Capítulo 4
Inventarios: nociones avanzadas

1. Objetivos del capítulo y requisitos

1.1 Contexto y requisitos

En este capítulo se tratarán distintos temas concernientes a los inventarios. Verá, especialmente, cómo almacenar datos sensibles (nombres de usuario y contraseñas) en un archivo, así como la noción de inventario dinámico de Ansible.

Con respecto a los inventarios dinámicos, verá algunas aplicaciones sobre productos del mercado con las que podría encontrarse. No tendrá, seguramente, ganas de volver a escribir un inventario para Ansible y de mantenerlo ya que esta información está disponible en algún lugar (y que ya ha pasado, además, mucho tiempo trabajando en este tema).

Los ejemplos que se tratarán en este capítulo partirán de la base de que usted dispone de permisos de administración en sus máquinas.

Para la parte de la escritura dinámica del inventario, necesitará tener algunas nociones básicas de programación en Python.

1.2 Archivos descargables

Puede encontrar los ejemplos de los directorios inventarios y variables en el archivo comprimido **capitulo-04.tar.gz** que se encuentra en la página del libro en el sitio de Ediciones ENI.

2. Cifrado de archivos

2.1 Contexto

En el capítulo del descubrimiento del inventario, vio cómo administrar este último y cómo almacenar nombres de usuarios o contraseñas. Esto supone un problema ya que estos datos están almacenados en archivos de texto plano. Cualquier persona que tenga acceso a sus inventarios podría extraer este tipo de información y utilizarla.

En este capítulo verá un mecanismo que permite cifrar sus archivos con el objetivo de evitar que las contraseñas estén accesibles directamente.

Observación

Todos los ejemplos de ahora en adelante utilizarán la contraseña «ansible».

2.2 Almacenamiento de los nombres de usuario para las conexiones

La primera etapa consistirá en crear un archivo en el que almacenará la información necesaria para conectarse a una base de datos MySQL.

Para ello, usará dos campos: `intranet_mysql_db_user` y `intranet_mysql_db_password`.

Esos dos nombres de usuario le serán útiles para una hipotética aplicación en una intranet.

He aquí la declaración que usará como ejemplo:

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

2.3 Cifrado de un archivo entero

El cifrado de elementos en Ansible se hace gracias a la herramienta `ansible-vault`. Esta puede funcionar con una contraseña o utilizando archivos como argumento.

En los dos casos, para el cifrado de un archivo entero, hay que ejecutar `ansible-vault` con la opción `encrypt` (cifrar en inglés) seguida del nombre del archivo.

Observación

En todos los casos en que Ansible utilice archivos, estos pueden estar cifrados. Usted puede, por ejemplo, cifrar todos los archivos en local antes de copiarlos remotamente.

2.3.1 Usando una contraseña

Aquí encontrará el comando que habría que ejecutar para cifrar el archivo **intranet-pass.yml**:

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

Ansible le pedirá introducir una contraseña que tendrá que volver a confirmar una segunda vez. Aquí encontrará el resultado del comando cuando el cifrado se desarrolla correctamente:

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

Y aquí un extracto del archivo **intranet-pass.yml** después del cifrado:

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

El uso del archivo se hará como siempre con la opción `-e @archivo-vault.yml`. Para indicar a Ansible que hay que descifrar los archivos `vault`, tendrá que añadir la opción `--ask-vault-pass`. He aquí la carga del archivo **intranet-pass.yml** combinado con el parámetro `debug` para mostrar el contenido de la variable `intranet_mysql_db_user`:

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

Y aquí el resultado del comando:

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

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

Ansible le pide que introduzca la contraseña y procede a la ejecución del módulo `debug`.

2.3.2 Uso de un archivo

El uso de una contraseña de cifrado conlleva la introducción de esta cada vez que ejecute Ansible. En efecto, no es posible indicar la contraseña con una variable de entorno. Afortunadamente puede utilizar un archivo: para ello basta con poner la contraseña, y solamente ella, en un archivo.

Solo nos queda precisar la ubicación de este archivo a Ansible, usando las opciones siguientes:

– La opción `--vault-password-file` seguida del nombre del archivo que contiene la contraseña.
– Exportando la variable `DEFAULT_VAULT_PASSWORD_FILE` con la ubicación del archivo.

▶Almacene la contraseña «ansible» en el archivo **vault.key** con este comando:

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

He aquí un ejemplo de declaración de la variable que especifica la ubicación del archivo que contiene la contraseña:

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

▶Ahora puede cifrar el archivo **intranet-pass.yml** ejecutando el comando:

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

Debería tener el resultado siguiente:

```
Encryption successful
```

Observación

Si el contenido de su archivo ya está cifrado, debería obtener el mensaje siguiente: ERROR! Input is already encrypted

▶Haga de nuevo el test anterior añadiendo la opción `--vault-password-file` seguida del archivo que contiene la contraseña:

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

He aquí el resultado de su comando:

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

2.3.3 Descifrado de un archivo

¿Quiere cambiar el contenido de un archivo? Tiene a su disposición distintas soluciones con el comando `ansible-vault`:

– La opción `decrypt` que descifrará completamente el archivo para poder modificar el contenido.

– La opción `edit` que editará el archivo en función de la configuración del editor por defecto de su sistema.

Observación

Puede cambiar el editor por defecto para la modificación del archivo operando sobre la variable de entorno `EDITOR` (ej.: `export EDITOR=vi` para forzar el uso de vi).

He aquí un ejemplo de uso de la opción `edit` con `ansible-vault`:

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

Solamente quedaría efectuar la modificación como lo hubiera hecho para cualquier otro archivo.

2.3.4 Cambio de la contraseña de cifrado

Su contraseña o el archivo de cifrado ha sido comprometido y quiere volver a hacer el cifrado. En ese caso, tendrá que utilizar la opción `rekey` en `ansible-vault`.

He aquí el comando que hay que ejecutar:

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

Se le va a pedir la antigua contraseña seguida de la nueva como puede ver aquí:

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

En el caso de que usted utilice archivos de contraseña, le interesará saber de la existencia de las opciones `--new-vault-password-file` y `--vault-password-file`. He aquí un ejemplo de recifrado de un archivo utilizando el antiguo archivo **vault.key** y el nuevo archivo **new-vault.key**:

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

En el caso de que el comando se realice satisfactoriamente:

```
Rekey successful
```

2.4 Cifrado de un campo

El cifrado de un archivo es una manera muy eficaz de proteger sus nombres de usuario y contraseñas en un archivo YAML. Sin embargo, el archivo pierde completamente la información sobre las variables que se encuentran en su interior. Una solución intermedia (disponible desde la versión 2.3) sería cifrar los campos de un archivo. De esta manera, se guardarían los nombres de las variables, pero su contenido estaría oculto.

Para esto `ansible-vault` ofrece la opción `encrypt_string` seguida de la cadena que se quiere cifrar.

He aquí un ejemplo de la ejecución para la cadena `1ntran3t!`:

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

Y aquí el resultado del comando:

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

Ahora solo queda copiar y pegar el resultado en el archivo YAML **partial-vault.yml**. Aquí encontrará el contenido que utilizará en su ejemplo:

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

Observación

La presencia de la cadena `!vault` indica a Ansible la existencia de un campo cifrado.