**RUN A CICD PIPELINE FOR TERRAFORM TO INIT,FORMAT,VALIDATE,PLAN AND APPLY.**

**1.JENKINS INSTALLATION**

First install Jenkins in a server

Create a ec2 instance and connect to that instance then switch to root user by using

(sudo su –) command.

install Jenkins using this commands

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

https://pkg.jenkins.io/debian/jenkins.io-2023.key

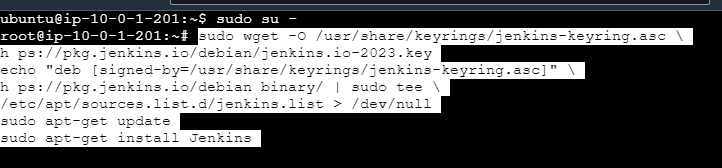
echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \

https://pkg.jenkins.io/debian binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update

sudo apt-get install Jenkins

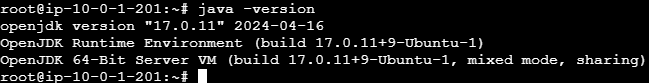


Jenkins requires java to run

**2.INSTALL JAVA**

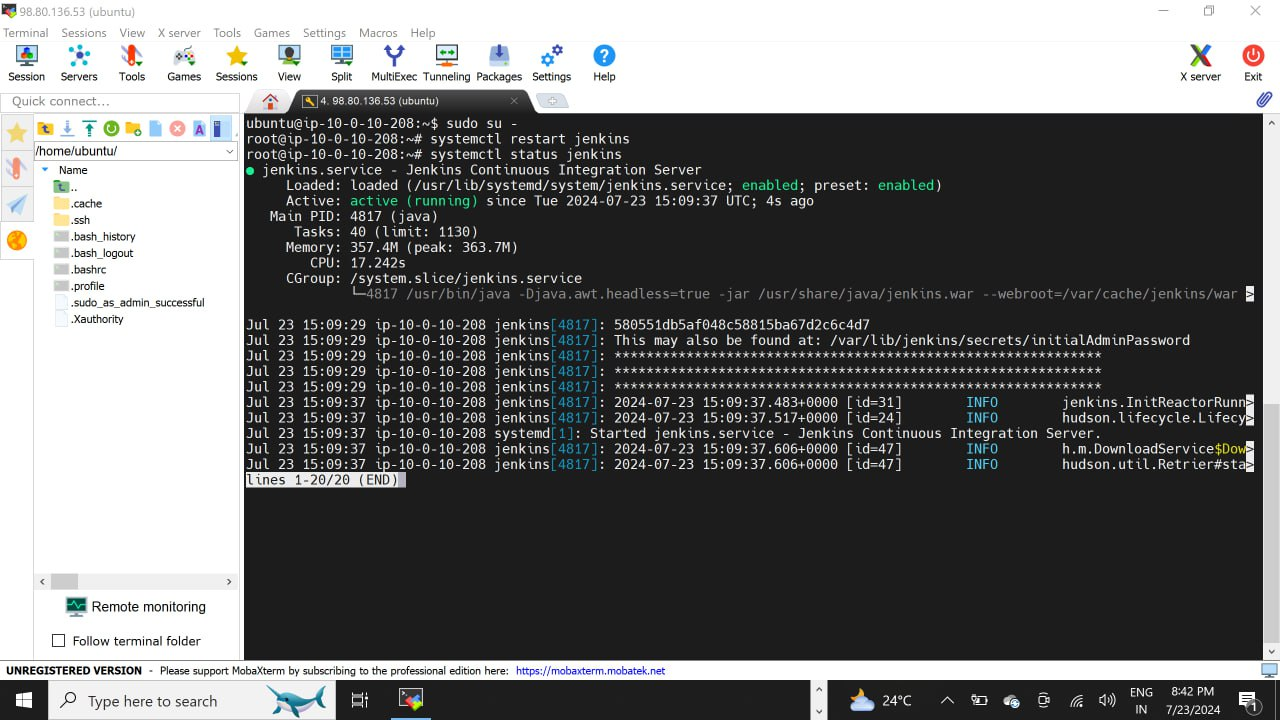
apt install openjdk-17-jdk -y : This command used to install java

java -version : To check the java version



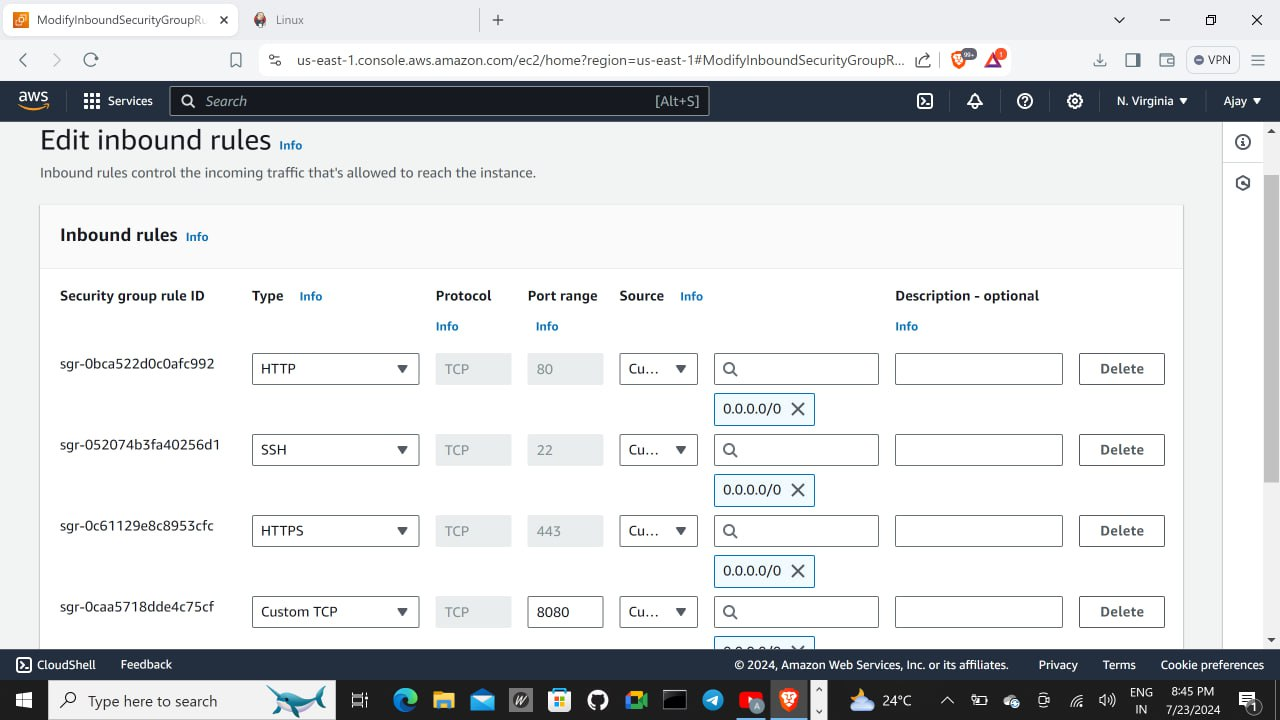
after installing Jenkins and java in the server we need to restart the Jenkins server

systemctl restart Jenkins : to restart the Jenkins

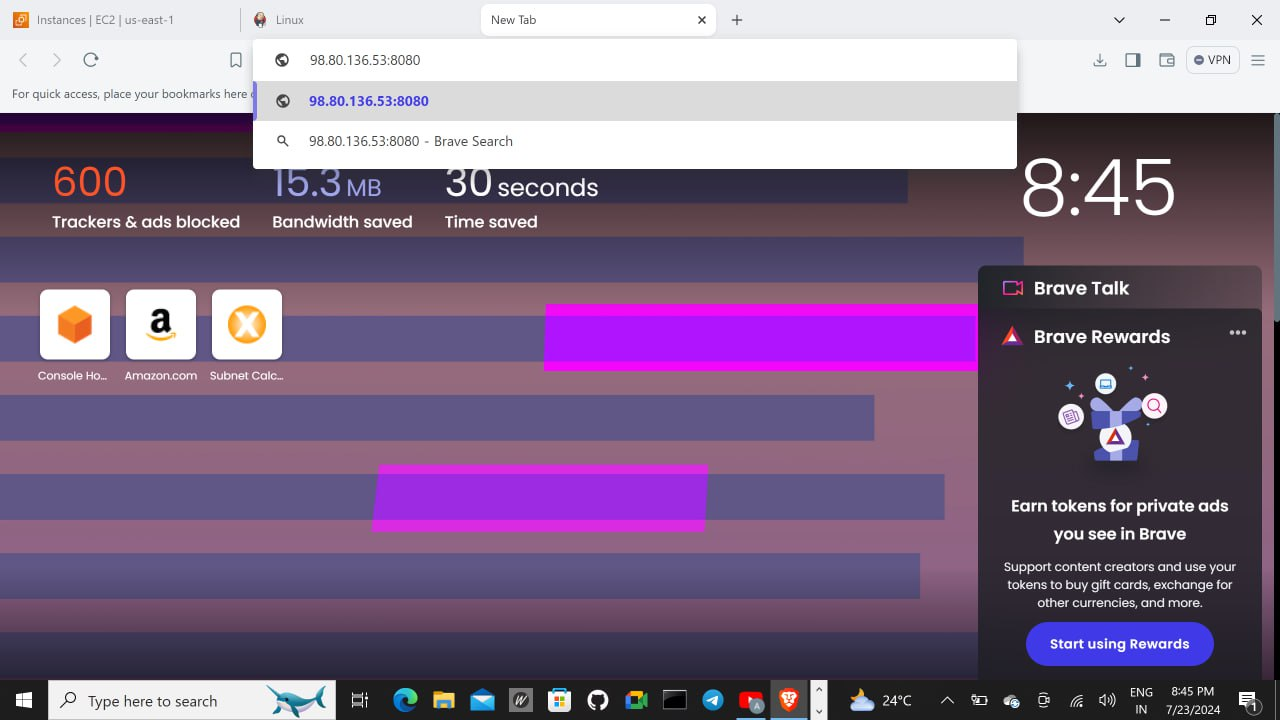
systemctl status Jenkins : to check the Jenkins status , whether its running or not 

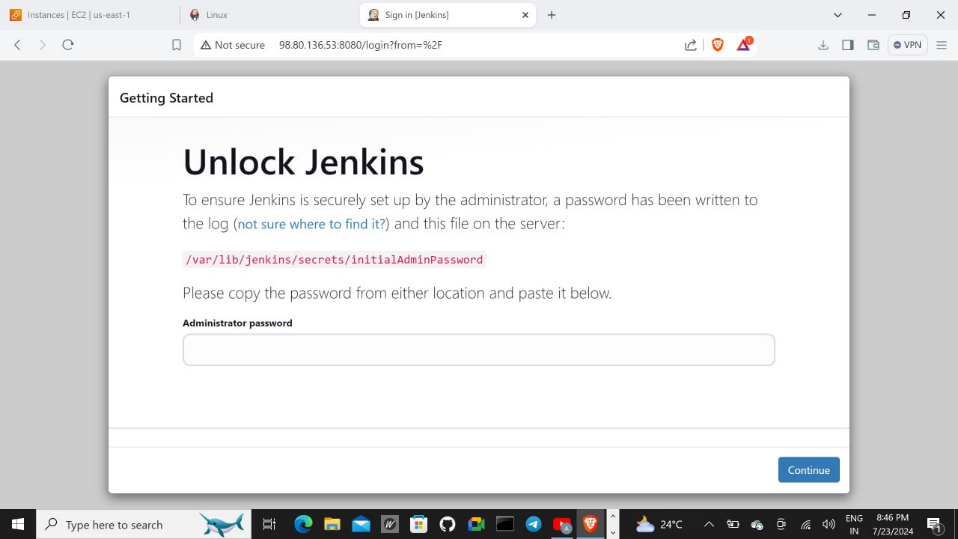
**3. LOGIN INTO JENKINS DASHBOARD**

To access Jenkins. We need to allow inbound traffic on PORT 8080 in the AWS security group.



Copy the public ip of the server and paste in the browser with port number 8080{Example : 54.89.243.242:8080}

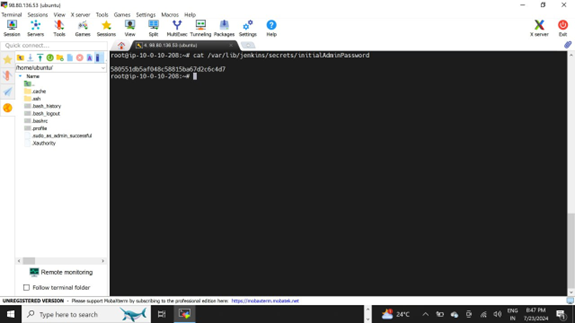


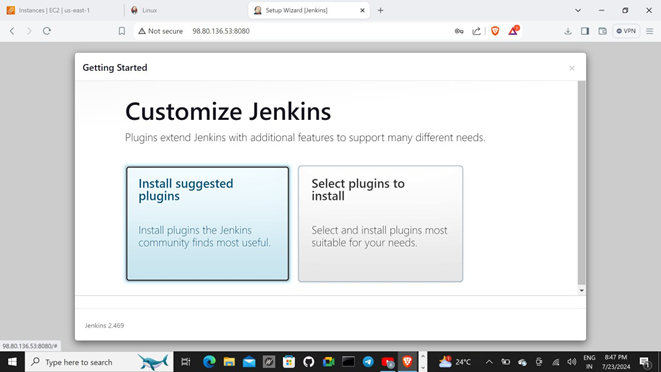


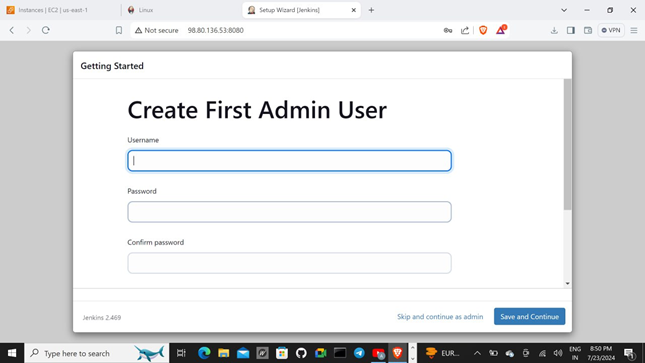
To unlock Jenkins we need administrator password

Go to your terminal and paste the command below to get the password to unlock Jenkins

Cat /var/lib/jenkins/secrets/initialAdminPassword

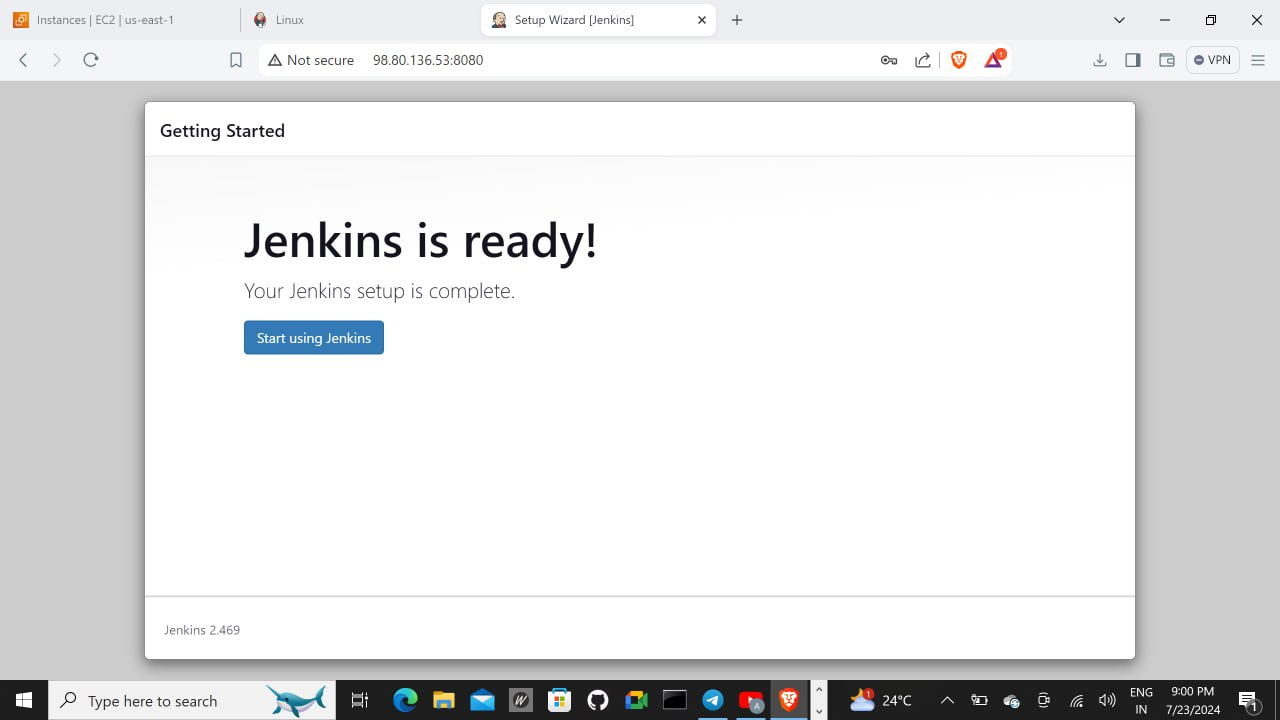


Select install suggested plugins

Create an admin user by filling in the required information

Click save and finish.

**Now Jenkins setup has been completed.**



in Jenkins dashboard select manage Jenkins and click on plugins and install terraform plugins.

Now, again select manage Jenkins and click on tools and install terraform and save.

Select credentials and add aws credentials and git credentials.

**Create a vpc ,public and private subnet , internet gateway ,rout table and create a ec2 instance in public subnet by using terraform.**

* **In main.tf**

  provider "aws" {

    region = "us-east-1"

  }

  //create a vpc

resource "aws\_vpc" "myvpc"{

    cidr\_block = "10.0.0.0/16"

    tags = {

        Name = "MyVPC"

    }

}

  //Create a public subnet

resource "aws\_subnet" "PublicSubnet"{

    vpc\_id = aws\_vpc.myvpc.id

    availability\_zone = "us-east-1a"

    cidr\_block = "10.0.1.0/24"

    tags = {

        Name = "terra\_sub-pub"

    }

}

  //create a private subnet

resource "aws\_subnet" "PrivSubnet"{

    vpc\_id = aws\_vpc.myvpc.id

    cidr\_block = "10.0.2.0/24"

    map\_public\_ip\_on\_launch = true

    tags = {

        Name = "terra\_sub-pvt"

    }

}

  //create IGW

resource "aws\_internet\_gateway" "myIgw"{

    vpc\_id = aws\_vpc.myvpc.id

    tags = {

        Name = "terra\_igw"

    }

}

  //route Tables for public subnet

resource "aws\_route\_table" "PublicRT"{

    vpc\_id = aws\_vpc.myvpc.id

    tags = {

        Name = "terra\_RT"

    }

    route {

        cidr\_block = "0.0.0.0/0"

        gateway\_id = aws\_internet\_gateway.myIgw.id

    }

}

  //route table association public subnet

resource "aws\_route\_table\_association" "PublicRTAssociation"{

    subnet\_id = aws\_subnet.PublicSubnet.id

    route\_table\_id = aws\_route\_table.PublicRT.id

}

//ec2 instance

resource "aws\_instance" "ec2" {

  ami           = "ami-04a81a99f5ec58529"

  instance\_type = "t2.micro"

  subnet\_id =  aws\_subnet.PublicSubnet.id

  associate\_public\_ip\_address = true

  tags = {

    Name = "terra\_ec2"

  }

}

* **In variable.tf**

variable "ami\_id" {

  description = "The AMI ID for the EC2 instance"

  type        = string

  default = "ami-04a81a99f5ec58529"

}

variable "instance\_type" {

  description = "The instance type"

  type        = string

  default = "t2.micro"

}

variable "public\_subnet\_id" {

  default = aws\_subnet.PublicSubnet.id

}

* **In output.tf**

output "vpc\_id" {

    value = aws\_vpc.myvpc.id

}

output "public\_subnet\_id" {

  value = aws\_subnet.PublicSubnet.id

}

output "private\_subnet\_id" {

  value = aws\_subnet.PrivSubnet.id

}

output "route\_table\_id" {

    value = aws\_route\_table.PublicRT.id

}

Create a backend.tf file to store state file in s3 bucket

terraform {

  backend "s3" {

    bucket         = "terraform-bucket12"

    key            = "state"

    region         = "us-east-1"

  }

}

Create a Jenkins script for terraform to run a pipeline

pipeline{

    agent any

    stages{

        stage('Git checkout'){

            steps{

                "your's git repo URL"

            }

        }

        stage('Terraform init'){

            steps{

                sh 'terraform init'

            }

        }

        stage('Terraform Format') {

            steps {

                    sh 'terraform fmt -check'

                }

            }

        stage('Terraform Validate') {

            steps {

                    sh 'terraform validate'

                   }

            }

        stage('Terraform plan'){

            steps{

                sh 'terraform plan'

            }

        }

        stage('Terraform apply'){

            steps{

                sh 'terraform apply --auto-approve'

            }

        }

    }

}

**Explanation**

**pipeline:** Defines a Jenkins pipeline.

**agent any:** Runs the pipeline on any available agent.

**stage('Checkout'):** Checks out the Terraform code from the specified Git repository.

**stage('Terraform Init'):** Initializes the Terraform configuration.

**stage('Terraform Format'):** Checks if the Terraform files are formatted correctly using terraform fmt -check.

**stage('Terraform Validate'):** Validates the Terraform configuration using terraform validate.

**stage('Terraform Plan'):** Creates an execution plan using terraform plan and saves it to a file named tfplan.

**stage('Terraform Apply'):** Applies the changes required to reach the desired state of the configuration using the previously created plan (tfplan).

After creating all files push them into git repository.

git init

git remote add origin <https://github.com/username/repository.git>

git add . # Add all fils

git commit -m "Commit message"

git push -u origin master

use this commands to add your local files into git repository

open the Jenkins dashboard and select the new item.

Enter an item name: terraform\_cicd

Select an item type: pipeline

After selecting pipline click on ok .It opens Configuration in that select pipeline

Select pipeline script from SCM in the definition

SCM: Git

Repository URL: <https://github.com/ajayguvva/terraform_cicd.git>

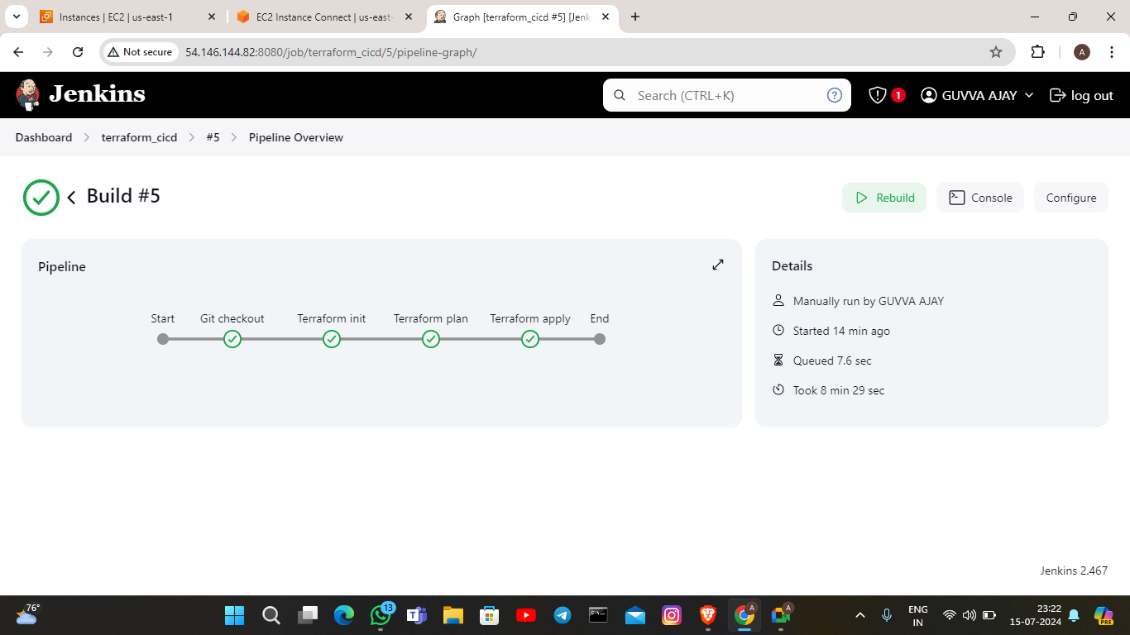
Credentials : give your Credential

Branch Specifier: main ( branch name where the files are stored )

Script Path: jenkinsfile (path name where the script is written)

After filling all this then click on apply and save it.

Now select build now and it will run cicd pipeline.



This setup will provision a VPC with public and private subnets, an internet gateway, route tables, and an EC2 instance in the public subnet using Terraform. The Jenkins pipeline will automate the Terraform workflow from initialization to applying changes.