WordPress Setup with Docker and Kubernetes

# Introduction

This document provides a step-by-step guide on setting up a WordPress site using Docker and Kubernetes. It also explains how Kubernetes handles multiple instances of WordPress and ensures data consistency through a shared database.

# Step 1: Set Up WordPress with Docker

## Dockerfile for WordPress:

Docker has an official WordPress image, so you don’t need to create a Dockerfile from scratch. You can directly use this image. The WordPress image needs a database, typically MySQL or MariaDB.

## Docker Compose:

Use Docker Compose to define and run a multi-container application, including WordPress and MySQL.

### Sample docker-compose.yml file:

version: '3.3'  
  
services:  
 wordpress:  
 image: wordpress:latest  
 ports:  
 - "8080:80"  
 environment:  
 WORDPRESS\_DB\_HOST: db  
 WORDPRESS\_DB\_USER: exampleuser  
 WORDPRESS\_DB\_PASSWORD: examplepass  
 WORDPRESS\_DB\_NAME: exampledb  
 volumes:  
 - wordpress\_data:/var/www/html  
  
 db:  
 image: mysql:5.7  
 environment:  
 MYSQL\_DATABASE: exampledb  
 MYSQL\_USER: exampleuser  
 MYSQL\_PASSWORD: examplepass  
 MYSQL\_ROOT\_PASSWORD: rootpass  
 volumes:  
 - db\_data:/var/lib/mysql  
  
volumes:  
 wordpress\_data: {}  
 db\_data: {}

This configuration creates a WordPress container and a MySQL container. They both share data using Docker volumes, ensuring that data persists even if the containers are restarted.

# Step 2: Deploy WordPress with Kubernetes

Now, let's deploy the WordPress setup using Kubernetes.

## Kubernetes Deployment Files:

### MySQL Deployment and Service:

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: mysql  
spec:  
 replicas: 1  
 selector:  
 matchLabels:  
 app: mysql  
 template:  
 metadata:  
 labels:  
 app: mysql  
 spec:  
 containers:  
 - name: mysql  
 image: mysql:5.7  
 env:  
 - name: MYSQL\_ROOT\_PASSWORD  
 value: rootpass  
 - name: MYSQL\_DATABASE  
 value: exampledb  
 - name: MYSQL\_USER  
 value: exampleuser  
 - name: MYSQL\_PASSWORD  
 value: examplepass  
 ports:  
 - containerPort: 3306  
---  
apiVersion: v1  
kind: Service  
metadata:  
 name: mysql  
spec:  
 ports:  
 - port: 3306  
 selector:  
 app: mysql

### WordPress Deployment and Service:

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:latest  
 env:  
 - name: WORDPRESS\_DB\_HOST  
 value: mysql  
 - name: WORDPRESS\_DB\_USER  
 value: exampleuser  
 - name: WORDPRESS\_DB\_PASSWORD  
 value: examplepass  
 - name: WORDPRESS\_DB\_NAME  
 value: exampledb  
 ports:  
 - containerPort: 80  
---  
apiVersion: v1  
kind: Service  
metadata:  
 name: wordpress  
spec:  
 ports:  
 - port: 80  
 selector:  
 app: wordpress  
 type: LoadBalancer

These files define how Kubernetes should deploy the MySQL and WordPress containers and expose them to the network.

# Handling Multiple Instances and Database Synchronization

When Kubernetes runs multiple instances of WordPress, all instances connect to the same shared database. This ensures consistency in data, as any changes made by one instance are immediately reflected in the database and accessible by all other instances.

## Database Replication and Persistent Storage:

To ensure high availability, you can set up database replication, where one database instance is the primary and others are replicas. Kubernetes uses persistent volumes to store the database data, ensuring that it remains intact even if the database pod is restarted.

## Example Scenario:

Consider a scenario where you have 3 WordPress instances running, all connected to a single MySQL database. If a user creates an account, this data is stored in the MySQL database and is immediately accessible by all WordPress instances.

If one WordPress instance fails, Kubernetes will restart it or create a new one, and it will connect to the same database, ensuring no data loss. If the MySQL database fails, a replica can take over, allowing the WordPress instances to continue functioning.