BLACKBUCKS INTERNSHIP REPORT

Data Flow from AWS- In house (Cloud9) to Third Party (GitHub)

SUBMITTED BY

MOVVA NAGA SAHITHI 20B91A5438
SEELAM SATYA SAI SHIVA RAMA KRISHNA 21B95A5405
TANGIRALA KRISHNA PRIYANKA 20B91A05U0

UNDER THE GUIDANCE OF MR. AASHU DEV
B.Tech , AWS Accredited Solution Architect (2x) , AWS Academy educator

Blackbuck Engineers Pvt. Ltd
Road No 36, Jubilee Hills, Hyderabad

BLACKBUCK INTERNSHIP WORK

Team Members:

- **MOVVA NAGA SAHITHI (20B91A5438)**
- **SEELAM SATYA SAI SHIVA RAMA KRISHNA (21B95A5405)**
- **TANGIRALA KRISHNA PRIYANKA (20B91A05U0)**

Title:

Using the Code Commit repository, we pull the repository file from the code commit to cloud9 and then push it to third - party repository GitHub.

Abstract:

Amazon Web Services offers a broad set of global cloud-based products including computing, storage, databases, analytics, networking, mobile, developer tools, management tools, IoT, security, and enterprise applications: on-demand, available in seconds, with pay-as-you-go pricing. From data warehousing to deployment tools, directories to content delivery, over 200 AWS services are available. New services can be provisioned quickly, without the upfront fixed expense. This allows enterprises, start-ups, small and medium-sized businesses, and customers in the public sector to access the building blocks they need to respond quickly to changing business requirements. This whitepaper provides an overview of the benefits of the AWS Cloud and introduces you to the services that make up the platform.

In this project “Data Flow from AWS In-House(CodeCommit) to Third Party(GitHub)” we use the CodeCommit repository to pull the repository file from the code commit to cloud9 and then push it to a third-party repository GitHub.

Data Flow from AWS In-House(CodeCommit) to Third Party(GitHub)

Table of Contents

S. No Name **Page No.**

Aim of the Project

Using CodeCommit repository , we pull the repository file from code commit to cloud9 and then push to third - party repository github.

Introduction To Amazon Web Services (AWS):

Amazon Web Services, Inc. (AWS) is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered, pay-as-you-go basis. Oftentimes, clients will use this in combination with autoscaling (a process that allows a client to use more computing in times of high application usage, and then scale down to reduce costs when there is less traffic). These cloud computing web services provide various services related to networking, computing, storage, middleware, IoT and other processing capacity, as well as software tools via AWS server farms. This frees clients from managing, scaling, and patching hardware, and operating systems.

One of the foundational services is Amazon Elastic Compute Cloud (EC2), which allows users to have at their disposal a virtual cluster of computers, with extremely high availability, which can be interacted with over the internet via REST APIs, a CLI or the AWS console. AWS's virtual computers emulate most of the attributes of a real computer, including hardware central processing units (CPUs) and graphics processing units (GPUs) for processing; local/RAM memory; hard disk /SSD storage; a choice of operating systems; networking; and pre-loaded application software such as web servers, databases, and customer relationship management (CRM).

AWS services are delivered to customers via a network of AWS server farms located throughout the world. Fees are based on a combination of usage (known as a "Pay-as-you-go" model), hardware, operating system, software, or networking features chosen by the subscriber required availability, redundancy, security, and service options. Subscribers can pay for a single virtual AWS computer, a dedicated physical computer, or clusters of either.

Amazon markets AWS to subscribers as a way of obtaining large-scale computing capacity more quickly and cheaply than building an actual physical server farm. All services are billed based on usage, but each service measures usage in varying ways. As of 2021 Q4, AWS has 33% market share for cloud infrastructure while the next two competitors Microsoft Azure and Google Cloud have 21%, and 10% respectively, according to Synergy Group.

Moreover, AWS offers services like Amazon Sage Maker for machine learning model development and deployment, and Amazon Elastic Beanstalk for simplified application deployment and management. With its global presence and numerous data centers worldwide, AWS enables users to deploy their applications closer to their target audience, resulting in improved performance and reliability. Overall, AWS is a powerful and extensively utilized cloud platform that empowers individuals, businesses, and organizations to leverage the benefits of cloud computing for their diverse needs.

Why AWS?

Easy to use:

AWS is designed to allow application providers, ISVs, and vendors to host your applications quickly and securely – whether an existing application or a new SaaS-based application. You can use the AWS Management Console or well-documented web services APIs to access AWS's application hosting platform.

Flexible:

AWS enables you to select the operating system, programming language, web application platform, database, and other services you need. With AWS, you receive a virtual environment that lets you load the software and services your application requires. This eases the migration process for existing applications while preserving options for building new solutions.

Cost-effective:

You pay only for the compute power, storage, and other resources you use, with no long-term contracts or up-front commitments. For more information on comparing the costs of other hosting alternatives with AWS, see the AWS Economics Center.

Reliable:

With AWS, you take advantage of a scalable, reliable, and secure global computing infrastructure, the virtual backbone of Amazon.com's multi-billion-dollar online business that has been honed for over a decade.

Scalable and High performance:

Using AWS tools, Auto Scaling, and Elastic Load Balancing, your application can scale up or down based on demand. Backed by Amazon's massive infrastructure,

you have access to compute and storage resources when you need them.

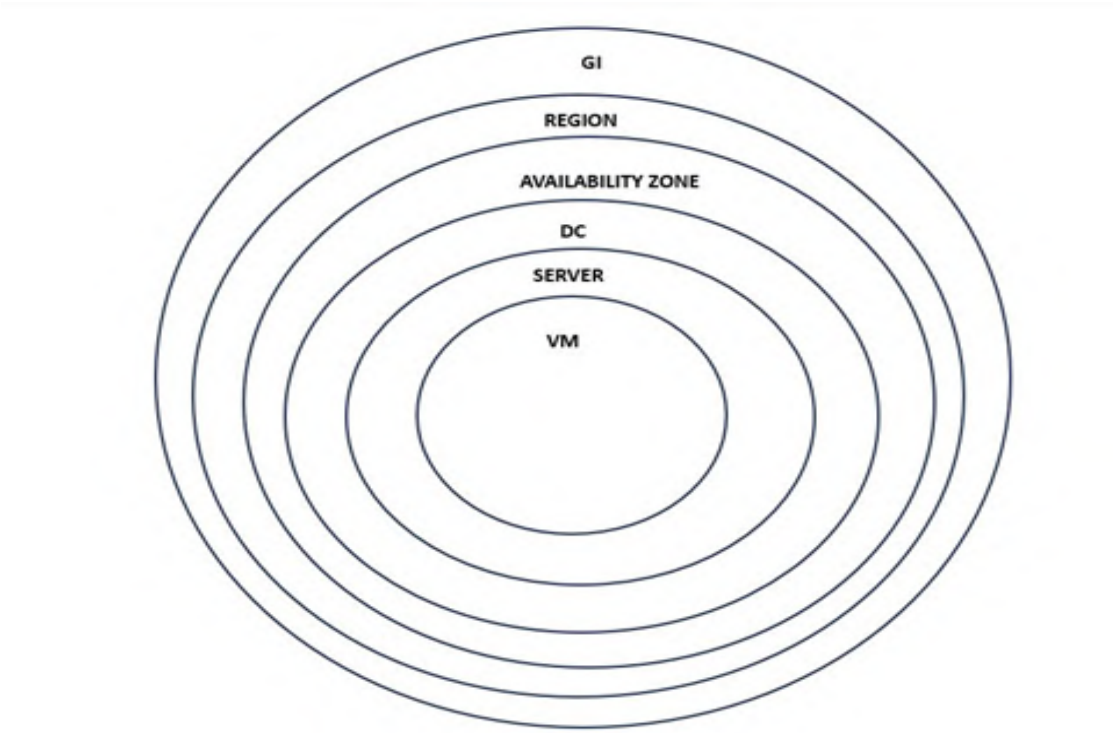
Secure:

Using AWS tools, Auto Scaling, and Elastic Load Balancing, your application can scale up or down based on demand. Backed by Amazon's massive infrastructure, you have access to compute and storage resources when you need them.

Managed Services:

AWS 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.

AWS Global Infrastructure



1. Regions: AWS regions are geographic areas where AWS data centers are located. Each region is designed to be isolated from other regions to provide fault tolerance and resilience. AWS regions are further divided into Availability Zones.

2. Availability Zones (AZs): Availability Zones are physically separate data centers within a region. They are connected via high-speed, low-latency networks. The purpose of AZs is to provide redundancy and ensure that if one AZ goes down, services can automatically failover to another AZ within the same region.

3. Global Accelerator: AWS Global Accelerator is a service that utilizes the AWS global network to improve the performance and availability of your applications. It

automatically routes traffic to the nearest AWS edge location, reducing latency and optimizing global network performance.

4. Direct Connect: AWS Direct Connect provides dedicated network connections between on-premises environments and AWS. These connections bypass the public internet, ensuring reliable and secure connectivity. Direct Connect locations are available in various regions around the world.

5. Regional Services: Some AWS services are region-specific, meaning they are available only in certain regions due to regional requirements or limitations. For example, certain compliance-related services or government-specific services may be available only in specific regions.

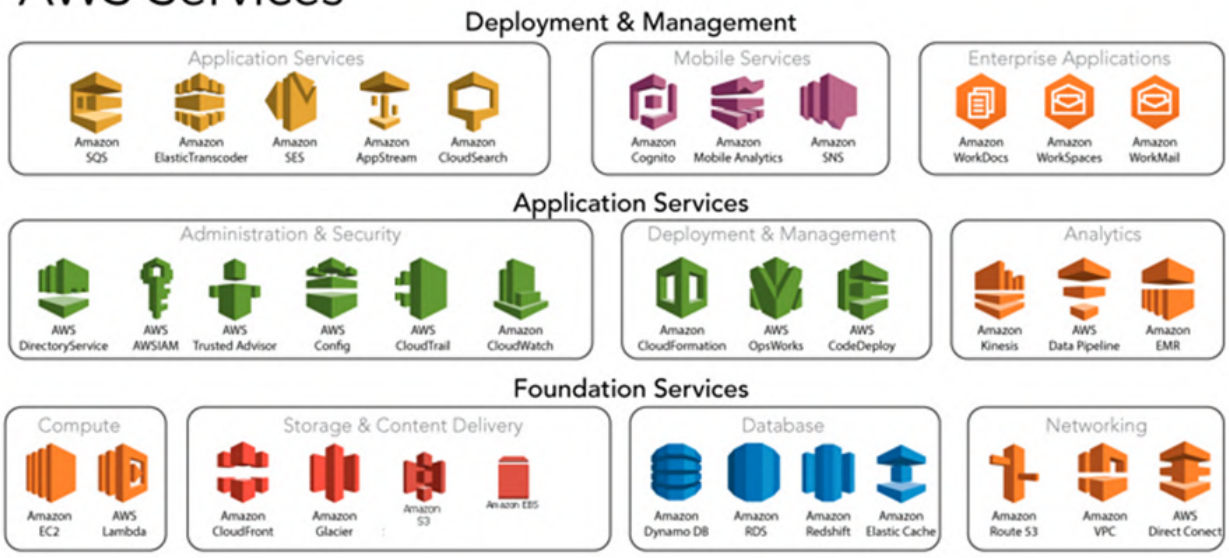
List of AWS Services:

Amazon, the preeminent cloud vendor, broke new ground by establishing the first cloud computing service, Amazon EC2, in 2008. AWS offers more solutions and features than any other provider and has free tiers with access to the AWS Console, where users can centrally control their ministries.

Designed around ease-of-use for various skill sets, AWS is tailored for those unaccustomed to software development utilities. Web applications can be deployed in minutes with AWS facilities, without provisioning servers or writing additional code.

- Amazon EC2 (Elastic Compute Cloud)
- Amazon S3 (Simple Storage Service)
- Amazon Lambda
- Amazon DynamoDB
- Amazon SNS (Simple Notification Service)
- Amazon VPC (Virtual Private Cloud)
- Amazon CloudWatch
- Amazon Cloud9
- Amazon Elastic Beanstalk
- Amazon RDS (Relational Database Services)

AWS Services



Top 7 Most Used AWS Services

Amazon EC2:

Amazon Elastic Compute Cloud (EC2) is a part of Amazon.com's cloud-computing platform, Amazon Web Services (AWS), that allows users to rent virtual computers on which to run their own computer applications. EC2 encourages scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image (AMI) to configure a virtual machine, which Amazon calls an "instance", containing any software desired. A user can create, launch, and terminate server-instances as needed, paying by the second for active servers – hence the term "elastic". EC2 provides users with control over the geographical location of instances that allows for latency optimization and high levels of redundancy. In November 2010, Amazon switched its own retail website platform to EC2 and AWS.

Amazon announced a limited public beta test of EC2 on August 25, 2006, offering access on a first-come, first-served basis. Amazon added two new instance types (Large and Extra-Large) on October 16, 2007. On May 29, 2008, two more types

were added, High-CPU Medium and High-CPU Extra Large. There were twelve types of instances available.

Amazon added three new features on March 27, 2008, static IP addresses, availability zones, and user selectable kernels. On August 20, 2008, Amazon added Elastic Block Store (EBS) This provides persistent storage, a feature that had been lacking since the service was introduced.

Instance types:

Initially, EC2 used Xen virtualization exclusively. However, on November 6, 2017, Amazon announced the new C5 family of instances that were based on a custom architecture around the KVM hypervisor, called Nitro. Each virtual machine, called an "instance", functions as a virtual private server. Amazon sizes instances based on "Elastic Compute Units". The performance of otherwise identical virtual machines may vary.

General Purpose	Compute Optimised	Memory Optimised	Accelerated Computing	Storage Optimised
A1 ARM based core and custom silicon	C4 Compute - CPU intensive apps and DBs	R4 RAM - Memory Intensive apps and DB's	P2 Processing optimised - Machine Learning	H1 High Disk Throughput - Big data clusters
T2 Tiny - Web servers and small DBs		X1 Xtreme RAM - For SAP/Spark	G3 Graphics Intensive - Video and Streaming	I3 IOPS - No SQL DBs
M4 Main - App servers and general purpose		z1d High Compute and High Memory - Gaming	F1 Field Programmable - Hardware acceleration	D2 Dense Storage - Data Warehousing

Amazon S3:

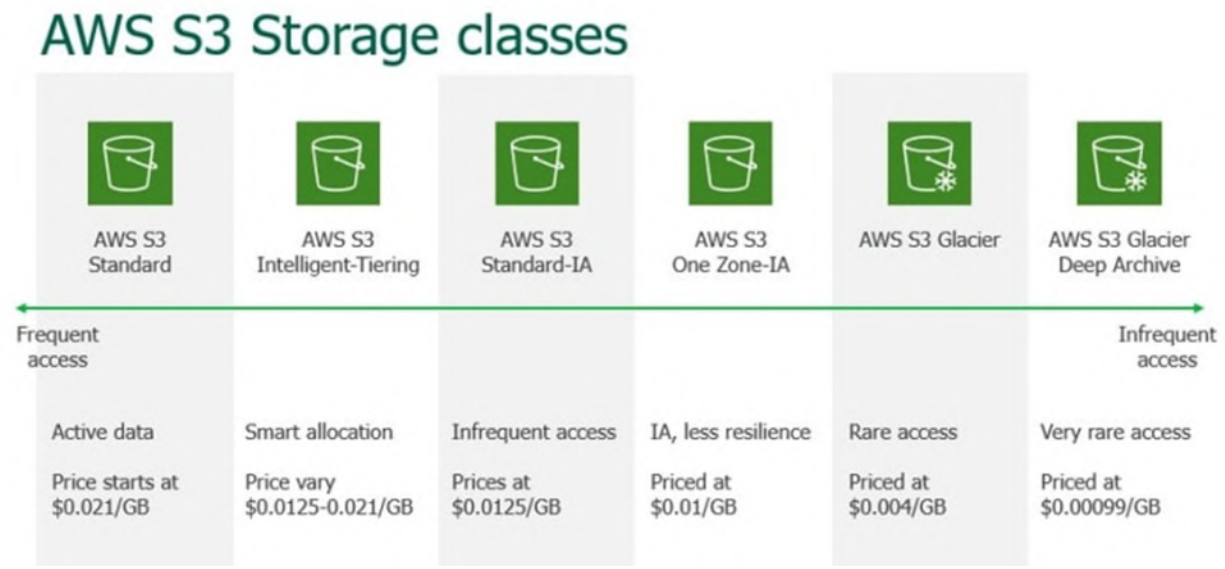
Amazon S3 manages data with an object storage architecture which aims to provide scalability, high availability, and low latency with high durability. The basic storage units of Amazon S3 are objects which are organized into buckets. Each object is identified by a unique, user-assigned key. Buckets can be managed using the console provided by Amazon S3, programmatically with the AWS SDK, or the REST application programming interface.

Objects can be up to five terabytes in size. Requests are authorized using an access control list associated with each object bucket and support versioning which is disabled by default. Since buckets are typically the size of an entire file system mount in other systems, this access control scheme is very coarse-grained. In other words, unique access controls cannot be associated with individual files. [citation needed] Amazon S3 can be used to replace static web-hosting infrastructure with HTTP client-accessible objects, index document support and error document support.

The Amazon AWS authentication mechanism allows the creation of authenticated URLs, valid for a specified amount of time. Every item in a bucket can also be served as a BitTorrent feed. The Amazon S3 store can act as a seed host for a torrent and

any BitTorrent client can retrieve the file. This can drastically reduce the bandwidth cost for the download of popular objects. A bucket can be configured to save HTTP log information to a sibling bucket; this can be used in data mining operations.

There are various User Mode File System (FUSE)–based file systems for Unix-like operating systems (for example, Linux) that can be used to mount an S3 bucket as a file system. The semantics of the Amazon S3 file system is not that of a POSIX file system, so the file system may not behave entirely as expected.

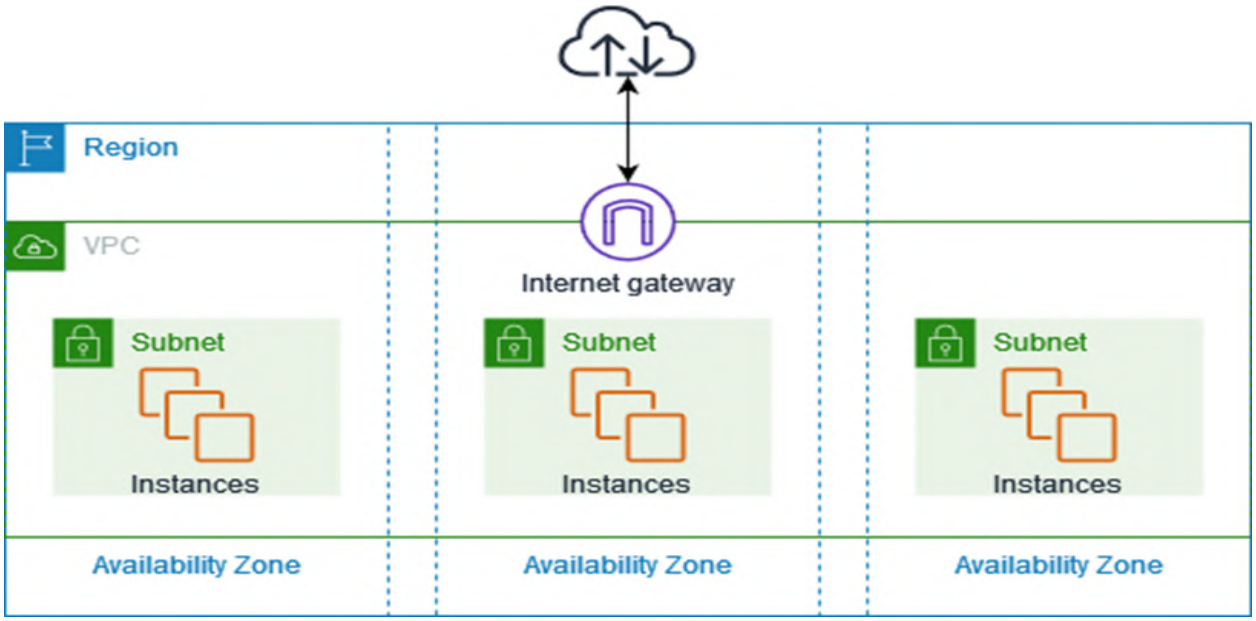


Amazon VPC:

Amazon Virtual Private Cloud (VPC) is a commercial cloud computing service that provides a virtual private cloud, by provisioning a logically isolated section of Amazon Web Services (AWS) Cloud. Enterprise customers are able to access the Amazon Elastic Compute Cloud (EC2) over an IPsec based virtual private network. Unlike traditional EC2 instances which are allocated internal and external IP numbers by Amazon, the customer can assign IP numbers of their choosing from one or more subnets.

Amazon Web Services launched Amazon Virtual Private Cloud on 26 August 2009, which allows the Amazon Elastic Compute Cloud service to be connected to legacy infrastructure over an IPsec VPN .In AWS, the basic VPC is free to use, with users being charged by usage for additional features.EC2 and RDS instances running in a VPC can also be purchased using Reserved Instances, however will have a limitation on resources being guaranteed.

Google Cloud Platform resources can be provisioned, connected, and isolated in a virtual private cloud (VPC) across all GCP regions. With GCP, VPCs are global resources and subnets within that VPC are regional resources. This allows users to connect zones and regions without the use of additional networking complexity as all data travels, encrypted in transit and at rest, on Google's own global, private network. Identity management policies and security rules allow for private access to Google's storage, big data, and analytics managed services. VPCs on Google Cloud Platform leverage the security of Google's data centers.



AWS Lambda:

AWS Lambda is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, and logging. With Lambda, all you need to do is supply your code in one of the language runtimes that Lambda supports.

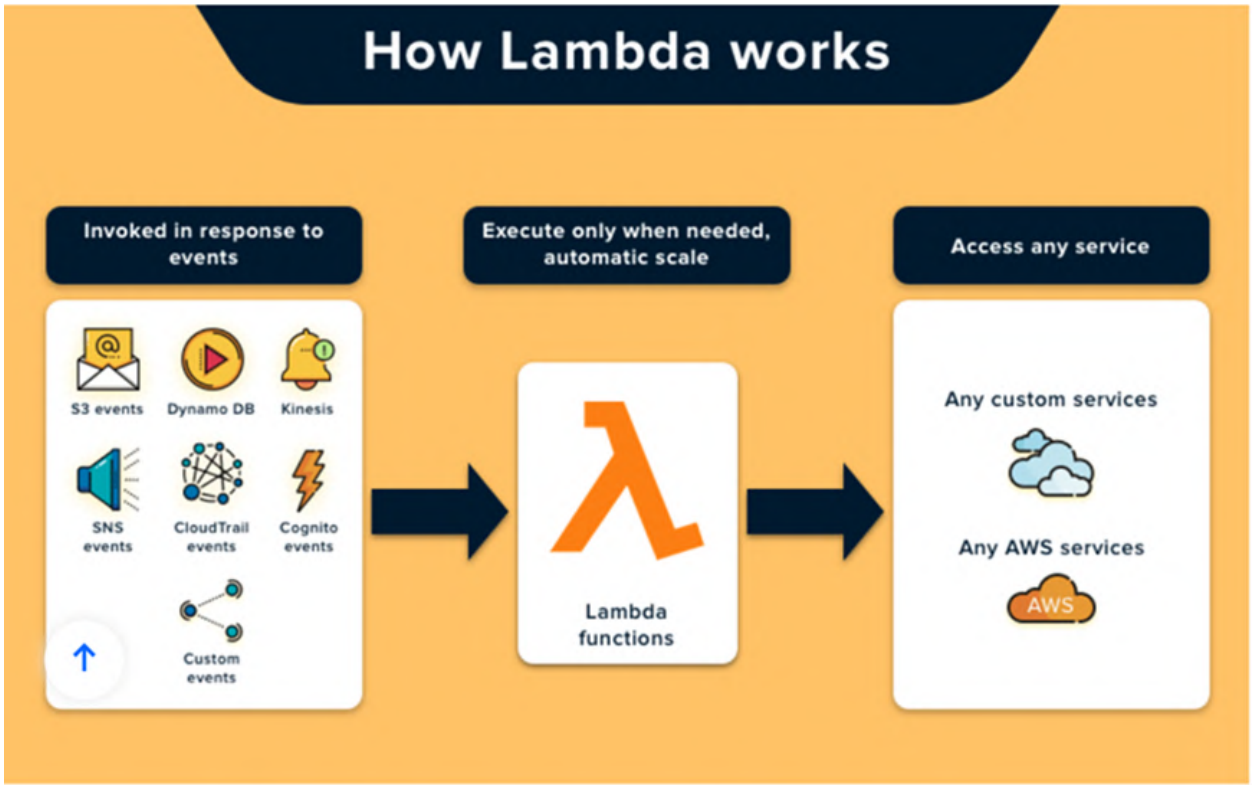
You organize your code into Lambda functions. The Lambda service runs your function only when needed and scales automatically. You only pay for the compute time that you consume—there is no charge when your code is not running.

When using Lambda, you are responsible only for your code. Lambda manages the compute fleet that offers a balance of memory, CPU, network, and other resources to run your code. Because Lambda manages these resources, you cannot log in to compute instances or customize the operating system on provided runtimes.

Lambda performs operational and administrative activities on your behalf, including managing capacity, monitoring, and logging your Lambda functions.

If you do need to manage your compute resources, AWS has other compute services to consider, such as:

· AWS App Runner builds and deploys containerized web applications automatically, balances traffic with encryption, scales to meet your traffic needs, and allows for the configuration of how services are accessed and communicated with other AWS applications in a private Amazon VPC.
· AWS Faregate with Amazon ECS runs containers without having to provision, configure, or scale clusters of virtual machines.
· Amazon EC2 lets you customize the operating system, network and security settings, and the entire software stack. You are responsible for provisioning capacity, monitoring fleet health and performance, and using Availability Zones for fault tolerance.



AWS Elastic BeanStalk:

Amazon Web Services (AWS) comprises over one hundred services, each of which exposes an area of functionality. While the variety of services offers flexibility for how you want to manage your AWS infrastructure, it can be challenging to figure out which services to use and how to provision them.

With Elastic Beanstalk, you can quickly deploy and manage applications in the AWS Cloud without having to learn about the infrastructure that runs those applications. Elastic Beanstalk reduces management complexity without restricting choice or control. You simply upload your application, and Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring.

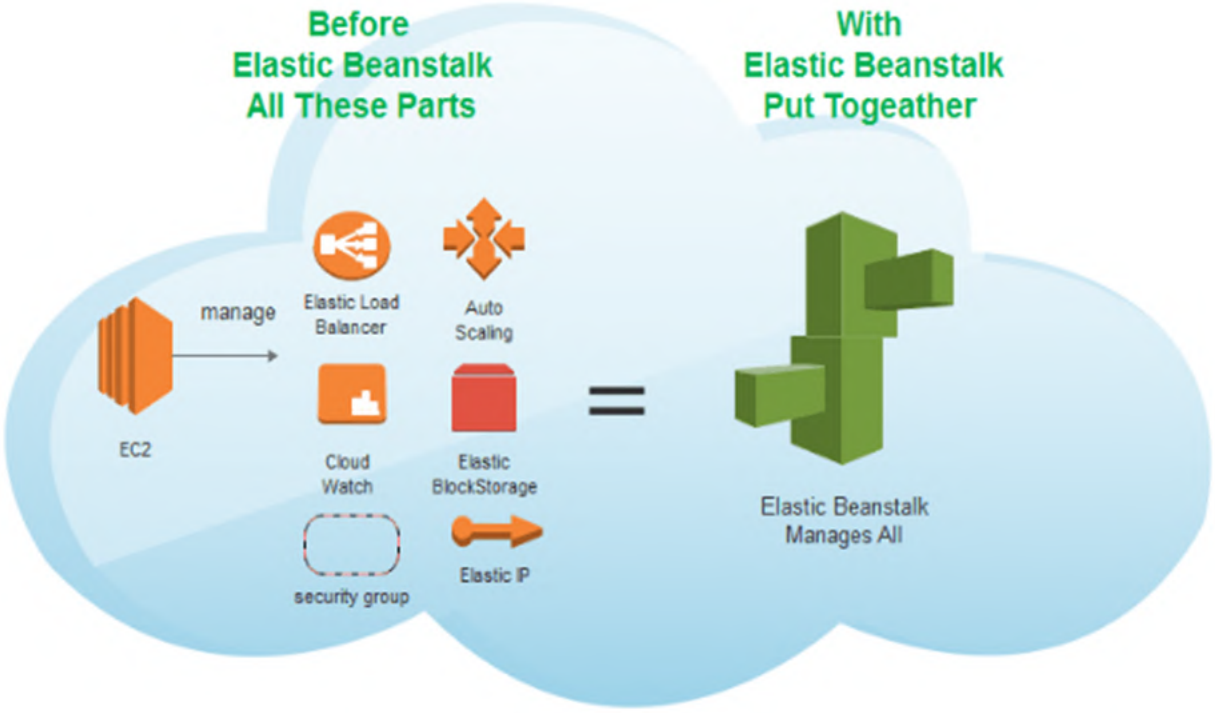
Elastic Beanstalk supports applications developed in Go, Java, .NET, Node.js, PHP, Python, and Ruby. When you deploy your application, Elastic Beanstalk builds the selected supported platform version and provisions one or more AWS resources, such as Amazon EC2 instances, to run your application.

You can interact with Elastic Beanstalk by using the Elastic Beanstalk console, the AWS Command Line Interface (AWS CLI), or **eb**, a high-level CLI designed specifically for Elastic Beanstalk.

To learn more about how to deploy a sample web application using Elastic Beanstalk, see Getting Started with AWS: Deploying a Web App.

You can also perform most deployment tasks, such as changing the size of your fleet of Amazon EC2 instances or monitoring your application, directly from the Elastic Beanstalk web interface (console).

To use Elastic Beanstalk, you create an application, upload an application version in the form of an application source bundle (for example, a Java .war file) to Elastic Beanstalk, and then provide some information about the application. Elastic Beanstalk automatically launches an environment and creates and configures the AWS resources needed to run your code. After your environment is launched, you can then manage your environment and deploy new application versions. The following diagram illustrates the workflow of Elastic Beanstalk.



Amazon IAM:

IAM provides the infrastructure necessary to control authentication and authorization for your AWS account. The IAM infrastructure is illustrated by the following diagram.

First, a human user or an application uses their sign-in credentials to authenticate with AWS. Authentication is provided by matching the sign-in credentials to a principal (an IAM user, federated user, IAM role, or application) trusted by the AWS account.

Next, a request is made to grant the principal access to resources. Access is granted in response to an authorization request. For example, when you first sign into the console and are on the console home page, you are not accessing a specific service. When you select a service, the request for authorization is sent to that service and it looks to see if your identity is on the list of authorized users, what policies are being enforced to control the level of access granted, and any other policies that might be in effect. Authorization requests can be made by principals within your AWS account or from another AWS account that you trust.

Once authorized, the principal can take action or perform operations on resources in your AWS account. For example, the principal could launch a new Amazon Elastic Compute Cloud instance, modify IAM group membership, or delete Amazon Simple Storage Service buckets.

IAM USES

o User Management: IAM allows you to create and manage individual users within your AWS account. You can assign unique access credentials (username and password) to each user, control their permissions, and enable multi-factor authentication for added security. IAM users are used to authenticate and authorize individuals who interact with your AWS resources.

o Group Management: IAM allows you to create groups and add IAM users to these groups. Groups simplify the management of permissions by allowing you to assign permissions to a group of users instead of individually managing permissions for each user. This makes it easier to manage access for teams or departments within your organization.

o Role-Based Access Control: IAM supports role-based access control (RBAC), where you can define roles with specific permissions and assign these roles to IAM users or AWS services. Roles are temporary access credentials that allow users or services to assume temporary permissions when needed. This is commonly used for granting access to AWS resources to external services or for cross-account access.

o Access Key Management: IAM enables the generation and management of access keys (access key ID and secret access key) that are used for programmatic access to AWS resources via the AWS Command Line Interface (CLI), SDKs, or API calls. IAM allows you to create, rotate, and revoke access keys for users to ensure secure programmatic access.

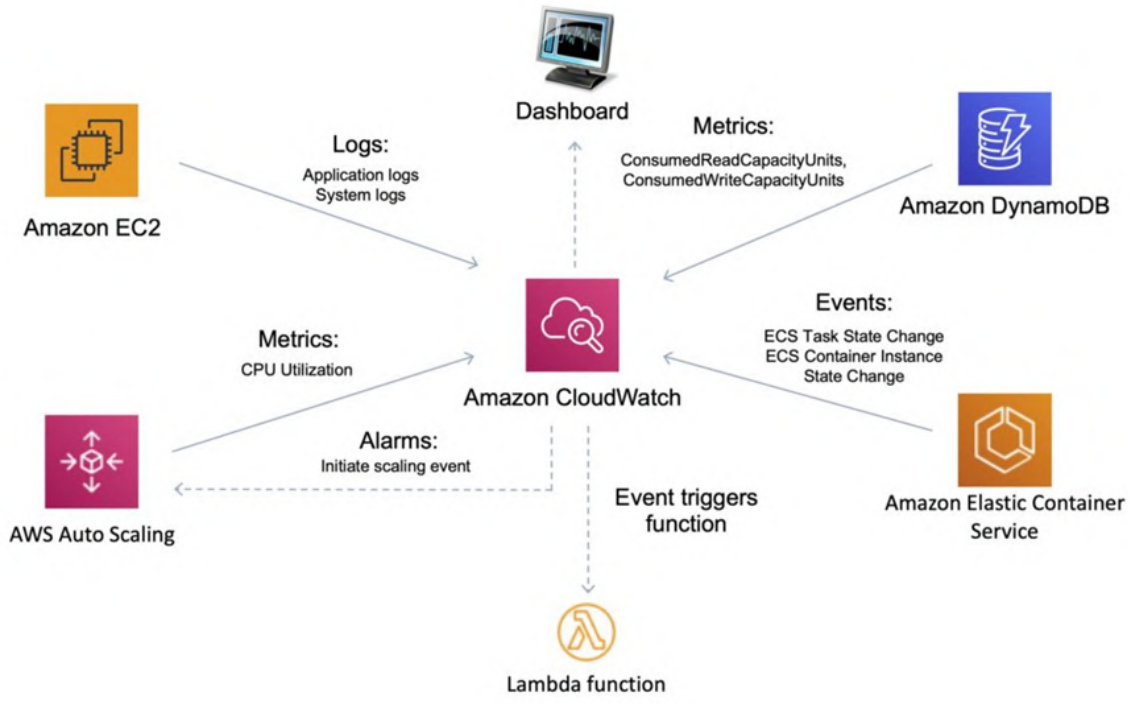
AWS CloudWatch

Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.

The CloudWatch home page automatically displays metrics about every AWS service you use. You can additionally create custom dashboards to display metrics about your custom applications and display custom collections of metrics that you choose.

You can create alarms that watch metrics and send notifications or automatically make changes to the resources you are monitoring when a threshold is breached. For example, you can monitor the CPU usage and disk reads and writes of your Amazon EC2 instances and then use that data to determine whether you should launch additional instances to handle increased load. You can also use this data to stop underused instances to save money.

With CloudWatch, you gain system-wide visibility into resource utilization, application performance, and operational health.



Services Used

 - Ec2 Instance

 -IAM User

 -CodeCommit

 - Cloud9

Amazon EC2 instance

An Amazon EC2 instance is a virtual server in Amazon's Elastic Compute Cloud (EC2) for running applications on the Amazon Web Services (AWS) infrastructure. AWS is a comprehensive, evolving cloud computing platform; EC2 is a service that enables business subscribers to run application programs in the computing environment. It can serve as a practically unlimited set of virtual machines (VMs).

Amazon provides various types of instances with different configurations of CPU, memory, storage and networking resources to suit user needs. Each type is available in various sizes to address specific workload requirements.

Instances are created from Amazon Machine Images (AMI). The machine images are like templates. They are configured with an operating system (OS) and other software, which determine the user's operating environment. Users can select an AMI provided by AWS, the user community or through the AWS Marketplace. Users also can create their own AMIs and share them.

IAM User

An IAM User is an entity created in AWS that provides a way to interact with AWS resources.

The main purpose of IAM Users is that they can sign in to the AWS Management Console and can make requests to the AWS services.

The newly created **IAM users** have no password and no access key. If a user wants to use the AWS resources using the AWS Management Console, you need to create the user password. If a user wants to interact using the AWS programmatically (using the CLI (Command Line Interface)), you need to create the access key for that user. The credentials created for IAM User are what exactly uniquely identify themselves to AWS.

The security of the user's credentials can be enhanced by using the feature, i.e., Multi-Factor Authentication.

The newly created IAM Users do not have permissions, i.e., they are not authorized to access the AWS resources.

An advantage of using individual IAM Users is that you can assign the permissions individually. You can even assign the administrative permissions, who can administer your AWS resources and also administer other IAM Users.

Mainly, the user's permissions are set to AWS tasks and resources, i.e., the job assigned to the IAM User. For example, you create an IAM User whose name is Advita, you create a password for the user and set the permissions that let her start Amazon EC2 instances and read the data from Amazon RDS database.

Each IAM User is associated with one and only one AWS account.

Users are defined within your account, so users do not have to do payment. Any AWS activity performed by a user is billed to your account.

Codecommit

CodeCommit is a secure, highly scalable, managed source control service that hosts private Git repositories. CodeCommit eliminates the need for you to manage your own source control system or worry about scaling its infrastructure. You can use CodeCommit to store anything from code to binaries. It supports the standard functionality of Git, so it works seamlessly with your existing Git-based tools.

With CodeCommit, you can:

- Benefit from a fully managed service hosted by AWS. CodeCommit provides high service availability and durability and eliminates the administrative overhead of managing your own hardware and software. There is no hardware to provision and scale and no server software to install, configure, and update.
- Store your code securely. CodeCommit repositories are encrypted at rest as well as in transit.
- Work collaboratively on code. CodeCommit repositories support pull requests, where users can review and comment on each other's code changes before merging them to branches; notifications that automatically send emails to users about pull requests and comments; and more.
- Easily scale your version control projects. CodeCommit repositories can scale up to meet your development needs. The service can handle

repositories with large numbers of files or branches, large file sizes, and lengthy revision histories.

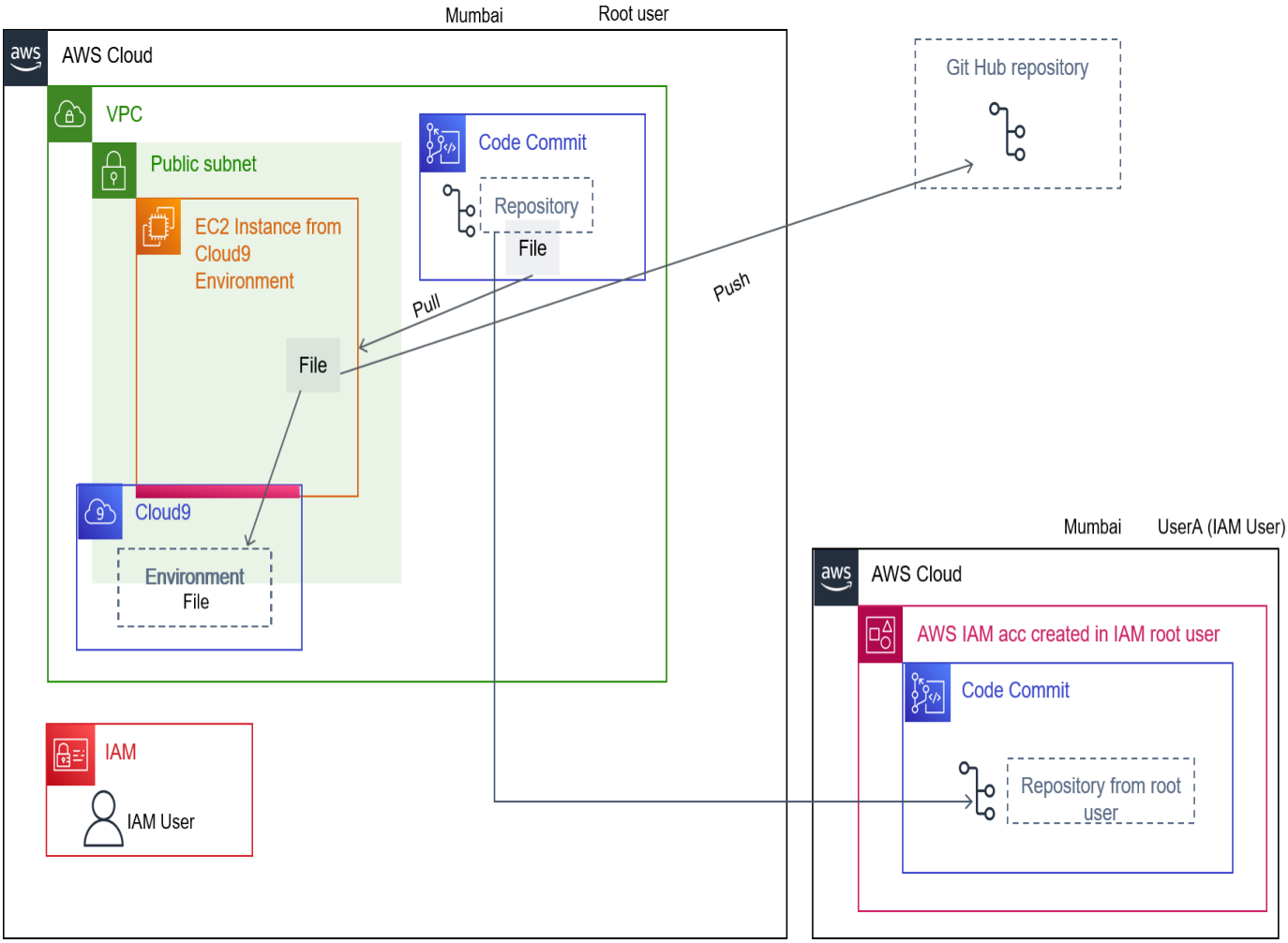
- Store anything, anytime. CodeCommit has no limit on the size of your repositories or on the file types you can store.
- Integrate with other AWS and third-party services. CodeCommit keeps your repositories close to your other production resources in the AWS Cloud, which helps increase the speed and frequency of your development lifecycle. It is integrated with IAM and can be used with other AWS services and in parallel with other repositories. For more information, see Product and service integrations with AWS CodeCommit.
- Easily migrate files from other remote repositories. You can migrate to CodeCommit from any Git-based repository.
- Use the Git tools you already know. CodeCommit supports Git commands as well as its own AWS CLI commands and APIs.

Features of CodeCommit

1. Users don't need to worry about provisioning hardware, server, scaling or software which must be installed, configured or updated.
2. Eliminates the administrative overhead from the user' end of managing their own software and hardware.
3. Provides highly available service, which is durable.
4. User's code is stored securely, since the Codecommit repositories are encrypted.
5. Software development team members can work collaboratively on the code.

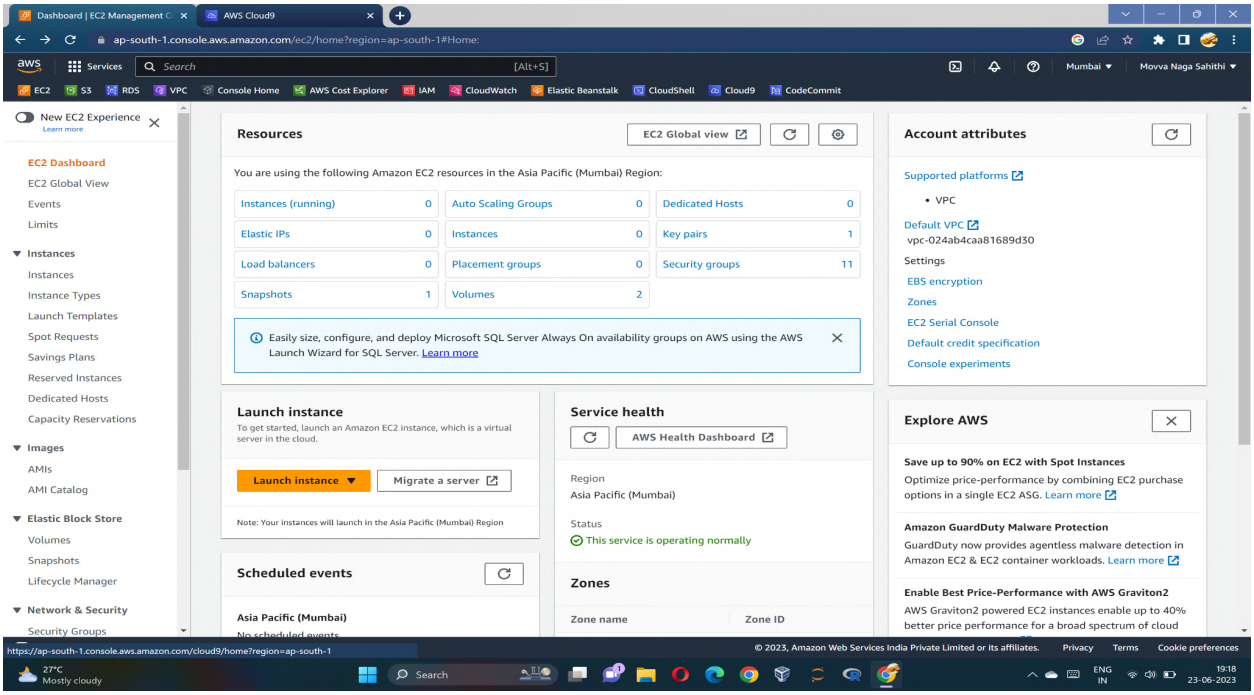
6. Codecommit repositories support pull requests, with the help of which users can review each other's code, comment on other's code and apply the changes required before merging them into branches.
7. There is a provision of email notifications which automatically sends emails to users about pull requests, commits, and comments.
8. It is an easy task to scale the version control for projects. Codecommit can handle repos that have large files or branches that have large sizes and a long revision history.
9. Anything can be stored in Codecommit repositories at any point in time, irrespective of the size of files or file types.
10. Repositories are stored close to production resources in the cloud, due to which the speed and frequency of software development life cycle improves.
11. If data from a remote repository is required, it can be easily migrated from any Git-based repository to Codecommit's repository.
12. Codecommit comes with support for the Git command, and it can be used with AWS CLI and API as well.

Architecture

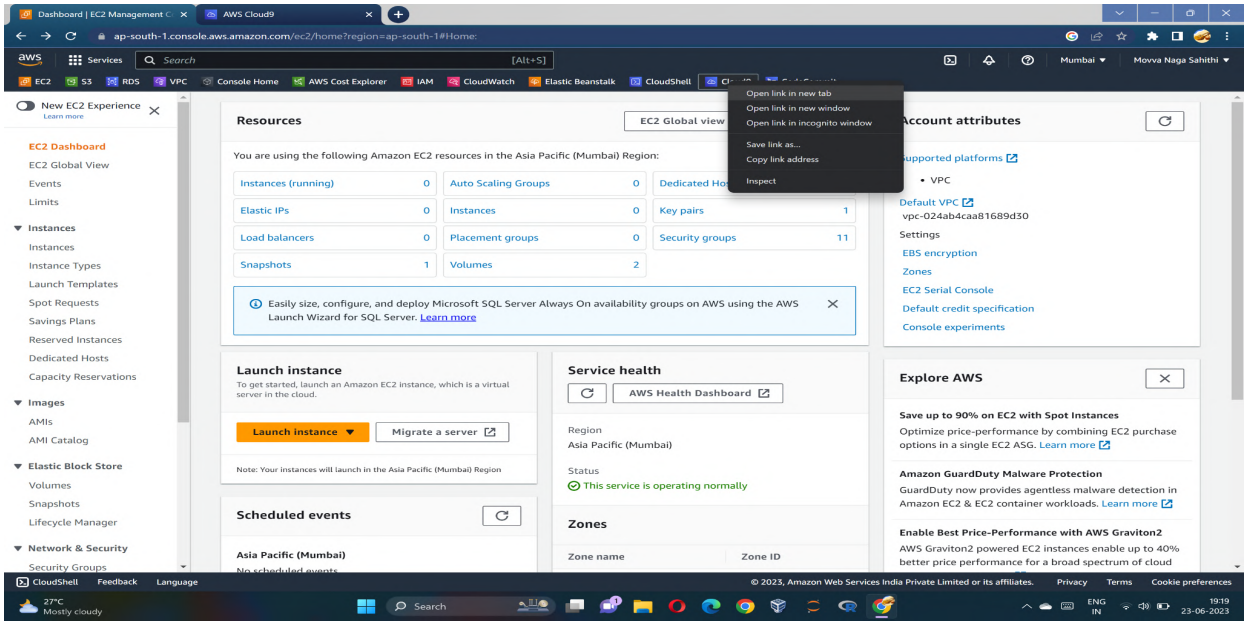


Implementation of the Project

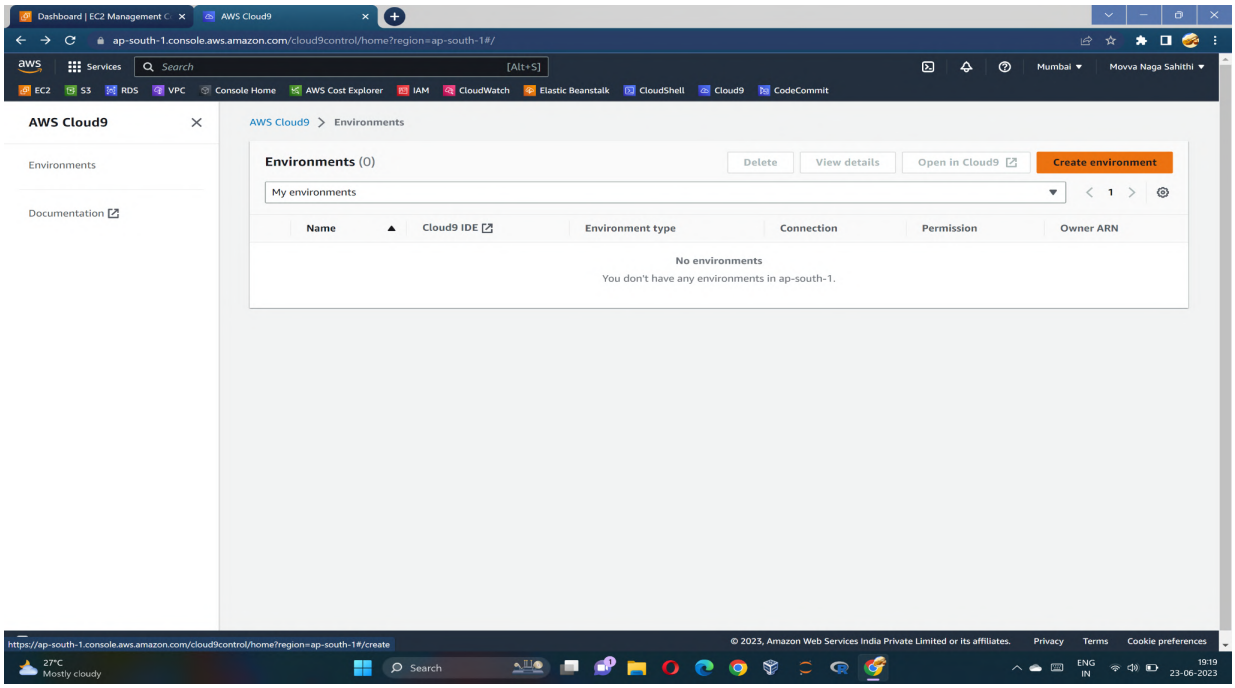
Step 1: Go to your Amazon EC2 Dashboard.



Step 2: Go to your Amazon Cloud9.



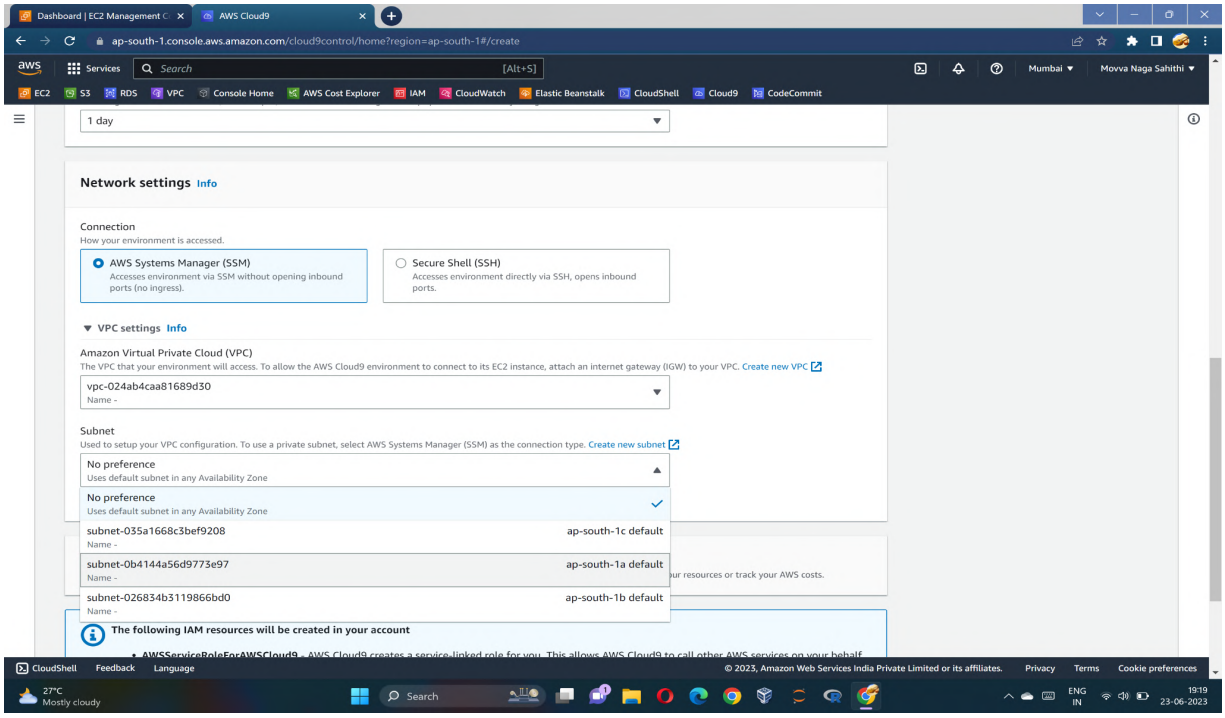
Step 3: Create Environment in Cloud9



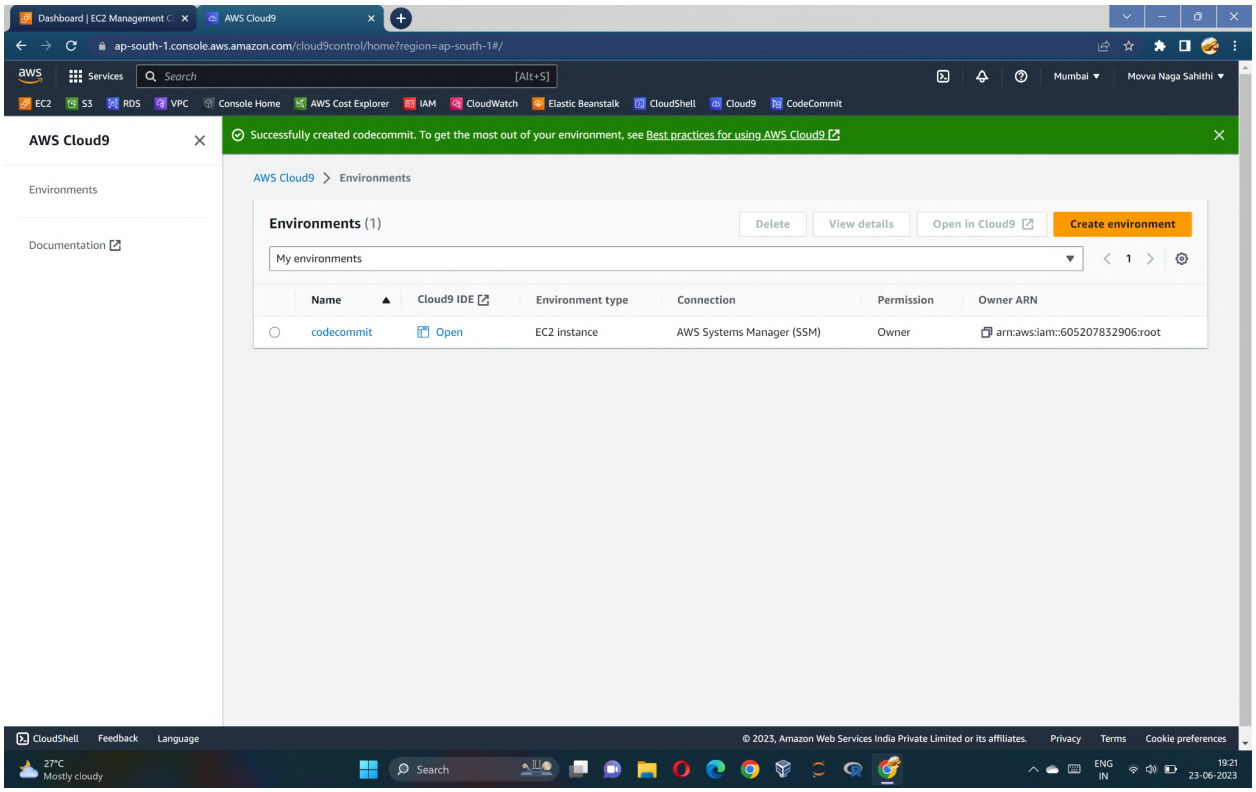
Step 4: Name the environment. While creating the environment , choose the option with which instance is automatically created.

Step 5: Network settings > SSM

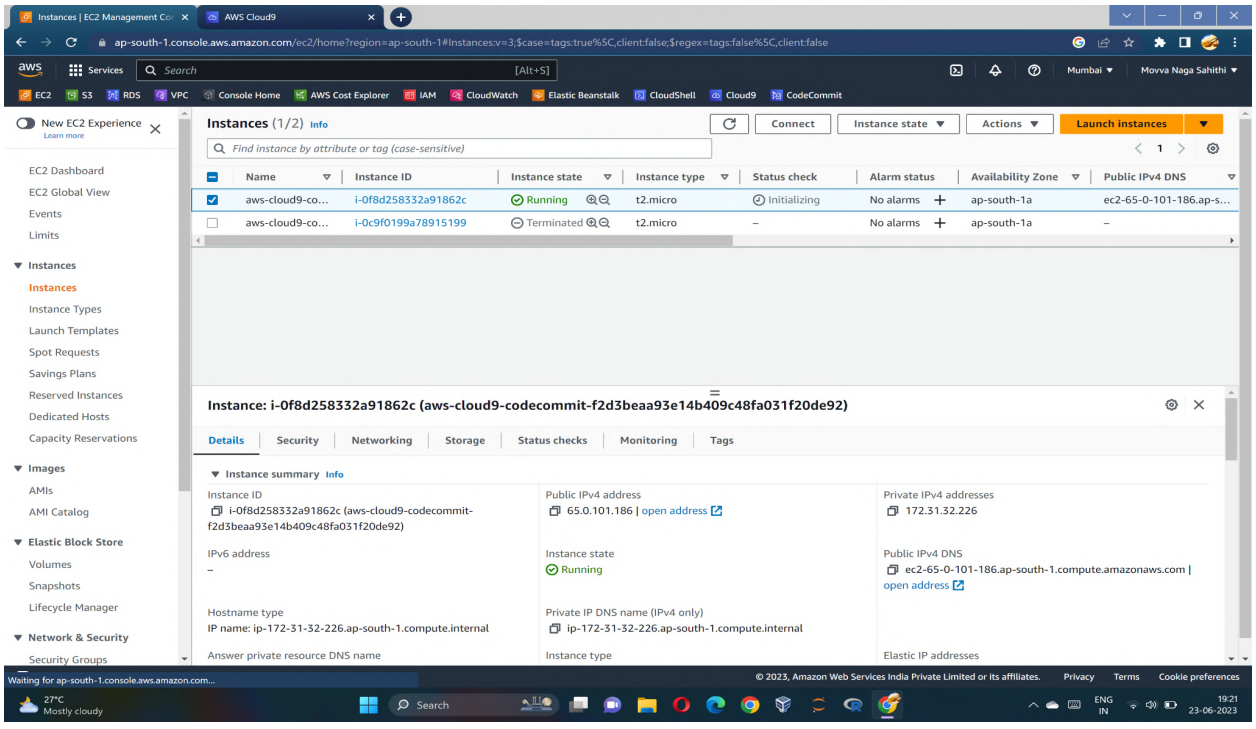
VPC Settings > subnet preference > ap-south-1a



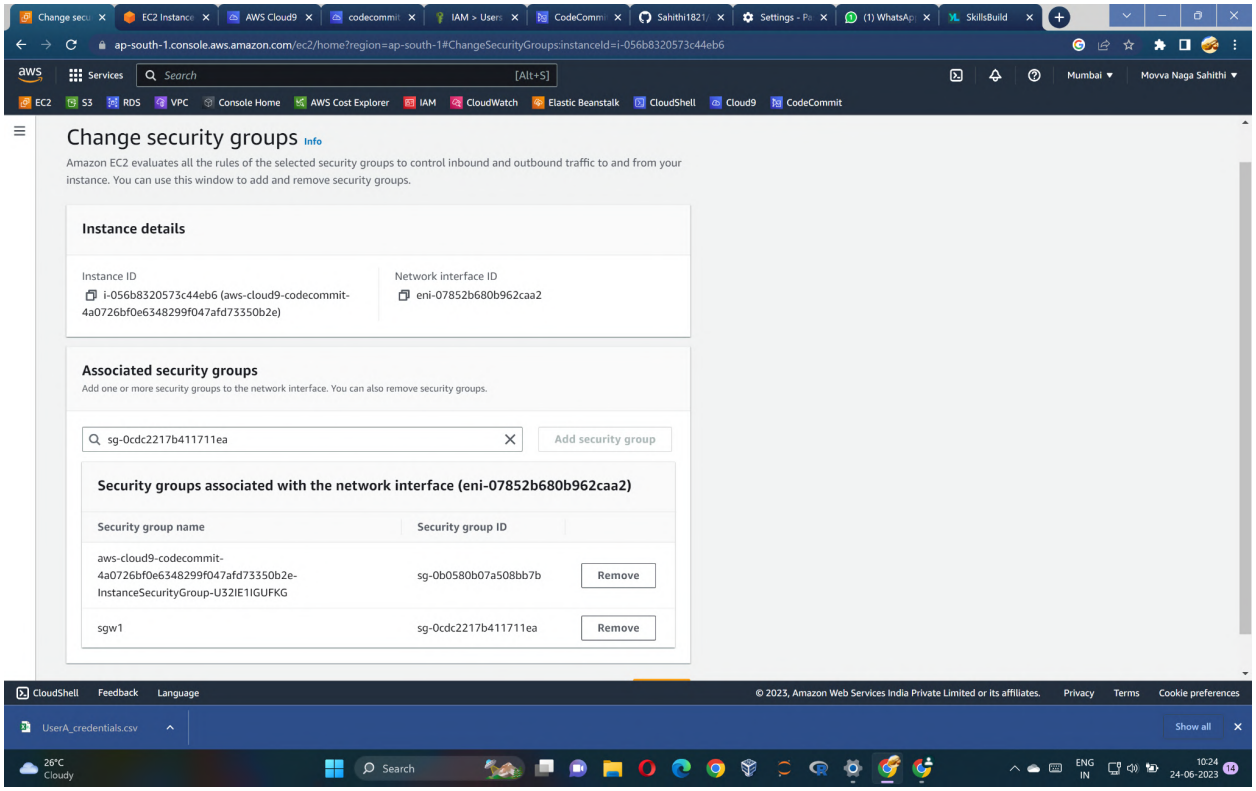
Step 6: Environment is created.



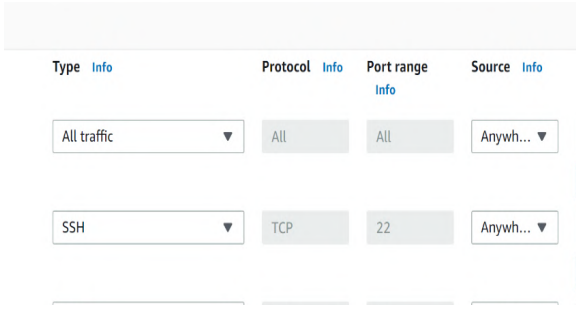
Step 7 : Check the Ec2 dashboard , you can see the instance created from the environment.



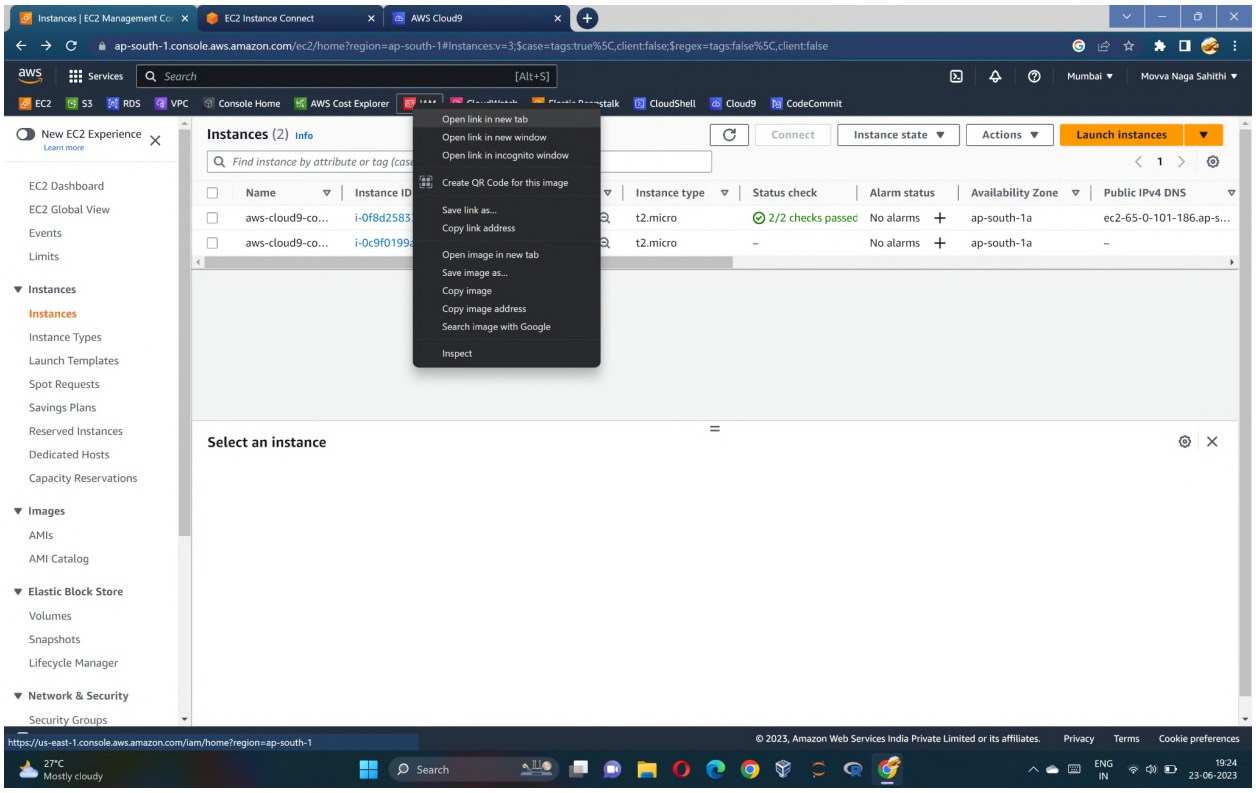
Step 8: Select instance > go to actions > Security > Change security groups > Associated security groups > add a security group.



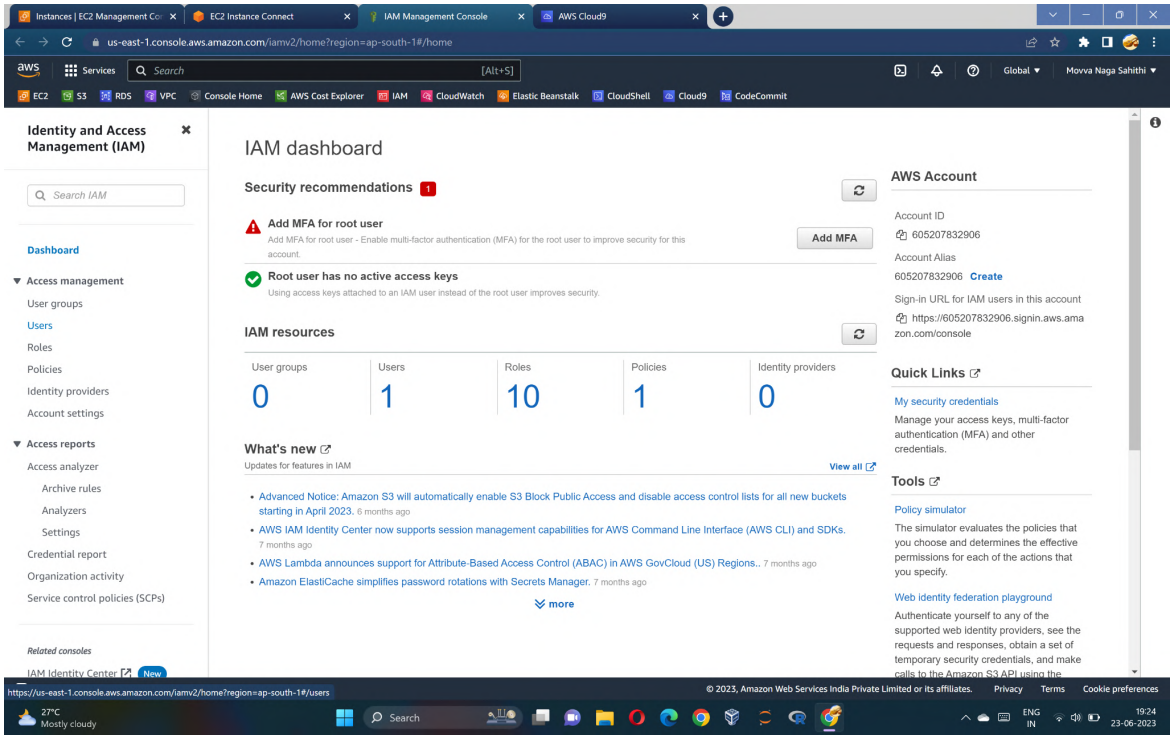
Step 9: Security group rules.



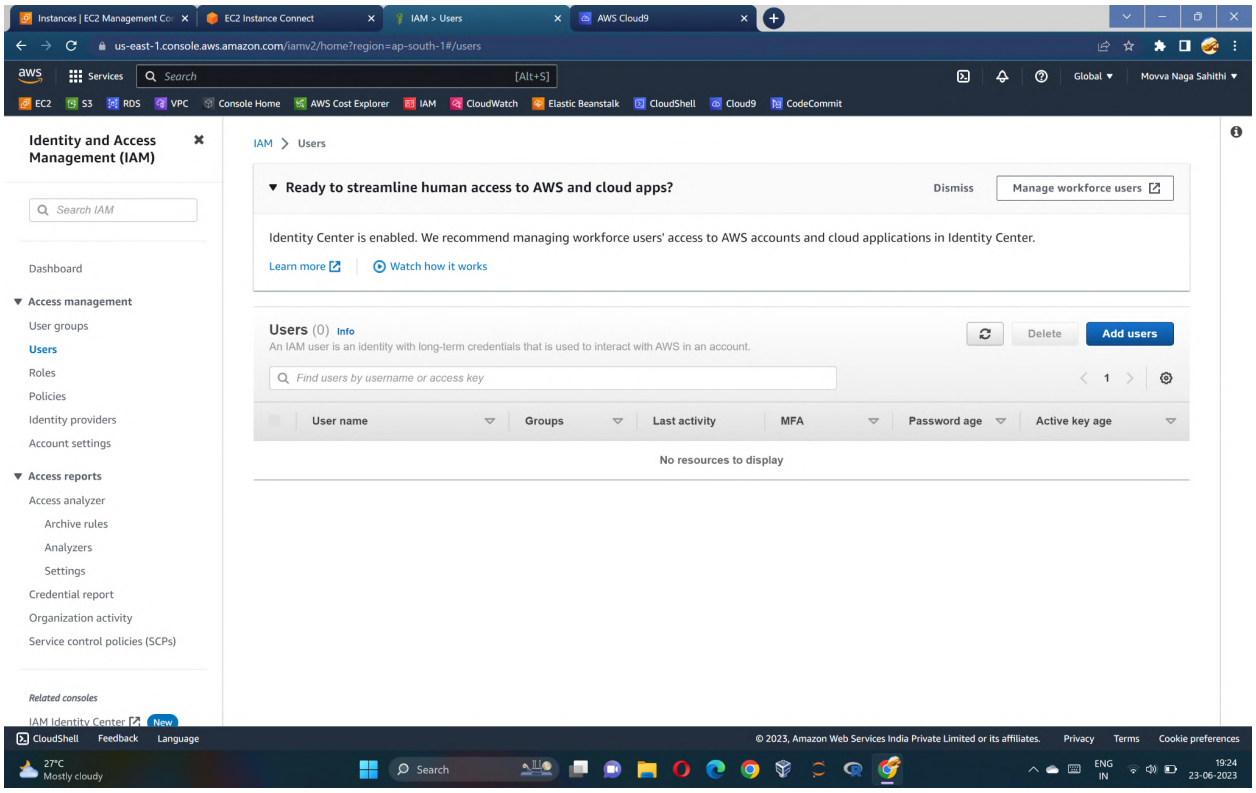
Step 10: Now create an IAM user in the IAM Dashboard.



Step 11 : IAM Dashboard.

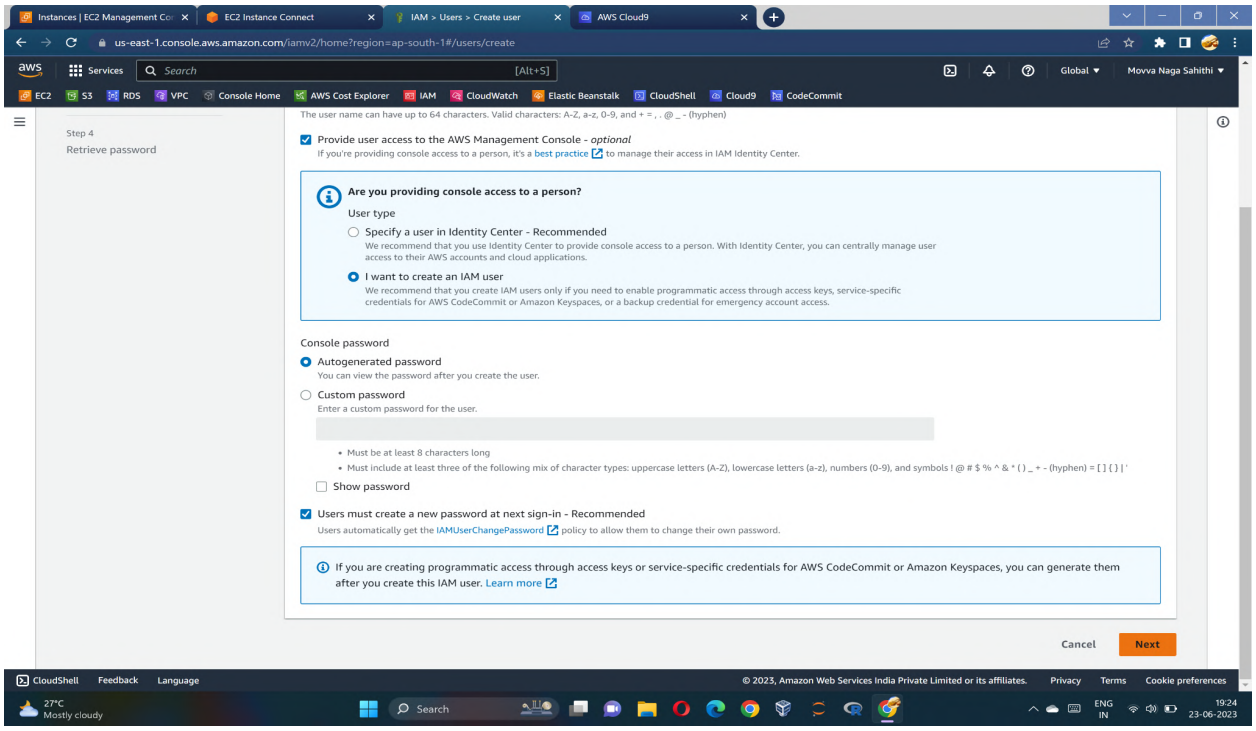


Step 12: Add User

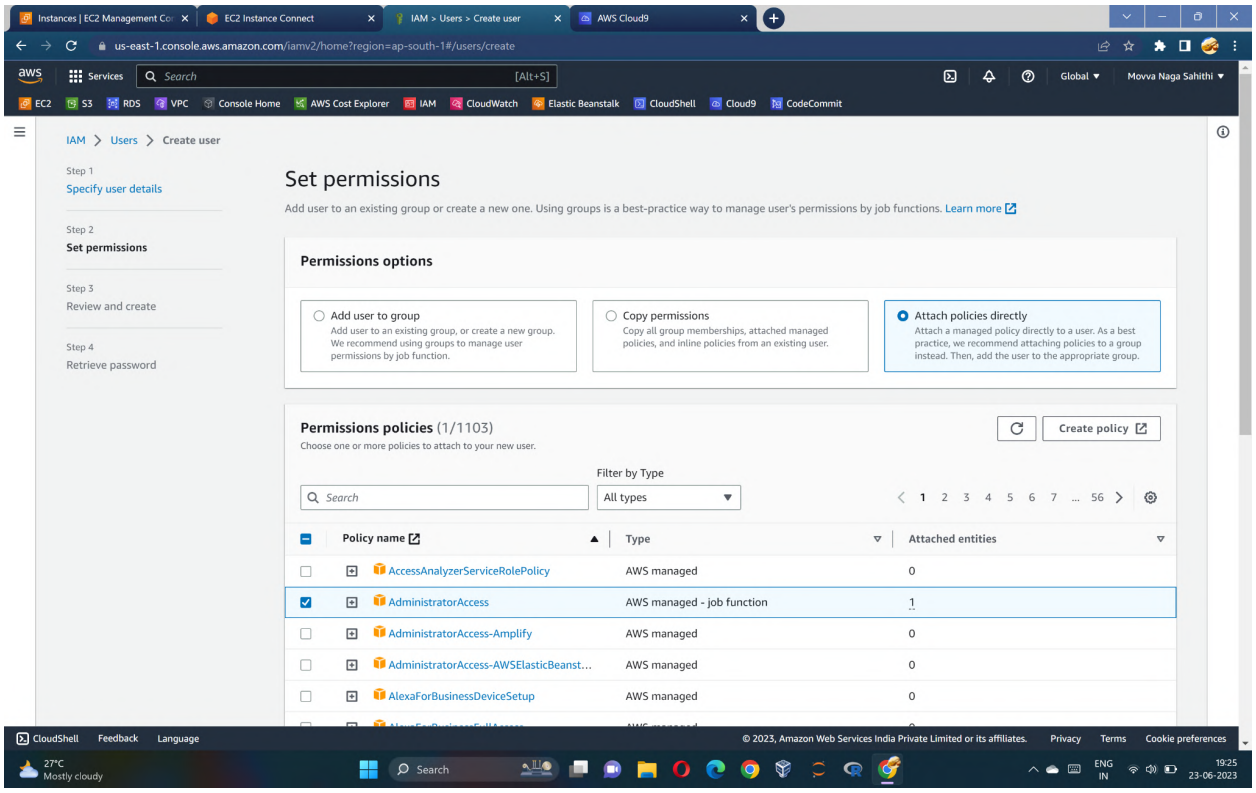


Step 13: Username : UserA

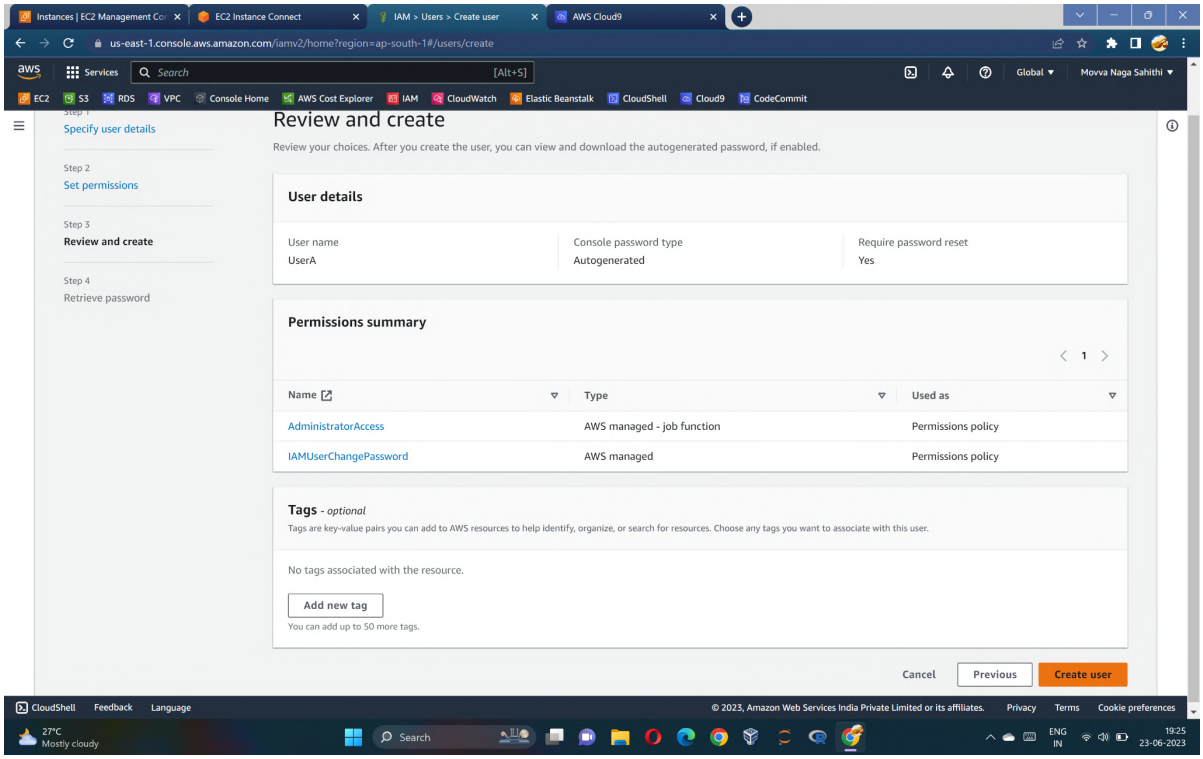
Create an IAM user with an auto-generated password.



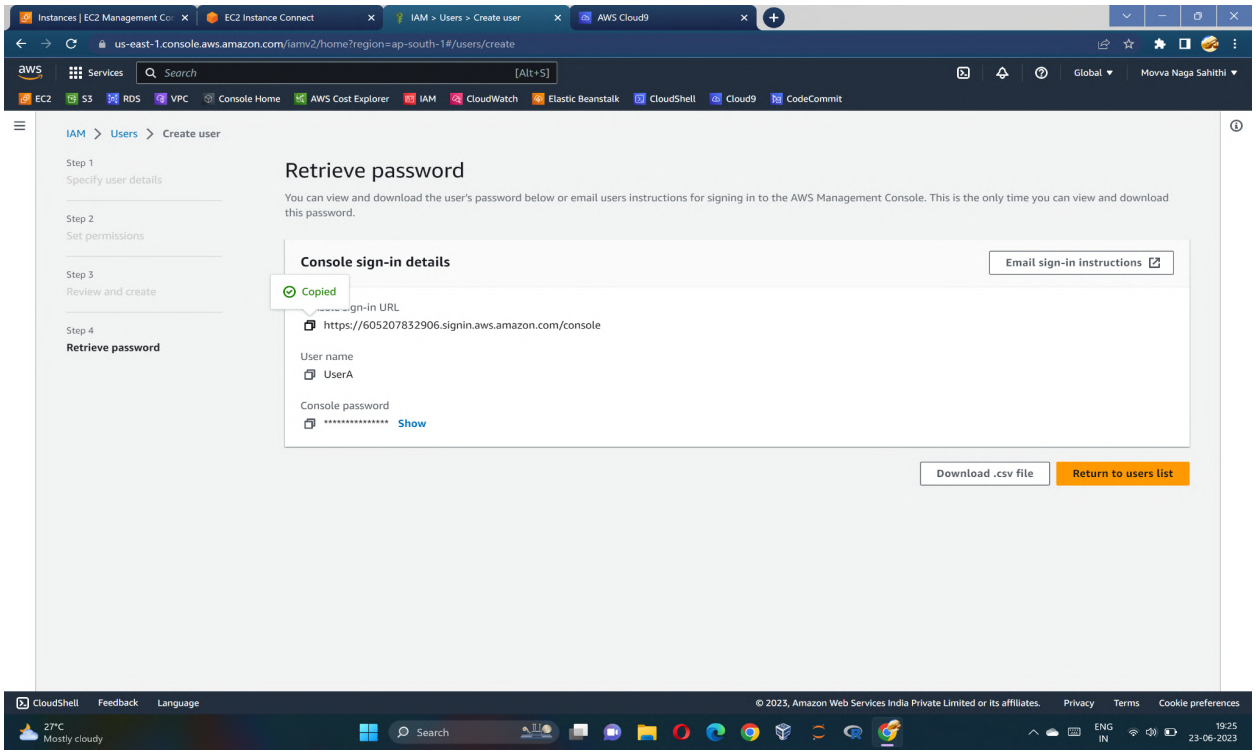
Step 14: Attach policy directly > Add Administrator Access policy.



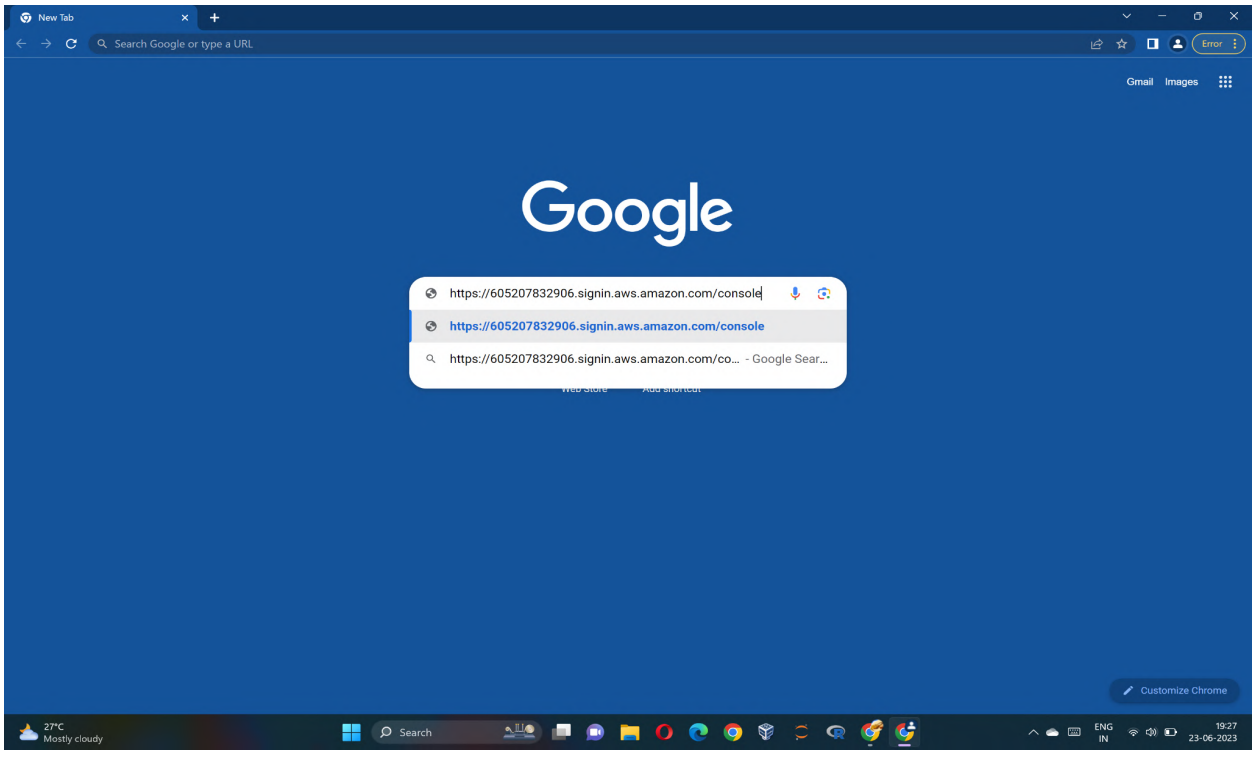
Step 15: Create User



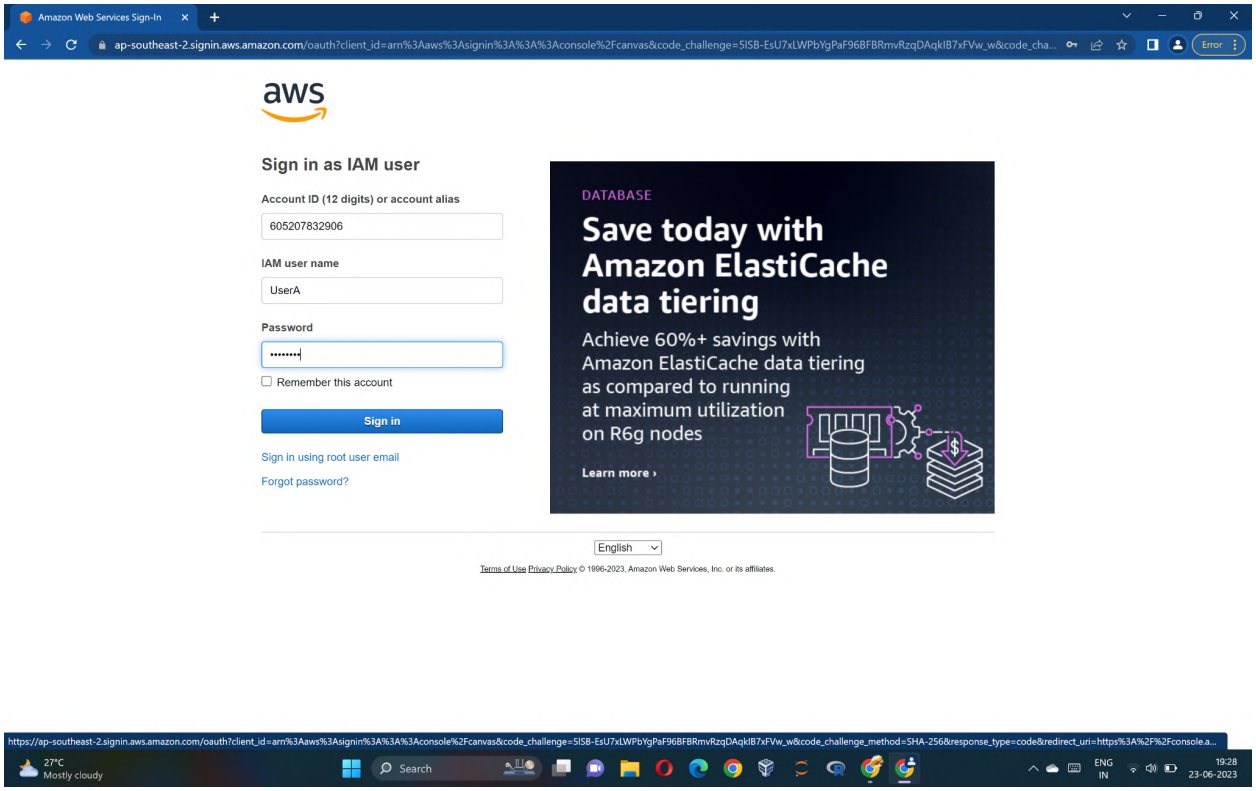
Step 16: Copy the Console Sign URL in a new search engine .(Not in the same account)



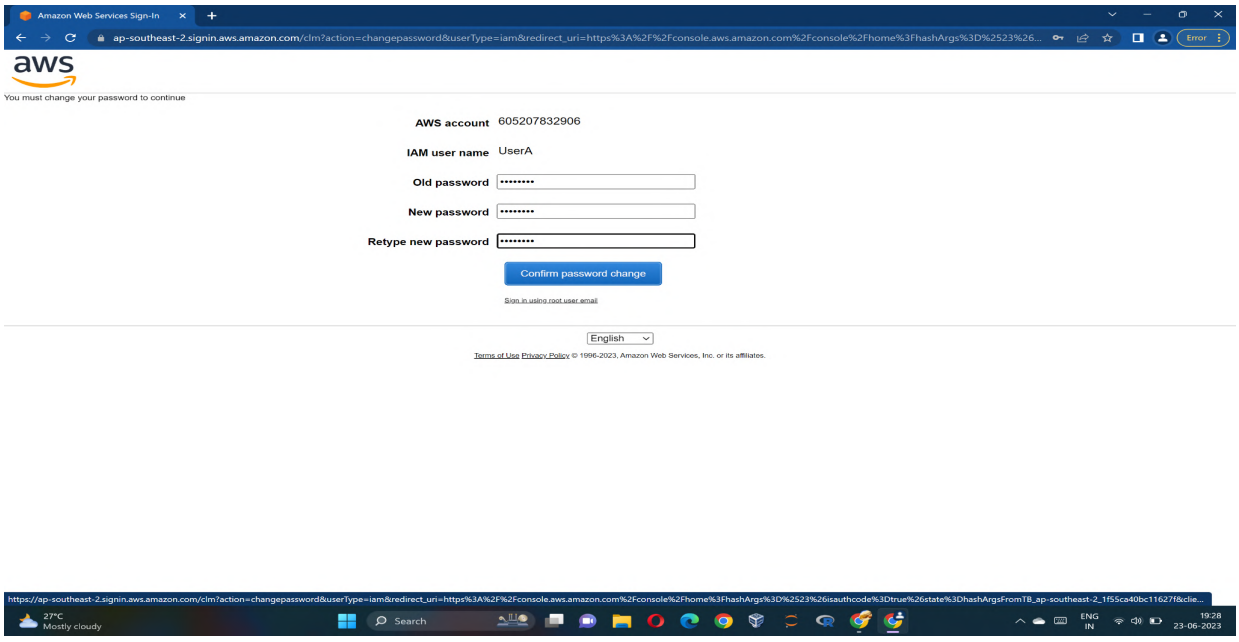
Step 17: Open the URL.



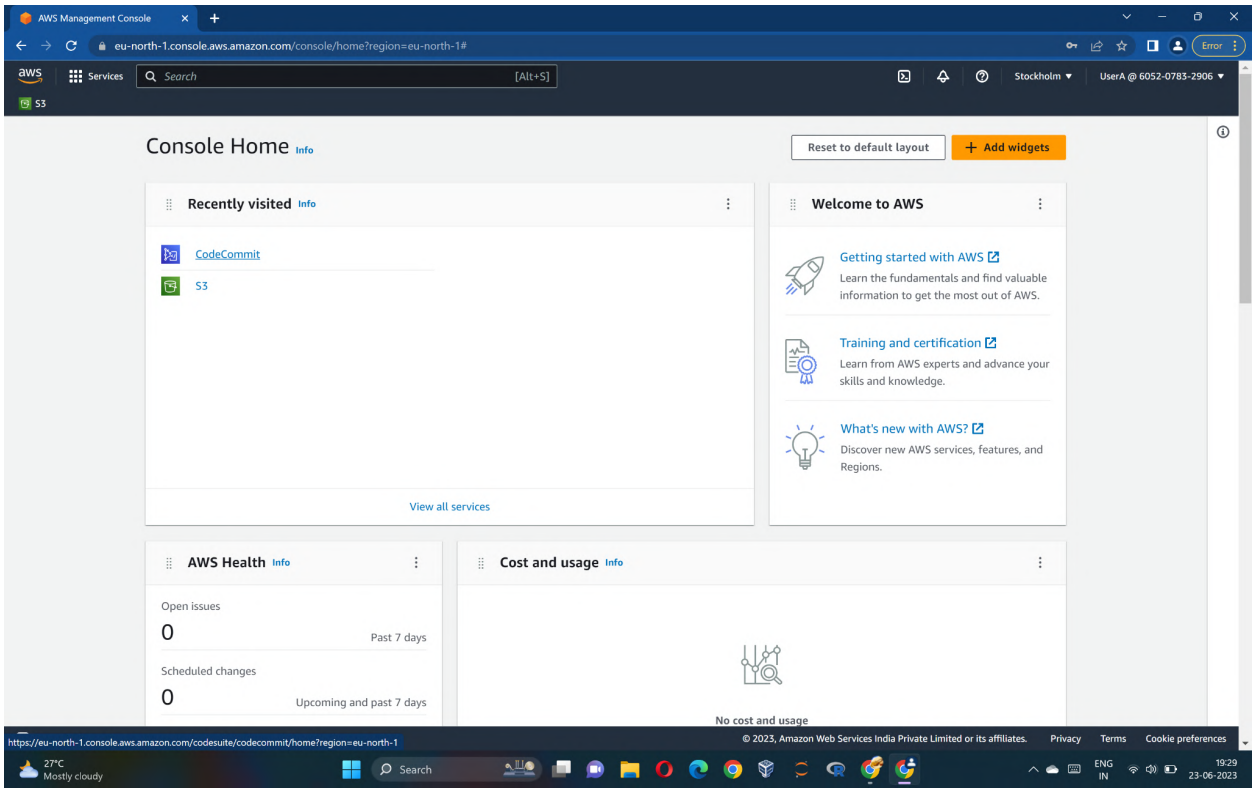
Step 18: Enter the credentials and sign-in. The password will be a one time password. If exited , can't access it again , if needed download the CSV file of credentials.



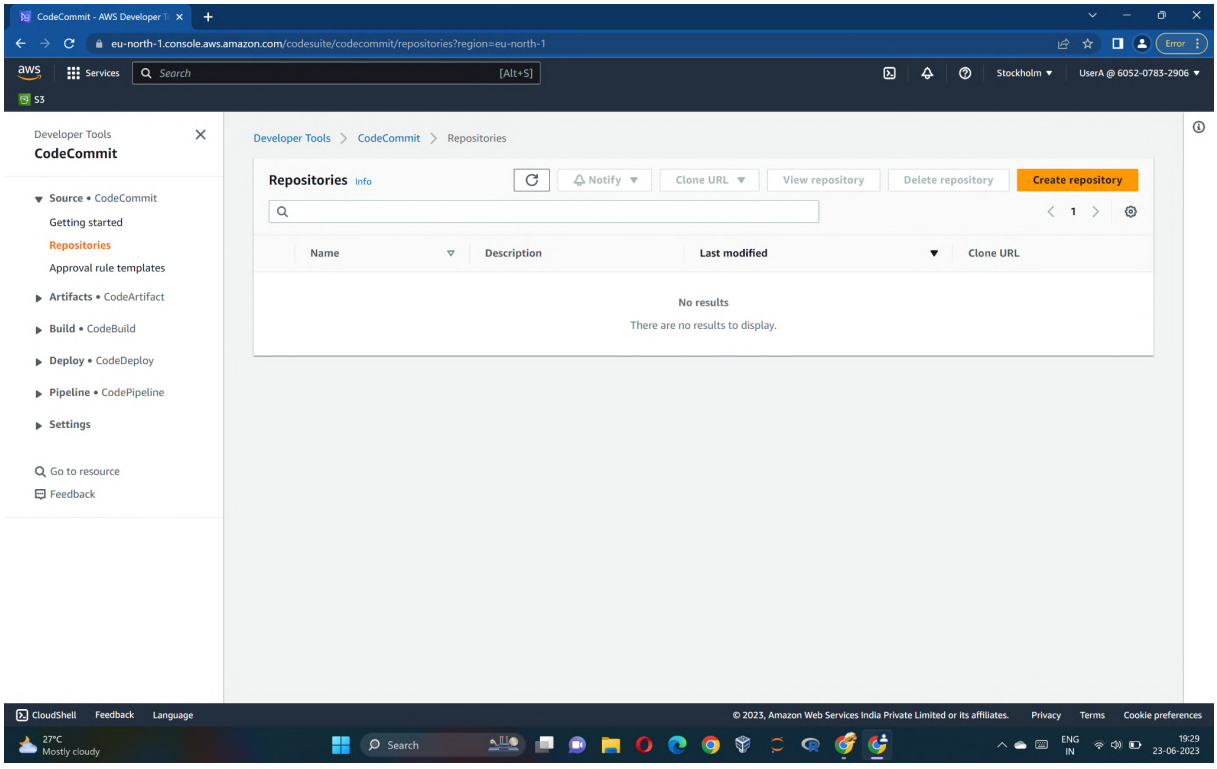
Step 19: Once signed-in, generate a new password.



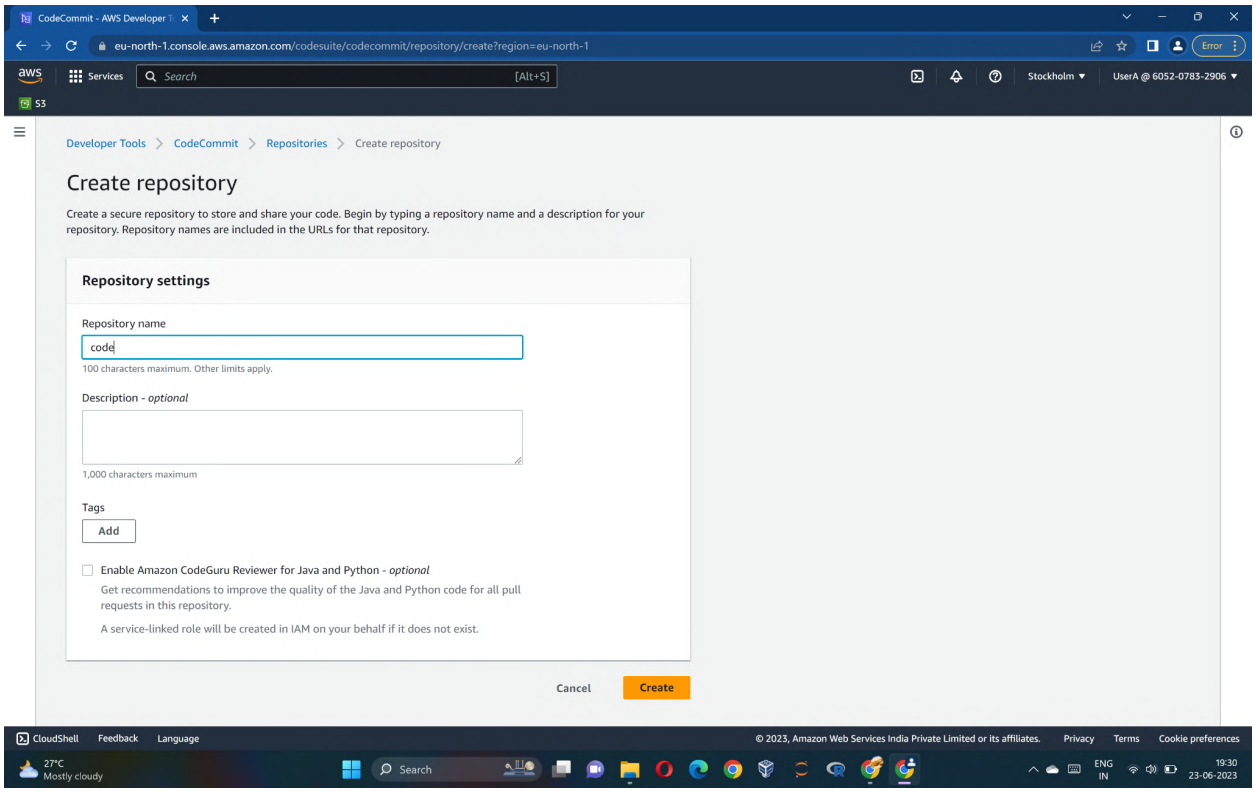
Step 20: Go-to code commit in the root user.



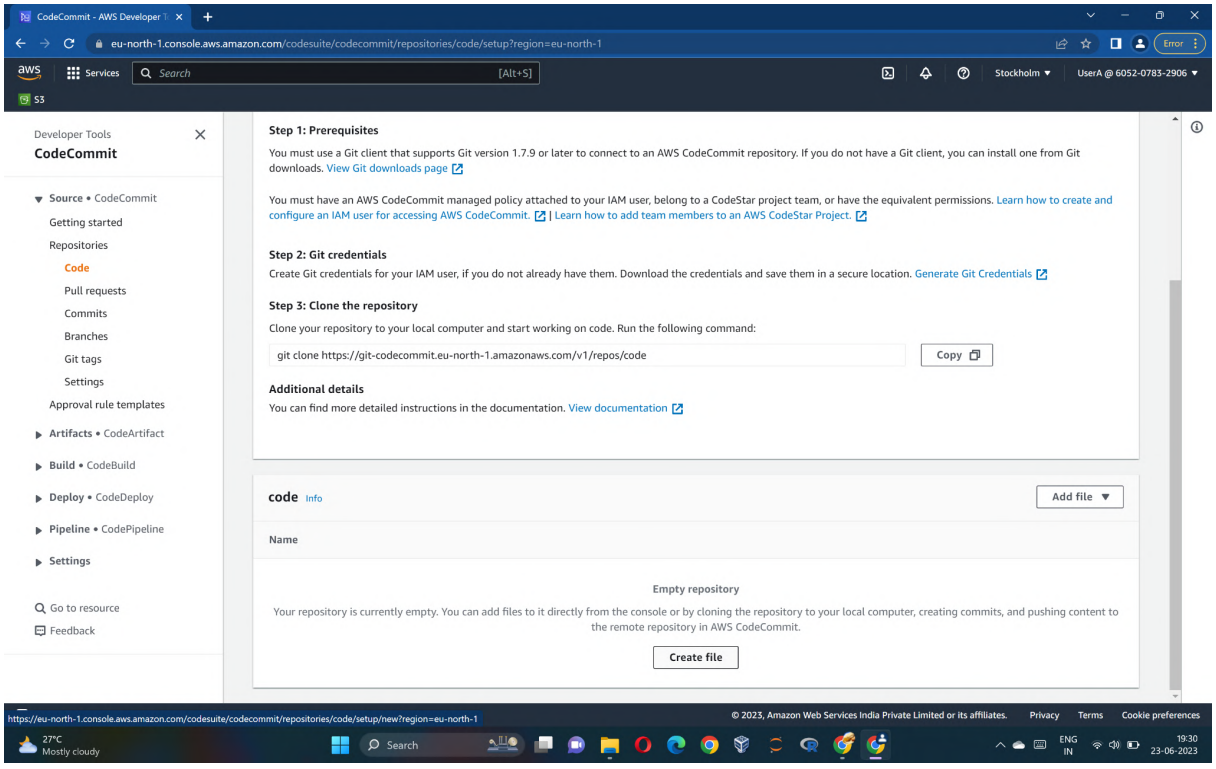
Step 21: Create a repository



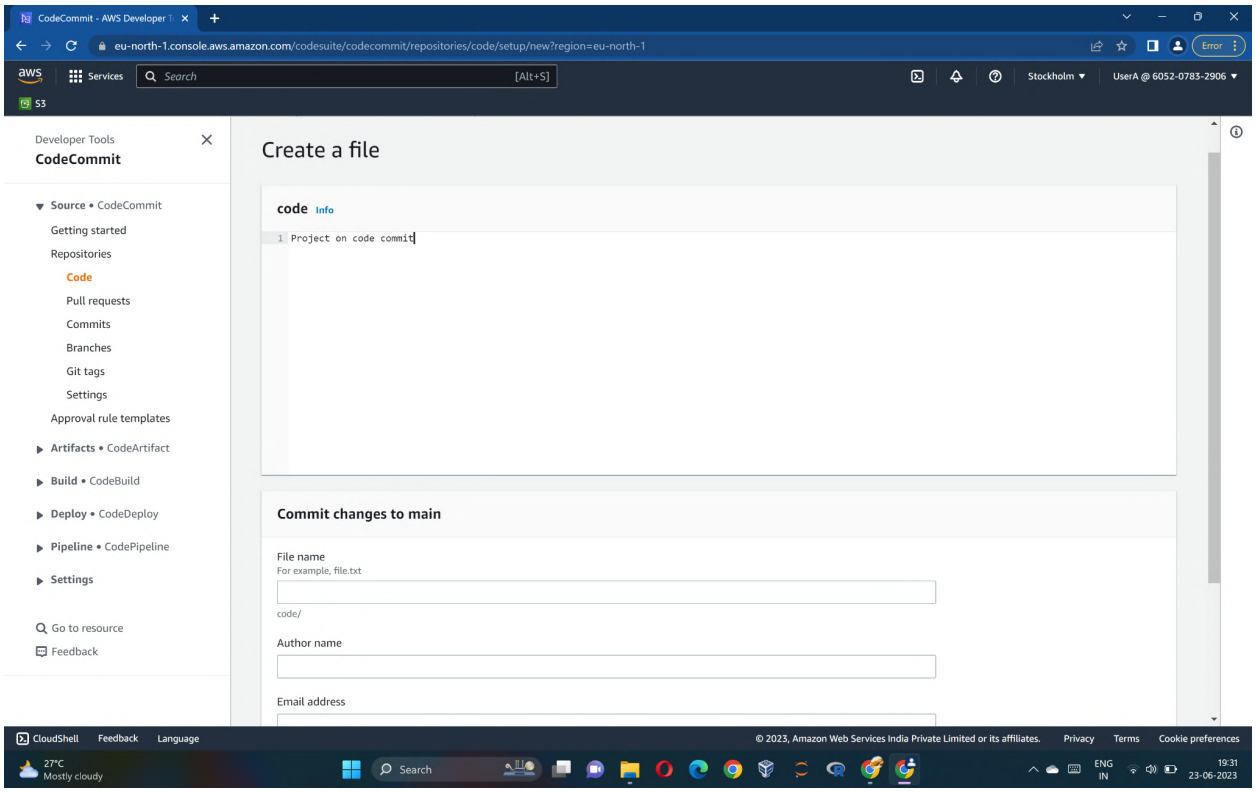
Step 22: Give the repo a name and create.



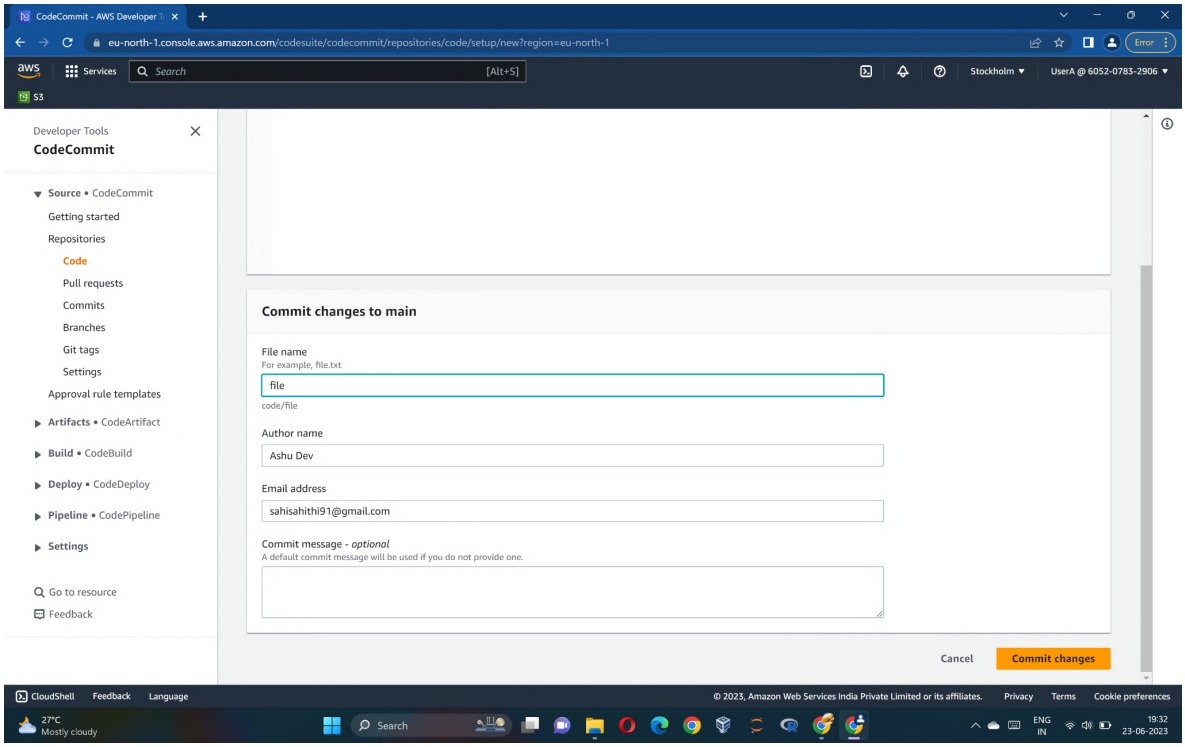
Step 23: Add a file in the repository.



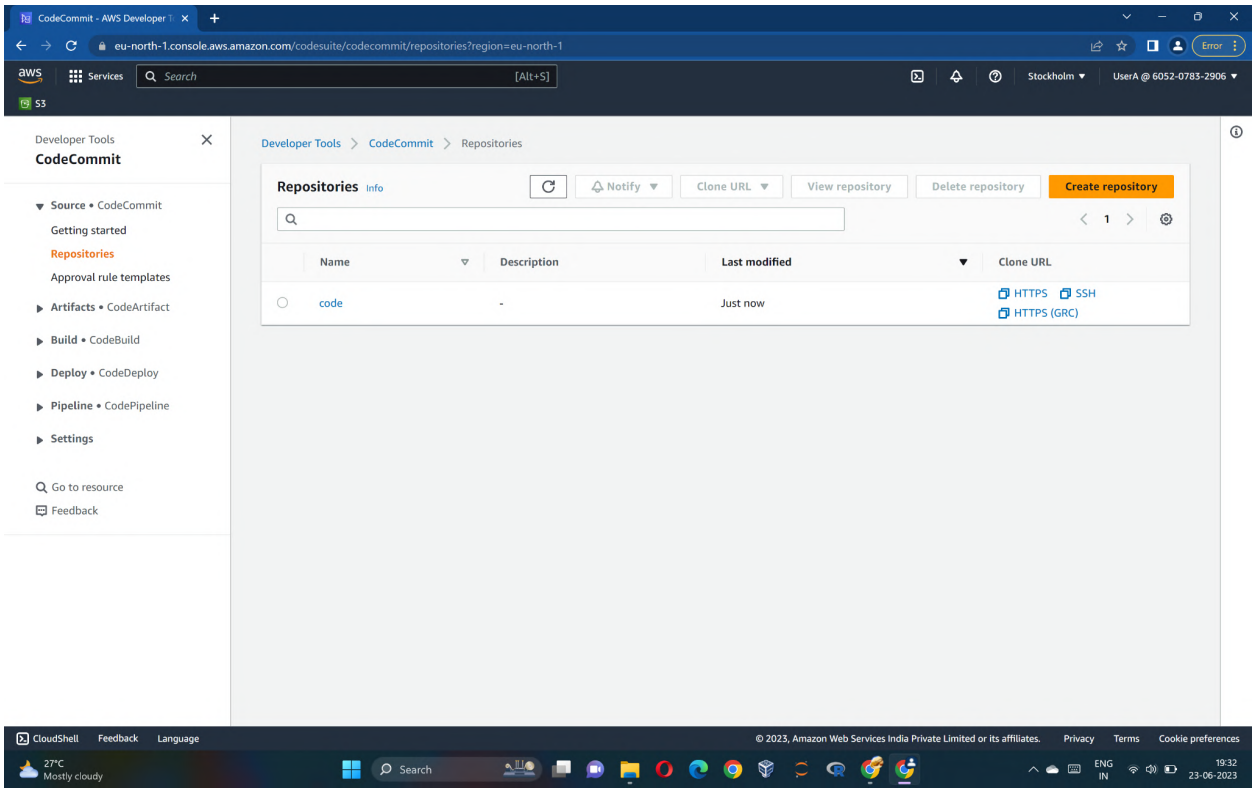
Step 24 : Enter something into the file



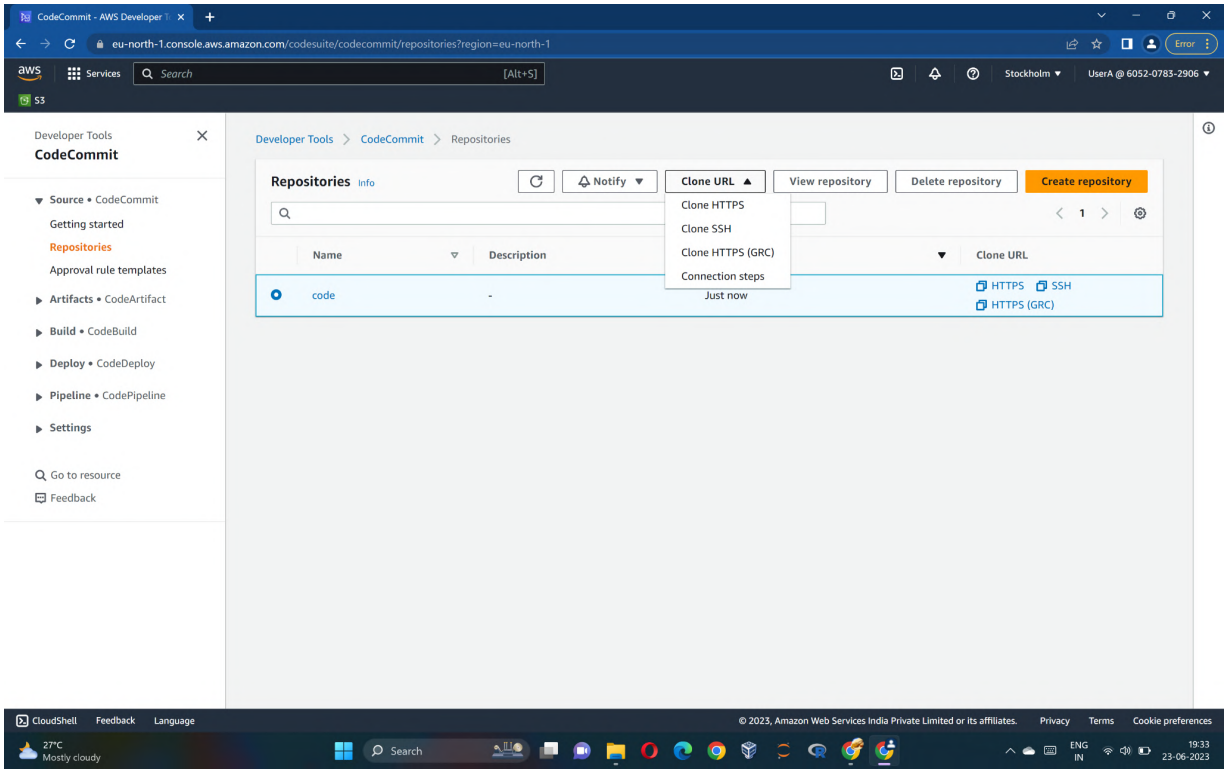
Step 25: Commit the change and save to the file. Enter a valid email address



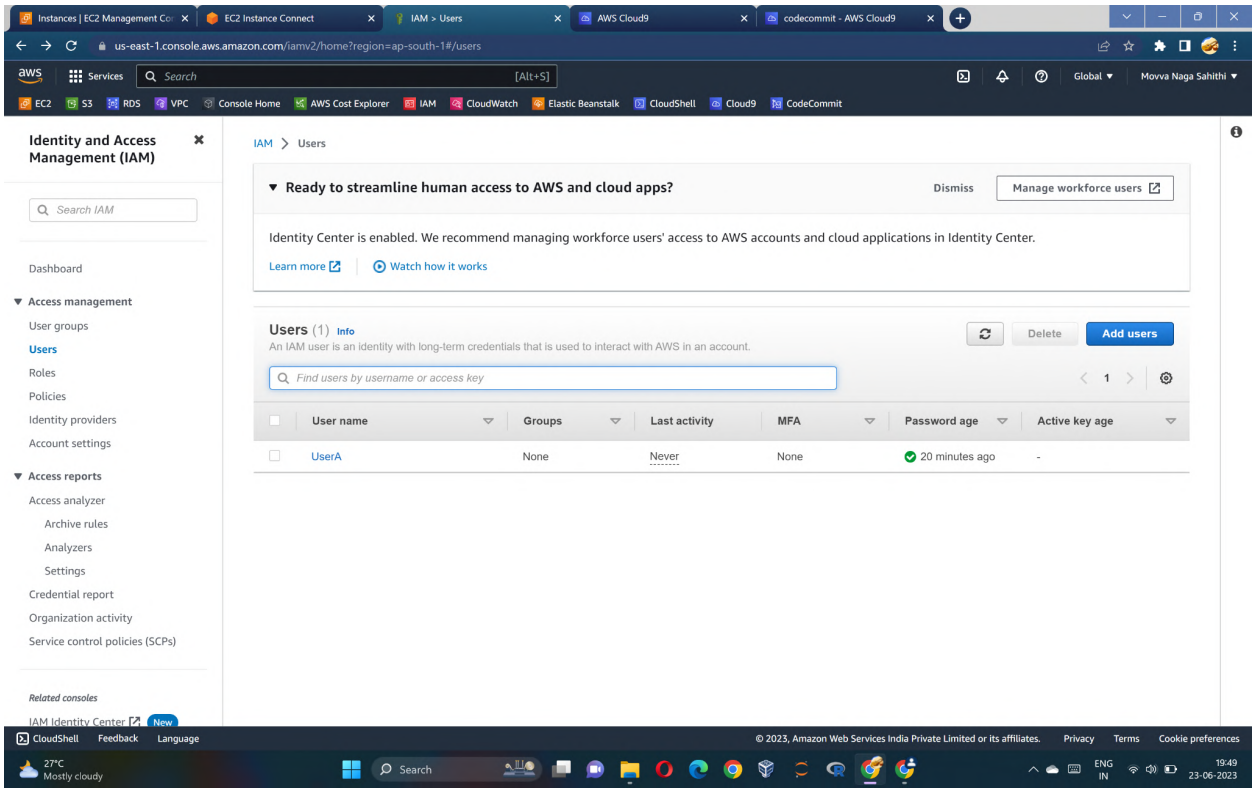
Step 26: Go to repositories > select Repo



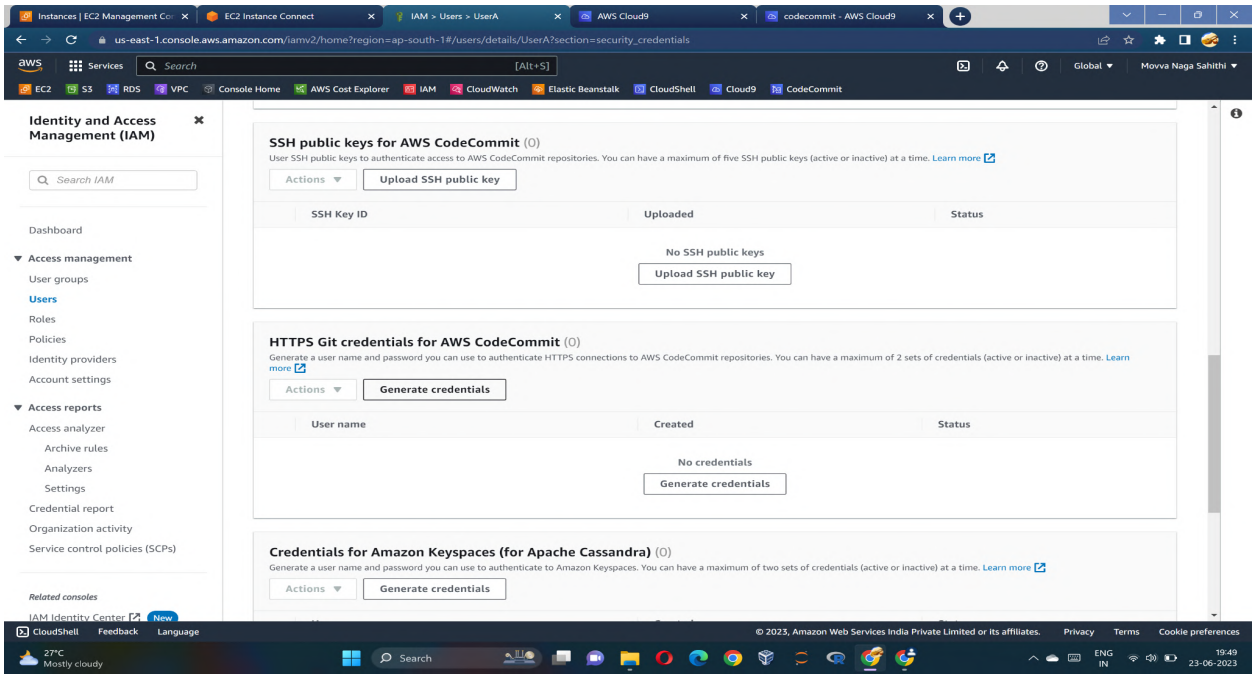
Step 27: Clone URL > Clone HTTPS



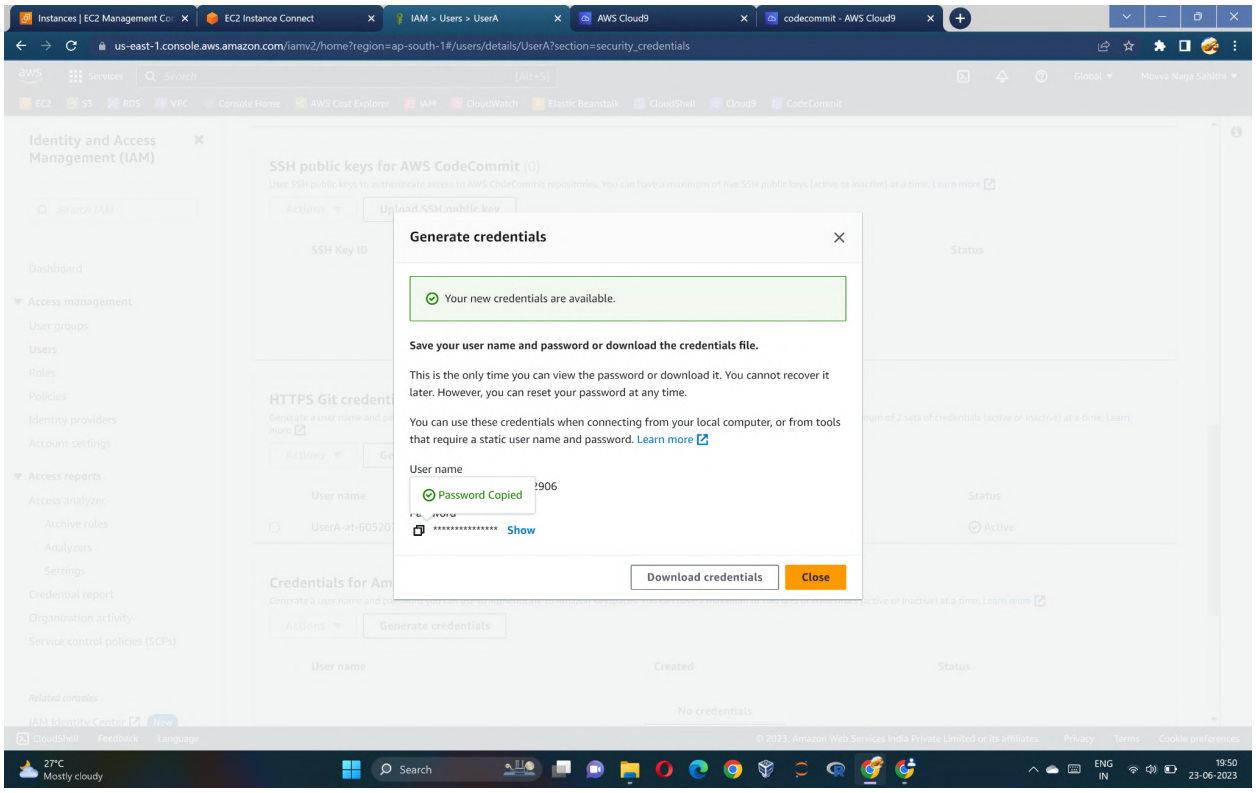
Step 28: Now we have to generate credentials to pull the file from code commit to ec2 connect and cloud9 environment.



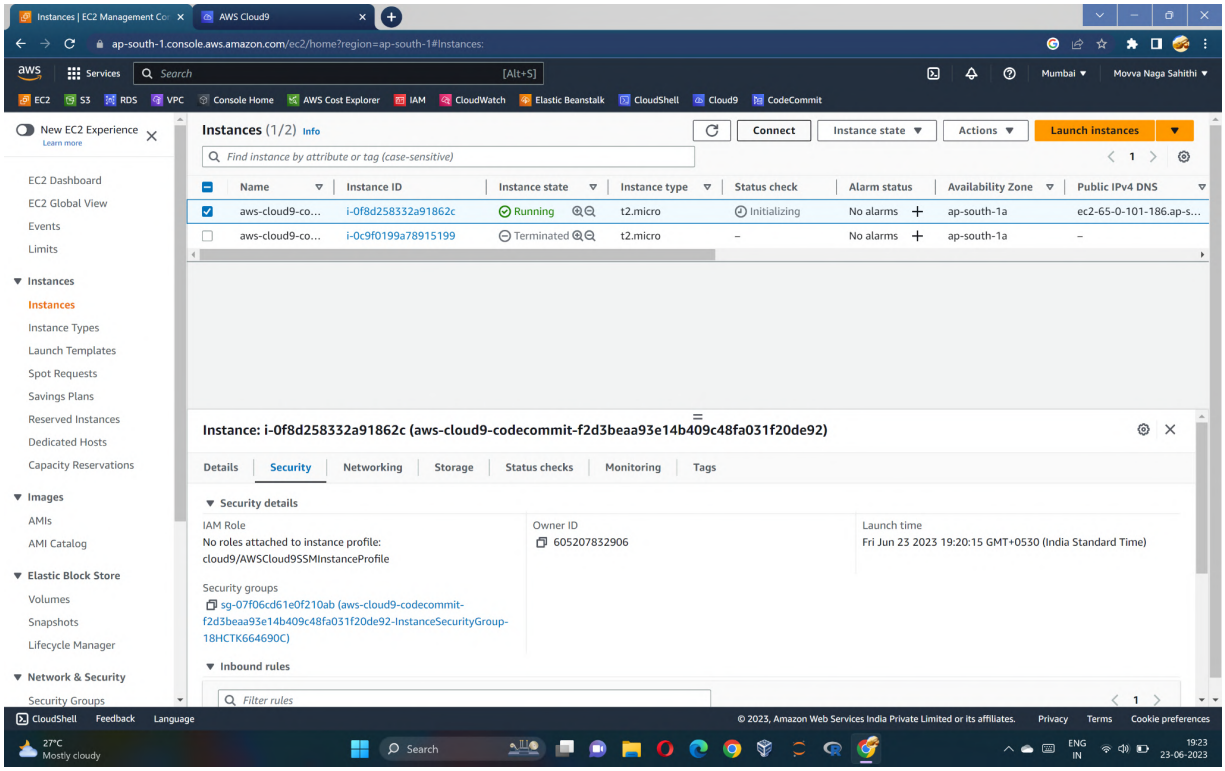
Step 29: Go to user > Security credentials > HTTPS git credentials for code commit > generate Credentials



Step 30: Copy the user name and password.



Step 31: Go to Ec2 Instance > select the instance > Connect



Step 32: Connect the Ec2 instance

```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[root@ip-172-31-44-109 ~]# sudo su
[root@ip-172-31-44-109 ~]# cd /home/ec2-user
[root@ip-172-31-44-109 ec2-user]# ls
environment  node_modules  package.json  package-lock.json
[root@ip-172-31-44-109 ec2-user]# cd environment
```

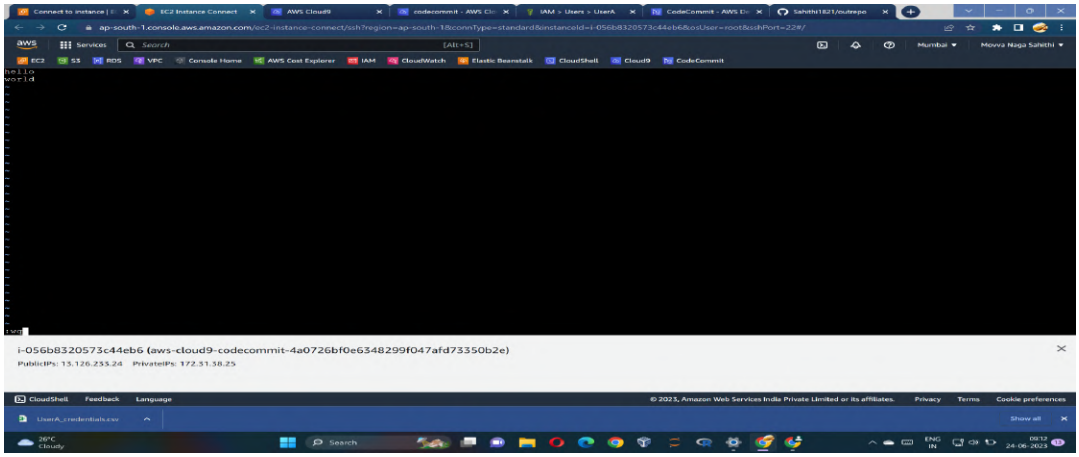
Step 33: Install git with the command “yum install git -y”. After installing , initiate the git with the “git init” command.

```
[root@ip-172-31-44-109 environment]# 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/environment/.git/
```

Step 34: Create a directory and go to the directory

```
[root@ip-172-31-44-109 ec2-user]# mkdir gitrip
[root@ip-172-31-44-109 ec2-user]# cd gitrip
```

Step 35: Create a file(file1) in the directory and write something in it.



Step 36: check the status of the file.

```
[root@ip-172-31-38-25 gitrep]# vi file1
[root@ip-172-31-38-25 gitrep]# git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file1
```

Step 37: Now add the file and check the status again.

```
[root@ip-172-31-38-25 gitrep]# git add .
[root@ip-172-31-38-25 gitrep]# git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file1
```

Step 38: Commit the git

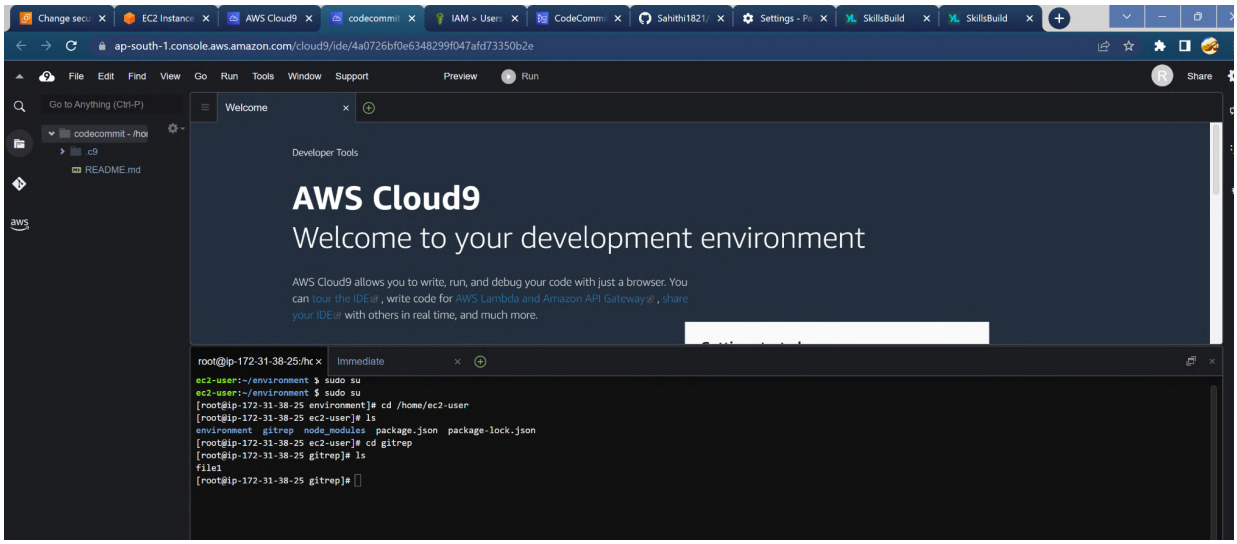
```
[root@ip-172-31-38-25 gitrep]# git commit -m "all student"
[master c034fd5] all student
 Committer: root <root@ip-172-31-38-25.ap-south-1.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 2 insertions(+)
[root@ip-172-31-38-25 gitrep]# git status
On branch master
nothing to commit, working tree clean
```

Step 39: Add a origin using the HTTPS link from the repository in userA.
Now pull the origin.
Enter the credentials copied from the IAM user > UserA > security credentials > generate credentials > credentials generated before.

```
[root@ip-172-31-38-25 gitrep]# git remote add origin https://git-codecommit.ap-south-1.amazonaws.com/v1/repos/codecommit
[root@ip-172-31-38-25 gitrep]# git pull origin main
Username for 'https://git-codecommit.ap-south-1.amazonaws.com': UserA-at-605207832906
Password for 'https://UserA-at-605207832906@git-codecommit.ap-south-1.amazonaws.com':
remote: Counting objects: 3, done.
Unpacking objects: 100% (3/3), 193 bytes | 193.00 KiB/s, done.
From https://git-codecommit.ap-south-1.amazonaws.com/v1/repos/codecommit
 * branch            main       -> FETCH_HEAD
 * [new branch]      main       -> origin/main
[root@ip-172-31-38-25 gitrep]# ls
file1
```

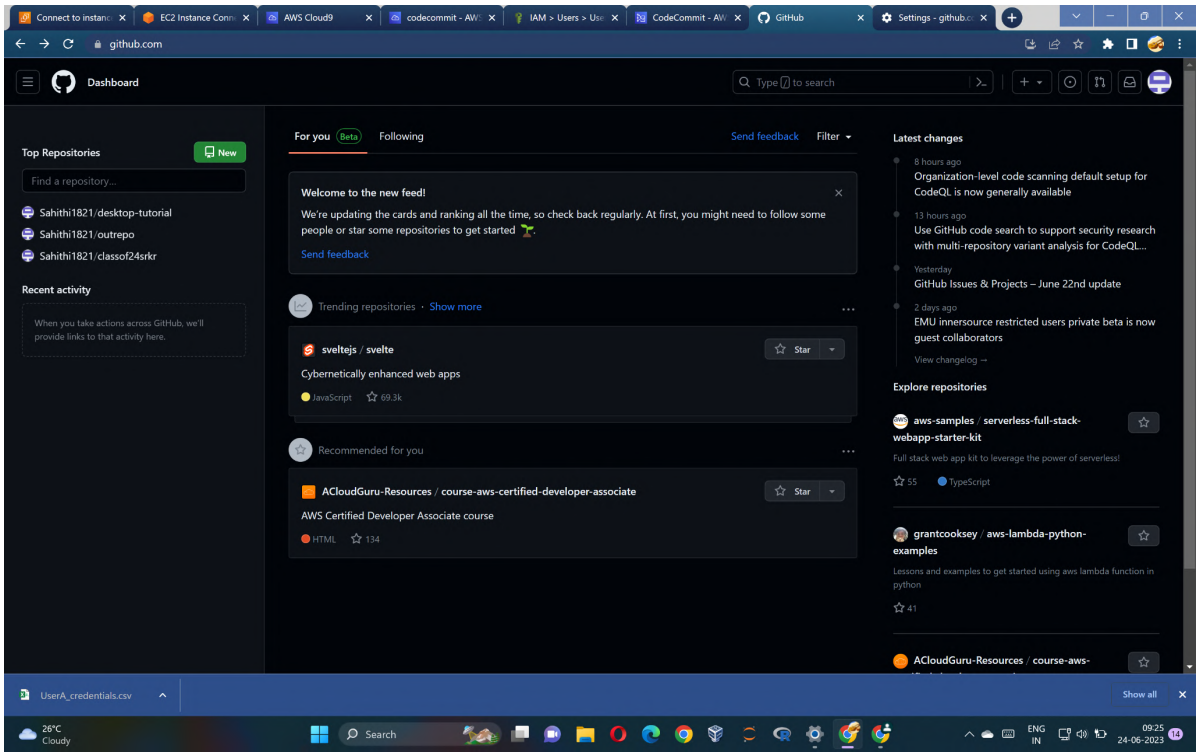
In the “ls” , you can find the file(file1) created in the codecommit repository.

Step 40: Open the console in the cloud9 and you can see the file also pulled to this console too.

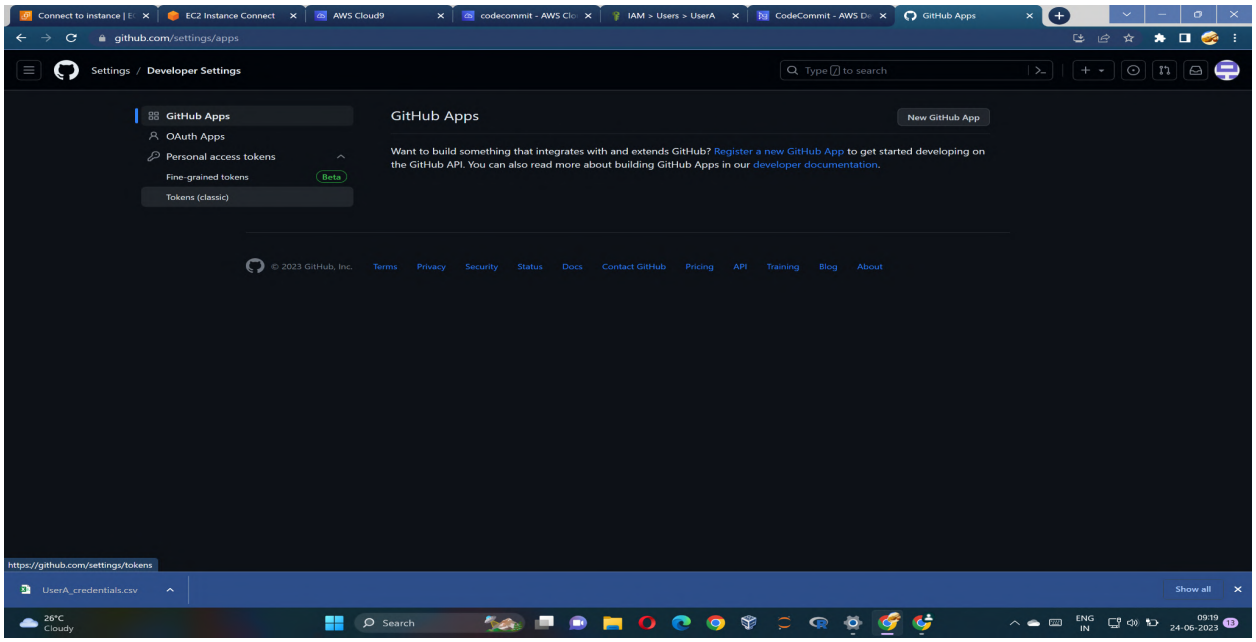


we pulled a file to our ec2 environment. Now we push the same file to a third - party repository name Git-Hub.

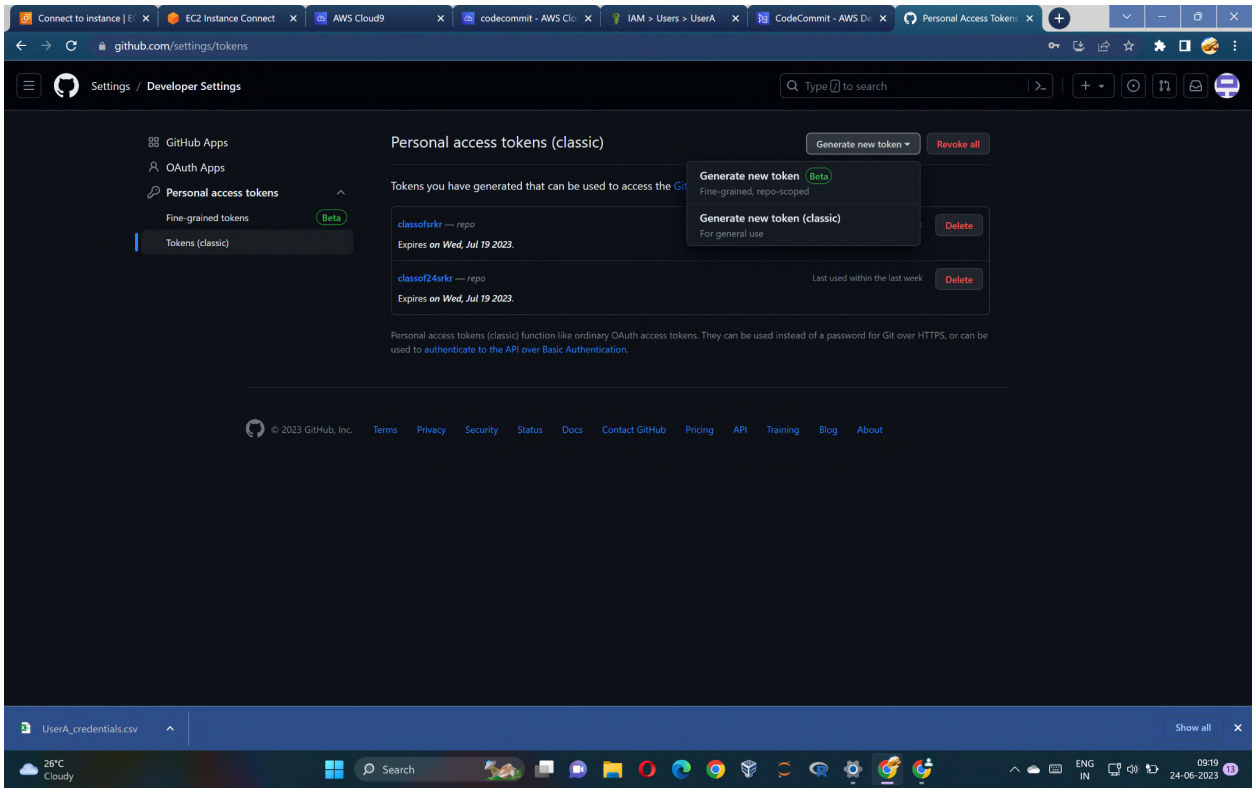
Step 40: Create a repository into Git-hub



Step 41: Generate a classic token at Settings > Developer Settings > Personal Access Tokens > Token(classic)



Step 42: Generate new token > generate new token classic



Select the repo option and create a token. Copy the password generated there.

Step 43: In EC2 Instance connect , create a new directory and copy the same file(file1) in this directory. Initiate the git in this directory

```
[root@ip-172-31-38-25 ec2-user]# mkdir gitrepo
[root@ip-172-31-38-25 ec2-user]# ls
environment  gitrep  gitrepo  node_modules  package.json  package-lock.json
[root@ip-172-31-38-25 ec2-user]# cd gitrepo
[root@ip-172-31-38-25 gitrepo]# 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/gitrepo/.git/
[root@ip-172-31-38-25 gitrepo]# cd ..
[root@ip-172-31-38-25 ec2-user]# cd gitrepo
[root@ip-172-31-38-25 gitrepo]# cd ..
[root@ip-172-31-38-25 ec2-user]# cd gitrip
bash: cd: gitrip: No such file or directory
[root@ip-172-31-38-25 ec2-user]# cd gitrep
[root@ip-172-31-38-25 gitrep]# cp file1 /home/ec2-user/gitrepo
[root@ip-172-31-38-25 gitrep]# cd /home/ec2-user/gitrepo
[root@ip-172-31-38-25 gitrepo]# ls
file1
```

Step 44: Again add the file and commit the file

```
[root@ip-172-31-38-25 gitrepo]# git add .
[root@ip-172-31-38-25 gitrepo]# git commit -m "add something"
[master (root-commit) 49e8ed8] add something
 Committer: root <root@ip-172-31-38-25.ap-south-1.compute.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 2 insertions(+)
 create mode 100644 file1
```

Step 45: Copy the repository link in the github. Add the remote origin here

```
[root@ip-172-31-38-25 gitrepo]# git remote add origin https://github.com/Sahithi1821/outrepo.git
[root@ip-172-31-38-25 gitrepo]# git push origin master
Username for 'https://github.com':
```

Step 46: Give the username of your github and enter the password copied from the token generation.

```
[root@ip-172-31-38-25 gitrepo]# git push origin master
Username for 'https://github.com': Sahithi1821
Password for 'https://Sahithi1821@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 236 bytes | 236.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Sahithi1821/outrepo.git
 * [new branch]      master -> master
[root@ip-172-31-38-25 gitrepo]#
```

Step 47: Goto Github > repository
You can see the file(file1)