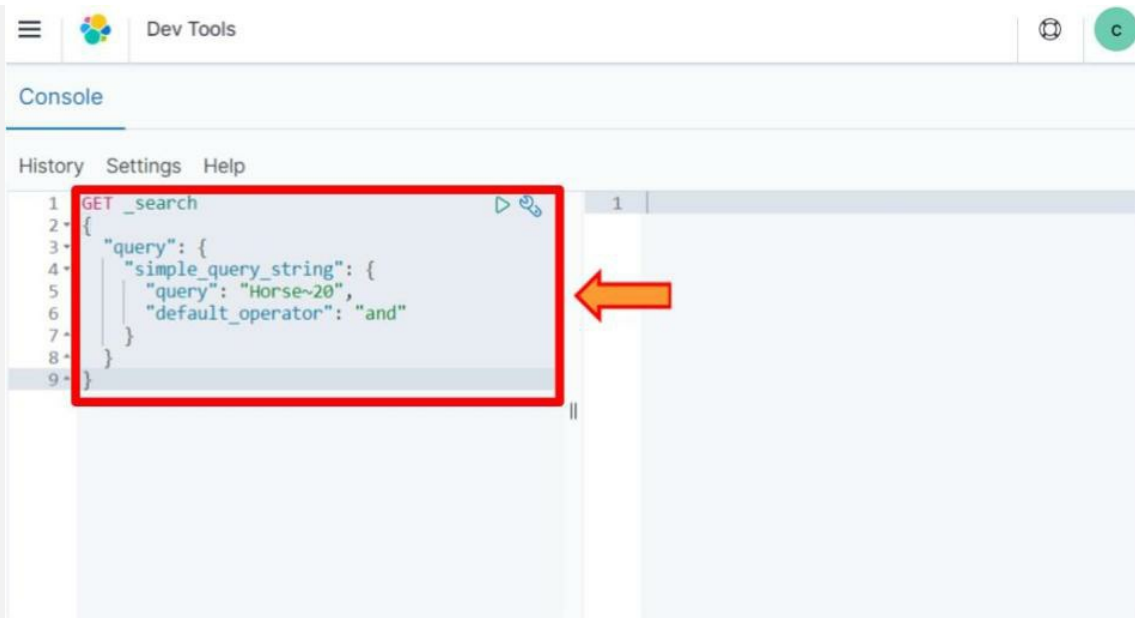


10. Click on the “**Dev Tools**.”

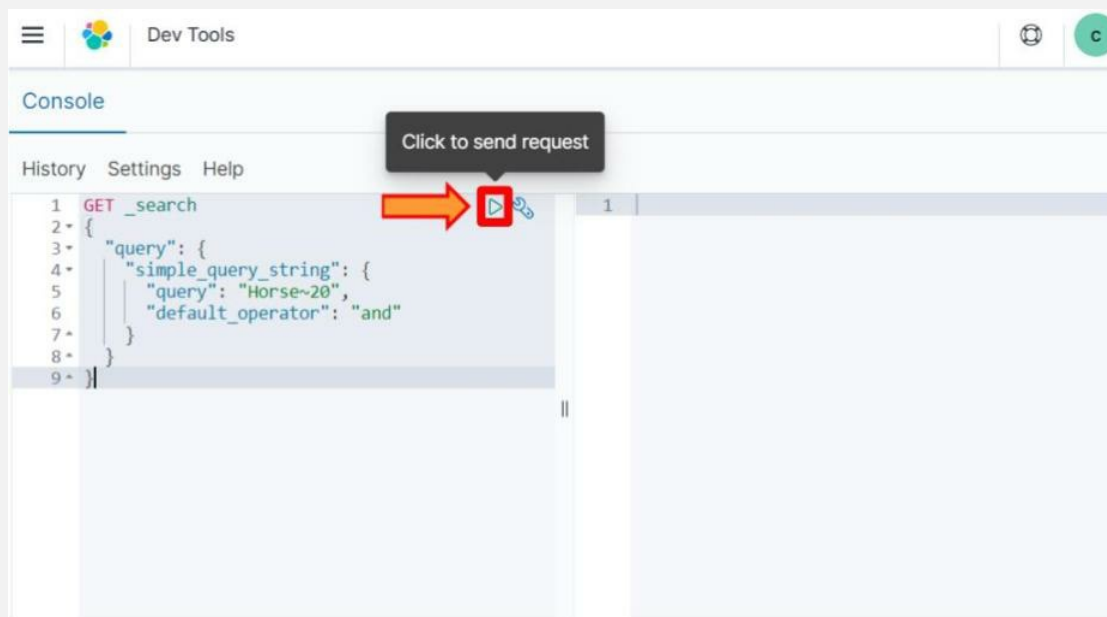


11. Copy the search query from below and paste it into the query editor.

```
GET _search
{
  "query": {
    "simple_query_string": {
      "query": "Horse~20",
      "default_operator": "and"
    }
  }
}
```



12. Click on the “**Play**” icon button.



13. The output shows 29 hits.

```
1 GET _search
2 {
3   "query": {
4     "simple_query_string": {
5       "query": "Horse~20",
6       "default_operator": "and"
7     }
8   }
9 }

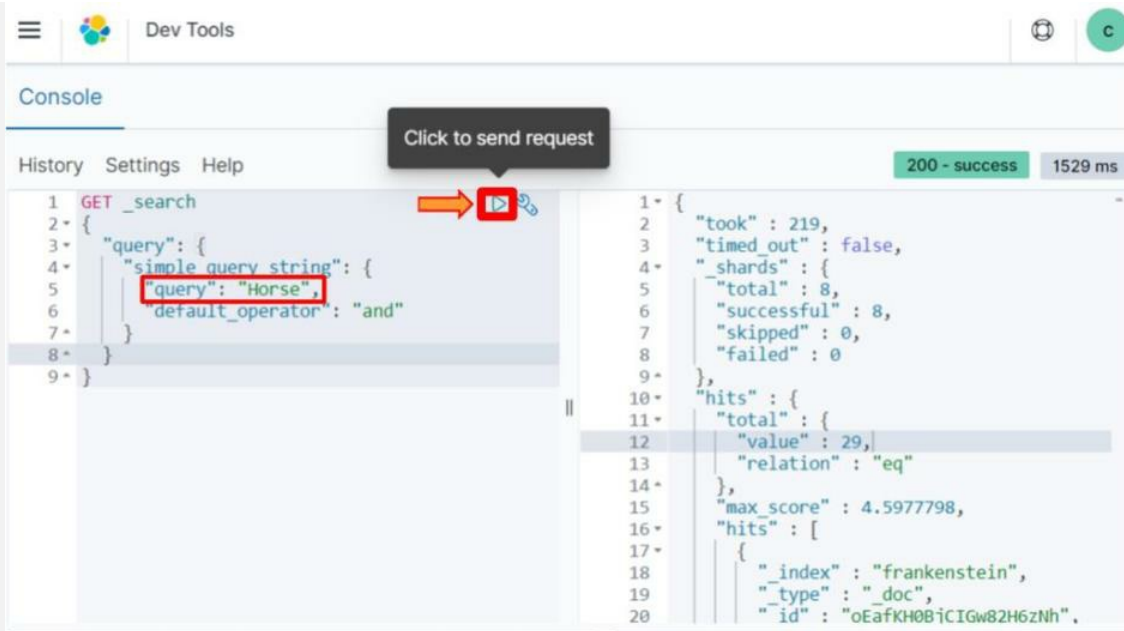
1 {
2   "took": 219,
3   "timed_out": false,
4   "shards": {
5     "total": 8,
6     "successful": 8,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": {
12      "value": 29,
13      "relation": "eq"
14    },
15    "max_score": 4.5977798,
16    "hits": [
17      {
18        "_index": "frankenstein",
19        "_type": "_doc",
20        "_id": "oEafKH0BjICIGw82H6zNh",
```

14. Delete the “~20” parameter from the “query” line to remove any fuzziness in the query editor.

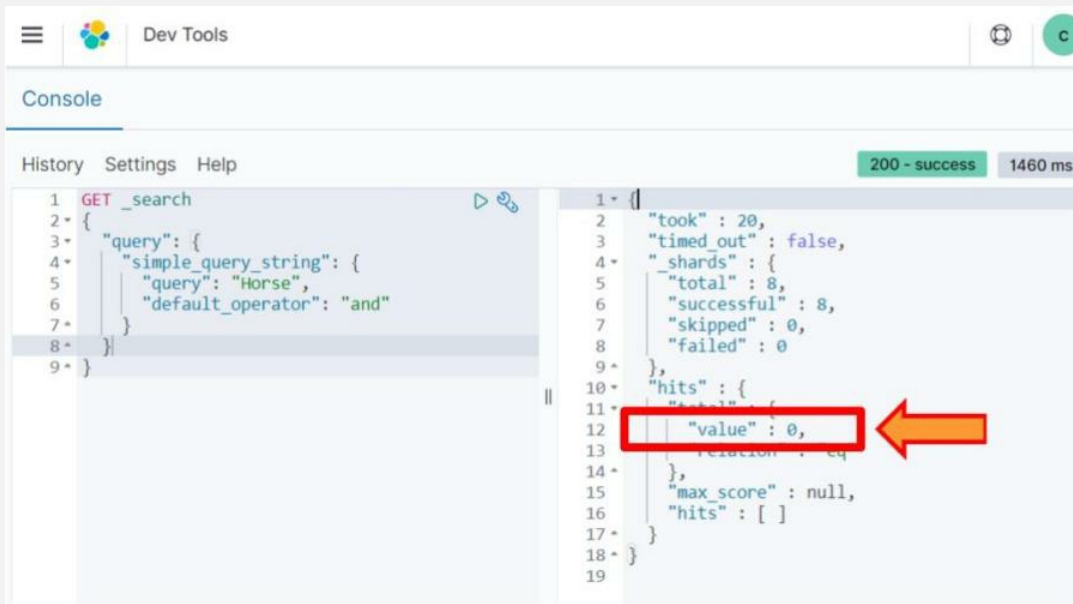
```
1 GET _search
2 {
3   "query": {
4     "simple_query_string": {
5       "query": "Horse~20",
6       "default_operator": "and"
7     }
8   }
9 }

1 {
2   "took": 219,
3   "timed_out": false,
4   "shards": {
5     "total": 8,
6     "successful": 8,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": {
12      "value": 29,
13      "relation": "eq"
14    },
15    "max_score": 4.5977798,
16    "hits": [
17      {
18        "_index": "frankenstein",
19        "_type": "_doc",
20        "_id": "oEafKH0BjICIGw82H6zNh",
```

15. Click on the “Play” icon button.



16. The output shows “o” hits.

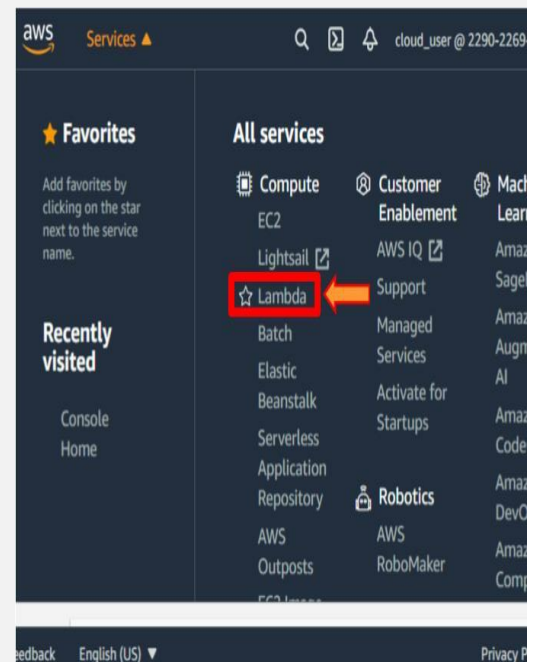


## Step 2: Update the Lambda Function

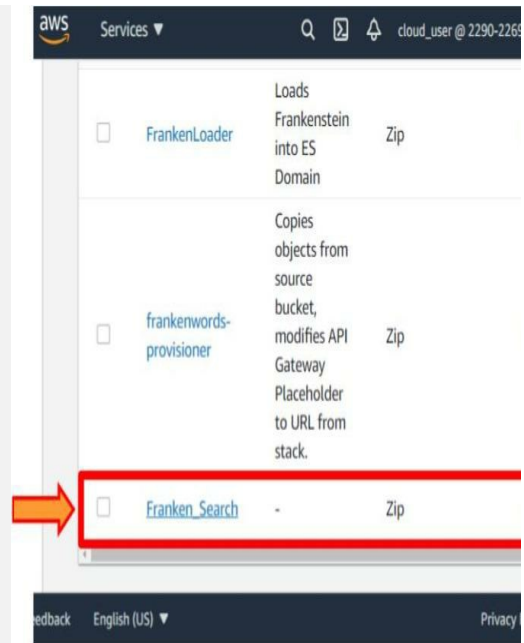
1. Click on the “Services.”



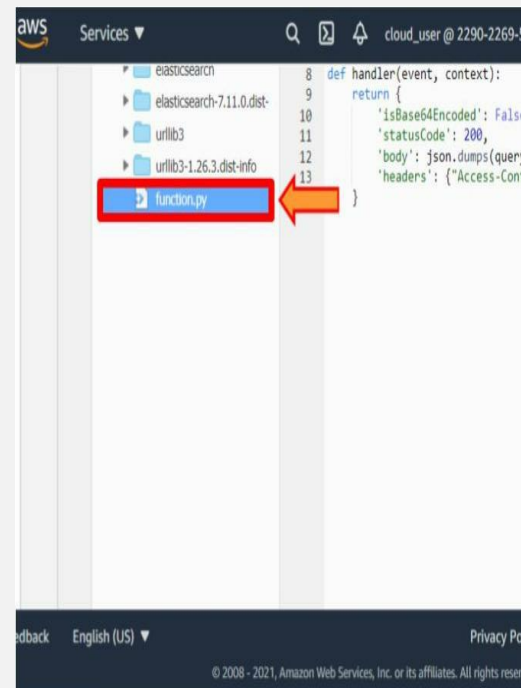
2. Select the “**Lambda**” from the “**Compute.**”



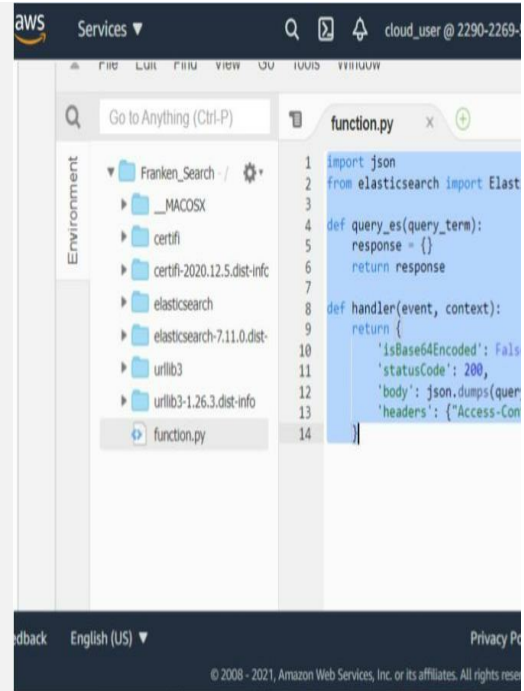
3. Click on the “**fraken\_search**” function.



4. Click on the “**function.py.**”



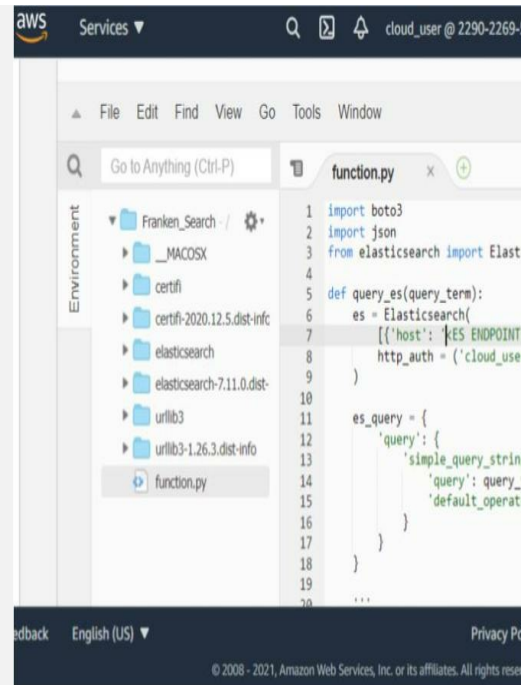
5. Here, you add the main functionality of the website.



6. The following provides a Github link of the code used in this lab: [https://github.com/AmazonWebServicesLabs/AWS-Certified-Data-Analytics---Speciality/blob/master/Lab\\_Assets/implementing\\_an\\_elasticsearch\\_lambda\\_function.py](https://github.com/AmazonWebServicesLabs/AWS-Certified-Data-Analytics---Speciality/blob/master/Lab_Assets/implementing_an_elasticsearch_lambda_function.py)



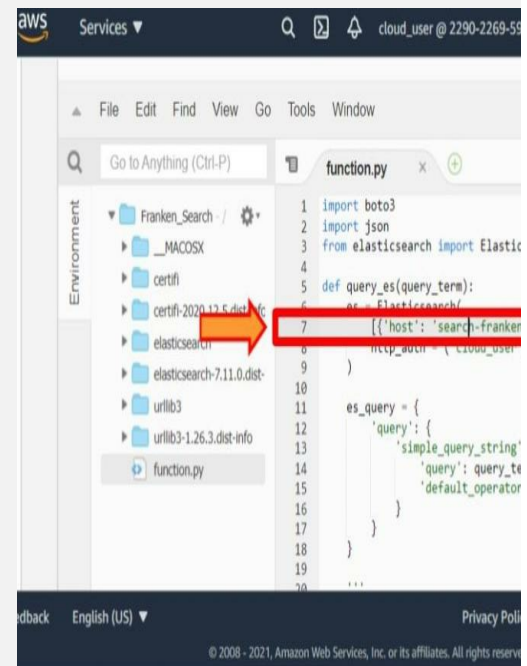
7. Copy and paste the code in the Lambda function code editor. Then click on the 'Save' button.



The screenshot shows the AWS Lambda console interface. The top bar includes the AWS logo, 'Services', a search icon, and the user 'cloud\_user @ 2290-2269-'. Below this is a menu bar with 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', and 'Window'. A search bar labeled 'Go to Anything (Ctrl-P)' is present. The left sidebar shows the 'Environment' tab with a tree view of the function's dependencies, including 'Franken\_Search - /', '.\_MACOSX', 'certifi', 'certifi-2020.12.5.dist-info', 'elasticsearch', 'elasticsearch-7.11.0.dist-info', 'urllib3', and 'urllib3-1.26.3.dist-info'. The main editor area shows the code for 'function.py'.

```
1 import boto3
2 import json
3 from elasticsearch import Elasticsearch
4
5 def query_es(query_term):
6     es = Elasticsearch(
7         [{'host': 'ES ENDPOINT'}],
8         http_auth = ('cloud_user', 'password')
9     )
10
11     es_query = {
12         'query': {
13             'simple_query_string': {
14                 'query': query_term,
15                 'default_operator': 'AND'
16             }
17         }
18     }
19
20     response = es.search(index='franken_search', body=es_query)
```

8. Copy the ES endpoint and paste it over the <ES ENDPOINT> placeholder in the code.



This screenshot is similar to the previous one, but with a red rectangular box highlighting line 7 of the code: `[{'host': 'ES ENDPOINT'}]`. An orange arrow points from the 'elasticsearch' dependency in the left sidebar to the red box. The rest of the interface is identical to the previous screenshot.

9. This code snippet is used to import the necessary python libraries and create an Elasticsearch client.

78 lines (67 sloc) | 2.01 KB

```
1 import boto3
2 import json
3 from elasticsearch import Elasticsearch
4
5 def query_es(query_term):
6     es = Elasticsearch(
7         [{'host': '<ES ENDPOINT>', 'port': 443
8         http_auth = ('cloud_user', 'Strongpass
9     )
10
11     es_query = {
```

10. This code snippet initializes the client.

```
1 import boto3
2 import json
3 from elasticsearch import Elasticsearch
4
5 def query_es(query_term):
6     es = Elasticsearch(
7         [{'host': '<ES ENDPOINT>', 'port': 443, 'use
8         http_auth = ('cloud_user', 'Strongpass1!')
9     )
10
11     es_query = {
12         'query': {
13             'simple_query_string': {
```

11. This query in the code brings the data into the API.

```
11     es_query = {
12         'query': {
13             'simple_query_string': {
14                 'query': query_term,
15                 'default_operator': 'and'
16             }
17         }
18     }
```

12. This code snippet sends the request to the Elasticsearch domain

```

41 es_response = json.loads(json.dumps(es.search(index = 'frankenstein', body = js
42 print(es_response)
43

```

13. This code snippet extracts the data from the Elasticsearch dom

```

43
44 response = {
45     'total_hits': es_response["hits"]["tot
46     'chapter_count': 0,
47     'total_score': 0.0,
48     'hits': []
49 }
50

```

14. This code snippet separates keys and values from the python returns the response.

```

51 for hit in es_response['hits']['hits']:
52     response['chapter_count'] += 1
53     for key, value in hit['_source'].items():
54         if key != 'Text':
55             location = f'{key} {value}'
56             score = hit['_score']
57             response['hits'].append(
58                 {
59                     'location': location,
60                     'score': score
61                 }
62             )
63     response['total_score'] += hit['_score']
64
65 response['total_score'] = round(response['total_score'], 2)
66
67 # print(json.dumps(response, indent = 4) )
68
69 return response
70

```

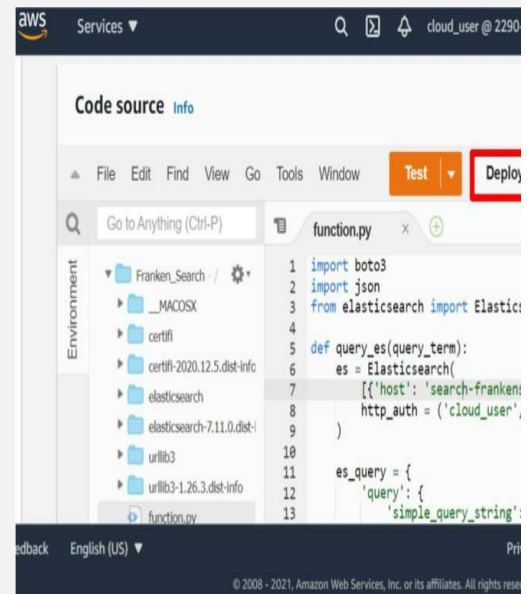
15. This code snippet converts the response into a JSON string and

```

71
72 def handler(event, context):
73     return {
74         'isBase64Encoded': False,
75         'statusCode': 200,
76         'body': json.dumps(query_es(event['queryStringParamete
77         'headers': {"Access-Control-Allow-Origin": ""})
78     }

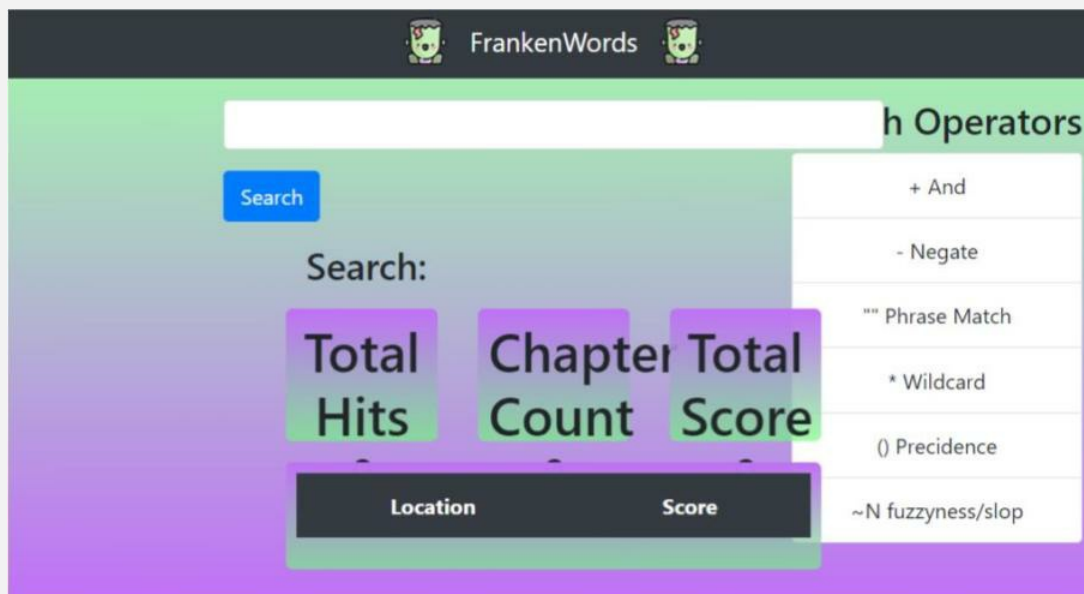
```

16. Click on the “**Deploy**” button to save the changes.

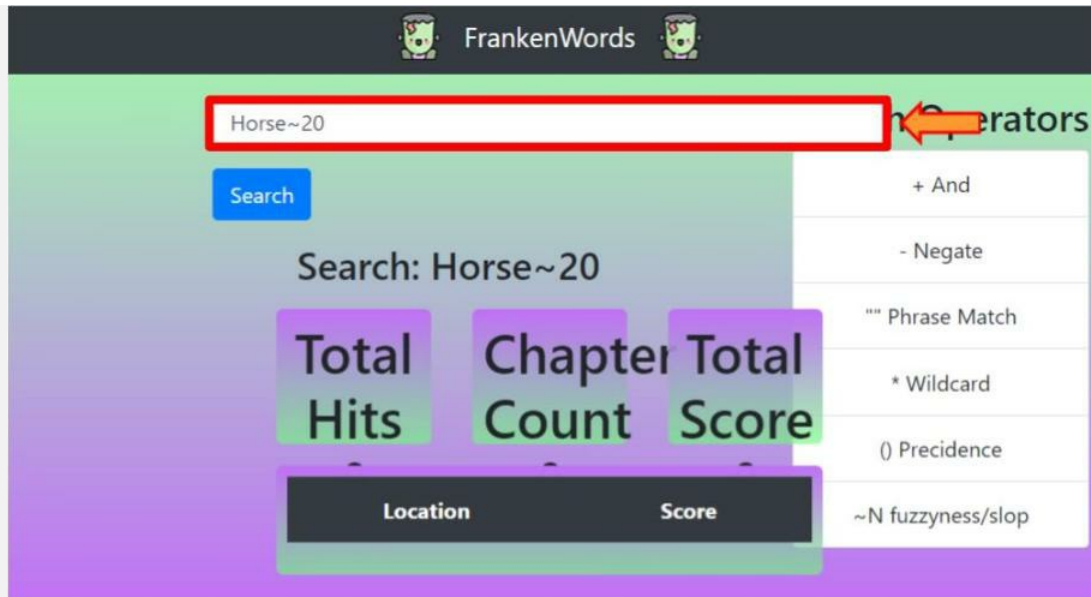


### Step 3: Test the Website

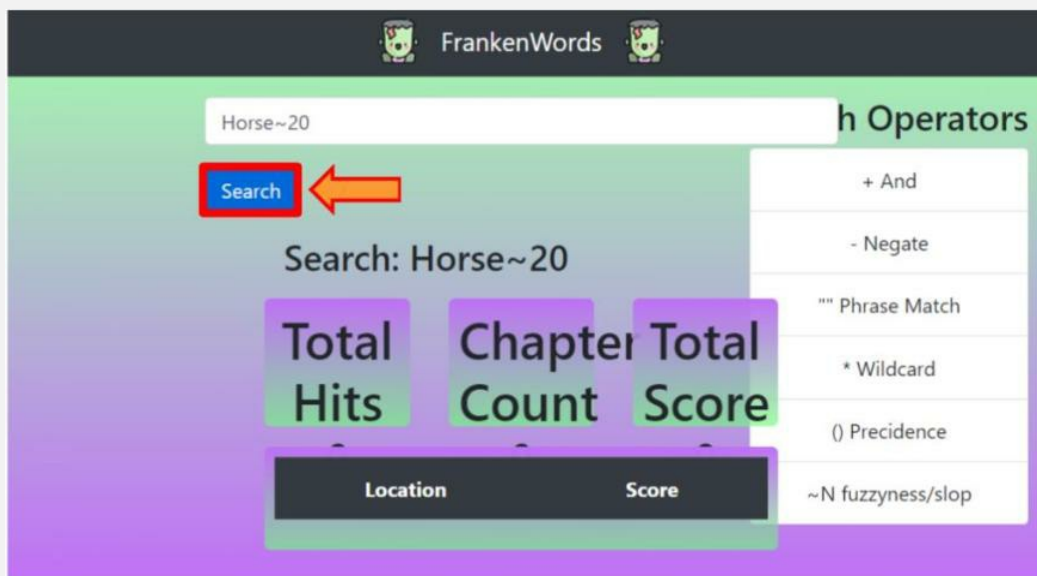
1. Go to the website by entering the website URL.



2. Enter the “**Horse~20**” into the search bar.





3. Click on the “**Search**” button.



4. The search should return 29 hits. You can also view how many chapters contained the term Horse and the term's total score.

| Location   | Score      |
|------------|------------|
| Chapter 24 | 4.5977798  |
| Chapter 5  | 3.2960591  |
| Chapter 16 | 1.672812   |
| Letter 4   | 1.3409745  |
| Chapter 6  | 0.9111862  |
| Chapter 23 | 0.8261573  |
| Chapter 8  | 0.8251403  |
| Chapter 21 | 0.81791276 |
| Chapter 15 | 0.6947806  |

5. Enter “**Food~20**” into the search bar.


FrankenWords


Operators

Search: Food~20

Total Hits

Chapter Count

Total Score

| Location   | Score     |
|------------|-----------|
| Chapter 24 | 4.5977798 |

+ And

- Negate

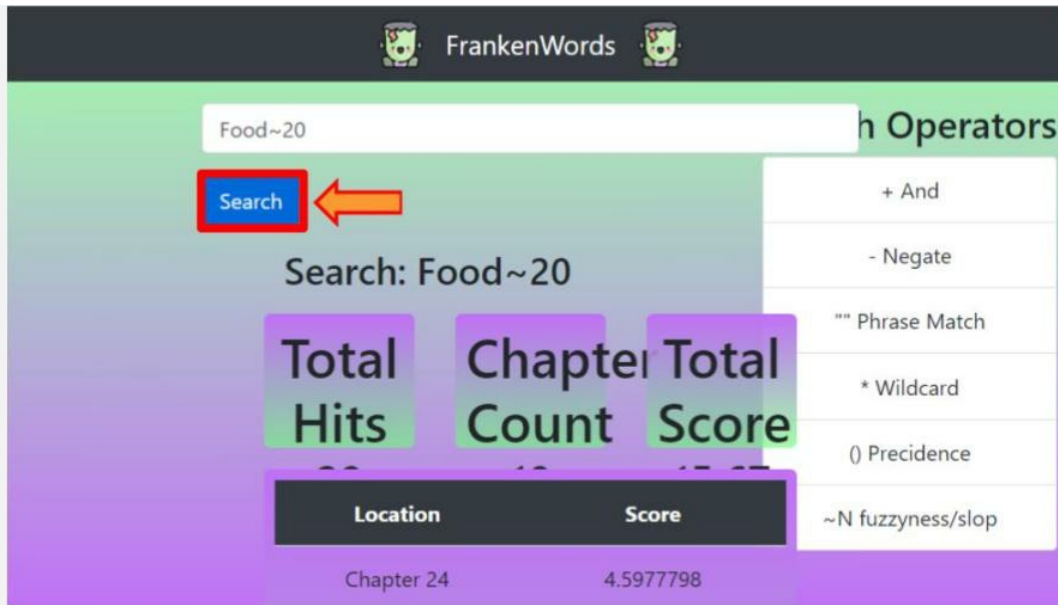
"" Phrase Match

\* Wildcard

() Precedence

~N fuzzyness/slop

6. Click on the “**Search**” button.

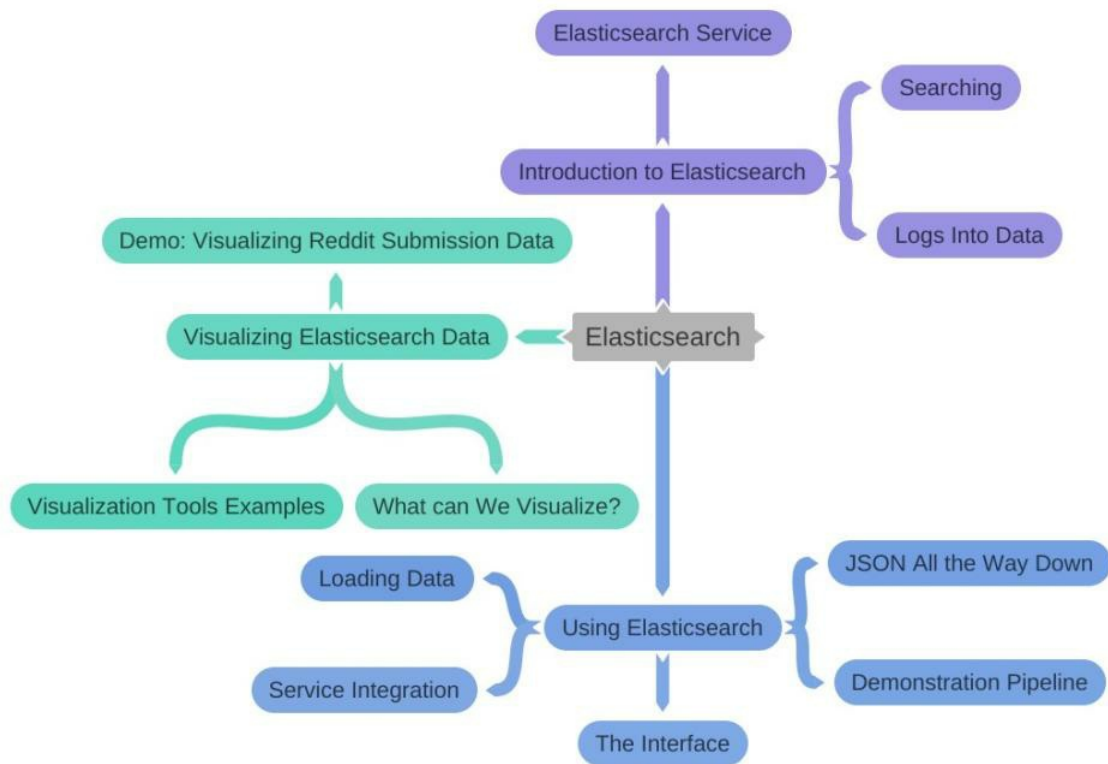


7. Compare the results to your previous results. Based on the total score, horses are described more often than food.

| Location   | Score      | ~N fuzzyness/slop |
|------------|------------|-------------------|
| Chapter 16 | 2.0311203  |                   |
| Chapter 5  | 1.9533178  |                   |
| Letter 4   | 1.9219112  |                   |
| Chapter 24 | 1.9218595  |                   |
| Chapter 6  | 1.448056   |                   |
| Chapter 12 | 1.0243163  |                   |
| Chapter 15 | 1.0188768  |                   |
| Letter 3   | 0.9802748  |                   |
| Chapter 2  | 0.88773155 |                   |

8. Hence, this is how you can add the functionality of statistical analysis of words and phrases to any website.

# Mind Map



*Figure 10-08: Mind Map*

## Practice Questions

1. Which service is a search domain that runs most of the ELK stack?  
A. Amazon ElasticSearch service  
B. EC<sub>2</sub>  
C. Athena  
D. None of the above
2. Which of the following is a Top-level organizational unit?

- A. Type
- B. Index
- C. Document
- D. None of the above

3. Which of the following is the second level used to categorize data?

- A. Index
- B. Document
- C. Type
- D. None of them

4. Which of the following is Elasticsearch data object?

- A. Index
- B. Document
- C. Type
- D. None of the above

5. Anything that can interact with the API can send data into \_\_\_\_\_.

- A. Amazon Elasticsearch
- B. EC<sub>2</sub>
- C. Athena
- D. None of the above

6. Which of the following Logs subscription can deliver to Elasticsearch Service?

- A. EC<sub>2</sub>
- B. Amazon Elasticsearch
- C. CloudWatch
- D. None of the above

7. Which of the following rules can send data to Elasticsearch Service?

- A. IoT
- B. Kinesis Data Firehose
- C. CloudWatch
- D. None of the above

8. Which of the following uses JSON for everything?

- A. EC<sub>2</sub>
- B. Elasticsearch
- C. CloudWatch
- D. None of the above

9. The Interface includes \_\_\_\_\_.

- A. REST interface

- B. GET
- C. PUSH
- D. DELETE
- E. POST
- F. All of them

10. Which of the following has built-in ES integration?

- A. Kinesis Data Firehose
- B. IoT
- C. CloudWatch
- D. None of the above



## Chapter 11: AWS Security Services

### Introduction

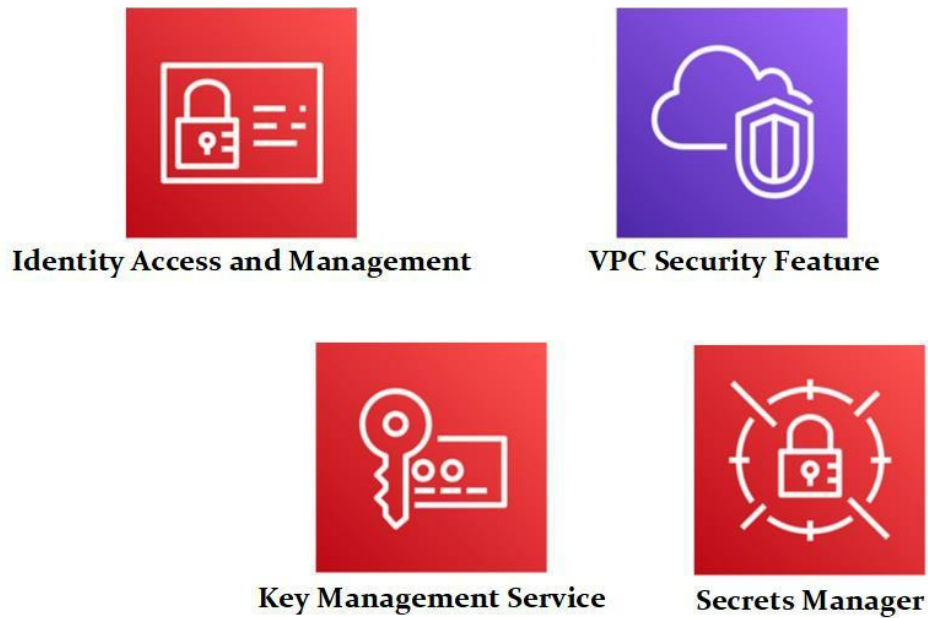
In this chapter, we are going to discuss AWS security services. Security services do not fit in any of the steps in the data analytics steps for success, but we need to secure our data. Data is the lifeblood that makes our application work. It is the most valuable thing in any application. It is what allows us to generate revenue from our application. Without data, our application does nothing.



*Figure 11-01: Introduction to AWS Security Services*

### Overview

The AWS security services are Identity and Access Management or IAM, VPC security features, the Key Management Service, and Secrets Manager. All of these have some integration or functionality that allows us to secure the other services.



*Figure 11-02: AWS Security Services*

### 1. Identity Access Management

Identity and Access Management, or IAM, allows us to control access to other AWS services. Any API call that you make is going to pass through Identity and Access Management. Several of the services you use may not use IAM for their primary authentication on their primary interface. You can use IAM users or groups to control access to RDS instances, Redshift clusters, or Elasticsearch service domains, and it also provides an external identity federation. Hence, if you have an active directory in an on-premises system, you can integrate that with IAM and allow Active Directory users to access AWS services with their Active Directory credentials.

### 2. VPC-Security

VPC security features are about network layer security hence security groups and network ACLs will allow us to control network traffic flow to and from our resources within our VPC.

### **3. Key Management Service**

Key Management Service is another very important service. It allows you to store your encryption keys as well as generate encryption keys in conjunction with IAM and CloudTrail. You can log and audit that key usage.

### **4. Secrets Manager**

Secrets Manager is a great service that acts as a vault for our passwords and API keys. You can use it to rotate those security objects in several of our services. You can manage who has access to those security objects through IAM. Hence, you can say this IAM group can use this set of API keys or passwords so on and so forth.

## **Identity Access Management**

### **Introduction**

AWS Identity and Access Management (IAM) is a web service that provides secured control access to AWS resources such as compute, storage, database, and application services in the AWS Cloud. IAM manages authentication and authorization by controlling who is signed in and has permission to utilize the resources. IAM uses access control concepts such as Users, Groups, Roles, and Policies to control which users can access specific services, the kinds of actions they can perform, and which resources are available. The IAM service is provided at no additional cost. However, your account will be charged upon the usage of other AWS services by your users.

When you establish an AWS account for the first time, you start with a single sign-in identity that has full access to all AWS services and

resources in the account. This identity is called the AWS account root user and is accessed by signing in with the email address and password you used to create the account. Since these credentials provide full access to the AWS account, it is strongly advised to use the Root User exclusively to create other users for persons within your business. Ensure the Root User account credentials are kept safe and used only for a few accounts and service management tasks.

**EXAM TIP:** Understand what the Root User is and know its privileges. As it always has full administrator access, you should never provide these account credentials to anybody. Instead, create a user for each employee in your business and always safeguard this root account with multi-factor authentication.

## **IAM Concept**

With IAM, you have Amazon Web Services that is at its core one big API. Hence, when you send a request to that API, it is then passed through IAM to check the permissions. If you present credentials that permit us, our request is sent to whatever service you are making an API call for. If you do not have permission, then you get a response that tells us no.

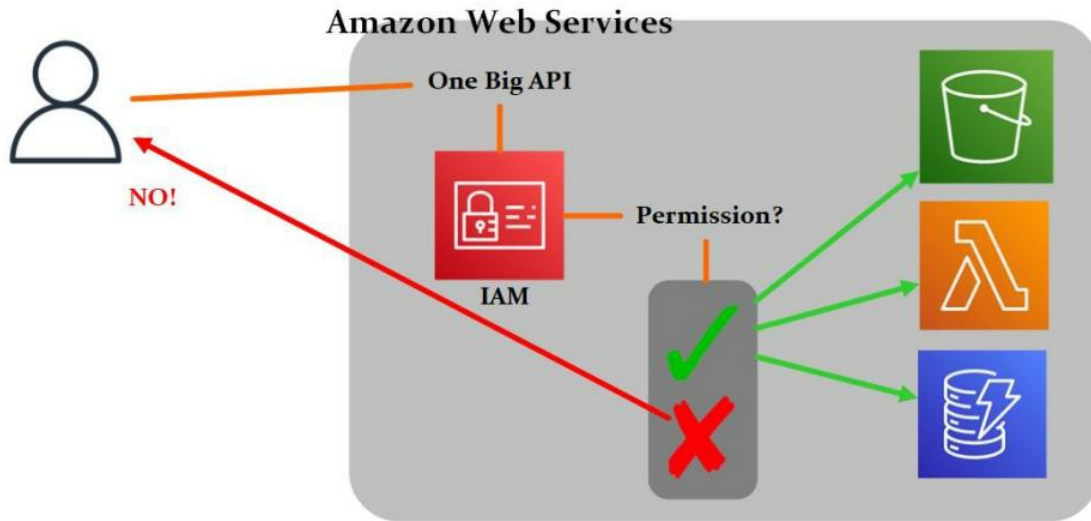


Figure 11-03: IAM Overview

## **IAM Permission Objects**

Policies govern AWS access by establishing and applying them to IAM identities (users, groups of users, or roles) or AWS resources. A policy in AWS is an object that specifies the rights of identity or resource when it is associated with it. AWS checks these policies when an IAM principal (user or role) submits a request. The permissions granted by the policies determine whether the request is authorized or refused. The bulk of AWS rules is stored as JSON documents. AWS supports all policy types, including identity-based policies, resource-based policies, permission boundaries, organizations SCPs, ACLs, and session rules.

When you get into that permissions area, things get a little more complicated. Hence, you will start with a policy in the IAM permission objects, which dictates what permissions you have. It dictates which pieces of the API you have access to. You can attach those policies directly to a user. It is the individual user. Therefore, when you log into the AWS console, you are logging in as an IAM user in most cases. That IAM user holds the information associated with the credentials you

provide for it. You can set those users to have just console access, or they can have programmatic access through access and secret keys generated from IAM. Those users can be placed in groups, and then you can use that to connect our policies to more than one user at a time. You can manage that policy attachment easier because you cannot attach any policies to our users and just place them in groups. That way, you can create policies that apply to groups of users instead of individual users. You also have roles. Roles are similar to users, but users or services can assume them.

Assume you have a Lambda function that needs to make calls to DynamoDB. That Lambda function needs to present the AWS API credentials that give it access to DynamoDB. Hence, it is going to assume a role. The same can apply to users. It is often used for cross-account permissions.

## **IAM Features**

The IAM service is part of AWS's secure worldwide infrastructure. With IAM, you can create and manage users and groups, security credentials such as passwords, access keys, and permission policies to allow and deny access to the AWS resources.

### **1. IAM Users**

An IAM user is a unique identity with limited access to an AWS account and its resources, defined by their IAM permissions and policies. Users of IAM can represent a person, a system, or an application. IAM policies assigned to users must grant explicit permissions to services or resources before viewing or using them. IAM lets you create individual users within your AWS account and give them their username,

password, and access keys. Individual users can then log into the console using a URL that is specific to their account. You can also create access keys for individual users to make programmatic calls to access AWS resources. You can provide user access to any or all of the AWS services linked with IAM, or you can utilize IAM in combination with external identity sources, such as Microsoft Active Directory, AWS Directory Service, or log in with Amazon.

Suppose the users in your organization already have a way to be authenticated, such as by signing in to your corporate network. In that case, you do not have to create separate IAM users for them. Instead, you can federate those user credentials into AWS. It is suggested that you create an IAM user even for yourself and not use your AWS account credentials for day-to-day AWS access.

## **2. IAM Groups**

A group is a group of IAM users. You may utilize groups to establish permissions for a group of users, making it easier to manage their access. For example, you may create a group named Admins and assign administrators generally require the rights. Any user in that group has the default permissions granted to the group. Assume a new user joins your organization and requires administrator capabilities. In that instance, you may provide the necessary rights by adding the user to the relevant group. Similarly, suppose a user changes jobs within your firm rather than modifying that user's permissions; you may delete them from the old groups and add them to the new ones in such a case.

Some key aspects of Groups:

- A user's membership in a group can be added or revoked
- A user can be a member of many groups

- A group cannot be a member of another group
- Access control policies can be used to provide permissions to groups. It is easier to manage permissions for a group of users than to manage permissions for each user
- Groups do not have security credentials and cannot directly access online services; they exist primarily to make managing user permissions easier

**EXAM TIP:** A group is a collection of IAM users. The users will inherit all of the group's permissions.

### 3. IAM Roles

An IAM role is an IAM object that allows you to define a set of permissions for a user or service to access resources. However, the permissions are not connected to a specific IAM user or group. Instead, IAM users, mobile and EC2-based applications, and AWS services (such as Amazon EC2) can programmatically acquire a role. The role results in temporary security credentials that the user or application may use to make AWS programmatic requests. These temporary security credentials have a configurable expiration date and are automatically cycled.

You do not have to keep long-term credentials and IAM users for every entity that wants access to a resource when using IAM roles and ephemeral security credentials. A single instance cannot have several IAM roles assigned to it. You may, however, assign a single IAM role to numerous instances. As a result, roles are far more secure and easier to manage than access key ids and secret access keys.

Roles, like everything else in identity access management, are universal. Unlike users, you do not need to identify what region they are in.

**EXAM TIP:** IAM resources are global. The IAM Roles can be used across regions.

#### 4. IAM Policies

An IAM policy is a rule or group of rules defining actions that may not be done on an AWS resource. Policies are used to give permissions. When a policy is associated with an identity or resource, it defines the permissions for that identity or resource. When a user makes a request, AWS examines these rules. The policies' permissions decide whether the request is approved or rejected. AWS rules are maintained as JSON documents and can be either identity-based or resource-based.

Policies can be granted in several ways:

- Include a controlled policy. AWS offers several pre-defined rules, such as AmazonS3 Read-Only Access
- Include an inline policy; an inline policy is a handwritten policy
- Add the user to a group with suitable permission policies
- Clone an existing IAM user's permissions

IAM users, groups, and roles have no permissions by default. Use the AWS Management Console, the IAM API, or the AWS CLI to configure permissions to create and attach policies. Policies may be created and assigned to IAM users, groups, and roles by users who have the relevant rights.

We have customers (customer-managed policies) or AWS (AWS

managed policies) handle managed policies (AWS-managed policies). Managed policies are IAM resources that use the IAM policy language to describe permissions. You may create, amend, and administer them independently of the IAM users, groups, and roles to which they are linked. You may change that policy in a single location after attaching a managed policy to multiple IAM users, groups, or roles. The rights are automatically extended to all associated entities.

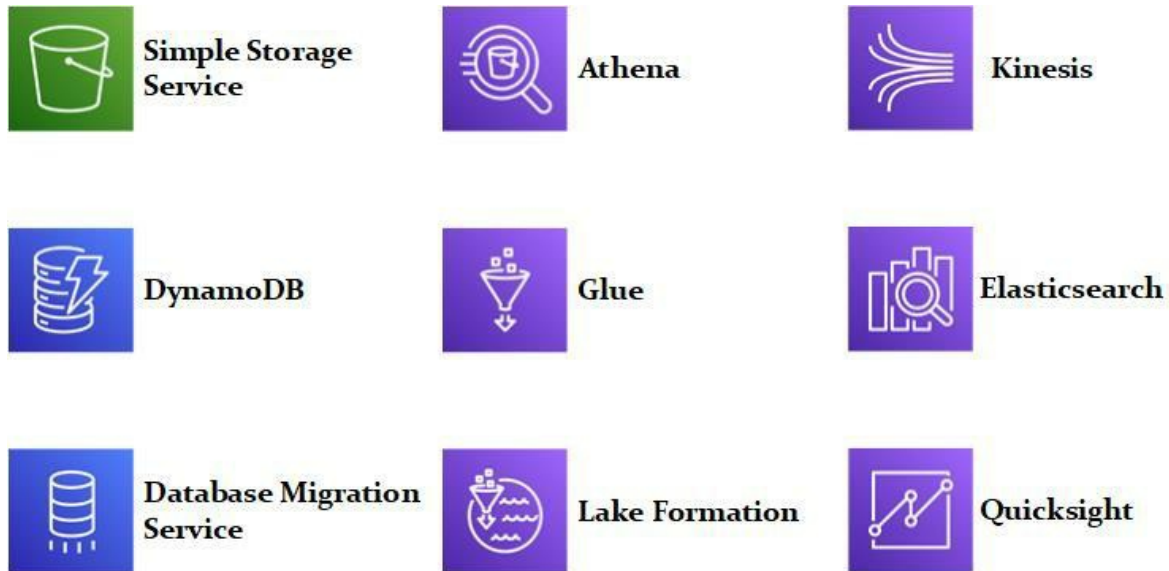
IAM groups may be used to organize IAM users and set common permissions for them. Managed policies may be used to distribute permissions among IAM users, groups, and roles. Suppose you want a group of users to launch an Amazon EC2 instance and the role on that instance to have the same rights as the group of users. In that case, you may build a managed policy and assign it to the group of users and the role on the Amazon EC2 instance.

**EXAM TIP:** To assign permissions to a group, you must first apply a policy to that group. JavaScript Object Notation is used to write policies (JSON).

## **IAM Secured Services**

There are a few examples of services that are secured with IAM. Simple Storage Service (S3) is often controlled with IAM. It also has its security features. S3 predates the AWS API. It was the second service launched as part of Amazon web services. Before the main larger API was created, bucket, object policies, and the ACLs are artifacts. DynamoDB access is controlled entirely with IAM. Database Migration Service, Athena, Glue, Lake Formation, Kinesis Elasticsearch, and QuickSight are also

controlled with IAM. Various code services are secured with IAM. There are more than the mentioned services that are secured with IAM. Anything that is primarily accessed through an API interface will flow through IAM for the most part.



*Figure 11-04: IAM Secured Service*

## External Identity Federation

In External Identity Federation, you can give access to our services to an outside identity provider. Hence, users in an Active Directory domain can federate IAM with SAML 2.0 or open ID connect. It lets us use any provider to create trust relationships with these protocols to access our IAM authentication framework. You also have the AWS single sign-on service, which works similarly to IAM. However, it stands up as an identity provider, which can be federated with external identity providers. Hence, this can be used as the primary source of our identity information that you will use to authenticate all of our various systems. You can federate with AWS single sign-on on-premise, AWS account, or other cloud providers with federation capabilities. It supports the

Octa Azure Active Directory in SAML 2.0 Federation protocols. You can use it as a single control plane to manage multiple AWS accounts and anything else that we can federate with an identity provider that uses one of the protocols it supports.

Suppose you currently handle user IDs outside of AWS. In that case, you may utilize IAM identity providers instead of creating IAM users in your AWS account. You may maintain your user identities outside of AWS using an identity provider (IdP) and grant these external user identities authorization to utilize AWS services in your account. It is handy if your business already has its identification system, such as a corporate user directory. It is also beneficial to develop a mobile app or a web application that utilizes AWS services.

When utilizing an IAM identity provider, you do not have to establish bespoke sign-in codes or manage your user IDs. The IdP handles that. External users sign in using a well-known IdP, such as Amazon, Facebook, or Google. You can grant such external identities access to your account's AWS resources. As you do not have to share or incorporate long-term security credentials, like access keys, IAM identity providers help keep your AWS account safe in your application.

Before you can utilize it, you must first build an IAM identity provider entity to establish trust between your AWS account and the IdP. IAM supports OpenID Connect (OIDC) and SAML 2.0 compatible IdPs (Security Assertion Markup Language 2.0).

## **Key Management System**

### **Introduction**

AWS Key Management Service (KMS) is a managed service that allows you to produce and govern the keys used in cryptographic activities. The service offers a highly available key generation, storage, administration, and auditing solution that allows you to encrypt or digitally sign data within your applications or govern data encryption across AWS services.

Suppose you are in charge of safeguarding your data across AWS services. In that case, you should utilize it to centrally manage the encryption keys that govern data access. Presume you are a developer who needs to encrypt data in your applications. In that case, you should use the AWS Encryption SDK in conjunction with AWS KMS to produce, utilize, and protect symmetric encryption keys in your code. Imagine you are a developer who wants to digitally sign or verify data using asymmetric keys. In that case, you should utilize the service to generate and maintain the necessary private keys. Assume you want scalable key management infrastructure to support your developers and their growing number of applications. In that case, you should utilize it to lower your licensing costs and operational strain. Suppose you are in charge of establishing data security for regulatory or compliance purposes. In that case, you should utilize it since it makes it easier to demonstrate that your data is constantly safeguarded. It is also subject to a wide range of industry and regional compliance requirements.

**EXAM TIP:** Suppose the database or service is located on a server farm where password changes take time to propagate to all member servers. In that case, the database may refuse calls that employ rotated credentials. You may reduce this risk by using an effective retry mechanism.

---

## KMS Concept

Let us assume you have super valuable data and want to turn it into encrypted data. It allows us to store and manage our encryption keys, and it can also be utilized to encrypt data. If our keys expire or become invalid, for some reason, you can use it to rotate those encryption keys. Doing this manually can be labor-intensive, which is part of why KMS exists. It is a service to facilitate these activities. KMS uses a customer master key. It can be generated within the KMS service itself, or you can supply a master key that you want to use. If you want to ensure you are in full control of our encryption material. It is the key that makes everything work.



*Figure 11-05: Key Concept*

## AWS KMS Keys

The fundamental resource in AWS KMS is AWS KMS keys (KMS keys). A KMS key can be used to encrypt, decrypt, and re-encrypt data. It can also create data keys for usage outside of AWS KMS. Typically, symmetric KMS keys are used, although asymmetric KMS keys can be created and used for encryption or signing.

A logical representation of an encryption key is an AWS KMS key. A

KMS key comprises metadata such as the key ID, creation date, description, and key status in addition to the key material required to encrypt and decode data.

The basic KMS key, also known as asymmetric KMS key, is a cryptographic key used for encryption and decryption. You may also generate an asymmetric KMS key (RSA or elliptic curve (ECC)) for encryption and decryption, as well as signing and verifying (but not both).

AWS KMS is used to generate KMS keys. Symmetric KMS keys and asymmetric KMS keys' private keys are never left unencrypted in AWS KMS. You must use AWS KMS to utilize or manage your KMS keys.

AWS KMS generates the key material for a KMS key by default. This critical material cannot be extracted, exported, viewed, or managed. Furthermore, you cannot remove this key material; instead, you must delete the KMS key. However, in the AWS CloudHSM cluster linked with an AWS KMS custom key store, you can import your key material into a KMS key or generate the key material for a KMS key.

AWS KMS now supports multi-region keys, which allow you to encrypt data in one AWS Region and decode it in another.

**EXAM TIP:** A logical representation of an encryption key is an AWS KMS key. A KMS key comprises metadata such as the key ID, creation date, description, and key status in addition to the key material required to encrypt and decode data.

## Customer Managed Keys

The customer handles the KMS keys you generate. Customer-managed

keys are KMS keys that you generate, own, and administer in your AWS account. You have entire authority over these KMS keys, including the ability to create and manage key policies, IAM policies, and grants, as well as activate and disable them. They rotate their cryptographic material, add tags, make aliases for the KMS keys, and schedule the destruction of the KMS keys.

Customer-controlled keys are displayed on the AWS Management Console's Customer-managed keys page for AWS KMS. Use the DescribeKey operation to definitively identify a customer-maintained key. The value of the KeyManager field of the DescribeKey response for customer-managed keys is CUSTOMER.

You may use your customer-managed key to perform cryptographic operations and audit use in AWS CloudTrail logs. Furthermore, many AWS services that integrate with AWS KMS allow you to generate a customer-managed key to safeguard data stored and maintained on your behalf.

**EXAM TIP:**

You may use your customer-managed key to perform cryptographic operations and audit use in AWS CloudTrail logs.

## **Symmetric KMS Keys**

When you generate an AWS KMS key, you are given a symmetric KMS key by default.

In AWS KMS, a symmetric KMS key is a 256-bit encryption key that never leaves the system unencrypted. It would help if you used AWS

KMS to utilize a symmetric KMS key. Symmetric keys are used in symmetric encryption, which employs the same key for encryption and decoding. Unless your job requires asymmetric encryption, symmetric KMS keys are a good option because they never leave AWS KMS unprotected.

AWS services that are connected with AWS KMS secure your data with symmetric KMS keys. These services do not support encryption with asymmetric KMS keys.

SYMMETRIC DEFAULT is the key specification for a symmetric key, ENCRYPT DECRYPT is the key use, and SYMMETRIC DEFAULT is the encryption technique.

You may encrypt, decrypt, and re-encrypt data in AWS KMS using a symmetric KMS key, as well as produce data keys and data key pairs. You can make multi-region symmetric KMS keys, import your key material into symmetric KMS keys, and make symmetric KMS keys in custom key stores.

#### **EXAM TIP:**

A symmetric KMS key in AWS KMS is a 256-bit encryption key never left unprotected in AWS KMS. It would be advantageous if you utilized AWS KMS to generate a symmetric KMS key.

## **Asymmetric KMS Keys**

AWS KMS allows you to generate asymmetric KMS keys. An asymmetric KMS key is a mathematically connected pair of public and private keys. The private key is never left unprotected in AWS KMS. It

would help if you used AWS KMS to utilize the private key. The public key can be used within AWS KMS by executing the AWS KMS API activities or downloaded and used outside of AWS KMS. Multi-Region asymmetric KMS keys can also be generated.

It is possible to produce asymmetric KMS keys that represent RSA key pairs for public-key encryption, signing and verification, or elliptic curve key pairs for signing and verification.

## **Data Keys**

Data keys are encryption keys that may be used to encrypt data, especially enormous volumes of data. Data keys are returned to you for use outside of AWS KMS instead of KMS keys, which cannot be downloaded.

When AWS KMS generates data keys, it provides you with a plaintext data key that you may use immediately away (optional), as well as an encrypted copy that you can preserve safely with the data. When you are ready to decrypt the data, ask AWS KMS to decode the encrypted data key.

AWS KMS is a service that produces, encrypts, and decrypts data keys. AWS KMS, on the other hand, does not store, manage, or monitor your data keys, nor does it execute cryptographic operations with data keys. Data keys must be used and managed outside of AWS KMS.

### **EXAM TIP:**

Data keys are encryption keys that may be used to encrypt data, particularly large amounts of data. Instead of KMS keys, which cannot be downloaded, data keys are provided to you for usage outside of AWS KMS.

---

## **Customer Master Key**

CMKs are classified into two types: AWS-managed and customer-managed. When you activate server-side encryption of an AWS resource for the first time under the AWS-managed CMK for that service, AWS-controlled CMK is produced (e.g., SSE-KMS). The AWS-managed CMK is specific to your AWS account and the region in which it is deployed. AWS-managed CMKs can only safeguard resources within the AWS service for which they were designed. It does not offer the granular control that a customer-managed CMK does. To get further control, employ a customer-managed CMK in all supported AWS services and your applications. A customer-managed CMK is generated at your request and should be set according to your specific use case.

## **Envelope Encryption**

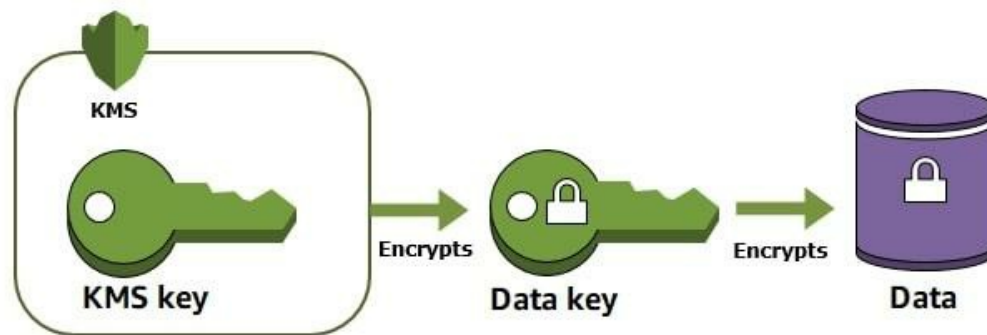
It is protected when you encrypt your data, but you must safeguard the encryption key. Encrypting plaintext data with a data key and then encrypting the data key with a separate key is known as envelope encryption.

You may even encrypt the data encryption key using another encryption key and then encrypt that encryption key again. However, one key must finally remain in plaintext so that you may decrypt the keys and your data. The root key is the top-level plaintext encryption key.



*Figure 11-06: Data Encryption Key under another Encryption Key*

AWS KMS protects your encryption keys by securely storing and managing them. Root keys held in AWS KMS, also known as AWS KMS keys, are never left unencrypted in the AWS KMS FIPS verified hardware security modules. To utilize a KMS key, you must first contact AWS KMS.



*Figure 11-07: Root Keys Stored in AWS KMS*

**EXAM TIP:** Encrypting plaintext material with a data key and then encrypting the data key with a separate key called envelope encryption.

Envelope encryption has various advantages:

- **Protecting Data Keys**

When you encrypt a data key, you do not have to worry about keeping the encrypted data key since encryption protects the data key by default. The encrypted data key can be safely stored alongside the encrypted data.

- **Encrypting the Same Data under Multiple Keys**

Encryption procedures can be time-intensive, especially if the data being encrypted is substantial. You can re-encrypt only the data keys that secure the raw data instead of re-encrypting raw data with multiple keys numerous times.

- **Combining the Strengths of Multiple Algorithms**

Symmetric key algorithms are quicker than public-key algorithms and create smaller ciphertexts. However, public-key algorithms allow intrinsic role separation and easier key management. Envelope encryption allows you to combine the advantages of each technique.

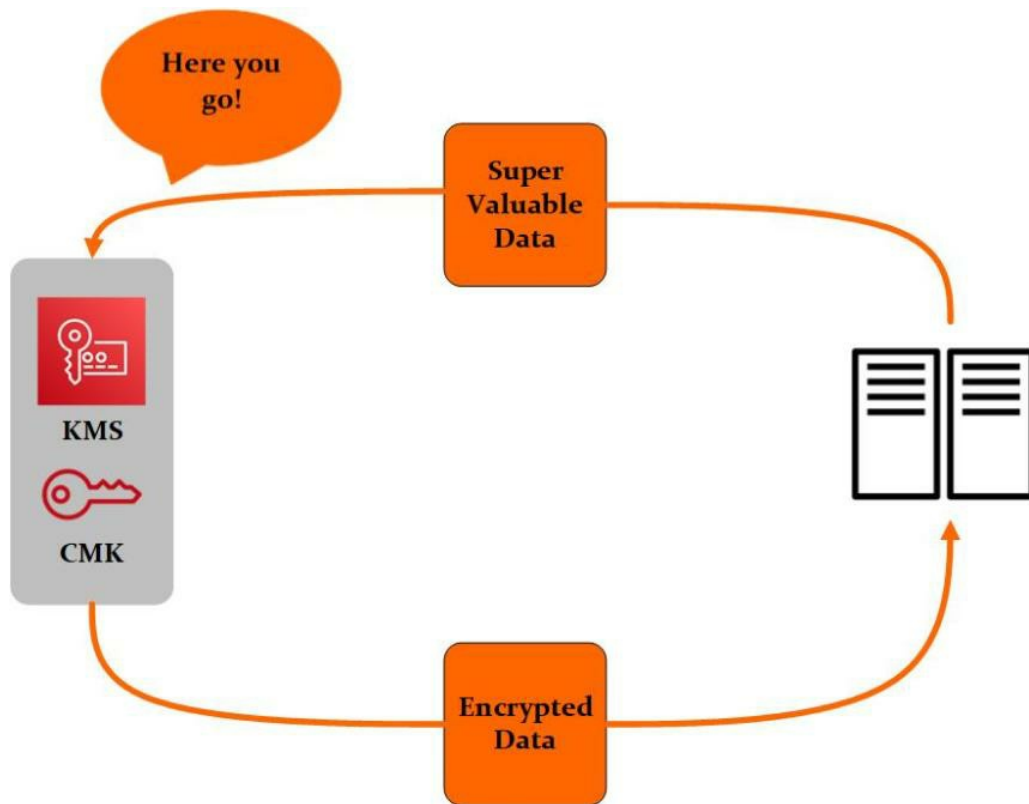
## **KMS Encryption Flows**

Following are the KMS encryption flows:

1. **Service Side Encryption**

KMS will have a customer master key, and you can send data to KMS. Assume you need to encrypt the data; KMS will encrypt the data and send it back. There is a flaw here, and if you encrypt many individual files, the network latency between our client and the KMS service will add up. The network latency from the requests to sending the file back will add up. It may only be tens of milliseconds, but if you are talking about thousands of files, that will add up quickly. It works well for infrequent encryption or situations where you do not necessarily have to compute available to encrypt our data, but if you do not want this

network latency in the mix, you can use Envelope Encryption.

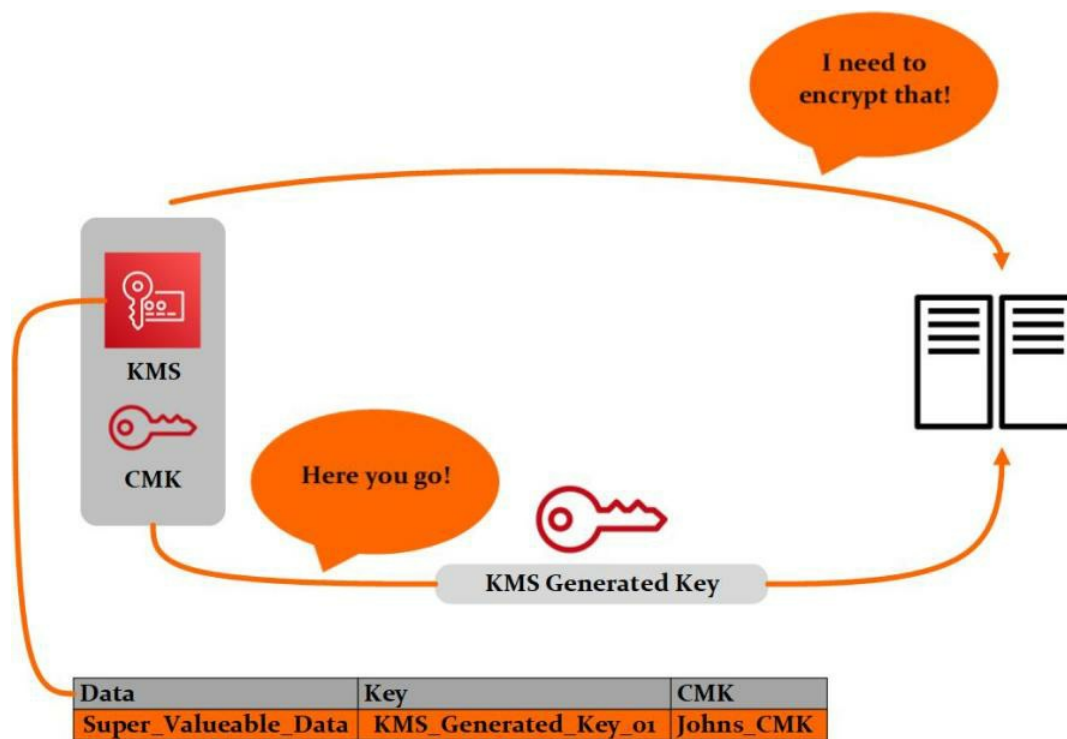


*Figure 11-08: Service Side Encryption*

## 2. Envelope Encryption

The other encryption flow you can use is called envelope encryption. Most of the services that will use KMS to encrypt data at rest are going to use Envelope encryption when they do encryption. If our client needs to encrypt data, KMS will generate a key, which it then encrypts with our customer master key and stores in the service. Then it generates a table of what data is encrypted with that token, not the actual data, and the key identifier for the key generated. The encryption key that is used to encrypt and decrypt that key from its storage then sends that generated key back. You can combine that with our super-valuable data and create our encrypted data. It means that if you are

encrypting a large number of files, you just need to collect one generated key. You can encrypt all our files, and you are good to go. You are just making a single trip to the KMS service and back. It saves a lot of time in the end, especially if you need to encrypt many individual files or a huge amount of data.



*Figure 11-09: Envelope Encryption*

## Secrets Manager

When you establish a custom application to obtain information from a database, you would often include the credentials, or secret, for directly accessing the database in the application. When it came time to rotate the credentials, you had to do more than just establish new ones. You needed to spend time updating the application to use the new credentials. The modified application was then distributed. If you have

numerous applications that share credentials and fail to update one of them, the application will fail. Due to this risk, many customers opt not to update credentials regularly, putting one risk in place of another.

Secrets Manager allows you to replace hardcoded credentials, such as passwords, in your code with an API call to Secrets Manager to get the secret programmatically. As the secret no longer resides in the code, this helps ensure that it cannot be compromised by someone reviewing your code. You may also set Secrets Manager to rotate the secret for you on a pre-defined period. It allows you to substitute long-term secrets with short-term ones, lowering the chance of compromise drastically.

**EXAM TIP:** Suppose the database or service is located on a server farm where password changes take time to propagate to all member servers. In that case, the database may refuse calls that employ rotated credentials. You may reduce this risk by using an effective retry mechanism.

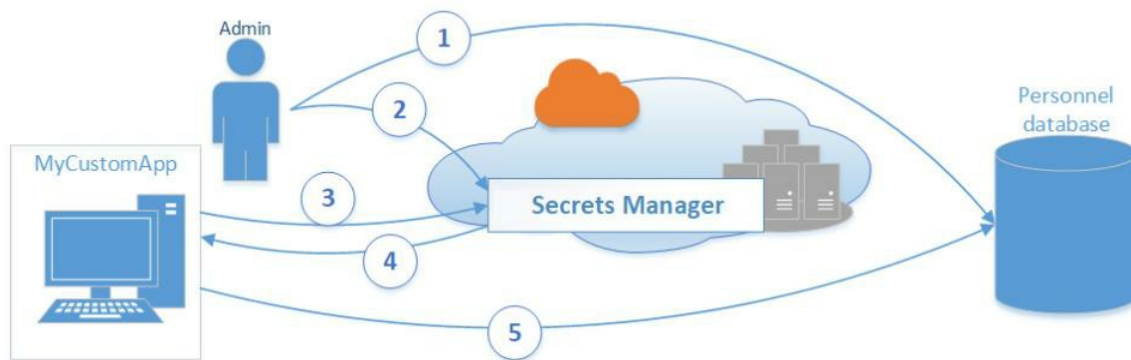
## Secrets Manager Concept

AWS Secrets Manager secures access to your applications, services, and IT resources without the upfront investment or ongoing maintenance expenses associated with running your infrastructure.

Secrets Manager is designed for IT administrators who need a safe and scalable way to store and manage secrets. Security administrators may use a secrets manager to meet regulatory and compliance standards to monitor and cycle secrets without interfering with applications. Secrets Manager may be retrieved programmatically by developers that want to replace hardcoded secrets in their applications.

It lets us store various text-based security tokens. You can control access to those tokens via IAM and encrypt those artifacts for an extra layer of security. The best security approach is always a multi-layered security approach. Hence, if you have very valuable secrets, you will want to encrypt them at rest as well. Using a secret vault-like HashiCorp Vault lets us keep all of our authentication material out of our code. Hence what you did write in our code that you are committing to a code repository is just the calls to Secrets Manager. You no longer have any secrets in our code to retrieve those secrets, and our code becomes more flexible. You do not have to update the code every time a password changes because the reference to Secrets Manager will point to that new password if you rotate it through Secrets Manager.

The figure below depicts the most basic case. The diagram shows how you can save database credentials in Secrets Manager and subsequently use those credentials in an application to access the database.



**Figure 11-10: AWS Secrets Manager Concept**

1. On the Personnel database, the database administrator generates a set of credentials for usage by an application called MyCustomApp. The administrator additionally configures those

credentials to grant the application access to the Personnel database.

2. The database administrator saves the credentials as a secret named `MyCustomAppCreds` in Secrets Manager. The credentials are then encrypted and stored as the protected secret text within the secret by Secrets Manager.
3. When `MyCustomApp` connects to the database, it searches Secrets Manager for the secret `MyCustomAppCreds`.
4. Secrets Manager collects the secret, decrypts the encrypted secret text, and provides it to the client app through a secure (HTTPS with TLS) connection.
5. The client application extracts the credentials, connection string, and any other necessary information from the response. It connects to the database server using it.

**EXAM TIP:** Secrets Manager supports a wide range of secrets. Secrets Manager can naturally rotate credentials for compatible AWS databases without further programming. Rotating secrets for other databases or services necessitates the creation of a new Lambda function to describe how Secrets Manager interacts with the database or service. To develop the function, you must have some programming knowledge.

## Secrets Manager Features

At runtime, encoded secret values can be retrieved programmatically. Secrets Manager improves your security posture by removing hard-coded credentials from your application source code and preventing credentials from being stored in any form within the application. Storing the credentials in or with the application exposes them to

compromise by anybody who has access to your program or its components. This technique makes rotating your credentials tough. Before you can depreciate the old credentials, you must update your application and distribute the updates to each client.

Secrets Manager allows you to acquire stored credentials by replacing them with a runtime call to the Secrets Manager Web service.

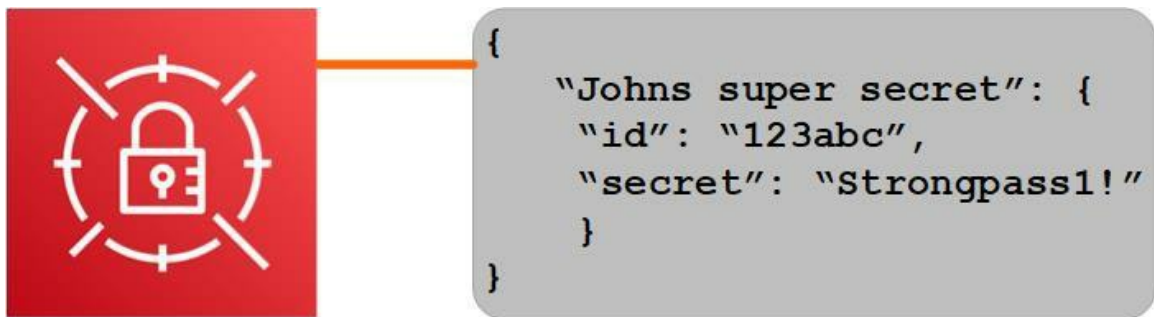
Your client usually requires the most recent version of the encrypted secret value. You can supply the secret name or Amazon Resource Name (ARN) without mentioning a version when obtaining the encrypted secret value. Secrets Manager will automatically return the most recent version of the secret value if you do this.

Most systems allow more complicated secrets than a simple password, such as whole sets of credentials that contain the connection information, the user ID, and the password. Secrets Manager enables you to keep numerous sets of these credentials in memory at the same time. Each set is saved in a distinct version of the secret in Secrets Manager. Other versions, however, may exist concurrently. Secrets Manager tracks the older credentials and the new credentials you want to start using during the secret rotation process until the rotation completes.

**EXAM TIP:** When AWS Secrets Manager rotates a secret, you may set up Amazon CloudWatch Events to get a notice. You may also use the Secrets Manager console or APIs to discover when Secrets Manager last rotated a secret.

## Secret Storage

The AWS secrets manager uses JSON to distort data, approximating what it might look like is shown in the figure below. It stores a string in the JSON. Hence those secrets can be passwords. They could be SSH keys. It could be API keys, really any string that fits within 64 kilobytes can be stored in Secrets Manager. You could have Base64 encoded strings that are stored in Secrets Manager that you know. You have to decode that Base64 encoding.



*Figure 11-11: Secret Storage*

## Different Secrets Types Store in AWS Secrets Manager

Secrets Manager allows you to store text in a secret encrypted secret data part. It normally comprises the database or service's connection information. This information may contain the server name, IP address, port number, and the user name and password used to access the service.

- Secret name and description
- Rotation or expiration settings
- ARN of the KMS key associated with the secret
- Any attached AWS tags

## Encrypt Secret Data

Secrets Manager encrypts a secret's protected text using AWS Key Management Service (AWS KMS). AWS KMS is used for key storage and encryption by many AWS services. When your secret is at rest, AWS KMS assures its safe encryption. Every secret is associated with a KMS key in Secrets Manager. It can be a customer-controlled key created in AWS KMS or an AWS-managed key for the account's Secrets Manager (aws/secretsmanager).

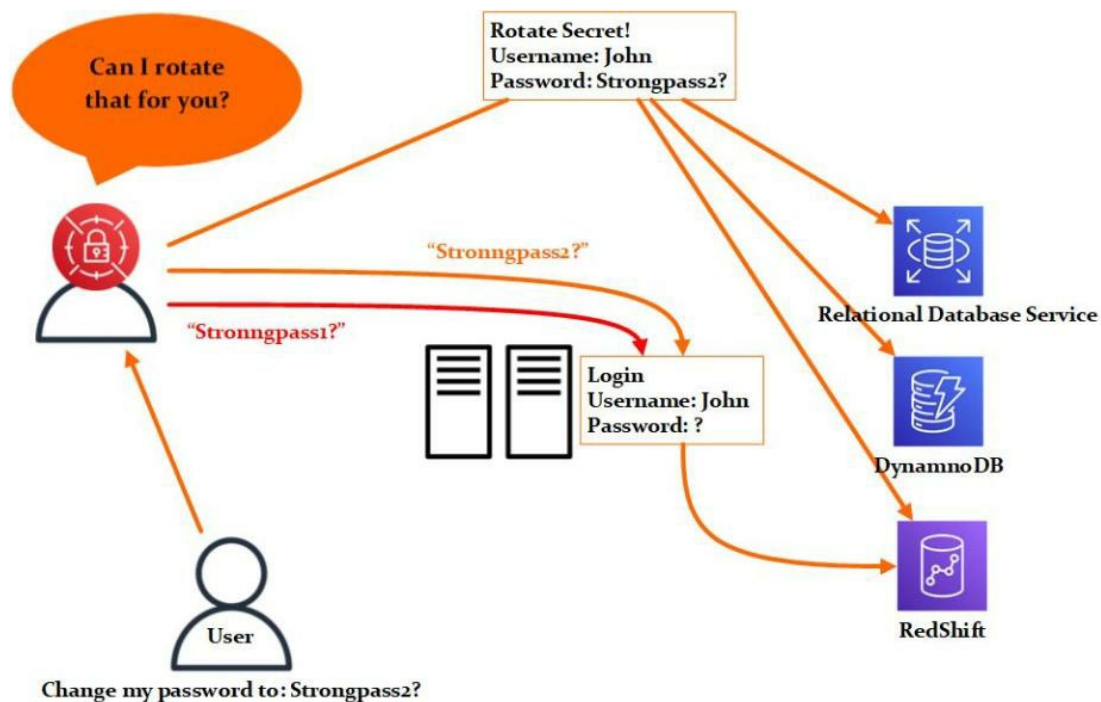
Secrets Manager requests AWS KMS to produce a new data key from the KMS key whenever it encrypts a new version of the protected secret data. Secrets Manager uses this data key for envelope encryption. Secrets Manager keeps the encrypted data key in the same folder as the protected secret data. Secrets Manager requests AWS KMS to decrypt the data key, which Secrets Manager then uses to decode the protected secret data whenever the secret requires decryption. Secrets Manager never saves the data key in an unencrypted state and always disposes of it soon after usage.

Furthermore, Secrets Manager only accepts requests from servers that use the open standard Transport Layer Security (TLS) and Perfect Forward Secrecy. Secrets Manager encrypts your secret as it travels between AWS and the systems from which you receive it.

**EXAM TIP:** Secrets Manager uses AWS Key Management Service to encrypt the protected text of a secret (AWS KMS). Many AWS services rely on AWS KMS for key storage and encryption.

## Secret Rotation

The process of updating a secret regularly is known as rotation. When you rotate a secret, the credentials in both the secret and the database or service are updated. You may set up an automatic rotation for your secrets in Secrets Manager. After rotation, applications that obtain the secret from Secrets Manager automatically receive the updated credentials.



*Figure 11-12: AWS Secret Manager Rotation*

**EXAM TIP:** AWS Secrets Manager allows you to arrange database credential rotation. It allows you to adhere to security best practices and properly rotate your database credentials. Secrets Manager utilizes the super database credentials you gave to create a clone user with the same rights but a new password when you start a rotation. The clone user information is then sent to databases and applications, which get database credentials.

## **Automatically Rotate Secrets**

Secrets Manager may be configured to automatically rotate your secrets on a pre-defined schedule and without human interaction.

Rotation is defined and implemented using an AWS Lambda function. This function specifies how Secrets Manager will carry out the following tasks:

- Creates a new version of the secret
- Stores the secret in Secrets Manager
- Configures the protected service to use the new version
- Verifies the new version
- Marks the new version as production-ready

Staging labels assist you in keeping track of the many versions of your secrets. Each version can have several staging labels associated with it. However, each stage label can only be associated with one version. Secrets Manager, for example, marks the presently active and in-use version of the secret with `AWSCURRENT`. Configure your applications to always request the most recent version of the secret. When the rotation process generates a new version of a secret, Secrets Manager adds the staging label `AWSPENDING` to the new version until testing and validation are completed. Secrets Manager only then assigns the `AWSCURRENT` staging label to this new version. When your applications inquire about the `AWSCURRENT` version, they instantly begin utilizing the new secret.

## **Rotation Strategies**

Secrets Manager has two rotation strategies:

1. **Single User Rotation Strategy**

The single-user technique refreshes one user's credentials in a single secret. It is the most basic rotation approach, and it is suitable for the majority of use scenarios. Single-user rotation can be used for:

- Using databases when a secret rotates, no database connections are lost, and new connections made after the rotation utilize the new credentials
- They access services that enable the user to create a single user account, such as using an email address as the user name. The service usually enables the user to change their password as frequently as they like. However, they cannot create new users or modify their user names
- Ad-hoc users are users who are generated as needed
- Users who submit their password interactively rather than having an application get it from Secrets Manager programmatically. This sort of user does not anticipate having to update their user name and password

There is a small amount of time between when the password in the database changes and when the related secret updates when this sort of rotation occurs. At this moment, there is a low chance that the database may reject calls that employ rotated credentials. You may reduce this risk by using an effective retry mechanism.

## **2. Alternating Users Rotation Strategy**

The alternating user's technique refreshes two users' credentials in a single secret. You make the first user, then rotate clones to make the second.

The other user is kept up to speed with each new version of a secret. If the first version contains `user1/password1`, the second version has `user2/password2`. `User1/password3` is used in the third version, while

user2/password4 is used in the fourth. You have two sets of valid credentials at any one time: the current and prior credentials.

While rotation produces the new version of the credentials, applications continue to utilize the current version. When the new version is complete, rotate the staging labels so that applications utilize the new version.

A separate secret holds credentials for an administrator or superuser who can create the second user and update the credentials of both users.

This method is suitable for:

- Applications and databases with permission models have access to the tables. One role owns the tables, and another role is in the application
- Applications that need a high level of availability-This sort of rotation has a lower risk of denying applications than single-user rotation

Suppose the database or service is located on a server farm where password changes take time to propagate to all member servers. In that case, the database may refuse calls that employ rotated credentials. You may reduce this risk by using an effective retry mechanism.

To employ this method, you will need a second secret with credentials for an administrator or superuser with the ability to create a user and change the password on both users. Guarantee that both users have identical rights, the first rotation clones this user to create the alternate user.

## **VPC Network Security Features**

Amazon VPC lets you provision a logically isolated section of the AWS cloud to launch AWS resources in a virtual network that you define. You have complete control over your virtual networking environment, including the ability to define route tables and network gateways, as well as choose IP address ranges and construct subnets.

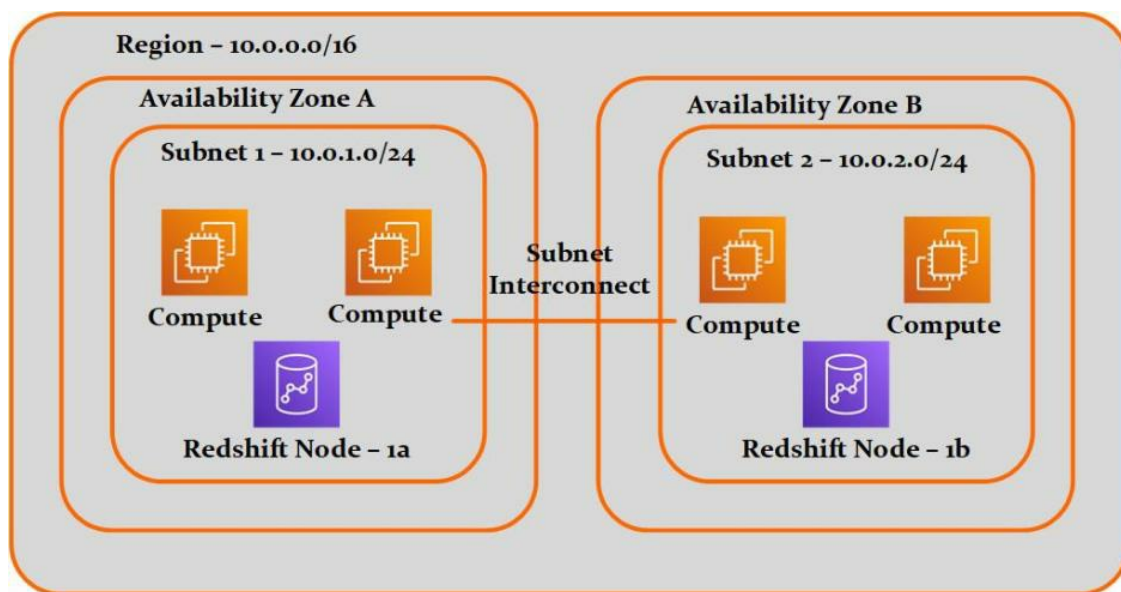
A Virtual Private Cloud (VPC) is a cloud computing concept that provides an on-demand modifiable pool of shared computing resources assigned inside a public cloud environment while also offering some level of separation from other public cloud users. As the cloud (pool of resources) in a VPC model is exclusively available to a single client, it provides privacy with more control and a safe environment where only the defined client may work.

The network settings for your Amazon VPC may be readily customized. For example, you may build a public-facing subnet with Internet connectivity for your web servers and a private-facing subnet with no Internet access for your backend systems such as databases or application servers. To help limit access to Amazon EC2 instances on each subnet, you may use various levels of security, such as security groups and network access control lists. You may also utilize AWS to extend your corporate data center by establishing a hardware Virtual Private Network (VPN) link between your corporate data center and your VPC.

**EXAM TIP:** To isolate cloud resources in a private virtual network, use Amazon VPC.

## VPC Concept

VPC is a network space that lives within a region. You define the top level of that network space with this CIDR range. Hence, in this case, 10.0.0.0/16, you have availability zones within that region, Availability zone A and availability zone B. Within those availability zones, you create subnets. Each of these gets a CIDR range, which is a subset of our primary VPC CIDR range. Subnet 1 is 1.0/24, and subnet 2 is 2.0/24. Those subnets within our VPC are all interconnected. If you have an internet gateway, they have a pathway out to the internet. These are considered subnet edges, and in the below figure, you are just showing the subnet interconnect. You then launch our resources within these subnets, and they are assigned IP addresses within the subnet CIDR range.



*Figure 11-13: VPC Concept*

## Components of VPC

- **Virtual Private Cloud:** VPC logically isolated virtual network in the AWS cloud. The IP address space of a VPC is defined by the ranges you choose

- **Subnet:** A segment of a VPC's IP address range to place groups of isolated resources
- **Internet Gateway:** The Amazon VPC side of a connection to the public Internet
- **NAT Gateway:** A highly available, managed Network Address Translation (NAT) service for your resources in a private subnet to access the Internet
- **Hardware VPN Connection:** A hardware-based VPN connection between your Amazon VPC and your data center, home network, or co-location facility
- **Virtual Private Gateway:** The Amazon VPC side of a VPN connection
- **Customer Gateway:** Your side of a VPN connection
- **Router:** Router interconnect subnets and direct traffic between Internet gateways, virtual private gateways, NAT gateways, and subnets
- **Peering Connection:** A peering connection enables you to route traffic via private IP addresses between two peered VPCs
- **VPC Endpoints:** Enables private connectivity to services hosted in AWS from within your VPC without using an Internet Gateway, VPN, Network Address Translation (NAT) devices, or firewall proxies
- **Egress-only Internet Gateway:** A stateful gateway provides egress-only IPv6 traffic from the VPC to the Internet

## Network Access Control List

A network access control list (NACL) is an optional security layer for your VPC that operates as a firewall to manage traffic in and out of one or more subnets. Set up network ACLs with rules similar to your security groups to provide your VPC with an extra layer of protection.

The first layer of network-level security is a Network Access Control

List or network ACL. These are attached at these subnet edges. They are stateless. They are attached at the subnet level. Hence anywhere there is a connection outside of the subnet, the Network Access Control List applies. You can create one network ACL that is attached to many subnets. Subnets can only have one ACL attached, but you can use that same ACL for multiple subnets.

Network ACLs use rules. They can be allowed or denied. You specify a source or destination. Hence, if it is an incoming rule, it will have a source, and if it is an outgoing rule, it will have a destination. Specify those with CIDR notation, and then the rule is going to specify a port range. You can also specify which protocol the rule applies to, and you can make rules for IPv4 and IPv6 traffic. The rules are checked in order. Hence they have a number in them, and you will see that in the figure below. When they are checked, it just goes down the list, and as soon as there is a match, it stops checking sends the traffic through, which makes them very efficient, but it can also make them difficult to work with.

Consider an example of a very basic network ACL. It is an incoming network ACL, and all you are allowing in is port 80. All other traffic is going to be denied. Also, note that you are only allowing IPv4 port 80 traffic or the TCP protocol.

| Rule# | Type             | Protocol | Port Range | Source    | Allow/Deny |
|-------|------------------|----------|------------|-----------|------------|
| 100   | All IPv4 Traffic | TCP      | 80         | 0.0.0.0/0 | ALLOW      |
| 900   | All Traffic      | UDP      | ALL        | 0.0.0.0/0 | DENY       |

*Figure 11-14: Network Access Control List*

**EXAM TIP:** An optional security layer that operates as a firewall to

regulate traffic in and out of a subnet. A single network ACL can be associated with several subnets. However, a subnet can only be associated with one network ACL at a time.

## Network ACL Concepts

The following are the fundamental concepts concerning network ACLs that you should understand:

- Your VPC includes a configurable default network ACL by default. Thus, it accepts all inbound and outgoing IPv4 traffic and IPv6 traffic if necessary
- A custom network ACL can be created and associated with a subnet. Unless you set rules, each custom network ACL prohibits all incoming and outgoing traffic by default
- Each subnet in your VPC must have its own network ACL. If a subnet is not explicitly assigned to a network ACL, it is assigned to the default network ACL
- A network ACL can be associated with many subnets. On the other hand, a subnet may only be linked with one network ACL at a time. The prior association is deleted when you pair a network ACL with a subnet
- An ACL on a network is a numbered set of rules. The maximum number that may be used for a rule is 32766. To determine whether traffic is authorized into or out of any subnet associated with the network ACL, we evaluate the rules in ascending order, beginning with the lowest numbered rule. We recommend that you begin by defining rules in increments (for example, increments of 10 or 100) so that you can add more rules as needed later
- An ACL on a network includes separate inbound and outgoing rules, and each rule can allow or refuse traffic

- Since network ACLs are stateless, answers to approve inbound traffic are subject to the rules for outgoing traffic (and vice versa)

The number of network ACLs per VPC and the number of rules per network ACL have limitations (limits).

## Network ACL Rules

You may modify the default network ACL by adding or removing rules, or you can establish new network ACLs for your VPC. When you update the rules in a network ACL, the changes are automatically applied to the subnets connected to it.

A network ACL rule is made up of the following components:

- **Rule Number:** Starting with the lowest-numbered, rules are reviewed in ascending order. When a rule matches traffic, it takes precedence over any higher-numbered rule that would contradict it
- **Type:** The kind of transmission, such as SSH. You can also specify all traffic or a specific range of traffic
- **Protocol:** Any protocol with a standard protocol number can be specified. See Protocol Numbers for further details. You can use any or all ICMP types and codes if you select ICMP as the protocol
- **Port Range:** The traffic's listening port or port range. For instance, 80 for HTTP traffic
- **Source:** [Only inbound rules] The origin of the traffic (CIDR range)
- **Destination:** [Only outbound rules] The traffic's final destination (CIDR range)
- **Deny/Allow:** Whether the given traffic should be allowed or denied

When you use a command-line tool or the Amazon EC2 API to add a rule, the CIDR range is automatically changed to canonical form. For example, if you give 100.68.0.18/18 for the CIDR range, we construct a rule with a CIDR range of 100.68.0.0/18.

## **Security Groups**

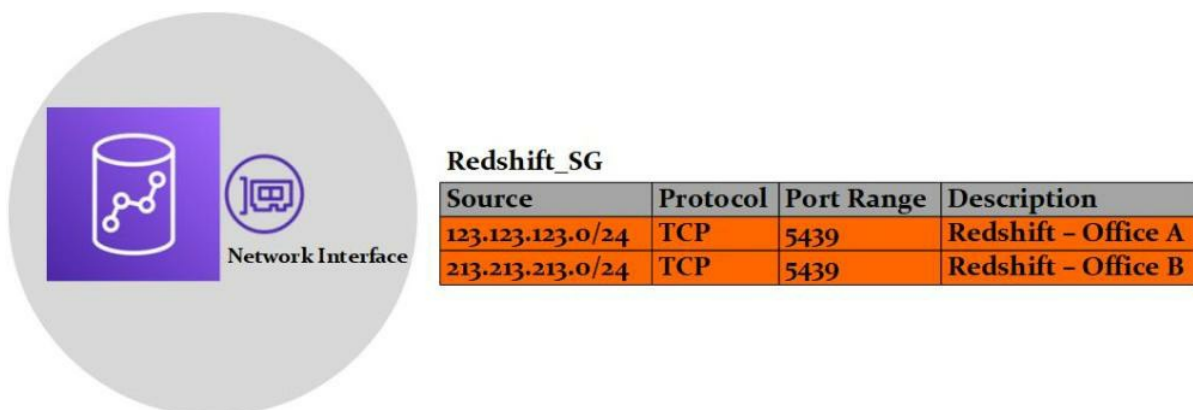
A security group acts as a virtual firewall, for your instance, controlling inbound and outgoing traffic. When you deploy an instance in a VPC, you may designate the instance up to five security groups. Security groups operate at the instance level rather than the subnet level. As a result, each instance in a VPC subnet can be allocated to a separate set of security groups.

Suppose you start an instance using the Amazon EC2 API or a command-line tool without specifying a security group. In that case, the instance is assigned to the VPC's default security group. When you start an instance using the Amazon EC2 interface, you may create a new security group, for example.

Suppose you start an instance using the Amazon EC2 API or a command-line tool without specifying a security group. In that case, the instance is assigned to the VPC's default security group. When you start an instance using the Amazon EC2 interface, you may create a new security group, for example. Set up network ACLs with rules similar to your security groups to provide your VPC with an extra layer of protection.

The next layer of security and security groups are attached at the resource network interface. Each resource within our VPC will have a network interface. They will have a security group attached to them.

Maybe the default, it may be one that you have defined yourselves. They only have allowed rules in them. They are, for the most part, stateful. They are going to keep track of the connections that are flowing through them. You can create inbound and outbound rules that specify the source or destination CIDR range. You can specify TCP, UDP, or ICMP protocols, and then you specify a port range. You also have the opportunity to describe each rule. They let us specify whether this is an IPv4 or an IPv6 rule through the source or destination. There is an example of a simple security group that is just allowing access to a Redshift cluster. Whatever Redshift cluster has. This security group assigned is going to allow access from these 2 IP ranges. Hence you have described them as Redshift office A and Redshift office B. The best practice is just to use security groups. They act as a firewall on each resource that has them assigned, and they are much more flexible and easier to work with than network ACLs.



*Figure 11-15: Security Group*

**EXAM TIP:** VPC's security groups define which traffic is permitted to or from an Amazon EC2 instance. ACLs in a network function at the subnet level evaluate traffic entering and departing a subnet. ACLs on the network may be used to create, allow and deny rules. ACLs on a

network do not filter traffic between instances on the same subnet. Furthermore, network ACLs use stateless filtering, whereas security groups use stateful filtering.

## **Traffic Monitoring**

Traffic Mirroring is an Amazon VPC feature that allows you to repeat network traffic from the elastic network interface of an Amazon EC2 instance. The traffic can then be routed to out-of-band security and monitoring equipment for:

- Content inspection
- Threat monitoring
- Troubleshooting

Individual instances of the security and monitoring appliances can be installed, as can a fleet of instances behind a Network Load Balancer with a UDP listener. Filters and packet truncation are included in Traffic Mirroring, allowing you to extract only the traffic of interest to monitor using your preferred monitoring tools.

In traffic monitoring, we have a few options available in VPC flow logs, which generate a log, as you would guess from the name of the service. These logs contain whether the traffic was allowed or denied, the addresses for the source and destination of that traffic, the ports being used, the protocol being used, and a count of the packets. You can also get a byte count of the bandwidth passed back and forth in that section. It is fairly general but can be very useful when troubleshooting bandwidth issues. Suppose traffic is being denied to a single node in a cluster. In that case, you can see in these logs that traffic from this

source to this destination is being denied over and over again. You can check our network ACLs, our security groups, and the configuration of those resources to find why that traffic is being denied. It is not super detailed, but you have a second option to get more detailed information about our network traffic. We can use a service called Traffic Mirroring. It mirrors the full packets from our connections, and you can specify those connections. You can send those packets to an elastic network interface or network load balancer. Whatever is behind that is going to receive those packets, it can capture and process them. It works similarly to TCP Dump. It is just configured through the API to target it at a source, destination, and the session hence the source. You can filter it like you would TCP Dump to port numbers or protocols or whatever filtering you want to include.

**EXAM TIP:** Traffic Mirroring is an Amazon VPC feature that allows you to repeat network traffic from the elastic network interface of an Amazon EC2 instance.

## Lab 11-01: Advanced S3 Security Configuration

### Introduction

#### Amazon Simple Storage Service

Amazon S3 is a type of object storage that allows you to store and recover any quantity of data from any location. It is a low-cost storage solution with business resilience, reliability, efficiency, privacy, and infinite expansion.

Amazon S3 is a web service that allows you to store and retrieve an

infinite quantity of data from any place and at any time. You may quickly create projects that integrate cloud-native storage using this service. As Amazon S3 is easily customizable and you only pay for what you use, you can start small and scale up as needed without sacrificing performance or dependability.

Amazon S3 is also built to be highly adaptable. Instead of finding out how to store their data, Amazon S3 allows developers to focus on innovation. Build a simple FTP system or a complex web application like the Amazon.com retail website. Read the same piece of data a million times or only for emergency disaster recovery; store whatever type and amount of data you desire.

### **AWS Identity and Access Management**

Individuals and groups can be granted secure access to your AWS resources by using IAM. It allows you to create and manage IAM users and provide them access to your resources. Additionally, you have the option of granting access to users outside of AWS (federated users).

- **Managed Policy:** This contains the permission required to stop an EC2 instance
- **Inline Policy:** This allows this role to be passed to another service
- **Trust Policy:** Allows System Manager and EC2 to assume the role. It enables EC2 to register with the Systems Manager and Systems Manager to stop the EC2 instance

### **Amazon Elastic Compute Cloud**

Amazon Elastic Compute Cloud (Amazon EC2) is a cloud computing service that gives the durability of computing power. It is intended to make web-scale computing more accessible to IT engineers.

Amazon EC2 offers “compute” in the cloud, such as Amazon Simple Storage Service (Amazon S3) enables “storage” in the cloud. The easy web service interface of Amazon EC2 allows you to obtain and configure capacity quickly. It lets you control your computer resources entirely and enable you to run on Amazon’s tried-and-true computing infrastructure. The time to buy and boot new Amazon EC2 server instances is reduced to minutes, allowing you to scale up quickly and down capacity as your computing needs change. Amazon EC2 revolutionizes computing economics by enabling you to pay only for the resources you utilize.

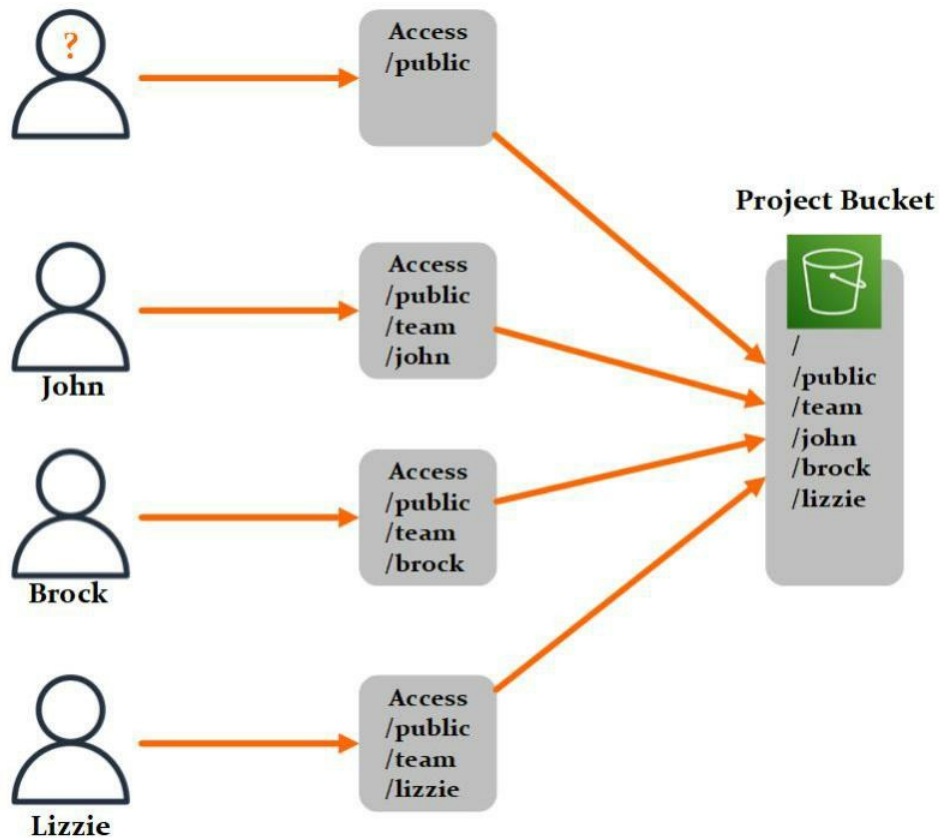
## **Problem**

Assume you are a Database Administrator in an organization. The organization gives you a task to create a secure cloud storage management system in a cloud. In the system, the public could not access team members’ data. Also, each member of the team could not access each team members’ data, and they only access the team data on the main team folder. Hence how can you create this type of secure cloud storage management system?

## **Solution**

The solution is you use the AWS service. You use the AWS S3 to create the storage management system for storing the data. For blocking public access to the main team bucket, you should use IAM roles and bucket policies to block the access.

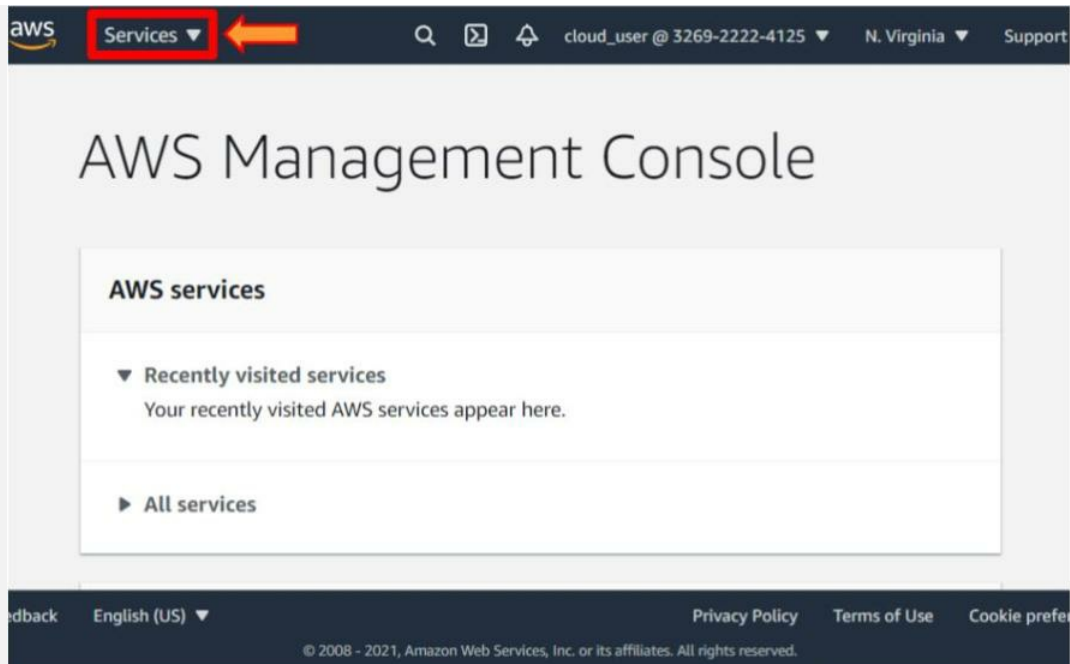
|                                                                                                       |
|-------------------------------------------------------------------------------------------------------|
| <p><b>Note:</b> Before starting the lab, create the AWS S3 bucket and IAM roles used in this lab.</p> |
|-------------------------------------------------------------------------------------------------------|



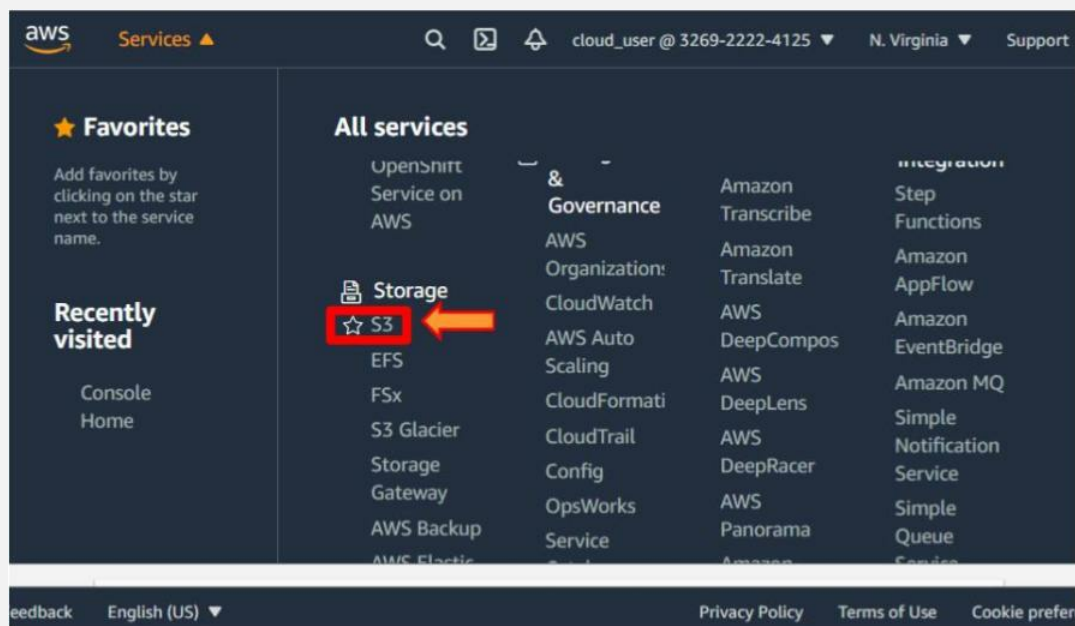
*Figure 11-16: Advanced S3 Security Configuration*

### Step 1: Investigate the Lab Services

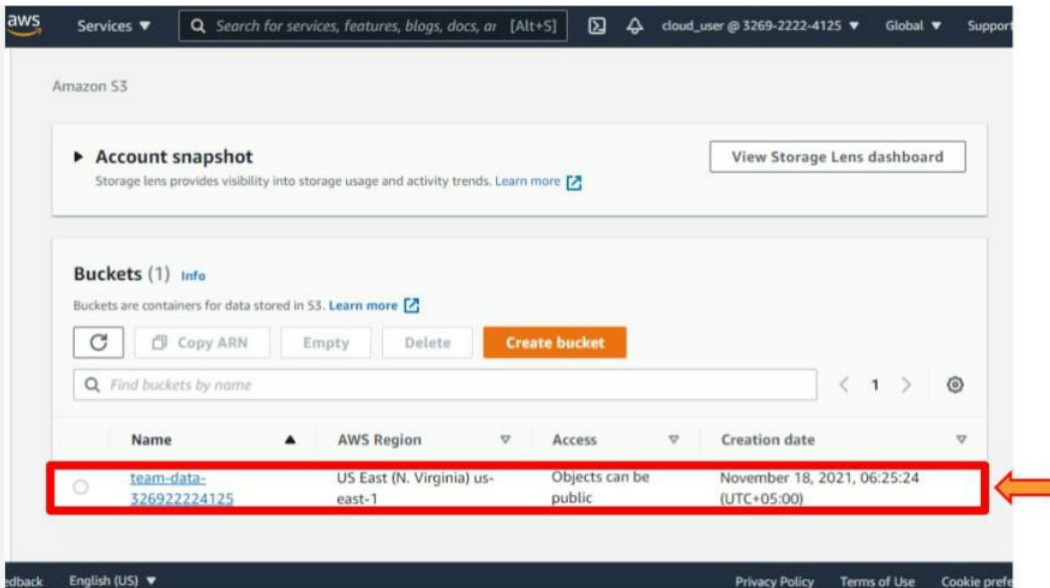
1. Log in to the “AWS Console.”
2. Click on the “Services.”



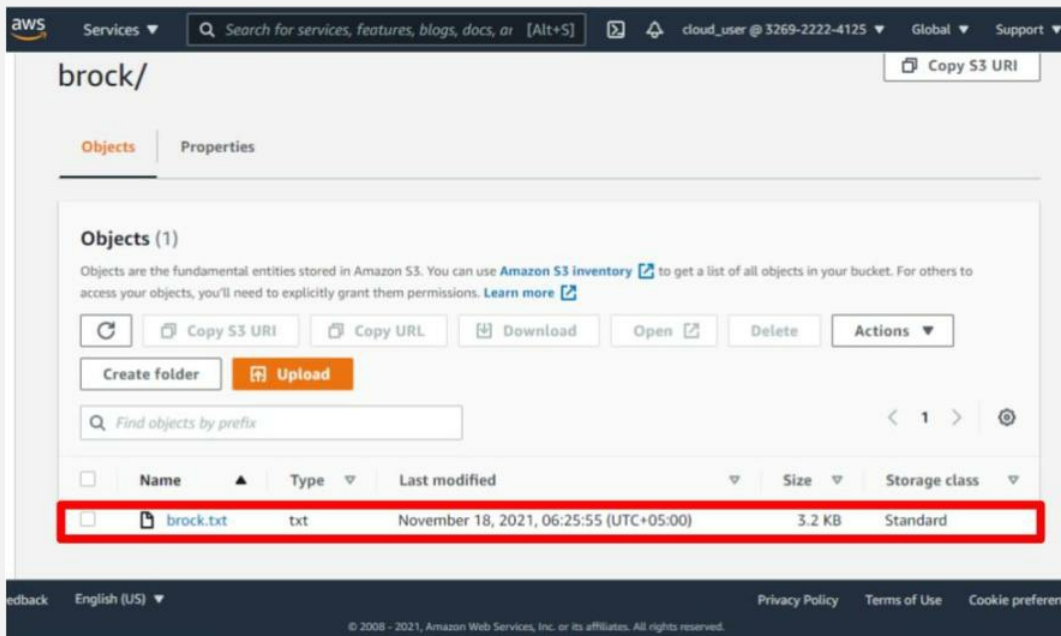
3. Select the “S3” from the “Storage.”



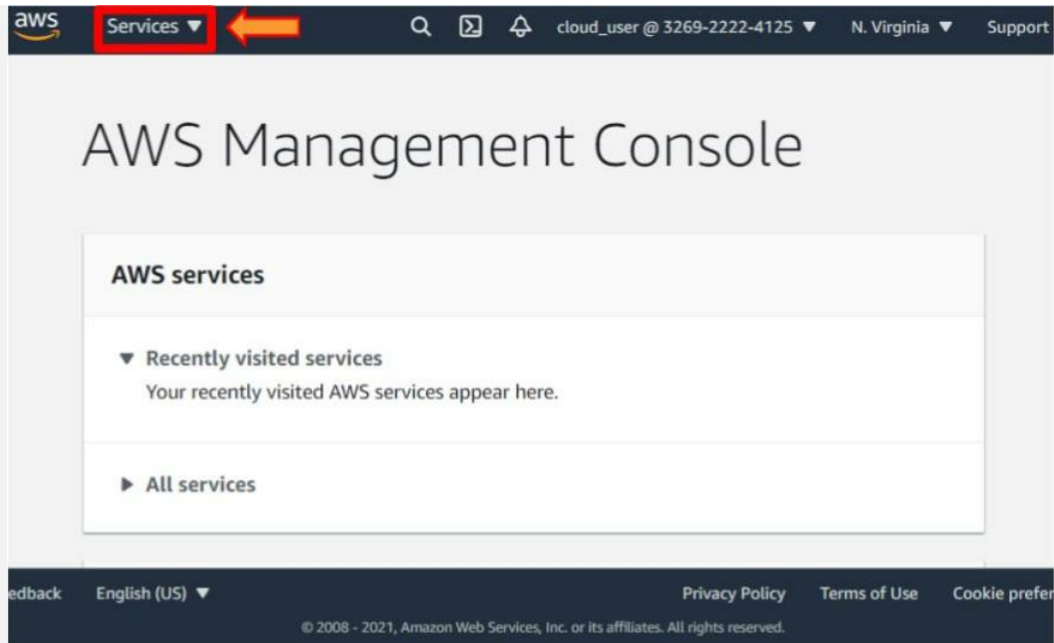
4. Click on the “Team-data” bucket.



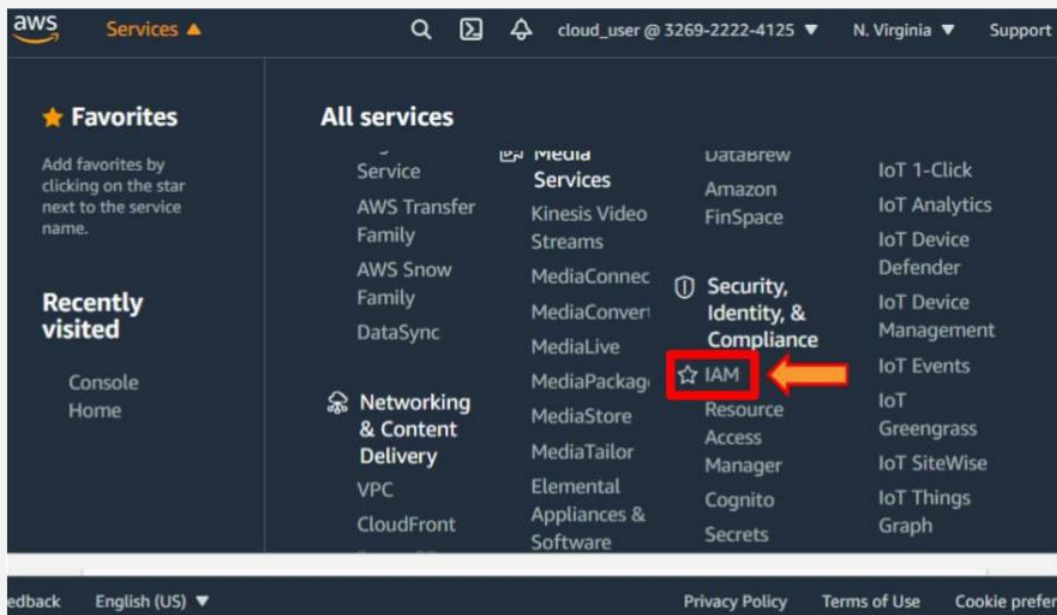
5. Click each of the “**Folders**” one at a time. Each “**Folder**” should contain a “.txt” object.



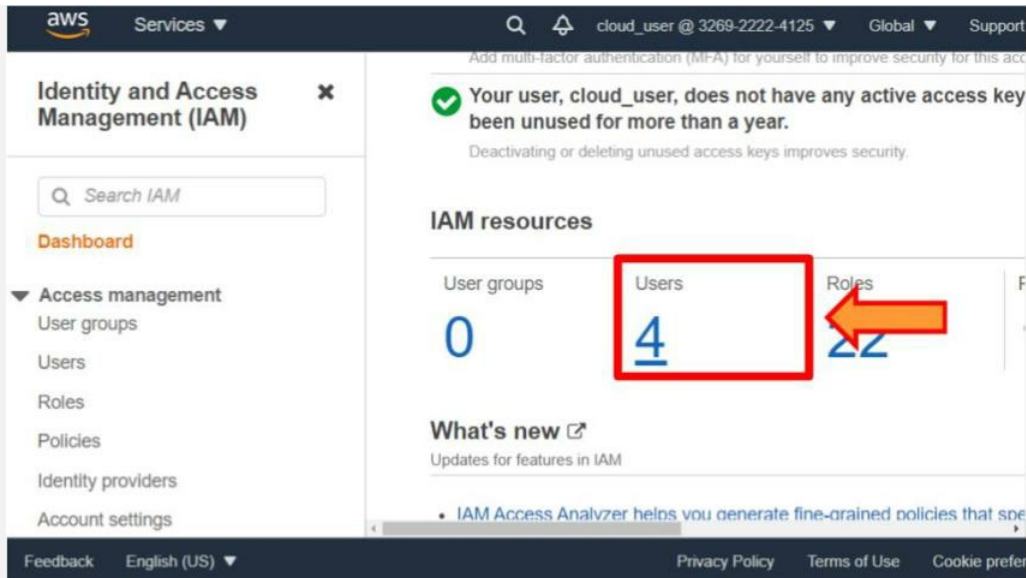
6. Click on the “**Services.**”



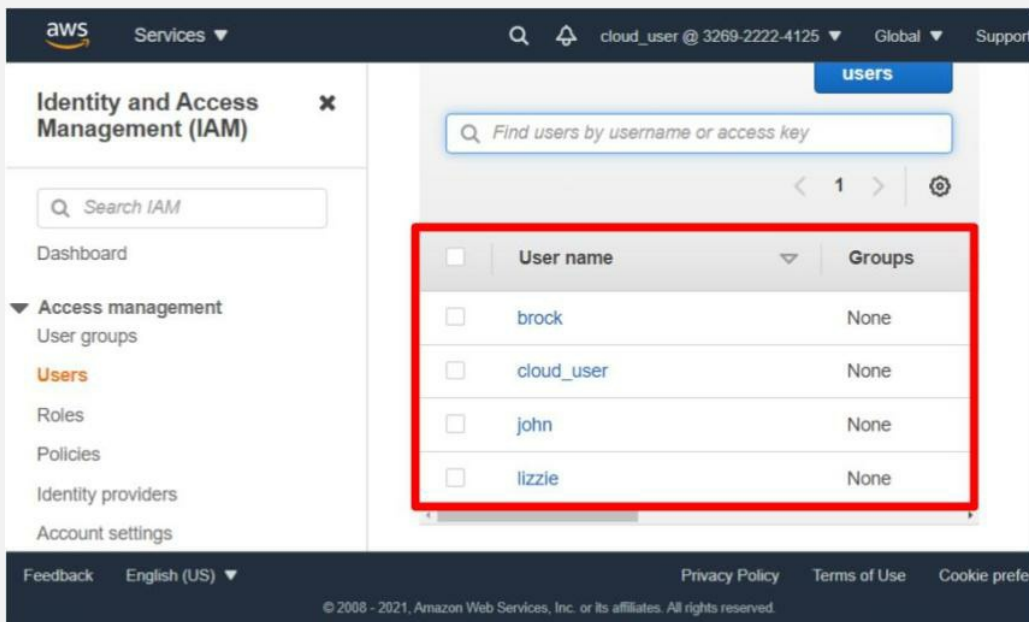
7. Select the “IAM” from the “Security, Identity, & Compliance.”



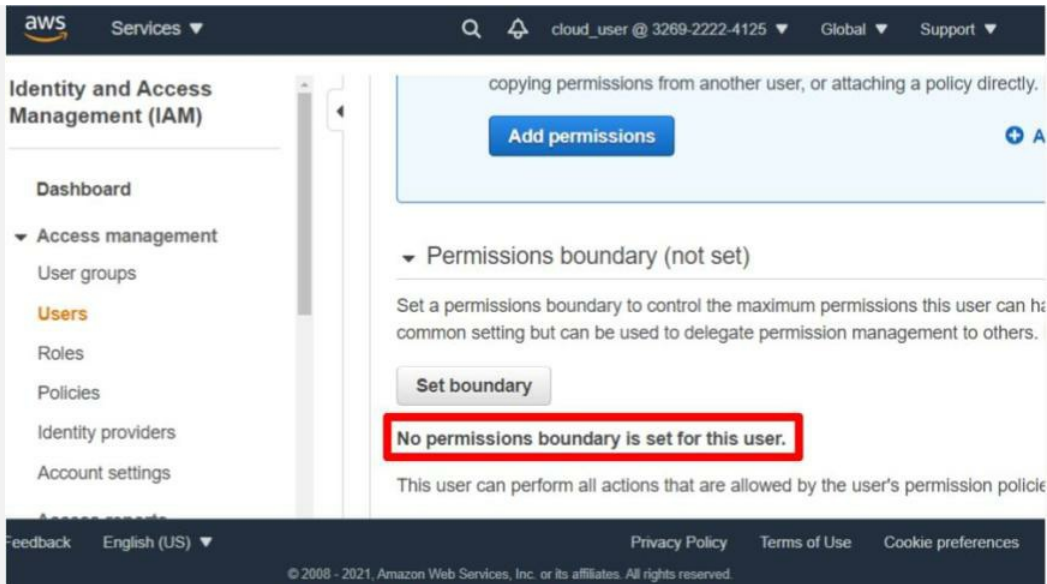
8. Click on the “Users: 4 link.”



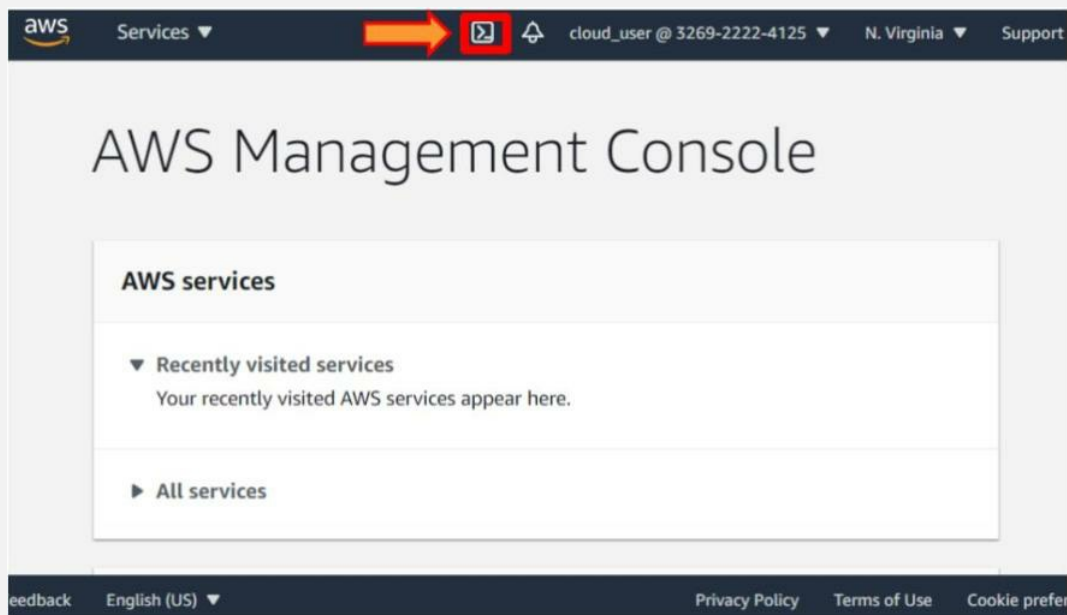
9. You should see four users: “Brock, John, Lizzie, and Cloud\_user.”



10. Select any user. Note that these users do not have any permissions and are not currently assigned to any groups.



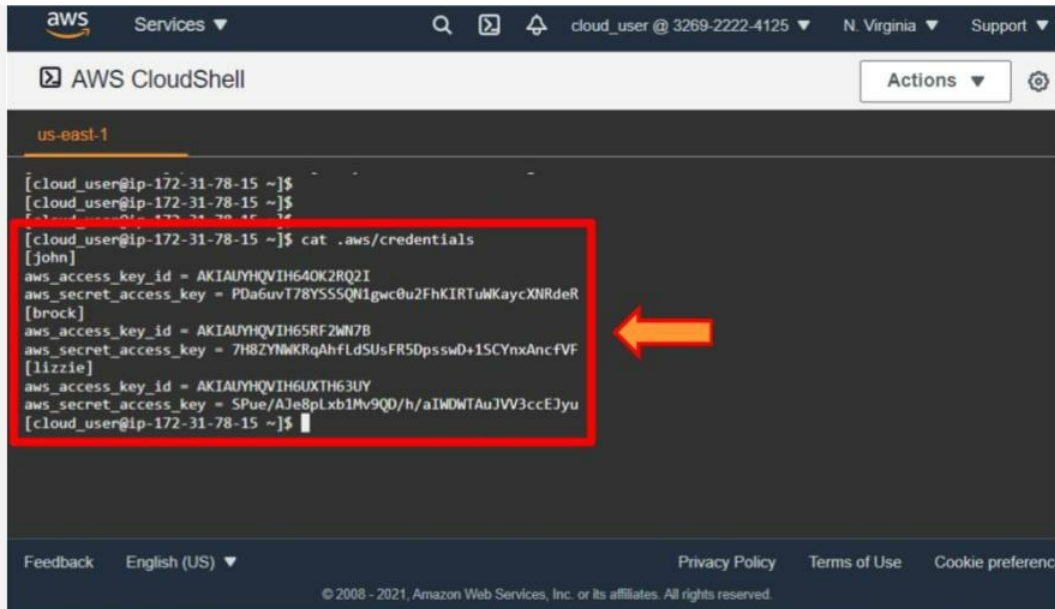
11. Go back to the AWS Management Console tab. Click on the “CloudShell” icon in the top right to open a new browser tab.



12. Close the Welcome to AWS CloudShell pop-up window and let the environment spin up.







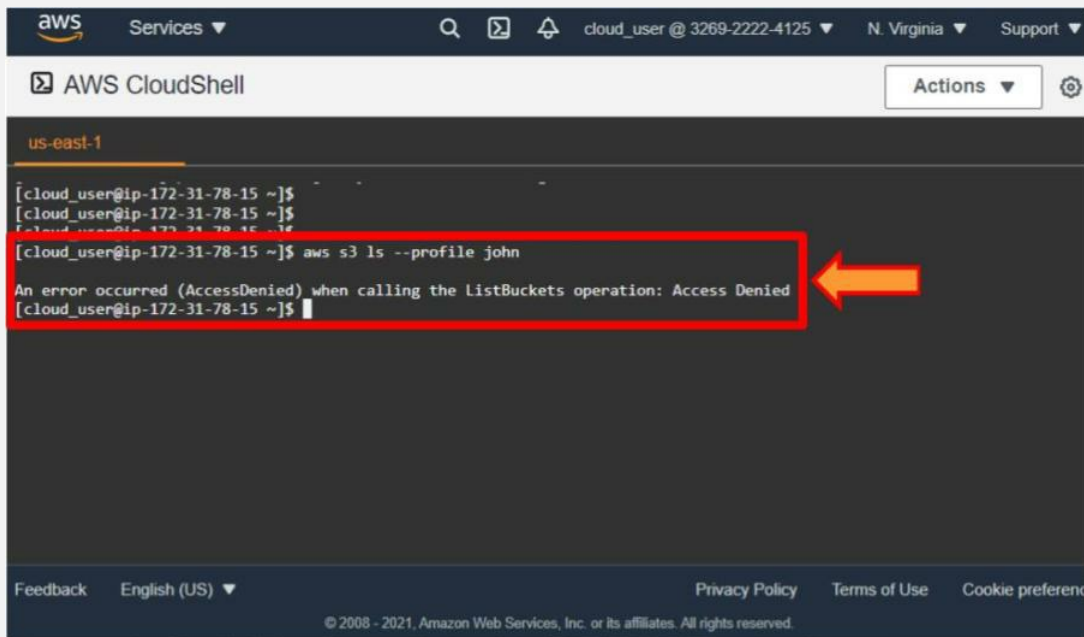
```
aws
Services
cloud_user @ 3269-2222-4125
N. Virginia
Support

AWS CloudShell

us-east-1

[cloud_user@ip-172-31-78-15 ~]$
[cloud_user@ip-172-31-78-15 ~]$
[cloud_user@ip-172-31-78-15 ~]$ cat .aws/credentials
[john]
aws_access_key_id = AKIAUYHQVTH64OK2RQ2I
aws_secret_access_key = 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
[brock]
aws_access_key_id = AKIAUYHQVTH65RF2WN7B
aws_secret_access_key = 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
[lizzie]
aws_access_key_id = AKIAUYHQVTH6UXTH63UY
aws_secret_access_key = SPue/AJe8pLxb1Mv9QD/h/aIWDWTAuJV3ccEJyu
[cloud_user@ip-172-31-78-15 ~]$
```

18. Try the following provided command “**aws s3 ls --profile john**” to view the ListBuckets in the account using john. You should receive an Access Denied error.



```
aws
Services
cloud_user @ 3269-2222-4125
N. Virginia
Support

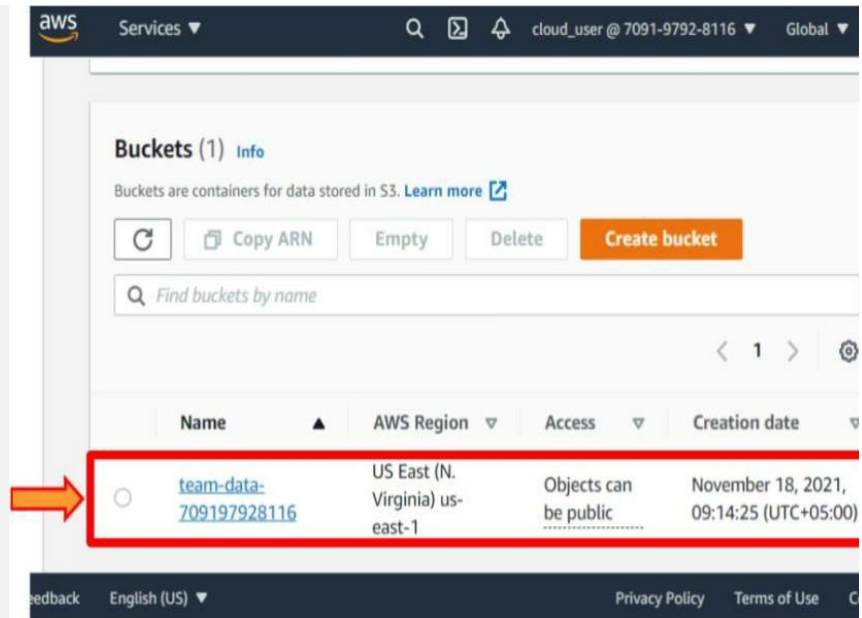
AWS CloudShell

us-east-1

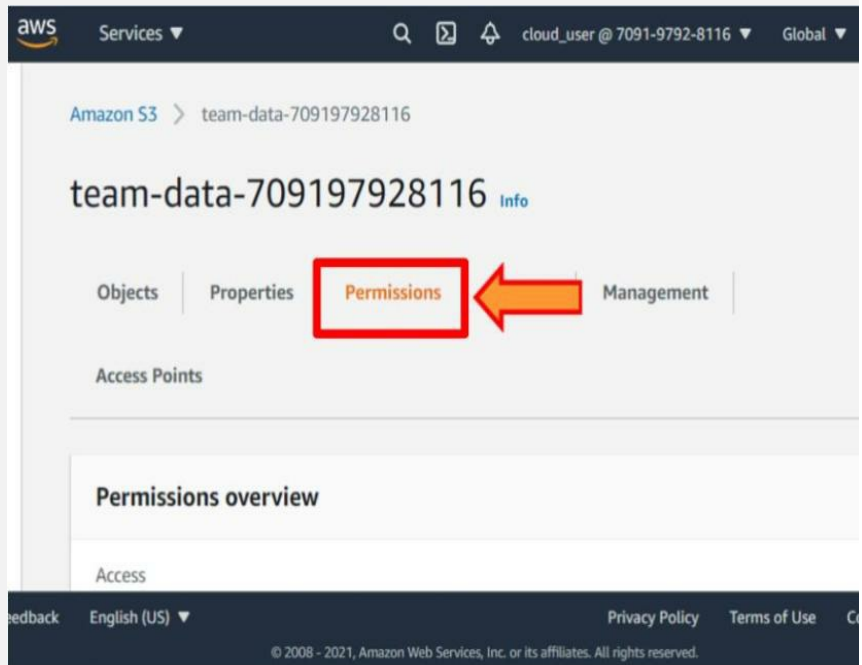
[cloud_user@ip-172-31-78-15 ~]$
[cloud_user@ip-172-31-78-15 ~]$
[cloud_user@ip-172-31-78-15 ~]$ aws s3 ls --profile john
An error occurred (AccessDenied) when calling the ListBuckets operation: Access Denied
[cloud_user@ip-172-31-78-15 ~]$
```

## Step 2: Provide Appropriate Public Access

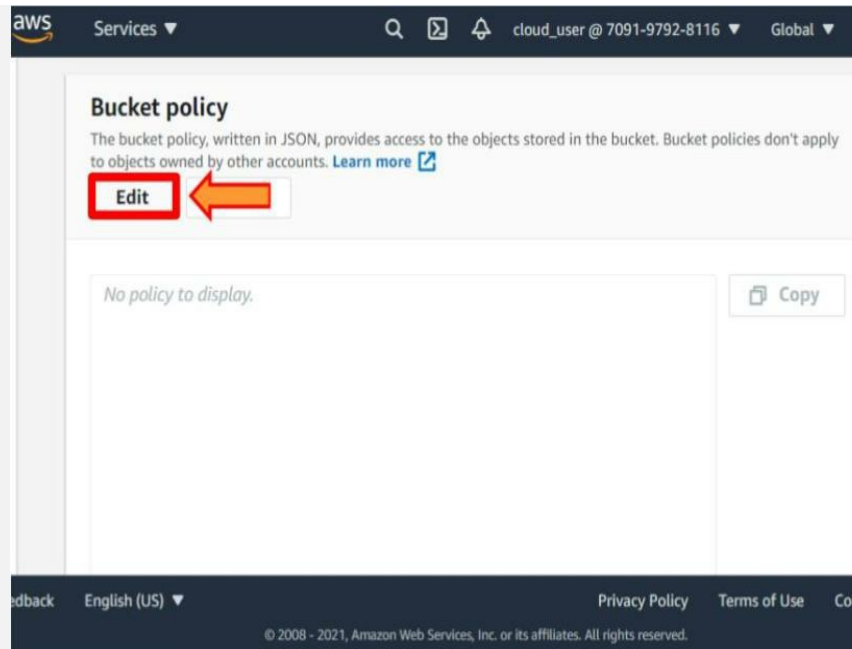
1. Go back to the “**S3**” dashboard. Click on the “**Team-data**” bucket



2. Click on the “Permissions” tab.



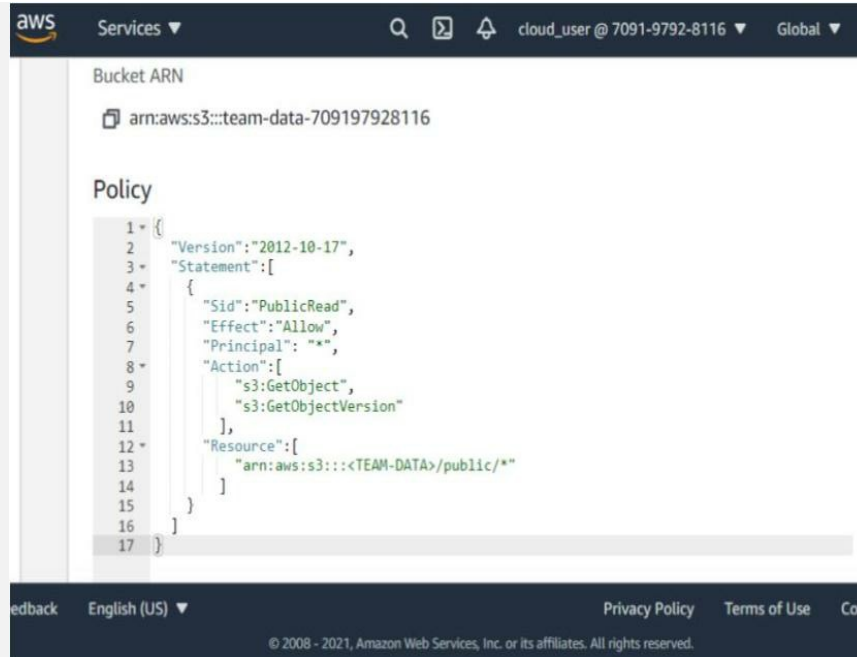
3. Scroll down to the Bucket policy section. Then click on the “Edit



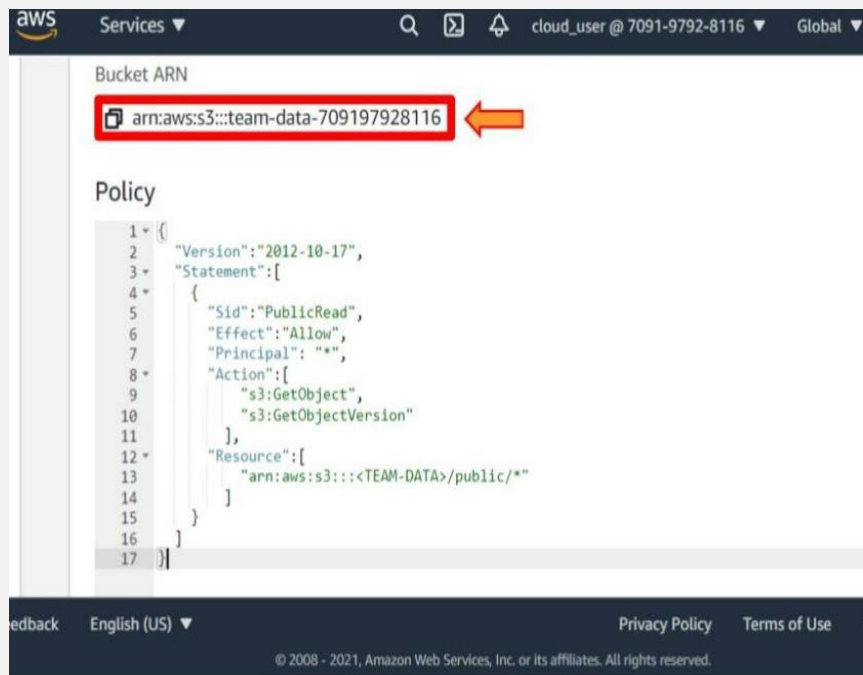
4. Copy the policy provided in the following link: [https://github.com/linuxacademy/Content-AWS-Certified-Data-Speciality/blob/master/Lab Assets/advanced\\_s3\\_security\\_config](https://github.com/linuxacademy/Content-AWS-Certified-Data-Speciality/blob/master/Lab%20Assets/advanced_s3_security_config)

```
17 lines (17 sloc) | 296 Bytes
Raw Blame
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "PublicRead",
6       "Effect": "Allow",
7       "Principal": "*",
8       "Action": [
9         "s3:GetObject",
10        "s3:GetObjectVersion"
11      ],
12      "Resource": [
13        "arn:aws:s3:::TEAM-DATA>/public/*"
14      ]
15    }
16  ]
17 }
```

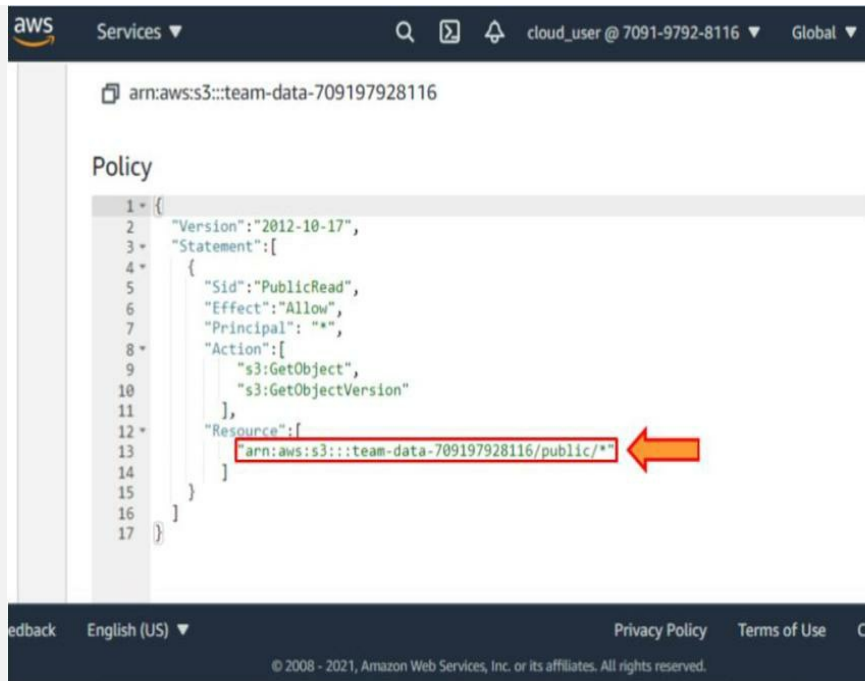
5. Paste it into the “Policy” editor.



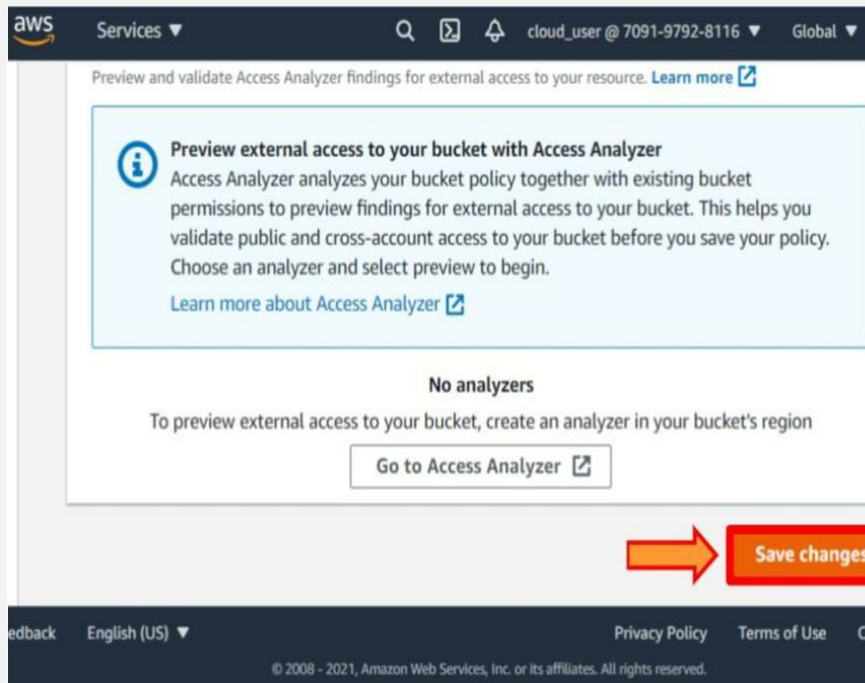
6. Above the “**Policy Editor**,” copy the “**team-data-<ACCOUNT\_ID>**” bucket ARN.



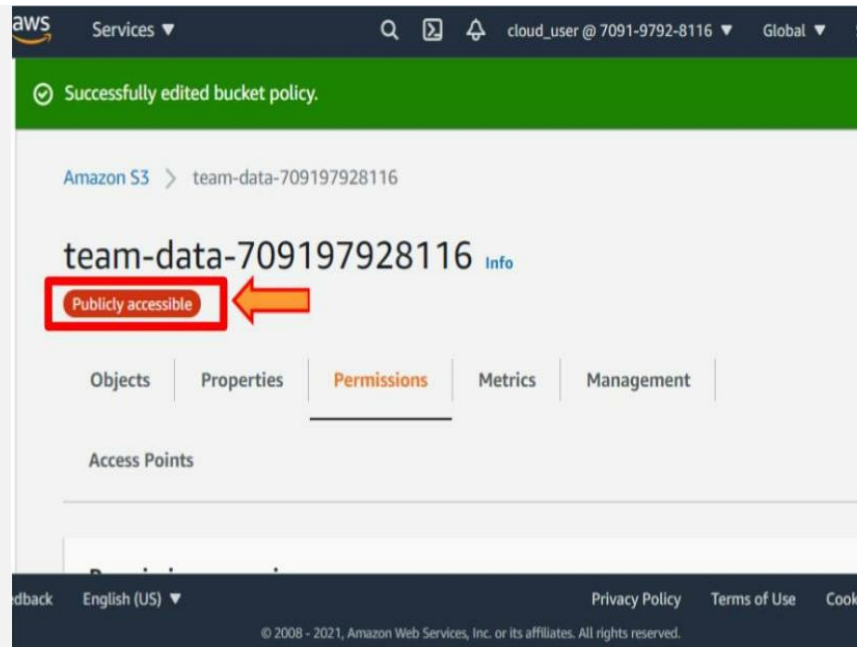
7. Then paste it into the “**<TEAM-DATA>**” placeholder.



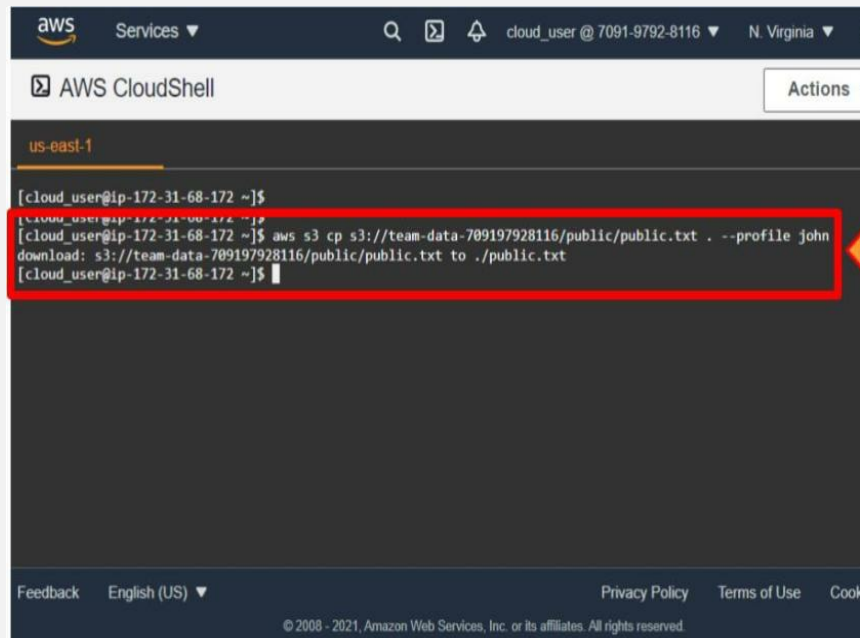
8. Click on the “Save Changes” button.



9. The S3 bucket now displays a “Publicly Accessible” warning.



10. Go back to “CloudShell” and copy the “public.txt” object to the to replace <BUCKET\_NMAE> with your “Bucket Name.”
11. The following command is used to co  
`s3://<BUCKET_NAME>/public/public.txt . --profile john.`



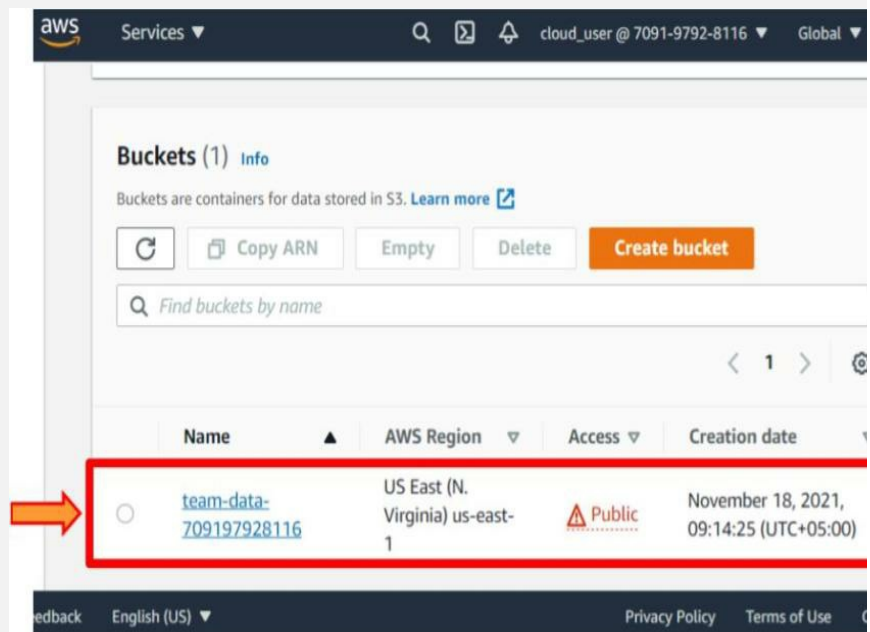
12. Run the “Cat” command on the “public.txt” object. You should  
 Response, indicating you have successfully provided public acces
13. The following command is used to display the output on the pro

The screenshot shows the AWS CloudShell interface. The terminal window displays the following commands and output:

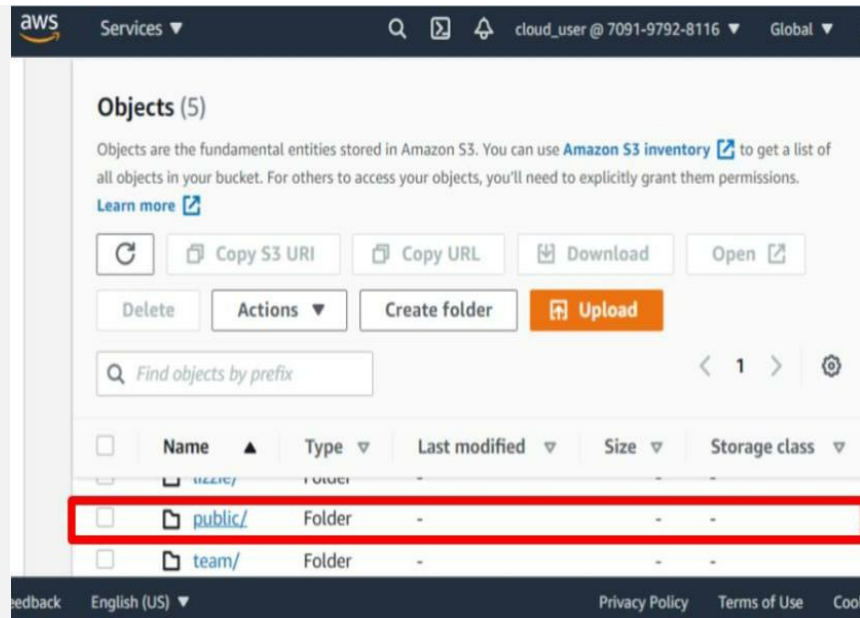
```
us-east-1  
[cloud_user@ip-172-31-68-172 ~]$  
[cloud_user@ip-172-31-68-172 ~]$  
[cloud_user@ip-172-31-68-172 ~]$ cat public.txt  
Hello everyone! [cloud_user@ip-172-31-68-172 ~]$
```

A red box highlights the command and its output, with an orange arrow pointing to the output line.

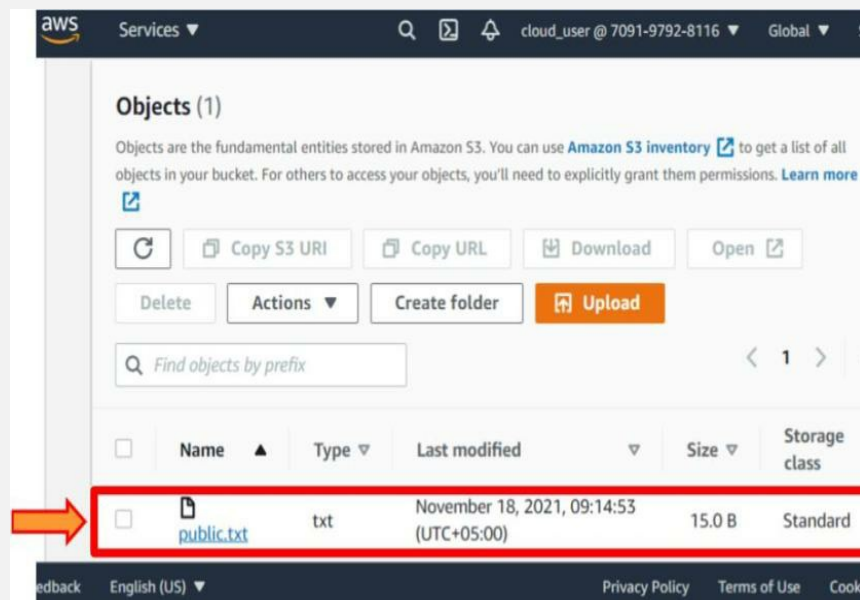
14. Go back to the S3 dashboard. Click on the “**Team-data**” bucket.



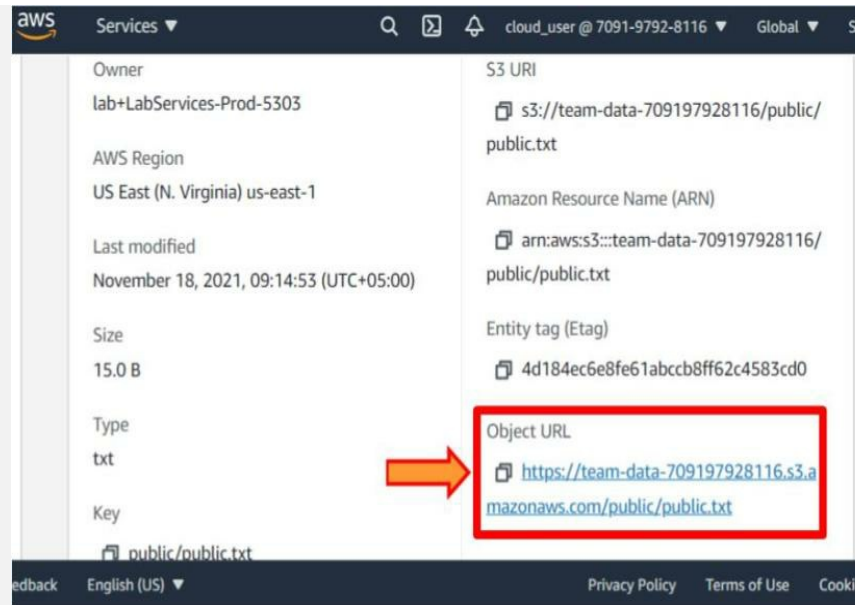
15. Click on the “**Public/ Folder.**”



16. Click on the “public.txt” object.



17. Open the “Object URL” in a separate browser.

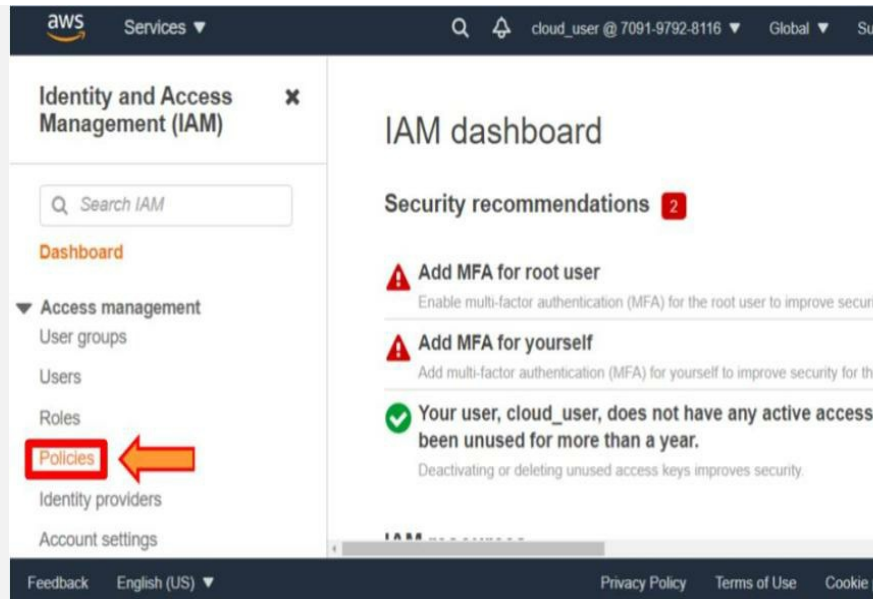


18. You should see a “**Hello, everyone!**” Text.

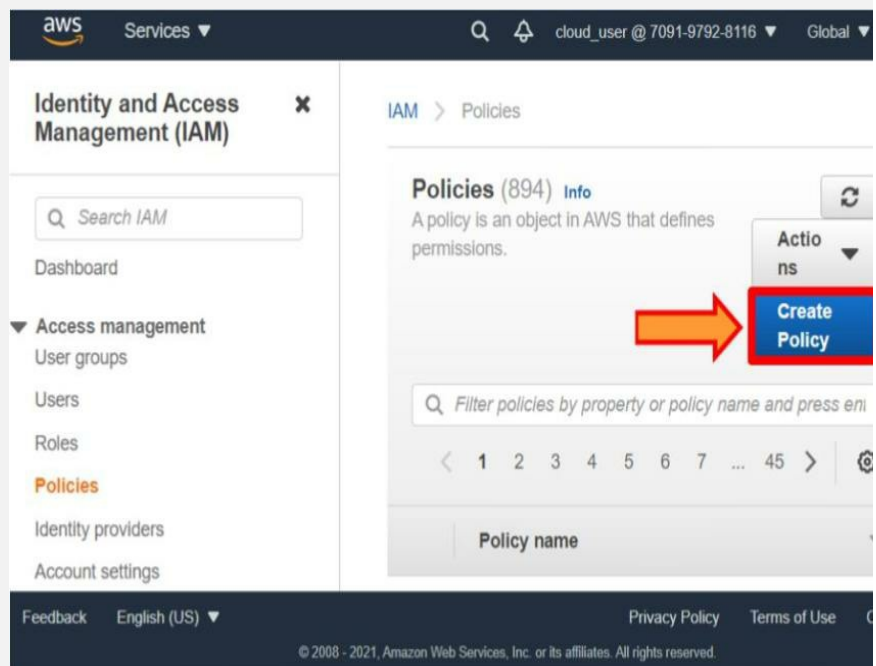


### Step 3: Provide Appropriate Team and User Access

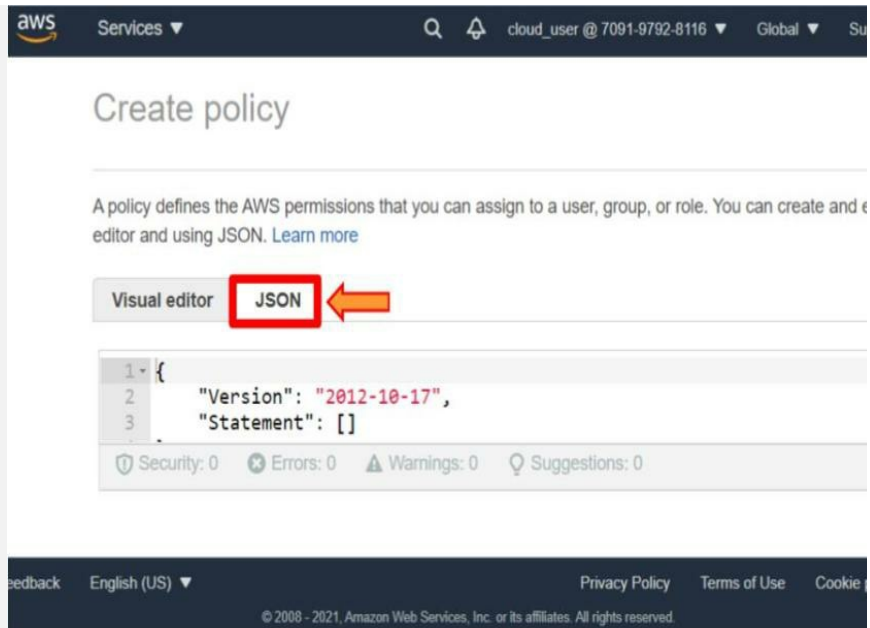
1. Create User and Group Policies
2. Go back to the “IAM” dashboard. Then click on the “Policies” menu.



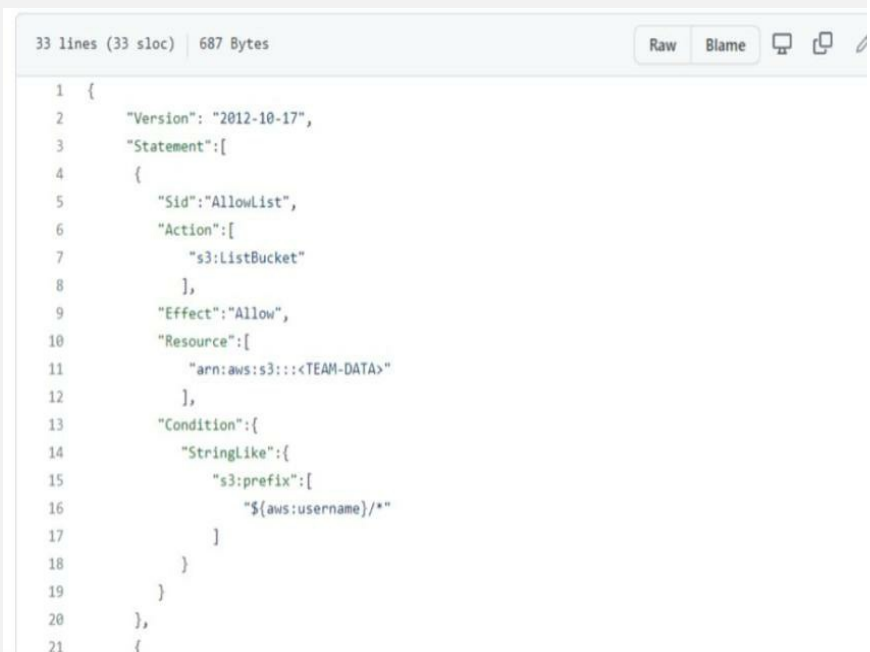
3. Click on the “Create Policy” button.



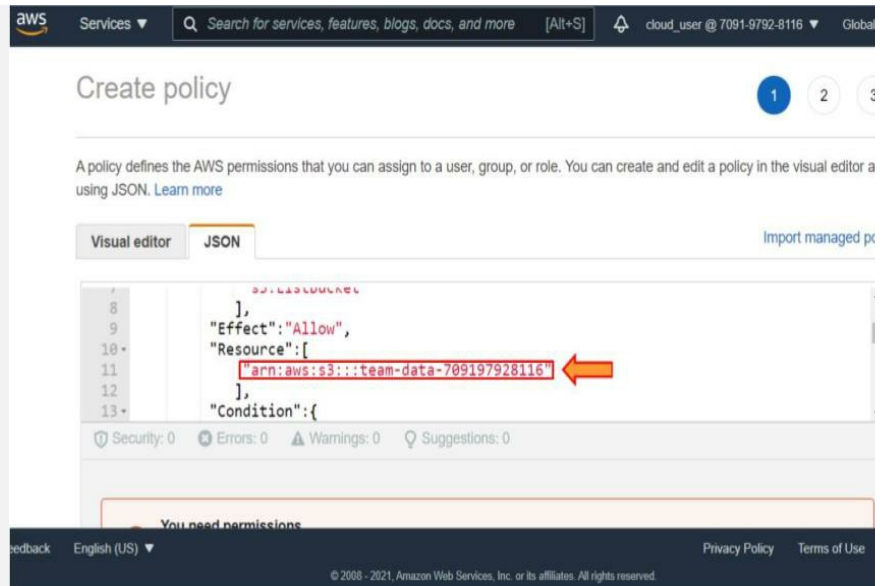
4. Click on the “JSON” tab.



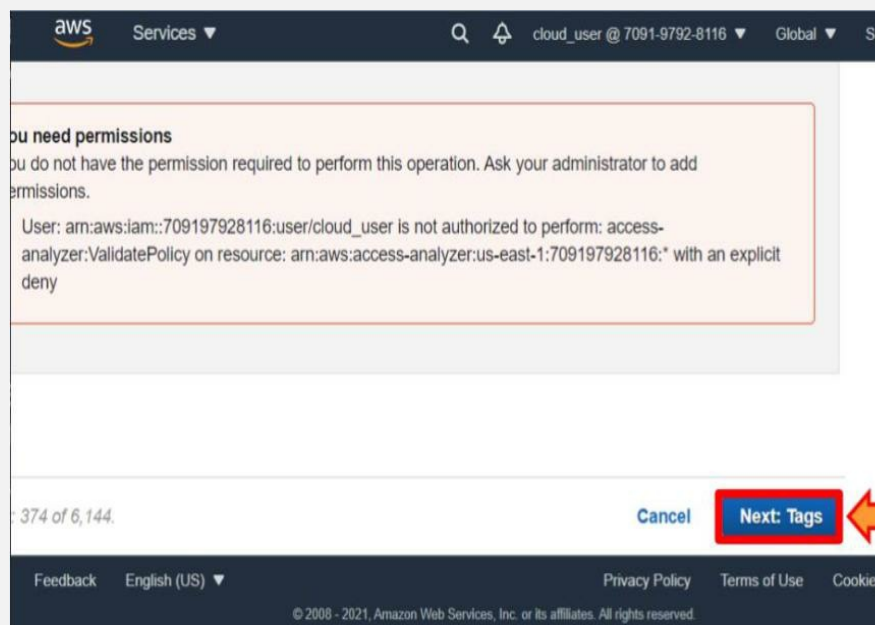
5. Copy the user policy provided in the folder [https://github.com/linuxacademy/Content-AWS-Certified-Data-Speciality/blob/master/Lab\\_Assets/advanced\\_s3\\_security\\_config](https://github.com/linuxacademy/Content-AWS-Certified-Data-Speciality/blob/master/Lab_Assets/advanced_s3_security_config). Then paste it into the JSON policy editor.



6. Copy your “**S3 Bucket**” name and paste it into any “<TEAM-D.” JSON policy editor.



7. Click on the “Next: Tags” button.



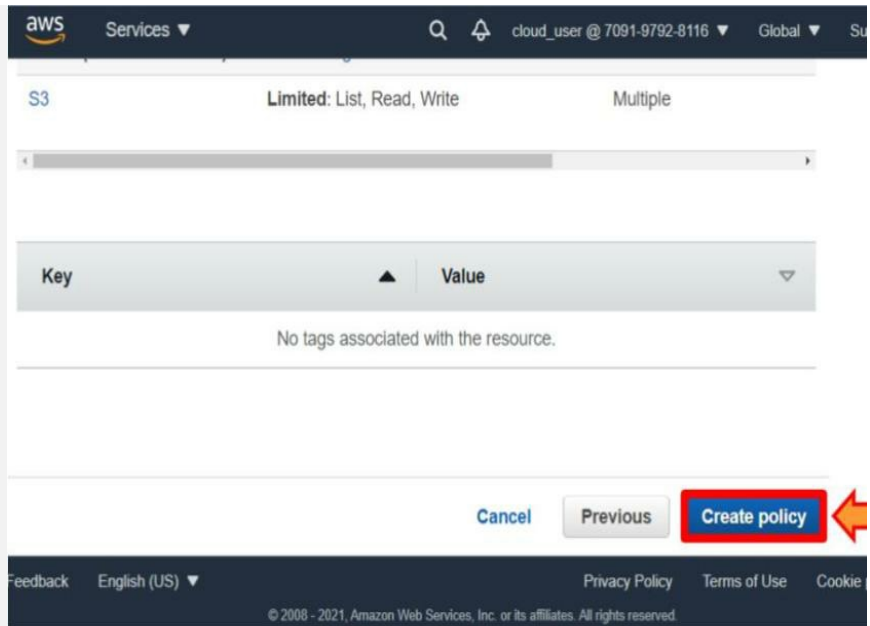
8. Click on the “Next: Review” button.

This screenshot shows the 'Add permissions' step in the AWS IAM console. The page title is 'Add permissions'. Below the title, there is a section for 'Optional) Tags' with a text input field and a note: 'pairs that you can add to AWS resources to help identify, organize, or search for resources.' Below this is a section for 'Permissions' with a text input field and a note: 'Select the permissions you want to add to the role. You can select multiple permissions. Select permissions that are associated with the resource.' Below the 'Permissions' section is a button labeled '50 more tags'. At the bottom of the form are three buttons: 'Cancel', 'Previous', and 'Next: Review'. The 'Next: Review' button is highlighted with a red box and an orange arrow pointing to it from the right. The AWS header is visible at the top, showing the user 'cloud\_user @ 7091-9792-8116' and the region 'Global'.

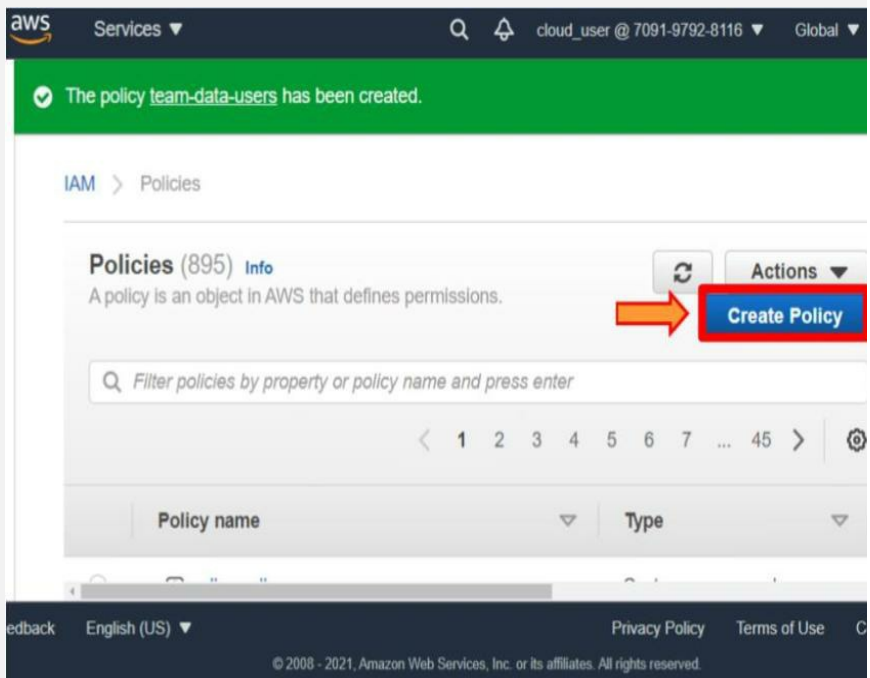
9. In the Name field, enter the **“team-data-users.”**

This screenshot shows the 'Create policy' step in the AWS IAM console. The page title is 'Create policy'. Below the title is a section for 'Review policy'. The 'Name\*' field is highlighted with a red box and contains the text 'team-data-users'. An orange arrow points to the 'Name\*' field from the right. Below the 'Name\*' field is a note: 'Use alphanumeric and '+=, @, \_' characters. Maximum 128 characters.' Below the 'Name\*' field is a 'Description' text input field. Below the 'Description' field is a note: 'Maximum 1000 characters. Use alphanumeric and '+=, @, \_' characters.' At the bottom of the form are three buttons: 'Cancel', 'Previous', and 'Next: Review'. The 'Next: Review' button is highlighted with a red box and an orange arrow pointing to it from the right. The AWS header is visible at the top, showing the user 'cloud\_user @ 7091-9792-8116' and the region 'Global'.

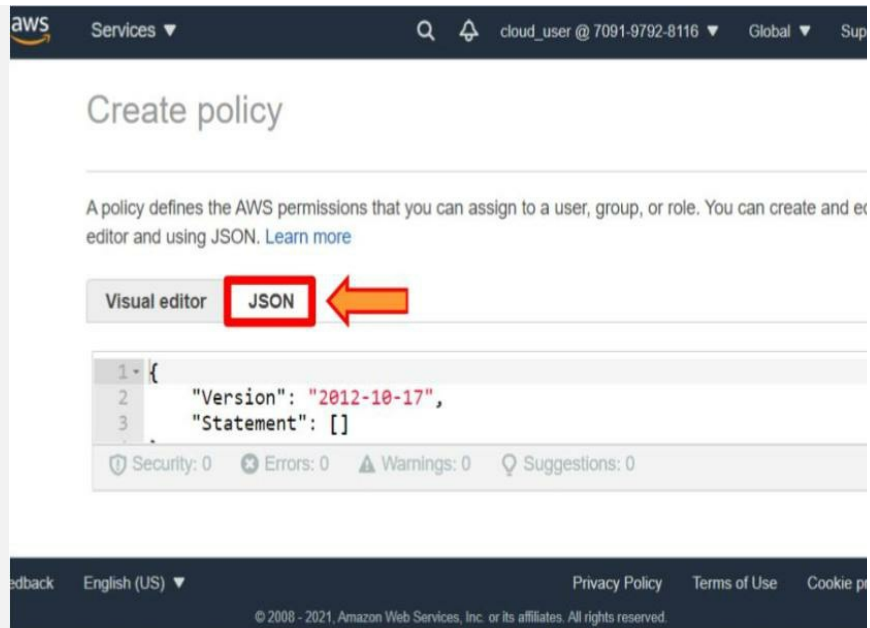
10. Click on the **“Create Policy”** button.



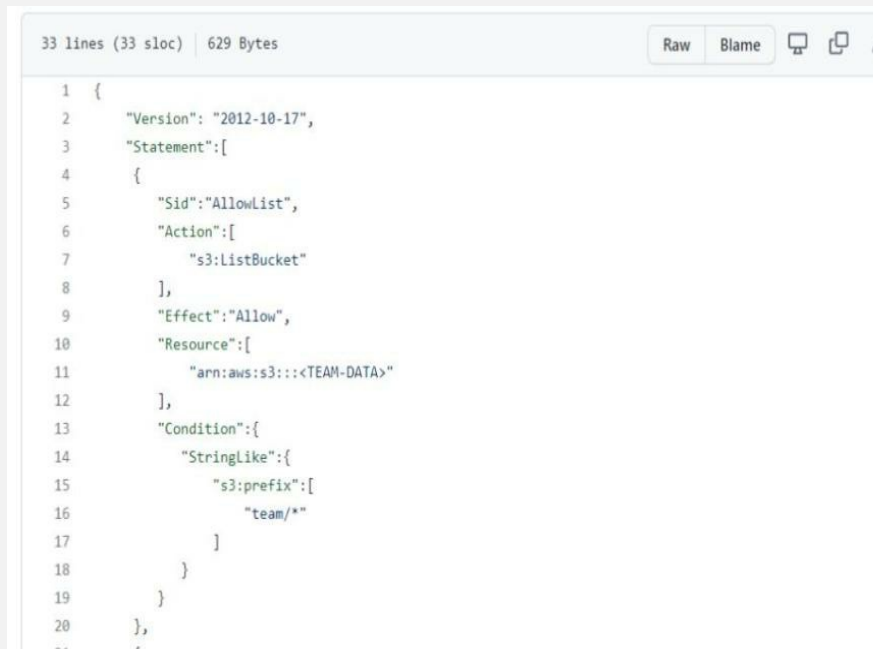
11. Click on the “Create Policy” to create another policy.



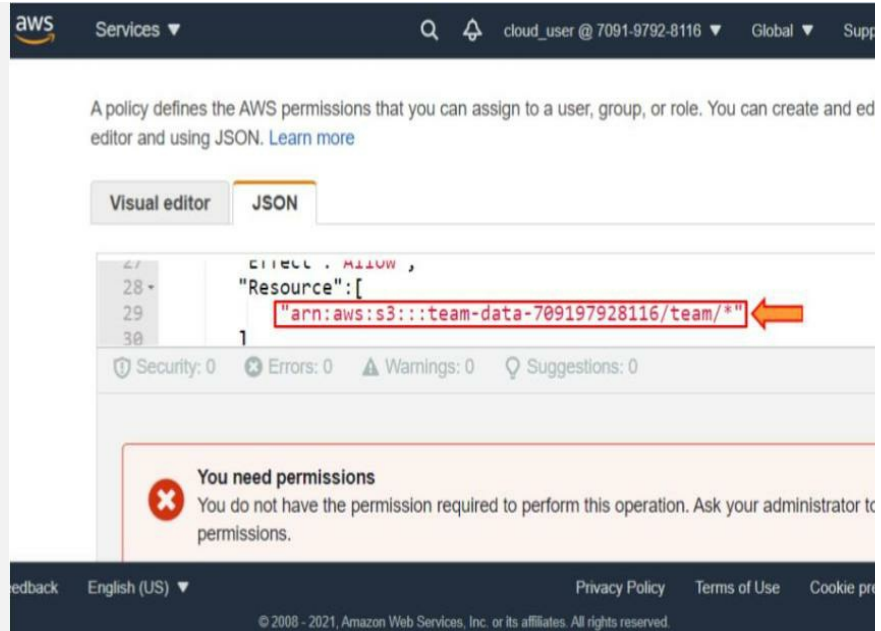
12. Click on the “JSON” tab.



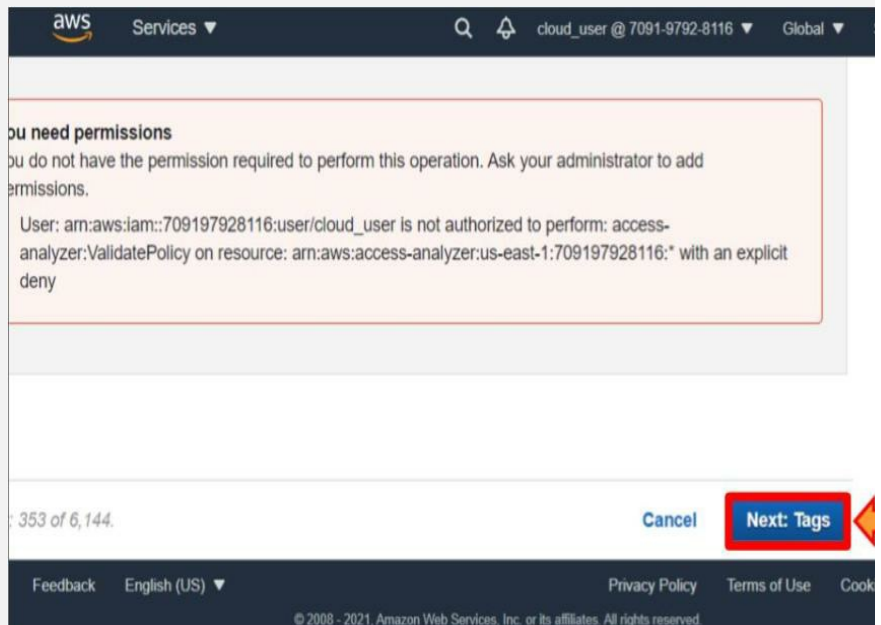
13. Copy the group policy provided in the [https://github.com/linuxacademy/Content-AWS-Certified-Data-Speciality/blob/master/Lab Assets/advanced\\_s3\\_security config](https://github.com/linuxacademy/Content-AWS-Certified-Data-Speciality/blob/master/Lab%20Assets/advanced_s3_security_config) and paste it into the JSON policy editor.



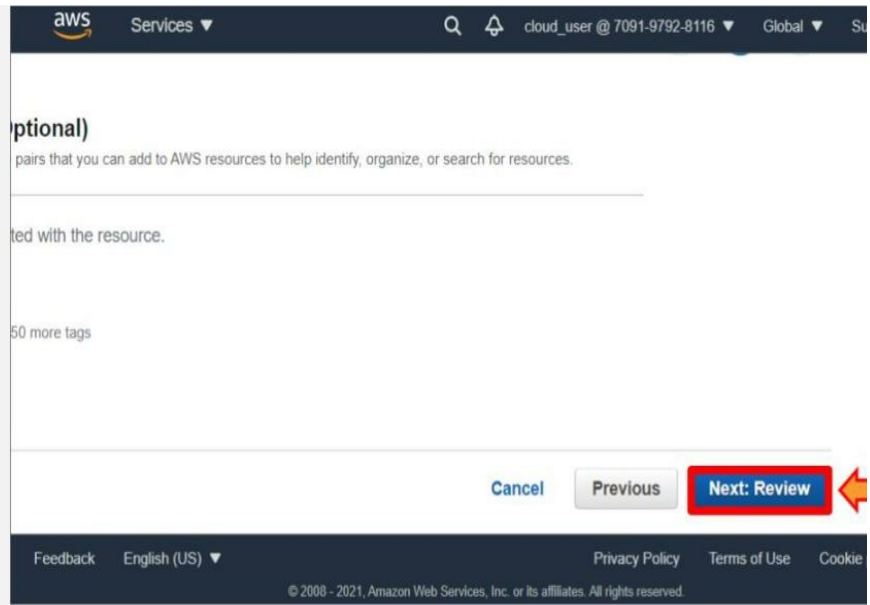
14. Copy your **"S3 Bucket"** name and paste it into any **"<TEAM-DATA>"** in the JSON policy editor.



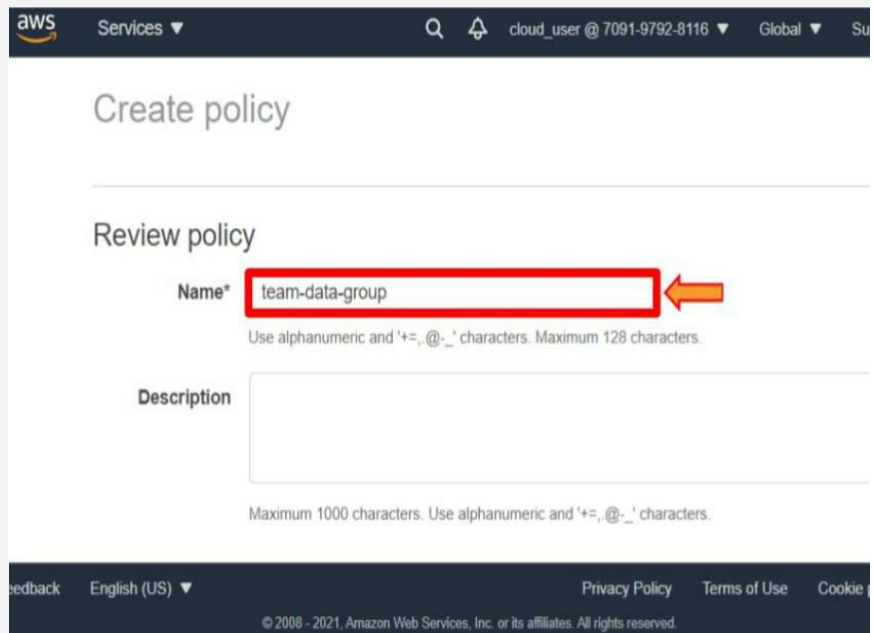
15. Click on the “Next: Tags” button.



16. Click on the “Next: Review” button.



17. In the Name field, enter the “**team-data-group**.”



18. Click on the “**Create Policy**” button.

aws Services 🔻 🔍 🔔 cloud\_user @ 7091-9792-8116 🔻 Global 🔻 S

Allow (1 of 302 services) Show remaining 301

S3 Limited: List, Read, Write Multiple

Key ▲ Value ▼

No tags associated with the resource.

Cancel Previous **Create policy** ➡

Feedback English (US) 🔻 Privacy Policy Terms of Use Cookie

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

19. Hence successfully create the users' groups.

aws Services 🔻 🔍 🔔 cloud\_user @ 7091-9792-8116 🔻 Global 🔻

✔ The policy team-data-group has been created.

IAM > Policies

**Policies (896)** Info

A policy is an object in AWS that defines permissions.

🔄 Actions ▼

**Create Policy**

🔍 Filter policies by property or policy name and press enter

< 1 2 3 4 5 6 7 ... 45 > ⚙️

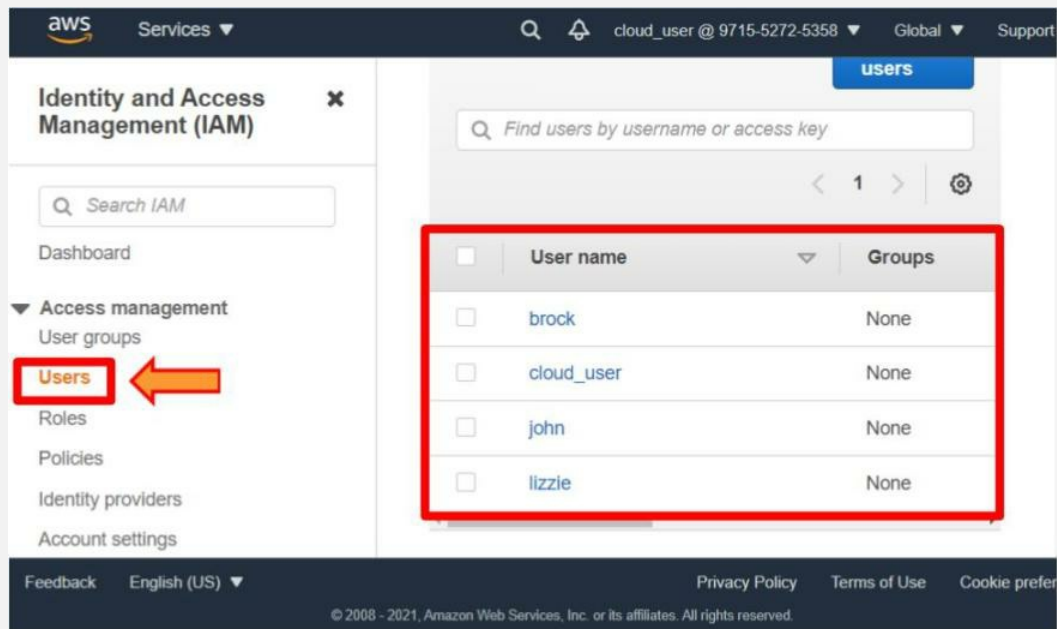
| Policy name                            | Type             |
|----------------------------------------|------------------|
| <input type="radio"/> <b>allow_all</b> | Customer managed |

Feedback English (US) 🔻 Privacy Policy Terms of Use Cookie

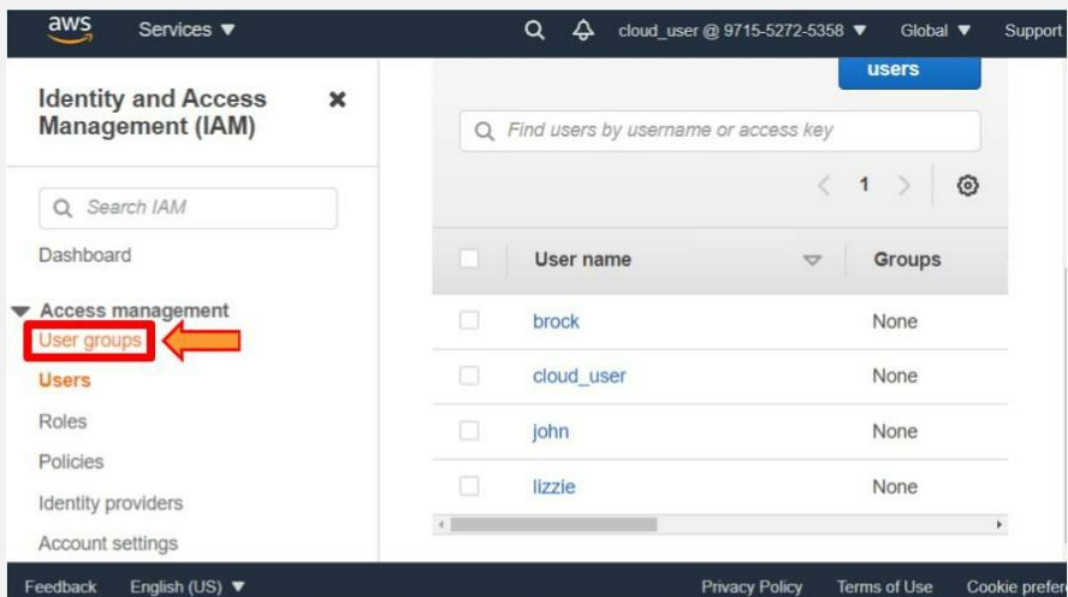
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

## Step 4: Apply the User and Group Policies

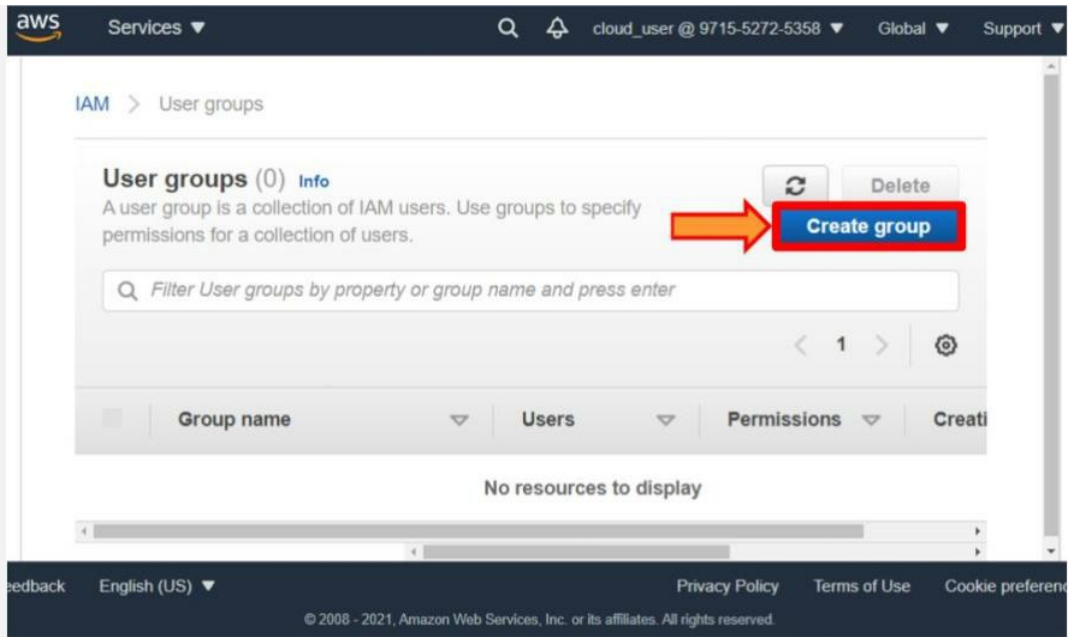
1. Click on the “Users” from the left-hand side menu to view users



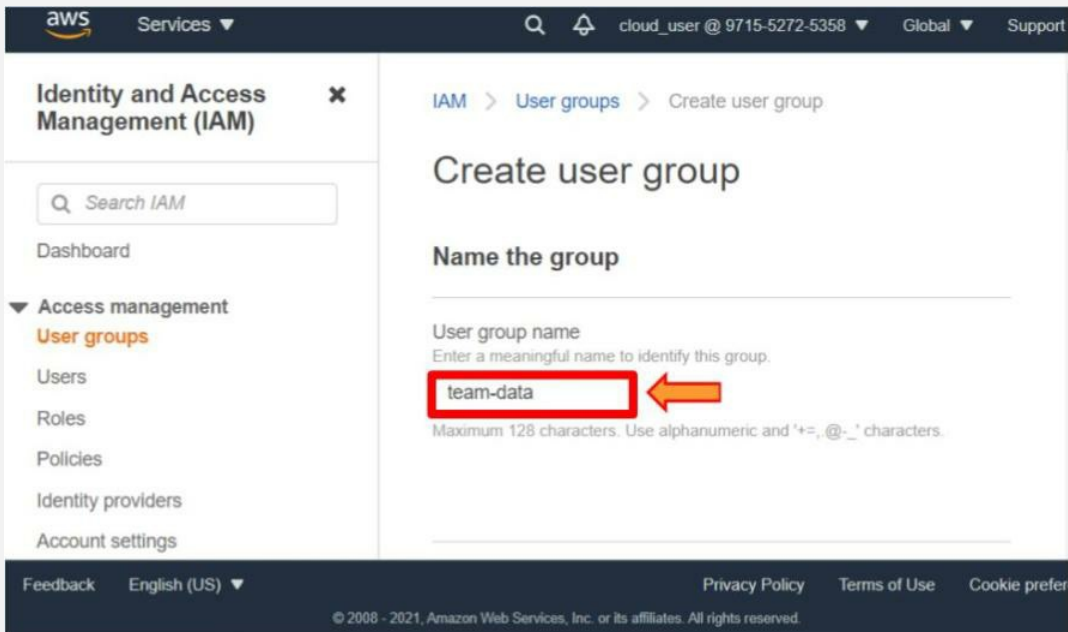
2. Click on the “Users Groups” from the left-hand side menu.



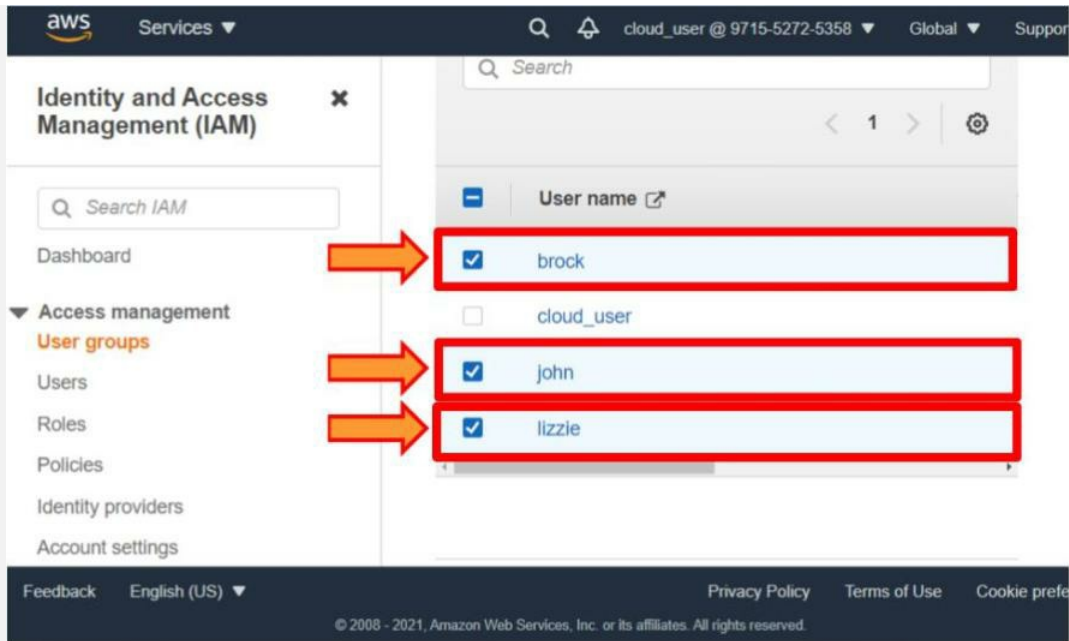
3. Click on the “Create Group” button.



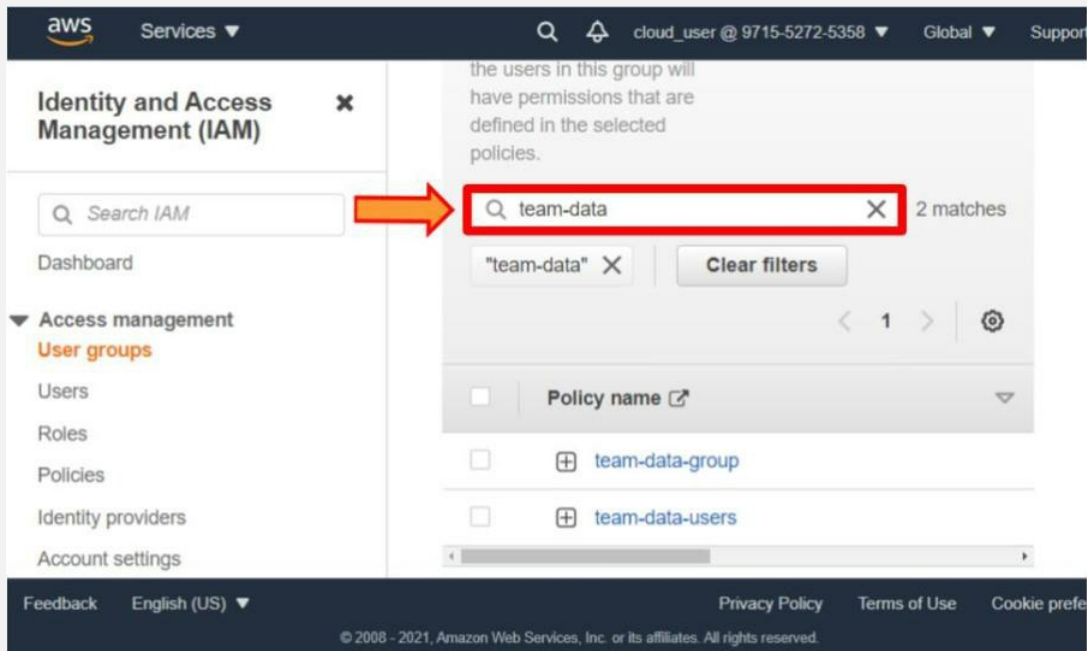
4. In the Group Name field, enter the “**team-data**.”



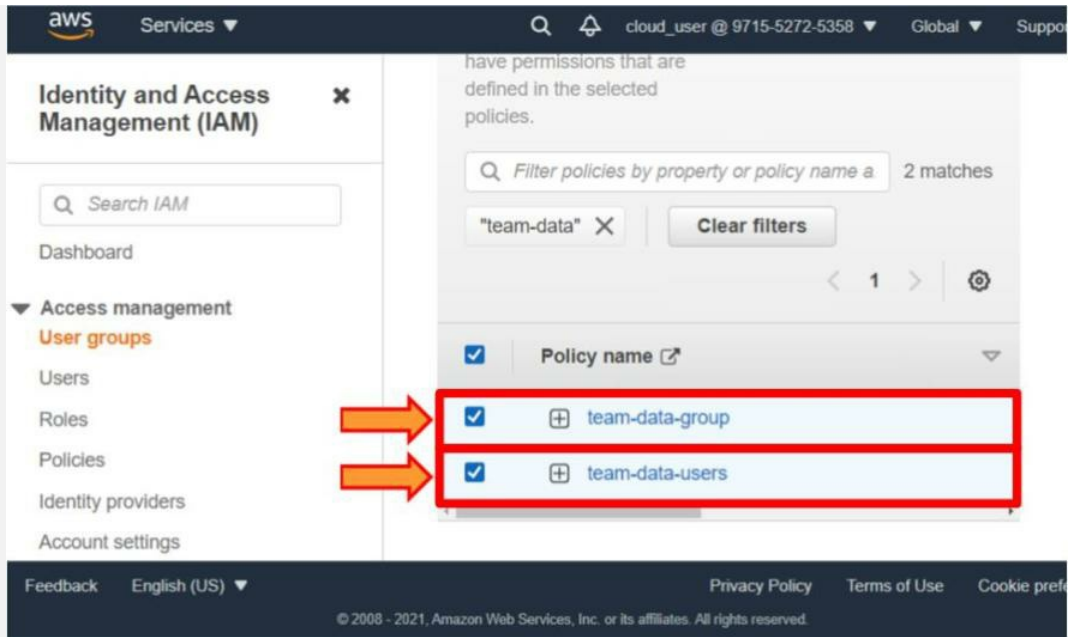
5. Scroll down. Select the “**Brock, John, and Lizzie**” users.



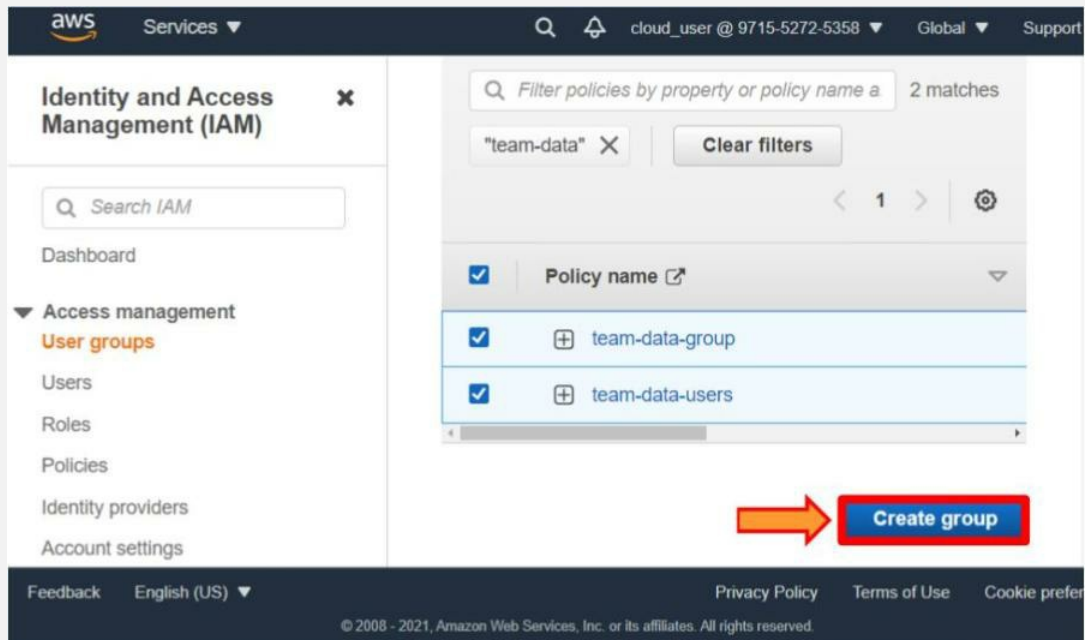
6. Scroll down. In the Policy Type field, search the “team-data.”



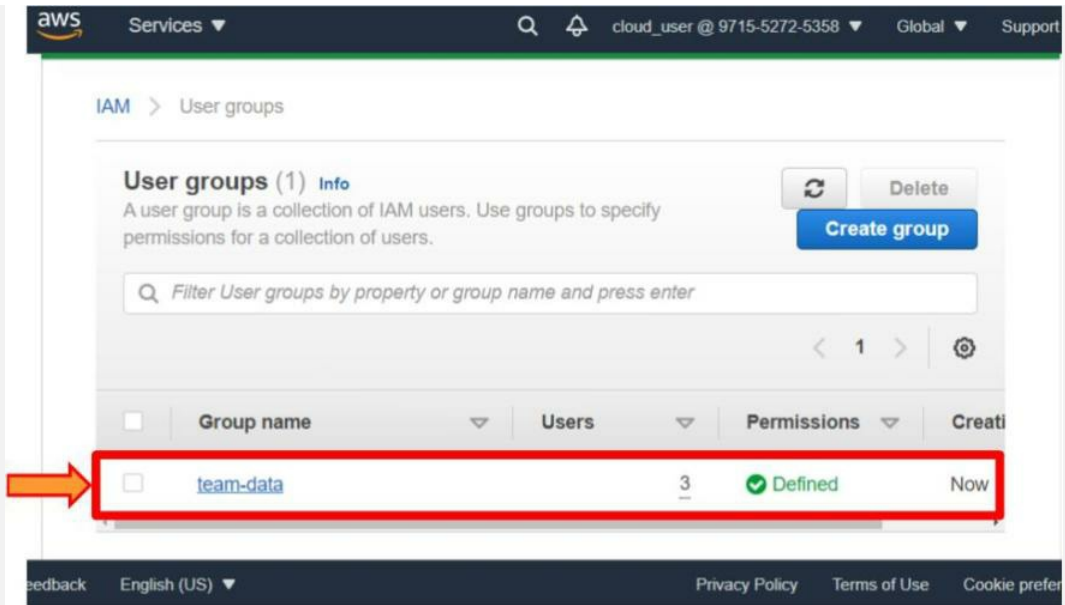
7. Select the two created policies, “team-data-group,” and “team data-users.”



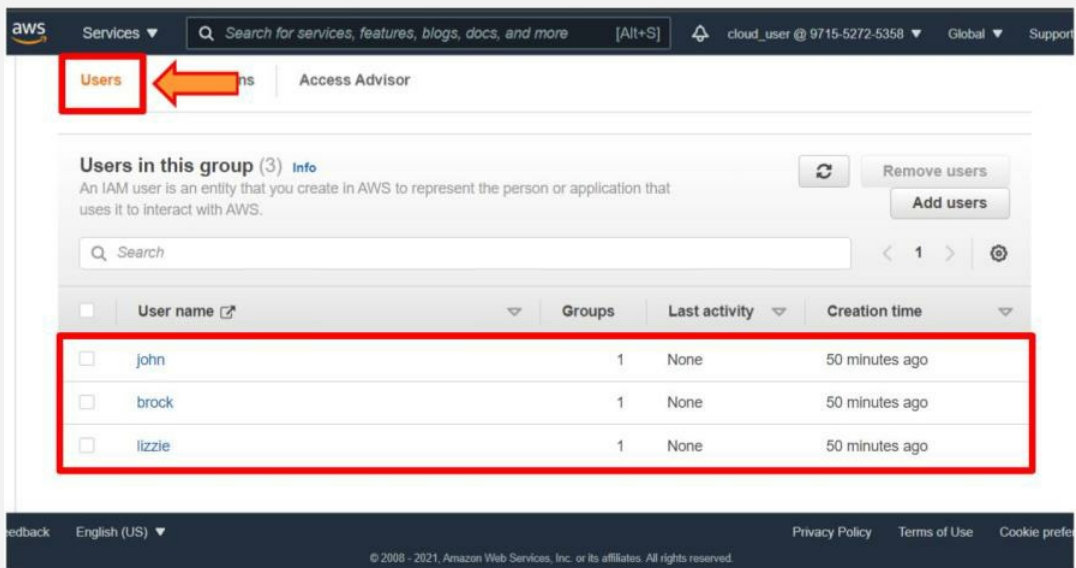
8. Click on the “Create Group” button.



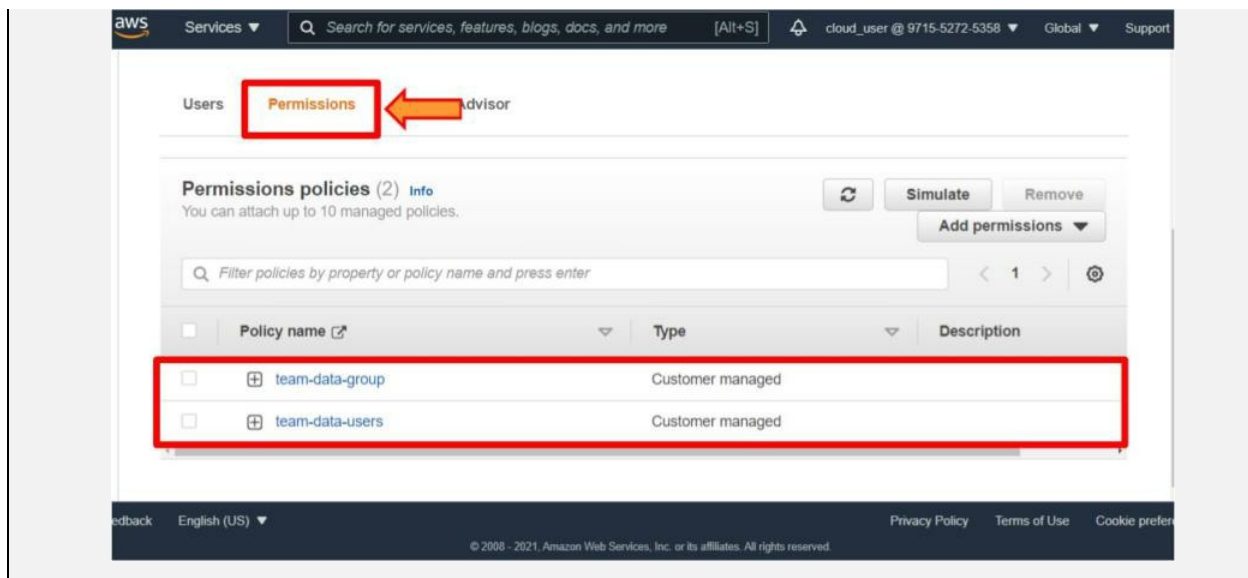
9. Click on the “team-data” group.



10. Click on the “Users” tab to confirm that users are added.

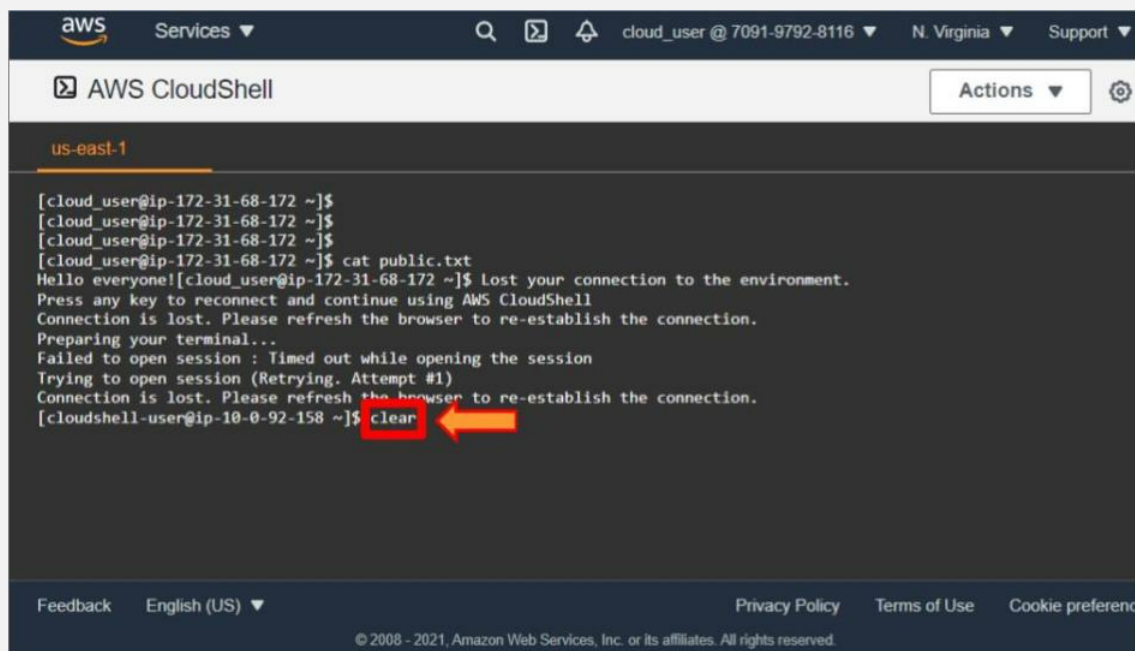


11. Click on the “Permissions” tab to confirm group policies.



## Step 5: Test Permissions

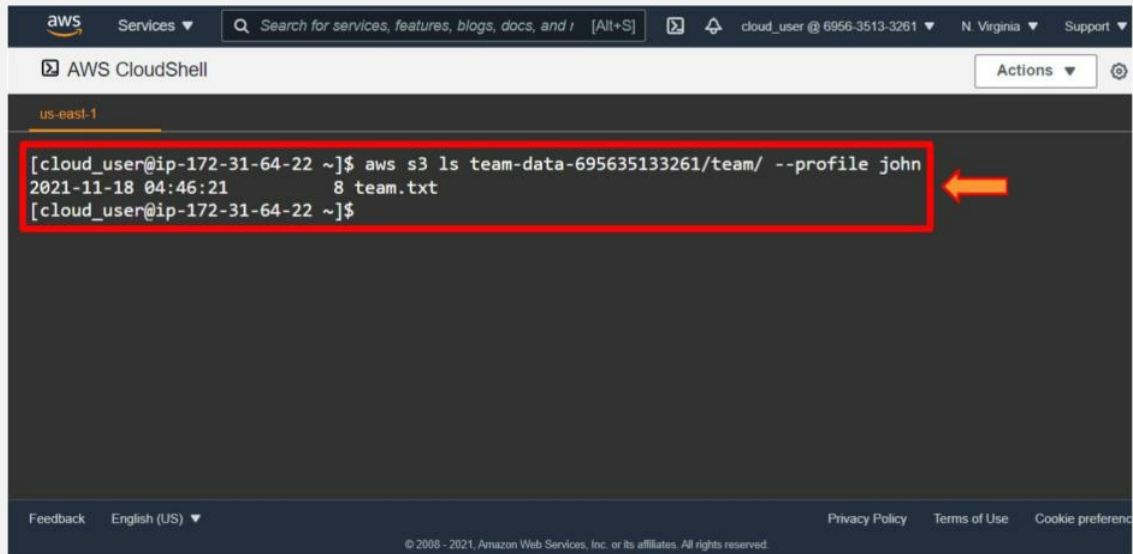
1. Go back to the “CloudShell” browser tab.
2. Type the “Clear” command to clear the terminal.



3. Run a listing to verify that “John” can view the contents of the “Team/ prefix” in your S3 bucket. Be sure to replace “<BUCKET\_NAME>” with your bucket name. You should not see the team.txt object by executing the below command in the

terminal.

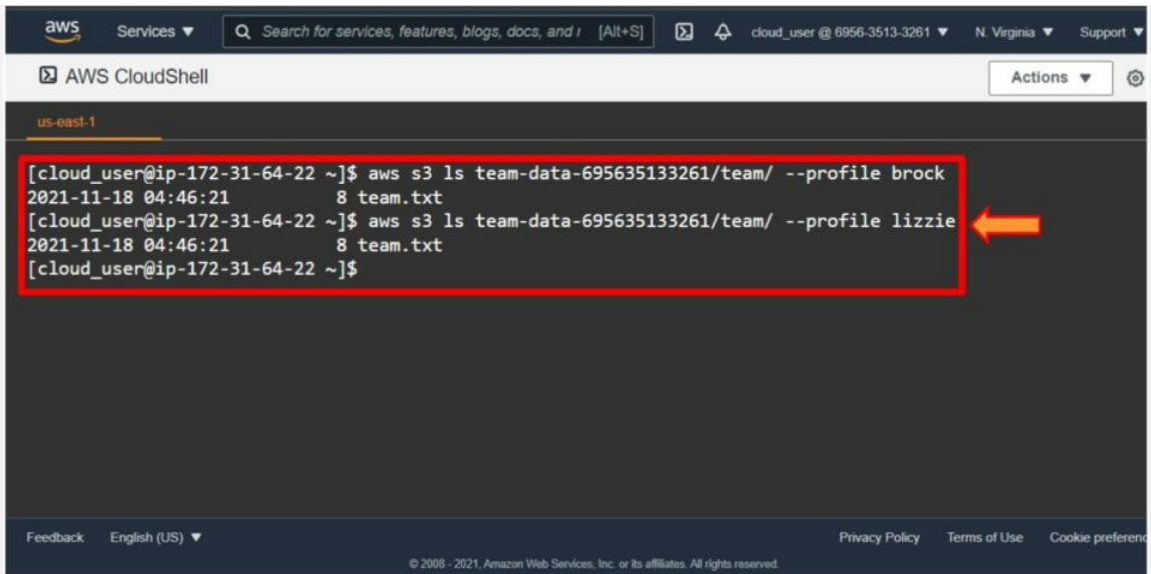
```
aws s3 ls <BUCKET_NAME>/team/ --profile john
```



4. Run the below command for “**Brock**” and “**Lizzie**” to verify the can also view the contents of the “**Team/** prefix” in your S bucket.

```
aws s3 ls <BUCKET_NAME>/team --profile brock
```

```
aws s3 ls <BUCKET_NAME>/team --profile lizzie
```



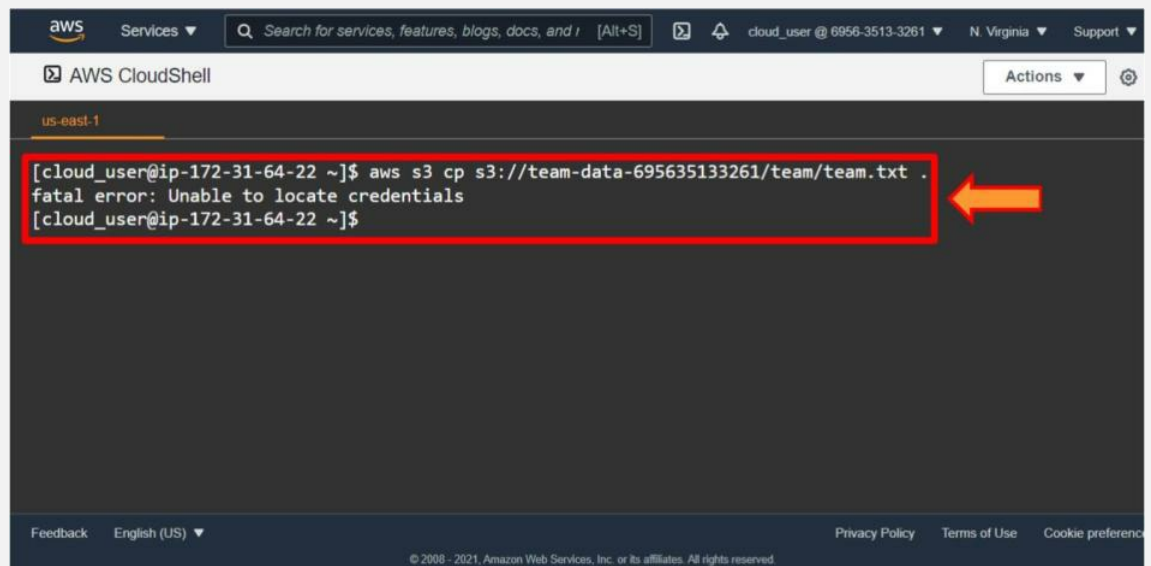
The screenshot shows the AWS CloudShell interface. The terminal output is as follows:

```
[cloud_user@ip-172-31-64-22 ~]$ aws s3 ls team-data-695635133261/team/ --profile brock
2021-11-18 04:46:21      8 team.txt
[cloud_user@ip-172-31-64-22 ~]$ aws s3 ls team-data-695635133261/team/ --profile lizzie
2021-11-18 04:46:21      8 team.txt
[cloud_user@ip-172-31-64-22 ~]$
```

A red box highlights the two successful command executions, and an orange arrow points to the second command.

5. Try to download the “**team.txt**” object without adding profile credentials. Be sure to replace “<BUCKET\_NAME>” with your bucket name. You should receive an error message by executing the below command on the terminal.

```
aws s3 cp s3://<BUCKET_NAME>/team/team.txt .
```



The screenshot shows the AWS CloudShell interface. The terminal output is as follows:

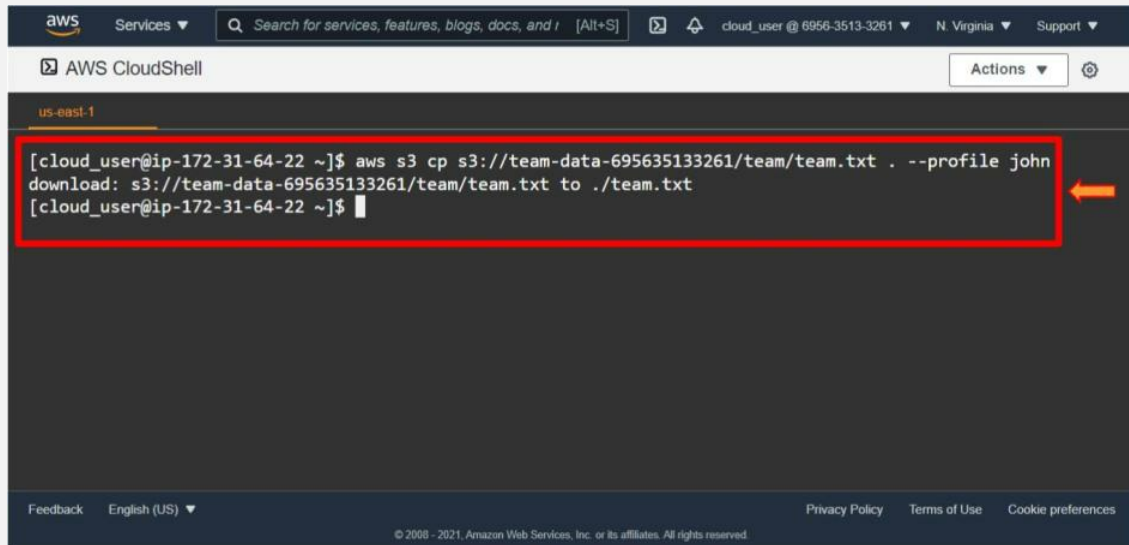
```
[cloud_user@ip-172-31-64-22 ~]$ aws s3 cp s3://team-data-695635133261/team/team.txt .
fatal error: Unable to locate credentials
[cloud_user@ip-172-31-64-22 ~]$
```

A red box highlights the error message, and an orange arrow points to it.

6. Run the below command, but this time add profile credentials for “**John**.” The command should now be successful.

```
aws s3 cp s3://<BUCKET_NAME>/team/team.txt . --
```

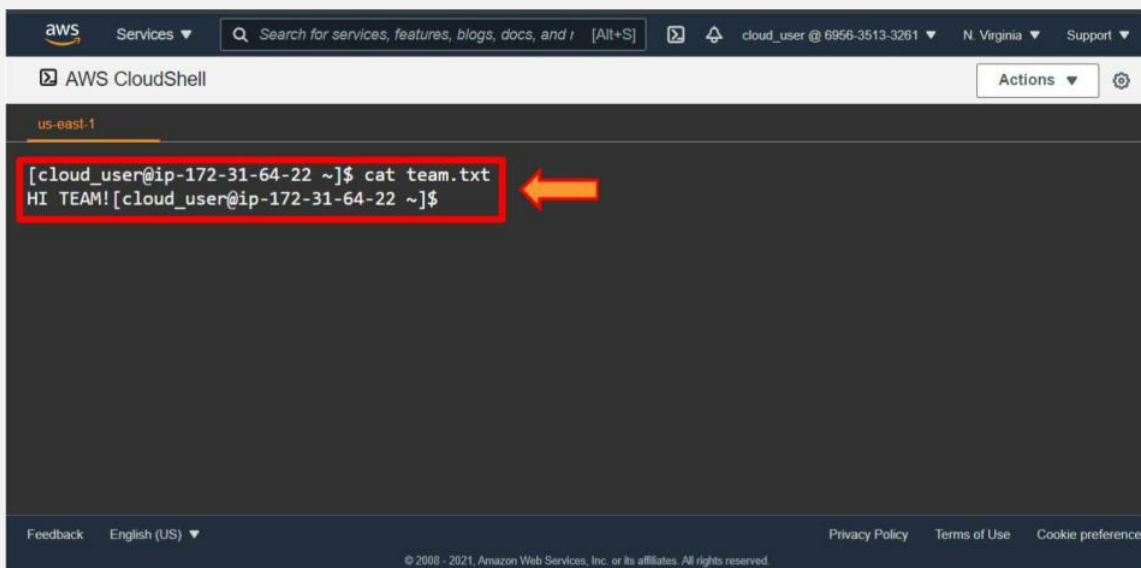
```
profile john
```



```
aws
Services
Search for services, features, blogs, docs, and i [Alt+S]
cloud_user @ 6956-3513-3261 N. Virginia Support
AWS CloudShell
Actions
us-east-1
[cloud_user@ip-172-31-64-22 ~]$ aws s3 cp s3://team-data-695635133261/team/team.txt . --profile john
download: s3://team-data-695635133261/team/team.txt to ./team.txt
[cloud_user@ip-172-31-64-22 ~]$
```

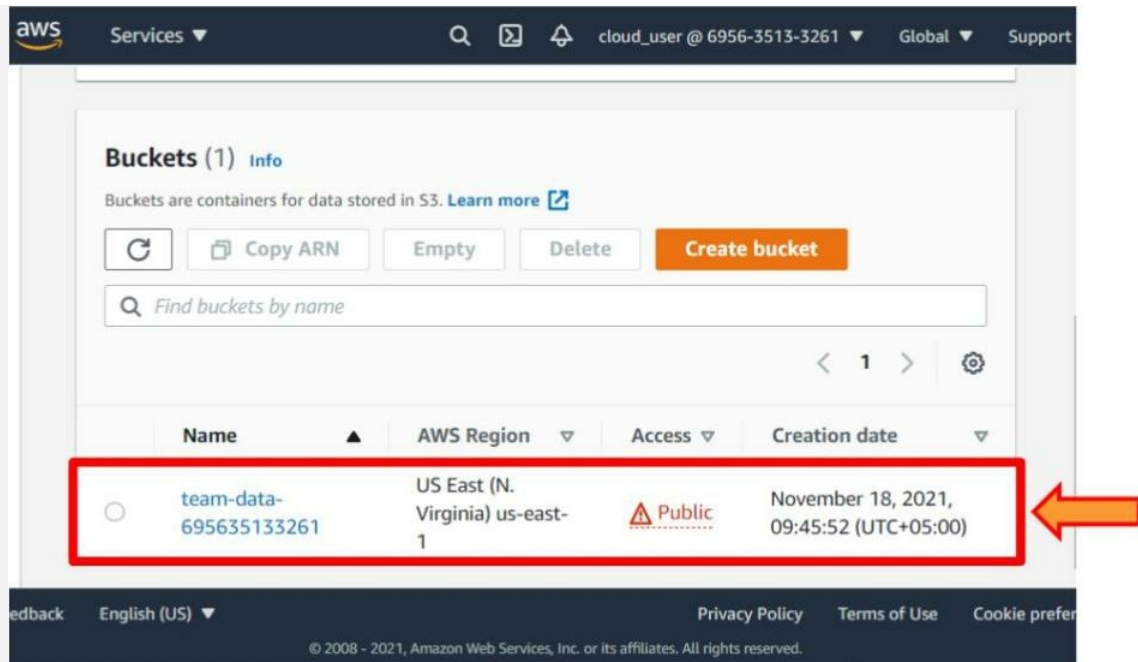
7. Run the “**cat**” command on “**team.txt**.” You should see a “**HI TEAM!**” message by executing the below command on the terminal.

```
cat team.txt
```

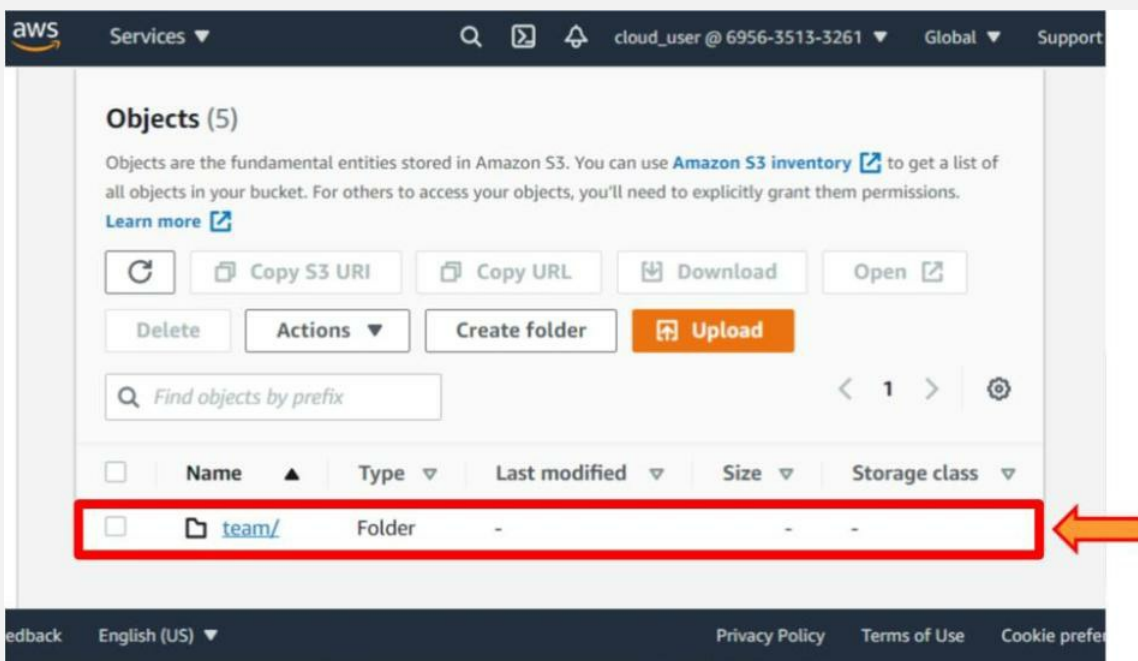


```
aws
Services
Search for services, features, blogs, docs, and i [Alt+S]
cloud_user @ 6956-3513-3261 N. Virginia Support
AWS CloudShell
Actions
us-east-1
[cloud_user@ip-172-31-64-22 ~]$ cat team.txt
HI TEAM! [cloud_user@ip-172-31-64-22 ~]$
```

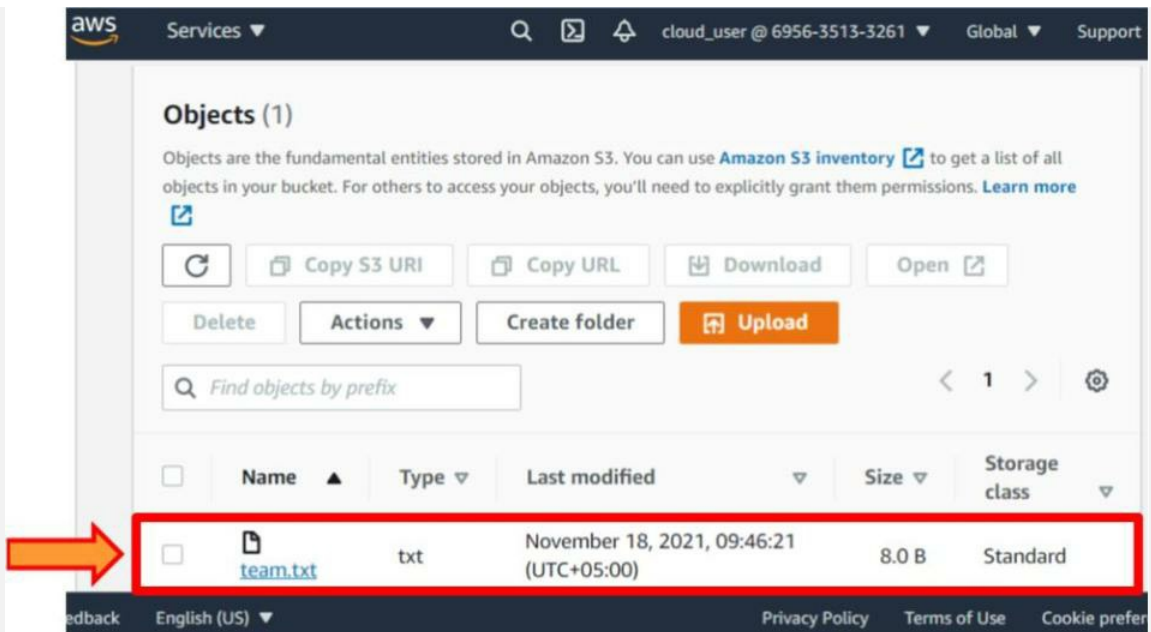
8. Verify that access to the “**team.txt**” object is not public.
9. Go back to the “**S3**” dashboard. Click on the “**team-data**” bucket



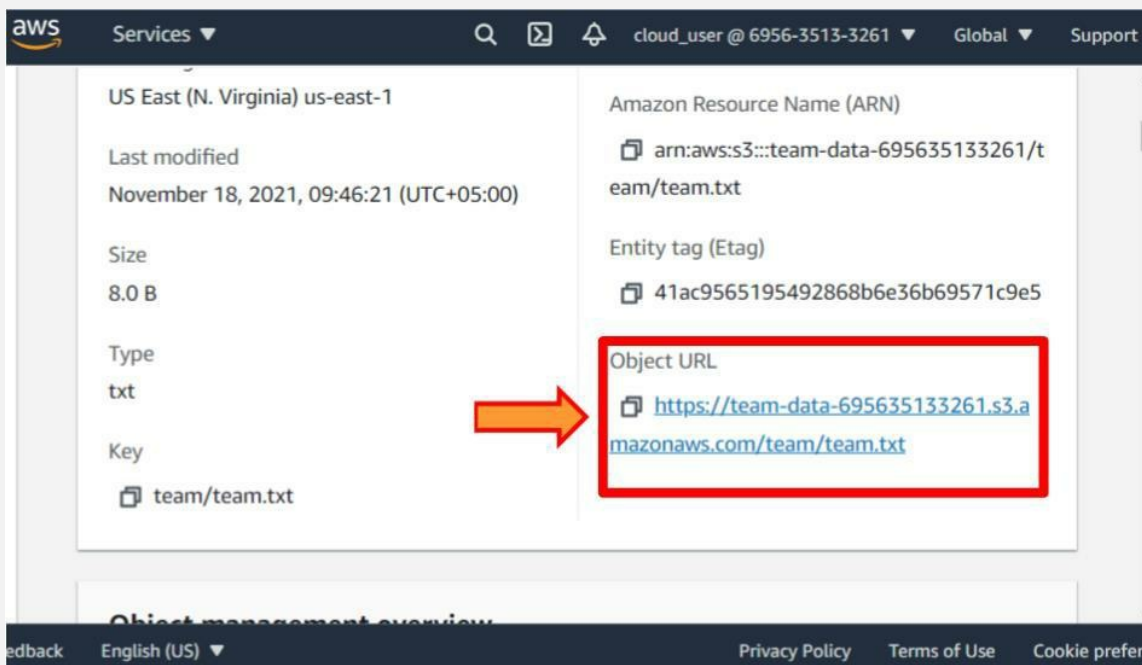
10. Click on the “Team/ folder.”



11. Click on the “team.txt” object.



12. Click on the “Object URL” in a separate browser.



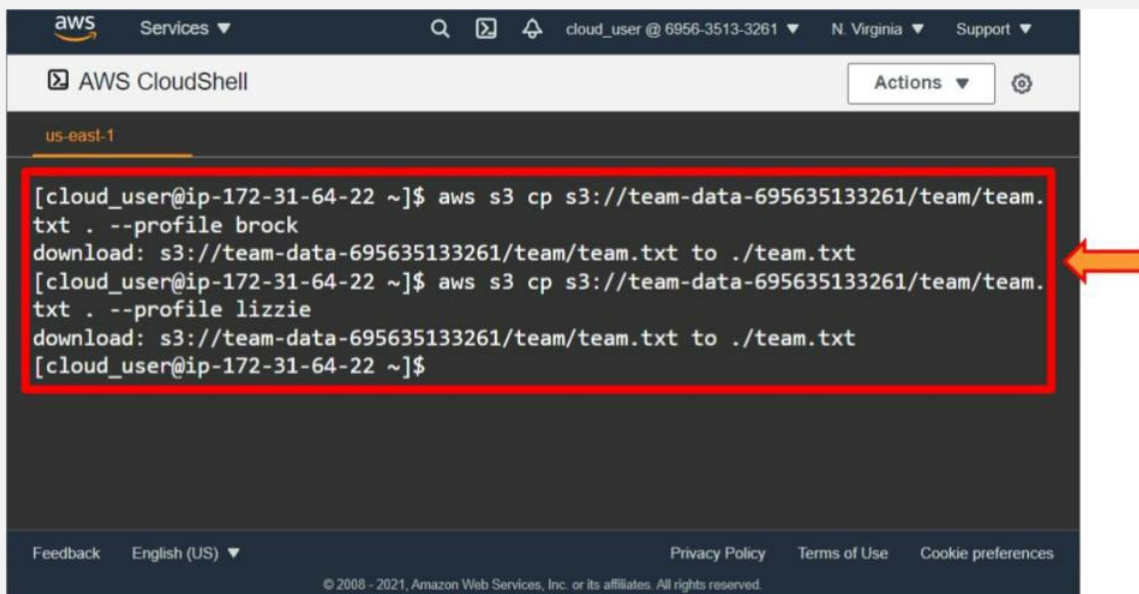
13. You should see an “Access Denied” error.



14. Go back to “**CloudShell**.” Download the “**team.txt**” object using profile credentials for “**Brock**” and “**Lizzie**.” Be sure to replace “**<BUCKET\_NAME>**” with your bucket name. The below commands should be successful for both user profiles.

```
aws s3 cp s3://<BUCKET_NAME>/team/team.txt . --  
profile brock
```

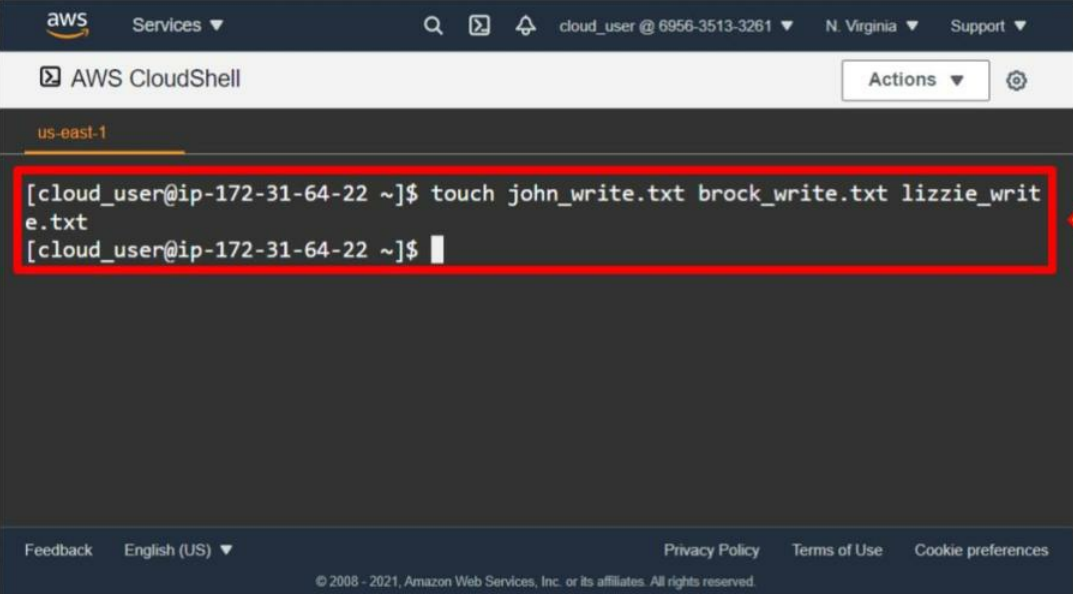
```
aws s3 cp s3://<BUCKET_NAME>/team/team.txt . --  
profile lizzie
```



15. Run the “**Touch**” command to verify that all three users can write to your S3 bucket prefixes. You should see the three “**.txt**” objects you entered in the command and the “**public.txt**” an

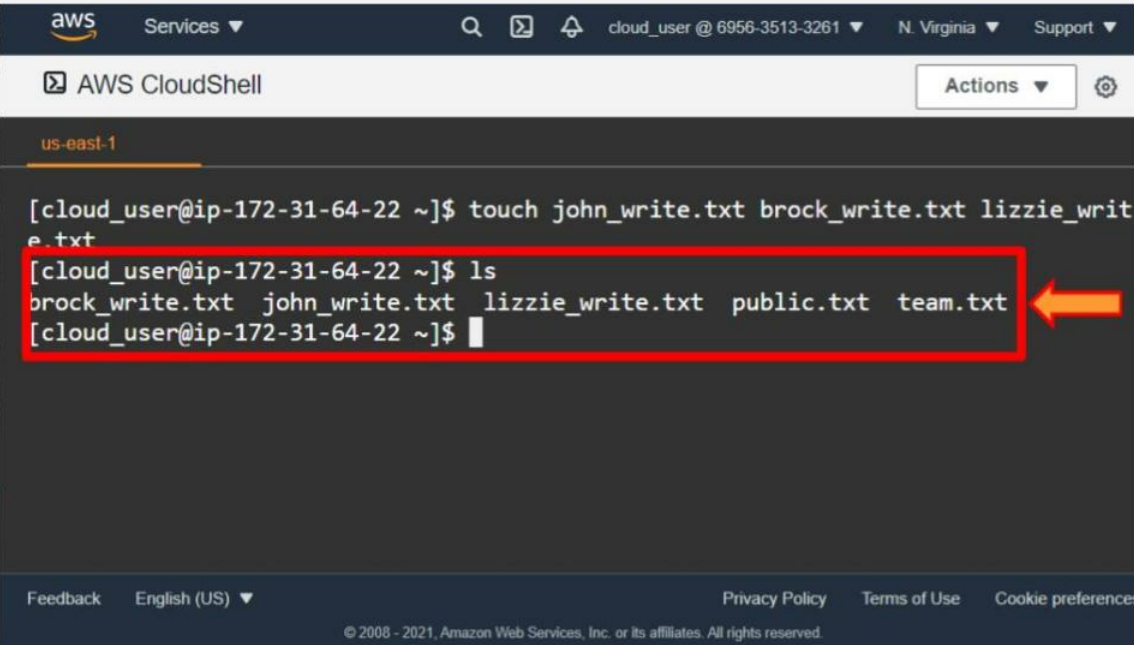
**“team.txt”** objects.

```
touch          john_write.txt          brock_write.txt  
lizzie_write.txt
```



The screenshot shows the AWS CloudShell interface. The terminal window displays the command `touch john_write.txt brock_write.txt lizzie_write.txt` being executed. The command is highlighted with a red box, and an orange arrow points to it from the right. The terminal prompt is `[cloud_user@ip-172-31-64-22 ~]$`. The AWS CloudShell header shows the user `cloud_user @ 6956-3513-3261` and the region `N. Virginia`.

16. Verify files were successfully created by executing the “ls” command.



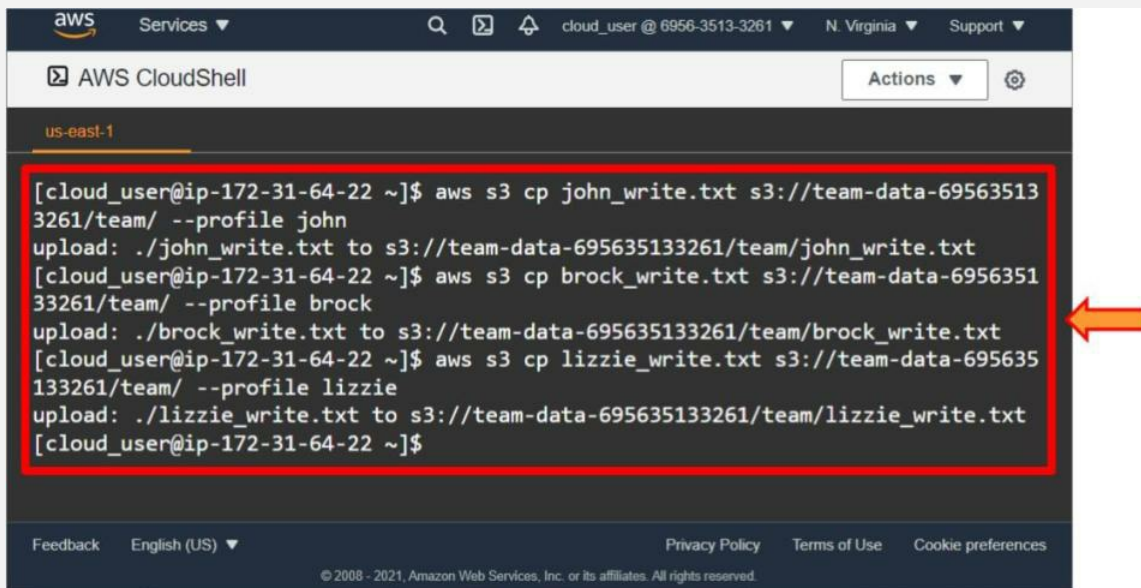
The screenshot shows the AWS CloudShell interface. The terminal window displays the command `ls` being executed. The command is highlighted with a red box, and an orange arrow points to it from the right. The terminal prompt is `[cloud_user@ip-172-31-64-22 ~]$`. The output of the command is `brock_write.txt john_write.txt lizzie_write.txt public.txt team.txt`. The AWS CloudShell header shows the user `cloud_user @ 6956-3513-3261` and the region `N. Virginia`.

17. Verify all three users can write to the “**team/** prefix.” Be sure to replace “<BUCKET\_NAME>” with your bucket name in the below commands.

```
aws s3 cp john_write.txt s3://<BUCKET_NAME>/team/
--profile john

aws s3 cp brock_write.txt s3://<BUCKET_NAME>/team/
--profile brock

aws s3 cp lizzie_write.txt
s3://<BUCKET_NAME>/team/ --profile lizzie
```

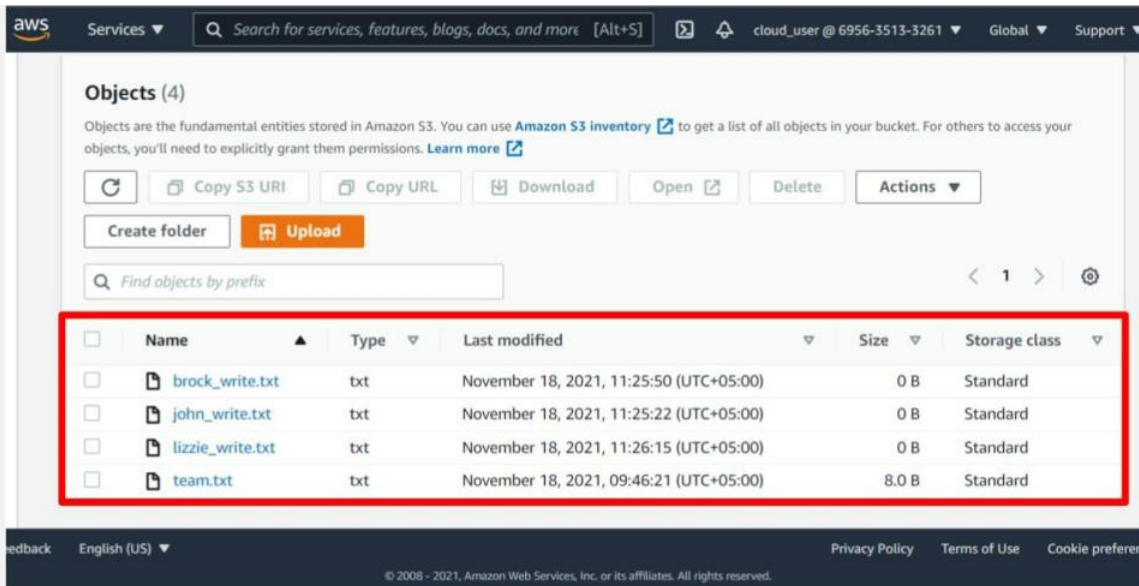


The screenshot shows the AWS CloudShell interface. The terminal output displays the following commands and their results:

```
[cloud_user@ip-172-31-64-22 ~]$ aws s3 cp john_write.txt s3://team-data-695635133261/team/ --profile john
upload: ./john_write.txt to s3://team-data-695635133261/team/john_write.txt
[cloud_user@ip-172-31-64-22 ~]$ aws s3 cp brock_write.txt s3://team-data-695635133261/team/ --profile brock
upload: ./brock_write.txt to s3://team-data-695635133261/team/brock_write.txt
[cloud_user@ip-172-31-64-22 ~]$ aws s3 cp lizzie_write.txt s3://team-data-695635133261/team/ --profile lizzie
upload: ./lizzie_write.txt to s3://team-data-695635133261/team/lizzie_write.txt
[cloud_user@ip-172-31-64-22 ~]$
```

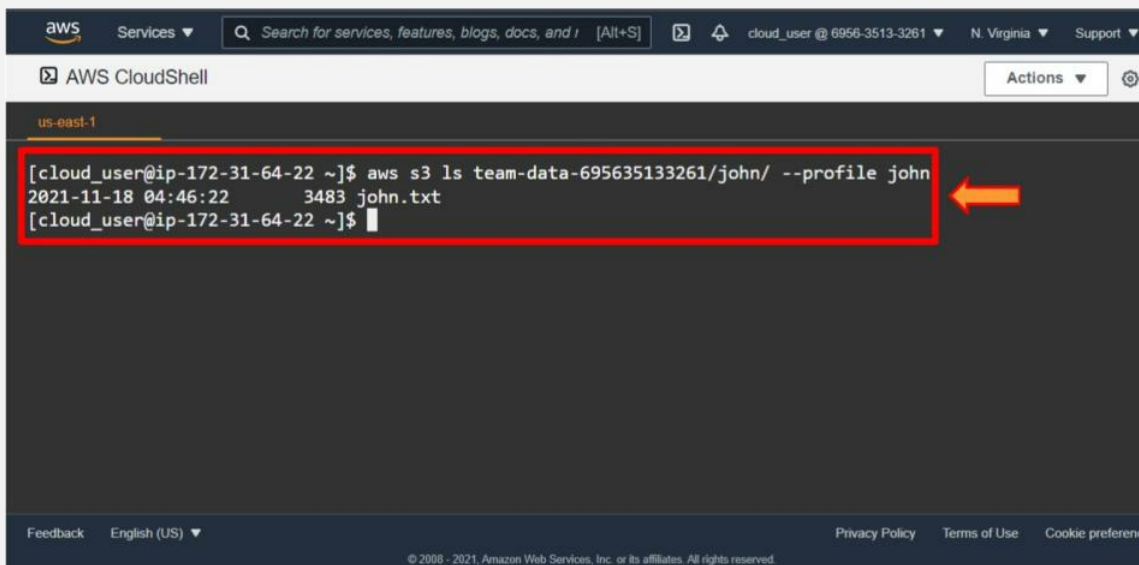
A red box highlights the three commands, and a red arrow points to the third command.

18. Go back to the “AWS S3” dashboard. Click on the “**Team/** folder in your S3 bucket. You should now see the “**john\_write.txt**”, “**brock\_write.txt**”, and “**lizzie\_write.txt**” objects you uploaded.



19. Go back to “CloudShell.” You should see the john.txt object b  
executing the below command on the terminal. Ensure john ca  
access his user folder. Be sure to replace “<BUCKET\_NAME>  
with your bucket name.

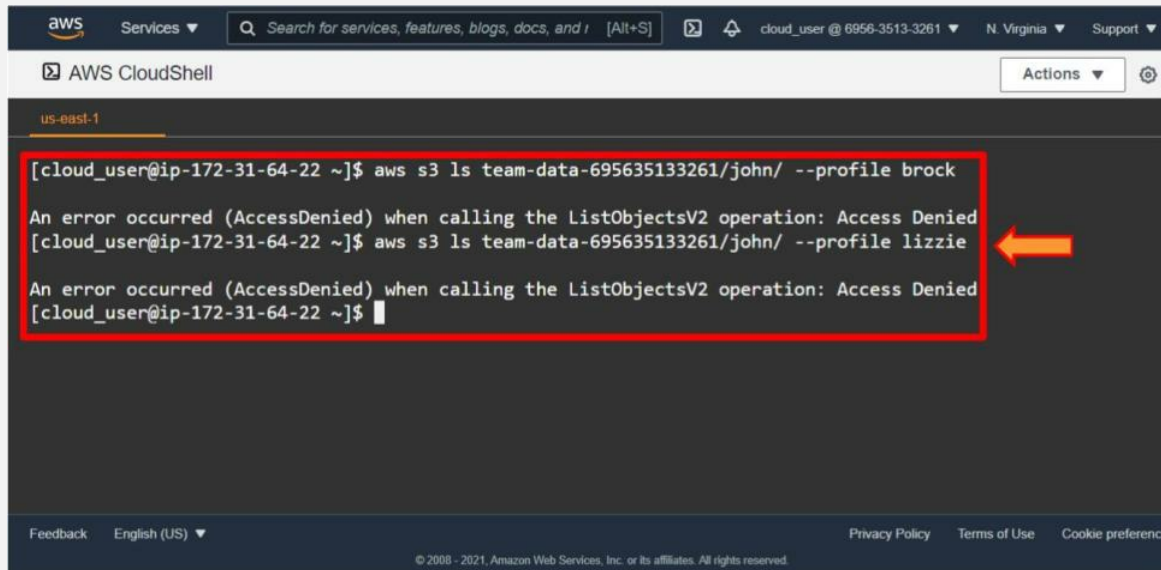
```
aws s3 ls <BUCKET_NAME>/john/ --profile john
```



20. Test whether the “Brock” and “Lizzie” user profiles can acces  
the “John” folder. You should see an “Access Denied” error  
indicating that only “John” can access his folder by executing th

below commands on the terminal.

```
aws s3 ls <BUCKET_NAME>/john/ --profile brock  
aws s3 ls <BUCKET_NAME>/john/ --profile lizzie
```



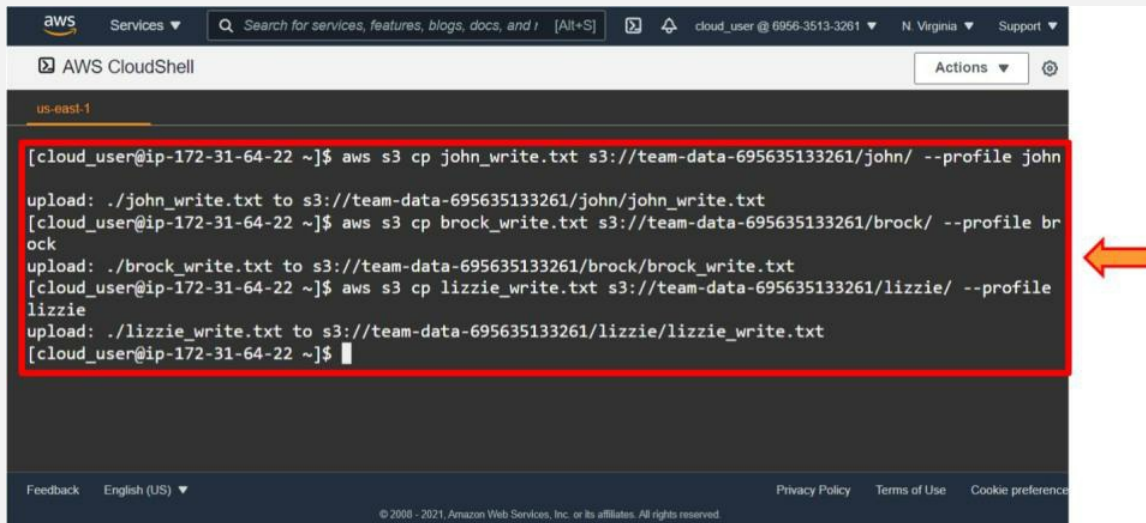
21. Verify that the “**John**” user profile can write to the “**John/**” folder by executing the below command on the terminal.

```
aws s3 cp s3://<BUCKET_NAME>/john/john.txt . --  
profile john
```



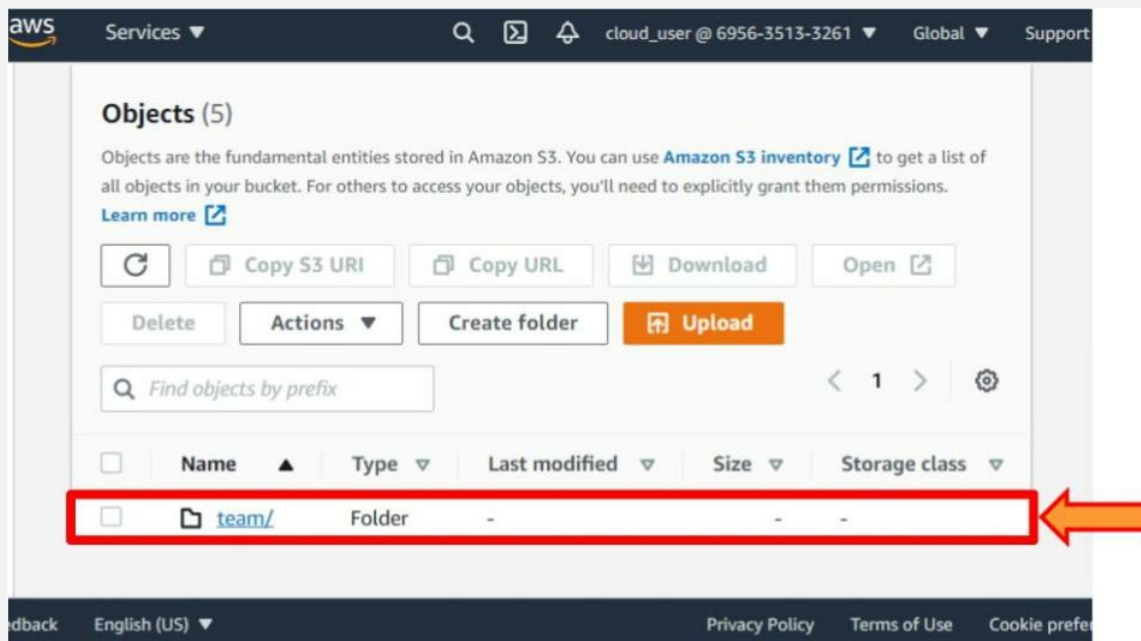
```
aws s3 cp brock_write.txt
s3://<BUCKET_NAME>/brock/ --profile brock

aws s3 cp lizzie_write.txt
s3://<BUCKET_NAME>/lizzie/ --profile lizzie
```

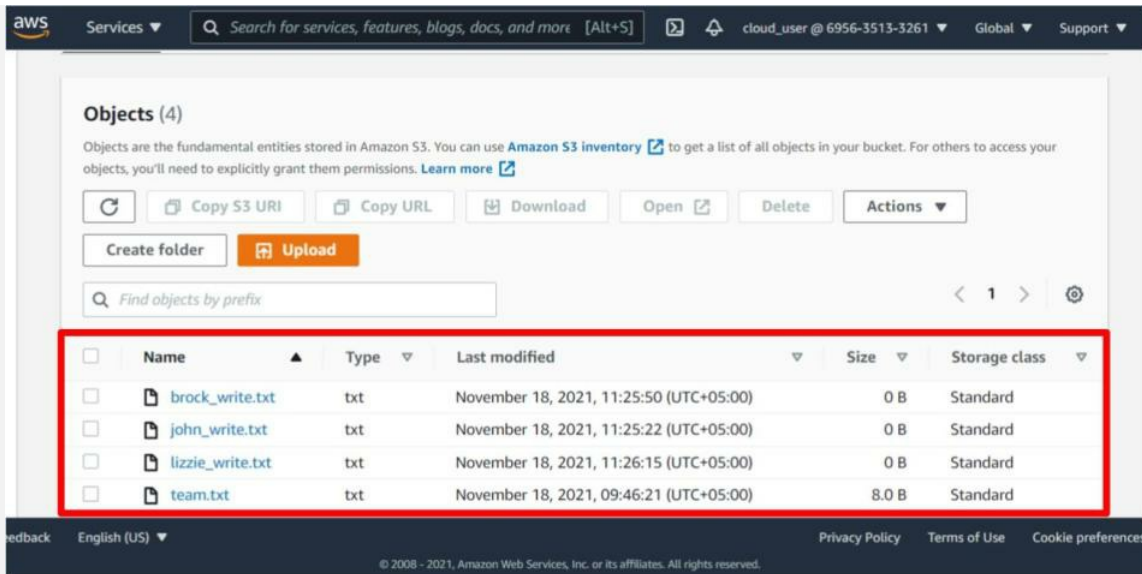


```
[cloud_user@ip-172-31-64-22 ~]$ aws s3 cp john_write.txt s3://team-data-695635133261/john/ --profile john
upload: ./john_write.txt to s3://team-data-695635133261/john/john_write.txt
[cloud_user@ip-172-31-64-22 ~]$ aws s3 cp brock_write.txt s3://team-data-695635133261/brock/ --profile brock
upload: ./brock_write.txt to s3://team-data-695635133261/brock/brock_write.txt
[cloud_user@ip-172-31-64-22 ~]$ aws s3 cp lizzie_write.txt s3://team-data-695635133261/lizzie/ --profile lizzie
upload: ./lizzie_write.txt to s3://team-data-695635133261/lizzie/lizzie_write.txt
[cloud_user@ip-172-31-64-22 ~]$
```

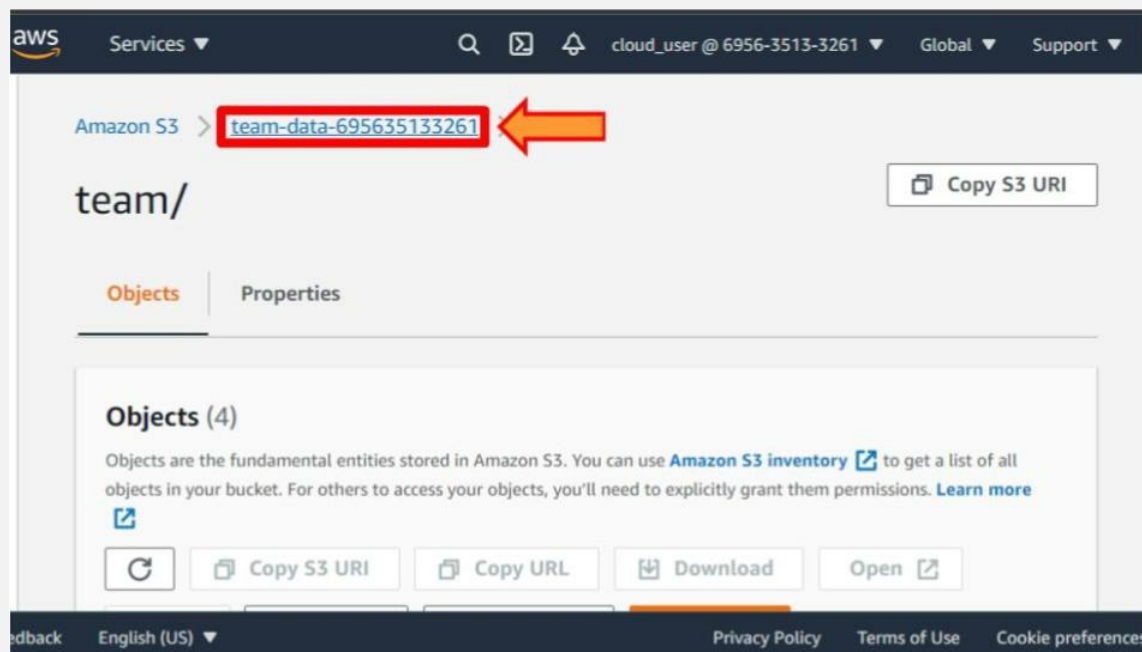
24. Go back to the “S3” dashboard. Click on the “Team/ folder.”



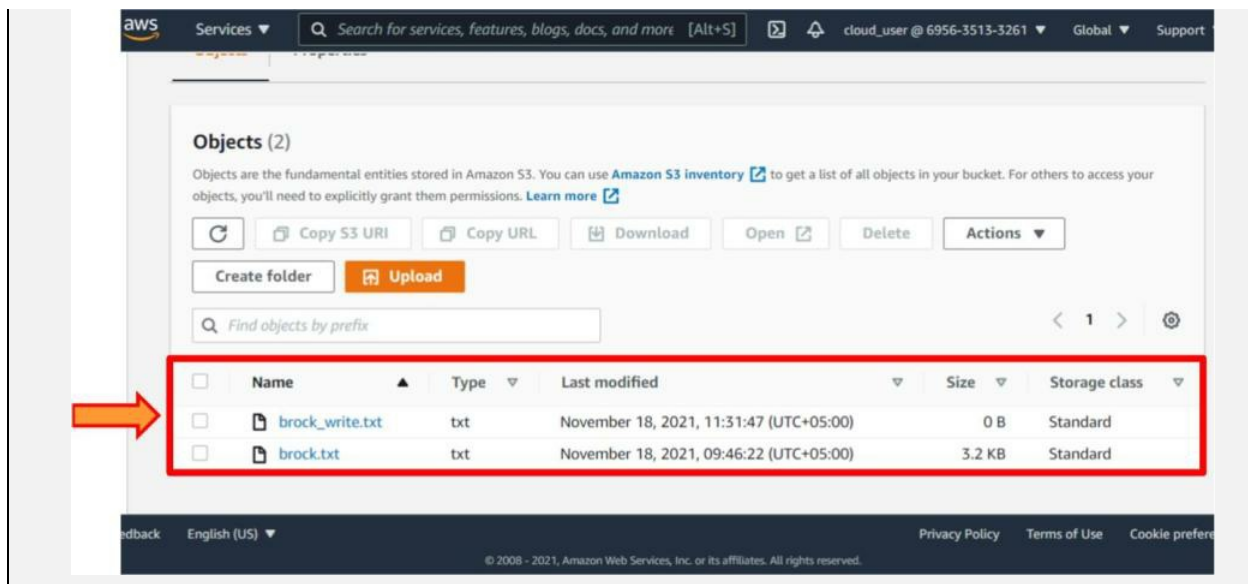
25. Confirm there are four objects listed “brock\_write.txt, john\_write.txt, lizzie\_write.txt, and team.txt.”



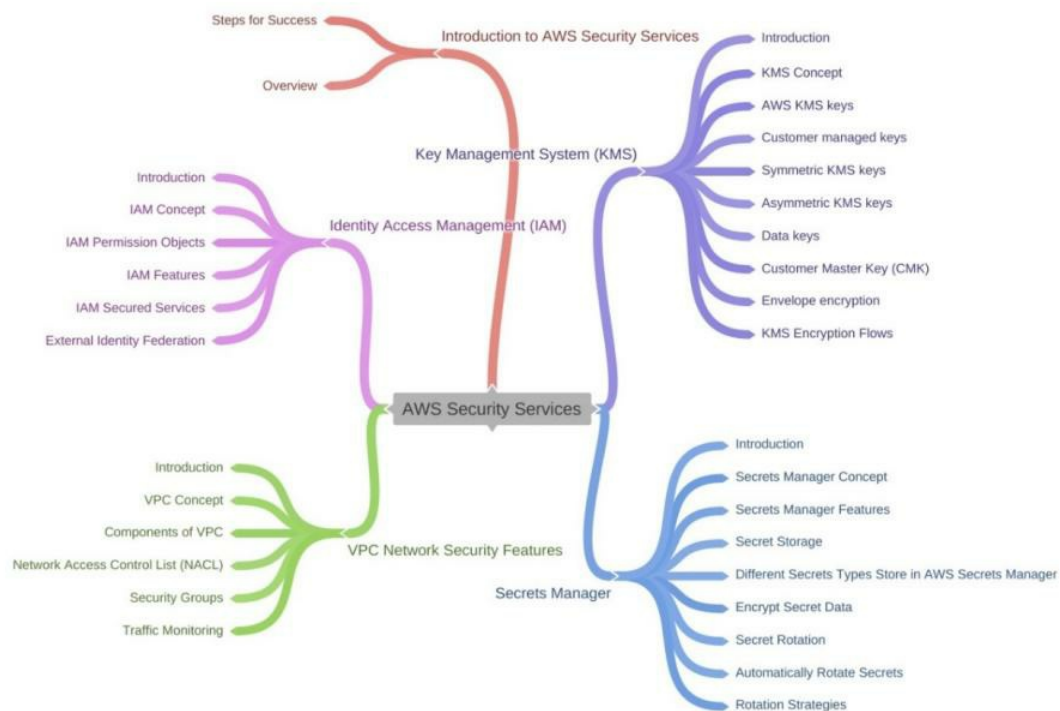
26. Click on the “team-data” bucket root.



27. Click on each “User” folder and verify they all “contain user\_write.txt” and “user.txt objects.”



## Mind Map



*Figure 11-17: Mind Map AWS Security Services*

## Practice Questions

1. Which AWS resource provides secured control access to AWS resources such as compute storage and database?
  - A. AWS VPC
  - B. AWS KMS
  - C. AWS IAM
  - D. AWS Secrets Manager
2. Which is a unique identity with limited access to an AWS account and its resources as defined by their IAM permissions and policies?
  - A. IAM Users
  - B. IAM Groups
  - C. IAM Roles
  - D. IAM Policies
3. Which is a collection of users you can use to specify permissions to collect users, making it easier to manage permissions for them?
  - A. IAM Users
  - B. IAM Groups
  - C. IAM Roles
  - D. IAM Policies
4. Which IAM entity lets you define a set of permissions to access the resources that a user or service needs?
  - A. IAM Users
  - B. IAM Groups
  - C. IAM Roles

D. IAM Policies

5. Which is a rule or set of rules defining the operations allowed/denied to be performed on an AWS resource?
  - A. IAM Users
  - B. IAM Groups
  - C. IAM Roles
  - D. IAM Policies
6. Which AWS service allows you to produce and govern the keys used in cryptographic activities?
  - A. AWS VPC
  - B. AWS KMS
  - C. AWS IAM
  - D. AWS Secrets Manager
7. Which key is a 256-bit encryption key that never leaves the system unencrypted?
  - A. AWS KMS
  - B. Symmetric KMS Key
  - C. Asymmetric KMS Key
  - D. AWS IAM
8. Which KMS key is a mathematically connected pair of the public?
  - A. Symmetric KMS Key
  - B. Data Keys
  - C. Customer Master Key
  - D. Asymmetric KMS Key
9. Which encryption key in AWS is used to encrypt data,

especially enormous volumes of data?

- A. Symmetric KMS Key
  - B. Data Keys
  - C. Customer Master Key
  - D. Asymmetric KMS Key
10. Which AWS resource allows you to replace hardcoded credentials such as passwords in your code with an API call?
- A. AWS VPC
  - B. AWS KMS
  - C. AWS IAM
  - D. AWS Secrets Manager
11. The process of updating a secret regularly in AWS Secrets Manager is known as \_\_\_\_\_.
- A. Rotation
  - B. Spinning
  - C. Swinging
  - D. Twisting
12. The AWS Secrets Manager stores data in which format?
- A. YAML
  - B. SQL
  - C. JSON
  - D. Text
13. Which AWS resource allows you to provision a logically isolated section of the AWS cloud to launch AWS services in a virtual private network that defines you?
- A. AWS VPC

- B. AWS KMS
  - C. AWS IAM
  - D. AWS Secrets Manager
14. Which is an optional security layer for your VPC that operates as firewall traffic to manage traffic in and out or more subnets?
- A. NAT Gateway
  - B. Network ACL
  - C. Router
  - D. Peering Connection
15. Which AWS VPC feature allows you to repeat network traffic from the elastic network interface of an AWS EC2 instance?
- A. Subnets
  - B. Security Group
  - C. Traffic Monitoring
  - D. Router



# ANSWERS

## Chapter 02: Amazon Simple Storage Service

1. (A) True

**Explanation:** Amazon S3 is a type of object storage that allows you to store and recover any quantity of data from any location. It is a low-cost storage solution with business resilience, reliability, efficiency, privacy, and infinite expansion. S3 uses an object called the bucket. A bucket is an atomic unit for S3.

2. (B) 3

**Explanation:** When we upload data at S3, we have three interfaces to work with. The AWS management console, the AWS CLI, and several AWS SDKs.

3. (C) AWS Management console

**Explanation:** When we use the management console, we use a graphical user interface.

4. (A) AWS CLI

**Explanation:** If we are using the AWS CLI, we enter commands in our terminal that will allow us to move data into our S3 bucket. We can also use these commands to retrieve data from our S3 buckets.

5. (A) True

**Explanation:** Transfer Acceleration is enabled per bucket.

6. (B) False

**Explanation:** There are additional costs for using Transfer Acceleration. You are charged 4 cents per gigabyte of data transferred

from the United States to Europe and Japan edge locations and 8 cents per gigabyte for all other AWS edge locations.

7. (A) True

**Explanation:** Transfer Acceleration leverages the edge locations to send data back to S3. Hence, this becomes a content ingestion network and not just a distribution network. This means that our users use optimized network links to get our data or send data to us.

8. (A) True

**Explanation:** To get a five terabyte object into S3, we will use multipart upload.

9. (C) Three

**Explanation:** We use three API calls to perform the multipart upload process.

CreateMultipartUpload, UploadPart and CompleteMultipartUpload API call.

10. (B) False

**Explanation:** We can overwrite the parts while the multipart upload is still in progress. Suppose something in your log file changes, or there is an update to a section of it. You can overwrite that part that area is in and have the latest version of the log file in your object when it uploads, or if one of the parts fails or has some corrupted data in it, you can write it again multipart upload. It will use whatever the latest data is for that piece of the upload when it is reassembled.

11. (A) S3 Standard

**Explanation:** S3 Standard is a storage type for general-purpose storage

of commonly accessed data.

12. (B) S3 Intelligent-Tiering

**Explanation:** S3 Intelligent-Tiering is utilized for data with uncertain or changing access patterns.

13. (A) S3 Glacier

**Explanation:** S3 Glacier offers three retrieval options, ranging from a few minutes to hours, to keep prices reasonable while meeting various demands.

14. (A) True

**Explanation:** S3 Glacier Deep Archive is Amazon S3's cheapest storage tier. It allows long-term data retention and digital preservation for material only viewed once or twice a year.

15. (A) S3 Glacier

**Explanation:** The S3 Glacier is an Archival storage class with minutes to hours of data retrieval time.

## Chapter 03: Databases in AWS

### 1. (C) Storage Engine

**Explanation:** A database engine (sometimes known as a storage engine) is the software component that enables a database management system (DBMS) to generate, read, update, and delete (CRUD) data from a database. Most database management systems have an application programming interface (API) that allows users to communicate with the underlying engine without going via the DBMS's user interface.

### 2. (A) Relational Database

**Explanation:** A relational database collects data objects with specified relationships and can be easily retrieved. In the relational database paradigm, data structures such as data tables, indexes, and views are maintained distinct from physical storage structures, allowing database managers to alter the physical data storage without affecting the logical data structure.

### 3. (C) Row Database

**Explanation:** row-oriented databases arrange data by the record and keep all data related with a record in memory next to each other. Row-oriented databases are the conventional method of data organization, and they still offer some important advantages for storing data fast. They have been designed to read and write rows quickly.

### 4. (D) Columnar Database

**Explanation:** A columnar database is a database management system (DBMS) in which data is stored in columns rather than rows. A columnar database's goal is to reduce the time it takes to return a query by quickly writing and reading data to and from hard disc storage. Columnar databases store data so that disc I/O speed is considerably improved. They are very useful for data warehousing and analytics.

#### 5. (B) Non-relational Database

**Explanation:** Non-relational databases (often called 'NoSQL' or 'JSON' or 'key: value' databases) differ from standard relational databases because they are stored in a non-tabular format. Non-relational databases are far more versatile than relational databases because they can digest and arrange many kinds of information concurrently. Non-relational databases, on the other hand, may be built on data formats such as documents. A document can be quite thorough while also including various data types in multiple forms.

#### 6. (B) Amazon Neptune

**Explanation:** Amazon Neptune is a fast, trustworthy, and fully-managed graph database service that simplifies the design and operation of applications that deal with vast, linked datasets. Neptune is designed around a purpose-made, high-performance graph database engine. Neptune drives graph application cases, including recommendation engines, fraud detection, knowledge graphs, medicine discovery, and network security.

#### 7. (C) Amazon DocumentDB

**Explanation:** DocumentDB is a document store engine. Document

store engines typically are going to format their data as JSON. Amazon DocumentDB (with MongoDB compatibility) is a highly scalable, dependable, and completely managed database service. Amazon DocumentDB simplifies the setup, operation, and scaling of MongoDB-compatible databases in the cloud.

#### 8. (A) Graph Structure

**Explanation:** A graph is a nonlinear data structure of nodes and edges. A graph is a data structure consisting of a finite number of nodes (or vertices) and edges that link them. The nodes are also known as vertices, while edges are lines or arcs that connect any two nodes in the graph. An edge (x,y) signals that the x vertex relates to the y.

#### 9. (D) OLTP

**Explanation:** These engines are excellent for OLTP or online transaction processing. They are used for rapid transactions where you deal with relatively small pieces of data that will protect the data through transaction rollback, which the columnar engines do

#### 10. (B) OLIVE

**Explanation:** Columnar engines are good for OLAP or online analytics processing, and when you are talking about data analytics and AWS, they will be fairly important, particularly RedShift. The RedShift database is excellent at handling large amounts of data.

#### 11. (A) S3 Select

**Explanation:** Using simple SQL expressions, S3 Select allows apps to get only a subset of data from an object. You may drastically improve speed by retrieving only the data required by your application using

S3 Select. In many circumstances, you might expect to see a 400% improvement.

12. (B) Athena

**Explanation:** Amazon Athena is an interactive query service that allows you to use conventional SQL to evaluate data directly in Amazon Simple Storage Service (Amazon S3). With a few clicks in the AWS Management Console, you can aim Athena at your Amazon S3 data and start running ad-hoc searches with conventional SQL to obtain results in seconds.

13. (C) DynamoDB

**Explanation:** Amazon DynamoDB is a fully managed NoSQL database service that delivers quick and predictable performance while seamless scaling. You can offload the administrative requirements of running and growing a distributed database using DynamoDB, so you do not have to worry about hardware provisioning, setup, configuration, replication, software patching, or cluster scalability. DynamoDB also supports encryption at rest, removing the operational load and complexity associated with securing sensitive data.

14. (A) Aurora

**Explanation:** Amazon Aurora (Aurora) is a relational database engine that is fully managed and compatible with MySQL and PostgreSQL. You already know that MySQL and PostgreSQL combine the performance and durability of high-end commercial databases with the simplicity and low cost of open-source databases. The code, tools, and applications you are now using with your

existing MySQL and PostgreSQL databases may be used with Aurora.

15. (D) Aurora-Serverless

**Explanation:** Amazon Aurora Serverless v1 (Amazon Aurora Serverless version 1) is a configuration for on-demand autoscaling in Amazon Aurora. Aurora Serverless v1 is a simple, low-cost choice for occasional, intermittent, or unexpected workloads. An Aurora Serverless DB cluster is a database cluster that dynamically scales processing capacity based on the needs of your application. In contrast, Aurora supplied DB clusters require manual capacity management.

## Chapter 04: Collecting Streaming Data

### 1. (A) Kinesis Data Stream

**Explanation:** You may gather and handle vast streams of data records in real-time using Amazon Kinesis Data Streams. Data-processing applications can be created, often known as Kinesis Data Streams applications. A typical Kinesis Data Streams application reads data records from a data stream. These applications may operate on Amazon EC2 instances and use the Kinesis Client Library. You may utilize the processed records to create dashboards, issue alerts, modify pricing and advertising tactics dynamically, and transfer data to several other AWS services.

### 2. (B) Kinesis Data Firehose

**Explanation:** Amazon Kinesis Data Firehose is a fully managed service that delivers real-time streaming data to destinations, such as Amazon S3, Amazon Redshift, Amazon OpenSearch Service (Amazon ES), Splunk, and any custom HTTP endpoint or HTTP endpoints owned by supported third-party service providers like Datadog, Dynatrace, LogicMonitor, MongoDB, New Relic, and Sumo Log.

### 3. (C) Kinesis Data Video streams

**Explanation:** Amazon Kinesis Video Streams is a fully managed AWS service. It allows you to

stream live video from devices to the AWS Cloud and develop applications for real-time video processing and batch-oriented video analytics.

Kinesis Video Streams offers more than simply video data storage. You may use it to see your video feeds as they arrive in the cloud in real-time. You may either watch your live streams via the AWS Management Console or create your monitoring application that displays live video using the Kinesis Video Streams API library.

#### 4. (D) Kinesis Data Analytics

**Explanation:** You may use standard SQL to handle and analyze streaming data using Amazon Kinesis Data Analytics for SQL Applications. The service lets you quickly develop and run sophisticated SQL code against streaming sources to do time-series analytics, feed real-time dashboards, and generate real-time metrics.

#### 5. (C) Shard

**Explanation:** A shard is a series of data records in a uniquely identifiable stream. A stream comprises one or more shards, each of which has a set capacity unit. Each shard can handle up to five read transactions per second, for a total data read at the pace of two megabytes per second. Up to 1,000 write transactions per second, for a total data write rate of one megabyte per second (including partition keys). The number of shards you choose for your stream determines the data capacity of the stream. The overall capacity of the stream is equal to the sum of its shards' capacities.

#### 6. (A) Kinesis Client Library

**Explanation:** KCL assists you in consuming and processing data from a Kinesis data stream by handling numerous difficult distributed computing jobs. Examples are load balancing across multiple consumer application instances, response to consumer application instance failures, checkpointing processed records, and response to re-sharding.

#### 7. (B) Data Collection

**Explanation:** The practice of gathering, measuring, and evaluating correct research insights using established procedures is called data collection. Based on the evidence gathered, a researcher can assess his hypothesis. Regardless of the subject of study, data gathering is the first and most significant stage in most situations. Depending on the information requested, the approach to data gathering differs for different topics of research.

#### 8. (C) Mobile Devices & Website

**Explanation:** Surveys, interviews, and focus groups are the most common methods for gathering information. Corporations may now collect data from mobile devices, website traffic, server activity, and other relevant sources using Web and analytics technologies depending on the project.

#### 9. (B) Big Data

**Explanation:** Big data describes voluminous amounts of structured, semi-structured, and unstructured data collected by organizations that collect and then mine large amounts of data for information. New approaches for collecting and analyzing data have emerged because it takes a lot of time and money to load big data into a traditional relational database for analysis. In a data lake, raw data with extra

information is aggregated. Machine learning and artificial intelligence systems then employ complicated algorithms to search for repeating patterns.

#### 10. (D) Apache Kafka

**Explanation:** Apache Kafka was originally developed by LinkedIn and was made open source in 2011. The Apache community then took it over and a distributed streaming platform with three key capabilities. These capabilities are the ability to publish and subscribe to streams of records similar to a message queue, streams of records can be effectively stored in the sequence in which they were created, and real-time processing of record streams.

Apache Kafka is a real-time data input and processing distributed data storage system. Streaming data is constantly created by hundreds of data sources, which often transmit data records simultaneously. A streaming platform must manage this continual input of data while still processing it sequentially and progressively.

#### 11. (D) Data Streaming

**Explanation:** Streaming data is continually created by hundreds of data sources, which generally send in data records in tiny batches (order of Kilobytes). Streaming data includes log files created by consumers using mobile or online applications, e-commerce purchases, in-game player activity, social network information, financial trading floors, geospatial services, and telemetry from connected devices or instrumentation in data

centers.

12. (C) Amazon Kinesis

**Explanation:** Amazon Kinesis simplifies the collection, processing, and analysis of real-time streaming data, allowing you to get rapid insights and respond quickly to new data. Amazon Kinesis provides essential features for cost-effectively processing streaming data at any scale, with the freedom to select the tools that best meet your application's needs. Amazon Kinesis may be used to store real-time data such as video, audio, application logs, website clickstreams, and IoT telemetry for machine learning, analytics, and other applications. Instead of needing to wait until all of your data is collected before processing can begin, Amazon Kinesis allows you to process and analyze data as it arrives and respond quickly.

13. (B) The Kinesis Producer Library must be installed as a Java application to use with Kinesis Data Streams.

**Explanation:** The KPL must be installed as a Java application before using your Kinesis Data Streams. There are techniques to process KPL serialized data in AWS Lambda and Java, Node.js, and Python, but none of these solutions mention Lambda.

14. (A) 1 Shard

**Explanation:** In this scenario, there will be a maximum of 10 records per second with a max payload size of 1000 KB (10 records x 100 KB = 1000KB) written to the shard. A single shard can ingest up to 1 MB of data per second, enough to ingest the 1000 KB from the streaming gameplay. Therefore 1 shard is enough to handle the streaming data.

15. (D) DeliveryStreamName and Record (containing the data)

**Explanation:** Kinesis Data Firehose is used as a delivery stream. We do not have to be concerned about shards or partition keys. The Firehose DeliveryStreamName and the Record object are all that is required (which contains the data).

## Chapter 05: Data Collection and Getting Data into AWS

### 1. A (Online Data Transfer)

**Explanation:** Online Data Transfer makes it simple and easy to transfer your data into and out of AWS via online methods.

### 2. B (Snowcone)

**Explanation:** Use Snowcone to collect, process, and move data to AWS online with AWS DataSync.

### 3. B (Snowcone)

**Explanation:** Snowcone loads data to Snowcone through Wi-fi wired 10 GbE networking. You can ship the device with data to AWS for offline data transfer.

### 4. C (Snowball)

**Explanation:** Snowball is used for petabyte-scale data transport with import and export to S3.

### 5. D (Snowmobile)

**Explanation:** Snowmobile is an exabyte-scale data transport solution that uses a secure semi-40-foot shipping container to transfer large amounts of data into and out of AWS.

### 6. B (Snowball Edge)

**Explanation:** Snowball Edge is local storage and large-scale data transfer. Also, local Lambda and EC2 instances compute, and AWS IoT

Greengrass.

7. D (All of them)

**Explanation:** The Snow Family includes Snowball, Snowball Edge, and Snowmobile.

8. A (1 Gbps to 10 Gbps)

**Explanation:** Direct Connect uses 1 Gbps to 10 Gbps dedicated networking.

9. B (Database Migration Service)

**Explanation:** Database Migration Service easily and securely migrates widely used commercial and open-source databases and data warehouses into the cloud.

10. B (2)

**Explanation:** Replication easily replicates your databases and data warehouse between two locations.

11. C (Cross-Region Replication)

**Explanation:** Cross-Region Replication allows you to create cross-region replications of your database for applications running in other regions.

12. A (Offload Analytics)

**Explanation:** With Offload Analytics, you can replicate data to the cloud and run analytics on your cloud databases rather than the original database users interact with.

13. D (All of them)

**Explanation:** We can use DMS for Migrations, Upgrade, Achieving

data, and Replications.

14. D (All of them)

**Explanation:** We can use a Snowball device to:

- Store 80 TB storage, 10 GB network.
- User interface similar to S3.
- All data is encrypted end-to-end.

15. C (Both A and B)

**Explanation:** Supported Migrations include heterogeneous and homogenous migrations.

## Chapter 06: Amazon Elastic Map Reduce

1. (A) True

**Explanation:** Map Reduce is a technique that data scientists can use to distribute workloads across many different computing nodes to process additional data and get the information back quicker than just on a single node.

2. (B) Hadoop Distributed File System

**Explanation:** Hadoop Distributed File System is open-source software that allows you to operate a distributed file system over several computers to tackle challenges requiring large amounts of data.

3. (C) Elastic Map Reduce

**Explanation:** Elastic Map Reduce is a fully managed AWS service that allows you to spin up Hadoop ecosystems.

4. (B) Primary Nodes

**Explanation:** Primary node tracks and directs the HDFS. The primary node knows how to lookup files and track data on the core nodes.

5. (A) Primary Nodes

**Explanation:** The primary node is responsible for the YARN resource management. EMR uses YARN (Yet another Resource Negotiator) to manage cluster resources for multiple data-processing frameworks.

6. (B) Core Nodes

**Explanation:** The primary node manages core nodes and runs Hadoop Map reduce tasks, Hive Scripts, and Spark executors.

#### 7. (C) Task Nodes

**Explanation:** Task nodes are optional and can add power to perform parallel computation tasks on data like Map reduce tasks and Spark executor.

#### 8. (A) Task Nodes

**Explanation:** Task nodes can be added and removed from the core nodes to ramp up extra CPU or memory for compute-intensive tasks.

#### 9. (A) True

**Explanation:** The EMR clusters only reside in a single availability zone. The main reason behind the single availability zone concept is that the cluster nodes can communicate faster. It means they do not have to traverse as much internet or the AWS backbone. They are closer together, and they are in the same availability zone.

#### 10. (A) Local File System (Instance Storage)

**Explanation:** Local File System (Instance Storage) is used for very high I/O performance and high IOPS at low cost. It is best used for temporary data (caches, buffers, and scratch data).

#### 11. (B) Local File System (EBS Volume)

**Explanation:** Local File System (EBS Volume) is used to add more storage for HDFS

#### 12. (C) Hadoop Distributed File System (HDFS)

**Explanation:** Hadoop Distributed File System (HDFS) is best used for caching the results produced by intermediate job-flow steps.

#### 13. (D) Elastic Map Reduce File System

**Explanation:** Elastic Map Reduce File System is best used for persistent store and S3 features that are needed, like server-side encryption and consistency.

14. (A) Transient Cluster

**Explanation:** If you set your cluster to terminate automatically, it will do so after completing all the steps. It is known as Transient Cluster.

15. (B) Long-Running Cluster

**Explanation:** If you set up the cluster to continue operating after processing is completed, the cluster is known as Long-Running Cluster.

## Chapter 07: Using Redshift

### 1. A (Redshift)

**Explanation:** Amazon Redshift is a fully managed petabyte-scale cloud data warehouse tool for storing and analyzing big data sets. Large-scale database migrations are also performed with it.

### 2. B (Redshift)

**Explanation:** Redshift is a data warehousing service. It can warehouse data at the petabyte scale, which means a huge amount of data can be stored in Redshift. It can also index and query that data so that data remains usable. We can store petabytes and exabytes of data in S3.

### 3. C (cluster)

**Explanation:** We have either leader nodes or worker nodes within the cluster.

### 4. A (Node)

**Explanation:** Node is the individual compute resources with storage attached for the Redshift cluster.

### 5. B (Postgres)

**Explanation:** In Redshift, Postgres can compress individual columns, which means different compression types are available depending on the data type.

### 6. C (Slice)

**Explanation:** A Slice divides the compute, memory, and storage of a

node into separate pieces so if recalled from the table in the node breakdown.

#### 7. A (Leader Node)

**Explanation:** The leader node manages the schema. It contains the data warehouse metadata and performs all the query planning and script generation.

#### 8. B (Worker Nodes)

**Explanation:** The worker nodes perform query execution, Slice management, and store all the data within the Slices.

#### 9. A (Cluster)

**Explanation:** Cluster - Organizational container for resources.

#### 10. B (Node)

**Explanation:** Node – similar to an instance in RDS.

#### 11. D (Query Process)

**Explanation:** Query Process – Query plan, Query, Execution Scripts.

#### 12. C (Slice)

**Explanation:** Slice – Logical subdivision of node resources.

#### 13. A (Cognito)

**Explanation** Cognito performs user authentication and management.

#### 14. C (Simple notification service)

**Explanation:** Simple notification service lets us send emails or text messages to end-users or the application administrator.

#### 15 D (DynamoDB)

**Explanation:** DynamoDB is used as a transactional database for our application.

## Chapter 08: Redshift Maintenance and Operations

1. D (All of them)

**Explanation:** Several interfaces for launching a Redshift cluster are Web console, AWS CLI, AWS SDKs.

2. B (Required Parameters)

**Explanation:** Required Parameters – Minimal parameters are required at launch.

3. C (Considerations)

**Explanation:** Considerations – Knowing the workload and use case helps define our cluster.

4. C (Both A and B)

**Explanation:** Vacuum can reclaim disk space maintain optimal query performance.

5. D (All of them)

**Explanation:** The vacuum process can reclaim disk space, sort Data, and reindex the table.

6. F (All of them)

**Explanation:** Vacuum options – Full, sort only, delete only, and reindex, to threshold percent, boost.

7. D (All of them)

**Explanation:** Automatic Vacuuming – Delete, sort, analyze.

8. A (Process to sort large amounts of unsorted data quickly)

**Explanation:** Deep Copy is a process to quickly sort large amounts of unsorted data.

9. D (All of them)

**Explanation:** Deep Copy Methods – Original table DDL, like table, temp table truncate.

10. A (Hours to days: only moves user objects)

**Explanation:** Classic Resize – Hours to days: only moves user objects.

11. B (Minutes: Migrate through the snapshot process)

**Explanation:** Elastic resize – Minutes: Migrate through the snapshot process.

12. C (Both A and B)

**Explanation:** Snapshot's Restoring – Creates a new cluster; cluster configuration can be modified.

13. A (copy command can copy data from S3 into the existing table)

**Explanation:** Loading Data from S3 – copy command can duplicate data from S3 into the current table.

14. B (unload command can export data from query to S3 in CSV or Parquet format)

**Explanation:** Unloading Data To S3 – unload command can export data from query to S3 in CSV or Parquet format.

15. C (Both A and B)

**Explanation:** CloudWatch provides a more granular view that can be used to combine metric graphs.

## Chapter 09: AWS Glue, Athena, and QuickSight

1. **A** (Fully managed ETL service to categorize, clean, and enrich your data.)

**Explanation:** AWS is a fully managed ETL service to categorize, clean, and enrich your data.

2. **D** (All of them)

**Explanation:** AWS Glue use cases include Query Data in S3, join data, and creating a centralized metadata catalog.

3. **F** (All of them)

**Explanation:** AWS Glue Components involve Source data stores, crawlers, catalogs, jobs, output data store, or services using the data catalog.

4. **D** (All of them)

**Explanation:** AWS Glue job performs the Extract, Transform, and Load (ETL) work in AWS Glue.

5. **A** (Compressing)

**Explanation:** The reason that JSON and CSV have an asterisk is that we have the option of compressing that data before it is stored off.

6. **B** (1-second)

**Explanation:** AWS Glue Version 2.0 is billed in 1-second increments with a 1-minute minimum.

7. **A** (1-second)

**Explanation:** AWS Glue Version 0.9 and 0.1 are billed in 1-second increments with a 10-minute minimum.

8. C (CloudWatch)

**Explanation:** You can use CloudWatch metrics to determine under or over-provisioned DPUs in the cluster by monitoring the total number of actively running executors, the number of completed stages, and the number of maximum needed executors.

9. C (AWS Glue Jobs)

**Explanation:** Glue jobs are the business logic that performs ETLs work in AWS Glue.

10. A (The units used for processing your Glue Jobs)

**Explanation:** Data Processing Units (DPUs) are used to process your Glue Jobs.

11. D (All of them)

**Explanation:** Glue Jobs run on virtual resources, glue jobs needs, and how traffic is governed.

12. A (S<sub>3</sub>)

**Explanation:** If you have data in sources other than S<sub>3</sub>, you can use Federated Query (beta) to query the data in place or build pipelines that extract data from multiple data sources and store them in Amazon S<sub>3</sub>

13. C (Both A and B)

**Explanation:** Athena natively supports querying datasets and data sources registered with the Glue Data Catalog.

14. A (Serverless querying tool to easily query data in S3)

**Explanation:** Athena is a serverless querying tool to query data in S3 easily.

15. D (All of them)

**Explanation:** Athena use cases involve Ad-hoc queries, joining data from multiple data sources, creating ETL pipelines, and transforming your data.

## Chapter 10: ElasticSearch

1. A (Amazon Elasticsearch service)

**Explanation:** The Amazon Elasticsearch service is a search domain that runs most of the ELK (Elasticsearch Logstash and Kibana) stack.

2. B (Index)

**Explanation:** Index is a Top-level organizational unit.

3. C (Type)

**Explanation:** Type is the second level used to categorize data.

4. B (Document)

**Explanation:** Document is Elasticsearch data object.

5. A (Elasticsearch)

**Explanation:** Anything that can interact with the API can send data into Elasticsearch.

6. C (CloudWatch)

**Explanation:** CloudWatch Logs subscription can deliver to Elasticsearch Service.

7. A (IoT)

**Explanation:** IoT rules can send data to Elasticsearch Service.

8. B (Elasticsearch)

**Explanation:** Elasticsearch uses JSON for everything.

9. F (All of them)

**Explanation:** The Interface – REST interface, GET, PUSH, DELETE, POST.

#### 10. B (IoT)

**Explanation:** IoT Has built-in ES integration.

## Chapter 11: AWS Security Services

### 1. (C) AWS IAM

**Explanation:** AWS Identity and Access Management (IAM) is a web service that provides secured control access to AWS resources such as compute, storage, database, and application services in the AWS Cloud. IAM manages authentication and authorization by controlling who is signed in and can utilize the resources. IAM uses access control concepts such as Users, Groups, Roles, and Policies to control which users can access specific services, the kind of actions they can perform, and which resources are available.

### 2. (A) IAM Users

**Explanation:** An IAM user is a unique identity with limited access to an AWS account and its resources, defined by their IAM permissions and policies. Users of IAM can represent a person, a system, or an application. IAM policies assigned to users must grant explicit permissions to services or resources before viewing or using them. IAM lets you create individual users within your AWS account and give them their username, password, and access keys.

### 3. (B) IAM Groups

**Explanation:** A group is a collection of IAM users. You can use

groups to specify permissions for a collection of users, making it easier to manage permissions for them. For example, you can have a group called Admins and give the permissions administrators typically need. Any user in that group has the permissions provided to the group by default.

#### 4. (C) IAM Roles

**Explanation:** An IAM role is an entity that lets you define a set of permissions to access the resources that a user or service needs, but the permissions are not attached to a specific IAM user or group. Instead, IAM users, mobile and EC2-based applications, or AWS services (such as Amazon EC2) can adopt a role programmatically.

#### 5. (D) IAM Policies

**Explanation:** An IAM policy is a rule or set of rules defining the operations allowed/denied to be performed on an AWS resource. Permissions are granted through policies. When attached to an identity or resource, a policy defines its permissions. When a user makes a request, AWS examines these rules.

#### 6. (B) AWS KMS

**Explanation:** AWS Key Management Service (KMS) is a managed service that allows you to create and manage cryptographic keys. The service offers a highly available key generation, storage, administration, and auditing solution that allows you to encrypt or digitally sign data within your applications or govern data encryption across AWS services.

#### 7. (B) Symmetric KMS Keys

**Explanation:** In AWS KMS, a symmetric KMS key is a 256-bit

encryption key that never leaves the system unencrypted. It would help if you used AWS KMS to utilize a symmetric KMS key. Symmetric keys are used in symmetric encryption, which employs the same key for encryption and decoding. Unless your task necessitates asymmetric encryption, symmetric KMS keys are a suitable choice since they never leave AWS KMS vulnerable.

#### 8. (D) Asymmetric KMS Keys

**Explanation:** AWS KMS allows you to generate asymmetric KMS keys. An asymmetric KMS key is a mathematically connected pair of public and private keys. The private key is never left unprotected in AWS KMS. It would help if you used AWS KMS to utilize the private key. The public key can be used within AWS KMS by executing the AWS KMS API activities or downloaded and used outside of AWS KMS. Multi-Region asymmetric KMS keys can also be generated.

#### 9. (B) Data Keys

**Explanation:** Data keys are encryption keys that may be used to encrypt data, especially enormous volumes of data. Data keys are returned to you for use outside of AWS KMS instead of KMS keys, which cannot be downloaded.

#### 10. (D) AWS Secrets Manager

**Explanation:** Secrets Manager allows you to replace hardcoded credentials, such as passwords, in your code with an API call to Secrets Manager to get the secret programmatically. Because the secret no longer resides in the code, this helps ensure that it cannot be compromised by someone reviewing your code. You may also set Secrets Manager to rotate the secret for you on a pre-defined period.

It allows you to substitute long-term secrets with short-term ones, drastically lowering the chance of compromise.

#### 11. (A) Rotation

**Explanation:** The process of updating a secret regularly is known as rotation. You may set up an automatic rotation for your secrets in Secrets Manager. When you rotate a secret, the credentials in both the secret and the database or service are updated. After rotation, applications that obtain the secret from Secrets Manager automatically receive the updated credentials.

#### 12. (C) JSON

**Explanation:** The AWS secrets manager uses JSON to distort data, approximating what it might look like in the figure below. It stores a string in the JSON. Hence those secrets can be passwords. They could be SSH keys. It could be API keys, really any string that fits within 64 kilobytes can be stored in Secrets Manager. You could have Base64 encoded strings that are stored in Secrets Manager that you know. You have to decode that Base64 encoding.

#### 13. (A) AWS VPC

**Explanation:** Amazon VPC lets you provision a logically isolated section of the AWS cloud to launch AWS resources in a virtual network that you define. You have complete control over your virtual networking environment, including the ability to define route tables and network gateways, as well as choose IP address ranges and construct subnets.

#### 14. (B) Network ACL

**Explanation:** A network access control list (ACL) is an optional

security layer for your VPC that operates as a firewall to manage traffic in and out of one or more subnets.

#### 15. (C) Traffic Monitoring

**Explanation:** Traffic Mirroring is an Amazon VPC feature that allows you to repeat network traffic from the elastic network interface of an Amazon EC2 instance. The traffic can then be routed to out-of-band security and monitoring equipment.

# ACRONYMS

|                          |                                             |
|--------------------------|---------------------------------------------|
| AAD                      | Additional Authenticated Data               |
| ACL                      | Access Control List                         |
| ACM PCA Authority        | AWS Certificate Manager Private Certificate |
| ACM Private CA Authority | AWS Certificate Manager Private Certificate |
| ACM                      | AWS Certificate Manager                     |
| AMI                      | Amazon Machine Image                        |
| ARN                      | Amazon Resource Name                        |
| ASN                      | Autonomous System Number                    |
| AUC                      | Area Under a Curve                          |
| AWS                      | Amazon Web Services                         |
| BGP                      | Border Gateway Protocol                     |
| CDN                      | Content Delivery Network                    |
| CIDR                     | Classless Inter-Domain Routing              |
| CLI                      | Command Line Interface                      |
| CMK                      | Customer Master Key                         |
| DB                       | Database                                    |
| DKIM                     | Domain Keys Identified Mail                 |

|                 |                                        |
|-----------------|----------------------------------------|
| DNS             | Domain Name System                     |
| EBS             | Elastic Block Store                    |
| EC <sub>2</sub> | Elastic Cloud Compute                  |
| ECR             | Elastic Container Registry             |
| ECS             | Elastic Container Service              |
| EFS             | Elastic File System                    |
| EMR             | Elastic Map Reduce                     |
| ES              | Elasticsearch Service                  |
| ETL             | Extract, Transform, and Load           |
| FBL             | Feedback Loop                          |
| FIM             | Federated Identity Management          |
| HMAC            | Hash-based Message Authentication Code |
| HPC             | High Performance Compute               |
| HSM             | Hardware Security Module               |
| IAM             | Identity and Access Management         |
| IdP             | Identity Provider                      |
| ISP             | Internet Service Provider              |
| JSON            | JavaScript Object Notation             |
| KMS             | Key Management Service                 |
| MFA             | Multi-factor Authentication            |
| MIME            | Multipurpose Internet Mail Extensions  |
| MITM            | Man in the Middle Attack               |

|      |                                |
|------|--------------------------------|
| MPLS | Multi Protocol Label Switching |
| MPP  | Massive Parallel Processing    |
| ML   | Machine Learning               |
| MTA  | Mail Transfer Agent            |
| OU   | Organizational Unit            |
| RDS  | Relational Database Service    |
| S3   | Simple Storage Service         |
| SCP  | Service Control Policy         |
| SDK  | Software Development Kit       |
| SES  | Simple Email Service           |
| SMTP | Simple Mail Transfer Protocol  |
| SNS  | Simple Notification Service    |
| SOAP | Simple Object Access Protocol  |
| SQS  | Simple Queue Service           |
| SSE  | Server-Side Encryption         |
| SSL  | Secure Sockets Layer           |
| SSO  | Single Sign-On                 |
| STS  | Security Token Service         |
| SWF  | Simple Workflow Service        |
| TLS  | Transport Layer Security       |
| VERP | Variable Envelope Return Path  |
| VPC  | Virtual Private Cloud          |

|      |                                   |
|------|-----------------------------------|
| VPG  | Virtual Private Gateway           |
| WAF  | Web Application Firewall          |
| WAM  | WorkSpaces Application Manager    |
| WSDL | Web Services Description Language |



# REFERENCES

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html>

[https://docs.aws.amazon.com/application-discovery/latest/userguide/data\\_collection.html](https://docs.aws.amazon.com/application-discovery/latest/userguide/data_collection.html)

<https://aws.amazon.com/blogs/big-data/harmonize-query-and-visualize-data-from-various-providers-using-aws-glue-amazon-athena-and-amazon-quicksight/>

<https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html>

<https://docs.aws.amazon.com/IAM/latest/UserGuide/intro-structure.html>

[https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction\\_access\\_management.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction_access_management.html)

<https://docs.aws.amazon.com/kms/latest/developerguide/overview.html>

<https://docs.aws.amazon.com/kms/latest/developerguide/overview.html>

<https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html>

[https://docs.aws.amazon.com/secretsmanager/latest/userguide/rotating\\_secrets.html](https://docs.aws.amazon.com/secretsmanager/latest/userguide/rotating_secrets.html)

<https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html>

<https://docs.aws.amazon.com/vpc/latest/userguide/how-it-works.html>

[https://www.google.com/aclk?sa=L&ai=DChcSEwjWh53AuKToAhUX-FEKHdTbDBQYABAAGgJ3cw&ae=2&sig=AOD64\\_1aQcZxBGMb-mQK8PJxDN7Ooh8ssw&q&adurl&ved=2ahUKEwiup5PAuKToAhWnSPF](https://www.google.com/aclk?sa=L&ai=DChcSEwjWh53AuKToAhUX-FEKHdTbDBQYABAAGgJ3cw&ae=2&sig=AOD64_1aQcZxBGMb-mQK8PJxDN7Ooh8ssw&q&adurl&ved=2ahUKEwiup5PAuKToAhWnSPF)

<https://aws.amazon.com/opensearch-service/the-elk-stack/what-is-elasticsearch/>

<https://docs.aws.amazon.com/opensearch-service/latest/developerguide/what-is.html>

<https://www.elastic.co/guide/en/elasticsearch/reference/current/getting>

[started.html](#)

<https://www.knowi.com/blog/what-is-elastic-search/>

<https://www.elastic.co/guide/en/kibana/6.8/tutorial-visualizing.html>

<https://www.freecodecamp.org/news/powerful-tools-for-elasticsearch-data-visualization-analysis/>

<https://qbox.io/blog/how-to-use-elasticsearch-to-visualize-data/>

<https://www.redhat.com/en/topics/api/what-does-an-api-gateway-do#:~:text=An%20API%20gateway%20is%20an,and%20return%20the%20>

<https://aws.amazon.com/api-gateway/>

<https://www.amazonaws.cn/en/cloudfront/>

[https://en.wikipedia.org/wiki/Amazon\\_CloudFront](https://en.wikipedia.org/wiki/Amazon_CloudFront)

<https://aws.amazon.com/blogs/big-data/use-amazon-athena-and-amazon-quicksight-in-a-cross-account-environment/>

<https://docs.aws.amazon.com/glue/latest/dg/populate-data-catalog.html>

<https://lakeformation.workshop.aws/glue-basics/glue-data-catalog.html>

<https://hevodata.com/learn/working-with-aws-glue-data-catalog/>

<https://docs.aws.amazon.com/glue/latest/dg/author-job.html>

<https://searchaws.techtarget.com/definition/AWS-Glue>

<https://docs.aws.amazon.com/glue/latest/dg/monitor-continuations.html>

<https://aws.amazon.com/premiumsupport/knowledge-center/glue-reprocess-data-job-bookmarks-enabled/>

<https://aprakash.wordpress.com/2020/05/07/implementing-glue-etl-job-with-job-bookmarks/>

<https://docs.aws.amazon.com/athena/latest/ug/getting-started.html>

<https://www.sqlshack.com/getting-started-with-amazon-athena-and-s3/>

<https://docs.aws.amazon.com/quicksight/latest/user/working-with-dashboards.html>

<https://www.bakertilly.com/insights/visualizing-your-data-with-amazon-quicksight>

<https://acloudguru.com/blog/engineering/amazon-quicksight-how-to-put-eyes-on-your-data-with-this-aws-bi-tool>

<https://www.bakertilly.com/insights/visualizing-your-data-with-amazon-quicksight>

[https://github.com/awsdocs/amazon-quicksight-user-guide/blob/master/doc\\_source/embedded-dashboards-security.md](https://github.com/awsdocs/amazon-quicksight-user-guide/blob/master/doc_source/embedded-dashboards-security.md)

<https://aws.amazon.com/big-data/datalakes-and-analytics/>

<https://docs.snowplowanalytics.com/docs/getting-started-on-snowplow-open-source/setup-snowplow-on-aws/setup-destinations/setup-redshift/launch-a-redshift-cluster/#:~:text=Go%20into%20the%20Amazon%20webservices,%2C%2>

<https://aws.amazon.com/premiumsupport/knowledge-center/resize-redshift-cluster/>

<https://www.flydata.com/blog/how-to-create-an-amazon-redshift-cluster/>

<https://www.intermix.io/blog/elastic-node-resizing-in-redshift/>

<https://docs.aws.amazon.com/redshift/latest/dg/performing-a-deep-copy.html>

<https://discourse.snowplowanalytics.com/t/delete-and-vacuum-vs-deep-copy/812>

<https://hevodata.com/blog/redshift-vacuum-and-analyze/>

<https://www.google.com/aclk?sa=L&ai=DChcSEwixm8bFtPzzAhUW7eoKHQ8uDZUYABAAGgJkZw&a>

<https://docs.aws.amazon.com/aws-backup/latest/devguide/restore-resource.html>

<https://docs.aws.amazon.com/redshift/latest/mgmt/metrics.html>

<https://github.com/aws-labs/amazon-redshift-monitoring>

<https://www.sumologic.com/blog/monitoring-amazon-redshift/>

<https://aws.amazon.com/redshift/>

[https://docs.aws.amazon.com/redshift/latest/dg/c\\_high\\_level\\_system\\_a](https://docs.aws.amazon.com/redshift/latest/dg/c_high_level_system_a)

<https://hevodata.com/blog/redshift-architecture/>

<https://towardsdatascience.com/amazon-redshift-architecture-b674513eb996>

<https://www.intermix.io/blog/amazon-redshift-architecture/>

<https://docs.aws.amazon.com/AmazonECS/latest/developerguide/creat>

<https://docs.aws.amazon.com/AmazonECS/latest/developerguide/cluste>

<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/getting-started-nodejs.html>

<https://www.google.com/aclk?>

[sa=L&ai=DChcSEwjpmqXJjvDzAhVYhdUKHUilAmoYABAAGgJ3cw&ae=EUC-hraMTrQ&q&adurl&ved=2ahUKEwidoJvJjvDzAhXNifoHHVq-DRkQoQx6BAgCEAE](https://www.google.com/aclk?sa=L&ai=DChcSEwjpmqXJjvDzAhVYhdUKHUilAmoYABAAGgJ3cw&ae=EUC-hraMTrQ&q&adurl&ved=2ahUKEwidoJvJjvDzAhXNifoHHVq-DRkQoQx6BAgCEAE)

<https://docs.aws.amazon.com/lumberyard/latest/userguide/component-slices-creating.html>

<https://aws.amazon.com/redshift/features/>

<https://sarasanalytics.com/blog/pros-and-cons-of-amazon-redshift>

<https://www.intermix.io/blog/amazon-redshift-use-cases/>

<https://www.trustradius.com/products/redshift/reviews?qs=product-usage>

<https://searchaws.techtarget.com/definition/Amazon-Redshift-Spectrum#:~:text=Amazon%20Redshift%20Spectrum%20is%20a,stored>

<https://blog.openbridge.com/how-is-aws-redshift-spectrum-different-than-aws-athena-9baa2566034b>

<https://stackshare.io/stackups/aws-direct-connect-vs-aws-snowball->

[edge](#)

<https://www.slideshare.net/AmazonWebServices/data-migration-using-aws-snowball-snowball-edge-snowmobile>

<https://aws.amazon.com/dms/>

<https://cloud.google.com/database-migration>

<https://www.stitchdata.com/resources/what-is-data-pipeline/#:~:text=A%20data%20pipeline%20is%20a,that%20provide%20c>

<https://www.qlik.com/us/data-integration/data-pipeline>

<https://docs.aws.amazon.com/solutions/latest/constructs/aws-cloudfront-apigateway-lambda.html>

<https://serverlessland.com/patterns/cloudfront-s3-lambda-cdk>

<https://aws.amazon.com/lambda/>

<https://www.redhat.com/en/topics/api/what-does-an-api-gateway-do#:~:text=An%20API%20gateway%20is%20an,and%20return%20the%2>

<https://aws.amazon.com/api-gateway/>

<https://www.amazonaws.cn/en/cloudfront/>

[https://en.wikipedia.org/wiki/Amazon\\_CloudFront](https://en.wikipedia.org/wiki/Amazon_CloudFront)

<https://awscli.amazonaws.com/v2/documentation/api/latest/reference/>

<https://awscli.amazonaws.com/v2/documentation/api/latest/reference/>

<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/s>

# ABOUT OUR PRODUCTS

Other products from IPSpecialist LTD regarding CSP technology are:



AWS Certified Cloud Practitioner Study guide



AWS Certified SysOps Admin - Associate Study guide



AWS Certified Solution Architect - Associate Study guide



AWS Certified Developer Associate Study guide



AWS Certified Advanced Networking – Specialty Study guide



AWS Certified Security – Specialty Study guide



AWS Certified Big Data – Specialty Study guide



AWS Certified Machine Learning – Specialty Study guide



Microsoft Certified: Azure Fundamentals



Microsoft Certified: Azure Administrator



Microsoft Certified: Azure Solution Architect



Microsoft Certified: Azure DevOps Engineer



Microsoft Certified: Azure Developer Associate



Microsoft Certified: Azure Security Engineer



Microsoft Certified: Azure Data Fundamentals



Microsoft Certified: Azure AI Fundamentals



Microsoft Certified: Azure Database Administrator Associate



Google Certified: Associate Cloud Engineer



Google Certified: Professional Cloud Developer



Microsoft Certified: Azure Data Engineer Associate



Microsoft Certified: Azure Data Scientist



Oracle Certified: OCI Foundations Associate



Oracle Certified: OCI Developer Associate



Oracle Certified: OCI Architect Associate