Managing Configuration Secrets from Ansible using Hashicorp Vault

Susan Foster
Raafiul Hossain

LA-UR-21-27954

Who are we?

Susan Foster

Computer Science Major, Statistics Minor

University of Maryland, College Park

Studying Computer Science with a focus on Cyber Security

Raafiul Hossain

Mechanical Engineering Major, Computer Science Minor

Binghamton University, NY

Interested in Cyber Security, Cloud Architecture

What is Ansible?

Configuration Management

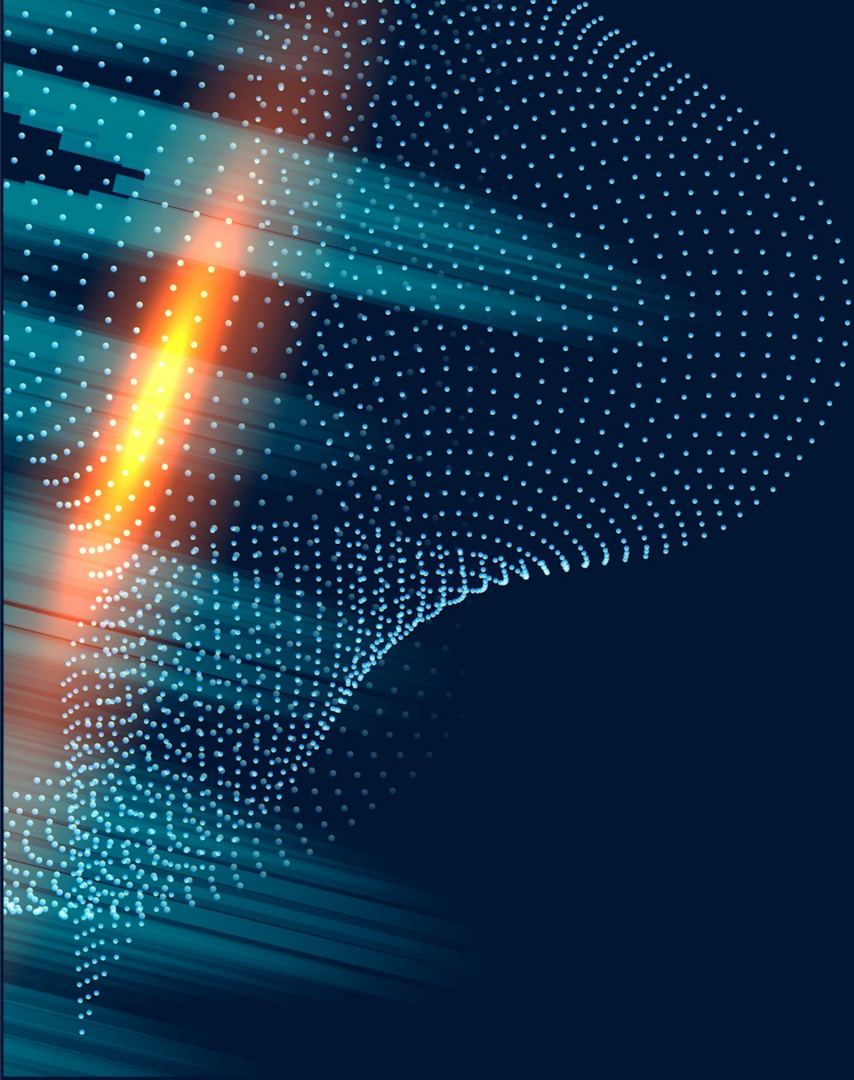
Ansible works by connecting various host servers to an ansible client which uses a push architecture to provide necessary configurations.

Application Deployment

The primary orchestration tool of Ansible are called Playbooks, and they are written in YAML Format

Red Hat Linux

Ansible allows us to pull from a centralized source called an Ansible client and push configuration to cluster nodes



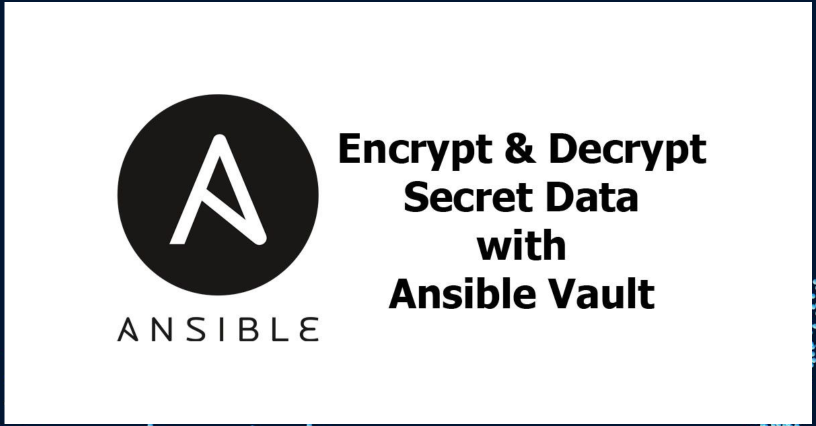
WHOA!

What about Secrets Management?

We have Ansible Vault!

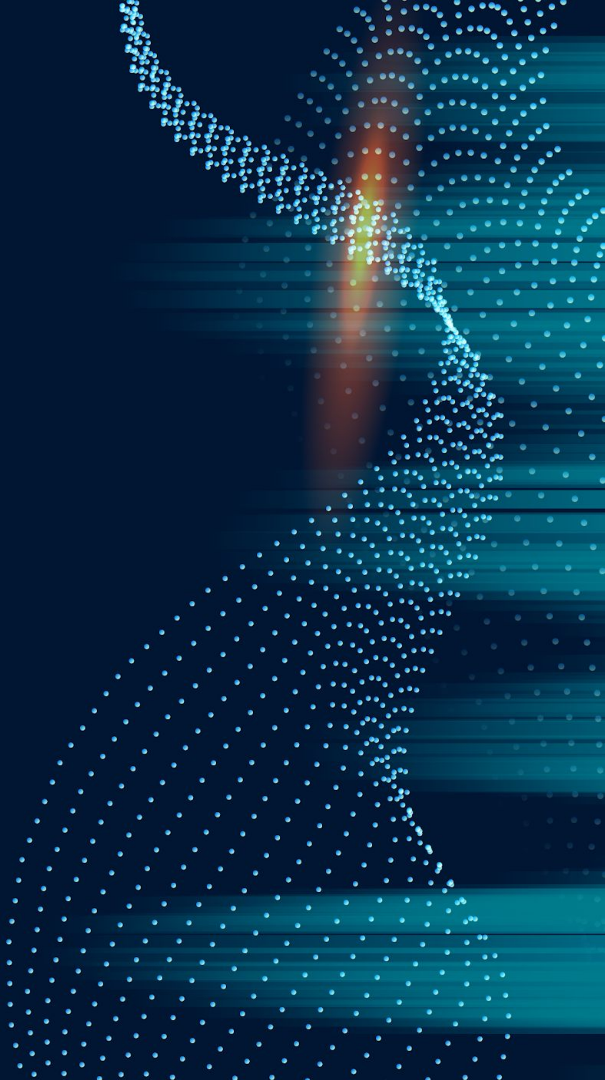
Built in Secrets Management Methodology: Ansible Vault

- Ansible Vault is included in Ansible as a feature that allows encryption and decryption of sensitive information directly from Playbooks
- Utilizes the same password for encryption and decryption and is user friendly.



What is Ansible Vault Missing?

- Ansible Vault does not manage secrets across multiple node deployments
- In Ansible Vault same encryption key must be used across every server.
- Thus In Ansible Vault, a key rotation requires manual updates for every server’s key.
- Limited in usage and scale with other configuration management sources, only fully compatible with Ansible.



Potential Solutions:
We Make Our Own Vault Management System

Pros:

- Custom to fit our needs
- Full control over system management
- Cheaper than 3rd party Platform.

Cons:

- Write custom application integration systems.
- Time to implement and develop
- Code must be secure and production level
- Potential vulnerabilities outside of HPC

Our Proposed Solution: Hashicorp Vault

HashiCorp Vault

Secrets Management

Vault manages secrets through trusted sources with the use of tokens and policies to ensure that only those with the correct permissions can access the secret

Data Encryption

All data stored in the Vault is encrypted. It can only be accessed with a valid token.

Data Protection

All data stored in the Vault is encrypted, and all tokens are managed with a key manager

What is Docker and Why Do We Use It?

Containers

Helps to manage, build, and deploy containers on any system. Standardizes components across systems and deployments

Isolation

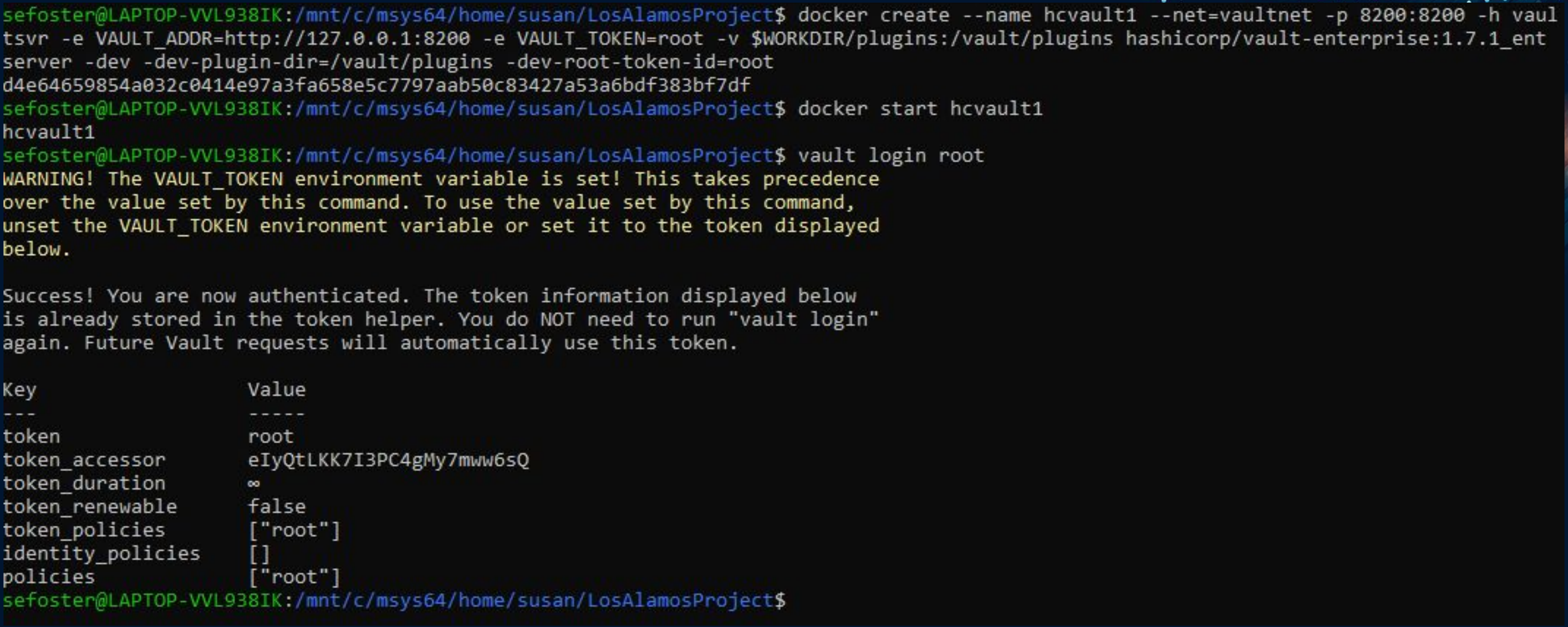
Containers are isolated from the standard Linux kernel, and have a smaller shares of memory

Ease of Use

Once you've obtained a container image, it's easy to build and run it via command line.

HashiCorp Vault in Action

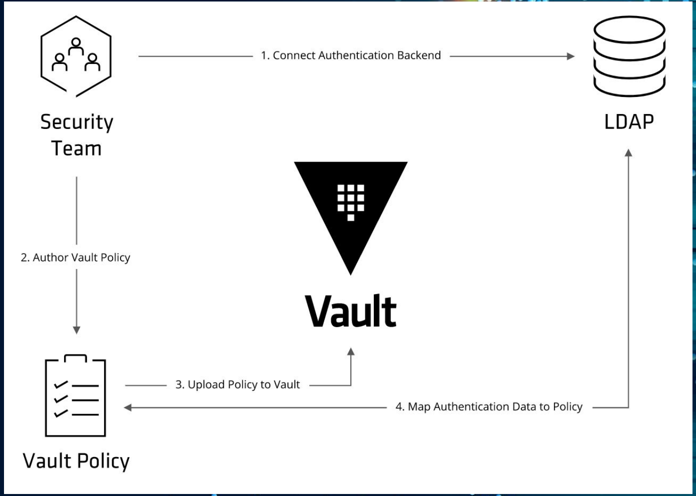
It's quick and easy to create a vault server instance with containers



```
sefoster@LAPTOP-VVL938IK:/mnt/c/msys64/home/susan/LosAlamosProject$ docker create --name hcvault1 --net=vaultnet -p 8200:8200 -h vaul
tsvr -e VAULT_ADDR=http://127.0.0.1:8200 -e VAULT_TOKEN=root -v $WORKDIR/plugins:/vault/plugins hashicorp/vault-enterprise:1.7.1_ent
server -dev -dev-plugin-dir=/vault/plugins -dev-root-token-id=root
d4e64659854a032c0414e97a3fa658e5c7797aab50c83427a53a6bdf383bf7df
sefoster@LAPTOP-VVL938IK:/mnt/c/msys64/home/susan/LosAlamosProject$ docker start hcvault1
hcvault1
sefoster@LAPTOP-VVL938IK:/mnt/c/msys64/home/susan/LosAlamosProject$ vault login root
WARNING! The VAULT_TOKEN environment variable is set! This takes precedence
over the value set by this command. To use the value set by this command,
unset the VAULT_TOKEN environment variable or set it to the token displayed
below.

Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.

Key                  Value
---                  -----
token                root
token_accessor       eIyQtLKK7I3PC4gMy7mww6sQ
token_duration       ∞
token_renewable      false
token_policies       ["root"]
identity_policies    []
policies             ["root"]
sefoster@LAPTOP-VVL938IK:/mnt/c/msys64/home/susan/LosAlamosProject$
```

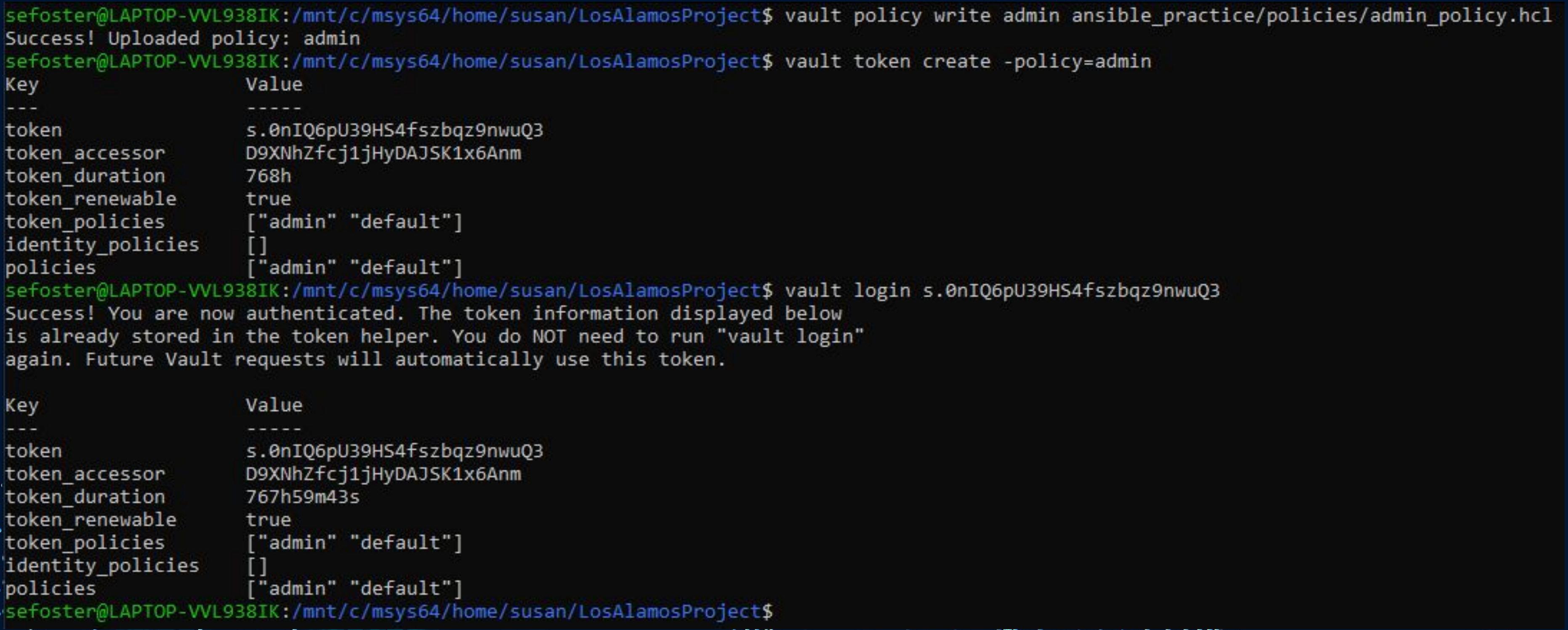
Before We Continue... What are Policies in Hashicorp Vault?

- Policies dictate who has what permissions in the Vault
- Policies are deny by default
- Assigned by group policies and managed by root or admin users
- Declares which Authentication Methods are allowed for each group and user
- Written in an HCL format


Security Team
1. Connect Authentication Backend
LDAP
2. Author Vault Policy
Vault
3. Upload Policy to Vault
4. Map Authentication Data to Policy
Vault Policy

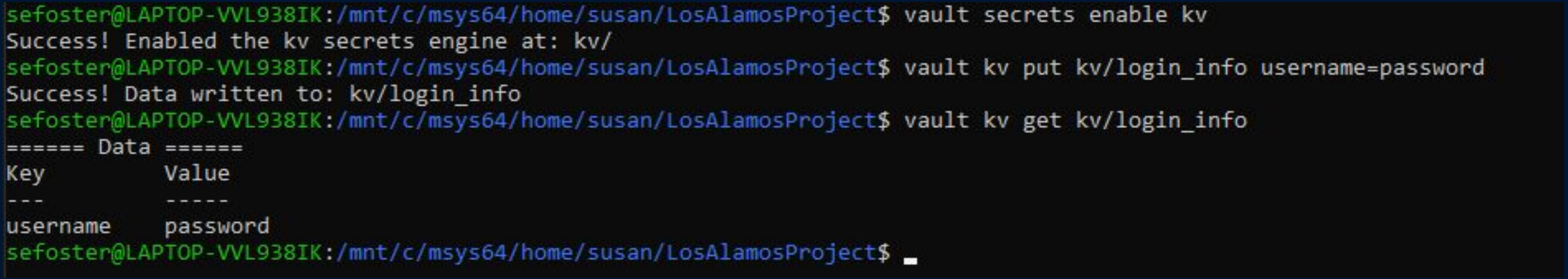
```
$ vault token create -policy=my-policy -policy=other-policy
```

Creating Policies and New Tokens



```
sefoster@LAPTOP-VVL938IK:/mnt/c/msys64/home/susan/LosAlamosProject$ vault policy write admin ansible_practice/policies/admin_policy.hcl
Success! Uploaded policy: admin
sefoster@LAPTOP-VVL938IK:/mnt/c/msys64/home/susan/LosAlamosProject$ vault token create -policy=admin
Key                  Value
---                  -----
token                s.0nIQ6pU39HS4fszbqz9nwuQ3
token_accessor       D9XNhZfcj1jHyDAJSK1x6Anm
token_duration       768h
token_renewable      true
token_policies       ["admin" "default"]
identity_policies    []
policies             ["admin" "default"]
sefoster@LAPTOP-VVL938IK:/mnt/c/msys64/home/susan/LosAlamosProject$ vault login s.0nIQ6pU39HS4fszbqz9nwuQ3
Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.

Key                  Value
---                  -----
token                s.0nIQ6pU39HS4fszbqz9nwuQ3
token_accessor       D9XNhZfcj1jHyDAJSK1x6Anm
token_duration       767h59m43s
token_renewable      true
token_policies       ["admin" "default"]
identity_policies    []
policies             ["admin" "default"]
sefoster@LAPTOP-VVL938IK:/mnt/c/msys64/home/susan/LosAlamosProject$
```

Writing Secrets



```
sefoster@LAPTOP-VVL938IK:/mnt/c/msys64/home/susan/LosAlamosProject$ vault secrets enable kv
Success! Enabled the kv secrets engine at: kv/
sefoster@LAPTOP-VVL938IK:/mnt/c/msys64/home/susan/LosAlamosProject$ vault kv put kv/login_info username=password
Success! Data written to: kv/login_info
sefoster@LAPTOP-VVL938IK:/mnt/c/msys64/home/susan/LosAlamosProject$ vault kv get kv/login_info
====== Data ======
Key         Value
---         -----
username    password
sefoster@LAPTOP-VVL938IK:/mnt/c/msys64/home/susan/LosAlamosProject$
```

That was Simple!

But how well does it work with Ansible?

Unsurprisingly, it's pretty easy in Ansible too!

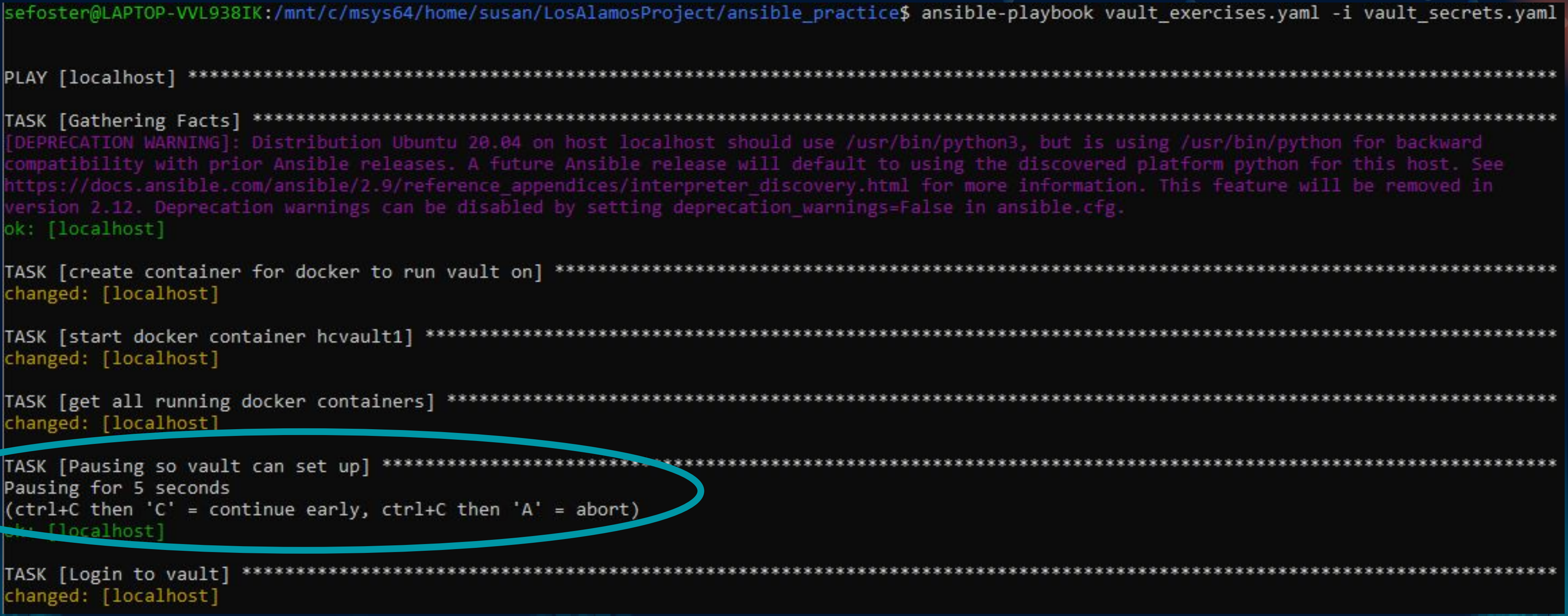
Let’s Try Making A Playbook!

Please note that this is not the entire playbook, some debugging tasks are excluded for readability

```
---
- hosts: localhost
  connection: local
  tasks:
    - name: "create container for docker to run vault on"
      command: docker create --name hcvault1 --net=vaultnet -p
8200:8200 -h vaultsvr -e VAULT_ADDR=http://127.0.0.1:8200 -e
VAULT_TOKEN=root -v $WORKDIR/plugins:/vault/plugins
hashicorp/vault-enterprise:1.7.1_ent server -dev
-dev-plugin-dir=/vault/plugins -dev-root-token-id=root
      register: docker_container
    - name: "start docker container hcvault1"
      command: docker start hcvault1
      register: docker_start
    - name: "Login to vault"
      command: vault login root -address=http://127.0.0.1:8200
      register: log_vault
    - name: "enable kv engine on the vault server"
      command: vault secrets enable kv
      register: enable_kv
    - name: "Create an admin policy for dev access"
      command: vault policy write admin
policies/admin_policy.hcl
      register: admin_policy
```

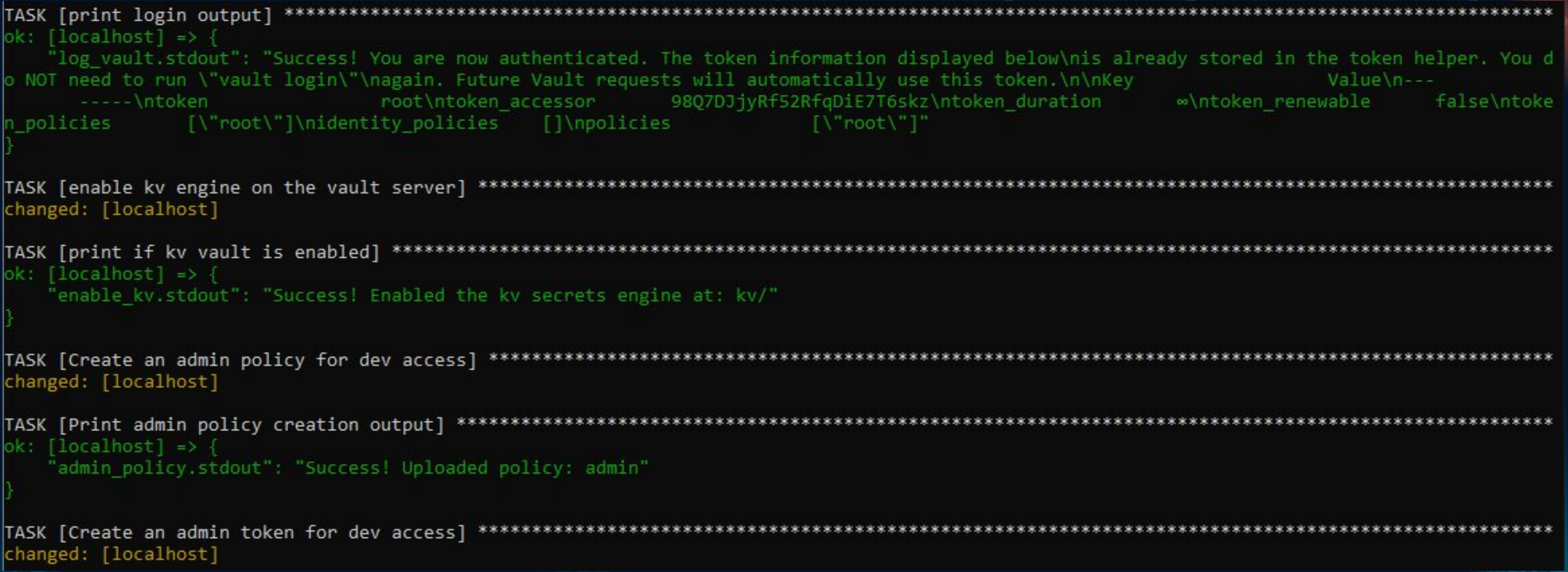
```
    - name: "Create an admin token for dev access"
      command: vault token create -policy=admin
      register: admin_token_full
    - name: "Writing admin token to file"
      copy:
        dest: admin-tokens.txt
        content: "{{ admin_token_full.stdout }}\n"
      name: "Isolating token information"
      shell: grep "token " $WORKDIR/admin-tokens.txt | grep -Go
"[^token^\s*][\.a-zA-Z0-9]*" | sed -e 's/^\s*//' -e '/^$/d'
      args:
        warn: no
      register: admin_token
    - name: "loging in as admin"
      command: vault login {{ admin_token.stdout }}
-address=127.0.0.1:8200
      register: log_vault
    - name: "write a secret to vault"
      command: vault kv put kv/location key=value
      register: write_vault
    - name: "collect secret from vault using command line"
      command: vault kv get kv/location
      register: get_vault
```

Now Let's See It In Action (1)



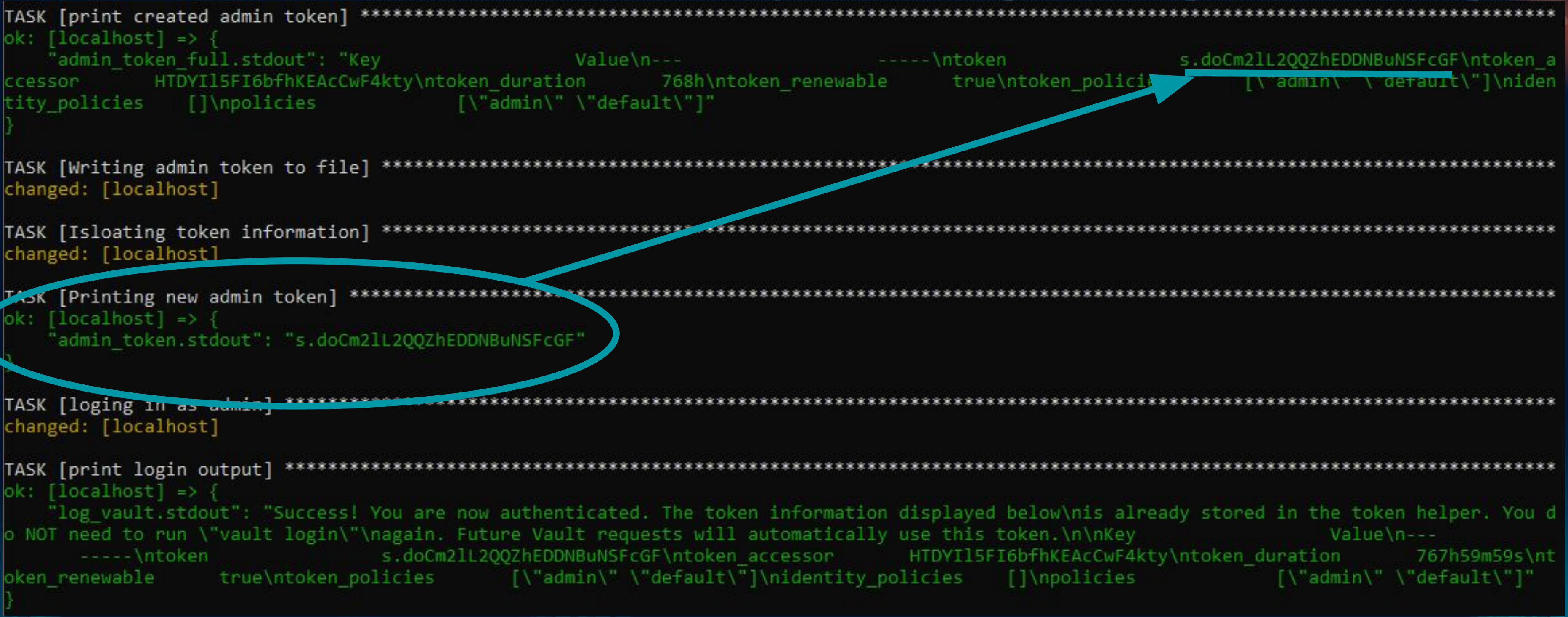
```
sefoster@LAPTOP-VVL938IK:/mnt/c/msys64/home/susan/LosAlamosProject/ansible_practice$ ansible-playbook vault_exercises.yaml -i vault_secrets.yaml


PLAY [localhost] ************************************************************************************************************

TASK [Gathering Facts] ******************************************************************************************************
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host localhost should use /usr/bin/python3, but is using /usr/bin/python for backward
compatibility with prior Ansible releases. A future Ansible release will default to using the discovered platform python for this host. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information. This feature will be removed in
version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
ok: [localhost]

TASK [create container for docker to run vault on] **************************************************************************
changed: [localhost]

TASK [start docker container hcvault1] **************************************************************************************
changed: [localhost]

TASK [get all running docker containers] ************************************************************************************
changed: [localhost]

TASK [Pausing so vault can set up] ******************************************************************************************
Pausing for 5 seconds
(ctrl+C then 'C' = continue early, ctrl+C then 'A' = abort)
ok: [localhost]

TASK [Login to vault] *******************************************************************************************************
changed: [localhost]
```

Now Let’s See It In Action (2)



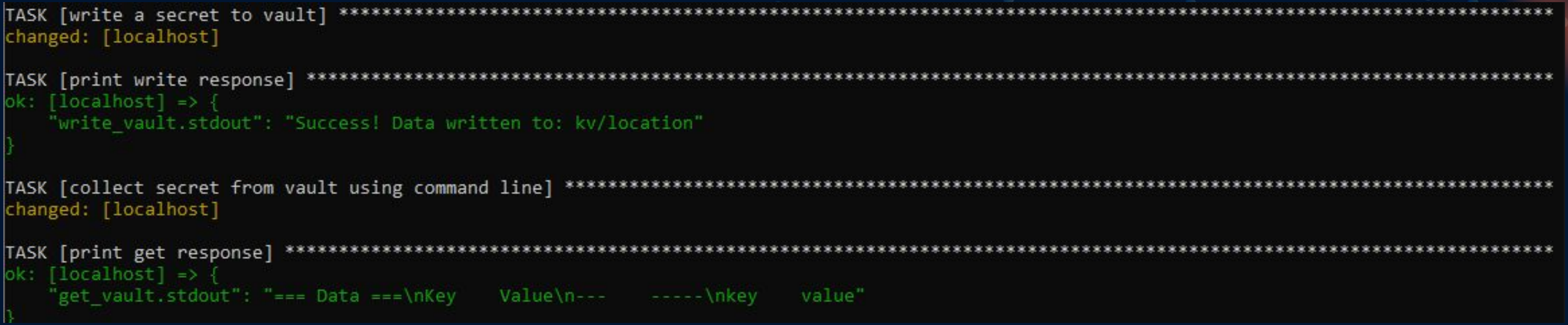
```
TASK [print login output] **********************************************************************************************************************
ok: [localhost] => {
    "log_vault.stdout": "Success! You are now authenticated. The token information displayed below\nis already stored in the token helper. You d
o NOT need to run \"vault login\"\nagain. Future Vault requests will automatically use this token.\n\nKey                  Value\n---
       -----\ntoken                root\ntoken_accessor       98Q7DJjyRf52RfqDiE7T6skz\ntoken_duration       ∞\ntoken_renewable      false\ntoke
n_policies       [\"root\"]\nidentity_policies    []\npolicies             [\"root\"]"
}

TASK [enable kv engine on the vault server] ****************************************************************************************************
changed: [localhost]

TASK [print if kv vault is enabled] ************************************************************************************************************
ok: [localhost] => {
    "enable_kv.stdout": "Success! Enabled the kv secrets engine at: kv/"
}

TASK [Create an admin policy for dev access] ***************************************************************************************************
changed: [localhost]

TASK [Print admin policy creation output] ******************************************************************************************************
ok: [localhost] => {
    "admin_policy.stdout": "Success! Uploaded policy: admin"
}

TASK [Create an admin token for dev access] ****************************************************************************************************
changed: [localhost]
```

Now Let’s See It In Action (3)



```
TASK [print created admin token] ****************************************************************************************
ok: [localhost] => {
    "admin_token_full.stdout": "Key                    Value\n---                    -----\ntoken                 s.doCm2lL2QQZhEDDNBuNSFcGF\ntoken_a
ccessor       HTDYIl5FI6bfhKEAcCwF4kty\ntoken_duration       768h\ntoken_renewable      true\ntoken_policies       [\"admin\" \"default\"]\niden
tity_policies    []\npolicies             [\"admin\" \"default\"]"
}

TASK [Writing admin token to file] **************************************************************************************
changed: [localhost]

TASK [Isloating token information] **************************************************************************************
changed: [localhost]

TASK [Printing new admin token] *****************************************************************************************
ok: [localhost] => {
    "admin_token.stdout": "s.doCm2lL2QQZhEDDNBuNSFcGF"
}

TASK [loging in as admin] ***********************************************************************************************
changed: [localhost]

TASK [print login output] ***********************************************************************************************
ok: [localhost] => {
    "log_vault.stdout": "Success! You are now authenticated. The token information displayed below\nis already stored in the token helper. You d
o NOT need to run \"vault login\"\nagain. Future Vault requests will automatically use this token.\n\nKey                  Value\n---
       -----\ntoken                s.doCm2lL2QQZhEDDNBuNSFcGF\ntoken_accessor       HTDYIl5FI6bfhKEAcCwF4kty\ntoken_duration       767h59m59s\nt
oken_renewable      true\ntoken_policies       [\"admin\" \"default\"]\nidentity_policies    []\npolicies             [\"admin\" \"default\"]"
}
```

Now Let's See It In Action (4)



```
TASK [write a secret to vault] ****************************************************************************************
changed: [localhost]

TASK [print write response] *******************************************************************************************
ok: [localhost] => {
    "write_vault.stdout": "Success! Data written to: kv/location"
}

TASK [collect secret from vault using command line] *******************************************************************
changed: [localhost]

TASK [print get response] *********************************************************************************************
ok: [localhost] => {
    "get_vault.stdout": "=== Data ===\nKey    Value\n---    -----\nkey    value"
}
```

That worked out pretty well, but using command line can be a little clunky and hard to read, does Ansible have anything to make this a little easier?

Ansible does, Hashicorp even maintains a useful plugin

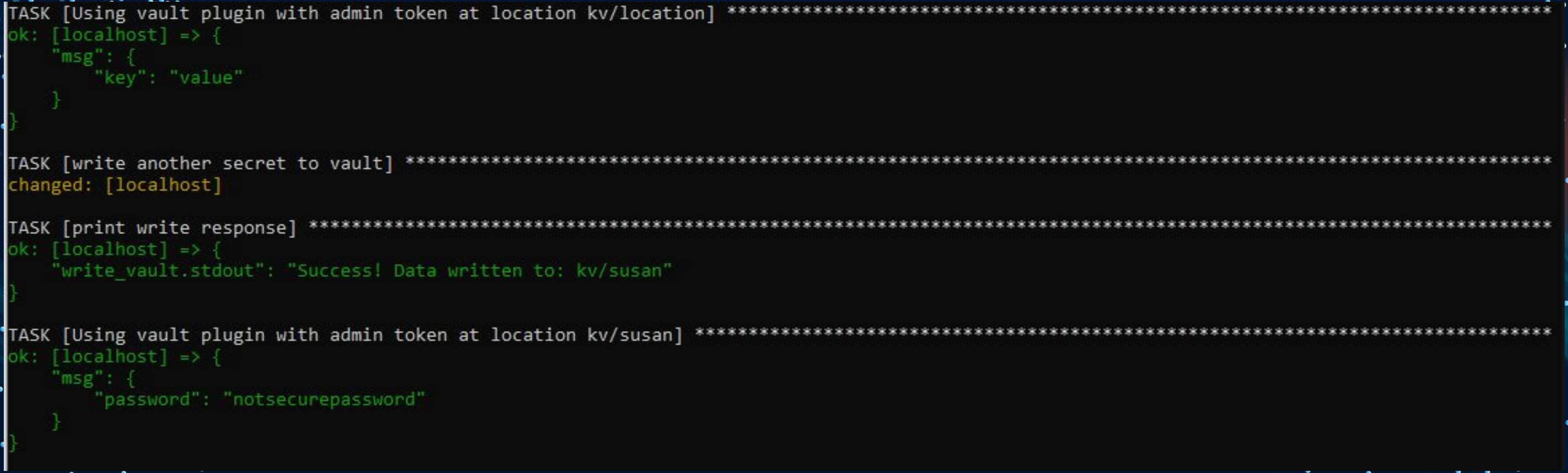
The Ansible Plugin: hashi_vault

hashi_vault is an Ansible lookup plugin that queries on the Hashicorp Vault server

Syntax example in a playbook:

```
    - name: "Using vault plugin with
admin token at location kv/{{
secret_tuple_template[0] }}"
      debug:
        msg: "{{ lookup('hashi_vault',
'secret=kv/{{ secret_tuple_template[0] }}
token={{ admin_token.stdout }} url={{
vault_address }}') }}"
```

Let’s see it in action…



```
TASK [Using vault plugin with admin token at location kv/location] ***************************************************************
ok: [localhost] => {
    "msg": {
        "key": "value"
    }
}

TASK [write another secret to vault] *****************************************************************************************
changed: [localhost]

TASK [print write response] *************************************************************************************************
ok: [localhost] => {
    "write_vault.stdout": "Success! Data written to: kv/susan"
}

TASK [Using vault plugin with admin token at location kv/susan] *************************************************************
ok: [localhost] => {
    "msg": {
        "password": "notsecurepassword"
    }
}
```

Documentation Used and Special Thanks

- https://docs.ansible.com/ansible/2.3/index.html
- https://www.vaultproject.io/docs

- Christian Storer
- Marc Santoro
- Conner Whitfield
- Matt Brown and Luke McCleary from Hashicorp
- And all of our bootcamp mentors for introducing us to HPC!