Task on VPC   
1) Create VPC with 2 private and 2 public subnets.

**1. Create a VPC**

* Go to the VPC Dashboard on AWS.
* Click Create VPC.
* Enter a name (e.g., MyVPC).
* Set an IPv4 CIDR block (e.g., 10.0.0.0/16).
* Click Create.

**2. Create Subnets**

You need 4 subnets in total: 2 public, 2 private. You should select different availability zones (AZs) for redundancy.

**Example:**

* 10.0.1.0/24 (Public Subnet 1, AZ1)
* 10.0.2.0/24 (Private Subnet 1, AZ1)
* 10.0.3.0/24 (Public Subnet 2, AZ2)
* 10.0.4.0/24 (Private Subnet 2, AZ2)

To create each subnet:

* Click Subnets in the VPC console.
* Click Create subnet.
* Select your VPC.
* For each subnet, specify:
  + A unique name
  + The availability zone
  + The CIDR block

**3. Create an Internet Gateway (IGW) for Public Subnets**

* Go to Internet Gateways > Create Internet Gateway.
* Attach the IGW to your VPC.

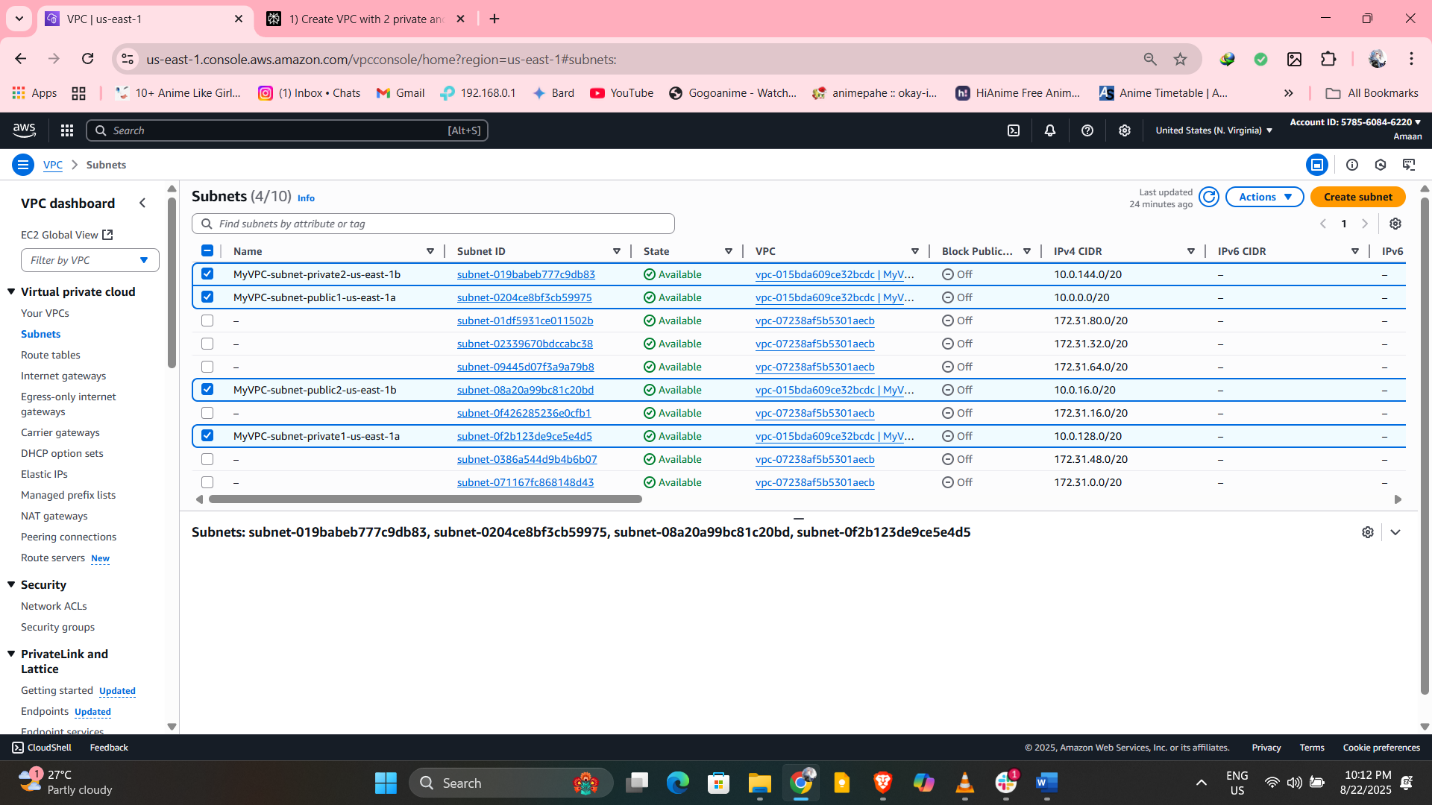
**4. Modify Routing Tables**

**Public Subnets:**

* Go to Route Tables.
* Select (or create) a route table for public subnets.
* Edit routes:
  + Add a route: Destination 0.0.0.0/0 → target is your IGW.
* Associate this route table with your two public subnets.

**Private Subnets:**

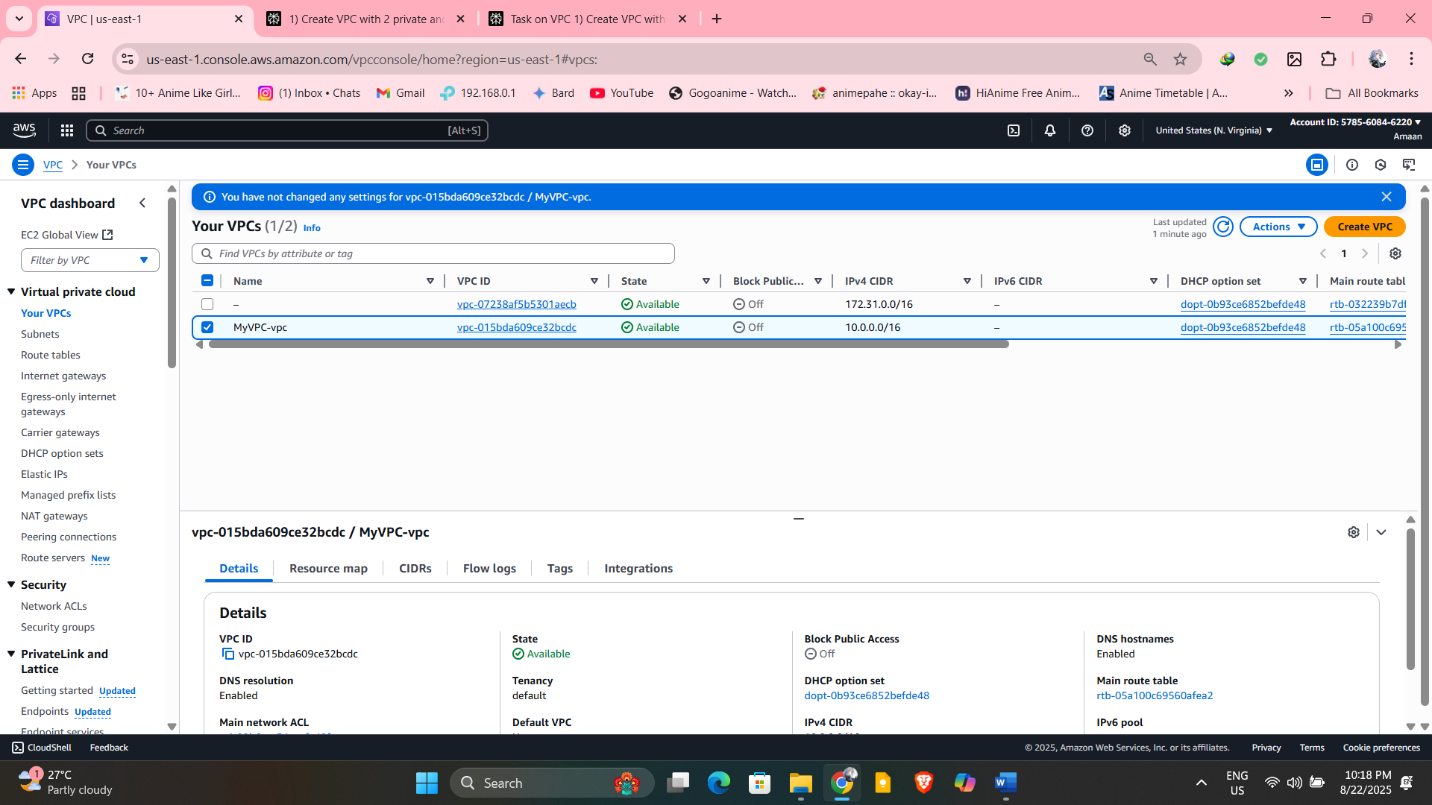
* Create (or use the default) a separate route table for private subnets.
* Do NOT add a 0.0.0.0/0 route to an IGW.
* Associate this route table with your two private subnets.



2) Enable DNS Hostname in VPC

**How to Enable DNS Hostnames in an AWS VPC**

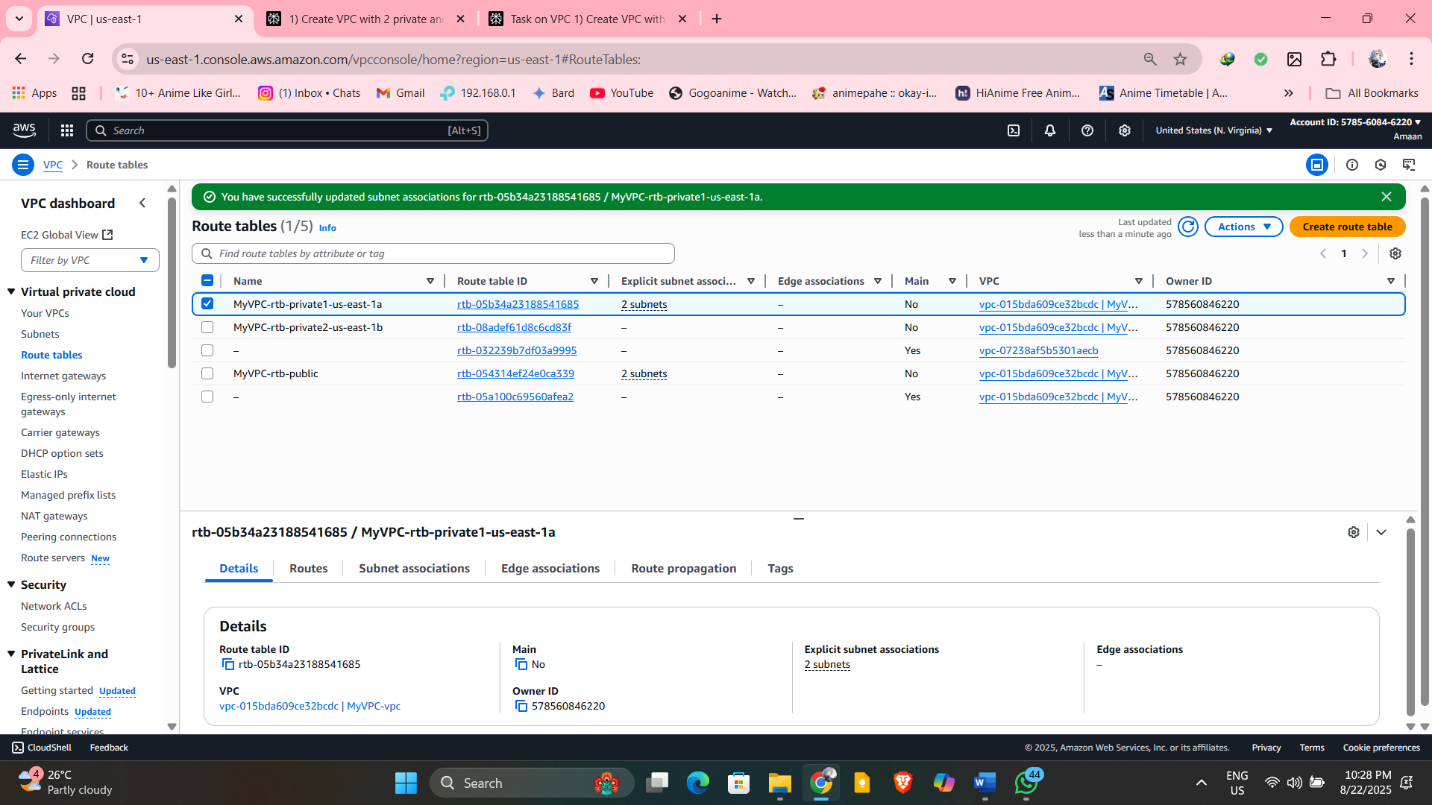
1. Open the VPC Dashboard
   * Go to the AWS Management Console.
   * Select “VPC” under Networking.
2. Select Your VPC
   * In the sidebar, click Your VPCs.
   * Find and select the VPC you want to enable DNS hostnames for.
3. Edit DNS Hostnames Setting
   * With your VPC selected, click the Actions button and choose Edit VPC settings (sometimes labeled as “Edit DNS Resolution and DNS Hostnames”).
   * Set DNS Hostnames to Enabled.
4. Save Changes
   * Click Save or Update to apply the changes.

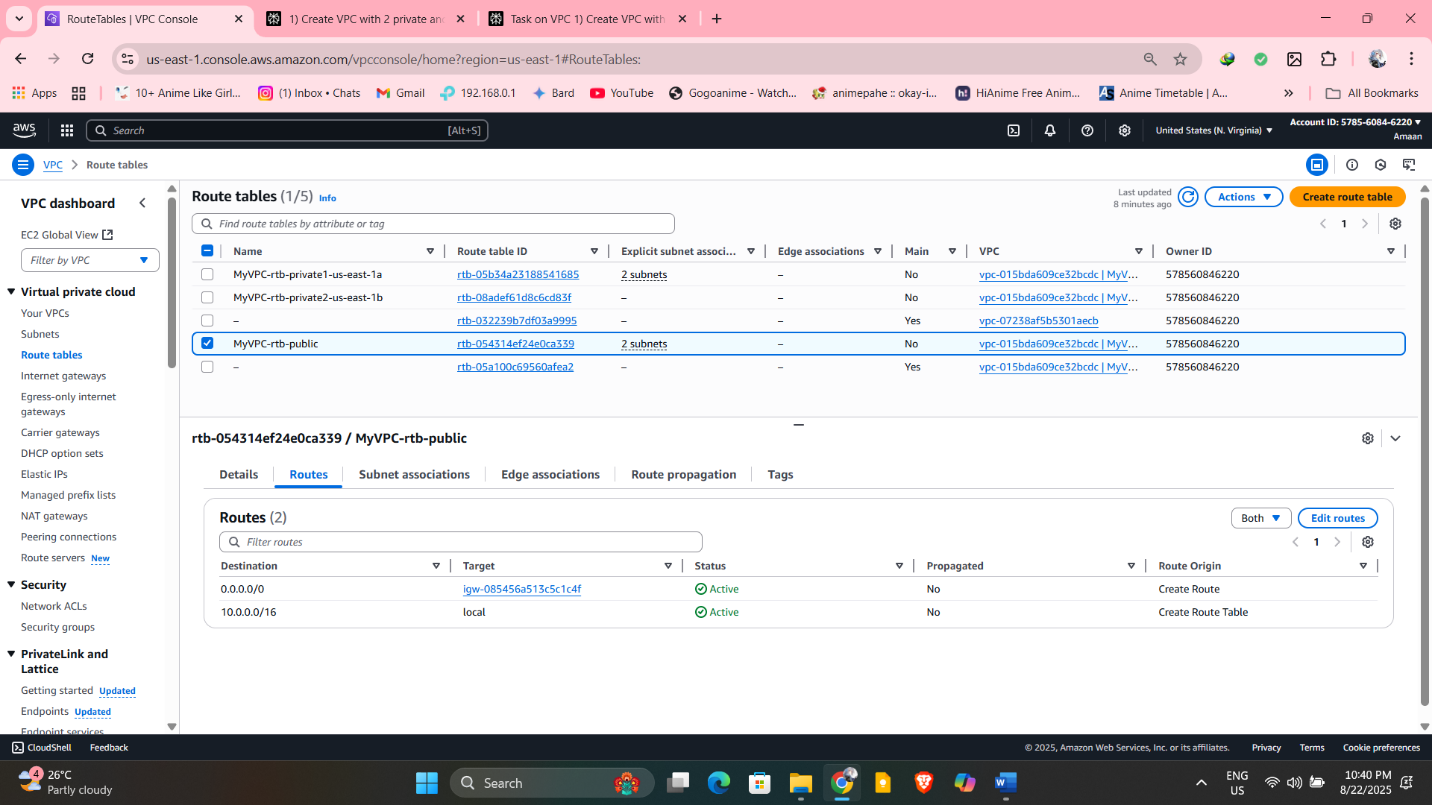


3) Enable Auto Assign Public ip in 2 public subnets

**Steps to Enable Auto-Assign Public IP for Subnets**

1. Go to the VPC Dashboard
   * Open the AWS Management Console.
   * Navigate to the VPC section.
2. Select “Subnets”
   * In the sidebar, click Subnets.
   * Locate your two public subnets (e.g., MyVPC-subnet-public1-us-east-1a and MyVPC-subnet-public2-us-east-1b).
3. Edit Subnet Settings
   * Select one public subnet.
   * Click Actions > Edit subnet settings.
   * Find the option Auto-assign IP settings.
   * Check Enable auto-assign public IPv4 address.
   * Click Save.
4. Repeat for the Second Public Subnet
   * Select your second public subnet and repeat the above steps.

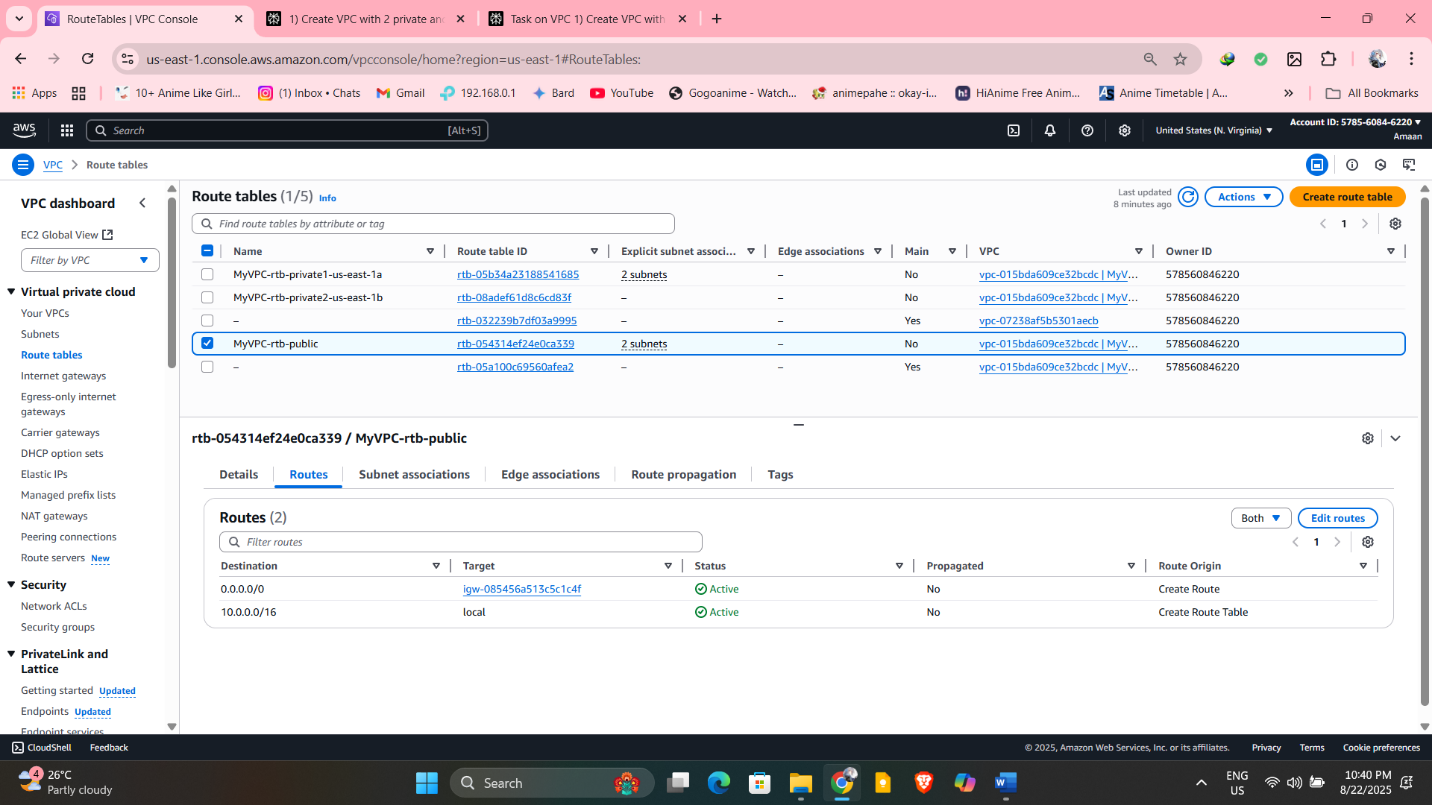




4) Add 2 private subnets in private route table

**Steps to Associate Private Subnets with Private Route Table**

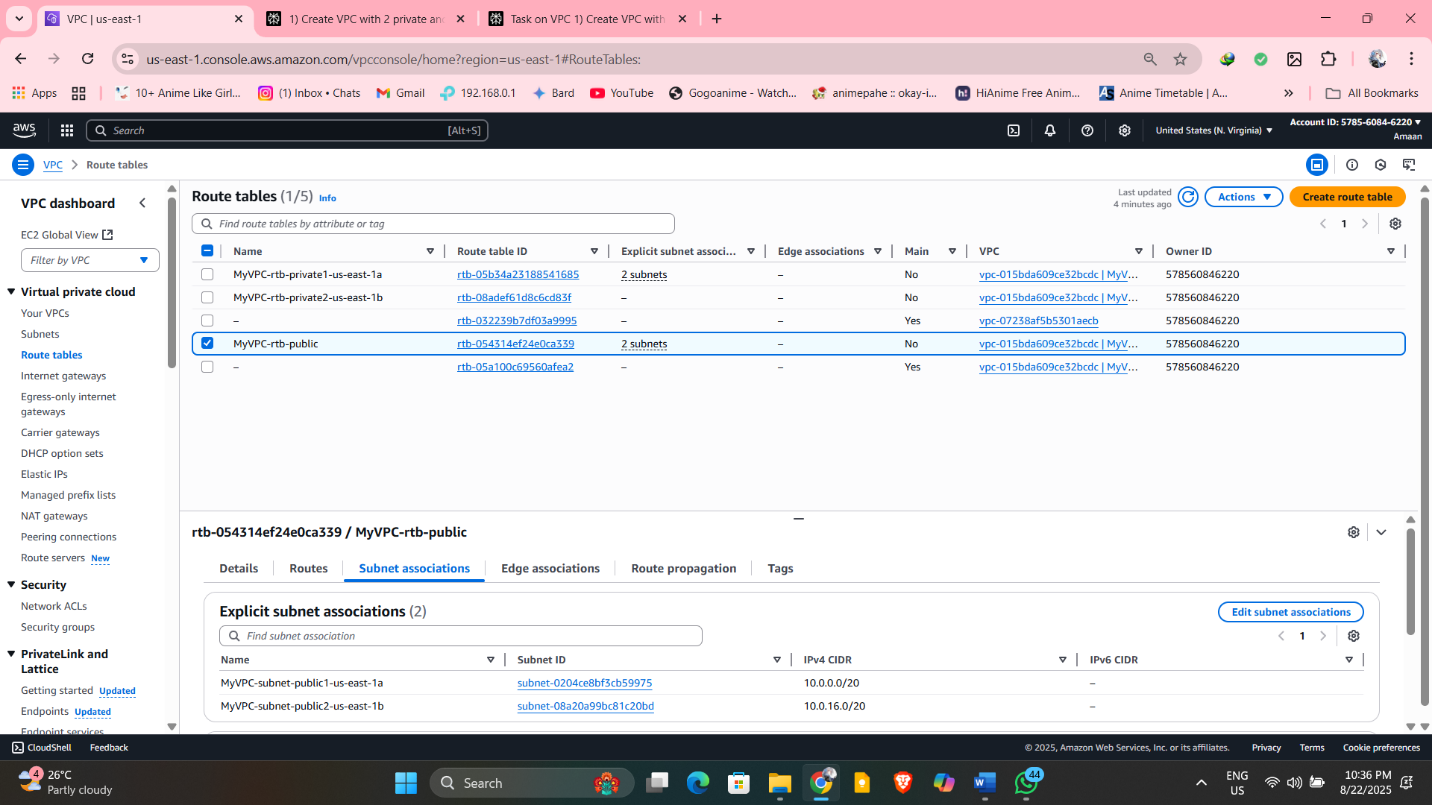
1. Open the VPC Dashboard
   * Go to the AWS Management Console.
   * Navigate to VPC.
2. Go to Route Tables
   * In the sidebar, click Route Tables.
   * Identify the route table you want to use as your private route table (one without an internet gateway route).
3. Associate Private Subnets
   * Select the private route table.
   * Click the Subnet Associations tab (or choose Edit subnet associations).
   * Select the two private subnets (e.g., MyVPC-subnet-private1-us-east-1a and MyVPC-subnet-private2-us-east-1b).
   * Click Save or Associate.



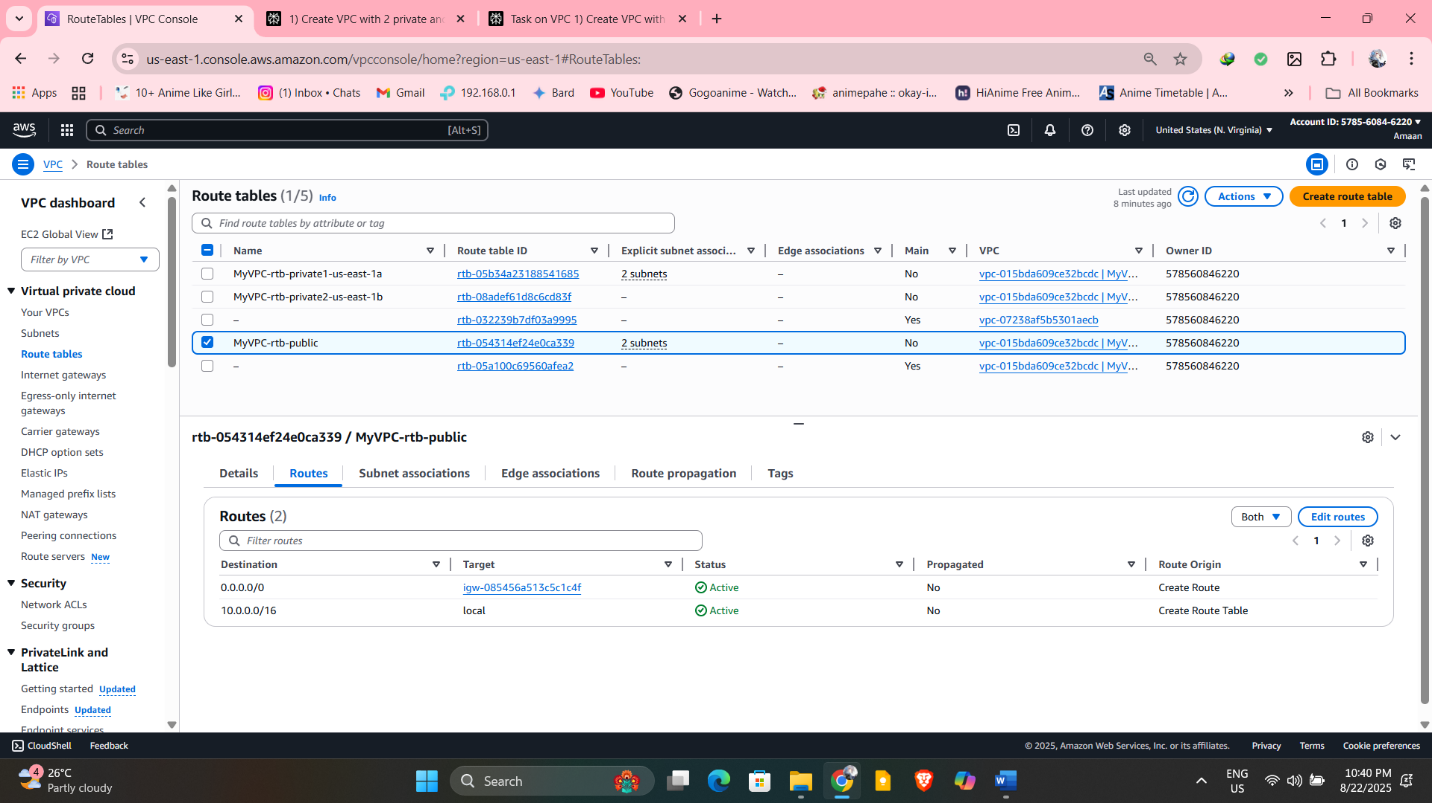
5) Add 2 public subnets in public route table

**Steps to Associate Public Subnets with Public Route Table**

1. Open the VPC Dashboard
   * Go to the AWS Management Console.
   * Navigate to VPC.
2. Go to Route Tables
   * Click on Route Tables in the sidebar.
   * Find the route table designated as your public route table (usually named something like "MyVPC-rtb-public").
3. Associate Public Subnets
   * Select the public route table.
   * Click the Subnet Associations tab or Edit subnet associations.
   * Select your two public subnets (e.g., MyVPC-subnet-public1-us-east-1a and MyVPC-subnet-public2-us-east-1b).
   * Click Save or Associate.



6) Public route table will have the routes to internet and local



7) Create Ec2 in public subnet with t2micro and install php

**Step 1: Launch an EC2 Instance in Public Subnet**

1. Go to the EC2 Dashboard in AWS Console.
2. Click Launch Instances.
3. Configure the instance:
   * Name: Give your instance a name.
   * AMI: Choose an Amazon Linux 2 AMI (or preferred Linux AMI).
   * Instance Type: Select t2.micro.
4. Configure network settings:
   * Select your VPC.
   * For Subnet: Choose one of your public subnets.
   * Enable Auto-assign Public IP (should be enabled by default in public subnets if you set it up earlier).
5. Configure other options as needed and proceed.
6. Add storage.
7. Add tags if needed.
8. Configure Security Group:
   * Add rules to allow SSH (port 22) from your IP.
   * Add HTTP (port 80) to allow web traffic.
9. Review and launch, then create or use an existing key pair for SSH access.

**Step 2: Connect to EC2 Instance**

* Use an SSH client to connect:

text

ssh -i your-key.pem ec2-user@<Public-IP>

**Step 3: Install PHP on EC2 Instance**

Once connected via SSH, run these commands to install PHP:

**For Amazon Linux 2:**

text

sudo yum update -y

sudo amazon-linux-extras enable php8.0

sudo yum clean metadata

sudo yum install -y php php-cli php-mbstring

php -v

**Or for Amazon Linux 1:**

text

sudo yum update -y

sudo yum install -y php php-cli php-mbstring

php -v

1. **Create the info.php file in the correct directory:**

First, make sure the web server directory exists and then create the PHP info file:

bash

sudo mkdir -p /var/www/html

echo "<?php phpinfo(); ?>" | sudo tee /var/www/html/info.php

1. **Ensure your web server (e.g., Apache) is installed and running:**

bash

sudo yum install -y httpd

sudo systemctl start httpd

sudo systemctl enable httpd

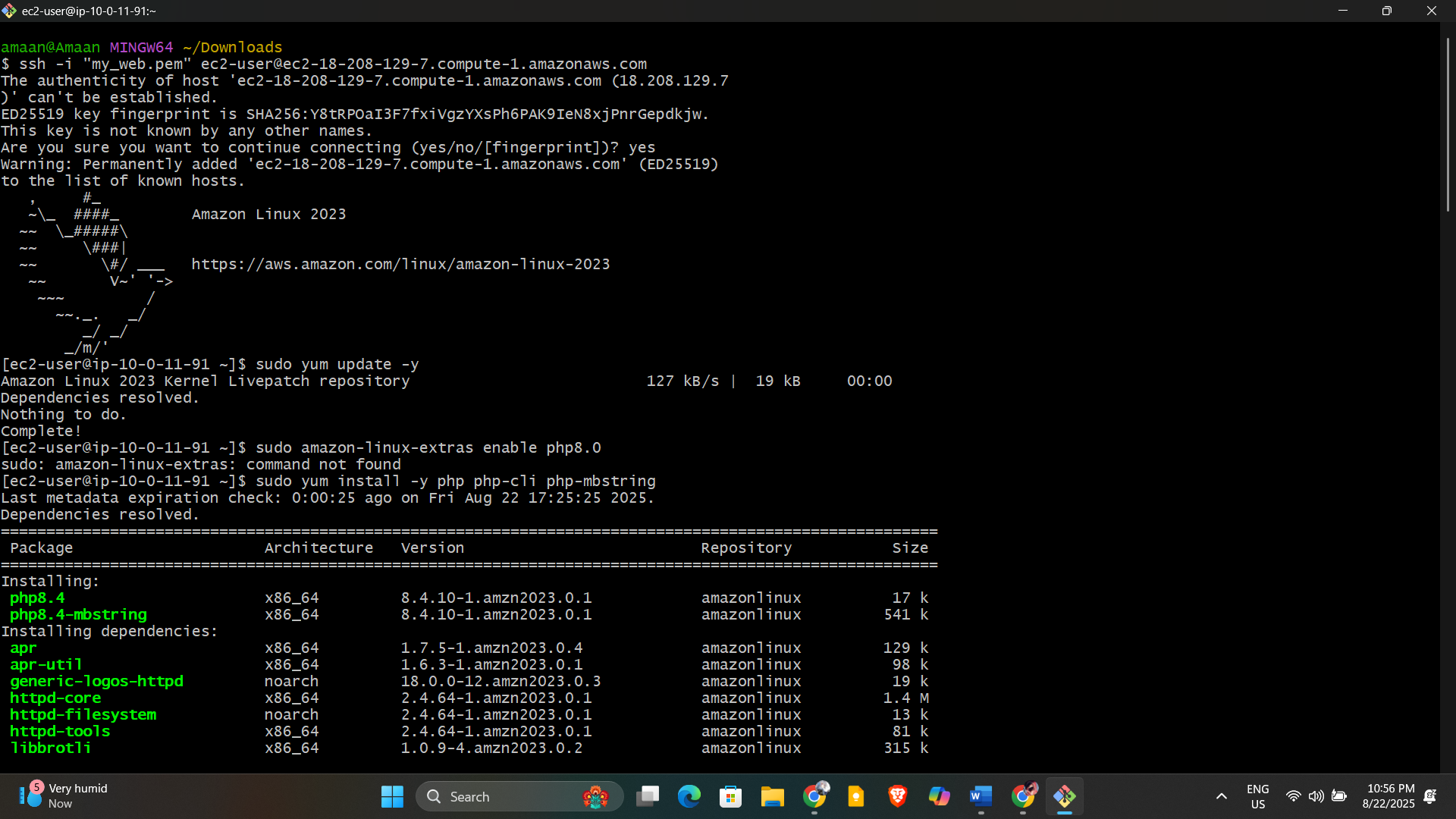
1. **Check the status of the web server:**

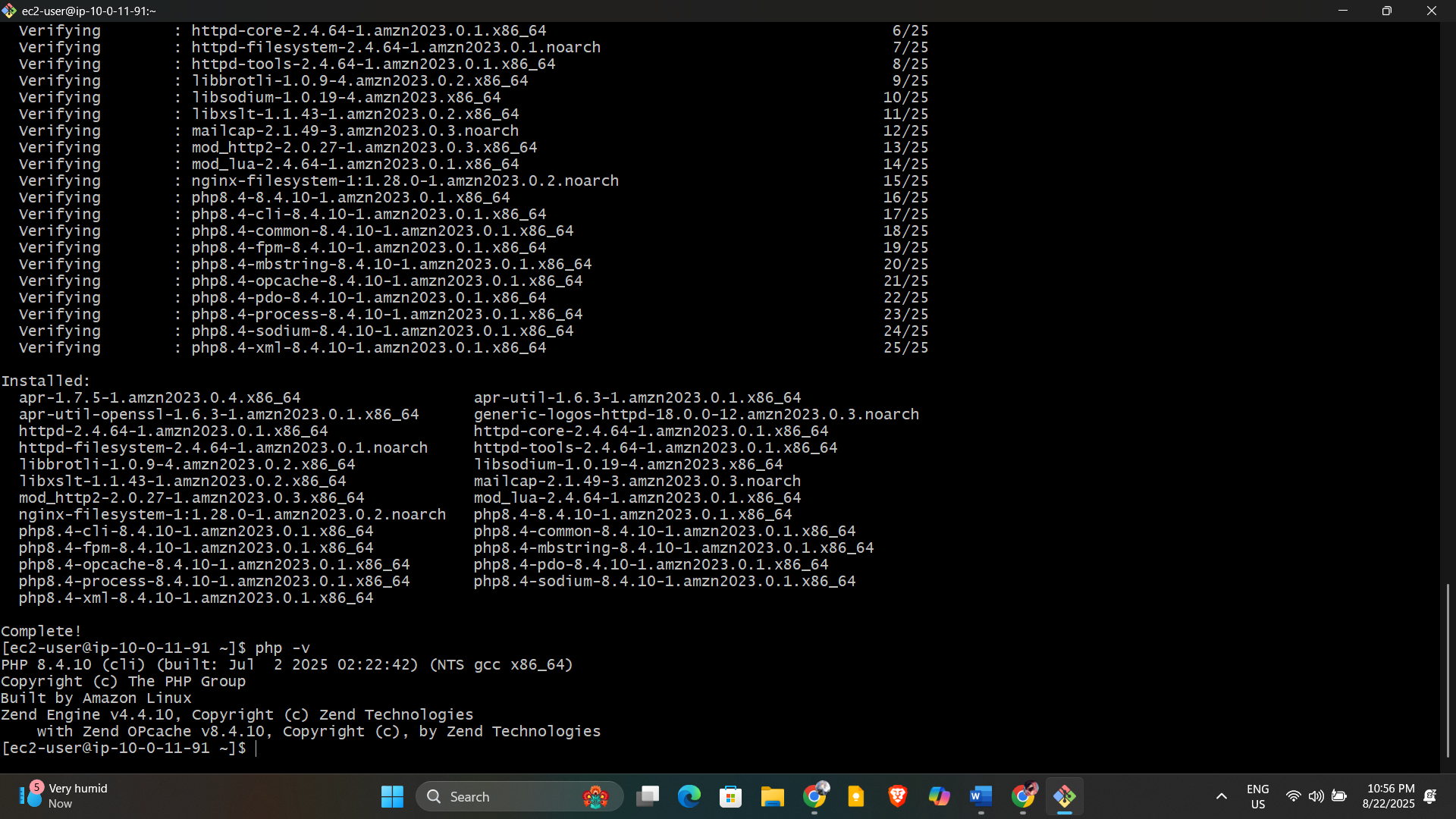
bash

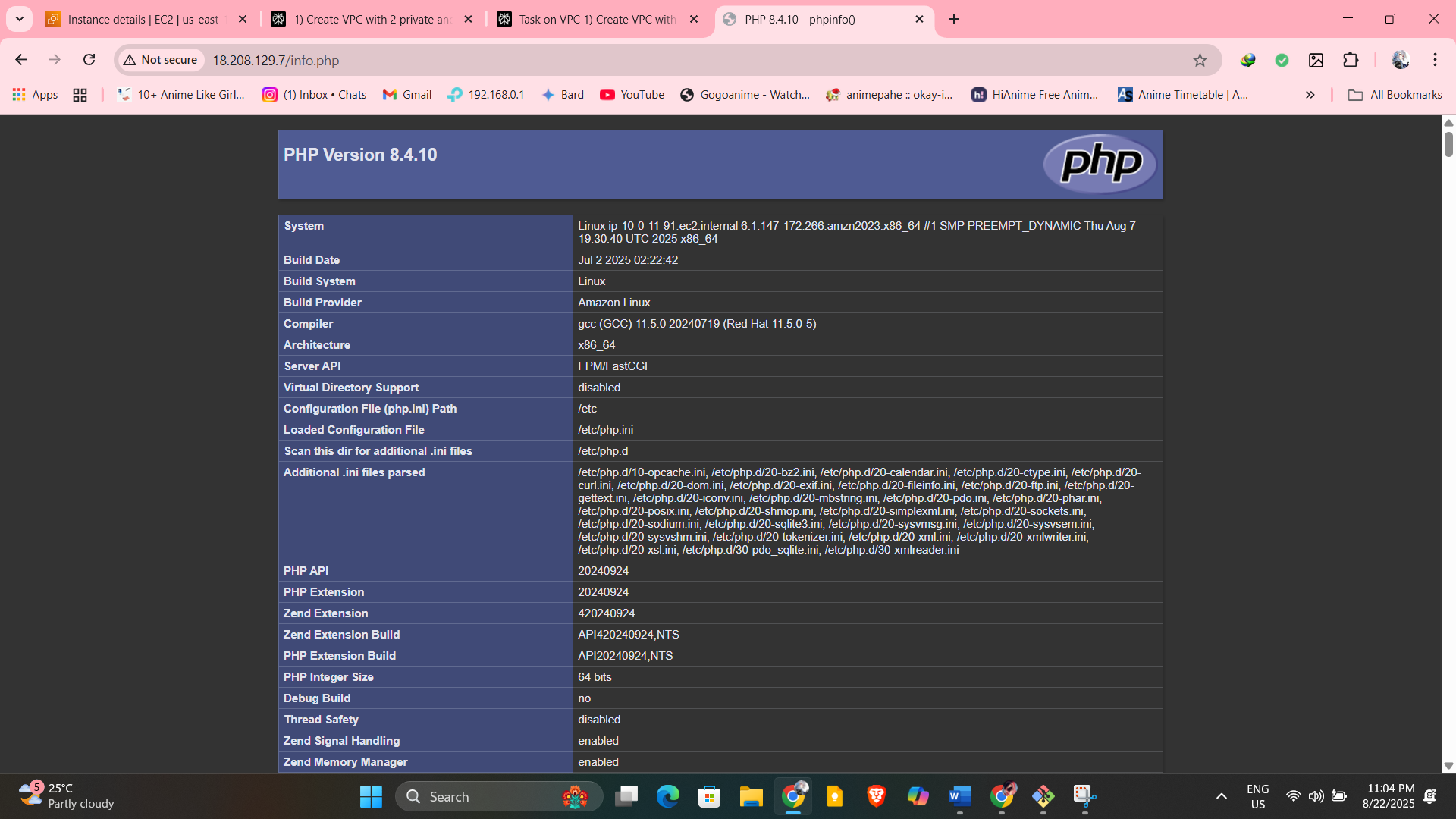
sudo systemctl status httpd

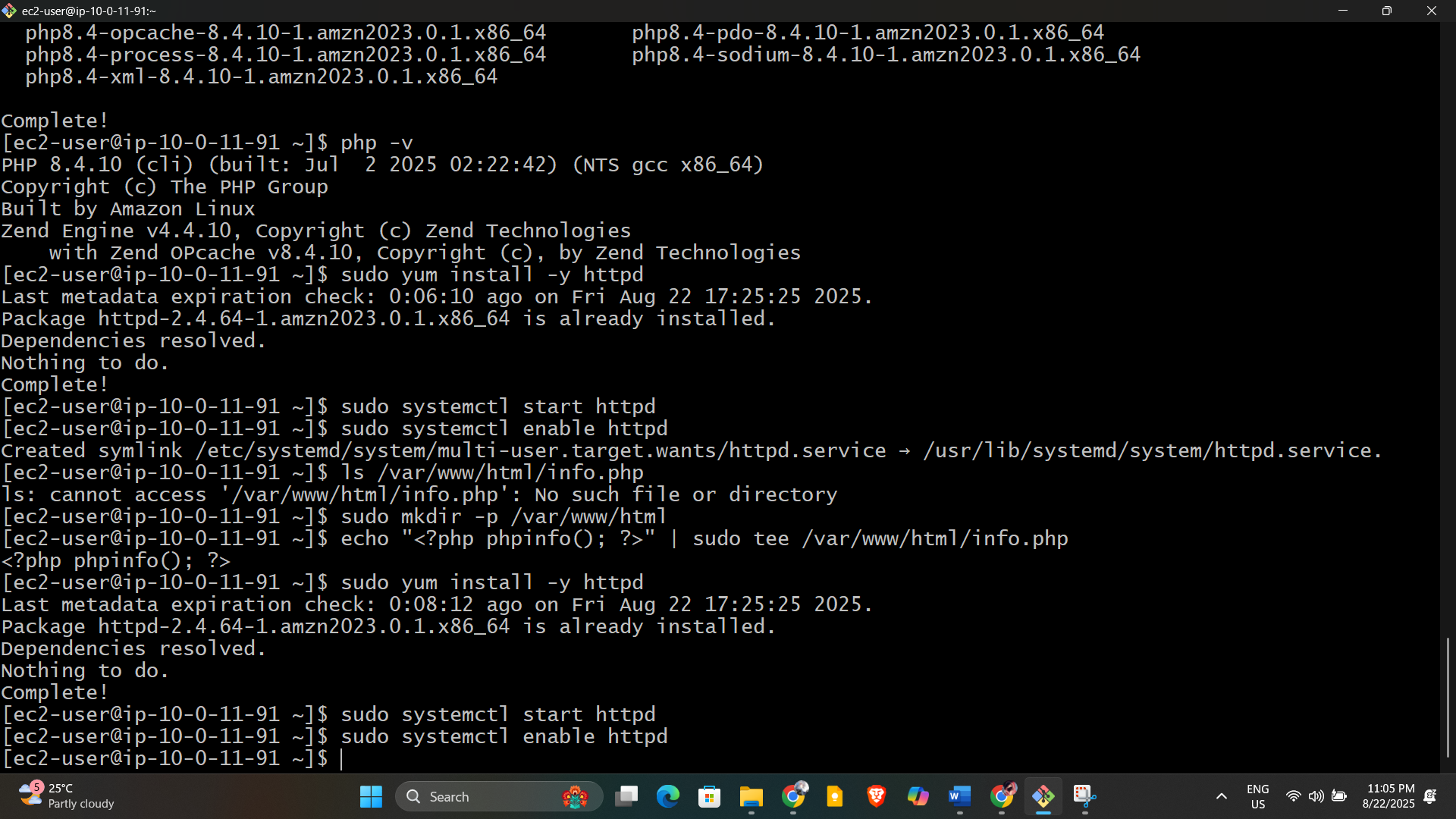
1. **Verify the security group of your EC2 instance allows HTTP traffic on port 80.**

After these steps, try accessing the URL again:  
http://<your-instance-public-ip>/info.php









8) Configure Nat gateway in public subnet and connect to private Instance

**Step 1: Create a NAT Gateway in a Public Subnet**

1. Go to the VPC Dashboard in AWS Console.
2. In the sidebar, click NAT Gateways.
3. Click Create NAT Gateway.
4. Select a public subnet from your VPC.
5. Allocate an Elastic IP address (create a new one if needed).
6. Click Create NAT Gateway.
7. Wait for the NAT Gateway to become available.

**Step 2: Update the Private Route Table to Use the NAT Gateway**

1. In the VPC console, go to Route Tables.
2. Select the route table associated with your private subnets.
3. Go to the Routes tab and click Edit routes.
4. Add a new route with:
   * Destination: 0.0.0.0/0
   * Target: Your newly created NAT Gateway
5. Save the route.

**Step 3: Verify Internet Access from Private Instance**

1. **Connect to your EC2 instance in the private subnet:**
   * Since the instance is in a private subnet (no public IP), you cannot SSH directly.
   * Use a bastion host or Session Manager (AWS Systems Manager) to connect.
     + If you have a bastion host in a public subnet:
       - SSH into the bastion host first:

text

ssh -i your-bastion-key.pem ec2-user@<Bastion-Public-IP>

* + - * From the bastion host, SSH into your private instance using its private IP:

text

ssh -i your-private-key.pem ec2-user@<Private-Instance-Private-IP>

* + - If you use Session Manager:
      * Open the AWS Systems Manager Console.
      * Go to Session Manager > Start session with your private instance.

1. **Test internet connectivity from the private instance:**
   * Once connected, try downloading updates or pinging a public server:

text

ping -c 3 google.com

or

text

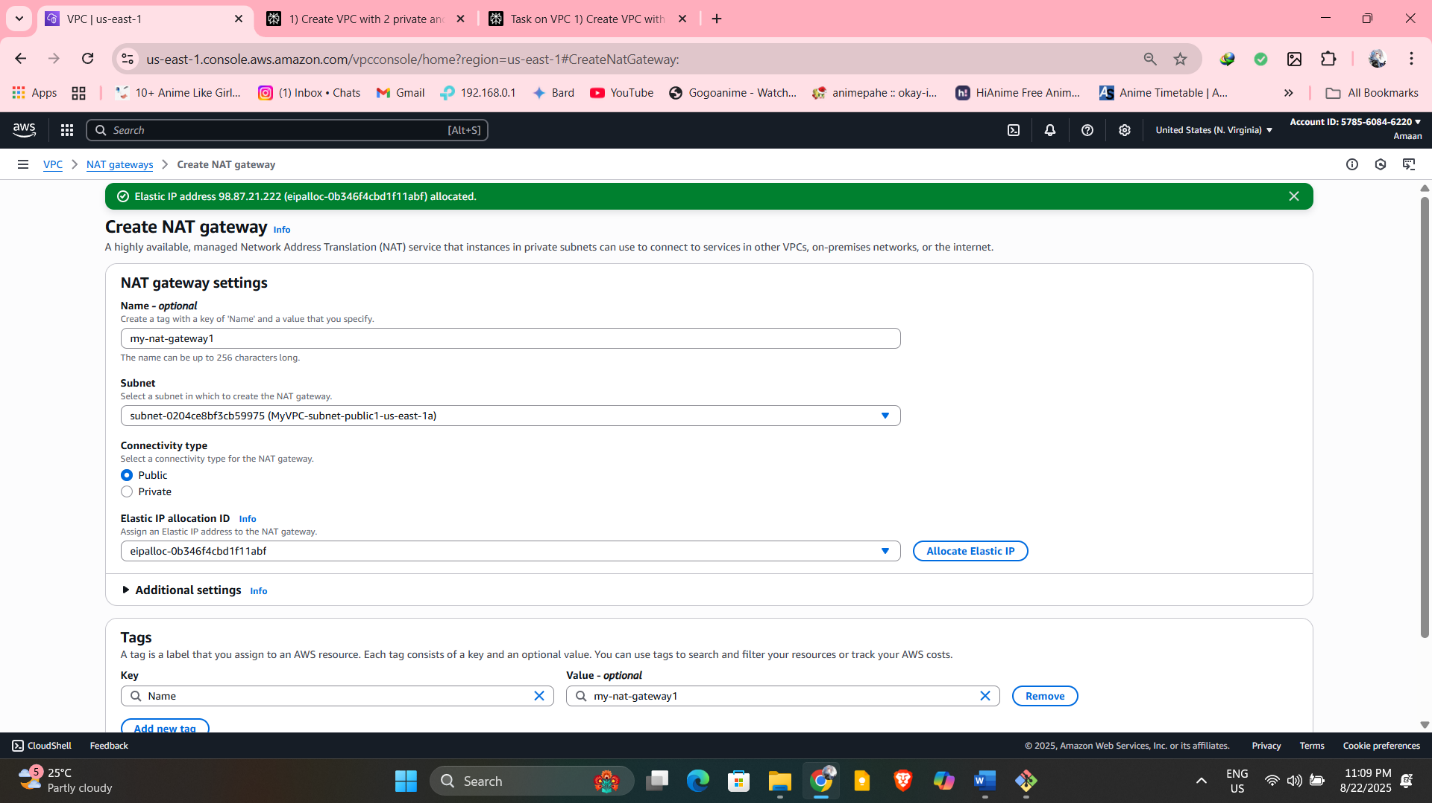
curl http://example.com

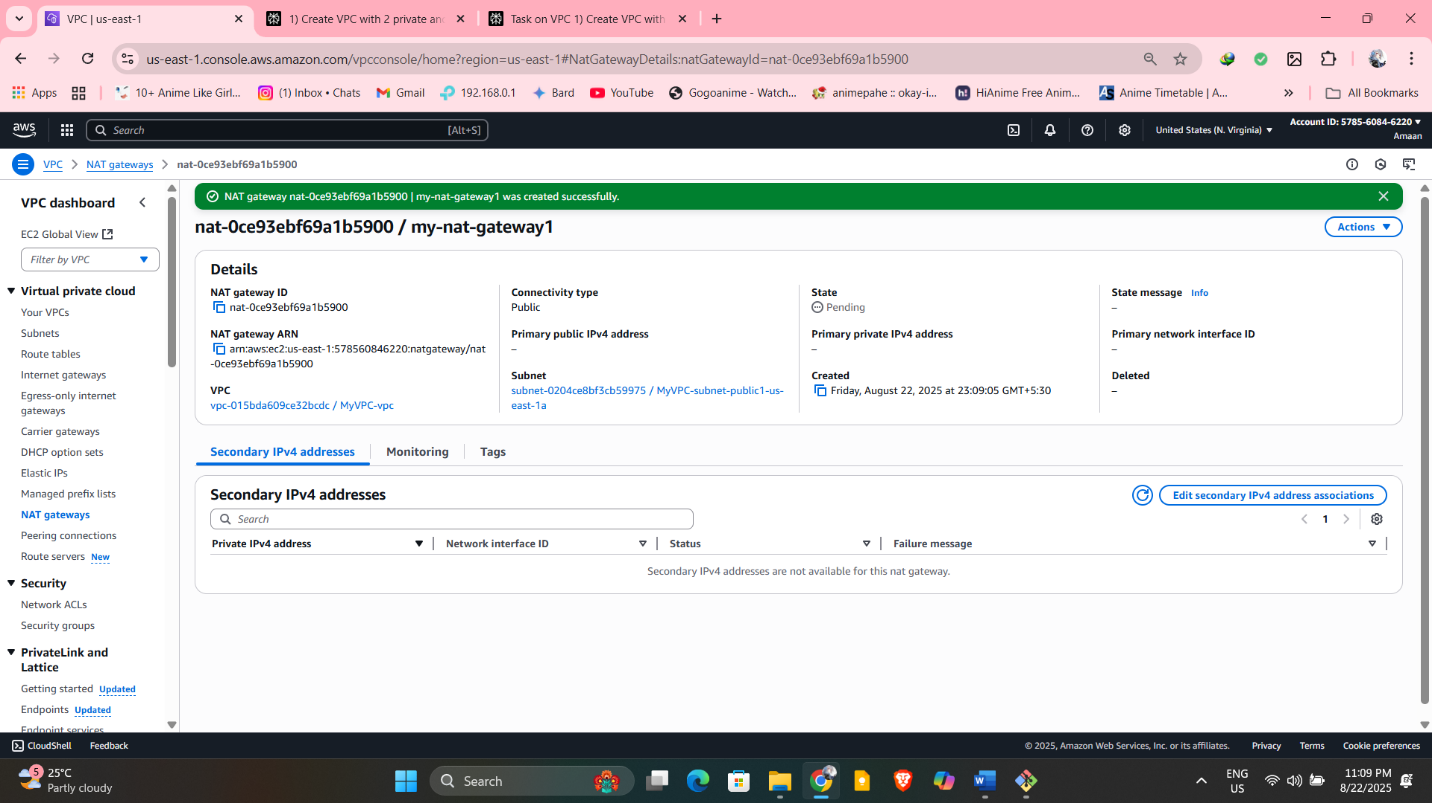
or update packages (Amazon Linux):

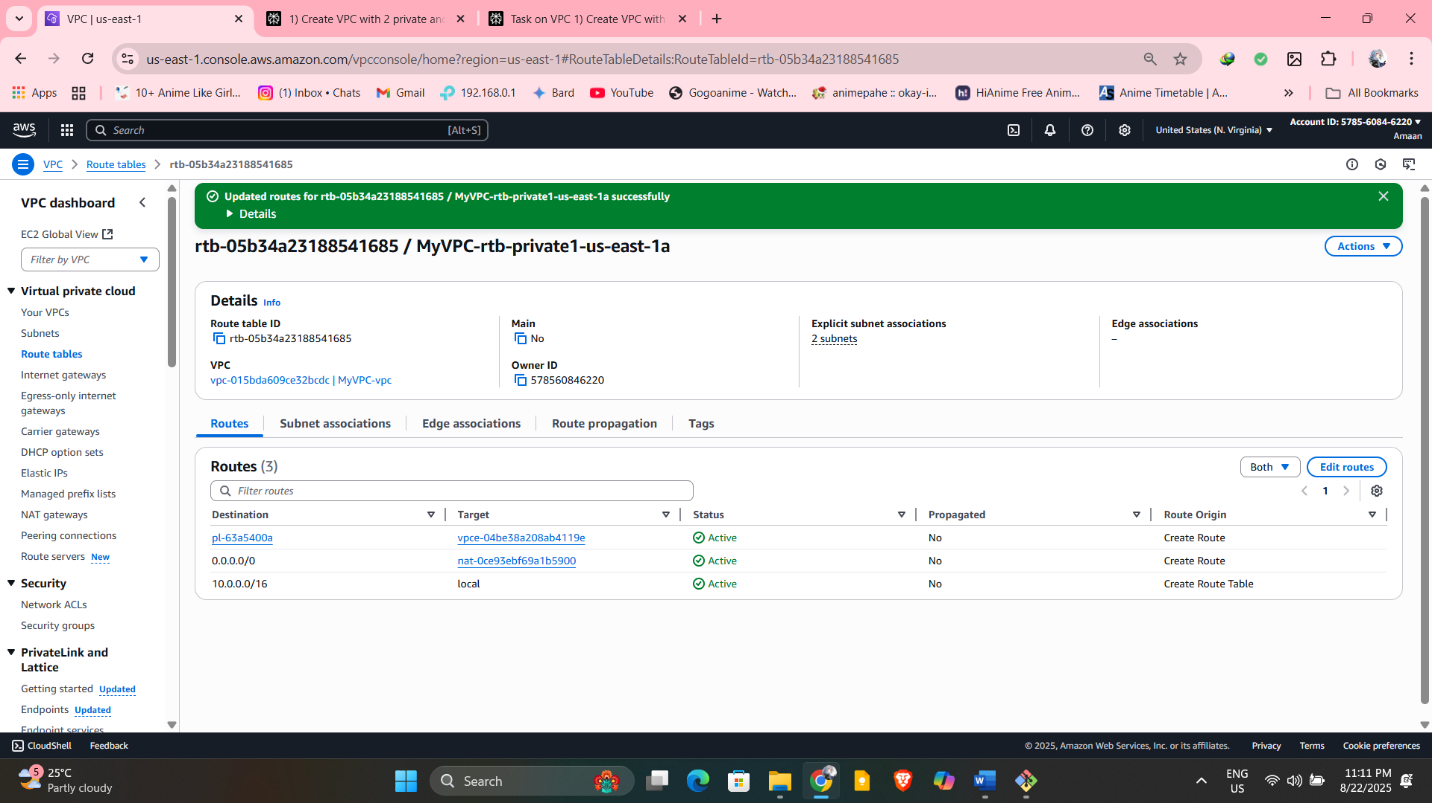
text

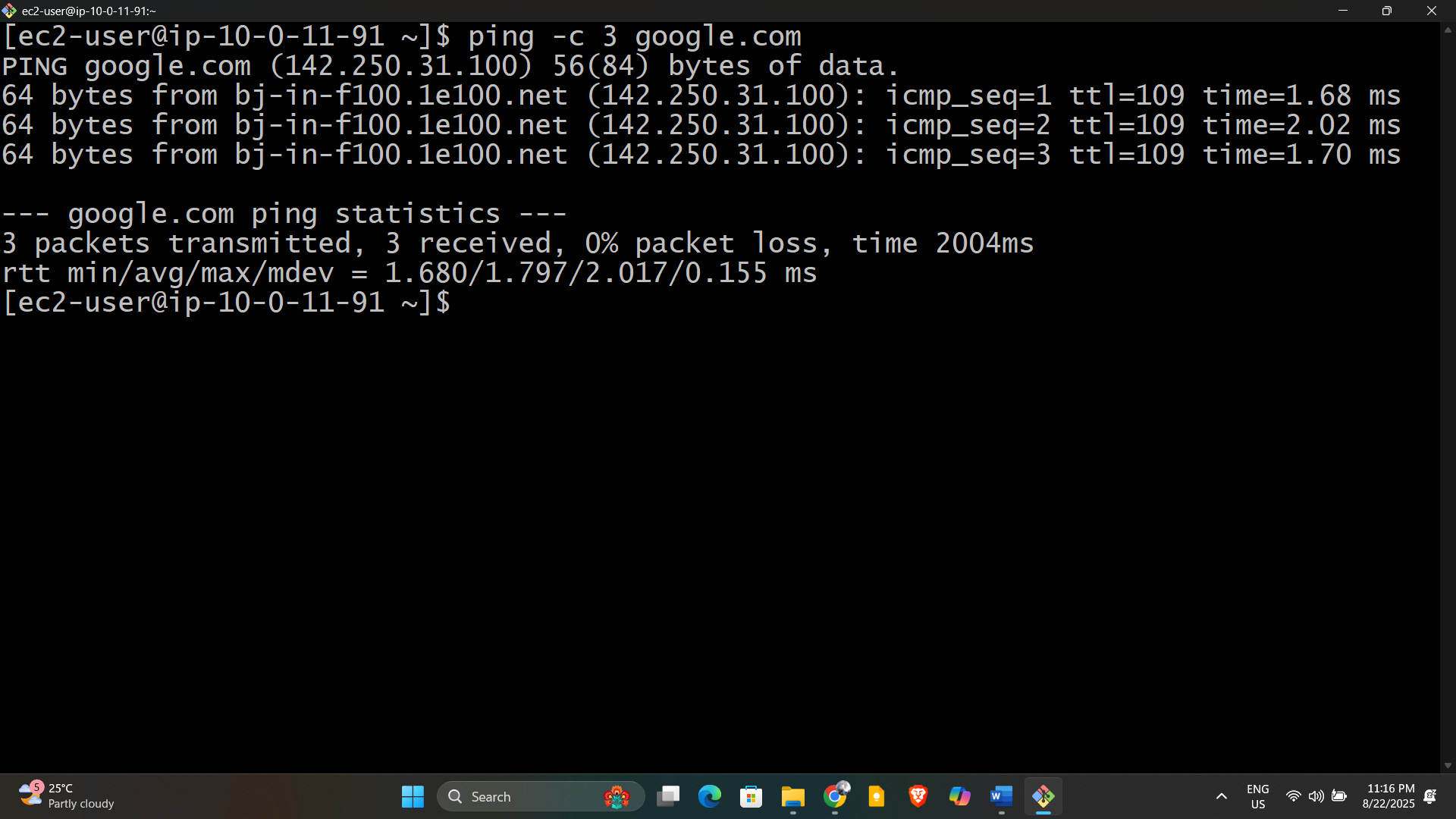
sudo yum update -y

1. **Successful output means:**
   * Your private instance is routing its internet-bound traffic through the NAT Gateway correctly.
   * If you get timeouts or errors, verify NAT Gateway and route table configuration, and security group settings.









9) Install Apache Tomcat in private ec2 and deploy a sample app.

**Step 1: Connect to Your Private EC2 Instance**

* Use a bastion host or AWS Session Manager to connect to your private instance via SSH.

**Step 2: Install Java (Tomcat Requirement)**

Apache Tomcat requires Java to run.

bash

sudo yum update -y

sudo yum install -y java-1.8.0-openjdk

java -version

**Step 3: Download and Install Apache Tomcat**

1. **Go to the official Tomcat download page to get the latest version link or use this example (Tomcat 9):**

cd /opt

sudo wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.108/bin/apache-tomcat-9.0.108.tar.gz

sudo tar xvf apache-tomcat-9.0.108.tar.gz

sudo mv apache-tomcat-9.0.108 tomcat9

sudo chown -R ec2-user:ec2-user /opt/tomcat9

Then proceed with starting Tomcat and deploying your apps as described earlier:

bash

cd /opt/tomcat9/bin

./startup.sh

This installs the latest version of Tomcat 9 as per your provided URL.

**Check if Tomcat is running:**

bash

ps -ef | grep tomcat

**Step 4: Deploy a Sample App to Tomcat**

1. **Create a simple WAR file or download a sample example:**

You can download a sample WAR (e.g., Hello World example):

bash

wget https://tomcat.apache.org/tomcat-7.0-doc/appdev/sample/sample.war -P /opt/tomcat9/webapps/

1. **Tomcat automatically deploys WAR files placed in /opt/tomcat9/webapps.**

**Step 5: Access the Tomcat Server (For Testing)**

* Since your EC2 instance is in a private subnet, you cannot access Tomcat directly from outside unless you set up a bastion host or VPN.
* To test locally, from the private instance, curl the Tomcat app:

**bash**

curl http://localhost:8080/sample

You should see the sample app content**.**

**To Verify Deployment:**

1. Wait about a minute for Tomcat to deploy the application automatically.
2. Access the sample application in your browser:

text

http://<EC2-Public-IP>:8080/sample

Replace <EC2-Public-IP> with your instance's public IP.

**Note on Your Commands:**

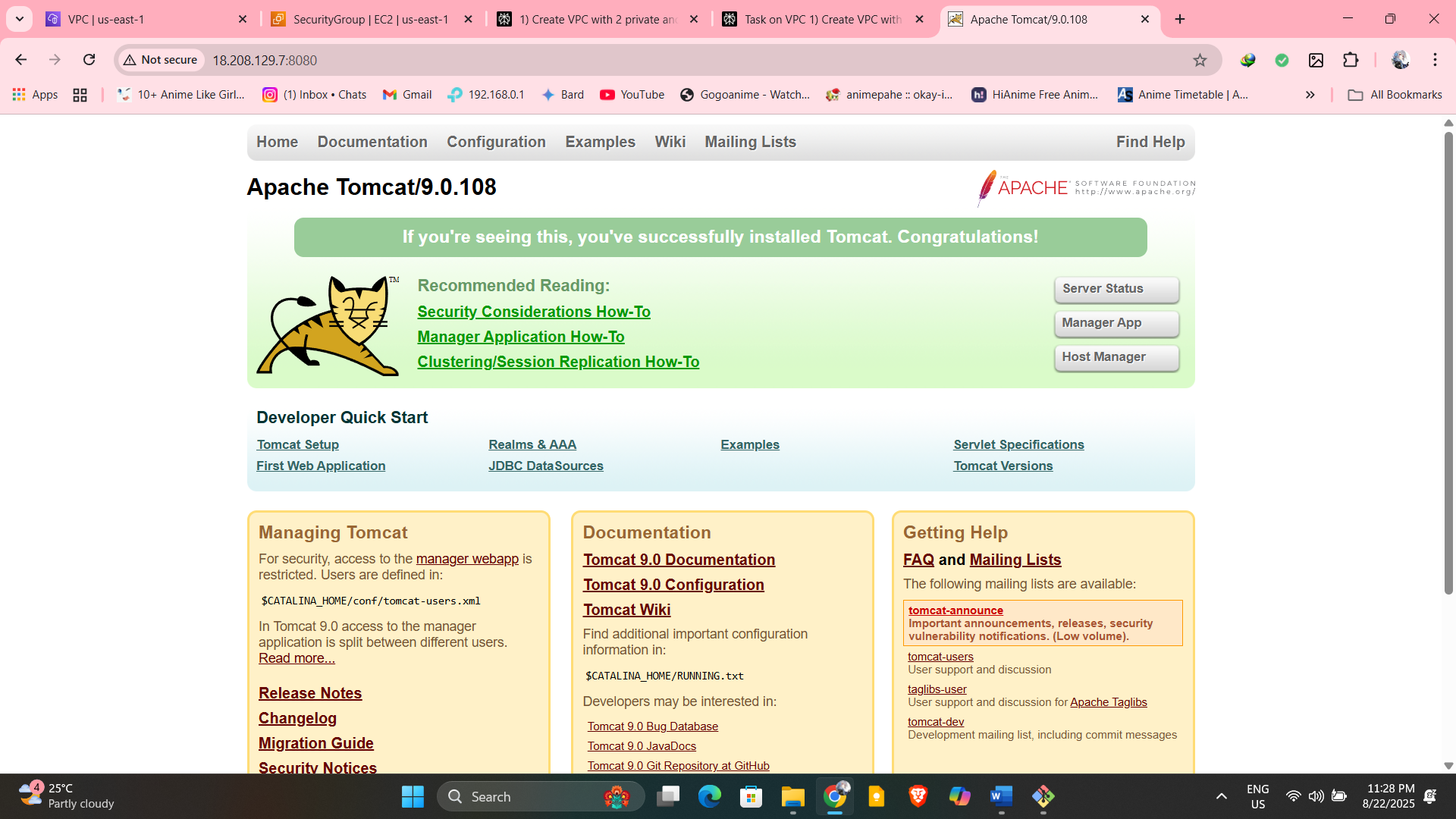
* /opt/tomcat9/webapps is a directory, so you cannot execute it directly.
* To navigate into the directory on your EC2 instance, use:

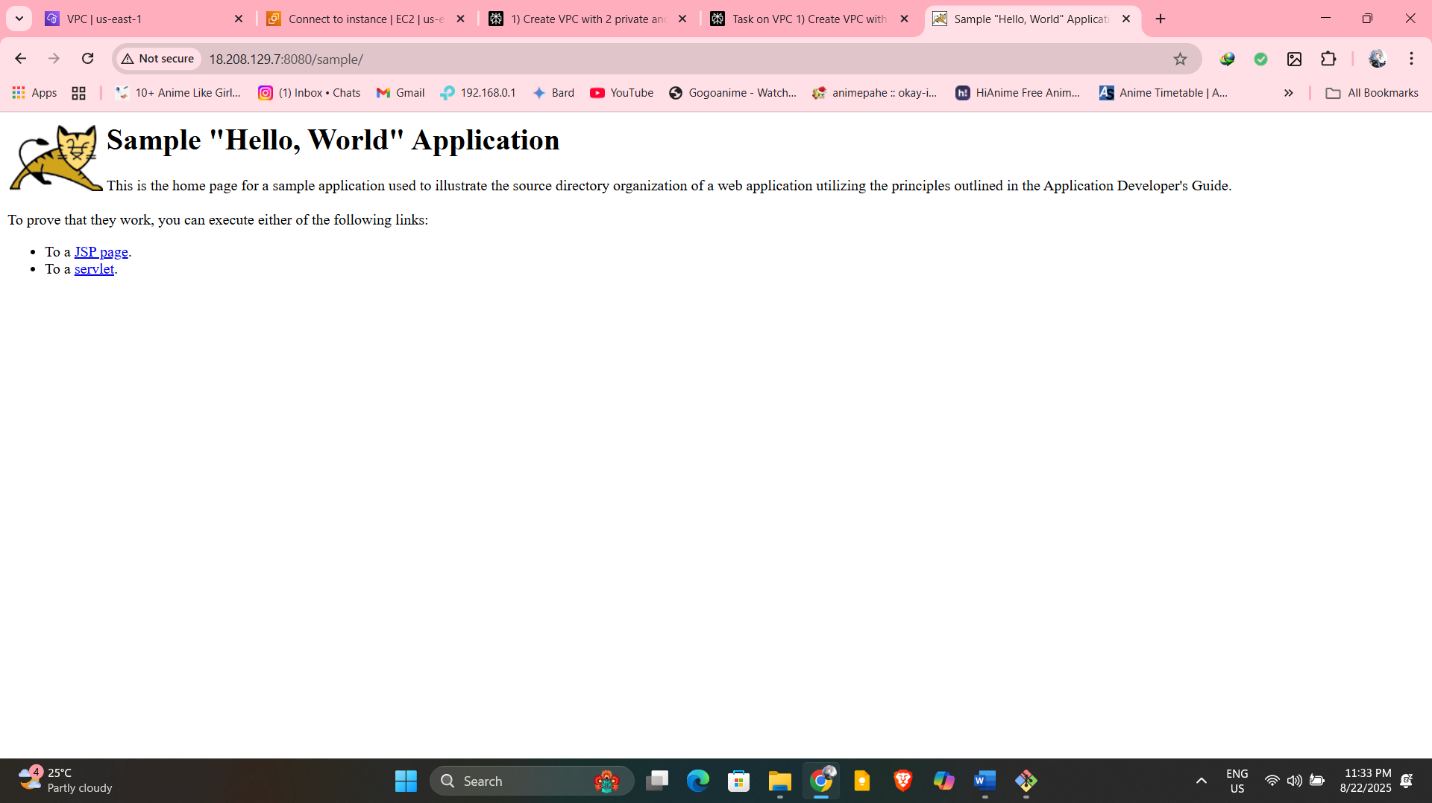
bash

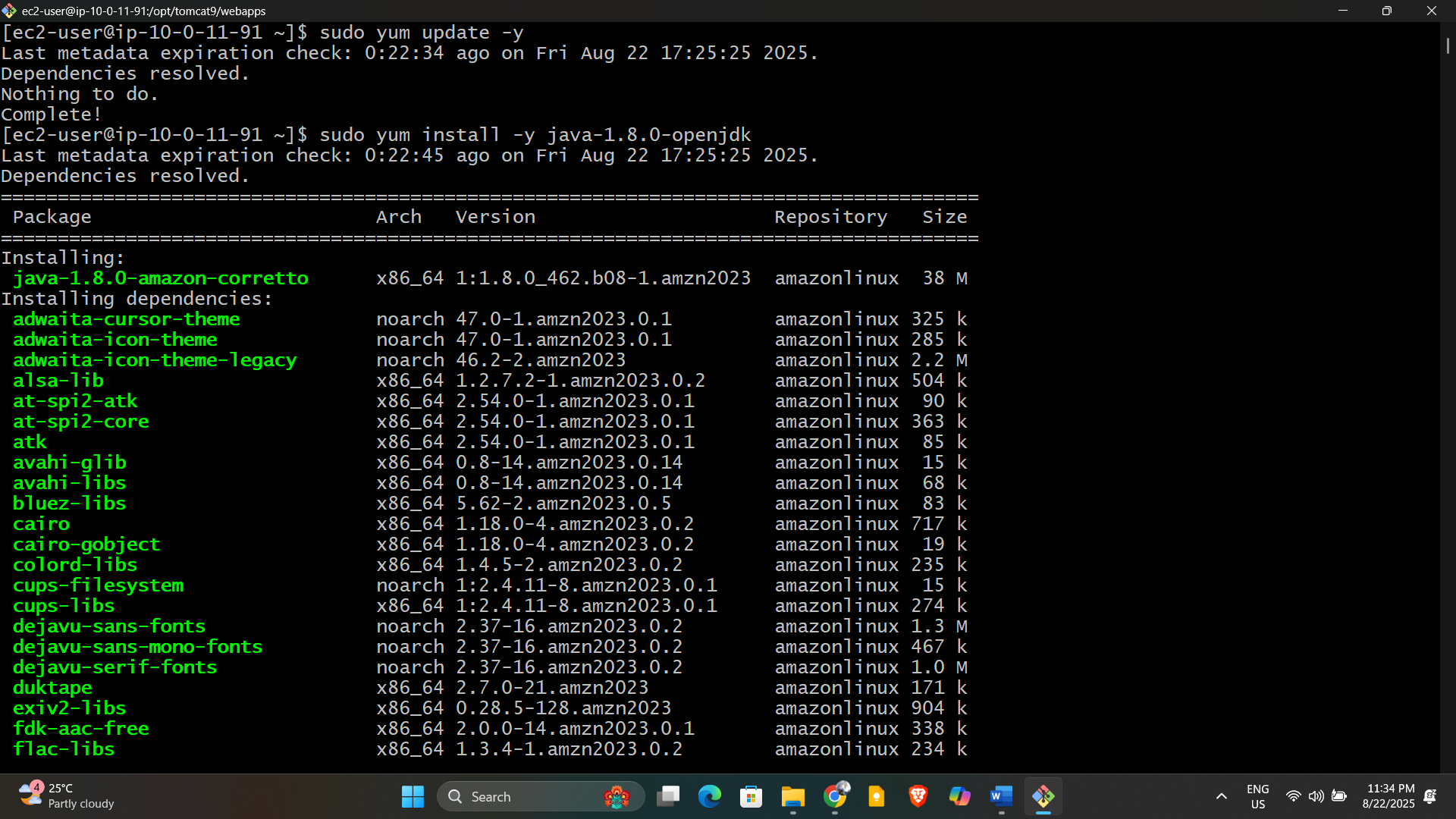
cd /opt/tomcat9/webapps

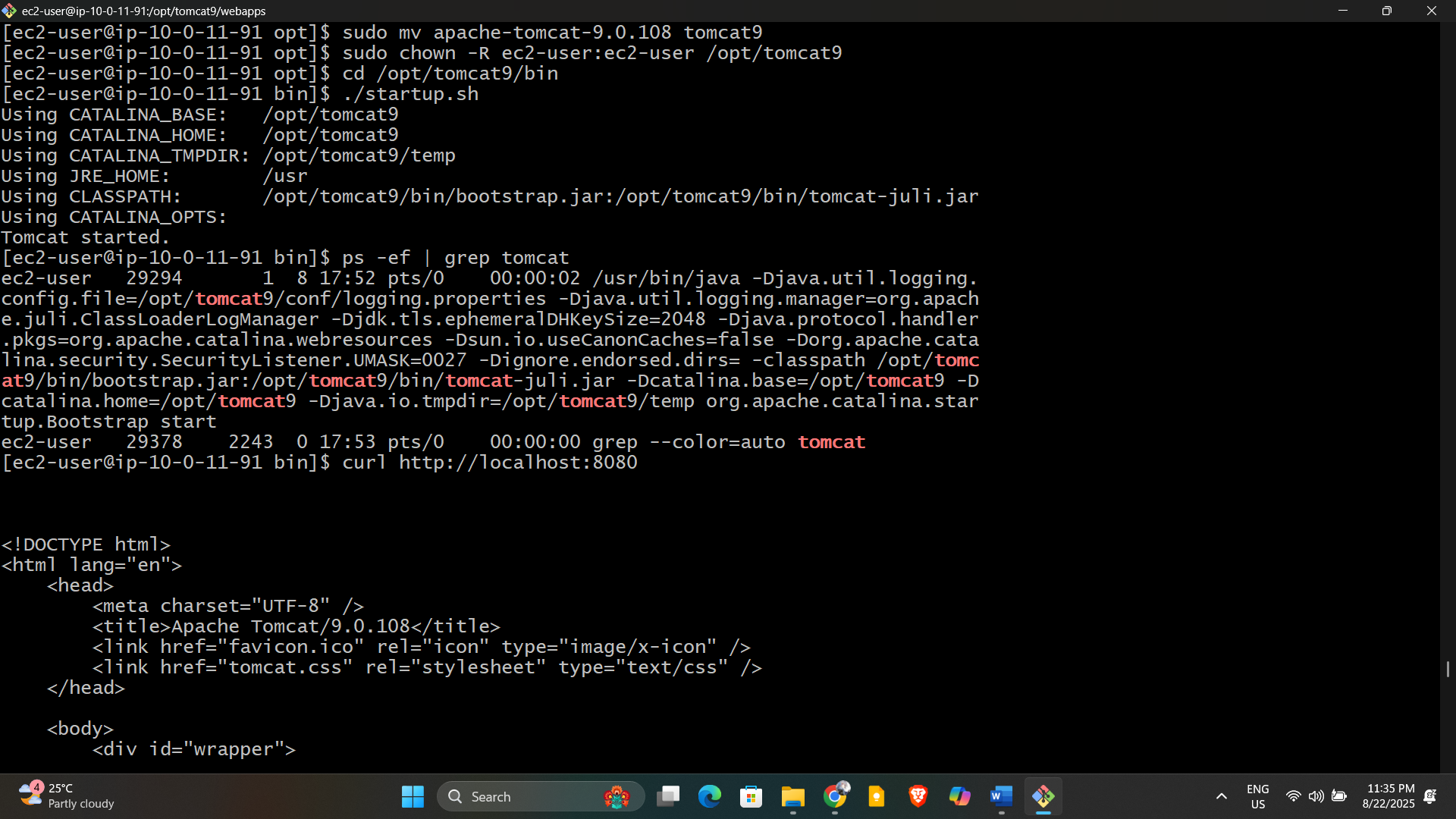
ls -l

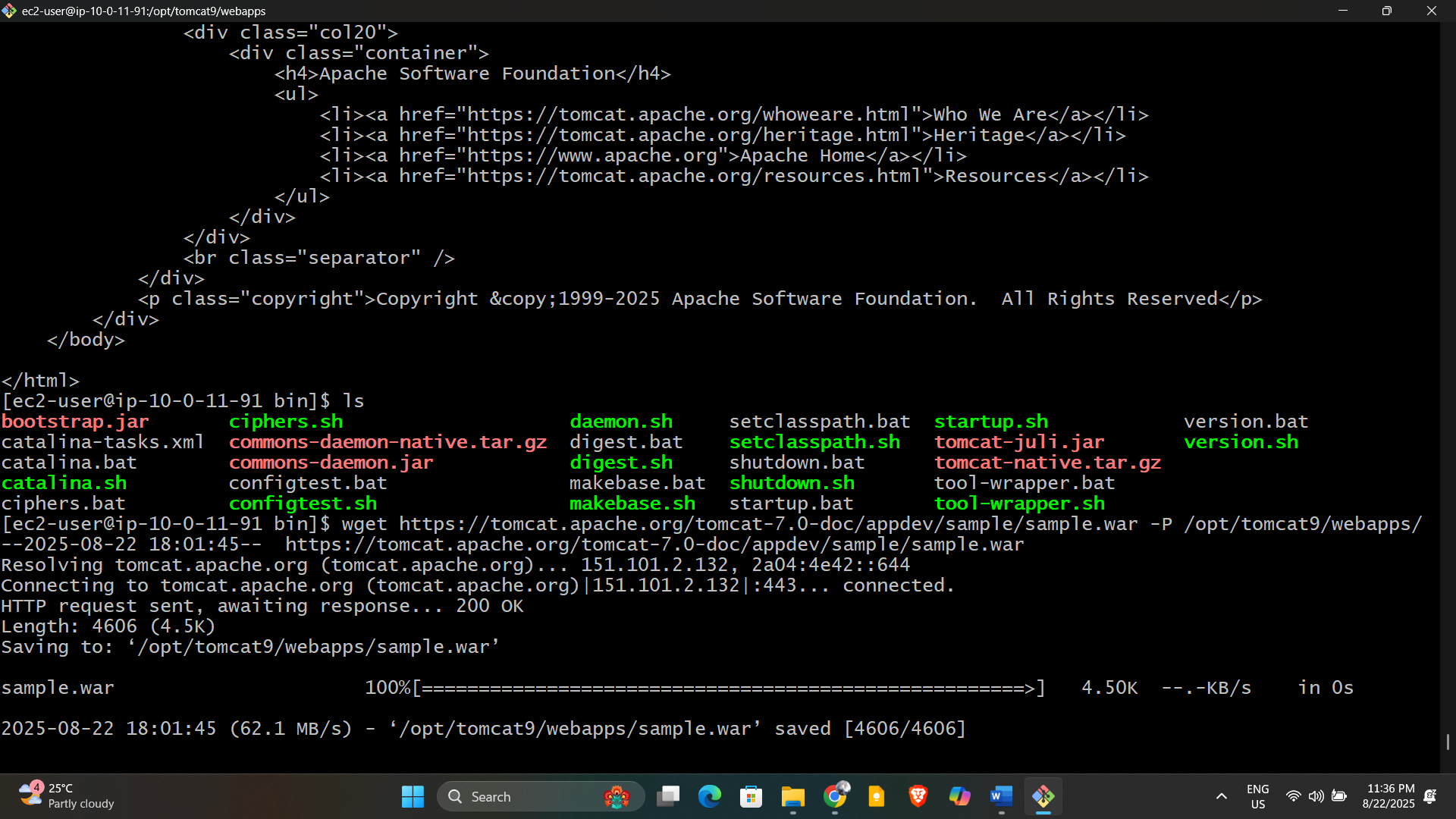
This will show the deployed WAR and exploded files.

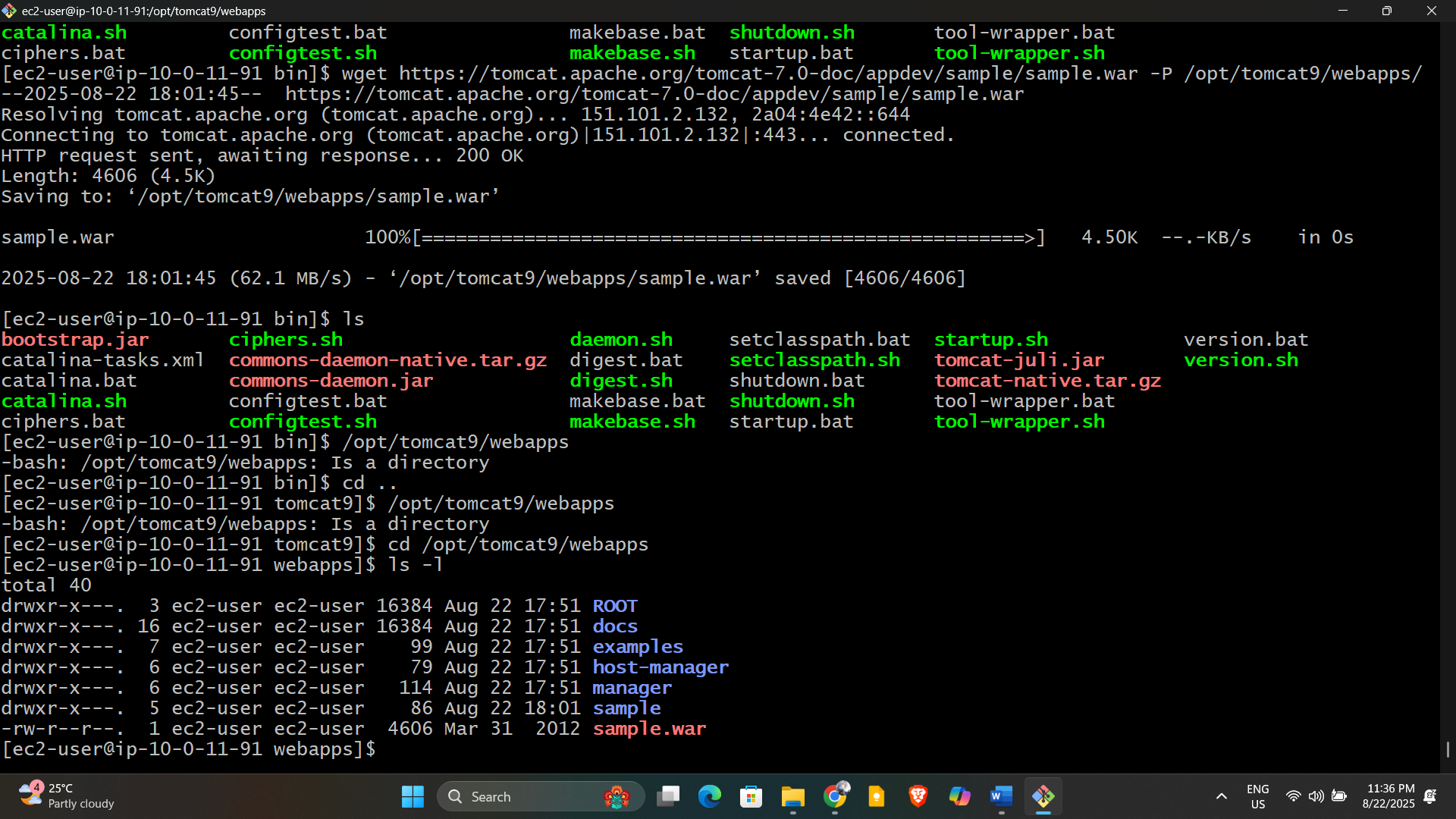












10) Configure VPC flow logs and store the logs in s3 and CloudWatch.

**Step 1: Create an S3 Bucket for VPC Flow Logs**

1. Open the S3 Console.
2. Click Create bucket.
3. Enter a unique bucket name (e.g., my-vpc-flow-logs-<unique-id>).
4. Choose region matching your VPC.
5. Complete the settings and create the bucket.

**Step 2: Create an IAM Role for Flow Logs**

1. Go to IAM Console > Roles > Create role.
2. Choose EC2 as trusted entity (for flow logs, you might use AWS service role for VPC Flow Logs).
3. Attach policy: AWSLogsDeliveryAccess or a custom policy allowing writing to CloudWatch and S3.
4. Name and create the role.

**How to Attach an Inline Policy to a Role for VPC Flow Logs**

1. **Click the Role Name**
   * Find the role you want to use for VPC Flow Logs (for example, Attached\_Policy or Attached\_Policy\_New if you created these for flow logs).
   * Click directly on the role name (the blue link), not the checkbox.
2. **Go to Permissions Tab**
   * When the role details page opens, click on the “Permissions” tab.
3. **Add Inline Policy**
   * In the Permissions tab, look for Add inline policy (usually at the top right or bottom of permissions list).
   * Click Add inline policy.
4. **Switch to the JSON Editor**
   * In the policy creation screen, switch to the JSON tab.
5. **Paste the Policy**
   * Copy and paste the following, replacing my-vpc-flow-logs1 with your bucket name if needed:

json

{

"Version": "2012-10-17",

"Statement": [

{

"Action": [

"logs:CreateLogGroup",

"logs:CreateLogStream",

"logs:PutLogEvents"

],

"Effect": "Allow",

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": [

"s3:PutObject",

"s3:GetBucketLocation"

],

"Resource": [

"arn:aws:s3:::my-vpc-flow-logs1/\*",

"arn:aws:s3:::my-vpc-flow-logs1"

]

}

]

}

1. **Review & Create**
   * Click “Review policy,” give it a name like VPCFlowLogsS3CloudWatchAccess, and then click Create policy.

After this, your role will have permissions for VPC Flow Logs to write to both S3 and CloudWatch. When you create the VPC Flow Log, select this IAM role in the role dropdown.

**Step-by-Step: Complete Your VPC Flow Log Setup**

1. Name:
   * You can use the default or enter a custom name for your flow log (e.g., my-flow-log-01).
2. Filter:
   * Leave it set to All to capture all network traffic.
3. Maximum Aggregation Interval:
   * Select 10 minutes or 1 minute depending on your log detail needs (1 minute gives more granular data but may incur higher costs).
4. Destination:
   * Choose your destination for logs:
     + Send to CloudWatch Logs:
       - In "Destination log group," select or create a log group (e.g., /vpc/flowlogs).
     + Send to Amazon S3 bucket:
       - Enter your bucket name (my-vpc-flow-logs1).
5. Service Access:
   * Choose Use an existing service role.
   * In the “Service role” dropdown, select the IAM role you just configured (Attached\_Policy). This will allow log delivery to both S3 and CloudWatch.
6. Review Settings and Click Create Flow Log
   * Double check all settings, especially the destination and service role, then click the Create flow log button.

After you finish, VPC Flow Logs will be enabled for your VPC, and your traffic logs will be delivered to CloudWatch and/or S3 as configured.

