

Version - **2025.01**

Last update : 2025/01/18 17:05

# DOE310 - StatefulSets, Advanced StorageClass Usage, Creating a Helm Chart and Monitoring

## Contents

- **DOE310 - StatefulSets, Advanced StorageClass Usage, Creating a Helm Chart and Monitoring**
    - Contents
    - StatefulSets
      - Overview
      - LAB #1 - Setting up a simple StatefulSet
        - 1.1 - Service and StatefulSet creation
        - 1.2 - Scaling Up a StatefulSet
        - 1.3 - Scaling Down a StatefulSet
        - 1.4 - Deleting a StatefulSet
    - Advanced StorageClass Usage
      - LAB #2 - Dynamic NFS provisioning
        - 2.1 - NFS Server Configuration
        - 2.2 - NFS Client Configuration
        - 2.3 - Configuring K8s
        - 2.4 - Creating a PersistentVolumeClaim
        - 2.5 - Using the PersistentVolumeClaim with a Pod
        - 2.6 - Creating a Second PersistentVolumeClaim
        - 2.7 - Deleting the PersistentVolumeClaims
    - Creating a Helm Chart
      - Overview
      - LAB #3 - Creating a Simple Helm Package
-

- 3.1 - The values.yaml File
- 3.2 - Templates
- 3.3 - Installation and Removal
- Monitoring
  - Overview
  - LAB #4 - Implementing a Prometheus Solution
    - 4.1 - Stack Deployment with Helm
    - 4.2 - Viewing Data with Grafana
    - 4.3 - Viewing Alerts with the Prometheus Web UI

## StatefulSets

### Overview

A StatefulSet is a Kubernetes component used for Stateful Applications.

Examples of Stateful Applications are :

- MySQL
- elasticsearch
- mongoDB

These applications record client data a one session for use in the next session. The recorded data is called the application's state.

Stateful Applications are deployed using a StatefulSet, while applications **without** state are deployed using a **Deployment**.

StatefulSets and Deployments are similar in that both replicate multiple pods based on an **identical** specification of a container.

The difference between a StatefulSet and a Deployment is that in a StatefulSet the pods are **not** identical and have what is known as a **Pod Identity**. As a result, pods :

- cannot be created or deleted at the same time,
  - cannot be randomly addressed.
-

Let's take the case of a StatefulSet containing three replicas of a MySQL pod:

- mysql-0
- mysql-1
- mysql-2

Note that :

- the pod name takes the form **\$(Name\_of\_StatefulSet)-\$(ordinal)** where the ordinal starts at **0**.
- the StatefulSet will not create the next pod until the previous pod is in a **Running** state
- When the StatefulSet is deleted, or in the case of a **scale down**, pods are deleted in the reverse order in which they were created, e.g. **mysql-2 > mysql-1 > mysql-0**. Each pod must be completely deleted before K8s deletes the next.

In this case of our StatefulSet, the three pods :

- cannot all accept write requests, as this would result in inconsistent data,
- can all accept read requests.

As a result, a StatefulSet mechanism chooses a **master** to accept write requests, for example:

- mysql-0 - write / read - **Master**
- mysql-1 - read only - **slave**
- mysql-2 - read only - **slave**

So there's a clear difference between the Master pod and the two Slave pods.

The difference between the two Slave pods can be explained by the fact that the pods do **not** use the same persistent, remote physical storage:

- mysql-0 - **/data/vol/pv1**
- mysql-1 - **/data/vol/pv2**
- mysql-2 - **/data/vol/pv3**

To ensure that each pod contains the same data, a continuous replication mechanism must be set up between the two slave pods and the master pod.

If a new pod is added to the MySQL **cluster**, it must first clone the data from the last pod into the existing cluster, then start replicating data with the Master:

---

- mysql-0 - data
- mysql-1 - data replicated from mysql-0
- mysql-2 - data replicated from mysql-0
- mysql-3 - clone of mysql-2 pod data, then mysql-0 replica data.

The state of each pod, including its Pod Identity, is stored in physical storage alongside the data. As a result, when a pod is replaced, and a new pod added, the new pod inherits the old pod's identity.

For example, if we delete the **mysql-1** pod, we get :

- mysql-0 - /data/vol/pv1
- pod deleted - /data/vol/pv2 = persistent, the remote physical storage is **not deleted**
- mysql-2 - /data/vol/pv3
- mysql-3 - /data/vol/pv4

By adding a replacement pod, we obtain :

- mysql-0
- mysql-1 ««««« The /data/vol/pv2 is attached to the pod. The new pod is called **mysql-1** and not **mysql-4**.
- mysql-2
- mysql-3

When a ReplicaSet is created, a load balancing service is created. This service assigns a unique **DNS Endpoint** to each pod. The DNS Endpoint takes the form **\$(Pod\_name).\$(Service\_name).\$(namespace).svc.cluster.local** :

- mysql-0 - **mysql-0.mysvc.default.svc.cluster.local**
- mysql-1 - **mysql-1.mysvc.default.svc.cluster.local**
- mysql-2 - **mysql-2.mysvc.default.svc.cluster.local**
- mysql-3 - **mysql-3.mysvc.default.svc.cluster.local**

This way, when a pod is restarted, although its IP address will change :

- its name will **not** change
- its DNS endpoint will **not** change.

To sum up:

---

- mysql-0
  - Role: **Master**
  - Data: write / read
  - Storage: /data/vol/pv1
  - Endpoint DNS: mysql-0..mysvc.default.svc.cluster.local
- mysql-1
  - Role: **Slave**
  - Data: read only
  - Storage: /data/vol/pv2
  - DNS endpoint: mysql-1.mysvc.default.svc.cluster.local
- mysql-2
  - Role: **Slave**
  - Data: read only
  - Storage: /data/vol/pv3
  - DNS endpoint: mysql-2.mysvc.default.svc.cluster.local
- mysql-3
  - Role: **Slave**
  - Data: read only
  - Storage: /data/vol/pv4
  - DNS endpoint: mysql-3.mysvc.default.svc.cluster.local

Lastly, a StatefulSet is a **complicated** K8s component that is difficult to implement because Kubernetes does not handle certain tasks such as:

- the configuration of the data cloning process,
- the configuration of the data replication process,
- the creation and configuration of persistent and remote physical storage,
- the configuration and management of data backups.

## LAB #1 - Setting up a simple StatefulSet

Create a **quarkus** namespace and modify the **kubernetes-admin@kubernetes context** to use it by default:

```
root@kubemaster:~# kubectl create ns quarkus
```

```
namespace/quarkus created
```

```
root@kubemaster:~# kubectl config set-context --current --namespace=quarkus
Context "kubernetes-admin@kubernetes" modified.
```



**Important:** Quarkus is a complete native Java framework for Kubernetes, designed for Java Virtual Machines (JVMs) and native compilation, which optimizes Java specifically for containers to make it an efficient platform for serverless, cloud and Kubernetes environments.

If you'd like to observe the results of the following commands in real time, open a second terminal and enter the following command:

```
root@kubemaster:~# watch -n 1 "kubectl get pods -o wide | awk '{print \$1 \" \" \$2 \" \" \$3 \" \" \$5 \" \" \$7}' | column -t"
```

## 1.1 - Service and StatefulSet creation

Now create the **quarkus-service.yaml** file:



To do: Copy the content from [here](#) and paste it into your file.

```
root@kubemaster:~# vi quarkus-service.yaml
root@kubemaster:~# cat quarkus-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: quarkus
```

```
labels:
  app: quarkus-statefulset
spec:
  ports:
  - port: 8080
    name: web
  clusterIP: None
  selector:
    app: quarkus-statefulset
```



**Important:** Note the service name - **quarkus**. The **None** value of the **ClusterIP** entry makes the service **headless**. In this case, the DNS server will return the IP addresses of the individual pods instead of the IP address of the service. The client can then connect to any of them.

Create the service:

```
root@kubemaster:~# kubectl apply -f quarkus-service.yaml
service/quarkus created
```

Now create the **statefulset.yaml** file:



To do: Copy the content from [here](#) and paste it into your file.

```
root@kubemaster:~# vi statefulset.yaml
root@kubemaster:~# cat statefulset.yaml
apiVersion: apps/v1
kind: StatefulSet
```

```
metadata:
  name: quarkus-statefulset
  labels:
    app: quarkus-statefulset
spec:
  serviceName: "quarkus"
  replicas: 2
  template:
    metadata:
      labels:
        app: quarkus-statefulset
    spec:
      containers:
        - name: quarkus-statefulset
          image: quay.io/rhdevelopers/quarkus-demo:v1
          ports:
            - containerPort: 8080
              name: web
      selector:
        matchLabels:
          app: quarkus-statefulset
```



**Important:** Note that the value of **serviceName** is **quarkus**.

Create the StatefulSet :

```
root@kubemaster:~# kubectl apply -f statefulset.yaml
statefulset.apps/quarkus-statefulset created
```

Note the presence of the two pods in Namespace:

```
Every 1,0s: kubectl get pods -o wide | awk '{print $1 " " $2 " " $3 " " $5 " " $7}' | column -t
kubemaster.ittraining.loc: Tue Dec 6 17:43:50 2022
```

NAME	READY	STATUS	AGE	NODE
quarkus-statefulset-0	1/1	Running	2m17s	kubemaster.ittraining.loc
quarkus-statefulset-1	1/1	Running	106s	kubemaster.ittraining.loc

Check the StatefulSet's status:

```
root@kubemaster:~# kubectl get statefulsets
NAME                READY   AGE
quarkus-statefulset 2/2     3m35s
```

as well as the presence of the service:

```
root@kubemaster:~# kubectl get services
NAME    TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
quarkus ClusterIP   None         <none>        8080/TCP   12m
```

## 1.2 - Scaling Up a StatefulSet

Perform a scale up:

```
root@kubemaster:~# kubectl scale sts quarkus-statefulset --replicas=3
```



**Important:** Note that the short name of a **serviceName** is **sts**.

Note the presence of **three** pods in the Namespace:

```
Every 1,0s: kubectl get pods -o wide | awk '{print $1 " " $2 " " $3 " " $5 " " $7}' | column -t
kubemaster.ittraining.loc: Tue Dec 6 17:46:42 2022
```

NAME	READY	STATUS	AGE	NODE
quarkus-statefulset-0	1/1	Running	5m9s	kubenode2.ittraining.loc
quarkus-statefulset-1	1/1	Running	4m38s	kubenode1.ittraining.loc
quarkus-statefulset-2	1/1	Running	13s	kubenode2.ittraining.loc

Note the pod creation order:

```
root@kubemaster:~# kubectl get events --sort-by=.metadata.creationTimestamp
LAST SEEN   TYPE      REASON          OBJECT                                     MESSAGE
6m35s       Normal   SuccessfulCreate statefulset/quarkus-statefulset         create Pod quarkus-statefulset-0 in
StatefulSet quarkus-statefulset successful
6m35s       Normal   Scheduled       pod/quarkus-statefulset-0               Successfully assigned quarkus/quarkus-
statefulset-0 to kubenode2.ittraining.loc
6m34s       Normal   Pulling        pod/quarkus-statefulset-0               Pulling image
"quay.io/rhdevelopers/quarkus-demo:v1"
6m5s        Normal   Pulled         pod/quarkus-statefulset-0               Successfully pulled image
"quay.io/rhdevelopers/quarkus-demo:v1" in 28.871622372s
6m4s        Normal   Created        pod/quarkus-statefulset-0               Created container quarkus-statefulset
6m4s        Normal   Started        pod/quarkus-statefulset-0               Started container quarkus-statefulset
6m3s        Normal   Scheduled       pod/quarkus-statefulset-1               Successfully assigned quarkus/quarkus-
statefulset-1 to kubenode1.ittraining.loc
6m3s        Normal   SuccessfulCreate statefulset/quarkus-statefulset         create Pod quarkus-statefulset-1 in
StatefulSet quarkus-statefulset successful
5m58s       Normal   Pulling        pod/quarkus-statefulset-1               Pulling image
"quay.io/rhdevelopers/quarkus-demo:v1"
5m22s       Normal   Pulled         pod/quarkus-statefulset-1               Successfully pulled image
"quay.io/rhdevelopers/quarkus-demo:v1" in 35.551473165s
5m21s       Normal   Created        pod/quarkus-statefulset-1               Created container quarkus-statefulset
5m21s       Normal   Started        pod/quarkus-statefulset-1               Started container quarkus-statefulset
99s        Normal   Scheduled       pod/quarkus-statefulset-2               Successfully assigned quarkus/quarkus-
statefulset-2 to kubenode2.ittraining.loc
```

```
99s      Normal    SuccessfulCreate   statefulset/quarkus-statefulset   create Pod quarkus-statefulset-2 in
StatefulSet quarkus-statefulset successful
98s      Normal    Pulled             pod/quarkus-statefulset-2         Container image
"quay.io/rhdevelopers/quarkus-demo:v1" already present on machine
97s      Normal    Created            pod/quarkus-statefulset-2         Created container quarkus-statefulset
97s      Normal    Started            pod/quarkus-statefulset-2         Started container quarkus-statefulset
```

Now create a pod to query the K8s DNS:

```
root@kubemaster:~# kubectl run -it --restart=Never --rm --image busybox:1.28 dns-test
If you don't see a command prompt, try pressing enter.
/ # nslookup quarkus-statefulset-0.quarkus
Server:      10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:       quarkus-statefulset-0.quarkus
Address 1: 192.168.150.2 quarkus-statefulset-0.quarkus.quarkus.svc.cluster.local
/ # exit
pod "dns-test" deleted
root@kubemaster:~#
```

### 1.3 - Scaling Down a StatefulSet

Now perform a scale down:

```
root@kubemaster:~# kubectl scale sts quarkus-statefulset --replicas=2
statefulset.apps/quarkus-statefulset scaled
```

Note the presence of **two** pods in Namespace :

```
Every 1,0s: kubectl get pods -o wide | awk '{print $1 " " $2 " " $3 " " $5 " " $7}' | column -t
kubemaster.ittraining.loc: Tue Dec  6 18:02:27 2022
```

NAME	READY	STATUS	AGE	NODE
quarkus-statefulset-0	1/1	Running	20m	kubnode2.ittraining.loc
quarkus-statefulset-1	1/1	Running	20m	kubnode1.ittraining.loc

## 1.4 - Deleting the StatefulSet

Finally, delete the StatefulSet, the service and the Namespace :

```
root@kubemaster:~# kubectl delete -f statefulset.yaml
statefulset.apps "quarkus-statefulset" deleted

root@kubemaster:~# kubectl delete -f quarkus-service.yaml
service "quarkus-statefulset-2" deleted

root@kubemaster:~# kubectl config set-context --current --namespace=default
Context "kubernetes-admin@kubernetes" modified.
```

# Advanced StorageClass Usage

## LAB #2 - Dynamic NFS provisioning

### 2.1 - NFS Server Configuration

Connect to the CentOS8 VM as a trainee at 10.0.2.45.

Become root and create the **/srv/nfs/kubedata** directory:

```
[root@centos8 ~]# mkdir -p /srv/nfs/kubedata
```

Now continue by activating and starting the **nfs-server** service:

```
[root@centos8 ~]# systemctl status nfs-server
● nfs-server.service - NFS server and services
   Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; disabled; vendor prese>
   Active: inactive (dead)

[root@centos8 ~]# systemctl enable nfs-server.service
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-server.service → /usr/lib/systemd/system/nfs-server.service.

[root@centos8 ~]# systemctl start nfs-server.service

[root@centos8 ~]# systemctl status nfs-server.service
● nfs-server.service - NFS server and services
   Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; enabled; vendor preset: disabled)
   Active: active (exited) since Mon 2022-11-21 11:02:13 CET; 9s ago
     Process: 3276 ExecStart=/bin/sh -c if systemctl -q is-active gssproxy; then systemctl reload gssproxy ; fi
 (code=exited, >
     Process: 3263 ExecStart=/usr/sbin/rpc.nfsd (code=exited, status=0/SUCCESS)
     Process: 3261 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS)
    Main PID: 3276 (code=exited, status=0/SUCCESS)

Nov 21 11:02:12 centos8.ittraining.loc systemd[1]: Starting NFS server and services...
Nov 21 11:02:13 centos8.ittraining.loc systemd[1]: Started NFS server and services.
```

Edit the **/etc/exports** file:

```
[root@centos8 ~]# vi /etc/exports
[root@centos8 ~]# cat /etc/exports
/srv/nfs/kubedata *(rw,sync,no_subtree_check,no_root_squash,no_all_squash,insecure)
```





**Important:** In this case, we've shared the `/srv/nfs/kubedata` directory with the world.

Apply the export :

```
[root@centos8 ~]# exportfs -rav
exporting */srv/nfs/kubedata

[root@centos8 ~]# exportfs -v
/srv/nfs/kubedata
<world>(sync,wdelay,hide,no_subtree_check,sec=sys,rw,insecure,no_root_squash,no_all_squash)
```

Set SELinux to permissive mode:

```
[root@centos8 ~]# getenforce
Enforcing

[root@centos8 ~]# setenforce permissive
```

Next, configure the firewall:

```
[root@centos8 ~]# firewall-cmd --permanent --add-service=nfs
success
[root@centos8 ~]# firewall-cmd --permanent --add-service=rpc-bind
success
[root@centos8 ~]# firewall-cmd --permanent --add-service=mountd
success
[root@centos8 ~]# firewall-cmd --reload
success
```

## 2.2 - NFS Client Configuration

Return to your gateway and connect as user **trainee** to **kubenode2** at 192.168.56.4. Then become the root user:

```
trainee@kubenode2:~$ su -  
Password: fenestros  
root@kubenode2:~#
```

Install the **nfs-common** package:

```
root@kubenode2:~# apt update  
...  
root@kubenode2:~# apt install nfs-common  
...
```

Check that you can see the directory exported by 10.0.2.45 :

```
root@kubenode2:~# showmount --exports 10.0.2.45  
Export list for 10.0.2.45:  
/srv/nfs/kubedata *
```

Check that you can mount the directory exported by 10.0.2.45 :

```
root@kubenode2:~# mount -t nfs 10.0.2.45:/srv/nfs/kubedata /mnt  
root@kubenode2:~# mount | grep kubedata  
10.0.2.45:/srv/nfs/kubedata on /mnt type nfs4  
(rw,relatime,vers=4.2,rsiz=524288,wsiz=524288,namlen=255,hard,proto=tcp,port=0,timeo=600,retrans=2,sec=sys,clientaddr=10.0.2.67,local_lock=none,addr=10.0.2.45)
```

Then unmount **10.0.2.45:/srv/nfs/kubedata** :

```
root@kubenode2:~# umount /mnt
```

```
root@kubenode2:~# mount | grep kubedata
```

Connect to kubenode1 at 192.168.56.3 :

```
root@kubenode2:~# ssh -l trainee 192.168.56.3
The authenticity of host '192.168.56.3 (192.168.56.3)' can't be established.
ECDSA key fingerprint is SHA256:sEfHBv9azmK60cjQF/aJgUc9jg56slNaZQdAUcvB0vE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.56.3' (ECDSA) to the list of known hosts.
trainee@192.168.56.3's password: trainee
Linux kubenode1.ittraining.loc 4.9.0-19-amd64 #1 SMP Debian 4.9.320-2 (2022-06-30) x86_64
```

```
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Sep 28 09:54:21 2022 from 192.168.56.2
```

```
trainee@kubenode1:~$ su -
Password: fenestros
root@kubenode1:~#
```

Then install the **nfs-common** package:

```
root@kubenode1:~# apt update
...

root@kubenode1:~# apt install nfs-common
...
```

Return to your gateway :

---

```
root@kubenode1:~# exit
logout

trainee@kubenode1:~$ exit
logout
Connection to 192.168.56.3 closed.

root@kubenode2:~# exit
logout

trainee@kubenode2:~$ exit
logout
Connection to 192.168.56.4 closed.
```

## 2.3 - Configuring K8s

Connect to your **kubemaster** at 192.168.56.2.

Then install the **nfs-common** package:

```
root@kubemaster:~# apt update
...

root@kubemaster:~# apt install nfs-common
...
```

Add the **nfs-subdir-external-provisioner** repository to **helm** :

```
root@kubemaster:~# helm repo add nfs-subdir-external-provisioner
https://kubernetes-sigs.github.io/nfs-subdir-external-provisioner/
"nfs-subdir-external-provisioner" has been added to your repositories
```

Install the helm **nfs-subdir-external-provisioner** Chart:

```
root@kubemaster:~# helm install nfs-subdir-external-provisioner nfs-subdir-external-provisioner/nfs-subdir-external-provisioner --set nfs.server=10.0.2.45 --set nfs.path=/srv/nfs/kubedata
NAME: nfs-subdir-external-provisioner
LAST DEPLOYED: Wed Dec 7 11:12:23 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Check the status of the created pod:

```
root@kubemaster:~# kubectl get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP
netshoot	1/1	Running	3 (25h ago)	70d	192.168.239.58
kubemaster1.ittraining.loc	<none>	<none>			
nfs-subdir-external-provisioner-59b4b5c476-wxkp4	0/1	ContainerCreating	0	19m	<none>
kubemaster1.ittraining.loc	<none>	<none>			
nginx-netshoot	1/1	Running	3 (25h ago)	70d	192.168.239.59
kubemaster1.ittraining.loc	<none>	<none>			
postgresql-6f885d8957-tnlbb	1/1	Running	3 (25h ago)	70d	192.168.239.62
kubemaster1.ittraining.loc	<none>	<none>			
sharedvolume	2/2	Running	6 (25h ago)	78d	192.168.150.60
kubemaster2.ittraining.loc	<none>	<none>			
troubleshooting	1/1	Running	3 (25h ago)	70d	192.168.239.60
kubemaster1.ittraining.loc	<none>	<none>			
volumeopod	0/1	Completed	0	78d	192.168.150.41
kubemaster2.ittraining.loc	<none>	<none>			



**Important:** If the **nfs-subdir-external-provisioner-yyyyyyyyy-xxxxx** pod remains in a



**ContainerCreating** state for more than 5 minutes, remove the three **calico-node-xxxxx** pods from the **kube-system** namespace and wait for them to be recreated.

Once recreated, you can see that the **nfs-subdir-external-provisioner-yyyyyyyy-xxxxx** pod is in a Running state:

```
root@kubemaster:~# kubectl get pods -o wide
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
default	netshoot	1/1	Running	3 (25h ago)	70d
192.168.239.58	kubenode1.ittraining.loc	<none>	<none>		
default	nfs-subdir-external-provisioner-59b4b5c476-wxkp4	1/1	Running	1 (3m18s ago)	36m
192.168.239.63	kubenode1.ittraining.loc	<none>	<none>		
default	nginx-netshoot	1/1	Running	3 (25h ago)	70d
192.168.239.59	kubenode1.ittraining.loc	<none>	<none>		
default	postgresql-6f885d8957-tnlbb	1/1	Running	3 (25h ago)	70d
192.168.239.62	kubenode1.ittraining.loc	<none>	<none>		
default	sharedvolume	2/2	Running	6 (25h ago)	78d
192.168.150.60	kubenode2.ittraining.loc	<none>	<none>		
default	troubleshooting	1/1	Running	3 (25h ago)	70d
192.168.239.60	kubenode1.ittraining.loc	<none>	<none>		
default	volumepod	0/1	Completed	0	78d
192.168.150.41	kubenode2.ittraining.loc	<none>	<none>		

Examination of the pod log **nfs-subdir-external-provisioner-yyyyyyyy-xxxxx** shows that everything works:

```
root@kubemaster:~# kubectl logs nfs-subdir-external-provisioner-59b4b5c476-wxkp4
I1207 10:45:38.321263      1 leaderelection.go:242] attempting to acquire leader lease default/cluster.local-nfs-subdir-external-provisioner...
I1207 10:45:59.097918      1 leaderelection.go:252] successfully acquired lease default/cluster.local-nfs-subdir-external-provisioner
I1207 10:45:59.097979      1 event.go:278] Event(v1.ObjectReference{Kind:"Endpoints", Namespace:"default", Name:"cluster.local-nfs-subdir-external-provisioner", UID:"986e4938-a054-4bf9-bfdd-903749c7f63f", APIVersion:"v1", ResourceVersion:"6690493", FieldPath:""}): type: 'Normal' reason: 'LeaderElection' nfs-subdir-
```

```
external-provisioner-59b4b5c476-wxkp4_1d17de3a-ac5b-442c-aa63-8253d33c2857 became leader
I1207 10:45:59.098098      1 controller.go:820] Starting provisioner controller cluster.local/nfs-subdir-
external-provisioner_nfs-subdir-external-provisioner-59b4b5c476-wxkp4_1d17de3a-ac5b-442c-aa63-8253d33c2857!
I1207 10:45:59.198332      1 controller.go:869] Started provisioner controller cluster.local/nfs-subdir-
external-provisioner_nfs-subdir-external-provisioner-59b4b5c476-wxkp4_1d17de3a-ac5b-442c-aa63-8253d33c2857!
```

Now consult the list of StorageClasses available:

```
root@kubemaster:~# kubectl get storageclass
NAME          PROVISIONER          RECLAIMPOLICY  VOLUMEBINDINGMODE  ALLOWVOLUMEEXPANSION  AGE
localdisk     kubernetes.io/no-provisioner  Delete         Immediate           true                  77d
nfs-client    cluster.local/nfs-subdir-external-provisioner  Delete         Immediate           true                  52m
```

## 2.4 - Creating a PersistentVolumeClaim

Now create the file **pvc.yaml**:



To do: Copy the content from [here](#) and paste it into your file.

```
root@kubemaster:~# vi pvc.yaml
root@kubemaster:~# cat pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvcl
spec:
  storageClassName: nfs-client
```

```
accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 500Mi
```

Apply the pvc.yaml file:

```
root@kubemaster:~# kubectl apply -f pvc.yaml
persistentvolumeclaim/pvc1 created
```

Now look at the list of PersistentVolumes and PersistentVolumeClaims:

```
root@kubemaster:~# kubectl get pv,pvc
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS
CLAIM				
STORAGECLASS				
REASON				
AGE				
persistentvolume/mypv	1Gi	RWO	Recycle	Available
localdisk				
			77d	
persistentvolume/pvc-721f5ed3-88b1-41bb-82c2-9eab3b4464da	500Mi	RWX	Delete	Bound
default/pvc1				
nfs-client				
				66s

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES
STORAGECLASS				
AGE				
persistentvolumeclaim/pvc1	Bound	pvc-721f5ed3-88b1-41bb-82c2-9eab3b4464da	500Mi	RWX
client				
				67s
				nfs-



**Important:** Note that the PersistentVolume **persistentvolume/pvc-721f5ed3-88b1-41bb-82c2-9eab3b4464da** has been created automatically.

Connect to the NFS server and view the contents of the **/srv/nfs/kubedata** directory:

```
root@kubemaster:~# ssh -l trainee 10.0.2.45
The authenticity of host '10.0.2.45 (10.0.2.45)' can't be established.
ECDSA key fingerprint is SHA256:Q7T/CP0SLiMbMAIgVzTuEHegYS/spPE5zzQchCHD5Vw.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.45' (ECDSA) to the list of known hosts.
trainee@10.0.2.45's password: trainee
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Wed Dec 7 10:34:25 2022 from 10.0.2.65

[trainee@centos8 ~]$ ls -l /srv/nfs/kubedata/
total 0
drwxrwxrwx. 2 root root 6 Dec 7 12:32 default-pvc1-pvc-721f5ed3-88b1-41bb-82c2-9eab3b4464da

[trainee@centos8 ~]$ exit
logout
Connection to 10.0.2.45 closed.
```

## 2.5 - Using the PersistentVolumeClaim with a Pod

Now create the file **nfs-busybox.yaml**:



To do: Copy the content from [here](#) and paste it into your file.

```
root@kubemaster:~# vi nfs-busybox.yaml
root@kubemaster:~# cat nfs-busybox.yaml
apiVersion: v1
kind: Pod
metadata:
```

```
name: nfs-pv-pod
spec:
  restartPolicy: Never
  containers:
  - name: busybox
    image: busybox
    command: ['sh', '-c', 'while true; do sleep 3600; done']
    volumeMounts:
    - name: pv-storage
      mountPath: /pv-pod-storage
  volumes:
  - name: pv-storage
    persistentVolumeClaim:
      claimName: pvcl
```

Apply the nfs-busybox.yaml file:

```
root@kubemaster:~# kubectl apply -f nfs-busybox.yaml
pod/nfs-pv-pod created
```

Check that pod status **nfs-pv-pod** is **Running** :

```
root@kubemaster:~# kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
netshoot	1/1	Running	3 (26h ago)	70d
nfs-pv-pod	1/1	Running	0	2m9s
nfs-subdir-external-provisioner-59b4b5c476-wxkp4	1/1	Running	1 (80m ago)	113m
nginx-netshoot	1/1	Running	3 (26h ago)	70d
postgresql-6f885d8957-tnlbb	1/1	Running	3 (26h ago)	70d
sharedvolume	2/2	Running	6 (26h ago)	78d
troubleshooting	1/1	Running	3 (26h ago)	70d
volumePod	0/1	Completed	0	78d

Connect to the **nfs-pv-pod** pod container:

```
root@kubemaster:~# kubectl exec -it nfs-pv-pod -- sh
/ #
```

Create the **hello** file in the **pv-pod-storage** directory:

```
root@kubemaster:~# kubectl exec -it nfs-pv-pod -- sh
/ # ls
bin          dev          etc          home         lib          lib64        proc
pv-pod-storage  root        sys         tmp          usr          var
/ # touch /pv-pod-storage/hello
/ # ls /pv-pod-storage/
hello
/ # exit
```

Connect to the NFS server and check the contents of the **/srv/nfs/kubedata** directory:

```
root@kubemaster:~# ssh -l trainee 10.0.2.45
trainee@10.0.2.45's password: trainee
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Wed Dec 7 12:37:00 2022 from 10.0.2.65

[trainee@centos8 ~]$ ls -lR /srv/nfs/kubedata/
/srv/nfs/kubedata/:
total 0
drwxrwxrwx. 2 root root 19 Dec 7 13:13 default-pvc1-pvc-721f5ed3-88b1-41bb-82c2-9eab3b4464da

/srv/nfs/kubedata/default-pvc1-pvc-721f5ed3-88b1-41bb-82c2-9eab3b4464da:
total 0
-rw-r--r--. 1 root root 0 Dec 7 13:13 hello

[trainee@centos8 ~]$ exit
logout
```

Connection to 10.0.2.45 closed.



**Important:** Note the presence of the **hello** file.

## 2.6 - Creating a Second PersistentVolumeClaim

Create the **pvc2.yaml** file:



To do: Copy the content from [here](#) and paste it into your file.

```
root@kubemaster:~# vi pvc2.yaml
root@kubemaster:~# cat pvc2.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc2
spec:
  storageClassName: nfs-client
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
```

Apply the **pvc2.yaml** file:

```
root@kubemaster:~# kubectl apply -f pvc2.yaml
```

```
persistentvolumeclaim/pvc2 created
```

Now consult the list of PersistentVolumes and PersistentVolumeClaims:

```
root@kubemaster:~# kubectl get pv,pvc
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS
CLAIM				
STORAGECLASS				
REASON				
AGE				
persistentvolume/myvp	1Gi	RWO	Recycle	Available
localdisk				77d
persistentvolume/pvc-6dbce6de-e473-4e4c-99be-0fbea26576de	100Mi	RWO	Delete	Bound
default/pvc2				nfs-client 58s
persistentvolume/pvc-721f5ed3-88b1-41bb-82c2-9eab3b4464da	500Mi	RWX	Delete	Bound
default/pvc1				nfs-client 53m

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES
STORAGECLASS				
AGE				
persistentvolumeclaim/pvc1	Bound	pvc-721f5ed3-88b1-41bb-82c2-9eab3b4464da	500Mi	RWX
client				nfs-53m
persistentvolumeclaim/pvc2	Bound	pvc-6dbce6de-e473-4e4c-99be-0fbea26576de	100Mi	RWO
client				nfs-58s



**Important:** Note that the PersistentVolume **persistentvolume/pvc-6dbce6de-e473-4e4c-99be-0fbea26576de** has been created automatically.

## 2.7 - Deleting the PersistentVolumeClaims

Start by deleting the **nfs-pv-pod** pod:

```
root@kubemaster:~# kubectl delete pod nfs-pv-pod
```

pod "nfs-pv-pod" deleted

Note that the pod has been deleted:

```
root@kubemaster:~# kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
netshoot	1/1	Running	3 (27h ago)	70d
nfs-subdir-external-provisioner-59b4b5c476-wxkp4	1/1	Running	1 (126m ago)	159m
nginx-netshoot	1/1	Running	3 (27h ago)	70d
postgresql-6f885d8957-tnlbb	1/1	Running	3 (27h ago)	70d
sharedvolume	2/2	Running	6 (27h ago)	78d
troubleshooting	1/1	Running	3 (27h ago)	70d
volume pod	0/1	Completed	0	78d

Now check the list of PersistentVolumes and PersistentVolumeClaims:

```
root@kubemaster:~# kubectl get pv,pvc
```

NAME	CLAIM	STORAGECLASS	REASON	AGE	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS
persistentvolume/mypv	localdisk			77d	1Gi	RWO	Recycle	Available
persistentvolume/pvc-6dbce6de-e473-4e4c-99be-0fbea26576de	default/pvc2	nfs-client		27m	100Mi	RWO	Delete	Bound
persistentvolume/pvc-721f5ed3-88b1-41bb-82c2-9eab3b4464da	default/pvc1	nfs-client		79m	500Mi	RWX	Delete	Bound

NAME	STORAGECLASS	AGE	STATUS	VOLUME	CAPACITY	ACCESS MODES	
persistentvolumeclaim/pvc1	client	79m	Bound	pvc-721f5ed3-88b1-41bb-82c2-9eab3b4464da	500Mi	RWX	nfs-
persistentvolumeclaim/pvc2	client	27m	Bound	pvc-6dbce6de-e473-4e4c-99be-0fbea26576de	100Mi	RWO	nfs-



**Important:** Note that the PersistentVolumes and the PersistentVolumeClaims are still present.

Delete both PersistentVolumeClaims:

```
root@kubemaster:~# kubectl delete pvc --all
persistentvolumeclaim "pvc1" deleted
persistentvolumeclaim "pvc2" deleted
```

Now you can see that both PersistentVolumes have been deleted automatically:

```
root@kubemaster:~# kubectl get pv,pvc
NAME                                CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM  STORAGECLASS  REASON
AGE
persistentvolume/mypv              1Gi       RW0           Recycle         Available  localdisk
77d
```

Connect to the NFS server and view the contents of the **/srv/nfs/kubedata** directory:

```
root@kubemaster:~# ssh -l trainee 10.0.2.45
trainee@10.0.2.45's password: trainee
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Wed Dec 7 13:15:49 2022 from 10.0.2.65

[trainee@centos8 ~]$ ls -lR /srv/nfs/kubedata/
/srv/nfs/kubedata/:
total 0
drwxrwxrwx. 2 root root 19 Dec 7 13:13 archived-default-pvc1-pvc-721f5ed3-88b1-41bb-82c2-9eab3b4464da
drwxrwxrwx. 2 root root 6 Dec 7 13:24 archived-default-pvc2-pvc-6dbce6de-e473-4e4c-99be-0fbea26576de
```

```
/srv/nfs/kubedata/archived-default-pvc1-pvc-721f5ed3-88b1-41bb-82c2-9eab3b4464da:  
total 0  
-rw-r--r--. 1 root root 0 Dec 7 13:13 hello  
  
/srv/nfs/kubedata/archived-default-pvc2-pvc-6dbce6de-e473-4e4c-99be-0fbea26576de:  
total 0  
  
[trainee@centos8 ~]$ exit  
logout  
Connection to 10.0.2.45 closed.
```



**Important:** Note that directories have a **archived-** prefix.

## Creating a Helm Chart

### Overview

A chart is a collection of files and directories that take the following form:

```
MyChart/  
  Chart.yaml  
  LICENSE  
  README.md  
  values.yaml  
  values.schema.json  
  charts/  
  crds/  
  templates/
```

templates/NOTES.txt

The helm template language is based on the GO language.

In the following LAB, you'll take the following two manifests, **ghost.yaml** and **ghost-service.yaml** and create a chart helm to install **Ghost**, a free blogging platform licensed under open source:



To do: Copy the content from [here](#) and paste it into your file.

```
root@kubemaster:~# vi ghost.yaml
root@kubemaster:~# cat ghost.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: blog
  labels:
    app: blog
spec:
  replicas: 1
  selector:
    matchLabels:
      app: blog
  template:
    metadata:
      labels:
        app: blog
    spec:
      containers:
      - name: blog
        image: ghost:2.6-alpine
        imagePullPolicy: Always
```

```
ports:
- containerPort: 2368
env:
- name: url
  value: http://exampleblog.com
```



To do: Copy the content from [here](#) and paste it into your file.

```
root@kubemaster:~# vi ghost-service.yaml
root@kubemaster:~# cat ghost-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: blog
spec:
  type: NodePort
  selector:
    app: blog
  ports:
  - protocol: TCP
    port: 80
    targetPort: 2368
```

### LAB #3 - Creating a Simple Helm Package

Start by creating the `~/ghost` directory and cd into it:

```
root@kubemaster:~# mkdir ghost
```

```
root@kubemaster:~# cd ghost
```

A chart requires a file named **Chart.yaml** to describe the chart in question. Create this file :

```
root@kubemaster:~/ghost# touch Chart.yaml
```

### 3.1 - The values.yaml File

A chart also needs a file called **values.yaml** which contains configuration values for the chart in question. Create a values.yaml file with the following contents:



To do: Copy the content from [here](#) and paste it into your file.

```
root@kubemaster:~/ghost# vi values.yaml
root@kubemaster:~/ghost# cat values.yaml
service:
  name: blog
  type: NodePort
  app: blog
  protocol: TCP
  port: 80
  targetPort: 2368
```

### 3.2 - Templates

Create the **templates** subdirectory in **ghost** :

```
root@kubemaster:~/ghost# mkdir templates
```

Copy the contents of `~/ghost-service.yaml` and paste it into `~/ghost/templates/service.yaml`:



To do: Copy the content from [here](#) and paste it into your file.

```
root@kubemaster:~/ghost# vi templates/service.yaml
root@kubemaster:~/ghost# cat templates/service.yaml
apiVersion: v1
kind: Service
metadata:
  name: blog
spec:
  type: NodePort
  selector:
    app: blog
  ports:
    - protocol: TCP
      port: 80
      targetPort: 2368
```

Then modify this file to read the values from the `values.yaml` file:

```
root@kubemaster:~/ghost# vi templates/service.yaml
root@kubemaster:~/ghost# cat templates/service.yaml
apiVersion: v1
kind: Service
metadata:
  name: {{ .Values.service.name }}
spec:
  type: {{ .Values.service.type }}
  selector:
    app: {{ .Values.service.app }}
```

```
ports:
- protocol: {{ .Values.service.protocol }}
  port: {{ .Values.service.port }}
  targetPort: {{ .Values.service.targetPort }}
```

Navigate to the parent directory of the **ghost** directory:

```
root@kubemaster:~/ghost# cd ..
```

Check that helm can read the list of values in the values.yaml file:

```
root@kubemaster:~# helm show values ghost
Error: validation: chart.metadata.name is required
```

The error refers to the **Chart.yaml** file, which is currently empty. Edit this file:

```
root@kubemaster:~# vi ghost/Chart.yaml
root@kubemaster:~# cat ghost/Chart.yaml
name: ghost
version: 1
```

Now check that helm can read the list of values in the values.yaml file:

```
root@kubemaster:~# helm show values ghost
service:
  name: blog
  type: NodePort
  app: blog
  protocol: TCP
  port: 80
  targetPort: 2368
```

Now check that the **service.yaml** manifest created by Helm is correct:

---

```
root@kubemaster:~# helm install check ghost --dry-run
NAME: check
LAST DEPLOYED: Thu Dec  8 15:54:13 2022
NAMESPACE: default
STATUS: pending-install
REVISION: 1
TEST SUITE: None
HOOKS:
MANIFEST:
---
# Source: ghost/templates/service.yaml
apiVersion: v1
kind: Service
metadata:
  name: blog
spec:
  type: NodePort
  selector:
    app: blog
  ports:
  - protocol: TCP
    port: 80
    targetPort: 2368
```

Now copy the contents of the `~/ghost.yaml` file and paste it into the `~/ghost/templates/ghost.yaml` file:

```
root@kubemaster:~# vi ghost/templates/ghost.yaml
root@kubemaster:~# cat ghost/templates/ghost.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: blog
  labels:
    app: blog
```

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: blog
  template:
    metadata:
      labels:
        app: blog
    spec:
      containers:
      - name: blog
        image: ghost:2.6-alpine
        imagePullPolicy: Always
        ports:
        - containerPort: 2368
        env:
        - name: url
          value: http://exampleblog.com
```

Then modify this file to read the values from the **values.yaml** file:

```
root@kubemaster:~# vi ghost/templates/ghost.yaml
root@kubemaster:~# cat ghost/templates/ghost.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Values.blog.name }}
  labels:
    app: {{ .Values.blog.label }}
spec:
  replicas: {{ .Values.blog.replicas }}
  selector:
    matchLabels:
```

```
  app: {{ .Values.blog.name }}
template:
  metadata:
    labels:
      app: {{ .Values.blog.name }}
  spec:
    containers:
      - name: {{ .Values.blog.name }}
        image: {{ .Values.blog.image }}
        imagePullPolicy: {{ .Values.blog.imagePullPolicy }}
        ports:
          - containerPort: {{ .Values.blog.containerPort }}
        env:
          - name: {{ .Values.blog.url }}
            value: {{ .Values.blog.urlValue }}
```

Now complete the contents of the values.yaml file:



To do: Copy the content from [here](#) and paste it into your file.

```
root@kubemaster:~# vi ghost/values.yaml
root@kubemaster:~# cat ghost/values.yaml
service:
  name: blog
  type: NodePort
  app: blog
  protocol: TCP
  port: 80
  targetPort: 2368
blog:
  name: blog
```

```
label: blog
replicas: 1
image: ghost:2.6-alpine
imagePullPolicy: Always
containerPort: 2368
url: url
urlValue: http://exampleblog.com
```

Now check that the **ghost.yaml** manifest created by Helm is correct:

```
root@kubemaster:~# helm install check ghost --dry-run
NAME: check
LAST DEPLOYED: Thu Dec  8 16:12:29 2022
NAMESPACE: default
STATUS: pending-install
REVISION: 1
TEST SUITE: None
HOOKS:
MANIFEST:
---
# Source: ghost/templates/service.yaml
apiVersion: v1
kind: Service
metadata:
  name: blog
spec:
  type: NodePort
  selector:
    app: blog
  ports:
  - protocol: TCP
    port: 80
    targetPort: 2368
---
```

```
# Source: ghost/templates/ghost.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: blog
  labels:
    app: blog
spec:
  replicas: 1
  selector:
    matchLabels:
      app: blog
  template:
    metadata:
      labels:
        app: blog
    spec:
      containers:
        - name: blog
          image: ghost:2.6-alpine
          imagePullPolicy: Always
          ports:
            - containerPort: 2368
          env:
            - name: url
              value: http://exampleblog.com
```

See now the organization of the **ghost** chart:

```
root@kubemaster:~# tree ghost
ghost
├── Chart.yaml
├── templates
│   └── ghost.yaml
```

```
|  └─ service.yaml
|  └─ values.yaml
```

```
1 directory, 4 files
```

### 3.3 - Installation and Removal

Install the **ghost** chart:

```
root@kubemaster:~# helm install live ghost
NAME: live
LAST DEPLOYED: Thu Dec 8 16:14:13 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Check the status of the service in the cluster:

```
root@kubemaster:~# kubectl get svc
NAME                TYPE          CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
blog                 NodePort     10.106.215.169  <none>        80:32070/TCP    52s
kubernetes           ClusterIP    10.96.0.1       <none>        443/TCP         95d
service-netshoot    ClusterIP    10.107.115.28  <none>        80/TCP          71d
```

Check the presence of the pod in the cluster:

```
root@kubemaster:~# kubectl get po
NAME                                READY   STATUS    RESTARTS   AGE
blog-8545df8764-hk8rc              1/1    Running   0          105s
netshoot                           1/1    Running   3 (2d6h ago)  71d
nfs-subdir-external-provisioner-59b4b5c476-wxkp4  1/1    Running   1 (28h ago)  29h
```

nginx-netshoot	1/1	Running	3 (2d6h ago)	71d
postgresql-6f885d8957-tnlbb	1/1	Running	3 (2d6h ago)	71d
sharedvolume	2/2	Running	6 (2d6h ago)	79d
troubleshooting	1/1	Running	3 (2d6h ago)	71d
volumepod	0/1	Completed	0	

Check the Chart status :

```
root@kubemaster:~# helm status live
NAME: live
LAST DEPLOYED: Thu Dec 8 16:14:13 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Lastly, delete the chart :

```
root@kubemaster:~# helm delete live
release "live" uninstalled
```

## Monitoring

### Overview

The Prometheus server consists of three modules:

- **Data Retrieval Worker**, which retrieves metrics
- **Time Series Database** for metrics storage
- **HTTP Server** which accepts PromQL requests and provides a Web UI for data consultation



**Important** : **PromQL**, short for Prometheus Querying Language, is the main way of querying metrics in Prometheus. You can display the return of an expression as a graph, or export it using the HTTP API. PromQL uses three types of data: scalars, range vectors and snapshot vectors. It also uses strings, but only as literals.

Alerts are then passed to the **Alertmanager**, which informs the people in charge of the configuration.

## LAB #4 - Implementing a Prometheus Solution

Connect to the VM **Gateway\_10.0.2.40\_VNC**.

### 4.1 - Stack Deployment with Helm

Add the **prometheus-community** repository:

```
trainee@gateway:~$ helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
"prometheus-community" has been added to your repositories

trainee@gateway:~$ helm repo update
```

Then install the **kube-prometheus-stack** Chart:

```
trainee@gateway:~$ helm install prometheus prometheus-community/kube-prometheus-stack
NAME: prometheus
LAST DEPLOYED: Thu Dec  8 17:04:17 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
```

**NOTES:**

kube-prometheus-stack has been installed. Check its status by running:

```
kubectl --namespace default get pods -l "release=prometheus"
```

Visit <https://github.com/prometheus-operator/kube-prometheus> for instructions on how to create & configure Alertmanager and Prometheus instances using the Operator.

Wait until all pods are in a **Running** state:

```
trainee@gateway:~$ kubectl --namespace default get pods -l "release=prometheus"
```

NAME	READY	STATUS	RESTARTS	AGE
prometheus-kube-prometheus-operator-689dd6679c-2th6f	1/1	Running	0	4m12s
prometheus-kube-state-metrics-6cfd96f4c8-wrw2n	1/1	Running	0	4m12s
prometheus-prometheus-node-exporter-8cb4s	1/1	Running	0	4m13s
prometheus-prometheus-node-exporter-ll4qp	1/1	Running	0	4m13s
prometheus-prometheus-node-exporter-x87f7	1/1	Running	0	4m13s

Now look at all the Prometheus objects created by the installation:

```
trainee@gateway:~$ kubectl get all -l "release=prometheus"
```

NAME	READY	STATUS	RESTARTS	AGE
pod/prometheus-kube-prometheus-operator-689dd6679c-2th6f	1/1	Running	0	13h
pod/prometheus-kube-state-metrics-6cfd96f4c8-wrw2n	1/1	Running	0	13h
pod/prometheus-prometheus-node-exporter-8cb4s	1/1	Running	0	13h
pod/prometheus-prometheus-node-exporter-ll4qp	1/1	Running	0	13h
pod/prometheus-prometheus-node-exporter-x87f7	1/1	Running	0	13h

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/prometheus-kube-prometheus-alertmanager	ClusterIP	10.103.114.236	<none>	9093/TCP	13h
service/prometheus-kube-prometheus-operator	ClusterIP	10.107.174.218	<none>	443/TCP	13h
service/prometheus-kube-prometheus-prometheus	ClusterIP	10.108.124.100	<none>	9090/TCP	13h
service/prometheus-kube-state-metrics	ClusterIP	10.109.13.26	<none>	8080/TCP	13h
service/prometheus-prometheus-node-exporter	ClusterIP	10.103.100.124	<none>	9100/TCP	13h

NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AVAILABLE	NODE
SELECTOR AGE						
daemonset.apps/prometheus-prometheus-node-exporter 13h	3	3	3	3	3	<none>
NAME	READY	UP-TO-DATE	AVAILABLE	AGE		
deployment.apps/prometheus-kube-prometheus-operator	1/1	1	1	13h		
deployment.apps/prometheus-kube-state-metrics	1/1	1	1	13h		
NAME	DESIRED	CURRENT	READY	AGE		
replicaset.apps/prometheus-kube-prometheus-operator-689dd6679c	1	1	1	13h		
replicaset.apps/prometheus-kube-state-metrics-6cfd96f4c8	1	1	1	13h		
NAME			READY	AGE		
statefulset.apps/alertmanager-prometheus-kube-prometheus-alertmanager			1/1	13h		
statefulset.apps/prometheus-prometheus-kube-prometheus-prometheus			1/1	13h		

In this output we see:

- 2 StatefulSets including:
  - the Prometheus server **statefulset.apps/prometheus-prometheus-kube-prometheus-prometheus**
  - the Alertmanager **statefulset.apps/alertmanager-prometheus-kube-prometheus-alertmanager**.
- 2 Deployments including:
  - the **deployment.apps/prometheus-kube-prometheus-operator** which created the two StatefulSets
  - the kube-state-metrics **deployment.apps/prometheus-kube-state-metrics** which is a dependency of Prometheus and therefore a **Subchart** of the latter
- 2 ReplicaSets created by Deployments:
  - **replicaset.apps/prometheus-kube-prometheus-operator-689dd6679c**
  - **replicaset.apps/prometheus-kube-state-metrics-6cfd96f4c8**
- 1 DaemonSet **daemonset.apps/prometheus-prometheus-node-exporter** :
  - the pods in this DaemonSet are responsible for transforming node metrics into Prometheus metrics.

The installation also created a large number of ConfigMaps :

```
trainee@gateway:~$ kubectl get configmap -l "release=prometheus"
```

NAME	DATA	AGE
prometheus-kube-prometheus-alertmanager-overview	1	13h
prometheus-kube-prometheus-apiserver	1	13h
prometheus-kube-prometheus-cluster-total	1	13h
prometheus-kube-prometheus-controller-manager	1	13h
prometheus-kube-prometheus-etcd	1	13h
prometheus-kube-prometheus-grafana-datasource	1	13h
prometheus-kube-prometheus-grafana-overview	1	13h
prometheus-kube-prometheus-k8s-coredns	1	13h
prometheus-kube-prometheus-k8s-resources-cluster	1	13h
prometheus-kube-prometheus-k8s-resources-namespace	1	13h
prometheus-kube-prometheus-k8s-resources-node	1	13h
prometheus-kube-prometheus-k8s-resources-pod	1	13h
prometheus-kube-prometheus-k8s-resources-workload	1	13h
prometheus-kube-prometheus-k8s-resources-workloads-namespace	1	13h
prometheus-kube-prometheus-kubelet	1	13h
prometheus-kube-prometheus-namespace-by-pod	1	13h
prometheus-kube-prometheus-namespace-by-workload	1	13h
prometheus-kube-prometheus-node-cluster-rsrc-use	1	13h
prometheus-kube-prometheus-node-rsrc-use	1	13h
prometheus-kube-prometheus-nodes	1	13h
prometheus-kube-prometheus-nodes-darwin	1	13h
prometheus-kube-prometheus-persistentvolumesusage	1	13h
prometheus-kube-prometheus-pod-total	1	13h
prometheus-kube-prometheus-prometheus	1	13h
prometheus-kube-prometheus-proxy	1	13h
prometheus-kube-prometheus-scheduler	1	13h
prometheus-kube-prometheus-workload-total	1	13h

as well as Secrets :

```
trainee@gateway:~$ kubectl get secrets
```

NAME	TYPE	DATA	AGE
------	------	------	-----

alertmanager-prometheus-kube-prometheus-alertmanager	Opaque	1	13h
alertmanager-prometheus-kube-prometheus-alertmanager-generated	Opaque	1	13h
alertmanager-prometheus-kube-prometheus-alertmanager-tls-assets-0	Opaque	0	13h
alertmanager-prometheus-kube-prometheus-alertmanager-web-config	Opaque	1	13h
my-secret	Opaque	2	88d
prometheus-grafana	Opaque	3	13h
prometheus-kube-prometheus-admission	Opaque	3	13h
prometheus-prometheus-kube-prometheus-prometheus	Opaque	1	13h
prometheus-prometheus-kube-prometheus-prometheus-tls-assets-0	Opaque	1	13h
prometheus-prometheus-kube-prometheus-prometheus-web-config	Opaque	1	13h
sh.helm.release.v1.nfs-subdir-external-provisioner.v1	helm.sh/release.v1	1	43h
sh.helm.release.v1.prometheus.v1	helm.sh/release.v1	1	13h

and some **Custom Resource Definitions** or crd:

```
trainee@gateway:~$ kubectl get crd
NAME                                     CREATED AT
alertmanagerconfigs.monitoring.coreos.com 2022-12-08T16:04:14Z
alertmanagers.monitoring.coreos.com      2022-12-08T16:04:14Z
bgpconfigurations.crd.projectcalico.org   2022-09-04T07:38:47Z
bgppeers.crd.projectcalico.org           2022-09-04T07:38:47Z
blockaffinities.crd.projectcalico.org     2022-09-04T07:38:48Z
caliconodestatuses.crd.projectcalico.org  2022-09-04T07:38:48Z
clusterinformations.crd.projectcalico.org  2022-09-04T07:38:48Z
felixconfigurations.crd.projectcalico.org  2022-09-04T07:38:48Z
globalnetworkpolicies.crd.projectcalico.org 2022-09-04T07:38:48Z
globalnetworksets.crd.projectcalico.org    2022-09-04T07:38:49Z
hostendpoints.crd.projectcalico.org        2022-09-04T07:38:49Z
ipamblocks.crd.projectcalico.org           2022-09-04T07:38:49Z
ipamconfigs.crd.projectcalico.org          2022-09-04T07:38:49Z
ipamhandles.crd.projectcalico.org          2022-09-04T07:38:50Z
ippools.crd.projectcalico.org              2022-09-04T07:38:50Z
ipreservations.crd.projectcalico.org       2022-09-04T07:38:50Z
kubecontrollersconfigurations.crd.projectcalico.org 2022-09-04T07:38:50Z
```

```

networkpolicies.crd.projectcalico.org      2022-09-04T07:38:50Z
networksets.crd.projectcalico.org         2022-09-04T07:38:50Z
podmonitors.monitoring.coreos.com         2022-12-08T16:04:14Z
probes.monitoring.coreos.com              2022-12-08T16:04:14Z
prometheuses.monitoring.coreos.com        2022-12-08T16:04:14Z
prometheusrules.monitoring.coreos.com     2022-12-08T16:04:14Z
servicemonitors.monitoring.coreos.com     2022-12-08T16:04:15Z
thanosrulers.monitoring.coreos.com        2022-12-08T16:04:15Z

```

## 4.2 - Viewing Data with Grafana

The previous installation also installed **Grafana**.

Grafana is an open source interactive data visualization platform, developed by Grafana Labs, that allows users to view their data via charts and graphs that are unified into a dashboard (or multiple dashboards) for easier interpretation and understanding.

Consult the list of Grafana objects:

```

trainee@gateway:~$ kubectl get all | grep grafana
pod/prometheus-grafana-5d9f5d6499-f4x6t      3/3      Running      1 (13h ago)      14h
service/prometheus-grafana                  ClusterIP  10.109.207.199  <none>           80/TCP
14h
deployment.apps/prometheus-grafana          1/1      1            1                14h
replicaset.apps/prometheus-grafana-5d9f5d6499  1        1            1                14h

```

Check the port used by Grafana:

```

trainee@gateway:~$ kubectl logs prometheus-grafana-5d9f5d6499-f4x6t -c grafana | grep HTTP
logger=http.server t=2022-12-08T16:16:51.215644746Z level=info msg="HTTP Server Listen" address=[::]:3000
protocol=http subUrl= socket=

```

and the user name to connect to Grafana :

```
trainee@gateway:~$ kubectl logs prometheus-grafana-5d9f5d6499-f4x6t -c grafana | grep "user="  
logger=sqlstore t=2022-12-08T16:16:50.536980031Z level=info msg="Created default admin" user=admin
```

The default password for the **admin** user can be obtained by consulting the contents of the [values.yaml](#) file.

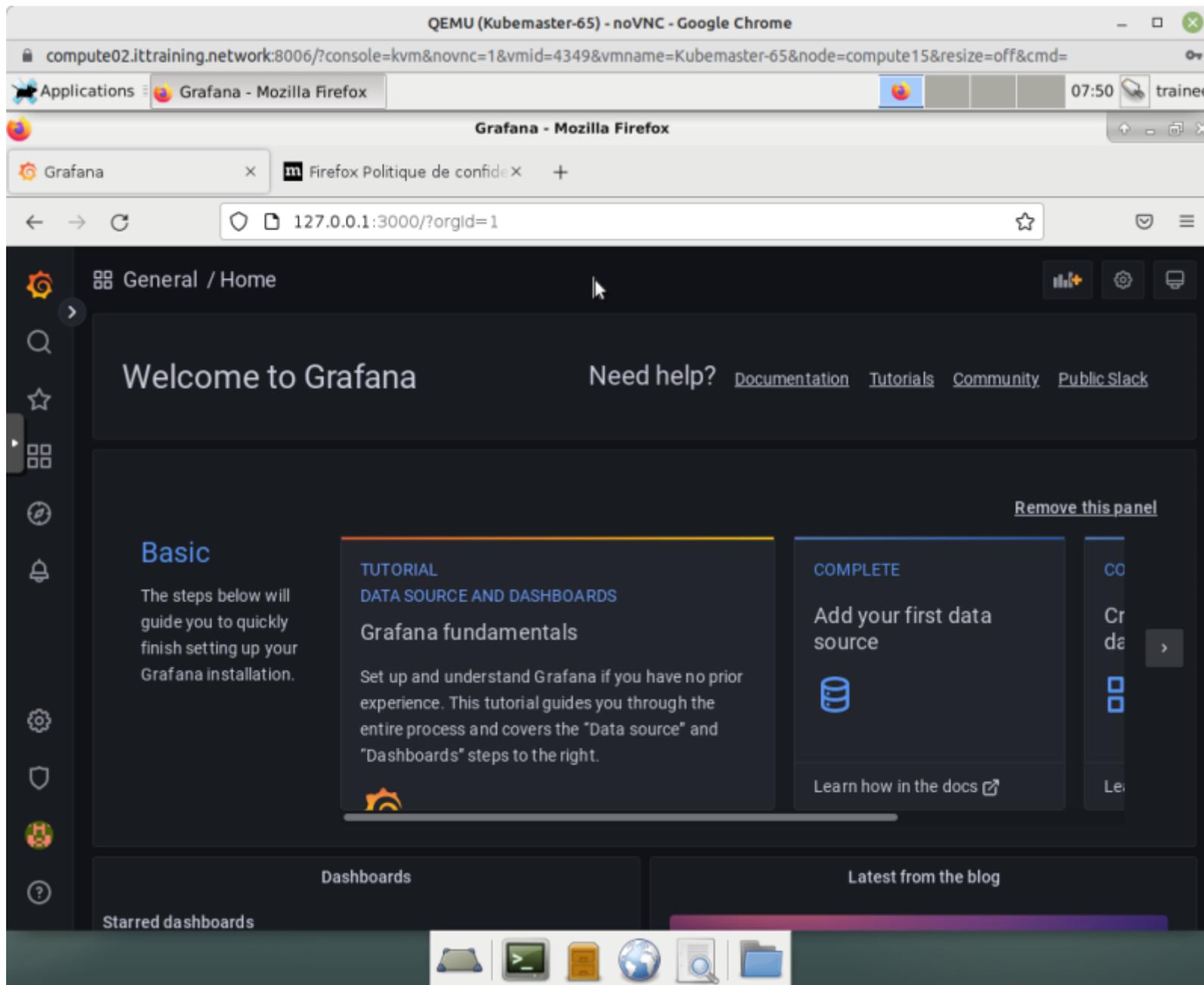


**Important:** Note that the password is **prom-operator**.

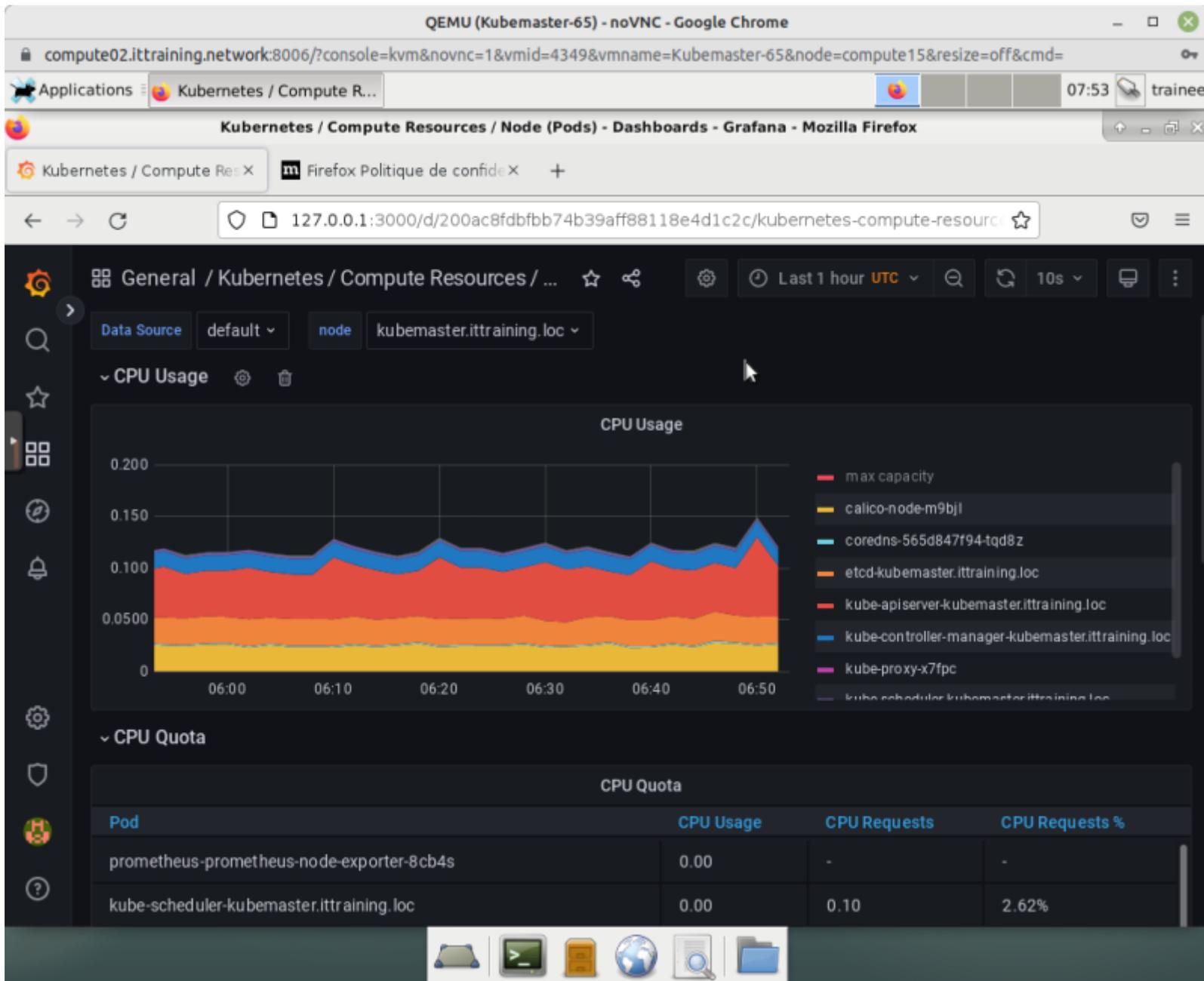
Set up port forwarding:

```
trainee@gateway:~$ kubectl port-forward deployment/prometheus-grafana 3000  
Forwarding from 127.0.0.1:3000 -> 3000  
Forwarding from [::1]:3000 -> 3000
```

Now go to the VM **Gateway\_10.0.2.40\_VNC** and launch the web browser. Enter the URL <http://127.0.0.1:3000> and connect to Grafana:

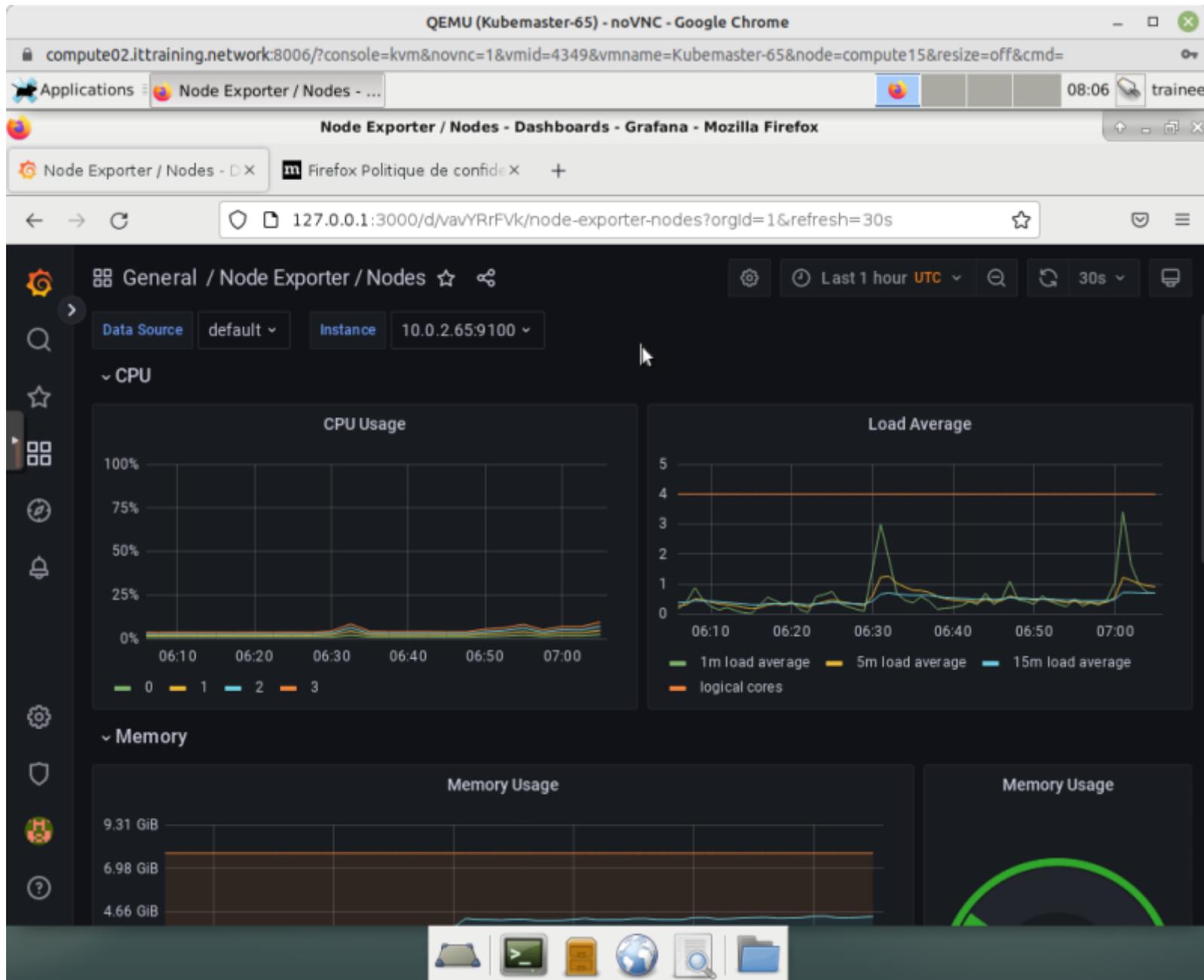


Then click on **Dashboards > Browse > Kubernetes / Compute Resources / Node (Pods)** :



Finally, click on **Dashboards > Browse > Node Exporter / Nodes** :

---



### 4.3 - Viewing Alerts with the Prometheus Web UI

To consult the Prometheus Web UI, set up a port redirection :

```
trainee@gateway:~$ kubectl port-forward prometheus-prometheus-kube-prometheus-prometheus-0 9090
Forwarding from 127.0.0.1:9090 -> 9090
Forwarding from [::1]:9090 -> 9090
```

Return to the VM GUI **Gateway\_10.0.2.40\_VNC** and enter the URL <http://127.0.0.1:9090> :

The screenshot shows a QEMU (Kubemaster-65) - noVNC - Google Chrome window. The browser address bar shows the URL: `compute02.ittraining.network:8006/?console=kvm&novnc=1&vmid=4349&vmname=Kubemaster-65&node=compute15&resize=off&cmd=`. The browser tabs include 'Prometheus Time Series...' and 'Firefox Politique de confide...'. The main content area displays the Prometheus Time Series Collection and Processing Server interface in Mozilla Firefox. The interface has a dark header with the Prometheus logo and navigation links: Alerts, Graph, Status, and Help. Below the header, there are several checkboxes: 'Use local time' (unchecked), 'Enable query history' (unchecked), 'Enable autocomplete' (checked), 'Enable highlighting' (checked), and 'Enable linter' (checked). A search bar contains the text 'Expression (press Shift+Enter for newlines)' and an 'Execute' button. Below the search bar, there are tabs for 'Table' and 'Graph', with 'Graph' selected. A 'Evaluation time' selector is visible, and the main area displays 'No data queried yet'. At the bottom right of the panel, there is a 'Remove Panel' link. A blue 'Add Panel' button is located below the panel. The bottom of the screenshot shows a taskbar with icons for a folder, a terminal, a file, a globe, a search, and another folder.

To consult the list of alerts, click on the Alerts link in the menu at the top of the page:

---

The screenshot shows a web browser window displaying the Prometheus Alerts page. The browser's address bar shows the URL `127.0.0.1:9090/alerts?search=`. The page header includes the Prometheus logo and navigation links for Alerts, Graph, Status, and Help. Below the header, there are summary statistics for alert states: Inactive (125), Pending (1), and Firing (8). A search bar labeled "Filter by name or labels" and a checkbox for "Show annotations" are also visible. The main content area lists several alert rules, each with a status indicator (e.g., "inactive") and a count of active instances (0). The rules listed are:

- `/etc/prometheus/rules/prometheus-prometheus-kube-prometheus-prometheus-rulefiles-0/default-prometheus-kube-prometheus-alertmanager.rules-64168678-2ff7-4fd3-bbf7-1b7de8005b74.yaml > alertmanager.rules` (inactive)
- `> AlertmanagerFailedReload` (0 active)
- `> AlertmanagerMembersInconsistent` (0 active)
- `> AlertmanagerFailedToSendAlerts` (0 active)
- `> AlertmanagerClusterFailedToSendAlerts` (0 active)
- `> AlertmanagerClusterFailedToSendAlerts` (0 active)
- `> AlertmanagerConfigInconsistent` (0 active)
- `> AlertmanagerClusterDown` (0 active)
- `> AlertmanagerClusterCrashlooping` (0 active)

Copyright © 2025 Hugh Norris

---