IBMP | Explorer le monde végétal

ANSIBLE

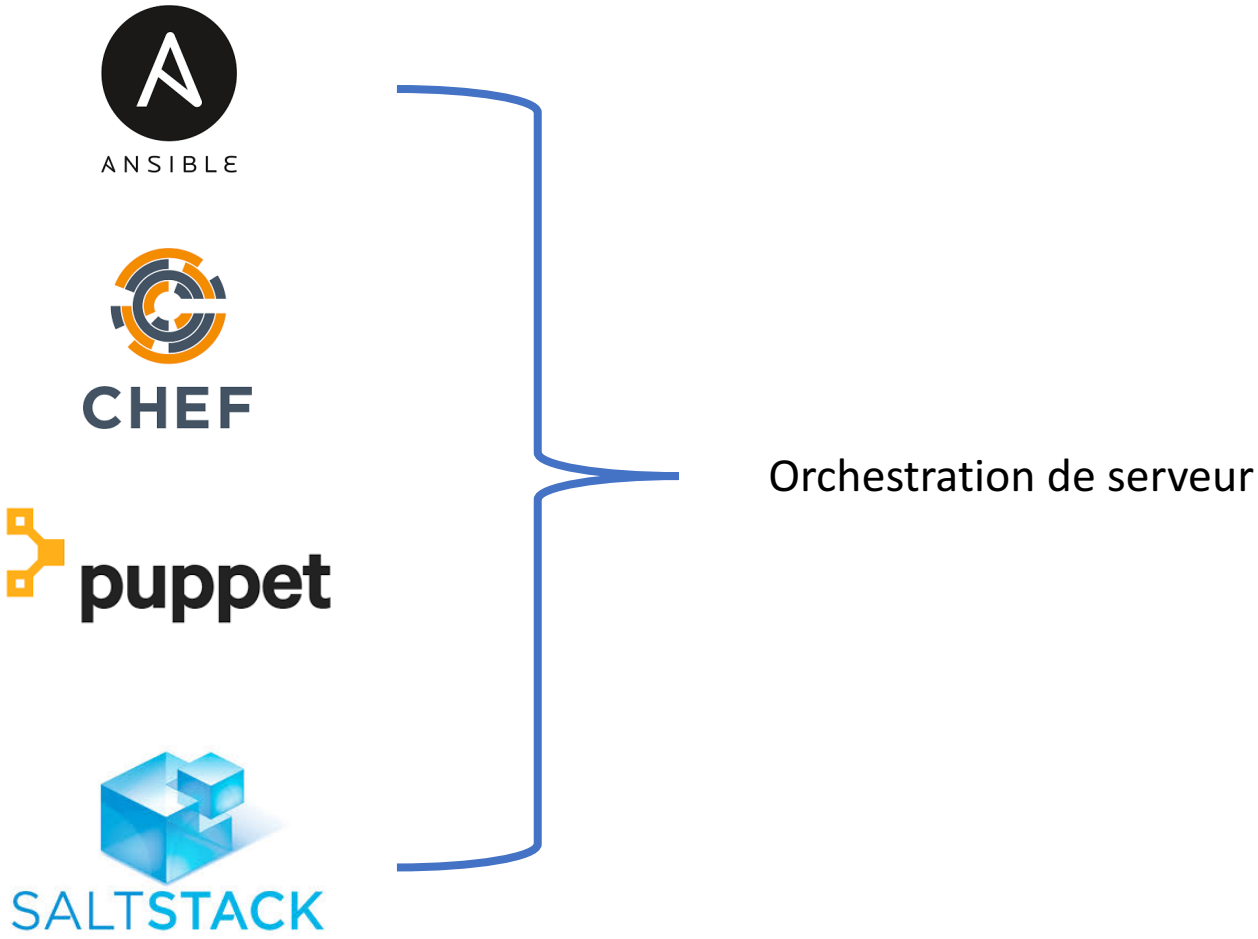
ANF – LYON – 2018

Francois Disdier

ANSIBLE ?

- Créer en 2012
- Développé en python
- Système agentless
- SSH ouvert + python installé
- Capable de piloter des systèmes
  - Windows
  - Linux
  - Unix
  - Equipements réseaux

ANSIBLE ?



Contexte IBMP

Plateforme bioimage

gm Génétique Moléculaire Génomique Microbiologie

LNCA

Cluster IBMP dédié à la Bioinformatique

ANSIBLE

Maintenance, installation
Administration niveau 1

- Sauvegarde des processus d'installation et mise à jour des outils bioinformatique
- Etablissement et maintient de workflow utile à l'administration du cluster

Mise en œuvre d'ansible

Installation

- Via pip
- Via dépot tier EPEL
- Si la machine utilise SELINUX. Il faut utiliser le paquet python-selinux

Pour fonctionner ansible ≧ python 2.5

Mise en œuvre d'ansible

Inventaire

- Le fichier hosts définit la liste des machines impactées par Ansible
- Il se trouve dans le dossier /etc/ansible/
- On peut organiser les machines par groupe

Exemple :

```
[Ordonnanceur]
10.2.2.5
[file_slow]
nlame1.example.com
nlame2.example.com
[File_fast]
10.2.2.10
```

Mise en œuvre d'ansible

Variables

- On peut définir des variables arbitraire dans le fichiers hosts
- Cela permet de mettre en valeur des machines dans l'inventaire, des conditions d'accès,…
- On peut utiliser des variables type ou ansible_ssh_user…

Exemple :

```
[file_slow]
nlame1.example.com type=master
nlame2.example.com type=slave
[File_fast]
10.2.2.10 ansible_ssh_user=root
```

Mise en œuvre d'ansible

Les outils

ansible fournit plusieurs outils en ligne de commande

Outil	Description
ansible	Execution d'une commande unique
ansible-playbook	Execution de playbook (ensemble de tâches à effectuer)
ansible-doc	Accès au listing + documentation des serveurs
ansible-vault	Gestion de fichiers chiffrés (stockage variable - mot de passe)
ansible-galaxy	Accès au dépôt des rôles d'ansible

Exemple :

ansible –i hosts all –m ping

Mise en œuvre d'ansible

Les accès

Les accès aux hôtes -> 3 méthodes différentes

- En utilisant les informations de connexion dans l'inventaire
- En précisant toutes les informations sur la ligne de commande ansible
- En configurant la machine de management pour une connexion transparente

Pour cela il faudra :

1. Créer une paire de clé SSH (ssh-keygen)
2. Déployer la clé publique sur chacun des serveurs (ssh-copy-id)
3. Créer une configuration sudo sans mot de passe sur les hôtes distants ou un login en tant que root.

Mise en œuvre d'ansible

Exemple

$ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
...
$ssh-copy-id ansible@nlame1.example.com

Nota: sudo peut être configuré pour ne pas demander de mot de passe

$cat /etc/sudoers.d/ansible
ansible ALL=(ALL) NOPASSWD : ALL

Les modules

Définition :

Les modules sont la base des actions exécutées par Ansible.
Chaque modules :

- Lance une action spécifique
- accepte des arguments

On peut lister les modules en exécutant la commande :

- *ansible-doc --list*

on retrouve cette liste à cette adresse :

http://docs.ansible.com/ansible/modules_by_category.html

Les modules

Liste de modules couramment utilisés

Outil	Description
ping	validation de l'inventaire
setup	retourne une liste d'informations matériels de l'hôte
shell /command	permettent d'exécuter des commandes sur les hôtes
user	permet de gérer des utilisateurs sur les hôtes
file	permet de gérer des droits sur des fichiers
service	permet de gérer les services systèmes tel que : arrêt/démarrage/redémarrage/activation ou pas au boot
yum/apt/ zypper	permet de gérer l'installation, la mise à jour et la suppression de paquets.

La syntaxe est :

ansible (hote/groupe/all) –m MODULE [-a *"arg1=val1"*]

Exemple :

$ ansible all –m yum –a *"name=httpd state=latest"*

Les modules

TP n°1

1. Installer ansible sur une VM de commande
   puis installer python sur toutes les VM
2. S'assurer que openssl est à jour sur tous les serveurs
3. Créer un utilisateur ansible avec un shell /bin/bash
4. Installer la clef publique SSH de notre utilisateur sur l'utilisateur ansible
5. Ajouter dans le fichier hosts :
   - node-head (groupe master)
   - node-compute-01 (groupe slave)
6. Faire un ping via ansible sur la VM node-head et la VM node-compute-01
7. Récupérer les informations matériel des VM distantes

Les playbooks

Définition :

Les playbooks permettent d'exécuter un ensemble de tâches à effectuer sur les machines hôtes de manière séquentielle sur tout ou partie de l'inventaire.

Un playbook est constitué au minimum :

- D'une variable hosts -> machine à impacter
- D'une variable task -> module(s) à activer

Les playbooks peuvent :

- Emettre des notifications
- Effectuer des actions conditionnelles
- Utiliser des templates pour manipuler des fichiers

Les playbooks

Syntaxe :

Les playbooks sont écrits avec le langage YAML (facile à lire et à écrire)

Règles de bases:

- On démarre un playbook avec ---
- On le finit avec …
- Les commentaires se font avec #
- L'indentation est de 2 espaces

Les tâches :

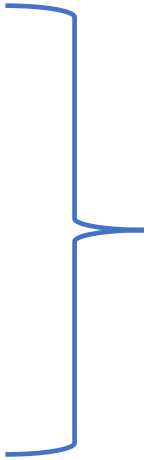
Chaque tâche est définie par un nom (rubrique *name*) pour commenter l'action et d'un module à appeler. Des attributs supplémentaires permettent de rendre les playbooks plus interactifs :

- *notify* : défini un Handler à appeler
- *register* : sauve le résultat d'une action dans une variable
- *include* : inclusion d'un fichier externe
- *when* : exécute une action sous certaines conditions

Les playbooks

Exemple : nom du playbook install_apache.yml

```
hosts: file_slow
tasks:
 - name: Install apache
   yum:
     name: httpd
     state: present
     tags:
       -install_httpd
```



$ansible-playbook install_apache.yml

$ansible-playbook install_apache.yml –tags install_httpd -l nlame1.example.com

Les playbooks

Notification et handlers :

Les handlers permettent de définir des actions qui ne seront exécutées qu'au déclenchement d'une notification. Les handlers ne sont pris en compte qu'après la fin de toutes les tâches présentes dans le playbook.

Exemple : Redémarrer le service apache si la config à changé

```
- hosts: webservers
  tasks:
  - name: Configure apache
    template:
      src: httpd.conf
      dest: /etc/apache2/apache2.conf
    when: ansible_os_family == 'Debian'
    notify: Restart apache
  handlers:
  - name: Restart apache
    service:
      name: apache2
      state: restarted
```

Les playbooks

TP n°2

1. Écrire un playbook de déploiement d'apache avec support de PHP sur la VM node-head
2. Reprendre ce même playbook et intégrer un déploiement de htop et curl pour la vm node-compute-01
3. Enfin reprendre toujours le même playbook et rajouter une commande pour s'assurer que vim et nano sont à jour.

Structures de contrôle

Définition :

Les structures de contrôle sont probablement la partie la plus utile. En effet grâce aux structures de contrôle on peut manipuler les données provenant des hôtes de façon très flexible et puissante.

Les *facts* :

La première action effectuée lors de l'exécution d'un playbook est de récolter des informations de chacun des hôtes distants.

- Ces *facts* alimentent des variables (préfixées par Ansible).
- Ces variables seront utilisables dans les filtres des tests et des boucles du playbook.
- Pour avoir une liste de tous les *facts* qu'Ansible met à votre disposition il faut utiliser le module setup

Exemple : ansible –m setup hostname

Structures de contrôle

Les conditions :

Il est utile que certaines tâches du playbook soient effectuables seulement si une condition spécifique est remplie ou non. La directive when permet d'effectuer ce test.

Exemple :

```
- name: Installation apache (Debian)
  apt:
    name: apache2
    state: present
  when: ansible_os_family == 'Debian'
- name: Installation apache (REHL)
  yum :
    name: httpd
    state: present
  when: ansible_os_family == 'RedHat'
```

Structures de contrôle

Les boucles :

Les boucles permettent d'itérer sur les variables de type liste et dictionnaire. Cette technique permet d'exécuter plusieurs fois la même action sur un nombre d'éléments indéfini lors de l'écriture du rôle ou playbook. La liste complète des boucles est disponible dans la documentation ansible.

Exemple :
with_items permet de boucler sur une liste. A chaque itération une variable item prend la valeur suivante de la liste :

```
- name: Installation d'éditeurs
  yum:
    name: "{{ item }}"
  with_items:
    - vim
    - emacs
    - nano
```

Structures de contrôle

Les inclusions :

La directive *include* permet d'importer une liste de tâches ou de handlers. Cette technique permet de rendre génériques certaines actions.

Exemple :

```
hosts: all
tasks:
  - include: common-setup.yml
  - name : something else
...
```

Structures de contrôle

TP n°3

1. Récrire le playbook précédent en installant curl et htop en une seule tâche
2. Rajouter dans cette même tâche une condition qui permet de s'assurer que le serveur apache est présent sur la VM node-head, s'il ne l'est pas il faudra l'installer.
3. Générer un fichier /tmp/hosts similaire à /etc/hosts pour toutes les machines de l'inventaire.

Templates

Définition :

Ansible fonctionne sur un système de modèles (templates). Ils s'appuient sur Jinja2 et permettent de gérer des boucles, des tests logiques, des listes ou des variables.

- Une variable se déclare en la mettant au milieu de deux accolades : variable : {{variable}}
- Les instructions sont délimitées par accolades et pourcentage : {% instruction %}
- Les commentaires sont délimités par accolade et dièse.

Exemple :

```
{# Définition de notre liste #}
{% set myList = [ 1 , 2 , 3 ] %}
{# Parcours de la liste #}
{# Pour afficher l'ensemble des valeurs présentes #}
{% for value in myList %}
Valeur : {{ value }}
{% endfor %}
```

Rôles

Définition :

Les rôles sont des playbooks génériques, qui peuvent être intégrés dans d'autres playbooks. Cette notion est essentielle afin de créer des tâches complexes.
Pour chaque élément d'un rôle (tâche, handlers, ...) un fichier nommé main.yml sert de point d'entrée.

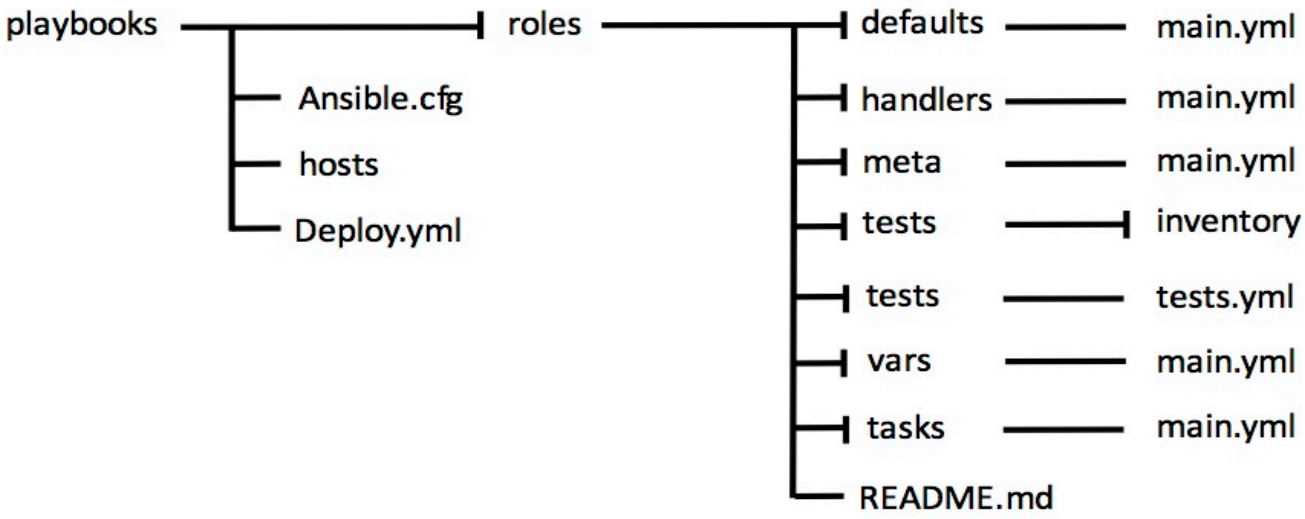
Exemple d'arborescence :

Dans le répertoire playbooks, où se trouvent ansible.cfg,hosts et deploy.yml, on va créer un dossier « roles » et on lancera la commande :

```
$ ansible-galaxy init -p roles/ test
```

Rôles

Notre rôle est bien créé et il contient les répertoires suivants :



Grâce à cette arborescence il ne sera pas nécessaire de préciser les directives vars, tasks ou handlers dans les fichiers vars/main.yml, tasks/main.yml et handlers.yml cette information sera obtenu à partir du nom du répertoire.

Rôles

Par exemple un fichier *tasks/main.yml* ne contiendra qu'une liste de tâches :

De la même façon, les modules *copy* et *template* iront chercher les fichiers sources directement dans les répertoires *files* et *templates*, sans besoin de les préciser dans le path.

Exemple : un fichier *tasks/main.yml* ne contiendra qu'une liste de tâches

```
- name: Install Apache
  apt: name=apache2  state=present
- name: Ensure service is registered and running service:
  name=apache2  enabled=true   state=started
```

Rôles

Obtenir des rôles :

Ansible fournit une bibliothèque de rôles en ligne : Galaxy.

Les rôles disponibles sur Galaxy sont soumis par la communauté Ansible, et leur qualité varie. Un système de notation permet de trouver les meilleurs rôles.

Exemple :

Installation
$ansible –galaxy install geerlingguy.apache

Suppression
$ ansible –galaxy remove geerlingguy.apache

Merci de votre attention