**Project 3: Enterprise Java CI/CD with Jenkins, GitOps & Kubernetes Observability (Lite)**

# Overview

This document summarizes my end-to-end CI/CD setup for a Java Spring Boot app using Jenkins, Docker, SonarQube, Kubernetes (Minikube/K3s), and Argo CD (Operator).

# Repository

[*https://github.com/GauravBhardwaj10/Jenkins-Zero-To-Hero*](https://github.com/GauravBhardwaj10/Jenkins-Zero-To-Hero)

# Services Used & What They Do

• AWS EC2 Instance – Ubuntu VM that hosts Jenkins, SonarQube, Docker, and Kubernetes tools.

• EC2 Security Group – Inbound rules for 22 (SSH), 8080 (Jenkins), 9000 (SonarQube), NodePort/Ingress.

• Jenkins – CI server building with Maven, scanning via SonarQube, building/pushing Docker images, and updating manifests (GitOps).

• GitHub – Source repo for code, Jenkinsfile, and Kubernetes manifests.

• Docker – Container builder and image registry (Docker Hub).

• kubectl – CLI to manage Kubernetes clusters (Minikube or K3s).

• Argo CD Operator – Installs and manages Argo CD for GitOps sync.

• SonarQube – Static code analysis platform used during CI.

# Process I Followed

1) Cloned the repo (Jenkins-Zero-To-Hero).

2) Created an EC2 instance + Security Group.

3) Installed Docker, kubectl, and Minikube (or K3s).

4) Installed Jenkins and configured Jenkins URL.

5) Installed SonarQube and generated an analysis token.

6) Added Jenkins credentials: docker-cred, github (PAT), sonarqube (token).

7) Ran pipeline: Maven build, Sonar scan, Docker build/push, GitOps update.

8) Installed Argo CD Operator from OperatorHub and created an ArgoCD instance.

9) Applied k8s manifests and validated rollout in Argo CD UI.

# Key Commands

## Clone repo

sudo apt-get update -y  
sudo apt-get install -y git  
git clone https://github.com/GauravBhardwaj10/Jenkins-Zero-To-Hero  
cd Jenkins-Zero-To-Hero

## Docker

sudo apt-get install -y docker.io ca-certificates  
sudo systemctl enable --now docker  
sudo usermod -aG docker $USER  
newgrp docker  
docker ps

## kubectl + Minikube (Option A)

sudo curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg --dearmor -o /usr/share/keyrings/kubernetes-apt-keyring.gpg  
echo "deb [signed-by=/usr/share/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /" | sudo tee /etc/apt/sources.list.d/kubernetes.list  
sudo apt-get update -y && sudo apt-get install -y kubectl conntrack  
  
curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64  
chmod +x minikube && sudo mv minikube /usr/local/bin/  
minikube start --driver=docker --cpus=2 --memory=4g  
kubectl get nodes -o wide

## K3s (Option B)

curl -sfL https://get.k3s.io | sh -  
mkdir -p $HOME/.kube  
sudo cp /etc/rancher/k3s/k3s.yaml $HOME/.kube/config  
sudo chown $(id -u):$(id -g) $HOME/.kube/config  
kubectl get nodes

## Jenkins

sudo apt-get install -y openjdk-17-jre  
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 -y  
sudo apt-get install -y jenkins  
sudo systemctl enable --now jenkins

## SonarQube

sudo apt-get install -y unzip  
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-10.4.1.88267.zip  
unzip sonarqube-10.4.1.88267.zip  
cd sonarqube-10.4.1.88267/bin/linux-x86-64  
./sonar.sh start  
# Generate token in UI and store in Jenkins as Secret Text 'sonarqube'

## Argo CD Operator (OperatorHub)

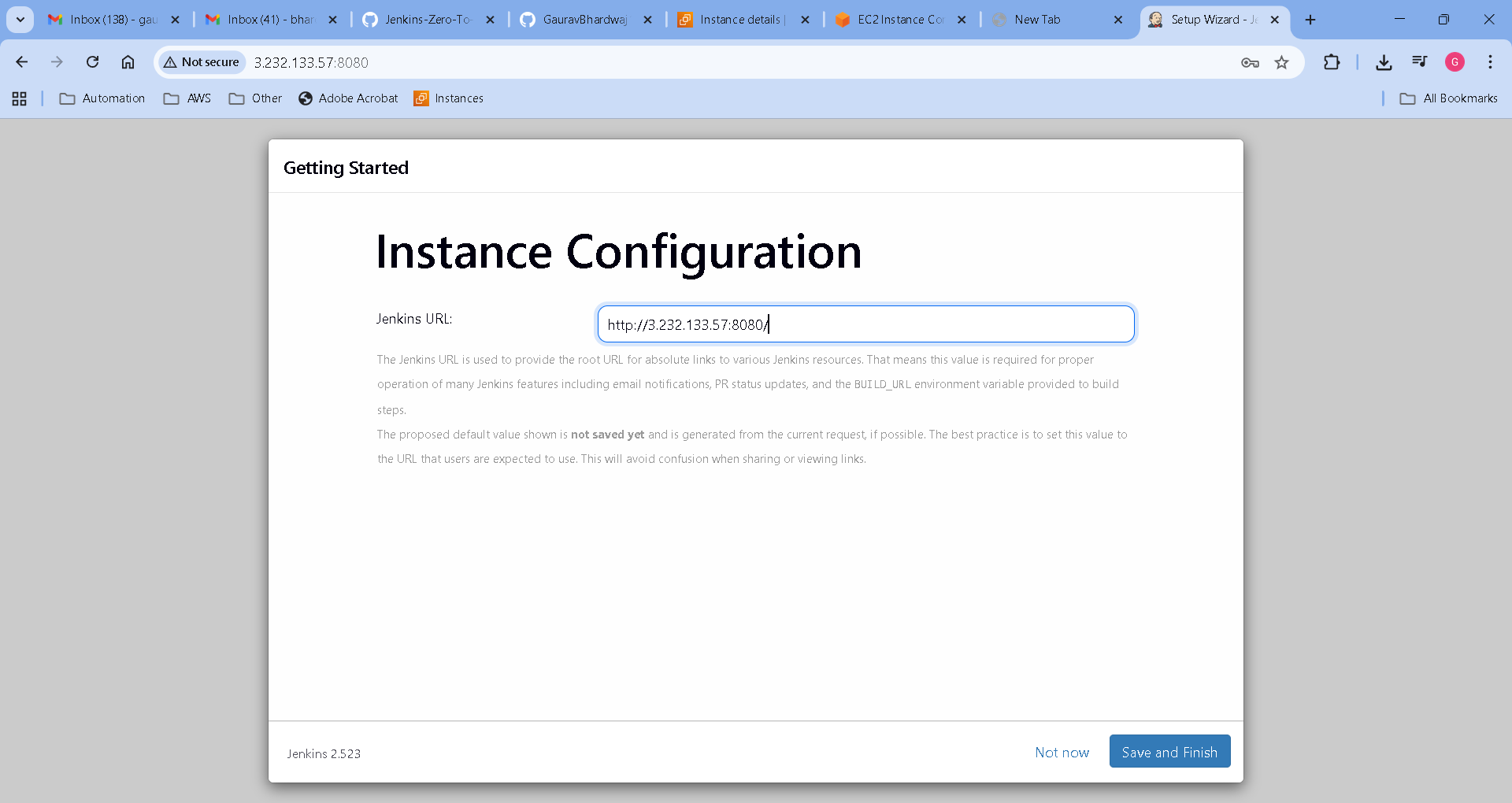
References: https://operatorhub.io/operator/argocd-operator and https://argocd-operator.readthedocs.io/en/latest/install/manual/

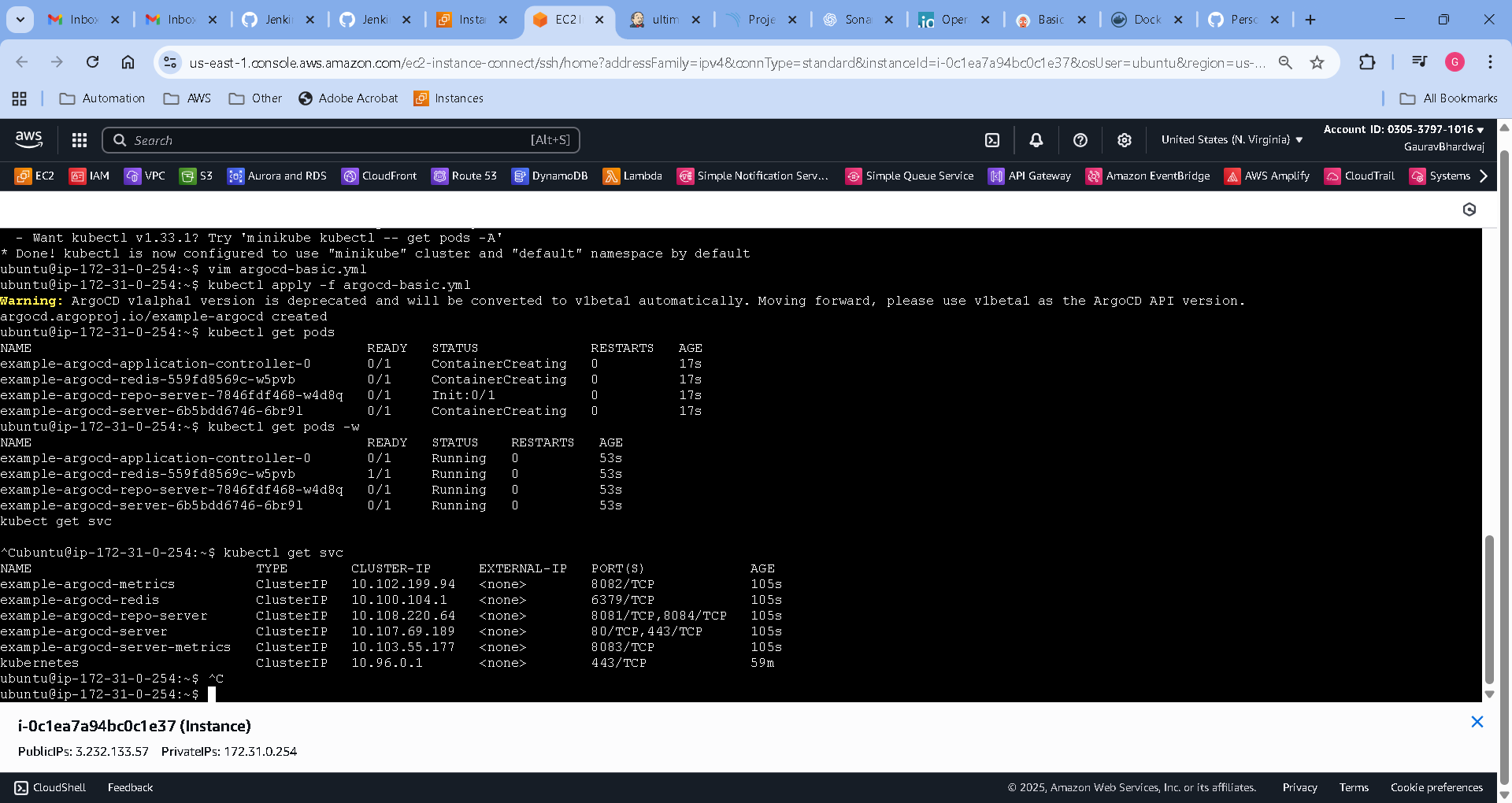
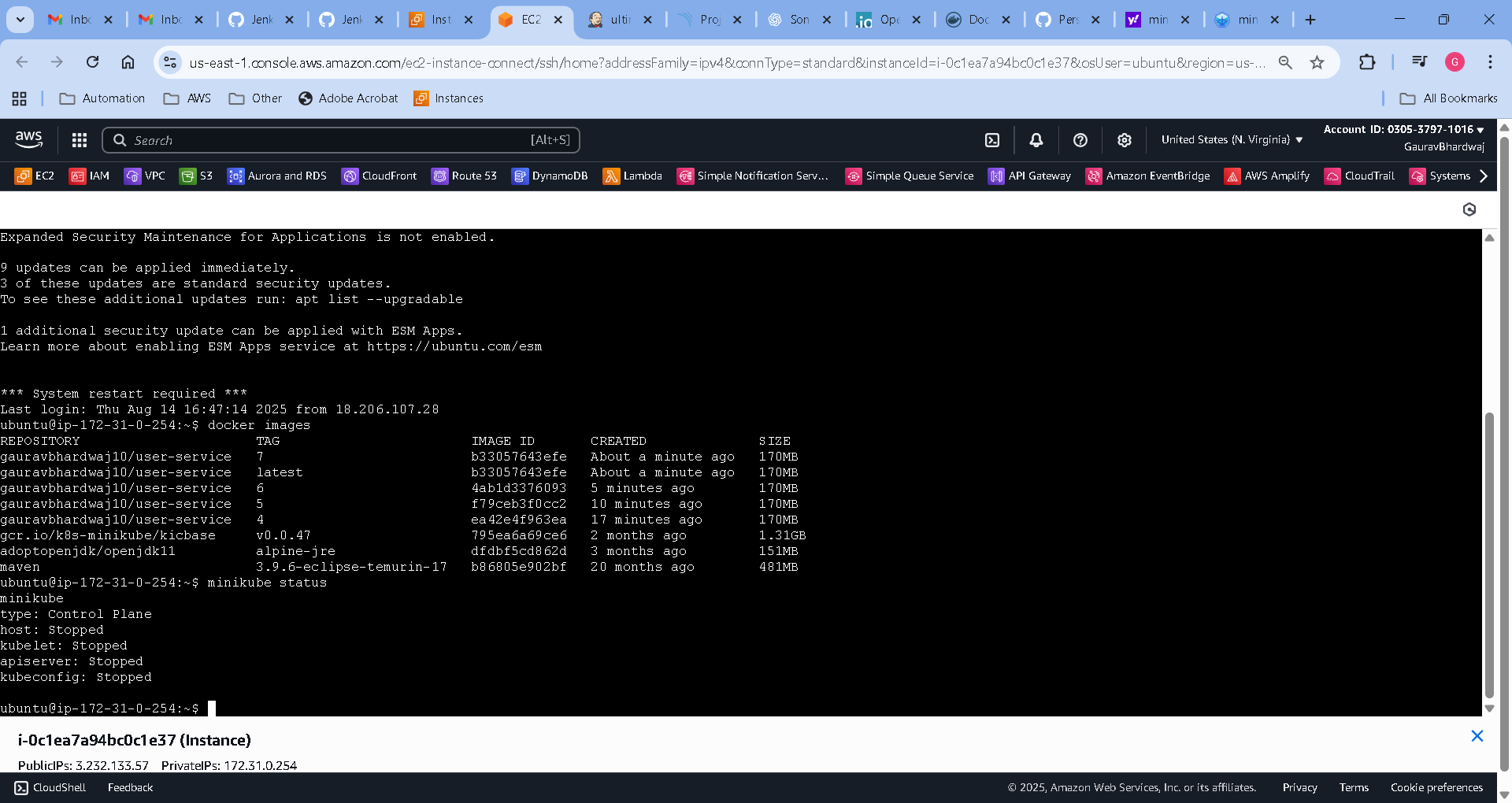
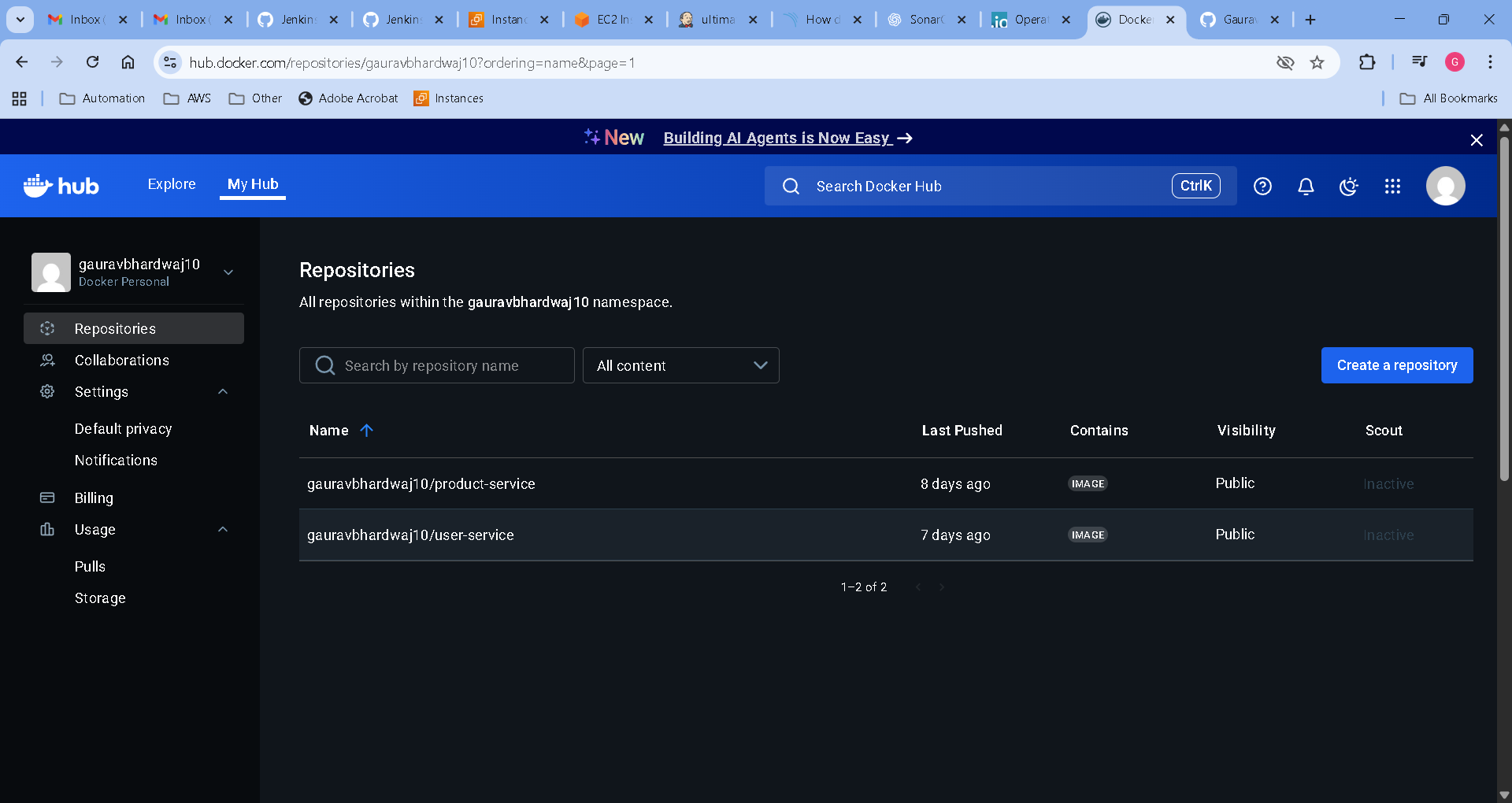
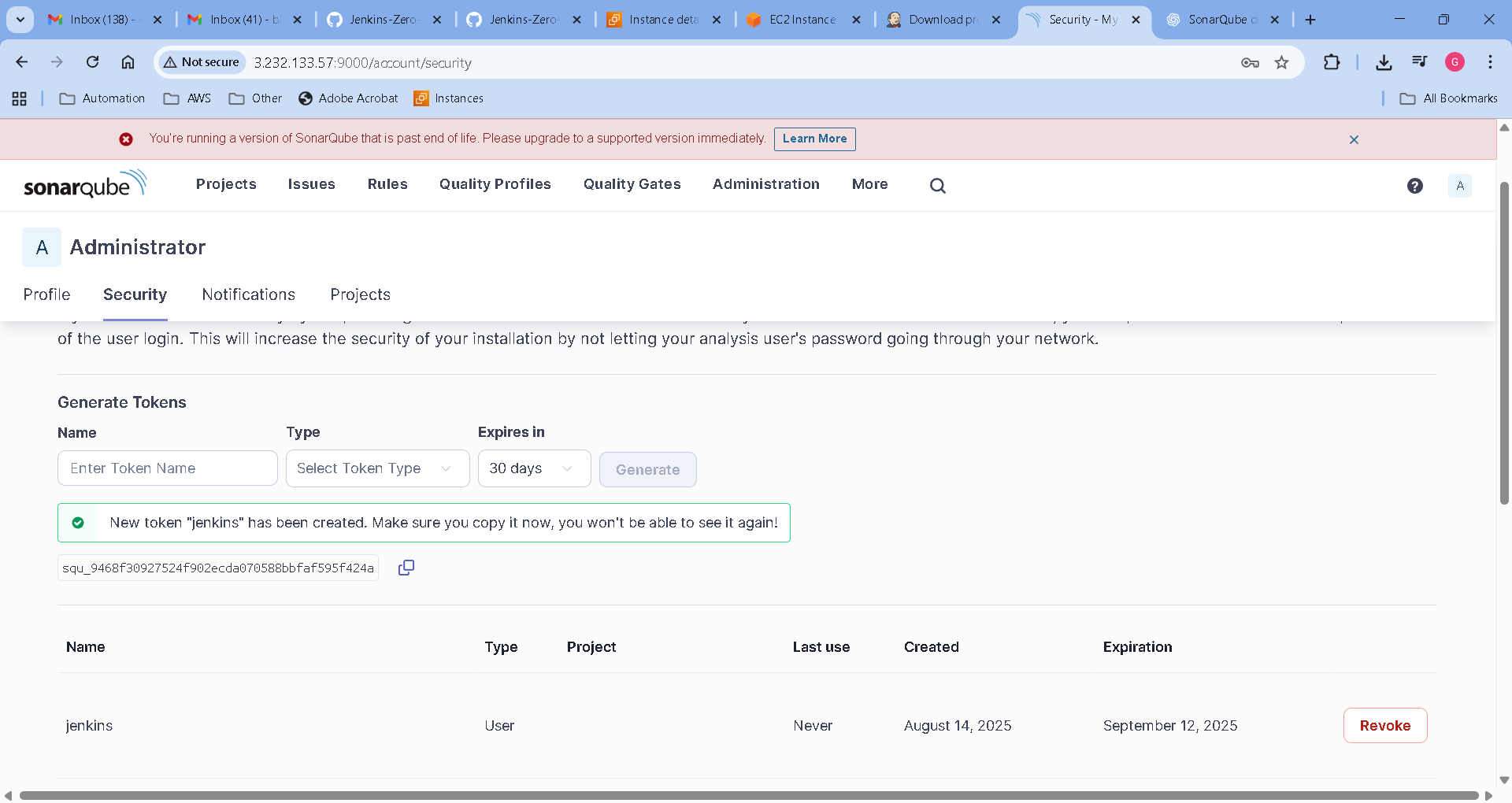
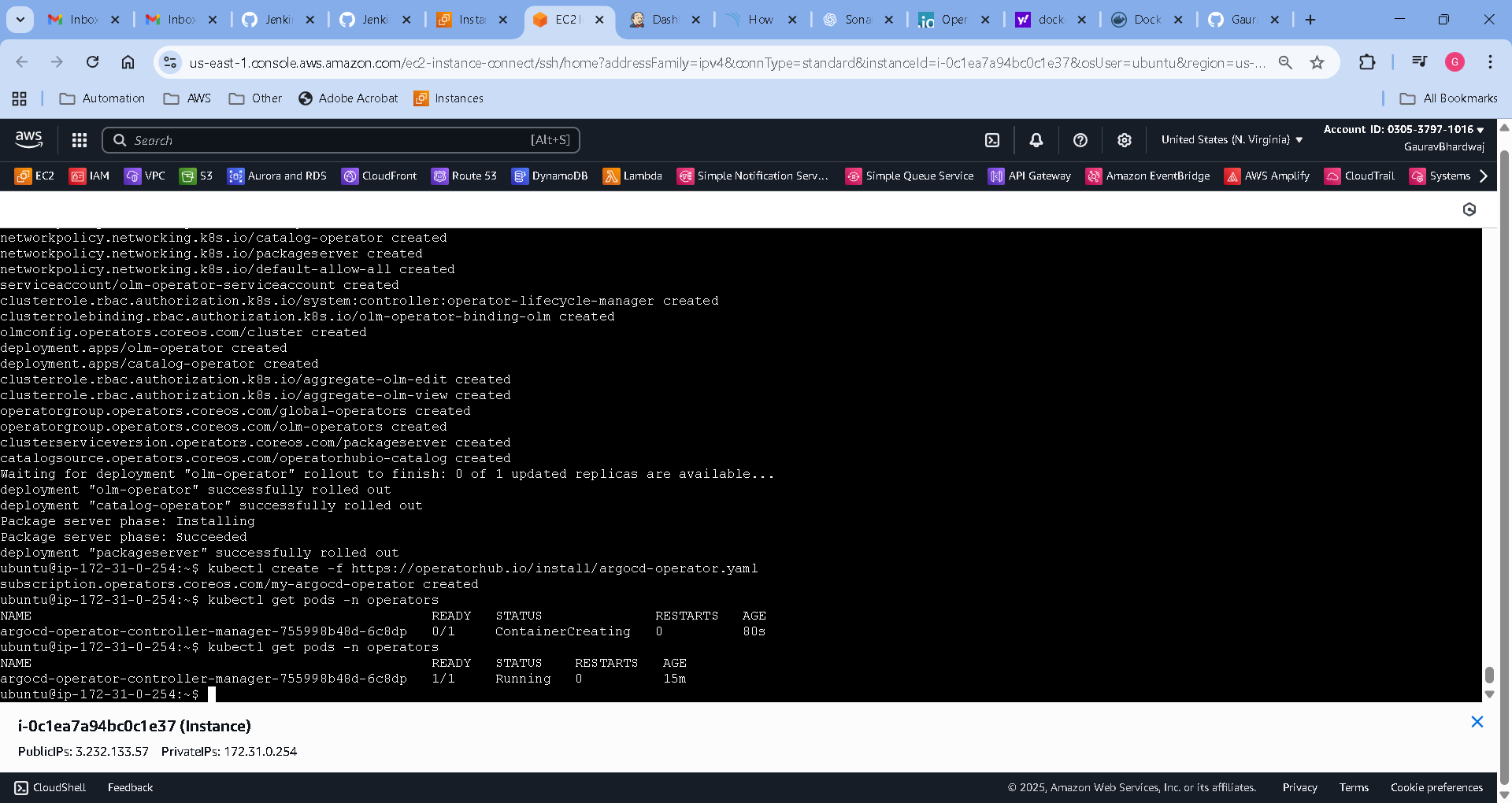
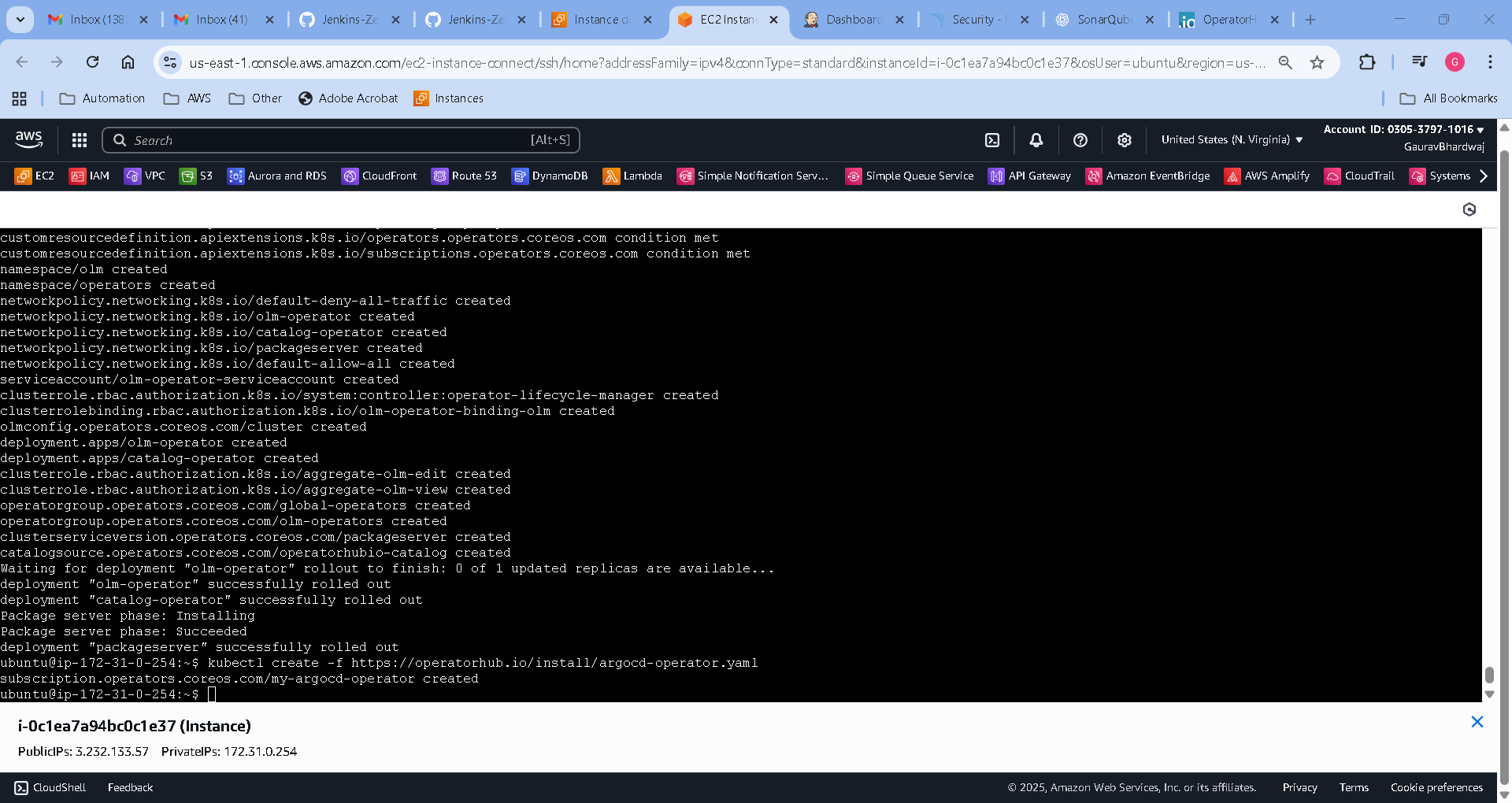
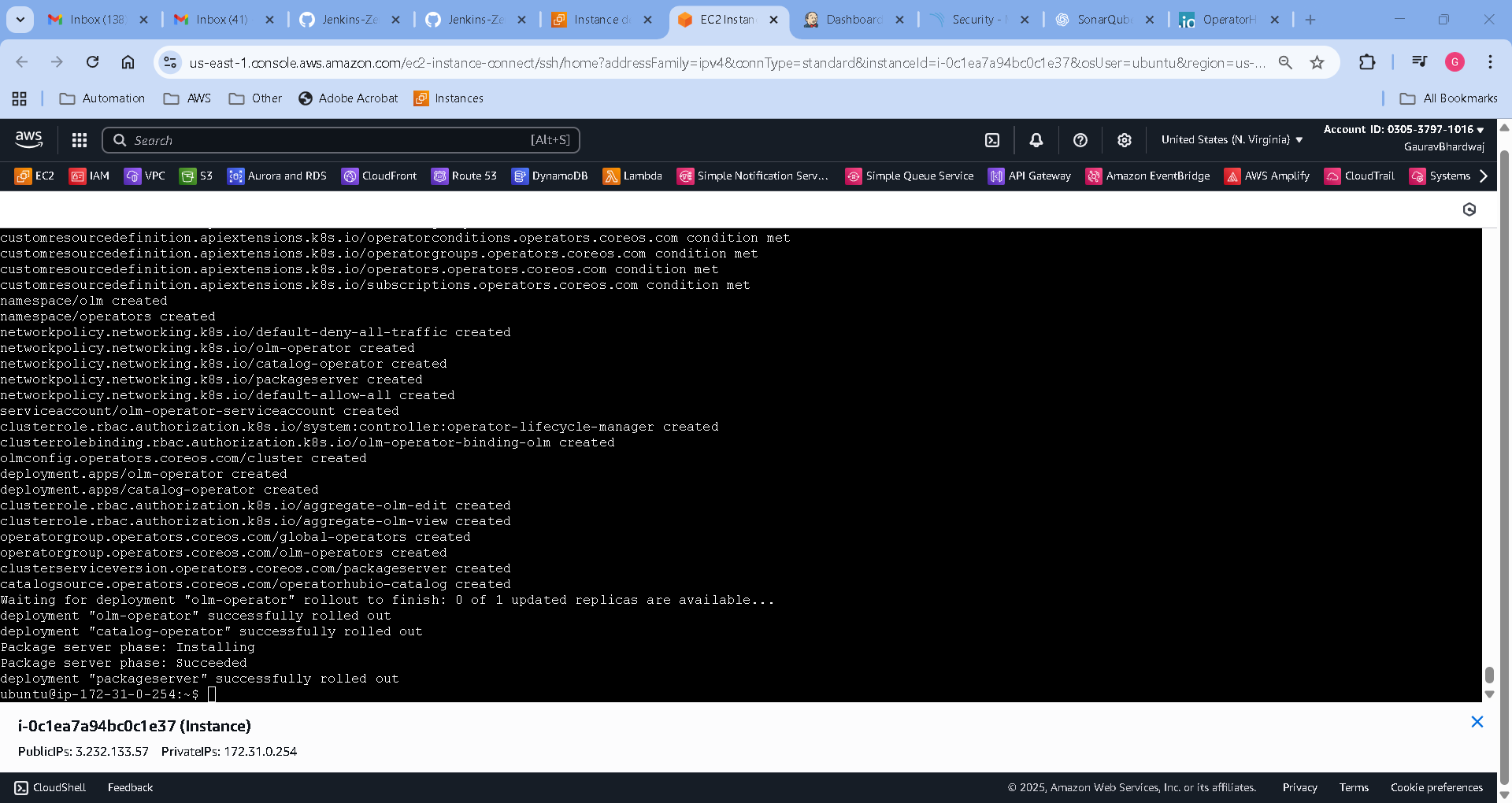
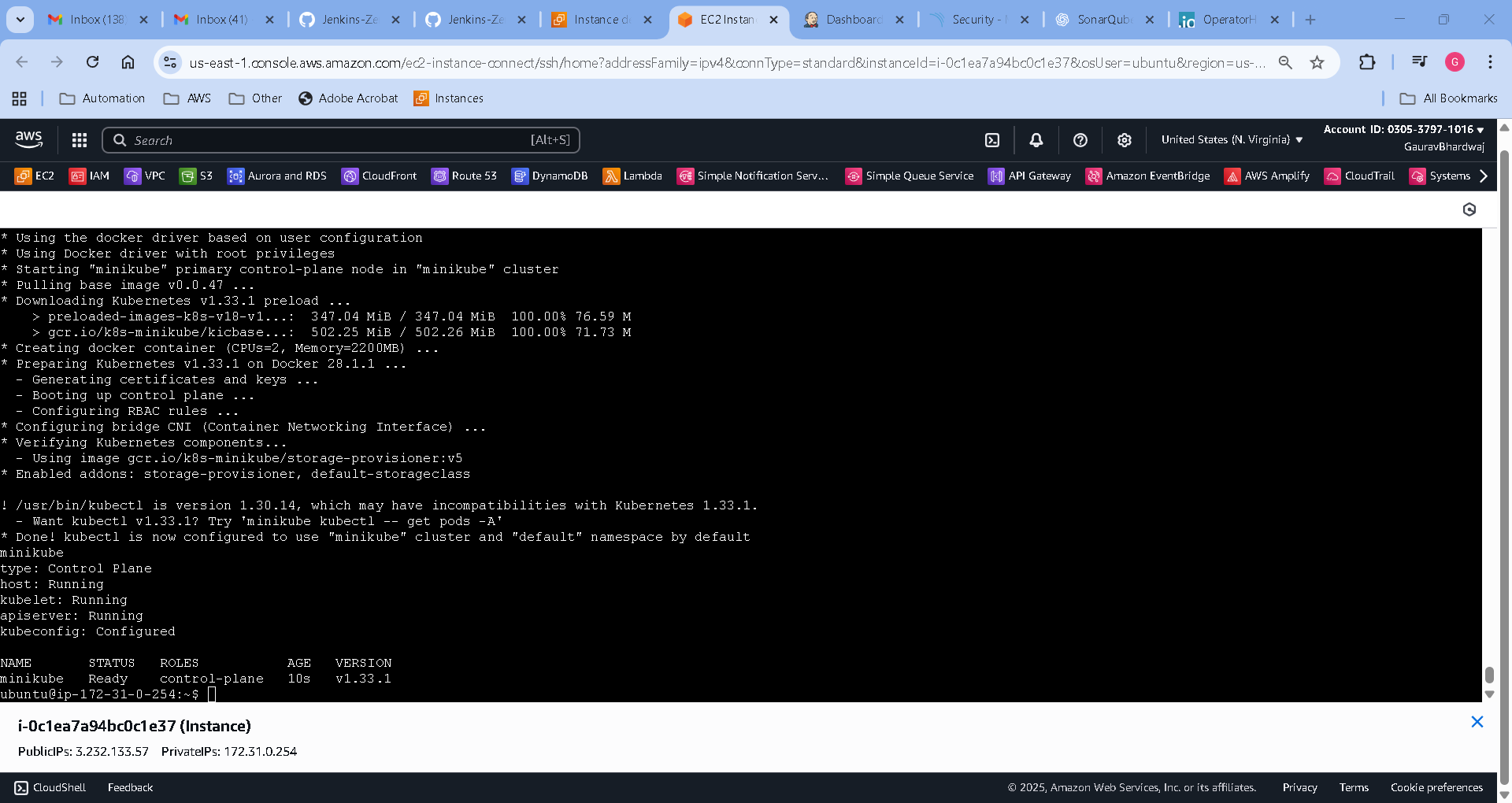
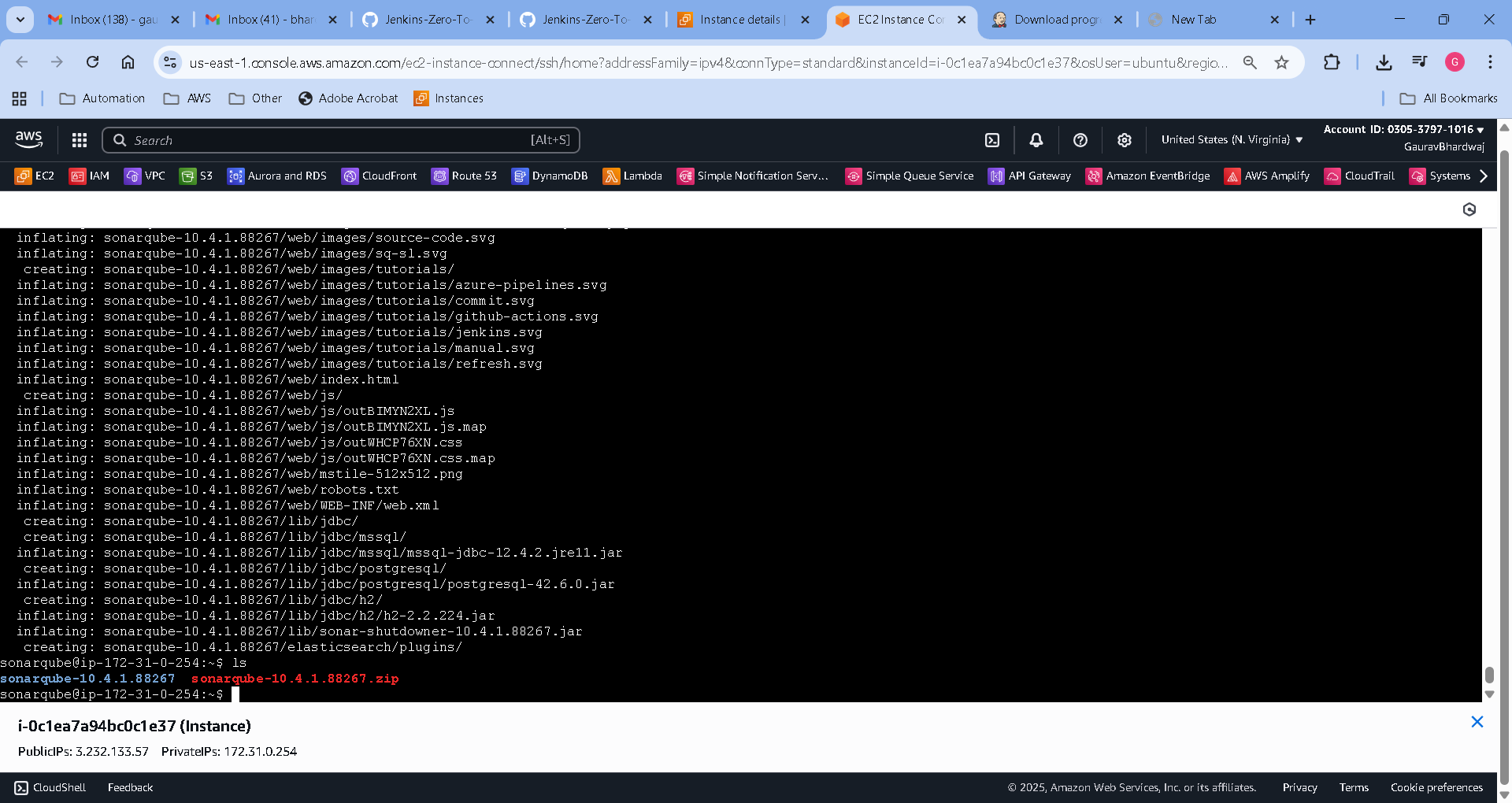
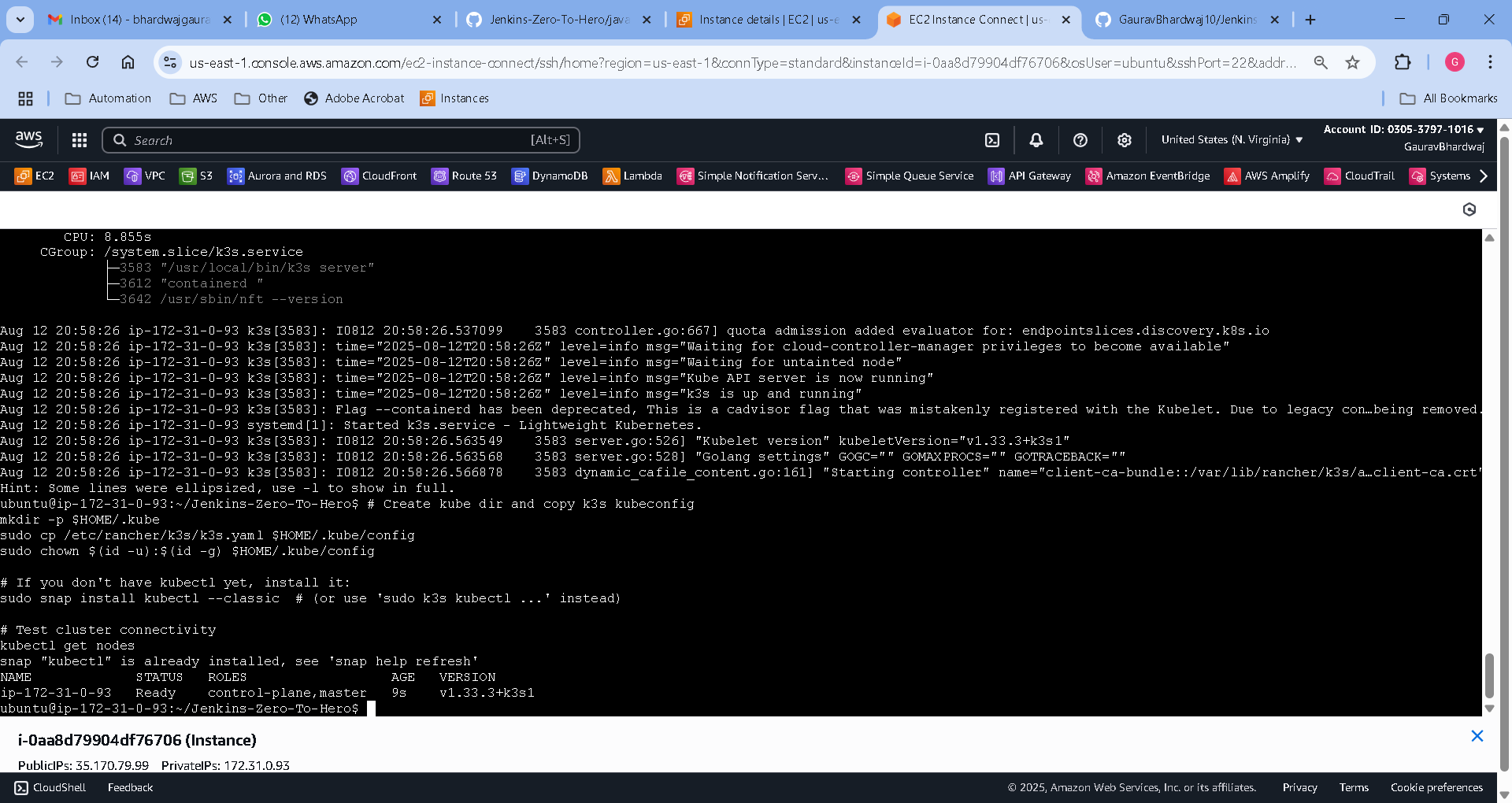
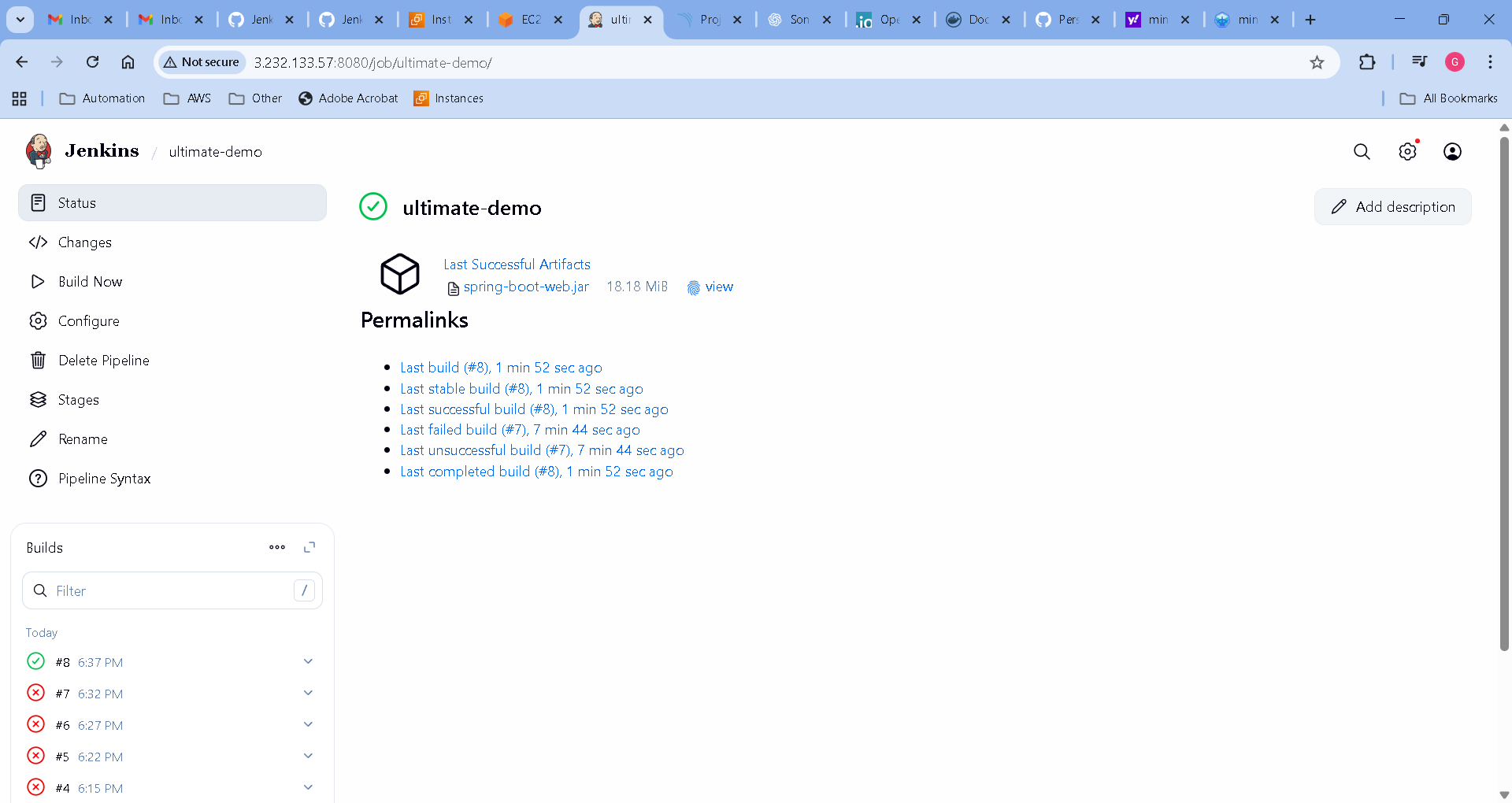
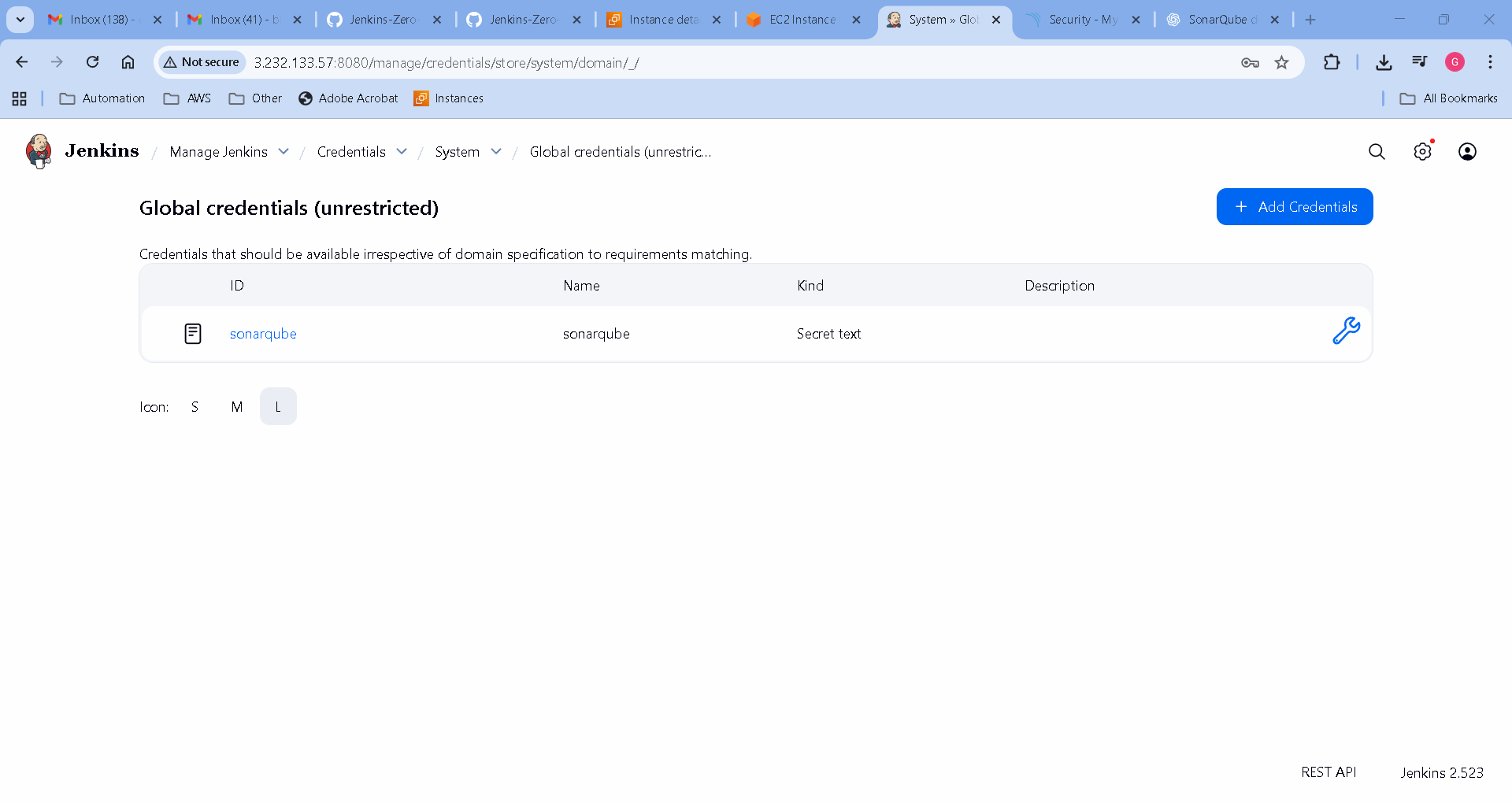
kubectl create -f https://operatorhub.io/install/argocd-operator.yaml  
kubectl get pods -n operators  
kubectl get pods -n olm  
  
kubectl create namespace argocd  
cat <<'EOF' | kubectl apply -f -  
apiVersion: argoproj.io/v1alpha1  
kind: ArgoCD  
metadata:  
 name: my-argocd  
 namespace: argocd  
spec: {}  
EOF  
kubectl get pods -n argocd

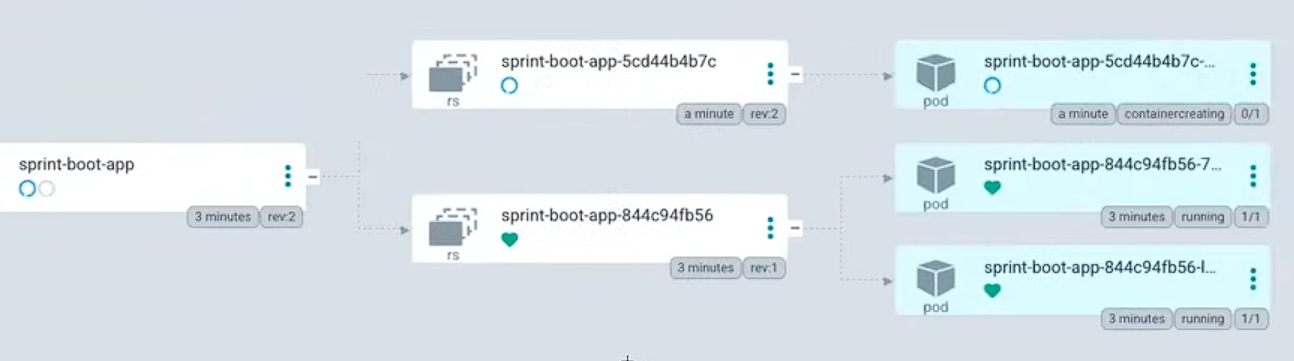
## Deploy app

kubectl apply -f java-maven-sonar-argocd-helm-k8s/spring-boot-app-manifests/  
kubectl get pods -A  
kubectl get svc -A

# Screenshots







# References

[*https://operatorhub.io/operator/argocd-operator*](https://operatorhub.io/operator/argocd-operator)

[*https://argocd-operator.readthedocs.io/en/latest/install/manual/*](https://argocd-operator.readthedocs.io/en/latest/install/manual/)