National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Anurag Pravin Shriniwar
Student ID: x18198171

School of Computing
National College of Ireland

Supervisor: Mr. Vikas Sahani

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Anurag Pravin Shriniwar
Student ID:	x18198171
Programme:	Cloud Computing
Year:	2020
Module:	MSc Research Project
Supervisor:	Mr. Vikas Sahani
Submission Due Date:	17/08/2020
Project Title:	Configuration Manual
Word Count:	XXX
Page Count:	9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

I agree to an electronic copy of my thesis being made publicly available on TRAP the National College of Ireland's Institutional Repository for consultation.

Signature:	
Date:	15th August 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Anurag Pravin Shriniwar
x18198171

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)

2. Install KOPS on local machine Kops.sigs.k8s.io (2020)

   a. Sudo apt-get update

   b. curl -LO https://github.com/kubernetes/kops/releases/download/ $(curl -s https://api.github.com/repos/kubernetes/kops/releases/latest — grep tag_name — cut -d ”’ -f 4)/kops-linux-amd64

   c. chmod +x kops-linux-amd64

   d. sudo mv kops-linux-amd64 /usr/local/bin/kops

3. 3) Configure AWS CLI

   a. Sudo apt-get install python-pip

   b. Pip install awscli

   c. Create AWS user with IAM policy – administrative access and access type – programmatic access. Note down Access key and Secret access key of user.

   d. Execute on local linux – aws configure and enter access key and secret access key.

4. Create S3 bucket

5. Domain Name Registration

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

   a. Create a hosted zone

   b. Get name-server values after creation of sub-domain in hosted zone of AWS

   c. Enter the name server with DNS name provider.

6. Create Public and Private SSH key.

   Ssh keygen -f .ssh/id_rsa

7. Install Kubectl on linux machine.

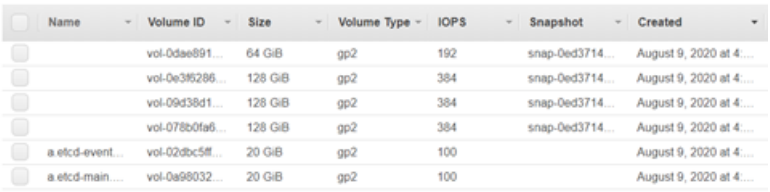
8. Create Kops cluster (1 Master 3 Client)

   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

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

Name	Instance ID	Instance Type	Availability Zone	Instance State
nodes.kuber...	i-081197731628d321a	t2.micro	eu-west-1a	running
master-eu-w...	i-08fe0fd625ace11c2	t2.micro	eu-west-1a	running
nodes.kuber...	i-0da0ead1483fb8379	t2.micro	eu-west-1a	running
nodes.kuber...	i-0e8d2e4f44a5d6a5a	t2.micro	eu-west-1a	running

Figure 1: EC2 instances created on AWS



Name	Volume ID	Size	Volume Type	IOPS	Snapshot	Created
	vol-0dae891...	64 GiB	gp2	192	snap-0ed3714...	August 9, 2020 at 4:...
	vol-0e3f5286...	128 GiB	gp2	384	snap-0ed3714...	August 9, 2020 at 4:...
	vol-09d38d1...	128 GiB	gp2	384	snap-0ed3714...	August 9, 2020 at 4:...
	vol-078b0fa6...	128 GiB	gp2	384	snap-0ed3714...	August 9, 2020 at 4:...
a.etcd-event...	vol-02dbc5ff...	20 GiB	gp2	100		August 9, 2020 at 4:...
a.etcd-main....	vol-0a98032...	20 GiB	gp2	100		August 9, 2020 at 4:...

Figure 2: Volumes created on AWS

3 Testing Setup Configuration using Kubeadm on virtual box

1. Install Ubuntu Linux machine on Virtual Box

```
anurag@kclient:~$ kops validate cluster --state=s3://kops-state-anurag
Using cluster from kubectl context: kubernetes.anuragmsc.tk

Validating cluster kubernetes.anuragmsc.tk

INSTANCE GROUPS
NAME                    ROLE    MACHINETYPE     MIN     MAX     SUBNETS
master-eu-west-1a       Master  t2.micro        1       1       eu-west-1a
nodes                   Node    t2.micro        3       3       eu-west-1a

NODE STATUS
NAME                                            ROLE    READY
ip-172-20-38-119.eu-west-1.compute.internal     node    True
ip-172-20-38-173.eu-west-1.compute.internal     node    True
ip-172-20-38-210.eu-west-1.compute.internal     master  True
ip-172-20-53-127.eu-west-1.compute.internal     node    True

Your cluster kubernetes.anuragmsc.tk is ready
anurag@kclient:~$
```

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)

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

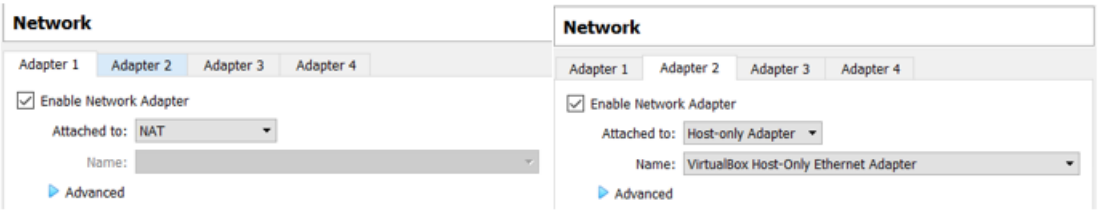


Figure 4: EC2 instances created on AWS

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

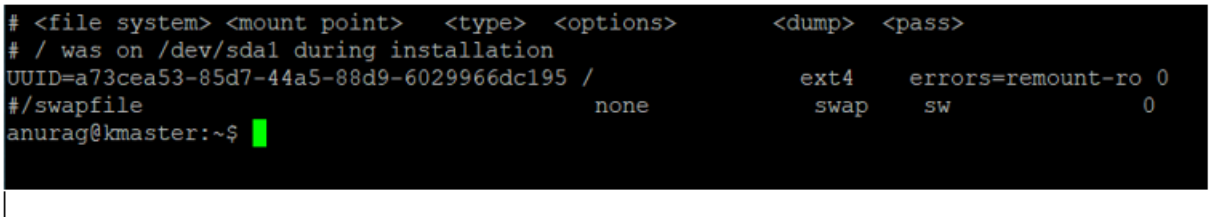


Figure 5: Swap off in fstab

4. Update the Hostname in /etc/hostname to kmaster on master node and kclient on client nodes

5. Set static IP address in file /etc/network/interfaces Insert following lines:

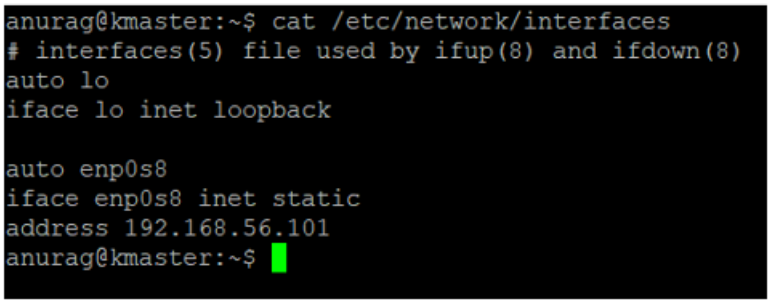


Figure 6: Static IP settings

6. Install Open SSH –

   Apt-get install openssh-server

7. Install Docker –

   Apt-get update

   Apt-get install -y docker.io

8. 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

9. Install Kubernetes Kubelet, Kubeadm, Kubectl
   sudo apt-get install -y kubelet kubeadm kubectl

10. 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”

```
anurag@kmaster:~$ sudo cat /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
[sudo] password for anurag:
# Note: This dropin only works with kubeadm and kubelet v1.11+
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet
.conf --kubeconfig=/etc/kubernetes/kubelet.conf"
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"
Environment="cgroup-driver=systemd/cgroup-driver=cgroupfs"
# This is a file that "kubeadm init" and "kubeadm join" generates at runtime, populating the
KUBELET_KUBEADM_ARGS variable dynamically
EnvironmentFile=-/var/lib/kubelet/kubeadm-flags.env
# This is a file that the user can use for overrides of the kubelet args as a last resort. Pr
eferably, the user should use
# the .NodeRegistration.KubeletExtraArgs object in the configuration files instead. KUBELET_E
XTRA_ARGS should be sourced from this file.
EnvironmentFile=-/etc/default/kubelet
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS $KUBELET_KUBEADM_ARG
S $KUBELET_EXTRA_ARGS
anurag@kmaster:~$
```

Figure 7: Environment Variables

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

```
anurag@kmaster:~$ cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       anurag-VirtualBox
192.168.56.101  kmaster
192.168.56.102  kclient
192.168.56.103  kclient2
192.168.56.104  kclient3
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
anurag@kmaster:~$
```

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.

2. 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

3. 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

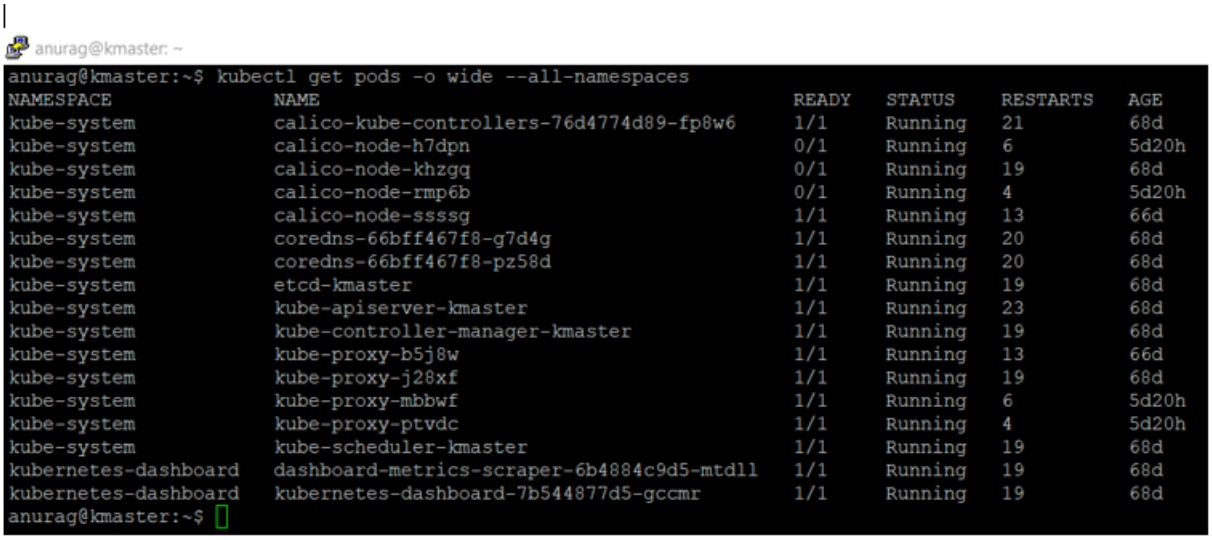


Figure 9: Pods in running state

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-token-ca-cert-hash sha256:a16547c5c5b34b4e34af953b6fae0c0be293c6e2a9c2c81f3e7b2e00bb24015c

3. Check the configuration on master node

```
anurag@kclient:~$ cat /etc/hosts
127.0.0.1       localhost
192.168.56.102  kclient
192.168.56.101  kmaster
# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
anurag@kclient:~$
```

Figure 10: Hosts file on client nodes

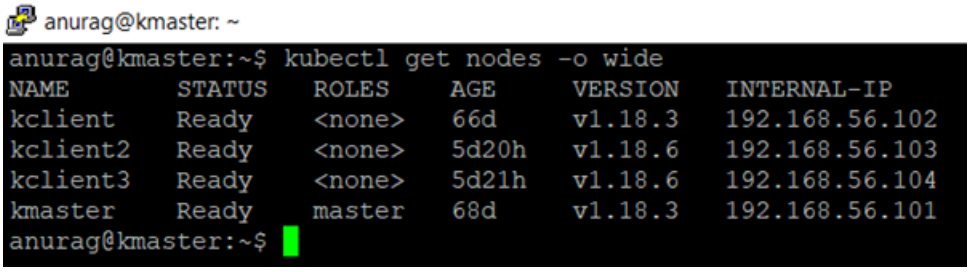


Figure 11: Hosts file on client nodes

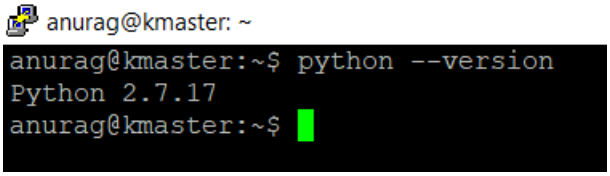


Figure 12: python version

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

5. 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.

```
anurag@kmaster:~/Desktop/Scheduler$ kubectl create -f kubenginxrc.yaml
replicationcontroller/nginx created
anurag@kmaster:~/Desktop/Scheduler$ kubectl get pods -o wide
NAME          READY   STATUS              RESTARTS   AGE   IP       NODE      N
OMINATED NODE   READINESS GATES
nginx-k2zlg   0/1     ContainerCreating   0          9s    <none>   kclient   <
none>           <none>
anurag@kmaster:~/Desktop/Scheduler$ kubectl get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP               NODE      NO
MINATED NODE   READINESS GATES
nginx-k2zlg   1/1     Running   0          17s   192.168.224.115   kclient   <n
one>           <none>
anurag@kmaster:~/Desktop/Scheduler$
```

Figure 13: kubernetes scheduler

2. 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.

```
anurag@kmaster:~/Desktop/Scheduler$ kubectl create -f nginxrc.yaml
replicationcontroller/nginx created
anurag@kmaster:~/Desktop/Scheduler$ kubectl get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP       NODE     NOMINATED NO
DE   READINESS GATES
nginx-wzqqz   0/1     Pending   0          9s    <none>   <none>   <none>
     <none>
anurag@kmaster:~/Desktop/Scheduler$ kubectl get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP       NODE     NOMINATED NO
DE   READINESS GATES
nginx-wzqqz   0/1     Pending   0          16s   <none>   <none>   <none>
     <none>
anurag@kmaster:~/Desktop/Scheduler$ kubectl get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP       NODE     NOMINATED NO
DE   READINESS GATES
nginx-wzqqz   0/1     Pending   0          23s   <none>   <none>   <none>
     <none>
anurag@kmaster:~/Desktop/Scheduler$ python custom\ scheduler.py
Input IP is 192.168.56.103
kclient2
Scheduling nginx-wzqqz
```

Figure 14: custom scheduler pod creation

3. Execute custom scheduler

   Command: python custom_scheduler.py

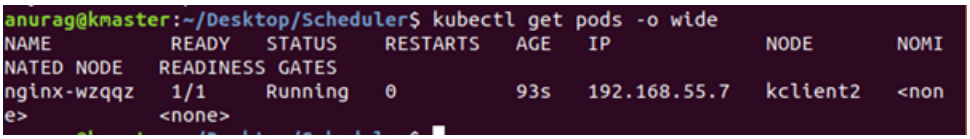


Figure 15: custom scheduler execution

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/getting*$_s$*tarted/install/*

kubernetes (2020). Installing kubeadm.
URL: *https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm*