Use Ansible vault to protect sensitive data witin your Ansible Roles!

**Command Rundown**

Let's see how to encrypt sensitive data in our Roles so we can use Ansible to handle sensitive data such as SSH private keys or passwords.

We'll create users on our server using Ansible, which will give us the opportunity to use Ansible Vault.

**Setup "Users" role:**

Creat

cd roles

mkdir users

cd users

mkdir files handlers meta templates tasks vars

**Meta**

We'll define no dependencies for this role:

---

dependencies: []

**Variables**

We want this file to be encrypted. Create roles/users/variables with Ansible Vault:

This command is run from the roles/users directory:

ansible-vault create vars/main.yml

Create a password for your vault and confirm it.

This will open a file to open in your editor of choice (defaults to Vim, or whatever is your default editor).

If you want to use nano over vim, export the EDITOR variable and set it to nano: export EDITOR=nano

Exit the main.yml file if you have not yet already. We'll create some passwords using the mkpasswdcommand using the SHA-512 method:

# Whois package contains "mkpasswd" command

sudo apt-get install -y whois

# Create a password for a user

mkpasswd --method=SHA-512

Copy the password hash generated, and edit the vars/main.yml file again using Vault:

ansible-vault edit vars/main.yml

Add your passwords and other data:

---

admin\_password: <a generated password hash>

deploy\_password: <another generated password hash>

shared\_publickey: <your SSH public key to be placed in servers authorized\_keys directory>

Save and quit that file. If you edit the file without Vault, you'll see that the file is encrypted.

Other ansible-vault commands available are seen via:

ansible-vault -h

**Tasks**

Now that we created secure variables, we can use them in a task. Create file tasks/main.yml:

---

- name: Create Admin User

user: name=admin password="{{ admin\_password }}" groups=sudo shell=/bin/bash

- name: Add Admin Authorized Key

authorized\_key: user=admin key="{{ shared\_publickey }}" state=present

- name: Create Deploy User

user: name=deploy password="{{ deploy\_password }}" groups=www-data shell=/bin/bash

- name: Add Deploy Authorized Key

authorized\_key: user=deploy key="{{ shared\_publickey }}" state=present

Now adjust the nginx.yml file to be named servers.yml and run it.

---

- hosts: web

sudo: yes

user: root

roles:

- nginx

- users

Test it and then run it:

ansible-playbook --syntax-check --ask-vault-pass servers.yml

ansible-playbook --ask-vault-pass --private-key=~/.ssh/id\_ansible servers.yml

Then we can log into our servers and check to ensure the servers exist.

On a remote server:

# Check for user "admin" and "deploy"

cat /etc/passwd

From the local computer, try logging in as one of the new users!

# For me, logging in using the Ansible SSH key looked like this:

ssh -i ~/.ssh/id\_ansible admin@104.131.43.90

https://serversforhackers.com/video/ansible-using-vault