THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

Deploying MySQL and WordPress with Persistent Volumes Using Minikube on RHEL

Description:

This project demonstrates the deployment of a WordPress application backed by a MySQL database on a local Kubernetes cluster using Minikube in a Red Hat Enterprise Linux (RHEL) environment. The key focus is to enable data persistence using Persistent Volumes (PV) and Persistent Volume Claims (PVC), ensuring that application data remains intact even if pods are restarted or rescheduled.

Minikube was selected to simulate a local Kubernetes environment, allowing for easy testing and development. The setup involves deploying MySQL and WordPress as separate pods, each configured with their respective persistent storage. Kubernetes services are used to manage internal communication between the two applications and to expose the WordPress application externally.

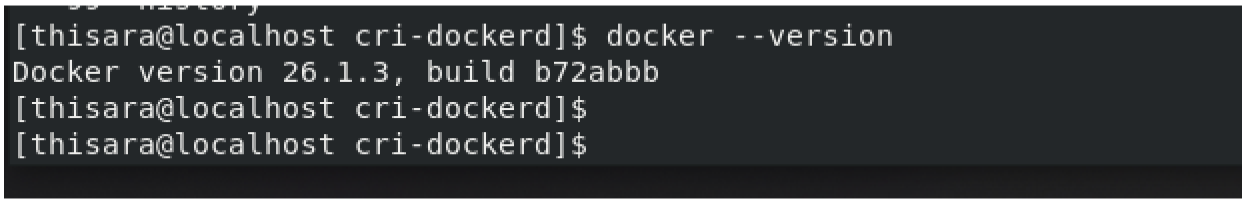
The MySQL pod is configured with required environment variables such as root password, database name, and user credentials. A Persistent Volume and PVC are created to store the MySQL data. Similarly, the WordPress pod is configured to connect to the MySQL service and uses its own persistent storage to retain WordPress content such as media uploads and themes.

The project includes YAML configuration files for deployments, services, and storage definitions. After deployment, functionality was verified by accessing WordPress via browser, completing the installation, and ensuring the persistence of data even after restarting Minikube.

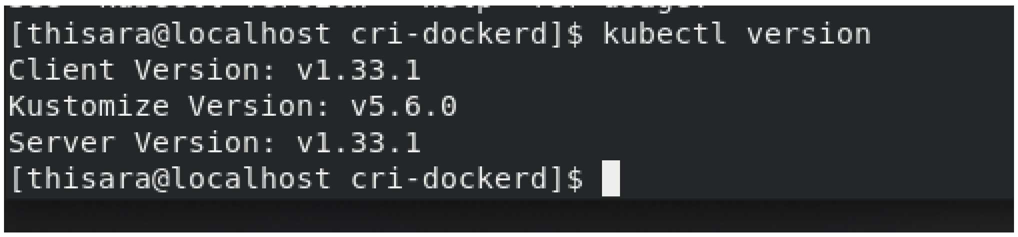
This project showcases practical use of Kubernetes features such as deployments, services, and persistent storage in a real-world web application scenario. It also emphasizes the benefits of container orchestration for managing scalable, resilient applications with reliable storage.

Prerequisites

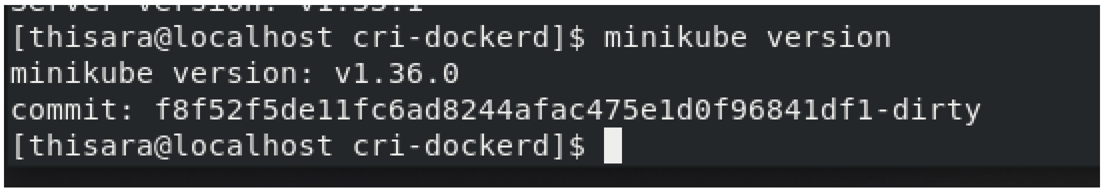
- Docker and Kubernetes already installed in my system.



```
[thisara@localhost cri-dockerd]$ docker --version
Docker version 26.1.3, build b72abbb
[thisara@localhost cri-dockerd]$
[thisara@localhost cri-dockerd]$
```



```
[thisara@localhost cri-dockerd]$ kubectl version
Client Version: v1.33.1
Kustomize Version: v5.6.0
Server Version: v1.33.1
[thisara@localhost cri-dockerd]$
```

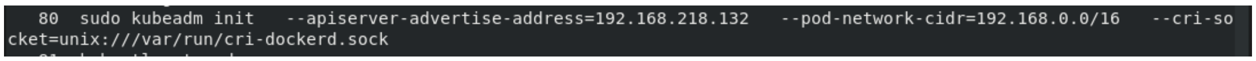


```
[thisara@localhost cri-dockerd]$ minikube version
minikube version: v1.36.0
commit: f8f52f5de11fc6ad8244afac475e1d0f96841df1-dirty
[thisara@localhost cri-dockerd]$
```

1. Initialize Kubernetes cluster on my master machine node

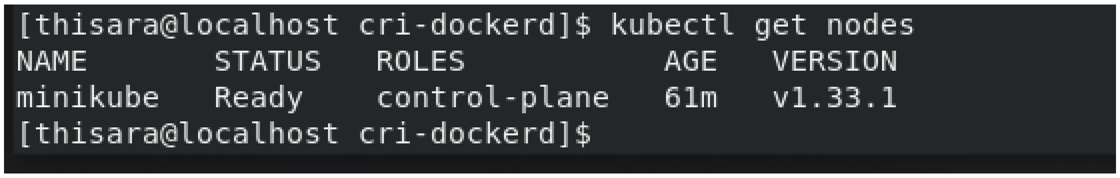
- Delete existing Kubernetes cluster that in my system.

  Command – kubeadm reset



```
   80  sudo kubeadm init   --apiserver-advertise-address=192.168.218.132   --pod-network-cidr=192.168.0.0/16   --cri-so
cket=unix:///var/run/cri-dockerd.sock
```

2. check the active nodes

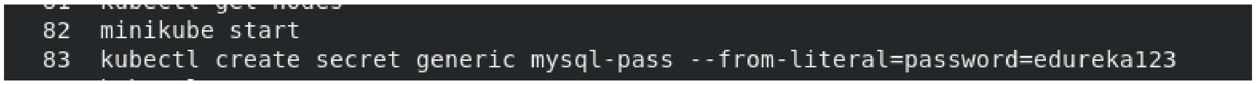


```
[thisara@localhost cri-dockerd]$ kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   61m   v1.33.1
[thisara@localhost cri-dockerd]$
```

3. start minikube for deployment

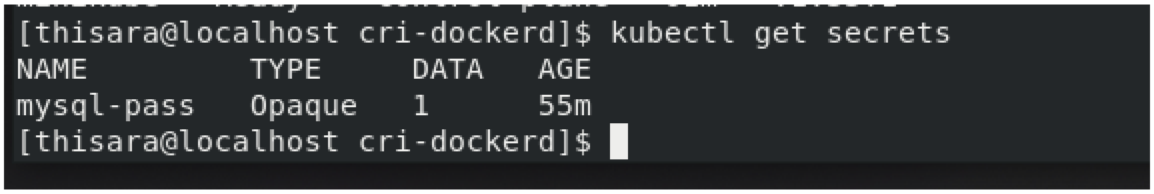
Command - sudo minikube start

4. Creating Secret for the MySQL password



```
  82  minikube start
  83  kubectl create secret generic mysql-pass --from-literal=password=edureka123
```

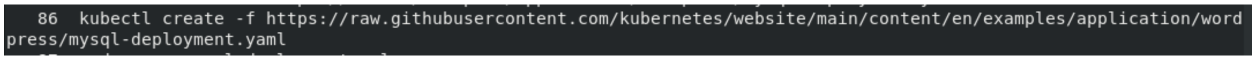
5.verify the secret



```
[thisara@localhost cri-dockerd]$ kubectl get secrets
NAME         TYPE     DATA   AGE
mysql-pass   Opaque   1      55m
[thisara@localhost cri-dockerd]$
```

6.Deploy a MySQL Container steps

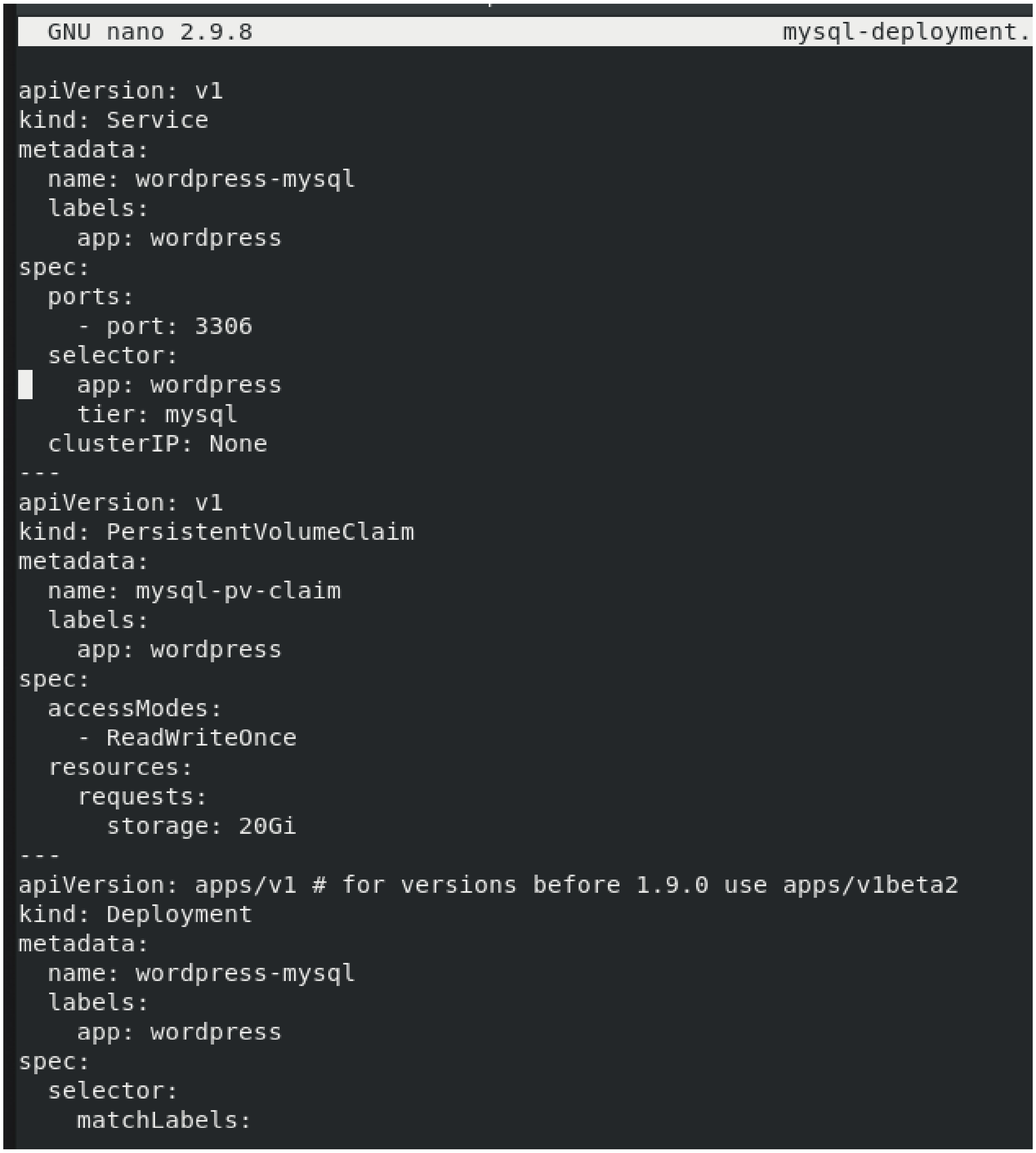
Command - kubectl create -f https://raw.githubusercontent.com/kubernetes/website/main/content/en/examples/application/wordpress/mysql-deployment.yaml



```
  86  kubectl create -f https://raw.githubusercontent.com/kubernetes/website/main/content/en/examples/application/word
press/mysql-deployment.yaml
```

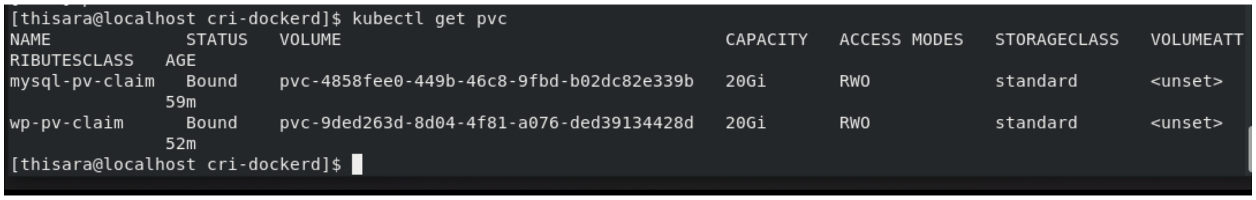
- **Open the mysql-deployment.yaml file**

Command - sudo nano mysql-deployment.yml



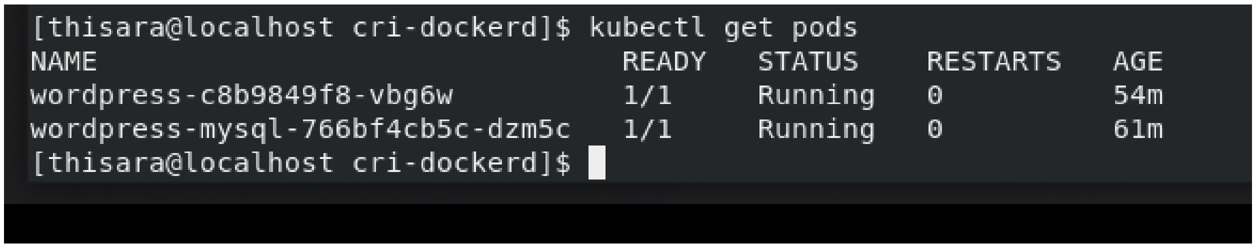
```
  GNU nano 2.9.8                                        mysql-deployment.

apiVersion: v1
kind: Service
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  ports:
    - port: 3306
  selector:
    app: wordpress
    tier: mysql
  clusterIP: None
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
---
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
```

- **Get verify the persistence volume got dynamically provisioned or not**



```
[thisara@localhost cri-dockerd]$ kubectl get pvc
NAME            STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATT
RIBUTESCLASS   AGE
mysql-pv-claim   Bound    pvc-4858fee0-449b-46c8-9fbd-b02dc82e339b   20Gi       RWO            standard       <unset>
               59m
wp-pv-claim      Bound    pvc-9ded263d-8d04-4f81-a076-ded39134428d   20Gi       RWO            standard       <unset>
               52m
[thisara@localhost cri-dockerd]$
```

- **Verify the pod is running or not**



```
[thisara@localhost cri-dockerd]$ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
wordpress-c8b9849f8-vbg6w          1/1     Running   0          54m
wordpress-mysql-766bf4cb5c-dzm5c   1/1     Running   0          61m
[thisara@localhost cri-dockerd]$
```

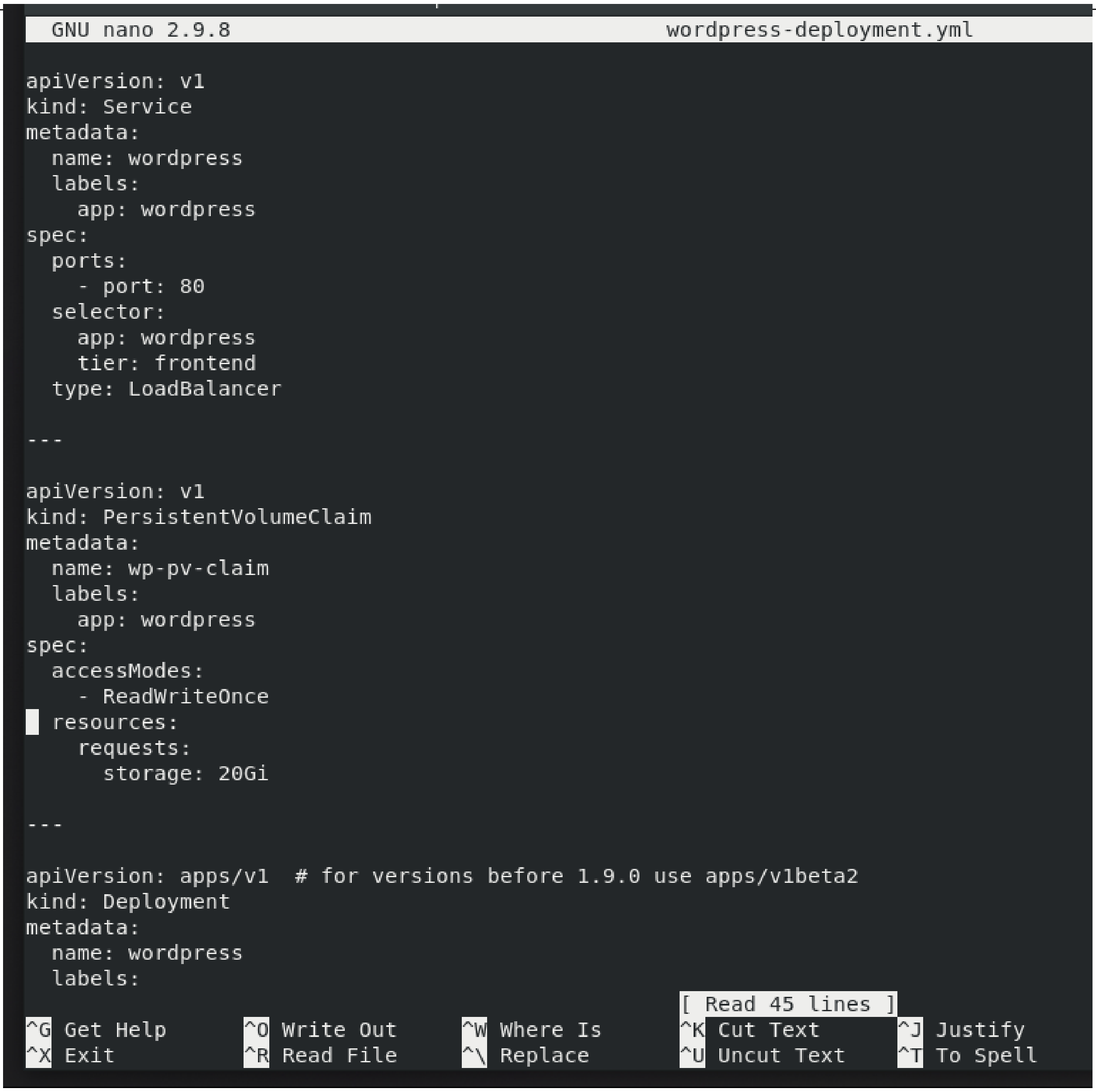
7. deploy WordPress application steps

- **Create WordPress service and deployment**

  Command = kubectl create -f https://raw.githubusercontent.com/kubernetes/website/main/content/en/examples/application/wordpress/wordpress-deployment.yaml

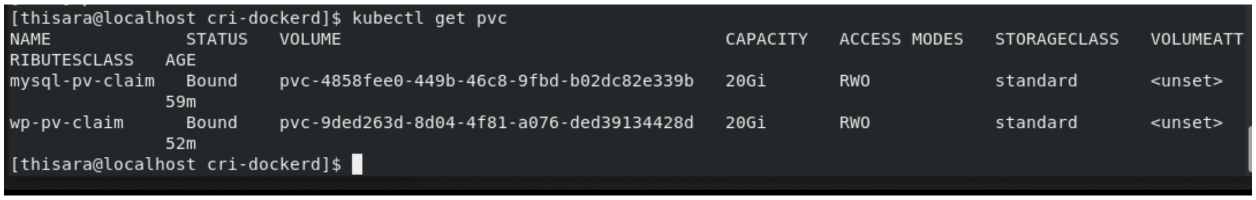
- **Open the wordpress-deployment.yml file**

  Command = sudo nano wordpress-deployment.yml



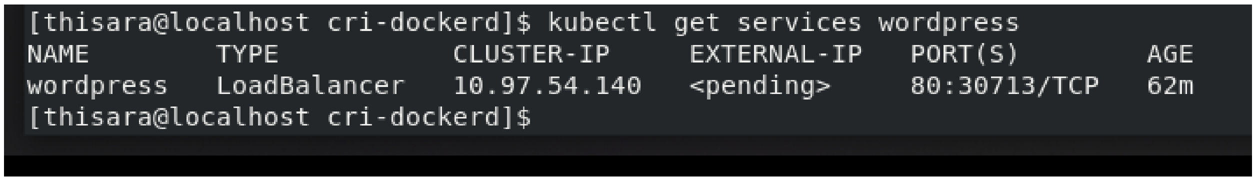
```
  GNU nano 2.9.8                                    wordpress-deployment.yml

apiVersion: v1
kind: Service
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  ports:
    - port: 80
  selector:
    app: wordpress
    tier: frontend
  type: LoadBalancer

---

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-pv-claim
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi

---

apiVersion: apps/v1  # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: wordpress
  labels:
                                             [ Read 45 lines ]
^G Get Help      ^O Write Out     ^W Where Is      ^K Cut Text      ^J Justify
^X Exit          ^R Read File     ^\ Replace       ^U Uncut Text    ^T To Spell
```

- **Get verify the persistence volume got dynamically provisioned or not**



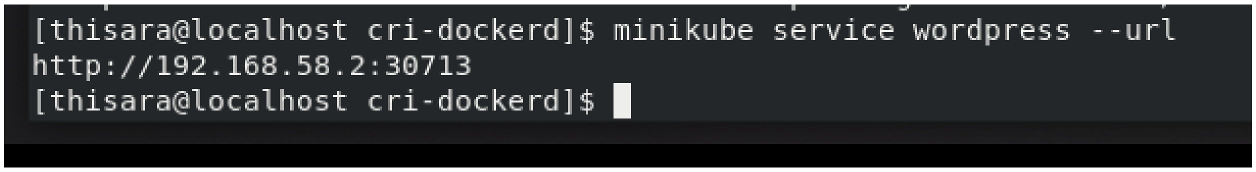
```
[thisara@localhost cri-dockerd]$ kubectl get pvc
NAME           STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATT
RIBUTESCLASS   AGE
mysql-pv-claim   Bound    pvc-4858fee0-449b-46c8-9fbd-b02dc82e339b   20Gi       RWO            standard       <unset>
               59m
wp-pv-claim      Bound    pvc-9ded263d-8d04-4f81-a076-ded39134428d   20Gi       RWO            standard       <unset>
               52m
[thisara@localhost cri-dockerd]$ 
```

- **Verify the services is running or not**

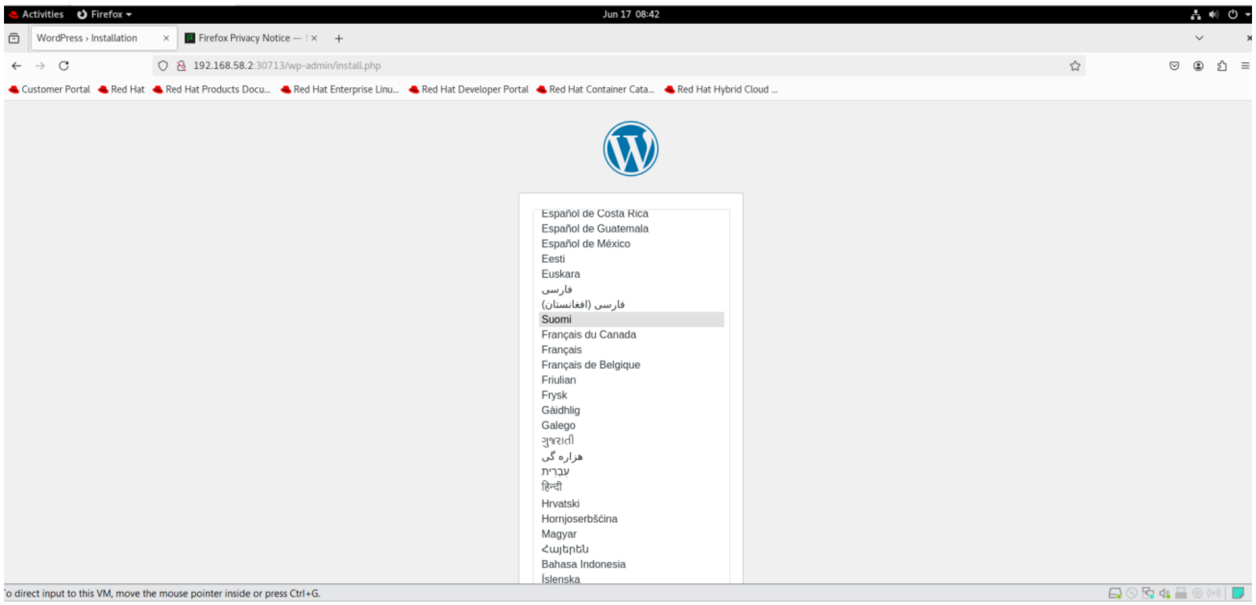


```
[thisara@localhost cri-dockerd]$ kubectl get services wordpress
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)        AGE
wordpress    LoadBalancer   10.97.54.140   <pending>      80:30713/TCP   62m
[thisara@localhost cri-dockerd]$
```

- **Get the ip address for WordPress service**

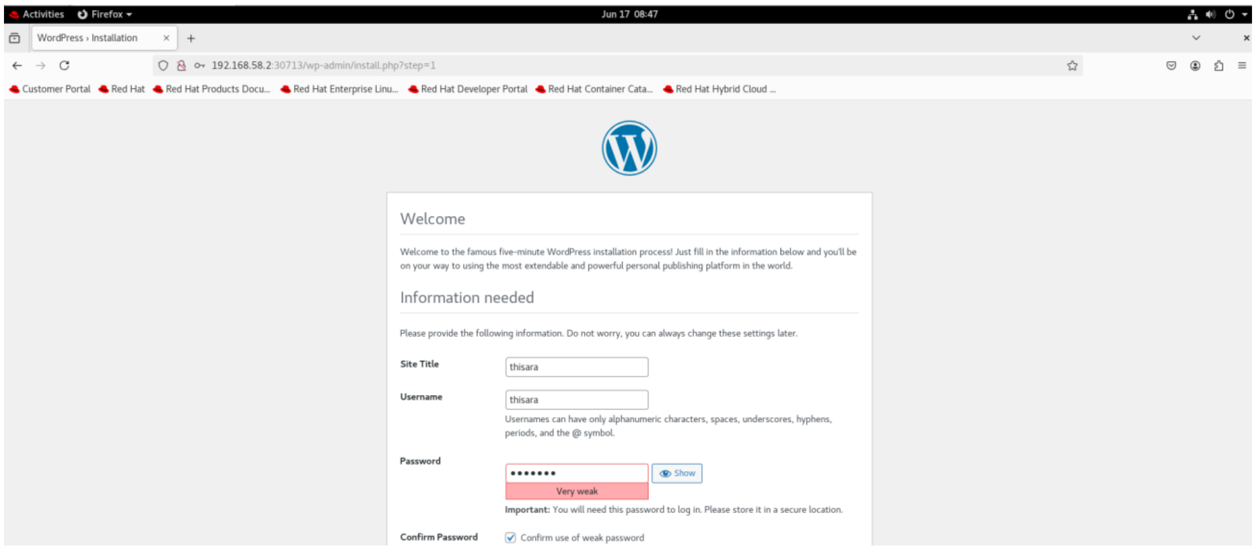


```
[thisara@localhost cri-dockerd]$ minikube service wordpress --url
http://192.168.58.2:30713
[thisara@localhost cri-dockerd]$
```



Español de Costa Rica
Español de Guatemala
Español de México
Eesti
Euskara
فارسی
(فارسی (افغانستان
Suomi
Français du Canada
Français
Français de Belgique
Friulian
Frysk
Gàidhlig
Galego
ગુજરાતી
هزاره گی
עִבְרִית
हिन्दी
Hrvatski
Hornjoserbšćina
Magyar
Հայերեն
Bahasa Indonesia
Íslenska

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.



Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

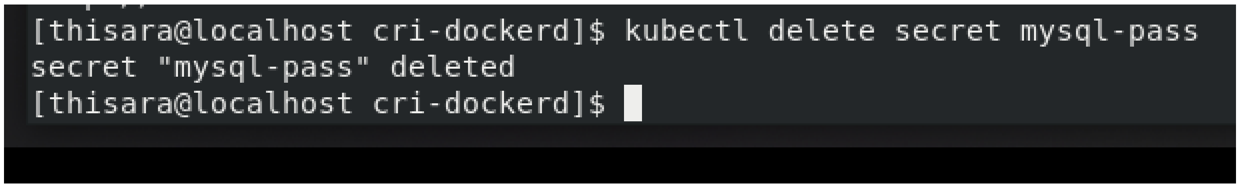
Information needed

Please provide the following information. Do not worry, you can always change these settings later.

Site Title	thisara
Username	thisara Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.
Password	••••••• Show Very weak **Important:** You will need this password to log in. Please store it in a secure location.
Confirm Password	✓ Confirm use of weak password

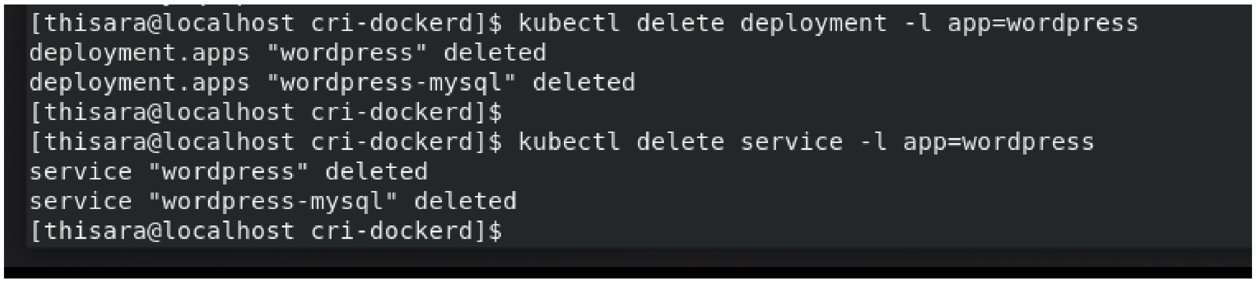
Cleanup Action (if needed)

7. after all done delete the secrets



```
[thisara@localhost cri-dockerd]$ kubectl delete secret mysql-pass
secret "mysql-pass" deleted
[thisara@localhost cri-dockerd]$ 
```

8.Delete all deployments and services



```
[thisara@localhost cri-dockerd]$ kubectl delete deployment -l app=wordpress
deployment.apps "wordpress" deleted
deployment.apps "wordpress-mysql" deleted
[thisara@localhost cri-dockerd]$
[thisara@localhost cri-dockerd]$ kubectl delete service -l app=wordpress
service "wordpress" deleted
service "wordpress-mysql" deleted
[thisara@localhost cri-dockerd]$
```

Summary:

Through this project, I successfully deployed a WordPress application with a MySQL database using Kubernetes on a Minikube cluster running on RHEL. I gained hands-on experience in writing and applying Kubernetes YAML manifests to define deployments, services, and persistent storage. I also learned how to configure environment variables for application setup and how to establish communication between multiple pods using Kubernetes services.

One of the most important lessons was understanding how Persistent Volumes (PVs) and Persistent Volume Claims (PVCs) work to retain application data even when pods are restarted or replaced. This reinforced the importance of data persistence in containerized environments. Additionally, I became more comfortable with using kubectl commands to manage and troubleshoot Kubernetes resources in a local development environment.

Overall, this project helped me understand how real-world web applications can be containerized, deployed, and managed using Kubernetes. It improved my knowledge of DevOps practices, especially in the areas of container orchestration, application configuration, and persistent storage management.