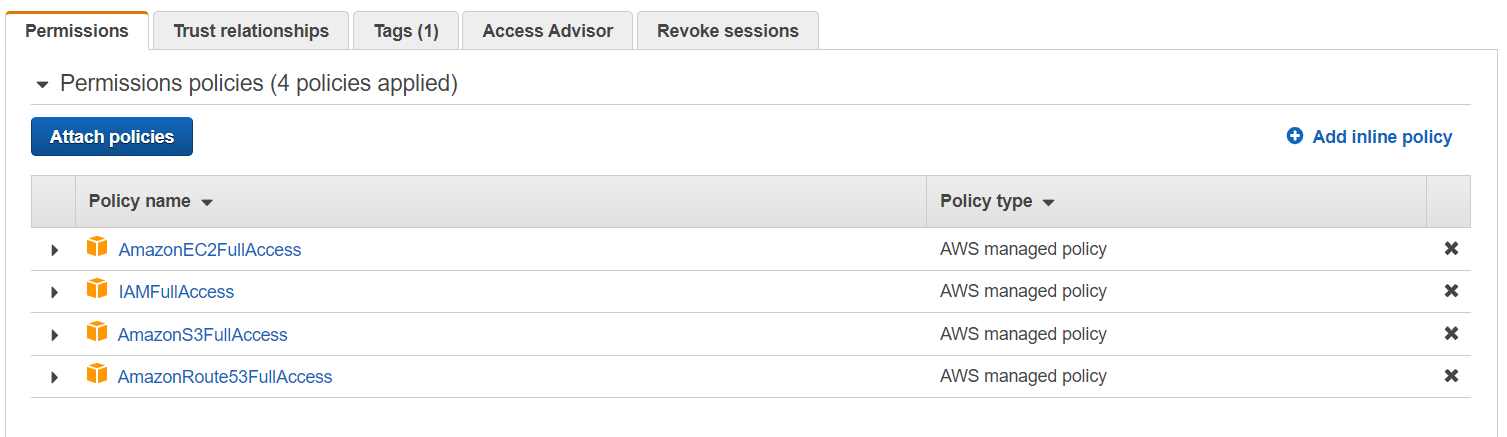


Once the new code is checked in to git repository , Jenkins will pull the code and create build using maven , then Ansible will run the play books to create docker image and push to docker hub, and deploy the image in kubernetes and expose the service.

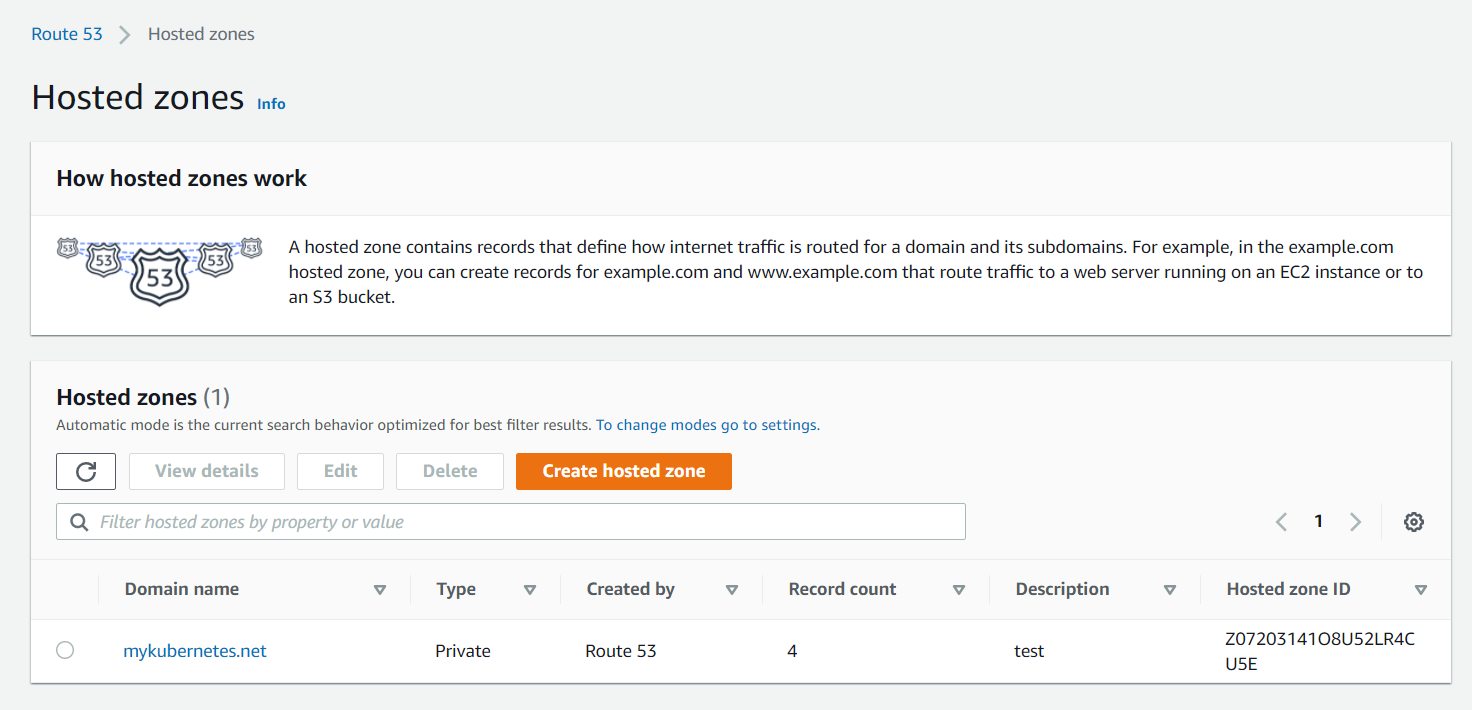
Pre-requisite:-

1. AWS account
2. Create a security group with custom TCP Rule which will input on port no. 8080
3. Jenkins server setup on linux o/s
4. Install Jenkins
5. Install Git
6. Install Maven
7. Install plugins (Publish over SSH, and Configure Maven, Git, Ansible, Docker
8. Docker on linux o/s
9. Create DockerHub account to push the created image.
10. Ansible (configure Ansible with Docker server, Kubernetes Master node) on linux o/s
11. Kubernetes on ubuntu o/s

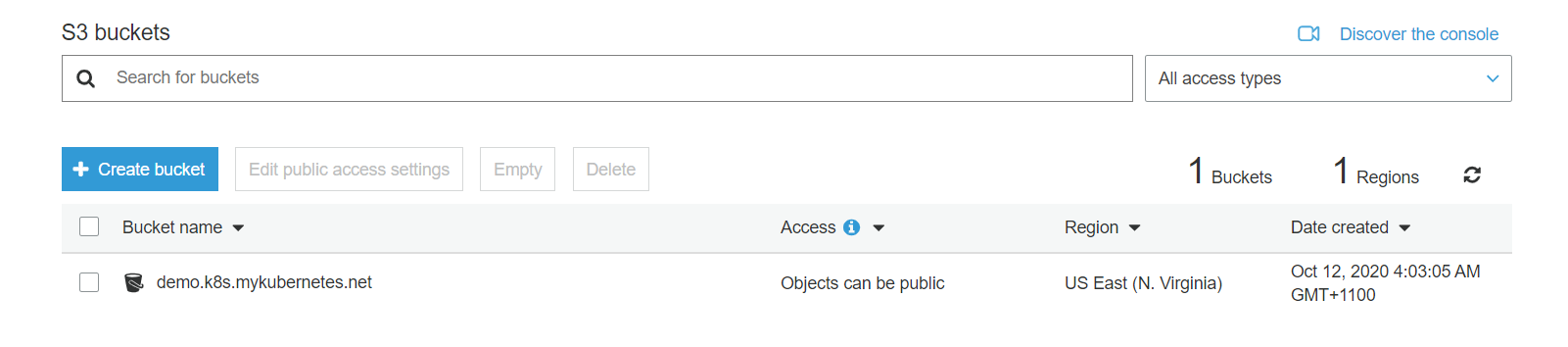
1. Create Iam user role with Route53, EC2, IAM and s3 Full Access



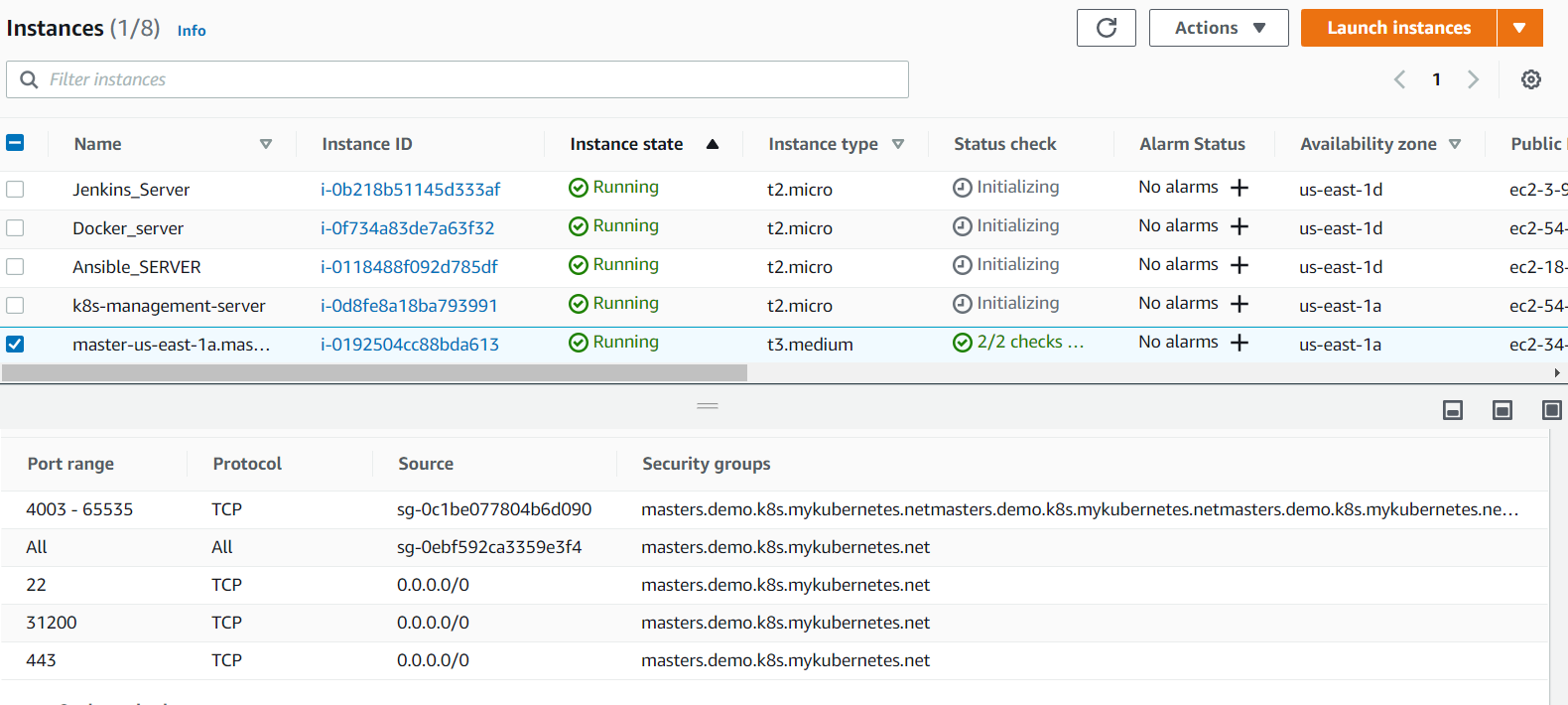
1. Route53



1. S3 bucket



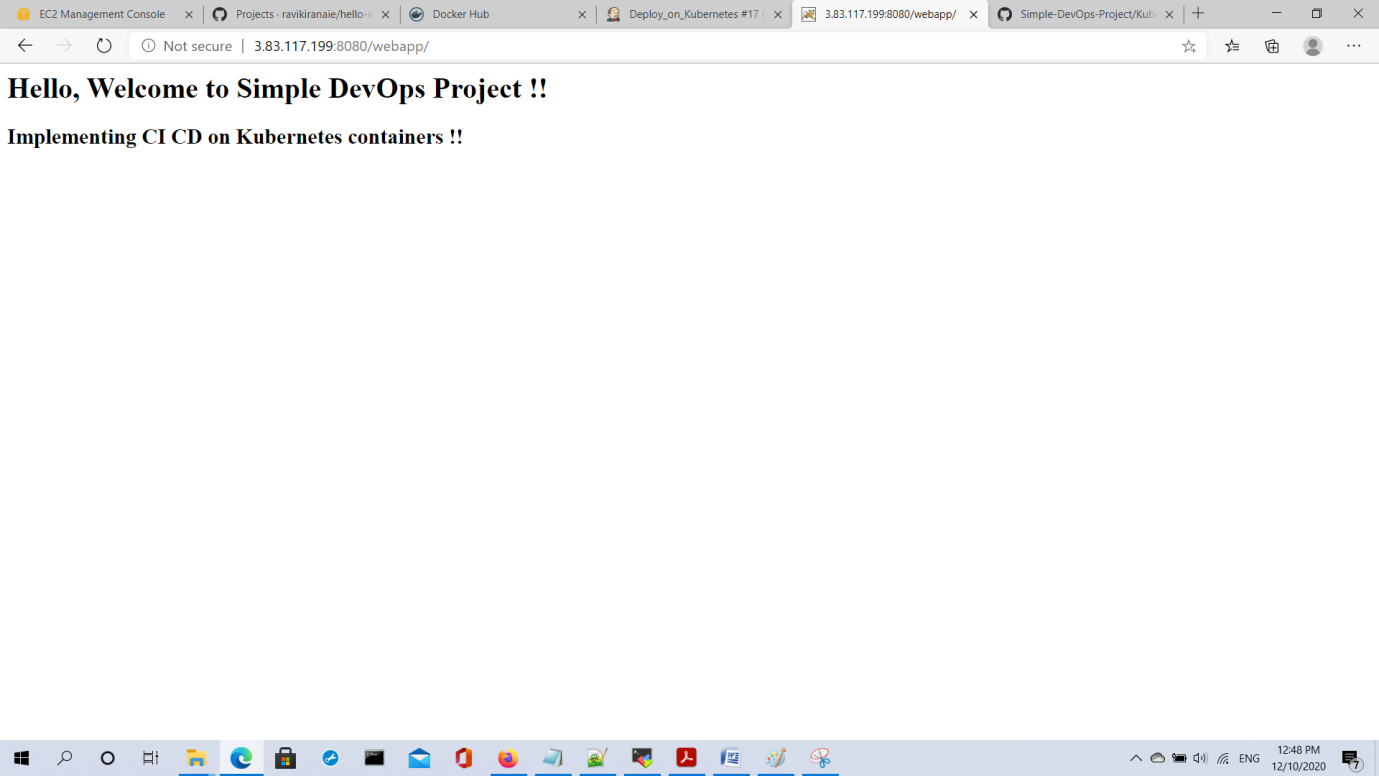
1. Instances and open inbound port 31200 for application access



1. Accessing deploymenton kubernetes masterip and port 31200

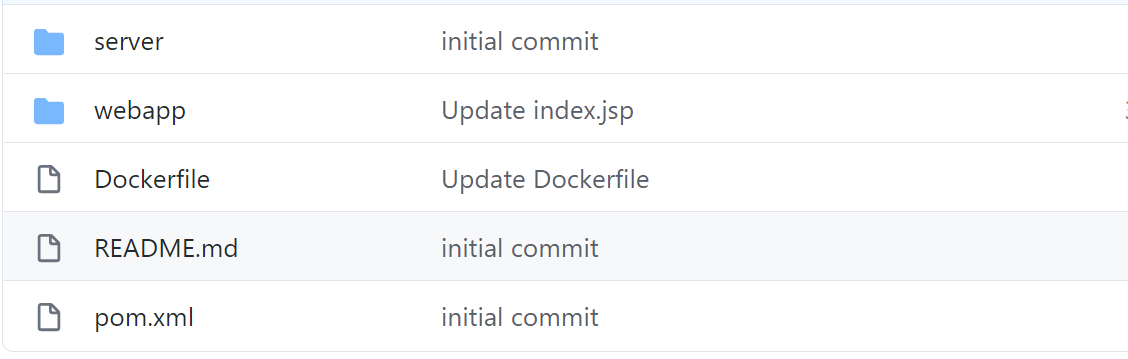


1. Accessing deployment on kubernetes master ip and port 8080

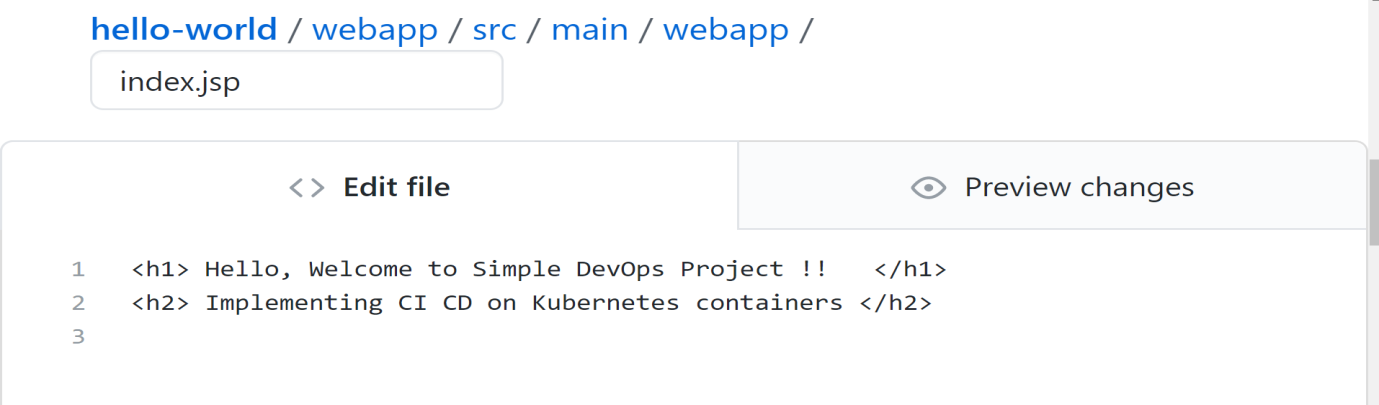


github

1. Folder structure

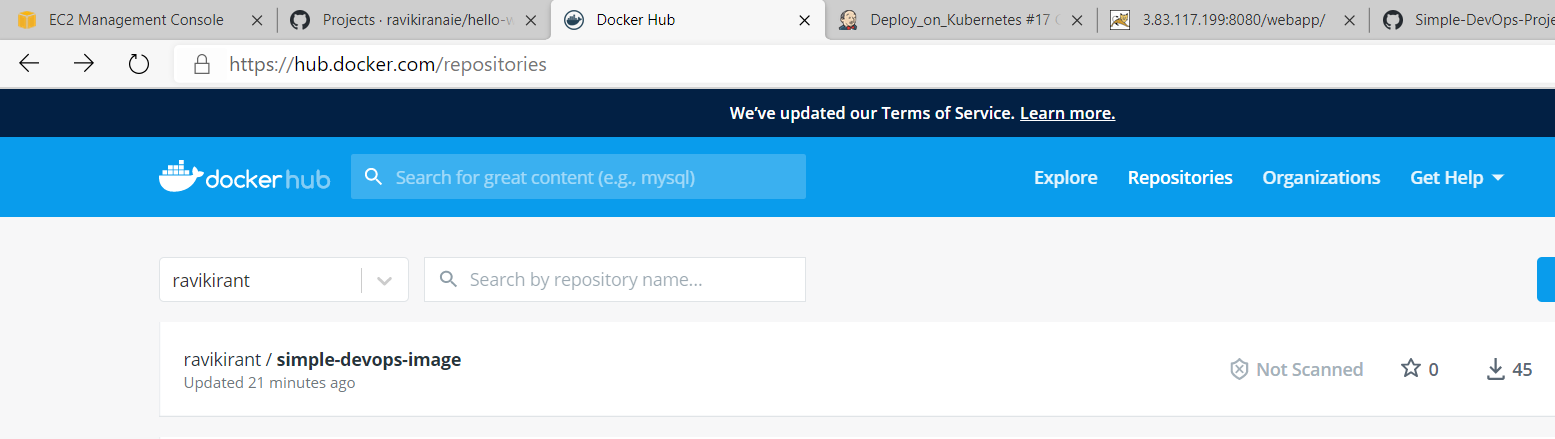


1. hello-world web-app - update



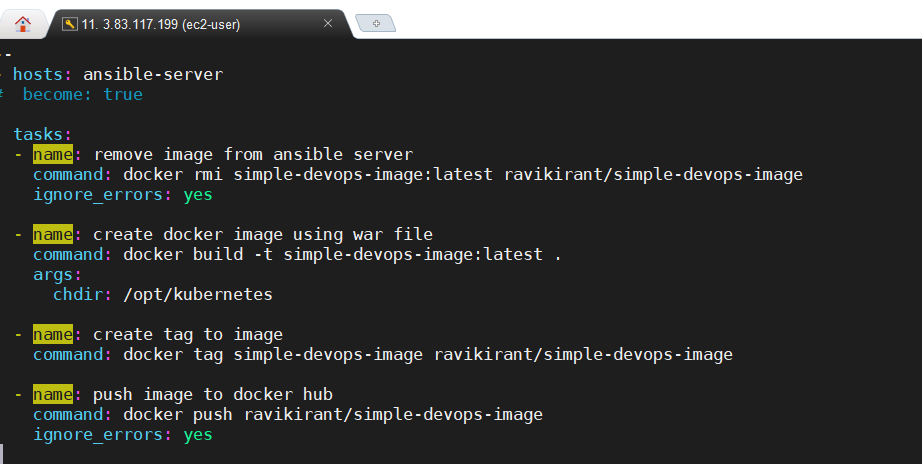
1. accessing the kubernetes using master ip and port no

DockerHub – storing images



Ansible Playbooks

Playbook 1 – createimage.yaml



---

- hosts: ansible-server

# become: true

tasks:

- name: remove image from ansible server

command: docker rmi simple-devops-image:latest ravikirant/simple-devops-image

ignore\_errors: yes

- name: create docker image using war file

command: docker build -t simple-devops-image:latest .

args:

chdir: /opt/kubernetes

- name: create tag to image

command: docker tag simple-devops-image ravikirant/simple-devops-image

- name: push image to docker hub

command: docker push ravikirant/simple-devops-image

ignore\_errors: yes

Playbook2-mydeployment.yaml

---

- name: Create pods using deployment

hosts: kubernetes

become: true

user: ubuntu

tasks:

- name: create a deployment

command: kubectl apply -f my-deploy.yaml

- name: update deployment with new pods if image updated in docker hub

command: kubectl rollout restart deployment.v1.apps/my-deployment

Playbook3-myservice.yaml

---

- name: create service for deployment

hosts: kubernetes

# become: true

user: ubuntu

tasks:

- name: create a service

command: kubectl apply -f my-service.yaml

Kubernetes deployment and service yaml

my-deploy.yaml

apiVersion: apps/v1 # deployment api version

kind: Deployment

metadata:

name: my-deployment # deployment name

spec:

selector:

matchLabels:

app: my-cicd-project # label

replicas: 2 # will create deployment to run 2 pods matching the template

strategy:

type: RollingUpdate # deployment type

rollingUpdate:

maxSurge: 1

maxUnavailable: 1

template:

metadata:

labels:

app: my-cicd-project

spec:

containers:

- name: my-cicd-project # container name

image: ravikirant/simple-devops-image # image

imagePullPolicy: Always

ports:

- containerPort: 8080

my-service.yaml

apiVersion: v1 # apiversion type of service

kind: Service

metadata:

name: my-service # service name

labels:

app: my-cicd-project # deployment lable

spec:

selector:

app: my-cicd-project

type: LoadBalancer # exposing the service on nodeport 31200

ports:

- port: 8080

targetPort: 8080

nodePort: 31200

Docker File:-

FROM tomcat:8.0

MAINTAINER RAVI KIRAN

COPY ./webapp.war /usr/local/tomcat/webapps