# **Deploy WordPress On Kubernetes**

If you have not done the MySQL setup then create MySQL Pod first before creating wordpress pod. Let’s deal with how to deploy the WordPress on the Kubernetes cluster and how to connect with the MySQL Pod. MySQL credentials are passed to the WordPress while creating the Pod configuration file for WordPress. So do not forget the MySQL username, password, and database name.

1. Create PVC for WordPress
2. Create Deployment for WordPress

3. Create Service for WordPress

4. Remove All Pods, PVCs, and Services

# ****1. Create PVC for WordPress****

WordPress stores Post data(text content) in MySQL. And stores the images on the hard disk. So WordPress also needs separate storage for storing images. So create PV and PVC for WordPress using the following code. Here PV is automatically created using the PVC.

apiVersion: v1  
kind: PersistentVolumeClaim  
metadata:  
 name: wordpress-volume  
spec:  
 accessModes:  
 - ReadWriteOnce  
 resources:  
 requests:  
 storage: 10Gi  
 storageClassName: block-storage-class-name

Save the code in a file name ****wp-volume.yaml****and execute using the following command.

kubectl apply -f wp-volume.yaml

Get the list of PV and PVC.

kubectl get pv

kubectl get pvc

Here the command displays both MySQL and WordPress PV and PVC.

# ****2. Create Deployment for WordPress****

---  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: wordpress  
spec:  
 replicas: 1  
 selector:  
 matchLabels:  
 app: wordpress  
 template:  
 metadata:  
 labels:  
 app: wordpress  
 spec:  
 containers:  
 - name: wordpress  
 image: wordpress:5.8.3-php7.4-apache  
 ports:  
 - containerPort: 80  
 name: wordpress  
 volumeMounts:  
 - name: wordpress-data  
 mountPath: /var/www  
 env:  
 - name: WORDPRESS\_DB\_HOST  
 value: mysql-service.default.svc.cluster.local  
 - name: WORDPRESS\_DB\_PASSWORD  
 valueFrom:  
 secretKeyRef:  
 name: wp-db-secrets  
 key: MYSQL\_ROOT\_PASSWORD  
 - name: WORDPRESS\_DB\_USER  
 value: root  
 - name: WORDPRESS\_DB\_NAME  
 value: wordpress  
 volumes:  
 - name: wordpress-data  
 persistentVolumeClaim:  
 claimName: wordpress-volume

Here a few important parameters.

1. ****image**** => It tells what Docker image to install. Use the latest version of WordPress from Docker Hub.

2. ****volumeMounts**** => Use to mount the path of WordPress folder and send the data(images, pdfs) to PV.

3. ****WORDPRESS\_DB\_HOST**** => Used to connect the MySQL Pod. The value for this parameter will be created using the following syntax.

#syntaxservice-name.namespace.svc.cluster.local

#Actual Valuemysql-service.default.svc.cluster.local

Save the code in a file name ****wp.yaml****and execute using the following command.

kubectl apply -f wp.yaml

The WordPress Pod is created.

Get the list of running Pods using the following command.

kubectl get pods

# ****3. Create Service for WordPress****

The WordPress Pod is up and running. You can not directly access the WordPress Pod. If you want to access the WordPress application, then you need to create a Load Balancer service for the WordPress Pod.

Create a Load Balancer service for WordPress Pod using the following code.

---  
kind: Service  
apiVersion: v1  
metadata:  
 name: wordpress-service  
spec:  
 type: LoadBalancer  
 selector:  
 app: wordpress  
 ports:  
 - name: http  
 protocol: TCP  
 port: 80  
 targetPort: 80

Save the code in a file name ****wp-service.yaml**** and execute using the following command.

kubectl apply -f wp-service.yaml

Just look at the running services, then it will show an IP address for the load balancer service. Access the application using the IP address.

kubectl get svc

Enter the IP address given by the Load Balancer service in the browser and you will get installation page. Click Continue to proceed with WordPress Installation

Then set up the WordPress username and password to install the application.

**4. Remove All Pods, PVCs, and Services**

you can delete the Kubernetes components using the below commands.

kubectl delete svc --all

kubectl delete pod --all

kubectl delete pvc --all

kubectl delete pv --all