IMPLEMENTING ANSIBLE VAULT

Overview	
Goal	Manage encryption with Ansible Vault.
Objectives	• Create, edit, rekey, encrypt, and decrypt files. • Run a playbook with Ansible Vault.
Sections	• Configuring Ansible Vault (and Guided Exercise) • Executing with Ansible Vault (and Guided Exercise)
Lab	• Implementing Ansible Vault

Configuring Ansible Vault

Ansible Vault

Ansible may need access to sensitive data such as passwords or API keys in order to configure remote servers. Normally, this information might be stored as plain text in inventory variables or other Ansible files. But in that case, any user with access to the Ansible files or a version control system which stores the Ansible files would have access to this sensitive data. This poses an obvious security risk.

There are two primary ways to store this data more securely:
• Use Ansible Vault, which is included with Ansible and can encrypt and decrypt any structured data file used by Ansible.
• Use a third-party key management service to store the data in the cloud, such as Vault by HashiCorp, Amazon's AWS Key Management Service, or Microsoft Azure Key Vault.

In this part of the course, you will learn how to use Ansible Vault.
To use Ansible Vault, a command line tool called **ansible-vault** is used to create, edit, encrypt, decrypt, and view files. Ansible Vault can encrypt any structured data file used by Ansible. This might include inventory variables, included variable files in a playbook, variable files passed as an argument when executing the playbook, or variables defined in Ansible roles.

Important

Ansible Vault does not implement its own cryptographic functions but uses an external Python toolkit. Files are protected with symmetric encryption using AES256 with a password as the secret key. Note that the way this is done has not been formally audited by a third party.

References

ansible-vault(1) man page
Vault — Ansible Documentation
http://docs.ansible.com/ansible/playbooks_vault.html

Guided Exercise: Configuring Ansible Vault

In this exercise, you will create a new encrypted file, edit the file, and change the password on an existing encrypted file. You will also learn how to encrypt and decrypt an existing file.

Outcomes

You should be able to :

- Create a new encrypted file.
- Edit an encrypted file.
- View content of an encrypted file.
- Change the password of an encrypted file.
- Encrypt and decrypt an existing file.

#cd /home/ansible/playbook/conf-ansible-vault

Create an encrypted file named **super-secret.yml** under **~/conf-ansible-vault**. Use **redhat** as the vault password and enter content as This is encrypted.

#ansible-vault create super-secret.yml

```
[ansible@robo conf-ansible-vault]$ ansible-vault create super-secret.yml
New Vault password:
Confirm New Vault password:
[ansible@robo conf-ansible-vault]$
```

Attempt to view the content of the encrypted file, **super-secret.yml**.

#cat super-secret.yml

```
[ansible@robo conf-ansible-vault]$ cat super-secret.yml
$ANSIBLE_VAULT;1.1;AES256
33363339316130653038363963316136646636323163623365346664646131343865653062613932
61353430383636653638626532386162316236646636363 20a376235303861613266373166616336
31643461303861343163731306134356137616231343836326365336432636136353933646439 35
32333861303663343 00a32353734393836353534666266393062663033613537343862636631366 6
37353239633861613533663737636636316435343837616261343363393262336437
[ansible@robo conf-ansible-vault]$
```

Since the file, **super-secret.yml**, is an encrypted file, you cannot view the content in plain text. The default cipher used is *AES* (which is shared-secret based).

#ansible-vault view super-secret.yml

```
[ansible@robo conf-ansible-vault]$ ansible-vault view super-secret.yml
Vault password:
This is encrypted.
[ansible@robo conf-ansible-vault]$
```

Now edit the encrypted file **super-secret.yml** to add some new content “This is also encrypted.” Use **redhat** as the vault password.

#ansible-vault edit super-secret.yml

#ansible-vault view super-secret.yml

```
[ansible@robo conf-ansible-vault]$ ansible-vault view super-secret.yml
Vault password:
This is encrypted.
This is also encrypted.
[ansible@robo conf-ansible-vault]$
```

Change the vault password of the **super-secret.yml** file, using the command **ansible-vault rekey super-secret.yml**. The current password for the **super-secret.yml** file is **redhat**. Change this password to **ansible**.

ansible-vault rekey super-secret.yml

```
[ansible@robo conf-ansible-vault]$ ansible-vault rekey super-secret.yml
Vault password:
New Vault password:
Confirm New Vault password:
Rekey successful
[ansible@robo conf-ansible-vault]$
```

Decrypt the encrypted file, **super-secret.yml**, and save the file as **super-secret-decrypted.yml**. Use the **ansible-vault decrypt** subcommand with the *--output* option. Enter **ansible** as the password.

ansible-vault decrypt super-secret.yml --output=super-secret-decrypted.yml

```
[ansible@robo conf-ansible-vault]$ ansible-vault decrypt super-secret.yml --output=super-secret-decrypted.yml
Vault password:
Decryption successful
[ansible@robo conf-ansible-vault]$
```

cat super-secret-decrypted.yml

```
[ansible@robo conf-ansible-vault]$ cat super-secret-decrypted.yml
This is encrypted.
This is also encrypted.
[ansible@robo conf-ansible-vault]$
```

#ls -lrt

```
[ansible@robo conf-ansible-vault]$ ls -lrt
total 8
-rw------- 1 ansible ansible 484 Apr 21 18:09 super-secret.yml
-rw------- 1 ansible ansible  43 Apr 21 18:13 super-secret-decrypted.yml
[ansible@robo conf-ansible-vault]$
```

ansible-vault encrypt super-secret-decrypted.yml --output=super-secret.yml

```
[ansible@robo conf-ansible-vault]$ ansible-vault encrypt super-secret-decrypted.yml --output=super-secret.yml
New Vault password:
Confirm New Vault password:
Encryption successful
[ansible@robo conf-ansible-vault]$
```

#ls -lrt

```
[ansible@robo conf-ansible-vault]$ ls -lrt
total 8
-rw------- 1 ansible ansible  43 Apr 21 18:13 super-secret-decrypted.yml
-rw------- 1 ansible ansible 484 Apr 21 18:16 super-secret.yml
[ansible@robo conf-ansible-vault]$
```

References

ansible-playbook(1) and **ansible-vault**(1) man pages
Running a Playbook With Vault — Ansible Documentation
http://docs.ansible.com/ansible/playbooks_vault.html#running-a-playbook-withvault
Variables and Vaults — Ansible Documentation
http://docs.ansible.com/ansible/playbooks_best_practices.html#best-practices-forvariables-and-vaults

Guided Exercise: Executing with Ansible Vault

In this exercise, you will use Ansible Vault to encypt the file containing passwords on the local system and use the encrypted file in a playbook to create users on the robo2 managed host.

#cd /home/ansible/playbook/exec-ansible-vault

Create an encrypted file named **secret.yml** in the **~/exec-ansible-vault/** directory. This file will define the password variables and store the passwords to be used in the playbook.
Use the associative array variable, newusers, to define users and passwords using the name and pw keys, respectively. Define the ansibleuser1 user and its redhat password. Also define the ansibleuser2 user and its **Re4H1T** password.
Set the vault password to **redhat**.

ansible-vault create secret.yml
ansible-vault view secret.yml

```
[ansible@robo exec-ansible-vault]$ ansible-vault view secret.yml
Vault password:
newusers:
  - name: ansibleuser1
    pw: redhat
  - name: ansibleuser2
    pw: Re4H1T
```

Create a playbook which will use the variables defined in the **secret.yml** encrypted file. Name the playbook **create_users.yml** and create it under the **~/exec-ansiblevault/** directory.
Configure the playbook to use the **webservers** host group, which was defined by the lab setup script in the inventory file. Run this playbook as the **ansible** user on the remote managed host. Configure the playbook to create the **ansibleuser1** and **ansibleuser2** users.
The password stored as plain text in the variable, pw, should be converted into password hash using hashing filters **password_hash** to get SHA512 hashed password and passed as an argument to the user module.

#cat create_users.yml

```
---
- name: create user accounts for all our servers
  hosts: webservers
  become: true
  remote_user: ansible
  vars_files:
    - secret.yml
  tasks:
    - name: Creating users from secret.yml
      user:
        name: "{{ item.name }}"
        password: "{{ item.pw | password_hash('sha512') }}"
      with_items: "{{ newusers }}"
```

#ansible-playbook --syntax-check --ask-vault-pass create_users.yml

Create a password file to use for the playbook execution instead of asking for a password. The file should be called **vault-pass** and it should store the **redhat** vault password as a plain text. Change the permission of the file to **0600**.

echo 'redhat' > vault-pass
chmod 0600 vault-pass

ansible-playbook --vault-password-file=vault-pass create_users.yml

```
[ansible@robo exec-ansible-vault]$ ansible-playbook --vault-password-file=vault-pass create_users.yml
SUDO password:

PLAY [create user accounts for all our servers] *********************************************

TASK [Gathering Facts] **********************************************************************
ok: [robo2]

TASK [Creating users from secret.yml] *******************************************************
changed: [robo2] => (item={u'name': u'ansibleuser1', u'pw': u'redhat'})
changed: [robo2] => (item={u'name': u'ansibleuser2', u'pw': u'Re4H1T'})

PLAY RECAP **********************************************************************************
robo2                      : ok=2    changed=1    unreachable=0    failed=0
```

Final Output:-

ssh -o PreferredAuthentications=redhat ansibleuser1@robo2

```
[ansible@robo exec-ansible-vault]$ ssh -o PreferredAuthentications=redhat ansibleuser1@robo2
Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
[ansible@robo exec-ansible-vault]$ ssh -o PreferredAuthentications=password ansibleuser1@robo2
ansibleuser1@robo2's password:
This is the system robo2.
Today's date is: 2019-04-18.
Only use this system with permission.
You can ask deepan.redhat@gmail.com for access.
[ansibleuser1@robo2 ~]$
```