Version - **2024.01**

Dernière mise-à-jour : 2024/12/20 13:54

DOF310 - StatefulSets, StorageClass Avancé, Helm Avancé et Monitoring

Contenu du Module

Ressources

Lab #1

- https://www.dropbox.com/scl/fi/xqqpanbovwrx7cknd0yam/quarkus-service.yaml?rlkey=buou4viy128u7cgxapwmpetpl&dl=0
- https://www.dropbox.com/scl/fi/zqrdnfhcxuzcfftgbokx6/statefulset.yaml?rlkey=tqs0xxdlxjlukv30crwy2gll1&dl=0

Lab #2

- https://www.dropbox.com/scl/fi/rk3xnorqu6gk6tstvlivz/pvc.yaml?rlkey=g1dr28lrs6ec6iejp07q2o4jf&dl=0
- https://www.dropbox.com/scl/fi/1rjljxupug5wra2zpu84n/nfs-busybox.yaml?rlkey=yta13fyr2rh6a6dmsnjl10p7b&dl=0
- https://www.dropbox.com/scl/fi/b2ocglzuqbadnnyzipfyc/pvc2.yaml?rlkey=xxc7wz3pwdo4ybfqa54zav63z&dl=0

Lab #3

- https://www.dropbox.com/scl/fi/0d5znog6rdou1doko43yy/ghost.yaml?rlkey=hebkdn9ch0v9nctimiayondwc&dl=0
- https://www.dropbox.com/scl/fi/m6fmpsz25lqugzqfqxpdt/ghost-service.yaml?rlkey=zyxh7ep17eujbuycddqjdsrqy&dl=0
- https://www.dropbox.com/scl/fi/zot4i0u0hf4yw2yj3kyey/values.yaml?rlkey=apv0grxwvomxa9c0avig87pzy&dl=0
- https://www.dropbox.com/scl/fi/zyf0mbbp3wuwnnfzzez3a/service.yaml?rlkey=47bpjs3f6u474f8v0tiunl3am&dl=0
- https://www.dropbox.com/scl/fi/kark41xnz5hlilag5on0y/ghost-values.yaml?rlkey=fohccb7rwc8z66qhn4heyn2lj&dl=0

StatefulSets

Un StatefulSet est un composant de Kubernetes qui est utilisé pour des applications avec état (*Stateful Applications*).

Des exemples d'applications avec état sont :

- MySQL
- elasticsearch
- mongoDB

Ces applications enregistrent les données client des activités d'une session pour les utiliser lors de la session suivante. Les données enregistrées sont appelées l'état de l'application.

Les applications avec état sont déployées en utilisant un StatefulSet tandis que des applications **sans** état sont déployées en utilisant un **Deployment**.

Les StatefulSets et les Deployments sont similaires dans la mesure où les deux répliquent de multiples pods basés sur une spécification **identique** d'un conteneur.

La différence entre un StatefulSet et un Deployment est que dans un StatefulSet les pods ne sont **pas** identiques et possèdent ce que l'on appelle un **Pod Identity**. De ce fait les pods :

- ne peuvent pas être créés ou supprimés en même temps,
- ne peuvent pas être adressés d'une manière aléatoire.

Prenons le cas d'un StatfulSet contenant trois répliques d'un pod MySQL :

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

Notez que :

- le nom du pod prend la forme **$(Nom_du_StatefulSet)-$(ordinal)** où l'ordinal commence à **0**
- le StatefulSet ne créera pas le pod suivant tant que le pod précédent n'est pas dans un état de **Running**
- dans le cas de la suppression du StatefulSet ou bien dans le cas d'un **scale down**, les pods sont supprimés dans l'ordre inverse de leur création,

par exemple **mysql-2 > mysql-1 > mysql-0**. Chaque pod doit être complètement supprimé avant que K8s procède à la suppression du suivant

Dans ce cas de notre StatefulSet, les trois pods :

- ne peuvent pas tous accepter des requêtes d'écriture car ceci donnerait des données incohérentes,
- peuvent tous accepter des requêtes de lecture.

De ce fait, un mécanisme du StatefulSet choisit un **maître** pour accepter des requêtes d'écriture, par exemple :

- mysql-0 - écriture / lecture - **Maître**
- mysql-1 - lecture seulement - **Esclave**
- mysql-2 - lecture seulement - **Esclave**

Il existe donc une différence précise entre le pod Maître et les deux pods Esclaves.

La différence entre les deux pods Esclaves s'expliquent par le fait que les pods n'utilisent **pas** le même stockage physique persistant et distant :

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

De façon à ce que chaque pod contient les mêmes données, un mecanisme de réplication en continu doit être mis en place entre les deux pods Esclaves et le pod Maître.

Dans le cas où un nouveau pod est ajouté au **cluster** MySQL, celui-ci doit commencer par cloner les données du dernier pod dans le cluster existant, puis il doit commencer la réplication des données avec le Maître :

- mysql-0 - données
- mysql-1 - données répliquées de mysql-0
- mysql-2 - données répliquées de mysql-0
- mysql-3 - clone des données du pod mysql-2 puis, par la suite, données répliquées de mysql-0

L'état de chaque pod, incluant sa Pod Identity, est stocké dans le stockage physique à côté des données. De ce fait, quand un pod est remplacé, et un nouveau pod ajouté, ce pod nouveau hérite de l'identité de l'ancien pod.

Par exemple, si on supprime le pod **mysql-1**, on obtient :

- mysql-0 - /data/vol/pv1
- pod supprimé - /data/vol/pv2 = stockage physique persistant et distant **non-supprimé**
- mysql-2 - /data/vol/pv3
- mysql-3 - /data/vol/pv4

En ajoutant un pod de remplaceement, on obtient :

- mysql-0
- mysql-1 «««««« Le /data/vol/pv2 est rattaché au pod. Le nouveau pod s'appelle **mysql-1** et non **mysql-4**.
- mysql-2
- mysql-3

Lors de la création d'un ReplicaSet, un service d'équilibrage de charge est créé. Ce service attribue un **Endpoint DNS** unique à chaque pod. L'Endpoint DNS prend la forme **$(Nom_du_pod).$(Nom_du_service).$(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**

De cette façon, quand un pod est redémarré bien que son adresse IP changera :

- son nom ne changera **pas**
- son Endpoint DNS ne changera **pas**

Pour résumer :

- mysql-0
  - Rôle : **Maître**
  - Données : écriture / lecture
  - Stockage : /data/vol/pv1
  - Endpoint DNS : mysql-0..mysvc.default.svc.cluster.local
- mysql-1
  - Rôle : **Esclave**
  - Données : lecture seulement

    - Stockage : /data/vol/pv2
    - Endpoint DNS : mysql-1.mysvc.default.svc.cluster.local
- mysql-2
    - Rôle : **Esclave**
    - Données : lecture seulement
    - Stockage : /data/vol/pv3
    - Endpoint DNS : mysql-2.mysvc.default.svc.cluster.local
- mysql-3
    - Rôle : **Esclave**
    - Données : lecture seulement
    - Stockage : /data/vol/pv4
    - Endpoint DNS : mysql-3.mysvc.default.svc.cluster.local

Dernièrement, un StatefulSet est un composant K8s **compliqué** et difficile à mettre en oeuvre car Kubernetes ne s'occupe pas de certaines tâches telles :

- la configuration du clonage des données
- la configuration de la réplication des données
- la création et la configuration du stockage physique persistant et distant
- la configuration et la gestion des sauvegardes des données

LAB #1 - Mise en Place d'un StatefulSet Simple

Créez un Namespace **quarkus** puis modifiez le **context** de **kubernetes-admin@kubernetes** :

```
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 est un framework Java natif pour Kubernetes complet, conçu pour

les machines virtuelles Java (JVM) et la compilation native, qui permet d'optimiser Java spécifiquement pour les conteneurs afin d'en faire une plateforme efficace pour les environnements serverless, cloud et Kubernetes.

Si vous souhaitez observer les résultats des commandes suivantes en temps réel, ouvrez un deuxième terminal et saisissez la commande suivante :

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

1.1 - Création du Service et du StatefulSet

Créez maintenant le fichier **quarkus-service.yaml** :

```
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 : Notez le nom du service - **quarkus**. La valeur **None** de l'entrée **ClusterIP**

rend le service **headless**. Dans ce cas, le serveur DNS renverra les adresses IP des pods individuels au lieu de l'adresse IP du service. Le client peut alors se connecter à n'importe lequel d'entre eux.

Créez le service :

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

Créez maintenant le fichier **statefulset.yaml** :

```
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 : Notez que la valeur de **serviceName** est **quarkus**.

Créez le StatefulSet :

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

Constatez la présence des deux pods dans le 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  kubenode2.ittraining.loc
quarkus-statefulset-1  1/1    Running  106s   kubenode1.ittraining.loc
```

Contrôlez l'état du StatefulSet :

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

ainsi que la présence du service :

```
root@kubemaster:~# kubectl get services
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
```

```
quarkus   ClusterIP   None         <none>        8080/TCP   12m
```

1.2 - Mise en Place d'un Scale Up

Procédez à un scale up :

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

Important : Notez que le nom court d'un **serviceName** est **sts**.

Constatez la présence des **trois** pods dans le 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
```

Constatez l'ordre de création des pods :

```
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
```

Créez maintenant un pod pour interroger le DNS de K8s :

```
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 - Mise en Place d'un Scale Down

Procédez maintenant à un scale down :

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

Constatez la présence de **deux** pods dans le 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  kubenode2.ittraining.loc
quarkus-statefulset-1  1/1    Running  20m  kubenode1.ittraining.loc
```

1.4 - Suppression du StatefulSet

Pour terminer, supprimez le StatefulSet, le service et le 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.
```

StorageClass Avancé

LAB #2 - Provisionnement NFS dynamique

2.1 - Configuration du Serveur NFS

Connectez-vous à la VM CentOS8 en tant que trainee au 10.0.2.45.

Devenez root puis créez le répertoire **/srv/nfs/kubedata** :

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

Continuez maintenant par activer et démarrer le service **nfs-server** :

```
[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.
```

Editez le fichier **/etc/exports** :

```
[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 : Dans ce cas, nous avons partagé le répertoire **/srv/nfs/kubedata** avec le monde.

Appliquez l'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)
```

Passez SELinux en mode permissive :

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

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

Configurez ensuite le pare-feu :

```
[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 - Configuration des Clients NFS

Revenez à votre gateway et connectez-vous en tant que l'utilisateur **trainee** à **kubenode2** au 192.168.56.4. Devenez ensuite l'utilisateur root :

```
trainee@kubenode2:~$ su -
Mot de passe : fenestros
root@kubenode2:~#
```

Installez le paquet **nfs-common** :

```
root@kubenode2:~# apt update
...

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

Vérifiez que vous pouvez voir le répertoire exporté par le 10.0.2.45 :

```
root@kubenode2:~# showmount --exports 10.0.2.45
```

```
Export list for 10.0.2.45:
/srv/nfs/kubedata *
```

Vérifiez que vous pouvez monter le répertoire exporté par le 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,rsize=524288,wsize=524288,namlen=255,hard,proto=tcp,port=0,timeo=600,retrans=2,sec=sys,clie
ntaddr=10.0.2.67,local_lock=none,addr=10.0.2.45)
```

Démontez ensuite **10.0.2.45:/srv/nfs/kubedata** :

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

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

Connectez-vous à kubenode1 au 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/aJgUc9jg56slNaZQdAUcvBOvE.
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 -
Mot de passe : fenestros
root@kubenode1:~#
```

Installez ensuite le paquet **nfs-common** :

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

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

Revenez à votre gateway :

```
root@kubenode1:~# exit
déconnexion
trainee@kubenode1:~$ exit
déconnexion
Connection to 192.168.56.3 closed.
root@kubenode2:~# exit
déconnexion
trainee@kubenode2:~$ exit
déconnexion
Connection to 192.168.56.4 closed.
```

2.3 - Configuration de K8s

Connectez-vous à votre **kubemaster** au 192.168.56.2.

Installez ensuite le paquet **nfs-common** :

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

```
...

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

Ajoutez le dépôt **nfs-subdir-external-provisioner** à **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
```

Installez le chart helm **nfs-subdir-external-provisioner** :

```
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
```

Contrôlez l'état du pod créé :

```
root@kubemaster:~# kubectl get pods -o wide
NAME                                          READY   STATUS              RESTARTS      AGE   IP
NODE                      NOMINATED NODE   READINESS GATES
netshoot                                      1/1     Running             3 (25h ago)   70d   192.168.239.58
kubenode1.ittraining.loc   <none>           <none>
nfs-subdir-external-provisioner-59b4b5c476-wxkp4   0/1     ContainerCreating   0             19m   <none>
kubenode1.ittraining.loc   <none>           <none>
nginx-netshoot                                1/1     Running             3 (25h ago)   70d   192.168.239.59
kubenode1.ittraining.loc   <none>           <none>
postgresql-6f885d8957-tnlbb                   1/1     Running             3 (25h ago)   70d   192.168.239.62
```

```
kubenode1.ittraining.loc   <none>           <none>
sharedvolume                                      2/2     Running            6 (25h ago)   78d   192.168.150.60
kubenode2.ittraining.loc   <none>           <none>
troubleshooting                                   1/1     Running            3 (25h ago)   70d   192.168.239.60
kubenode1.ittraining.loc   <none>           <none>
volumepod                                         0/1     Completed          0             78d   192.168.150.41
kubenode2.ittraining.loc   <none>           <none>
```

Important : Si le pod **nfs-subdir-external-provisioner-yyyyyyyyyy-xxxxx** reste dans un état de **ContainerCreating** pour plus de 5 minutes, supprimez les trois pods **calico-node-xxxxx** du Namespace **kube-system** et attendez qu'ils soient recréés.

Une fois recréés, vous pouvez constater que le pod **nfs-subdir-external-provisioner-yyyyyyyyyy-xxxxx** est dans un état de Running :

```
root@kubemaster:~# kubectl get pods -o wide
NAMESPACE          NAME                                                READY   STATUS      RESTARTS         AGE
IP               NODE                        NOMINATED NODE   READINESS GATES
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>
```

L'examen du log du pod **nfs-subdir-external-provisioner-yyyyyyyyyy-xxxxx** démontre que tout fonctionne :

```
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!
```

Consultez maintenant la liste des StorageClasses :

```
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 - Création d'un PersistentVolumeClaim

Créez maintenant le fichier **pvc.yaml** :

```
root@kubemaster:~# vi pvc.yaml
root@kubemaster:~# cat pvc.yaml
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc1
spec:
  storageClassName: nfs-client
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 500Mi
```

Appliquez le fichier pvc.yaml :

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

Constatez maintenant la liste de PersistentVolumes et de 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            nfs-
client     67s
```

Important : Notez que le PersistentVolume **persistentvolume/pvc-721f5ed3-88b1-41bb-82c2-9eab3b4464da** a été créé automatiquement.

Connectez-vous au serveur NFS et constatez le contenu du répertoire **/srv/nfs/kubedata** :

```
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 - Utilisation du PersistentVolumeClaim avec un pod

Créez maintenant le fichier **nfs-busybox.yaml** :

```
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: pvc1
```

Appliquez le fichier nfs-busybox.yaml :

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

Vérifiez que le statut du pod **nfs-pv-pod** est **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
```

Connectez-vous au conteneur du pod **nfs-pv-pod** :

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

Créez le fichier **hello** dans le répertoire **pv-pod-storage** :

```
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
```

Connectez-vous au serveur NFS et constatez le contenu du répertoire **/srv/nfs/kubedata** :

```
root@kubemaster:~# ssh -l trainee 10.0.2.45
trainee@10.0.2.45's password:
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 : Notez la présence du fichier **hello**.

2.6 - Création d'un Deuxième PersistentVolumeClaim

Vréez le fichier **pvc2.yaml** :

```
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
```

Appliquez le fichier **pvc2.yaml** :

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

Constatez maintenant la liste de PersistentVolumes et de 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-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            nfs-
client     53m
persistentvolumeclaim/pvc2   Bound    pvc-6dbce6de-e473-4e4c-99be-0fbea26576de   100Mi      RWO            nfs-
client     58s
```

Important : Notez que le PersistentVolume **persistentvolume/pvc-6dbce6de-e473-4e4c-99be-0fbea26576de** a été créé automatiquement.

2.7 - Suppression des PersistentVolumeClaims

Commencer par supprimer le pod **nfs-pv-pod** :

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

Constatez la suppression effective du pod :

```
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
volumepod                                         0/1     Completed   0              78d
```

Constatez maintenant la liste de PersistentVolumes et de 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-6dbce6de-e473-4e4c-99be-0fbea26576de   100Mi      RWO            Delete           Bound
default/pvc2   nfs-client              27m
persistentvolume/pvc-721f5ed3-88b1-41bb-82c2-9eab3b4464da   500Mi      RWX            Delete           Bound
default/pvc1   nfs-client              79m

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

Important : Notez que les PersistentVolumes et les PersistentVolumeClaims sont toujours présents.

Supprimes les deux PersistentVolumeClaims :

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

Constatez ensuite que les deux PersistentVolumes ont été supprimés automatiquement :

```
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
```

Connectez-vous au serveur NFS et constatez le contenu du répertoire **/srv/nfs/kubedata** :

```
root@kubemaster:~# ssh -l trainee 10.0.2.45
trainee@10.0.2.45's password:
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 : Notez que les répertoires ont un préfixe **archived-**

Helm Avancé

Un chart est une collection de fichiers et de répertoires qui prennent la forme suivante :

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

Le langage des templates de helm est basé sur le langage GO.

Dans le LAB suivant, vous allez prendre les deux manifests suivants, **ghost.yaml** et **ghost-service.yaml** et créer un chart helm pour installer **Ghost**, une plateforme de blogs gratuite, sous licence logiciel libre :

```
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
```

```
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 - Création d'un Paquet Helm Simple

Commencez par créer le répertoire **~/ghost** et placez-vous dedans :

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

Un chart necéssite la présence d'un fichier nommé **Chart.yaml** qui doit décrire le chart en question. Créez donc ce fichier :

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

3.1 - Le Fichier values.yaml

Un chart a aussi besoin d'un fichier dénommé **values.yaml** qui contient des valeurs de configuration du chart en question. Créez donc le fichier values.yaml avec le contenu suivant :

```
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 - Les Templates

Créez le sous-répertoire **templates** dans **ghost** :

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

Copiez le contenu du fichier **~/ghost-service.yaml** et collez-le dans le fichier **~/ghost/templates/service.yaml** :

```
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
```

Modifiez ensuite ce fichier pour lire les valeurs du fichier **values.yaml** :

```
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 }}
```

Naviguez vers le répertoire parent du répertoire **ghost** :

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

Vérifiez que helm peut lire la liste des valeurs du fichier values.yaml :

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

L'erreur obtenu fait référence au fichier **Chart.yaml**, actuellement vide. Editez donc ce fichier :

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

Vérifiez maintenant que helm peut lire la liste des valeurs du fichier values.yaml :

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

Vérifiez maintenant que le manifest **service.yaml** qui sera créé par Helm est 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
```

Copiez maintenant le contenu du fichier **~/ghost.yaml** et collez-le dans le fichier **~/ghost/templates/ghost.yaml** :

```
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
```

Modifiez ensuite ce fichier pour lire les valeurs du fichier **values.yaml** :

```
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 }}
```

Completez maintenant le contenu du fichier values.yaml :

```
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
```

Vérifiez maintenant que le manifest **ghost.yaml** qui sera créé par Helm est 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
```

Consultez maintenant l'organisation du chart **ghost** :

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

```
1 directory, 4 files
```

3.3 - Installation et Suppression

Installez le chart **ghost** :

```
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
```

Vérifiez l'état du service dans le 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
```

Vérifiez la présence du pod dans le 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
```

Vérifiez le statut du chart :

```
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
```

Dernièrement, supprimez le chart :

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

Monitoring

Le serveur Prometheus est composé de trois modules :

- **Data Retrieval Worker** qui récupére les métriques
- **Time Series Database** qui stocke les métriques
- **HTTP Server** qui accepte des requêtes PromQL et qui fournit un Web UI pour la consultation des données

> **Important** : **PromQL**, abréviation de Prometheus Querying Language, est le principal moyen d'interroger les métriques dans Prometheus. Vous pouvez afficher le retour d'une expression sous forme de graphique ou l'exporter à l'aide de l'API HTTP. PromQL utilise trois types de données : les scalaires, les vecteurs de plage et les vecteurs instantanés. Il utilise également des chaînes, mais uniquement en tant que littéraux.

Des alertes sont ensuite passées à l'**Alertmanager** qui informe des personnes en focntion de la configuration mise en place.

LAB #4 - Mise en Place d'une Solution Prometheus

Connectez-vous à la VM **Gateway_10.0.2.40_VNC**.

4.1 - Déploiement du Stack avec Helm

Ajoutez le dépôt **prometheus-community** :

```
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
```

Installez ensuite le chart **kube-prometheus-stack** :

```
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.
```

Patientez jusqu'à ce que tous les pods soient dans un état de **Running** :

```
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
```

Consultez maintenant l'ensemb le des objets Prometheus créés par l'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   3         3         3       3            3           <none>
13h

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
```

Dans cette sortie on constate :

- 2 StatefulSets dont :
  - le serveur Prometheus **statefulset.apps/prometheus-prometheus-kube-prometheus-prometheus**
  - l'Alertmanager **statefulset.apps/alertmanager-prometheus-kube-prometheus-alertmanager**
- 2 Deployments dont :
  - l'operator **deployment.apps/prometheus-kube-prometheus-operator** qui a créé les deux StatefulSets
  - le kube-state-metrics **deployment.apps/prometheus-kube-state-metrics** qui est une dépendance de Prometheus et donc un **Subchart** de ce dernier
- 2 ReplicaSets créés par les Deployments :
  - **replicaset.apps/prometheus-kube-prometheus-operator-689dd6679c**
  - **replicaset.apps/prometheus-kube-state-metrics-6cfd96f4c8**
- 1 DaemonSet **daemonset.apps/prometheus-prometheus-node-exporter** :
  - les pods de ce DaemonSet sont responsables pour la transformation des métriques des noeuds en métriques Prometheus

L'installation a aussi créé un grand nombre de 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
```

ainsi que des 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
```

des **Custom Resource Definitions** ou 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 - Consultation des Données avec Grafana

L'installation du chart helm a aussi installé **Grafana**.

Grafana est une plate-forme de visualisation de données interactive open source, développée par Grafana Labs, qui permet aux utilisateurs de voir leurs données via des tableaux et des graphiques qui sont unifiés dans un tableau de bord (ou plusieurs tableaux de bord) pour une interprétation et une compréhension plus faciles.

Consultez les objets Grafana :

```
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
```

Vérifiez le port utilisé par 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=
```

ainsi que le nom de l'utilisateur pour se connecter à 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
```

Le mot de passe par défaut de l'utilisateur **admin** peut être obtenu en consultant le contenu du fichier **values.yaml**

> **Important** : Notez que le mot de passe est **prom-operator**.

Mettez en place une redirection de port :

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

Consultez maintenant la VM **Gateway_10.0.2.40_VNC** et lancez le navigateur web. Saisissez l'URL **http://127.0.0.1:3000** et connectez-vous à Grafana :

Cliquez ensuite sur **Dashboards > Browse > Kubernetes / Compute Resources / Node (Pods)** :

Dernièrement, cliquez ensuite sur **Dashboards > Browse > Node Exporter / Nodes** :

4.3 - Consultation des Alertes avec le Web UI de Prometheus

Pour consultez le Web UI de Prometheus, mettez en place une redirection de port :

```
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
```

Retournez dans l'interface graphique de la VM **Gateway_10.0.2.40_VNC** et saisissez l'URL **http://127.0.0.1:9090** :

Pour consultez la liste des alertes, cliquez sur le lien Alerts dans le menu en haut de la page :