## 1 Introduction

This document is divided into 4 sections

1. Setup guidelines on AWS in Section 2
2. Setup guidelines on local laptop in section 3
3. Code setup and configuration in section 4
4. Execution instructions of code in section 5

Document should be followed based on the chosen setup. Follow section 2, 4 and 5 for cloud implementation and follow section 3, 4 and 5 for local implementation.

## 2 Configuration for Cloud Setup using KOPS on AWS

1. Setup ubuntu Linux machine on virtual Box

Configuration – 1CPU, 30GB storage, 2GB RAM. (Minimum requirements)

1. Install KOPS on local machine Kops.sigs.k8s.io (2020)
   1. Sudo apt-get update
   2. curl -LO https://github.com/kubernetes/kops/releases/download/ $(curl -s ht-tps://api.github.com/repos/kubernetes/kops/releases/latest — grep tag name — cut -d ’”’ -f 4)/kops-linux-amd64
   3. chmod +x kops-linux-amd64
   4. sudo mv kops-linux-amd64 /usr/local/bin/kops
2. 3) Configure AWS CLI
   1. Sudo apt-get install python-pip
   2. Pip install awscli
   3. Create AWS user with IAM policy – administrative access and access type –programmatic access. Note down Access key and Secret access key of user.
   4. Execute on local linux – aws configure and enter access key and secret accesskey.
3. Create S3 bucket
4. Domain Name Registration

DNS name was already present so that only domain so only DNS management services of AWS are used.

* 1. Create a hosted zone
  2. Get name-server values after creation of sub-domain in hosted zone of AWS
  3. Enter the name server with DNS name provider.

1. Create Public and Private SSH key. Ssh keygen -f .ssh/id rsa
2. Install Kubectl on linux machine.
3. Create Kops cluster (1 Master 3 Client) kops create cluster –name=kubernetes.anuragmsc.tk –state=s3://kops-state-anurag –zones=eu-west-1a –node-count=3 –nodee-size=t2.micro –master-sizee=t2.micro –

SSH Public Key=/home/anurag/.ssh/id rsa.pub –dns-zone=kubernetes.anuragmsc.tk

1. kops update cluster –name kubernetes.anuragmsc.tk –yes –state=s3://kops-stateanurag This command will create 4 EC2 Machines of t2.micro size, 6 Volumes, 4 Security groups and 1 Key pair.

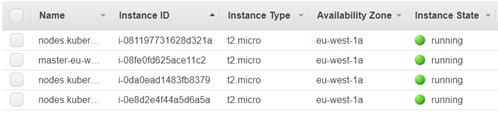


Figure 1: EC2 instances created on AWS

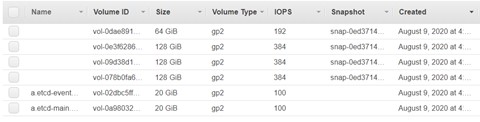


Figure 2: Volumes created on AWS

## 3 Testing Setup Configuration using Kubeadm on virtual box

1. Install Ubuntu Linux machine on Virtual Box

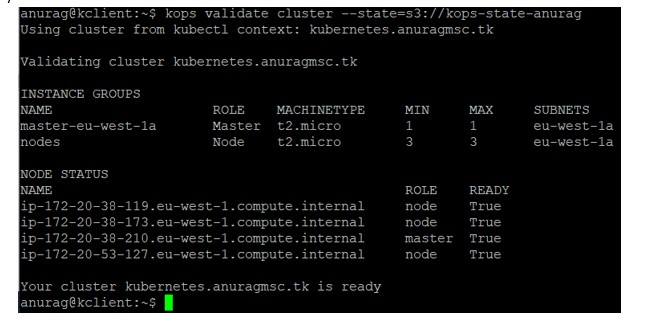


Figure 3: Validate cluster on AWS

Configuration of 1 Master Node – 2CPU, 30GB storage, 3GB RAM. (Minimum Requirement)

Configuration of 3 Client Node – 1CPU, 30GB storage, 2GB RAM. (Minimum requirements)

1. Network Configuration Virtual Box Settings
   1. Adopter 1 – NAT (Enabled)
   2. Adopter 2 – Host Only Adopter (Enabled)

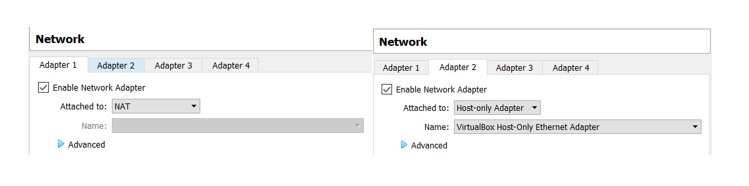


Figure 4: EC2 instances created on AWS

### 3.1 Common Configuration for Master and Client Nodes

1. Sudo su –
2. Apt-get update
3. Swapoff -a (Turning off the swap space)

Nano /etc/fstab

Comment the Swap Space line with

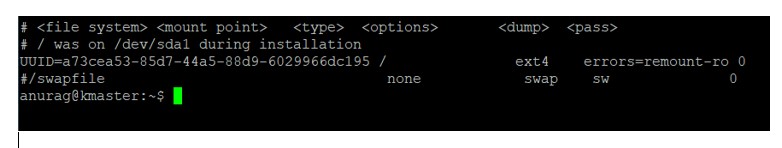


Figure 5: Swap off in fstab

1. Update the Hostname in /etc/hostname to kmaster on master node and kclient on client nodes
2. Set static IP address in file /etc/network/interfaces Insert following lines:

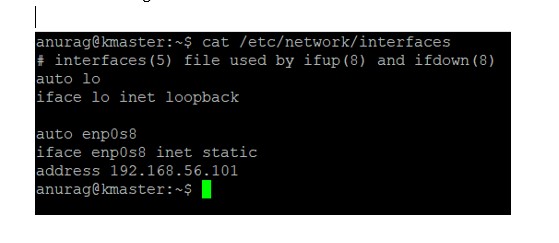


Figure 6: Static IP settings

1. Install Open SSH –

Apt-get install openssh-server

1. Install Docker –

Apt-get update

Apt-get install -y docker.io

1. Run following commands for Kubernetes installation kubernetes (2020) sudo apt-get update && sudo apt-get install -y apt-transport-https curl curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg — sudo apt-key add -

cat EOF — sudo tee /etc/apt/sources.list.d/kubernetes.list deb https://apt.kubernetes.io/ kubernetes-xenial main

EOF

sudo apt-get update

1. Install Kubernetes Kubelet, Kubeadm, Kubectl sudo apt-get install -y kubelet kubeadm kubectl
2. Edit following file sudo nano /etc/systemd/system/kubelet.service.d/10-kubeadm.conf add following line at the end of Environment variables.

Environment=”cgroup-driver=systemd/cgroup-driver=cgroupfs”

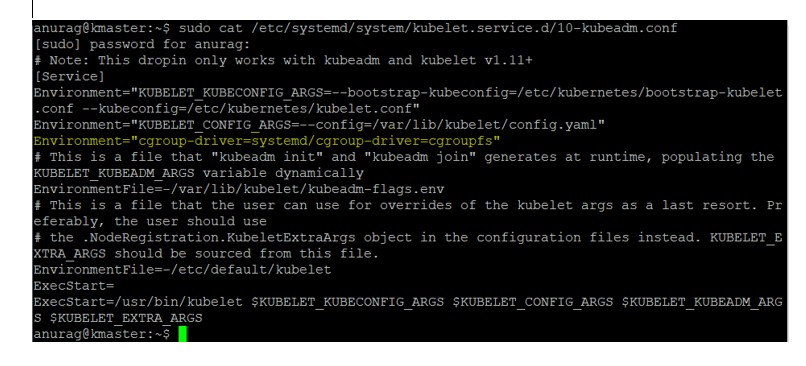


Figure 7: Environment Variables

1. Update /etc/hosts file with all the hostnames and IP addresses.

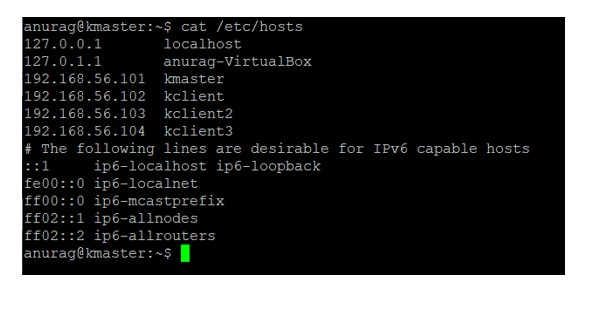


Figure 8: Host file on client machine

### 3.2 Commands only on Master Node

1. Initiate the Kubernetes cluster (Calico Network Commands)

Sudo kubeadm init –pod-network-cidr=192.168.0.0/16 –api-server-advertise-addresss=192.168.56.1

1. Run following commands as normal user mkdir -p $Home/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config sudo chown $(id -u):$(id -g) $HOME/.kube/config

1. Apply Calico Network Commands Docs.projectcalico.org (2020) kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml 4. Make sure all the internally created pods are in running state

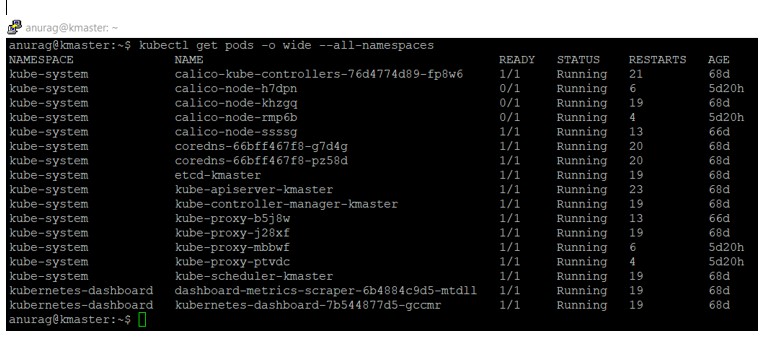


Figure 9: Pods in running state

### 3.3 Commands only on client nodes

1. Update the hosts file with IP address of master node
2. Join the Cluster – (Command to join the cluster will be displayed after initialisation of kubernetes cluster in step 1 of commands of master node)

kubeadm join 192.168.56.101:6443 –token ki8aa8.uylhd9rwtj4glbtq –discovery-tokenca-cert-hash sha256:a16547c5c5b34b4e34af953b6fae0c0be293c6e2a9c2c81f3e7b2e00bb24015c

1. Check the configuration on master node

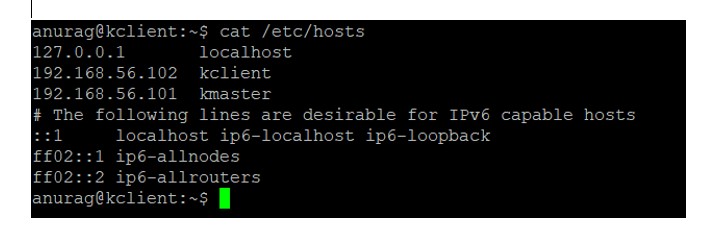


Figure 10: Hosts file on client nodes

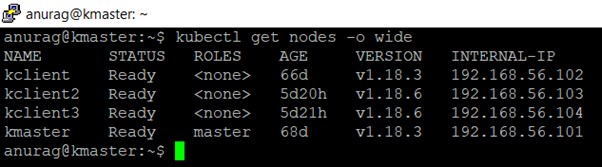


Figure 11: Hosts file on client nodes

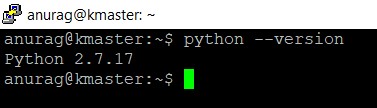


Figure 12: python version

## 4 Code Setup and configuration

1. Python and Libraries installation Install Python 2.7 version
2. Install python-pip on each node sudo apt-get install python-pip
3. Install Kubernetes, Psutil, Paramiko and numpy on each node with pip pip install Kubernetes psutil paramiko numpy
4. Place python file in home directory

On Master node - custom scheduler.py, topsisWithoutPrint.py, kubenginxrc.yaml, nginxrc.yaml.

On Client Node - systemConfiguration.py

1. Generate Pub and Private SSH key on master node and place .pub key on private nodes

## 5 Execution of the code instructions

1. Execution of existing Kubernetes scheduler kubenginxrc.yaml is used to create pods with default Kubernetes scheduler.

Command: kubectl create -f kubenginxrc.yaml

Node name is displayed as kclient. This node is used for scheduling the nginx pod created with kubenginxrc.yaml.

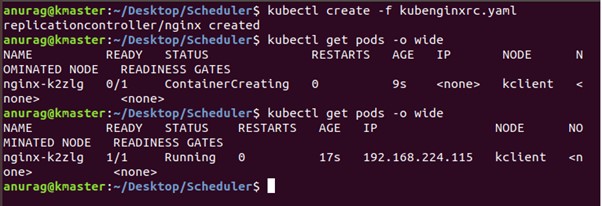


Figure 13: kubernetes scheduler

1. Execution of custom scheduler nginxrc.yaml is used to create pods with custom scheduler.

Command: kubectl create -f nginxrc.yaml

Status of pod will remain pending until custom scheduler.py is not executed.

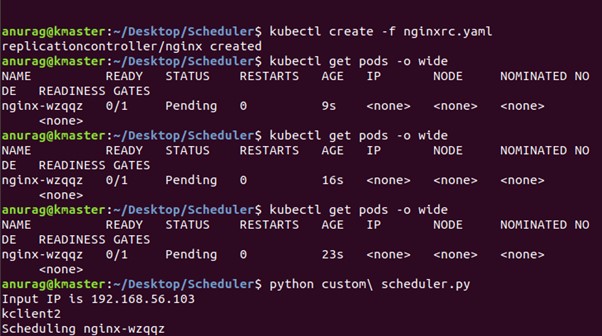


Figure 14: custom scheduler pod creation

1. Execute custom scheduler

Command:pythoncustom

scheduler.py

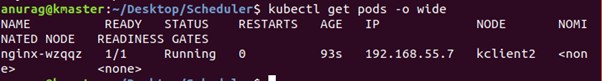


Figure15:

customschedulerexecution

## References

Docs.projectcalico.org (2020). Quickstart for calico on kubernetes.

**URL:** *https://docs.projectcalico.org/getting-started/kubernetes/quickstart*

Kops.sigs.k8s.io (2020). Installing - kubernetes operations - kops. **URL:** *https://kops.sigs.k8s.io/gettingstarted/install/*

kubernetes (2020). Installing kubeadm.

**URL:** *https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/installkubeadm*