simplilearn

Get Certified. Get Ahead.

PG DO - CONTAINER ORCHESTRATION USING KUBERNETES

DEPLOY THE APPLICATION USING THE KUBERNETES DASHBOARD

Git Hub Repository: https://github.com/sravan1990/simplilearn_Deploy_Application_Using-the-Kubernetes_Dashboard.git

PRESENTED BY: VENKATA SRAVAN KUMAR CHIVUKULA

.

.

DEPLOY THE APPLICATION USING THE KUBERNETES DASHBOARD

DESCRIPTION

OBJECTIVES

To deploy a multi-tier PHP and MySQL application using Kubernetes with specific configurations for user roles, storage, service verification, namespace restrictions, quota limits, and data management.

PROBLEM STATEMENT AND MOTIVATION

Real-Time Scenario:

Karen is a DevOps engineer at a tech startup. Her team has developed a new application using PHP and MySQL. Now, it is her task to deploy that application.

The company plans to utilize Kubernetes for its robust container orchestration capabilities.

Karen must create a Kubernetes dashboard with specific configurations, user roles, storage, service verification, and data management.

INDUSTRY RELEVANCE :

The following tools utilized in this project are widely applied in the industry:

1. **kubeadm**: A utility that offers kubeadm init and kubeadm join as efficient ways to bootstrap Kubernetes clusters. It focuses on bootstrapping rather than machine provisioning.

2. **kubectl**: A command-line interface for Kubernetes that allows execution of commands against Kubernetes clusters. It can be used for deploying applications, managing cluster resources, and viewing logs.

3. **kubelet**: An essential node agent present on every node in a Kubernetes cluster. It ensures that the containers described in the provided PodSpecs are running and healthy.

4. **Docker**: Docker is a tool designed to facilitate developers in building, sharing, and running applications in containers. It takes care of the setup, allowing developers to concentrate on the code.

SOLUTION EXECUTION STEPS

1. CREATE THE KUBERNETES CLUSTER

- Kubernetes is already installed on our lab setup which consist of one control node and two worker nodes.

```
labsuser@ip-172-31-35-60:~$ kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"28", GitVersion:"v1.28.2", GitCommit:"89a4ea3e1e4ddd7f7572286090359983e0387b2f", GitTre
eState:"clean", BuildDate:"2023-09-13T09:34:32Z", GoVersion:"go1.20.8", Compiler:"gc", Platform:"linux/amd64"}
labsuser@ip-172-31-35-60:~$ kubelet version
E0114 02:16:53.757913   11133 run.go:74] "command failed" err="unknown command version"
labsuser@ip-172-31-35-60:~$ kubectl version
Client Version: v1.28.2
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Server Version: v1.28.2
labsuser@ip-172-31-35-60:~$
```

- We need to enable the cluster using kubeadm using the following command

sudo kubeadm init --pod-network-cidr 10.96.0.0/12 --kubernetes-version 1.28.2

```
labsuser@ip-172-31-35-60:~$ sudo kubeadm init --pod-network-cidr 10.96.0.0/12 --kubernetes-version 1.28.2
[init] Using Kubernetes version: v1.28.2
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
W0114 02:10:08.780360    6680 checks.go:835] detected that the sandbox image "k8s.gcr.io/pause:3.6" of the
tent with that used by kubeadm. It is recommended that using "registry.k8s.io/pause:3.9" as the CRI sandbo
[certs] Using certificateDir folder "/etc/kubernetes/pki"
```

- Once executed successfully a command with token will appear at the end.

```
Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.33.187:6443 --token i2m2vf.e58a0tmbf2vputt3 \
        --discovery-token-ca-cert-hash sha256:fdcf45da532e7a858523d15ba83a829f111530f5d775f41cbd24503a597abfa3
```

- We will need to copy this over to the worker nodes and execute it as root.

Worker 1

```
labsuser@ip-172-31-41-51:~$ sudo kubeadm join 172.31.33.187:6443 --token i2m2vf.e58a0tmbf2vputt3 \
        --discovery-token-ca-cert-hash sha256:fdcf45da532e7a858523d15ba83a829f111530f5d775f41cbd24503a597abfa3
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Worker 2

```
labsuser@ip-172-31-46-81:~$ sudo kubeadm join 172.31.33.187:6443 --token i2m2vf.e58a0tmbf2vputt3 \
        --discovery-token-ca-cert-hash sha256:fdcf45da532e7a858523d15ba83a829f111530f5d775f41cbd24503a597abfa3
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

- Check all nodes

```
labsuser@ip-172-31-33-187:~$ kubectl get nodes
NAME          STATUS   ROLES           AGE   VERSION
k8s-control   Ready    control-plane   16m   v1.28.2
k8s-worker1   Ready    <none>          32s   v1.28.2
k8s-worker2   Ready    <none>          14s   v1.28.2
labsuser@ip-172-31-33-187:~$
```

* Our Kubernetes cluster is successfully created.

```
labsuser@ip-172-31-33-187:~$ kubectl cluster-info
Kubernetes control plane is running at https://172.31.33.187:6443
CoreDNS is running at https://172.31.33.187:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
labsuser@ip-172-31-33-187:~$ kubectl get ns
NAME              STATUS   AGE
default           Active   28m
kube-node-lease   Active   28m
kube-public       Active   28m
kube-system       Active   28m
labsuser@ip-172-31-33-187:~$
```

2. INSTALL THE KUBERNETES DASHBOARD (PS: CHANGE IN IP IS DUE TO LAB RESET)

- To install dashboard, we will need the following prerequisites

Install Calico (add-on)

kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v3.29.1/manifests/tigera-operator.yaml kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v3.29.1/manifests/custom-resources.yaml

```
labsuser@ip-172-31-0-15:~$ kubectl get pods -A
NAMESPACE          NAME                                       READY   STATUS    RESTARTS   AGE
calico-apiserver   calico-apiserver-bb8b5d6b6-dts7t           1/1     Running   0          127m
calico-apiserver   calico-apiserver-bb8b5d6b6-pcrj5           1/1     Running   0          127m
calico-system      calico-kube-controllers-6dbff566cd-8cf8q   1/1     Running   0          127m
calico-system      calico-node-29fgn                          1/1     Running   0          119m
calico-system      calico-node-5zmfg                          1/1     Running   0          119m
calico-system      calico-node-sg9bb                          1/1     Running   0          127m
calico-system      calico-typha-85d7889d6f-lgkdz              1/1     Running   0          119m
calico-system      calico-typha-85d7889d6f-w8z98              1/1     Running   0          127m
calico-system      csi-node-driver-cfv92                      2/2     Running   0          119m
calico-system      csi-node-driver-gntsx                      2/2     Running   0          119m
calico-system      csi-node-driver-kttt2                      2/2     Running   0          127m
kube-system        coredns-5dd5756b68-m8pzg                   1/1     Running   0          130m
kube-system        coredns-5dd5756b68-tkl25                   1/1     Running   0          130m
kube-system        etcd-ip-172-31-0-15                        1/1     Running   1          131m
kube-system        kube-apiserver-ip-172-31-0-15              1/1     Running   1          131m
kube-system        kube-controller-manager-ip-172-31-0-15     1/1     Running   0          131m
kube-system        kube-proxy-7bfgh                           1/1     Running   0          130m
kube-system        kube-proxy-8mzfm                           1/1     Running   0          119m
kube-system        kube-proxy-bxd8t                           1/1     Running   0          119m
kube-system        kube-scheduler-ip-172-31-0-15              1/1     Running   1          131m
tigera-operator    tigera-operator-c7ccbd65-r9fwc             1/1     Running   0          128m
```

Install helm

labsuser@ip-172-31-0-15:~$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
labsuser@ip-172-31-0-15:~$ chmod 700 get_helm.sh
labsuser@ip-172-31-0-15:~$ ./get_helm.sh

```
labsuser@ip-172-31-0-15:~$ helm repo add kubernetes-dashboard https://kubernetes.github.io/dashboard/
Command 'helm' not found, but can be installed with:
sudo snap install helm
labsuser@ip-172-31-0-15:~$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
labsuser@ip-172-31-0-15:~$ chmod 700 get_helm.sh
labsuser@ip-172-31-0-15:~$ ./get_helm.sh
Downloading https://get.helm.sh/helm-v3.17.0-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
```

Install Kubernetes dashboard

kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.5.0/aio/deploy/recommended.yaml

```
labsuser@ip-172-31-0-15:~$ kubectl apply -f      https://raw.githubusercontent.com/kubernetes/dashboard/v2.5.0/aio/deploy/recommended.yaml
namespace/kubernetes-dashboard created
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created
```

- Now the dashboard has been installed but is only accessible via the cluster port which is not exposed.

```
labsuser@ip-172-31-0-15:~$ kubectl get svc -n kubernetes-dashboard -o wide
NAME                        TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE   SELECTOR
dashboard-metrics-scraper   ClusterIP   10.110.100.172   <none>        8000/TCP   29s   k8s-app=dashboard-metrics-scraper
kubernetes-dashboard        ClusterIP   10.98.28.179     <none>        443/TCP    29s   k8s-app=kubernetes-dashboard
```

- So, we will need to change the service type from **ClusterIP** to **NodePort** to be able to access the dashboard externally via the browser.
  kubectl edit svc -n kubernetes-dashboard kubernetes-dashboard

Before.

```
  selector:
    k8s-app: kubernetes-dashboard
  sessionAffinity: None
  type: ClusterIP
status:
  loadBalancer: {}
```

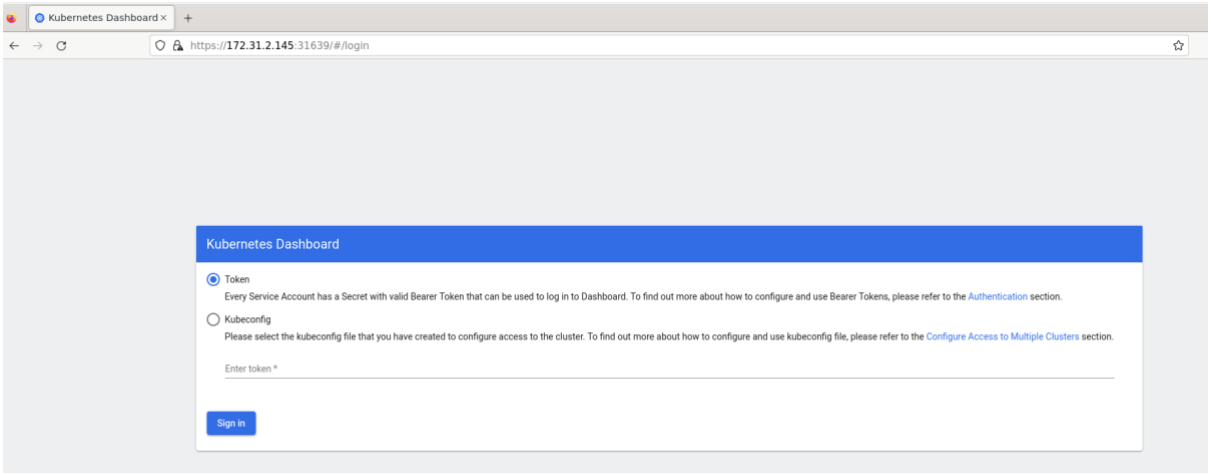
After

```
  selector:
    k8s-app: kubernetes-dashboard
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
```

- Now we see a new service with type **NodePort** running on port **31639.**

```
labsuser@ip-172-31-0-15:~$ kubectl get svc -n kubernetes-dashboard -o wide
NAME                        TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)         AGE   SELECTOR
dashboard-metrics-scraper   ClusterIP   10.110.100.172   <none>        8000/TCP        117s  k8s-app=dashboard-metrics-scraper
kubernetes-dashboard        NodePort    10.98.28.179     <none>        443:31639/TCP   117s  k8s-app=kubernetes-dashboard
```

- Now the dashboard can be accessed via the browser using the nodeport.



3. CONFIGURE SPECIFIC ROLES, STORAGE, SERVICE VERIFICATION, DATA MGMT

- Create a user (service account) called Simplilearn and assign admin role to dashboard.

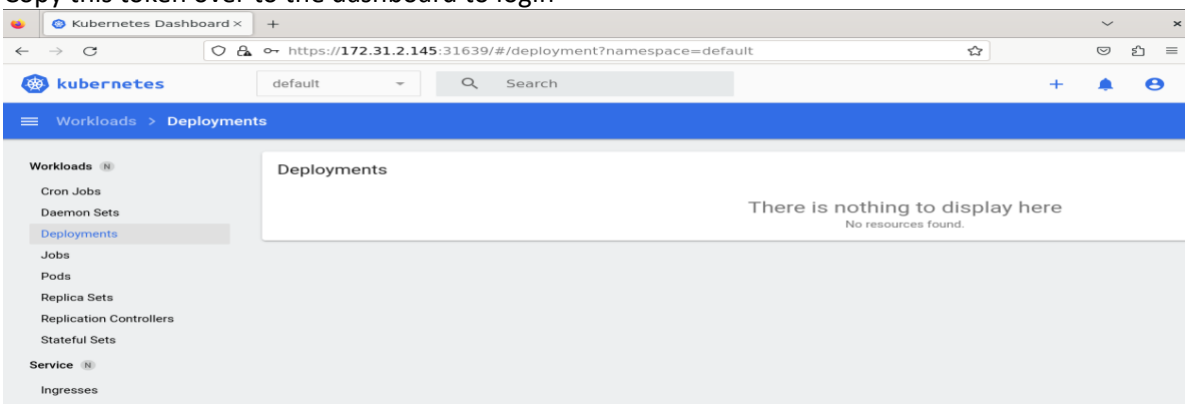
  $ kubectl create sa simplilearn -n kubernetes-dashboard
  $ kubectl get sa -n kubernetes-dashboard
  $ kubectl create clusterrolebinding simplilearn --clusterrole=admin --serviceaccount=kubernetes-dashboard:simplilearn

```
labsuser@ip-172-31-0-15:~$ kubectl create sa simplilearn -n kubernetes-dashboard
serviceaccount/simplilearn created
labsuser@ip-172-31-0-15:~$ kubectl get sa -n kubernetes-dashboard
NAME                                 SECRETS   AGE
default                              0         40m
kubernetes-dashboard-api             0         40m
kubernetes-dashboard-kong            0         40m
kubernetes-dashboard-metrics-scraper 0         40m
kubernetes-dashboard-web             0         40m
simplilearn                          0         63s
labsuser@ip-172-31-0-15:~$ kubectl create clusterrolebinding simplilearn --clusterrole=admin --serviceaccount=kubernetes-dashboard:simplilearn
clusterrolebinding.rbac.authorization.k8s.io/simplilearn created
labsuser@ip-172-31-0-15:~$
```

- Create a token for working on dashboard
  $ kubectl create token simplilearn -n kubernetes-dashboard

```
labsuser@ip-172-31-0-15:~$ kubectl create token simplilearn -n kubernetes-dashboard
eyJhbGciOiJSUzI1NiIsImtpZCI6IkdXc3JHWVpKM0xnWnBWMERwazM3Sy12WjZZdTlrRUJBUFFFbWhiOWFxME0ifQ.eyJhdWQiOlsiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjLmNsdXN0ZXIubG9
jYWwiXSwiZXhwIjoxNzM3NTE4NDA3LCJpYXQiOjE3Mzc1MTQ4MDcsImlzcyI6Imh0dHBzOi8va3ViZXJuZXRlcy5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2FsIiwia3ViZXJuZXRlcy5pbyI6eyJuYW1lc3BhY
2UiOiJrdWJlcm5ldGVzLWRhc2hib2FyZCIsInNlcnZpY2VhY2NvdW50Ijp7Im5hbWUiOiJzaW1wbGlsZWFybiIsInVpZCI6IjQ4ZjFjYWNkLTljODItNDM1MC1iOGU5LTc2ZmQyY2VmN2MyYSJ9fSwibmJmIjo
xNzM3NTE0ODA3LCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6a3ViZXJuZXRlcy1kYXNoYm9hcmQ6c2ltcGxpbGVhcm4ifQ.VYjbx5DXtDWsV8qbsDpDjH3uVT1tLc0_sSX8EUDTQcC_XsTRNKFwLl6SXs
V8rImw2Kw7BlctLJYCKyaFYbvcEPnZ9Lc0f4c6JVWj1LLJpyk8_dTIaY0VjJIuTn1vpszgfpN8JgLQXc2u2tT3gkIjDOt9fkveA06NauD3AZalk30HUM68M1pIv3IfXl6AIYjH0koawlBlNYs-G6RmZR29J61Z
jxyP900VT-Cy_TAChS88ea0_ghA7-ZqCD8npYF3ePJ9vnbqfTCSf9tyCjGgAwJj6_WRwQNPZgxK_qOhk7ldmgj9E6CO09OeZ5msGT8VmljWthz7CyHtcIvcD4fDCRA
```

- Copy this token over to the dashboard to login



4. DEPLOY THE APPLICATION

- Configure NFS-Server for MySQL and WordPress Deployment

```
sudo mkdir /nfswp
sudo mkdir /nfsms
```

Install NFS server on cluster plane

```
sudo apt install nfs-kernel-server
```

Grant permissions to the folders

```
sudo vi /etc/exports
```

```
/nfswp      *(rw,sync,no_root_squash)
/nfsms      *(rw,sync,no_root_squash)
```

Install NFS common on both workers

```
sudo apt install nfs-common
```

Make folders public

```
sudo exportfs -rv

sudo chown nobody:nogroup /nfswp/
sudo chown nobody:nogroup /nfsms/

sudo chmod 777 /nfswp/
sudo chmod 777 /nfsms/
```

Restart services

```
sudo systemctl restart nfs-kernel-server
```

- Create a namespace for MySQL and WordPress deployment

```
kubectl create namespace simplilearn-project7
```

```
labsuser@ip-172-31-0-15:~$ kubectl create namespace simplilearn-project7
namespace/simplilearn-project7 created
labsuser@ip-172-31-0-15:~$ kubectl get namespace
NAME                   STATUS   AGE
calico-apiserver       Active   4h16m
calico-system          Active   4h16m
default                Active   4h20m
kube-node-lease        Active   4h20m
kube-public            Active   4h20m
kube-system            Active   4h20m
kubernetes-dashboard   Active   129m
simplilearn-project7   Active   22s
tigera-operator        Active   4h18m
```

- Create a yaml file for MySQL persistent volume and then run it to be created via kubectl.

  vim persistent.yaml
  kubectl create -f persistent.yaml -n simplilearn-project7

  kubectl get persistentvolume/volume1 -n simplilearn-project7
  NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS AGE
  volume1 10Gi RWX Retain Available 90s

```
labsuser@ip-172-31-0-15:~$ vim persistent.yaml
labsuser@ip-172-31-0-15:~$ kubectl create -f persistent.yaml -n simplilearn-project7
persistentvolume/volume1 created
labsuser@ip-172-31-0-15:~$ kubectl get persistent -n simplilearn-project7
error: the server doesn't have a resource type "persistent"
labsuser@ip-172-31-0-15:~$ kubectl get volume1 -n simplilearn-project7
error: the server doesn't have a resource type "volume1"
labsuser@ip-172-31-0-15:~$ kubectl get persistentvolume/volume1 -n simplilearn-project7
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
volume1   10Gi       RWX            Retain           Available                                   90s
```

- Create another yaml file for persistent volume claim
  vim pv_claim.yaml
  kubectl create -f pv_claim.yaml -n simplilearn-project7
  kubectl get persistentvolumeclaim/mypvc1 -n simplilearn-project7
  NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE
  mypvc1 Bound volume1 10Gi RWX 48s

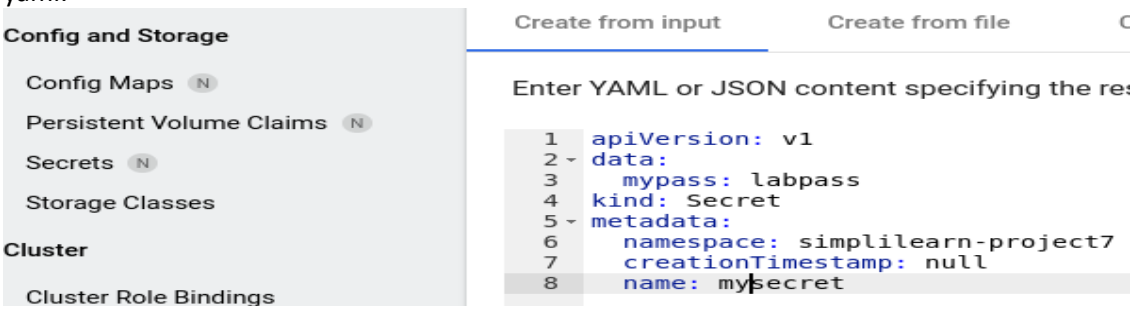
- Create a volume for wordpress

```
labsuser@ip-172-31-0-15:~$ vim wordpress.yaml
labsuser@ip-172-31-0-15:~$ kubectl create -f wordpress.yaml -n simplilearn-project7
persistentvolume/wordpress created
labsuser@ip-172-31-0-15:~$ kubectl get persistentvolume/wordpress -n simplilearn-project7
NAME        CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
wordpress   10Gi       RWX            Retain           Available                                   33s
labsuser@ip-172-31-0-15:~$
```

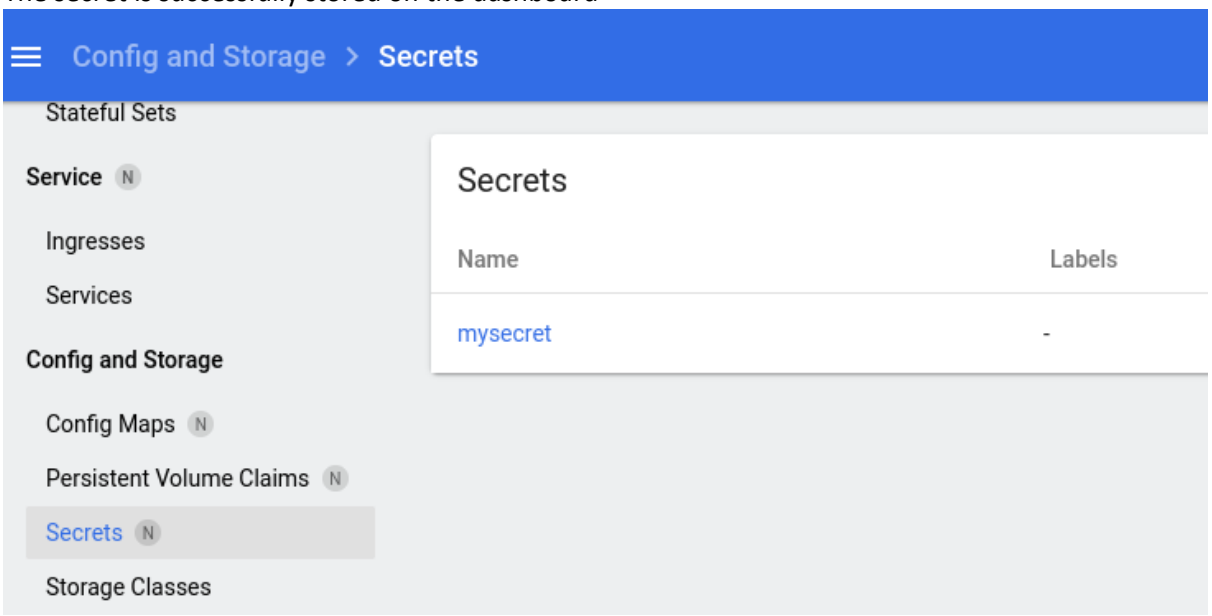
- Create another volume claim for wordpress

```
labsuser@ip-172-31-0-15:~$ vim wp_claim.yaml
labsuser@ip-172-31-0-15:~$ kubectl create -f wp_claim.yaml -n simplilearn-project7
persistentvolumeclaim/wpclaim created
labsuser@ip-172-31-0-15:~$ kubectl get persistentvolume/wpclaim -n simplilearn-project7
Error from server (NotFound): persistentvolumes "wpclaim" not found
labsuser@ip-172-31-0-15:~$ kubectl get persistentvolumeclaim/wpclaim -n simplilearn-project7
NAME      STATUS   VOLUME      CAPACITY   ACCESS MODES   STORAGECLASS   AGE
wpclaim   Bound    wordpress   10Gi       RWX                           63s
labsuser@ip-172-31-0-15:~$
```

- Creating a secret for Mysql Deployment
  In the dashboard navigate to the **secrets** section on the sideboard and enter the following as yaml.



- The secret is successfully stored on the dashboard

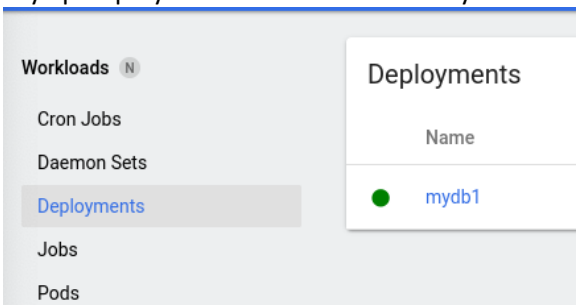


- Deploy Mysql DB application on dashboard

  Enter the following yaml file into the dashboard **Deployments** section

  mysql_deployment.yaml

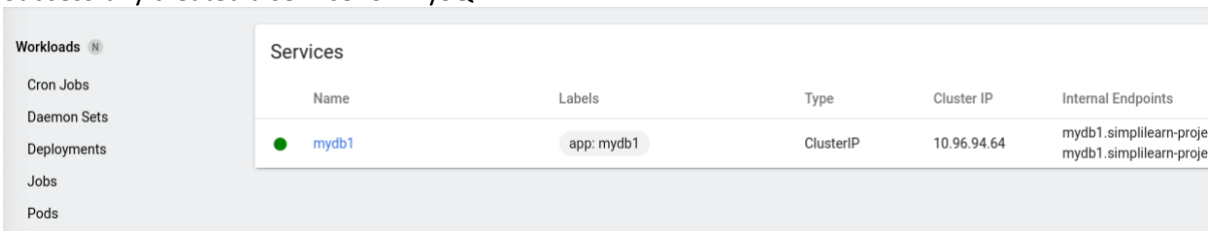
  Mysql Deployment created successfully



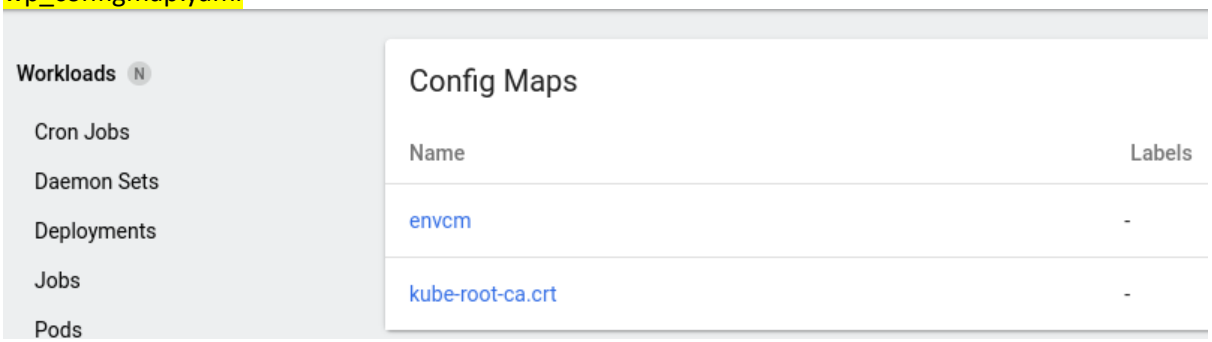
- Create a service for MySQL deployment using yaml

  mysql_service.yaml

  Successfully created a service for MySQL.

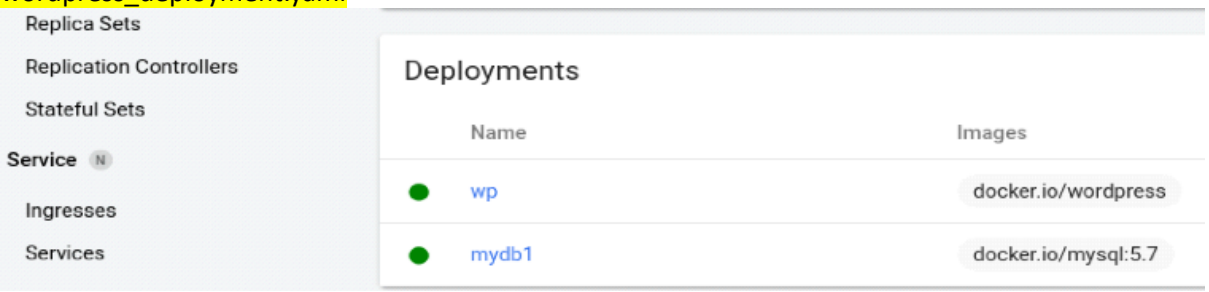


- Creating a config map for Wordpress application
  wp_configmap.yaml



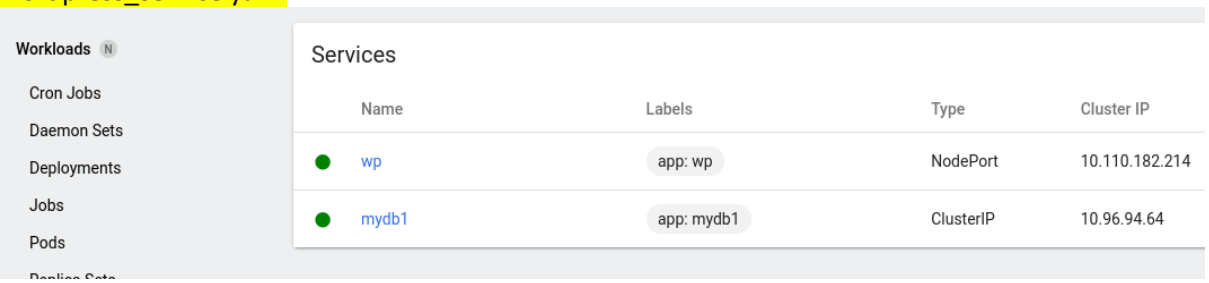
- Deploy the WordPress application on the dashboard yaml

  wordpress_deployment.yaml



- Deploy a service for wordpress application

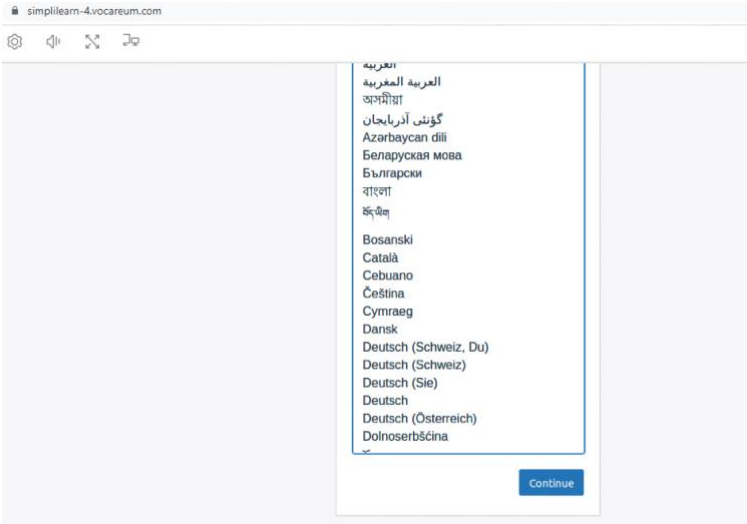
  wordpress_service.yaml

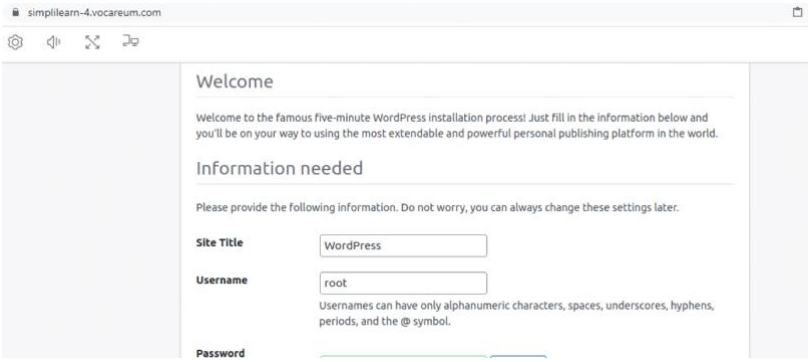


- On terminal verify WordPress Pod IP and NodePort and access that via browser.

  kubectl get svc -n Simplilearn-project7 -o wide

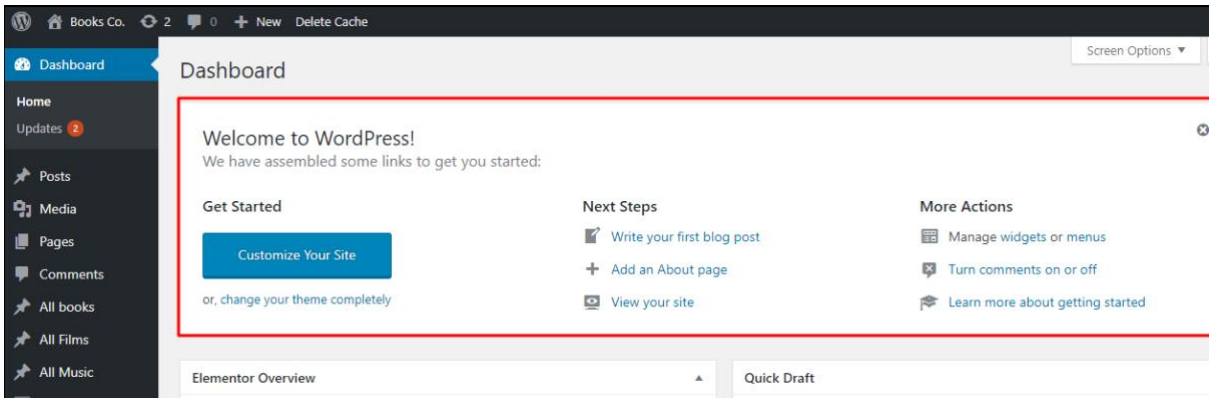
```
labsuser@ip-172-31-0-15:~$ kubectl get svc -n simplilearn-project7
NAME     TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
mydb1    ClusterIP   10.96.94.64      <none>        3306/TCP       41m
wp       NodePort    10.110.182.214   <none>        80:30390/TCP   17m
```





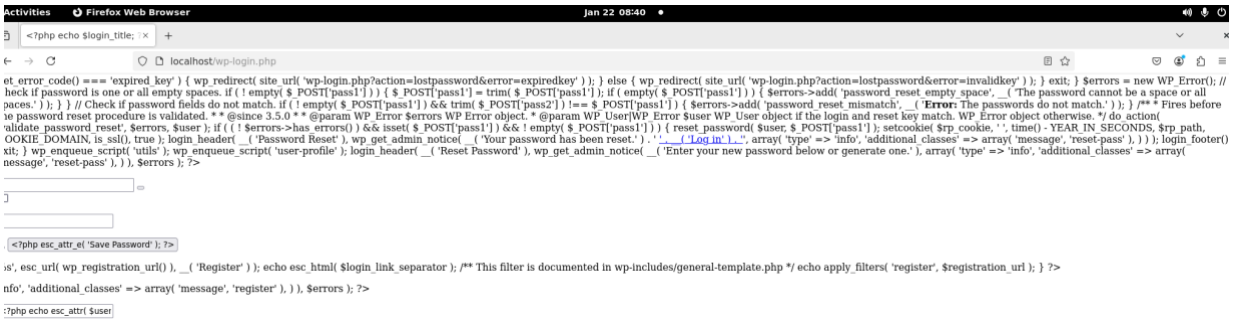
CONCLUSION / RESULT

- Successfully deployed WordPress service and MySQL DB on Kubernetes dashboard!!! ☺☺



ISSUES / WORKAROUNDS DURING THE PROJECT IMPLEMENTATAION.

- I was facing a weird issue where the WordPress page was not properly rendered when accessed via the IP address.



- I was not able to replicate this issue on my local setup.
  So this was a bug on the lab env only.