

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

1.Introduction.....	1-2
1.a. Aws Cloud Compute.....	1
1.b Need of Cloud Computing.....	1
1.c Amazon Web Services.....	1
1.d Why AWS?.....	3
2.Architecture	4-5
2.a Description	4
2.b Architecture.....	4
2.c Workflow	5
3.Services included in architecture	6-18
3.a. Amazon VPC	6
3.b. Amazon EC2.....	7
3.c. Amazon S3.....	8
3.d. AWS CodeCommit.....	10
3.e. AWS Glue.....	11
3.f. AWS IAM Role.....	12
Other popular services.....	13
3.g. Amazon RDS	13
3.h. Amazon DynamoDB	14
3.i. Amazon SNS	16
3.h. Amazon Lambda.....	17
4. Implementation.....	19-51
4.a. Custom VPC Creation	19
4.b. Creation of IAM Users.....	26
4.c. Creation of IAM Role For S3FullAccess	31
4.d. Upload the CSV File to the CodeCommit Repository.....	33
4.e. Creation of S3 Bucket to Store the CSV File	36
4.f. Upload the CSV to S3 Bucket Using EC2 Instance Connect.....	39

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:

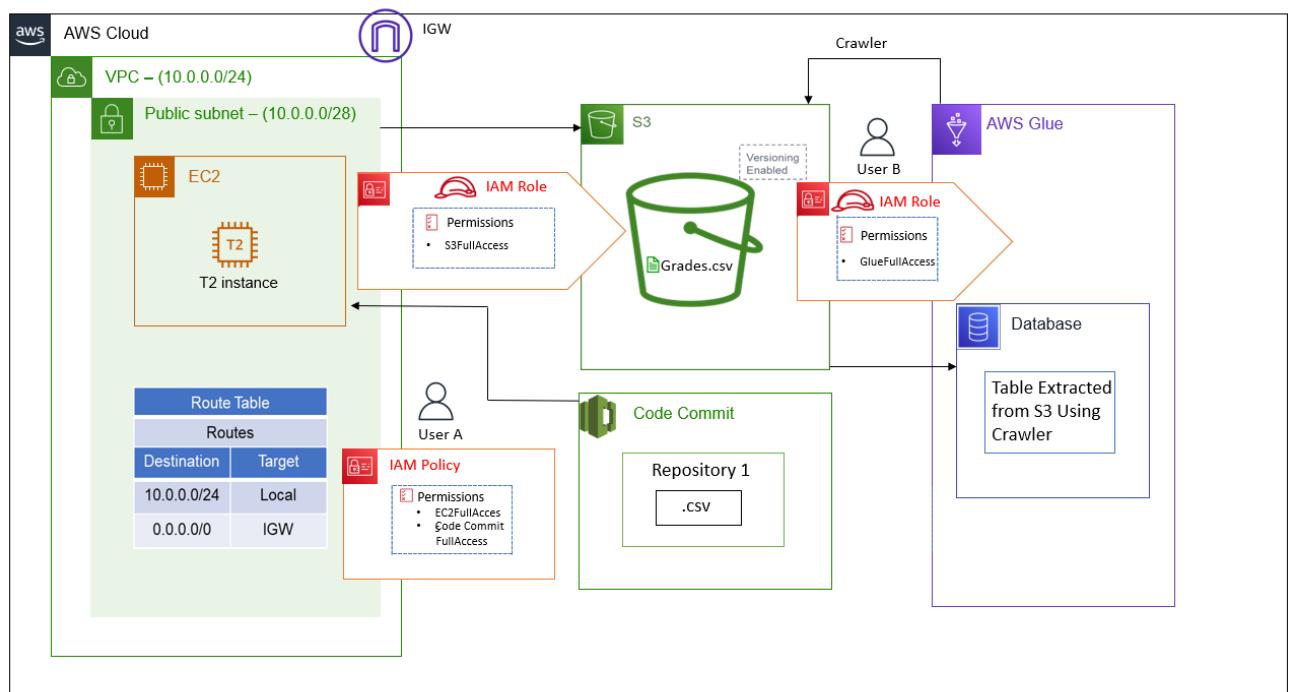
Transferring CSV File from CodeCommit to S3 and Creating a Table in AWS Glue using Crawler

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

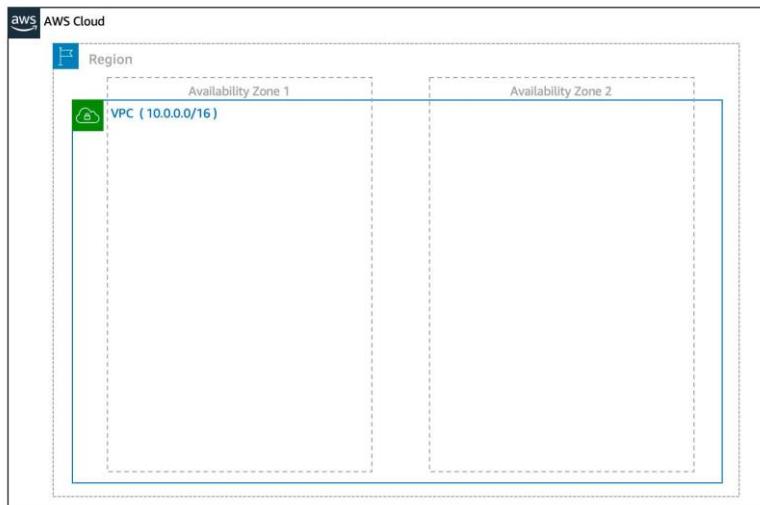
Workflow:

- ❖ 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):



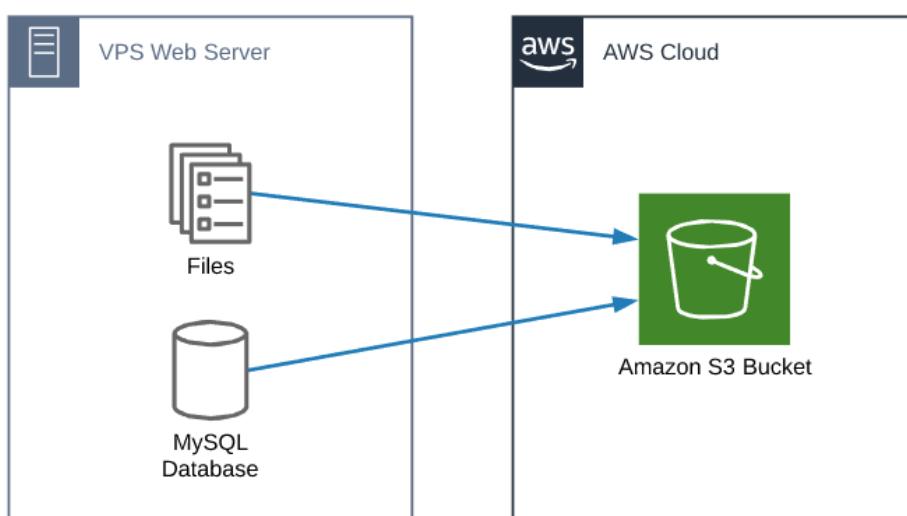
Amazon Elastic Compute Cloud (Amazon EC2) is a web service provided by Amazon Web Services (AWS) that offers scalable computing capacity in the cloud. It allows you to provision virtual servers, known as instances, and run your applications on them. EC2 instances can be easily scaled up or down based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



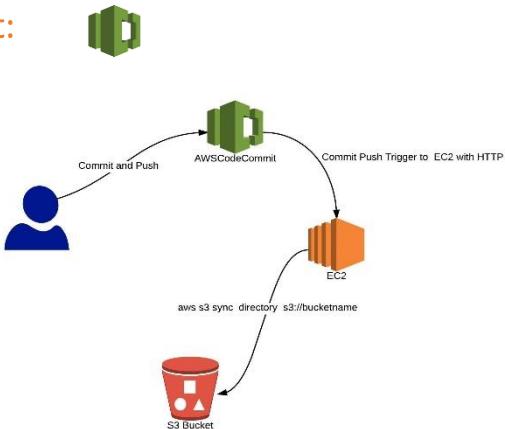
Amazon Simple Storage Service (S3) is a highly scalable and secure object storage service provided by Amazon Web Services (AWS). It allows businesses to store and retrieve vast amounts of data over the internet. S3 provides durability, availability, and performance to store and retrieve any amount of 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:

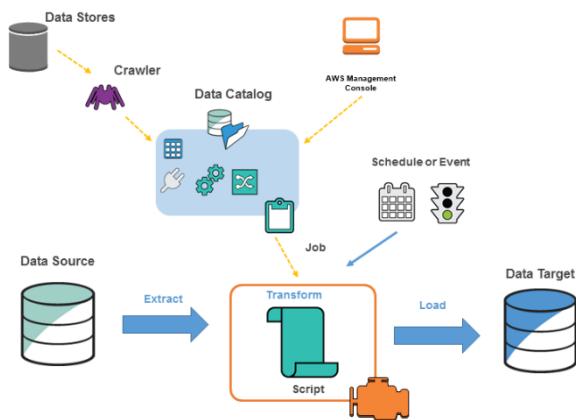


When it comes to code commit components, there are several key elements involved in the process of committing code changes to a version control system. AWS CodeCommit is a fully managed source control service that allows you to securely host private Git repositories in the cloud. It provides a scalable 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.

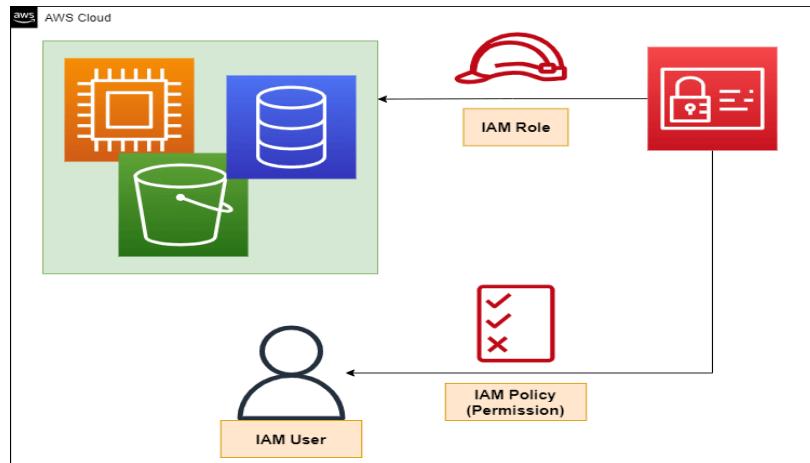
AWS GLUE:



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):



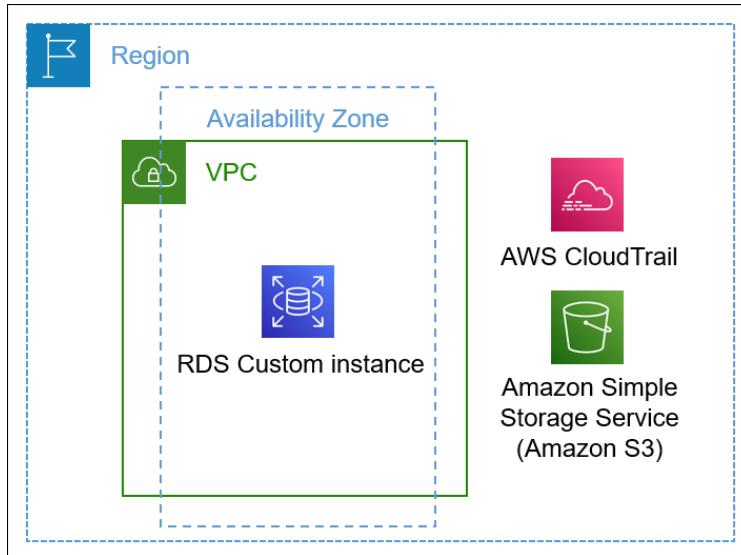
In AWS (Amazon Web Services), IAM stands for Identity and Access Management. It is a service that helps you manage access to AWS resources securely. IAM consists of various components that work together to control user access, permissions, and policies. The main components of IAM in AWS include:

- 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:

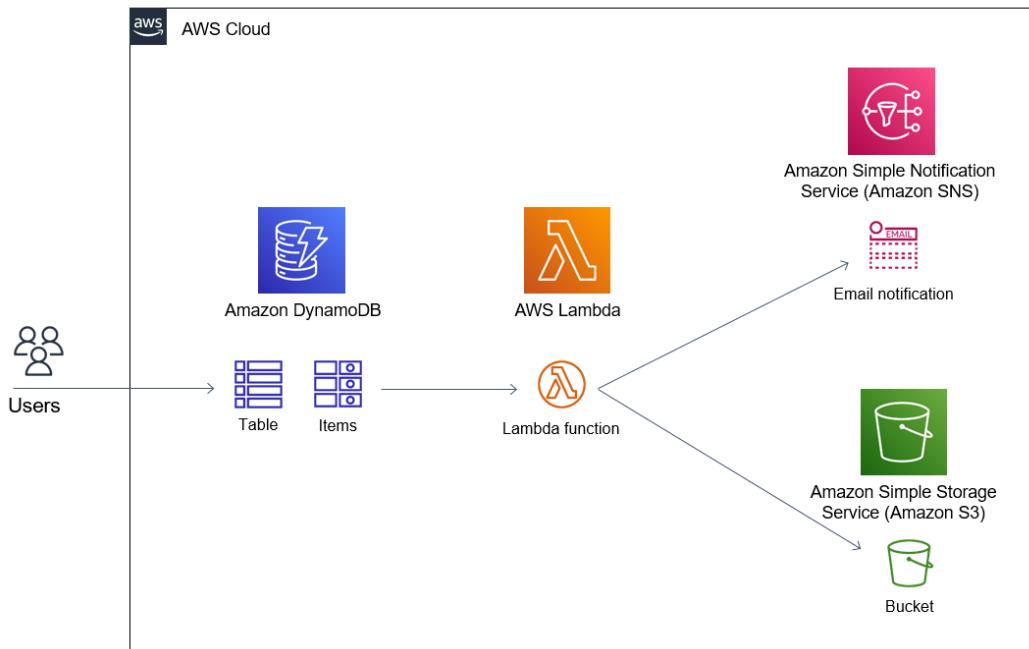
- Database Engines: AWS RDS supports various popular relational database engines, including Amazon Aurora (MySQL and PostgreSQL-compatible), MySQL, MariaDB, PostgreSQL, Oracle Database, and Microsoft SQL Server. Each engine has its own set of features, compatibility, and performance characteristics.
- 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:

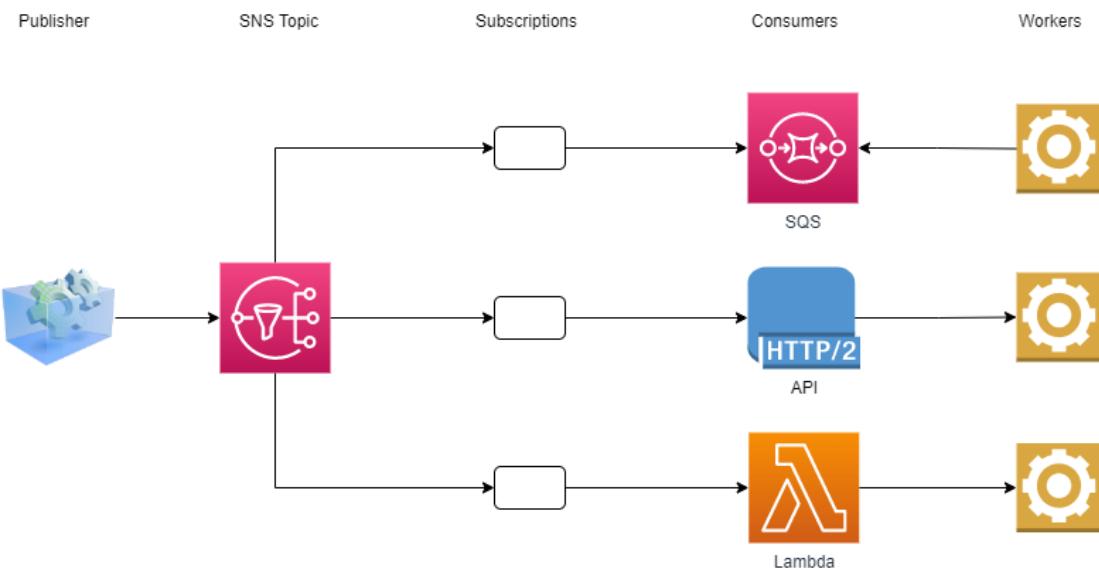


AWS DynamoDB is a fully managed NoSQL database service provided by Amazon Web Services. It is designed to provide fast and scalable storage for applications that require low-latency and high-performance data access. The key components of AWS DynamoDB are as follows:

- 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:

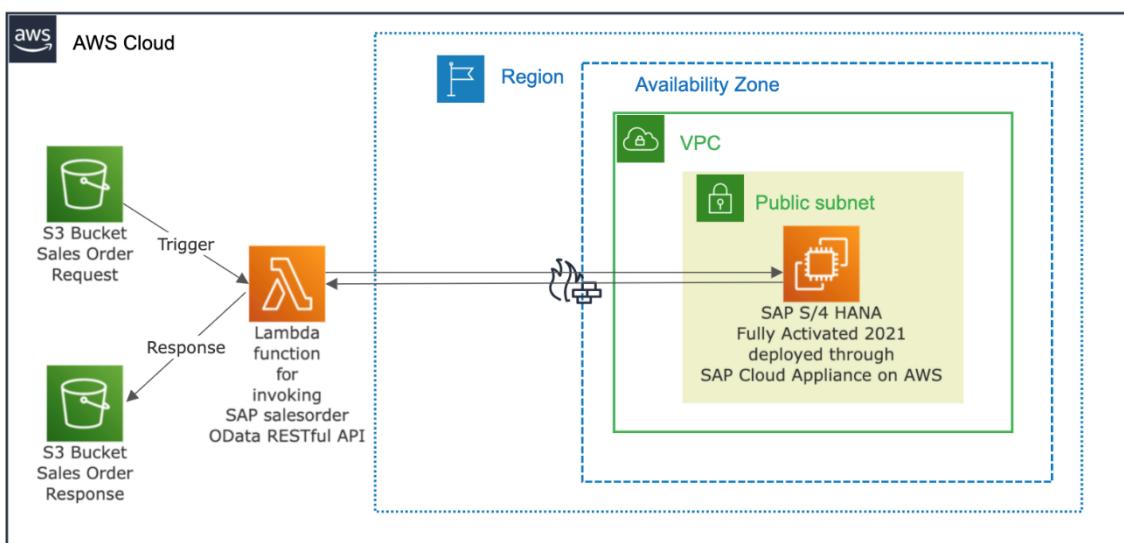
- Publish-Subscribe Model: SNS follows a publish-subscribe paradigm. Publishers send messages to topics, and subscribers who are interested in those topics receive the messages. This decoupled architecture allows for easy scaling and flexibility in message distribution.
- Multi-Channel Messaging: SNS 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. AWS Lambda 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 is a 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 AWS ecosystem 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.

The screenshot shows the AWS Management Console home page. At the top, there's a navigation bar with various service icons: EC2, VPC, IAM, S3, RDS, DynamoDB, CloudWatch, Simple Notification Service, EFS, Cloud9, Lambda, CodeCommit, and Amazon SageMaker. Below the navigation bar is the 'Console Home' section with a 'Recently visited' list containing EC2, EFS, DynamoDB, Amazon SageMaker, Lambda, CodeCommit, and VPC. To the right of this is a 'Welcome to AWS' panel with links for 'Getting started with AWS', 'Training and certification', and 'What's new with AWS?'. At the bottom of the home page, there are sections for 'AWS Health' and 'Cost and usage'.

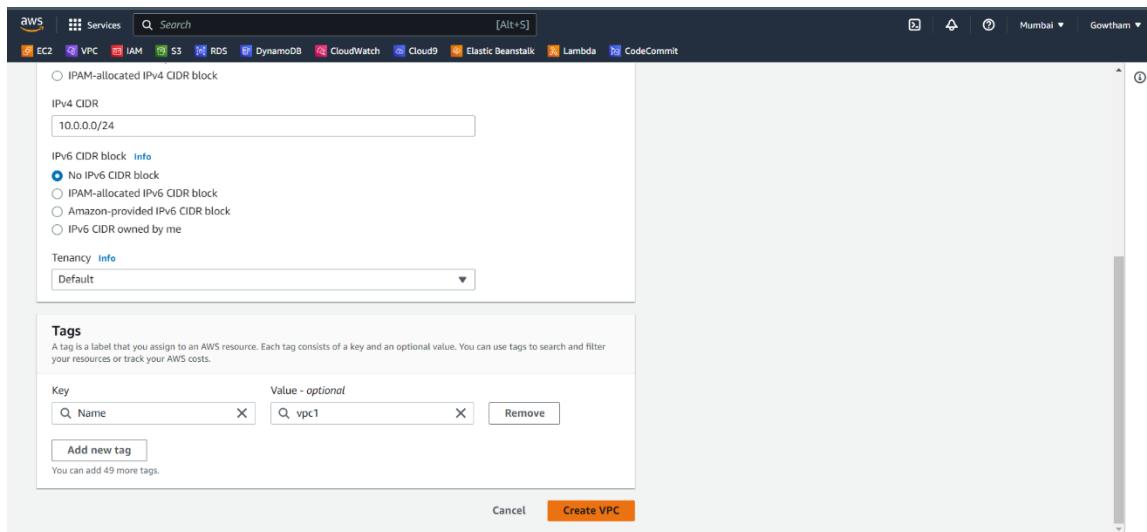
Next Go to VPC and click on create VPC.

The screenshot shows the VPC dashboard. On the left, there's a sidebar with options like 'Virtual private cloud', 'Your VPCs', 'Subnets', 'Route tables', 'Internet gateways', 'Egress-only internet gateways', 'DHCP option sets', 'Elastic IPs', 'Managed prefix lists', 'Endpoints', and 'Endpoint connectors'. The main area has a 'Create VPC' button and a 'Launch EC2 Instances' button. It displays 'Resources by Region' with sections for VPCs, Subnets, Route Tables, Internet Gateways, NAT Gateways, VPC Peering Connections, Network ACLs, and Security Groups. On the right, there are sections for 'Service Health', 'Settings' (with 'Zones' and 'Console Experiments'), 'Additional Information' (with 'VPC Documentation', 'All VPC Resources', 'Forums', and 'Report an Issue'), and 'AWS Network Manager'.

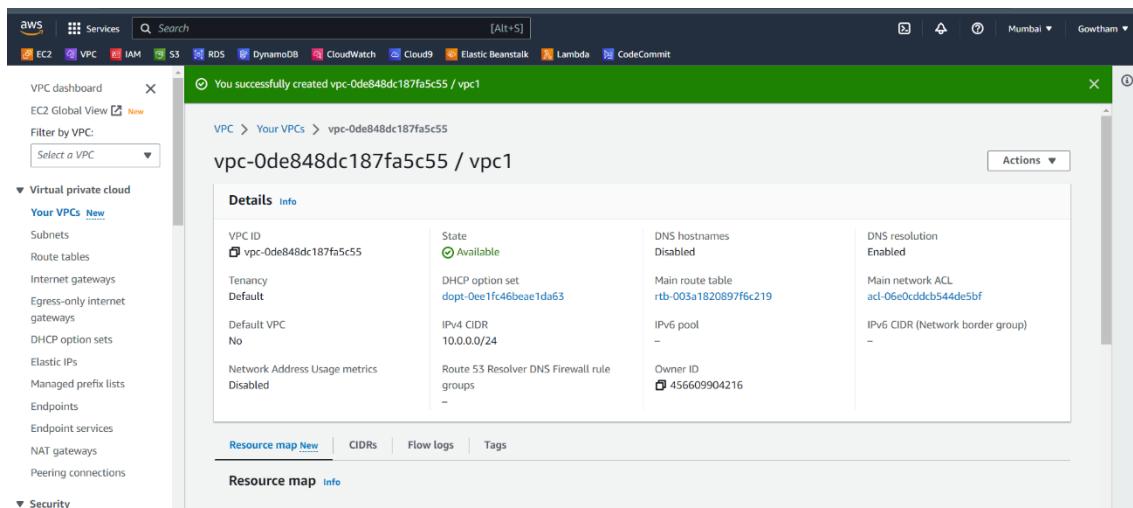
Go to VPC Settings-> Select VPC only, name the custom VPC and give IPV4 CIDR (10.0.0.0/24).

The screenshot shows the 'Create VPC' settings page. The 'VPC settings' section has a 'Resources to create' dropdown where 'VPC only' is selected. There's a 'Name tag - optional' field with 'vpc1' entered. Under 'IPv4 CIDR block', 'IPv4 CIDR manual input' is selected, and the 'IPv4 CIDR' field contains '10.0.0.0/24'. Under 'IPv6 CIDR block', 'No IPv6 CIDR block' is selected. The page also includes sections for 'AWS Network Manager' and 'AWS Lambda'.

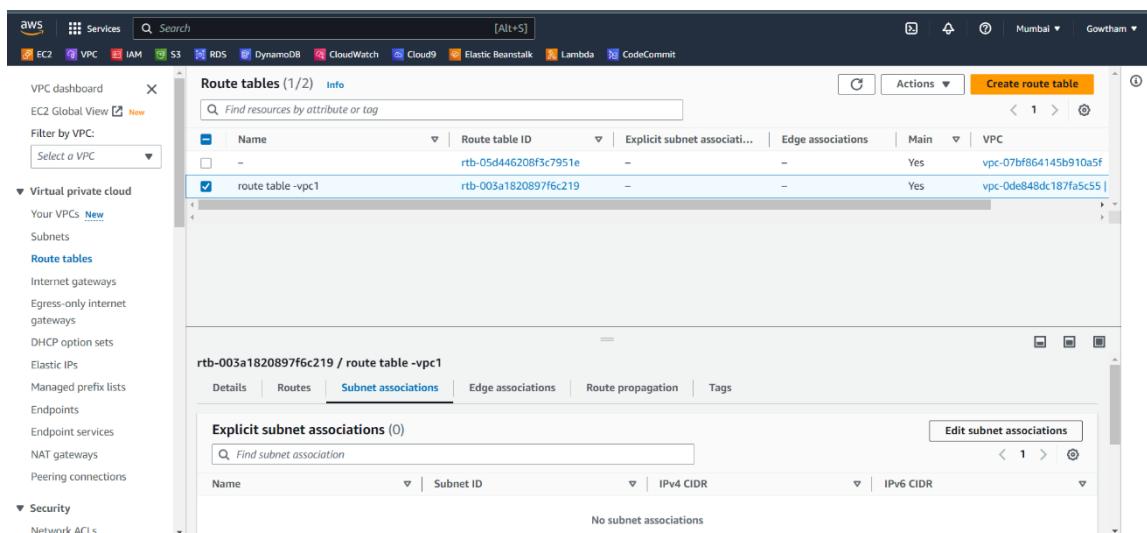
Now Create the custom VPC.



Successfully created custom VPC.



Rename the route table created for custom VPC



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

The screenshot shows the AWS VPC Subnets list page. The top navigation bar includes services like EC2, VPC, IAM, S3, RDS, DynamoDB, CloudWatch, Cloud9, Lambda, and CodeCommit. The left sidebar shows a tree view with 'Virtual private cloud' expanded, showing 'Your VPCs' (1), 'Subnets' (selected), and other options like Route tables, Internet gateways, Egress-only internet gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services, NAT gateways, and Peering connections. The main content area displays a table titled 'Subnets (3) info' with columns: Name, Subnet ID, State, VPC, IPv4 CIDR, and IPv6 CIDR. The subnets listed are: subnet-0d7cc0515d1e7cf6a, subnet-06b295a178aaabd9f, and subnet-06c9f9aa3ee95d868, all in the 'Available' state and associated with 'vpc-07bf864145b910a5f'. A 'Create subnet' button is located at the top right of the table.

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

The screenshot shows the 'Create subnet' wizard. Step 1 is 'VPC'. It shows a dropdown menu for 'VPC ID' containing 'vpc-0de848dc187fa5c55 (vpc1)'. Below it, 'Associated VPC CIDRs' shows 'IPv4 CIDRs' as '10.0.0.0/24'. Step 2 is 'Subnet settings', which asks for a 'Subnet name' (with placeholder 'subnet in vpc1') and specifies a CIDR block of '10.0.0.0/24'. The overall URL in the browser is 'VPC > Subnets > Create subnet'.

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

The screenshot shows the 'Create subnet' wizard. Step 2 is 'Subnet 1 of 1'. It has sections for 'Subnet name' (set to '10.0.0.0/28'), 'Availability Zone' (set to 'Asia Pacific (Mumbai) / ap-south-1a'), 'IPv4 CIDR block' (set to '10.0.0.0/24'), and 'Tags - optional' (a single tag 'Name: 10.0.0.0/28'). At the bottom are 'Cancel' and 'Create subnet' buttons.

We are successfully created subnet1.

The screenshot shows the AWS VPC Subnets page. A green success message at the top states: "You have successfully created 1 subnet: subnet-089359e7fdc1538e5". The main table lists one subnet named "subnet1" with the following details:

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR
subnet1	subnet-089359e7fdc1538e5	Available	vpc-02bb9a34110310971 ...	10.0.0.0/28	-

Now we need to create Internet Gateway for Custom VPC

The screenshot shows the AWS Internet Gateways page. A table displays one internet gateway named "igw-0ba79bbefb0e6f62f" with the following details:

Name	Internet gateway ID	State	VPC ID	Owner
-	igw-0ba79bbefb0e6f62f	Attached	vpc-07bf864145b910a5f	456609904216

Now name the internet gateway and create it.

The screenshot shows the "Create internet gateway" wizard. The "Internet gateway settings" step is active, showing a "Name tag" field with the value "igw1". The "Tags - optional" step shows a single tag named "Name" with the value "igw1". At the bottom, there are "Cancel" and "Create internet gateway" buttons.

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

The screenshot shows the AWS VPC Internet Gateways page. A green banner at the top indicates that an internet gateway was created: "The following internet gateway was created: igw-0c4422e2a0412be91 - igw1. You can now attach to a VPC to enable the VPC to communicate with the Internet." Below the banner, the page title is "igw-0c4422e2a0412be91 / igw1". The "Details" tab is selected, showing the Internet gateway ID (igw-0c4422e2a0412be91), State (Detached), VPC ID (-), and Owner (456609904216). The "Tags" section shows a single tag named "Name" with the value "igw1".

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

The screenshot shows the "Attach to VPC" dialog box. It asks to "Attach an internet gateway to a VPC to enable the VPC to communicate with the internet. Specify the VPC to attach below." A search bar contains the VPC ID "vpc-0de848dc187fa5c55". Below the search bar is a link "▶ AWS Command Line Interface command". At the bottom are "Cancel" and "Attach internet gateway" buttons.

Now we are successfully attached our IG to the custom VPC

The screenshot shows the AWS VPC Internet Gateways page again. A green banner at the top indicates that the internet gateway was successfully attached: "Internet gateway igw-0c4422e2a0412be91 successfully attached to vpc-0de848dc187fa5c55". Below the banner, the page title is "igw-0c4422e2a0412be91 / igw1". The "Details" tab is selected, showing the Internet gateway ID (igw-0c4422e2a0412be91), State (Attached), VPC ID (vpc-0de848dc187fa5c55 | vpc1), and Owner (456609904216). The "Tags" section shows a single tag named "Name" with the value "igw1".

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

The screenshot shows the AWS VPC dashboard with the 'Route tables' section selected. A context menu is open over route table 'rtb-0d4144756171cc541'. The 'Edit subnet associations' option is highlighted. The main pane displays the 'Explicit subnet associations' table:

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
subnet in vpc1	subnet-08dc1370cb2635857	10.0.0.0/28	-

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

The screenshot shows the 'Edit subnet associations' dialog for route table 'rtb-003a1820897f6c219'. It lists the available subnets and the selected subnet:

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
subnet in vpc1	subnet-021c78c137d9c260c	10.0.0.0/24	-	Main (rtb-003a1820897f6c219 / route ...)

The 'Selected subnets' section shows the selected subnet: 'subnet-021c78c137d9c260c / subnet in vpc1'. The 'Save associations' button is visible at the bottom right.

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

The screenshot shows the AWS VPC dashboard with the 'Route tables' section selected. A context menu is open over route table 'rtb-0d4144756171cc541'. The 'Edit subnet associations' option is highlighted. The main pane displays the 'Explicit subnet associations' table:

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
subnet in vpc1	subnet-08dc1370cb2635857	10.0.0.0/28	-

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

The screenshot shows the 'Edit routes' page for a specific route table. In the 'Destination' column, there are two entries: '10.0.0.0/24' with a target of 'local' and 'Status' 'Active', and a second entry with 'Q 0.0.0.0/0' in the 'Target' field, which is currently empty. Below this input field is a dropdown menu listing various targets, with 'Internet Gateway' selected. At the bottom right of the table are 'Cancel', 'Preview', and 'Save changes' buttons.

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

This screenshot shows the same 'Edit routes' page after the configuration has been updated. The second route entry now has 'Q igw-0c4422e2a0412be91' in the 'Target' field, indicating that the Internet Gateway has been selected. The rest of the interface remains the same with 'Save changes' being the active button.

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

The screenshot shows the 'VPC dashboard' with a green notification bar at the top stating 'Updated routes for rtb-003a1820897f6c219 / route table -vpc1 successfully'. Below this, the 'Route tables' section for 'rtb-003a1820897f6c219' is displayed. A message says 'You can now check network connectivity with Reachability Analyzer'. The 'Details' tab shows route table ID 'rtb-003a1820897f6c219', VPC 'vpc-0de848dc187fa5c55 | vpc1', and other configurations. The 'Routes' tab shows one route entry with a 'Filter routes' search bar and pagination controls.

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

The screenshot shows the AWS VPC dashboard with the following details:

- VPC dashboard**: Shows 'Your VPCs (1/2) Info' with one entry: 'vpc1' (VPC ID: 'vpc-098fbce4e1e377d90', State: Available, IPv4 CIDR: '10.0.0.0/24', IPv6 CIDR: '172.31.0.0/16').
- Subnets**: Shows one subnet: 'ap-south-1a' (Subnet ID: 'subnet in vpc1').
- Route tables**: Shows one route table: 'route table-vpc1'.
- Internet gateways**: Shows one internet gateway: 'igw1'.
- Network connections**: Shows a connection from the subnet to the route table, and from the route table to the internet gateway.

Step:02 Creation of IAM Users

Now go to IAM and Click on Users

The screenshot shows the 'Create user' wizard Step 1: Specify user details:

- User details**:
 - User name: UserA
 - Provide user access to the AWS Management Console - optional
- Are you providing console access to a person?**
 - Specify a user in Identity Center - Recommended
 - I want to create an IAM user

Create Custom Password for UserA.

The screenshot shows the 'Create user' wizard Step 2: Set permissions:

- Console password**:
 - Autogenerated password
 - Custom password
- If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user.**

Select Attach Policies Directly and Click on Create Policy.

The screenshot shows the AWS IAM 'Create user' wizard at Step 2: Set permissions. On the left, there's a sidebar with steps: Step 1 Specify user details, Step 2 Set permissions (which is active), Step 3 Review and create, and Step 4 Retrieve password. The main area is titled 'Set permissions' with the sub-section 'Permissions options'. It lists three methods: 'Add user to group', 'Copy permissions', and 'Attach policies directly'. The 'Attach policies directly' option is selected and highlighted with a blue border. Below this, a section titled 'Permissions policies (1109)' shows a table of policies. The table has columns for 'Policy name', 'Type', and 'Attached entities'. One policy is listed: 'AccessAnalyzerServiceRolePolicy' (AWS managed, Type: AWS managed, Attached entities: 0). There are buttons for 'Create policy' and 'Filter by Type'.

Specify the permissions that should be attached to the policy.

The screenshot shows the AWS IAM 'Create user' wizard at Step 3: Review and create. The sidebar shows Step 1: Specify permissions and Step 2: Review and create. The main area is titled 'Specify permissions' and contains a 'Policy editor' section. It lists several services with their actions: EC2 (Allow 1 Actions), S3 (Allow 1 Actions), EC2 Instance Connect (Allow 1 Actions), and a section for 'Select a service' which is currently empty. There is a button '+ Add more permissions'. At the bottom, there are status indicators: Security: 0, Errors: 0, Warnings: 0, and Suggestions: 0. A 'Cancel' and 'Next' button are at the bottom right.

Specify the policy details.

The screenshot shows the AWS IAM 'Create policy' wizard at Step 2: Review and create. The sidebar shows Step 1: Specify permissions and Step 2: Review and create. The main area is titled 'Review and create' and 'Policy details'. It includes fields for 'Policy name' (containing 'custom_policy') and 'Description - optional' (containing 'Created for the ec2 and s3 full access along with Ec2 Instance connect'). At the bottom, there is a 'Permissions defined in this policy' section with an 'Edit' button and a 'Search' bar.

Now review the permissions and create the policy.

The screenshot shows the 'Create policy' wizard. In the 'Allow' section, three services are granted full access: S3, EC2, and EC2 Instance Connect. Below this, there's an optional 'Add tags' step where users can add key-value pairs to identify resources. At the bottom right, there are 'Cancel', 'Previous', and 'Create policy' buttons.

The policy has been created successfully.

The screenshot shows the 'Policies' list in the IAM console. A green banner at the top indicates 'Policy custom_policy created.' The list includes several other policies like 'AWSGlueServiceRole-gowtham-EZCRC-s3Policy' and 'AWSLambdaBasicExecutionRole-ef276860-d670-4033-b...'. The 'custom_policy' is highlighted. The interface includes a search bar, a filter, and a 'Create policy' button.

Now attach the newly created policy to UserA.

The screenshot shows the 'Set permissions' wizard. It starts with a 'Step 3 Review and create' step, followed by a 'Step 4 Retrieve password' step. The main area is titled 'Permissions options' and shows three options: 'Add user to group', 'Copy permissions', and 'Attach policies directly'. The 'Attach policies directly' option is selected. Below this, the 'Permissions policies' step shows a list of policies, with 'custom_policy' selected. The interface includes a search bar, a filter, and a 'Create policy' button. At the bottom, there's a 'Set permissions boundary - optional' section and navigation buttons for 'Cancel', 'Previous', and 'Next'.

Now review the permissions and create the user.

User details

Name	Type	Used as
custom_policy	Customer managed	Permissions policy

Permissions summary

Tags - optional

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Cancel Previous Create user

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

User created successfully

You can view and download the user's password and email instructions for signing in to the AWS Management Console.

IAM > Users > Create user

Step 1 Specify user details

Step 2 Set permissions

Step 3 Review and create

Step 4 Retrieve password

Console sign-in details

Console sign-in URL
https://456609904216.sigin.aws.amazon.com/console

User name
UserA

Console password
***** Show

Email sign-in instructions

Download .csv file Return to users list

Now create another user by specifying the name as UserB.

IAM > Users > Create user

Step 1 Specify user details

Step 2 Set permissions

Step 3 Review and create

Step 4 Retrieve password

User details

User name
UserB

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = _ @ - (hyphen)

Provide user access to the AWS Management Console - optional

If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.

Are you providing console access to a person?

User type

Specify a user in Identity Center - Recommended

We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage user access to their AWS accounts and cloud applications.

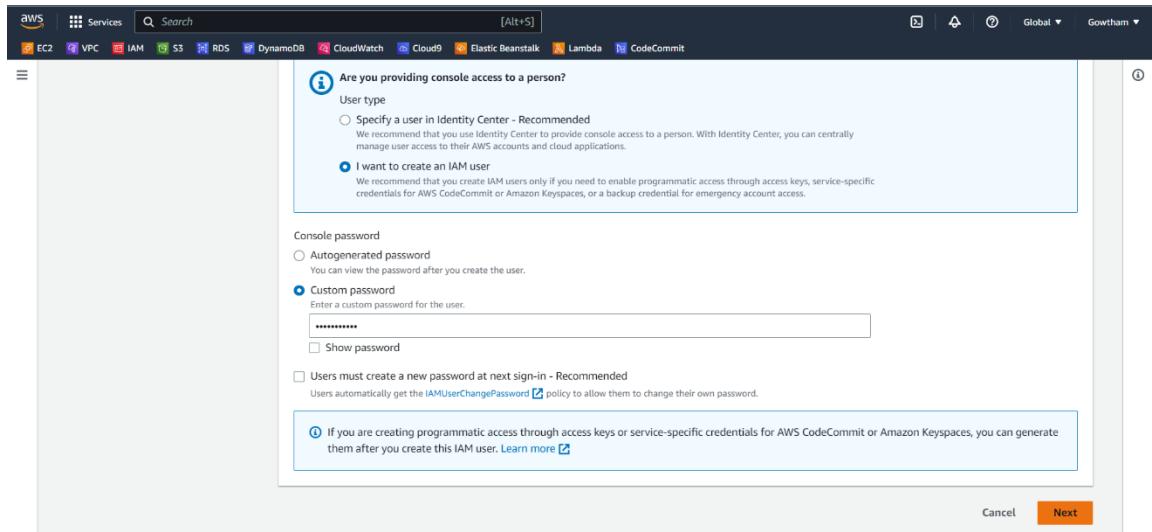
I want to create an IAM user

We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keenases, or a backup credential for emergency account access.

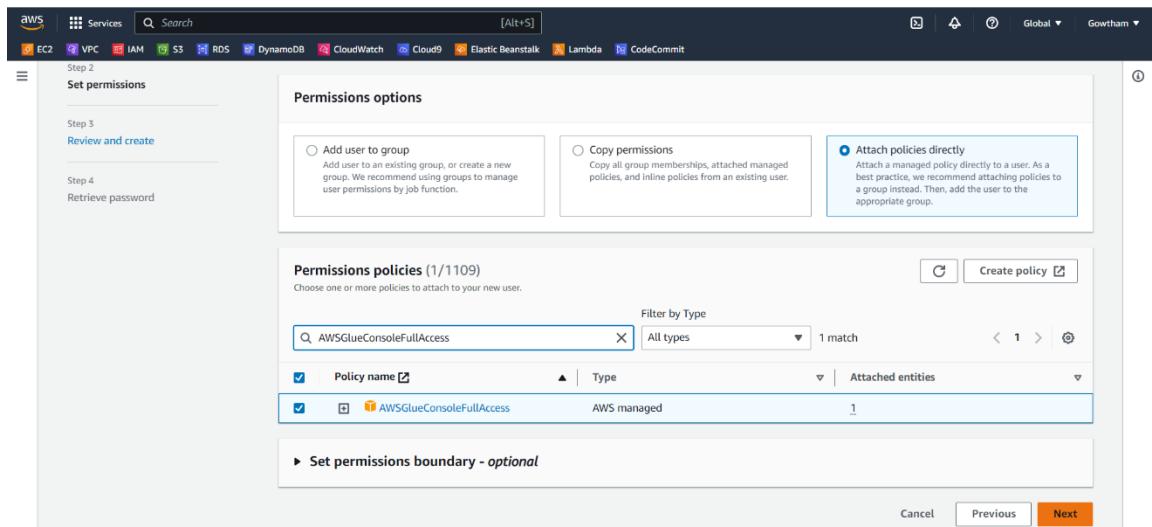
Console password
 Autogenerated password

You can view the password after you create the user.

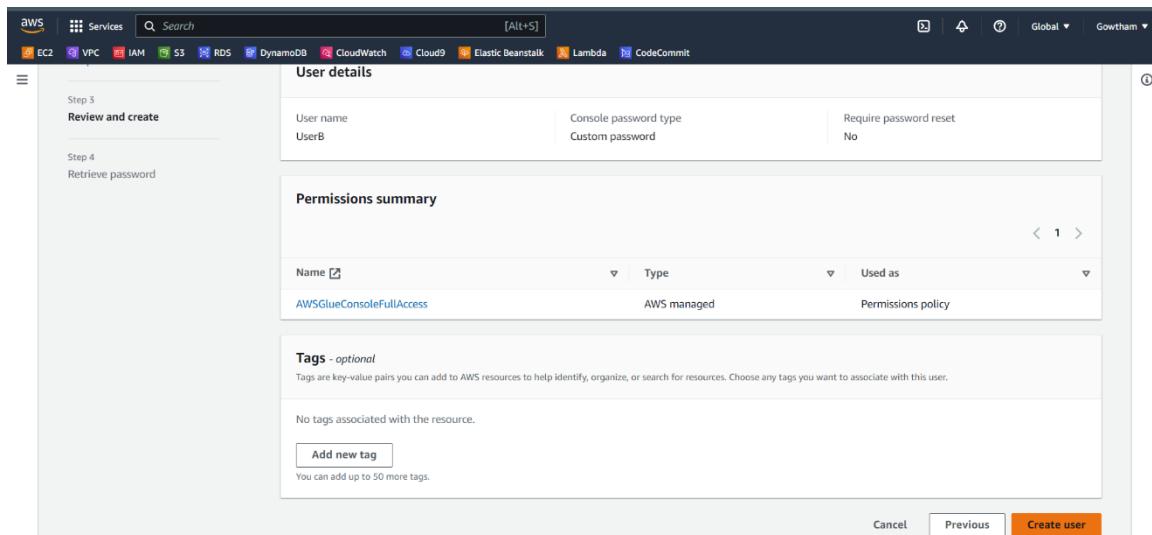
Enter the custom password for the user.



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.

The screenshot shows the AWS Management Console with the IAM service selected. A green banner at the top indicates "User created successfully". The main area displays the "Create user" wizard, specifically Step 4: Retrieve password. It shows the "Console sign-in details" section with the following information:

- Console sign-in URL: <https://456609904216.signin.aws.amazon.com/console>
- User name: UserB
- Console password: ***** (Masked)

Buttons at the bottom include "Download .csv file" and "Return to users list".

Now we have successfully created two users

The screenshot shows the AWS Management Console with the IAM service selected. The left sidebar shows the "Identity and Access Management (IAM)" menu with "Users" selected. The main area displays the "Users (2) Info" table:

User name	Groups	Last activity	MFA	Password age	Active key age
UserA	None	Never	None	Now	-
UserB	None	Never	None	5 minutes ago	-

Step:03 Creation of IAM Role For S3Full Access

Goto IAM > Roles and click on create role

The screenshot shows the AWS Management Console with the IAM service selected. The left sidebar shows the "Identity and Access Management (IAM)" menu with "Roles" selected. The main area displays the "Roles (24) Info" table:

Role name	Trusted entities	Last activity
aws-elasticbeanstalk-service-role	AWS Service: elasticbeanstalk	6 days ago
AWSCloud9SSMAccessRole	AWS Service: cloud9, and 1 more.	4 days ago
AWSGlueServiceRole-gowtham	AWS Service: glue	13 hours ago
AWSGlueServiceRole-puneet	AWS Service: glue	14 hours ago
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable	AWS Service: dynamodb.application-autoscaling (Service-Linked Role)	3 days ago
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Linked Role)	4 days ago
AWSServiceRoleForAWSCloud9	AWS Service: cloud9 (Service-Linked Role)	4 days ago
AWSServiceRoleForEc2InstanceConnect	AWS Service: ec2-instance-connect (Service-Linked Role)	5 days ago
AWSServiceRoleForElasticLoadBalancing	AWS Service: elasticloadbalancing (Service-Linked Role)	4 days ago
AWSServiceRoleForGlobalAccelerator	AWS Service: globalaccelerator (Service-Linked Role)	-

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

Trusted entity type

- AWS service

Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account

Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity

Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation

Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy

Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

EC2
Allows EC2 instances to call AWS services on your behalf.

Lambda
Allows Lambda functions to call AWS services on your behalf.

Use cases for other AWS services:

Choose a service to view use case ▾

Cancel Next

Give a name to the IAM Role as “calls3onbehalfofec2”

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.
calls3onbehalfofec2

Maximum 64 characters. Use alphanumeric and '+-, @_-' characters.

Description
Add a short explanation for this role.
Allows EC2 Instances to call AWS services on your behalf.

Maximum 1000 characters. Use alphanumeric and '+-, @_-' characters.

Step 1: Select trusted entities

```

1+ []
2+   "Version": "2012-10-17",
3+   "Statement": [
4+     {
5+       "Effect": "Allow",
6+       "Action": [
7+         "sts:AssumeRole"
8+       ],
9+       "Principal": [
10+         {
11+           "Service": [
12+             "ec2.amazonaws.com"
13+           ]
14+         }
15+       ]
16+     }
  
```

Edit

Verify the trusted entity EC2(AWS Service)

Description
Add a short explanation for this role.
Allows EC2 Instances to call AWS services on your behalf.

Maximum 1000 characters. Use alphanumeric and '+-, @_-' characters.

Step 1: Select trusted entities

```

1+ []
2+   "Version": "2012-10-17",
3+   "Statement": [
4+     {
5+       "Effect": "Allow",
6+       "Action": [
7+         "sts:AssumeRole"
8+       ],
9+       "Principal": [
10+         {
11+           "Service": [
12+             "ec2.amazonaws.com"
13+           ]
14+         }
15+       ]
16+     }
  
```

Edit

Verify the Permission added and create the role

Step 2: Add permissions

Policy name	Type	Attached as
AmazonS3FullAccess	AWS managed	Permissions policy

Tags

Add tags - optional Info
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

Add tag

You can add up to 50 more tags.

IAM Role "calls3onbehalfofec2" has been created

Identity and Access Management (IAM)

View role

Roles (25) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last act...
aws-elasticsearch-service-role	AWS Service: elasticsearch	6 days ago
AWSCloud9SSMAccessRole	AWS Service: ec2, and 1 more. <small>Info</small>	4 days ago
AWSGlueServiceRole-gowtham	AWS Service: glue	13 hours ago
AWSGlueServiceRole-puneet	AWS Service: glue	14 hours ago
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable	AWS Service: dynamodb.application-autoscaling (Service-Linked Role)	3 days ago
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Linked Role)	4 days ago
AWSServiceRoleForAWSCloud9	AWS Service: cloud9 (Service-Linked Role)	4 days ago
AWSServiceRoleForEc2InstanceConnect	AWS Service: ec2-instance-connect (Service-Linked Role)	5 days ago

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

Sign in as IAM user

Account ID (12 digits) or account alias
456609904216

IAM user name
UserA

Password
.....

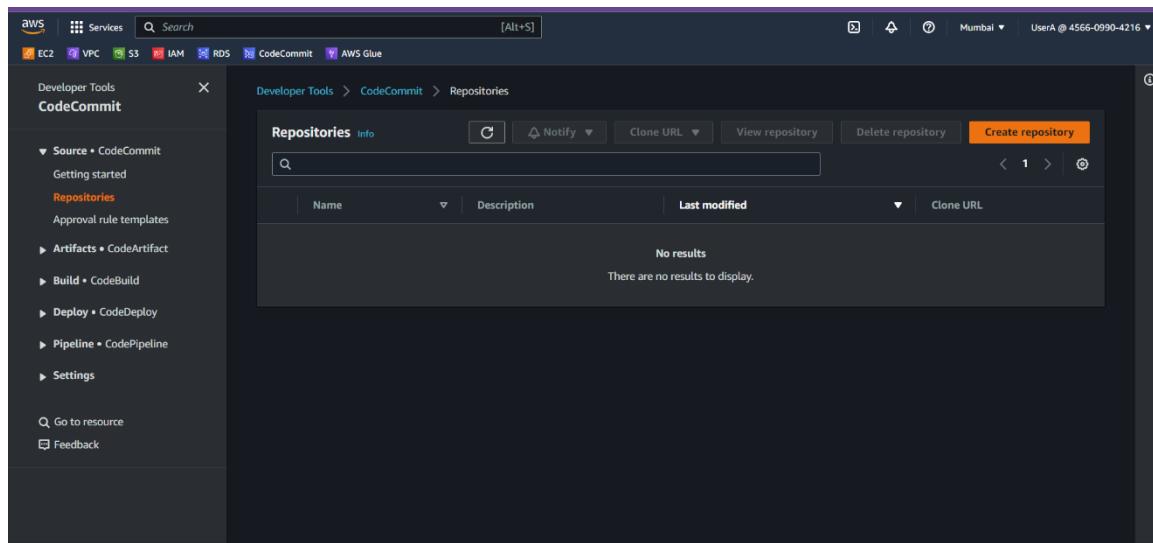
Remember this account

Sign in

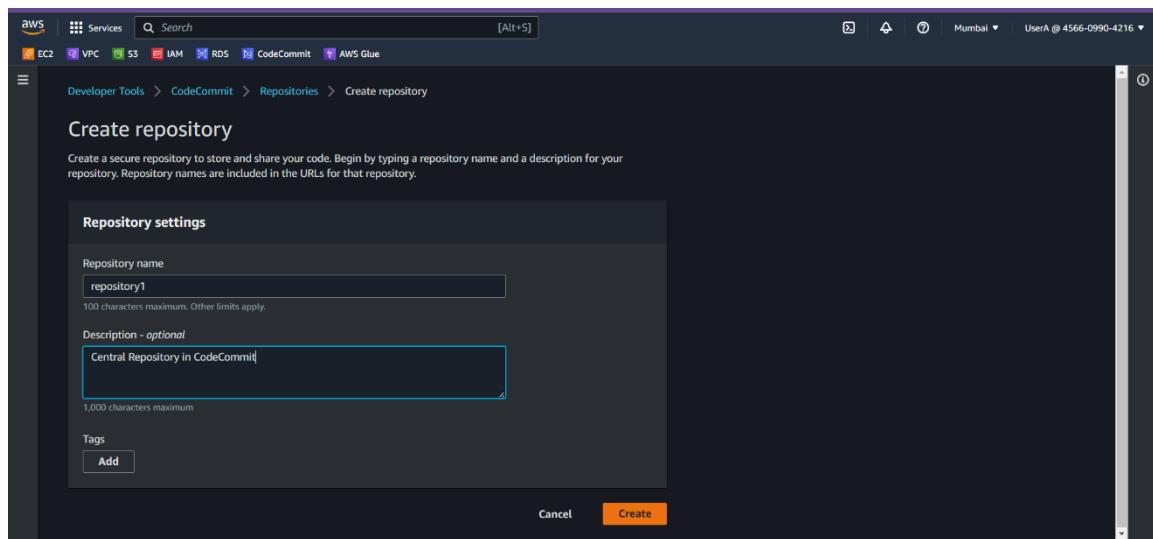
Sign in using root user email
Forgot password?

Amazon Lightsail
Lightsail is the easiest way to get started on AWS
Learn more »

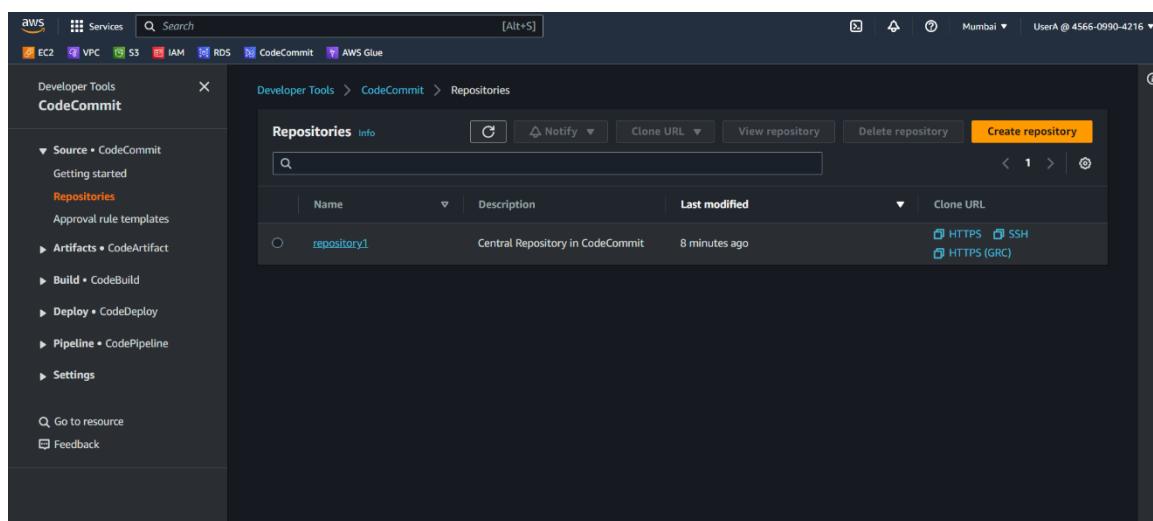
Go to Developer Tools > CodeCommit > Repositories and click on Create Repository



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

The screenshot shows the AWS CodeCommit service in the AWS Management Console. On the left, the navigation pane is open with the 'Code' section selected under 'Source - CodeCommit'. The main area displays steps for repository creation: 'Step 2: Git credentials' (instructions to generate credentials), 'Step 3: Clone the repository' (with a command line example and a 'Copy' button), and 'Additional details' (link to documentation). Below this, the 'repository1' info card is shown, indicating it is an 'Empty repository'. There are three buttons on the right: 'Add file', 'Create file', and 'Upload file'.

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

This screenshot shows the 'File' upload interface within the AWS CodeCommit service. A file selection dialog is open, showing files like 'grades' and 'UserA_codecommit_credentials'. The main panel has a 'Choose file' button and a 'Commit changes to main' form with fields for 'Author name' (set to 'Gowtham') and 'Email address' (set to 'gowthamsalumuri@gmail.com').

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

The screenshot shows the AWS CodeCommit service after a file has been uploaded. The 'repository1' info card now lists 'grades.csv' with a size of '1 KB'. Below it, the 'Commit changes to main' interface is shown, with the commit message field containing 'committed grades.csv file'. The 'Commit changes' button is highlighted in orange at the bottom right.

The csv file has been uploaded into the CodeCommit Repository

```

1 last_name,First_name,SSN,Test1,Test2,Test3,Test4,Final_Grade
2 AlfaFa,Aloysius,123-45-6789,49,98,100,83,49,D-
3 Alfred,University,123-12-1234,61,97,96,97,48,D-
4 George,Washington,123-45-6789,49,98,100,83,49,D-
5 Andrevia,Electric,987-65-4321,42,23,96,45,A7,B-
6 Dunkin,Fred,456-78-9012,43,78,88,77,45,A
7 Rubbie,Betty,234-56-7890,44,90,80,90,46,C-
8 NoShaw,Cecil,345-67-8901,45,11,1,4,43,F-
9 Hopper,Marty,123-45-6789,49,98,100,83,49,D-
10 Airpump,Andrew,223-45-6789,49,0,1,0,00,100,B3,A,
11 Rockus,Tim,143-12-1234,48,1,97,96,97,A+
12 Carnivore,Art,505-89-0123,44,1,80,60,40,D+
13 Elroy,Jean,123-45-6789,49,98,100,83,49,D-
14 Elroykins,Lisa,456-71-9012,45,1,78,88,77,B-
15 Franklin,Benny,234-56-2890,50,1,98,86,90,B-
16 George,Boy,345-67-3901,40,1,11,1,4,B
17 Heffalump,Harvey,652-79-9439,30,1,20,30,40,C
    
```

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

Go To Amazon S3 and click on Create Bucket

Amazon S3

Buckets
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3

Block Public Access settings for this account

Storage

Amazon S3
Store and retrieve any amount of data from anywhere

Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance.

Create a bucket

Every object in S3 is stored in a bucket. To upload files and folders to S3, you'll need to create a bucket where the objects will be stored.

Create bucket

Pricing

With S3, there are no minimum fees. You only pay for what you use. Prices are based on the location of your S3 bucket.

Estimate your monthly bill using the [AWS Simple Monthly Calculator](#)

[View pricing details](#)

How it works

Introduction to Amazon S3

aws

Copy link

aws

Give the name to the bucket and specify the AWS region

Create bucket

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

General configuration

Bucket name: s3bucketforawsgluedemo

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

AWS Region: Asia Pacific (Mumbai) ap-south-1

Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.
[Choose bucket](#)

Object Ownership

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

Set Object Ownership as the ACLs enabled

The screenshot shows the 'Object Ownership' configuration page for an S3 bucket. The 'ACLs enabled' option is selected, indicated by a blue border around the radio button. A note below states: 'Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.' A warning message in a red-bordered box says: '⚠️ We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.' Below this, there are sections for 'Bucket owner preferred' and 'Object writer', with a note about enforcing ownership.

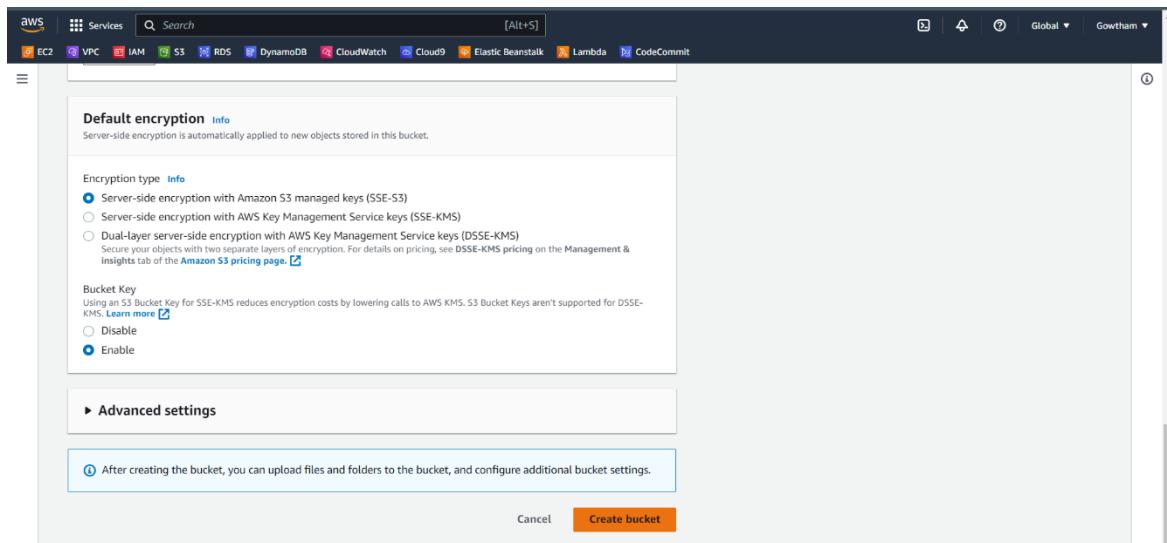
You can give access to your bucket to others by unchecking the block public access

The screenshot shows the 'Block Public Access settings for this bucket' page. The 'Block all public access' checkbox is unchecked. A note below says: 'Turning off block all public access might result in this bucket and the objects within becoming public. AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.' There are also several other checkboxes for blocking specific types of access.

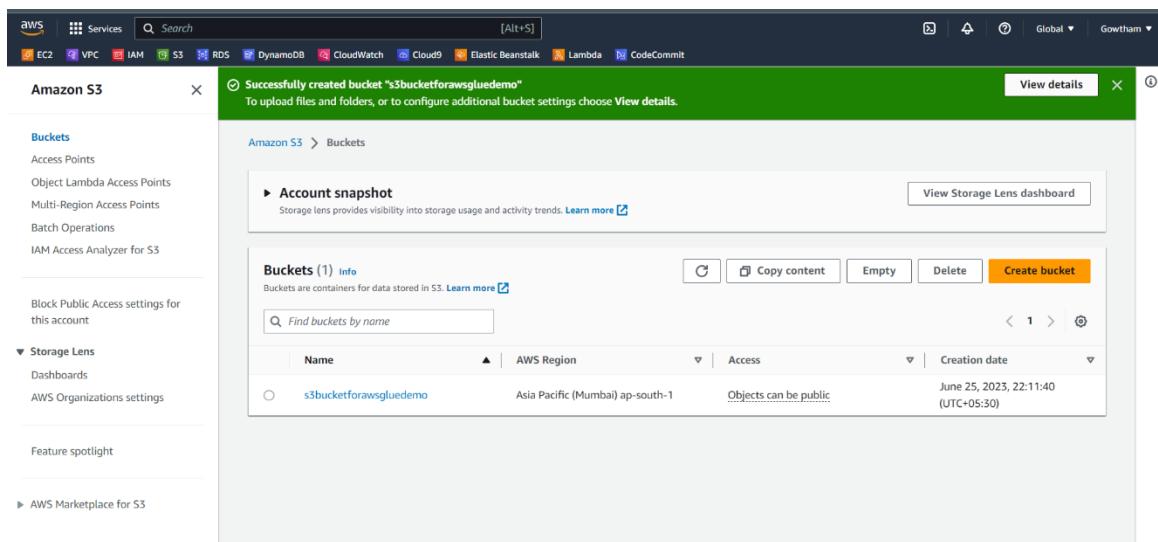
Click on the acknowledgement check box and disable the bucket versioning

The screenshot shows the 'Bucket Versioning' and acknowledgement page. The acknowledgement checkbox is checked, and the note below it says: '⚠️ Turning off block all public access might result in this bucket and the objects within becoming public. AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.' The 'Bucket Versioning' section has 'Disable' selected. Below this is a 'Tags (0) - optional' section with an 'Add tag' button.

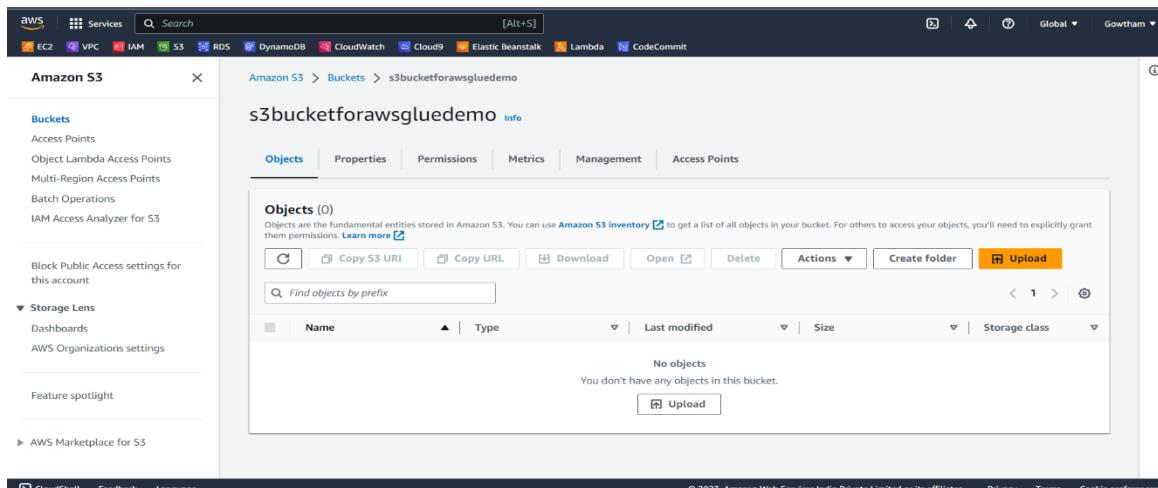
Use the Default encryption type as SSE-S3 and click on create bucket



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

The screenshot shows the AWS EC2 Dashboard. On the left, there's a sidebar with links like 'EC2 Dashboard', 'Instances', 'Images', etc. The main area has a 'Resources' section with various metrics (Instances running: 0, Auto Scaling Groups: 0, Dedicated Hosts: 0, etc.). Below it is a 'Launch instance' button. To the right, there's an 'Account attributes' section with details like 'Default VPC' (vpc-07bf864145b910a5f) and 'Supported platforms' (VPC). There's also an 'Explore AWS' section with a 'Launch an instance' button.

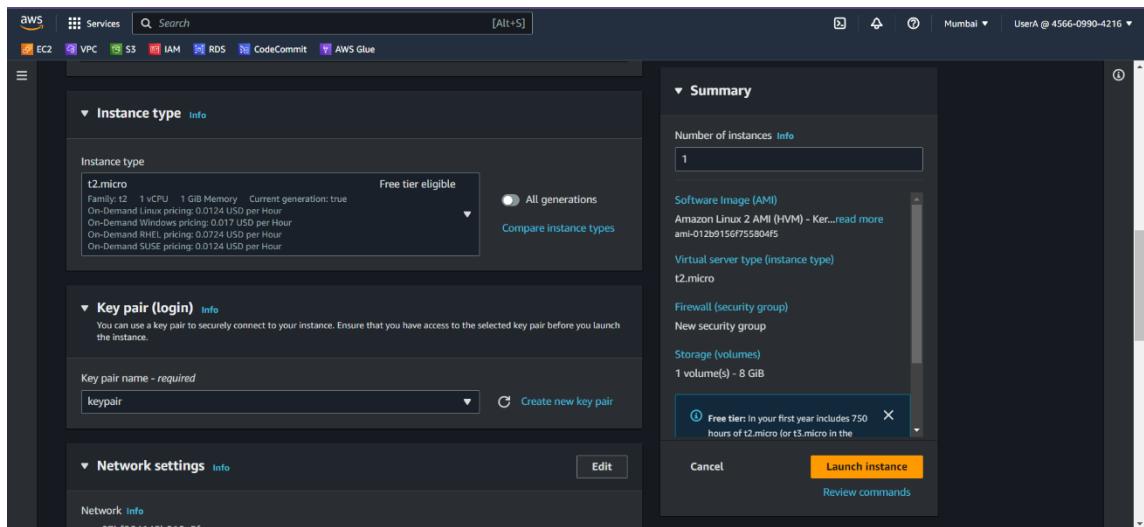
Goto EC2 > Instances > Launch Instance then specify name of the instance

The screenshot shows the 'Launch an instance' wizard. It starts with a 'Name and tags' step where 'instance' is entered. Then it moves to an 'Application and OS Images (Amazon Machine Image)' step, where 'Amazon Linux 2 AMI (HVM - Ker...' is selected. The summary on the right shows 1 instance, the selected AMI, instance type t2.micro, and 1 volume (8 GiB). A tooltip indicates a free tier of 750 hours. At the bottom are 'Cancel', 'Launch instance', and 'Review commands' buttons.

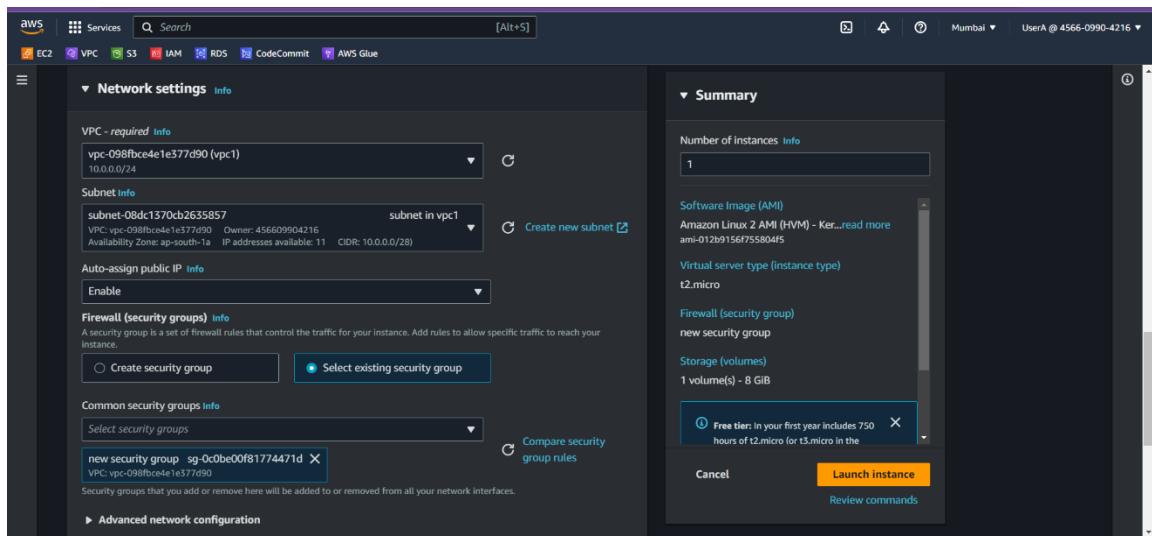
Select Amazon Linux 2 AMI

The screenshot shows the 'Application and OS Images (Amazon Machine Image)' catalog. It lists the 'Amazon Machine Image (AMI)' amzn2-ami-kernel-5.10-hvm-2.0.20230612.0-x86_64-gp2. The summary on the right is identical to the previous screenshot, showing 1 instance, the selected AMI, instance type t2.micro, and 1 volume (8 GiB). A tooltip indicates a free tier of 750 hours. At the bottom are 'Cancel', 'Launch instance', and 'Review commands' buttons.

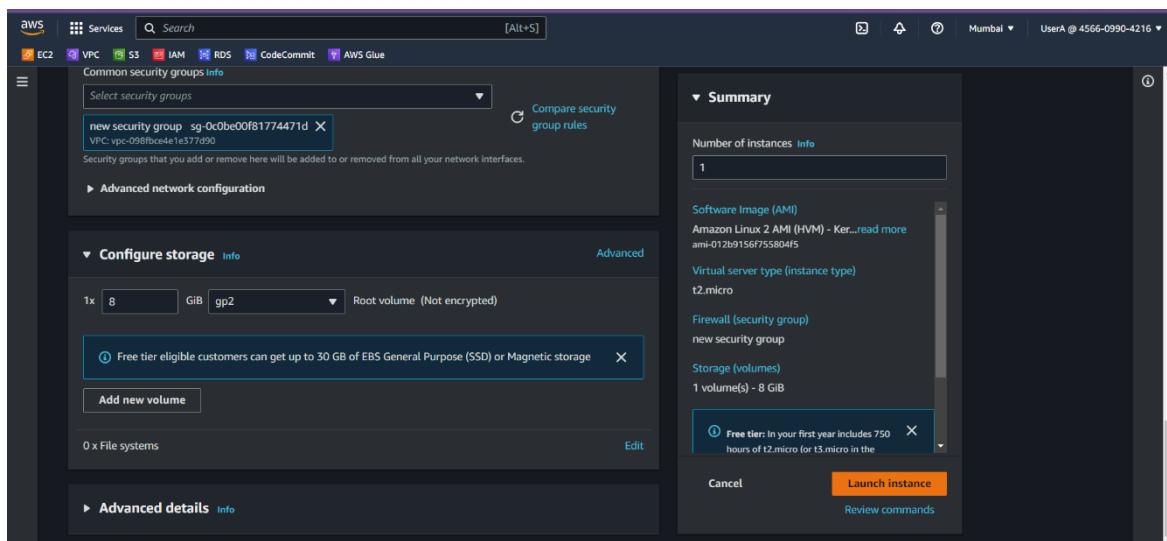
Select the instance as t2.micro and a key pair you have already created



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

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like EC2 Dashboard, EC2 Global View, Events, Limits, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, AMI Catalog), and Elastic Block Store. The main content area displays a table of instances. One row is selected, showing the instance ID i-013606b06007db782, which is labeled 'instance'. The status is 'Running' with a green dot, and the instance type is 't2.micro'. A progress bar indicates '2/2 checks passed'. Below the table, a modal window titled 'Select an instance' is open, showing the same instance details.

Go to Actions > Security > Modify IAM Role

This screenshot is similar to the previous one, but the 'Actions' dropdown menu is open over the instance table. The 'Modify IAM role' option is highlighted with a blue border. The main content area shows the instance details for i-013606b06007db782, including its public IPv4 address (13.233.93.224) and private IPv4 address (10.0.7).

Select the already created role and update IAM Role

This screenshot shows the 'Modify IAM role' dialog box. At the top, it says 'Attach an IAM role to your instance.' Below that, the 'Instance ID' dropdown is set to 'i-013606b06007db782 (instance)'. A note below the dropdown states: 'Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.' There are two buttons at the bottom: 'Cancel' and 'Update IAM role'.

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

The screenshot shows the AWS Management Console with the EC2 service selected. The main pane displays a table of instances, with one row highlighted for an instance named 'instance' with ID 'i-013606b06007db782'. A green banner at the top of the table indicates that the IAM role 'calls3onbehalfofec2' has been successfully attached to this instance. The instance details pane below shows the instance summary, including its public IPv4 address (15.233.95.224), private IPv4 address (10.0.0.7), and other configuration details.

Now connect the instance using EC2 Instance Connect

The screenshot shows the 'EC2 Instance Connect' dialog box. It lists two connection options: 'Connect using EC2 Instance Connect' (selected) and 'Connect using EC2 Instance Connect Endpoint'. The 'Connect using EC2 Instance Connect' option requires a 'Public IP address' (13.233.95.224) and a 'User name' (ec2-user). A note at the bottom states: 'Note: In most cases, the default user name, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.' At the bottom right are 'Cancel' and 'Connect' buttons.

Instance has been connected successfully

The screenshot shows a terminal window with a black background and white text. It displays a series of commands being run on an Amazon Linux 2 AMI instance. The commands include navigating to the root directory, updating package lists, and installing Git. The output shows the progress of the download and installation, ending with a success message: 'Resolving Dependencies' and '--> Package perl-git.x86_64 0:2.40.1-1.amzn2.0.1 will be installed'. The terminal window also shows the public and private IP addresses of the instance.

Make a directory and initialize the git

```
Verifying : perl-Git-2.40.1-1.amzn2.0.1.noarch
Verifying : libperl-Error-0.17020-2.amzn2.noarch
Verifying : git-2.40.1-1.amzn2.0.1.x86_64

Installed:
git.x86_64 0:2.40.1-1.amzn2.0.1

Dependency Installed:
git-core.x86_64 0:2.40.1-1.amzn2.0.1      git-core-doc.noarch 0:2.40.1-1.amzn2.0.1      perl=Error.noarch 1:0.17020-2.amzn2      perl-Git.noarch 0:2.40.1-1.amzn2.0.1

Complete!
[root@ip-10-0-0-7 ec2-user]# mkdir dir1
[root@ip-10-0-0-7 ec2-user]# cd dir1
[root@ip-10-0-0-7 dir1]# git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
initialized empty Git repository in /home/ec2-user/dir1/.git/
[root@ip-10-0-0-7 dir1]# 
```

i-013606b06007db782 (instance)
PublicIPs: 13.233.93.224 PrivateIPs: 10.0.0.7

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

Developer Tools > CodeCommit > Repositories

Name	Description	Last modified	Clone URL
repository1	Central Repository in CodeCommit	17 hours ago	HTTPS SSH HTTPS (GRC)

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

Identity and Access Management (IAM)

UserA

Summary

ARN	Console access	Access key 1
arn:aws:iam::456609904216:user/UserA	Enabled without MFA	Not enabled
Created	Last console sign-in	Access key 2
June 26, 2023, 14:12 (UTC+05:30)	Never	Not enabled

Permissions policies (2)

Permissions are defined by policies attached to the user directly or through groups.

Filter by Type

Generate the credentials to access the AWS CodeCommit

The screenshot shows the AWS IAM console with the 'Identity and Access Management (IAM)' service selected. A modal window titled '1 policy added' is open, indicating a successful operation. Below it, the 'HTTPS Git credentials for AWS CodeCommit (0)' section is visible, showing a table with one row for 'User name' (UserA-at-416) and a 'Generate credentials' button.

Now download the generated credentials

The screenshot shows the 'Generate credentials' modal for AWS CodeCommit. It displays a message: 'Your new credentials are available.' and instructions to save the user name and password or download the credentials file. It also shows the user name (UserA-at-416) and password (redacted). A 'Download credentials' button is present at the bottom.

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

```

< - > C https://ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-013606b06007db782&osUser=ec2-user&region=ap-south-1&sshPort=22#/
AWS Services Search [Alt+S] Global Gowtham
EC2 VPC S3 IAM RDS CodeCommit AWS Glue
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint: git config --global init.defaultBranch <name>
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint: git branch -m <name>
Initialized empty Git repository in /home/ec2-user/dirl/.git/
[root@ip-10-0-0-7 dirl]# git remote add origin https://git-codecommit.ap-south-1.amazonaws.com/v1/repos/repository1
[root@ip-10-0-0-7 dirl]# git pull add origin main
fatal: 'add' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and that the repository exists.
[root@ip-10-0-0-7 dirl]# git pull origin main
Username for 'https://git-codecommit.ap-south-1.amazonaws.com': UserA-at-456609904216
Password for 'https://UserA-at-456609904216@git-codecommit.ap-south-1.amazonaws.com':
remote: counting objects: 3, done.
Unpacking objects: 100% (3/3), 648 bytes | 64.00 KB/s, done.
From https://git-codecommit.ap-south-1.amazonaws.com/v1/repos/repository1
 * branch      main    -> FETCH_HEAD
 * [new branch] main    -> origin/main
[root@ip-10-0-0-7 dirl]#

```

i-013606b06007db782 (instance)
PublicIPs: 13.233.93.224 PrivateIPs: 10.0.0.7

The bucket "s3bucketforawsgluedemo" has no objects

The screenshot shows the AWS S3 console interface. On the left, a sidebar titled 'Amazon S3' contains navigation links like 'Buckets', 'Access Points', and 'Storage Lens'. The main area displays the 's3bucketforawsgluedemo' bucket's details. The 'Objects' tab is selected, showing a table with one row: 'No objects'. A message below the table states, 'You don't have any objects in this bucket.' At the bottom right of the table, there is a 'Upload' button.

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

```
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint: git branch -m <name>
Initialized empty Git repository in /home/ec2-user/dirl/.git/
[root@ip-10-0-0-7 dirl]# git remote add origin https://git-codecommit.ap-south-1.amazonaws.com/v1/repos/repository1
[root@ip-10-0-0-7 dirl]# git pull add origin main
fatal: 'add' does not appear to be a git repository
fatal: Could not read from remote repository.

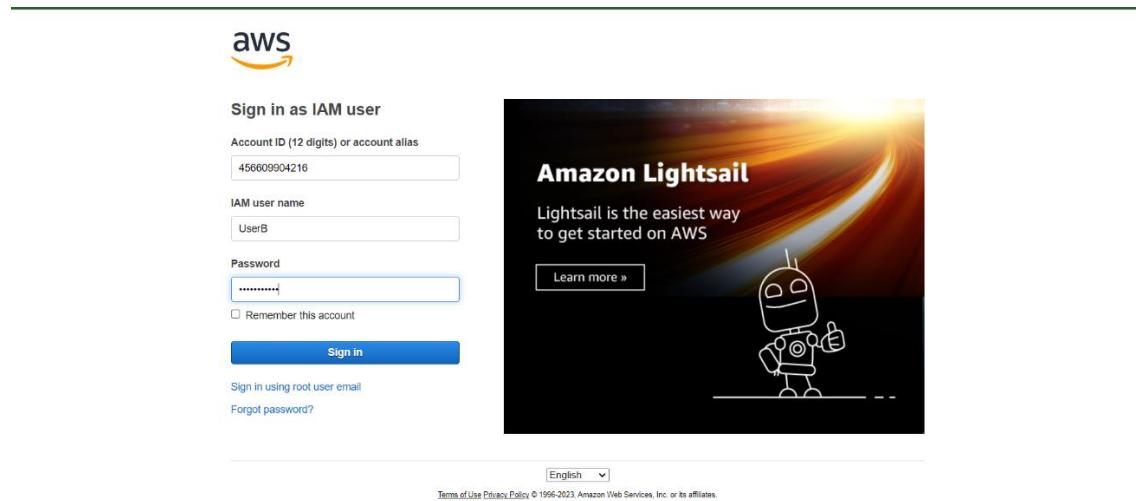
Please make sure you have the correct access rights
and the repository exists.
[root@ip-10-0-0-7 dirl]# git pull origin main
Username for 'https://git-codecommit.ap-south-1.amazonaws.com': UserA-at-456609904216
Password for 'https://UserA-at-456609904216@git-codecommit.ap-south-1.amazonaws.com':
remote: Counting objects: 3, done.
Unpacking objects: 100% (3/3), 648 bytes | 648.00 KIB/s, done.
From https://git-codecommit.ap-south-1.amazonaws.com/v1/repos/repository1
 * branch            main      -> FETCH_HEAD
 * [new branch]      main      -> origin/main
[root@ip-10-0-0-7 dirl]# ls
grades.csv
[root@ip-10-0-0-7 dirl]# aws s3 ls
2023-06-25 16:41:40 s3bucketforawsgluedemo
[root@ip-10-0-0-7 dirl]# aws s3 cp grades.csv s3://s3bucketforawsgluedemo
upload: ./grades.csv to s3://s3bucketforawsgluedemo/grades.csv
[root@ip-10-0-0-7 dirl]#
```

i-013606b06007db782 (instance)
Public IPs: 13.233.93.224 Private IPs: 10.0.0.7

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

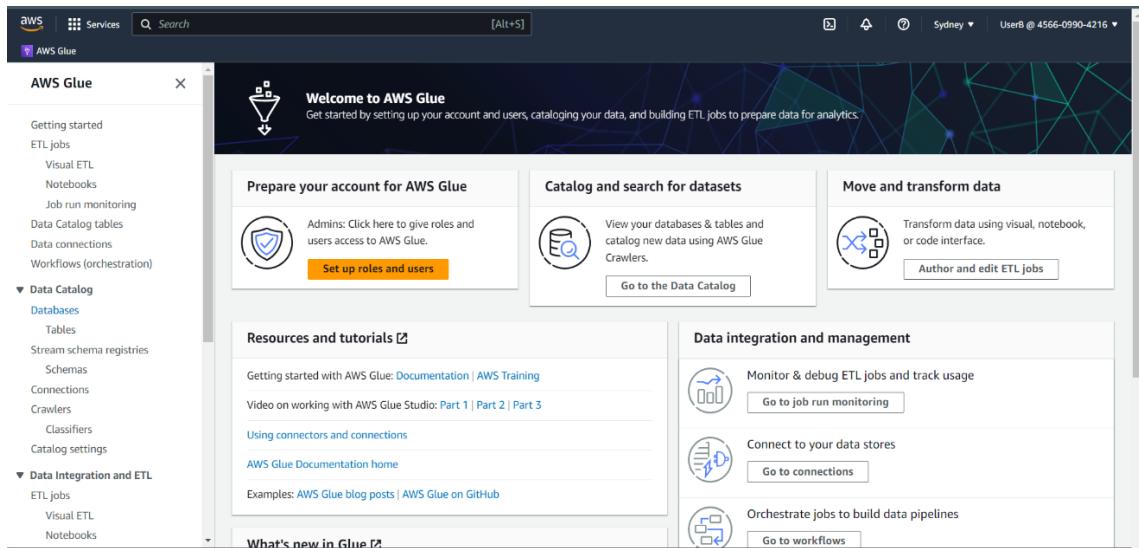
The screenshot shows the AWS S3 console interface. The 'Objects' table in the main area now lists a single item: 'grades.csv'. The table includes columns for 'Name', 'Type', 'Last modified', 'Size', and 'Storage class'. The 'Last modified' column shows 'June 26, 2023, 15:38:43 (UTC+05:30)' and the 'Size' column shows '772.0 B'. The 'Storage class' column shows 'Standard'.

Log in to the IAM User-User B



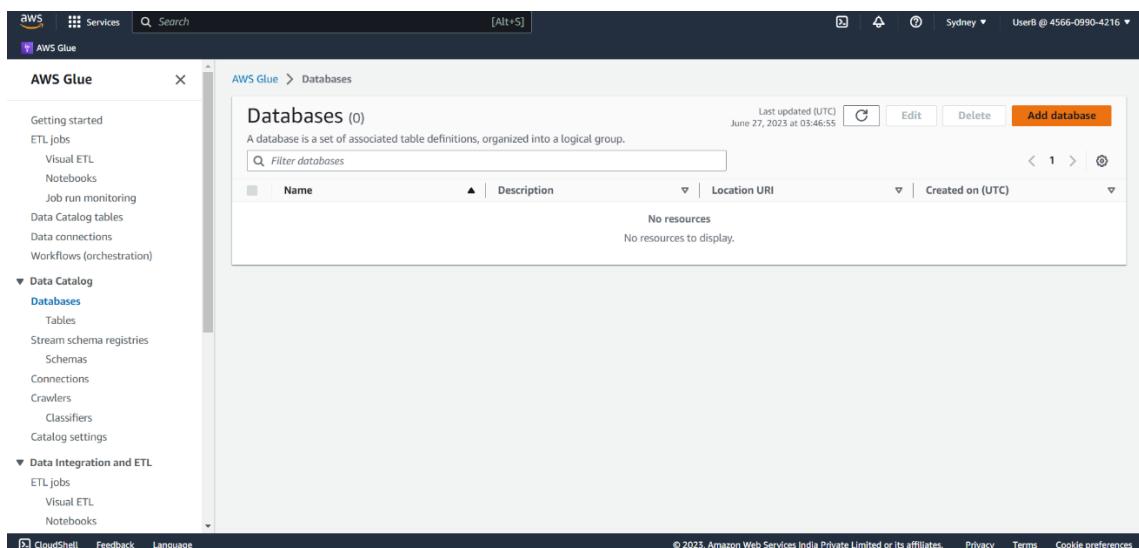
The image shows two side-by-side screenshots. On the left is the AWS Sign in as IAM user page. It has fields for Account ID (12 digits) or account alias (containing '456609904216'), IAM user name ('UserB'), Password ('*****'), and a Remember this account checkbox. Below these are 'Sign in' and 'Forgot password?' links. On the right is the Amazon Lightsail landing page with a banner stating 'Lightsail is the easiest way to get started on AWS' and a 'Learn more »' button, featuring a cartoon robot icon.

Goto the AWS Glue and click on Databases



The screenshot shows the AWS Glue service dashboard. The left sidebar includes options like Getting started, ETL jobs, Visual ETL, Notebooks, Job run monitoring, Data Catalog tables, Data connections, Workflows (orchestration), Data Catalog (Tables, Stream schema registries, Schemas, Connections, Crawlers, Classifiers, Catalog settings), Data Integration and ETL (ETL jobs, Visual ETL, Notebooks). The main content area features a 'Welcome to AWS Glue' section with a 'Set up roles and users' button, a 'Catalog and search for datasets' section with a 'Go to the Data Catalog' button, and a 'Move and transform data' section with a 'Author and edit ETL jobs' button. Below these are sections for Resources and tutorials, Data integration and management, and a 'What's new in Glue' summary.

Now click on add database



The screenshot shows the 'Databases' page under the AWS Glue service. The left sidebar is identical to the previous screenshot. The main area displays a table titled 'Databases (0)' with columns for Name, Description, Location URI, and Created on (UTC). A message indicates 'No resources' and 'No resources to display.' At the top right, there are buttons for Edit, Delete, and Add database (highlighted in orange).

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

AWS Glue > Databases > Add database

Create a database

Database details

Name: databaseinawsglue

Location - optional

Description - optional

Create database

The database has been created successfully

AWS Glue > Databases

Databases (1)

Name	Description	Location URI	Created on (UTC)
databaseinawsglue	-	-	June 27, 2023 at 04:01:58

Add database

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

AWS Glue > Databases > databaseinawsglue

databaseinawsglue

Database properties

Name	Description	Location	Created on (UTC)
databaseinawsglue	-	-	June 27, 2023 at 04:01:58

Tables (0)

Name	Database	Location	Classification	Deprecated	View data
No available tables	-	-	-	-	-

Add tables using crawler

Enter the crawler name and click on next

AWS Glue > Crawlers > Add crawler

Set crawler properties

Crawler details Info

Name: Name can be up to 255 characters long. Some character set including control characters are prohibited.

Description - optional:
Crawler infers the Schema of data fetched from s3 and converts into catalog tables

Descriptions can be up to 2048 characters long.

Tags - optional:

Use tags to organize and identify your resources.

Cancel **Next**

Click on Add a data source

AWS Glue > Crawlers > Add crawler

Choose data sources and classifiers

Data source configuration

Is your data already mapped to Glue tables?

Not yet Select one or more data sources to be crawled.

Yes Select existing tables from your Glue Data Catalog.

Data sources (0) Info

The list of data sources to be scanned by the crawler.

Type	Data source	Parameters
S3		

Add a data source

Custom classifiers - optional

A classifier checks whether a given file is in a format the crawler can handle. If it is, the classifier creates a schema in the form of a StructType object that matches that data format.

Cancel Previous **Next**

Select the data source as S3 and specify the S3 path

Add data source

Data source

Choose the source of data to be crawled.

Network connection - optional

Optionally include a Network connection to use with this S3 target. Note that each crawler is limited to one Network connection so any other S3 targets will also use the same connection (or none, if left blank).

Location of S3 data

In this account In a different account

S3 path

Browse for or enter an existing S3 path.

All folders and files contained in the S3 paths are crawled. For example, type `s3://MyBucket/MyFolder/` to crawl all objects in MyFolder within MyBucket.

Subsequent crawler runs

This field is a global field that affects all S3 data sources.

Crawl all sub-folders Crawl all folders again with every subsequent crawl.

Crawl new sub-folders only Only Amazon S3 folders that were added since the last crawl will be crawled. If the schemas are compatible, they will be added to existing tables.

Crawl based on events Rely on Amazon S3 events to control what folders to crawl.

Sample only a subset of files

Exclude files matching pattern

Cancel **Add an S3 data source**

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

The screenshot shows the AWS Glue interface with the 'Crawlers' section selected. The 'Add crawler' button is being used to start a new crawler. The process is at Step 3: 'Choose data sources and classifiers'. A table lists the data sources to be crawled, with one entry for an S3 bucket named 's3://s3bucketforawsgluedemo/'.

Configure the security settings by creating a new IAM Role

The screenshot shows the 'Configure security settings' step of the crawler creation process. It displays an 'Existing IAM role' dropdown containing 'AWSGlueServiceRole-gowtham'. Below this, there are sections for 'Lake Formation configuration - optional' and 'Security configuration - optional'.

IAM role has been successfully updated

The screenshot shows the 'Set output and scheduling' step. Under 'Output configuration', the 'Target database' is set to 'databaseinawsglue'. The 'Crawler schedule' section indicates the crawler runs 'On demand'.

Specify the target database and the crawler schedule

The screenshot shows the AWS Glue 'Add crawler' wizard. The current step is 'Step 4: Set output and scheduling'. In the 'Output configuration' section, the 'Target database' dropdown is set to 'databaseinawsglue'. There is a 'Table name prefix - optional' input field which is currently empty. Below that is a 'Maximum table threshold - optional' input field containing 'Type a number greater than 0'. Under 'Crawler schedule', the 'Frequency' dropdown is set to 'On demand'. At the bottom right, there are 'Cancel', 'Previous', and 'Next' buttons.

The crawler has started and is in the Running state

The screenshot shows the AWS Glue 'Crawlers' page. A green banner at the top says 'Crawler successfully starting. The following crawler is now starting: "crawler1"'. The main table shows one crawler named 'crawler1' with a state of 'Running'. The table includes columns for Name, State, Schedule, Last run, Last run timestamp, Log, and Table changes from last run. At the top right, there are buttons for Action, Run, and Create crawler.

The crawler state changes to the Ready State

The screenshot shows the AWS Glue 'Crawlers' page. A green banner at the top says 'Crawler successfully starting. The following crawler is now starting: "crawler1"'. The main table shows one crawler named 'crawler1' with a state of 'Ready'. The table includes columns for Name, State, Schedule, Last run, Last run timestamp, Log, and Table changes from last run. The 'Last run' column shows 'Succeeded' and the 'Last run timestamp' is 'June 25, 2023 at 19:10:18'. A 'View log' button is located in the 'Log' column. At the top right, there are buttons for Action, Run, and Create crawler.

Now the data has been fetched from S3 using crawler

The screenshot shows the AWS Glue Tables page. At the top, a green banner displays the message "Crawler successfully starting" and "The following crawler is now starting: 'crawler1'". Below the banner, the "Tables" section is visible, showing a single table named "s3bucketforawsgluedemo" located in the "databaseinawsglue" database. The table has a CSV classification and was last updated on June 25, 2023, at 19:12:39. There are buttons for "Delete", "Data quality New", "Add tables using crawler", and "Add table". The left sidebar includes sections for Data Catalog Tables, Data Catalog, Data Integration and ETL, and Legacy pages.

Click on the table to view the table details

The screenshot shows the "Table details" tab for the "s3bucketforawsgluedemo" table. It displays basic information such as Name, Description, Database, Classification, Location, Connection, Deprecated, and Last updated. Under the "Schema" tab, it shows the table's schema with four columns: last name, first name, ssn, and test1, all of type string. The "Edit schema as JSON" and "Edit schema" buttons are visible at the bottom right of the schema table.

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

This screenshot is identical to the previous one, showing the "Table details" tab for the "s3bucketforawsgluedemo" table. It displays the same schema information, including the four columns: last name, first name, ssn, and test1, all of type string. The "Edit schema as JSON" and "Edit schema" buttons are visible at the bottom right of the schema table.