Manual de Instalación y Despliegue de WordPress con MySQL en Kubernetes distribuido en 3 máquinas

Juan Edwin Calizaya Llanos

July 30, 2025

Introducción

Esta guía explica cada paso para desplegar WordPress con MySQL en un clúster de Kubernetes sin usar scripts: se crean y editan archivos de configuración manualmente, garantizando comprensión y control.

1. Preparación del Sistema

1.1. Deshabilitar swap y cargar módulos

1. Desactive swap en caliente y evite su reactivación en reinicios editando '/etc/fstab':
   - Ejecute:

```
swapoff -a
```

   - Abra '/etc/fstab' y comente la línea que contiene 'swap' colocando un # al inicio. Guarde y cierre.
   2. Cargue los módulos de red necesarios:
   - Ejecute:

```
modprobe overlay
modprobe br_netfilter
```

   - Cree '/etc/modules-load.d/k8s.conf' con el contenido:

```
overlay
br_netfilter
```

   3. Configure parámetros sysctl abriendo o creando '/etc/sysctl.d/k8s.conf' con:

```
net.bridge.bridge-nf-call-iptables = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward = 1
```

   4. Aplique los cambios:

```
sysctl --system
```

   5. Configure la IP en `/etc/netplan/50-cloud-init.yaml`

```
network:
  version: 2
  ethernets:
    enp0s3:
      dhcp4: no
```

```
    addresses:
      - IPADDR/24
    nameservers:
      addresses: [8.8.8.8, 8.8.4.4]
    routes:
      - to: 0.0.0.0/0
        via: 10.10.10.1
        metric: 100
```

```
sudo netplan apply
```

2. Instalación de containerd

1. Instale containerd:

```
apt update
apt install -y containerd
```

2. Genere la configuración predeterminada:
- Cree el directorio: `mkdir -p /etc/containerd`
- Genere el archivo:

```
containerd config default > /etc/containerd/config.toml
```

3. Habilite systemd como controlador de cgroup editando la línea ‘SystemdCgroup‘ dentro de ‘/etc/containerd/config.toml‘ de ‘false‘ a ‘true‘. Guarde.
4. Reinicie y habilite el servicio:

```
systemctl restart containerd
systemctl enable containerd
```

3. Instalación de Kubernetes (kubelet, kubeadm, kubectl)

1. Añada la clave GPG y repositorio:
- Descargue y registre la clave:

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key \
  | gpg --dearmor -o /usr/share/keyrings/kubernetes-archive-keyring.gpg
```

- Cree ‘/etc/apt/sources.list.d/kubernetes.list‘ con:

```
deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] \
    https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /
```

2. Instale los paquetes:

```
apt update
apt install -y kubelet kubeadm kubectl
```

3. Habilite kubelet:

```
systemctl enable kubelet
```

4. Inicializar clúster en nodo master

1. Inicialice con CIDR de pod 10.10.0.0/16 y omita errores de CPU/mem:

```
kubeadm init --pod-network-cidr=10.10.0.0/16 --ignore-preflight-errors=NumCPU,Mem
```

2. Configure kubectl:

```
mkdir -p $HOME/.kube
cp -f /etc/kubernetes/admin.conf $HOME/.kube/config
chown $(id -u):$(id -g) $HOME/.kube/config
```

3. Permita scheduling en el master:

```
NODE_NAME=$(kubectl get nodes -o jsonpath='{.items[0].metadata.name}')
kubectl taint nodes "$NODE_NAME" node-role.kubernetes.io/control-plane- || true
```

5. Red Calico

1. Descargue el manifiesto y cambie el CIDR interno a 10.10.0.0/16. 2. Aplíquelo:

```
kubectl apply -f calico.yaml
```

6. Almacenamiento dinámico (local-path)

Aplique el manifiesto:

```
kubectl apply -f https://raw.githubusercontent.com/rancher/local-path-provisioner/master/deploy/local-path-
 storage.yaml
```

7. Despliegue de WordPress y MySQL

7.1. Crear archivo `mysqlPass.yaml`

Incluya:

```
apiVersion: v1
kind: Secret
metadata:
  name: mysql-pass
type: Opaque
stringData:
  password: "1234"
```

Aplique con:

```
kubectl apply -f mysqlPass.yaml
```

7.2. Crear archivo `storage.yaml`

Incluya:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
spec:
  storageClassName: local-path
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wordpress-pv-claim
spec:
  storageClassName: local-path
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

Aplique:

```
kubectl apply -f storage.yaml
```

7.3. Crear archivo `mysqlFull.yaml`

Incluya:

```
apiVersion: v1
kind: Service
metadata:
  name: mysql-service
spec:
  ports:
    - port: 3306
  selector:
    app: mysql
  clusterIP: None
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment
spec:
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - name: mysql
        image: mysql:5.7
        env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
```

```
            name: mysql-pass
            key: password
      - name: MYSQL_DATABASE
        value: "wordpress"
      - name: MYSQL_USER
        value: "wordpress"
      - name: MYSQL_PASSWORD
        valueFrom:
          secretKeyRef:
            name: mysql-pass
            key: password
      ports:
      - containerPort: 3306
        name: mysql
      volumeMounts:
      - name: mysql-persistent-storage
        mountPath: /var/lib/mysql
    volumes:
    - name: mysql-persistent-storage
      persistentVolumeClaim:
        claimName: mysql-pv-claim
```

Aplique:

```
kubectl apply -f mysqlFull.yaml
```

7.4. Crear archivo `wordpress.yaml`

Incluya:

```
apiVersion: v1
kind: Service
metadata:
  name: wordpress-service
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30090
  selector:
    app: wordpress
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: wordpress
  template:
    metadata:
      labels:
        app: wordpress
    spec:
      containers:
      - name: wordpress
        image: wordpress:latest
        env:
        - name: WORDPRESS_DB_HOST
          value: mysql-service
        - name: WORDPRESS_DB_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-pass
```

```
            key: password
      - name: WORDPRESS_DB_USER
        value: "wordpress"
      - name: WORDPRESS_DB_NAME
        value: "wordpress"
      ports:
      - containerPort: 80
        name: wordpress
      volumeMounts:
      - name: wordpress-persistent-storage
        mountPath: /var/www/html
    volumes:
    - name: wordpress-persistent-volume
      persistentVolumeClaim:
        claimName: wordpress-pv-claim
```

Aplique:

```
kubectl apply -f wordpress.yaml
```

8. Nodos Worker

En cada máquina que actuará como nodo trabajador (worker), se realizan los mismos pasos de preparación de red y sistema que en el master, y luego se une al clúster mediante el comando generado en el master.

8.1 Verifique su IP

```
CUR_IP=$(hostname -I | awk '{print $1}')
```

Asegúrese de que corresponda a uno de los nodos definidos (por ejemplo `10.10.10.11` o `10.10.10.12`).

8.2 Configure la red estática si es necesario

```
% Edite 50-cloud-init.yaml, reemplace IPADDR por \texttt{$CUR_IP}
netplan apply
```

8.3 Deshabilite swap, cargue módulos y aplique sysctl (igual que en el master)

```
swapoff -a && sed -i '/ swap / s/^/#/' /etc/fstab
modprobe overlay br_netfilter
echo -e "overlay\nbr_netfilter" > /etc/modules-load.d/k8s.conf
cp files/etc-sysctl.d-k8s.conf/k8s.conf /etc/sysctl.d/k8s.conf
sysctl --system
```

8.4 Instale y configure containerd y herramientas de Kubernetes (igual que en el master)

```
apt update && apt install -y containerd curl ca-certificates gnupg lsb-release
mkdir -p /etc/containerd && containerd config default > /etc/containerd/config.toml
sed -i 's/SystemdCgroup = false/SystemdCgroup = true/' /etc/containerd/config.toml
systemctl enable --now containerd

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key \
  | gpg --dearmor -o /usr/share/keyrings/kubernetes-archive-keyring.gpg
```

```
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://pkgs.k8s.io/core:/stable:/
 v1.31/deb/ /" \
  > /etc/apt/sources.list.d/kubernetes.list
apt update && apt install -y kubelet kubeadm kubectl
systemctl enable --now kubelet
```

8.5 Limpie configuraciones previas y purgue containerd

```
systemctl stop kubelet containerd
kubeadm reset -f || true
rm -rf /etc/cni/net.d /etc/kubernetes /var/lib/kubelet /var/lib/etcd /opt/cni/bin/*
rm -rf /var/lib/containerd
```

8.6 Reinstale containerd limpio y reinicie servicios

```
apt install -y containerd
mkdir -p /etc/containerd && containerd config default > /etc/containerd/config.toml
sed -i 's/SystemdCgroup = false/SystemdCgroup = true/' /etc/containerd/config.toml
systemctl enable --now containerd
systemctl enable --now kubelet
```

8.7 Obtenga el comando de unión generado en el master y únase al clúster. EJEMPLO

```
kubeadm join 10.10.10.10:6443 --token 0fchvu.tyfu5twwhc61h1tn --discovery-token-ca-cert-hash sha256:07
 aa1a55f0fb8600bfc6ac22380aba35892a8cf176445d3fa8246556b61836ae
```

9. Verificación

Verifique:

```
curl -s --max-time 2 http://$(hostname -I | awk '{print $1}'):30090 | grep -q 'WordPress'
```

Manual elaborado por Juan Edwin Calizaya Llanos.