

AWX 201: Advanced Automation Techniques with the Ansible AWX Platform

Tim Glen
Security Solutions Engineer



cisco Live !



Cisco Webex App

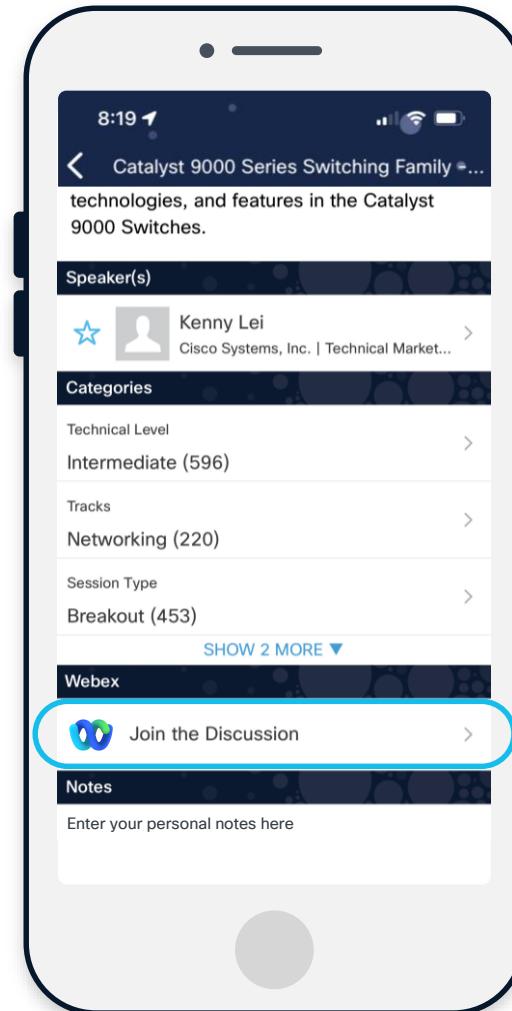
Questions?

Use Cisco Webex App to chat with the speaker after the session

How

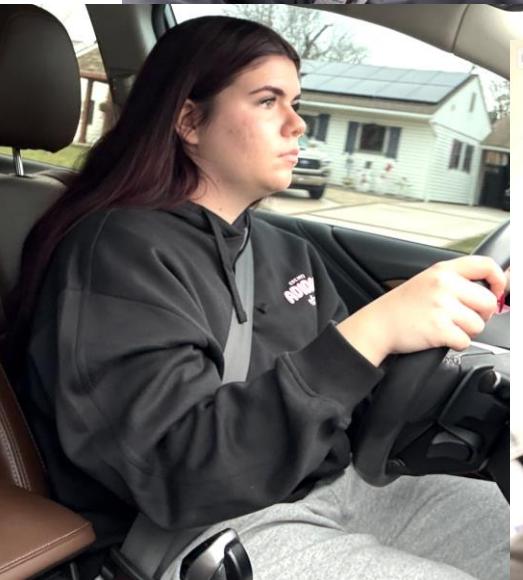
- 1 Find this session in the Cisco Live Mobile App
- 2 Click “Join the Discussion”
- 3 Install the Webex App or go directly to the Webex space
- 4 Enter messages/questions in the Webex space

Webex spaces will be moderated by the speaker until June 13, 2025.



Tim (Personal)

- Human
- Husband
- Father
- Dog Dad
- Outside
- Biking
- Driving
- Travel



Tim (Professional)

- Started in IT in 1995, Telephone Tech Support
- Worked 23 years at Web Hosting Provider
 - Managed all routers, switches, firewalls, wireless, security
- Worked at Cisco 6 years
 - Security Systems Engineer



github.com/timmayg



linkedin.com/in/timglen



cs.co/TimGlen



Agenda

Press here to
get started



- 1 Introduction
- 2 Execution Environments
- 3 Custom Credentials with HashiCorp Vault
- 4 Certificate Automation
- 5 Conclusion

Check Here for Updates



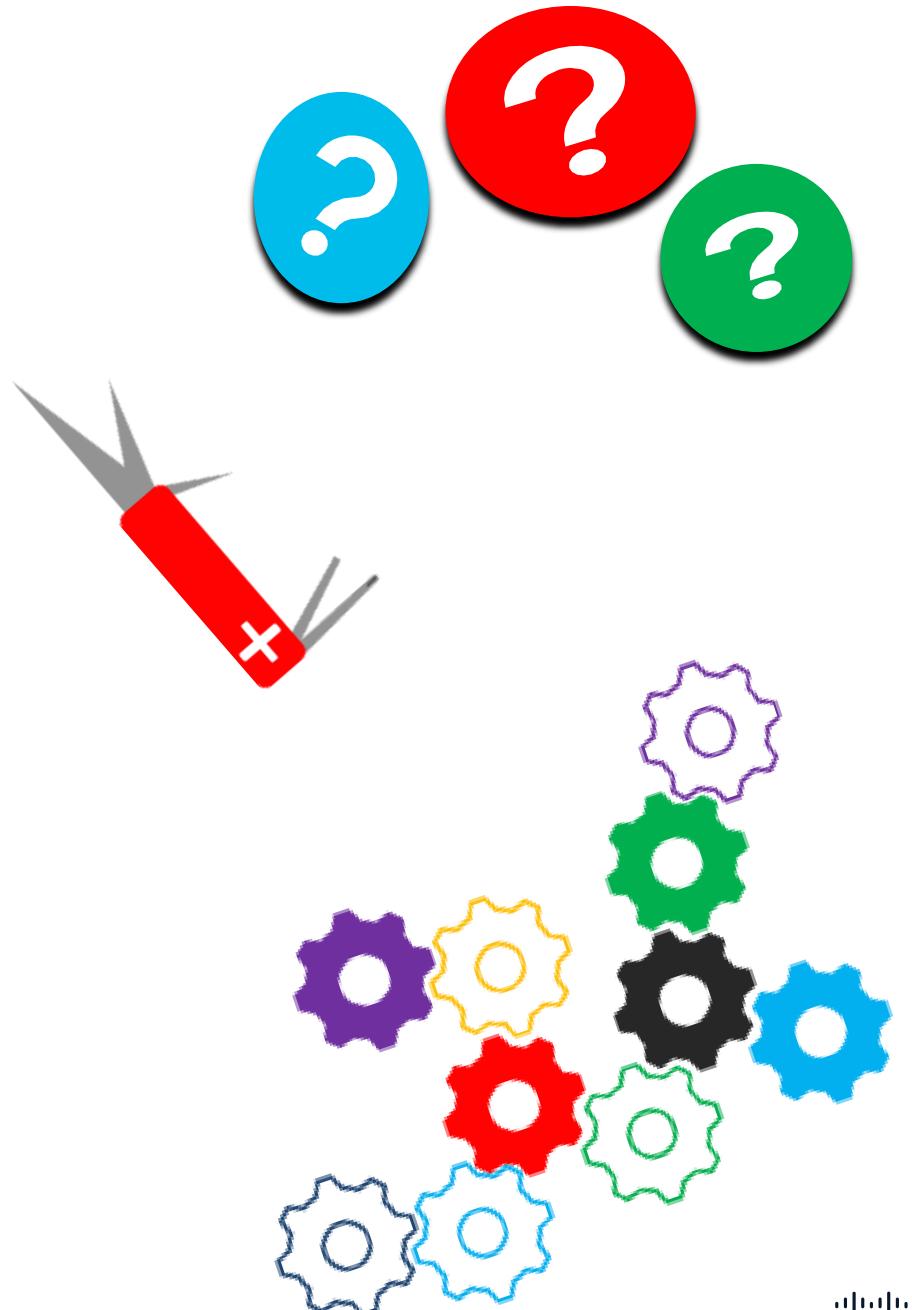
<https://github.com/timmayg/devnet-2517>



A screenshot of a GitHub repository page. The title of the repository is "2025 Cisco Live San Diego". The main content of the README file is a slide for a session titled "AWX 201: Advanced Automation Techniques with the Ansible AWX Platform" presented by Tim Glen, a Security Solutions Engineer, at Cisco Live, San Diego, on Thursday, June 12, at 11:30 am. The slide features the AWX logo, the Cisco Live logo, and logos for Let's Encrypt and HashiCorp Vault. Below the slide, the README text reads: "Welcome to the repository for 'AWX 201: Advanced Automation Techniques with the Ansible AWX Platform and GitHub Copilot' presented at Cisco Live San Diego 2025! This session, DEVNET-2517, is designed to help you dive deeper into the Ansible AWX Platform. During this session, we'll cover the we will learn some advanced AWX topics like:"

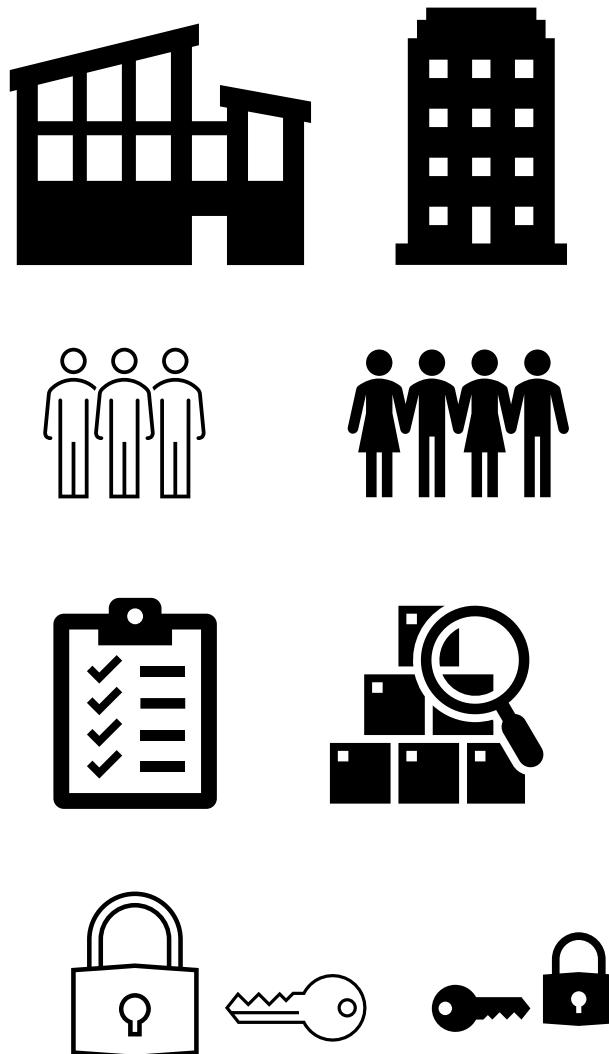
Why Ansible, Why AWX

- Ansible is tool of choice
- Flexibility – Manages Everything
- Easy to write playbooks
- AWX is the next step
- Powerful task engine!
- AWX Web UI & REST API
- Scheduler, Logging



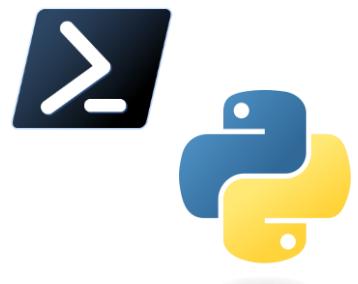
AWX - High Level

- Ansible Automation Platform
- Orgs, Teams & Users
- Templates & Projects
- Hosts & Inventories
- Credentials & Credential Types
- Instances, scaling
- Execution Environments

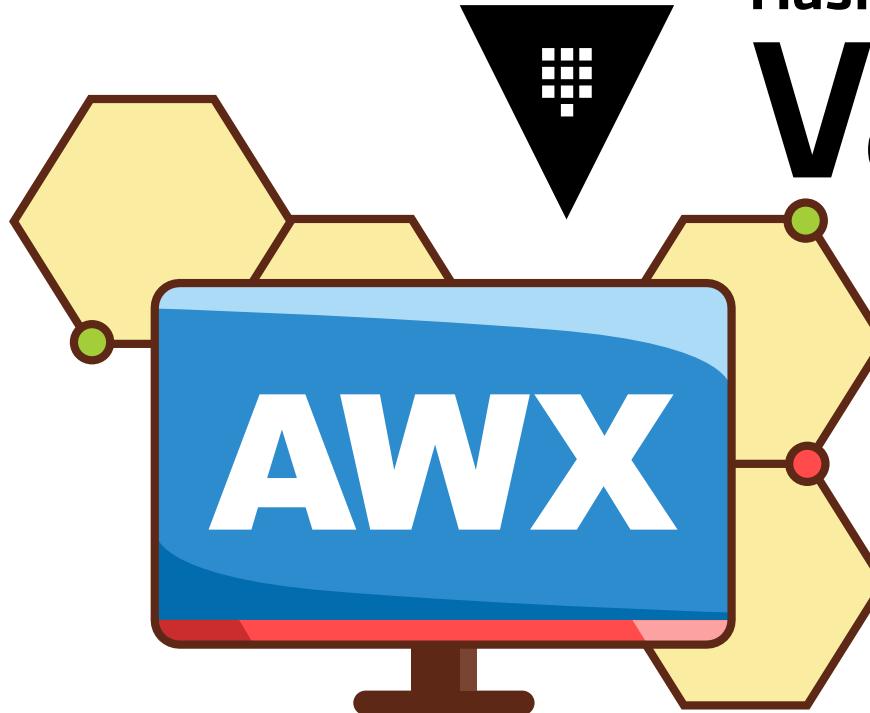


AWX Integrations & Compatibility

 **Let's Encrypt**

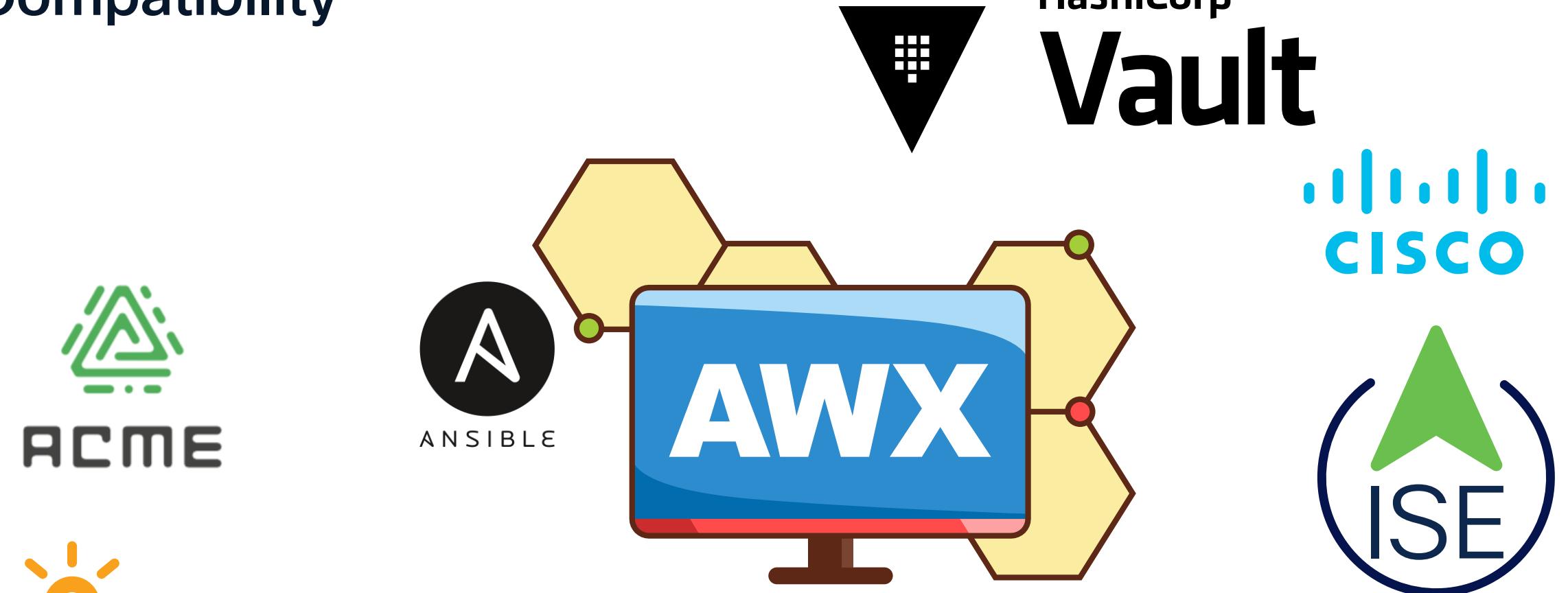


 **Terraform**
 **serviceNow™**



PagerDuty

AWX Integrations & Compatibility



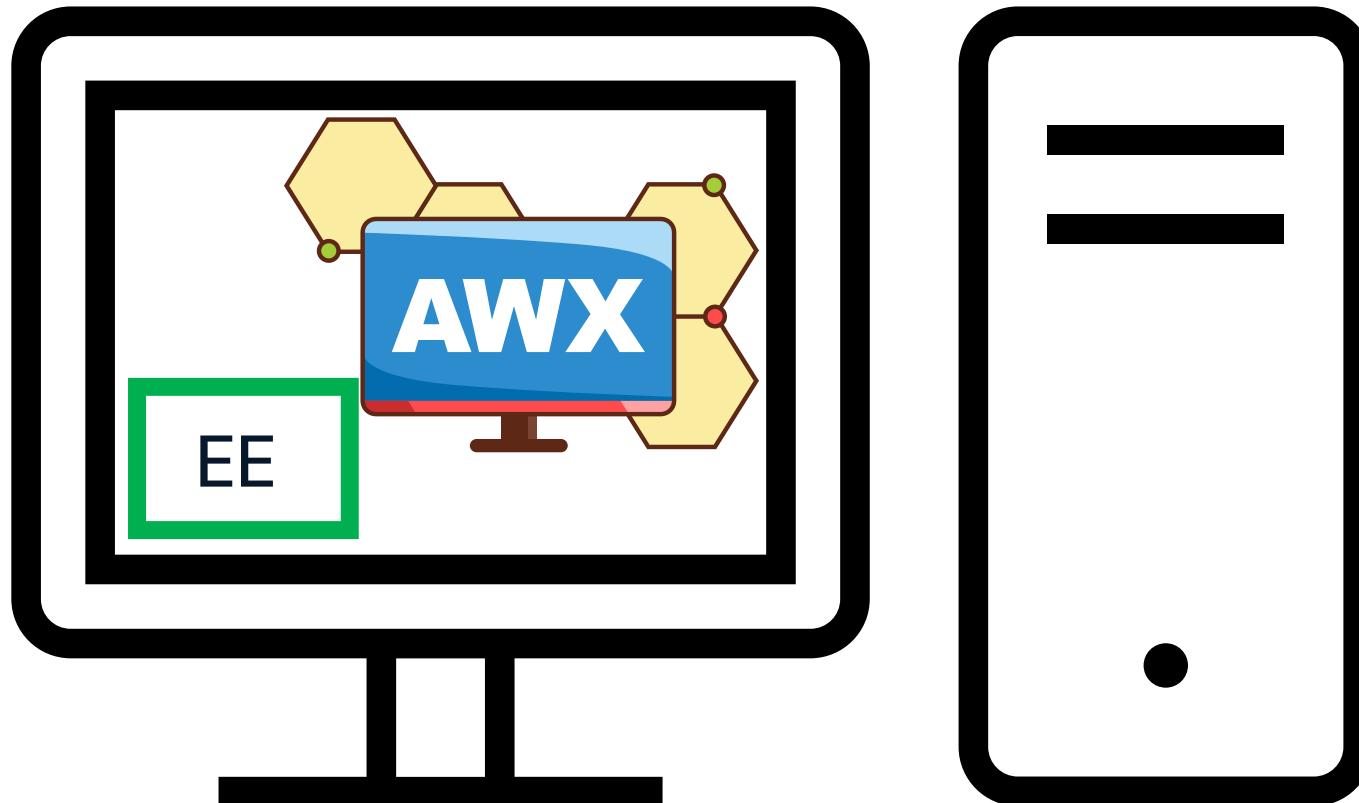
Run ISE Cert Job RIGHT NOW

Label: demo

The screenshot shows a software interface with a light blue gradient background. At the top, there is a navigation bar with icons for back, forward, and search. Below the navigation bar, the title '01 - ISE Cert Demo - CLUS' is displayed in blue text. To the right of the title, there are two buttons: 'Workflow Job Template' and 'Homenet'. The main area of the interface is currently empty, suggesting a preview or a blank state.

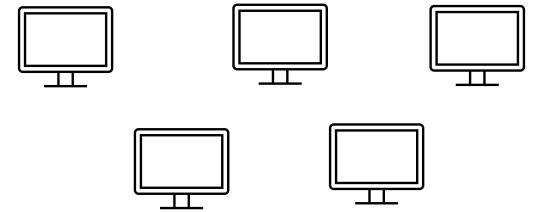
Execution Environments

Execution Environment



- Playbooks \ templates do not run on AWX host OS
- Playbooks run inside of specialized EE container
- AWX server stays clean
- EE gets python libraries, modules etc.

Execution Environment



- Linux container, spun up by AWX at playbook runtime
- Functional Execution Environment is built into AWX - awx-ee
- Build your own!
- Includes ansible-core, ansible-runner, python, python library & package, ansible collections, system dependencies
- Container provides consistency, and scalability, assuring playbooks run
- Stored Container Registries, Docker Hub, Quay IO, private repos

https://docs.ansible.com/ansible/latest/getting_started_ee/index.html

Execution Environment

The screenshot shows the AWX web interface with the following details:

- Left Sidebar:** Contains links for Templates, Credentials, Projects, Inventories, Hosts, Access (with Organizations, Users, Teams), Administration (with Credential Types, Notifications, Management Jobs, Instance Groups, Instances, Applications), and Execution Environments.
- Header:** Shows the AWX logo, user 'admin', and navigation icons.
- Page Title:** Execution Environments
- Table Headers:** Name (sorted), Image, Organization, Actions
- Data Rows:**
 - AWX EE (24.6.1) - Image: quay.io/ansible/awx-ee:24.6.1 - Organization: Globally Available - Actions: Edit, Delete
 - AWX EE (latest) - Image: quay.io/ansible/awx-ee:latest - Organization: Globally Available - Actions: Edit, Delete
 - Control Plane Execution Environment - Image: quay.io/ansible/awx-ee:24.6.1 - Organization: Globally Available - Actions: Edit, Delete
- Pagination:** 1-3 of 3 items, 1 of 1 page

https://quay.io/repository/ansible/awx-ee?tab=tags&tag=latest

Import bookmarks... Digital Commons N... Homenet Prod Homenet Lab ProtoV

RED HAT® Quay.io EXPLORE TUTORIAL PRICING search SIGN IN

Quay.io Support Team is transitioning to Red Hat Customer Portal. For that reason, support(at)quay.io e-mail address will be disabled after January 1st. Please check out the following article for more information: <https://access.redhat.com/articles/7099134>.

← Repositories ↑ Organization

ansible / awx-ee

Repository Tags		Compact	Expanded	Show Signatures			
<input type="checkbox"/>	latest	1 - 25 of 47	Filter Tags...	Download			
<input type="checkbox"/>	24.6.1	6 months ago	See Child Manifests	N/A	Never	SHA256 966752fe0e86	Download
<input type="checkbox"/>	24.6.0	6 months ago	See Child Manifests	N/A	Never	SHA256 854a454e5d1c	Download
<input type="checkbox"/>	24.5.0	7 months ago	See Child Manifests	N/A	Never	SHA256 12ad436d2e5b	Download

Updating the Execution Environment

- Update one or more of the following files
 - execution-environment.yaml ← **1** Required - base OS, build template
 - bindep.txt ← **2** Optional - system-level dependencies
 - requirements.txt ← **3** Optional - python packages
 - requirements.yaml ← **4** Optional - Ansible Galaxy
- Use ansible-builder to build a new Image
- Verify Image contents using podman
- Upload Image to Container Registry
- AWX downloads the latest image next playbook run

<https://developers.redhat.com/articles/2023/05/08/how-create-execution-environments-using-ansible-builder>

execution-environment.yaml

- Execution Environment Schema Definition

```
① execution-environment.yaml 1 ×  
Users > tiglen > Library > CloudStorage > OneDrive-Cisco > git > timmay-prod > ① execution-environment.yaml  
1   version: 3  
2  
3   images:  
4     base_image:  
5       name: quay.io/ansible/awx-ee:latest ← Base Image  
6  
7   dependencies:  
8     system: bindep.txt  
9     python: requirements.txt ← System - OS Level  
10    Python Package Req's  
11  
12   additional_build_steps:  
13     prepend_base:  
14       - RUN yum -y update && yum install -y ftp ← Additional Steps  
15     append_base:  
16       - ENV timmay_prod_version=v1.6  
17       - RUN ln -sf /usr/share/zoneinfo/America/New_York /etc/localtime  
18       - RUN alternatives --install /usr/bin/python3 python3 /usr/bin/python3.11 0  
19       - RUN curl -o /runner/ansible.cfg https://raw.githubusercontent.com/timmayg/timmay-prod/main/ansible.cfg  
20       - RUN curl -o /runner/release-notes.json https://raw.githubusercontent.com/timmayg/timmay-prod/main/release-notes.json  
21       - ENV ANSIBLE_CONFIG=/runner/ansible.cfg  
22       - RUN curl https://packages.microsoft.com/config/rhel/7/prod.repo > /etc/yum.repos.d/microsoft.repo  
23       - RUN yum install -y powershell  
24       - RUN pip install ansible-builder
```

The diagram illustrates the structure of the execution-environment.yaml file. It highlights three main sections with arrows pointing to specific lines:

- Base Image**: Points to the line `name: quay.io/ansible/awx-ee:latest` under the `base_image` key.
- System - OS Level Python Package Req's**: Points to the lines `system: bindep.txt` and `python: requirements.txt` under the `dependencies` key.
- Additional Steps**: Points to the lines under the `additional_build_steps` key, specifically `prepend_base` and `append_base`.

Execution Environment Base Image Options

awx-ee - the default

`quay.io/ansible/awx-ee`

ee-minimal-rhel8

`registry.redhat.io/ansible-automation-platform/ee-minimal-rhel8`

CentOS stream

`quay.io/centos/centos:stream9`

- Others too!
- Why ???
- Rebuilding using awx-ee takes > 20 minutes
- Rebuilding & Launching can be faster with lighter

bindep.txt

- OS Level Requirements

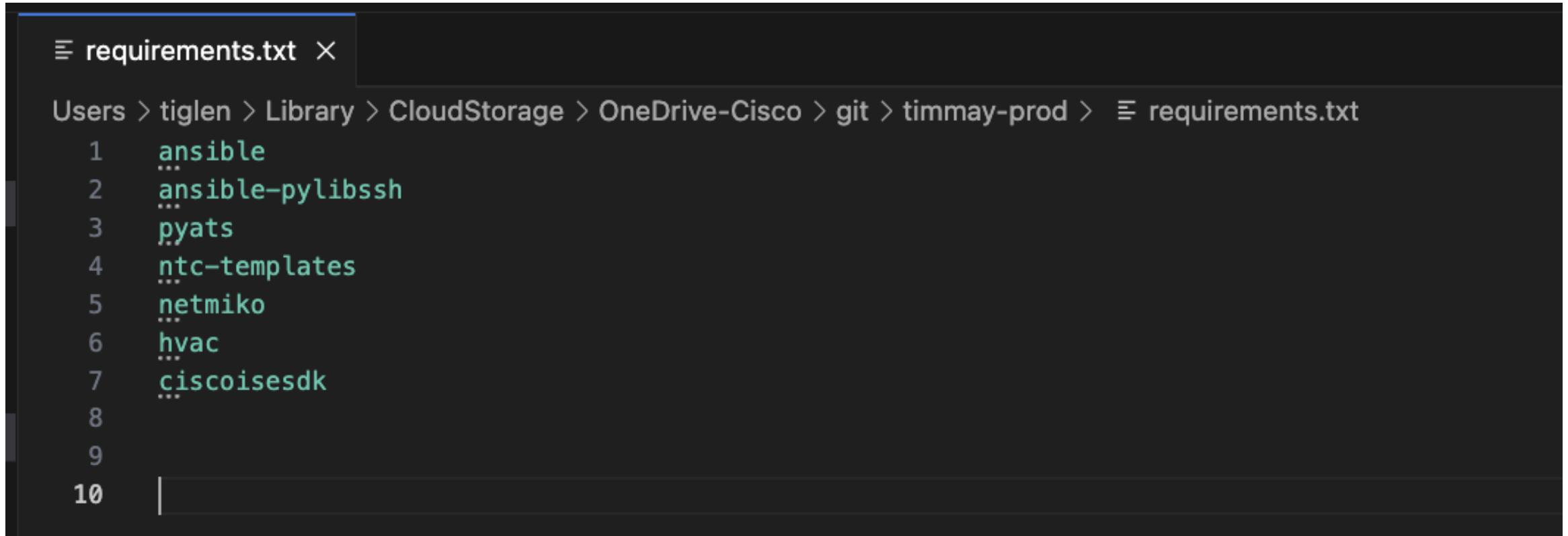
```
≡ bindep.txt ×

Users > tiglen > Library > CloudStorage > OneDrive-Cisco > git > timmay-prod > ≡ bindep.txt

1 git [platform:rpm]
2 iutils [platform:rpm]
3 nano [platform:rpm]
4 podman [platform:rpm]
5
```

requirements.txt

- Python Package requirements



The screenshot shows a terminal window with the title bar "requirements.txt X". The path displayed is "Users > tiglen > Library > CloudStorage > OneDrive-Cisco > git > timmay-prod > requirements.txt". The file content is listed below:

```
1 ansible
2 ansible-pylibssh
3 pyats
4 ntc-templates
5 netmiko
6 hvac
7 ciscoisesdk
8
9
10 |
```

requirements.yaml

- Ansible Galaxy Collections



The screenshot shows a terminal window with the title bar "④ requirements.yaml 2 X". The path in the title bar is "Users > tiglen > Library > CloudStorage > OneDrive-Cisco > git > timmay-prod > ④ requirements.yaml". The main content of the terminal is the YAML file "requirements.yaml" with the following code:

```
1 ---  
2 collections:  
3   - name: ansible.netcommon  
4   - name: ansible.utils  
5   - name: community.crypto  
6   - name: community.general  
7   - name: cisco.ios  
8   - name: cisco.ise  
9   - name: cisco.fmcansible.fmc_configuration  
10 ~~~
```

Collections in EE vs Collections in Projects

- Collections in EE are present for any \ all playbooks that are run

- This is speedy

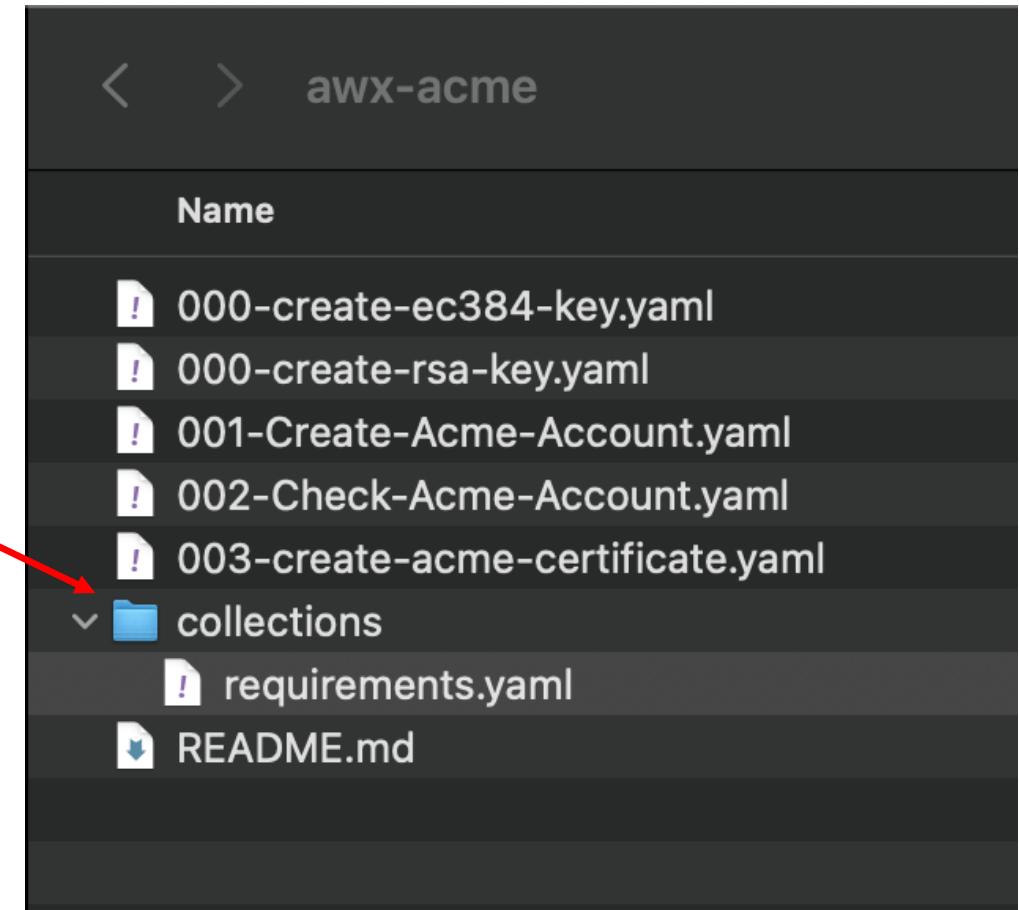
or

- Collections are specified in collections/requirements.yaml

- This is specified in the Project

- Every time you Sync an AWX Project the Galaxy Collections need to be downloaded

- This slows syncing down



Ansible Builder

The `build` command

The `ansible-builder build` command:

- takes an [execution environment definition file](#) as an input,
- outputs a build instruction file (Containerfile for Podman, Dockerfile for Docker),
- creates a build context necessary for building an execution environment image,
- builds the image.

By default, it looks for a file named `execution-environment.yml` (or `execution-environment.yaml`) in the current directory.

<https://ansible.readthedocs.io/projects/builder/en/latest/>

Upload your EE to a Container Registry

RED HAT® Quay.io EXPLORE REPOSITORIES TUTORIAL Current UI New UI search + ⚡ timmayg

Repositories Account timmayg / timmay-prod

Repository Tags

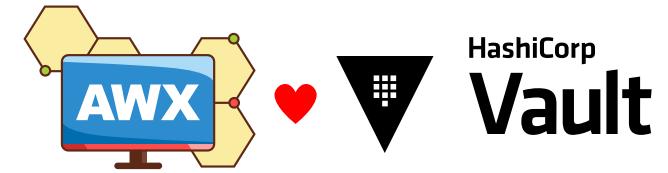
Compact Expanded Show Signatures

1 - 5 of 5 Filter Tags...

TAG	LAST MODIFIED	SECURITY SCAN	SIZE	EXPIRES	MANIFEST
v1.6	18 days ago	1 Critical · 5 fixable	1.2 GiB	Never	SHA256 e2520efd441c
latest	19 days ago	1 Critical · 11 fixable	1.2 GiB	Never	SHA256 e2520efd441c
v1.51	21 days ago	1 Critical · 9 fixable	1.2 GiB	Never	SHA256 b6711cb890e5
v1.2	a month ago	1 Critical · 15 fixable	989.4 MiB	Never	SHA256 b386e4886175
v1.4	a month ago	1 Critical · 5 fixable	1.1 GiB	Never	SHA256 4245ae9c2965

AWX Custom Credentials & External Secret Management & Hashi Corp Vault

Secrets Managers



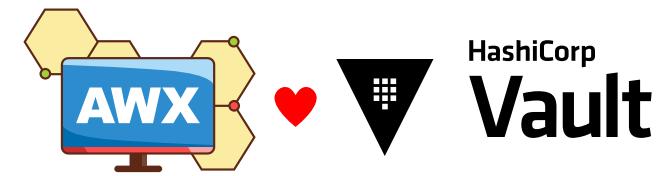
- AWX = Excellent Task Engine
- HashiCorp Vault = Excellent Secrets Management
- AWS Secrets Manager = Excellent Secrets Management
- Azure Key Vault = Excellent Secrets Management

Challenges with AWX built-in Credentials

- Machine Cred Challenges
- Familiarity but limited options

Lessons Learned

Limitation of Built-in Credentials



```
1 ---  
2 - name: Get a List of the ISE Nodes v1.0  
3   hosts: localhost  
4   gather_facts: false  
5  
6   tasks:  
7  
8     - name: 01 - Read a ISE Credentials from Vault  
9       community.hashi_vault.vault_kv2_get:  
10      path: "ise1_credentials"  
11      url: "https://vault.theglens.net:8200"  
12      engine_mount_point: "kv"  
13      auth_method: token  
14      token: "{{ ansible_password }}"  
15      register: ise_creds  
16  
17  
18     - name: 02 - Get a List of the ISE Nodes  
19       cisco.ise.node_info:  
20         ise_hostname: "{{ ise_creds.data.data.ise_hostname }}"  
21         ise_username: "{{ ise_creds.data.data.ise_username }}"  
22         ise_password: "{{ ise_creds.data.data.ise_password }}"  
23         ise_verify: true  
24         ise_debug: false  
25         register: ise_node_list  
26         timeout: 120
```

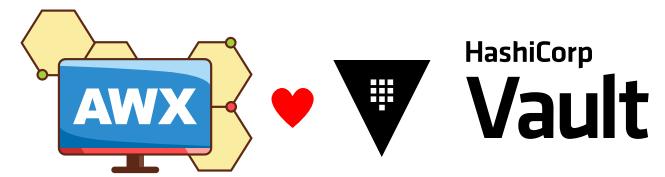
We need to run ansible playbooks to perform ISE Tasks.

Retrieve ISE User, Pass & hostname from Vault

Each ISE task requires authentication

Lessons Learned

Limitation of Built-in Credentials



```
1 ---  
2 - name: Get a List of the ISE Nodes v1.0  
3   hosts: localhost  
4   gather_facts: false  
5  
6   tasks:  
7  
8     - name: 01 - Read a ISE Credentials from Vault  
9       community.hashi_vault.vault_kv2_get:  
10      path: "ise1_credentials"  
11      url: "https://vault.theglens.net:8200"  
12      engine_mount_point: "kv"  
13      auth_method: token  
14      token: "{{ ansible_password }}"  
15      register: ise_creds  
16  
17  
18     - name: 02 - Get a List of the ISE Nodes  
19       cisco.ise.node_info:  
20         ise_hostname: "{{ ise_creds.data.data.ise_hostname }}"  
21         ise_username: "{{ ise_creds.data.data.ise_username }}"  
22         ise_password: "{{ ise_creds.data.data.ise_password }}"  
23         ise_verify: true  
24         ise_debug: false  
25         register: ise_node_list  
26         timeout: 120
```

This task only runs to query \ obtain a cred from Vault.

Not very efficient.

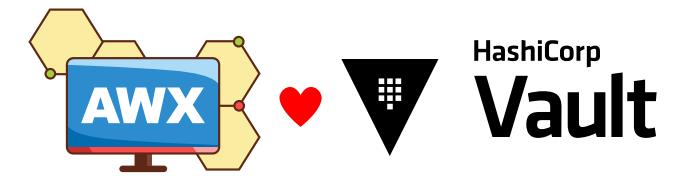
What if we need multiple creds?

Copy \ paste this code into **how many** playbooks ?

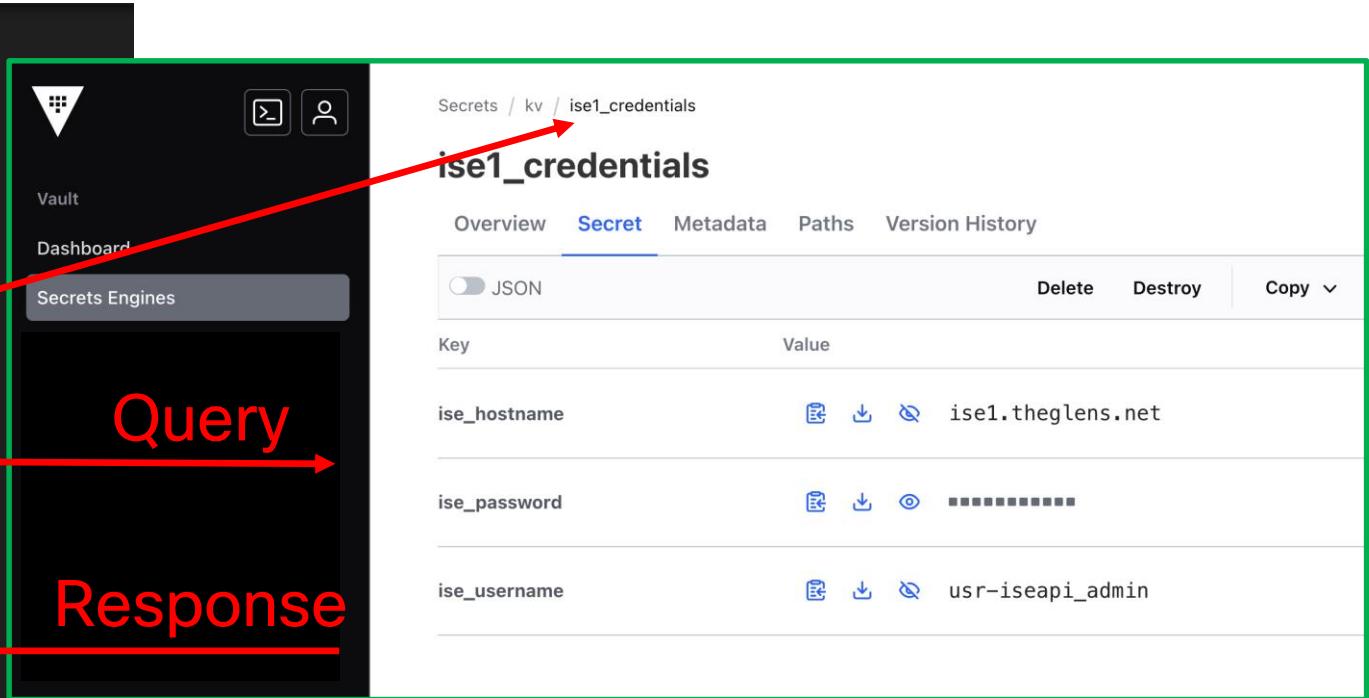
How readable is this for the next person?

Lessons Learned

Limitation of Built-in Credentials

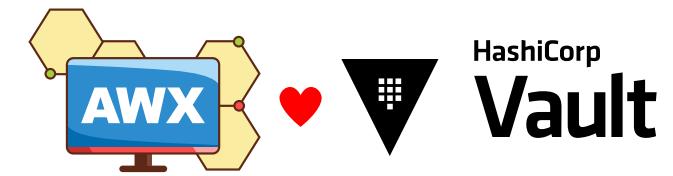


```
1 ---  
2 - name: Get a List of the ISE Nodes v1.0  
3   hosts: localhost  
4   gather_facts: false  
5  
6   tasks:  
7     - name: 01 - Read a ISE Credentials from Vault  
8       community.hashi_vault.vault_kv2_get:  
9         path: "ise1_credentials"  
10        url: "https://vault.theglens.net:8200"  
11        engine_mount_point: "kv"  
12        auth_method: token  
13        token: "{{ ansible_password }}"  
14        register: ise_creds  
15  
16     - name: 02 - Get a List of the ISE Nodes  
17       cisco.ise.node_info:  
18         ise_hostname: "{{ ise_creds.data.data.ise_hostname }}"  
19         ise_username: "{{ ise_creds.data.data.ise_username }}"  
20         ise_password: "{{ ise_creds.data.data.ise_password }}"  
21         ise_verify: true  
22         ise_debug: false  
23         register: ise_node_list  
24  
25         timeout: 120
```



Lessons Learned

Limitation of Built-in Credentials



```
1 ---  
2 - name: Get a List of the ISE Nodes v1.0  
3   hosts: localhost  
4   gather_facts: false  
5  
6   tasks:  
7     - name: 01 - Read a ISE Credentials from Vault  
8       community.hashi_vault.vault_kv2_get:  
9         path: "ise1_credentials"  
10        url: "https://vault.theglens.net:8200"  
11        engine_mount_point: "kv"  
12        auth_method: token  
13        token: "{{ ansible_password }}" ——————→  
14        register: ise_creds  
15  
16  
17     - name: 02 - Get a List of the ISE Nodes  
18       cisco.ise.node_info:  
19         ise_hostname: "{{ ise_creds.data.data.ise_hostname }}"  
20         ise_username: "{{ ise_creds.data.data.ise_username }}"  
21         ise_password: "{{ ise_creds.data.data.ise_password }}"  
22         ise_verify: true  
23         ise_debug: false  
24         register: ise_node_list  
25  
26         timeout: 120
```

Limitation AWX Machine Credential ansible_password

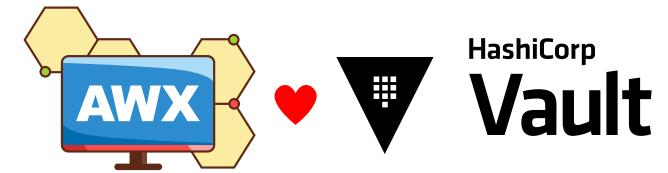
- Reserved variable name
- Limited usage
- Not designed to be used this way
- Doesn't feel good, it's a hack

Why did I do it this way?



- Familiar with ‘machine credential’ / SSH cred in Ansible
 - ansible_username / ansible_password
- Knew that AWX would encrypt ansible_password
- Comfortable storing api_key, tokens, in ansible_password
- Knew it was a hack but it worked, till it didn’t

Welcome, Custom Credential Type



Credential Types > aa - Cisco ISE Cred Type

Details

◀ Back to credential types Details

Name aa - Cisco ISE Cred Type

Input configuration ⓘ YAML JSON

```
1 fields:  
2   - id: ise_hostname  
3     type: string  
4     label: ISE hostname  
5   - id: ise_username  
6     type: string
```

Injector configuration ⓘ YAML JSON

```
1 extra_vars:  
2   ise_hostname: '{{ise_hostname}}'  
3   ise_password: '{{ise_password}}'  
4   ise_username: '{{ise_username}}'
```

Input configuration

fields:

- **id:** ise_hostname
type: string
label: ISE hostname
- **id:** ise_username
type: string
label: Username
- **id:** ise_password
type: string
label: Password
secret: true

required:

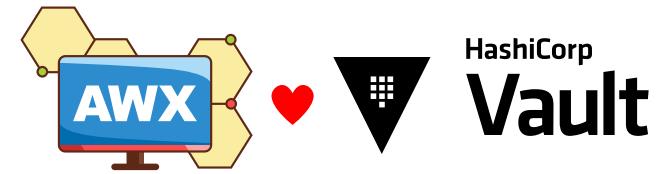
- ise_hostname
- ise_username
- ise_password

Injector configuration

extra_vars:

```
ise_hostname: '{{ise_hostname}}'  
ise_password: '{{ise_password}}'  
ise_username: '{{ise_username}}'
```

Welcome, Custom Credential Type



Credential Types > aa - Cisco ISE Cred Type

Details

◀ Back to credential types Details

Name aa - Cisco ISE Cred Type

Input configuration ⓘ YAML JSON

```
1 fields:  
2   - id: ise_hostname  
3     type: string  
4     label: ISE hostname  
5   - id: ise_username  
6     type: string
```

Injector configuration ⓘ YAML JSON

```
1 extra_vars:  
2   ise_hostname: '{{ise_hostname}}'  
3   ise_password: '{{ise_password}}'  
4   ise_username: '{{ise_username}}'
```

Input configuration

fields:

- **id: ise_hostname**
type: string
label: ISE hostname
- **id: ise_username**
type: string
label: Username
- **id: ise_password**
type: string
label: Password
secret: true

required:

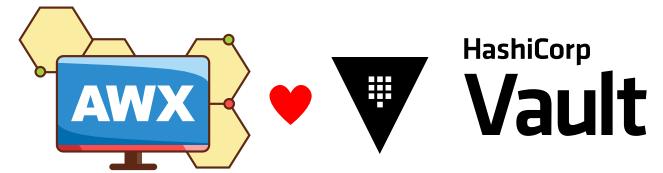
- ise_hostname
- ise_username
- ise_password

Injector configuration

extra_vars:

```
ise_hostname: '{{ise_hostname}}'  
ise_password: '{{ise_password}}'  
ise_username: '{{ise_username}}'
```

Welcome, Custom Credential Type



Credential Types > aa - Cisco ISE Cred Type

Details

[Back to credential types](#) Details

Name aa - Cisco ISE Cred Type

Input configuration ⓘ [YAML](#) [JSON](#)

```
1 fields:  
2   - id: ise_hostname  
3     type: string  
4     label: ISE hostname  
5   - id: ise_username  
6     type: string
```

Injector configuration ⓘ [YAML](#) [JSON](#)

```
1 extra_vars:  
2   ise_hostname: '{{ise_hostname}}'  
3   ise_password: '{{ise_password}}'  
4   ise_username: '{{ise_username}}'
```

Input configuration

fields:

- **id:** ise_hostname
type: string
label: ISE hostname
- **id:** ise_username
type: string
label: Username
- **id:** ise_password
type: string
label: Password
secret: true

required:

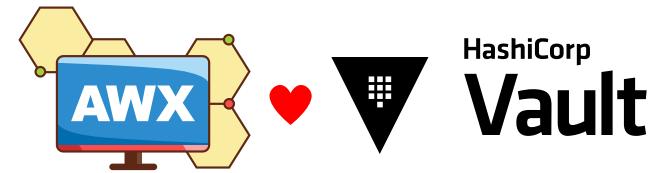
- ise_hostname
- ise_username
- ise_password

Injector configuration

extra_vars:

```
ise_hostname: '{{ise_hostname}}'  
ise_password: '{{ise_password}}'  
ise_username: '{{ise_username}}'
```

Welcome, Custom Credential Type



Credential Types > aa - Cisco ISE Cred Type

Details

◀ Back to credential types Details

Name aa - Cisco ISE Cred Type

Input configuration ⓘ YAML JSON

```
1 fields:  
2   - id: ise_hostname  
3     type: string  
4     label: ISE hostname  
5   - id: ise_username  
6     type: string
```

Injector configuration ⓘ YAML JSON

```
1 extra_vars:  
2   ise_hostname: '{{ise_hostname}}'  
3   ise_password: '{{ise_password}}'  
4   ise_username: '{{ise_username}}'
```

Input configuration

fields:

- **id:** ise_hostname
type: string
label: ISE hostname
- **id:** ise_username
type: string
label: Username
- **id:** ise_password
type: string
label: Password
secret: true

required:

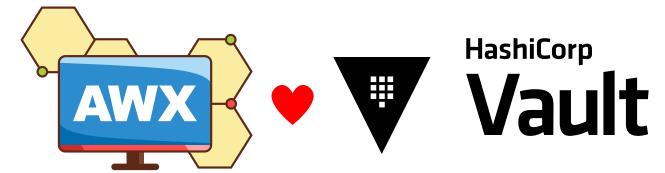
- ise_hostname
- ise_username
- ise_password

Injector configuration

extra_vars:

```
ise_hostname: '{{ise_hostname}}'  
ise_password: '{{ise_password}}'  
ise_username: '{{ise_username}}'
```

Welcome, Custom Credential Type



Credential Types > aa - Cisco ISE Cred Type

Details

◀ Back to credential types Details

Name aa - Cisco ISE Cred Type

Input configuration ⓘ YAML JSON

```
1 fields:  
2   - id: ise_hostname  
3     type: string  
4     label: ISE hostname  
5   - id: ise_username  
6     type: string
```

Injector configuration ⓘ YAML JSON

```
1 extra_vars:  
2   ise_hostname: '{{ise_hostname}}'  
3   ise_password: '{{ise_password}}'  
4   ise_username: '{{ise_username}}'
```

Input configuration

fields:

- **id:** ise_hostname
type: string
label: ISE hostname
- **id:** ise_username
type: string
label: Username
- **id:** ise_password
type: string
label: Password
secret: true

required:

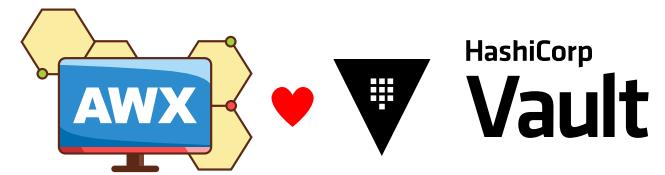
- ise_hostname
- ise_username
- ise_password

Injector configuration

extra_vars:

```
ise_hostname: '{{ise_hostname}}'  
ise_password: '{{ise_password}}'  
ise_username: '{{ise_username}}'
```

Welcome, Custom Credential Type



Credential Types > aa - Cisco ISE Cred Type

Details

◀ Back to credential types Details

Name aa - Cisco ISE Cred Type

Input configuration ⓘ YAML JSON

```
1 fields:  
2   - id: ise_hostname  
3     type: string  
4     label: ISE hostname  
5   - id: ise_username  
6     type: string
```

Injector configuration ⓘ YAML JSON

```
1 extra_vars:  
2   ise_hostname: '{{ise_hostname}}'  
3   ise_password: '{{ise_password}}'  
4   ise_username: '{{ise_username}}'
```

Input configuration

fields:

- **id**: ise_hostname
type: string
label: ISE hostname
- **id**: ise_username
type: string
label: Username
- **id**: ise_password
type: string
label: Password
secret: true

required:

- ise_hostname
- ise_username
- ise_password

Injector configuration

extra_vars:

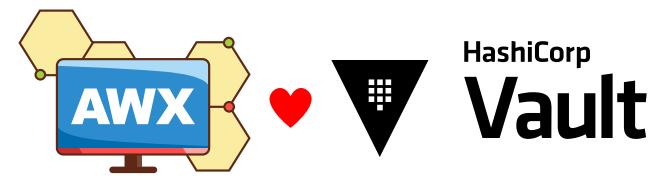
```
ise_hostname: '{{ise_hostname}}'  
ise_password: '{{ise_password}}'  
ise_username: '{{ise_username}}'
```

id

vars in
playbook

https://docs.ansible.com/automation-controller/4.4/html/userguide/credential_types.html

Template to Playbook Cred Mapping



Screenshot of the AWX UI showing a template details page.

Left Sidebar: Views (Dashboard, Jobs, Schedules, Activity Stream, Workflow Approvals), Resources (Templates, Credentials, Projects, Inventories, Hosts), Access (Organizations, Users).

Details Page:

- Template Path:** Templates > 1b - Get ISE Nodes v2 - CLEMEA - DEVNET-2517 - PROD@LIVE
- Name:** 1b - Get ISE Nodes v2 - CLEMEA - DEVNET-2517 - PROD@LIVE
- Inventory:** Local Host Inventory
- Playbook:** Get-ISE-Nodes-v2.yaml
- Timeout:** 0
- Created:** 2/2/2025, 1:14:14 PM by timmay
- Credentials:** Cloud: aa - ise.thegle... (highlighted with a red box)
- Labels:** devnet-2517
- Variables:** YAML (selected), JSON

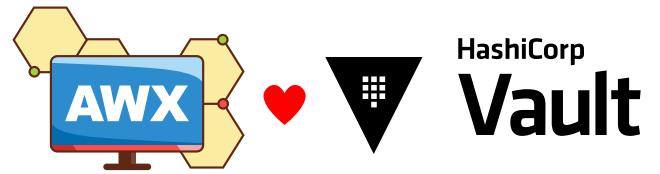
Playbook YAML Content (Lines 1-17):

```
1 ---  
2 - name: Get a List of the ISE Nodes v2.0  
3 hosts: localhost  
4 gather_facts: false  
5  
6 tasks:  
7  
8   - name: 01 - Get a List of the ISE Nodes  
9     cisco.ise.node_info:  
10       ise_hostname: "{{ ise_hostname }}"  
11       ise_username: "{{ ise_username }}"  
12       ise_password: "{{ ise_password }}"  
13       ise_verify: true  
14       ise_debug: false  
15     register: ise_node_list  
16     timeout: 15  
17
```

Annotations:

- A red box highlights the 'ise_password' line in the YAML code.
- A red arrow points from the 'Credentials' field in the sidebar to the highlighted line in the code.
- A red box highlights the 'ise_password' line in the code.
- A red exclamation mark (!!) is placed next to the highlighted line in the code.
- A large red annotation 'Injector Config' is placed on the right side of the code area.

Old vs New Playbook



```
1 ---  
2 - name: Get a List of the ISE Nodes v1.0  
3   hosts: localhost  
4   gather_facts: false  
5  
6   tasks:  
7  
8     - name: 01 - Read a ISE Credentials from Vault  
9       community.hashi_vault.vault_kv2_get:  
10      path: "ise1_credentials"  
11      url: "https://vault.theglens.net:8200"  
12      engine_mount_point: "kv"  
13      auth_method: token  
14      token: "{{ ansible_password }}"  
15      register: ise_creds  
16  
17     - name: 02 - Get a List of the ISE Nodes  
18       cisco.ise.node_info:  
19         ise_hostname: "{{ ise_creds.data.data.ise_hostname }}"  
20         ise_username: "{{ ise_creds.data.data.ise_username }}"  
21         ise_password: "{{ ise_creds.data.data.ise_password }}"  
22         ise_verify: true  
23         ise_debug: false  
24  
25       register: ise_node_list  
26       timeout: 120
```

Which credential should we use?

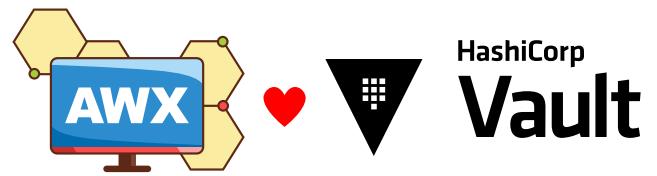
```
1 ---  
2 - name: Get a List of the ISE Nodes v2.0  
3   hosts: localhost  
4   gather_facts: false  
5  
6   tasks:  
7  
8     - name: 01 - Get a List of the ISE Nodes  
9       cisco.ise.node_info:  
10      ise_hostname: "{{ ise_hostname }}"  
11      ise_username: "{{ ise_username }}"  
12      ise_password: "{{ ise_password }}"  
13      ise_verify: true  
14      ise_debug: false  
15      register: ise_node_list  
16      timeout: 15  
17
```

Easy &
Reusable

DEMO – Build a Simple ISE Cred

- Show Existing Custom Credential Type – aa – Cisco ISE Cred Type
- Create new Credential
- Type aa- ISE
- Name

Custom Cred to Vault Mapping



The screenshot shows the AWX interface on the left and the HashiCorp Vault interface on the right.

AWX Interface (Left):

- Sidebar:** Views (Dashboard, Jobs, Schedules, Activity Stream, Workflow Approvals), Resources (Templates, **Credentials** highlighted with a red box), Projects, Inventories, Hosts, Access (Organizations, Users, Teams), Administration (Credential Types, Notifications, Management Jobs, Instance Groups, Instances, Applications).
- Credentials Page:** Shows a credential named "aa - ise.theglens.net Cred". The "Details" tab is selected. The "ISE hostname" field contains "Hashivault Kv: vault...". Below it is a JSON configuration block:

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_hostname",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",  
6 }  
  
The "Username" field contains "Hashivault Kv: vault...". Below it is another JSON configuration block:
```

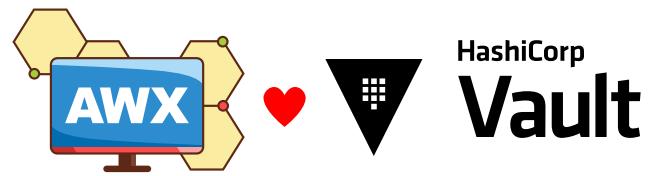
```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_username",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",  
6 }  
  
The "Password" field contains "Hashivault Kv: vault...". Below it is a third JSON configuration block:
```

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_password",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",  
6 }
```

Vault Interface (Right):

- Dashboard:** Shows "Secrets / kv / ise1_credentials".
- Secrets List:** Title "ise1_credentials". Subtitle "Secret".
 - Key: ise_hostname Value: ise1.theglens.net
 - Key: ise_password Value: (redacted)
 - Key: ise_username Value: usr-iseapi_admin

Custom Cred to Vault Mapping



The screenshot shows the AWX UI interface for managing credentials. On the left, there's a sidebar with a "Credentials" tab selected. Three red boxes with white text are overlaid on the sidebar:

- JSON expression for this External Secret** (top)
- JSON expression for this External Secret** (middle)
- JSON expression for this External Secret** (bottom)

Red arrows point from these boxes to the corresponding "Secret" fields in the "Details" section of the main content area. The "Details" section shows a credential named "aa - ise.theglens.net Cred". It has three fields: "ISE hostname", "Username", and "Password", each with a "Hashivault Kv: vault..." placeholder. Below each field is a JSON expression:

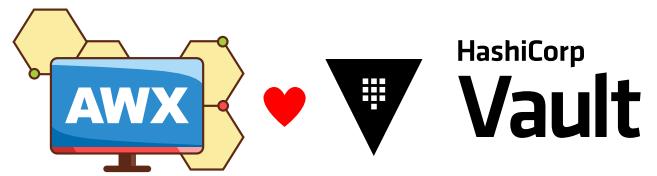
```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_hostname",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",  
}  
  
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_username",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",  
}  
  
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_password",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",  
}
```

On the right side of the screenshot, there is a separate window showing the HashiCorp Vault interface. It displays a "Secrets / kv / ise1_credentials" page. The "Secret" tab is selected, showing three key-value pairs:

Key	Value
ise_hostname	ise1.theglens.net
ise_password	*****
ise_username	usr-iseapi_admin

Below the table, there are "Delete", "Destroy", and "Copy" buttons.

Custom Cred to Vault Mapping



The screenshot shows the AWX UI on the left and the HashiVault KV interface on the right. The AWX UI displays a credential named "aa - ise.theglens.net Cred" with three fields: "ISE hostname", "Username", and "Password". Each field has a "Hashivault Kv: vault..." placeholder. The "ISE hostname" field is highlighted with a red box. The HashiVault KV interface shows a list of secrets under the path "ise1_credentials":

Key	Value
ise_hostname	ise1.theglens.net
ise_password	*****
ise_username	usr-iseapi_admin

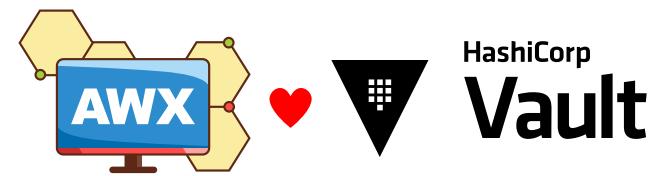
Both the AWX UI and the KV interface show JSON snippets of the secret values:

```
ISE hostname * Hashivault Kv: vault...
1 {
2   "auth_path": "",
3   "secret_key": "ise_hostname",
4   "secret_path": "ise1_credentials",
5   "secret_backend": "kv",
```

```
Username * Hashivault Kv: vault...
1 {
2   "auth_path": "",
3   "secret_key": "ise_username",
4   "secret_path": "ise1_credentials",
5   "secret_backend": "kv",
```

```
Password * Hashivault Kv: vault...
1 {
2   "auth_path": "",
3   "secret_key": "ise_password",
4   "secret_path": "ise1_credentials",
5   "secret_backend": "kv",
```

Custom Cred to Vault Mapping



The screenshot illustrates the configuration of a custom credential in AWX (Ansible Web Interface) that maps to a secret in HashiCorp Vault.

AWX Credentials Details:

- Name:** aa - ise.theglens.net Cred
- ISE hostname ***: Hashivault Kv: vault...
- JSON Configuration (for ISE hostname):**

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_hostname",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",  
6 }
```

A red arrow points from this configuration to the "Secrets Engines" section in the Vault sidebar.
- Username ***: Hashivault Kv: vault...
- JSON Configuration (for Username):**

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_username",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",  
6 }
```
- Password ***: Hashivault Kv: vault...
- JSON Configuration (for Password):**

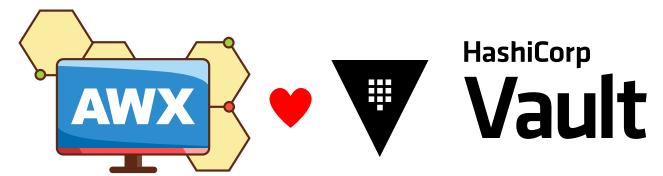
```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_password",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",  
6 }
```

HashiCorp Vault Secrets View:

- Path:** Secrets / kv / ise1_credentials
- ise1_credentials Secret Data:**

Key	Value
ise_hostname	ise1.theglens.net
ise_password	*****
ise_username	usr-iseapi_admin

Custom Cred to Vault Mapping



The screenshot illustrates the configuration of a custom credential in AWX (left) and its corresponding secret in HashiCorp Vault (right), with a red arrow indicating the mapping between them.

AWX Credentials Details:

- Name:** aa - ise.theglens.net Cred
- ISE hostname ***: Hashivault Kv: vault...
- JSON Configuration (for ISE hostname):**

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_hostname",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",  
6 }
```
- Username ***: Hashivault Kv: vault...
- JSON Configuration (for Username):**

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_username",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",  
6 }
```
- Password ***: Hashivault Kv: vault...
- JSON Configuration (for Password):**

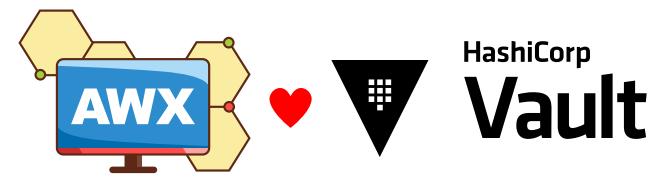
```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_password",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",  
6 }
```

HashiCorp Vault Secret:

- Path:** Secrets / kv / ise1_credentials
- Secret Tab:** Overview, Secret (selected), Metadata, Paths, Version History
- Key Value Pairs:**

Key	Value
ise_hostname	ise1.theglens.net
ise_password	*****
ise_username	usr-iseapi_admin

Custom Cred to Vault Mapping



The screenshot illustrates the process of mapping custom credentials in AWX to a HashiCorp Vault secret. A red arrow points from the "ISE hostname" field in the AWX "Details" tab to the "ise_hostname" entry in the Vault "Secret" tab. Another red arrow points from the "Username" field in the AWX "Details" tab to the "ise_username" entry in the Vault "Secret" tab. A third red arrow points from the "Password" field in the AWX "Details" tab to the "ise_password" entry in the Vault "Secret" tab.

AWX Details Tab (Left):

- Name: aa - ise.theglens.net Cred
- Organization:
- ISE hostname *: Hashivault Kv: vault...
- JSON configuration:

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_hostname",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",
```
- Username *: Hashivault Kv: vault...
- JSON configuration:

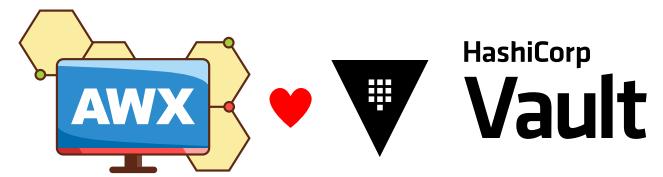
```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_username",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",
```
- Password *: Hashivault Kv: vault...
- JSON configuration:

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_password",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",
```

HashiCorp Vault Secret Tab (Right):

- Path: Secrets / kv / ise1_credentials
- Secret Tab selected.
- Key Value pairs:
 - ise_hostname: ise1.theglens.net
 - ise_password: [REDACTED]
 - ise_username: usr-iseapi_admin

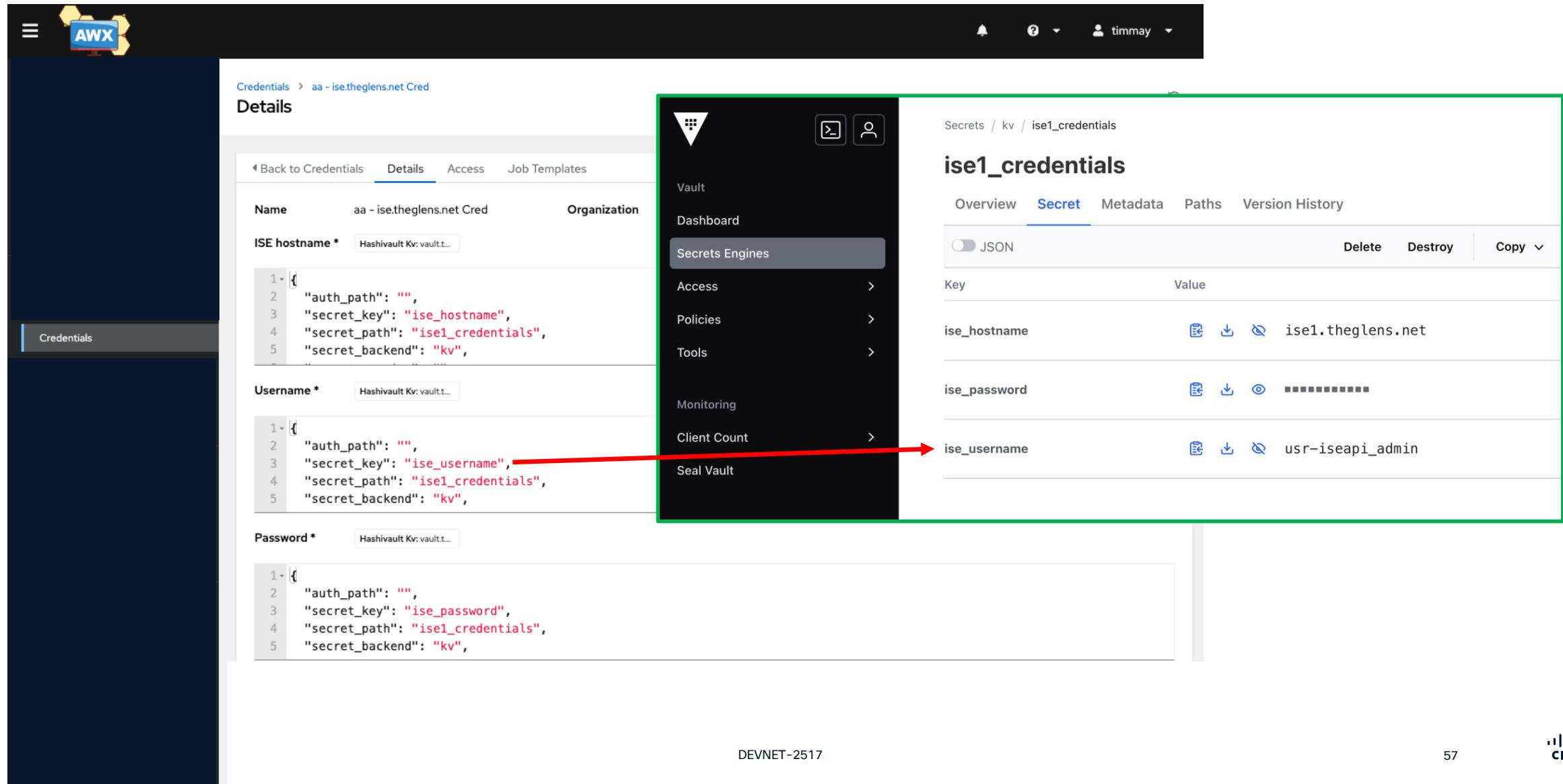
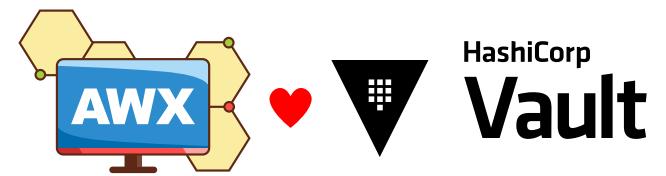
Custom Cred to Vault Mapping



The screenshot illustrates the process of mapping custom credentials in AWX to secrets in HashiCorp Vault. On the left, the AWX interface shows a credential named "aa - ise.theglens.net Cred" with three fields: "ISE hostname", "Username", and "Password". Each field has a JSON configuration dropdown. Red arrows point from the "ISE hostname" dropdown to the "Secrets Engines" section of the Vault dashboard, and from the "Username" and "Password" dropdowns to the "ise1_credentials" secret in Vault. The Vault interface on the right shows the "ise1_credentials" secret with three key-value pairs: "ise_hostname" (value: ise1.theglens.net), "ise_password" (value: masked), and "ise_username" (value: usr-iseapi_admin).

Key	Value
ise_hostname	ise1.theglens.net
ise_password	*****
ise_username	usr-iseapi_admin

Custom Cred to Vault Mapping



The screenshot illustrates the configuration of a custom credential in AWX (left) and its corresponding secret in HashiCorp Vault (right).

AWX Credentials Details:

- Name:** aa - ise.theglens.net Cred
- Organization:** (empty)
- ISE hostname ***: Hashivault Kv: vault...
- JSON Configuration (for ISE hostname):**

```
1: {  
2:   "auth_path": "",  
3:   "secret_key": "ise_hostname",  
4:   "secret_path": "ise1_credentials",  
5:   "secret_backend": "kv",  
6: }
```
- Username ***: Hashivault Kv: vault...
- JSON Configuration (for Username):**

```
1: {  
2:   "auth_path": "",  
3:   "secret_key": "ise_username",  
4:   "secret_path": "ise1_credentials",  
5:   "secret_backend": "kv",  
6: }
```

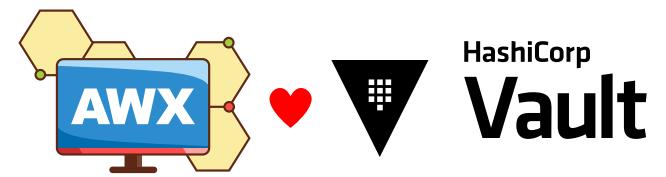
A red arrow points from the 'secret_key' value 'ise_username' in the JSON to the 'ise_username' key in the HashiCorp Vault Secret table below.
- Password ***: Hashivault Kv: vault...
- JSON Configuration (for Password):**

```
1: {  
2:   "auth_path": "",  
3:   "secret_key": "ise_password",  
4:   "secret_path": "ise1_credentials",  
5:   "secret_backend": "kv",  
6: }
```

HashiCorp Vault Secrets / kv / ise1_credentials:

Key	Value
ise_hostname	ise1.theglens.net
ise_password	*****
ise_username	usr-iseapi_admin

Custom Cred to Vault Mapping



The screenshot illustrates the process of mapping custom credentials in AWX to secrets in HashiCorp Vault.

AWX Credentials Details:

- Name:** aa - ise.theglens.net Cred
- ISE hostname ***: Hashivault Kv: vault...
- JSON Configuration (for ISE hostname):**

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_hostname",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",  
6 }
```
- Username ***: Hashivault Kv: vault...
- JSON Configuration (for Username):**

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_username",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",  
6 }
```
- Password ***: Hashivault Kv: vault...
- JSON Configuration (for Password):**

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_password",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",  
6 }
```

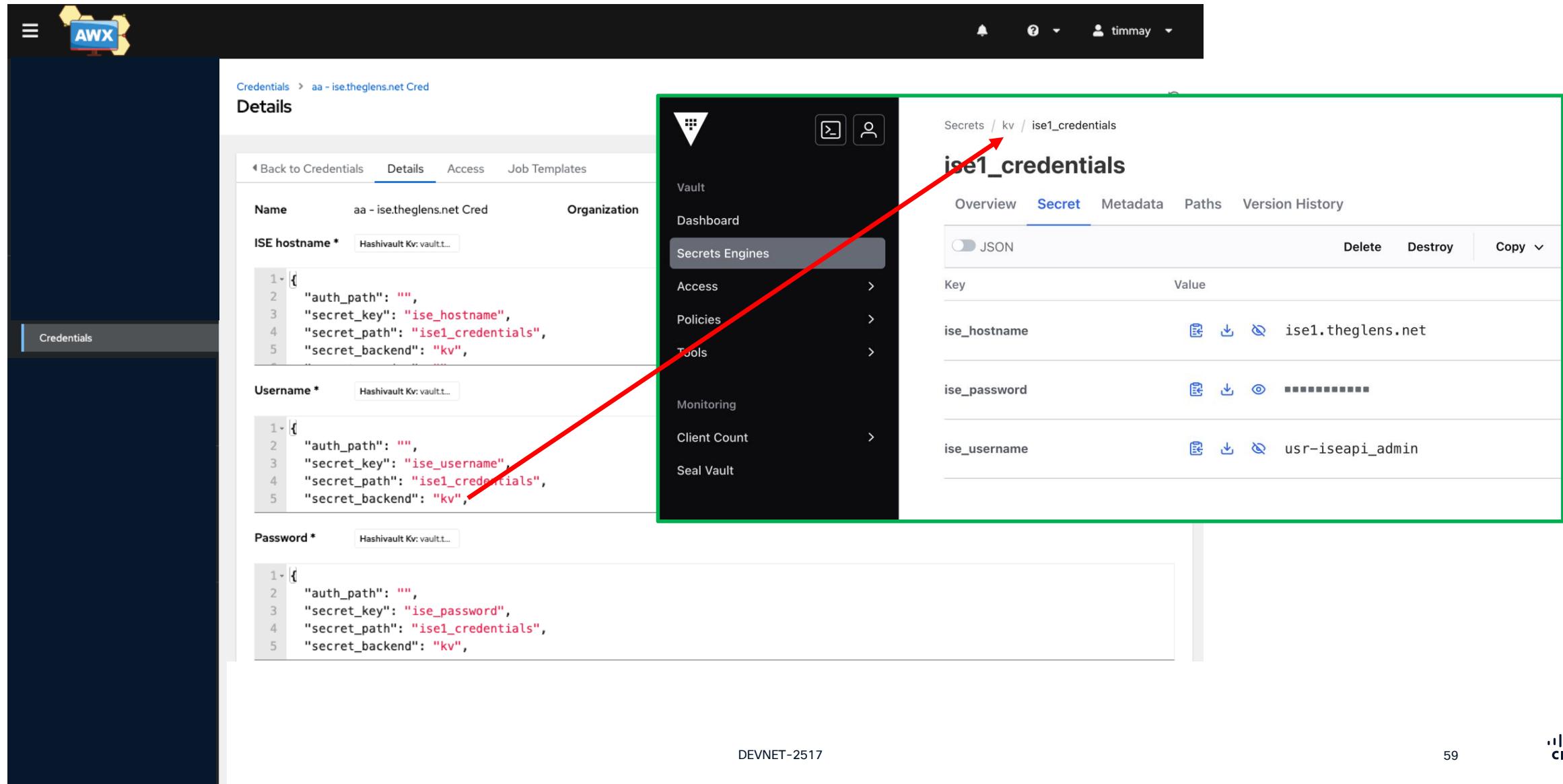
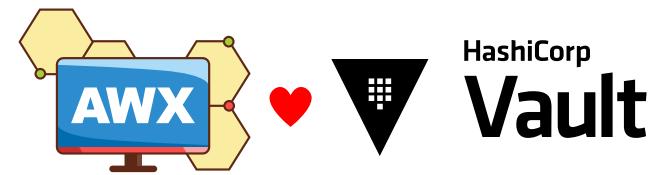
HashiCorp Vault Secrets View:

- Path:** Secrets / kv / ise1_credentials
- Secret Tab:** Overview, Secret (selected), Metadata, Paths, Version History
- Secret Data:**

Key	Value
ise_hostname	ise1.theglens.net
ise_password	*****
ise_username	usr-iseapi_admin

A red arrow points from the "secret_path" field in the AWX credential configuration to the "ise1_credentials" secret path in the Vault interface, indicating the mapping between the two.

Custom Cred to Vault Mapping



The screenshot illustrates the process of mapping custom credentials in AWX to a HashiCorp Vault secret engine. A red arrow points from the 'ISE hostname' field in the AWX 'Details' screen to the 'ise1_credentials' secret in the Vault interface. Another red arrow points from the 'Username' field in AWX to the same secret in Vault. Both fields reference the 'ise1_credentials' secret, which contains three key-value pairs: 'ise_hostname' (value: ise1.theglens.net), 'ise_password' (value: masked), and 'ise_username' (value: usr-iseapi_admin).

AWX Details Screen (Left):

- Name: aa - ise.theglens.net Cred
- Organization:
- ISE hostname *: Hashivault Kv: vault...
- JSON preview:

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_hostname",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",  
6 }
```
- Username *: Hashivault Kv: vault...
- JSON preview:

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_username",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",  
6 }
```
- Password *: Hashivault Kv: vault...
- JSON preview:

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_password",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",  
6 }
```

HashiCorp Vault Interface (Right):

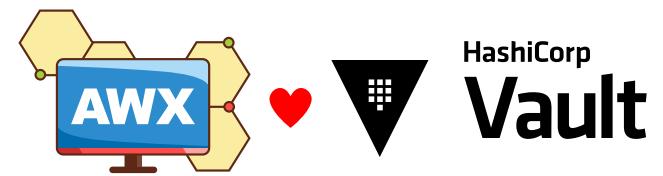
- Secrets / kv / ise1_credentials
- ise1_credentials
- Overview Secret Metadata Paths Version History
- JSON toggle
- Delete Destroy Copy
- Key Value
- ise_hostname ise1.theglens.net
- ise_password (masked)
- ise_username usr-iseapi_admin

DEVNET-2517

59

CISCO

Custom Cred to Vault Mapping



The screenshot shows the AWX UI for managing credentials. A specific credential named "aa - ise.theglens.net Cred" is selected. The "Details" tab is active. The credential type is "aa - Cisco ISE Cred Type". The "ISE hostname" field is set to "Hashivault Kv: vault.t...". Below this, there are three fields with JSON configuration:

- Username ***: Hashivault Kv: vault.t...
JSON: { "auth_path": "", "secret_key": "ise_hostname", "secret_path": "ise1_credentials", "secret_backend": "kv", ... }
- Password ***: Hashivault Kv: vault.t...
JSON: { "auth_path": "", "secret_key": "ise_password", "secret_path": "ise1_credentials", "secret_backend": "kv", ... }
- Secrets**: Hashivault Kv: vault.t...
JSON: { "auth_path": "", "secret_key": "ise_username", "secret_path": "ise1_credentials", "secret_backend": "kv", ... }

Red arrows point from the "Secrets" field's JSON path to the "secret_key" fields in the Username and Password fields. Red boxes highlight the "Secrets" field and its JSON path.

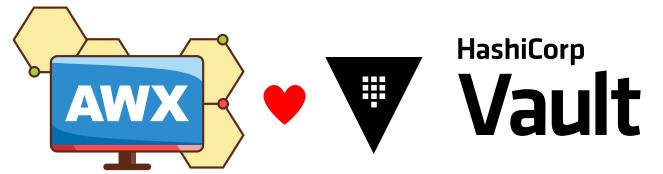
Vault Dashboard (Left):

- Secrets Engines: Access, Policies, Tools, Monitoring, Client Count, Seal Vault.

Vault Secret View (Right):

- Path: Secrets / kv / ise1_credentials
- Secret: ise1_credentials
- Overview, Secret, Metadata, Paths, Version History tabs.
- JSON toggle.
- Key Value pairs:
 - ise_hostname: ise1.theglens.net
 - ise_password: [REDACTED]
 - ise_username: usr-iseapi_admin
- Delete, Destroy, Copy buttons.

Playbook to Custom Cred Mapping



```
1 ---  
2 - name: Get a List of the ISE Nodes v2.0  
3   hosts: localhost  
4   gather_facts: false  
5  
6 tasks:  
7  
8   - name: 01 - Get a List of the ISE Nodes  
9     cisco.ise.node_info:  
10    ise_hostname: "{{ ise_hostname }}"  
11    ise_username: "{{ ise_username }}"  
12    ise_password: "{{ ise_password }}"  
13    ise_verify: true  
14    ise_debug: false  
15    register: ise_node_list  
16    timeout: 15  
17
```

Credentials > aa - ise.theglens.net Cred

Details

Back to Credentials Details Access Job Templates

Name aa - ise.theglens.net Cred Organization Homenet

ISE hostname * Hashivault Kv: vault.t...

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_hostname",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",
```

Username * Hashivault Kv: vault.t...

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_username",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",
```

Password * Hashivault Kv: vault.t...

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_password",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",
```

Created 1/25/2025, 2:10:33 PM by timmay Last Modified 1/30/2025, 7:05:19 PM by timmay

* This field will be retrieved from an external secret management system using the specified credential.

Edit Delete

**Sorry for making you focus on all
those arrows.**

**There will be a Cisco certification
test after this session, and you will
have to drag and drop from one
side to the other.**

I hope you were paying attention!!!

Custom Cred Reuse



Credentials > aa - ise.theglens.net Cred

Details

[Back to Credentials](#) **Details** Access Job Templates

Name aa - ise.theglens.net Cred Organization Homenet

ISE hostname * Hashivault Kv:vault...

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_hostname",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",
```

Username * Hashivault Kv:vault...

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_username",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",
```

Password * Hashivault Kv:vault...

```
1 {  
2   "auth_path": "",  
3   "secret_key": "ise_password",  
4   "secret_path": "ise1_credentials",  
5   "secret_backend": "kv",
```

Created 1/25/2025, 2:10:33 PM by timmay

Last Modified 1/30/2025, 7:05:19 PM by timmay

* This field will be retrieved from an external secret management system using the specified credential.

[Edit](#) [Delete](#)

Credentials > aa - ise.theglens.net Cred

Job Templates

[Back to Credentials](#) Details Access **Job Templates**

Name Add Delete

Name ↑

Type ↓

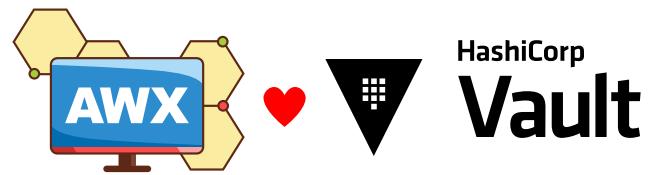
[0 - ALL of your ISE Templates Can Use that Cred !!!](#) Job Template

[1b - Get ISE Nodes v2 - PROD@LIVE](#) Job Template

[Get ISE Deployment](#) Job Template

[Get ISE Nodes](#) Job Template

Custom Cred Reuse



Credentials > aa - ise.theglens.net Cred

Job Templates

Back to Credentials Details Access Job Templates

Name Add Delete

Name
0 - ALL of your ISE Templates Can Use that Cred !!!
1b - Get ISE Nodes v2 - CLEMEA - DEVNET-2517 - PROD@LIVE
Get ISE Deployment
Get ISE Nodes

```
1 ---  
2 - name: Get a List of the ISE Nodes v2.0  
3   hosts: localhost  
4   gather_facts: false  
5  
6 tasks:  
7  
8   - name: 01 - Get a List of the ISE Nodes  
9     cisco.ise.node_info:  
10    ise_hostname: "{{ ise_hostname }}"  
11    ise_username: "{{ ise_username }}"  
12    ise_password: "{{ ise_password }}"  
13    ise_verify: true  
14    ise_debug: false  
15    register: ise_node_list  
16    timeout: 15  
17
```



Easy
Button

Demo – External Secret Management

- Replace the Certificate on ISE during the Business Day

 **San Diego, CA, USA ***
[PDT \(UTC -7\)](#)

Thu, Jun 12, 2025

11:30 am 

 **Philadelphia, PA, USA ***
[EDT \(UTC -4\)](#) 3 hour(s) ahead

Thu, Jun 12, 2025

2:30 pm 

- Show Playbooks
- ISE Should be finished restarting soon

Demo

Certificate Automation



P Phoenix0783 Smack-Fu Master, in training Popular
8y 50

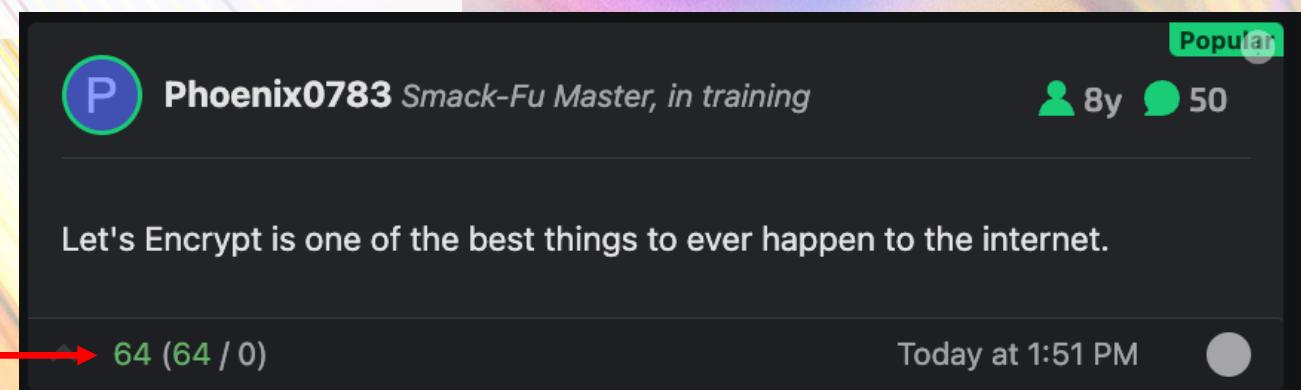
Let's Encrypt is one of the best things to ever happen to the internet.

64 (64 / 0) Today at 1:51 PM

Demo

Certificate Automation

No
Downvotes



A screenshot of a social media post from Phoenix0783. The post text reads: "Let's Encrypt is one of the best things to ever happen to the internet." It has 64 upvotes and 0 downvotes, indicated by a red arrow pointing to the upvote count. The post is 8 years old and has 50 comments. A "Popular" badge is visible in the top right corner.

Shorter Certificate Lifetimes are Coming...



Documentation Get Help Blog Donate ▾ About Us ▾

Donate Now

Blog

Announcing Six Day and IP Address Certificate Options in 2025

By Josh Aas · January 16, 2025

This year we will continue to pursue our commitment to improving the security of the Web PKI by introducing the option to get certificates with six-day lifetimes ("short-lived certificates"). We will also add support for IP addresses in addition to domain names. Our longer-lived certificates, which currently have a lifetime of 90 days, will continue to be available alongside our six-day offering. Subscribers will be able to opt in to short-lived certificates via a certificate profile mechanism being added to our ACME API.

Shorter Certificate Lifetimes Are Good for Security

<https://letsencrypt.org/2025/01/16/6-day-and-ip-certs/>

Shorter Certificate Lifetimes are Coming...



Documentation Get Help Blog Donate ▾ About Us ▾

Donate Now

Blog

We Issued Our First Six Day Cert

By Josh Aas · February 20, 2025

Earlier this year we [announced](#) our intention to introduce short-lived certificates with lifetimes of six days as an option for our subscribers. Yesterday we issued our first short-lived certificate. You can see the certificate at the bottom of our post, or [here](#) thanks to Certificate Transparency logs. We issued it to ourselves and then immediately revoked it so we can observe the certificate's whole lifecycle. This is the first step towards making short-lived certificates available to all subscribers.

The next step is for us to make short-lived certificates available to a small set of our subscribers so we can make sure our systems scale as expected prior to general availability. We expect this next phase to begin during Q2 of this year.

We expect short-lived certificates to be generally available by the end of this year.

<https://letsencrypt.org/2025/02/20/first-short-lived-cert-issued/>

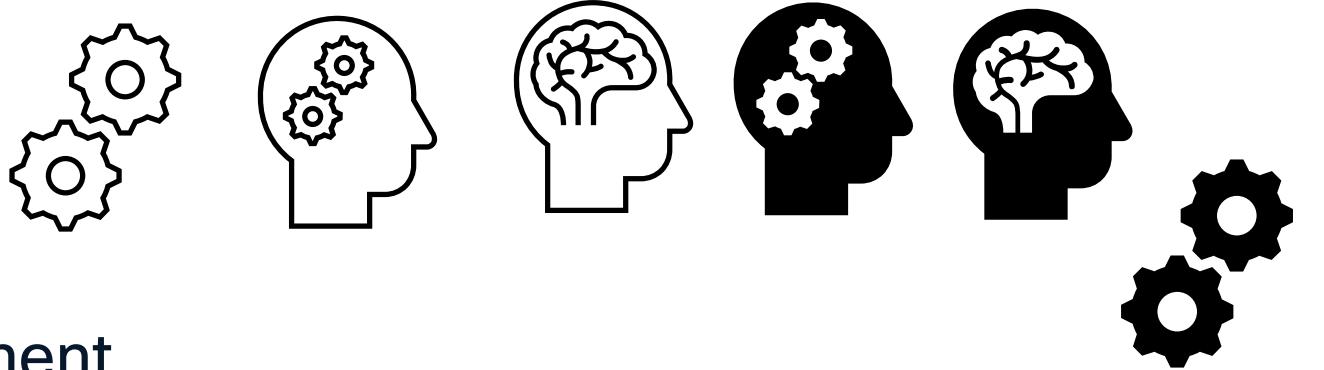
Demo - Let's Encrypt for Newbies

- Create a new Let's Encrypt Account
- Generate a new Let's Encrypt Certificate

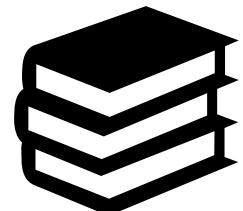
The background of the slide features a dynamic, abstract design composed of numerous thin, diagonal lines in various colors, including yellow, orange, red, purple, blue, and white. These lines create a sense of motion and depth against a light gray gradient.

Wrap this party up!!!

Recap



- Build your own Execution Environment
- Build Custom Credentials for your AuthC needs
- Certificates are great Automation Use Case



- Use AWX for your basic and for your advanced ansible playbooks
- Store your automation credentials securely in a External Vault
- Create Custom Credential Types when more than a 'password' is required
- Configure a credential (if req'd)
Create a Job Tempate
- Run the Job
Monitor the output
Schedule the job to run again!



Complete your session evaluations



Complete a minimum of 4 session surveys and the Overall Event Survey to be entered in a drawing to win 1 of 5 full conference passes to Cisco Live 2026.



Earn 100 points per survey completed and compete on the Cisco Live Challenge leaderboard.



Level up and earn exclusive prizes!



Complete your surveys in the Cisco Live mobile app.

Continue your education



Visit the Cisco Showcase for related demos



Book your one-on-one Meet the Engineer meeting



Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs



Visit the On-Demand Library for more sessions at www.CiscoLive.com/on-demand

Contact me at: Webex

Thank you

CISCO Live !

