GESTION DES INVENTAIRES - FORMAT INI

FORMAT INI - SYNTAXE

STRUCTURE DE BASE

Les fichiers **INI** sont composés de **sections** et de paires **clé-valeur** :

```
[section]
cle = valeur
```

COMMENTAIRES

Les commentaires commencent par `;` ou `#` :

```
; Ceci est un commentaire
# Ceci est également un commentaire
```

Note : Bien que les deux symboles fonctionnent pour les commentaires, il est préférable d'utiliser `#` pour une meilleure lisibilité et cohérence avec d'autres langages de programmation.

EXEMPLES D'UTILISATION - FORMAT INI

CRÉATION D'UN FICHIER D'INVENTAIRE SIMPLE

```
serveur-web01 ansible_host=192.168.1.1 ansible_user=admin
serveur-web02 ansible_host=192.168.1.2 ansible_user=admin
```

GESTION DES GROUPES

Pour regrouper les hôtes, ajoutez une section `[groupe]` :

```
[serveurs-web]
serveur-web01 ansible_host=192.168.1.1 ansible_user=admin
serveur-web02 ansible_host=192.168.1.2 ansible_user=admin
```

VARIABLES D'INVENTAIRE

Pour définir des variables propres à un **groupe** ou un **hôte** :

```
[serveurs-web:vars]
http_port = 80
max_clients = 200

serveur-web01 ansible_host=192.168.1.1 ansible_user=admin http_port=8080
serveur-web02 ansible_host=192.168.1.2 ansible_user=admin
```

Ici, `serveur-web01` utilisera le port 8080 tandis que `serveur-web02` utilisera le port par défaut 80.

GESTION DES INVENTAIRES : FORMAT YAML

FORMAT YAML - SYNTAXE

STRUCTURE DE BASE

Un fichier **YAML** doit respecter une structure basée sur l'**indentation** des éléments. L'indentation est généralement réalisée avec deux espaces.

Type d'élément	Syntaxe
Commentaires	# Commentaire
Listes	- Item
Dictionnaires	clé : valeur

COMMENTAIRES

Les commentaires sont écrits après un dièse (**#**) et ne sont pas interprétés par **Ansible**.

Exemple de commentaire dans un fichier YAML :

```yaml
# Ceci est un commentaire
- name: Tâche avec un commentaire
  command: echo "Hello world"
```

FORMAT YAML - EXEMPLES D'UTILISATION

CRÉATION D'UN FICHIER D'INVENTAIRE SIMPLE

```
all:
  hosts:
    machine1:
    machine2:
    machine3:
```

GESTION DES GROUPES

```
all:
  children:
    groupe1:
      hosts:
        machine1:
        machine2:
    groupe2:
      hosts:
        machine3:
```

Exemple de fichier d'inventaire groupant les nœuds par groupes

groupe	machines
groupe1	machine1
	machine2
groupe2	machine3

VARIABLES D'INVENTAIRE

```
all:
  hosts:
    machine1:
      var1: valeur1
      var2: valeur2
```

GESTION DES GROUPES ET SOUS-GROUPES

GESTION DES GROUPES - YAML

CRÉATION DE GROUPES

```
all:
  children:
    groupe1:
      hosts:
        machine1:
        machine2:
```

AJOUT DE MACHINES AUX GROUPES

```
all:
  children:
    groupe1:
      hosts:
        machine1:
        machine2:
```

VARIABLES DE GROUPE

```
all:
  children:
    groupe1:
      hosts:
        machine1:
      vars:
        var1: valeur1
        var2: valeur2
```

Note : Les variables de groupe permettent de définir des variables spécifiques à un groupe d'hôtes dans Ansible. Elles sont déclarées sous la clé 'vars' pour chaque groupe.

GESTION DES SOUS-GROUPES - YAML

CRÉATION DE SOUS-GROUPES

```
all:
  children:
    groupe1:
      children:
        sous_groupe1:
          hosts:
            machine1:
```

AJOUT DE MACHINES AUX SOUS-GROUPES

```
all:
  children:
    groupe1:
      children:
        sous_groupe1:
          hosts:
            machine1:
            machine2:
```

Colonne 1	Colonne 2
all	Niveau racine
children	Groupes parents
groupe1	Nom du groupe parent
sous_groupe1	Nom du sous-groupe
hosts	Liste des machines
machineX	Nom des machines

VARIABLES DE SOUS-GROUPE

```
all:
  children:
    groupe1:
      children:
        sous_groupe1:
          hosts:
            machine1:
          vars:
            var1: valeur1
            var2: valeur2
```

GESTION DES GROUPES ET SOUS-GROUPES

GROUPES

CRÉATION DE GROUPES

En **INI** :

```
[group_name]
host1
host2
```

En **YAML** :

```
group_name:
  hosts:
    host1:
    host2:
```

AJOUT DE MACHINES AUX GROUPES

En **INI** :

```
[group_name]
host1
host2
```

En **YAML** :

```
group_name:
  hosts:
    host1:
    host2:
```

VARIABLES DE GROUPE

En **INI** :

```
[group_name:vars]
variable1=value1
variable2=value2
```

En **YAML** :

```
group_name:
  vars:
    variable1: value1
    variable2: value2
```

SOUS-GROUPES

CRÉATION DE SOUS-GROUPES

En **INI** :

```
[group_name:children]
subgroup1
subgroup2
```

En **YAML** :

```
group_name:
  children:
    subgroup1: {}
    subgroup2: {}
```

AJOUT DE MACHINES AUX SOUS-GROUPES

En **INI** :

```
[subgroup1]
host3
host4
```

En **YAML** :

```
subgroup1:
  hosts:
    host3:
    host4:
```

VARIABLES DE SOUS-GROUPE

En **INI** :

```
[subgroup1:vars]
variable3=value3
variable4=value4
```

En **YAML** :

```
subgroup1:
  vars:
    variable3: value3
    variable4: value4
```

COMPARAISON DES FORMATS

AVANTAGES ET INCONVÉNIENTS DU FORMAT INI

Avantages :

- Facilité de lecture et d'écriture
- Syntaxe simple et familière
- Légèrement plus rapide à analyser

Inconvénients :

- Moins puissant que le format **YAML**
- Non hiérarchique
- Limites dans la représentation de données complexes

AVANTAGES ET INCONVÉNIENTS DU FORMAT YAML

Avantages :

- Représentation de données hiérarchiques et complexes
- Syntaxe plus propre et moins encombrée
- Plus facile à maintenir et à étendre

Inconvénients :

- Moins performant en termes de rapidité d'analyse
- Syntaxe sensible à l'indentation
- Peut être moins intuitive pour les utilisateurs non expérimentés

CHOIX DU FORMAT EN FONCTION DU PROJET ET DES BESOINS

- Utiliser le format **INI** pour les projets **simples** et **petits**
- Choisir le format **YAML** pour les projets **complexes** et **larges**

BONNES PRATIQUES

ORGANISATION DES FICHIERS D'INVENTAIRE

- Utilisez des dossiers pour organiser les inventaires par **environnement** (prod, dev, test)
- Adoptez une structure **claire** et **compréhensible** pour faciliter la maintenance et l'évolution du projet

Exemple de structure

```
inventaire/
├── prod/
│   ├── hosts
│   └── group_vars/
├── dev/
│   ├── hosts
│   └── group_vars/
└── test/
    ├── hosts
    └── group_vars/
```

GESTION DES VARIABLES ET DES SECRETS

- Utilisez des **fichiers de variables séparés** pour stocker et gérer les variables spécifiques à un groupe ou à une machine
- Utilisez **Ansible Vault** pour sécuriser les secrets et les données sensibles dans vos fichiers d'inventaire

Méthode	Description
Fichiers de variables séparés	Permet la gestion plus simple des variables spécifiques aux groupes et machines en les rassemblant dans un fichier distinct.
Ansible Vault	Chiffre les données sensibles tel que les mots de passe, clés API et autres secrets pour éviter l'accès non autorisé.

UTILISATION DES ALIAS ET DES MÉTADONNÉES

- Utilisez des **alias** pour améliorer la lisibilité et la maintenance des fichiers d'inventaire
- Ajoutez des **métadonnées** pour décrire l'objectif, l'utilisation et les responsables de chaque inventaire ou groupe

Syntaxe d'alias	Exemple
Alias simple	webserver_alias = web01.example.com
Alias avec métadonnées	[web_group:vars] webserver_alias = web01.example.com description = "Serveurs web" responsible_person = "John Doe"

RÉPARTITION DES TÂCHES ET DES RESPONSABILITÉS

- Assignez les responsabilités de gestion des **inventaires** et des **variables** aux membres de l'équipe en fonction de leurs compétences et de leur expertise
- Automatisez autant que possible pour réduire les risques d'**erreurs humaines** et améliorer l'**efficacité** des mises à jour des inventaires