# Ansible Vault

A typical Ansible setup will involve needing some sort of secret to fully setup a server or application.
Common types of "secret" include passwords, SSH keys, SSL certificates, API tokens and anything else you don't want the public to see.

Since it's common to store Ansible configurations in version control, we need a way to store secrets securely.

Ansible Vault is the answer to this. Ansible Vault can encrypt anything inside of a YAML file, using a password of your choice.

### Using Ansible Vault

A typical use of Ansible Vault is to encrypt variable files. Vault can encrypt any YAML file, but the most common files to encrypt are:

1. Files within the group_vars directory.
2. A role's defaults/main.yml file
3. A role's vars/main.yml file.
4. Any other file used to store variables.

Let's see how to use Ansible Vault with some variable files.

Let’s take below host inventory file we have defined host details along with username & password to connect to the host. If you observe password is visible to everyone. As per standards we should not expose passwords to everyone.

```
[ansible@ip-172-31-21-155 project]$ cat hosts
172.31.29.109 ansible_user=ansible ansible_ssh_pass=DevOps@2018
```

**We have two options to here.**

1) Encrypt complete host inventory file using ansible vault. But this is not suggestable as we are encrypting complete file to encrypt password.
2) Create group variables or host variables file and encrypt using ansible vault. And refer variable in host inventory from group/host variables file.

**Step1:** Create group variables for all groups.

```
[ansible@ip-172-31-21-155 project]$ mkdir group_vars
[ansible@ip-172-31-21-155 project]$ vi group_vars/all.yml
```

Add your password (Key value pair) in group variables yml

```
[ansible@ip-172-31-21-155 project]$ cat group_vars/all.yml
ansible_pwd: DevOps@2018
[ansible@ip-172-31-21-155 project]$
```

**Step 2:** Encrypt existing group variables yml file using ansible vault.

The typical use case is to have a normal, plaintext variable file that we want to encrypt. Using ansible-vault, we can encrypt this and define the password needed to later decrypt it:

```
[ansible@ip-172-31-21-155 project]$ ansible-vault encrypt group_vars/all.yml
New Vault password:
Confirm New Vault password:
Encryption successful
[ansible@ip-172-31-21-155 project]$
```

If see the content the complete file is encrypted. Content is not in human readable format.

```
[ansible@ip-172-31-21-155 project]$ cat group_vars/all.yml
$ANSIBLE_VAULT;1.1;AES256
36306339363662323338336335313861633862653937386538343938306238636531383364616363
66636235363535363034616331363561663030623864646 60a393034373663623165643932616562
66613364623731656665306163626432303762643833663961306661356134333831643231613163
37313864653565316 50a32386334616438326133636430323461653134333339663239306131373 3
65323466633261653261646664353161653233643633303136376463393634633361
[ansible@ip-172-31-21-155 project]$
```

**Step 3:** Update your host inventory to refer password from group_vars/all.yml file make sure you use the same key name which you defined in group_vars/all.yml.

```
[ansible@ip-172-31-21-155 project]$ cat hosts
172.31.29.109 ansible_user=ansible ansible_ssh_pass={{ansible_pwd}}
[ansible@ip-172-31-21-155 project]$
```

**Step 4:** Execute ansible adhoc command ping to test the connectivity.

```
2. ansible@ip-172-31-21-155:~/proje
[ansible@ip-172-31-21-155 project]$ ansible -i hosts -m ping all
ERROR! Attempting to decrypt but no vault secrets found
[ansible@ip-172-31-21-155 project]$
```

We will get error since we are referring password from group_vars/all.yml which is encrypted using vault. So, while executing play book or adhoc commands we must pass ansible vault password what ever You have typed in while encrypting.

Use below command to execute.

```
[ansible@ip-172-31-21-155 project]$ ansible -i hosts -m ping all --ask-vault-pass
Vault password:
172.31.29.109 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[ansible@ip-172-31-21-155 project]$
```

**Go through below to know more about other ansible vault commands or options.**

### Encrypting an Existing File

The typical use case is to have a normal, plaintext variable file that we want to encrypt. Using ansible-vault, we can encrypt this and define the password needed to later decrypt it:

```
# Encrypt a role's defaults/main.yml file
ansible-vault encrypt defaults/main.yml

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

The ansible-vault command will prompt you for a password twice (a second time to confirm the first). Once that's done, the file will be encrypted! If you edit the file directly, you'll just see encrypted text. It looks something like this:

```
$ANSIBLE_VAULT;1.1;AES256
653262333637316636316461343065633532366533386464333438383734373734303764
464616339
333338323337346535313132323763653836336131643138 0a643336643862663739623
631616530
353563616264346530663166613738633133623961626463653431666462316531653033
431636139
623036616436313834 0a356631633930323032653466626531383261613539633365366
631623238
323966376238666331353632313466643037303532306234396336663866623464323633
164393438
336536663730643262333733373839343163353038623138383839666231346462303463
330303136
663332323630623038373335333031303862383231656236323462393835383434376633
437373730
35666532333065383439
```

### Creating an Encrypted File

If you are creating a new file instead of encrypting an existing one, you can use the create command:

```
ansible-vault create defaults/secrets.yml
> New Vault password:
> Confirm New Vault password:
```

### Editing a File

Once you encrypt a file, you can only edit the file by using ansible-vault again (unless there's an editor out there that can integrate with Vault to let you edit in the IDE?!). Here's how to edit that file after it's been encrypted:

```
ansible-vault edit defaults/main.yml
> Vault password:
```

This will ask for the password used to encrypt the file.

> You'll lose your data if you lose your password!

Since we're running these commands in the CLI, this will open the file in a terminal-based app. This usually means your default editor as defined the EDITOR environment variable, if that is set:

```
echo $EDITOR
> vim
```

My EDITOR environment variable isn't set, but my default is Vim. To use Nano instead of Vim, you can set that variable while running Vault:

```
EDITOR=nano ansible-vault edit defaults/main.yml
```

### Decrypting a File

You can decrypt a file to get it back to plaintext as well:

```
ansible-vault decrypt defaults/main.yml
> Vault password:
```

### Encrypting Specific Variables

You don't have to encrypt a whole file! This is nice to track changes better in git, where you don't have an entire file changing for just a small change (even just opening an encrypted file will change the encrypted hash).

The most basic use case is to just run it interactively on the CLI to get the properly formatted YAML as output:

```
ansible-vault encrypt_string
> New Vault password:
> Confirm New Vault password:
> Reading plaintext input from stdin. (ctrl-d to end input)
> this is a plaintext string
> !vault |
>          $ANSIBLE_VAULT;1.1;AES256
> 393937666637616533373864366364663965313532613832376135313565313439306633
133623839
> 343661383430326461303862343230383739326166323364 0a36363334333762306561 3
166306363
> 373361323634623861383435353462643330616561346366313261646430353134333933
831616131
> 363561356537393931 0a316132313764356432333366396533663965333162336538663
432323334
> 336563653037333036643539613635633132363962623137393434613830363335 61
> Encryption successful
```

That string could be used in a variable file like so (as variable some string):

```
---
some_string: !vault |
    $ANSIBLE_VAULT;1.1;AES256

393937666637616533373864366364663965313532613832376135313565313439306633
133623839

343661383430326461303862343230383739326166323364 0a36363334333762306561 3
166306363

373361323634623861383435353462643330616561346366313261646430353134333933
831616131

363561356537393931 0a316132313764356432333366396533663965333162336538663
432323334

336563653037333036643539613635633132363962623137393434613830363335 61
```

You can do this in one line also:

```
ansible-vault encrypt_string 'this is a plaintext string' --name
'some_string'
> New Vault password:
> Confirm New Vault password:
> some_string: !vault |
>          $ANSIBLE_VAULT;1.1;AES256
>
343962326431333230346663353139396338653565343030643962386439393433376263
330666164
>
623130306137366632626438653866656437376266333231 0a32393862623936376334363
638353264
>
646462666633613866333863316561633534386230336266333666643035363961363533
834336364
>
636330353230326564 0a396264616562663963653034376462613035383333373437653
362616566
>          3531
> Encryption successful
```

The output can be copied/pasted or appended into an existing YAML file!

**Running Ansible with EncryptedVariables**

---

If your roles or playbooks reference encrypted variables, you need to have given Ansible the password to decrypt them. You can do this in two ways:
Ask for Vault Password

Have Ansible ask for the vault password as a prompt:

```
ansible-playbook --ask-vault-pass -i inventory_file some_playabook.yml
```

Using the --ask-vault-pass flag will instruct Ansible to ask for the vault password so it can decrypt the

variable files correctly.
Use a Vault File

Another handy thing you can do is store a vault password in a file and use that. This is handy for automation.
To do so, first create a file with a password:

```
echo "secret_password" > vault_password
```

Then you can reference that file with the --vault-password-file flag. This flag can be used with any ansible-playbook or ansible-vault command to pre-define the password, so you do not get a prompt:

```
# When creating/editing/encrypting/decrypting/rekeying a file:
ansible-vault --vault-password-file=vault_password create
defaults/foo.yml
ansible-vault --vault-password-file=vault_password edit
defaults/foo.yml
```

```
# When running ansible playbooks
ansible-playbook --vault-password-file=vault_password -i inventory_file
some_playabook.yml
```

**Issues**

---

There are only a few issues to really worry about:

1. If you lose your password, you lose your data. One common way to make this more manageable is to use a shared password manager.
2. If you're tracking your Ansible roles in git, you'll notice that even opening an encrypted file (and not changing it) will change the file; Merely opening a file means a new git commit. This is an annoying result of how the encryption is done when opening (decrypting) and closing (re-encrypting) a file for editing. This can be mitigated by encrypting only specific variables within a file as shown above.