**DevOps Tasks 3**

* Save the below python “3.7” script into a “.py” file then create a **Dockerfile** that will create a docker image for it which will run the python script into **CMD** command.
* Create a **yaml** file which will be used to deploy that docker image onto a Kubernetes cluster any cluster a minikube will be good.
* Create a service file for it which should expose the port into the python script.
* Create a horizontal pod autoscaler file for it.
* Push all files to a github repository.
* Write a **Jenkins pipeline script** to:
  + Clone the repository you’ve created.
  + Build the docker image and give it a build number tag.
  + Push it on any docker registry.
  + Deploy it on a Kubernetes cluster.

#!/usr/bin/env python

import socket

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def gethostname():

s = 'Container ID is: ' + socket.gethostname() + "This is the dev branch."

return s

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=8080

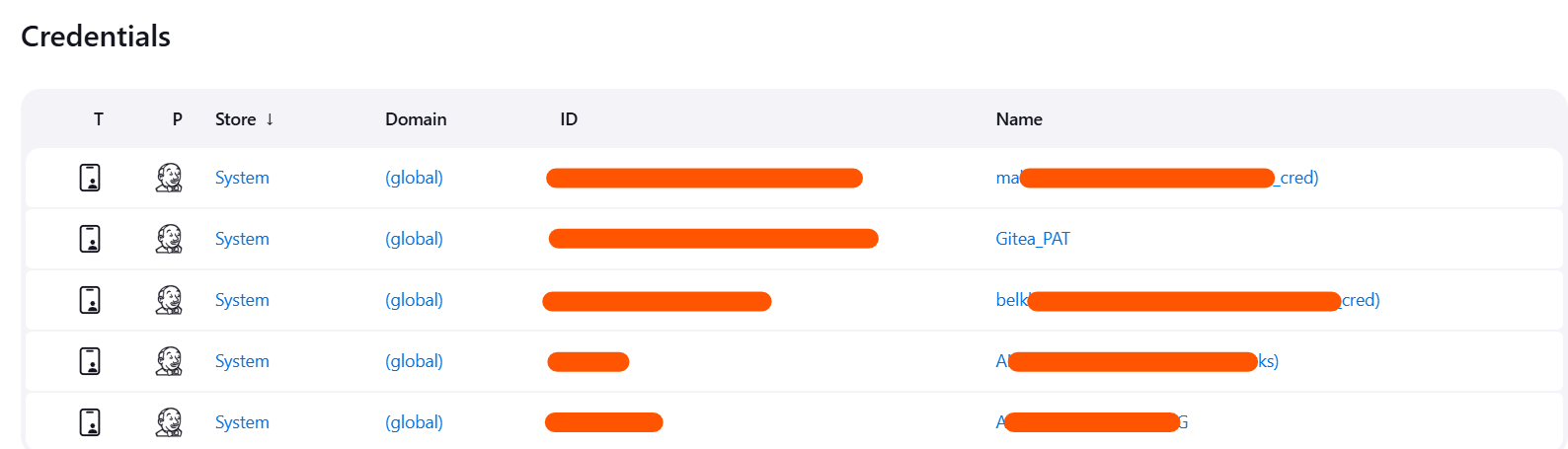


**Pre-requisite:**

1. Running Jenkins server with Git plugin and credentials of Github, DockerHub, and AWS access key and secret key must be created in Jenkins credentials.
2. Running AWS EKS cluster on AWS.

**STEPS:**

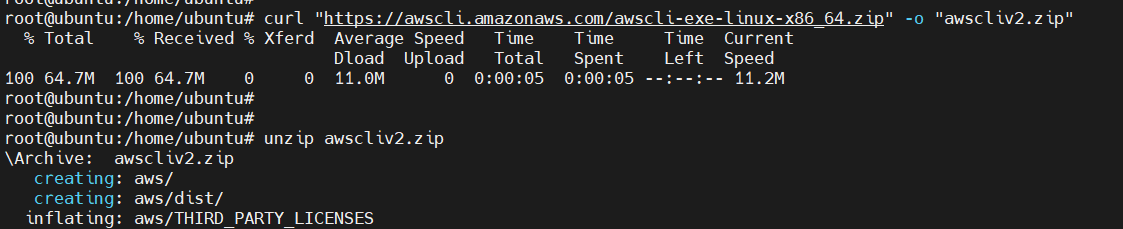
**STEP 1: Create all credentials on Jenkins server as follows, with Global access.**



**STEP 2: Install AWS-cli**,

# curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

# unzip awscliv2.zip

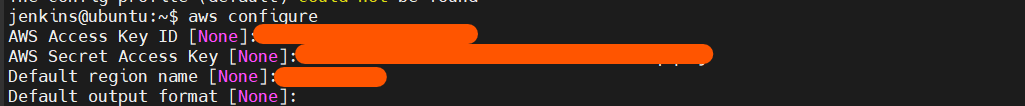


# sudo ./aws/install

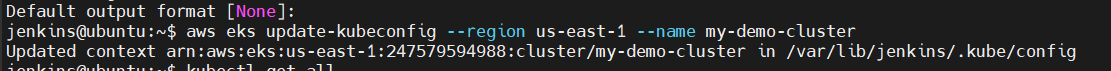
# aws –version



$ aws configure



$ aws eks update-kubeconfig --region us-east-1 --name my-demo-cluster



**STEP 3: write Dockerfile, app.py, deployment.yaml, hpa.yaml, service.yaml as below,**

* Dockerfile

FROM python:3.7

WORKDIR /app

COPY app.py /app/

RUN pip install flask

EXPOSE 8080

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

* app.py

#!/usr/bin/env python

import socket

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def gethostname():

return f'Container ID is: {socket.gethostname()} - This is the dev branch.'

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=8080)

* deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: python-app

spec:

selector:

matchLabels:

app: python-app

template:

metadata:

labels:

app: python-app

spec:

containers:

- name: python-app

image: my-dockerhub-username/my-python-app:BUILD\_NUMBER

ports:

- containerPort: 8080

* hpa.yaml

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: python-app-hpa

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: python-app

minReplicas: 2

maxReplicas: 3

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

averageUtilization: 50

* service.yaml

apiVersion: v1

kind: Service

metadata:

name: python-service

spec:

selector:

app: python-app

ports:

- protocol: TCP

port: 80

targetPort: 8080

type: LoadBalancer

* Create using nano editor and Upload all files on Github repository to get in Jenkinsfile pipeline using below command.



$ git add .

$ git commit -m "all files"

$ git remote add origin https://github.com/maheshbelkhude/Devops\_test.git

$ git branch -M main

$ git push -u origin main

**STEP 4: Create and Write a Jenkinsfile as below,**

pipeline {

agent any

environment {

AWS\_DEFAULT\_REGION = 'us-east-1'

CLUSTER\_NAME = 'my-demo-cluster'

KUBE\_CONFIG\_PATH = '/var/lib/jenkins/.kube/config'

}

stages {

stage('Checkout') {

steps {

checkout([

$class: 'GitSCM',

branches: [[name: '\*/main']],

userRemoteConfigs: [[

url: 'https://github.com/maheshbelkhude/Devops\_test.git',

credentialsId: '6acf10f7-97fd-4a0f-a27f-5d6c3d6f8f55'

]]

])

}

}

stage('Configure AWS & Kubectl') {

steps {

withCredentials([[$class: 'AmazonWebServicesCredentialsBinding', credentialsId: 'AWS-EKS-01']]) {

sh '''

aws eks update-kubeconfig --region $AWS\_DEFAULT\_REGION --name $CLUSTER\_NAME

'''

}

}

}

stage('Build Docker Image') {

steps {

sh "docker build -t belkhudemahesh/python\_app:${BUILD\_NUMBER} ."

}

}

stage('Push to Docker Hub') {

steps {

withDockerRegistry([credentialsId: 'dockerhub\_mahesh\_cred', url: 'https://index.docker.io/v1/']) {

sh "docker push belkhudemahesh/python\_app:${BUILD\_NUMBER}"

}

}

}

stage('Update Deployment YAML for Image') {

steps {

sh """

sed -i 's|my-dockerhub-username/my-python-app:BUILD\_NUMBER|belkhudemahesh/python\_app:${BUILD\_NUMBER}|' deployment.yaml

"""

}

}

stage('Deploy to EKS') {

steps {

sh """

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

kubectl apply -f hpa.yaml

"""

}

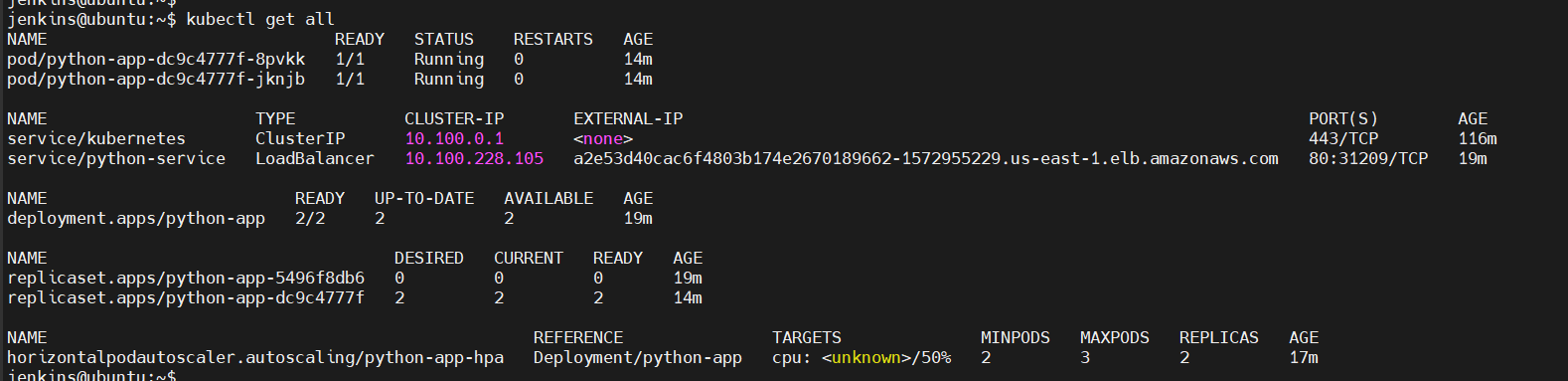
}

}

}

$ kubectl get all

Next, Run this pipeline using “Build NOW” option. After running successfully we will get deployment output as below screenshot,



$ kubectl exec -it python-app-dc9c4777f-8pvkk -- /bin/bash



:~# netstat -antp | grep LISTEN

