LES VAULTS SUR ANSIBLE

Introduction

Dans le chapitre précédent, nous avions abouti à un Playbook modulable en utilisant le système de rôle proposé par Ansible. Néanmoins, on peut davantage améliorer notre playbook en utilisant le système de Vaults.

C'est quoi Ansible Vault ?

Ansible Vault (coffre-fort en français) est une fonctionnalité d'Ansible qui vous permet de **conserver des données sensibles** telles que des mots de passe, des clés SSH, de certificats SSL, des jetons d'API et tout ce que vous ne voulez pas que le public voie, plutôt que de les stocker sous un format brut dans des playbooks ou des rôles.

Comme il est courant de stocker des configurations Ansible dans un contrôleur de version tel que git, nous avons besoin d'un moyen de stocker ces données secrètes en toute sécurité. Le Vault est la réponse à cela, puisqu'il permet de chiffrer n'importe quoi à l'intérieur de votre fichier YAML, ces données de Vault peuvent ensuite être distribuées ou placées dans le contrôle de code source.

Nous utiliserons le système de Vault dans notre ancien projet afin de chiffrer le nouveau mot de passe root et le nom et le mot de passe du nouvel utilisateur. Commencez par télécharger notre ancien projet complet en cliquant ici.

Manipulation du Vault

Pour activer les fonctionnalités Vaults, il suffit d'utiliser l'outil de ligne de

commande `ansible-vault` afin d'utiliser ou modifier les fichiers de données secrètes.

Création notre fichiers chiffrés

Dans notre cas nous allons déplacer les variables `mysql_user`, `mysql password` et `root password` de notre fichier `vars/main.yml` vers notre nouveau fichier de variables secrètes, que nous créons tout de suite.

Placez-vous d'abord à la racine du projet et **créons un nouveau notre fichier de données chiffré**, en exécutant la commande suivante :

```
ansible-vault create vars/mysql-users.yml
```

Vous serez d'abord invité à saisir et confirmer un mot de passe (**retenez ce mot de passe, il est très important !**)

```
New Vault password: Confirm New Vault  password:
```

Après avoir fourni votre mot de passe, l'outil lancera l'éditeur que vous avez défini dans la variable d'environnement `$EDITOR` (par défaut vi). Voici les données remplîtes dans notre nouveau fichier secret:

```
---
mysql_user: "admin"
mysql_password: "Test_34535$"root_password:  "Test_34049$"
```

Voici à quoi ressemble le contenu de mon fichier chiffré quand je tente de l'afficher :

```
cat vars/mysql-users.yml
```

Résultat :

```
$ANSIBLE_VAULT;1.1;AES256
393165373534636365333034306439633065353866653263613639346135 66
626332616337663362
```

```
3661646139356433643736616537656266656137313632320a623032383533
353334373662623766
3633343530376235343436633537323061396663636364363434323465303365
366534313638636364
6664366330626632360a32356336666438633231323364346232643632316 3
373963363636303439
3563343762333238306635306464636338316236373731393964333039613032366431386538376 4
3762313666343361613264353736353663363033316532356264306566663632303061353161653 7
3361326261643863363965653634306138636539643832376230333861306236373863333438366 2
3436303839396665653038623733346434663432623834633931613132653764396362373733323 4
3437
```

Information

Le chiffrement par défaut est AES.

Édition de fichiers chiffrés

Pour **modifier un fichier chiffré** sur place, utilisez la commande suivante :

```
ansible-vault edit  vars/mysql-users.yml
```

Modifier le mot de passe des fichiers chiffrés

Si vous souhaitez **modifier votre mot de passe** sur un ou plusieurs fichiers chiffrés par le Vault, vous pouvez le faire avec la commande `rekey` :

```
ansible-vault rekey vars/mysql-users.yml
```

Vous pouvez aussi spécifier une liste de fichiers comme suit :

```
ansible-vault rekey fichier1.yml fichier2.yml  fichier3.yml
```

Dans ce cas le Vault Ansible vous demandera le mot de passe d'origine ainsi que le nouveau mot de passe pour chaque fichier.

Chiffrage de fichiers non chiffrés

Si vous souhaitez **chiffrer des fichiers bruts existants**, alors utilisez l'option `encrypt`.

Cette commande peut également fonctionner sur plusieurs fichiers à la fois:

```
ansible-vault encrypt  mon-fichier-non-chiffre.yml
```

Déchiffrement des fichiers déjà chiffrés

Si vous avez des fichiers existants que vous ne souhaitez plus conserver chiffrés, vous pouvez les **déchiffrer de façon permanente** en exécutant l'option `decrypt`:

```
ansible-vault decrypt  mon-fichier-deja-chiffre.yml
```

Affichage du contenu des fichiers chiffrés

Si vous souhaitez **afficher le contenu d'un fichier chiffré sans le modifier**, vous pouvez utiliser l'option `view` comme suit :

```
ansible-vault view vars/mysql-users.yml
```

Utilisez la touche "q" pour quitter le mot d'affichage.

Utilisation des fichiers chiffrés dans un Playbook

Il existe différents moyens pour lire vos variables chiffrées directement depuis vos playbooks. On peut utiliser la méthode classique en important notre fichier de variables chiffrées grâce à l'instruction `vars_files` . L'utilisation de cette méthode nous donnera le fichier playbook `playbook.yml` suivant :

```
---
# WEB SERVER
- hosts: web
  become: yes
  vars_files:
    - vars/main.yml
    - vars/mysql-users.yml
```

```
  roles:
    - web
# DATABASE SERVER
- hosts: db
  become: yes
  vars_files:
    - vars/main.yml
    - vars/mysql-users.yml
  roles:    - database
```

Lors du lancement de votre playbook utilisez l'option `--ask-vault-pass` afin d'avoir une saisie utilisateur pour fournir votre mot de passe Vault, comme suit :

```
ansible-playbook playbook.yml  --ask-vault-pass
```

Si vous ne souhaitez pas saisir le mot de passe Vault à chaque fois que vous exécutez votre playbook, vous pouvez ajouter votre mot de passe Vault à un fichier et référencer le fichier pendant l'exécution. Voici un exemple :

```
echo 'votre_mdp' > .vault_pass
```

Si vous utilisez un contrôleur de version (git), assurez-vous d'ajouter le fichier de mot de passe au fichier `.gitignore` afin de l'ignorer pendant votre commit :

```
echo '.vault_pass' >> .gitignore
```

Lors du lancement de votre playbook remplacez l'option `--ask-vault-pass` par l'option `--vault-password-file` comme suit :

```
ansible-playbook playbook.yml  --vault-password-file=.vault_pass
```

Vous pouvez aussi ignorer l'option `--ask-vault-pass` en rajoutant le fichier contenant votre mot de passe vault directement sur votre fichier ansible.cfg :

```
ansible.cfg
[defaults]
. . .vault_password_file = ./.vault_pass
```

Lancez ensuite votre playbook comme vous avez l'habitude de le faire :

```
ansible-playbook playbook.yml
```

Utilisation d'une variable chiffrée dans un Playbook

Une autre méthode d'utilisation d'une donnée secrète consiste à alimenter simplement une variable chiffrée dans l'instruction `vars` . Pour ce faire vous devez d'abord **créer votre variable secrète** avec l'option `encrypt_string` , comme suit :

```
ansible-vault encrypt_string 'secret-text' --name  'secret'
```

Résultat :

```
secret: !vault |
  $ANSIBLE_VAULT;1.1;AES256
  33616163316438306663033303763343638623434303634323435363138353231323539393764383
  34393337303535323466306261336664343639323031336 50a343137353735363138646536633432
  39303765306633643830353238646433356230623537363466373438323931353763666537386163
  30623033343438303 70a333939346632346134313063363136363038356266303835363331376631
  3664
```

Ansible vous demandera ensuite de saisir le mot de passe Vault pour chiffrer votre variable :

```
New Vault password: Confirm New Vault  password:
```

Information

Si vous avez déjà spécifié l'option `vault password file` dans le fichier `ansible.cfg`

, alors ansible nous vous demandera pas de saisir un mot de passe et utilisera le mot de passe de votre fichier Vault.

Ensuite il suffit de copier le résultat dans votre instruction `vars` comme suit :

```
---
- hosts: localhost
  vars:
    - secret: !vault |
          $ANSIBLE_VAULT;1.1;AES256
          336161633164383066633033303763343638623434303634323435363138353231323539393734393337303535323466306261336664343639323031336 50a343137353735363138646536639303765306633643830353238646433356230623537363466373438323931353763666537383062303334343830370a333939346632346134313063363136363038356266303835363331373664
  tasks:
    - debug: var=secret
    - fail:       when: 1 == 1
```

Exécutez ensuite votre playbook avec la commande habituelle :

```
ansible-playbook playbook.yml  --ask-vault-pass
```

Conclusion

Nous avons enfin réussi à rendre notre playbook modulable grâce à l'utilisation des rôles et sécurisé grâce à l'utilisation des Vaults Ansible. Vous retrouverez l'intégralité du projet depuis mon repository Github ici.