# Chapitre 4
# Inventaires : notions avancées

## 1. Objectifs du chapitre et prérequis

### 1.1 Contexte et prérequis

Ce chapitre va être consacré à plusieurs thématiques autour de l'inventaire. Vous verrez notamment comment stocker des données sensibles (identifiants et mots de passe) dans un fichier ainsi que la notion d'inventaire dynamique d'Ansible.

Au niveau des inventaires dynamiques, vous verrez quelques applications sur des produits du marché que vous pourriez rencontrer. Vous n'avez sûrement pas envie de réécrire un inventaire pour Ansible et de le maintenir alors que cette information est déjà disponible quelque part (et que vous y avez déjà consacré un temps précieux).

Les exemples traités dans ce chapitre partent du principe que vous disposez de droits d'administration sur vos machines.

Pour la partie sur l'écriture d'inventaire dynamique, vous aurez besoin de quelques notions en programmation Python.

### 1.2 Fichiers téléchargeables

Vous pouvez récupérer les exemples des répertoires inventaires et variables sur le repository GitHub suivant : https://github.com/EditionsENI/ansible

Vous pouvez également récupérer ces fichiers dans l'archive **chapitre-04.tar.gz** depuis la page du livre sur le site des Éditions ENI.

## 2. Chiffrement de fichiers

### 2.1 Contexte

Dans le chapitre de la découverte de l'inventaire, vous avez vu comment gérer ce dernier et stocker des identifiants ou des mots de passe. Petit problème, ces données sont stockées dans des fichiers plats. Quiconque ayant accès à vos inventaires pourrait en extraire des informations et s'en servir.

Dans ce qui va suivre, vous verrez un mécanisme qui permet de chiffrer vos fichiers afin d'éviter que des mots de passe en clair soient directement accessibles.

Remarque

*Tous les exemples qui vont suivre utiliseront le mot de passe « ansible ».*

### 2.2 Stockage d'identifiants de connexion

La première étape va être de créer un fichier dans lequel vous allez stocker des informations de connexion à une base de données MySQL.
Vous utiliserez pour cela deux champs : `intranet_mysql_db_user` et `intranet_mysql_db_password`.

Ces deux identifiants vous serviront pour une hypothétique application intranet.

Ci-dessous, la déclaration que vous utiliserez comme exemple pour la suite :

```
intranet_mysql_db_user: "intranet"
intranet_mysql_db_password: "1ntran3t!"
```

## 2.3 Chiffrement du fichier entier

Le chiffrement d'éléments dans Ansible se fait à l'aide de l'outil `ansible-vault`. Ce dernier peut fonctionner avec un mot de passe ou en passant des fichiers en argument.

Dans les deux cas, pour le chiffrement d'un fichier entier, il faut lancer `ansible-vault` avec l'option `encrypt` (chiffrer en anglais) suivie du nom de fichier.

Remarque

*Ansible peut utiliser des fichiers chiffrés pour tous les fichiers qu'il manipule. Vous pouvez ainsi chiffrer les fichiers en local avant recopie à distance.*

### 2.3.1 Par mot de passe

Ci-dessous la commande à lancer pour chiffrer le fichier **intranet-pass.yml** :

```
$ ansible-vault encrypt intranet-pass.yml
```

Ansible vous demandera alors un mot de passe qu'il faudra ensuite entrer une seconde fois pour confirmation. Ci-dessous la sortie de la commande lorsque le chiffrement s'est bien passé :

```
New Vault password:
Confirm New Vault password:
Encryption successful
```

Ci-dessous, un extrait du fichier **intranet-pass.yml** après chiffrement :

```
$ANSIBLE_VAULT;1.1;AES256
6536346161663[...]
```

L'utilisation du fichier se fera comme d'habitude avec l'option `-e @fichier-vault.yml`. Pour indiquer à Ansible qu'il faut déchiffrer les fichiers `vault`, vous ajouterez l'option `--ask-vault-pass`. Ci-dessous le chargement du fichier **intranet-pass.yml** combiné à un appel à `debug` pour afficher le contenu de la variable `intranet_mysql_db_user` :

```
$ ansible -m debug -a var=intranet_mysql_db_user localhost \
          -e @intranet-pass.yml --ask-vault-pass
```

Ci-dessous le résultat de cet appel :

```
Vault password:
[WARNING]: provided hosts list is empty, only localhost is available

localhost | SUCCESS => {
   "intranet_mysql_db_user": "intranet"
}
```

Ansible vous fait saisir le mot de passe et procède ensuite à l'exécution du module `debug`.

### 2.3.2 Utilisation d'un fichier

L'utilisation d'un mot de passe de chiffrement entraîne la saisie de ce dernier à chaque lancement d'Ansible. En effet, il n'est pas possible d'indiquer ce dernier par une variable d'environnement. Heureusement, vous pouvez utiliser un fichier : il suffit pour cela de mettre le mot de passe seul dans un fichier.

Reste à préciser l'emplacement à Ansible à l'aide des options suivantes :

– L'option `--vault-password-file` suivie du nom du fichier contenant le mot de passe.

– En exportant la variable `DEFAULT_VAULT_PASSWORD_FILE` avec l'emplacement du fichier.

▶ Stockez le mot de passe « ansible » dans le fichier **vault.key** avec la commande suivante :

```
$ echo ansible > vault.key
```

Ci-dessous un exemple de déclaration de variable spécifiant l'emplacement du fichier contenant le mot de passe :

```
$ export DEFAULT_VAULT_PASSWORD_FILE=vault.key
```

- Vous pouvez maintenant chiffrer le fichier **intranet-pass.yml** avec la commande suivante :

```
$ ansible-vault encrypt intranet-pass.yml \
   --vault-password-file vault.key
```

Vous devriez obtenir le message suivant :

```
Encryption successful
```

Remarque

*Si le contenu de votre fichier est déjà chiffré, vous devriez obtenir le message suivant : ERROR! Input is already encrypted*

- Refaites le test précédent en ajoutant l'option `--vault-password-file` suivie du fichier contenant votre mot de passe :

```
$ ansible -m debug -a var=intranet_mysql_db_user localhost \
          -e @intranet-pass.yml --ask-vault-pass
```

Ci-dessous le résultat de votre commande :

```
localhost | SUCCESS => {
   "intranet_mysql_db_user": "intranet"
}
```

### 2.3.3 Déchiffrement d'un fichier

Vous souhaitez changer le contenu d'un fichier ? Vous avez à votre disposition plusieurs solutions avec `ansible-vault` :

– L'option `decrypt` qui va déchiffrer complètement le fichier pour vous permettre de modifier son contenu.

– L'option `edit` qui va éditer le fichier en fonction de la configuration de l'éditeur par défaut de votre système.

Remarque

*Vous pouvez changer l'éditeur par défaut pour la modification du fichier en modifiant la variable d'environnement `EDITOR` (ex. : `export EDITOR=vi` pour forcer l'utilisation de vi).*

Ci-dessous, un exemple d'utilisation de l'option `edit` avec `ansible-vault` :

```
$ ansible-vault edit intranet-pass.yml \
    --vault-password-file ./vault.key
```

Reste ensuite à effectuer votre modification comme vous l'auriez fait pour n'importe quel autre fichier.

### 2.3.4 Changement du mot de passe de chiffrement

Votre mot de passe ou le fichier de chiffrement a été compromis et vous souhaitez refaire votre chiffrement. Dans ce cas, il faudra passer l'option `rekey` à `ansible-vault`.

Ci-dessous, la commande à lancer :

```
$ ansible-vault rekey intranet-pass.yml
```

Il vous sera alors demandé l'ancien mot de passe suivi du nouveau comme indiqué ci-dessous :

```
Vault password:
New Vault password:
Confirm New Vault password:
Rekey successful
```

Dans le cas où vous feriez appel à des fichiers de mots de passe, vous serez intéressé d'apprendre l'existence des options `--new-vault-password-file` et `--vault-password-file`. Ci-dessous, un exemple de rechiffrement d'un fichier en passant par l'ancien fichier **vault.key** et le nouveau fichier **new-vault.key** :

```
$ ansible-vault rekey \
    --new-vault-password-file ./new-vault.key \
    --vault-password-file ./vault.key \
  intranet-pass.yml
```

Et ci-dessous le résultat en cas de succès :

```
Rekey successful
```

## 2.4 Chiffrement d'un champ

Le chiffrement d'un fichier est un moyen très efficace pour protéger vos identifiants et mots de passe dans un fichier YAML. En revanche, le fichier perd complètement les informations sur les variables qui s'y trouvent. Une solution intermédiaire (disponible depuis la version 2.3) est de passer par le chiffrement des champs d'un fichier. Vous gardez ainsi le nom de vos variables, mais leur contenu reste inconnu.

Pour cela `ansible-vault` offre l'option `encrypt_string` suivie de la chaîne à chiffrer.

Ci-dessous un exemple de lancement pour la chaîne `1ntran3t!` :

```
$ ansible-vault encrypt_string '1ntran3t!' \
    --vault-password-file ./vault.key
```

Ci-dessous, le résultat de cette commande :

```
!vault |
          $ANSIBLE_VAULT;1.1;AES256
          64663162646664[...]
          65633133396562[...]
          63323332396364[...]
          36383461366234[...]
          3934
Encryption successful
```

Reste maintenant à copier-coller ce résultat dans le fichier YAML **partial-vault.yml**. Ci-dessous le contenu que vous utiliserez dans votre exemple :

```
intranet_mysql_db_user: "intranet"
intranet_mysql_db_password: !vault |
           $ANSIBLE_VAULT;1.1;AES256
           64663162646664[...]
           65633133396562[...]
           63323332396364[...]
           36383461366234[...]
           3934
```

Remarque

*La présence de la chaîne `!vault` indique à Ansible un champ chiffré.*