Cloud Computing Applications and Services
Kubernetes

October 29, 2023

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 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 guides.

Tasks

K8s Cluster Configuration and Initialization

1. Setup three Ubuntu 20.04 virtual machines. Each VM needs to have 2GB of RAM and 2VCPUs.
   (a) Use the provided *Vagrantfile* to setup 1 Master and 2 Worker nodes, each running at an independent VM.
   (b) The *Vagrantfile* contains three provision scripts:
   - *provision_all*: configure SSH authentication, and install Docker and Kubernetes.
   - *provision_master*: Configure K8s cluster, Master node, and network overlay (flannel).
   - *provision_worker*: Add to the K8s cluster and configure Worker node.
2. Check the nodes of the cluster with `kubectl get nodes` (at the Master VM).

Persistent Volumes Configuration

Persistent Volumes (PV) are used to ensure data persistency for Pods. We will create two Local Persistent Volumes, one per Worker node. The YAML file to create PVs on K8s is provided along with this guide. Be sure to inspect it and understand it while checking the steps below. Remember to consult K8s documentation at `https://kubernetes.io/docs/home/`.

Important notes when inspecting the *persistent-volume.yml* file:

- The file specifies a Storage Class (*local-storage*) with the *volumeBindingMode* equal to *WaitForFirstConsumer* to instruct K8s to wait to bind a PVC until the Pod (e.g., MySQL) using it is actually scheduled.
- Two Local PVs are defined. The *pv001* PV is assigned to *Worker1* and *pv002* PV is assigned to *Worker2*.

1. Since Local PVs do not support dynamic volume provision, the administrator must create/delete the PV objects and the associated disks/folders. Therefore, at the Workers VMs run the following command for creating the corresponding PV's folder:

```
sudo mkdir -p /mnt/data
```

2. At the Master VM run the command `kubectl apply -f persistent-volume.yml` to create a Storage Class and two PV objects.

3. Use the command `kubectl get sc` to list the existing Storage Class objects.

4. Use the command `kubectl get pv` to list the existing PV objects.

MySQL Container Deployment

The YAML files to deploy MySQL on K8s are provided along with this guide. Be sure to inspect these and understand them while doing the steps below. Remember to consult K8s documentation at `https://kubernetes.io/docs/home/`.

1. First, we must create a Persistent Volume Claim (PVC) object (*e.g., mysql-pv-claim*) to which the MySQL Pod will bound. To create this PVC run the following command: `kubectl apply -f mysql/mysql-pvc.yml`.

2. Now, let us use the command `kubectl get pvc` to verify if the *mysql-pv-claim* PVC was correctly created. Note that the status for the PVC remains *Pending*.

3. Next, we need to set up a MySQL database for the Swap application.
   The file *mysql/mysql-deployment.yml* defines a Deployment object that specifies the Pod template for deploying MySQL and is expecting a PVC named *mysql-pv-claim*. Run the following command, at the Master VM, to create the MySQL Deployment:
   ```
   kubectl apply -f mysql/mysql-deployment.yml
   ```

4. Use command `kubectl get all` to verify the objects that were created.
   It should show a ReplicaSet, a Deployment and a Pod for MySQL.
   Explore the following commands to get informations about a specific MSQL object:

   - `kubectl get deployment [DEPLOYMENT_NAME]`
   - `kubectl get replicaset [REPLICASET_NAME]`
   - `kubectl get pod [POD_NAME]`

5. Check again the status of the PVC `kubectl get pvc`. If correctly configured, the status should now appear as *Bound* the Volume column should have the name of one of the PVs created previously (*pv001* or *pv002*).

6. Rerun the command `kubectl get all` and verify if all MySQL objects are ready.

7. Now, run the command `kubectl apply -f mysql/mysql-service.yml` to create a Service for exposing MySQL to the other Pods in the cluster. When executing the command `kubectl get svc`, there should be a Service named *mysql-service* with the type ClusterIP.

8. Run the following command to verify if you can access the MySQL database:
   ```
   kubectl exec -it <MYSQL_POD_NAME> -- mysqladmin --host=0.0.0.0 --user=<
       MYSQL_USER> --password=<MYSQL_PASSWORD> status
   ```

9. Explore the `--selector` option to select objects with a specific label. For example, the command `kubectl get all --selector=tier=database` will show the objects with the label *tier=database*.

Swap Container Deployment

Deploy the Swap application in K8s. For this, create a Deployment object and a Service object for Swap. Take in account the following guidelines:

1. The Swap Pod should be configured to use the Docker image *ascnuminho/swap* and should set the Environment Variables `DB_HOST`, `DB_DATABASE`, `DB_USERNAME`, `DB_PASSWORD`, `MIGRATE` to the correct values. Also, it should expose the port where the Swap application is listening to (port 8000).

2. The Swap Service object should be of type NodePort to allow opening a specific port (e.g., 30007) on all the Nodes and forward any external traffic sent to this port to the Swap Service.
3. Use `kubectl` to deploy the aforementioned K8s objects.
4. Try it out! Access Swap from your browser. The url should contain the IP of one of the nodes and port defined on the Swap Service (e.g., `<IP_MASTER>:30007`). Note that you can also access Swap through the IP addresses of Worker VMs.
5. Use the command `kubectl exec -it <SWAP_POD_NAME> -- php artisan db:seed` to seed the database.

Extras

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.