ANSIBLE CONNECTION SECURITY ENHANCING WITH OTP_SSH

Presentor: Huong Nguyen

Under the guidance of Mentor: Duc Nguyen

Agenda

1. Key management solution – Harshicorp Vault
2. OTP – SSH with VAULT
3. Implementation & Result

1. Key management solution – Harshicorp Vault

What is Vault?

- Secret management service / Secret as a service
- Secured secret storage
- Key rolling

Use Cases

- General Secret Storage
- Data Encryption
- Identity – Based Access
- Key Management

1. Key management solution – Harshicorp Vault

Vault key features:

- Secure secret storage
- Dynamic secrets
- Data encryption
- Leasing and renewal
- Revocation

1. Key management solution – Harshicorp Vault

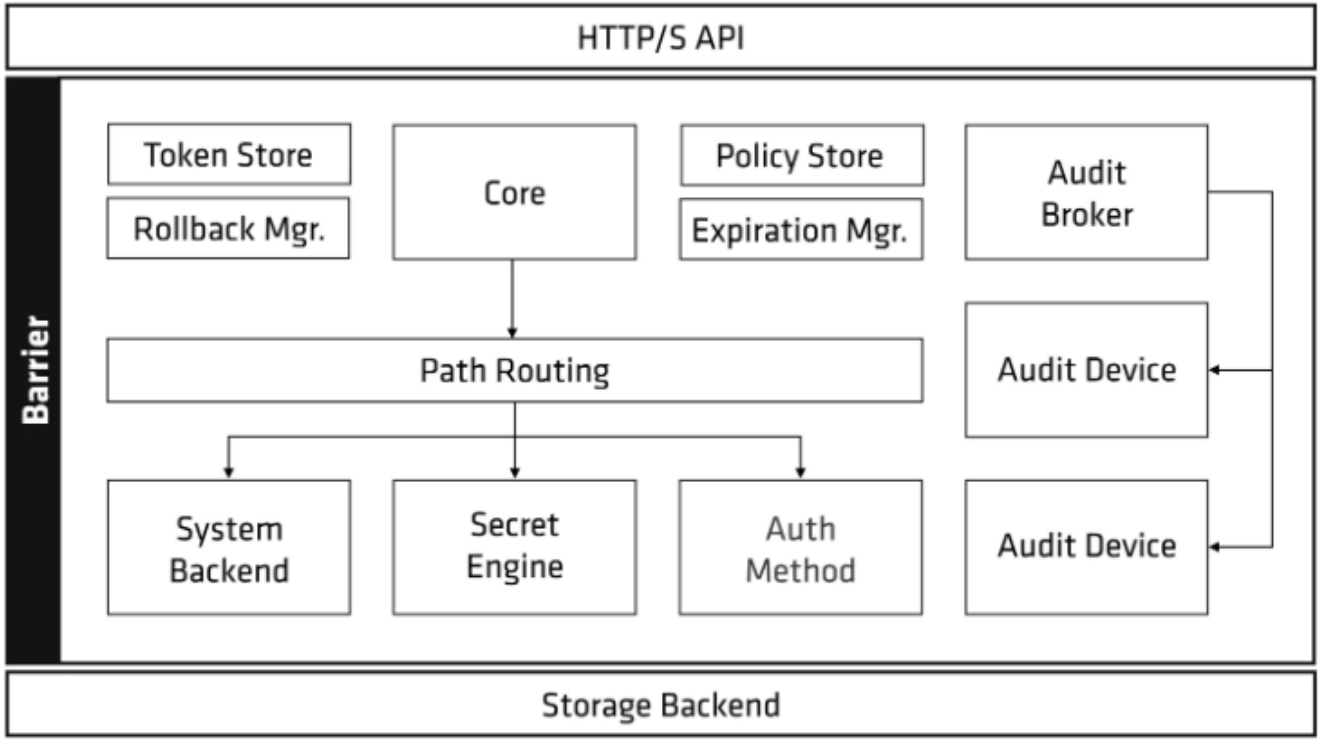


Fig 1.1 Vault architecture

1. Key management solution – Harshicorp Vault

Vault initialization concept:

- Share key, Master key, encryption key
- Master key is never stored anywhere
- Vault always starts sealed
- Unsealing requires multiple key shares
- Decryption key is kept in locked memory

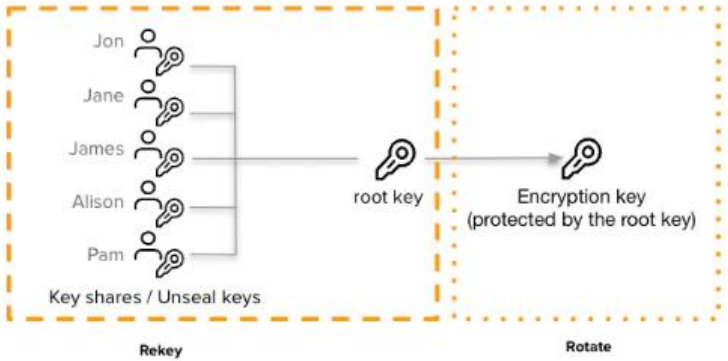


Fig 1.2 Shamir's secret sharing

1. Key management solution – Harshicorp Vault

1.1 Secret Engine :

- Store, generate, encrypt data.
- Lifecycle: enable, disable, move, tune
- Types of SE: Active Directory, AliCloud, AWS, Azure, Consul, Cubbyhole, Databases, GC, SSH, OpenLDAP…

1. Key management solution – Harshicorp Vault

1.2 Auth Method :

- Perform authentication and are responsible for assigning identity and a set of policies to a user.
- Types of AM: AppRole, AliCloud, AWS, Azure, Github, TLS cert, Tokens, Username & Password…

1. Key management solution – Harshicorp Vault

1.3 Audit Device :

- Collectively keep a detailed log of all requests and response to Vault
- Audit log contains every authenticated interaction with Vault, including errors.
- Types of AD: File audit device, syslog audit device, socket

2. OTP – SSH with Vault

2.1 SSH secret engine

- Providing secure authentication and authorization for access to machines via the SSH protocol
- Managing access to machine infrastructure, providing several ways to issue SSH credentials
- Supporting 3 modes: Signed SSH certificates, One – time SSH Passwords, Dynamic SSH keys.

2. OTP – SSH with Vault

Action	Method	Path
Create/ Update role	POST	/ssh/keys/:name
Delete key	DELETE	/ssh/keys/:name
Create role	POST	/ssh/roles/:name
Read role	GET	/ssh/roles/:name
Delete role	DELETE	/ssh/roles/:name
Generate SSH Credentials	POST	/ssh/creds/:name
Verify SSH OTP	POST	/ssh/verify

Table 2.1 SSH secret engine API

2. OTP – SSH with Vault

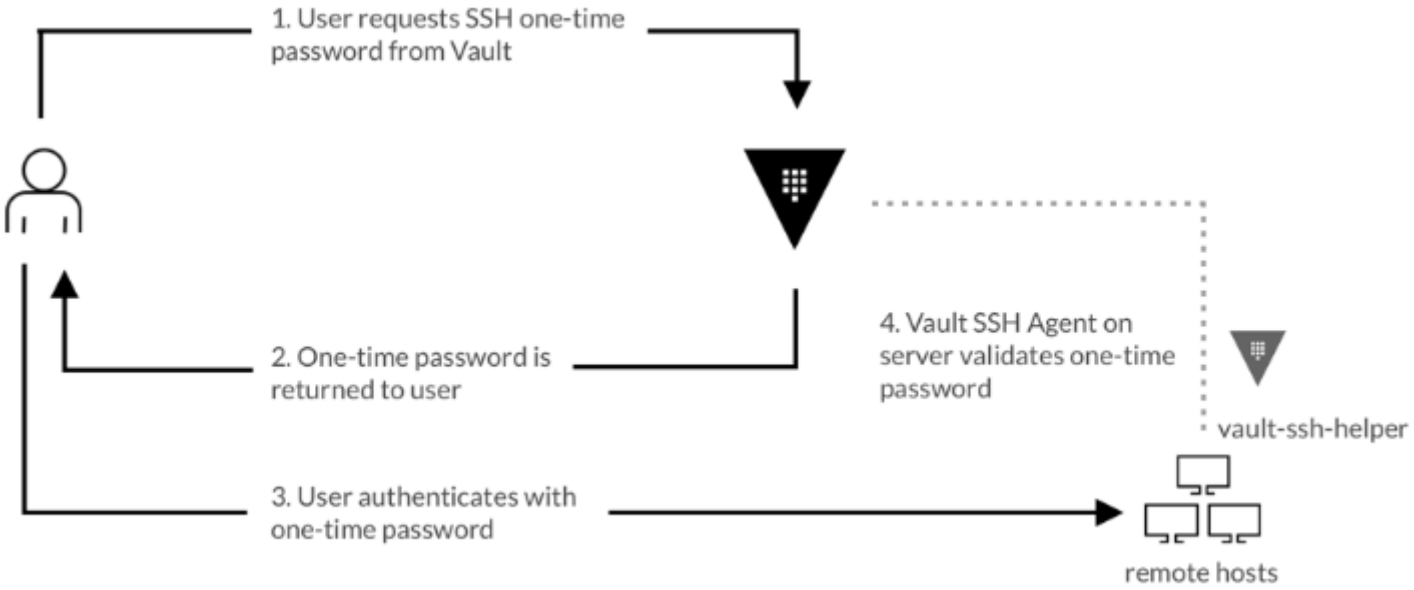


Fig 2.1 Workflow of mode: OTP – SSH with Vault

2. OTP – SSH with Vault

Vault – ssh – helper:

- A counterpart to HashiCorp Vault's SSH backend.
- Allows a machine to consume One-Time-Passwords (OTP) created by Vault servers
- Install Vault – ssh – helper in remote host.
- Vault-ssh-helper's binary is run as an external command using *pam_exec.so*

2. OTP – SSH with Vault

Vault – ssh – helper usage

Option	Description
Verify - only	Verifies that vault – ssh – helper is install correctly and is able to communicate with Vault
config	The path to the configuration file.
dev	Vault – ssh – helper communicates with Vault with TLS disable.

Table 2.2 Vault – ssh – helper usage

2. OTP – SSH with Vault

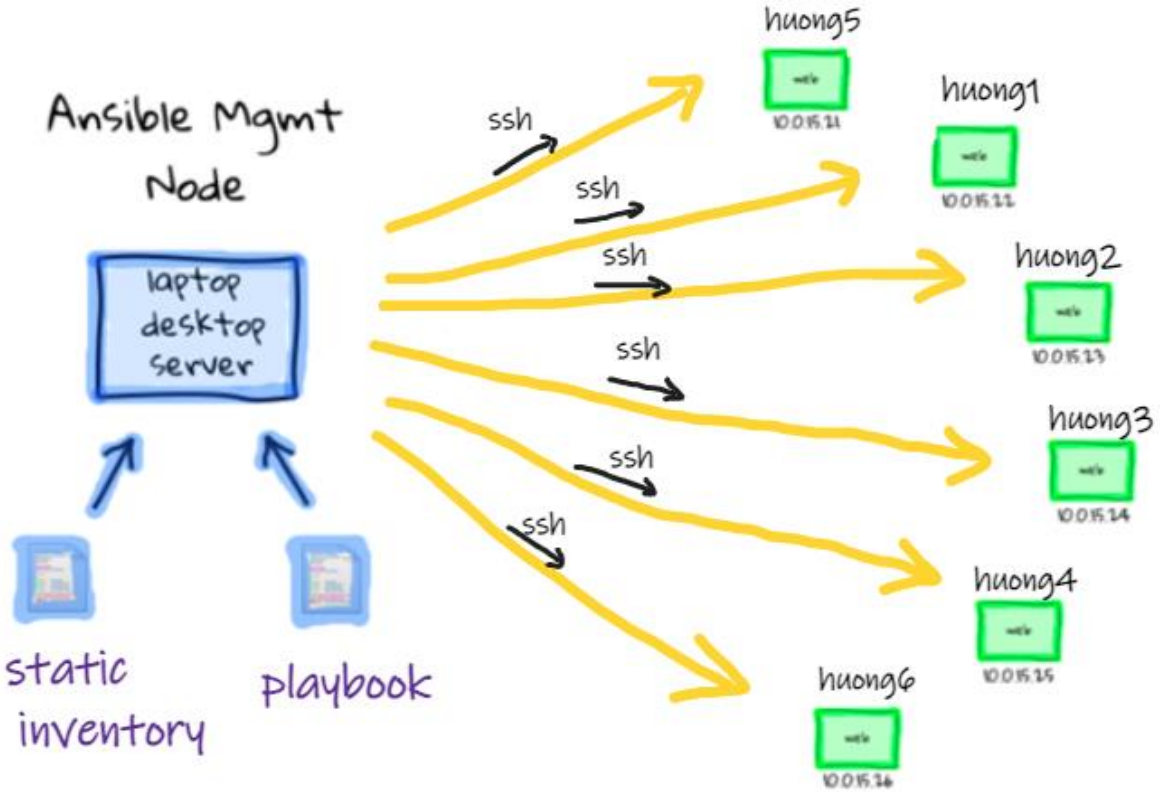
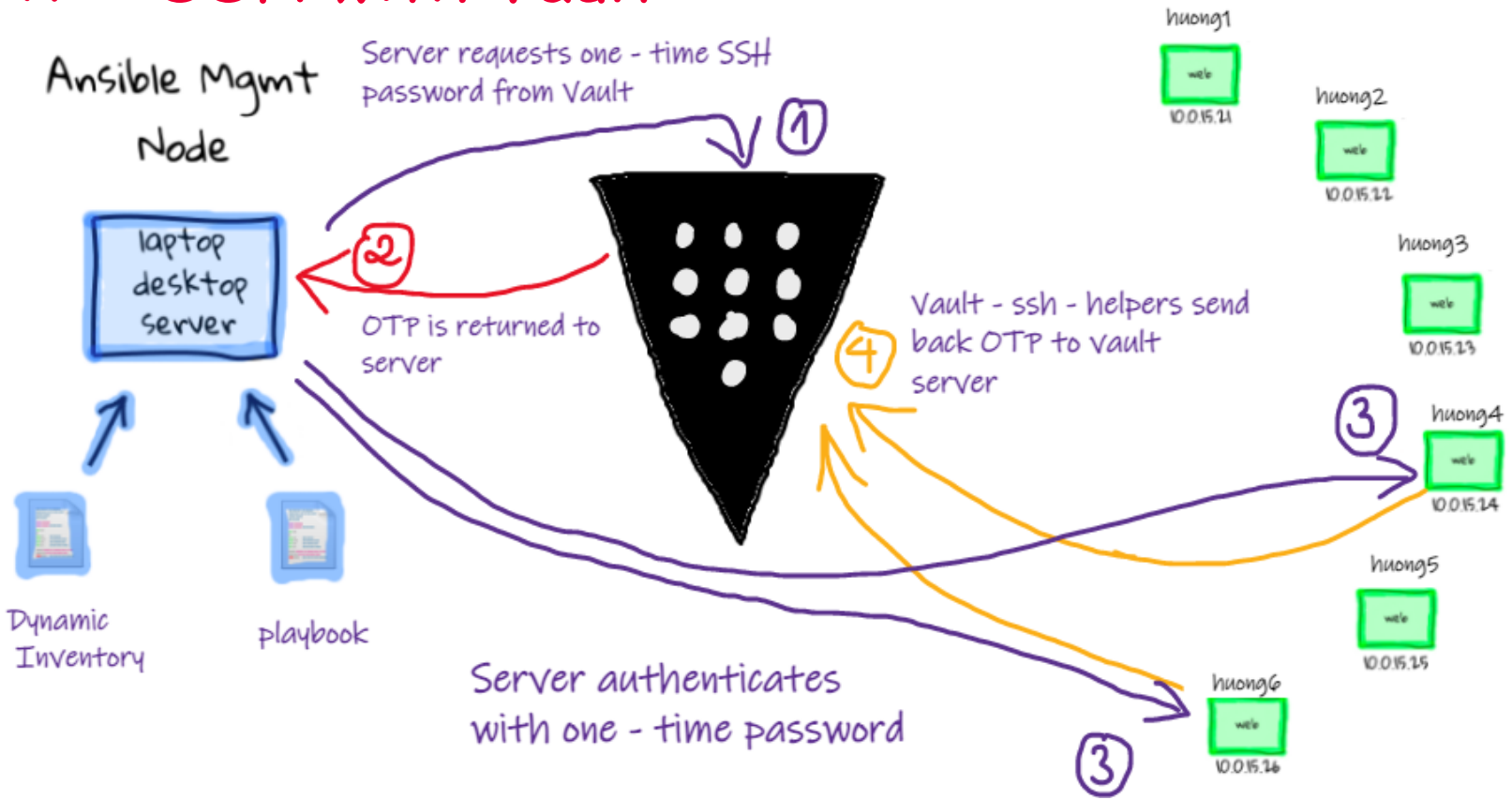


Fig 2.2 Previous Ansible architecture

2. OTP – SSH with Vault



3. Implementation & Result

3.1 Start Vault

- Access to Vault server

```
huong@py:~$ vault server -dev -dev-root-token-id root -dev-listen-address 10.208.182.77:8200
```

- Enable SSH secret engine

```
huong@py:~$ vault secrets enable ssh
Success! Enabled the ssh secrets engine at: ssh/
huong@py:~$
```

- Create a role

```
huong@py:~$ vault write ssh/roles/otp_key_role \
    key_type=otp \
    default_user=huong \
    cidr_list=10.208.180.0/22
Success! Data written to: ssh/roles/otp_key_role
huong@py:~$
```

3. Implementation & Result

3.1 Start Vault

- Create policy to allow access to the SSH OTP role and then attach the policy to an authentication method.

```
huong@py:~$ tee test.hcl <<EOF
# To list SSH secrets paths
path "ssh/*" {
  capabilities = [ "list" ]
}
# To use the configured SSH secrets engine otp_key_role role
path "ssh/creds/otp_key_role" {
  capabilities = ["create", "read", "update"]
}
EOF
```

- Enable the userpass auth method and create a user assign to test policy

```
huong@py:~$ vault auth enable userpass
Success! Enabled userpass auth method at: userpass/
huong@py:~$ vault write auth/userpass/users/ubuntu password="training" policies="test"
Success! Data written to: auth/userpass/users/ubuntu
huong@py:~$
```

3. Implementation & Result

3.2 Config remote hosts

- Install and config Vault – ssh - helper

```
$ sudo tee /etc/vault-ssh-helper.d/config.hcl <<EOF
vault_addr = "$VAULT_EXTERNAL_ADDR"
tls_skip_verify = false
ssh_mount_point = "ssh"
allowed_roles = "*"
EOF
```

- Mofdify PAM sshd configuration and sshd configuration

3. Implementation & Result

3.3 Server gets OTP and implements SSH

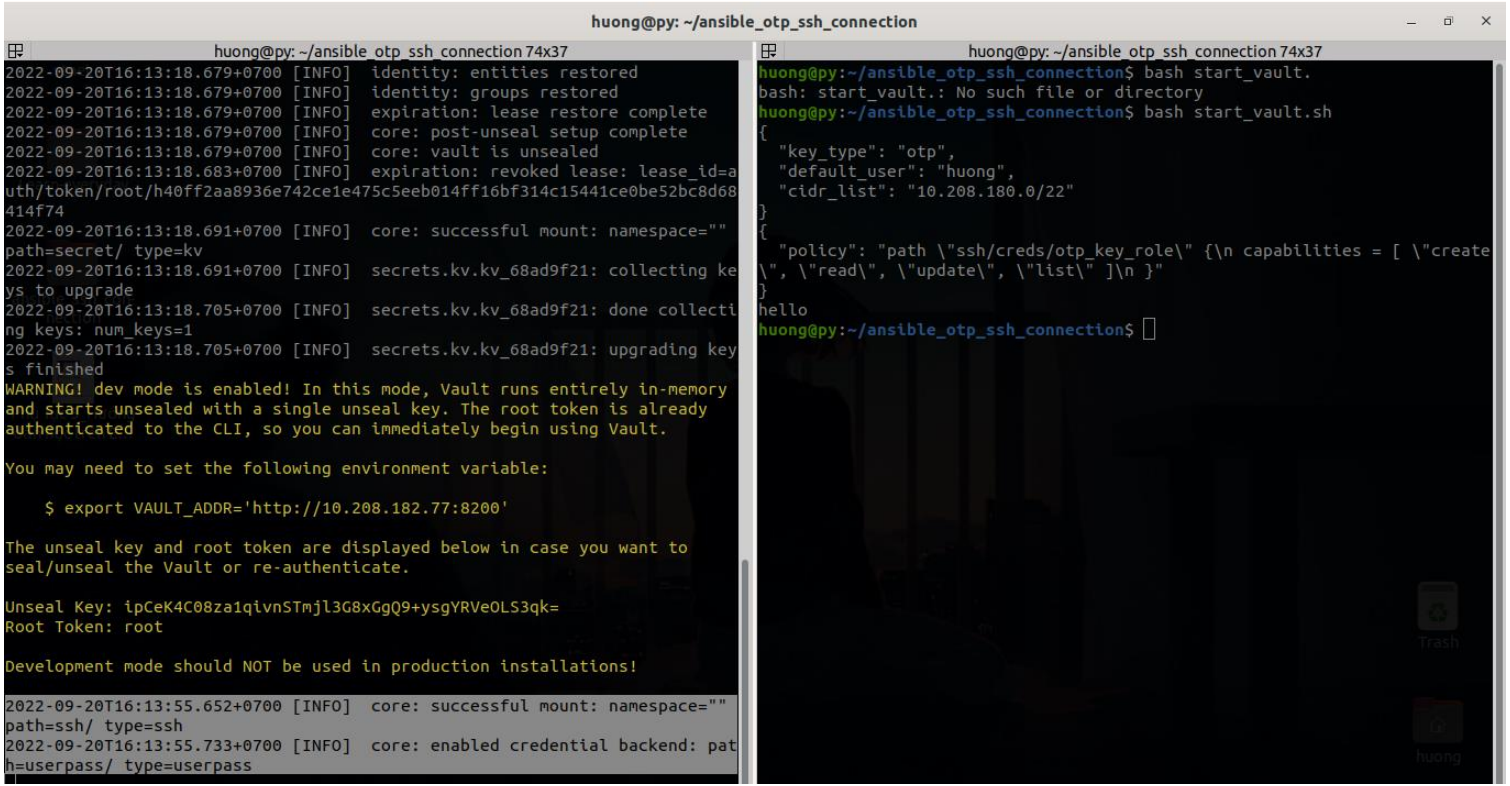
- Get OTP via API

```
#!/bin/bash
VAULT_ADDR='http://10.61.131.188:8200'
UBUNTU_TOKEN=$(curl --silent --request POST  --data '{"password": "training"}' $VAULT_ADDR/v1/auth/userpass/login/ubuntu | jq -r '.auth | .client_token')
TOKEN1=$(curl --silent --header "X-Vault-Token: $UBUNTU_TOKEN" --request POST --data '{"ip": "'"10.61.131.188"'"}' $VAULT_ADDR/v1/ssh/creds/otp_key_role | jq -r .data.key)
#TOKEN2=$(curl --silent --header "X-Vault-Token: $UBUNTU_TOKEN" --request POST --data '{"ip": "'"10.0.0.1"'"}' $VAULT_ADDR/v1/ssh/creds/otp_key_role | jq -r .data.key)
```

- Using Script plugin to generate dynamic inventory

3. Implementation & Result

3.3 Result



3. Implementation & Result

3.3 Result

```
huong@py:~/ansible_otp_ssh_connection$ /home/huong/ansible_otp_ssh_connection/otp_generate.py --list
{
  "_meta": {
    "hostvars": {
      "huong": {
        "ansible_host": "10.208.182.77",
        "ansible_ssh_pass": "5f5c8493-1526-092c-e5a8-0b80528671bf"
      }
    }
  },
  "all": {
    "children": [
      "ungrouped"
    ]
  },
  "ungrouped": {
    "hosts": [
      "huong"
    ]
  }
}
huong@py:~/ansible_otp_ssh_connection$ /home/huong/ansible_otp_ssh_connection/otp_generate.py --list
{
  "_meta": {
    "hostvars": {
      "huong": {
        "ansible_host": "10.208.182.77",
        "ansible_ssh_pass": "33e97fce-cfc1-e991-e2c5-6720238c3c73"
      }
    }
  },
  "all": {
    "children": [
      "ungrouped"
```

3. Implementation & Result

```
huong@py:~/ansible_otp_ssh_connection$ /home/huong/ansible_otp_ssh_connection/otp_generate.py --list
{
  "_meta": {
    "hostvars": {
      "huong": {
        "ansible_host": "10.208.182.77",
        "ansible_ssh_pass": "3cfc509e-c5b0-7344-da7c-be03df366e81"
      }
    }
  },
  "all": {
    "children": [
      "ungrouped"
    ]
  },
  "ungrouped": {
    "hosts": [
      "huong"
    ]
  }
}
huong@py:~/ansible_otp_ssh_connection$ ansible --inventory otp_generate.py all -m ping
huong | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
huong@py:~/ansible_otp_ssh_connection$
```

References

- One-Time SSH Passwords (OTP) - SSH - Secrets Engines | Vault | HashiCorp Developer
- Secure Shell: How Does SSH Work (slashroot.in)
- SSH - Secrets Engines | Vault | HashiCorp Developer
- hashicorp/vault-ssh-helper: Vault SSH Agent is used to enable one time keys and passwords (github.com)
- ansible/ansible.md at master · HaManhDong/ansible (github.com)
- Inventory plugins — Ansible Documentation
- How to write a Python script to create dynamic Ansible inventories | Enable Sysadmin (redhat.com)

Thanks!

Any questions?

You can find me at:

@huongnt499

huong.set@gmail.com