Securing Secrets at Scale:
Integrating Ansible Automation with Conjur

James Freeman

@jamesfreeman959
James Freeman

Who Am I?

- Sr Technical Account Manager at AWS
- Published author (Ansible)
- Reiki Master Teacher
- Autism Dad

Agenda

- A brief history of secret management
- Introducing CyberArk Conjur
- A worked example
- Next steps
- Lessons Learned

Secret Management

- Unsecured secrets have always been a risk
- Our ever more connected world means even greater risks
- Generative AI lowers the bar for entry for bad actors
- Secrets stored in GitHub et al are the 21st Century “Post-It Note”
- Single factor/static secrets are no longer sufficient

Ansible Vault

- Integrates natively with Ansible
- Secrets encrypted using AES encryption
- But...
  - Vaults often secured by a single static password
  - If you can obtain the vault, you can brute force it eventually (no rate limiting)
  - All-or-nothing protection

AWX/Ansible Automation Platform

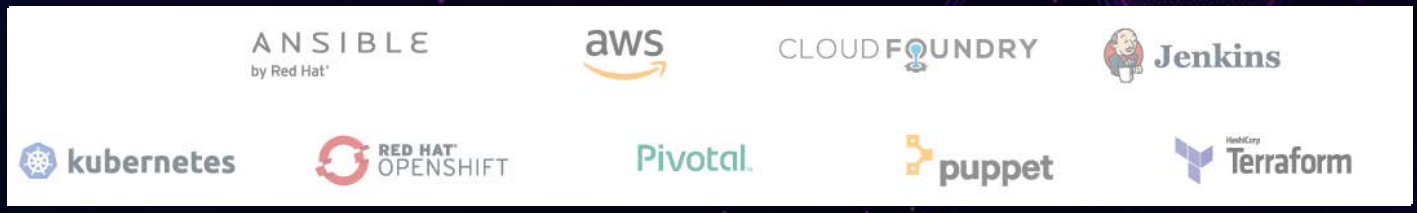
- Adds Role Based Access Control (RBAC) to secrets
- More difficult to obtain encrypted data
- But...
  - Ansible-only solution
  - Require something else to provide credentials to hosts
  - No dynamic/ephemeral secret management (yet)

Ansible should not be a high-value target

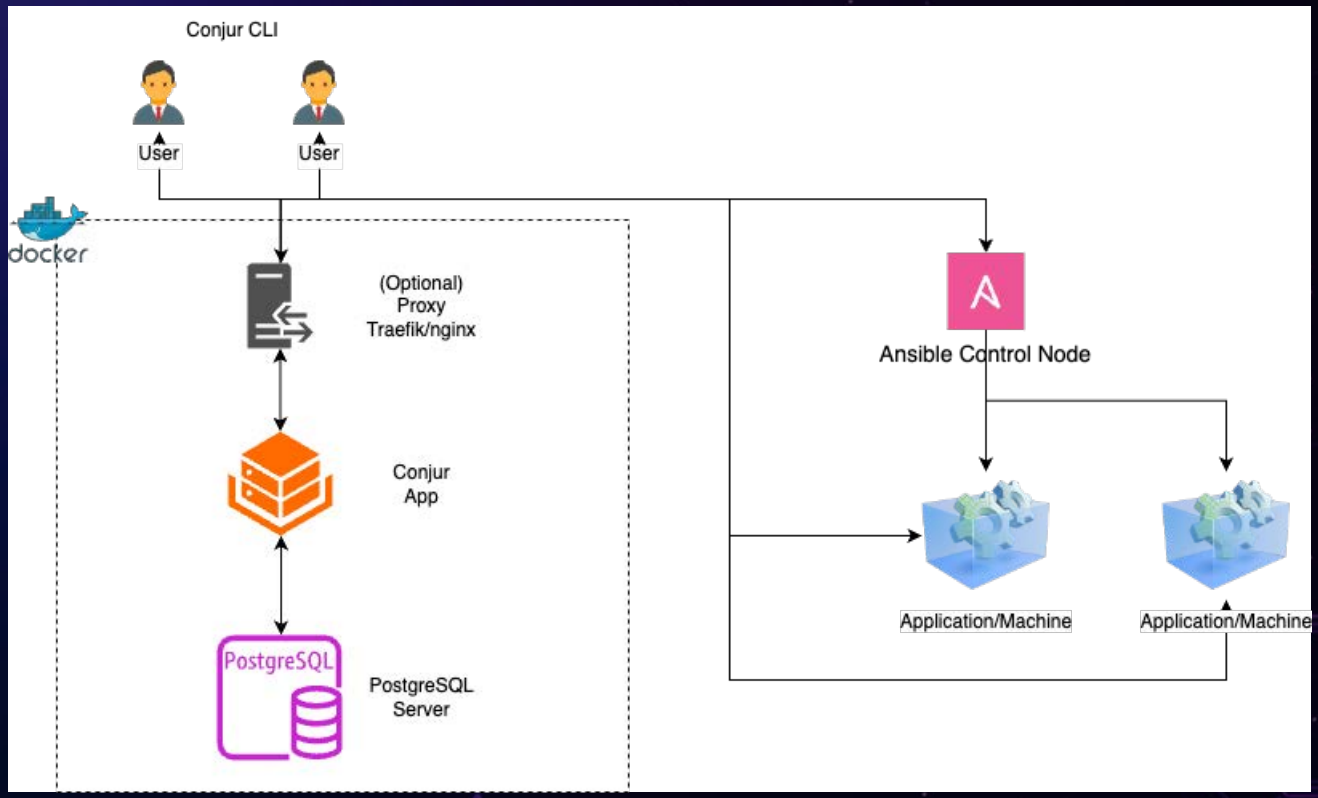
- Ansible host with passwords, SSH keys, vaults?
  - High value target
- Let’s not create this situation in our infrastructure
- Use a dedicated product designed for secret management

CyberArk Conjur

- Open Source and Enterprise variants
- Dedicated Secret Management platform
- Simple to deploy and maintain
- Provides:
  - Role Based Access Control for secrets
  - Secure, managed access for non-human entities (apps, machines)
  - Support for dynamic and ephemeral credentials
  - Native integration with many popular DevOps tools



Simple Architecture



Get started quickly with:
https://github.com/cyberark/conjur-quickstart

Ansible Integration

- Ansible <= 2.8 - Ansible Role
  - https://github.com/cyberark/ansible-conjur-host-identity
- Ansible 2.9+ - Ansible Collection
  - https://github.com/cyberark/ansible-conjur-collection
  - Enables configuration of Ansible Control Node with a Conjur Machine account
  - Lookup plugin for querying secret data

Working with Conjur

- No GUI/Web UI
- CLI/API Driven
- Self-contained Go CLI available for most platforms
- https://github.com/cyberark/conjur-cli-go

Conjur Example Policy - Ansible

```
$ cat conjur.yml
- !policy
  id: server

- !policy
  id: ansible

$ conjur policy load -b root -f conjur.yml
```

Conjur Example Policy - Ansible Control Node

```
$ cat ansible.yml
- !layer

- !host ansible-01

- !grant
  role: !layer
  member: !host ansible-01

- !host-factory
  layers: [ !layer ]

$ conjur policy load -b ansible -f ansible.yml
```

Host Factory

- Automated solution for adding hosts to a give machine role
  - e.g. Ansible Control Nodes
- Human user generates a short-lived “invite” token
- Control Node can be granted a Host Credential using this token
  - Grant only possible when token is valid
  - Machine Credentials live on (once granted) after invite token expires
- Can also add individual hosts manually if desired

Conjur Example Policy - Server

```
$ cat server.yml
- &variables
  - !variable host01.example.com/host
  - !variable host01.example.com/user
  - !variable host01.example.com/pass
  - !variable host02/host
  - !variable host02/user
  - !variable host02/pass

- !group secrets-users

- !permit
  resource: *variables
  privileges: [ read, execute ]
  roles: !group secrets-users

# Entitlements
- !grant
  role: !group secrets-users
  member: !layer /ansible
```

```
$ conjur policy load -b ansible -f server.yml
```

Setting Secret Data

- No secret data contained in policy
  - No secrets lying around in plaintext
- Use Conjur CLI

```
$ conjur variable set --id server/host01.example.com/host -v "192.168.1.57"
$ conjur variable set --id server/host01.example.com/user -v "ansiblectl"
$ conjur variable set --id server/host01.example.com/password -v "Sup3rS3cr3t!"
```

Retrieve with:

```
$ conjur variable get --id server/host01.example.com/password
```

Grant Ansible Control Host ID - Prep

```
$ ansible-galaxy collection install cyberark.conjur

$ openssl s_client -showcerts -connect conjur.example.com:443 < /dev/null
2> /dev/null | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' >
/tmp/conjur.pem

$ conjur hostfactory tokens create --hostfactory-id ansible
[
  {
    "expiration": "2025-01-10T21:55:09Z",
    "cidr": [],
    "token": "xxxxxxxxxxxxxx"
  }
]

$ export HFTOKEN="xxxxxxxxxxxxxx"
```

Grant Ansible Control Host ID - Grant

```
$ cat inventory
[servers]
conjur-ansiblectl.example.com

$ cat grant_conjur_id.yml
- hosts: servers
  roles:
    - role: cyberark.conjur.conjur_host_identity
      conjur_appliance_url: 'https://conjur.example.com'
      conjur_account: `demo'
      conjur_host_factory_token: "{{ lookup('env', 'HFTOKEN') }}"
      conjur_host_name: "{{ inventory_hostname }}"
      conjur_ssl_certificate: "{{ lookup('file', '/tmp/conjur.pem') }}"
      conjur_validate_certs: yes

$ ansible-playbook -i inventory grant_conjur_id.yml --become [--ask-become-
pass] [--ask-pass]
```

One time setup complete!

- Let’s automate
- Simple example - consider the following inventory file

```
$ cat inventory
[demo_servers]
conjur-ansible-node01.example.com
```

- We have no SSH keys set up, no password stored locally

The Playbook

```
- hosts: demo_servers
  vars:
    ansible_connection: ssh
    ansible_host: "{{ lookup('conjur_variable', 'server/' + inventory_hostname +
'/host', validate_certs=true) }}"
    ansible_user: "{{ lookup('conjur_variable', 'server/' + inventory_hostname +
'/user', validate_certs=true) }}"
    ansible_ssh_pass: "{{ lookup('conjur_variable', 'server/' + inventory_hostname +
'/pass', validate_certs=true) }}"

  tasks:
    - name: Get user name
      shell: whoami
      register: theuser

    - name: Get host name
      shell: hostname
      register: thehost

    - debug: msg="I am {{ theuser.stdout }} at {{ thehost.stdout }}"
```

Playbook Run

```
$ ansible-playbook -i inventory playbook.yml
PLAY [demo_servers]
***********************************************************************************
TASK [Gathering Facts]
***********************************************************************************
ok: [conjur-ansible-node01.example.com]

TASK [Get user name]
***********************************************************************************
changed: [conjur-ansible-node01.example.com]

TASK [Get host name]
***********************************************************************************
changed: [conjur-ansible-node01.example.com]

TASK [debug]
***********************************************************************************
ok: [conjur-ansible-node01.example.com] => {
    "msg": "I am ansiblectl at conjur-ansible-node01"
}
PLAY RECAP
***********************************************************************************
conjur-ansible-node01.example.com : ok=4    changed=2    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

What have we achieved?

- We have an Ansible Control Node with no static credentials stored on it, accessing another host!
- Ansible Control Node is no longer a high value target
- RBAC restricts which secrets can be accessed
- Scalable, cross platform solution

Next Steps

- Say we have a 2-tier application with a database backend
- Database credentials can be stored in Conjur using least privilege
  - Use a separate Policy with separate permissions
  - Application tier can read database password
  - Ansible Control Node cannot read this
  - Application tier cannot read machine credentials
- Add ephemeral/dynamic credentials on supported platforms (e.g. AWS)
- Audit use of Conjur

Things I learned

- You cannot have the Conjur CLI logged in as a user on the Ansible Control Node once integrated
  - It uses the Host identity, not the identity that you log in with
- The lookup plugin currently fails to verify the TLS certificate of the Conjur server if you use LetsEncrypt/ACME
  - Workarounds:
    - Turn off TLS verification (not good)
    - `export CONJUR_CERT_FILE=/etc/ssl/certs/ca-certificates.crt` (or equivalent for your distro)
    - Manually extract the entire CA trust chain into `/etc/conjur.pem`

Demo code

- https://github.com/jamesfreeman959/ansible-conjur-integration-demo

Questions?

Feedback (optional)