Hosting a two-tier application with Kubernetes

How to host a two-tier "MySQL-WordPress" application using Kubernetes:

Step 1: Create a deployment to manage MySQL image pods

Define the deployment attributes in a YAML file

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  selector:
    matchLabels:
      app: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - image: mysql:5.6
        name: mysql
        env:
          # Use secret in real usage
        - name: MYSQL_ROOT_PASSWORD
          value: centos
        - name: MYSQL_USER
          value: user1
        - name: MYSQL_PASSWORD
          value: linux
        - name: MYSQL_DATABASE
          value: mydata
        ports:
        - containerPort: 3306
          name: mysql
```

File contents:

- apiVersion: apps/v1 - Specifies the Kubernetes API version being used.
- kind: Deployment - Indicates this is a Deployment resource.
- metadata:
- name: mysql - Names the deployment "mysql".
- spec:
- selector:
- matchLabels: app: mysql - Selects pods with the label "app: mysql".
- strategy: type: Recreate - Specifies the update strategy. "Recreate" means all existing pods will be killed before new ones are created.
- template: Defines the pod template.
- metadata:
- labels: app: mysql - Labels the pods with "app: mysql".
- spec:
- containers: Specifies the containers in the pod.
- image: mysql:5.6 - Uses MySQL version 5.6.
- name: mysql - Names the container "mysql".
- env: Sets environment variables:
- MYSQL_ROOT_PASSWORD: centos
- MYSQL_USER: user1
- MYSQL_PASSWORD: linux
- MYSQL_DATABASE: mydata
- ports:
- containerPort: 3306 - Exposes MySQL's default port.
- name: mysql - Names this port "mysql".

Create a deployment using the above file

```
controlplane $ kubectl create -f mydatabase.yaml
deployment.apps/mysql created
```

Successfully created deployment “mysql” to manage the application’s database pods with “mysql” image

```
controlplane $ kubectl get deployments.apps
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
mysql   1/1     1            1           11s
```

pod managed by the the deployment “mysql”

```
controlplane $ kubectl get pod
NAME                     READY   STATUS    RESTARTS   AGE
mysql-7f68d9d7f9-rzjx9   1/1     Running   0          12s
```

Step 2: Scale up the mysql pods as required

Now, let’s scale up the pods to create 3 replicas

```
controlplane $ kubectl scale deployment mysql --replicas 3
deployment.apps/mysql scaled
```

3 replicas of the mysql pod

```
controlplane $ kubectl get pod
NAME                     READY   STATUS    RESTARTS   AGE
mysql-7f68d9d7f9-cppg9   1/1     Running   0          3m58s
mysql-7f68d9d7f9-rzjx9   1/1     Running   0          5m57s
mysql-7f68d9d7f9-wvcqr   1/1     Running   0          3m58s
```

Step 3: Create a ClusterIP service for the mysql deployment:

Expose the deployment to create a ClusterIP service for the mysql pods. This will make the deployment accessible to the clients within the cluster

```
controlplane $ kubectl expose deployment mysql --port=3306 --target-port=3306 --type=ClusterIP
service/mysql exposed
controlplane $ kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP    3d19h
mysql        ClusterIP   10.105.72.136   <none>        3306/TCP   6s
```

Step 4: Create a deployment to manage wordpress image pods

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  selector:
    matchLabels:
      app: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - image: mysql:5.6
        name: mysql
        env:
          # Use secret in real usage
        - name: MYSQL_ROOT_PASSWORD
          value: centos
        - name: MYSQL_USER
          value: user1
        - name: MYSQL_PASSWORD
          value: linux
        - name: MYSQL_DATABASE
          value: mydata
        ports:
        - containerPort: 3306
          name: mysql
```

In a file, define the attributes of the deployment of WordPress application

- apiVersion: apps/v1 - Specifies the Kubernetes API version being used.
- kind: Deployment - Indicates this is a Deployment resource, which manages a replicated application.
- metadata:
- name: mysql - Names the deployment "mysql".
- spec: Defines the desired state for this deployment.
- selector: matchLabels: app: mysql - Selects which pods are managed by this deployment.
- strategy: type: Recreate - Specifies the update strategy. "Recreate" means all existing pods will be killed before new ones are created.
- template: Defines the pod template.
- metadata: labels: app: mysql - Labels the pods with "app: mysql".
- spec: containers: Specifies the containers in each pod.
- image: mysql:5.6 - Uses MySQL version 5.6 Docker image.
- name: mysql - Names the container "mysql".
- env: Sets environment variables for the MySQL container:
- MYSQL_ROOT_PASSWORD: centos
- MYSQL_USER: user1
- MYSQL_PASSWORD: linux
- MYSQL_DATABASE: mydata
- ports:
- containerPort: 3306 - Exposes MySQL's default port.
- name: mysql - Names this port "mysql".

Now, create a deployment for the worpdress

```
controlplane $ kubectl create -f wordpress.yaml
deployment.apps/wordpress created
```

Pod managed by the deployment wordpress created successfully

```
controlplane $ kubectl get pod
NAME                        READY   STATUS    RESTARTS   AGE
mysql-7f68d9d7f9-cppg9      1/1     Running   0          3m58s
mysql-7f68d9d7f9-rzjx9      1/1     Running   0          5m57s
mysql-7f68d9d7f9-wvcqr      1/1     Running   0          3m58s
wordpress-f97fc676-4mvtc    1/1     Running   0          33s
```

Step 5: scale up the wordpress pods :

Scale up the pods as required

```
controlplane $ kubectl scale deployment wordpress --replicas 3
deployment.apps/wordpress scaled
```

3 replicas of wordpress pods successfully created

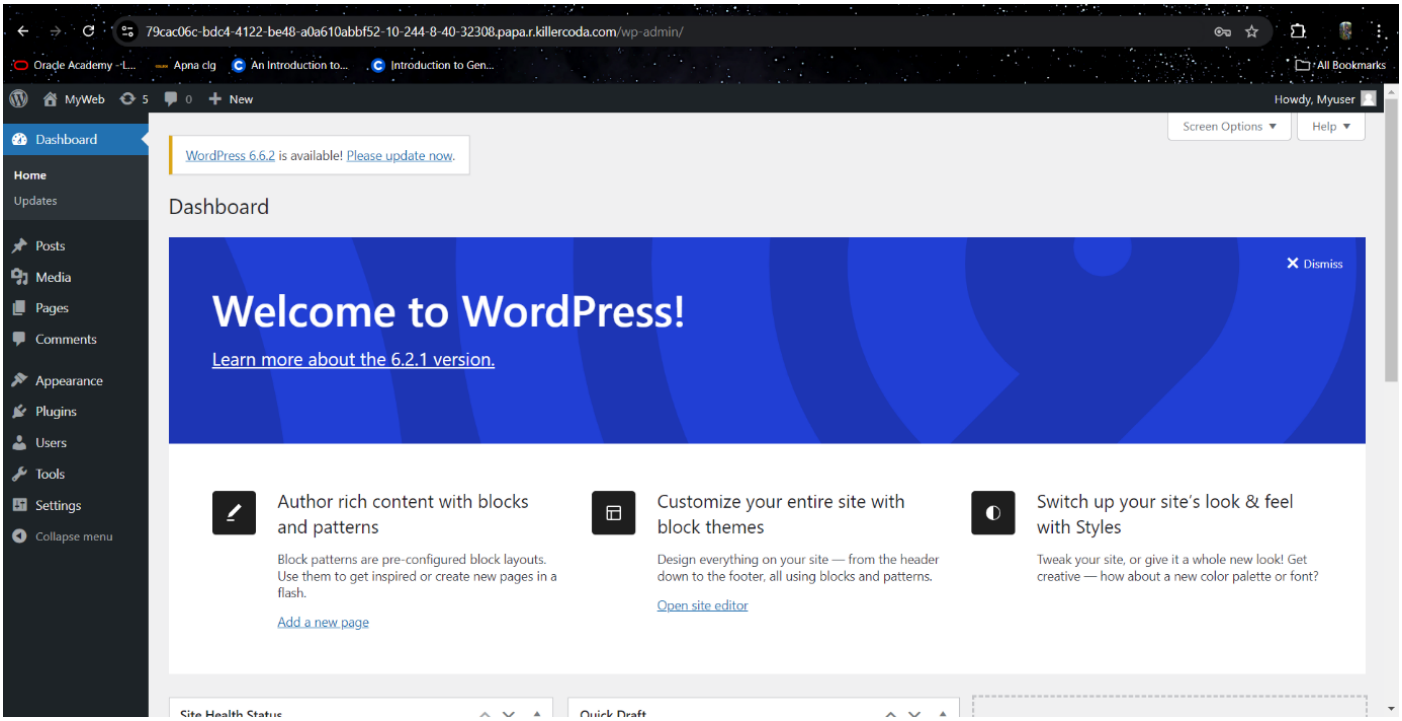
```
controlplane $ kubectl get pod
NAME                        READY   STATUS    RESTARTS   AGE
mysql-7f68d9d7f9-cppg9      1/1     Running   0          5m14s
mysql-7f68d9d7f9-rzjx9      1/1     Running   0          7m13s
mysql-7f68d9d7f9-wvcqr      1/1     Running   0          5m14s
wordpress-f97fc676-4mvtc    1/1     Running   0          109s
wordpress-f97fc676-89g8w    1/1     Running   0          37s
wordpress-f97fc676-wwxhj    1/1     Running   0          37s
```

Step 6: Creating a NodePort service for the Wordpress deployment

Now expose the deployment to create a NodePort service this will make the deployment accessible to the clientsoutside the cluster

```
controlplane $ kubectl expose deployment wordpress --port 80 --target-port 80 --type NodePort
service/wordpress exposed
controlplane $ kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        3d19h
mysql        ClusterIP   10.105.72.136   <none>        3306/TCP       7m11s
wordpress    NodePort    10.106.67.187   <none>        80:32308/TCP   10s
```

Thus, the application is successfully hosted



When we login to the database using the credentials we assigned, we can see the data of the wordpress app successfully accessed by the mysql database hosted at the mysql pods of the created deployment deployment

mydata is the database linked with the wordpress application

```
controlplane $ mysql -uuser1 -plinux -h10.105.72.136
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 5.6.51 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mydata             |
+--------------------+
2 rows in set (0.006 sec)
```

Wordpress data tables in mysql database

```
MySQL [(none)]> use mydata;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [mydata]> show tables;
+-----------------------+
| Tables_in_mydata      |
+-----------------------+
| wp_commentmeta        |
| wp_comments           |
| wp_links              |
| wp_options            |
| wp_postmeta           |
| wp_posts              |
| wp_term_relationships |
| wp_term_taxonomy      |
| wp_termmeta           |
| wp_terms              |
| wp_usermeta           |
| wp_users              |
+-----------------------+
12 rows in set (0.001 sec)
```

Thus, we have successfully deployed a two-tier “MySQL-WordPress” application using Kubernetes