Kubernetes

Projet GitHub vagrant-kube-2

Kick start

Release: **Edition date:**	0.0 19/06/2020

Summary:

Mise en œuvre d'un Kubernetes Kubeadm.

Ce document aborde l'ensemble des processus d'installation nécessaires basés sur le projet GitHub vagrant-kube-2 : https://github.com/Varappe/vagrant-kube-2.git

- Prérequis
  - DNS
  - Proxy
  - Paquets
  - Environnement Vagrant
- Kubernetes Kubeadm avec Vagrant
- Exposition des services sur le serveur hôte
  - Attribution d'une adresse IP externe avec MetalLB
- Dashboard Kubernetes
- Exposition des services sur internet
  - Reverse proxy avec apache2

Nœuds : peut-être modifier dans le fichier `Vagrantfile`

- 1 master subnet.2
- 10 workers du subnet.5 à subnet.14 (2 par défaut)

Services : peut-être modifier et étendu dans le fichier `metallb-config.yaml`

- 240 services avec une adresse IP externe de subnet.15 à subnet.254

Keywords:	Administration Linux, Vagrant, Kubernetes, MetalLB, reverse proxy, apache2, exposer un service Kubernetes

Contenu

1 Prérequis sur le serveur hôte

Le serveur hôte servant pour ce Kick start est un Dell power Edge R630:

- OS: Linux Ubuntu 16.04.6 LTS
- Kernel: Linux 4.15.0-91-lowlatency (x86_64)
- CPU: 56 x Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz
- Cores: 56 x 14 = 784
- RAM: 264043MB

1.1 DNS

Le programme `resolvconf` génère automatiquement et écrase le fichier `/etc/resolv.conf`.

La configuration du DNS se fait dans le fichier `/etc/network/interfaces` :

```
cd /etc/network
sudo vi interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eno1
iface eno1 inet static
        address <adresse IP du serveur>
        netmask 255.255.255.0
        network <adresse de réseau>
        broadcast <adresse de broadcast>
        gateway <adresse de la gateway>
        # dns-* options are implemented by the resolvconf package, if installed
        dns-nameservers <adresse du DNS 1> <adresse du DNS 2> <…>
        dns-search <nom du domaine du réseau>
```

Application de la configuration du DNS :

```
sudo systemctl restart networking
```

Vérifiez que la configuration apparait bien dans le fichier `/etc/resolv.conf`.

1.2 PROXY

Si vous êtes derrière un proxy l'environnement de votre session doit avoir ces informations. Vous devez également renseigner le `no_proxy` notamment avec les adresses utilisées par vos environnements docker (172.0.0.0/8) et le subnet de Kubernetes (192.168.0.0/16) :

```
export {http,https,ftp}_proxy='http://<URL de votre proxy>:<port de votre proxy>'
export {HTTP,HTTPS,FTP}_PROXY='http://<URL de votre proxy>:<port de votre proxy>'

export {no_proxy,NO_PROXY}='localhost,127.0.0.0/8,*.<nom du domaine de votre réseau>,<vos adresses et
URLs ne devants pas passer par le proxy>,10.96.0.0/12,10.244.0.0/16,172.17.0.0/16,192.168.0.0/16'


export {no_proxy,NO_PROXY}='localhost,127.0.0.0/8,*.<nom du domaine de votre réseau>,<vos URL ne
devants pas passer par le proxy>,172.0.0.0/8,192.168.0.0/16'
```

Pour que ces variables d'environnement soient chargés à chaque démarrage de session il est possible de rajouter ces lignes à la fin de votre fichier `~/.bashrc`.

1.3 Paquets

Mettre à jour la liste des paquets disponibles et des paquets installés :

```
sudo apt-get update
sudo apt-get upgrade
```

1.3.1 Installer virtualbox et Vagrant

```
sudo apt-get install virtualbox vagrant
```

1.3.2 Installer kubectl

Pour plus de détails voir https://kubernetes.io/fr/docs/tasks/tools/install-kubectl/.

```
sudo apt-get update && sudo apt-get install -y apt-transport-https
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a
/etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install kubectl
```

1.3.3 Installer Brew

Pour plus d'informations : https://brew.sh/

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"


# Install the Homebrew dependencies:
sudo apt-get update
sudo apt-get install build-essential

# Configure Homebrew in your ~/.profile by running
echo 'eval $(/home/linuxbrew/.linuxbrew/bin/brew shellenv)' >> ~/.profile

# Add Homebrew to your PATH
eval $(/home/linuxbrew/.linuxbrew/bin/brew shellenv)

# Install helm completion
sudo cp /home/linuxbrew/.linuxbrew/etc/bash_completion.d/brew /etc/bash_completion.d/

# Install GCC:
sudo apt-get install gcc
```

1.3.4 Installer Helm

Pour plus d'informations : https://helm.sh/

```
brew install helm

# Install helm completion
sudo cp /home/linuxbrew/.linuxbrew/etc/bash_completion.d/helm /etc/bash_completion.d/

# Initialize a Helm Chart Repository
helm repo add stable https://kubernetes-charts.storage.googleapis.com/

# Update helm repo
helm repo update
```

1.4 Environnement proxy de Vagrant

Si vous êtes derrière un proxy vous avez besoin d'installer le plugin `vagrant-proxy` :

```
vagrant plugin install vagrant-proxyconf
```

Il existe plusieurs façons de configurer l'environnement proxy pour Vagrant que vous trouverez en annexes. J'ai choisie de configurer le proxy dans ficher `Vagrantfile` du projet `vagrant-kube-2`.

Vous devez renseigner le `http_proxy`, `https_proxy`, `ftp_proxy` et le `no_proxy` notamment avec les adresses des clusters Kubernetes (10.96.0.0/12), les adresses de Kubernetes Container Network Interface (*CNI*) et les adresses des Pods (*CIDR Flannel*) (10.244.0.0/16), les adresses des clusters des installations à venir (10.0.0.0/8), les adresses utilisées par docker dans les VMs (172.17.0.0/16) et le subnet de Kubernetes (192.168.0.0/16).

Voici la configuration proxy dans le fichier `Vagrantfile` :

```
...
Vagrant.configure("2") do |config|
# Vagrant proxy
  if Vagrant.has_plugin?("vagrant-proxyconf")
    config.proxy.http     = "http://<URL de votre proxy>:<port de votre proxy>"
    puts "http_proxy: " + config.proxy.http
    config.proxy.https    = "http://<URL de votre proxy>:<port de votre proxy>"
    puts "https_proxy: " + config.proxy.https
    config.proxy.ftp      = "http://<URL de votre proxy>:<port de votre proxy>"
    puts "ftp_proxy: " + config.proxy.ftp
    config.proxy.no_proxy = "localhost,127.0.0.0/8,10.0.0.0/8,172.17.0.0/16,192.168.0.0/16"
    puts "no_proxy: " + config.proxy.no_proxy
  end
...
```

2 Installation de Kubernetes Kubeadm avec Vagrant

Cloner le projet vagrant-kube-2 :

```
git clone https://github.com/Varappe/vagrant-kube-2.git
```

2.1 Configuration de Kubernetes

Paramètres modifiables :

- VAGRANT_MASTER_RAM : Taille de la mémoire pour le master
- VAGRANT_WORKER_RAM : Taille de la mémoire pour le ou les workers
- VAGRANT_MASTER_CPU : Nombre de CPUs pour le master
- VAGRANT_WORKER_CPU : Nombre de CPUs pour le ou les workers
- VAGRANT_SUBNET : Subnet des VMs Vagrant
- VAGRANT_WORKERS : Nombre de nodes workers

La ligne `$master_ip = $subnet_ip + ".2"` signifie que l'adresse IP du nœud master sera l'adresse « 2 » du subnet.

La ligne `$ip = $subnet_ip + "." + "#{i+4}"` signifie que l'adresse IP des nœuds worker commence à l'adresse « 5 » du subnet.

Vérifier que le subnet que vous voulez donner aux VMs Kubernetes n'est pas déjà utilisé sur le serveur hôte :

```
ip a
```

Editer le fichier Vagrantfile pour configurer votre Kubernetes avec vos paramètres. Les paramètres par défaut sont les paramètres minimum conseillés :

```
vi Vagrantfile
```

2.2 Construction des nœuds et lancement de Kubernetes

La configuration faite il ne reste plus qu'à créer les VMs Vagrant ; une par nœud. Comme c'est la première fois que vous lancer Kubernetes la création des VMs va prendre un certain temps :

```
vagrant up
```

2.3 Interaction avec Kubernetes depuis le serveur hôte

La façon la plus simple à faire et de récupérer le fichier `config` de Kubernetes sur le nœud master et de la copier dans votre répertoire de travail sur le serveur hôte :

```
ssh vagrant@<adresse IP du master> -i .vagrant/machines/master/virtualbox/private_key sudo cat
/etc/kubernetes/admin.conf > config
```

Autre solution pour récupérer le fichier `config` :

```
vagrant ssh master
cp .kube/config /vagrant
exit
```

Il suffit ensuite de charger ce fichier dans votre environnement de travail sur le serveur hôte :

```
export KUBECONFIG=$PWD/config
```

Il est ainsi possible de communiquer avec le Kubernetes que vous venez d'installer sans avoir besoin de se connecter sur la VM du nœud master. Si vous avez plusieurs Kubernetes d'installé vous pouvez ainsi passer de l'un à l'autre avec l'export du fichier `config` correspondant.

Attention cet export n'est valable que votre session. Si vous n'avez qu'un seul Kubernetes à administrer vous pouvez mettre cet export directement dans votre fichier `~/.bashrc`.

Voir la configuration du Kubernetes que vous venez de configurer :

```
kubectl config view
```

Vérifiez que les nœuds sont tous bien montés et Ready :

```
kubectl get nodes -o wide

NAME       STATUS   ROLES    AGE   VERSION   INTERNAL-IP      EXTERNAL-IP   OS-IMAGE
KERNEL-VERSION       CONTAINER-RUNTIME
master     Ready    master   16d   v1.18.3   192.168.66.1     <none>        Ubuntu 18.04.4 LTS
4.15.0-101-generic   docker://19.3.8
worker-1   Ready    <none>   16d   v1.18.3   192.168.66.5    <none>        Ubuntu 18.04.4 LTS   4.15.0-
101-generic   docker://19.3.8
worker-2   Ready    <none>   16d   v1.18.3   192.168.66.6    <none>        Ubuntu 18.04.4 LTS   4.15.0-
101-generic   docker://19.3.8
```

Voir les pods tournants sur Kubernetes :

```
kubectl get pods --all-namespaces -o wide
```

Voir les services tournants sur Kubernetes :

```
kubectl get svc --all-namespaces -o wide
```

3 Exposition des services sur le serveur hôte

A ce stade les services de Kubernetes ne sont pas accessibles. Il faut les rendre accessible.

J'ai choisie de rendre accessible les services Kubernetes sur le serveur hôte en leurs attribuant une adresse IP externe privé. MetalLB identifie les services en mode LoadBalancer et les expose en leurs attribuant une adresse IP parmi le pool d'adresses IP défini.

3.1 Installation de MetalLB

L'installation s'appuie sur la procédure fournie à ce jour par MetalLB. Suivre le lien pour les mises à jour de la procédure.

Par défaut kube-proxy utilise le mode iptables. MetalLB a besoin du mode IPVS et du mode strictARP. L'ensemble des modules noyaux et paquets nécessaires à l'utilisation de IPVS (IP Virtual Server) pour le service kube-proxy ont été installés précédemment dans la procédure du projet `vagrant-kube-2` :

```
kubectl edit configmap -n kube-system kube-proxy
```

Pour avoir:

```
...
    ipvs:
      excludeCIDRs: null
      minSyncPeriod: 0s
      scheduler: ""
      strictARP: true
      syncPeriod: 0s
      tcpFinTimeout: 0s
      tcpTimeout: 0s
      udpTimeout: 0s
    kind: KubeProxyConfiguration
    metricsBindAddress: ""
    mode: ipvs
...
```

Il est nécessaire de redémarrer l'ensemble des pods `kube-proxy` pour que le changement de configuration soit pris en compte :

```
kubectl get pods -n kube-system
kubectl delete pods -n kube-system <kube-proxy pod-name>
```

Avant de continuer, vérifier que l'ensemble des pods kube-proxy est bien redémarré avec le mode IPVS :

```
kubectl get pods -n kube-system
kubectl logs -n kube-system <kube-proxy pod-name> | grep "Using ipvs Proxier"
```

Installation de MetalLB :

```
kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.9.3/manifests/namespace.yaml
kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.9.3/manifests/metallb.yaml

# On first install only for RBAC authentification
kubectl create secret generic -n metallb-system memberlist --from-literal=secretkey="$(openssl rand -
base64 128)"
```

3.2 Configuration du pool d'adresse MetalLB

J'utilise le mode Layer 2 pour attribuer les adresses privées aux services.

Adapter le pool d'adresse du fichier `metallb-config.yaml` avec votre `VAGRANT_SUBNET` :

```
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    address-pools:
    - name: default
      protocol: layer2
      addresses:
      - <VAGRANT_SUBNET>.15-<VAGRANT_SUBNET>.254
```

Vous avez ainsi 240 services qui peuvent être adressés et qui seront routés sur le subnet de vos VMs vagrant donc vers votre serveur hôte. Si vous en voulez plus modifier le fichier en conséquence.

Appliquer la configuration Layer 2 de MetalLB :

```
kubectl apply -f metallb-config.yaml
```

4 Installation du Dashboard Kubernetes

Pour plus de détails :

- https://kubernetes.io/fr/docs/tasks/access-application-cluster/web-ui-dashboard/
- https://github.com/kubernetes/dashboard
- https://github.com/kubernetes/dashboard/blob/master/docs/user/access-control/creating-sample-user.md
- https://github.com/kubernetes/dashboard/tree/master/docs/user/accessing-dashboard

Installation du Dashboard :

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

Pour pouvoir se connecter au Dashboard de Kubernetes nous avons besoin d'un token. Création du Service Account et du ClusterRoleBinding :

```
kubectl apply -f serviceaccount.yaml
kubectl apply -f dashboard-adminuser.yaml
```

Récupération du Token du Dashboard de Kubernetes qui nous sera nécessaire pour s'authentifier :

```
kubectl -n kubernetes-dashboard describe secret $(kubectl -n kubernetes-dashboard get secret | grep
admin-user | awk '{print $1}')
```

4.1 Passer le service en mode LoadBalancer

MetalLB a besoin que le service Frontend des services soit en mode LoadBalancer et non ClusterIP (par défaut).

Modification du service du Dashboard Kubernetes pour le mode LoadBalancer :

```
kubectl edit svc kubernetes-dashboard -n kubernetes-dashboard


...
spec:
  clusterIP: 10.110.148.73
  ports:
  - port: 443
    protocol: TCP
    targetPort: 8443
  selector:
    k8s-app: kubernetes-dashboard
  sessionAffinity: None
  type: LoadBalancer
...
```

Le service du Dashboard Kubernetes est passé en mode LoadBalancer :

```
kubectl get svc -n kubernetes-dashboard -o wide

NAME                        TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)          AGE
SELECTOR
dashboard-metrics-scraper   ClusterIP      10.100.19.189    <none>          8000/TCP         6d2h
k8s-app=dashboard-metrics-scraper
kubernetes-dashboard        LoadBalancer   10.111.183.100   192.168.66.15   8443:31732/TCP   43s
k8s-app=kubernetes-dashboard
```

Le Dashboard Kubernetes est accessible depuis le serveur hôte : https://192.168.66.15

Utiliser le Token du Dashboard Kubernetes généré précédemment pour se loger au Dashboard.

5 Exposition des services sur internet

J'ai choisie de rendre accessible les services Kubernetes depuis internet en utilisant un reverse proxy.

5.1 Installation d'apache2

Nous allons utiliser apache2 comme reverse-proxy et openssl pour le https:

```
sudo apt-get update
sudo apt-get install apache2 openssl
```

5.2 Génération de clef et certificat pour SSL

Clef privée :

```
sudo openssl genrsa -aes256 -out certificat.key 4096
```

Déverrouiller la clef privée pour éviter de devoir renseigner la `pass phrase` à chaque démarrage d'apache2 :

```
sudo mv certificat.key certificat.key.lock
sudo openssl rsa -in certificat.key.lock -out certificat.key
```

Génération du fichier de demande de signature :

```
sudo openssl req -new -key certificat.key.lock -out certificat.csr
```

Génération du certificat :

```
openssl x509 -req -days 365 -in certificat.csr -signkey certificat.key.lock -out certificat.crt
```

5.3 Activation du mode reverse proxy

Pour utiliser Apache en mode reverse-proxy il faut activer, via la commande `a2enmod`, les deux modules `proxy`, `proxy_http` et pour l'accès en https sur le port 443 il faut le module `ssl` :

```
sudo a2enmod proxy proxy_http ssl
sudo systemctl restart apache2
```

5.4 Configurer le reverse proxy pour le service du Dashboard Kubernetes

```
cd /etc/apache2/sites-available
sudo vi https-<URL du serveur hôte>.conf

<VirtualHost *:443>
    ServerName kubernetes.<URL du serveur hôte>
    ServerAlias www.kubernetes.<URL du serveur hôte>
    ServerAdmin <Adresse mail de l'administrateur>

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    # LogLevel info ssl:warn
    ErrorLog ${APACHE_LOG_DIR}/https_error.log
    CustomLog ${APACHE_LOG_DIR}/https_access.log combined

    # Activation de SSL pour le trafic entrant, cas dans le cas d'un
    # serveur web classique.
    SSLEngine On
    SSLCertificateFile    /etc/ssl/apache2/certificat.crt
    SSLCertificateKeyFile /etc/ssl/apache2/certificat.key

    # Activation de SSL pour la communication avec les backends.
    SSLProxyEngine          On
    SSLProxyVerify          none
    SSLProxyCheckPeerCN     off
    SSLProxyCheckPeerName   off
    SSLProxyCheckPeerExpire off

    # Configuration classique du mod_proxy, mis a part qu'on specifie bien
    # https dans l'URL du backend.
    ProxyRequests     Off
    ProxyPreserveHost On

    ## Kubernetes
    ProxyPass / https://192.168.66.15:443/
    ProxyPassReverse / https://192.168.66.15:443/
</VirtualHost>
```

Pour chaque nouveaux services https sur le port 443 vous n'aurez plus faire un copier/coller de la section `<VirtualHost *:443> … </VirtualHost>` à rajouter en fin de fichier et modifier, avec les bonnes valeurs, les champs :

- ServerName
- ServerAlias
- ProxyPass
- ProxyPassReverse

Activation de ce fichier de configuration pour une prise en compte par apache2 :

```
sudo a2ensite https-<URL du serveur hôte>.conf
```

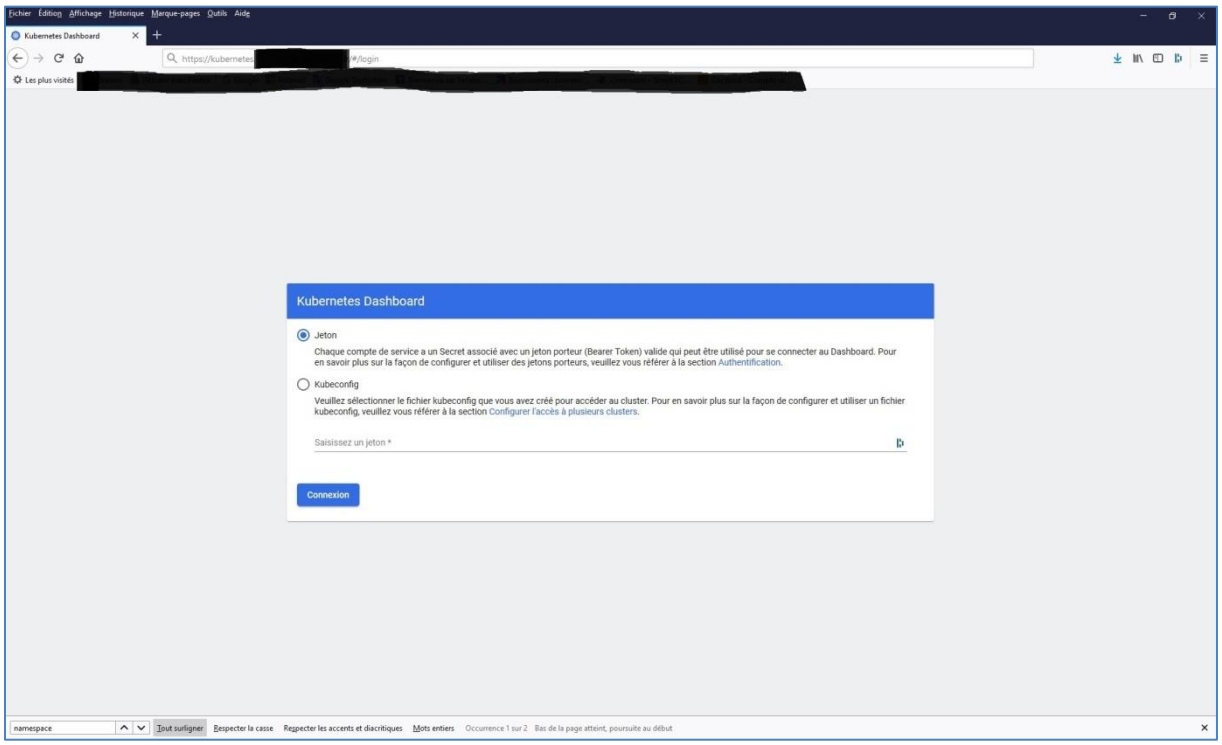
Que ce soit pour la création de ce fichier ou l'ajout d'un nouveau service, il faut qu'apache2 charge le fichier :

```
sudo systemctl restart apache2
```

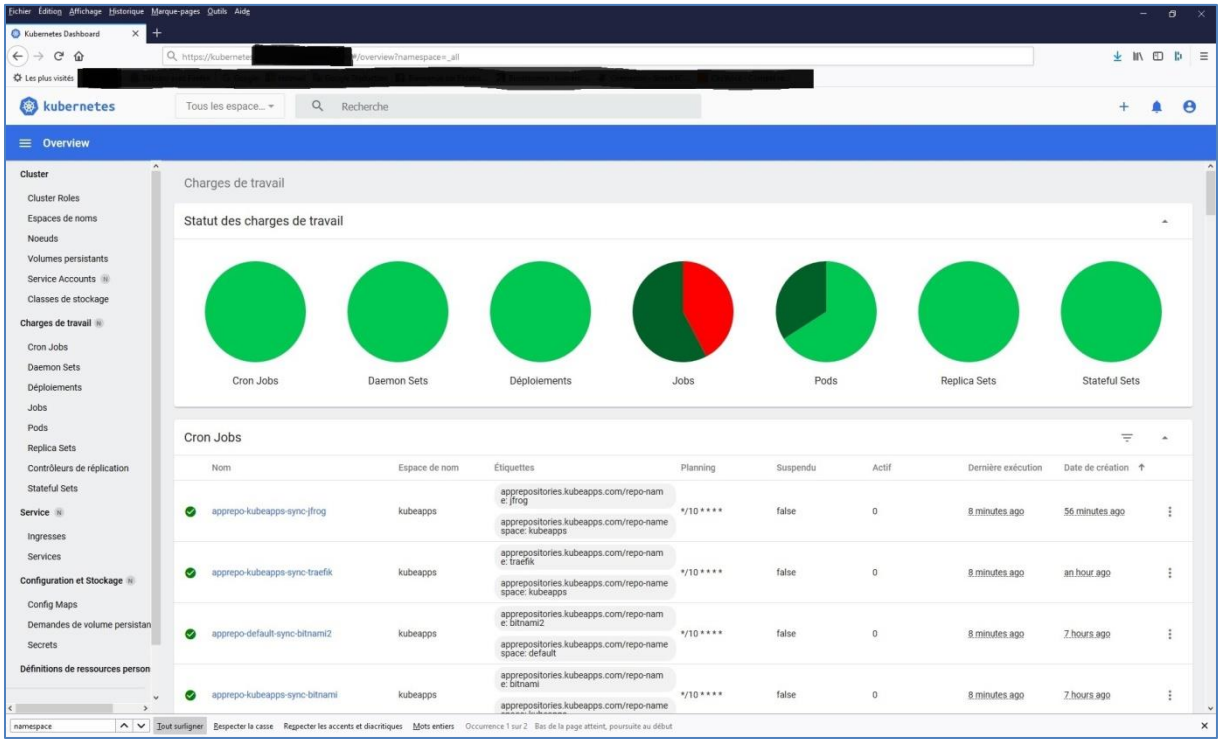
`<URL du serveur hôte>` est connu par le DNS qui résout son adresse IP. Une fois la requête http arrivée sur le serveur hôte, le serveur apache2 va rediriger `kubernetes.<URL du serveur hôte>` vers `https://192.168.66.15:443/`.

Le Dashboard Kubernetes est maintenant accessible depuis internet avec l'URL :
`http://kubernetes.<URL du serveur hôte>`

Utiliser le Token Dashboard de Kubernetes généré précédemment pour se loger au Dashboard :



Le Dashboard :



ANNEXES

1 Solutions pour configurer le proxy pour vagrant

La syntaxe ci-dessous se sert de vos variables d'environnement. Vous pouvez adapter ces configurations avec d'autres valeurs.

1.1.1.1 Pour l'ensemble de vos projets Vagrant

Ajouter cette configuration dans le fichier `~/.vagrant.d/Vagrantfile` :

```
vi ~/.vagrant.d/Vagrantfile

Vagrant.configure("2") do |config|
  puts "proxyconf..."
  if Vagrant.has_plugin?("vagrant-proxyconf")
    puts "find proxyconf plugin !"
    if ENV["http_proxy"]
      puts "http_proxy: " + ENV["http_proxy"]
      config.proxy.http     = ENV["http_proxy"]
    end
    if ENV["https_proxy"]
      puts "https_proxy: " + ENV["https_proxy"]
      config.proxy.https    = ENV["https_proxy"]
    end
    if ENV["ftp_proxy"]
      puts "ftp_proxy: " + ENV["ftp_proxy"]
      config.proxy.ftp    = ENV["ftp_proxy"]
    end
    if ENV["no_proxy"]
      puts "no_proxy: " + ENV["no_proxy"]
      config.proxy.no_proxy = ENV["no_proxy"]
    end
  end
end
```

1.1.1.2 Pour votre session en cours uniquement

```
export VAGRANT_HTTP_PROXY=$http_proxy
export VAGRANT_HTTPS_PROXY=$https_proxy
export VAGRANT_FTP_PROXY=$ftp_proxy
export VAGRANT_NO_PROXY=$no_proxy
```

1.1.1.3 Pour un projet Vagrant

Ajouter cette configuration dans le fichier `Vagrantfile` de votre projet Vagrant :

```
if Vagrant.has_plugin?("vagrant-proxyconf")
  config.proxy.http     = ENV["http_proxy"]
  config.proxy.https    = ENV["https_proxy"]
  config.proxy.ftp      = ENV["ftp_proxy"]
  config.proxy.no_proxy = ENV["no_proxy"]
end
```

2 Administration de Vagrant et Virtualbox

Voici quelques commandes vagrant qui sont à exécuter depuis votre répertoire vagrant.

2.1 Démarrer les VMs

Cette commande démarre les VMs vagrant et les crée si elle n'existe pas :

```
vagrant up
```

2.2 Voir l'état des VMs

```
vagrant status
```

2.3 Arrêter les VMs

```
vagrant halt
```

2.4 Supprimer les VMs

```
vagrant destroy
```

2.4.1 Supprimer les interfaces réseaux Virtualbox

Voir les interfaces réseaux de Virtualbox :

```
ip a | grep vboxnet
```

Pour supprimer l'interface `vboxnet0` :

```
VBoxManage hostonlyif remove vboxnet0
```

Pour supprimer toutes les interfaces `vboxnet#` :

```
for ((i=0 ; $(ip a | grep vboxnet | wc -l) - $i ; i++)); do VBoxManage hostonlyif remove vboxnet$i ;
done
```

3 Désinstallation de MetalLB

```
kubectl delete -f metallb-config.yaml

kubectl delete -f https://raw.githubusercontent.com/metallb/metallb/v0.9.3/manifests/metallb.yaml
kubectl delete -f https://raw.githubusercontent.com/metallb/metallb/v0.9.3/manifests/namespace.yaml
```

4 Désinstallation le Dashboard Kubernetes

```
kubectl delete -f dashboard-adminuser.yaml
kubectl delete -f serviceaccount.yaml
kubectl delete -f
https://raw.githubusercontent.com/kubernetes/dashboard/master/aio/deploy/recommended.yaml
```

5 Aide-mémoire kubectl

Aide-mémoire kubectl : https://kubernetes.io/fr/docs/reference/kubectl/cheatsheet/