Deployment of WordPress using Kubernetes Dashboard

By Jesús Manuel Mariño Valcarce

(jesus.manuel.marino at vodafone dot com)

2023/07/10

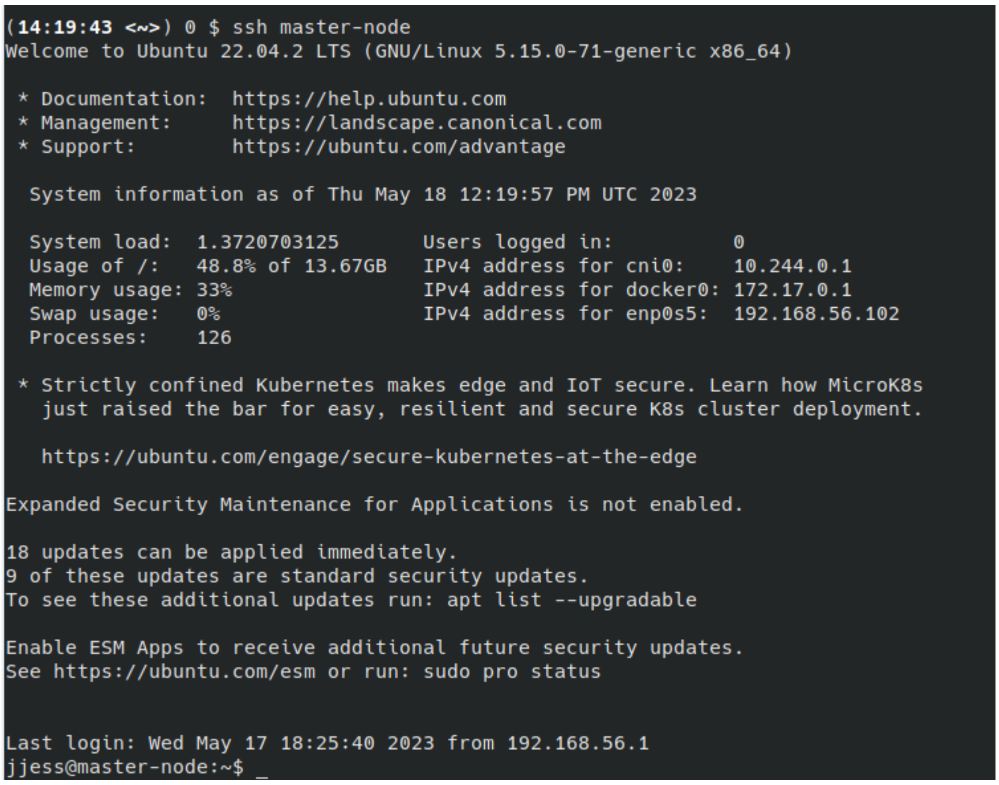
Considerations

I didn't use the SimpliLearn's Lab because I started to play with components discused in the DevOps module before they would be available. I've used my own computer, deploying all necesary components in it by my own means. My lab setup will be detailed below.

Lab Setup

My computer runs FreeBSD 13.2. With the native FreeBSD hypervisor, **bhyve**, I've created 4 vms running Ubuntu 22.04LTS:

master-node / 192.168.56.102



```
(14:19:43 <~>) 0 $ ssh master-node
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-71-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu May 18 12:19:57 PM UTC 2023

  System load:  1.3720703125       Users logged in:          0
  Usage of /:   48.8% of 13.67GB   IPv4 address for cni0:    10.244.0.1
  Memory usage: 33%                IPv4 address for docker0: 172.17.0.1
  Swap usage:   0%                 IPv4 address for enp0s5:  192.168.56.102
  Processes:    126

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

18 updates can be applied immediately.
9 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Wed May 17 18:25:40 2023 from 192.168.56.1
jjess@master-node:~$ _
```

worker-node / 192.168.56.103

worker-node-2 / 192.168.56.104

worker-node-3 / 192.168.56.105

This 4 nodes will have:

- master and worker nodes will conform a Kubernetes cluster
- worker-node-3 will act as NFS server with static volumes.

Kubernetes installation

Kubernetes installation is not covered in this document as it was explained in the previous project deploying Wordpress on Kubernetes with Jenkins.

Project in GitHub

This project is available at:

https://github.com/jjess/Deploy_Application_Using_Kubernetes_Dashboard

Kubernetes Dashboard Installation

I’ve followed the steps explained in the official kubernetes dashboard page:

https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/

First the deployment:

```
> kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/
recommended.yaml
```

As the original deployment uses ClusterIP we must change it to NodePort in order to get connectivity with the frontend:

```
> kubectl edit service/kubernetes-dashboard -n kubernetes-dashboard


apiVersion: v1
kind: Service
metadata:
  annotations:

  ...

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

Then we delete the pod in order to recreated with the NodePort change:

```
> kubectl delete pod kubernetes-dashboard-78c79f97b4-xuc2k -n kubernetes-dashboard


> kubectl get all -n kubernetes-dashboard -o wide

NAME                                             READY   STATUS    RESTARTS        AGE     IP
       NODE            NOMINATED NODE   READINESS GATES
pod/dashboard-metrics-scraper-5cb4f4bb9c-bvchn   1/1     Running   1 (2d18h ago)   2d18h
10.244.2.85   worker-node     <none>           <none>
pod/kubernetes-dashboard-6967859bff-fzdj5        1/1     Running   1 (2d18h ago)   2d18h
10.244.1.8    worker-node-2   <none>           <none>

NAME                                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE     SELECTOR
service/dashboard-metrics-scraper   ClusterIP   10.104.114.198   <none>        8000/TCP
2d18h   k8s-app=dashboard-metrics-scraper
service/kubernetes-dashboard        NodePort    10.109.127.192   <none>        443:32582/TCP
2d18h   k8s-app=kubernetes-dashboard

NAME                                        READY   UP-TO-DATE   AVAILABLE   AGE     CONTAINERS
               IMAGES                                 SELECTOR
deployment.apps/dashboard-metrics-scraper   1/1     1            1           2d18h   dashboard-
metrics-scraper   kubernetesui/metrics-scraper:v1.0.8   k8s-app=dashboard-metrics-scraper
deployment.apps/kubernetes-dashboard        1/1     1            1           2d18h   kubernetes-
dashboard        kubernetesui/dashboard:v2.7.0         k8s-app=kubernetes-dashboard

NAME                                                   DESIRED   CURRENT   READY   AGE
CONTAINERS                  IMAGES                                SELECTOR
replicaset.apps/dashboard-metrics-scraper-5cb4f4bb9c   1         1         1       2d18h
dashboard-metrics-scraper   kubernetesui/metrics-scraper:v1.0.8   k8s-app=dashboard-metrics-
scraper,pod-template-hash=5cb4f4bb9c
replicaset.apps/kubernetes-dashboard-6967859bff        1         1         1       2d18h
kubernetes-dashboard        kubernetesui/dashboard:v2.7.0         k8s-app=kubernetes-
dashboard,pod-template-hash=6967859bff
```

The TCP port for the dashboard has been highlighted in red.

Now the dashboard is available at:

https://192.168.56.102:32582

In order to authenticate into the dashboard I created two users, one is **admin-user** and the other one is **sandry**. Sandry is restricted to the namespace **myproject**.

So, to create the **admin-user** I applied the following yaml files:

dashboard_admin_ServiceAccount.yml :

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
```

dashboard_admin_ClusterRoleBinding.yml :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
```

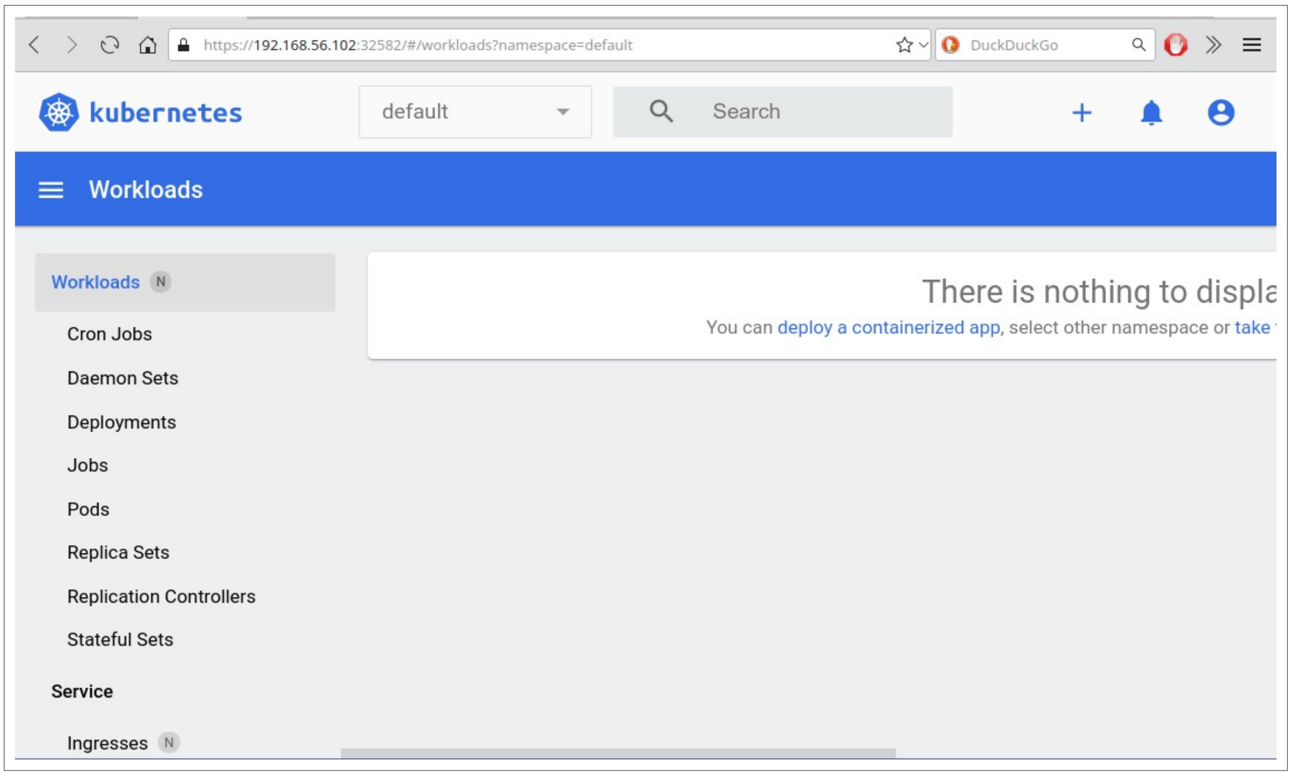
To get the token for the dashboard:

```
> kubectl apply -f dashboard_ServiceAccount.yml
serviceaccount/admin-user created

> kubectl apply -f dashboard_ClusterRoleBinding.yml
clusterrolebinding.rbac.authorization.k8s.io/admin-user created


> kubectl -n kubernetes-dashboard create token admin-user
```

With the token the dashboard looks like:



https://192.168.56.102:32582/#/workloads?namespace=default

There is nothing to displa

You can deploy a containerized app, select other namespace or take

After that I created a namespace called **myproject** to host all the project deployments:

```
> kubectl create namespace myproject
namespace/myproject created
```

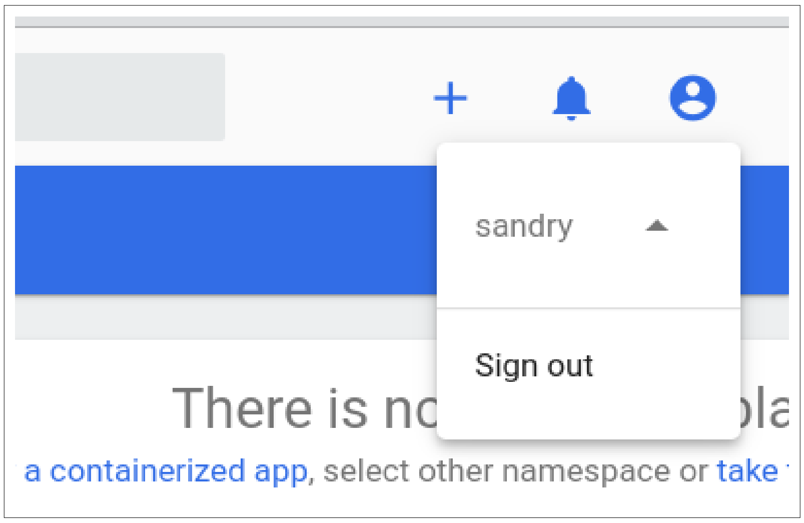
The **Sandry** user was created from command line with:

```
> kubectl create serviceaccount sandry --namespace myproject
serviceaccount/sandry created

> kubectl create clusterrolebinding crb_sandry --serviceaccount=myproject:sandry --
clusterrole=cluster-admin
clusterrolebinding.rbac.authorization.k8s.io/crb_sandry created
```

In order to get the token to authenticate as **sandry**:

```
> kubectl -n kubernetes-dashboard create token sandry -n myproject
```


sandry
Sign out
There is no
a containerized app, select other namespace or take

NFS Server for Storage

As the project ask for a NFS server in worker3 for the storage, I created a NFS Server in my **worker-node-3**.

From a terminal connected to **worker-node-3** I did:

```
> sudo apt update

> sudo apt install nfs-common nfs-kernel-server -y

> sudo mkdir -p /data/nfs3 /data/nfs3/mariadb_data /data/nfs3/wordpress_data
> sudo chown -fR nobody:nogroup /data/nfs3
> sudo chmod -fR g+rwxs /data/nfs3
```

The **/etc/exports** file must include the following lines:

```
# 20230710 Kubernetes NFS Server
/data/nfs3                      192.168.0.0/16(rw,sync,no_subtree_check,no_root_squash)
/data/nfs3/mariadb_data         192.168.0.0/16(rw,sync,no_subtree_check,no_root_squash)
/data/nfs3/wordpress_data       192.168.0.0/16(rw,sync,no_subtree_check,no_root_squash)
```

Check the exported paths:

```
> sudo exportfs -av

exporting 192.168.0.0/16:/data/nfs3/wordpress_data
exporting 192.168.0.0/16:/data/nfs3/mariadb_data
exporting 192.168.0.0/16:/data/nfs3
```

It only rest to install the NFS Client package in all the cluster nodes besides of worker-node-3:

```
> sudo apt update
> sudo apt install nfs-common -y
```

MariaDB deployment

The mariadb deployment as requirement for Wordpress consist of 3 yaml files:

- mariadb-configmap.yaml. Basically contains the URL for the mariadb service.
- mariadb-secret.yaml. It has the password for the root user.
- mariadb-deployment-pvc.yaml. It contains the PV and PVC, the service and the deployment itself.

This files look like:

mariadb-configmap.yaml :

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mariadb-configmap
data:
  database_url: mariadb-internal-service #name of service
```

mariadb-secret.yaml :

```
apiVersion: v1
kind: Secret
metadata:
    name: mariadb-secret
type: Opaque
data:
  mariadb-root-password: c2VjcmV0 #echo -n 'secret'|base64
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mariadb-nfs-pv-claim
  labels:
    app: mariadb
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 300Mi
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mariadb-nfs-pv
  labels:
spec:
  persistentVolumeReclaimPolicy: Delete
  capacity:
    storage: 500Mi
  accessModes:
    - ReadWriteMany
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /data/nfs3/mariadb_data
    server: 192.168.56.105
```

mariadb-deployment-pvc.yaml (PV and PVC):

mariadb-deployment-pvc.yaml (Service):

```
apiVersion: v1
kind: Service
metadata:
  name: mariadb-internal-service
spec:
  type: NodePort
  selector:
    app: mariadb
  ports:
    - port: 3306
      targetPort: 3306
      protocol: TCP
```

mariadb-deployment-pvc.yaml (Deployment):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mariadb-deployment
spec: # specification for deployment resource
  replicas: 1
  selector:
    matchLabels:
      app: mariadb
  template: # blueprint for pods
    metadata:
      labels:
        app: mariadb # service will look for this label
    spec: # specification for pods
      containers:
      - name: mariadb
        image: mariadb
        ports:
        - containerPort: 3306 #default one
        env:
        - name: MARIADB_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mariadb-secret
              key: mariadb-root-password
        - name: MARIADB_DATABASE
          value: wordpress
        volumeMounts:
        - name: mariadb-nfs-pv
          mountPath: /var/lib/mysql
      readinessProbe:
        tcpSocket:
          port: 3306
        initialDelaySeconds: 150
        periodSeconds: 10
      livenessProbe:
        tcpSocket:
          port: 3306
        initialDelaySeconds: 120
        periodSeconds: 20
      volumes:
      - name: mariadb-nfs-pv
        persistentVolumeClaim:
          claimName: mariadb-nfs-pv-claim
```

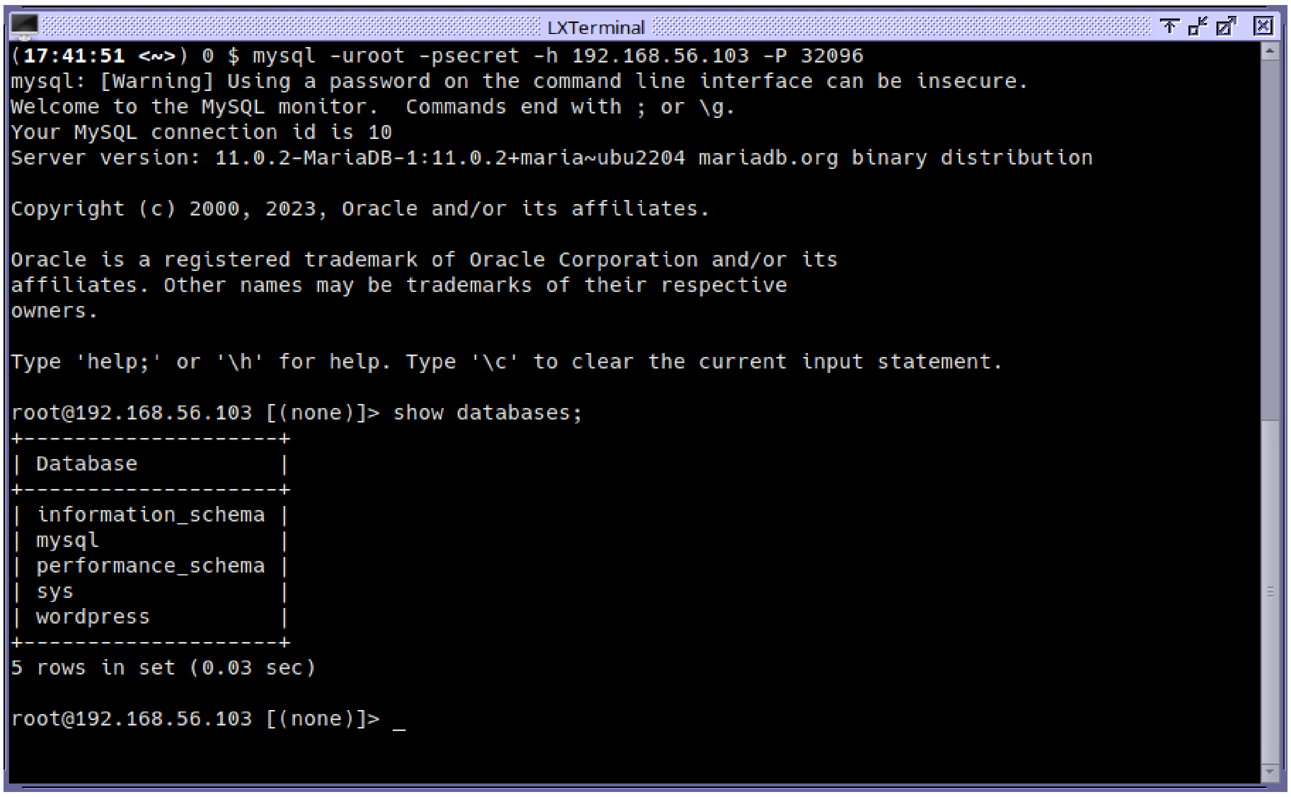
Note (in red) the probes configured into this pod.

As the service was created with NodePort, the database server should be accessed outside Kubernetes:

```
> kubectl get svc -n myproject -o wide
NAME                        TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
SELECTOR
mariadb-internal-service    NodePort   10.103.45.95    <none>        3306:32096/TCP   151m
app=mariadb
```

From a terminal having network connectivity with the cluster:

```
> mysql -uroot -psecret -h 192.168.56.103 -P 32096
```



```
(17:41:51 <~>) 0 $ mysql -uroot -psecret -h 192.168.56.103 -P 32096
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 11.0.2-MariaDB-1:11.0.2+maria~ubu2204 mariadb.org binary distribution

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

root@192.168.56.103 [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| wordpress          |
+--------------------+
5 rows in set (0.03 sec)

root@192.168.56.103 [(none)]> _
```

WordPress deployment

In a similar way as with mariadb, Wordpress is deployed with a PV/PVC using NFS and a Service configured with type NodePort. The whole deployment is located in the file:

wordpress-deployment-pvc.yaml (PV and PVC):

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wordpress-nfs-pv-claim
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 300Mi
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: wordpress-nfs-pv
  labels:
spec:
  persistentVolumeReclaimPolicy: Delete
  capacity:
    storage: 500Mi
  accessModes:
    - ReadWriteMany
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /data/nfs3/wordpress_data
    server: 192.168.56.105
```

wordpress-deployment-pvc.yaml (Service):

```
apiVersion: v1
kind: Service
metadata:
  name: wordpress
spec:
  selector:
    app: wordpress
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP #default
      nodePort: 31000
  type: NodePort
```

Note the NodePort 31000 manually specified.

wordpress-deployment-pvc.yaml (Deployment):

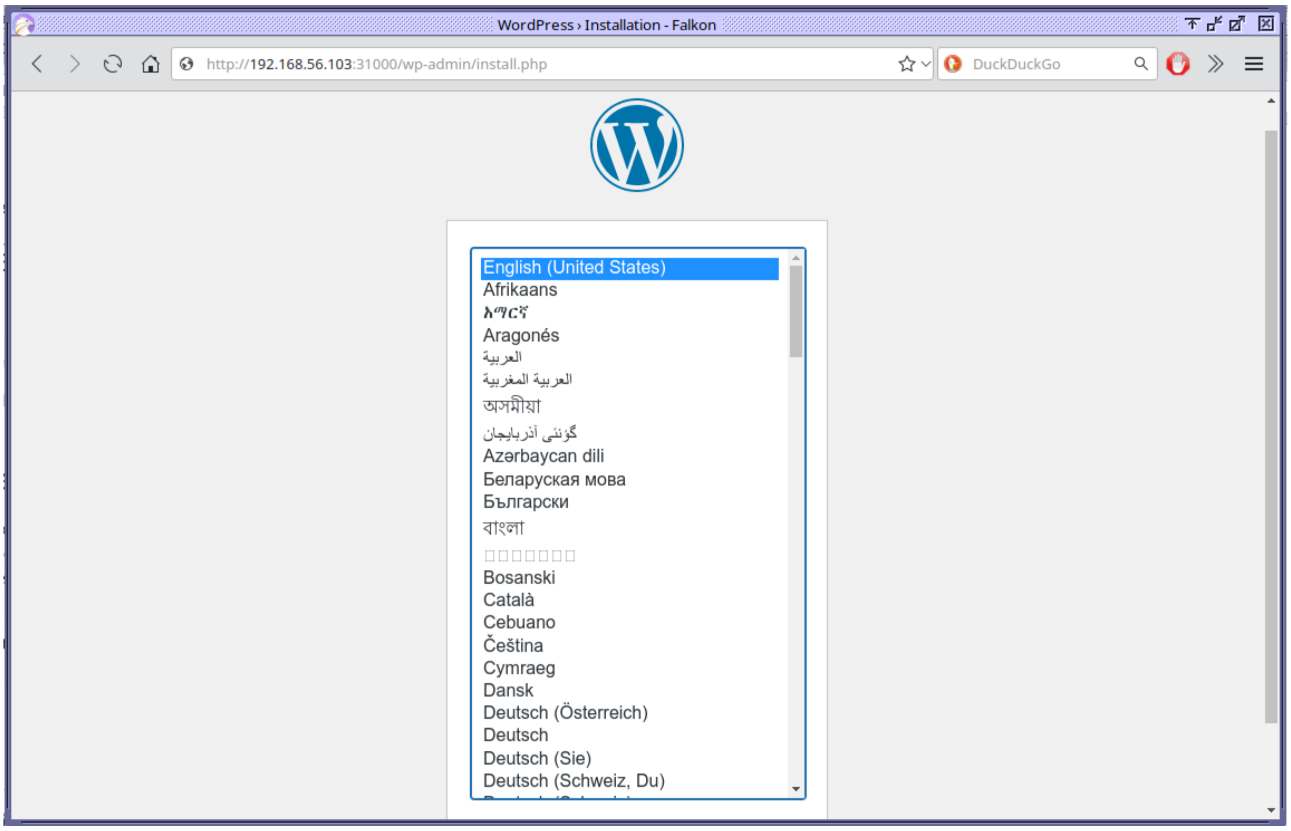
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-deployment
spec: # specification for deployment resource
  replicas: 1
  selector:
    matchLabels:
      app: wordpress
  template: # blueprint for Pod
    metadata:
      labels:
        app: wordpress
    spec: # specification for Pod
      containers:
      - name: wordpress
        image: wordpress:latest
        ports:
        - containerPort: 80
        env:
        - name: WORDPRESS_DB_HOST
          valueFrom:
            configMapKeyRef:
              name: mariadb-configmap
              key: database_url
        - name: WORDPRESS_DB_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mariadb-secret
              key: mariadb-root-password
        - name: WORDPRESS_DB_USER
          value: root
        - name: WORDPRESS_DEBUG
          value: "1"
        volumeMounts:
        - name: wordpress-nfs-pv
          mountPath: /var/www/html
      livenessProbe:
        tcpSocket:
          port: 3306
        initialDelaySeconds: 120
        periodSeconds: 20
      volumes:
      - name: wordpress-nfs-pv
        persistentVolumeClaim:
          claimName: wordpress-nfs-pv-claim
```

Note the livenessProbe (in red) configured. If the mariadb port is not available (tcp port 3306 reachable) the pod will be recreated.

In the Wordpress Service we specify the NodePort TCP port, 31000, so now we can test the connectivity openning a browser and typing the following URL:

http://192.168.56.103:31000

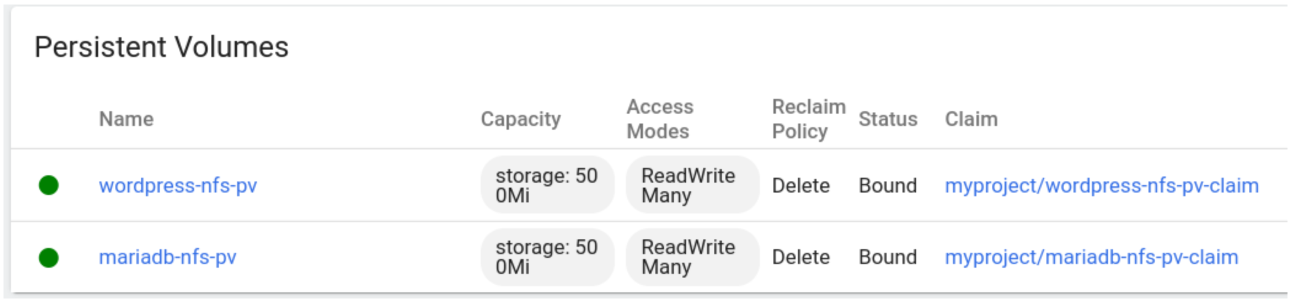
The browser looks like:


WordPress › Installation - Falkon
http://192.168.56.103:31000/wp-admin/install.php
DuckDuckGo

English (United States)
Afrikaans
አማርኛ
Aragonés
العربية
العربية المغربية
অসমীয়া
گؤنئی آذربایجان
Azərbaycan dili
Беларуская мова
Български
বাংলা
□□□□□□□
Bosanski
Català
Cebuano
Čeština
Cymraeg
Dansk
Deutsch (Österreich)
Deutsch
Deutsch (Sie)
Deutsch (Schweiz, Du)

Screenshots in Kubernetes Dashboard

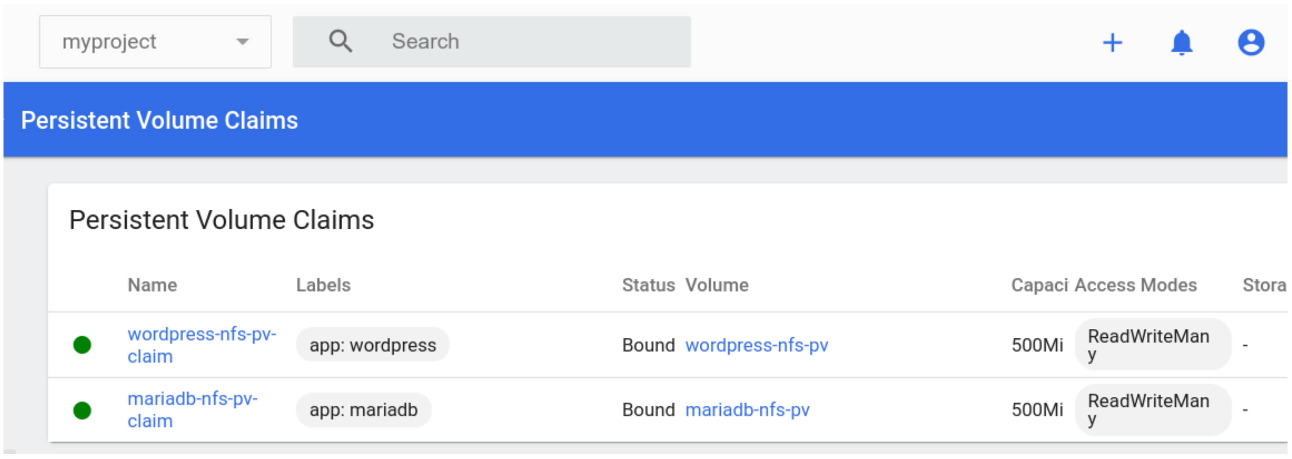
Persistent Volumes:



Persistent Volumes

	Name	Capacity	Access Modes	Reclaim Policy	Status	Claim
●	wordpress-nfs-pv	storage: 500Mi	ReadWriteMany	Delete	Bound	myproject/wordpress-nfs-pv-claim
●	mariadb-nfs-pv	storage: 500Mi	ReadWriteMany	Delete	Bound	myproject/mariadb-nfs-pv-claim

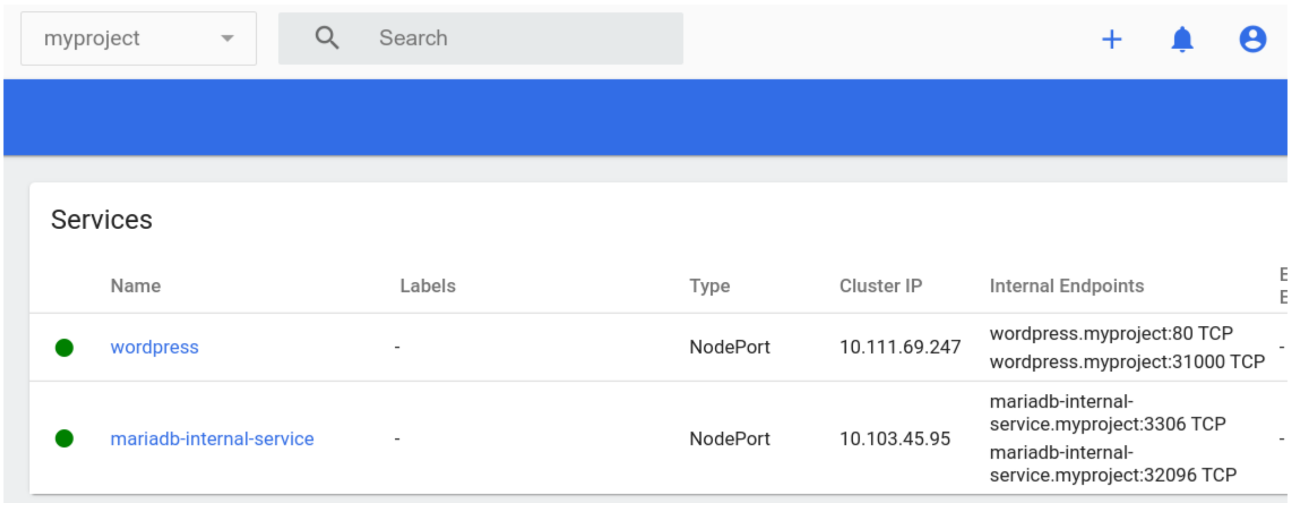
Persistent Volume Claims:



Persistent Volume Claims

	Name	Labels	Status	Volume	Capaci	Access Modes	Stora
●	wordpress-nfs-pv-claim	app: wordpress	Bound	wordpress-nfs-pv	500Mi	ReadWriteMany	-
●	mariadb-nfs-pv-claim	app: mariadb	Bound	mariadb-nfs-pv	500Mi	ReadWriteMany	-

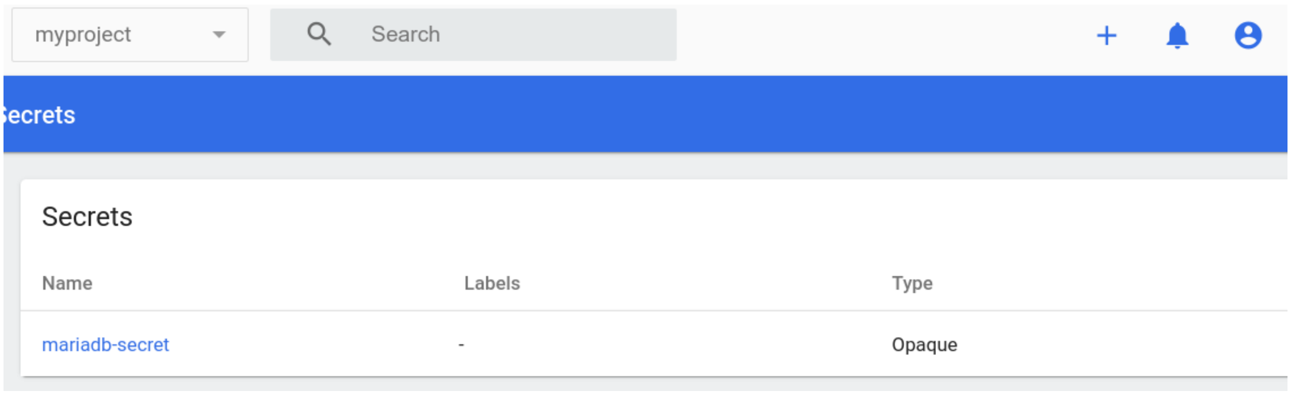
Services:



Services

	Name	Labels	Type	Cluster IP	Internal Endpoints	E E
●	wordpress	-	NodePort	10.111.69.247	wordpress.myproject:80 TCP wordpress.myproject:31000 TCP	-
●	mariadb-internal-service	-	NodePort	10.103.45.95	mariadb-internal-service.myproject:3306 TCP mariadb-internal-service.myproject:32096 TCP	-

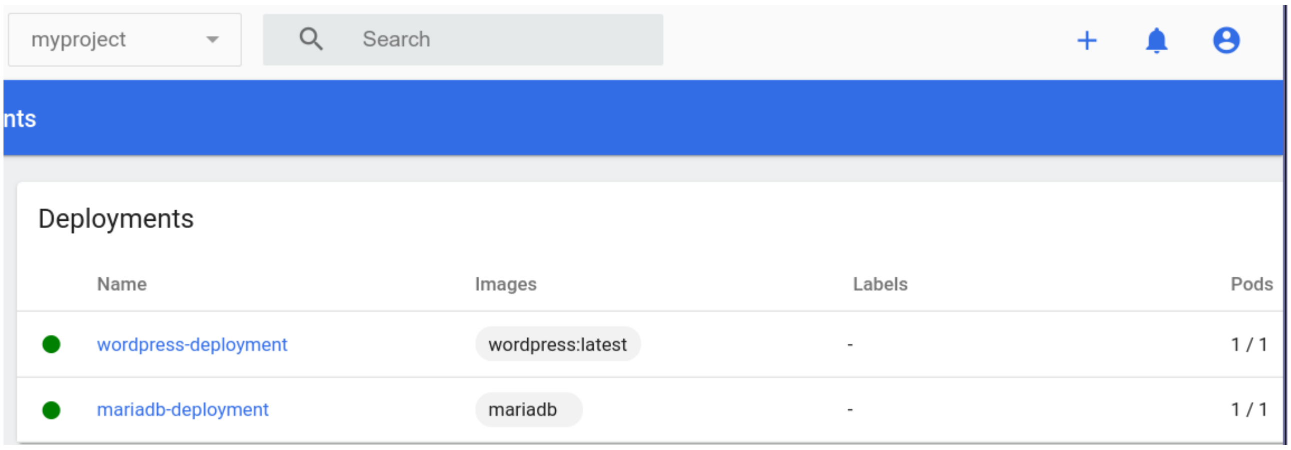
Secrets:



Secrets

Name	Labels	Type
mariadb-secret	-	Opaque

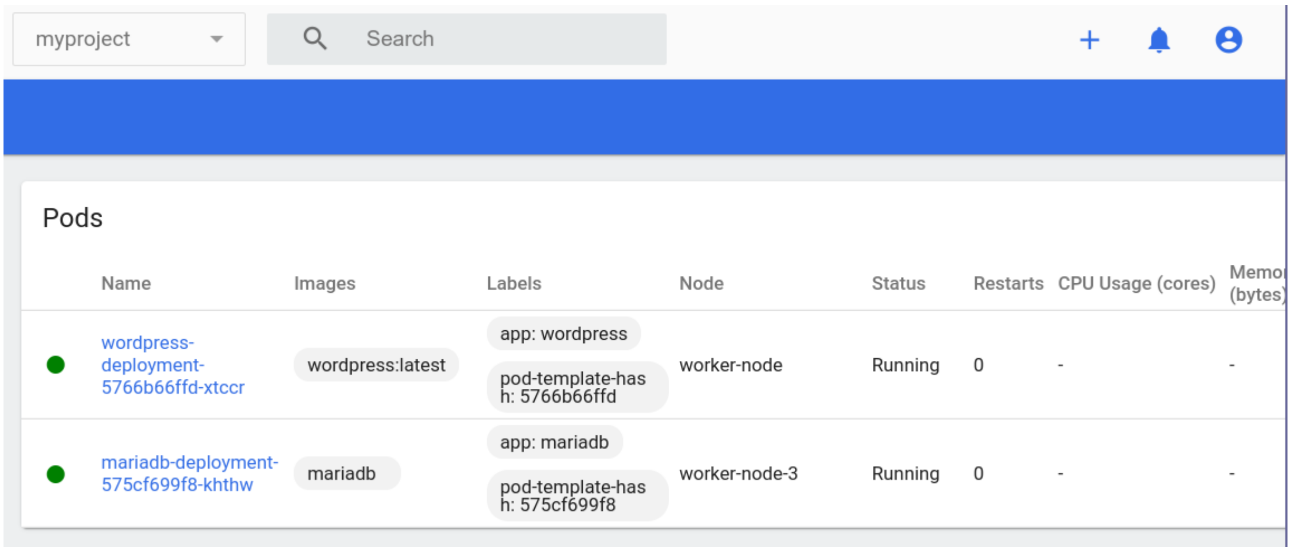
Deployments:



Deployments

	Name	Images	Labels	Pods
●	wordpress-deployment	wordpress:latest	-	1 / 1
●	mariadb-deployment	mariadb	-	1 / 1

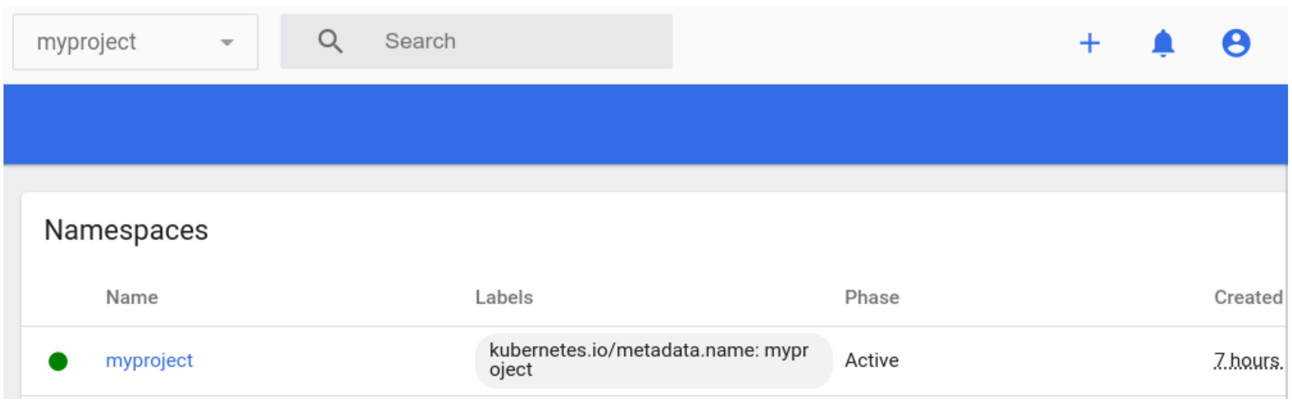
Pods:



Pods

	Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memo (bytes)
●	wordpress-deployment-5766b66ffd-xtccr	wordpress:latest	app: wordpress pod-template-hash: 5766b66ffd	worker-node	Running	0	-	-
●	mariadb-deployment-575cf699f8-khthw	mariadb	app: mariadb pod-template-hash: 575cf699f8	worker-node-3	Running	0	-	-

Namespaces:



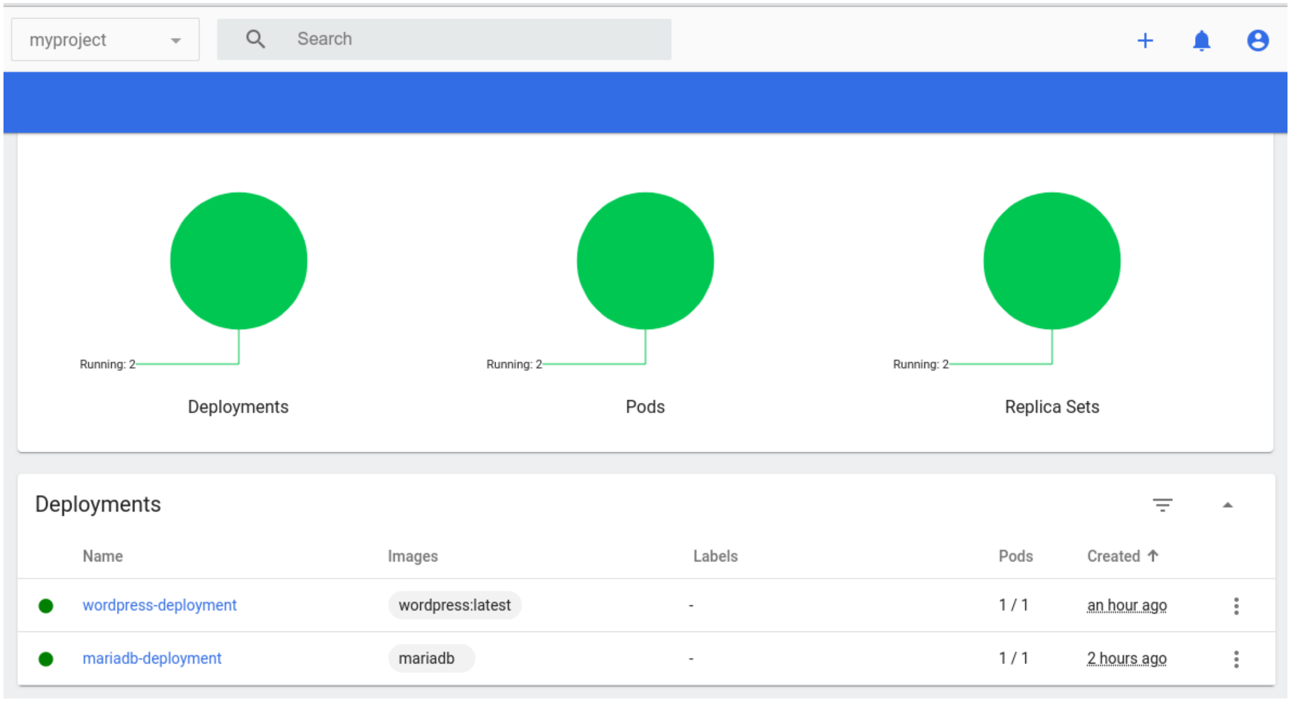
myproject

Search

Namespaces

	Name	Labels	Phase	Created
●	myproject	kubernetes.io/metadata.name: myproject	Active	7 hours

Workloads:



myproject

Search

Deployments

	Name	Images	Labels	Pods	Created ↑
●	wordpress-deployment	wordpress:latest	-	1 / 1	an hour ago
●	mariadb-deployment	mariadb	-	1 / 1	2 hours ago