



Global Knowledge®



Configuration Management with ANSIBLE

ANSIBLE

Agenda

- Configuration Management
- Ansible
- Prise en main

Configuration Management



Systèmes de Gestion de Configuration

- Selon Wikipedia : un système de gestion des configurations est défini comme suit :
 - « La gestion de configuration consiste à gérer la description technique d'un système (et de ses divers composants), ainsi qu'à gérer l'ensemble des modifications apportées au cours de l'évolution du système. »
- En d'autres termes, il s'agit de l'ensemble des processus permettant d'assurer la conformité d'un produit aux exigences, tout au long de son cycle de vie.
- La gestion de configuration est utilisée pour la configuration de systèmes complexes

Systèmes de Gestion de Configuration

- Résoud les problèmes d'administration systèmes de grands parcs :
 - Taches **répétitives**
 - Supervision de **l'état** du parc
 - Audit
 - etc.

Systèmes de Gestion de Configuration

- Méthodes possibles d'administration de grands parcs
 - Manuellement → longue, lente et source d'erreurs humaines

Systemes de Gestion de Configuration

- Méthodes possibles d'administration de grands parcs
 - **Boucle SSH (script)**
 - Long!
 - Comment vérifier les résultats ?
 - Reprise sur erreur
 - Prérequis: s'assurer que tous les systèmes sont iso (ou s'assurer que tous les cas de figure sont gérés)
 - **automate.sh (script au boot ou par cron/scheduler)**
 - Comment s'assurer que tous les nœuds ont bien exécuté le script?
 - Mêmes problèmes
 - Comment distinguer les nœuds?

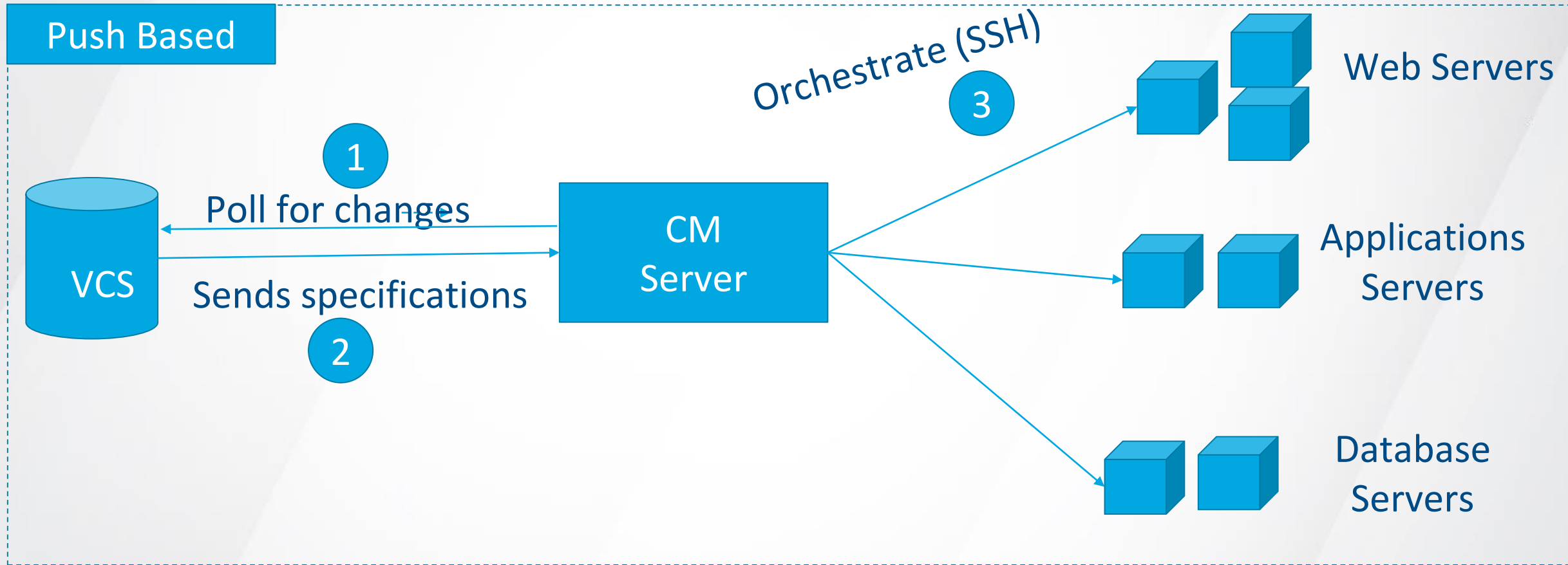
Systèmes de Gestion de Configuration

- Méthodes possibles d'administration de grands parcs
 - CM Tools
 - Ansible
 - Puppet
 - Saltstack
 - Chef
 - ...

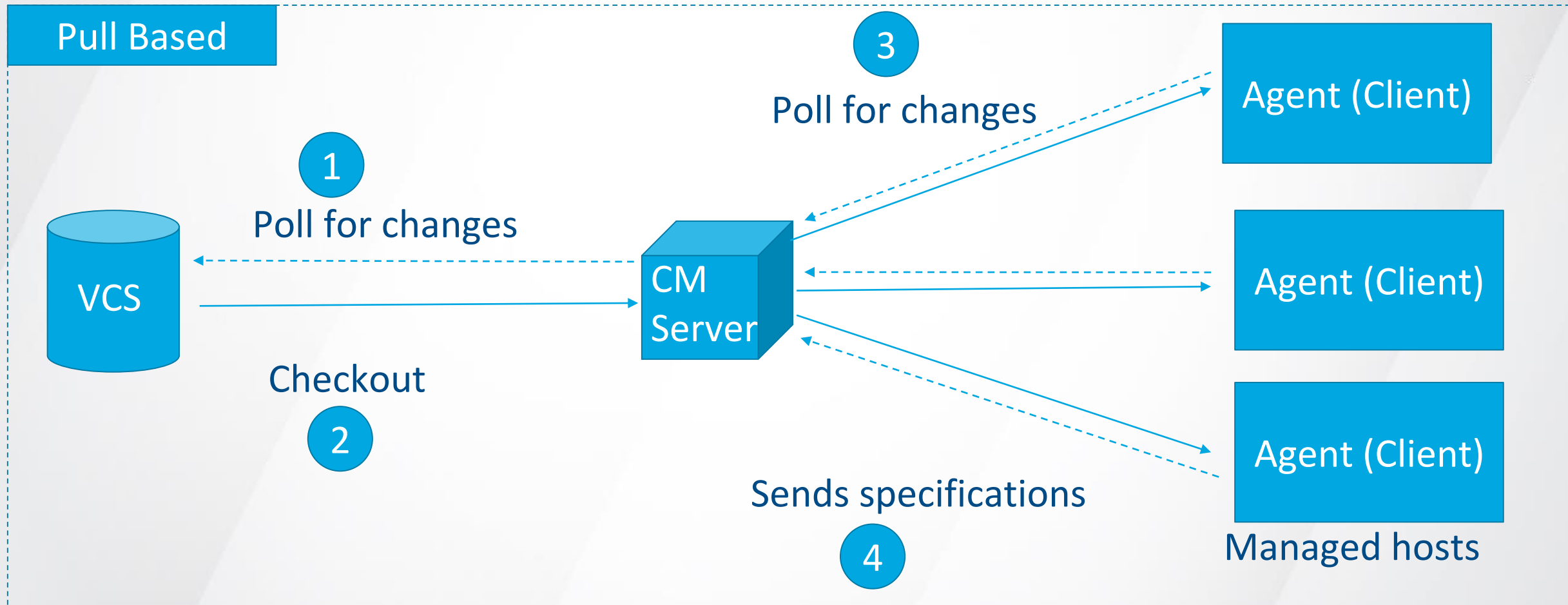
Systemes de Gestion de Configuration

- Configuration Management Tools
- Deux mécanismes
 - Push based
 - Pull based

Push Based vs Pull Based



Push Based vs Pull Based



Modèle Push vs Modèle Pull

- Des outils comme Puppet et Chef sont basés sur le modèle Pull:
 - Les agents sur le serveur vérifient périodiquement les informations de configuration du serveur central (maître).
- Ansible est basé sur le modèle Push:
 - Le serveur central envoie les informations de configuration sur les serveurs cibles
 - Vous contrôlez le moment où les modifications sont effectuées sur les serveurs



ANSIBLE



Objectives

- Qu'est ce que Ansible?
- Ansible features
- Architecture de Ansible
- Commandes Ad-hoc
- Playbook & Modules
- Ansible Roles & Galaxy
- Ansible Vault

Qu'est ce que Ansible?

- Ansible est un moteur d'automatisation de
 - Provisionnement
 - Configuration
 - Déploiement d'applications
 - Orchestration interserveurs
 - ...

Qu'est ce que Ansible?

- Gestion de configuration
 - Configuration logicielle
 - Historique des changement de configuration
 - Reproduction de la même configuration sur plusieurs machines

Qu'est ce que Ansible?

➤ Provisioning

- tous les paquetages nécessaires et tous les logiciels sont téléchargés et installés sur votre ordinateur afin d'exécuter votre application

Qu'est ce que Ansible?

- utilise des '**Playbooks**' pour
 - déployer,
 - gérer,
 - créer,
 - tester
 - configurer

Pourquoi encore un autre outil?

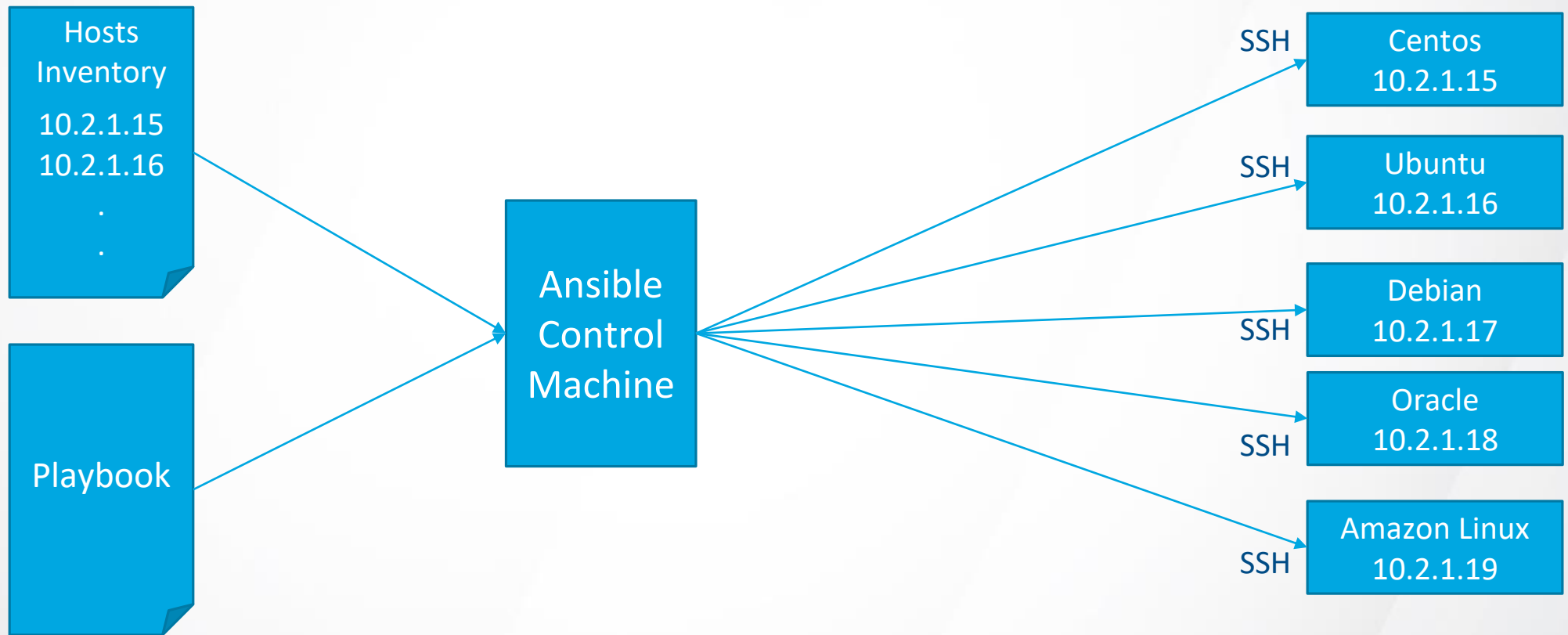
“I wrote Ansible because none of the existing tools fit my brain. I wanted a tool that I could not use for 6 months, come back later, and still remember how it worked.”

➤ Michael DeHaan, Ansible Founder

“We need to do a rolling deployment of changes that have certain dependencies (including external services).
With Ansible this becomes trivial.”

➤ User IUseRhetoric on [reddit.com](https://www.reddit.com)

Ansible : Architecture Agentless



Vocabulaire d'Ansible

- **Poste de gestion** : la machine sur laquelle Ansible est installée.
- **Inventaire** : les serveurs gérés.
- **Tâches (tasks)** : une procédure à exécuter
- **Module** : abstraction de tâche.
- **Playbooks** : les serveurs cibles et les tâches à exécuter.
- **Rôle** : partage et réutilisation des configurations.
- **Facts** : Des informations à propos du système géré
- **Handlers**: Exécution conditionnée

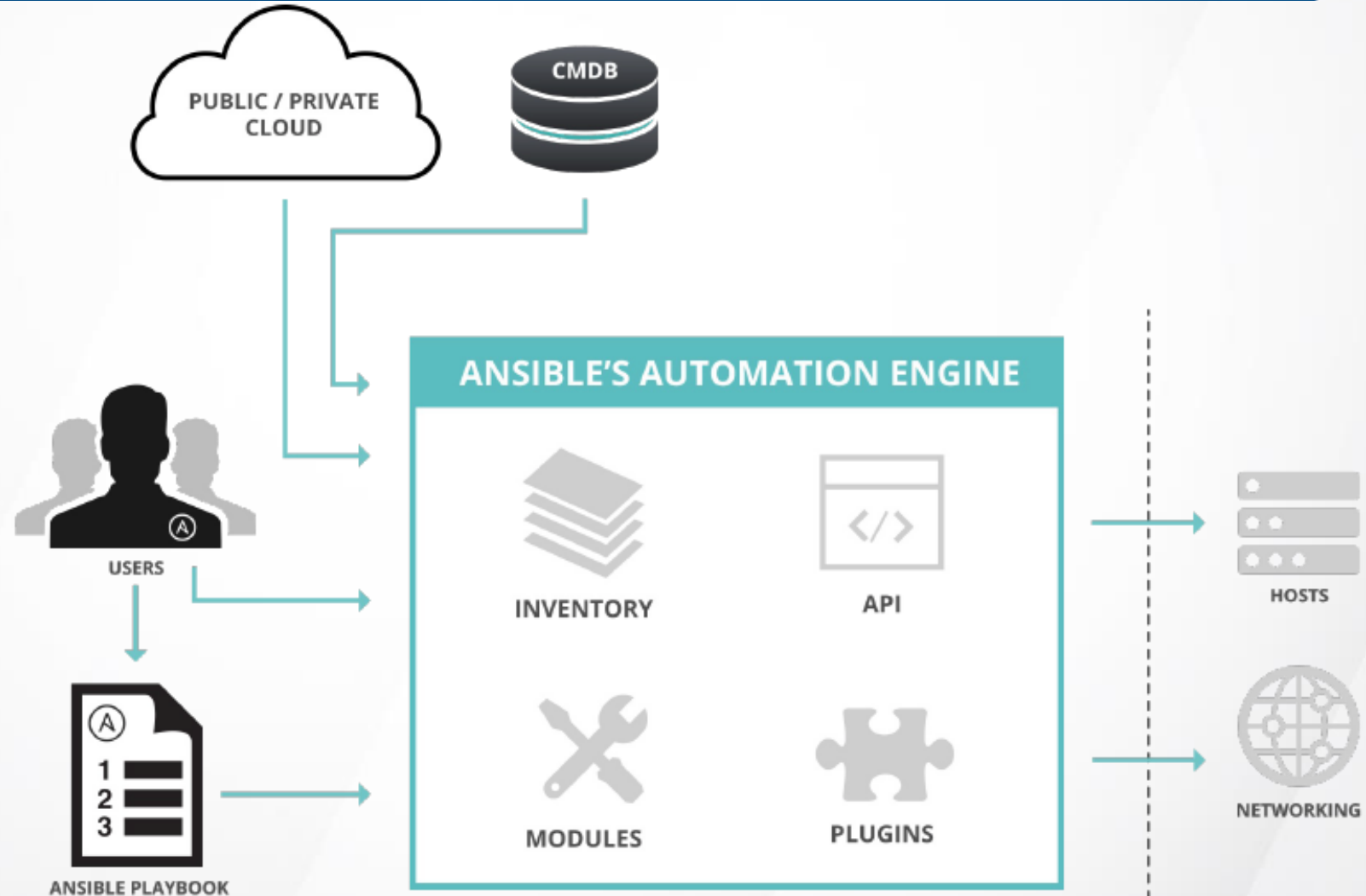
Caractéristiques Ansible (1/2)

- **Agentless**: Sans agent: pas besoin d'installation ni de gestion d'agent
- Construit avec **Python** et fournit donc beaucoup de fonctionnalités de Python
- Utilise **SSH** pour des connexions sécurisées
- Suit le modèle **Push** pour l'envoi de configurations
- Très facile et rapide à installer, exigences minimales

Caractéristiques Ansible (2/2)

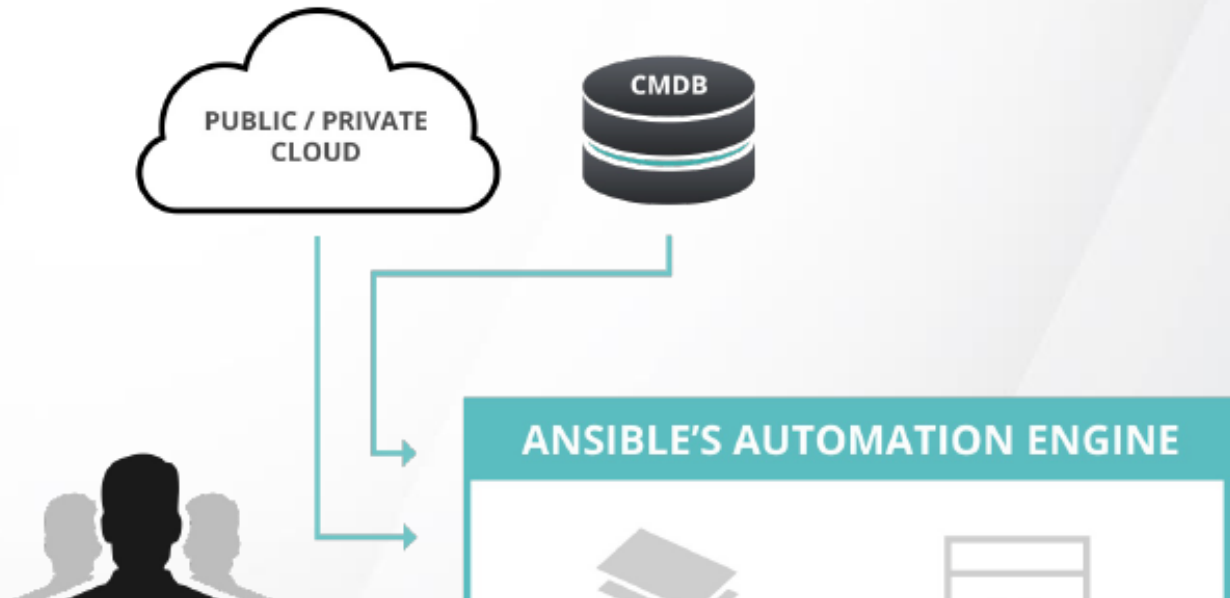
- Fichiers de configuration simples à écrire et à lire : pas de programmation complexe! (En YAML)
- Notion de Playbook : ensemble de commandes à appliquer pour une certaine tâche
 - Ex : playbook configuration base serveur centos ou playbook apache
- Notion de **Role** : niveau d'abstraction au dessus d'un playbook
 - Ex: permet d'appliquer au serveur centos-apache les rôles centos et apache .
 - Ex: si serveur Ubuntu, alors on appliquera les rôles ubuntu et apache

Architecture d'Ansible (1/2)



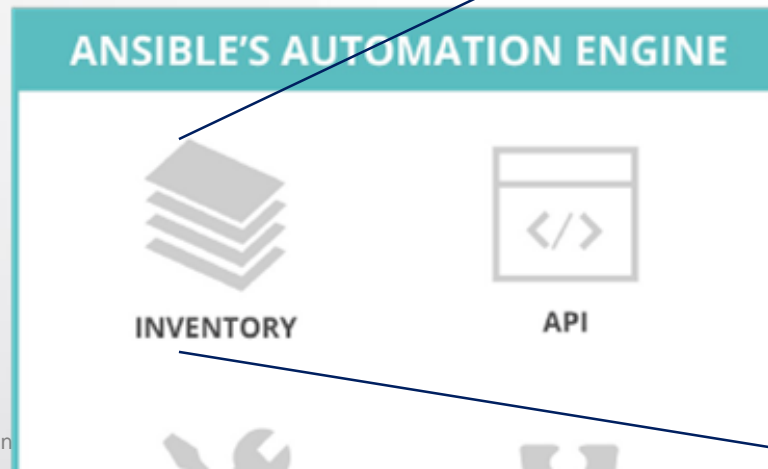
Architecture d'Ansible (2/2)

- Le moteur d'automatisation Ansible interagit directement avec les utilisateurs:
 - écrire des playbooks pour exécuter le moteur d'automatisation Ansible .
- Il interagit également avec les services Cloud et la base de données de gestion de la configuration (CMDB).



Hosts Inventory

- L'inventaire est une collection d'hôtes (nœuds) avec des données et des regroupements que Ansible peut gérer.
- L'inventaire par défaut est **`/etc/ansible/hosts`**



Exemple: Static Inventory

Group

```
# prod inventory
```

```
[balancers]  
www.example.com
```

Host

```
[webservers]  
www[0-9].example.com
```

Intervalle
numérique

```
[dbservers]  
db[a:f].example.com
```

Intervalle
alphabétique

```
[monitoring]  
dynatrace.example.com
```

ANSIBLE'S AUTOMATION ENGINE



INVENTORY



API



MODULES



PLUGINS

Dynamic Inventories

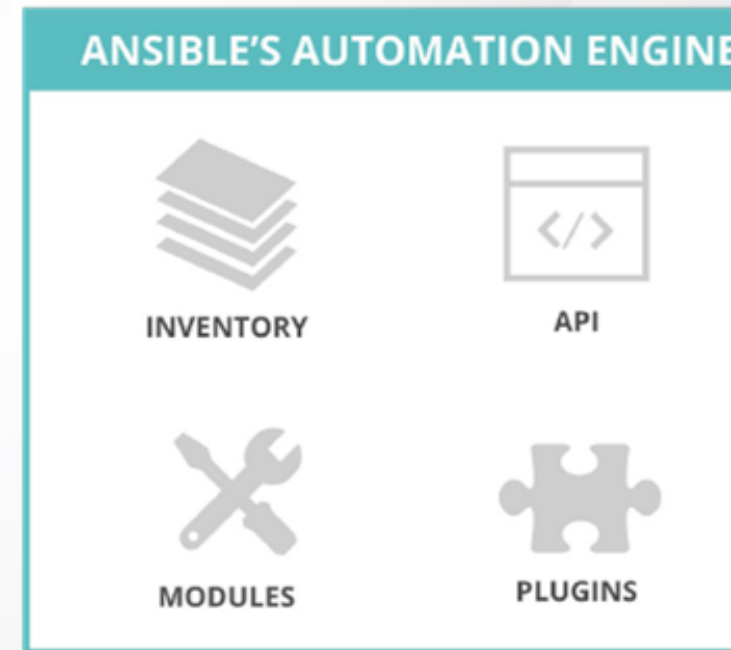
- Utilisation de plusieurs fichiers d'inventaire simultanément et extraction des inventaires à partir de sources dynamiques ou de cloud:
 - Cloud: Amazon, DigitalOcean, Google, OpenShift, OpenStack, etc.
 - Distributed Information Services: LDAP, etcd, etc.
- <https://github.com/ansible-collections/community.general/tree/main/scripts/inventory>

Les API

- l'API Ansible: Python:Contrôle les nœuds
- Les API REST : fournir un accès aux ressources via des chemins URI
- Ansible Tower : Rendre Ansible utilisable via un service Web.

Ansible Modules

- Les modules (également appelés "task plugins" ou "library plugins") sont ceux qui sont réellement exécutés dans un playbook
- Ce sont des scripts fournis avec Ansible et effectuent des actions sur les hôtes
- Exemples:
 - **apt**: install and remove packages using the apt package manager
 - **copy**: copies a file from local machine to hosts
 - **file**: sets the attribute of a file, symlink, or directory
 - **service**: starts, stops or restarts a service



Les plugins

➤ plugins d'Action :

- agissent conjointement avec les modules pour exécuter les actions requises par les tâches du playbook
- Exécution automatiquement en arrière-plan

➤ plugins de cache:

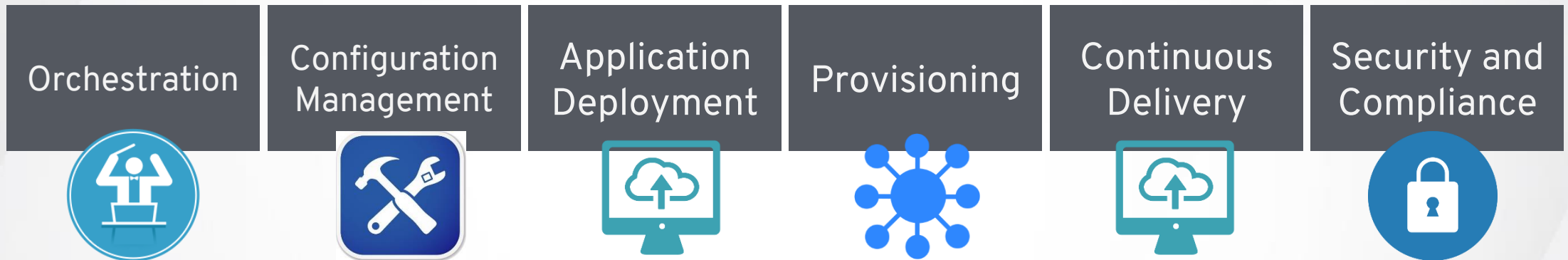
- implémentation du mécanisme de cache d'arrière-plan
- Permettant à Ansible de stocker des faits collectés
- stocker des données source d'inventaire sans avoir récupérer les données à partir de la source.

Les plugins

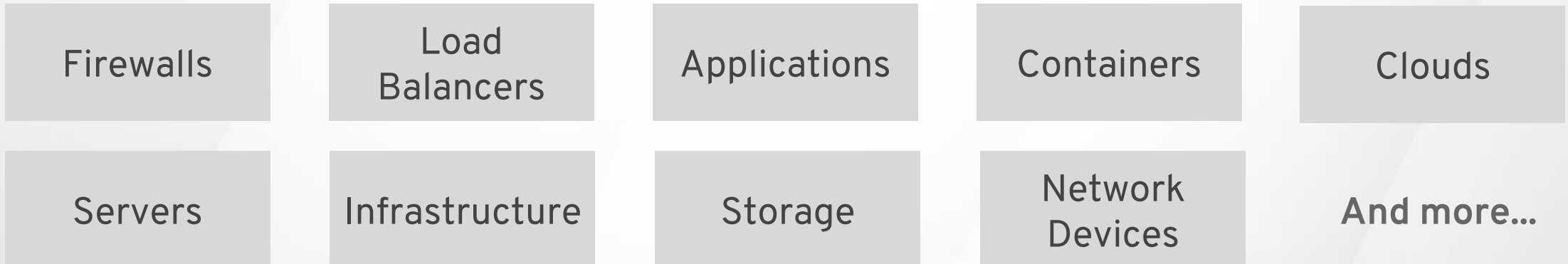
- Callback plugins:
 - Permettent de connecter à des événements Ansible afin d'affichage le journal.
- Plugins de rappel:
 - permettant d'ajouter de nouveaux comportements à Ansible lors de la réponse à des événements
- Plugins de connexion:
 - permettant de se connecter aux hôtes cibles pour lui permettre d'exécuter des tâches.
- Les plugins de recherche:
 - permettant d'accéder à des données provenant de sources extérieures.

Ansible can be used for

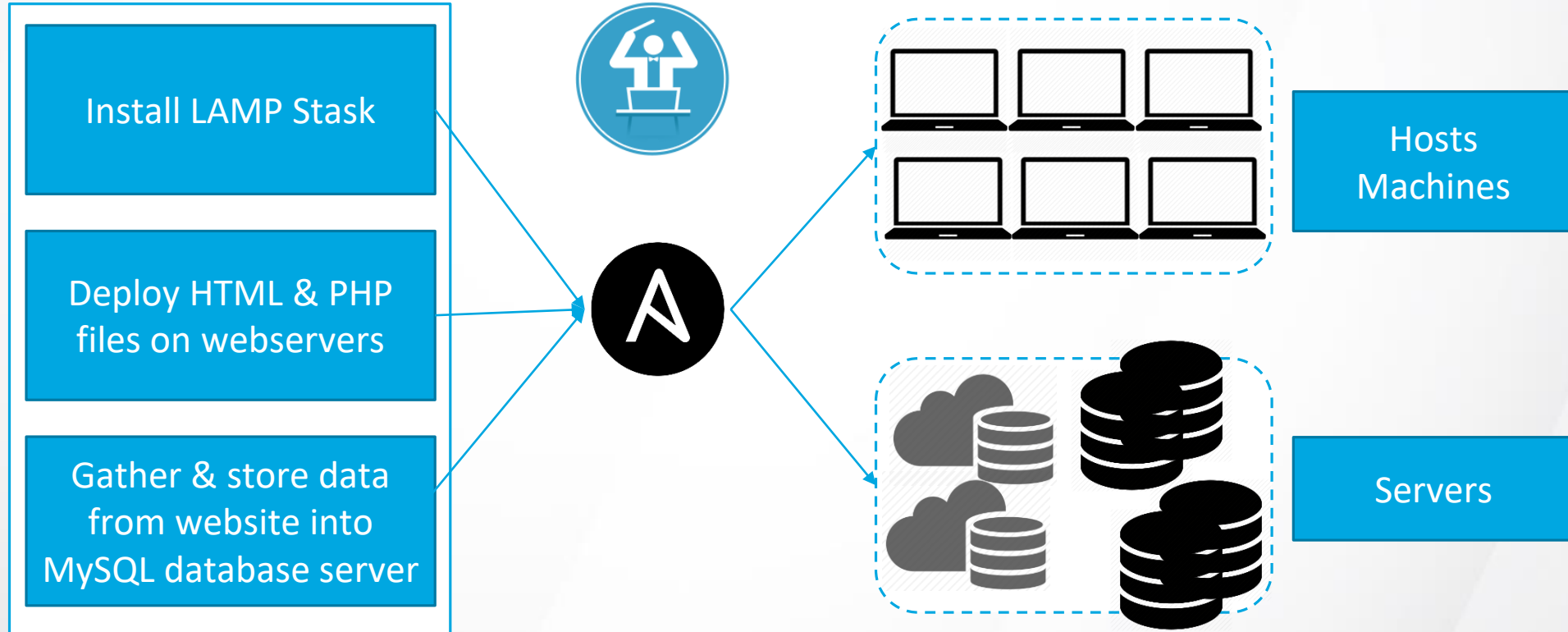
Do this...



On these...

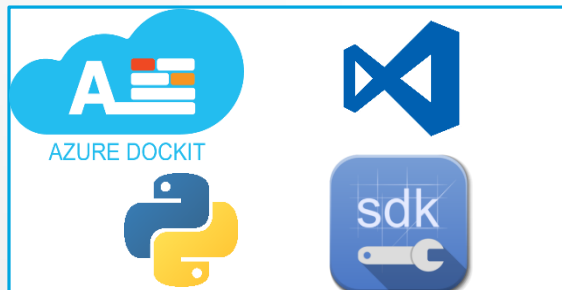


Orchestration

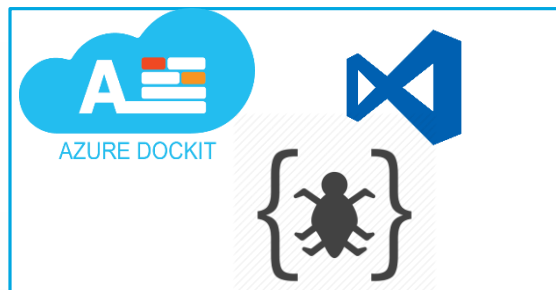


Provisioning

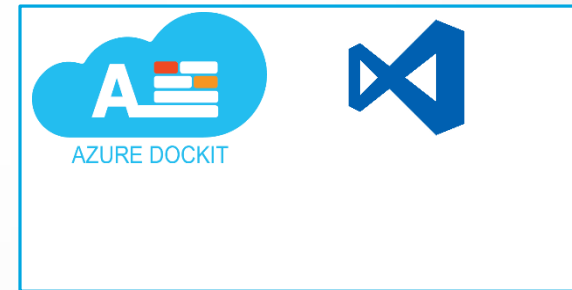
- Provisionner une application web Python hébergée sur le Cloud



Coding



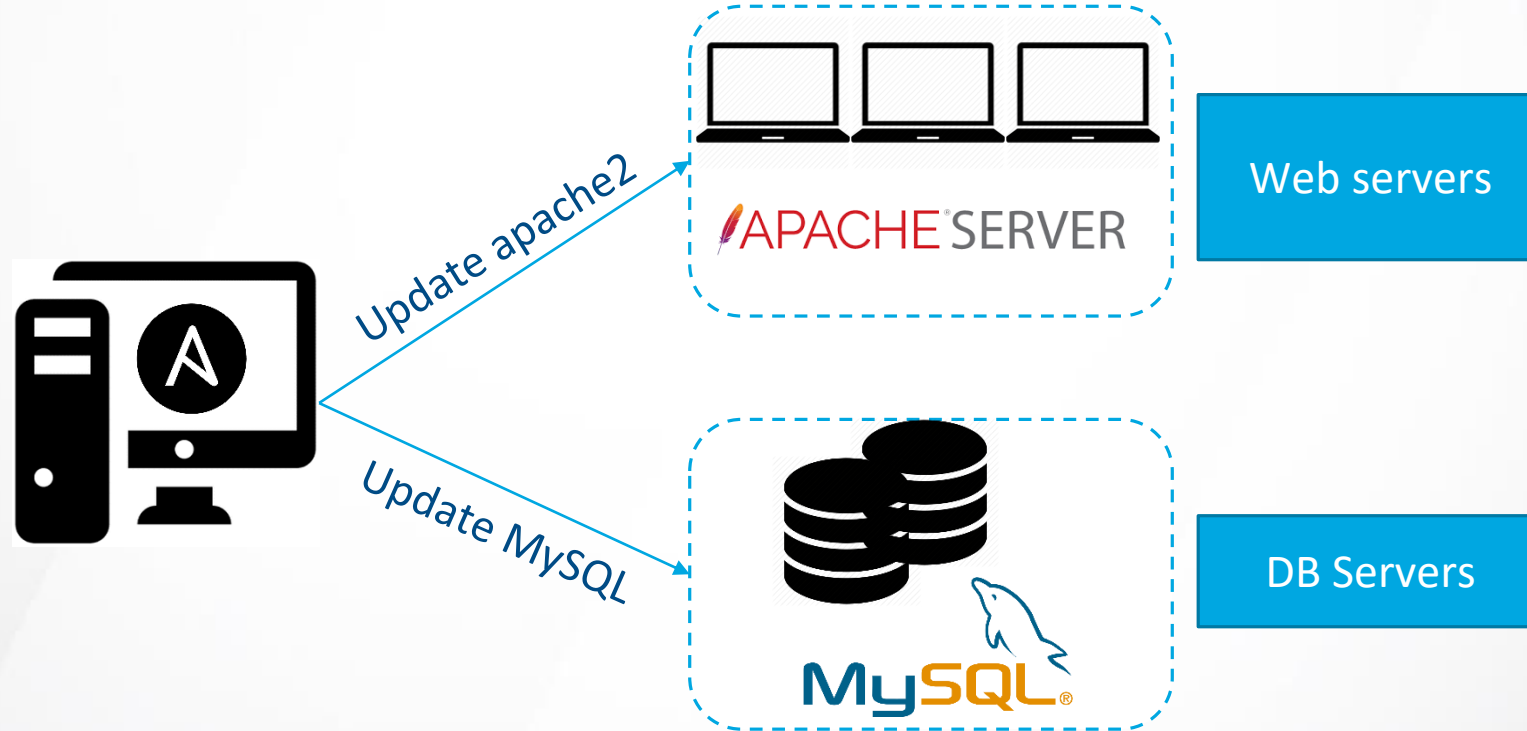
Testing



Deploying

Configuration Management

➤ LAMP Stack



Ansible Hands-on



Installation

- Ansible est un outil disponible en ligne de commande.
- Automatisation de configuration de système et le déploiement de logiciels est son objectif principal.
- Pour tout type d'installation de Ansible, consulter la documentation sur https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

Installation

➤ RHEL et CentOS

- Acquérir Ansible sur RHEL et CentOS en installant à partir du canal Ansible
- Il faut activer le Ansible Engine repository:

➤ **subscription-manager repos --enable=rhel-7-server-ansible-2.x-rpms**

- OU installer le EPEL repository

➤ yum install <https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm>

- Installer par yum:

➤ **sudo yum install ansible**

Installation

- Ansible sur Ubuntu
- Acquérir Ansible sur Ubuntu en utilisant le PPA Ansible

```
$ sudo add-apt-repository ppa:ansible/ansible  
$ sudo apt update  
$ sudo apt install ansible
```

Installation

➤ Installer Ansible avec “**pip**”

- Installer les paquets Python

- `curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py`

- `python get-pip.py --user`

- Installer Ansible avec pip

- `python -m pip install --user ansible`

Installation

- Tester l'installation Ansible en récupérant sa version à l'aide de la commande suivante.

➤ `ansible --version`

Accès SSH



IMPORTANT

Configuration de l'accès SSH aux hôtes Ansible

Accès SSH

- Ansible control machine contrôle les nœuds gérés via SSH.
- L'authentification SSH peut être basée sur:
 - Un mot de passe,
 - Une paire de clés SSH.
- Sous Linux, la configuration de l'accès SSH est dans le fichier:
 - **/etc/ssh/sshd_config**

Accès SSH: configuration

- Générer une pair de clés SSH

ssh-keygen

- Installer la clé SSH dans les nœuds gérés
 - Par copie manuelle de la clé publique dans le fichier « **~/.ssh/authorized_keys** »
 - Par copie automatique en utilisant la commande « **ssh-copy-id** »

Accès SSH: configuration

- Sur le **serveur** Ansible, afficher la clé publique SSH de votre utilisateur:

```
$ cat ~/.ssh/id_rsa.pub
```

- Sur le nœud géré, ouvrez le fichier **authorized_keys**:

```
$ vi ~/.ssh/authorized_keys
```

- collez votre clé SSH de l'utilisateur du serveur Ansible, puis enregistrez le fichier.
- Testez la connexion ssh sans mot de passe à partir du serveur:

```
$ ssh user@node_ip
```

➤ Lab# 1: Installation de Ansible



Configurer Ansible

- Ansible prend en charge plusieurs sources pour configurer son comportement:
 - un fichier ini nommé **ansible.cfg**,
 - des variables d'environnement,
 - des options de ligne de commande,
 - des mots-clés dans le playbook
 - des variables

Ansible Inventory

- Ansible conserve la trace de tous les serveurs connus via un fichier hosts.
- Devoir configurer ce fichier avant de pouvoir commencer à communiquer avec les nœuds.
- Le fichier d'inventaire par défaut est `/etc/ansible/hosts`

Ansible Inventory: Host et Group

- Une hôte est simplement une machine distante gérée par Ansible.
- Les hôtes peuvent avoir des variables individuelles qui leur sont assignées, et peuvent également être organisés en groupes.
- Un groupe se compose de plusieurs hôtes affectés à un pool qui peuvent être facilement ciblés ensemble, ainsi que des variables qu'ils partagent en commun.

[webserver]

192.168.35.140
192.168.35.141
192.168.35.142
192.168.35.143

[appserver]

192.168.100.1
192.168.100.2
192.168.100.3

[dbserver]

172.35.0.5

Ansible Inventory: Host & Group

Des ports SSH Diffèrent :

```
web1:2222
```

Alias

```
web2 ansible_port=22 ansible_host=192.168.35.102
```

Ranges:

```
[webservers]  
www[01:50].example.com
```

Ansible Inventory: Host & Group Variables

- **Attribuer des variables aux hôtes qui seront utilisées plus tard dans les playbooks**

```
[webservers]
web1 http_port=80 https_port=443
web2 http_port=8080 https_port=8443
```

- **Les variables peuvent également être appliquées à un groupe entier à la fois**

```
[webservers:vars]
ntp_server=fr.pool.ntp.org
proxy=proxy.example.com
```


Ansible Inventory: Host & Group Variables

➤ **ansible_user**

- Le nom d'utilisateur ssh par défaut à utiliser.

➤ **ansible_ssh_private_key_file**

- Fichier de clé privée utilisé par ssh. Utile si vous utilisez plusieurs clés et que vous ne souhaitez pas utiliser l'agent SSH.

Ansible Inventory: Host & Group Variables

➤ Connexion SSH :

- `ansible_ssh_host` : Nom de l'hôte.
- `ansible_ssh_port` : Numéro du port (si différent de 22).
- `ansible_ssh_user` : Nom de l'utilisateur.
- `ansible_ssh_pass` : Password à utiliser.
- `ansible_ssh_private_key_file` : fichier de clé privée à utiliser.

Ansible Inventory: Host & Group Variables

➤ Escalade de privilèges

- `ansible_become` : Equivalent de `ansible_sudo`.
- `ansible_become_method` : Permet de définir la méthode d'escalade de privilège.
- `ansible_become_user` : Permet de définir l'utilisateur à devenir.
- `ansible_become_pass` : Permet de définir le mot de passe élevant les privilèges utilisateur.

Ansible Inventory: Host & Group Variables

➤ Paramètres d'environnement d'hôte distant :

- `ansible_shell_type` : Type de shell distant.
- `ansible_python_interpreter` : L'interpréteur python de l'hôte distante.

Configurer Ansible: le fichier de configuration

- Des changements dans la configuration peuvent être effectués dans:
 - Variable d'environnement: `ANSIBLE_CONFIG`
 - `ansible.cfg` (dans le répertoire courant)
 - `~/.ansible.cfg` (dans le home directory)
 - `/etc/ansible/ansible.cfg`
- Le fichier de configuration est divisé en sections repérées par un mot entre crochets:
 - `[defaults]`
 - `[ssh_connection]`
 - `[selinux]`
 - `[inventory]`

Configurer Ansible: le fichier de configuration

- La commande **ansible-config** permet de consulter la configuration de Ansible
- Les sous commande suivantes sont disponibles:
 - list Print all config options
 - dump Dump configuration
 - view View configuration file

Configurer Ansible: le fichier de configuration

➤ Exemples de configuration:

```
[defaults]
inventory = /etc/ansible/hosts
forks = 5
sudo_user = root
ask_sudo_pass = True
ask_pass = True
remote_port = 22
```

Configurer Ansible: le fichier de configuration

➤ Exemples de configuration:

```
[defaults]
```

```
# smart - gather by default, but don't regather if already gathered
```

```
# implicit - gather by default, turn off with gather_facts: False
```

```
# explicit - do not gather by default, must say gather_facts: True
```

```
gathering = implicit
```

```
# uncomment this to disable SSH key host checking
```

```
host_key_checking = False
```


Configurer Ansible: le fichier de configuration

➤ Exemples de configuration:

```
# default user to use for playbooks if user is not specified  
# (/usr/bin/ansible will use current user as default)  
remote_user = vagrant
```

```
# logging is off by default unless this path is defined  
# if so defined, consider logrotate  
log_path = /var/log/ansible.log
```

➤ Lab# 2: Ansible Configuration



Ansible: Les commandes ad-hoc



Ansible Tasks

- Un Task ou une tâche est une action discrète qui est une déclaration sur l'état d'un système.
- Exemples de tâches:
 - Le répertoire doit exister
 - Le package doit être installé
 - Le service doit être en cours d'exécution
 - L'instance cloud doit exister

Commandes Ad-hoc

- Une commande ad-hoc est une tâche Ansible unique à effectuer rapidement, mais on ne souhaite pas l'enregistrer pour plus tard.
 - # check all my inventory hosts are ready to be managed by Ansible
 - \$ ansible all -m ping
 - # collect and display the discovered facts for the localhost
 - \$ ansible localhost -m setup
 - # run the uptime command on all hosts in the web group
 - \$ ansible web -m command -a "uptime"
 - Uptime off all the machines
 - \$ ansible all --s --n shell --a 'uptime'

Les modules

- Les modules sont les bouts de code copiés sur le système cible pour être exécutés pour satisfaire la déclaration de tâche.
 - Le code n'a pas besoin d'exister sur l'hôte distante, il est possible de le copier
 - De nombreux modules sont livrés avec Ansible
 - Les modules personnalisés peuvent être développés facilement
 - Les modules Command / Shell existent pour des commandes simples
 - Le module Script existe pour utiliser le code existant
 - Le module Raw existe pour exécuter des commandes brutes sur ssh

Les modules

- Liste des modules et documentation accessible par ansible-doc:
 - \$ ansible-doc -l
 - \$ ansible-doc apt
- Module indexe:
 - https://docs.ansible.com/ansible/2.9/modules/modules_by_category.html

Lab# 3:

Ad-Hoc Commands



Ansible: Les Playbooks



Plays & Playbooks

- Plays sont des ensembles ordonnés de tâches à exécuter par rapport à des sélections d'hôtes de votre inventaire.
- Un Playbook est un fichier contenant un ou plusieurs Plays.

Ansible playbook

- Langage de configuration, de déploiement et d'orchestration d'Ansible
- Ecrit en YAML, définit de manière déclarative vos configurations
- Lisibles par l'homme et sont développés dans un langage textuel
- Les modules Ansible sont utilisés dans le Playbook pour exécuter l'opération.
- Adaptés au déploiement d'applications complexes.
- Plus susceptibles d'être conservés dans le système de contrôle de version

Ansible playbook

- Assure le Suivi de la configuration des serveurs
- Un playbook est un scénario décrivant les actions qui seront réalisées par les serveurs
 - les paquets à installer
 - les fichiers à créer
 - les services à démarrer ou arrêter, ...
- Faut faire attention à l'ordre d'exécution des tâches.

Structure de Playbook

Hosts

- list

Variables

- list

Tasks

- list

Handlers

- list

- name: install and start apache

hosts: web

become: yes

vars:

http_port: 80

tasks:

- name: install nginx server

apt: pkg=nginx state=installed

notify:

- start nginx

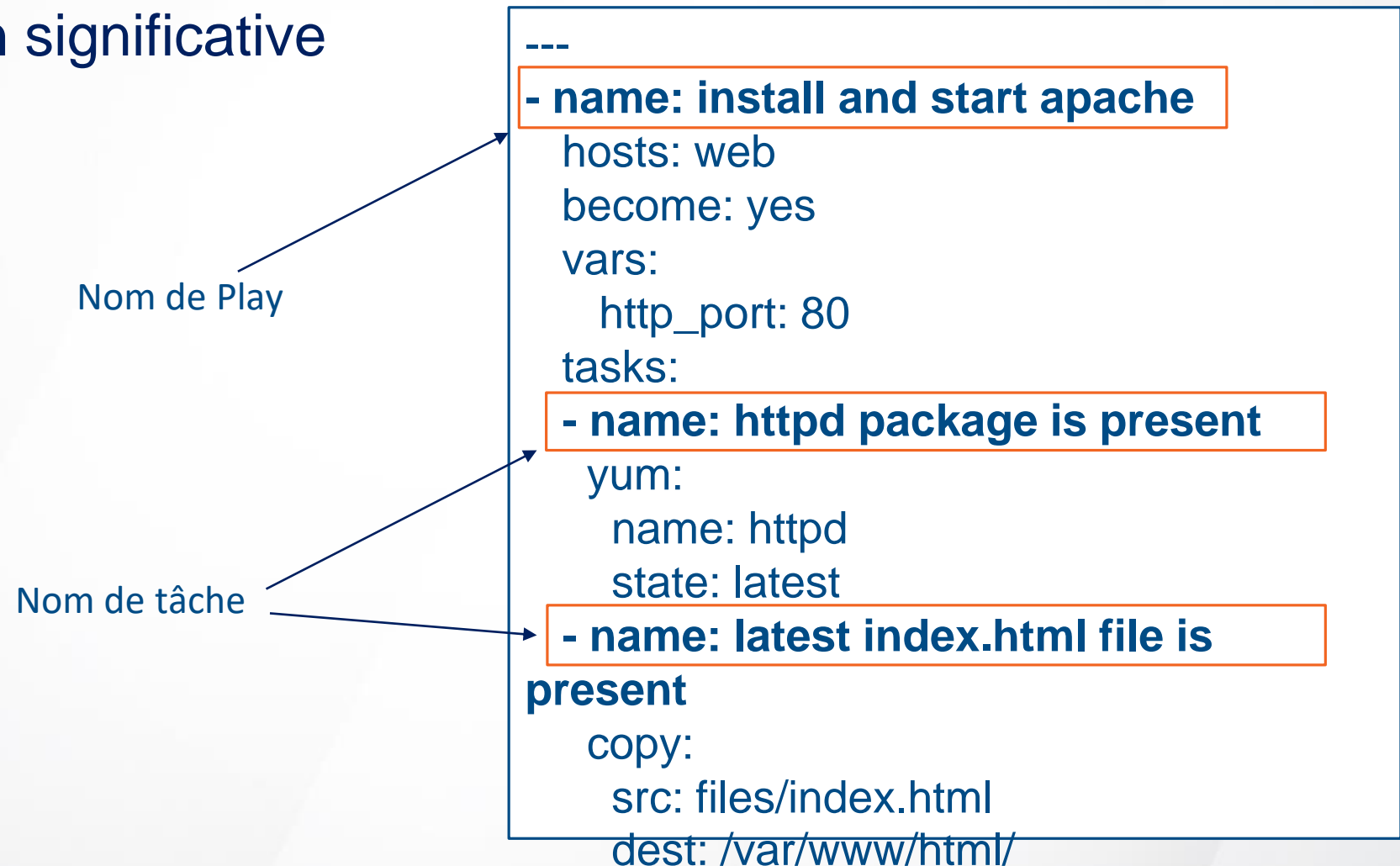
handlers:

- name: start nginx

service: name=nginx state=started

Structure de Playbook

➤ Dénomination significative



Structure de Playbook

➤ Sélection des hôtes cibles

```
---  
- name: install and start apache  
  hosts: web  
  become: yes  
  vars:  
    http_port: 80  
  tasks:  
    - name: httpd package is present  
      yum:  
        name: httpd  
        state: latest  
    - name: latest index.html file is present  
      copy:  
        src: files/index.html  
        dest: /var/www/html/
```

Structure de Playbook

➤ Privilege Escalation

```
---  
- name: install and start apache  
  hosts: web  
  become: yes  
  vars:  
    http_port: 80  
  tasks:  
    - name: httpd package is present  
      yum:  
        name: httpd  
        state: latest  
    - name: latest index.html file is present  
      copy:  
        src: files/index.html  
        dest: /var/www/html/
```


Structure de Playbook

➤ Variables des plays

```
---  
- name: install and start apache  
  hosts: web  
  become: yes  
  vars:  
    http_port: 80  
  tasks:  
    - name: httpd package is present  
      yum:  
        name: httpd  
        state: latest  
    - name: latest index.html file is present  
      copy:  
        src: files/index.html  
        dest: /var/www/html/
```

Structure de Playbook

➤ Tâches (tasks)

Exécution de
module

```
---  
- name: install and start apache  
  hosts: web  
  become: yes  
  vars:  
    http_port: 80  
  tasks:  
    - name: httpd package is present  
      yum:  
        name: httpd  
        state: latest  
    - name: latest index.html file is present  
      copy:  
        src: files/index.html  
        dest: /var/www/html/
```

Exécution de playbook

- `ansible-playbook --help`
 - Afficher l'aide
- `ansible-playbook script.yml`
 - Lancer le playbook `script.yml`, attention il faut que le playbook soit dans le même dossier
- `ansible-playbook /path/to/script.yml :`
 - Lancer le playbook `script.yml` peu importe où vous vous trouvez dans l'arborescence de votre poste

Exécution de playbook

- `ansible-playbook -i mesjolishosts script.yml`
 - Ansible prend par défaut le fichier inventaire `/etc/ansible/hosts`
 - Avec l'option `-i`, on prend le fichier **mesjolishosts** dans le dossier courant
- `ansible-playbook -i mesjolishosts script.yml -e MAVARIABLE=VALEUR:`
 - `-e` ou `--extra-vars` pour ajouter des variables
- `ansible-playbook script.yml -vvv :`
 - `-v` verbose mode,
 - `-vvv` encore plus de verbosité,
 - `-vvvv` pour déboguer

Lab# 3

Ansible Playbooks



Playbooks: plus de fonctionnalités

- Voici quelques fonctionnalités plus essentielles du playbook que vous pouvez appliquer:
 - Templates
 - Loops
 - Conditionals
 - Tags
 - Blocks


Templates

- Ansible intègre le moteur de création de modèles **Jinja2** qui peut être utilisé pour:
 - Définir et modifier les variables dans les plays
 - Logique conditionnelle
 - Générer des fichiers tels que des configurations à partir de variables

Loops

- Les boucles permettent d'exécuter une seule tâche sur plusieurs choses, telles que
 - créer de nombreux utilisateurs,
 - installer de nombreux packages
 - ou répéter une étape d'interrogation jusqu'à ce qu'un certain résultat soit atteint.

```
- yum:  
  name: "{{ item }}"  
  state: latest  
  loop:  
    - httpd  
    - mod_wsgi
```



Conditionals

- Ansible prend en charge l'exécution conditionnelle d'une tâche basée sur l'évaluation à l'exécution, d'une variable, d'un fait ou du résultat de la tâche précédente.

```
- yum:  
  name: httpd  
  state: latest  
  when: ansible_os_family == "RedHat"
```

- Lab #4:
Variables
- Lab #5:
Contrôles
- Lab #6:
Templates



Ansible Roles

- Les rôles sont des moyens de charger automatiquement certains fichiers vars_files, tasks et handlers en fonction d'une structure de fichiers connue.
- Le regroupement du contenu par rôles permet de partager facilement les rôles avec d'autres utilisateurs.
- Les rôles s'attendent à ce que les fichiers soient dans certains noms de répertoire.
- Les rôles doivent inclure au moins un de ces répertoires, mais il est parfaitement normal d'en exclure ceux qui ne sont pas utilisés.
- Lorsqu'il est utilisé, chaque répertoire doit contenir un fichier main.yml, qui contient le contenu pertinent

Ansible Roles - Directories

- **tasks** - contient la liste principale des tâches à exécuter par le rôle.
- **handlers** - contient des gestionnaires, qui peuvent être utilisés par ce rôle ou même n'importe où en dehors de ce rôle.
- **defaults** - variables par défaut pour le rôle
- **vars** - autres variables pour le rôle.
- **files** - contient des fichiers qui peuvent être déployés via ce rôle.
- **templates** - contient des modèles qui peuvent être déployés via ce rôle.
- **meta** - définit certaines métadonnées pour ce rôle.

Ansible Roles - Example project structure:

site.yml

roles/

common/

tasks/

handlers/

files/

templates/

vars/

defaults/

meta/

webservers/

tasks/

defaults/

meta/

Ansible Roles –

Exemple structure de projet :

D'autres fichiers YAML peuvent être inclus dans certains répertoires. Il est courant d'inclure des tâches spécifiques à la plate-forme à partir du fichier `tasks/main.yml`:

```
# roles/example/tasks/main.yml
- name: added in 2.4, previously you used 'include'
  import_tasks: redhat.yml
  when: ansible_os_platform | lower == 'redhat'
- import_tasks: debian.yml
  when: ansible_os_platform | lower == 'debian'
```

```
# roles/example/tasks/redhat.yml
- yum: name="httpd" state=present
```

```
# roles/example/tasks/debian.yml
- apt: name="apache2" state=present
```

Ansible Galaxy

- Ansible Galaxy est un hub pour trouver, réutiliser et partager du contenu Ansible.
- Démarrez votre projet d'automatisation avec du contenu fourni et révisé par la communauté Ansible.
- <http://galaxy.ansible.com>

➤ Lab #7:
A Playbook Using Roles
Using an Ansible Galaxy Role



Ansible Vault

- “Vault” is a feature of ansible that allows keeping sensitive data such as passwords or keys in encrypted files
- To enable this feature, a command line tool, *ansible-vault* is used to edit files
- a command line flag *–ask-vault-pass* or *–vault-password-file* is used
 - `ansible-playbook site.yml --ask-vault-pass`

Lab #8

➤ Ansible Vault

