Configuration Manual

Research Project
MSc Cloud Computing

Saurabh Kumar
Student ID: x18193188

School of Computing
National College of Ireland

Supervisor: Manuel Tova-Izquierdo

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Saurabh Kumar
Student ID:	x18193188
Programme:	MSc Cloud Computing
Year:	2020
Module:	Research Project
Supervisor:	Manuel Tova-Izquierdo
Submission Due Date:	17/08/2020
Project Title:	Configuration Manual
Word Count:	XXX
Page Count:	7

Signature:	
Date:	7th August 2020

Configuration Manual

Saurabh Kumar
x18193188

1 Introduction

In order to run the artifact, a platform needs to be configured and setup. Following are the components that need to be installed. Each component installation is discussed in the following sections.

- kubernetes[1]
- docker[2]
- golang[3]
- NATS server[4]
- linux based server[5]

1.1 VMs Configuration

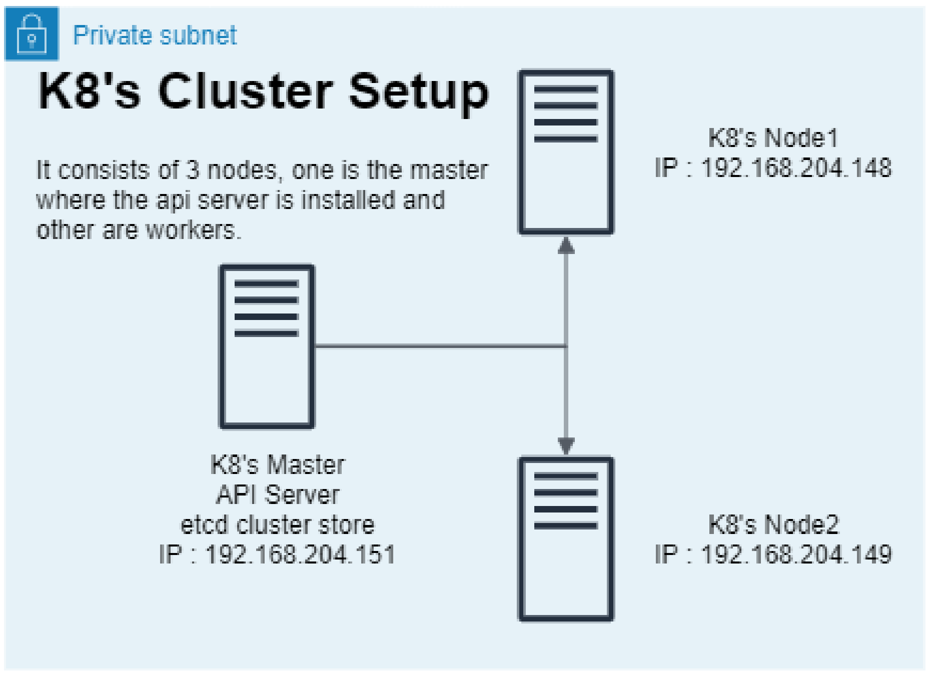


Figure 1: VMs setup on openstack

There are 3 VMs configured for installing the above components on openstack. Each VM has the following configuration.

- RAM : 2 GB
- CPU : 2.0 GHz
- Disk Space : 10 GB
- CPU cores : 2 physical cores 2 virtual cores

Among the 3 VMs that is taken for component installation, one acts as the kubernetes master and other 2 act as kubernetes workers, all container initialization and spawning of functions are done on the workers. Figure 1 shows the cluster setup for running the artifact.

1.2 Component Versions

Below table 1 consists of the versions corresponding to the components used in setting up the environment for deploying the artifact. The versions are important to ensure the smooth running of the artifact.

Component Name	Version
kubernetes	1.18
docker	19.03
golang	1.13
NATS server	2.1.7
linux based server	ubuntu 18.04
kernel	4.15.0

Table 1: Component and version table

2 Installation

2.1 Installation on all 3 VMs components

- Disable swap, swapoff then edit fstab removing any entry for swap partitions
  1. swapoff -a
  2. vi /etc/fstab
- Adding Google's apt repository gpg key
  1. curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg — sudo apt-key add -
- Adding the Kubernetes apt repository
  1. sudo bash -c 'cat <<EOF >/etc/apt/sources.list.d/kubernetes.list deb https://apt.kubernetes.io/ kubernetes-xenial main EOF'

- Update the package list and use apt-cache to inspect versions available in the repository

    1. sudo apt-get update
    2. apt-cache policy kubelet — head -n 20
    3. apt-cache policy docker.io — head -n 20

- Install the required packages

    1. sudo apt-get install -y docker.io kubelet kubeadm kubectl
    2. sudo apt-mark hold docker.io kubelet kubeadm kubectl

- Check the status of our kubelet and our container runtime, docker

    1. sudo systemctl status kubelet.service
    2. sudo systemctl status kubelet.service

- Add the above services to the inittab, so they are started on system boot

    1. sudo systemctl enable kubelet.service
    2. sudo systemctl enable docker.service

- Setup docker daemon

    1. sudo bash -c 'cat >/etc/docker/daemon.json <<EOF "exec-opts": ["native.cgroupdriver=systemd"], "log-driver": "json-file", "log-opts": "max-size": "100m" , "storage-driver": "overlay2" EOF'

- Restart all the VMs for the changes to take effect

2.2 Installation on master node

- Download the yaml files for the pod network - calico yaml file might have changed since the publication of the thesis and is now avaialble at the URL below.

    1. wget https://docs.projectcalico.org/manifests/calico.yaml

- Inside calico.yaml, find the network range CALICO_ IPV4POOL_ CIDR, adjust if needed.

    1. vi calico.yaml

- kubernetes cluster, specifying a pod network range matching that in calico.yaml, the pod ip ranges will be decided from this file.

    1. sudo kubeadm init –pod-network-cidr=192.168.0.0/16

- Our account on the master to have admin access to the API server from a non-privileged account.

    1. mkdir -p $ HOME / .kube

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

- Deploy yaml file of the pod network

   1. kubectl apply -f calico.yaml

2.3 Installation on worker node

- On the master, a list of token can be fetched using the following command.

   1. kubeadm token list

- Generate a new token if the old one has expired.

   1. kubeadm token create

- On the master, the CA cert hash can be found by following command.

   1. openssl x509 pubkey in /etc/kubernetes/pki/ca.crt — openssl rsa pubin outform der 2>/dev/null — openssl dgst -sha256 -hex

- Using the master (API Server) IP address, the token and the cert, use the below command to let the worker node join the master.

   1. sudo kubeadm join <IP_ ADDRESS_ OF_ MASTER > – token <TOKEN >–discovery-token-ca-cert-hash <CERTIFICATE >

2.4 Installation of NATS server

- The NATS server should be installed in order for the artifact to enable scatter-gather communication.

   1. Create a new file "nats-server-pod.yaml"
   2. Deploy the file on kubernetes : kubectl apply -f nats-server-pod.yaml
   3. Figure 2 shows the config details of the nats-server-pod.yaml

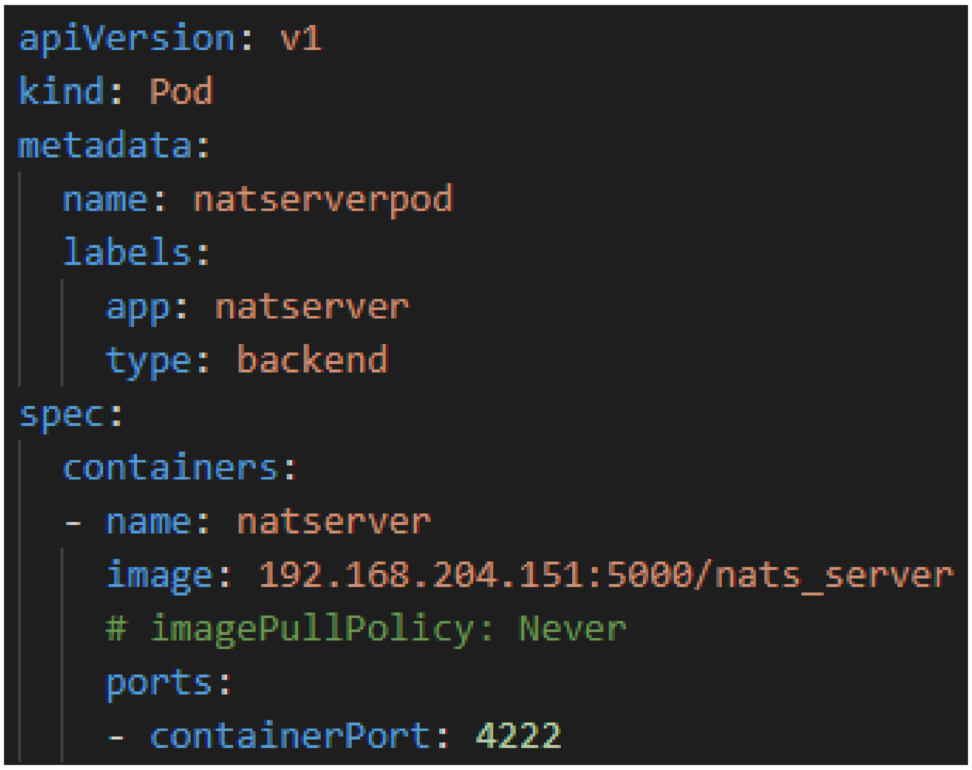
2.5 Installation of private registry in docker

- Create a private registry in docker

   1. docker run -d -p 5000:5000 –restart=always –name registry registry:2

- All the images are created and pushed on to the private registry so it can be pulled by the kubernetes engine while launching worker and master containers

- The purpose of creating a private registry is to ensure that the kubernetes engine doesnot pull the image from docker hub and there is a proper segregation of images created by the user and the pre-existing images.



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: natserverpod
  labels:
    app: natserver
    type: backend
spec:
  containers:
  - name: natserver
    image: 192.168.204.151:5000/nats_server
    # imagePullPolicy: Never
    ports:
    - containerPort: 4222
```

Figure 2: NATS server config yaml file

2.6 Installation of artifact

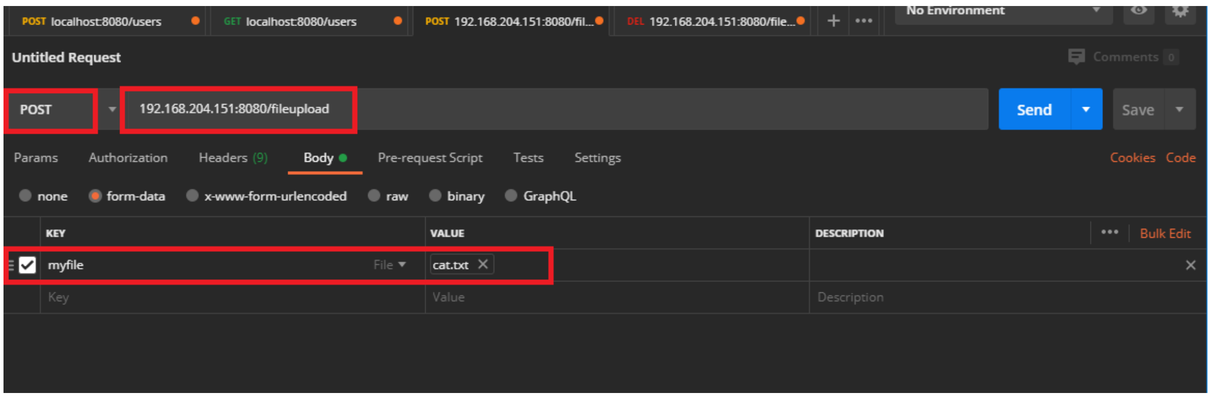
- Download the repo from Github - https://github.com/saurabh7517/thesis
- Save the golang program to the ip addresses where kubernetes is installed
- Run the following commnand
  1. go run github.com/saurabh7517/thesis
- The above command will launch a web server and a file can be uploaded using a post request, subsequently that file once uploaded will be divided into chunks and processed by the workers spawned by the artifact.

3 Execution

After the artifact and the environment is setup, a HTTP client application like postman needs to be installed to upload a file to trigger the artifact. Since the application is a serverless function it needs to be triggered by an external remote procedure call. In this case the RPC is a post request. Figure 3 illustrates the upload of a file using a HTTP client application Postman.

3.1 Output

Once the file is uploaded, this will trigger the artifact to launch master function, the master function will calculate the file depending on the configurable file size and launch



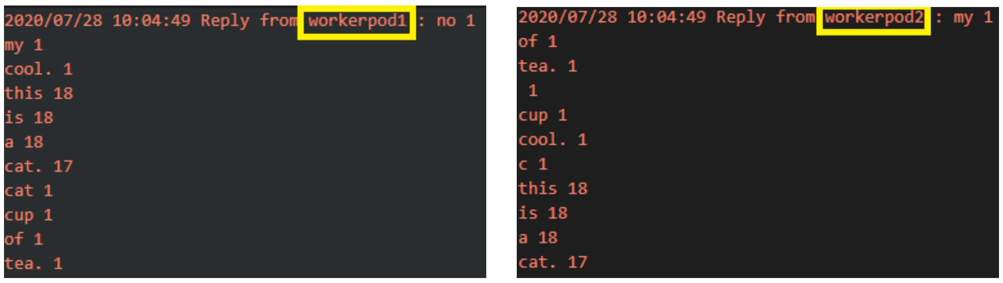
POST 192.168.204.151:8080/fileupload

KEY	VALUE	DESCRIPTION
myfile (File)	cat.txt	

Figure 3: Postman configuration for triggering the artifact

worker functions. For the following setup

- file size = 1,654 bytes
- block size = 300
- pods created = 6



```
2020/07/28 10:04:49 Reply from workerpod1 : no 1
my 1
cool. 1
this 18
is 18
a 18
cat. 17
cat 1
cup 1
of 1
tea. 1
```

```
2020/07/28 10:04:49 Reply from workerpod2 : my 1
of 1
tea. 1
 1
cup 1
cool. 1
c 1
this 18
is 18
a 18
cat. 17
```

Figure 4: Outputs from 2 worker functions

The above outputs can be seen if the access to the terminal on which the artifact is running is granted.

References

[1] G. Sayfan, *Mastering Kubernetes: Large scale container deployment and management.* Birmingham Mumbai: Packt Publishing, 2017, oCLC: 967366064.

[2] C. Boettiger, "An introduction to Docker for reproducible research," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 71–79, Jan. 2015. [Online]. Available: https://dl.acm.org/doi/10.1145/2723872.2723882

[3] M. Tsoukalos, *Mastering Go: create Golang production applications using network libraries, concurrency, and advanced Go data structures*, 2019, oCLC: 1127080883.

[4] "NATS - Open Source Messaging System | Secure, Native Cloud Application Development." [Online]. Available: https://nats.io/

[5] U. R. Sawant and Packt Publishing, *Ubuntu Server cookbook: arm yourself to make the most of the versatile, powerful Ubuntu Server with over 100 hands-on recipes*, 2016, oCLC: 949751073.