[**Ansible Vault**](http://docs.ansible.com/ansible/latest/vault.html#id5)

**Topics**

* [Ansible Vault](http://docs.ansible.com/ansible/latest/vault.html#ansible-vault)
  + [What Can Be Encrypted With Vault](http://docs.ansible.com/ansible/latest/vault.html#what-can-be-encrypted-with-vault)
  + [Creating Encrypted Files](http://docs.ansible.com/ansible/latest/vault.html#creating-encrypted-files)
  + [Editing Encrypted Files](http://docs.ansible.com/ansible/latest/vault.html#editing-encrypted-files)
  + [Rekeying Encrypted Files](http://docs.ansible.com/ansible/latest/vault.html#rekeying-encrypted-files)
  + [Encrypting Unencrypted Files](http://docs.ansible.com/ansible/latest/vault.html#encrypting-unencrypted-files)
  + [Decrypting Encrypted Files](http://docs.ansible.com/ansible/latest/vault.html#decrypting-encrypted-files)
  + [Viewing Encrypted Files](http://docs.ansible.com/ansible/latest/vault.html#viewing-encrypted-files)
  + [Use encrypt\_string to create encrypted variables to embed in yaml](http://docs.ansible.com/ansible/latest/vault.html#use-encrypt-string-to-create-encrypted-variables-to-embed-in-yaml)
  + [Vault Ids and Multiple Vault Passwords](http://docs.ansible.com/ansible/latest/vault.html#vault-ids-and-multiple-vault-passwords)
  + [Providing Vault Passwords](http://docs.ansible.com/ansible/latest/vault.html#providing-vault-passwords)
    - [Multiple vault passwords](http://docs.ansible.com/ansible/latest/vault.html#multiple-vault-passwords)
  + [Speeding Up Vault Operations](http://docs.ansible.com/ansible/latest/vault.html#speeding-up-vault-operations)
  + [Vault Format](http://docs.ansible.com/ansible/latest/vault.html#vault-format)
  + [Vault Payload Format 1.1](http://docs.ansible.com/ansible/latest/vault.html#vault-payload-format-1-1)

New in Ansible 1.5, “Vault” is a feature of ansible that allows keeping sensitive data such as passwords or keys in encrypted files, rather than as plaintext in your playbooks or roles. These vault files can then be distributed or placed in source control.

To enable this feature, a command line tool - [ansible-vault](http://docs.ansible.com/ansible/latest/ansible-vault.html#ansible-vault) - is used to edit files, and a command line flag ([--ask-vault-pass](http://docs.ansible.com/ansible/latest/ansible-playbook.html#cmdoption-ansible-playbook-ask-vault-pass) or [--vault-password-file](http://docs.ansible.com/ansible/latest/ansible-playbook.html#cmdoption-ansible-playbook-vault-password-file)) is used. Alternately, you may specify the location of a password file or command Ansible to always prompt for the password in your ansible.cfg file. These options require no command line flag usage.

For best practices advice, refer to [Variables and Vaults](http://docs.ansible.com/ansible/latest/playbooks_best_practices.html#best-practices-for-variables-and-vaults).

[**What Can Be Encrypted With Vault**](http://docs.ansible.com/ansible/latest/vault.html#id6)

The vault feature can encrypt any structured data file used by Ansible. This can include “group\_vars/” or “host\_vars/” inventory variables, variables loaded by “include\_vars” or “vars\_files”, or variable files passed on the ansible-playbook command line with “-e @file.yml” or “-e @file.json”. Role variables and defaults are also included!

Ansible tasks, handlers, and so on are also data so these can be encrypted with vault as well. To hide the names of variables that you’re using, you can encrypt the task files in their entirety. However, that might be a little too much and could annoy your coworkers :)

The vault feature can also encrypt arbitrary files, even binary files. If a vault-encrypted file is given as the ‘src’ argument to the [copy](http://docs.ansible.com/ansible/latest/copy_module.html#copy), [template](http://docs.ansible.com/ansible/latest/template_module.html#template),[unarchive](http://docs.ansible.com/ansible/latest/unarchive_module.html#unarchive), [script](http://docs.ansible.com/ansible/latest/script_module.html#script) or [assemble](http://docs.ansible.com/ansible/latest/assemble_module.html#assemble) modules, the file will be placed at the destination on the target host decrypted (assuming a valid vault password is supplied when running the play).

As of version 2.3, Ansible also supports encrypting single values inside a YAML file, using the *!vault* tag to let YAML and Ansible know it uses special processing. This feature is covered in more details below.

[**Creating Encrypted Files**](http://docs.ansible.com/ansible/latest/vault.html#id7)

To create a new encrypted data file, run the following command:

ansible-vault create foo.yml

First you will be prompted for a password. The password used with vault currently must be the same for all files you wish to use together at the same time.

After providing a password, the tool will launch whatever editor you have defined with $EDITOR, and defaults to vi (before 2.1 the default was vim). Once you are done with the editor session, the file will be saved as encrypted data.

The default cipher is AES (which is shared-secret based).

[**Editing Encrypted Files**](http://docs.ansible.com/ansible/latest/vault.html#id8)

To edit an encrypted file in place, use the [ansible-vault edit](http://docs.ansible.com/ansible/latest/ansible-vault.html#ansible-vault-edit) command. This command will decrypt the file to a temporary file and allow you to edit the file, saving it back when done and removing the temporary file:

ansible-vault edit foo.yml

[**Rekeying Encrypted Files**](http://docs.ansible.com/ansible/latest/vault.html#id9)

Should you wish to change your password on a vault-encrypted file or files, you can do so with the rekey command:

ansible-vault rekey foo.yml bar.yml baz.yml

This command can rekey multiple data files at once and will ask for the original password and also the new password.

[**Encrypting Unencrypted Files**](http://docs.ansible.com/ansible/latest/vault.html#id10)

If you have existing files that you wish to encrypt, use the [ansible-vault encrypt](http://docs.ansible.com/ansible/latest/ansible-vault.html#ansible-vault-encrypt) command. This command can operate on multiple files at once:

ansible-vault encrypt foo.yml bar.yml baz.yml

[**Decrypting Encrypted Files**](http://docs.ansible.com/ansible/latest/vault.html#id11)

If you have existing files that you no longer want to keep encrypted, you can permanently decrypt them by running the [ansible-vault decrypt](http://docs.ansible.com/ansible/latest/ansible-vault.html#ansible-vault-decrypt)command. This command will save them unencrypted to the disk, so be sure you do not want [ansible-vault edit](http://docs.ansible.com/ansible/latest/ansible-vault.html#ansible-vault-edit) instead:

ansible-vault decrypt foo.yml bar.yml baz.yml

[**Viewing Encrypted Files**](http://docs.ansible.com/ansible/latest/vault.html#id12)

*Available since Ansible 1.8*

If you want to view the contents of an encrypted file without editing it, you can use the [ansible-vault view](http://docs.ansible.com/ansible/latest/ansible-vault.html#ansible-vault-view) command:

ansible-vault view foo.yml bar.yml baz.yml

[**Use encrypt\_string to create encrypted variables to embed in yaml**](http://docs.ansible.com/ansible/latest/vault.html#id13)

The [ansible-vault encrypt\_string](http://docs.ansible.com/ansible/latest/ansible-vault.html#ansible-vault-encrypt-string) command will encrypt and format a provided string into a format that can be included in [ansible-playbook](http://docs.ansible.com/ansible/latest/ansible-playbook.html#ansible-playbook)YAML files.

To encrypt a string provided as a cli arg:

ansible-vault encrypt\_string --vault-id a\_password\_file 'foobar' --name 'the\_secret'

Result:

some\_foo: **!vault** |

$ANSIBLE\_VAULT;1.1;AES256

62313365396662343061393464336163383764373764613633653634306231386433626436623361

6134333665353966363534333632666535333761666131620a663537646436643839616531643561

63396265333966386166373632626539326166353965363262633030333630313338646335303630

3438626666666137650a353638643435666633633964366338633066623234616432373231333331

6564

To use a vault-id label for ‘dev’ vault-id:

ansible-vault encrypt\_string --vault-id dev@password 'foooodev' --name 'the\_dev\_secret'

Result:

the\_dev\_secret: **!vault** |

$ANSIBLE\_VAULT;1.2;AES256;dev

30613233633461343837653833666333643061636561303338373661313838333565653635353162

3263363434623733343538653462613064333634333464660a663633623939393439316636633863

61636237636537333938306331383339353265363239643939666639386530626330633337633833

6664656334373166630a363736393262666465663432613932613036303963343263623137386239

6330

To encrypt a string read from stdin and name it ‘db\_password’:

echo 'letmein' | ansible-vault encrypt\_string --vault-id dev@password --stdin-name 'db\_password'

Result:

Reading plaintext input from stdin. (ctrl-d to end input)

db\_password: **!vault** |

$ANSIBLE\_VAULT;1.2;AES256;dev

61323931353866666336306139373937316366366138656131323863373866376666353364373761

3539633234313836346435323766306164626134376564330a373530313635343535343133316133

36643666306434616266376434363239346433643238336464643566386135356334303736353136

6565633133366366360a326566323363363936613664616364623437336130623133343530333739

3039

To be prompted for a string to encrypt, encrypt it, and give it the name ‘new\_user\_password’:

ansible-vault encrypt\_string --vault-id dev@./password --stdin-name 'new\_user\_password'

Output:

Reading plaintext input from stdin. (ctrl-d to end input)

User enters ‘hunter42’ and hits ctrl-d.

Result:

new\_user\_password: **!vault** |

$ANSIBLE\_VAULT;1.2;AES256;dev

37636561366636643464376336303466613062633537323632306566653533383833366462366662

6565353063303065303831323539656138653863353230620a653638643639333133306331336365

62373737623337616130386137373461306535383538373162316263386165376131623631323434

3866363862363335620a376466656164383032633338306162326639643635663936623939666238

3161

See also [Single Encrypted Variable](http://docs.ansible.com/ansible/latest/playbooks_vault.html#single-encrypted-variable)

[**Vault Ids and Multiple Vault Passwords**](http://docs.ansible.com/ansible/latest/vault.html#id14)

*Available since Ansible 2.4*

A vault id is an identifier for one or more vault secrets. Since Ansible 2.4, Ansible supports multiple vault passwords. Vault ids is a way to provide a label for a particular vault password.

Vault encrypted content can specify which vault id it was encrypted with.

Prior to Ansible 2.4, only one vault password could be used at a time. Post Ansible 2.4, multiple vault passwords can be used each time Ansible runs, so any vault files or vars that needed to be decrypted all had to use the same password.

Since Ansible 2.4, vault files or vars can be that are encrypted with different passwords can be used at the same time.

For example, a playbook can now include a vars file encrypted with a ‘dev’ vault id and a ‘prod’ vault id.

[**Providing Vault Passwords**](http://docs.ansible.com/ansible/latest/vault.html#id15)

Since Ansible 2.4, the recommended way to provide a vault password from the cli is to use the [--vault-id](http://docs.ansible.com/ansible/latest/ansible-playbook.html#cmdoption-ansible-playbook-vault-id) cli option.

For example, to use a password store in the text file /path/to/my/vault-password-file:

ansible-playbook --vault-id /path/to/my/vault-password-file site.yml

To prompt for a password:

ansible-playbook --vault-id @prompt site.yml

To get the password from a vault password executable script my-vault-password.py:

ansible-playbook --vault-id my-vault-password.py

The value for [--vault-id](http://docs.ansible.com/ansible/latest/ansible-playbook.html#cmdoption-ansible-playbook-vault-id) can specify the type of vault id (prompt, a file path, etc) and a label for the vault id (‘dev’, ‘prod’, ‘cloud’, etc)

For example, to use a password file dev-password for the vault-id ‘dev’:

ansible-playbook --vault-id dev@dev-password site.yml

To prompt for the ‘dev’ vault id:

ansible-playbook --vault-id dev@prompt site.yml

*Prior to Ansible 2.4*

To be prompted for a vault password, use the [--ask-vault-pass](http://docs.ansible.com/ansible/latest/ansible-playbook.html#cmdoption-ansible-playbook-vault-id) cli option:

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

To specify a vault password in a text file ‘dev-password’, use the [--vault-password-file](http://docs.ansible.com/ansible/latest/ansible-playbook.html#cmdoption-ansible-playbook-vault-password-file) option:

ansible-playbook --vault-password-file dev-password site.yml

There is a config option ([DEFAULT\_VAULT\_PASSWORD\_FILE](http://docs.ansible.com/ansible/latest/config.html#default-vault-password-file)) to specify a vault password file to use without requiring the [--vault-password-file](http://docs.ansible.com/ansible/latest/ansible-playbook.html#cmdoption-ansible-playbook-vault-password-file) cli option.

[**Multiple vault passwords**](http://docs.ansible.com/ansible/latest/vault.html#id16)

Since Ansible 2.4 and later support using multiple vault passwords, [--vault-id](http://docs.ansible.com/ansible/latest/ansible-playbook.html#cmdoption-ansible-playbook-vault-id) can be provided multiple times.

If multiple vault passwords are provided, by default Ansible will attempt to decrypt vault content by trying each vault secret in the order they were provided on the command line.

For example, to use a ‘dev’ password read from a file and to be prompted for the ‘prod’ password:

ansible-playbook --vault-id dev@dev-password --vault-id prod@prompt site.yml

In the above case, the ‘dev’ password will be tried first, then the ‘prod’ password for cases where Ansible doesn’t know which vault id is used to encrypt something.

If the vault content was encrypted using a [--vault-id](http://docs.ansible.com/ansible/latest/ansible-vault.html#cmdoption-ansible-vault-vault-id) option, then the label of the vault id is stored with the vault content. When Ansible knows the right vault-id, it will try the matching vault id’s secret first before trying the rest of the vault-ids.

There is a config option ([DEFAULT\_VAULT\_ID\_MATCH](http://docs.ansible.com/ansible/latest/config.html#default-vault-id-match) ) to force the vault content’s vault id label to match with one of the provided vault ids. But the default is to try the matching id first, then try the other vault ids in order.

There is also a config option ([DEFAULT\_VAULT\_IDENTITY\_LIST](http://docs.ansible.com/ansible/latest/config.html#default-vault-identity-list)) to specify a default list of vault ids to use. For example, instead of requiring the cli option on every use, the ([DEFAULT\_VAULT\_IDENTITY\_LIST](http://docs.ansible.com/ansible/latest/config.html#default-vault-identity-list)) config option can be used:

ansible-playbook --vault-id dev@dev-password --vault-id prod@prompt site.yml

The [--vault-id](http://docs.ansible.com/ansible/latest/ansible-playbook.html#cmdoption-ansible-playbook-vault-id) can be used in lieu of the [--vault-password-file](http://docs.ansible.com/ansible/latest/ansible-playbook.html#cmdoption-ansible-playbook-vault-password-file) or [--ask-vault-pass](http://docs.ansible.com/ansible/latest/ansible-playbook.html#cmdoption-ansible-playbook-ask-vault-pass) options, or it can be used in combination with them.

When using [ansible-vault](http://docs.ansible.com/ansible/latest/ansible-vault.html#ansible-vault) commands that encrypt content ([ansible-vault encrypt](http://docs.ansible.com/ansible/latest/ansible-vault.html#ansible-vault-encrypt), [ansible-vault encrypt\_string](http://docs.ansible.com/ansible/latest/ansible-vault.html#ansible-vault-encrypt-string), etc) only one vault-id can be used.

**Note**

Prior to Ansible 2.4, only one vault password could be used in each Ansible run. The [--vault-id](http://docs.ansible.com/ansible/latest/ansible-playbook.html#cmdoption-ansible-playbook-vault-id) option is not support prior to Ansible 2.4.

[**Speeding Up Vault Operations**](http://docs.ansible.com/ansible/latest/vault.html#id17)

By default, Ansible uses PyCrypto to encrypt and decrypt vault files. If you have many encrypted files, decrypting them at startup may cause a perceptible delay. To speed this up, install the cryptography package:

pip install cryptography

[**Vault Format**](http://docs.ansible.com/ansible/latest/vault.html#id18)

A vault encrypted file is a UTF-8 encoded txt file.

The file format includes a new line terminated header.

For example:

$ANSIBLE\_VAULT;1.1;AES256

The header contains the vault format id, the vault format version, and a cipher id, seperated by semi-colons ‘;’

The first field $ANSIBLE\_VAULT is the format id. Currently $ANSIBLE\_VAULT is the only valid file format id. This is used to identify files that are vault encrypted (via vault.is\_encrypted\_file()).

The second field (*1.1*) is the vault format version. All supported versions of ansible will currently default to ‘1.1’.

The ‘1.0’ format is supported for reading only (and will be converted automatically to the ‘1.1’ format on write). The format version is currently used as an exact string compare only (version numbers are not currently ‘compared’).

The third field (AES256) identifies the cipher algorithmn used to encrypt the data. Currently, the only supported cipher is ‘AES256’. [vault format 1.0 used ‘AES’, but current code always uses ‘AES256’]

Note: In the future, the header could change. Anything after the vault id and version can be considered to depend on the vault format version. This includes the cipher id, and any additional fields that could be after that.

The rest of the content of the file is the ‘vaulttext’. The vaulttext is a text armored version of the encrypted ciphertext. Each line will be 80 characters wide, except for the last line which may be shorter.

[**Vault Payload Format 1.1**](http://docs.ansible.com/ansible/latest/vault.html#id19)

The vaulttext is a concatenation of the ciphertext and a SHA256 digest with the result ‘hexlifyied’.

‘hexlify’ refers to the hexlify() method of pythons binascii module.

hexlify()’ied result of:

* hexlify()’ed string of the salt, followed by a newline (‘n’)
* hexlify()’ed string of the crypted HMAC, followed by a newline. The HMAC is:
  + a *RFC2104 <https://www.ietf.org/rfc/rfc2104.txt>* style HMAC
    - inputs are:
      * The AES256 encrypted ciphertext
      * A PBKDF2 key. This key, the cipher key, and the cipher iv are generated from:
        + the salt, in bytes
        + 10000 iterations
        + SHA256() algorithmn
        + the first 32 bytes are the cipher key
        + the second 32 bytes are the HMAC key
        + remaining 16 bytes are the cipher iv
  + hexlify()’ed string of the ciphertext. The ciphertext is:
  + AES256 encrypted data. The data is encrypted using:
    - AES-CTR stream cipher
    - b\_pkey1
    - iv
    - a 128 bit counter block seeded from an integer iv
    - the plaintext
      * the original plaintext
      * padding up to the AES256 blocksize. (The data used for padding is based on *RFX5652 <https://tools.ietf.org/html/rfc5652#section-6.3>*)