Sécuriser un registry open source

Dans cette mise en pratique, nous allons mettre en place un registry open source et ajouter un certificats TLS pour le sécuriser et le rendre accessible à un daemon docker distant.

Mise en place de l'environnement

Création de 2 hôtes Docker

Pour faire simple, nous allons créer 2 hôtes Docker, *node1* et *node2*. Le registry sera lancé depuis node2.

Ces 2 hôtes seront des machines virtuelles créées dans VirtualBox avec Vagrant (vous pouvez également utiliser d'autres outils si vous le souhaitez). Assurez vous d'avoir VirtualBox et Vagrant installés sur votre machine.

Dans un nouveau répertoire, créez le fichier *Vagrantfile* avec le contenu suivant.

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  config.vm.define "node1" do |node|
    node.vm.box = "ubuntu/bionic64"
    node.vm.hostname = "node1"
    node.vm.network "private_network", ip: "192.168.99.100"
  end

  config.vm.define "node2" do |node|
    node.vm.box = "ubuntu/bionic64"
    node.vm.hostname = "node2"
    node.vm.network "private_network", ip: "192.168.99.101"
  end

  config.vm.provision "shell", inline: <<-SHELL
    curl -fsSL https://get.docker.com -o get-docker.sh
    sudo sh get-docker.sh
    sudo usermod -aG docker vagrant
  SHELL
end
```

Ce fichier définit 2 VMs *node1* et *node2* , avec les IPs respectives *192.168.99.100* et *192.168.99.101*. Ces VMs sont basées sur Ubuntu, le daemon Docker est installé sur chacune

d'entre elles.

Une fois le fichier créé, lancez le provionning avec la commande suivante.

```
$ vagrant up
```

Au bout de quelques minutes les VMs seront créées et configurées.

Note: pour accéder en ssh à l'une de ces machines il suffira de lancer la commande `vagrant ssh node1` ou `vagrant ssh node2` depuis le répertoire dans lequel se trouve le fichier *Vagrantfile*.

Mise à jour du /etc/hosts

Nous allons faire en sorte que le registry tournant sur le node2 soit accessible par le daemon Docker du node1 via l'URL *registry.mydom.com*.

Dans le fichier */etc/hosts* du node1, ajoutez l'entrée suivante de façon à mapper cette URL avec l'IP du node2.

```
192.168.99.101   registry.mydom.com
```

Registry sans TLS

Dans un premier temps nous allons lancer le registry sans les options de chiffrement de la communication. Il écoutera donc sur du HTTP.

Lancement du registry

Placez vous sur le node2 et lancez le registry avec la commande suivante:

```
vagrant@node2:~$ docker run -d -p 5000:5000 --restart=always --name registry
registry:2
```

Access au Registry

Depuis le node1, récupérez une l'image nginx:1.14 depuis le Docker Hub.

```
vagrant@node1:~$ docker image pull nginx:1.14
```

Tagguez ensuite cette image de façon à ce qu'elle puisse être envoyée sur le registry tournant sur le node2. Il faut pour cela ajouter l'URL du registry (host + port). Utilisez la commande suivante pour créer ce nouveau tag.

```
vagrant@node1:~$ docker tag nginx:1.14 registry.mydom.com:5000/nginx:1.14
```

Le node1 a donc 2 tags de la même image, comme le montre la commande suivante:

```
vagrant@node1:~$ docker image ls
REPOSITORY                      TAG                 IMAGE ID            CREATED
SIZE
registry.mydom.com:5000/nginx   1.14                e706cb01fa6b        11 days
ago         109MB
nginx                           1.14                e706cb01fa6b        11 days
ago         109MB
```

Toujours depuis le node1, utilisez la commande qui suit pour pusher ce nouveau tag dans le registry tournant sur le node2.

```
vagrant@node1:~$ docker image push registry.mydom.com:5000/nginx:1.14
The push refers to repository [registry.mydom.com:5000/nginx]
Get https://registry.mydom.com:5000/v2/: http: server gave HTTP response to
HTTPS client
```

Le message d'erreur était attendu, il s'explique par le fait qu'un daemon Docker ne peut pas communiquer avec un registry distant si la communication ne se fait pas en HTTPS. Il est cependant possible d'autoriser cette communication en ajoutant l'URL du registry dans les options de démarrage du daemon Docker.

Le fichier */etc/docker/daemon.json* permet d'ajouter / modifier facilement ses options de démarrage. Sur le node1, créez le fichier */etc/docker/daemon.json* et ajoutez le contenu suivant:

```
{
  "insecure-registries" : ["registry.mydom.com:5000"]
}
```

Redémarrez ensuite le daemon avec la commande suivante (sous Ubuntu le daemon est géré avec systemd)

```
$ sudo systemctl restart docker
```

Vous pouvez ensuite vérifier qu'il est à présent posible d'envoyer l'image dans le registry.

```
vagrant@node1:~$ docker image push registry.mydom.com:5000/nginx:1.14
The push refers to repository [registry.mydom.com:5000/nginx]
a988eeefbe01: Pushed
e53052eb0904: Pushed
6744ca1b1190: Pushed
1.14: digest:
sha256:9ed9873861f42054ea621811575ae06d4001ca6cc9a0f351de194e7253746e89 size:
948
```

L'exception ajoutée précédemment permet donc la communication en HTTP entre le daemon Docker et le registry. Communiquer avec un registry en HTTP n'est cependant pas conseillé, dans la suite vous allez créer un certificat pour que cette communication utilise HTTPS.

Cleanup

Sur le node1

Supprimez le fichier */etc/docker/daemon.json* et rédemarrez le daemon Docker une nouvelle fois de façon à ne pas autoriser la communication non sécurisée entre le daemon tournant sur le node1 et le registry du node2.

```
vagrant@node1:~$ sudo rm /etc/docker/daemon.json
vagrant@node1:~$ sudo systemctl restart docker
```

Sur le node2

Supprimez le container *registry* avec la commande suivante

```
vagrant@node2:~$ docker rm -f registry
```

Registry avec TLS (certificat auto-signé)

Nous allons maintenant créer un certificat de façon à sécuriser la communication entre le daemon et le registry. Pour cet exemple, le certificat sera auto-signé.

Création du certificat

Sur le node2, créez le fichier *registry.cnf* et ajoutez le contenu suivant:

```
[ req ]
prompt                 = no
default_bits           = 4096
default_md             = sha256
distinguished_name     = req_distinguished_name
x509_extensions        = v3_req

[ req_distinguished_name ]
CN = registry.mydom.com

[ v3_req ]
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName = @alt_names

[ alt_names ]
DNS.1 = registry.mydom.com
IP.1  = 192.168.99.101
```

Ce fichier permet de définir différentes options de configuration du certificat que nous allons créer dans la suite. Il contient notamment le CN (CommonName), c'est à dire le nom de domaine auquel le certificat sera attaché, ainsi que des SANs (Subject Alternative Names) qui seront ajoutés au certificat.

Lancez ensuite la création du certificat avec la commande suivante:

```
vagrant@node2:~$ mkdir certs
vagrant@node2:~$ openssl req -x509 -newkey rsa:4096 -nodes -sha256 -days 365 \
     -keyout certs/domain.key \
    -out certs/domain.crt \
    -config registry.cnf
```

Lancement du registry

Une fois le certificat créé, lancez le registry avec la commande suivante. Celle ci permet de

faire un bind-mount du répertoire *certs* dans lequel se trouvent le certificat et la clé privée associée, et de configurer le registry avec ceux-ci.

```
vagrant@node2:~$ docker run -d \
  --restart=always \
  --name registry \
  -v "$(pwd)"/certs:/certs \
  -e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt \
  -e REGISTRY_HTTP_TLS_KEY=/certs/domain.key \
  -p 5000:5000 \
  registry:2
```

Access au Registry

Depuis le node1, utilisez la commande suivante pour pusher l'image *registry.mydom.com:5000/nginx:1.14*.

```
vagrant@node1:~$ docker image push registry.mydom.com:5000/nginx:1.14
```

Vous devriez obtenir le masseage d'erreur suivant:

```
Get https://registry.mydom.com:5000/v2/: x509: certificate signed by unknown
authority
```

Pour que le certificat auto-signé que nous avons créé soit reconnu comme un certificat de confiance par le Docker daemon il faut créer le répertoire */etc/docker/certs.d/registry.mydom.com:5000* sur le node1 et ajouter le certificat dans celui-ci (en le renommant *ca.crt* au passage).

Note: pour copier ce certificat de node2 à node1, on peut utiliser le répertoire local de la machine hôte (celui qui contient le fichier *Vagrantfile*), qui est monté dans */vagrant* dans chacune des VMs

Depuis *node2* :

```
vagrant@node2:~$ cp certs/domain.crt /vagrant/
```

Depuis *node1* :

```
vagrant@node1:~$ sudo mkdir /etc/docker/certs.d/registry.mydom.com:5000
vagrant@node1:~$ sudo cp certs/domain.crt
/etc/docker/certs.d/registry.mydom.com:5000/ca.crt
```

Depuis le node1, essayez une nouvelle fois de pusher l'image *registry.mydom.com:5000/nginx:1.14*.

```
vagrant@node1:~$ docker image push registry.mydom.com:5000/nginx:1.14
The push refers to repository [registry.mydom.com:5000/nginx]
a988eeefbe01: Pushed
e53052eb0904: Pushed
6744ca1b1190: Pushed
1.14: digest:
sha256:9ed9873861f42054ea621811575ae06d4001ca6cc9a0f351de194e7253746e89 size:
948
```

Cette fois-ci l'image peut être pusher car le daemon Docker fait confiance au certificat.

Notes:

- dans le cas d'utilisation d'un certificat auto-signé, il sera donc nécessaire d'ajouter le certificat à chaque daemon Docker qui a besoin d'intéragir avec le registry. Cela ne sera pas nécessaire si le certificat est signé avec une authorité de certification (CA) reconnue, le certificat de celles-ci étant déjà ajoutés dans la configuration du daemon Docker
- dans cet exemple, le registry est déployé sur une machine du LAN. Dans le cas ou le registry devrait être accessible via internet, il faudrait utiliser un certificat signé par une CA reconnue en passant par example par Let's Encrypt

Changement du port d'écoute

Jusqu'à présent, nous avons utilisé le port 5000 pour exposer le registry, c'est le port d'écoute par défaut. Maintenant que le registry écoute sur HTTPS, nous allons utiliser le port 443, cela simplifiera notamment le tag des images.

Sur le node2, relancez le registry en ajoutant la variable d'environment *REGISTRY_HTTP_ADDR* et en exposant le port 443 sur la machine hôte:

```
vagrant@node2:~$ docker rm -f registry
vagrant@node2:~$ docker run -d \
```

```
  --restart=always \
  --name registry \
  -v "$(pwd)"/certs:/certs \
  -e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt \
  -e REGISTRY_HTTP_TLS_KEY=/certs/domain.key \
  -e REGISTRY_HTTP_ADDR=0.0.0.0:443 \
  -p 443:443 \
  registry:2
```

Sur le node1, renommez le répertoire contenant le certificat du registry.

```
vagrant@node1:~$ sudo mv registry.mydom.com\:5000/ registry.mydom.com
```

Il est maintenant possible de tagger les images sans le numéro de port, la communication utilisant le port 443 par défaut. C'est ce que vous pourrez observer en taggant l'image de la façon suivante:

```
vagrant@node1:~$ docker tag nginx:1.14 registry.mydom.com/nginx:1.14
```

Vous pourrez alors la pusher dans le registry

```
vagrant@node1:~$ docker image push registry.mydom.com/nginx:1.14
The push refers to repository [registry.mydom.com/nginx]
a988eeefbe01: Pushed
e53052eb0904: Pushed
6744ca1b1190: Pushed
1.14: digest:
sha256:9ed9873861f42054ea621811575ae06d4001ca6cc9a0f351de194e7253746e89 size:
948
```