

Lesson 03 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 (refer to Demo 01 from Lesson 01 for setting up a cluster)

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:
  name: mysql-deployment
  labels:
    app: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
```

```

containers:
- name: mysql
  image: mysql:5.6
  env:
  - name: MYSQL_ROOT_PASSWORD
    value: simplilearn
  - name: MYSQL_DATABASE
    value: database1

```

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment
  labels:
    app: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - name: mysql
        image: mysql:5.6
        env:
        - name: MYSQL_ROOT_PASSWORD
          value: simplilearn
        - name: MYSQL_DATABASE
          value: database1

```

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 is 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:
  name: wordpress
  labels:
    app: wordpress
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wordpress
  template:
    metadata:
      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: {}
```

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

```

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-qbp1d        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 is 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-deployment --port=3306

kubectl expose deployment wordpress --port=80

```
labsuser@master:~$ kubectl expose deployment mysql-deployment --port=3306
service/mysql-deployment exposed
labsuser@master:~$ kubectl expose deployment wordpress --port=80
service/wordpress exposed
```

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-deployment

kubectl edit svc wordpress

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

```
# 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: {}  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~
```

Note: Change the 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
myapp-pod	0/1	Init:0/2	0	47m	192.168.47.131	worker-node-1.example.com	<none>	<none>
mysql-deployment-84dc46cb49-j8xn2	1/1	Running	0	33m	192.168.232.195	worker-node-2.example.com	<none>	<none>
wordpress-9b874c79f-fpt9r	1/1	Running	0	20m	192.168.47.132	worker-node-1.example.com	<none>	<none>

```
labsuser@master:~$ kubectl get nodes -o wide
```

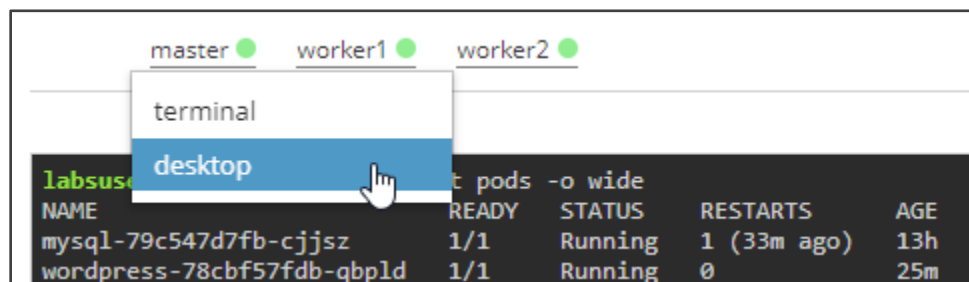
NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	KERNEL-VERSION	CONTAINER-RUNTIME
master.example.com	Ready	control-plane	19h	v1.30.4	172.31.37.115	<none>	Ubuntu 22.04.3 LTS	6.2.0-1013-aws	containerd://1.6.8
worker-node-1.example.com	Ready	<none>	19h	v1.30.4	172.31.33.72	<none>	Ubuntu 22.04.3 LTS	6.2.0-1013-aws	containerd://1.6.8
worker-node-2.example.com	Ready	<none>	19h	v1.30.4	172.31.38.39	<none>	Ubuntu 22.04.3 LTS	6.2.0-1013-aws	containerd://1.6.8

Note: Copy the following things to your Notepad for the next step:

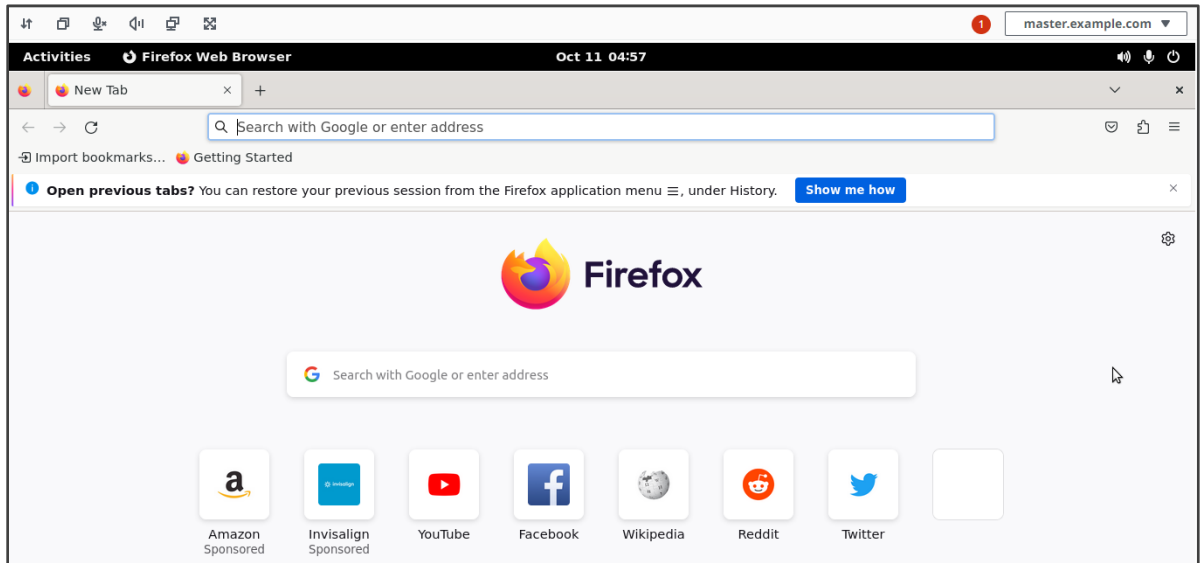
1. WordPress pod IP address or the respective nodes' **Internal-IP** (for example, 192.168.47.132)
2. Service NodePort of WordPress (for example, 172.31.33.72)

Step 4: Verify the deployment of the application

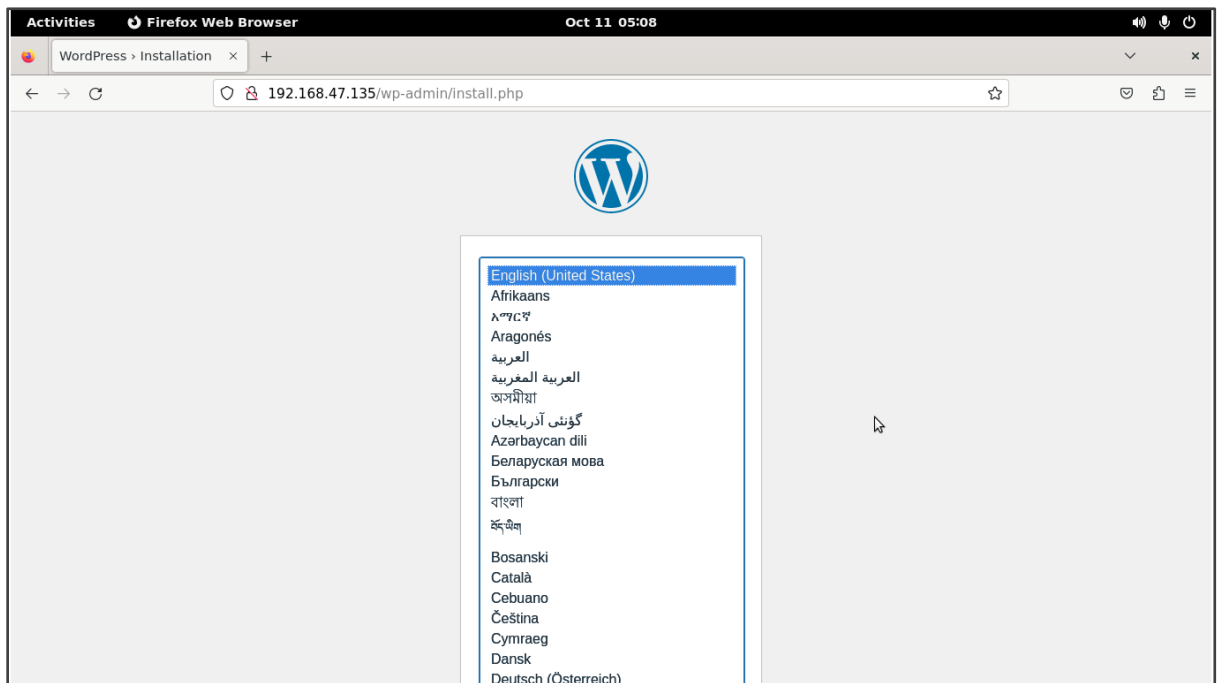
4.1 In the **master** node, go to **desktop** mode



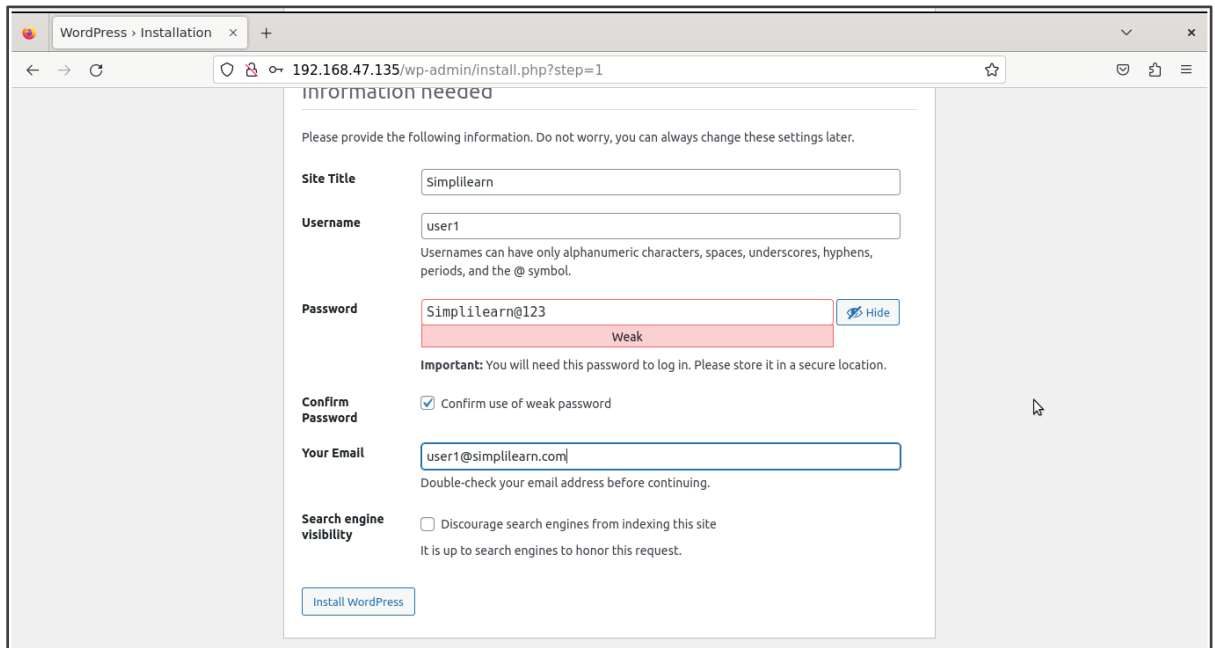
4.2 Open the **Firefox** web browser



4.3 In the browser, enter the IP address and service NodePort (refer to step 3.4) Example: 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**

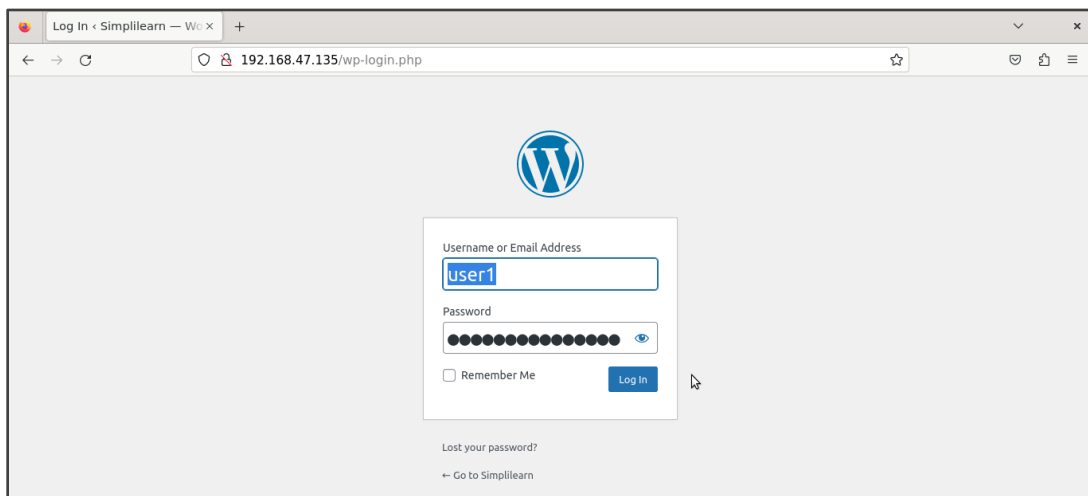


The screenshot shows the WordPress installation 'Information needed' screen. The browser address bar indicates the URL is 192.168.47.135/wp-admin/install.php?step=1. The form contains the following fields and options:

- Site Title:** A text input field containing 'Simplilearn'.
- Username:** A text input field containing 'user1'. Below it, a note states: 'Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.'
- Password:** A text input field containing 'Simplilearn@123'. To its right is a 'Hide' button. Below the field, a red box indicates the password is 'Weak'.
- Important:** A message states: 'You will need this password to log in. Please store it in a secure location.'
- Confirm Password:** A checkbox labeled 'Confirm use of weak password' is checked.
- Your Email:** A text input field containing 'user1@simplilearn.com'. Below it, a note states: 'Double-check your email address before continuing.'
- Search engine visibility:** A checkbox labeled 'Discourage search engines from indexing this site' is unchecked. Below it, a note states: 'It is up to search engines to honor this request.'

At the bottom of the form is a blue button labeled 'Install WordPress'.

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



The screenshot shows the WordPress login page. The browser address bar indicates the URL is 192.168.47.135/wp-login.php. The page features the WordPress logo at the top center. Below the logo is a login form with the following elements:

- Username or Email Address:** A text input field containing 'user1'.
- Password:** A password input field with masked characters (dots) and a toggle icon to the right.
- Remember Me:** An unchecked checkbox.
- Log In:** A blue button.

Below the login form, there are two links: 'Lost your password?' and '← Go to Simplilearn'.

The WordPress application is successfully deployed as seen in the screenshot above.

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