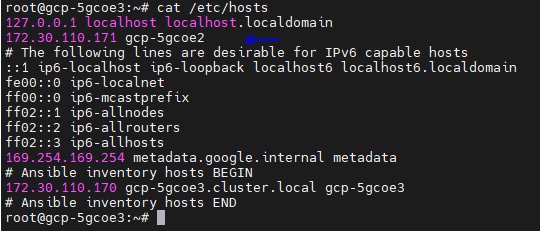
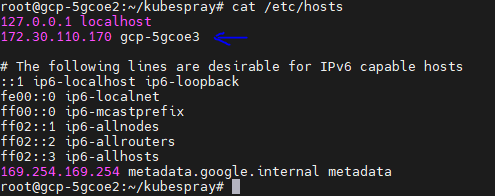
1. On VM3:

cat /etc/hosts



1. On VM2:

Cat /etc/hosts



1. On both VMs:

apt-get update

apt-get install python3-pip python3-dev -y

sudo apt update

sudo apt install software-properties-common

sudo add-apt-repository --yes --update ppa:ansible/ansible

sudo apt install ansible

ssh-keygen

1. on VM2:

ssh-copy-id root@172.30.110.171

ssh root@172.30.110.171

exit

1. on VM3:

ssh-copy-id root@172.30.110.170

ssh root@172.30.110.170

exit

**Kubespray repo steps:**

On VM2:

1. git clone https://github.com/kubernetes-sigs/kubespray.git
2. cd kubespray
3. sudo pip3 install -r requirements.txt
4. vim host.ini ----- >>> copy-paste the following contents and save

[all]

gcp-5gcoe3 ansible\_host=172.30.110.170

[kube-master]

gcp-5gcoe3 ansible\_host=172.30.110.170

[etcd]

gcp-5gcoe3 ansible\_host=172.30.110.170

[kube-node]

gcp-5gcoe3 ansible\_host=172.30.110.170

[k8s-cluster:children]

kube-master

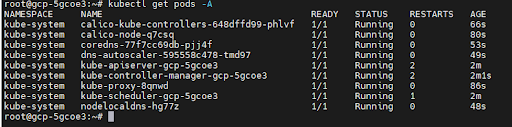
kube-node

1. in VM2: under /root/kubespray folder shoot the following command:

ansible-playbook -i hosts.ini cluster.yml

1. Once the script gets executed successfully, check on **VM3** by shooting the following command:

Kubectl get pods -A



**In order to add Argocd, OAI core, Prometheus and Grafana playbook steps I made the following changes in the system**

1. cd /root/kubespray

**modify the cluster.yml contents as shown below:**

root@gcp-5gcoe2:~/kubespray# cat cluster.yml

---

- name: Install Kubernetes

ansible.builtin.import\_playbook: playbooks/cluster.yml

- name: Install Kubernetes

ansible.builtin.import\_playbook: playbooks/cluster2.yml

1. cd /root/kubespray/playbooks

**add another file called cluster2.yml**

vim cluster2.yml

and paste the following contents and save the file:

- name: Install and Configure ArgoCD

hosts: gcp-5gcoe3

become: yes # Use sudo to run commands as root

tasks:

- name: Run update script

shell: |

#!/bin/bash

sudo apt-get update

ansible-galaxy collection install geerlingguy.argocd

sudo apt-get install -y python3-pip # Added '-y' to automatically accept installation

pip3 install kubernetes

kubectl create namespace argocd

kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "NodePort"}}'

kubectl get svc -n argocd

register: update\_result

changed\_when: "'0 upgraded' not in update\_result.stdout"

- name: Wait for the argocd-initial-admin-secret to be ready

shell: kubectl -n argocd wait --for=condition=ready pod -l app.kubernetes.io/name=argocd-server --timeout=300s

register: wait\_result

until: wait\_result.rc == 0

retries: 5

delay: 10

changed\_when: false # Mark this task as not changed, as it doesn't affect the playbook state

- name: Get the password of ARGOCD GUI

shell: |

#!/bin/bash

kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d

register: argocd\_password

when: wait\_result.rc == 0 # Only execute when the wait was successful

- name: Display update output

debug:

var: update\_result.stdout\_lines

- name: Display ARGOCD Password

debug:

var: argocd\_password.stdout

- name: Install ArgoCD CLI

shell: |

#!/bin/bash

curl -sSL -o argocd-linux-amd64 https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64

sudo install -m 555 argocd-linux-amd64 /usr/local/bin/argocd

become: yes

- name: Extract Port Number from kubectl output

hosts: gcp-5gcoe3

gather\_facts: no

tasks:

- name: Run kubectl command

shell: kubectl get svc -n argocd

register: kubectl\_output

changed\_when: false # Mark this task as not changed, as it's just gathering information

- name: Extract the port number

set\_fact:

argocd\_port: "{{ (kubectl\_output.stdout\_lines | select('match', '^argocd-server\\s+NodePort')) | map('regex\_replace', '.\*80:(\\d+)/TCP.\*', '\\1') | first }}"

when: kubectl\_output.stdout\_lines is defined and kubectl\_output.stdout\_lines | select('match', '^argocd-server\\s+NodePort') | list | count > 0

- name: Display the extracted port number

debug:

var: argocd\_port

- name: Get the port number

set\_fact:

argocd\_ip\_with\_port: "172.30.110.170:{{ argocd\_port }}"

when: argocd\_port is defined

- name: Log in to ArgoCD CLI

set\_fact:

argocd\_login\_command: "argocd login {{ argocd\_ip\_with\_port }} --insecure --username=admin --password='{{ argocd\_password.stdout }}'"

when: argocd\_port is defined

- name: Create ArgoCD Application

hosts: gcp-5gcoe3

gather\_facts: no

tasks:

- name: Run ArgoCD login command

shell: "{{ argocd\_login\_command }}"

register: argocd\_login\_result

changed\_when: false

- name: Create Kubernetes Secret for Git Credentials

shell: |

kubectl get secret argocd-repo-secret -n argocd || \

kubectl create secret generic argocd-repo-secret \

--from-literal=username=neha.mishra2@globallogic.com \

--from-literal=password=ghp\_4YTjmdZrCUeYpDMFSfLPmGaF6I8Gps4Etm6p \

--type=kubernetes.io/basic-auth \

-n argocd

when: argocd\_login\_result.rc == 0

- name: Create ArgoCD Application

shell: |

argocd app create my-oai-application \

--repo https://github.com/nehamishra2globallogic/oaicorebasic.git \

--path charts/oai-5g-core/oai-5g-basic \

--dest-namespace argocd \

--dest-server https://kubernetes.default.svc \

--sync-policy automated \

--auto-prune

argocd app patch my-oai-application \

--patch '{"spec": {"source": {"repoURL": "https://github.com/nehamishra2globallogic/oaicorebasic.git", "targetRevision": "HEAD"}, "upsert": {"secretName": "argocd-repo-secret"}}}'

argocd app sync my-oai-application

when: argocd\_login\_result.rc == 0

- name: Install Docker

hosts: gcp-5gcoe3 # Replace with the appropriate host or group

become: yes # Run tasks with sudo privileges

tasks:

- name: Update apt package cache

apt:

update\_cache: yes

- name: Install dependencies

apt:

name:

- apt-transport-https

- ca-certificates

- curl

- gnupg

- lsb-release

- name: Clean up existing Docker GPG key file

become: yes

file:

path: /usr/share/keyrings/docker-archive-keyring.gpg

state: absent

- name: Download Docker GPG key

become: yes

shell: curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /tmp/docker-archive-keyring.gpg

- name: Import Docker GPG key

become: yes

shell: gpg --dearmor --batch -o /usr/share/keyrings/docker-archive-keyring.gpg /tmp/docker-archive-keyring.gpg

- name: Set up Docker stable repository

become: yes

shell: echo "deb [signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null

- name: Set up Docker stable repository

shell: echo "deb [signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null

- name: Update apt package cache (again with Docker repository)

apt:

update\_cache: yes

- name: Install Docker

apt:

name: docker-ce

state: present

- name: Ensure Docker service is started

service:

name: docker

state: started

- name: Setup Prometheus and Grafana

hosts: gcp-5gcoe3

gather\_facts: false

become: yes

tasks:

- name: Install docker Python module

pip:

name: docker

executable: /usr/bin/pip3 # Adjust this path based on your Python version

become: yes

- name: Pull Docker images

docker\_image:

name: "{{ item }}"

source: pull

with\_items:

- grafana/grafana

- prom/prometheus

- bitnami/node-exporter

- name: Run Grafana container

docker\_container:

name: grafana

image: grafana/grafana

ports:

- "3001:3000"

state: started

become: yes

- name: Check Grafana port

shell: "netstat -anpl | grep -i 3001"

- name: Create Prometheus configuration file directory

file:

path: /etc/prometheus

state: directory

become: yes

- name: Create Prometheus configuration file

copy:

content: |

global:

scrape\_interval: 10s

scrape\_configs:

- job\_name: 'node'

scrape\_interval: 5s

static\_configs:

- targets: ['172.30.110.170:9100']

dest: /etc/prometheus/prometheus.yml

become: yes

- name: Create Prometheus data directory

file:

path: /tmp/prometheus-data

state: directory

mode: "0777"

- name: Adjust Prometheus directory permissions

file:

path: /etc/prometheus

state: directory

mode: "0755"

become: yes

- name: Adjust Prometheus configuration file permissions

file:

path: /etc/prometheus/prometheus.yml

state: file

mode: "0644" # Adjust permissions as needed

become: yes

- name: Run Prometheus container

docker\_container:

name: prometheus

image: prom/prometheus

ports:

- "9091:9090"

volumes:

- /etc/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml

- /tmp/prometheus-data:/prometheus

detach: true

register: prometheus\_result

- name: Check Prometheus container status

docker\_container\_info:

name: prometheus

register: prometheus\_info

ignore\_errors: yes

- name: Fail if Prometheus container is not running

fail:

msg: "Prometheus container failed to start. Check the logs for details."

when: "'State' in prometheus\_info and 'Status' in prometheus\_info.State and 'exited' in prometheus\_info.State.Status"

- name: Debug Prometheus container info

debug:

var: prometheus\_info

- name: Run Node Exporter container

docker\_container:

name: node-exporter

image: bitnami/node-exporter

ports:

- "9100:9100"

state: started

- name: Check Node Exporter port

shell: "netstat -anpl | grep -i 9100"

Now finally shoot this command:

root@gcp-5gcoe2:~/kubespray# ansible-playbook -i host.ini cluster.yml

in VM3: you can check with the following commands:

kubectl get pods -A

docker images

docker ps

open browser and open Argocd webpage: take ip as 172.30.110.170 and port take from the output and password also take from output, login and check if the application is created and all the pods are running.

Open browser and follow Prometheus and Grafana document steps which is available in this folder itself.

The final output must look in VM3 similar to the following screenshot:

