CONFIGURER VOTRE ENVIRONNEMENT ANSIBLE

Introduction

Dans ce chapitre, nous découvrirons **comment configurer notre environnement Ansible**. Pour ce faire il nous faut principalement deux types de machines :

- **Machine de contrôle** (ou machine maître) : Machine à partir de laquelle nous pouvons gérer d'autres machines. et pour gérer ces nœuds distants, nous devons installer Ansible sur la machine maître.
- **Machine esclave** : Machines manipulées / contrôlées par une machine de contrôle.

Machine de contrôle

Prérequis et conseils

Actuellement, Ansible peut être exécuté à partir de n'importe quelle machine sur laquelle Python 2 (version 2.7) ou Python 3 (version 3.5 et supérieures) est installé (l'interpréteur python est installé par défaut sur les machines Linux). Cela inclut Red Hat, Debian, CentOS, macOS, n'importe lequel des BSD, etc. Cependant, **Windows n'est pas pris en charge pour le nœud de contrôle**.

Conseil

Lors du choix de votre nœud de contrôle, gardez à l'esprit qu'il vaut mieux que le nœud de contrôle exécute vos playbooks à proximité des machines gérées, car les modules sont lancés à partir du protocole de communication SSH. Par exemple, si vous exécutez Ansible dans un cloud, envisagez de l'exécuter à partir d'une machine à l'intérieur de ce cloud.

Installation d'Ansible

Il existe **différentes façons d'installer le moteur Ansible**. Vous avez le choix entre les méthodes suivantes :

- Installation de la dernière version proposée par votre **gestionnaire de package** de votre OS.
- Installation avec le **gestionnaire de packages python pip**.
- Installation depuis les **sources Ansible** afin d'utiliser et tester les dernières fonctionnalités.

Installer Ansible sur Ubuntu

Commencez d'abord par mettre à jour la liste de vos dépôts APT :

```
sudo apt update
```

Installez ensuite le logiciel software-properties-common qui vous permettra de gérer plus facilement les dépôts indépendants (ppa) de votre distribution.

```
sudo apt install software-properties-common
```

Ajoutez le repository ansible en version stable :

```
sudo apt-add-repository --yes --update ppa:ansible/ansible
```

Enfin, installez le moteur ansible :

```
sudo apt install ansible
```

Installation d'Ansible sur RHEL, CentOS ou Fedora

L'installation se tient sur une ligne de code :

```
sudo yum install ansible
```

Installer depuis pip

Comme dit tout à l'heure, Ansible peut être installé avec le gestionnaire de paquets Python pip. Cette méthode peut être intéressante, quand la dernière version stable n'est pas proposée par le gestionnaire de package de votre OS.

Si pip n'est pas déjà disponible sur votre système Python, exécutez les commandes suivantes pour l'installer:

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
python get-pip.py --user
```

Installez ensuite ansible :

```
pip install --user ansible
```

Les nœuds esclaves

Sur les nœuds esclaves, vous avez besoin d'un moyen de communication SSH, le port SSH doit donc être ouvert. Vous avez également besoin de Python 2 (version

2.6 ou ultérieure) ou Python 3 (version 3.5 ou ultérieure).

Je ne cesserai de vous le dire chaque jour : *"Un bon devopsien, est un informaticien qui automatise ses tâches !"*. Et comme nous sommes malins , nous allons automatiser le provisionnement de nos nœuds esclaves depuis l'outil Vagrant dans l'hyperviseur Virtualbox.

Rendez-vous sur la page d'installation officielle de vagrant et téléchargez le package correspondant à votre système d'exploitation et à votre architecture.

Par exemple sur Fedora 30, il faut choisir le package 64 bits sous Centos. L'installation se passera ainsi comme ça :

```
sudo rpm -Uvh https://releases.hashicorp.com/vagrant/2.2.6/vagrant_2.2.6_x86_64.rpm
```

Pour ubuntu, il faut choisir le package 64 bits sous Debian, soit :

```
curl -O https://releases.hashicorp.com/vagrant/2.2.6/vagrant_2.2.6_x86_64.deb
sudo apt install ./vagrant_2.2.6_x86_64.deb
```

Testez ensuite le bon déroulement de votre installation, en vérifiant la version de vagrant :

```
vagrant --version
```

Résultat :

```
Vagrant 2.2.6
```

Vagrantfile

C'est à partir du fichier Vagrantfile que toute la procédure débute. Ce fichier décrit comment nos nouvelles machines esclaves seront provisionnées. J'ai mis un maximum de commentaires, histoire de comprendre la finalité de chaque

instruction. Voici déjà à quoi ressemble donc notre fichier Vagrantfile :

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
# ##########################################################################
# ################### CONFIGURATION VARIABLES ##############################
# ##########################################################################
IMAGE_NAME = "bento/ubuntu-18.04"   # Image to use
MEM = 2048                          # Amount of RAM
CPU = 1                             # Number of processors
SLAVE_NBR = 2                       # Number of slaves node
NETWORK_ADAPTER="wlp1s0"            # Bridge network adapter


Vagrant.configure("2") do |config|
    # RAM and CPU config
    config.vm.provider "virtualbox" do |v|
        v.memory = MEM
        v.cpus = CPU
    end

    # Slave node config
    (1..SLAVE_NBR).each do |i|
        config.ssh.insert_key = false
        config.vm.define "slave-#{i}" do |slave|
            # OS and Hostname
            slave.vm.box = IMAGE_NAME
            slave.vm.hostname = "slave-#{i}"
            slave.vm.network "public_network", bridge: NETWORK_ADAPTER
        end
    end
end
```

Vous pouvez personnaliser vos machines esclaves depuis le fichier Vagrantfile à partir des variables de configuration. Par exemple vous pouvez agrandir le nombre de nœuds en changeant la variable `SLAVE_NBR`, voire attribuer davantage de ressources à vos nœuds en revalorisant les variables `CPU` et/ou `RAM`.

Pour ce cours, j'ai choisi le mode réseau accès par pont (bridge) afin de faire communiquer mes machines virtuelles totalement entre elles vers l'extérieur via la machine hôte s'approchant. Dans mon cas j'utilise la carte réseau de mon ordinateur `wlp1s0` (lancez la commande `ip a` afin de connaître la votre).

Information

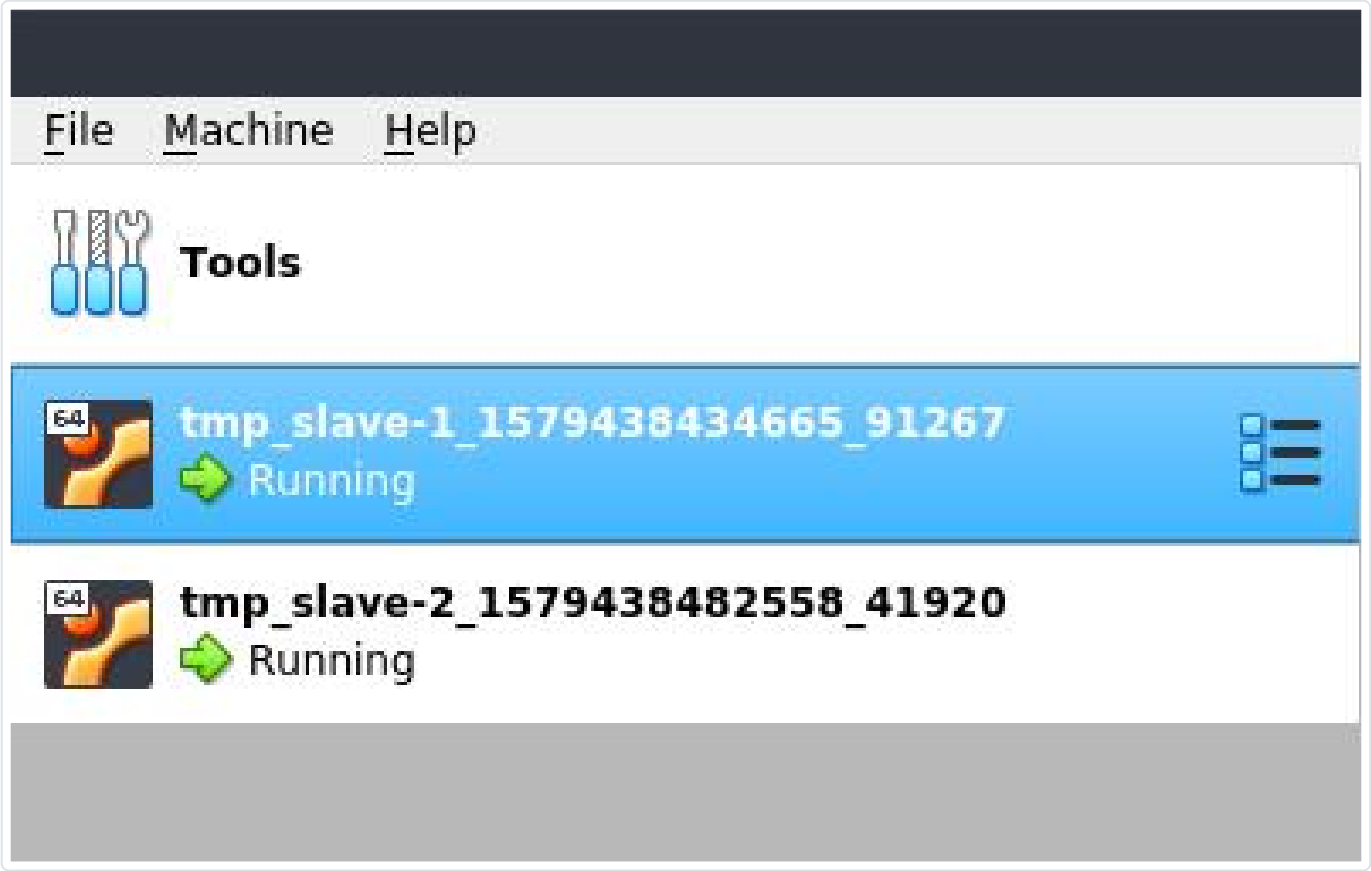
Vous pouvez aussi définir des IPs privées en ajoutant l'option `ip: "MON IP"` après le paramètre `bridge`. Par exemple si vous souhaitez avoir l'ip 192.169.0.21 sur votre nœud distant 1 et l'IP 192.168.0.22 sur votre nœud distant 2, alors vous écrirez alors la ligne suivante : `slave.vm.network "public_network", bridge: NETWORK_ADAPTER, ip: "192.168.0.2#{i}"`

Enfin nous travaillerons sur des machines virtuelles Ubuntu sous sa version 18.04.

Pour provisionner vos deux nouvelles VMs. Ouvrez un terminal et placez vous d'abord au même niveau que le fichier Vagrantfile et lancez ensuite la commande suivante :

```
vagrant up
```

Après la fin de l'exécution, si on retourne sur Virtualbox, on peut visualiser nos deux nouvelles machines :



Conclusion

Après avoir exécuté les instructions ci-dessus, vous êtes prêt à gérer les machines distantes via Ansible. Dans le prochain chapitre suivant nous verrons comment exécuter nos premiers modules sur nos machines esclaves.