

+  
◦ : Introduction pratique  
d'Ansible

# Qu'est-ce qu'Ansible

- C'est un langage d'automatisation simple qui peut parfaitement décrire une infrastructure d'applications informatiques dans des Playbooks Ansible.
- C'est un moteur d'automatisation qui exécute Ansible Playbooks.
- Ansible Tower est un framework d'entreprise pour contrôler, sécuriser et gérer votre automatisation Ansible avec une interface utilisateur et des APIs RESTful.

# Ansible est ...

Ansible est un moteur d'automatisation informatique open source, qui peut éliminer les corvées de votre vie professionnelle, et améliorer également considérablement l'évolutivité, la cohérence et la fiabilité de votre environnement informatique.

Vous pouvez utiliser Ansible pour automatiser trois types de tâches :

- **Provisioning** : configurer les différents serveurs dont vous avez besoin dans votre infrastructure.
- **Configuration** : modifier la configuration d'une application, d'un système d'exploitation ou d'un périphérique, démarrer et arrêter les services, installer ou mettre à jour des applications, mettre en œuvre une politique de sécurité, ou effectuer une grande variété d'autres tâches de configuration.
- **Déploiement d'applications** : adopter une démarche DevOps en automatisant le déploiement d'applications développées en interne sur vos environnements de production.



# Avantages d'Ansible

- **Gratuit:** Ansible est un outil open source.
- **Simple :** Ansible utilise une syntaxe simple écrite en YAML. Aucune compétence en programmation particulière n'est nécessaire pour créer les playbooks d'Ansible. Il est également simple à installer.
- **Puissant:** Ansible vous permet de modéliser des workflows très complexes.
- **Flexible:** Ansible vous fournit des centaines de modules prêts à l'emploi pour gérer vos tâches, quel que soit l'endroit où ils sont déployés. Vous pouvez réutiliser le même playbook sur un parc de machines Red Hat, Ubuntu ou autres.
- **Agentless :** vous n'avez pas besoin d'installer d'autres logiciels ou d'ouvrir des ports de pare-feu supplémentaires sur les systèmes clients que vous souhaitez automatiser. Ansible réduit encore l'effort requis pour que votre équipe commence à automatiser immédiatement.
- **Efficace:** Parce que vous n'avez pas besoin d'installer de logiciel supplémentaire, il y a plus de place pour les ressources d'application sur votre serveur.

# ANSIBLE

Que peut faire  
Ansible

---

# Déploiement d'applications

- Ansible vous permet de déployer rapidement et facilement des applications à plusieurs niveaux.
- Vous n'aurez pas besoin d'écrire du code personnalisé pour automatiser vos systèmes; vous listez les tâches à effectuer en écrivant un playbook, et Ansible trouvera comment amener vos systèmes à l'état dans lequel vous voulez qu'ils soient.
- Vous n'aurez pas à configurer manuellement les applications sur chaque machine.
- Lorsque vous exécutez un playbook à partir de votre machine de contrôle, Ansible utilisera le protocole SSH pour communiquer avec les hôtes distants et exécuter toutes les tâches (**Tasks**).

---

# Orchestration

- Consiste à amener différentes éléments à interagir ensemble sans incohérence.
- Par exemple, avec le déploiement d'applications, vous devez gérer non seulement les services frontend, mais également les services backend comme les bases de données, le réseau, le stockage, etc...
- Vous devez également vous assurer que toutes les tâches sont gérées dans le bon ordre.
- Grâce à Ansible vous orchestrez les éléments de votre infrastructure à l'aide des playbooks Ansible, et vous pouvez les réutiliser sur différents types de machines, grâce à la portabilité des modules Ansible.



# Conformité et sécurité

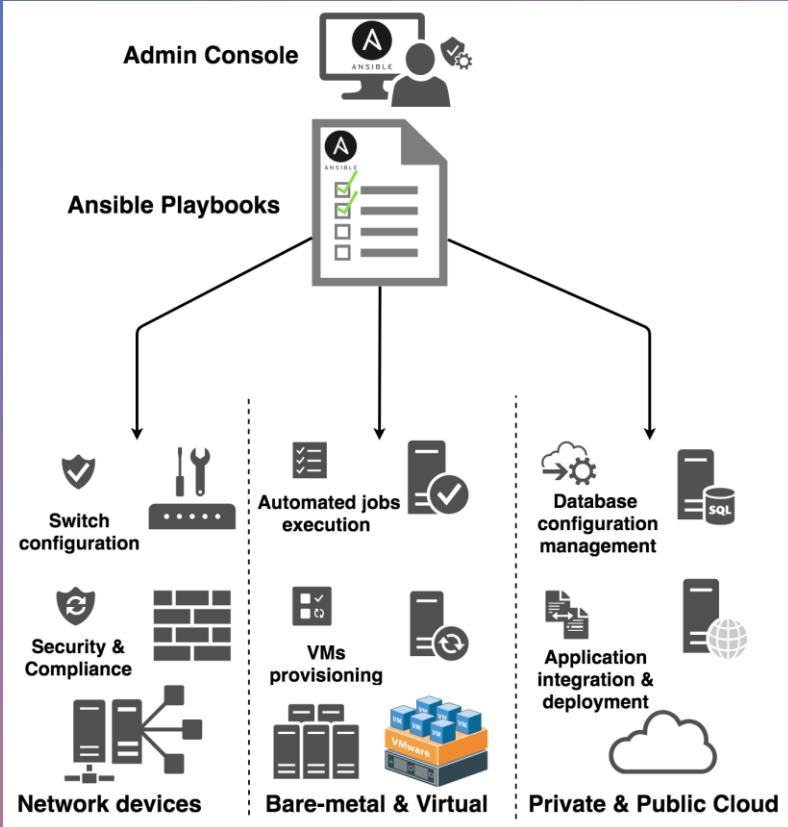
- Comme pour le déploiement d'applications, des politiques de sécurité de votre entreprise (telles que des règles de pare-feu ou le verrouillage des utilisateurs) peuvent être mises en œuvre avec d'autres processus automatisés.
- Si vous configurez les détails de sécurité sur la machine de contrôle et exédez le playbook associé, tous les hôtes distants seront automatiquement mis à jour avec ces détails.
- Cela signifie que vous n'aurez pas besoin de surveiller chaque machine pour vérifier la conformité de la sécurité en continu manuellement.
- De plus, tous les identifiants (identifiants et mots de passe des utilisateurs admin) qui sont stockés dans vos playbooks ne sont récupérables en brut par aucun utilisateur.



# Idempotence

- On décrit l'état souhaité plutôt que les étapes pour y arriver (vs. Script shell)
- Idempotence : une recette (Playbook) aboutit au même résultat qu'on l'applique une ou plusieurs fois
- Les Playbooks doivent être écrites de cette façon pour que l'état soit stable.

# Provisionnement du cloud



- La première étape de l'automatisation du cycle de vie de vos applications consiste à automatiser l'approvisionnement de votre infrastructure.
- Avec Ansible, vous pouvez provisionner des plateformes cloud, des hôtes virtualisés, des périphériques réseau et des serveurs physiques.

---

# Quel rapport avec DevOps

## Centralisation des configurations

- Infrastructure as a code: L'état d'une l'infra décrite par des recettes
- Travail en équipe (utilisation d'un VCS)
- Partage du savoir : Les devs connaissent l'env de prod, les ops connaissent les modification à appliquer à la prod. avant le déploiement d'une appli.

## Automatisation

- Moins de tâches répétitives, manuelles → Raccourcir les cycles de dev./publication/déploiement/utilisation
- Il devient plus simple d'instancier une plateforme de test

# ANSIBLE

## Fonctionnement de Ansible

# Comment fonctionne Ansible

- Dans Ansible, il existe deux catégories d'ordinateurs: le nœud maître (**master**) et les nœuds esclaves (**slaves**).
- Le nœud maître est une machine sur laquelle est installé l'outil Ansible. Il doit y avoir au moins un nœud maître, bien qu'un nœud maître de sauvegarde puisse également exister.
- Ansible fonctionne en se connectant à vos nœuds en SSH et en y poussant de petits programmes, appelés **modules**. Ces modules sont définis dans un fichier nommé le **Playbook**. Le nœud maître, se base sur un fichier d'**inventaire** qui fournit la liste des hôtes sur lesquels les modules Ansible doivent être exécutés.

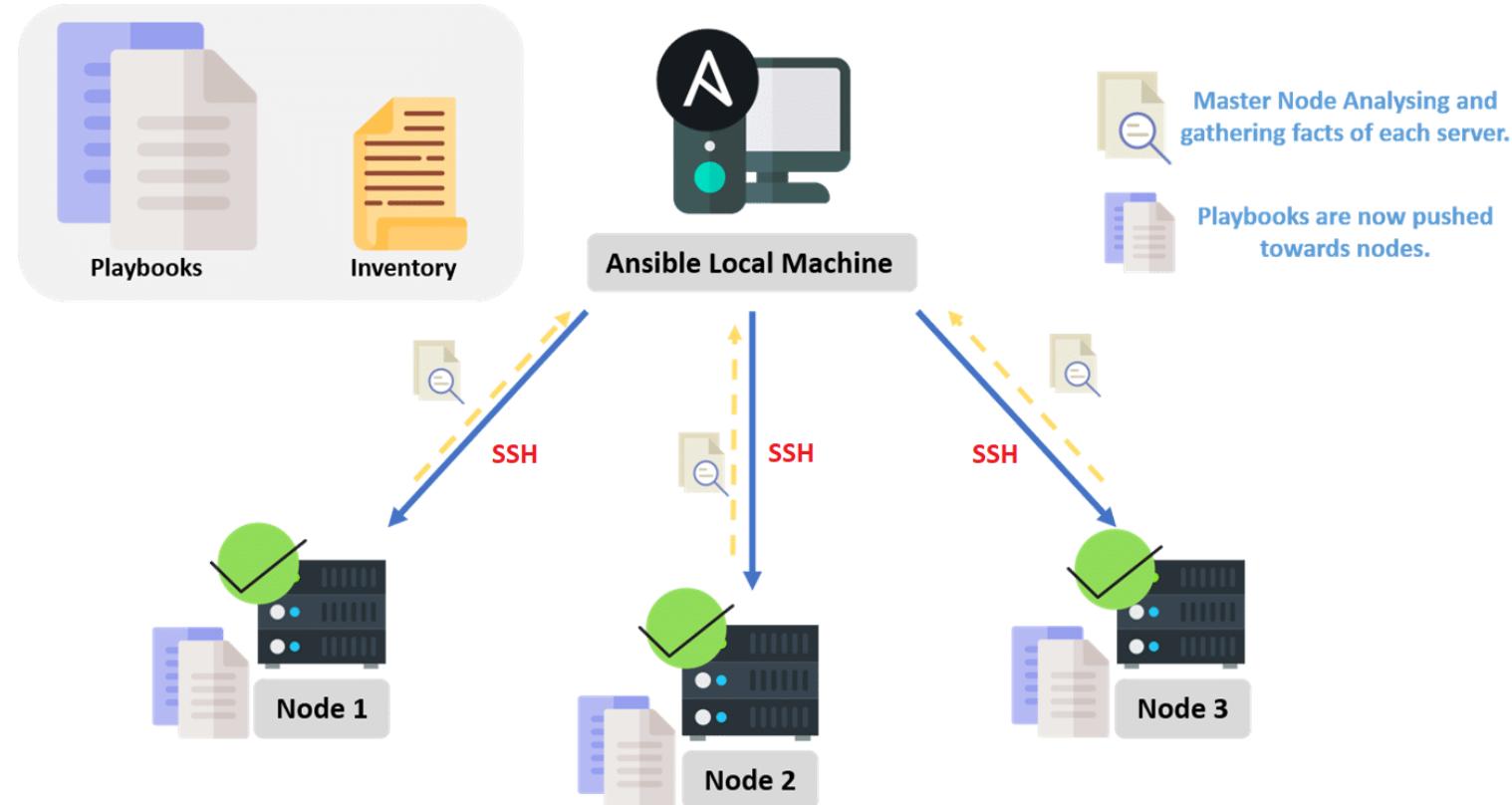


# Comment fonctionne Ansible

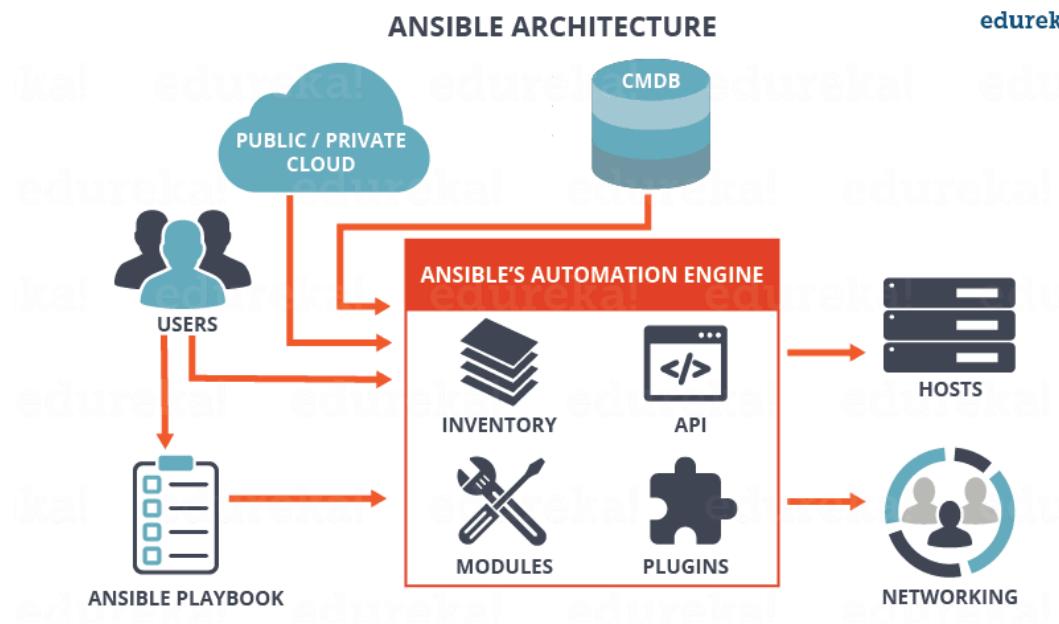
Ansible exécute ces modules en SSH et les supprime une fois terminé.

La seule condition requise pour cette interaction est que votre nœud maître Ansible dispose d'un accès de connexion aux nœuds esclaves.

Les clés SSH sont le moyen le plus courant de fournir un accès, mais d'autres formes d'authentification sont également prises en charge.



# Architecture Ansible



+

•

## Modules

- Ansible se connecte aux nœuds et envoie de petits «modules Ansible». Les modules sont exécutés par Ansible, puis supprimés une fois terminés. Ces modules peuvent résider sur n'importe quelle machine ; aucun serveur, démon ou base de données n'est requis ici.

○

## Plugins

- Un plugin est un morceau de code qui étend les fonctionnalités de base d'Ansible. Il existe de nombreux plugins pratiques, et nous pouvons également écrire nos propres plugins.

## Inventaire

- Nous en avons déjà parlé dans la section Terminologies Ansible. Un inventaire est une liste d'hôtes/nœuds, ayant des adresses IP, des serveurs, des bases de données, etc., qui doivent être gérés.

## Playbook

- Un Playbook contient le code à réaliser. Il est simple et écrit au format YAML et décrit essentiellement les tâches qui sont censées être exécutées via Ansible. Nous pouvons lancer des tâches de manière synchrone ou asynchrone avec des playbooks.

## APIs

- Les API Ansible fonctionnent comme transport pour les services cloud, qu'ils soient publics ou privés.

## CMDB

- CMDB est un type de référentiel qui agit comme un entrepôt de données pour les installations informatiques.

# Modules

- Les modules sont des bits de code transférés au système cible et exécuté pour satisfaire la déclaration de tâche. Ansible est livré avec plusieurs centaines aujourd'hui !

- ✓ apt/yum
- ✓ copy
- ✓ file
- ✓ get\_url
- ✓ git
- ✓ ping
- ✓ debug
- ✓ service
- ✓ synchronize
- ✓ template

# Documentation des Modules

<http://docs.ansible.com>

05/09/2024

The screenshot shows the Ansible Documentation website for version 2.8. The top navigation bar includes links for ANSIBLEFEST, PRODUCTS, COMMUNITY, WEBINARS & TRAINING, and BLOG. A search bar is located at the top left. On the right, there's a message about reading an unmaintained version of the documentation, mentioning security vulnerabilities (CVE). The main content area is titled "Module Index" and lists various module categories: All modules, Cloud modules, Clustering modules, Commands modules, Crypto modules, Database modules, Files modules, Identity modules, Inventory modules, Messaging modules, Monitoring modules, Net Tools modules, Network modules, Notification modules, Packaging modules, Remote Management modules, Source Control modules, Storage modules, System modules, Utilities modules, Web Infrastructure modules, and Windows modules.

A Documentation

Ansible 2.8

Search docs

Installation, Upgrade & Configuration

Installation Guide

Ansible Porting Guides

Using Ansible

User Guide

Ansible Quickstart

Getting Started

Working with Command Line Tools

Introduction To Ad-Hoc Commands

Working with Inventory

Working With Dynamic Inventory

Working With Playbooks

Understanding Privilege Escalation

Ansible Vault

Working with Patterns

Working With Modules

Introduction

Return Values

Module Maintenance & Support

Module Index

All modules

Docs » User Guide » Working With Modules » Module Index

You are reading an unmaintained version of the Ansible documentation. Unmaintained Ansible versions can contain unfixed security vulnerabilities (CVE). Please upgrade to a maintained version. See [the latest Ansible documentation](#).

## Module Index

- All modules
- Cloud modules
- Clustering modules
- Commands modules
- Crypto modules
- Database modules
- Files modules
- Identity modules
- Inventory modules
- Messaging modules
- Monitoring modules
- Net Tools modules
- Network modules
- Notification modules
- Packaging modules
- Remote Management modules
- Source Control modules
- Storage modules
- System modules
- Utilities modules
- Web Infrastructure modules
- Windows modules

Search this site

Elies Jebri

17

# Documentation des Modules

```
# List out all modules installed
$ ansible-doc -l
...
copy
cron
...

# Read documentation for installed module
$ ansible-doc copy
> COPY
```

The [copy] module copies a file on the local box to remote locations. Use the [fetch] module to copy files from remote locations to the local box. If you need variable interpolation in copied files, use the [template] module.

\* note: This module has a corresponding action plugin.

Options (= is mandatory):

---

# Modules: Run Commands

- Si Ansible n'a pas de module qui convient à vos besoins, il y a un module " run commande"
- **command**: prend la commande et l'exécute sur l'hôte. Le plus sûr et prévisible.
- **shell** : s'exécute via un shell comme /bin/sh afin que vous puissiez utiliser des pipes, etc.
- **script** : exécute un script local sur un nœud distant après l'avoir transféré.
- **raw** : exécute une commande sans passer par le module Ansible subsystem.

**REMARQUE** : Contrairement aux modules standard, les commandes d'exécution n'ont aucun concept d'état et ne doit être utilisés qu'en dernier recours.

---

# Inventory

- L'inventaire est une collection d'hôtes (nœuds) avec des données et des regroupements associés qu'Ansible peut utiliser pour se connecter et gérer.
  - Hosts (nodes)
  - Groups
  - Inventory-specific data (variables)
  - Sources Static ou dynamic



# Exemple Static Inventory

```
10.42.0.2
10.42.0.6
10.42.0.7
10.42.0.8
10.42.0.100
host.example.com
```

```
[control]
control ansible_host=10.42.0.2

[web]
node-[1:3] ansible_host=10.42.0.[6:8]

[haproxy]
haproxy ansible_host=10.42.0.100

[all:vars]
ansible_user=vagrant
ansible_ssh_private_key_file=~/vagrant.d/insecure_private_key
```

# Commandes Ad-Hoc

```
# 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"
```

- Une commande ad-hoc est une tâche Ansible unique à effectuer rapidement, mais que vous ne voulez pas sauvegarder pour plus tard.

# Discovered Facts

- Les Facts sont des informations dérivées de l'examen d'un système hôte qui sont stockées en tant que variables pour une utilisation ultérieure dans un play.

```
$ ansible localhost -m setup
localhost | success >> {
  "ansible_facts": {
    "ansible_default_ipv4": {
      "address": "192.168.1.37",
      "alias": "wlan0",
      "gateway": "192.168.1.1",
      "interface": "wlan0",
      "macaddress": "c4:85:08:3b:a9:16",
      "mtu": 1500,
      "netmask": "255.255.255.0",
      "network": "192.168.1.0",
      "type": "ether"
    },
  }
}
```



## Lab # 1: Ad-Hoc Commands

---

# Variables

- Ansible peut travailler avec des métadonnées provenant de diverses sources et gérer leur contexte sous forme de variables.
- Paramètres de ligne de commande
- Plays et tâches
- Des dossiers
- Inventaire
- Faits découverts
- Les rôles



---

# Tasks

- Les tâches sont l'application d'un module pour effectuer une unité de travail spécifique.
  - **file**: A directory should exist
  - **yum**: A package should be installed
  - **service**: A service should be running
  - **template**: Render a configuration file from a template
  - **get\_url**: Fetch an archive file from a URL
  - **git**: Clone a source code repository





# Exemple de Tasks dans un Play

```
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/  
  
- name: restart httpd  
service:  
  name: httpd  
  state: restarted
```

---

# Handler Tasks

- Les Handlers sont des tâches spéciales qui s'exécutent à la fin d'un play si elles sont notifiées par une autre tâche lorsqu'un changement se produit.
  - Si un fichier de configuration est modifié,ifiez une tâche de redémarrage du service pour qu'il doit s'exécuter.



# Exemple de Handler Task dans un Play

```
tasks:  
- name: httpd package is present  
  yum:  
    name: httpd  
    state: latest  
    notify: restart httpd  
  
- name: latest index.html file is present  
  copy:  
    src: files/index.html  
    dest: /var/www/html/  
  
handlers:  
- name: restart httpd  
  service:  
    name: httpd  
    state: restarted
```

# Plays & Playbooks



- Les plays sont des ensembles ordonnés de tâches à exécuter sur des sélections d'hôtes de votre inventaire.
- Un playbook est un fichier contenant un ou plusieurs plays.

# Exemple de Playbook

```
---
- 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/
```

# Dénomination humaine

```
---
- 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/
```

# Host Selector

```
---
- 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/
```

# Escalade de privileges

```
---
- 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/
```

# Variables d'un Play

```
---
```

```
- 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/
```

# Tasks

```
---
```

```
- name: install and start apache
hosts: web
become: yes
vars:
  http_port: 80

tasks:
- name: latest 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/
```



## Lab # 2: Un Simple Playbook

---

# Faire plus avec les Playbooks

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



# Templates

```
<VirtualHost *: {{ http_port }}>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/
    <Directory "/var/www/">
        AllowOverride All
    </Directory>
</VirtualHost>
```



- Ansible embarque un Jinja2 templating engine qui peut être utilisé pour dynamiquement :
  - Définir et modifier les variables de lecture
  - Logique conditionnelle
  - Générer des fichiers tels que des configurations à partir de variables

# Loops

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



- Les boucles peuvent effectuer une 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.

# Conditionnels

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



- Ansible prend en charge l'exécution conditionnelle d'une tâche en fonction de l'évaluation au moment de l'exécution d'une variable, d'un Fact ou du résultat de la tâche précédente.

# Tags

Les tags (balises) sont utilisées pour pouvoir exécuter un sous-ensemble d'un playbook à la demande.

```
- yum:  
  name: "{{ item }}"  
  state: latest  
with_items:  
- httpd  
- mod_wsgi  
tags:  
- packages  
  
- template:  
  src: templates/httpd.conf.j2  
  dest: /etc/httpd/conf/httpd.conf  
tags:  
- configuration
```

# Blocks

```
- block:  
  - yum:  
      name: "{{ item }}"  
      state: latest  
  with_items:  
    - httpd  
    - mod_wsgi  
  
  - template:  
      src: templates/httpd.conf.j2  
      dest: /etc/httpd/conf/httpd.conf  
  when: ansible_os_family == "RedHat"
```

- Les blocs réduisent les directives de tâches répétitives, permettent un regroupement logique des tâches et même la gestion des erreurs de Play.

# Roles

```
roles /  
    apache-server /  
        tasks /  
            main . yml  
        handlers /  
            main . yml  
        files /  
            index . html  
        template /  
            apache . conf . j2
```



- Les rôles sont des packages de contenu Ansible étroitement liés qui peuvent être partagés plus facilement que les Plays seuls.
- Améliorent la lisibilité et la maintenabilité des Plays complexes
- Facilitent le partage, la réutilisation et la standardisation des processus d'automatisation
- Permettent au contenu Ansible d'exister indépendamment des playbooks, des projets et même des organisations
- Fournissent des commodités fonctionnelles telles que la résolution de chemin de fichier et les valeurs par défaut



# Exemple de projet avec rôles intégrés

```
site.yml  
roles/  
  common/  
  files/  
  templates/  
  tasks/  
  handlers/  
  vars/  
  defaults/  
  meta/  
  apache/  
    files/  
    templates/  
    tasks/  
    handlers/  
    vars/  
    defaults/
```

Exemple de projet  
avec rôles intégrés  
*appelés depuis un  
playbook de plus  
haut niveau*

```
# site.yml
---
- hosts: web
  roles:
    - common
    - apache
```

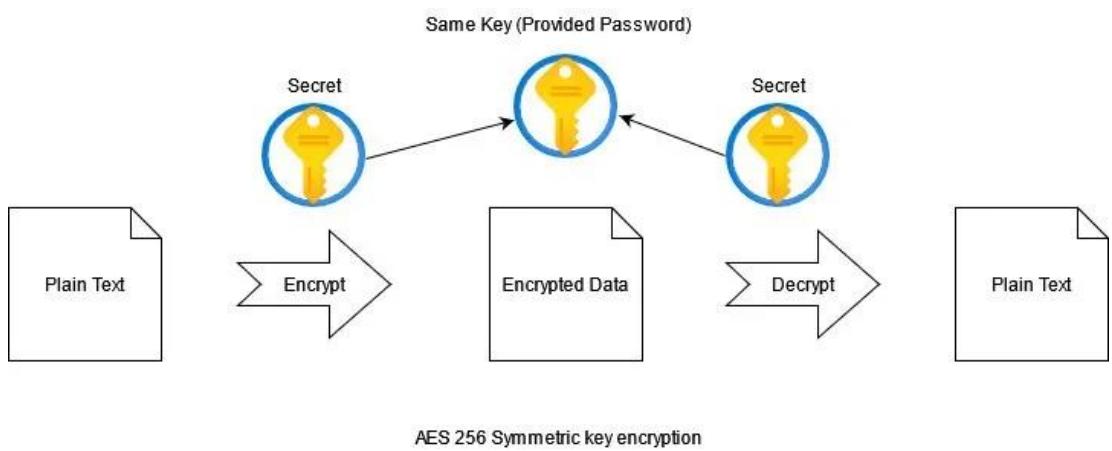


## Les Vaults sur Ansible

# C'est quoi Ansible Vault

- Ansible Vault (coffre-fort en français) est une fonctionnalité d'Ansible qui vous permet de conserver des données sensibles telles que des mots de passe, des clés SSH, de certificats SSL, des jetons d'API et tout ce que vous ne voulez pas que le public voie, plutôt que de les stocker sous un format brut dans des playbooks ou des rôles.
- Comme il est courant de stocker des configurations Ansible dans un contrôleur de version tel que git, nous avons besoin d'un moyen de stocker ces données secrètes en toute sécurité.
- Le Vault est la réponse à cela, puisqu'il permet de chiffrer n'importe quoi à l'intérieur de votre fichier YAML, ces données de Vault peuvent ensuite être distribuées ou placées dans le contrôle de code source.

# Que propose ansible-vault



- Ansible-Vault utilise l'algorithme AES256 pour fournir un cryptage symétrique associé à un mot de passe fourni par l'utilisateur.
- Cela signifie que le même mot de passe est utilisé pour crypter et décrypter le contenu.
- Ansible est capable d'identifier et de décrypter tous les fichiers cryptés dans le coffre-fort qu'il trouve lors de l'exécution d'un playbook ou d'une tâche.
- Il existe d'autres moyens plus avancées pour sécuriser vos données comme Hashicorp Vault. Mais pour des petits projets Ansible-Vault suffira largement.