Lab 1 – Préparation des machines Labs

Etapes

- Installez VirtualBox installateurs disponibles ici : https://www.virtualbox.org/wiki/Downloads
- Installez Vagrant disponibles ici: https://www.vagrantup.com/downloads.html
- cd dans puppet-training/
- Nous installons un plugin Vagrant pour gérer les VirtualBox guest additions
- Puis vagrant up puppet pour provisionner la VM

```
cd devops-iac/puppet
vagrant plugin install vagrant-vbguest --plugin-version=0.21
vagrant up
```

Arrêt de votre VM gérée par Vagrant

Lorsque vous avez terminé de travailler dans un lab en particulier et que vous souhaitez arrêter votre VM, exécutez la commande vagrant halt suivie du nom de la VM, comme suit:

```
vagrant halt puppet        # and wait for VM to shutdown
```

Démarrage de votre VM gérée par Vagrant

Si vous avez exécuté vagrant halt puppet pour tester l'arrêt, vous pouvez exécuter vagrant up puppet avant de continuer pour la redémarrer ...

Si tout se passe bien avec ce qui précède, nous pouvons être sûrs que notre VM fonctionne comme prévu et passons à autre chose.

Connectez-vous à nouveau à votre VM ...

Reconnectons-nous à la machine virtuelle Puppet avec vagrant ssh puppet

Explorez plus de commandes Vagrant

Familiarisez-vous avec Vagrant et connectez-vous à votre VM. Les commandes les plus utiles dont vous aurez besoin sont:

```
vagrant up <name>          # start the VM, and provision it if it's the first time
vagrant halt <name>        # shutdown the VM cleanly
vagrant global-status      # show the status of your VMs (powered on or off)
vagrant destroy <name>     # if you want to completely destroy your VM and start
over
vagrant ssh <name>         # connect to your VM with ssh
vagrant ssh-config <name>  # show ssh config in case you want to use PuTTY to
connect to your VM
```

Provisionnez deux machines virtuelles supplémentaires

Vous devrez également provisionner 2 VM supplémentaires, une pour GitLab et une de plus qui exécutera l'agent puppet . (En fait, les 3 exécuteront la Puppet agent, mais nous créons une petite machine virtuelle distincte pour les tests afin de ne pas avoir s'inquiéter de casser quelque chose pendant que nous testons.)

```
vagrant up gitlab     # provision the VM for your GitLab server
vagrant up agent      # provision your VM that will be one of your puppet agents
```

Une fois que vous avez provisionné vos VM gitlab et agent, vous devriez les voir dans la sortie de vagrant global-status

```
$ vagrant global-status
id       name         provider   state    directory
------------------------------------------------------------------------------
4dd5ed7  puppet       virtualbox running  /Users/Mark/Vagrant/puppet-training
070258c  gitlab       virtualbox running  /Users/Mark/Vagrant/puppet-training
682eafe  agent        virtualbox running  /Users/Mark/Vagrant/puppet-training
```

Validez votre environnement de formation

Nous avons utilisé Vagrant pour provisionner 3 machines virtuelles d'entraînement. Si vous êtes curieux de savoir comment Les VM sont configurées, jetez un œil au puppet-training/Vagrantfile

Nous avons utilisé une boîte de base personnalisée qui contient déjà certains éléments dont nous avons besoin (packages, configuration, etc.) Cette boîte de base personnalisée a déjà des entrées dans /etc/hosts que nous voulons/avons besoin (puisque nous allons comptez sur/etc/hosts pour résoudre les noms dans notre environnement de formation.)

Chaque VM doit avoir les contnets suivants dans leur fichier /etc/hosts:

```
127.0.0.1       localhost
192.168.198.10  puppet.example.com  puppet
192.168.198.11  agent.example.com   agent
192.168.198.12  gitlab.example.com  gitlab
```

Avec les 3 de vos machines virtuelles opérationnelles, vérifiez que vous pouvez envoyer un ping à chaque FQDN ainsi que le nom court.

Par exemple:

```
[vagrant@puppet ~]$ ping puppet.example.com
PING puppet.example.com (192.168.198.10) 56(84) bytes of data.
64 bytes from puppet.example.com (192.168.198.10): icmp_seq=1 ttl=64 time=0.043 ms
64 bytes from puppet.example.com (192.168.198.10): icmp_seq=2 ttl=64 time=0.098 ms
64 bytes from puppet.example.com (192.168.198.10): icmp_seq=3 ttl=64 time=0.075 ms
^C
--- puppet.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.043/0.072/0.098/0.022 ms
```

```
[vagrant@puppet ~]$ ping puppet
PING puppet.example.com (192.168.198.10) 56(84) bytes of data.
64 bytes from puppet.example.com (192.168.198.10): icmp_seq=1 ttl=64 time=0.052 ms
64 bytes from puppet.example.com (192.168.198.10): icmp_seq=2 ttl=64 time=0.088 ms
64 bytes from puppet.example.com (192.168.198.10): icmp_seq=3 ttl=64 time=0.098 ms
^C
--- puppet.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.052/0.079/0.098/0.021 ms
```

Vous devriez également pouvoir effectuer des ssh entre chaque VM en tant qu'utilisateur vagrant ou l'utilisateur root. Le mot de passe par défaut est simplement "vagrant". Vous devriez également pouvoir effectuer une connexion SSH vers l'une de vos VM à partir de l'hôte elle-même.