LAB – Cloud Platform Operations

Hands on experience with Kubernetes management operations

References:

- Kubernetes documentation https://kubernetes.io/docs/home/

Define a Pod

- Kubernetes takes as input the description of Pods, group of one or more applications to be deployed together in the instance
- The description of the Pod and its configuration is provided as a Yaml file

pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
    name: helloworld
    labels:
        app: helloworld
spec:
    containers:
    - name: helloworld
      image: luksa/kubia
      ports:
      - containerPort: 8080
```

Pod metadata

Containers composing the pods

Instantiate the hello world Pod

```
kubectl apply -f pod.yaml
```

```
root@SNHOYM5GWPGME2L:~/kubia# kubectl apply -f pod.yaml
pod/helloworld created
```

Check pod’s status

`kubectl get pods`

```
root@SNHOYM5GWPGME2L:~/kubia# kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
helloworld    1/1     Running   0          62s
```

Delete the Pod

```
kubectl delete pod helloworld
kubectl get pods
```

```
root@SNHOYM5GWPGME2L:~/kubia# kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
helloworld    1/1     Running   0          62s
root@SNHOYM5GWPGME2L:~/kubia# kubectl delete pod helloworld
pod "helloworld" deleted



root@SNHOYM5GWPGME2L:~/kubia#
root@SNHOYM5GWPGME2L:~/kubia#
root@SNHOYM5GWPGME2L:~/kubia#
root@SNHOYM5GWPGME2L:~/kubia# kubectl get pods
No resources found in default namespace.
```

Create a Deployment with Replicas

- Pods allow to deploy applications without any supporting functions
- If you want Kubernetes also to handle some of the functions to manage the application lifecycle you need to define a Deployment
- In a deployment pods can be deployed specifying also additional functionalities, e.g. the number of replicas

deployment.*yaml*

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: helloworld
spec:
  selector:
    matchLabels:
      run: helloworld
  replicas: 2
  template:
    metadata:
      labels:
        run: helloworld
    spec:
      containers:
      - name: helloworld
        image: luksa/kubia
        ports:
        - containerPort: 8080
```

Number of replicas to be instantiated

Instantiate the deployment

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

```
root@SNHOYM5GWPGME2L:~/kubia# kubectl apply -f deployment.yaml
deployment.apps/helloworld created
```

Check the deployment

```
kubectl get deployments
kubectl get pods
```

```
root@SNHOYM5GWPGME2L:~/kubia# kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
helloworld    2/2     2            2           63s
root@SNHOYM5GWPGME2L:~/kubia# kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
helloworld-958544d5c-lrjww     1/1     Running   0          78s
helloworld-958544d5c-m55fv     1/1     Running   0          78s
```

Try to kill a pod

kubectl delete pod helloworld-958544d5c-lrjww

```
root@SNHOYM5GWPGME2L:~/kubia# kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
helloworld    2/2     2            2           63s
root@SNHOYM5GWPGME2L:~/kubia# kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
helloworld-958544d5c-lrjww     1/1     Running   0          78s
helloworld-958544d5c-m55fv     1/1     Running   0          78s
root@SNHOYM5GWPGME2L:~/kubia# kubectl delete pod helloworld-958544d5c-lrjww
pod "helloworld-958544d5c-lrjww" deleted
root@SNHOYM5GWPGME2L:~/kubia# kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
helloworld-958544d5c-lhckq     1/1     Running   0          62s
helloworld-958544d5c-m55fv     1/1     Running   0          3m23s
root@SNHOYM5GWPGME2L:~/kubia#
```

Accessing the service

- Kubia is container that exposes a simple web application that returns the hostname on which it is running. It listens on port 8080
- https://github.com/luksa/kubernetes-in-action/blob/master/Chapter02/kubia/app.js

```
const http = require('http');
const os = require('os');

console.log("Kubia server starting...");

var handler = function(request, response) {
  console.log("Received request from " + request.connection.remoteAddress);
  response.writeHead(200);
  response.end("You've hit " + os.hostname() + "\n");
};

var www = http.createServer(handler);
www.listen(8080);
```

Expose the deployment internally by creating a service

- In order to expose a service running in a container (so it can be reached from outside) you need to create a service:

```
kubectl expose deployment/helloworld
```

- The command create a new service of type ClusterIP

```
kubectl get services
```

```
root@SNHOYM5GWPGME2L:~/kubia# kubectl expose deployment/helloworld
service/helloworld exposed
root@SNHOYM5GWPGME2L:~/kubia# kubectl get services
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
helloworld    ClusterIP   10.152.183.71   <none>        8080/TCP   31s
kubernetes    ClusterIP   10.152.183.1    <none>        443/TCP    12d
```

Endpoints

- Each Pod has an endpoint associated
- It is a private IP address on a virtual LAN accessible only to Pods and the kubernetes nodes of the platform (the IP of the containers in the private network)

`kubectl describe service helloworld`

```
root@SNHOYM5GWPGME2L:~/kubia# kubectl describe service helloworld
Name:                     helloworld
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 run=helloworld
Type:                     ClusterIP
IP:                       10.152.183.146
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
Endpoints:                10.1.43.28:8080,10.1.76.22:8080
Session Affinity:         None
Events:                   <none>
```

Endpoints

- An endpoint can be used to reach the service on a specific pod
- Endpoints are associated with pods, when a pod dies a new pod is created with a new IP address. The services accessing the pod should change the IP for their requests (unfeasible!)

```
root@SWNUMUYKY62JPXH:~# curl http://10.1.43.28:8080
You've hit helloworld-958544d5c-m55fv
root@SWNUMUYKY62JPXH:~# curl http://10.1.76.22:8080
You've hit helloworld-958544d5c-lhckq
root@SWNUMUYKY62JPXH:~#
```

ClusterIP

- In the ClusterIP method this issue is resolved by using Virtual IPs
- Specifically, the platform create a VirtualIP that is is associated with a deployed service
- A Request from any other service running in the platform directed to the VirtualIP associated with the service is dispatched to one of the pods

Test

- Run **`curl http://ClusterIP:8080`**
- One pod (**random**) handles and serves the request

```
root@SWNUMUYKY62JPXH:~# curl http://10.152.183.71:8080
You've hit helloworld-958544d5c-m55fv
root@SWNUMUYKY62JPXH:~# curl http://10.152.183.71:8080
You've hit helloworld-958544d5c-lhckq
root@SWNUMUYKY62JPXH:~# curl http://10.152.183.71:8080
You've hit helloworld-958544d5c-m55fv
root@SWNUMUYKY62JPXH:~# curl http://10.152.183.71:8080
You've hit helloworld-958544d5c-m55fv
root@SWNUMUYKY62JPXH:~# curl http://10.152.183.71:8080
You've hit helloworld-958544d5c-m55fv
root@SWNUMUYKY62JPXH:~#
```

DNS name

- An application (or another service) that wants to contact one of the instances of a service can use the ClusterIP or the internal DNS service
- Check if DNS service is running:

```
kubectl get services kube-dns --namespace=kube-system
```

```
root@SNHOYM5GWPGME2L:~/kubia# kubectl get services kube-dns --namespace=kube-system
NAME       TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                  AGE
kube-dns   ClusterIP   10.152.183.161   <none>        53/UDP,53/TCP,9153/TCP   13d
```

Test DNS alias

- Run a different Pod

kubectl run curl --image=radial/busyboxplus:curl -i --tty

```
root@SNHOYM5GWPGME2L:~/kubia# kubectl run curl --image=radial/busyboxplus:curl -
i --tty
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in
a future version. Use kubectl run --generator=run-pod/v1 or kubectl create inste
ad.
If you don't see a command prompt, try pressing enter.
[ root@curl-69c656fd45-cvmmb:/ ]$
[ root@curl-69c656fd45-cvmmb:/ ]$
[ root@curl-69c656fd45-cvmmb:/ ]$
```

Check the DNS alias

`nslookup helloworld`

```
[ root@curl-69c656fd45-cvmmb:/ ]$ nslookup helloworld
Server:    10.152.183.161
Address 1: 10.152.183.161 kube-dns.kube-system.svc.cluster.local

Name:      helloworld
Address 1: 10.152.183.71 helloworld.default.svc.cluster.local
[ root@curl-69c656fd45-cvmmb:/ ]$ curl http://helloworld:8080
You've hit helloworld-958544d5c-lhckq
[ root@curl-69c656fd45-cvmmb:/ ]$ curl http://helloworld:8080
You've hit helloworld-958544d5c-m55fv
[ root@curl-69c656fd45-cvmmb:/ ]$
```

Expose the service to external systems

- To expose a service to external system a different service creation method has to be adopted since the ClusterIP virtual address is reachable only from within the Kubernetes platform
- To expose the service so it is accessible from external networks first, destroy the service

```
kubectl delete service helloworld
```

- Then create a service of type *NodePort*
- NodePort allows to expose a service by configuring Port Forwarding on each Kubernetes node in the cluster
- A public port is selected by the system and opened on the public IP address of each Kubernetes node
- The traffic received by each node on that port is forwarded to the ClusterIP associated with the service on its exposed port

Expose the service to external systems

```
kubectl expose deployment/helloworld --type=NodePort
kubectl get services
```

- The port is selected randomly

```
root@SNHOYM5GWPGME2L:~/kubia# kubectl expose deployment/helloworld --type=NodePort
service/helloworld exposed
root@SNHOYM5GWPGME2L:~/kubia# kubectl get services
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
helloworld    NodePort    10.152.183.254   <none>        8080:31841/TCP   47s
kubernetes    ClusterIP   10.152.183.1     <none>        443/TCP          13d
```

Test

```
root@SNHOYM5GWPGME2L:~/kubia# curl http://172.16.0.110:31841
You've hit helloworld-958544d5c-lhckq
root@SNHOYM5GWPGME2L:~/kubia# curl http://172.16.0.110:31841
You've hit helloworld-958544d5c-lhckq
root@SNHOYM5GWPGME2L:~/kubia# curl http://172.16.0.110:31841
You've hit helloworld-958544d5c-lhckq
root@SNHOYM5GWPGME2L:~/kubia# curl http://172.16.0.109:31841
You've hit helloworld-958544d5c-m55fv
```

Load Balancer

- Expose a service using an external load balancer is the best option
- A specific public IP address is allocated from a pool of available addresses and assigned to a specific service (a pod) responsible for dispatching the traffic to different pods following a load balancing policy

```
kubectl delete service helloworld
kubectl expose deploy helloworld --port 8080 --type
LoadBalancer
```

Check and test the service

```
root@SNHOYM5GWPGME2L:~# kubectl get services
NAME          TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
helloworld    LoadBalancer   10.152.183.145   172.16.1.1    8080:32699/TCP   10m
kubernetes    ClusterIP      10.152.183.1     <none>        443/TCP          13d
```

```
root@SNHOYM5GWPGME2L:~# curl http://172.16.1.1:8080
You've hit helloworld-958544d5c-m55fv
```

Minikube Bug

https://github.com/kubernetes/minikube/issues/13898

Before running the next exercise

```
minikube stop

minikube start --extra-config=kubelet.housekeeping-interval=10s
```

Horizontal Autoscaling

- Create a deployment that performs some intensive computation task and has a cap on the resources it can use:

autoscale.*yaml*

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: php-apache
spec:
  selector:
    matchLabels:
      run: php-apache
  replicas: 2
  template:
    metadata:
      labels:
        run: helloworld
spec:
      containers:
      - name: php-apache
        image: k8s.gcr.io/hpa-example
        ports:
        - containerPort: 80
        resources:
          limits:
            cpu: "500m"
          requests:
            cpu: "200m"
```

```
kubectl apply -f autoscale.yaml
```

```php
<?php
  $x = 0.0001;
  for ($i = 0; $i <= 1000000; $i++) {
    $x += sqrt($x);
  }
  echo "OK!";
?>
```

Instantiate the autoscaler

```
kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10

kubectl expose deployment/php-apache --type=ClusterIP
```

- It aims at maintaining a cpu target of 50% of load

```
root@SNHOYM5GWPGME2L:~# kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10
horizontalpodautoscaler.autoscaling/php-apache autoscaled
root@SNHOYM5GWPGME2L:~# kubectl get hpa
NAME         REFERENCE                TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache    <unknown>/50%   1         10        0          5s
```

Trigger some load

```
while true; do wget -q -O- http://ClusterIP; done
```

```
kubectl get services (to retrieve the ClusterIP)
```

Check autoscaling

```
kubectl  get hpa
```

```
root@SWNUMUYKY62JPXH:~# kubectl get hpa
NAME          REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache    Deployment/php-apache    0%/50%    1         10        1          75s
root@SWNUMUYKY62JPXH:~# kubectl get hpa
NAME          REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache    Deployment/php-apache    0%/50%    1         10        1          76s
root@SWNUMUYKY62JPXH:~# kubectl get hpa
NAME          REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
php-apache    Deployment/php-apache    147%/50%   1         10        3          2m9s
root@SWNUMUYKY62JPXH:~# kubectl get hpa
NAME          REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
php-apache    Deployment/php-apache    147%/50%   1         10        3          2m12s
root@SWNUMUYKY62JPXH:~# kubectl get hpa
NAME          REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache    Deployment/php-apache    86%/50%   1         10        3          2m45s
root@SWNUMUYKY62JPXH:~# kubectl get hpa
NAME          REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache    Deployment/php-apache    38%/50%   1         10        6          3m15s
root@SWNUMUYKY62JPXH:~# kubectl get hpa
NAME          REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache    Deployment/php-apache    38%/50%   1         10        6          3m20s
root@SWNUMUYKY62JPXH:~# kubectl get hpa
NAME          REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache    Deployment/php-apache    33%/50%   1         10        6          3m42s
root@SWNUMUYKY62JPXH:~# kubectl get hpa
NAME          REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache    Deployment/php-apache    0%/50%    1         10        6          5m14s
```

References

- https://medium.com/google-cloud/kubernetes-nodeport-vs-loadbalancer-vs-ingress-when-should-i-use-what-922f010849e0