Cloud Computing Applications and Services
Containers Part II

November 21, 2021

Kubernetes

Kubernetes (`https://kubernetes.io`), also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.

Along this exercise guide, we will go through the steps of configuring and deploying a Kubernetes cluster and using its resources to deploy the Swap and Mysql containers used in the last practical guide.

Cluster configuration

1. Setup three Ubuntu virtual machines. Each VM needs to have 2GB of RAM and 2VCPUs
   (a Vagrant file is provided along with this guide if you want to use it).
   In each VM apply the following steps:

2. Install and enable Docker

```
sudo apt-get install docker.io -y
sudo su
echo '{"exec-opts": ["native.cgroupdriver=systemd"]}' >> /etc/docker/daemon.json
exit
sudo systemctl enable docker
sudo systemctl restart docker
```

3. Install Kubernetes

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
sudo apt-add-repository "deb https://apt.kubernetes.io/ kubernetes-xenial main"
sudo apt-get update
sudo apt-get install -y kubeadm kubelet kubectl
sudo apt-mark hold kubeadm kubelet kubectl
sudo swapoff -a
```

Cluster initialization

VM1 will hold the Kubernetes Master node. At this VM:

1. Initialize Kubernetes cluster with
   sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address=10.0.0.3 (at VM1)
   Note that we are using the host-only ip for VM1 configured in the previous practical guides.
   Once the last command finishes, a kubeadm join command will be displayed. Save it for later.

2. Create a directory for the Kubernetes cluster (at VM1).

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

3. Edit file */etc/kubernetes/manifests/kube-scheduler.yaml* and remove the line "- *--port=0*". Then restart the kubelet service with *sudo systemctl restart kubelet.service*.
4. Check the status of kubernetes services with *kubectl get cs*.
5. Deploy the Flannel network
   (a) Check the name of the host-only interface (it should be either *"eth1"* or *"enp0s8"*): `ip add`
   (b) Edit the following line on provided file named "kube-flannel.yaml" to the correct interface name:

```
- --iface=eth1
```

   (c) Deploy the Flannel network with: *kubectl apply -f kube-flannel.yaml*

VM2 and VM3 will hold a Worker node. At these VMs:

1. Run (as sudo) the join command outputted when creating the Kubernetes Cluster.
2. Check the nodes of the cluster with *kubectl get nodes* (at VM1).

MySQL container deployment

1. Now, we are going to configure Persistent Volume (PV) and Persistent Volume Claim (PVC) resources for the MySQL pod.

   Similar to containers, when a Pod is killed or exits, all the data stored in its internal storage is lost. When the data needs to be persisted, the pod must be attached with a respective Persistent Volume Claim (PVC).

   File "mysql-pv.yaml" first creates a Persistent Volume (VP), named "mysql-pv-volume", that claims 20GiB from storage with ReadWriteOne access mode. The Host path where the data will reside will be "/mnt/data".

   Then, it creates a Persistent Volume Claim (PVC), named "mysql-pv-claim", that claims 20GB from the Persistent Volume created earlier, with the same access mode.

   Run the following command to create the PV and PVC resources for MySQL:

```
kubectl apply -f mysql-pv.yaml
```

   To see PV / PVC objects run:

```
kubectl get pv
kubectl get pvc
```

2. Next, we need to define a Service object and a Deployment object for MySQL

   The file "mysql-deployment.yaml" creates two Kubernetes objects: a Service and a Deployment. The Service object, named "mysql", uses the default Kubernetes service type (ClusterIP), and enables access to the application (mysql) from any application inside the cluster, on port 3306.

   The Deployment object, named "mysql" configures the deployment of the MySQL server in the cluster. The pods are configured to run the image "mysql:latest" on port 3360. In the field *"env"* we set the mandatory environment variable MYSQL_ROOT_PASSWORD to specify the password that will be set for the MySQL root superuser account. We also set the variables MYSQL_DATABASE, MYSQL_USER and MYSQL_PASSWORD to create a database, a user and a password for use later with the Swap application. In the end, we configure a volume from the PersistentStorageClaim resource created earlier to be mounted on the path `/var/lib/mysql`.

   Run the following command to create the MySQL deployment:

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

To see Services / Deployments / Pods objects run:

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

Swap container deployment

1. Finally, we configure a Service object and a Deployment object for Swap

   The file "swap-deployment.yaml" creates a Service and a Deployment object to run the Swap application. The Service object, named "swap", is of the type NodePort and allows us to open a specific port on all the Nodes and forward any traffic sent to this port to the service.

   You can check more advanced (*i.e.*, other than NodePort) types of service publishing at `https://kubernetes.io/docs/concepts/services-networking/service/`.

   The Deployment object, named "swap", configures two replicas of the Swap pod, containing a container running the image "ascn21:swap" on port 8000. We also set the environment variables required by the Swap image: DB_HOST, DB_DATABASE, DB_USERNAME and DB_PASSWORD.

   Make sure the MySQL deployment is ready:

   ```
   kubectl get deployment mysql
   ```

   Then, run the following command to create the Swap deployment:

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

2. Check if the Swap Deployment is ready

   ```
   kubectl get deployment swap
   ```

   Run the following command the see how many Pods for swap are running:

   ```
   kubectl get pods --selector=app=swap -o wide
   ```

   The column "node" indicates the node where the pod is running.

3. Try it out!

   (a) access swap from your browser. The url should contain the ip of VM1 and port 30007 (e.g., 10.0.0.3:30007)

Extra

1. Explore ConfigMaps for decoupling environment-dependent configurations from containerized applications, allowing them to remain portable across environments.
   `https://kubernetes.io/docs/concepts/configuration/configmap/`
2. Explore Secrets for storing sensitive data such as passwords, tokens, or keys.
   `https://kubernetes.io/docs/concepts/configuration/secret/`

Learning outcomes Hands-on experience with software container technology and orchestration (Docker and Kubernetes). Deployment of a distributed application on top of a container platform.