Lesson-End Project

Installing MySQL and WordPress in a Kubernetes Environment

Project agenda: To deploy a MySQL service and install a WordPress application, ensuring a robust, integrated web platform that leverages secure database management and dynamic content management capabilities

Description: Your team lead has given you the task of deploying a MySQL and WordPress application with specific requirements. You need to add all users using ConfigMaps and incorporate all sensitive data using secrets. The service should be configured to run on NodePort. Additionally, ensure that the WordPress application only deploys once the MySQL service is verified to be up and running.

Tools required: kubeadm, kubectl, kubelet, and containerd

Prerequisites: A Kubernetes cluster (refer to Demo 01 from Lesson 01 for setting up a cluster)

Expected deliverables: A WordPress Application running successfully

Steps to be followed:

1. Create a secret for storing the MySQL password securely
2. Create a MySQL YAML file
3. Deploy the WordPress application
4. Access the WordPress application using NodePort

Step 1: Create a secret for storing the MySQL password securely

1.1 Run the command below to encrypt your password with base64 encoding to make it more secure:
echo -n 'mysql@Temp123' | base64

```
labsuser@master:~$ echo -n 'mysql@Temp123' | base64
bXlzcWxAVGVtcDEyMw==
```

1.2 Enter the following command to create a YAML file:
nano secret.yaml

```
labsuser@master:~$ nano secret.yaml
```

1.3 Add the following code to the YAML file to save the generated password to the secret:

apiVersion: v1
kind: Secret
metadata:
name: mysql-secret-password
type: kubernetes.io/basic-auth
stringData:
username: root
password: bXlzcWxAVGVtcDEyMw==

```
  GNU nano 6.2                                                    secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysql-secret-password
type: kubernetes.io/basic-auth
stringData:
  username: root
  password: bXlzcWxAVGVtcDEyMw==
```

1.4 Run the following command to create a secret:
kubectl apply -f secret.yaml

```
labsuser@master:~$ kubectl apply -f secret.yaml
secret/mysql-secret-password created
```

1.5 Run the following command to describe the secret:
kubectl describe secret mysql-secret-password

```
labsuser@master:~$ kubectl describe secret mysql-secret-password

Name:         mysql-secret-password
Namespace:    default
Labels:       <none>
Annotations:  <none>

Type:  kubernetes.io/basic-auth

Data
====
password:  20 bytes
username:  4 bytes
```

Step 2: Create a MySQL YAML file

2.1 Run the following command to create a YAML file:
nano mysql.yaml

```
labsuser@master:~$ nano mysql.yaml
```

2.2 Add the following code to the YAML file:

apiVersion: v1
kind: Service
metadata:
name: mysql
labels:
app: mysql-wordpress
spec:
ports:
- port: 3306
selector:
app: mysql-wordpress
product: mysql
---
apiVersion: apps/v1
kind: Deployment
metadata:
name: mysql
labels:

```
app: mysql-wordpress
spec:
selector:
matchLabels:
app: mysql-wordpress
product: mysql
strategy:
type: Recreate
template:
metadata:
labels:
app: mysql-wordpress
product: mysql
spec:
containers:
- image: mysql
name: mysql-container
env:
- name: MYSQL_DATABASE
value: wordpress
- name: MYSQL_ROOT_PASSWORD
valueFrom:
secretKeyRef:
name: mysql-secret-password
key: password
ports:
- containerPort: 3306
name: mysql
```

```
 GNU nano 6.2                                                    mysql.yaml
apiVersion: v1
kind: Service
metadata:
  name: mysql
  labels:
    app: mysql-wordpress
spec:
  ports:
    - port: 3306
  selector:
    app: mysql-wordpress
    product: mysql
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
  labels:
    app: mysql-wordpress
spec:
  selector:
    matchLabels:
      app: mysql-wordpress
      product: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql-wordpress
        product: mysql
    spec:
      containers:
      - image: mysql
        name: mysql-container
        env:
        - name: MYSQL_DATABASE
```

2.3 Run the following command to create a service and deployment:
kubectl apply -f mysql.yaml

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

2.4 Run the following commands to verify the state of the pod, service, and deployment:
kubectl get deployments -o wide
kubectl get pods -o wide
kubectl get services -o wide

```
labsuser@master:~$ kubectl get deployments -o wide
NAME         READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS        IMAGES                    SELECTOR
mysql        0/1     1            0           13s   mysql-container   mysql                     app=mysql-wordpress,product=mysql
php-apache   1/1     1            1           22h   php-apache        k8s.gcr.io/hpa-example    run=php-apache
```

```
labsuser@master:~$ kubectl get pods -o wide
NAME                          READY   STATUS    RESTARTS      AGE   IP               NODE                       NOMINATED NODE   READINESS GATES
frontend-6xkgb                1/1     Running   2 (22h ago)   22h   192.168.47.148   worker-node-1.example.com  <none>           <none>
frontend-7q6qg                1/1     Running   2 (35m ago)   22h   192.168.47.147   worker-node-1.example.com  <none>           <none>
frontend-bltgs                1/1     Running   2 (35m ago)   22h   192.168.232.206  worker-node-2.example.com  <none>           <none>
mysql-7748c687bf-n9gdf        1/1     Running   0             30s   192.168.232.211  worker-node-2.example.com  <none>           <none>
php-apache-5f9f45d488-d4lv7   1/1     Running   1 (22h ago)   22h   192.168.47.145   worker-node-1.example.com  <none>           <none>
pod-env-var                   1/1     Running   2 (35m ago)   23h   192.168.47.149   worker-node-1.example.com  <none>           <none>
pod-env12                     1/1     Running   2 (35m ago)   23h   192.168.232.207  worker-node-2.example.com  <none>           <none>
testconfig                    0/1     Unknown   0             23h   <none>           worker-node-2.example.com  <none>           <none>
```

```
labsuser@master:~$ kubectl get services -o wide
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE   SELECTOR
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP    24h   <none>
mysql        ClusterIP   10.105.69.161   <none>        3306/TCP   45s   app=mysql-wordpress,product=mysql
php-apache   ClusterIP   10.96.172.52    <none>        80/TCP     22h   run=php-apache
labsuser@master:~$ kubectl logs mysql-7748c687bf-n9gdf
2023-10-13 10:48:06+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.1.0-1.el8 started.
```

2.5 Run the following command to check the MySQL logs to ensure that they are running properly in the container:
kubectl logs <mysql-pod-name>

```
labsuser@master:~$ kubectl logs mysql-7748c687bf-n9gdf
2023-10-13 10:48:06+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.1.0-1.el8 started.
2023-10-13 10:48:06+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2023-10-13 10:48:06+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.1.0-1.el8 started.
2023-10-13 10:48:06+00:00 [Note] [Entrypoint]: Initializing database files
2023-10-13T10:48:06.922493Z 0 [System] [MY-015017] [Server] MySQL Server Initialization - start.
2023-10-13T10:48:06.924578Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future release. Please use SET GLOBAL host_cache_size=0
 instead.
2023-10-13T10:48:06.924685Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.1.0) initializing of server in progress as process 80
2023-10-13T10:48:06.934099Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2023-10-13T10:48:07.731327Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
2023-10-13T10:48:10.121245Z 6 [Warning] [MY-010453] [Server] root@localhost is created with an empty password ! Please consider switching off the --initialize-insecure option.
2023-10-13T10:48:16.551460Z 0 [System] [MY-015018] [Server] MySQL Server Initialization - end.
2023-10-13 10:48:16+00:00 [Note] [Entrypoint]: Database files initialized
2023-10-13 10:48:16+00:00 [Note] [Entrypoint]: Starting temporary server
2023-10-13T10:48:16.604174Z 0 [System] [MY-015015] [Server] MySQL Server - start.
2023-10-13T10:48:16.918158Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future release. Please use SET GLOBAL host_cache_size=0
 instead.
2023-10-13T10:48:16.920538Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.1.0) starting as process 122
2023-10-13T10:48:16.939461Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2023-10-13T10:48:17.181210Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
2023-10-13T10:48:17.565265Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
2023-10-13T10:48:17.565306Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel.
2023-10-13T10:48:17.568139Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosin
g a different directory.
2023-10-13T10:48:17.587739Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Socket: /var/run/mysqld/mysqlx.sock
2023-10-13T10:48:17.588003Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.1.0'  socket: '/var/run/mysqld/mysqld.sock'  port: 0  MySQL Community Ser
ver - GPL.
```

Step 3: Deploy the WordPress application

3.1 Run the command **nano wordpress.yaml** to create a YAML file

```
labsuser@master:~$ nano wordpress.yaml
```

3.2 Add the following code to the YAML file:

apiVersion: v1
kind: Service
metadata:
name: wordpress
labels:
app: mysql-wordpress
spec:
ports:
- port: 80

```
selector:
app: mysql-wordpress
tier: frontend
type: NodePort
---
apiVersion: apps/v1
kind: Deployment
metadata:
name: wordpress
labels:
app: mysql-wordpress
spec:
selector:
matchLabels:
app: mysql-wordpress
tier: frontend
strategy:
type: Recreate
template:
metadata:
labels:
app: mysql-wordpress
tier: frontend
spec:
containers:
- image: wordpress
name: wordpress
env:
- name: WORDPRESS_DB_HOST
value: mysql
- name: WORDPRESS_DB_USER
value: root
- name: WORDPRESS_DB_PASSWORD

valueFrom:
secretKeyRef:
name: mysql-secret-password
key: password
ports:
- containerPort: 80
name: wordpress
```

```
 GNU nano 6.2                                                wordpress.yaml
apiVersion: v1
kind: Service
metadata:
  name: wordpress
  labels:
    app: mysql-wordpress
spec:
  ports:
    - port: 80
  selector:
    app: mysql-wordpress
    tier: frontend
  type: NodePort
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress
  labels:
    app: mysql-wordpress
spec:
  selector:
    matchLabels:
      app: mysql-wordpress
      tier: frontend
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql-wordpress
        tier: frontend
    spec:
      containers:
      - image: wordpress
        name: wordpress
```

3.3 Run the following command to create a WordPress service and deployment:
kubectl apply -f wordpress.yaml

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

3.4 Verify the state of the pods and services by running the following commands:
kubectl get services
kubectl get pods

```
labsuser@master:~$ kubectl get services
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP        24h
mysql        ClusterIP   10.105.69.161    <none>        3306/TCP       7m11s
php-apache   ClusterIP   10.96.172.52     <none>        80/TCP         22h
wordpress    NodePort    10.104.107.230   <none>        80:30674/TCP   50s
```

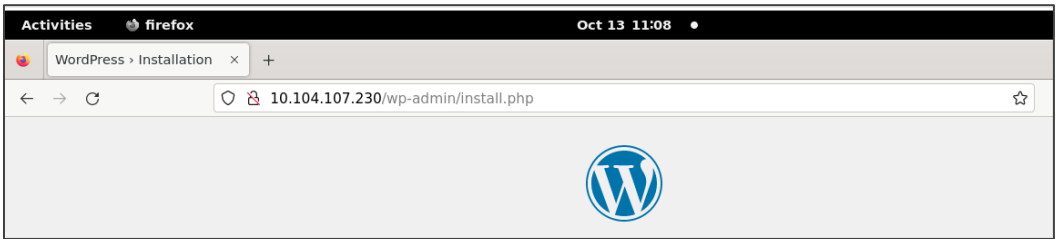
```
labsuser@master:~$ kubectl get pods
NAME                          READY   STATUS    RESTARTS      AGE
frontend-6xkgb                1/1     Running   2 (22h ago)   23h
frontend-7q6qg                1/1     Running   2 (42m ago)   23h
frontend-bltgs                1/1     Running   2 (42m ago)   23h
mysql-7748c687bf-n9gdf        1/1     Running   0             7m40s
php-apache-5f9f45d488-d4lv7   1/1     Running   1 (22h ago)   22h
pod-env-var                   1/1     Running   2 (42m ago)   23h
pod-env12                     1/1     Running   2 (42m ago)   23h
testconfig                    0/1     Unknown   0             23h
wordpress-6ff4d555d5-tglfv    1/1     Running   0             79s
```

Step 4: Access the WordPress application using NodePort

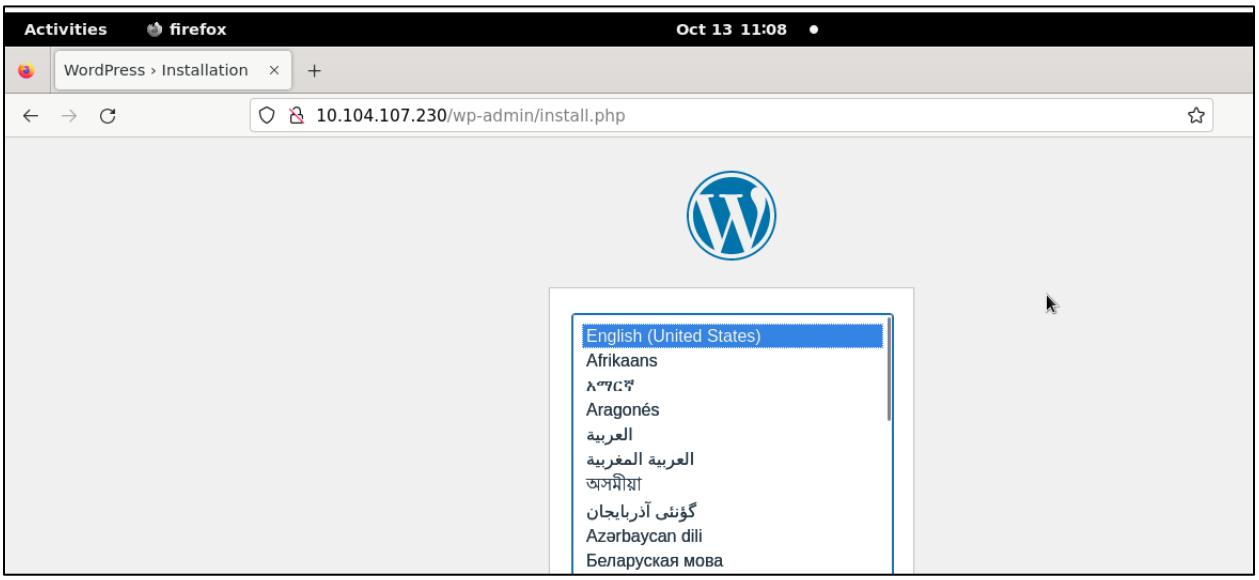
4.1 Use the command below to get the NodePort service and access the WordPress application:
kubectl get services

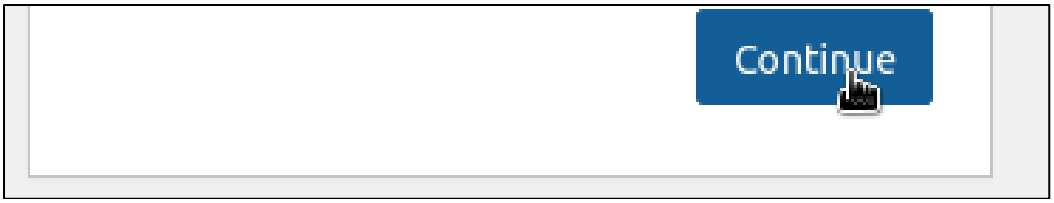
```
labsuser@master:~$ kubectl get services
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP        24h
mysql        ClusterIP   10.105.69.161    <none>        3306/TCP       8m28s
php-apache   ClusterIP   10.96.172.52     <none>        80/TCP         22h
wordpress    NodePort    10.104.107.230   <none>        80:30674/TCP   2m7s
```

4.2 Navigate to the desktop tab of the master node, open the Firefox browser, and then enter the URL **http://<node_port>** to access the WordPress application, replacing **<node_port>** with the cluster IP found in the previous step

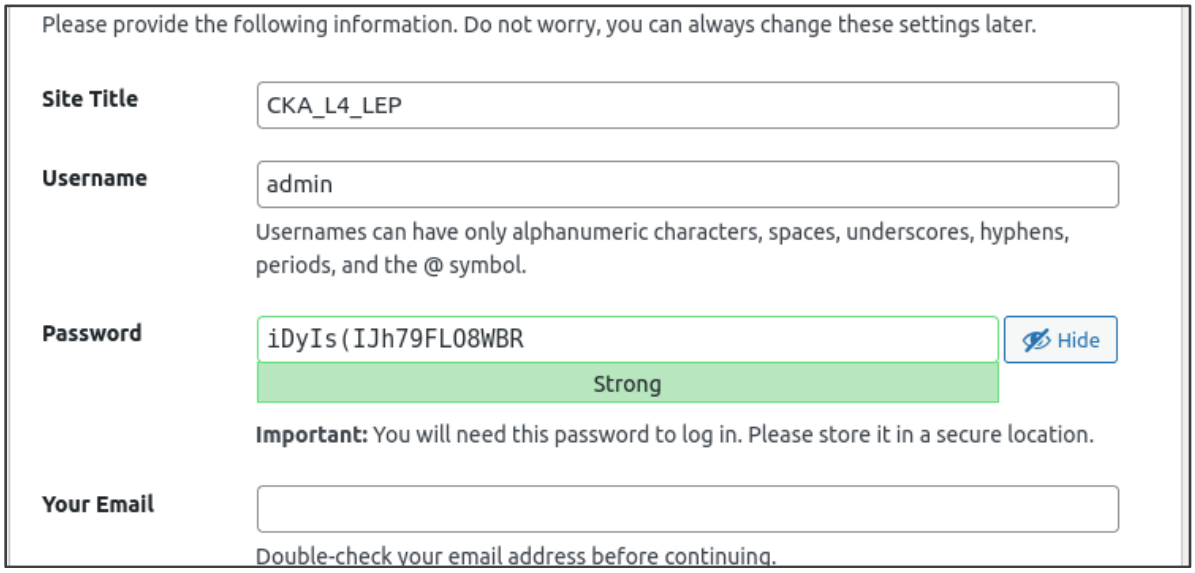


4.3 Once the WordPress application opens, choose English as the language and then click **Continue**

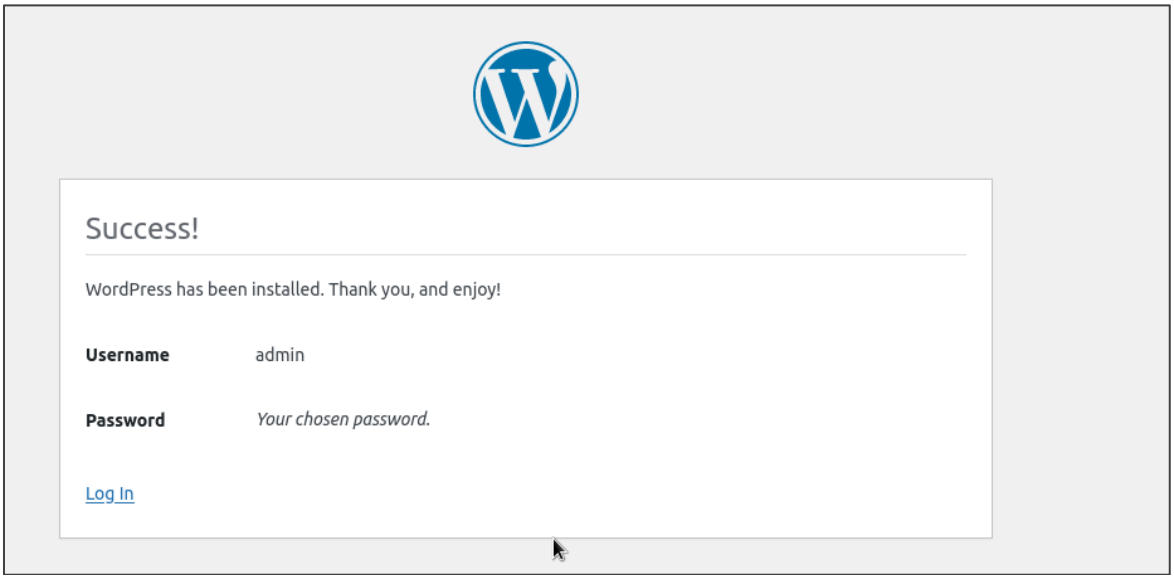


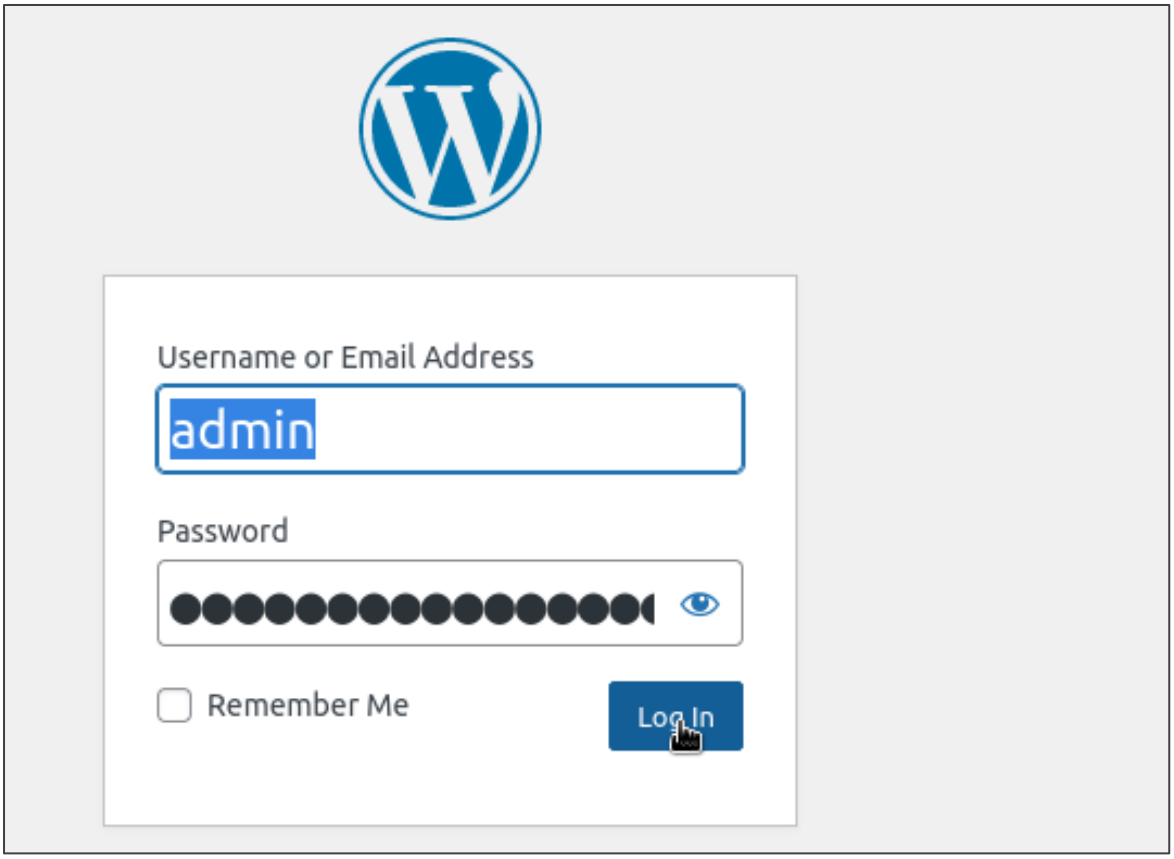


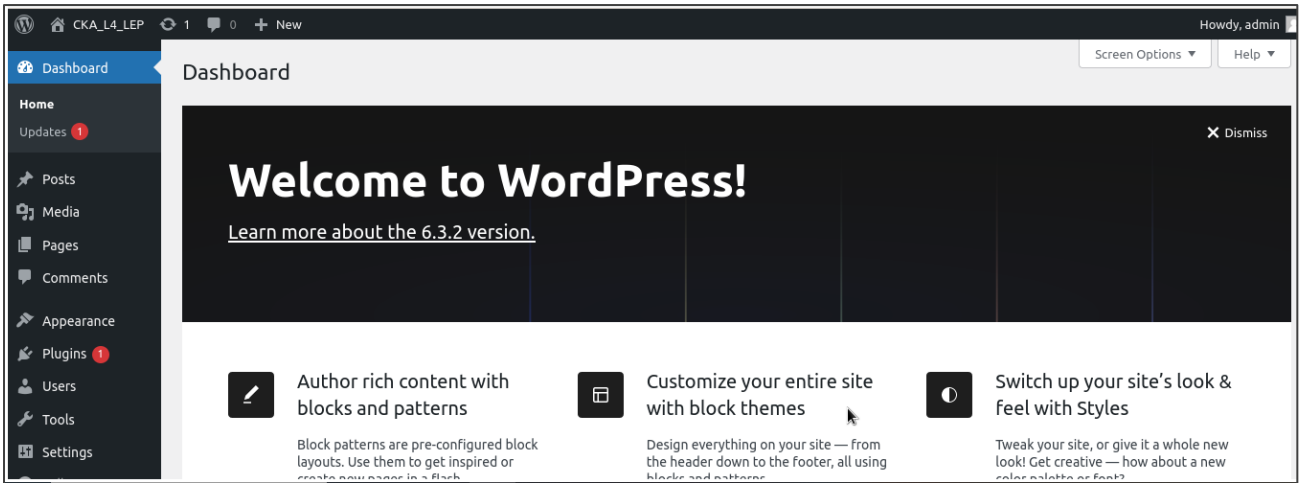
4.4 Provide the website-related information and proceed with the installation



4.5 Once the installation is complete, log in to the WordPress application







From the output, you can observe that the WordPress application is running successfully.

By following these steps, you have successfully deployed a MySQL service and installed a WordPress application, creating a robust, integrated web platform with secure database management and dynamic content management capabilities.