Lesson 04 Demo 06

Deploying Multitier Applications Using Kubernetes

Objective: To deploy and verify a multitier WordPress and MySQL application on Kubernetes for managing, scaling, and maintaining them

Tools required: kubeadm, kubectl, kubelet, and containerd

Prerequisites: A Kubernetes cluster should already be set up (refer to the steps in Lesson 02, Demo 01 for guidance).

Steps to be followed:

1. Create a deployment for MySQL
2. Create a deployment for WordPress
3. Expose the service for WordPress and MySQL deployment
4. Verify the deployment of the application

Step 1: Create a deployment for MySQL

1.1 To create a MySQL database deployment, draft the following YAML code and save it in the **mysql.yaml** file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
labels:
    app: mysql
    name: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
 strategy: {}
 template:
```

```
  metadata:

  creationTimestamp: null
labels:
      app: mysql
   spec:
     containers:
     - image: mysql:5.6
       name: mysql
       env:
       - name: MYSQL_ROOT_PASSWORD
         value: simplilearn
       - name: MYSQL_DATABASE
         value: database1
       resources: {}
status: {}
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: mysql
  name: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: mysql
    spec:
      containers:
      - image: mysql:5.6
        name: mysql
        env:
        - name: MYSQL_ROOT_PASSWORD
          value: simplilearn
        - name: MYSQL_DATABASE
          value: database1
        resources: {}
status: {}

~
~
~
```

```
labsuser@master:~$ kubectl get nodes
NAME                       STATUS   ROLES           AGE   VERSION
master.example.com         Ready    control-plane   33h   v1.28.2
worker-node-1.example.com  Ready    <none>          32h   v1.28.2
worker-node-2.example.com  Ready    <none>          32h   v1.28.2
labsuser@master:~$ vi mysql.yaml
labsuser@master:~$ cat mysql.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: mysql
  name: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: mysql
    spec:
      containers:
      - image: mysql:5.6
        name: mysql
        env:
        - name: MYSQL_ROOT_PASSWORD
          value: simplilearn
        - name: MYSQL_DATABASE
          value: database1
        resources: {}
status: {}

labsuser@master:~$
```

1.2 To create a deployment for MySQL, use the following command:

kubectl create -f mysql.yaml

```
labsuser@master:~$ kubectl create -f mysql.yaml
deployment.apps/mysql created
labsuser@master:~$
```

1.3 To verify pods and deployments, run the following commands:

kubectl get pods

kubectl get deployments

```
labsuser@master:~$ kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
mysql-79c547d7fb-cjjsz   1/1     Running   0          5m10s
labsuser@master:~$ kubectl get deployments
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
mysql   1/1     1            1           6m34s
labsuser@master:~$
```

Upon execution, a deployment for MySQL will be created.

Step 2: Create a deployment for WordPress

2.1 To create a WordPress deployment, draft the following YAML code and save it in the **wordpress.yaml** file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: wordpress
  name: wordpress
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wordpress
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: wordpress
    spec:
      containers:
      - image: wordpress
        name: wordpress
```

```
      env:

      - name: WORDPRESS_DB_HOST
        value: mysql
      - name: WORDPRESS_DB_PASSWORD
        value: simplilearn
      - name: WORDPRESS_DB_USER

        value: root
      - name: WORDPRESS_DB_NAME
        value: database1
      resources: {}
status: {}
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: wordpress
  name: wordpress
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wordpress
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: wordpress
    spec:
      containers:
      - image: wordpress
        name: wordpress
        env:
        - name: WORDPRESS_DB_HOST
          value: mysql
        - name: WORDPRESS_DB_PASSWORD
          value: simplilearn
        - name: WORDPRESS_DB_USER
          value: root
        - name: WORDPRESS_DB_NAME
          value: database1
        resources: {}
status: {}
```

```
labsuser@master:~$ kubectl get deployments
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
mysql   1/1     1            1           6m34s
labsuser@master:~$ vi wordpress.yaml
labsuser@master:~$ cat wordpress.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: wordpress
  name: wordpress
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wordpress
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: wordpress
    spec:
      containers:
      - image: wordpress
        name: wordpress
        env:
        - name: WORDPRESS_DB_HOST
          value: mysql
        - name: WORDPRESS_DB_PASSWORD
          value: simplilearn
        - name: WORDPRESS_DB_USER
          value: root
        - name: WORDPRESS_DB_NAME
          value: database1
        resources: {}
status: {}

labsuser@master:~$
```

2.2 To create a deployment for WordPress, use the following command:
kubectl create -f wordpress.yaml

```
labsuser@master:~$ kubectl create -f wordpress.yaml
deployment.apps/wordpress created
labsuser@master:~$
```

2.3 To verify pods and deployments, run the following commands:
kubectl get pods
kubectl get deployments

```
labsuser@master:~$ kubectl get pods
NAME                         READY   STATUS    RESTARTS        AGE
mysql-79c547d7fb-cjjsz       1/1     Running   1 (9m27s ago)   13h
wordpress-78cbf57fdb-qbpld   1/1     Running   0               60s
labsuser@master:~$ kubectl get deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
mysql       1/1     1            1           13h
wordpress   1/1     1            1           67s
labsuser@master:~$
```

Upon execution, a deployment for WordPress will be created.

Step 3: Expose the service for WordPress and MySQL deployment

3.1 To expose the service for **WordPress** and **MySQL** deployment, run the following commands:
kubectl expose deployment mysql --port=3306
kubectl expose deployment wordpress --port=80

```
labsuser@master:~$ kubectl expose deployment mysql --port=3306
service/mysql exposed
labsuser@master:~$ kubectl expose deployment wordpress --port=80
service/wordpress exposed
labsuser@master:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP    46h
mysql        ClusterIP   10.97.40.149    <none>        3306/TCP   3m7s
wordpress    ClusterIP   10.111.42.189   <none>        80/TCP     2m12s
labsuser@master:~$
```

Note: Use the **kubectl get svc** command to list the services in the cluster

3.2 Change the service type for both **MySQL** and **WordPress** from **ClusterIP** to **NodePort** using the following commands:
kubectl edit svc mysql
kubectl edit svc wordpress

```
labsuser@master:~$ kubectl edit svc mysql
service/mysql edited
labsuser@master:~$ kubectl edit svc wordpress
service/wordpress edited
labsuser@master:~$
```

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: "2023-10-11T04:26:12Z"
  labels:
    app: mysql
  name: mysql
  namespace: default
  resourceVersion: "47249"
  uid: 99360c16-67f9-4018-8833-b96cdac9caa5
spec:
  clusterIP: 10.97.40.149
  clusterIPs:
  - 10.97.40.149
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - port: 3306
    protocol: TCP
    targetPort: 3306
  selector:
    app: mysql
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
~
~
~
~
~
~
~
~
~
~
~
~
:wq
```

Note: Edit service type from **ClusterIP** to **NodePort**

3.3 To verify the service type of **MySQL** and **WordPress**, run the following command:
kubectl get svc

```
labsuser@master:~$ kubectl edit svc wordpress
service/wordpress edited
labsuser@master:~$ kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP          46h
mysql        NodePort    10.97.40.149    <none>        3306:31999/TCP   14m
wordpress    NodePort    10.111.42.189   <none>        80:30376/TCP     13m
labsuser@master:~$
```

3.4 To get detailed information on the pods, use the following commands:
kubectl get pods -o wide
kubectl get nodes -o wide

```
labsuser@master:~$ kubectl get pods -o wide
NAME                        READY   STATUS    RESTARTS      AGE   IP               NODE                        NOMINATED NODE   READINESS GATES
mysql-79c547d7fb-cjjsz      1/1     Running   1 (33m ago)   13h   192.168.232.198  worker-node-2.example.com   <none>           <none>
wordpress-78cbf57fdb-qbpld  1/1     Running   0             25m   192.168.47.135   worker-node-1.example.com   <none>           <none>
labsuser@master:~$ kubectl get nodes -o wide
NAME                        STATUS   ROLES           AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION   CONTAINER-RUNTIME
master.example.com          Ready    control-plane   46h   v1.28.2   172.31.5.255    <none>        Ubuntu 22.04.3 LTS   6.2.0-1013-aws   containerd://1.6.8
worker-node-1.example.com   Ready    <none>          46h   v1.28.2   172.31.24.170   <none>        Ubuntu 22.04.3 LTS   6.2.0-1012-aws   containerd://1.6.8
worker-node-2.example.com   Ready    <none>          46h   v1.28.2   172.31.26.42    <none>        Ubuntu 22.04.3 LTS   6.2.0-1013-aws   containerd://1.6.8
labsuser@master:~$
```

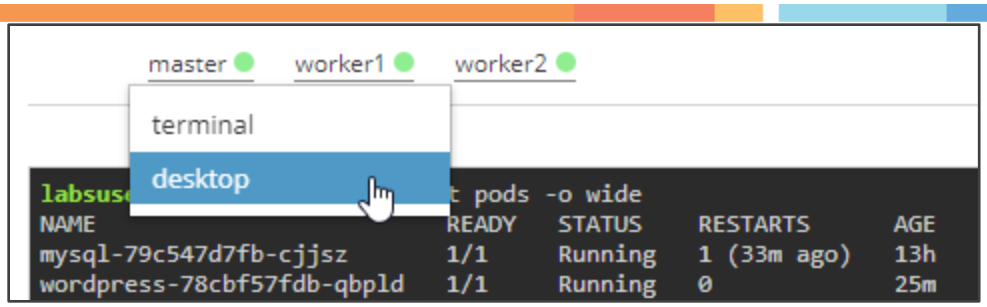
Note: Copy the following things for the next step:

1. WordPress pod IP address or the respective nodes' **Internal-IP**
2. Service NodePort of WordPress

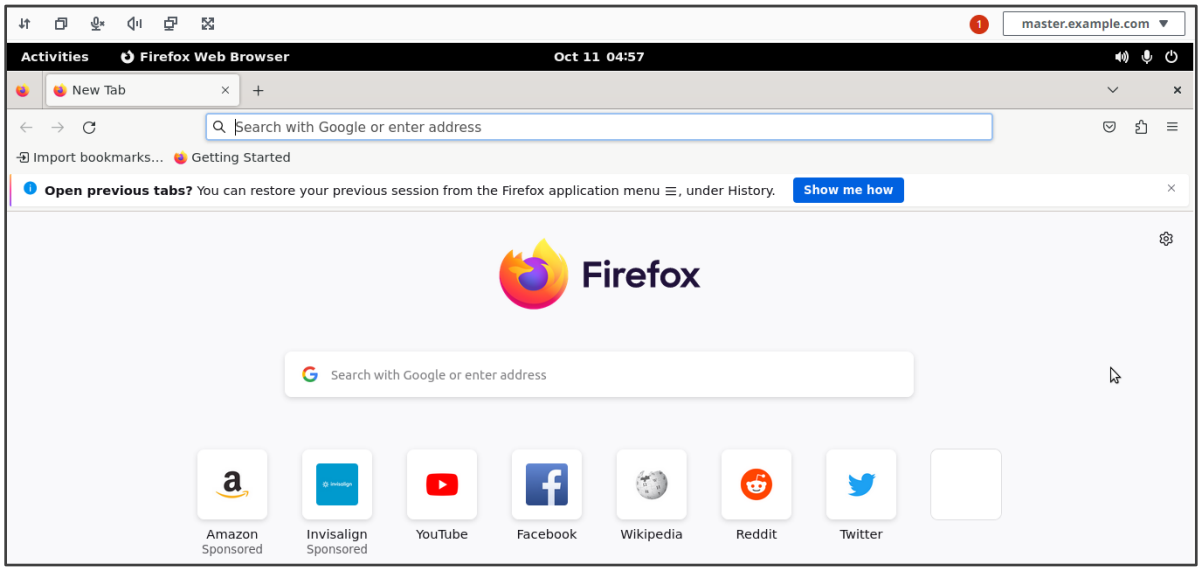
By executing the above steps, you will be able to expose the service for WordPress and MySQL deployment.

Step 4: Verify the deployment of the application

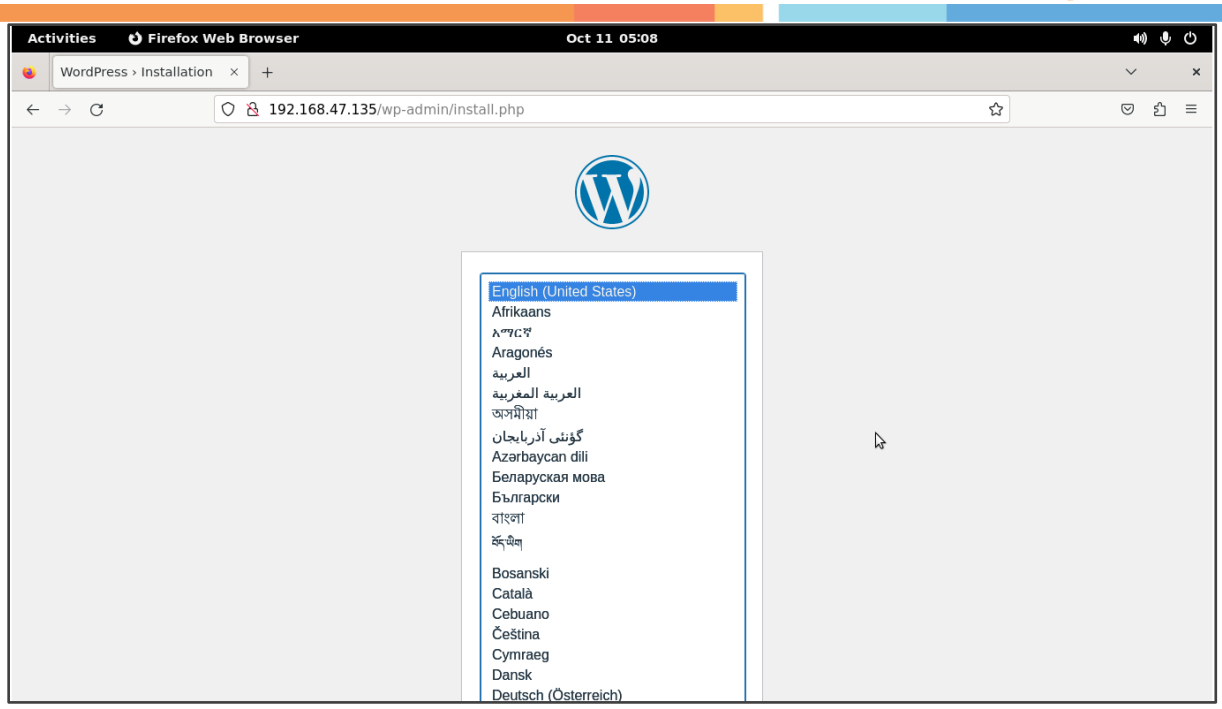
4.1 In the **master** node, go to **desktop** mode



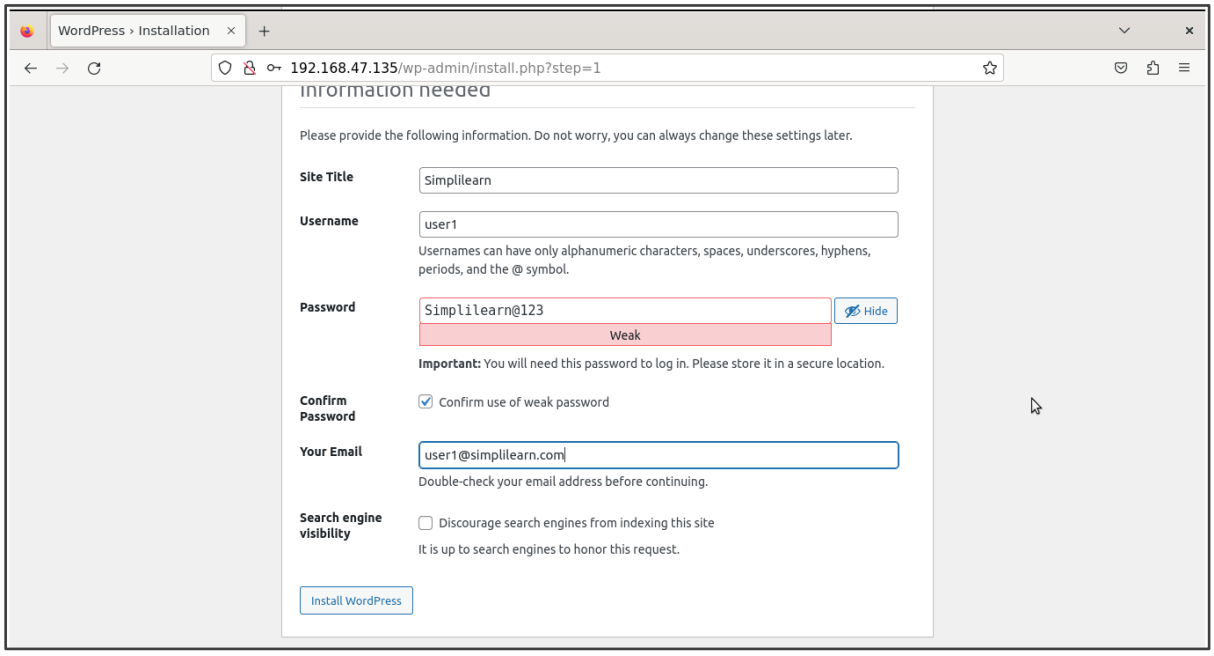
4.2 Open the **Firefox** web browser



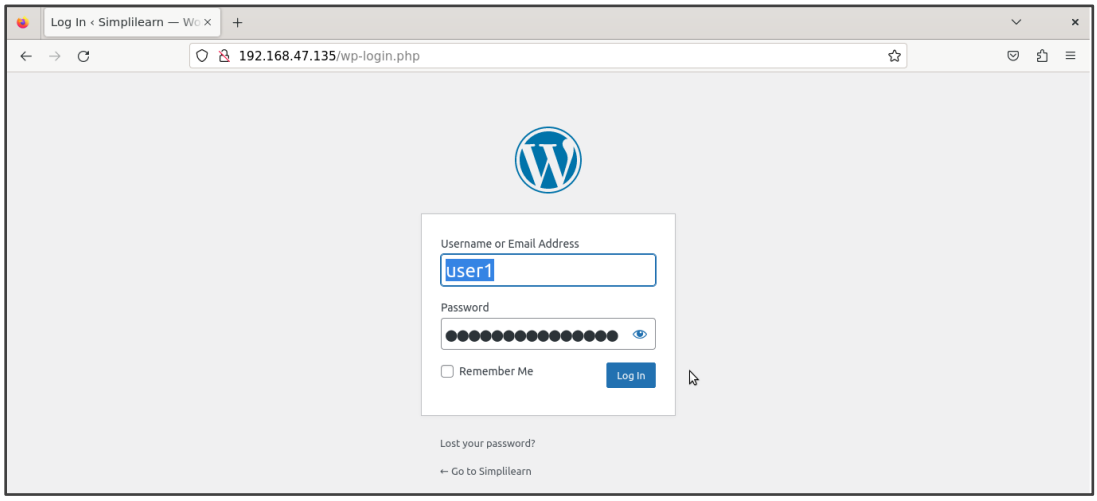
4.3 In the browser, enter the following IP address and service NodePort (refer to step 3.4):
172.31.24.170:30376 or 192.168.47.135:80



4.4 The WordPress default page will appear. Finish the installation process and provide the necessary account information. Click on **Install Wordpress.**



4.5 On the login page, enter the username and password provided during the installation process



The WordPress application has been successfully deployed, as seen in the screenshot above.

By following these steps, you have successfully deployed a multitier application using Kubernetes.