BEWIJSSTUKKEN STAGE

Kubernetes-cluster - lokaal

r0743766
Alessandro Dierickx
3 CCS A

THOMAS MORE

Inhoudsopgave

INLEIDING

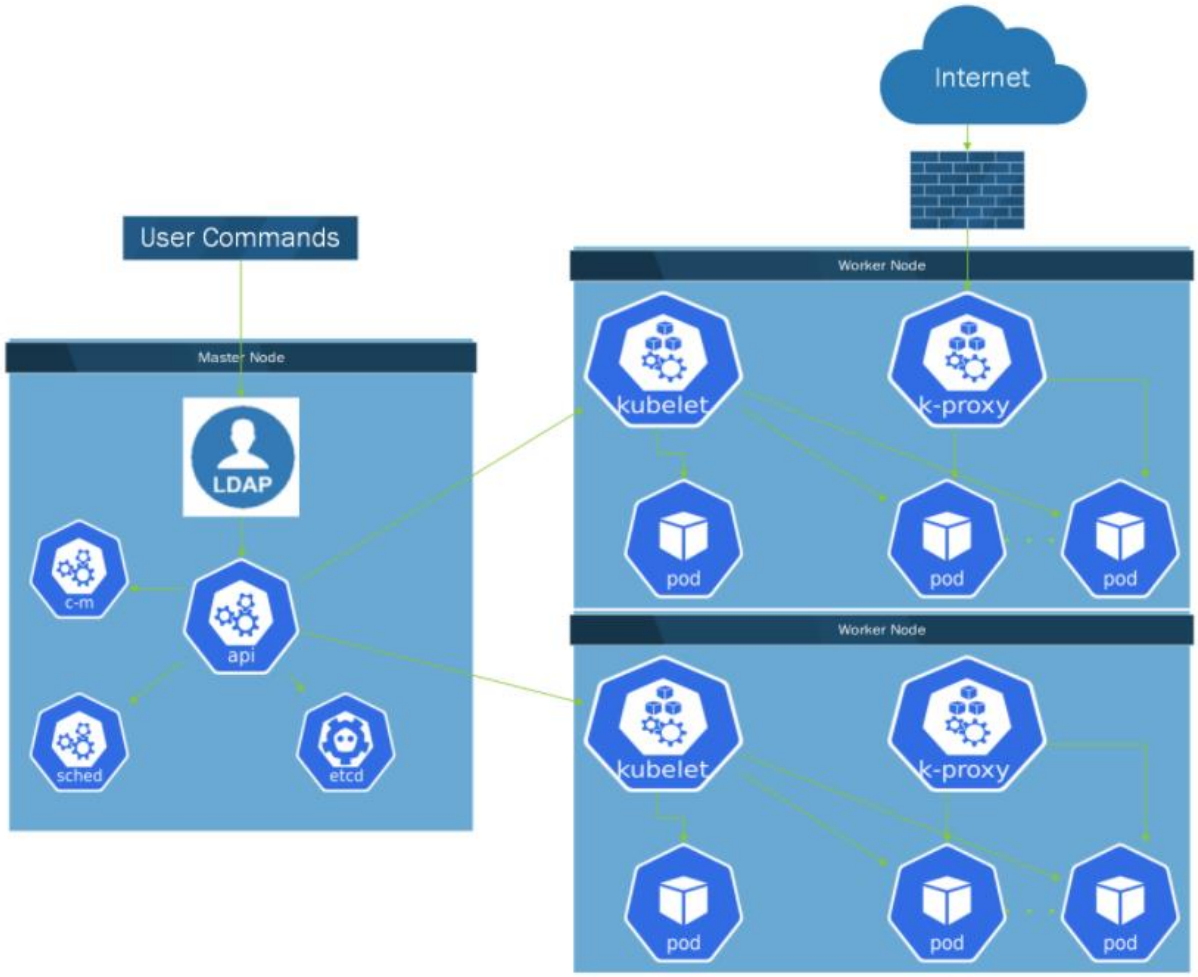
In dit document zal ik het hebben over een aantal dingen. Ten eerste zal ik schetsen waarover mijn stage gaat, waarna ik verder ga met mijn aanpak hierrond. Ten laatste zal ik nog alle bewijsstukken uitleggen.

Mijn stage bij Vito in Mol is het opzetten van een lokale Kubernetes-cluster om een probleem op te lossen dat zich momenteel voordoet. Dat probleem is het feit dat de verscheidene afdelingen van Vito allemaal hun eigen Docker-omgeving hebben om hun applicaties op te draaien.

Dit is een enorm onoverzichtelijke manier van werken, waardoor men bij Vito graag een Kubernetes-cluster wil hebben waar iedereen op kan.

Wat is Kubernetes?

Kubernetes is een open-source oplossing voor het implementeren en beheren van containers. Het is met Kubernetes eenvoudiger om deze zaken te doen, alleen is de leercurve hiervan nogal hoog. Hieronder vindt u een voorbeeld van een Kubernetes-cluster



AANPAK + BEWIJSSTUKKEN

Aanpak

Om dit te kunnen realiseren heb ik een groot stuk van de stage onderzoek gedaan. Dit onderzoek ging in het begin over de algemene aanpak van het opstellen van een cluster en hoe ik eventuele functionaliteiten zou kunnen implementeren.

Deze functionaliteiten waren extra's, maar zijn toch een belangrijk deel van de voorbereiding geweest. Dit is vooral zo omdat er hierbij enorm veel benodigdheden tevoorschijn kwamen. Door het onderzoek naar de functionaliteiten, zoals bijvoorbeeld Multitenancy of High Availability, heb ik veel kunnen leren over de werking van Kubernetes en verscheidene extensies ervan. Hierdoor vond ik het belangrijk om deze onderzoeksfase aan te halen, aangezien dit er zeker voor heeft gezorgd dat ik enorm veel had bijgeleerd voordat ik echt met de opdracht kon beginnen.

Nadat ik met de opdracht begonnen was, heb ik wel op verscheidene momenten blokkades gehad. Deze blokkades waren onvoorziene problemen die ik moest oplossen, problemen die me enorm veel tijd hebben gekost over de tijdspanne van de stage. Hierdoor moet ik jammer genoeg vaststellen dat zelfs mijn extensief onderzoek niet genoeg was om alle aspecten van het opstellen van een Kubernetes-cluster te kunnen bekijken en begrijpen.

In het stuk van bewijsstukken zal ik stap voor stap de installatie van mijn Kubernetes-cluster tonen. Zo kan ik van mijn scripts en bewijsstukken een verhaal maken dat iedereen kan volgen.

Bewijsstukken

Begin Installatie

We zullen beginnen bij het begin, namelijk de initiële installatie van Kubernetes. We gaan er bij deze installatie vanuit dat we twee Linux-machines hebben met daarop Ubuntu 18.04.
Bij voorkeur zijn deze machines servers.

Als eerste taak voordat we kunnen starten, moeten we zorgen dat onze machines aan het internet kunnen raken. Dit doen we door netplan aan te passen zoals u hieronder kan zien.

Om te starten moeten we zorgen dat onze servers up to date zijn. Dit doen we aan de hand van het sudo apt-get update commando. Hierna kunnen we Docker installeren met
sudo apt-get install docker.io en moeten we Docker laten opstarten wanneer de machines booten. Dit doen we via sudo systemctl enable docker.

De volgende stap is het toevoegen van een ondertekening-sleutel om de authenciteit van een Kubernetes-repository te kunnen bevestigen. Dit gebeurd via het commando curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add.

Wanneer we de sleutel hebben toegevoegd, kunnen we de Kubernetes-repository toevoegen door sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main" te doen.

De volgende stap is het toevoegen van Kubeadm, de tool waarmee we de Kubernetes-cluster zullen initiëren. Dit gebeurd via deze twee commando's:

```
sudo apt-get install kubeadm kubelet kubectl
sudo apt-mark hold kubeadm kubelet kubectl
```

Dan komt er nu een belangrijk stuk voor de toekomst, het is van enorm belang om swap af te zetten op alle servers. Permanent liefst, door het swap geheugen helemaal te verwijderen van de server. Swap kan voor problemen zorgen die je liever niet hebt tijdens het opzetten van Kubernetes.

Het initiëren van Kubernetes gebeurd via het sudo kubeadm init commando. Hierbij moet men dan nog het pod network cidr meegeven. Dit laatste is eigenlijk de ip range waaruit Kubernetes pods ip adresses krijgen. Het commando wordt dan sudo kubeadm init –pod-network-cidr='ip_range_cluster'. Hierna krijgt men een join commando te zien dat ons toelaat worker nodes aan de cluster toe te voegen. Dit commando moet men dus bijhouden.

De volgende stap is het aanmaken van een cluster directory via deze commando's:

```
kubernetes-master:~$ mkdir -p $HOME/.kube
kubernetes-master:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.
kube/config
kubernetes-master:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/conf
ig
```

De volgende stap is het opzetten van een pod network, hiervoor heb ik twee verschillende mogelijkheden klaarstaan, namelijk calico en kube-ovn.

Calico

Calico met Tigera

```
kubectl create -f https://docs.projectcalico.org/manifests/tigera-operator.yaml
kubectl create -f https://docs.projectcalico.org/manifests/custom-resources.yaml
```

bovenstaande custom-resources.yaml aanpassen → hieronder yaml zetten

```
watch kubectl get pods -n calico-system
kubectl taint nodes --all node-role.kubernetes.io/master-
kubectl get nodes -o wide
```

Calicoctl

```
curl -o kubectl-calico -O -L  "https://github.com/projectcalico/calicoctl/releases/download/v3.19.0/calicoctl"
chmod +x kubectl-calico
kubectl calico -h
```

You can now run any `calicoctl` subcommands through `kubectl calico.`

Kube-OVN

Om Kube-OVN te installeren, moeten we het installatiescript ervan aanpassen naar onze eigen noden. Deze noden hangen af van wat we exact willen doen met dit pod-netwerk.

```
ENABLE_SSL=${ENABLE_SSL:-false}
ENABLE_MIRROR=${ENABLE_MIRROR:-false}
HW_OFFLOAD=${HW_OFFLOAD:-false}
IFACE=""                                # The nic to support container network can be a nic name or a group of regex separated by comma,

REGISTRY="kubeovn"
VERSION="v1.6.2"
IMAGE_PULL_POLICY="IfNotPresent"
POD_CIDR=                                  # Do NOT overlap with NODE/SVC/JOIN CIDR
SVC_CIDR="10.96.0.0/12"                 # Do NOT overlap with NODE/POD/JOIN CIDR
JOIN_CIDR="100.64.0.0/16"               # Do NOT overlap with NODE/POD/SVC CIDR
PINGER_EXTERNAL_ADDRESS=                # Pinger check external ip probe
PINGER_EXTERNAL_DOMAIN=                    # Pinger check external domain probe
if [ "$IPv6" = "true" ]; then
  POD_CIDR="fd00:10:16::/64"               # Do NOT overlap with NODE/SVC/JOIN CIDR
  SVC_CIDR="fd00:10:96::/112"              # Do NOT overlap with NODE/POD/JOIN CIDR
  JOIN_CIDR="fd00:100:64::/64"             # Do NOT overlap with NODE/POD/SVC CIDR
  PINGER_EXTERNAL_ADDRESS="2400:3200::1"
  PINGER_EXTERNAL_DOMAIN="google.com"
fi
if [ "$DualStack" = "true" ]; then
  POD_CIDR="10.16.0.0/16,fd00:10:16::/64"              # Do NOT overlap with NODE/SVC/JOIN CIDR
  SVC_CIDR="10.96.0.0/12"                              # Do NOT overlap with NODE/POD/JOIN CIDR
  JOIN_CIDR="100.64.0.0/16,fd00:100:64::/64"           # Do NOT overlap with NODE/POD/SVC CIDR
  PINGER_EXTERNAL_ADDRESS="114.114.114.114"
  PINGER_EXTERNAL_DOMAIN="google.com"
fi

EXCLUDE_IPS=""                          # EXCLUDE_IPS for default subnet
LABEL="node-role.kubernetes.io/master" # The node label to deploy OVN DB
NETWORK_TYPE="hybrid"                   # geneve or vlan

# VLAN Config only take effect when NETWORK_TYPE is vlan
PROVIDER_NAME="provider"
VLAN_INTERFACE_NAME="ens192"
VLAN_NAME="ovn-vlan"
VLAN_ID="0"
VLAN_RANGE="1,4095"
```

Het is hier vooral belangrijk om de CIDR in te stellen aangezien dit de ip ranges zijn waarin het netwerk zich zal bevinden. Pinger address en domain slaan op de nameserver en het domein van je bedrijf.

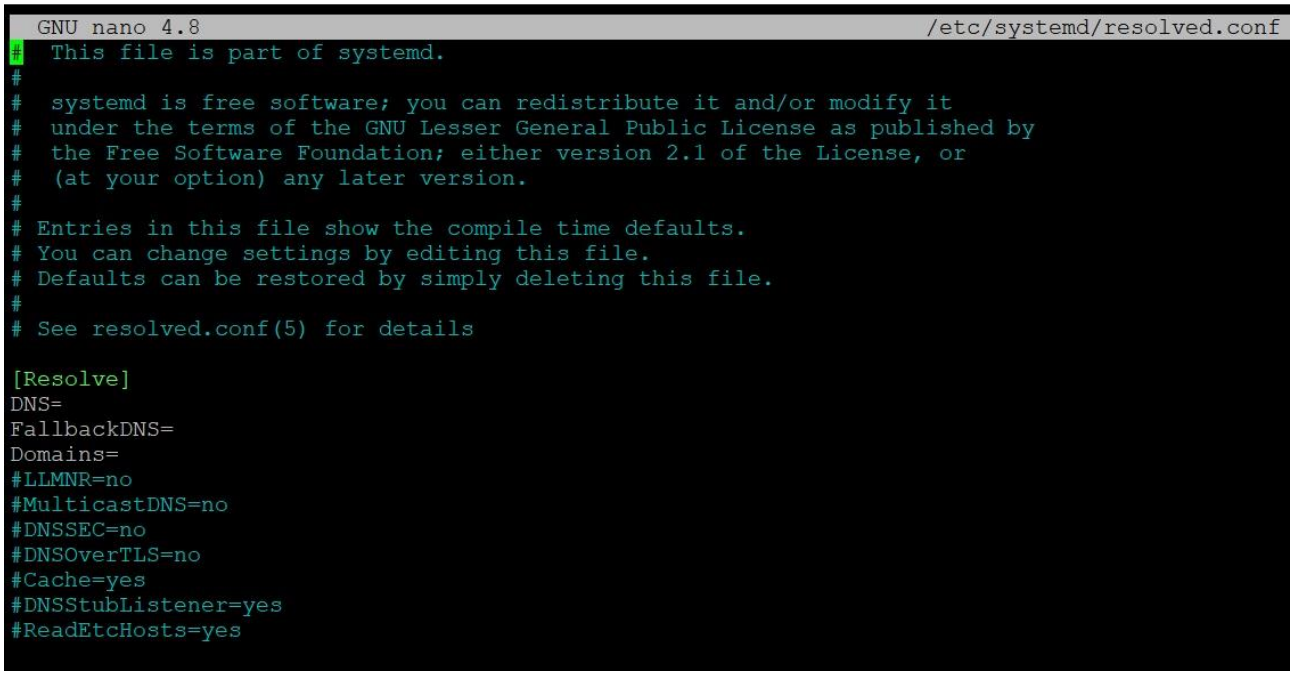
Network type slaat op het soort netwerk dat je wil opstellen. Hier hebben we voor hybrid gekozen omdat dit ons toelaat een gewoon netwerk op te stellen naast een vlan netwerk.

DNS

Vervolgens gaan we de dns gegevens moeten aanpassen zodat deze overeenkomen met de gegevens van onze omgeving. In het belang van Vito kan ik de exacte configuratie niet laten zien, dus zal deze vervaagd worden.

Het is de bedoeling om op zowel de master-node als de worker-node de resolved.conf file te editen. Dit kunnen we doen door sudo nano /etc/systemd/resolved.conf te gebruiken.

In deze file moeten we de dns en fallback dns invullen met de nameservers die aanwezig zouden moeten zijn in je bedrijf. Het is ook van belang om het domein aan te passen zodat de Kubernetes-cluster kan communiceren met alles in het bedrijf.



Om deze nieuwe configuratie toe te passen, moeten we het commando sudo service systemd-resolved restart runnen.

Installatie NFS

Om uiteindelijk Portainer te kunnen runnen als interface voor het beheer van Kubernetes, moeten we zorgen dat we storage kunnen voorzien voor onze pods. Dit heb ik in mijn stage gedaan via NFS omdat dit de meest betrouwbare en realistische oplossing was.

De installatie van NFS eist dat we eerst een NFS server hebben waarvan we de storage kunnen halen. Voor deze installatie gaan we er vanuit dat er al een NFS server aanwezig is in het bedrijf. Het eerste commando dat we nodig hebben is

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

Dit commando zorgt ervoor dat we de basis nfs-commandos kunnen uitvoeren op onze nodes.

Hierna moeten we een mount point voorzien waarop we de shared NFS folder kunnen plaatsen. Dit gebeurt via het

```
sudo mkdir -p /mnt/sharedfolder_client
```

commando. Het commando voor het mounten van de NFS shared folder hangt wat af van je eigen omgeving, maar ik ga de versimpelde versie geven van het Vito commando, namelijk sudo mount -vvvv -t nfs -o nfsvers=”nfs-version” “locatie_NFS_shared_folder” “locatie_mount_point”

Vervolgens moeten we nog een volume aanmaken waar Portainer gebruik van kan maken. Dit doen we door onderstaande yaml file aan te passen naar onze persoonlijke noden en omgevingsfactoren zoals de locatie van de shared folder, de storage capacity en het ip adres van de NFS server.

Wanneer we dit alles hebben, moeten we nog de NFS Provisioner aanmaken. De NFS Provisioner is een service die ervoor zorgt dat volumes gevuld kunnen worden en mogelijks groter of kleiner gemaakt kunnen worden. Dit is noodzakelijk om containers zoals Portainer fatsoenlijk te laten runnen.

Om deze provisioner te installen, moet men een aantal yaml files downloaden en aanpassen naar eigen omgevingsfactoren, op dezelfde manier als dat moest gebeuren voor de volumes in de vorige alinea. Wanneer deze fatsoenlijk zijn aangepast, moet men enkel nog het kubectl apply -f commando gebruiken op deze yaml files, en dan zou het moeten werken.

Indien er problemen zijn met het gebruik van nfs, kan het nodig zijn om de commando’s sudo systemctl enable rpcbind en sudo systemctl start rpcbind te runnen.

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-pv
spec:
  capacity:
    storage: 10Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: nfs
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /opt/k8s-pods/data
    server: 192.168.1.40
```

Portainer

Om Portainer, onze Kubernetes interface, te kunnen installeren hebben we een storageclass nodig. Deze hebben we aangemaakt door middel van de NFS Provisioner. Hierna kunnen we verder gaan met de installatie.

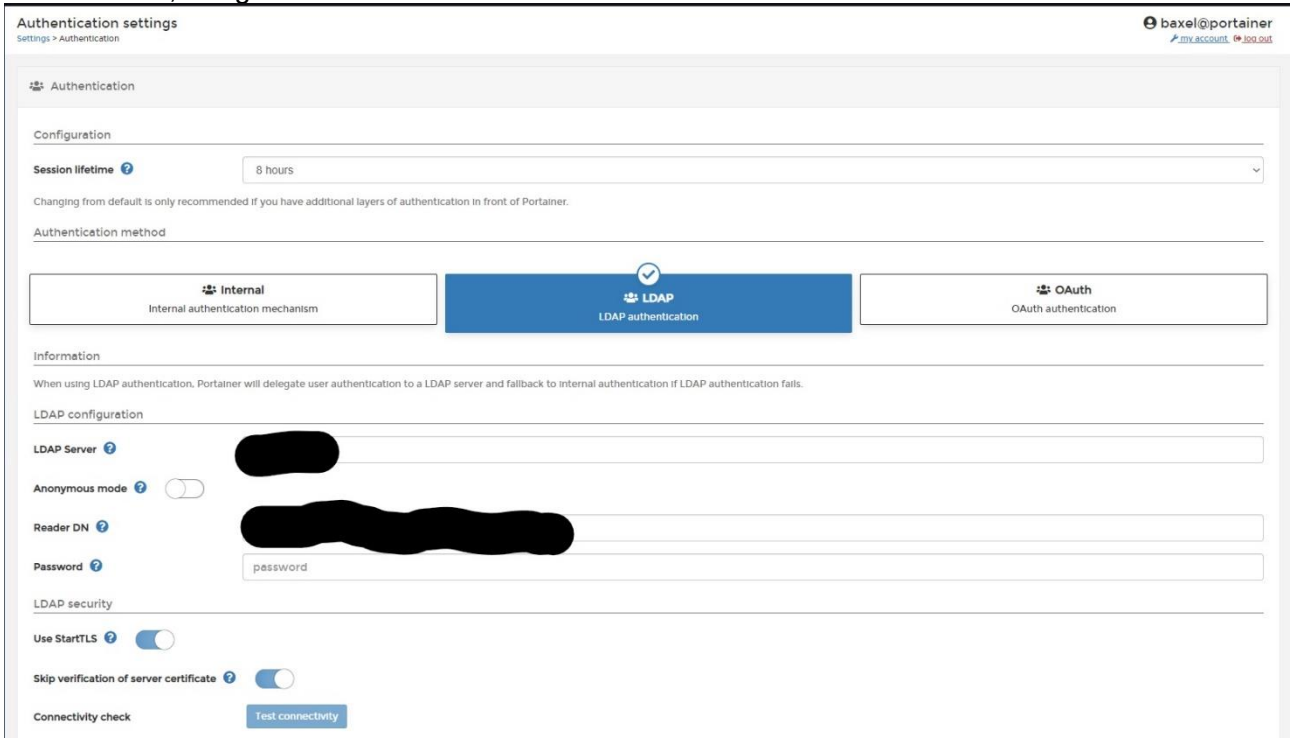
```
kubectl apply -n portainer -f
https://raw.githubusercontent.com/portainer/k8s/master/deploy/manifests/portainer/
portainer-ee.yaml
```

Met dit commando zorgen we ervoor dat alles rond Portainer, met andere woorden alle pods, deployments en services, geïnstalleerd worden op onze node. Omdat we dit via NodePort doen, komt Portainer dan zichtbaar op poort 30777.

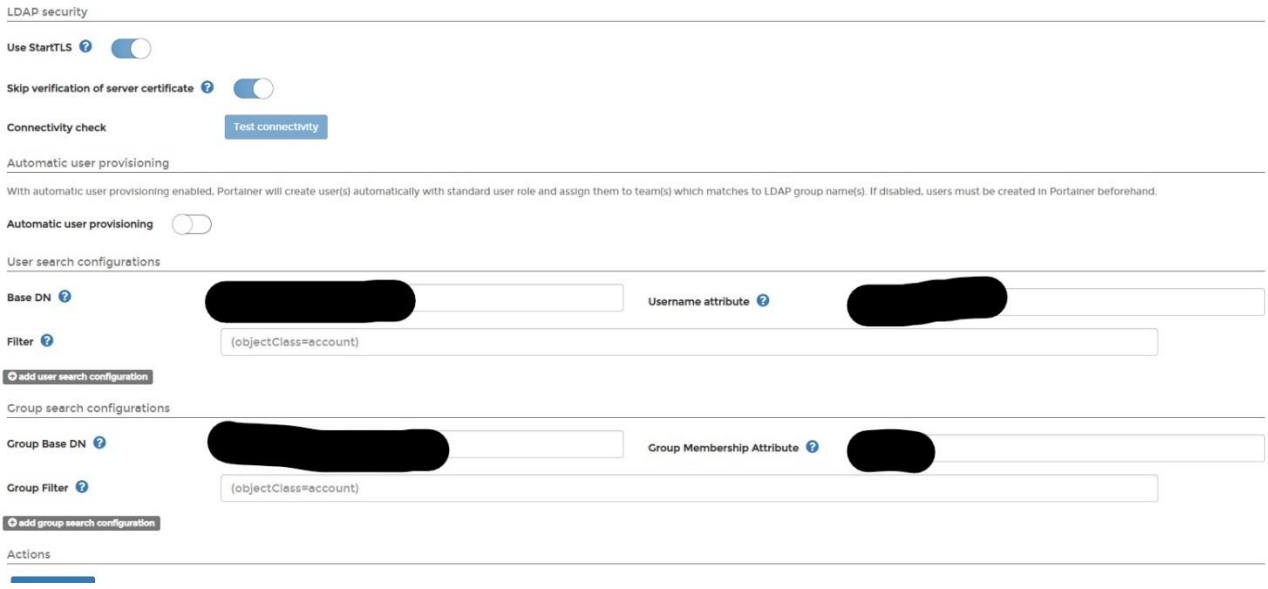
Het is natuurlijk belangrijk om te zien welke versie van Portainer men wilt gebruiken. In dit voorbeeld zijn we voor de Business of Enterprise Edition gegaan. Deze editie geeft de gebruiker een paar extra mogelijkheden en functionaliteiten, zoals built-in RBAC wat zeker handig is in een bedrijfsomgeving.

LDAP

Als laatste gaan we nog LDAP implementeren in onze Kubernetes-cluster. Dit doen we via Portainer, aangezien dit een functionaliteit van Portainer is.



Ten eerste moeten we zorgen dat we onze LDAP server specifiëren. Hierna moeten we inloggen met onze Reader DN, die ik jammer genoeg niet mag laten zien hier. Qua security moeten we kiezen tussen TLS of StartTLS, wat in mijn geval StartTLS is gebleken. Indien men het verkeerde aanduid komt er een foutmelding op van Portainer uit.



Verder kan u hier zien dat we nog wat search configurations moeten specifiëren. Deze kan ik spijtig genoeg ook niet laten zien, maar dit is vaak gewoon de default optie met misschien licht andere attributen voor username en group membership.

Bronvermelding

- Tucakov, D. (z.d.). *How to Install Kubernetes on Ubuntu 18.04*. Knowledge Base by PhoenixNAP. Geraadpleegd op 5 maart 2021, van https://phoenixnap.com/kb/install-kubernetes-on-ubuntu
- Project Calico. (z.d.). *Install Calico networking and network policy for on-premises deployments*. Project Calico. Geraadpleegd op 23 maart 2021, van https://docs.projectcalico.org/getting-started/kubernetes/self-managed-onprem/onpremises
- Buzdar, K. (2020, 28 juni). *Install NFS Server and Client on Ubuntu*. VITUX. Geraadpleegd op 7 april 2021, van https://vitux.com/install-nfs-server-and-client-on-ubuntu/
- Kumar, P. (2020, 26 augustus). *How to Configure NFS based Persistent Volume in Kubernetes*. LinuxTechi. Geraadpleegd op 8 april 2021, van https://www.linuxtechi.com/configure-nfs-persistent-volume-kubernetes/
- pohly, K. (2021, 5 maart). *kubernetes-sigs/nfs-subdir-external-provisioner*. GitHub. Geraadpleegd op 12 april 2021, van https://github.com/kubernetes-sigs/nfs-subdir-external-provisioner
- Oilbeater, K. (2020, 30 mei). *kubeovn/kube-ovn*. GitHub. Geraadpleegd op 22 mei 2021, van https://github.com/kubeovn/kube-ovn