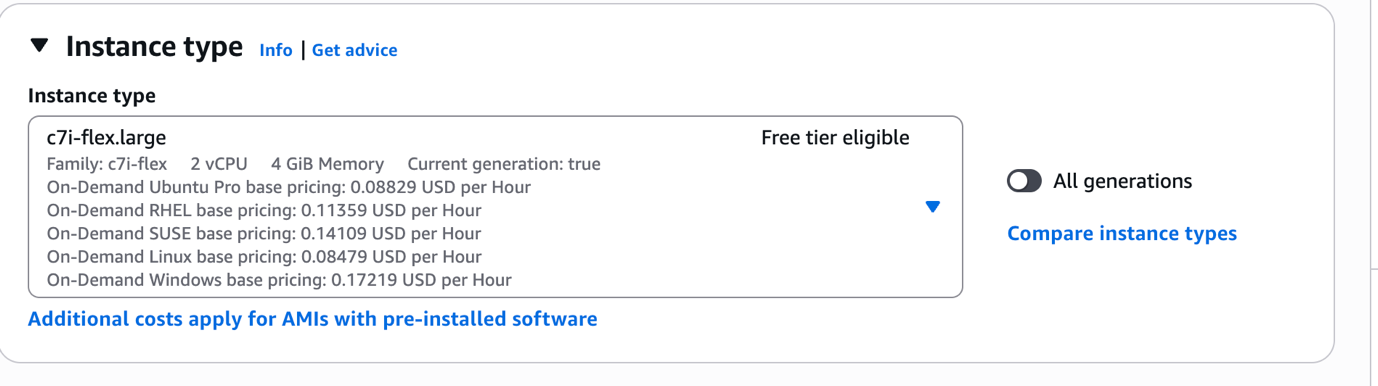
Launch an EC2 Instance with specifications as c7i-flex.large



sudo apt update -y && sudo apt upgrade -y

**# Install Java (required for Jenkins)**

sudo apt install openjdk-17-jre -y

java -version

**# Install Git**

sudo apt install git -y

git –version

**# Install Docker**

sudo apt install docker.io -y

sudo systemctl enable docker

sudo systemctl start docker

sudo usermod -aG docker $USER

docker --version

# Install Kubernetes CLI (kubectl)

# Download the latest release of kubectl

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

# Make the kubectl binary executable

chmod +x kubectl

# Move it into your PATH

sudo mv kubectl /usr/local/bin/

# Verify the installation

kubectl version –client

# Install eksctl (if you want AWS EKS)

# Download and extract the latest eksctl release

curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" | tar xz -C /tmp

# Move the binary into your PATH

sudo mv /tmp/eksctl /usr/local/bin

# Verify installation

eksctl version

# Install Ansible

sudo apt install ansible -y

ansible --version

# Install Jenkins

# Add the Jenkins key

curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null

# Add the Jenkins apt repository

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null

# Update packages

sudo apt-get update

# Install Jenkins

sudo apt-get install jenkins -y

# Enable Jenkins to start at boot

sudo systemctl enable jenkins

# Start Jenkins service

sudo systemctl start jenkins

# Check Jenkins service status

sudo systemctl status jenkins

# Switch to Jenkins user

sudo su – jenkins

# Generate SSH Key

ssh-keygen -t rsa -b 4096 -C "jenkins@apache"

cat ~/.ssh/id\_rsa.pub

Add this public key into GitHub → Repo → Settings → Deploy Keys →

Add Key (with write access).

we have to give the visudo permissions to Jenkins

jenkins ALL=(ALL:ALL) NOPASSWD: ALL

Inside Jenkins workspace or infra repo, create apache-setup.yml:

- hosts: all

become: yes

tasks:

- name: Install Apache

apt:

name: apache2

state: present

update\_cache: yes

- name: Start and Enable Apache

service:

name: apache2

state: started

enabled: yes

This is Inventory file: hosts.ini

[web]

localhost ansible\_connection=local

ansible-playbook -i hosts.ini apache-setup.yml

A screenshot of a computer

Description automatically generated

We Have to push image to dockerhub:

git clone https://github.com/nachiketbagad/apachewebsite.git

cd apachewebsite

# Build image from Dockerfile

docker build -t apache-website:v1 .

# Run container locally

docker run -d -p 8081:80 apache-website:v1

# Test

http://<publicip>:8081

If it showing permissions denied then use the below commands to

give permissions:

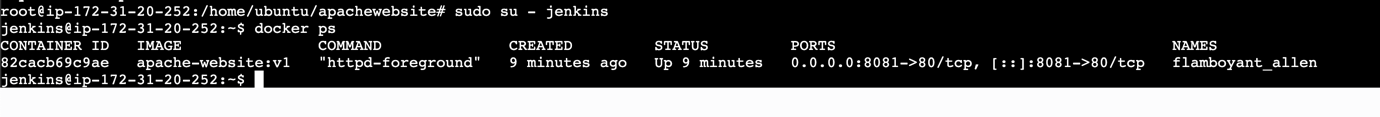
docker login -u nachiketbagad

sudo usermod -aG docker jenkins

sudo systemctl restart jenkins

sudo su - jenkins

docker ps



Re-run your Jenkins build step:

docker login

docker tag apache-website:v1 nachiketbagad/apache-website:v1

docker push nachiketbagad/apache-website:v1

before doing this make sure you have a cluster with nodes if you

don't have create it.

first we have to install aws cli

# Update and install dependencies

sudo apt update -y

sudo apt install unzip curl -y

# Download AWS CLI v2 installer

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

# Unzip the installer

unzip awscliv2.zip

# Run the installer

sudo ./aws/install

# Verify installation

aws --version

now we have to install eksctl now and then create cluster

# Install eksctl

curl -sL "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" -o eksctl.tar.gz

tar -xzf eksctl.tar.gz -C /tmp

sudo mv /tmp/eksctl /usr/local/bin

eksctl version

# Create a cluster (takes ~15–20 mins)

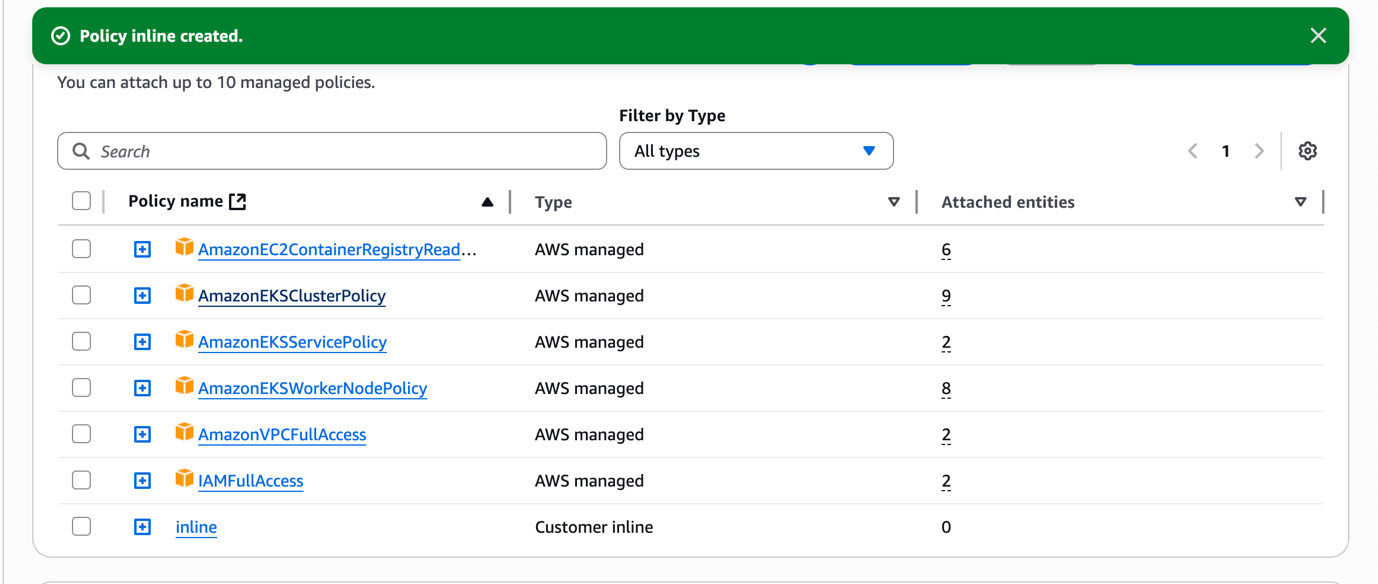
eksctl create cluster \

--name nachiket-eks \

--region ap-south-1 \

--nodes 2 \

--node-type c7i-flex.large



A screenshot of a computer

Description automatically generated

Inline Policy code:-

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"eks:\*",

"ec2:\*",

"iam:CreateRole",

"iam:AttachRolePolicy",

"iam:PutRolePolicy",

"iam:CreateInstanceProfile",

"iam:AddRoleToInstanceProfile",

"iam:PassRole",

"cloudformation:\*",

"autoscaling:\*"

],

"Resource": "\*"

}

]

}

Create deployment.yml using vi command:

apiVersion: apps/v1

apiVersion: apps/v1

kind: Deployment

metadata:

name: apache-website

spec:

replicas: 2

selector:

matchLabels:

app: apache-website

template:

metadata:

labels:

app: apache-website

spec:

containers:

- name: apache-website

image: nachiketbagad/apache-website:v1

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: apache-service

spec:

selector:

app: apache-website

ports:

- protocol: TCP

port: 80

targetPort: 80

type: LoadBalancer

kubectl apply -f deployment.yaml

kubectl get deployments

kubectl get pods

kubectl get svc

Copy the LoadBalancer EXTERNAL-IP and open in browser → you

should see your Apache website.

Generate kubeconfig for your EKS cluster, This creates a kubeconfig

file that points to your EKS cluster.

mkdir -p /var/lib/jenkins/.kube

aws eks update-kubeconfig --name nikk-eks --region ap-south-1 –kubeconfig /var/lib/jenkins/.kube/config

sudo chown -R jenkins:jenkins /var/lib/jenkins/.kube

sudo chmod 600 /var/lib/jenkins/.kube/config

sudo su - jenkins

kubectl get nodes

Now we have to build the pipeline by pushing the files which we

have created using playbooks. then build the pipeline using the

GitHub repo. Add jenkinsfile in your repo and build the pipeline in

it.

Add the GitHub and docker credential in Jenkins. then click on

build.

Pipeline:

pipeline {

agent any

stages {

stage('Checkout') {

steps {

git branch: 'master', url: 'https://github.com/Challakumar241/apachewebsite.git'

}

}

stage('Ansible Install Apache') {

steps {

sh 'ansible-playbook -i hosts.ini apache-setup.yml'

}

}

stage('Docker Build & Push') {

steps {

withCredentials([usernamePassword(credentialsId: 'dockerhub', usernameVariable: 'DOCKER\_USER', passwordVariable: 'DOCKER\_PASS')]) {

sh """

echo $DOCKER\_PASS | docker login -u $DOCKER\_USER --password-stdin

docker build -t apache-website:v1 .

docker tag apache-website:v1 $DOCKER\_USER/apache-website:v1

docker push $DOCKER\_USER/apache-website:v1

"""

}

}

}

stage('Deploy to K8s') {

steps {

sh 'kubectl --kubeconfig=/var/lib/jenkins/.kube/config apply -f deployment.yml'

}

}

}

}

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

