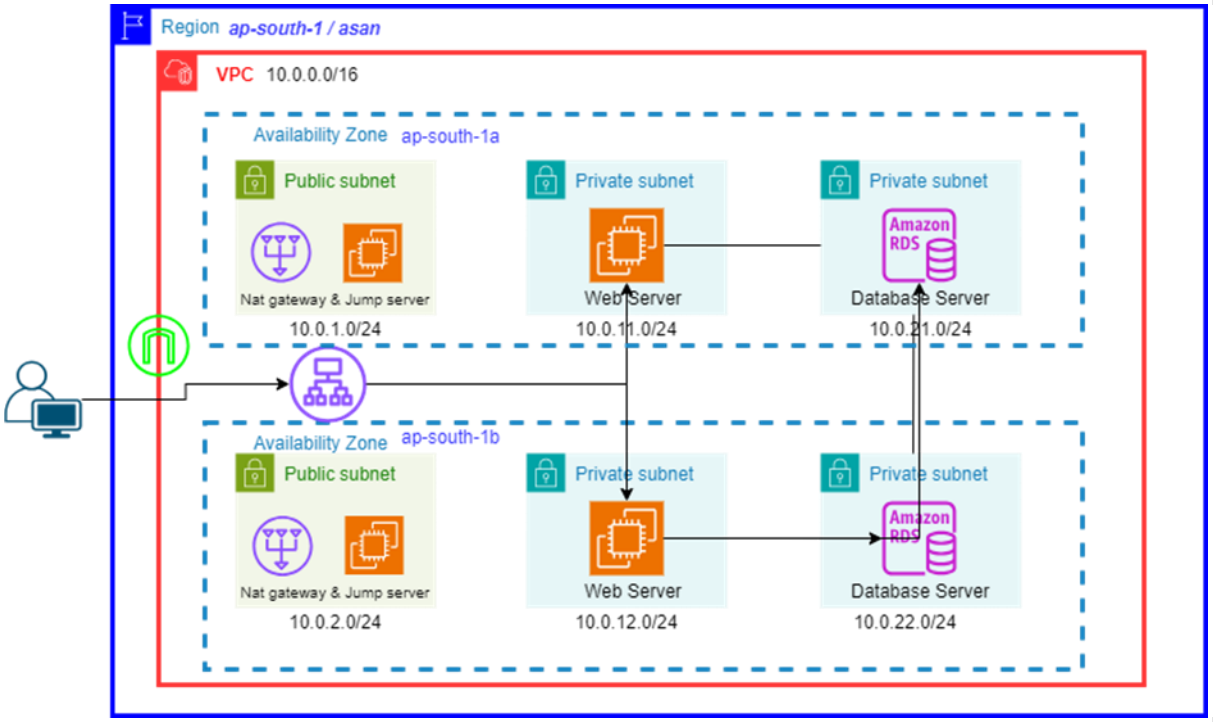
🚀 Step-by- Step Deployment of a 3-Tier Architecture on AWS! 🌐

I'm successfully deployed a simple comprehensive 3-tier architecture on AWS, ensuring a secure, scalable, and high-performing environment. Here's a breakdown of the deployment:



🏗 Architecture Components:

1. **Virtual Private Cloud (VPC)**: Created a dedicated VPC for network isolation and control.
    - **Internet Gateway (IGW)**: Attached to the VPC to enable internet access.
    - **NAT Gateway**: Configured with an Elastic IP for secure internet access from private instances.
2. **Subnets**:
    - **Public Subnet**: Hosts the Internet Gateway and Load Balancer.
    - **Private Subnets**: Deployed across multiple availability zones for high availability and security. Hosts the web server and database server in this subnet.
3. **Jump Server**:
    - **Security Group**: Configured to allow secure SSH access to the private instances.

4. **Application Servers**:

    - **Instances**: Two EC2 instances running Apache and PHP, hosted in private subnets.
    - **Security Group**: Configured to manage traffic to the application servers. Allow Load balancer security group only.
    - **PHP Installation**: Set up and configured PHP, along with database endpoint additions.

5. **Database Server**:

    - **Amazon RDS**: Running MySQL, deployed in a private subnet for database management and reliability.
    - **Security Group**: Ensures secure database connections. Allow web server security group only

6. **Load Balancer**:

    - **Target Group**: Configured to include the private instances.
    - **Testing**: Verified load balancer functionality to ensure optimal fperformance.

🔍 Project Highlights:

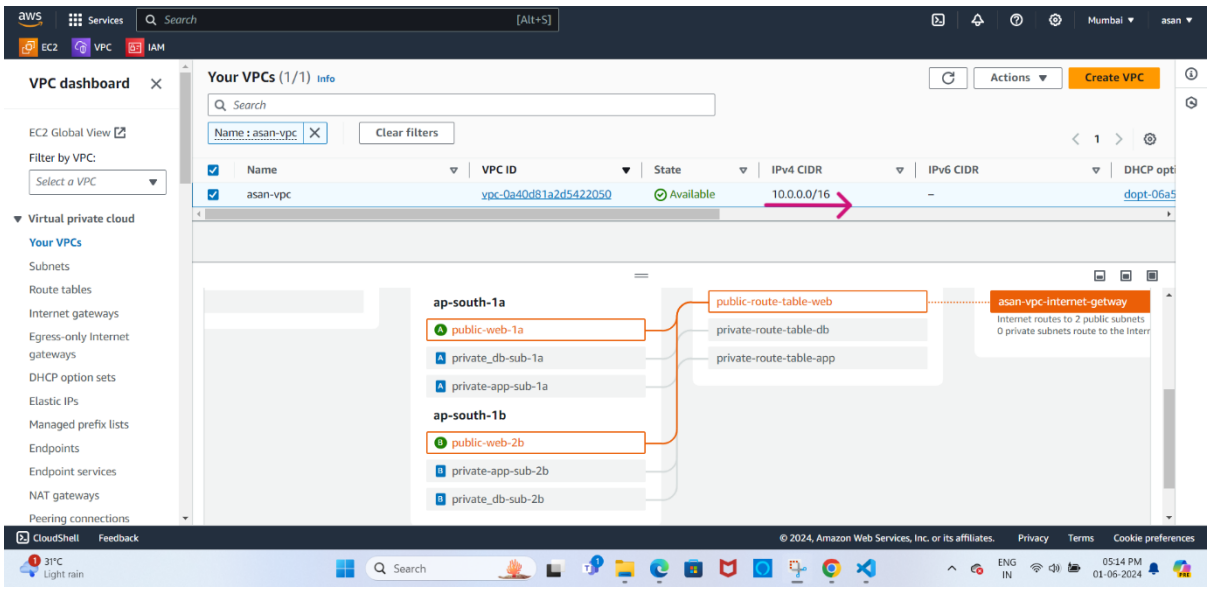
- **Security**: Implemented strict security group rules and network ACLs.
- **Load Balancer**: Depons upon the health check and attributes pass the load to different web servers.
- **High Availability**: Multi-AZ deployment for both application and database layers.

Step 1 : Create VPC

1. Sign in to the AWS Management Console.
2. In the top left corner, click on the **Services** drop-down menu and select **VPC**.
3. In the VPC Dashboard, click on **Your VPCs** in the left navigation pane.
4. Click on the **Create VPC** button.
5. In the Create VPC wizard:

    - **Name tag**: Enter a name for your VPC (e.g., **asan-vpc**).
    - **IPv4 CIDR block**: Enter **10.0.0.0/16**.
    - Leave the other settings as default.

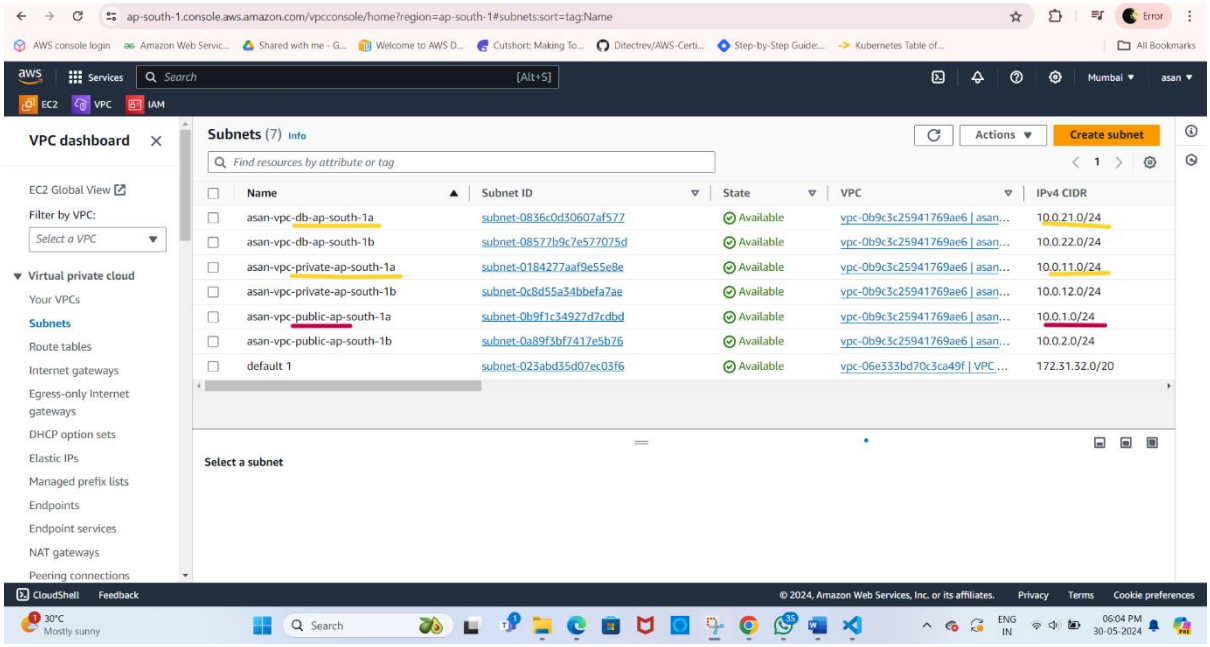
6. Click **Create VPC**.



Step 2: Create Subnets

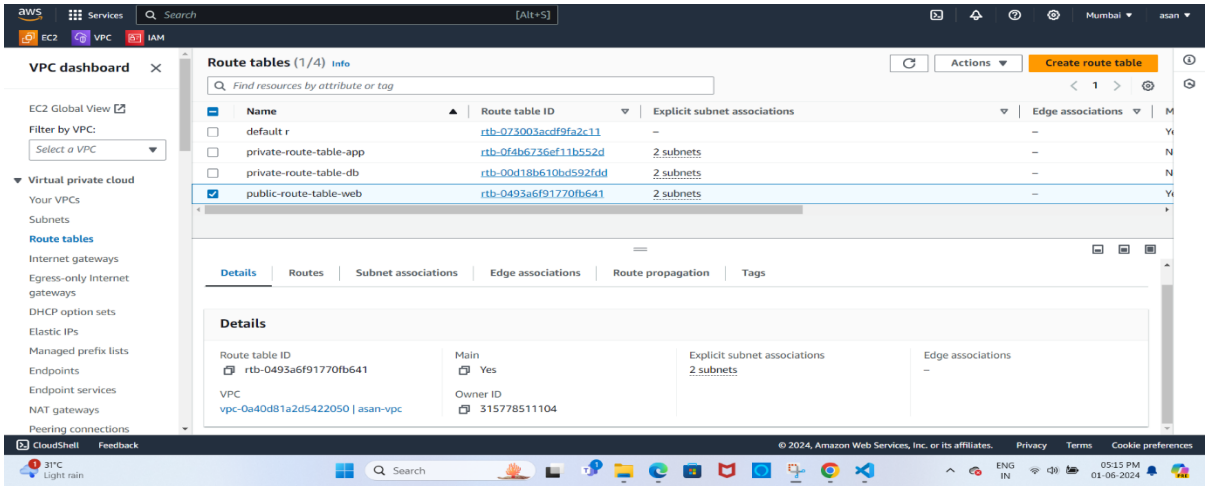
You need to create public and private subnets in two availability zones (AZs).

1. In the left navigation pane, click on **Subnets**.
2. Click on the **Create Subnet** button.
3. In the Create Subnet wizard:
    - **Name tag**: Enter a name for the public subnet (e.g., **public-web-1a**).
    - **VPC**: Select the VPC you created (**asan-vpc**).
    - **Availability Zone**: Select an AZ (e.g., **ap-south-1a**).
    - **IPv4 CIDR block**: Enter **10.0.1.0/24**.
4. **Public Subnet in 2nd AZ**: **public-web-1b** with CIDR **10.0.2.0/24** in **ap-south-1b**.
5. **Private App Subnet in 1st AZ**: **private-app-sub-1a** with CIDR **10.0.11.0/24** in **ap-south-1a**.
6. **Private App Subnet in 2nd AZ**: **private-app-sub-2b** with CIDR **10.0.12.0/24** in **ap-south-1b**.
7. **Private DB Subnet in 1st AZ**: **private-db-sub-1a** with CIDR **10.0.21.0/24** in **ap-south-1a**.
8. **Private DB Subnet in 2nd AZ**: **private-db-sub-2b** with CIDR **10.0.22.0/24** in **ap-south-1b**.



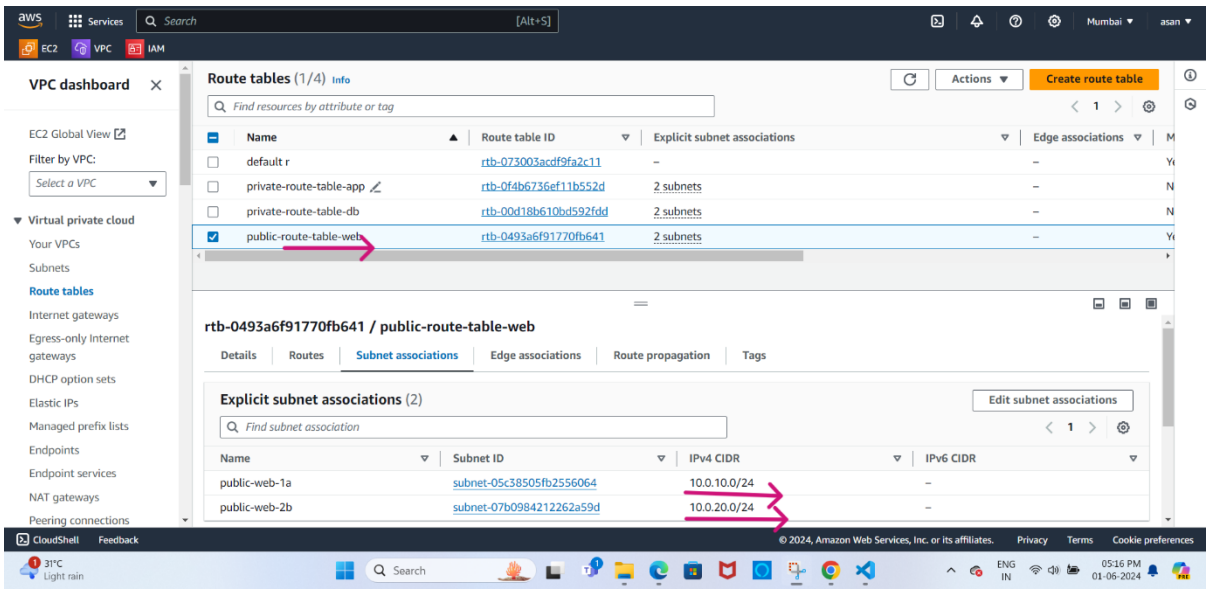
Step 3: Create Route Tables

1. In the left navigation pane, click on **Route Tables**.
2. Click on the **Create route table** button.
3. Enter a name for the public route table (e.g., **public-route-table-web**).
4. Select your VPC (**asan-vpc**) and click **Create route table**.



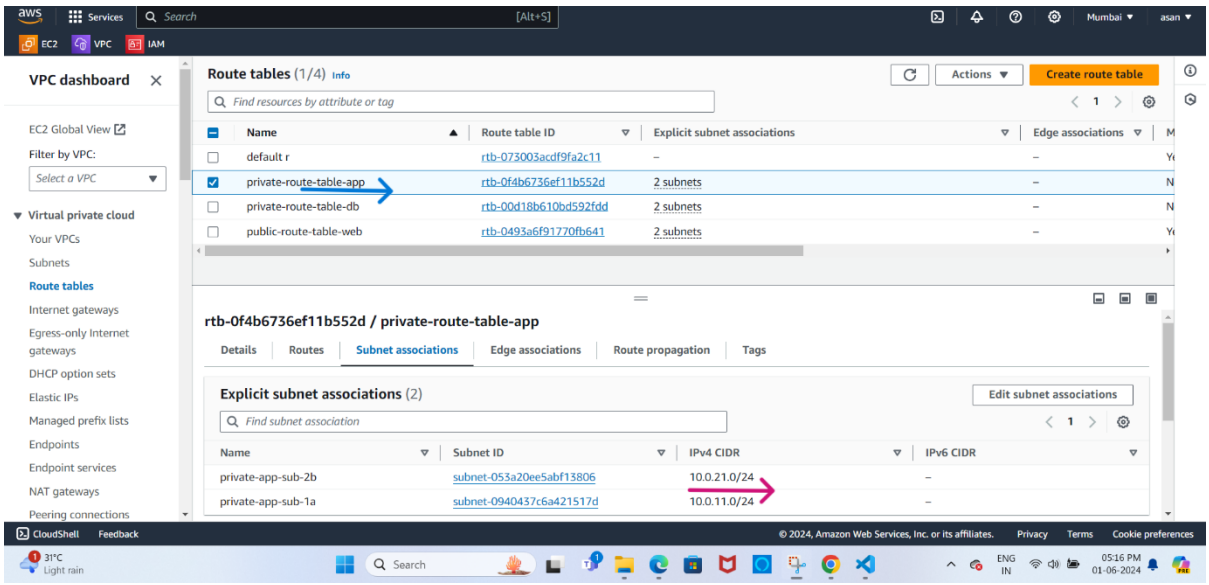
Step 4: Subnet associations

- Select the public route table (**public-route-table-web**).
- Click on the **Subnet associations** tab.
- Click **Edit subnet associations**.
- Select the public subnets (**public-web-1a** and **public-web-1b**) and **Save**.



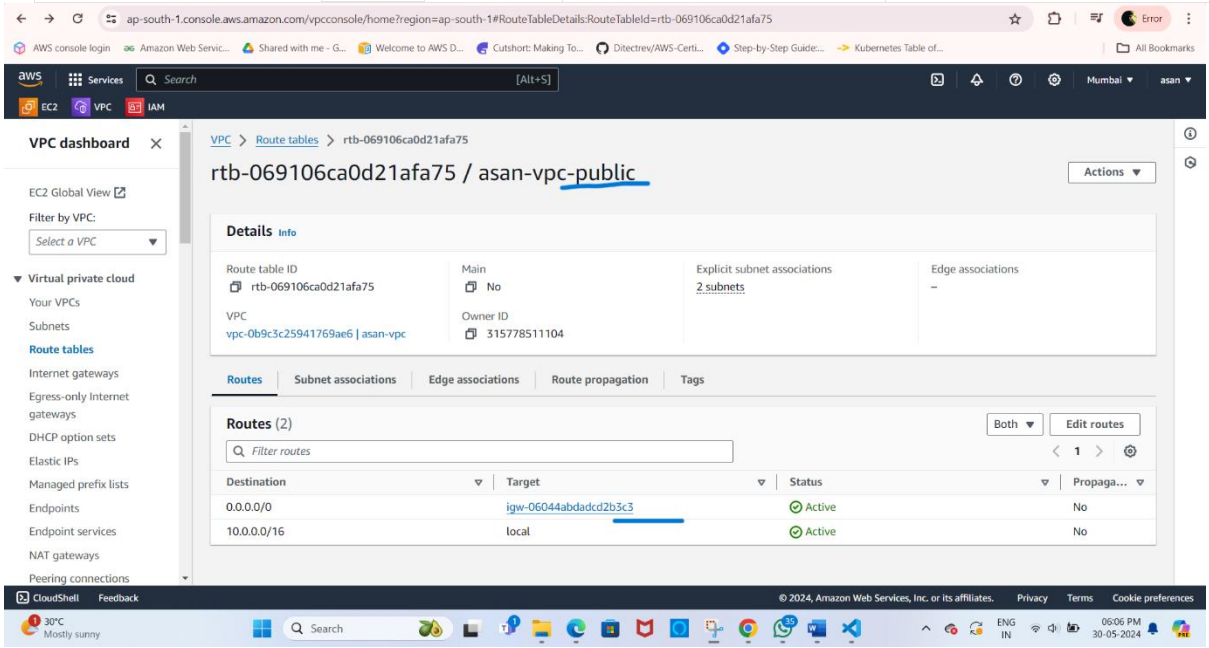
Private Route Tables for Web Server and Database Server

1. Same like a public route table, click on **Route Tables**.
2. Click on the **Create route table** button.
3. Enter a name for the public route table (e.g., **private-route-table-app**) for web server and (**private-route-table-db**) subnet for database server .
4. Select your VPC (**asan-vpc**) and click **Create route table**.
5. Select the public route table (**public-route-table-web**).
6. Click on the **Subnet associations** tab.
7. Click **Edit subnet associations**.
8. Select the public subnets (**private-app**) for private route table app and **(private-db**) subnet for private route table db
9. Click **Save**.



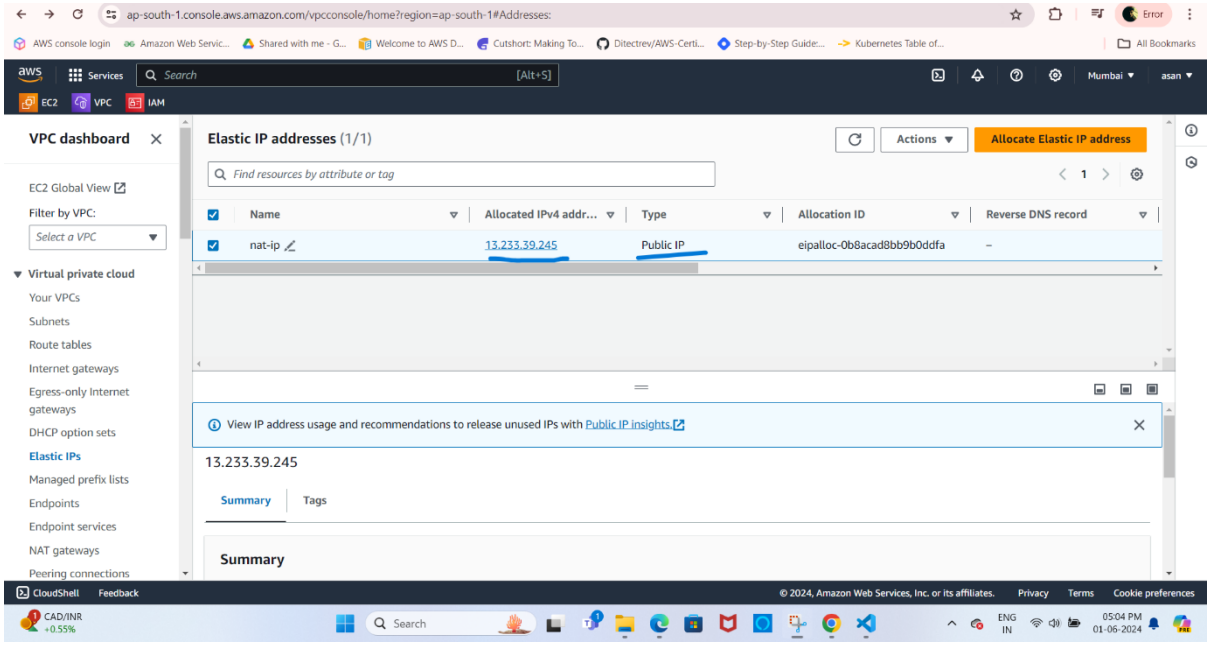
Step 5: Create an Internet Gateway and attach to VPC

1. In the left navigation pane, click on **Internet Gateways**.
2. Click on the **Create internet gateway** button.
3. Enter a name for the internet gateway (e.g., **asan-vpc-internet-gateway**).
4. Click **Create internet gateway**.
5. Select the newly created internet gateway and click **Actions**, then **Attach to VPC**.
6. Select your VPC (**asan-vpc**) and click **Attach internet gateway**.



Step 6: Allocate Elastic IP

1. Navigate to VPC and then select Elastic Ips
2. Select Allocate Elastic IP address
3. In Network border group select the public subnet like (public-subnet)
4. Then allocate



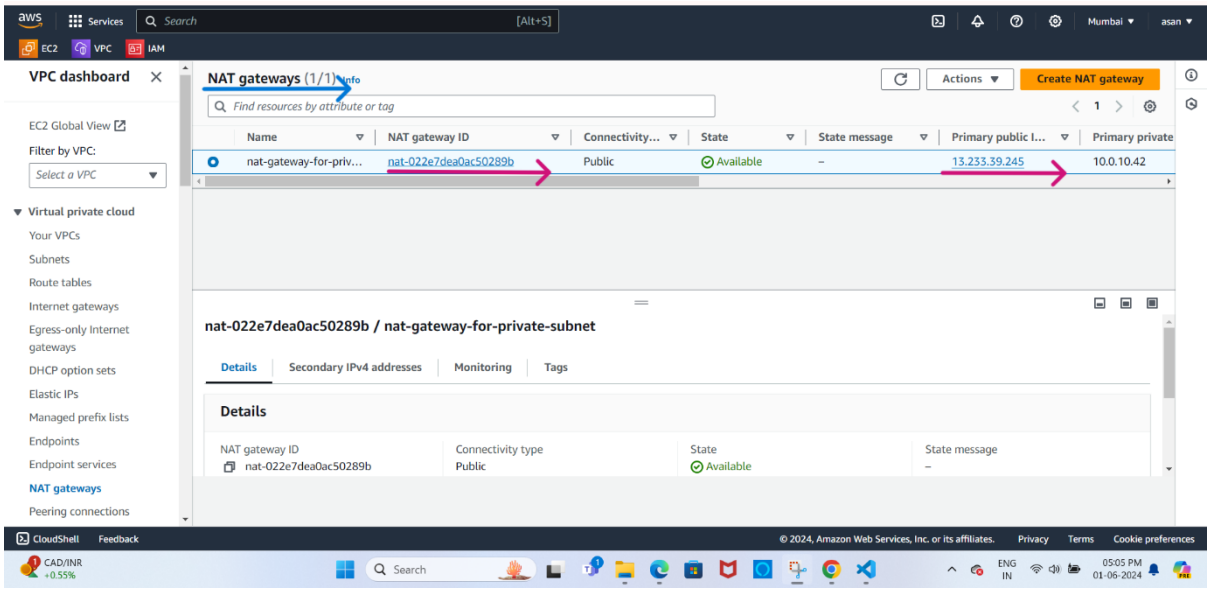
Step 7: Create a NAT Gateway and Allocate an Elastic IP

1. **Navigate to NAT Gateways**:

   - Go to the VPC Dashboard in your AWS Management Console.
   - Click on **NAT Gateways** in the left navigation pane.
   - Click on the **Create NAT Gateway** button.

2. **Create the NAT Gateway**:

   - Select a public subnet for the NAT Gateway (e.g., **public-web-1a**).
   - Allocate a new Elastic IP by clicking **Allocate Elastic IP** and then **Allocate**.
   - Click **Create NAT Gateway**. The NAT Gateway will be created and its status will change to "Available".



Step 8: Update Route Tables for Private Subnets and add route to Nat Gateway

1. **Navigate to Private Route Tables App**:
    - In the left navigation pane, click on **Route Tables**.
    - Select the route table for your private application subnets (**private-route-table-app**).
2. **Edit Routes**:
    - Click on the **Routes** tab, then click **Edit routes**.
    - Click **Add route**.
    - In the **Destination** field, enter **0.0.0.0/0**.
    - In the **Target** field, select the NAT gateway you created.
    - Click **Save routes**.
    -

Step 9: Create EC2 Instances (jump server)

Create Security Group for Instance (jump server)

1. **Navigate to VPC**:
    - In the left navigation pane, click on **Security Groups** under **Security**.
2. **Create Security Group**:
    - Click on the **Create security group** button at the top right.
3. **Configure Security Group Details**:
    - **Name tag**: Enter a name for your Security Group (e.g., **sg-Bastion_Host**).
    - **Description**: Provide a description (e.g., **Security group for jump server**).
    - **VPC**: Select the VPC where this Security Group will be applied.

Add Inbound Rules

1. **Inbound Rules**:
    - Click on the **Inbound rules** tab and then click **Add rule**.
    - **Type**: Select the type of traffic to allow (e.g., **ssh**).
    - **Protocol**: This field will be auto-filled based on the type selected or "**TCP**".
    - **Port range**: Specify the port range (e.g., **22** for ssh).

- **Source**: Define the source IP range or security group (e.g., **0.0.0.0/0** to allow traffic from anywhere).

Add Outbound Rules

1. **Outbound Rules**:

   - Click on the **Outbound rules** tab and then click **Add rule**.
   - **Type**: Select the type of traffic to allow (e.g., **All traffic**).
   - **Protocol**: This field will be auto-filled based on the type selected.
   - **Port range**: Specify the port range (e.g., **0-65535** for all ports).
   - **Destination**: Define the destination IP range or security group (e.g., **0.0.0.0/0** to allow traffic to anywhere).

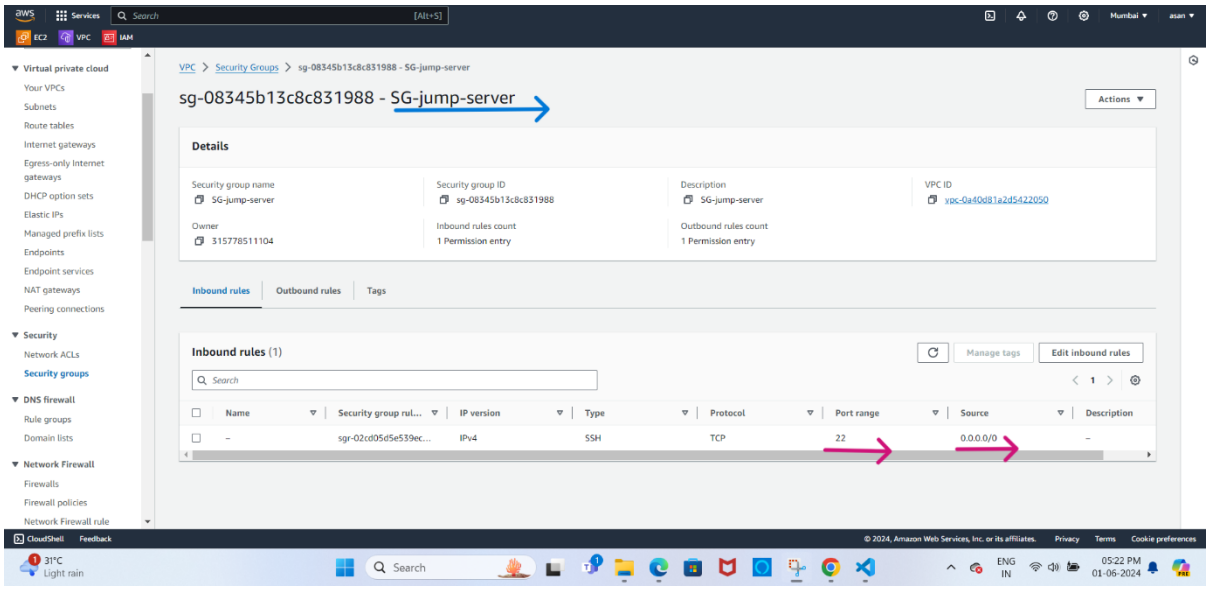
Step 5: Review and Create

1. **Review the Rules**:

   - Ensure that all the inbound and outbound rules are correctly configured as per your requirements.

2. **Create Security Group**:

   - Click **Create security group** to finalize the creation.



Step 10: Create EC2 Instance (Jump server)

Open the EC2 Dashboard:

- In the AWS Management Console, click on **Services** at the top left.
- Under **Compute**, select **EC2**.
- **Click on Launch Instance**:
- In the EC2 Dashboard, click the **Launch Instance** button.

Configure the Instance

1. **Choose an Amazon Machine Image (AMI)**:
    - Select an AMI based on your needs. For example, select **Amazon Linux 2 AMI** (for general purposes) or another AMI that suits your use case.
    - Click **Select**.
2. **Choose an Instance Type**:
    - Select the instance type. For general purposes, **t2.micro** (eligible for the free tier) is a good starting point.
    - Click **Next: Configure Instance Details**.
3. **Select an Existing Key Pair or Create a New Key Pair**:
    - Select an existing key pair if you have one.
    - If you don’t have a key pair, create a new one. This key pair will be used to securely connect to your instance.
    - Download the key pair and store it securely. You won't be able to download it again.
4. **Configure Network Settings**:
    - Ensure that the instance is launched in the correct VPC and subnet.
    - Select the vpc and public subnet and make sure Auto-assign public IP is enable.
    - Click **Next: Add Storage**.
5. **Add Storage**:
    - Specify the size and type of storage. The default of 8 GB General Purpose (SSD) is typically sufficient for testing.
    - Click **Next: Add Tags**.
6. **Add Tags**:
    - (Optional) Add tags to help you identify your instances. For example, you might add a tag with the key **Name** and value **MyEC2Ins1tance**.
    - Click **Next: Configure Security Group**.

Step 4: Configure Security Group

1. **Add Security Group**:
    - select an existing one that we create for jump server.
    - Click **Review and Launch**.

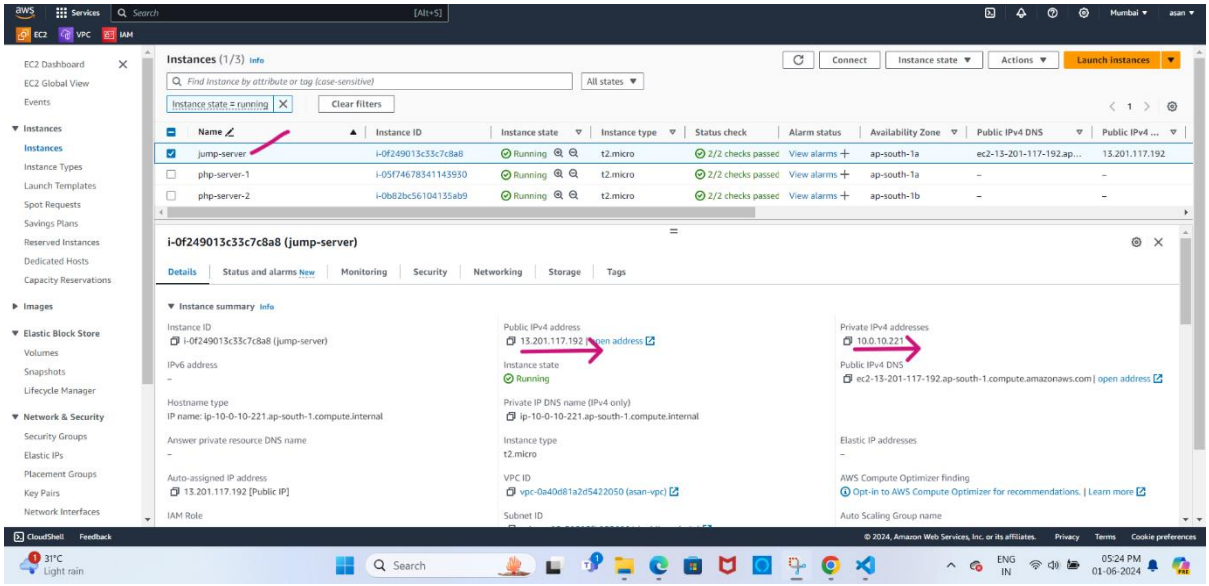
Step 5: Review and Launch

1. **Review Your Instance Launch**:

   - Review all the settings you have configured.
   - Click **Launch**.

2. **Launch Your Instance**:

   - Click **Launch Instances**.
   - Your instance will start launching.



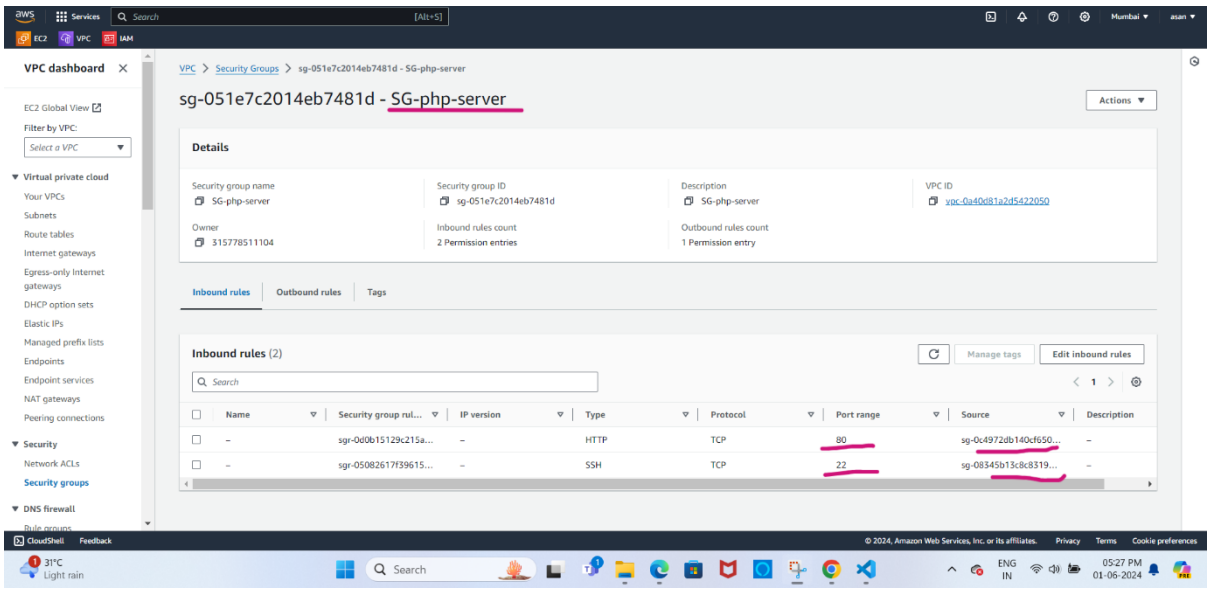
Step 11: Create Two EC2 Instance in Different Private Subnet (Php server)

Create security group for php server

Follow the previous same steps which are use to create a security group for jump server, but in inbound rules

Inbound Rules:

- Click on the **Inbound rules** tab and then click **Add rule**.
- **Type**: Select the type of traffic to allow (**ssh** to allow jump server, **http** for load balancer).
- **Protocol**: This field will be auto-filled based on the type selected or "**TCP**".
- **Port range**: Specify the port range (**22** for ssh , **80** for http).
- **Source**: Define the source IP range or security group (**security group of jump server** and **security group of load balancer** to allow traffic from specific node.).



Create a php server

Configure the Instance

1. **Choose an Amazon Machine Image (AMI)**:

   - Select an AMI based on your needs. For example, select **Amazon Linux 2 AMI** (for general purposes) or another AMI that suits your use case.
   - Click **Select**.

2. **Choose an Instance Type**:

   - Select the instance type. For general purposes, **t2.micro** (eligible for the free tier) is a good starting point.
   - Click **Next: Configure Instance Details**.

3. **Select an Existing Key Pair or Create a New Key Pair**:

   - Select an existing key pair if you have one.
   - If you don't have a key pair, create a new one. This key pair will be used to securely connect to your instance.
   - Download the key pair and store it securely. You won't be able to download it again.

4. **Configure Network Settings**:

   - Ensure that the instance is launched in the correct VPC and subnet.
   - Select the vpc and **private subnet** and make sure Auto-assign public IP is **disable**.
   - Click **Next: Add Storage**.

5. **Add Storage**:

- Specify the size and type of storage. The default of 8 GB General Purpose (SSD) is typically sufficient for testing.
- Click **Next: Add Tags**.

6. **Add Tags**:

- (Optional) Add tags to help you identify your instances. For example, you might add a tag with the key **Name** and value **MyEC2Ins1tance**.
- Click **Next: Configure Security Group**.

Step 4: Configure Security Group

2. **Add Security Group**:

- select an existing one that we create for jump server.
- Click **Review and Launch**.

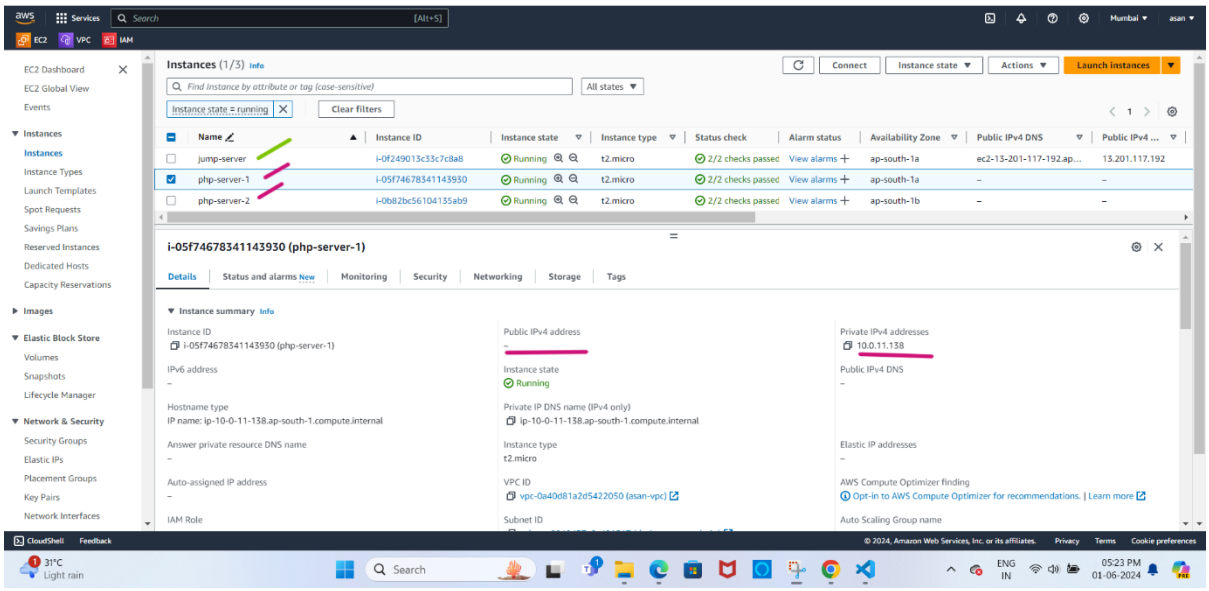
Step 5: Review and Launch

3. **Review Your Instance Launch**:

- Review all the settings you have configured.
- Click **Launch**.

4. **Launch Your Instance**:

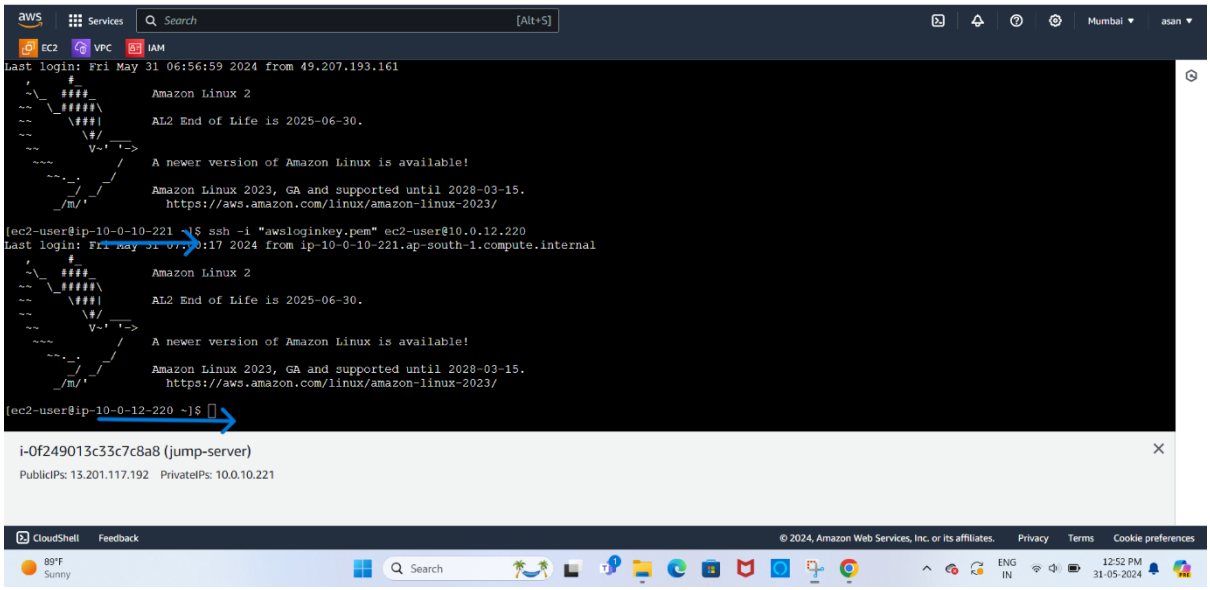
- Click **Launch Instances**.
- Your instance will start launching.



After creating the instance connect to the jump server and take an ssh action to the php server to install Apache and Php.

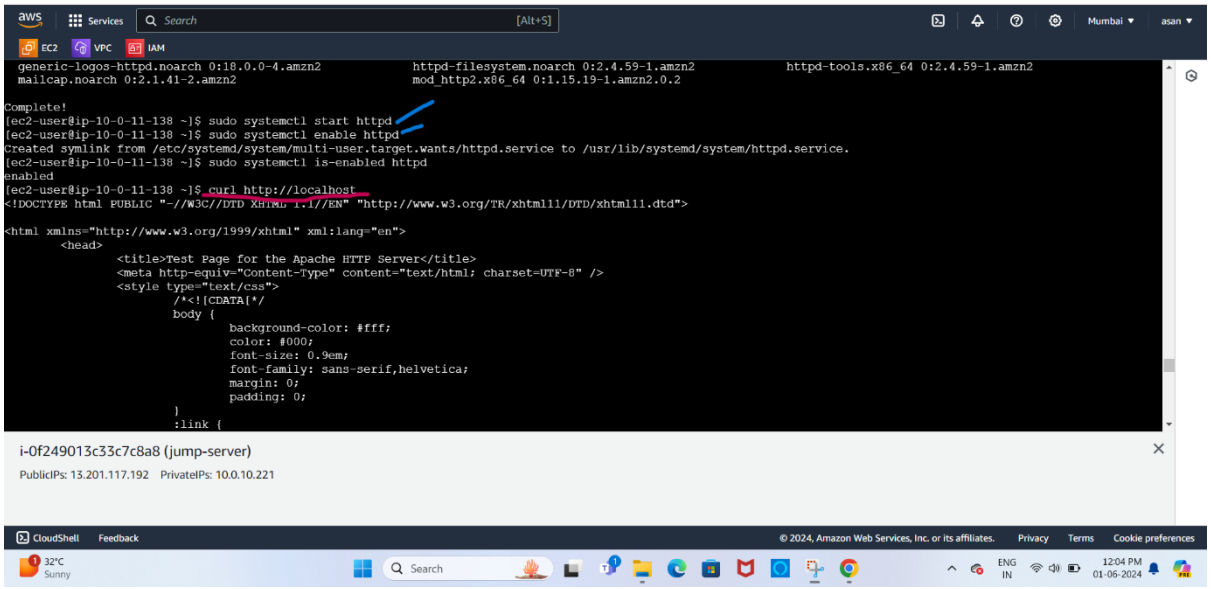
Step1: connect to jump server

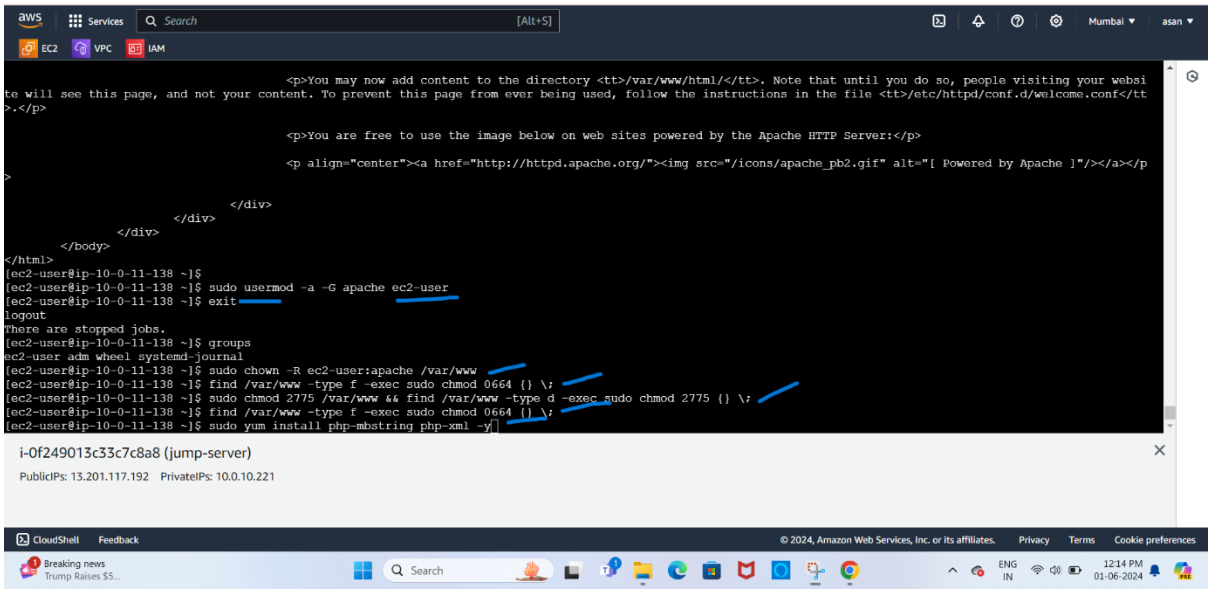
Step2: copy the private key and paste in jump server as the same name by the Linux comment **vim awsloginkey.pem**



Follow the comment to install the Apache and php configuration file
Tutorial: Install a LAMP server on AL2
https://docs.aws.amazon.com/linux/al2/ug/ec2-lamp-amazon-linux-2.html

```
yum update -y
amazon-linux-extras install -y lamp-mariadb10.2-php7.2 php7.2
yum install -y httpd
systemctl start httpd
systemctl enable httpd
usermod -a -G apache ec2-user
chown -R ec2-user:apache /var/www
chmod 2775 /var/www
find /var/www -type d -exec chmod 2775 {} \;
find /var/www -type f -exec chmod 0664 {} \;
echo '<?php phpinfo(); ?>' > /var/www/html/phpinfo.php
sudo yum install php-mbstring php-xml -y
sudo systemctl restart httpd
sudo systemctl restart php-fpm
cd /var/www/html
wget https://www.phpmyadmin.net/downloads/phpMyAdmin-latest-all-languages.tar.gz
mkdir phpMyAdmin && tar -xvzf phpMyAdmin-latest-all-languages.tar.gz -C phpMyAdmin --strip-components 1
rm phpMyAdmin-latest-all-languages.tar.gz
echo '<?php phpinfo(); ?>' > /var/www/html/phpinfo.php
cd phpMyAdmin
mv config.sample.inc.php config.inc.php
```





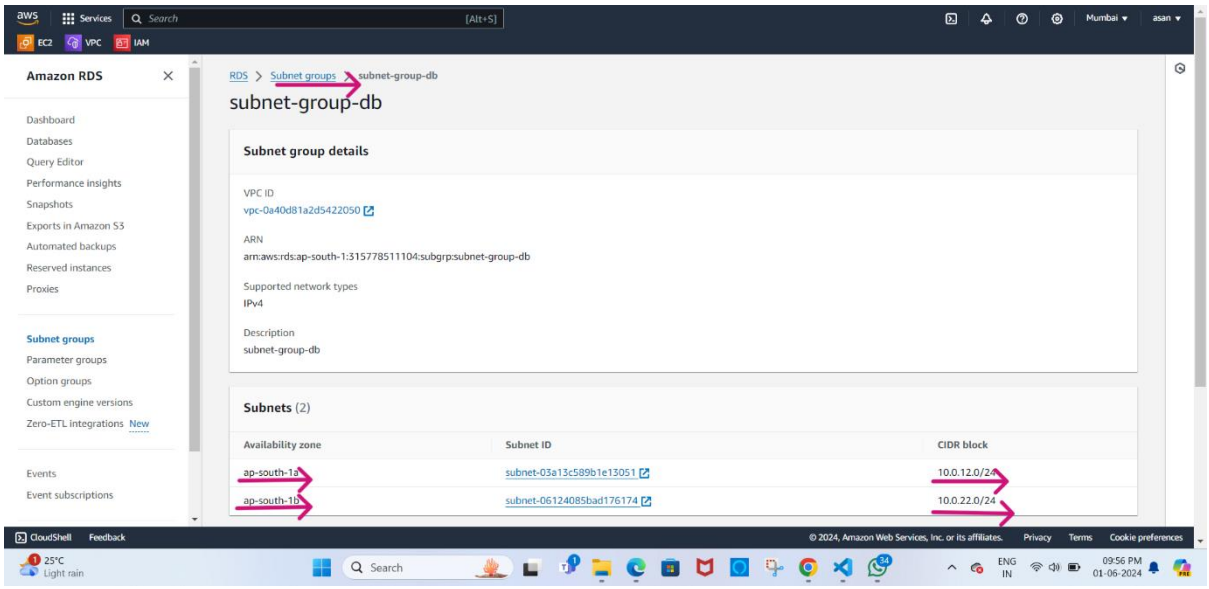
Step 12: Create Two MYSQL Database in Different Private Subnet (DB server)

Step 1: Create a Subnet Group

1. **Log in to AWS Management Console**: Navigate to the RDS service.
2. **Subnet Group Creation**:
    - Go to the **Subnet groups** section under **Subnet groups** in the left-hand menu.
    - Click on **Create DB Subnet Group**.
    - Fill in the required details:
        - **Name**: Give your subnet group a name.
        - **Description**: Provide a description.
        - **VPC**: Select the VPC where you want your RDS instances to be hosted.
    - Add subnets:

- Click on **Add all subnets** or manually select the subnets within different availability zones in your VPC.

- Click **Create**.

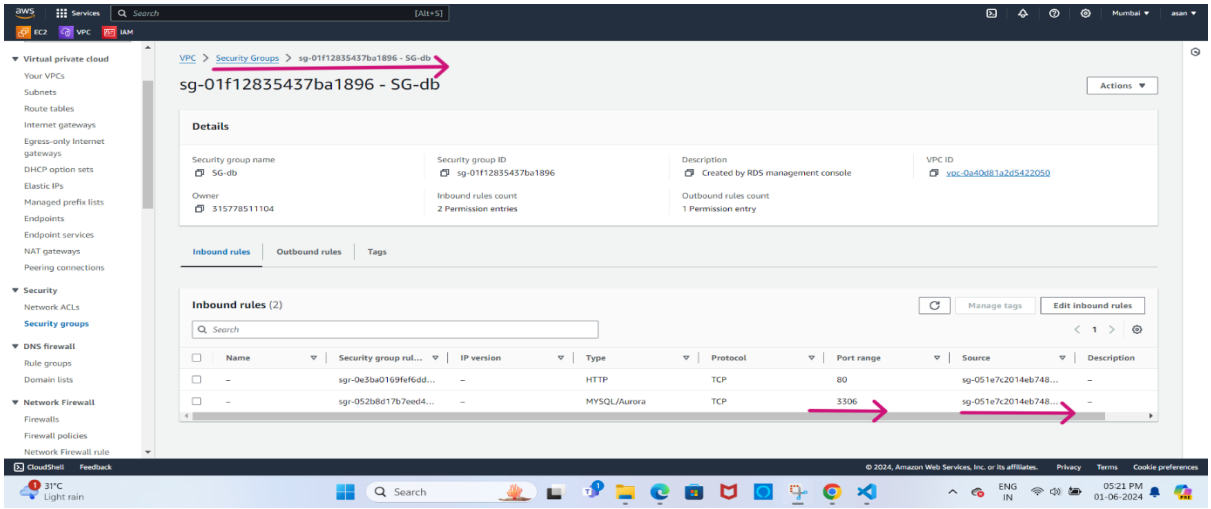


Create security group for Database server

Follow the previous same steps which are use to create a security group for php server, but in inbound rules

Inbound Rules:

- Click on the **Inbound rules** tab and then click **Add rule**.
- **Type**: Select the type of traffic to allow (**mysql/aurora** to allow php server, **http** for load balancer).
- **Protocol**: This field will be auto-filled based on the type selected or "**TCP**".
- **Port range**: Specify the port range (**3306** for mysql/aurora , **80** for http).
- **Source**: Define the source IP range or security group (**security group of php server** to allow traffic from php server only.).

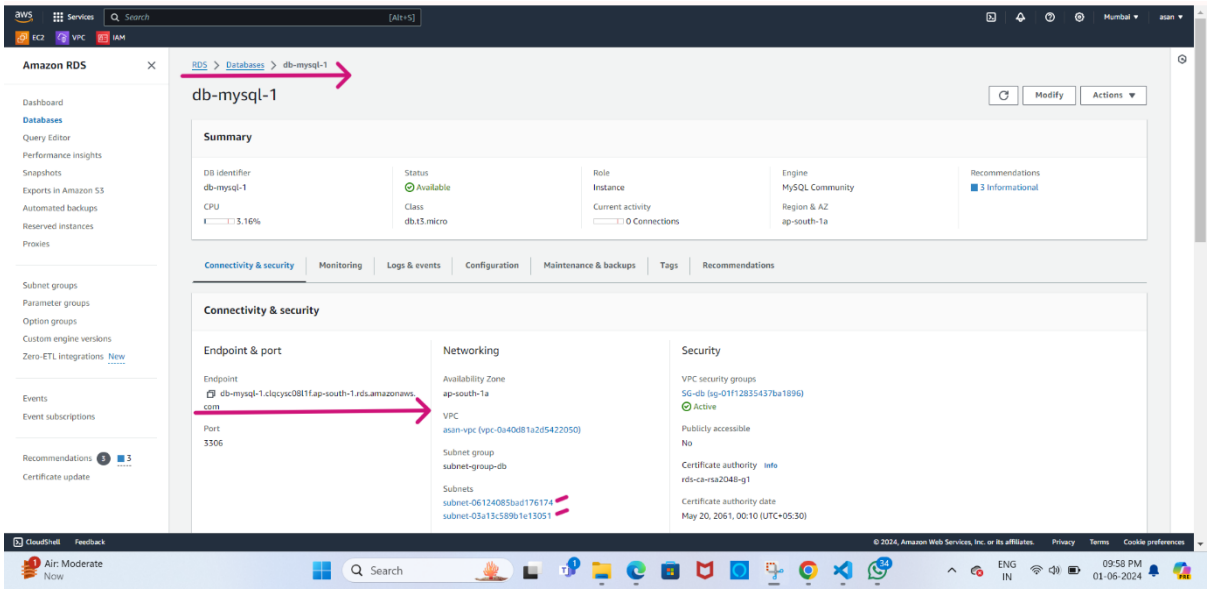


Step 2: Create an RDS MySQL Instance

1. **RDS Instance Creation**:
    - Navigate to the **Databases** section in the RDS service.
    - Click on **Create database**.
    - Select **Standard Create** for a more customizable setup.
    - Choose the **template** that best suits your use case (Production, Dev/Test, etc.) use free tier.
    - Choose **MySQL** as the database engine.
    - Select the **version** of MySQL as **MySQL 8.0.35.**
2. **Database Settings**:
    - **DB instance identifier**: Give your database instance a unique identifier.
    - **Master username**: Enter the master username.
    - **Master password**: Enter and confirm the master password.
3. **Instance Configuration**:
    - **DB instance class**: Choose the instance class based on your requirements (e.g., db.t3.micro for low cost, db.m5.large for more performance).
    - **Multi-AZ deployment**: Choose if you need high availability.
    - **Storage**: Select the storage type (SSD recommended) and allocate the required storage size.
4. **Connectivity**:
    - **VPC**: Select the VPC you chose for your subnet group.
    - **Subnet group**: Select the subnet group created in Step 1 (**private subnet db (SG-db)**).
    - **Public accessibility**: Choose if you want your database instance to be publicly accessible in our case **no**.
    - **VPC security groups**: Select the security group(s) of **SG-db** which only allow access from php server to control access to the instance.
    - **Availability zone**: You can leave this as "No preference" or choose a specific AZ.
5. **Database Options**:
    - **DB parameter group**: You can leave the default or choose an existing one.
    - **Option group**: Leave the default unless you need specific options.
    - **Backup**: Configure the backup retention period and other backup settings.
    - **Encryption**: Enable encryption if needed.
6. **Monitoring & Maintenance**:
    - **Enhanced monitoring**: Enable if you need detailed monitoring.

- **Auto minor version upgrade**: Enable if you want AWS to automatically upgrade minor versions.
- **Maintenance window**: Specify a maintenance window if required.

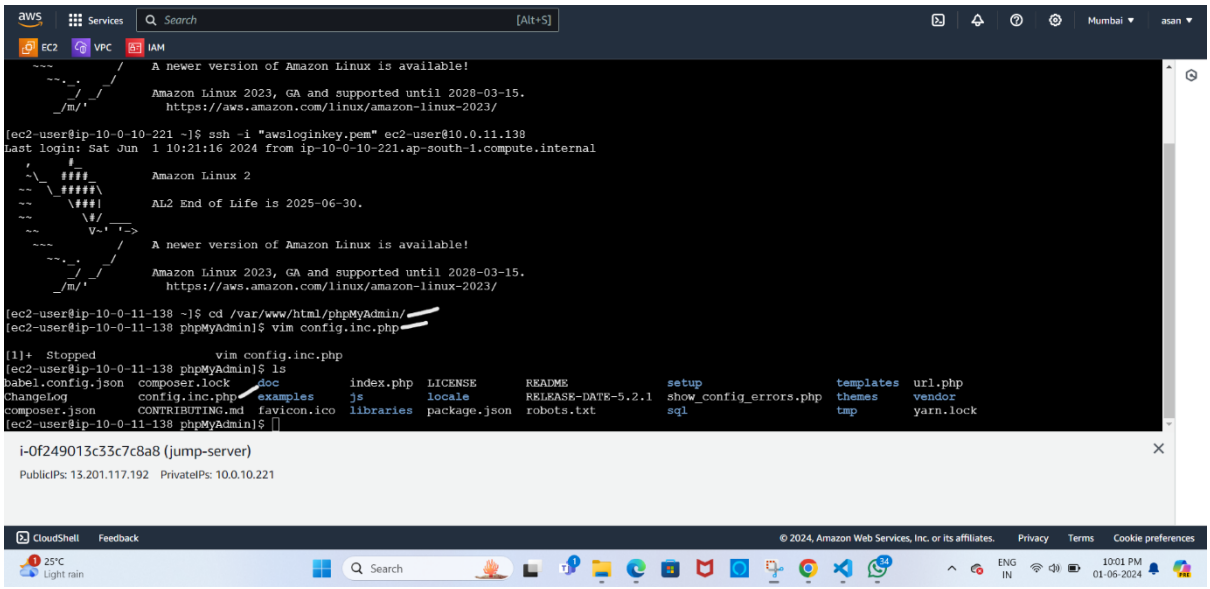
7. **Review and Create**:
   - Review all the settings.
   - Click **Create database**.



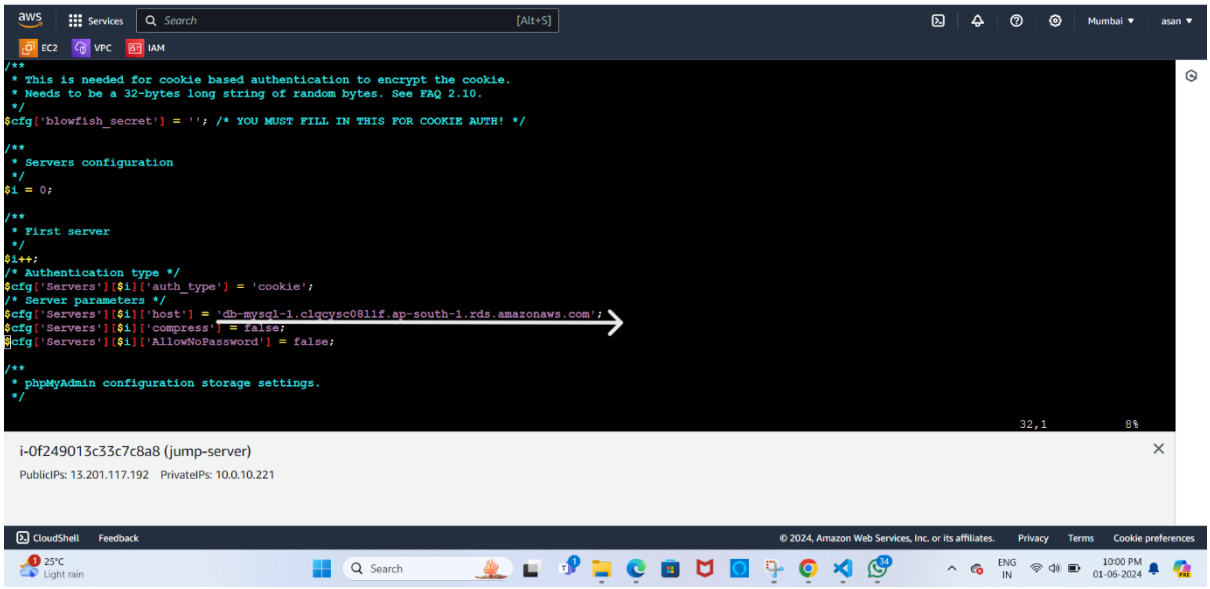
Attach Database server end point to php server

- Connect to php server and go to the repo of **cd /var/www/html/PhpMyAdmin**
- Then **ls**
- Remane the file config.sample.inc.php to config.inc.php by **mv config.sample.inc.php config.inc.php**
- Open config.sample.php by linux command **vim config.inc.php**
- Navigate to local host and replace that by mysql database endpoint

`



Add Database server end point to php server config file

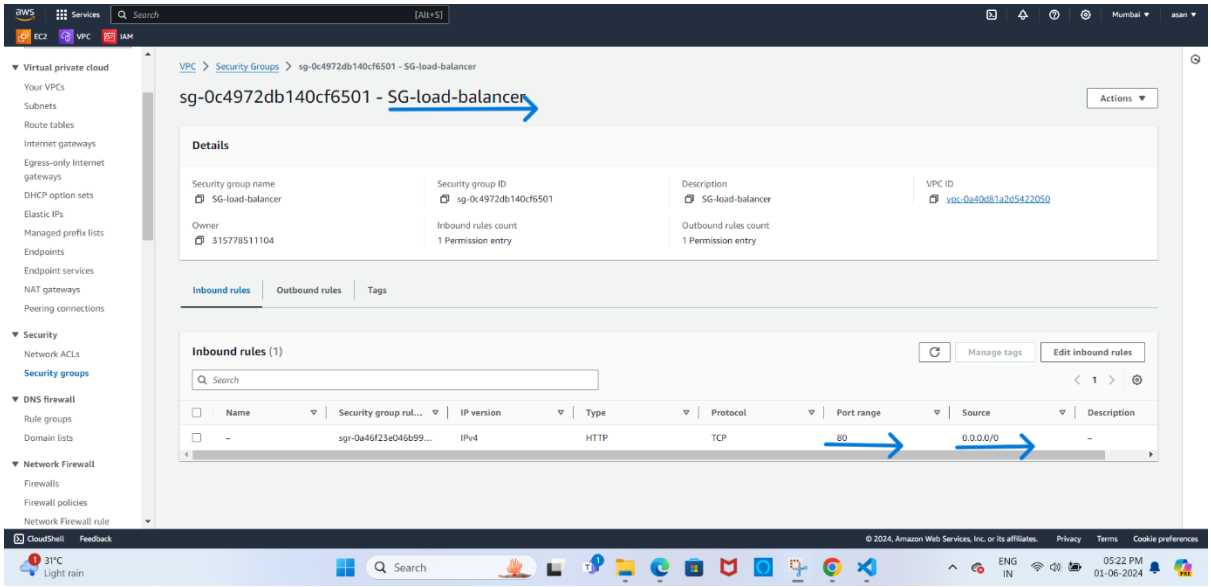


Step 1: create Load Balancer

Create security group for load balancer

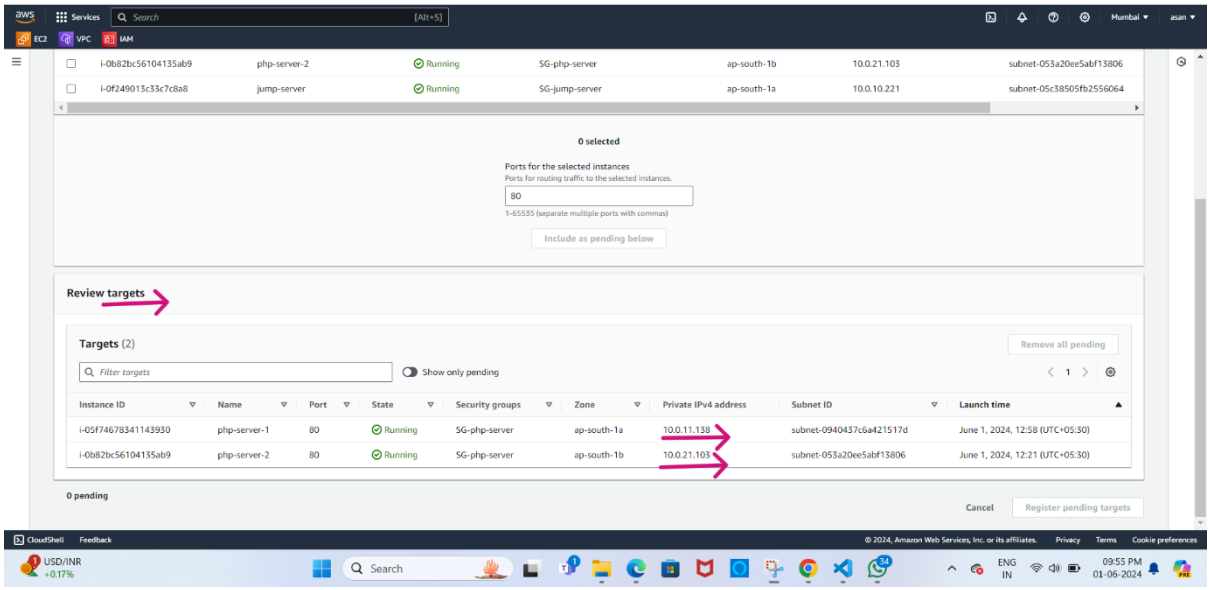
Inbound Rules:

- Click on the **Inbound rules** tab and then click **Add rule**.
- **Type**: Select the type of traffic to allow (**http** for outside user).
- **Protocol**: This field will be auto-filled based on the type selected or "**TCP**".
- **Port range**: Specify the port range (**80** for http).
- **Source**: Define the source IP range or security group (**0.0.0.0/0** to allow traffic to all.).



Step 1: Create a Target Group

1. **Navigate to Target Groups**:
    - In the AWS Management Console, go to the EC2 Dashboard.
    - On the left-hand side, under "Load Balancing," click on "Target Groups."
2. **Create Target Group**:
    - Click on the "Create target group" button.
    - Select the target type (Instances, IP addresses, or Lambda function). For as, select "Instances."
    - Configure the following:
        - **Name**: Provide a name for your target group.
        - **Protocol**: Choose the protocol (HTTP).
        - **Port**: Specify the port (80 for HTTP).
        - **VPC**: Select the VPC where your instances are running.
    - Click "Next."
3. **Health Checks**:
    - Configure health check settings. The defaults are usually fine, but you can adjust them based on your application's needs.
    - Click "Next."
4. **Register Targets**:
    - Select the two php instances in different subnet to include in this target group.
    - Click "Include as pending below."
    - Once your instances are added, click "Create target group."



Step 2: Create a Load Balancer

1. **Navigate to Load Balancers**:

- In the AWS Management Console, go to the EC2 Dashboard.
- On the left-hand side, under "Load Balancing," click on "Load Balancers."

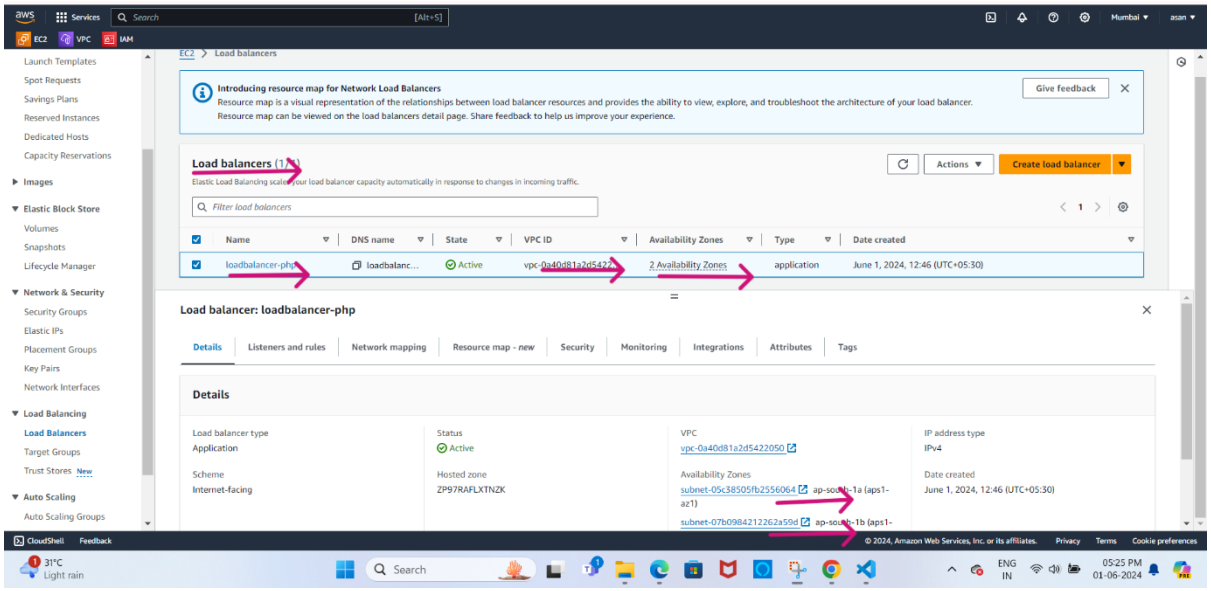
2. **Create Load Balancer**:
   - Click on the "Create load balancer" button.
   - Choose the type of load balancer you want to create (Application Load Balancer, Network Load Balancer, or Gateway Load Balancer). For as, select "Application Load Balancer."
   - Click "Create."
3. **Configure Load Balancer**:
   - **Name**: Provide a name for your load balancer (e.g., "loadbalancer-php").
   - **Scheme**: Choose "internet-facing" for public access or "internal" for internal access.
   - **IP address type**: Select IPv4 or dualstack (for IPv4 and IPv6).
   - **Listeners**: Ensure there is a listener for HTTP.
   - **Availability Zones**: Select the VPC and the availability zones where your instances are located.
   - Click "Next: Configure Security Settings."
4. **Configure Security Settings**:
   - If using HTTPS, configure an SSL certificate. For HTTP, you can skip this step.
   - Click "Next: Configure Security Groups."
5. **Configure Security Groups**:
   - Select an existing security group or create a new one to control traffic to your load balancer.
   - Click "Next: Configure Routing."
6. **Configure Routing**:
   - **Target group**: Select the target group you created earlier.
   - **Name**: Provide a name for your listener rule.
   - **Path Pattern**: Specify the path pattern to route traffic ("/") after checking the load balancer perfectly then we change it to the php config path ("/PhpMyAdmin")
   - Click "Next: Register Targets."
7. **Register Targets**:
   - Review the target instances. You should see the instances you added to the target group.
   - Click "Next: Review."
8. **Review and Create**:
   - Review all the settings you have configured.
   - Click "Create" to create your load balancer.

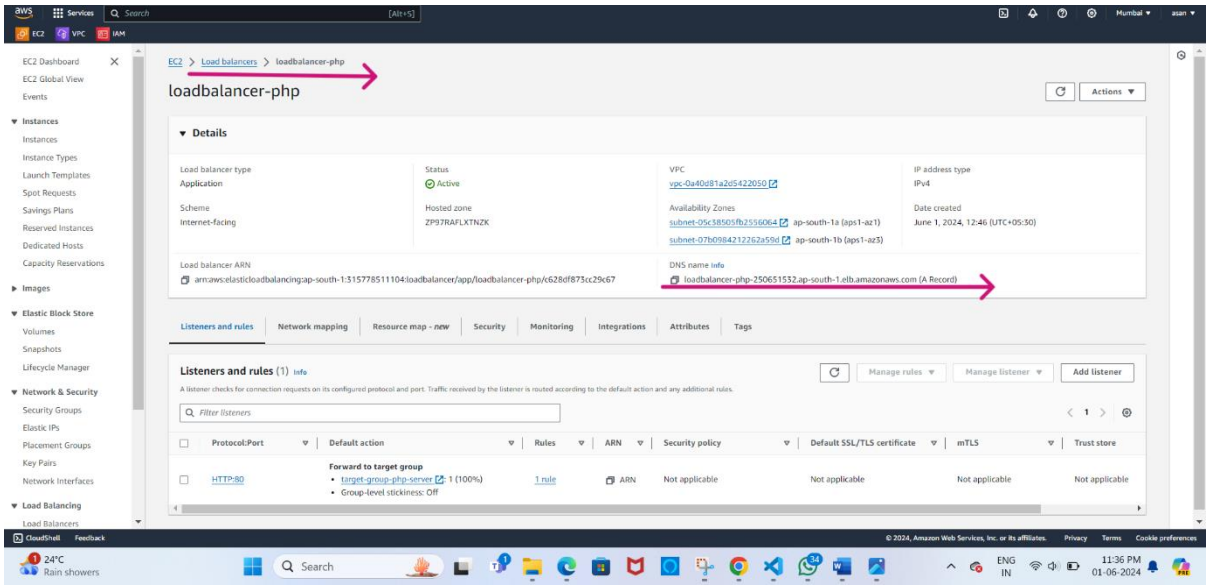
Step 3: Verify the Load Balancer

1. Go back to the "Load Balancers" section in the EC2 Dashboard.
2. Select your new load balancer.
3. Check its status to ensure it is "Active."

Step 4: Test the Load Balancer

1. Access the DNS name of the load balancer from a web browser.
2. Verify that the traffic is correctly routed to your instances.





Step 1: Obtain the Load Balancer DNS Name

1. **Navigate to Load Balancers**:
    - Log in to your AWS Management Console.
    - Go to the EC2 Dashboard.
    - Under "Load Balancing" on the left sidebar, click on "Load Balancers."

- Select your load balancer (e.g., loadbalancer-php).

2. **Copy the DNS Name**:
   - In the load balancer details section, copy the DNS name. It should look something like this: loadbalancer-php-55061532.ap-south-1.elb.amazonaws.com.

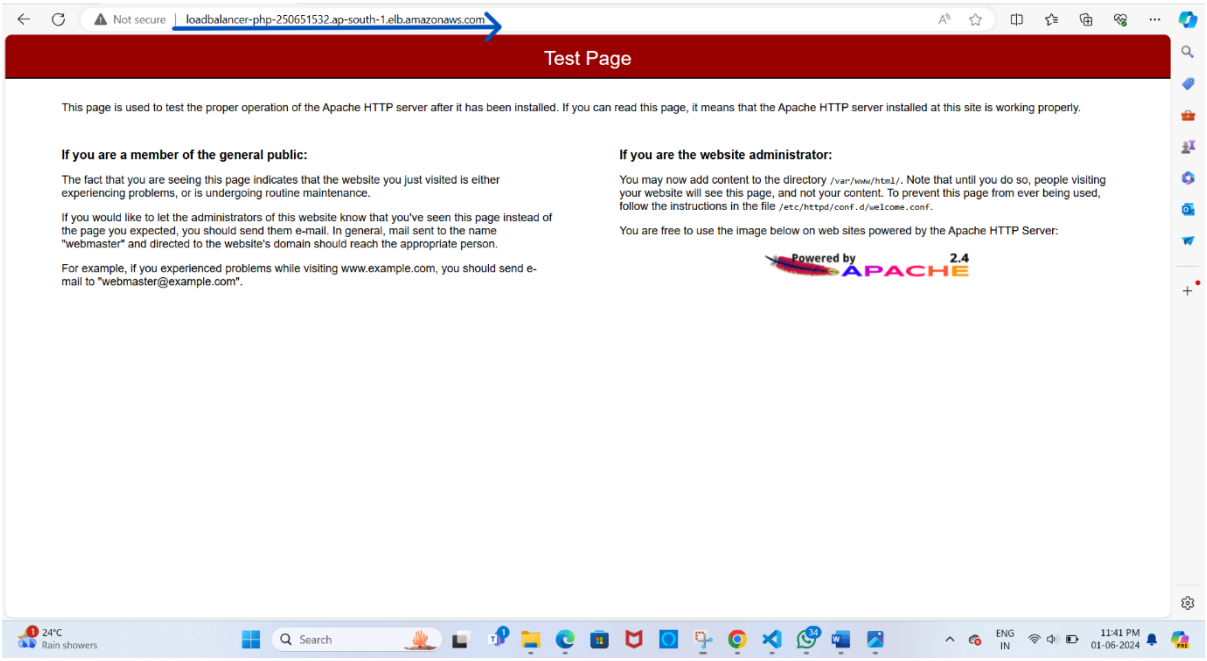
Step 2: Test Connectivity

1. **Access the DNS Name**:
   - Open a web browser.
   - Enter the Load Balancer DNS name in the address bar and hit Enter.
2. **Verify the Response**:
   - You should see the response from one of your instances.
   - To verify routing, refresh the page multiple times. This should show responses from different instances if the load balancer is correctly distributing traffic.

Step 3: Verify Instances

Different Responses:

- If you have configured each instance to respond differently (e.g., instance 1 returns "Instance 1" and instance 2 returns "Instance 2"), you should see these different responses as you refresh the browser.
- Yes, it route to php server 1



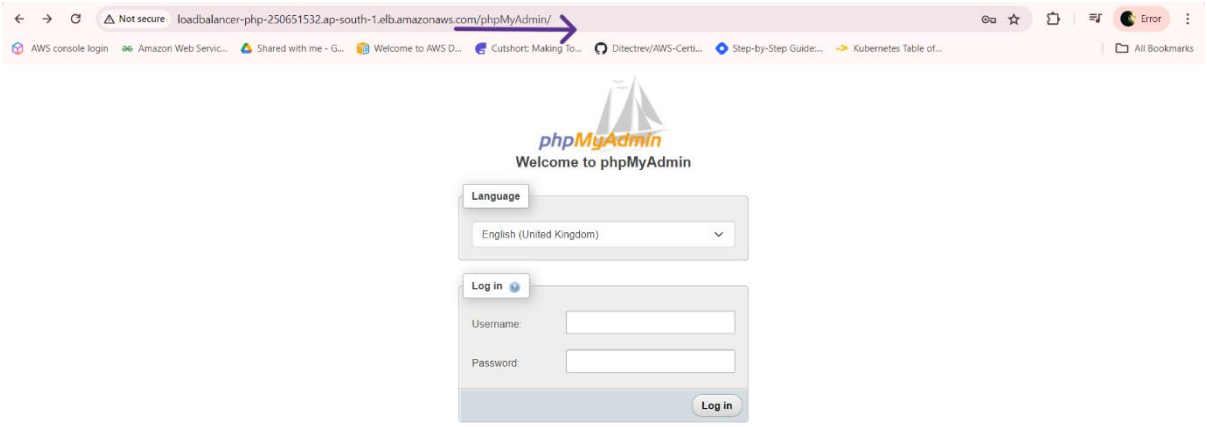
Refresh the web page. yes, it goes to Php server 2

Now change a health check path in target groups to redirect a traffic to PhpMyAdmin page, by changing a path.

- Navigate to target group select the target
- Select the health checks and make change in path /PhpMyAdmin
- Then go to Attributes and enable stickiness because phpMyAdmin is statefull application so we stick a single instances in back ground to stick with the services.

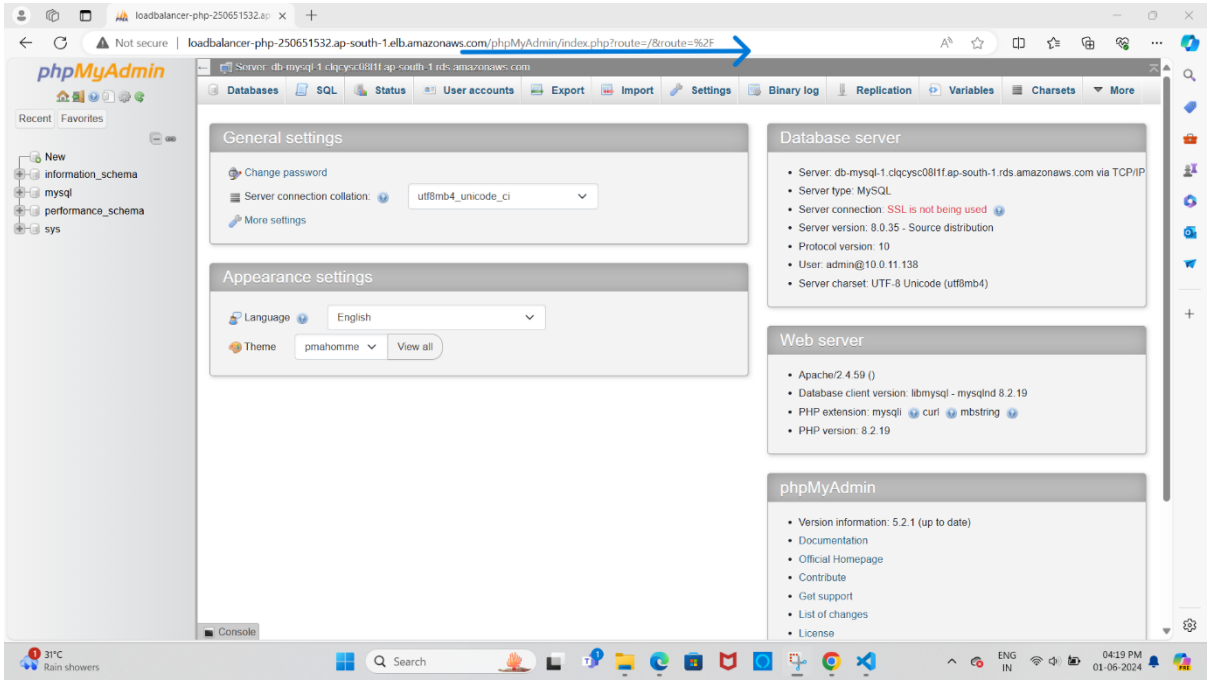
Copy the DNS Name and phpMyAdmin in http header

In the load balancer details section, copy the DNS name and make change last of header ***loadbalancer-php-55061532.ap-south-1.elb.amazonaws.com/PhpMyAdmin***



Provide a username and password to login to the database

Now, we successfully login to the aws rds mysql database application



By following these steps, we ensure your AWS load balancer is effectively managing traffic to your target instances, maintaining high availability, and optimizing performance.

Conclusion

Successfully deploying a 3-tier web application on AWS involves careful planning and configuration of various AWS services. By following this guide, we set up a VPC with public and private subnets, configured security groups to ensure secure communication, deployed instances for the web, application, and database layers, and set up a load balancer to distribute traffic efficiently.

Key Benefits of This Deployment:

1. **Scalability**: Easily add more instances to handle increased load.
2. **Security**: Use of private subnets and security groups ensures secure communication.
3. **High Availability**: Load balancer distributes traffic and ensures redundancy.
4. **Ease of Management**: Jump server allows secure access to instances in private subnets.

This deployment on AWS provides a scalable, secure, and highly available infrastructure for your web applications. By leveraging AWS's robust services, you can ensure your application is well-prepared to handle varying loads and provide a seamless user experience.