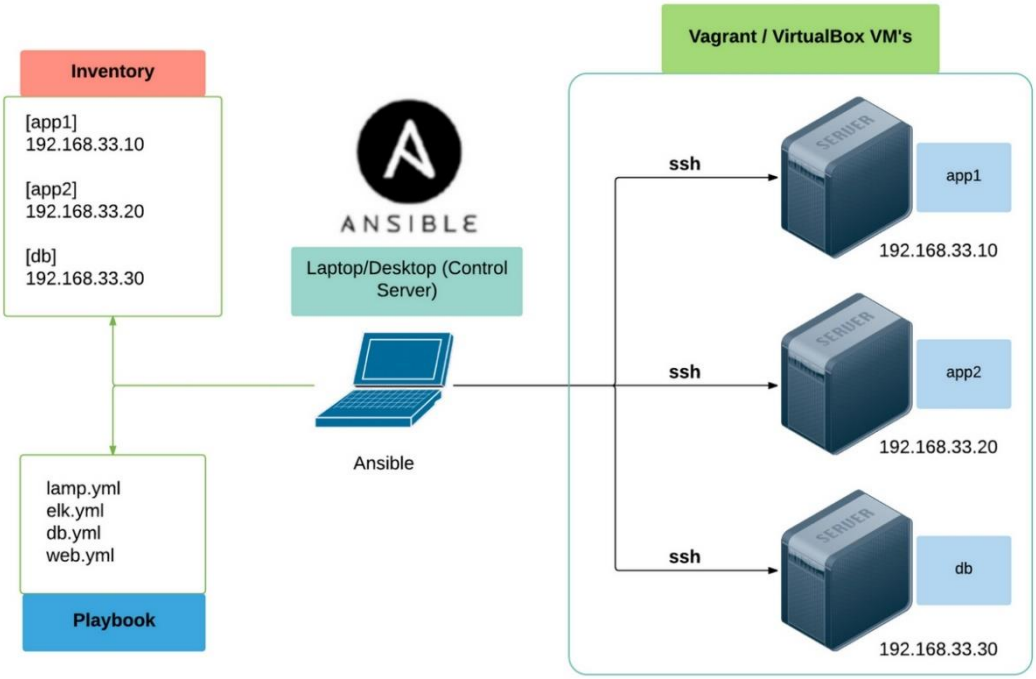
Rapport Ansible

I. Problématique

Auparavant, les administrateurs système géraient les serveurs manuellement, installaient les logiciels, changeaient les configurations et administraient les services sur des serveurs individuels. À mesure que les Datacenters se développaient et que les applications hébergées devenaient plus complexes, les administrateurs ont réalisé qu'ils ne pouvaient pas faire évoluer leur gestion manuelle des systèmes aussi rapidement que les applications qu'ils activaient. Cela a également entravé la vitesse du travail des développeurs car l'équipe de développement était agile et publiait fréquemment des logiciels, mais les opérations informatiques passaient plus de temps à configurer les systèmes. C'est pourquoi des outils de provisionnement de serveur et de gestion de la configuration automatique ont prospéré. Une des solutions est Ansible.

II. Présentation de Ansible



Ansible est un gestionnaire de configuration et un outil de déploiement et d'orchestration très populaire et central dans le monde de l'infrastructure as code (IaC). Il fait donc également partie

de façon centrale du mouvement DevOps car il s'apparente à un véritable couteau suisse de l'automatisation des infrastructures.

Ansible a été créé en 2012 (plus récent que ses concurrents Puppet et Chef) autour d'une recherche de simplicité et du principe de configuration agentless.

Très orienté linux/opensource et versatile il obtient rapidement un franc succès et s'avère être un couteau suisse très adapté à l'automatisation DevOps et Cloud dans des environnements hétérogènes.

Ansible est agentless c'est à dire qu'il ne nécessite aucun service/daemon spécifique sur les machines à configurer.
La simplicité d'Ansible provient également du fait qu'il s'appuie sur des technologies linux omniprésentes et devenues universelles.

- **ssh** : connexion et authentification classique avec les comptes présents sur les machines.
- **python** : multiplateforme, un classique sous linux, adapté à l'administration système et à tous les usagés.

De fait Ansible fonctionne efficacement sur toutes les distributions linux, debian, centos, ubuntu en particulier (et maintenant également sur Windows).

1. Ansible pour la configuration

Ansible est semi-déclaratif c'est à dire qu'il s'exécute séquentiellement mais idéalement de façon idempotente.

Il permet d'avoir un état descriptif de la configuration :

- Qui soit auditable
- Qui peut évoluer progressivement
- Qui permet d'éviter que celle-ci ne dérive vers un état inconnu

2. Ansible pour le déploiement et l'orchestration

Peut être utilisé pour des opérations ponctuelles comme le déploiement :

- Vérifier les dépendances et l'état requis d'un système
- Récupérer la nouvelle version d'un code source
- Effectuer une migration de base de données
- Tests opérationnels (vérifier qu'un service répond)

3. Ansible à différentes échelles

Les cas d'usages d'Ansible vont de :

- Petit :
    - Un petit playbook (~script) fournit avec le code d'un logiciel pour déployer en mode test.
    - La configuration d'une machine de travail personnelle.
    - etc.
- Moyen :
    - Faire un lab avec quelques machines.
    - Déployer une application avec du code, une runtime (php/java etc) et une base de données à migrer.
- Grand :
    - Gestion de plusieurs DC avec des produits multiples.
    - Gestion multi-équipes et logging de toutes les opérations grâce à Ansible Tower.
    - etc.

III. Concepts ansibles

Ces concepts sont communs à tous les usages d'Ansible. Vous devez les comprendre avant d'utiliser Ansible :

1. Nœud de contrôle

C'est la machine à partir de laquelle vous exécutez les outils Ansible CLI (`ansible-playbook` , `ansible` , `ansible-vault et` autres). Vous pouvez utiliser n'importe quel ordinateur qui répond aux exigences logicielles comme nœud de contrôle - les ordinateurs portables, les bureaux partagés et les serveurs peuvent tous exécuter Ansible. Plusieurs nœuds de contrôle sont possibles, mais les serveur Ansible ne se coordonne pas entre eux.

2. Nœuds gérés

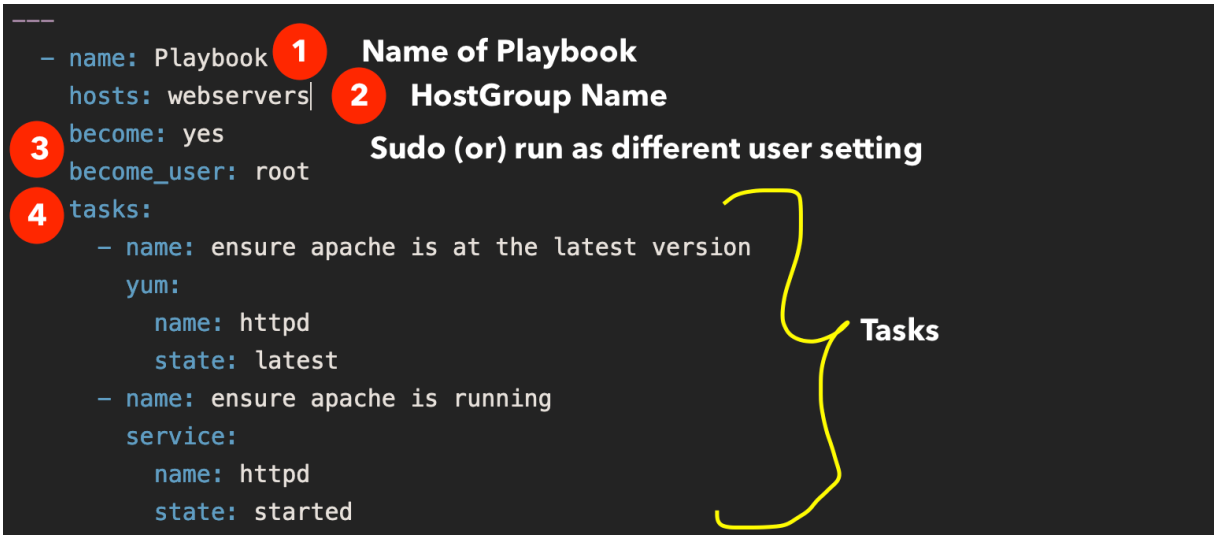
Également appelés "hôtes", il s'agit des périphériques cibles (serveurs, appareils réseau ou tout ordinateur) que vous souhaitez gérer avec Ansible. Ansible n'est normalement pas installé sur les nœuds gérés, sauf si vous utilisez `ansible-pull` , mais cela est rare et n'est pas la configuration recommandée.

3. Inventaire

Une liste de nœuds gérés fournie par une ou plusieurs « sources d'inventaire ». Votre inventaire peut spécifier des informations spécifiques à chaque nœud, comme l'adresse IP. Il

est également utilisé pour affecter des groupes, qui permettent tous deux la sélection de nœuds dans la lecture et l'affectation de variables en bloc. Parfois, un fichier source d'inventaire est également appelé "fichier hôte".

4. Playbooks Ansible



Un playbook est un fichier YAML qui contient un ou plusieurs plays (jeux de tâches) et qui permet de définir l'état souhaité d'un système, tandis qu'un module Ansible est un script autonome qui peut être utilisé à l'intérieur d'un playbook Ansible.

- Les plays consistent en une série de tâches à exécuter sur les hôtes sélectionnés dans le fichier d'inventaire Ansible.
- Les tâches sont les éléments qui constituent un play et qui appellent les modules Ansible. Dans un play, les tâches sont exécutées dans l'ordre de déclaration.

Pendant son exécution, Ansible suit l'état du système. S'il analyse un système et détecte une incohérence entre la description du système dans le playbook et son état réel, il fait les ajustements nécessaires pour que l'état réel corresponde à celui du playbook.

Ansible inclut un mode de vérification, qui vous permet de confirmer les playbooks et commandes ad hoc avant qu'un changement soit effectué sur le système. Ainsi, vous pouvez voir ce qu'Ansible modifierait, sans pour autant faire la modification.

- Les gestionnaires Ansible servent à exécuter une tâche spécifique uniquement après un changement du système. Ils sont déclenchés par des tâches et exécutés une seule fois, après tous les autres plays du playbook.

- Les variables Ansible permettent de modifier la manière dont un playbook s'exécute. Elles sont utilisées pour prendre en compte la différence entre les

systèmes, notamment les versions de paquet et les chemins d'accès aux fichiers, car Ansible vous permet d'exécuter des playbooks sur plusieurs systèmes.

Les variables Ansible doivent être définies en fonction de l'action effectuée par votre playbook.

Les variables respectent les règles de priorité qui définissent l'ordre dans lequel elles s'appliquent. C'est un point à bien comprendre si vous souhaitez inclure des variables dans vos playbooks.

Pour utiliser Ansible, vous devez également comprendre le concept de collection. Les collections représentent un format de distribution de contenu Ansible qui peut inclure des playbooks, rôles, modules et plug-ins.

Les rôles Ansible correspondent à un type spécifique de playbook totalement autonome et portable. Ils contiennent les tâches, variables, modèles de configuration et tous les autres fichiers requis pour une orchestration complexe.

- Modules

Le module c'est le code ou les fichiers binaires qu'Ansible copie et exécute sur chaque nœud géré (si nécessaire) pour accomplir l'action définie dans chaque tâche. Chaque module a une utilisation particulière, de l'administration des utilisateurs sur un type spécifique de base de données à la gestion des interfaces VLAN sur un type spécifique de périphérique réseau. Vous pouvez invoquer un seul module avec une tâche ou invoquer plusieurs modules différents dans un playbook. Les modules Ansible sont regroupés en collections.

IV. Installation de Ansible

Prérequis : Nous aurons besoins de deux machines au moins

- **Un nœud de contrôle Ansible** : Le nœud de contrôle Ansible est la machine que nous utiliserons pour nous connecter aux hôtes Ansible et les contrôler par SSH. Votre nœud de contrôle Ansible peut être votre machine locale ou un serveur dédié à l'exécution d'Ansible. Assurez-vous que le nœud de contrôle ai un utilisateur non-root avec des privilèges sudo.
- **Un ou plusieurs hôtes Ansible** : Un hôte Ansible est toute machine que votre nœud de contrôle Ansible est configuré pour automatiser. Assurez-vous que chaque hôte Ansible possède la clé publique SSH du nœud de contrôle Ansible ajoutée aux authorized_keys d'un utilisateur système. Cet utilisateur peut être root ou un utilisateur normal avec des privilèges sudo.

Étape 1 — Installation d'Ansible

Pour commencer nous devons installer Ansible sur la machine qui va constituer le nœud de contrôle.

- Exécutez la commande suivante pour inclure le PPA (personal package archive) du projet officiel dans la liste des sources de votre système :

```
sudo apt-add-repository ppa:ansible/ansible
```

Pour commencer à utiliser Ansible comme moyen de gestion de votre infrastructure de serveur, vous devez installer le logiciel Ansible sur la machine qui servira de nœud de contrôle Ansible. Nous utiliserons pour cela les dépôts Ubuntu par défaut.

Tout d'abord, rafraîchissez l'index des packages de votre système avec :

```
sudo apt update
```

Suite à cette mise à jour, vous pouvez installer le logiciel Ansible avec :

```
1. sudo apt install ansible
```

Appuyez sur Y lorsque vous êtes invité à confirmer l'installation.

Votre nœud de contrôle Ansible dispose maintenant de tous les logiciels nécessaires pour administrer vos hôtes. Ensuite, nous verrons comment mettre en place un fichier d'inventaire, afin qu'Ansible puisse communiquer avec vos nœuds gérés.

Étape 2 — Configuration du fichier d'inventaire

Le *fichier d'inventaire* contient des informations sur les hôtes que vous gérerez avec Ansible. Vous pouvez inclure entre un et plusieurs centaines de serveurs dans votre fichier d'inventaire, et les hôtes peuvent être organisés en groupes et sous-groupes. Le fichier d'inventaire est également souvent utilisé pour définir des variables qui ne seront valables que pour des hôtes ou des groupes spécifiques, afin d'être utilisées dans les modèles de développement et les modèles. Certaines variables peuvent également affecter la façon dont un modèle de développement est exécuté, comme la variable ansible_python_interpreter que nous allons voir dans un instant.

Pour modifier le contenu de votre inventaire Ansible par défaut, ouvrez-le fichier/etc/ansible/hosts en utilisant l'éditeur de texte de votre choix sur votre nœud de contrôle Ansible :

```
1. sudo nano /etc/ansible/hosts
```

Note : bien qu'Ansible crée généralement un fichier d'inventaire par défaut à l'adresse etc/ansible/hosts, vous êtes libre de créer des fichiers d'inventaire à l'endroit qui répond le mieux à vos besoins. Dans ce cas, vous devrez fournir le chemin d'accès à votre fichier d'inventaire personnalisé avec le paramètre -i lors de l'exécution des commandes Ansible et des modèles de développement. L'utilisation

de fichiers d'inventaire par projet est une bonne pratique pour minimiser le risque de lancer un modèle de développement sur le mauvais groupe de serveurs.

Le fichier d'inventaire par défaut fourni par l'installation Ansible contient un certain nombre d'exemples que vous pouvez utiliser comme références pour établir votre inventaire. L'exemple suivant définit un groupe nommé `[servers]` avec trois serveurs différents, chacun identifié par un alias personnalisé : **server1**, **server2** et **server3**. Veuillez à remplacer les IP mises en évidence par les adresses IP de vos hôtes Ansible.

/etc/ansible/hosts

```
[servers]

server1 ansible_host=203.0.113.111

server2 ansible_host=203.0.113.112

server3 ansible_host=203.0.113.113



[all:vars]

ansible_python_interpreter=/usr/bin/python3
```

Le sous-groupe all:vars définit le paramètre d'hôte ansible_python_interpreter qui sera valable pour tous les hôtes inclus dans cet inventaire. Ce paramètre permet de s'assurer que le serveur distant utilise l' exécutable /usr/bin/python3 Python 3 au lieu de /usr/bin/python (Python 2.7) qui n'est pas présent sur les versions récentes d'Ubuntu.
Lorsque vous avez terminé, enregistrez et fermez le fichier en appuyant sur CTRL+X puis Y et ENTER pour confirmer vos modifications.
Chaque fois que vous voulez vérifier votre inventaire, vous pouvez exécuter :

```
1. ansible-inventory --list -y
```

Vous verrez une sortie similaire à celle-ci, mais contenant votre propre infrastructure de serveur telle que définie dans votre fichier d'inventaire :

```
Output

all:

  children:
```

```
servers:
  hosts:
    server1:
      ansible_host: 203.0.113.111
      ansible_python_interpreter: /usr/bin/python3
    server2:
      ansible_host: 203.0.113.112
      ansible_python_interpreter: /usr/bin/python3
    server3:
      ansible_host: 203.0.113.113
      ansible_python_interpreter: /usr/bin/python3
ungrouped: {}
```

Maintenant que vous avez configuré votre fichier d'inventaire, vous avez tout ce dont vous avez besoin pour tester la connexion à vos hôtes Ansible.

Étape 3 — Generation de cle SSH

Après avoir configuré le fichier d'inventaire pour inclure vos serveurs, il est temps de vérifier si Ansible est capable de se connecter à ces serveurs et d'exécuter des commandes via SSH.

SSH, ou shell sécurisé, est un protocole crypté utilisé par administrer pour communiquer avec les serveurs. Lorsque vous travaillez avec un serveur Ubuntu, il est probable que vous passiez la plupart de votre temps dans une session de terminal connectée à votre serveur via SSH.

Les clés SSH offrent un moyen sécurisé de se connecter à votre serveur et sont recommandées pour tous les utilisateurs.

- Création de la paire de clés

La première étape consiste à créer une paire de clés sur la machine cliente (généralement votre ordinateur) :

1. ssh-keygen

Par défaut, les versions récentes de ssh-keygen créeront une paire de clés RSA 3072 bits, ce qui est suffisamment sécurisé pour la plupart des cas .

Après avoir saisi la commande, vous devriez voir le résultat suivant :

```
Output

Generating public/private rsa key pair.

Enter file in which to save the key (/your_home/.ssh/id_rsa):
```

Appuyez sur Entrée pour enregistrer la paire de clés dans `.ssh/` le sous- répertoire de votre répertoire personnel ou spécifiez un autre chemin.

Si vous aviez précédemment généré une paire de clés SSH, vous pouvez voir l'invite suivante :

```
Output

/home/your_home/.ssh/id_rsa already exists.

Overwrite (y/n)?
```

Si vous choisissez d'écraser la clé sur le disque, vous **ne** pourrez plus vous authentifier avec la clé précédente. Soyez très prudent lorsque vous sélectionnez oui, car il s'agit d'un processus destructeur qui ne peut pas être inversé.

Vous devriez alors voir l'invite suivante :

```
Output

Enter passphrase (empty for no passphrase):
```

Ici, vous pouvez éventuellement entrer une phrase de passe sécurisée, ce qui est fortement recommandé. Une phrase de passe ajoute une couche de sécurité supplémentaire pour empêcher les utilisateurs non autorisés de se connecter.

Vous devriez alors voir une sortie semblable à la suivante :

```
Output

Your identification has been saved in /your_home/.ssh/id_rsa

Your public key has been saved in /your_home/.ssh/id_rsa.pub

The key fingerprint is:

SHA256:/hk7MJ5n5aiqdfTVUZr+2Qt+qCiS7BIm5Iv0dxrc3ks user@host

The key's randomart image is:

+---[RSA 3072]----+
```

```
|             .|
|              + |
|             +  |
| .          o . |
|o       S   . o  |
| + o. .oo. ..  .o|
|o = oooooEo+ ...o|
|.. o *o+=.*+o....|
|    =+=ooB=o.... |
+----[SHA256]-----+
```

Vous disposez maintenant d'une clé publique et privée que vous pouvez utiliser pour vous authentifier. L'étape suivante consiste à placer la clé publique sur votre serveur afin que vous puissiez utiliser l'authentification basée sur la clé SSH pour vous connecter.

Pour ce guide, nous utiliserons le compte **root** Ubuntu, car c'est généralement le seul compte disponible par défaut sur les serveurs nouvellement créés. Si vos hôtes Ansible ont déjà créé un utilisateur sudo régulier, vous êtes encouragé à utiliser ce compte à la place.

- **Copie de la clé publique sur votre serveur Ubuntu**

Le moyen le plus rapide de copier votre clé publique sur l'hôte Ubuntu consiste à utiliser un utilitaire appelé ssh-copy-id. En raison de sa simplicité, cette méthode est fortement recommandée si elle est disponible. Si vous n'en avez pas ssh-copy-id sur votre ordinateur client, vous pouvez utiliser l'une des deux méthodes alternatives fournies dans cette section (copie via SSH basé sur un mot de passe ou copie manuelle de la clé).

- **Copie de la clé publique à l'aide de ssh-copy-id**

L' ssh-copy-id outil est inclus par défaut dans de nombreux systèmes d'exploitation, vous pouvez donc l'avoir disponible sur votre système local. Pour que cette méthode fonctionne, vous devez déjà disposer d'un accès SSH basé sur un mot de passe à votre serveur.

Pour utiliser l'utilitaire, vous spécifiez l'hôte distant auquel vous souhaitez vous connecter et le compte d'utilisateur auquel vous avez un accès SSH basé sur un mot de passe. Il s'agit du compte sur lequel votre clé SSH publique sera copiée.

La syntaxe est :

```
1. ssh-copy-id username@remote_host
```

Vous pouvez voir le message suivant :

```
Output

The authenticity of host '203.0.113.1 (203.0.113.1)' can't be
established.

ECDSA key fingerprint is
fd:fd:d4:f9:77:fe:73:84:e1:55:00:ad:d6:6d:22:fe.

Are you sure you want to continue connecting (yes/no)? yes
```

Cela signifie que votre ordinateur local ne reconnaît pas l'hôte distant. Cela se produira la première fois que vous vous connecterez à un nouvel hôte. Tapez "oui" et appuyez sur ENTERpour continuer.

Ensuite, l'utilitaire analysera votre compte local à la recherche de la id_rsa.pub clé que nous avons créée précédemment. Lorsqu'il trouve la clé, il vous demandera le mot de passe du compte de l'utilisateur distant :

```
Output

/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s),
to filter out any that are already installed

/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you
are prompted now it is to install the new keys

username@203.0.113.1's password:
```

Tapez le mot de passe (votre saisie ne sera pas affichée, pour des raisons de sécurité) et appuyez sur ENTER. L'utilitaire se connectera au compte sur l'hôte distant à l'aide du mot de passe que vous avez fourni. Il copiera ensuite le contenu de votre ~/.ssh/id_rsa.pubclé dans un fichier du ~/.ssh répertoire personnel du compte distant appelé authorized_keys.

Vous devriez voir la sortie suivante :

```
Output

Number of key(s) added: 1



Now try logging into the machine, with:   "ssh 'username@203.0.113.1'"
```

```
and check to make sure that only the key(s) you wanted were added.
```

Etape4 :Test de connexion

Vous pouvez utiliser l'argument -u pour spécifier l'utilisateur du système à distance. Lorsqu'il n'est pas fourni, Ansible essaiera de se connecter en tant qu'utilisateur actuel de votre système sur le nœud de contrôle.

Depuis votre machine locale ou votre nœud de contrôle Ansible, exécutez :

```
1. ansible all -m ping -u root
```

Cette commande utilisera le module ping intégré d'Ansible pour effectuer un test de connectivité sur tous les nœuds de votre inventaire par défaut, en se connectant en tant que **root**. Le module ping testera :

- si les hôtes sont accessibles ;
- si vous avez des identifiants SSH valides ;
- si les hôtes sont capables d'exécuter les modules Ansible en utilisant Python.

Vous devriez obtenir un résultat similaire à ceci :

```
Output

server1 | SUCCESS => {

    "changed": false,

    "ping": "pong"

}

server2 | SUCCESS => {

    "changed": false,

    "ping": "pong"

}

server3 | SUCCESS => {

    "changed": false,

    "ping": "pong"

}
```

Si c'est la première fois que vous vous connectez à ces serveurs via SSH, il vous sera demandé de confirmer l'authenticité des hôtes auxquels vous vous connectez via Ansible. Lorsque vous y êtes invité, tapez yes et appuyez ensuite sur ENTRÉE pour confirmer.

Une fois que vous obtenez une réponse « pong » d'un hôte, cela signifie que vous êtes prêt à exécuter les commandes Ansible et les playbooks sur ce serveur.

Note : si vous ne parvenez pas à obtenir une réponse positive de vos serveurs, consultez notre guide de triche pour plus d'informations sur la manière d'exécuter les commandes Ansible avec différentes options de connexion.

Étape 4 — Exécution des commandes Ad-Hoc (facultatif)

Après avoir confirmé que votre nœud de contrôle Ansible est capable de communiquer avec vos hôtes, vous pouvez commencer à exécuter des commandes ad-hoc et des modèles de développement sur vos serveurs.

Toute commande que vous exécuteriez normalement sur un serveur distant par SSH peut être exécutée avec Ansible sur les serveurs spécifiés dans votre fichier d'inventaire. A titre d'exemple, vous pouvez vérifier l'utilisation des disques sur tous les serveurs avec :

```
1. ansible all -a "df -h" -u root
```

Référence

https://openclassrooms.com/fr/courses/2035796-utilisez-ansible-pour-automatiser-vos-taches-de-configuration/6371875-preparez-la-communication-avec-les-nodes

https://www.digitalocean.com/community/tutorials/how-to-set-up-ssh-keys-on-ubuntu-20-04

https://www.redhat.com/fr/topics/automation/learning-ansible-tutorial

https://cours.hadrienpelissier.fr/all_content/