Lab 2 - Configuration de Ansible

Étape 1 - Création d'un fichier d'inventaire personnalisé

Lors de l'installation, Ansible crée un fichier d'inventaire qui se trouve généralement dans /etc/ansible/hosts. Il s'agit de l'emplacement par défaut utilisé lorsqu'un fichier d'inventaire personnalisé n'est pas fourni avec l'option -i, lors d'un playbook ou de l'exécution d'une commande.

Même si vous pouvez utiliser ce fichier sans problème, l'utilisation de fichiers d'inventaire par projet est une bonne pratique pour éviter de mélanger les serveurs lors de l'exécution de commandes et de playbooks. Le fait de disposer de fichiers d'inventaire par projet facilitera également le partage de la configuration de l'approvisionnement avec les collaborateurs, étant donné que vous incluez le fichier d'inventaire dans le référentiel de code du projet.

Pour commencer, accédez à votre dossier de départ et créez un nouveau fichier d'inventaire à l'aide de l'éditeur de texte de votre choix.

```
vi hosts
```

Une liste de vos nœuds, avec un serveur par ligne, suffit pour mettre en place un fichier d'inventaire fonctionnel. Les noms d'hôte et les adresses IP sont interchangeables :

```
192.168.208.164
192.168.208.165
192.168.208.166
```

Une fois que vous avez configuré un fichier d'inventaire, vous pouvez utiliser la commande **ansible-inventory** pour valider et obtenir des informations sur votre inventaire Ansible:

```
ansible-inventory -i hosts --list
{
    "_meta": {
        "hostvars": {}
    },
    "all": {
        "children": [
            "ungrouped"
        ]
    },
    "ungrouped": {
        "hosts": [
            "192.168.208.164",
            "192.168.208.165",
            "192.168.208.166"
        ]
    }
}
```

Même si nous n'avons configuré aucun groupe dans notre inventaire, la sortie montre 2 groupes distincts qui sont automatiquement déduits par Ansible: **`all`** et **`ungrouped`**.

Comme son nom l'indique, **`all`** est utilisé pour désigner tous les serveurs de votre fichier d'inventaire, quelle que soit leur organisation. Le groupe **`ungrouped`** est utilisé pour faire référence aux serveurs qui ne sont pas répertoriés dans un groupe.

Exécution de commandes et de playbooks avec des inventaires personnalisés

Pour exécuter des commandes Ansible avec un fichier d'inventaire personnalisé, utilisez l'option **-i** comme suit:

```
ansible all -i hosts -m ping
```

Cela exécuterait le module ping sur toutes les hôtes répertoriées dans votre fichier d'inventaire personnalisé.

De même, voici comment exécuter des playbooks Ansible avec un fichier d'inventaire personnalisé:

```
ansible-playbook -i hosts playbook.yml
```

Étape 2 - Organisation des serveurs en groupes et sous-groupes

Dans le fichier d'inventaire, vous pouvez organiser vos serveurs en différents groupes et sous-groupes. En plus d'aider à garder vos hôtes en ordre, cette pratique vous permettra d'utiliser des variables de groupe, une fonctionnalité qui peut grandement faciliter la gestion de plusieurs environnements de préparation avec Ansible.

Un hôte peut faire partie de plusieurs groupes. Le fichier d'inventaire suivant au format INI montre une configuration avec quatre groupes:

- **webservers**,
- **dbservers**,
- **development**
- **production**.

Vous remarquerez que les serveurs sont regroupés selon deux qualités différentes: leur finalité (web et base de données) et leur utilisation (développement et production).

```
[webservers]
192.168.208.164
192.168.208.165

[dbservers]
192.168.208.166
```

```
[development]
192.168.208.165
192.168.208.166

[production]
192.168.208.164
192.168.208.166
```

Si vous exécutez à nouveau la commande `ansible-inventory` avec ce fichier d'inventaire, vous verrez la disposition suivante:

```
[ansible@server ~]$ ansible-inventory -i hosts --list
{
    "_meta": {
        "hostvars": {}
    },
    "all": {
        "children": [
            "dbservers",
            "development",
            "production",
            "ungrouped",
            "webservers"
        ]
    },
    "dbservers": {
        "hosts": [
            "192.168.208.166"
        ]
    },
    "development": {
        "hosts": [
            "192.168.208.165",
            "192.168.208.166"
        ]
    },
    "production": {
        "hosts": [
            "192.168.208.164",
            "192.168.208.166"
        ]
    },
    "webservers": {
        "hosts": [
            "192.168.208.164",
            "192.168.208.165"
        ]
    }
}
```

Il est également possible d'agréger plusieurs groupes en tant qu'enfants sous un groupe «parent». Le «parent» est alors appelé un **métagroupe**. L'exemple suivant montre une autre façon d'organiser l'inventaire précédent à l'aide de métagroupes pour obtenir une disposition comparable, mais plus granulaire:

```
[web_dev]
192.168.208.164
[web_prod]
192.168.208.165
[db_dev]
192.168.208.166
[db_prod]
192.168.208.165
[webservers:children]
web_dev
web_prod
[dbservers:children]
db_dev
db_prod
[development:children]
web_dev
db_dev
[production:children]
web_prod
db_prod
```

Plus vous avez de serveurs, plus il est judicieux de séparer les groupes ou de créer des arrangements alternatifs afin de pouvoir cibler de plus petits groupes de serveurs selon vos besoins.

Étape 3 - Configuration des alias d'hôte

Vous pouvez utiliser des alias pour nommer les serveurs de manière à faciliter le référencement de ces serveurs ultérieurement, lors de l'exécution de commandes et de playbooks.

Pour utiliser un alias, incluez une variable nommée `ansible_host` après le nom de l'alias, contenant l'adresse IP ou le nom d'hôte correspondant du serveur qui doit répondre à cet alias:

```
[ansible@server ~]$ cat hosts
# ... contenu éliminé
server1 ansible_host=192.168.208.164
server2 ansible_host=192.168.208.165
server3 ansible_host=192.168.208.166
# ... contenu éliminé
```

Si vous exécutiez la commande **ansible-inventory** avec ce fichier d'inventaire, vous verriez une sortie similaire à celle-ci:

```
[ansible@server ~]$ ansible-inventory -i hosts --list
{
    "_meta": {
        "hostvars": {
            "server1": {
                "ansible_host": "192.168.208.164"
            },
            "server2": {
                "ansible_host": "192.168.208.165"
            },
            "server3": {
                "ansible_host": "192.168.208.166"
            }
        }
    },
    "all": {
        "children": [
            "dbservers",
            "development",
            "production",
            "ungrouped",
            "webservers"
        ]
    },
    "db_dev": {
        "hosts": [
            "192.168.208.166"
        ]
    },
    "db_prod": {
        "hosts": [
            "192.168.208.165"
        ]
    },
    "dbservers": {
        "children": [
            "db_dev",
            "db_prod"
        ]
    },
    "development": {
        "children": [
            "db_dev",
            "web_dev"
        ]
    },
    "production": {
        "children": [
            "db_prod",
            "web_prod"
        ]
    },
    "ungrouped": {
        "hosts": [
            "server1",
            "server2",
            "server3"
        ]
    },
    "web_dev": {
        "hosts": [
            "192.168.208.164"
        ]
    },
```

```
    "web_prod": {
        "hosts": [
            "192.168.208.165"
        ]
    },
    "webservers": {
        "children": [
            "web_dev",
            "web_prod"
        ]
    }
}
```

Notez comment les serveurs sont désormais référencés par leurs alias au lieu de leurs adresses IP ou noms d'hôte. Cela facilite le ciblage de serveurs individuels lors de l'exécution de commandes et de playbooks.

Étape 4 - Configuration des variables d'hôte

Il est possible d'utiliser le fichier d'inventaire pour configurer des variables qui modifieront le comportement par défaut d'Ansible lors de la connexion et de l'exécution de commandes sur vos nœuds. C'est en fait ce que nous avons fait à l'étape précédente, lors de la configuration des alias d'hôte. La variable `ansible_host` indique à Ansible où trouver les nœuds distants, au cas où un alias serait utilisé pour faire référence à ce serveur.

Les variables d'inventaire peuvent être définies par hôte ou par groupe. En plus de personnaliser les paramètres par défaut d'Ansible, ces variables sont également accessibles à partir de vos playbooks, ce qui permet une personnalisation supplémentaire pour les hôtes et groupes individuels.

L'exemple suivant montre comment définir l'utilisateur distant par défaut lors de la connexion à chacun des nœuds répertoriés dans ce fichier d'inventaire:

```
# ... contenu éliminé
centos01 ansible_host=192.168.208.164 ansible_user=devops
# ... contenu éliminé
ubuntu01 ansible_host=192.168.208.165 ansible_user=ubuntu
# ... contenu éliminé
centos02 ansible_host=192.168.208.166 ansible_user=devops
# ... contenu éliminé
```

Vous pouvez également créer un groupe pour agréger les hôtes avec des paramètres similaires, puis configurer leurs variables au niveau du groupe:

```
# ... contenu éliminé
centos01 ansible_host=192.168.208.164
# ... contenu éliminé
ubuntu01 ansible_host=192.168.208.165
```

```
# ... contenu éliminé
centos02 ansible_host=192.168.208.166
# ... contenu éliminé

[webservers:vars]
ansible_user=ansible

[dbservers:vars]
ansible_user=ubuntu
```

Notez que toutes les variables d'inventaire sont répertoriées dans le nœud `_meta` dans la sortie JSON produite par `ansible-inventory`.

Étape 5 - Utilisation de modèles pour cibler l'exécution

Lors de l'exécution de commandes et de playbooks avec Ansible, vous devez fournir une cible. Les modèles vous permettent de cibler des hôtes, des groupes ou des sous-groupes spécifiques dans votre fichier d'inventaire. Ils sont très flexibles et prennent en charge les expressions régulières et les caractères génériques. Considérez le fichier d'inventaire suivant:

```
[web_dev]
192.168.208.164

[web_prod]

192.168.208.165

[db_dev]
192.168.208.166

[db_prod]
192.168.208.165

[webservers:children]
web_dev
web_prod

[dbservers:children]
db_dev
db_prod

[development:children]

web_dev
db_dev
```

```
[production:children]
web_prod
db_prod

[webservers:vars]
ansible_user=devops

[dbservers:vars]
ansible_user=ubuntu
```

Imaginons maintenant que vous deviez exécuter une commande ciblant uniquement le(s) serveur(s) web qui s'exécutent en production. Dans cet exemple, ce critère ne correspond qu'à 10.0.0.22; cependant, il se peut que vous ayez un grand groupe de serveurs de base de données dans ce groupe. Au lieu de cibler individuellement chaque serveur, vous pouvez utiliser le modèle suivant:

```
[ansible@server ~]$ ansible webservers:\&production -m ping -i hosts
```

Le caractère **&** représente l'opération logique AND, ce qui signifie que les cibles valides doivent être dans les deux groupes. Comme il s'agit d'une commande ad hoc exécutée sur Bash, nous devons inclure le caractère **** d'échappement dans l'expression.

L'exemple précédent ne ciblerait que les serveurs présents à la fois dans les groupes dbservers et production. Si vous vouliez faire le contraire, en ciblant uniquement les serveurs qui sont présents dans le groupe dbservers mais pas dans le groupe production, vous utiliseriez à la place le modèle suivant:

```
[ansible@server ~]$ ansible dbservers:\!production -m ping -i hosts
```

Pour indiquer qu'une cible ne doit pas faire partie d'un certain groupe, vous pouvez utiliser le caractère **!**. Une fois de plus, nous incluons le caractère **** d'échappement dans l'expression pour éviter les erreurs de ligne de commande, car les deux & et ! sont des caractères spéciaux qui peuvent être analysés par Bash.

```
[ansible@server ~]$ ansible server1:server3 -m ping -i hosts
```

Le tableau suivant contient quelques exemples différents de modèles courants que vous pouvez utiliser lors de l'exécution de commandes et de playbooks avec Ansible:

Modèle	Objectif de résultat
all	Tous les hôtes de votre fichier d'inventaire
host1	Un seul hôte (host1)
host1:host2	Les deux host1 et host2
group1	Un seul groupe (group1)
group1:group2	Tous les serveurs dans group1 et group2

group1:&group2	Seuls les serveurs qui sont à la fois dans group1 et group2
group1:!group2	Serveurs en group1 sauf ceux également en group2

Personnaliser la Configuration de Ansible

Grâce à un modèle de surcharge simple, il est possible de donner à Ansible un fichier de configuration ansible.cfg dans lequel vous pouvez reconfigurer partiellement Ansible pour vos besoins.

Ansi, qu'il s'agisse de spécifier un hostfile alternatif pour ne plus avoir à préciser -i meshosts à chaque lancement, ou bien supprimer les inutiles fichiers .retry, ou toute autre option Ansible, vous n'avez bien souvent qu'à créer un fichier ansible.cfg là ou vous lancez vos playbooks pour qu'Ansible aille automatiquement chercher ce fichier en premier lors de son démarrage. L'ordre de recherche est celui-ci :

- ANSIBLE_CONFIG (an environment variable)
- ansible.cfg (in the current directory)
- .ansible.cfg (in the home directory)
- /etc/ansible/ansible.cfg

créez le fichier ansible.cfg dans le répertoire courant:

```
vi ansible.cfg
```

Commencer par modifier la section **defaults**

```
[defaults]
```

Spécifier le fichier hosts comme inventaire

```
inventory = hosts
```

Arrêter la vérification des clés pour les nouveaux noeuds

```
host_key_checking = False
```

Récupérer le temps passé dans l'execution des tâches

```
callback_whitelist = timer
```

Activer la parallélisme

```
forks = 5
```

Changer le facts gathering en explicite

```
gathering = explicit
```

Modifier la section **ssh_connection**

```
[ssh_connection]
```

Activer le pipeline python pour optimiser le temps d'exécution des playbooks

```
pipelining = True
```

Ajuster les arguments de connexion SSH

```
ssh_args = -o ControlMaster=auto -o ControlPersist=60s
```

Vérifier l'état de votre configuration avec la commande ansible-config:

```
ansible-config view | grep inventory
```