**Ansible: Setup, Configure, and Ad Hoc Commands Deep Dive**

# Introduction

## About Ansible

In this lesson, we discuss the basic functions of Ansible. There is also some discussion on how Ansible operates at a high level. This sets the foundation for the course.

Ansible is a tool/framework/API for doing remote things on hosts

# Setup and Configuration

## Ansible Setup

**How to install Ansible**

sudo yum install ansible ou apt-get install ansible

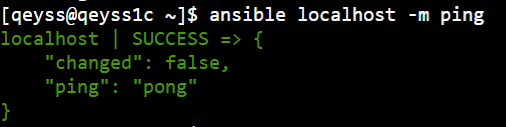
Sous Redhat faire d’abord un : yum list epel-release

EPEL ou encore (Extra Package for Entreprise Linux) est un repo qui fournit 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!

Il peut être aussi intéressant d’installer git : sudo yum install git

For SHSH configuration it’s recommended to create an ansible user



**Configuring SSH and Sudo for Ansible**

Ansible is best implemented using a common user across all Ansible controlled systems.

**1° Création de l’utilisateur ansible**

Se connecter en SSH au host

Passer en root sudo su -

useradd ansible (adduser sous Debian)

passwd ansible

logout (pour se déconnecter du root)

logout (pour se déconnecter du serveur distant)

**2° Créer la paire de clé de l’utilisateur ansible**

Se connecter avec le user ansible

ssh-keygen

ssh-copy-id NomDuServeur

ssh NomDuServeur

Configure sudo for the ansible user

On se connecte en root ou avc un user ayant déjà les droits d’utiliser sudo puis on tape:

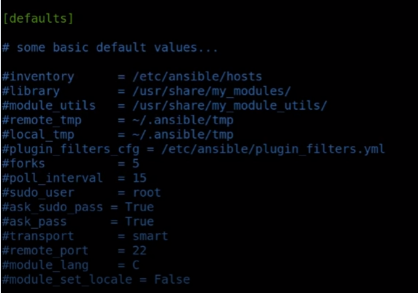
**sudo visudo** puis on ajoute cette ligne à cet endroit comme dans l’image ci-dessous :



## Ansible Configuration

**The Ansible Configuration File**

The default Ansible configuration file is in /etc/ansible/ansible.cfg



Configuration file which will be searched for in the following order:

* ANSIBLE\_CONFIG ( environment variable if set)
* Ansible.cfg ( in the current directory)
* Ansible.cfg ( in the home directory)
* /etc/ansible/ansible.cfg

**Setting Up the Ansible Inventory**

An inventory is a list of hosts that Ansible manages

The default Ansible Inventory File is /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 par exemple je peux ajouter ce server qeyss2c.mylabserver.com ainsi:

qeyss ansible\_host=qeyss2c.mylabserver.com

Mon serveur s’intitule maintenant qeyss

You can also create groups:

[webserver]

qeyss2c.mylabserver.com

qeyss3c.mylabserver.com ou son IP

[databaseserver]

qeyss4c.mylabserver.com

qeyss5c.mylabserver.com

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

- mkdir ansible

- vim inv

Ensuite on utilise le –i pour spécifier l’inventaire voulu

Exemple d’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.

Je peux spécifier le groupe de serveurs ainsi :

ansible -i inv webserver -m ping

# Ansible Ad Hoc Commands

## The Ansible Command

Ansible ad-hoc commands are comparable to bash commands.

Playbook is comparable to a bash script

Syntax:

* Ansible <HOST> -b –m <MODULE> -a “<ARG1 ARG2 ARG3>” –f <NUM\_FORKS>

Exemple:

* ansible –i inv –m ping
* ansible –b –m yum –a “name=git state=present”

Le -**b** indique qu’il va falloir mettre le sudo

Le **-m** : indique à Ansible d’utiliser un module spécifique

## Understanding Ansible Modules

Modules are discrete units of code that can be used from the command line or in a playbook task.

Modules may take arguments.

Examples of modules: ping, setup, yum, service, copy, apt

There are a lot of modules, see: docs.ansible.com

Tu peux aussi avoir des infos en ligne de commande en tapant par exemple: **ansible-doc** NomDuModule

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 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.

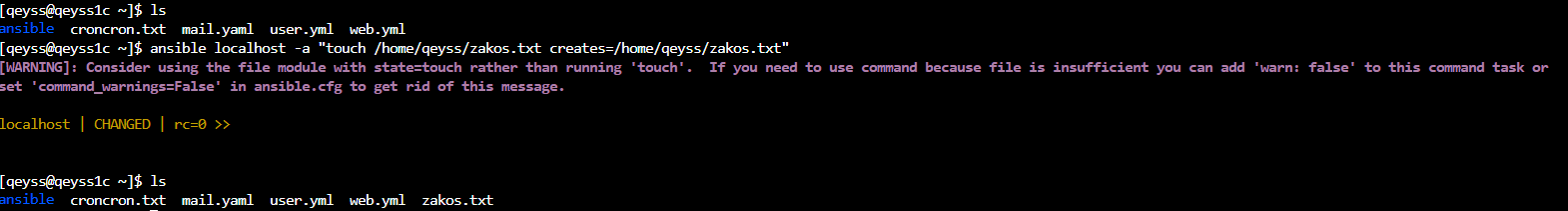
## The Shell and Command Modules

The **Shell** and **command** modules are both effective ways to run raw commands on a target host

The **command** module is the default module used with Ansible ad-hoc when no module is specified.

Donc si tu ne mets pas de –m suivi d’un module, ça sera le module **command** qui sera utilisé par défaut.

Examples with the **command** module:



Pour le supprimer je tapes:

ansible localhost -a "rm -f /home/qeyss/zakos.txt removes=/home/qeyss/zakos.txt"

Examples with the **Shell** module:

ansible localhost -m shell -a "touch /home/qeyss/zakos.txt creates=/home/qeyss/zakos.txt"

ansible localhost -m shell -a "rm-f /home/qeyss/zakos.txt removes=/home/qeyss/zakos.txt"

## Collecting System Information

Ansible facts allow you to collect system information from a remote system.

We use the **setup** module

ansible -m setup | less

Facts may be filtered using the setup module ad-hoc by passing a value for the filter parameter, ex:

ansible -m setup -a "filter=\*dist\*"

Ansible command output may be directed to a file using the - - tree outputfile flag which may be helpful when working with facts :

ansible -m setup - - tree facts

It is possible to use {{ ansible\_facts }} for conditional plays based on facts.

# File Manipulation with Ansible

## Working with the File and Copy Modules

The **file** module can be used to create, delete and modify file properties.

$ ansible -i inv qeyss2 -m file -a "name=testfile state=touch"

**Si je veux le créer dans un dossier particulier je dois donner le chemin, ex :**

$ ansible -i inv qeyss2 -m file -a "name=/james/testfile state=touch"

Si je veux le créer dans le dossier root, il faudra ajouter le -b pour qu’il utilise sudo avant la commande.

$ ansible -i inv qeyss2 -b -m file -a "name=/root/testfile state=touch"

**Pour le supprimer je tape:**

$ ansible -i inv qeyss2 -m file -a "name=/james/testfile state=absent"

The **copy** module may be used to copy files from a number of sources

$ ansible -i inv qeyss2 -m copy -a "src=zorro dest=newzorro"

## Editing File Contents with the lineinfile Module

The lineinfile module is a great way for adding content to a file in a specific place.

The key feature of the module is inserting a line of provided text into a given file.

Ex:

$ ansible -i inv qeyss2 -m lineinfile -a "path=/home/qeyss/newzorro line='I love you Nunu' state=present"

The module will place the line at the end of a file by default however it can place the line after or before a line matched text.

$ ansible -i inv qeyss2 -m lineinfile -a "path=/home/qeyss/newzorro insertbefore='Nunu' line='Sukayna is not boring ' state=present"

If the module detects the provided text already in place it will not add it again.

If the text does not exist you add create=yes, like:

$ ansible -i inv qeyss2 -m lineinfile -a "line='I love you Nunu' path=/home/qeyss/newzorro create=yes"

It may also remove lines from a text

## Downloading Files with the get\_url Module

The get\_url module provides functionality for download files over http with variety of options

$ ansible -i inv qeyss2 -m get\_url -a "url=http://google.com dest=/home/qeyss/google.html"

## Working with File Archives

The **archive** and **unarchive** modules may be used to do work with various types of file archives

These modules work with common archives such as tar, gunzip(default) and zip.

These modules allow for compression (or expansion) of files and directories.

**I create many files**

$ ansible -i inv qeyss2 -m file -a "name=testfile2 state=touch"

$ ansible -i inv qeyss2 -m file -a "name=testfile3 state=touch"

**I create the zip file**

$ ansible -i inv qeyss2 -m archive -a " path=/home/qeyss/ format=zip dest=/tmp/files.zip"

**For unarchive**

$ ansible -i inv qeyss2 -m unarchive -a "remote\_src=yes src=/tmp/files.zip dest=/tmp"

# Configuring System Users and Groups

## Creating System Users with the User Module

Creating and manipulating users is a fundamental administration task. This may be done using the Ansible **user** module.

The user module allows to:

* create users
* remove users
* change user properties

**Création d’un user**

$ ansible -i inv qeyss2 -b -m user -a "name=hajar"

**Suppression d’un user**

$ ansible -i inv qeyss2 -b -m user -a "name=hajar state=absent remove=yes"

Si tu ne mets pas le remove, il va supprimer l’utilisateur mais pas son home directory.

Tu peux vérifier ensuite la suppression en faisant un cat /etc/passwd et à la fin tu ne verras pas ton user normalement.

## Working with the Group Module

The **group** module works with groups in the same way as the **user** module interacts with users.

**Création d’un groupe:**

$ ansible -i inv qeyss2 -b -m group -a "name=consultants"

Une fois le groupe créé pour y ajouter un user en le créant:

$ ansible -i inv qeyss2 -b -m user -a "name=hajar group=consultants"

**Suppression d’un groupe:**

$ ansible -i inv qeyss2 -b -m user -a "name=hajar group=consultants state=absent"

# Installing Software and Daemon Management

## Installing Software

The ability to install software is essential for daily systems administration.

We can use 3 modules: **package**, **yum** and **apt**.

The package module will automatically detect the target host distribution and use the appropriate method of installing software.

**With package**

$ ansible -i inv qeyss2 -b -m **package** -a "name=httpd state=latest" ( voir si c’est pas started qu’il faut mettre)

**With yum**

$ ansible -i inv qeyss2 -b -m **yum** -a " name=git state=latest"

## Controlling Daemons with the Service Module

The Ansible service module may be used to control system daemons using either the BSD init system or the newer systemd approach.

$ ansible -i inv qeyss2 -b -m **service** -a " name=httpd state=started enabled=yes"

# Advanced features

1

## Managing Long-running Commands

1

## Parallelism in Ansible

Ansible uses what are known as forks to execute tasks in parallel.

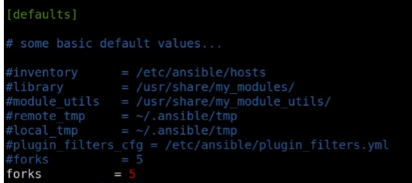
Each forks is able to rn a task against a target host.

By default Ansible executes 5 forks.

Quand tu lances une commande, s’il y a 5 hosts tout sera fait simultanément.

If there is more than 5 hosts, hosts after the first 5 must wait for a fork to become free.

The number of forks used may be configured in ansible.cfg or an as-needed basis using -f or – forks flag



Tu peux mettre la valeur en function du nombre de hosts ou alors avec -f

$ ansible all –m ping -f 50