sudo rm -rf /var/lib/jenkins/workspace/* except cluster

AWSCLI

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
sudo apt install unzip
unzip awscliv2.zip
sudo ./aws/install
aws configure
```

KUBECTL

```
curl -o kubectl
https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux
/amd64/kubectl
chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin
kubectl version --short --client
```

EKSCTL

```
curl --silent --location
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(una
me -s)_amd64.tar.gz" | tar xz -C /tmp
sudo mv /tmp/eksctl /usr/local/bin
eksctl version
```

……………………………………………………………………………………………
……………………………………………………………………………………………
…………………………….

Microservices project :

https://github.com/Sravyatirumala/Microservice

Create Ec2 instance, jenkins, Kube cluster.

In jenkins plugin : Multibranch Scan Webhook trigger plugin.

……

Create Pipeline : Multibranch Pipeline —> Git URL —> Build Configuration : Jenkinsfile —>
 Scan Multibranch Pipeline : Scan by Webhook : Sravya ? Apply save.

http://3.138.66.92:8080/multibranch-webhook-trigger/invoke?token

JENKINS_URL/multibranch-webhook-trigger/invoke?token=

Copy and go to setting in git : web hooks : Payload URL
http://3.138.66.92:8080/multibranch-webhook-trigger/invoke?token .   Add web hook.

……

This will run the pipeline and it willl be successful.

……
For CD process.

```
 kubectl   create namespace webapps
```

```
Vi service.yml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: jenkins
  namespace: webapps
```

```
kubectl apply -f service.yml
```

……

```
Vi role.yml
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
        - 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"]
```

```
kubectl apply -f role.yml
```

……
……………………………………………………………………………………………………….
…………………………….

```
Vi bindrole.yml
```

```
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
```

```
kubectl apply -f bindrole.yml
```

………………………………………………………………………………………
……………………………………………………………………………………
…………………………….
Generate token :

```
Vi   secret.yml
```

```
apiVersion: v1
kind: Secret
type: kubernetes.io/service-account-token
metadata:
  name: mysecretname
  annotations:
    kubernetes.io/service-account.name: jenkins.   (Jenkins name Given in
bindrole name)
```

```
kubectl apply -f secret.yml -n webapps
```

………………………………………………………………………………………
……………………………………………………………………………………
…………………………….

```
kubectl get all -n webapps
```

```
kubectl describe secret mysecretname -n webapps.   // To get secret .
```

token:
eyJhbGciOiJSUzI1NiIsImtpZCI6Ik5WUE9VU0ttd01kbnpaTnhZX1B2YzVRYzF
mTFJBZl9nRjRjcVdzdzJqbG8ifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2Vh
Y2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY
2UiOiJ3ZWJhcHBzIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZW
NyZXQubmFtZSI6Im15c2VjcmV0bmFtZSIsImt1YmVybmV0ZXMuaW8vc2Vyd
mljZWFjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJqZW5raW5zIiwia3
ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQud
WlkIjoiMGY0MDAxMGEtNzEyNy00OGM0LTlmOTQtMDkzNmUwZjE3ZGU3Ii

wic3ViIjoic3lzdGVtOnNlcnZpY2VhY2NvdW50OndlYmFwcHM6amVua2lucyJ9.
ZokZ-Ua23Y5PoY17ohSIvjf_DvR3mUOzJlpyzQPL9-5Xbeqo3QLKCbxENtPZ
D5VIPpIA5McLXxpepUFHHma0_aKvSY2yC5q28atroOZSWhk_UmmiR7XFo8
L5R4YrvuI1JA1k6Hn4HsjU3_p7AloZuoNaMNSzAqyJ99k-cqAIXSyxwudUQFD
RkeVI3T3N4AGbRzlzS5uIrS5MrgWMmciIUNO2Jlq_YGnY1LI03Lmb-yBLuBt3
srxMyk_wSG3lpvsobBX3iqlaYEpXZTT6zVi-WAFXAC7fuI74yG81-CWx31vcbl
GO9l6AnAWzGDg1ShYKgJJbybBRw70Dqjbu3C1zYg

Keep this token in jenkins —> Credentials : k8-token
And give cluster end -point URL in Jenkinfile.

……………………………………………………………………………………
……………………………………………………………………………………
……………………………

Run the project .

```
kubectl get all -n webapps
```

To delete from local after pushing :

```
        post {
          always {
               // Cleanup actions, such as removing temporary files or notifying
users
               echo 'Pipeline finished, cleaning up workspace.'
               cleanWs()   // Optional: you can clean the workspace again if
necessary
          }
          success {
               echo 'Build and Push completed successfully!'
          }
          failure {
               echo 'Build or Push failed, please check the logs!'
          }
```

Dockerfile

```
FROM
python:3.11.1-slim@sha256:1591aa8c01b5b37ab31dbe5662c5bdcf40c2f1bce
4ef1c1fd24802dae3d01052 as base
```

```
FROM base as builder

COPY requirements.txt .

RUN pip install --prefix="/install" -r requirements.txt

FROM base

WORKDIR /loadgen

COPY --from=builder /install /usr/local

# Add application code.
COPY locustfile.py .

# enable gevent support in debugger
ENV GEVENT_SUPPORT=True

ENTRYPOINT locust --host="http://${FRONTEND_ADDR}" --headless -u
"${USERS:-10}" 2>&1
```

Jenkinsfile

```
pipeline {
    agent any

    environment {
        SCANNER_HOME = tool 'sonar-scanner'
    }

    stages {
        stage('Clean Workspace') {
            steps {
                cleanWs()
            }
        }

        stage('SonarQube Analysis') {
            steps {
                withSonarQubeEnv('sonar-server') {
                    sh '''
                    $SCANNER_HOME/bin/sonar-scanner \
                        -Dsonar.projectName=Microservice \
                        -Dsonar.projectKey=Microservice
                    '''
                }
            }
        }

        stage('Quality Gate') {
            steps {
                script {
                    waitForQualityGate abortPipeline: false, credentialsId:
'Sonar-token'
                }
            }
        }

        stage('Trivy FS Scan') {
            steps {
                sh 'trivy fs . > trivyfs.txt'
            }
        }

        stage('Build & Tag Docker Image') {
            steps {
                script {
                    withDockerRegistry(credentialsId: 'Docker-creds',
toolName: 'docker') {
                        sh "docker build --no-cache -t
sravyatirumala/loadgenerator:latest ."
                    }
                }
            }
        }

        stage('Push Docker Image') {
            steps {
                script {
                    withDockerRegistry(credentialsId: 'Docker-creds',
toolName: 'docker') {
                        sh "docker push sravyatirumala/loadgenerator:latest"
                    }
                }
            }
        }
    }

    post {
        always {
            // Cleanup actions, such as removing temporary files or notifying
```

```
users
                echo 'Pipeline finished, cleaning up workspace.'
                cleanWs()   // Optional: you can clean the workspace again if
necessary
            }
            success {
                echo 'Build and Push completed successfully!'
            }
            failure {
                echo 'Build or Push failed, please check the logs!'
            }
        }
}
```

```
pipeline {
    agent any

    environment {
        SCANNER_HOME = tool 'sonar-scanner'
    }

    stages {
        stage('Clean Workspace') {
            steps {
                cleanWs()
            }
        }

        stage('Checkout from Git') {
            steps {
                git branch: 'main', url:
'https://github.com/Sravyatirumala/Microservice.git'
                sh 'ls -la'   // Verify files after checkout
            }
        }

        stage('SonarQube Analysis') {
            steps {
                withSonarQubeEnv('sonar-server') {
                    sh '''
                    $SCANNER_HOME/bin/sonar-scanner
-Dsonar.projectName=Microservice \
                    -Dsonar.projectKey=Microservice
                    '''
                }
            }
        }

        stage('Quality Gate') {
            steps {
                script {
                    waitForQualityGate abortPipeline: false, credentialsId:
'Sonar-token'
                }
            }
        }

        stage('Build & Tag Docker Image') {
            steps {
                script {
                    // Check for Dockerfile existence before building
                    def dockerFileExists = fileExists('adservice/Dockerfile')
                    if (dockerFileExists) {
                        withDockerRegistry(credentialsId: 'docker', toolName:
'docker') {
                            sh '''
                            echo "Building Docker image..."
                            docker build --no-cache -t
sravyatirumala/adservice:latest ."
                            '''
                        }
                    } else {
                        error "Dockerfile not found in the workspace!"
                    }
                }
            }
        }

        stage('Push Docker Image') {
            steps {
                script {
                    withDockerRegistry(credentialsId: 'Docker-creds',
toolName: 'docker') {
                        sh "docker push sravyatirumala/adservice:latest"
                    }
                }
            }
        }
    }
    post {
        always {
            // Cleanup actions, such as removing temporary files or notifying
users
            echo 'Pipeline finished, cleaning up workspace.'
            cleanWs()   // Optional: you can clean the workspace again if
necessary
        }
        success {
            echo 'Build and Push completed successfully!'
        }
        failure {
            echo 'Build or Push failed, please check the logs!'
        }
    }
}
```