**1. Ansible vault encrypt file**

With ansible-vault you can encrypt any structured data file used by Ansible. So ansible vault encrypt file can encrypt files or playbooks used by Ansible. You can check man page of ansible-vault to get the list of supported arguments

The --ask-vault-pass and --vault-password-file options can be used as long as only a single password is needed for any given run. We will use this in upcoming chapters.  
Alternatively the --vault-id option can be used to provide the password and indicate which vault label it’s for. Here ansible vault encrypt file is performed using default vault-id label:

[ansible@controller base]$ ansible-vault create --vault-id @prompt secret.yml

New vault password (default):

Confirm new vault password (default):

Once the passphrase is entered, ansible vault encrypt file opens using default editor and we're able to put content into the file, as shown below:

---

- name: This is a secret file

hosts: server2.example.com

tasks:

- name: Execute command 'date'

command: date

Now, we save our playbook file. Post ansible vault encrypt file if we try to read the contents, we'll see that they are in fact encrypted, with a small header hint for Ansible to use later

[ansible@controller base]$ cat secret.yml

$ANSIBLE\_VAULT;1.1;AES256

61383366323932306663616165376432623838373662636464313136373833313533393961623139

3433326563386161393130626539643763643439656236320a646365363666396562626563333032

34623939363131366337643131653564646437353130666636393033393866346361646666326361

3239633533386466360a653436666665333365346462373935383030326563313263663962663866

66336532633662633930363864343537353265323565333535363634303038653133383465333237

34316431666139633530323566303636353437343362303961613563316261353532356663623064

35313735626234363131626662326463656163393630386232646561653233336536656431613832

39353837313531623934366663333662663966396237386337373235666237303766316431616363

65663635316631383539393935336636393838363033323831386163373265663935623734353563

6566306531623365623830623437363939376366386465383566

As you can see from the headers, **AES256** is used for Vault encryption at present, meaning that as long as you use a good password when creating your Vault, your data is very secure.

Advertisement

**2. Ansible vault view encrypted files**

As you see now we cannot see the content of our encrypted file. So to view the encrypted file content using ansible vault use ansible-vault view command with the playbook file as shown in the below ansible vault example:

[ansible@controller base]$ ansible-vault view --vault-id @prompt secret.yml

Vault password (default):

---

- name: This is a secret file

hosts: server2.example.com

tasks:

- name: Execute command 'date'

command: date

**3. Ansible vault edit encrypted files**

Once a file has been encrypted with ansible-vault, it cannot be directly edited. Opening the file in an editor would result in the encrypted data being shown. Making any changes to the file would damage the file and Ansible would be unable to read the contents correctly. We must use ansible vault edit to first decrypt the contents of a file, allow us to edit those contents, and then encrypt the new contents before saving it back to the file.

To edit our ansible vault encrypt file secret.yml, use ansible-vault edit with the playbook file which will prompt for the password before decrypting and opening the editor

[ansible@controller base]$ ansible-vault edit secret.yml

Vault password:

ansible-vault opens uses default editor with a temporary file as the file path. The editor will save this, and then ansible-vault will encrypt it and move it to replace the original

**4. Use ansible vault password file**

In our earlier ansible vault example we used vault-id to encrypt the playbook, now here we will use ansible vault password file method for encryption. You can create a plain text file and provide your password in the file which we will further use to ansible vault encrypt file.

**HINT:**

This is again not so secure to use ansible vault password file as you will end up creating a plain text password file which can be read by others unless you restrict the permissions. To overcome this you can also create hidden files under hidden directories to keep it safe from unknown users.

Here in this ansible vault example I am storing my password (mypassword) in a temporary password\_file

[ansible@controller base]$ echo "mypassword" > password\_file

Now you can use this password file to create or edit ansible vault encrypt file.

[ansible@controller base]$ ansible-vault edit --vault-id password\_file secret.yml

OR

[ansible@controller base]$ ansible-vault edit --vault-password-file password\_file secret.yml

You can use either directive with ansible vault password file to create/edit/view the playbook file.

**5. Ansible vault example to encrypt existing file**

The previous ansible vault examples all dealt with creating new encrypted files using the create subcommand. But what if we want to encrypt an existing playbook file? You can use ansible-vault encrypt in such scenario:

I have an existing un-encrypted playbook file secret\_conditonal.yml which I wish to ansible vault encrypt file by prompting for the new password. Here I am not using any ansible vault password file, instead I will use ansible-vault with default label:

[ansible@controller base]$ ansible-vault encrypt --vault-id @prompt secret\_conditonal.yml

New vault password (default):

Confirm new vault password (default):

Encryption successful

So our ansible vault encrypt file was successful. Next you can try to check the content of the file, it should be encrypted:

[ansible@controller base]$ cat secret\_conditonal.yml

$ANSIBLE\_VAULT;1.1;AES256

33303661356362363565383834623763353431346135393634366164613733633639393338313333

3865353734336139653266386431356537663033363565620a643763366666323462353365653634

34323338363534663063336633626433653266376633623039353261303261616137343266336133

6430646335633065360a393637663738363739346365323932623332653631356261643865376432

61663836373262373631633733363831393937363461373938326533366231663734653139336436

33326564666437363462636163313466363733396636666361393464636363353739363331653936

63386638623261343662633764333931643431303439306531313437333066303936623466316433

3939643536366463343362383937656330313832333064653934

**6. Ansible vault change password of encrypted files**

Over time, as contributors come and go, it is a good idea to rotate the password used to encrypt your secrets. Encryption is only as good as the protection of the password. ansible-vault provides a subcommand that allows us to change the password named rekey

You can use below command to get the list of available supported parameters with "rekey"

[ansible@controller base]$ ansible-vault rekey --help

We use --ask-vault-pass to prompt ansible-vault for password change.

[ansible@controller base]$ ansible-vault rekey --ask-vault-pass secret.yml

Vault password:

New Vault password:

Confirm New Vault password:

Rekey successful

So we successfully changed our ansible-vault password for secret.yml playbook file

**7. Ansible vault decrypt file**

You can also use ansible vault decrypt file to decrypt playbook files in Ansible. If at some point, the need to encrypt data files goes away, you can use ansible-vault decrypt subcommand to remove encryption for one or more encrypted files. Instead of using ansible vault password file I will use --ask-vault-pass. Check below ansible vault example:

[ansible@controller base]$ ansible-vault decrypt --ask-vault-pass secret.yml

Vault password:

Decryption successful

So we were able to ansible vault decrypt file here. Now you can use any editor to view the content of your playbook file.

**8. Using ansible vault in playbook**

* For using ansible vault in playbook, we need to be able to inform ansible-playbook how to access any encrypted data it might encounter.
* Unlike ansible-vault, which exists solely to deal with file encryption or decryption, ansible-playbook is more general-purpose, and it will not assume it is dealing with encrypted data by default.
* Luckily, all of our familiar --vault-id parameters from the previous examples work just the same in ansible-playbook as they do in ansible-vault and requires no special handling using ansible vault in playbook
* Ansible will hold the provided passwords and IDs in memory for the duration of the playbook execution.

I have a secret.yml playbook file with below content

---

- name: This is a secret file

hosts: server2.example.com

tasks:

- name: Execute command 'date'

command: date

In the below ansible vault example, We will execute this playbook with "--vault-id @prompt" which will prompt us with the password for the yml file before processing the playbook.

[ansible@controller base]$ ansible-playbook --vault-id @prompt secret.yml

Vault password (default):

PLAY [This is a secret file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Gathering Facts] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [server2.example.com]

TASK [Execute command 'date'] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [server2.example.com]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

server2.example.com : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

So here we are using ansible vault in playbook, after we provide the password ansible-playbook will process the yml file tasks.

**9. Using Ansible playbook with vault password file**

We know Ansible is all about automating things but with the above example, ansible-playbook prompts for user input to get the password so this can be a problem with some org. To overcome you can create a plan text hidden file which contains the password of the playbook. For example I have created a text file password\_file which contains my password

[ansible@controller base]$ echo "password" > password\_file

Now I can re-reun my ansible playbook with this vault password file using the below syntax

[ansible@controller base]$ ansible-playbook secret.yml --vault-password-file=password\_file

**10. Ansible vault encrypt string**

Before the release of Ansible 2.3, secure data had to be encrypted in a separate file. Now we don't need to encrypt entire file and instead we can only ansible vault encrypt string such as passwords from the playbook file. You can use encrypt\_string subcommand of ansible-vault, which produces an encrypt string that can be placed into an Ansible YAML file.

In the below ansible vault example we have a sample playbook file where we use ansible vault encrypt string

[ansible@controller base]$ cat secret1.yml

---

- name: inline secret variable demonstration

hosts: server1.example.com

gather\_facts: false

vars:

my\_secret: secure\_password

tasks:

- name: print the secure variable

debug:

var: my\_secret

Now here in the YML file we have provided the password secure\_password in plain text format. Anybody can open this yml file and get the password information which is insecure. So we can either encrypt the entire yml file or we also have an option to encrypt only the secure\_password string using ansible vault encrypt string.

To encrypt secure\_password string use the below command, again here we use default vault-id label:

[ansible@controller base]$ ansible-vault encrypt\_string --vault-id @prompt secure\_password

New vault password (default):

Confirm new vault password (default):

!vault |

$ANSIBLE\_VAULT;1.1;AES256

32373330386330643061613237393466393363333031303965383063316338393261616134353233

6364623038303966323834396636646463613837373461380a303662373161393466326139383565

64396538393562343464666337633337353130306365666637373266393965633766366436623836

3430396234626533340a613637393563333131626665626535393462653139346563383062343535

6335

Encryption successful

The ansible vault encrypt string is successful. This gives us the encryption key for secure\_password string which we will use in our ansible vault example playbook file in the below format

[ansible@controller base]$ cat secret1.yml

---

- name: inline secret variable demonstration

hosts: server1.example.com

gather\_facts: false

vars:

my\_secret: !vault |

$ANSIBLE\_VAULT;1.1;AES256

32373330386330643061613237393466393363333031303965383063316338393261616134353233

6364623038303966323834396636646463613837373461380a303662373161393466326139383565

64396538393562343464666337633337353130306365666637373266393965633766366436623836

3430396234626533340a613637393563333131626665626535393462653139346563383062343535

6335

tasks:

- name: print the secure variable

debug:

var: my\_secret

As you see I have replaced secure\_password string with the encrypted key we got from ansible vault encrypt string command above:

Now you can re-reun this ansible using ansible vault in playbook:

[ansible@controller base]$ ansible-playbook --vault-id @prompt secret1.yml

Vault password (default):

PLAY [inline secret variable demonstration] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [print the secure variable] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [server1.example.com] => {

"my\_secret": "secure\_password"

}

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

server1.example.com : ok=1 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

Note