# DEVOPS CAPSTONE PROJECT
# PROJECT 1 – INFRA OPTIMIZATION

WRITTEN BY: TAN KAH XUAN

LAST UPDATED: 2021-11-24

# Contents


Introduction ..... 2

Project 1 – Infra Optimization ..... 2

Task 1 – Provisioning of EC2 instances with Ansible ..... 3

Task 2 – Setting up a Kubenetes Cluster ..... 6

Task 3 – Configuration of Kubenetes Clusters for Web Application ..... 11

Setup Load Balancer ..... 11

Setup MongoDB Database ..... 11

Setup Web Application ..... 15

Task 4 – Create Users in the Kubrnetes Cluster ..... 20

Task 5 – Backup ETCD cluster data ..... 21

Task 6 – Auto Scaling ..... 22

Appendix – To reset kuberneter cluster ..... 23

## Introduction

### Project 1 – Infra Optimization

Create a DevOps infrastructure for an e-commerce application to run on high-availability mode.

**Background of the problem statement:**

A popular payment application, EasyPay where users add money to their wallet accounts, faces an issue in its payment success rate. The timeout that occurs with the connectivity of the database has been the reason for the issue. While troubleshooting, it is found that the database server has several downtime instances at irregular intervals. This situation compels the company to create their own infrastructure that runs in high-availability mode.

Given that online shopping experiences continue to evolve as per customer expectations, the developers are driven to make their app more reliable, fast, and secure for improving the performance of the current system.

**Implementation requirements:**

1. Create the cluster (EC2 instances with load balancer and elastic IP in case of AWS)
2. Automate the provisioning of an EC2 instance using Ansible or Chef Puppet
3. Install Docker and Kubernetes on the cluster
4. Implement the network policies at the database pod to allow ingress traffic from the front-end application pod
5. Create a new user with permissions to create, list, get, update, and delete pods
6. Configure application on the pod
7. Take snapshot of ETCD database
8. Set criteria such that if the memory of CPU goes beyond 50%, environments automatically get scaled up and configured

**Project Requirement Checklist**

The Project will be splitted into 6 major tasks as listed below. The following tools will be involved in the setting up the project.

1. Ansible – The software provisioning tool for enabling infrastrucutre as code
2. AWS EC2 – The ubuntu environment nodes used to setup the Kubenetes Clusters
3. Docker – The container orchestration tool
4. Kubenetes – The containerized application management tool
5. Github – Source Code Repository

| S/N | Tasks | Status |
|---|---|---|
| 1 | Provisioning 3 x AWS EC2 instance (with load balancer and elastic IP) with Ansible<br>• 1 x Kubenetes Master Node<br>• 2 x Kubenetes Worker Node | Done |
| 2 | Install Docker and setup a Kubenetes Cluster | Done |
| 3 | Configuration of Kubenetes Cluster for Web Application<br>• Create Database Pod – MongoDB / Elasticsearch<br>• Create Web Application Pod – NodeJS backend application<br>• Implement relevant network polices at the database pod to allow ingress traffic from front-end application pod | Done |
| 4 | Configuration of the Pods<br>• Create a new user with permissions to create, list, get, update, and delete pods | Done |
| 5 | Taking a snapshot of ETCD database | Done |
| 6 | Auto Scaling – If the memory of CPU goes beyong 50%, environments automatically get scaled and configured | Done |

## Task 1 – Provisioning of EC2 instances with Ansible

**Create new Ubuntu EC2 instance**

The EC2 instance will be used to host ansible. There are several steps to prepare the EC2 instance.

1.a.1 Create AWS user

On the AWS console, search for IAM (Identity and Access Management) and create a new user. Here, take note of the name of the key pair generated which will be used by Ansible to setup the instances.

Create an IAM role with AmazonEC2FullAccess policy.

1.a.2 Provision an EC2 Instance

The t2.mirco instance is lauched. Attach the IAM Role with AmazonEC2FullAccess to the EC2 instance. Open port 22 for SSH.

**Install ansible**

1.b.1 Login to the EC2 instance using Putty/ MobaXterm or othr tools to SSH into the ubuntu machine. Execute the following codes to install ansible on the ubuntu machine:

# Install Ansible

```
$ sudo apt-get install -y ansible

$ sudo apt-get install python-pip -y
```

# Install the python AWS SDK – Boto Framework

```
$ sudo pip install boto boto3

$ sudo apt-get install python-boto -y
```

Ansible will be using boto SDK to access various AWS resources.

# Verify ansible installation

```
$ ansible --version
```

**Create Ansible Playbook**

1.c.1 Edit Ansible hosts or create an inventory file

```
$ sudo nano /etc/ansible/hosts
```

Add the contents at the end of the file:

```
[localhost]
localhost
```

1.c.2 Create Ansible playbook

```
$ mkdir playbook
$ cd playbook

$ sudo nano create_ec2.yml
```

```
---
 - name:  provisioning EC2 instances using Ansible
   hosts: localhost
   connection: local
   gather_facts: False
   tags: provisioning

   vars:
```

```
    keypair: 06Dec
    instance_type: t2.micro
    image: ami-0907c2c44ea451f84
    wait: yes
    group: k8cluster
    count: 3
    region: ap-southeast-1
    zones: ap-southeast-1b
    security_group: k8cluster-security-group

  tasks:

    - name: Create security group
      local_action:
        module: ec2_group
        name: "{{ security_group }}"
        description: Security Group for webserver Servers
        region: "{{ region }}"
        rules:
           - proto: tcp
             from_port: 22
             to_port: 22
             cidr_ip: 0.0.0.0/0
           - proto: tcp
             from_port: 8080
             to_port: 8080
             cidr_ip: 0.0.0.0/0
           - proto: tcp
             from_port: 80
             to_port: 80
             cidr_ip: 0.0.0.0/0
        rules_egress:
           - proto: all
             cidr_ip: 0.0.0.0/0
      register: basic_firewall

    - name: Launch the new EC2 Instance
      local_action:  ec2
                     group={{ security_group }}
                     instance_type={{ instance_type}}
                     image={{ image }}
                     wait=true
                     region={{ region }}
                     keypair={{ keypair }}
                     count={{count}}
      register: ec2

    - name: Add Tagging to EC2 instance
      local_action:
        ec2_tag resource={{ item.id }}
        region={{ region }}
        state=present
      with_items: "{{ ec2.instances }}"
      args:
        tags:
          Name: k8cluster

    - name: associate new elastic IPs with each of the instances
      ec2_eip:
       device_id: "{{ item }}"
       region: "{{ region }}"
      with_items: "{{ ec2.instance_ids }}"

    - name: setup a simple load balancer
      ec2_elb_lb:
       name: myelb
       state: present
```

```
        region: "{{ region }}"
        zones:
          - "{{ zones }}"
        listeners:
          - protocol: http
            load_balancer_port: 80
            instance_port: 80
      register: myelb

    - name: add the webservers to the load balancer
      local_action: ec2_elb
      args:
        instance_id: "{{ item }}"
        ec2_elbs: myelb
        state: present
        region: "{{ region }}"
      with_items: "{{ ec2.instance_ids }}"
```

Change content Highlighed in yellow accordingly.

1.c.3 Execute the playbook

```
$ sudo ansible-playbook create_ec2.yml
```

Expected output:

```
TASK [Create security group] ***************************************************************************************************
/usr/lib/python2.7/dist-packages/requests/__init__.py:80: RequestsDependencyWarning: urllib3 (1.26.7) or chardet (3.0.4) doesn't match a
 supported version!
  RequestsDependencyWarning)
[DEPRECATION WARNING]: Distribution Ubuntu 18.04 on host localhost should use /usr/bin/python3, but is using /usr/bin/python for
backward compatibility with prior Ansible releases. A future Ansible release will default to using the discovered platform python for
this host. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information. This feature
will be removed in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
ok: [localhost -> localhost]

TASK [Launch the new EC2 Instance] *********************************************************************************************
/usr/lib/python2.7/dist-packages/requests/__init__.py:80: RequestsDependencyWarning: urllib3 (1.26.7) or chardet (3.0.4) doesn't match a
 supported version!
  RequestsDependencyWarning)
changed: [localhost -> localhost]
/usr/lib/python2.7/dist-packages/requests/__init__.py:80: RequestsDependencyWarning: urllib3 (1.26.7) or chardet (3.0.4) doesn't match a
 supported version!
  RequestsDependencyWarning)

TASK [Add Tagging to EC2 instance] *********************************************************************************************
changed: [localhost -> localhost] => (item={u'ramdisk': None, u'kernel': None, u'root_device_type': u'ebs', u'private_dns_name': u'ip-17
2-31-30-103.ap-southeast-1.compute.internal', u'block_device_mapping': {u'/dev/sda1': {u'status': u'attached', u'delete_on_termination':
 True, u'volume_id': u'vol-0dd8ede9bac0a5ffc'}}, u'key_name': u'06Dec', u'public_ip': u'18.141.164.125', u'image_id': u'ami-0907c2c44ea4
51f84', u'tenancy': u'default', u'private_ip': u'172.31.30.103', u'groups': {u'sg-0496dfb419b1e463c': u'k8cluster-security-group'}, u'pu
blic_dns_name': u'ec2-18-141-164-125.ap-southeast-1.compute.amazonaws.com', u'state_code': 16, u'id': u'i-08f4444cd41a430ec', u'tags': {
}, u'placement': u'ap-southeast-1b', u'ami_launch_index': u'1', u'dns_name': u'ec2-18-141-164-125.ap-southeast-1.compute.amazonaws.com',
 u'region': u'ap-southeast-1', u'ebs_optimized': False, u'launch_time': u'2021-12-06T15:15:32.000Z', u'instance_type': u't2.micro', u'st
ate': u'running', u'root_device_name': u'/dev/sda1', u'hypervisor': u'xen', u'virtualization_type': u'hvm', u'architecture': u'x86_64'})
changed: [localhost -> localhost] => (item={u'ramdisk': None, u'kernel': None, u'root_device_type': u'ebs', u'private_dns_name': u'ip-17
2-31-21-6.ap-southeast-1.compute.internal', u'block_device_mapping': {u'/dev/sda1': {u'status': u'attached', u'delete_on_termination': T
rue, u'volume_id': u'vol-058e0869759a04b05'}}, u'key_name': u'06Dec', u'public_ip': u'3.1.5.191', u'image_id': u'ami-0907c2c44ea451f84',
 u'tenancy': u'default', u'private_ip': u'172.31.21.6', u'groups': {u'sg-0496dfb419b1e463c': u'k8cluster-security-group'}, u'public_dns_
name': u'ec2-3-1-5-191.ap-southeast-1.compute.amazonaws.com', u'state_code': 16, u'id': u'i-0cbc1e230a298f093', u'tags': {}, u'placement
': u'ap-southeast-1b', u'ami_launch_index': u'0', u'dns_name': u'ec2-3-1-5-191.ap-southeast-1.compute.amazonaws.com', u'region': u'ap-so
utheast-1', u'ebs_optimized': False, u'launch_time': u'2021-12-06T15:15:32.000Z', u'instance_type': u't2.micro', u'state': u'running', u
'root_device_name': u'/dev/sda1', u'hypervisor': u'xen', u'virtualization_type': u'hvm', u'architecture': u'x86_64'})

TASK [associate new elastic IPs with each of the instances] ********************************************************************
changed: [localhost] => (item=i-08f4444cd41a430ec)
changed: [localhost] => (item=i-0cbc1e230a298f093)

TASK [setup a simple load balancer] ********************************************************************************************
ok: [localhost]
```

1.c.4 On the AWS Console, search for the ec2 service.

Expected output:

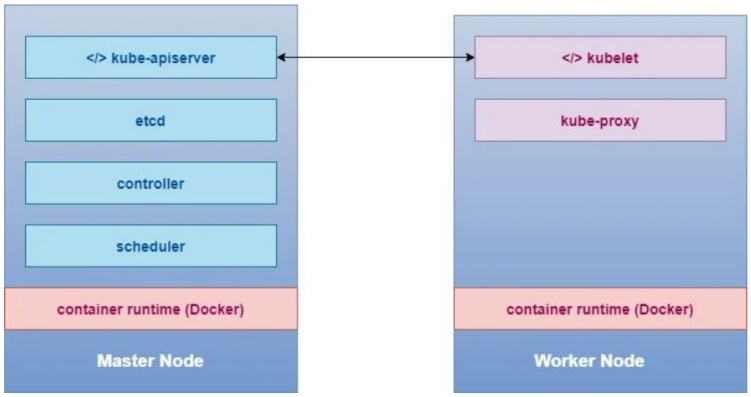
## Task 2 – Setting up a Kubenetes Cluster

Due to the cost involed in the EC2 instances on AWS, the Kubenestes Cluster is setup using 3 ubuntu machine  on VirtualBox.

The master node: 192.168.56.51
The worker1 node: 192.168.56.52
The worker2 node: 192.168.56.53

Each node is connected to the NAT network adapter for internet access and Host-Only network adaptor for communication between the Host Machine and among the three nodes.

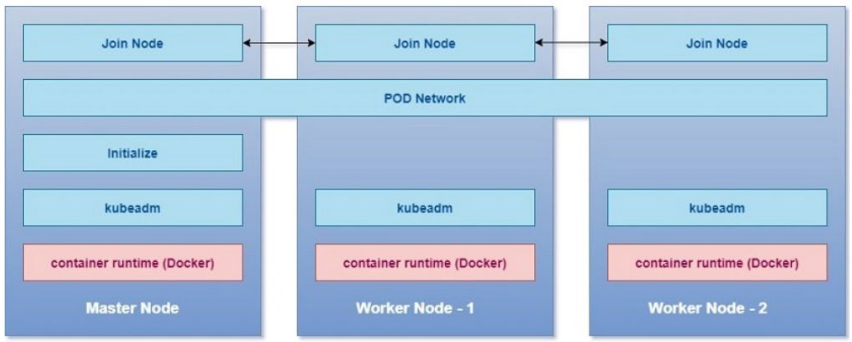
In general, we will have the following setup for the master node and all the worker nodes.




**General Steps**

1) Have multiple nodes to be formed as a kubenestes cluster
2) Install a container runtime on all nodes, here we are using Docker
3) Install kubeadmn on all the nodes
4) Initailize the master server
5) Create a POD Network (or cluster network) for communication between the master node and all worker node
6) For all worker nodes, join node to the master node

Following the steps, we will have the setup as shown below. From step 1 (bottom) to step 6 (Top).




| **Pre-requisities** | |
|---|---|
| **Step** | **Description** |
| **1.a** | <u>Install a Container Runtime – Docker CE</u><br>On all the nodes, we need to install a container runtime such as Docker.<br>** Repeat this for each nodes<br><br><u>Installing docker</u><br>`$ sudo apt-get update`<br>`$ sudo apt-get install -y docker.ce` |

To verify the installation:

```
$ docker --version
```

**1.b**

Configuring Docker

Create a new file `/etc/docker/daemon.json` and add the following content in the file.

```
$ sudo vi /etc/docker/daemon.json
```

```
{
      "exec-opts": ["native.cgroupdriver=systemd"]
}
```

*daemon.json*

**On the master node: Setup a local registry

```
docker run -d -p 5000:5000 --restart=always --name registry registry:2
```

```
$ sudo nano /etc/docker/daemon.json
```

```
{
      "exec-opts": ["native.cgroupdriver=systemd"],
       "insecure-registries":["192.168.56.51:5000"]
}
```

*daemon.json*

**1.c**

Restart Docker Service

```
$ service docker restart
$ service docker status
```

**2.a**

Disable Swap

It is required to disable swap memory for `kubelet` to work properly.

** Repeat this for each nodes

```
$ sudo swapoff -a
```

Output:

```
ubuntu-master@ubuntumaster:~$ free -m
              total        used        free      shared  buff/cache   available
Mem:           1979         519          99           7        1360        1312
Swap:             0           0           0
```

**2.b**

To ensure that swap is not re-assigned afer node reboot, we need to comment out the swap filesystem entry from `/etc/fstab`

```
#/dev/mapper/rhel-swap   swap                    swap    defaults        0 0
```

**3.a**

Disable SELinux

We need to disable selinux or change it for Permissive mode to allow containers to access the host filesystem which is needed by pod networks.

** Repeat this for each nodes

```
$ setenforce 0
```

To verify:

```
$ getenforce
```

Output: `Permissive`

**3.b**

To make the changes persistent after reboot:

```
$ sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g'
/etc/sysconfig/selinux
```

**4.a**

Enable Firewall

We also need to enable certain pre-defined ports on the master and worker nodes.

** On the Master Node

Requries all workers nodes to access the ports

| Ports | Purpose |
|---|---|
| 6443/tcp | For Kubernetes API |
| 6783/tcp | For weave-net |
| 6783/udp | For weave-net |
| 6784/udp | For weave-net |
| 2379/tcp – 2380/tp | For etcd server client API |
| 10250/tcp | For Kubelet API |
| 10251/tcp | For kube-scheduler |
| 10252tcp | For kube-controller manager |

** On all Worker Nodes

Requries all other nodes to access the ports

| Ports | Purpose |
|---|---|
| 6783/tcp | For weave-net |
| 6783/udp | For weave-net |
| 6784/udp | For weave-net |

**Install Kubernetes Component**

You will install these packages on all of your machines:

- **kubeadm**: the command to bootstrap the cluster.
- **kubelet**: the component that runs on all of the machines in your cluster and does things like starting pods and containers.
- **kubectl**: the command line util to talk to your cluster.

**On the Master Node**

| Step | Description |
|---|---|
| 2.a.1 | **Install Kubernetes Component** |

```
$ sudo apt-get update

$ sudo apt-get install -y apt-transport-https ca-certificates curl

$ sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg

$ echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list

$ sudo apt-get update

$ sudo apt-get install -y kubelet kubeadm kubectl
```

To verfiy:

```
$ kubeadm version -o short
```

Output:

```
ubuntu-master@ubuntumaster:~$ kubeadm version -o short
v1.22.3
```

```
$  kubectl version --short --client
```

Output:

```
ubuntu-master@ubuntumaster:~$ kubectl version –short --client
Client Version: version.Info{Major:"1", Minor:"22", GitVersion:"v1.22.3", GitCommit:"c92036820499fedefec0f847e20
54d824aea6cd1", GitTreeState:"clean", BuildDate:"2021-10-27T18:41:28Z", GoVersion:"go1.16.9", Compiler:"gc", Pla
tform:"linux/amd64"}
```

2.a.2 **Initialize Master Node**

```
$   kubeadm init --apiserver-advertise-address 192.168.56.51 --pod-network-
cidr=10.244.0.0/16
```

Output:

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 10.0.2.4:6443 --token 4wkd19.res0ndk41x38nc5x \
        --discovery-token-ca-cert-hash sha256:1a61c915a43d77a17ccaebd00fbe548a557e06efb09aedaf32d000ed3d121f1e
```

Then execute the following commands:

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

```
ubuntu-master@ubuntumaster:~$ kubectl get nodes
NAME           STATUS   ROLES                  AGE    VERSION
ubuntumaster   Ready    control-plane,master   6m9s   v1.22.3
```

**Note**: You should now see a single node **master** deployed

- Copy the **kubeadmn join command** that you can see on the screen of your master node after running **kubeadm init** command
- Run the **kubeadm join** command on every worker nodes as a root user to form the kubernetes cluster

2.a.3 **Apply the weave-net**

```
$  kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl
version | base64 | tr -d '\n')"
```

Output:

```
ubuntu-master@ubuntumaster:~$ kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
```

2.a.4 **Verfiy the setup**

```
$  kubectl get nodes -o wide
```

| | |
|---|---|
| | Output:<br>`ubuntu-master@ubuntumaster:~$ kubectl get nodes -o wide`<br>`NAME STATUS ROLES AGE VERSION INTERNAL-IP EXTERNAL-IP OS-IMAGE KERNEL-VERSION CONTAINER-RUNTIME`<br>`ubuntumaster Ready control-plane,master 3m49s v1.22.3 10.0.2.4 <none> Ubuntu 20.04.3 LTS 5.11.0-40-generic docker://20.10.7` |

## On all Worker Nodes

| Step | Description |
|---|---|
| 2.b.1 | **Install Kubernetes Component** |

```
$ sudo apt-get update

$ sudo apt-get install -y apt-transport-https ca-certificates curl

$ sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg

$ echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list

$ sudo apt-get update

$ sudo apt-get install -y kubelet kubeadm kubectl
```

| | |
|---|---|
| 2.b.2 | **Join Worker Node to Master node** |

```
$ sudo kubeadm join 10.0.2.4:6443 --token 4wkd19.res0ndk41x38nc5x \
        --discovery-token-ca-cert-hash
sha256:1a61c915a43d77a17ccaebd00fbe548a557e06efb09aedaf32d000ed3d121f1e
```

Note: Execute the kubeadm join command obtain from your screen in step 5b

**Output:**

```
ubuntu-worker1@ubuntuworker1:~$ sudo kubeadm join 10.0.2.4:6443 --token 4wkd19.res0ndk41x38nc5x         --discovery-token-ca-cert-hash sha256:1a61c915a43d77a1
7ccaebd00fbe548a557e06efb09aedaf32d000ed3d121f1e
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

| | |
|---|---|
| 2.b.3 | **To verify setup** |

On the **master** node:

```
$  kubectl get nodes -o wide
```

Output:

```
ubuntu-master@ubuntumaster:~$ kubectl get nodes -o wide
NAME           STATUS   ROLES                  AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
ubuntumaster   Ready    control-plane,master   74s   v1.22.3   10.0.2.101    <none>        Ubuntu 20.04.3 LTS   5.11.0-43-generic   docker://20.10.7
ubuntuworker1  Ready    <none>                 27s   v1.22.3   10.0.2.102    <none>        Ubuntu 20.04.3 LTS   5.11.0-43-generic   docker://20.10.7
ubuntuworker2  Ready    <none>                 22s   v1.22.3   10.0.2.103    <none>        Ubuntu 20.04.3 LTS   5.11.0-41-generic   docker://20.10.7
```

**Note that now the worker node has been added. Repeat this step for all other worker nodes**

## Task 3 – Configuration of Kubenetes Clusters for Web Application

### Setup Load Balancer

**Setup MetalLB Load Balancing for Bare Metal Kubernetes**

**Skip this section if the Kubernetes is deployed on any cloud service provider with their own load balancer. MetalLB is one of the load balancer solution for our kubernetes cluster provision on-prem.

Reference: https://metallb.universe.tf/installation/

3.a.1 Installation by Manifest

To install MetalLB by applying the manifest

```
$ kubectl apply -f
https://raw.githubusercontent.com/metallb/metallb/v0.11.0/manifests/namespace.yaml

$ kubectl apply -f
https://raw.githubusercontent.com/metallb/metallb/v0.11.0/manifests/metallb.yaml
```

3.a.2 To deploy metalLB using a ConfigMap by the Layer2 configuration

```
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    address-pools:
    - name: default
      protocol: layer2
      addresses:
      - 192.168.56.0-192.168.56.50
```

*loadBalance.yaml*

The ip address must be a range of IP addresses that is from the same network with the kubernetes cluster. We need to serve this range of IP addresss for load balancing.

3.a.3 Apply the configMap

```
$ kubectl apply -f loadBalance.yaml
```

### Setup MongoDB Database

**Database** - Deploy MongoDB

3.b.1 Create Storage Class

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard
provisioner: local
parameters:
        type: standard
reclaimPolicy: Retain
allowVolumeExpansion: true
```

*storage_class.yaml*

```
$ kubectl apply -f storage_class.yaml
```

To verify:

```
$ kubectl get sc
```

Expected Output:

```
ubuntu-master@ubuntumaster:~$ kubectl get sc
NAME       PROVISIONER   RECLAIMPOLICY   VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
standard   local         Retain          Immediate           true                   62m
```

3.b.2 Create a persistence volume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume
  labels:
    type: local
spec:
  storageClassName: standard
  capacity:
    storage: 50Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
```

persistence_vol.yaml

```
$ kubectl apply -f persistence_vol.yaml
```

To verfiy:

```
$ kubectl get pv
```

```
ubuntu-master@ubuntumaster:~$ kubectl get pv
NAME             CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM         STORAGECLASS   REASON   AGE
task-pv-volume   50Gi       RWO            Retain           Bound    default/pvc   standard                64m
```

3.b.3 Create a MongoDB service

Create the mongodb-service.yaml below:

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: mongo
  name: mongo-nodeport-svc
spec:
  ports:
  - port: 27017
    protocol: TCP
    targetPort: 27017
    nodePort: 32000
  selector:
    app: mongo
```

*mongo-service.yaml*

On the same directory, execute the command:

```
$ kubectl apply -f mongodb-service.yaml
```

To verify:

```
$ kubectl get services
```

```
ubuntu-master@ubuntumaster:~$ kubectl get services
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)           AGE
kubernetes          ClusterIP      10.96.0.1       <none>          443/TCP           71m
mongo-nodeport-svc  LoadBalancer   10.97.135.205   192.168.56.10   27017:32000/TCP   56m
```

With the load balance in place, the mongoDB can be access from other remote machines on the EXTERNAL_IP **192.168.56.10**

3.b.4 Create a persistent volume claim for mongoDB

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc
spec:
  storageClassName: "standard"
  accessModes:
    - ReadWriteOnce
  volumeName: task-pv-volume
  resources:
    requests:
      storage: 1Gi
```

*mongodb-pvc.yaml*

To apply:

```
$ kubectl apply -f mongodb-pvc.yaml
```

To verify:

```
$ kubectl get pvc
```

```
ubuntu-master@ubuntumaster:~$ kubectl get pvc
NAME   STATUS   VOLUME           CAPACITY   ACCESS MODES   STORAGECLASS   AGE
pvc    Bound    task-pv-volume   50Gi       RWO            standard       68m
```

3.b.5 Create the mongoDB deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: mongo
  name: mongo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongo
  strategy: {}
  template:
    metadata:
```

```
      creationTimestamp: null
      labels:
        app: mongo
    spec:
      containers:
      - image: mongo
        name: mongod-container
        args: ["--dbpath","/data/db"]
        livenessProbe:
          exec:
            command:
              - mongo
              - --disableImplicitSessions
              - --eval
              - "db.adminCommand('ping')"
          initialDelaySeconds: 30
          periodSeconds: 10
          timeoutSeconds: 5
          successThreshold: 1
          failureThreshold: 6
        readinessProbe:
          exec:
            command:
              - mongo
              - --disableImplicitSessions
              - --eval
              - "db.adminCommand('ping')"
          initialDelaySeconds: 30
          periodSeconds: 10
          timeoutSeconds: 5
          successThreshold: 1
          failureThreshold: 6
        env:
        - name: MONGO_INITDB_ROOT_USERNAME
          value: admin  #user-defined username for root user of mongoDB
        - name: MONGO_INITDB_ROOT_PASSWORD
          value: admin123  #user-defined password root user of mongoDB
        volumeMounts:
        - name: "mongo-data-dir"
          mountPath: "/data/db"
      volumes:
      - name: "mongo-data-dir"
        persistentVolumeClaim:
          claimName: "pvc"
```

*mongodb-deployment.yaml*

On the same directory, execute the command:

```
$  kubectl apply -f mongodb-deployment.yaml
```

To verify:

```
$ kubectl get deployment
```

```
ubuntu-master@ubuntumaster:~$ kubectl get deployments
NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
mongo                   1/1     1            1           71m
```

Along with the external IP address created when the mongoDB service is created, we can connect to the mongoDB using the url:

***mongodb://admin:admin123@192.168.56.10:27017/?authSource=admin***

In general, the URl is formated

| | |
|---|---|
| | ***mongodb://<_username>:<password>@<external_IP_address_of_mongoDB_service>:27017/?authSource=admin*** |

### Setup Web Application

Both the backend and the frontend of the web application is based on NodeJS. The nodeJS server is reponsible for communicating with the mongoDB for CRUD tasks, as well as rendering the html files on client machines.

| **Prepare Web Application docker image** | |
|---|---|
| 3.c.1 | Login to docker using CLI<br><br>`$ sudo docker login`<br><br>When prompt, enter your docker username and password. This will allow us to push docker image build locally to docker hub.<br><br>Expected Output:<br>`ubuntu-master@ubuntumaster:~$ sudo docker login`<br>`[sudo] password for ubuntu-master:`<br>`Authenticating with existing credentials...`<br>`WARNING! Your password will be stored unencrypted in /root/.docker/config.json.`<br>`Configure a credential helper to remove this warning. See`<br>`https://docs.docker.com/engine/reference/commandline/login/#credentials-store`<br><br>`Login Succeeded` |
| 3.c.2 | <u>Dockerize NodeJS Application</u><br><br>The NodeJS Application has the following folder structure:<br><br>`├── controller`<br>`│ └── app.js`<br>`├── models`<br>`├── package-lock.json`<br>`├── server.js`<br>`├── views`<br>`│ ├── images`<br>`│ ├── reviews.ejs`<br>`│ ├── css`<br>`│ └── home.ejs`<br>`├── package.json`<br>`└── Dockerfile`<br><br>`FROM node:16`<br><br>`RUN mkdir /opt/SummerMelts`<br><br>`WORKDIR /opt/SummerMelts`<br><br>`COPY package.json ./` |

```
RUN npm install

COPY . .

EXPOSE 3000

CMD [ "node", "server.js" ]
```

*Dockerfile*

The web application will be expose on port 3000.

Build the image:

```
$ sudo docker build -t summerapp:v1 .
```

3.c.3 Re-tagging an existing local image

```
$ docker tag summerapp:v1 tankahxuan/summerapp:v1
```

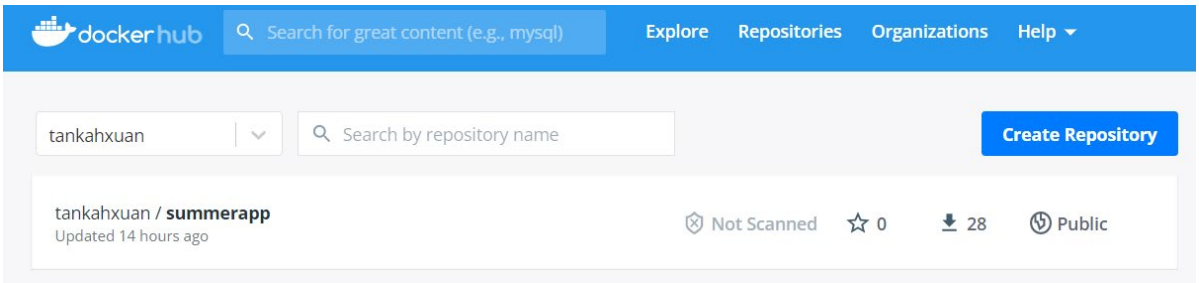
Note: In future, where any changes is made, execute the command to commit the changes

```
$ docker docker commit <existing-container> <hub-user>/<repo-name>[:<tag>]
```

3.c.4 Push the image to the docker registry

```
$ docker push  tankahxuan/summerapp:v1
```

On docker hub:




https://hub.docker.com/repository/docker/tankahxuan/summerapp

**Deploy Web Application on K8s Cluster**

3.c.5 Create a deployment and service object on K8s cluster

```
apiVersion: v1
kind: Service
metadata:
 name: summerapp-backend
spec:
 type: LoadBalancer
 selector:
   app: summerapp-app
 ports:
 - port: 3000
   targetPort: 3000
   protocol: 'TCP'

---

apiVersion: apps/v1
kind: Deployment
metadata:
```

```
 name: summerapp-deployment
 labels:
   app: summerapp
spec:
 replicas: 1
 selector:
   matchLabels:
     app: summerapp-app
     tier: summerapp-backend
 template:
   metadata:
     labels:
       app: summerapp-app
       tier: summerapp-backend
   spec:
     containers:
     - name: summerapp-container
       image: tankahxuan/summerapp:v1
       imagePullPolicy: Always
       livenessProbe:
        httpGet:
          path: /
          port: 3000
        periodSeconds: 20
        initialDelaySeconds: 10
       ports:
       - containerPort: 3000
       env:
       - name: DATABASE_URI
         value: 'mongodb://admin:admin123@192.168.56.0:27017/?authSource=admin'
```

*myappDeploy.yaml*

3.c.5

On the same directory, execute the command:

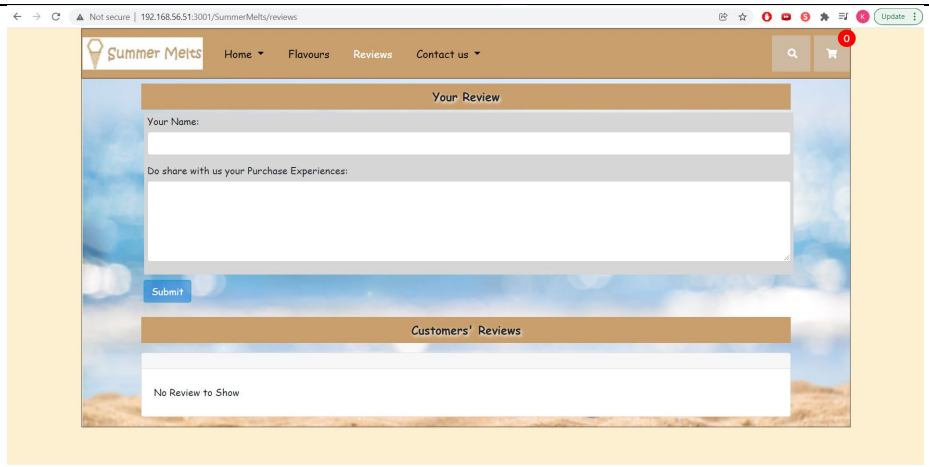
```
$  kubectl apply -f myappDeploy.yaml
```

To verify:

```
ubuntu-master@ubuntumaster:~/learnK8/myapp$ kubectl get deployment
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
mongo               1/1     1            1           103m
summerapp-deployment   0/1     1            0           93m
```

```
ubuntu-master@ubuntumaster:~/learnK8/myapp$ kubectl get services
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)           AGE
kubernetes          ClusterIP      10.96.0.1       <none>          443/TCP           110m
mongo-nodeport-svc  LoadBalancer   10.97.135.205   192.168.56.10   27017:32000/TCP   94m
summerapp-backend   LoadBalancer   10.98.144.68    192.168.56.11   3000:30150/TCP    93m
```

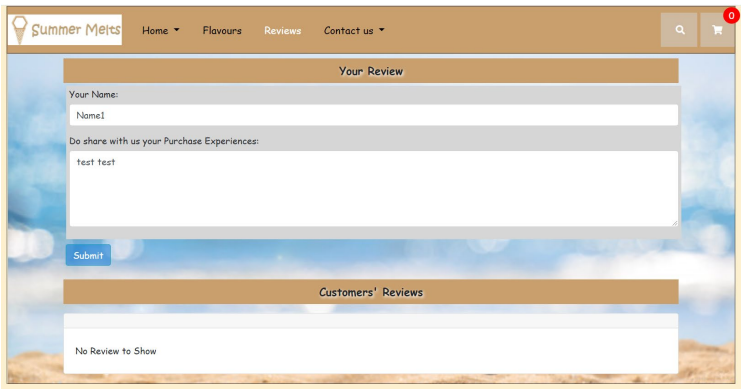
To access the web application, open a browser and enter:
*http://192.168.56.11:3000/SummerMelts/reviews*



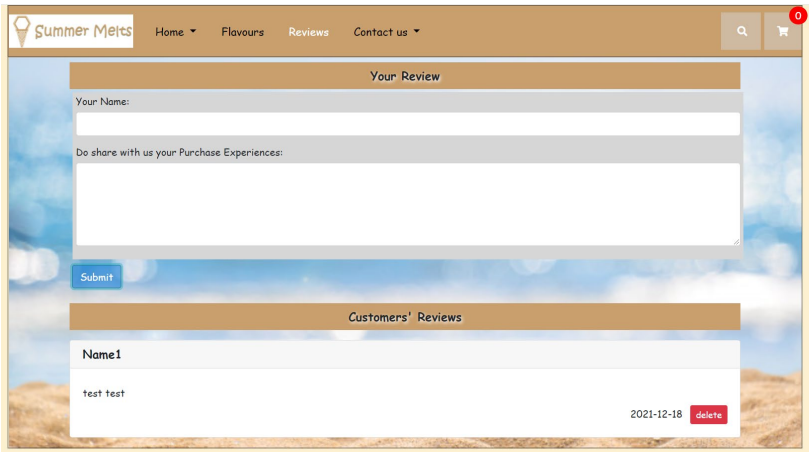

## Web Application Functions

This is a simple web application for users to enter their reviews. The reviews will then be saved in mongoDB. Thereafter, all reviews in mongoDB will be retrieved and display on the same web page.

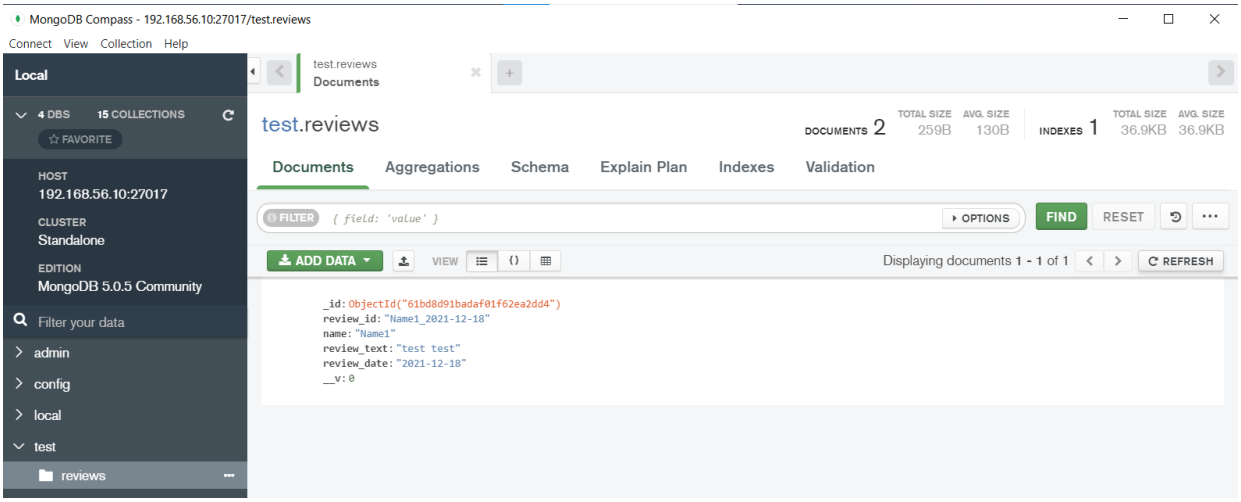
On the web page, enter any random name and describe and click the submit button.




The data will then be saved in the database and retrieve by the web application to be display under the customer's review.




In the database:


MongoDB Compass - 192.168.56.10:27017/test.reviews
Connect View Collection Help
Local
4 DBS 15 COLLECTIONS
FAVORITE
HOST
192.168.56.10:27017
CLUSTER
Standalone
EDITION
MongoDB 5.0.5 Community
Filter your data
admin
config
local
test
reviews
test.reviews
Documents
test.reviews
DOCUMENTS 2 TOTAL SIZE 259B AVG. SIZE 130B
INDEXES 1 TOTAL SIZE 36.9KB AVG. SIZE 36.9KB
Documents Aggregations Schema Explain Plan Indexes Validation
FILTER { field: 'value' }
OPTIONS
FIND
RESET
ADD DATA
VIEW
Displaying documents 1 - 1 of 1
REFRESH
_id: ObjectId("61bd8d91badaf01f62ea2dd4")
review_id: "Name1_2021-12-18"
name: "Name1"
review_text: "test test"
review_date: "2021-12-18"
__v: 0

## Task 4 – Create Users in the Kubrnetes Cluster

### Create Client Certificate

**4.a.1** Create a client key

```
$ openssl genrsa -out ubuntu-master.key 2048
```

```
ubuntu-master@ubuntumaster:~$ openssl genrsa -out ubuntu-master.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.............+++++
.............+++++
e is 65537 (0x010001)
```

**4.a.2** Create a certificate signing request

```
$  openssl req -new -key ubuntu-master.key -out ubuntu-master.csr
```

**4.a.3** Copy the CA certification key to the root folder

Navigate to the root folder: /home/ubuntu-master/

```
$  cd /etc/kubernetes/pki

$  sudo cp ca.crt ca.key /home/ubuntu-master/
```

```
ubuntu-master@ubuntumaster:~$ ls
ca.crt  ca.srl   Documents  learnK8  Pictures  rolebinding.yaml  Templates          ubuntu-master.csr  Videos
ca.key  Desktop  Downloads  Music    Public    role.yaml         ubuntu-master.crt  ubuntu-master.key
```

**4.a.4** Sign the user key and request with cluster certificate

```
$  sudo openssl x509 -req -in ubuntu-master.csr -CA ca.crt -CAkey ca.key -
CAcreateserial -out ubuntu-master.crt -days 300
```

### Add user credentials to the kubeconfig file

**4.b.1** Add user creadentials

```
$ kubectl config set-credentials ubuntu-master --client-certificate=ubuntu-
master.crt --client-key=ubuntu-master.key
```

### Create Role and Role Binding

**4.c.1** Create a role with permission to create, list, update and delete pods

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 name: get-pods
rules:
 - apiGroups: ["*"]
   resources: ["pods"]
   verbs: ["list","get","update","delete"]
```

*role.yaml*

```
$  kubectl apply -f role.yaml
```

**4.c.2** Bind the role to a user

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: labuser-get-pods
subjects:
- kind: User
  name: ubuntu-master
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: get-pods
  apiGroup: rbac.authorization.k8s.io
```

*rolebinding.yaml*

```
$  kubectl apply -f rolebinding.yaml
```

## Task 5 – Backup ETCD cluster data

**Install etcd-client**

5.a.1 On the master node, install the etcd-client

```
$ sudo apt install etcd-client
```

```
ubuntu-master@ubuntumaster:~$ sudo apt install etcd-client
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  etcd-client
0 upgraded, 1 newly installed, 0 to remove and 46 not upgraded.
Need to get 4,563 kB of archives.
After this operation, 17.2 MB of additional disk space will be used.
Get:1 http://sg.archive.ubuntu.com/ubuntu focal/universe amd64 etcd-client amd64 3.2.26+dfsg-6 [4,563 kB]
Fetched 4,563 kB in 2s (3,002 kB/s)
Selecting previously unselected package etcd-client.
(Reading database ... 194672 files and directories currently installed.)
Preparing to unpack .../etcd-client_3.2.26+dfsg-6_amd64.deb ...
Unpacking etcd-client (3.2.26+dfsg-6) ...
Setting up etcd-client (3.2.26+dfsg-6) ...
Processing triggers for man-db (2.9.1-1) ...
```

5.a.2 List the pods under the kube-system namespace

```
$  kubectl get pods -n kube-system
```

```
ubuntu-master@ubuntumaster:~$ kubectl get pods -n kube-system
NAME                                   READY   STATUS    RESTARTS   AGE
coredns-78fcd69978-bsksb               1/1     Running   0          69m
coredns-78fcd69978-nr272               1/1     Running   0          69m
etcd-ubuntumaster                      1/1     Running   6          70m
kube-apiserver-ubuntumaster            1/1     Running   2          70m
kube-controller-manager-ubuntumaster   1/1     Running   4          70m
kube-proxy-227sv                       1/1     Running   0          69m
kube-proxy-7flhr                       1/1     Running   0          69m
kube-proxy-sprx6                       1/1     Running   0          69m
kube-scheduler-ubuntumaster            1/1     Running   4          70m
weave-net-fwmw8                        2/2     Running   0          69m
weave-net-jjk22                        2/2     Running   0          69m
weave-net-nz86b                        2/2     Running   0          69m
```

| | |
|---|---|
| 5.a.3 | Obtain the advertis-client-url of the etcd-pod<br><br>`$ export advertise_url=https://192.168.56.51:2379`<br>`$ echo $advertise_url`<br><br>`ubuntu-master@ubuntumaster:~$ kubectl describe pod etcd-ubuntumaster -n kube-system`<br>`Name: etcd-ubuntumaster`<br>`Namespace: kube-system`<br>`Priority: 2000001000`<br>`Priority Class Name: system-node-critical`<br>`Node: ubuntumaster/10.0.2.101`<br>`Start Time: Sat, 18 Dec 2021 21:35:38 +0800`<br>`Labels: component=etcd`<br>`tier=control-plane`<br>`Annotations: kubeadm.kubernetes.io/etcd.advertise-client-urls: https://192.168.56.51:2379`<br>`kubernetes.io/config.hash: b2a8f224c1fc27870bae1f3185ef904d`<br>`kubernetes.io/config.mirror: b2a8f224c1fc27870bae1f3185ef904d`<br>`kubernetes.io/config.seen: 2021-12-18T21:35:38.731450760+08:00`<br>`kubernetes.io/config.source: file`<br>`seccomp.security.alpha.kubernetes.io/pod: runtime/default`<br>`Status: Running`<br>`IP: 10.0.2.101`<br>`IPs:`<br>`IP: 10.0.2.101`<br>`Controlled By: Node/ubuntumaster`<br>`Containers:`<br>`etcd:`<br>`Container ID: docker://ba7361cc30b36ae638e6910806cb8b8fac71fc32ad3f97ab8849abd9980ae3e5`<br>`Image: k8s.gcr.io/etcd:3.5.0-0`<br>`Image ID: docker-pullable://k8s.gcr.io/etcd@sha256:9ce33ba33d8e738a5b85ed50b5080ac746deceed4a7496c550927a7a19ca3b6d`<br>`Port: <none>`<br>`Host Port: <none>`<br>`Command:`<br>`etcd`<br>`--advertise-client-urls=https://192.168.56.51:2379`<br>`--cert-file=/etc/kubernetes/pki/etcd/server.crt`<br>`--client-cert-auth=true`<br>`--data-dir=/var/lib/etcd`<br>`--initial-advertise-peer-urls=https://192.168.56.51:2380`<br>`--initial-cluster=ubuntumaster=https://192.168.56.51:2380`<br>`--key-file=/etc/kubernetes/pki/etcd/server.key`<br>`--listen-client-urls=https://127.0.0.1:2379,https://192.168.56.51:2379`<br>`--listen-metrics-urls=http://127.0.0.1:2381`<br><br>Here, the advertise-client-urls is *https://192.168.56.51:2379* |
| 5.a.4 | Set the advertis-client-url to environment variable<br><br>`ubuntu-master@ubuntumaster:~$ export advertise_url=https://192.168.56.51:2379`<br>`ubuntu-master@ubuntumaster:~$ echo $advertise_url`<br>`https://192.168.56.51:2379` |
| 5.a.5 | Create the backup<br>Execute the following command to create a snapshot of the etcd database, named "etcd_backup.db"<br>`sudo ETCDCTL_API=3 etcdctl \`<br>`--endpoints $advertise_url \`<br>`--cacert /etc/kubernetes/pki/etcd/ca.crt \`<br>`--key /etc/kubernetes/pki/etcd/server.key \`<br>`--cert /etc/kubernetes/pki/etcd/server.crt snapshot save etcd_backup.db` |

## Task 6 – Auto Scaling

| **Reset kubernetes cluster** | |
|---|---|
| If there is a need to reconfigure or reset your existing Kubernetes cluster, follow the steps below. | |
| a | Create the horizontal scaling object<br><br>`apiVersion: autoscaling/v2beta2`<br>`kind: HorizontalPodAutoscaler`<br>`metadata:`<br>`name: horizontal-scale`<br>`spec:`<br>`scaleTargetRef:`<br>`apiVersion: apps/v1` |

```
    kind: Deployment
    name: summerapp-deployment
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

*autoScale.yaml*

```
$ kubectl apply -f autoScale.yaml
```

## Appendix – To reset kuberneter cluster

**Reset kubernetes cluster**

If there is a need to reconfigure or reset your existing Kubernetes cluster, follow the steps below.

a

**Reset kubernetes cluster using kubeadm**

```
$ sudo kubeadm reset -f
```

Output:

```
ubuntu-master@ubuntumaster:~$ sudo kubeadm reset
[reset] WARNING: Changes made to this host by 'kubeadm init' or 'kubeadm join' will be reverted.
[reset] Are you sure you want to proceed? [y/N]: y
[preflight] Running pre-flight checks
W1124 11:25:59.094585   13080 removeetcdmember.go:80] [reset] No kubeadm config, using etcd pod spec to get data directory
[reset] No etcd config found. Assuming external etcd
[reset] Please, manually reset etcd to prevent further issues
[reset] Stopping the kubelet service
[reset] Unmounting mounted directories in "/var/lib/kubelet"
[reset] Deleting contents of config directories: [/etc/kubernetes/manifests /etc/kubernetes/pki]
[reset] Deleting files: [/etc/kubernetes/admin.conf /etc/kubernetes/kubelet.conf /etc/kubernetes/bootstrap-kubelet.conf /etc/kubernetes
/controller-manager.conf /etc/kubernetes/scheduler.conf]
[reset] Deleting contents of stateful directories: [/var/lib/kubelet /var/lib/dockershim /var/run/kubernetes /var/lib/cni]

The reset process does not clean CNI configuration. To do so, you must remove /etc/cni/net.d

The reset process does not reset or clean up iptables rules or IPVS tables.
If you wish to reset iptables, you must do so manually by using the "iptables" command.

If your cluster was setup to utilize IPVS, run ipvsadm --clear (or similar)
to reset your system's IPVS tables.

The reset process does not clean your kubeconfig files and you must remove them manually.
Please, check the contents of the $HOME/.kube/config file.
```

b

**Remove all the data from all below location**

```
$ sudo rm -rf /etc/cni /etc/kubernetes /var/lib/dockershim /var/lib/etcd
/var/lib/kubelet /var/run/kubernetes ~/.kube/*
```

**Restart the docker service**

```
$ systemctl restart docker
```

**Reset kubernetes cluster**

If there is a need to reconfigure or reset your existing Kubernetes cluster, follow the steps below.

a

## Reset kubernetes cluster using kubeadm

```
$ sudo kubeadm reset -f
```

Output:

```
ubuntu-master@ubuntumaster:~$ sudo kubeadm reset
[reset] WARNING: Changes made to this host by 'kubeadm init' or 'kubeadm join' will be reverted.
[reset] Are you sure you want to proceed? [y/N]: y
[preflight] Running pre-flight checks
W1124 11:25:59.094585   13080 removeetcdmember.go:80] [reset] No kubeadm config, using etcd pod spec to get data directory
[reset] No etcd config found. Assuming external etcd
[reset] Please, manually reset etcd to prevent further issues
[reset] Stopping the kubelet service
[reset] Unmounting mounted directories in "/var/lib/kubelet"
[reset] Deleting contents of config directories: [/etc/kubernetes/manifests /etc/kubernetes/pki]
[reset] Deleting files: [/etc/kubernetes/admin.conf /etc/kubernetes/kubelet.conf /etc/kubernetes/bootstrap-kubelet.conf /etc/kubernetes
/controller-manager.conf /etc/kubernetes/scheduler.conf]
[reset] Deleting contents of stateful directories: [/var/lib/kubelet /var/lib/dockershim /var/run/kubernetes /var/lib/cni]

The reset process does not clean CNI configuration. To do so, you must remove /etc/cni/net.d

The reset process does not reset or clean up iptables rules or IPVS tables.
If you wish to reset iptables, you must do so manually by using the "iptables" command.

If your cluster was setup to utilize IPVS, run ipvsadm --clear (or similar)
to reset your system's IPVS tables.

The reset process does not clean your kubeconfig files and you must remove them manually.
Please, check the contents of the $HOME/.kube/config file.
```

b

## Remove all the data from all below location

```
$ sudo rm -rf /etc/cni /etc/kubernetes /var/lib/dockershim /var/lib/etcd
/var/lib/kubelet /var/run/kubernetes ~/.kube/*
```

## Restart the docker service

```
$ systemctl restart docker
```