

Intersight Automation and Monitoring with Ansible

DEVWKS-1542

Speaker:

David Soper, Technical Marketing Engineer

Table of Contents

Learning Objectives	2
Overview	2
Prerequisites	3
Getting Started.....	3
Using your own Intersight Account (Optional)	3
Sign In to the DevNet Intersight Account	3
Ansible and API Configuration.....	5
Task 1: Verify Ansible is Installed	6
Task 2: Get Example Playbooks	6
Task 3: Customize API Settings.....	6
Step 1: Generate Intersight API Keys	6
Server Configuration Lab Tasks	9
Task 1: View and Customize Server Inventory	9
Step 1: Edit API Variables for the Server Inventory	9
Step 2: Update the Inventory for Standalone Servers	9
Task 2: View and Run Server Configuration Playbooks.....	10
Step 1: View the Server Profile Configuration Playbook	10
Step 2: Run the Server Profile Configuration Playbook	11
Step 3: View Policy/Profile Roles.....	12
Step 4: Run the Profiles Playbook to Configure Multiple Servers	13
HyperFlex Configuration Lab Tasks (Optional)	16
Task 1: Switch User Role for HyperFlex Cluster Administration.....	16
Task 2: Generate API Keys for the new Role	16
Task 3: View and Customize HyperFlex Configuration Playbooks	17
Step 1: View the HyperFlex Configuration Playbook.....	17
Step 2: Run the HyperFlex Configuration Playbook.....	17
Step 3: View Policy/Profile Roles.....	18
Step 4: View the Inventory and host_vars.....	21
Step 5: Run Complete Playbook	21

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.

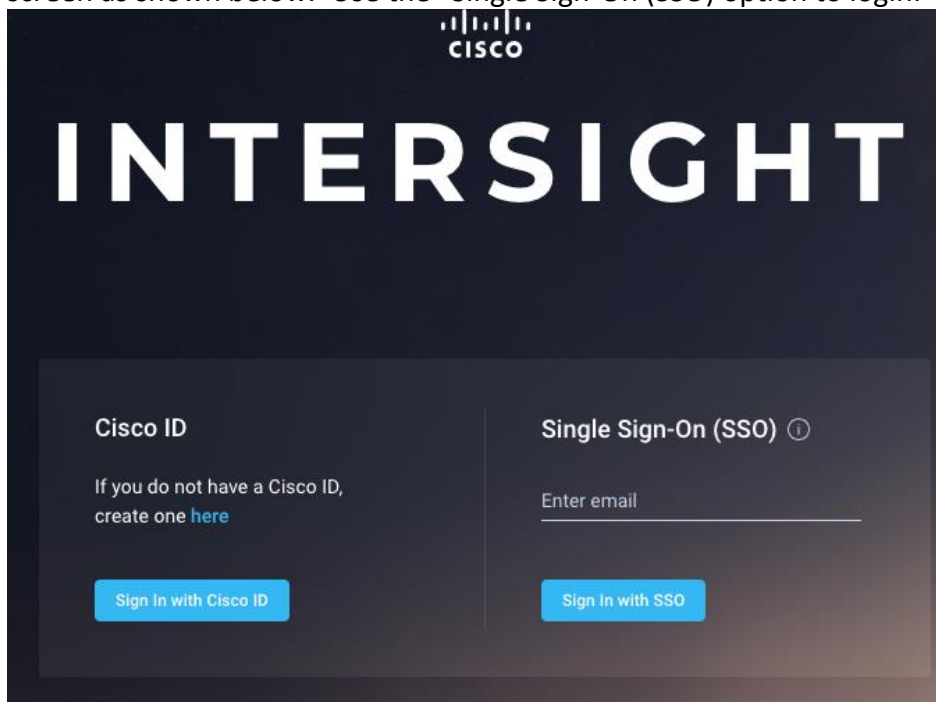
Using your own Intersight Account (Optional)

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.

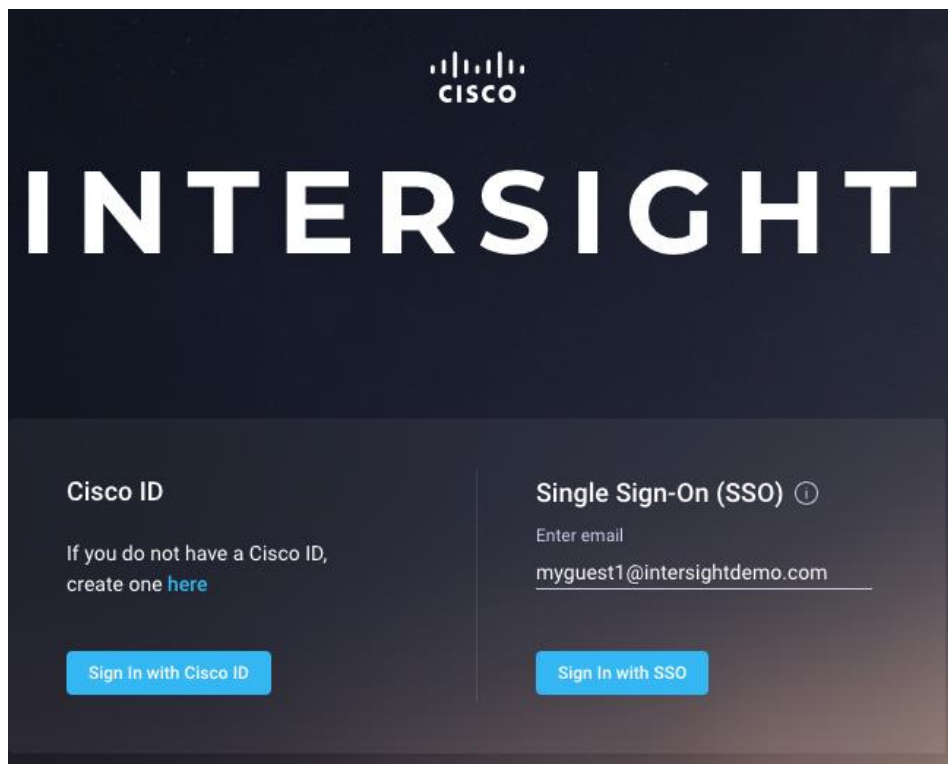
Sign In to the DevNet Intersight Account

To verify Intersight is accessible, point your browser (Chrome is preferred) to the following URL:

<https://5a3404ac3768393836093cab.intersight.com>. You should see the Intersight Sign-On screen as shown below. Use the "Single Sign-On (SSO)" option to login:



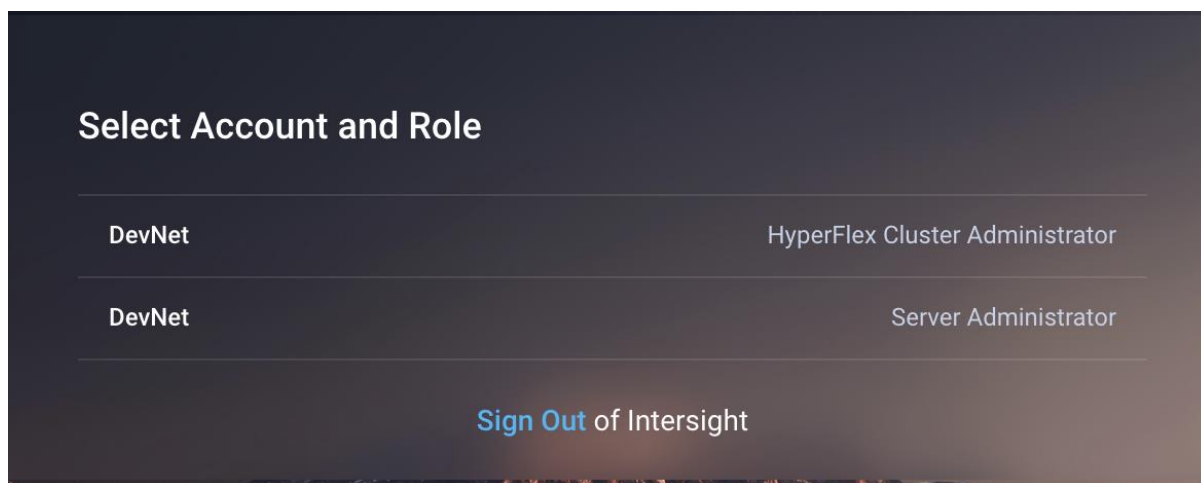
In the "Enter email" box, type "myguest<your workstation number>@intersightdemo.com". For example, if you are at workstation 1 you will enter "myguest1@intersightdemo.com":



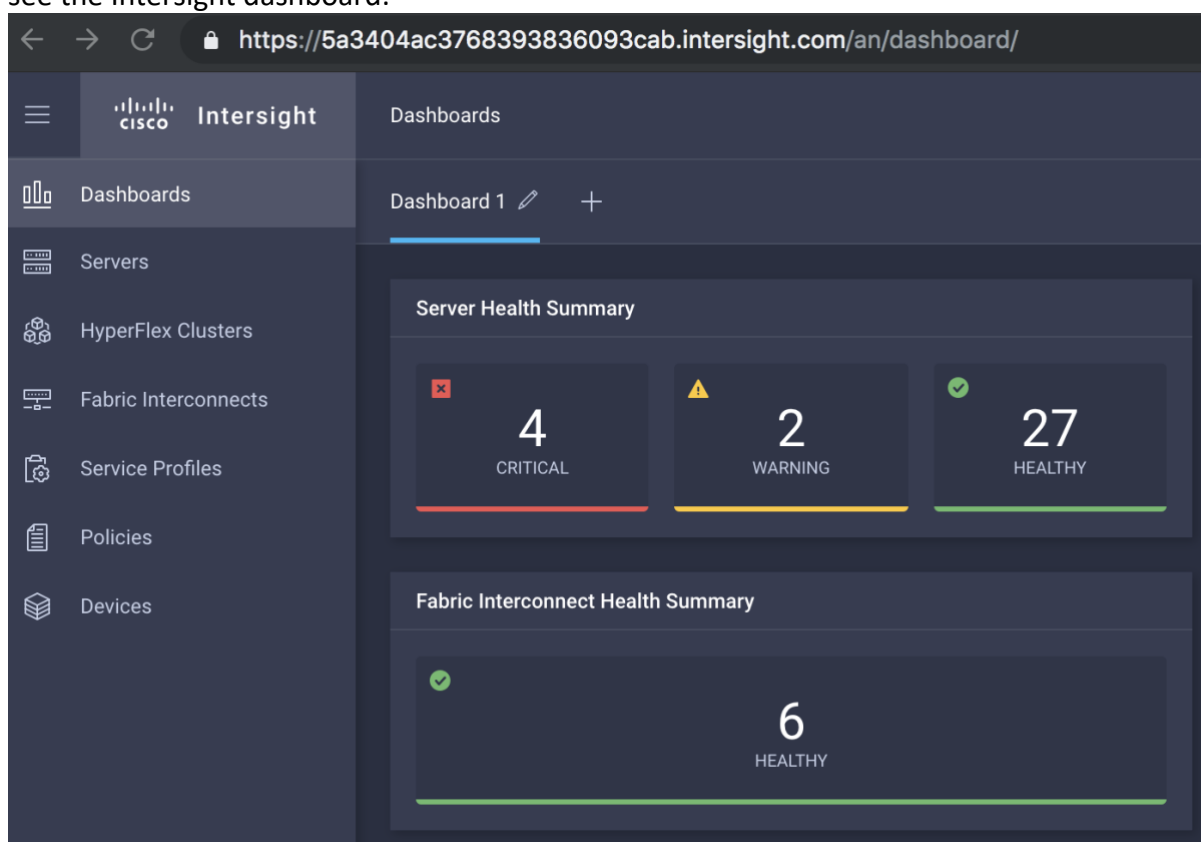
You should be redirected to a Single Sign-On screen where you can provide the user's password (given to you during the lab):

The image shows a "Single Sign-On" screen. At the top, there is a user icon followed by the text "Single Sign-On". Below this, a light blue banner contains the text "Please log in with your d-90671525b6 credentials". Underneath the banner, there are two input fields. The first is labeled "Username" and contains the email address "myguest1@intersightdemo.com". The second is labeled "Password" and contains a series of dots, indicating a masked password. Below these fields is a large orange button labeled "Sign in".

Select the Server Administrator role:




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

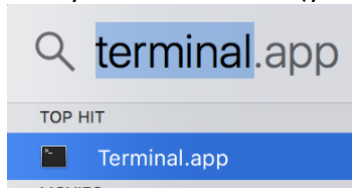



Ansible and API Configuration

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 2.8 or later:

```
$ ansible --version
ansible 2.8.0
  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: Get Example Playbooks

Several example playbooks are hosted on GitHub at <https://github.com/CiscoUcs/intersight-ansible>. In the terminal window, create a working directory and change to it:

```
$ mkdir ~/devwks-1542; cd ~/devwks-1542
```

Then use git to clone the intersight-ansible repo and change directories to it:

```
$ git clone https://github.com/CiscoUcs/intersight-ansible
Cloning into 'intersight-ansible'...
<snip>
Resolving deltas: 100% (74/74), done.
$ cd intersight-ansible/
```

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.

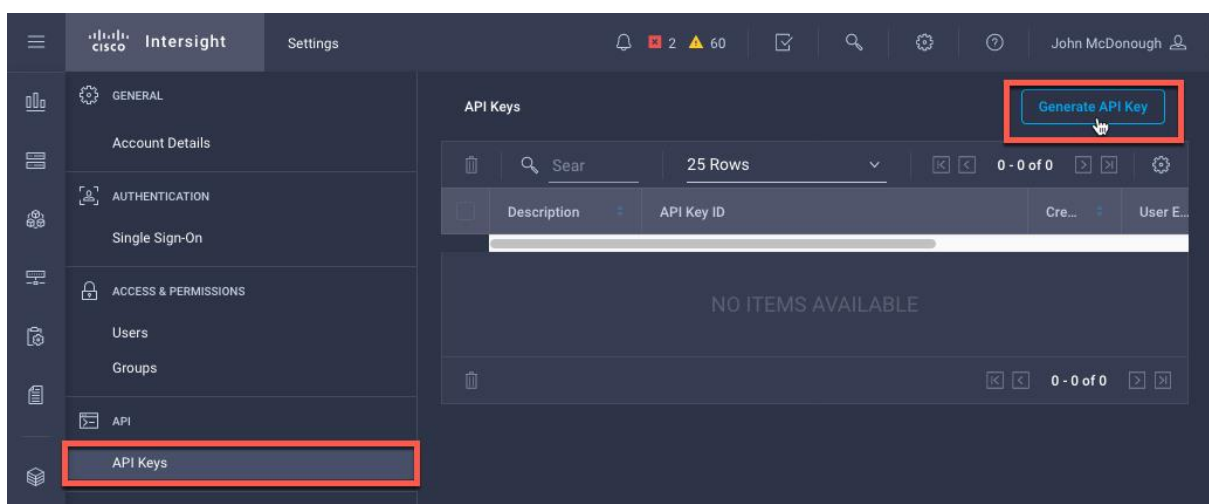
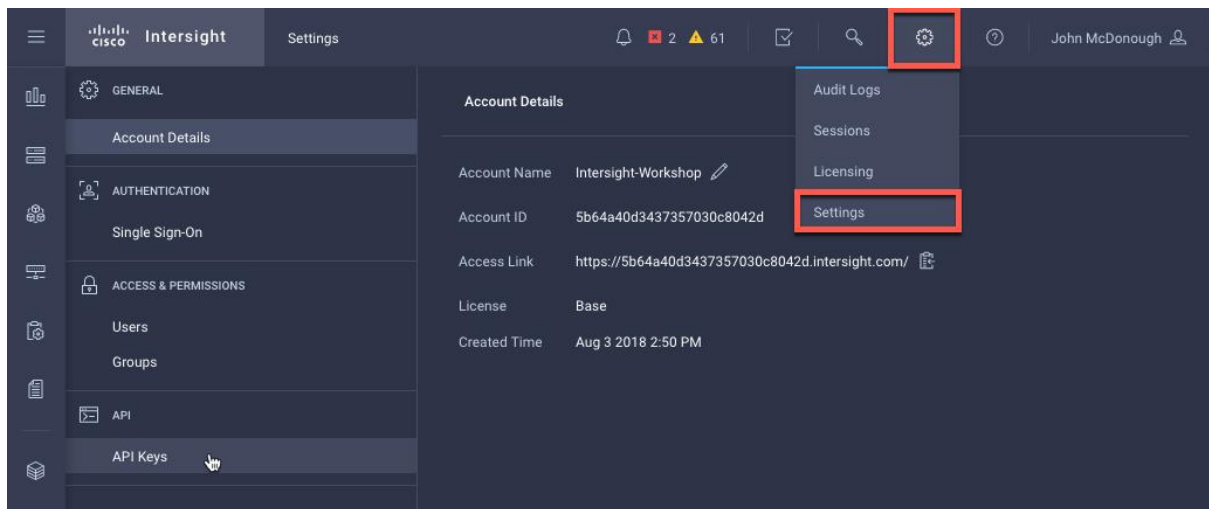
Step 1: Generate Intersight API Keys

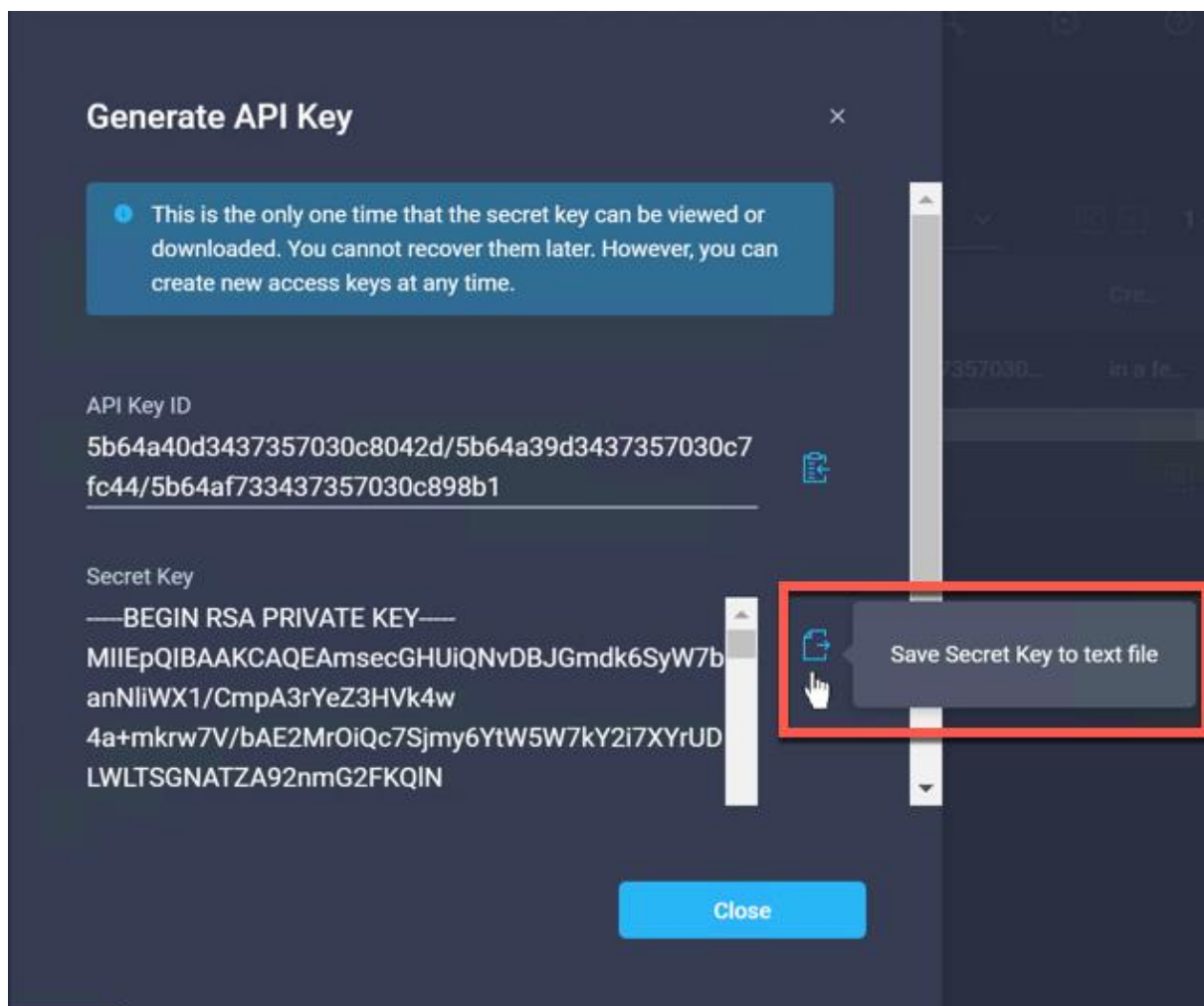
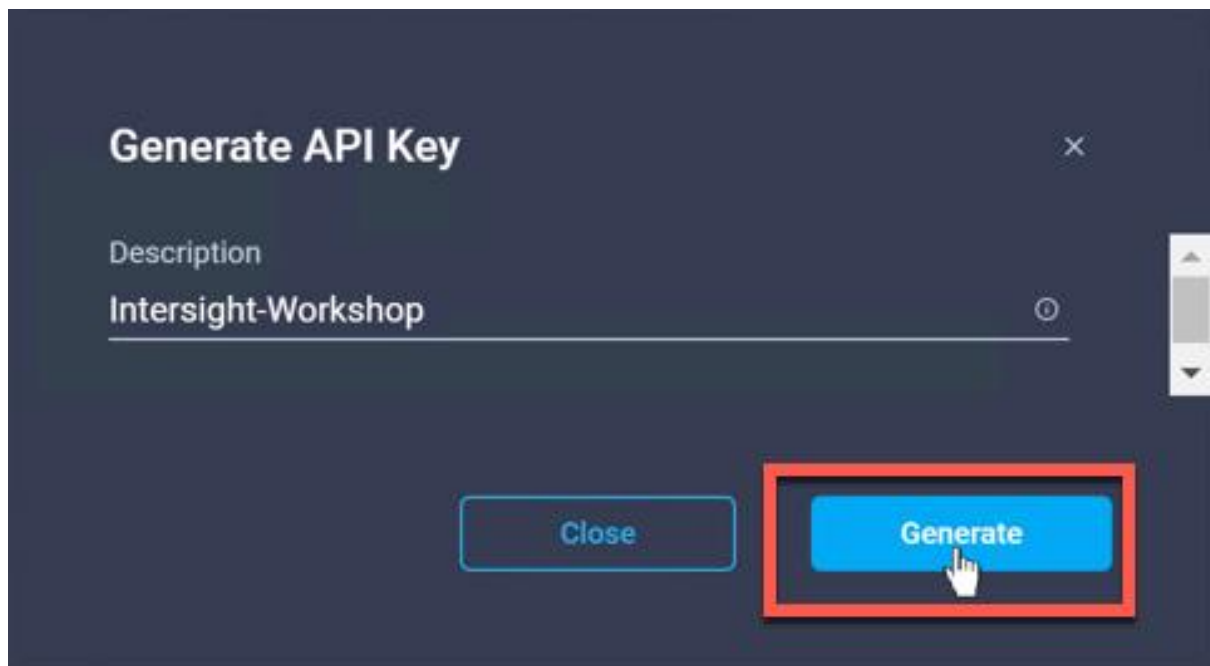
Intersight *API keys* are two part keys; an *API key ID* and a *secret key*. The API Key ID is a multi-character string always visible after initial key creation. The secret key is an *RSA Private Key* only available at API Key creation.

To create API keys in Intersight, login at <https://5a3404ac3768393836093cab.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.





API Keys				
Generate API Key				
	<input type="text" value="Search"/>	25 Rows	1 - 1 of 1	
<input type="checkbox"/>	Description	API Key ID	Cre...	L
<input type="checkbox"/>	Intersight-Works...	5b64a40d3437357030c8042d/5b64a39d3437357030c7fc44/5b64af733437357030c898b1	7 minut...	n

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

Server Configuration Lab Tasks

Task 1: View and Customize Server Inventory

Step 1: Edit API Variables for the Server Inventory

The intersight-ansible repository 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/5b02fa7e6d6c356772394170/5b02fad36d6c356772394449
```

Note that the "vars" section above contains API authentication info that will be used by the playbooks.

Step 2: Update the Inventory for Standalone Servers

The cloned repo 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
```

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

TASK [intersight_facts] ***
```

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',
```

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

Task 2: View and Run Server Configuration Playbooks

Step 1: View the Server Profile Configuration Playbook

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

```
tasks:
  #
  # 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:
      <<: *api_info
      resource_path: /server/Profiles
      query_params:
        $filter: "Name eq '{{ profile_name }}'"
      api_body: {
        "Name": "{{ profile_name }}",
        "AssignedServer": {
          "Moid": "{{ server_moid }}"
        }
      }
    register: profile_resp
    when: server_moid is defined
    delegate_to: localhost
    tags: server_profiles
```

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 server_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 server_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 of the repo to configure each specific policy. We won't look at the specific roles just yet, but here's what an 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 config
  - import_role:
    name: policies/server_policies
  vars:
    resource_path: /boot/PrecisionPolicies
    api_body: {
      "Name": "vmedia-local-disk",
      "ConfiguredBootMode": "Legacy",
      "BootDevices": [
        {
          "ObjectType": "boot.VirtualMedia",
          "Enabled": true,
          "Name": "remote-vmedia",
          "Subtype": "cimc-mapped-dvd"
        },
        {
          "ObjectType": "boot.LocalDisk",
          "Enabled": true,
          "Name": "localdisk",
          "Slot": ""
        }
      ]
    }
  tags: boot_order
```

...

A few things to note in the example `server_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/boot_order/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 server_profiles.yml --tags
boot_order

PLAY [Intersight_Servers]
*****

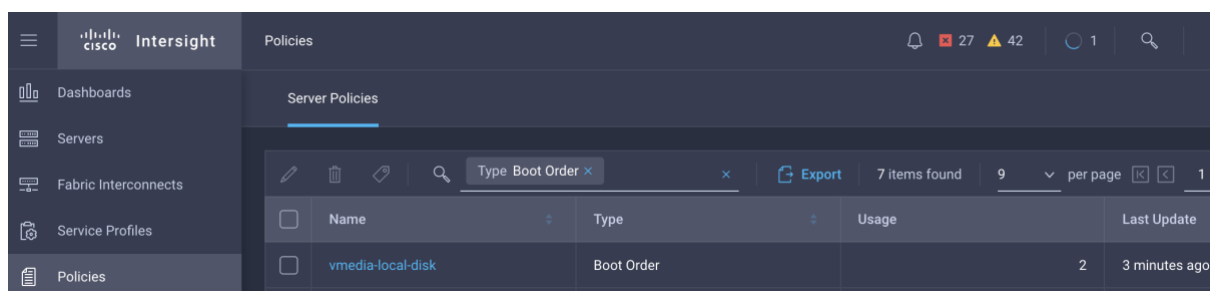
TASK [policies/server_policies : Configure vmedia-local-disk
Server Policy] ****
ok: [C240M4-FCH1906V37P]

TASK [policies/server_policies : set_fact]
*****
skipping: [C240M4-FCH1906V37P]

TASK [policies/server_policies : Update Server Profiles used
by vmedia-local-disk Server Policy] ***
skipping: [C240M4-FCH1906V37P]

PLAY RECAP
*****
*****
C240M4-FCH1906V37P          : ok=1    changed=0
unreachable=0    failed=0    skipped=2    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	Last Update
vmedia-local-disk	Boot Order	2	3 minutes ago

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/boot_order/tasks/main.yml`:

```

---
- name: "Configure {{ boot_policy }} Boot Policy"
  intersight_rest_api:
    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) }}"
    resource_path: /boot/PrecisionPolicies
    query_params:
      $filter: "Name eq '{{ boot_policy }}'"
    api_body: {
      "Name": "{{ boot_policy }}",
      "ConfiguredBootMode": "Legacy",
      "BootDevices": [
        {
          "ObjectType": "boot.VirtualMedia",
          "Enabled": true,
          "Name": "remote-vmmedia",
          "Subtype": "cimc-mapped-dvd"
        },
        {
          "ObjectType": "boot.LocalDisk",
          "Enabled": true,
          "Name": "localdisk",
          "Slot": "{{ disk_slot }}",
          "Bootloader": null
        }
      ]
    }

```

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 server_profiles.yml

PLAY [Intersight_Servers]
*****

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

```

```

TASK [policies/server_policies : Configure vmedia-local-disk
Server Policy] ****
ok: [C240M4-FCH1906V37P]

TASK [policies/server_policies : set_fact]
*****
ok: [C240M4-FCH1906V37P]

TASK [policies/server_policies : Update Server Profiles used
by vmedia-local-disk Server Policy] ***
ok: [C240M4-FCH1906V37P]

TASK [policies/server_policies : Configure nfs-cdd Server
Policy] *****
ok: [C240M4-FCH1906V37P]

TASK [policies/server_policies : set_fact]
*****
ok: [C240M4-FCH1906V37P]

TASK [policies/server_policies : Update Server Profiles used
by nfs-cdd Server Policy] ***
ok: [C240M4-FCH1906V37P]

TASK [policies/server_policies : Configure adaptive-memory
Server Policy] *****
ok: [C240M4-FCH1906V37P]

TASK [policies/server_policies : set_fact]
*****
ok: [C240M4-FCH1906V37P]

TASK [policies/server_policies : Update Server Profiles used
by adaptive-memory Server Policy] ***
ok: [C240M4-FCH1906V37P]

TASK [policies/server_policies : Configure snmp-local Server
Policy] *****
ok: [C240M4-FCH1906V37P]

TASK [policies/server_policies : set_fact]
*****
ok: [C240M4-FCH1906V37P]

TASK [policies/server_policies : Update Server Profiles used
by snmp-local Server Policy] ***
ok: [C240M4-FCH1906V37P]

TASK [policies/server_policies : Configure MVP_NTP Server
Policy] *****
ok: [C240M4-FCH1906V37P]

```

```

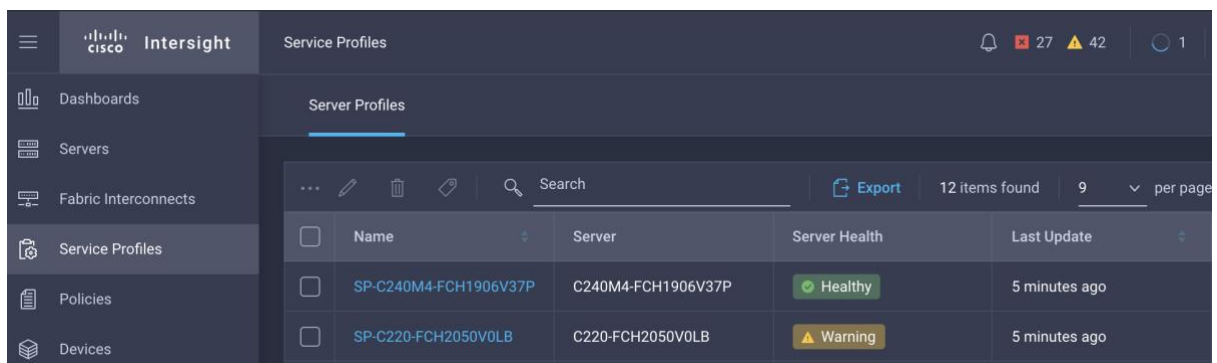
TASK [policies/server_policies : set_fact]
*****
ok: [C240M4-FCH1906V37P]

TASK [policies/server_policies : Update Server Profiles used
by MVP_NTP Server Policy] ***
ok: [C240M4-FCH1906V37P]

PLAY RECAP
*****
*****
C220-FCH2050V0LB           : ok=1    changed=0
unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
C240M4-FCH1906V37P        : ok=16    changed=0
unreachable=0    failed=0    skipped=0    rescued=0
ignored=0

```

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



	Name	Server	Server Health	Last Update
<input type="checkbox"/>	SP-C240M4-FCH1906V37P	C240M4-FCH1906V37P	Healthy	5 minutes ago
<input type="checkbox"/>	SP-C220-FCH2050V0LB	C220-FCH2050V0LB	Warning	5 minutes ago

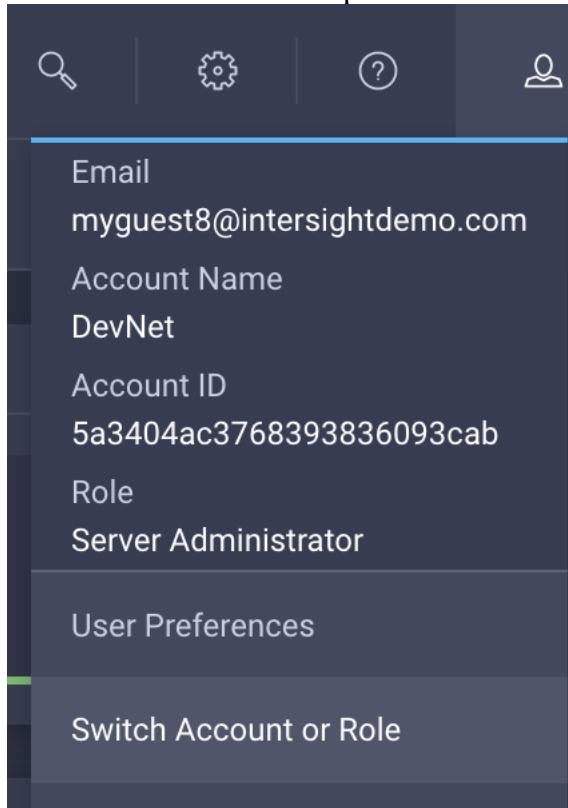
The server_profiles.yml file also includes a task to deploy profiles. Try to read the playbook and instructions on GitHub to run the deploy task. Please ask the instructor if you have any questions or encounter issues in running the server configuration tasks.

Congratulations on completing the Intersight and Ansible DevNet workshop!

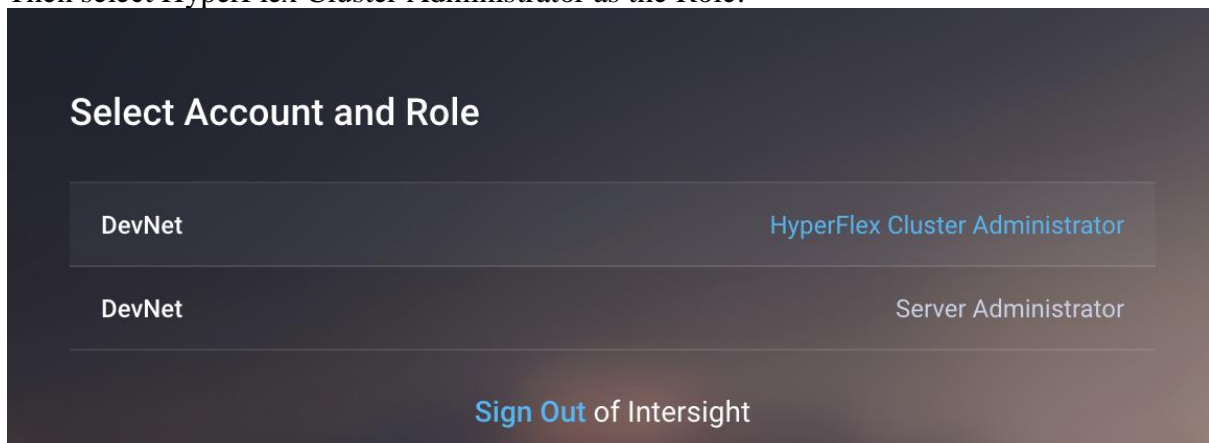
HyperFlex Configuration Lab Tasks (Optional)

Task 1: Switch User Role for HyperFlex Cluster Administration

For HyperFlex related configuration in the DevNet account, you will need to select Switch Account or Role from the profile menu in the upper right of the UI:



Then select HyperFlex Cluster Administrator as the Role:



You may have noticed during the server configuration portion of the lab that with a Server Administrator Role you were not able to see HyperFlex UI menus – now you will be able to see those in the UI/API!

Task 2: Generate API Keys for the new Role

See the API key generation section above for instructions on generating keys for the HyperFlex role and using in the inventory. You can create a new vars section in the inventory for the HX hosts if you would like.

Task 3: View and Customize HyperFlex Configuration Playbooks

Step 1: View the HyperFlex Configuration Playbook

The `hyperflex_cluster_profiles.yml` playbook has 2 plays and several tasks in each play that configure all of the policies and profiles needed to deploy multiple HyperFlex Edge clusters. The playbook uses roles defined in the “roles” subdirectory of the repo to configure each specific policy. We won’t look at the specific roles just yet, but we can edit variables to configure policies/profiles:

```
tasks:
  # Policy configuration is shared across clusters, so
  # enclose in a block and only run once
  - block:
    # DNS
    - import_role:
      name: policies/hyperflex_policies/dns_ntp_timezone
    vars:
      hx_sys_config_policy: devnet-dns-ntp
      hx_sys_config_timezone: America/Los_Angeles
      hx_sys_config_dns_servers:
        - 171.70.168.183
      hx_sys_config_ntp_servers:
        - ntp.esl.cisco.com
    tags: ['dns']
...
  delegate_to: localhost
  run_once: true
```

A few things to note in the example playbook:

- We use “block” to wrap the policy related tasks so there are common directives for all the enclosed tasks. “run_once: true” avoids repeated execution for every “host” in the inventory. “delegate_to: localhost” is used to run with the same python interpreter used to install Ansible (what “which python” shows for the current user).
- The `import_role` line above is used for DNS policies and will run tasks in `roles/policies/hyperflex_policies/dns_ntp_timezone/tasks/main.yml`. The vars defined above will be pass the defined vars to those tasks.
- A “dns” tag is defined so we can run only this part of the playbook if needed.

You can change any of the `hx_*` variables in the vars section to customize for your workstation/user. For example, the `hx_sys_config_policy` value can be changed to `devnet-dns-ntp-<workstation number>`, e.g., “devnet-dns-ntp-1”.

Step 2: Run the HyperFlex Configuration Playbook

Run the playbook after any edits and pass a dns tag so only DNS policies are configured:

```
$ ansible-playbook -i inventory hyperflex_cluster_profiles.yml
--tags dns
```

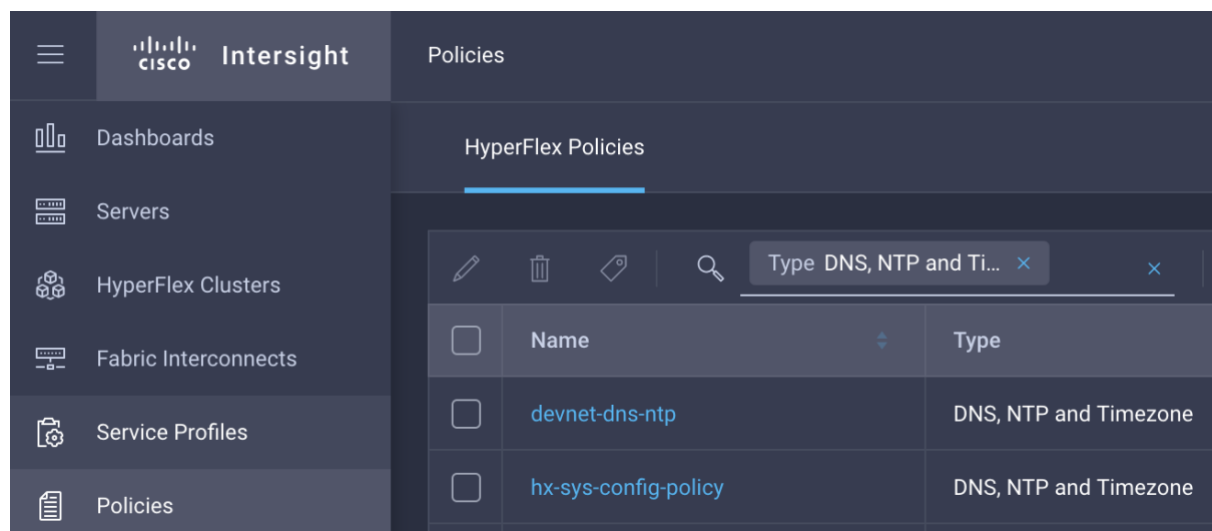
```
PLAY [Intersight_HX]
*****
```

```
TASK [policies/hyperflex_policies/dns_ntp_timezone : Configure
devnet-dns-ntp DNS, NTP, and Timezone Policy] ***
ok: [sjc07-r13-501]
```

PLAY RECAP

```
*****
*****
sjc07-r13-501          : 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->HyperFlex Policies menu:



Step 3: View Policy/Profile Roles

The DNS and other roles described above use the `intersight_rest_api` Ansible module.

Here's the DNS role defined at

`roles/policies/hyperflex_policies/dns_ntp_timezone/tasks/main.yml`:

```
---
- name: "Configure {{ hx_sys_config_policy }} DNS, NTP, and
Timezone Policy"
  intersight_rest_api:
    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) }}"
    resource_path: /hyperflex/SysConfigPolicies
    query_params:
      $filter: "Name eq '{{ hx_sys_config_policy }}'"
    api_body: {
      "Name": "{{ hx_sys_config_policy }}",
      "Timezone": "{{ hx_sys_config_timezone }}",
      "DnsServers": "{{ hx_sys_config_dns_servers }}",
      "NtpServers": "{{ hx_sys_config_ntp_servers }}"
    }
```

```
register: sys_config
```

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. API responses are returned by the module and can be registered for use in subsequent tasks:

```
register: sys_config
```

Each HyperFlex policy role used in this lab registers return variables for use in HyperFlex Cluster Profile configuration as shown in the 2nd play of the `hyperflex_cluster_profiles.yml`:

```
# Node and Profile configuration can be performed on
multiple clusters at once
- block:
    # Node IP and Hostname
    - import_role:
        name: policies/hyperflex_policies/node_ip_hostname
        tags: ['nodes']
    # Cluster Profile
    - import_role:
        name: service_profiles/hyperflex_cluster_profiles
        tags: ['profile']
  vars:
    <<: *api_info
    hx_node_config_policy: "{{ inventory_hostname }}"-node-
config"
    hx_mgmt_netmask: 255.255.254.0
    hx_mgmt_gateway: 172.28.224.1
    hx_profile_name: "{{ inventory_hostname }}"
    hx_data_platform_version: 3.5(2a)
    hx_mgmt_platform: EDGE
    hx_profile_description: 3 node Edge M5 cluster
    delegate_to: localhost
```

Again we enclose tasks in a block to share vars and other directives across the tasks. For this play, multiple node policies and cluster profiles can be configured at the same time. The `import_role` line above for `hyperflex_cluster_profiles` will run the tasks in `roles/service_profiles/hyperflex_cluster_profiles/tasks/main.yml`:

```
---
# Cluster Profile Attributes
- name: "Configure {{ hx_profile_name }} HyperFlex Cluster
Profile"
  intersight_rest_api:
    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) }}"
resource_path: /hyperflex/ClusterProfiles
query_params:
    $filter: "Name eq '{{ hx_profile_name }}'"
api_body: {
    "Name": "{{ hx_profile_name }}",
    "HxdpVersion": "{{ hx_data_platform_version }}",
    "MgmtPlatform": "{{ hx_mgmt_platform }}",
    "Description": "{{ hx_profile_description }}",
    "StorageDataVlan": {
        "VlanId": "{{ hx_storage_vlan }}"
    },
    "MgmtIpAddress": "{{ mgmt_ip }}",
    "Tags": [
        {
            "Key": "Demo",
            "Value": "Yes"
        }
    ]
}

# Update Cluster Profile if Policies defined
- name: "Policies for {{ hx_profile_name }} HyperFlex Cluster Profile"
  intersight_rest_api:
    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) }}"
    resource_path: /hyperflex/ClusterProfiles
    query_params:
        $filter: "Name eq '{{ hx_profile_name }}'"
    api_body: {
        "Name": "{{ hx_profile_name }}",
        "LocalCredential": {
            "Moid": "{{ local_credential.api_response.Moid }}"
        },
        "SysConfig": {
            "Moid": "{{ sys_config.api_response.Moid }}"
        },
        "ClusterNetwork": {
            "Moid": "{{ cluster_network.api_response.Moid }}"
        },
        "VcenterConfig": {
            "Moid": "{{ vcenter.api_response.Moid }}"
        },
        "ClusterStorage": {
            "Moid": "{{ storage_setting.api_response.Moid }}"
        },
        "NodeConfig": {
            "Moid": "{{ node_config.api_response.Moid }}"
        }
    }

```

```

    "ProxySetting":{
        "Moid":"{{ proxy_setting.api_response.Moid }}"
    }
}
when:
- local_credential.api_response is defined
- sys_config.api_response is defined
- cluster_network.api_response is defined
- vcenter.api_response is defined
- node_config.api_response is defined
- proxy_setting.api_response is defined
- storage_setting.api_response is defined

```

The 2nd task shown above updates the cluster profile with desired policies when return variables have been registered sys_config (dns) and other policies.

Step 4: View the Inventory and host_vars

For this lab, think of a HyperFlex Cluster Profile as a “host” you are configuring. The inventory specifies the names of HyperFlex Cluster Profiles and there is a corresponding host_vars file for each host.

View the host_vars/sjc07-r13-501 file to see host specific variables used in configuration:

```

---
mgmt_ip: 172.28.224.171
hx_storage_vlan: 501
hx_mgmt_ip_start: 172.28.224.165
hx_mgmt_ip_end: 172.28.224.167
hx_dp_ip_start: 172.28.224.168
hx_dp_ip_end: 172.28.224.170
hx_servers:
- C220-WZP2204097Z
- C220-WZP2204090C
- C220-WZP220400B3

```

Host specific variables such as the mgmt_ip and hx_storage_vlan are used when configuring cluster/host specific policies and the profile.

Step 5: Run Complete Playbook

Run the full playbook:

```

$ ansible-playbook -i inventory hyperflex_cluster_profiles.yml

PLAY [Intersight_HX]
*****

TASK [policies/hyperflex_policies/dns_ntp_timezone : Configure
devnet-dns-ntp DNS, NTP, and Timezone Policy] ***
ok: [sjc07-r13-501]

```

```
TASK [policies/hyperflex_policies/security : Configure devnet-
creds Security Policy] ***
ok: [sjc07-r13-501]
```

...

```
TASK [policies/hyperflex_policies/node_ip_hostname : Configure
sjc07-r13-501-node-config Node IP and Hostname Policy] ***
ok: [sjc07-r13-501]
ok: [sjc07-r13-503]
```

```
TASK [service_profiles/hyperflex_cluster_profiles : Configure
sjc07-r13-501 HyperFlex Cluster Profile] ***
ok: [sjc07-r13-501]
ok: [sjc07-r13-503]
```

```
TASK [service_profiles/hyperflex_cluster_profiles : Configure
HyperFlex versions] ***
ok: [sjc07-r13-503]
ok: [sjc07-r13-501]
```

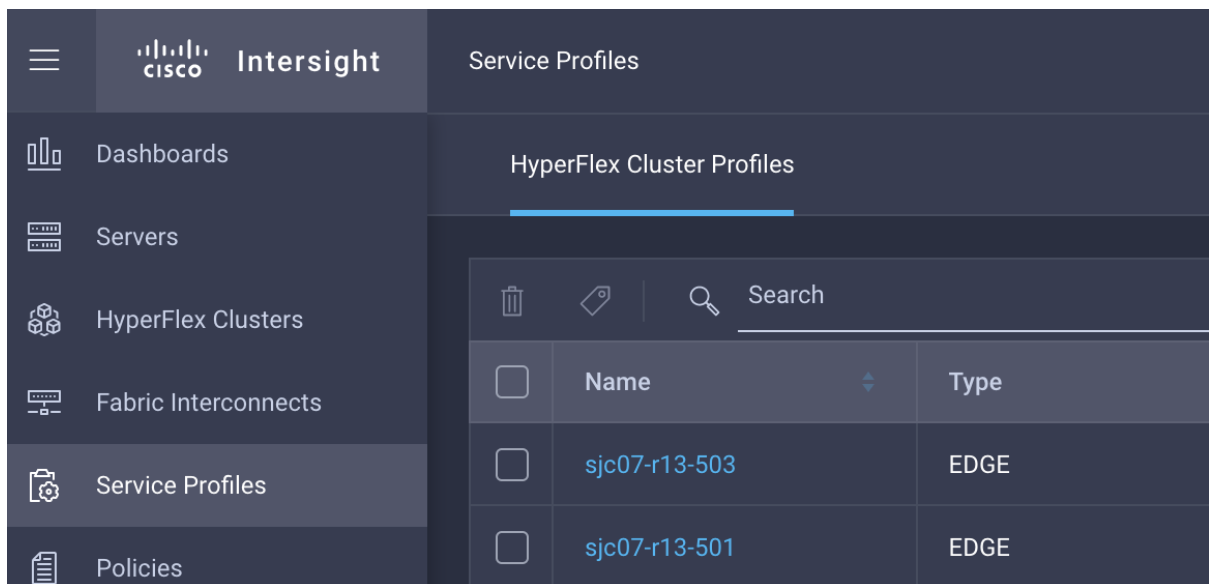
```
TASK [service_profiles/hyperflex_cluster_profiles : Policies
for sjc07-r13-501 HyperFlex Cluster Profile] ***
ok: [sjc07-r13-503]
ok: [sjc07-r13-501]
```

PLAY RECAP

```
*****
*****
sjc07-r13-501          : ok=10    changed=0
unreachable=0         failed=0    skipped=0    rescued=0
ignored=0
sjc07-r13-503          : ok=4     changed=0
unreachable=0         failed=0    skipped=0    rescued=0
ignored=0
```

Ask the lab instructor for help with any failures or other issues.

Once you have a successful run, you should see multiple cluster profiles configured from the single playbook:



Congratulations on completing the Intersight and Ansible DevNet workshop!

