Microservice orchestration platforms Using Kubernetes

Project Report

Implementation of the ownCloud system based on the Kubernetes cluster

Authors:

- **Ismoil Atajanov**
- **Brijesh Varsani**
- **Jiahao Tang**
- **Maria Vazquez**

Task description

- The aim of this exercise is to design and deploy production-ready (in terms of reliability, security, and efficiency) Kubernetes cluster hosting system of ownCloud services hereinafter referred to as system services.
- The system should be composed of software components and publicly available images encapsulating services necessary to implement the following features.

Solution

Following features have been implemented:

- [x] system services are available at the dedicated DNS name or at least exposed locally
- [x] all configuration is contained in a dedicated namespace
- [x] database service is not available from outside the cluster
- [x] data persistence is ensured (i.e. data is stored independently of the system services container(s))
- [x] system services instances are multiplied to achieve basic availability
- [x] basic cluster monitoring is deployed (e.g. Kuberbetes Dashboard)

Implementation

Following kubectl commands aliases were set for the sake of simplicity:

Alias	Command
kget	kubectl get
klogs	kubectl logs
kpods	kget pods
kdelete	kubectl delete
kdeletef	kubectl delete -f
kapply	kubectl apply -f
krestart	kubectl rollout restart
kdrestart	krestart deployment

Using prepared yaml configuration files, following main kubernetes components have been created:

1. Deployment + service - owncloud
2. StatefulSet + service - mariadb
3. Ingress

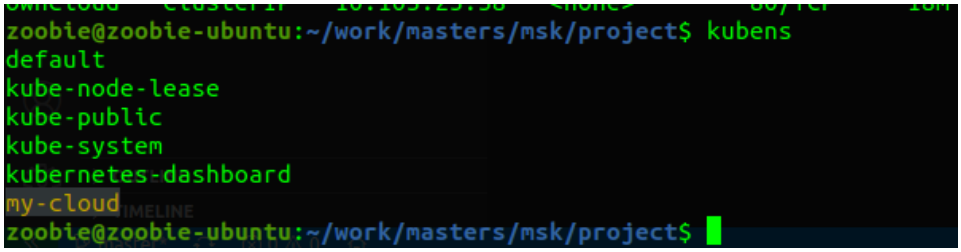
Namespace

All kubernetes components were placed in a new **my-cloud** namespace created from init.yaml

```
apiVersion: v1
kind: Namespace
metadata:
  name: my-cloud
```

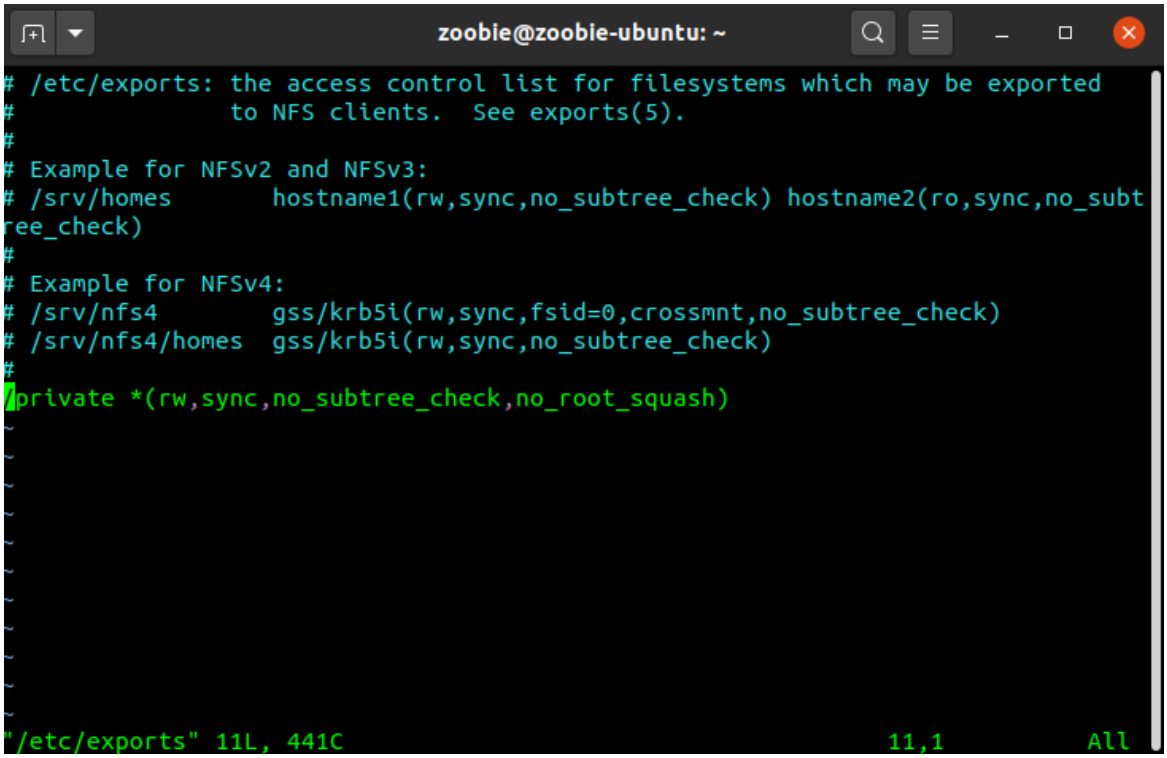
```
    labels:
      name: my-cloud
```

With ease of namespaces management in mind, **kubens** tool was installed and set to **my-cloud** as the default namespace.



Storage

For storing the data NFS Persistence Volume was configured.
Nfs server was set up locally on the same machine using nfs-kernel-server. /private directory was created with 777 access parameters and exported:



Commands to export nfs directory and start nfs server:

```
sudo exportfs -arvf
sudo systemctl start nfs-kernel-server
```

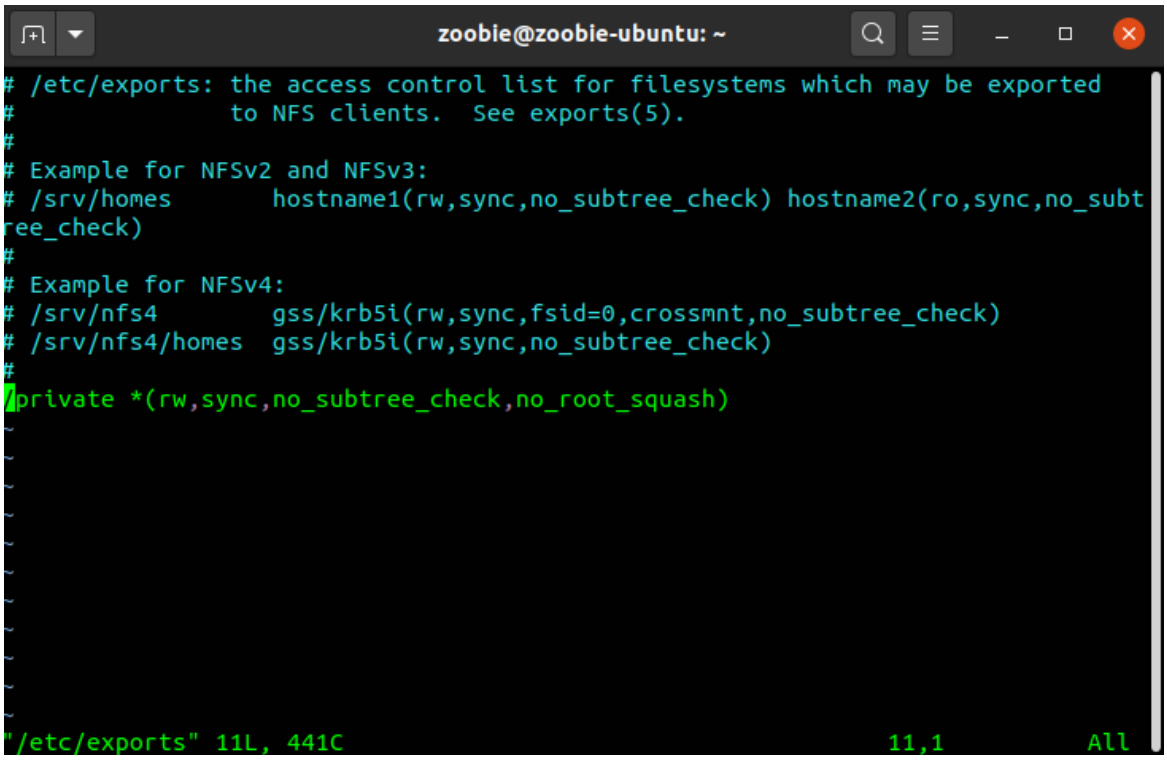
Persistence Volume and Persistence Volume Claim

Volume configuration file for PV & PVC using NFS Storage storage.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-owncloud
  namespace: my-cloud
spec:
  capacity:
    storage: 1Gi
  volumeMode: Filesystem
  accessModes:
```

```
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: nfs
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /private
    server: 192.168.0.24
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-owncloud
  namespace: my-cloud
spec:
  storageClassName: nfs
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

As a result following PV and PVC resources were created:



Created volume was used further in implementation to mount volumes for the database and owncloud.

Database configuration

MariaDB was chosen as the main database server, and it was implemented as a single replica stateful application defined in mariadb.yaml

- MariaDB username and password were defined in **mariadb-secret.yaml**
- MariaDB environment variables were defined in **config-map.yaml**
- MariaDB application configuration was defined in **mariadb.yaml**

  > In order to configure the database correctly **all the configurations** must be applied *in the same order*

####Previously created persistent volume claim is used here to mount volume for mariadb database.

```
        volumeMounts:
          - name: storage
            mountPath: /var/lib/mysql
            subPath: mysql
        volumes:
          - name: storage
            persistentVolumeClaim:
              claimName: pvc-owncloud
```

Owncloud configuration

- In order to start only single configuration owncloud.yaml is required.

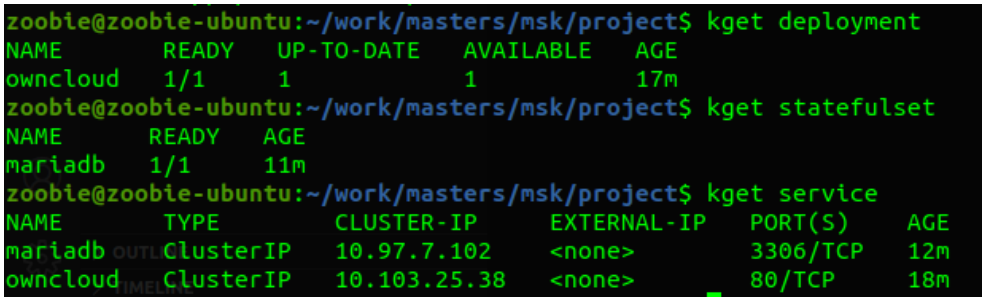
Most important part of the configuration is the container image which was set to ***owncloud*** and volume mounts suggested by the official documentation for the image. Volume mounts are again mounted on the PVC created earlier:

```
        volumeMounts:
          - name: owncloud-storage
            mountPath: /var/www/html/data
            subPath: owncloud/data
        volumeMounts:
          - name: owncloud-storage
            mountPath: /var/www/html/apps
            subPath: owncloud/apps
        volumeMounts:
          - name: owncloud-storage
            mountPath: /var/www/html/config
            subPath: owncloud/config
        volumes:
          - name: owncloud-storage
            persistentVolumeClaim:
              claimName: pvc-owncloud
```

- Owncloud application can be easily re-scaled using `kubectl scale deployment owncloud --replicas=5`

```
zoobie@zoobie-ubuntu:/private/owncloud$ kpods
NAME                       READY   STATUS    RESTARTS   AGE
mariadb-0                  1/1     Running   0          34m
owncloud-7c5b5b9c59-94rqh  1/1     Running   0          16s
owncloud-7c5b5b9c59-rzm2k  1/1     Running   0          4m24s
zoobie@zoobie-ubuntu:/private/owncloud$ kubectl scale deployment owncloud --replicas=3
deployment.apps/owncloud scaled
zoobie@zoobie-ubuntu:/private/owncloud$ kpods
NAME                       READY   STATUS    RESTARTS   AGE
mariadb-0                  1/1     Running   0          35m
owncloud-7c5b5b9c59-94rqh  1/1     Running   0          54s
owncloud-7c5b5b9c59-pp7g4  1/1     Running   0          10s
owncloud-7c5b5b9c59-rzm2k  1/1     Running   0          5m2s
zoobie@zoobie-ubuntu:/private/owncloud$ kubectl scale deployment owncloud --replicas=1
deployment.apps/owncloud scaled
zoobie@zoobie-ubuntu:/private/owncloud$ kpods
NAME                       READY   STATUS    RESTARTS   AGE
mariadb-0                  1/1     Running   0          35m
owncloud-7c5b5b9c59-rzm2k  1/1     Running   0          5m25s
```

The two created applications (deployment and statefulset) were created together with internal services to provide access to the pods



Ingress

Next step of the implementation was to configure nginx ingress to provide external access to owncloud service. The host for the owncloud service access was set to ***my-cloud.site***. After applying the ingress.yaml configuration owncloud application was exposed at the given host.

- Ingress rules

```yaml
rules:
  - host: my-cloud.site
    http:
      paths:
        - path: "/"
          pathType: Prefix
          backend:
            service:
              name: owncloud
              port:
                number: 80
```

Kubernetes Dashboard

Kubernetes dashboard resources come together with minikube installation. To use them a number of minikube addons have to be enabled.

- Below is the list of all minikube addons enabled for this project:

```
zoobie@zoobie-ubuntu:~/work/masters/msk/project$ minikube addons list
|-----------------------------|----------|--------------|
|         ADDON NAME          | PROFILE  |    STATUS    |
|-----------------------------|----------|--------------|
| ambassador                  | minikube | disabled     |
| csi-hostpath-driver         | minikube | disabled     |
| dashboard                   | minikube | enabled ✅   |
| default-storageclass        | minikube | enabled ✅   |
| efk                         | minikube | disabled     |
| freshpod                    | minikube | disabled     |
| gcp-auth                    | minikube | disabled     |
| gvisor                      | minikube | disabled     |
| helm-tiller                 | minikube | disabled     |
| ingress                     | minikube | enabled ✅   |
| ingress-dns                 | minikube | disabled     |
| istio                       | minikube | disabled     |
| istio-provisioner           | minikube | disabled     |
| kubevirt                    | minikube | disabled     |
| logviewer                   | minikube | disabled     |
| metallb                     | minikube | disabled     |
| metrics-server              | minikube | enabled ✅   |
| nvidia-driver-installer     | minikube | disabled     |
| nvidia-gpu-device-plugin    | minikube | disabled     |
| olm                         | minikube | disabled     |
| pod-security-policy         | minikube | disabled     |
| registry                    | minikube | disabled     |
| registry-aliases            | minikube | disabled     |
| registry-creds              | minikube | disabled     |
| storage-provisioner         | minikube | enabled ✅   |
| storage-provisioner-gluster | minikube | disabled     |
| volumesnapshots             | minikube | disabled     |
|-----------------------------|----------|--------------|
```

Dashboard ingress

As dashboard service exists in a different namespace (kubernetes-dasboard), a new dashboard-ingress.yaml was created to configure an external host to access the dashboard service. The service was exposed at host ***dashboard.my-cloud.size***.

DNS Domain names

- After ingress configurations were applied both of them can be view using `kubectl get ingress` command

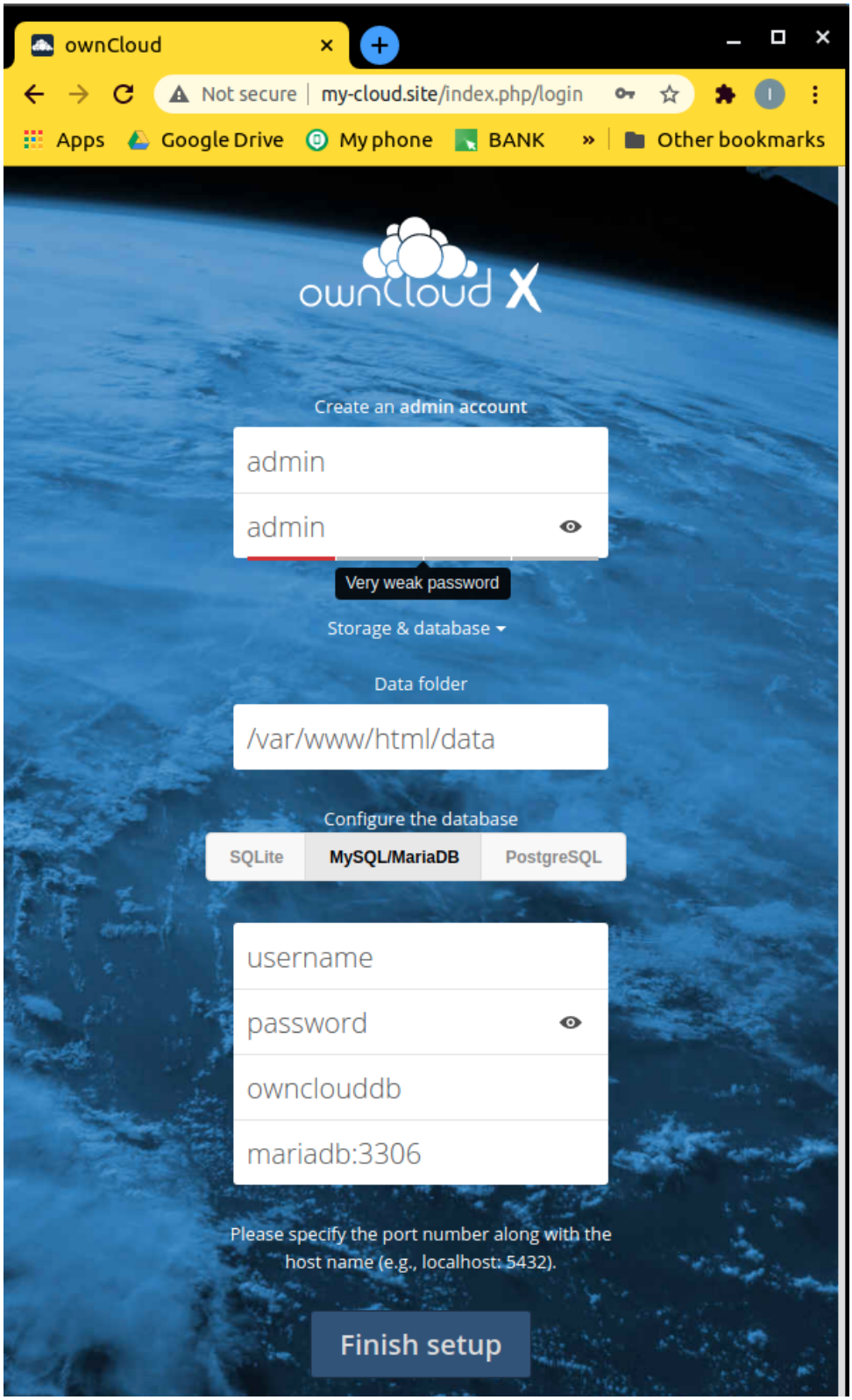
```
zoobie@zoobie-ubuntu:~/work/masters/msk/project$ kget ingress
NAME        CLASS    HOSTS           ADDRESS        PORTS   AGE
owncloud    <none>   my-cloud.site   192.168.49.2   80      8m31s
zoobie@zoobie-ubuntu:~/work/masters/msk/project$ kget ingress -n kubernetes-dashboard
NAME                CLASS    HOSTS                     ADDRESS        PORTS   AGE
dashboard-ingress   <none>   dashboard.my-cloud.site   192.168.49.2   80      7m42s
```

- However, in order for this to work the hosts have to be added to /etc/hosts to be resolved properly by DNS.

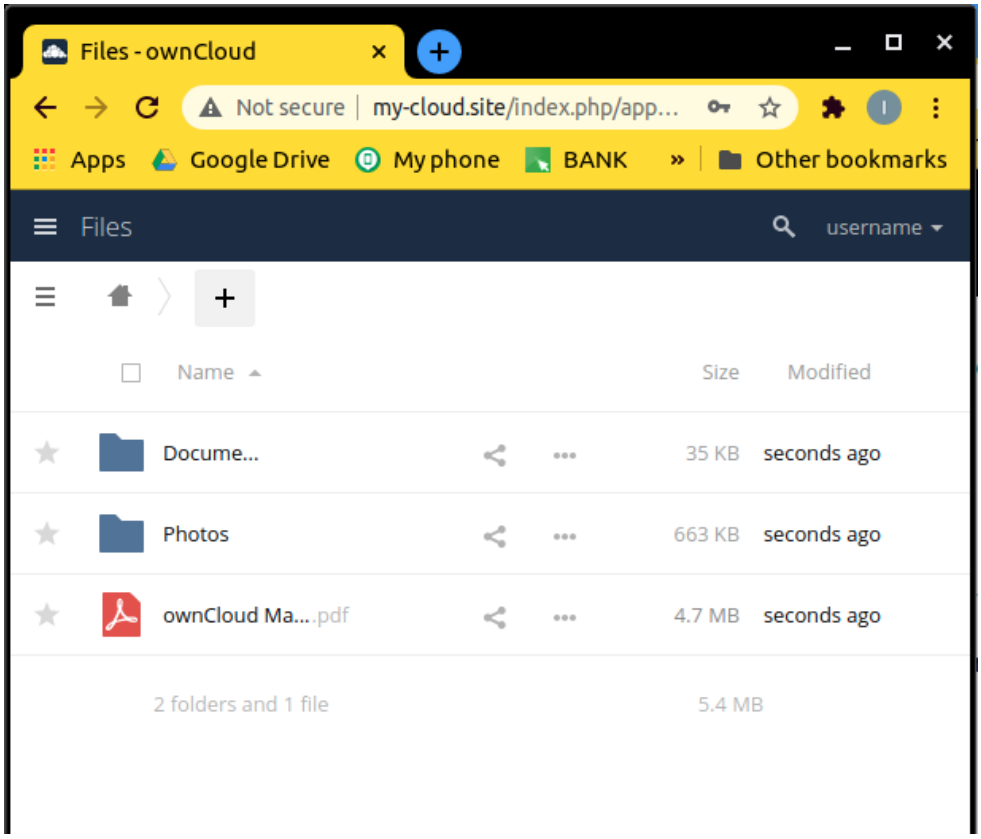
```
zoobie@zoobie-ubuntu: ~/work/masters/msk/projec
127.0.0.1       localhost
127.0.1.1       zoobie-ubuntu

# The following lines are desirable for IPv6 capable hos
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
192.168.49.2 my-cloud.site
192.168.49.2 dashboard.my-cloud.site
```

Results:

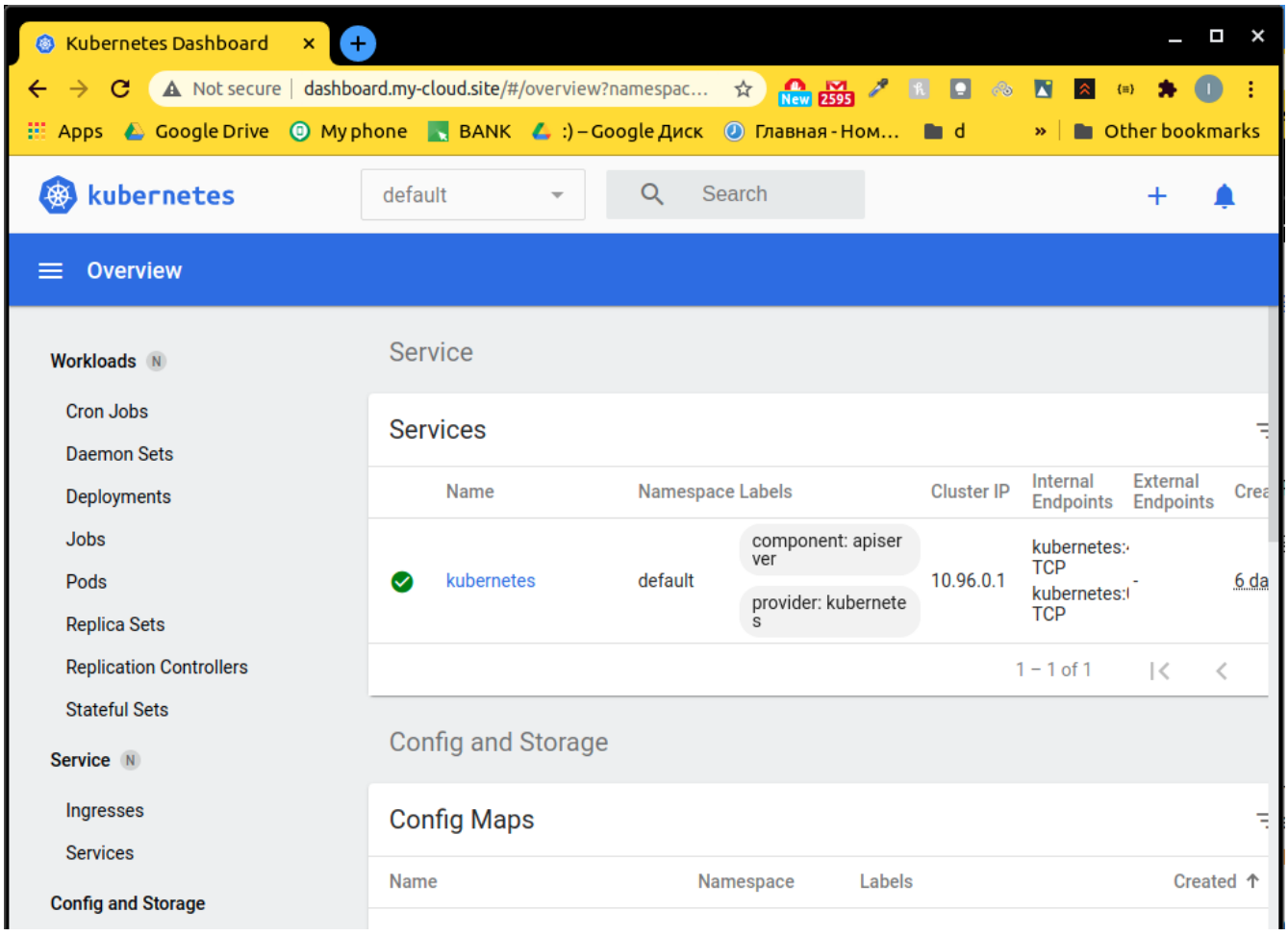
- After all the steps completion ***owncloud*** service is available at my-cloud.site and ***dashboard*** at dashboard.my-cloud.site



> To start using owncloud one has to provide credentials and select preferred database and db credentials, which in the case are mariadb/mysql.



> After log in, main page of owncloud appears, and the application is ready to use at this point.



> Dashboard page is also present and functional at dashboard.my-cloud.site