Encrypting content with Ansible Vault

Ansible Vault encrypts variables and files so you can protect sensitive content such as passwords or keys rather than leaving it visible as plaintext in playbooks or roles. To use Ansible Vault you need one or more passwords to encrypt and decrypt content. If you store your vault passwords in a third-party tool such as a secret manager, you need a script to access them. Use the passwords with the :ref:`ansible-vault` command-line tool to create and view encrypted variables, create encrypted files, encrypt existing files, or edit, re-key, or decrypt files. You can then place encrypted content under source control and share it more safely.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst`, **line 7**); ***backlink***

Unknown interpreted text role "ref".

Warning

- Encryption with Ansible Vault ONLY protects 'data at rest'. Once the content is decrypted ('data in use'), play and plugin authors are responsible for avoiding any secret disclosure, see :ref:`no_log <keep_secret_data>` for details on hiding output and :ref:`vault_securing_editor` for security considerations on editors you use with Ansible Vault.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst`, **line 10**); ***backlink***

  Unknown interpreted text role "ref".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst`, **line 10**); ***backlink***

  Unknown interpreted text role "ref".

You can use encrypted variables and files in ad hoc commands and playbooks by supplying the passwords you used to encrypt them. You can modify your `ansible.cfg` file to specify the location of a password file or to always prompt for the password.

Managing vault passwords

Managing your encrypted content is easier if you develop a strategy for managing your vault passwords. A vault password can be any string you choose. There is no special command to create a vault password. However, you need to keep track of your vault passwords. Each time you encrypt a variable or file with Ansible Vault, you must provide a password. When you use an encrypted variable or file in a command or playbook, you must provide the same password that was used to encrypt it. To develop a strategy for managing vault passwords, start with two questions:

- Do you want to encrypt all your content with the same password, or use different passwords for different needs?
- Where do you want to store your password or passwords?

Choosing between a single password and multiple passwords

If you have a small team or few sensitive values, you can use a single password for everything you encrypt with Ansible Vault. Store your vault password securely in a file or a secret manager as described below.

If you have a larger team or many sensitive values, you can use multiple passwords. For example, you can use different passwords for different users or different levels of access. Depending on your needs, you might want a different password for each encrypted file, for each directory, or for each environment. For example, you might have a playbook that includes two vars files, one for the dev environment and one for the production environment, encrypted with two different passwords. When you run the playbook, select the correct vault password for the environment you are targeting, using a vault ID.

Managing multiple passwords with vault IDs

If you use multiple vault passwords, you can differentiate one password from another with vault IDs. You use the vault ID in three ways:

- Pass it with :option:`--vault-id <ansible-playbook --vault-id>` to the :ref:`ansible-vault` command when you create encrypted content

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst`, **line 39**); *backlink*

  Unknown interpreted text role "option".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst`, **line 39**); *backlink*

  Unknown interpreted text role "ref".

- Include it wherever you store the password for that vault ID (see :ref:`storing_vault_passwords`)

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst`, **line 40**); *backlink*

  Unknown interpreted text role "ref".

- Pass it with :option:`--vault-id <ansible-playbook --vault-id>` to the :ref:`ansible-playbook` command when you run a playbook that uses content you encrypted with that vault ID

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst`, **line 41**); *backlink*

  Unknown interpreted text role "option".

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst`, **line 41**); *backlink*

  Unknown interpreted text role "ref".

When you pass a vault ID as an option to the :ref:`ansible-vault` command, you add a label (a hint or nickname) to the encrypted content. This label documents which password you used to encrypt it. The encrypted variable or file includes the vault ID label in plain text in the header. The vault ID is the last element before the encrypted content. For example:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst`, **line 43**); *backlink*

Unknown interpreted text role "ref".

```
my_encrypted_var: !vault |
          $ANSIBLE_VAULT;1.2;AES256;dev
          30613233633461343837653833666333643061636561303338373661313838333565653635353162
          32633634346237333435386534626130643336343334646 60a6636336239393934393166366338 63
          61636237636537333938306331383339353265363239643939666639386530626330633337633833
          66646563343731666 30a3637363932626664656634326139326130363039633432636231373862 39
          6330
```

In addition to the label, you must provide a source for the related password. The source can be a prompt, a file, or a script, depending on how you are storing your vault passwords. The pattern looks like this:

```
--vault-id label@source
```

If your playbook uses multiple encrypted variables or files that you encrypted with different passwords, you must pass the vault IDs when you run that playbook. You can use :option:`--vault-id <ansible-playbook --vault-id>` by itself, with :option:`--vault-password-file <ansible-playbook --vault-password-file>`, or with :option:`--ask-vault-pass <ansible-playbook --ask-vault-pass>`. The pattern is the same as when you create encrypted content: include the label and the source for the matching password.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 61); backlink

Unknown interpreted text role "option".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 61); backlink

Unknown interpreted text role "option".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 61); backlink

Unknown interpreted text role "option".

See below for examples of encrypting content with vault IDs and using content encrypted with vault IDs. The :option:`--vault-id <ansible-playbook --vault-id>` option works with any Ansible command that interacts with vaults, including :ref:`ansible-vault`, :ref:`ansible-playbook`, and so on.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 63); backlink

Unknown interpreted text role "option".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 63); backlink

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 63); backlink

Unknown interpreted text role "ref".

Limitations of vault IDs

Ansible does not enforce using the same password every time you use a particular vault ID label. You can encrypt different variables or files with the same vault ID label but different passwords. This usually happens when you type the password at a prompt and make a mistake. It is possible to use different passwords with the same vault ID label on purpose. For example, you could use each label as a reference to a class of passwords, rather than a single password. In this scenario, you must always know which specific password or file to use in context. However, you are more likely to encrypt two files with the same vault ID label and different passwords by mistake. If you encrypt two files with the same label but different passwords by accident, you can :ref:`rekey <rekeying_files>` one file to fix the issue.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 68); backlink

Unknown interpreted text role "ref".

Enforcing vault ID matching

By default the vault ID label is only a hint to remind you which password you used to encrypt a variable or file. Ansible does not check that the vault ID in the header of the encrypted content matches the vault ID you provide when you use the content. Ansible decrypts all files and variables called by your command or playbook that are encrypted with the password you provide. To check the encrypted content and decrypt it only when the vault ID it contains matches the one you provide with `--vault-id`, set the config option :ref:`DEFAULT_VAULT_ID_MATCH`. When you set :ref:`DEFAULT_VAULT_ID_MATCH`, each password is only used to decrypt data that was encrypted with the same label. This is efficient, predictable, and can reduce errors when different values are encrypted with different passwords.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst`, **line 73**); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst`, **line 73**); *backlink*

Unknown interpreted text role "ref".

Note

Even with the :ref:`DEFAULT_VAULT_ID_MATCH` setting enabled, Ansible does not enforce using the same password every time you use a particular vault ID label.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst`, **line 76**); *backlink*

Unknown interpreted text role "ref".

Storing and accessing vault passwords

You can memorize your vault password, or manually copy vault passwords from any source and paste them at a command-line prompt, but most users store them securely and access them as needed from within Ansible. You have two options for storing vault passwords that work from within Ansible: in files, or in a third-party tool such as the system keyring or a secret manager. If you store your passwords in a third-party tool, you need a vault password client script to retrieve them from within Ansible.

Storing passwords in files

To store a vault password in a file, enter the password as a string on a single line in the file. Make sure the permissions on the file are appropriate. Do not add password files to source control.

Storing passwords in third-party tools with vault password client scripts

You can store your vault passwords on the system keyring, in a database, or in a secret manager and retrieve them from within Ansible using a vault password client script. Enter the password as a string on a single line. If your password has a vault ID, store it in a way that works with your password storage tool.

To create a vault password client script:

- Create a file with a name ending in either `-client` or `-client.EXTENSION`
- Make the file executable
- Within the script itself:
  - Print the passwords to standard output
  - Accept a `--vault-id` option
  - If the script prompts for data (for example, a database password), send the prompts to standard error

When you run a playbook that uses vault passwords stored in a third-party tool, specify the script as the source within the `--vault-id` flag. For example:

```
ansible-playbook --vault-id dev@contrib/vault/vault-keyring-client.py
```

Ansible executes the client script with a `--vault-id` option so the script knows which vault ID label you specified. For example a script loading passwords from a secret manager can use the vault ID label to pick either the 'dev' or 'prod' password. The example command above results in the following execution of the client script:

```
contrib/vault/vault-keyring-client.py --vault-id dev
```

For an example of a client script that loads passwords from the system keyring, see the vault-keyring-client script.

Encrypting content with Ansible Vault

Once you have a strategy for managing and storing vault passwords, you can start encrypting content. You can encrypt two types of content with Ansible Vault: variables and files. Encrypted content always includes the `!vault` tag, which tells Ansible and YAML that the content needs to be decrypted, and a `|` character, which allows multi-line strings. Encrypted content created with

`--vault-id` also contains the vault ID label. For more details about the encryption process and the format of content encrypted with Ansible Vault, see :ref:`vault_format`. This table shows the main differences between encrypted variables and encrypted files:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 123); *backlink*

Unknown interpreted text role "ref".

	Encrypted variables	Encrypted files
How much is encrypted?	Variables within a plaintext file	The entire file
When is it decrypted?	On demand, only when needed	Whenever loaded or referenced [1]
What can be encrypted?	Only variables	Any structured data file

[1] Ansible cannot know if it needs content from an encrypted file unless it decrypts the file, so it decrypts all encrypted files referenced in your playbooks and roles.

Encrypting individual variables with Ansible Vault

You can encrypt single values inside a YAML file using the :ref:`ansible-vault encrypt_string <ansible_vault_encrypt_string>` command. For one way to keep your vaulted variables safely visible, see :ref:`tip_for_variables_and_vaults`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 147); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 147); *backlink*

Unknown interpreted text role "ref".

Advantages and disadvantages of encrypting variables

With variable-level encryption, your files are still easily legible. You can mix plaintext and encrypted variables, even inline in a play or role. However, password rotation is not as simple as with file-level encryption. You cannot :ref:`rekey <rekeying_files>` encrypted variables. Also, variable-level encryption only works on variables. If you want to encrypt tasks or other content, you must encrypt the entire file.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 152); *backlink*

Unknown interpreted text role "ref".

Creating encrypted variables

The :ref:`ansible-vault encrypt_string <ansible_vault_encrypt_string>` command encrypts and formats any string you type (or copy or generate) into a format that can be included in a playbook, role, or variables file. To create a basic encrypted variable, pass three options to the :ref:`ansible-vault encrypt_string <ansible_vault_encrypt_string>` command:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 159); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 159); *backlink*

Unknown interpreted text role "ref".

- a source for the vault password (prompt, file, or script, with or without a vault ID)
- the string to encrypt
- the string name (the name of the variable)

The pattern looks like this:

```
ansible-vault encrypt_string <password_source> '<string_to_encrypt>' --name '<string_name_of_variable>'
```

For example, to encrypt the string 'foobar' using the only password stored in 'a_password_file' and name the variable 'the_secret':

```
ansible-vault encrypt_string --vault-password-file a_password_file 'foobar' --name 'the_secret'
```

The command above creates this content:

```
the_secret: !vault |
      $ANSIBLE_VAULT;1.1;AES256
      623133653966623430613934643361633837643737646136336536343062313864336264366233 61
      613433366535396636353433363266653533376166613162 0a663537646436643839616531643561
      633962653339663861663736326265393261663539653632626330303336303133386463353036 30
      343862666666137650a353638643435666633633964366338633066623234616432373231333331
      6564
```

To encrypt the string 'fooooodev', add the vault ID label 'dev' with the 'dev' vault password stored in 'a_password_file', and call the encrypted variable 'the_dev_secret':

```
ansible-vault encrypt_string --vault-id dev@a_password_file 'fooooodev' --name 'the_dev_secret'
```

The command above creates this content:

```
the_dev_secret: !vault |
          $ANSIBLE_VAULT;1.2;AES256;dev
          306132336334613438376538336663336430616365613033383736613138383335656536353531 62
          326336343462373334353865346261306433363433346466 0a6636336239393943931663663386 3
          616362376365373339383063313833393532653632396439396666393865306263306333376338 33
          666465633437316663 0a363736393262666465663432613932613036303963343263623137386239
          6330
```

To encrypt the string 'letmein' read from stdin, add the vault ID 'dev' using the 'dev' vault password stored in *a_password_file*, and name the variable 'db_password':

```
echo -n 'letmein' | ansible-vault encrypt_string --vault-id dev@a_password_file --stdin-name 'db_password'
```

> **Warning**
>
> Typing secret content directly at the command line (without a prompt) leaves the secret string in your shell history. Do not do this outside of testing.

The command above creates this output:

```
Reading plaintext input from stdin. (ctrl-d to end input, twice if your content does not already have a ne
db_password: !vault |
          $ANSIBLE_VAULT;1.2;AES256;dev
          613239313538666663363061393739373163663661386561313238633738663766663533643737 61
          353963323431383634643532376630616462613437656433 0a373530313635343535343133316133
          366436663064346162663764343632393464336432383364646435663861353563343037363531 36
          656563313336636636 0a326566323363363936613664616364623437336130623133343530333739
          3039
```

To be prompted for a string to encrypt, encrypt it with the 'dev' vault password from 'a_password_file', name the variable 'new_user_password' and give it the vault ID label 'dev':

```
ansible-vault encrypt_string --vault-id dev@a_password_file --stdin-name 'new_user_password'
```

The command above triggers this prompt:

```
Reading plaintext input from stdin. (ctrl-d to end input, twice if your content does not already have a new li
```

Type the string to encrypt (for example, 'hunter2'), hit ctrl-d, and wait.

> **Warning**
>
> Do not press `Enter` after supplying the string to encrypt. That will add a newline to the encrypted value.

The sequence above creates this output:

```
new_user_password: !vault |
          $ANSIBLE_VAULT;1.2;AES256;dev
          376365613666366434643763363034666130626335373236323065666535333838333664623666662
          656535306330306530383132353965613865386335323062 0a653638643639333133306331336365
          623737376233376161303861373734613065353835383731623162633861653761316236313234 34
          386636386236333562 0a376466656164383032633338306162326639643635663936623939666238
          3161
```

You can add the output from any of the examples above to any playbook, variables file, or role for future use. Encrypted variables are larger than plain-text variables, but they protect your sensitive content while leaving the rest of the playbook, variables file, or role in plain text so you can easily read it.

Viewing encrypted variables

You can view the original value of an encrypted variable using the debug module. You must pass the password that was used to encrypt the variable. For example, if you stored the variable created by the last example above in a file called 'vars.yml', you could view the unencrypted value of that variable like this:

```
ansible localhost -m ansible.builtin.debug -a var="new_user_password" -e "@vars.yml" --vault-id dev@a_password

localhost | SUCCESS => {
```

```
    "new_user_password": "hunter2"
}
```

Encrypting files with Ansible Vault

Ansible Vault can encrypt any structured data file used by Ansible, including:

- group variables files from inventory
- host variables files from inventory
- variables files passed to ansible-playbook with `-e @file.yml` or `-e @file.json`
- variables files loaded by `include_vars` or `vars_files`
- variables files in roles
- defaults files in roles
- tasks files
- handlers files
- binary files or other arbitrary files

The full file is encrypted in the vault.

> **Note**
>
> Ansible Vault uses an editor to create or modify encrypted files. See :ref:`vault_securing_editor` for some guidance on securing the editor.
>
> > **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 295); *backlink***
> >
> > Unknown interpreted text role "ref".

Advantages and disadvantages of encrypting files

File-level encryption is easy to use. Password rotation for encrypted files is straightforward with the :ref:`rekey <rekeying_files>` command. Encrypting files can hide not only sensitive values, but the names of the variables you use. However, with file-level encryption the contents of files are no longer easy to access and read. This may be a problem with encrypted tasks files. When encrypting a variables file, see :ref:`tip_for_variables_and_vaults` for one way to keep references to these variables in a non-encrypted file. Ansible always decrypts the entire encrypted file when it is when loaded or referenced, because Ansible cannot know if it needs the content unless it decrypts it.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 301); *backlink***
>
> Unknown interpreted text role "ref".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 301); *backlink***
>
> Unknown interpreted text role "ref".

Creating encrypted files

To create a new encrypted data file called 'foo.yml' with the 'test' vault password from 'multi_password_file':

```
ansible-vault create --vault-id test@multi_password_file foo.yml
```

The tool launches an editor (whatever editor you have defined with $EDITOR, default editor is vi). Add the content. When you close the editor session, the file is saved as encrypted data. The file header reflects the vault ID used to create it:

```
``$ANSIBLE_VAULT;1.2;AES256;test``
```

To create a new encrypted data file with the vault ID 'my_new_password' assigned to it and be prompted for the password:

```
ansible-vault create --vault-id my_new_password@prompt foo.yml
```

Again, add content to the file in the editor and save. Be sure to store the new password you created at the prompt, so you can find it when you want to decrypt that file.

Encrypting existing files

To encrypt an existing file, use the :ref:`ansible-vault encrypt <ansible_vault_encrypt>` command. This command can operate on multiple files at once. For example:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 333); *backlink***

Unknown interpreted text role "ref".

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

To encrypt existing files with the 'project' ID and be prompted for the password:

```
ansible-vault encrypt --vault-id project@prompt foo.yml bar.yml baz.yml
```

Viewing encrypted files

To view the contents of an encrypted file without editing it, you can use the :ref:`ansible-vault view <ansible_vault_view>` command:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 351); *backlink*

Unknown interpreted text role "ref".

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

Editing encrypted files

To edit an encrypted file in place, use the :ref:`ansible-vault edit <ansible_vault_edit>` command. This command decrypts the file to a temporary file, allows you to edit the content, then saves and re-encrypts the content and removes the temporary file when you close the editor. For example:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 363); *backlink*

Unknown interpreted text role "ref".

```
ansible-vault edit foo.yml
```

To edit a file encrypted with the `vault2` password file and assigned the vault ID `pass2`:

```
ansible-vault edit --vault-id pass2@vault2 foo.yml
```

Changing the password and/or vault ID on encrypted files

To change the password on an encrypted file or files, use the :ref:`rekey <ansible_vault_rekey>` command:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 381); *backlink*

Unknown interpreted text role "ref".

```
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. To set a different ID for the rekeyed files, pass the new ID to `--new-vault-id`. For example, to rekey a list of files encrypted with the 'preprod1' vault ID from the 'ppold' file to the 'preprod2' vault ID and be prompted for the new password:

```
ansible-vault rekey --vault-id preprod1@ppold --new-vault-id preprod2@prompt foo.yml bar.yml baz.yml
```

Decrypting encrypted files

If you have an encrypted file that you no longer want to keep encrypted, you can permanently decrypt it by running the :ref:`ansible-vault decrypt <ansible_vault_decrypt>` command. This command will save the file unencrypted to the disk, so be sure you do not want to :ref:`edit <ansible_vault_edit>` it instead.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 399); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 399); *backlink*

Unknown interpreted text role "ref".

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

Steps to secure your editor

Ansible Vault relies on your configured editor, which can be a source of disclosures. Most editors have ways to prevent loss of data, but these normally rely on extra plain text files that can have a clear text copy of your secrets. Consult your editor documentation to configure the editor to avoid disclosing secure data. The following sections provide some guidance on common editors but should not be taken as a complete guide to securing your editor.

vim

You can set the following `vim` options in command mode to avoid cases of disclosure. There may be more settings you need to modify to ensure security, especially when using plugins, so consult the `vim` documentation.

1. Disable swapfiles that act like an autosave in case of crash or interruption.

```
set noswapfile
```

2. Disable creation of backup files.

```
set nobackup
set nowritebackup
```

3. Disable the viminfo file from copying data from your current session.

```
set viminfo=
```

4. Disable copying to the system clipboard.

```
set clipboard=
```

You can optionally add these settings in `.vimrc` for all files, or just specific paths or extensions. See the `vim` manual for details.

Emacs

You can set the following Emacs options to avoid cases of disclosure. There may be more settings you need to modify to ensure security, especially when using plugins, so consult the Emacs documentation.

1. Do not copy data to the system clipboard.

```
(setq x-select-enable-clipboard nil)
```

2. Disable creation of backup files.

```
(setq make-backup-files nil)
```

3. Disable autosave files.

```
(setq auto-save-default nil)
```

Using encrypted variables and files

When you run a task or playbook that uses encrypted variables or files, you must provide the passwords to decrypt the variables or files. You can do this at the command line or in the playbook itself.

Passing a single password

If all the encrypted variables and files your task or playbook needs use a single password, you can use the :option:`--ask-vault-pass <ansible-playbook --ask-vault-pass>` or :option:`--vault-password-file <ansible-playbook --vault-password-file>` cli options.

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 484); *backlink*
>
> Unknown interpreted text role "option".

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 484); *backlink*
>
> Unknown interpreted text role "option".

To prompt for the password:

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

To retrieve the password from the :file:`/path/to/my/vault-password-file` file:

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 492); *backlink*
>
> Unknown interpreted text role "file".

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

To get the password from the vault password client script :file:`my-vault-password-client.py`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 498); *backlink*

Unknown interpreted text role "file".

```
ansible-playbook --vault-password-file my-vault-password-client.py
```

Passing vault IDs

You can also use the :option:`--vault-id <ansible-playbook --vault-id>` option to pass a single password with its vault label. This approach is clearer when multiple vaults are used within a single inventory.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 510); *backlink*

Unknown interpreted text role "option".

To prompt for the password for the 'dev' vault ID:

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

To retrieve the password for the 'dev' vault ID from the :file:`dev-password` file:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 518); *backlink*

Unknown interpreted text role "file".

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

To get the password for the 'dev' vault ID from the vault password client script :file:`my-vault-password-client.py`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 524); *backlink*

Unknown interpreted text role "file".

```
ansible-playbook --vault-id dev@my-vault-password-client.py
```

Passing multiple vault passwords

If your task or playbook requires multiple encrypted variables or files that you encrypted with different vault IDs, you must use the :option:`--vault-id <ansible-playbook --vault-id>` option, passing multiple `--vault-id` options to specify the vault IDs ('dev', 'prod', 'cloud', 'db') and sources for the passwords (prompt, file, script). . For example, to use a 'dev' password read from a file and to be prompted for the 'prod' password:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 533); *backlink*

Unknown interpreted text role "option".

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

By default the vault ID labels (dev, prod and so on) are only hints. Ansible attempts to decrypt vault content with each password. The password with the same label as the encrypted data will be tried first, after that each vault secret will be tried in the order they were provided on the command line.

Where the encrypted data has no label, or the label does not match any of the provided labels, the passwords will be tried in the order they are specified. In the example above, 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.

Using `--vault-id` without a vault ID

The :option:`--vault-id <ansible-playbook --vault-id>` option can also be used without specifying a vault-id. This behavior is equivalent to :option:`--ask-vault-pass <ansible-playbook --ask-vault-pass>` or :option:`--vault-password-file <ansible-playbook --vault-password-file>` so is rarely used.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 546); *backlink*

Unknown interpreted text role "option".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 546); *backlink*

Unknown interpreted text role "option".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 546); *backlink*

Unknown interpreted text role "option".

For example, to use a password file :file:`dev-password`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 548); *backlink*

Unknown interpreted text role "file".

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

To prompt for the password:

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

To get the password from an executable script :file:`my-vault-password-client.py`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 560); *backlink*

Unknown interpreted text role "file".

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

Configuring defaults for using encrypted content

Setting a default vault ID

If you use one vault ID more frequently than any other, you can set the config option :ref:`DEFAULT_VAULT_IDENTITY_LIST` to specify a default vault ID and password source. Ansible will use the default vault ID and source any time you do not specify :option:`--vault-id <ansible-playbook --vault-id>`. You can set multiple values for this option. Setting multiple values is equivalent to passing multiple :option:`--vault-id <ansible-playbook --vault-id>` cli options.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 573); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 573); *backlink*

Unknown interpreted text role "option".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 573); *backlink*

Unknown interpreted text role "option".

Setting a default password source

If you use one vault password file more frequently than any other, you can set the :ref:`DEFAULT_VAULT_PASSWORD_FILE` config option or the :envvar:`ANSIBLE_VAULT_PASSWORD_FILE` environment variable to specify that file. For example, if you set `ANSIBLE_VAULT_PASSWORD_FILE=~/.vault_pass.txt`, Ansible will automatically search for the password in that file. This is useful if, for example, you use Ansible from a continuous integration system such as Jenkins.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)

(user_guide)vault.rst, line 578); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 578); *backlink*

Unknown interpreted text role "envvar".

When are encrypted files made visible?

In general, content you encrypt with Ansible Vault remains encrypted after execution. However, there is one exception. If you pass an encrypted file as the `src` argument to the :ref:`copy <copy_module>`, :ref:`template <template_module>`, :ref:`unarchive <unarchive_module>`, :ref:`script <script_module>` or :ref:`assemble <assemble_module>` module, the file will not be encrypted on the target host (assuming you supply the correct vault password when you run the play). This behavior is intended and useful. You can encrypt a configuration file or template to avoid sharing the details of your configuration, but when you copy that configuration to servers in your environment, you want it to be decrypted so local users and processes can access it.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 583); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 583); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 583); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 583); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\(ansible-devel)(docs)(docsite)(rst)(user_guide)vault.rst, line 583); *backlink*

Unknown interpreted text role "ref".

Format of files encrypted with Ansible Vault

Ansible Vault creates UTF-8 encoded txt files. The file format includes a newline terminated header. For example:

```
$ANSIBLE_VAULT;1.1;AES256
```

or

```
$ANSIBLE_VAULT;1.2;AES256;vault-id-label
```

The header contains up to four elements, separated by semi-colons (`;`).

1. The format ID (`$ANSIBLE_VAULT`). Currently `$ANSIBLE_VAULT` is the only valid format ID. The format ID identifies content that is encrypted with Ansible Vault (via vault.is_encrypted_file()).
2. The vault format version (`1.X`). All supported versions of Ansible will currently default to '1.1' or '1.2' if a labeled vault ID is supplied. 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').
3. The cipher algorithm used to encrypt the data (`AES256`). Currently `AES256` is the only supported cipher algorithm. Vault format 1.0 used 'AES', but current code always uses 'AES256'.
4. The vault ID label used to encrypt the data (optional, `vault-id-label`) For example, if you encrypt a file with `--vault-id dev@prompt`, the vault-id-label is `dev`.

Note: In the future, the header could change. Fields after the format ID and format version depend on the format version, and future vault format versions may add more cipher algorithm options and/or additional fields.

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

Ansible Vault payload format 1.1 - 1.2

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

'hexlify' refers to the `hexlify()` method of the Python Standard Library's binascii module.

hexlify()'ed result of:

- hexlify()'ed string of the salt, followed by a newline (`0x0a`)
- hexlify()'ed string of the crypted HMAC, followed by a newline. The HMAC is:
    - a RFC2104 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() algorithm
                - 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
        - cipher key
        - 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 RFC5652)