Ansible Vault: Practical Guide

What is Ansible Vault?

Ansible Vault is a feature in Ansible that allows users to encrypt and securely store sensitive data, such as passwords, API keys, and certificates. This ensures that sensitive information does not appear in plain text within your playbooks, inventories, or configuration files. In an organizational setting, this is crucial for maintaining security and compliance when managing infrastructure at scale.

Use Cases in Organizations

- **Secure credentials**: Protect secrets like database passwords or SSH keys from unauthorized access.
- **Compliance requirements**: Meet security policies by ensuring sensitive data is encrypted.
- **Collaboration safety**: Share playbooks with team members without exposing confidential information.

Inside Ansible Vault

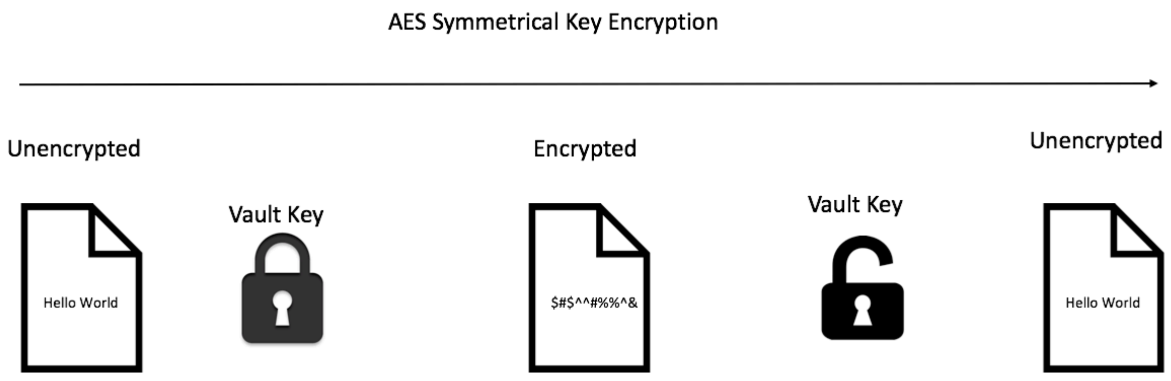
1. Vault Files: These are the files encrypted by Ansible Vault. They can contain variables, configurations, or any sensitive data.

2. Encryption Key: A password or key is used to encrypt and decrypt the vault. Without it, the contents are inaccessible.

3. Vault Commands: Commands like `ansible-vault create`, `edit`, and `view` let you manage encrypted files.

4. Vault IDs: Useful for handling multiple encryption keys for different environments (e.g., staging and production).

Practical: Using Ansible Vault



Let's walk through creating and using Ansible Vault with simple steps.

Step 1: Create a Vault File

```
ansible-vault create secrets.yml
```

Explanation: This command creates a new encrypted file named `secrets.yml`. You'll be prompted to set a password. Once done, any content added to this file will be encrypted.

Step 2: Add Sensitive Data

After running the command, you enter an editor to write sensitive data, for example:

```
db_password: MySecurePassword123

api_key: ABCD1234XYZ
```

Explanation: The editor allows you to define key-value pairs securely. Upon saving, the file is encrypted and cannot be read without the password.

Step 3: View Vault File

```
ansible-vault view secrets.yml
```

Explanation: This command decrypts the file temporarily for viewing purposes. It ensures no plain-text data is exposed directly in files.

Step 4: Edit Vault File

```
ansible-vault edit secrets.yml
```

Explanation: This opens the encrypted file for editing. You can add, update, or delete sensitive data securely.

Step 5: Use Vault File in Playbooks

Reference the encrypted variables in a playbook like this:

```
- name: Use sensitive data

 hosts: all

 vars_files:

   - secrets.yml

 tasks:

   - name: Print database password

     debug:
      msg: "{{ db_password }}"
```

Explanation: By including `vars_files`, Ansible loads the encrypted file, decrypts it during execution, and allows you to use its contents in tasks.

Step 6: Encrypt an Existing File

```
ansible-vault encrypt file.yml
```

Explanation: This command encrypts an existing file that contains sensitive data. It's a quick way to protect already-created configuration files.

Step 7: Decrypt a File

```
ansible-vault decrypt file.yml
```

Explanation: This removes encryption from a file, converting it back to plain text. Use this carefully to avoid exposing sensitive data.

Step 8: Re-Key a Vault File

```
ansible-vault rekey secrets.yml
```

Explanation: This changes the password for the encrypted file. Useful when rotating keys or securing access after personnel changes.

Final Thoughts

Ansible Vault is an essential tool for managing sensitive data in automated workflows. Its encryption capabilities ensure that organizations can securely scale their infrastructure management without compromising on security. By mastering its commands and workflows, you can seamlessly integrate security into your Ansible practices.