AWS HACKATHON DOCUMENTATION

**TEAM MEMBERS:**

1.DARLA JYOTHSNA

2.SHAIK SAHEDA

3.GAMINI PRAVEENA

4.GUBBALA SATISH

5.PUTTA HEMANTH SAI

6.POTNURI CHAITANYA

September 29, 2023

BRAINOVISION

# AWS PROJECT REPORT – DOCUMENTATION

**What Is AWS?**

AWS (Amazon Web Services) is a comprehensive, evolving cloud computing platform provided by Amazon that includes a mixture of infrastructure-as-a-service (IaaS), platform-as-a-service (PaaS) and packaged-software-as-a-service (SaaS) offerings. AWS services can offer an organization tools such as compute power, database storage and content delivery services.

Amazon.com Web Services launched its first web services in 2002 from the internal infrastructure that Amazon.com built to handle its online retail operations. In 2006, it began offering its defining IaaS services

AWS offers many different tools and solutions for enterprises and software developers that can be used in data centres in up to 190 countries. Groups such as government agencies, education institutions, non-profits and private organizations can use AWS services.

**What is DEVOPS?**

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market.

**Benefits of DevOps**

* Speed and Recovery
* Rapid Delivery
* Reliability
* Scale
* Security
* Improved collaboration

**AWS TOOLS**

AWS provides a wide range of tools and services However, there are a few tools that are widely used and considered fundamental in many AWS deployments. Here are some of the most commonly used tools of AWS:

* Amazon EC2 (Elastic Compute Cloud)

It provides a virtual server in the cloud, allowing you to run applications and services on a variety of operating systems.it also used to create various instances.

* Amazon RDS (Relational Database Service)

It Provides managed database services for various relational database engines, including Amazon Aurora, MySQL, PostgreSQL, Oracle, and SQL Server.

* Amazon VPC (Virtual Private Cloud)

It Offers a logically isolated virtual network within the AWS cloud. It allows you to launch AWS resources in a defined virtual network, providing control over IP addressing, subnets, and network gateways.

* *Amazon SQS (Simple Queue Service)*

It A fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications.

* AWS CloudTrail

It Captures and logs API activity and events within your AWS infrastructure, providing audit trails for compliance, security analysis, and troubleshooting.

**ADVANTAGES:**

Amazon Web Services (AWS) offers several advantages that have contributed to its popularity and widespread adoption among businesses and developers. Some of the key advantages of AWS are:

* Scalability : scalability enables businesses to handle sudden traffic spikes, accommodate growth, and optimize costs by paying only for the resources they use.
* Cost-effectiveness: AWS operates on a pay-as-you-go pricing model. It eliminates the need for upfront capital investment in hardware and infrastructure, making it cost-effective for businesses of all sizes.
* Global Infrastructure : AWS has a vast global infrastructure with multiple data centers located across different regions worldwide.
* Security : AWS has implemented a robust and comprehensive security framework to protect customer data and infrastructure. It offers various security features, such as identity and access management (IAM), encryption, network security, and data protection.
* Flexibility : WS provides a wide range of services and tools that cater to different application requirements and use cases. It supports various operating systems, programming languages, databases, and frameworks, allowing businesses to choose the technologies that best suit their needs.

**REAL LIFE APPLICATIONS**

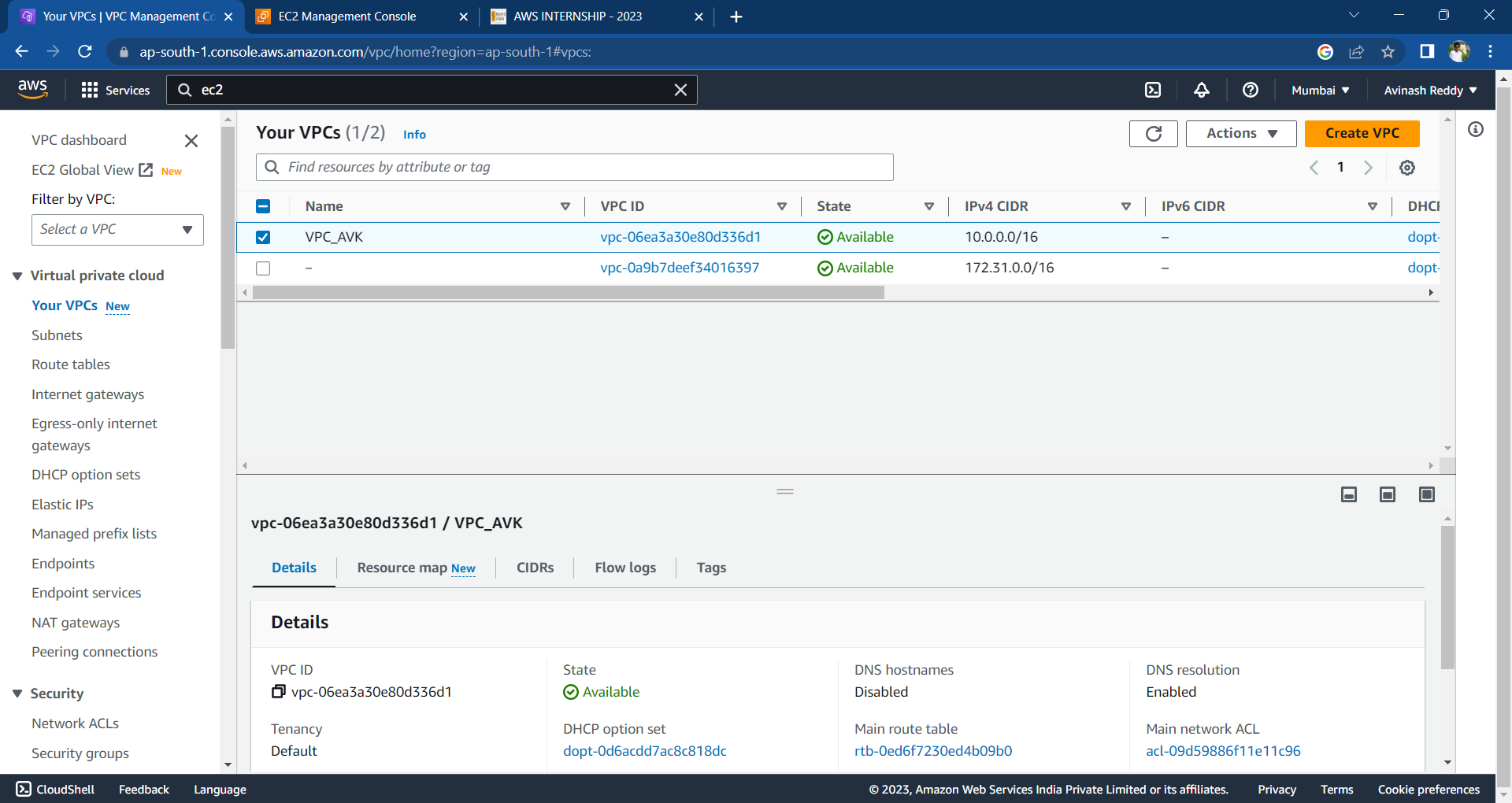
Amazon Web Services (AWS) offers a broad range of services and tools that can be utilized in various applications across different industries. Some of them were :

* NETFLIX : Netflix utilizes AWS's scalability and global infrastructure to stream its content to millions of users worldwide.Its content delivery services enable Netflix to deliver a seamless streaming experience.
* NASA : NASA utilizes AWS for various purposes, including data storage, processing, and analysis.It allows scientists and researchers to access and analyze data efficiently, enabling advancements in space exploration and research.
* PHILIPS : Philips is a global healthcare technology company that utilizes AWS for cloud based computing technology. It is used to store patient data and provide remote monitoring capabilities, and develop AI-driven healthcare applications.
* CAPITAL ONE : capital one is a leading financial institution.it uses AWS for its digital banking services and customer-facing applications. AWS provides the scalability, security, and compliance required to handle financial transactions, store sensitive customer data, and deliver real-time financial services to its customers.

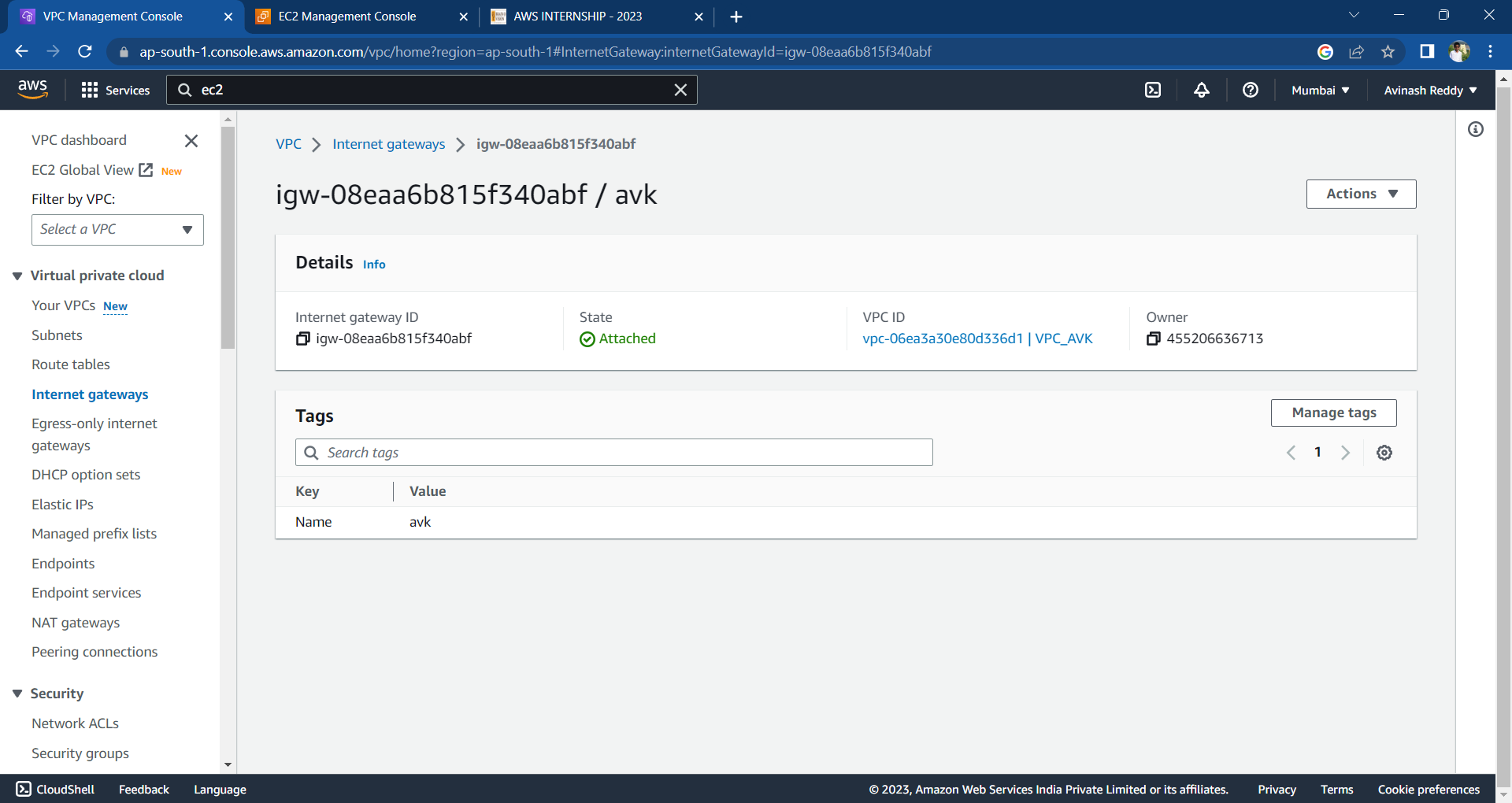
**STAGE-1:**

**Creating a VPC (Virtual Private Cloud):**

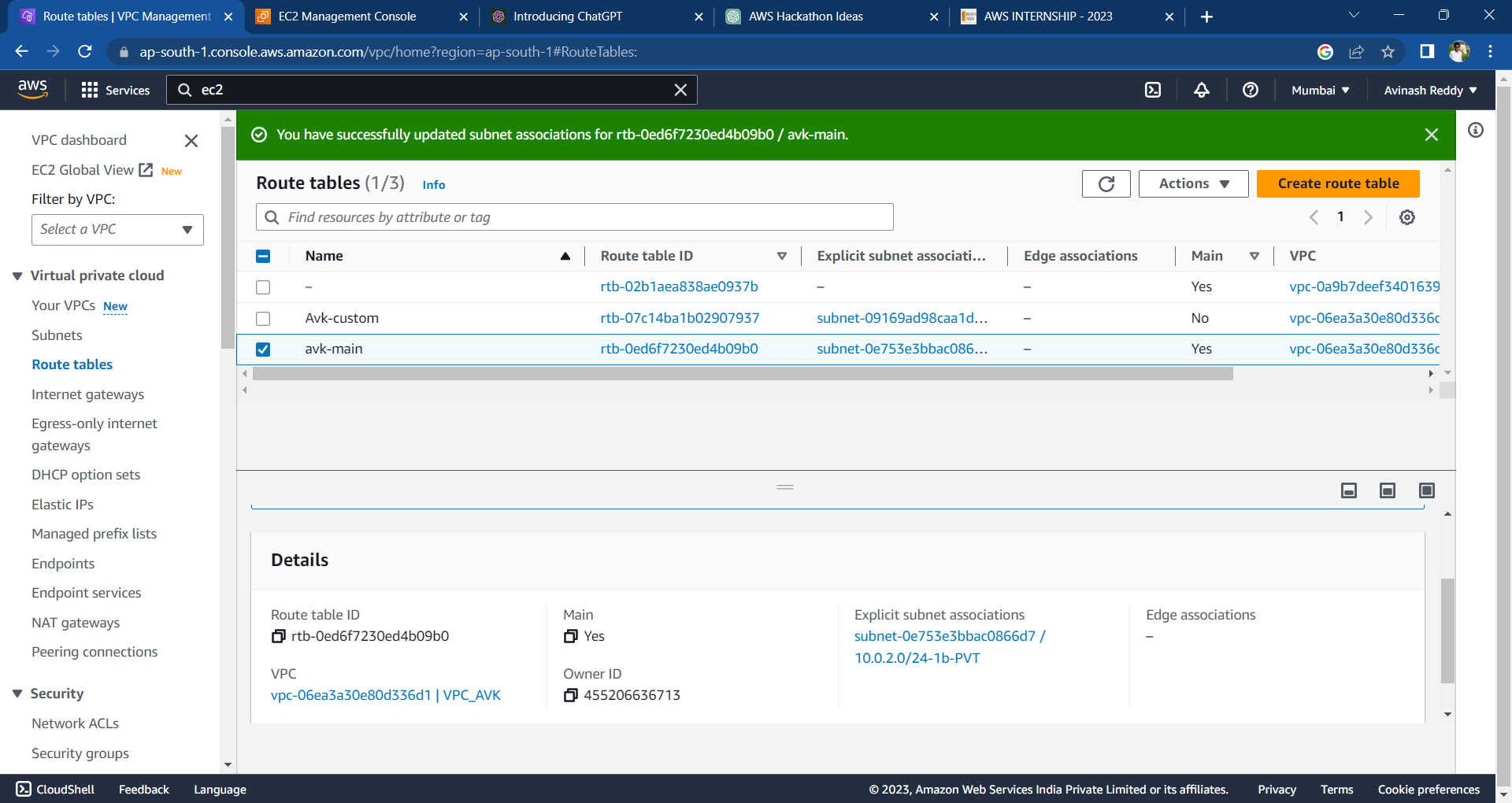
* In the VPC dashboard click on the “create VPC” Button to start the VPC creation wizard.
* Configure the VPC settings:
  + - Provide a name for your VPC.
    - Specify the IPv4 CIDR block for your VPC's IP address range ( 10.0.0.0/16).
    - Optionally, you can assign an IPv6 CIDR block to your VPC.
* Configure the VPC's subnets:
* Specify the IPv4 CIDR block for your first subnet (e.g., 10.0.0.0/24).
* Choose the availability zone where you want to create the subnet.
* Repeat this step to create additional subnets if needed.
* Configure the VPC's route table:
* Create a new route table or select an existing one.
* Associate the subnets created in the previous step with the route table
* Configure the VPC's internet gateway:
* Create a new internet gateway or select an existing one.
* Attach the internet gateway to your VPC.
* Configure the VPC's security groups:
* Create new security groups or select existing ones.
* Define the inbound and outbound rules for each security group to control network traffic.
* Review all the configuration details and settings for your VPC. If everything looks correct, click on the "Create VPC" button to create your VPC.

****

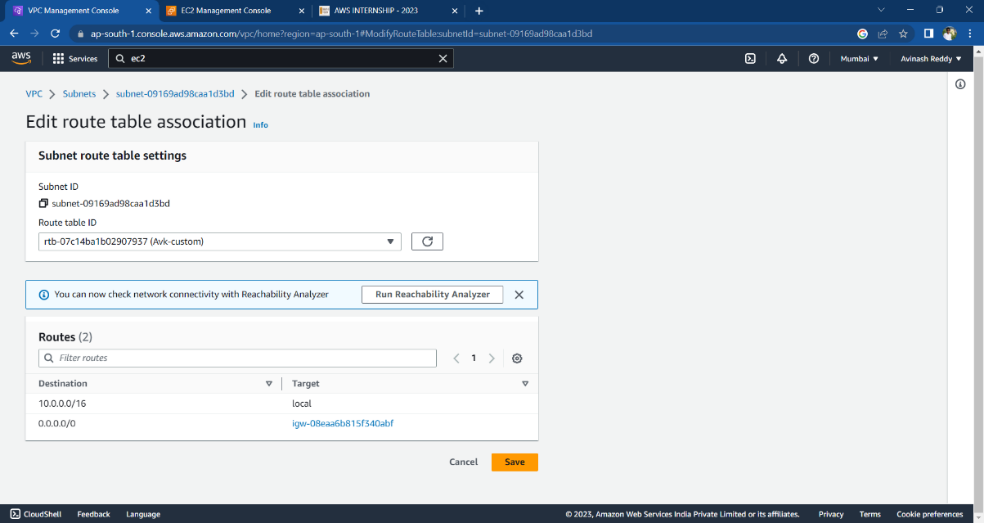
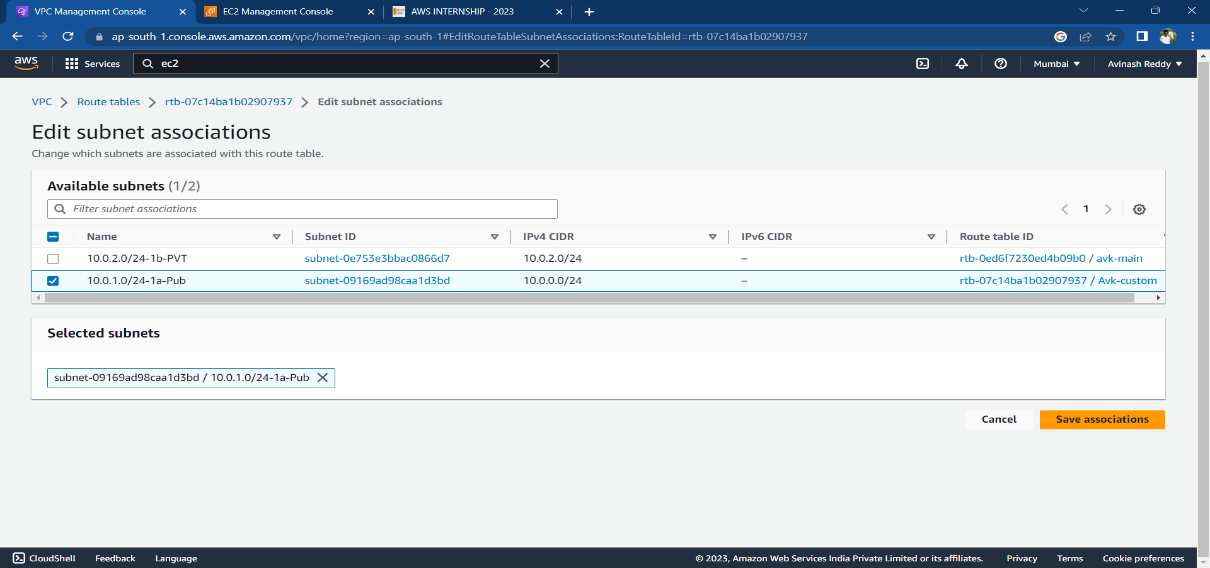
**INTERNET GATEWAYS (IGW)**

****

**ROUTE TABLES (RT):**

* Go to the "Route Tables" section: Within the selected VPC, click on the "Route Tables" option in the left navigation menu. This will display the list of existing route tables in the selected VPC**.**
* Create a new route table: Click on the "Create Route Table" button to create a new route table within the selected VPC.
* Configure the route table settings:
* Provide a name for the route table to identify it.
* Select the VPC in which you want to create the route table.
* Choose the desired subnet associations for the route table. Subnets can be associated with multiple route tables, and each subnet must be associated with at least one route table.
* Configure the routes:
* Click on the "Edit routes" button to add or edit routes in the route table.
* Add the desired routes by specifying the destination IP range and the target (e.g., an internet gateway, a virtual private gateway, or a NAT gateway)
* Save the route table: Click on the "Save" button to save the configured route table.
* Associate subnets with the route table:
* In the "Associations" tab of the route table, click on the "Edit subnet associations" button.
* Select the subnets you want to associate with the route table and click on the "Save" button.
* Review the route table: Verify the route table settings, associations, and routes in the AWS Management Console.
* Create a new route table: Click on the "Create Route Table" button to create a new route table within the selected VPC.

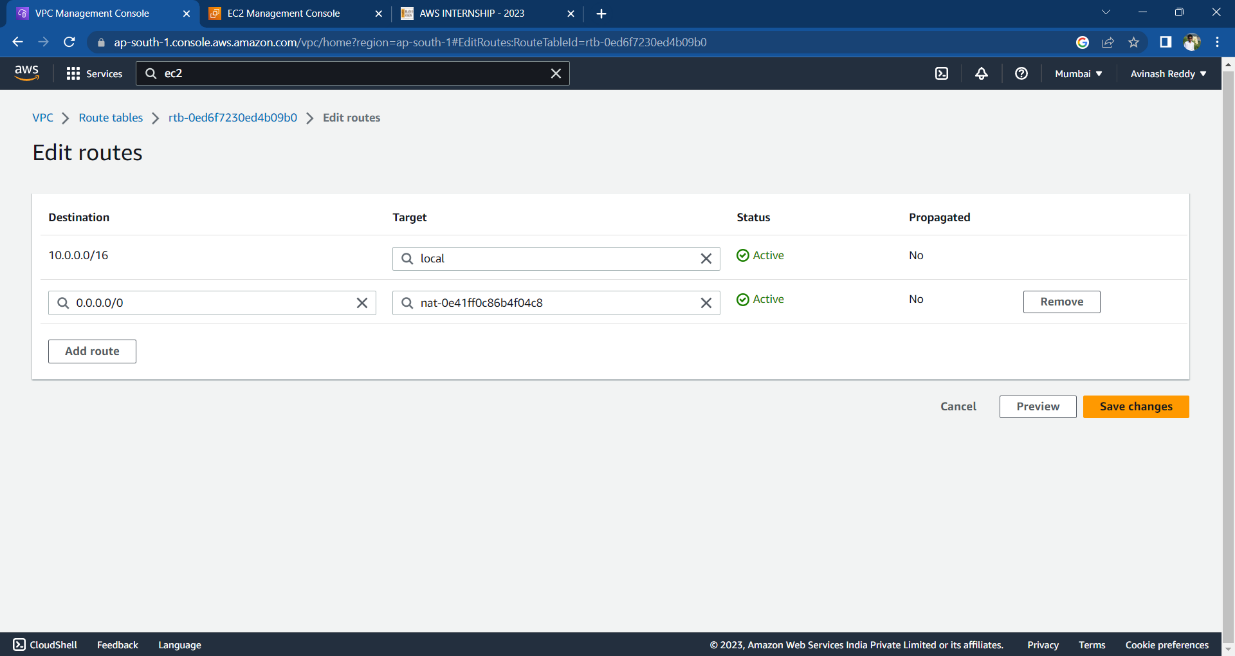
**PUBLIC SUBNET**

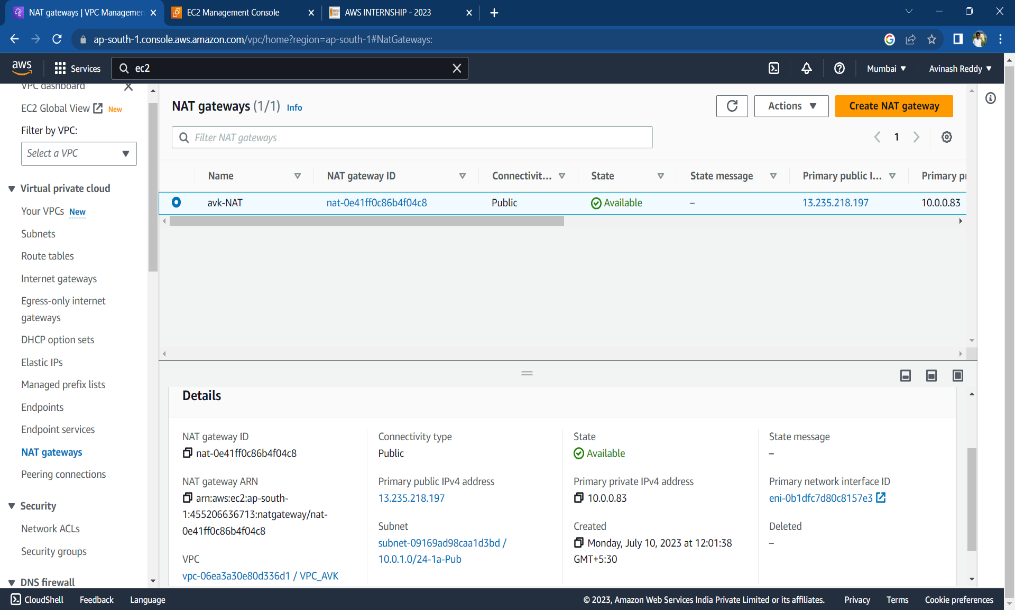
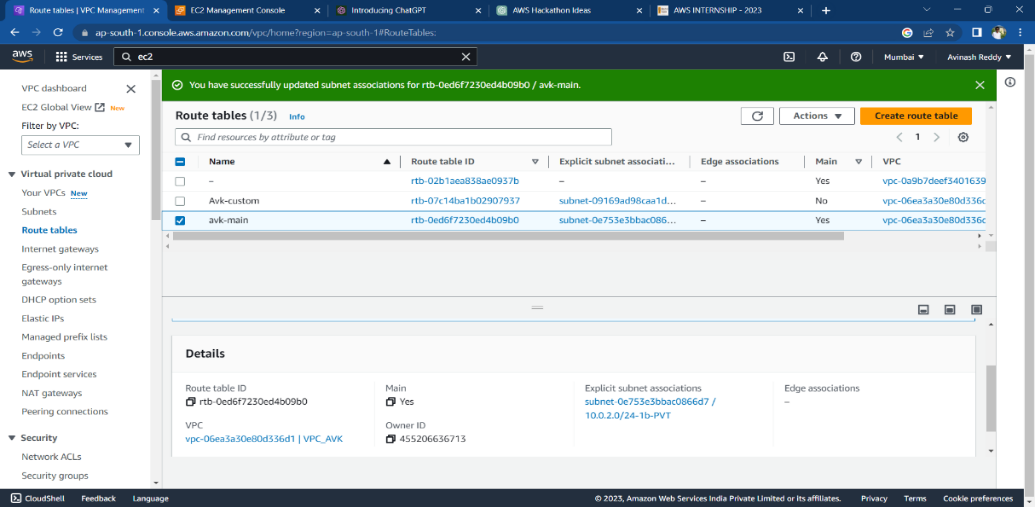
* Click on the "Create Subnet" button to create a new subnet.
* Configure the subnet settings:
* Select the VPC in which you want to create the subnet.
* Provide a name and a suitable CIDR block for the subnet. Ensure that the CIDR block falls within the IP address range of the VPC and doesn't overlap with other subnets
* Select the desired availability zone for the subnet. It's recommended to create subnets in multiple availability zones for high availability and fault tolerance.
* Configure the subnet's route table:
* ****Choose an existing route table or create a new one for the subnet. To make the subnet public, associate it with a route table that has a route to an internet gateway.
* Verify the details of the subnet, including the VPC, CIDR block, availability zone, route table, and NACL settings
* Once you have reviewed and confirmed the configuration, click on the "Create" button to create the public subnet.
* If you require multiple public subnets across different availability zones, repeat the above steps to create them.

**NAT GATEWAYS**

* Click on the "Create NAT Gateway" button to create a new NAT gateway.
* Configure the NAT gateway settings:
* Select the subnet in which you want to create the NAT gateway. The subnet must be a public subnet, meaning it should have a route to an internet gateway.
* Choose an existing Elastic IP address or allocate a new one to associate with the NAT gateway. The Elastic IP address serves as a public IP address for the NAT gateway.
* Verify the configuration details for the NAT gateway, including the selected subnet and Elastic IP address
* Click on the "Create NAT Gateway" button to create the NAT gateway. The creation process may take a few moments.
* Update route tables: After the NAT gateway is created, you need to update the route tables to direct the outbound traffic from private subnets to the NAT gateway.
* Go to the "Route Tables" section in the VPC Dashboard.
* Select the route table associated with the private subnets that need access to the internet via the NAT gateway.
* Add a new route with a destination of "0.0.0.0/0" (or the desired IP range) and set the target as the newly created NAT gateway.
* Test the connectivity by launching an instance in a private subnet and ensuring it can access the internet through the NAT gateway.

**ROUTE TABLE**

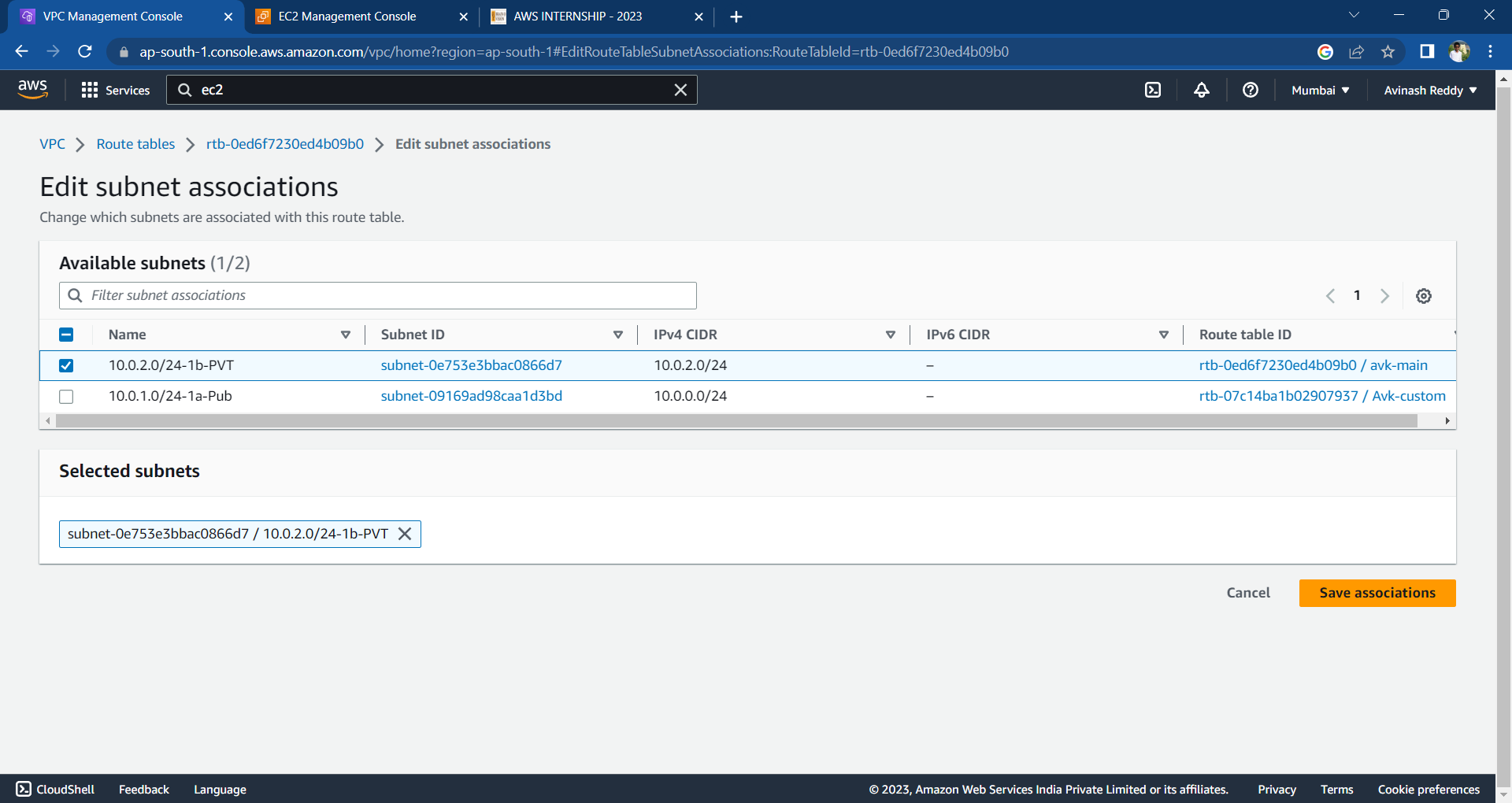
****

****

**PRIVATE SUBNET**

* Click on the "Create Subnet" button to create a new subnet.
* Configure the subnet settings:
* Select the VPC in which you want to create the subnet.
* Provide a name and a suitable CIDR block for the subnet. Ensure that the CIDR block falls within the IP address range of the VPC and doesn't overlap with other subnets
* Select the desired availability zone for the subnet. It's recommended to create subnets in multiple availability zones for high availability and fault tolerance.
* Configure the subnet's route table:
* Choose an existing route table or create a new one for the subnet. To make the subnet public, associate it with a route table that has a route to an internet gateway.

Verify the details of the subnet, including the VPC, CIDR block, availability zone, route table, and NACL settings.

****

**STAGE 2:**

Create an Auto Scaling Group:

* In the EC2 Auto Scaling console, click on "Auto Scaling Groups" in the sidebar.
* Click on "Create an Auto Scaling group" and follow the wizard.
* Select the launch configuration you created in the previous step.
* Configure the desired minimum, maximum, and desired capacity of instances.
* Configure any additional settings such as scaling policies and tags.
* Save your Auto Scaling group.

Create an Elastic Load Balancer:

* Go to the EC2 console.
* Click on "Load Balancers" in the sidebar.
* Click on "Create Load Balancer" and select the type of load balancer you want to create (e.g., Application Load Balancer, Network Load Balancer, or Classic Load Balancer).
* Configure the load balancer's settings, such as listeners, target groups, security groups, and availability zones.
* Save your load balancer.

Configure the Auto Scaling Group with the Load Balancer:

* In the EC2 Auto Scaling console, select your Auto Scaling group.
* Click on the "Edit" button in the "Details" tab.
* In the "Load balancing" section, select "Enable" for "Classic Load Balancer" or "Target groups" for "Application Load Balancer" or "Network Load Balancer."
* Select the load balancer and target groups you created in the previous step.
* Save your changes.

After following the above steps, we can connect to the webserver.

Hosting Jenkins and deploying a js project with the help of AWS code pipeline:

1) Launch an EC2 Instance in AWS using t2.micro with ports 22,80, and 8080 (Hosts jenkins)

2) Connect to EC2 instance securely through Putty

Downloading and installing Jenkins:

sudo yum update –y

sudo wget -O /etc/yum.repos.d/jenkins.repo \https://pkg.jenkins.io/redhat-stable/jenkins.repo

sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key

sudo yum upgrade

Install Java: sudo amazon-linux-extras install java-openjdk11 -y

Install Jenkins: sudo yum install jenkins -y

sudo systemctl enable jenkins

sudo systemctl start Jenkins

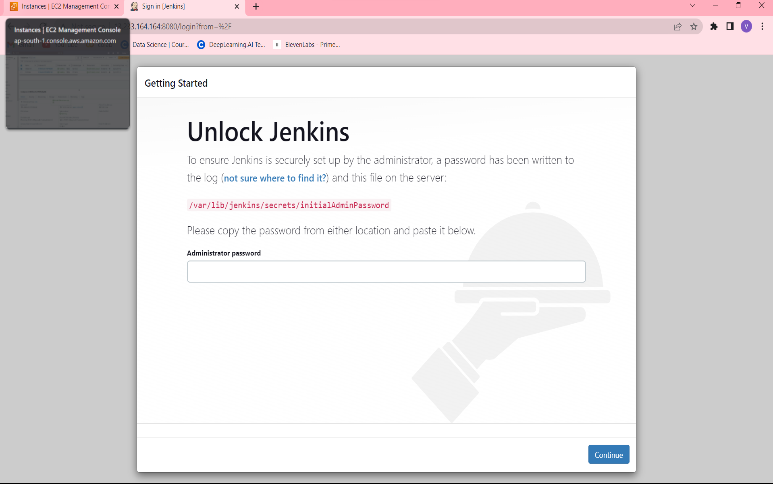
sudo systemctl status jenkins

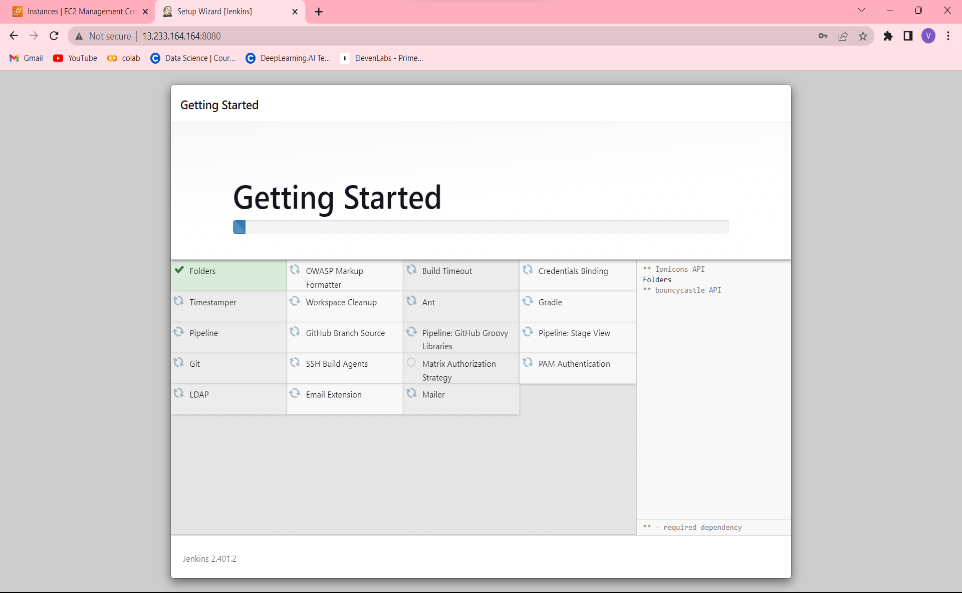
**Configuring Jenkins:**

1) Connect to http://<your\_server\_public\_DNS>:8080 from your browser. You will be able to access Jenkins through its management interface.

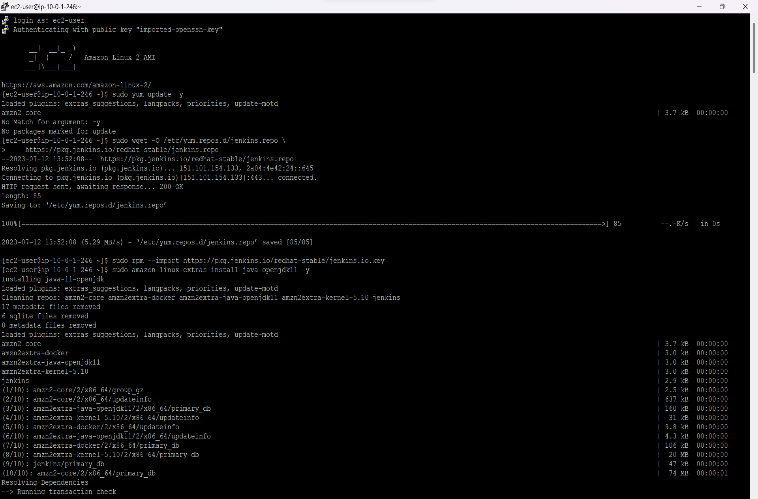
2) As prompted, enter the password found in sudo cat /var/lib/jenkins/secrets/initialAdminPassword.

3) The Jenkins installation script directs you to the Customize Jenkins page. Click Install suggested plugins.





4) Once the installation is complete, the Create First Admin User will open. Enter your information, and then select Save and Continue.

**Configuring Jenkins after hosting in AWS EC2:**

1. After connecting to jenkins, first go to your profile to change time zone from default to your local IST.

Profile -> Configure -> Scroll down -> defined time zone -> IST -> save

2. Next, go to dashboard to install few plugins which are required

Dashboard -> Manage Jenkins -> Manage Plugins -> Go to available -> search for AWS Codepipeline Integration -> select -> Install without resatart.

3. Now go back to dashboard to configure a program

Dashboard -> New Item -> Give a name to the item -> SampleCodePipelineProject -> Freestyle Project -> OK

Inside the item :General -> select - Execute concurrent buils if necessary

Source Code Management -> AWS CodePipeline -> Give necessary Region

In place of Access Code and Secret Code, attach an IAM Role to the Given EC2 instance

Open -> IAM in console -> Roles -> Create Roles -> one end EC2 -> other end AWS Codepipeline\_FullAccess & S3 -> create role

Attach this IAM role to the current running instance from security settings in EC2. Then in in Jenkins

Category -> Build

Provider -> JenkinsBuild

Version -> 1

Build Triggers -> Poll SCM -> Schedule -> \* \* \* \* \*

Build Environment -> for the moment keep it nothing

Build -> Execute Shell -> echo "Hello World"

Post-Build Actions -> AWS CodePipeline Publisher -> Output Location -> / -> SAVE

Now Jenikins is ready. Now we need to configure AWS CodePipeline

Configuring AWS CodePipeline post Jenkins Configuration

1. In aws console open AWS Codepipeline

Create Pipeline ->

Name - JenkinsDemoPipeline ->

newservice role is fine ->

Advanced Settings ->

Articat Store -> going with default

Encryption key -> Default

NEXT -> Source -> AWS Codecommit / Github

Repository name/ Connect with repo and rest everything default

NEXT -> Code BUild -> Add Jenkins -> Provider name - JenkinsBuild (Should Match exactly) -> URL - HTTP/Public iP:8080 -> Project Name - copy from jenkins project

NEXT ->Since it is just showing jenkins integration we will skip deploy step

Skip deploy -> Review -> Create Pipeline.

Dashboard -> Manage Jenkins -> manage plugins -> Available - Nodejs -> install without restart

Go back to Manage Jenkins -> Global Tool Configuration -> Insatall intended Nodejs Version ->

Click Add Nodejs -> Give a name - Node16 -> select the version - Nodejs 16.1.0 ( you have to select your verison according to your projet requirement) -> SAVE.

Now, Go back to your project in jenkins and configure it to run the project.

Go to Build Environment section -> Provide Node & npm bin/folder to PATH ->

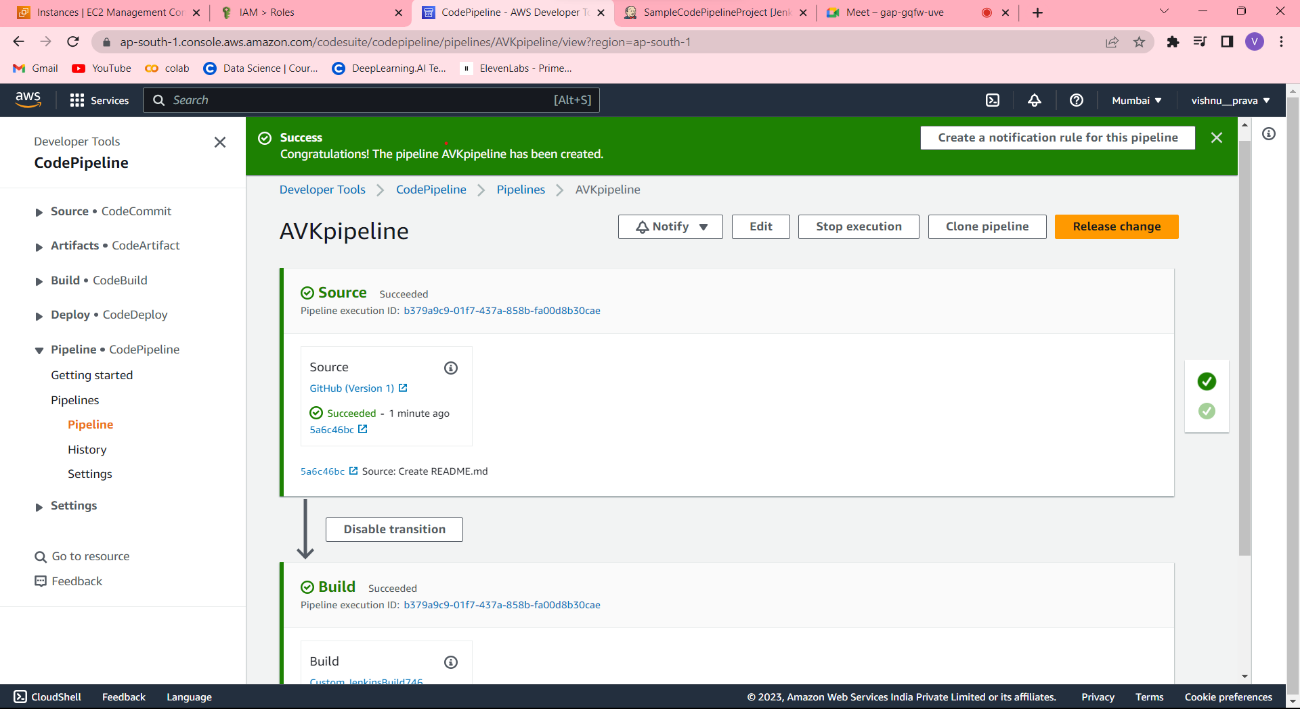
In Build Steps -> Execute shell ->

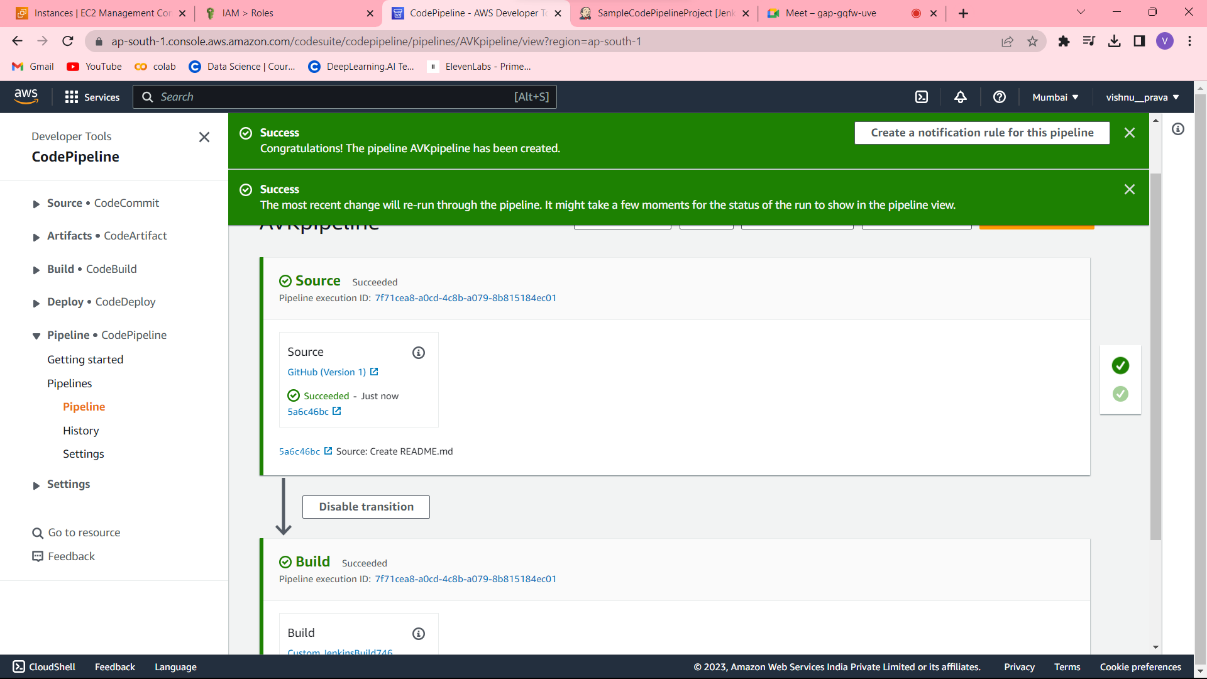
echo "Installing Nodejs Dependencies"

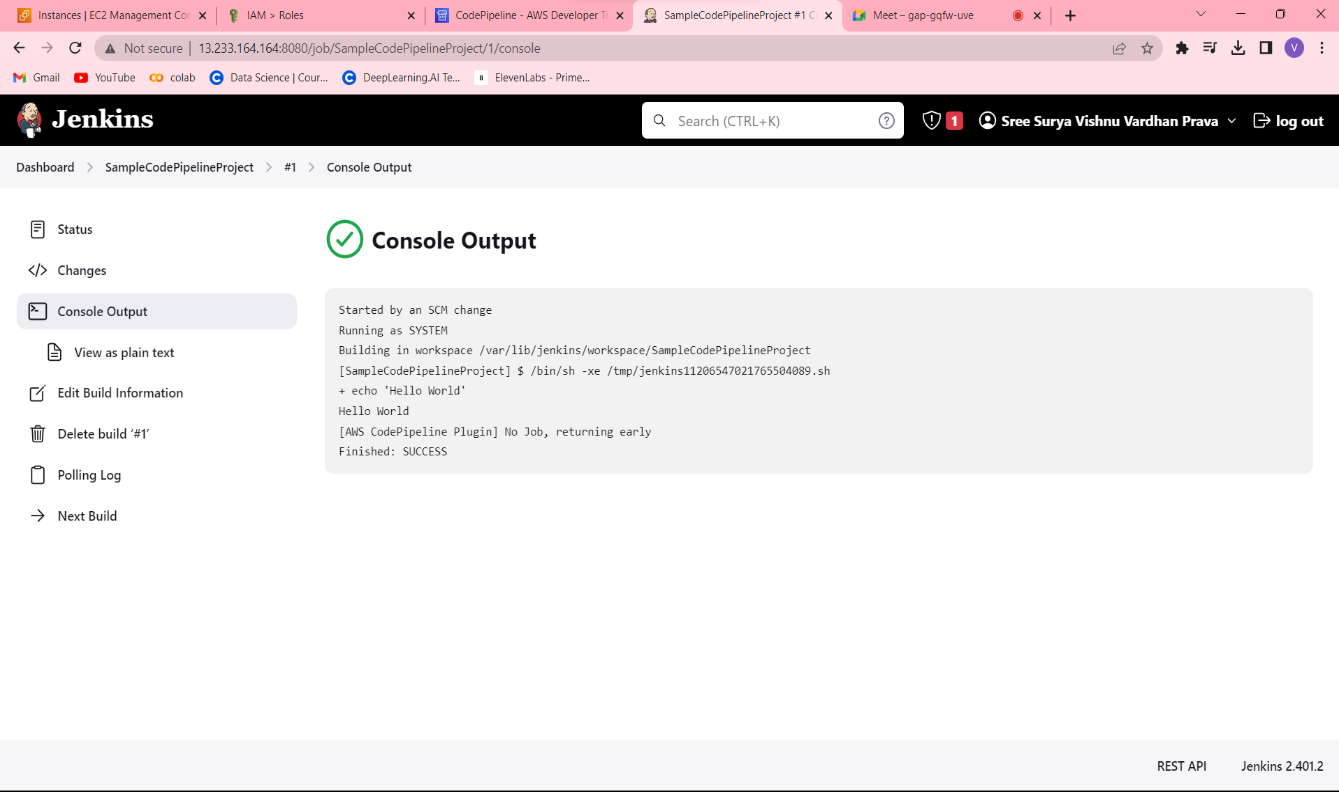
npm install

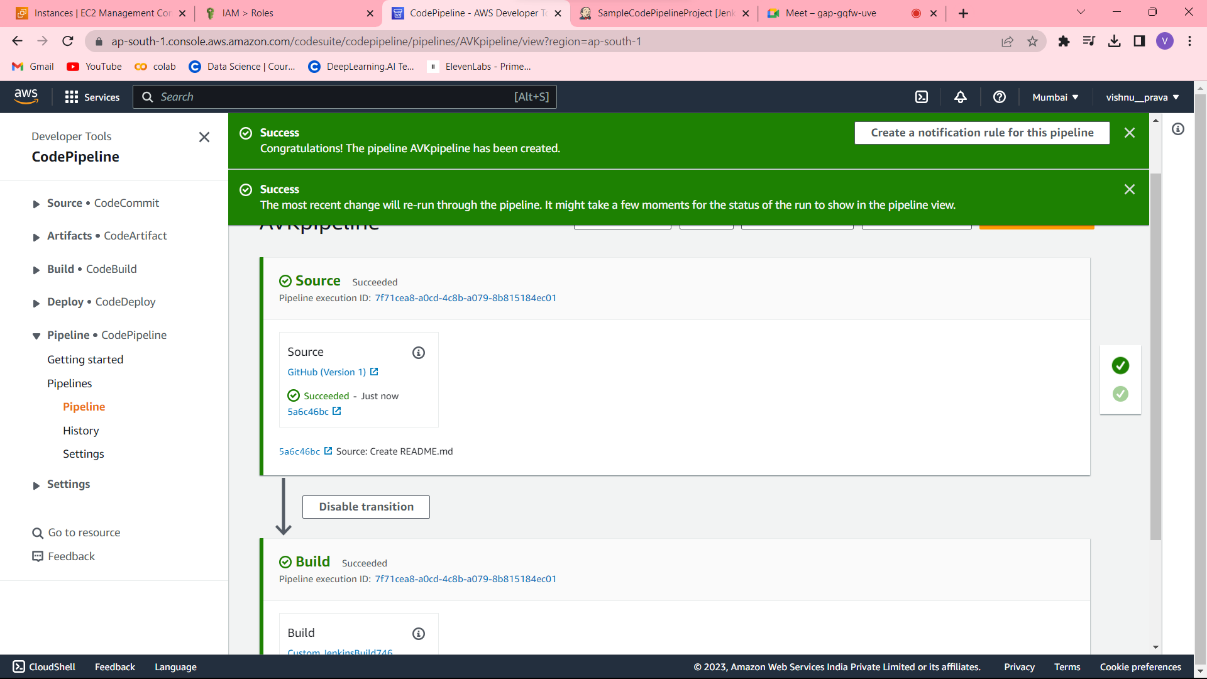
SAVE

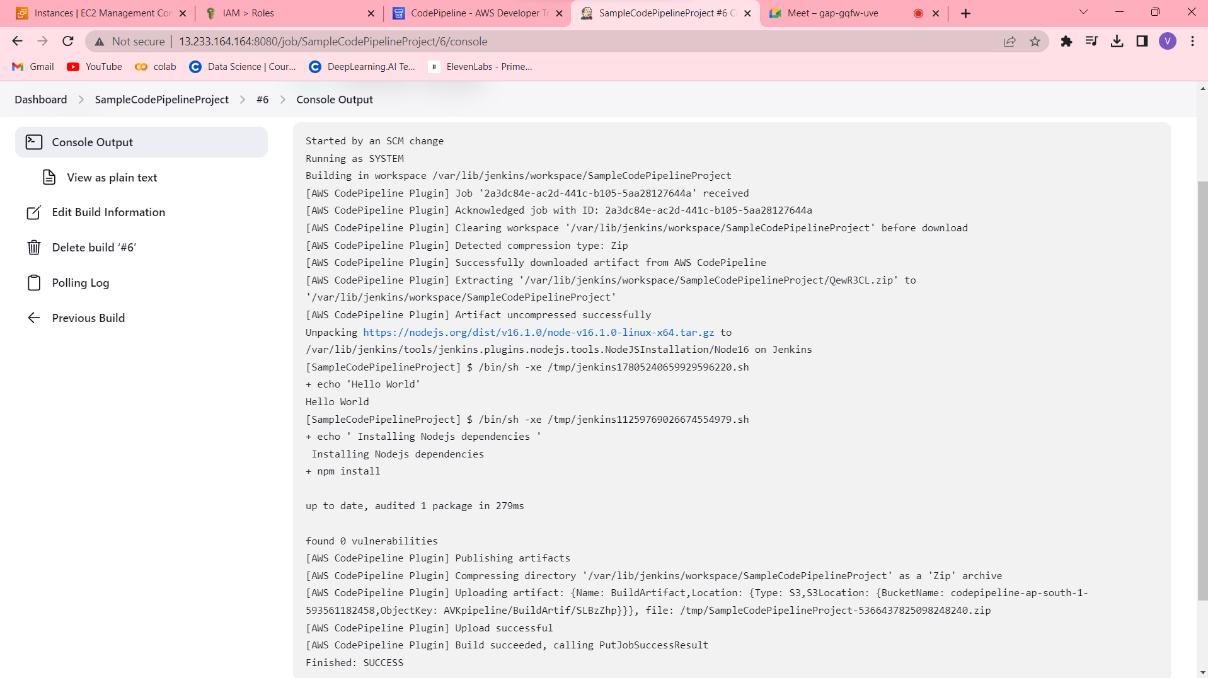
Release the change in AWS CodePipeline as well.Show success or failure both in console & Jenkins as well.

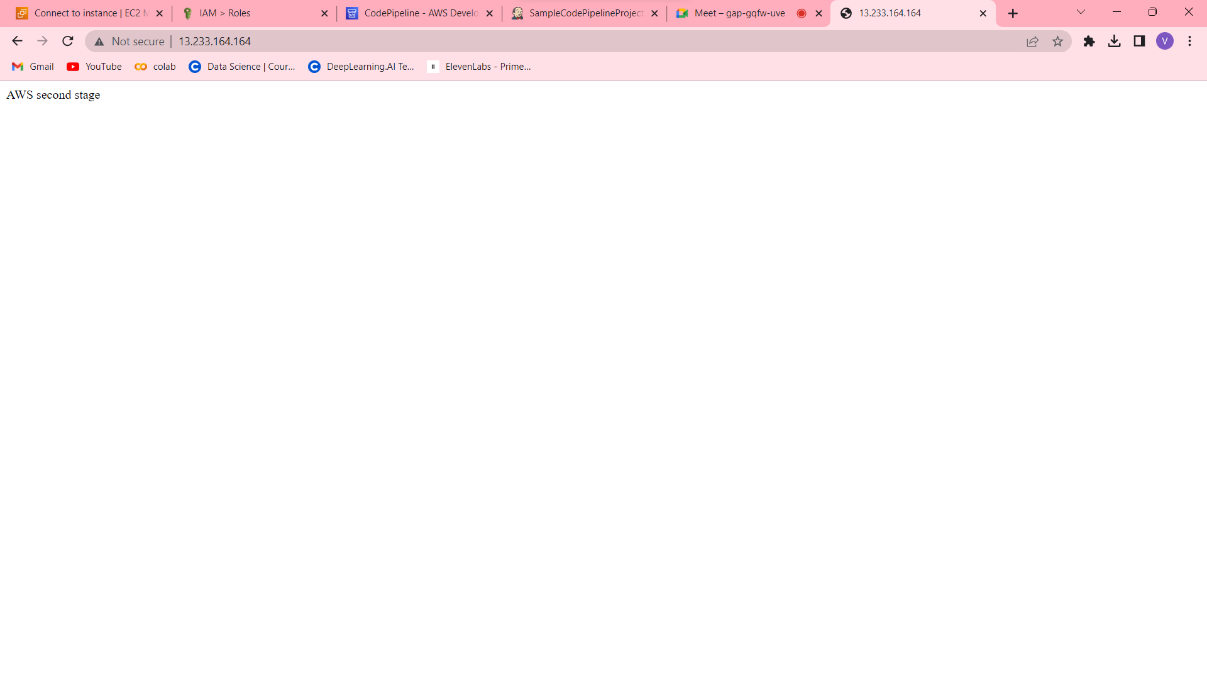












**STAGE 3:**

To connect a jump bastion instance to a database instance in AWS, you can follow these steps:

1. Launch a Jump Bastion Instance:

- Go to the EC2 console in the AWS Management Console.

- Click on "Launch Instances" to start the instance creation wizard.

- Choose an appropriate Amazon Machine Image (AMI) for the jump bastion instance, such as an Amazon Linux or Ubuntu Server.

- Select an instance type that suits your needs.

- Configure the instance details, including network settings, security groups, and storage options.

- Review your configuration and launch the instance.

2. Configure Security Groups:

- While launching the instance or after the launch, configure the security groups for both the jump bastion instance and the database instance.

- Create a new security group or choose an existing one for each instance.

- Define inbound and outbound rules to allow the necessary network traffic.

- For the jump bastion instance, allow SSH access (port 22) from your IP address or a specific range of IP addresses.

- For the database instance, allow the required ports for database connectivity (e.g., port 3306 for MySQL, port 5432 for PostgreSQL).

3. Connect to the Jump Bastion Instance:

- Retrieve the public IP address or DNS name of the jump bastion instance from the EC2 console.

- Use an SSH client (e.g., Terminal on macOS/Linux, PuTTY on Windows) to connect to the jump bastion instance.

- Provide the necessary credentials (SSH key pair or username/password) to establish the connection.

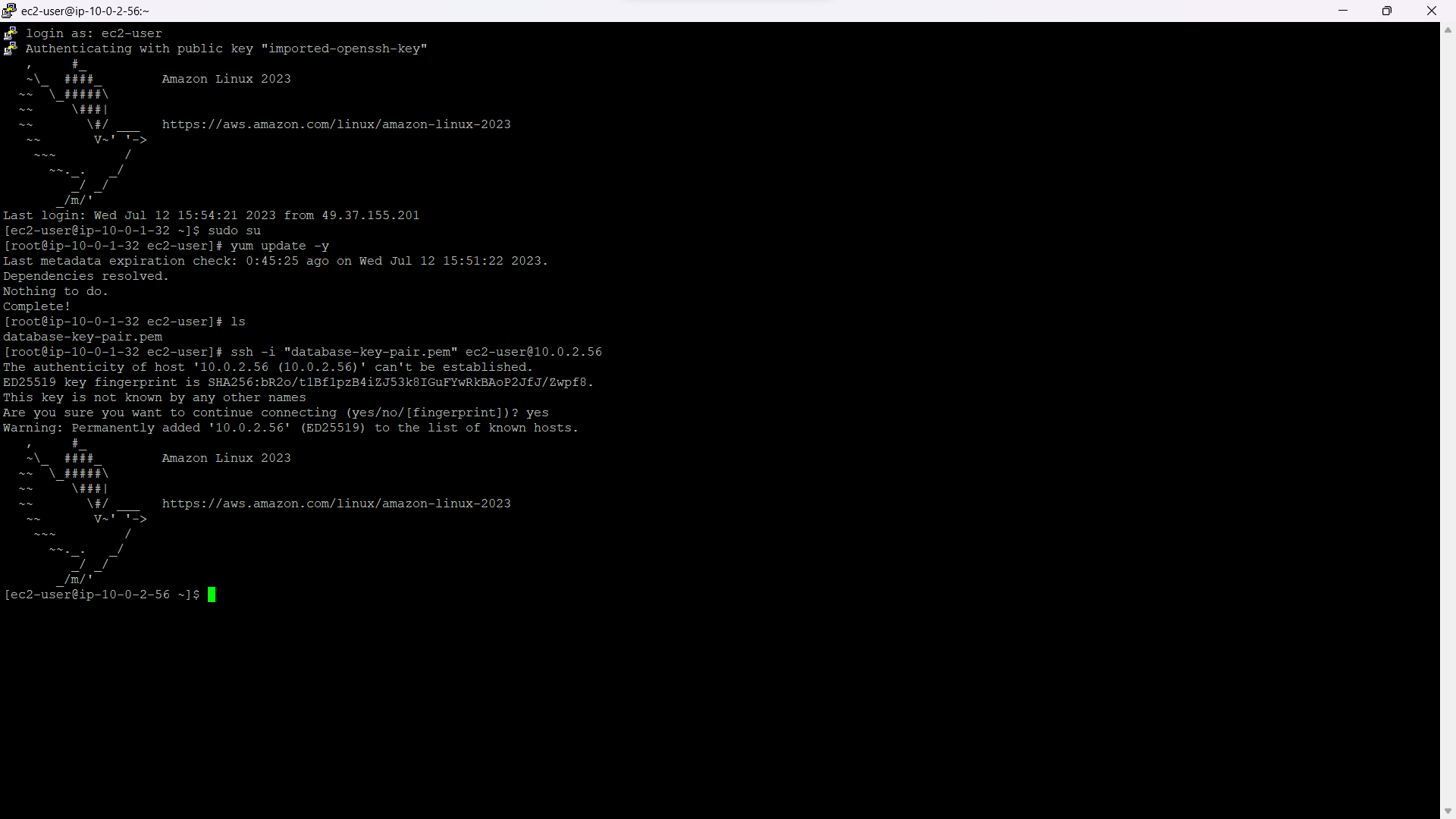
4. Connect to the Database Instance:

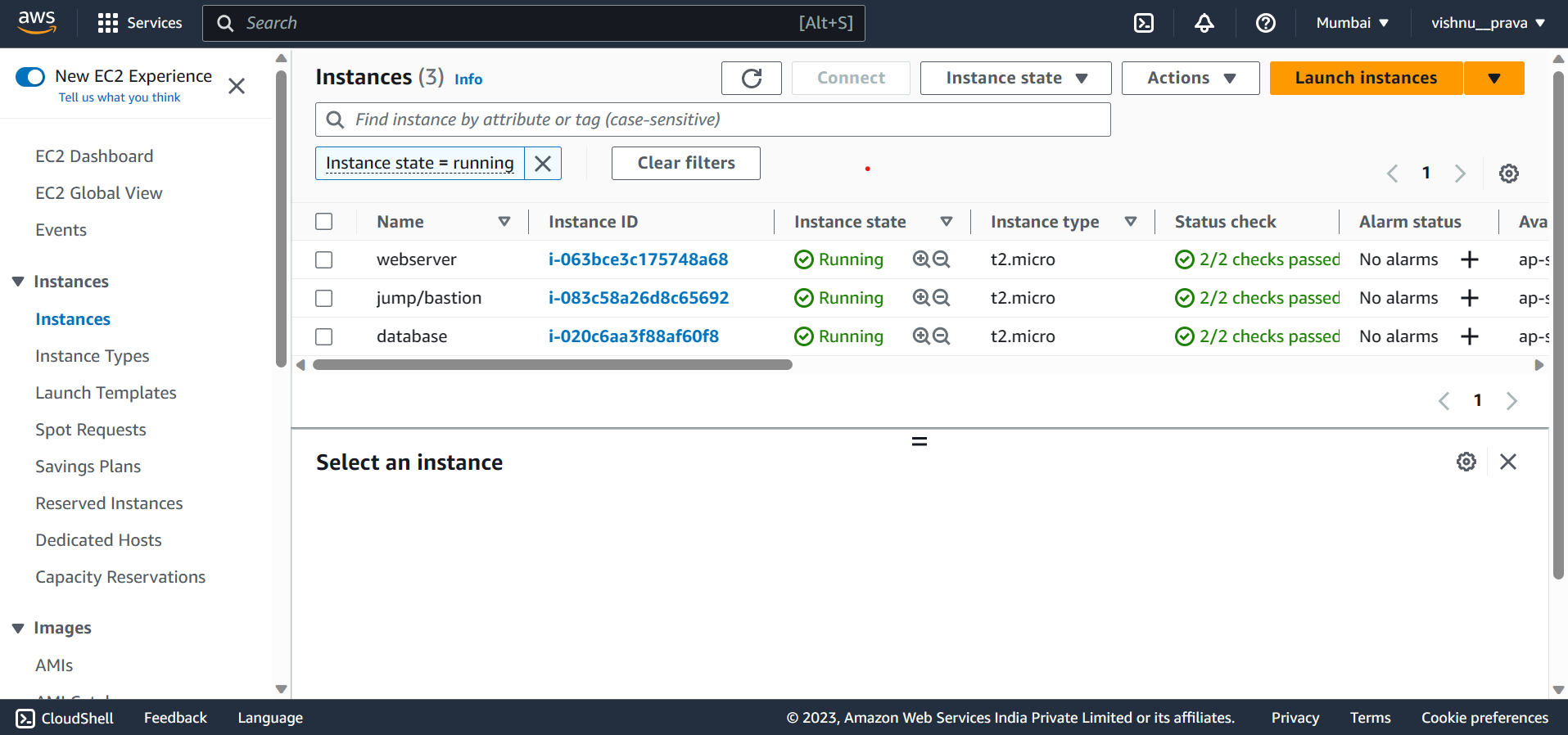
- From the jump bastion instance, you can establish a secure connection to the database instance.

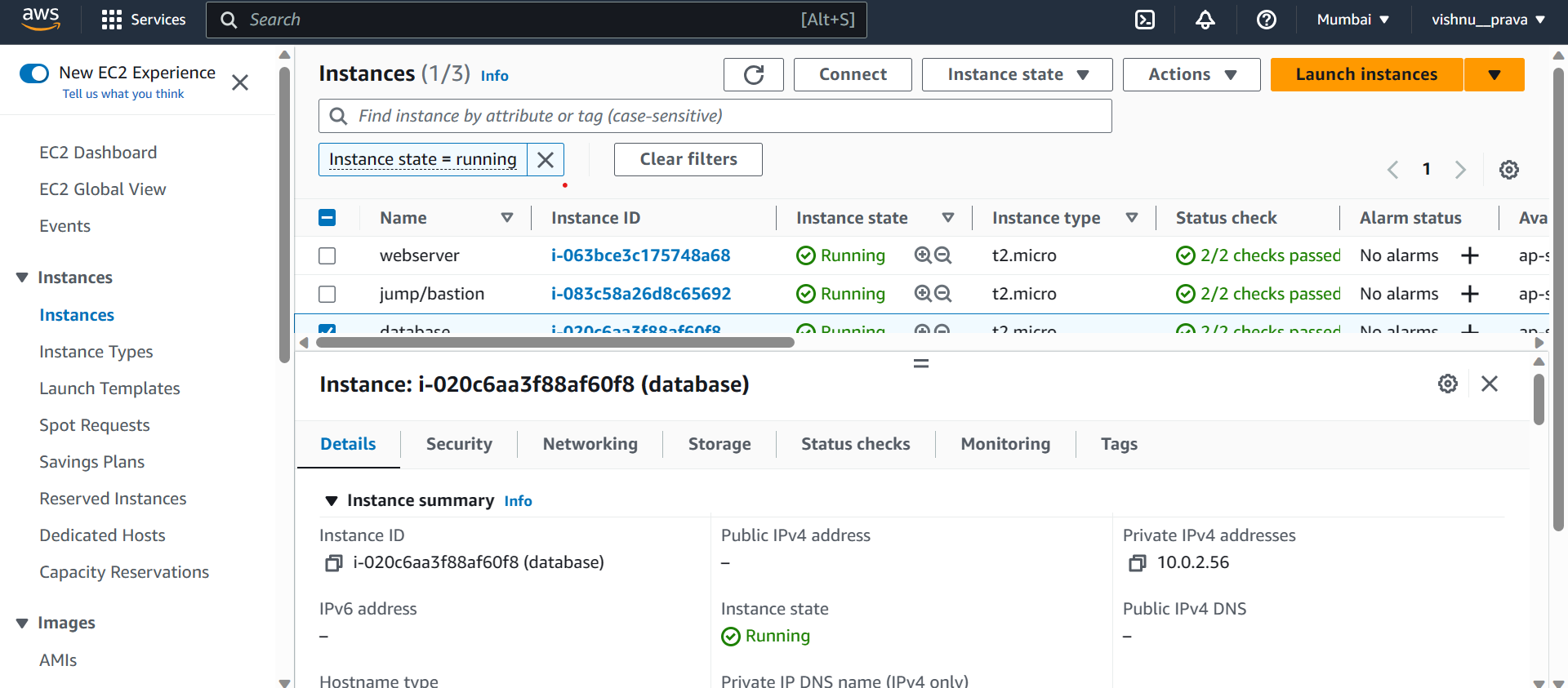
- Use the appropriate client or command-line tool for your database (e.g., MySQL Workbench for MySQL, psql for PostgreSQL).

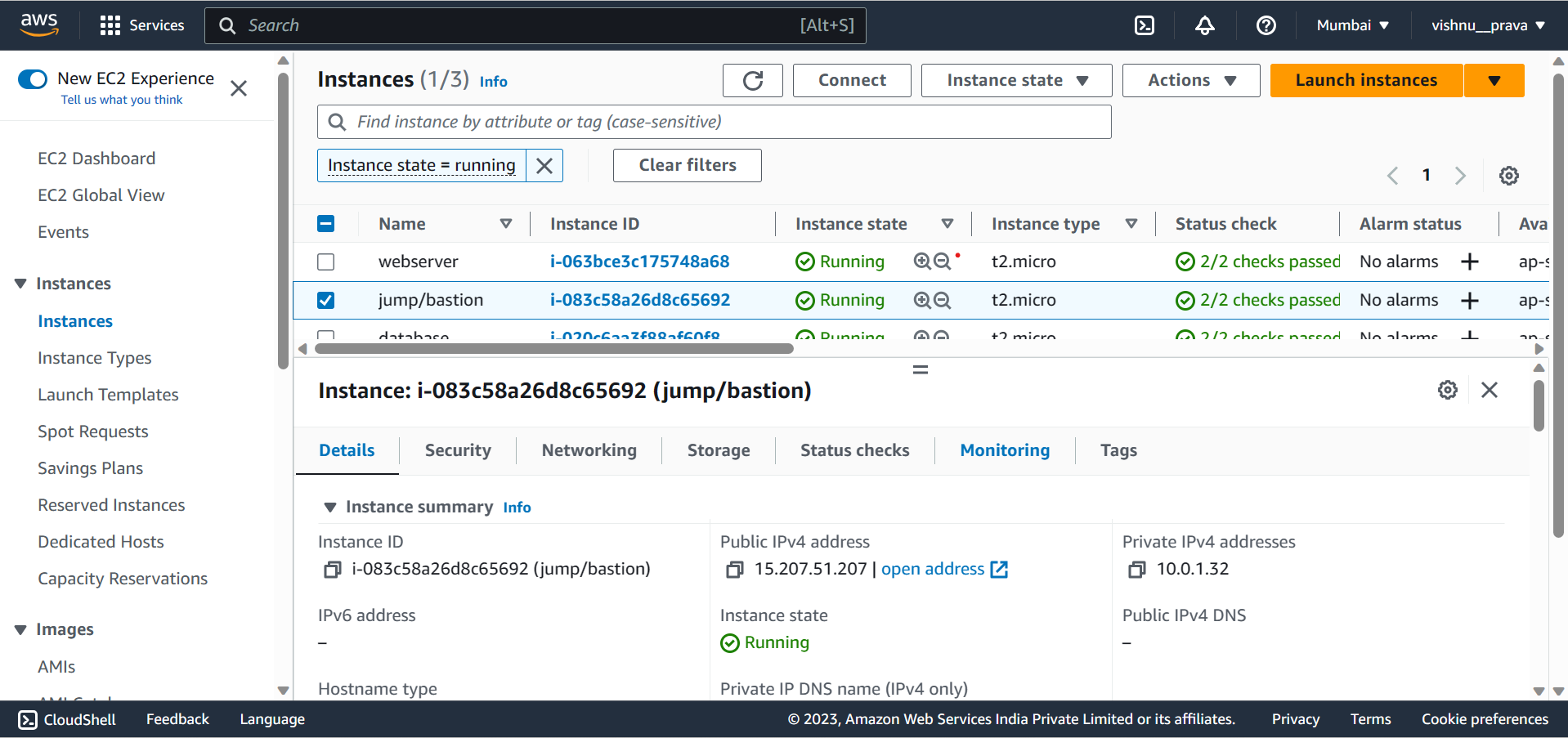
- Provide the necessary connection details such as the database host, port, username, and password.

- Connect to the database instance.









**Stage-4:**

In this stage we created nacl for subnets.

In AWS, Network Access Control Lists (NACLs) are an optional layer of security that act as stateless firewalls for controlling inbound and outbound traffic at the subnet level. NACLs operate at the subnet level and evaluate rules in a sequential order to determine whether to allow or deny traffic. Each subnet in AWS can be associated with one NACL, and by default, a subnet is associated with the default NACL.

Here are the steps to create and attach a Network Access Control List (NACL) to subnets in AWS:

1. Open the Amazon VPC Console:

- Go to the Amazon VPC console in the AWS Management Console.

2. Create a Network Access Control List (NACL):

- In the VPC dashboard, click on "Network ACLs" in the sidebar.

- Click on "Create Network ACL" and specify a name and the VPC for the NACL.

- Once created, the NACL will have default rules that allow all inbound and outbound traffic.

3. Configure Inbound and Outbound Rules:

- Select the newly created NACL and click on the "Inbound Rules" or "Outbound Rules" tab.

- Click on "Edit" and add or modify the rules according to your requirements.

- Rules can be based on IP addresses, port ranges, protocols, and whether to allow or deny traffic.

- Rules are evaluated in the order they appear, so ensure they are ordered correctly.

4. Associate the NACL with Subnets:

- Click on the "Subnet Associations" tab in the NACL configuration.

- Click on "Edit" and select the subnets you want to associate with the NACL.

- You can choose to associate multiple subnets with the same NACL.

- Make sure to confirm your changes to associate the NACL with the selected subnets.

5. Verify and Test:

- Once the NACL is associated with the subnets, it will start governing the traffic.

- Verify that the NACL rules are correctly configured and are allowing or denying traffic as intended.

- Test the connectivity to ensure that the traffic is behaving as expected.

By following these steps, you can create and attach a Network Access Control List (NACL) to subnets in AWS. NACLs provide an additional layer of security by controlling inbound and outbound traffic at the subnet level, allowing you to define fine-grained rules for network traffic in your VPC.

