Intersight Automation and Monitoring with Ansible

DEVWKS-1542

Speaker:

David Soper, Technical Marketing Engineer

Table of Contents

Learning Objectives

Overview

Cisco Intersight™ provides intelligent cloud-based infrastructure management for the Cisco Unified Computing System™ (Cisco UCS®) and Cisco HyperFlex® platforms. Intersight offers an intelligent level of management that enables IT organizations to analyze, simplify, and automate their environments in more advanced ways than the prior generation of tools. Within this workshop, you will use Ansible to interact with the Intersight API and perform a variety of resource data collection and management tasks.

Prerequisites

While not required prior to starting this lab, familiarity with the Linux/MacOS command line and use of a text editor such as Vi will be helpful. Working knowledge of Ansible will also be helpful, but again is not required.

Getting Started

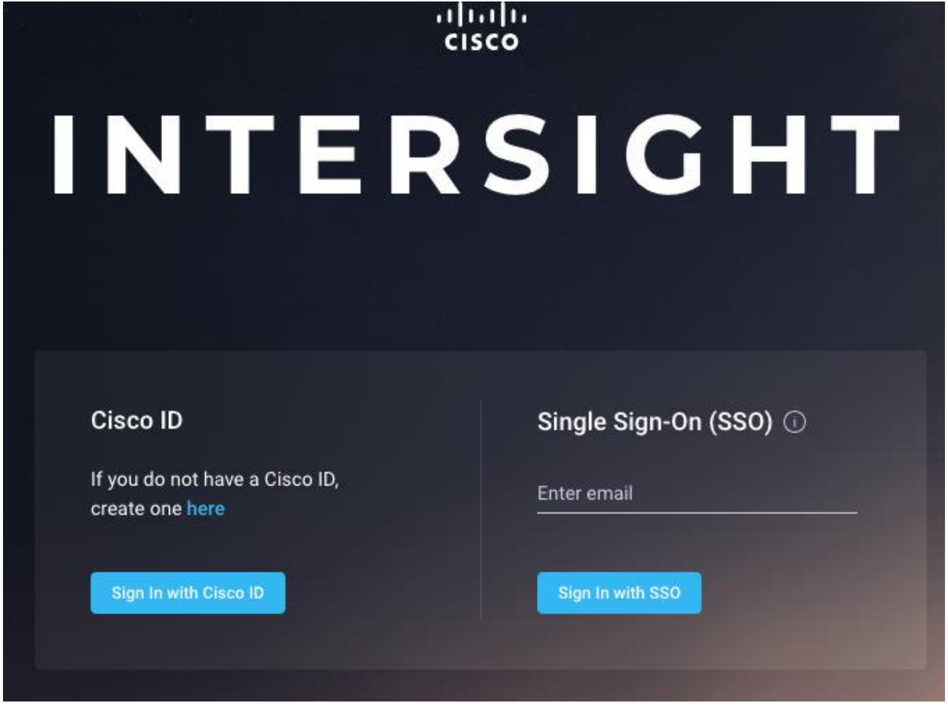
This lab will interact with Intersight's API from a Linux/MacOS workstation. A public internet connection is needed to communicate with Intersight.

Using your own Intersight Account (Optional)

If you already have an Intersight account, you can perform the tasks in this lab using your account at https://intersight.com. You will need to login using your credentials, and you will need to customize API key settings for your account.

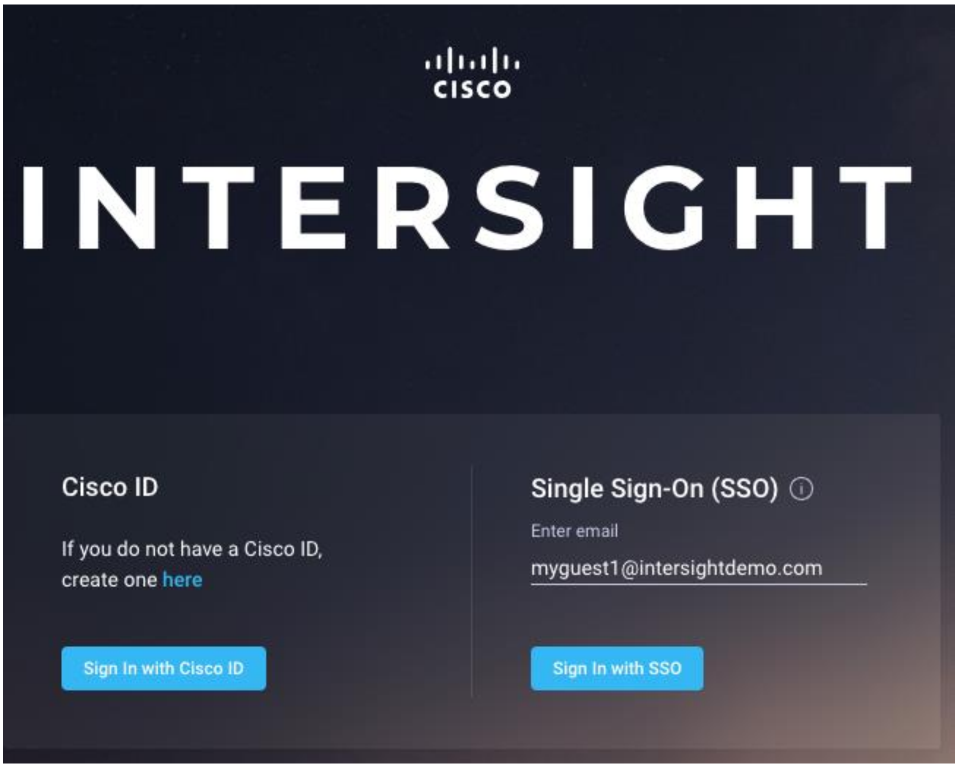
Sign In to the DevNet Intersight Account

To verify Intersight is accessible, point your browser (Chrome is preferred) to the following URL:
https://intersight.com/. You should see the Intersight Sign-On screen as shown below. Use the "Single Sign-On (SSO) option to login:



If you do not have a Cisco ID, create one here

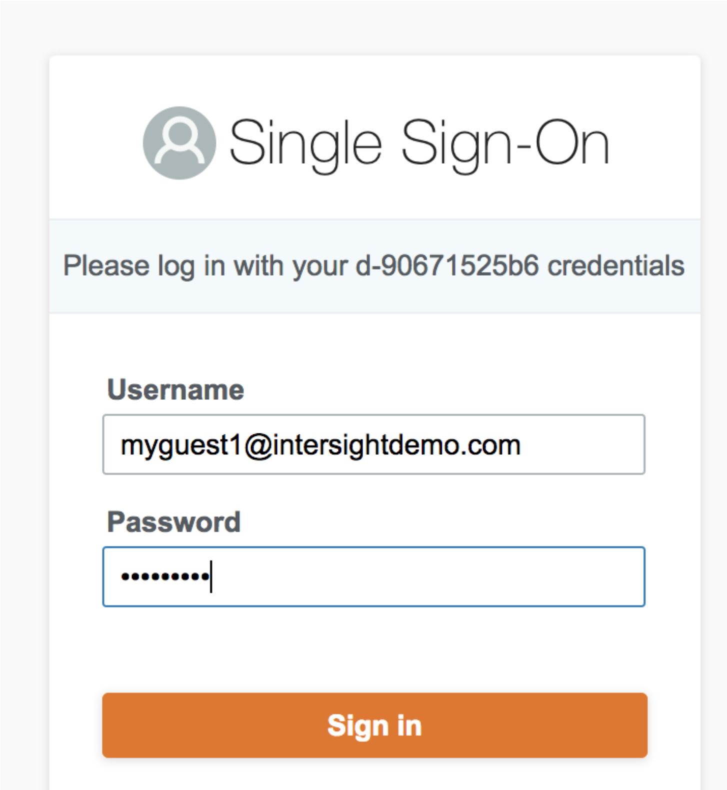
In the "Enter email" box, type "myguest<your workstation number> @intersightdemo.com". For example, if you are at workstation 1 you will enter "myguest1@intersightdemo.com":



If you do not have a Cisco ID, create one here

Enter email
myguest1@intersightdemo.com

You should be redirected to a Single Sign-On screen where you can provide the user’s password (given to you during the lab):

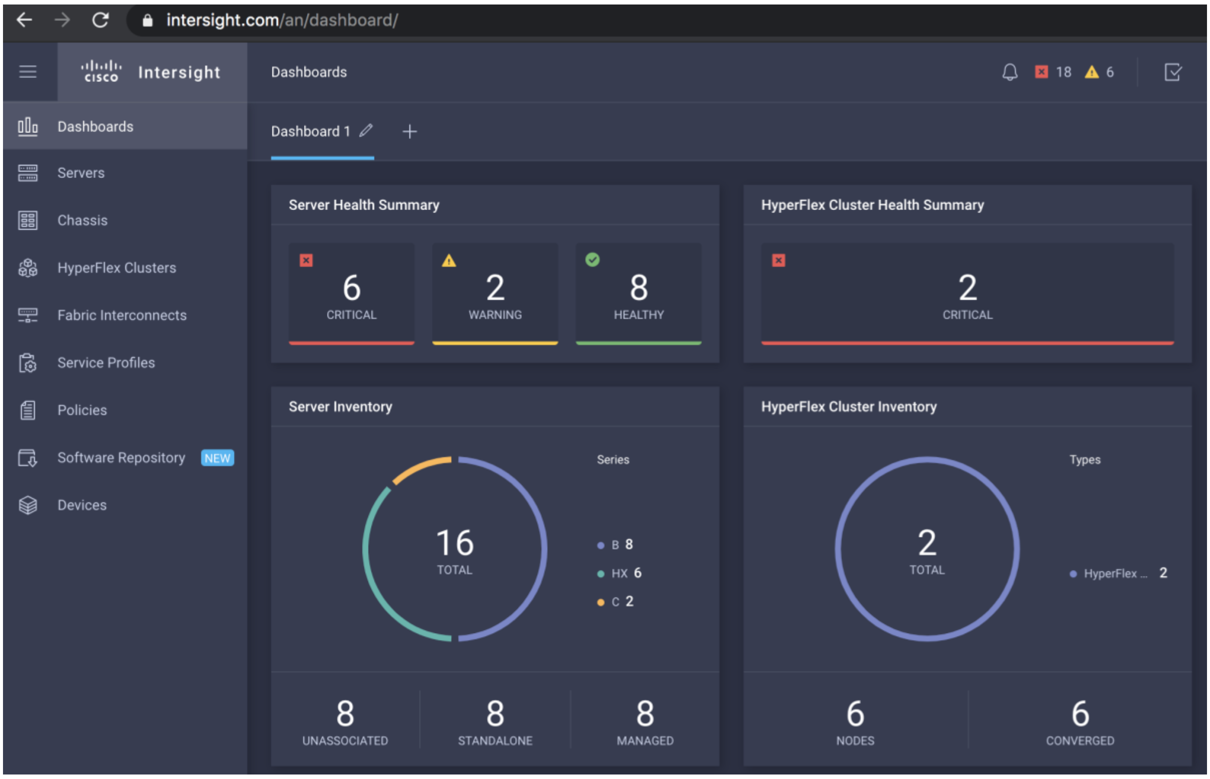


Please log in with your d-90671525b6 credentials

Username
myguest1@intersightdemo.com

Password
•••••••••

You may be prompted to take a site tour, but you can close that dialog box and you should see the Intersight dashboard:

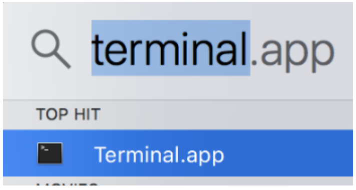


Ansible and API Configuration

The following tasks will configure Ansible on your system, retrieve a set of example Ansible playbooks, and customize the playbooks for use in this lab.

Task 1: Verify Ansible is Installed

Open a terminal on your workstation (you can spotlight search 🔍 for “terminal” if you are on MacOS), and type “ansible --version”. You should see 2.9 or later:



```
$ ansible --version
ansible 2.9.1
  config file = None
  configured module search path =
['/Users/dsoper/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ansible python module location =
/usr/local/lib/python3.6/site-packages/ansible
  executable location = /usr/local/bin/ansible
  python version = 3.6.5 (default, Apr 20 2018, 18:22:17) [GCC
4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.39.2)]
```

If Ansible is not installed or reports an older version, you can type “pip install -U ansible” to install/update.

Task 2: Ansible Collection for Intersight and Example Playbooks

Ansible is moving most modules out of Ansible core and into Collections. Cisco Collections are available in the Cisco namespace of Ansible Galaxy at https://galaxy.ansible.com/cisco. Collections provide a way to develop and distribute modules without dependencies and maintenance within Ansible core.

The ansible-galaxy command provides an interface to Ansible Galaxy and the ability to install and manage collections. Install instructions and additional details for the Cisco Intersight Collection are at https://galaxy.ansible.com/cisco/intersight. We will now install the collection and use some of the provided example playbooks.

In the terminal window, install cisco.intersight using ansible-galaxy:

```
$ ansible-galaxy collection install cisco.intersight
```

Ansible-galaxy should display installation status and the directory location of the collection:

```
Process install dependency map
Starting collection install process
Installing 'cisco.intersight:1.0.0' to
'/Users/admin/.ansible/collections/ansible_collections/cisco/i
ntersight'
```

Change directory to the collection install location and view the directory's contents:

```
$ cd
/Users/admin/.ansible/collections/ansible_collections/cisco/in
tersight
$ ls
Development.md MANIFEST.json  docs      plugins
FILES.json     README.md playbooks roles
```

Intersight modules, module_utils, and documentation are in the plugins directory. Several example playbooks are available in the playbooks directory. Change to the playbooks directory and view the example playbooks:

```
$ cd playbooks/
$ ls
cos_server_policies_and_profiles.yml    server_actions.yml
deploy_server_profiles.yml         server_firmware.yml
example_inventory              update_all_inventory.yml
roles                          update_standalone_inventory.yml
```

Task 3: Customize API Settings

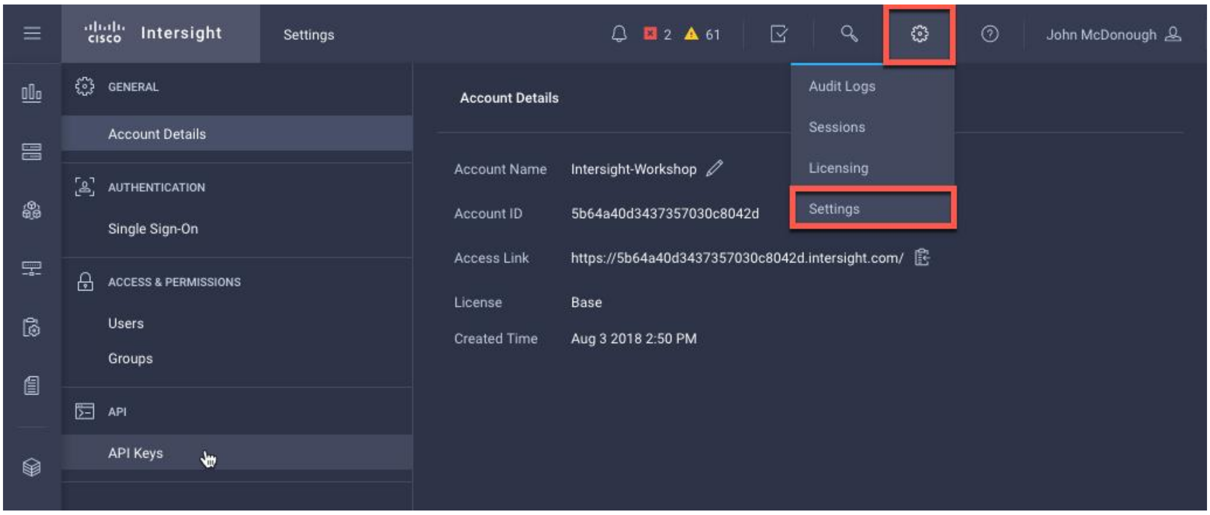
The example playbooks use API keys to authenticate with the Intersight API. API keys for your Intersight account will need to be created and specified in the playbooks. The playbooks also contain variables that you can edit as needed.

Step 1: Generate Intersight API Keys

Intersight *API keys* are two part keys; an *API key ID* and a *secret key*. The API Key ID is a multi-character string always visible after initial key creation. The secret key is an *RSA Private Key* only available at API Key creation.

To create API keys in Intersight, login at https://intersight.com/ as described above and perform the following steps:

1. Click the **Settings** icon.
2. Click **API Keys** in the left-hand navigation pane.
3. Click **Generate API Key**.
4. Enter a **Description** for the key
5. Click **Generate**.
6. Click the **Save Secret Key to text file** icon.



Account Details

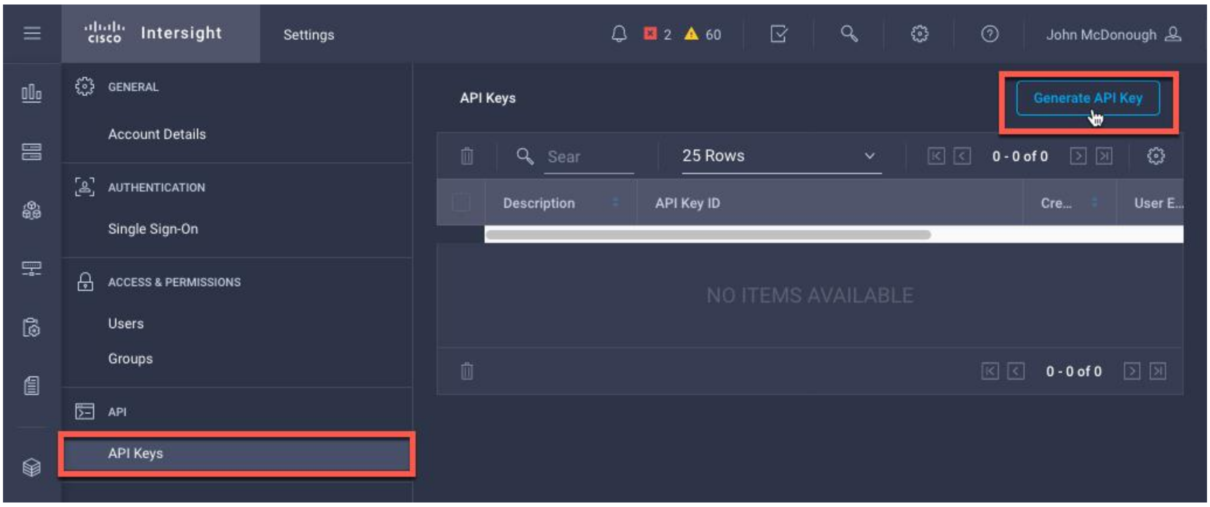
Account Name: Intersight-Workshop

Account ID: 5b64a40d3437357030c8042d

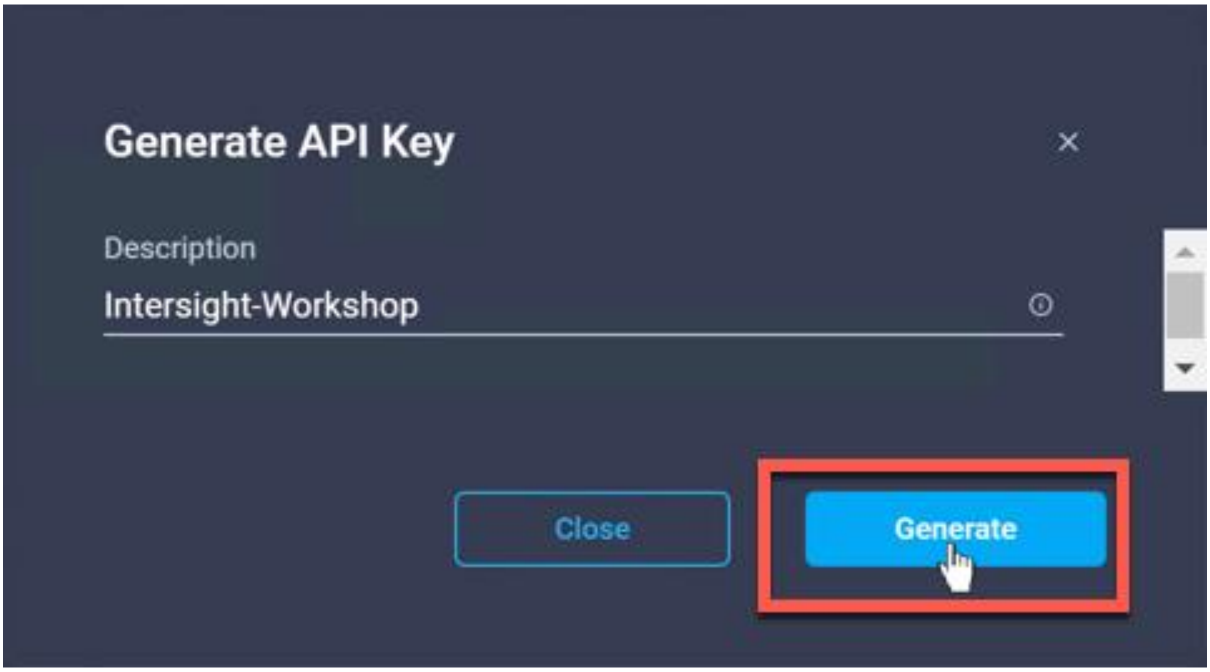
Access Link: https://5b64a40d3437357030c8042d.intersight.com/

License: Base

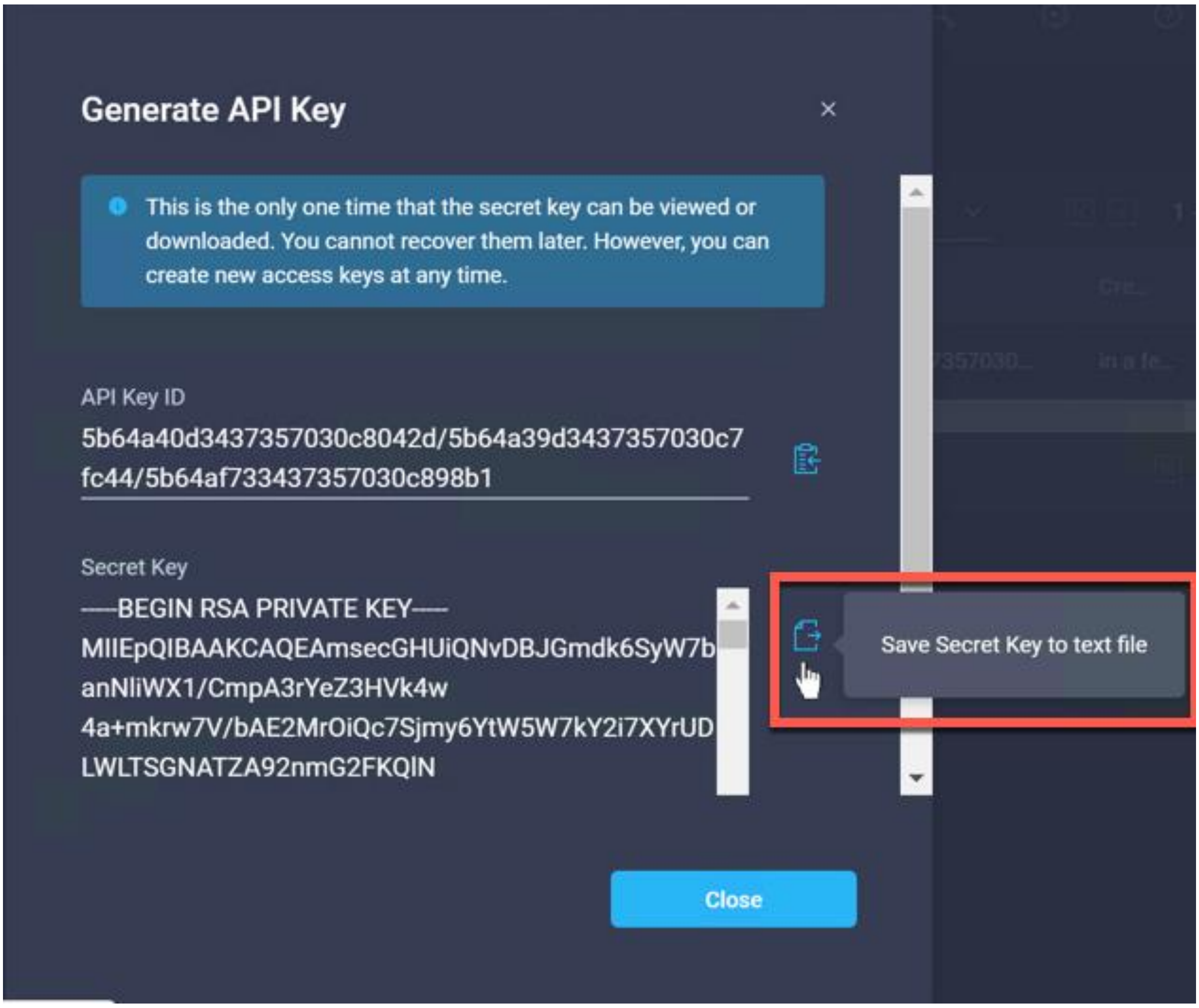
Created Time: Aug 3 2018 2:50 PM



NO ITEMS AVAILABLE



Description: Intersight-Workshop

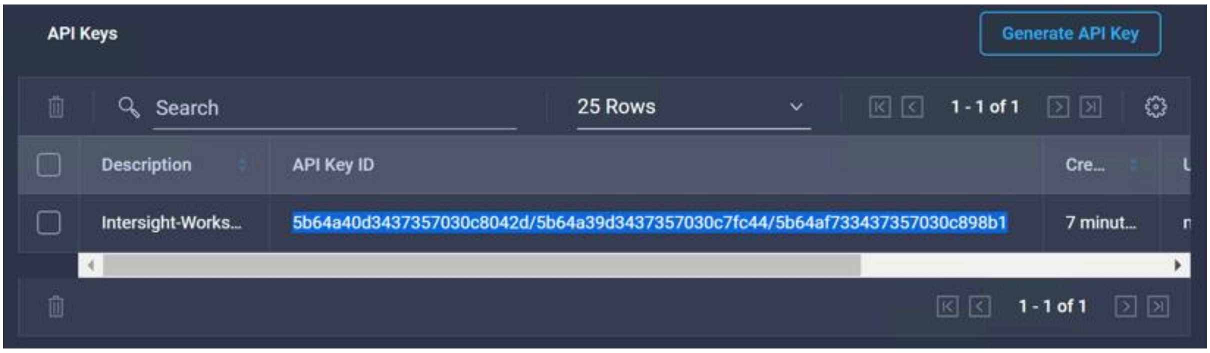


This is the only one time that the secret key can be viewed or downloaded. You cannot recover them later. However, you can create new access keys at any time.

API Key ID: 5b64a40d3437357030c8042d/5b64a39d3437357030c7fc44/5b64af733437357030c898b1

Secret Key:

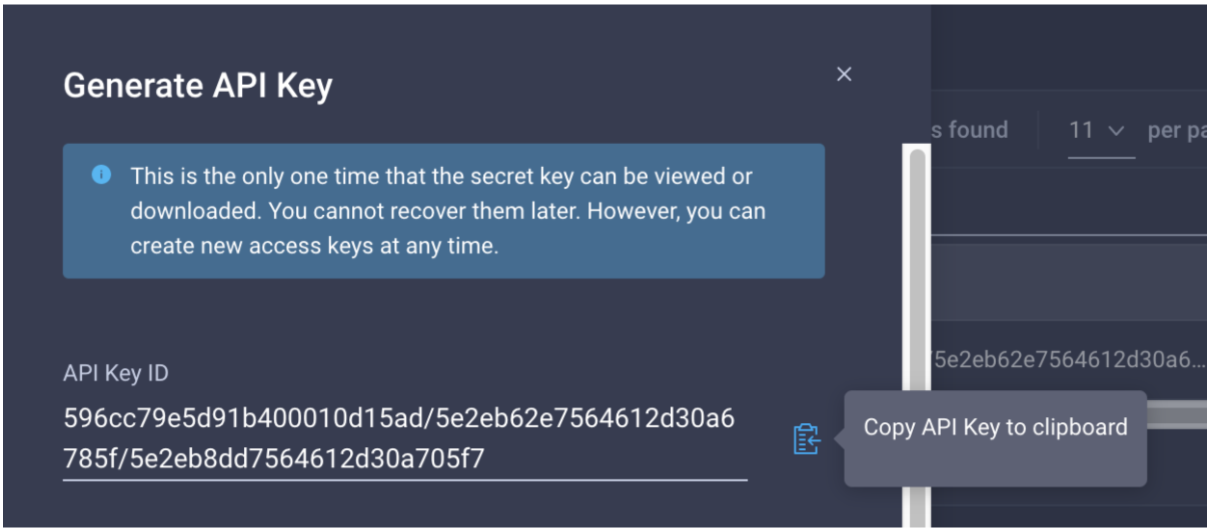
```
——BEGIN RSA PRIVATE KEY——
MIIEpQIBAAKCAQEAmsecGHUiQNvDBJGmdk6SyW7b
anNliWX1/CmpA3rYeZ3HVk4w
4a+mkrw7V/bAE2MrOiQc7Sjmy6YtW5W7kY2i7XYrUD
LWLTSGNATZA92nmG2FKQIN
```



☐	Description	API Key ID	Cre...
☐	Intersight-Works...	5b64a40d3437357030c8042d/5b64a39d3437357030c7fc44/5b64af733437357030c898b1	7 minut...

A "SecretKey.txt" file is downloaded to your default downloads location. For the DevNet workstations, the file should be downloaded to ~/Downloads/SecretKey.txt.

You can also copy the API Key ID to the clipboard for use in the next step:



This is the only one time that the secret key can be viewed or downloaded. You cannot recover them later. However, you can create new access keys at any time.

API Key ID
596cc79e5d91b400010d15ad/5e2eb62e7564612d30a6785f/5e2eb8dd7564612d30a705f7

Server Configuration Lab Tasks

Task 1: View and Customize Server Inventory

Step 1: Edit API Variables for the Server Inventory

The cisco.intersight collection playbooks directory has an example_inventory file which includes API key information (any Ansible variable source or playbook could contain key information, but we've used the inventory for this lab). Copy the example_inventory to a new file named inventory:

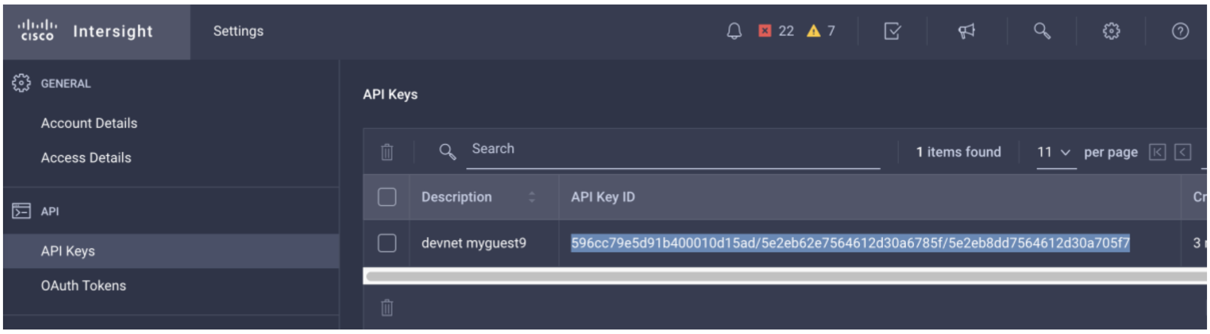
```
cp example_inventory inventory
```

Edit the inventory file to use your API private key (if you saved to a different location above) and your API Key ID:

```
[Intersight_HX]
sjc07-r13-501
sjc07-r13-503

[Intersight_Servers]

[Intersight:children]
```

```
Intersight_HX
Intersight_Servers

[Intersight:vars]
api_private_key=~/Downloads/SecretKey.txt
api_key_id=5a3404ac3768393836093cab/5b02fa7e6d6c356772394170/5b02fad
36d6c356772394449
```

Note that the "vars" section above contains API authentication info that will be used by the playbooks. You can paste the API Key ID saved above into the api_key_id line or you can get the Key ID from the Intersight UI (Settings (Gear icon)->API Keys):



☐	Description	API Key ID	Cr
☐	devnet myguest9	596cc79e5d91b400010d15ad/5e2eb62e7564612d30a6785f/5e2eb8dd7564612d30a705f7	3

Step 2: Update the Inventory for Standalone Servers

The collection playbooks directory has an update_standalone_inventory.yml playbook that will collect server information from Intersight and update the inventory file. Try to run the playbook using the current inventory file which has your API key information:

```
$ ansible-playbook -i inventory update_standalone_inventory.yml
[WARNING]: running playbook inside collection cisco.intersight


PLAY [Intersight]
*****************************************************************

TASK [cisco.intersight.intersight_facts] ***
```

The WARNING is ok for this lab, although typically you would place your playbooks outside of this collection directory since future updates to the collection may overwrite your local changes.

Don't worry if you get a failure, but do read the displayed message to determine what's causing the failure. If you see "Unauthorized" or similar wording, there is an issue with API key authentication (possibly due to issues in the vars section of the inventory file).

```
fatal: [sjc07-r13-501]: FAILED! => {"changed": false, "msg":
"API error: (401, 'HTTP Error 401: Unauthorized',
```

If you see a message that the module "jmespath" is not installed (jmespath is used for filtering return data in the playbook), you can install with pip:

```
pip install jmespath
```

Try to resolve any issues in running the playbook before continuing.

Task 2: View and Run Server Policy/Profile Configuration Playbooks

Step 1: View the Server Profile Configuration Playbook

The cloned repo has a server profile configuration playbook named cos_server_policies_and_profiles.yml. View the playbook which has several tasks to configure both server profiles and server policies:

```
- hosts: "{{ group | default('Intersight_Servers') }}"
  connection: local
  collections:
    - cisco.intersight
...
  tasks:
    # Get the Organization Moid used by all profiles and
policies
    - name: "Get Organization {{ org_name }} Moid"
      intersight_rest_api:
        <<: *api_info
        resource_path: /organization/Organizations
        query_params:
          $filter: "Name eq '{{ org_name }}'"
      register: org_resp
      delegate_to: localhost
      tags: always
    #
    # Configure profiles specific to server (run for each
server in the inventory)
    # Server Profiles role will register a profile_resp and
profile_resp list (from all hosts) can be used by policy tasks
    #
    - name: "Configure {{ profile_name }} Server Profile"
      intersight_rest_api:
...
```

Note the collections line near the top of the file that specifies collections used in the playbook. Modules like intersight_rest_api will be looked for in the cisco.intersight collection first. Use of the collections statement avoids the need to fully specify the module location in each task. intersight_rest_api can be used for the module name instead of fully qualifying cisco.intersight.intersight_rest_api.

Optional Steps to Check YAML Syntax

YAML syntax can be challenging to debug, but there are utilities such as yamllint that can help identify issues prior to running playbooks:

```
$ yamllint cos_server_policies_and_profiles.yml
```

(“pip install yamllint” if yamllint is not already installed on the workstation)

You can ignore "line too long" errors/warnings, and note that these can be turned off by creating and editing a ~/.config/yamllint/config file:

```
# yamllint config file
extends: default

rules:
  # 140 chars should be enough, but don't fail if a line is
longer
  line-length:
    max: 140
    level: warning
```

Step 2: Run the Server Profile Configuration Playbook

The cos_server_policies_and_profiles.yml playbook's 1st play will configure Server Profiles for each server in the inventory. Once profiles are created, policies are configured and the playbook uses roles defined in the "roles" subdirectory to configure each specific policy. We won't look at the specific roles just yet, but here's what a policy configuration role import should look like:

```
    #
    # Enclose policy tasks in a block that runs once
    # Policy API body is specified in a role specific vars section
for each role import
    # See https://intersight.com/apidocs/ or
https://intersight.com/mobrowser/ for information on setting
resource_path and api_body
    #
    - block:
        # Boot Order policy
        - import_role:
            name: policies/server_policies
          vars:
            resource_path: /boot/PrecisionPolicies
            api_body: {
              "Name": "COS-Boot",
              "ConfiguredBootMode": "Legacy",
              "BootDevices": [
                {
                  "ObjectType": "boot.LocalDisk",
                  "Enabled": true,
                  "Name": "Disk",
                  "Slot": "MRAID"
                },
                {
                  "ObjectType": "boot.VirtualMedia",
                  "Enabled": true,
                  "Name": "VM",
                  "Subtype": "cimc-mapped-dvd"
                }
              ],
              "Organization": {
                "Moid": "{{ org_resp.api_response.Moid }}"
              }
            }
          tags: boot_order
    ...
```

A few things to note in the example cos_server_policies_and_profiles.yml playbook:

- We use “block” to wrap the boot order and other policy tasks so those tasks can be run once and run on the localhost.
- The import_role line for boot_order above is used for Boot Order policies and will run tasks in roles/policies/ server_policies/tasks/main.yml. The vars defined above will be passed to those tasks.

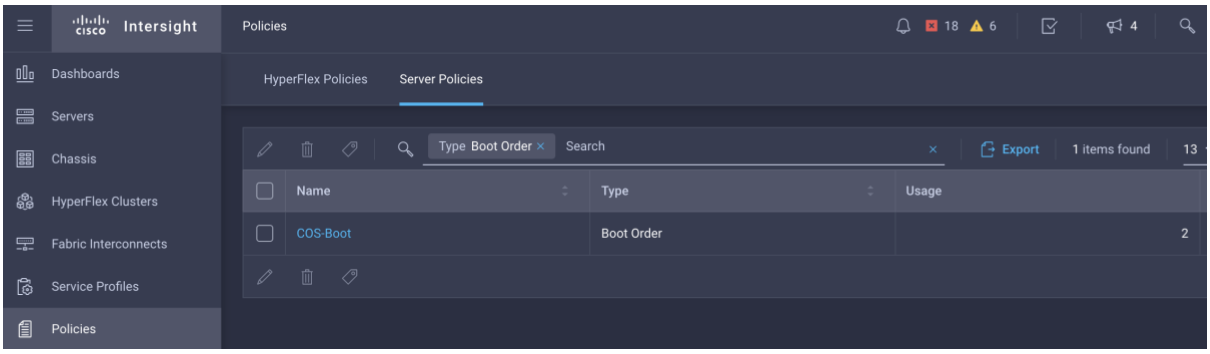
Run the playbook using the inventory file and pass a boot_order tag so only Boot Order policies are configured:

```
$ ansible-playbook -i inventory
cos_server_policies_and_profiles.yml --tags boot_order
```

You should see information on tasks run and any changes made:

```
PLAY [Intersight_Servers] ****************************************************

TASK [Get Organization DevNet Moid] ******************************************
ok: [C240M4-FCH1906V37P -> localhost]
ok: [C220-FCH2050V0LB -> localhost]

TASK [policies/server_policies : Configure COS-Boot Server Policy] *************
ok: [C220-FCH2050V0LB -> localhost]

TASK [policies/server_policies : set_fact] ***********************************
skipping: [C220-FCH2050V0LB]

TASK [policies/server_policies : Update Server Profiles used by COS-Boot Server
Policy (change may always be reported)] ***
skipping: [C220-FCH2050V0LB]

PLAY RECAP *******************************************************************
C220-FCH2050V0LB           : ok=2    changed=0    unreachable=0    failed=0
skipped=2    rescued=0    ignored=0
C240M4-FCH1906V37P         : ok=1    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

You can view the configuration that has been done in the Intersight UI from the Policies->Server Policies menu:



Name	Type	Usage
COS-Boot	Boot Order	2

Step 3: View Policy/Profile Roles

The roles described above use the intersight_rest_api Ansible module. Here’s the Boot Order role defined at roles/policies/server_policies/tasks/main.yml:

```
---
- name: "Configure {{ api_body.Name }} Server Policy"
  vars:
    # Create an anchor for api_info that can be used
throughout the file
    api_info: &api_info
      api_private_key: "{{ api_private_key }}"
      api_key_id: "{{ api_key_id }}"
      api_uri: "{{ api_uri | default(omit) }}"
      validate_certs: "{{ validate_certs | default(omit) }}"
      state: "{{ state | default(omit) }}"
  cisco.intersight.intersight_rest_api:
    <<: *api_info
    resource_path: "{{ resource_path }}"
    query_params:
      $filter: "Name eq '{{ api_body.Name }}'"
    api_body: "{{ api_body }}"
  register: policy_resp
```

Options to the intersight_rest_api module allow the user to directly specify the resource_path and api_body required to configure a given resource. Additional information on how the resource_path and api_body should be specified can be found in the Intersight API Reference at https://intersight.com/apidocs.

The intersight_rest_api module will check desired state and report if changes are made based on the api_body definition.

Step 4: Run the Profiles Playbook to Configure Multiple Servers

Re-run the server_profiles.yml playbook to configure the servers specified in the inventory:

```
$ ansible-playbook -i inventory cos_server_policies_and_profiles.yml
```

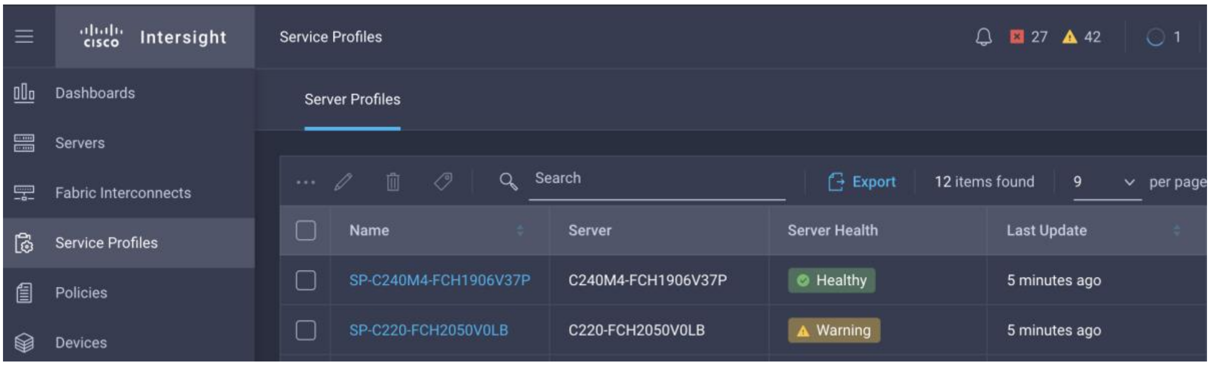
Ansible should display information for each task:

```
PLAY [Intersight_Servers] ******************************************************

TASK [Get Organization DevNet Moid] ********************************************
ok: [C240M4-FCH1906V37P -> localhost]
ok: [C220-FCH2050V0LB -> localhost]

TASK [Configure SP-C220-FCH2050V0LB Server Profile] ****************************
ok: [C220-FCH2050V0LB -> localhost]
ok: [C240M4-FCH1906V37P -> localhost]

TASK [policies/server_policies : Configure COS-Boot Server Policy] *************
ok: [C220-FCH2050V0LB -> localhost]

TASK [policies/server_policies : set_fact] *************************************
ok: [C220-FCH2050V0LB -> localhost]

TASK [policies/server_policies : Update Server Profiles used by COS-Boot Server
Policy (change may always be reported)] ***
changed: [C220-FCH2050V0LB -> localhost]

TASK [policies/server_policies : Configure COS-Adapter Server Policy] **********
ok: [C220-FCH2050V0LB -> localhost]
...
PLAY RECAP *********************************************************************
C220-FCH2050V0LB           : ok=26   changed=7    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

```
C240M4-FCH1906V37P         : ok=2    changed=0    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

In the above, you'll see tasks that "Update Server Profiles" used by each policy. Intersight's policy model provides linkages from the policy to the profiles that use the policy. This construct allows policies to be added or changed without requiring changes in the profile framework.

You can view the profiles and policies used by the profiles in the UI:



Name	Server	Server Health	Last Update
SP-C240M4-FCH1906V37P	C240M4-FCH1906V37P	Healthy	5 minutes ago
SP-C220-FCH2050V0LB	C220-FCH2050V0LB	Warning	5 minutes ago

Task 3: View and Run Server Configuration Playbooks

Step 1: View the Server Deploy Playbook

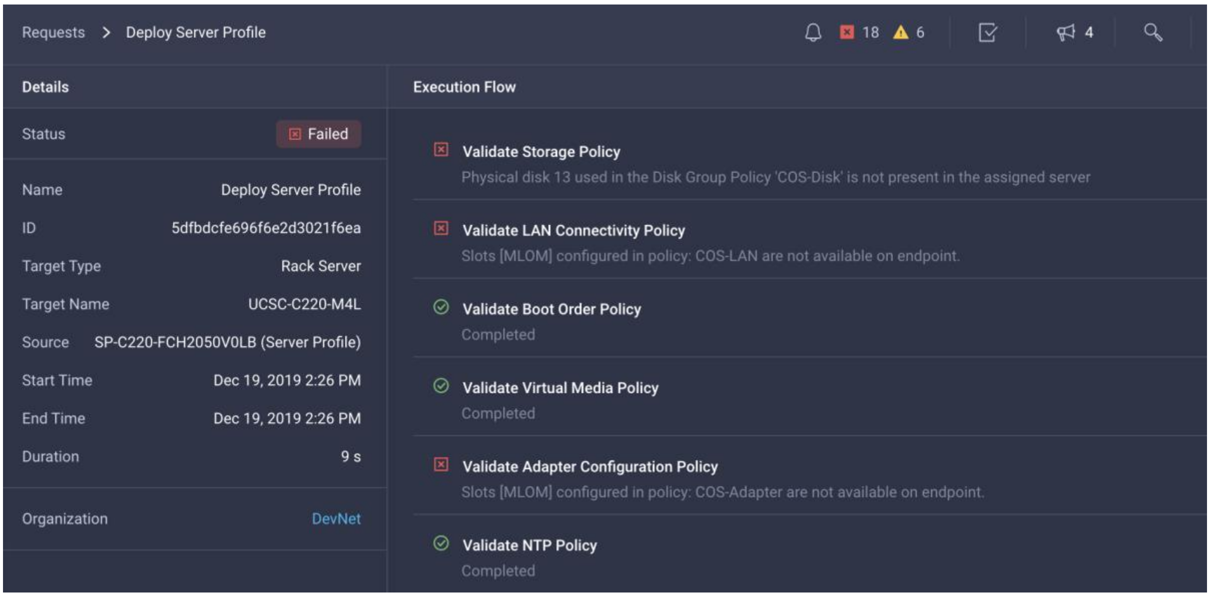
The collection has an example server profile deploy playbook named deploy_server_profiles.yml. View the playbook which has tasks to deploy or perform other profile actions for server profiles:

```
---
#
# Deploy Server Profiles
#
# The hosts group used is provided by the group variable or defaulted to
'Intersight_Servers'.
# You can specify a specific host (or host group) on the command line:
#   ansible-playbook ... -e group=<your host group>
#   e.g., ansible-playbook server_profiles.yml -e group=TME_Demo
#
- hosts: "{{ group | default('Intersight_Servers') }}"
  ...
  tasks:
    # Deploy (or perform other action)
    # action can be given on the command line if needed, e.g., ansible-playbook ...
-e action=Unassign
    # to delete a profile (profile must 1st be unassigned): ansible-playbook ... -e
state=absent -e action=No-op
    - name: Deploy (or user defined action) Server Profile
      cisco.intersight.intersight_rest_api:
        <<: *api_info
        resource_path: /server/Profiles
        query_params:
          $filter: "Name eq '{{ profile_name }}'"
        api_body: {
          "Action": "{{ action | default('Deploy') }}"
        }
      delegate_to: localhost
```

If you would like, run the playbook to attempt a server profile deployment:

```
$ ansible-playbook -i inventory deploy_server_profiles.yml
[WARNING]: running playbook inside collection cisco.github_intersight


PLAY [Intersight_Servers] ******************************************************

TASK [Deploy (or user defined action) Server Profile] **************************
changed: [C240M4-FCH1906V37P -> localhost]
changed: [C220-FCH2050V0LB -> localhost]

PLAY RECAP *********************************************************************
C220-FCH2050V0LB           : ok=1    changed=1    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
C240M4-FCH1906V37P         : ok=1    changed=1    unreachable=0    failed=0
skipped=0    rescued=0    ignored=0
```

The deploy should fail as the servers in this demo Intersight account are claimed read-only and may not have the correct configuration for validation checks. You can view the reason for failure in the Intersight UI:



Status ☒ Failed

Name Deploy Server Profile

ID 5dfbdcfe696f6e2d3021f6ea

Target Type Rack Server

Target Name UCSC-C220-M4L

Source SP-C220-FCH2050V0LB (Server Profile)

Start Time Dec 19, 2019 2:26 PM

End Time Dec 19, 2019 2:26 PM

Duration 9 s

Organization DevNet

☒ Validate Storage Policy
Physical disk 13 used in the Disk Group Policy 'COS-Disk' is not present in the assigned server

☒ Validate LAN Connectivity Policy
Slots [MLOM] configured in policy: COS-LAN are not available on endpoint.

✓ Validate Boot Order Policy
Completed

✓ Validate Virtual Media Policy
Completed

☒ Validate Adapter Configuration Policy
Slots [MLOM] configured in policy: COS-Adapter are not available on endpoint.

✓ Validate NTP Policy
Completed

The deploy playbook currently doesn’t check status of the deploy, but that could be added to the playbook as all UI visible activity is backed by the Intersight API.

You can also view and run the server_action.yml playbook if time permits. The file structure and instructions to run are similar to the deploy example above.

Please ask the instructor if you have any questions or encounter issues in running the server configuration tasks.

Congratulations on completing the Intersight and Ansible DevNet workshop!