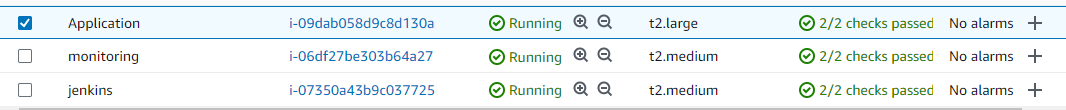
**Deploying a Python Web Application on Kubernetes Cluster for Warner Sisters Inc- Capstone Project**

1. *Create 3 tyre Infra setups, create 3 VMs 1 for Application, 1 for Jenkins and other for Prometheus and Grafana*



1. *Install docker and minikube on application server ,Install docker on Jenkins Server and Install Prometheus and Grafana on Monitoring Server.*

On All 3 servers Install node-exporter.

1. Install Docker and Minikube on Application server.

**Installing docker:**

*sudo apt-get update*

*sudo apt-get install -y ca-certificates curl gnupg*

*sudo install -m 0755 -d /etc/apt/keyrings*

*curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg*

*sudo chmod a+r /etc/apt/keyrings/docker.gpg*

*echo \*

*"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \*

*"$(. /etc/os-release && echo "$VERSION\_CODENAME")" stable" | \*

*sudo tee /etc/apt/sources.list.d/docker.list > /dev/null*

*sudo apt-get update*

*sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin*

*sudo usermod -aG docker ubuntu*

*sudo su - ubuntu*

**Installing minikube:**

*curl -LO* [*https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64*](https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64)

*sudo install minikube-linux-amd64 /usr/local/bin/minikube*

*minikube start*

**Installing kubectl:**

*curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"*

*sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl*

1. Installing Docker & Jenkins on Jenkins server

For docker installation refer to Section 2.a

**Installing Jenkins:**

*sudo apt update*

*sudo apt install fontconfig openjdk-17-jre*

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

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

*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. Installing Prometheus and Grafana on Monitoring server.

**Installing Prometheus:**

*wget https://github.com/prometheus/prometheus/releases/download/v2.29.1/prometheus-2.29.1.linux-amd64.tar.gz*

*tar -xvf prometheus-2.29.1.linux-amd64.tar.gz*

**Installing Grafana:**

*wget -q -O - https://packages.grafana.com/gpg.key | gpg --dearmor | sudo tee /usr/share/keyrings/grafana.gpg > /dev/null*

*echo "deb [signed-by=/usr/share/keyrings/grafana.gpg] https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list*

*sudo apt update*

*sudo apt install -y grafana*

*sudo systemctl start grafana-server*

*sudo systemctl status grafana-server*

1. Installing node-exporter for Prometheus to scrape data

*sudo apt-get install -y prometheus-node-exporter*

1. Git repository of python app:

<https://github.com/amithtg-199/py-app.git>

1. Docker file to build docker image:

*FROM python:latest*

*WORKDIR .*

*RUN apt-get update -y*

*RUN apt install -y python3-pip*

*COPY requirements.txt requirements.txt*

*ADD . ./*

*RUN pip3 install --no-cache-dir -r requirements.txt*

*RUN python3 manage.py makemigrations*

*RUN python3 manage.py migrate*

*EXPOSE 80*

*CMD ["python3", "manage.py", "runserver", "0.0.0.0:80"]*

1. Create k8s deployment, service and hpa yaml file.

File: deployment.yaml

*apiVersion: apps/v1*

*kind: Deployment*

*metadata:*

*name: app-deployment*

*spec:*

*replicas: 3*

*strategy:*

*type: RollingUpdate*

*rollingUpdate:*

*maxSurge: 2*

*maxUnavailable: 1*

*selector:*

*matchLabels:*

*name: app-pod*

*template:*

*metadata:*

*labels:*

*name: app-pod*

*spec:*

*containers:*

*- name: app-pod*

*image: amithtg/py-app:50*

*ports:*

*- containerPort: 80*

*resources:*

*requests:*

*cpu: .2*

*memory: 1Gi*

*limits:*

*cpu: .5*

*memory: 2Gi*

*---*

*apiVersion: v1*

*kind: Service*

*metadata:*

*name: pyapp-service*

*labels:*

*name: pyapp-service*

*spec:*

*type: LoadBalancer*

*ports:*

*- port: 80*

*targetPort: 80*

*nodePort: 30004*

*selector:*

*name: app-pod*

*---*

*apiVersion: autoscaling/v1*

*kind: HorizontalPodAutoscaler*

*metadata:*

*name: hpa-test*

*spec:*

*scaleTargetRef:*

*apiVersion: apps/v1*

*kind: Deployment*

*name: app-deployment*

*minReplicas: 3*

*maxReplicas: 10*

*targetCPUUtilizationPercentage: 50*

1. Deploy the application on k8s on application server

*Kubectl create -f deployment.yaml*

Enable assign the EC2 private IP as external IP

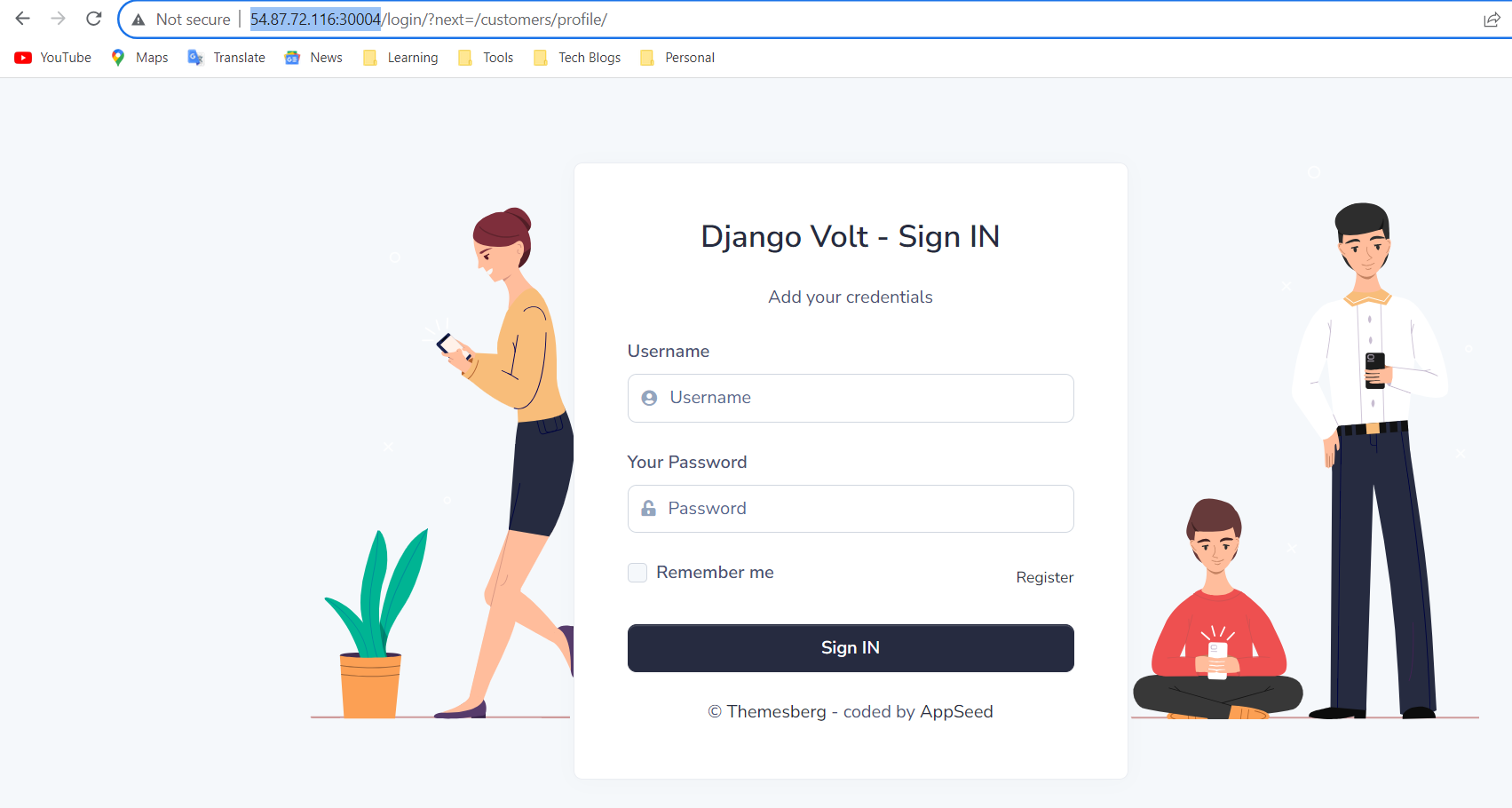
*kubectl patch svc pyapp-service -p '{"spec":{"externalIPs":["10.10.1.208"]}}'*

Enable Port forwarding on k8s to access the application over internet:

*nohup kubectl port-forward --address 0.0.0.0 services/pyapp-service 30004:80 > /dev/null 2>&1 &*

Now you can access the application using public IP of Application server EC2, as we have used NodePort=30004 we need to use the same port, open 30004 port to public on EC2 Security Group.

http:// <Public\_IP>:30004/

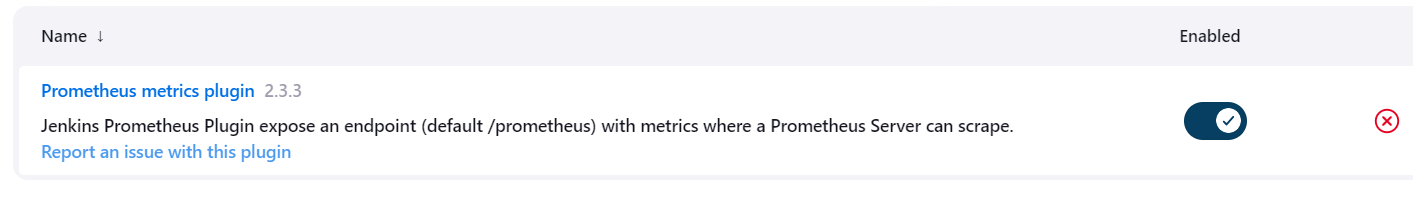


1. Integrating Prometheus and Grafana with Application and Jenkins server
2. Initialize Jenkins and install Prometheus plugin

Jenkins will use 8080 port, so open port 8080 on security group.

Initialize Jenkins as prompted and navigate to *Manage Jenkins > plugins*

Install Prometheus plugin on Jenkins



1. Configure Prometheus.yaml file to integrate the application and Jenkin server nodes

- job\_name: "Prometheus" 🡪 **To monitor HTTP Server of Prometheus**

# metrics\_path defaults to '/metrics'

# scheme defaults to 'http'.

static\_configs:

- targets: ["localhost:9090"]

- job\_name: "jenkins-node" 🡪 **To monitor Jenkins node status**

# metrics\_path defaults to '/metrics'

# scheme defaults to 'http'.

static\_configs:

- targets: ["10.10.1.120:9100"]

- job\_name: "jenkins-tool" **🡪 To Monitor Jenkins Application**

# metrics\_path defaults to '/metrics'

# scheme defaults to 'http'.

metrics\_path: "/prometheus"

static\_configs:

- targets: ["10.10.1.120:8080"]

- job\_name: "application-node" **🡪 To Monitor Application & Prometheus node**

# metrics\_path defaults to '/metrics'

# scheme defaults to 'http'.

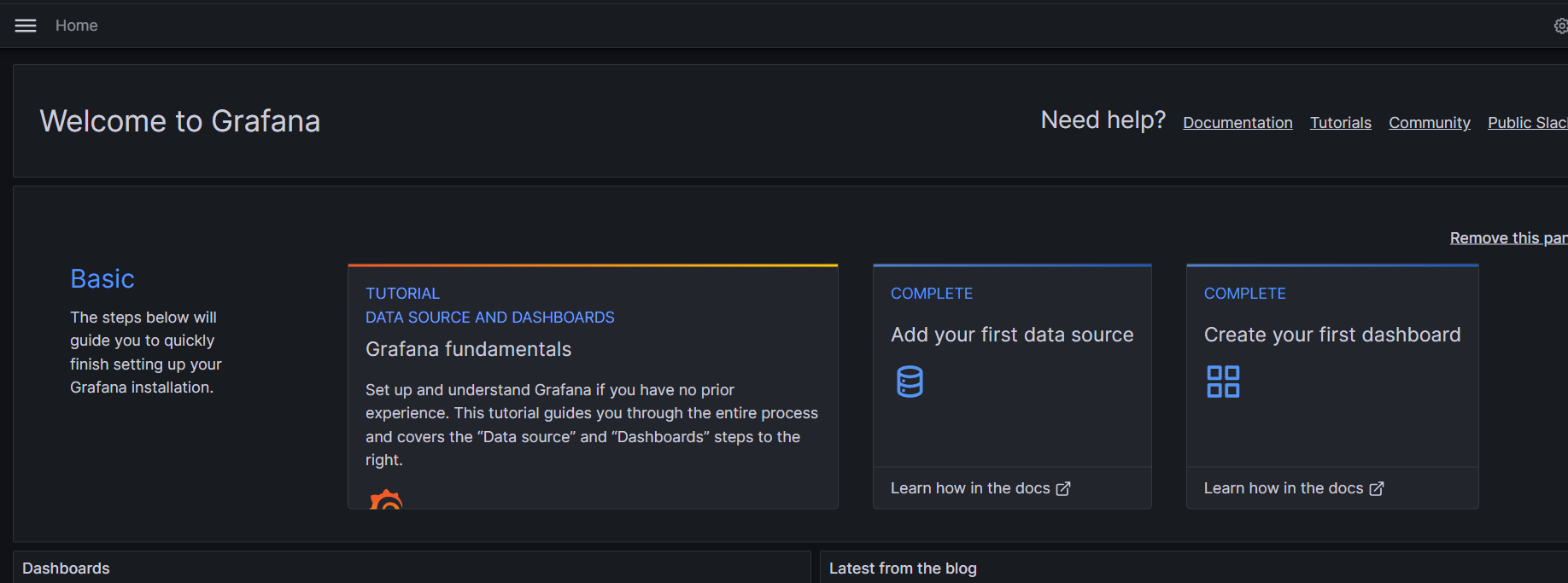
static\_configs:

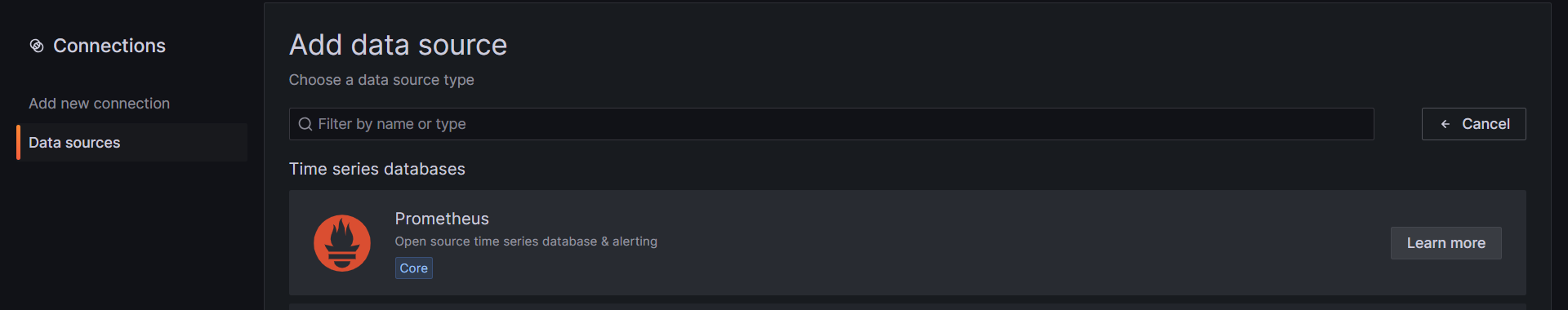
- targets: ["10.10.1.208:9100"]

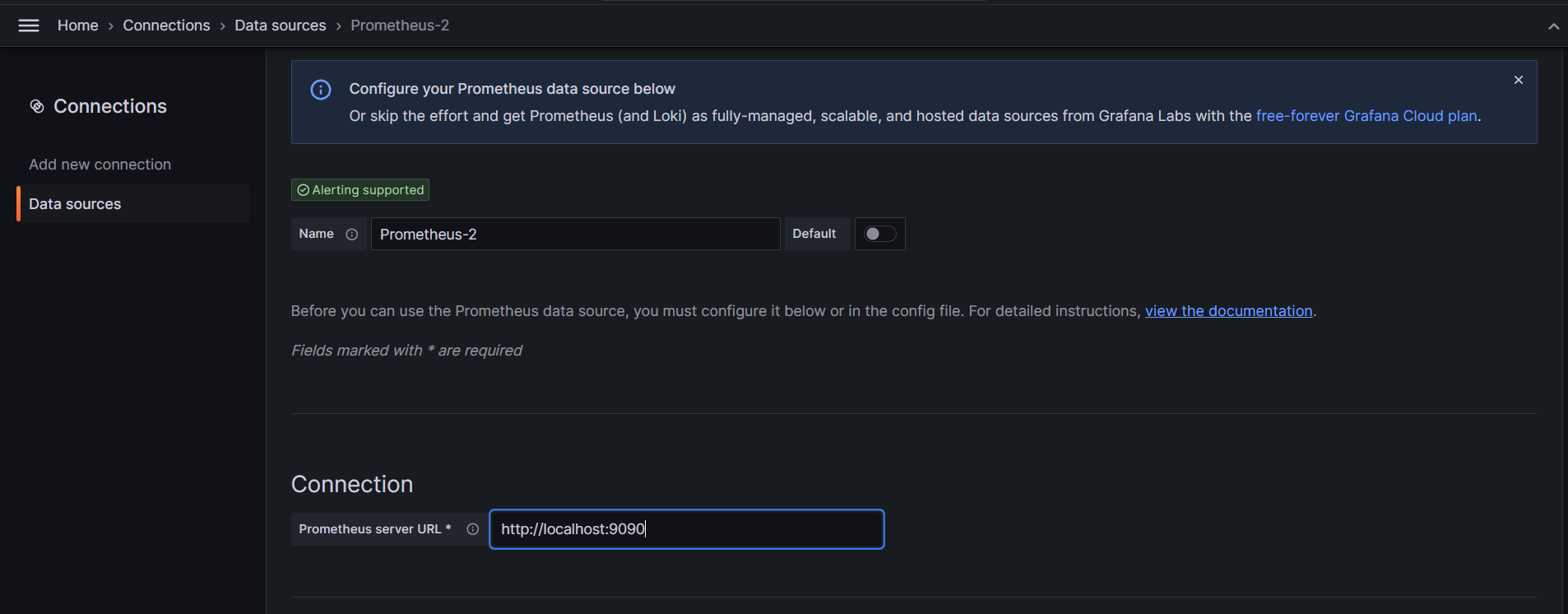
- targets: ["localhost:9100"]

**Note:** Node-port will use port 9100, Grafana uses 3000 port and Prometheus uses 9090 port, which all needs to be opened on Monitoring EC2 SG.

1. Adding Grafana Dashboard for monitoring Node status of 3 server and Jenkins application
2. Add Prometheus DS on Grafana

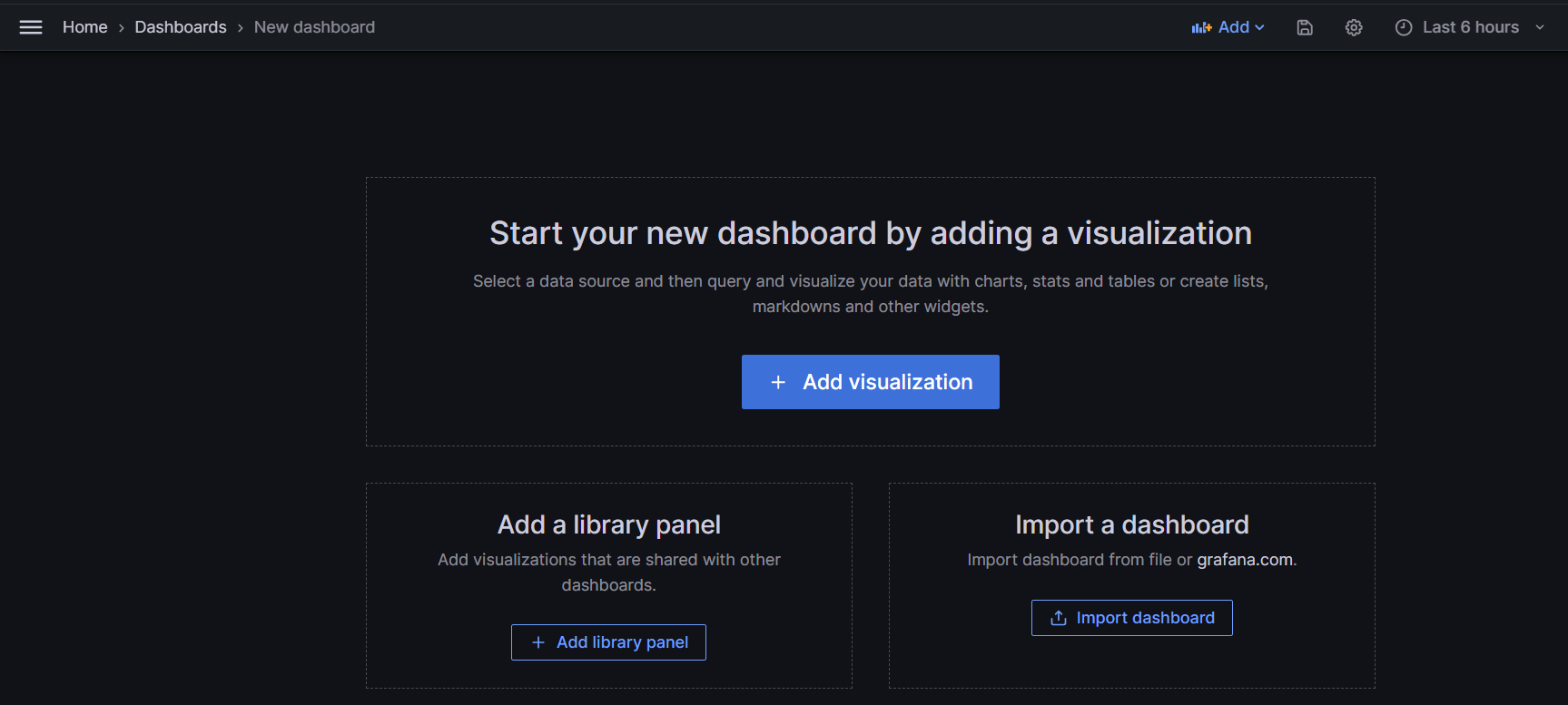




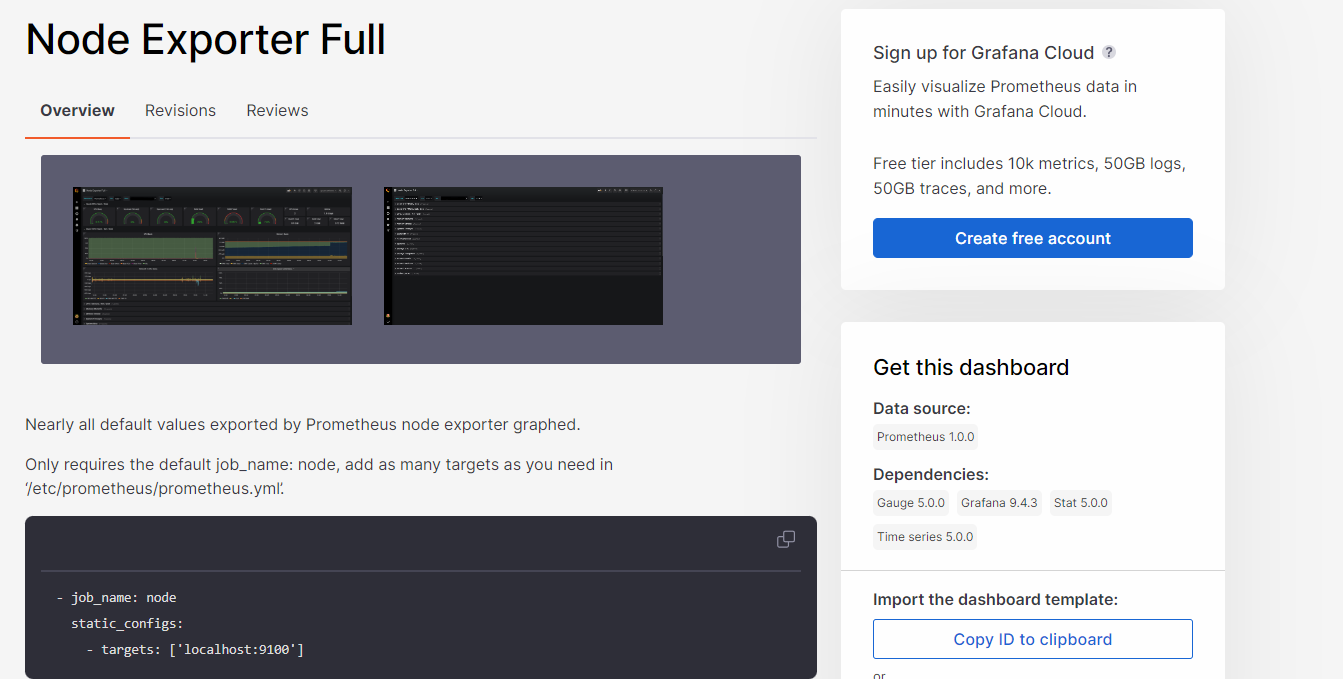


1. Import Node exporter and Jenkins dashboard

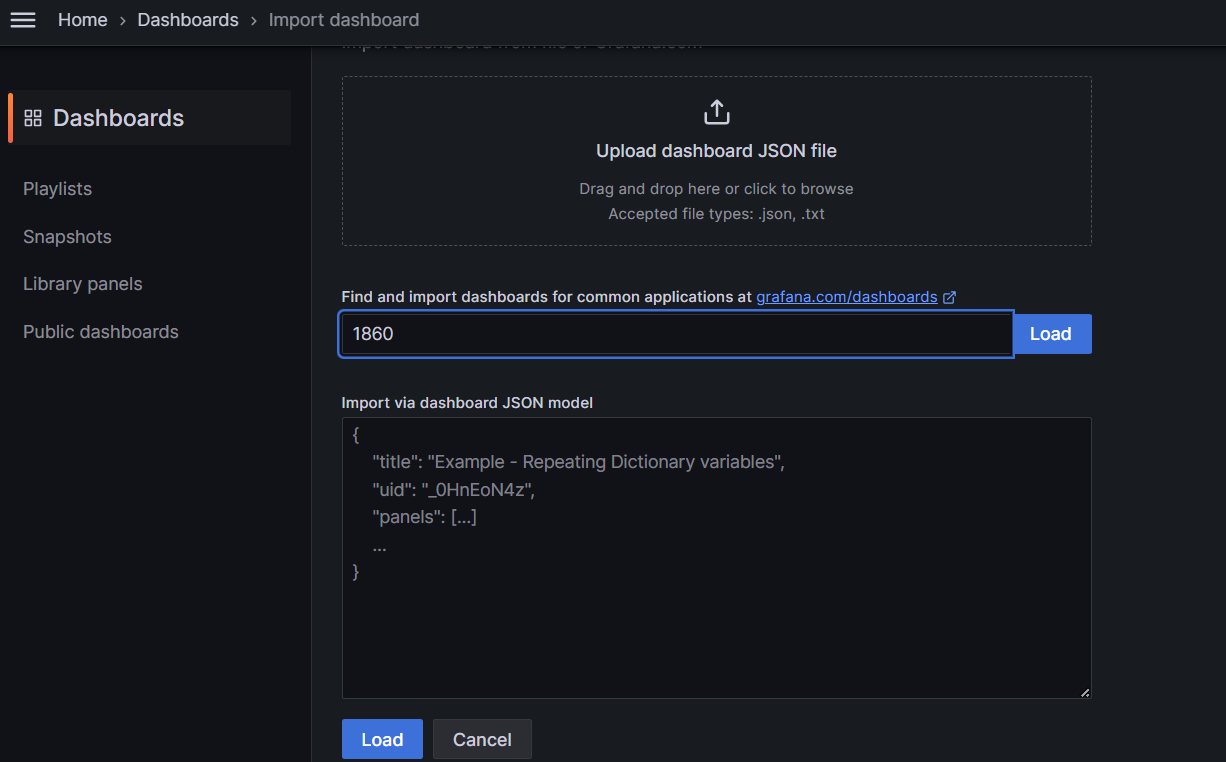
Click on New DashBorad > Import Dashboard



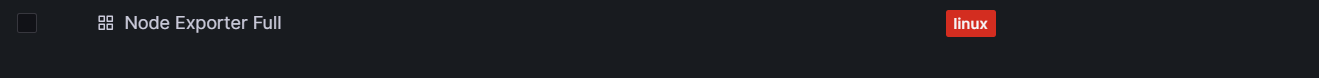
Copy ID of node-exporter



And Add it into the dashboard and load it

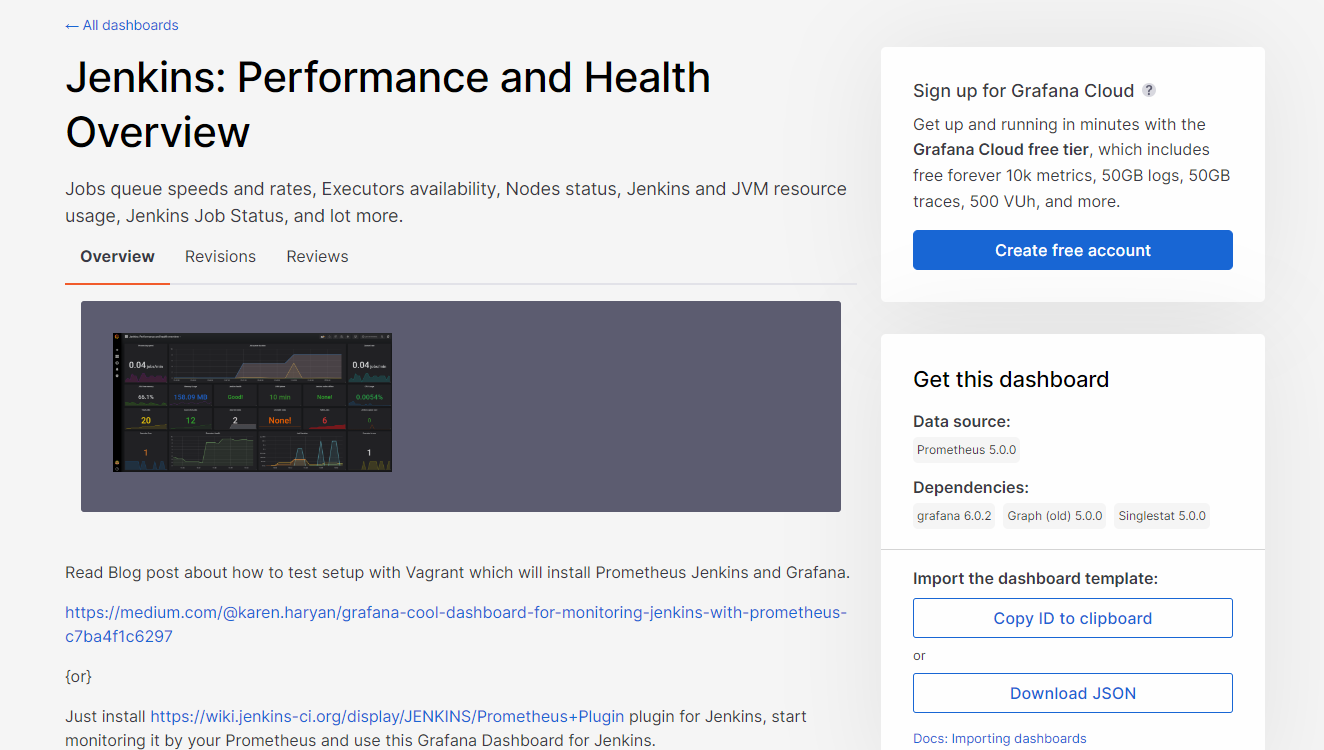


Load and Import it under the Prometheus Data source



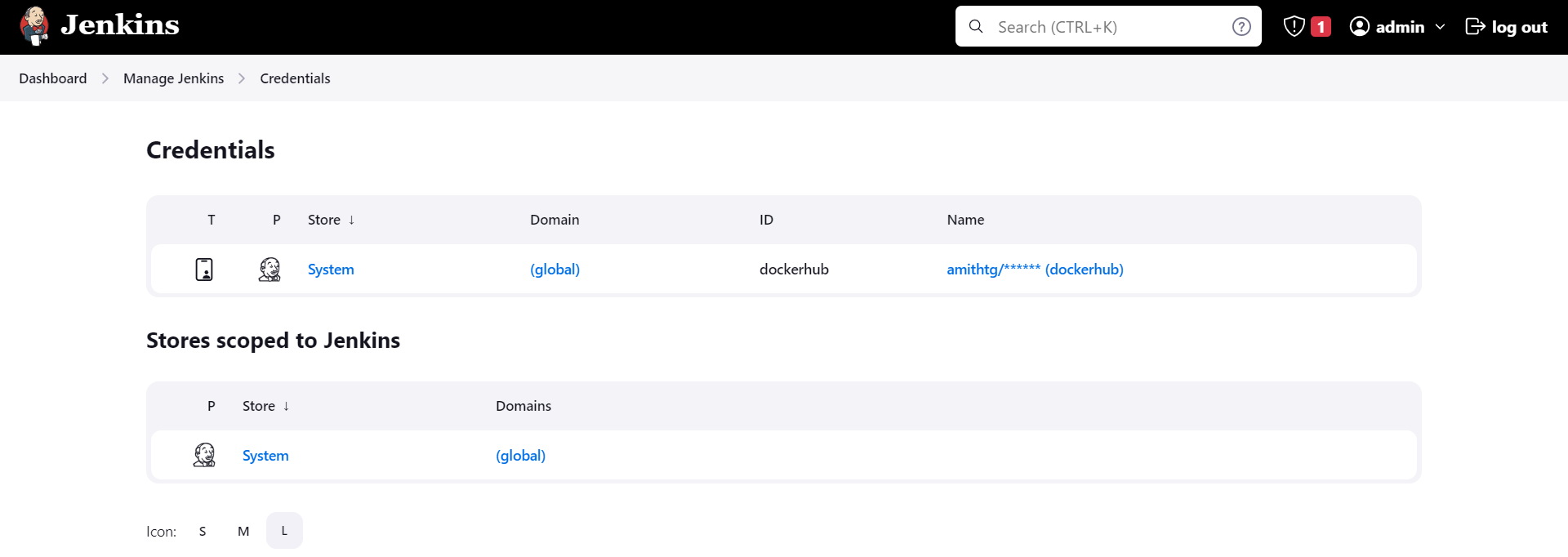
1. Import Jenkins Dashboard

Copy the ID of Jenkins DashBoard

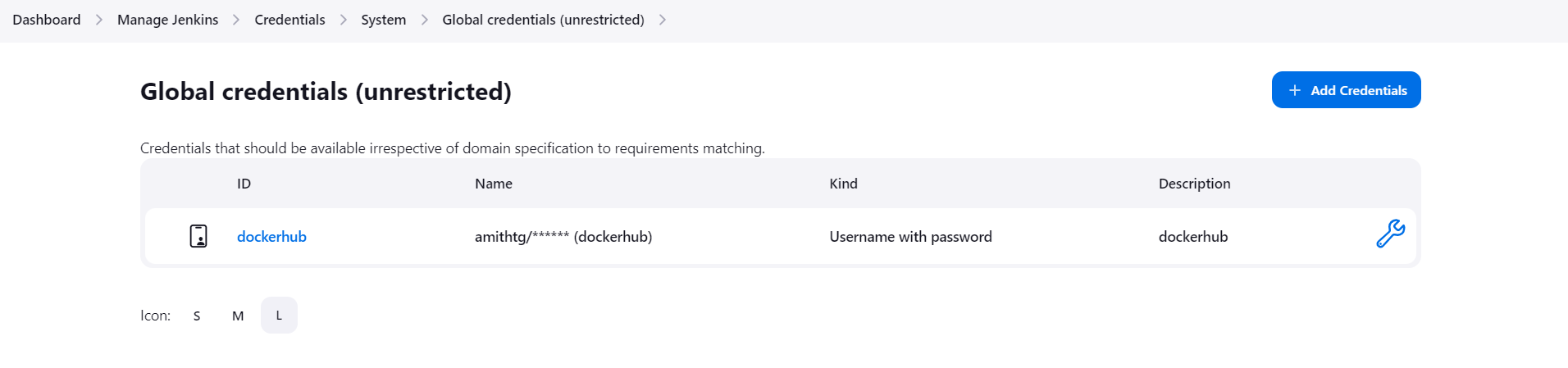


And import it into Grafana

1. Create Jenkins CI-CD flow to auto detect GIT changes and deploy the new version
2. *Add Docker hub credentials on Jenkins*



Click on global and Create User:



1. *Create Jenkins file as below*

pipeline {

agent any

options {

buildDiscarder(logRotator(numToKeepStr: '5'))

}

environment {

DOCKERHUB\_CREDENTIALS = credentials('dockerhub')

}

triggers {

pollSCM '\* \* \* \* \*'

}

stages {

stage('Build') {

steps {

script {

sh """

#!/bin/bash

git clone 'https://github.com/amithtg-199/py-app.git'

cd /var/lib/jenkins/workspace/pythonapp/py-app

docker build -t amithtg/py-app:"${env.BUILD\_NUMBER}" .

"""

}

}

}

stage('docker\_push') {

steps {

script {

sh """

#!/bin/bash

echo $DOCKERHUB\_CREDENTIALS\_PSW | docker login -u $DOCKERHUB\_CREDENTIALS\_USR --password-stdin

docker push amithtg/py-app:"${env.BUILD\_NUMBER}"

"""

}

}

}

stage('Deploy') {

steps {

script {

sh """

#!/bin/bash

ssh -i /var/lib/jenkins/my\_pair.pem ubuntu@ec2-54-87-72-116.compute-1.amazonaws.com << EOF

sed -i 's|amithtg/py-app.\*|amithtg/py-app:${env.BUILD\_NUMBER}|g' deployment.yaml

kubectl apply -f deployment.yaml

kubectl patch svc pyapp-service -p '{"spec":{"externalIPs":["10.10.1.208"]}}'

exit 0

<< EOF

"""

}

}

}

}

post {

always {

sh 'docker logout'

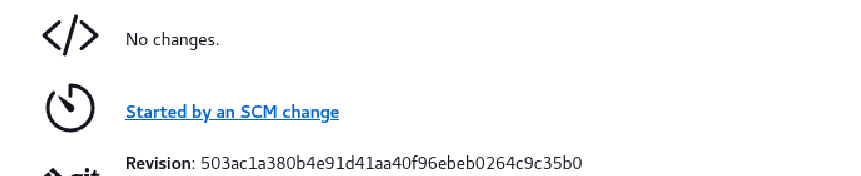
cleanWs notFailBuild: true

}

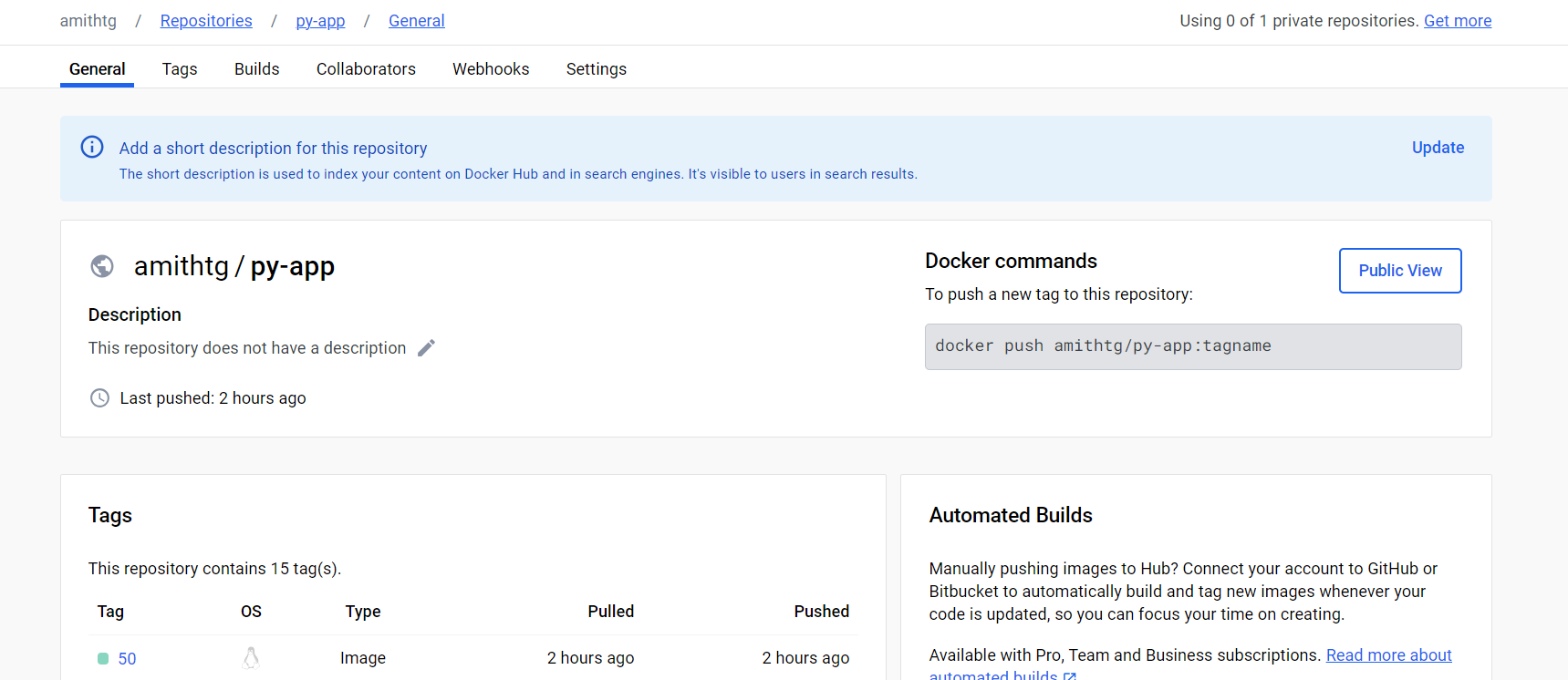
}

}

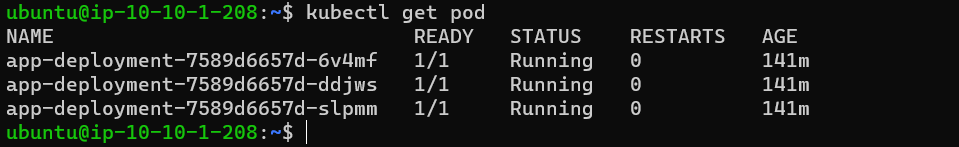
1. Execute the Build after the SCM changes were made.



*Check the image on docker hub*

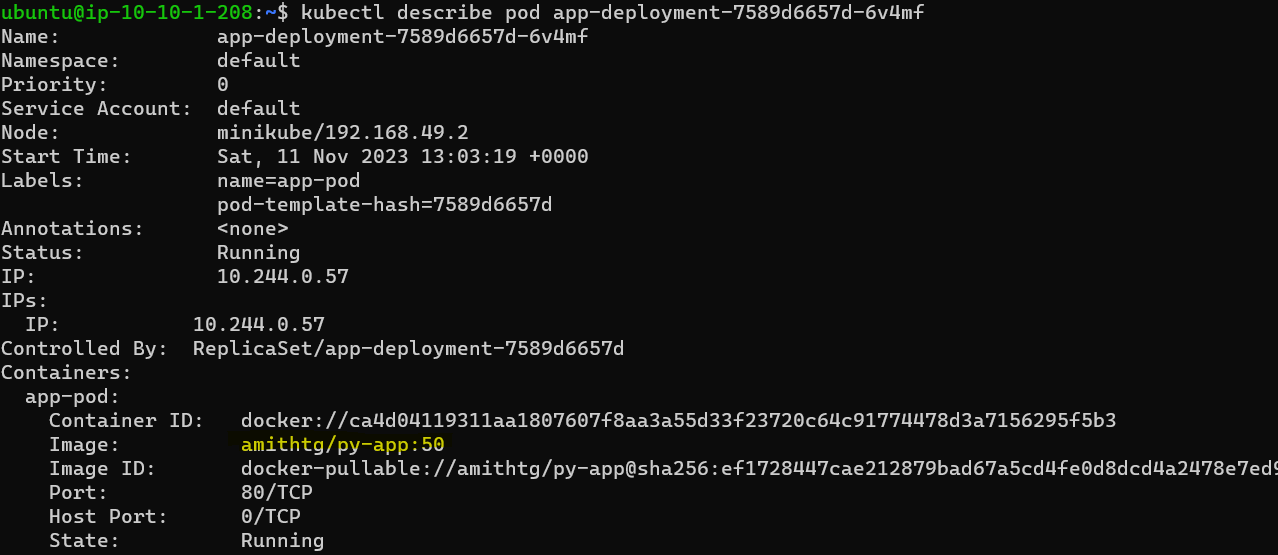


*Check deployment-pods*



Enable Port Forwarding:

nohup kubectl port-forward --address 0.0.0.0 services/pyapp-service 30004:80 > /dev/null 2>&1 &



Check Application:

