Global Knowledge

Outils DevOps : Automatisation avec Ansible

Introduction : Présentation d'Ansible

Context :

- L'innovation des entreprises augmente
- Les clients, connectés ou pas, demandent toujours plus. Et plus vite
- Concurrence généralisée, venant de l'extérieur du segment traditionnel du marché

Conséquences

- Environnements de plus en plus complexes
- Travail plus collaboratif (business, dev, QA, opérations)
- Equipes de développement et d'opérations très chargées
- Passer de rôle d'administrateur système au rôle de prestataire des services (ITaaS)

Solutions:

- Modernisation de l'IT (Transformation Numérique)
- Amélioration/Innovation Continues
- Optimisation des Ressources

Introduction : Présentation d'Ansible

Ansible : cas d'utilisation

- Configuration management
- Provisioning
- Orchestration
- Déploiement d'application
- Patching
- Sécurité et conformité

Introduction : Présentation d'Ansible

Ansible et l'automatisation:

- Accélérer les opérations
- Augmenter la cohérence
- Augmenter les capacités libre-service
- Augmenter la disponibilité
- Mieux répondre aux besoins métier
- Améliorer la sécurité et conformité

Introduction : Place et intérêt d'Ansible dans l'éco système DevOps

Ansible et les DEV

Défis :
- Trop de temps passé sur l'outillage
- Pas assez de temps pour les développements

Besoin :
- Travailler au rythme de la demande

Apports d'Ansible :
- Accélération de la boucle de feedback
- Découverte des bugs plus tôt (fail fast/early detection)
- Déploiements plus rapides, coordonnés et plus fiables
- Réduit le risque de « Tribal Knowledge »

Introduction : Place et intérêt d'Ansible dans l'éco système DevOps

Ansible et les OPS

Défis :

- Besoin d'une technologie simple et fiable (i.e : accessible au plus grand nombre)
- Equipes aux compétences différentes

Besoin :

- Gouvernance centralisée d'un système informatique souvent disparate
- Surveillance centralisée du système informatique

Apports d'Ansible :

- Réduire le risque de l'Informatique cachée/fantôme (Shadow IT)
- Réduire le temps de déploiement
- Réduire le risque de connaissance tribale (tribal knowledge)
- Déployer les correctifs de manière automatisée

Introduction : Place et intérêt d'Ansible dans l'éco système DevOps

Ansible et les QA/Sécurité

Défi :

- Détecter ce qui a changé, où et quand

Besoin :

- Réduire le risque des erreurs humaines
- Réduire le risque de malveillance

Apports d'Ansible :

- Créer des environnements DEV, QA et PROD identiques
- Déployer plus rapidement, de manière plus coordonnée et plus fiable
- Créer les bases de la sécurité du système informatique
- Augmentez la visibilité et la précision des exigences de conformité
- Soulager le poids de la documentation traditionnelle (création d'une documentation vivante et testable)

Introduction : Place et intérêt d'Ansible dans l'éco système DevOps

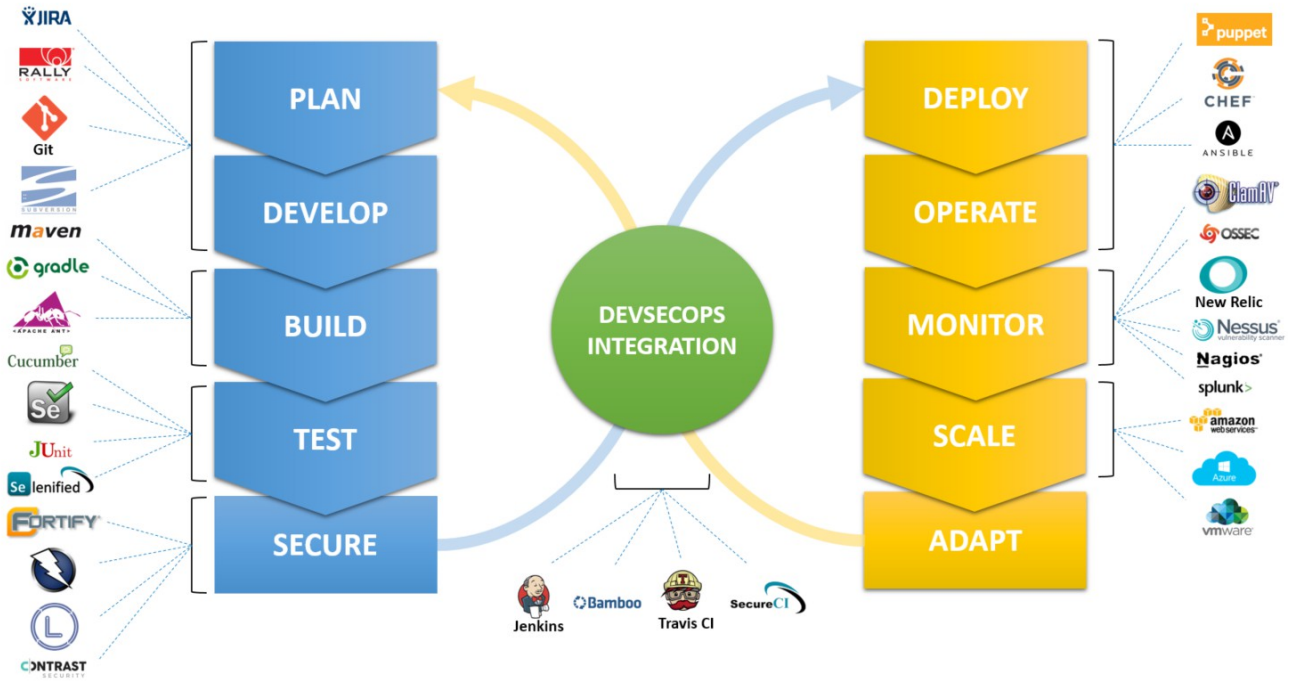
Ansible et le Business

Défi :
- Améliorer le Time To Market (TTM)
- Améliorer le Time To Value (TTV)

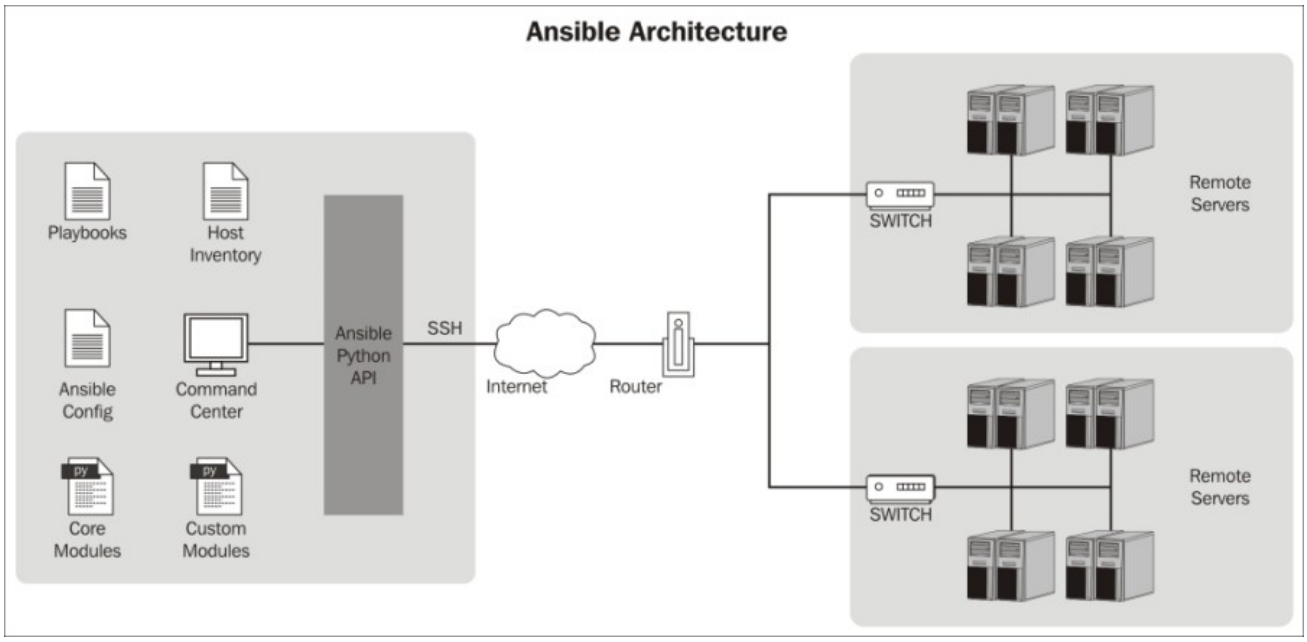
Besoin :
- Acquérir un avantage concurrentiel

Apports d'Ansible :
- Aligner l'IT avec le business
- Augmenter le temps pour l'innovation et la stratégie

Introduction : Place et intérêt d'Ansible dans l'éco système DevOps



Vue d'ensemble d'Ansible : Architecture d'Ansible



Vue d'ensemble d'Ansible : Ansible et la concurrence

Ansible et la concurrence:

Pourquoi Ansible:

- Agent-less: pas besoin d'installer un agent sur les machines distantes
- Pas de serveur centralisé
- Communication sécurisée basée sur SSH
- Outil Plusieurs en 1 : gestion des configurations, déploiement, orchestration, provisioning …etc.
- Modules custom (pouvant être écrits dans n'importe quel langage)
- Utilisation du format YAML ou JSON

Vue d'ensemble d'Ansible : Ansible Inventory

L'inventaire Ansible (host inventory) :

- Liste des machines gérées par Ansible
- Utilisé par Ansible pour communiquer avec les machines
- Groupe logique des machines par rôles
- Statique ou dynamique
- Statique par défaut: fichier /etc/ansible/hosts

Vue d’ensemble d’Ansible : Ansible Inventory

Exemple d’inventaire statique:

```
[webservers]
webserver1.example.com
webserver2.example.com
webserver3.example.com

[databases]
database.example.com

[loadbalancers]
loadbalancer.example.com
```

Vue d'ensemble d'Ansible : Ansible Dynamic Inventory

Exemple d'inventaire dynamique:

La liste des machines est créée dynamiquement en interrogeant une source externe.

- AWS EC2
- Rackspace
- OpenStack
- Scripts
- …etc.

Vue d'ensemble d'Ansible : Ansible Configuration Settings

Fichier de configuration principale par défaut : /etc/ansible/ansible.cfg

Ansible recherche le fichier de configuration dans les emplacements ci-dessous et prendra le premier trouvé :

- variable d'environnement ANSIBLE_CONFIG
- ansible.cfg dans le répertoire courant
- ~/.ansible.cfg dans le répertoire home
- /etc/ansible/ansible.cfg

Exemple de fichier de configuration ansible.cfg :

```
#chemin de l'inventaire par défaut
inventory = /etc/ansible/inventory
```

Vue d'ensemble d'Ansible : Ansible Modules

Ansible Modules :
- Scripts réutilisables exécutés sur la machine cliente
- Prennent des arguments en entrée
- Retournent une réponse au format JSON

Ansible Core Modules :
- Développés et maintenus par Ansible Core Team (i.e: les développeurs RedHat)
- Livrés dans la distribution Ansible officielle

Vue d'ensemble d'Ansible : Ansible Modules

Ansible Community Modules :
- Développés et maintenus par la communauté
- Non maintenus par Ansible Core Team
- Peuvent être promus Core Module
- Livrés dans la distribution Ansible officielle

Ansible Custom Modules :
- Développés par l'utilisateur
- Peuvent être proposés à la communauté

Vue d'ensemble d'Ansible : Installation d'Ansible

Installation uniquement sur le contrôleur Ansible :

- Redhat/Centos :
  - sudo yum update && sudo yum -y upgrade #Upgrade
  - sudo yum -y install epel-release
  - sudo yum -y install ansible

- Debian/ubuntu :
  - sudo apt-add-repository -y ppa:ansible/ansible
  - sudo apt-get update && sudo apt-get -y upgrade
  - sudo apt-get install python-software-properties
  - sudo apt-get install -y ansible

Vue d'ensemble d'Ansible : Ansible ad-hoc CLI

Binaire utilisé pour exécuter les commandes ad hoc : /usr/bin/ansible

- Exécution rapide de commandes CLI à la volée
- Permet d'appréhender la puissance d'Ansible
- Commandes non sauvegardées pour une utilisation ultérieure

Syntaxe :
$ ansible **target_hosts** -i **inventory** -m **module** -a **arguments** -options
target_hosts : machines ou groupe de machines concernées par la commande
inventory : l'inventaire des machines (contient au moins target_hosts)
module : module ansible exécuté
arguments : paramètres passés au module ansible
-options : les options ansible (taper ansible --helps pour voir la liste complète)

Scripts Ansible contenant les travaux pratiques :
https://github.com/jptraining/ansible.git:

– Le fichier Vagrantfile permet de créer le VMs Linux utilisées dans les TP avec Vagrant sur VirtualBox.
– Le répertoire playbook contient les principaux playbook

Vue d'ensemble d'Ansible : Ansible ad-hoc CLI

Exécuter la commande pwd sur les machines distantes avec le module command

- On utilise le fichier d'inventaire hosts
- On exécute la commande sur les machines du groupe webservers
- On passe l'argument pwd au module command
- On exécute le module command

Vue d'ensemble d'Ansible: Ansible ad-hoc CLI

Exécution de commande ansible avec les options -u et -k :

```
$ ansible webservers -i hosts -m command -a "pwd" -u user -k
```

Vue d’ensemble d’Ansible : Ansible ad-hoc CLI

Exécution de commande ansible:
Lister le contenu du répertoire racine des machines distantes avec le module command :

```
$ ansible webservers -i hosts -m command -a "ls -l /" -u user -k
```

Vue d’ensemble d’Ansible : Ansible ad-hoc CLI

Exécuter le module ping [ping] :

```
$ ansible webservers -i hosts -m ping -u user -k
```

Vue d’ensemble d’Ansible : Ansible ad-hoc CLI

Exécuter un script shell sur les machines distantes avec le module script:

```
$ ansible webservers -i hosts -m script -a "my_script.sh 76542" -u
user -k
```

Afficher les faits (metadata) des machines distantes avec le module setup:

```
$ansible all -i hosts -m setup -u user -k
$ansible all -i hosts -m setup -a 'filter=*ipv4*' -u user -k
$ansible all -i hosts -m setup -a 'filter=ansible_eth[0-2]'  -u
user -k
```

Vue d'ensemble d'Ansible : Ansible ad-hoc CLI

Création et enregistrement d'une clef SSH

Ansible Best Practices :

- Utiliser un compte de service pour piloter les machines managées
- Ne jamais utiliser le compte root pour piloter les machines managées
- Ansible doit se connecter aux machines managées par clef ssh
- Bloquer la connexion du compte de service par login/password
- Un humain ne devrait [quasi plus] jamais se connecter sur un serveur managé ! (i.e: si un humain se connecte sur un serveur managé, ce serveur est souillé !)

Créer le group ansible sur les machines distantes avec le module group :

```
$ansible all -i hosts -m group -a "state=present name=ansible" -u
user -k -b -K
```

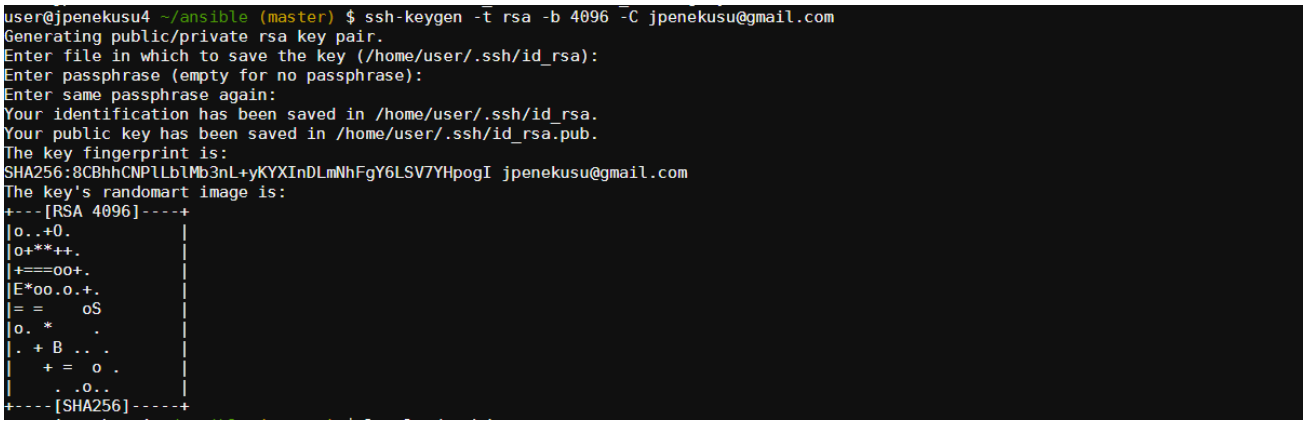
Créer l'utilisateur « ansible » sur les machines distantes avec le module user:

```
$ansible all -i hosts -m user -a "state=present name=ansible
groups=ansible shell=/bin/bash" -u user -k -b -K
```

Vue d’ensemble d’Ansible : Ansible ad-hoc CLI

Générer une clef ssh

`$ssh-keygen`



Vue d'ensemble d'Ansible : Ansible ad-hoc CLI

Copier la clef publique sur les machines distantes avec le module authorized_key:
Copier coller la clef publique dans le paramètre key ou saisir l'url de la clef publique.

```
$ansible all -i hosts -m authorized_key -a "state=present
user=ansible
key='https://raw.githubusercontent.com/jptraining/ansible/master/k
eys/id_rsa.pub'" -u user -k -b -K
```

Exécuter des commandes en s'authentifiant avec la clef ssh:

```
$ansible -h
$ansible all -i hosts --private-key="/home/user/.ssh/id_rsa" -a
"ls -ltr" -u ansible
```

Vue d’ensemble d’Ansible : Ansible ad-hoc CLI

Enregistrer les paramètres par défaut dans le fichier ansible.cfg local :
Fichier ansible.cfg local

```
[defaults]
# uncomment this to disable SSH key host checking
host_key_checking = False

# if set, always use this private key file for authentication, same as
# if passing --private-key to ansible or ansible-playbook
private_key_file = /home/user/.ssh/id_rsa

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

Vue d’ensemble d’Ansible : Ansible ad-hoc CLI

Se connecter en utilisant les paramètres enregistrés dans ansible.cfg :

```
$ansible all -i hosts -a "ls -ltr"
```

Playbook Ansible: Le Format Des Données YAML

Qu’est ce que et pourquoi YAML

- Un format des données (i.e : comme XML, JSON, ...etc)
- Plus facile à lire et à écrire que d’autres formats des données
- Un fichier YAML devrait commencer par trois traits d’union et se terminer par trois points

Playbook Ansible: Le Format Des Données YAML

Exemples de fichiers YAML :

Type liste:
Les éléments d'une liste ont une même indentation et commencent par un trait d'union suivi d'un espace

```
---
cours_deau:
    - Fleuve
    - Riviere
    - Ocean
    - Lac

cours_deau_JSON: [ Fleuve, Riviere, Ocean, Lac ]

...
```

Playbook Ansible: Le Format Des Données YAML

Type clef valeur ou dictionnaire :

Les éléments d'un dictionnaire sont représentés sous la forme clef: valeur (le : est suivi d'un espace) :

```
---
identite:
    nom: Dupont
    prenom: Julien
    fonction: "Ingenieur de production"
    telephone: 0254565455
    departement: Informatique

identite_JSON: { nom: Dupont, prenom: Julien, fonction: "Ingenieur de
production", telephone: 0254565455, departement: Informatique }
...
```

Playbook Ansible: Le Format Des Données YAML

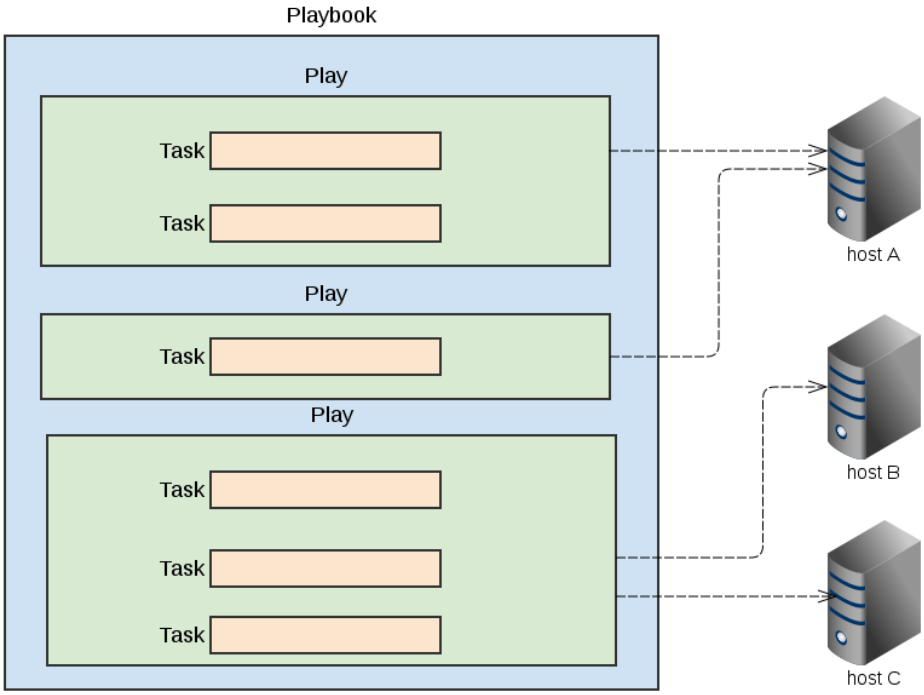
Structure YAML complexe:

```
---
employes:
-  jdupont:
     nom: Dupont
     prenom: Julien
     fonction: Développeur
     Departement: informatique
     languages:
       - python
       - scala
     commentaires : >
        Ce commentaire est très long
        sur une seule ligne grâce au
        caractère > ci-dessus
-  abell:
     nom: Bell
     .etc..

...
```

Validation des fichiers YAML : http://www.yamllint.com/

Playbook Ansible: Ecriture D’un Playbook Ansible

Qu’est-ce qu’un playbook Ansible:



Playbook Ansible: Ecriture D'un Playbook Ansible

Conversion des commandes ad-hoc du $Ansible ad-hoc CLI en Playbooks

Exécuter la commande pwd sur les serveurs managés avec le module command:

Version ad-hoc :
ansible webservers -i hosts -m command -a "pwd"
ansible webservers -i hosts -m command -a "ls -l /"

Version playbook:
Voir le fichier command-playbook.yaml

```yaml
---
- name: "This Is PLAY Documentation: This PLAY Executes Some Basic Commands On The Target Hosts"
  hosts: webservers
  tasks:
  - name: "This Is Task Documentation: This Task Executes The Command pwd On Target Hosts"
    command: pwd
  - name: "This Is Task Documentation: This Task Executes The Command 'ls -l /' On Target Hosts"
    command: ls -l /
```

Playbook Ansible: Ecriture D’un Playbook Ansible

$ansible-playbook -i hosts command-playbook.yaml

Exécuter un script shell sur les machines distantes avec le module script:

$ansible webservers -i hosts -m script -a "my_script.sh Jerome"

Voir le fichier script-playbook.yaml

$ansible-playbook -i hosts script-playbook.yaml

Playbook Ansible: Ecriture D’un Playbook Ansible

Créer des utilisateurs sur les machines managées avec les modules group et user:

Version commandes ad-hoc:

ansible all -i hosts -m group -a "state=present name=ansible"
ansible all -i hosts -m user -a "state=present name=ansible groups=ansible shell=/bin/bash"

Playbook Ansible: Ecriture D'un Playbook Ansible

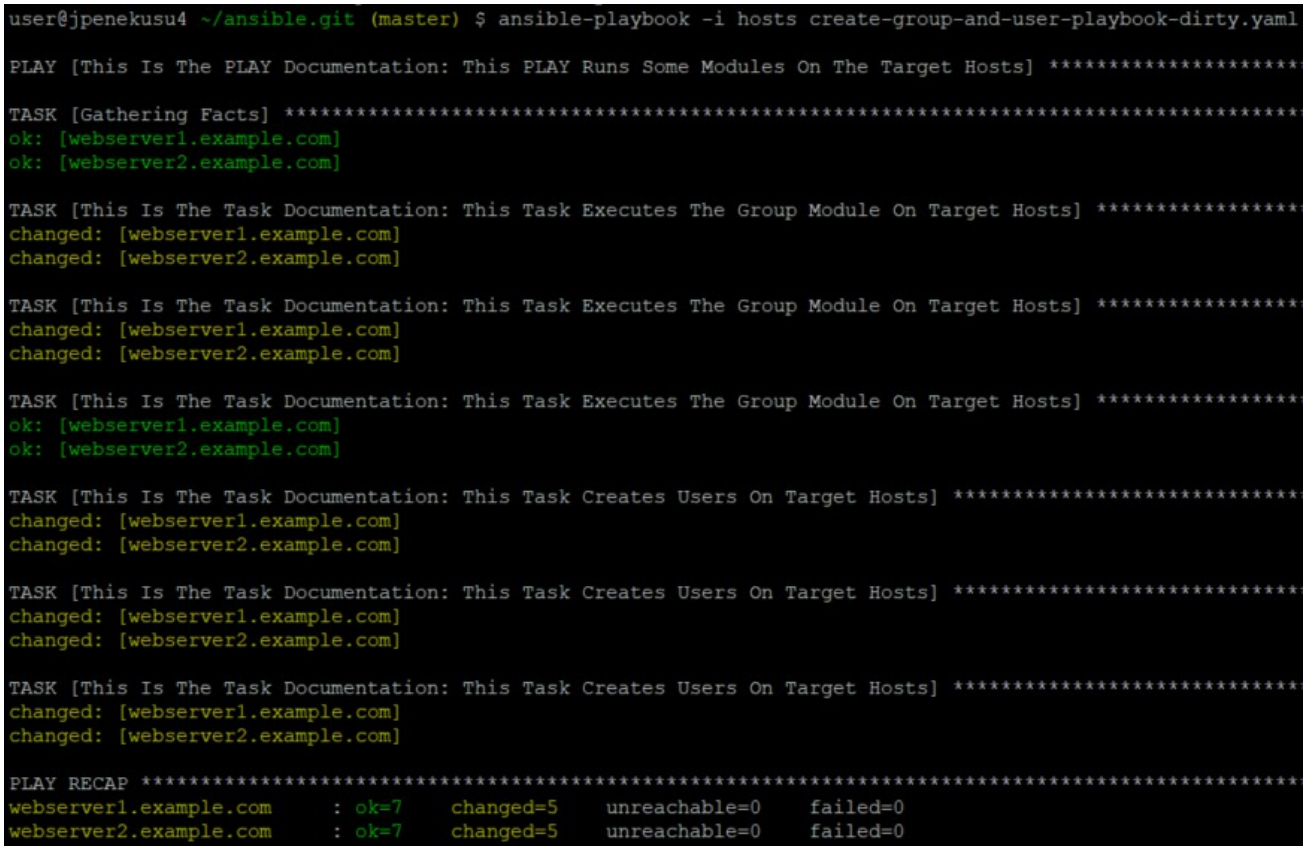
Versions playbook:

Voir le fichier create-group-and-user-playbook-dirty.yaml

$ansible-playbook -i hosts create-group-and-user-playbook-dirty.yaml

Exécuter le playbook sudoers.yaml pour donner les droits sudo NOPASSWD au compte technique ansible:

$ansible-playbook -i hosts sudoers.yaml -e "subject=ansible" -u vagrant -k -K -b



Playbook Ansible: Ecriture D’un Playbook Ansible

Pourquoi dirty dans le nom du fichier create-group-and-user-playbook-**dirty**.yaml :
Le playbook create-group-and-user-playbook-**dirty**.yaml contient plusieurs anti-patterns:

- Les données sont mélangées aux traitements
- Mots de passe en clair (même s’il est hashé)
- L’ajout de nouveaux utilisateurs nécessite la modification du playbook
- Le playbook n’est pas réutilisable
- Maintenance difficile (playbook non réutilisable)

Gestion des variables Ansible: Déclaration et utilisation

- Un playbook doit être réutilisable
- Nécessité d'utiliser les variables pour rendre le code réutilisable

Ansible définit 16 types de variables, fonction de l'endroit où elles sont déclarées.

Gestion des variables Ansible: Déclaration et utilisation

Ci-dessous les principaux types de variables Ansible dans l'ordre de priorité décroissante :

1. Variables extra vars :

Ce sont des variables passées en ligne de commande.
Exemple :
$ansible-playbook -i hosts variables-extra-vars.yaml -e "ma_variable=TOTO"

2. Variables déclarées en paramètre de roles :

Ce sont des variables passées en paramètres aux rôles Ansible.
Exemple:

```
  roles:
    - { role: variables-role, ma_variable: "variable de role" }
```

$ansible-playbook -i hosts variables-role.yaml

Gestion des variables Ansible: Déclaration et utilisation

3. Variables registered vars

Ce sont des variables déclarées avec les instructions set_facts ou register.
Exemple:
$ansible-playbook -i hosts variables-register.yaml

Gestion des variables Ansible: Déclaration et utilisation

4. Variables « facts » récupérées sur les machines distantes avec la directive gather_facts : yes

Ce sont des metadata récupérées par Ansible sur les machines distantes.
Exemples :
$ansible-playbook -i hosts variables-facts.yaml

Gestion des variables Ansible: Déclaration et utilisation

5. Variables spécifiques à une machine distante et déclarées dans le répertoire host_vars

Ce sont des variables appartenant à la machine distante dont le fichier porte le nom.
Exemples :
Soit la machine distante jpenekusu2.mylabserver.com faisant partie de l'inventaire et le fichier host_vars/jpenekusu2.mylabserver.com

Contenu du fichier host_vars/jpenekusu2.mylabserver.com :

```
---
mon_certificat_ssl: |
      mon_certificat_ssl dans host_vars
      jpenekusu2.mylabserver.com
```

Gestion des variables Ansible: Déclaration et utilisation

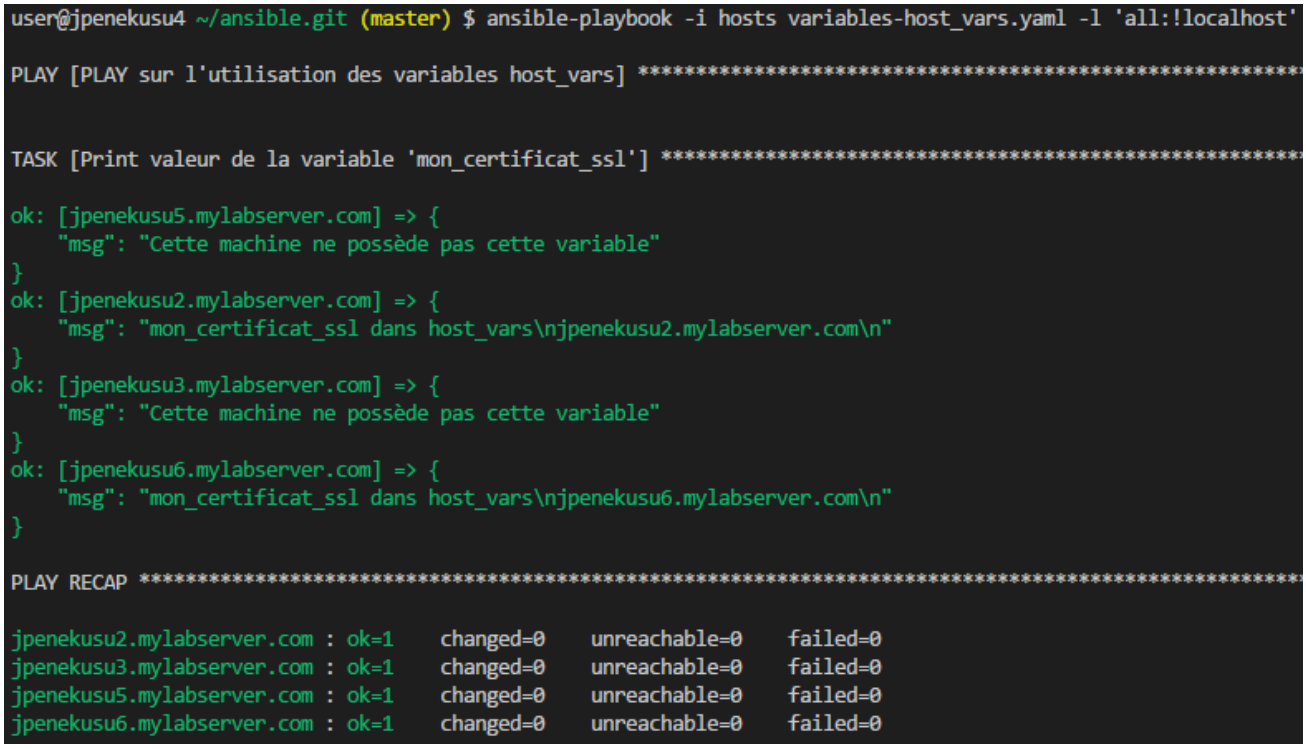
Soit la machine distante jpenekusu6.mylabserver.com faisant partie de l'inventaire et le fichier host_vars/jpenekusu6.mylabserver.com

Contenu du fichier host_vars/jpenekusu6.mylabserver.com :

```
---
mon_certificat_ssl: |
      mon_certificat_ssl dans host_vars
      jpenekusu6.mylabserver.com
```

Gestion des variables Ansible: Déclaration et utilisation

$ansible-playbook -i hosts variables-host_vars.yaml



Gestion des variables Ansible: Déclaration et utilisation

6. Variables spécifiques à un groupe de machines distantes et déclarées dans le répertoire group_vars

Ce sont des variables appartenant à un groupe de machines distantes dont le fichier ou le répertoire porte le nom.
Exemples :
Soit des machines distantes du groupe webservers faisant partie de l'inventaire et le fichier group_vars/webservers/vars.yaml

Contenu du fichier group_vars/webservers/vars.yaml:

```
---
user_groups:
  - sysadmins
  - developpers
```

Gestion des variables Ansible: Déclaration et utilisation

Soit des machines distantes du groupe database faisant partie de l'inventaire et le fichier group_vars/database.yaml

Contenu du fichier group_vars/database.yaml:

```
---
user_groups:
  - dba
  - developpers
```

Gestion des variables Ansible: Déclaration et utilisation

$ansible-playbook -i hosts variables-group_vars.yaml

```
user@jpenekusu4 ~/ansible.git (master) $ ansible-playbook -i hosts variables-group_vars.yaml -l 'all:!localhost'

PLAY [PLAY sur l'utilisation des variables group_vars] ***************************************************


TASK [Print variable 'user_groups'] **********************************************************************

ok: [jpenekusu5.mylabserver.com] => {
    "msg": "[u'dba', u'developpers'] got from group [u'database', u'lampws']"
}
ok: [jpenekusu2.mylabserver.com] => {
    "msg": "[u'sysadmins', u'developpers'] got from group [u'lampws', u'linuxservers', u'webservers']"
}
ok: [jpenekusu3.mylabserver.com] => {
    "msg": "[u'sysadmins', u'developpers', u'whell'] got from group [u'lampws']"
}
ok: [jpenekusu6.mylabserver.com] => {
    "msg": "[u'sysadmins', u'developpers'] got from group [u'linuxservers', u'loadbalancers', u'webservers']"
}

PLAY RECAP ***********************************************************************************************

jpenekusu2.mylabserver.com : ok=1    changed=0    unreachable=0    failed=0
jpenekusu3.mylabserver.com : ok=1    changed=0    unreachable=0    failed=0
jpenekusu5.mylabserver.com : ok=1    changed=0    unreachable=0    failed=0
jpenekusu6.mylabserver.com : ok=1    changed=0    unreachable=0    failed=0
```

Gestion des variables Ansible: Déclaration et utilisation

7. Variables globales au playbook et déclarées dans le fichier group_vars/all

Ce sont des variables globales, accessibles partout dans un playbook ou un rôle et à toutes les machines de l'inventaire.

Exemples :

Soit le fichier group_vars/all.yaml dont le contenu est:

```
---
user_groups:
  - all
  - developpers
  - whell
```

Gestion des variables Ansible: Déclaration et utilisation

$ansible-playbook -i hosts variables-group_vars.yaml

```
user@jpenekusu4 ~/ansible.git (master) $ ansible-playbook -i hosts variables-group_vars.yaml

PLAY [PLAY sur l'utilisation des variables group_vars] *************************************


TASK [Print variable 'user_groups'] ********************************************************

ok: [localhost] => {
    "msg": "[u'all', u'developpers', u'whell'] got from group ['ungrouped']"
}
ok: [jpenekusu2.mylabserver.com] => {
    "msg": "[u'sysadmins', u'developpers'] got from group [u'lampws', u'linuxservers', u'webs
}
ok: [jpenekusu3.mylabserver.com] => {
    "msg": "[u'all', u'developpers', u'whell'] got from group [u'lampws']"
}
ok: [jpenekusu5.mylabserver.com] => {
    "msg": "[u'dba', u'developpers'] got from group [u'database', u'lampws']"
}
ok: [jpenekusu6.mylabserver.com] => {
    "msg": "[u'sysadmins', u'developpers'] got from group [u'linuxservers', u'loadbalancers',
}

PLAY RECAP *********************************************************************************

jpenekusu2.mylabserver.com : ok=1    changed=0    unreachable=0    failed=0
jpenekusu3.mylabserver.com : ok=1    changed=0    unreachable=0    failed=0
jpenekusu5.mylabserver.com : ok=1    changed=0    unreachable=0    failed=0
jpenekusu6.mylabserver.com : ok=1    changed=0    unreachable=0    failed=0
localhost                  : ok=1    changed=0    unreachable=0    failed=0
```

Gestion des variables Ansible: Déclaration et utilisation

8. Variables par défaut d'un rôle

Ce sont des variables déclarées dans le fichier role/defaults/main.yaml.

main.yml

```
# defaults file for variables-role
ma_variable: ma_varaible role/default
ma_variable_default: ma_variable role/default
```

Exemples:
$ansible-playbook -i hosts variables-role-defaults.yaml

Rôles Ansible: Intérêt des rôles Ansible

Les playbooks permettent d'automatiser des taches simples (i.e : ansible ad-hoc CLI évolués)
Les playbooks complexes ne sont pas réutilisables
Les playbooks complexes ne sont pas maintenables
Un rôle Ansible est un ensemble des taches, fichiers, templates de fichiers et variables groupés dans une structure afin de jouer un rôle déterminé et de rendre les playbooks réutilisables et maintenables.

Rôles Ansible: Intérêt des rôles Ansible

Organisation d'un projet Ansible :

```
site.yml
webservers.yml
databaseserver.yml
roles/
   common/
     tasks/
     handlers/
     files/
     templates/
     vars/
     defaults/
     meta/
   webservers/
     tasks/
     defaults/
     meta/
…etc.
```

Rôles Ansible: Intérêt des rôles Ansible

Chaque répertoire d’un rôle contient au moins un fichier, le main.yml

Listes et rôles de chaque répertoire d’un rôle ansible :

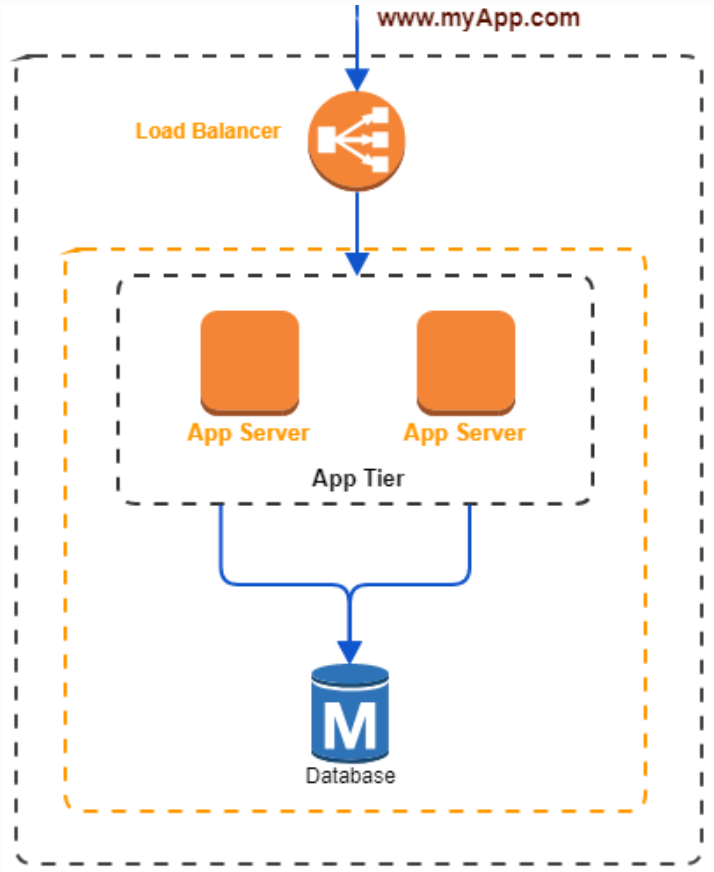
- tasks – contient la liste de principales taches exécutées par le rôle.
- handlers - contient les taches exécutées uniquement si elles sont notifiées.
- defaults – variables par défaut du rôle, cfr. $Gestion des variables Ansible.
- vars - variables vars du rôle.
- files - contient les fichiers déployés par le rôle.
- templates - contient les fichiers template déployés par le rôle.
- meta – contient les metadata du rôle (voir les détails plus bas).

Rôles Ansible: Intérêt des rôles Ansible

Soit le schéma suivant :

Ecrire les playbooks et les rôles permettant de déployer :

- Un serveur d'application
- Une base des données



Rôles Ansible: Intérêt des rôles Ansible

Liste des composants à écrire pour implémenter l'architecture ci-dessus:

- un rôle create-automation-user de création de l'utilisateur technique ansible
- un rôle common d'installation du socle commun
- un rôle webserver d'installation du serveur web
- un rôle webserver-app de déploiement de l'application web
- un rôle database d'installation de la base des données
- un playbook principal d'orchestration de différents rôles

Rôles Ansible: Intérêt des rôles Ansible

Créer un nouveau répertoire :

$mkdir tp-roles

$cd tp-roles

Rôles Ansible: Intérêt des rôles Ansible

Ecrire le playbook principal site.yaml:

Fichier site.yaml (les rôles seront décommentés au fur et à mesure qu'ils seront écrits et prêts à être testés)

Le fichier site.yaml est le playbook chapeau qui va installer notre architecture. Il faut maintenant créer les composants sous-jacents.

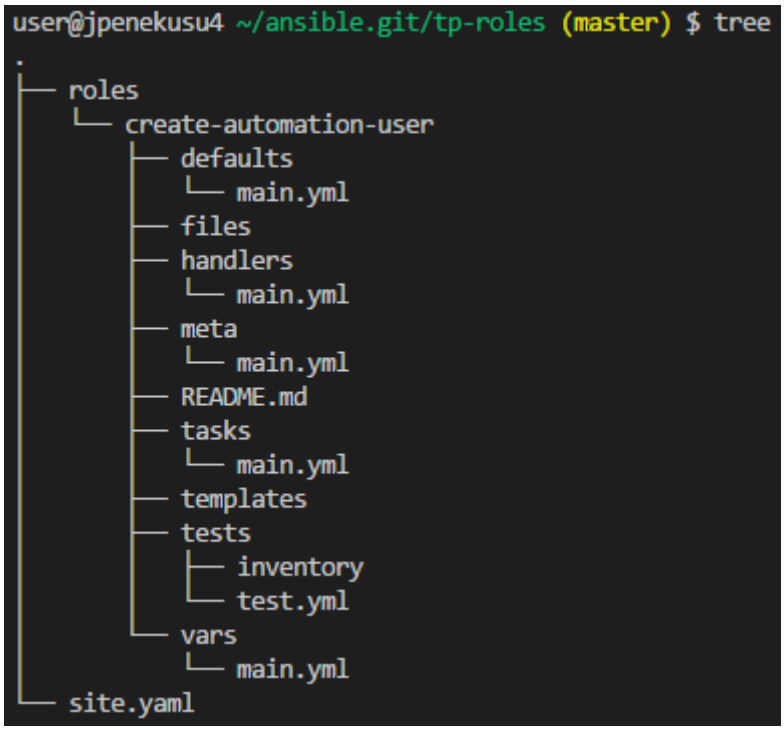
```
---
- name: "PLAY de création de l'utilisateur technique ansible"
  hosts: all
  remote_user: vagrant
  become: yes
  roles:
  - role: create-automation-user

- name: "PLAY de création de la base des données"
  hosts: database
  remote_user: ansible
  become: yes
  roles:
  - role: common
#  - role: database

- name: "PLAY de création du serveur web"
  hosts: webservers
  remote_user: ansible
  become: yes
  roles:
  - role: common
#  - role: webserver

- name: "PLAY de création de l'application web"
  hosts: webservers
  remote_user: ansible
#  roles:
#  - role: webapp
```

Rôles Ansible: Intérêt des rôles Ansible

Ecrire le rôle create-automation-user:

$mkdir roles

$ansible-galaxy init roles/create-automation-user



Rôles Ansible: Intérêt des rôles Ansible

- Copier le contenu du fichier create-automation-user.yaml (cfr. $playbook) dans roles/create-automation-user/tasks/main.yml

- Supprimer toutes les lignes jusqu'à l'étiquette « tasks » incluse

- Décaler le code à gauche jusqu'à ce que les traits le plus à gauche touchent la bordure

- Variabiliser le maximum des données avec des valeurs par défaut

```
---
# tasks file for roles/create-automation-user---
- name: Ensure ansible user exists
  user:
    name: ansible
    state: present
    comment: Ansible automation user
- name: Ensure ansible user accepts the SSH key
  authorized_key:
    user: ansible
    state: present
    key: https://raw.githubusercontent.com/jptraining/ansible/master/keys/id_rsa.pub
- name: Ensure the line /etc/sudoers.d is present in sudoers file
  lineinfile:
    dest: /etc/sudoers
    state: present
    create: yes
    regexp: '^#includedir /etc/sudoers.d'
    line: '#includedir /etc/sudoers.d'
    validate: 'visudo -cf %s'
- name: Ensure the ansible user is sudoer with no password required
  lineinfile:
    dest: /etc/sudoers.d/ansible
    state: present
    create: yes
    regexp: '^ansible ALL\='
    line: 'ansible ALL=(ALL) NOPASSWD:ALL'
    validate: 'visudo -cf %s'
```

- Ce rôle n'est pas facilement reutilisable :

  - Les données sont mélangées aux traitements. Il doit être optimisé en variabilisant les données.

  - La présence des données obligatoires doit être vérifiée

Rôles Ansible: Intérêt des rôles Ansible

Créer un fichier ansible.cfg dans le répertoire tp-roles avec le contenu suivant :

```
[defaults]
# uncomment this to disable SSH key host checking
host_key_checking = False

# if set, always use this private key file for authentication, same as
# if passing --private-key to ansible or ansible-playbook
private_key_file = /home/user/.ssh/jptraining_ansible_id_rsa

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

inventory = hosts
```

Rôles Ansible: Intérêt des rôles Ansible

Créer un fichier inventaire hosts dans le répertoire tp-roles avec le contenu suivant :

```
[webservers]
Webserver1.mylabserver.com
Webserver2.mylabserver.com

[database]
database.mylabserver.com
```

Rôles Ansible: Intérêt des rôles Ansible

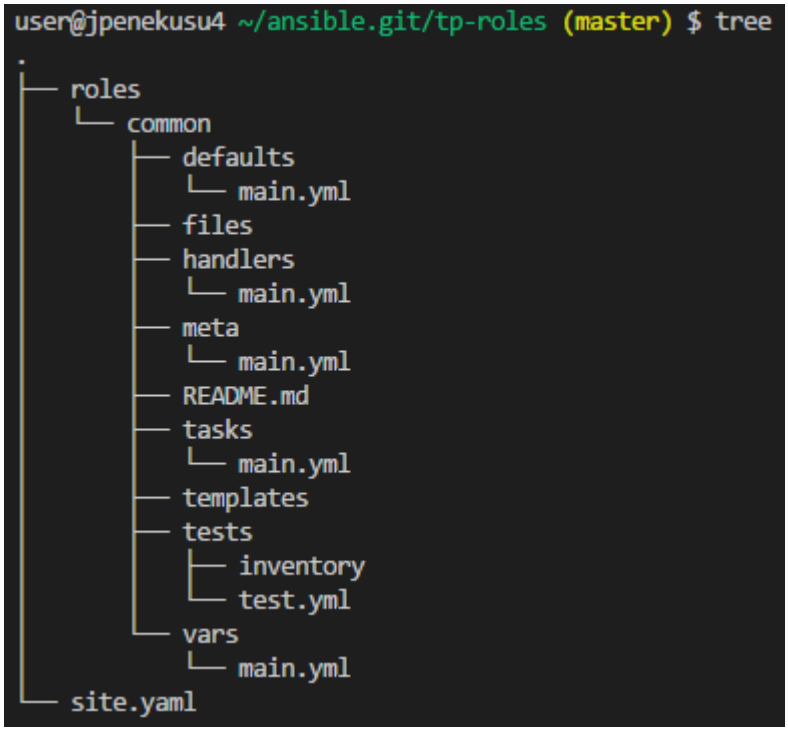
Ecrire le rôle common:

$mkdir roles

$ansible-galaxy init roles/common

- roles/common was created successfully

$tree



Rôles Ansible: Intérêt des rôles Ansible

- Copier le contenu du fichier common-config.yaml (cfr. $playbook) dans roles/common/tasks/main.yml

- Supprimer toutes les lignes jusqu'à l'étiquette « tasks » incluse

- Décaler le code à gauche jusqu'à ce que les traits le plus à gauche touchent la bordure

```
---
# tasks file for roles/common
- name: Vérfifier que le dépot des binaires yum EPEL est présent
  yum:
    name: epel-release
    state: present
- name: Vérfifier que libselinux-python est present
  yum:
    name: libselinux-python
    state: present
- name: Vérfifier que libsemanage-python est present
  yum:
    name: libsemanage-python
    state: present
- name: Vérfifier que NTP est present
  yum:
    name: ntp
    state: present
- name: Vérfifier que le timezone est positionné à UTC
  file:
    src: /usr/share/zoneinfo/Europe/Paris
    dest: /etc/localtime
    state: link
- name: Vérfifier que le service NTP est lancé et est activé
  service:
    name: ntpd
…etc.
```

Rôles Ansible: Intérêt des rôles Ansible

- Ce rôle n'est pas facilement reutilisable :

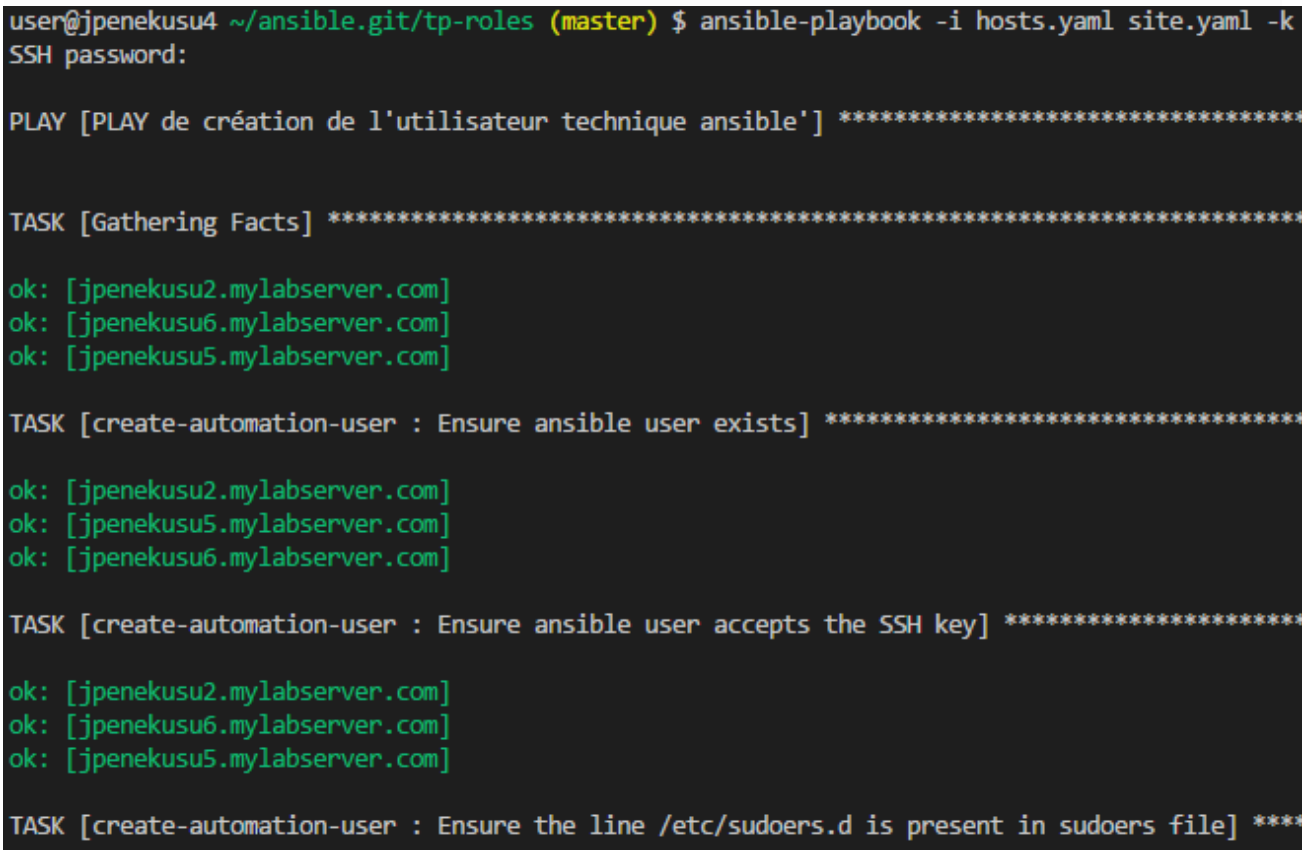
o Les données sont mélangées aux traitements. Il doit être optimisé en variabilisant les données.

o Il faut réduire le nombre des variables en itérant sur les installations yum

o La présence des données obligatoires doit être vérifiée

Exécuter le playbook site.yaml:

$ ansible-playbook -i hosts.yaml site.yaml -k



Rôles Ansible: Intérêt des rôles Ansible

Tester le playbook et le rôle common :

$ansible-playbook site.yaml

```
user@jpenekusu4 ~/ansible.git/tp-roles (master) $ ansible-playbook site.yaml

PLAY [PLAY de création de la base des données] ****************************

TASK [Gathering Facts] ****************************************************
ok: [jpenekusu5.mylabserver.com]

TASK [common : Vérfifier que le dépot des binaires yum EPEL est présent] ****
ok: [jpenekusu5.mylabserver.com]

TASK [common : Vérfifier que libselinux-python est present] ***************
ok: [jpenekusu5.mylabserver.com]

TASK [common : Vérfifier que libsemanage-python est present] ***************
ok: [jpenekusu5.mylabserver.com]

TASK [common : Vérfifier que NTP est present] *****************************
```

Rôles Ansible: Intérêt des rôles Ansible

Créer et écrire le rôle webserver:

$ansible-galaxy init roles/webserver

- roles/webserver was created successfully

- Taper la commande tree pour afficher l'arborescence des répertoires
- Copier le contenu du fichier webserver.yaml (cfr. $playbook) dans roles/webserver/tasks/main.yml
- Supprimer toutes les lignes jusqu'à l'étiquette « tasks » incluse

Rôles Ansible: Intérêt des rôles Ansible

- Décaler le code à gauche jusqu’à ce que les traits le plus à gauche touchent la bordure

```
---
# tasks file for roles/webserver
- name: Vérifiez que le package httpd est installé
  yum:
    name: httpd
    state: present
- name: Vérifiez que le service httpd est activé et en cours d'exécution
  service:
    name: httpd
    state: started
    enabled: True
- name: Vérifier que la configuration du serveur apache est à jour
  template:
    src: httpd.conf.j2
    dest: /etc/httpd/conf/httpd.conf
  notify: Rédemarrer le serveur apache
- name: vérifier que http peut passer le pare-feu
  firewalld:
    service: http
    state: enabled
    permanent: True
    immediate: True
- name: vérifier que https peut passer le pare-feu
  firewalld:
    service: https
    state: enabled
    permanent: True
    immediate: True
```

Rôles Ansible: Intérêt des rôles Ansible

Il est possible de rendre ce script plus maintenable en variabilisant et/ou en itérant sur le module firewalld (cfr. les fichiers optimal.x.main.yaml).

- Copier le contenu ci-dessous dans le fichier des handler webserver/handlers/main.yml

```yaml
---
# handlers file for roles/webserver
- name: Rédemarrer le serveur apache
  service:
    name: httpd
    state: restarted
```

Rôles Ansible: Intérêt des rôles Ansible

- Copier le contenu ci-dessous dans le fichier des variables par défaut webserver/default/main.yml

```
---
# defaults file for roles/webserver
http_port: 80
http_charset: "UTF-8"
```

Rôles Ansible: Intérêt des rôles Ansible

- Créer et écrire le rôle database: ce rôle installe et configure la base des données
- $mkdir roles
- $ansible-galaxy init roles/database
- roles/database was created successfully
- $tree

Rôles Ansible: Intérêt des rôles Ansible

- Nous n'allons pas écrire ce rôle entièrement, nous le ferons dans un TP plus complet plus tard. Nous allons juste afficher un message en lieu et place de la base des données.

- Ecrire le contenu ci-dessous dans le fichier roles/database/tasks/main.yml :

```
---
# tasks file for roles/database
- name: Vérifier que la base des données est installée
  debug:
    msg: The database is up and running
```

Rôles Ansible: Intérêt des rôles Ansible

Créer et écrire le rôle webapp:

ansible-galaxy init roles/webapp

- roles/webapp was created successfully

- Taper la commande tree pour afficher l'arborescence des répertoires

- Copier le contenu du fichier webapp.yaml (cfr. $playbook) dans roles/webapp/tasks/main.yml

Rôles Ansible: Intérêt des rôles Ansible

- Supprimer toutes les lignes jusqu'à l'étiquette « tasks » incluse
- Décaler le code à gauche jusqu'à ce que les traits le plus à gauche touchent la bordure

```
---
# tasks file for roles/webapp
- name: Vérifier que le site web est présent et à jour
  template:
    src: index.html.j2
    dest: /var/www/html/index.html
    owner: root
    group: root
    mode: 0644
  become: True
```

- Copier le ficher index.html.j2 (cfr. $playbook) dans roles/webapp/templates/index.html.j2
- Décommenter le rôle webapp dans site.yaml et exécuter le playbook

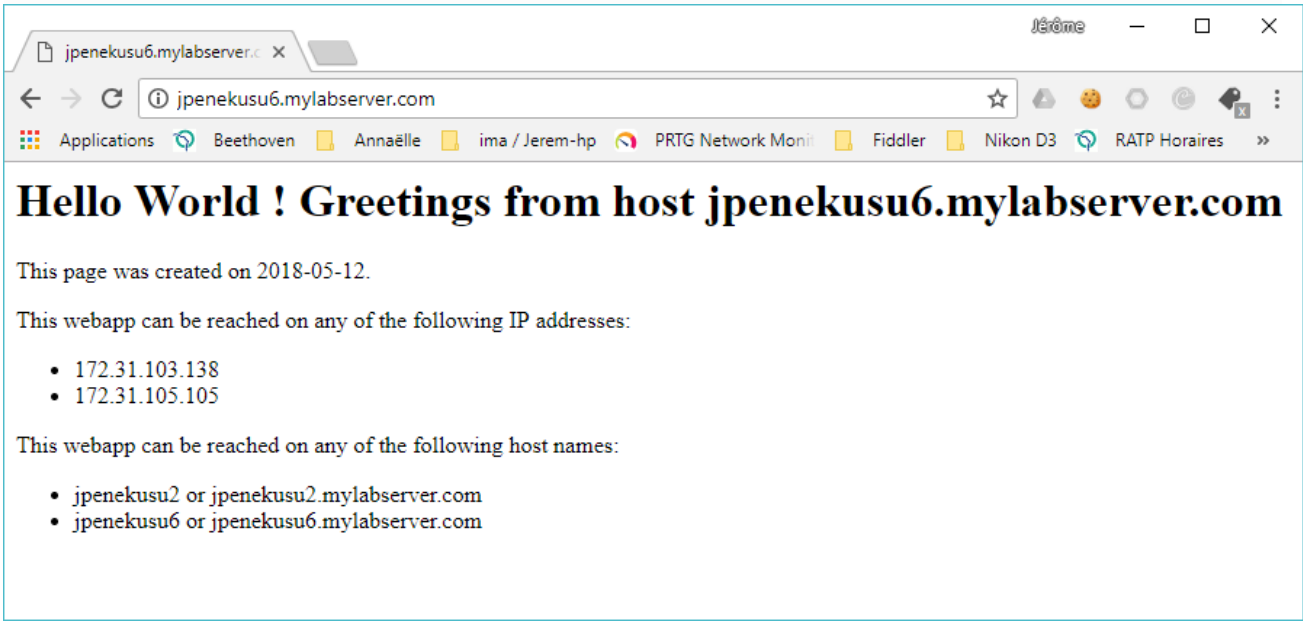
$ ansible-playbook -i hosts.yaml site.yaml

- Regarder et analyser les logs d'exécution du playbook

Rôles Ansible: Intérêt des rôles Ansible

- Se connecter les serveurs d'application et vérifier qu'ils fonctionnent

http://un_server_du_groupe_webservers.com



Concepts avancés: les itérations avec loop

Les loops sont :

- Equivalent aux boucles en langage de programmation
- Permettent d'effectuer plusieurs traitements dans une même tache.

Cfr. répertoire loops dans le projet des TP.

Doc officielle: http://docs.ansible.com/ansible/playbooks_loops.html

Concepts avancés, les templates Jinja2

Jinja 2 est un moteur de templates pour python.

Les templates sont des fichiers texte réutilisables :

- Séparation de données (i.e : le Quoi) et de traitements (i.e : le Comment).
- Exécution rapide (très performant).
- Possibilité d'hériter les templates.
- Enabler DevOps

Concepts avancés, les templates Jinja2

Les variables sont délimitées par **{{ … }}**
Les expressions (boucles, conditions, macros, blocs) sont délimitées **{% … %}**

L'expression **for**:

```
system {
        host_name {{ host_name }};
        server_names {
        {% for dns_server in dns_servers %}
        {{ dns_server }};
        {% endfor %}
        }
}
```

Soit le fichier group_vars/all

```
host_name: myserver
dns_servers:
  - dnsserver1.com
  - dnsserver2.com
  - dnsserver3.com
```

Résultat de l'exécution du template par Jinja2 :

```
system {
        host_name myserver;
        server_names {
        dnsserver1.com;
        dnsserver2.com;
        dnsserver3.com;
        }
}
```

Concepts avancés, les templates Jinja2

L'expression **if**:

```
user {{ user.name }} {
        authentication {
        {% if user.password %}
                password is {{ user.pwd }}
        {% elif user.ssh-key %}
                ssk key is {{ user.ssh-key }}
        {% endif %}
        }
}
```

Résultat de l'exécution du template par Jinja2 :

user **Jeremy** {
    authentication {
        password is $hmmlkj+T
    }
}

Concepts avancés, les filtres Jinja2

Les filtres Jinja 2 :

- Fonctions Python prenant des arguments :
  - {{ ma_variable | filtre_jinja2 }}
  - {{ ma_variableN | filtre_jinja2(ma_variable1, ma_variableX, ...etc) }}
- Manipulent directement les données dans les templates ou les fichiers classiques.
- La librairie Jinja 2 fournit nativement plusieurs filtres (cfr. http://jinja.pocoo.org/docs/dev/templates/#builtin-filters)

```
{% for interface_name in  interfaces %}
        {{ interface_name.name }} {
                interface description "{{ interface_name.desc | upper }}"
        }
{% endfor %}
```

Résultat de l'exécution du template par Jinja2 :

```
interface1 {
        interface description "DESCRIPTION INTERFACE 1"
}
interface2 {
        interface description "DESCRIPTION INTERFACE 2"
}
interface3 {
        interface description "DESCRIPTION INTERFACE 3"
}
```

Autres exemples d'utilisation de filtres:

Voir le playbook filters.yaml

Concepts avancés, la directive include template Jinja2

Les templates Jinja 2 peuvent inclure :

- le contenu de fichiers (ex. clef ssh, ...etc)
- le contenu d'autres templates

Le but est de diviser un fichier template en plusieurs.

```
configuration {
        {% include "service1.j2" %}
        {% include "service2.j2" %}
}
```

Concepts avancés, l'héritage des templates Jinja2

Jinja 2 offre le mécanisme d'héritage des templates:

- défintion d'un template de base avec des éléments communs et des valeurs par défaut
- définition des blocks pouvant être surchargés par les templates héritant du template de base

Soit le template parent parent.j2 ci-dessous contenant un block dhcp :

```
services {
        https;
        ssh;
        {% block dhcp %}
        dhcp{
                pool 128.128.128.0/24;
        }
        {% endblock dhcp %}
}
```

La configuration se trouvant à l'intérieur du block dhcp est la valeur par défaut utilisée si elle n'est pas surchargée par le template fils.

Concepts avancés, l'héritage des templates Jinja2

Soit le template fils fils.j2 qui hérite du template parent parent.j2 surchargeant le block dhcp :

```
{% extends "parent.j2" %}
{% block dhcp %}
        dhcp {
                pool 10.10.10.0/24 {
                        domain-name mydomain.com;
                }
        }
{% endblock dhcp %}
```

Concepts avancés, l'héritage des templates Jinja2

Résultat de l'exécution du template:

```
services {
        https;
        ssh;
        dhcp{
                pool 10.10.10.0/24 {
                        domain-name mydomain.com;
                }
        }
}
```

Concepts avancés, delegate_to

Le plugin delagate_to permet d'exécuter une taches sur une machine différente de la courante et est utilisé dans les scénarios de déploiements complexes:

- Appel d'une API pour activer/désactiver un serveur web du loadbalancer
- Exécution d'une tache quelconque sur une machine différente

Exemple:

$ansible-playbook delegate_to.yaml

```
user@jpenekusu4 ~/ansible.git (master) $ ansible-playbook delegate_to.yaml

PLAY [database] ****************************************************************


TASK [Compter les process sur le serveur distant] ******************************

changed: [jpenekusu5.mylabserver.com]

TASK [Afficher le nombre de process sur le serveur distant] ********************

ok: [jpenekusu5.mylabserver.com] => {
    "msg": "Nombre de process sur le serveur distant : 7"
}

TASK [Compter les process sur le serveur local] ********************************

changed: [jpenekusu5.mylabserver.com -> localhost]

TASK [Afficher le nombre de process sur le serveur local] **********************

ok: [jpenekusu5.mylabserver.com] => {
    "msg": "Nombre de process sur le serveur local : 11"
}

PLAY RECAP *********************************************************************

jpenekusu5.mylabserver.com : ok=4    changed=2    unreachable=0    failed=0
```

Concepts avancés, les conditions : when

L'instruction when permet de gérer les conditions.

Soit le code suivant en langage impératif :

```
If os = "redhat"
  Install httpd
Else if os = "debian"
  Install apache2
End
```

Son équivalent en langage déclaratif Ansible est :

```yaml
tasks:
  - name: Afficher la valeur de la variable fact ansible_os_family
    debug:
      msg: 'ansible_os_family = {{ ansible_os_family }}'
  - name: Vérifier que apache est installé sur les machines RedHat et à jour
    yum:
      name: httpd
      state: latest
    when: ansible_os_family == 'RedHat'
  - name: Vérifier que apache est installé sur les machines Debian et à jour
    apt:
      name: apache2
      state: latest
    when: ansible_os_family == 'Debian'
```

Concepts avancés, les conditions : when

L'instruction when peut également:

- utiliser tous les opérateurs de comparaison : !=, <, <=, >, >=
- utiliser les booléens : True ou False
- Vérifier qu'une variable est renseignée ou pas avec l'expression
    - when: ma_variable is [not] defined

Exemple :

$ansible-playbook when-conditional.yaml

Concepts avancés, les conditions : when

```
user@jpenekusu4 ~/ansible.git (master) $ ansible-playbook when-conditional.yaml

PLAY [webservers] ***************************************************************


TASK [Gathering Facts] **********************************************************

ok: [jpenekusu6.mylabserver.com]
ok: [jpenekusu2.mylabserver.com]

TASK [Afficher la valeur de la variable fact ansible_os_family] ***************

ok: [jpenekusu2.mylabserver.com] => {
    "msg": "ansible_os_family = RedHat"
}
ok: [jpenekusu6.mylabserver.com] => {
    "msg": "ansible_os_family = RedHat"
}

TASK [Vérifier que apache est installé sur les machines RedHat et à jour] ******

changed: [jpenekusu6.mylabserver.com]
changed: [jpenekusu2.mylabserver.com]

TASK [Vérifier que apache est installé sur les machines Debian et à jour] ******

skipping: [jpenekusu2.mylabserver.com]
skipping: [jpenekusu6.mylabserver.com]

PLAY RECAP **********************************************************************

jpenekusu2.mylabserver.com : ok=3    changed=1    unreachable=0    failed=0
jpenekusu6.mylabserver.com : ok=3    changed=1    unreachable=0    failed=0
```

Concepts avancés, les conditions : when

Exemple de vérification d'une variable:

```yaml
tasks:
  - name: Vérifier que la variable obligatoire directory est renseignée
    fail:
      msg: 'La variable directory doit être renseigné'
    when: directory is not defined
  - name: Lister le contenu du répertoire directory
    command: ls -latr {{ directory }}
    when: True
```

$ansible-playbook when-conditional-testing-mandatory-variables.yaml -e "direc
tory=/" –vvv

Concepts avancés, les instructions : import_playbook

L'instruction import_playbook permet :

- de garder les playbooks de petite taille
- d'inclure un playbook réutilisable dans un autre (i.e : DRY principle)

Exemple :

```yaml
- name: Inclure le playbook local_action_plugin.yaml
  import_playbook: local_action_plugin.yaml

- name: Inclure le playbook when-conditional-testing-mandatory-variables-best-practice.yaml
  import_playbook: when-conditional-testing-mandatory-variables-best-practice.yaml directory="/home/$USER"

- name: PLAY utilisation de delegate_to
  hosts: database
  remote_user: ansible
  gather_facts: yes

- name: Inclure le playbook delegate_to.yaml sous condition
  import_playbook: delegate_to.yaml
  when: ansible_os_family == "Redhat"
```

$ansible-playbook import_playbooks.yaml

Concepts avancés, les handlers

Dans certaines situations, vous voulez exécuter des actions uniquement sous certaines conditions:

- Vous voulez redémarrer un serveur web uniquement si une nouvelle version de l'application a été installée
- Vous voulez recharger ou redémarrer un service uniquement si sa configuration a été mise à jour
- Vous pouvez utiliser l'action notify pour déclencher des taches sur un handler

Concepts avancés, les handlers

Les handlers sont des taches exécutées sous certaines conditions :

- Lorsqu'ils ont été notifiés par leur nom ou le nom du topic auquel ils sont abonnés
- Une tache a effectué une modification sur la machine distante
- Le handler est exécuté une fois quel que soit le nombre de taches qui l'ont notifié
- Dans un play, un handler est exécuté à la fin du bloc dans lequel il a été notifié (i.e : pre_tasks, tasks, roles, post_tasks)
- Il est possible d'exécuter un handler immédiatement avec l'instruction :
  - \- meta: flush_handlers

Concepts avancés, les handlers

Exemple :

```yaml
  tasks:
  - name: Vérifier que apache est installé
    yum:
      name: httpd
      state: present
  - name: Vérifier que apache est activé et démarré
    service:
      name: httpd
      state: started
      enabled: yes
    ignore_errors: true
  - name: Vérifier que la configuration du serveur apache est à jour
    template:
      src: httpd.conf.j2
      dest: /etc/httpd/conf/httpd.conf
    notify: Restart httpd

  handlers:
  - name: Restart httpd
    service:
      name: httpd
      state: restarted
```

Concepts avancés, les handlers

$ ansible-playbook handler.yaml

```
user@jpenekusu4 ~/ansible.git (master) $ ansible-playbook handler.yaml -e "http_charset=UTF-8"

PLAY [webservers] **************************************************************************


TASK [Vérifier que apache est installé] ****************************************************

ok: [jpenekusu2.mylabserver.com]
ok: [jpenekusu6.mylabserver.com]

TASK [Vérifier que apache est activé et démarré] *******************************************

ok: [jpenekusu2.mylabserver.com]
ok: [jpenekusu6.mylabserver.com]

TASK [Vérifier que la configuration du serveur apache est à jour] **************************

changed: [jpenekusu2.mylabserver.com]
changed: [jpenekusu6.mylabserver.com]

RUNNING HANDLER [Rédemarrer le serveur apache] *********************************************

changed: [jpenekusu2.mylabserver.com]
changed: [jpenekusu6.mylabserver.com]

PLAY RECAP *********************************************************************************

jpenekusu2.mylabserver.com : ok=4    changed=2    unreachable=0    failed=0
jpenekusu6.mylabserver.com : ok=4    changed=2    unreachable=0    failed=0
```

Concepts avancés, les handlers

```
user@jpenekusu4 ~/ansible.git (master) $ ansible-playbook handler.yaml

PLAY [webservers] ***************************************************************


TASK [Vérifier que apache est installé] *****************************************

ok: [jpenekusu2.mylabserver.com]
ok: [jpenekusu6.mylabserver.com]

TASK [Vérifier que apache est activé et démarré] ********************************

ok: [jpenekusu2.mylabserver.com]
ok: [jpenekusu6.mylabserver.com]

TASK [Vérifier que la configuration du serveur apache est à jour] **********

ok: [jpenekusu2.mylabserver.com]
ok: [jpenekusu6.mylabserver.com]

PLAY RECAP **********************************************************************

jpenekusu2.mylabserver.com : ok=3    changed=0    unreachable=0    failed=0
jpenekusu6.mylabserver.com : ok=3    changed=0    unreachable=0    failed=0
```

Le handler n’est pas notifié si aucun changement n’est reporté.

Concepts avancés, Gestion des erreurs : la directive ignore_errors

Si une tache échoue, Ansible exclut la machine courante de l'inventaire.
La directive ignore_errors :

- Permet d'ignorer l'erreur et de garder la machine dans l'inventaire
- L'erreur peut être sauvegardée (i.e : dans la variable register) et traitée plus tard

Exemple :

```yaml
---
- name: PLAY for error handling
  hosts: all
  remote_user: ansible

  tasks:
  - name: This will not be counted as a failure
    command: /bin/false
    ignore_errors: yes
    register: response

  - name: Display the response from
    debug:
      msg: response = {{ response }}

  - name: Fail if some characters are found in response
    fail:
      msg: response = {{ response }}
    when: "some string" in response
```

Concepts avancés, Gestion des erreurs : la directive failed_when

La directive failed_when :

- Permet de contrôler ce qui cause l'erreur

Exemple

```
---
- name: PLAY for error handling
  hosts: all
  remote_user: ansible

  tasks:
  - name: Fail when the response code equals 2
    shell: whatever
    register: response
    failed_when: response.rc == 0 or response.rc >= 2
```

Concepts avancés, Gestion des erreurs : la directive changed_when

La directive changed_when :

- Permet de surcharger le status de l'exécution de la commande précédente

Exemple :

```yaml
---
- name: PLAY to decide when a change occurs
  hosts: all
  remote_user: ansible

  tasks:
  - name: Cette commande reportera un changement si le code retour est != 2
    shell: /usr/bin/billybass --mode="take me to the river"
    register: result
    changed_when: "result.rc != 2"

  - name: Cette tache ne reportera jamais un changement
    shell: wall 'beep'
    changed_when: False
```

Concepts avancés, Gestion des erreurs : la directive any_errors_fatal

La directive any_errors_fatal :

- Des fois, on veut arrêter le play lorsqu'une erreur se produit
- Permet d'arrêter le play à la première erreur

Exemple :

```yaml
---
- name: PLAY for error handling
  hosts: all
  remote_user: ansible
  any_errors_fatal: true

  tasks:
  - name: Cette tache reportera un changement si le code retour est != 2
    shell: /usr/bin/billybass --mode="take me to the river"
    register: result
    changed_when: "result.rc != 2"
```

Concepts avancés, Gestion des erreurs : la directive max_fail_percentage

La directive max_fail_percentage :

- Permet d'arrêter le play lorsqu'un certain pourcentage du nombre des machines hôtes en erreur est atteint.

Exemple :

```yaml
---
- name: PLAY for error handling
  hosts: all
  remote_user: ansible
  max_fail_percentage: 20

  tasks:
  - name: Ce play s'arrêtera lorsque 20% des machines tomberont en erreur
    shell: /bin/false
```

Concepts avancés, Gestion des erreurs : la directive block

La directive block :

- Permet de gérer les exceptions comme dans un langage de programmation
- Block est l'équivalent de « try … catch … finally » en langage Java

Exemple :

```yaml
---
- name: PLAY for error handling whith block
  hosts: all
  remote_user: ansible

  tasks:
  - name: Attempt and gracefull roll back demo
    block:
      - debug:
          msg: 'I execute normally'
      - command: /bin/false
      - debug:
          msg: 'I never execute, due to the above task failing'
    rescue:
      - debug:
          msg: 'I caught an error'
      - command: /bin/false
      - debug:
          msg: 'I also never execute :-('
    always:
      - debug:
          msg: "this always executes"
```

Concepts avancés, Gestion des erreurs : la directive block

La directive block :

- Permet de forcer les handlers en cas d'erreur

Exemple :

```yaml
---
- name: PLAY for managing handlers in error handling
  hosts: all
  remote_user: ansible

  tasks:
   - name: Attempt and gracefull roll back demo
     block:
       - debug:
           msg: 'I execute normally'
         notify: run me even after an error
       - command: /bin/false
     rescue:
       - name: make sure all handlers run
         meta: flush_handlers
  handlers:
   - name: run me even after an error
     debug:
       msg: 'this handler runs even on error'
```

Concepts avancés, Regroupement des taches : la directive block

La directive block :

- Permet de factoriser les instructions communes à plusieurs taches

Exemple :

```yaml
---
- name: PLAY for managing handlers in error handling
  hosts: all
  remote_user: ansible
  vars:
    installs:
      - httpd
      - memcached
  tasks:
    - name: Install Apache
      block:
        - yum:
            name: "{{ installs }}"
            state: installed
        - shell: ls -l /tmp
        - service:
            name: apache
            state: started
            enabled: True
      when: ansible_distribution == 'CentOS'
      become: true
      become_user: root
```

Ansible Vault: Gestion des données secrète

Ansible vault permet de chiffrer les données secrètes:

- Mots de passe
- Clefs privées ssh
- Toute donnée sensible
- Cryptage d'un fichier ou d'une chaine de caractères

Le fichier ou la chaine de caractères crypté :

- Peut alors être distribué
- Ou stocké dans un gestionnaire de configuration

Ansible Vault: Créer un fichier crypté

Créer un fichier crypté à la volée :

$ansible-vault create fichier-encrypte.txt

```
$ ansible-vault create fichier-encrypte.txt
New Vault password:
Confirm New Vault password:
```

Ouvrier le fichier crypté dans un éditeur :

```
$ANSIBLE_VAULT;1.1;AES256
30653763323031643639663235643364333364646432386237393034653232666163303331386665
38636436626364653763346566303065343263333134636630a666561313231383431303834383730
31313132663066636237316562303438316263396162346533373434616463343633363665373538
393937303532343034 0a623934356636623266643664 64
```

Ansible Vault: crypter une donnée sécrète

Crypter le fichier donneesSecretes.yaml:

Soit le fichier donneesSecretes.yaml contenant les données suivantes :
vault_postgres-admin-password: passowrd1
vault_tomcat-admin-password: passowrd2
vault_whatever-secret-data: secret-data

```
$ ansible-vault encrypt donneesSecretes.yaml
New Vault password:
Confirm New Vault password:
Encryption successful
```

Ouvrir le fichier avec un éditeur de texte :

```
$ANSIBLE_VAULT;1.1;AES256
37363563336362343539353633532616330383538353034303033303337333063376139336364643934326238633564326233656537666231393936396630620a363765623235326439633366323537633162653435656164303433643562303231366132656665633337303537646665373230313563383063343438663966320a653737666666663323038346130633438386164613838636661353661393036386631666362366661393836333
```

Ansible Vault: Déchiffrer un fichier crypté

Déchiffrer le fichier donneesSecretes.yaml:

```
$ ansible-vault decrypt donneesSecretes.yaml
Vault password:
Decryption successful
```

Déchiffrer le fichier donneesSecretes.yaml en fournissant le fichier mot de passe:

```
$ ansible-vault decrypt donneesSecretes.yaml --vault-password-file=vault-pass-file
Decryption successful
```

Ansible Vault: Editer un fichier crypté

Editer un fichier crypté :

$ ansible-vault edit fichier-encrypte.txt

```
$ ansible-vault edit fichier-encrypte.txt
```

Le fichier sera à nouveau crypté à la fermeture de l'éditeur.

Ansible Vault: Changer le mot de passe Vault

Changer le mot de passe Vault :

```
$ ansible-vault rekey donneesSecretes.yaml
Vault password:
New Vault password:
Confirm New Vault password:
Rekey successful
```

Changer le mot de passe Vault en fournissant le fichier mot de passe :

```
$ ansible-vault rekey donneesSecretes.yaml --vault-password-file=vault-pass-file --new-vault-password-
file=new-vault-pass-file
Rekey successful
```

Ansible Vault: Changer le mot de passe Vault

- Les données sensibles ne doivent pas être affichées dans les logs :
  - Ajouter la directive no_log: True en entête du playbook
  - Exécuter le playbook
  - Observer les logs

Ansible Vault: Organisation des variables cryptées

Deux possibilités d'organisation des variables cryptées :

- Cryptage unitaire des chaines de caractères des variables secrètes
- Cryptage de la totalité du fichier des variables secrètes

Cryptage unitaire des chaines de caractères des variables secrètes

Si le nombre des variables à crypter n'est pas élevé :

- Crypter uniquement les variables avec l'option encrypt_string
- Les variables secrètes sont dans le même fichier que les variables standard
- Labelliser les variables cryptées

Exemple :

- Soit le fichier des variables en clair group_vars/production/database

```
myql_state: present
mysq_group: mysql
mysql_password: T7yu!pPPa6
mysql_user: mysql
mysql_port: 3306
```

Ansible Vault: Organisation des variables cryptées

- Crypter uniquement la variable secrète mysql_password

$ ansible-vault encrypt_string --vault-id prod@vault-pass-file 'T7yu!pPPa6' --name mysql_password

```
$ ansible-vault encrypt_string --vault-id prod@vault-pass-file 'T7yu!pPPa6' --name mysql_password
mysql_password: !vault |
          $ANSIBLE_VAULT;1.2;AES256;prod
          766888888456575757575757732396636346661346465396338326430366665306566316639393O
          074647466664646464343533663430396462626361663336087764446474744539383166626332 39
          383838663739366566663239396332653136643635623131613061666876646454545455545798 33
          376264626633666461 0a6562353661653030646234316332666633636236653732646333333131 30
          8929
Encryption successful
```

Remarquer le label prod dans la variable cryptée

Ansible Vault: Organisation des variables cryptées

- Copier coller la variable cryptée dans le fichier

```
myql_state: present
mysq_group: mysql
mysql_password: !vault |
          $ANSIBLE_VAULT;1.2;AES256;prod
          766888888456575575757575773239663634666134646539633832643036666530656631663939 30
          074647466664646464343533663430396462626361663336087764446474744539383166626332 39
          383838663739366566663239396332653136643635623131613061666876646454545455545798 33
          376264626633666461 0a6562353661653030646234316332666633636236653732646333333131 30
          8929
mysql_user: mysql
mysql_port: 3306
```

- L’appel du playbook se fait de la façon suivante :
  - $ ansible-playbook my_playbook.yaml --vault-id prod@vault-pass-file

Ansible Vault: Organisation des variables cryptées

Cryptage du fichier entier des variables secrètes

- Les variables secrètes sont externalisées dans un fichier
- Le fichier des variables secrètes est crypté en entier

Ansible best practice:

- Les variables originaux restent dans le fichier non crypté
- Les variables originaux pointent sur les variables secrètes correspondant préfixés par vault_

Exemple :

- Soit le fichier des variables en clair group_vars/production/database/vars

```
myql_state: present
mysq_group: mysql
mysql_password: "TGF867Yujjy&"
variable_secrete_x: "çidkdui__87666"
variable_secrete_y: "jhgyzefizefuhubgb$2"
variable_secrete_z: "ayshgdydyjjejej"
mysql_user: mysql
mysql_port: 3306
```

Ansible Vault: Organisation des variables cryptées

- Affecter la variable mysql_password par la variable secrète vault_mysql_password

```
myql_state: present
mysq_group: mysql
mysql_password: "{{ vault_mysql_password }}"
variable_secrete_x: "{{ vault_variable_secrete_x }}"
variable_secrete_y: "{{ vault_variable_secrete_y }}"
variable_secrete_z: "{{ vault_variable_secrete_z }}"
mysql_user: mysql
mysql_port: 3306
```

- Créer un deuxième fichier contenant uniquement les variables cryptées group_vars/production/database/secret_vars

```
vault_mysql_password: "TGF867Yujjy&"
vault_variable_secrete_x: "çidkdui__87666"
vault_variable_secrete_y: "jhgyzefizefuhubgb$2"
vault_variable_secrete_z: "ayshgdydyjjejej"
```

Ansible Vault: Organisation des variables cryptées

- Crypter le fichier group_vars/production/database/secret_vars

$ ansible-vault encrypt group_vars/production/database/secret_vars --vault-id prod@vault-pass-file

```
$ ansible-vault encrypt group_vars/production/database/secret_vars --vault-id prod@vault/vault-pass-file
Encryption successful
```

- Editer le fichier crypté group_vars/production/database/secret_vars

```
$ANSIBLE_VAULT;1.2;AES256;prod
31346364303764313064643437393439333035356263393365316661633764626438383966373038
65393033366462336666656533616162666562646435313 00a643164613162393638306637306332
66623937666137626438626564636262366431356165643637626434303235613965323630336631
32313663373565656 20a3839633363306261376264376237303365323663326661623137653338 63
…etc.
38333766646135653766383537316630393361356135333334393462326164623730623536616665
30346332313737393434626137376232623666313939373436 63
```

Remarquer le label prod dans le fichier crypté

- L’appel du playbook se fait de la façon suivante :
  - $ ansible-playbook my_playbook.yaml --vault-id prod@vault-pass-file

Ansible best practice:

- utiliser un seul mot de passe Vault par environnement !

Cas d'utilisation : Déploiement disruptif

Un déploiement disruptif interrompt momentanément le service. C'est le comportement par défaut du déploiement avec Ansible.

Dans ce TP, nous allons déployer :

- Deux serveurs web hébergeant une application java ou php
- Une base des données mysql ou postgres
- Un outil de supervision nagios
- Un load balancer
- Le déploiement d'une nouvelle version se fera avec interruption de service.
- Les alertes nagios doivent être désactivées avant l'installation et réactivée après l'installation.

Cas d'utilisation : Déploiement non disruptif

Un déploiement non disruptif (zero downtime deployment) n'interrompt pas le service.
Ce comportement est obtenu avec le mot clef serial dans l'entête du playbook.
Dans ce TP, nous allons déployer :

- Deux serveurs web hébergeant une application java ou php
- Une base des données mysql
- Un outil de supervision nagios
- Un load balancer
- Le déploiement d'une nouvelle version se fera sans interruption de service.
- Les alertes nagios doivent être désactivées avant l'installation et réactivée après l'installation.

Conclusion et perspectives

Ansible :

- Un outil simple et en évolution constante
- Peut réduire le temps de mise sur le marché des nouvelles fonctionnalités applicatives
- Un outil capital dans la transformation numérique avec DevOps.

Liste non-exhaustive de best practices Ansible:

- Ne pas surcharger les variables à plusieurs endroits au risque de se mêler les pinceaux
- Utiliser en priorité un module spécifique en lieu et place des modules command et/ou shell
- Privilégier la création d'un module custom en remplacement des scripts shell legacy
- Le code source Ansible doit être stocké dans un gestionnaire de configuration
- Utiliser uniquement les privilèges nécessaires
- Contrôler la présence des variables obligatoires avant de déclencher les traitements
- Toujours commenter/documenter les taches
- Toujours KISS (Keep It Simple, Stupid)

Appendices

- https://docs.ansible.com
- http://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html
- http://docs.ansible.com/ansible/latest/modules/list_of_all_modules.html
- http://docs.ansible.com/ansible/latest/user_guide/playbooks.html
- http://docs.ansible.com/ansible/latest/reference_appendices/faq.html
- http://docs.ansible.com/ansible/latest/reference_appendices/glossary.html
- http://jinja.pocoo.org/docs/dev/templates/
- http://www.yamllint.com