**Ansible**

# Introduction

Ansible est un outil qui permet – entre autre choses – d’automatiser l’installation, le déploiement et la gestion de serveurs.

Mettre en place un serveur manuellement peut rapidement s'avérer pénible, surtout si vous avez plusieurs machines à configurer de la même manière. Pas mal de personnes se sont penchées sur cette problématique et proposent des solutions pour automatiser le déploiement. Je vous propose de découvrir l'une d'entre elles aujourd'hui : [Ansible](https://www.ansible.com).

Ansible permet de décrire la configuration à appliquer à nos différents serveurs au travers de recettes écrites en YAML. Ces recettes contiennent une série de taches qui seront lancées les unes après les autres. Les taches utilisent des modules internes à Ansible qui permettent de décrire les opérations à effectuer et les conditions de lancement.

Ansible is a configuration management tool

Ansible was first released by Michael DeHaan in 2012 as a small side project, and it has had a meteoric rise in popularity, with over 17,000 stars and 1,410 unique contributors on Github.

Beyond being a successful open-source project, it has been successfully used in enterprise by companies like Apple and NASA, who rely on it for their configuration management needs.

Ansible permet de créer des « Playbooks », qui ne sont autre que des scripts à la sauce Ansible, et permettent de configurer vos serveurs.

Sa grande force est qu’il est *agentless*, autrement dit, rien n’est à placer sur vos serveurs. Vous installez Ansible sur votre laptop par exemple, et le tour est joué. Vous pouvez ensuite lancer l’installation de vos 40 serveurs de base de données en une seule commande !

Ansible ne peut fonctionner sur un windows.

Configuration management was developed by the US Department of Defense in the 1950s

as a method of keeping track of hardware material items.A log is kept of every change made to a system along with documentation about who made the change, when the change was made, and why it was made. This allows us to know the exact state of a system at any moment in time.

Ansible uses YAML files as its main source of information at run time. YAML is a data

representation language that is commonly used for configuration.

**Qu’est ce que YAML ?**

Le nom YAML veut dire “YAML Ain’t Markup Language”, soit “YAML n’est pas un langage de balises”. Si cela met d’emblée des distances avec XML, cela ne nous dit pas ce qu’est YAML. YAML est un langage de sérialisation de données conçu pour être lisible par des humains et travaillant bien avec les langage de programmation modernes pour les tâches de tous les jours.

Concrètement, on pourrait noter la liste des ingrédients pour un petit déjeuner de la manière suivante :

- croissants

- chocolatines

- Lait

- oeufs

Ansible is written entirely in Python. The main runner and all of the modules are in Python

Ansible works by running commands via SSH, so there’s no need to install any server software.

Puppet and Chef are most commonly used today than Ansible.

Ansible more closely resembles a tool called SaltStack (Salt), which also uses YAML files for configuration and is also written in Python.

# Lecture: Ansible Introduction and Architecture

Is a tool that can execute tasks on remote systems. It allows for agentless system configuration and deployment.

The power of Ansible is that we can install one software package on one computer and from that computer we can install and configure and manage software and system configurations on many different machines as long as we have the SSH client installed and that is a default with Linux. Everything has got SSH, that's how we get to the servers. That's how we interact with them.

# Lecture: Ansible Installation and Configuration

## Installation de EPEL

EPEL ou encore (Extra Package for Entreprise Linux) est un repo qui founit des package additionnels pour les distibution type RedHat, Centos etc...(maintenu par la communauté FEDORA)

En installant EPEL vous aurez un nombre de packages disponibles via votre gestionnaire de paquets yum beaucoup plus importants!

Pour l’installer on tape : yum list epel-release

## Installation

sudo yum install ansible

**Installation d’Ansible on Debian**

apt-get install ansible

Contrairement à d’autres outils de déploiement, Ansible va attaquer les serveurs lui même en SSH. Cependant pour faire cela, nous allons créer une clé publique/privée en local puis envoyer la clé publique sur le serveur sur lequel on désire autoriser Ansible à intervenir :

Création de la paire de clé en local :

ssh-keygen

Envoi de la clé publique sur le serveur sur lequel on désire autoriser Ansible à intervenir :

ssh-copy-id -i id\_rsa.pub login@ip 🡪 sur mac

ssh-copy-id -i ~/.ssh/id\_rsa.pub login@ip 🡪 sur windows

Si tu veux permettre la connexion ssh en root sur la machine cible :

Dans le fichier de config de ssh mettre permitrootlogin yes → vim /etc/ssh/sshd\_config

systemctl restart ssh

Tu auras peut-être à faire ceci : apt install dirmngr

## Configuration

We must open the inventory to configure Ansible

An **inventory** is a list of hosts that ansible manages.

The default inventory is at etc/ansible/hosts

Inventory can be set in ansible.cfg

You can define a bunch number of servers like this: web[01:50], its mean all servers web from web01 to web50.

If your server name is webserver\_qeyss2c.com you can refer to by a quick name like this :

qeyss2 ansible\_host=webserver\_qeyss2c.com

Donc dans le etc/ansible/hosts je vais ajouter mes deux serveurs ainsi:

qeyss2 ansible\_host=qeyss2c.mylabserver.com

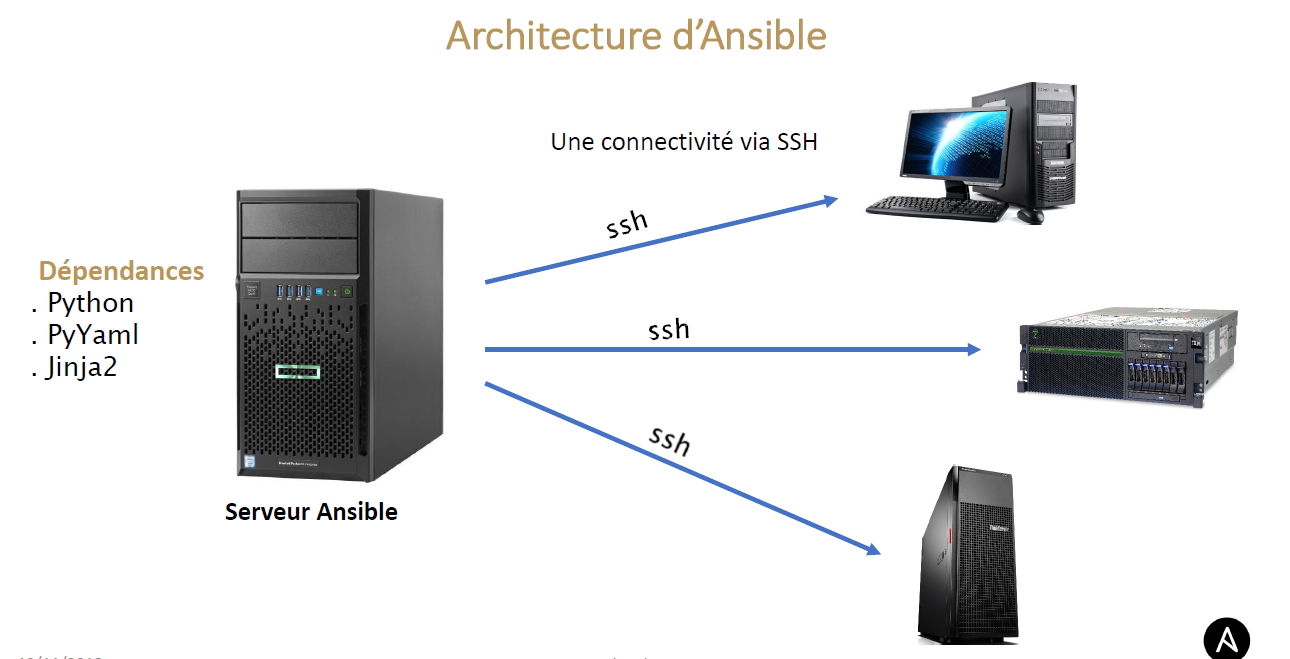
qeyss3 ansible\_host=qeyss3c.mylabserver.com

Normally you use the default inventory in etc/ansible/hosts, but you can set up your inventory file.

- mkdir ansible

- vim inv

Le contrôleur Ansible possède des modules pour effectuer différentes opérations sur le(s) serveur(s) cible auxquels il se connecte en SSH



Au lieu de devoir se connecter à chaque serveur et taper le mot de passe de chaque user, but the common practice is to use an SSH pre-shared key with a designated user for Ansible to connect to your various Ansible managed hosts. Using a pre-shared key prevents you from having to use a password for authentification

First we need to create a user that we are going to designate for ansible purposes

sudo useradd qeyss

Je me connecte en ssh (avec le user de base et non celui créé) au remote server by typing : ssh NameOfServer ; like : ssh qeyss2c.mylabserver.com

Then I create a new user on this server and I give him a password with: sudo passwd qeyss

Then I log off with logout

Then I connect with the new user ( il n’a pas de password, je m’y connecte en tapant: sudo su – Nom) and I generate an ssh key : ssh-keygen

Puis je copie la cle ssh sur le serveur souhaité :

ssh-copy-id qeyss2c.mylabserver.com

Sur le serveur souhaité je me suis connecté en direct avec le user créé pour ansible

Puis je me connecte en ssh à ce serveur à partir du serveur où Ansible est installé :

ssh qeyss2c.mylabserver.com

[**L'utilitaire « sudo »**](https://doc.ubuntu-fr.org/sudo) peut autoriser ou refuser à un utilisateur ou à un groupe d'utilisateur l'exécution de tâches privilégiées avec ou sans saisie d'un mot de passe. Cette gestion des droits accordés aux utilisateurs est consignée dans le fichier /etc/***sudoers***.

La configuration de sudo est enregistrée dans le fichier de configuration */etc/****sudoers***.

La modification de ce fichier s'effectue à travers un utilitaire de vérification appelé **visudo**.

sudo visudo

A cet endroit la : %wheel ALL=(ALL) NOPASSWD: ALL

Tu peux ajouter la même ligne mais pour le user créé à la place de %wheel afin de le mettre parmi les sudo.

# Where to find documentation

<https://docs.ansible.com/>

Ou avec la commande : ansible-doc

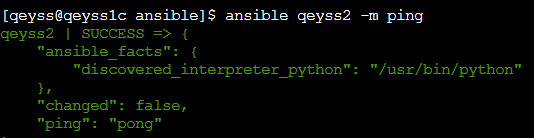
# Lecture : Ad-hoc Ansible Commands

Ansible ad-hoc commands are the foundation of how to work with Ansible.

Le mode ad-hoc signifie en ligne de commande

Les commandes  **ad-hoc** sont des actions rapides qui ne nécessitent pas forcément de les  sauvegarder pour plus tard. Vous pouvez vous référer à [la documentation d'Ansible sur les commandes ad-hoc](https://docs.ansible.com/ansible/latest/user_guide/intro_adhoc.html) pour approfondir.

Lancez maintenant un **ping** avec **Ansible**



L’option :

* **-m** : indique à Ansible d’utiliser un module spécifique, ici ça sera le **m**odule ping ;

Le retour de la commande vous indique que l’action est un succès et répond **pong** au **ping** ! Le node qeyss2 est bien joignable.

Ansible ne lance pas la commande ping, il lance un module qui fait la même chose que la commande ping*.*

Un **module** est un programme utilisé pour exécuter une tâche ou une commande Ansible. Chaque tâche utilise **un module et un seul**, qui peut prendre des arguments pour être exécuté de manière personnalisée. Ansible fournit de nombreux modules, mais vous pouvez créer le vôtre, personnalisé.

Tous les modules sont accessibles sur [la documentation d'Ansible](https://docs.ansible.com/ansible/latest/modules/list_of_all_modules.html) ou avec la commande  $ ansible-doc --list.

Tous les modules officiels d’Ansible sont téléchargés sur votre poste lors de l’installation d’Ansible. Lorsque vous utilisez un module, Ansible ira chercher le code à exécuter dans le dossier du module sur votre poste.

Faire un ping en utilisant l’inventaire que j’ai ajouté :

ansible -i inv -m ping all

Le –i permet ensuite de préciser l’inventaire souhaitée, si tu veux utiliser celui qui est par défaut pas la peine de le mettre.

**Utilisateur non root**

La variable **ansible\_user\_id** permet de récupérer le nom de l’utilisateur avec lequel vous vous connectez (root,qeyss, etc.). Pour lancer cette opération, passez les options suivantes à la commande ansible:

* + 1. -m setup: lancement du module setup;
    2. -a filter=ansible\_user\_id: filtre sur la variable ansible\_user\_id;
    3. -i inventaire : inventaire à utiliser ;
    4. all : nom des machines à sélectionner.

$ ansible -i inv -m setup -a'filter=ansible\_user\_id' all

Si je tape juste inv c’est parce que je suis dans le dossier où il y a ce fichier, sinon il faudrait taper le nom absolu

**Installation de httpd**

On tape la commande suivante :

$ ansible qeyss2 -m yum -a "name=httpd state=present"

Ça ne fonctionnera pas car tu dois être root, donc on la retape ainsi avec –b après avoir ajouté le user qeyss dans le sudo visudo du serveur cible à pouvoir passer des commandes sudo sans password

qeyss ALL=(ALL) NOPASSWD: ALL

$ ansible qeyss2 -b -m yum -a "name=httpd state=latest"

$ ansible qeyss2 -b -m service -a "name=httpd state=started"

Puis tu te connectes en SSH au serveur souhaité et tu tapes la commande suivante :

$ sudo systemctl statuts httpd

### The Sudoers File

This file is the seedy underbelly of sudo.  It controls who can use the sudo command to gain elevated privileges.  It is usually located at /etc/sudoers.  The best and safest way to edit this file is by using the visudo command.

Ajouter le user mike dans le fichier sudoers

Il va falloir ajouter la ligne suivante dans le fichier /etc/sudoers:

mike ALL=(ALL:ALL) NOPASSWD: ALL

Pour cela, vous passerez par le modulelineinfile(option-m lineinfile). Outre le nom du module, vous devrez passer d’autres arguments et notamment:

Les arguments (entre double quote) du module avec l’option-a. Dans ces arguments, vous retrouverez:

Le fichier à modifier avec l’optionpath=/etc/sudoers;

La ligne à ajouter, iciline=’deploy ALL=(ALL:ALL) NOPASSWD: ALL’;

comme vous éditez un fichier sudo, il est possible de le valider avec l’optionvalidate=’visudo -cf %s’;

Comme vu un peu plus tôt, vous allez indiquer que vous souhaitez passer en tant que root avec la méthode su. Il s’agit là de l’option--become-method=su.

L’option--become(-ben version courte) indiquera à Ansible de faire une escalade (par défaut en tant que root).

Enfin, vous allez demander à l’utilisateur de saisir le mot de passe au moment du lancement d’Ansible avec l’option--ask-become-pass(ou -K pour la version courte).

Si vous reprenez toutes ces options, vous allez donc lancer la commande suivante:

**ansible –i inv -m lineinfile\**

**-a "path=/etc/sudoers\**

**line=’mike ALL=(ALL:ALL) NOPASSWD: ALL’" \**

**--become-method=su--become --ask-become-pass all**

# Understanding privilege escalation: become

Ansible uses existing privilege escalation systems to execute tasks with root privileges or with another user’s permissions. Because this feature allows you to ‘become’ another user, different from the user that logged into the machine (remote user), we call it become. The become keyword leverages existing privilege escalation tools like sudo, su, pfexec, doas, pbrun, dzdo, ksu, runas, machinectl and others.

For example, to manage a system service (which requires root privileges) when connected as a non-root user, you can use the default value of become\_user (root)

En gros il permet d’ajouter sudo avant chaque commande à exécuter.

To run a command as the apache user:

- name: Run a command as the apache user

command: somecommand

become: yes

become\_user: apache

### [Become command-line options](https://docs.ansible.com/ansible/latest/user_guide/become.html#id4)

|  |  |
| --- | --- |
| --ask-become-pass, -K | |
|  | ask for privilege escalation password; does not imply become will be used. Note that this password will be used for all hosts. |
| --become, -b | run operations with become (no password implied) |
| --become-method=BECOME\_METHOD | |
|  | privilege escalation method to use (default=sudo), valid choices: [ sudo | su | pbrun | pfexec | doas | dzdo | ksu | runas | machinectl ] |
| --become-user=BECOME\_USER | |
|  | run operations as this user (default=root), does not imply –become/-b |

#### [Passwords for enable mode](https://docs.ansible.com/ansible/latest/user_guide/become.html#id13)

If you need a password to enter enable mode, you can specify it in one of two ways:

* providing the [--ask-become-pass](https://docs.ansible.com/ansible/latest/cli/ansible-playbook.html#cmdoption-ansible-playbook-k) command line option

### [Accounts without a password](https://docs.ansible.com/ansible/latest/user_guide/become.html#id19)

Warning

As a general security best practice, you should avoid allowing accounts without passwords.

Ansible can be used to become a Windows account that does not have a password (like the Guest account). To become an account without a password, set up the variables like normal but set ansible\_become\_password: ''.

Before become can work on an account like this, the local policy [Accounts: Limit local account use of blank passwords to console logon only](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/jj852174(v=ws.11)) must be disabled. This can either be done through a Group Policy Object (GPO) or with this Ansible task:

- name: allow blank password on become

win\_regedit:

path: HKLM:\SYSTEM\CurrentControlSet\Control\Lsa

name: LimitBlankPasswordUse

data: 0

type: dword

state: present

Note

This is only for accounts that do not have a password. You still need to set the account’s password under ansible\_become\_password if the become\_user has a password.

## Ansible Playbooks

Ansible playbooks are the main way we utilize Ansible. We discuss what makes up an Ansible playbook and the ansible-playbook command. Ansible inventory is briefly touched on and we cover essential playbook syntax.

Playbooks are like scripts

Playbooks run using **ansible-playbook** command

A playbook is a YAML file, and contains different elements called plays.

Each task has a name and module

Modules may have parameters.

vim web.yml

# Install httpd, start it, create index.html, insertline in file

- hosts: webservers

become: yes

tasks:

- name: install httpd

yum:

name: httpd

state: latest

- name: create index.html file

file:

name: /var/www/html/index.html

state: touch

- name: add web content

lineinfile:

line: " It was a clear sunday morning"

path: /var/www/html/index.html

- name: start httpd

service:

name: httpd

state: started

...

Met ce file dans le même dossier du new inventory.

Ensuite lance la commande : ansible-playbook –i inv web.yml

**Playbooks and Idempotency**

*Idempotent* is a fancy word that means that you can do something multiple times and the

outcome will be the same. In Ansible terms, a playbook is considered idempotent if you

can run it multiple times and after the first run the machine is in a certain state, which

doesn’t change if you run the same playbook again at any point in time after that.

If you run a playbook again, instead of saying changed , it says ok . This is because

the apt module checks to see if a package is installed before attempting to install it. As the

packages are already installed, it won’t make any changes. This playbook is idempotent

If you want to jump straight in and see some sample playbooks, there are lots of them

available in the ansible-examples repository on Github

<https://github.com/ansible/ansible-examples>

## Working with variables

**Les variables**

Une configuration doit s'adapter suivant les besoins ! Pour cela on peut définir des variables que l'on pourra utiliser dans nos recettes.

Variables may be used within Ansible to work with dynamic data

Variables can be scoped by group, host or within a playbook

Variable are referenced using double curly braces

Exemple du precedent playbook avec une variable :

# Install httpd, start it, create index.html, insertline in file

- hosts: webservers

become: yes

tasks:

- name: "{{ target-service}}"

yum:

name: httpd

state: latest

- name: create index.html file

file:

name: /var/www/html/index.html

state: touch

- name: add web content

lineinfile:

line: " It was a clear sunday morning"

path: /var/www/html/index.html

- name: start httpd

service:

name: "{{ target-service}}"

state: started

...

Ensuite on lance la commande ainsi : ansible-playbook –I inv web.yml –e "target-service=httpd"

Il te suffit ensuite de désigner un autre paquet à la place de httpd.

## How to Use Ansible Facts

Facts are a great way to use existing system properties with Ansible. This lesson talks through what Ansible facts are and some use cases for them.

Facts are simply various properties regarding a given remote system

La commande **ansible qeyss2 –m setup** permet d’obtenir un max d’informations sur le système du node en question.

On peut filtrer les infos ainsi : **ansible qeyss2 –m setup** **-a filter=\*ipv4\***

The setup module can retrieve facts

The filter parameter takes regex to allow you to prune fact output

Facts are gathered by default in the Ansible Playbook execution

**C’est quoi les facts dans Ansible**

Si vous avez déjà lancé un playbook, vous avez sûrement remarqué lors de son exécution, que la première tâche qui est réalisée n’est pas une tâche que vous avez demandée explicitement.

1. ansible-playbook -i inventory/prod/blop -u blop --private-key=blop.key configure-blop.yml
2. [...]]
3. TASK [Gathering Facts] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Cette tâche **[Gathering Facts]**, qui ne fait à première vue rien, correspond en fait à la connexion à la ou les machines sur lesquelles seront exécutées les playbooks. Si la connexion échoue, le playbook échouera sur cette cible, et continuera éventuellement sur les autres (s’il en reste). Si l’hôte répond, alors Ansible en profite pour récupérer moult informations en rapport avec cette machines et les stockent dans une variable **ansible\_facts**

**A quoi ça ressemble ?**

Comme je ne disais, à première vue, cette commande de fait rien. Si on ne sait pas qu’elle stocke des informations dans une variables, on ne peut rien en faire.

Un bon moyen d’avoir un premier aperçu de ce qu’on peut en faire est de les afficher. On peut faire ça avec [le module « setup »](http://docs.ansible.com/ansible/setup_module.html), qui est en réalité le même module qui est appelé lors de l’étape [Gathering facts].

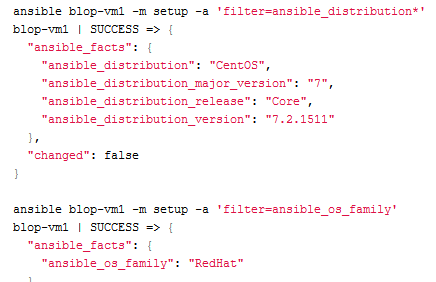
Le retour se fera au format JSON, ce qui est certes peu digeste, mais facilitera grandement les manipulations de type « filtre » (via le flag filter intégré ou via l’utilitaire jq par exemple).

**Ok, on en fait quoi ?**

Du coup je profite de cet article pour faire un tuto tout simple et qui peut être utile dans de nombreux cas, réaliser des opérations différentes en fonction de la distribution de la machine concernée.

L’information qui va nous intéresser ici concerne donc les remontées en tant que « nom » ou « version » d’une distribution.

**Comment récupérer la version de l’OS**



**Utiliser les facts dans un playbook**

Pour continuer dans l’exemple, on va faire une chose qu’il ne faut jamais faire : désactiver le firewall et SELinux.

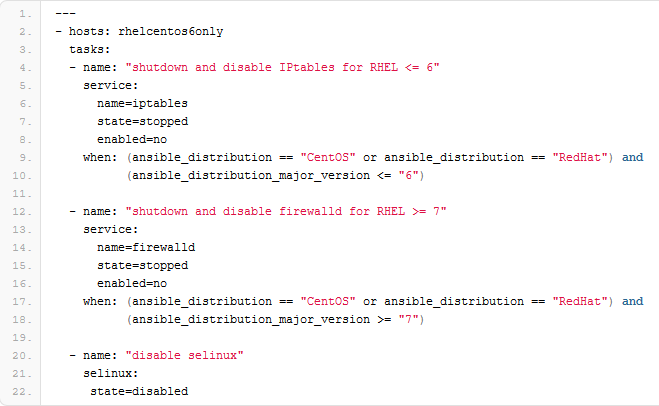
Imaginons que vous souhaitiez quand même le faire. Si vous lancez ce playbook sur tout votre parc et qu’il n’est pas homogène, vous allez tomber sur des groupes de machines Ubuntu, CentOS 5, 6, 7…

Un moyen de gérer les cas particuliers pourrait être de les classer tous vos hosts dans des groupes, et de créer un playbook pour chaque OS/version, restreint à un groupe de machines uniquement.

Ce n’est d’ailleurs pas une mauvaise idée, surtout pour gérer les distrib aussi différentes que RHEL et Ubuntu par exemple.

En revanche, dans le cas de RHEL, on a des différences qui apparaissent à partir de la RHEL 7, notamment la gestion du Firewall qui passe de IPtables à firewalld.

On va donc faire appel à la clause **when:** de ansible, qui conditionne l’exécution d’une tâche (ou d’un rôle ou d’un block) à une validation. Et ça donnerait quelque chose comme ça :



Dans cet exemple, lors de l’exécution du playbook, on aura donc, pour chaque VM CentOS ou RHEL, une tâche qui sera exécutée, et l’autre qui sera marquée « skipping » (en bleu clair) et le playbook marchera pour toutes les RHEL, quelque soit leur version.

## Troubleshooting and Debugging Ansible

Being able to troubleshoot an Ansible playbook is a crucial skill for working with Ansible.

The debug module may be used to help troubleshoot plays

Understanding Handlers in Ansible

1