**Final-Task:**

**Using ansible to deploy sample nginx/python application into kubernetes cluster.**

**- Build docker image for sample nginx/python and push to aws elastic container registry(ECR) using docker cli.**

**- Create ansible role to deploy nginx application into kubernetes cluster.**

**- Create ansible playbooks to deploy nginx applications.**

**- Container image should be pulled from aws elastic container registry(ECR).**

**- Outcome is an application deployed and run in k8s cluster. Able to access nginx website with url http://<ip\_address>:<port>.**

**Pre-requisites:**

1. **VIRTUAL MACHINE**
2. **DOCKER**
3. **ANSIBLE**
4. **KUBERNETES CLUSTER**
5. **AWS CLI INSTALLATION**

**SOLUTION**:

1. Created an Ubuntu 20.04 ec2-instance
2. Docker was installed using docker.io repo

→ apt install docker.io

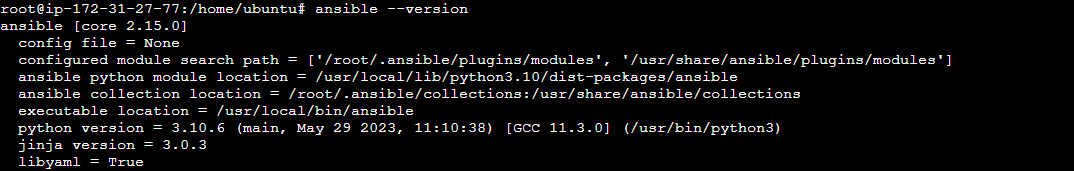
Check the docker version using docker --version



1. Ansible was installed using pip3 package

→ pip3 install ansible

Ansible –version



1. Build docker file for nginx image

Vim Dockerfile



2) create a repository in ELASTIC CONTAINER REGISTRY(ECR) and push the image

Install AWS CLI in your machine using these commands

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

—> apt install unzip

—> unzip awscliv2.zip

—> sudo ./aws/install

3. Minikube installation:

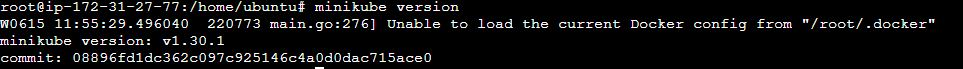
—> apt-get install curl wget apt-transport-https -y

—> wget https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64

—> cp minikube-linux-amd64 /usr/local/bin/minikube

—> chmod 755 /usr/local/bin/minikube

—> minikube version



--Install kubectl

→ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -

→ echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" | tee /etc/apt/sources.list.d/kubernetes.list

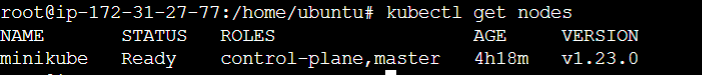
—>apt-get update -y

—> apt-get install kubectl -y

**Start Minikube**

**→ minikube start**

If we see kubectl get nodes



**Build docker image for sample nginx/python and push to aws elastic container registry(ECR) using docker cli.**

To authenticate docker client with ECR registry

→ aws ecr get-login-password --region us-east-2 | docker login --username AWS --password-stdin 543066576745.dkr.ecr.us-east-2.amazonaws.com

Build and tag the image we created

→ docker docker tag finaltask:latest

→ 543066576745.dkr.ecr.us-east-2.amazonaws.com/finaltask:latestbuild -t finaltask .

To push the image we already created using above mentioned Dockerfile

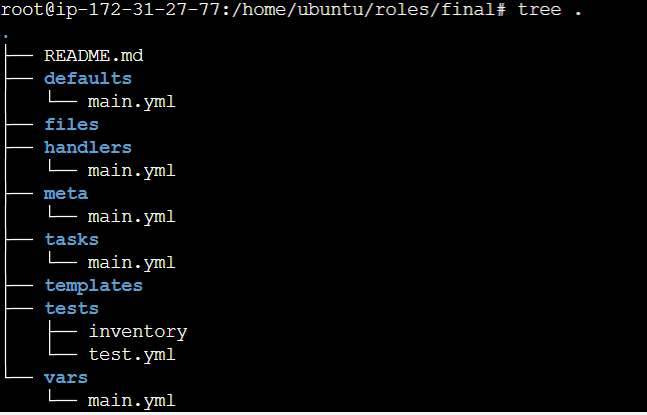
→docker push 543066576745.dkr.ecr.us-east-2.amazonaws.com/finaltask:latest

**Create ansible role to deploy nginx application into kubernetes cluster.**

created an ansible role using

→ ansible-galaxy init roles/role\_name

This will create a role with a directory structure



Every directory has a separate role to do

Here, inside tasks directory , i mentioned all my plays

And to pull an image from ECR repository we need to provide authentication details in /root/.docker/config.json with base64 format of your ECR respo credentials

Steps:

→aws ecr get-login-password --region <your\_region>

It will display you key, then convert it into base 64 format using

→ echo -n “<key>” | base64

It will give base64 format key, copy it and we need to provide these authentication details in

/root/.docker/config.json in the format

{

"auths": {

"your\_account\_id.dkr.ecr.your\_region.amazonaws.com": {

"auth": "your\_base64\_encoded\_credentials"

}

}

}

**tasks/main.yml**

**---**

**# tasks file for final**

**- name: Deploy NGINX Deployment**

**kubernetes.core.k8s:**

**state: present**

**definition:**

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**namespace: default**

**name: nginx-app**

**labels:**

**app: nginx-app**

**spec:**

**replicas: 2**

**selector:**

**matchLabels:**

**app: nginx-app**

**template:**

**metadata:**

**labels:**

**app: nginx-app**

**spec:**

**containers:**

**- name: nginx-app**

**image: 543066576745.dkr.ecr.us-east-2.amazonaws.com/finaltask:latest**

**imagePullPolicy: Always**

**ports:**

**- containerPort: 80**

**imagePullSecrets:**

**- name: regcred**

**- name: Create NGINX Service**

**kubernetes.core.k8s:**

**state: present**

**definition:**

**apiVersion: v1**

**kind: Service**

**metadata:**

**namespace: default**

**name: nginx-app**

**labels:**

**name: nginx-app**

**spec:**

**selector:**

**app: nginx-app**

**type: NodePort**

**ports:**

**- protocol: TCP**

**port: 8080**

**nodePort: 30010**

**targetPort: 80**

**- Create ansible playbooks to deploy nginx applications.**

Created a playbook.yml file to perform deployment on kubernetes cluster using that role i created

**---**

**- name: deployment**

**hosts: localhost**

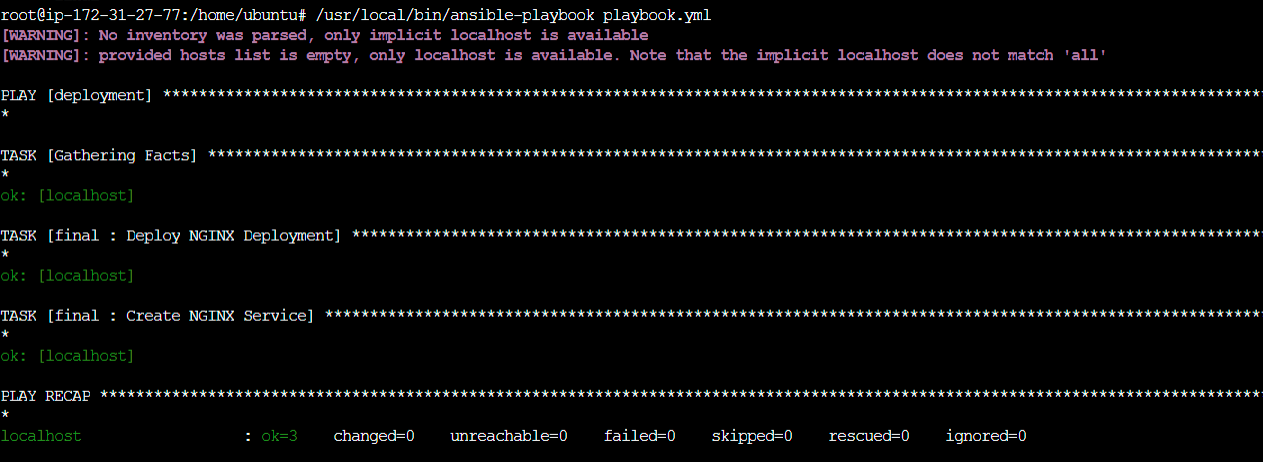
**remote\_user: root**

**roles:**

**- final**

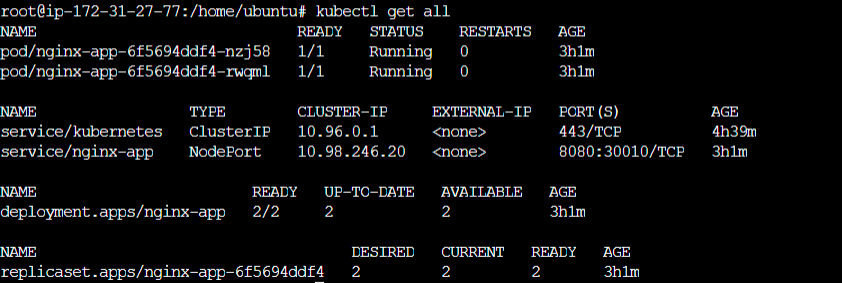
Run the playbook using

→ ansible-playbook playbook.yml



Kubernetes deployment and services are created

Kubectl get all → to see all the kubernetes objects



**-Outcome is an application deployed and run in k8s cluster. Able to access nginx website with url http://<ip\_address>:<port>.**

Before accessing the application from the browser, we will enable a secure tunnel from local machine to pod using

**—> kubectl port-forward --address 0.0.0.0 svc/<service\_name> portnumber**

**In this case**

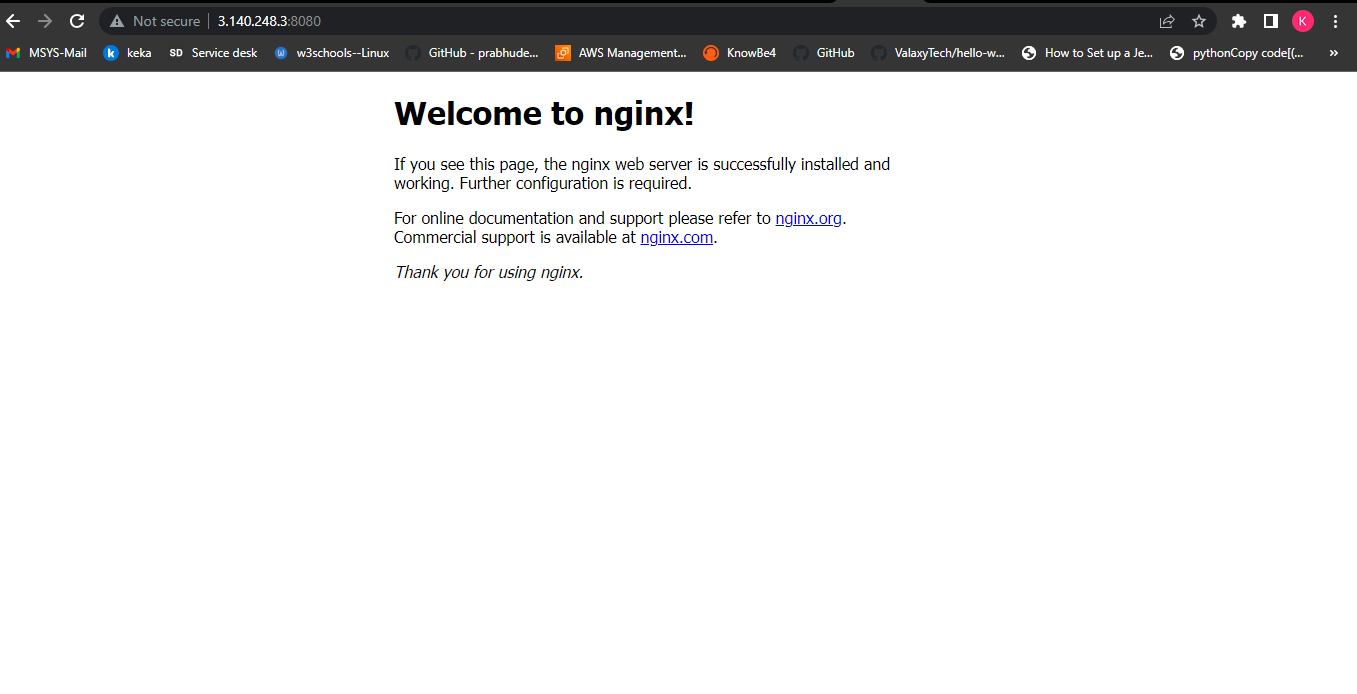
**—>kubectl port-forward --address 0.0.0.0 svc/nginx-app 8080:8080**

And now we can access it.

To open the nginx web application from browser

[**http://3.140.248.3:8080/**](http://3.140.248.3:8080/)

**The output will be like this**

****