Formation Ansible, automatiser la gestion des serveurs
Sujet des travaux pratiques

Matthieu Falce

Mars 2025

1 Ansible

1.1 `Vagrant` pour créer une machine

Objectifs : 1) découvrir `vagrant` 2) découvrir les principes de `ansible` sur une machine

Nous allons utiliser `Vagrant` qui permet de manipuler des machines virtuelles (VM). Cet outil, qui n'a pas de rapport avec Ansible, facilite l'automatisation de différents gestionnaires de machines virtuelles (`VirtualBox` ou `VMWare` par exemple) pour la création et la configuration matérielle des VM.

La façon la plus classique d'utiliser `Vagrant` passe par la création d'un `Vagrantfile` qui va décrire l'architecture matérielle qui nous intéresse. Un exemple de fichier `Vagrantfile` est présent dans le dossier `vagrant_une_machine`. Il crée une machine qui s'appelle `node1`, basée sur le dernier Ubuntu LTS et qui a 1 CPU et 500 Mo de RAM.

Principes de base pour `Vagrant` :

- toujours nommer le fichier de configuration `Vagrantfile`
- un fichier `Vagrantfile` par dossier : pour les différents exercices, vous allez faire un dossier par exercice
- ne pas modifier un `Vagrantfile` sans avoir éteint les machines avant (vous risquez de ne plus y avoir accès facilement)
- on ne se rend pas compte que les machines sont encore allumées, alors quand vous finissez éteignez les avec `vagrant halt`

Cycle de vie des machines :

- créer les machines : `vagrant up`
- se connecter en SSH à une machine : `vagrant ssh [nom de la machine]`
- mettre à jour les machines une fois qu'elles ont été crées : `vagrant up --provision`
- éteindre les machines : `vagrant halt [nom de la machine]`
- détruire les machines : `vagrant destroy [nom de la machine]`

Il peut y avoir des problèmes de configuration du réseau des machines. Cela est dû à une mise à jour de `virtualbox 6.1.26` qui modifie les adreses IP acceptées... Pour régler cela [1], il faut modifier le fichier `/etc/vbox/networks.conf` et y mettre les lignes suivantes (asterisque inclue) :

```
* 10.0.0.0/8 192.168.0.0/16
* 2001::/64
```

Étapes :

1. copier le `Vagrantfile` et démarrer la machine
2. créer un inventaire ansible contenant la machine, comment connaître son IP ?
3. lancer une commande ansible *ad hoc* pour savoir quelle est son OS, IP, hostname, RAM et disque dur disponibles. Cela semble-t-il cohérent avec ce que nous avons indiqué à Vagrant ?

1. Source de la solution : `https://stackoverflow.com/questions/70704093/the-ip-address-configured-for-the-host-only-network-is-not-within-the-allowed-ra`

1.2 Architecture 3 tiers

Objectifs : 1) découvrir les principes de `ansible` sur plusieurs machines 2) utiliser des playbooks 3) utiliser des groupes de machines

Nous allons simuler une infrastructure classique 3 tiers. Ce sont des infrastructures classiques utilisées en développement web, où l'on sépare la couche de présentation (le navigateur), de la couche de traitement (le serveur web), de la couche d'accès aux données (le serveur de base de données).

Pour cela, nous devons devoir modifier le `Vagrantfile` pour créer des machines qui seront un serveur web et une de base de données.

Étapes :

1. exporter la configuration réseau `Vagrant` pour créer l'inventaire ansible
2. créer un groupe pour les machines de base de données et un pour les serveurs web

Sur toutes les machines, utiliser `ansible` (avec et sans création de rôles) pour :

— installer `vim`, `zsh` (un shell différent)

— créer un utilisateur appelé `app` ayant les droits sudo (il doit appartenir au groupe `sudo`), lui mettre le shell `zsh`. Comment gérer le mot de passe du compte ?

— déposer le fichier `/home/app/.zshrc` permettant de configurer le shell

— ajouter la ligne `alias lt='ls --human-readable --size -1 -S --classify'` au `/home/app/.zshrc`

Sur le serveur web :

— installer `nginx`

— tester la connexion (`Vagrant` est configuré pour que le port 80 des machines web soit sur le port 9898 de la machine hôte)

— rajouter une page web statique (un fichier HTML) avec le `hostname` de la machine [2]

— faire en sorte que le serveur web soit lancées au démarrage comme un service (ou un `deamon`)

Sur les serveurs de base de données

— installer `mariaDB` (le remplaçant de `MySQL`) et `mongodb` [3]

— tester la connexion

— faire en sorte que les bases de données soient lancées au démarrage comme un service (ou un `deamon`)

Étapes :

1. le relancer sur les deux machines. Que se passe-t-il ? Le `.zshrc` a-t-il la dernière ligne dupliquée ? Si oui, pourquoi et comment le régler ?
2. supprimer une des machines virtuelles avec `vagrant destroy web1`
3. re-créer l'instance `vagrant up`
4. re-lancer le playbook. Que se passe-t-il ?

Sur la machine de contrôle, installer le module `cowsay`. Que fait-il ? Relancer le `playbook`, que se passe-t-il ?

Questions :

1. quand utiliser des rôles ? Quel est leur avantage ?
2. quelle est l'importance de l'idempotence dans `ansible` ?
3. quand utiliser le module `template` par rapport au module `copy`
4. comment ne plus avoir les vaches lors des déploiements ?

2. regarder ici `https://www.digitalocean.com/community/tutorials/how-to-set-up-nginx-server-blocks-virtual-hosts-on-ubuntu-16-04`

3. en regardant comment faire ici : `https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/`

1.3 Gestion de l'inventaire

Objectifs : 1) manipuler les hôtes de l'inventaire 2) gérer des inventaires plus complexes

Nous allons simuler une infrastructure 3 tiers plus complexe, car elle va contenir plus de machines (3 serveurs web et 3 serveurs de base de données).

Pour cela, nous devons devoir modifier le `Vagrantfile` pour créer des machines qui seront un serveur web et une de base de données.

Étapes :

1. mettre à jour l'inventaire avec les nouvelles machines
2. créer un playbook qui lance des commandes de mise à jour sur toutes les machines
3. lancer l'exécution du playbook.
4. essayer de manipuler l'inventaire et les groupes (par exemple, créer des machines à Paris et à Dublin), n'essayer que de toucher certaines de ces machines.
5. relancer le playbook précédent sur les machines et tester les différents types de stratégie de déploiement (1 machine à la fois, `free strategy`), testez en mettant des caractéristiques différentes (RAM, CPU) aux machines

Questions :

1. comment devons-nous modifier le `Vagrantfile` ?
2. Que se passe-t-il quand on essaie de lancer une commande sur les 6 machines en même temps ? Comment faire en sorte d'exécuter toutes les machines en parallèle ?
3. comment vérifier quelles machines sont concernées par nos groupes (c'est-à-dire avoir la liste des machines qui seront ciblées) ?
4. quel est l'impact de la `strategy` sur les performances ?

1.4 Projet – Installation de wordpress

Objectifs : 1) gérer une installation réelle complexe du début à la fin

Nous nous mettons dans la peau d'un DevOps chez un grand compte de l'hébergement. La plupart de nos revenus proviennent de l'hébergement de sites `wordpress`, que nous proposons *as a service*. Comment pouvons-nous automatiser ce processus ?
Au départ nous allons installer Wordpress de façon monolithique (toutes les dépendances sur la même machine) puis, nous allons séparer la base de donnée de la partie web, puis nous mettrons plusieurs machines web pour avoir un système de haute disponibilité.

Étapes :

1. regarder des tutoriels d'installation de wordpress et le faire à la main
2. créer un playbook d'installation de wordpress sur une seule machine
3. passer la base de données sur une VM à part

Une fois que cela fonctionne, nous pouvons essayer de passer Wordpress en Haute Disponibilité (2 instances Wordpress servies derrière un HAProxy).

Étapes :

1. dans Vagrant ajouter une nouvelle machine pour le HAProxy et pour l'autre serveur Wordpress
2. avec Ansible, installer le logiciel HAProxy et le configurer pour servir les machines Wordpress
3. installer l'autre machine Wordpress
4. modifier le playbook pour effectuer du déploiement Haute Disponibilité (notifier HAProxy pour sortir et re-rentrer l'hôte en cours de déploiement, ne déployer qu'une machine à la fois)