*Internship Report on*

**INFRASTRUCTURE AND CI/CD PIPELINE IMPLEMENTATION TO DEPLOY AN APPLICATION**

*Submitted in partial fulfilment of the requirements for the Award of Degree of*

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

by

**KANDULAPATI LEKHANA GAYATRI**

**(21B91A05D9)**



**S.R.K.R ENGINEERING COLLEGE (A)**

SRKR MARG, CHINNA AMIRAM, BHIMAVARAM-534204, A.P

(Recognized by A.I.C.T.E New Delhi) (Accredited by NBA & NAAC)

(Affiliated to JNTU, KAKINADA)

[2023-2024]

**S.R.K.R ENGINEERING COLLLEGE(A)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**CERTIFICATE**

This is to certify that the Summer Internship Report titled “Infrastructure and CI/CD pipeline implementation to deploy an Application” is the bonafide work done by Miss.Kandulapati Lekhana Gayatri(21B91A05D9) at the end of third year second semester at NIELIT, CHENNAI in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

**Dean -T & P Cell Internship Coordinator Head of the Department**

**SELF DECLARATION**

The Summer Internship Project report entitled “Infrastructure and CI/CD pipeline implementation to deploy an Application” has been carried out by me in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in the Department of Computer Science and Engineering, Sagi Rama Krishnam Raju Engineering College (A), Bhimavaram. I hereby declare that this work has not been submitted to any other university/institute for the award of any other degree/diploma.

**Roll. No. Name Signature**

21B91A05D9Kandulapati Lekhana Gayatri

**INDEX**

**S.NO Contents Page No.**

1. ABSTRACT 1

2. BLOCK DIAGRAM 2

3. HARDWARE AND SOFTWARE TOOLS 3

4. IMPLEMENTATION 4-10

5. SCRIPTS & MANIFESTS 11-21

6. CONCLUSION 22

**ABSTRACT**

This paper presents the deployment of a Flask-based bookstore website, utilizing a robust cloud infrastructure orchestrated with Terraform on AWS. The architecture includes a master node for Kubernetes, two web server nodes, and dedicated instances for Jenkins and SonarQube. Configuration management is achieved through Ansible, facilitating the setup of the Kubernetes cluster using kubeadm and the installation of Jenkins and SonarQube. The continuous integration and continuous deployment (CI/CD) pipeline, orchestrated by Jenkins, incorporates key stages such as code checkout, static code analysis via SonarQube, Docker containerization, and deployment to the Kubernetes cluster. Post-deployment, the infrastructure is monitored comprehensively with Prometheus for real-time metrics collection and Grafana for performance visualization, ensuring the health and stability of both the applications and underlying infrastructure.

**BLOCK DIAGRAM**



**Hardware & Software Tools**

**Hardware:**

4GB RAM, 2.5 GHZ,CPU,300GB HDD, ec2(t3.large)

**Software:**

Operating System: Ubuntu

**Tools & Applications:**

Terraform (infrastructure orchestration)

Ansible (configuration management)

Kubernetes (container orchestration)

Jenkins (CI/CD automation)

SonarQube (static code analysis)

Docker & Docker Compose (containerization)

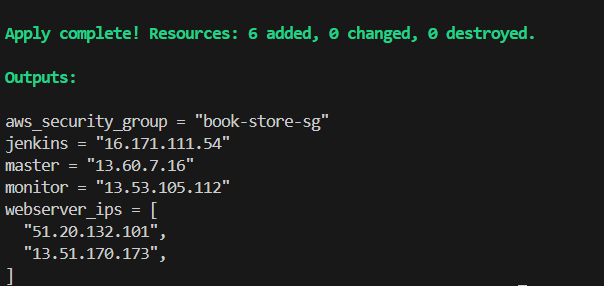
Prometheus & Grafana (monitoring)

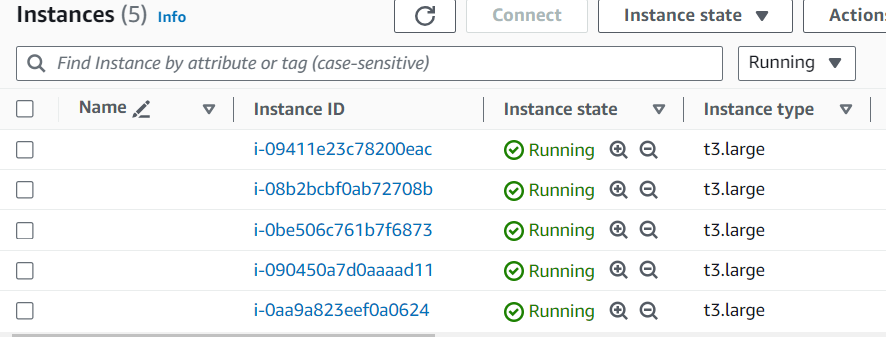
**IMPLEMENTATION**

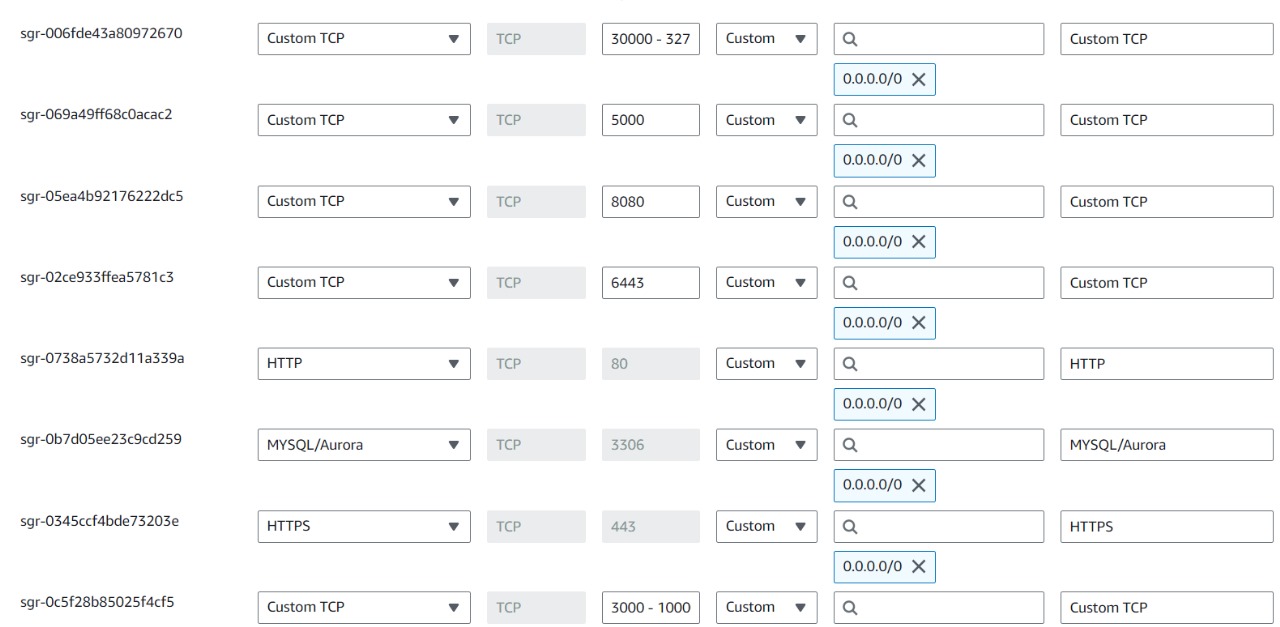
**Terraform:**

Begin by installing Terraform on the machine and initialize it by executing the command terraform init. This command will create the necessary configuration files.To enable remote access to the state file, store it in an **S3 bucket**, while using **DynamoDB** to facilitate state file locking. Review the resources that will be created by executing the command **terraform plan.**Finally, create the infrastructure by running the command **terraform apply –auto-approve.**And open the following ports in the security group that has to attached to every instance created for setting up the infrastructure

.



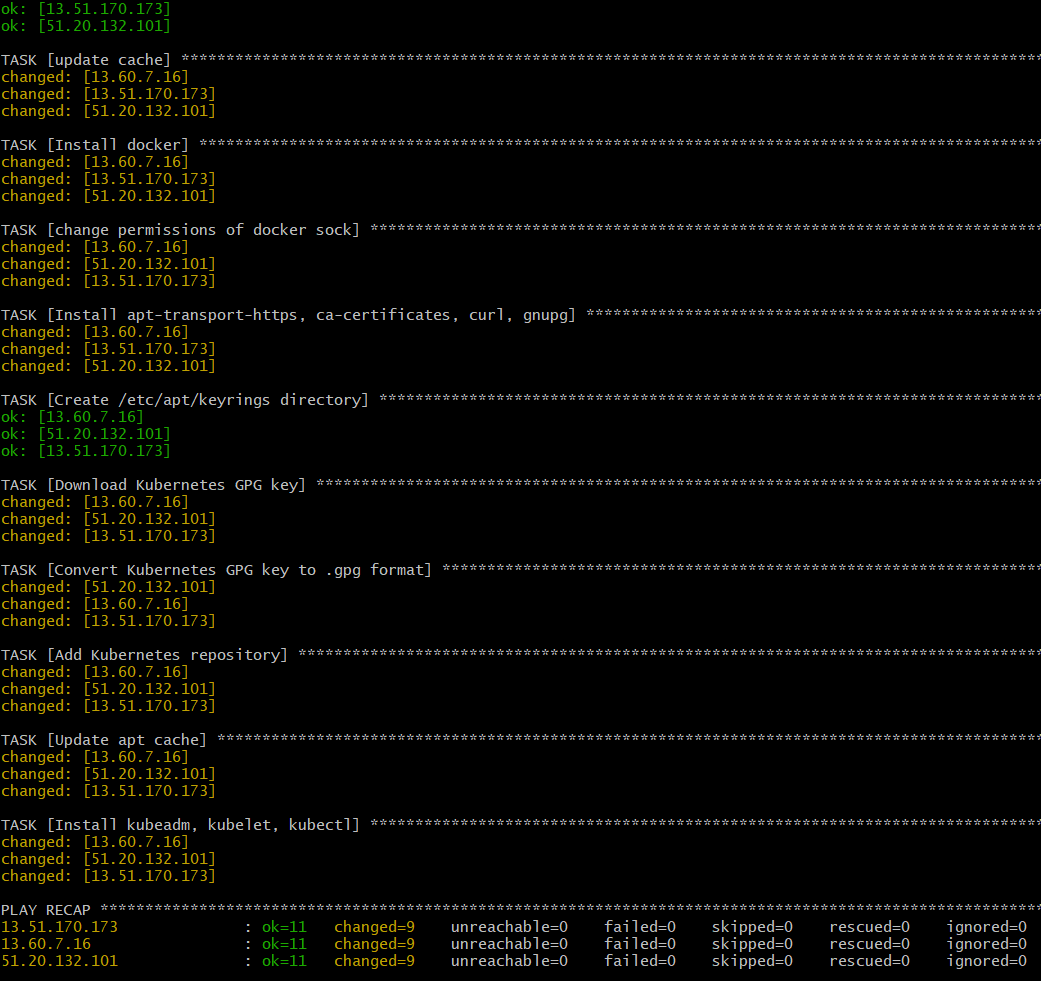




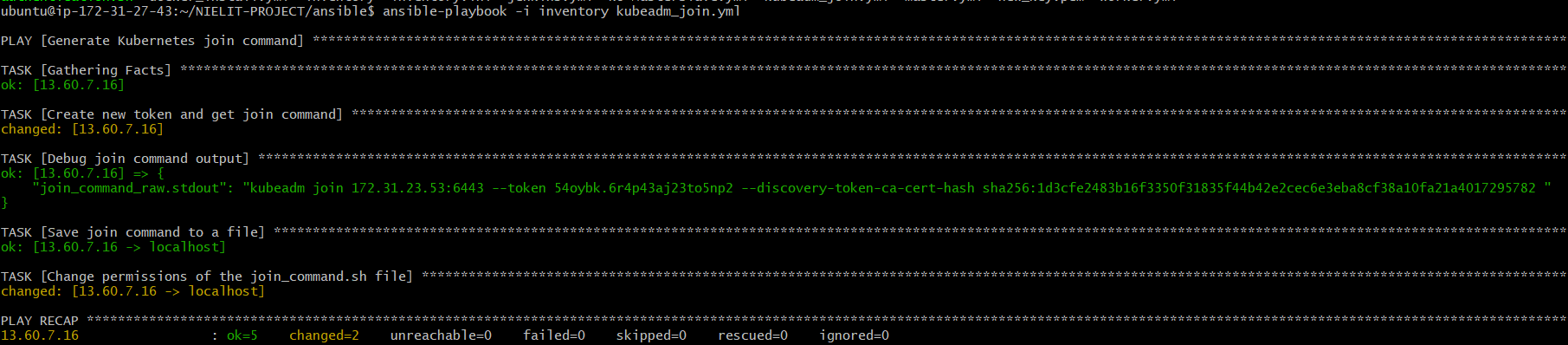
**2.Ansible:**

Using the python script create inventory and authenticate each server using   authenticate.sh (provided in github) .Run the K8-MasterSlave.yml playbook to install kubeadm,kubectl,kubelet and other commonly needed installation for master and slave nodes using the command **ansible- playbook –i inventory k8-MasterSlave.yml.**Run the master.yml to initiate the Kubernetes cluster . The playbook contains “**kubeadm init --pod-network-cidr=10.244.0.0/16 “** command.Create new token and join the other two slave nodes to control node worker nodes will be attached to control node and a cluster will be formed.Install docker on Jenkins server to run SonarQube container using ansible playbooks.

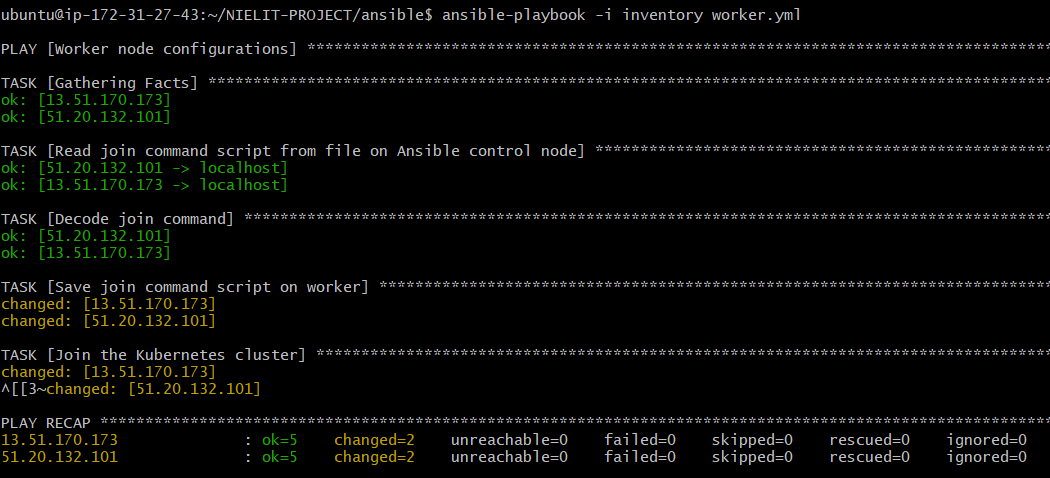
Passwordless authentication between inventory and the ansible server



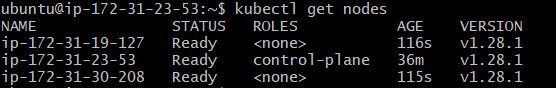
Output for generation of join token



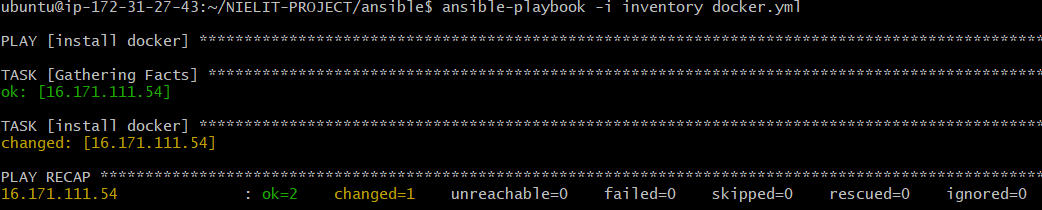
Configuring the worker nodes



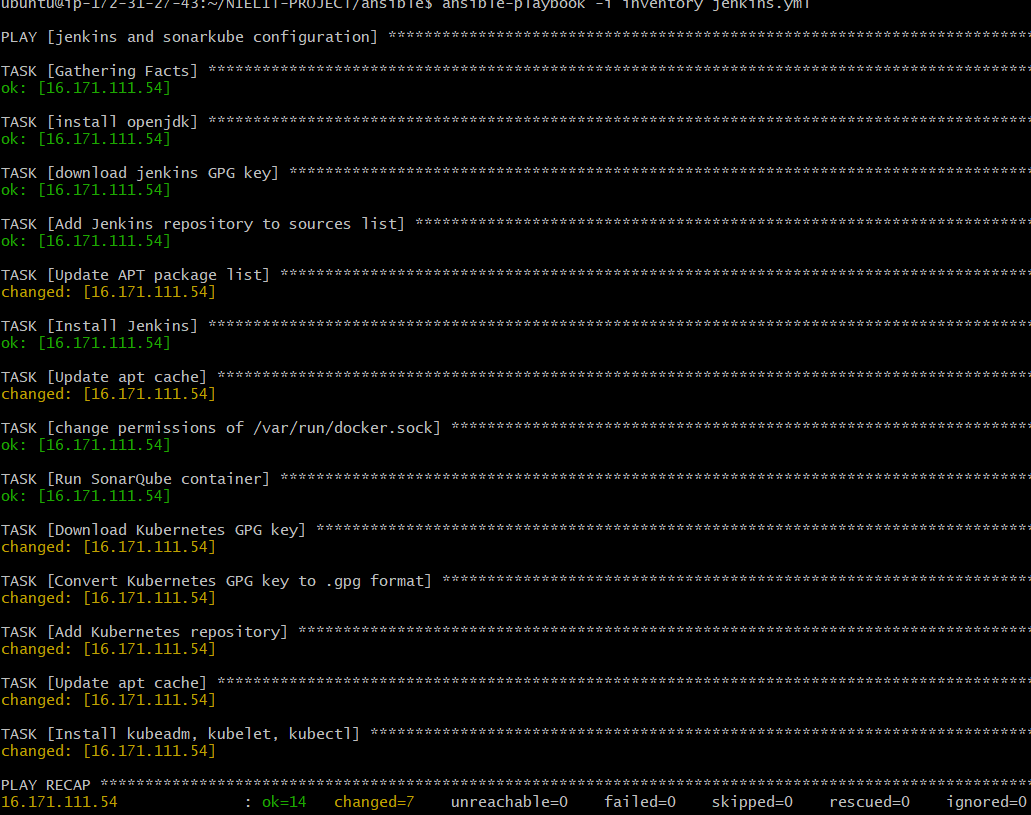
Resultant nodes in the Kubernetes cluster



Installing docker



Configuring the Jenkins, sonarqube server



**Jenkins:**

Once Jenkins is install open its IP with port 8080 then we can see the login form copy the administration password from Jenkins server and paste it here.Install the suggested plugins and required plugins (docker,docker pipeline,Kubernetes clientAPI, Kubernetes credentials,Kubernetes CLI,kubernetes,sonarqube scanner).Add docker and sonarqube scanner in Jenkins tools.Open the SonarQube server running on port 9000.Add SonarQube server to Jenkins systems configuration by generating a token in sonarqube server and adding it to global credential in Jenkins .Then create pipeline.Create a service account for kubernetes with namespace webapps and generate a token.Add the server endpoint and Kubernetes name in Jenkins script for Kubernetes deployment which are available in master node in .kube/config file.Add these stages in pipeline syntax (groovy script provided in github)

**STAGES:**

Git Checkout

SonarQube Analysis

Quality Gate

Build & Tag Docker Image

Push Docker Image

Deploy To Kubernetes

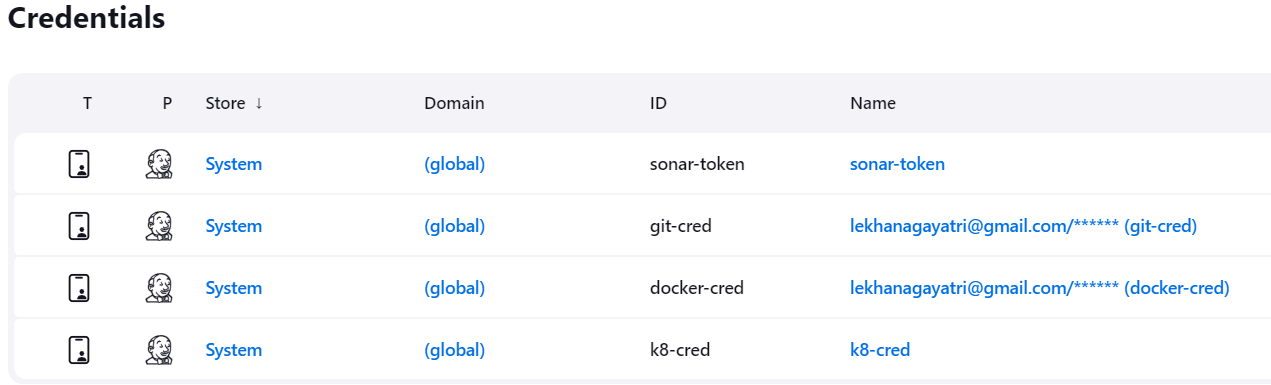
Verify the Deployment

Click Build now to start pipeline .

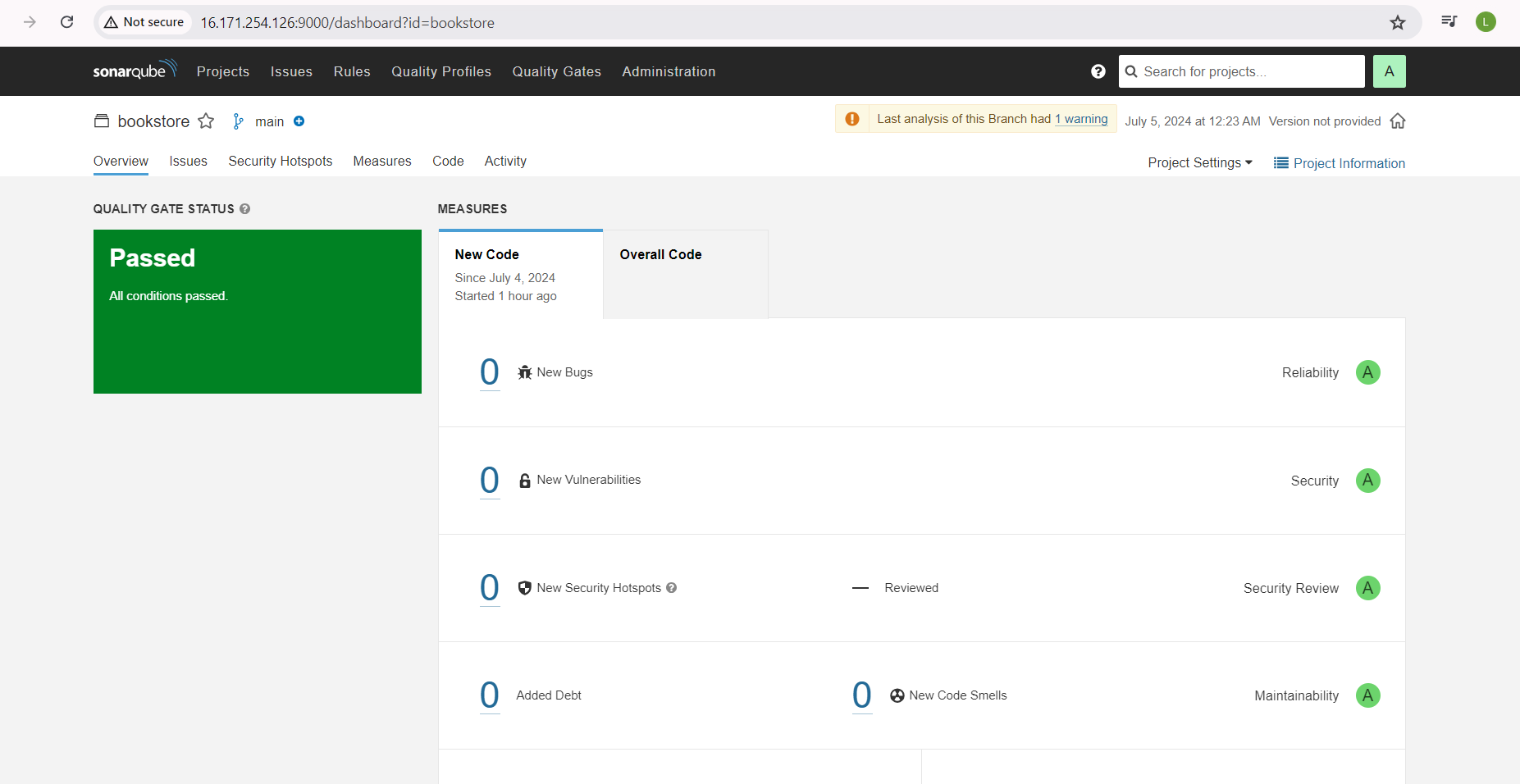
Once the pipeline execution completed, docker image will be build and pushed to dockerhub and two pods will be created in Kubernetes cluster since deployment have 2 replica sets count

IMG_256

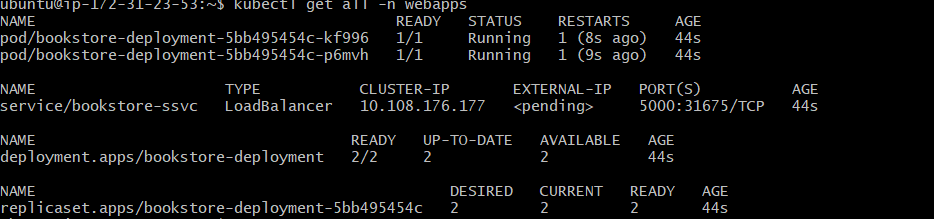
Credentials stored in Jenkins server



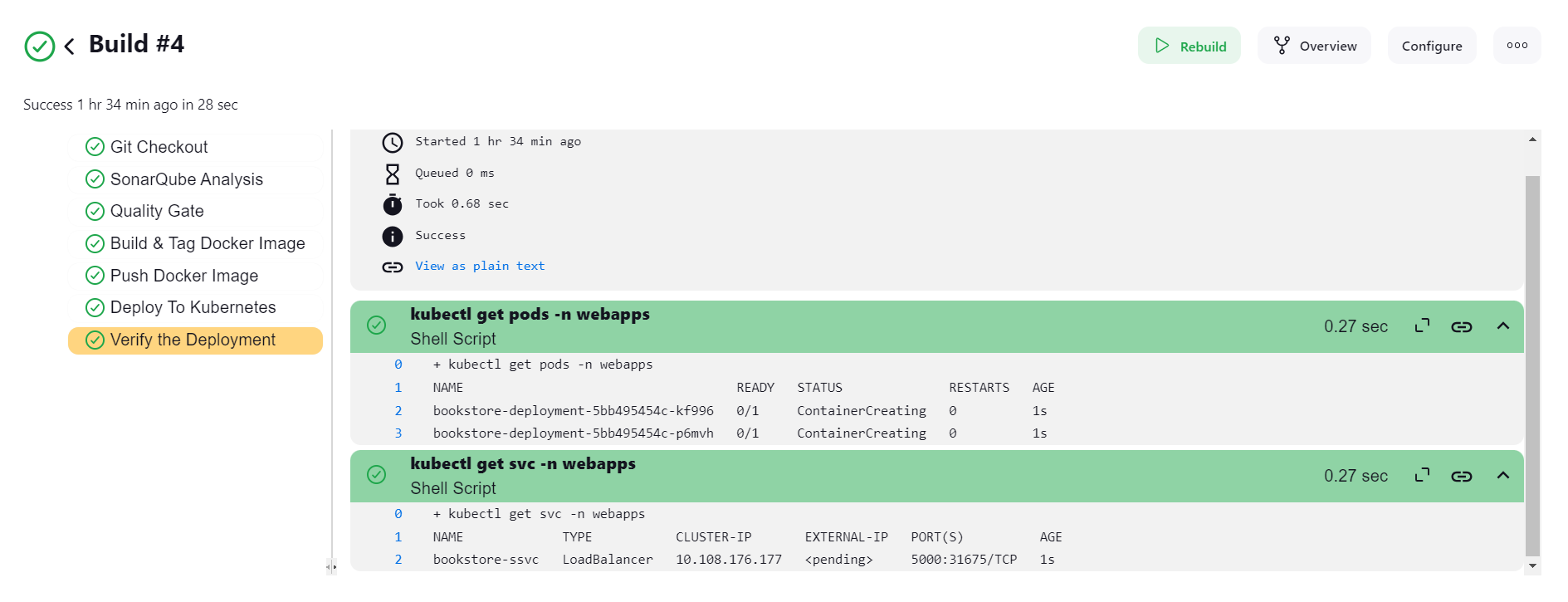
Code smells or bugs in the code after code quality analysis



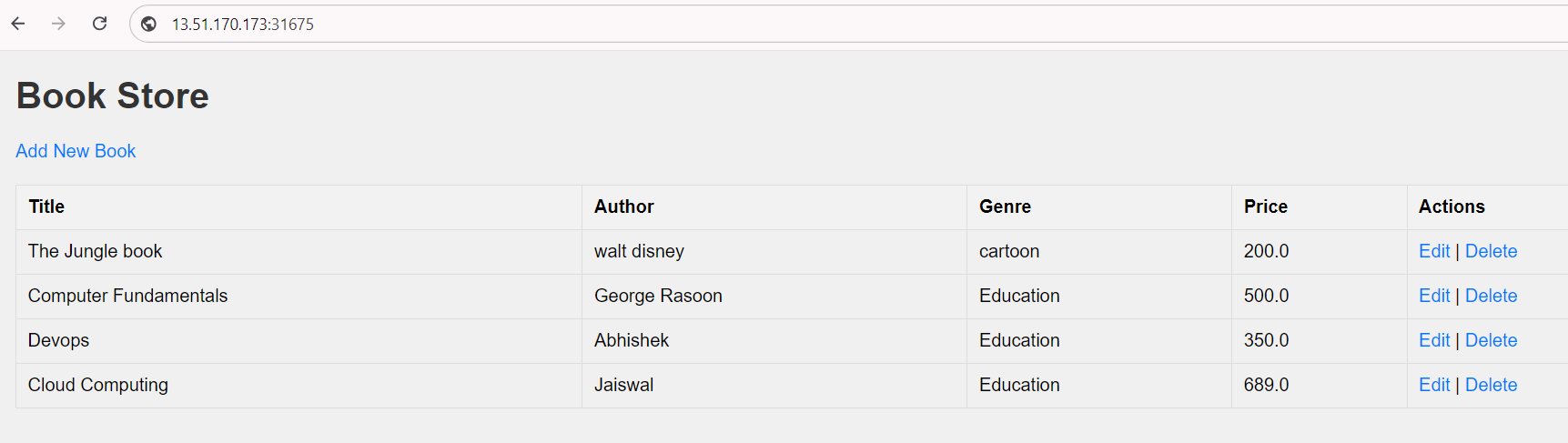
Running resources in kubernetes after jenkins pipeline execution

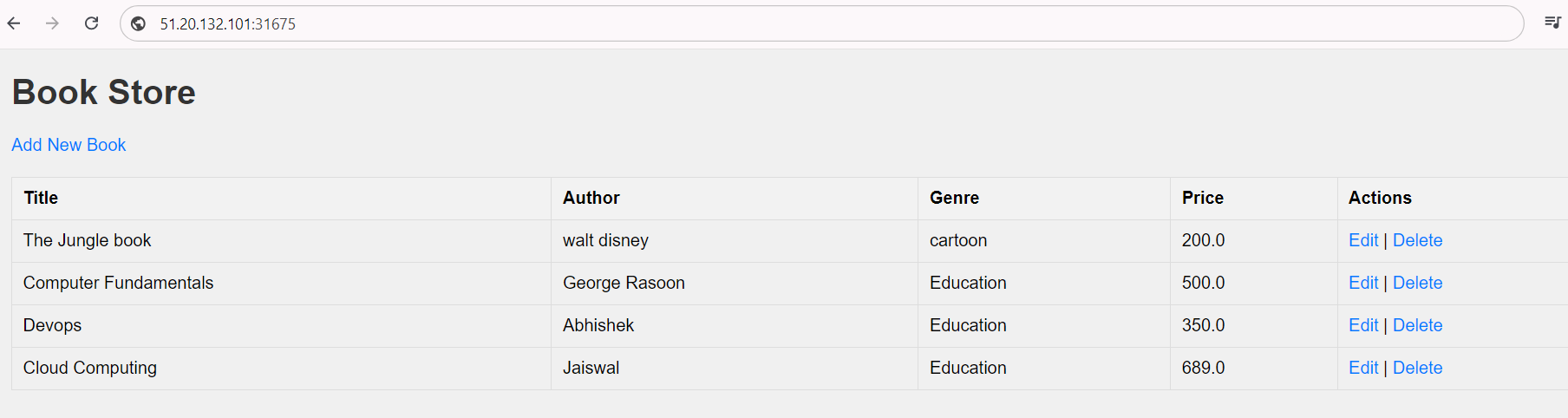


Build status in jenkins server



Application running in kubernetes cluster





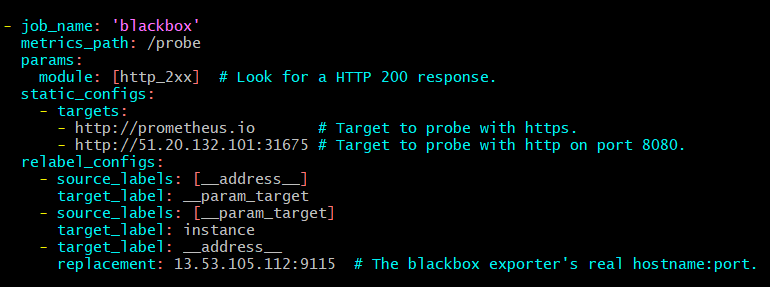
**Monitoring:**

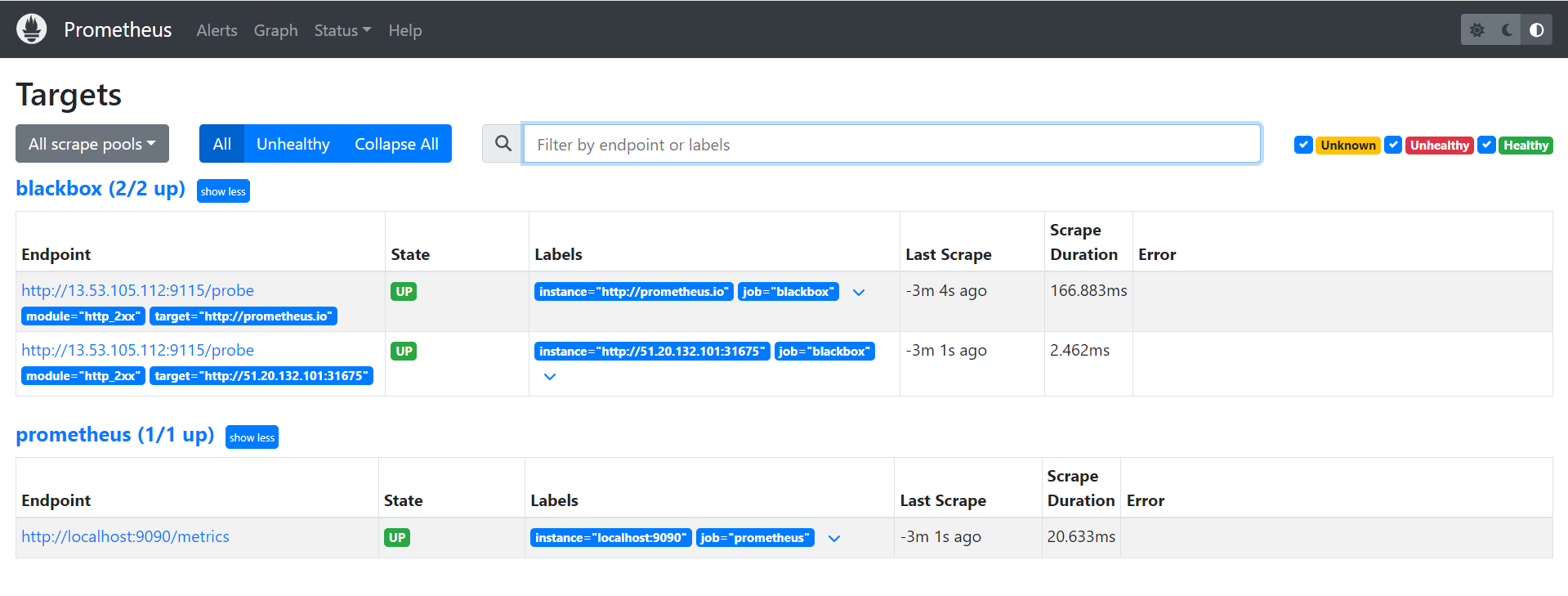
 1.Download and install Prometheus, blackbox exporter in the monitoring server and run them using shell files available. They run on ports 9090 and 9115 respectively.

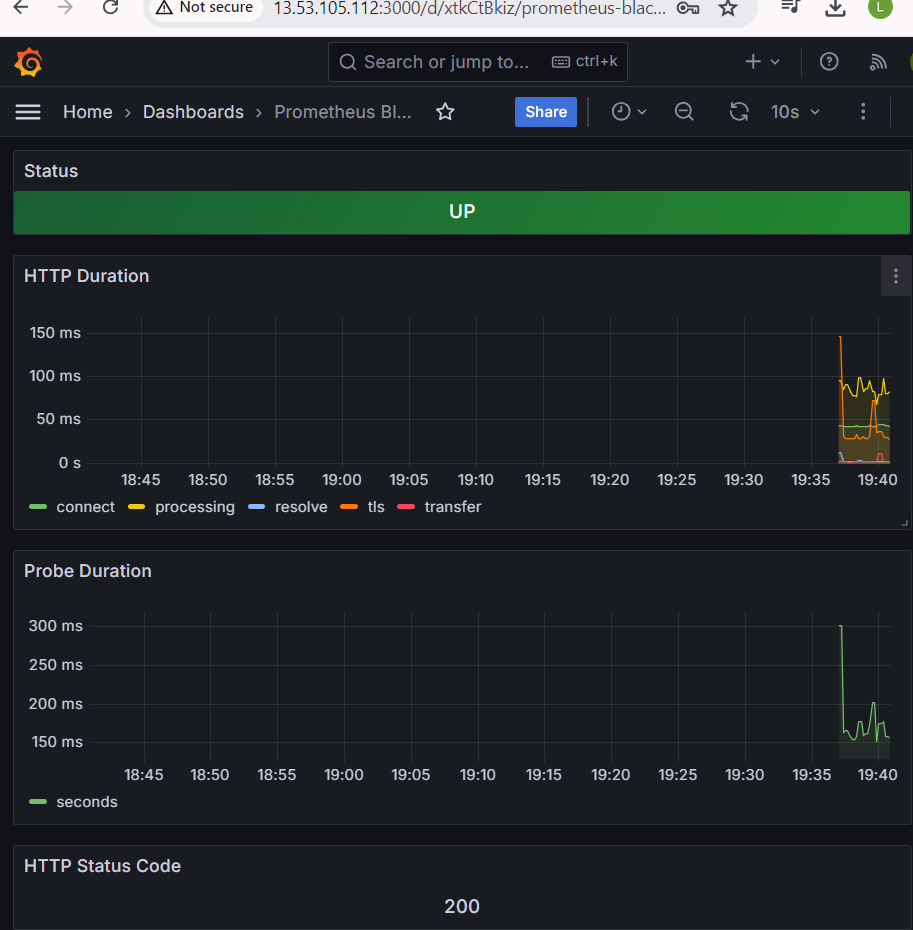
Install Graphana using the following commands

sudo apt-get install -y adduser libfontconfig1 musl  
wget <https://dl.grafana.com/enterprise/release/Grafana>enterprise\_11.1.0\_amd64.deb  
sudo dpkg -i grafana-enterprise\_11.1.0\_amd64.deb

by default it will run on port 3000.Add Prometheus as datasource to graphana to see the metrics visually and also import a Prometheus blackbox dashboard( id :7587)







**SCRIPTS:**

**Dockerfile:**

FROM python:3.8-alpine AS build

COPY requirements.txt .

RUN apk update && apk add --no-cache \

mariadb-connector-c-dev \

gcc \

musl-dev \

&& pip install --no-cache-dir -r requirements.txt

WORKDIR /app

FROM python:3.8-alpine

RUN apk update && apk add --no-cache \

mariadb-connector-c

WORKDIR /app

COPY app.py /app

COPY templates /app/templates

COPY static /app/static

COPY --from=build /usr/local/lib/python3.8/site-packages /usr/local/lib/python3.8/site-packages

COPY --from=build /usr/local/bin /usr/local/bin

CMD ["python3", "app.py"]

**JENKINS GROOVY SCRIPT:**

pipeline {

agent any

environment {

SCANNER\_HOME = tool 'sonar-scanner'

}

stages {

stage('Git Checkout') {

steps {

git branch: 'main', credentialsId: 'git-cred', url: 'https://github.com/Lekhana-Gayatri/NIELIT-PROJECT.git'

}}

stage('SonarQube Analysis') {

steps {

withSonarQubeEnv('sonar') {

sh '''

$SCANNER\_HOME/bin/sonar-scanner \

-Dsonar.projectName=FlaskApp \

-Dsonar.projectKey=FlaskApp \

-Dsonar.sources=. \

-Dsonar.language=py \

-Dsonar.python.version=3.8 \

-Dsonar.exclusions=\*\*/tests/\*\*

'''

}}}

stage('Quality Gate') {

steps {

script {

waitForQualityGate abortPipeline: false, credentialsId: 'sonar-token'

}}}

stage('Build & Tag Docker Image') {

steps {

script {

withDockerRegistry(credentialsId: 'docker-cred', toolName: 'docker') {

sh "docker build -t lekhana2004/bookstore-app:latest ."

}}}}

stage('Push Docker Image') {

steps {

script {

withDockerRegistry(credentialsId: 'docker-cred', toolName: 'docker') {

sh "docker push lekhana2004/bookstore-app:latest"

}}}}

stage('Deploy To Kubernetes') {

steps {

withKubeConfig(caCertificate: '', clusterName: 'kubernetes', contextName: '', credentialsId: 'k8-cred', namespace: 'webapps', restrictKubeConfigAccess: false, serverUrl: 'https://172.31.47.143:6443') {

sh "kubectl apply -f deployment-service.yaml"

}}}

stage('Verify the Deployment') {

steps {

withKubeConfig(caCertificate: '', clusterName: 'kubernetes', contextName: '', credentialsId: 'k8-cred', namespace: 'webapps', restrictKubeConfigAccess: false, serverUrl: 'https://172.31.47.143:6443') {

sh "kubectl get pods -n webapps"

sh "kubectl get svc -n webapps"

}

}

}

}}

**KUBERNETES MANIFESTS:**

**Service-Deployment.yaml**

apiVersion: apps/v1

kind: Deployment # Kubernetes resource kind we are creating

metadata:

name: bookstore-deployment

spec:

selector:

matchLabels:

app: bookstore

replicas: 2 # Number of replicas that will be created for this deployment

template:

metadata:

labels:

app: bookstore

spec:

containers:

- name: bookstore

image: lekhana2004/bookstore-app:latest # Image that will be used to containers in the cluster

imagePullPolicy: Always

ports:

- containerPort: 5000 # The port that the container is running on in the cluster

---

apiVersion: v1 # Kubernetes API version

kind: Service # Kubernetes resource kind we are creating

metadata: # Metadata of the resource kind we are creating

name: bookstore-ssvc

spec:

selector:

app: bookstore

ports:

- protocol: "TCP"

port: 5000

targetPort: 5000

type: LoadBalancer # type of the service.

**Service-Account.yaml:**

Creating Service Account

apiVersion: v1

kind: ServiceAccount

metadata:

name: jenkins

namespace: webapps

**Create Role**

apiVersion: rbac.authorization.k8s.io/v1

kind: Role

metadata:

name: app-role

namespace: webapps

rules:

- apiGroups:

- ""

- apps

- autoscaling

- batch

- extensions

- policy

- rbac.authorization.k8s.io

resources:

- pods

- secrets

- componentstatuses

- configmaps

- daemonsets

- deployments

- events

- endpoints

- horizontalpodautoscalers

- ingress

- jobs

- limitranges

- namespaces

- nodes

- pods

- persistentvolumes

- persistentvolumeclaims

- resourcequotas

- replicasets

- replicationcontrollers

- serviceaccounts

- services

verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]

**Bind the role to service account**

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

name: app-rolebinding

namespace: webapps

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: Role

name: app-role

subjects:

- namespace: webapps

kind: ServiceAccount

name: jenkins

**create Token**

apiVersion: v1

kind: Secret

type: kubernetes.io/service-account-token

metadata:

name: mysecretname

annotations:

kubernetes.io/service-account.name: myserviceaccount

**ANSIBLE PLAYBOOKS:**

**Master-slave.yaml:**

---

- name: Install kubedm

become: yes

hosts: master,slaves

tasks:

- name: update cache

apt:

update\_cache: yes

- name : Install docker

apt:

name: docker.io

state: present

update\_cache: yes

- name: change permissions of docker sock

ansible.builtin.file:

path: /var/run/docker.sock

mode: 666

- name: Install apt-transport-https, ca-certificates, curl, gnupg

apt:

name:

- apt-transport-https

- ca-certificates

- curl

- gnupg

state: present

update\_cache: yes

- name: Create /etc/apt/keyrings directory

file:

path: /etc/apt/keyrings

state: directory

mode: '0755'

owner: root

group: root

- name: Download Kubernetes GPG key

ansible.builtin.shell: curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key -o /tmp/kubernetes-apt-keyring.gpg

args:

creates: /tmp/kubernetes-apt-keyring.gpg

- name: Convert Kubernetes GPG key to .gpg format

ansible.builtin.shell: gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg /tmp/kubernetes-apt-keyring.gpg

args:

creates: /etc/apt/keyrings/kubernetes-apt-keyring.gpg

- name: Add Kubernetes repository

ansible.builtin.apt\_repository:

repo: "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /"

filename: "kubernetes.list"

state: present

- name: Update apt cache

ansible.builtin.apt:

update\_cache: yes

- name: Install kubeadm, kubelet, kubectl

ansible.builtin.apt:

name:

- kubeadm=1.28.1-1.1

- kubelet=1.28.1-1.1

- kubectl=1.28.1-1.1

state: present

update\_cache: yes

**kubedam\_join.yaml**

---

- name: Generate Kubernetes join command

hosts: master

become: yes

tasks:

- name: Create new token and get join command

command: kubeadm token create --print-join-command

register: join\_command\_raw

- name: Debug join command output

debug:

var: join\_command\_raw.stdout

- name: Save join command to a file

local\_action:

module: copy

content: "{{ join\_command\_raw.stdout }}"

dest: "./join\_command.sh"

changed\_when: false # To ensure this task doesn't always appear as changed

- name: Change permissions of the join\_command.sh file

local\_action:

module: ansible.builtin.command

cmd: chmod +x ./join\_command.sh

**jenkinsSonar.yaml:**

---

- name: jenkins and sonarkube configuration

hosts: jenkins

become: yes

tasks:

- name: install openjdk

apt:

name: openjdk-17-jre-headless

state: present

update\_cache: yes

- name: download jenkins GPG key

get\_url:

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

dest: /usr/share/keyrings/jenkins-keyring.asc

- name: Add Jenkins repository to sources list

ansible.builtin.lineinfile:

path: /etc/apt/sources.list.d/jenkins.list

line: 'deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/'

create: yes

- name: Update APT package list

apt:

update\_cache: yes

- name: Install Jenkins

apt:

name: jenkins

state: present

- name: Update apt cache

ansible.builtin.apt:

update\_cache: yes

- name: change permissions of /var/run/docker.sock

ansible.builtin.file:

path: /var/run/docker.sock

mode: 666

- name: Run SonarQube container

docker\_container:

name: sonar

image: sonarqube:lts-community

state: started

ports:

- "9000:9000"

- name: Download Kubernetes GPG key and save it

ansible.builtin.command:

cmd: curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

args:

creates: /etc/apt/keyrings/kubernetes-apt-keyring.gpg

- name: Add Kubernetes repository

ansible.builtin.apt\_repository:

repo: "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /"

filename: "kubernetes.list"

state: present

- name: Update apt cache

ansible.builtin.apt:

update\_cache: yes

- name: Install kubeadm, kubelet, kubectl

ansible.builtin.apt:

name: kubectl=1.28.1-1.1

state: present

update\_cache: yes

**REFEREANCES**: <https://github.com/Lekhana-Gayatri/NIELIT-PROJECT/>

**CONCLUSION:**

In conclusion, this project demonstrates a robust and scalable infrastructure using Terraform, Ansible, and Kubernetes on AWS EC2, with an efficient CI/CD pipeline managed by Jenkins. Terraform ensures consistent and repeatable environments, while Ansible automates configuration management. Jenkins and SonarQube enhance code quality and streamline deployment, reducing manual errors. Prometheus and Grafana provide powerful monitoring and visualization, ensuring application health and performance. Deploying a Flask-based CRUD bookstore application highlights the practical application of this setup, enhancing operational efficiency and fostering a strong DevOps culture.