AWS PROJECT
DOCUMENTATION

BLACKBUCKS INTERNSHIP REPORT

Transferring CSV File from CodeCommit to S3 and Creating a Table in AWS Glue using Crawler

SUBMITTED BY

Mr. NARAYANA DEEPAK **RegdNo:20B91A05K6**

Mr. SALUMURI DURGA GOWTHAM **RegdNo:20B91A05Q9**

Mr. KUCHERLAPATI PUNEETH SAI RAMA RAJU **RegdNo:20B91A05F4**

UNDER THE GUIDANCE OF

MR. AASHU DEV

B. Tech, AWS solution Architect (2x) certified,

AWS Academy Accredited Educator

Blackbuck Engineers Pvt.

Ltd Road No 36, Jubilee Hills, Hyderabad

Table of Contents

I. INTRODUCTION

AWS CLOUD COMPUTE:

Cloud, in the context of computing, refers to the practice of using remote servers hosted on the internet to store, manage, and process data, as well as provide various services. It involves accessing and utilizing resources and applications over the internet rather than relying solely on local infrastructure or hardware.

Cloud computing is a model of computing that involves the delivery of various computing services over the internet. Instead of relying on a local server or personal computer to store data and run applications, cloud computing utilizes remote servers hosted on the internet to store and process data, and to provide various services.

NEED OF CLOUD COMPUTING:

Cloud computing offers several advantages compared to physical storage in ancient times. Here are a few key reasons why cloud computing is preferred over traditional physical storage methods:

- Scalability: Cloud storage offers easy and dynamic scalability, while physical storage had limited capacity.
- Cost Efficiency: Cloud computing operates on a pay-as-you-go model, eliminating upfront investments and enabling cost optimization.
- Accessibility and Availability: Cloud storage provides ubiquitous access from anywhere with an internet connection, ensuring high availability even during emergencies.
- Data Preservation and Durability: Cloud storage employs advanced preservation techniques, ensuring data durability and long-term preservation.
- Collaboration and Sharing: Cloud computing enables seamless collaboration and sharing of data among individuals and teams across different locations.
- Security: Cloud storage offers robust security measures to protect data from theft, damage, or unauthorized access.
- Flexibility and Innovation: Cloud computing provides a wide range of services beyond storage, enabling organizations to leverage advanced technologies and innovate without significant infrastructure investments.

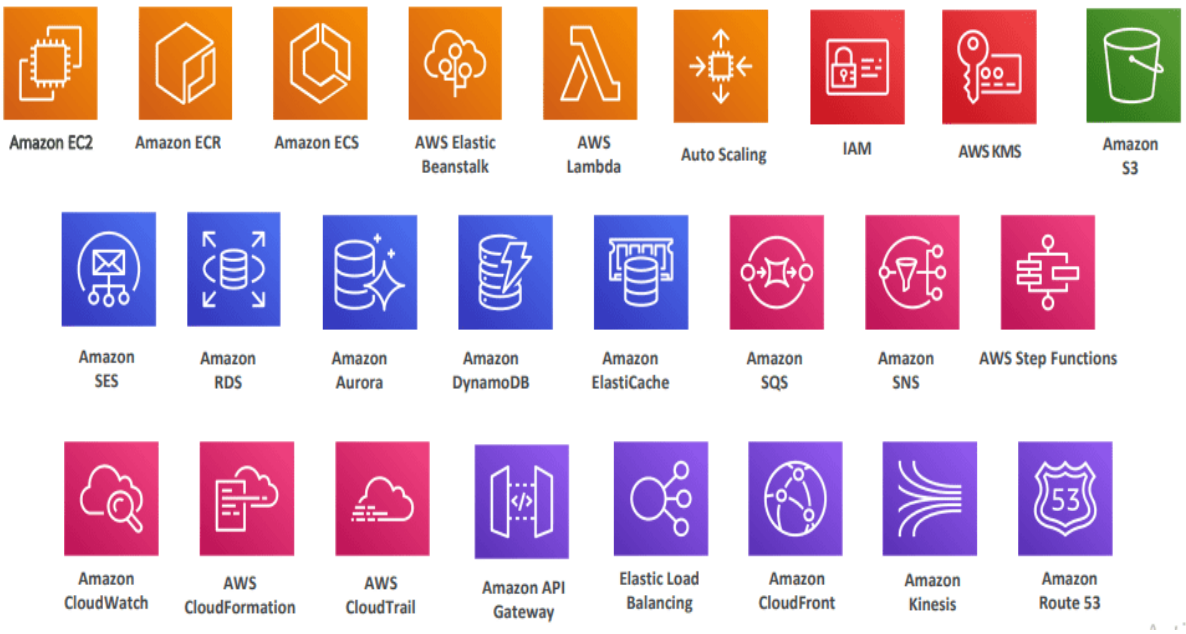
AMAZON WEB SERVICES (AWS):

AWS, or Amazon Web Services, is a comprehensive cloud computing platform offered by Amazon.com. It provides a wide range of cloud services that enable businesses and individuals to build and deploy various applications and services in a flexible and scalable manner.

AWS offers a vast array of services across different categories, including:

- Computing:
  AWS Elastic Compute Cloud (EC2) provides virtual servers in the cloud, allowing users to run applications and workloads. It offers a wide selection of instance types with different compute, memory, and storage capabilities.

- **Storage**:
  AWS provides multiple storage options, including Amazon Simple Storage Service (**S3**) for object storage, Amazon Elastic Block Store (EBS) for persistent block storage, and Amazon Glacier for long-term archival storage. These services offer durability, scalability, and high availability for storing and retrieving data.
- **Databases**:
  AWS offers various database services, such as Amazon Relational Database Service (RDS) for managed relational databases, Amazon DynamoDB for NoSQL databases, and Amazon Aurora for a MySQL and PostgreSQL-compatible database. These services handle database management tasks, such as backups, patching, and replication.
- **Networking**:
  AWS provides networking services that enable users to establish secure and reliable connections. Amazon Virtual Private Cloud (VPC) allows users to create isolated virtual networks, and AWS Direct Connect offers dedicated network connections between on-premises environments and AWS.
- **Security and Identity**:
  AWS offers a range of security services, including AWS Identity and Access Management (IAM) for managing access to AWS resources, AWS Key Management Service (KMS) for encryption key management, and AWS Certificate Manager for managing SSL/TLS certificates.
- **Analytics**:
  AWS provides services for data analytics and processing, such as Amazon Redshift for data warehousing, Amazon Athena for interactive query analysis, and Amazon Kinesis for real-time data streaming and processing.
- **Artificial Intelligence (AI) and Machine Learning (ML):**
  AWS offers services like Amazon SageMaker for building, training, and deploying machine learning models, Amazon Recognition for image and video analysis, and Amazon Lex for building conversational interfaces.



Why AWS?

The purpose of AWS (Amazon Web Services) is to provide a comprehensive and scalable cloud computing platform to individuals, businesses, and organizations. AWS offers a wide range of cloud services that enable users to access computing power, storage, and other resources on-demand, without the need for upfront infrastructure investment or long-term commitments.

Some of the key purposes and benefits of AWS include:

- Scalability and Flexibility: AWS allows users to scale their computing resources up or down based on their needs, providing the ability to handle fluctuations in demand and avoid overprovisioning or underutilization of resources.
- Cost-Effectiveness: With AWS, users only pay for the services and resources they use, eliminating the need for large upfront investments in hardware or software. It offers a pay-as-you-go pricing model, enabling cost optimization and efficient resource allocation.
- Reliability and Availability: AWS operates from a global infrastructure of data centers, ensuring high availability and fault tolerance. It provides built-in redundancy, data replication, and backup mechanisms, minimizing the risk of data loss or service disruptions.
- Security: AWS offers a robust and secure cloud environment. It implements a wide array of security measures, including encryption, access controls, and compliance with various industry standards, to protect customer data and applications.
- Broad Service Portfolio: AWS provides a vast array of cloud services across various domains, such as compute, storage, databases, networking, machine learning, analytics, IoT, and more. This comprehensive service catalog caters to diverse requirements and enables users to build, deploy, and manage applications and infrastructure effectively.
- Global Reach: AWS has a global presence with data centers located in multiple regions around the world. This allows users to deploy their applications closer to their target audience, reducing latency and enhancing performance.
- Ease of Use: AWS offers intuitive management consoles, extensive documentation, and developer-friendly tools, making it easier for users to get started and manage their cloud resources effectively. It also provides APIs and SDKs for seamless integration with existing systems and automation of tasks.
- Managed Services offers a range of managed services, such as Amazon RDS for managing databases, Amazon Elastic Beanstalk for application deployment, and Amazon ECS for container orchestration. These services handle infrastructure management tasks, allowing you to focus more on your applications and business logic.
- Ecosystem and Integration: AWS has a robust ecosystem with a wide range of partners and third-party integrations. This ecosystem includes technology partners, consulting partners, managed service providers, and a vast marketplace of pre-built solutions and services. It facilitates easy integration with other systems and tools, enabling users to leverage existing investments and extend their capabilities within the AWS environment.

Overall, the purpose of AWS is to empower businesses and individuals with the tools and infrastructure necessary to leverage the benefits of cloud computing, including scalability, cost savings, reliability, security, and innovation.

II. ARCHITECTURE

Title:

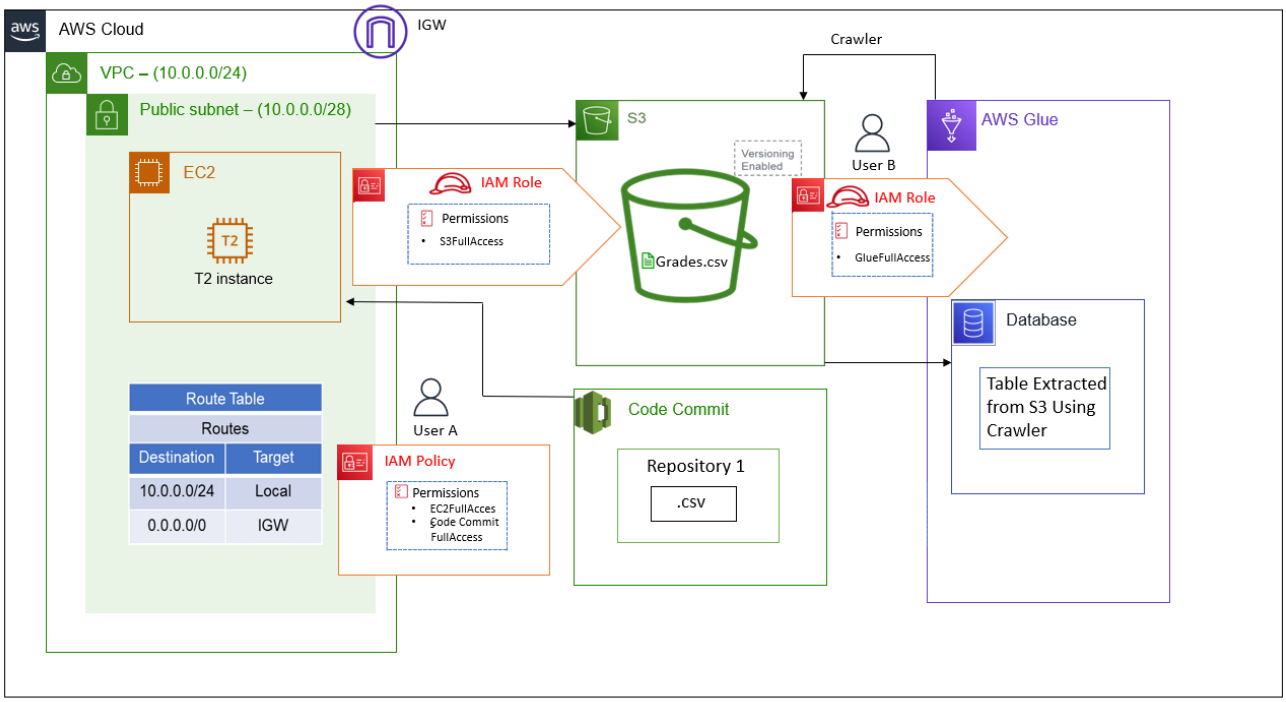
Transferring CSV File from CodeCommit to S3 and Creating a Table in AWS Glue using Crawler

Description:

This project involves transferring a CSV file from a CodeCommit repository to an S3 bucket using EC2 Instance Connect. Subsequently, a table is created in an AWS Glue database by running a crawler that extracts the CSV file's schema from the S3 bucket. This workflow facilitates seamless data integration and analysis within AWS services, leveraging EC2, CodeCommit, S3, and AWS Glue.

Architecture:

The architecture we used to develop this project is as follows.



We use the following services in our project

VPC	AWS Glue
EC2	Code Commit
S3	Database

Workflow:

- UserA Setup:

  - UserA is created and granted full access to EC2, EC2 Instance Connect, and CodeCommit.

  - An EC2 instance is created for UserA with an IAM role that has S3FullAccess.

- UserB Setup:

  - UserB is created and granted full access to the AWS Glue console.

- CSV File Transfer from CodeCommit to S3 using EC2 Instance Connect:

  a. UserA authenticates and accesses the AWS Management Console.

  b. UserA navigates to the CodeCommit service and creates a new repository

  c. UserA creates the CSV file within the repository.

  d. UserA establishes a secure connection to the EC2 instance using EC2 Instance Connect.

  e. UserA transfers the CSV file from the CodeCommit repository to the EC2 instance

  f. UserA confirms the successful transfer of the CSV file to the EC2 instance.

- Upload CSV File to S3:

  a. UserA accesses the AWS Management Console and navigates to the S3 service.

  b. UserA creates an S3 bucket to store the CSV file.

  c. UserA uploads the CSV file using EC2 Instance Connect to S3 Bucket

  d. UserA verifies the successful upload of the CSV file to the designated S3 bucket.

- Table Creation in AWS Glue Database using Crawler:

  a. UserB logs into the AWS Management Console.

  b. UserB navigates to the AWS Glue service and selects the Glue Crawler option.

  c. UserB configures a new crawler and specifies the S3 bucket location that contains the uploaded CSV File

  d. UserB defines the output database and table settings for the crawler.

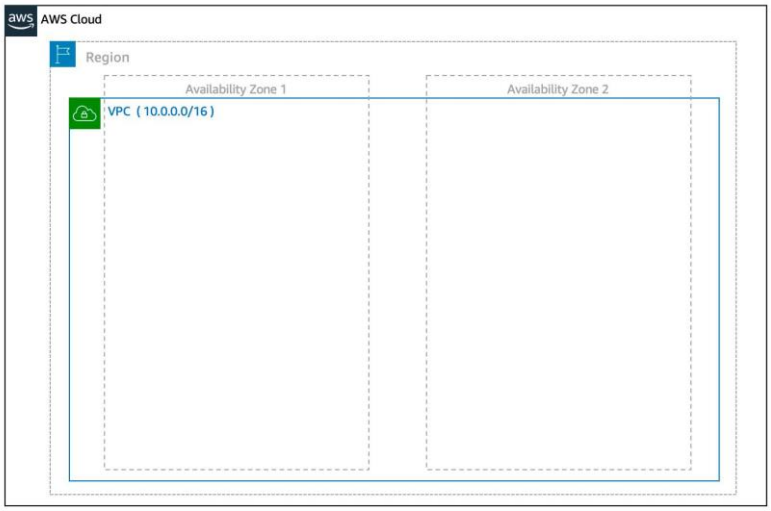
  e. UserB initiates the crawler, which scans the CSV file in the specified S3 bucket, extracts the schema, and creates the table in the AWS Glue database.

  f. UserB monitors the crawler's progress and verifies the successful completion of the table creation process.

This workflow enables seamless data integration and analysis within AWS services.

III. SERVICES INCLUDED IN ARCHITECTURE

Amazon VPC (Virtual Private Cloud):



Amazon Virtual Private Cloud (VPC) is a crucial component of Amazon Web Services (AWS) that allows users to create a logically isolated section within the AWS cloud. VPC enables users to have complete control over their virtual networking environment, including the selection of IP address ranges, subnets, and configuration of route tables and network gateways. With VPC, users can seamlessly launch AWS resources such as EC2 instances, RDS databases, and Elastic Load Balancers in a virtual network of their own. VPC provides advanced security features like network access control lists (ACLs) and security groups to protect resources from unauthorized access.

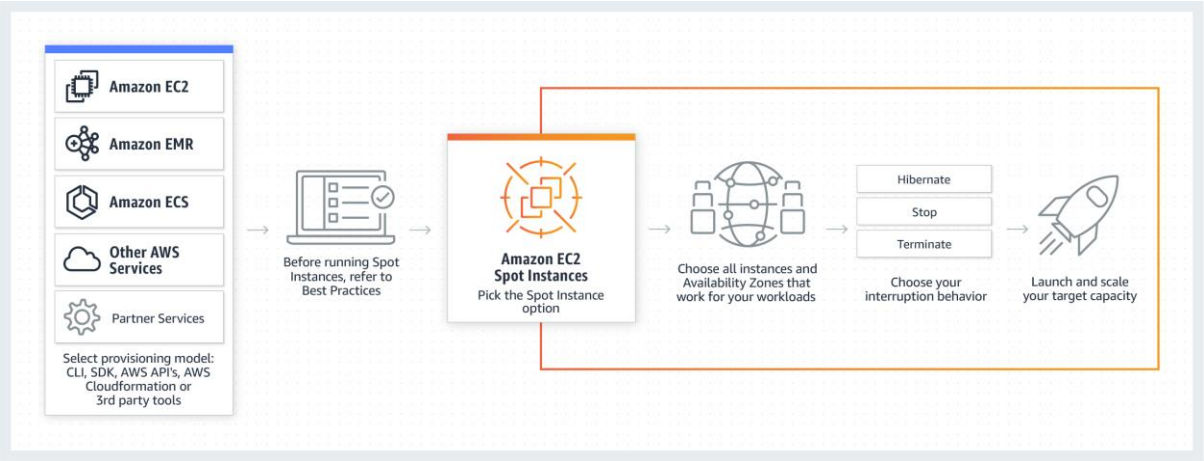
- Subnets: Subnets are subdivisions of a VPC's IP address range. They allow you to partition the VPC into smaller networks for better organization and control. Subnets are associated with availability zones within the cloud provider's data center, enabling high availability and fault tolerance.
- Route Tables: Route tables define the routing rules for network traffic within the VPC. They specify how traffic should be directed between subnets, the internet, and other network resources. Route tables can be used to control access to and from the VPC and implement network security policies.
- Internet Gateway: An internet gateway is a horizontally scalable and highly available service that allows communication between instances within the VPC and the internet. It acts as a bridge, enabling outbound and inbound internet connectivity for resources within the VPC.
- Network Access Control Lists (NACLs): NACLs are stateless firewalls that control inbound and outbound traffic at the subnet level. They allow you to define rules that permit or deny specific types of traffic based on protocols, ports, and IP addresses. NACLs provide an additional layer of security for your VPC.
- Security Groups: Security groups are stateful firewalls that control inbound and outbound traffic at the instance level. They act as virtual firewalls for individual instances or groups of instances,

allowing you to define rules that dictate which traffic is allowed to reach the instances based on protocols, ports, and IP addresses.

- Network Address Translation (NAT) Gateway: A NAT gateway allows instances within private subnets to communicate with the internet while keeping them hidden from inbound traffic originating from the internet. It provides a managed solution for instances that require internet access but do not have a public IP address assigned to them.
- Virtual Private Network (VPN) Connections: VPN connections establish secure and encrypted connections between your on-premises network and the VPC. They allow you to extend your existing network infrastructure into the cloud, providing secure access to resources within the VPC.
- Peering Connections: Peering connections allow you to connect multiple VPCs within the same cloud provider's infrastructure. Peering enables private and secure communication between VPCs without the need to go through the public internet.

These are some of the core components of a Virtual Private Cloud. Different cloud providers may have additional features and services that enhance VPC functionality, but the components listed above form the foundation of most VPC implementations.

Amazon EC2(Elastic Computing):



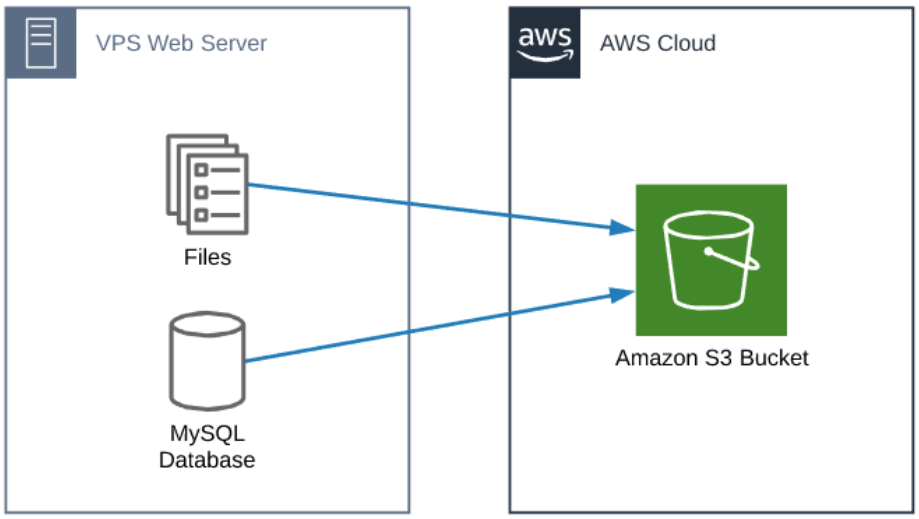
Amazon Elastic Compute Cloud (Amazon EC2) is a web service provided by Amazon Web Services (AWS) that offers scalable computing capacity in the cloud. It allows you to provision virtual servers, known as instances, and run your applications on them. EC2 instances can be easily scaled up or down based on your requirements, providing flexibility and cost optimization.

Amazon EC2 consists of several key components that work together to provide the infrastructure for running virtual server instances. Here are the main components of EC2:

- Instances: EC2 instances are virtual servers that run within the EC2 environment. They can be provisioned with different combinations of CPU, memory, storage, and networking capacity, known as instance types. Instances can be launched from pre-configured Amazon Machine Images (AMIs) or custom AMIs created by users.

- Amazon Machine Images (AMIs): An AMI is a template that contains the necessary operating system, software, and configurations required to launch an instance. It serves as the basis for creating new instances. Amazon provides a wide range of pre-configured AMIs, including different operating systems and software stacks, or you can create your own custom AMIs.
- Key Pairs: EC2 instances use key pairs for secure access. A key pair consists of a public key that is placed on the instance during launch and a private key that is retained by the user. The private key is used to authenticate and securely connect to the instance via Secure Shell (SSH) or Remote Desktop Protocol (RDP).
- Security Groups: Security groups act as virtual firewalls for EC2 instances. They control inbound and outbound traffic by allowing or denying specific protocols, ports, and IP ranges. Multiple instances can be associated with a security group, and you can define rules to permit or restrict traffic between instances or from external sources.
- Elastic IP Addresses: An Elastic IP address is a static IPv4 address that can be assigned to an EC2 instance. It provides a fixed public IP address that remains associated with the instance, even if it is stopped or restarted. Elastic IP addresses are useful when you need a consistent IP address for your instance, such as for hosting a website or setting up network connections.
- Elastic Block Store (EBS): EBS provides persistent block-level storage volumes for EC2 instances. These volumes can be attached to instances as virtual hard drives, and they retain data even if the instance is terminated. EBS volumes offer different types (e.g., SSD-backed or HDD-backed) and can be managed independently from instances, allowing for data durability and availability.
- Virtual Private Cloud (VPC): A VPC is a logically isolated virtual network within the AWS cloud where EC2 instances reside. It allows you to define your own IP address range, subnets, route tables, and network gateways. VPCs provide control over network security, connectivity, and traffic routing for your EC2 instances.
- Load Balancers: EC2 provides load balancing services, such as the Elastic Load Balancer (ELB), that distribute incoming traffic across multiple EC2 instances. Load balancers help improve the availability and scalability of applications by spreading the load and automatically adjusting traffic based on configurable rules.

Amazon S3(Simple Storage Service):



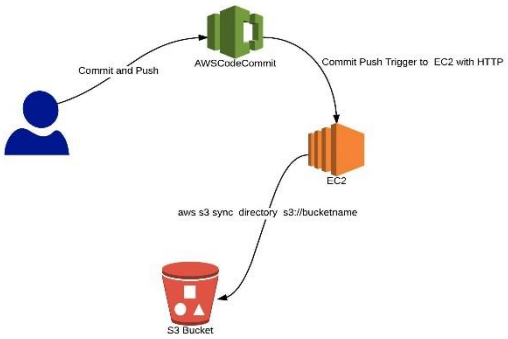
Amazon Simple Storage Service (S3) is a highly scalable and secure object storage service provided by Amazon Web Services (AWS). It allows businesses to store and retrieve vast amounts of data over the internet. S3 provides durability, availability, and performance to store and retrieve any amount of data from anywhere at any time.S3 offers a simple and intuitive web interface, as well as a set of APIs, enabling seamless integration with various applications and services. With S3, users can create "buckets" to store objects such as images, videos, documents, and backups. These objects are organized into a flat structure, and each object is assigned a unique key. S3 offers a simple and intuitive web interface, as well as a set of APIs, enabling seamless integration with various applications and services. With S3, users can create "buckets" to store objects such as images, videos, documents, and backups.

Overall, Amazon S3 is a versatile and reliable storage solution, trusted by millions of organizations worldwide, providing secure, scalable, and cost-effective storage for a wide range of applications and use cases.

Amazon S3 is a scalable cloud storage. The main components of Amazon S3 are:

- Buckets: An S3 bucket is a container for storing objects. You can think of it as a top-level folder that holds your data. Each object in S3 is stored in a bucket. You can create multiple buckets to organize your data and control access to them.
- Objects: Objects are the fundamental entities stored in Amazon S3. They consist of data (such as files) and associated metadata. An object can range in size from 0 bytes to 5 terabytes, and you can store an unlimited number of objects in a bucket.
- Keys: Keys are unique identifiers for objects stored in S3. They are similar to file paths or names and are used to organize and retrieve objects within a bucket. For example, if you have an object named "example.jpg" in a bucket called "mybucket," the key for that object would be "mybucket/example.jpg."
- Regions: Amazon S3 is available in different geographical regions around the world. Each region represents a separate geographic area where S3 resources are located. You can choose the region where you want to create your buckets based on factors like data residency requirements, latency, and compliance regulations.
- Access Control Lists (ACLs): ACLs are used to manage access to your S3 buckets and objects. They define who can perform operations (such as read, write, or delete) on your resources. ACLs can be applied at both the bucket level and the object level, allowing you to control access on a granular basis.
- Bucket Policies: Bucket policies are JSON-based access control policies that are attached to S3 buckets. They provide a way to define fine-grained permissions for your buckets and objects. With bucket policies, you can specify rules that grant or deny access based on various conditions, such as IP addresses, user agents, or time of day.
- Lifecycle Policies: Lifecycle policies allow you to automate the management of your objects in S3. You can define rules that automatically transition objects between different storage classes (e.g., from Standard to Glacier) or delete objects after a certain period of time. Lifecycle policies help optimize storage costs and data lifecycle management.
- Storage Classes: Amazon S3 offers different storage classes that provide varying levels of durability, availability, and cost. The available storage classes include Standard, Intelligent-Tiering, Standard-IA (Infrequent Access), One Zone-IA, Glacier, and Glacier Deep Archive. You can choose the appropriate storage class based on your data access patterns and costs.

AWS Code Commit:

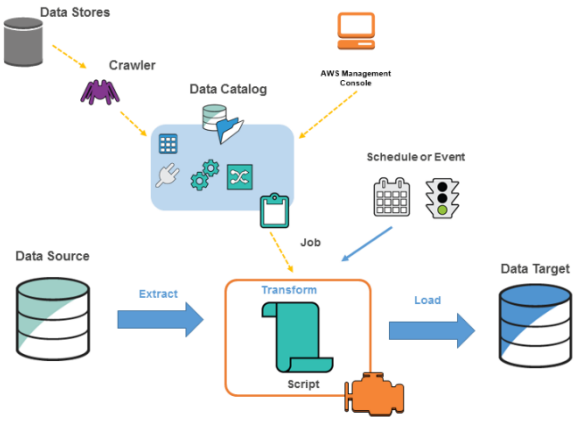


When it comes to code commit components, there are several key elements involved in the process of committing code changes to a version control system. AWS CodeCommit is a fully managed source control service that allows you to securely host private Git repositories in the cloud. It provides a scalable and highly available solution for storing and managing your source code, enabling collaborative development and version control. These components typically include:

- Version Control System (VCS): A VCS is a software tool that manages the storage and versioning of code changes. Popular VCSs include Git, Mercurial, and Subversion. The VCS is responsible for tracking changes, merging code, and maintaining a history of commits.
- Repository: A repository is a central location where the codebase and its version history are stored. It is a container for all project files and directories. The repository can be hosted locally or on a remote server, such as GitHub, Bitbucket, or GitLab.
- Working Directory: The working directory is a local copy of the repository where developers make changes to the code. It contains the most up-to-date version of the codebase, along with any modifications that have not been committed yet.
- Staging Area (Index): The staging area, also known as the index, acts as a buffer between the working directory and the repository. Developers use the staging area to select which changes they want to include in the next commit. Files added to the staging area are prepared for the commit but haven't been permanently saved to the repository yet.
- Commit: A commit is a record of a set of changes made to the codebase. It represents a logical unit of work and is accompanied by a commit message that describes the changes. Commits are permanent and create a new version in the repository's history.
- Branches: Branches are independent lines of development within a repository. They allow developers to work on different features or bug fixes concurrently without affecting the main codebase. When committing code, developers can choose to commit to a specific branch or merge changes from one branch to another.
- Hooks: Hooks are scripts that are triggered at specific points in the commit process. They allow developers to perform custom actions, such as running tests, formatting code, or sending notifications, before or after a commit.

These components work together to facilitate code collaboration, version tracking, and the ability to roll back changes when needed. They provide a structured and organized approach to managing code commits and ensuring the integrity of the codebase.

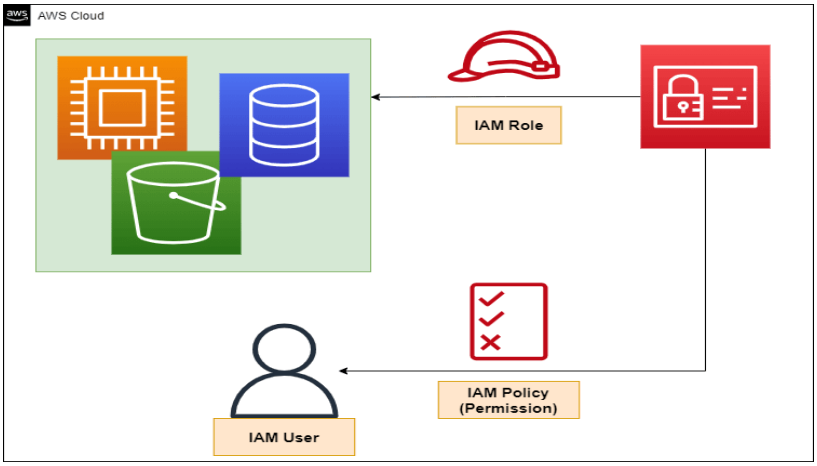
AWS GLUE:



AWS Glue is a fully managed extract, transform, and load (ETL) service provided by Amazon Web Services. It simplifies the process of preparing and loading data from various sources into data lakes, data warehouses, and analytics platforms. AWS Glue consists of several components that work together to enable data integration and transformation tasks. Here are the key components of AWS Glue:

- AWS Glue Data CatLog: This component acts as the central metadata repository for AWS Glue. It stores metadata about data sources, such as databases, tables, schemas, and transformations. The Data CatLog provides a unified view of the available data and allows users to define and manage schemas and partitions.
- AWS Glue Crawler: The Crawler automatically discovers and catalogs data sources in the AWS Glue Data Catalog. It scans various data stores, such as Amazon S3, Amazon RDS, Amazon Redshift, and more, to infer schemas and create table definitions in the Data Catalog. The Crawler can schedule regular scans to keep the catalog up to date.
- AWS Glue Job: A Glue Job is an ETL script or workflow that defines the transformation logic for data processing. You can create and schedule Glue Jobs to extract data from various sources, transform it using code or visual tools, and load it into target destinations. Glue Jobs can be written in Python or Scala, and AWS Glue provides a serverless runtime environment to execute the jobs.
- AWS Glue Development Endpoint: A Development Endpoint allows you to interactively develop, debug, and test your Glue ETL scripts. It provides an environment where you can write and execute code using tools like Jupyter notebooks or integrated development environments (IDEs). Development Endpoints offer a way to iterate and refine your ETL logic before deploying it as a Glue Job.
- AWS Glue DataBrew: DataBrew is a visual data preparation tool that complements AWS Glue. It provides a visual interface for data wrangling and transformation tasks. With DataBrew, you can explore and profile data, clean and normalize it, apply transformations, and create recipes for repeatable data preparation steps. The recipes can then be executed using AWS Glue Jobs.
- AWS Glue Studio: Glue Studio is a visual ETL authoring tool that helps you build ETL workflows without writing code. It provides a drag-and-drop interface to create data flows, apply transformations, and define data sources and targets. Glue Studio simplifies the process of designing and orchestrating complex ETL pipelines by abstracting the underlying code.

IAM ROLE (Identity and Access Management:



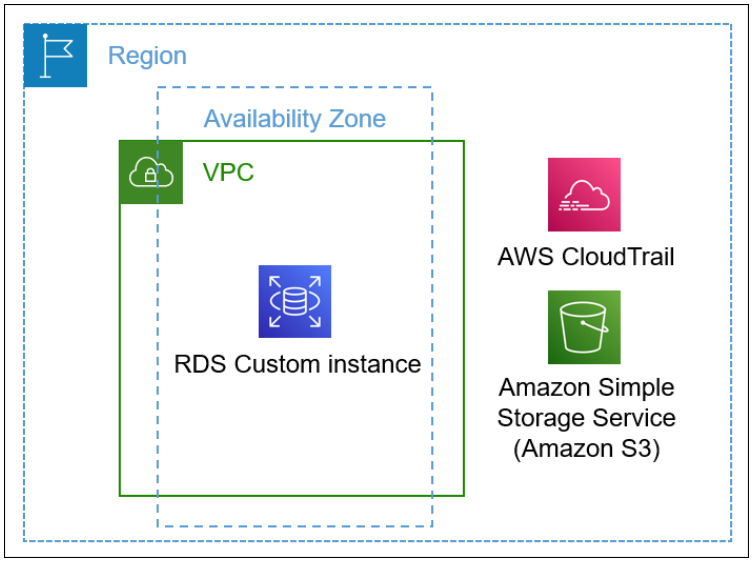
In AWS (Amazon Web Services), IAM stands for Identity and Access Management. It is a service that helps you manage access to AWS resources securely. IAM consists of various components that work together to control user access, permissions, and policies. The main components of IAM in AWS include:

- Users: Users represent individual entities (people or applications) who interact with AWS resources. Each user is assigned a unique username and security credentials, including access keys, password, or IAM roles.
- Groups: Groups are collections of IAM users. Instead of defining permissions for individual users, you can assign permissions to groups, making it easier to manage access for multiple users. Users within a group inherit the permissions assigned to that group.
- Roles: IAM roles are similar to users, but they are not tied to a specific identity. Roles are created and associated with AWS services or resources, and they can be assumed by users, applications, or AWS services. Roles are often used to grant permissions to services or to establish trust relationships between different AWS accounts.
- Policies: IAM policies are JSON documents that define permissions. They specify what actions are allowed or denied on which AWS resources. Policies can be attached to users, groups, or roles, granting or restricting access to specific resources and services.
- Permissions: Permissions define what actions an IAM entity (user, group, or role) can perform on AWS resources. These actions can include managing resources, reading or modifying data, or interacting with AWS services.
- Access Keys: Access keys consist of an access key ID and a secret access key. They are used to programmatically authenticate and access AWS resources through APIs, command-line tools, or SDKs. Access keys are typically associated with IAM users.
- Multi-Factor Authentication (MFA): MFA adds an extra layer of security to user sign-ins. It requires users to provide an additional authentication factor, such as a time-based one-time password (TOTP) generated by a virtual or hardware MFA device.

These components work together to provide fine-grained control over access to AWS resources, allowing you to manage identities, assign permissions, and enforce security policies within your AWS environment.

OTHER AWS SERVICES:

AWS RDS:



AWS RDS (Relational Database Service) is a managed database service provided by Amazon Web Services. It simplifies the deployment, management, and scaling of relational databases. The key components of AWS RDS are as follows:

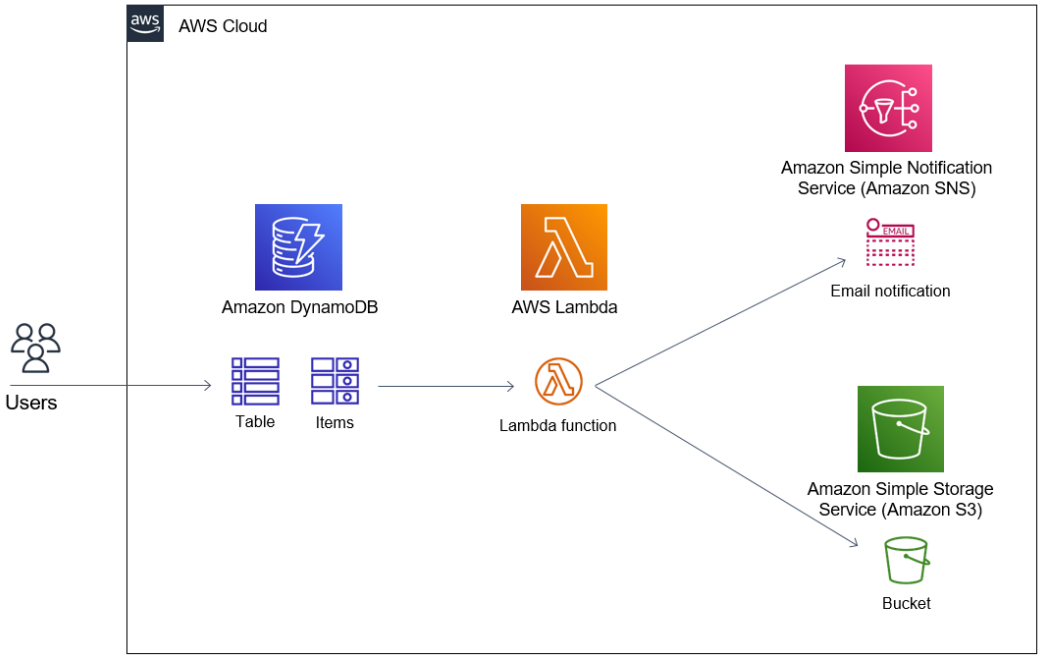
- Database Engines: AWS RDS supports various popular relational database engines, including Amazon Aurora (MySQL and PostgreSQL-compatible), MySQL, MariaDB, PostgreSQL, Oracle Database, and Microsoft SQL Server. Each engine has its own set of features, compatibility, and performance characteristics.
- DB Instances: A DB Instance is a running database environment within AWS RDS. It represents a single deployment of a database engine and associated resources. Users can select the desired database engine, instance type, storage capacity, and other configurations when creating a DB Instance.
- Parameter Groups: Parameter Groups allow users to configure database engine settings for their DB Instances. They provide fine-grained control over database configuration parameters, such as memory allocation, caching, replication, timeouts, and many others. Parameter Groups can be shared across multiple DB Instances to maintain consistency in settings.
- DB Snapshots: DB Snapshots are point-in-time backups of DB Instances. They capture the entire state of a database at a specific moment, including data, schema, settings, and log files. DB Snapshots are stored in Amazon S3 and can be used to restore a database or create new DB Instances.
- Multi-AZ Deployment: AWS RDS provides a Multi-AZ (Availability Zone) deployment option for high availability and automatic failover. In Multi-AZ mode, a standby replica of the primary database is created in a different Availability Zone, ensuring redundancy and minimizing downtime in case of a failure.
- Read Replicas: Read Replicas allow users to create one or more read-only copies of their DB Instances. They can be used to offload read traffic from the primary database, improve read

scalability, and reduce the impact on the primary instance. Read Replicas can be located in the same region or different regions for disaster recovery purposes.

- Backups and Restore: AWS RDS automates backups for DB Instances, capturing regular snapshots of the database. Users can also manually initiate backups as needed. In addition to DB Snapshots, RDS supports automated backups, which enable point-in-time recovery within a specified retention period.
- Security: AWS RDS offers various security features to protect databases, including network isolation using Amazon VPC, encryption at rest using AWS Key Management Service (KMS), and encryption in transit using SSL/TLS certificates. It also provides fine-grained access control through IAM roles, database users, and security groups.
- Monitoring and Metrics: AWS RDS integrates with Amazon CloudWatch, allowing users to monitor the performance of their DB Instances. CloudWatch provides metrics, logs, and alarms for monitoring CPU utilization, storage, I/O activity, and other database-related metrics. These insights help in optimizing performance and detecting any issues.
- Parameter Store Integration: AWS RDS integrates with AWS Systems Manager Parameter Store, which allows users to securely store and manage database credentials, configuration settings, and other sensitive information. This integration simplifies the management of secrets and enables secure access to databases.

These components work together to provide a managed and scalable relational database service in AWS RDS. Users can leverage these features to easily provision, manage, and scale their databases without the need for manual administration and infrastructure management.

AWS DynamoDB:

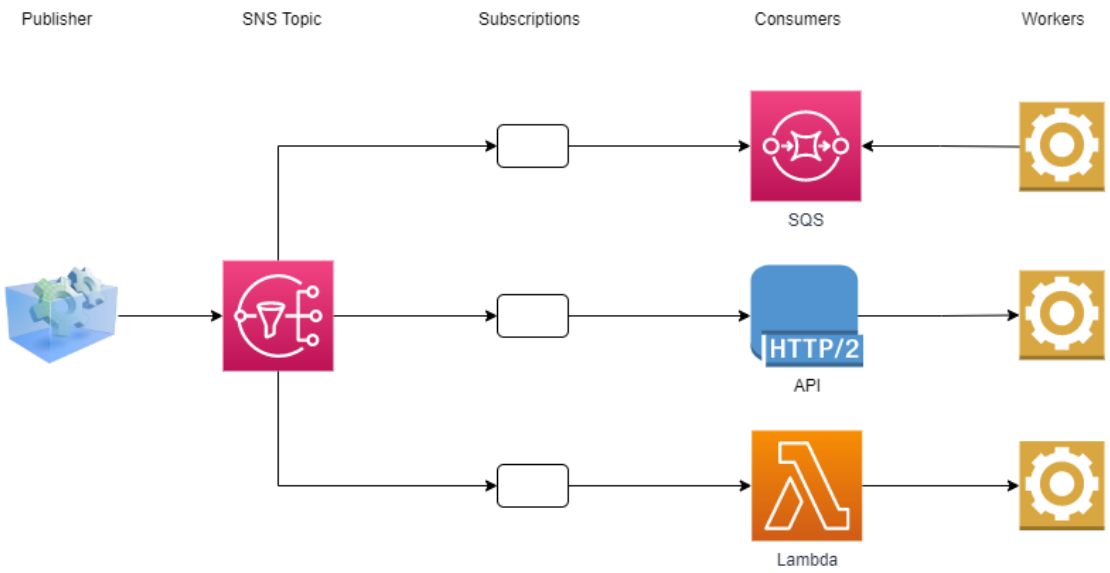


AWS DynamoDB is a fully managed NoSQL database service provided by Amazon Web Services. It is designed to provide fast and scalable storage for applications that require low-latency and high-performance data access. The key components of AWS DynamoDB are as follows:

- Tables: A DynamoDB table is a collection of items, similar to a table in a traditional relational database. Tables in DynamoDB have a flexible schema, allowing each item within a table to have a different set of attributes. Tables are distributed across multiple servers to provide scalability and high availability.
- Items: An item is a single data record within a DynamoDB table. Each item is uniquely identified by a primary key, which can be either a single attribute (simple primary key) or a combination of partition key and sort key (composite primary key). Items can have multiple attributes, including strings, numbers, Booleans, binary data, nested JSON objects, or even arrays.
- Attributes: Attributes are the key-value pairs that make up the data within DynamoDB items. Each item can have multiple attributes, and the attributes do not need to be predefined in a schema. DynamoDB is schema-less, allowing flexible addition or modification of attributes within items.
- Partitions and Partition Keys: DynamoDB uses partitioning to distribute the data across multiple servers for scalability. The partition key is used to determine the partition in which an item is stored. The partition key should have a high cardinality to evenly distribute the data across partitions and achieve optimal performance.
- Sort Keys: In tables with a composite primary key, the sort key is used to determine the order of items with the same partition key. It enables efficient querying and sorting of items based on different criteria.
- Read and Write Capacity Units: DynamoDB provisions read and write capacity units to define the throughput for a table. Read capacity units determine the maximum number of strongly consistent or eventually consistent reads per second, while write capacity units define the maximum number of writes per second.
- Global Tables: Global Tables provide automatic and synchronous replication of DynamoDB tables across multiple regions for global data distribution and disaster recovery. It enables users to achieve low-latency access to data from any region and provides built-in resilience against regional failures.
- Streams: DynamoDB Streams is a feature that captures the changes made to the items in a DynamoDB table. It provides a time-ordered sequence of item-level modifications, such as inserts, updates, and deletes. Streams can be used to trigger actions in real-time, such as updating caches, maintaining search indexes, or triggering downstream processes.
- Indexes: DynamoDB supports two types of indexes: Global Secondary Indexes (GSIs) and Local Secondary Indexes (LSIs). GSIs allow efficient querying of data based on non-primary key attributes, while LSIs enable querying within a partition key's range of a table. Indexes improve query performance by creating additional data structures that are optimized for specific access patterns.
- On-Demand Capacity Mode: DynamoDB offers an On-Demand Capacity Mode, where users do not need to explicitly provision read and write capacity units. Instead, the service automatically scales to handle the traffic and charges based on actual usage. This mode simplifies capacity management and is well-suited for workloads with unpredictable or bursty traffic patterns.

These components work together to provide a highly scalable, fully managed NoSQL database service in AWS DynamoDB. Developers can leverage these features to store, retrieve, and manage large amounts of data with low latency and high availability.

AMAZON SNS:



AWS SNS (Simple Notification Service) is a fully managed messaging service provided by Amazon Web Services (AWS). It enables developers to send notifications to various endpoints, such as email, SMS, mobile push notifications, and more. SNS simplifies the process of sending messages to multiple recipients or subscribers, allowing for easy communication and notification delivery.

Key features and benefits of AWS SNS include:

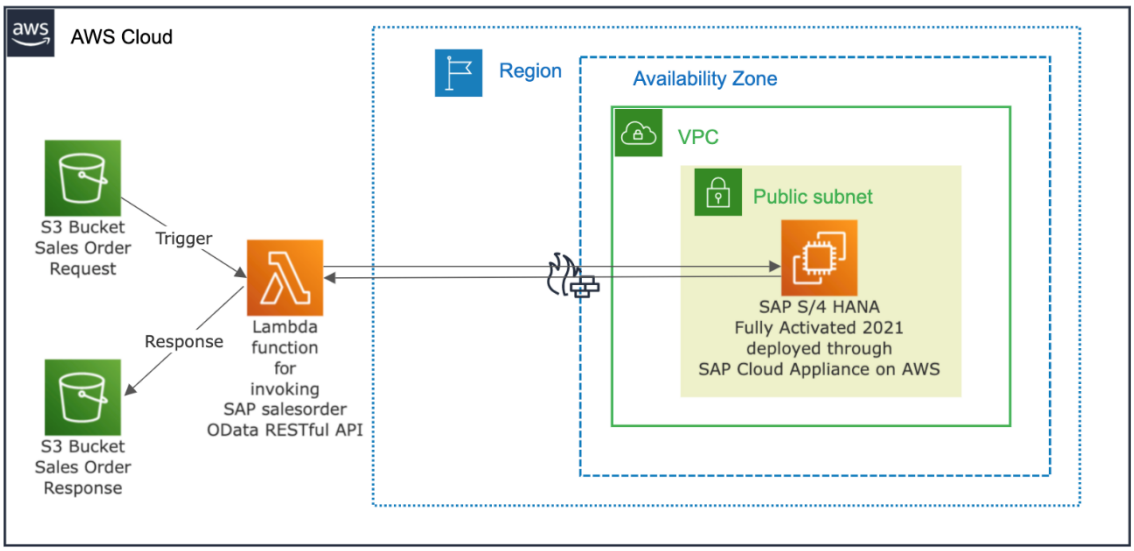
- Publish-Subscribe Model: SNS follows a publish-subscribe paradigm. Publishers send messages to topics, and subscribers who are interested in those topics receive the messages. This decoupled architecture allows for easy scaling and flexibility in message distribution.
- Multi-Channel Messaging: SNS supports multiple delivery protocols, including email, SMS, mobile push notifications (for iOS, Android, etc.), HTTP/HTTPS endpoints, AWS Lambda functions, and more. This versatility enables developers to choose the most appropriate channel for message delivery based on the target audience and their preferences.
- Highly Available and Scalable: SNS is designed to be highly available and scalable. It operates on a distributed infrastructure with redundant components, ensuring message delivery reliability. SNS can handle large message volumes and scale automatically to accommodate increasing traffic.
- Flexible Message Filtering: SNS allows for message filtering based on attributes or conditions. Subscribers can specify filter policies to receive only the messages that match their criteria. This filtering capability helps reduce unnecessary message distribution and allows subscribers to receive targeted and relevant notifications.
- Message Attributes: SNS supports the inclusion of custom metadata or attributes with messages. These attributes can provide additional context or instructions to subscribers or downstream systems, enhancing the usefulness and flexibility of the messages.
- Fanout and Fanout Patterns: SNS supports the fanout pattern, where a single message can be sent to multiple subscribers simultaneously. This simplifies the process of broadcasting

notifications to multiple endpoints or subscribers without the need for complex message routing or distribution logic.

- Integration with Other AWS Services: SNS integrates well with other AWS services. For example, it can be used in conjunction with AWS Lambda to trigger serverless functions based on incoming messages. It can also integrate with Amazon SQS (Simple Queue Service) to create reliable and decoupled message queues.
- Monitoring and Management: AWS provides monitoring and management capabilities for SNS through Amazon CloudWatch. CloudWatch allows you to collect and analyze metrics, set alarms, and gain insights into the performance and health of your SNS topics and messages.

AWS SNS is commonly used for various use cases, such as sending application notifications, broadcasting updates to subscribers, triggering automated workflows, and distributing system alerts. It simplifies the process of sending messages across different communication channels, making it an efficient solution for building notification systems and real-time messaging applications.

AMAZON LAMBDA:



AWS Lambda is a serverless compute service provided by Amazon Web Services (AWS) that allows developers to run code without provisioning or managing servers. It enables users to focus on writing and deploying their application logic without the need to worry about infrastructure management. AWSLambda automatically scales and manages the underlying compute resources based on the incoming request volume, ensuring efficient and cost-effective execution of code.

Key features and benefits of AWS Lambda include:

- Serverless Architecture: Lambda enables you to execute your code in a serverless environment, meaning you don't have to worry about server provisioning, capacity planning, or infrastructure management. AWS takes care of the server infrastructure, scaling, and availability, allowing you to focus on writing code and delivering value.

- Event-Driven Execution**: Lambda functions are triggered in response to events from various AWS services or custom-defined events. This event-driven architecture allows you to build applications that respond to changes in data, system events, user actions, or scheduled tasks.
- Multiple Language Support**: Lambda supports a variety of programming languages, including Python, Node.js, Java, C#, Ruby, Go, and PowerShell. This language flexibility allows developers to choose the language they are most comfortable with or that best fits their application requirements.
- Automatic Scaling**: Lambda automatically scales your functions based on incoming request rates. It dynamically provisions and manages the necessary resources to handle concurrent invocations, ensuring that your code can handle spikes in traffic without manual intervention.
- Pay-per-Use Pricing**: AWS Lambda follows a pay-per-use pricing model. You are only charged for the compute time consumed by your functions and the number of requests processed. You don't have to pay for idle resources, making it a cost-effective option for applications with variable or unpredictable workloads.
- Integration with AWS Services**: Lambda seamlessly integrates with various AWS services, allowing you to build serverless applications that leverage the full capabilities of the AWS ecosystem. For example, you can use Lambda with Amazon S3 for processing file uploads, with Amazon DynamoDB for real-time data processing, or with Amazon API Gateway for building RESTful APIs.
- Event Sources and Triggers**: Lambda functions can be triggered by a wide range of event sources, including changes in data stored in AWS services, incoming HTTP requests, scheduled events (using AWS CloudWatch Events), messaging services (such as Amazon SNS and Amazon SQS), and more. This enables you to build reactive and event-driven architectures.
- Scalable Backend Processing**: Lambda is commonly used for backend processing tasks, such as data transformation, file processing, image resizing, log analysis, and ETL (Extract, Transform, Load) operations. It provides a flexible and scalable platform for running these types of workloads without the need for managing infrastructure.

AWS Lambda offers a serverless approach to building and running applications, providing flexibility, scalability, and cost efficiency. It empowers developers to focus on writing code and delivering value while leaving the operational aspects to AWS.

- Lambda Functions:
  In AWS Lambda, code is organized and executed in the form of functions. A Lambda function isa piece of code that performs a specific task or implements a particular functionality. It can be written in various supported programming languages, such as Python, Node.js, Java, C#, and Go. Each Lambda function can be triggered by specific events or invoked directly via API Gateway or other AWS services.

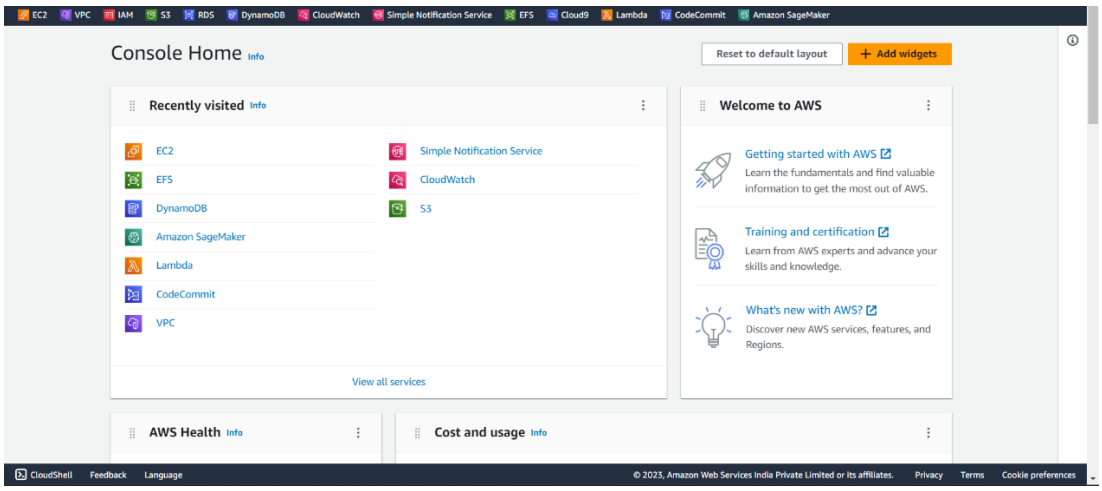
- Lambda Triggers:
  Lambda functions can be triggered by different types of events, known as Lambda triggers. Triggers define what causes the execution of a Lambda function. AWS offers a wide range of triggers that can be used to invoke Lambda functions based on various events within the AWSecosystem or external services.

IV IMPLEMENTATION

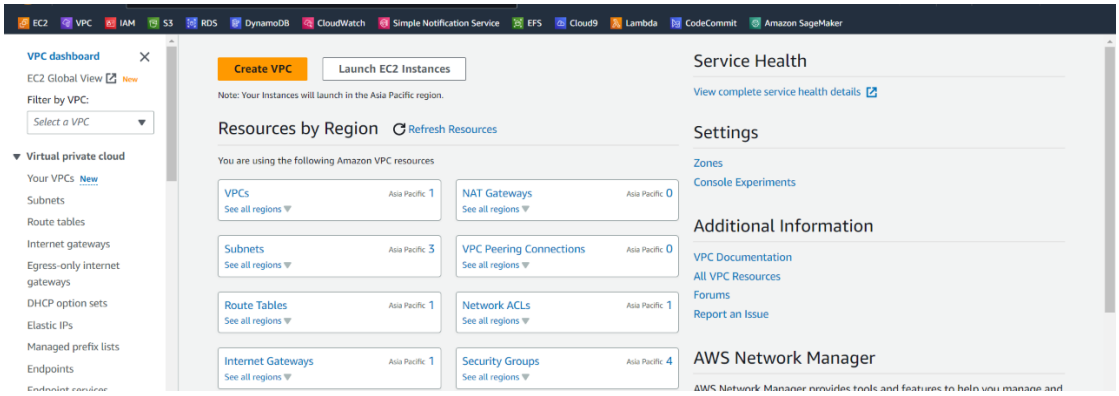
The implementation of the architecture is as follows:

Step:01 Custom VPC Creation

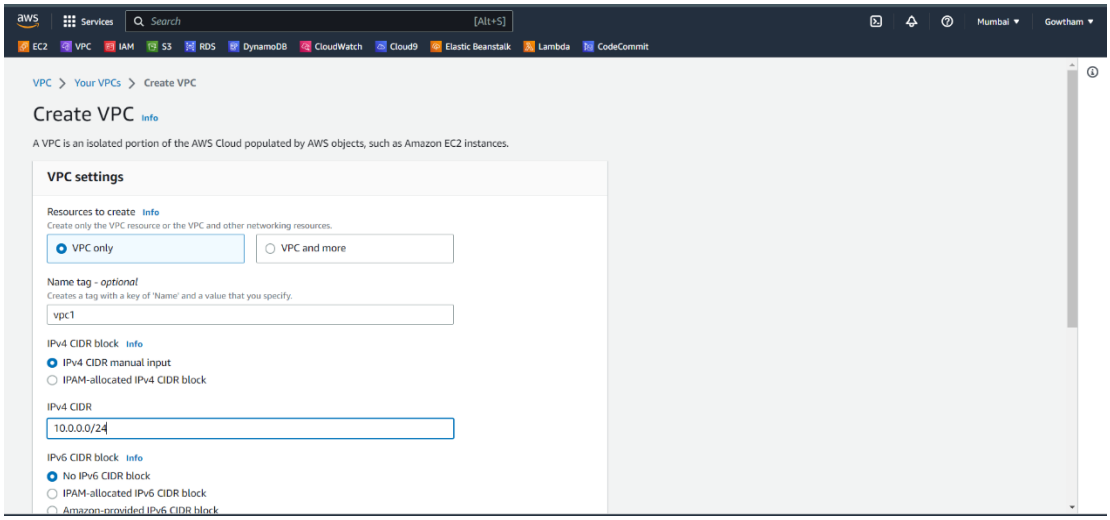
Open the AWS Management Console and Sign-In to your account.



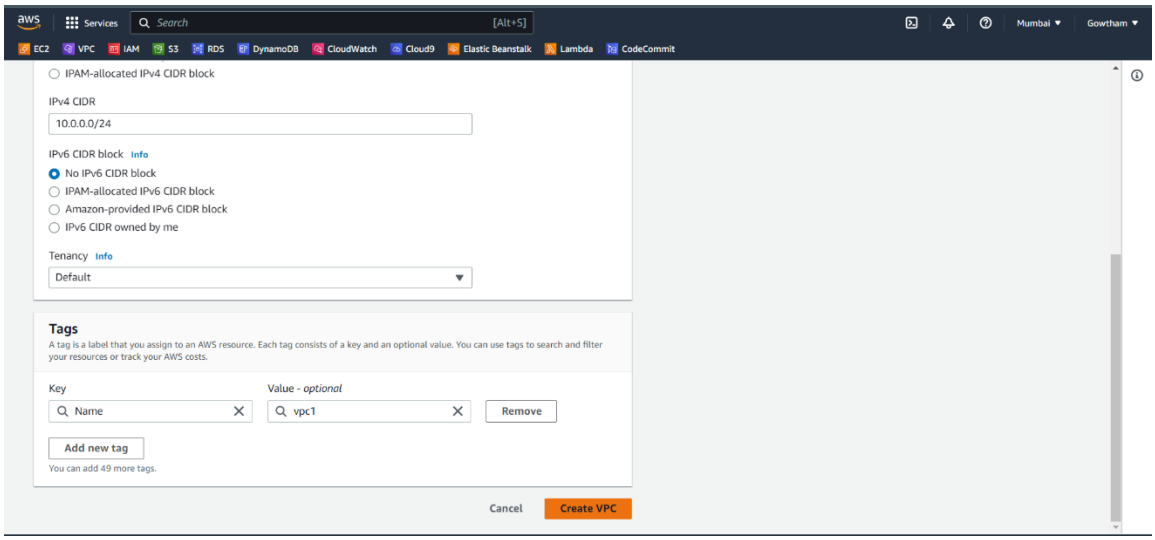
Next Go to VPC and click on create VPC.



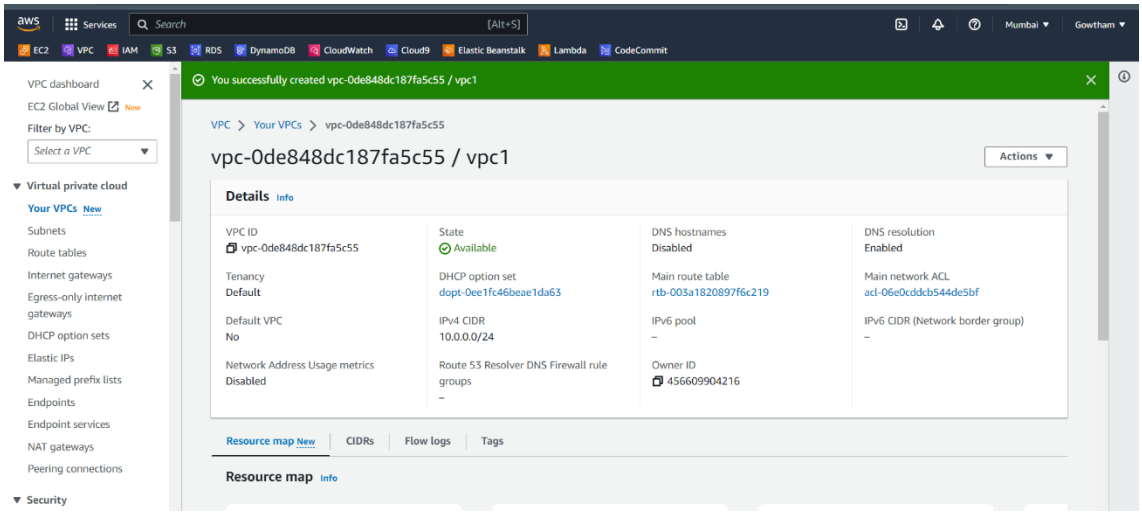
Go to VPC Settings-> Select VPC only, name the custom VPC and give IPV4 CIDR (10.0.0.0/24).



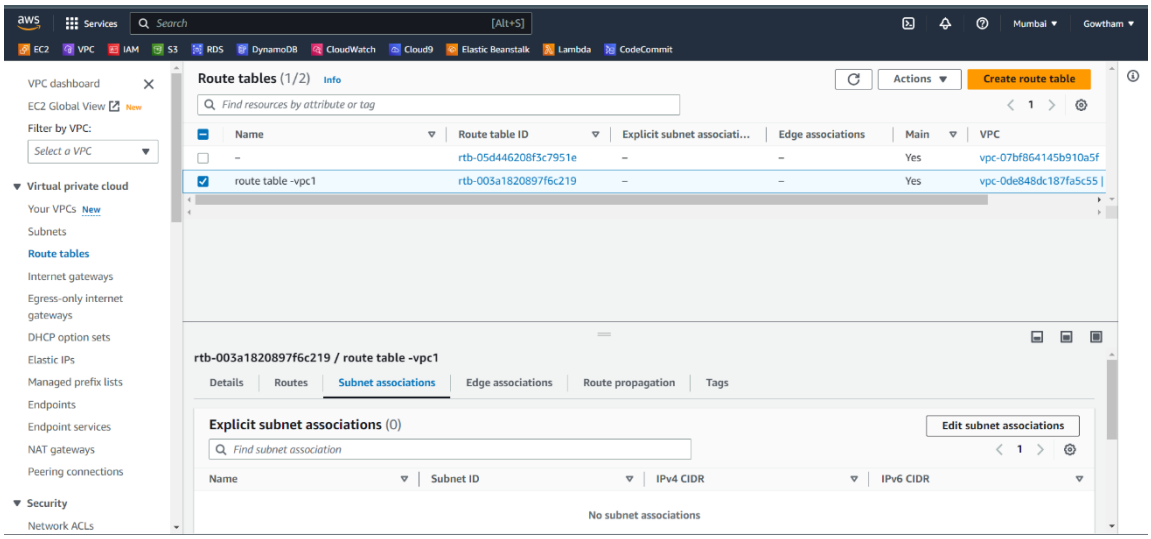
Now Create the custom VPC.



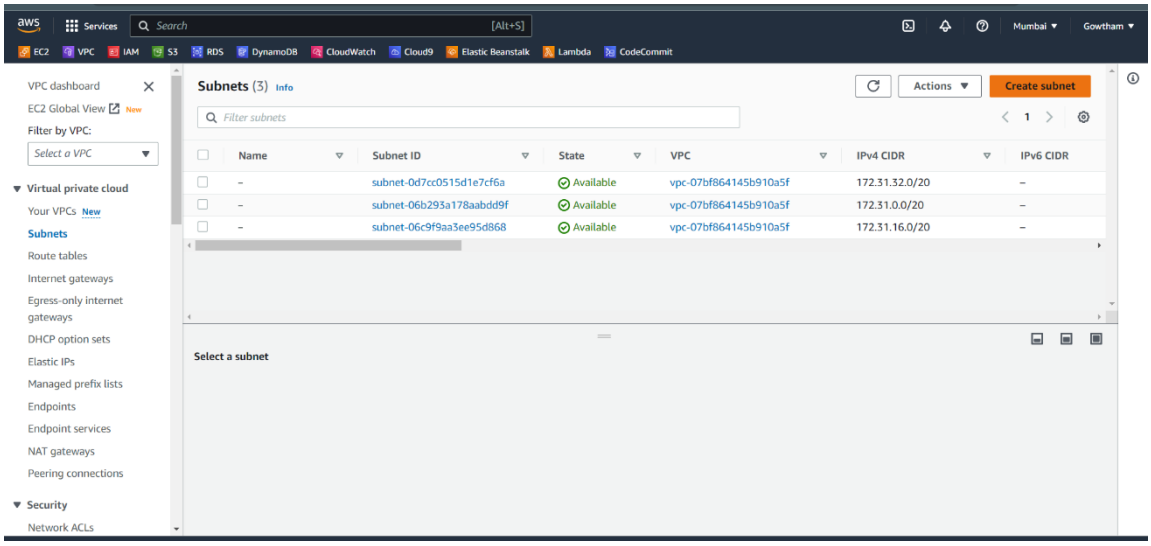
Successfully created custom VPC.



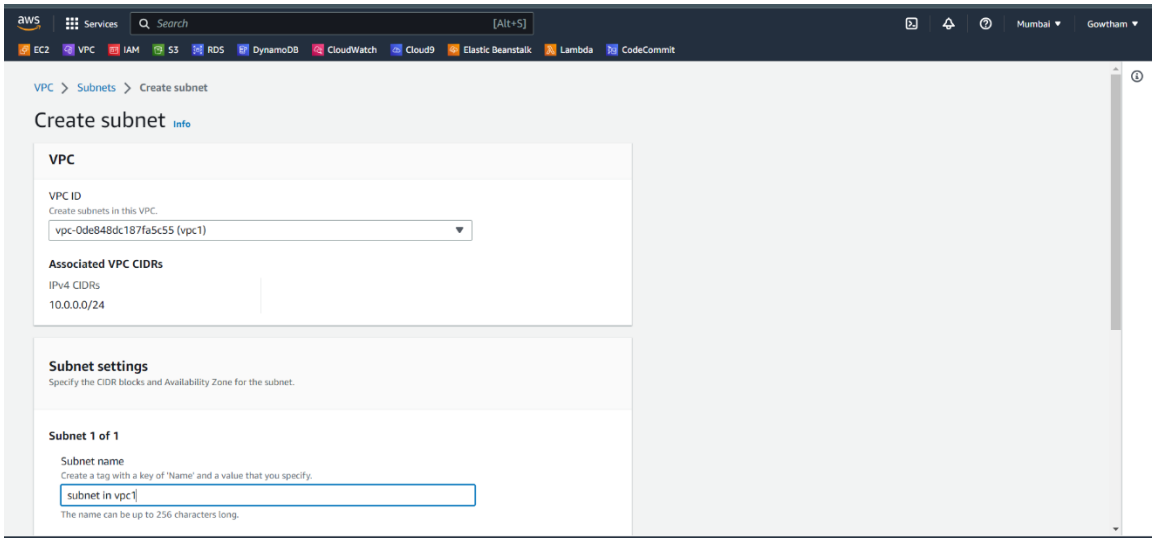
Rename the route table created for custom VPC



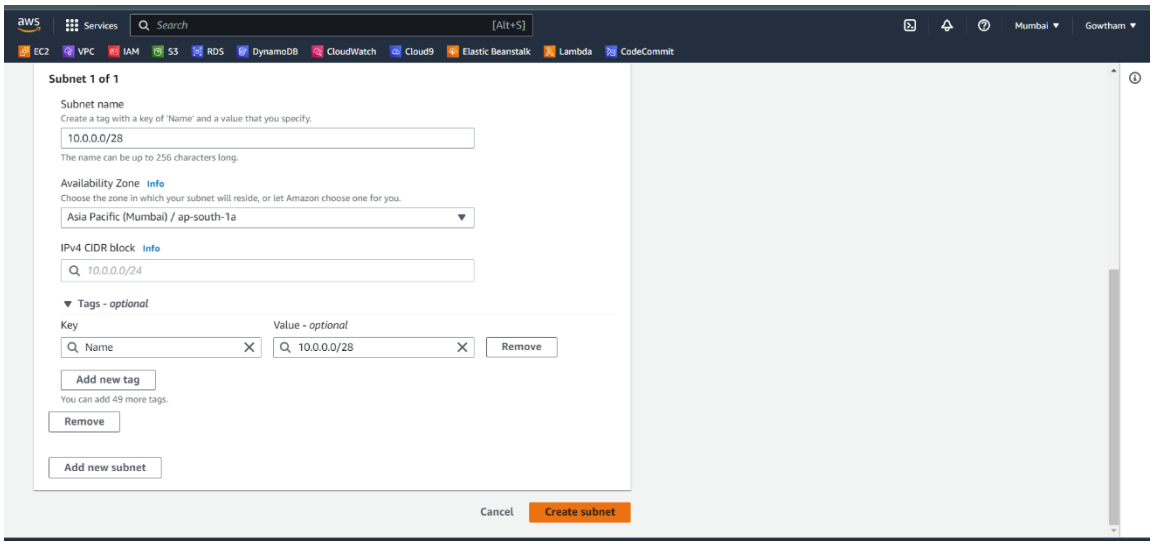
Now we need to create the subnets for our custom VPC.



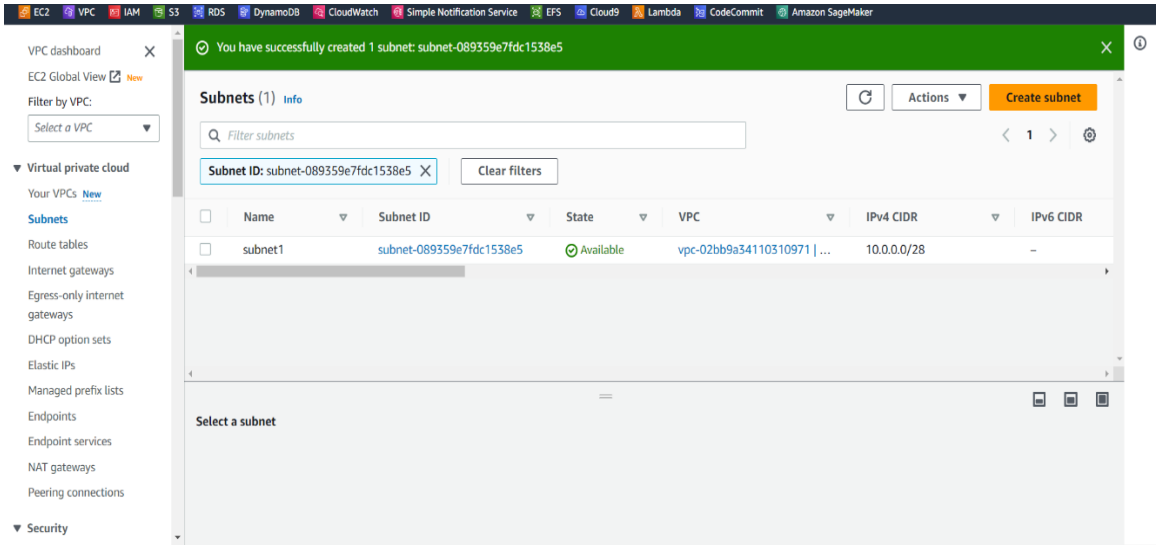
Select the created Custom VPC and give subnet name as subnet1.



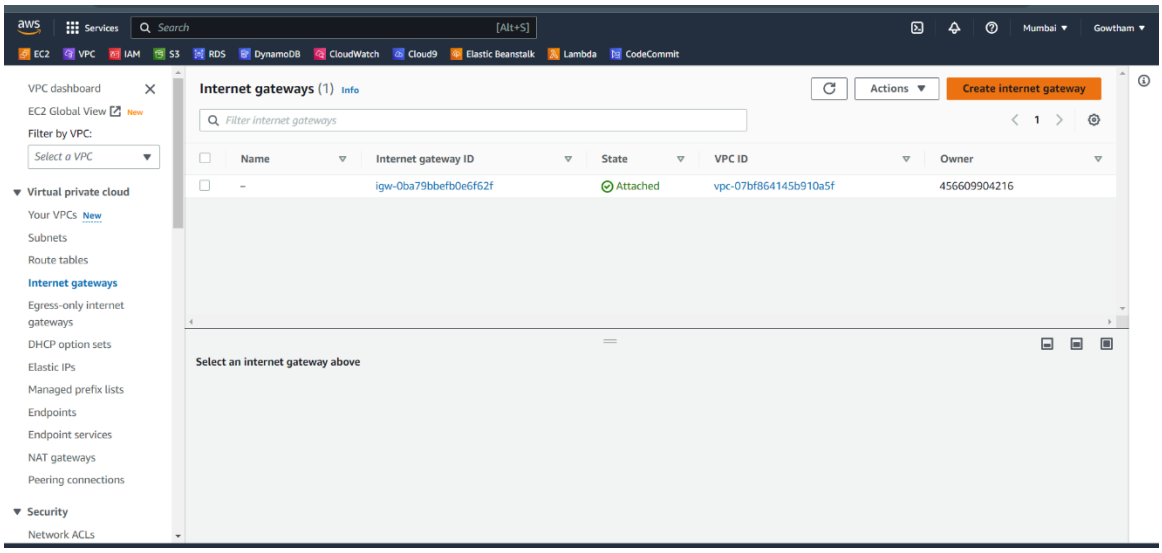
Give subnet1 CIDR (10.0.0.0/28) and create the subnet1.



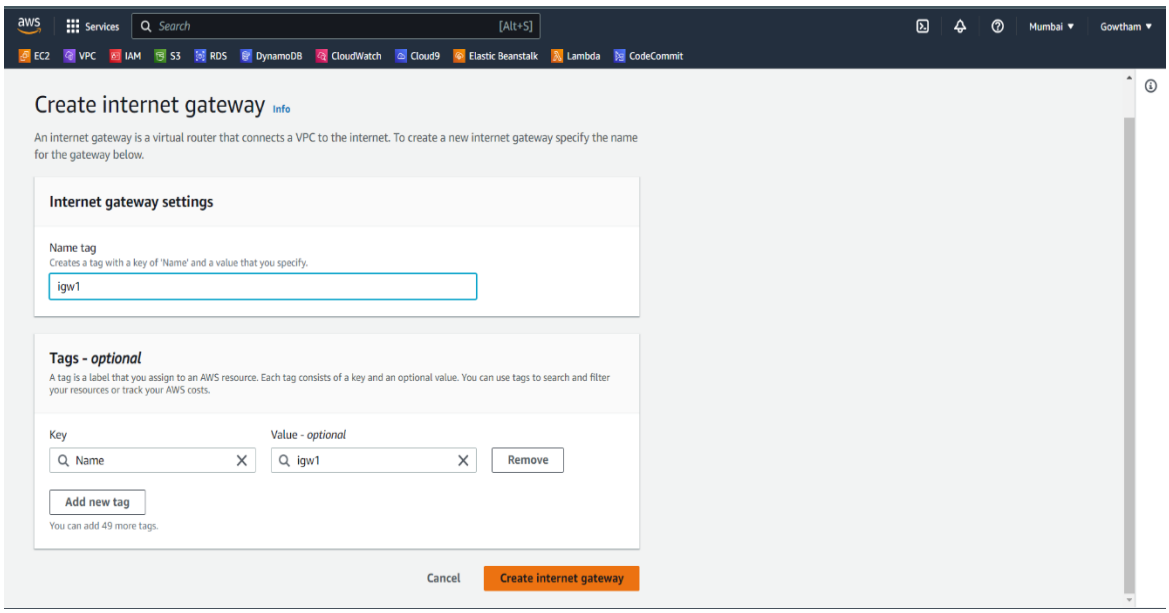
We are successfully created subnet1.



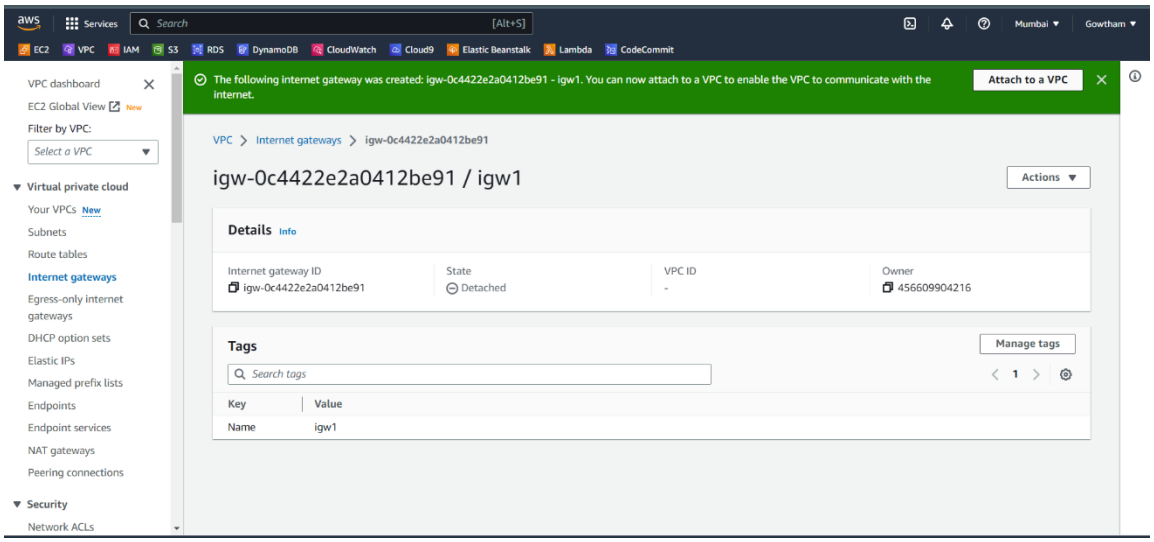
Now we need to create Internet Gateway for Custom VPC



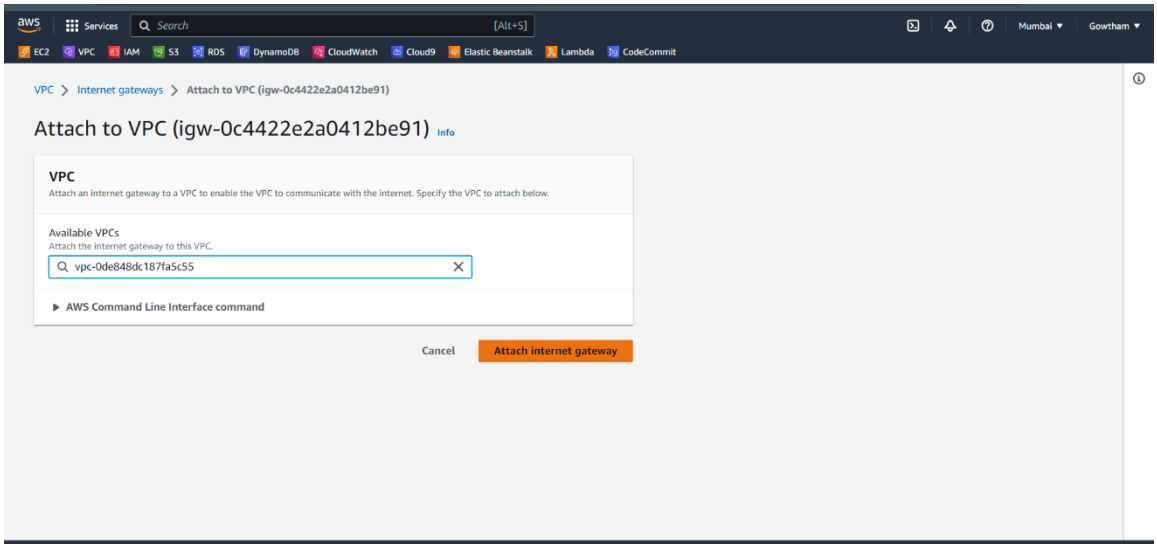
Now name the internet gateway and create it.



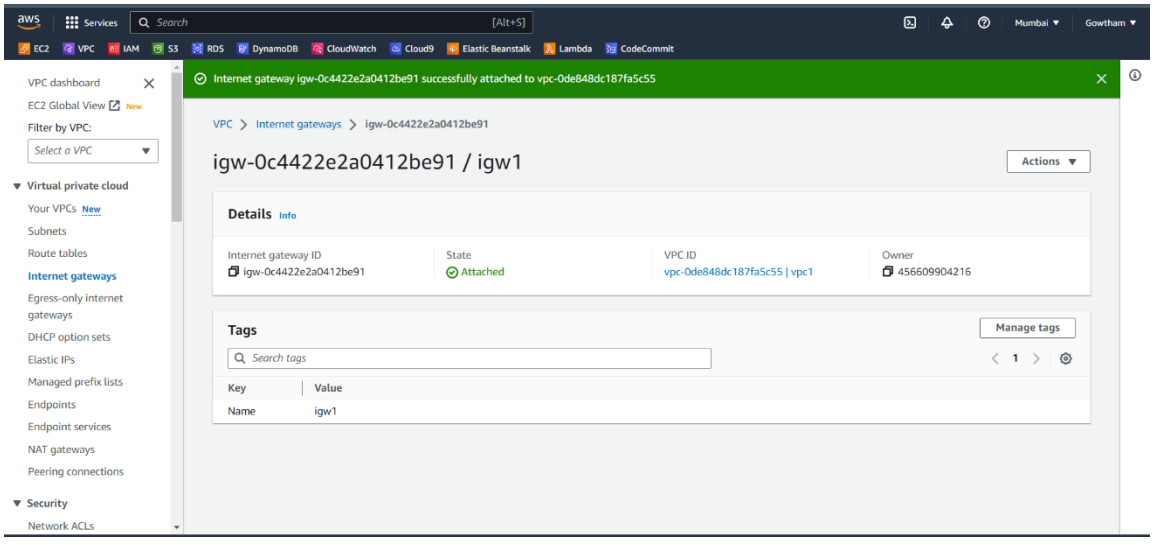
Now we created our IG for the custom VPC , its time to attach the IG with Custom VPC.



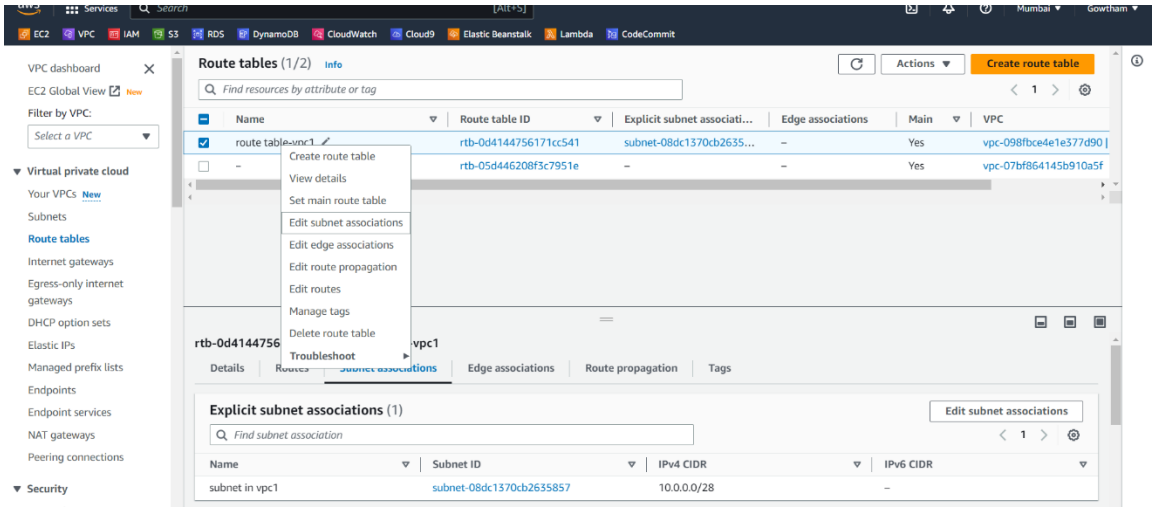
Select the custom VPC and attach it to our internet gateway.



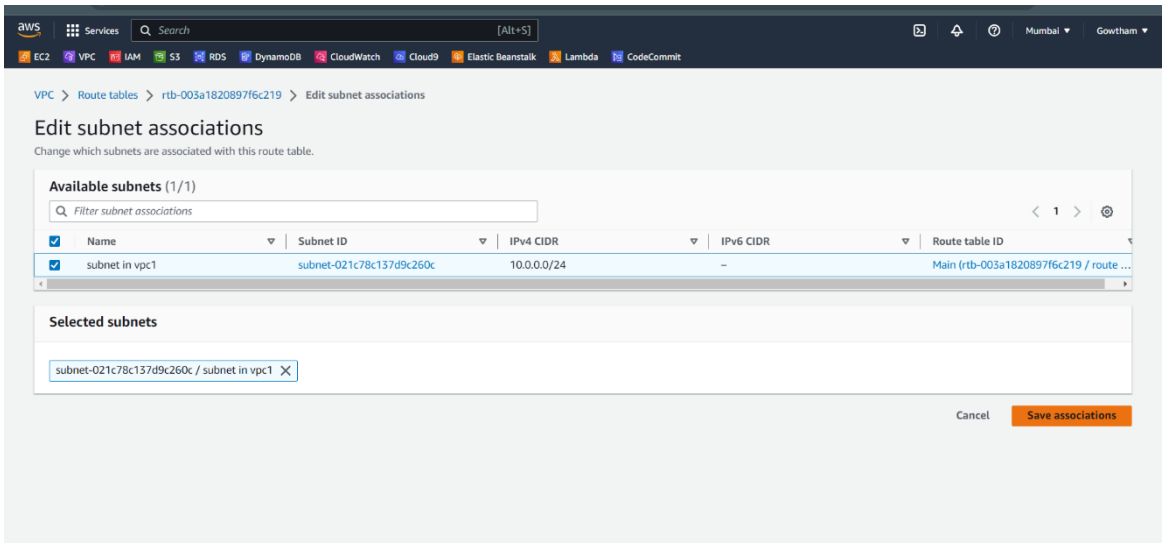
Now we are successfully attached our IG to the custom VPC



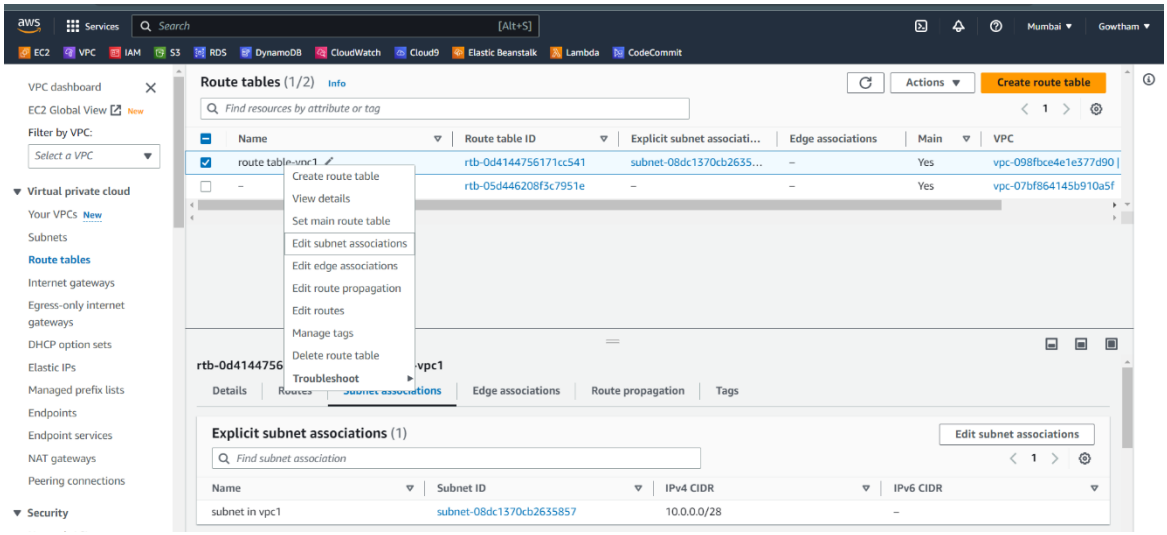
Now we need to update our route table of the Custom VPC. Edit the subnet associations of Route table.



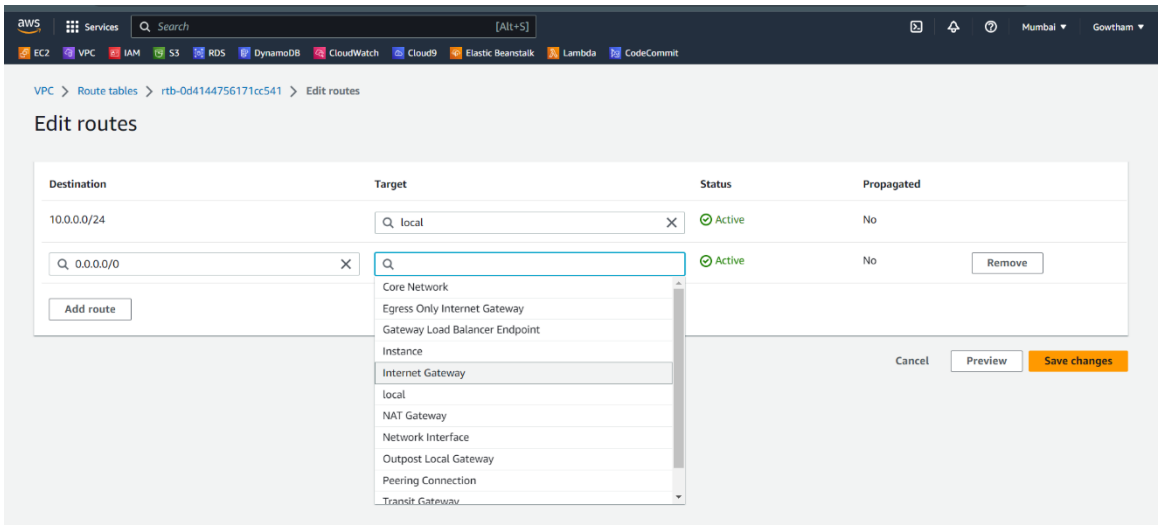
Select the created subnet1 available for this VPC and Save the associations.



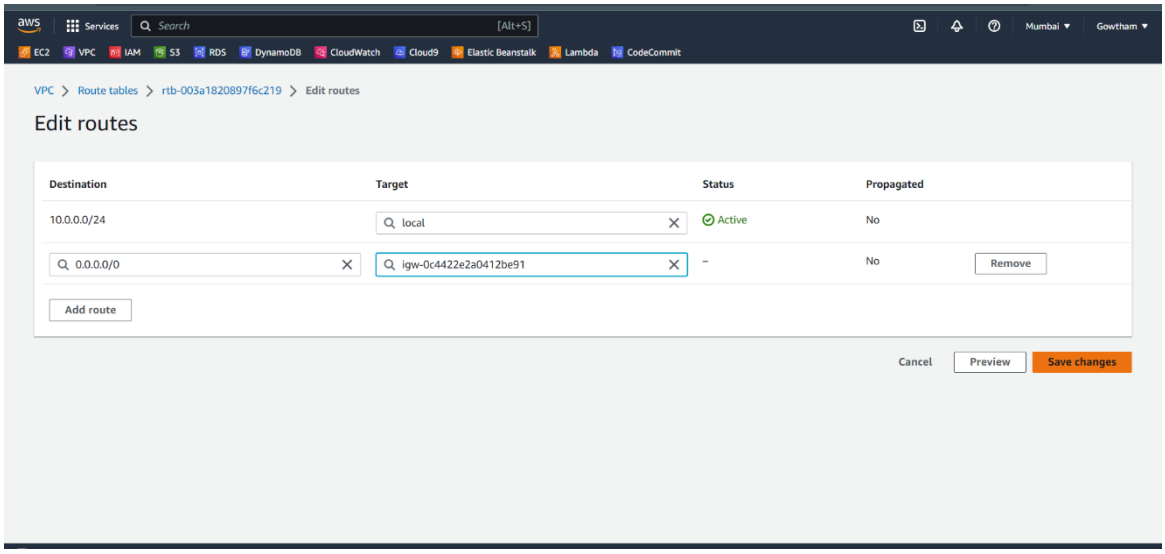
Next edit the routes of the route table to add Internet Gateway route.



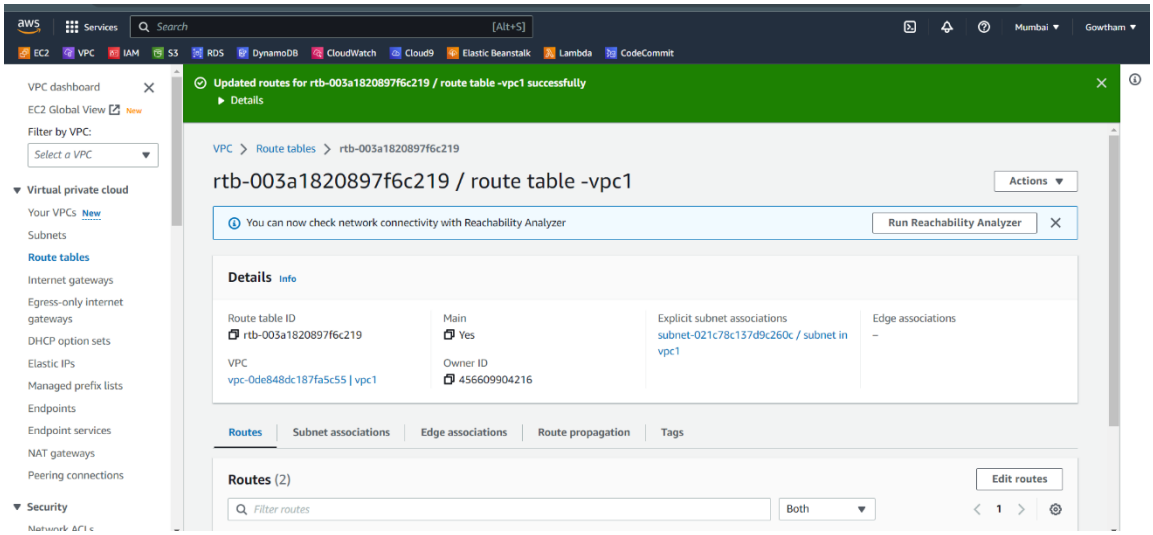
Add new route with destination CIDR 0.0.0.0/0 for IG and select the internet gateway.



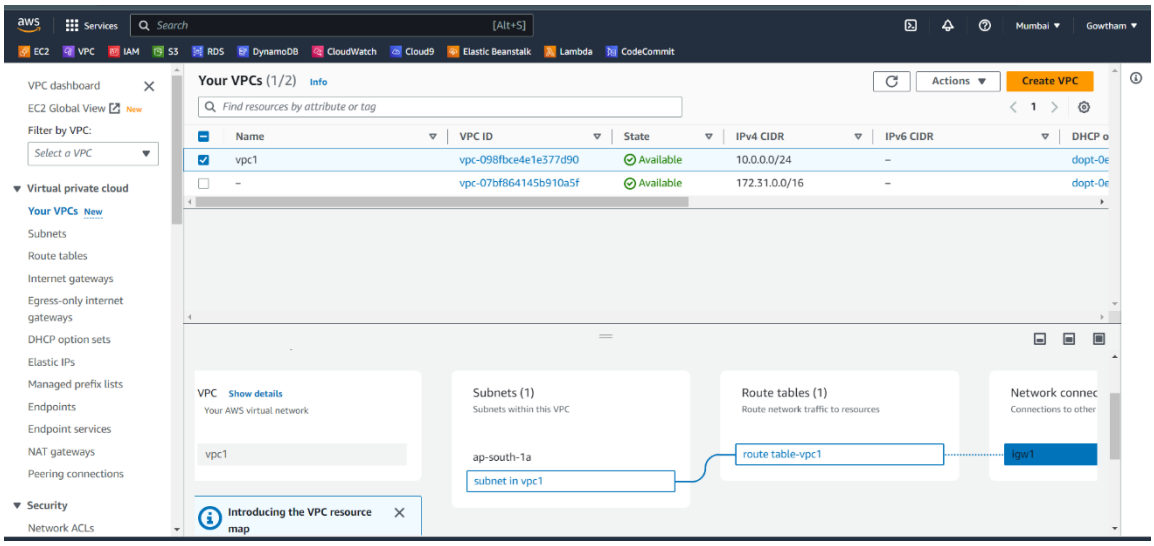
We can find our IG attached to custom VPC , select it and save the changes.



Now we are successfully updated our Route table of the custom VPC.

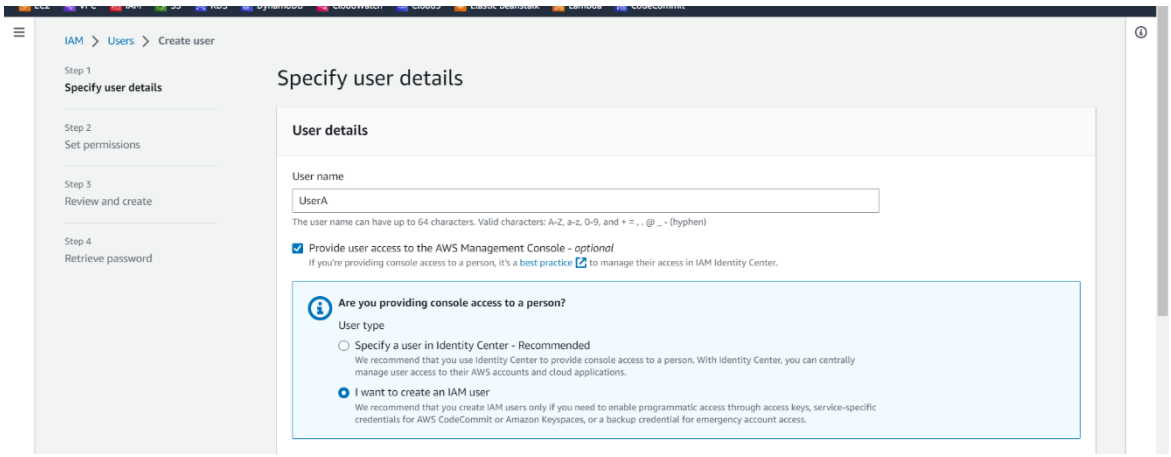


Thus we created our Custom VPC successfully, we can check its resource map as follows.

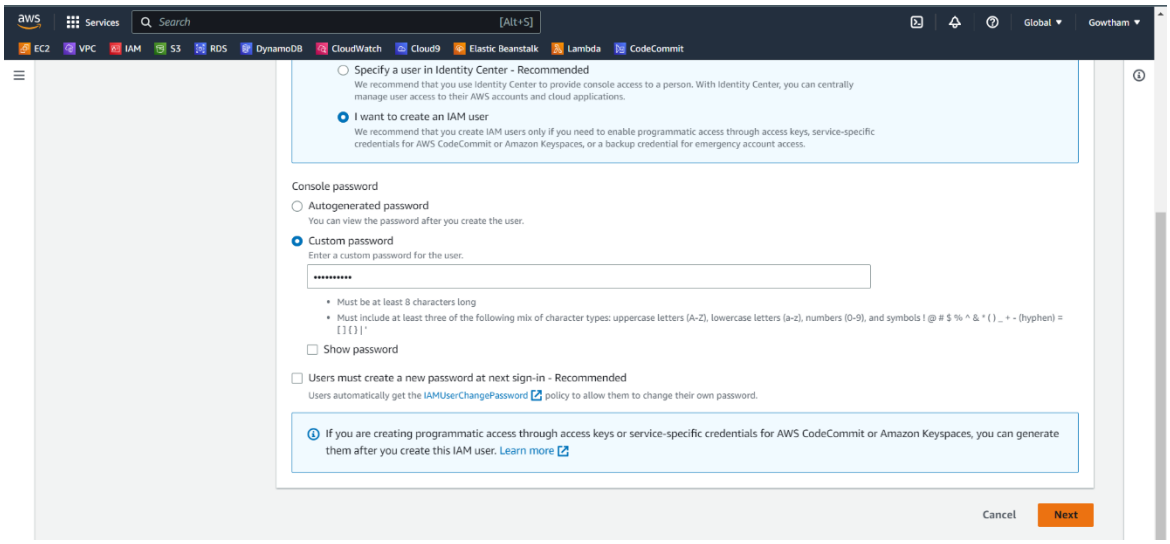


Step:02 Creation of IAM Users

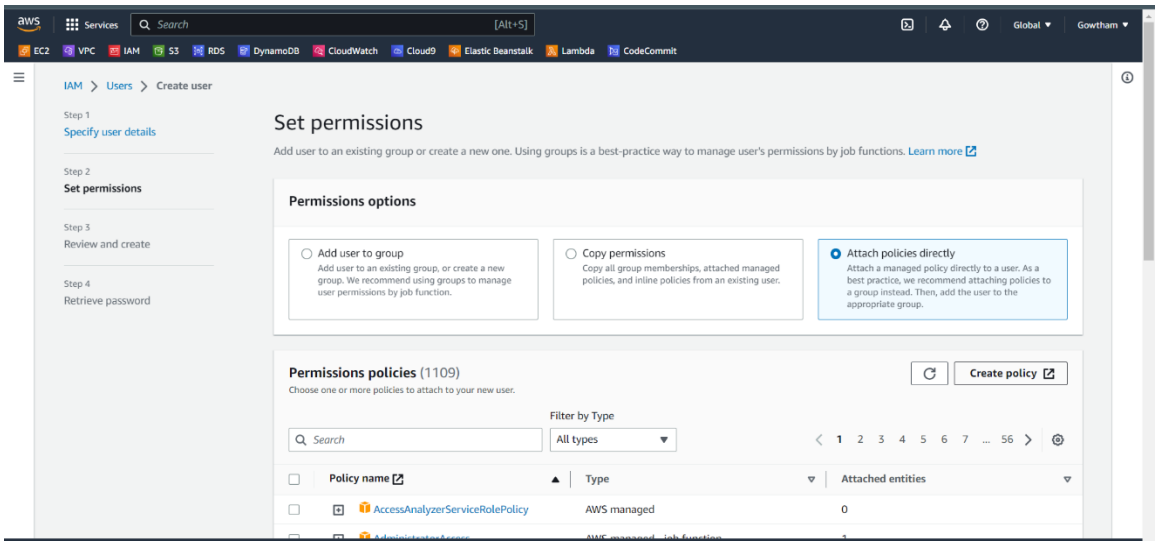
Now go to IAM and Click on Users



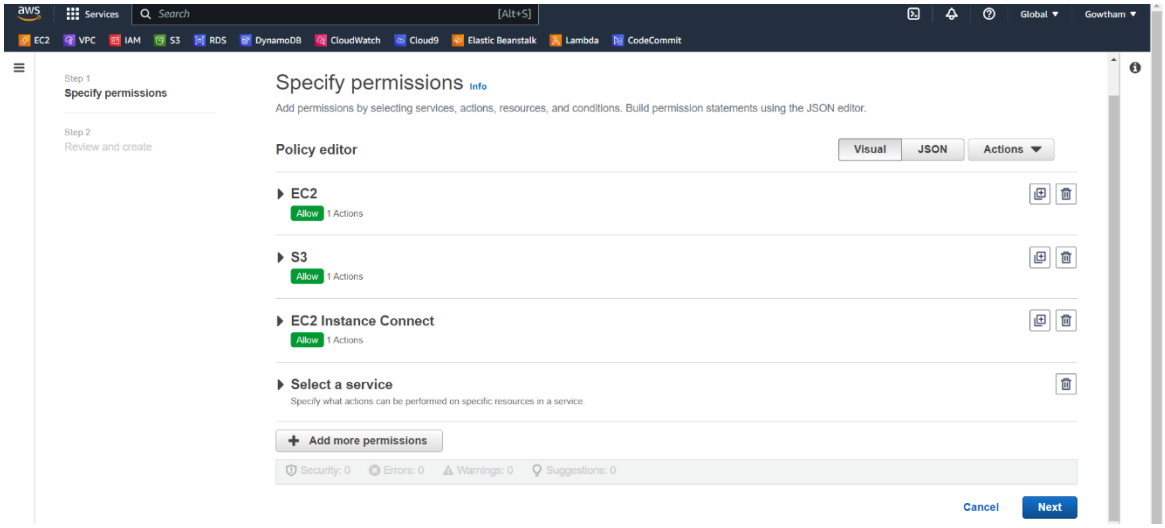
Create Custom Password for UserA.



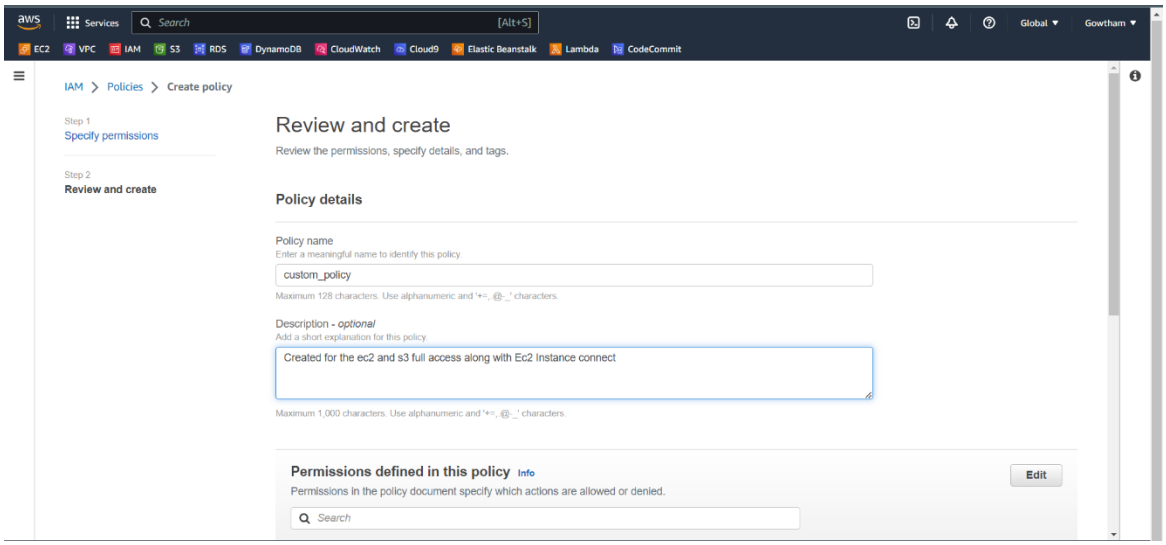
Select Attach Policies Directly and Click on Create Policy.



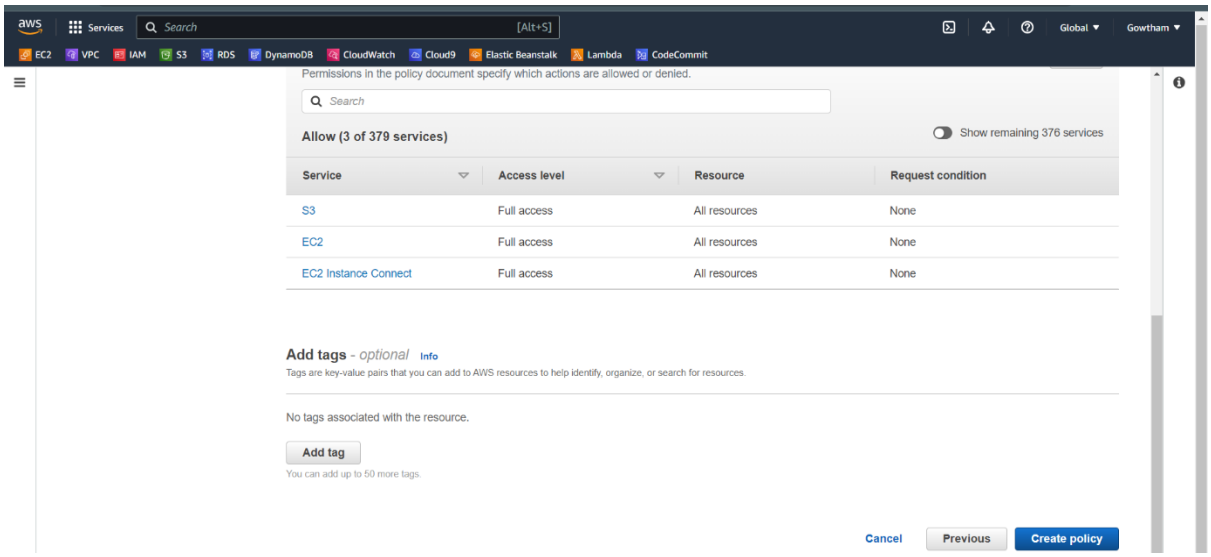
Specify the permissions that should be attached to the policy.



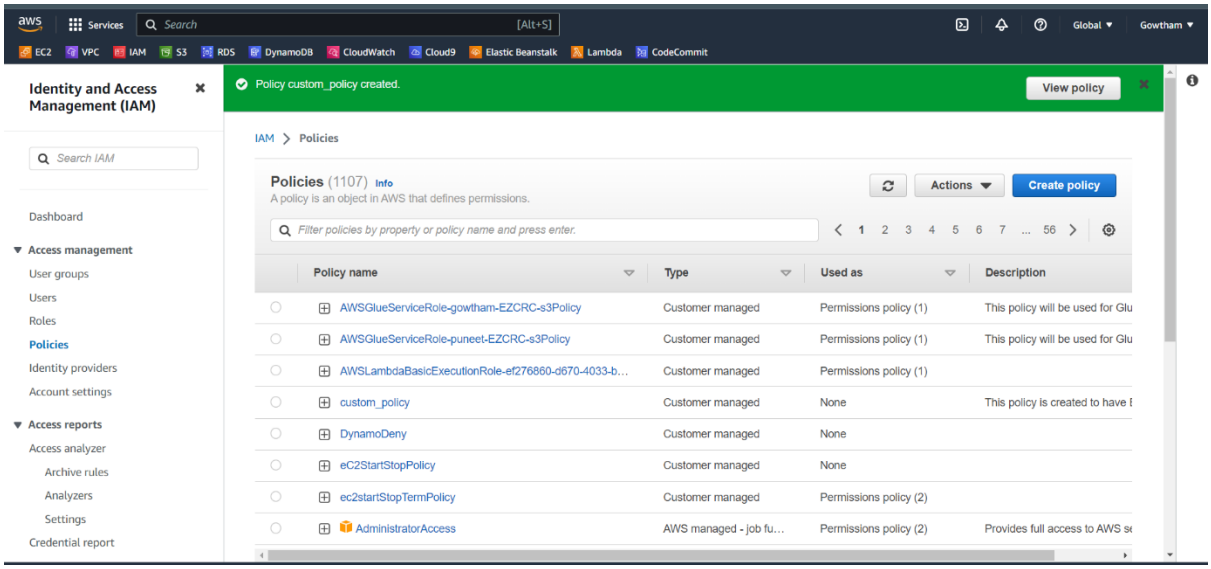
Specify the policy details.



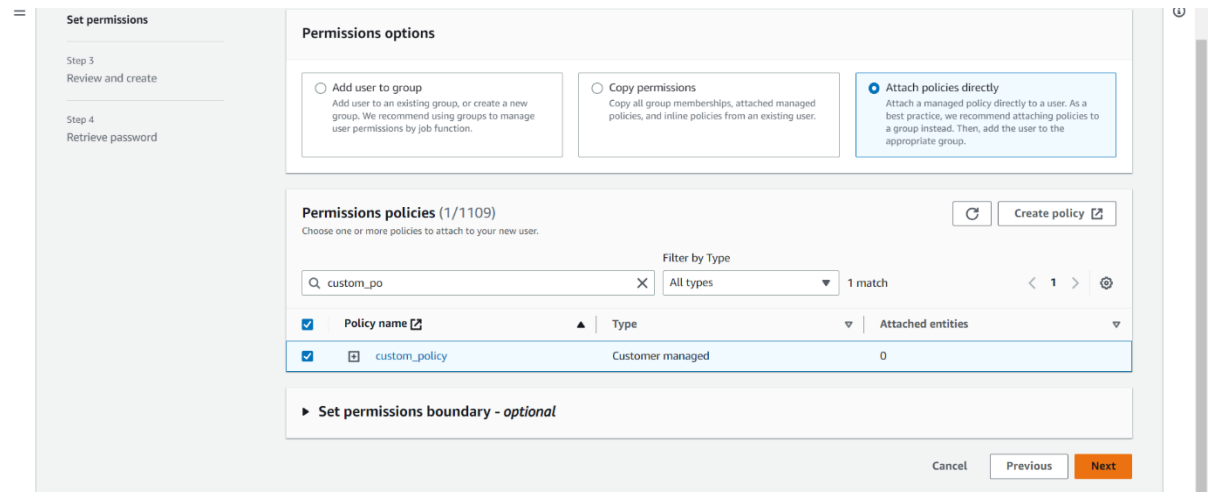
Now review the permissions and create the policy.



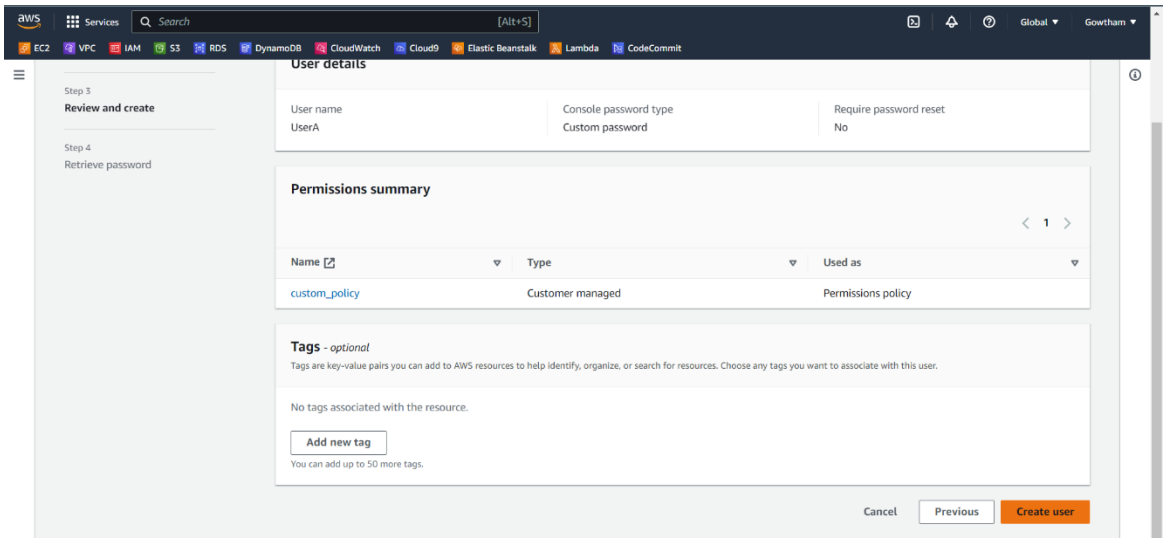
The policy has been created successfully.



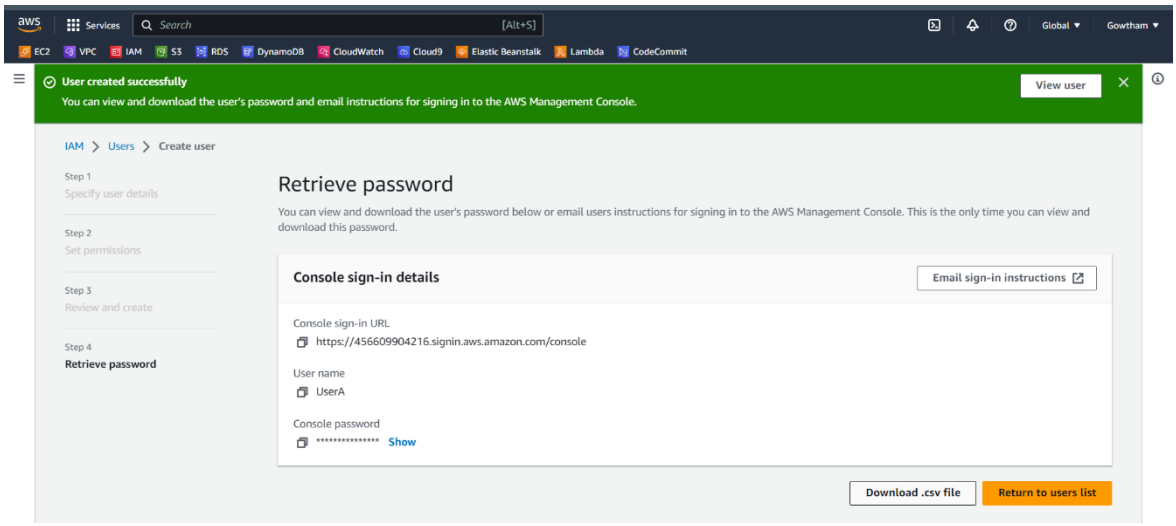
Now attach the newly created policy to UserA.



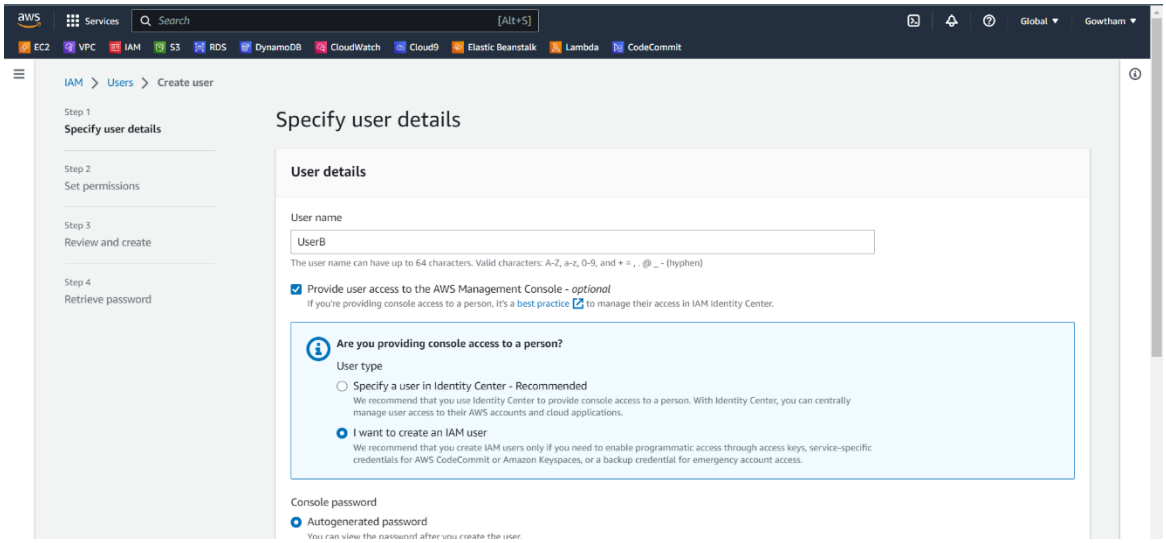
Now review the permissions and create the user.



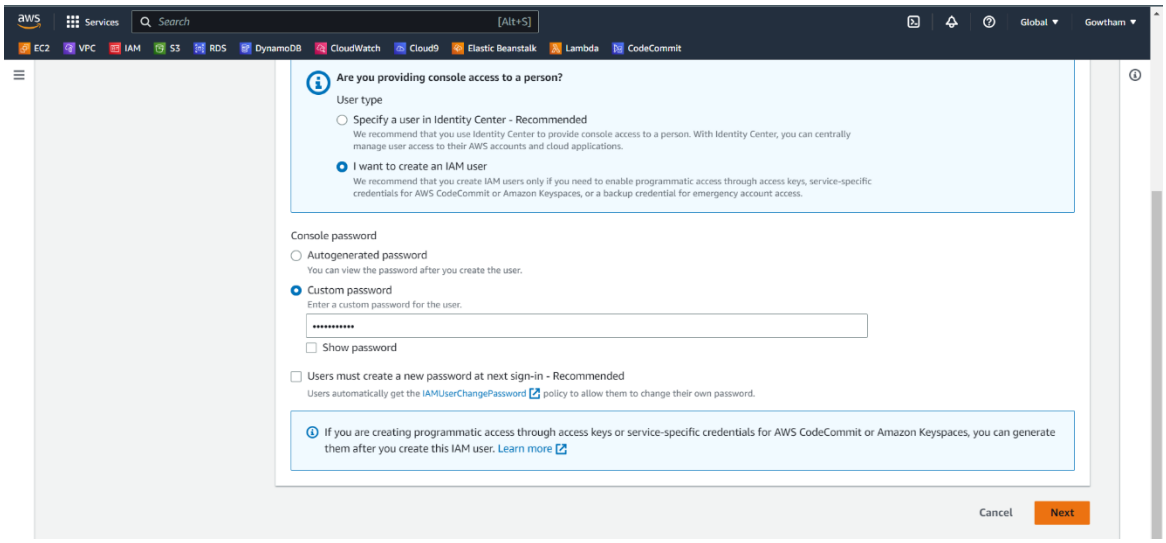
After the successful creation of the user, retrieve credentials of user to sign in



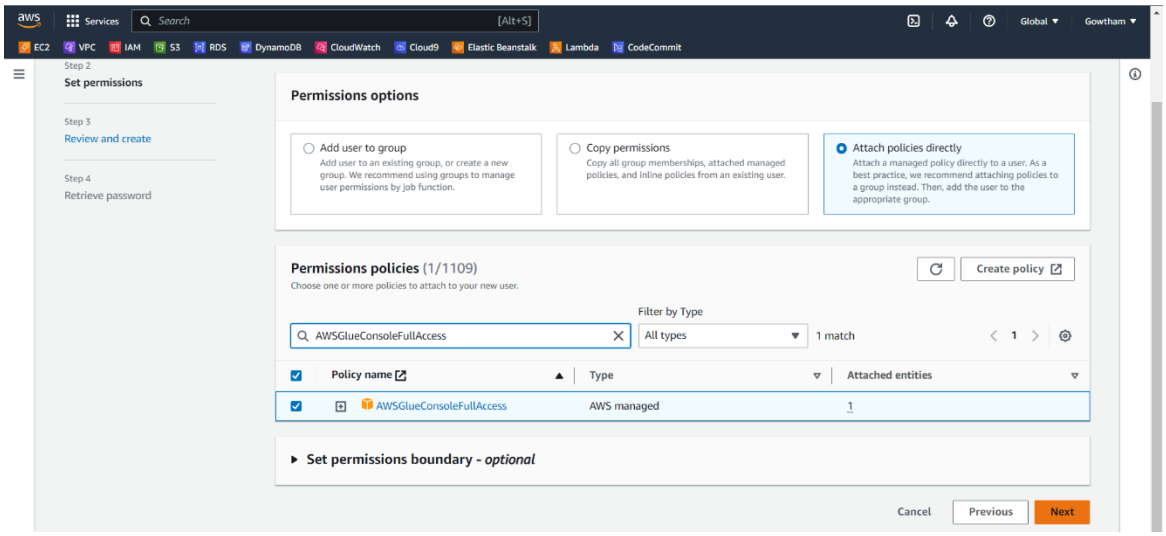
Now create another user by specifying the name as UserB.



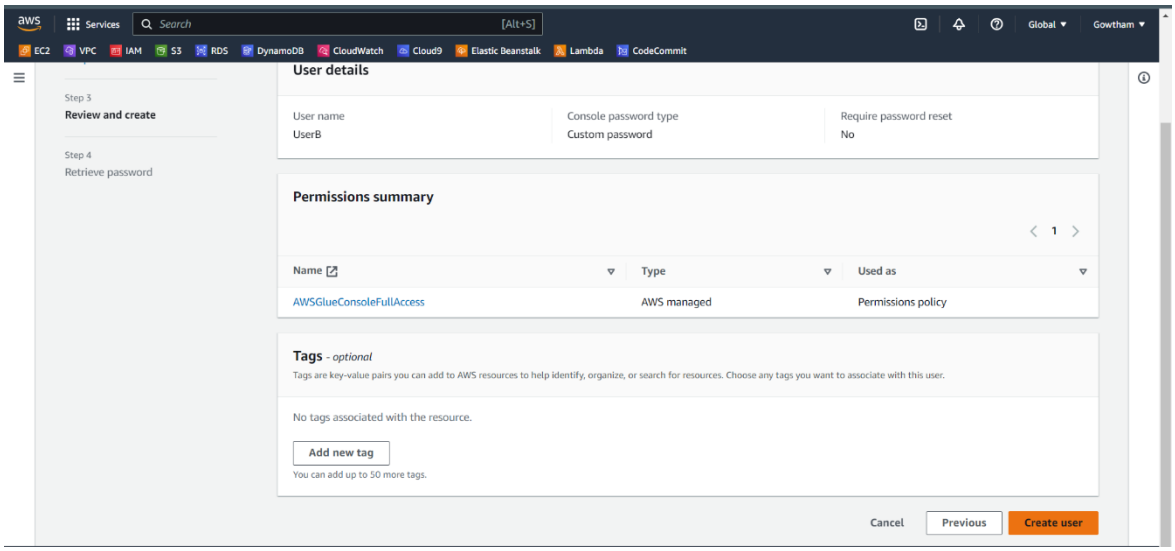
Enter the custom password for the user.



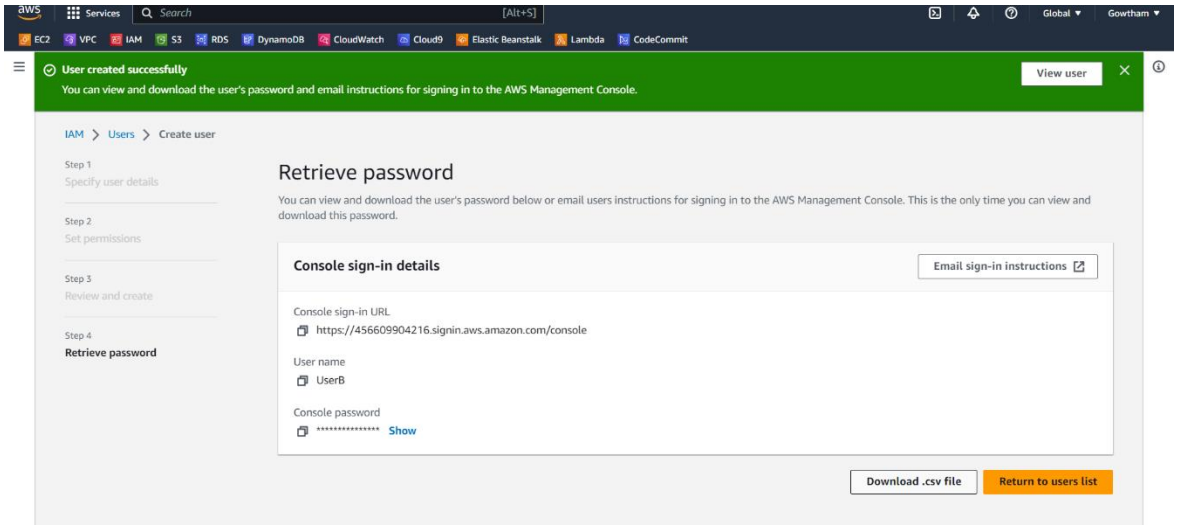
Attach the policy AwsGlueConsoleFullAccess directly



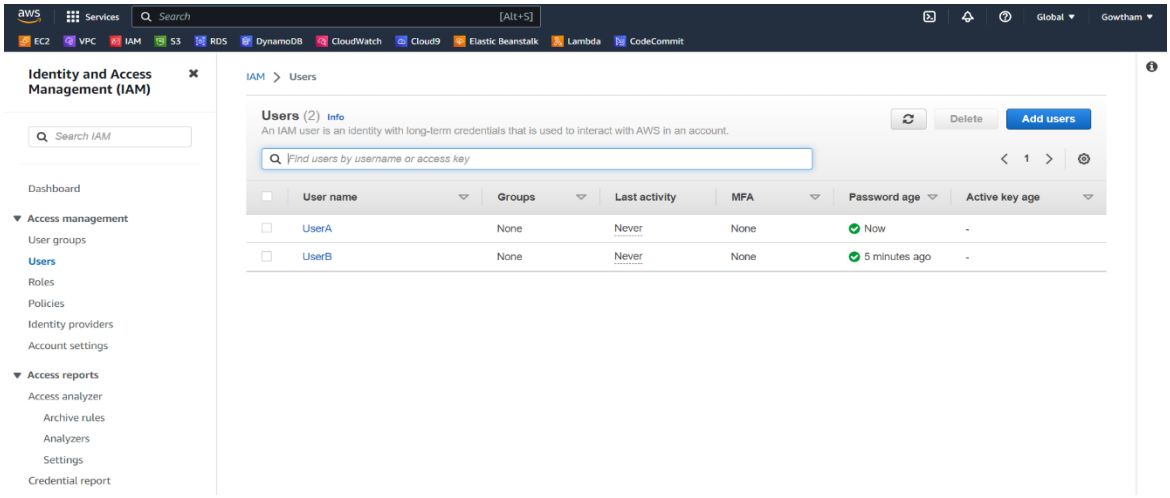
Now review the user details and permissions and create the user.



Now retrieve the password after the creation of UserB.

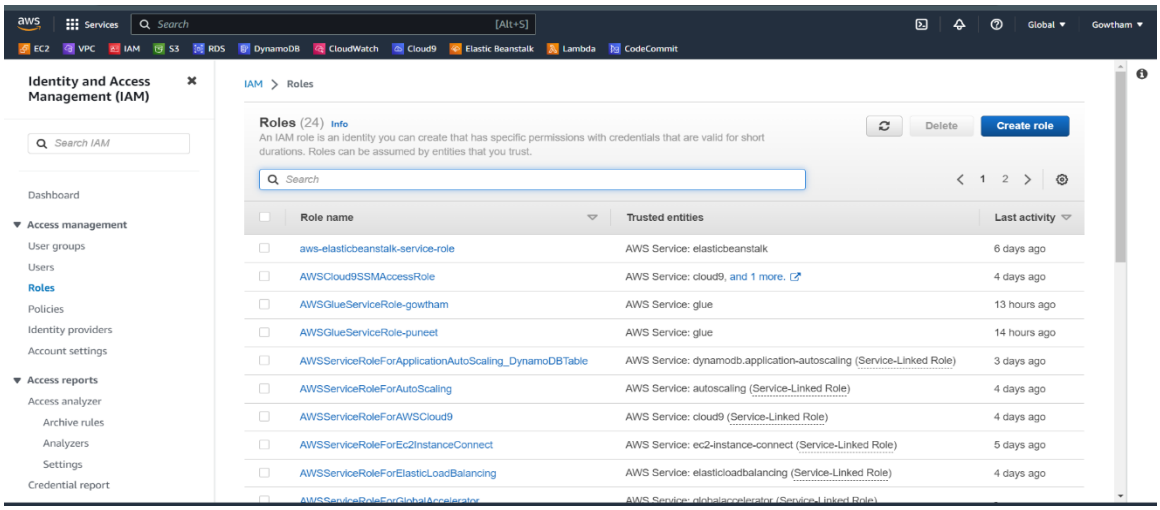


Now we have successfully created two users

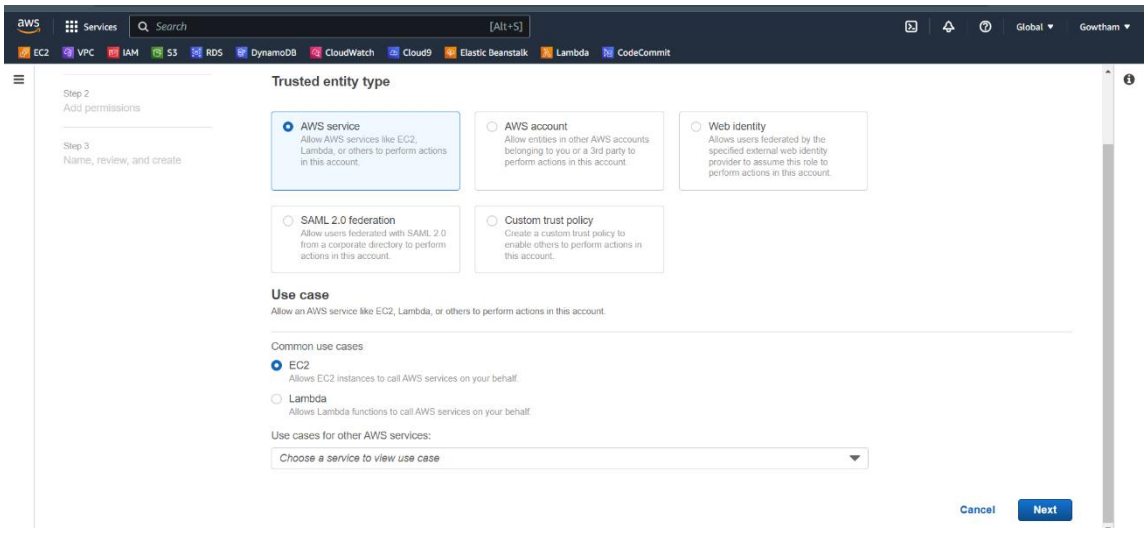


Step:03 Creation of IAM Role For S3Full Access

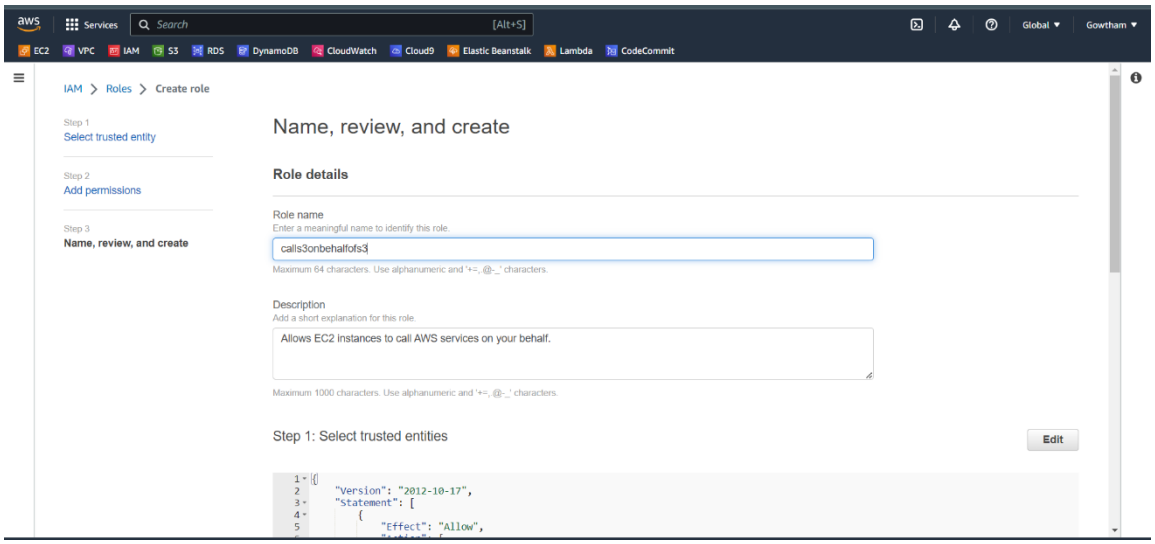
Goto IAM > Roles and click on create role



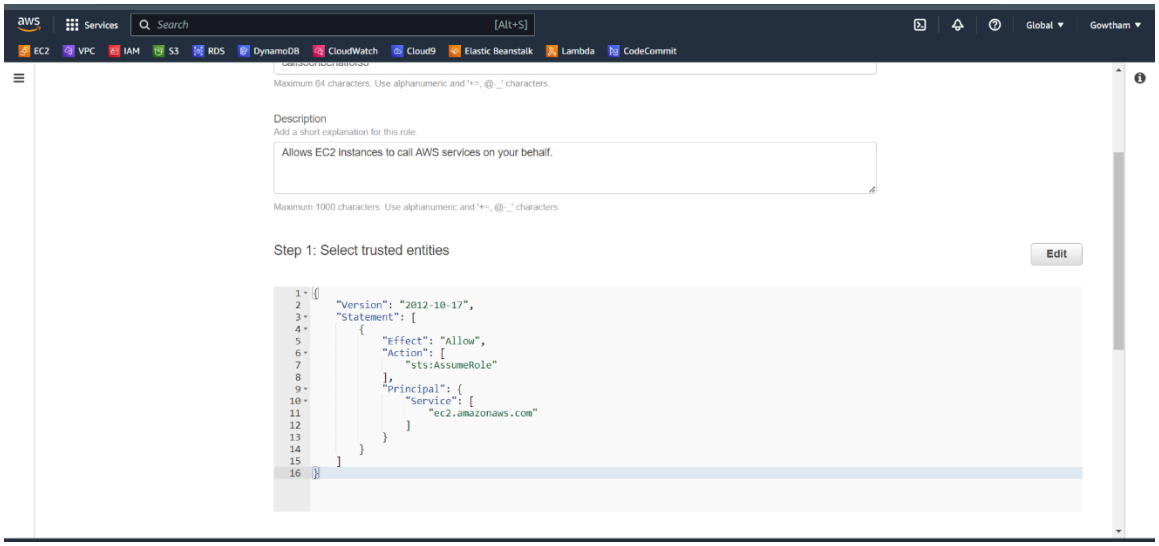
Selected the trusted entity as AWS Service and use case as EC2



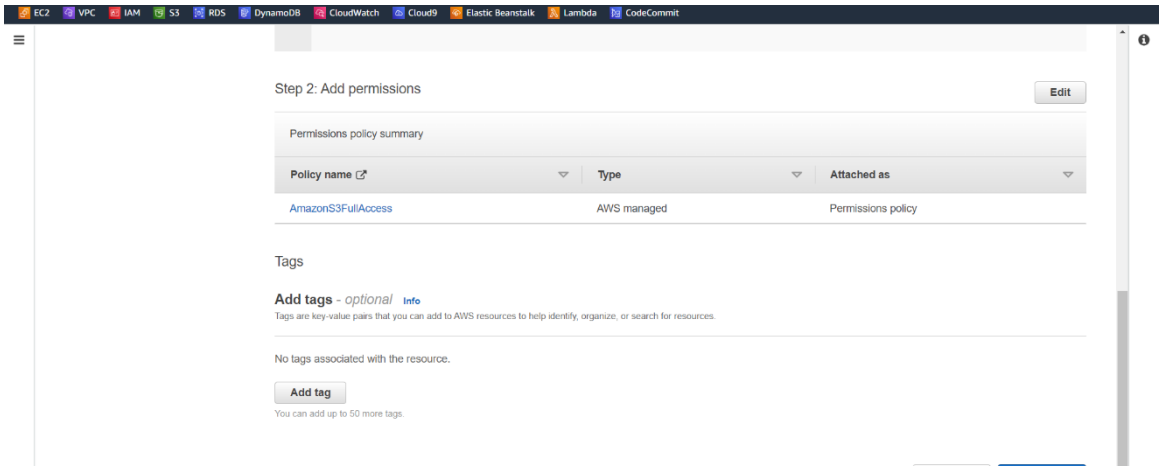
Give a name to the IAM Role as “calls3onbehalfofec2”



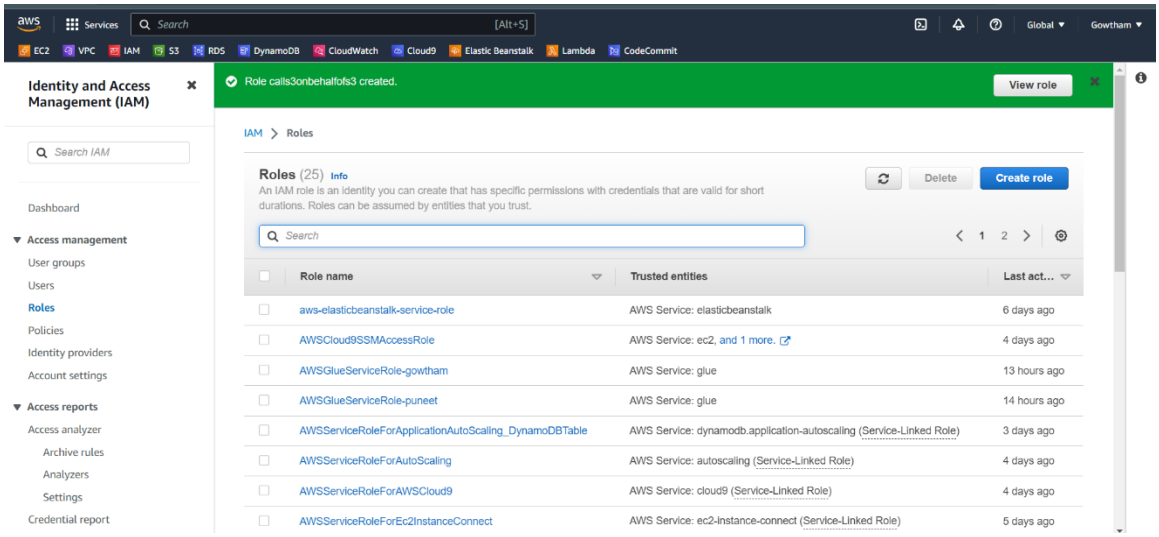
Verify the trusted entity EC2(AWS Service)



Verify the Permission added and create the role

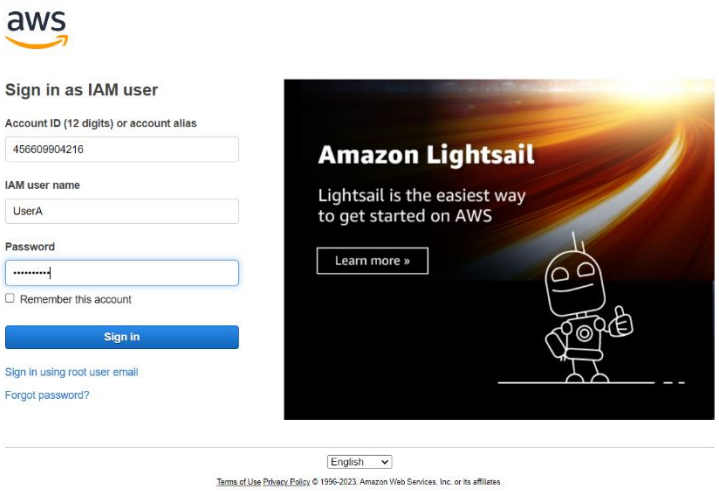


IAM Role “calls3onbehalfofec2” has been created

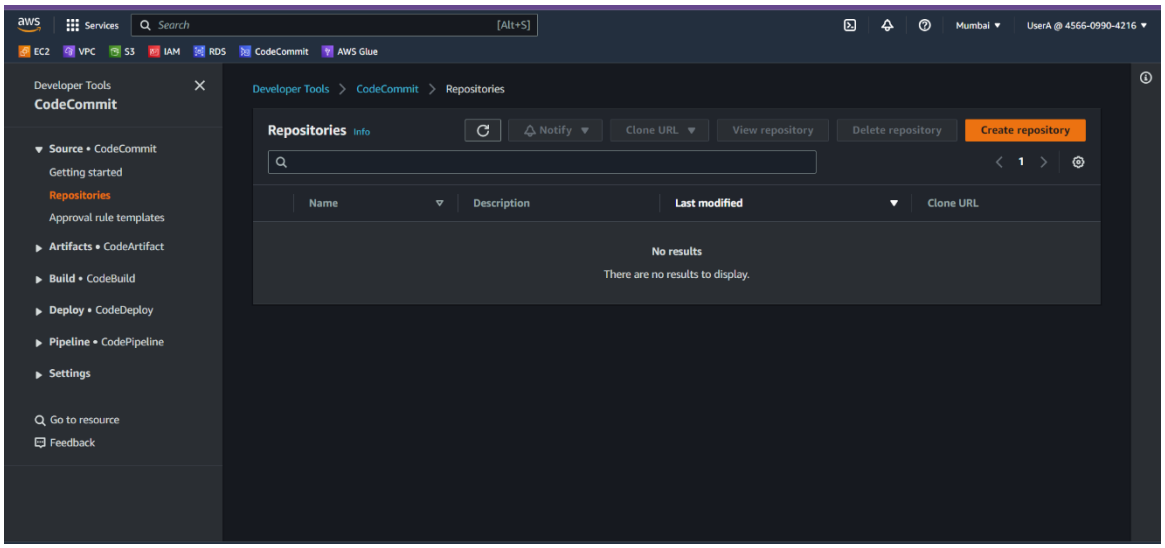


Step:04 Upload the csv file to the CodeCommit Repository.

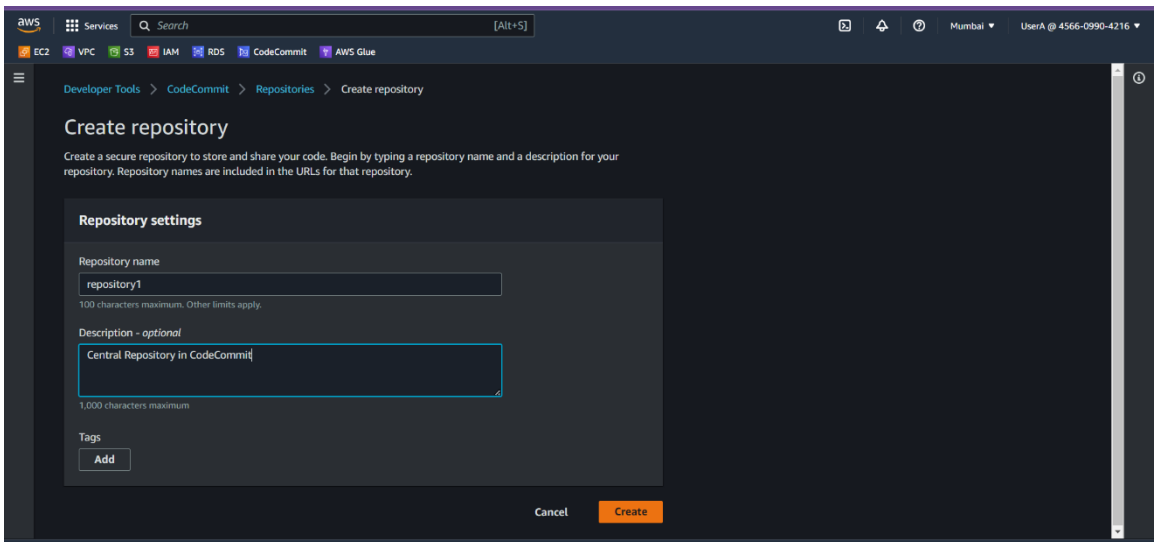
Enter the credentials like user name and password to sign in to the IAM User account



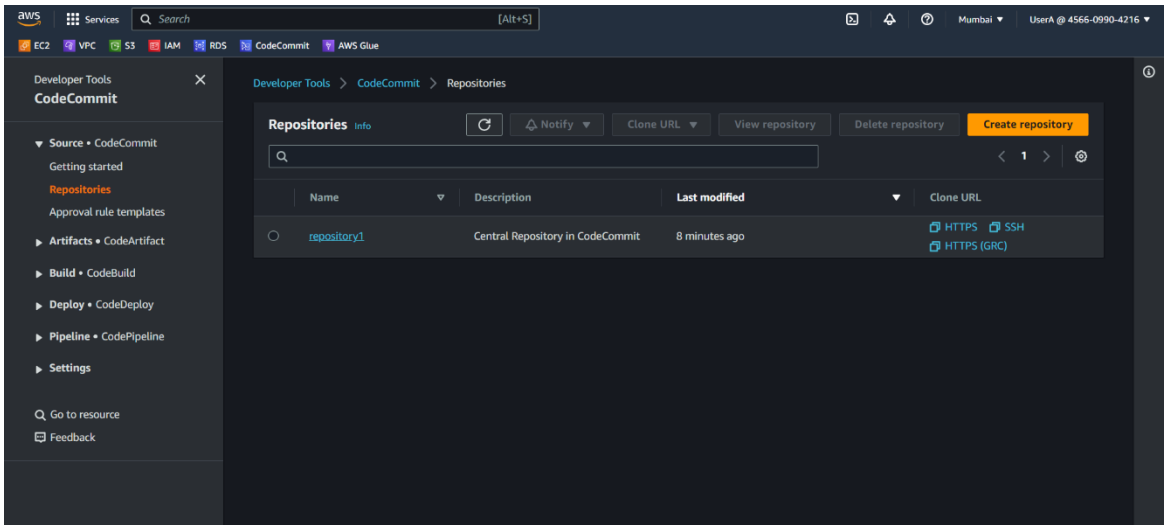
Goto Developer Tools > CodeCommit > Repositories and click on Create Repository



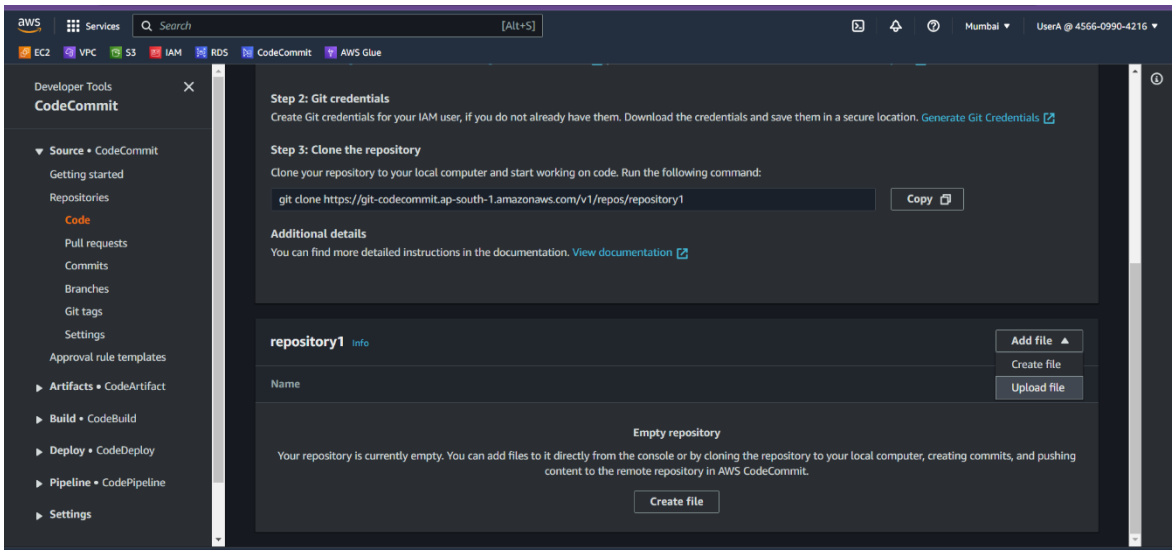
Give a name to your repository and click on create



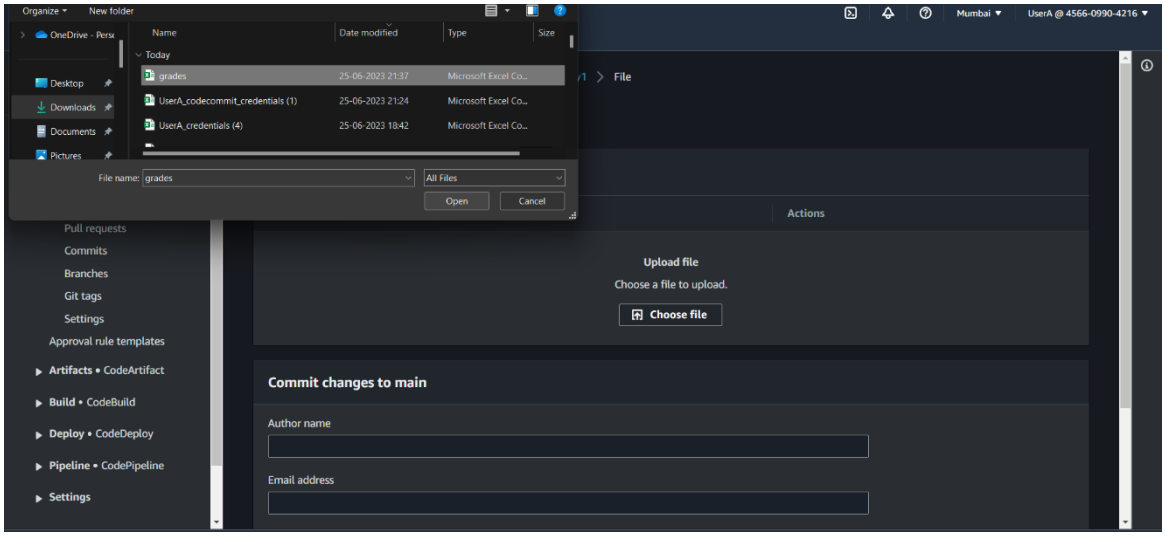
A new repository has been created and now copy the Clone URL(HTTPS)



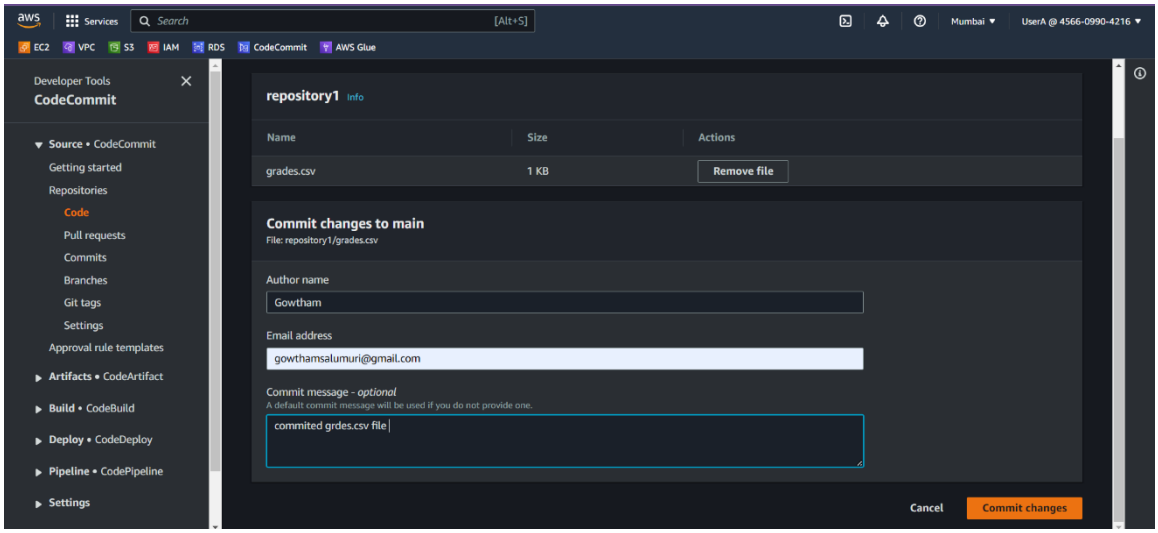
Click on add file to upload a file into the newly created repository



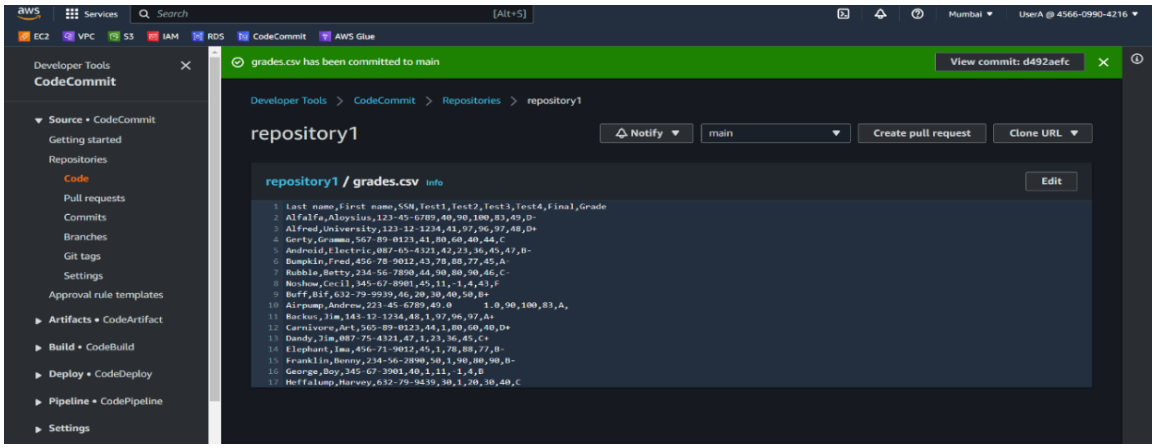
Now browse the file from your folders and select it to upload



Now enter the details of the commit and click on Commit Changes

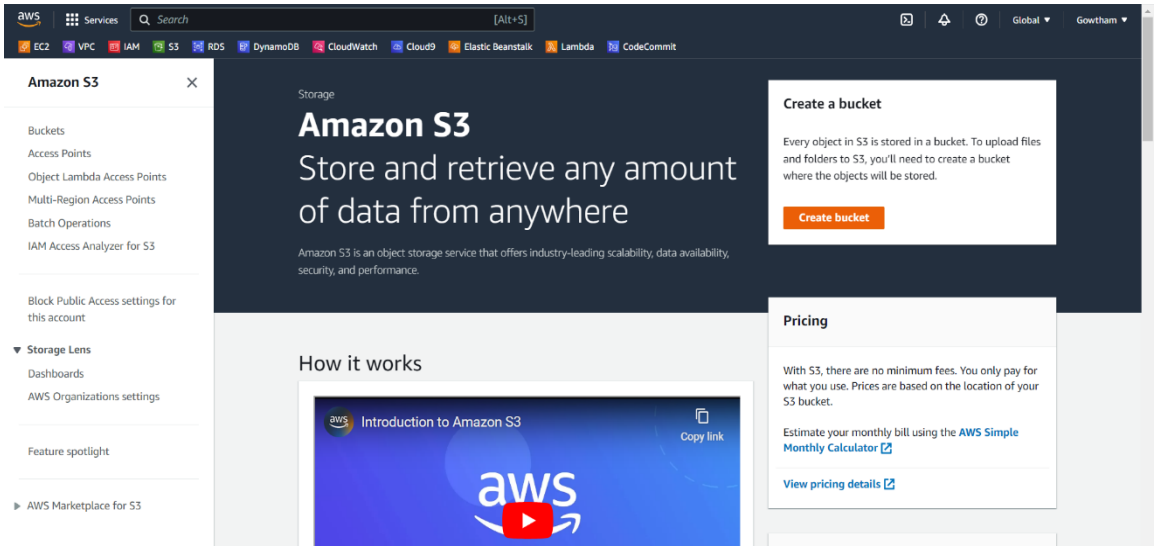


The csv file has been uploaded into the CodeCommit Repository

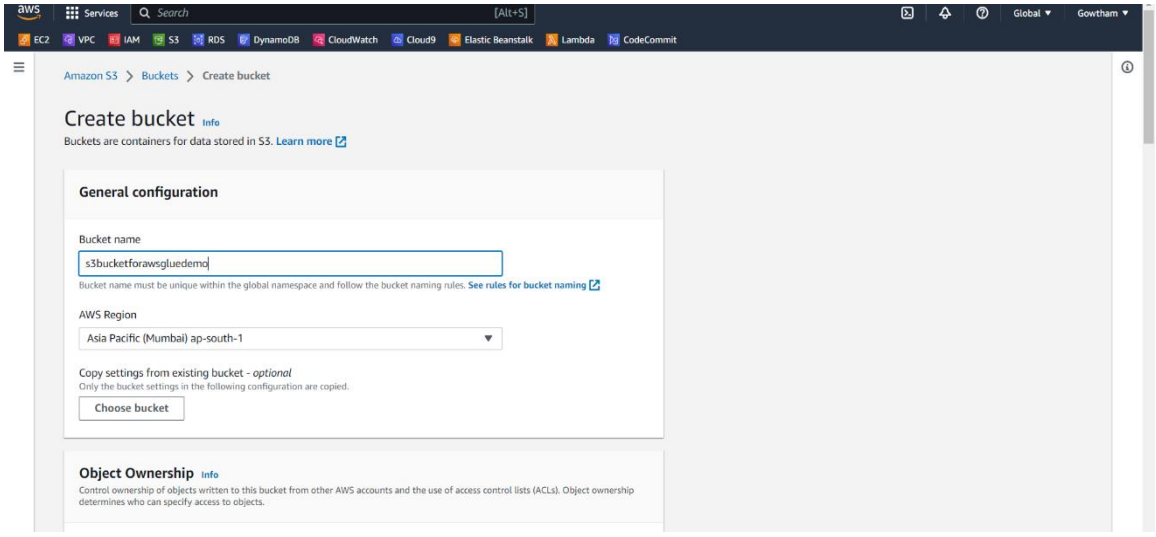


Step:05 Creation of S3 Bucket to store the csv file

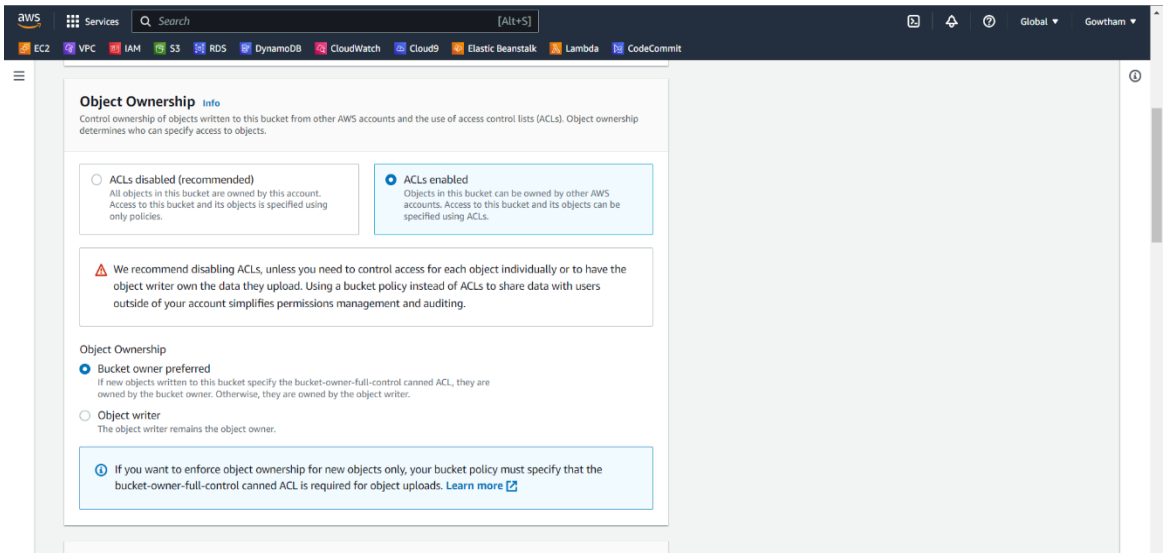
Go To Amazon S3 and click on Create Bucket



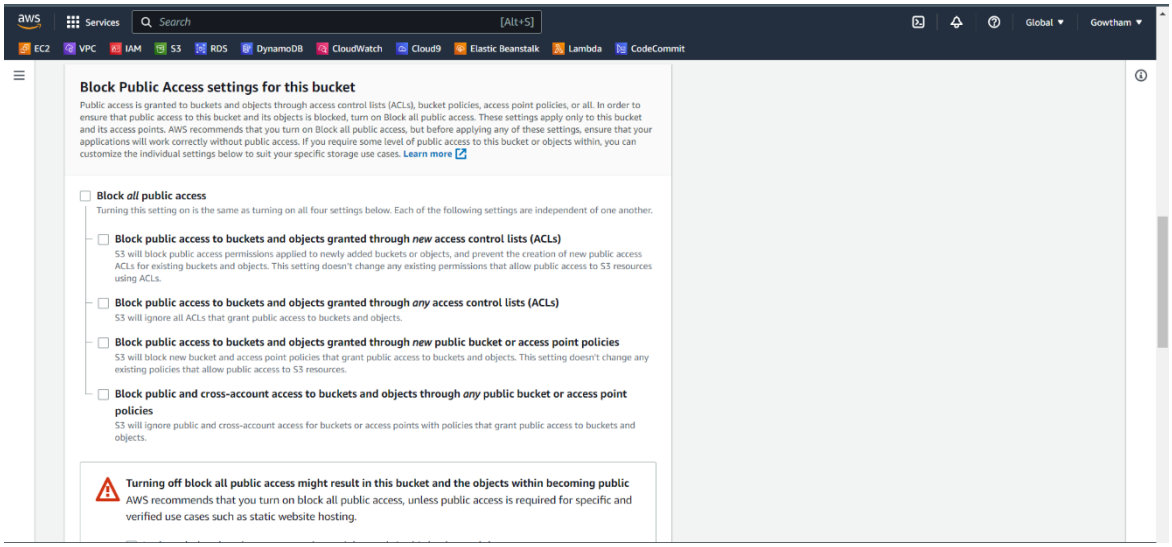
Give the name to the bucket and specify the AWS region



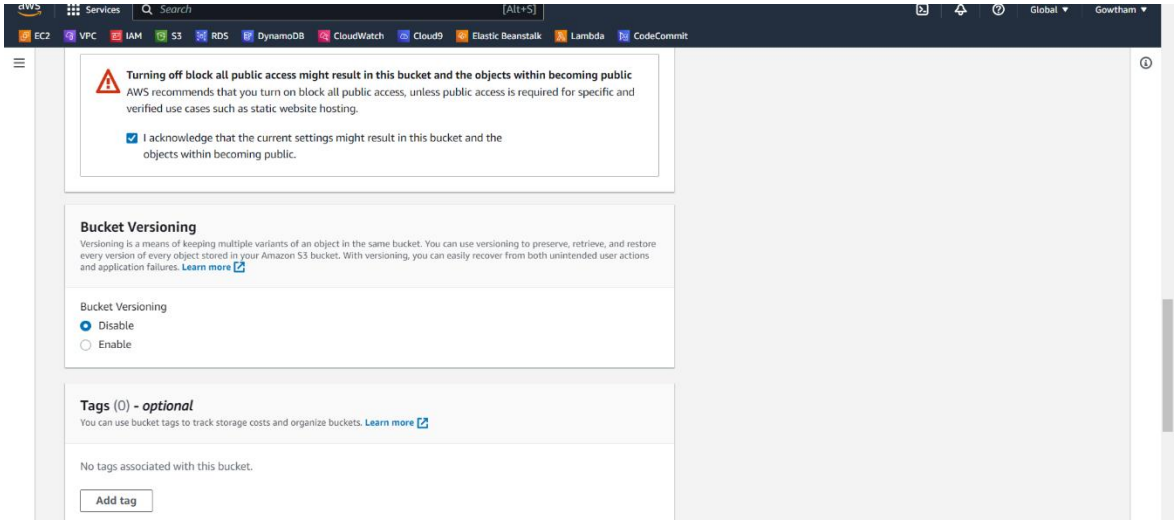
Set Object Ownership as the ACLs enabled



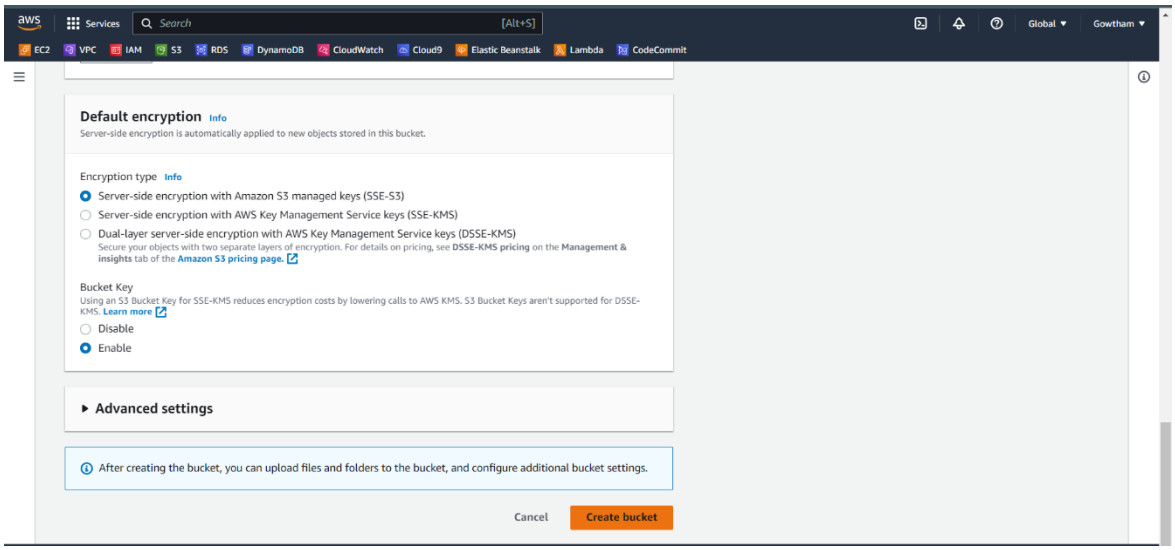
You can give access to your bucket to others by unchecking the block pubic access



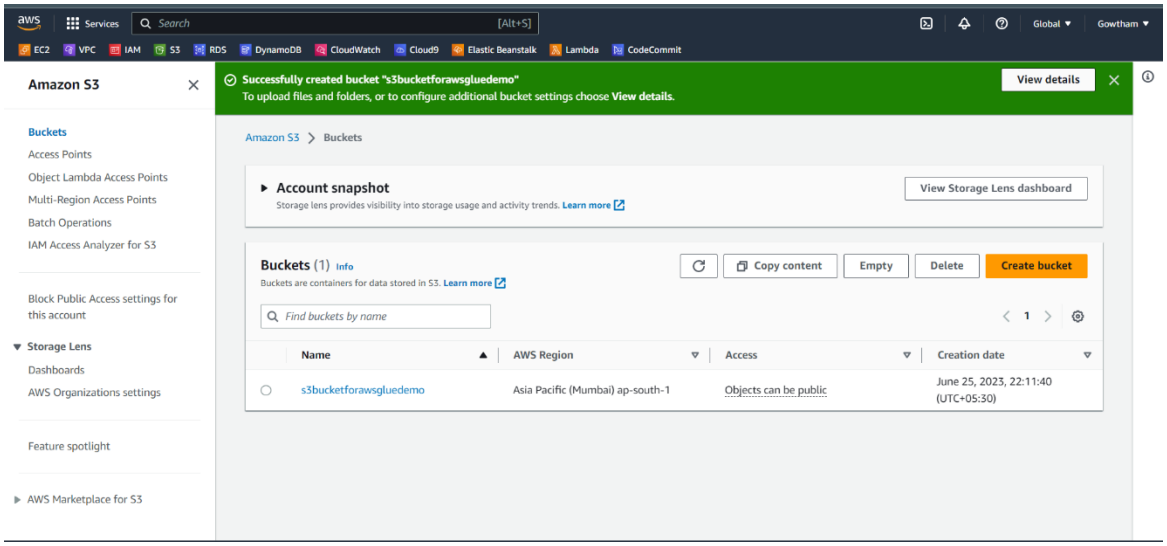
Click on the acknowledgement check box and disable the bucket versioning



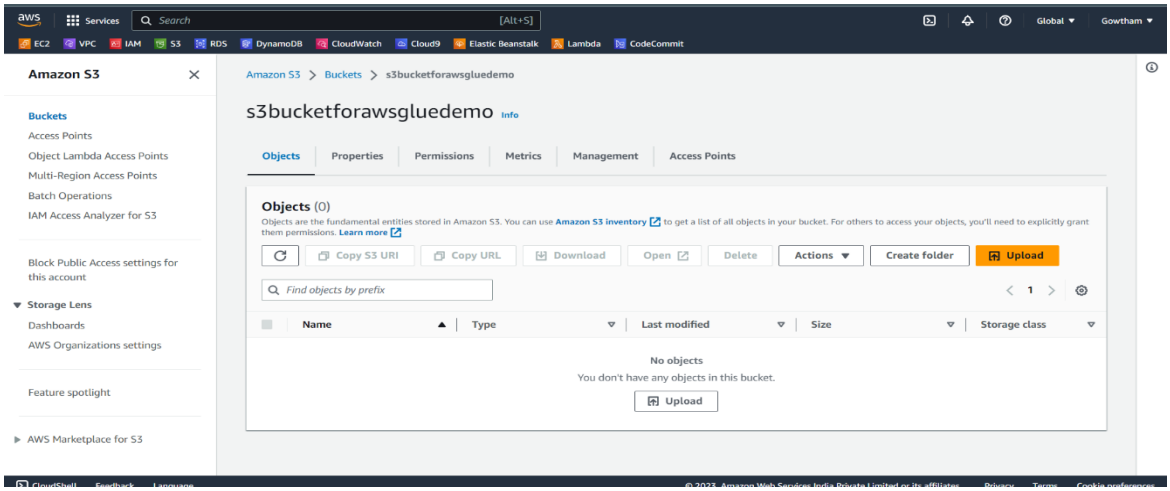
Use the Default encryption type as SSE-S3 and click on create bucket



Now a bucket with name “s3bucketforawsgluedemo” has been created

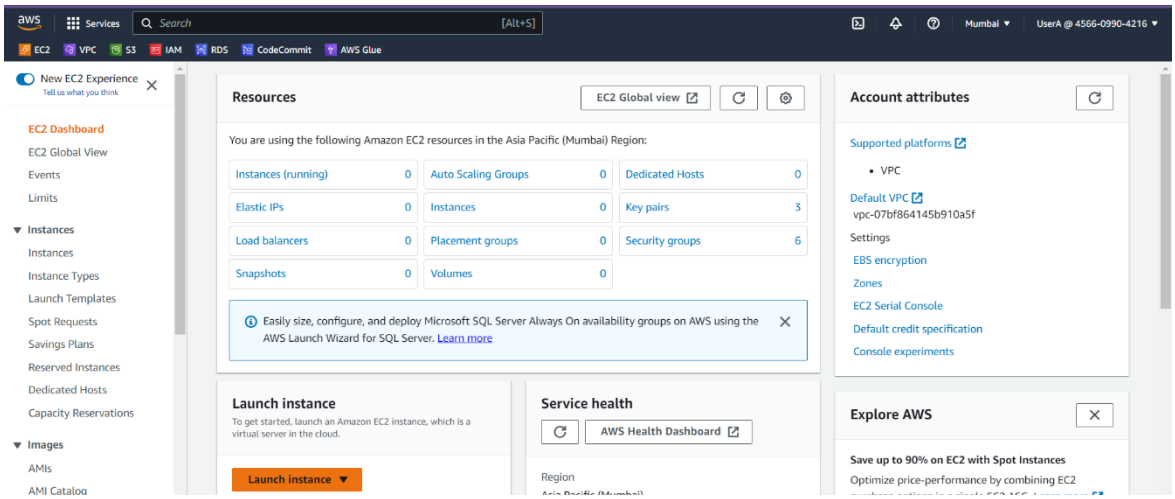


The bucket is empty as we have not uploaded any objects to the bucket

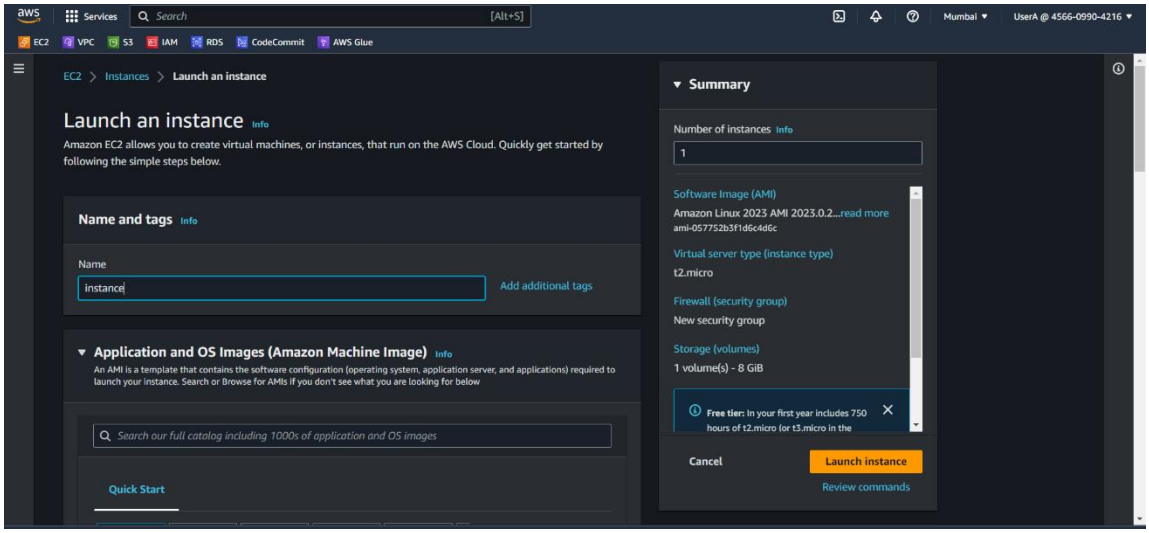


Step:06 Upload the csv to S3 Bucket using EC2 Instance Connect

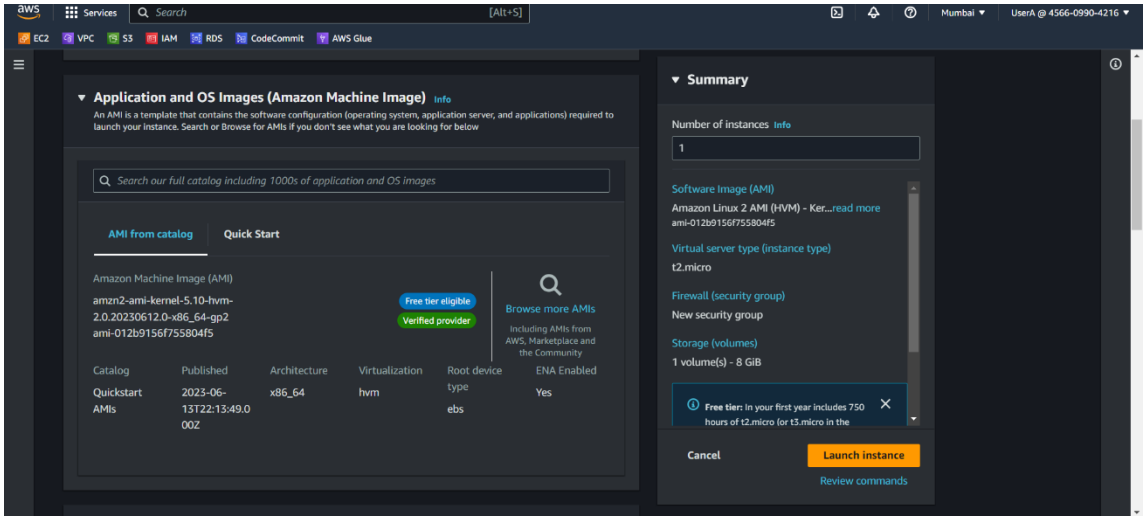
Goto EC2 Dashboard and click on launch instance



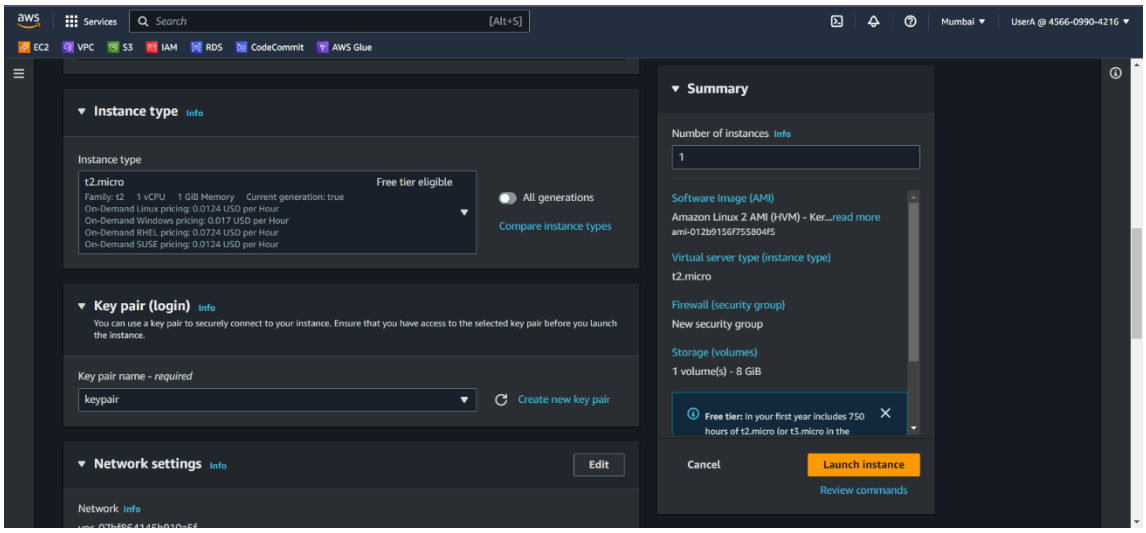
Goto EC2 > Instances > Launch Instance then specify name of the instance



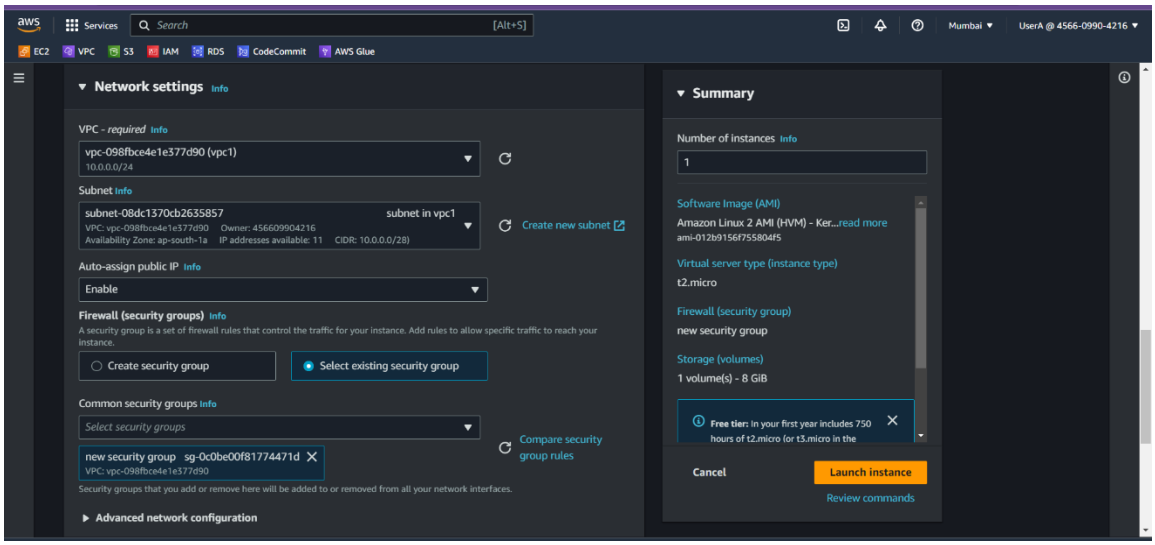
Select Amazon Linux 2 AMI



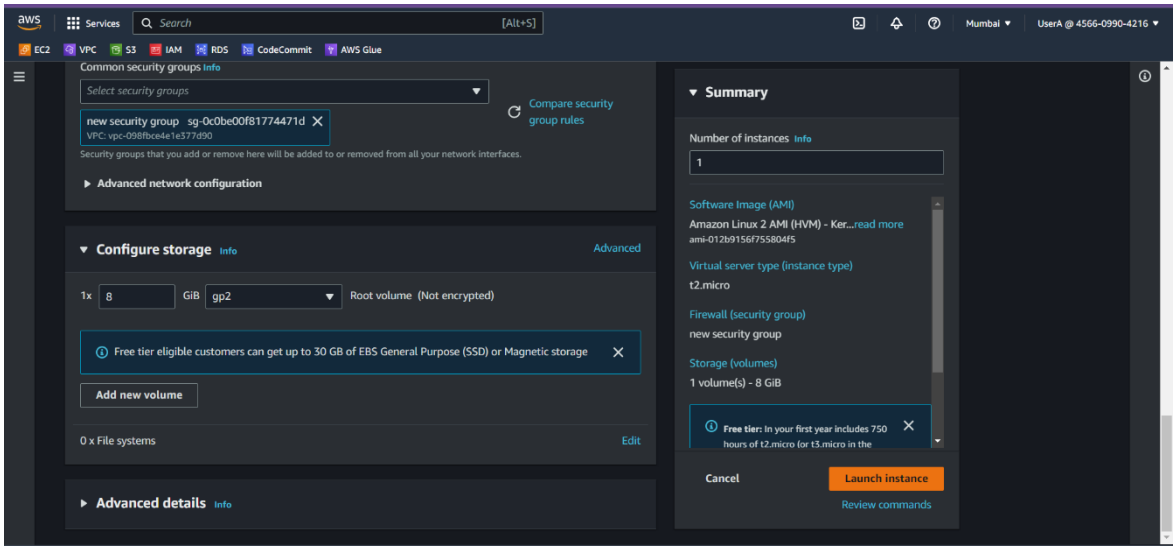
Select the instance as t2.micro and a key pair you have already created



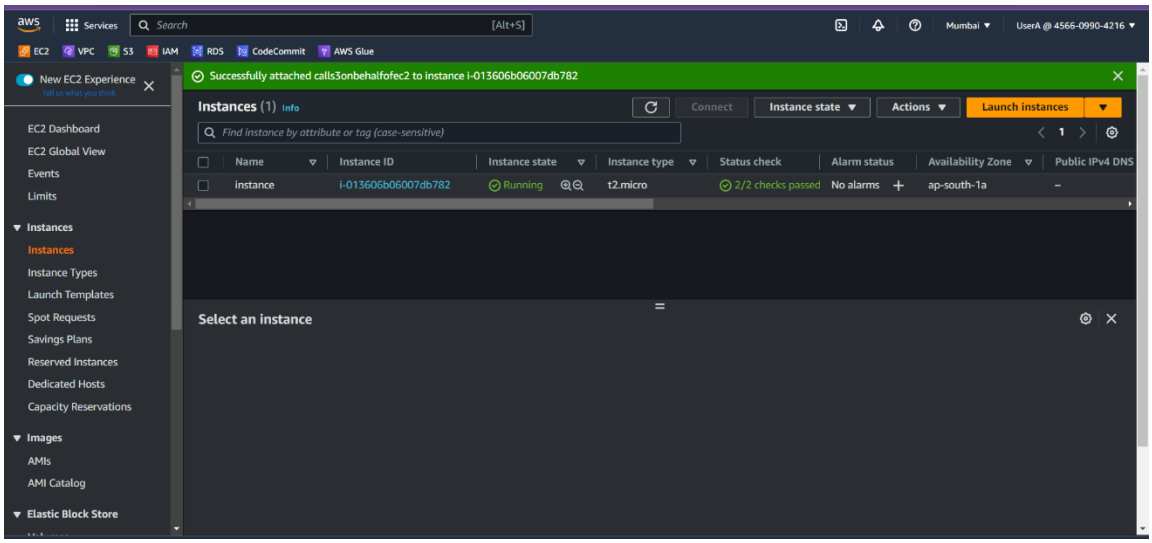
Select the custom vpc and subnet and enable auto assign public IP



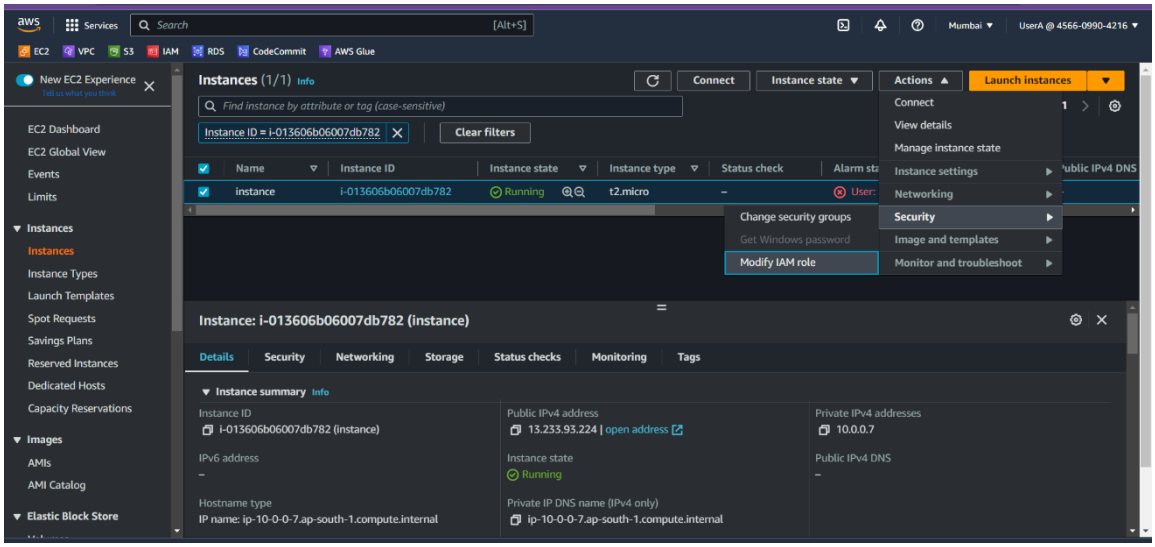
Select the existing security group and configure the storage(root volume)



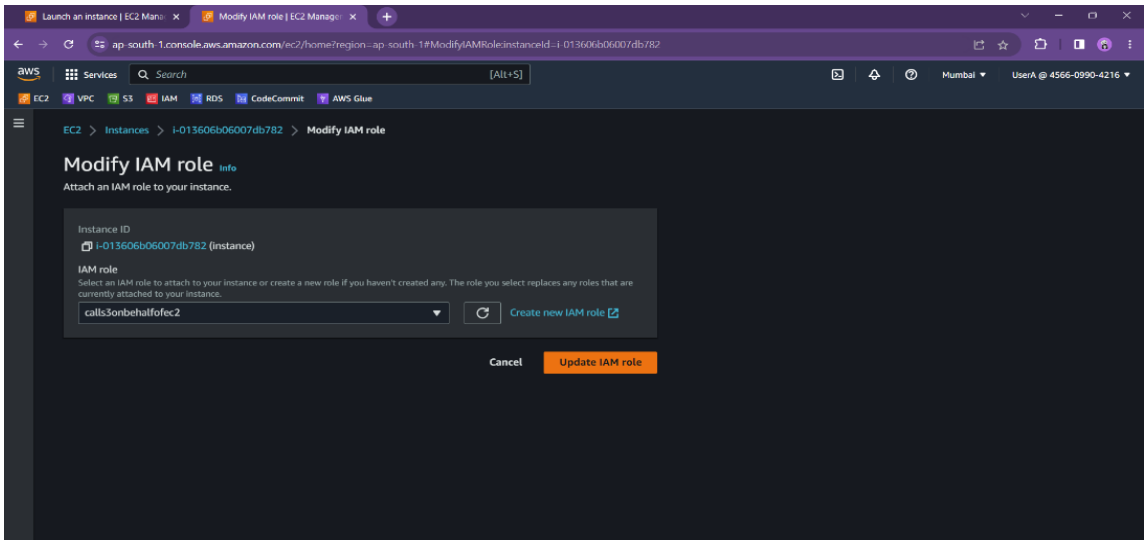
Now an instance has been launched successfully



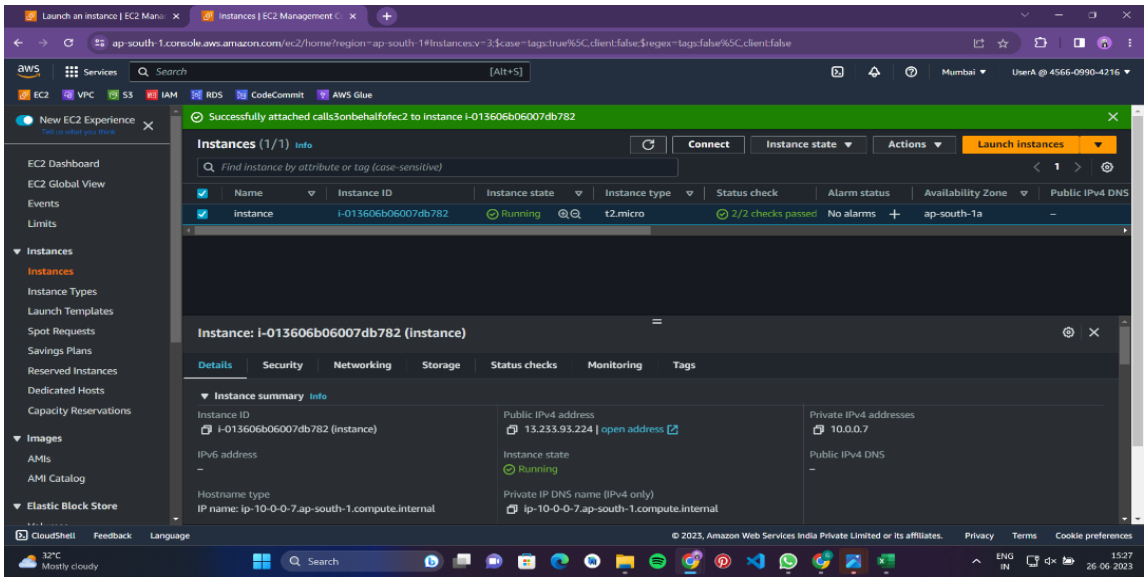
Go to Actions > Security > Modify IAM Role



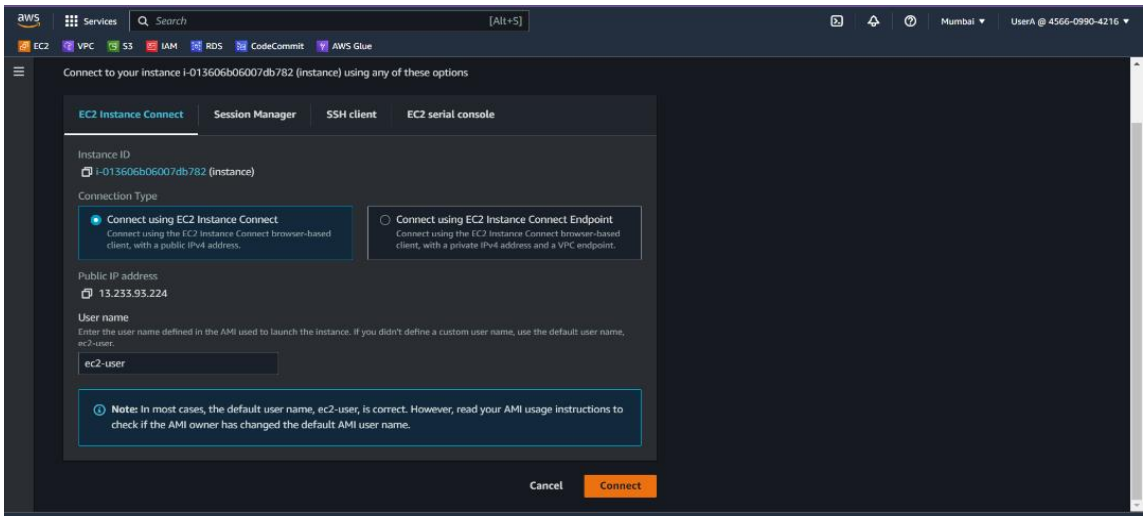
Select the already created role and update IAM Role



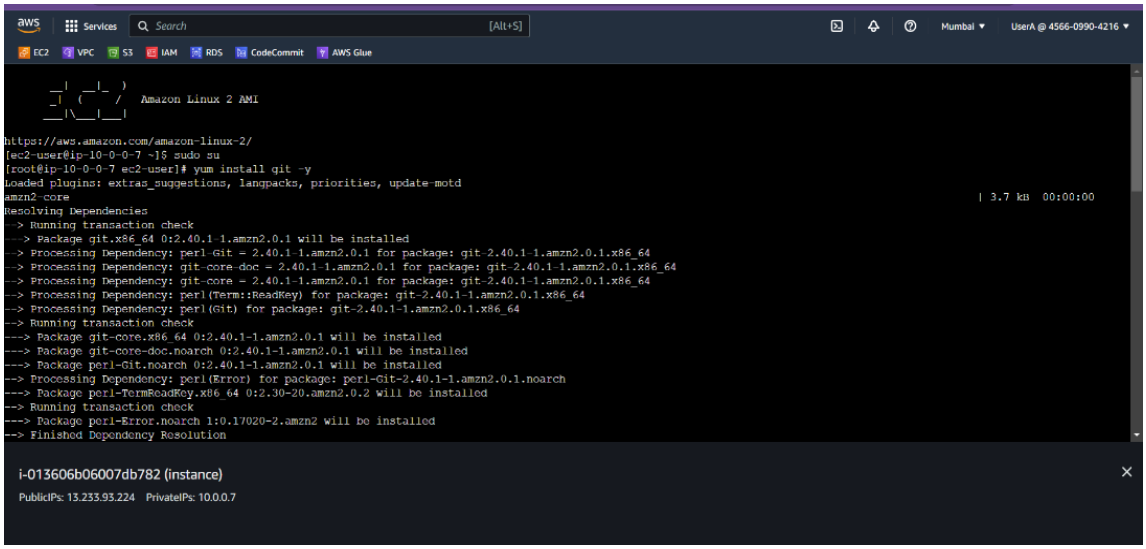
Now the IAM role has been attached to the running instance successfully



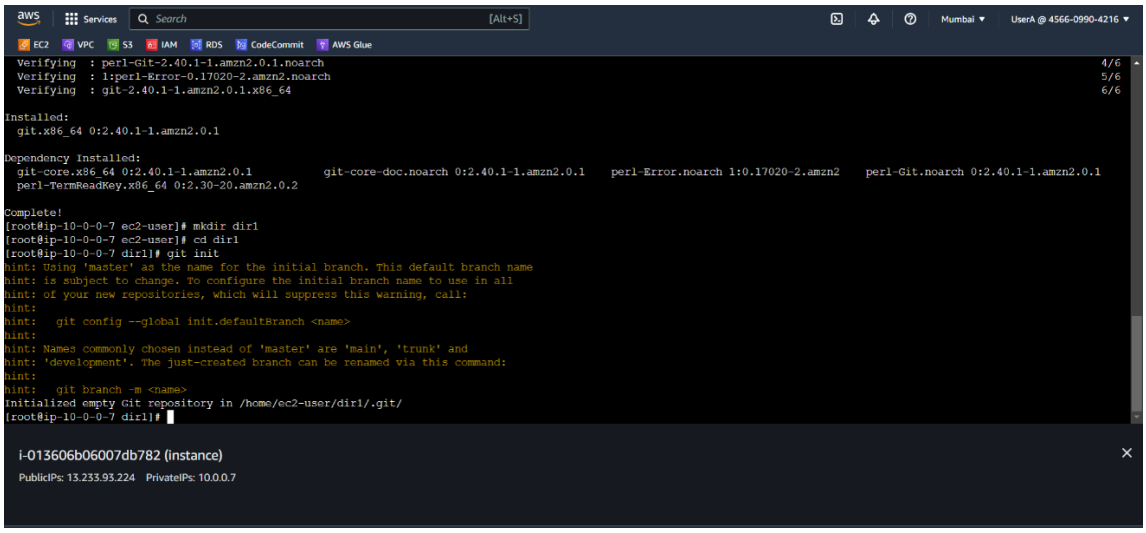
Now connect the instance using EC2 Instance Connect



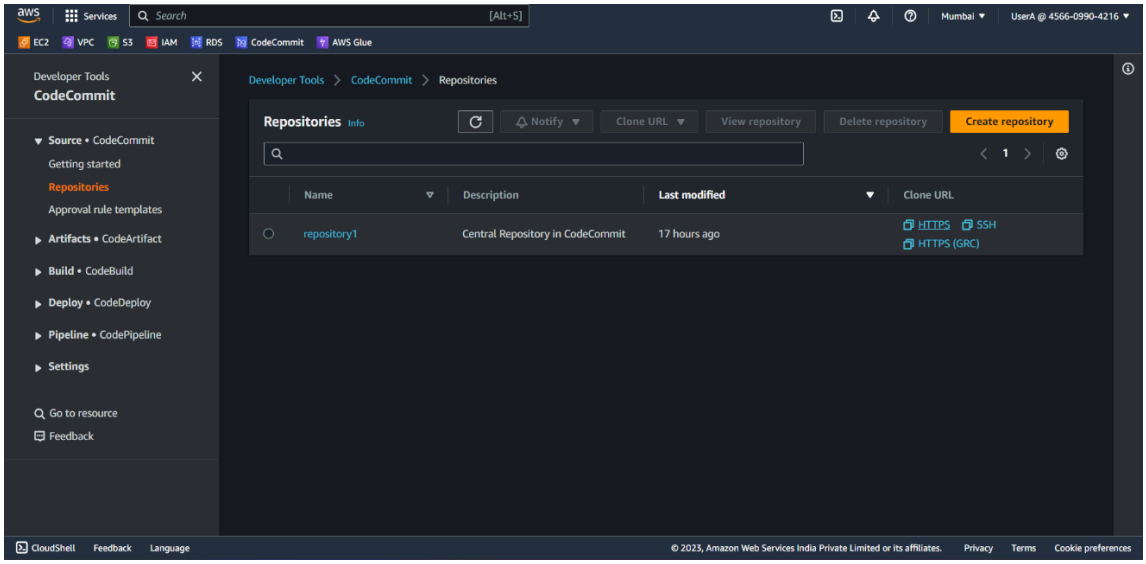
Instance has been connected successfully



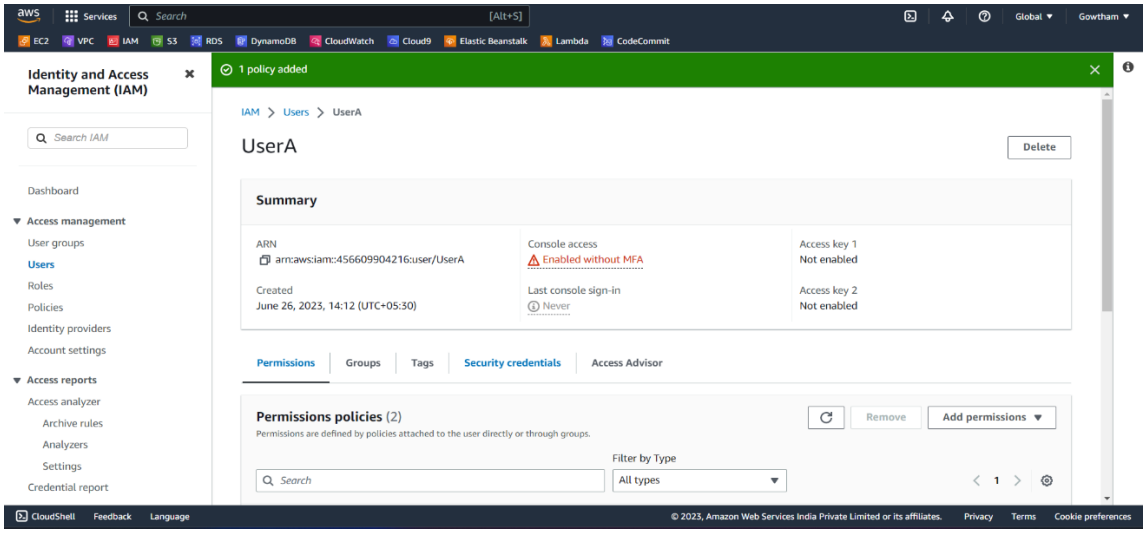
Make a directory and initialize the git



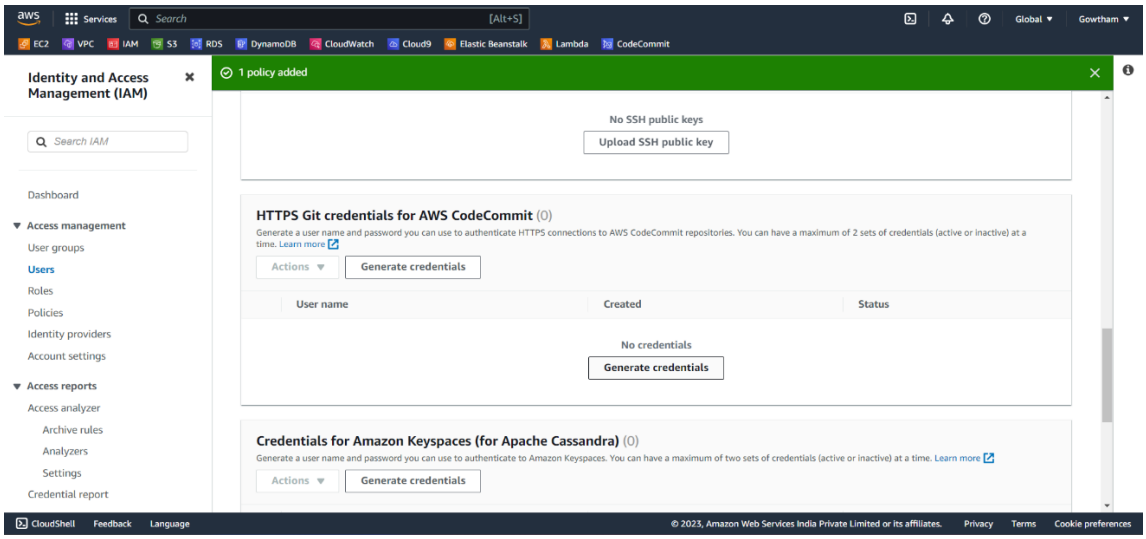
Go to the repository that we have created copy the clone url



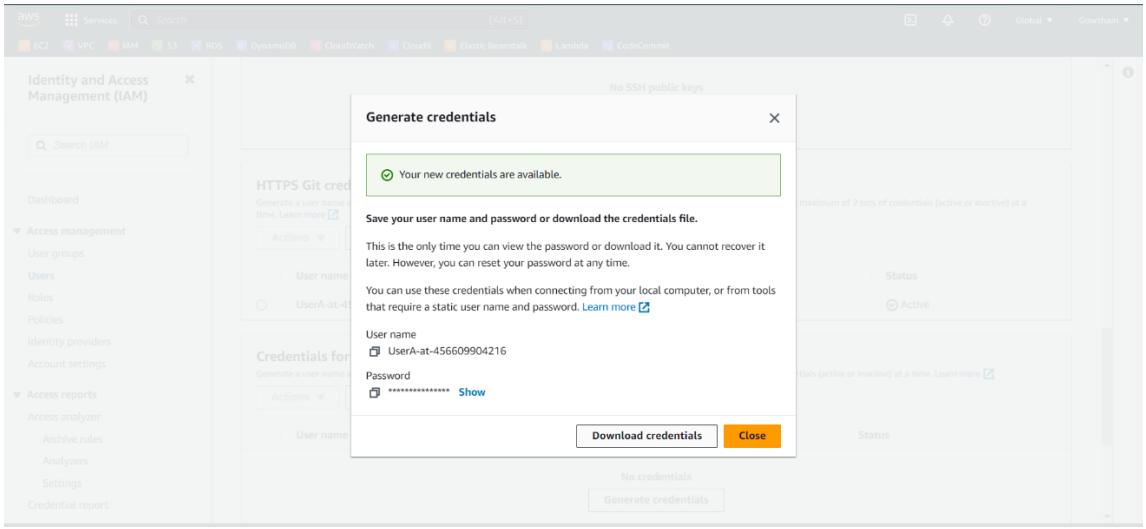
In the root user account go to IAM > Users > User A and click on security credentials



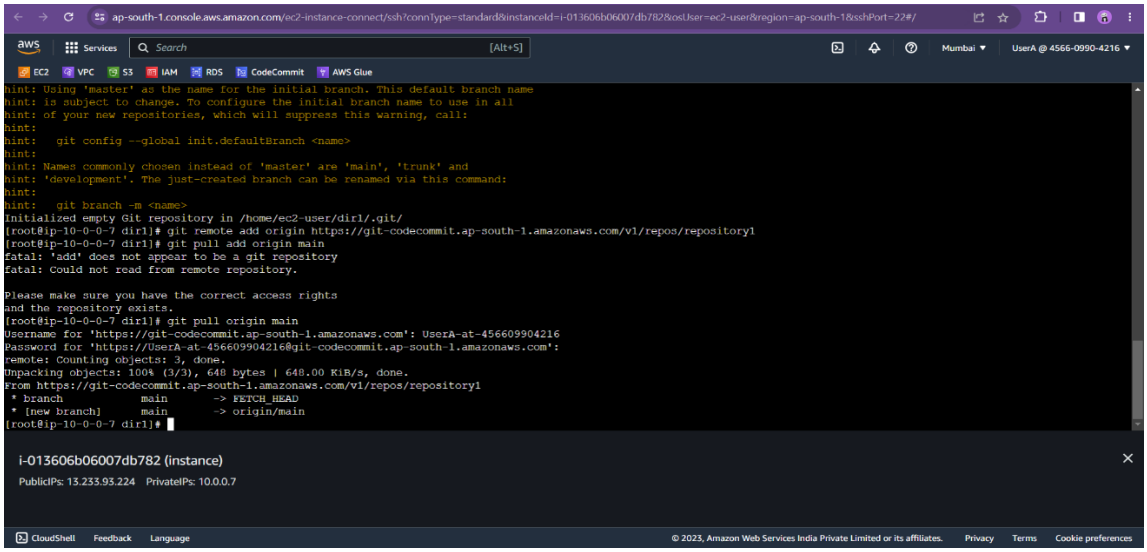
Generate the credentials to access the AWS CodeCommit



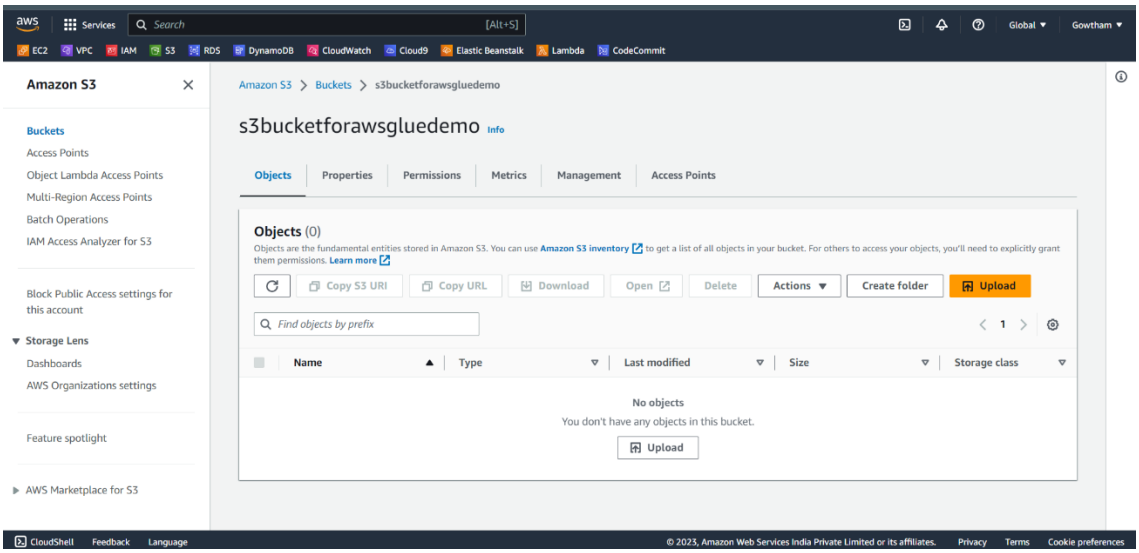
Now download the generated credentials



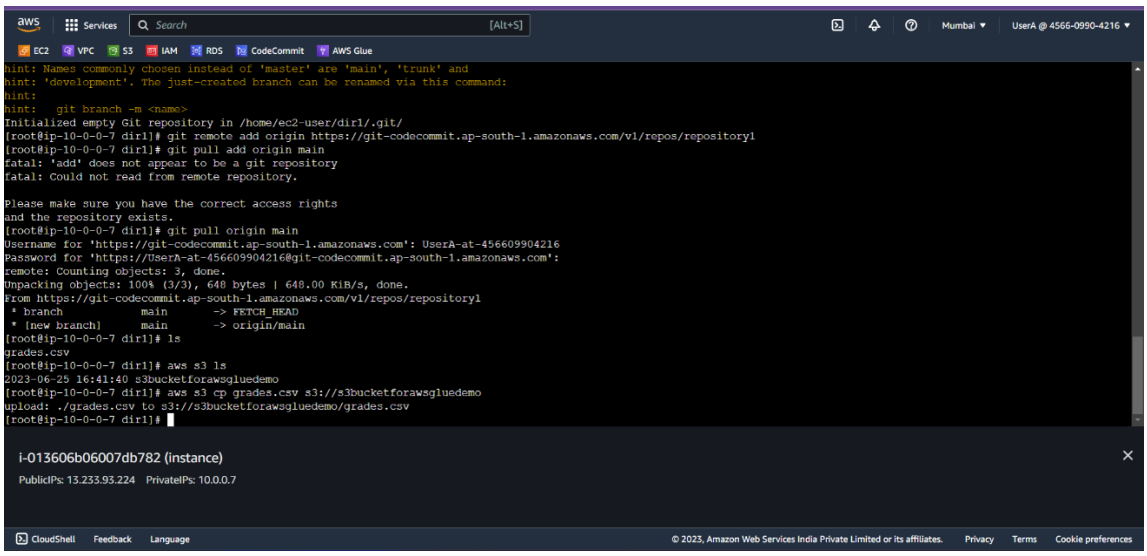
Using the copied clone url and code commit credentials pull the repository into instance



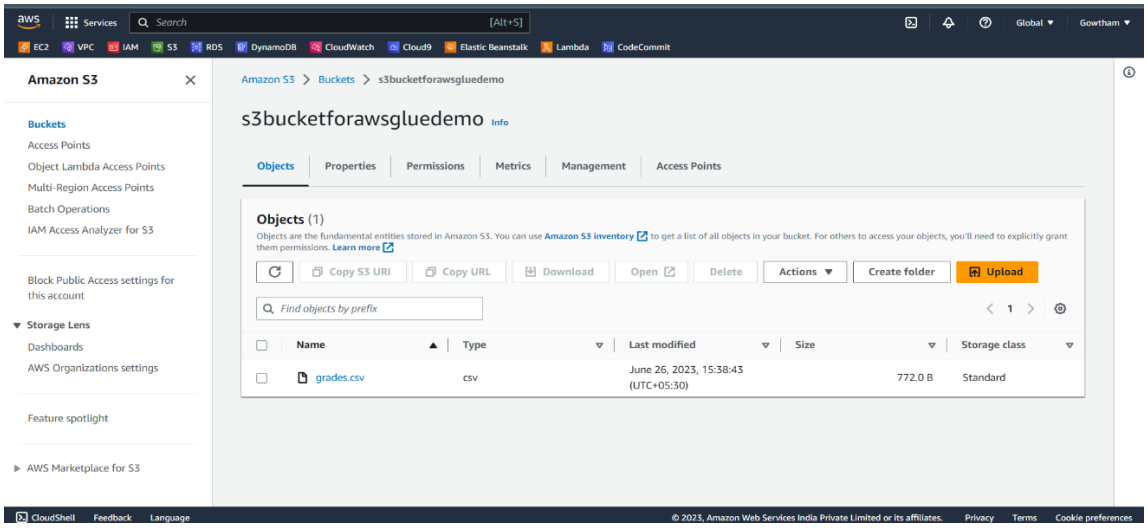
The bucket “s3bucketforawsgluedemo” has no objects



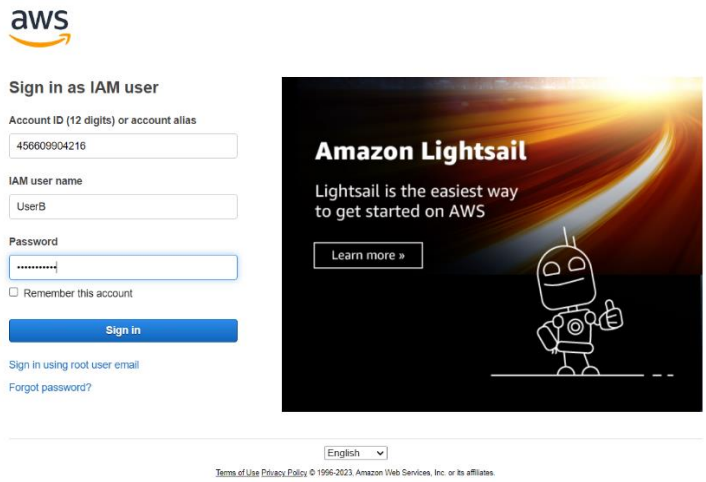
Copy the csv file to s3 bucket using the command aws s3 cp grades.csv s3://s3bucketforawsgluedemo



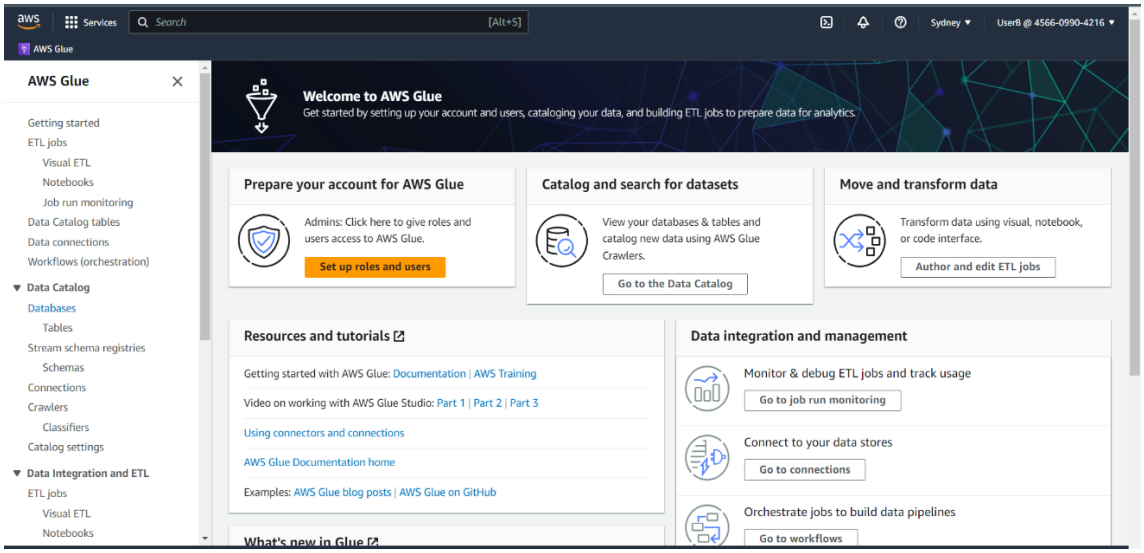
The same csv file in the repository is now in the S3 bucket



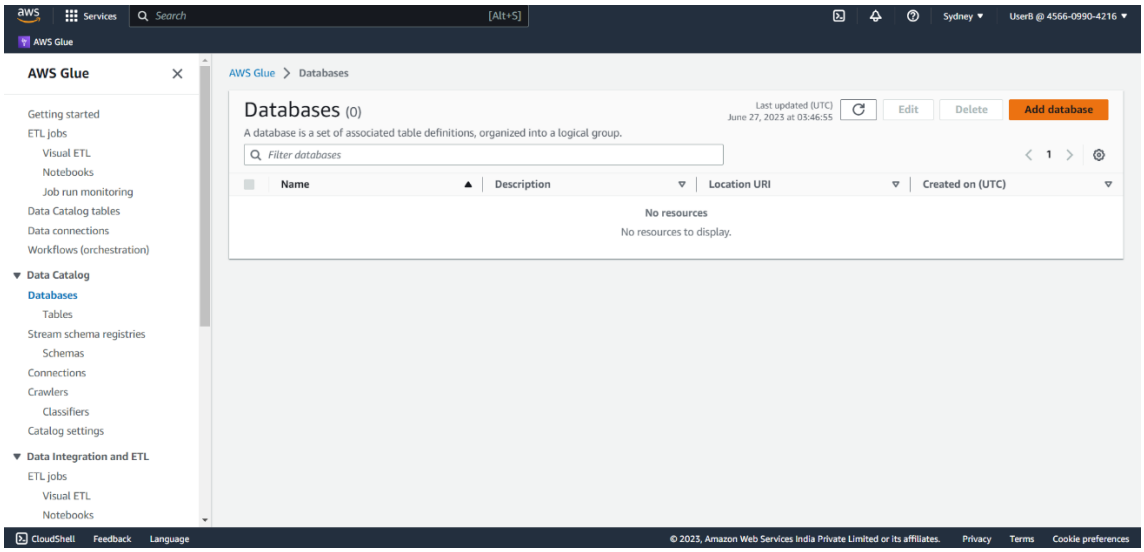
Log in to the IAM User-User B



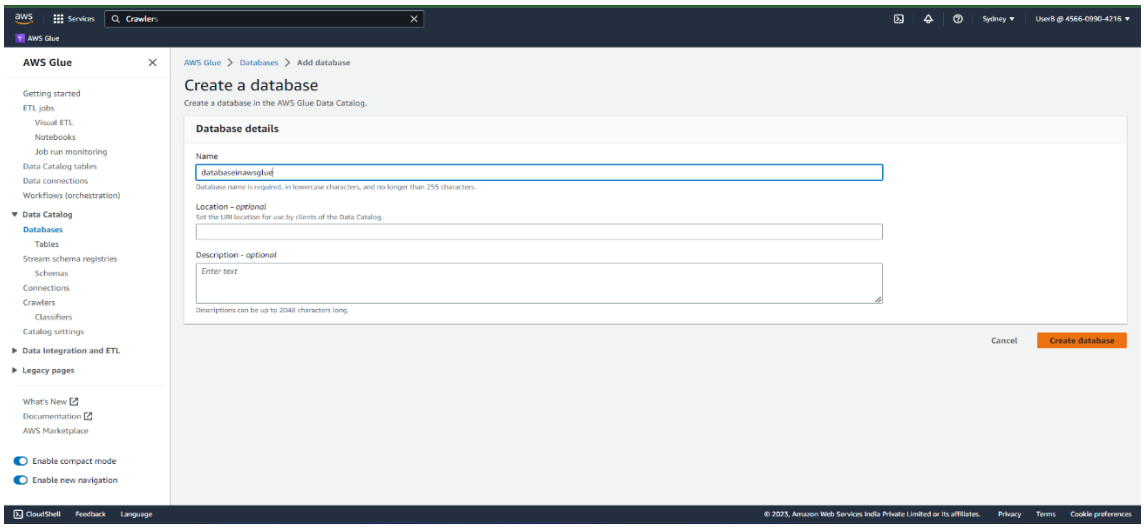
Goto the AWS Glue and click on Databases



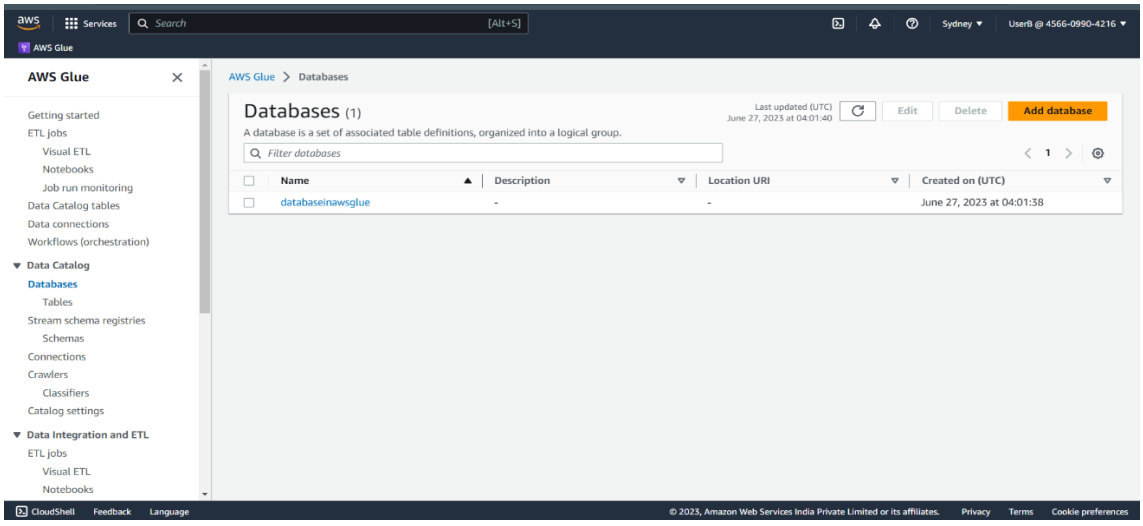
Now click on add database



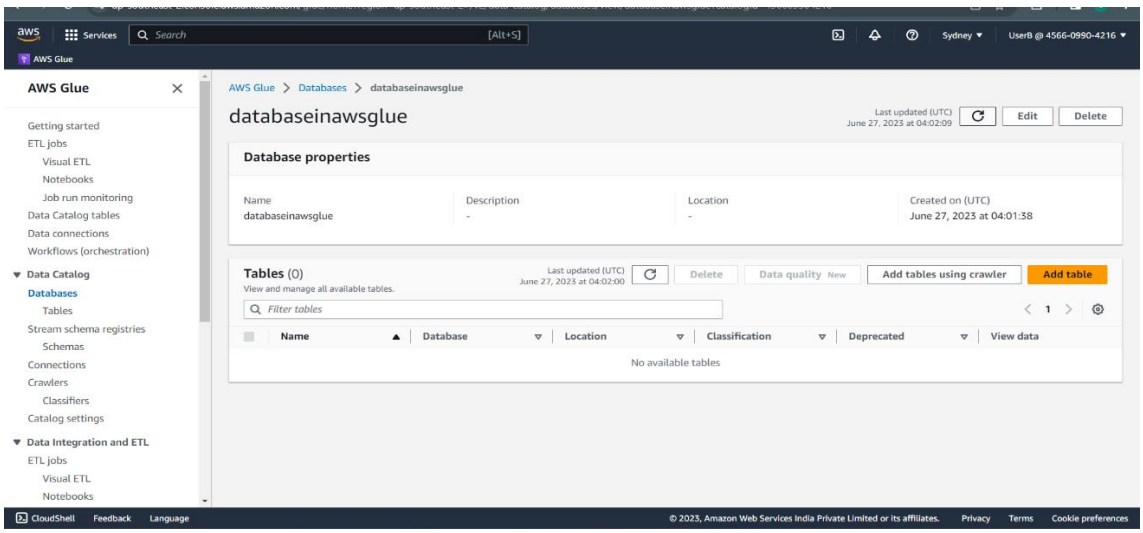
Give the name as “databaseinawsglue” and click on create database



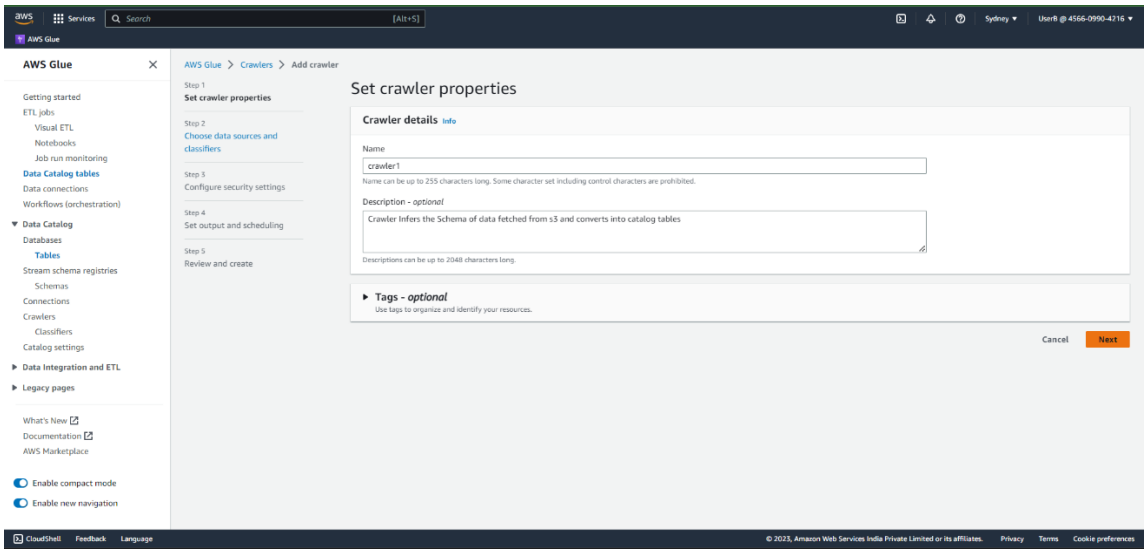
The database has been created successfully



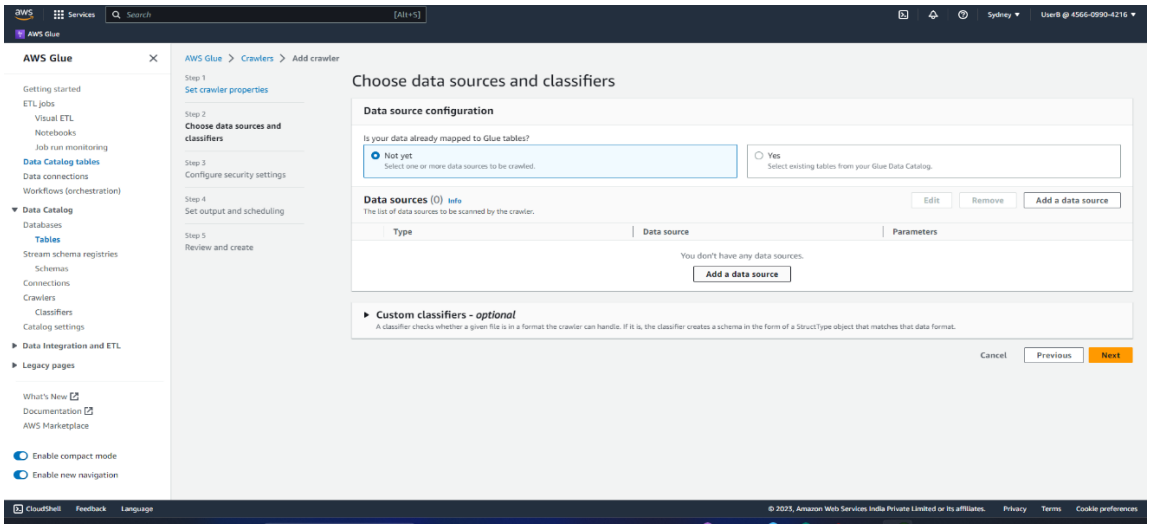
Now click on created database and then on Add tables using crawler



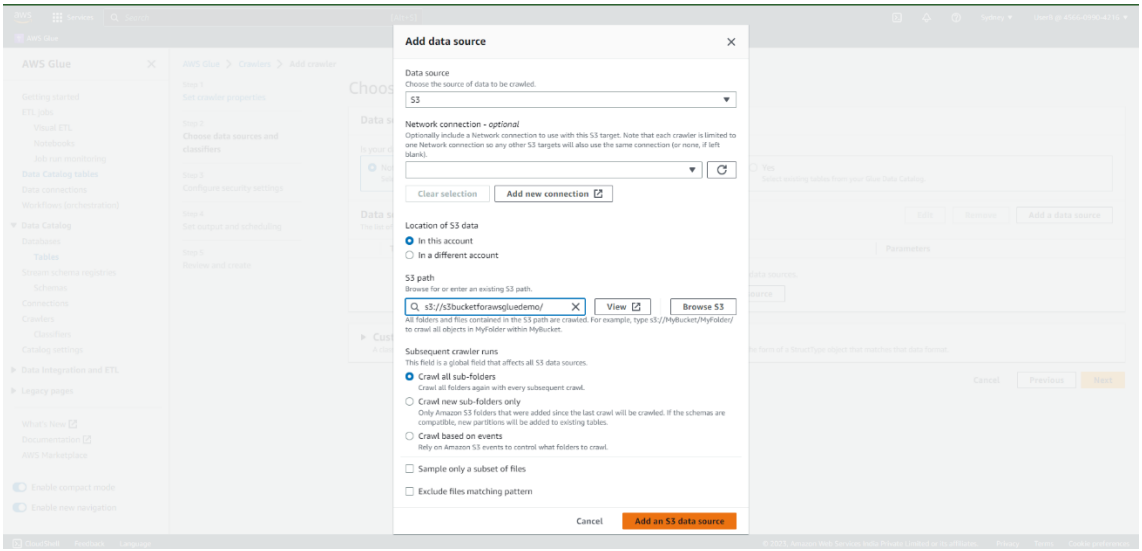
Enter the crawler name and click on next



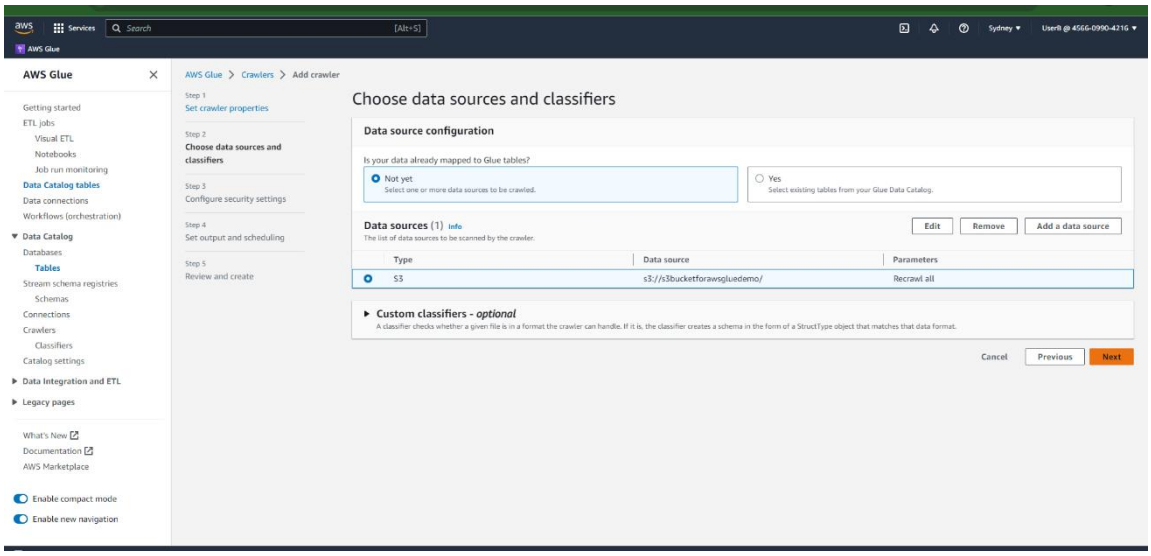
Click on Add a data source



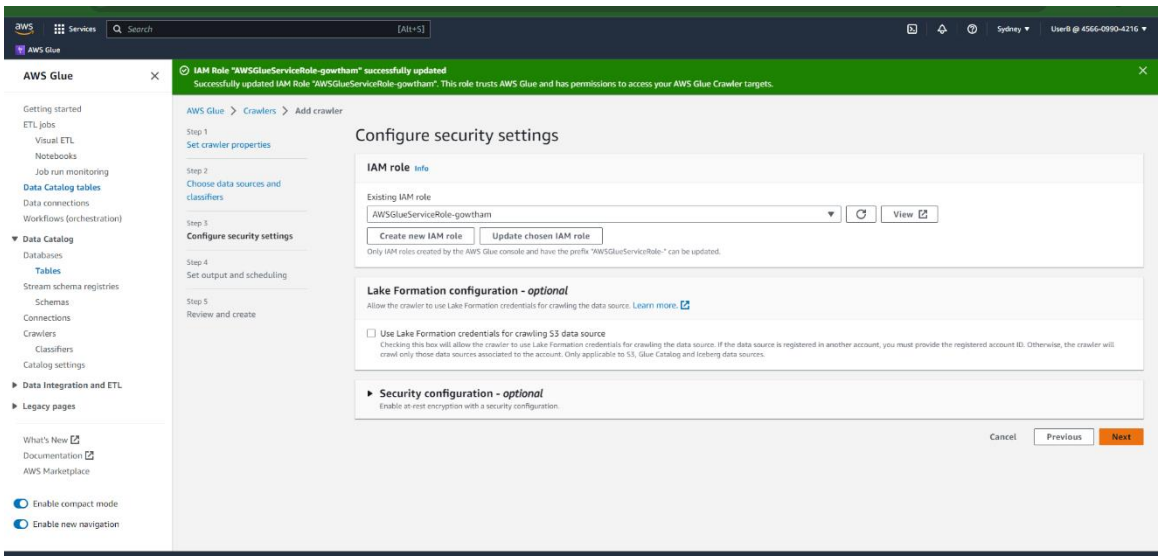
Select the data source as S3 and specify the S3 path



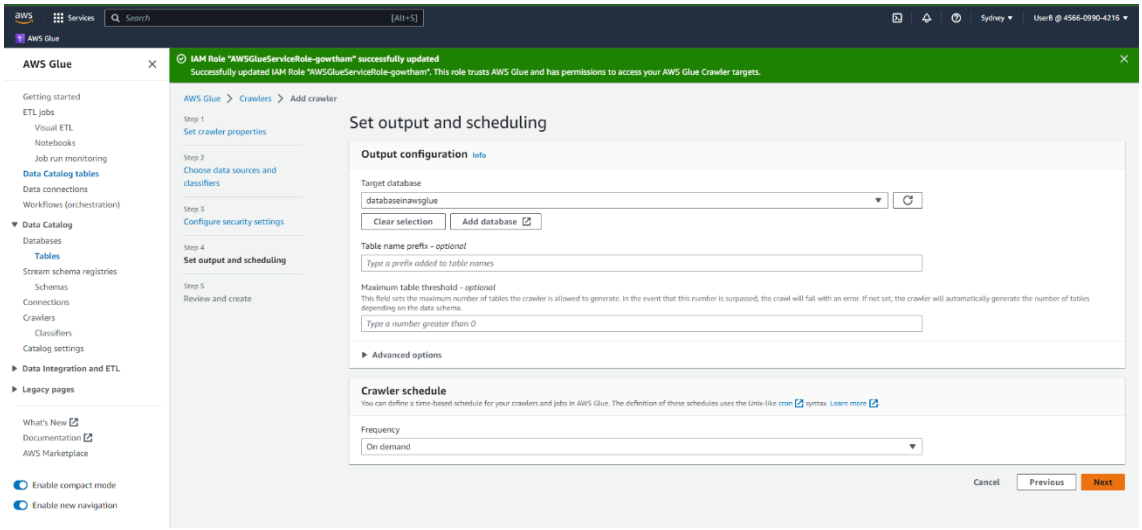
After clicking on Add an S3 data source the mentioned S3 and its path will be shown under Data Sources



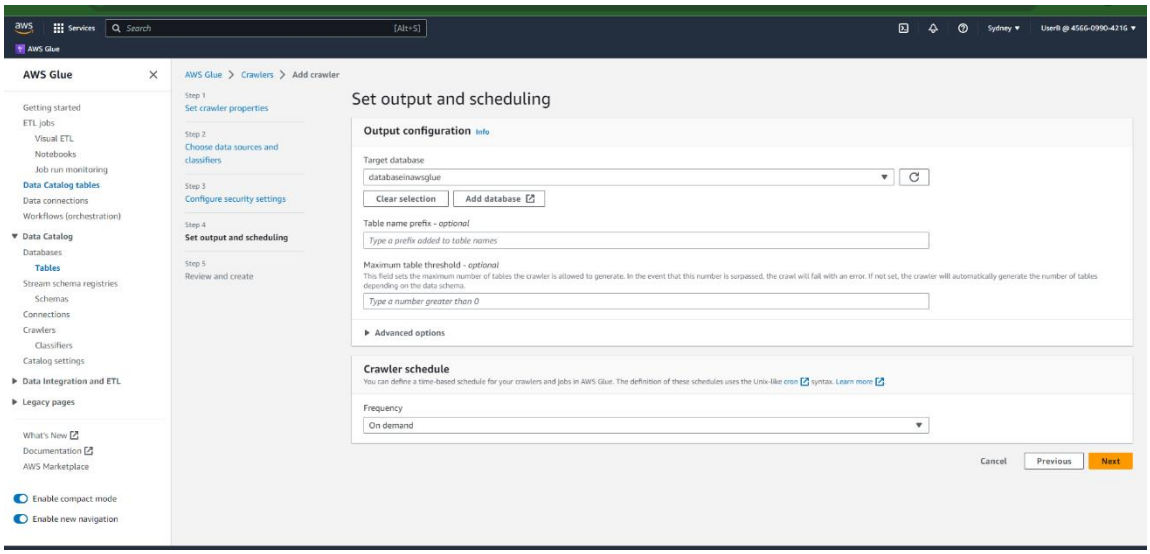
Configure the security settings by creating a new IAM Role



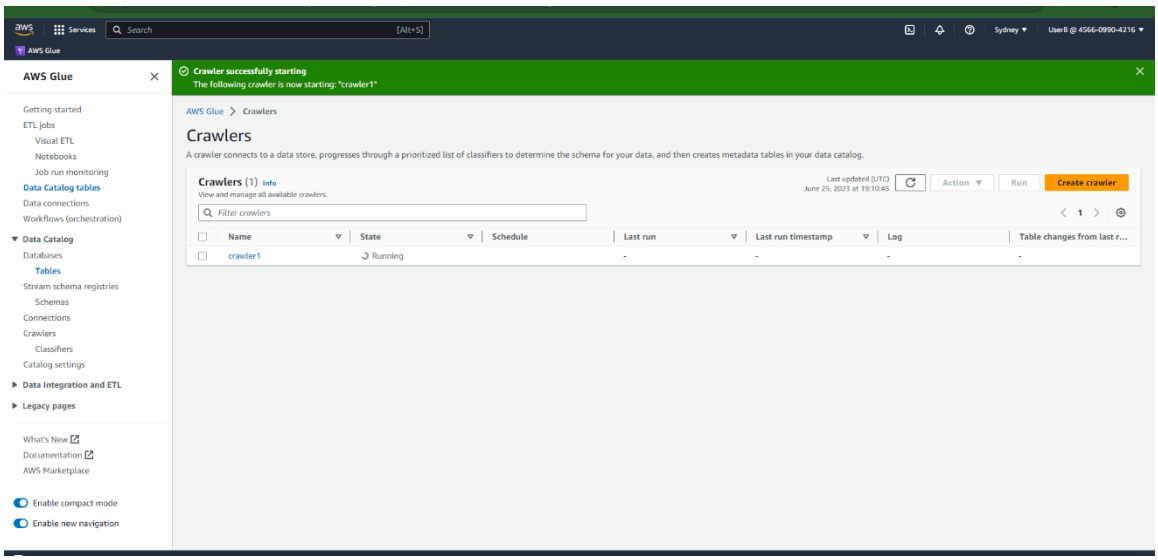
IAM role has been successfully updated



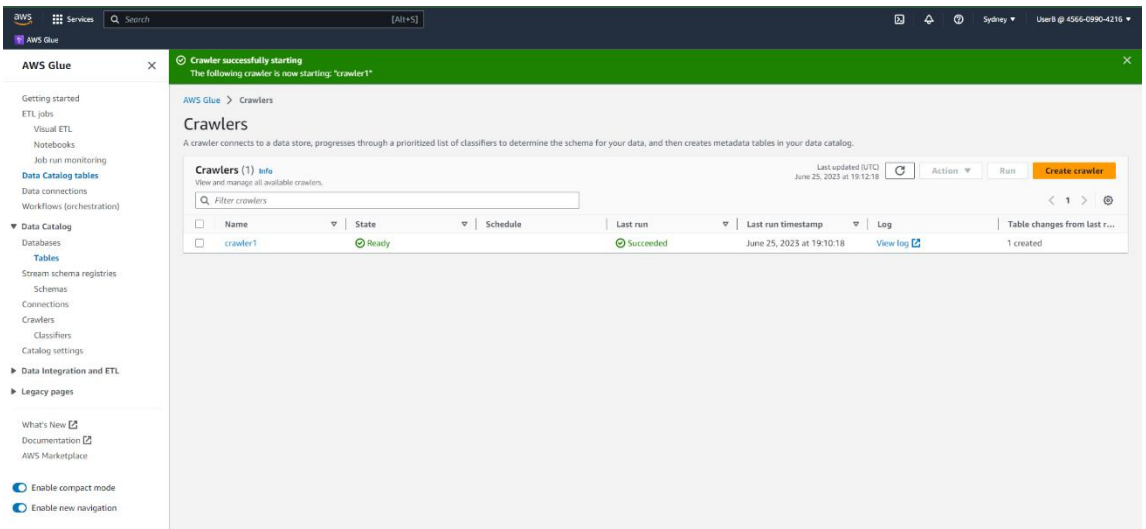
Specify the target database and the crawler schedule



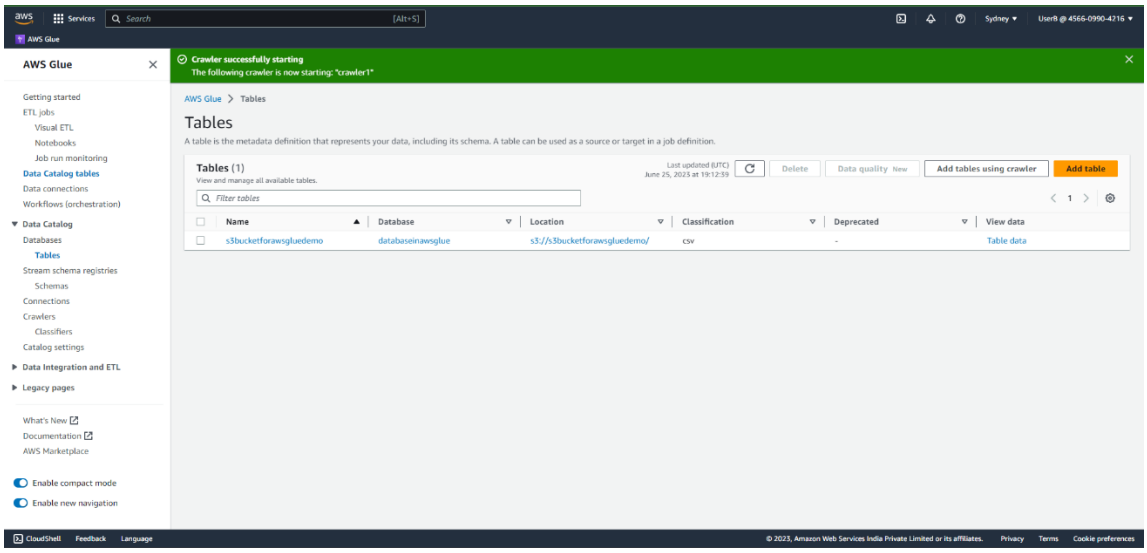
The crawler has started and in the Running state



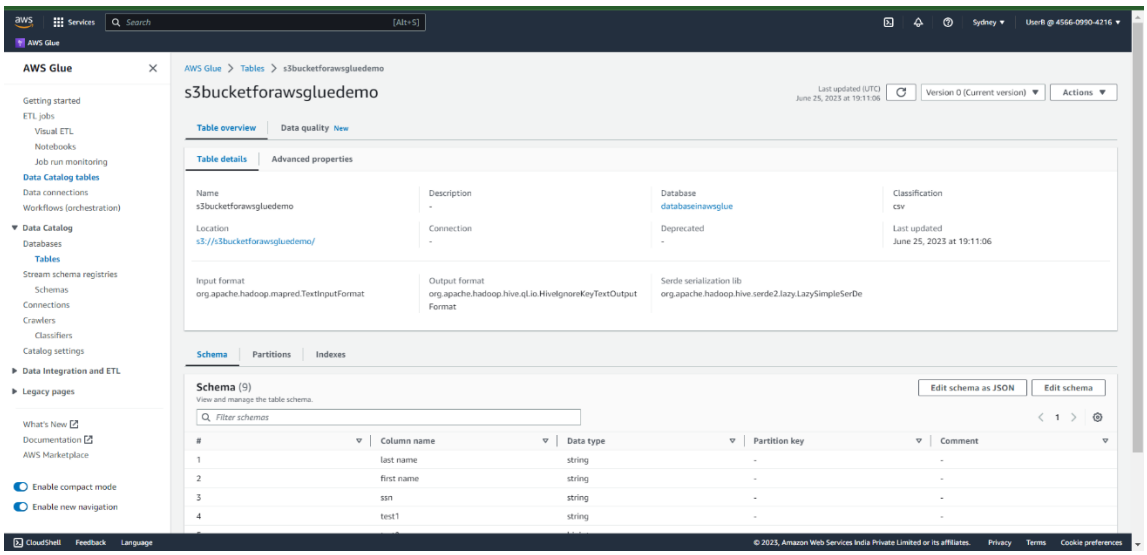
The crawler state changes to the Ready State



Now the data has been fetched from S3 using crawler



Click on the table to view the table details



Now verify the schema of the table which should be same as the content of the csv file