Red Hat Enterprise Linux Automation with Ansible

Managing Secrets

Objectives

After completing this section, you should be able to encrypt sensitive variables using Ansible Vault, and run playbooks that reference Vault-encrypted variable files.

Introducing Ansible Vault

Ansible may need access to sensitive data such as passwords or API keys in order to configure managed hosts. Normally, this information might be stored as plain text in inventory variables or other Ansible files. In that case, however, any user with access to the Ansible files or a version control system which stores the Ansible files would have access to this sensitive data. This poses an obvious security risk.

Ansible Vault, which is included with Ansible, can be used to encrypt and decrypt any structured data file used by Ansible. To use Ansible Vault, a command-line tool named `ansible-vault` is used to create, edit, encrypt, decrypt, and view files. Ansible Vault can encrypt any structured data file used by Ansible. This might include inventory variables, included variable files in a playbook, variable files passed as arguments when executing the playbook, or variables defined in Ansible roles.

> **IMPORTANT**
>
> Ansible Vault does not implement its own cryptographic functions but rather uses an external Python toolkit. Files are protected with symmetric encryption using AES256 with a password as the secret key. Note that the way this is done has not been formally audited by a third party.

Creating an Encrypted File

To create a new encrypted file, use the `ansible-vault create` *`filename`* command. The command prompts for the new vault password and then opens a file using the default editor, `vi`. You can set and export the `EDITOR` environment variable to specify a different default editor by setting and exporting. For example, to set the default editor to `nano`, `export EDITOR=nano`.

```
[student@demo ~]$ ansible-vault create secret.yml
New Vault password: redhat
Confirm New Vault password: redhat
```

Instead of entering the vault password through standard input, you can use a vault password file to store the vault password. You need to carefully protect this file using file permissions and other means.

```
[student@demo ~]$ ansible-vault create --vault-password-file=vault-pass secret.yml
```

The cipher used to protect files is AES256 in recent versions of Ansible, but files encrypted with older versions may still use 128-bit AES.

Viewing an Encrypted File

You can use the `ansible-vault view` *`filename`* command to view an Ansible Vault-encrypted file without opening it for editing.

```
[student@demo ~]$ ansible-vault view secret1.yml
Vault password: secret
my_secret: "yJJvPqhsiusmmPPZdnjndkdnYNDjdj782meUZcw"
```

Editing an Existing Encrypted File

To edit an existing encrypted file, Ansible Vault provides the `ansible-vault edit` *`filename`* command. This command decrypts the file to a temporary file and allows you to edit it. When saved, it copies the content and removes the temporary file.

```
[student@demo ~]$ ansible-vault edit secret.yml
Vault password: redhat
```

> **NOTE**
>
> The `edit` subcommand always rewrites the file, so you should only use it when making changes. This can have implications when the file is kept under version control. You should always use the `view` subcommand to view the file's contents without making changes.

Encrypting an Existing File

To encrypt a file that already exists, use the `ansible-vault encrypt` *`filename`* command. This command can take the names of multiple files to be encrypted as arguments.

```
[student@demo ~]$ ansible-vault encrypt secret1.yml secret2.yml
New Vault password: redhat
Confirm New Vault password: redhat
Encryption successful
```

Use the `--output=OUTPUT_FILE` option to save the encrypted file with a new name. You can only use one input file with the `--output` option.

Decrypting an Existing File

An existing encrypted file can be permanently decrypted by using the `ansible-vault decrypt` *`filename`* command. When decrypting a single file, you can use the `--output` option to save the decrypted file under a different name.

```
[student@demo ~]$ ansible-vault decrypt secret1.yml --output=secret1-decrypted.yml
Vault password: redhat
Decryption successful
```

Changing the Password of an Encrypted File

You can use the `ansible-vault rekey` *`filename`* command to change the password of an encrypted file. This command can rekey multiple data files at once. It prompts for the original password and then the new password.

```
[student@demo ~]$ ansible-vault rekey secret.yml
Vault password: redhat
New Vault password: RedHat
Confirm New Vault password: RedHat
Rekey successful
```

When using a vault password file, use the `--new-vault-password-file` option:

```
[student@demo ~]$ ansible-vault rekey \
> --new-vault-password-file=NEW_VAULT_PASSWORD_FILE secret.yml
```

Playbooks and Ansible Vault

To run a playbook that accesses files encrypted with Ansible Vault, you need to provide the encryption password to the `ansible-playbook` command. If you do not provide the password, the playbook returns an error:

```
[student@demo ~]$ ansible-playbook site.yml
ERROR: A vault password must be specified to decrypt vars/api_key.yml
```

To provide the vault password to the playbook, use the `--vault-id` option. For example, to provide the vault password interactively, use `--vault-id @prompt` as illustrated in the following example:

```
[student@demo ~]$ ansible-playbook --vault-id @prompt site.yml
Vault password (default): redhat
```

IMPORTANT

If you are using a release of Ansible earlier than version 2.4, you need to use the `--ask-vault-pass` option to interactively provide the vault password. You can still use this option if all vault-encrypted files used by the playbook were encrypted with the same password.

```
[student@demo ~]$ ansible-playbook --ask-vault-pass site.yml
Vault password: redhat
```

Alternatively, you can use the `--vault-password-file` option to specify a file that stores the encryption password in plain text. The password should be a string stored as a single line in the file. Because that file contains the sensitive plain text password, it is vital that it be protected through file permissions and other security measures.

```
[student@demo ~]$ ansible-playbook --vault-password-file=vault-pw-file site.yml
```

You can also use the `ANSIBLE_VAULT_PASSWORD_FILE` environment variable to specify the default location of the password file.

IMPORTANT

Starting with Ansible 2.4, you can use multiple Ansible Vault passwords with `ansible-playbook`. To use multiple passwords, pass multiple `--vault-id` or `--vault-password-file` options to the `ansible-playbook` command.

```
[student@demo ~]$ ansible-playbook \
> --vault-id one@prompt --vault-id two@prompt site.yml
Vault password (one):
Vault password (two):
...output omitted...
```

The vault IDs `one` and `two` preceding `@prompt` can be anything and you can even omit them entirely. If you use the `--vault-id` *`id`* option when you encrypt a file with `ansible-vault` command, however, when you run `ansible-playbook` then the password for the matching ID is tried before any others. If it does not match, the other passwords you provided will be tried next. The vault ID `@prompt` with no ID is actually shorthand for `default@prompt`, which means to prompt for the password for vault ID `default`.

Recommended Practices for Variable File Management

To simplify management, it makes sense to set up your Ansible project so that sensitive variables and all other variables are kept in separate files. The files containing sensitive variables can then be protected with the `ansible-vault` command.

Remember that the preferred way to manage group variables and host variables is to create directories at the playbook level. The `group_vars` directory normally contains variable files with names matching host groups to which they apply. The `host_vars` directory normally contains variable files with names matching host names of managed hosts to which they apply.

However, instead of using files in `group_vars` or `host_vars`, you also can use directories for each host group or managed host. Those directories can then contain multiple variable files, all of which are used by the host group or managed host. For example, in the following project directory for `playbook.yml`, members of the `webservers` host group uses variables in the `group_vars/webservers/vars` file, and `demo.example.com` uses the variables in both `host_vars/demo.example.com/vars` and `host_vars/demo.example.com/vault`:

```
.
├── ansible.cfg
├── group_vars
│   └── webservers
│       └── vars
├── host_vars
│   └── demo.example.com
│       ├── vars
│       └── vault
├── inventory
└── playbook.yml
```

In this scenario, the advantage is that most variables for `demo.example.com` can be placed in the `vars` file, but sensitive variables can be kept secret by placing them separately in the `vault` file. Then the administrator can use `ansible-vault` to encrypt the `vault` file, while leaving the `vars` file as plain text.

There is nothing special about the file names being used in this example inside the `host_vars/demo.example.com` directory. That directory could contain more files, some encrypted by Ansible Vault and some which are not.

Playbook variables (as opposed to inventory variables) can also be protected with Ansible Vault. Sensitive playbook variables can be placed in a separate file which is encrypted with Ansible Vault and which is included in the playbook through a `vars_files` directive. This can be useful, because playbook variables take precedence over inventory variables.

If you are using multiple vault passwords with your playbook, make sure that each encrypted file is assigned a vault ID, and that you enter the matching password with that vault ID when running the playbook. This ensures that the correct password is selected first when decrypting the vault-encrypted file, which is faster than forcing Ansible to try all the vault passwords you provided until it finds the right one.

REFERENCES

`ansible-playbook`(1) and `ansible-vault`(1) man pages

Encrypting content with Ansible Vault — Ansible Documentation (https://docs.ansible.com/ansible/2.9/user_guide/vault.html)

Keep vaulted variables safely visible — Ansible Documentation (https://docs.ansible.com/ansible/2.9/user_guide/playbooks_best_practices.html#keep-vaulted-variables-safely-visible)

← PREVIOUS (/ROL/APP/COURSES/RH294-8.4/PAGES/CH03S02) → NEXT (/ROL/APP/COURSES/RH294-8.4/PAGES/CH03S04)

All policies and guidelines (https://www.redhat.com/en/about/all-policies-guidelines)

Release Notes (https://learn.redhat.com/t5/Red-Hat-Learning-Subscription/Red-Hat-Learning-Subscription-Release-Notes/ba-p/22952)

Cookie Preferences