**Create Build and Deploy Pipeline for K8s Application Deployment**

**TASK:**

Create Build and Deploy Pipeline for K8s application deployment.

Create Job DSL code to create build and deploy pipeline.

Create Free Style Job to Run Job DSL code to create above pipeline jobs.

Build pipeline is image pipeline to build docker file and push image to AWS elastic container registry (ECR).

Deploy pipeline is used to deploy the application using kubectl into k8s cluster.

Outcome is application deployed and run in k8s cluster. Able to access nginx website with url <http://<ip_address>:<port>.>

This document provides a step-by-step guide to create a build and deploy pipeline for deploying a Kubernetes (K8s) application. The pipeline includes building a Docker image and pushing it to AWS Elastic Container Registry (ECR), as well as deploying the application using kubectl into a K8s cluster. The desired outcome is to have the application deployed and running in the K8s cluster, accessible through the URL http://<ip\_address>:<port>.

**Solution:**

**Step 1: AWS Login**

* Creating the one new instance
* Installation Jenkins
* Installation of minikube
* Installation of Docker
* Installation AWS Cli

**Step 2: Create Jenkins Job**

* Install plugins Docker ,kubernetes, Job DSL
* Create free style Project
* Click on the DSL script
* write DSL script pipeline on build and deploy
* Build the sample code using git repository

**Step 3: Docker images creation**

* Inside Sample project creating Docker file
* Create ECR repository.
* Build the Docker images.
* Write Code on ECR Credential in pipeline pushing the image on ECR repository **.**
* After pushing image Deploying the application on Kubernetes.

**Step 4: Deploy the Kubernetes Application**

* Write Yaml file on deployment and service
* The minikube application Started deploy the deployment.yaml file .
* Checking Ec2 terminal
* Go to root user sudo -i
* Inside root user Jenkins working
* Checking kubectl get pods
* Checking the SVC

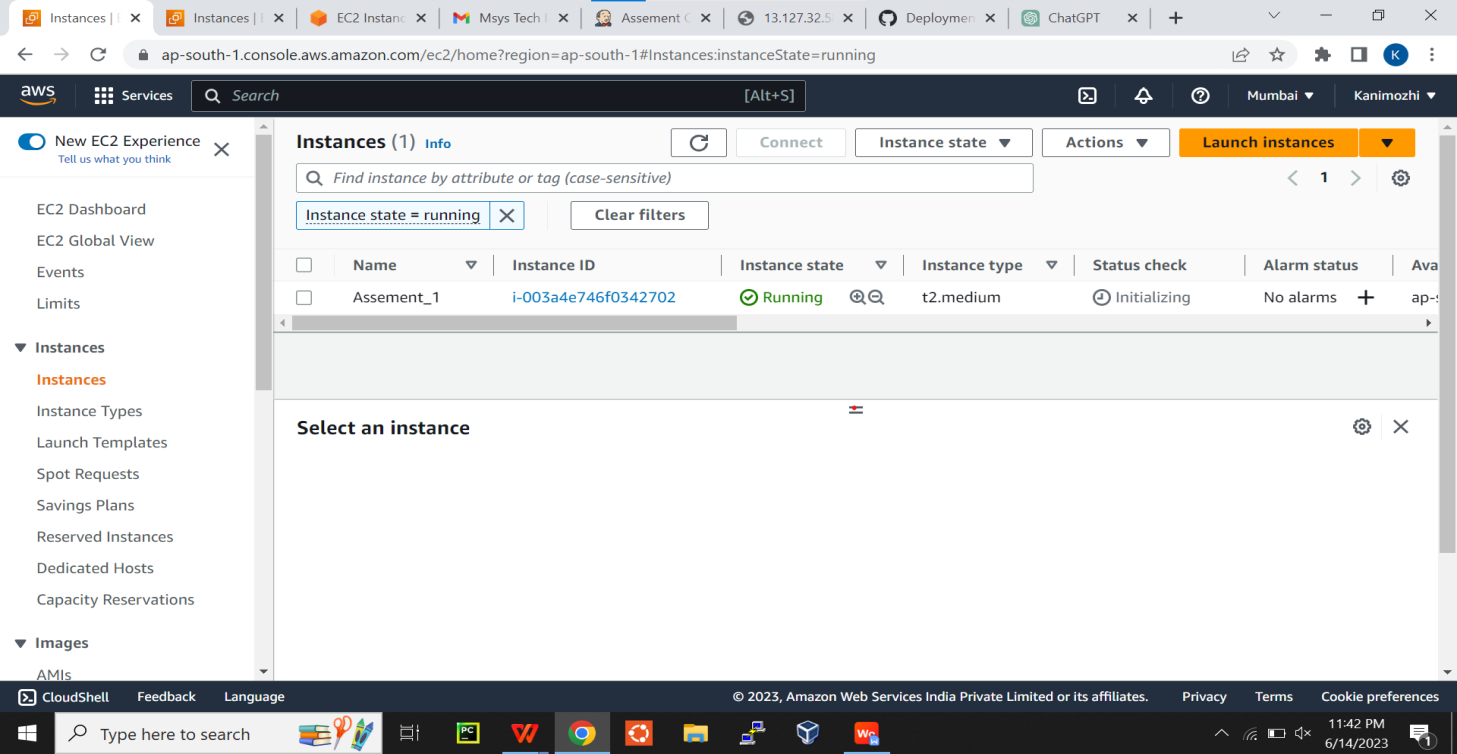
**Step 5: Validate Deployment**

After the "Deploy Pipeline" job completes successfully, verify that the application is deployed and running in the K8s cluster.

* Obtain the IP address and port of th**e.**

**Screen Shots:**

1. **Lanuch Instance**



1. **Connect to Ec2 Instance.**

* Install jenkins
* Sudo apt-get update
* sudo apt install openjdk-11-jre
* Check java --version
* Jenkins install following steps

curl -fsSL <https://pkg.jenkins.io/debian-stable/jenkins.io-> 2023.key | sudo tee \

/usr/share/keyrings/jenkins-keyring.asc > /dev/null

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

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

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

* sudo apt-get update
* sudo apt-get install jenkins

1. **Install On Minikube**

* sudo apt-get update && sudo apt-get install -y curl wget virtualbox
* sudo apt-get update && sudo apt-get install -y apt-transport-https
* curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
* echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
* sudo apt-get update
* sudo apt-get install -y kubectl
* curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
* sudo install minikube-linux-amd64 /usr/local/bin/minikube
* Check minikube version
* Minikube start

1. **Install On AWS CLI**

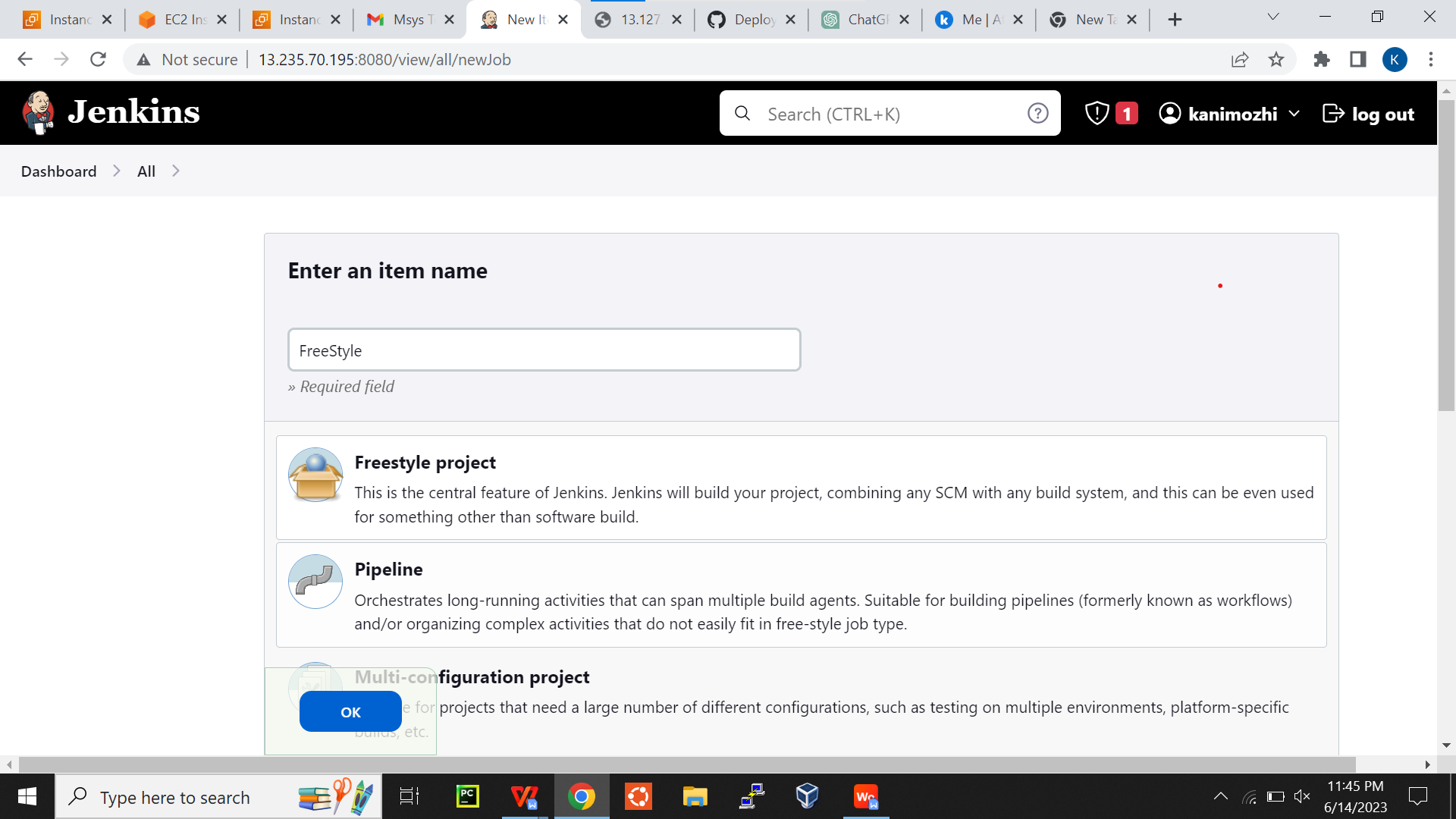
* sudo apt install aws-cli
* Sudo apt-get update

**5.Prerequisites**

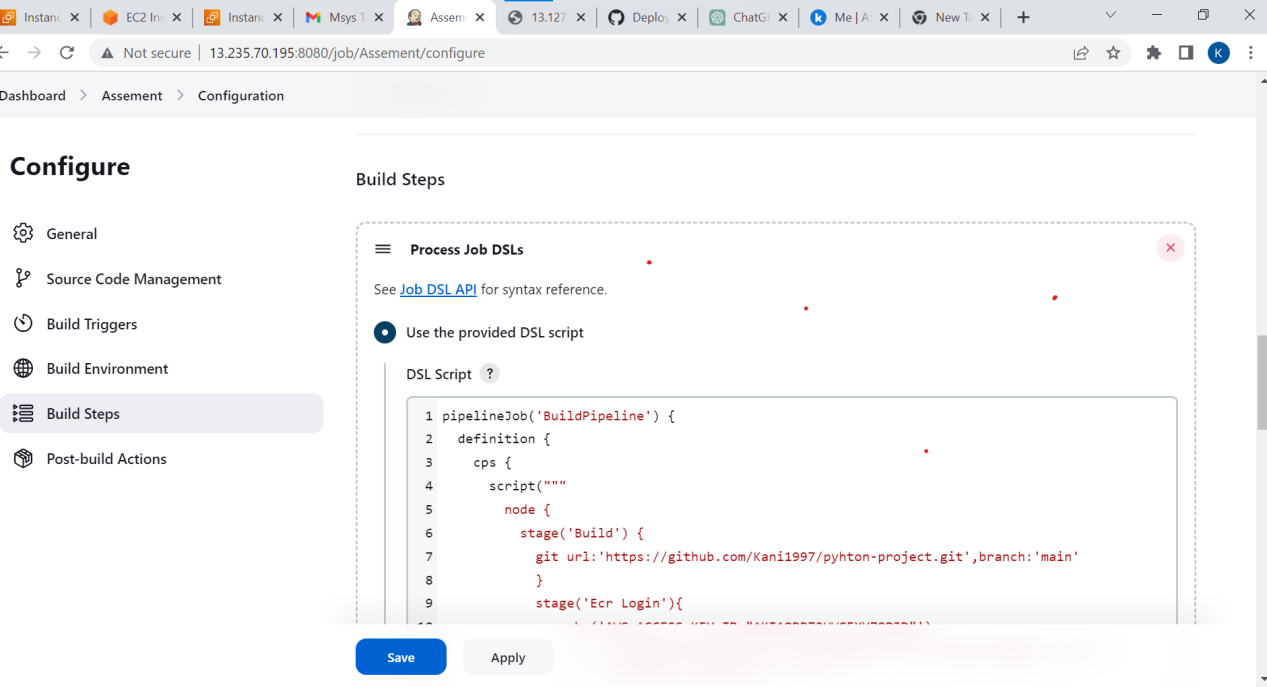
Before starting the pipeline setup, ensure that the following prerequisites are met:

* A K8s cluster is set up and accessible.
* AWS credentials are available for pushing the Docker image to ECR.
* Jenkins is installed and configured with necessary plugins (Job DSL and AWS credentials plugin).

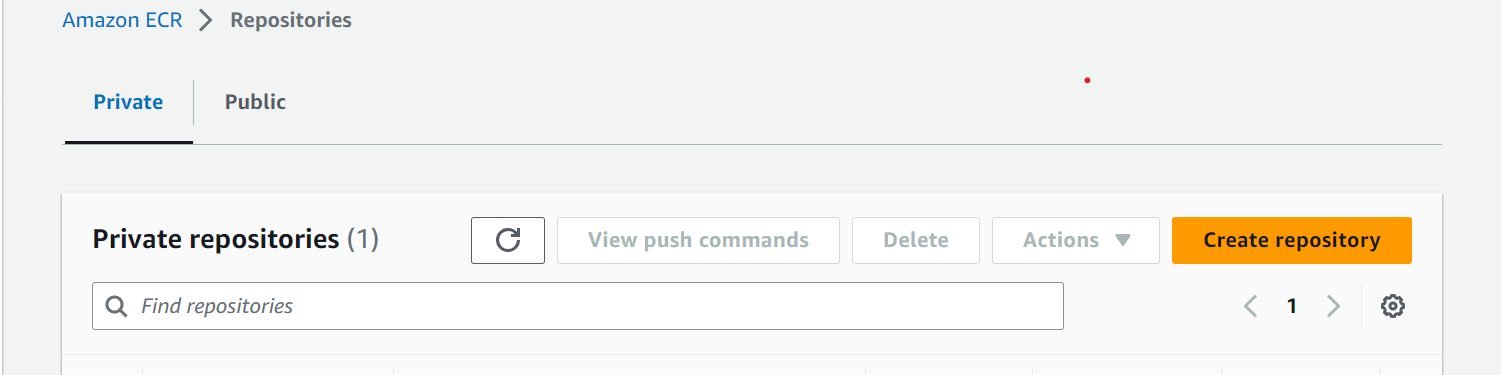
**6.Create Freestyle Project**



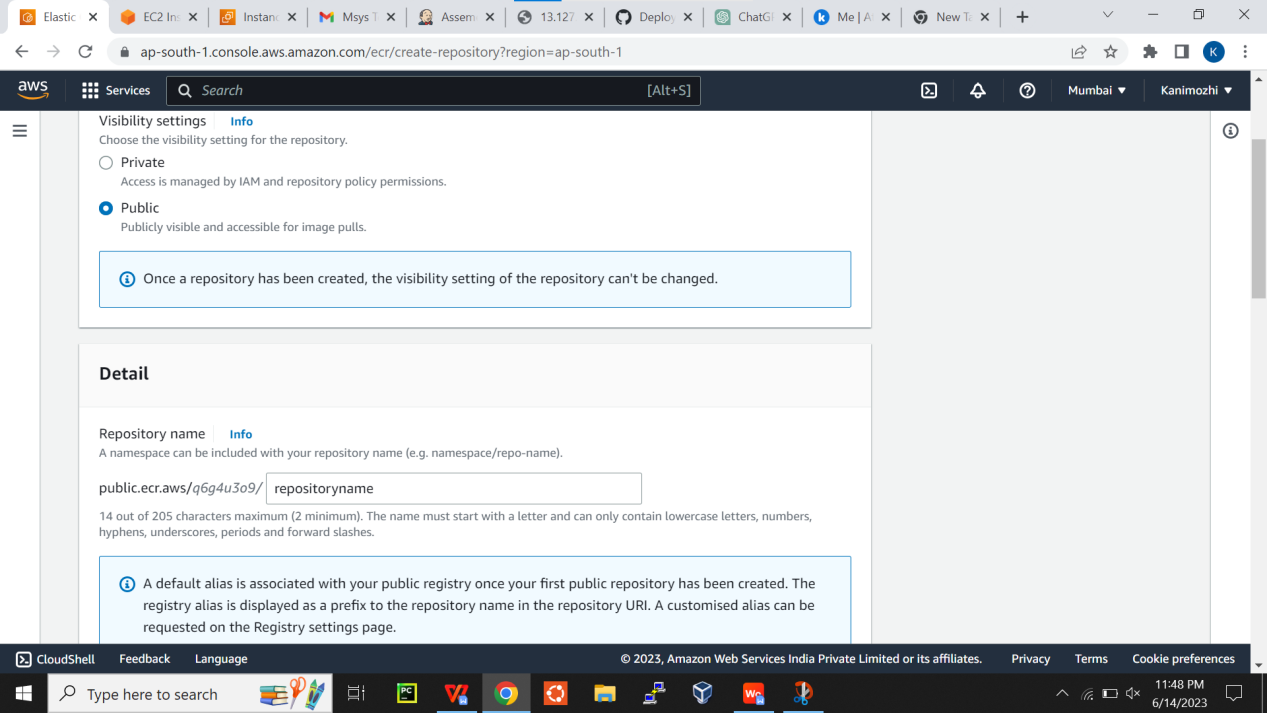
**7.Create DSL script**



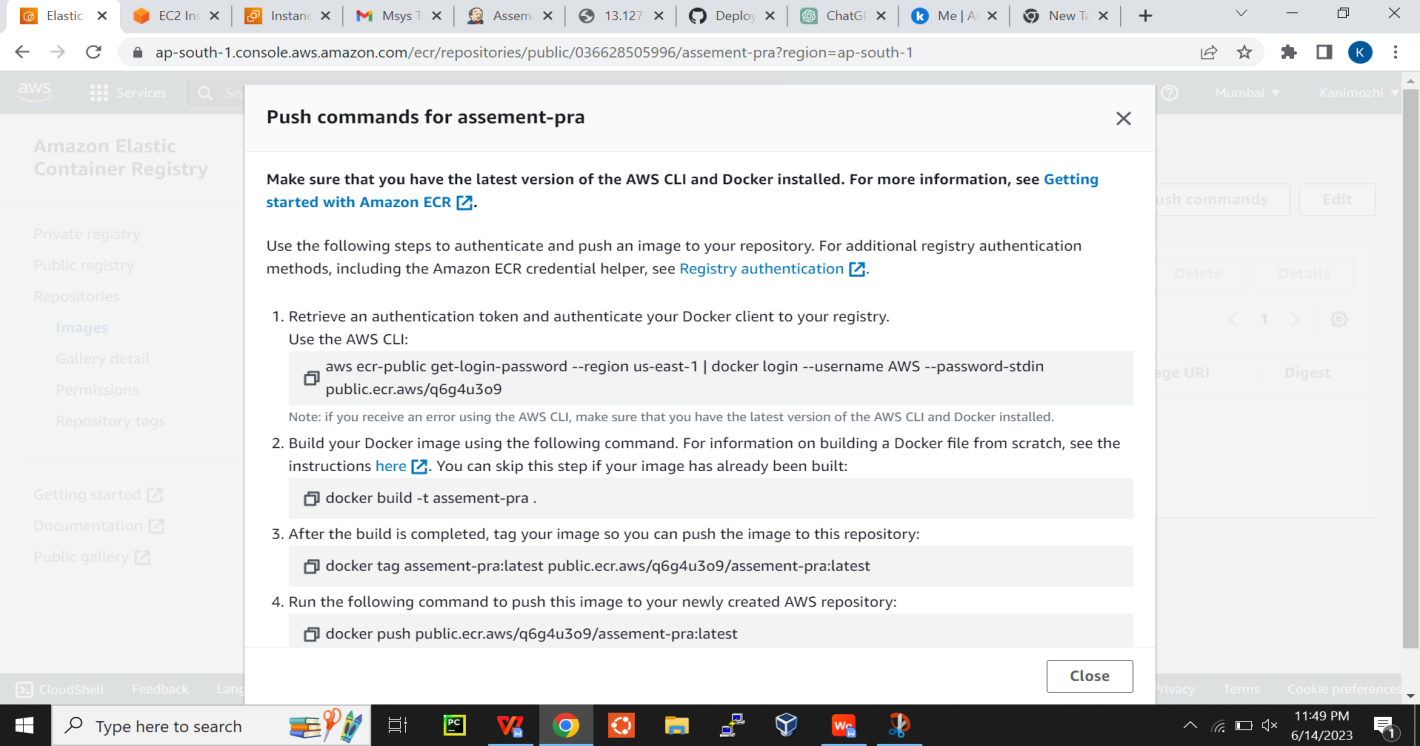
**8.Create ECR Repository**



**9.Create Public Repository**



**10. Viewing the Pushing Commands**



1. **Write DSLPipeline Build pipeline**

DSL Pipeline:

pipelineJob('BuildPipeline') {

  definition {

    cps {

      script("""

        node {

          stage('Build') {

            git url:'https://github.com/Kani1997/FinalProject.git',branch:'main'

            }

          stage('Ecr Login'){

                sh('aws configure set aws\_access\_key\_id AKIAZ4ET6RZOUVRNVK6I')

                  sh('aws configure set aws\_secret\_access\_key dBd9rFu0hWVvw0wb5rcoUIvLL4ltJBsCaoSU6JJ3')

                sh ('AWS\_DEFAULT\_REGION="ap-northeast-1"')

                sh ('aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws/k2z5q5p0')

                }

          stage('Docker'){

                sh 'docker build -t myrepos .'

            }

          stage('docker push'){

                sh 'docker tag myrepos:latest public.ecr.aws/k2z5q5p0/myrepos:latest'

                sh 'docker push public.ecr.aws/k2z5q5p0/myrepos:latest'

                }

           stage('Kubernetes'){

                sh 'minikube delete'

                sh 'minikube start'

                sh 'kubectl apply -f deployment.yaml'

                sh 'kubectl get pods'

                sh 'kubectl get nodes'

          }

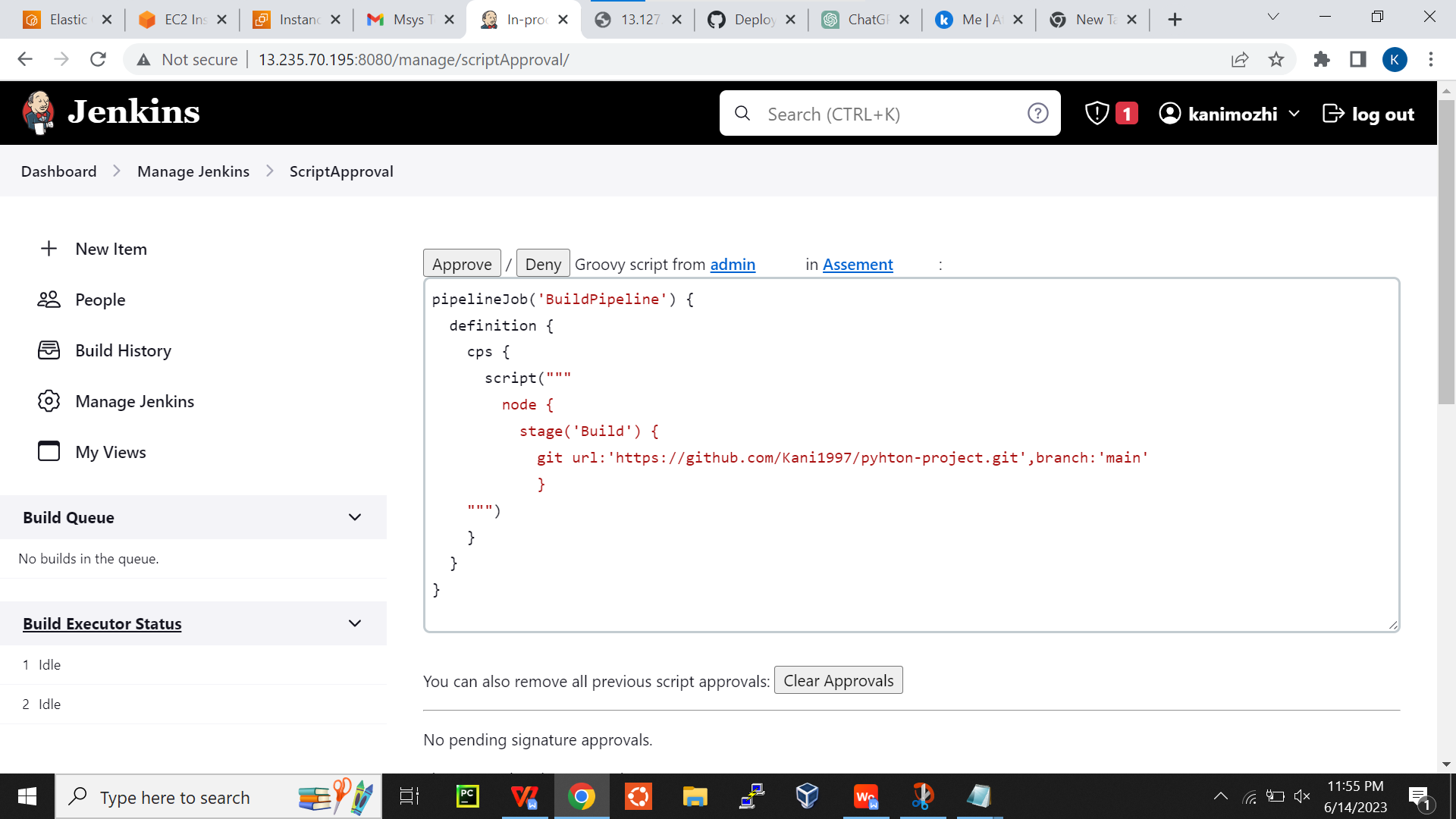
        }

    """)

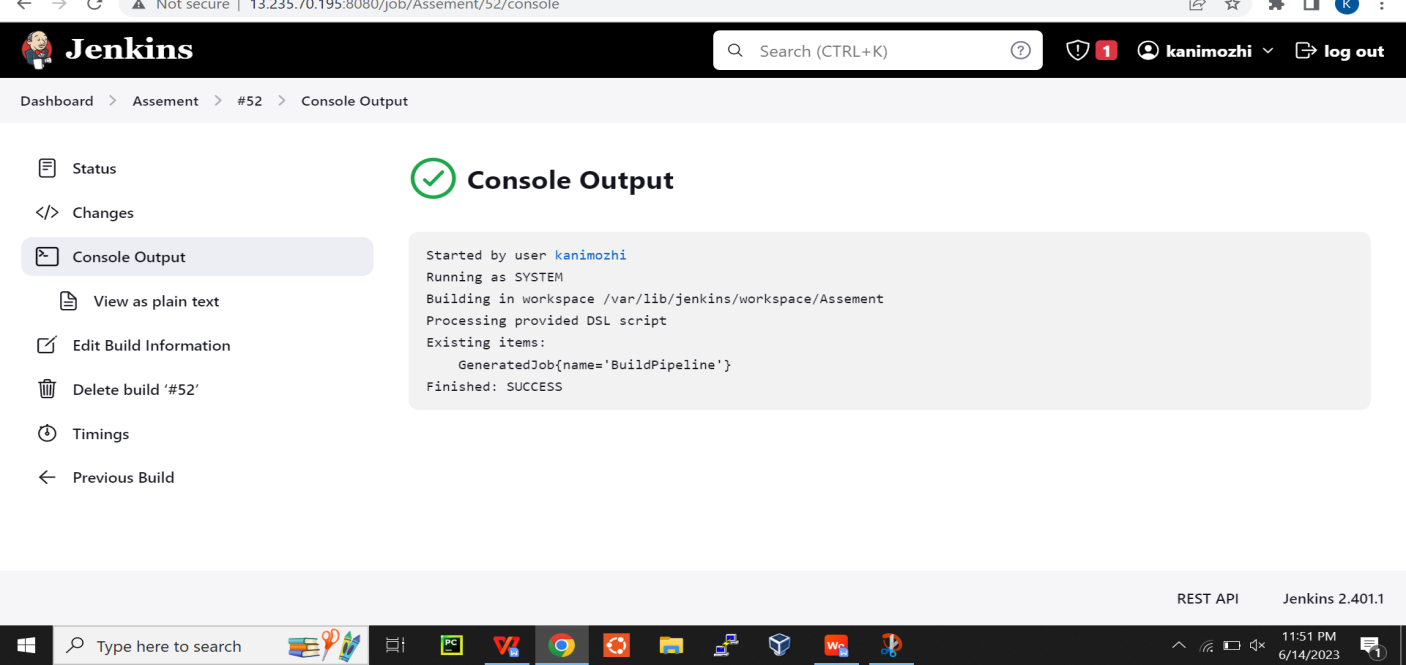
    }

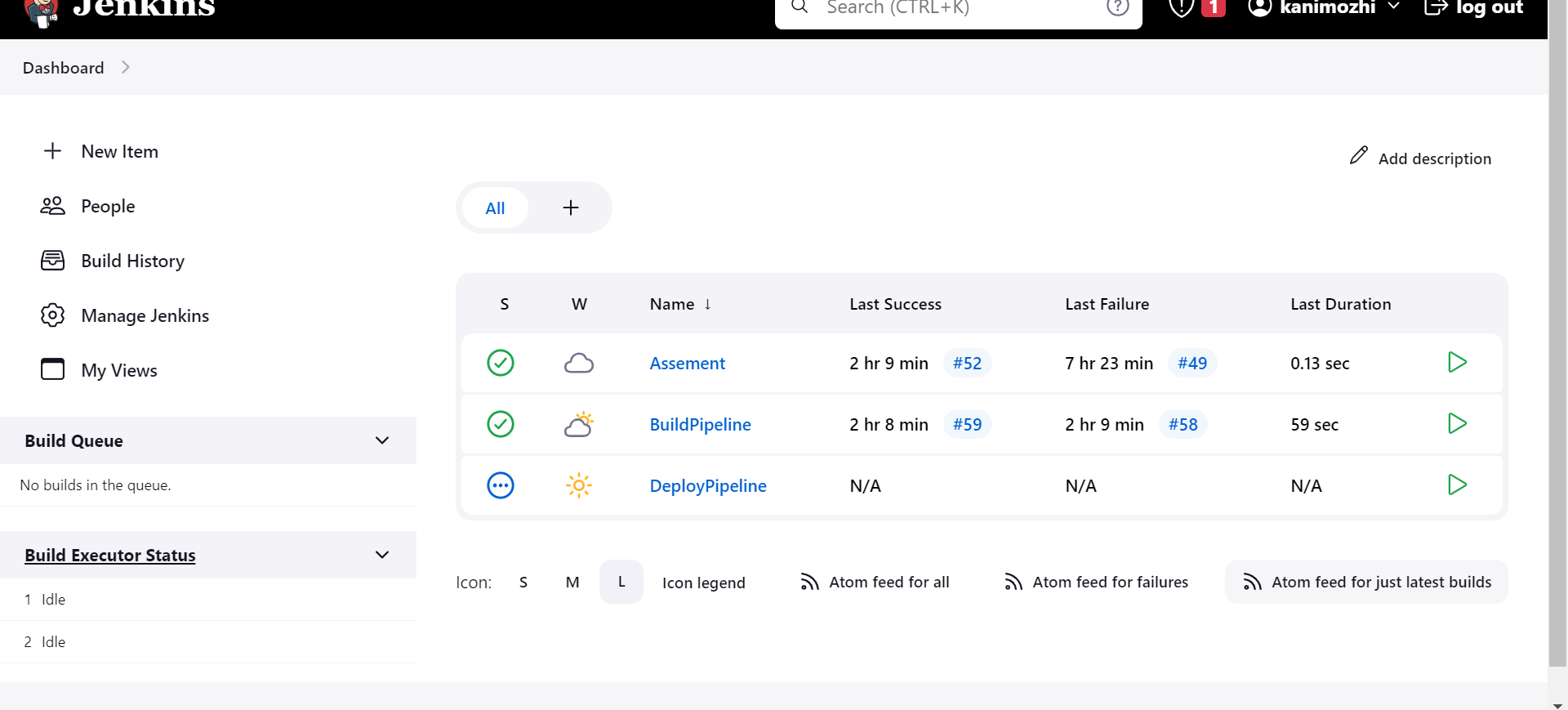
  }

}



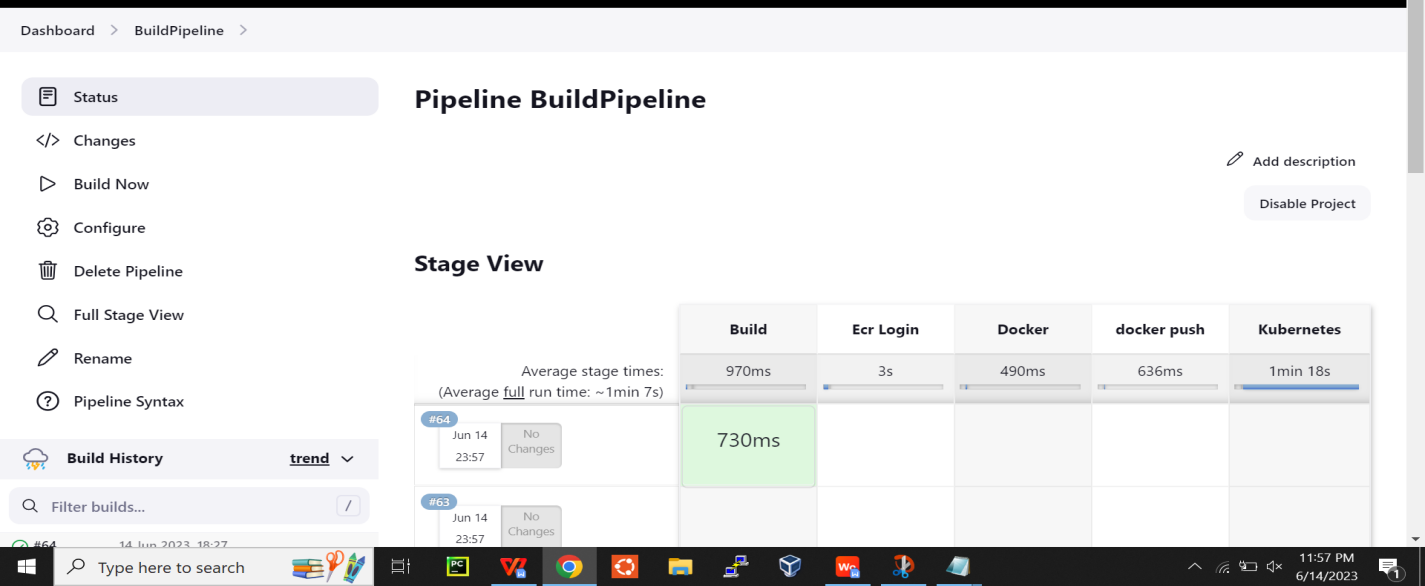
**12.Build pipeline successfully**



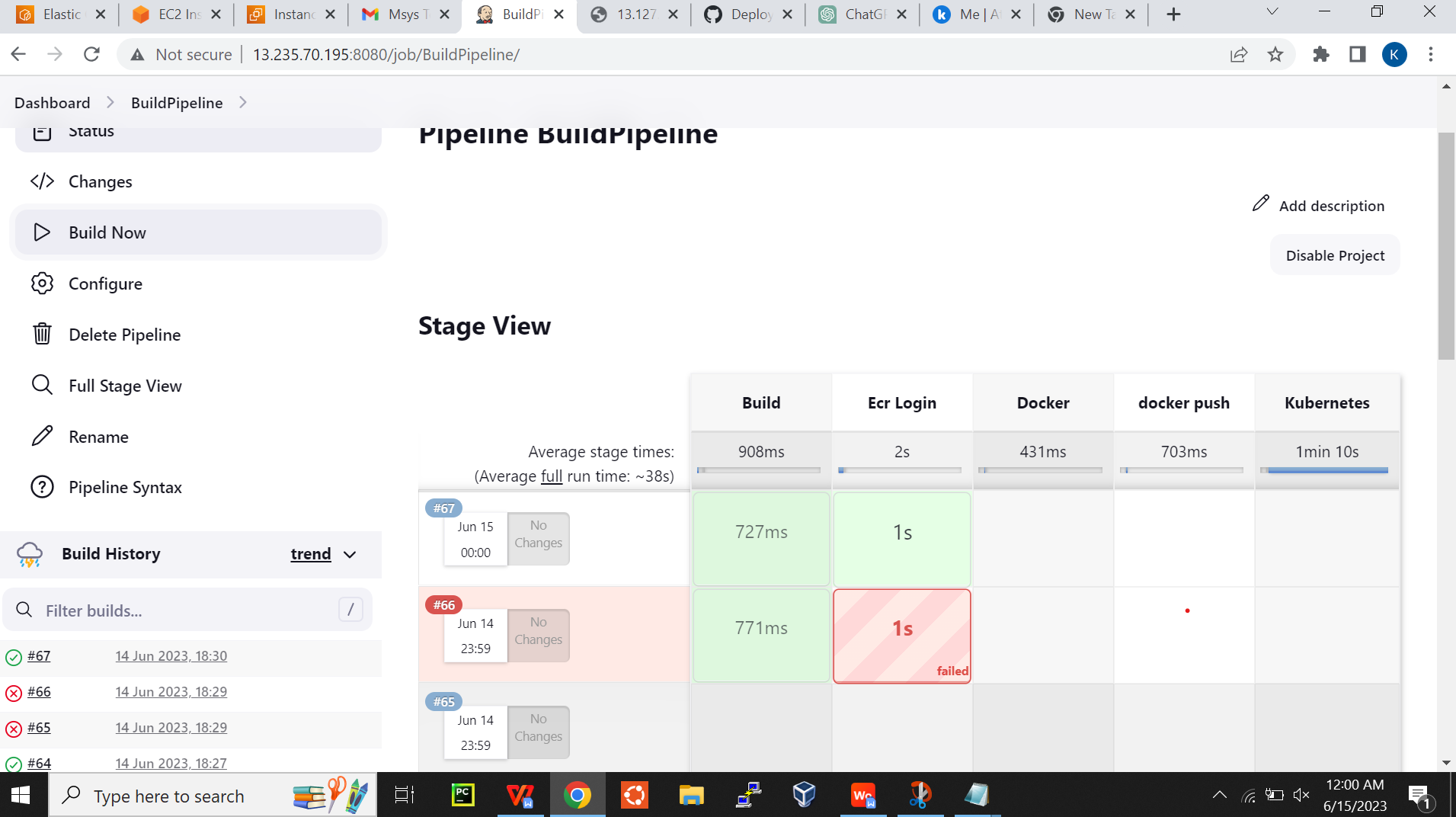


**13.Viewing stage of build pipeline**

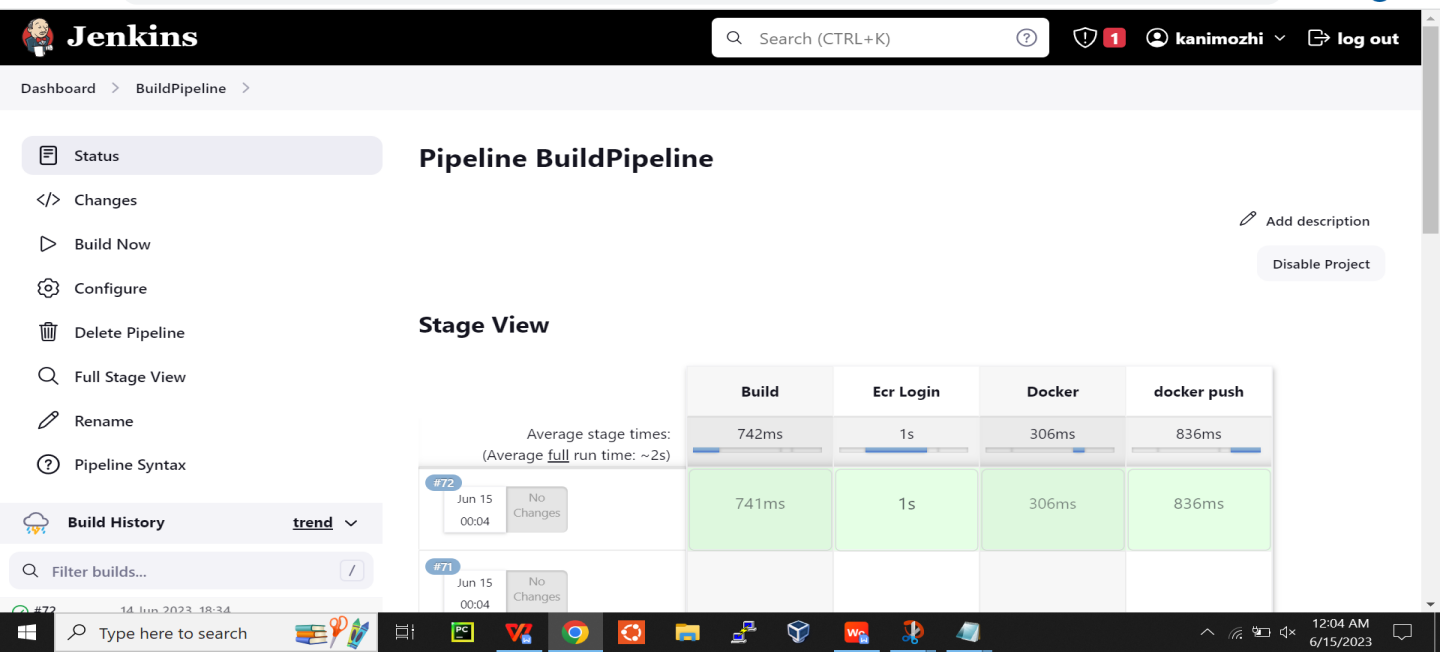
**Notes:** After Build Steps Completed go t o ec2 instance terminal give permission, go to cd/var/run -> sudo chmod 777 docker.sock.



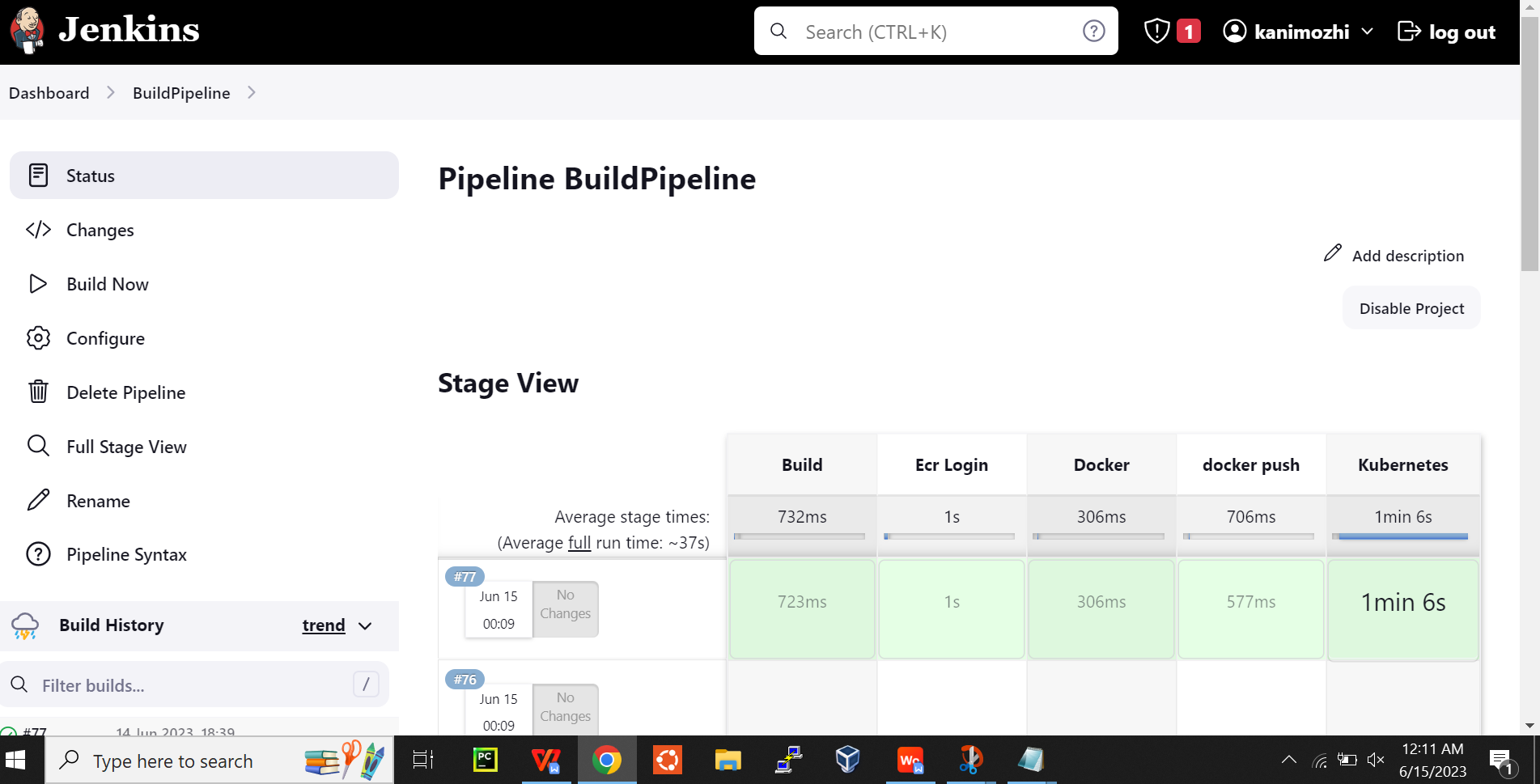
**14.ECR login Build stage view**



**15.Docker Build image Pushing to ECR**



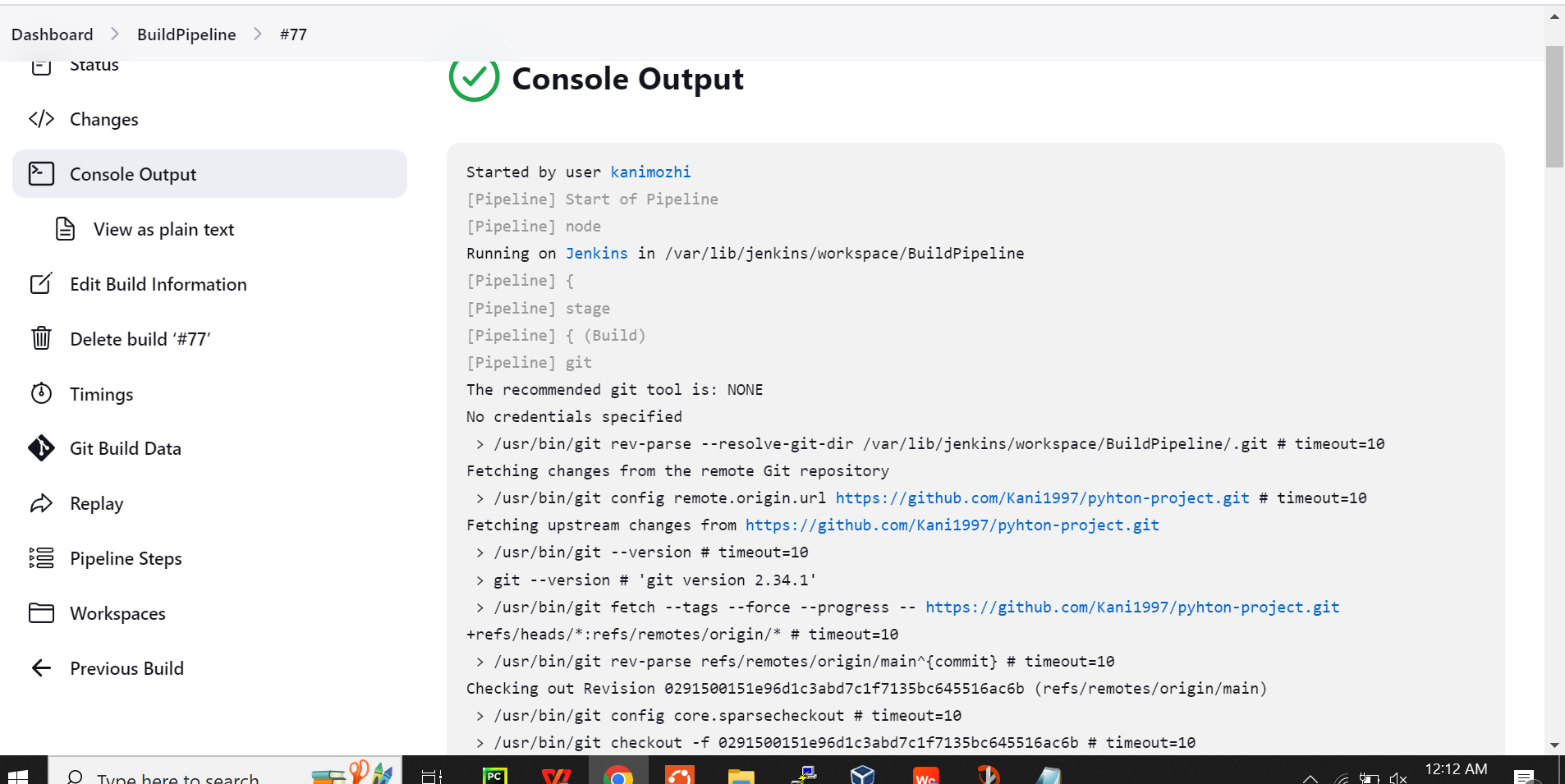
**16.Kubernetes creation**

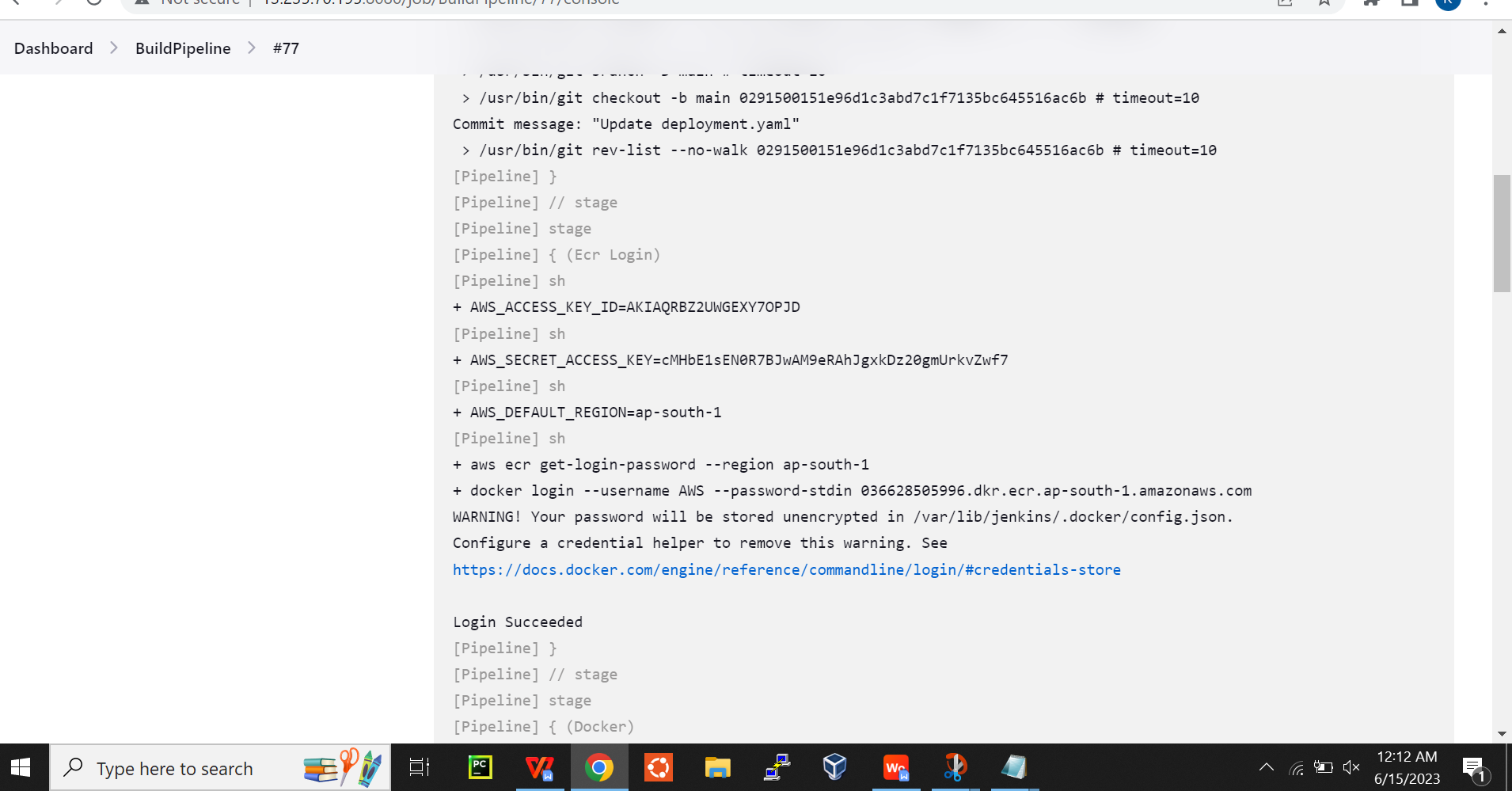


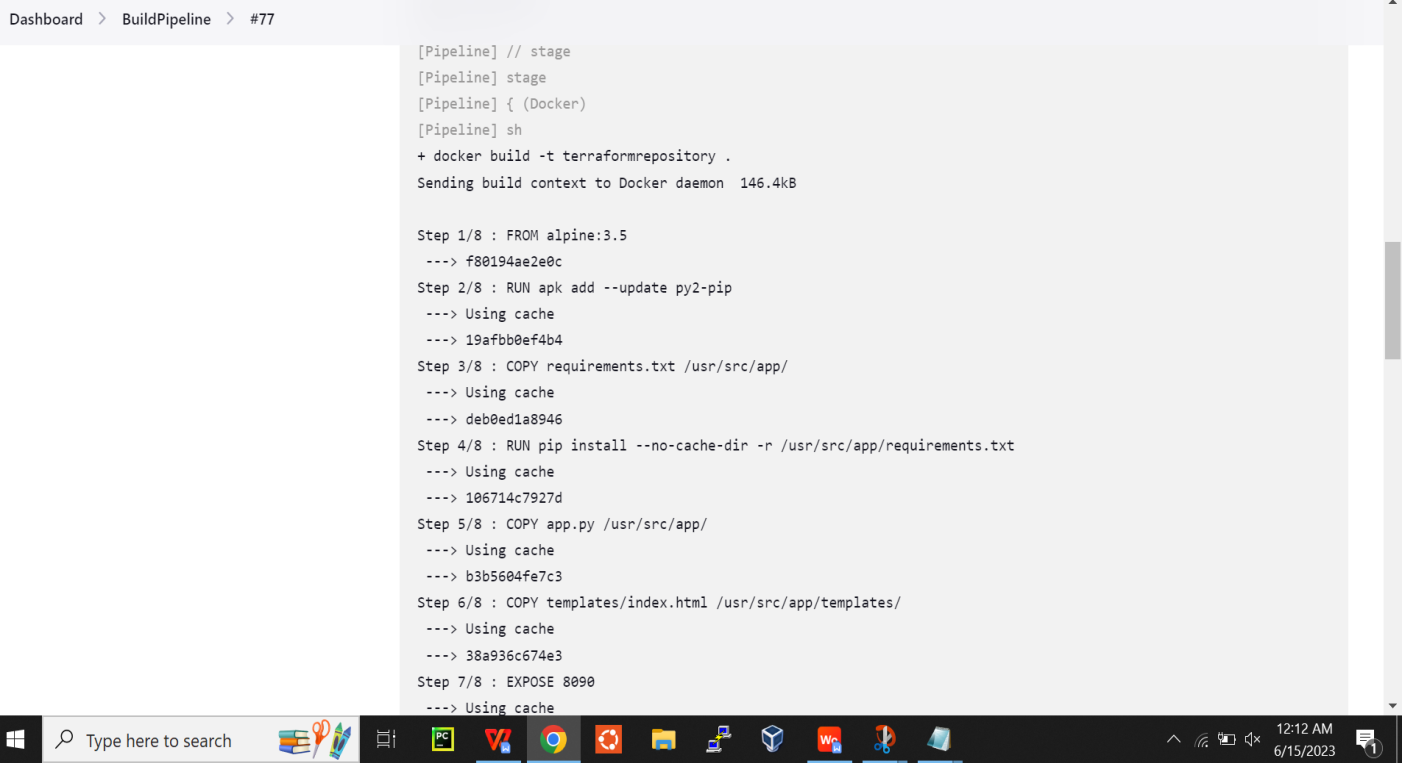
**17.Kubernetes Deployment Checking in EC2 Instance**

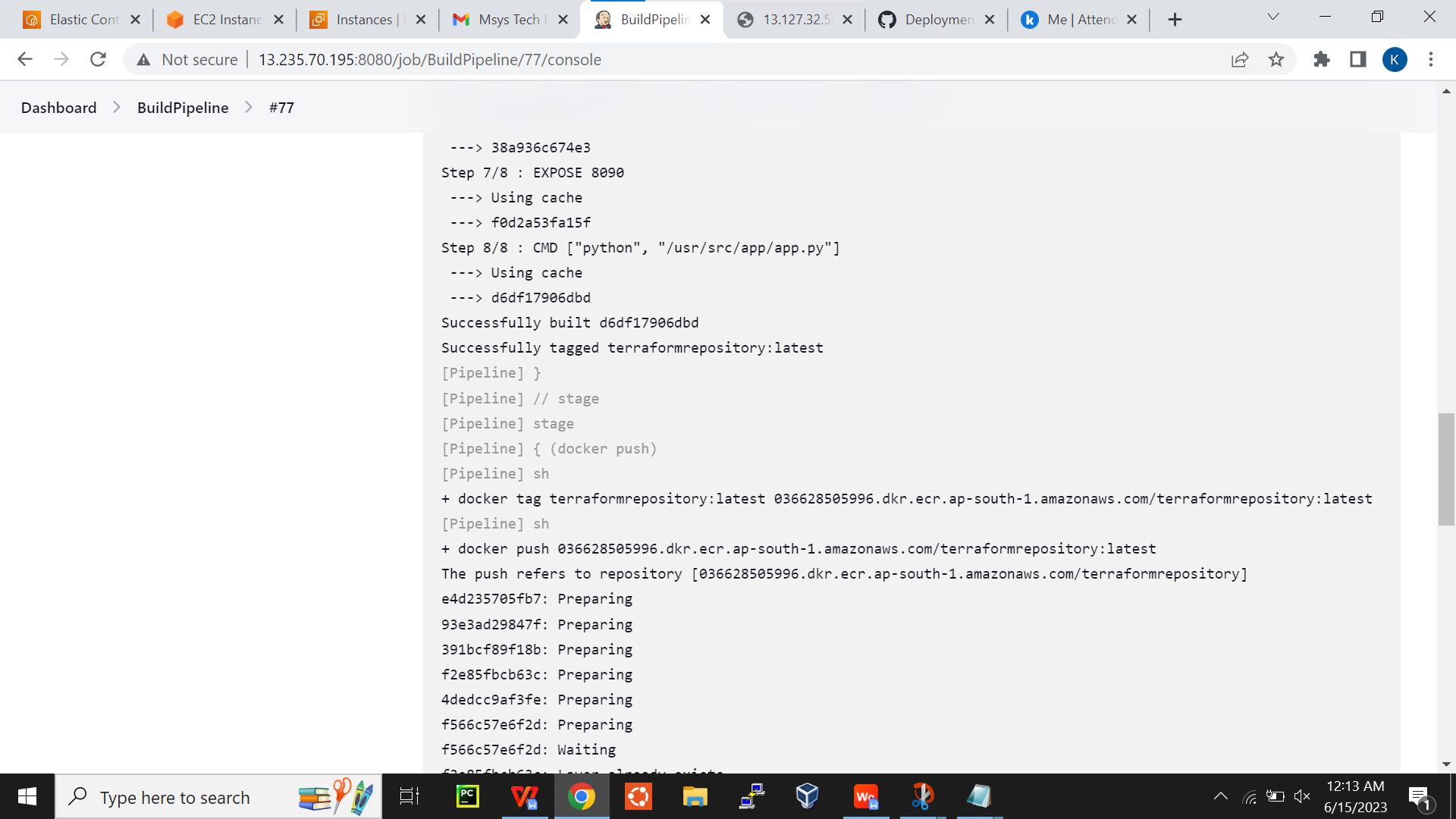
* Go to Ec2 Terminal
* Switch Root user
* Switch to su jenkins root user
* Check the Deployment file and svc File
* Kubect get (pods, svc, nodes)
* Change the port forwarding commands to jenkins root user
* Kubectl port-forward --address 0.0.0.0 svc/svc name 8081:8081
* Copy paste public ip address in web give to port number 8081, it will be showing our output on web pages.

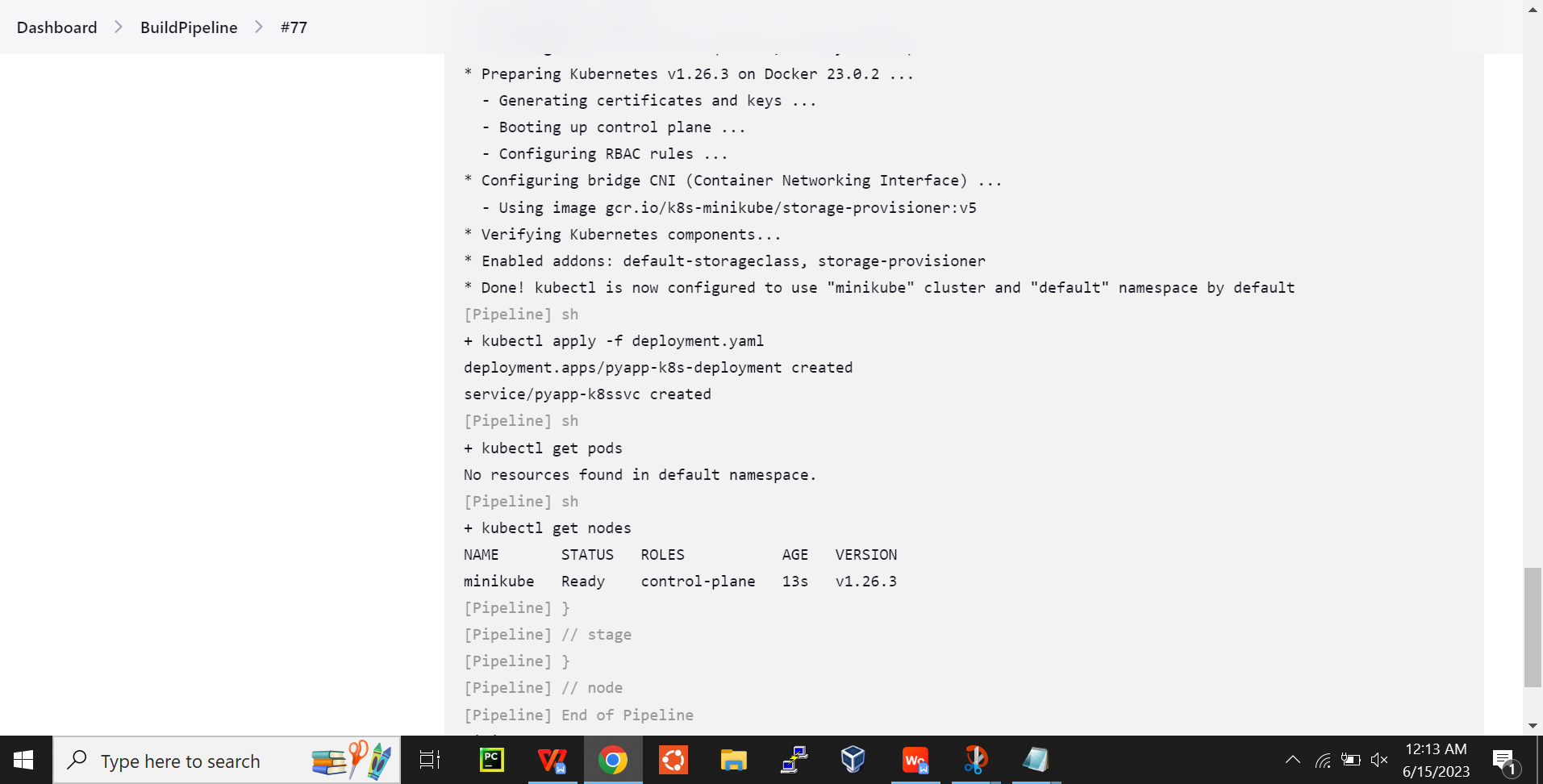
**Console Output:**

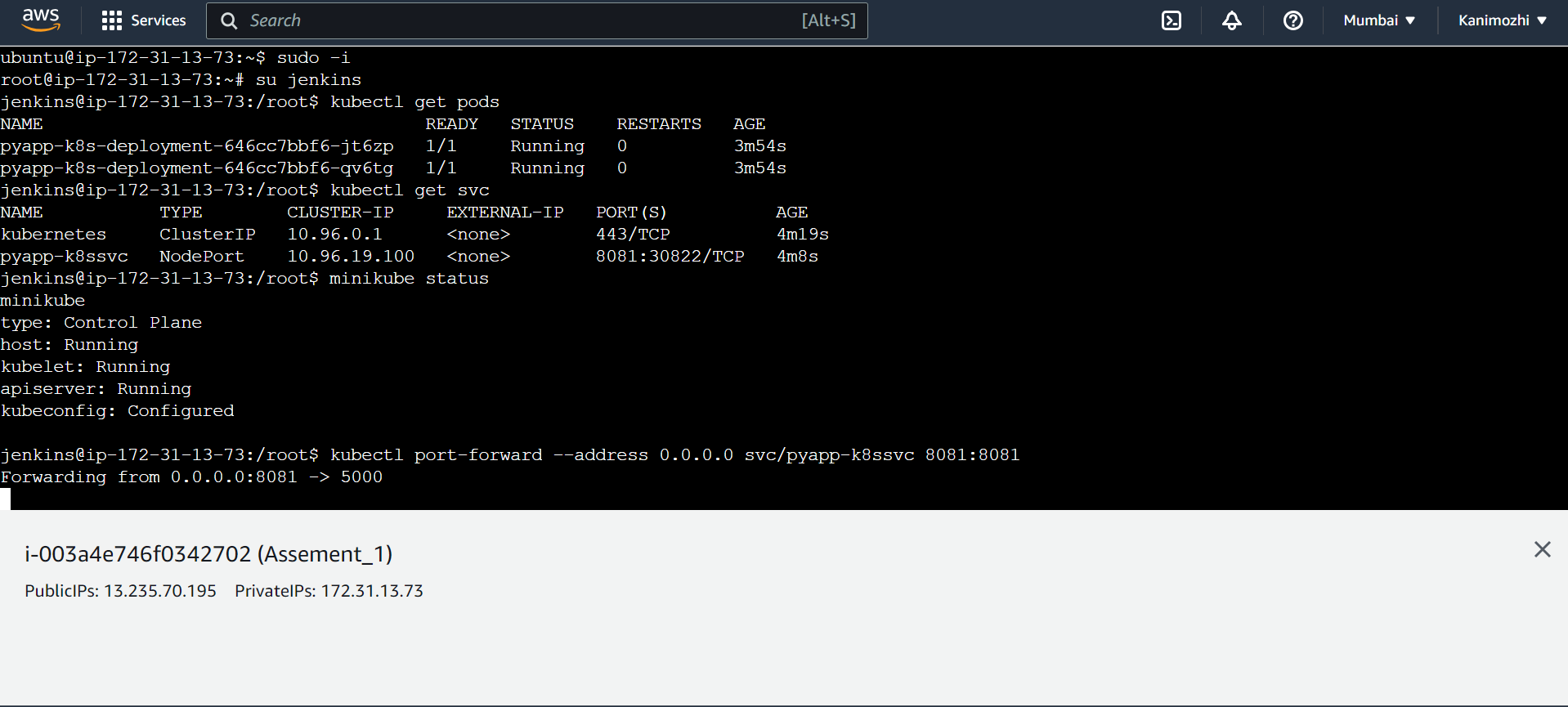












Output:

