Introduction to Cloud Computing – Exercise 4

Scope: Minikube, Kubectl

Introduction:

In this lab, we will explore minikube, a tool to run Kubernetes locally. Minikube runs a single node inside a VM in our VM.

1. Kubectl and minukube installation:

The first step is to download the repository and install it

```
curl -LO https://dl.k8s.io/release/$(curl -L -s 
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl
```

Next install kubectl

```
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

To make sure we have the latest version, run the command

```
kubectl version --client
```

Now that we have kubectl we will install minikube. Download the latest stable version

```
curl-LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

After the correct installation, we can proceed to launch our cluster.

```
minikube start
```

If minikube fails to start, driver settings are required
https://minikube.sigs.k8s.io/docs/drivers/docker/

At this point we have cluster installed, to have access to it we need CLI for kubernetes installed.

2. First application

When we have minikube running, we can see our available cluster

```
kubectl get after -A
```

What score did you get? What components are running?

Then create your first deployment and expose your application on port 8080

```
kubectl create deployment hello-minikube --image=k8s.gcr.io/echoserver:1.4
kubectl expose deployment hello-minikube --type=NodePort --port=8080
```

After a while, you should be able to see the running apps

```
kubectl get services hello-minikube
```

We can access the application in two ways. The first is to use a minikube

```
MiniKube Service Hello-MiniKube
```

The second way is port forwarding

```
kubectl port-forward service/hello-minikube 7080:8080
```

Then open your browser and go to http://localhost:7080/

You should be able to see request metadata from nginx, such as CLIENT VALUES, SERVER VALUES, HEADERS RECEIVED in the application output. You can make a POST request and watch the content appear in the BODY section.

Include screenshots in the report

3. LoadBalancer

In this case, we will implement deploy by selecting the type not *NodePort* but *LoadBalancer*.

```
kubectl create deployment balanced --image=k8s.gcr.io/echoserver:1.4
kubectl expose deployment balanced --type=LoadBalancer --port=8080
```

In another window, run the tunnel to create a pinned IP address for our implementation

```
Minikube tunnel
```

To find the address, call the command below and you will find the address in the EXTERNAL-IP column

```
kubectl get balanced services
```

The deployment is available at <EXTERNAL-IP>:8080

Take a screenshot. Explain the difference between type=NodePort and type=LoadBalancer.

4. Deploy using YAML files

In this task, we will run an application consisting of a controller, service and frontend part – Guestbook.

The first stage of launching the application is to launch Redis Master. A Kubernetes deployment consists of two or more parts - replication controller and service.

The replication controller defines the number of instances to run, the Docker image to use, and the name that identifies the service. Additional options can be used for configuration.

If something wrong with Redis replication, the replication controller would restart it on the active node.

First of all, we need to run the redist controller. Create YAML file (nano command)

```
apiVersion: v1
kind: ReplicationController
metadata:
  Name: redis-master
  Tags:
    Name: redis-master
Spec:
  replicas: 1
  Selector:
    Name: redis-master
  Template:
    metadata:
      Tags:
        Name: redis-master
    Spec:
      containers:
```

```
      - name: master
        image: redis:3.0.7-alpine
        Ports:
        - containerPort: 6379
```

Save the file as *redist-master-controller.yaml*

It will then call *the kubectl* command

```
kubectl create -f redis-master-controller.yaml
```

At this point we have created the Replication Controller. You can see the available controllers after calling the command

```
kubectl get rc
```

The second part is Service. Service is a named load balancer that forwards traffic to one or more containers. The proxy works even when the containers are located on different nodes.

Service Proxy communicates in a cluster and rarely providesaccess to an external interface.

When you start a service, it looks like you can't connect using curl or netcat unless you run it as part of Kubernetes. The recommended approach is to have a LoadBalancer service to handle external communications. Create a new YAML file

```
apiVersion: v1
kind: Service
metadata:
  Name: redis-master
  Tags:
    Name: redis-master
Spec:
  Ports:
    # the port that this service should serve on
  - Port: 6379
    targetPort: 6379
  Selector:
    Name: redis-master
```

And save it as *redis-master-service.yaml.* To start the website using the YAML file, call

```
Kubectl create –f redis-master-service.yaml
```

To check the running services, execute

Kubectl get services

Follow the same steps for the configuration files below. Ultimately, we want to run the following components:

- Redis master controller

- Redis master service

- Redis slave controller

- Redis slave service

- Frontend controller

- Frontend service

Redis-slave-controller.yaml file:

```
apiVersion: v1
kind: ReplicationController
metadata:
  Name: redis-slave
  Tags:
    Name: redis-slave
Spec:
  replicas: 2
  Selector:
    Name: redis-slave
  Template:
    metadata:
      Tags:
        Name: redis-slave
    Spec:
      containers:
      - name: worker
        image: gcr.io/google_samples/gb-redisslave:v1
        Env:
        - name: GET_HOSTS_FROM
          value: DNS
          # If your cluster config does not include a dns service, then to
          # instead access an environment variable to find the master
          # service's host, comment out the 'value: dns' line above, and
          # uncomment the line below.
          # value: env
        Ports:
        - containerPort: 6379
```

Redis-slave-service.yaml file:

```
apiVersion: v1
kind: Service
metadata:
  Name: redis-slave
  Tags:
    Name: redis-slave
Spec:
  Ports:
    # the port that this service should serve on
  - Port: 6379
  Selector:
    Name: redis-slave
```

Frontend-controller.yaml file:

```
apiVersion: v1
kind: ReplicationController
metadata:
  Name: frontend
  Tags:
    Name: frontend
Spec:
  Replicas: 3
  Selector:
    Name: frontend
  Template:
    metadata:
      Tags:
        Name: frontend
    Spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
        Env:
        - name: GET_HOSTS_FROM
          value: DNS
          # If your cluster config does not include a dns service, then to
          # instead access environment variables to find service host
          # info, comment out the 'value: dns' line above, and uncomment the
          # line below.
          # value: env
        Ports:
        - containerPort: 80
```

frontend-service.yaml file:

```
apiVersion: v1
kind: Service
metadata:
  Name: frontend
  Tags:
    Name: frontend
Spec:
  # if your cluster supports it, uncomment the following to automatically create
  # an external load-balanced IP for the frontend service.
  # type: LoadBalancer
  type: NodePort
  Ports:
    # the port that this service should serve on
    - port: 80
      nodePort: 30080
  Selector:
    Name: frontend
```

Once all the components are up and running, you should be able to access your guestbook. To do this, you need to know the port under which your service is running. Call

```
kubectl describe service frontend | grep NodePort
```

At what address and port dos isour service blunt?

Present a working book to the presenter and add a screenshot to the report.

Where is the amount of redist slave replication defined? Point to the file and specific line.

5. Exercise*

Find the Docker image in the docker repository that you think will be interesting, then run it on the Kubernetes cluster.

Include in the report what image you downloaded, commands required to run and a screenshot of the launch.