Ansible

Christophe DUFOUR

Ansible

- Logiciel de “provisioning” et d’orchestration de serveurs
- Permet le déploiement de logiciel
- Exécute des tâches
- Gère les configurations
- Libre et open source
- Développé en Python
- Fichiers d’instructions/config au format YAML
- Première version: 2012
- Développé par Red Hat

Principaux logiciels d’automatisation/provisioning

Particularités d’Ansible

- Aucun agent (agentless) à installer sur les serveurs contrôlés
- Tout par SSH
- Facile à installer, configurer et utiliser
- Pas de compétence poussée en scripting requise
- Langages simples et accessibles: Yaml, Jinja2 (template)
- Gestion d’état (idempotence)

Idempotence

Le sujet de l'idempotence dans le système informatisé est un sujet central. En mathématiques et en informatique, le concept d'idempotence signifie qu'une opération produit le même effet, qu'elle soit exécutée une ou plusieurs fois. En informatique, on caractérise l'effet de l'opération par l'état du système qui en résulte

- *hack-my-domain.fr*

Schéma

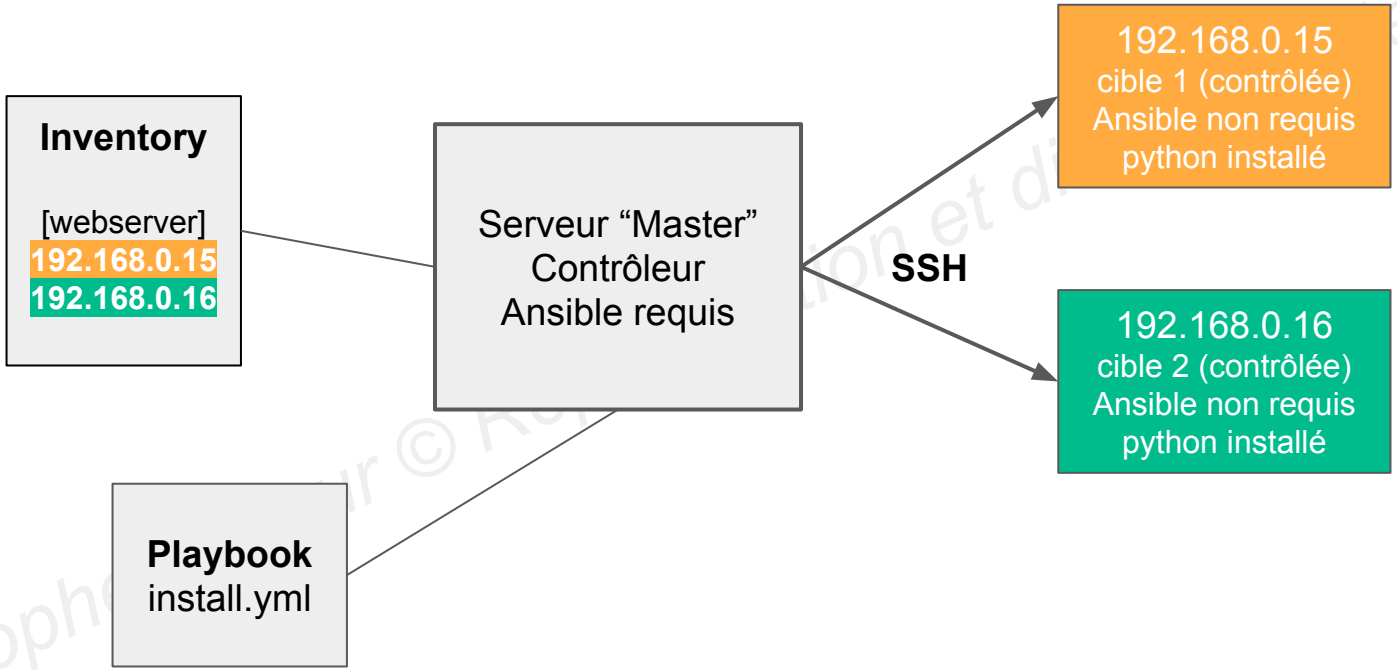
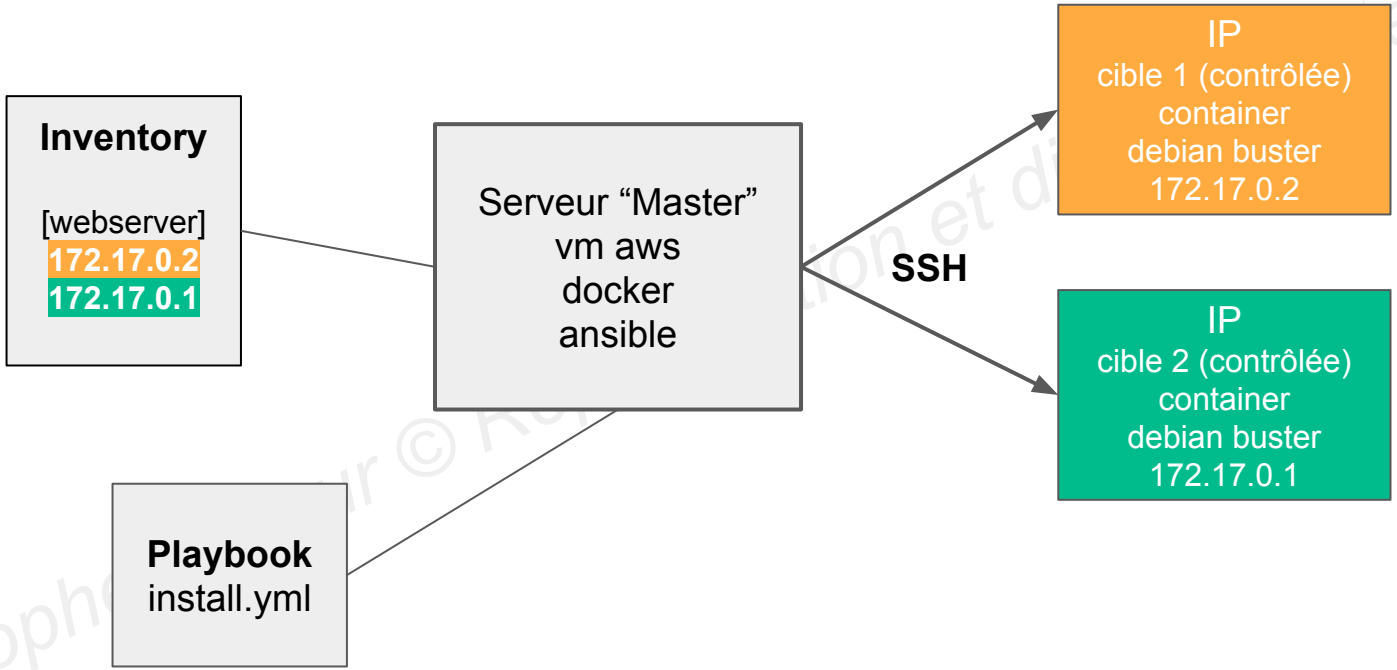


Schéma formation BNF



Installation

- Uniquement disponible pour Linux et Mac OS X (VM pour Windows)
- Installation par le gestionnaire de paquet de l'OS (apt, apk, yum, etc.)
- Ou par pip

```
# Ubuntu, debian
$ apt install ansible

# Centos
$ yum  install ansible

# Via pip
$ pip install ansible
```

Les fichiers *hosts*

```
[web]
192.168.0.1
192.168.0.2
192.168.0.3

[database]
192.168.0.4
192.168.0.5

[remote]
35.44.56.120
```

```
[web]
web1 ansible_host=192.168.0.1
web2 ansible_host=192.168.0.2
web3 ansible_host=192.168.0.3

[web:vars]
ansible_user=toto
ansible_ssh_private_key=~/.ssh/key.pem
```

La connexion aux machines du groupe web se fera par défaut en tant qu'utilisateur **toto**

La connexion aux machines du groupe web se fera par défaut en utilisant le fichier **key.pem** (clé privée) situé dans le dossier ~/.ssh de la machine maître

La clé publique de la machine maître devra se trouver (~/.ssh/authorized_keys) sur chacune des machines du group web

Commandes “one-liner” (ad-hoc)

ansible *[machine|groupe]* **-i** *fichier_hosts* **-m** *module*

Fichier hosts

```
[web]
web1 ansible_host=192.168.0.1
web2 ansible_host=192.168.0.2
web3 ansible_host=192.168.0.3

[web:vars]
ansible_user=admin
ansible_ssh_private_key=~/.ssh/key.pem
```

```
# Exécute le module ping sur la machine aliasée web1
$ ansible web1 -i hosts -m ping
```

```
# Exécute le module ping sur la machine aliasée web2
$ ansible web2 -i hosts -m ping
```

```
# Exécute le module ping sur toutes les machines du groupe web
$ ansible web -i hosts -m ping
```

Le fichier *playbook.yml*

Le **playbook** est un fichier au format **yaml** définissant une ou plusieurs séquences de **tâches** à accomplir sur les machines cibles

playbook.yml

```
- hosts: web
  remote_user: toto
  tasks:
    - name: Ping test
      ping:

- hosts: database
  remote_user: root
  tasks:
    - name: Install Mariadb
    - apt :
        - name=mariadb
        - state=present
```

Effectue un ping sur les machines du groupe web en tant qu'utilisateur toto

commande d'exécution

```
ansible-playbook playbook.yml -i hosts
```

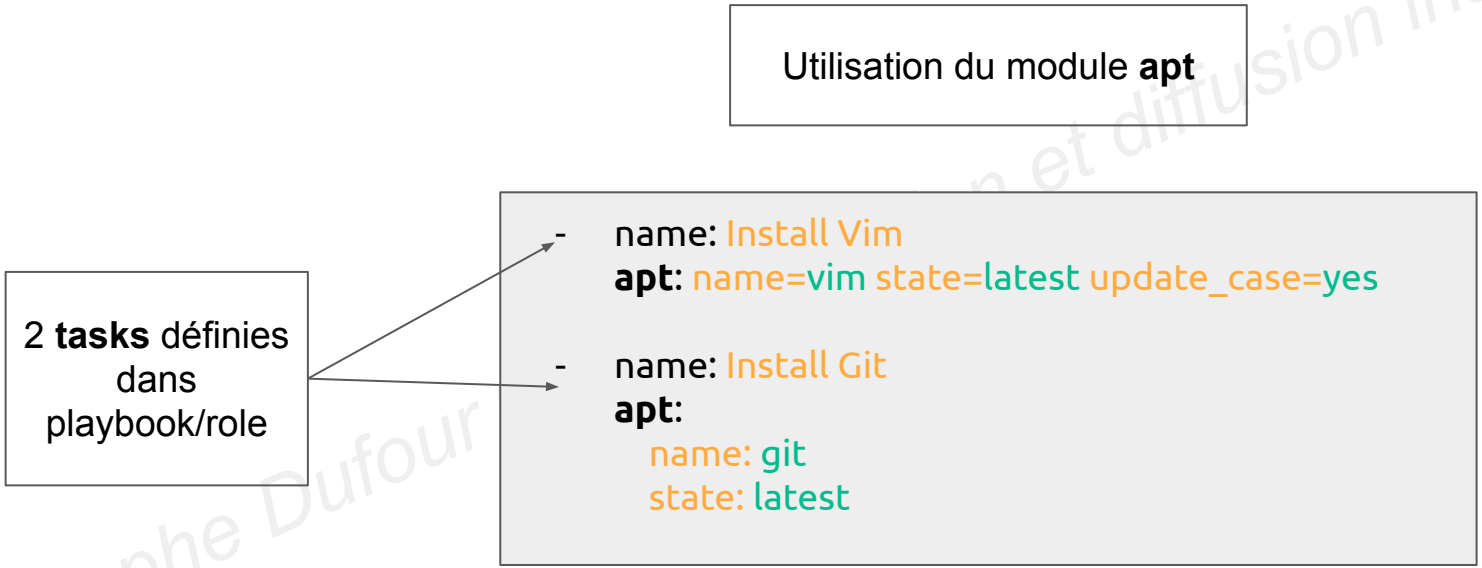
Installe si non déjà installé le logiciel Mariadb sur les machines du groupe database en tant qu'utilisateur root

Les modules

- **Commands**: command, shell, script, expect...
- **System**: service, user, group…
- **Files**: copy, fetch, lineinfile, template…
- **Packaging**: apt, yum, pip, bower, npm…
- **Database**: mysql, postgres, redis…
- **Docker**: image, container
- **Cloud**: aws, azure, cloudstack…
- **Notification**: irc, mail, jabber, slack...
- **Messaging**: rabbitmq
- **Source control**: git, mercurial, svn...

https://docs.ansible.com/ansible/latest/modules/modules_by_category.html

Utilisation d’un module



Boucle (sur des chaînes) - with_*

Item est remplacé par chaque élément (string) listé par **with_items**

```
- name: Install common tools
  apt: name={{ item }} state=latest update_cache=yes
  with_items:
    - vim
    - git
    - tree
    - htop
    - zsh
```

Boucle (sur des objets) - with_*

Item.name accède à la propriété **name** de l'objet item
Item.groups accède à la propriété **groups** de l'objet item

```
- name: Create users
  user: name={{ item.name }} state=present groups={{ item.groups }}
  with_items:
    - { name: alessandro, groups: developer }
    - { name: gabriele, groups: developer }
    - { name: aramis, groups: musketeer}
    - { name: dartagnan, groups: musketeer}
```

Boucle (sur des objets) - loop

Item.name accède à la propriété **name** de l'objet item
Item.groups accède à la propriété **groups** de l'objet item

```
- name: Create users
  user: name={{ item.name }} state=present groups={{ item.groups }}
  loop:
    - { name: alessandro, groups: developer }
    - { name: gabriele, groups: developer }
    - { name: aramis, groups: musketeer}
    - { name: dartagnan, groups: musketeer}
```

Condition - when

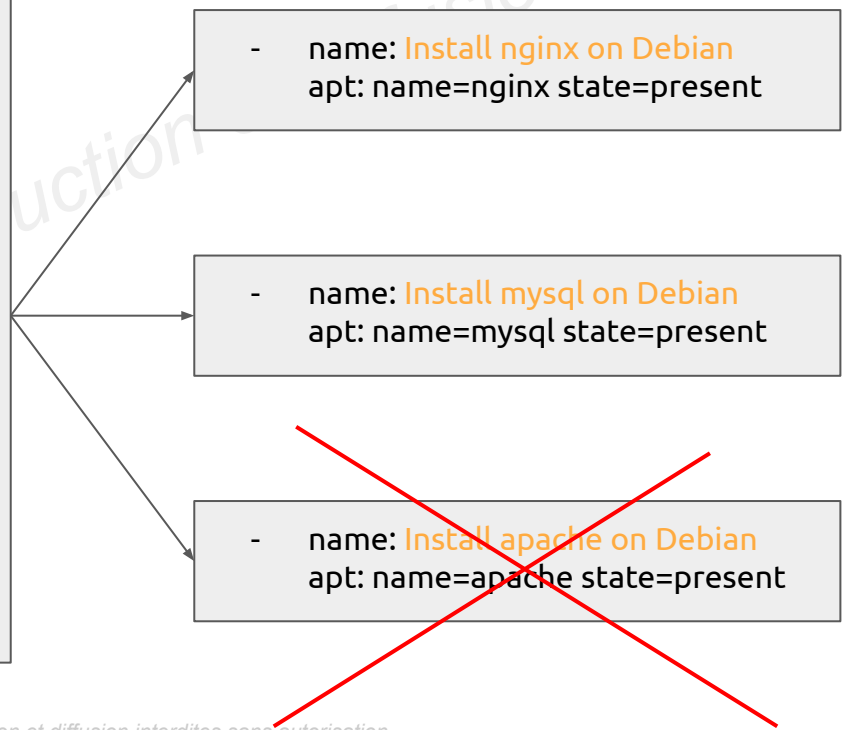
```
- name: Install Nginx
  hosts: all
  tasks:
    - name: Install Nginx on Debian
      apt: name= nginx state=present
      when: << condition >>

    - name: Install Nginx on Redhat
      yum: name= nginx state=present
      when: << condition >>
```

ansible_os_family == “Debian” **and**
ansible_distribution_version == “18.04”

ansible_os_family == “RedHat” **or**
ansible_os_family == “SUSE”

Condition dans une boucle

```
- name: Install Softwares
  hosts: all
  vars:
      packages:
        - name: nginx
          required: true
        - name: mysql
          required: true
        - name: apache
          required: false
  tasks:
    - name: Install "{{ item.name }}" on Debian
      apt: name= {{ item.name }} state=present
      loop: "{{ packages }}"
```



Condition et register

```
- name: Check status of a service and email if down
  hosts: localhost
  tasks:
    - command: service httpd status
      register: result
    - mail:
        to: admin@company.com
        subject: Service Alert
        body: Httpd Service is down
        when: result.stdout.find('down') != -1
```

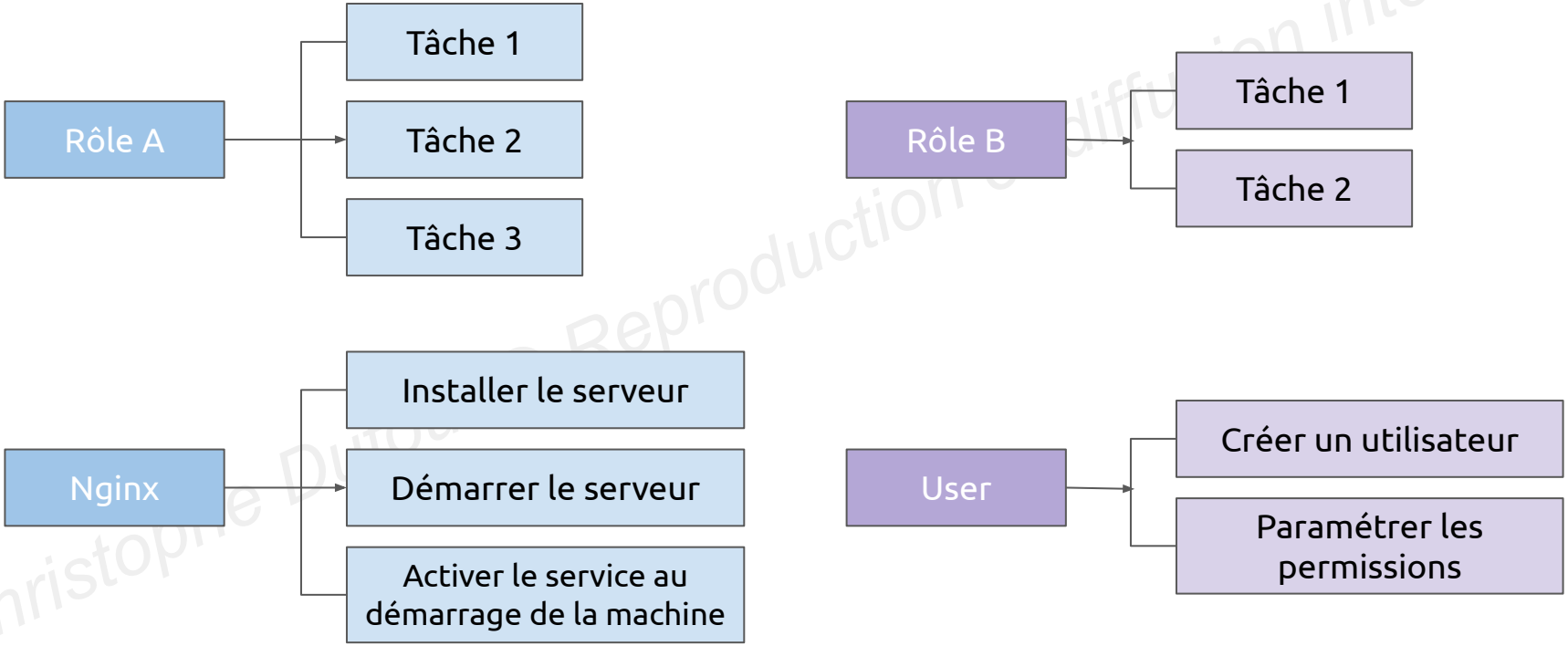
La commande (et sa sortie) est enregistrée dans la variable **result**

L'email est envoyé **si result** contient dans sa propriété stdout (sortie standard) la chaîne "down" signifiant que le service httpd n'est pas en cours

Les rôles

- Facilitent l’organisation les tâches en les regroupant de manière logique (roles/role_name/tasks/main.yml)
- Favorisent la réutilisation des tâches (inclusion du rôle dans le playbook cible)
- Permettent le partage communautaire (galaxy)

Roles

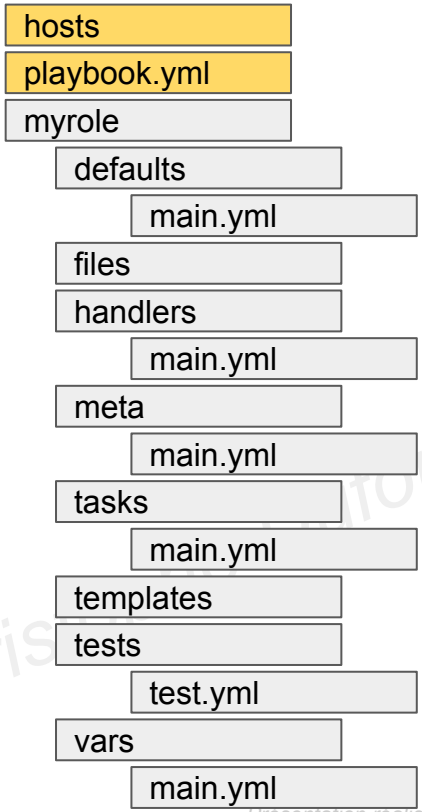


Ansible vault

Fonctionnalité permettant de protéger des données sensibles telles que des mots de passe dans des fichiers cryptés

- **create**: crée un fichier encrypté
- **view**: visualise les données du fichier encrypté
- **edit**: modifie fichier encrypté
- **encrypt**: encrypte un fichier non crypté
- **decrypt**: décrypte un fichier crypté
- **--ask-vault-pass**: fournit le mot de passe durant l'exécution du playbook
- **--vault-password-file**: passe le mot de passe via un fichier

Structure



La commande **ansible-galaxy init <role_name>** permet de générer automatiquement un template complet de role structuré en dossiers multiples

```
# Création de la structure
$ ansible-galaxy init myrole
```