Wordpress Deployment on Kubernetes Cluster

1. **Create 2 Namespace**
2. **Create 2 deployment 1 for wordpress and 1 for MySql**
3. **Create 2 service**
    1. **Nodeport for wordpress**
    2. **ClusterIP for MySql**

1. Namespace creation

```
# kubectl create ns mywebsite
```

2. Create deployment for mysql

```
# vim mysql.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: mysql
 namespace: mywebsite
spec:
 replicas: 1
 selector:
  matchLabels:
   app: mysql
 template:
  metadata:
```

```
  name: sdf
  labels:
    app: mysql
spec:
  containers:
    - name: db
      image: mysql
      env:
            - name: MYSQL_ROOT_PASSWORD
              value: "redhat"
            - name: MYSQL_DATABASE
              value: "bigdata"
```

```
root@master-node:~# kubectl apply -f mysql.yaml
deployment.apps/mysql created
root@master-node:~#
root@master-node:~# kubectl get deployments.apps
No resources found in default namespace.
root@master-node:~#
root@master-node:~# kubectl get deployments.apps -n mywebsite
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
mysql   1/1     1            1           55s
root@master-node:~#
```

Now Lets create a service to communicate with deployment.

OR

We can create it maualy by command

```
kubectl expose deployment -n mywebsite mysql --port 3306 --target-port 3306 mysql_svc
```

```
root@master-node:~# kubectl expose deployment -n mywebsite mysql --port 3306 --target-port 33
06 mysql_svc
service/mysql exposed
Error from server (NotFound): deployments.apps "mysql_svc" not found
root@master-node:~#
root@master-node:~# kubectl get svc -n mywebsite
NAME    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
mysql   ClusterIP   10.105.248.252   <none>        3306/TCP   116s
root@master-node:~#
root@master-node:~#
```

```
root@master-node:~# kubectl describe svc -n mywebsite mysql
Name:              mysql
Namespace:         mywebsite
Labels:            <none>
Annotations:       <none>
Selector:          app=mysql
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.105.248.252
IPs:               10.105.248.252
Port:              <unset>  3306/TCP
TargetPort:        3306/TCP
Endpoints:         10.46.0.2:3306
Session Affinity:  None
Events:            <none>
```

```
root@master-node:~# mysql -u root -p -h 10.105.248.252
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 9.2.0 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)]>
```

As we can see that we can connect to myslq.

3. Create deployment for wordpress

```
apiVersion: apps/v1

kind: Deployment
```

```
metadata:
   name: wordpress
   namespace: mywebsite
spec:
   replicas: 1
   selector:
       matchLabels:
         app: wordpress
   template:
      metadata:
        name: sdf
        labels:
            app: wordpress
       spec:
        containers:
          - name: wp
             image: wordpress
             env:
                    - name: WORDPRESS_DB_HOST
                      value: "10.96.164.193"
                    - name: WORDPRESS_DB_USER
                      value: "root"
                    - name: WORDPRESS_DB_PASSWORD
                      value: "redhat"
                    -  name: WORDPRESS_DB_NAME
```

```
                        value: "bigdata"
```

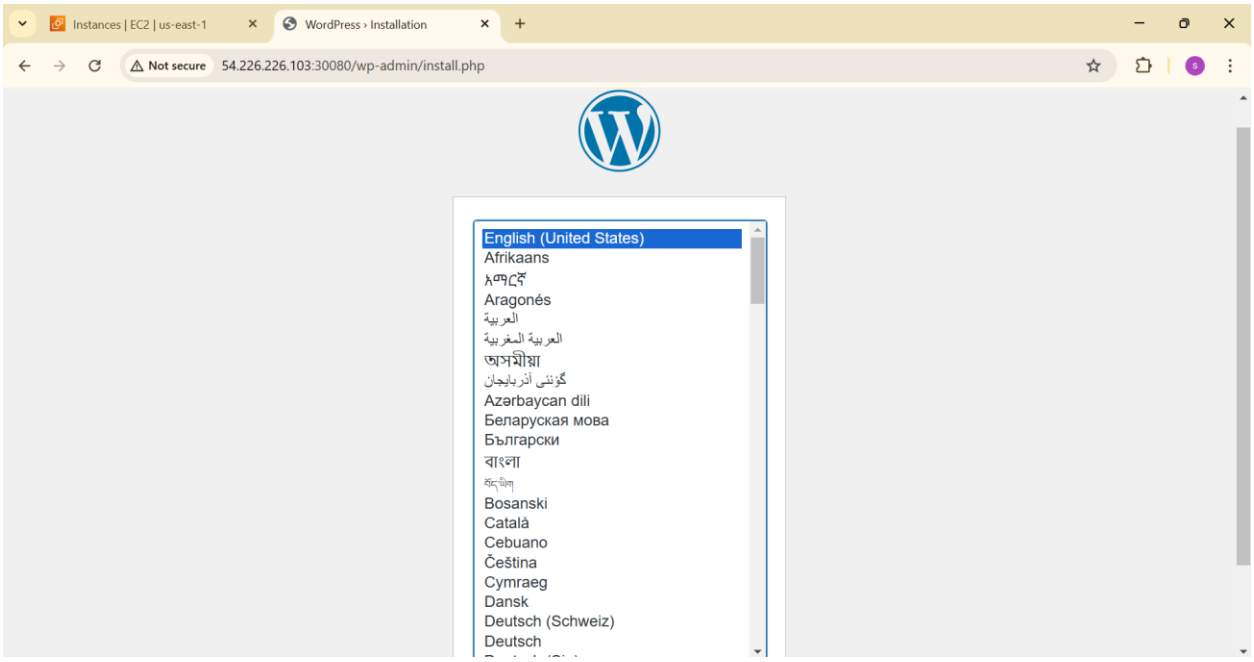
```
root@master-node:~# vim wordpress.yaml
root@master-node:~#
root@master-node:~# kubectl apply -f wordpress.yaml
deployment.apps/wordpress created
root@master-node:~#
root@master-node:~# kubectl get deployments.apps -n mywebsite
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
mysql       1/1     1            1           28m
wordpress   0/1     1            0           13s
root@master-node:~# kubectl get deployments.apps -n mywebsite
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
mysql       1/1     1            1           28m
wordpress   1/1     1            1           20s
root@master-node:~#
```

Now let's create a service for wordpress

```
apiVersion: v1
kind: Service
metadata:
 name: wp_svc
 namespace: mywebsite
spec:
 selector:
  app: wordpress
 type: NodePort
 ports:
  - protocol: TCP
    port: 80
    targetPort: 80
    nodePort: 30080
```

```
root@master-node:~# vim wpsvc.yaml
root@master-node:~# kubectl apply -f wpsvc.yaml
service/wpsvc created
root@master-node:~#
root@master-node:~# kubectl get svc -n mywebsite
NAME    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
mysql   ClusterIP   10.105.248.252   <none>        3306/TCP       27m
wpsvc   NodePort    10.104.241.140   <none>        80:30080/TCP   15s
root@master-node:~#
```

Now let's access wordpress on browser



ConfigMap

In Kubernetes, a **ConfigMap** is an API object used to store non-confidential configuration data in key-value pairs. It allows you to separate your application configuration from the container image, making your application more portable and manageable.

```
root@master-node:~# kubectl get configmaps
NAME               DATA   AGE
kube-root-ca.crt   1      24d
root@master-node:~#
```

Let's create a new configmap

```
kubectl create configmap app-db --from-literal MYSQL_ROOT_PASSWORD=redhat --from-literal MYSQL_DATABASE=test_db -n mywebsite
```

```
root@master-node:~# kubectl get configmaps -n mywebsite
NAME               DATA   AGE
app-db             2      47s
kube-root-ca.crt   1      6d18h
root@master-node:~#
```

```
root@master-node:~# kubectl describe configmaps app-db -n mywebsite
Name:         app-db
Namespace:    mywebsite
Labels:       <none>
Annotations:  <none>

Data
====
MYSQL_ROOT_PASSWORD:
----
redhat
MYSQL_DATABASE:
----
test_db

BinaryData
====

Events:  <none>
root@master-node:~#
```

```
kubectl create configmap app-wp -n mywebsite

--from-literal=WORDPRESS_DB_HOST= 10.110.46.197

--from-literal=WORDPRESS_DB_USER=root

--from-literal=WORDPRESS_DB_PASSWORD=redhat
```

```
--from-literal=WORDPRESS_DB_NAME=test_db
```

Now let's use configmap in deployment defination file.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
  namespace: mywebsite
spec:
  replicas: 1
  selector:
    matchLabels:
       app: mysql
  template:
    metadata:
      name: sdf
      labels:
        app: mysql
    spec:
      containers:
        - name: db
          image: mysql
          envFrom:
            - configMapRef:
                  name: app-db
```

```
root@master-node:~# vim mysql.yaml
root@master-node:~# kubectl apply -f mysql.yaml
deployment.apps/mysql created
root@master-node:~#
root@master-node:~# kubectl get deployments.apps
No resources found in default namespace.
root@master-node:~# kubectl get deployments.apps  -n mywebsite
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
mysql   0/1     1            0           12s
root@master-node:~# kubectl get deployments.apps  -n mywebsite
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
mysql   1/1     1            1           14s
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress
  namespace: mywebsite
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wordpress
  template:
    metadata:
      name: sdf
      labels:
        app: wordpress
    spec:
      containers:
        - name: wp
          image: wordpress
          envFrom:
                - configMapRef:
                      name: app-wp
```

```
root@master-node:~# kubectl apply -f wordpress.yaml
deployment.apps/wordpress created
root@master-node:~#
root@master-node:~# kubectl get deployments.apps  -n mywebsite
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
mysql       1/1     1            1           2m48s
wordpress   1/1     1            1           3s
root@master-node:~#
```

Let’s access wordpress