Instances

Jenkins-master 🡪 ubuntu 24.04 , storage 15 G, t2 Medium

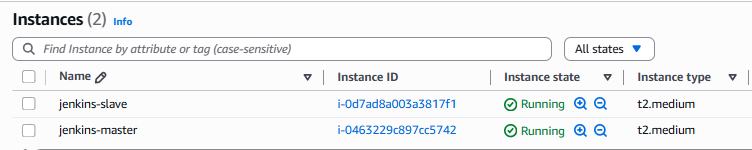
Jenkins-slave -> ubuntu 24.04 , 20G storage , t2medium

Sonarqube -> ubuntu 24.04 , 15G storage , t2medium

Bootsrap-server -> ubuntu 24.04 , 8G storage , t2xmicro

Steps to install Jenkins

**Jenkins Installation Steps**

****

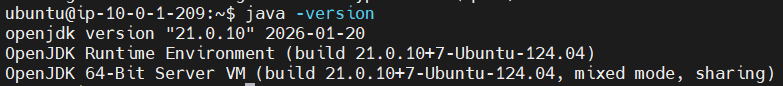
**On master node**

1. **Installation of Java**

sudo apt update

sudo apt install fontconfig openjdk-21-jre

java -version

****

2- **Adding LTS Jenkins Repo and Installation**

sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc \

https://pkg.jenkins.io/debian-stable/jenkins.io-2026.key

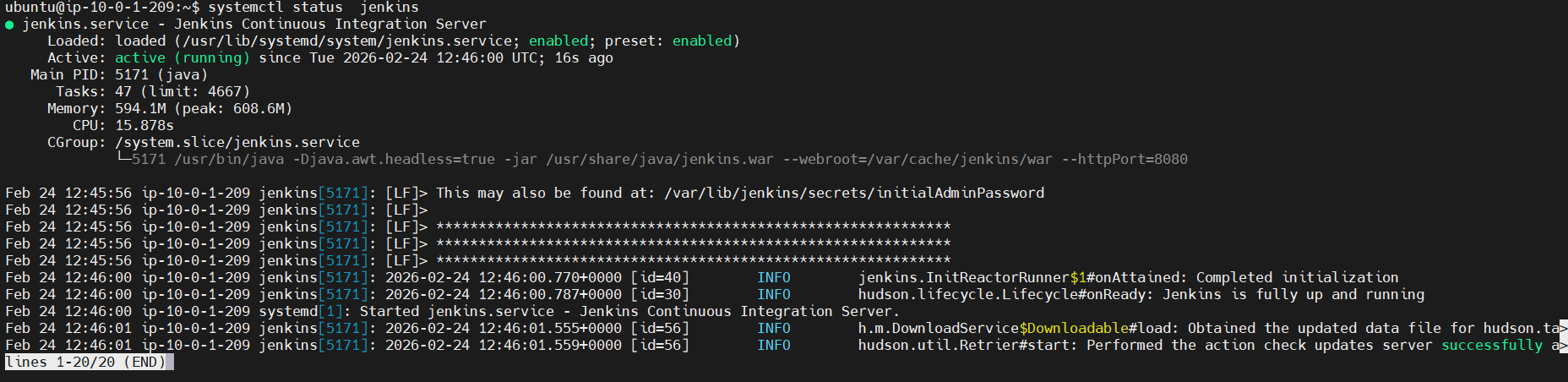
echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc]" \

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

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

sudo apt update

sudo apt install jenkins

**3- Starting and enabling the Service**

sudo systemctl enable --now jenkins

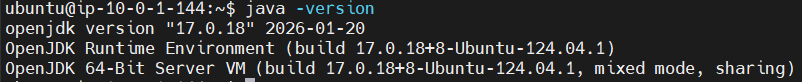
**On Agent node**

1. **Installation of Java**

sudo apt update

sudo apt install -y openjdk-17-jdk

java -version



1. **Installation of Docker**

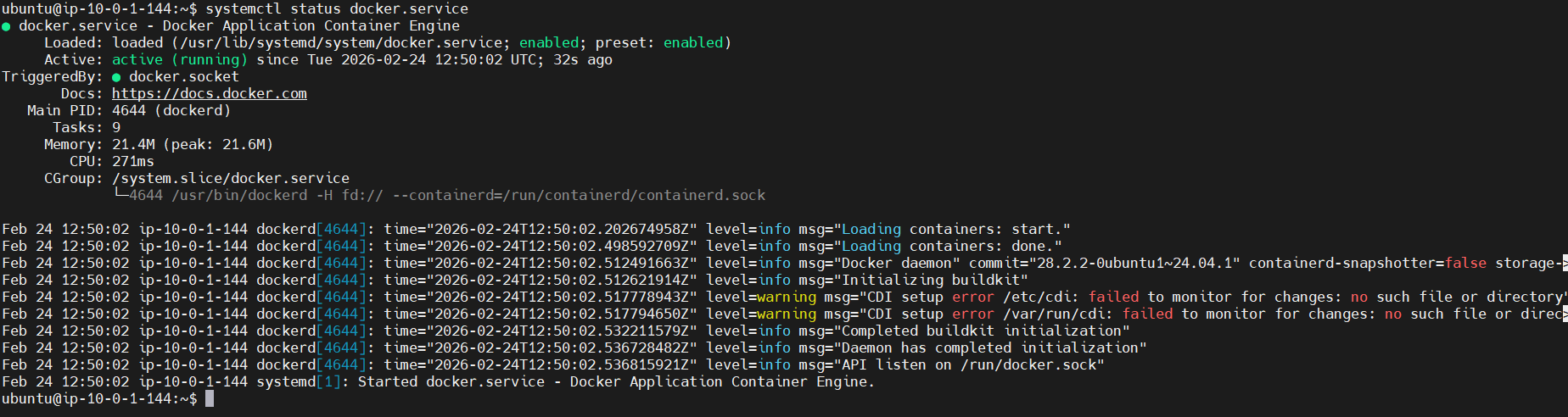
sudo apt install -y docker.io  
sudo systemctl enable docker  
sudo systemctl start docker

**Allow ubuntu user to run Docker:**

sudo usermod -aG docker ubuntu

**Apply group:**

newgrp docker



1. **Installation of Trivy**

sudo apt update  
sudo apt install -y wget gnupg lsb-release  
  
# Add Aqua Security repo key  
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | \  
sudo gpg --dearmor -o /usr/share/keyrings/trivy.gpg  
  
# Add repo  
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb $(lsb\_release -sc) main" | \  
sudo tee /etc/apt/sources.list.d/trivy.list  
  
# Install Trivy  
sudo apt update  
sudo apt install -y trivy

#Create Trivy template directory:

sudo mkdir -p /usr/share/trivy/templates

#Download official templates:

sudo wget -O /usr/share/trivy/templates/html.tpl \  
https://raw.githubusercontent.com/aquasecurity/trivy/main/contrib/html.tpl  
  
sudo wget -O /usr/share/trivy/templates/junit.tpl \  
https://raw.githubusercontent.com/aquasecurity/trivy/main/contrib/junit.tpl

sudo chmod -R 755 /usr/share/trivy



Enrollment of Slave inside the master

1. **Create SSH Key**

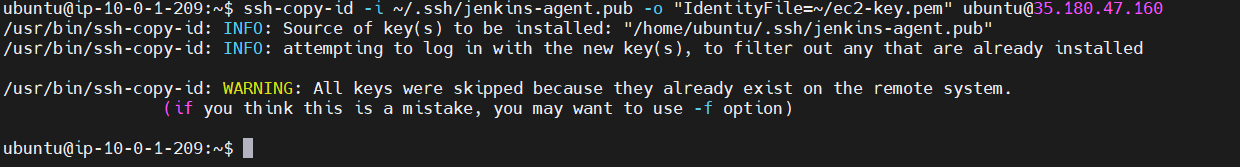
*Happens on: MASTER*

ssh-keygen -t rsa -b 4096 -f ~/.ssh/jenkins-agent

1. **Copy Public Key to slave**

*happens on: MASTER*

ssh-copy-id -i ~/.ssh/jenkins-agent.pub -o "IdentityFile=~/ec2-key.pem" ubuntu@ 35.180.47.160

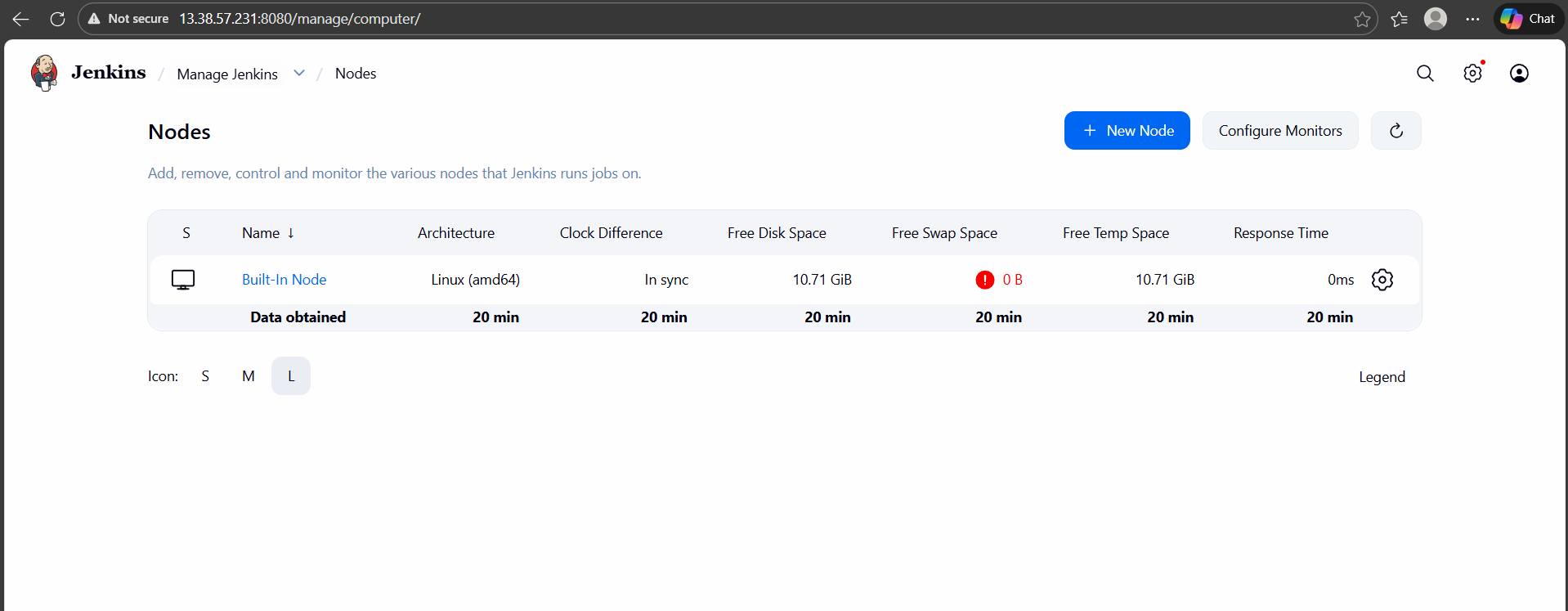
**

1. **Create Jenkins working dir**

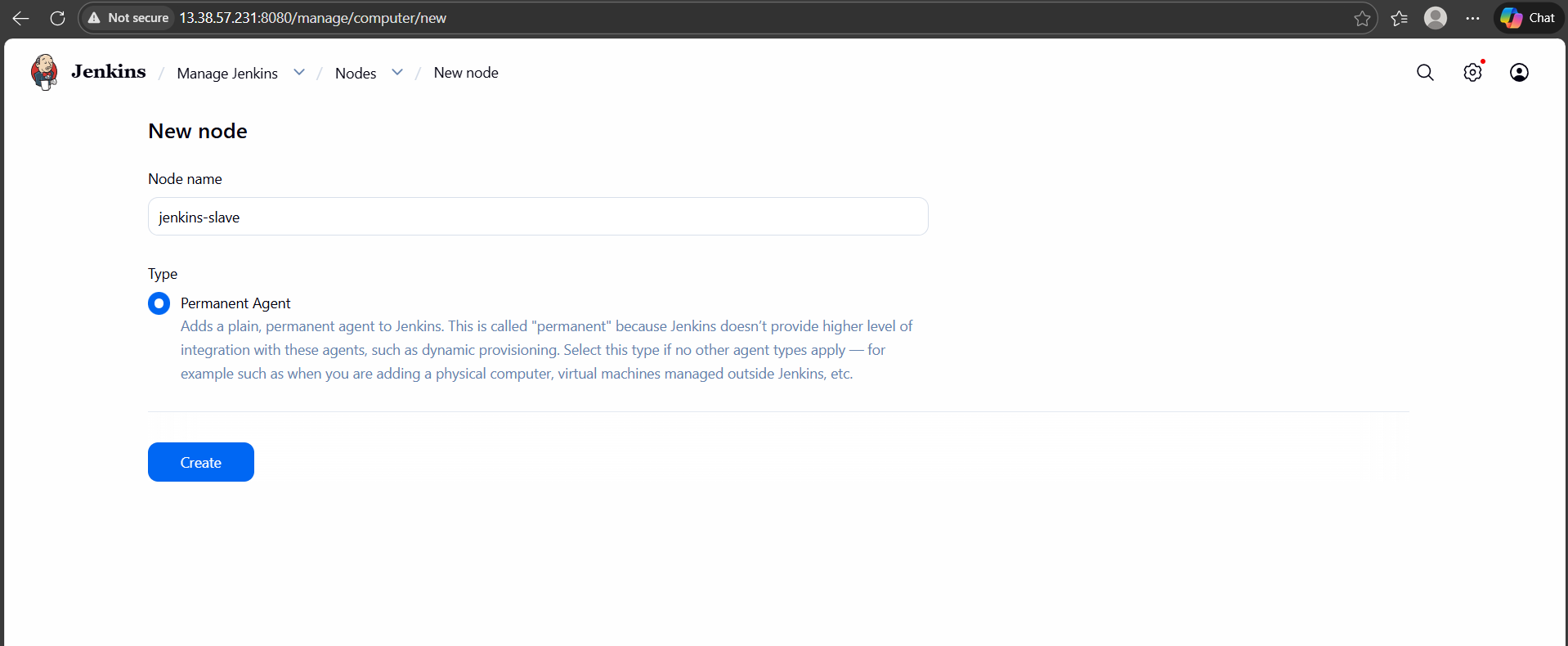
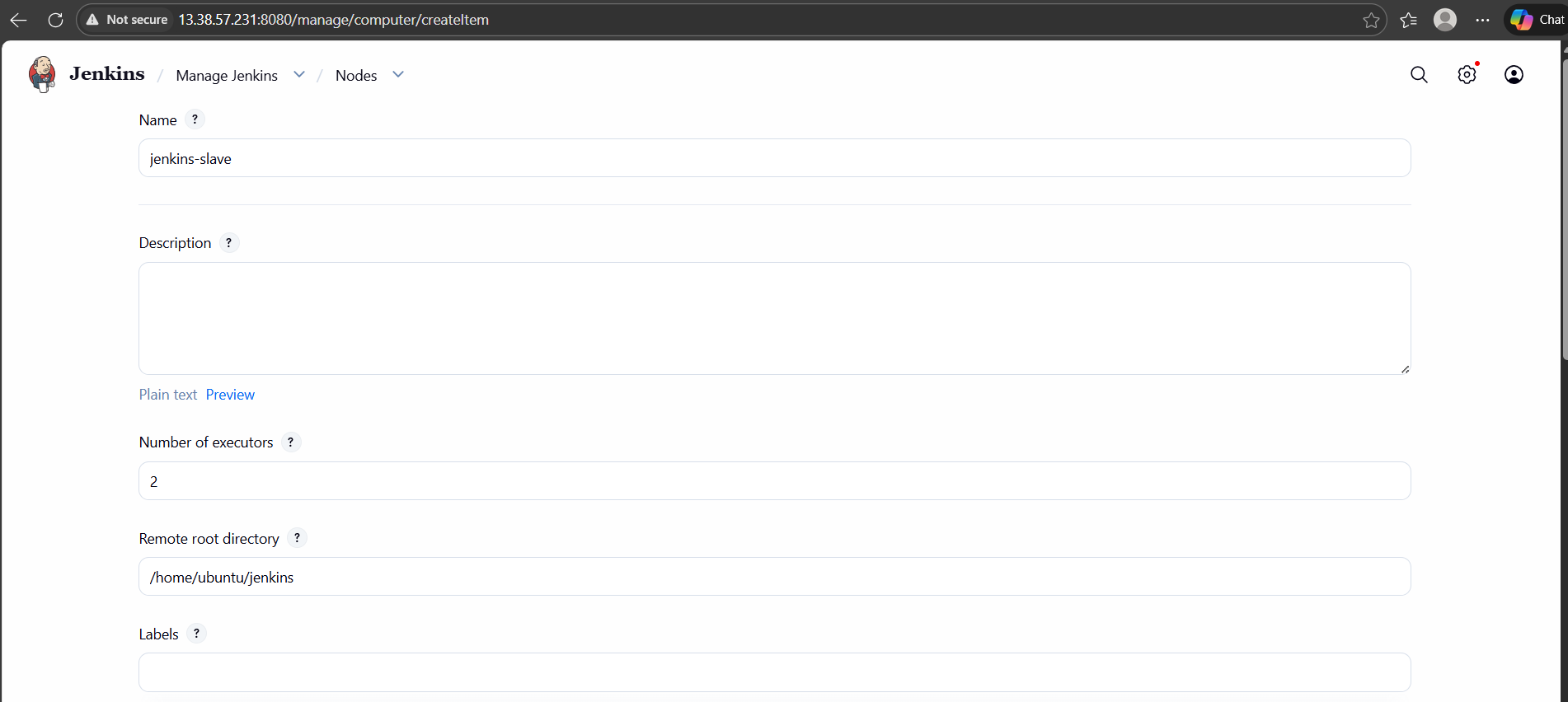
*Happens on: Slave*

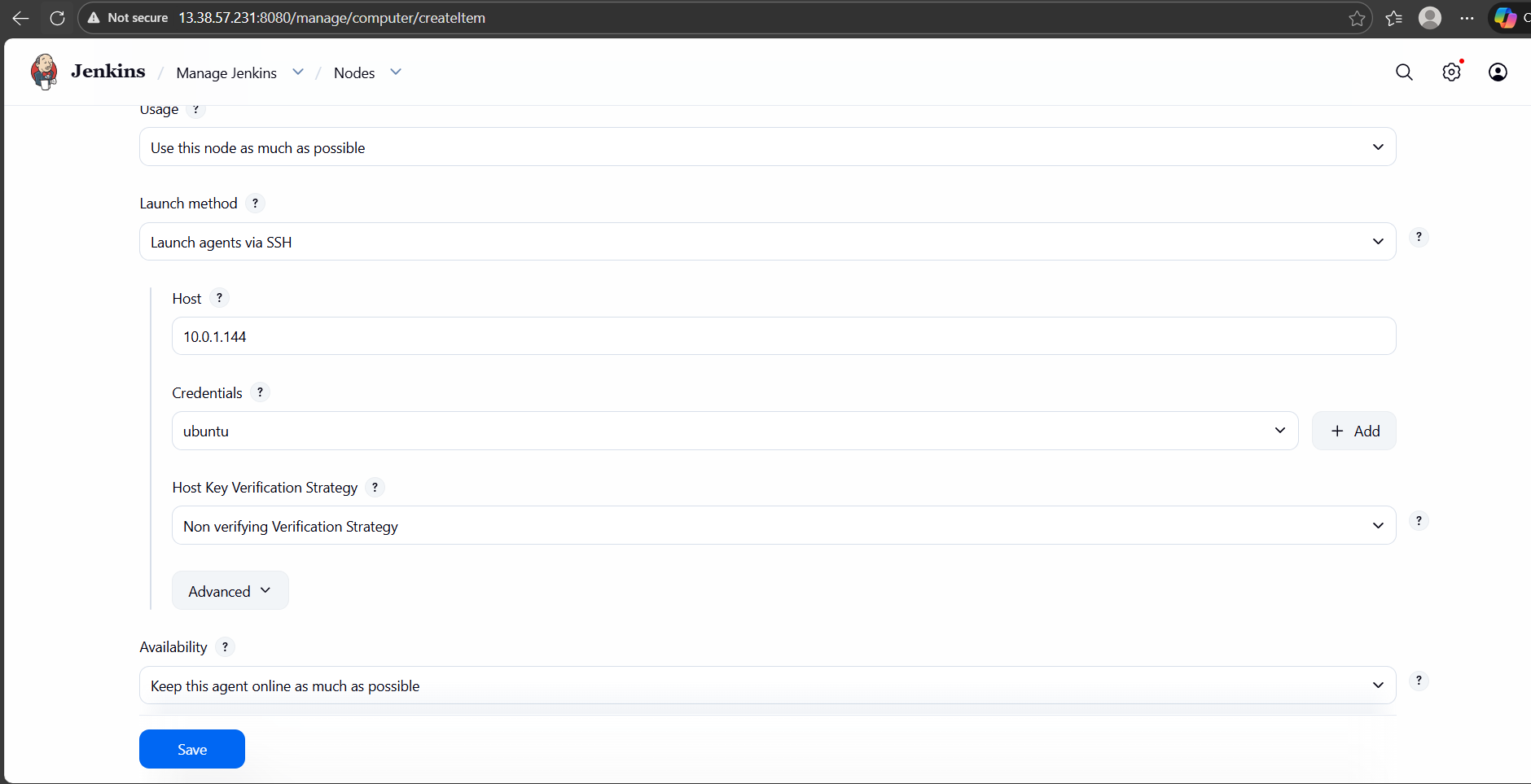
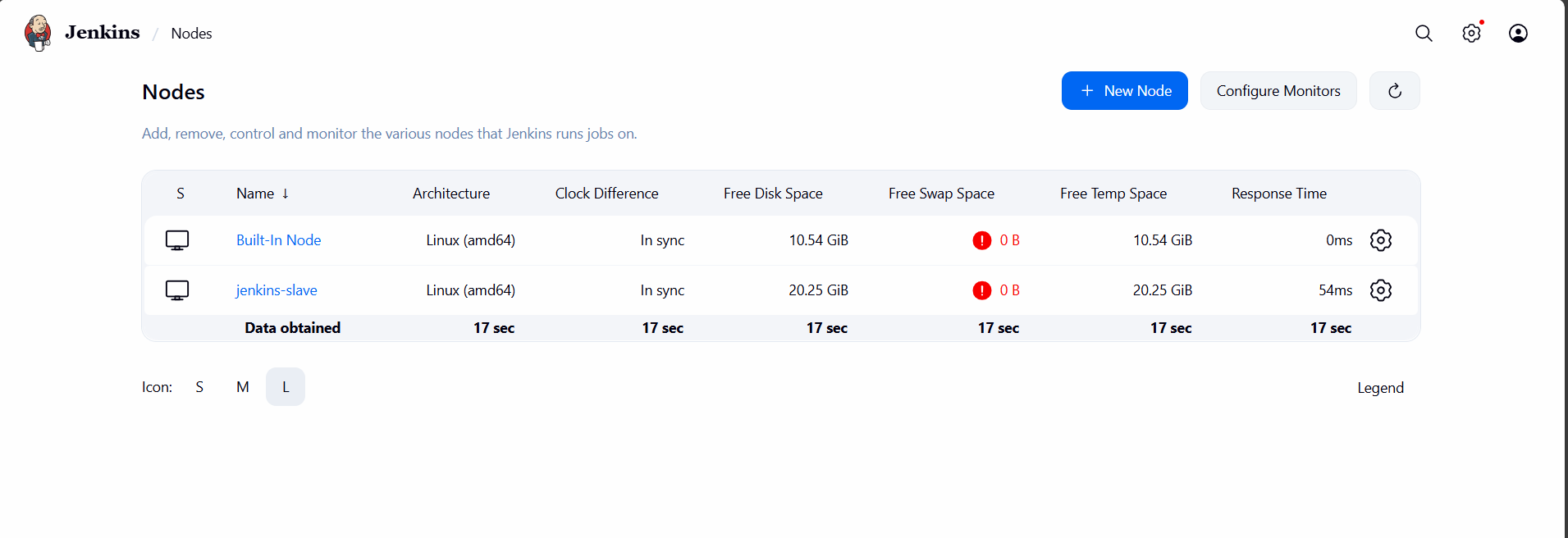
mkdir -p /home/ubuntu/jenkins

1. **Register Jenkins Agent (SSH Node)**

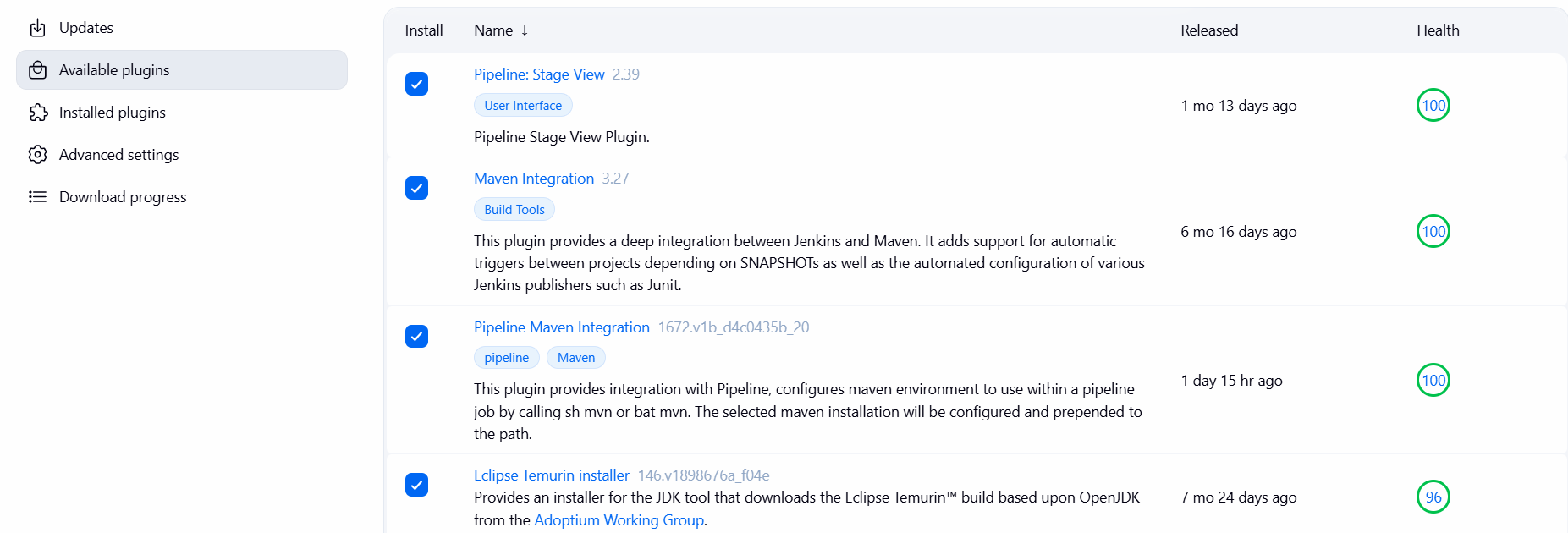
*happens on: MASTER*

ssh-copy-id -i ~/.ssh/jenkins-agent.pub ubuntu@AGENT\_IP

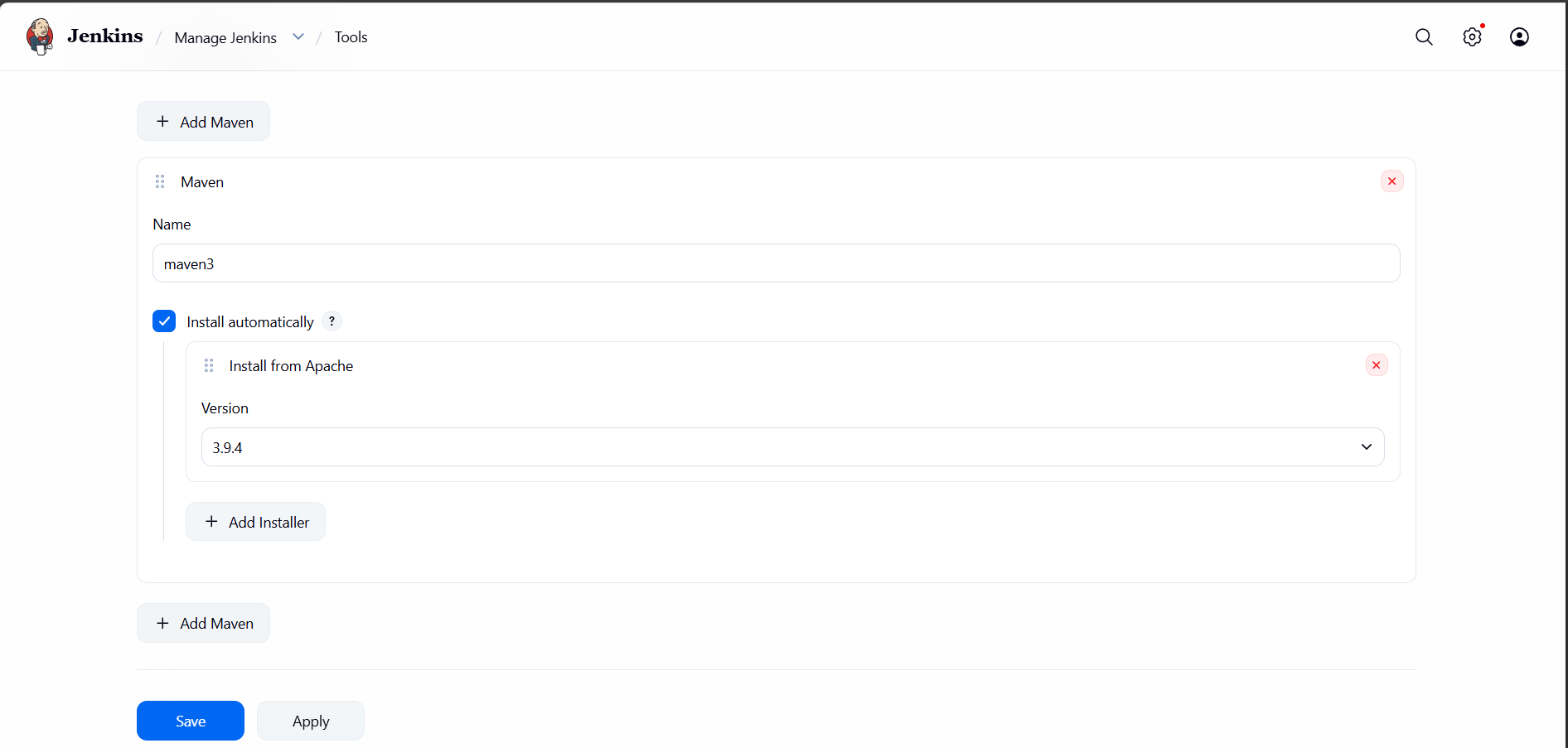
**

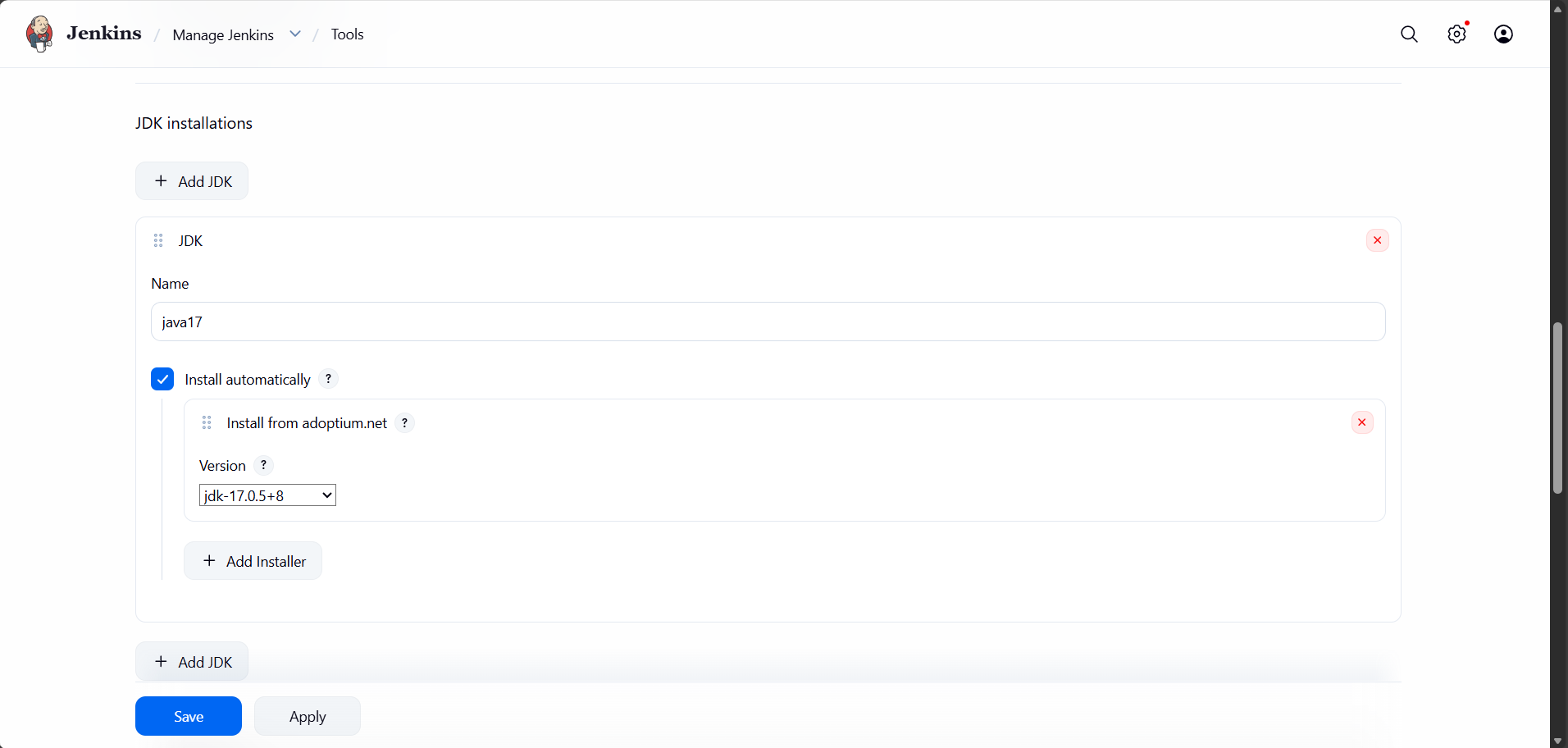


Plugins Installed



Plugins Configurations:





Setting up sonarqube



1. **Installation of Docker**

sudo apt update  
sudo apt install -y docker.io  
sudo systemctl enable docker  
sudo systemctl start docker

**Allow ubuntu user to run Docker:**

sudo usermod -aG docker ubuntu

**Apply group:**

newgrp docker

1. **Creating SonarQube Container**

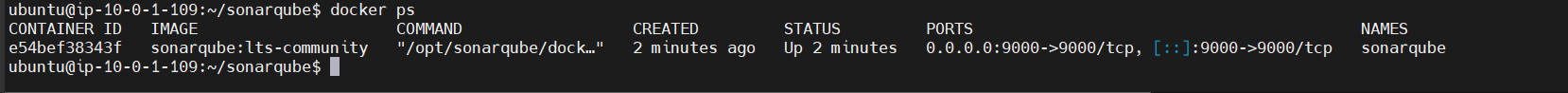
#Create SonarQube Directories

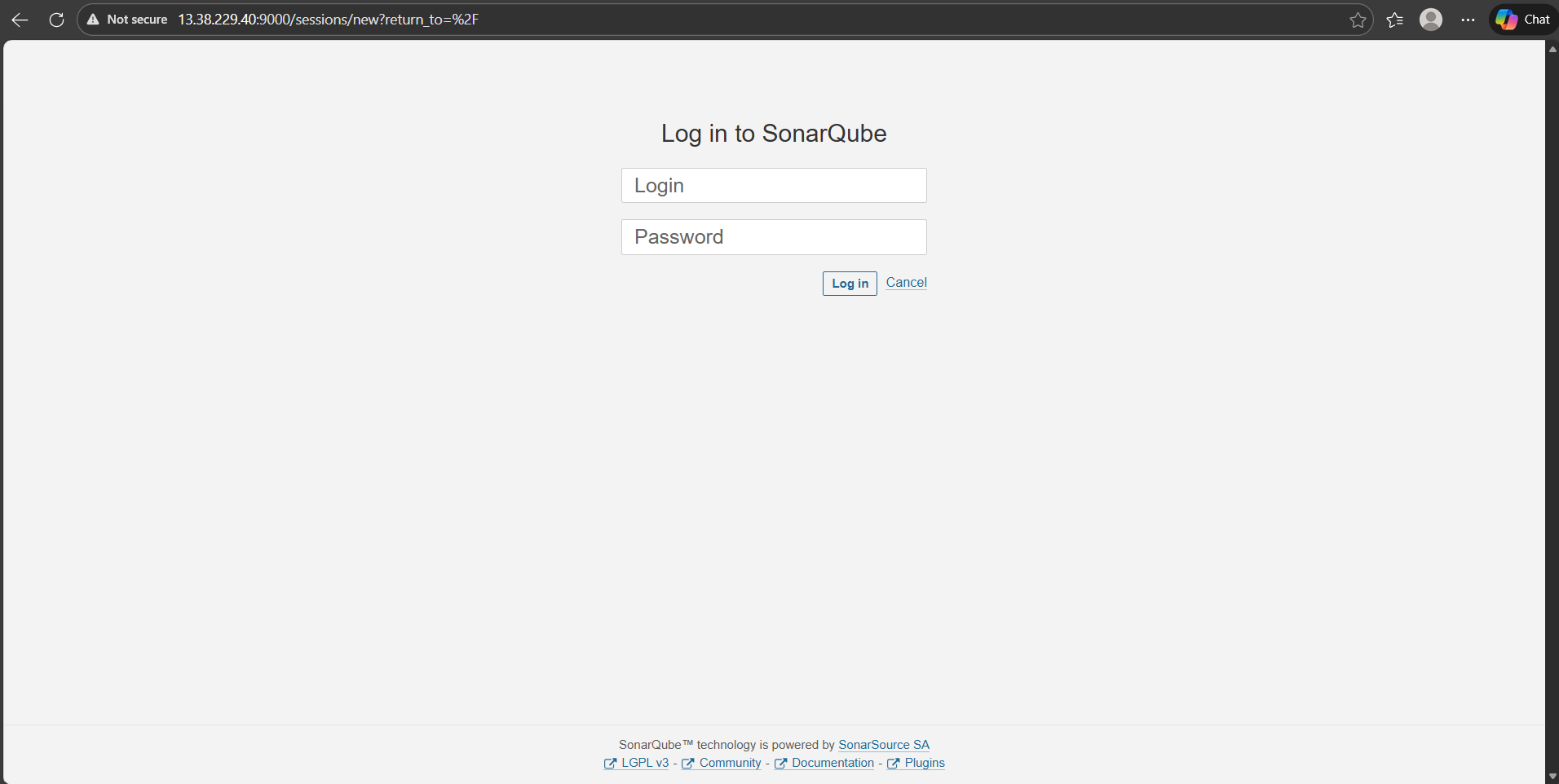
mkdir -p ~/sonarqube/{data,logs,extensions}

cd ~/sonarqube

#Run SonarQube Container

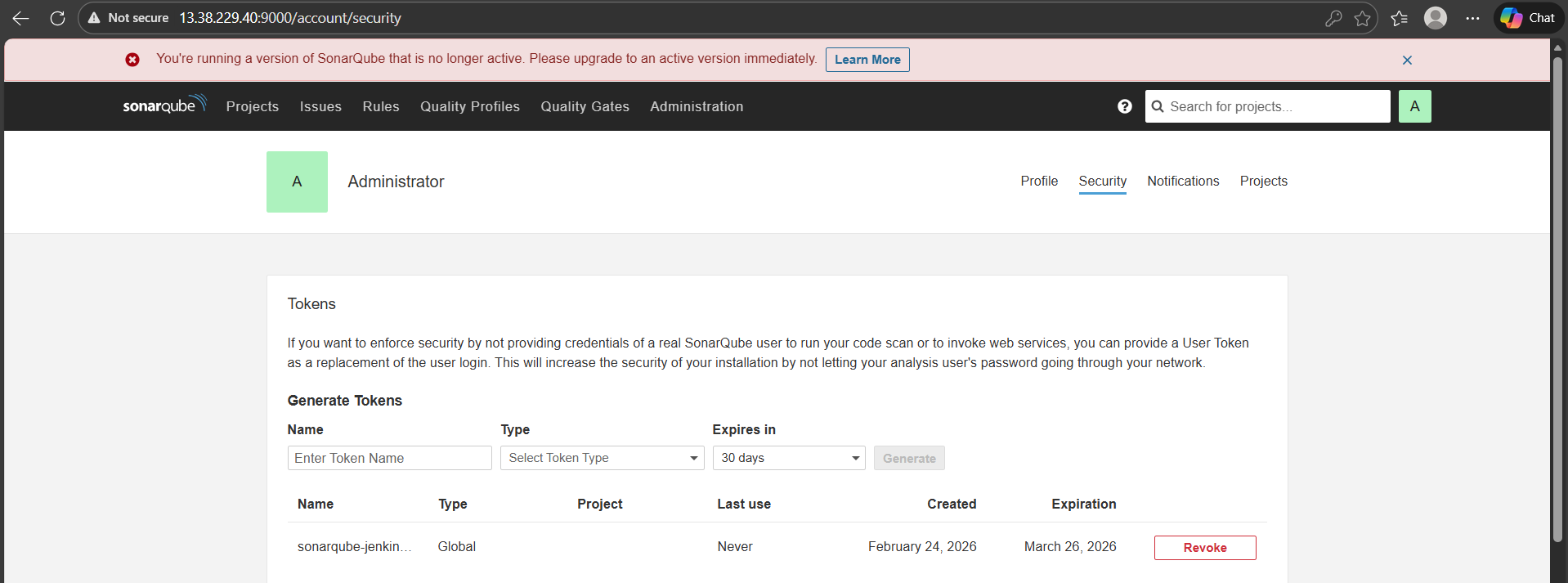
docker run -d \  
--name sonarqube \  
-p 9000:9000 \  
-v $(pwd)/data:/opt/sonarqube/data \  
-v $(pwd)/logs:/opt/sonarqube/logs \  
-v $(pwd)/extensions:/opt/sonarqube/extensions \  
sonarqube:lts-community



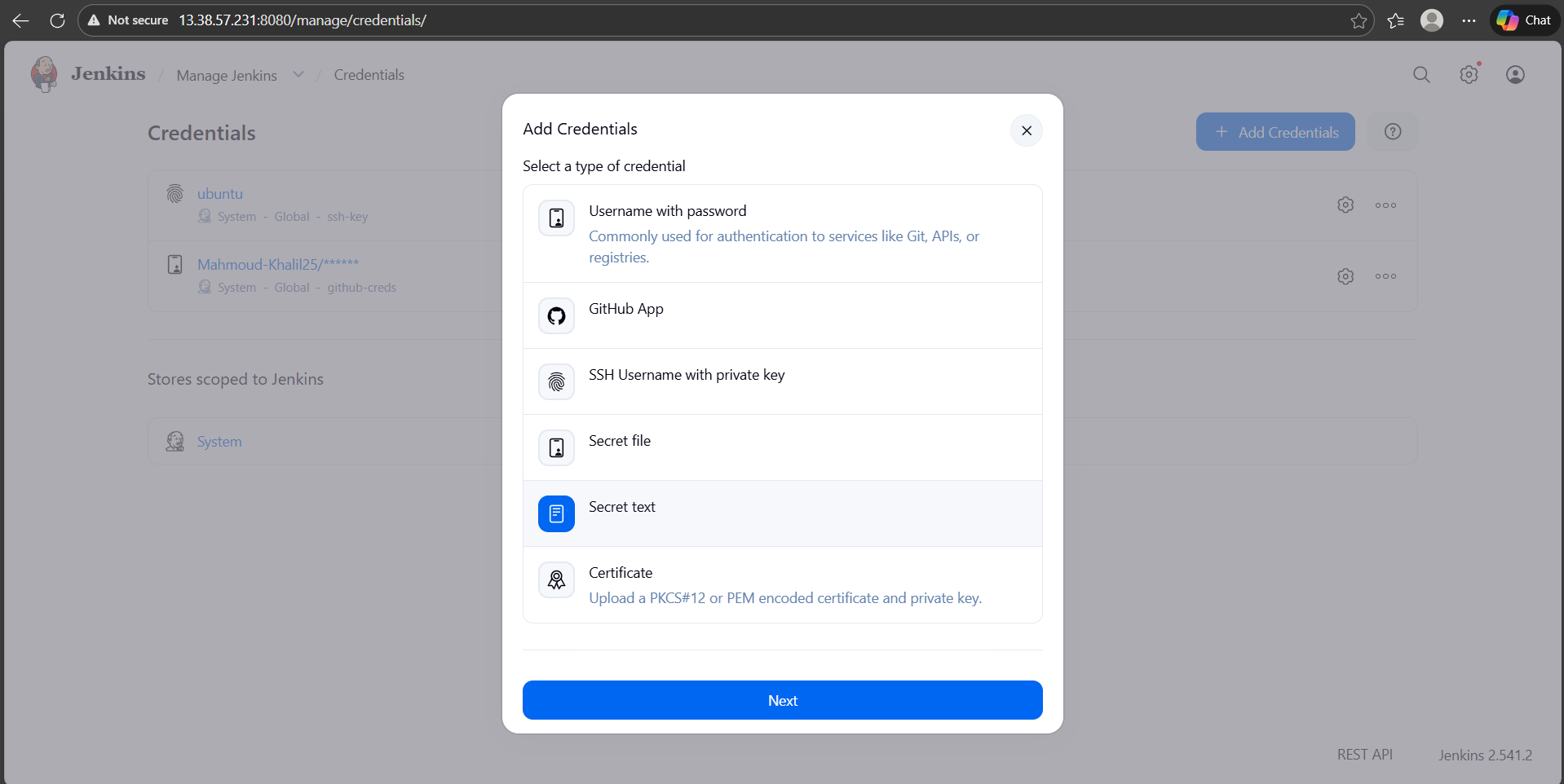


**Integrating Sonarqube with Jenkins**

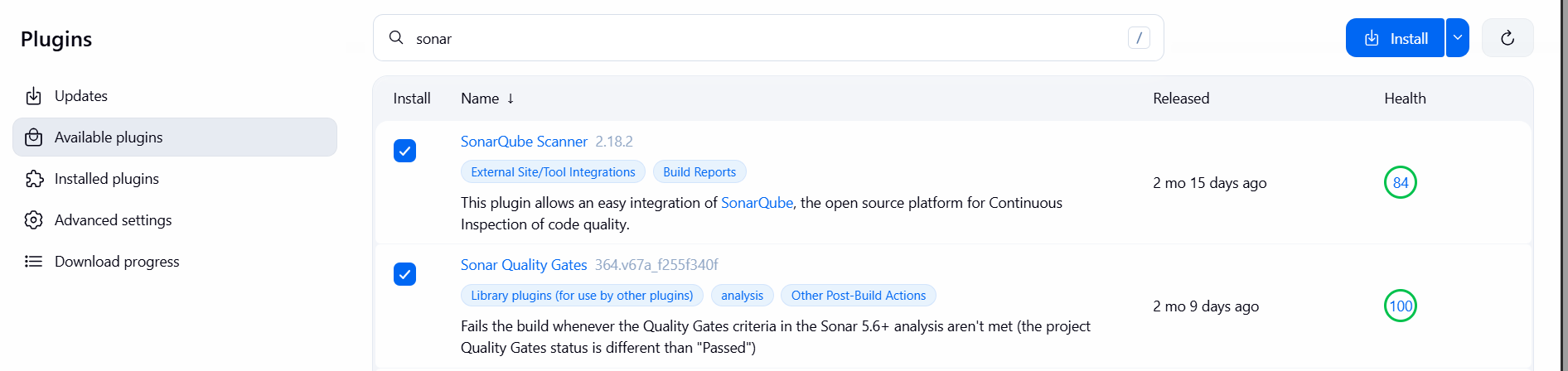
1. **Generate a token for Jenkins to use in analysis**

****

1. **Create credential of the newly generated token inside Jenkins**

****

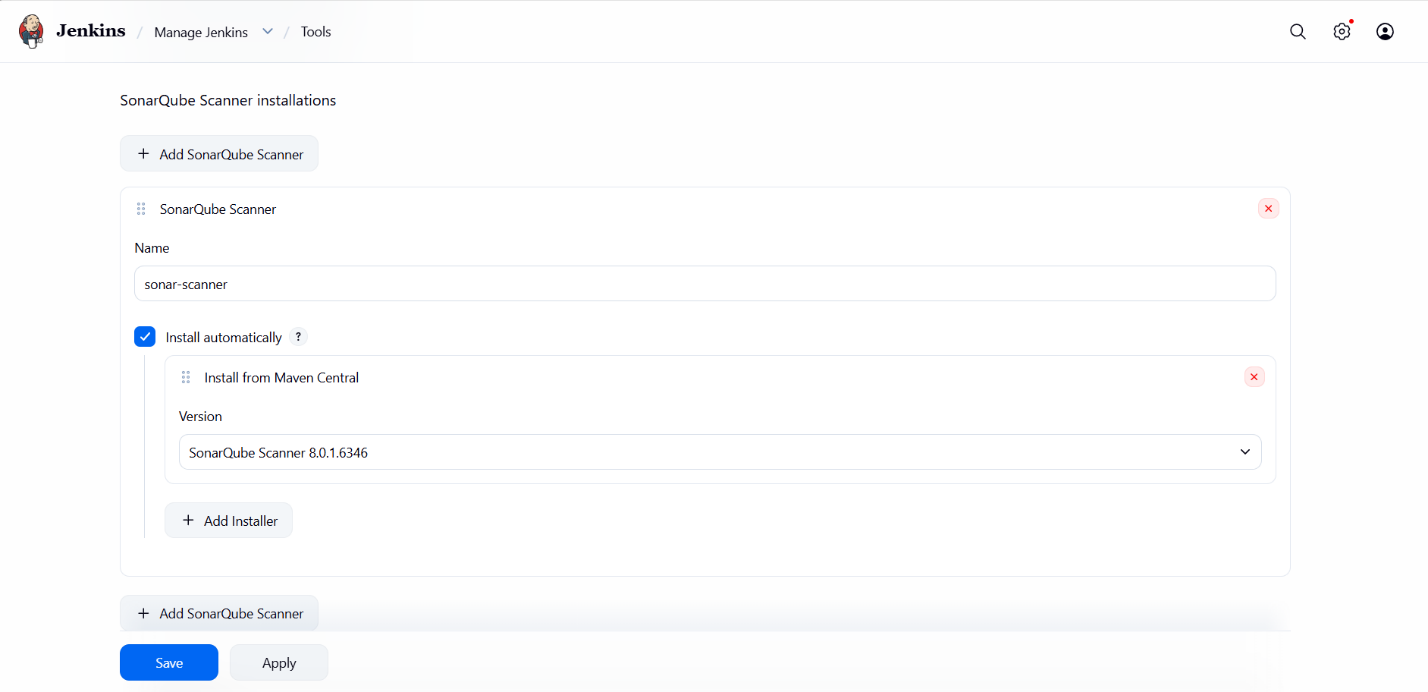
1. **Install the following plugins on jenkins**

****

1. **Configure sonarqube server on Jenkins**

****

1. **Configure sonarqube scanner on Jenkins**

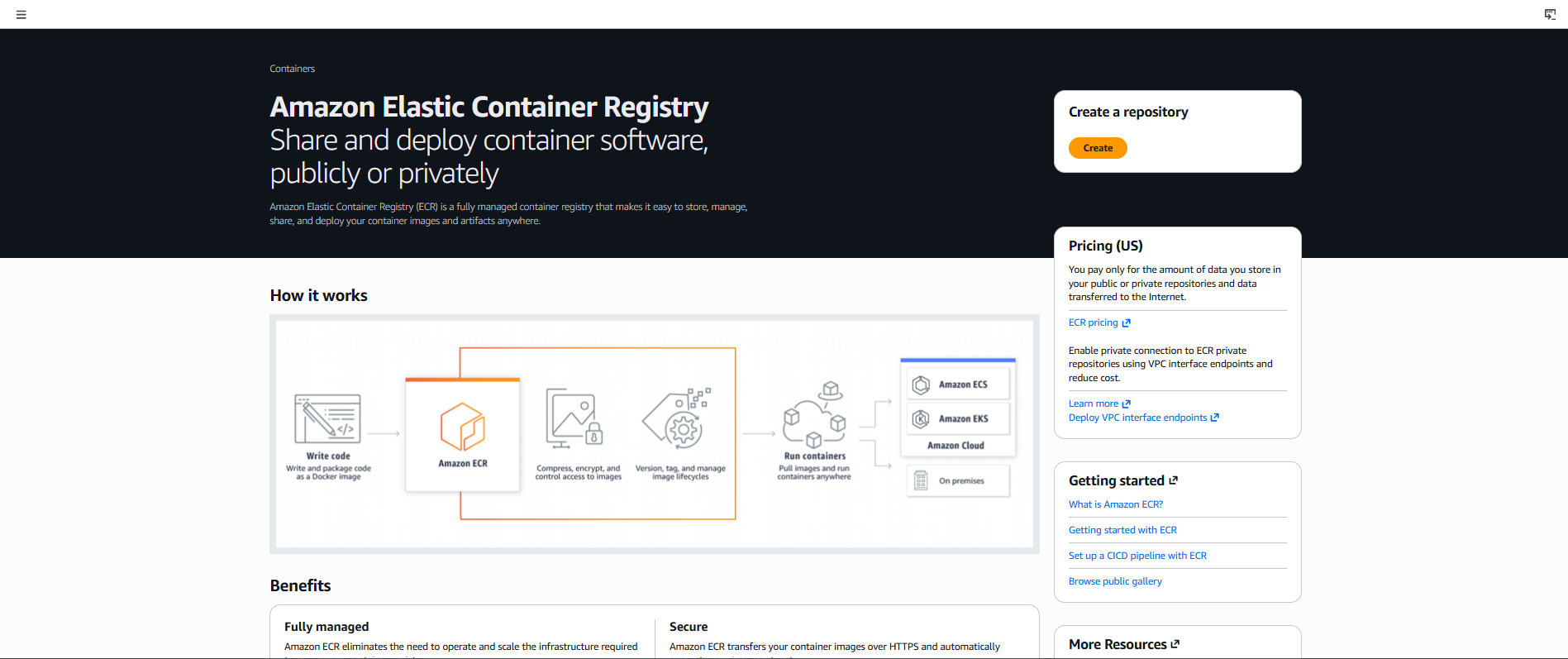


1. **Configure webhook on sonarqube**

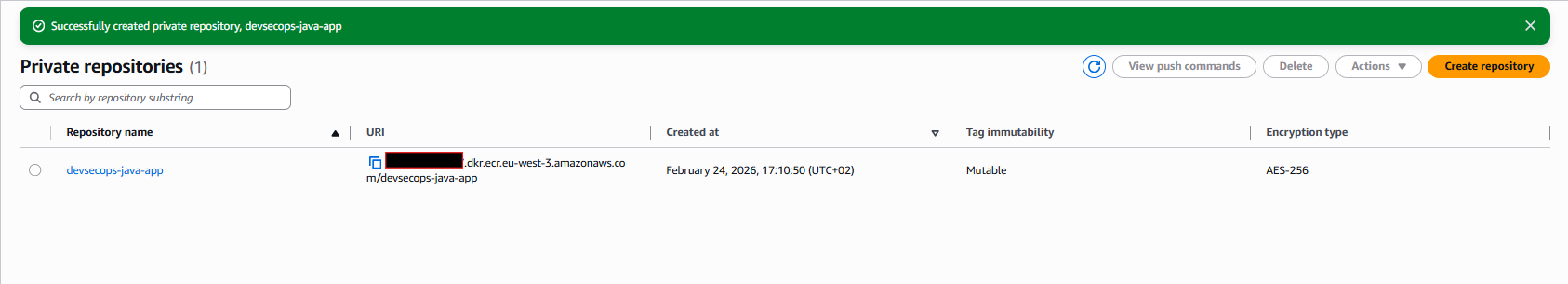
****

**Create ECR Repository to push built image to**

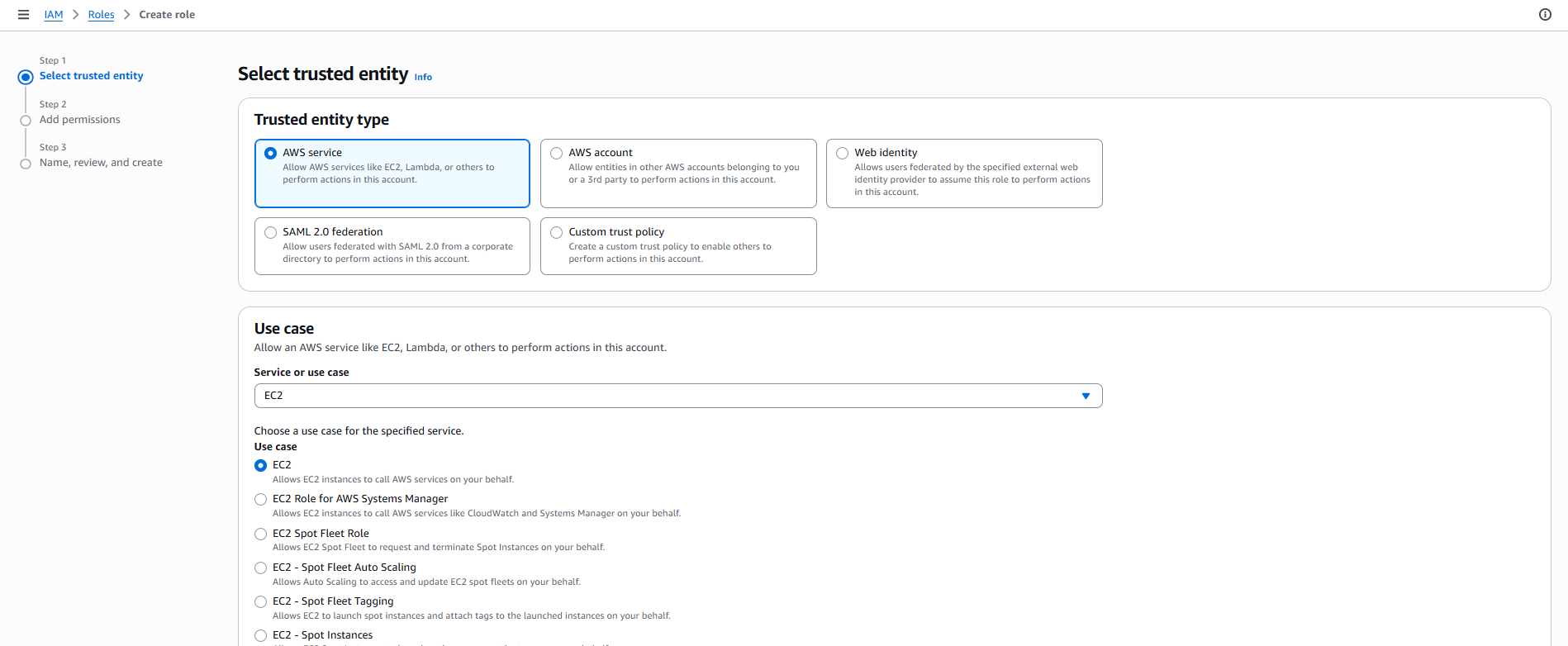
1. **Create Private Repo to push docker images**

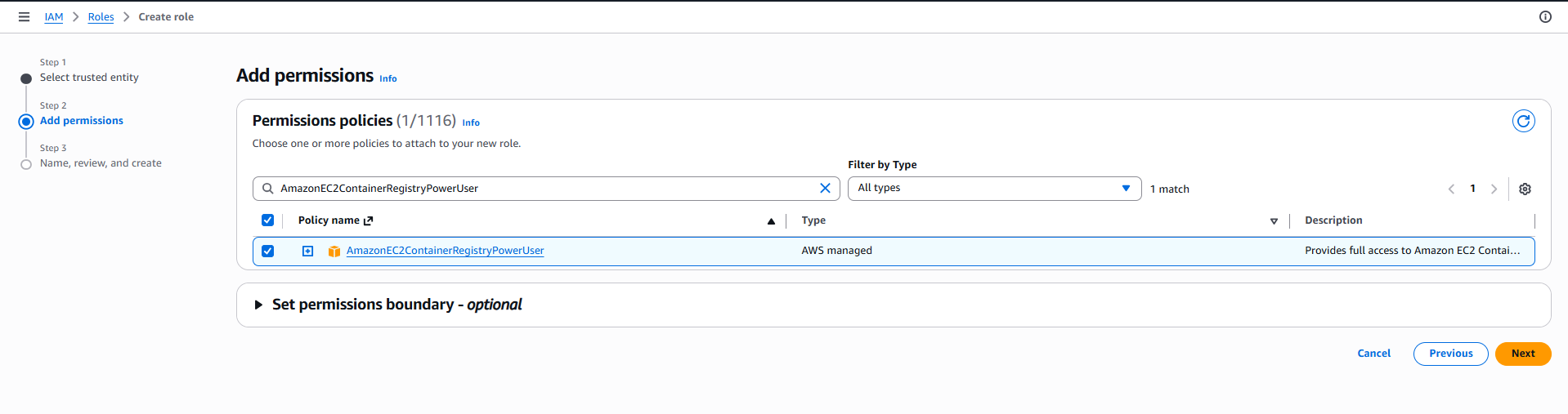
****

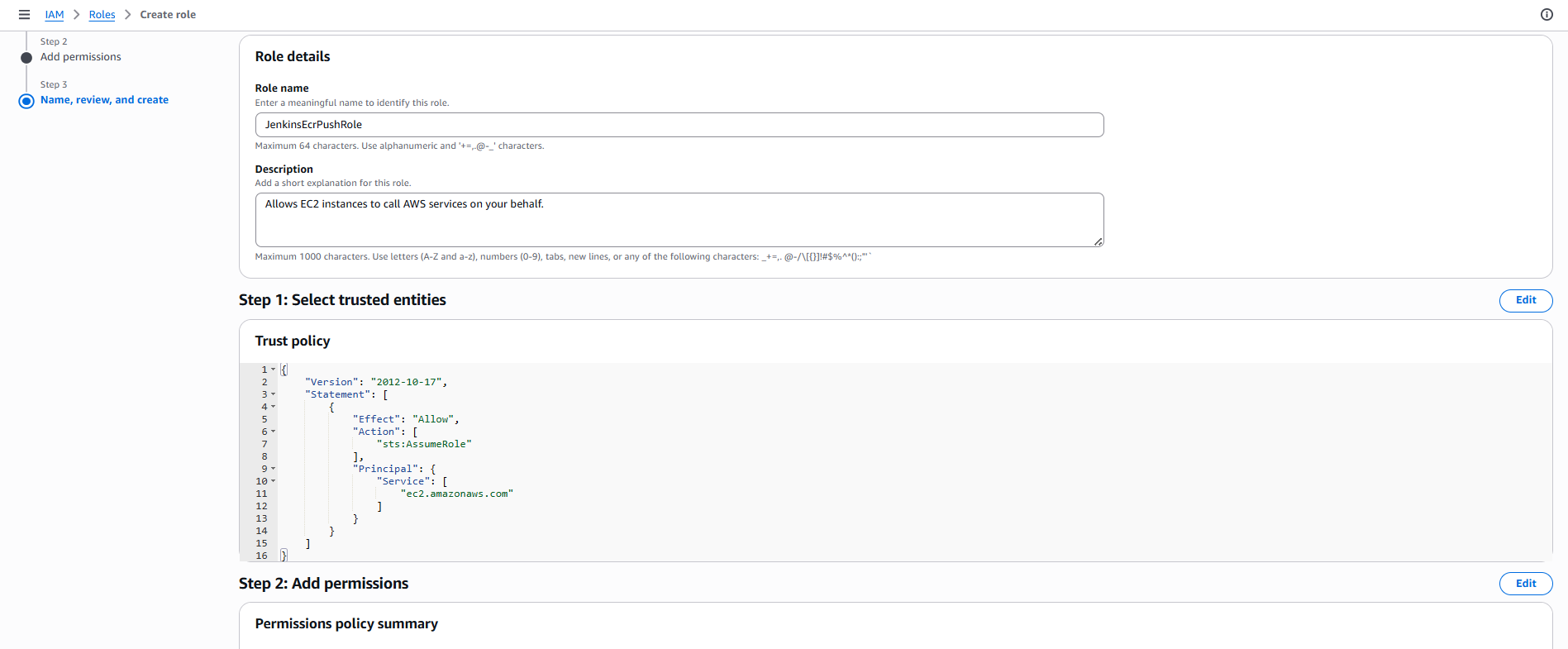
****

****

1. **Create IAM Role to allow EC2 instance to access ECR**

****

****

****

1. **Attach Role on EC2 Instance**

****

Steps on Bootstarp server

1. **Install AWS CLI v2**

sudo apt update  
sudo apt install -y unzip curl  
  
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"  
unzip awscliv2.zip  
sudo ./aws/install

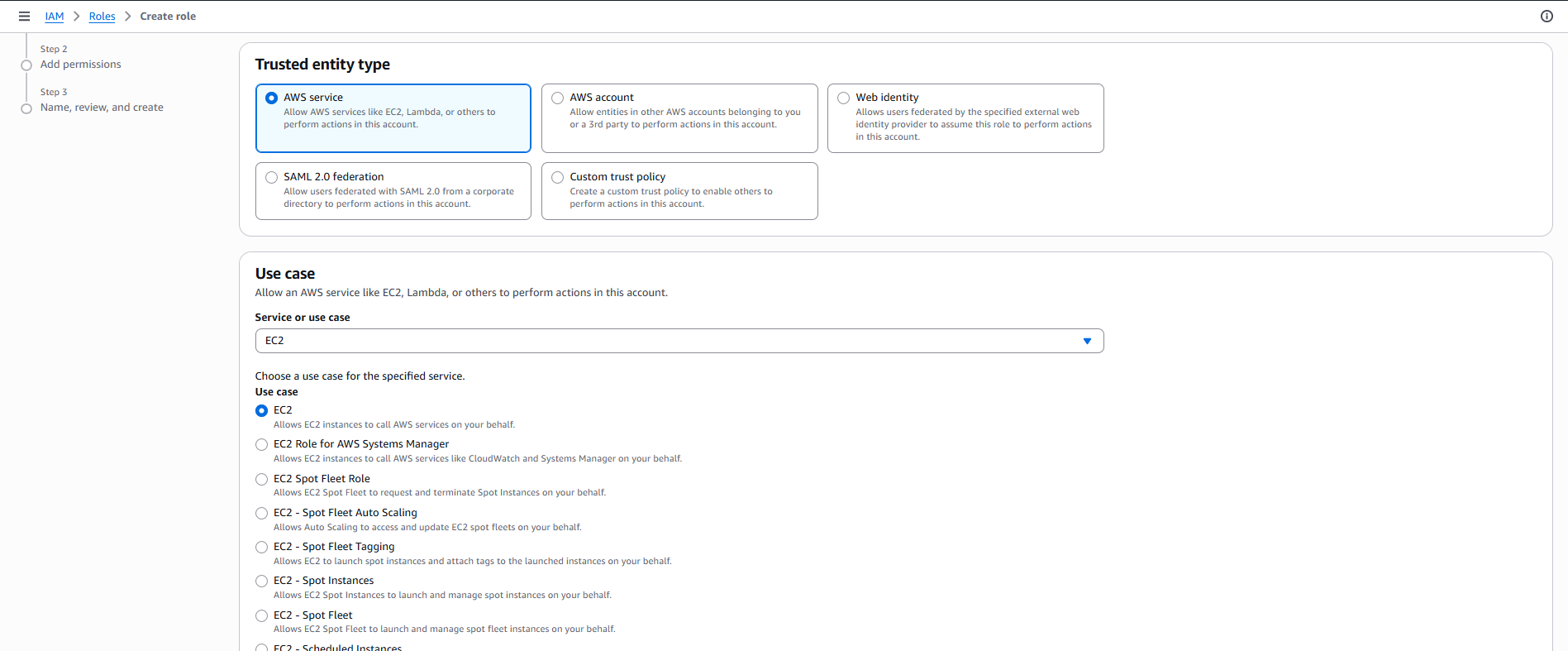
1. **Install Kubectl**

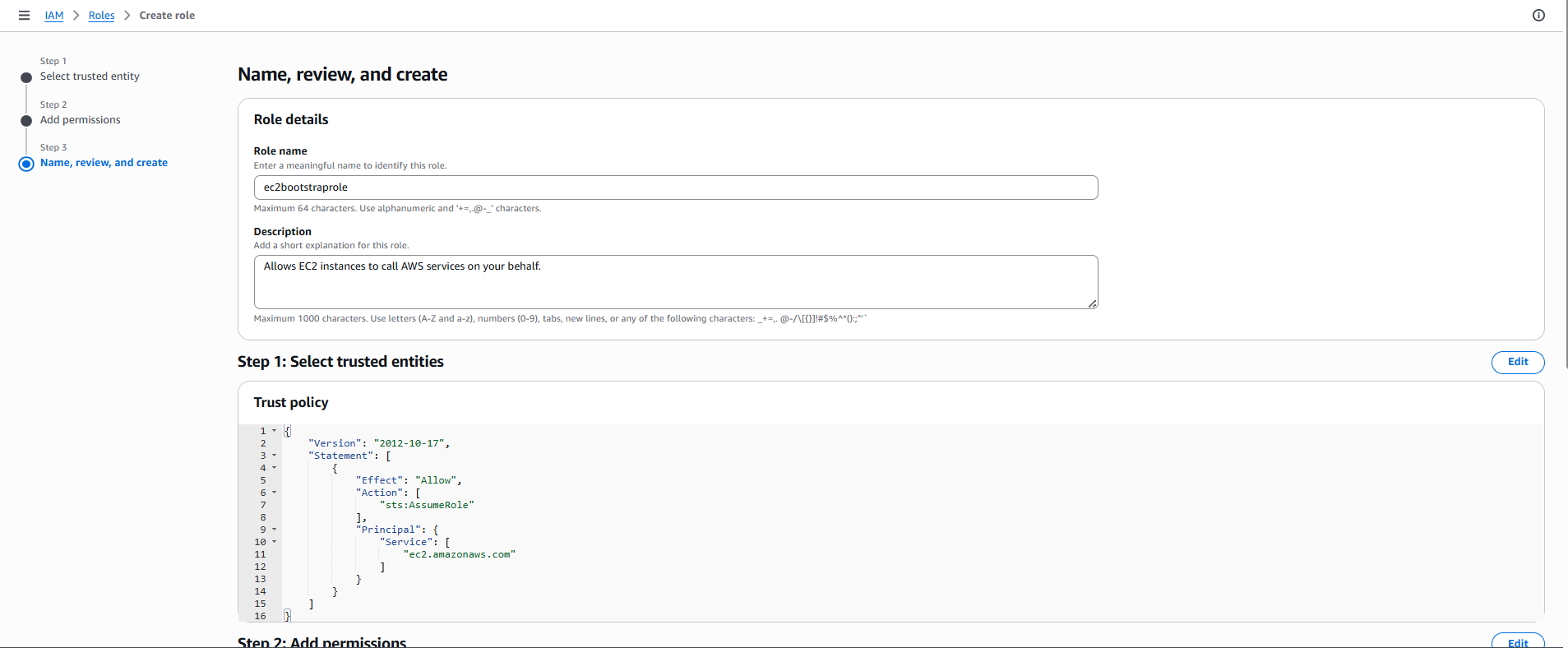
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"  
  
chmod +x kubectl  
sudo mv kubectl /usr/local/bin/

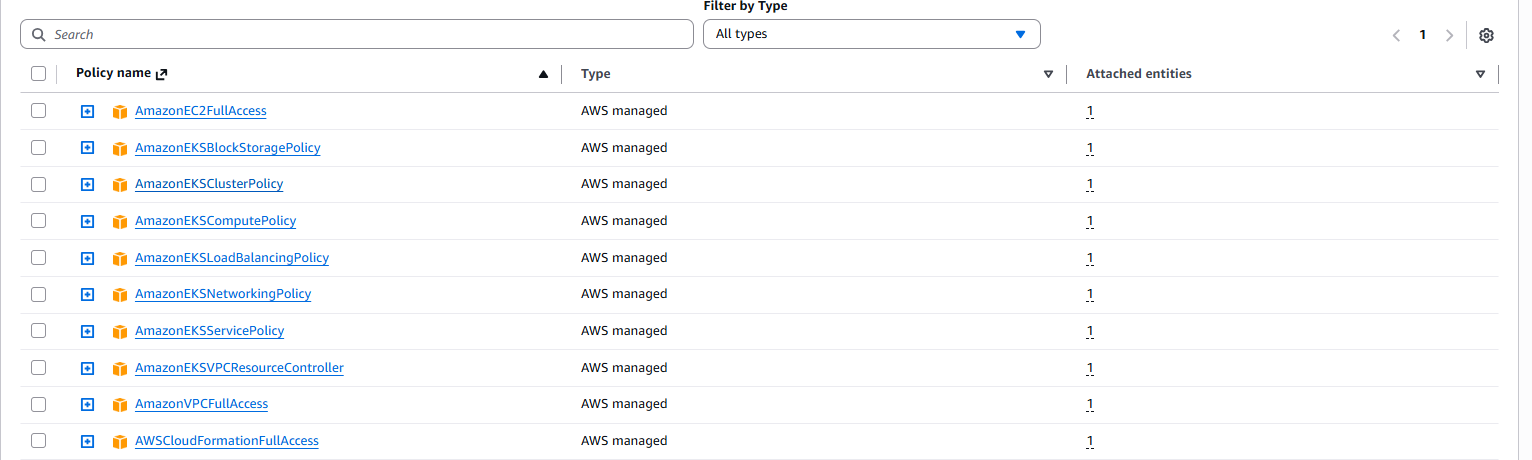
1. **Install eksctl**

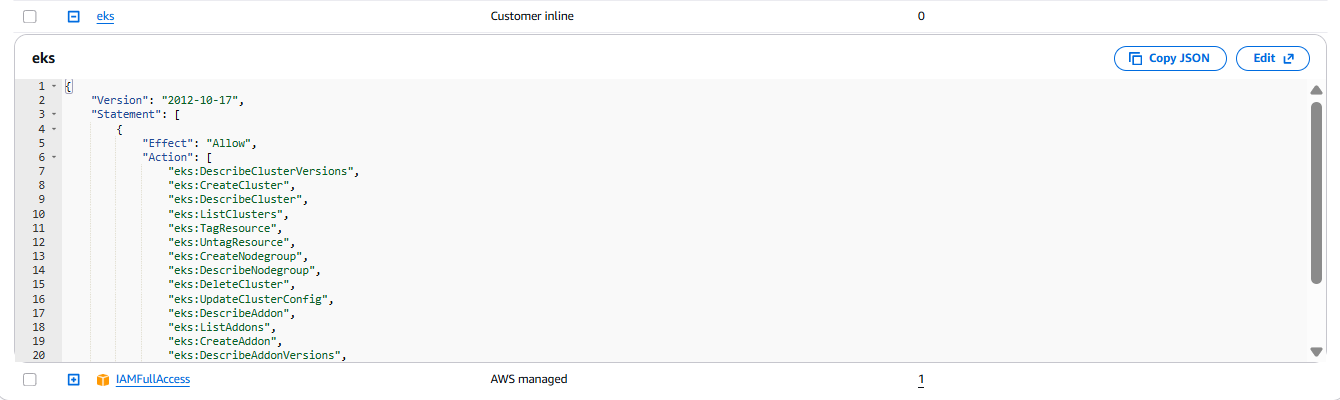
curl -sLO "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl\_Linux\_amd64.tar.gz"  
  
tar -xzf eksctl\_Linux\_amd64.tar.gz  
sudo mv eksctl /usr/local/bin/

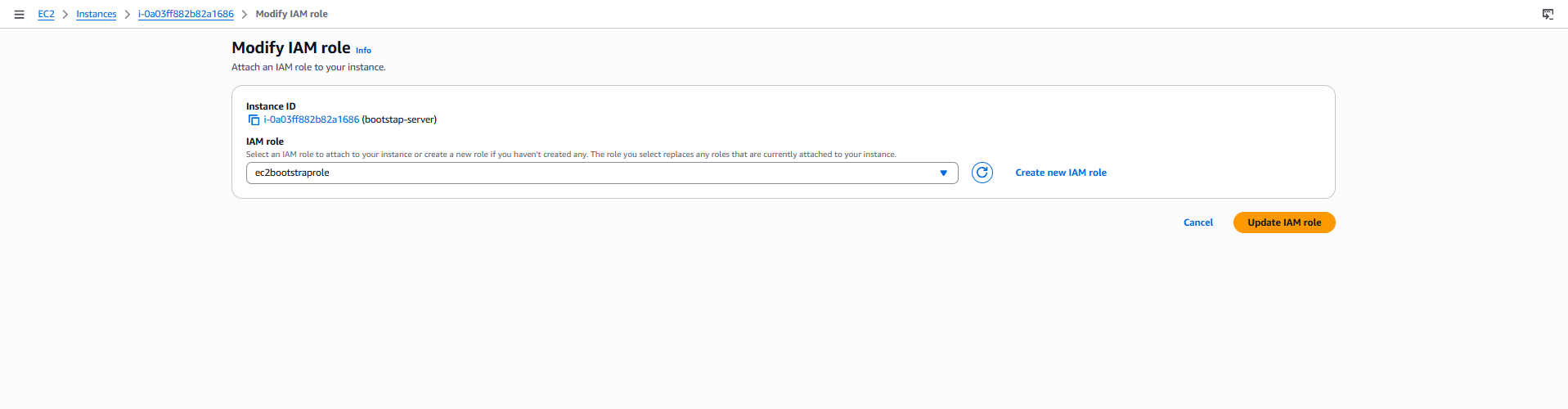
1. **Create and Attach IAM Role to Bootstrap EC2 to create EKS cluster**

****

****

****

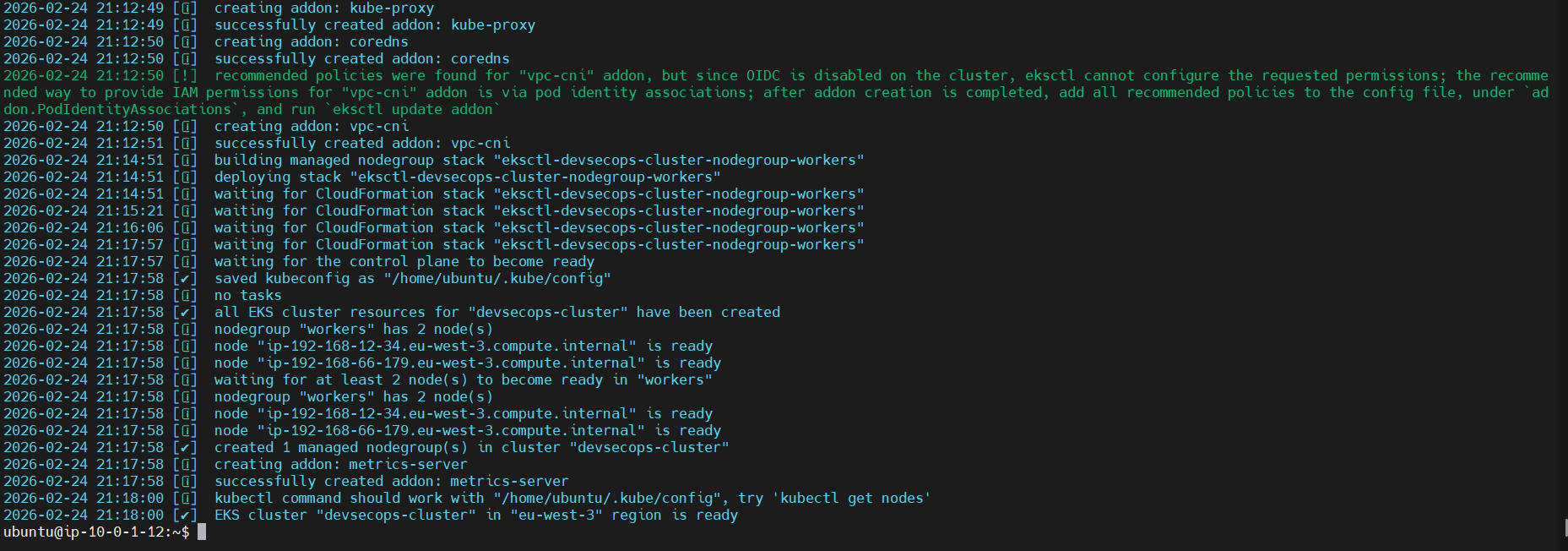
****

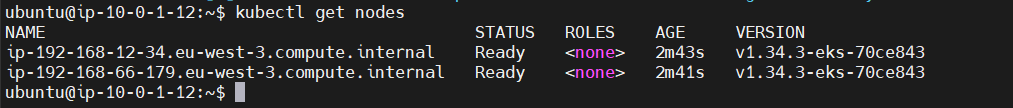
****

1. **Create EKS cluster**

eksctl create cluster \  
--name devsecops-cluster \  
--region eu-west-3 \  
--nodegroup-name workers \  
--node-type t3a.medium \  
--nodes 2 \  
--nodes-min 2 \  
--nodes-max 2 \  
--node-volume-size 30 \  
--managed

****

****

****

1. **Install Helm**

curl -fsSL https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash  
helm version

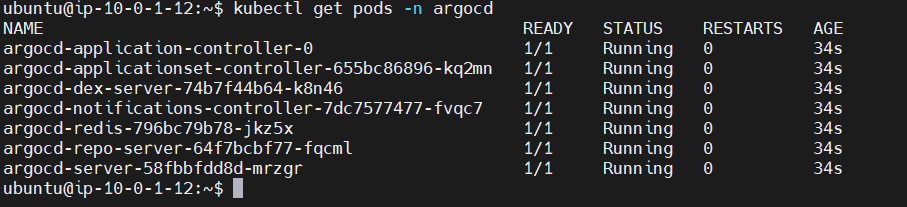
**Installation and configuration of Argocd**

1. **Create Argo CD namespace**

**kubectl create namespace argocd**

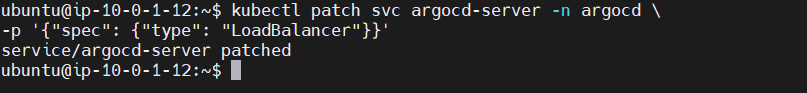
1. **Install Argo CD manifests**

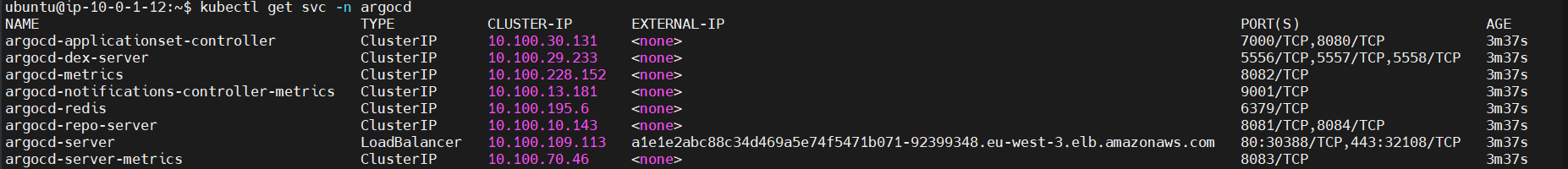
kubectl create -n argocd \  
-f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

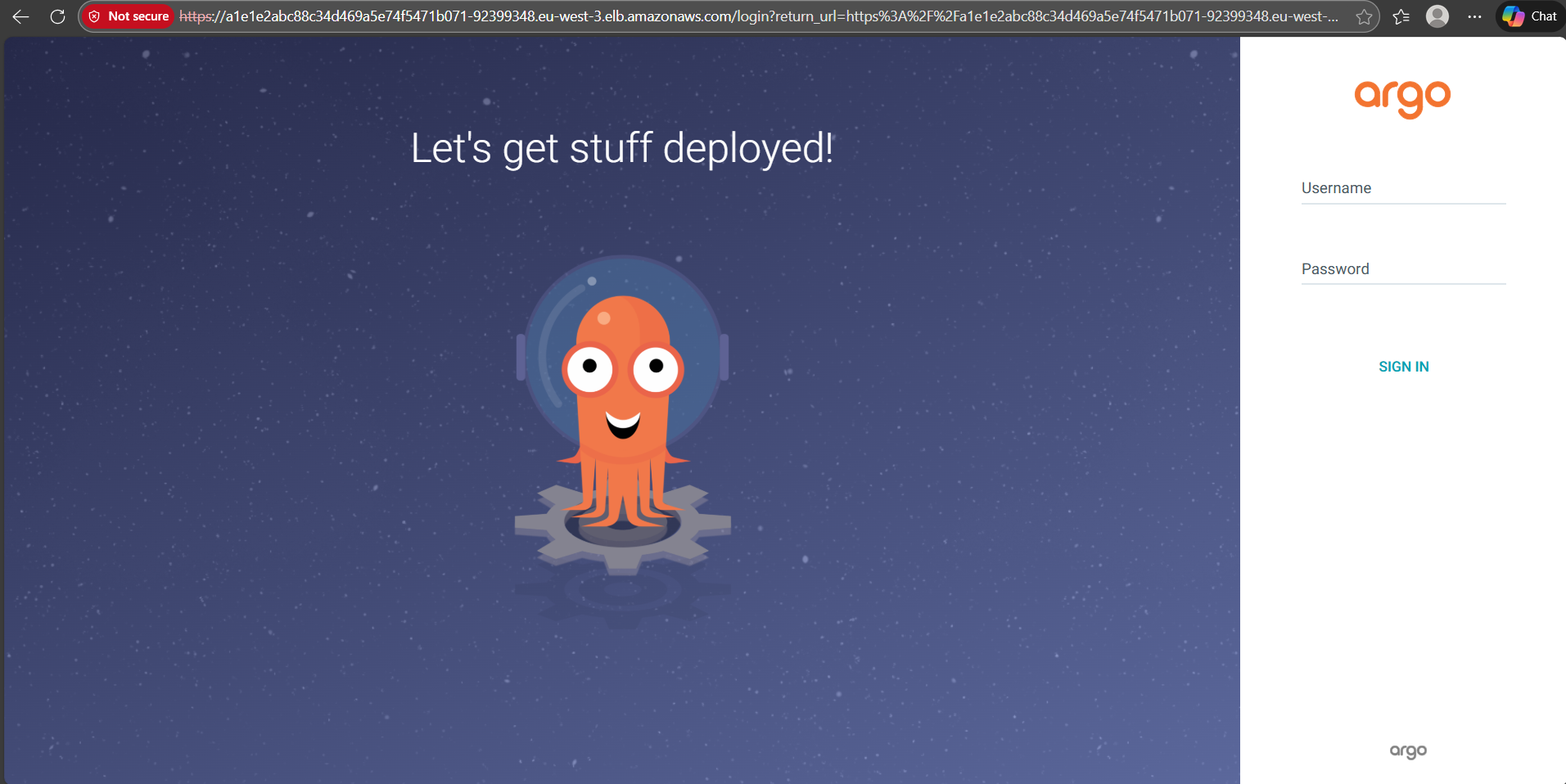
****

1. **Expose Argo CD Externally (LoadBalancer)**

**kubectl patch svc argocd-server -n argocd \  
-p '{"spec": {"type": "LoadBalancer"}}'**

****

****

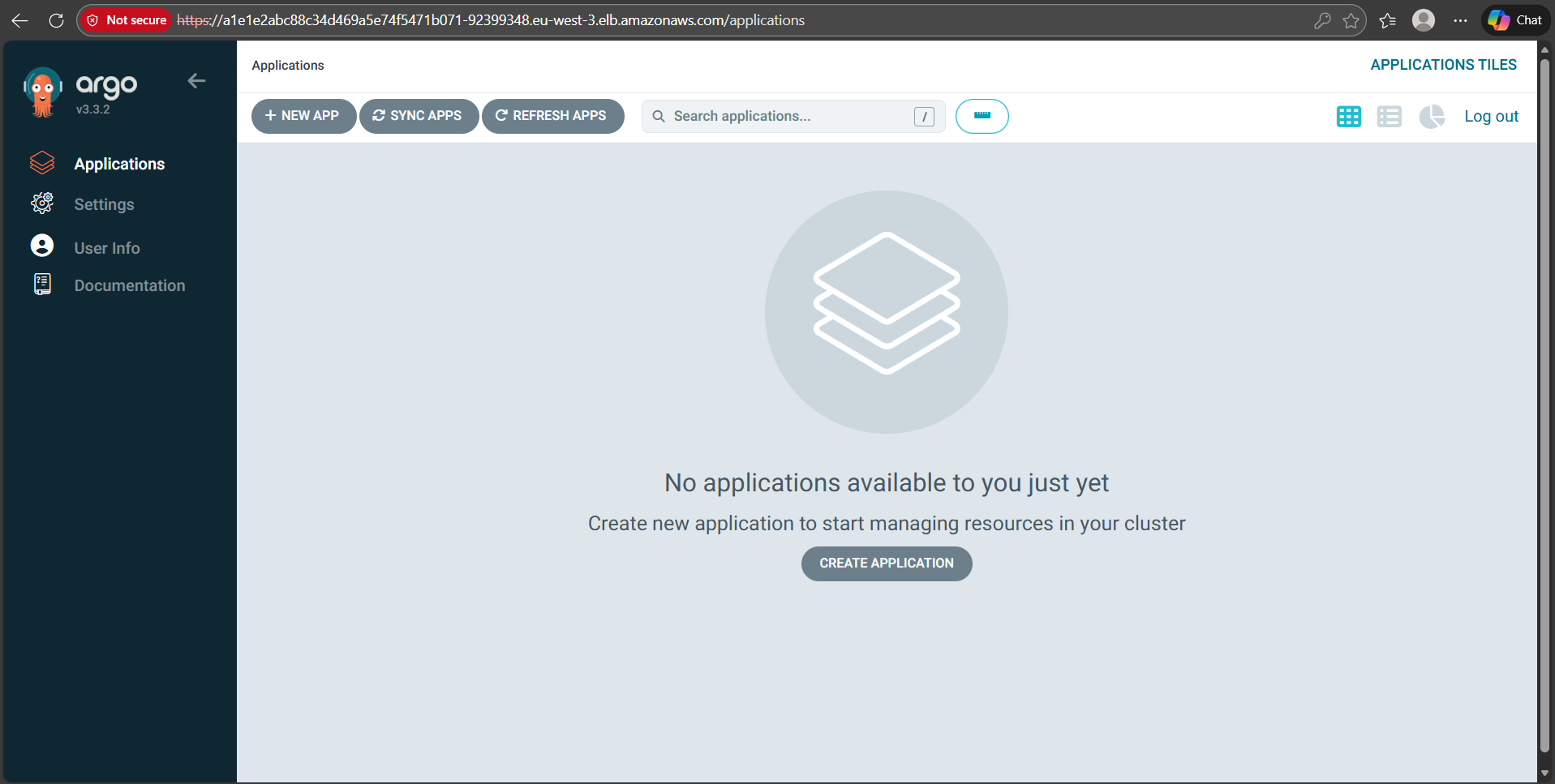
****

1. **Install Argo CD CLI**

curl -sSL -o argocd \  
https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64  
  
chmod +x argocd  
sudo mv argocd /usr/local/bin/

1. **Retrieve and decode initial admin password**

curl -sSL -o argocd \  
https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64  
  
chmod +x argocd  
sudo mv argocd /usr/local/bin/

****

1. **Add EKS cluster to Argo CD**

**Login to Argo CD via CLI**

**get password:**

ARGOCD\_PASS=$(kubectl -n argocd get secret argocd-initial-admin-secret \  
-o jsonpath="{.data.password}" | base64 -d)

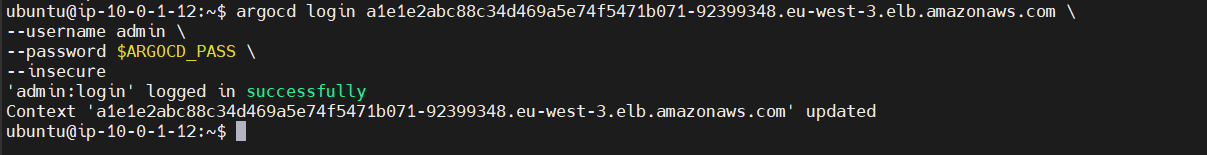
**login**

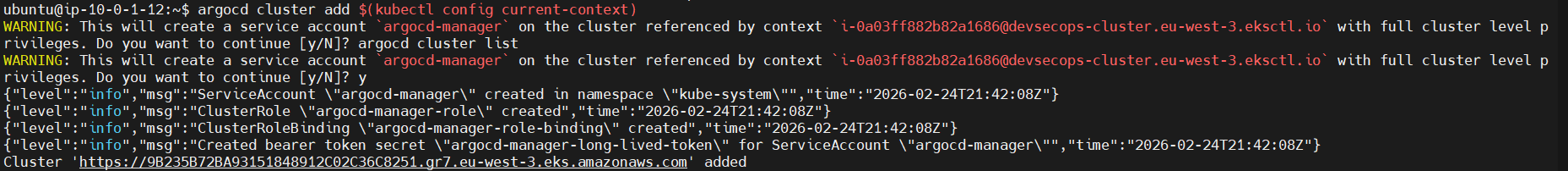
argocd login a1e1e2abc88c34d469a5e74f5471b071-92399348.eu-west-3.elb.amazonaws.com \  
--username admin \  
--password $ARGOCD\_PASS \  
--insecure  
  
**Add EKS cluster to Argo CD**

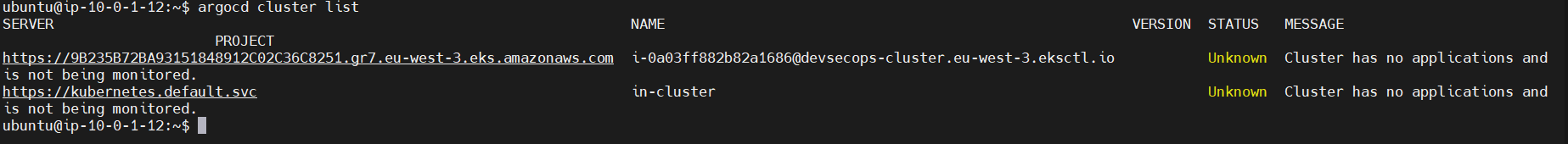
argocd cluster add $(kubectl config current-context)

**Confirm cluster registered**

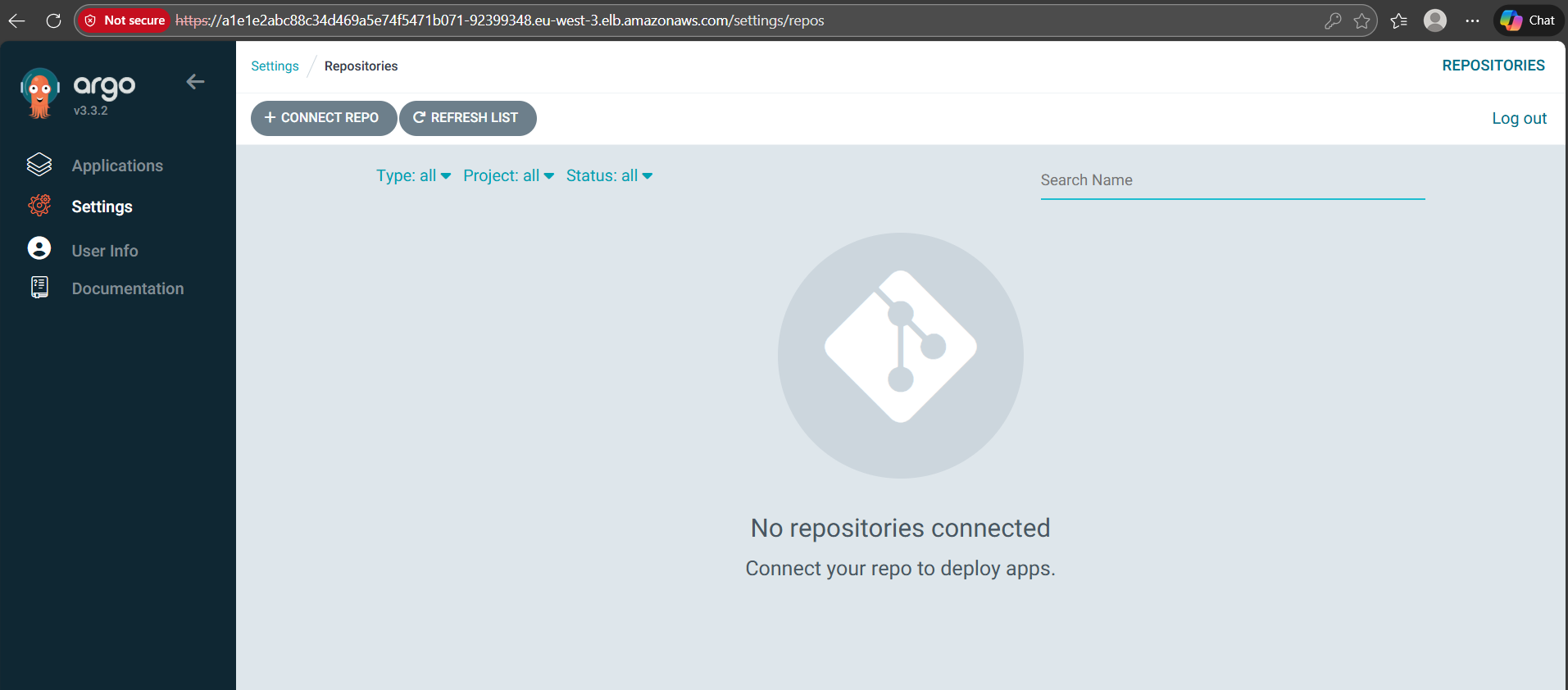
argocd cluster list

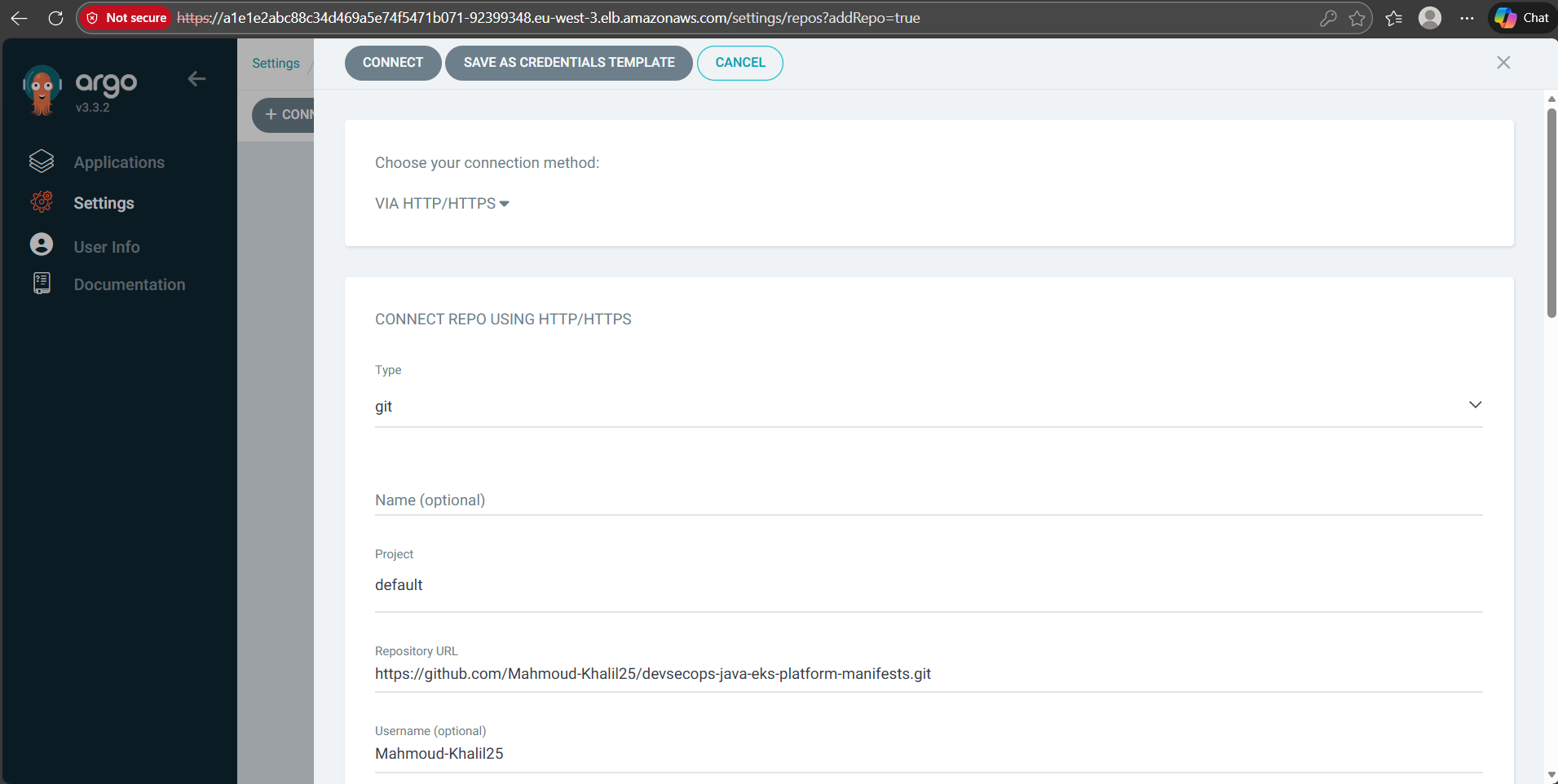
****

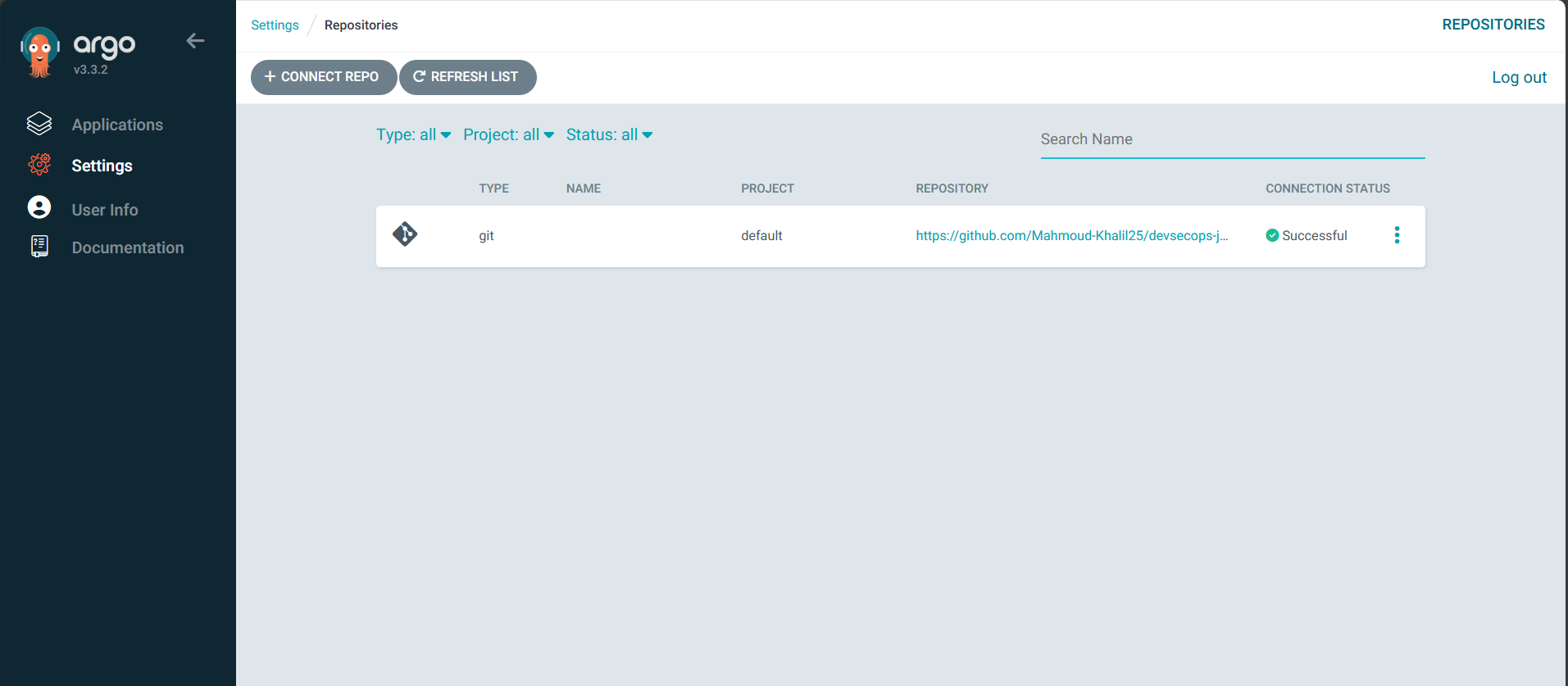
****

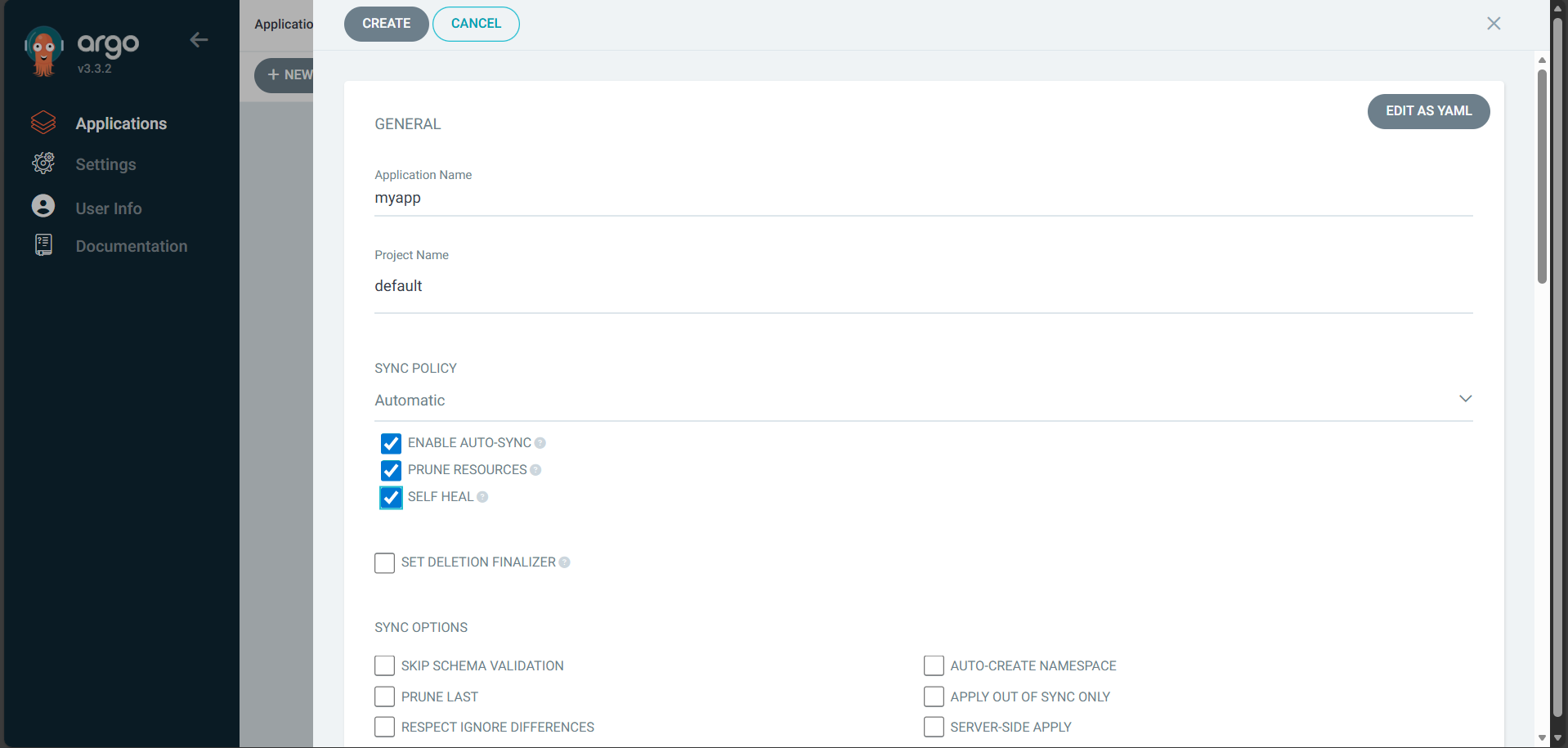
****

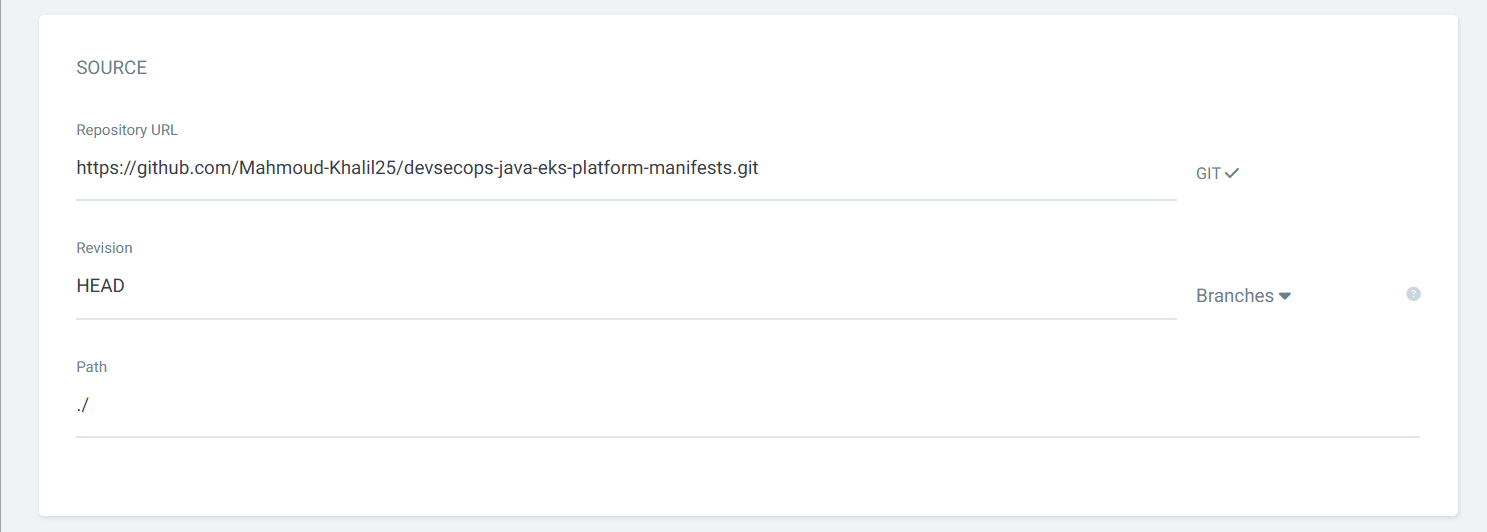
**Connect Manifests Repo to argocd**

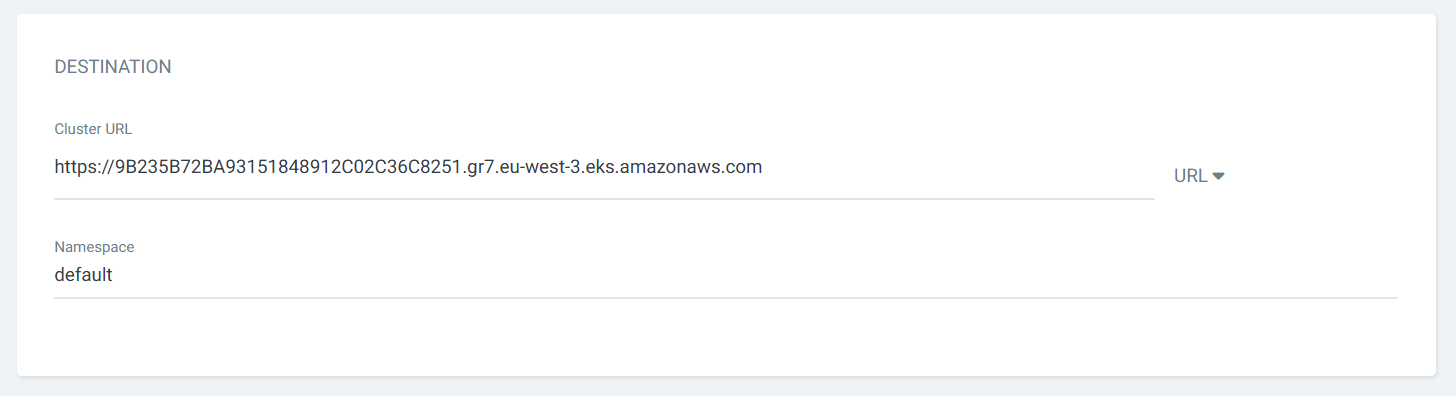
****

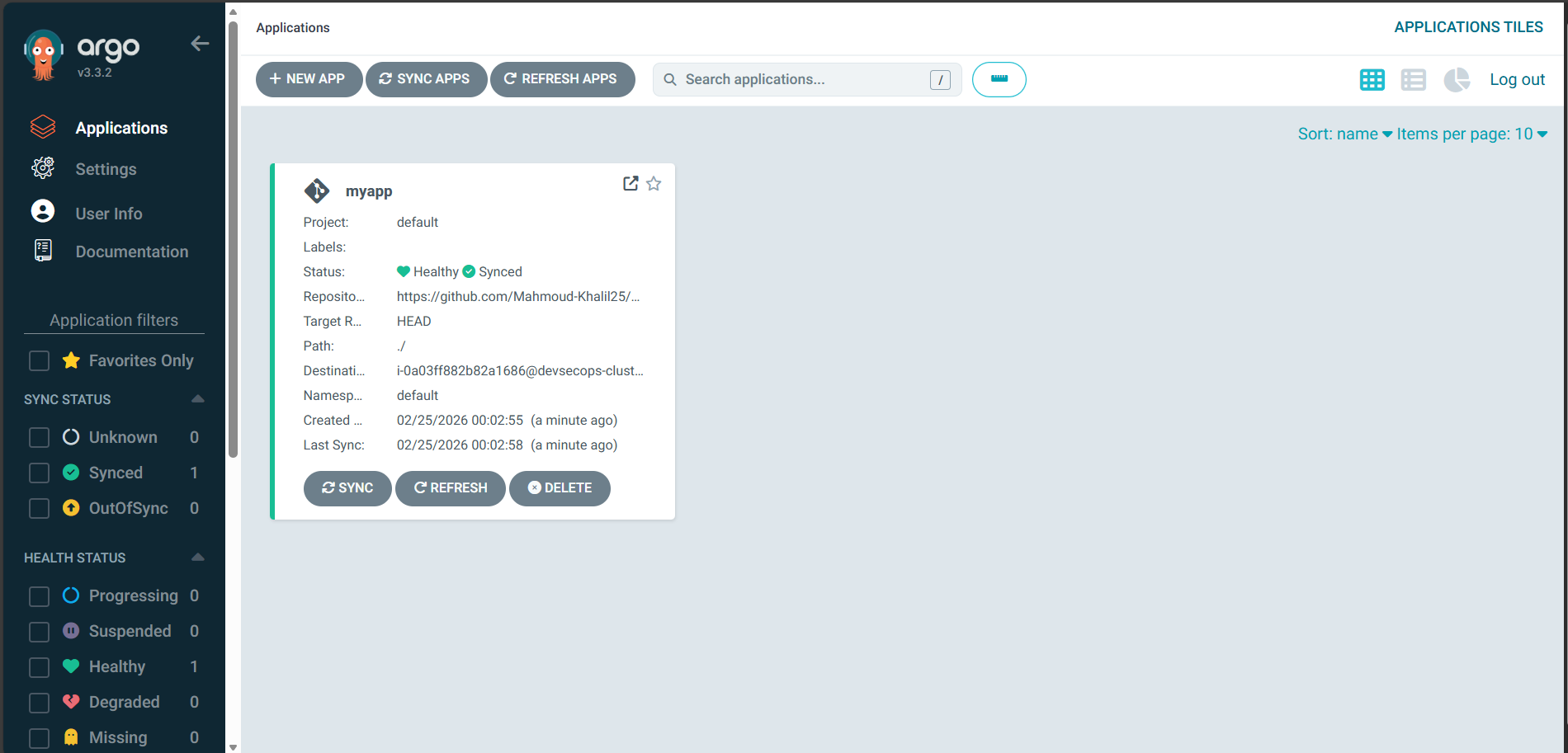
****

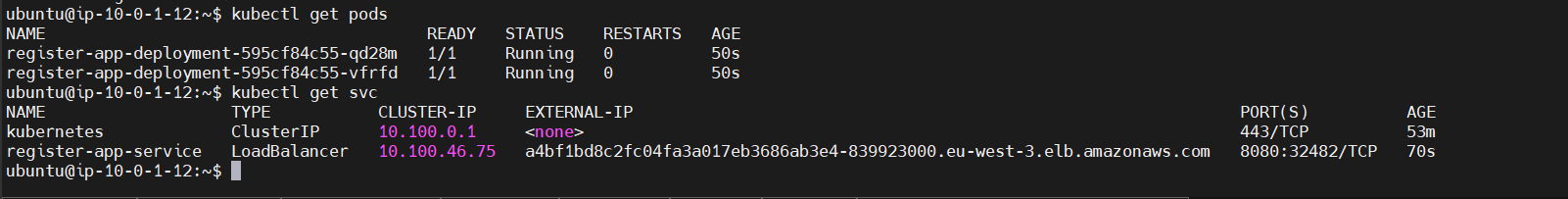
****

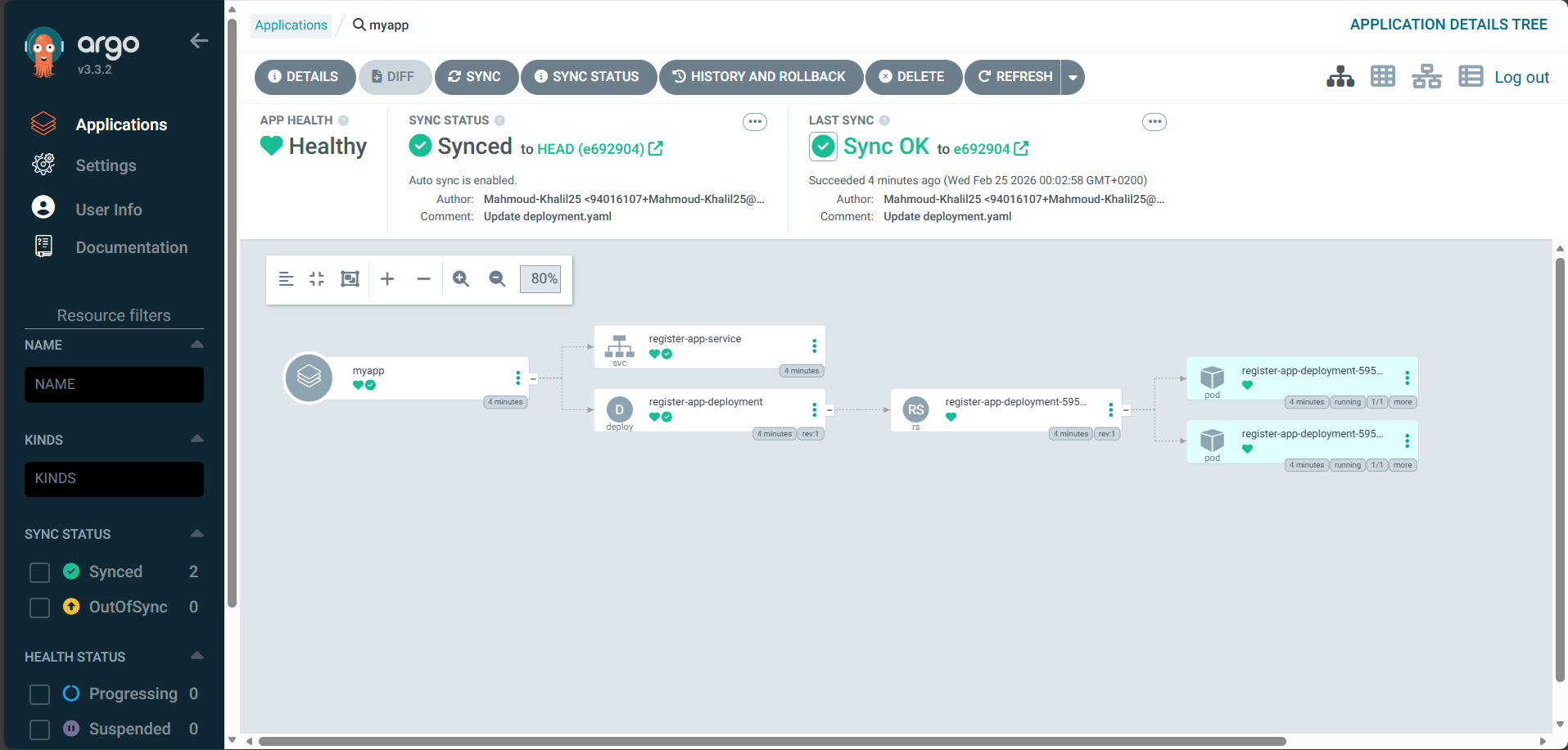
****

****

****

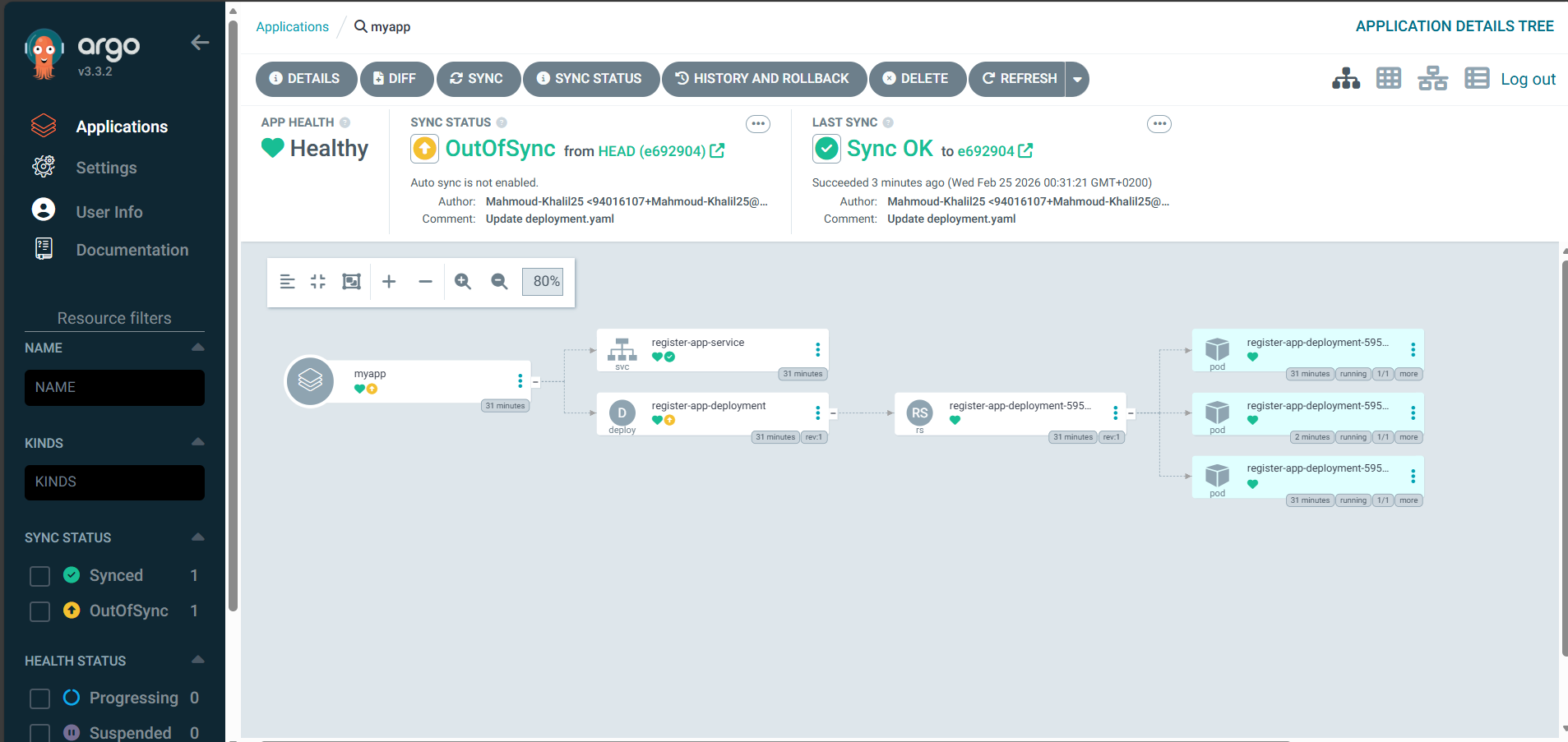
****

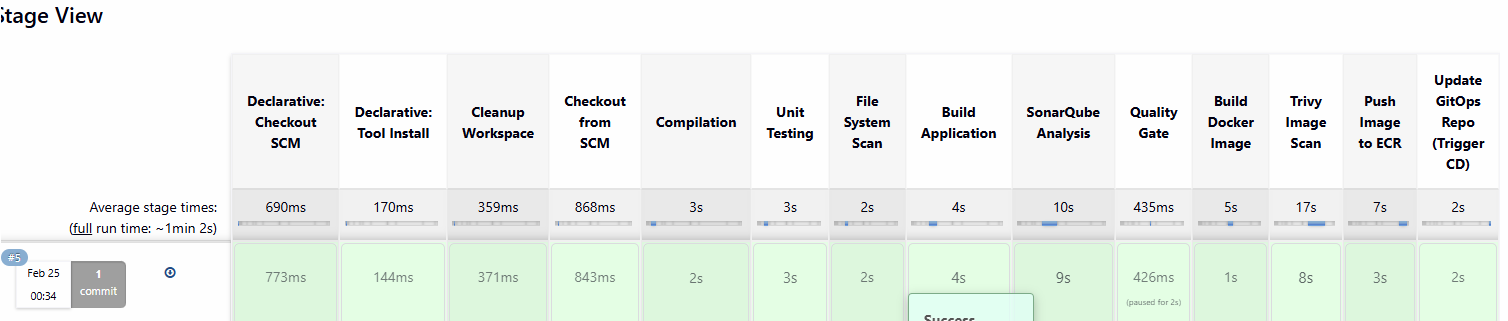
****

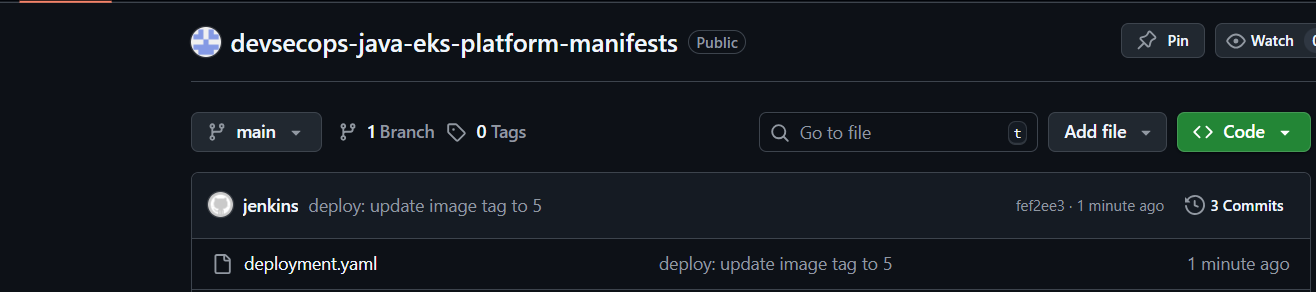
****

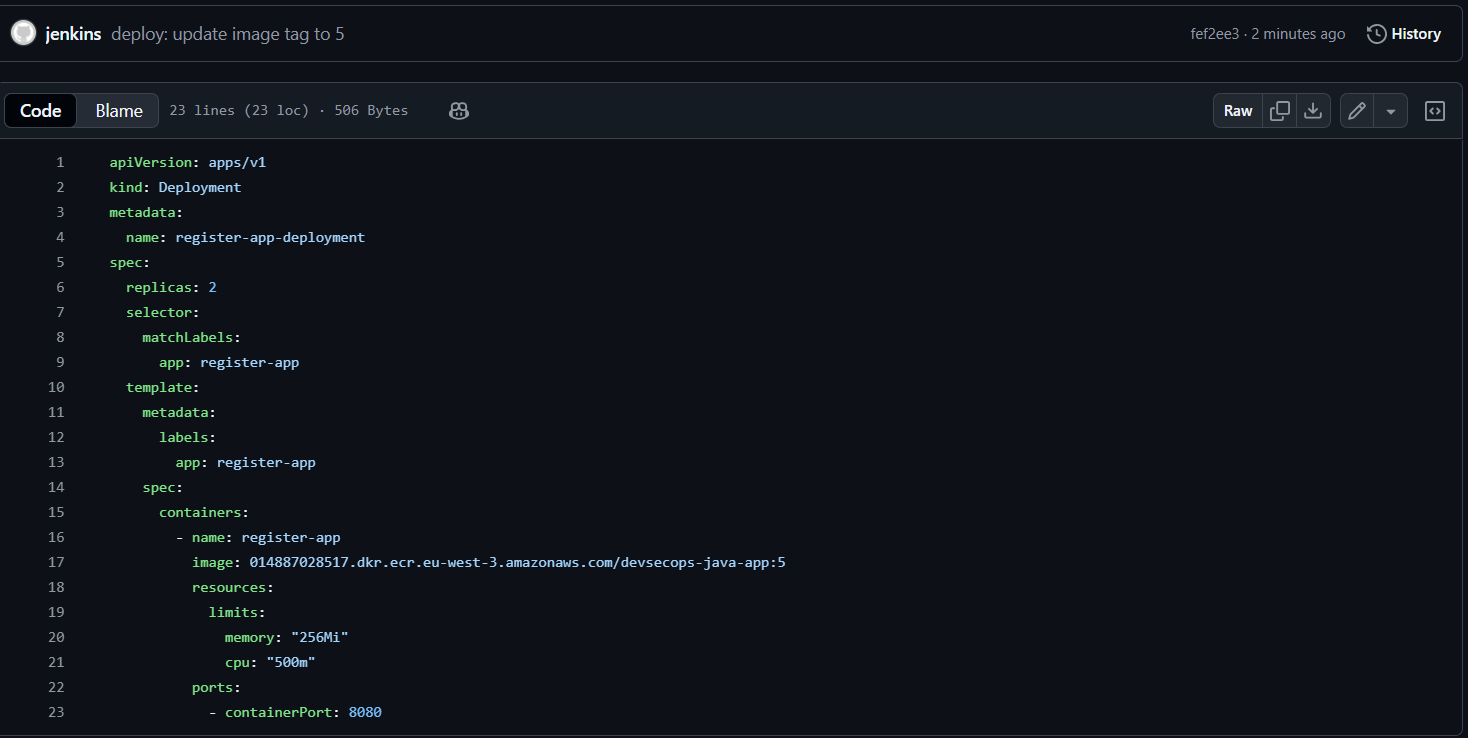
**Causing Manual Drift by changing number of replicas and disabling self heal and autosync to test the effect**

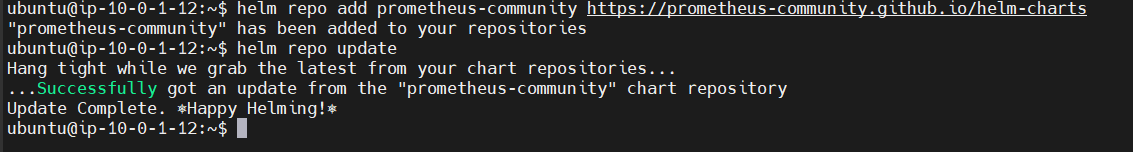
****

****

****

****

****

1. **Add the Prometheus community repo**

helm repo add prometheus-community https://prometheus-community.github.io/helm-charts  
helm repo update

1. **Install kube-prometheus-stack with PVC + external Grafana**

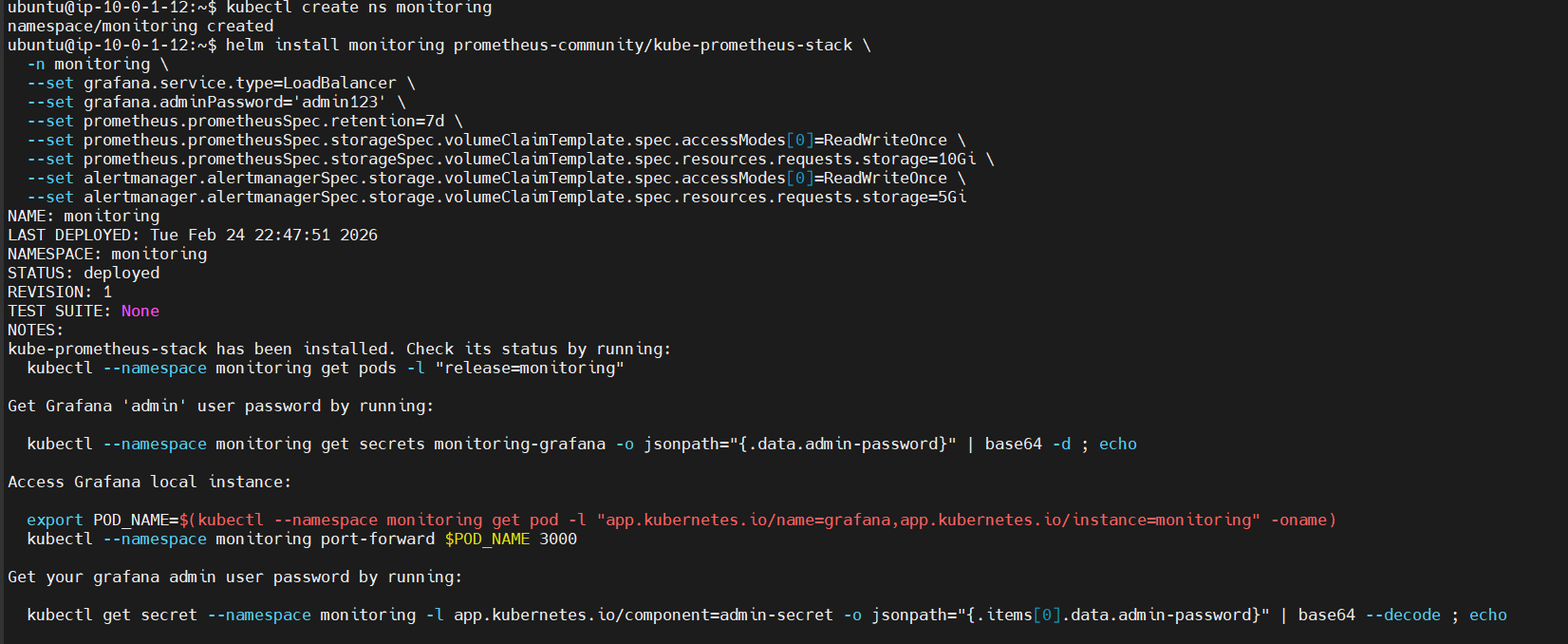
**Create namespace**

kubectl create ns monitoring

**Install with PVC + LB Grafana**

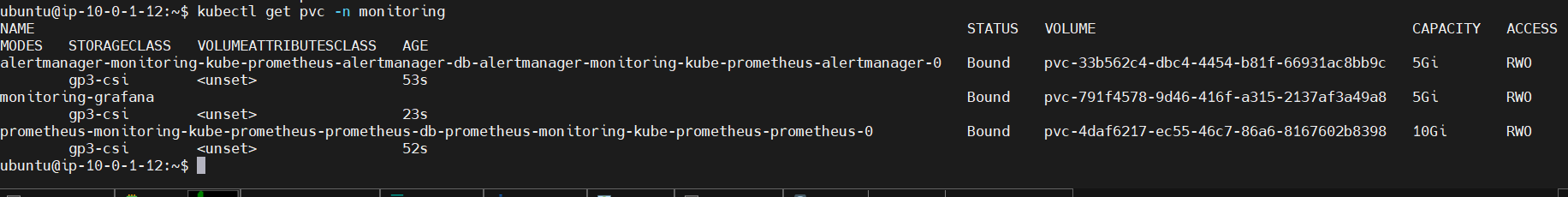
This installs: Prometheus, Grafana, Alertmanager, node-exporter, kube-state-metrics, dashboards.

helm install monitoring prometheus-community/kube-prometheus-stack \  
 -n monitoring \  
 --set grafana.service.type=LoadBalancer \  
 --set grafana.adminPassword='admin123' \  
 --set prometheus.prometheusSpec.retention=7d \  
 --set prometheus.prometheusSpec.storageSpec.volumeClaimTemplate.spec.accessModes[0]=ReadWriteOnce \  
 --set prometheus.prometheusSpec.storageSpec.volumeClaimTemplate.spec.resources.requests.storage=10Gi \  
 --set alertmanager.alertmanagerSpec.storage.volumeClaimTemplate.spec.accessModes[0]=ReadWriteOnce \  
 --set alertmanager.alertmanagerSpec.storage.volumeClaimTemplate.spec.resources.requests.storage=5Gi

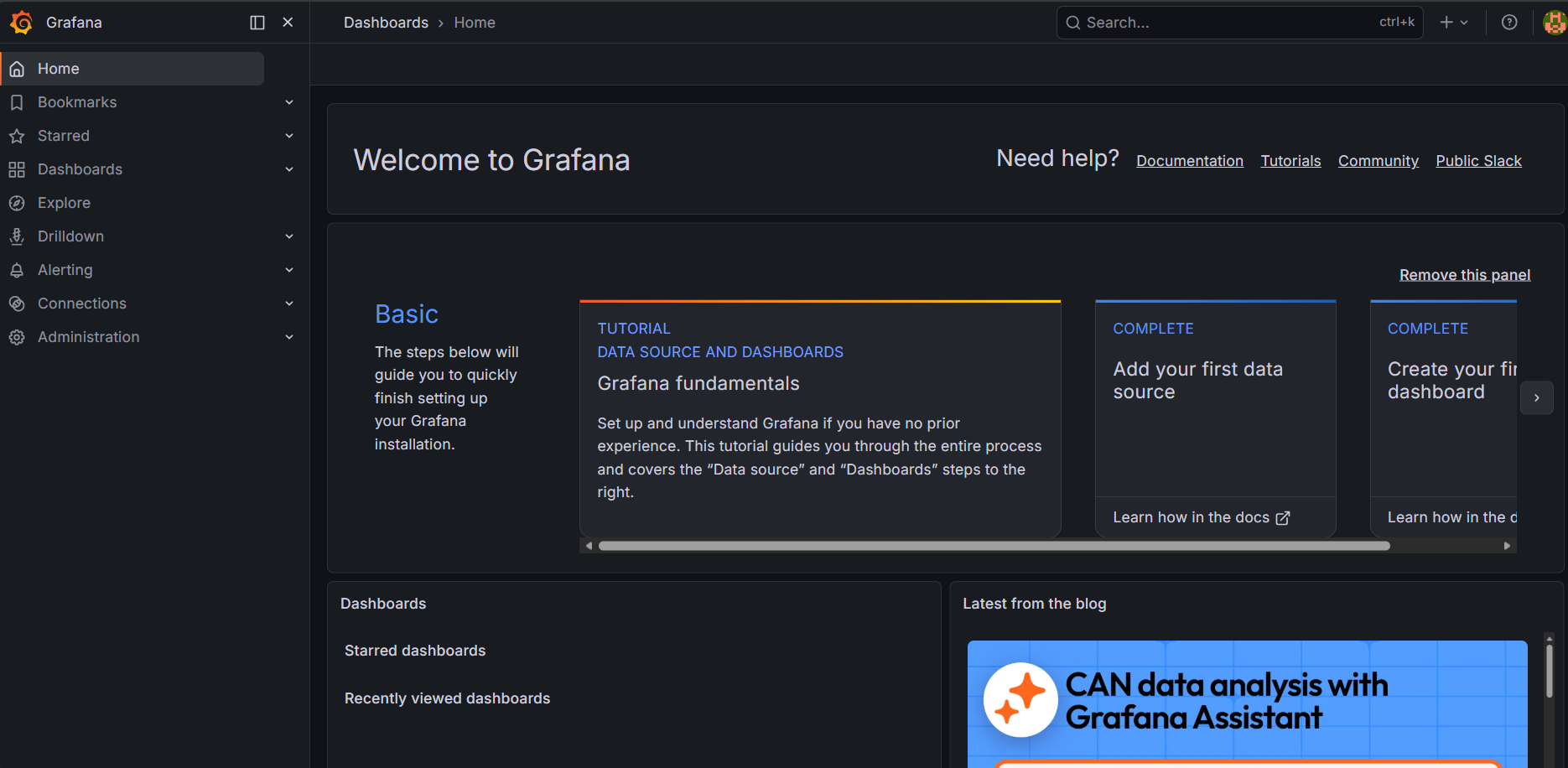


1. **Add Grafana PVC**

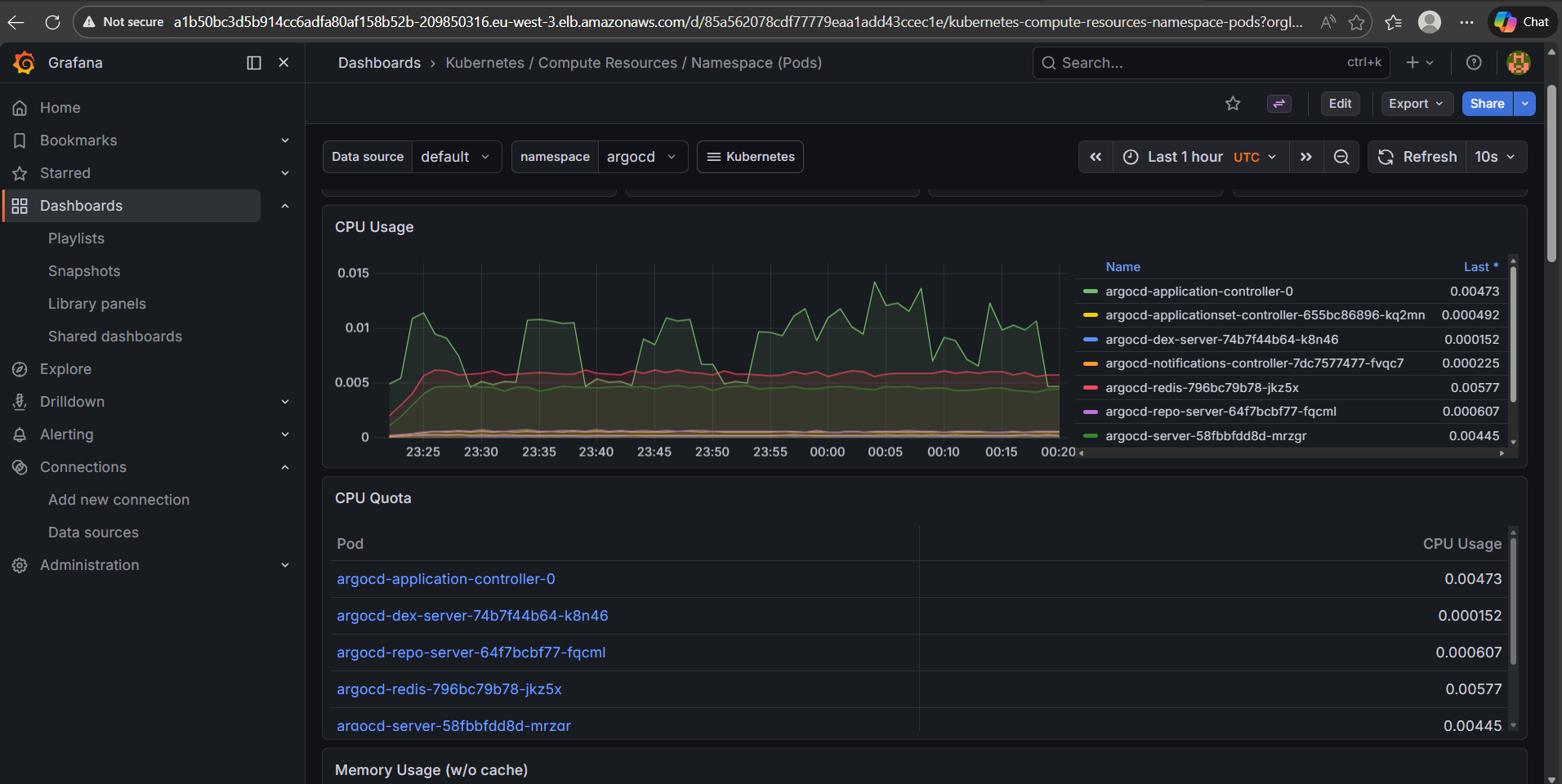
helm upgrade monitoring prometheus-community/kube-prometheus-stack \  
 -n monitoring \  
 --reuse-values \  
 --set grafana.persistence.enabled=true \  
 --set grafana.persistence.size=5Gi

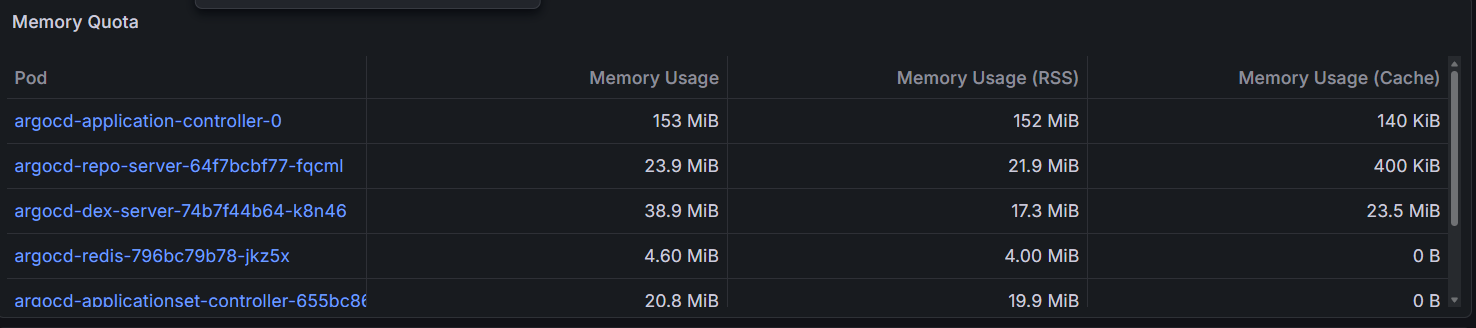
****

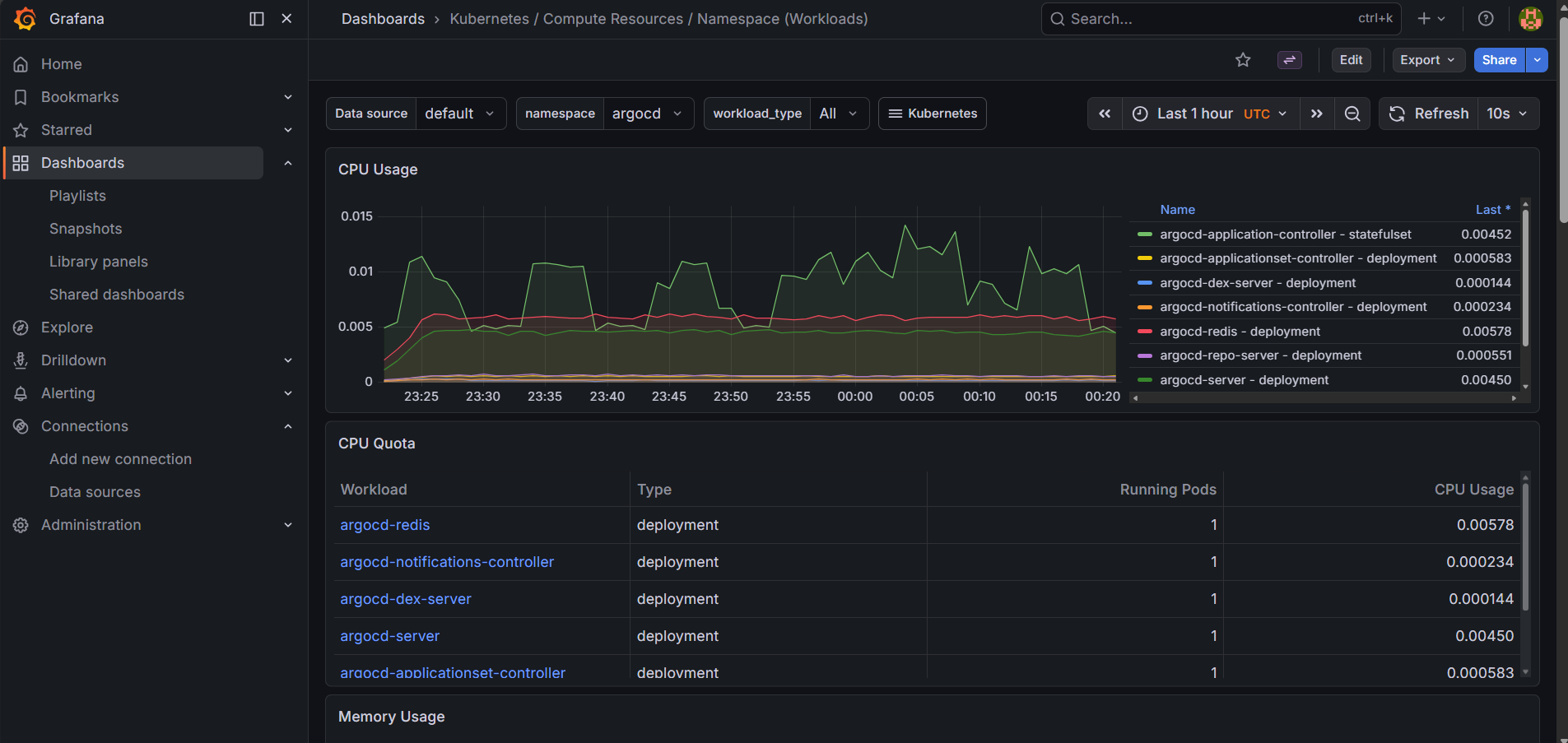
****

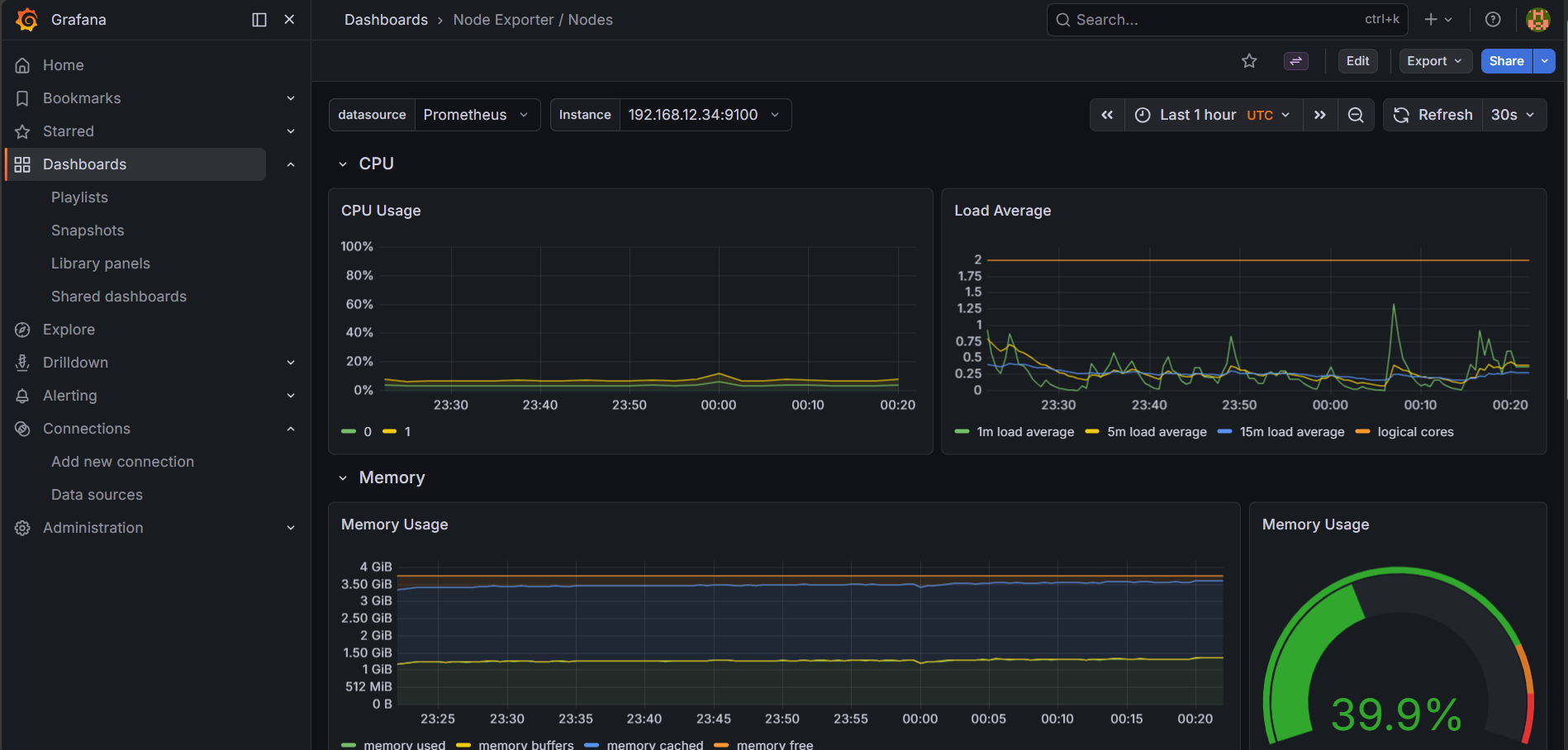
****

**You can see dashboards related to eks**

****

****

****

****