National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Neha Deshpande
Student ID: x19203896

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Neha Deshpande
Student ID:	x19203896
Programme:	Cloud Computing
Year:	2021
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	16/08/2020
Project Title:	Configuration Manual
Word Count:	952
Page Count:	14

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.

Signature:	Neha Deshpande
Date:	15th August 2021

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

Neha Deshpande
x19203896

1 Introduction

This two main part of this research are cloud-native application and Kubernetes cluster. Autoscaling of the cloud-native application is done with the help of kubernetes. The cloud-native application consists of microservices which are computing intensive.

1.1 Before you begin

Before you begin, please make sure you have installed below tools on your laptop.

- **git bash** : Installation Procedure available at `https://www.educative.io/edpresso/how-to-install-git-bash-in-windows`
- **windows Subsystem for Linux (WSL)** : If you do not have linux operating system on your machine please follow the procedure available at `https://www.windowscentral.com/install-windows-subsystem-linux-windows-10`

This paper is divided into structure

2 Cloud-Native Application

The Cloud-Native application is built using PHP 7.4. The application is available on `https://github.com/Nehadeshpande89/Masters_Thesis`. To clone the code of the application please use the below command :

git clone "`https://github.com/Nehadeshpande89/Masters_Thesis`"

Below figure shows the repository of the github.

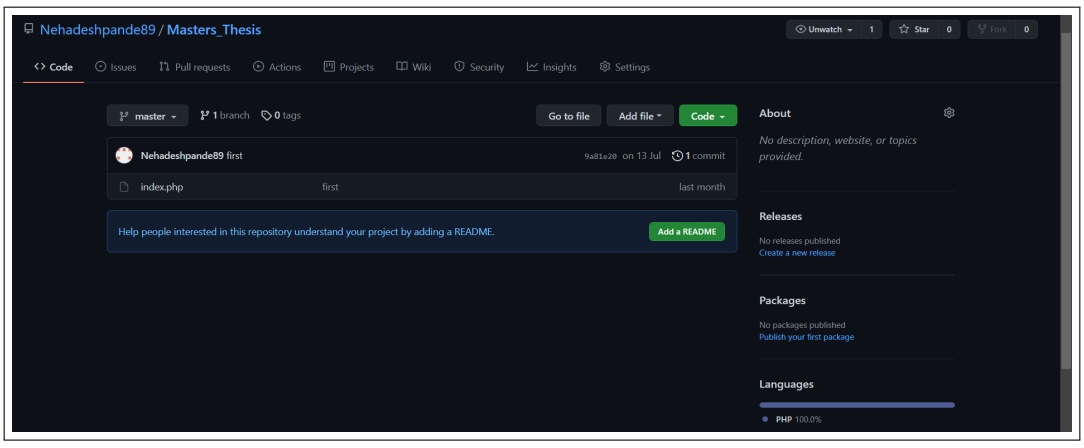

Figure 1: GitHub Repository

2.1 Microservice YAML File

Create a yaml of of the cloud-native application and name it as **php-apache.yaml**. The content of the yaml image is given in 2.

Figure 2: php-apache.yaml

3 System Specification

For this research a virtual machine with the required configuration is shown in the below figure 3.

Virtual Machine	
vCPU	8
Memory	32GiB
Network Performance	upto 5 gbps
Cost	$0.3341/ hr

Figure 3: System Specifications

4 Creation of AWS Instance

For this research the Amazon AMI machine **Ubuntu Server 20.04 LTS (HVM). SSD Volume Type(64 bit)** and t3.2xlarge EC2 instance(Not free tire eligible) is used.

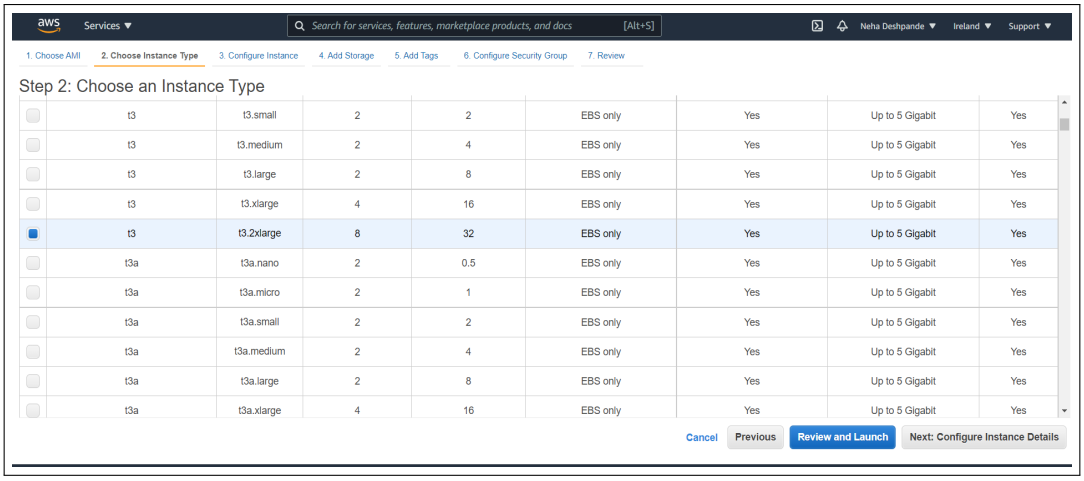

Figure 4: EC2 instance

4.1 Security Group for the instance

Before launching the instance, configure the security group **All Traffic (port range (0-65535))**.

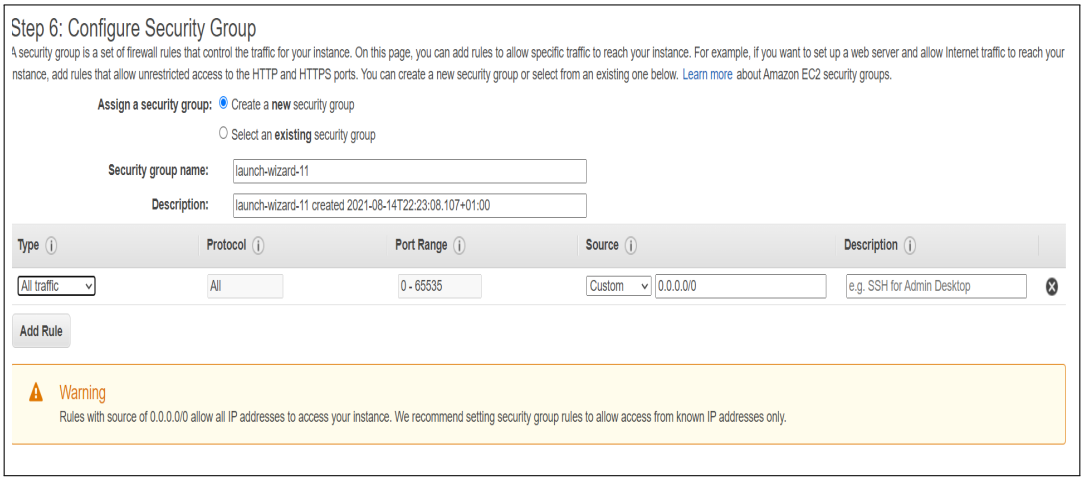

Figure 5: Security group instance

4.2 Connect to the instance

Connection of the instance is to be made with SSH. Before that, download the .pem file given by AWS and store in WSL folder. To connect the instance use the SSH command given by AWS. For example :

`ssh-i"kubernetes.dev.finalthesis.ie-23:6e:11:e5:52:f0:24:cf:05:74:85:b7:98:5a:a5:58.pem"ubuntu@ec2-52-213-8-175.eu-west-1.compute.amazonaws.com`

4.3 AWS CLI

After connecting to the instance successfully, to install the AWS CLI follow the below steps :

1. Go to the root user by using **sudo su -** command.
2. Update the dependenices by using **sudo apt update** command.
3. Next, follow the procedure available at `https://docs.aws.amazon.com/cli/latest/userguide/install-windows.html#awscli-install-windows-pip`
4. To check the AWS CLI is installed successfully, type **AWS –version** command.

5 Installation of Kubernetes Cluster

To install the Kubernetes Cluster , follow the below steps :

5.1 Install Docker Engine

To install the docker update the packages by using **sudo get apt update** and type the command shows as 7 :

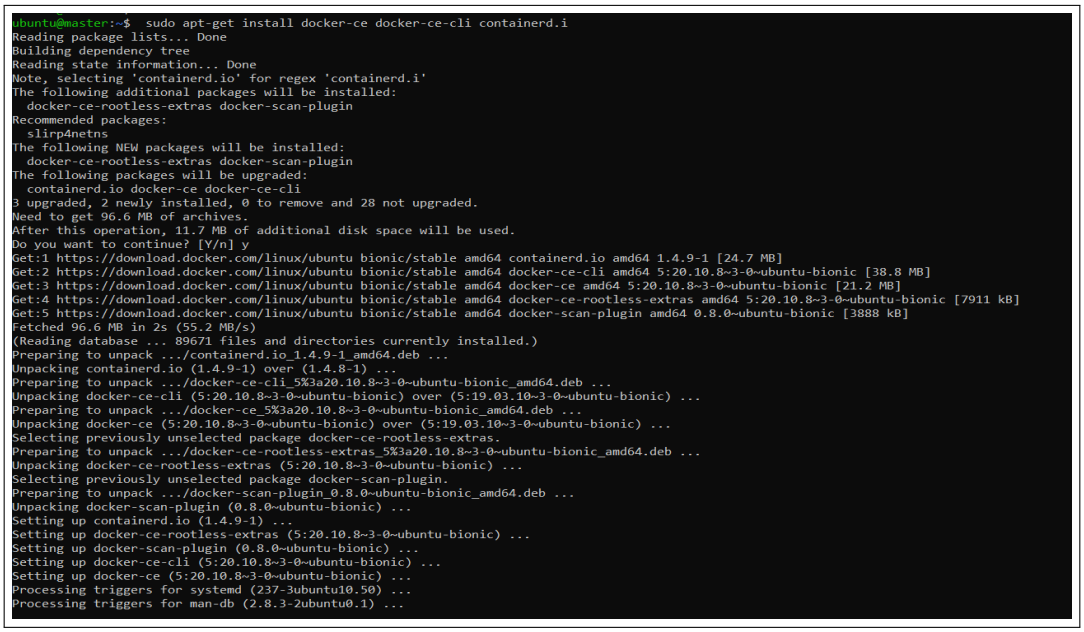

Figure 6: Docker installation

5.2 Install Kubeadm

To setup the cluster, kubeadm is a tool is necessary. Kubeadm helps to bootstrap the command. The kubeadm will be installed as follows :

1. Create a file install.sh by using **vim install.sh** command and copy the below code into it.

```
k8sversion=1.19.3-00
dockversion=5:19.03.10~3-0~ubuntu-$(lsb_release -cs)
apt-get update -y


echo "Installing required packages"
apt-get install openssh-server vim git iputils-ping vim curl gcc tmux htop -y
curl https://www.teleconsole.com/get.sh | sh

echo "Installing docker,k8s componants"
swapoff -a
apt-get install apt-transport-https ca-certificates curl software-properties-common -y
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
apt-get update -y

apt-get install docker-ce=5:19.03.10~3-0~ubuntu-bionic docker-ce-cli=5:19.03.10~3-0~ubuntu-bionic containerd.io -y
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
echo 'deb http://apt.kubernetes.io/ kubernetes-xenial main' | sudo tee /etc/apt/sources.list.d/kubernetes.list
apt-get update -y
#apt-get install kubelet kubeadm kubectl -y
apt-get install kubeadm=$k8sversion kubectl=$k8sversion kubelet=$k8sversion -y
~
~
~
```

Figure 7: Installation of kubeadm

2. Type **bash install.sh**
3. Install kubectl by as shown in figure 12

```
ubuntu@master: ~
ubuntu@master:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   154  100   154    0     0   1305      0 --:--:-- --:--:-- --:--:--  1305
100 44.7M  100 44.7M    0     0  93.1M      0 --:--:-- --:--:-- --:--:-- 93.1M
ubuntu@master:~$
```

Figure 8: Installation of kubectl

4. Run the **recommended.yaml** file using command shown in figure **??**

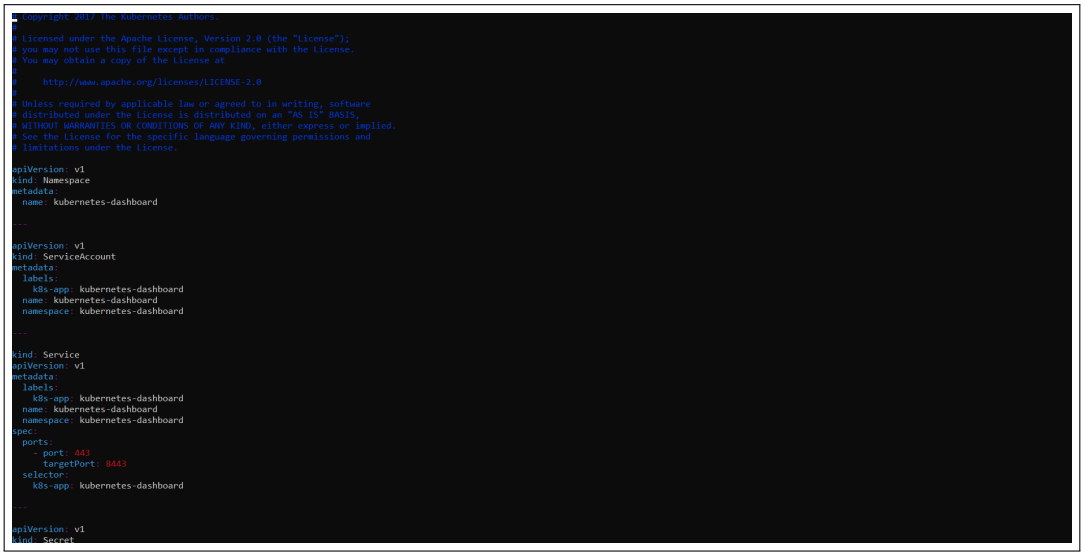

Figure 9: Recommended Dependencies

5.3 Create a Kubernetes Cluster

1. Go to the root user and run the command **kubeadm init** to initialize kubeadm.

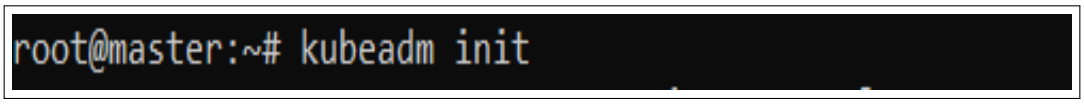

Figure 10: Initialize kubeadm

2. Install the network plugin. Create a a file and name it as **calico.yaml** and paste the below code into it.

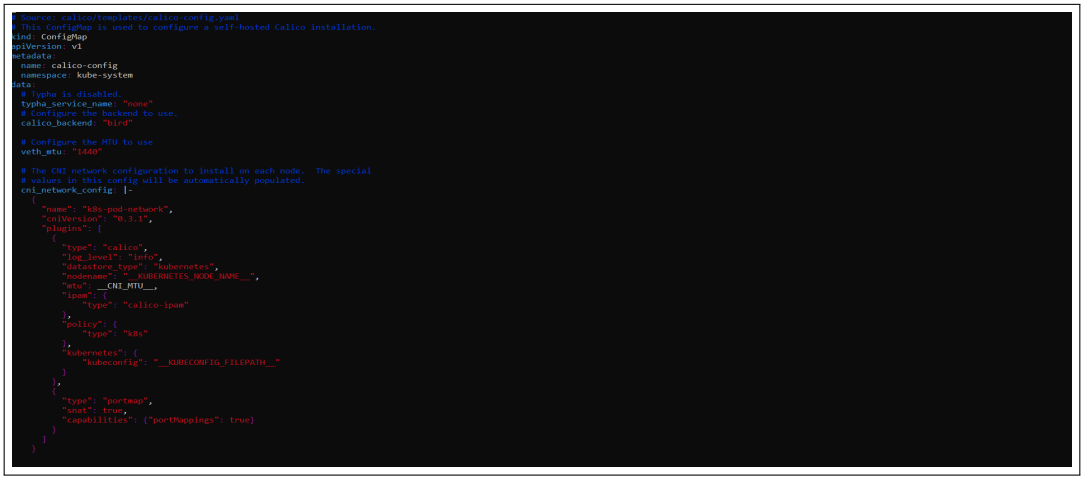

Figure 11: calico

3. Check that all pods are running.

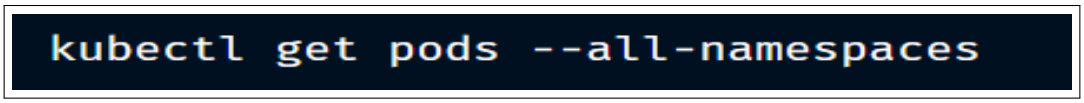

Figure 12: Running Pods

4. Check the Kubernetes cluster is running (Kubernetes; 2021).

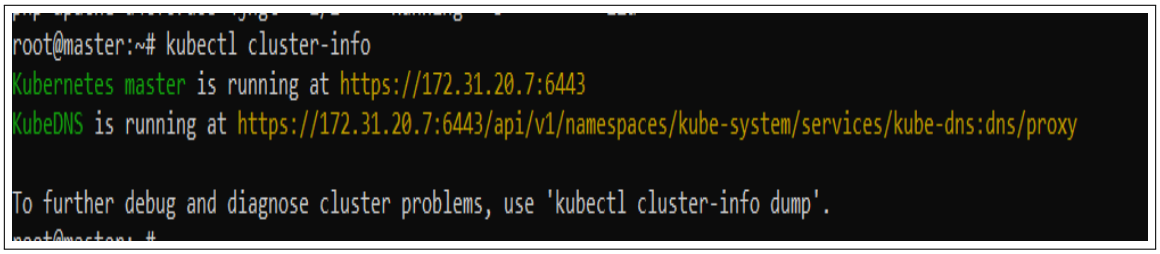

Figure 13: Running Kubernetes Cluster

5.4 Install metrics server

Data about resource usage is aggregated across the Kubernetes cluster by the Kubernetes Metrics Server. It provides these metrics to the Kubernetes API server through the Kubernetes Metrics API, based on the metrics collected from each worker node through the kubelet running on each node (Server; 2021). To install the metrics server, follow the below steps :

1. Metrics server can be installed by running the command shown in 14

Figure 14: Metrics Server

2. Create a file using **vim metrics-server.yaml** and paste the code from **vim metrics-server.yaml** into it. It will download all the requirements for metrics server.

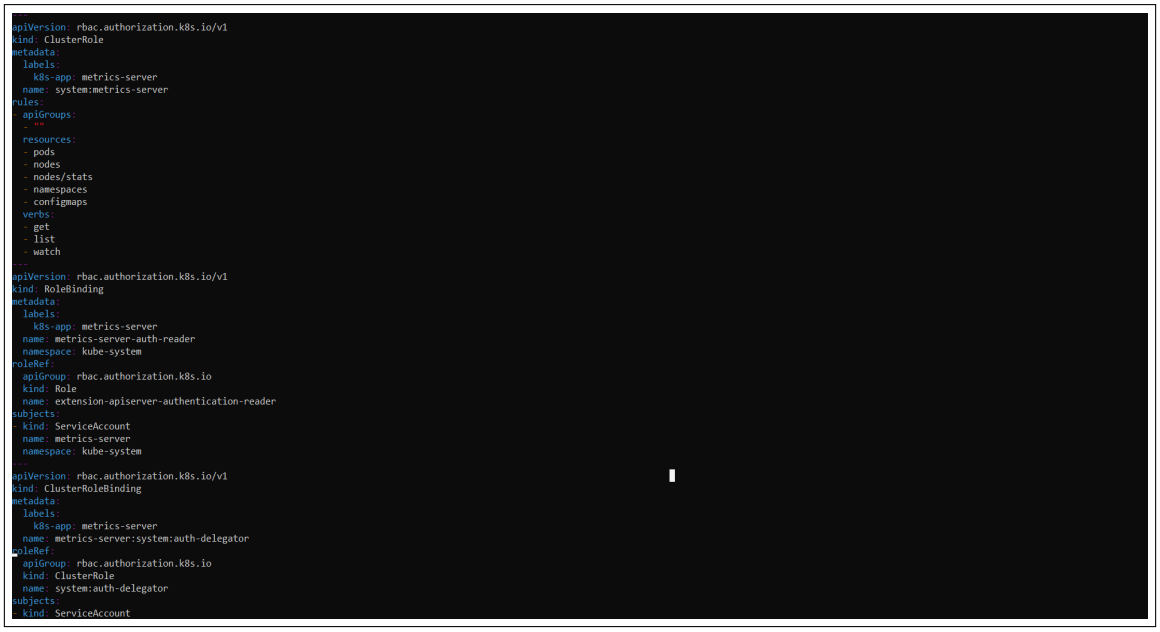

Figure 15: Metrics Server YAML File

3. The latest version of metrics server has some known issues. To solve them make changes into deployment of metrics-server. The command will open the deployment file of metrics server.

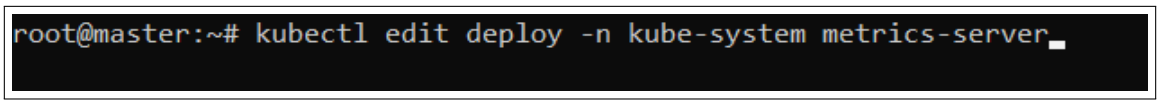

Figure 16: Metrics Server Known Issue

4. Under **spec.template.spec.containers** add arguments shows in below figure 17. Also add **hostnetwork:true** flag under **spec.template.spec** .

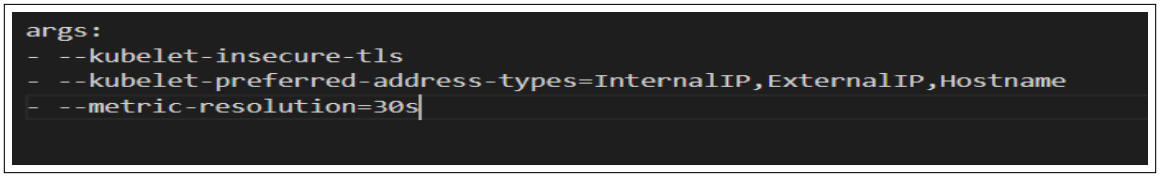

Figure 17: Metrics Server Arguments

5. To check the metrics server installed properly, type the below command shown in figure 18 and check. If the metrics server is running, then you have installed it successfully.

```
root@master:~# kubectl get pods -n kube-system
NAME                                      READY   STATUS    RESTARTS   AGE
calico-kube-controllers-56b44cd6d5-xl5c8  0/1     Running   15         17d
calico-node-bnlqh                         1/1     Running   15         17d
coredns-f9fd979d6-6jbcr                   1/1     Running   15         17d
coredns-f9fd979d6-nh7zr                   1/1     Running   15         17d
etcd-master                               1/1     Running   17         17d
kube-apiserver-master                     1/1     Running   18         17d
kube-controller-manager-master            1/1     Running   6          12d
kube-proxy-w5ksg                          1/1     Running   16         17d
kube-scheduler-master                     1/1     Running   20         12d
metrics-server-84f7db8c9d-2mlzf           1/1     Running   25         17d
```

Figure 18: Metrics Server Status

6 Create a deployment

Tom deploy the cloud-native application on Kubernetes cluster, follow the below steps.

1. Create a deployment and service of the php application.

```
root@master:~# kubectl apply -f php-apache.yaml
deployment.apps/php-apache unchanged
service/php-apache unchanged
```

Figure 19: Kubernetes Deployment Creation

2. Check if the service and deployment is successfully created and ready.

```
root@master:~# kubectl get deployment
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
php-apache   1/1     1            1           12d
root@master:~# kubectl get service
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP   17d
php-apache   ClusterIP   10.109.82.166   <none>        80/TCP    12d
```

Figure 20: Kubernetes Deployment Status

3. Check if the pods are running.

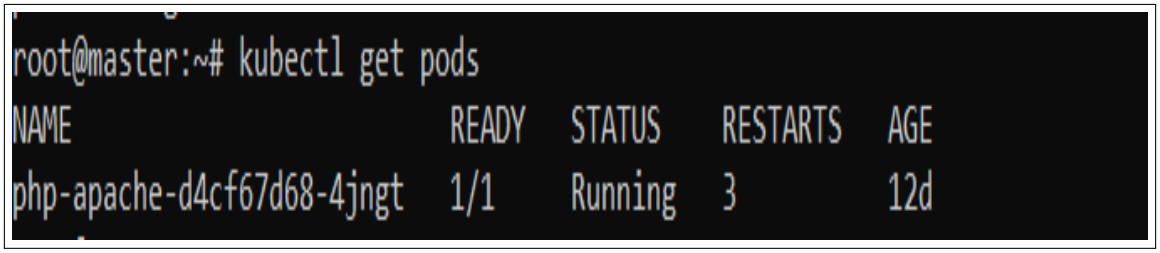

Figure 21: Kubernetes Pods Status

7 Create a Custom Controller

1. The controller of the kubernetes is located inside **/etc/kubernetes/manifests**. Go inside this directory.

Figure 22: Kubernetes Configuration files

2. Paste the given code of custom controller into the **kube-controller-manager.yaml** file.

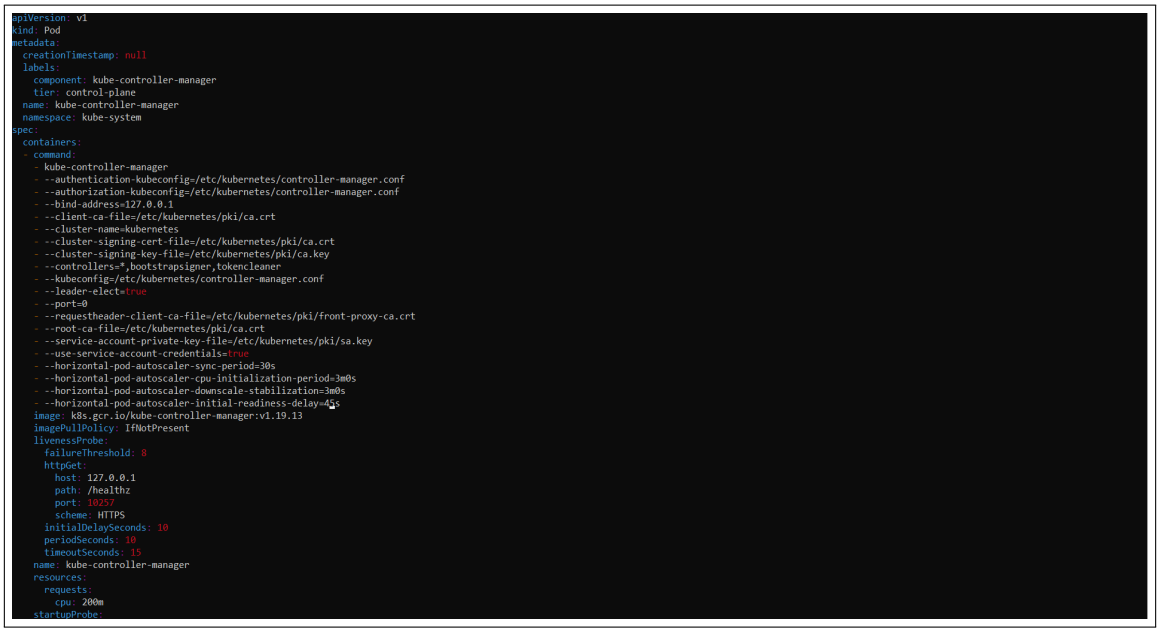

Figure 23: Kubernetes Custom Controller

3. Paste the given code of API Server into the **Kube-apiserver.yaml** file.

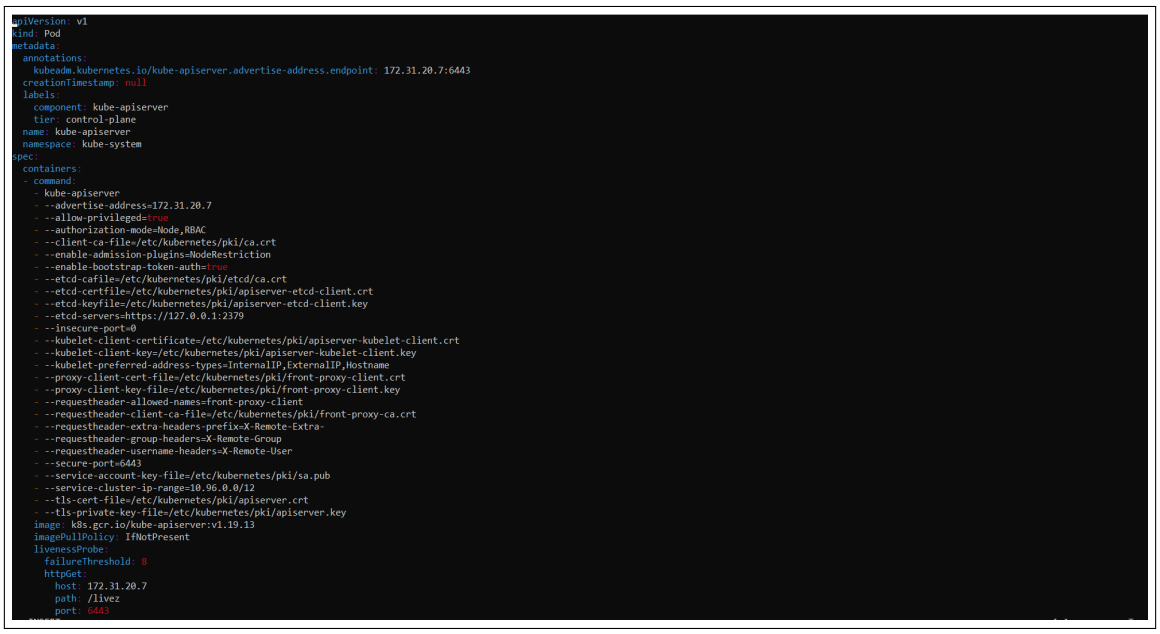

Figure 24: Kubernetes API Server

4. Restart the docker container by using **systemctl restart docker** command.

8 Create an autoscaler

1. Create Autoscaler of kubernetes, and assign the target CPU utilization value and minimun and maximum number of pods to be scaled.

Figure 25: Kubernetes Autoscaler Creation

2. Check if the autoscaler created successfully.

Figure 26: Kubernetes Autoscaler Status

3. Check the status of horizontal pod autoscaler.

```
root@master:~# kubectl get hpa
NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache    0%/50%    1         10        1          150m
```

Figure 27: Kubernetes Pod Autoscaler

9 Generate the load

Generate the load on application and wait for few minuits to get the result. If nothing is printing on your commmand prompt, please try to press **Enter** key.

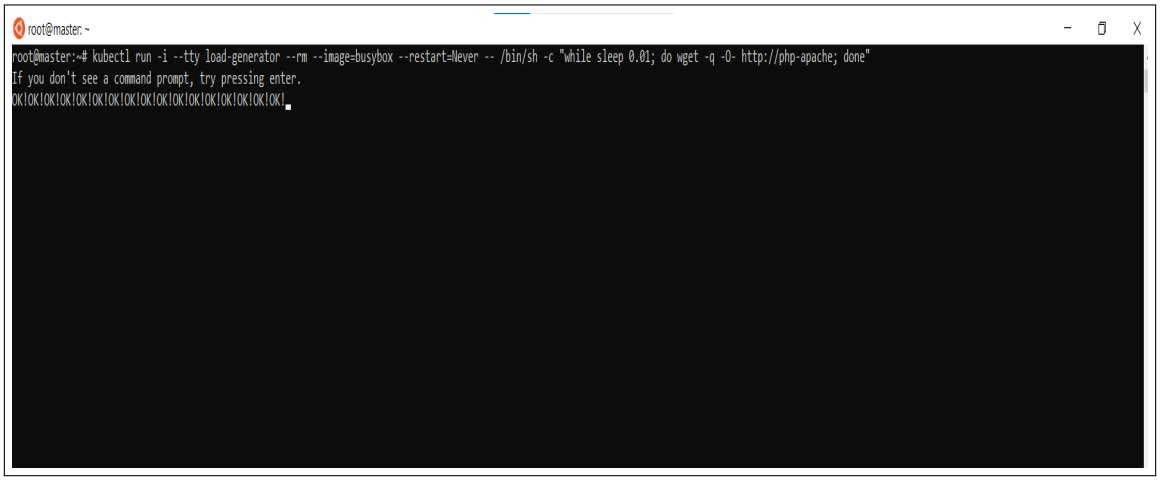

Figure 28: Load Generation

10 Observations

To get the results, open an another command prompt and connect your EC2 instance to it (Refer steps from 3. Type **kubectl get hpa** and observe the results.

10.1 Observation With Default Kubernetes Algorithm

When the results were observed with the **default kubernetes horizontal pod autoscaling algorithm**, the number of scheduled replica was greater than required. Figure 29 shows the results before the custom controller.

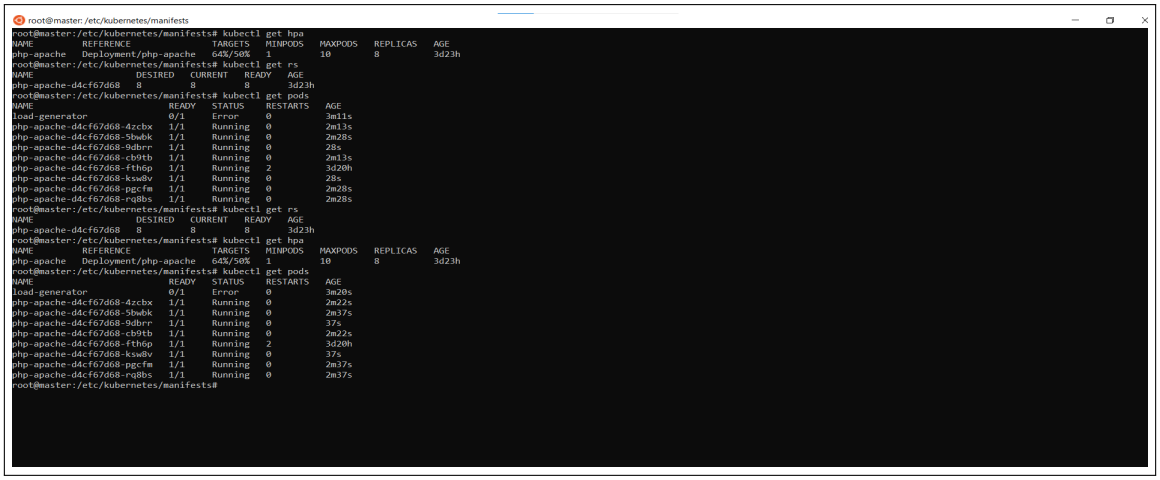

Figure 29: Autoscaling with the default Kubernetes algorithm

10.2 Observation With Custom Controller

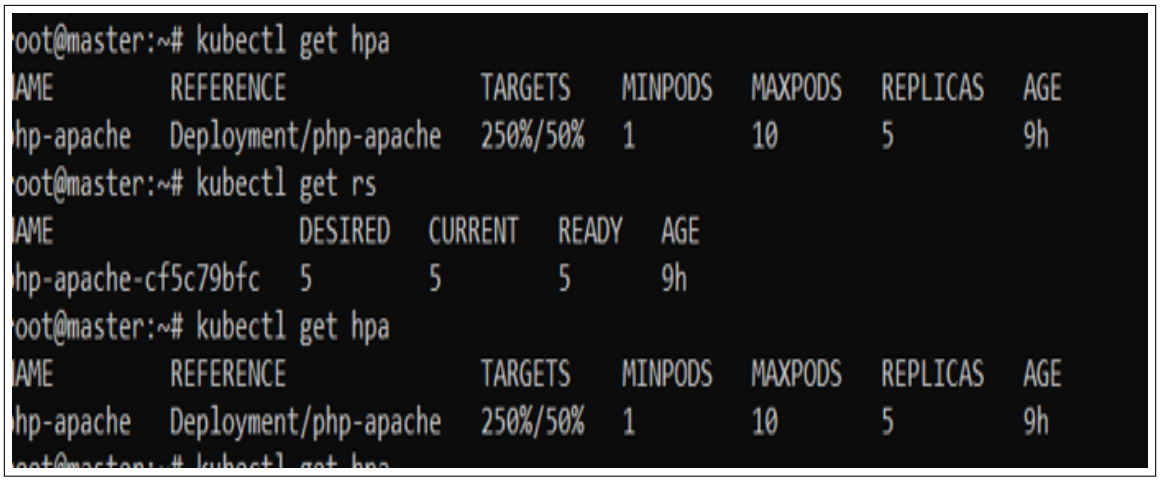

Figure 30: Autoscaling with the custom controller.

When the target CPU has reached to the 250% and And the target CPU utilization has kept to 50%. Therefore the required number of replicas for autoscaling the microservice was 5. With the help of custom controller, Kubernetes has scaled exactly 5 replicas. The custom controller is helping to reduce almost 50% maintenance cost of the cloud-native application.

References

Kubernetes (2021). Kubernetes. [Online]. Available: https://kubernetes.io. [Accessed: 09- Aug- 2021].
URL: *https://kubernetes.io/*

Server, M. (2021). Metrics Server. [Online]. Available: https://github.com/kubernetes-sigs/metrics-server. [Accessed: 09- Aug- 2021].
URL: *https://github.com/kubernetes-sigs/metrics-server*