Alves Lobo Michael

- https://www.linkedin.com/in/michael-alves-lobo
- https://github.com/kairel
- devops, dev, réseau et administration système
- En poste chez Vade
- https://github.com/kairel/learning/blob/main/ANSIBLE.pdf
- Automatisation
  - ansible
  - puppet
  - terraform
  - docker

Alves Lobo Michael

Liens utils

- https://www.youtube.com/c/xavki-linux
- https://docs.ansible.com/
- https://blog.stephane-robert.info/

Planning

- 5 jours sur l’automatisation
- 2 - 2,5 jours sur Ansible
- 0,5 - 1 jour sur terraform
- 2 jours sur l’installation d’une infra via ansible (Supervision Prometheus et docker ?)

ANSIBLE

Pre-requis

- 2 VM(s) Linux : Debian
    - fqdn ansible-server & ansible-client-1
    - 2 users sur chaques VM avec au moins 1 user : “devops_user” dans le groupe sudo et le groupe devops_admin

Sur la VM ansible-server on installera ansible c’est à partir de celle-là qu’on lancera nos commandes

SOMMAIRE

Ansible

Ansible - Présentation

Comment on va procéder ?

- Pas bcp de théorie , bcp de pratique
- On va déployer des logiciels et de la conf sur chacun de vos serveurs, chacun pourra donc déployer le logiciel qu'il veut (un simple: fail2ban, iptables , une db , un haproxy, un ldap etc ...)
- On va d'abord essayer de faire des choses qui fonctionnent , et on le modifiera pour suivre les bonnes pratiques et que ce soit réutilisable et compréhensible
- on va se partager les "playbooks" et essayer de faire quelque chose de collectifs

Ansible - Présentation
1.1 définition

Définition (wikipédia)

Ansible est une plate-forme logicielle libre pour la configuration et la gestion des ordinateurs. Elle combine le déploiement de logiciels (en) multi-nœuds, l'exécution des tâches ad-hoc, et la gestion de configuration. Elle gère les différents nœuds à travers SSH et ne nécessite l'installation d'aucun logiciel supplémentaire sur ceux-ci. Les modules communiquent via la sortie standard en notation JSON et peuvent être écrits dans n'importe quel langage de programmation. Le système utilise YAML pour exprimer des descriptions réutilisables de systèmes

Crée par Michael DeHaan puis **racheté par RedHat an 2015**

Ansible - Présentation:
1.2 Mot clés

Pourquoi Ansible ?

- Automatisation
- Provisionnement (de serveurs par exemple)
- Gestion de configuration
- Orchestration
- Déploiement
- Infrastructure as a code

Ansible - Présentation
1.3 Utilisation

A quoi ça sert, des exemples ?

- Déployer des serveurs dans des clouds
- Installer un iptables sur 50 serveurs
- Mettre à jour les clés ssh de tous les serveurs web
- Installer postgresql , créer les user , la database
- Déploiement d'une pile applicative complète (rails/nginx/db)
- Déployer des Firewall virtuels dans les différents Datacenter

Ansible - Présentation
1.4 Les concepts

- Configuration par défaut se trouve dans /etc/ansible/ansible.cfg
- Le fichier hosts et les playbooks
- les variables
- les rôles

Ansible - Présentation
1.4 Les concepts

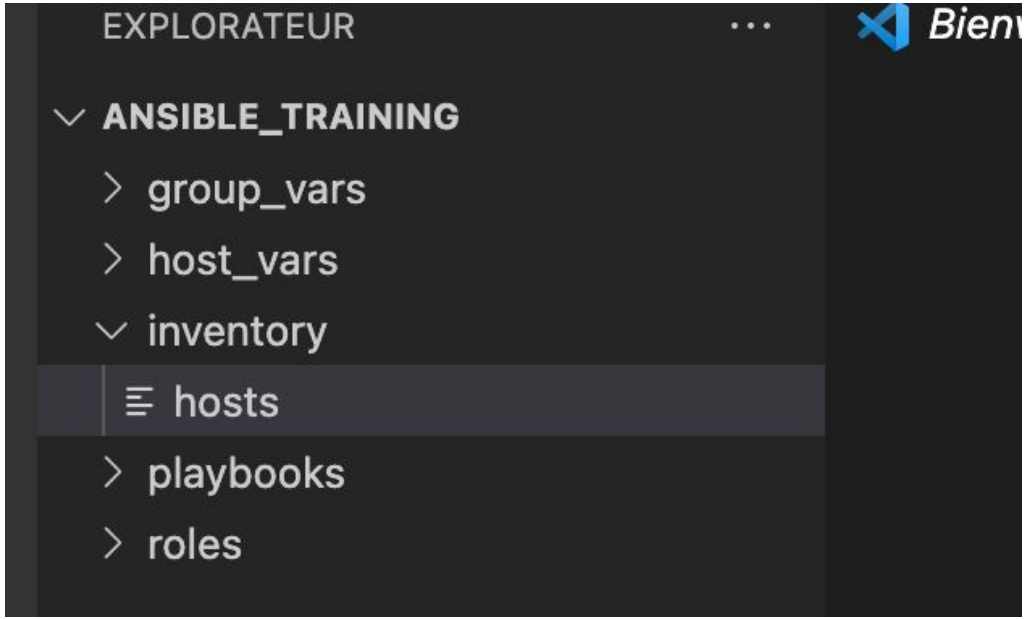
Le fichier de configuration

```
# Example config file for ansible -- https://ansible.com/
# ===================================================

# Nearly all parameters can be overridden in ansible-playbook
# or with command line flags. Ansible will read ANSIBLE_CONFIG,
# ansible.cfg in the current working directory, .ansible.cfg in
# the home directory, or /etc/ansible/ansible.cfg, whichever it
# finds first

# For a full list of available options, run ansible-config list or see the
# documentation: https://docs.ansible.com/ansible/latest/reference_appendices/config.html.

[defaults]
#inventory       = /Users/malveslobo/c2b_devops/ansible/
#library         = ~/.ansible/plugins/modules:/usr/share/ansible/plugins/modules
#module_utils    = ~/.ansible/plugins/module_utils:/usr/share/ansible/plugins/module_utils
#remote_tmp      = ~/.ansible/tmp
#local_tmp       = ~/.ansible/tmp
#forks           = 5
#poll_interval   = 0.001
#ask_pass        = False
#transport       = smart
inventory        = /Users/malveslobo/c2b_devops/ansible/inventory/
roles_path        = /Users/malveslobo/c2b_devops/ansible/roles
host_key_checking = False
# Plays will gather facts by default, which contain information about
# the remote system.
#
# smart - gather by default, but don't regather if already gathered
# implicit - gather by default, turn off with gather_facts: False
```

Ansible - Présentation
1.4 Les concepts

La structure

- On peut modifier la structure
- Le fichier hosts et les playbooks
- les variables
- les rôles



Ansible - Installation
2.1 Installation

As of Ansible 4.0.0, new releases will only be generated for Ubuntu 18.04 (Bionic) or later releases.

Add the following line to `/etc/apt/sources.list` or `/etc/apt/sources.list.d/ansible.list` :

```
deb http://ppa.launchpad.net/ansible/ansible/ubuntu MATCHING_UBUNTU_CODENAME_HERE main
```

Example for Debian 11 (Bullseye)

```
deb http://ppa.launchpad.net/ansible/ansible/ubuntu focal main
```

Then run these commands:

```
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 93C4A3FD7BB9C367
$ sudo apt update
$ sudo apt install ansible
```

Ansible - Installation
2.1 Installation

```
deb http://ppa.launchpad.net/ansible/ansible/ubuntu focal main
```

```
sudo apt-key adv -keyserver keyserver.ubuntu.com -recv-keys 93C4A3FD7BB9C367
sudo apt update
sudo apt install ansible
```

Ansible - Installation
2.1 Check de l'installation

Une fois l'installation terminée sur le serveur ansible-server on va tester avec la commande

```
ansible -version
```

```
➜  ~ ansible --version
ansible [core 2.12.1]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/Users/malveslobo/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/local/Cellar/ansible/5.1.0/libexec/lib/python3.10/site-packages/ansible
  ansible collection location = /Users/malveslobo/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/local/bin/ansible
  python version = 3.10.1 (main, Dec  6 2021, 23:20:29) [Clang 13.0.0 (clang-1300.0.29.3)]
  jinja version = 3.0.3
  libyaml = True
```

Ansible - Installation

A vous de jouer:

- installe vos 2 vms debians avec un user devops_user dans le groupe devops_admin
- Installer ansible
- checker les infos

Ansible - Installation
2.2 Les différents répertoire d'installation

- config file: le fichier de configuration de ansible
- configured module search path: Répertoire ou se trouve les plugins pour ansible
- ansible python module location: Répertoire ou se trouve les plugins python utile à ansible
- ansible collection location: Répertoire on se trouve les collections de ansible: les installations se font avec ansible-galaxy

Ansible - Installation
2.2 Les différents répertoires d'installations

- exécutable location: Emplacement du binaire/exécutable
- python version: Version de python utilisée(défini la version de ansible)
- jinja version: Version du moteur de template utilisé
- libyaml: Utilisation de la librairie yaml

Ansible - Installation
2.3 La commande ansible

- ansible –help

```
options:
  --ask-vault-password, --ask-vault-pass
                        ask for vault password
  --become-password-file BECOME_PASSWORD_FILE, --become-pass-file BECOME_PASSWORD_FILE
                        Become password file
  --connection-password-file CONNECTION_PASSWORD_FILE, --conn-pass-file CONNECTION_PASSWORD_FILE
                        Connection password file
  --list-hosts          outputs a list of matching hosts; does not execute anything else
  --playbook-dir BASEDIR
                        Since this tool does not use playbooks, use this as a substitute playbook directory.This sets th
  --syntax-check        perform a syntax check on the playbook, but do not execute it
  --task-timeout TASK_TIMEOUT
                        set task timeout limit in seconds, must be positive integer.
  --vault-id VAULT_IDS  the vault identity to use
  --vault-password-file VAULT_PASSWORD_FILES, --vault-pass-file VAULT_PASSWORD_FILES
                        vault password file
  --version             show program's version number, config file location, configured module search path, module locat
  -B SECONDS, --background SECONDS
                        run asynchronously, failing after X seconds (default=N/A)
  -C, --check           don't make any changes; instead, try to predict some of the changes that may occur
  -D, --diff            when changing (small) files and templates, show the differences in those files; works great with
  -K, --ask-become-pass
                        ask for privilege escalation password
  -M MODULE_PATH, --module-path MODULE_PATH
                        prepend colon-separated path(s) to module library (default=~/.ansible/plugins/modules:/usr/share
```

Ansible - Installation
2.3 La commande ansible

- –list-hosts: renvoie la liste des tous les hosts
- –check : essaie de prédire si le playbook est fonctionnel
- –diff : Montre les différences entre les fichiers sources et destination
- –ask-pass: Demande le password de connection pour chaque serveur et pour chaque playbook

Ansible - Installation
2.4 Les autres exécutables

- ansible-config: Liste la config
- ansible-playbook: Le coeur d'Ansible: permet de lancer les playbooks
- ansible-galaxy: Permet d'installer des collections(équivalent d'un apt install)
- ansible-connection, ansible-console, ansible-doc etc ...., tester les si vous voulez

Ansible - Configuration
3.1 Le fichier ansible.cfg

- Il peut se trouve à plusieurs endroits selon les distributions et les versions. Comme pour tout dans ansible, il y a un ordre de recherche des configurations, la première configuration qui est trouvée gagne
  - ANSIBLE_CONFIGS(variables d'environnement)
  - ansible.cfg (répertoire courant)
  - ~/.ansible.cfg
  - /etc/ansible/ansible.cfg

Ansible - Configuration
3.1 Le fichier ansible.cfg

- inventory: Emplacements du fichier ou des plugins nécessaire pour lister les hosts
- ask-pass: Demande un password par défaut
- roles_path: Emplacements des rôles
- remote_user: user de connection par défaut
- jinja_extensions: Liste des extensions valides pour le moteur de template jinja

Ansible - Configuration
3.1 Le fichier ansible.cfg

- private_key_file: Emplacement du fichier de clé privée
- no_log: Log des tâches, désactivé par défaut
- module_compression: Méthode compression d'envoi des fichiers/modules
- connection_timeout: Temps de connexion max par session
- host_key_checking: Chek la validité de la clé ssh de connexion

Ansible - Configuration
3.2 Les autres méthodes de config

- La commande ansible-config
- Dans les playbooks eux-mêmes

Ansible - Configuration

A vous de jouer

- Sur le serveur ansible-serveur créer tous les fichiers de configuration listé plus haut si il n'existe pas et surcharger la configuration suivante: "host_key_checking" et regarder ce que ça donne avec la commande suivante

```
ansible-config view | grep host_key_checking
```

Ansible - Les Hosts
4.1 Définitions

Les hosts sont le coeur de ansible , ce sont les serveurs sur lesquels on va déployer/installer nos éléments d'infrastructure (logiciel)

Ansible fonctionne sur de multiples systèmes en même temps, ces systèmes sont regroupés dans ce qu'on appel un inventaire, cet inventaire est constitué de hosts et de groupe de hosts

Ansible - Les Hosts
4.1 Définitions

L’inventaire des hosts se trouvent pas défaut dans le fichier /etc/ansible/hosts mais il est tout à fait possible de le changer d’emplacement

Ce fichier est formaté de la même façon qu’un fichier “ini”

```
mail.example.com

[webservers]
foo.example.com
bar.example.com

[dbservers]
one.example.com
two.example.com
three.example.com
```

Ansible - Les Hosts
4.1 Configuration

Dans ce fichier il est donc possible d'ajouter des hosts

```
web-front-1.domain.com
web-db-master.domain.com
```

Mais aussi des groups de hosts (par défaut 2 groupes existes: all, ungrouped)

```
[WEBSERVERS]
web-front-1.domain.com
web-db-master.domain.com
```

Ansible - Les Hosts
4.1 Configuration

Il est également possible d’utiliser des patterns

```
[webservers]
web-front[01-50].domain.com

[databases]
db-[a:f].domain.com
```

Ansible - Les Hosts
4.1 Configuration

Chaque host déclaré peut avoir des paramètres qui lui sont propre

web-front-1.domain.com ansible_port=55555 ansible_user=devops_user

web-front-2.domain.com ansible_python_interpreter=/usr/bin/python3

Ansible - Les Hosts
4.1 Configuration

On peut assigner directement des variables

par host:

- web-front-1.domain.com group_user=sudo

par groupe d’host

```
[webservers]
web-front-[01-10].domain.com
[webservers.vars]
group_user=sudo
```

Ansible - Les Hosts
4.1 Configuration

Liste utile des paramètres des hosts:

- ansible_user: l'utilisateur par défaut de connection
- ansible_ssh_private_key: la clé privée à utiliser
- ansible_become: permet de changer d'utilisateur (en sudo par exemple)
- ansible_python_interpreter: l'interpréteur python à utilise: par défaut sur certains serveurs c'est python d'autres python3

Ansible - Les Hosts
4.1 Configuration

Il est également possible de ne pas utiliser ssh

- local: deploy le playbook sur la machine en local
- docker: deploy sur docker en utilisant le client docker
- winrm: pour windows

Ansible - Les Hosts
4.1 Configuration

- Le fichier peut ne pas s’appeler hosts, on peut le nommer comme on veut , on peut même en avoir plusieurs.
  - il suffit de le modifier dans le fichier ansible.cfg
    - `- inventory: /etc/ansible/new_file_host.cfg`

Ansible - Les Hosts
4.1 Configuration

- Via la commande ansible-playbook que l'on verra plus tard on peut appeler des fichiers d'inventaire, ce qui permet d'isoler par exemple les environnements
    - `- ansible-playbook -i production`

- Ou production est un fichier d'inventaire qui ne contient que des hosts de vos environnements de production

Ansible - Les Hosts
4.1 Configuration

- Il est également possible d'avoir un inventaire dynamique, notamment dans le cas des providers de cloud
- https://docs.ansible.com/ansible/latest/collections/openstack/cloud/openstack_inventory.html
- https://docs.ansible.com/ansible/latest/collections/community/vmware/vmware_vm_inventory_inventory.html
-

Ansible - Les Hosts
4.1 Configuration

A vous de jouer

- créer le fichier hosts si il n'existe dans un nouveau répertoire
- configurer ansible pour qu'il utilise ce fichier
- modifier ce fichier en ajoutant des hosts qui existent (ceux des autres ou juste votre serveur local)
- Lister l'inventaire (ansible-inventory --list)
- Bonus: creer plusieurs fichiers hosts et lancer lister les hosts avec la commande ansible-inventory

Ansible - Les Hosts
4.2 Connection

Maintenant que l'on sait définir des hosts et donc construire un inventaire , on va voir comment se connecter sur ces hosts

Avant de commencer et pour tester que tout est ok lancer cette commande:

```
ansible localhost -m ping
```

faites la même chose pour un host en particulier

```
ansible host-que-vous-voulez -m ping
```

Comment vous interprétez le résultat ?

Ansible - Les Hosts
4.2 Connection

Pour se connecter à un host (valable sur ssh et winrm) il y a plusieurs façon de faire

- par credentials (user et password)
- par clé

On va utiliser la connection par clé qui est la plus sécurisée

Ansible - Les Hosts
4.2 Connection

Par défaut lorsque vous déployez un serveur vous pouvez le déployer avec une clé autorisée par défaut

Il faut donc ajouter dans le fichier “authorized_keys” de l’utilisateur que vous utilisez pour vous connecter votre clé publique, ce qui implique donc une 1ère étape de configuration des serveurs pour ansible

Il est quand même à noter que contrairement à d’autres outils comme puppet ou chef , ansible nécessite pas l’installation d’agents , il n’est pas intrusif

Ansible - Les Hosts
4.2 Connection

Par défaut on utilise la connexion en ssh mais il est possible d'utiliser d'autres types de connections

- On peut définir les infos de connections a 2 endroits:
    - dans le fichier de configuration ansible.cfg
    - dans le ou les fichiers d'inventaires

Ansible - Les Hosts
4.2 Connection

- Dans le fichier ansible.cfg
- - **remote_user**: devops_user
- - **private_key_file**: id_rsa

Ansible - Les Hosts
4.2 Connection

- Dans le fichier d‘inventaire
- `- ansible_user: debian`
- `- ansible_ssh_private_key_file: id_rsa`

Ansible - Les Hosts
4.2 Connection

A vous de jouer : sur votre serveur

- Créer au moins 2 users ansible(sudo si possible)
- générer une paire de clé ssh pour ansible pour chaucun des 2 users
- donner la clé publique générée à une ou plusieurs personnes
- ajouter cette clé publique au fichier authorized_keys à l'utilisateur de votre serveur qui servira à ansible
- tester que vous accéder bien aux serveurs (ansible -i production all -m shell -a "ls")
- surcharger les infos de connections a un serveur dans le fichier d'inventaire

Ansible - Command ad-hoc
5.1 Définitions

Une commande ad-hoc c'est un langage d'orchestration de Ansible.

On peut utiliser ansible de 2 façons différentes

- via des commandes ad-hoc
- via des playbook que l'on verra juste après

Ansible - Command ad-hoc
5.1 Utilisations

Une commande ad-hoc permet d'automatiser une tâche de manière simple et rapide mais non réutilisables , ça permet de montrer rapidement la simplicité d'ansible et de ces concepts

Pour cela on utilise directement le binaire ansible

```
$ ansible [pattern] -m [module] -a "[module options]"
```

Ansible - Command ad-hoc
5.1 Utilisations

On utilise en règle générale les commandes ad-hoc pour des tâches uniques non réutilisables comme par exemple le reboot rapide de plusieurs serveurs

On va donc faire le test de suite :

- connectez vous à votre VM ansible-server et écrivez une commande ad-hoc qui écrit

“Hello ansible” dans un fichier texte dans le répertoire tmp du serveur ansible-client-1

Ansible - Playbooks
6.1 Définitions

Un playbook c'est un langage d'orchestration de Ansible.

Il permet de d'ordonnancer le déploiement de configuration ou de logiciel sur de multiple serveurs

Comme pour la plupart des éléments d'Ansible il utilise le format YAML

```yaml
---
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum:
      name: httpd
      state: latest
  - name: write the apache config file
    template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running
    service:
      name: httpd
      state: started
  handlers:
    - name: restart apache
      service:
        name: httpd
        state: restarted
```

Ansible - Playbooks
6.1 Définitions

Les playbooks peuvent contenir des tâches à exécuter sur de multiples serveurs

```
- hosts: webservers
  remote_user: root

  tasks:
  - name: ensure apache is at the latest versio
    yum:
      name: httpd
      state: latest
  - name: write the apache config file
    template:
      src: /srv/httpd.j2
      dest: /etc/httpd.conf

- hosts: databases
  remote_user: root

  tasks:
  - name: ensure postgresql is at the latest ve
    yum:
      name: postgresql
      state: latest
  - name: ensure that postgresql is started
    service:
      name: postgresql
      state: started
```

Ansible - Playbooks
6.2 Hosts

Un playbook se lance sur des hosts , il est donc possible de les définir dans un playbook.

Un host se trouve toujours en premier dans un playbook

```
---
- hosts: webservers
  remote_user: root
  tasks:
    - name: test connection
      ping:
      remote_user: yourname
```

Ansible - Playbooks
6.2 Hosts

Les Hosts sont ceux repris dans le fichier hosts , vous pouvez donc préciser un host ou un groupe de hosts

```
---
- hosts: webservers
  remote_user: root
  tasks:
    - name: test connection
      ping:
      remote_user: yourname
```

Ansible - Playbooks
6.3 Paramètres

On peut aussi définir les paramètres de connection dans le playbook comme le remote_user

```
---
- hosts: webservers
  remote_user: root
  tasks:
    - name: test connection
      ping:
      remote_user: yourname
```

Ansible - Playbooks
6.4 Les tasks

Les tasks sont des tâches exécutées par le playbook, elles sont lancées les unes à la suite des autres

```
---
- hosts: kairel
  vars:
    update_keys: true
  tasks:
    - name: Info debug
      ansible.builtin.debug:
        msg: "Update ssh keys"
    - name: Include ssh update keys
      include_role:
        name: common
```

Ansible - Playbooks
6.4 Les tasks

Chaque tâche possède des attributs

- name: une description de la tâche
- un nom de module: le module exécuté pour la tâche
- les attributs obligatoires pour un module
- des attributs optionnels à la tâche

Ansible - Playbooks
6.4 Les tasks

Les modules

Par défaut le “core” d’ansible est livré avec certains modules mais vous pouvez en ajouter d’autres notamment avec ansible-galaxy

Exemple ici avec le module file

```
- name: find old log file
  ansible.builtin.find:
    paths: /var/log/hadoop/hdfs/
    patterns: '*.log.*'
    age: 5d
  register: files_to_delete
- name: remove older than 10
  ansible.builtin.file:
    path: "{{ item.path }}"
    state: absent
  with_items: "{{ files_to_delete.files }}"
```

Ansible - Playbooks
6.4 Les tasks

Les modules

Par défaut le “core” d’ansible est livré avec certains modules mais vous pouvez en ajouter d’autres notamment avec ansible-galaxy

Exemple ici avec le module file

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

Ansible - Playbooks
6.4 Les tasks

Les modules les plus utilisés:

- file
- template
- service/systemd
- command
- apt
- copy

Ansible - Playbooks
6.4 Les tasks

Le module File

```yaml
- name: Create log directory
  ansible.builtin.file:
    path: '{{ log_directory }}/{{ applicat
    state: directory
    mode: '0755'
    owner: '{{ user }}'
    group: '{{ group }}'
  become: yes

- name: Create a log symbolic link
```

Ansible - Playbooks
6.4 Les tasks

Le module Template

```
- name: Create database file
  template:
    src: '{{ templates_path }}
    dest: '/home/{{ user }}/{
    owner: '{{ user }}'
    group: '{{ group }}'
  become: yes
```

Ansible - Playbooks
6.4 Les tasks

Le module service/systemd

```yaml
name: Add service to list servic
ansible.builtin.systemd:
  enabled: yes
  name: click2find-app.service
  daemon_reload : yes
become: yes
```

Ansible - Playbooks
6.4 Les tasks

Le module command

```
- name: Import rvm  gpg key
  command:
    cmd:  '{{ gpg_rvm_install_cmd
  become: yes
  become_user: '{{ user }}'
```

Ansible - Playbooks
6.4 Les tasks

Le module apt

```yaml
- name: Install curl , gpg, nginx, posgresql
  apt:
    pkg:
      - curl
      - gpg
      - nginx
      - postgresql
      - python-setuptools
      - python-virtualenv
      - python-psycopg2
      - python-ipaddress
      - gawk
      - autoconf
      - automake
      - bison
      - pkg-config
      - sqlite
      - git
      - libpq-dev
```

Ansible - Playbooks
6.4 Les tasks

Le module copy

```
- name: copy rvm install script
  copy:
    src: '{{ templates_path }}/shared/rvm_i
    dest: '{{ dest_rvm_install }}'
    mode: '0777'
  become: yes
  become_user: '{{ user }}'
```

Ansible - Playbooks
6.4 Les tasks

Les attributs optionnels/keyword

- become (escalation)
- when (conditionnal)
- ignore_errors(errors)
- debug(output)

https://docs.ansible.com/ansible/latest/reference_appendices/playbooks_keywords.html#playbook-keywords

Ansible - Playbooks
6.4 Les tasks

Les attributs optionnels/ Become

Permet de devenir un autre utilisateur dans l'exécution de la tâche

- become: yes/no
- become_user: root
- become_method : sudo/su

Ansible - Playbooks
6.4 Les tasks

Les attributs optionnels/ Ignore error

Ansible lance les tâches les unes après les autres, si une des tâches donne une erreur le playbook s'arrête

l'attribut ignore error permet de passer à la tâche suivante même en cas d'erreur

- ignore_errors: yes/no
- ignore_unreachable: yes/no

Ansible - Playbooks
6.4 Les tasks

Les attributs optionnels/ when

Il est possible d'exécuter des tâches selon des conditions:

when: Check if la tâche soit être lancer ou pas

until: “même” que when mais dans une boucle

failed_when: condition inverse de “when”

```
- name: run docker-compose
  command:
    cmd: 'docker-compose -f {{ docker_compo
  ignore_errors: yes
  when: start_container is defined
```

Ansible - Playbooks
6.5 Un exemple

On va créer un fichier dans le répertoire tmp avec du texte dedans: avant de commencer

- vérifier que vous avez bien accès au host sur lequel vous allez exécuter le playbook
- ouvrir votre éditeur préféré

Créer un fichier first_playbook.yml ou vous le voulez

Ansible - Playbooks
6.5 Un exemple

```yaml
- hosts: michael

  tasks:

    - name: Create first file
      file:
        path: /tmp/first_playbook.txt
        state: touch
```

Ansible - Playbooks
6.5 Un exemple

Lancer la commande suivante:

ansible-playbook first_playbook.yml –check

Le check comme déjà vu aura pour effet de vérifier que la syntax et que l'exécution est ok mais n'exécute pas le playbook

si tout est ok lancer la commande suivante (ne pas lancer le playbook sur le même serveur qu' autre)

ansible-playbook first_playbook.yml

Ansible - Playbooks

A vous de jouer: créer votre premier playbook

- créer un playbook qui installe curl sur les serveurs
- créer un playbook qui :
  - créer un répertoire dans /opt (votre prénom sans espace ni caractère spéciaux)
  - créer un fichier avec le texte “Hello ansible” dans ce répertoire

Ansible - Variables
6.1 Définitions

Les variables permettent d'utiliser ansible sur différents OS

Les variables permettent donc à ansible d'être plus souple , de factoriser ses playbooks, et de déployer avec une seule commande sur plein de système différents

Il est possible de définir des variables un peu partout dans ansible aussi bien dans le fichier host que dans un playbook, un répertoire dédié et directement via la ligne de commande.

Comme pour la configuration un système d'ordre est mis en place

Ansible - Variables
6.1 Définitions

Une variable doit avoir une syntaxe valide

elle ne doit pas:

- comporter des caractères tel que *,/,\ etc ... (uniquement _ accepté)
- ne pas utiliser de mots clé ansible comme : async, environment, task etc ...

Ansible - Variables
6.1 Définitions

Type de variables:

- simple: clé:valeur

```
remote_install_path: /opt/my_app_config
```

Ansible - Variables
6.1 Définitions

Type de variables:

- list: tableau en yaml

You can define variables with multiple values

```
region:
  - northeast
  - southeast
  - midwest
```

Referencing list variables

When you use variables defined as a list (also
example:

```
region: "{{ region[0] }}"
```

Ansible - Variables
6.1 Définitions

Type de variables:

- dictionary: hash en yaml

```yaml
foo:
  field1: one
  field2: two
```

Referencing key:value dictionary

When you use variables defined as a key:valu

```
foo['field1']
foo.field1
```

Ansible - Variables
6.1 Définitions

- Les variables pour être interprété dans une chaine de caractère doivent être encadrer par 2 accolades

```
- hosts: app_servers
  vars:
       app_path: "{{ base_path }}/22"
```

Ansible - Variables
6.1 Définitions

- Les variables pour être interprété dans une chaine de caractère doivent être encadrer par 2 accolades

```
- hosts: app_servers
  vars:
       app_path: "{{ base_path }}/22"
```

Ansible - Variables
6.2 Utilisation

- Les variables peuvent être créés de manière statique comme vu précédemment mais elles peuvent être dynamique , pour cela on utilise une register

```yaml
- hosts: web_servers

  tasks:

     - name: Run a shell command and register its output as a variable
       ansible.builtin.shell: /usr/bin/foo
       register: foo_result
       ignore_errors: true

     - name: Run a shell command using output of the previous task
       ansible.builtin.shell: /usr/bin/bar
       when: foo_result.rc == 5
```

Ansible - Variables
6.2 Utilisation

L'endroit est déclaré la variable est très important ,

c'est lui qui définit si la variable sera prise en compte ou si elle sera surchargée. Ansible regardera par exemple d'abord dans les valeurs de la ligne de commande, mais si on mets la même variable dans le playbook,

c'est la valeur de cette variable qui sera pris en compte

1. command line values (for example, `-u my_user`, these are not variable
2. role defaults (defined in role/defaults/main.yml) [1]
3. inventory file or script group vars [2]
4. inventory group_vars/all [3]
5. playbook group_vars/all [3]
6. inventory group_vars/* [3]
7. playbook group_vars/* [3]
8. inventory file or script host vars [2]
9. inventory host_vars/* [3]
10. playbook host_vars/* [3]
11. host facts / cached set_facts [4]
12. play vars
13. play vars_prompt
14. play vars_files
15. role vars (defined in role/vars/main.yml)
16. block vars (only for tasks in block)
17. task vars (only for the task)
18. include_vars
19. set_facts / registered vars
20. role (and include_role) params
21. include params
22. extra vars (for example, `-e "user=my_user"`)(always win precedence)

Ansible - Variables
6.2 Utilisation

Par exemple si on considère la variable

memory

Si on la définit comme suit

- ansible-playbook -u memory 500
- dans le playbook avec var = 1000
- ansible-playbook –extra-vars “memory=1500”

c’est la dernière valeur trouvée qui sera prise en compte donc

1500.

1. command line values (for example, `-u my_user`, these are not variable
2. role defaults (defined in role/defaults/main.yml) [1]
3. inventory file or script group vars [2]
4. inventory group_vars/all [3]
5. playbook group_vars/all [3]
6. inventory group_vars/* [3]
7. playbook group_vars/* [3]
8. inventory file or script host vars [2]
9. inventory host_vars/* [3]
10. playbook host_vars/* [3]
11. host facts / cached set_facts [4]
12. play vars
13. play vars_prompt
14. play vars_files
15. role vars (defined in role/vars/main.yml)
16. block vars (only for tasks in block)
17. task vars (only for the task)
18. include_vars
19. set_facts / registered vars
20. role (and include_role) params
21. include params
22. extra vars (for example, `-e "user=my_user"`)(always win precedence)

Ansible - Variables
6.2 Utilisation

Cela va nous permettre de mettre des

valeurs par défaut et de venir les surcharger

pour certains hosts

1. command line values (for example, `-u my_user`, these are not variable
2. role defaults (defined in role/defaults/main.yml) [1]
3. inventory file or script group vars [2]
4. inventory group_vars/all [3]
5. playbook group_vars/all [3]
6. inventory group_vars/* [3]
7. playbook group_vars/* [3]
8. inventory file or script host vars [2]
9. inventory host_vars/* [3]
10. playbook host_vars/* [3]
11. host facts / cached set_facts [4]
12. play vars
13. play vars_prompt
14. play vars_files
15. role vars (defined in role/vars/main.yml)
16. block vars (only for tasks in block)
17. task vars (only for the task)
18. include_vars
19. set_facts / registered vars
20. role (and include_role) params
21. include params
22. extra vars (for example, `-e "user=my_user"`)(always win precedence)

Ansible - Variables
6.3 Exemple

on veut définir la variable mémory pour tous les serveurs web et venir la surcharger uniquement pour un déploiement

On vient donc ajouter la variable dans le playbook

```
- hosts: prod-uk1-scrapping-1.server.clic2buy.com

  vars:
    memory: 4000
```

Ansible - Variables
6.3 Exemple

On vient la surcharger au niveau de l'appel de ansible-playbook

```
~ ansible-playbook first_playbook.yml --extra-vars "memory=8000"
```

Ici il viendra donc prendre la valeur memory = 8000 car extra-vars est celui qui sera pris en compte

Ansible - Variables

A vous de jouer:

reprenez votre playbook first playbook first_playbook.yml et:

- ajouter une tâche qui permet de creer un fichier
- mettre le nom du fichier dans une variable file_name au niveau du playbook et tester
- lancer votre playbook avec une extra-var file_name et tester

Ansible - Templates
7.1 Définition

Les templates permettent de générer des fichiers dynamiquement , vous pouvez créer des documents dans lesquels une partie du contenu peut être remplacé par des variables

On peut donc générer des fichiers de conf, des documents, des fichiers d'initialisations etc

Ansible - Templates
7.1 Définition

Ansible utilise par défaut le moteur de template jinja2

Jinja (moteur de template)

Pour les articles homonymes, voir Jinja.

Cet article est une ébauche concernant un logiciel libre.
Vous pouvez partager vos connaissances en l'améliorant (**comment**
correspondants.

Jinja est un moteur de templates pour le langage de programmation Python.

Il aurait inspiré Twig, moteur de template PHP[2].

Exemple [modifier | modifier le code]

```
<!DOCTYPE html>
<html>
  <head>
    <title>{{ variable|escape }}</title>
  </head>
  <body>
  {%- for item in item_list %}
    {{ item }}{% if not loop.last %},{% endif %}
  {%- endfor %}
  </body>
</html>
```

Notes et références [modifier | modifier le code]

Ansible - Templates
7.2 Utilisation

Les modèles **Jinja2** sont de simples fichiers textes **contenant des variables** qui sont **remplacées par les valeurs définies** lors de l'éxecution du **module template d'Ansible**. Ces données peuvent être celles définies dans des **inventaires** Ansible, dans des **variables** ou bien encore **collectées** lors de l'exécution du playbook.

Les fichiers de templates jinja2 utilisent, en plus des variables et expressions, des balises qui permettent de contrôler la logique du document. La syntaxe utilise des délimiteurs qui sont du type suivant:

- {{ ... }}pour les variables, comme dans les playbooks
- {% ... %}pour les instructions de controle: if, for, raw, block, ...
- {# ... #} pour les commentaires qui ne seront pas inclus le fichier résultat
- # ... ## pour les instructions de ligne

Ansible - Templates
7.2 Utilisation

Les filtres , par exemple pour définir une valeur par défaut

```
{{ variable | filtre }}
```

```
switchport access vlan: {{ interface.vlan | default('8') }}
```

Ansible - Templates
7.2 Utilisation

Les conditions

```
<div>
    {% if True %}
        yay
    {% endif %}
</div>
```

Ansible - Templates
7.2 Utilisation

Les tests (pour contrôler si une variable existe par exemple)

```
{% if variable is defined %}
    value of variable: {{ variable }}
{% else %}
    variable is not defined
{% endif %}
```

Ansible - Templates
7.2 Utilisation

Les boucles

```
% for user in users %}
    {%- if loop.index is even %}{% continue %}{% endif %}
    ...
{% endfor %}
```

Ansible - Templates
7.2 Utilisation

Ou se trouve le répertoire de template ?

Ansible suit cette structure pour chercher ou se trouve les templates

le répertoire de template se trouve à la racine de vos playbooks si vous lancez votre playbook depuis le répertoire /home/user/ansible/first_playbook.yml , le répertoire template (à créer) se trouve dans /home/user/ansible/templates

Vous pouvez également utiliser

- une variable spécifique dans le paramètre extra-vars

Ansible - Templates
7.2 Utilisation

- Les templates ansible sont en fait un module (appelé template) qui se comporte donc comme un module ansible. Il possède une liste d'attributs

```yaml
- name: Template a file to /etc/file.conf
  ansible.builtin.template:
    src: /mytemplates/foo.j2
    dest: /etc/file.conf
    owner: bin
    group: wheel
    mode: '0644'

- name: Template a file, using symbolic modes (equivalent to 0644)
  ansible.builtin.template:
    src: /mytemplates/foo.j2
    dest: /etc/file.conf
    owner: bin
    group: wheel
    mode: u=rw,g=r,o=r
```

Ansible - Templates
7.2 Utilisation

```yaml
- name: Copy a version of named.conf that is dependent on the OS. setype o
  ansible.builtin.template:
    src: named.conf_{{ ansible_os_family }}.j2
    dest: /etc/named.conf
    group: named
    setype: named_conf_t
    mode: 0640

- name: Create a DOS-style text file from a template
  ansible.builtin.template:
    src: config.ini.j2
    dest: /share/windows/config.ini
    newline_sequence: '\r\n'

- name: Copy a new sudoers file into place, after passing validation with
  ansible.builtin.template:
    src: /mine/sudoers
    dest: /etc/sudoers
    validate: /usr/sbin/visudo -cf %s

- name: Update sshd configuration safely, avoid locking yourself out
  ansible.builtin.template:
    src: etc/ssh/sshd_config.j2
    dest: /etc/ssh/sshd_config
    owner: root
    group: root
    mode: '0600'
    validate: /usr/sbin/sshd -t -f %s
    backup: yes
```

Ansible - Templates
7.3 Example

On va maintenant créer notre second playbook second_playbook.yml

Ce playbook va nous permettre de déployer le même fichier texte que tout à l'heure mais avec un template

On va donc définir une variable text_to_display que l'on va afficher dans le fichier texte généré

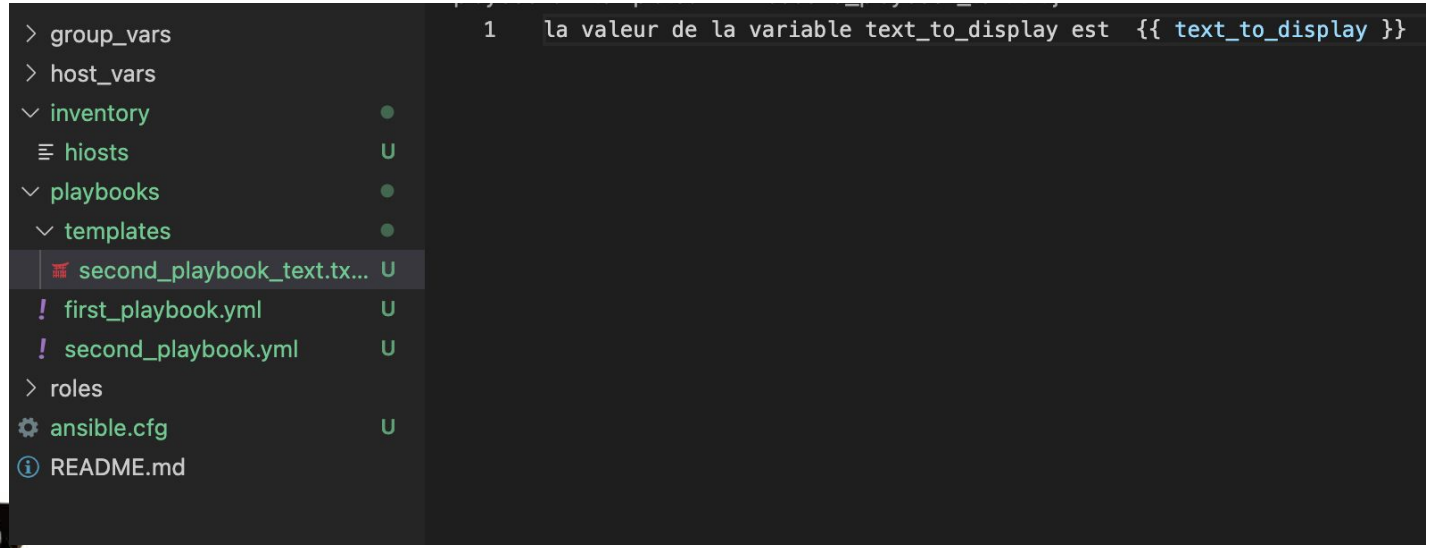
Ansible - Templates
7.3 Example

créer le playbook second_playbook.yml

```yaml
---

- hosts: michael

  tasks:

    - name: Create first file
      file:
        path: /tmp/first_playbook.txt
        state: touch

    - name: Create file with template
      template:
        src: 'second_playbook_text.txt.j2'
        dest: /tmp/second_playbook.txt
```

Ansible - Templates
7.3 Example

créer le fichier de template second_playbook_text.txt.j2 dans le répertoire template



Ansible - Templates
7.3 Example

lancer le playbook

ansible-playbook playbooks/second_playbook.yml --extra-vars "text_to_display=toto"

Ansible - Templates
7.3 Example

Vérifier que le fichier est bien créé et qu'il affiche bien votre variable

ansible-playbook playbooks/second_playbook.yml --extra-vars "text_to_display=toto"

```
training git:(main) ✗
training git:(main) ✗ cat /tmp/second_playbook.txt
valeur de la variable text_to_display est  toto%
training git:(main) ✗
training git:(main) ✗
training git:(main) ✗
training git:(main) ✗
training git:(main) ✗
training git:(main) ✗
```

Ansible - Pratiques
7.4 Projet

A vous de jouer

Une faille de sécurité vient d'être découverte:

https://logging.apache.org/log4j/2.x/security.html

On vous demande d'aller au plus pressé et de mettre en place le correctif via fail2ban dans un 1er temps sur tous vos serveurs

Le correctif fail2ban consiste à ajouter une règle de filtrage :

- dans le fichier jail.conf ajouter la section

```
[log4j-jndi]
maxretry = 1
enabled = true
port = 80,443
logpath = /var/log/fail2ban.log
```

Ansible - Pratiques
7.4 Projet

- Ajouter le fichier log4j-jndi.conf dans le répertoire /etc/fail2ban/filter.d
- contenu de ce fichier

```
[Definition]
failregex    = (?i)^<HOST> .* ".*$.*(7B|{).*(lower:)?.*j.*n.*d.*i.*:.*".*?$
```

Ansible - Liens

Pour ce cours

- https://www.redhat.com/fr/topics/automation/learning-ansible-tutorial
- https://linux.goffinet.org/ansible/presentation-produit-ansible