

Ansible pour DevOps

Ulrich MONJI

Plan

- Intérêt d'Ansible et de la formation
- Présentation du formateur
- Introduction au DevOps et Ansible
- Installation et configuration de Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating, loop et condition
- Include et import, tags
- Sécurité
- Rôle ansible
- Mini-projet

Plan

- Intérêt d'Ansible et de la formation
 - Présentation du formateur
 - Introduction au DevOps et Ansible
 - Installation et configuration de Ansible
 - Commandes AD-HOC
 - Découverte du yaml
 - Inventaire ansible
 - Playbook
 - Templating, loop et condition
 - Include et import, tags
 - Sécurité
 - Rôle ansible
 - Mini-projet

Intérêt d'Ansible et de la formation

- Historiquement, les OPS géraient manuellement les serveurs
- Les devs sont agiles, ils publient fréquemment des releases logicielles
- Avec le développement des Datacenters, les ops ont du mal à suivre cette croissance exponentielle
- Cette difficulté va freiner le travail des devs, et ralentir de façon globale la livraison d'un projet informatique
- Une solution au problème est la naissance de Ansible en 2012
- L'objectif d'ansible est de gagner en productivité sur différentes thématiques d'automatisation
- Le projet est racheté par Redhat en 2015
- Redhat est le leader de l'open source d'entreprise

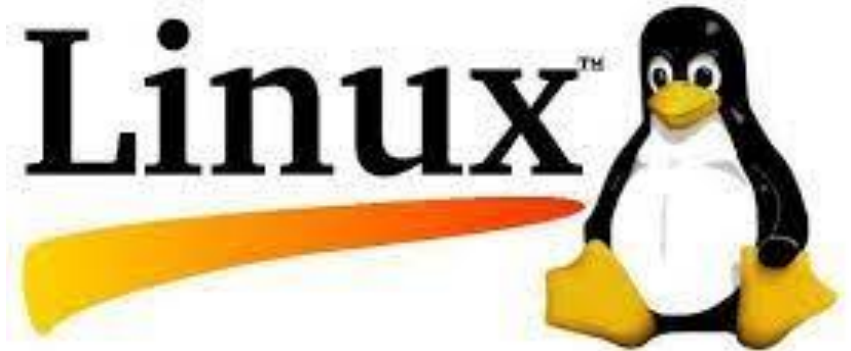


Intérêt d'Ansible et de la formation : Avantages

- Libre et Opensource
- Simple d'utilisation : installation simple, utilisation du yaml
- Puissant : Provisionning, Configuration Management, Orchestration, troubleshooting, déploiement d'application, test, etc ... il offre un large choix de uses cases
- Flexible : via les nombreux modules disponibles
- Agentless : pas de réflexion à se poser sur l'agent



Prérequis du cours



Plan

- Intérêt d'Ansible et de la formation
- **Présentation du formateur**
- Introduction au DevOps et Ansible
- Installation et configuration de Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating, loop et condition
- Include et import, tags
- Sécurité
- Rôle ansible
- Mini-projet

Ulrich MONJI - Ingénieur en Systèmes, Réseaux et Telecommunications - UTT

Atos-Worldline - Ingénieur Système

- Build et Run de plateforme Cloud
- Virtualisation - Stockage - Automatisation
- Comptes clients: Carrefour, Auchan, ARJEL, SAMU

Adneom - Consultant IT

Groupe SII - Consultant IT (Cloud/Devops)

- Consultant chez Orange France
- Migration d'une application monolithique en microservice

Formateur et blogueur chez eazytraining



Expérience Ansible

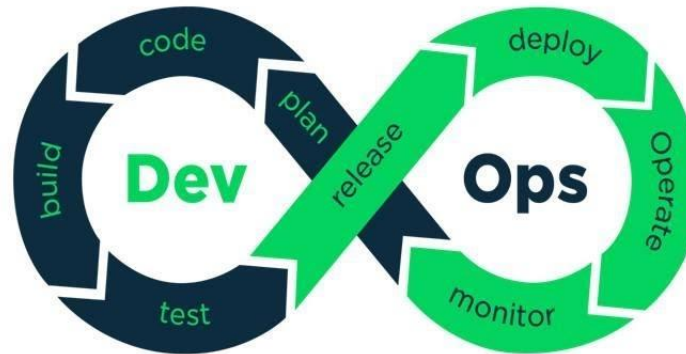
- Build et Run des VMs
 - Le process de Build est manuel, et plusieurs outils internes sont installés sur les VMs
 - Une automatisation était nécessaire pour gagner du temps
 - Ansible a été implémenté
- Administration du projet Arjel : Agence de Régulation des Jeux en Ligne
 - Le parc Arjel était piloté manuellement par l'admin
 - On avait vingtaine de machines par environnement
 - Pour des besoins de sécu (Norme PCI), l'outil puppet ne pouvait être utilisé sur Arjel
 - Les patchs se faisaient 1 fois / mois par environnement, et on en avait 3
 - 2 Jours par environnement, soit 6 jours dans le mois dédiés au patch Arjel
 - Ansible était la solution pour automatiser les actions
 - On passe d'une journée complète de travail à 1h, soit 6h par mois avec des playbooks ansible
- Migration réseau du Projet Carrefour
 - Staging et recette/qualif sur les mêmes réseaux
 - Il était question de migrer la recette/qualif sur un autre réseau
 - Les Vlan sont mis à dispo par les équipes réseau, il fallait reconfigurer toute la couche Network des VMs
 - Ansible a été une solution

Plan

- Intérêt d'Ansible et de la formation
- Présentation du formateur
- **Introduction au DevOps et Ansible**
- Installation et configuration de Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating, loop et condition
- Include et import, tags
- Sécurité
- Rôle ansible
- Mini-projet

Introduction au DevOps et Ansible (1/3): DevOps

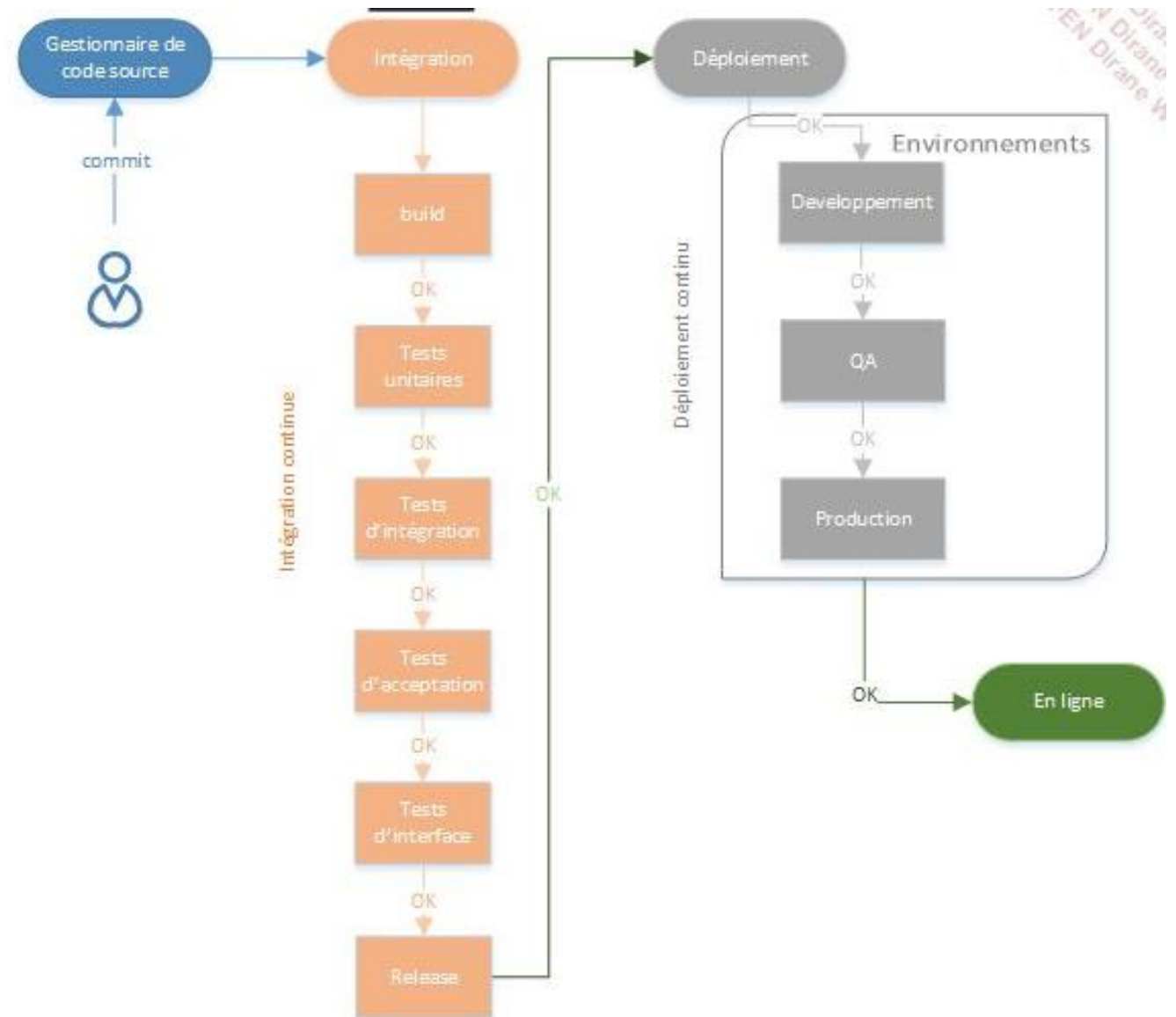
Agile vs. DevOps



- Agile: méthode de développement
- DevOps: agilité dans le Dev et l'Ops = CI + CD

Introduction au DevOps et Ansible (2/3): CI/CD

- Intégration en continu
- Test en continu
- Déploiement en continu
- Automatisation 😊
- Docker



Introduction au DevOps et Ansible (3/3): Ansible

- C'est quoi ?
 - ✓ Cloud provisionneur
 - ✓ Configuration Management
 - ✓ Déploiement d'Application
- Pourquoi ?
 - ✓ Efficace: pas d'agent, copie de petit bout de code sur les machines distantes
 - ✓ Sécurisé: pas d'agent, utilise OpenSSH
 - ✓ Contribution: open-source, python

Les principaux composants d'ansible

L'exécutable

Les modules

L'inventaire

Les playbooks

Composants d'ansible : exécutable

- La commande ansible est disponible dans le terminal dès lors que nous avons installé ansible
- Ideal pour démarrer un projet et tester nos configurations ansible
- S'utilise facilement avec les modules
- Permet de piloter facilement une infrastructure cible

Composants d'ansible : les modules

- Ils font toute la puissance de ansible
- C'est le code python permettant de réaliser une action
- Ils sont disponibles sur la doc officielle et sur github:
 - https://docs.ansible.com/ansible/latest/collections/index_module.html
 - <https://github.com/ansible/ansible-modules-core/>
- On distingue les modules core et ceux customs
- Ils sont regroupés par catégorie
- Vous pouvez écrire vos propres modules
- On en distingue pour différents contextes :
 - Cloud computing
 - Virtualisation
 - Provisionning
 - Configuration Management
 - Networking
 - Conteurisation etc ...



Les modules

Setup module : https://docs.ansible.com/ansible/latest/collections/ansible/builtin/setup_module.html

- Appeler automatiquement par le playbook pour recueillir les facts

File module : https://docs.ansible.com/ansible/latest/collections/ansible/builtin/file_module.html

- Permet de manager les fichiers sur la cible

Copy module : https://docs.ansible.com/ansible/latest/collections/ansible/builtin/copy_module.html

- Permet de copier des fichiers depuis le serveur ansible sur la machine cible, ou depuis un serveur distant vers un autre serveur distant
- Le module fetch permet de faire l'inverse (target -> serveur ansible)
- Pour windows, le module win_copy existe

Command module : https://docs.ansible.com/ansible/latest/collections/ansible/builtin/command_module.html

- Permet de lancer des commandes sur le système distant, mais pas dans un shell, ie que \$HOME, |, <, >, & etc ne sont pas valides, pour cela il faudrait utiliser le module shell
- Pour les machines windows, il existe le module win_command

Codes couleurs

- Rouge : Echec
- Jaune : Succès avec des changements
- Vert : Succès sans aucun changement

la documentation des modules est aussi disponible en ligne de commande avec la commande ansible-doc

- **ansible-doc <module>**

Idempotence :

- Une opération a le même effet qu'on l'applique une ou plusieurs fois.
- C'est un des principes d'Ansible via ses modules, l'idée est de définir un état souhaité, si l'état est déjà satisfait alors rien n'est fait



L'inventaire ansible

- C'est l'inventaire du parc de Machines à gérer
- Cela peut être des serveurs (virtuel ou physiques), des switches, des routeurs, des containers, des baies de stockage, etc ...
- On pourrait rajouter des informations supplémentaires caractérisant les machines
- On distingue les inventaires statiques et dynamiques

TP-0: Déploiement du lab de travail

- Le Lab sera constitué essentiellement de VMs installées sous virtualbox
 - Cela sous entend qu'on devra installer virtualbox.
- Pour faciliter le déploiement sous virtualbox, nous utilisons l'outil vagrant pour faire de Infra as Code (IaC)
 - Cela sous entend l'installation de vagrant
- Comme VMs, nous allons avoir le serveur ansible et une machine cible à piloter
- Les instructions détaillées de ce TP vous sont fournies par le formateur

Plan

- Intérêt d'Ansible et de la formation
- Présentation du formateur
- Introduction au DevOps et Ansible
- **Installation et configuration de Ansible**
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating, loop et condition
- Include et import, tags
- Sécurité
- Rôle ansible
- Mini-projet



Ansible: Installation



Méthodes 1 : Le gestionnaire de package du système

- On récupère la version disponible sur les repos configurés
- Elle peut ne pas être la dernière version

Méthodes 2 : L'utilitaire pip de python

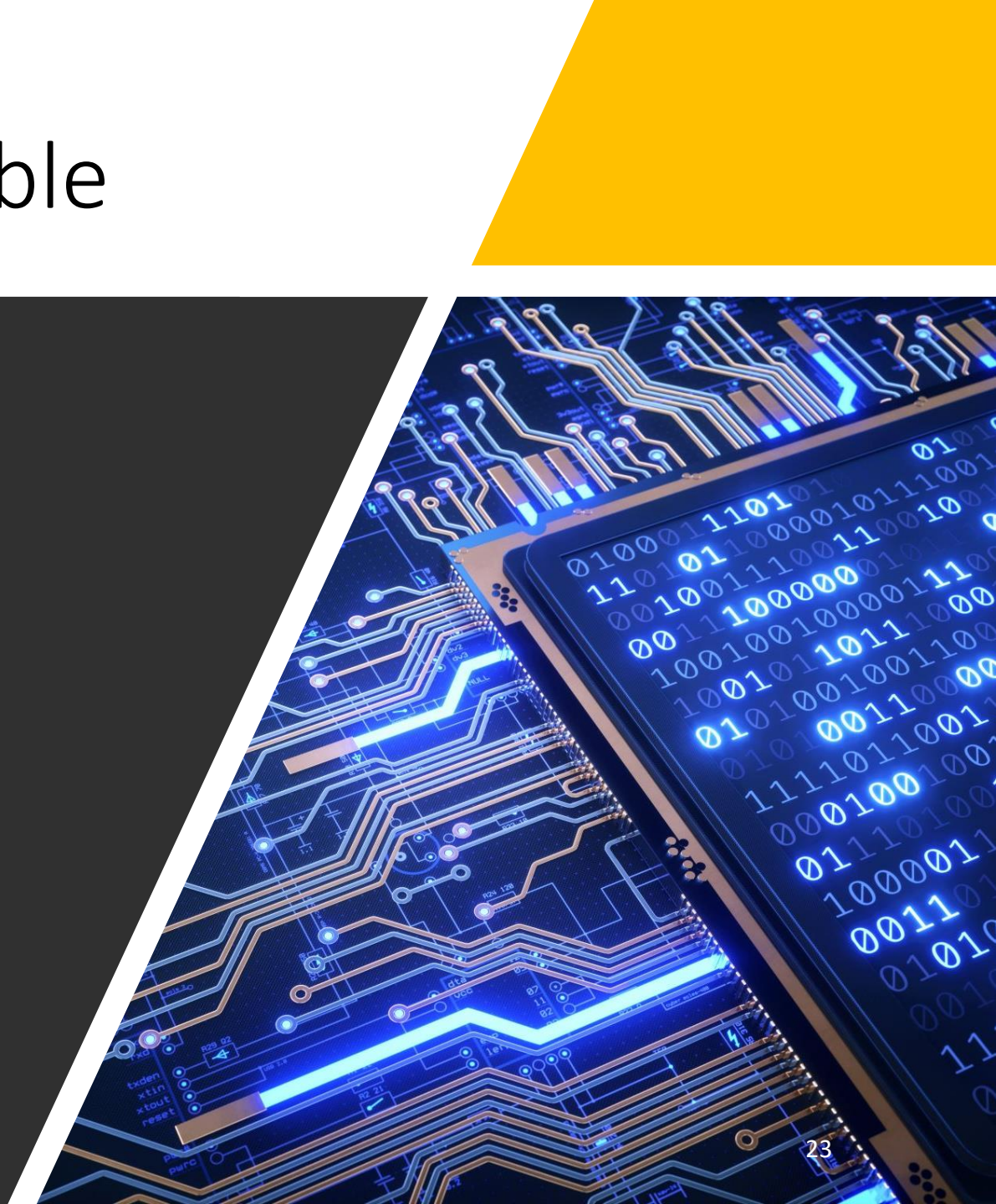
- Ansible a besoin de python pour travailler
- Par défaut python est sur les systèmes linux, d'où les gens pensent qu'on n'a pas besoin de dépendances
- Via la librairie pip de python, on peut aussi installer ansible
- L'avantage est qu'on disposera de la version la plus à jour possible de Ansible

Ansible: fichier de configuration ansible.cfg

- Par ordre croissant de priorité, on a :
 - **/etc/ansible/ansible.cfg** : fournit (ou pas) par le package après l'installation
 - **~/.ansible.cfg** : dans votre home, créé par vos soins
 - **./ansible.cfg** : dans votre répertoire de travail
 - La variable d'env **ANSIBLE_CONFIG**
 - `export ANSIBLE_CONFIG=un_fichier_sur_le_disque.cfg`
- La commande **ansible --version** permet de savoir celui qui est pris en compte en temps réel

TP-1: Installation de Ansible

- Installation sur CentOS
- Toujours travaillez avec vagrant et pas root pour des raisons de sécurité
- On va installer via l'outil pip3 de python 3
- Test de la version installée
- Test du fichier de configuration chargé



Plan

- Intérêt d'Ansible et de la formation
- Présentation du formateur
- Introduction au DevOps et Ansible
- Installation et configuration de Ansible
- **Commandes AD-HOC**
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating, loop et condition
- Include et import, tags
- Sécurité
- Rôle ansible
- Mini-projet

Commandes AD-HOC : Principes

ANSIBLE AD HOC COMMANDS - SYNTAX			
	Host Group	Module	Arguments to the module
ansible	webserver	-m yum	-a "name=httpd state=latest"
ansible	allservers	-m shell	-a " find /opt/oracle -type f -mtime +10 -name '*.log' "
ansible	appserver	-m user	-a "name=saravak group=admins append=yes shell=bin/bash"

- Test de module
- Lancement de tâche rapidement
- Peut être lancé sur un groupe de machine

www.middlewareinventory.com

TP-2: Utilisation des commandes ad-hoc

- Créez un fichier d'inventaire hosts
- Utilisez une commande ad-hoc pour tentez de ping le client ansible
- Utilisez une commande ad-hoc pour créer un fichier toto.txt sur le client
- Vérifiez que le fichier a bien été créé avec le contenu
- Rajoutez un client et modifier le fichier inventaire afin de rajouter le nouveau client
- Relancez l'action de ping et de création de fichier sur les 2 clients maintenant et vérifiez le résultat
- Testez l'effet du module « setup » sur votre inventaire
- Comment connaitre sa RAM via Ansible ?

Plan

- Intérêt d'Ansible et de la formation
- Présentation du formateur
- Introduction au DevOps et Ansible
- Installation et configuration de Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating, loop et condition
- Include et import, tags
- Sécurité
- Rôle ansible
- Mini-projet

YAML

- Les playbooks se font en YAML ou en JSON
- Le YAML est le plus utilisé
- C'est facile à utiliser, et idéal pour la collaboration
- LE YAML est supporté sur plusieurs langages de programmation
- Généralement, on rajoute les extension .yaml ou .yml sur les fichiers

```
# Un fichier yaml démarre par les 3 tirets ci-dessus
# Déclaration simple
chaine_simple: "Une chaîne simple"
# Ici _42 va contenir un entier :
_42: 42
# _33 va contenir un chiffre à virgule :
_33: 33.333
```

Découverte du yaml (1/5): Déclaration de variables

Version compacte

a: [1, 2, "trois"]

Version très compacte

a: [1,2,"trois"]

Version invalide (manque l'espace après

a:[1,2,"trois"]

a:

- 1

- 2

- "trois"

Découverte du yaml (2/5): les tableaux

```
utilisateur1:  
  nom: pierre  
  prenom: yannig
```

```
utilisateur2:  
  nom: pierre  
  prenom: sarah
```

```
utilisateur1:  
  nom: pierre  
  prenom: yannig  
  date_de_naissance:  
    jour: 7  
    mois: 11  
    annee: 1977
```

Découverte du yaml (3/5): Structures clé/valeur

Découverte du yaml (4/5): Tableau de tables de hachage

```
# Version un peu plus compacte
users:
  - { nom: perre, prenom: yannig }
  - { nom: perre, prenom: sarah }
# Version plus compacte, mais moins lisible
users: [{nom: perre, prenom: yannig},{nom: perre, prenom: sarah}]
```

```
liste_utilisateurs:
  - nom: perre
    prenom: yannig
  - nom: perre
    prenom: sarah
```


Découverte du yaml (5/5): Inventaire

```
all:  
  hosts:  
    rec-apache-1:  
      ansible_user: root
```

- Depuis la version 2.4 de Ansible
- All et Hosts

```
rec-apache-1 ansible_user=root
```

TP-3: Inventaire au format yaml

- Modifiez le fichier hosts que vous avez écrits au format INI afin qu'il soit en au format yaml
- Testez à nouveau vos commandes ad-hoc avec le nouveau fichier d'inventaire au format yaml
- Le formateur pourra éventuellement vous donner plus d'instructions.

Plan

- Intérêt d'Ansible et de la formation
- Présentation du formateur
- Introduction au DevOps et Ansible
- Installation et configuration de Ansible
- Commandes AD-HOC
- Découverte du yaml
- **Inventaire ansible**
- Playbook
- Templating, loop et condition
- Include et import, tags
- Sécurité
- Rôle ansible
- Mini-projet

Inventaire ansible : Problématique

[webservers]

192.168.35.140

192.168.35.141

192.168.35.142

192.168.35.143

[appservers]

192.168.100.1

192.168.100.2

192.168.100.3

[dbservers]

172.35.0.5

Inventaire ansible : Fonctionnalités

- 3 Formats possibles : INI, YAML, JSON
- Regrouper les machines par groupes
- Définir des groupes enfants d'un groupe
- Fournir les moyens de connexion à un hôte spécifique
- Fournir les moyens de connexion à un groupe d'hôte
- Les `host_vars` : inventaire d'un hôte
- Les `group_vars` : inventaire d'un groupe de machines

```
[all:vars]
ansible_connection=local

[apache]
rec-apache-1 apache_url=rec.wiki.localdomain

[mysql]
rec-mysql-1 mysql_user_password=MyPassword!

[active-directory]
active-directory-1

[microservices]
container-1 ansible_connection=docker

[linux:children]
apache
mysql

[windows:children]
active-directory

[container:children]
microservices

[windows:vars]
ansible_connection=winrm

[container:vars]
ansible_connection=localhost
```

Inventaire ansible : Structure INI

```

all :
  vars:
    ansible_connection: local

linux:
  children:
    apache:
      hosts:
        rec-apache-1:
          apache_url: "rec.wiki.localdomain"

    mysql:
      hosts:
        rec-mysql-1:
          mysql_user_password: "MyPassWord!"

windows:
  children:
    active-directory:
      hosts:
        active-directory-1: {}

  vars:
    ansible_connection: "winrm"

container:
  children :
    microservices :
      hosts :
        container-1 :
          ansible_connection : "docker"
  vars :
    ansible_connection : "localhost"

```

Inventaire ansible : Structure YAML

Inventaire ansible : Variable d'inventaire

```
[apache]
```

```
rec-apache-1 apache_url=rec.wiki.localdomain
```

Cette déclaration prendra la forme suivante au format YAML :

```
apache:
  hosts:
    rec-apache-1:
      apache_url: "rec.wiki.localdomain"
```

```
[mysql:vars]
```

```
mysql_user_password=MyPassWord!
```

Et la même déclaration au format YAML :

```
mysql:
  vars:
    mysql_user_password: "MyPassWord!"
```



```
[vagrant@ansible tmp]$ cat fichier_de_variables.yaml
---
liste:
  - item1
  - item2

liste_2:
  [ item3, item4 ]
...
```

```
vars_files:
  - fichier_de_variables.yaml
```

```
vars_prompt:
  - name: ma_variable
```

Variables et mots clés

- **all** : C'est l'ancêtre de tous les groupes
- **hosts** :
 - Dans un inventaire, sert à définir les machines d'un group
 - Dans un playbook, définit les cible où appliquer les actions
- **children** : Pour définir les sous groupes d'un groupe
- **vars** : Dans le playbook ou l'inventaire, pour définir les variables d'un groupe
- **vars_files** : Dans le playbook, pour indiquer un fichier de variables
- **vars_prompt** : Pour demander à l'utilisateur de saisir la valeur de la variable en début de playbook

PS : On parlera des playbook plus tard

Variables spéciales, facts

- Elles sont disponibles lors de l'exécution du playbook
- Le module setup (gathering_fact dans le playbook) vous permet aussi de les avoir
- La doc officielle:
https://docs.ansible.com/ansible/latest/reference_appendices/special_variables.html

```
hosts: client1
gather_facts: False

vars:
  hash:
    hash_key: valeur du hash

tasks:
  - name: Test de la variable hash
    debug:
      msg: "{{ hash }}"

  - name: Test de la valeur de la variable hash.key
    debug:
      msg: "{{ hash.hash_key }}"
```

Variable de
type
dictionnaire

```
hosts: client1
gather_facts: False

vars:
  liste:
    - element1
    - element2
  liste_2:
    [ element4, element5 ]

tasks:
  - name: Test de la liste
    debug:
      msg: "{{ liste }}"

  - name: Test du premier element
    debug:
      msg: "{{ liste.0 }}"
```

Variable de
type liste
(tableau)

- variables du groupe dans le fichier **host** ;
- variables du groupe dans les fichiers **group_vars** ;
- variables de la machine au niveau du fichier **host** ;
- variables de la machine au niveau du fichier **host_vars** ;
- variables se trouvant dans un fichier YAML (`-e @fichier.yml`) ;
- variables directement passées à Ansible (`-e variable=valeur`).

Inventaire ansible : Application des variables

Inventaire ansible : ansible-inventory

- La commande ansible-inventory permet de piloter l'inventaire

```
[vagrant@ansible TP5]$ ansible-inventory --host client1
[DEPRECATION WARNING]: Ansible will require Python 3.8 or r
Current version: 3.6.8 (default, Nov 16 2020, 16:55:22) [GC
be removed from ansible-core in version 2.12. Deprecation w
deprecation_warnings=False in ansible.cfg.
{
  "ansible_host": "192.168.99.11",
  "ansible_ssh_pass": "vagrant",
  "ansible_user": "vagrant"
}
[vagrant@ansible TP5]$
```

```
[vagrant@ansible TP5]$ ansible-inventory --graph
[DEPRECATION WARNING]: Ansible will require Python 3.8 or r
Current version: 3.6.8 (default, Nov 16 2020, 16:
be removed from ansible-core in version 2.12. Dep
deprecation_warnings=False in ansible.cfg.
@all:
  |--@prod:
  |   |--client1
  |--@ungrouped:
[vagrant@ansible TP5]$
```

TP-4: Inventaire et variable

- Créez un fichier hosts.ini au format INI avec les modalités d'inventaire suivant
 - Tous les hôtes via le groupe « all » devront avoir pour login user vagrant
 - Le client devra faire partie d'un groupe appeler « prod »
 - Un groupe contenant l'hôte ansible sera présent
 - Le mot de passe à utiliser pour toutes connexion ssh devra être vagrant pour toutes les machines du groupe « prod »
 - La variable « env » devra être égale à « production » pour toutes les machines du groupe « prod »
- Créez ensuite un fichier hosts.yaml, version yaml du fichier ini
- Transformer le fichier yaml en json et tester de nouveau les commandes
- Testez les commandes ad-hoc de ping et setup avec les trois fichiers d'inventaire

Plan

- Intérêt d'Ansible et de la formation
- Présentation du formateur
- Introduction au DevOps et Ansible
- Installation et configuration de Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- **Playbook**
- Templating, loop et condition
- Include et import, tags
- Sécurité
- Rôle ansible
- Mini-projet

Playbook : principales sections

```
# YAML documents begin with the document separator ----  
  
# The minus in YAML this indicates a list item. The playbook contains a list  
# of plays, with each play being a dictionary  
-  
  
  # Hosts: where our play will run and options it will run with  
  
  # Vars: variables that will apply to the play, on all target systems  
  
  # Tasks: the list of tasks that will be executed within the play, this section  
  #         can also be used for pre and post tasks  
  
  # Handlers: the list of handlers that are executed as a notify key from a task  
  
  # Roles: list of roles to be imported into the play  
  
# Three dots indicate the end of a YAML document  
...
```

- On utilise la commande **ansible-playbook** pour lancer les playbook

Playbooks

- Options du host :
 - **become** : autoriser l'escalade de privilège
 - **Connection** : Pour changer le mode de connexion
 - **gather_facts** : module setup, exécuté par défaut au début du playbook
 - **user** : User utilisé
- Variables : surcharge possible lors du lancement avec l'option -e
- Mots clés :
https://docs.ansible.com/ansible/latest/reference_appendices/playbooks_keywords.html

Playbooks : Pattern

https://docs.ansible.com/ansible/latest/user_guide/intro_patterns.html

Description	Pattern(s)	Targets
All hosts	all (or *)	
One host	host1	
Multiple hosts	host1:host2 (or host1,host2)	
One group	webserver	
Multiple groups	webserver:dbserver	all hosts in webserver plus all hosts in dbserver
Excluding groups	webserver:!atlanta	all hosts in webserver except those in atlanta
Intersection of groups	webserver:&staging	any hosts in webserver that are also in staging

Playbook : handlers et notify

```
---  
- name: "first deployment"  
  become: yes  
  hosts: prod  
  tasks:  
    - name: "Clean Folder"  
      file:  
        path: /usr/share/nginx/html  
        state: absent  
    - name: "Create Folder"  
      file:  
        path: /usr/share/nginx/html/  
        state: directory  
    - name: "Copy index.html"  
      copy:  
        content: "Bonjour"  
        dest: /usr/share/nginx/html/index.html  
        notify: index.html changed  
  handlers:  
    - name: index.html changed  
      service:  
        name: httpd  
        state: restarted  
...
```

Playbook : Structure

```
- name: "Apache Installation"
  hosts: all
  tasks:
    - name: "Install apache package"
      yum:
        name: "httpd"
        state: "present"
    - name: "Start apache service"
      service:
        name: "httpd"
        state: "started"
        enabled: yes
    - name: "Allow http connections"
      firewallld:
        service: "http"
        permanent: yes
        state: "enabled"
    - name: "Copy test.html"
      copy:
        src: "test.html"
        dest: "/var/www/html"
        owner: "apache"
        group: "apache"
```

Playbook :

Variables

- Surcharge des host et group vars
- Variables à partir d'un fichier

```
- name: "Generate html file for each host"
hosts: all
gather_facts: yes
vars:
  host_inventory: "central-inventory"
  inventory_path: "/var/www/html/inventory"
tasks:
  - name: "Create template directory"
    file:
      path: "{{inventory_path}}"
      owner: "apache"
      group: "apache"
      mode: "0755"
      state: "directory"
    delegate_to: "{{host_inventory}}"
  - name: "html file generation"
    template:
      src: "machine.html.j2"
      dest: "{{inventory_path}}/{{inventory_hostname}}.html"
    delegate_to: "{{host_inventory}}"
```

Playbook : Simple Project Ansible

```
production          # inventory file for production server
staging              # inventory file for staging environment

group_vars/
  group1.yml         # here we assign variables to particular groups
  group2.yml
host_vars/
  hostname1.yml      # here we assign variables to particular hosts
  hostname2.yml

library/             # if any custom modules, put them here
module_utils/        # if any custom module_utils to support modules
filter_plugins/      # if any custom filter plugins, put them here

site.yml             # master playbook
webservers.yml       # playbook for webserver tier
dbservers.yml        # playbook for dbserver tier
```

- Inventaire par environnement
- Utilisation des `group_vars` et `host_vars`
- Un playbook pour chaque grande action à faire

TP-5: Déployez un serveur web

- Créez un cluster (1 ansible et 1 client)
- Créez un dossier webapp qui va contenir tous les fichiers de notre projet
- Créez un fichier d'inventaire appelé prod.yml (au format yaml) contenant un groupe prod avec comme seul membre notre client
- Créez un dossier group_vars qui va contenir un fichier nommé prod qui contiendra les informations de connexion à utiliser par ansible
- Créez un playbook nommé nginx.yml permettant de déployez nginx
- Vous avez le droit d'installer tout prérequis que vous jugerez nécessaire à l'aide du module yum
- Vérifiez la syntaxe de votre playbook avec la commande ansible-lint (installez là si elle n'est pas disponible)
- Explorez les options de debug de ansible

Plan

- Intérêt d'Ansible et de la formation
- Présentation du formateur
- Introduction au DevOps et Ansible
- Installation et configuration de Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- **Templating, loop et condition**
- Include et import, tags
- Sécurité
- Rôle ansible
- Mini-projet

Templating avec Jinja

- C'est un moteur de Template pour le langage python
- Il permet de faire des modèles de fichier
- On peut mettre des bloc conditionnels (if, elif, else)
- On peut faire des boucles
- Bien d'autres features disponibles
- La documentation : <https://jinja.palletsprojects.com/en/3.0.x/>
- Filtres ansible :
https://docs.ansible.com/ansible/latest/user_guide/playbooks_filters.html
- Le module **template** sera utilisé
 - https://docs.ansible.com/ansible/latest/collections/ansible/builtin/template_module.html



```
tasks:
- name: Ansible Jinja2 if
  debug:
    msg: >
      --== Ansible Jinja2 condition if ==--

      {# Si le hostname est clien1, alors inclus le message -#}
      {% if ansible_hostname == "client1" -%}
          Ceci est la machine client1
      {% endif %}
```

Jinja2 : If

Jinja2 : if / else

```
tasks:
- name: Ansible Jinja2 if
  debug:
    msg: >
      --== Ansible Jinja2 condition if ==--

      {# Si le hostname est client1, alors inclus le message -#}
      {% if ansible_hostname == "client1" -%}
        Ceci est la machine client1
      {% elif ansible_hostname == "client2" -%}
        Ceci est plutot la cible client2
      {% endif %}
```

```
e: Ansible Jinja2 if
ug:
sg: >
    --== Ansible Jinja2 condition if ==--

    {# Si le hostname est client1, alors inclus le m
    {% if ansible_hostname == "client1" -%}
        Ceci est la machine client1
    {% elif ansible_hostname == "client2" -%}
        Ceci est plutot la cible client2
    {% else -%}
        Aucun de client1 et client2 ici, c'est pl
    {% endif %}
```

Jinja2 :
if/elif/else

```
sk:
- name: Ansible Jinja2 if
  debug:
    msg: >
      --== Ansible Jinja2 condition if
      {% if ma_variable is defined -%}
        ma_variable est définie
      {% else -%}
        ma_variable est non définie
      {% endif %}
```

Jinja2 : is
defined

Jinja2 : Boucle for

```
tasks:
- name: Ansible Jinja2 boucle for range
  debug:
    msg: >
      --== Ansible Jinja2 boucle for range

      {% for compteur in range(1, 5) -%}
        {{ compteur }}
      {% endfor %}
```

```
tasks:
- name: Ansible Jinja2 boucle for
  debug:
    msg: >
      ---== Ansible Jinja2 boucle for ---

      {% for compteur in ansible_interfaces -%}
        Interface compteur {{ loop.index }} = {{ compteur }}
      {% endfor %}
```

Exemple de fichier Template

```
<html>
  <head>
    <title>Machine {{inventory_hostname}}</title>
  </head>
  <body>
    <p>Cette machine s'appelle {{inventory_hostname}}</p>
  </body>
</html>
```

```
- name: "Generate html file for each host"
  hosts: all
  connection: local
  tasks:
    - name: "html file generation"
      template:
        src: "machine.html.j2"
        dest: "{{playbook_dir}}/{{inventory_hostname}}.html"
```



```
---  
- hosts: ubuntu_webserver  
  become: yes  
  tasks:  
    - name: Create new users  
      user:  
        name: '{{ item }}'  
        state: present  
  
    loop:  
      - john  
      - mike  
      - andrew
```

Loop

Clause with_items

```
tasks:
- name: Test de copie de fichier
  copy:
    content: "fichier copié sur le serveur {{ item }} \n"
    dest: /etc/toto
  notify: file copied
  with_items:
    - client1
    - client2
  when: ansible_distribution == item

handlers:
- name: file copied
  debug:
    msg: Le fichier a été copié
```

```
tasks:
- name: Test de copie de fichier
  copy:
    content: "fichier copié sur le serveur {{ item }} \n"
    dest: /etc/toto
  notify: file copied
  with_items: [ 'client1', 'client2' ]
  when: ansible_distribution == item

handlers:
- name: file copied
  debug:
    msg: Le fichier a été copié
```

Clause with_dict:

```
tasks:
  - name: Creating new users
    user:
      name: "{{ item.key }}"
      comment: "{{ item.value.nom }}"
    with_dict:
      lil:
        nom: Lil wayne
      sara:
        nom: Sara godard
```

La documentation officielle sur les boucles dans ansible :

https://docs.ansible.com/ansible/latest/user_guide/playbooks_loops.html

```
---  
  
- hosts: group1  
  tasks:  
    - name: Enable Selinux  
      selinux:  
        state: enabled  
1  when: ansible_os_family == 'Debian'  
    register: enable_selinux 2  
  
    - debug:  
      msg: "Selinux Enabled. Please restart the server to apply changes."  
      when: enable_selinux.changed == true 3  
  
- hosts: group2  
  tasks:  
    - name: Install apache  
      yum:  
        name: httpd  
        state: present  
4  when: ansible_system_vendor == 'HP' and ansible_os_family == 'RedHat'
```

Condition

Plan

- Intérêt d'Ansible et de la formation
- Présentation du formateur
- Introduction au DevOps et Ansible
- Installation et configuration de Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating, loop et condition
- Include et import, tags
- Sécurité
- Rôle ansible
- Mini-projet

Include et Import

- **Include_tasks** (dynamique)
- **Import_tasks** (statique)
- **Import_playbook** (statique)
- Statique : appliqué au chargement du playbook
- Dynamique : appliqué durant l'exécution du playbook
- https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_includes.html

Tags

- https://docs.ansible.com/ansible/latest/user_guide/playbooks_tags.html
- Applicable au niveau d'une task ou de tout un play en entier
- always et never sont des tags speciaux
- `ansible-playbook my_playbooks.yaml --tags filesharing`
- `ansible-playbook my_playbooks.yaml --skip-tags filesharing`

```
- name: Enable and run ntpd
  ansible.builtin.service:
    name: ntpd
    state: started
    enabled: yes
  tags: ntp

- name: Install NFS utils
  ansible.builtin.yum:
    name:
      - nfs-utils
      - nfs-util-lib
    state: present
  tags: filesharing
```

TP-6: Déployer un conteneur docker

- Il est question pour ce TP de voir comment deployer une image docker via Ansible from scratch
- On l'appliquera à l'image de mario
<https://hub.docker.com/r/pengbai/docker-supermario>
- On va créer un nouveau répertoire de travail, nommé TP_mario dans lequel On reprendra l'inventaire du TP précédent.
- Il faut créer trois playbook :
 - docker.yml permettant d'installer docker
 - mario.yml permettant de déployer le conteneur Mario
 - deploy_mario.yaml qui appelle les deux autres



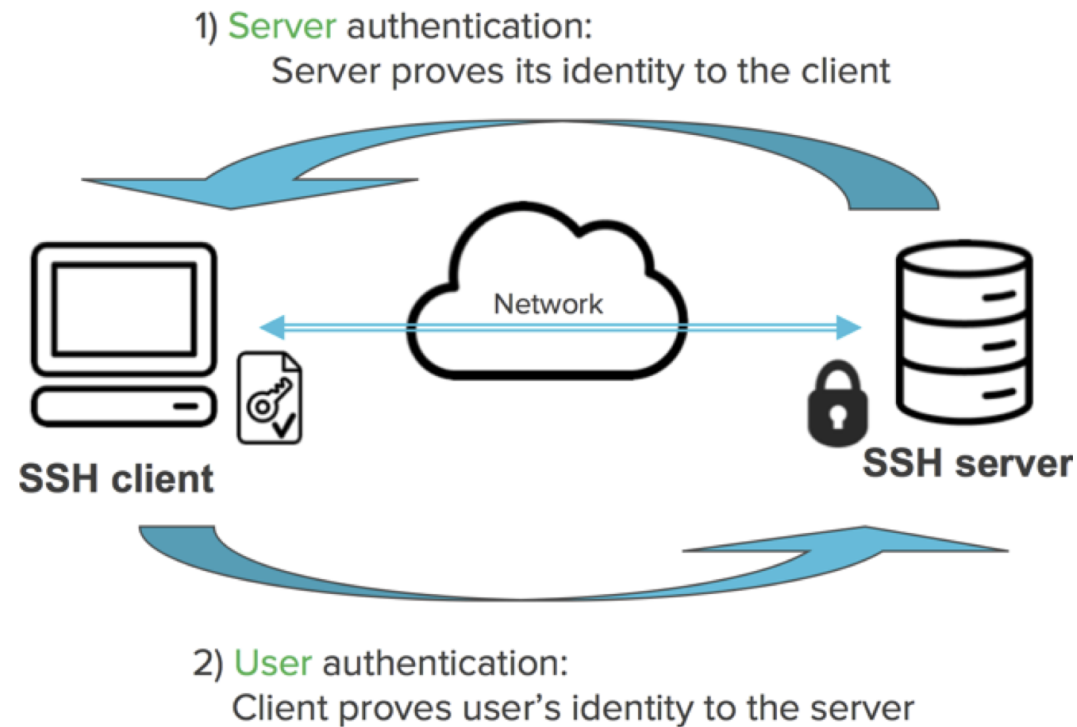
TP-7: Conteneurisation du site web

Conteneurisez l'application web du TP 5

Plan

- Intérêt d'Ansible et de la formation
- Présentation du formateur
- Introduction au DevOps et Ansible
- Installation et configuration de Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating, loop et condition
- Include et import, tags
- Sécurité
- Rôle ansible
- Mini-projet

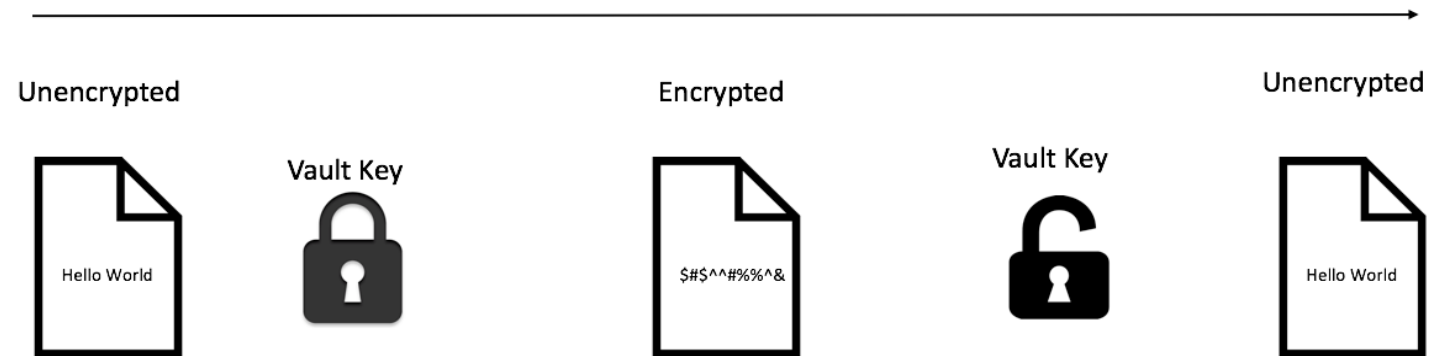
Sécurité (1/2): Paire de clé



- ssh-keygen
- authorized_keys
- ansible_ssh_common_args='-o StrictHostKeyChecking=no'

Sécurité (2/2): Vault

AES Symmetrical Key Encryption



```
ubuntu@ip-10-0-2-54:~$ cat secrets.txt
$ANSIBLE_VAULT;1.1;AES256
64633735613636656665343436316337626635316161323130323236343039303935656132613937
3031353664346635613431616631663731313339346231390a343233666163633433613232613631
32353163313033323739656664376536333038633038326639653738333435383961666233333661
3633363037366533610a663565646230353239353462333338623164393361386431316330343962
65643839663737623134306366653239626636303866323030323634656365306165373730353935
31333465323839656564633464663962386666663130373032396363323863633936316264663439
653764323731653964653332366564633739
```

Encrypter avec ansible-vault (**encrypt**)

```
[vagrant@ansible secrets]$ cat credentials.yml
---
vault_ansible_password: vagrant
```

ansible-vault encrypt credential.yml

```
[vagrant@ansible secrets]$ ansible-vault encrypt credentials.yml
[DEPRECATION WARNING]: Ansible will require Python 3.8 or newer on the controller
(Red Hat 4.8.5-44)]. This feature will be removed from ansible-core in version 2.1
New Vault password:
Confirm New Vault password:
Encryption successful
[vagrant@ansible secrets]$ cat credentials.yml
$ANSIBLE_VAULT;1.1;AES256
32343762333535336466373863613164633965613437393432343438656235656136666663363334
6161343436333866626130623939313339386262353065630a306539356166626533613737336333
66653963616538336163383737363233333864343061323036616537376331316166353532386361
6530316162303234330a393134303265383565343633613666663630326264333566613364656430
64393130393063356261386336633037306464326430303265396666343135303231376139383665
3937363634313938333464643562613039386261643439626434
```

Decrypter avec ansible-vault (decrypt)

```
[vagrant@ansible secrets]$ ansible-vault decrypt credentials.yml
[DEPRECATION WARNING]: Ansible will require Python 3.8 or newer
(Red Hat 4.8.5-44)]. This feature will be removed from ansible-core
Vault password:
Decryption successful
[vagrant@ansible secrets]$
```

ansible-vault decrypt credential.yml



```
[vagrant@ansible secrets]$ cat credentials.yml
---
vault_ansible_password: vagrant
```

Edition avec ansible-vault (edit)

```
[vagrant@ansible secrets]$ cat credentials.yml
$ANSIBLE_VAULT;1.1;AES256
39653430383238346161653164326139373663326235323831623338666262663664323663353136
3137313065346439303137653562363536383934336362630a366137326331333537303537356636
31313863313436313063393030353438363737323530356436643632376339656234663435366436
3762326263316239330a666239383865613932626162626365323035646637626238333737353966
31363562656536343365626439313036656664663332663962303932323661373362646635373165
3531623563356661646333663163666335333436356532336136
```



```
[vagrant@ansible secrets]$ ansible-vault edit credentials.yml
[DEPRECATION WARNING]: Ansible will require Python 3.8 or newer (Red Hat 4.8.5-44)]. This feature will be removed from ansible-core in a future version.
vault password:
```



```
---
vault_ansible_password: vagrant
czduciencudon
```



```
[vagrant@ansible secrets]$ cat credentials.yml
$ANSIBLE_VAULT;1.1;AES256
66666230396339343536363865343233326538396434353230346462356336663862333363373633
3965613566656630393230636239313264613763613836380a3965353333333931333313163336436
39363635643335383565626331333932363036326663323030343861316330313361366234656437
3062653234333761650a653537343165396234643835373137343534383030623962383166333735
39656238386564373566663931653065393934636365626366343965616438643732396138346130
38323662323237383235653863356562343266303636373132663537333732313833313035343362
326661356233623230616337386432623839
[vagrant@ansible secrets]$
```


ansible-vault : lancement de playbook

```
[vagrant@ansible TP8 - sécurité]$ ansible-playbook webapp-v6.yml --ask-vault-pass
[DEPRECATION WARNING]: Ansible will require Python 3.8 or newer on the controller starting with
(Red Hat 4.8.5-44)]. This feature will be removed from ansible-core in version 2.12. Deprecat
Vault password:
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit
[WARNING]: Could not match supplied host pattern, ignoring: prod

PLAY [install webapp v6] *****
skipping: no hosts matched

PLAY RECAP *****

[vagrant@ansible TP8 - sécurité]$
```

Il faut utiliser l'option **--ask-vault-pass** ou alors configure cela dans le fichier de configuration **ansible.cfg**

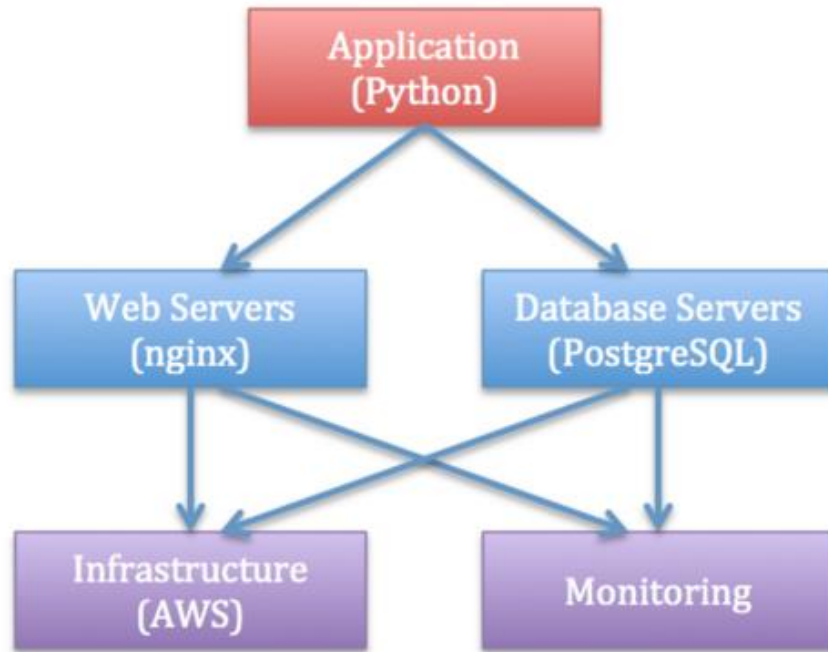
TP-8: Key pair et Vault

- Sécurisation du projet ansible précédent

Plan

- Intérêt d'Ansible et de la formation
- Présentation du formateur
- Introduction au DevOps et Ansible
- Installation et configuration de Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating, loop et condition
- Include et import, tags
- Sécurité
- [Rôle ansible](#)
- Mini-projet

Rôle ansible (1/3): Objectif



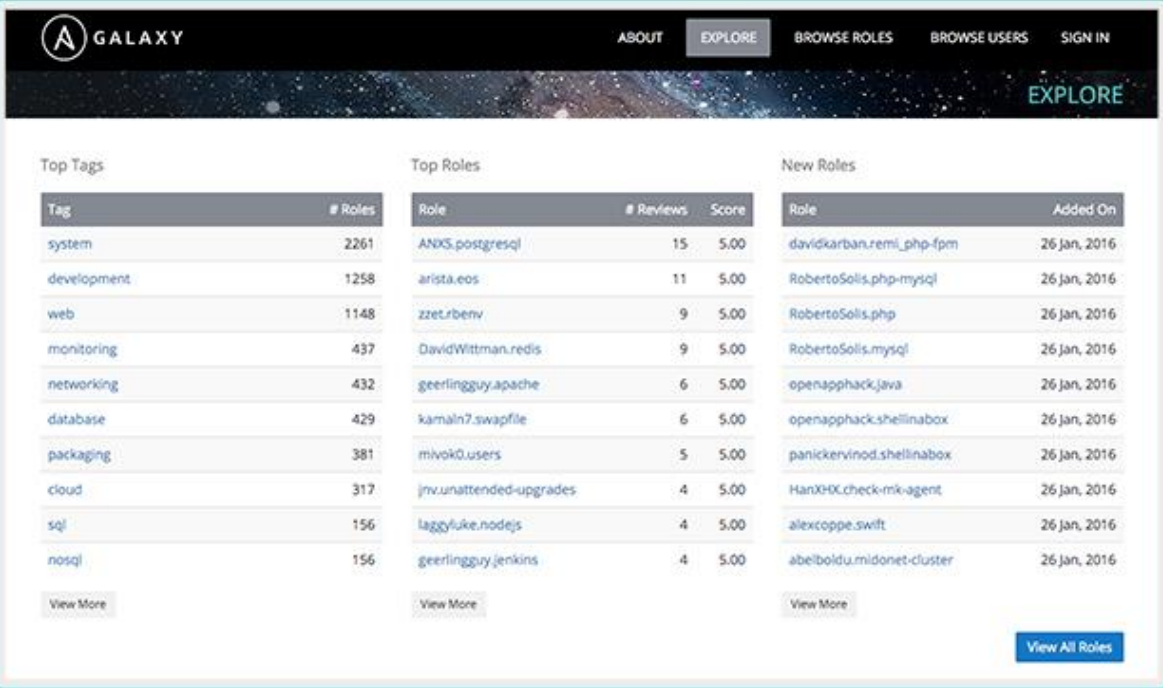
```
roles/  
  app/  
    files/  
    tasks/  
  nginx/  
    templa  
    tasks/  
    vars/  
  postgresql  
    templa  
    tasks/  
    vars/  
  aws/  
    tasks/  
    vars/  
  monitoring  
    tasks/
```

- Templating
- Réutilisable
- Evolutif
- Adaptable

```
├── ansible.cfg
├── hosts
├── roles
│   ├── mongodb
│   │   ├── defaults
│   │   │   └── main.yml
│   │   ├── files
│   │   ├── handlers
│   │   │   └── main.yml
│   │   ├── meta
│   │   │   └── main.yml
│   │   ├── README.md
│   │   ├── tasks
│   │   │   └── main.yml
│   │   ├── templates
│   │   ├── tests
│   │   │   ├── inventory
│   │   │   └── test.yml
│   │   └── vars
│   │       └── main.yml
│   └── ...
```

Rôle ansible (2/3): Structure

Rôle ansible (3/3): Galaxy Ansible



The screenshot shows the Ansible Galaxy website interface. At the top, there is a navigation bar with links: ABOUT, EXPLORE, BROWSE ROLES, BROWSE USERS, and SIGN IN. Below the navigation bar is a header with the Ansible Galaxy logo and a large 'EXPLORE' button. The main content area is divided into three columns: Top Tags, Top Roles, and New Roles.

Top Tags

Tag	# Roles
system	2261
development	1258
web	1148
monitoring	437
networking	432
database	429
packaging	381
cloud	317
sql	156
nosql	156

[View More](#)

Top Roles

Role	# Reviews	Score
ANXS.postgresql	15	5.00
arista.eos	11	5.00
zzet.rbnv	9	5.00
DavidWittman.redis	9	5.00
geerlingguy.apache	6	5.00
kamain7.swapfile	6	5.00
mlvok0.users	5	5.00
jnv.unattended-upgrades	4	5.00
laggyluke.nodejs	4	5.00
geerlingguy.jenkins	4	5.00

[View More](#)

New Roles

Role	Added On
davidkarban.remi_php-fpm	26 Jan, 2016
RobertoSolis.php-mysql	26 Jan, 2016
RobertoSolis.php	26 Jan, 2016
RobertoSolis.mysql	26 Jan, 2016
openapphack.java	26 Jan, 2016
openapphack.shellinabox	26 Jan, 2016
panickervinod.shellinabox	26 Jan, 2016
HanXHHC.check-mik-agent	26 Jan, 2016
alexcoppe.swift	26 Jan, 2016
abelboldu.midonet-cluster	26 Jan, 2016

[View More](#)

[View All Roles](#)

- Rôle communautaire
- Classé par catégorie
- Activement maintenu
- Chaque rôle est lié à un repo github
- Ansible-galaxy

TP-8: Déployez wordpress

- Créez un cluster (1 ansible et 1 client) et récupérez votre code provenant du TP-7 depuis github
- Créez un playbook wordpress.yml afin de déployez wordpress (sous forme de conteneur docker) sur le client à l'aide du rôle suivant :
<https://github.com/diranetafen/ansible-role-containerized-wordpress.git>
- Lancez le playbook et vérifiez que marche comme sur des roulettes et que vous avez bien wordpress (vous pouvez exposer wordpress sur le port externe de votre choix, 80 ou 8080 ou autre, tant qu'il n'est pas déjà utilisé sur votre machine client par un autre conteneur)

Plan

- Intérêt d'Ansible et de la formation
- Présentation du formateur
- Introduction au DevOps et Ansible
- Installation et configuration de Ansible
- Commandes AD-HOC
- Découverte du yaml
- Inventaire ansible
- Playbook
- Templating, loop et condition
- Include et import, tags
- Sécurité
- Rôle ansible
- **Mini-projet**

Mini-projet: Créez votre propre rôle webapp

- Vous avez reçu la demande d'une autre équipe qui souhaiterait utiliser votre playbook webapp, mais sous forme de rôle car sous cette forme ils pourront mieux variabiliser et adapter à leur situation
- Leur objectif est que votre rôle possède un playbook tests afin de leur permettre de tester rapidement votre rôle et ainsi l'intégrer à leur process de déploiement, exactement comme le rôle wordpress
- La dernière raison pour laquelle vous devez faire ce rôle est que votre entreprise souhaite mettre en place une galaxy privée pour stocker tous les rôles fait dans l'entreprise <https://github.com/diranetafen/ansible-role-containerized-wordpress#example-playbook>
- A la fin de votre travail, poussez votre rôle sur github et envoyez nous le lien de votre repo à eazytrainingfr@gmail.com et nous vous dirons si votre solution respecte les bonnes pratiques et si votre rôle est utilisable par une entreprise

Merci et à bientôt sur
eazytraining