Setup a CI/CD pipeline using the tools of your choice (or preferably the mentioned tools).

1. It should deploy a simple web application to a server on a code push to a repository.

2. The deployed web application should be reachable on any web browser.

3. Make it scalable such that when load increases the number of servers scale up and down making sure, the new servers have the updated code.

Git repo: https://github.com/vommidapuchinni/simple-web-app.git

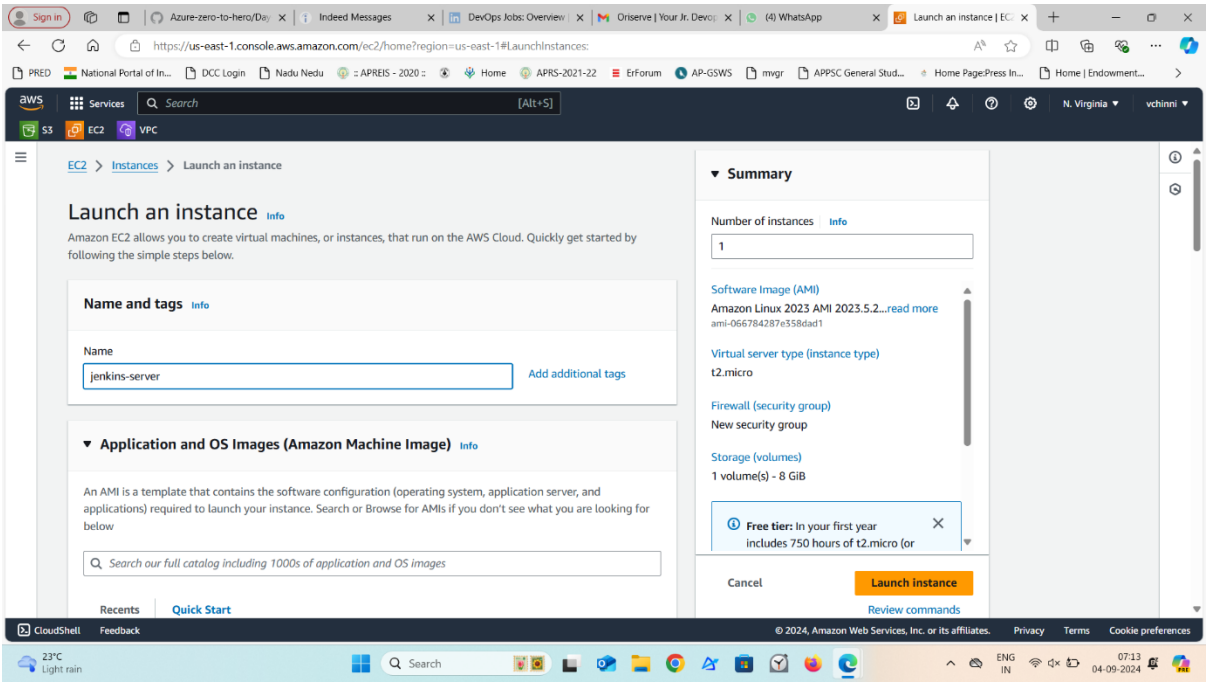
Step 1: Create a Git Repository

1. **Create a GitHub Account:**
    - Sign up or log in to GitHub.
2. **Create a New Repository:**
    - Click on the "New" button or go to GitHub New Repository.
    - Name the repository, simple-web-app.
    - Keep it public.
    - Do not initialize with a README, .gitignore, or license.
    - Click "Create Repository."
3. **Clone the Repository Locally:**
    - Open your terminal and run the following commands to clone the repository: git clone https://github.com/your-username/simple-web-app.git

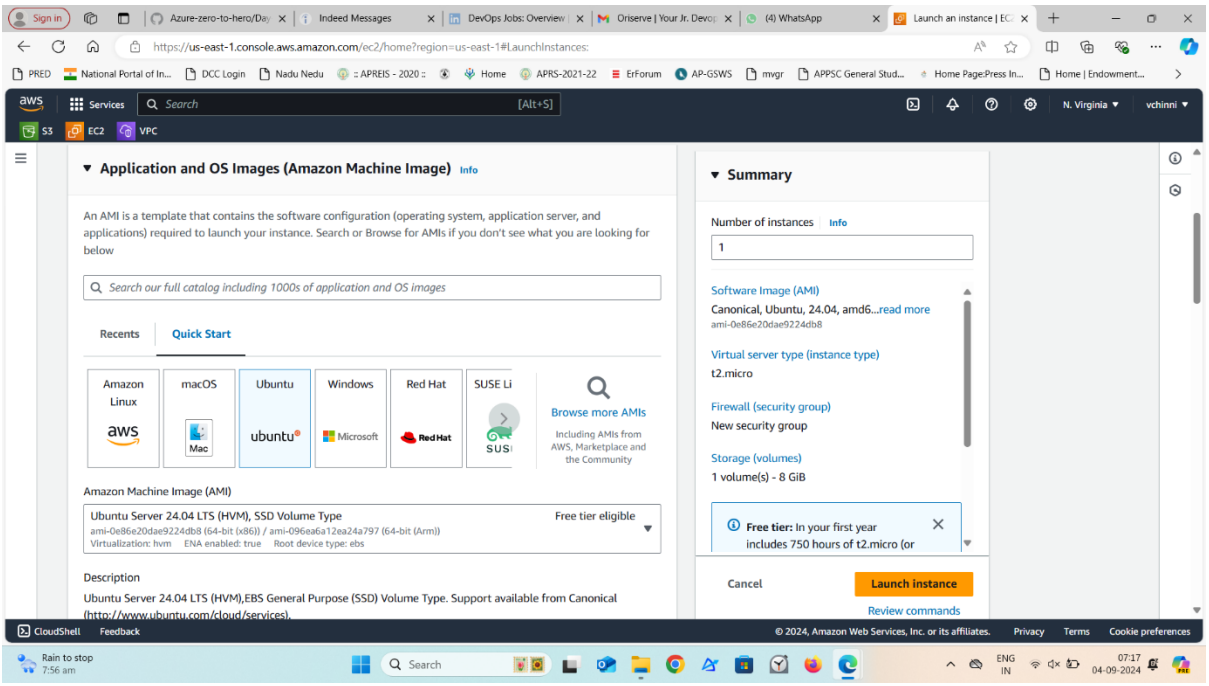
      cd simple-web-app

Step 2: Launch EC2 Instances

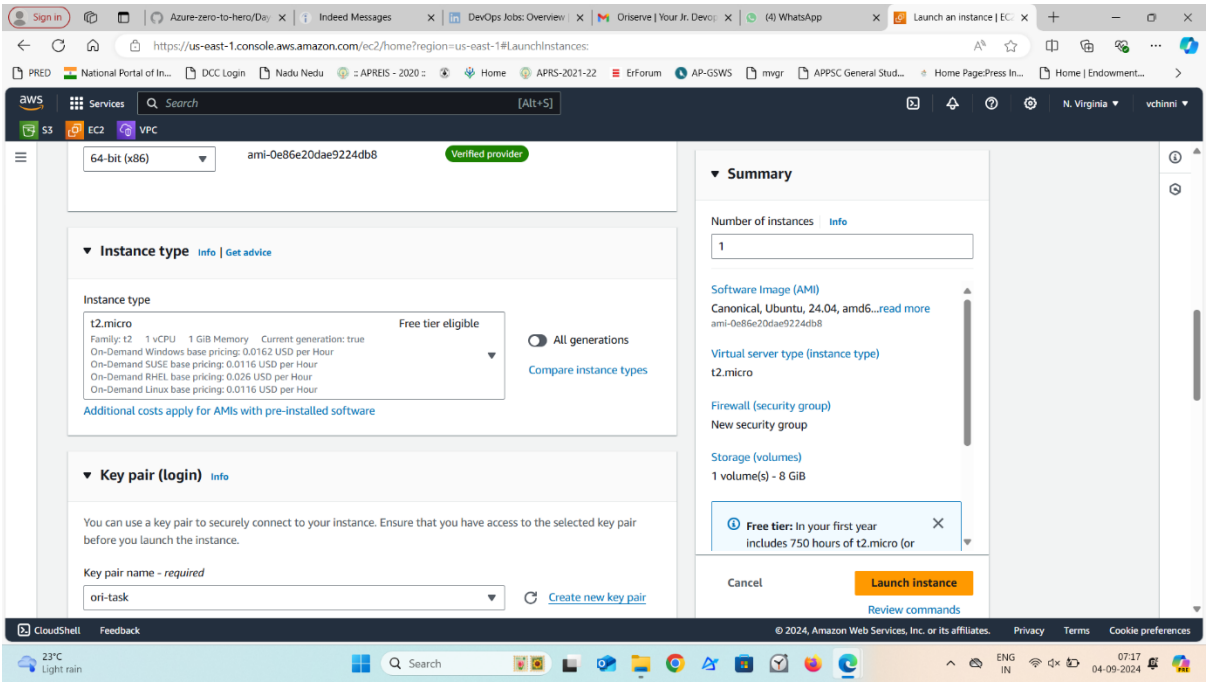
1. **Launch the Jenkins Server Instance:**
    - Go to the AWS EC2 Console.
    - Click **Launch Instance**.

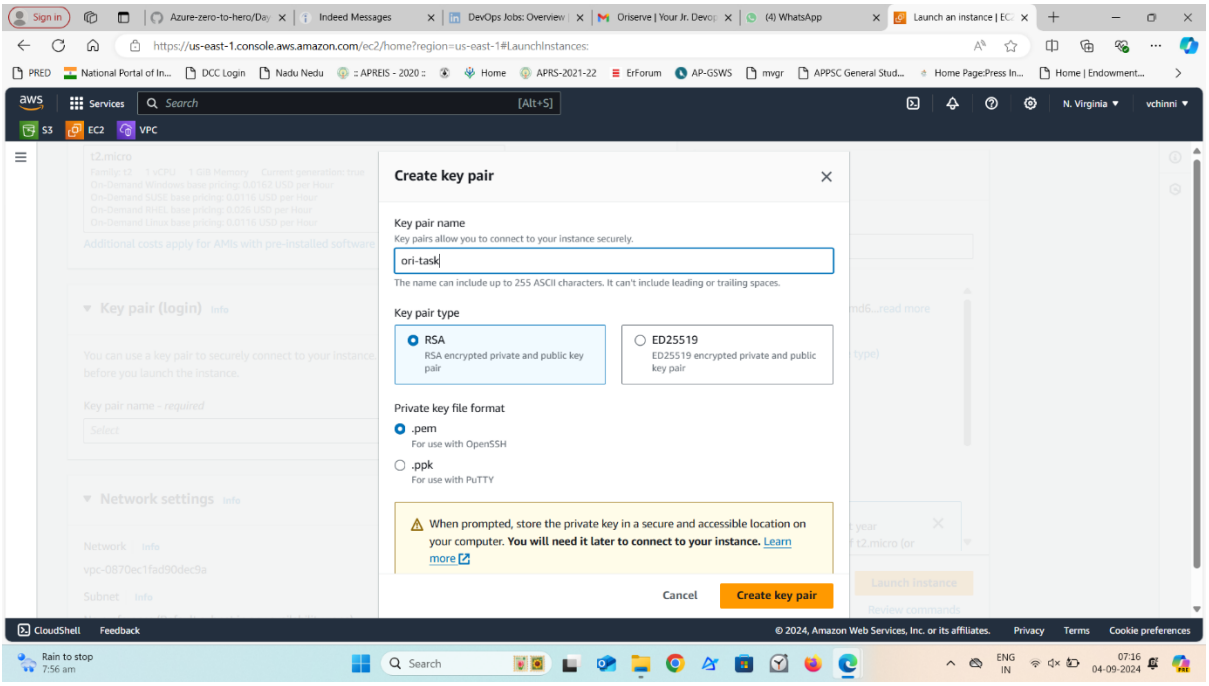


- **Name and Tags**: Name the instance jenkins-server.
- **AMI**: Choose **Ubuntu Server 20.04 LTS** (or any preferred version).



- **Instance Type**: Choose t2.micro for testing (eligible for free tier).
- **Key Pair**: create a new one.

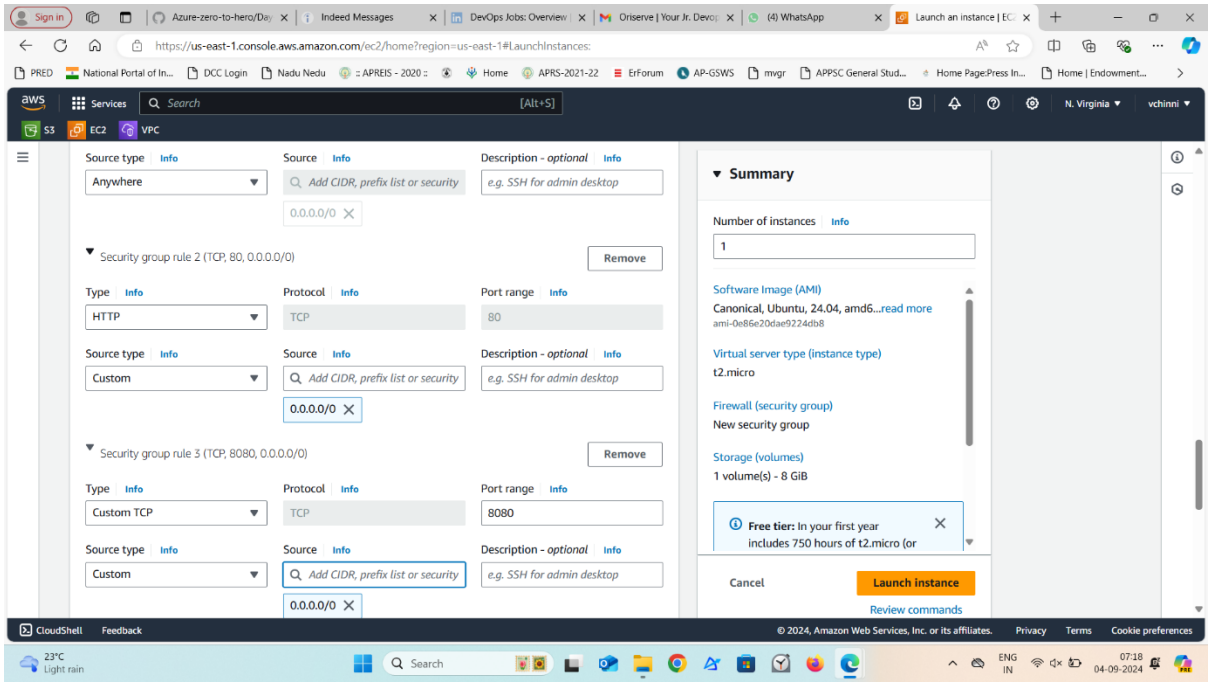


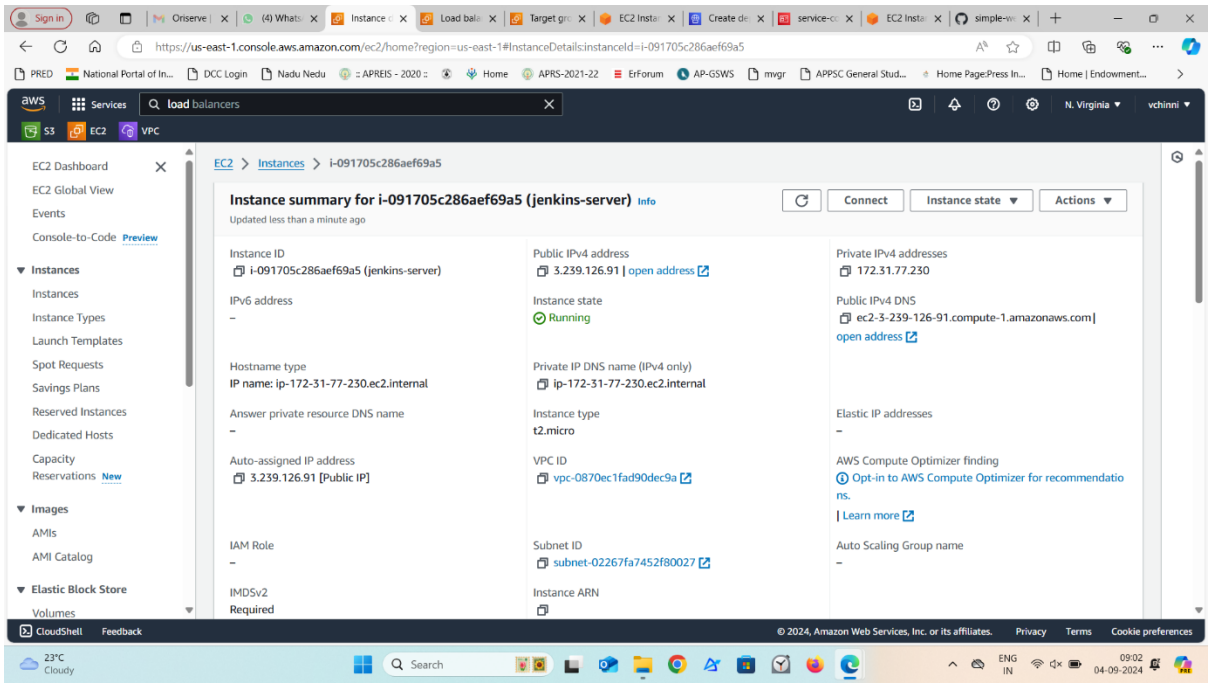


2.

- **Network Settings**:
  - Create a new security group allowing:
    - **SSH (port 22)** from your IP for secure access.
    - **HTTP (port 80)** for web access.
    - **Custom TCP (port 8080)** for Jenkins access.

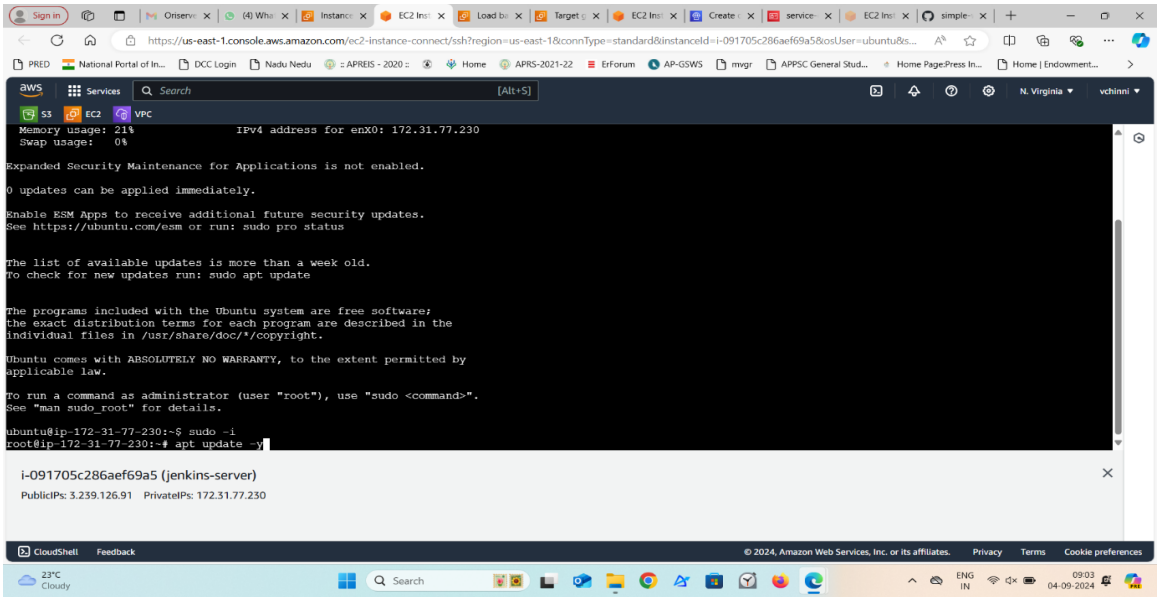
- Click **Launch Instance**.

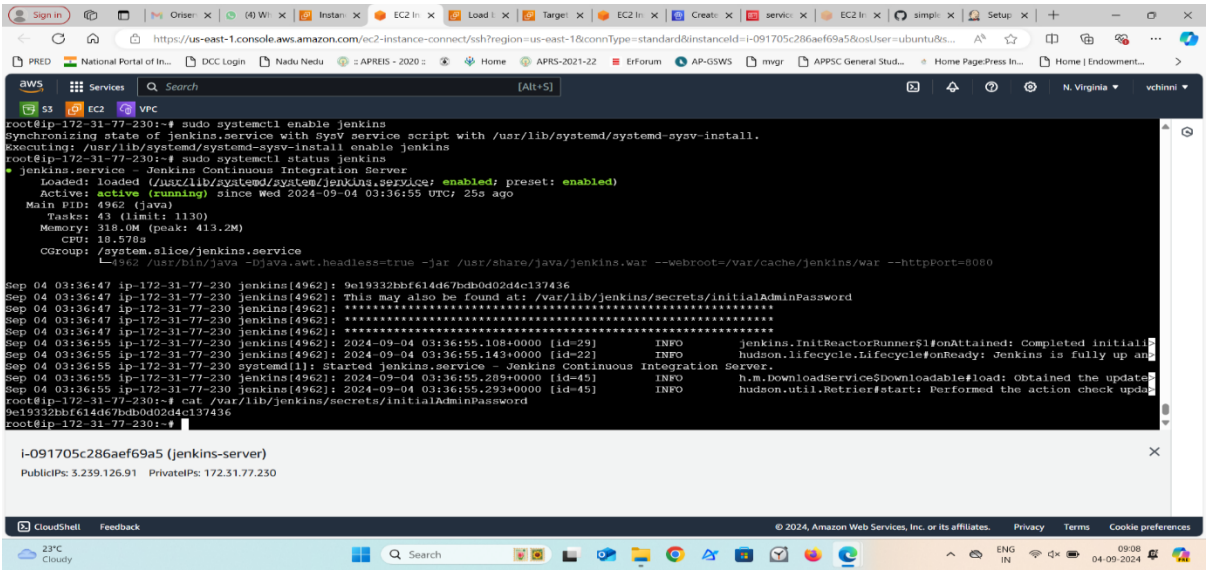


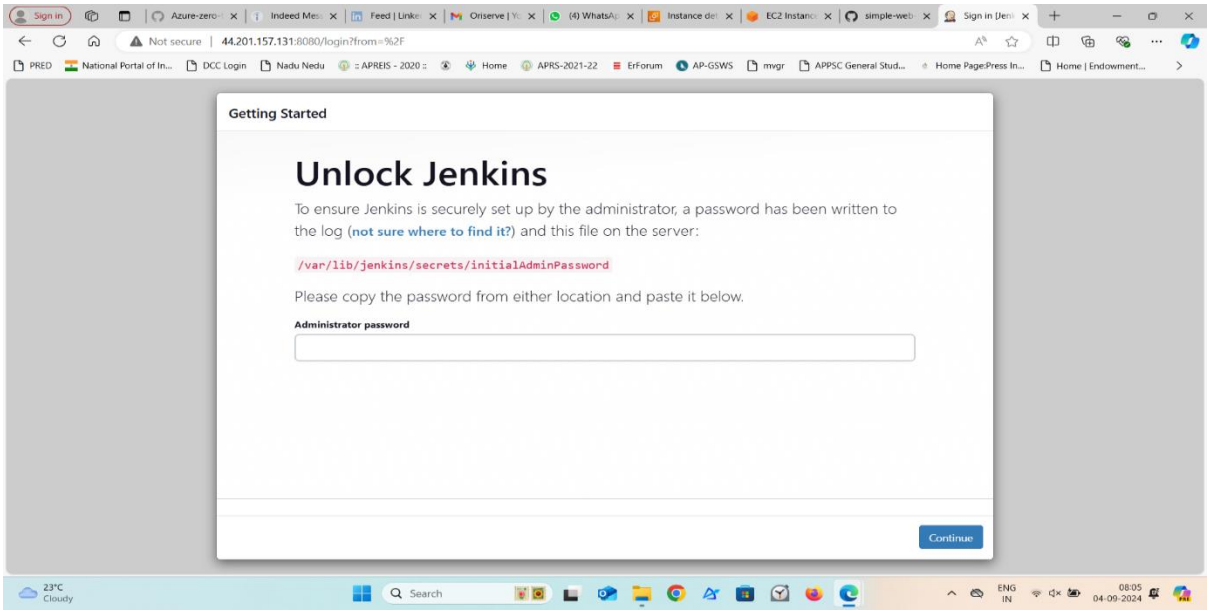


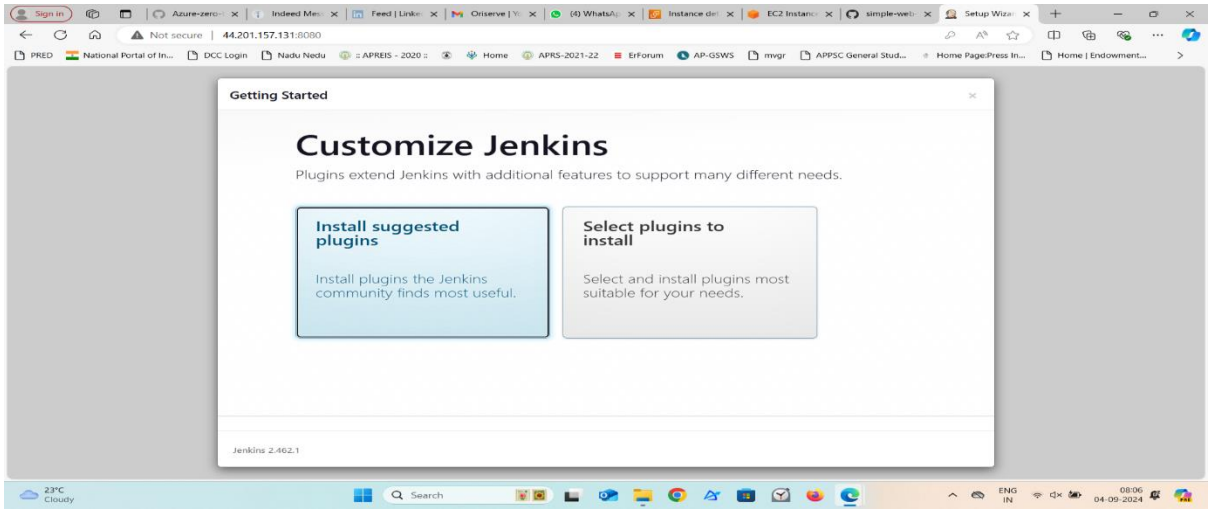
Step 3: Configure Jenkins Server

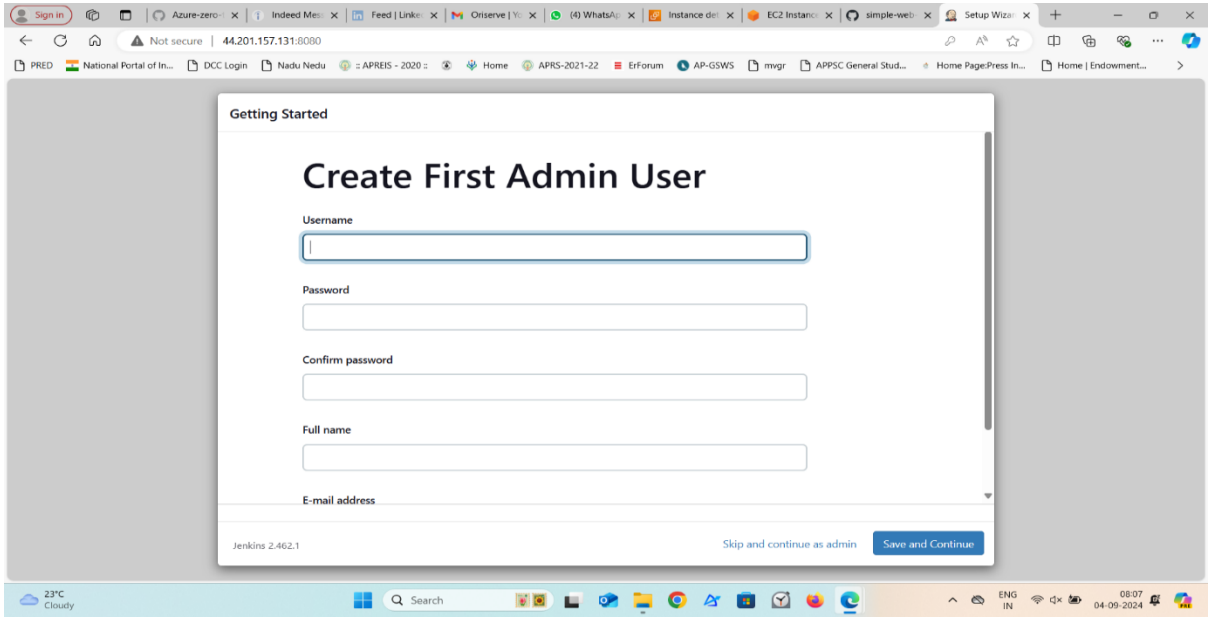
1. **Connect to the Jenkins Server Instance:**
   - Click on connect

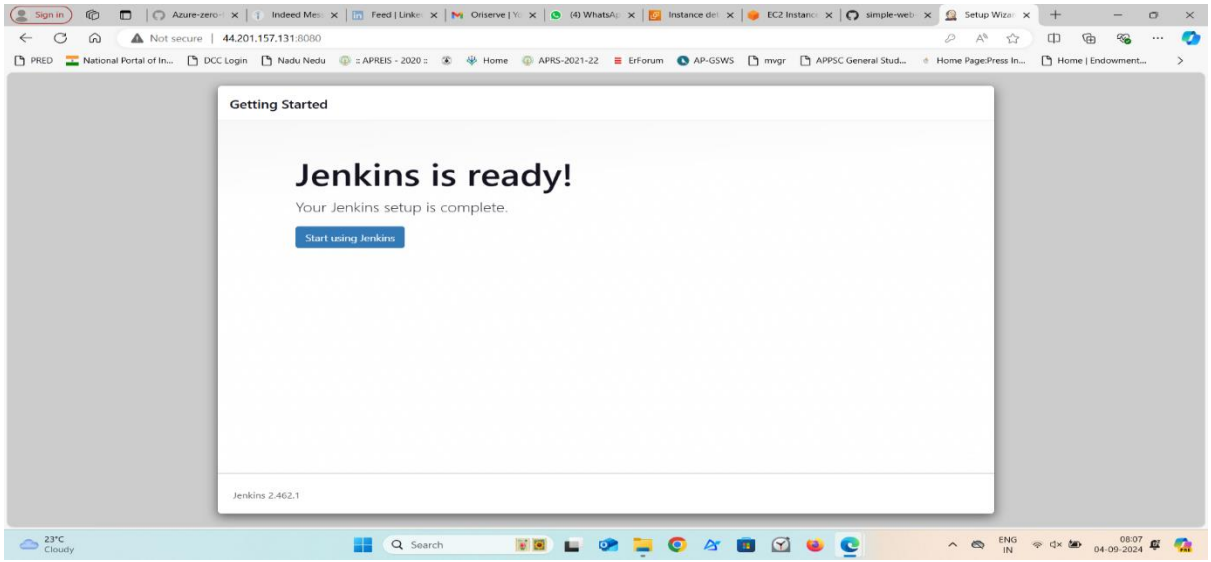

Memory usage: 21%
IPv4 address for enX0: 172.31.77.230
Swap usage: 0%
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
ubuntu@ip-172-31-77-230:~$ sudo -i
root@ip-172-31-77-230:~# apt update -y
i-091705c286aef69a5 (jenkins-server)
PublicIPs: 3.239.126.91 PrivateIPs: 172.31.77.230

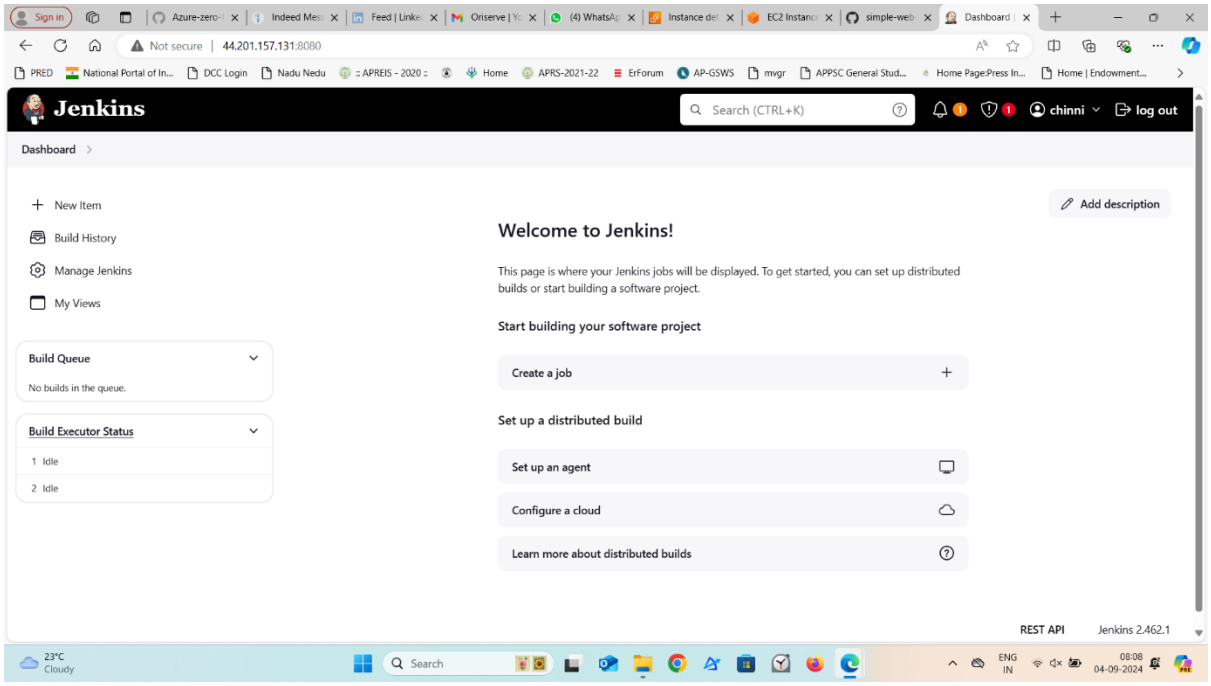

root@ip-172-31-77-230:~# sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
root@ip-172-31-77-230:~# sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
Active: active (running) since Wed 2024-09-04 03:36:55 UTC; 25s ago
Main PID: 4962 (java)
Tasks: 43 (limit: 1130)
Memory: 318.0M (peak: 413.2M)
CPU: 18.578s
CGroup: /system.slice/jenkins.service
└─4962 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080
Sep 04 03:36:47 ip-172-31-77-230 jenkins[4962]: 9e19332bbf614d67bdb0d02d4c137436
Sep 04 03:36:47 ip-172-31-77-230 jenkins[4962]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Sep 04 03:36:47 ip-172-31-77-230 jenkins[4962]: **
Sep 04 03:36:47 ip-172-31-77-230 jenkins[4962]: **
Sep 04 03:36:47 ip-172-31-77-230 jenkins[4962]: **
Sep 04 03:36:55 ip-172-31-77-230 jenkins[4962]: 2024-09-04 03:36:55.108+0000 [id=29] INFO jenkins.InitReactorRunner$1#onAttained: Completed initializ
Sep 04 03:36:55 ip-172-31-77-230 jenkins[4962]: 2024-09-04 03:36:55.143+0000 [id=22] INFO hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up an
Sep 04 03:36:55 ip-172-31-77-230 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Sep 04 03:36:55 ip-172-31-77-230 jenkins[4962]: 2024-09-04 03:36:55.289+0000 [id=45] INFO h.m.DownloadService$Downloadable#load: Obtained the updated
Sep 04 03:36:55 ip-172-31-77-230 jenkins[4962]: 2024-09-04 03:36:55.293+0000 [id=45] INFO hudson.util.Retrier#start: Performed the action check upda
root@ip-172-31-77-230:~# cat /var/lib/jenkins/secrets/initialAdminPassword
9e19332bbf614d67bdb0d02d4c137436
root@ip-172-31-77-230:~#
i-091705c286aef69a5 (jenkins-server)
PublicIPs: 3.239.126.91 PrivateIPs: 172.31.77.230


Getting Started
Unlock Jenkins
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:
/var/lib/jenkins/secrets/initialAdminPassword
Please copy the password from either location and paste it below.
Administrator password
Continue


Not secure | 44.201.157.131:8080
Getting Started
Customize Jenkins
Plugins extend Jenkins with additional features to support many different needs.
Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.
Jenkins 2.462.1


Not secure | 44.201.157.131:8080
Getting Started
Create First Admin User
Username
Password
Confirm password
Full name
E-mail address
Jenkins 2.462.1
Skip and continue as admin
Save and Continue


Not secure | 44.201.157.131:8080
Getting Started
Jenkins is ready!
Your Jenkins setup is complete.
Start using Jenkins
Jenkins 2.462.1

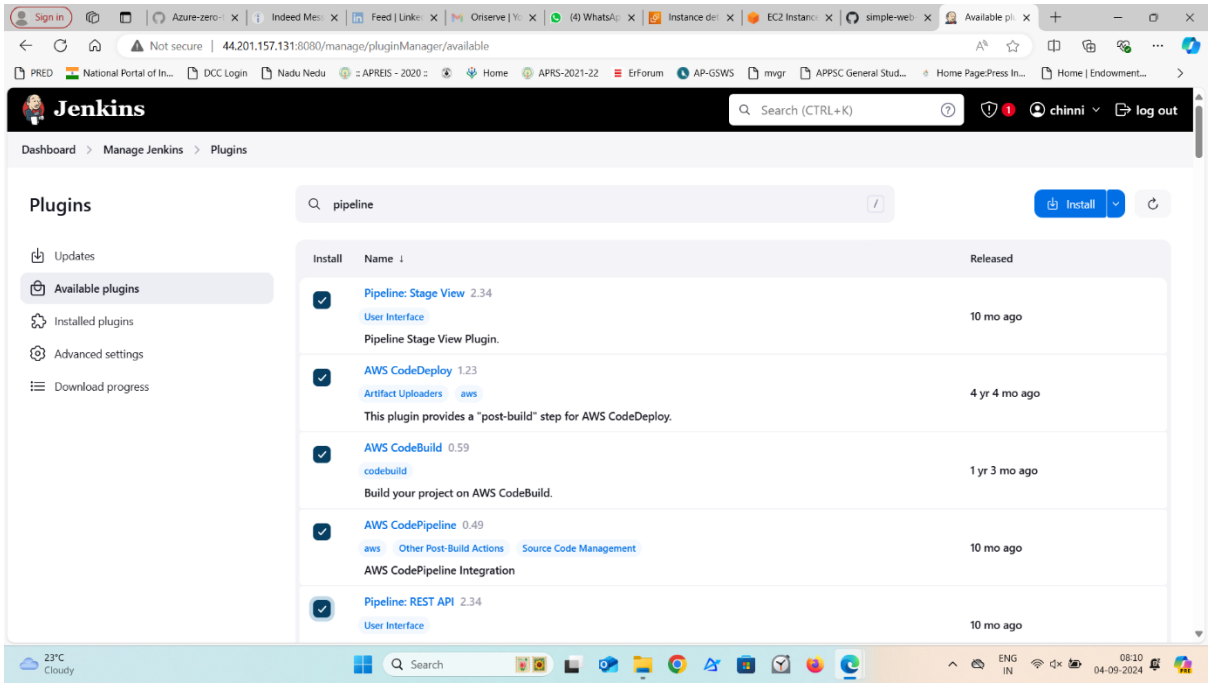


2. Install Jenkins
3. Access Jenkins

   Retrieve the initial admin password using:

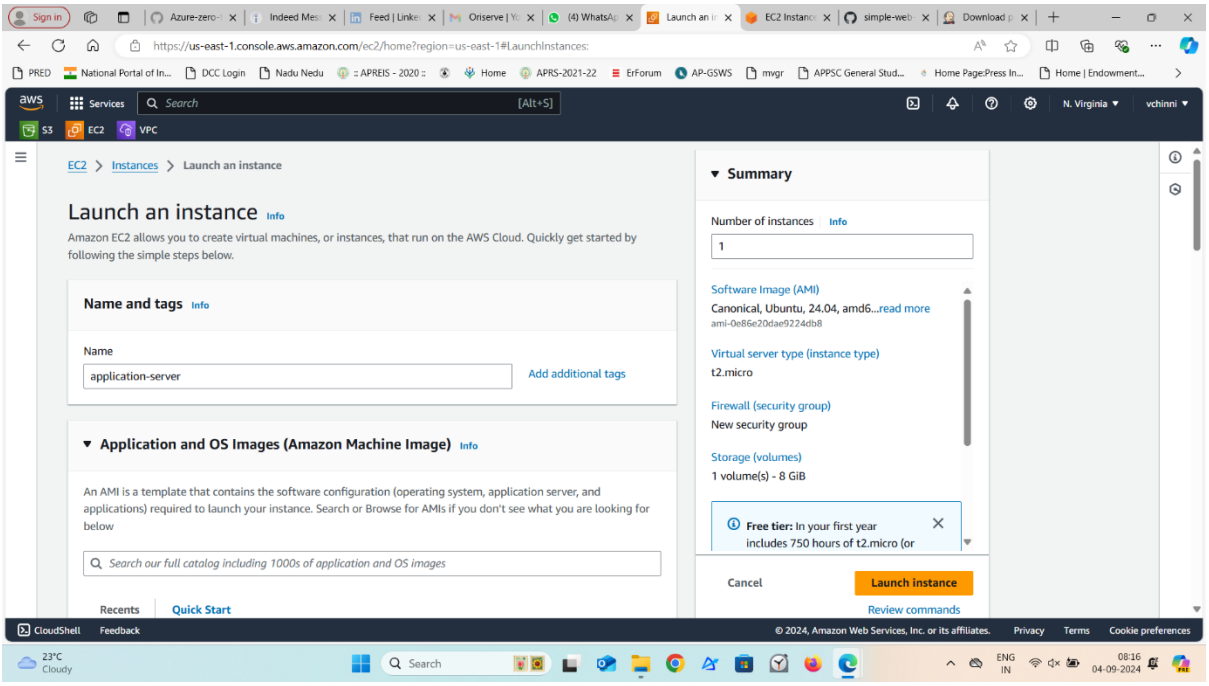
   sudo cat /var/lib/jenkins/secrets/initialAdminPassword

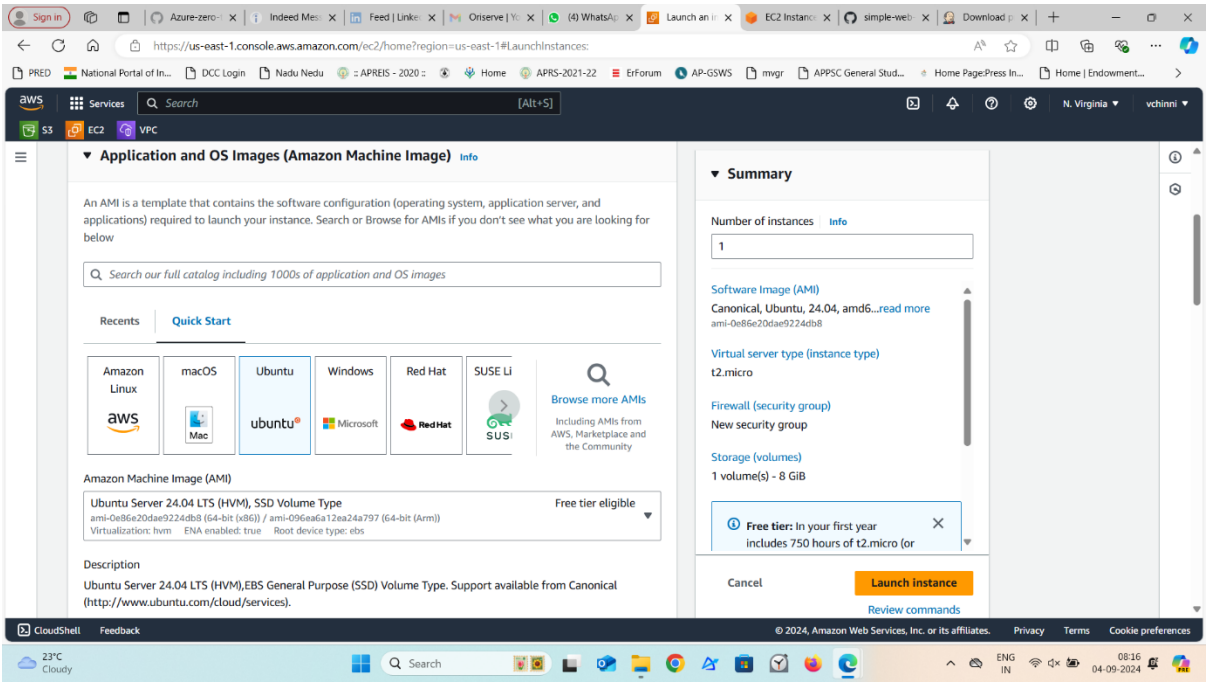
4. Install Required Jenkins Plugins: Install AWS CodeDeploy, Pipeline, and any other required plugins

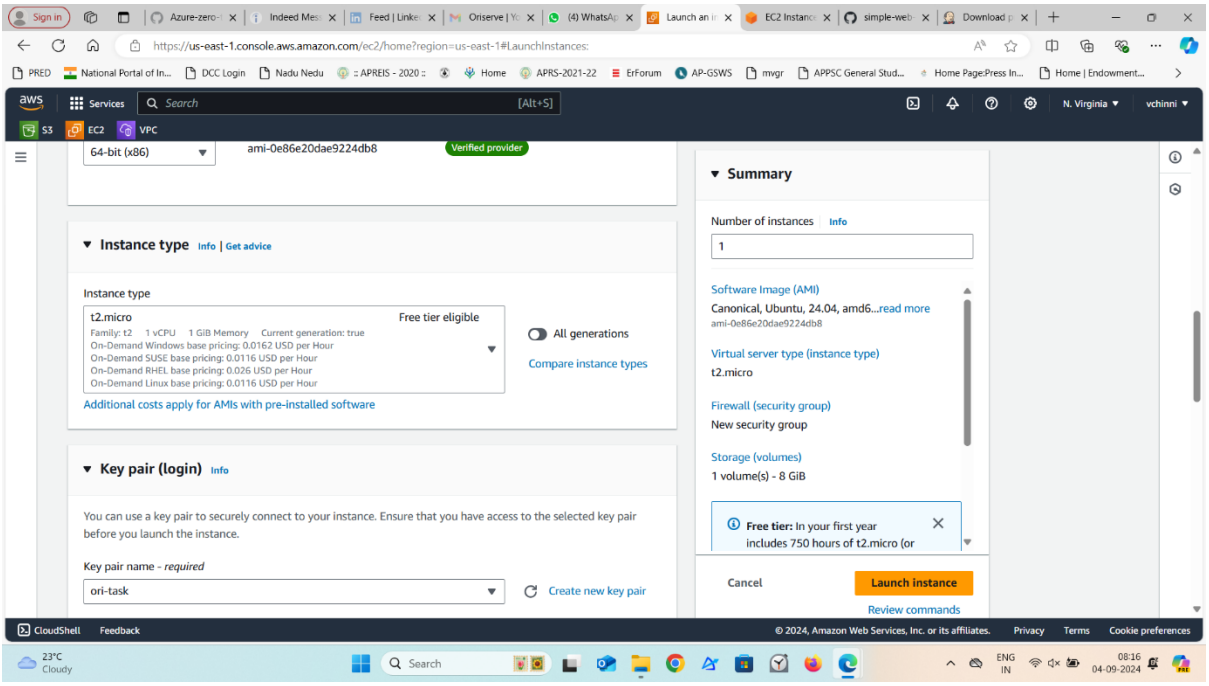


Step 4: Launch Application Server Instances (For CodeDeploy and Scaling):

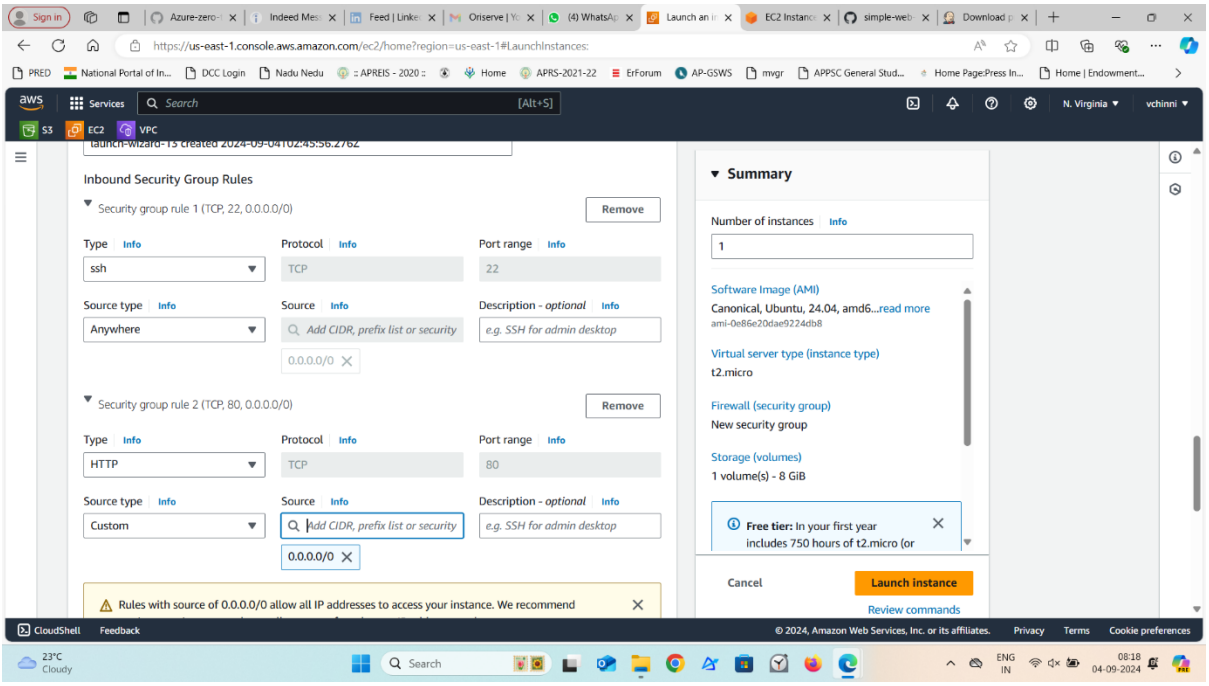
- Repeat the steps above to launch an instance.
- **Name**: Name the instance application-server.





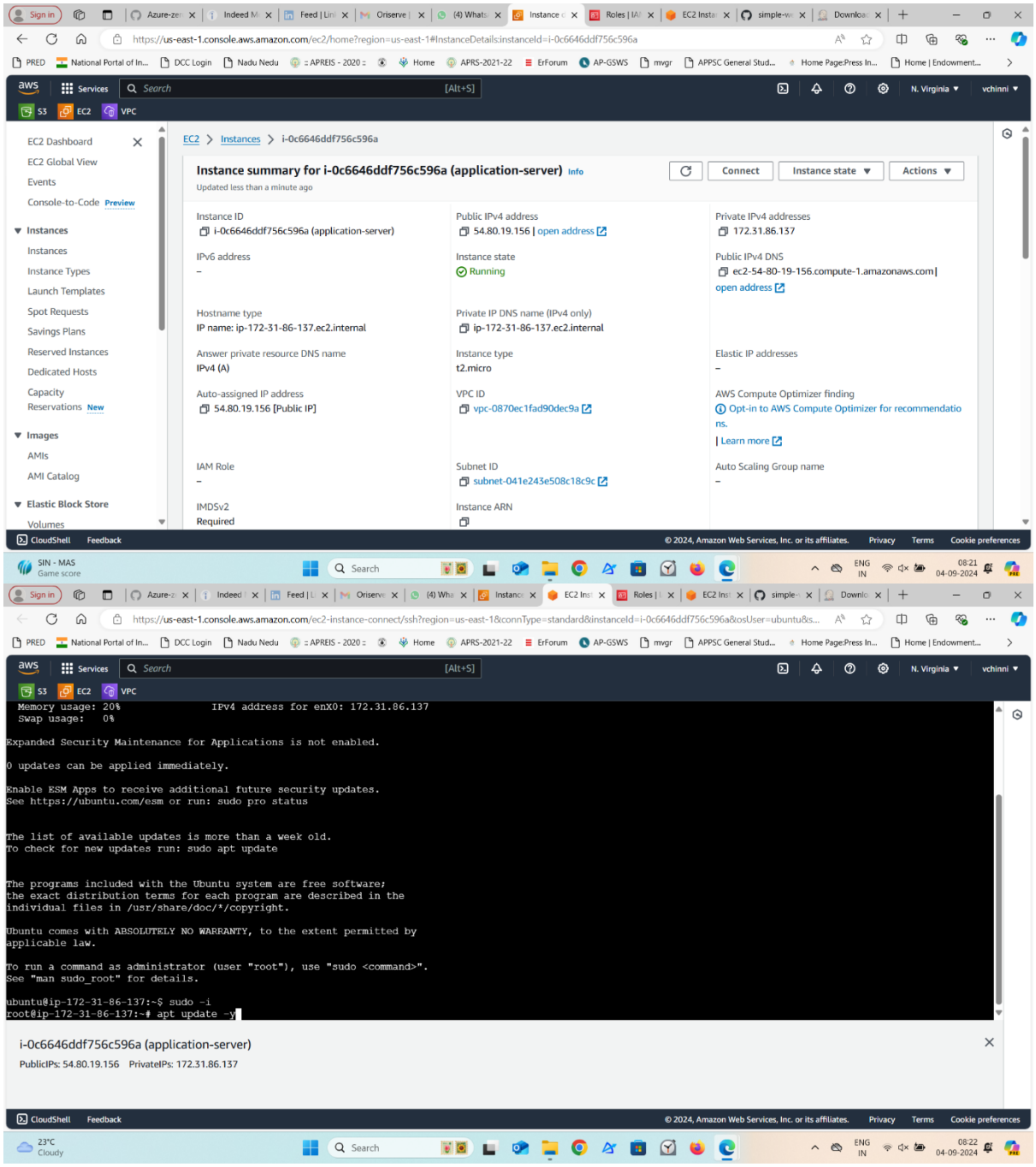


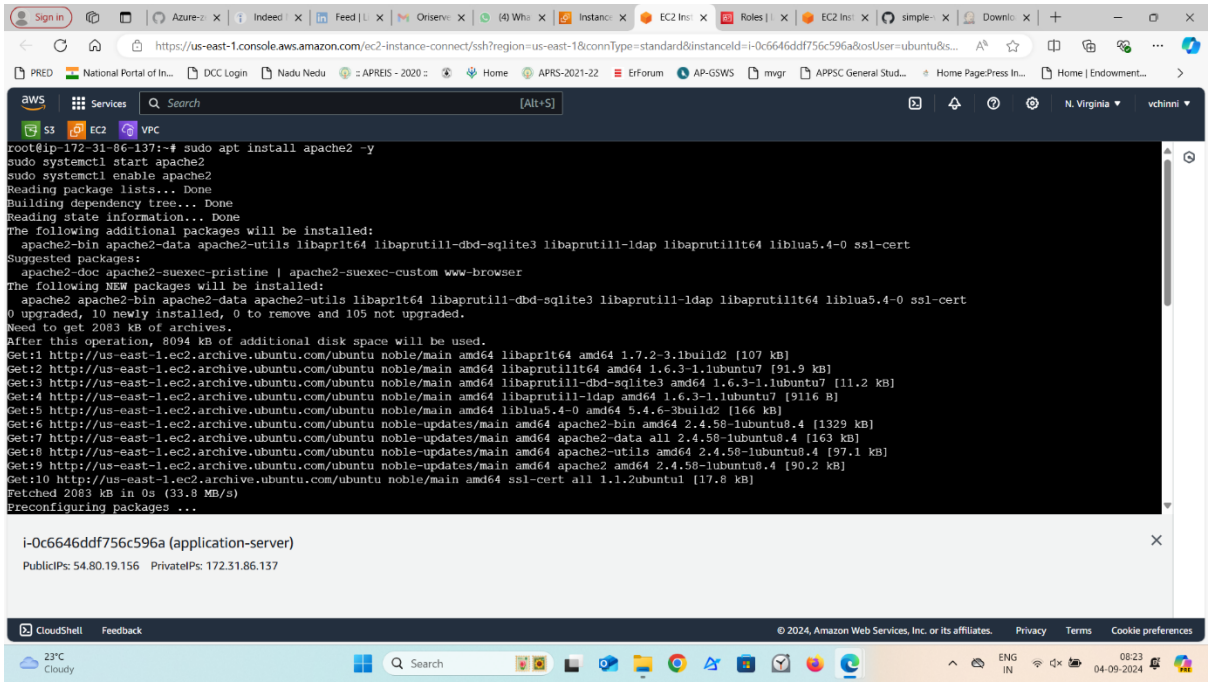
- Ensure the instance is within the same VPC and subnet as Jenkins for easier communication.
- Security group settings:
  - **SSH (port 22)** for access.
  - **HTTP (port 80)** for the application.



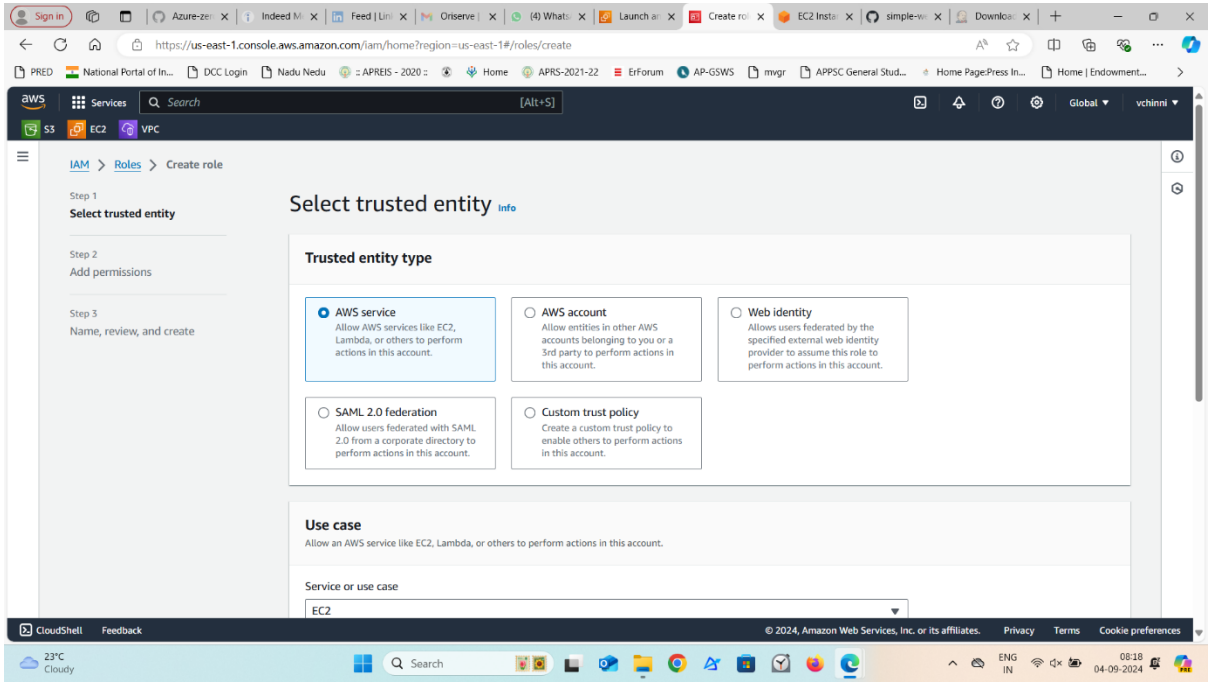
Step 5: Configure Application Server (For CodeDeploy and Scaling)

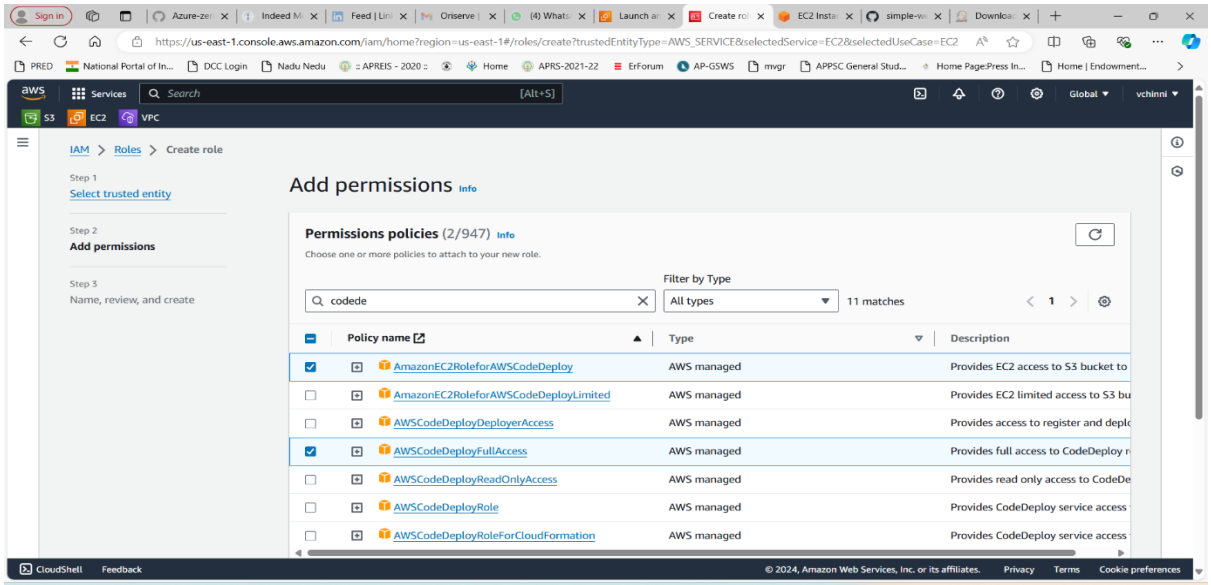
1. **Connect to the Application Server Instance:** click on connect
2. Install Apache and AWS CodeDeploy Agent
3. Attach IAM Role to Application Servers

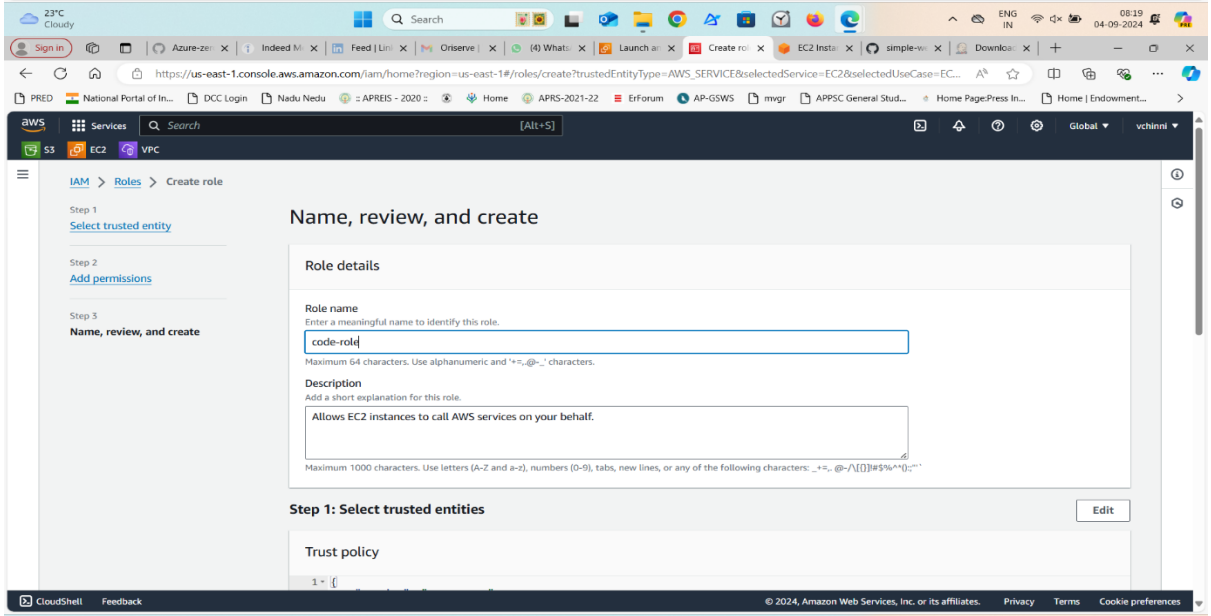


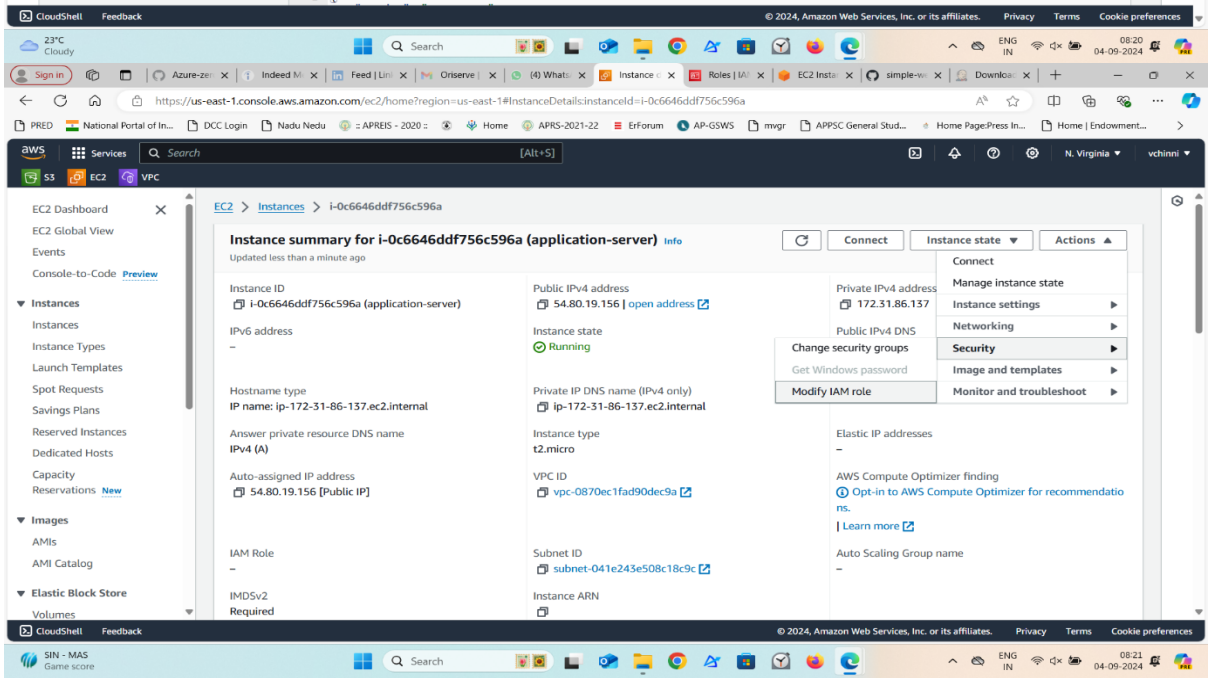


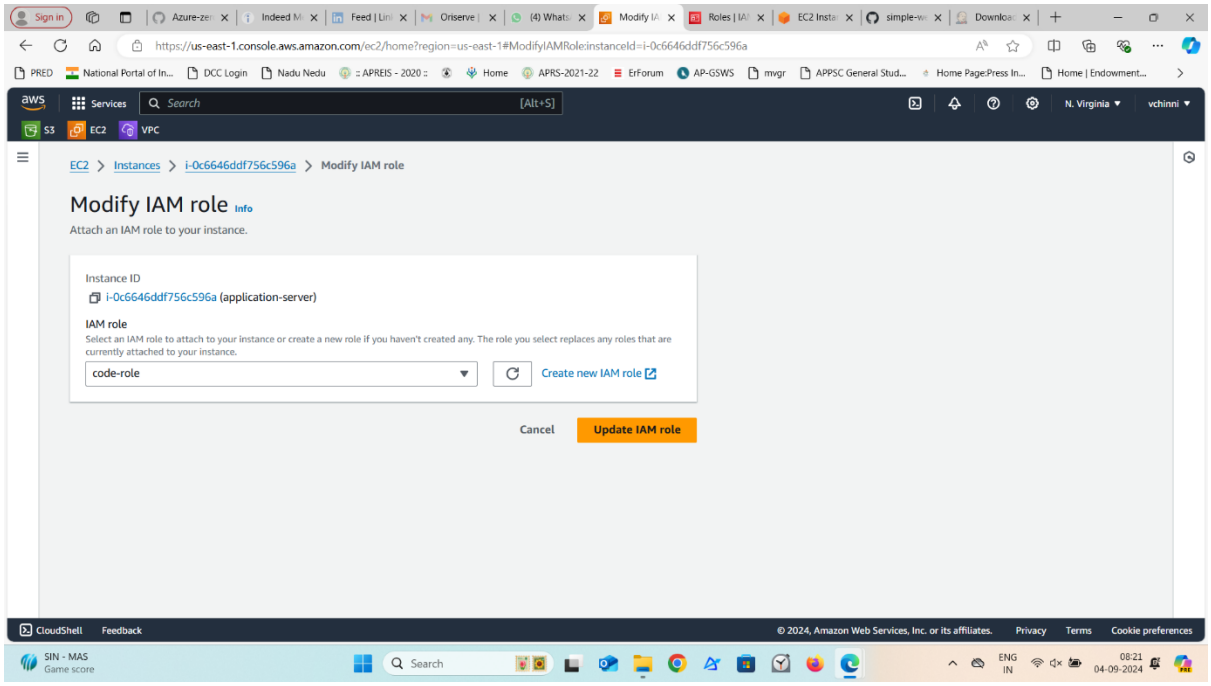
Ensure your EC2 instance has the AmazonEC2RoleforAWSCodeDeploy policy attached, allowing CodeDeploy access.




IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Add permissions Info
Permissions policies (2/947) Info
Choose one or more policies to attach to your new role.
codede
Filter by Type
All types
11 matches
Policy name
Type
Description
AmazonEC2RoleforAWSCodeDeploy
AWS managed
Provides EC2 access to S3 bucket to
AmazonEC2RoleforAWSCodeDeployLimited
AWS managed
Provides EC2 limited access to S3 bu
AWSCodeDeployDeployerAccess
AWS managed
Provides access to register and deplo
AWSCodeDeployFullAccess
AWS managed
Provides full access to CodeDeploy r
AWSCodeDeployReadOnlyAccess
AWS managed
Provides read only access to CodeDe
AWSCodeDeployRole
AWS managed
Provides CodeDeploy service access
AWSCodeDeployRoleForCloudFormation
AWS managed
Provides CodeDeploy service access
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Name, review, and create
Role details
Role name
Enter a meaningful name to identify this role.
code-role
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.
Description
Add a short explanation for this role.
Allows EC2 instances to call AWS services on your behalf.
Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,. @-/\[{}]!#$%^*();:"'`
Step 1: Select trusted entities
Edit
Trust policy
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


EC2 > Instances > i-0c6646ddf756c596a
Instance summary for i-0c6646ddf756c596a (application-server) Info
Updated less than a minute ago
Connect
Instance state
Actions
Connect
Manage instance state
Instance settings
Networking
Security
Image and templates
Monitor and troubleshoot
Change security groups
Get Windows password
Modify IAM role
Instance ID
i-0c6646ddf756c596a (application-server)
Public IPv4 address
54.80.19.156 | open address
Private IPv4 address
172.31.86.137
IPv6 address
–
Instance state
Running
Public IPv4 DNS
Hostname type
IP name: ip-172-31-86-137.ec2.internal
Private IP DNS name (IPv4 only)
ip-172-31-86-137.ec2.internal
Answer private resource DNS name
IPv4 (A)
Instance type
t2.micro
Elastic IP addresses
–
Auto-assigned IP address
54.80.19.156 [Public IP]
VPC ID
vpc-0870ec1fad90dec9a
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations.
Learn more
IAM Role
–
Subnet ID
subnet-041e243e508c18c9c
Auto Scaling Group name
–
IMDSv2
Required
Instance ARN
EC2 Dashboard
EC2 Global View
Events
Console-to-Code Preview
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



Step 6: Set Up Auto Scaling for Application Servers

Auto Scaling allows your application to automatically scale up or down based on demand, ensuring availability and performance while optimizing costs.

1. Create a Launch Template

A Launch Template defines the configuration of your EC2 instances, including AMI, instance type, security groups, and key settings. This template will be used to create instances in your Auto Scaling Group.

Steps to Create a Launch Template:

1. **Navigate to the EC2 Launch Templates:**
   - Go to the AWS EC2 Console.
   - On the left panel, click on **Launch Templates** under **Instances**.
2. **Create a New Launch Template:**
   - Click on **Create launch template**.
   - **Template Name**: Enter a descriptive name
   - **Template Version Description**: Optionally, add a version description.
3. **Configure the Template Details:**
   - **AMI (Amazon Machine Image)**: Choose the AMI used for your application servers, such as Ubuntu 20.04 LTS.
   - **Instance Type**: Select an instance type, such as t2.micro or another suitable type based on your needs.
   - **Key Pair**: Choose the key pair you use for SSH access.

- **Security Groups**: Select the security group that allows HTTP (port 80) and SSH (port 22).
- **IAM Instance Profile**: Choose the IAM role that has permissions for AWS CodeDeploy (AmazonEC2RoleforAWSCodeDeploy).

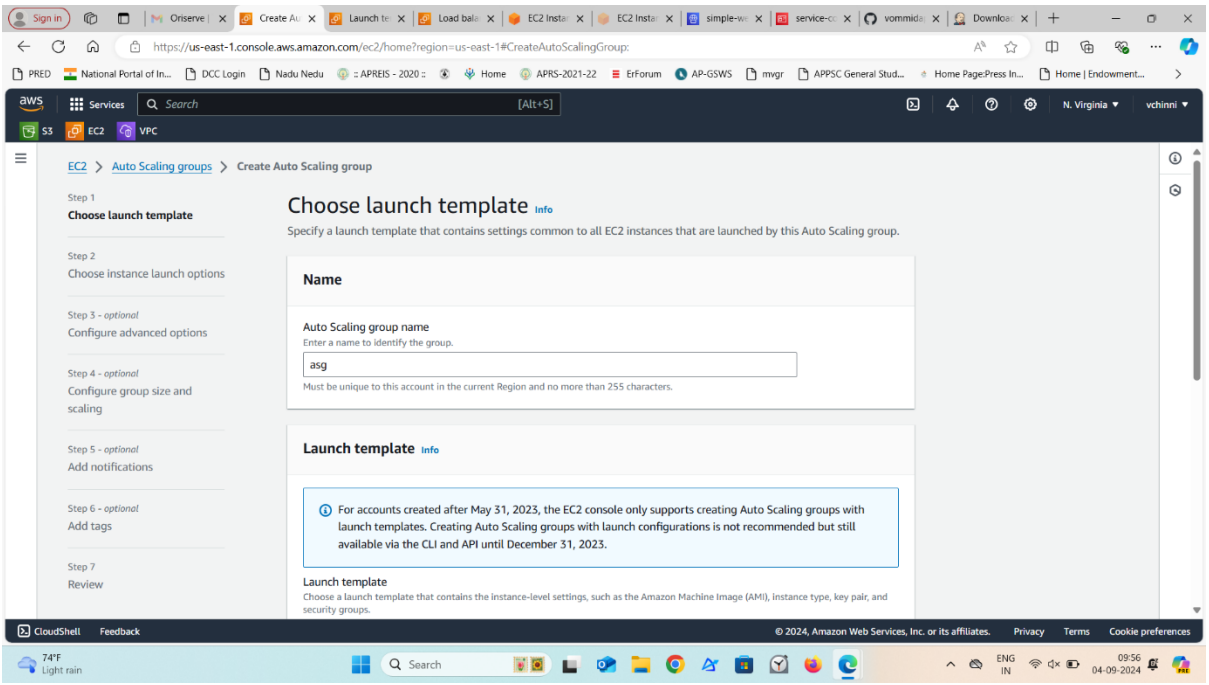
4. **Create the Template:**
   - Review the settings and click **Create launch template**.

Step 7: Create an Auto Scaling Group

The Auto Scaling Group uses the Launch Template to automatically manage the number of instances based on defined scaling policies.

Steps to Create an Auto Scaling Group:

1. **Navigate to Auto Scaling Groups:**
   - Go to Auto Scaling Groups in the AWS EC2 Console.
2. **Create Auto Scaling Group:**
   - Click on **Create Auto Scaling group**.
3. **Configure Group Details:**
   - **Auto Scaling Group Name**: Enter a name
   - **Launch Template**: Select the Launch Template you created



4. **Configure Network:**

- **VPC**: Choose the same VPC where your application servers will run.
- **Subnets**: Select the subnets where the instances should be launched.

5. **Configure Scaling Policies:**
   - **Desired Capacity**: Set the initial number of instances
   - **Minimum Capacity**: Set the minimum number of instances
   - **Maximum Capacity**: Set the maximum number of instances
6. **Add Scaling Policies:**
   - **Target Tracking Scaling Policy**: Add a scaling policy based on CPU utilization or another metric.
   - For example, set the scaling policy to maintain the average CPU utilization at 50%. The Auto Scaling Group will automatically add or remove instances to maintain this target.
7. **Create the Auto Scaling Group:**
   - Review all settings and click **Create Auto Scaling group**.

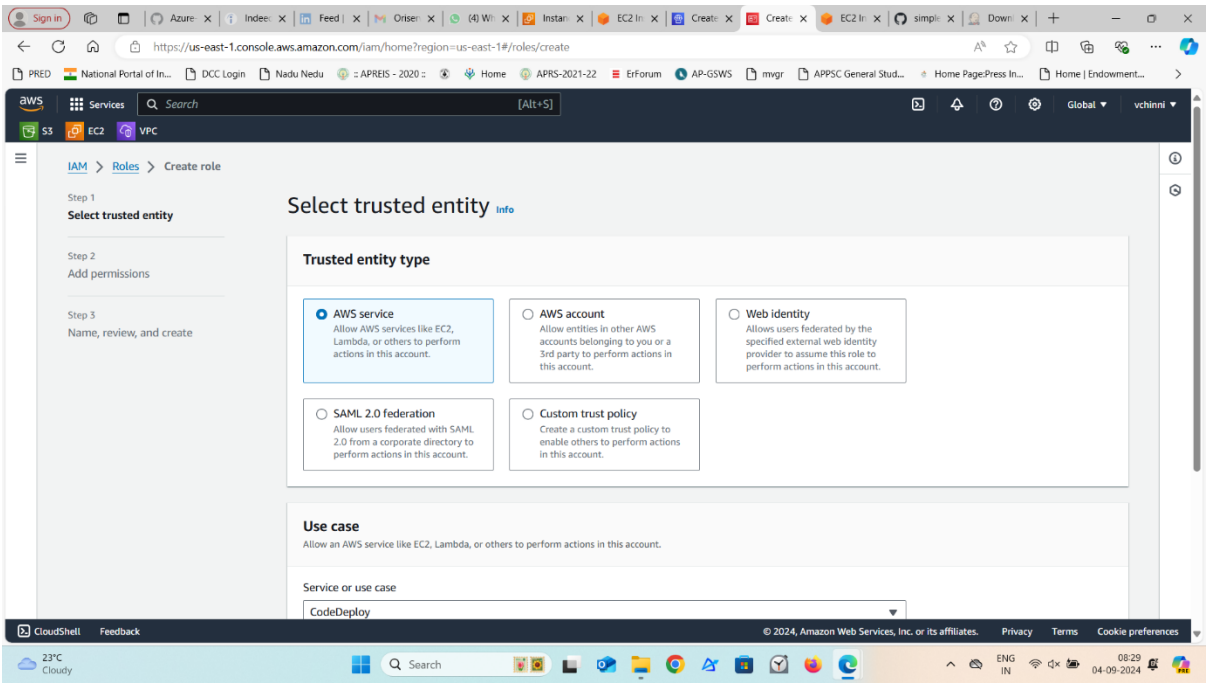
Step 8: Create or Select an IAM Role for CodeDeploy

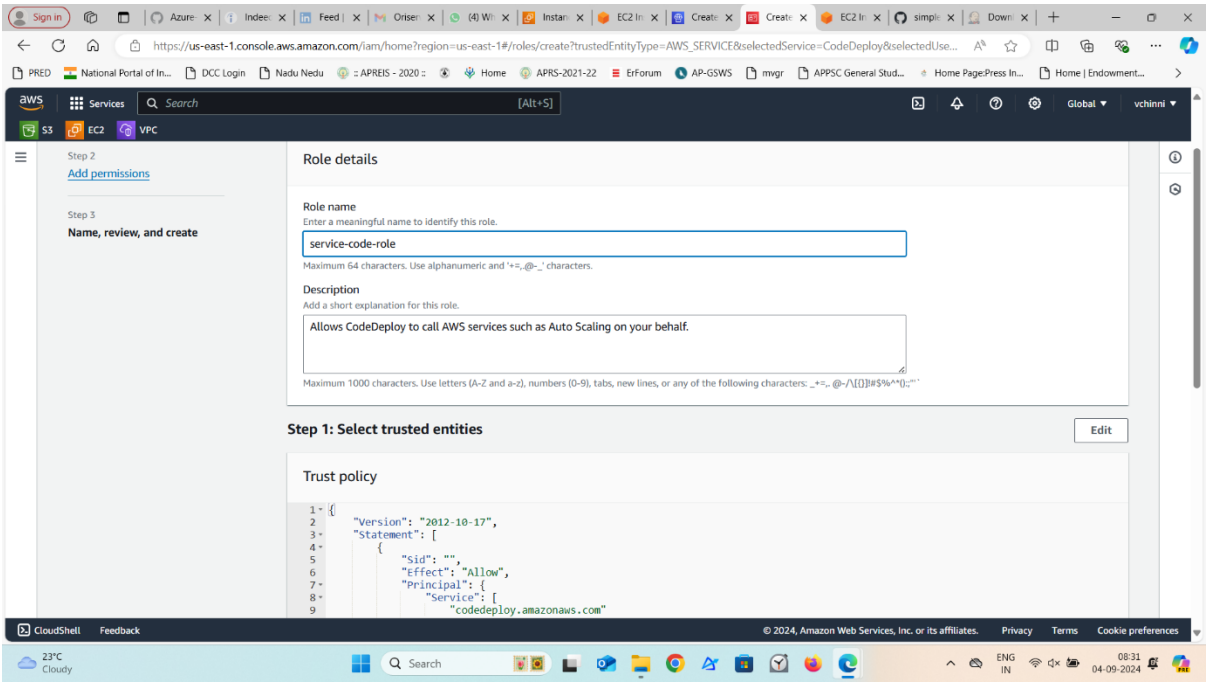
Creating a New IAM Role

1. **Navigate to the IAM Console:**

Go to the IAM Management Console.

2. **Create a New Role:**
   - Click on **"Roles"** in the sidebar.
   - Click **"Create role"**.





3.
4. **Select Trusted Entity Type:**
    - Choose **"AWS service"** as the trusted entity type.
    - Select **"CodeDeploy"** from the list of services.
5. **Attach Permissions Policies:**
    - Attach the **"AWSCodeDeployRole"** policy, which provides the necessary permissions for CodeDeploy. You can also attach other policies as needed, such as AmazonEC2ReadOnlyAccess if you need read-only access to EC2 resources.
6. **Review and Create Role:**
    - Click **"Next: Tags"** (optional) to add any tags if needed.
    - Click **"Next: Review"**.
    - Enter a name for the role (e.g., CodeDeployRole).
    - Click **"Create role"**.
7. **Attach the Role to CodeDeploy:**
    - Return to the CodeDeploy Console.
    - Navigate to your application and click **"Create deployment group"**.
    - In the **"Service Role"** field, select the role you just created (e.g., CodeDeployRole).

Selecting an Existing IAM Role

If you already have a suitable IAM role:

1. **Navigate to the CodeDeploy Console:**

Go to the CodeDeploy Console.

2. **Create Deployment Group:**
    - Click on the application you created (e.g., simple-web-app).
    - Click **"Create deployment group"**.
3. **Select Existing Role:**
    - In the **"Service Role"** dropdown, select the IAM role that has the required permissions for CodeDeploy. Ensure that this role has the **AWSCodeDeployRole** policy attached, and any additional policies needed to interact with your EC2 instances and other AWS services.
4. **Complete the Configuration:**
    - Continue with the remaining configuration for your deployment group, such as choosing the deployment type and environment configuration.
    - Click **"Create deployment group"** to finish setting up.

Step 9:

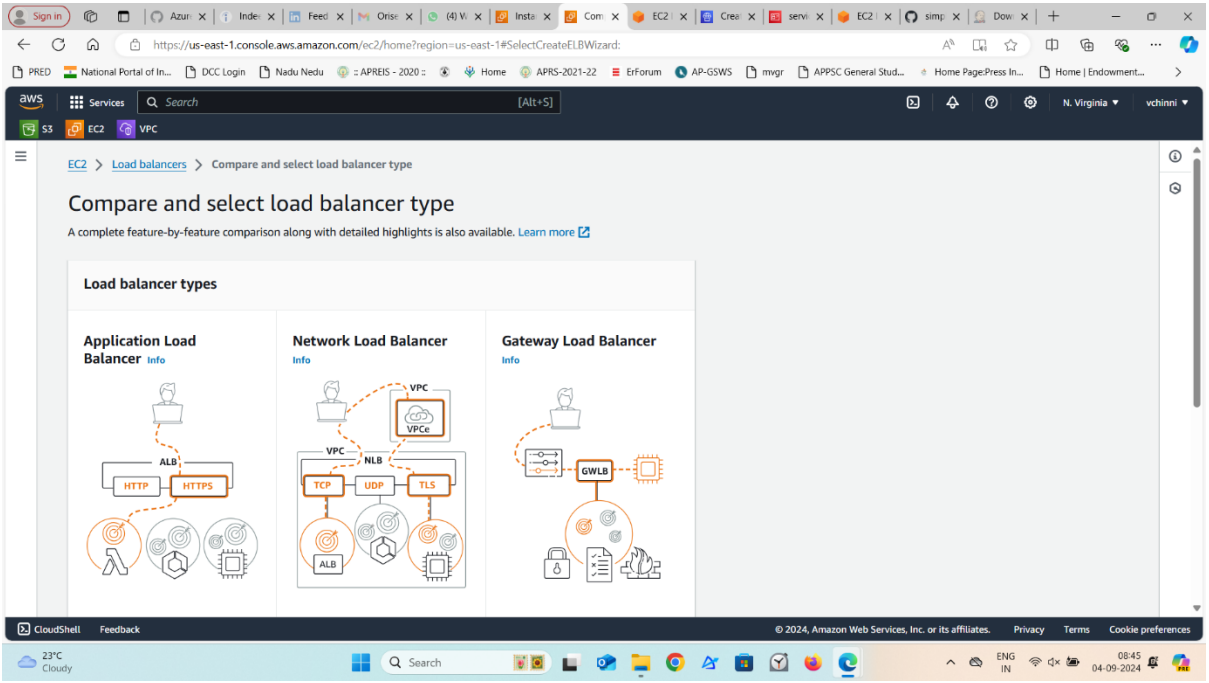
1. Create an Application Load Balancer (ALB)

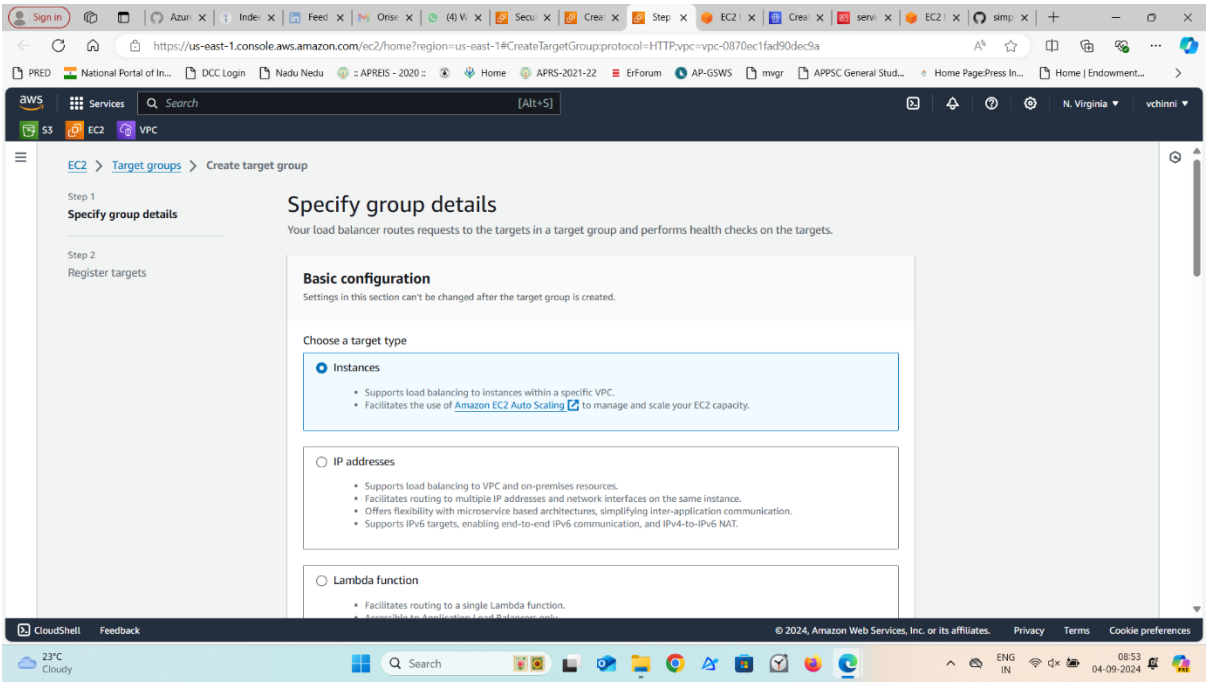
1.1 Navigate to the Load Balancer Console:

- Go to the EC2 Console.
- In the left-hand menu, under **"Load Balancing"**, click **"Load balancers"**.

1.2 Create a New Load Balancer:

- Click **"Create Load Balancer"**.
- Choose **"Application Load Balancer"** (recommended for HTTP/HTTPS traffic).





1.3 Configure the Load Balancer:

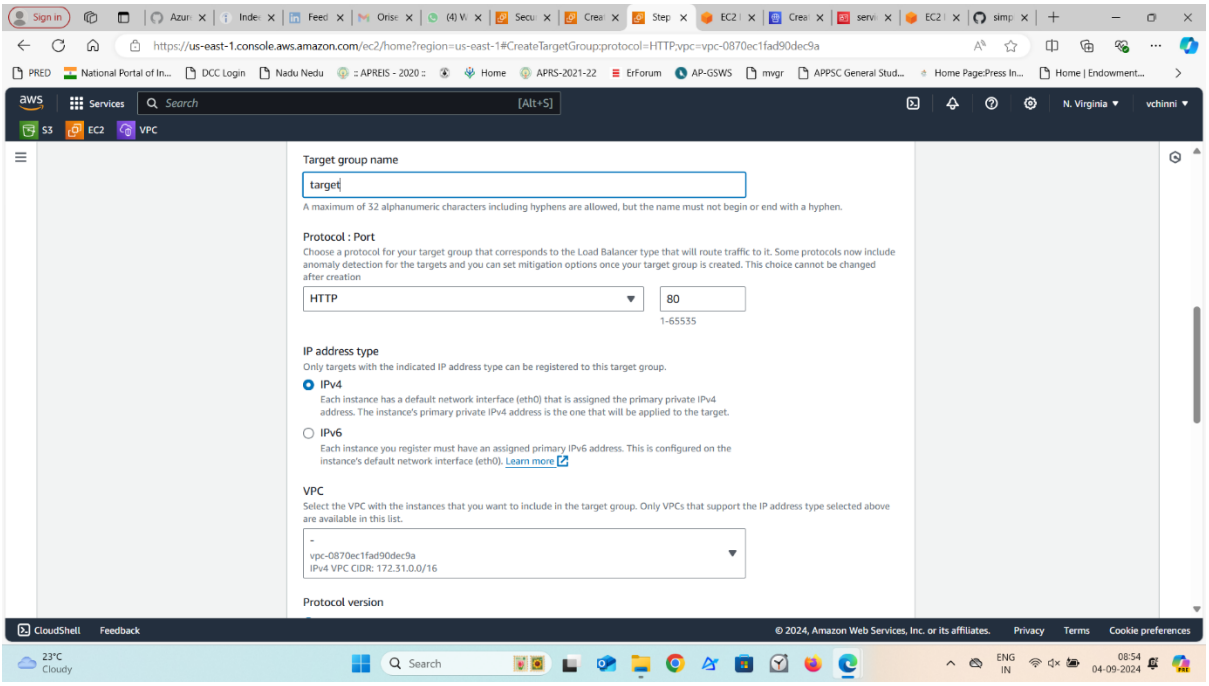
- **Basic Configuration:**
  - **Name:** Enter a name for your load balancer.
  - **Scheme:** Choose "Internet-facing" if your application is public or "Internal" for internal use.
  - **IP address type:** Choose "ipv4".
- **Listeners:** Configure listeners (usually HTTP on port 80 or HTTPS on port 443).
- **Availability Zones:** Select the Availability Zones where your EC2 instances are located.

1.4 Configure Security Settings (for HTTPS):

- **Choose a Security Policy:** If using HTTPS, select an existing SSL certificate or create a new one.
- **Configure Security Groups:** Ensure the security group allows traffic on the ports you specified (e.g., port 80 for HTTP, port 443 for HTTPS).

1.5 Configure Routing:

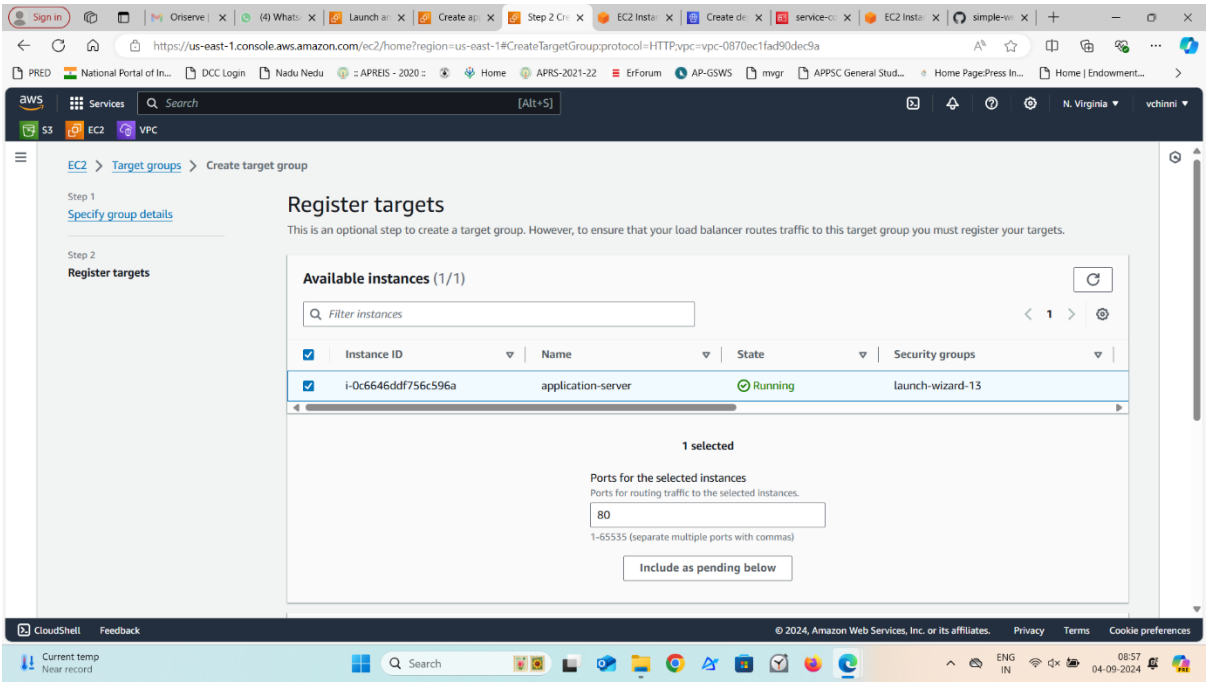
- **Create a Target Group:**
  - **Target Group Name:** Enter a name for your target group.
  - **Target Type:** Choose "Instance" (if your targets are EC2 instances).

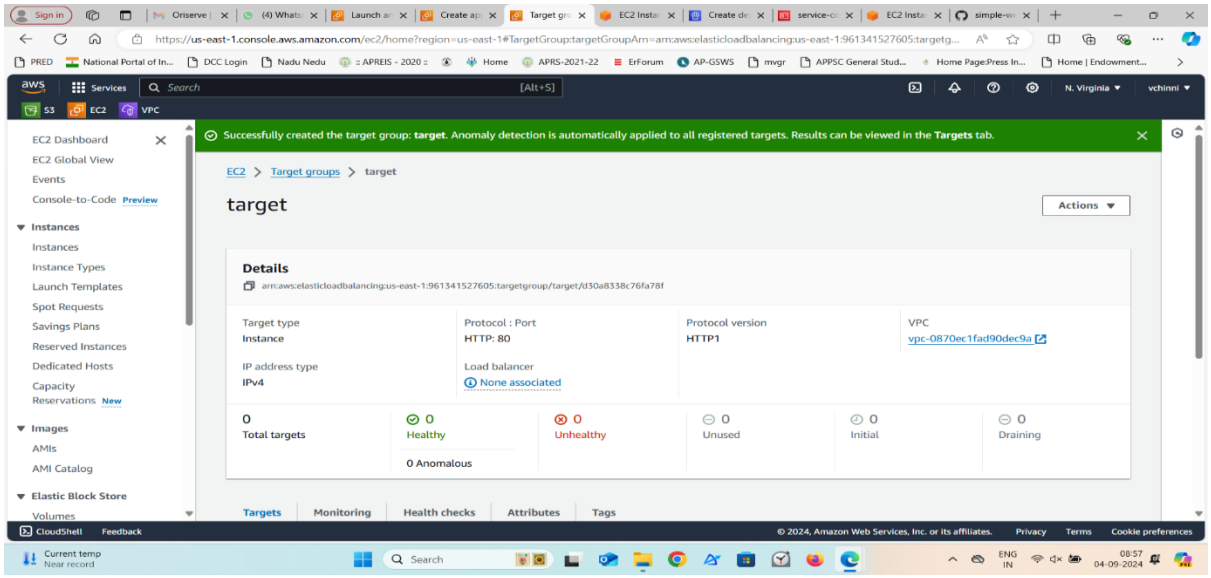


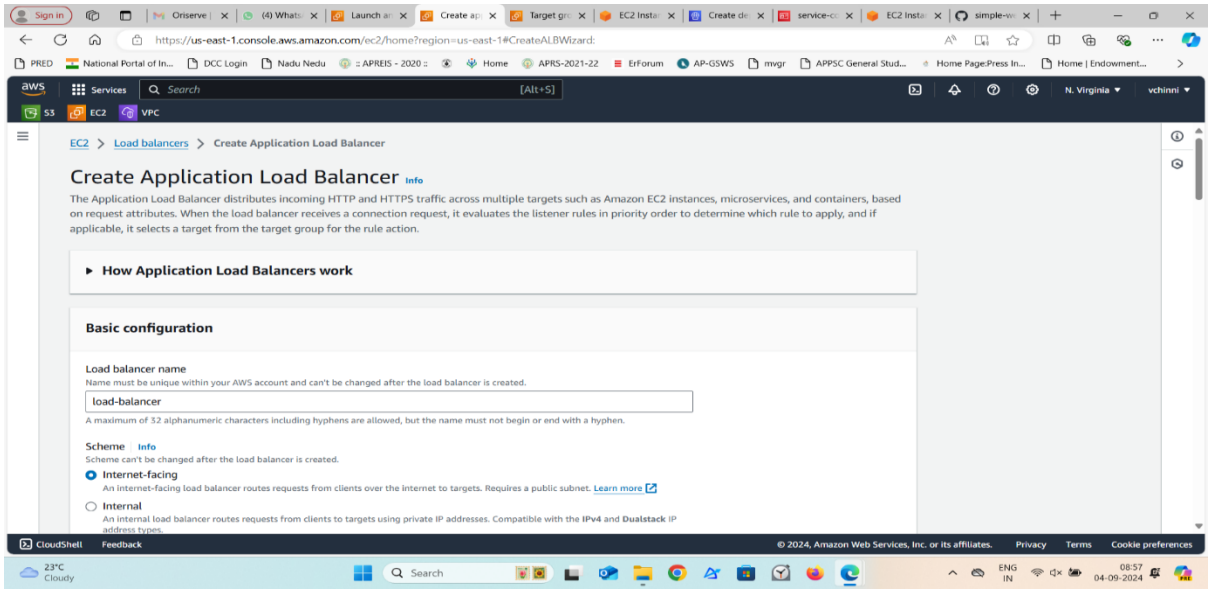
-
    - **Protocol:** Choose HTTP or HTTPS, depending on your setup.
    - **Port:** Enter the port where your application is running on the EC2 instances (e.g., port 80).

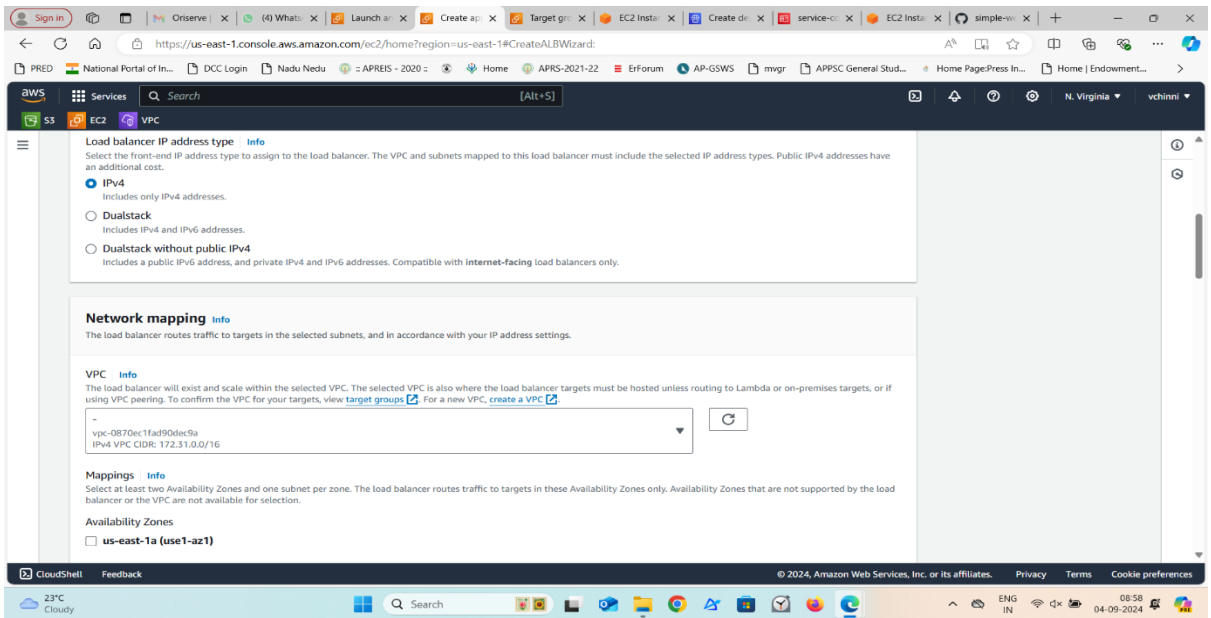
1.6 Register Targets:

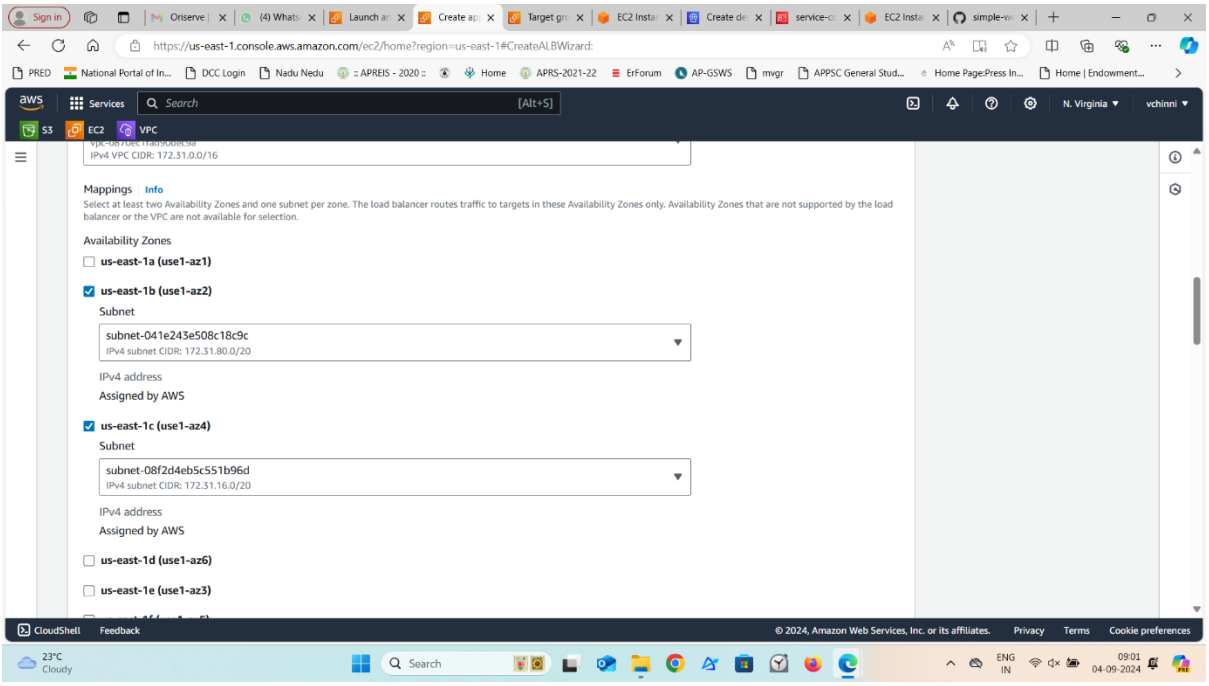
- Register the EC2 instances you want to route traffic to.

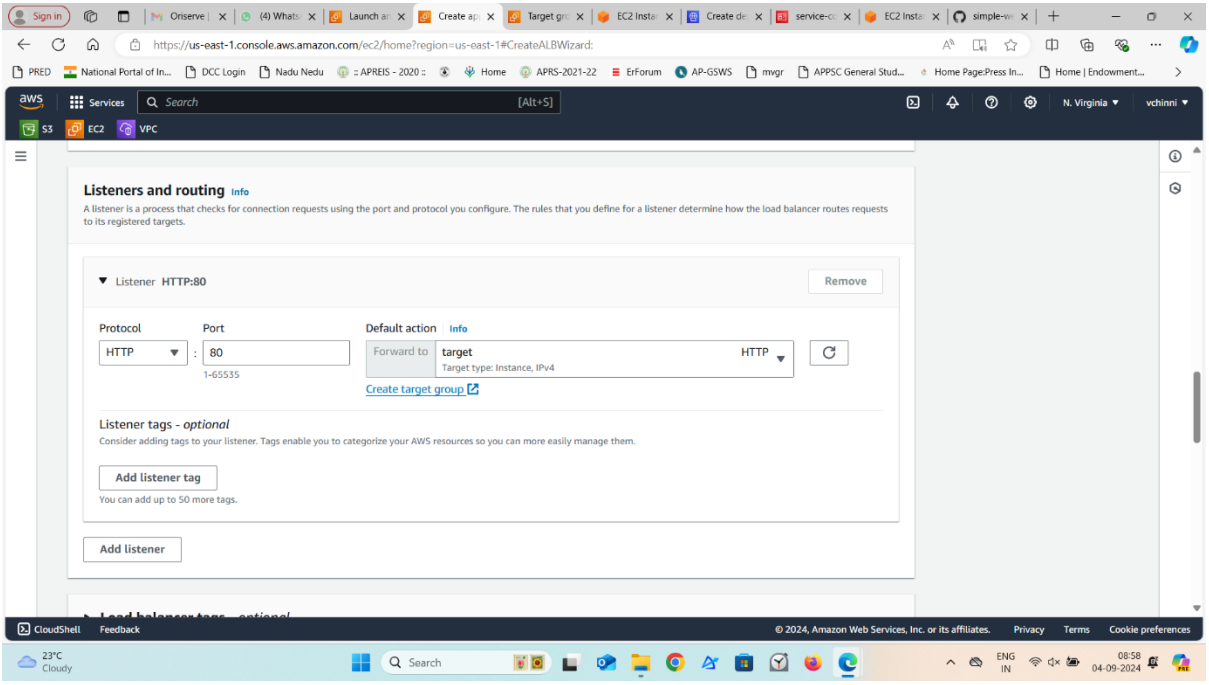



Successfully created the target group: target. Anomaly detection is automatically applied to all registered targets. Results can be viewed in the Targets tab.
EC2 > Target groups > target
target
Actions
Details
arn:aws:elasticloadbalancing:us-east-1:961341527605:targetgroup/target/d30a8338c76fa78f
Target type
Instance
Protocol : Port
HTTP: 80
Protocol version
HTTP1
VPC
vpc-0870ec1fad90dec9a
IP address type
IPv4
Load balancer
None associated
0
Total targets
0
Healthy
0 Anomalous
0
Unhealthy
0
Unused
0
Initial
0
Draining
Targets
Monitoring
Health checks
Attributes
Tags


EC2 > Load balancers > Create Application Load Balancer
Create Application Load Balancer Info
The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, microservices, and containers, based on request attributes. When the load balancer receives a connection request, it evaluates the listener rules in priority order to determine which rule to apply, and if applicable, it selects a target from the target group for the rule action.
How Application Load Balancers work
Basic configuration
Load balancer name
Name must be unique within your AWS account and can't be changed after the load balancer is created.
load-balancer
A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.
Scheme Info
Scheme can't be changed after the load balancer is created.
Internet-facing
An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. Learn more
Internal
An internal load balancer routes requests from clients to targets using private IP addresses. Compatible with the IPv4 and Dualstack IP address types.


Load balancer IP address type Info
Select the front-end IP address type to assign to the load balancer. The VPC and subnets mapped to this load balancer must include the selected IP address types. Public IPv4 addresses have an additional cost.
IPv4
Includes only IPv4 addresses.
Dualstack
Includes IPv4 and IPv6 addresses.
Dualstack without public IPv4
Includes a public IPv6 address, and private IPv4 and IPv6 addresses. Compatible with internet-facing load balancers only.
Network mapping Info
The load balancer routes traffic to targets in the selected subnets, and in accordance with your IP address settings.
VPC Info
The load balancer will exist and scale within the selected VPC. The selected VPC is also where the load balancer targets must be hosted unless routing to Lambda or on-premises targets, or if using VPC peering. To confirm the VPC for your targets, view target groups. For a new VPC, create a VPC.
-
vpc-0870ec1fad90dec9a
IPv4 VPC CIDR: 172.31.0.0/16
Mappings Info
Select at least two Availability Zones and one subnet per zone. The load balancer routes traffic to targets in these Availability Zones only. Availability Zones that are not supported by the load balancer or the VPC are not available for selection.
Availability Zones
us-east-1a (use1-az1)





1.7 Review and Create:

- Review your configuration and click **"Create"**.

2. Create a Target Group

2.1 Navigate to the Target Groups Console:

- Go to the EC2 Console.

- In the left-hand menu, under **"Load Balancing"**, click **"Target Groups"**.

2.2 Create a New Target Group:

- Click **"Create target group"**.
- **Target Type:** Choose "Instance".
- **Protocol:** Choose HTTP or HTTPS.
- **Port:** Enter the port where your application is running (e.g., port 80).

2.3 Configure Health Checks:

- Configure health checks based on the endpoint that determines the health of your instances.

2.4 Register Targets:

- Register your EC2 instances with the target group.

2.5 Review and Create:

- Review your configuration and click **"Create"**.

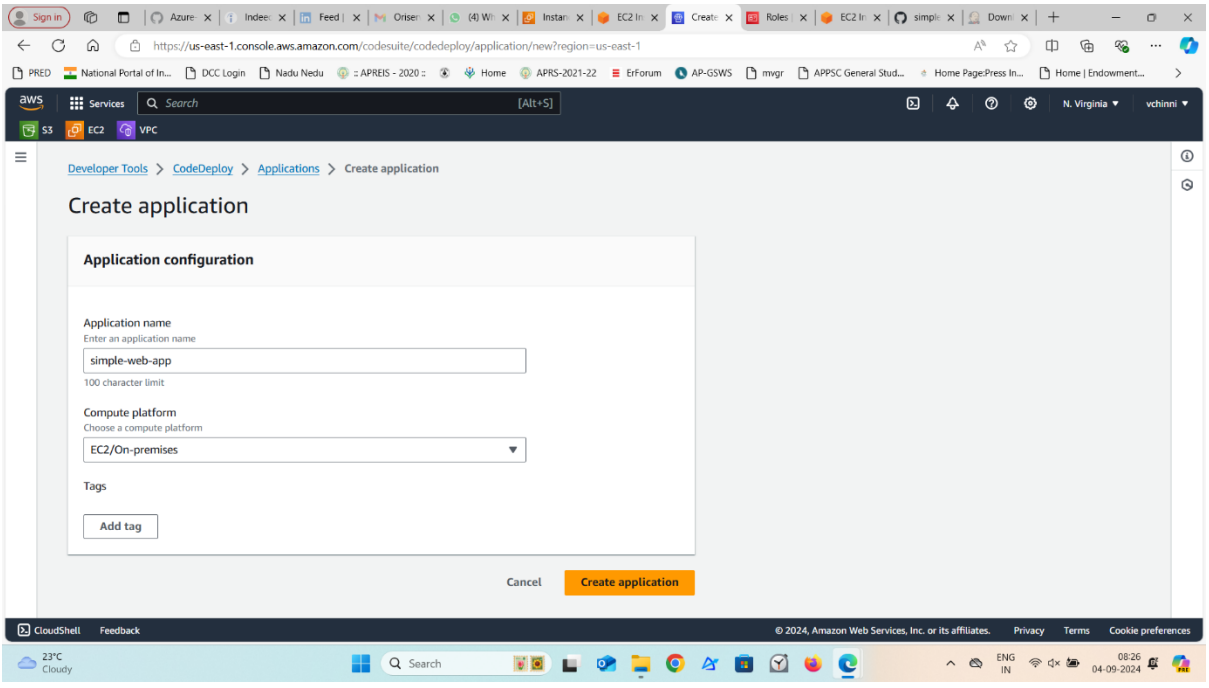
3. Integrate Load Balancer and Target Group with CodeDeploy

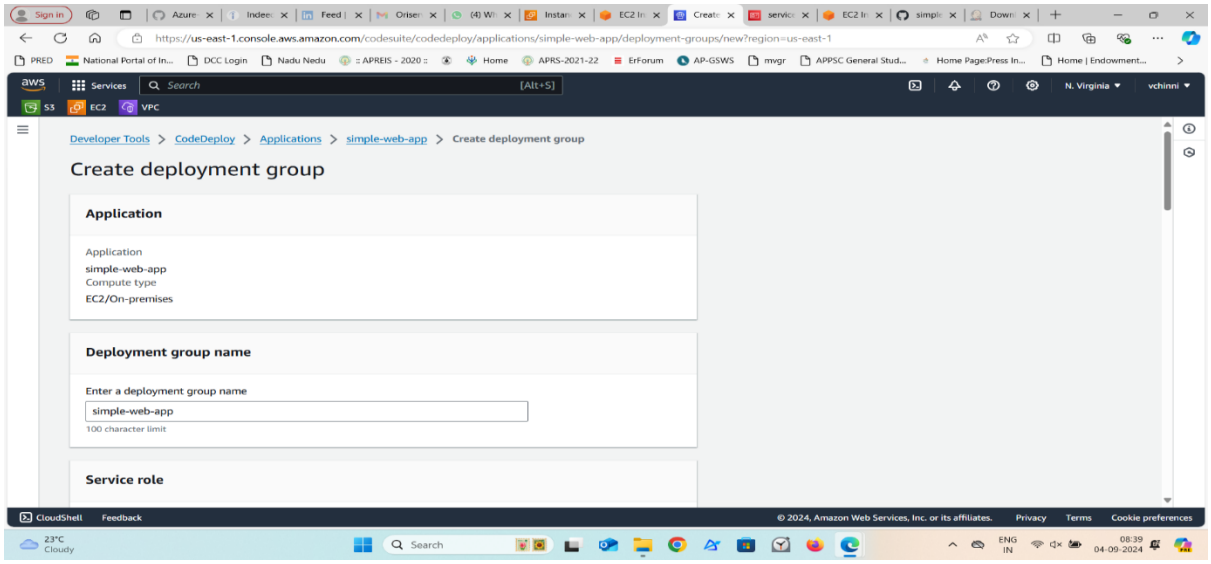
3.1 Navigate to CodeDeploy Console:

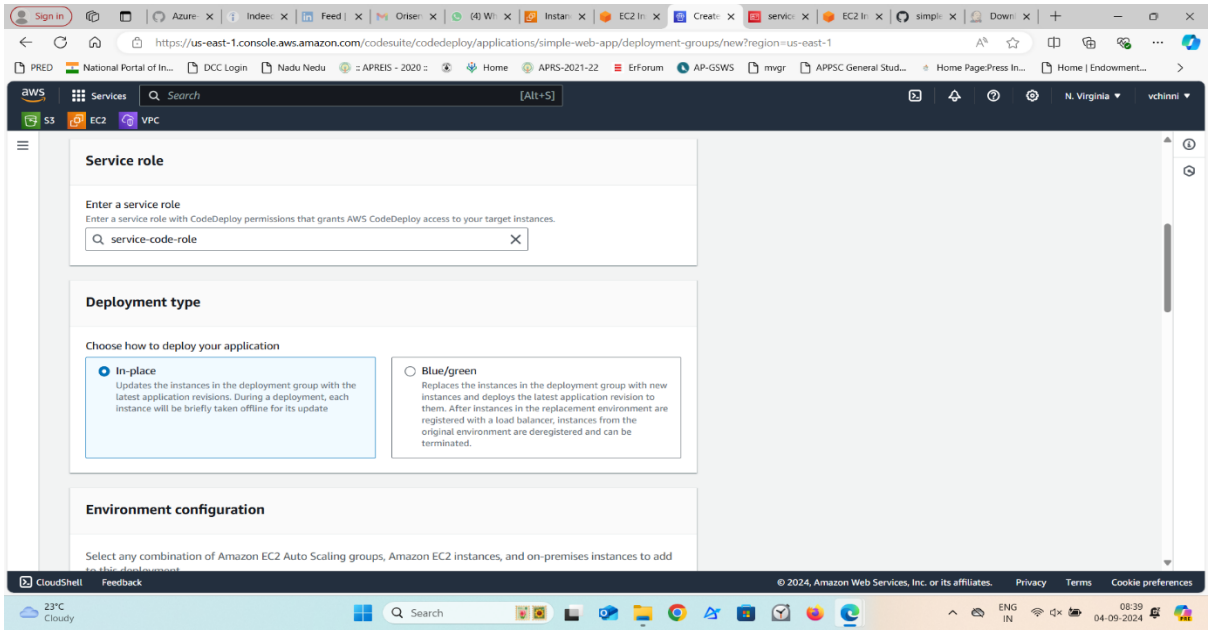
- Go to the CodeDeploy Console.

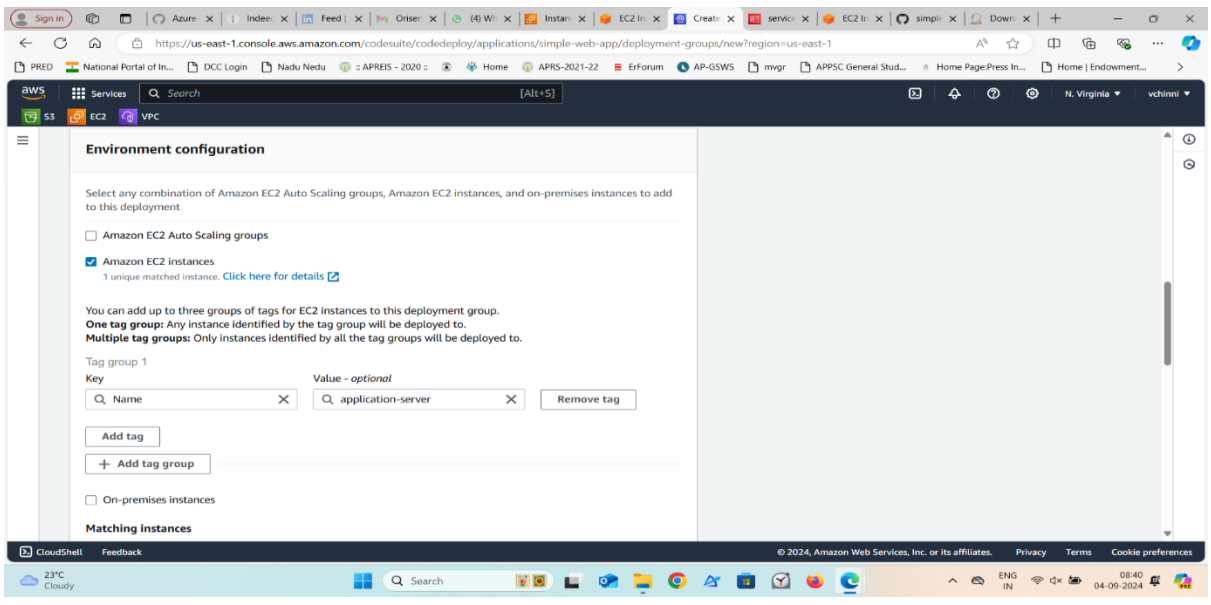
3.2 Edit/Create Deployment Group:

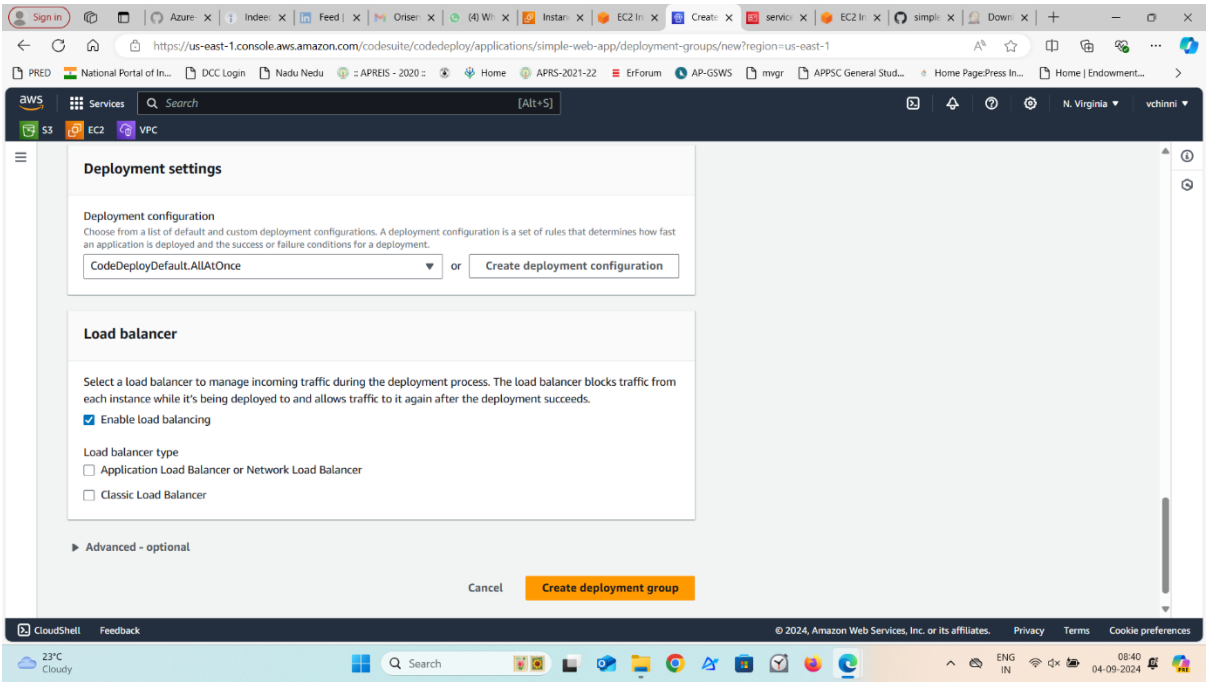
- Select your application (e.g., simple-web-app).

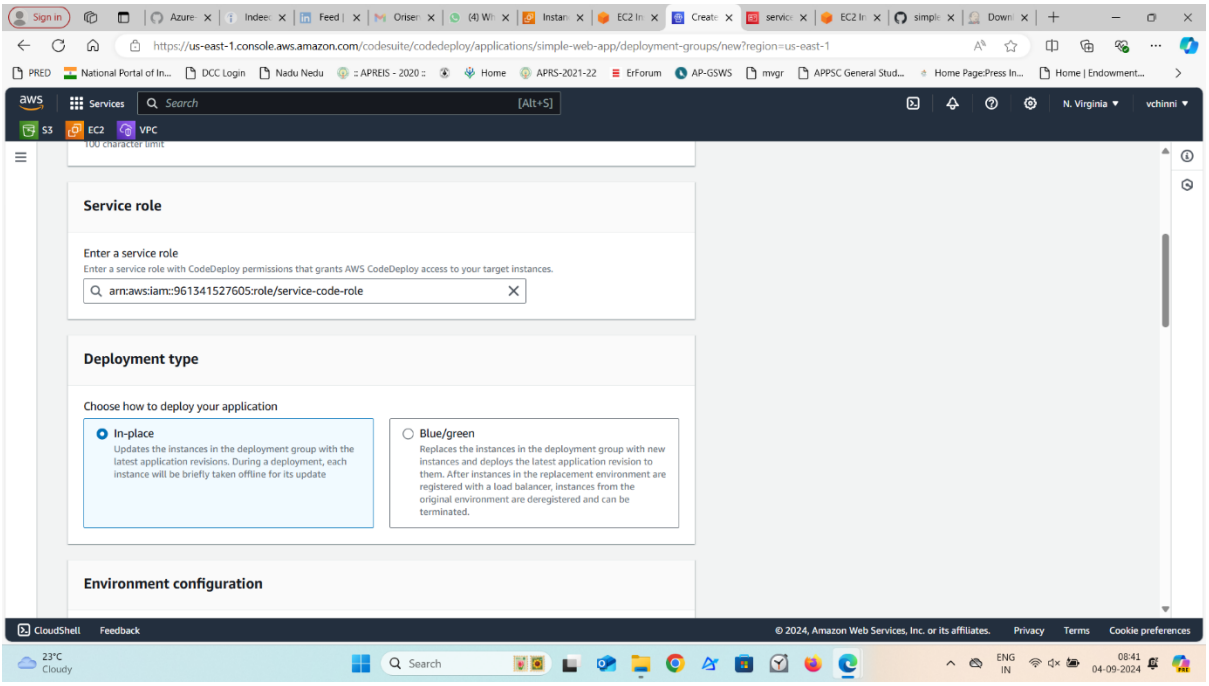



Developer Tools > CodeDeploy > Applications > simple-web-app > Create deployment group
Create deployment group
Application
Application
simple-web-app
Compute type
EC2/On-premises
Deployment group name
Enter a deployment group name
simple-web-app
100 character limit
Service role


Service role
Enter a service role
Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.
service-code-role
Deployment type
Choose how to deploy your application
In-place
Updates the instances in the deployment group with the latest application revisions. During a deployment, each instance will be briefly taken offline for its update
Blue/green
Replaces the instances in the deployment group with new instances and deploys the latest application revision to them. After instances in the replacement environment are registered with a load balancer, instances from the original environment are deregistered and can be terminated.
Environment configuration
Select any combination of Amazon EC2 Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add


Environment configuration
Select any combination of Amazon EC2 Auto Scaling groups, Amazon EC2 instances, and on-premises instances to add to this deployment
Amazon EC2 Auto Scaling groups
Amazon EC2 instances
1 unique matched instance. Click here for details
You can add up to three groups of tags for EC2 instances to this deployment group.
One tag group: Any instance identified by the tag group will be deployed to.
Multiple tag groups: Only instances identified by all the tag groups will be deployed to.
Tag group 1
Key
Value - optional
Name
application-server
Remove tag
Add tag
Add tag group
On-premises instances
Matching instances

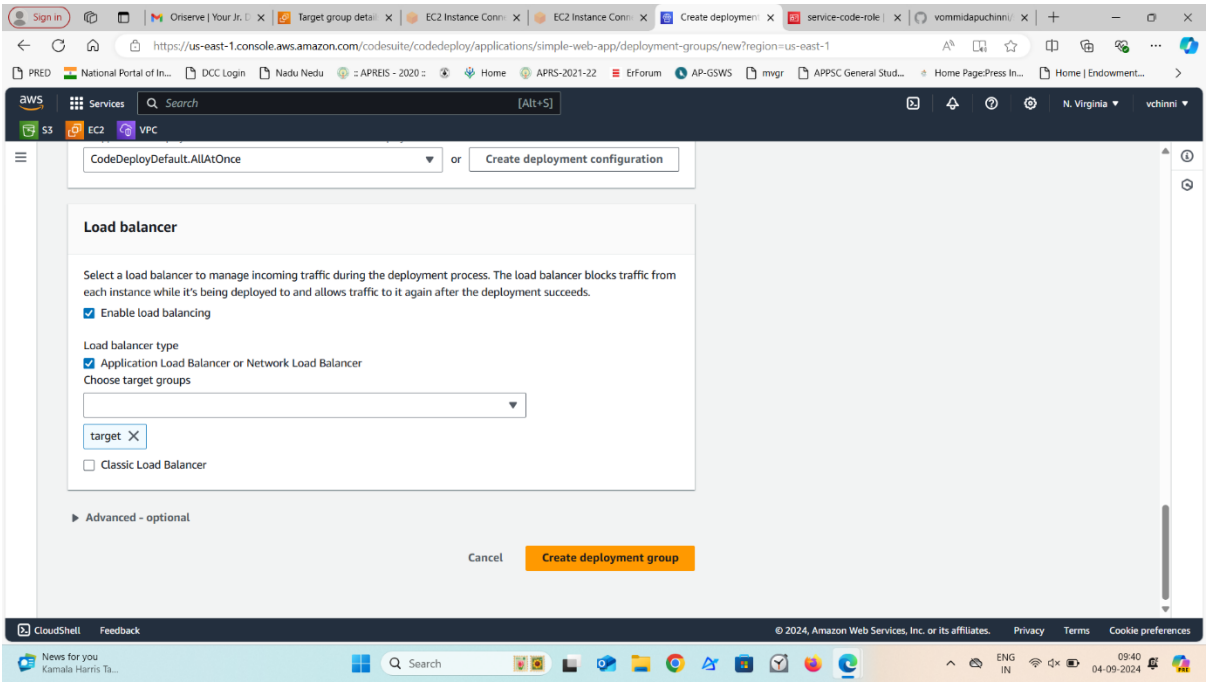




- Click **"Create deployment group"** or select an existing one to edit.

3.3 Configure Load Balancer:

- In the **"Load Balancer"** section:
  - **Load Balancer Type:** Choose "Application Load Balancer" (or "Network Load Balancer" if that's what you set up).
  - **Load Balancer:** Select the load balancer you created.
  - **Target Group:** Select the target group associated with the load balancer.



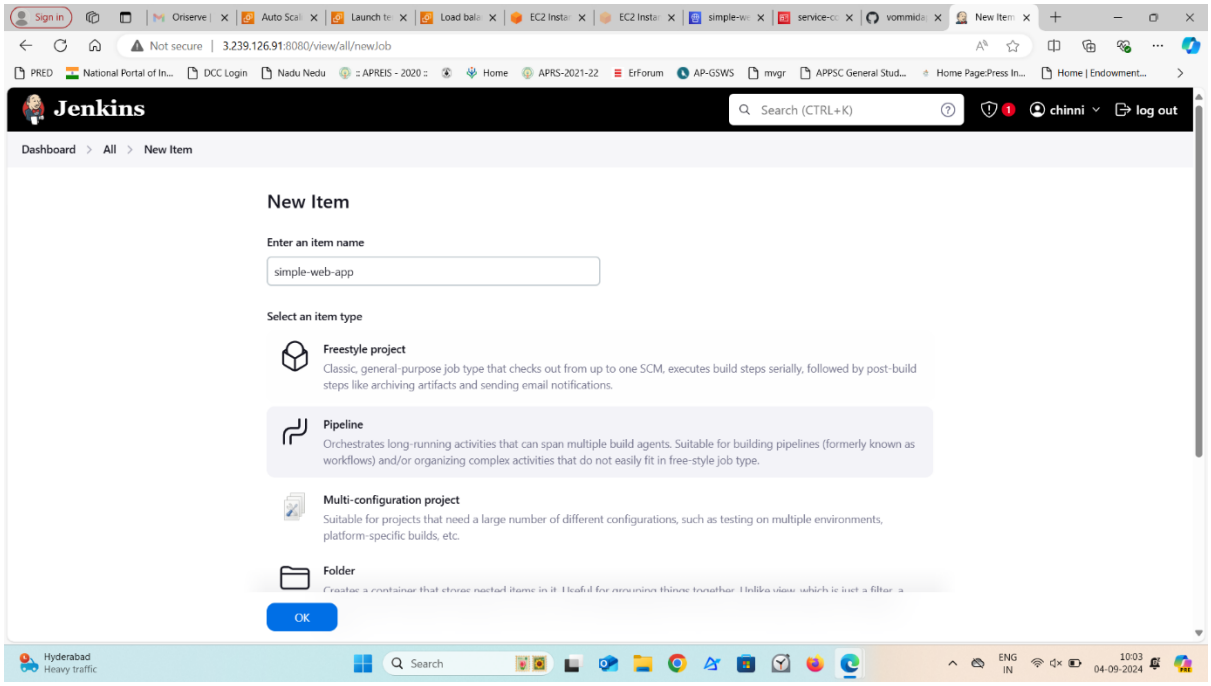
3.4 Save Configuration:

- Save your deployment group configuration.

Step 10: Configure the Jenkins Pipeline

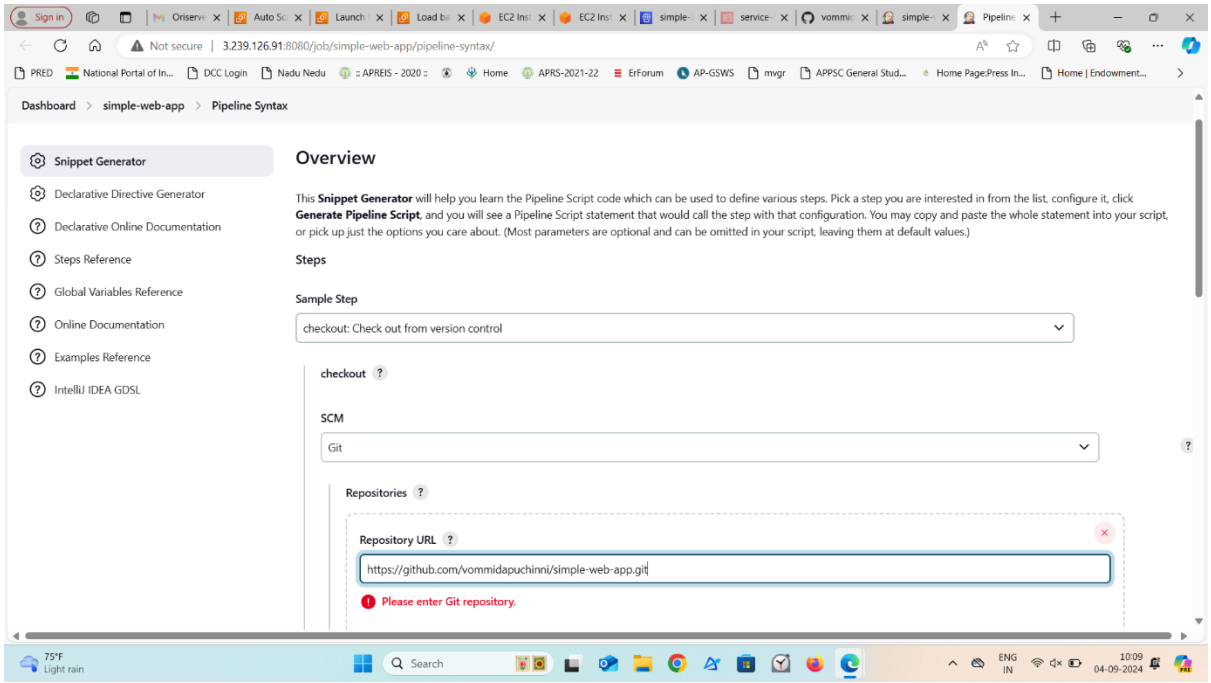
1. Create a New Pipeline Job in Jenkins:

1. **Access Jenkins Dashboard**:
    - Log into your Jenkins server.
    - Click on "**New Item**" in the left sidebar.
2. **Create a New Pipeline Job**:
    - Enter the name: simple-web-app-pipeline.
    - Select "**Pipeline**" as the job type.
    - Click **OK**.

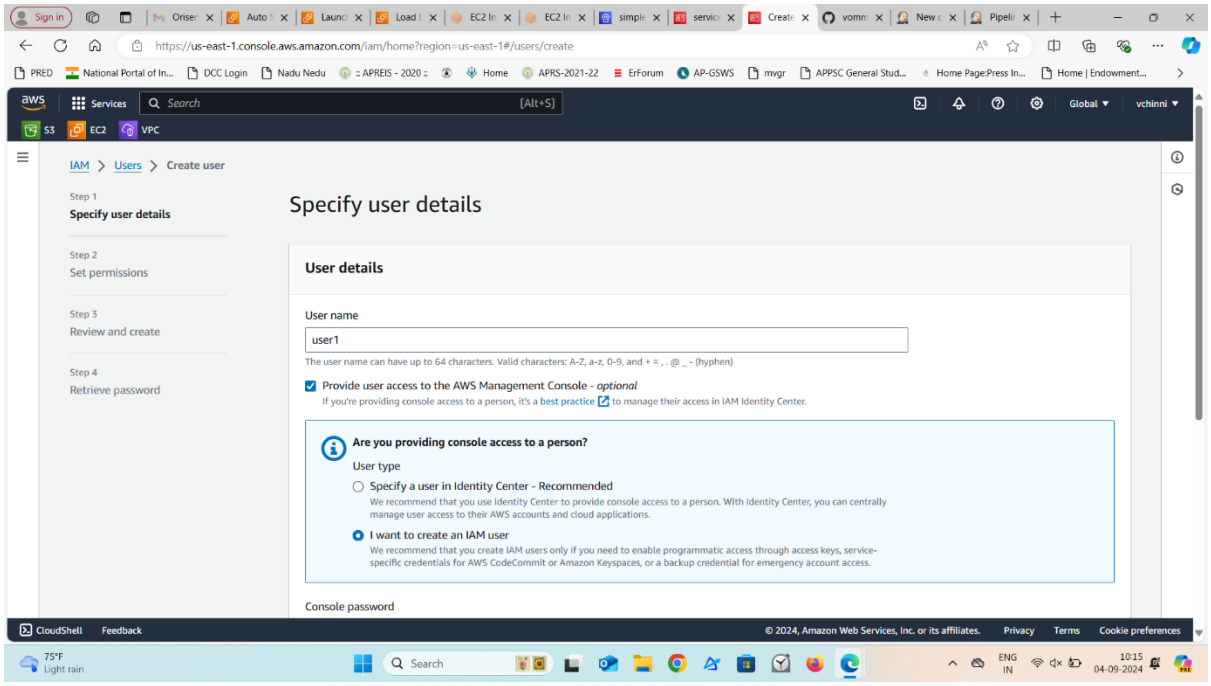


2. Create the Jenkinsfile:

1. **Add a Jenkinsfile to Your GitHub Repository**:
    - Create a file named Jenkinsfile in the root of your GitHub repository.

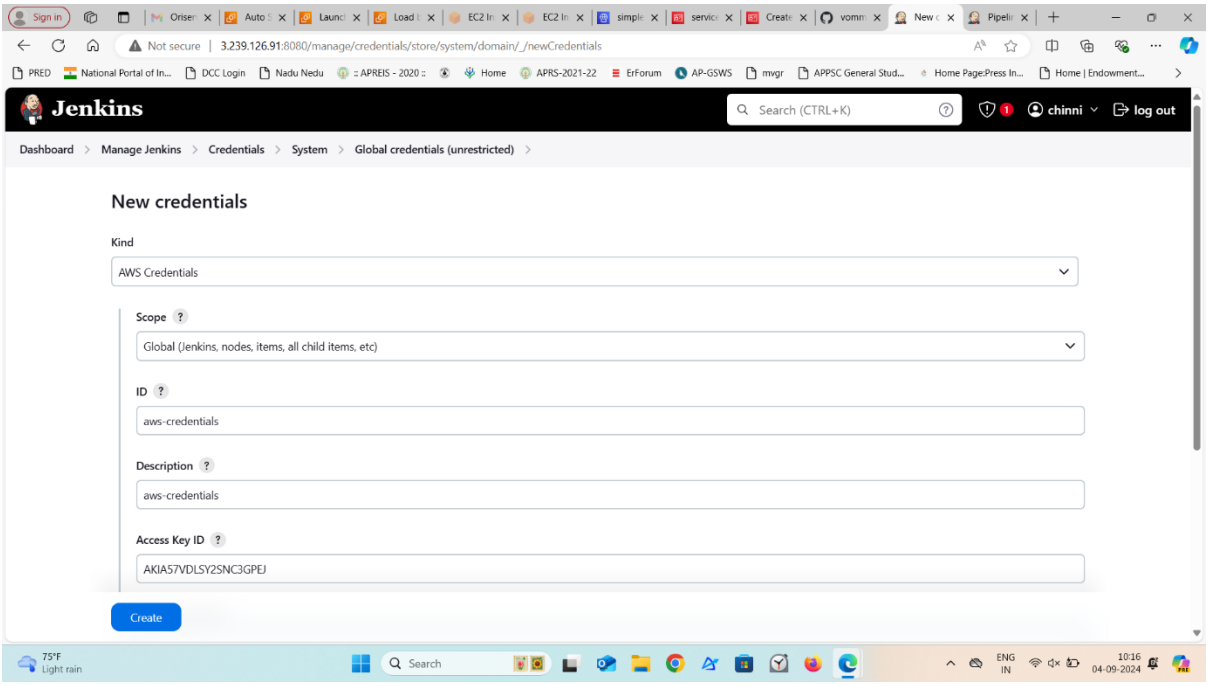


Step 11: Add AWS Credentials in Jenkins



After the AWS Credentials plugin is installed:

1. **Add Credentials**:
   - Go back to **Manage Jenkins → Manage Credentials**.
   - Click on the relevant domain (e.g., **Global**).
   - Click **Add Credentials**.

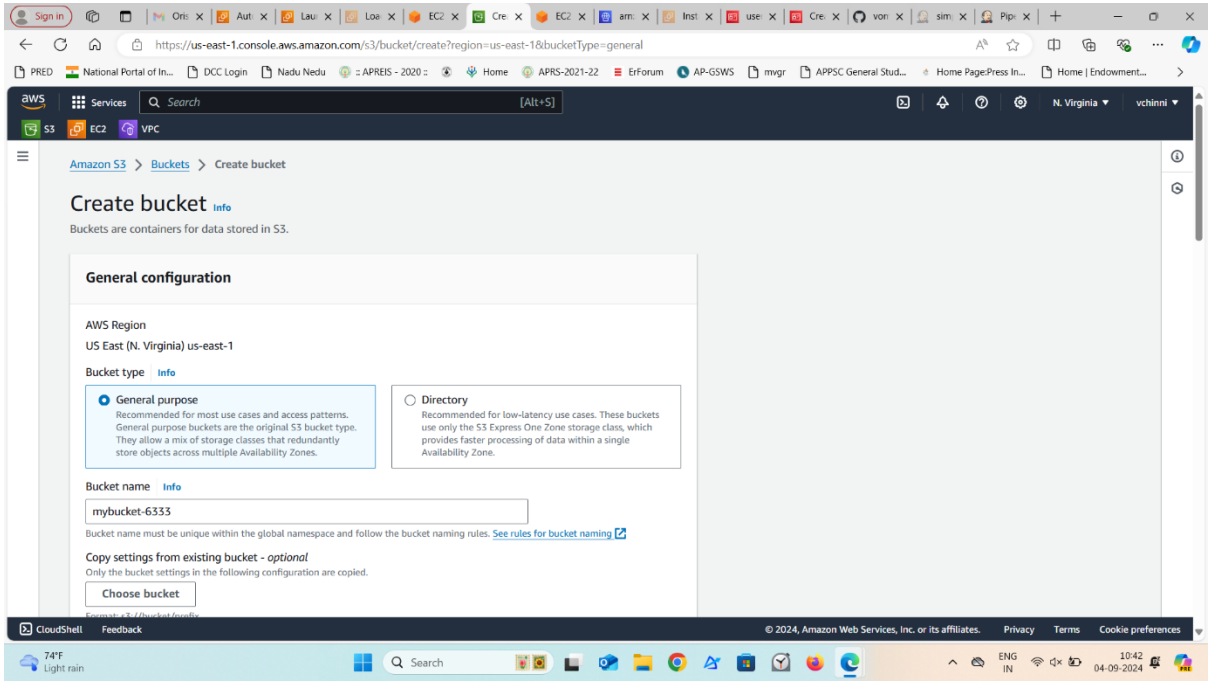


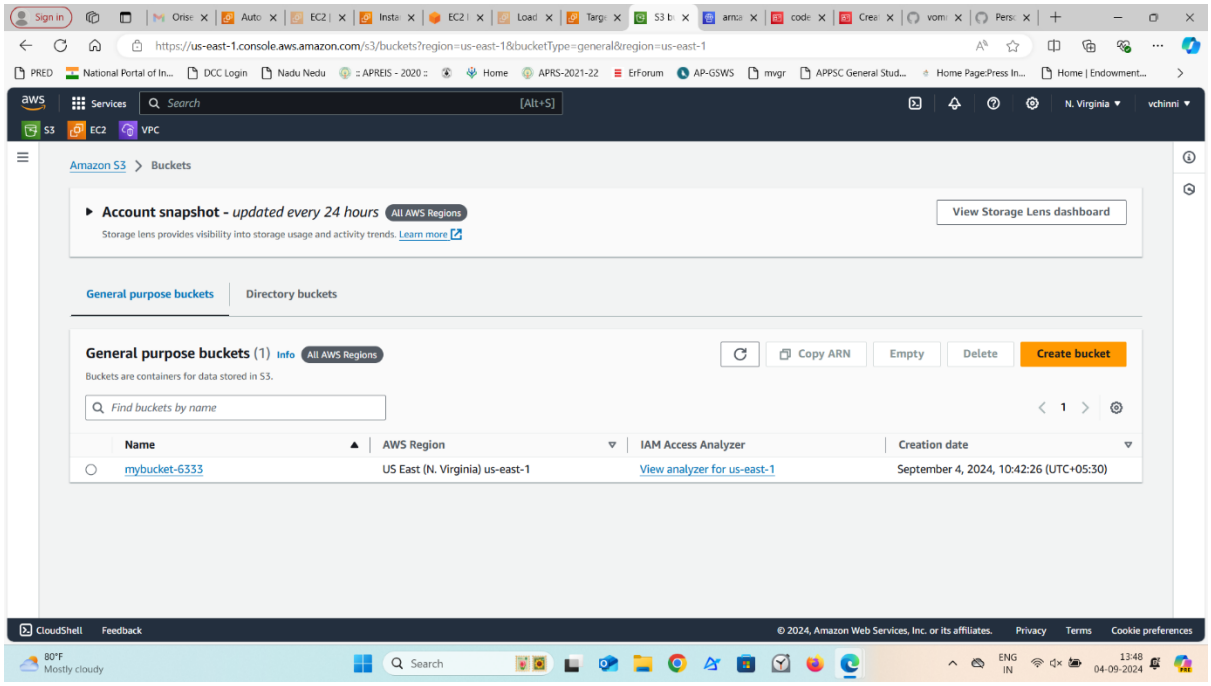
2. **Select the Correct Credential Type**:
   - In the **Kind** dropdown, you should now see **"AWS Credentials"**.

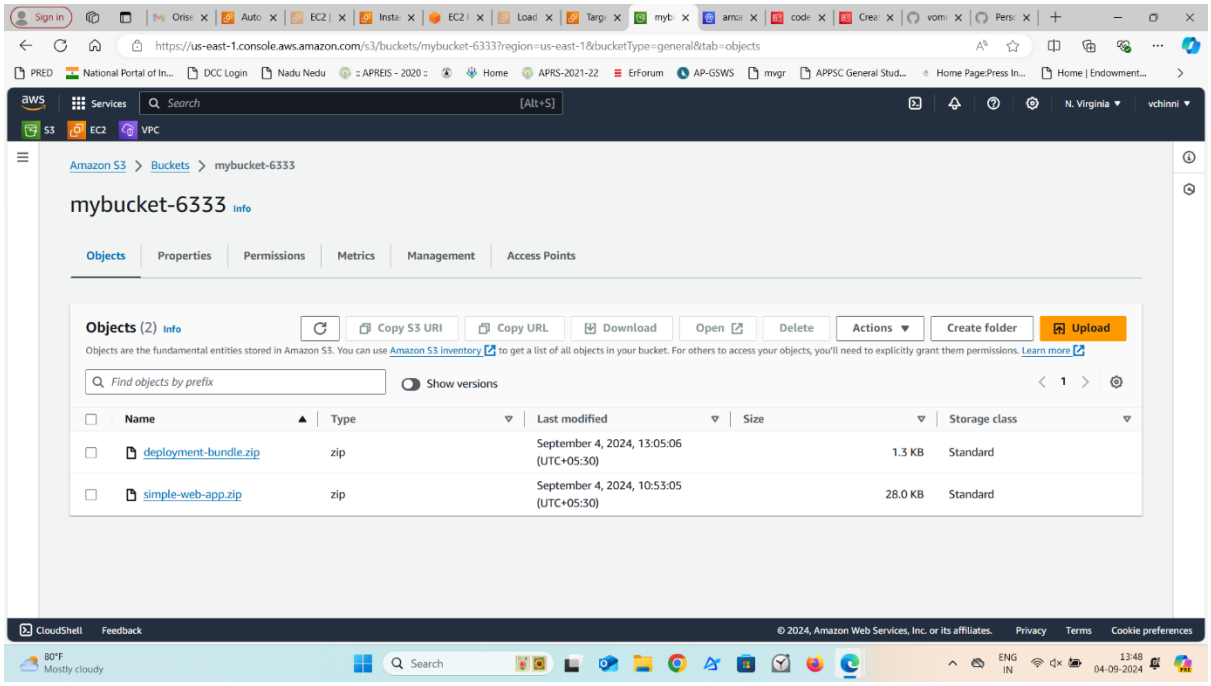
- Enter your **AWS Access Key ID** and **Secret Access Key**.
- Provide an **ID** and a **description** to identify these credentials.

Create an S3 Bucket

1. **Open the S3 Console:**
   - Go to the Amazon S3 console.
2. **Create a New Bucket:**
   - Click on “Create bucket.”
   - Enter a unique name for your bucket.
   - Choose the region where you want the bucket to be created.
   - Configure options as needed and click “Create bucket.”
3. **Upload Deployment Bundle:**
   - Go to the newly created bucket.
   - Click “Upload” and add your deployment bundle (simple-web-app.zip).
   - Click “Upload” to complete the process.
4. aws s3 cp ~/Desktop/simple-web-app.zip s3://your-bucket-name/



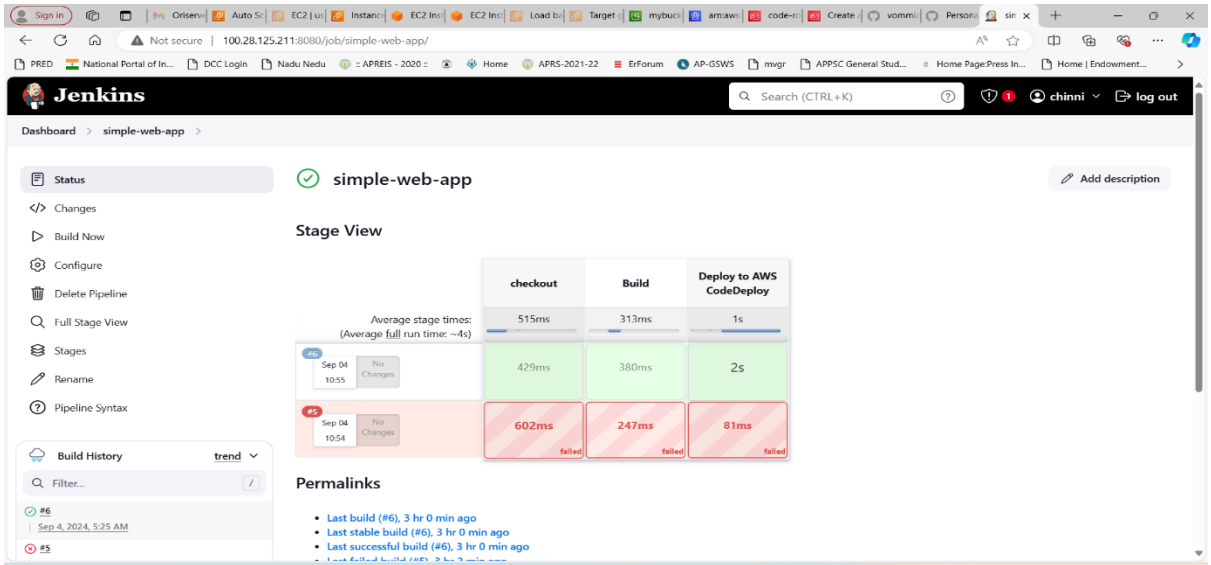


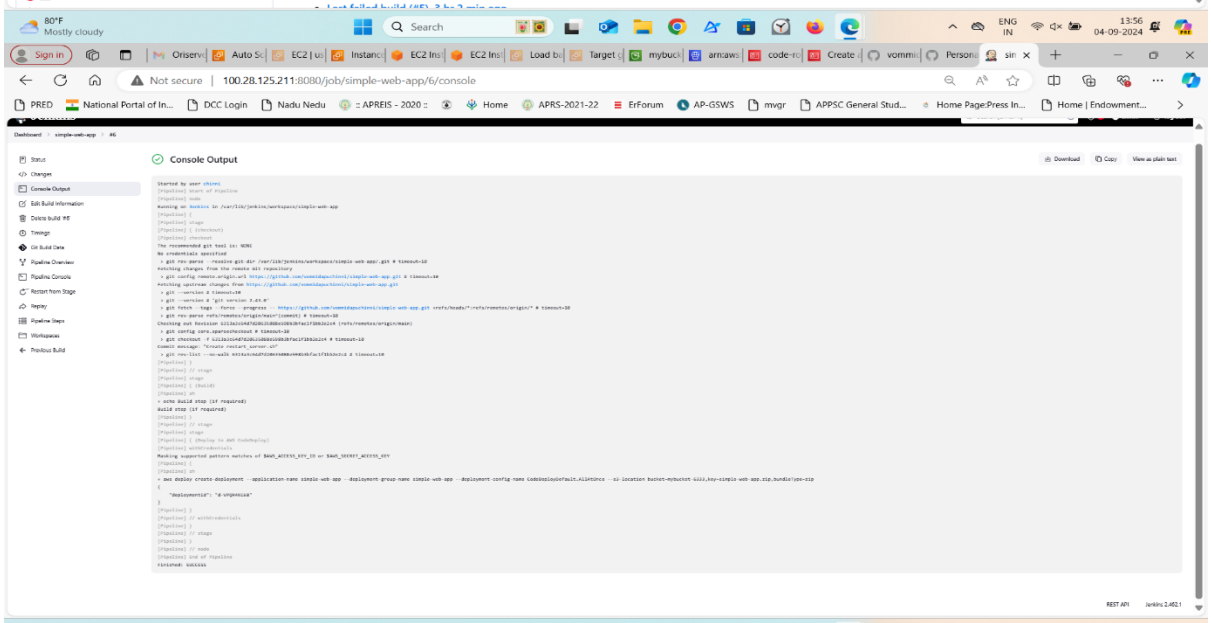


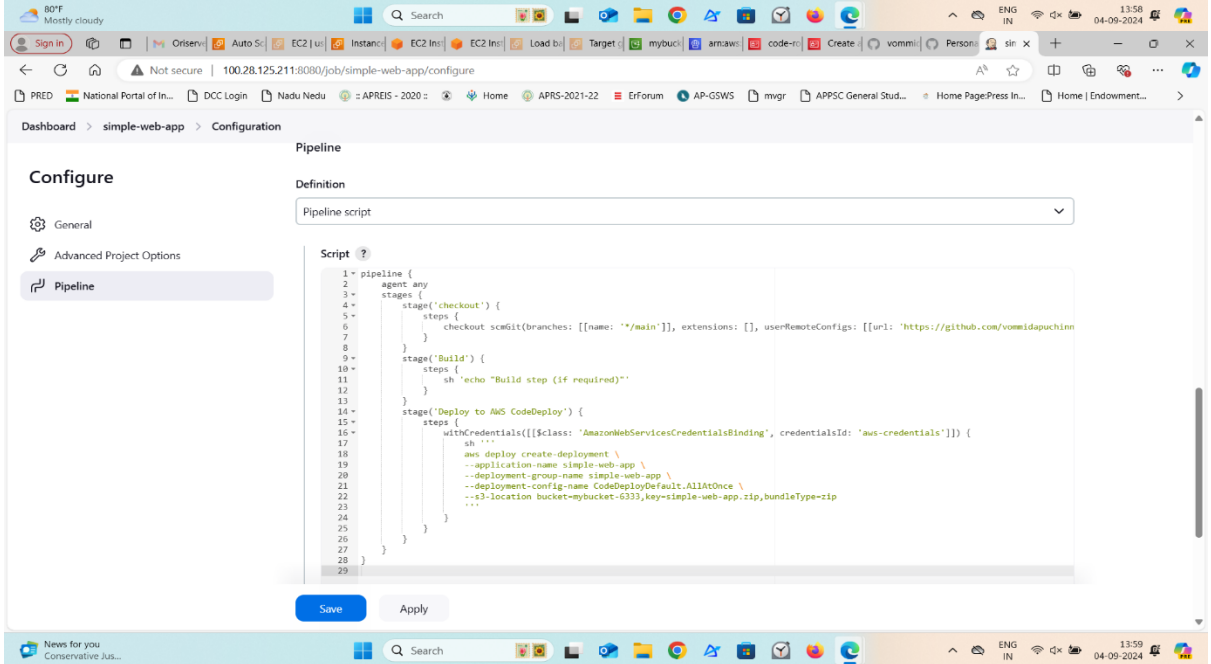
Update the Jenkinsfile to Use the Credentials:

```
pipeline {
    agent any
    stages {
        stage('Checkout') {
            steps {
```

```
            // Checkout code from GitHub repository
            checkout scmGit(branches: [[name: '*/main']],
                        extensions: [],
                        userRemoteConfigs: [[url:
'https://github.com/vommidapuchinni/simple-web-app.git']])
            }
        }
        stage('Build') {
            steps {
                // Placeholder for build steps
                sh 'echo "Build step (if required)"'
            }
        }
        stage('Deploy to AWS CodeDeploy') {
            steps {
                // Use AWS credentials from Jenkins
                withCredentials([aws(credentialsId: 'aws-credentials')]) {
                    sh '''
                    aws deploy create-deployment \
                    --application-name simple-web-app \
                    --deployment-group-name simple-web-app-deployment-group \
                    --deployment-config-name CodeDeployDefault.AllAtOnce \
                    --region us-east-1 \
                    --s3-location bucket=mybucket,key=simple-web-
app.zip,bundleType=zip
                    '''
                }
            }
        }
    }
}
```


Jenkins
Search (CTRL+K)
chinni
log out
Dashboard > simple-web-app >
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Stages
Rename
Pipeline Syntax
Build History
trend
Filter...
#6
Sep 4, 2024, 5:25 AM
#5
simple-web-app
Add description
Stage View
checkout
Build
Deploy to AWS CodeDeploy
Average stage times:
(Average full run time: ~4s)
515ms
313ms
1s
Sep 04
10:55
No Changes
429ms
380ms
2s
Sep 04
10:54
No Changes
602ms
failed
247ms
failed
81ms
failed
Permalinks
Last build (#6), 3 hr 0 min ago
Last stable build (#6), 3 hr 0 min ago
Last successful build (#6), 3 hr 0 min ago


Not secure | 100.28.125.211:8080/job/simple-web-app/6/console
Console Output


Not secure | 100.28.125.211:8080/job/simple-web-app/configure
Dashboard > simple-web-app > Configuration
Configure
General
Advanced Project Options
Pipeline
Pipeline
Definition
Pipeline script
Script
pipeline {
agent any
stages {
stage('checkout') {
steps {
checkout scmGit(branches: [[name: '*/main']], extensions: [], userRemoteConfigs: [[url: 'https://github.com/vommidapuchinn
}
}
stage('Build') {
steps {
sh 'echo "Build step (if required)"'
}
}
stage('Deploy to AWS CodeDeploy') {
steps {
withCredentials([[$class: 'AmazonWebServicesCredentialsBinding', credentialsId: 'aws-credentials']]) {
sh '''
aws deploy create-deployment \
--application-name simple-web-app \
--deployment-group-name simple-web-app \
--deployment-config-name CodeDeployDefault.AllAtOnce \
--s3-location bucket=mybucket-6333,key=simple-web-app.zip,bundleType=zip
'''
}
}
}
}
}
Save
Apply

Verify CodeDeploy Agent Installation

- Once the installation is complete, you can verify that the CodeDeploy agent is running on your instances:
    - **SSH into your EC2 instance** and run: sudo service codedeploy-agent status

Install awscli on Jenkins server

chmod +x restart_server.sh
sudo ./restart_server.sh

edit trust relationship in iam role for code-role

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "ec2.amazonaws.com",
          "codedeploy.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Step 12:

Check Console Navigation

1. **Go to CodeDeploy Console:**
    - Navigate to the AWS CodeDeploy Console.
2. **Select Application:**
    - Click on your application name (e.g., simple-web-app).
3. **Find Deployment Options:**
    - Look for **"Deployments"** in the left-hand menu or under the application details.
4. **Locate Create Deployment:**

- Inside the **"Deployments"** section or tab, you should see options like **"Create deployment"** or **"Create new deployment"**.

2. Check Permissions

Ensure you have the necessary IAM permissions to create deployments:

1. **IAM Role/Policy:**
   - Make sure the IAM user or role you're using has the codedeploy:CreateDeployment permission.

Application and Deployment Group

- **Application:** Select simple-web-app.
- **Deployment Group:** Select simple-web-app.

2. Compute Platform

- **Compute platform:** Ensure EC2/On-premises is selected.

3. Deployment Type

- **Deployment type:** Choose In-place if you want to update the existing instances with the latest application revision.

4. Revision Type

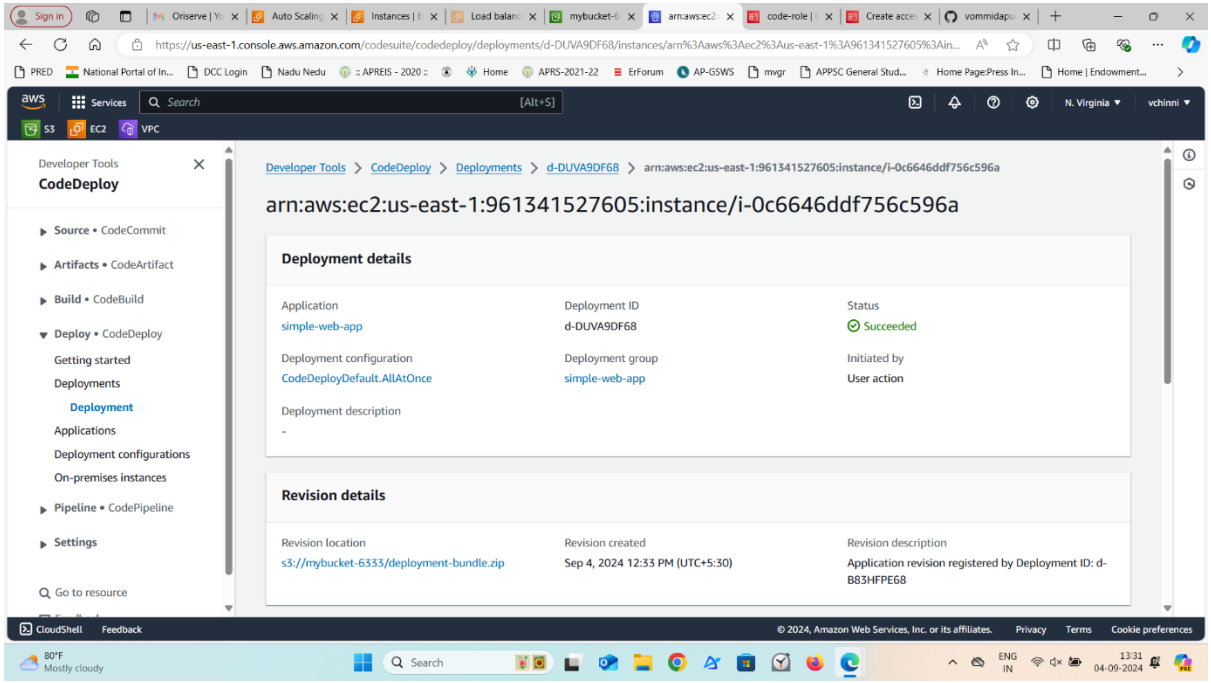
- **Revision type:** Choose My application is stored in Amazon S3.

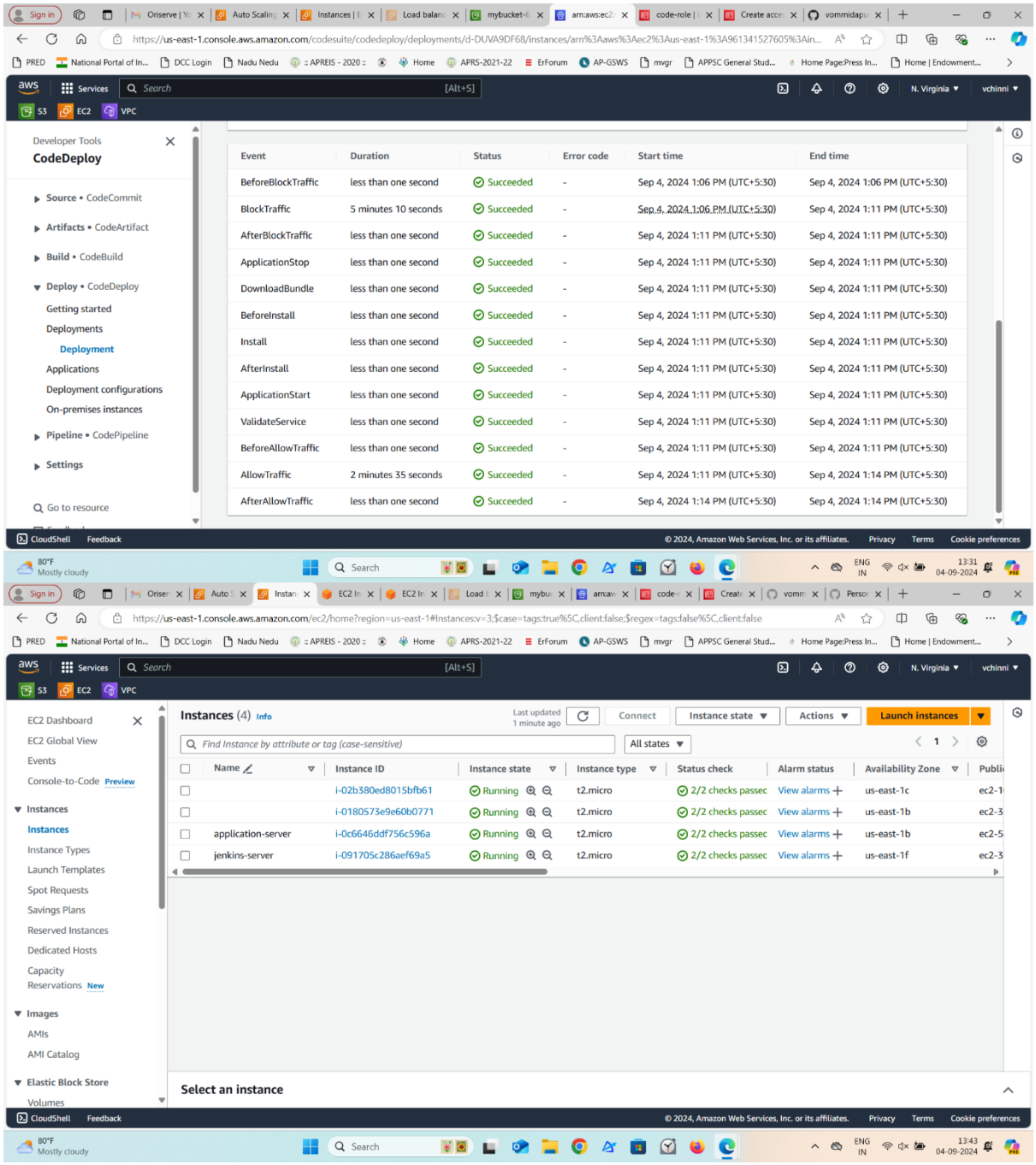
5. Revision Location

- **Revision location:** Enter the S3 bucket URL where your deployment bundle is stored. Example: s3://mybucket-6333/deployment-bundle.zip.

6. Revision File Type

- **Revision file type:** Confirm or select .zip if your deployment bundle is a zip file.

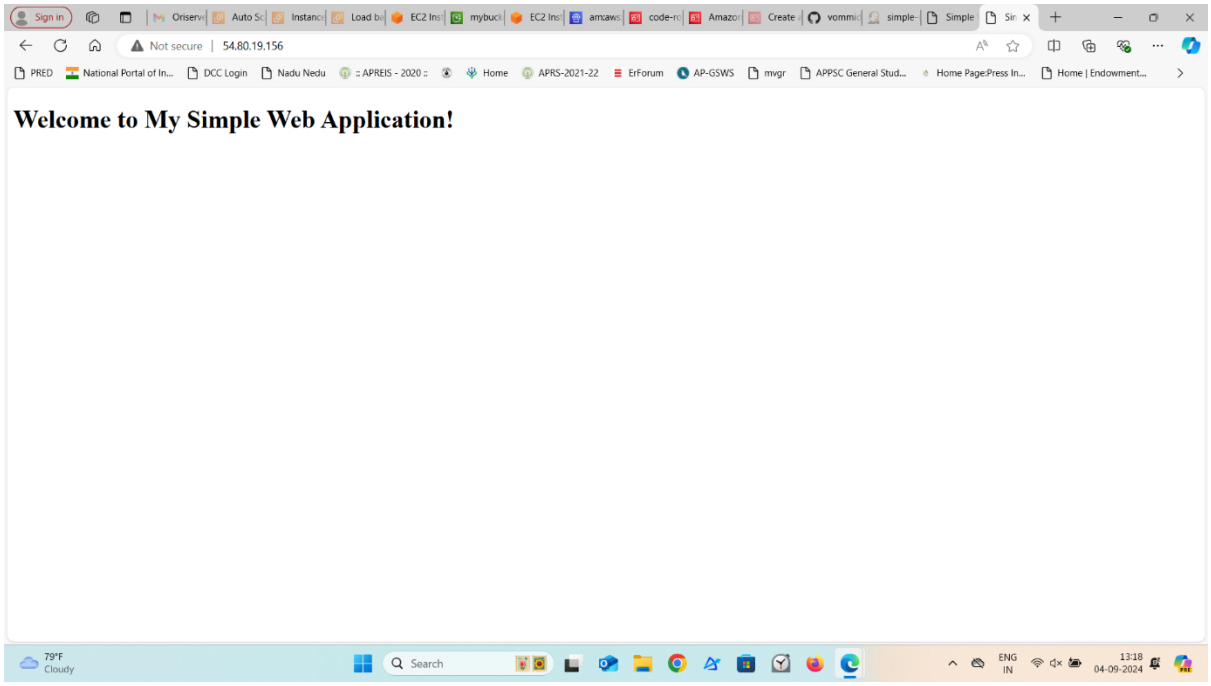




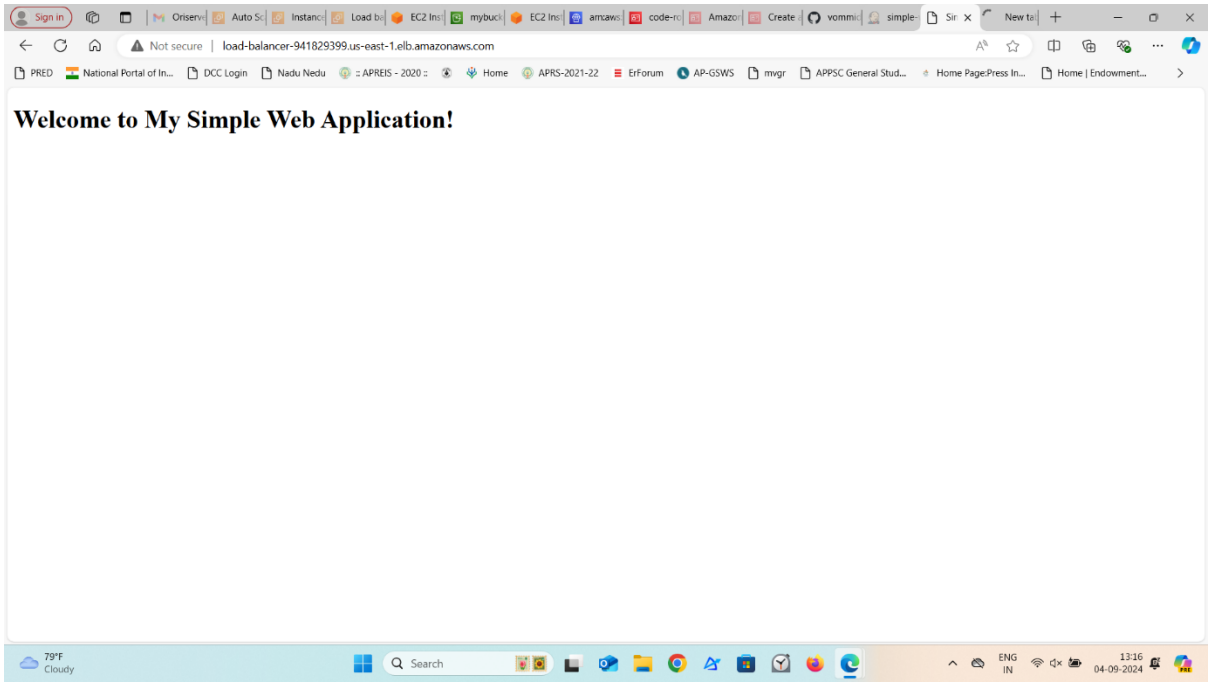
Access via Public IP or Domain Name:

For EC2 Instances:

- **Public IP Address:**
  - If your EC2 instance has a public IP address, you can access your application by entering this IP address in your web browser.
  - Example: http://<public-ip-address>

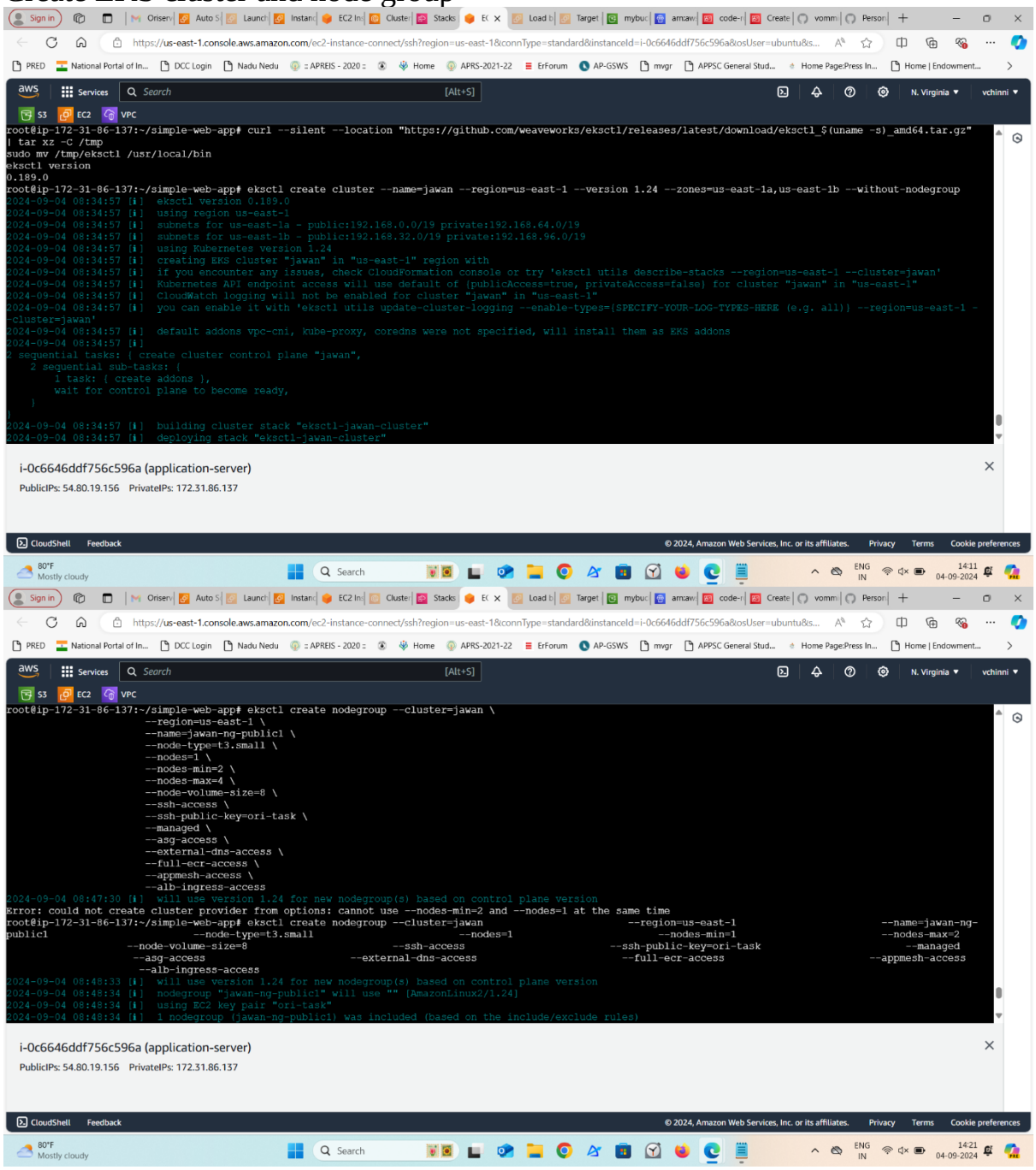


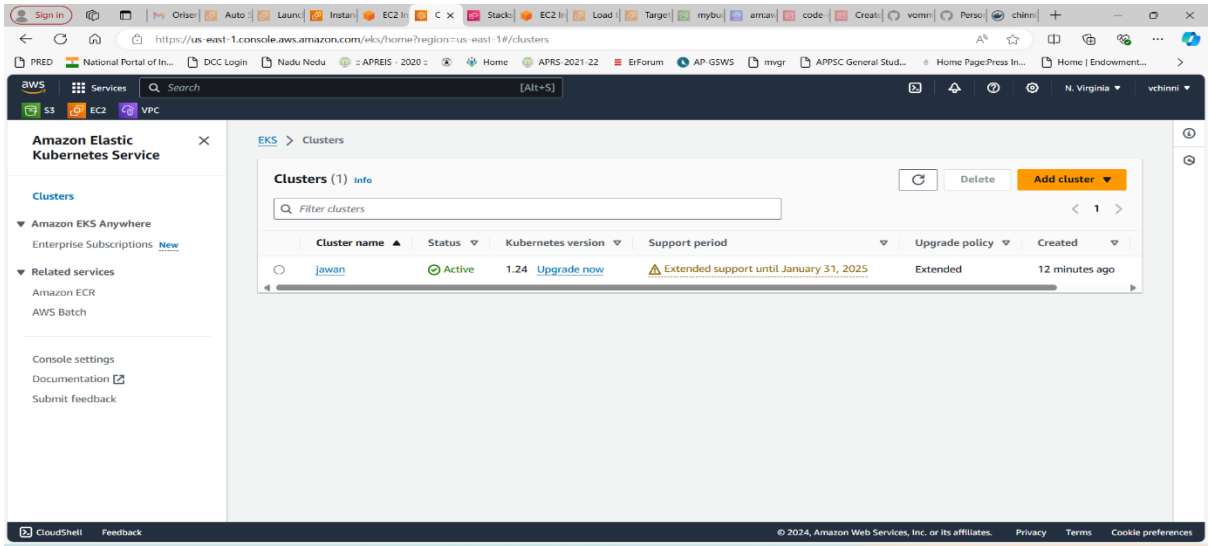
-
- **Elastic Load Balancer (ELB):**
  - If you are using an Elastic Load Balancer (ELB), you can access your application using the ELB DNS name.
  - You can find this DNS name in the EC2 Management Console under "Load Balancers."

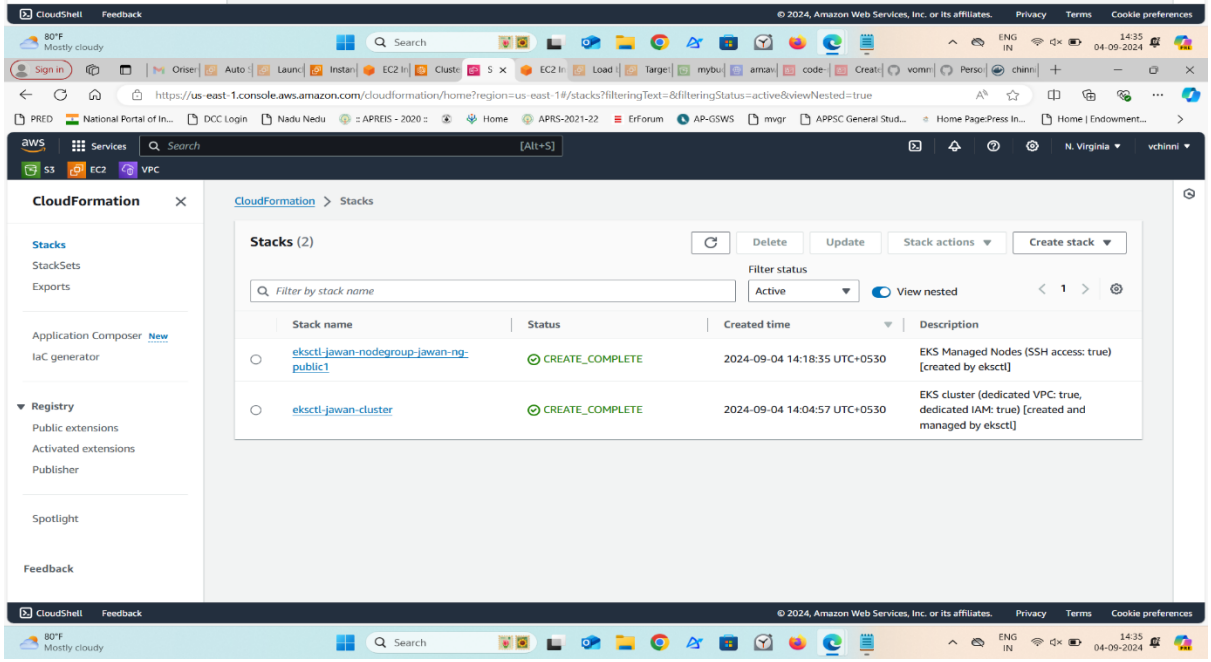


Deploying and accessing with Kubernetes:

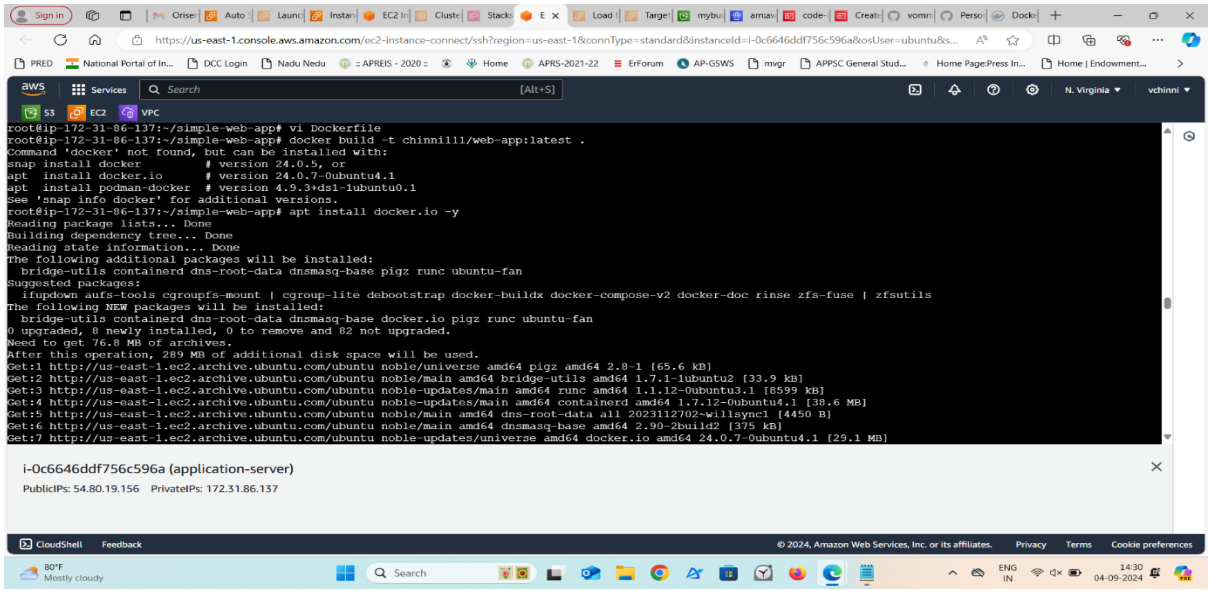
Create EKS cluster and node group



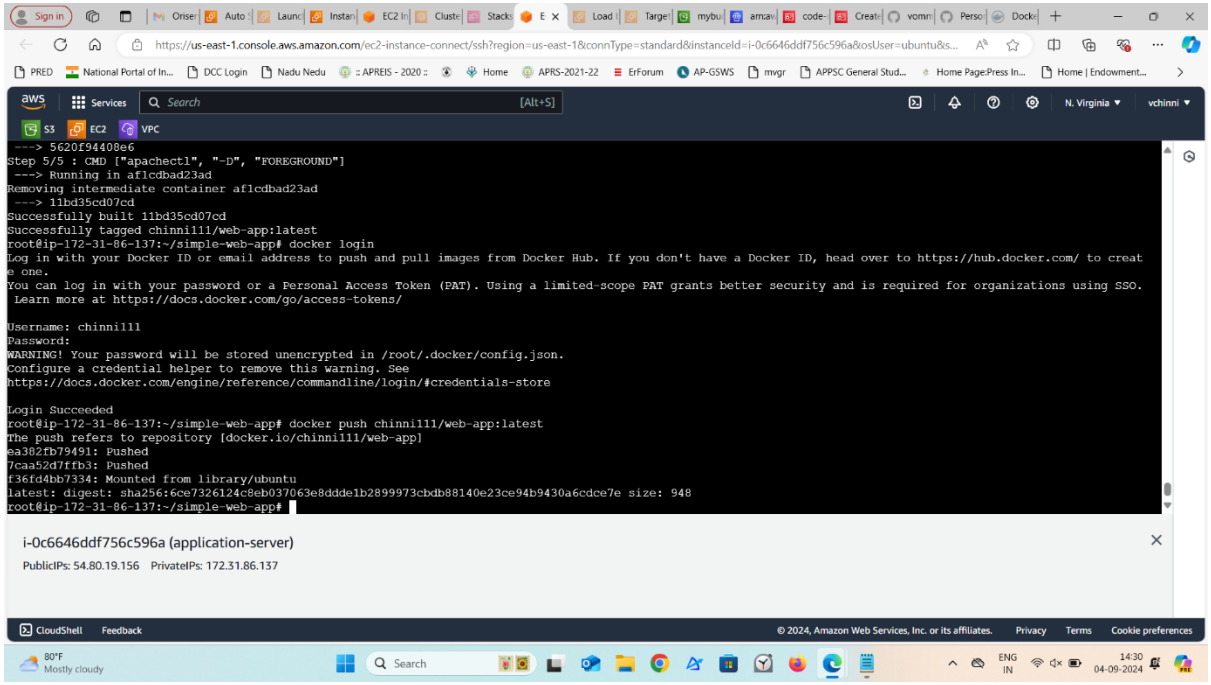




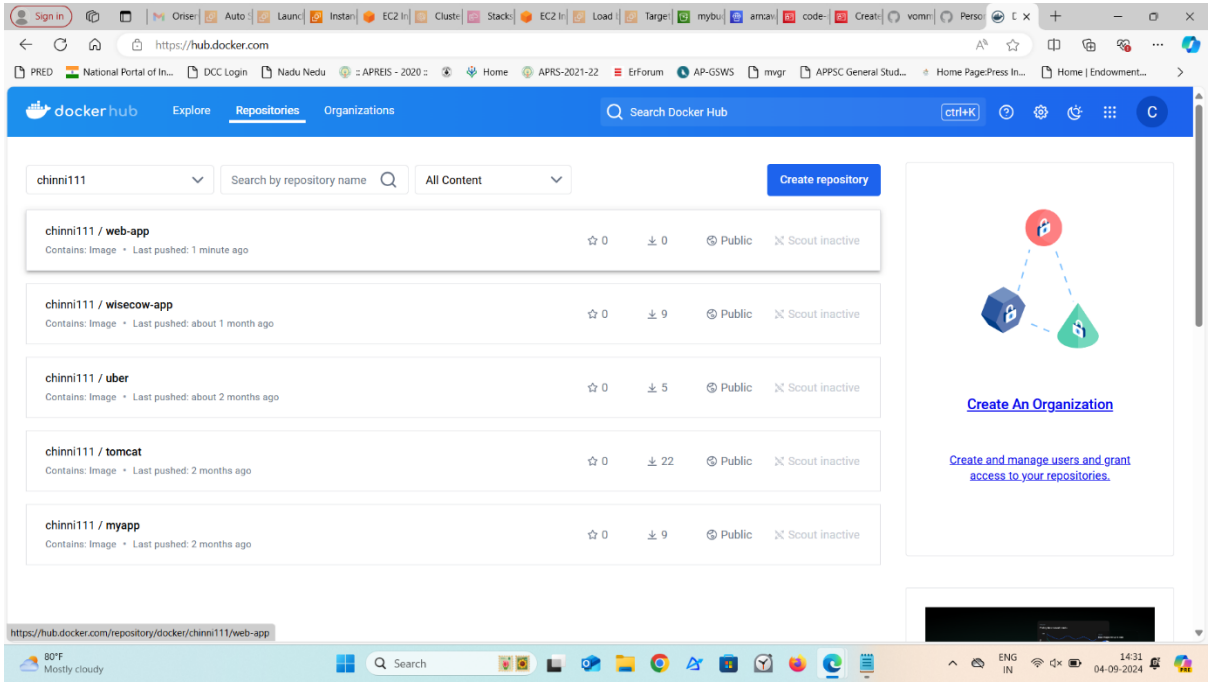
Created a docker file



Build the docker file



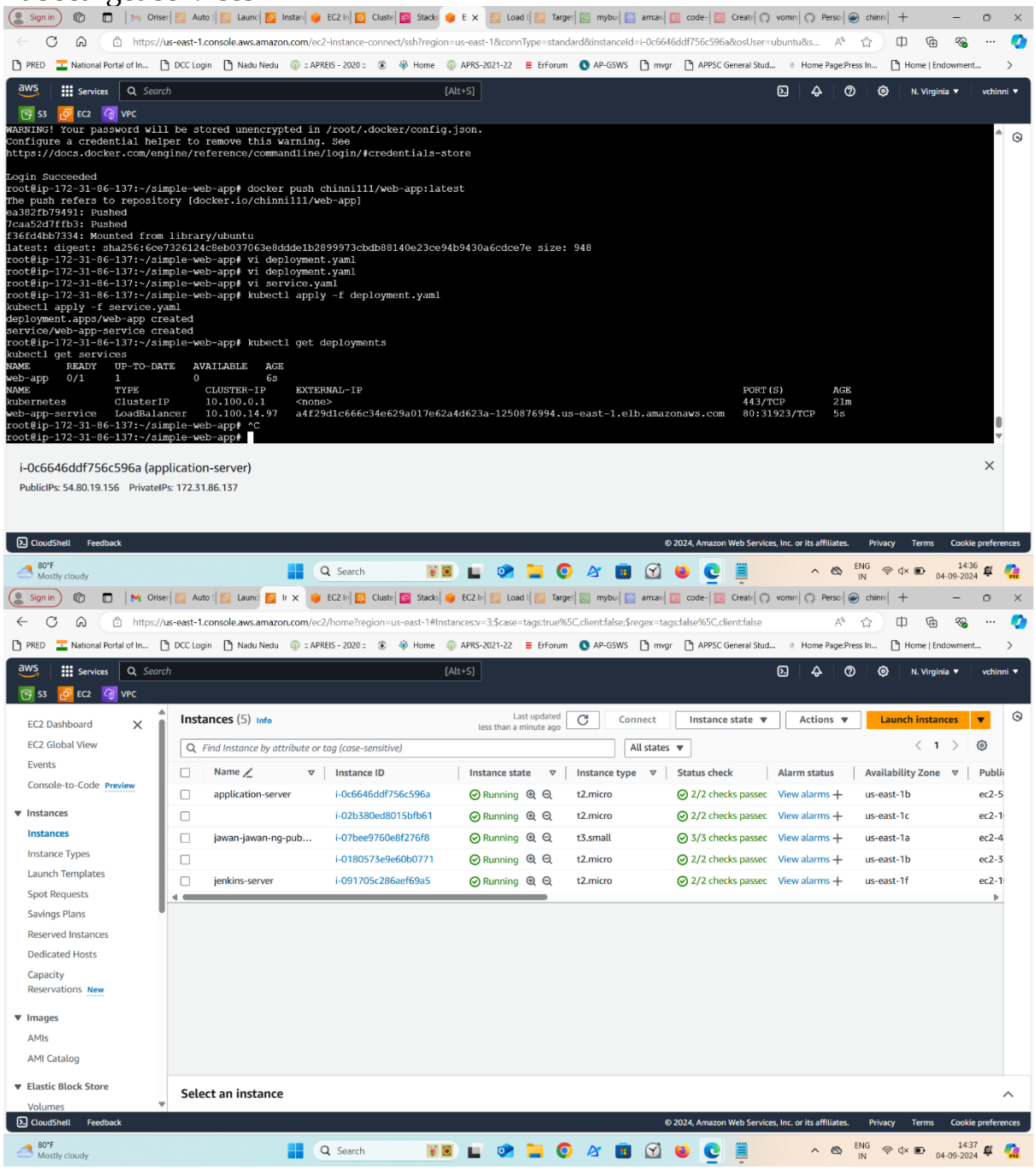
Push to docker hub



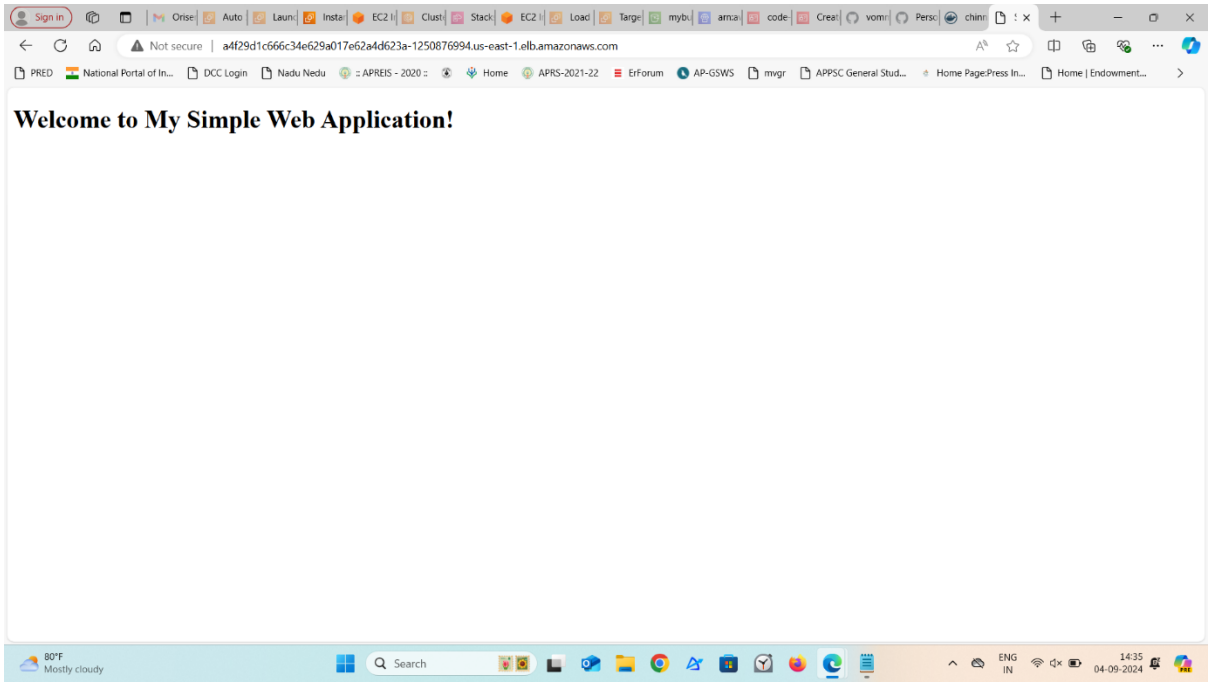
Created deployment and service yaml files
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml

check the status of your deployment:

kubectl get deployments
kubectl get services



The LoadBalancer service will provide an external IP address to access your application.



Problems Encountered and Solutions

a. AWS CodeDeploy Issues

- Issue: File already exists on the deployment target.
- Solution: Manually removed the existing file or adjusted deployment settings to overwrite or retain existing files.

b. EKS Cluster Deletion

- Issue: EKS cluster was deleted, but resources remained in CloudFormation.
- Solution: Deleted remaining CloudFormation stacks and associated IAM roles.

c. Docker Image Issues

- Issue: Docker image not available in Docker Hub.
- Solution: Built and pushed the Docker image to Docker Hub.

d. Kubernetes Cluster Issues

- Issue: Problems accessing or managing the EKS cluster.
- Solution: Verified AWS CLI and kubectl configurations.

Conclusion: This project demonstrates a comprehensive approach to deploying a simple web application using AWS services, Docker, and Kubernetes. Starting with setting up EC2 instances and utilizing AWS CodeDeploy for automated deployments, the project extends to containerizing the application with Docker and managing it with Kubernetes. This setup enhances the application's scalability, maintainability, and reliability. Through this journey, we addressed deployment challenges, effectively managed resources, and ensured smooth application delivery, providing a robust framework for deploying web applications in a cloud-native environment.