# Introduction au DevOps

## Ansible

Jolan PHILIPPE

September 15, 2025

Université d'Orléans

# Secure Shell (SSH)

# Rappel

## Protocole

SSH (*Secure Shell*) est un protocole réseau chiffré permettant une connexion sécurisée entre un client et un serveur. Il repose généralement sur TCP et utilise le port **22** par défaut.

## Utilisation courante

Connexion distante à un serveur pour exécuter des commandes, transfert de fichiers (`scp`, `sftp`), tunneling de ports, etc.

```
ssh utilisateur@serveur.example.com
```

# Démarrer une application

Dans un terminal :

```
> ssh user@mon-serveur
> sudo apt update && sudo apt install -y nginx
> nohup nginx -g "daemon off;" &
```

Cette suite de commandes :

- se connecte en SSH,
- installe Nginx avec privilèges sudo,
- lance Nginx en arrière-plan grâce à `nohup` et `&`.

# Multi-SSH

Pour communiquer avec plusieurs machines en parallèle :

- Utiliser `ssh` dans des boucles shell (for/parallel),
- Outils spécialisés comme `pssh` ou `parallel-ssh` (ou Ansible lui-même).

**Gestion de la concurrence**

Attention ! Il faut gérer l'ordre d'exécution, la tolérance aux pannes et le parallélisme afin d'éviter les conflits d'accès ou la surcharge réseau.

# Le configuration management, c'est quoi ?

# Configuration management

**Définition**

La **gestion de configuration** (*Configuration Management*) est la pratique qui consiste à automatiser l'installation, la configuration, la mise à jour et la maintenance d'applications ou de systèmes.

# Configuration management

**Définition**

La **gestion de configuration** (*Configuration Management*) est la pratique qui consiste à automatiser l'installation, la configuration, la mise à jour et la maintenance d'applications ou de systèmes.

**Multitude de stratégies**

Chaque solution de configuration management adopte son propre paradigme en composant avec différents stratégies. Il n'y en a pas une meilleure que l'autre, juste une plus adaptée en fonction de votre cas d'utilisation.

# Différentes approches

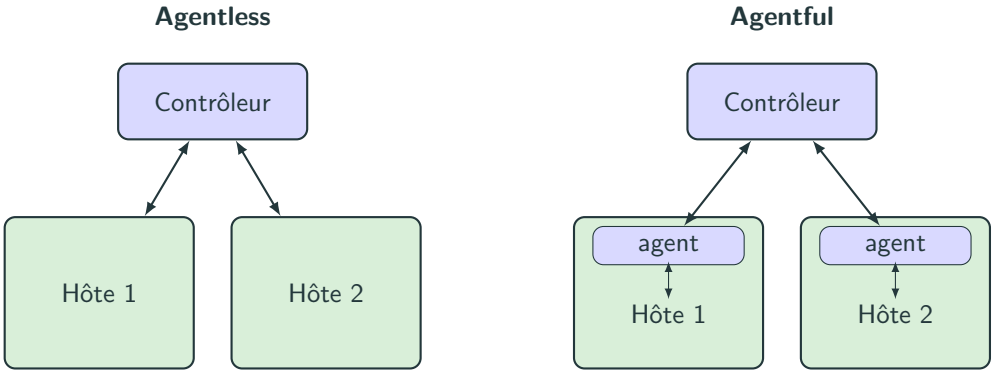
## Pull- ou Push- based

- **Pull** : les nœuds gérés récupèrent régulièrement leur configuration depuis un serveur central;
- **Push** : un contrôleur envoie directement la configuration vers les nœuds.

# Différentes approches

## Agentless vs. Agentful

- **Agentful** : chaque machine gérée exécute un agent dédié, installé sur la machine cible;
- **Agentless** : aucune installation côté cible, on utilise des protocoles standards (ex: SSH).



# Différentes approches

**Static vs. Event-driven**

- **Static** : la configuration est appliquée de manière periodique ou à la demande;
- **Event-driven** : les changements sont déclenchés par des événements (alertes, hooks, messages).

# Configurer, démarrer, et maintenir son application

## Pull-based et agent-driven solutions

- Chef (à la demande + periodic)
- Puppet (periodic)
- CFEngine (periodic)

## Event-driven

- SaltStack (à la demande + periodic + event)
- Juju (à la demande + periodic + event)

## Push-based et agentless solution

- Ansible (à la demande)

# Ansible

Dans ce module, nous utiliserons un outil de configuration management: Ansible.


Ansible est un outil

- Semi-déclaratif : on décrit l'état souhaité avec des **Tasks** et **Playbooks**, tout en pouvant créer ses propres modules si besoin.
- Stateless: l'état courant de l'infrastructure et des applications déployées n'est pas conservé entre deux exécutions.
- Conccurrent : exécute des tâches en parallèle sur plusieurs hôtes via SSH.

# Ansible

# Vue d'ensemble d'Ansible

Les différents éléments de Ansible

- **CLI** (`ansible`, `ansible-playbook`, etc.)
- **Inventaire** des hôtes (fichier INI ou YAML)
- Langage **YAML** pour décrire les tâches
- **Tasks** : appels à des actions atomiques (ex: apt, copy, service)
- **Playbooks** : regroupements ordonnés de tasks

# Vue d'ensemble d'Ansible

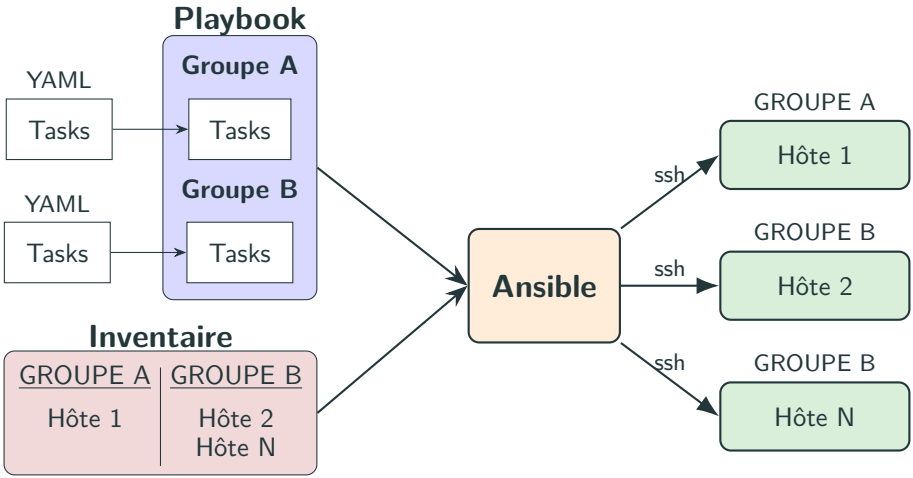
Les différents éléments de Ansible

- **CLI** (`ansible`, `ansible-playbook`, etc.)
- **Inventaire** des hôtes (fichier INI ou YAML)
- Langage **YAML** pour décrire les tâches
- **Tasks** : appels à des actions atomiques (ex: apt, copy, service)
- **Playbooks** : regroupements ordonnés de tasks

## Les rôles d'Ansible

- Définir un ensemble de tâches à éxécuter sur des hôtes distants
  - Commandes à éxécuter
  - Installation de dépendances
  - Installation d'application
  - etc.
- Grouper les hôtes par rôles pour limiter les répétitions
- Assurer la gestion de la concurrence dans le processus de reconfiguration

# Vue d'ensemble d'Ansible



# Inventaire

L'inventaire définit les machines cibles et leurs groupes. Il peut être au format INI ou YAML.

```
# inventory.ini
[web]
web1.example.com
web2.example.com

[db]
db1.example.com ansible_user=admin
```

Note : On peut aussi utiliser des inventaires dynamiques (scripts, plugins cloud).

# Tasks et Playbooks

- **Task** : unité de travail, appel d'une instruction (appelée module) avec des paramètres.
- **Play** : ensemble de tasks, souvent stocké dans un fichier YAML.
- **Playbook** : fichier YAML qui regroupe des plays. Chaque play associe des hôtes, des variables et une liste de tasks.

**Note**: Il existe un registre de rôles et de tâches: `https://galaxy.ansible.com/`

# Exemple de Tasks

Exemple de Play:

```yaml
# taches.yaml
- name: Installer curl
  become: yes          # sudo
  apt:                 # module apt pour Debian
    name: curl         # package a installer
    state: present     # doit etre present

- name: Creer un fichier de test
  file:                # module pour creer un fichier
    path: /tmp/hello.txt   # chemin du fichier
    state: touch           # creer vide
    # content: "Hello"     # remplace le contenu
    # line: "Hello"        # ajoute le contenu
```

# Exemple de Playbook

A partir du fichier `taches.yaml`, je peux créer un playbook:

```yaml
# site.yaml
- hosts: web
  become: yes # sudo
  tasks:
    # on appelle les taches du fichier externe
    - include_tasks: taches.yaml
    - name: Ecrire un message dans le fichier
      copy:
        content: "Bonjour depuis Ansible !\n"
        dest: /tmp/hello.txt
- hosts: db
  ...
```

# Quelques modules

# Module apt

Options principales :

- `name` : nom du paquet
- `state` : état désiré (`present`, `absent`, `latest`)
- `update_cache` : mettre à jour l'index (`yes`/`no`)

```
- name: Installer nginx
  apt:
    name: nginx
    state: present
    update_cache: yes
```

# Module git

Options principales :

- `repo` : URL du dépôt
- `dest` : répertoire de destination
- `version` : branche/tag/commit
- `update` : mettre à jour si déjà cloné

```
- name: Cloner un depot
  git:
    repo: "https://github.com/ansible/ansible-examples.git"
    dest: /opt/ansible-examples
    version: master
    update: yes
```

# Module shell

Options principales :

- `cmd` : commande à exécuter
- `chdir` : répertoire d'exécution
- `creates` : exécuter seulement si fichier n'existe pas
- `removes` : exécuter seulement si fichier existe

```
- name: Lancer un script shell
  shell: ./install.sh
  args:
    chdir: /opt/scripts
```

# Module copy

Options principales :

- `src` : chemin local (copié depuis la machine de contrôle)
- `dest` : chemin de destination
- `content` : contenu direct à écrire
- `owner`, `group`, `mode` : permissions

```
- name: Creer un fichier avec contenu
  copy:
    dest: /tmp/hello.txt
    content: "Bonjour Ansible!\n"
    owner: root
    group: root
    mode: '0644'
```

# Module file

Options principales :

- `path` : chemin du fichier ou dossier
- `state` : file, directory, absent, touch, link
- `owner`, `group`, `mode`

```
- name: Creer un repertoire
  file:
    path: /opt/app
    state: directory
    mode: '0755'
```

# Module lineinfile

Options principales :

- `path` : chemin du fichier
- `line` : ligne à ajouter
- `regexp` : expression régulière pour remplacer
- `insertafter`, `insertbefore` : position

```
- name: Ajouter une ligne dans /etc/hosts
  lineinfile:
    path: /etc/hosts
    line: "192.168.1.10 myapp.local"
```

# CLI

**ansible**

Exécuter une commande ad-hoc sur un ou plusieurs hôtes

```
> ansible all -m ping
```

# CLI

**`ansible`**

Exécuter une commande ad-hoc sur un ou plusieurs hôtes

```
> ansible all -m ping
```

**`ansible-playbook`**

Lancer un Playbook YAML complet

```
> ansible-playbook site.yaml
```

# CLI

**ansible**

Exécuter une commande ad-hoc sur un ou plusieurs hôtes

```
> ansible all -m ping
```

**ansible-playbook**

Lancer un Playbook YAML complet

```
> ansible-playbook site.yaml
```

**ansible-inventory**

Lister ou valider l'inventaire

```
> ansible-inventory -list -y
```

# Structure d'un projet

Exemple d'un projet très simple:

```
ansible-project/
├── inventory.ini
├── playbook.yaml
└── taches.yaml
```