IVC-Vagrant-2 : Gestion de configuration avec Ansible

Lom M. HILLAH

Table des matières

1 Introduction **3**

1.1 Fichiers sources pour cet atelier 3

1.2 Objectifs 3

1.3 Conventions 4

1.4 Pré-requis 4

2 Présentation d'Ansible **5**

2.1 Outils support 6

3 Créer un inventaire **6**

4 Modules **6**

5 Playbook **7**

5.1 Validation des playbook 8

5.2 Validation des tâches 8

5.3 Clé SSH 8

6 Installer un logiciel **9**

6.1 Démarrer un logiciel 9

7 Copier des fichiers 9

8 Handlers et notification 10

9 Liens symboliques 10

10 Créer des fichiers 11

11 Cron Jobs 11

12 Commandes utilisateur à distance 12

13 Créer des utilisateurs et des groupes 12

14 Les rôles 12

15 Vagrant avec Ansible 12

15.1 Documentation 13

15.2 Spécifier un gestionnaire de configuration dans le Vagrantfile . . . 13

16 Feedback 14

17 Licence 14

1 Introduction

Atelier pratique sur l'utilisation de Ansible pour automatiser la gestion de configuration d'environnement virtuel, en liaison avec Vagrant.

1.1 Fichiers sources pour cet atelier

Tous les fichiers sources utiles pour réaliser les travaux dans cet atelier sont fournis dans l'archive ou le répertoire que vous avez téléchargé, qui contient le présent document.

1.2 Objectifs

L'atelier Vagrant-1 explique comment créer et gérer des environnements virtuels avec Vagrant. La machine virtuelle (VM) seule ne suffit pas ; il faut la configurer pour le projet qui nous intéresse en y installant tous les logiciels nécessaires au travail dans le projet. Il peut s'agir par exemple d'installer Java, PHP, installer et démarrer des serveurs (p. ex., Apache, Nginx), gérer des utilisateurs, programmer des tâches cron, gérer des fichiers, etc.

Dans cet atelier, nous utilisons Ansible pour automatiser ces tâches. Ansible est un outil d'automatisation de la gestion de configuration d'infrastructure informatique (**IT automation tool** en anglais). Il existe aussi dans cette catégorie : Puppet et Chef, pour citer les plus répandus.

Le programme est le suivant :

- Présentation d'Ansible.
- Utilisation d'Ansible pour :
  - installer des logiciels,
  - gérer des fichiers,
  - gérer des tâches cron,
  - lancer des commandes,
  - gérer des utilisateurs et des groupes.
- Utilisation des rôles Ansible créés par d'autres développeurs.
- Lancement d'Ansible via ligne de commande pour configurer une machine.
- Gestion de la configuration d'une machine Vagrant avec Ansible.

1.3 Conventions

Nous utiliserons la ligne de commande dans un terminal. Le prompt sera symbolisé par :

```
$>
```

Les configurations seront indiquées dans une police monofont comme dans l'exemple suivant :

```
config.vm.synced_folder "/Users/aUser/deployed-assets/", \
"/var/www/assets", disabled: true
```

Dans l'exemple ci-dessus, le caractère \ indique que la configuration est écrite sur une seule ligne. Le passage à la ligne est employé pour améliorer la lisibilité et éviter la troncation de la ligne par la marge.

1.4 Pré-requis

Vous devez avoir suivi l'atelier Vagrant-1.

Dans le cas nominal, Ansible n'a pas besoin d'installation spécifique côté serveur (machine virtuelle - invitée) pour être mis en oeuvre, mais pour l'utiliser il faut l'installer sur le poste de contrôle (machine hôte). Malheureusement, il ne peut être exécuté sur une machine Windows, même s'il peut être utilisé pour gérer un serveur Windows.

Alternativement, vous pouvez l'installer à distance dans la machine à administrer (invitée), puis l'exécuter à partir de là, après y avoir déployé le playbook (que nous étudierons plus loin) par vos propres moyens (par exemple via les répertoires synchronisés par Vagrant).

Pour continuer cet atelier sous Mac OS ou Linux, installez Ansible et assurez-vous qu'il est accessible dans votre `$PATH` sur la ligne de commande, en tapant par exemple :

```
$> which ansible-playbook
```

- https://docs.ansible.com/ansible/latest/installation_guide/index.html
- https://releases.ansible.com/ansible/

Sur Mac OS, Ansible est disponible à partir de MacPorts. Cependant, la méthode préférée à partir des versions ultérieures à 2.9 est d'utiliser pip. Il faut d'abord s'assurer que vous disposez de python 3 (ex., 3.8 ou 3.9).

```
$> python -m pip install --user ansible "ansible-lint[yamllint]"
```

2 Présentation d'Ansible

Ansible est un outil permettant d'automatiser la gestion de configuration, l'administration de serveurs, le déploiement d'applications et l'exécution de tâches distantes. Il n'est pas nécessaire de l'installer sur les serveurs à administrer, mais il doit être installé sur la machine de contrôle. Ansible est **rootless** et **agentless.**

Dans le cas d'utilisation avec Vagrant, vous pouvez l'installer sur la machine hôte, mais également le déployer directement sur la machine guest via un provisionner comme le Shell, puis l'exécuter à partir de là.

Les configurations pour Ansible sont écrites dans des fichiers YAML, ce qui les rend assez concises, simples à lire et écrire. Une configuration pour Ansible est décrite dans un **playbook**. Un **playbook** est constitué de **plays** qui à leur tour sont constitués de tâches (**tasks**).

Les tâches utilisent des modules Ansible pour effectuer les actions à réaliser. Un module est un plugin spécialisé qui permet de réaliser des tâches liées à un domaine (ex. command pour exécuter des lignes de commande, dnf, etc.). La liste des modules disponibles est publiée dans l'index des collections sur le site de la documentation. En particulier, les modules natifs d'Ansible sont listés dans la collection Ansible.Builtin

Dans un contexte plus avancé, la réutilisation peut être mise en oeuvre à travers la notion de **rôles**, de telle sorte qu'il devient encore plus simple de réutiliser des rôles existants pour configurer des serveurs.

Il existe un dépôt de rôles dans Ansible Galaxy Vous y trouverez par exemple un rôle pour installer Sonarqube : https://galaxy.ansible.com/lrk/sonarqube.

2.1 Outils support

Une liste d'outils de développement supportant la syntaxe ou la validation de playbook Ansible est disponible sur le site de la documentation

3 Créer un inventaire

L'inventaire indique les adresses et les détails de connexion aux machines à administrer. Ansible utilise cet inventaire pour se connecter aux machines, vérifier leur statut et y exécuter les tâches de configuration.

Au minimum, un inventaire (dans un simple fichier texte) contient un nom d'hôte et une adresse IP. La configuration suivante est fournie dans le fichier `inventory.txt` disponible parmi les fichiers sources fournis pour cet atelier.

Pour se documenter sur ce qu'est un inventaire :

- https://linuxbuz.com/linuxhowto/what-is-ansible-inventory-and-how-it-works
- https ://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html

```
default ansible_ssh_host=127.0.0.1 ansible_ssh_port=2222 \
 ansible_ssh_user=vagrant \
 ansible_ssh_private_key=vagrant_id_rsa
```

La configuration ci-dessus indique le nom d'une machine à administrer, ainsi que les détails de connexion à celle-ci.

Il vous est fourni une paire de clés publique/privé à employer pour permettre à Ansible de se connecter en SSH à la VM pour réaliser les tâches d'administration. Vous pouvez en régénérer d'autres en remplacement de celles-là. Dans tous les cas, il faudra installer la clé publique dans la VM par vos propres moyens (`vagrant ssh`). Ce point est abordé en détail plus loin dans la section **Clé SSH**.

4 Modules

Les modules Ansible permettent de mettre en oeuvre les tâches programmées dans le playbook. Quelques modules sont :

- **apt** ou **dnf, pour gérer les packages apt*, ou fedora
- **git**, pour utiliser des dépôts Git
- **service**, pour gérer des services s'exécutant sur le serveur à administrer
- **copy** pour copier des fichiers.
- **command** pour exécuter des lignes de commande
- **shell** pour exécuter des script shell
- etc..

Cf. les modules built-in de Ansible.

5 Playbook

Un playbook pour Ansible est un fichier YAML contenant l'ensemble des plays (constitués de tâches) à jouer pour réaliser le travail de configuration demandé. L'indentation dans ce fichier est crucial, car sinon il sera mal parsé ce qui causera une erreur rapportée par Ansible.

Par exemple, si nous voulons mettre à jour le gestionnaire de packages `apt` sur le serveur (ce qui est une bonne pratique dès qu'on souhaite installer des logiciels via ce gestionnaire) :

```
---
- hosts: all

  user: vagrant
  sudo: yes

  tasks:
  - name: update apt cache if older than 1 hour
    apt: update_cache=yes cache_valid_time=3600
```

Les tâches sont décrites selon **l'objectif qu'elles doivent remplir**, pas comment elles doivent le réaliser.

La configuration ci-dessus peut être appliquée en lançant en ligne de commande :

```
$> ansible-playbook -vvvv MASTER-playbook.yml -i inventory.txt
```

L'option `-vvvv` servira à afficher le log de la commande SSH.

5.1 Validation des playbook

Ansible Lint est l'outil recommandé pour valider les playbook.

5.2 Validation des tâches

Ansible fournit deux options pour valider les tâches : * –check : n'applique aucun changement aux systèmes à administrer, mais indique les changements qui seraient appliqués ; * –diff : applique les changements aux systèmes à administrer et rapporte les changements effectués (before / after). Le mode diff peut être utilisé en combinaison avec check. Dans cette configuration cela rapporte les changements qui auraient été appliqués. Le mode diff est très verbeux !

5.3 Clé SSH

Si Ansible vous indique une erreur de connexion SSH par rapport à la clé d'authentification, vous pouvez générer une autre paire de clés publique/privée dans le répertoire courant :

```
$> ssh-keygen -t rsa -f vagrant_id_rsa
```

Connectez-vous ensuite à la VM avec `vagrant ssh`. Puis, rajoutez la clé publique (extension `.pub`) dans le fichier `.ssh/authorized_keys` dans le home directory de vagrant dans la VM :

```
$> cat vagrant_id_rsa.pub >> ~/.ssh/authorized_keys
```

Déconnectez-vous ensuite de la VM avec `Ctrl-D`. Il faut ensuite rajouter votre clé privée à `ssh-agent` :

```
$> ssh-agent && ssh-add vagrant_id_rsa
```

Enfin, relancez la commande `ansible-playbook`. Vous verrez le rapport d'exécution d'Ansible au fur de la réalisation des tâches. À la fin de ce rapport, Ansible effectue un récapitulatif de toutes les tâches réalisées avec succès, des tâches qui ont été exécutées suite à un changement dans la configuration depuis la dernière fois, des tâches inaccessibles et des tâches qui ont échoué.

6 Installer un logiciel

Pour installer Nginx par exemple :

```
- name: ensure that nginx is installed
    apt: pkg=nginx state=present update_cache=yes
```

Cette configuration demande à Ansible d'installer le package `nginx` s'il n'est pas présent, tout en ayant préalablement mis à jour le cache du gestionnaire de package apt.

Il est important de noter à ce stade qu'Ansible est **idempotent**. Cela signifie que chaque fois qu'Ansible s'exécute, il ne réalisera les actions de configuration que si quelque chose a changé dans le serveur. Par exemple, si Nginx fut installé lors d'une précédente exécution, une prochaine exécution n'installera plus Nginx car il fut déjà installé.

Les autres outils principaux d'automatisation de la gestion de configuration (p. ex. Puppet, Chef) sont également idempotents.

6.1 Démarrer un logiciel

Pour démarrer le serveur Nginx installé par la tâche précédente, on utilisera le module **service** :

```
- name: ensure that nginx is running
  service: name=nginx state=started enabled=yes
```

7 Copier des fichiers

Le module **template** vous sera très utile pour gérer des fichiers, y compris positioner les droits d'accès et les possesseurs :

```
- name: write nginx configuration file
  template: >
    src=nginx-config-file
    dest=/etc/nginx/sites-available/default.conf
    owner:www-data
```

```
    group=www-data
```

Cette configuration copie le fichier local `nginx-config-file` de la machine hôte vers le fichier de destination spécifié par le paramètre `dest` dans la VM.

Notez que la configuration ci-dessus dans `template` peut être écrite sur une seule (et longue) ligne.

8 Handlers et notification

Suite à l'exécution d'une tâche vous pouvez notifier un handler, qui ne sera activé que si la tâche est terminée. Par exemple, nous pouvons nous assurer que le service nginx est redémarré après avoir copié son fichier de configuration :

```
- name: write nginx configuration file
  template: >
    src=nginx-config-file
    dest=/etc/nginx/sites-available/default.conf
    owner:www-data
    group=www-data
    mode=0644
  notify: restart nginx
- name: ensure that nginx is running
  service: name=nginx state=started
handlers:
  - name: restart nginx
    service: name=nginx state=restarted
```

9 Liens symboliques

La création de liens symboliques s'effectue à l'aide du module **file**. Par exemple, si nous souhaitons lier un répertoire public d'un serveur web au répertoire synchronisé par Vagrant (`/vagrant`) avec la machine hôte :

```
- name: the vagrant synced folder becomes web root
  file: >
    src=/vagrant
```

```
    dest=/var/www/site
    owner=www-data
    group=www-data
    state=link
```

10 Créer des fichiers

La création de fichier est également simple :

```
- name: create a folder for uploads
  file: >
    path=/var/www/uploads
    owner=www-data
    group=www-data
    mode=0777
    state=directory
```

11 Cron Jobs

Le module `cron` permet de gérer des tâches cron. Par exemple :

```
- name: cron job for web server
  cron: >
    name="cron_web"
    hour="3-4"
    minute="15,30"
    job="/usr/bin/php /vagrant/web_cron.php"
```

La configuration ci-dessus fournit le nom de la tâche cron, la commande à lancer et la programmation de la fréquence. La programmation de la fréquence d'exécution de la tâche cron peut utiliser les éléments suivants :

- **hour** : entre 0 et 23 inclus
- **minute** : entre 0 et 59 inclus
- **month** : entre 1 et 12 inclus
- **day** : entre 1 et 31 incluse
- **weekday** : entre dimanche (0) et samedi (6).

12 Commandes utilisateur à distance

Pour lancer des commandes utilisateur à travers le terminal sur le serveur administré, les modules **`command`** et `shell` sont à votre disposition. Le module `shell` lancera la commande à travers le shell sur le système distant. Il permet d'avoir accès aux variables d'environnement et aux opérateurs tels que `&`, `|`, `>`, et<.

13 Créer des utilisateurs et des groupes

```
- name: create a new user group
  group: name=anewgroup state=present
```

```
- name: create a new user in group anewgroup
  user: name=anewuser group=anewgroup state=present
```

14 Les rôles

Un rôle est un ensemble de configurations spécifiques déjà écrites pour un objectif global donné (par exemple installer et configurer entièrement un serveur web). Ce sont des configurations réutilisables (recettes), publiés dans la galaxie d'Ansible : https ://galaxy.ansible.com/.

Vous y trouverez par exemple un rôle pour déployer sur Ubuntu le serveur de gestion de la qualité de code SonarQube, ou encore un autre pour installer le serveur de gestion des artéfacts Maven, Nexus-OSS. Cela vaut la peine d'y créer un compte, et d'y faire un tour pour rechercher un rôle correspondant à votre besoin avant de l'écrire vous-même si vous n'en trouvez pas.

Vous pouvez examiner ces rôles, les reprendre et les modifier pour répondre à votre besoin, sans nécessairement partir de rien.

15 Vagrant avec Ansible

Il est intéressant d'utiliser un outil d'automatisation de la gestion de configuration tel qu'Ansible (ou Pupppet ou Chef) pour gérer une VM contrôlée par

Vagrant. Le gain en productivité est important, et la reproduction à l'identique des environnements virtuels ainsi créés et entièrement configurés est facilitée.

Chaque membre d'une équipe de développement disposera ainsi sur sa machine du même environnement, avec un partage des fichiers communs à l'ensemble du projet via SVN ou GIT. Toute mise à jour de la configuration pourra être répercutée immédiatement auprès de chaque environnement individuel juste avec `git pull` et `vagrant up --provision` (ou `vagrant provision` si la VM est déjà en train de tourner). Les nouveaux venus dans l'équipe seront intégrés très rapidement dans le projet juste avec `git clone` et `vagrant up --provision`.

15.1 Documentation

- https ://www.vagrantup.com/docs/provisioning/ansible_intro
- https ://www.vagrantup.com/docs/provisioning/ansible
- https ://www.vagrantup.com/docs/provisioning/ansible_local

15.2 Spécifier un gestionnaire de configuration dans le Vagrantfile

Pour indiquer Ansible comme le gestionnaire de configuration de la machine Vagrant, il faut établir la configuration suivante dans le `Vagrantfile`, à l'intérieur de `Vagrant.configure(2)do |config|` :

```
config.vm.provision "ansible" do |ansible|
    ansible.playbook = "path/to/the/playbook.yml"
    ansible.inventory_path = "inventory.txt"
end
```

Comme indiqué dans la section Pré-requis de cet atelier, Ansible est principalement installé sur la machine de contrôle d'où il est exécuté. Alternativement, vous pouvez également l'installer à distance dans la machine à administrer, et l'exécuter à partir de celle-là.

Il est possible de spécifier plusieurs gestionnaires de configuration dans le `Vagrantfile` par ordre de préférence. Par exemple, la configuration suivante indique le shell comme gestionnaire de configuration, ce qui permet d'utiliser un script pour configurer soi-même la machine à administrer :

```
config.vm.provision :shell, :path => "bootstrap.sh"
```

16 Feedback

Lom M. Hillah (lhillah@parisnanterre.fr)

17 Licence