

# 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



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

<b>Signature:</b>	Neha Deshpande
<b>Date:</b>	15th August 2021

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

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

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

<b>Office Use Only</b>	
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](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.

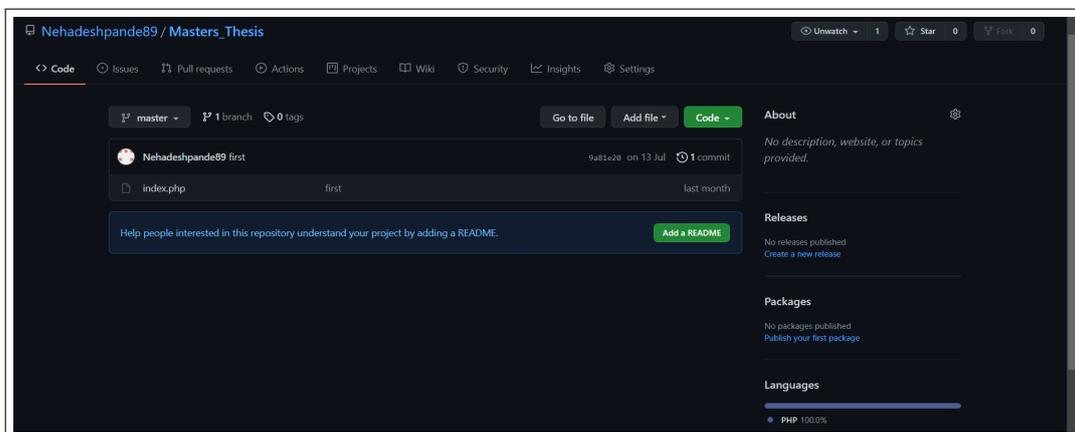


Figure 1: GitHub Repository

## 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.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: php-apache
spec:
  selector:
    matchLabels:
      run: php-apache
  replicas: 1
  template:
    metadata:
      labels:
        run: php-apache
    spec:
      containers:
        - name: php-apache
          image: k8s.pcr.io/hpa-example
          ports:
            - containerPort: 80
          resources:
            limits:
              cpu: 500m
              requests:
                cpu: 200m
---
apiVersion: v1
kind: Service
metadata:
  name: php-apache
labels:
  run: php-apache
spec:
  ports:
    - port: 80
  selector:
    run: php-apache
```

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.

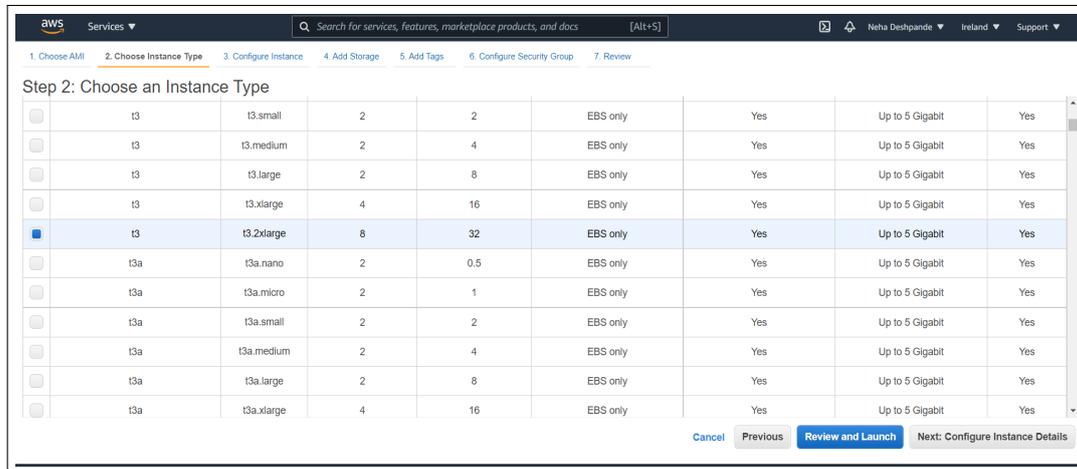


Figure 4: EC2 instance

### 4.1 Security Group for the instance

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

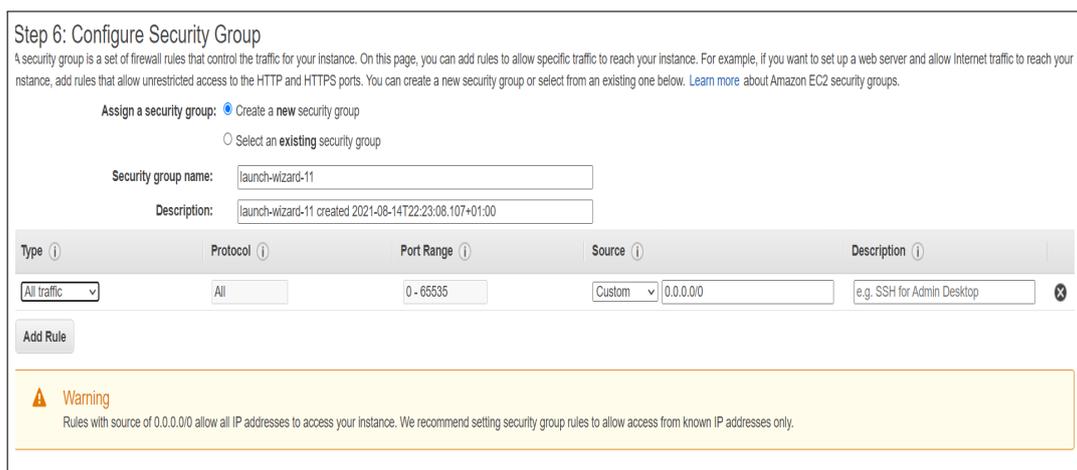


Figure 5: Security group instance

## 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 dependencies 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 :

```
ubuntu@master:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'containerd.io' for regex 'containerd.i'
The following additional packages will be installed:
  docker-ce-rootless-extras docker-scan-plugin
Recommended packages:
  slirp4netns
The following NEW packages will be installed:
  docker-ce-rootless-extras docker-scan-plugin
The following packages will be upgraded:
  containerd.io docker-ce docker-ce-cli
3 upgraded, 2 newly installed, 0 to remove and 28 not upgraded.
Need to get 96.6 MB of archives.
After this operation, 11.7 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
get:1 https://download.docker.com/linux/ubuntu bionic/stable amd64 containerd.io amd64 1.4.9-1 [24.7 MB]
get:2 https://download.docker.com/linux/ubuntu bionic/stable amd64 docker-ce-cli amd64 5:20.10.8-3-0-ubuntu-bionic [38.8 MB]
get:3 https://download.docker.com/linux/ubuntu bionic/stable amd64 docker-ce amd64 5:20.10.8-3-0-ubuntu-bionic [21.2 MB]
get:4 https://download.docker.com/linux/ubuntu bionic/stable amd64 docker-ce-rootless-extras amd64 5:20.10.8-3-0-ubuntu-bionic [7911 kB]
get:5 https://download.docker.com/linux/ubuntu bionic/stable amd64 docker-scan-plugin amd64 0.8.0~ubuntu-bionic [3888 kB]
Fetched 96.6 MB in 2s (55.2 MB/s)
(Reading database ... 89671 files and directories currently installed.)
Preparing to unpack .../containerd.io_1.4.9-1_amd64.deb ...
Unpacking containerd.io (1.4.9-1) over (1.4.8-1) ...
Preparing to unpack .../docker-ce-cli_5%3a20.10.8-3-0-ubuntu-bionic_amd64.deb ...
Unpacking docker-ce-cli (5:20.10.8-3-0-ubuntu-bionic) over (5:19.03.10-3-0-ubuntu-bionic) ...
Preparing to unpack .../docker-ce_5%3a20.10.8-3-0-ubuntu-bionic_amd64.deb ...
Unpacking docker-ce (5:20.10.8-3-0-ubuntu-bionic) over (5:19.03.10-3-0-ubuntu-bionic) ...
Selecting previously unselected package docker-ce-rootless-extras.
Preparing to unpack .../docker-ce-rootless-extras_5%3a20.10.8-3-0-ubuntu-bionic_amd64.deb ...
Unpacking docker-ce-rootless-extras (5:20.10.8-3-0-ubuntu-bionic) ...
Selecting previously unselected package docker-scan-plugin.
Preparing to unpack .../docker-scan-plugin_0.8.0~ubuntu-bionic_amd64.deb ...
Unpacking docker-scan-plugin (0.8.0~ubuntu-bionic) ...
Setting up containerd.io (1.4.9-1) ...
Setting up docker-ce-rootless-extras (5:20.10.8-3-0-ubuntu-bionic) ...
Setting up docker-scan-plugin (0.8.0~ubuntu-bionic) ...
Setting up docker-ce-cli (5:20.10.8-3-0-ubuntu-bionic) ...
Setting up docker-ce (5:20.10.8-3-0-ubuntu-bionic) ...
Processing triggers for systemd (237-3ubuntu10.50) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
```

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

Figure 8: Installation of kubectl

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

```
Copyright 2017 The Kubernetes Authors.
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
    http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

apiVersion: v1
kind: Namespace
metadata:
  name: kubernetes-dashboard
---
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard
---
kind: Service
apiVersion: v1
metadata:
  labels:
    k8s-app: kubernetes-dashboard
  name: kubernetes-dashboard
  namespace: kubernetes-dashboard
spec:
  ports:
    - port: 443
      targetPort: 8443
  selector:
    k8s-app: kubernetes-dashboard
---
apiVersion: v1
kind: Secret
```

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.

```
root@master:~# kubeadm init
```

Figure 10: Initialize kubeadm

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

```
source: https://raw.githubusercontent.com/projectcalico/calico/master/manifests/calico.yaml
# This ConfigMap is used to configure a self-hosted Calico installation.
kind: ConfigMap
apiVersion: v1
metadata:
  name: calico-config
  namespace: kube-system
data:
  # Typha is disabled.
  typha_service_name: "none"
  # Configure the backend to use.
  calico_backend: "bird"
  # Configure the MTU to use
  veth_mtu: "1400"
  # The BGP network configuration to install on each node. The special
  # values in this config will be automatically populated.
  cni_network_config: |-
    {
      "name": "k8s-pod-network",
      "cniVersion": "0.3.1",
      "plugins": [
        {
          "type": "calico",
          "log_level": "info",
          "datastore_type": "kubernetes",
          "nodename": "__KUBERNETES_NODE_NAME__",
          "mtu": __CNI_MTU__,
          "ipam": {
            "type": "calico-ipam"
          },
          "policy": {
            "type": "k8s"
          },
          "kubernetes": {
            "kubeconfig": "__KUBECONFIG_FILEPATH__"
          }
        },
        {
          "type": "portmap",
          "snat": true,
          "capabilities": {"portMappings": true}
        }
      ]
    }

```

Figure 11: calico

3. Check that all pods are running.

```
kubectl get pods --all-namespaces
```

Figure 12: Running Pods

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

```
root@master:~# kubectl cluster-info
Kubernetes master is running at https://172.31.20.7:6443
KubeDNS is running at https://172.31.20.7:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
root@master:~#
```

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

```
root@master:~# kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
```

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.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    k8s-app: metrics-server
  name: system:metrics-server
rules:
  apiGroups:
  - ""
  resources:
  - pods
  - nodes
  - nodes/stats
  - namespaces
  - configmaps
  verbs:
  - get
  - list
  - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  labels:
    k8s-app: metrics-server
  name: metrics-server-auth-reader
  namespace: kube-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: extension-apiserver-authentication-reader
subjects:
  - kind: ServiceAccount
    name: metrics-server
    namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  labels:
    k8s-app: metrics-server
  name: metrics-server:system:auth-delegator
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:auth-delegator
subjects:
  - kind: ServiceAccount
```

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.

```
root@master:~# kubectl edit deploy -n kube-system 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** .

```
args:
  - --kubelet-insecure-tls
  - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
  - --metric-resolution=30s
```

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-x15c8  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
root@master:~#

```

Figure 18: Metrics Server Status

## 6 Create a deployment

To 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.

```
root@master:~# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
php-apache-d4cf67d68-4jngt         1/1     Running   3           12d
```

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.

```
root@master:/etc/kubernetes/manifests# ls
etcd.yaml kube- kube-apiserver.yaml kube-controller-manager.yaml kube-scheduler.yaml
```

Figure 22: Kubernetes Configuration files

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

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    component: kube-controller-manager
    tier: control-plane
  name: kube-controller-manager
  namespace: kube-system
spec:
  containers:
    - command:
      - kube-controller-manager
      - --authentication-kubeconfig=/etc/kubernetes/controller-manager.conf
      - --authorization-kubeconfig=/etc/kubernetes/controller-manager.conf
      - --bind-address=127.0.0.1
      - --client-ca-file=/etc/kubernetes/pki/ca.crt
      - --cluster-name=kubernetes
      - --cluster-signing-cert-file=/etc/kubernetes/pki/ca.crt
      - --cluster-signing-key-file=/etc/kubernetes/pki/ca.key
      - --controllers=*,bootstrapsigner,tokencleaner
      - --kubeconfig=/etc/kubernetes/controller-manager.conf
      - --leader-elect=true
      - --port=0
      - --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
      - --root-ca-file=/etc/kubernetes/pki/ca.crt
      - --service-account-private-key-file=/etc/kubernetes/pki/sa.key
      - --use-service-account-credentials=true
      - --horizontal-pod-autoscaler-sync-period=30s
      - --horizontal-pod-autoscaler-cpu-initialization-period=30s
      - --horizontal-pod-autoscaler-downscale-stabilization=30s
      - --horizontal-pod-autoscaler-initial-readiness-delay=45s
      image: k8s.gcr.io/kube-controller-manager:v1.19.13
      imagePullPolicy: IfNotPresent
      livenessProbe:
        failureThreshold: 8
        httpGet:
          host: 127.0.0.1
          path: /healthz
          port: 10257
          scheme: HTTPS
        initialDelaySeconds: 10
        periodSeconds: 10
        timeoutSeconds: 15
      name: kube-controller-manager
      resources:
        requests:
          cpu: 200m
      startupProbe:
```

Figure 23: Kubernetes Custom Controller

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

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubernetes.io/kube-apiserver.advertise-address.endpoint: 172.31.28.7:6443
  creationTimestamp: null
  labels:
    component: kube-apiserver
  tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
    - command:
      - kube-apiserver
      - --advertise-address=172.31.28.7
      - --allow-privilege-escalation
      - --authorization-mode=Node,RBAC
      - --client-ca-file=/etc/kubernetes/pki/ca.crt
      - --enable-admission-plugins=NodeRestriction
      - --enable-bootstrap-token-auth
      - --etcd-cafiles=/etc/kubernetes/pki/etcd/ca.crt
      - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
      - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
      - --etcd-servers=https://127.0.0.1:2379
      - --insecure-port=0
      - --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
      - --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
      - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
      - --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt
      - --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key
      - --requestheader-allowed-names=front-proxy-client
      - --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
      - --requestheader-extra-headers-prefix=X-Remote-Extra
      - --requestheader-group-headers=X-Remote-Group
      - --requestheader-username-headers=X-Remote-User
      - --secure-port=6443
      - --service-account-key-file=/etc/kubernetes/pki/sa.pub
      - --service-cluster-ip-range=10.96.0.0/12
      - --tls-cert-file=/etc/kubernetes/pki/apiserver.crt
      - --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
    image: k8s.gcr.io/kube-apiserver:v1.19.13
    imagePullPolicy: IfNotPresent
    livenessProbe:
      failureThreshold: 8
      httpGet:
        host: 172.31.28.7
        path: /livez
        port: 6443
```

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 minimum and maximum number of pods to be scaled.

```
root@master:/etc/kubernetes/manifests# kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10
```

Figure 25: Kubernetes Autoscaler Creation

2. Check if the autoscaler created successfully.

```
root@master:~# horizontalpodautoscaler.autoscaling/php-apache autoscaled
```

Figure 26: Kubernetes Autoscaler Status



```

root@master:/etc/kubernetes/manifests# kubectl get hpa
NAME                REFERENCE                TARGETS  MINPODS  MAXPODS  REPLICAS  AGE
php-apache          Deployment/php-apache    64%/50%  1         10        8          3d23h
root@master:/etc/kubernetes/manifests# kubectl get rs
NAME                DESIRED  CURRENT  READY  AGE
php-apache-d4cf67d68  8         8         8       3d23h
root@master:/etc/kubernetes/manifests# kubectl get pods
NAME                READY  STATUS   RESTARTS  AGE
load-generator      0/2    Error    0          2m11s
php-apache-d4cf67d68-4zcbx  1/1    Running  0          2m13s
php-apache-d4cf67d68-5bwkx  1/1    Running  0          2m28s
php-apache-d4cf67d68-9dbrn  1/1    Running  0          2s
php-apache-d4cf67d68-cb9tb  1/1    Running  0          2m13s
php-apache-d4cf67d68-fthep  1/1    Running  2          3d20h
php-apache-d4cf67d68-kxwbu  1/1    Running  0          2s
php-apache-d4cf67d68-pgcfa  1/1    Running  0          2m28s
php-apache-d4cf67d68-rq8bs  1/1    Running  0          2m28s
root@master:/etc/kubernetes/manifests# kubectl get rs
NAME                DESIRED  CURRENT  READY  AGE
php-apache-d4cf67d68  8         8         8       3d23h
root@master:/etc/kubernetes/manifests# kubectl get hpa
NAME                REFERENCE                TARGETS  MINPODS  MAXPODS  REPLICAS  AGE
php-apache          Deployment/php-apache    64%/50%  1         10        8          3d23h
root@master:/etc/kubernetes/manifests# kubectl get pods
NAME                READY  STATUS   RESTARTS  AGE
load-generator      0/2    Error    0          2m20s
php-apache-d4cf67d68-4zcbx  1/1    Running  0          2m22s
php-apache-d4cf67d68-5bwkx  1/1    Running  0          2m37s
php-apache-d4cf67d68-9dbrn  1/1    Running  0          37s
php-apache-d4cf67d68-cb9tb  1/1    Running  0          2m22s
php-apache-d4cf67d68-fthep  1/1    Running  2          3d20h
php-apache-d4cf67d68-kxwbu  1/1    Running  0          37s
php-apache-d4cf67d68-pgcfa  1/1    Running  0          2m37s
php-apache-d4cf67d68-rq8bs  1/1    Running  0          2m37s
root@master:/etc/kubernetes/manifests#

```

Figure 29: Autoscaling with the default Kubernetes algorithm

## 10.2 Observation With Custom Controller

```

root@master:~# kubectl get hpa
NAME                REFERENCE                TARGETS  MINPODS  MAXPODS  REPLICAS  AGE
hp-apache          Deployment/php-apache    250%/50%  1         10        5          9h
root@master:~# kubectl get rs
NAME                DESIRED  CURRENT  READY  AGE
hp-apache-cf5c79bfc  5         5         5       9h
root@master:~# kubectl get hpa
NAME                REFERENCE                TARGETS  MINPODS  MAXPODS  REPLICAS  AGE
hp-apache          Deployment/php-apache    250%/50%  1         10        5          9h
root@master:~# kubectl get hpa

```

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>*