

CISCO *Live!*

DevNet Zone

ALL
IN

#CiscoLiveAPJC



The bridge to possible

ClickOps to GitOps

Cisco DNA Center Infrastructure-as-Code Use Cases

Gabi Zapodeanu,
Technical Marketing Engineer
@zapodeanu
DEVNET-2739

GitHub

github.com/cisco-en-programmability



<https://YouTube.com/c/CiscoENProgrammability>

CISCO *Live!*

DevNet Zone

#CiscoLiveAPJC

Cisco Webex App

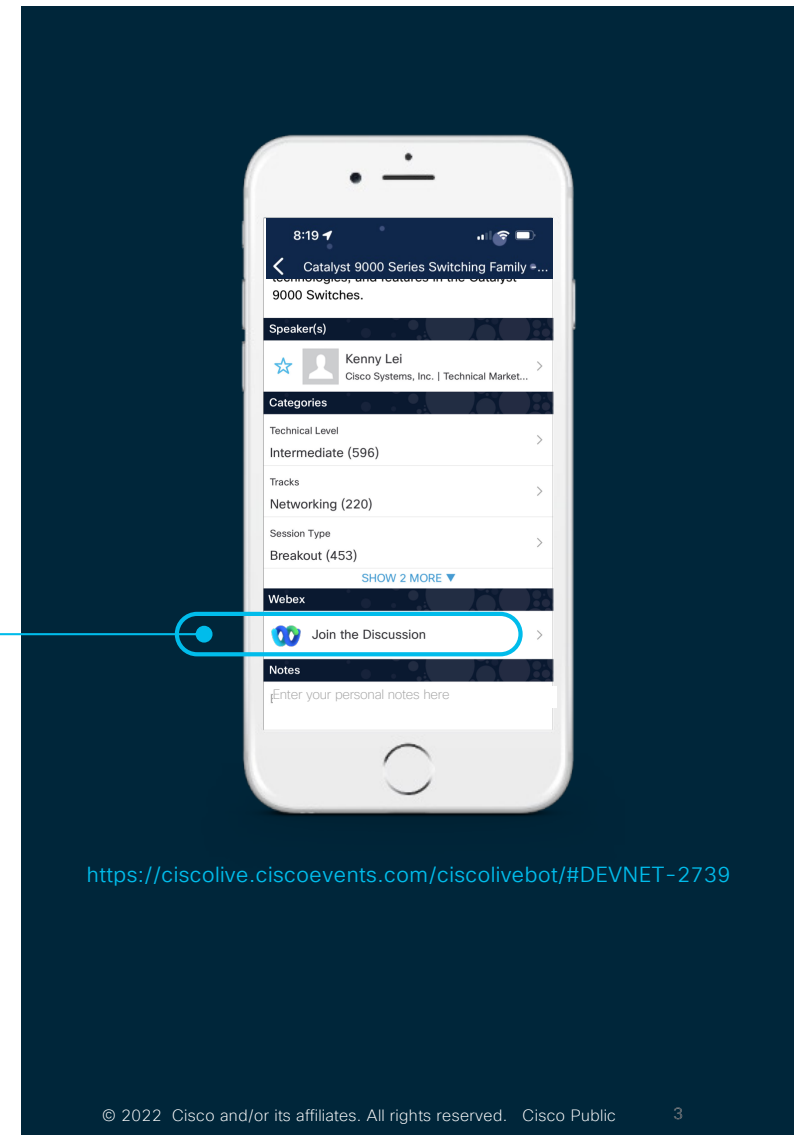
Questions?

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 Thursday 22 December, 2022.





Agenda

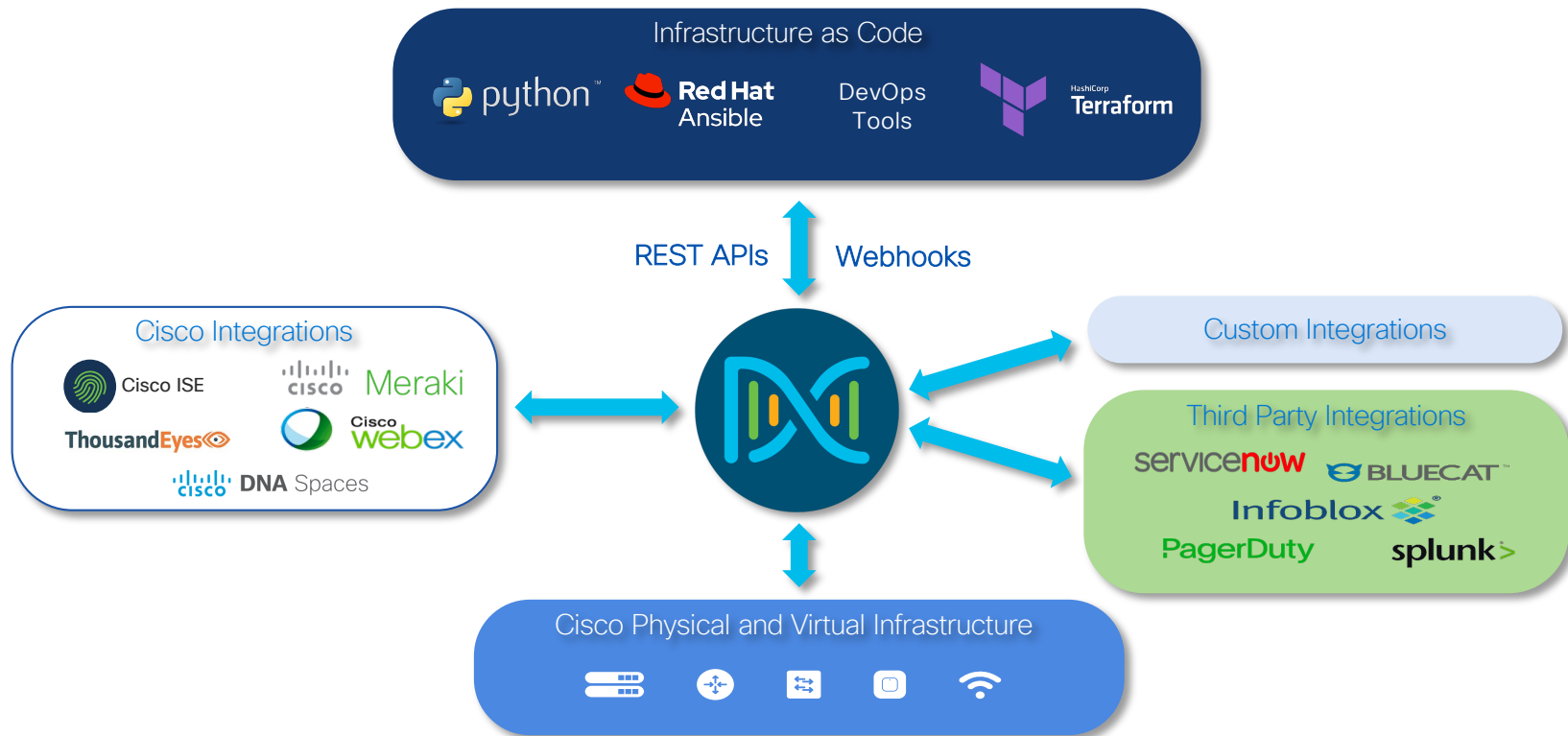
- Cisco DNA Center Platform
- Inventory Collection – Python SDK
- Device Configuration – Ansible
- Software Upgrades – Terraform
- Developer Resources
- Lessons Learned



Agenda

- Cisco DNA Center Platform
- Inventory Collection – Python SDK
- Device Configuration – Ansible
- Software Upgrades – Terraform
- Developer Resources
- Lessons Learned

Cisco DNA Center Platform



Cisco DNA Center Platform Overview

Event Notifications

- Assurance Issues
- AI/ML Insights
- System Health
- Integration Connectivity
- License Management
- Webhooks
- PagerDuty
- Webex
- Syslog
- SNMP

Northbound REST APIs

- Network Inventory
- Network Topology
- Network Design
- Provisioning
- SWIM, PnP
- Path Trace
- Assurance
- SDA
- Templates
- RMA
- Config Archive
- Sensors

IT Ecosystem Integrations

- IT Service Management
- IP Address Management
- Reporting
- Wireless Planning
- Incident Response
- SIEM, Splunk

Developer Resources

- Sample Code, Videos
- Python SDK, Ansible, Terraform
- Cisco DevNet
 - Sandboxes, Learning Labs
 - Developer Guides
 - Sample Code



Cisco DNA Center

Platform / Overview

Welcome to the Cisco DNA Center Platform. Programmatically access your network through Intent APIs, integrate with your preferred IT systems to create end-to-end solutions and add support for multi-vendor devices.

Bundles

Bundles are easy to use feature sets for consuming Intent APIs, integrations, events and notifications. View all the available bundles, enable relevant bundles and customize the configuration preferences to consume events as per your application(s) or IT system(s) needs.

Developer Toolkit

Discover APIs to manage your network, configure integration flows and access network data to analyze, export and visualize complex reports.

Runtime Dashboard

Get insights into API usage, view events published to IT systems such as number of API calls, response time(s), events published, bundles activated etc.

Configurations

View and set global or bundle specific settings to manage your integration configurations and modify event specific settings.

Notifications

Find

BUNDLE UPDATE

Basic ITSM (ServiceNow) CMDB synchronization successfully configured.

Apr 06 2022, 10:07 pm

[View Details](#) | [Dismiss](#)

BUNDLE UPDATE

The Cisco DNA Center Automation events for ITSM (ServiceNow) bundle is enabled and ready for configuration.

Apr 06 2022, 10:06 pm

[View Details](#) | [Dismiss](#)

BUNDLE UPDATE

Basic ITSM (ServiceNow) CMDB synchronization successfully configured.

Mar 30 2022, 03:19 pm

[View Details](#) | [Dismiss](#)

DNA Center Platform – Developer Toolkit

Cisco DNA Center

Platform / Developer Toolkit

APIsIntegration FlowsMultivendor SupportEvent Notifications

Search

Authentication

Know Your Network

Sites

Topology

Devices

Clients

Sensors

Users

Applications

Issues

Security Advisories

Compliance

Site Management

Connectivity

Search API

Devices

Method	Name	Description	URL	Actions
GET	Get Chassis Details for Device	Returns chassis details for given device ID	/network-device/\${deviceId}/chassis	
GET	Get Module Info by Id	Returns Module info by 'module id'	/network-device/module/\${id}	
DELETE	Delete Device by Id	Deletes the network device for the given Id	/network-device/\${id}	
GET	Return PowerSupply/ Fan details for the Given device	Return PowerSupply/ Fan details for the Given device	/network-device/\${deviceUuid}/equipment	
GET	Get Device list	Returns list of network devices based on filter criteria such as management IP address, mac address, hostname, etc. You can use the .* in any value to conduct...	/network-device	
GET	Get Device Interface VLANs	Returns Device Interface VLANs	/network-device/\${id}/vlan	

Try

Call an API without writing any code

Try

Try an API Call

Cisco DNA Center Get Device list

Try "Get Device list"

Method: **GET** Public URL :https://10.93.141.45/dna/intent/api/v1/network-device

PARAMETERS

QUERY PARAMETERS

hostname ⓘ
NYC-ACCESS ⓘ

managementIpAddress ⓘ

macAddress ⓘ

locationName ⓘ

serialNumber ⓘ

Reset Run

Response Headers Status Code: 200

```
1 {
2   "response": [
3     {
4       "role": "ACCESS",
5       "apManagerInterfaceIp": "",
6       "bootDateTime": "2021-09-09 16:55:54",
7       "collectionStatus": "Managed",
8       "family": "Switches and Hubs",
9       "hostname": "NYC-ACCESS",
10      "lastUpdateTime": "1650324294617",
11      "locationName": null,
12      "managementIpAddress": "10.93.141.26",
13      "platformId": "C9300-24UX",
14      "reachabilityFailureReason": "",
15      "reachabilityStatus": "Reachable",
16      "series": "Cisco Catalyst 9300 Series Switches",
17      "snmpContact": "",
18      "snmpLocation": "",
19      "softwareType": "IOS-XE",
20      "softwareVersion": "17.3.2a",
21      "serialNumber": "FJC2325T0SL",
22      "deviceSupportLevel": "Supported",
23      "upTime": "221 days, 6:29:02.05",
24      "inventoryStatusDetail": "<status><general code=\\\"SUCCESS\\\"/></status>",
25      "lastUpdated": "2022-04-18 23:24:54",
26      "collectionInterval": "Global Default",
```

Code Preview

Cisco DNA Center

APIs Integration Flows Multi...

Search

Authentication

Know Your Network

Sites

Topology

Devices

Clients

Sensors

Users

Applications

Issues

Security Advisories

Compliance

Site Management

Site Design

Network Settings

Software Image Management (SWIM)

Get Device list

GET `https://10.93.141.45/dna/intent/api/v1/network-device`

Returns list of network devices based on filter criteria such as management IP address, mac address, hostname, etc. You can use the .* in any value to conduct a wildcard search. For example, to find all hostnames beginning with myhost in the IP address range 192.25.18.n, issue the following request: GET /dna/intent/api/v1/network-device?hostname=myhost.*&managementIpAddress=192.25.18.n.* If id parameter is provided with comma separated ids, it will return the list of network-devices for the given ids and ignores the other request parameters. You can also specify offset & limit to get the required list.

[Cisco DevNet API Guide](#)

Parameters Responses **Code Preview**

Language: Python

```
1 // token-value'
2 // tion/json'
3
4
5 co Node - HTTP connection("10.93.141.45")
6
7 co Node - Unirest intent/api/v1/network-device?hostname=<hostname>&managementIpAddress=<managementIpAddress>&macAddress=<macAddress>&locationName
8
9 res = conn.getResponse()
```

Close Try

Code Preview
Generate code in few programming languages

CISCO *Live!*

DevNet Zone

#CiscoLiveAPJC DEVNET-2739

© 2022 Cisco and/or its affiliates. All rights reserved. Cisco Public

11

GitOps for Cisco DNA Center?

```
project_name: Ansible_Project
template_name: csr_ntp_snmp
filter: device_family
filter_value: Cisco Cloud Services Router 1000V
cli_template: |
!
no snmp-server host 10.93.135.30 version 2c R0
no snmp-server host 10.93.130.50 version 2c RW
!
ntp server 171.68.48.78
no ntp server 171.68.48.80

project_name: Ansible_Project
template_name: access_switches_logging
filter: role
filter_value: ACCESS
cli_template: |
!
logging host 10.93.141.37
logging host 10.93.141.37 transport udp port 8514
!
service timestamps debug datetime localtime
service timestamps log datetime localtime
```

- GitOps uses Git repositories as a single source of truth to deliver Infrastructure-as-Code:
 - Automations and integrations
 - Intent-based configurations documented as code
- **Code changes, intent configurations, and network state are tracked, and version controlled**
- Consistency across Cisco DNA Center clusters and IT teams
- Enhanced security and visibility

Cisco DNA Center GitOps

Inventory Collection Python SDK:

- Collect and parse the device inventory from Cisco DNA Center
- Create these files:
Devices and AP lists, non-compliant devices
- Files saved in JSON, YAML formats
- Pushed to GitHub and saved locally

Device Configuration Ansible Playbook:

- Git pull, or clone, the repos with device inventory and deployment template
- Apply filter to select devices to be configured
- Identify if devices are managed and reachable
- Deploys CLI Templates, checks status
- Creates deployment status report

Software Upgrades Terraform Plan:

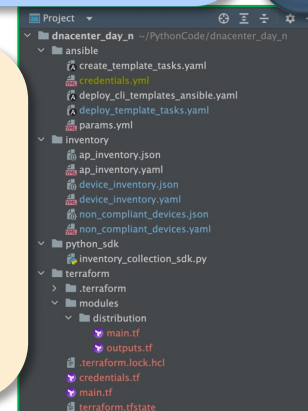
- Identify the devices that are software image non-compliant
- Automate the image distribution to network devices
- Terraform provider, plans and device inventory pulled from GitHub

- Always up-to-date inventory and Apps code, hosted on GitHub
- Network engineers run the same automations written using Python SDK, Ansible and Terraform
- Consistent outcomes, validated and secure

GitHub

Network Engineer + DevOps create automation workflows:

- Device inventory
 - Ansible playbooks
 - Terraform plans
 - Intent configurations
- All workflows are pushed to GitHub



CISCO *Live!*

DevNet Zone

#CiscoLiveAPJC DEVNET-2739

© 2022 Cisco and/or its affiliates. All rights reserved. Cisco Public

13



Agenda

- Cisco DNA Center Platform
- Inventory Collection – Python SDK
- Device Configuration – Ansible
- Software Upgrades – Terraform
- Developer Resources
- Lessons Learned

Automation Workflow: Inventory Collection

Inventory Collection

```
dnacenter_day_n > python_sdk > inventory_collection_sdk.py
inventory_collection_sdk.py x
177
178 # create device inventory [{"hostname": "", "device_ip": "", "device_id": "", "version": "", "device_family": "",
179 # "role": "", "site": "", "site_id": ""},...]
180 device_inventory = []
181 ap_inventory = []
182 for device in device_list:
183     # select which inventory to add the device to
184     if device.family != "Unified AP":
185         device_details = {'hostname': device['hostname']}
186         device_details.update({'device_ip': device['managementIpAddress']})
187         device_details.update({'device_id': device['id']})
188         device_details.update({'version': device['softwareVersion']})
189         device_details.update({'device_family': device['type']})
190         device_details.update({'role': device['role']})
191
192     # get the device site hierarchy
193     response = dnac_api.devices.get_device_detail(identifier='uuid', search_by=device['id'])
194     site = response['response'][0]['location']
195     device_details.update({'site': site})
196
197     # get the site id
198     response = dnac_api.sites.get_site(name=site)
199     site_id = response['response'][0]['id']
200     device_details.update({'site_id': site_id})
201     device_inventory.append(device_details)
202 else:
203     device_details = {'hostname': device['hostname']}
204     device_details.update({'device_ip': device['managementIpAddress']})
205     device_details.update({'device_id': device['id']})
206     device_details.update({'version': device['softwareVersion']})
207     device_details.update({'device_family': device['type']})
208     device_details.update({'role': device['role']})
```

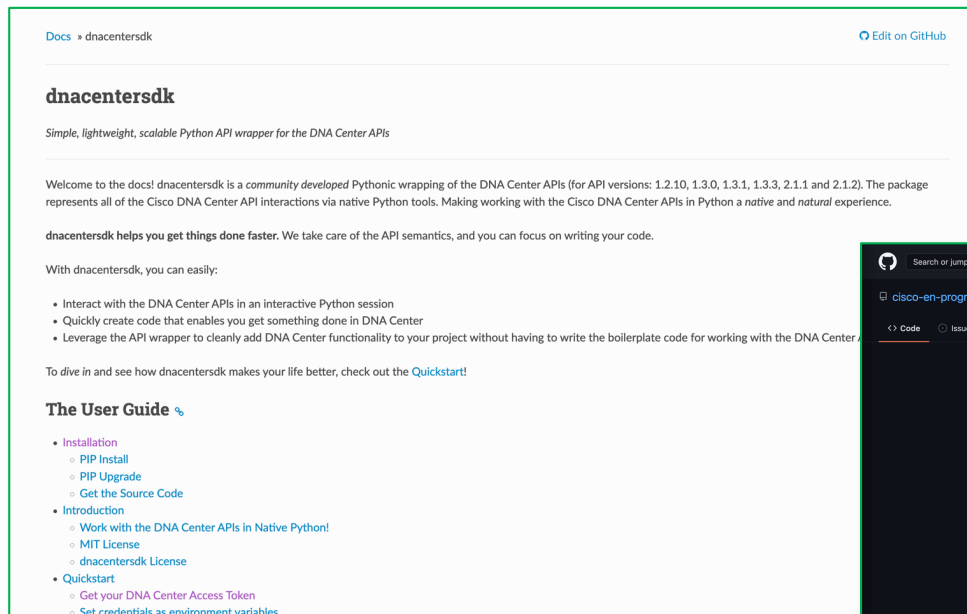
Inventory Files

```
dnacenter_day_n > inventory > device_inventory.yaml
device_inventory.yaml x
1 device_inventory:
2   - hostname: C9800-CL
3     device_ip: 10.93.141.38
4     device_id: e5facea9-6897-45b7-9c1f-1c8c7224e812
5     version: 17.4.1
6     device_family: Cisco Catalyst 9800-CL Wireless Controller for Cloud
7     role: ACCESS
8     site: Global/OR/PDX/Floor-2
9     site_id: 17d5b2b9-ff29-4320-bdab-fd6c3d7df09d
10  - hostname: L0-BN
11    device_ip: 10.93.141.28
12    device_id: db05ad49-2c8d-4094-9df6-3a03f9b28069
13    version: 17.6.1
14    device_family: Cisco Catalyst 9300 Switch
15    role: ACCESS
16    site: Global/OR/L0/Floor-3
17    site_id: 3bb9e022-b230-40f2-a17e-f588376f31e3
18  - hostname: L0-PN
```

- May run on-demand, or scheduled
- Will collect rich device information and topology
- Inventory files:
 - Pushed to GitHub
 - Saved locally
 - Formatted: JSON, YAML, CSV, ...
- Always up-to-date inventory

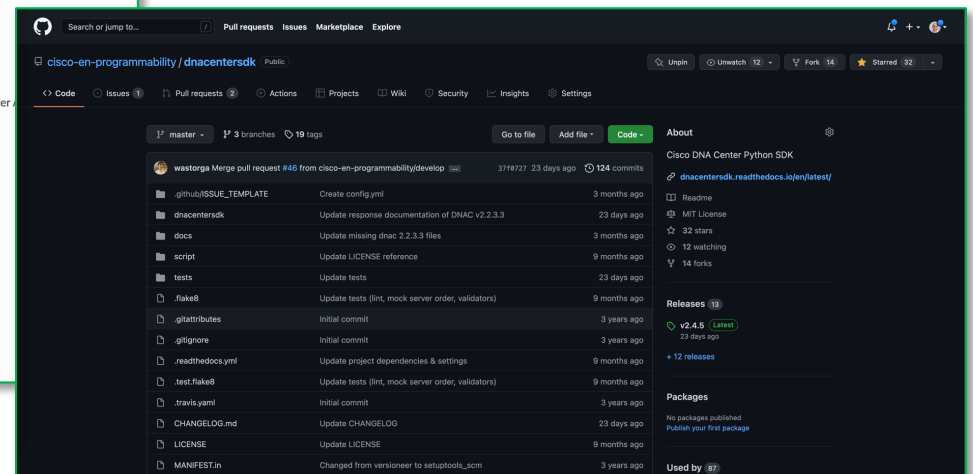
Cisco DNA Center Python SDK

Cisco DNA Center Version Support 1.2.10 to 2.3.3.x



The screenshot shows the documentation for the **dnacentersdk** package. It includes a header with 'Docs' and 'dnacentersdk', a link to 'Edit on GitHub', and a description: 'Simple, lightweight, scalable Python API wrapper for the DNA Center APIs'. A welcome message states that the package is a community-developed Pythonic wrapping of the DNA Center APIs for versions 1.2.10, 1.3.0, 1.3.1, 1.3.3, 2.1.1, and 2.1.2. It mentions that the package represents all Cisco DNA Center API interactions via native Python tools. A section titled 'With dnacentersdk, you can easily:' lists three bullet points: interacting with APIs in an interactive Python session, quickly creating code for DNA Center functionality, and leveraging the API wrapper to avoid boilerplate code. A 'Quickstart' link is provided. Below this is 'The User Guide' with links for Installation (PIP Install, PIP Upgrade, Get the Source Code), Introduction (Work with the DNA Center APIs in Native Python!, MIT License, dnacentersdk License), and Quickstart (Get your DNA Center Access Token, Set credentials as environment variables).

- Includes all Cisco DNA Center REST APIs
- Support for Cisco DNA Center version 2.3.3.x
- First SDK version August 2019
- Average 1,500+ downloads/month



The screenshot shows the GitHub repository for **cisco-en-programmability/dnacentersdk**. It displays the repository name, a search bar, and navigation tabs for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The repository has 3 branches and 19 tags. A table of recent commits is shown, including a merge pull request #46 from cisco-en-programmability/develop. The right sidebar shows repository statistics: 124 commits, 3718727 lines of code, 23 days ago, 12 forks, 32 stars, and 12 watching. It also shows the latest release, v2.4.5, and a section for packages.

<https://dnacentersdk.readthedocs.io>

<https://github.com/cisco-en-programmability/dnacentersdk>

Python SDK Installation and Upgrade

Search or jump to... Pull requests Issues Marketplace Explore

cisco-en-programmability / dnacentersdk Public

Edit Pins Unwatch 12 Fork 15 Starred 34

<> Code Issues 1 Pull requests 2 Actions Projects Wiki Security Insights Settings

master 3 branches 22 tags Go to file Add file Code

fmunozmiranda Merge pull request #53 from cisco-en-programmability/develop a490531 5 days ago 146 commits

README.rst

Introduction

Installation

Installing and upgrading dnacentersdk is easy:

Install via PIP

```
$ pip install dnacentersdk
```

Upgrading to the latest Version

```
$ pip install dnacentersdk --upgrade
```

About

Cisco DNA Center Python SDK

dnacentersdk.readthedocs.io/en/latest/

Readme MIT License 34 stars 12 watching 15 forks

Releases 17

v2.4.9 Latest 5 days ago

+ 16 releases

Packages

No packages published Publish your first package

Calling Cisco DNA Center REST APIs

```
55 def get_dnac_jwt_token(dnac_auth):
56     """
57     Create the authorization token required to access DNA C
58     Call to Cisco DNA Center - /api/system/v1/auth/login
59     :param dnac_auth - Cisco DNA Center Basic Auth string
60     :return: Cisco DNA Center token
61     """
62     url = DNAC_URL + '/dna/system/api/v1/auth/token'
63     header = {'content-type': 'application/json'}
64     response = requests.post(url, auth=dnac_auth, headers=header, verify=False)
65     pprint(response.json())
66     dnac_jwt_token = response.json()['Token']
67     return dnac_jwt_token
68
69 def get_all_device_info(limit, dnac_jwt_token):
70     """
71     The function will return all network devices info, using the specified limit of devices/API Call
72     :param limit: the number of devices to return per API call
73     :param dnac_jwt_token: Cisco DNA Center token
74     :return: DNA C device inventory info
75     """
76     offset = 1
77     all_devices_list = []
78     all_devices_info = [''] # assign a value, to make sure the API call will run at least once
79     while all_devices_info:
80         all_devices_info = ''
81         url = DNAC_URL + '/dna/intent/api/v1/network-device?offset=' + str(offset) + '&limit=' + str(limit)
82         header = {'content-type': 'application/json', 'x-auth-token': dnac_jwt_token}
83         all_devices_response = requests.get(url, headers=header, verify=False)
84         all_devices_json = all_devices_response.json()
85         all_devices_info = all_devices_json['response']
86         all_devices_list += all_devices_info
87         offset += limit
88     return all_devices_list
89
```

Obtain the auth token

Get the device list using the
API pagination

Calling Cisco DNA Center REST APIs – Python SDK

- Accelerates and simplifies apps development
- Developer may focus on the use case, and not each API call

```
89
90 # create a DNACenterAPI "Connection Object" to use the Python SDK
91 dnac_api = DNACenterAPI(username=DNAC_USER, password=DNAC_PASS, base_url=DNAC_URL, version='2.3.3.0',
92                          verify=False)
93
94 # get the device count
95 response = dnac_api.devices.get_device_count()
96 device_count = response['response']
97 logging.info('Number of devices managed by Cisco DNA Center: ' + str(device_count))
98
99 # get the device info list
100 offset = 1
101 limit = 500
102 device_list = []
103 while offset <= device_count:
104     response = dnac_api.devices.get_device_list(offset=offset)
105     offset += limit
106     device_list.extend(response['response'])
107 logging.info('Collected the device list from Cisco DNA Center')
108
```

Specify Cisco DNA Center

Get the device count

Get the device list using
API pagination

How to Use Integrated Development Environment (IDE)

```
86 # get the device count
87 response = dnac_api.devices.get_device_count()
88 device_count = response['response']
89 logging.info('Number of devices managed by Cisco DNA Center: '
90
91 # get the device info list
92 offset = 1
93 limit = 500
94 device_list = []
95 while offset <= device_count:
96     response = dnac_api.devices.get_device_count()
97     device_count = response['response']
98     logging.info('Number of devices managed by Cisco DNA Center: ' + str(device_count))
99
100 # get the device info list
101 offset = 1
102 limit = 500
103 device_list = []
104 while offset <= device_count:
105     response = dnac_api.devices.get_device_list(offset=offset, limit=limit)
106     offset += limit
107     device_list.extend(response['response'])
108     logging.info('Collected the device List from Cisco DNA Center')
```

```
82 # create a DNACenterAPI "Connection Object" to use the Python SDK
83 dnac_api = DNACenterAPI(username=DNAC_USER, password=DNAC_PASS, base_url=DNAC_URL, version='2.2.3.3',
84                         verify=False)
85
86 self.Devices, associated_wic_ip=None, collection_interval=None,
87 collection_status=None, device_support_level: str = None,
88 error_code=None, error_description=None, family=None,
89 hostname=None, id: str = None, license_name=None,
90 license_status=None, license_type=None, location=None,
91 location_name=None, mac_address=None, management_ip_address=None,
92 module_equipmenttype=None, module_name=None, module_operationstatecode=None,
93 module_partnumber=None, module_servicestate=None,
94 module_vendorequipmenttype=None, not_synced_for_minutes=None,
95 platform_id=None, reachability_status=None, role=None,
96 serial_number=None, series=None, software_type=None,
97 software_version=None, type=None, up_time=None, headers: dict = None,
98 device_list=None, **request_parameters
99 while offset <= device_count:
100     response = dnac_api.devices.get_device_count()
101     device_count = response['response']
102     logging.info('Number of devices managed by Cisco DNA Center: ' + str(device_count))
103
104 # get the device info list
105 offset = 1
106 limit = 500
107 device_list = []
108 while offset <= device_count:
109     response = dnac_api.devices.get_device_list(offset=offset, limit=limit)
110     offset += limit
111     device_list.extend(response['response'])
112     logging.info('Collected the device List from Cisco DNA Center')
```

- get_device_count
- get_device_list
- get_device_count
- devices(self, device_id)
- add_device(self, device_id, device_name, device_type, device_model, device_ip, device_mac, device_location, device_tags)
- sync_devices_usage(self, device_id)
- sync_devices_status(self, device_id)
- get_network_device(self, device_id)
- get_network_device_by_ip(self, ip_address)
- get_device_interface_count(self, headers, request_parameters)
- get_interface_by_ip(self, ip_address, headers, request_parameters)
- delete_device_by_id(self, device_id, headers, request_parameters)



Inventory Collection

```
83
84 # create a DNACenterAPI "Connection Object" to use the Python SDK
85 dnac_api = DNACenterAPI(username=DNAC_USER, password=DNAC_PASS, base_url=DNAC_URL, version='2.2.3.3',
86                          verify=False)
87
88 # get the device count
89 response = dnac_api.devices.get_device_count()
90 device_count = response['response']
91 logging.info('Number of devices managed by Cisco DNA Center: ' + str(device_count))
92
93 # get the device info list
94 offset = 1
95 limit = 500
96 device_list = []
97 while offset <= device_count:
98     logging.info('Collected the device list from Cisco DNA Center')
99     # create device inventory [{"hostname": "", "device_ip": "", "device_id": "", "version": "", "device_family": "",
100     # "role": "", "site": "", "site_id": ""},...]
101     device_inventory = []
102     for device in device_list:
103
104         logging.info('Collected the device inventory')
105
106     # save inventory to file
107     with open('device_inventory.json', 'w') as f:
108         f.write(json.dumps(device_inventory))
109     logging.info('Saved the device inventory to file "device_inventory.json"')
110
111 # retrieve the device image compliance state
112 image_non_compliant_devices = []
113 response = dnac_api.compliance.get_compliance_detail(compliance_type='IMAGE', compliance_status='NON_COMPLIANT')
114 image_non_compliant_list = response['response']
115 for device in image_non_compliant_list:
116     logging.info('Number of devices image non-compliant: ' + str(len(image_non_compliant_devices)))
117     logging.info('Image non-compliant devices: ')
118     for device in image_non_compliant_devices:
119         logging.info(' ' + device['hostname'] + ' , Site Hierarchy: ' + device['site'])
120
121 # save non compliant devices to file
122 with open('non_compliant_devices.json', 'w') as f:
123     f.write(json.dumps(image_non_compliant_devices))
124 logging.info('Saved the image non-compliant device inventory to file "non_compliant_devices.json"')
```

YAML and JSON formatted files
saved locally and pushed to GitHub

Specify the Cisco DNA Center

Get the device count

Get the device list

Parse the device information

Save device inventory to file

Identify image non-compliant devices

Parse the non-compliant devices information

Save non-compliant devices to file

Sample Inventory Files

Note:

- Data format - JSON, YAML, ...
- Files pushed to GitHub and/or saved locally

```
12- {
13-   "hostname": "C9800-CL",
14-   "device_ip": "10.93.141.38",
15-   "device_id": "e5facea9-6097-45b7-9c1f-1c8c7224e812",
16-   "version": "17.4.1",
17-   "device_family": "Cisco Catalyst 9800-CL Wireless Controller
    for Cloud",
18-   "role": "ACCESS",
19-   "site": "Global/OR/PDX/Floor-2",
20-   "site_id": "17d5b2b9-ff29-4320-bdab-fd6c3d7df09d"
21- },
22- {
23-   "hostname": "L0-BN",
24-   "device_ip": "10.93.141.28",
25-   "device_id": "db05ad49-2c8d-4094-9df6-3a03f9b28069",
26-   "version": "17.6.1",
27-   "device_family": "Cisco Catalyst 9300 Switch",
28-   "role": "ACCESS",
29-   "site": "Global/OR/L0/Floor-3",
30-   "site_id": "3bb9e022-b230-40f2-a17e-f588376f31e3"
31- },
32- {
33-   "hostname": "L0-CN",
34-   "device_ip": "10.93.141.20",
35-   "device_id": "6a612b05-1c89-4d0a-a4d3-94ec5e5a7a57",
36-   "version": "17.6.1",
37-   "device_family": "Cisco Catalyst 9300 Switch",
38-   "role": "ACCESS",
39-   "site": "Global/OR/L0/Floor-3",
40-   "site_id": "3bb9e022-b230-40f2-a17e-f588376f31e3"
41- },
```

"device_inventory.json"

```
1- non_compliant:
2- - hostname: L0-BN
3-   device_ip: 10.93.141.28
4-   device_id: db05ad49-2c8d-4094-9df6-3a03f9b28069
5-   version: 17.6.1
6-   device_family: Cisco Catalyst 9300 Switch
7-   role: ACCESS
8-   site: Global/OR/L0/Floor-3
9-   site_id: 3bb9e022-b230-40f2-a17e-f588376f31e3
10- - hostname: NYC-ACCESS
11-   device_ip: 10.93.141.26
12-   device_id: 20937bce-b8d8-42c2-b3a0-08cda93fae1e
13-   version: 17.3.2a
14-   device_family: Cisco Catalyst 9300 Switch
15-   role: ACCESS
16-   site: Global/NY/NYC/Floor-8
17-   site_id: 755df6b0-6bdb-4d17-8983-906645e6bf98
18- - hostname: L0-CN
19-   device_ip: 10.93.141.20
20-   device_id: 6a612b05-1c89-4d0a-a4d3-94ec5e5a7a57
21-   version: 17.6.1
22-   device_family: Cisco Catalyst 9300 Switch
23-   role: ACCESS
24-   site: Global/OR/L0/Floor-3
25-   site_id: 3bb9e022-b230-40f2-a17e-f588376f31e3
```

"non_compliant_devices.yaml"



Demo Inventory Collection



FirefoxFileEditViewHistoryBookmarksToolsWindowHelp

zapodeanu/dnacenter_day_n...Home - Cisco DNA Center

https://10.93.141.45/dna/home110%

Cisco Systems, IncAPIsToolsLAB_AccessCiscoLiveMeraki Dashboard L...Home - AsanaEssentials of Machi...Data & AnalyticsCisco BoxSFEH-NAS - Synolo...CDETSDNACaaPCisco.DNA.Center...SharePointOther Bookmarks

Cisco DNA Center

SearchHelpFeedback

Assurance Summary

Health ⓘ

Healthy as of Jun 8, 2022 11:32 AM

100%

Network Devices

--%

Wireless Clients

100%

Wired Clients

View Details

Critical Issues

Last 24 Hours

0

P1

2

P2

View Details

Trends and Insights

0

AP Performance Advisories

0

Trend Deviations

View Details

Network Snapshot

Sites

As of Jun 8, 2022 11:35 AM

13

DNS Servers : 1

NTP Servers : 1

Add Sites

Network Devices

As of Jun 8, 2022 11:35 AM

13

Unclaimed: 2

Unprovisioned: 1

Unreachable: 0

Find New Devices

Application QoS Policies

As of Jun 8, 2022 11:35 AM

0

Successful Deploys: 0

Errored Deploys: 0

Stale Policies: 0

Add New Policy

Network Profiles

As of Jun 8, 2022 11:35 AM

Images

As of Jun 8, 2022 11:35 AM

Software Image Upgrades

As of Jun 8, 2022 11:35 AM

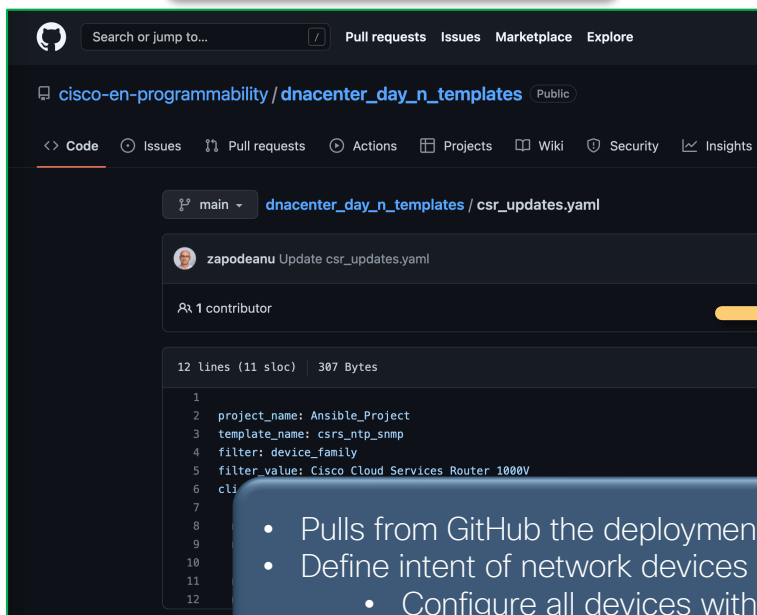


Agenda

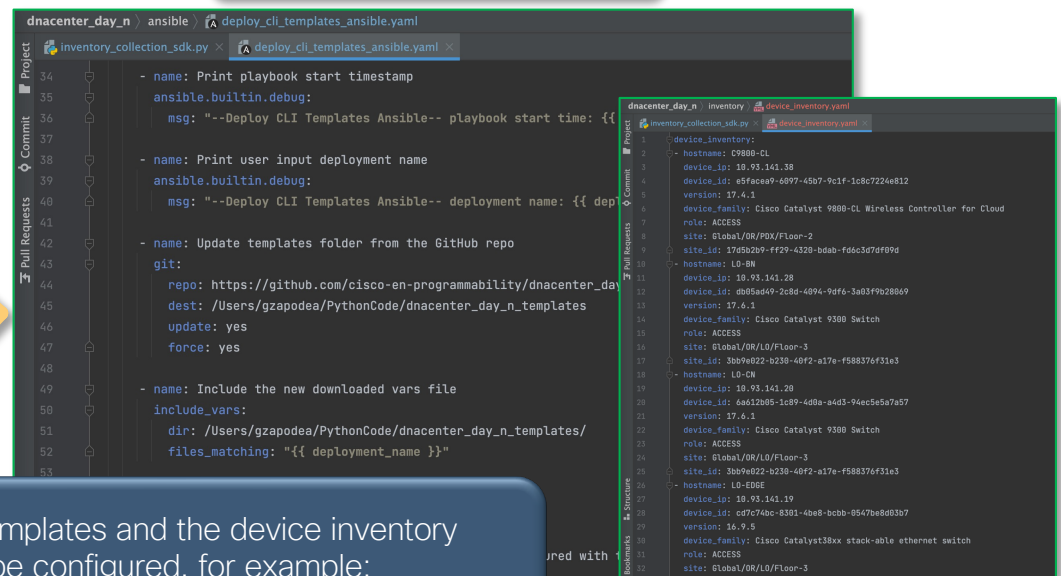
- Cisco DNA Center Platform
- Inventory Collection – Python SDK
- Device Configuration – Ansible
- Software Upgrades – Terraform
- Developer Resources
- Lessons Learned

Automation Workflow: Device Configuration

Template GitHub Repo



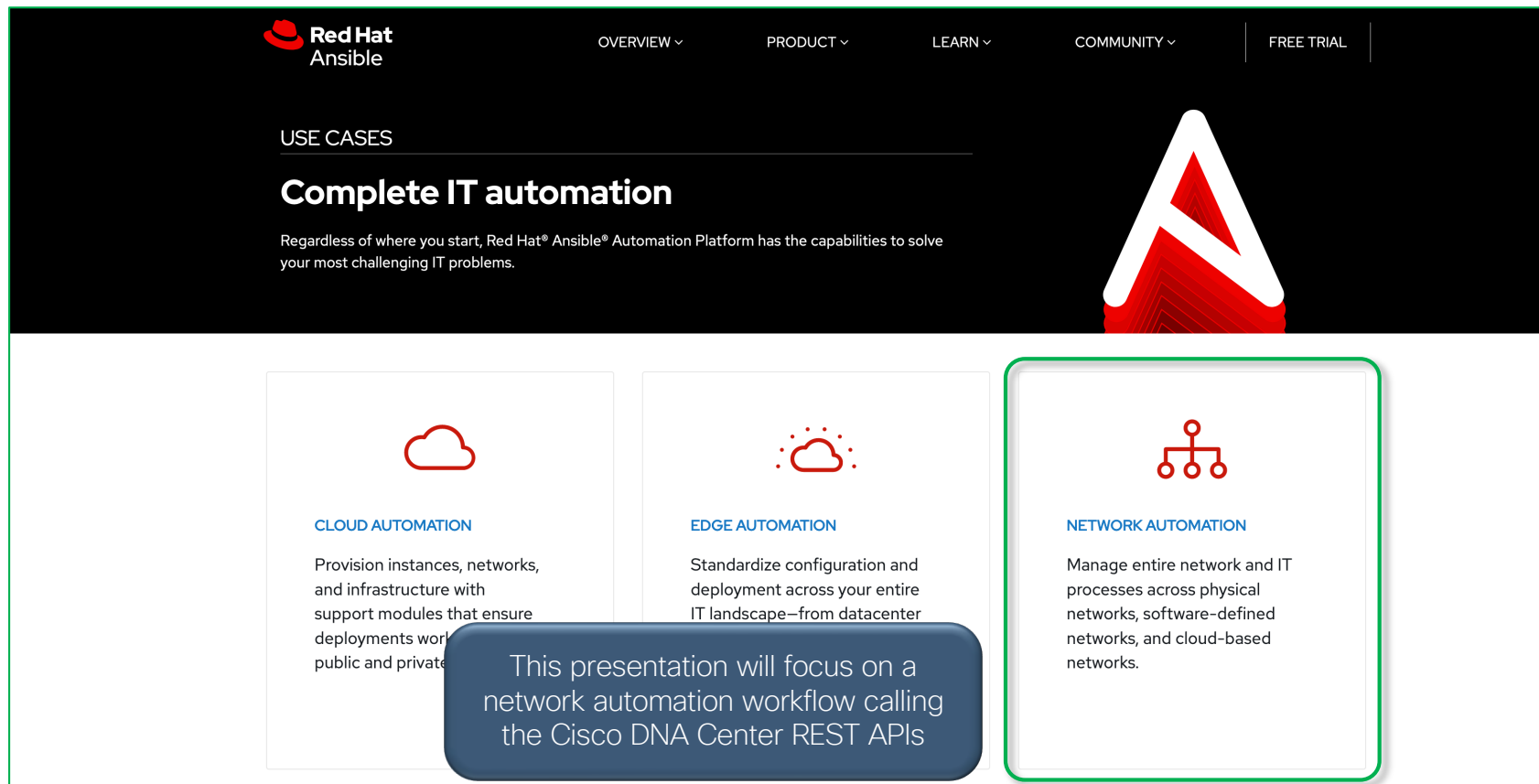
Ansible Playbook



- Pulls from GitHub the deployment templates and the device inventory
- Define intent of network devices to be configured, for example:
 - Configure all devices with role “ACCESS” at a specific location
- Automate the deployment of CLI templates to network devices
- May run on-demand, or scheduled

Requires Device Inventory

Ansible Automation Use Cases



Red Hat Ansible

OVERVIEW ▾ PRODUCT ▾ LEARN ▾ COMMUNITY ▾ FREE TRIAL

USE CASES

Complete IT automation

Regardless of where you start, Red Hat® Ansible® Automation Platform has the capabilities to solve your most challenging IT problems.



CLOUD AUTOMATION

Provision instances, networks, and infrastructure with support modules that ensure deployments work in public and private clouds.



EDGE AUTOMATION

Standardize configuration and deployment across your entire IT landscape—from datacenter to edge.



NETWORK AUTOMATION

Manage entire network and IT processes across physical networks, software-defined networks, and cloud-based networks.

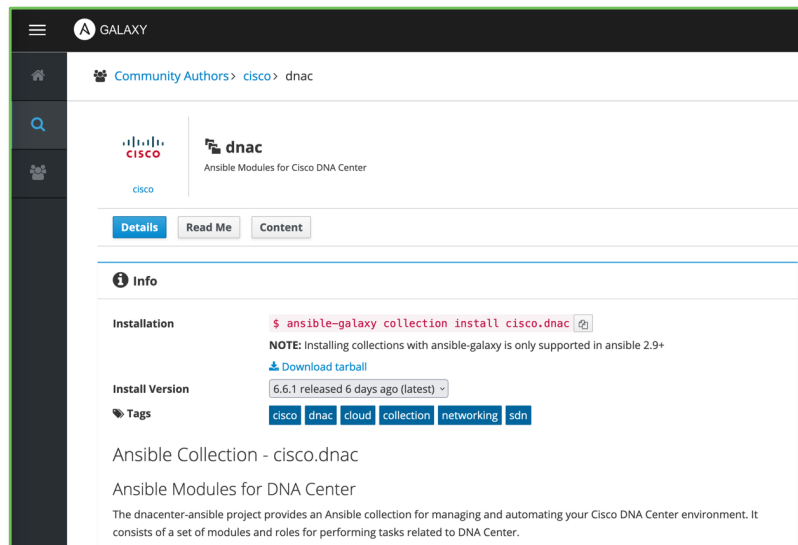
This presentation will focus on a network automation workflow calling the Cisco DNA Center REST APIs

Ansible One Slide

- **YAML** – Yet Another Markup Language or YAML Ain't Markup Language. Ansible uses YAML to define playbook configurations and variable files. It is human readable and may be used with many programming languages.
- **Collection** – A packaging format for bundling and distributing Ansible content, including plugins, roles, modules, and more.
- **Module/Plugin** – Code, typically written in Python, that will perform some action on a host. Cisco DNA Center modules provide the documentation, the plugins execute the actions
- **Playbook** – Repeatable, re-usable, simple configuration management that will push a new configuration or confirm the existing configuration. They are composed of plays and tasks.
- **Play** – Execution of a set of tasks to a host or group of hosts – lab Cisco DNA Center vs production Cisco DNA Centers
- **Task** – Execute a module with specific arguments. When a task has executed on all target machines, Ansible moves on to the next task.
- **Task Lists and Blocks** – re-usable groups of tasks that are executed based on specific conditions or counts

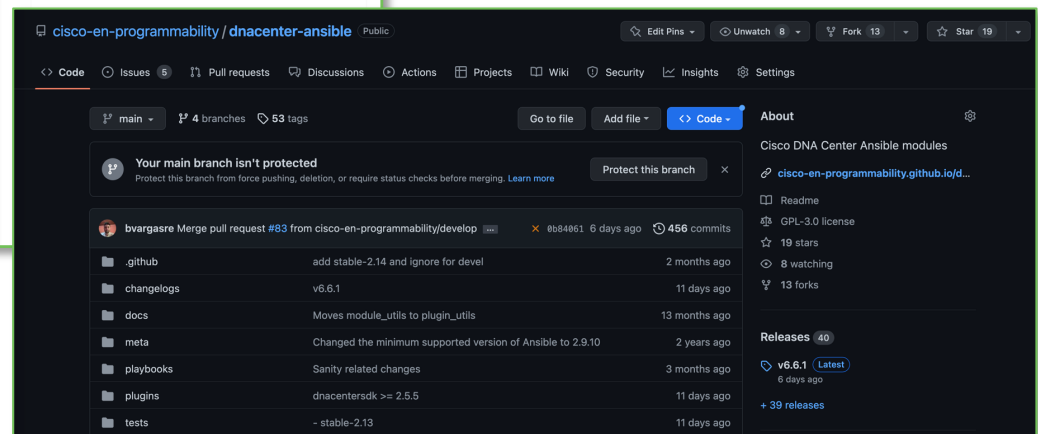
Ref: <https://docs.ansible.com>

Cisco DNA Center Ansible Collection



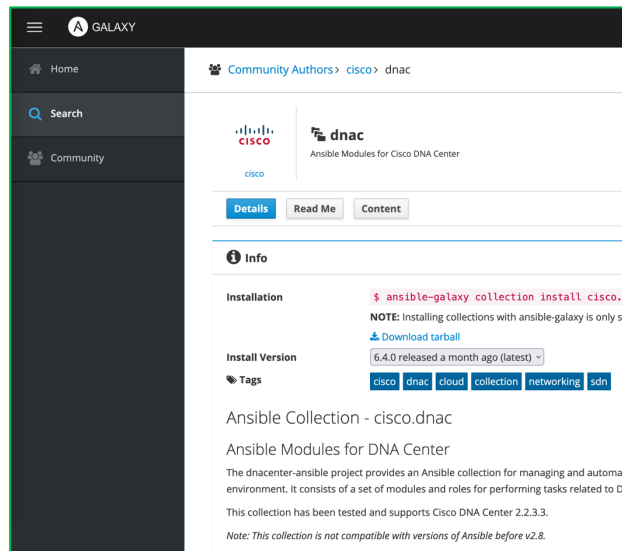
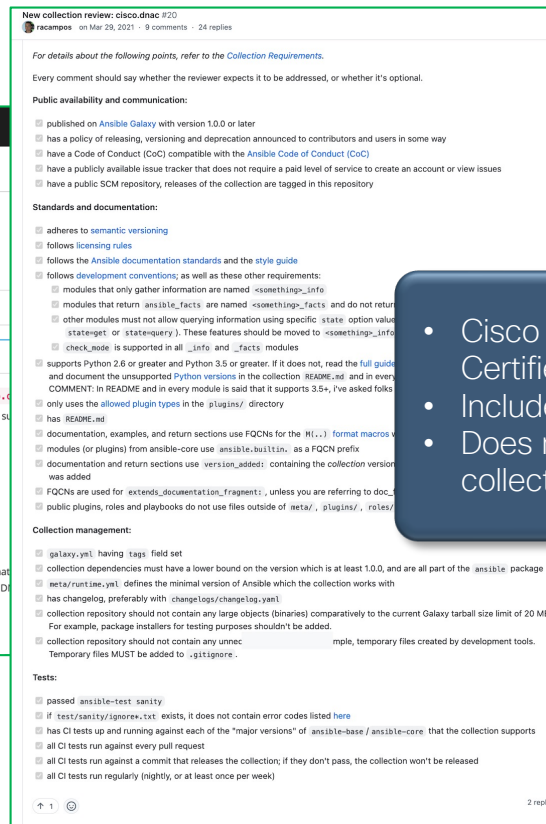
<https://galaxy.ansible.com/cisco/dnac>

- Complete set of Ansible modules for all Cisco DNA Center REST APIs
- First library version Dec 2020
- Certified by RedHat v 2.9 – 04/2021
- Community certified – 05/2022
- Support for Cisco DNA Center version 2.3.3.x
- 7,300 + downloads



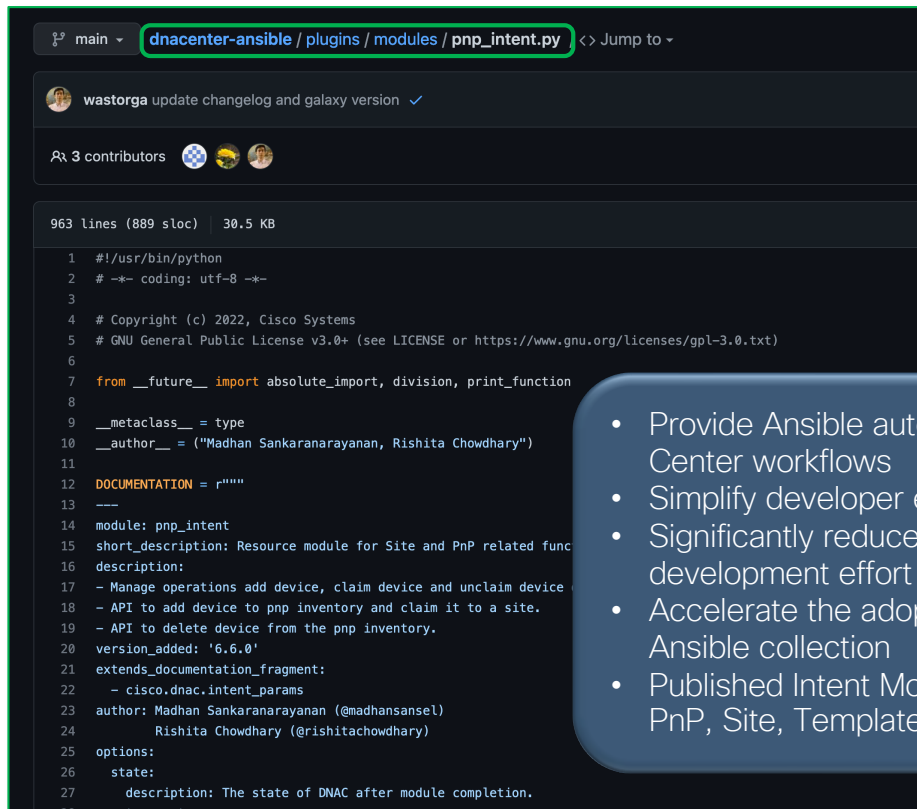
<https://github.com/cisco-en-programmability/dnacenter-ansible>

Ansible Collection – Community Certification

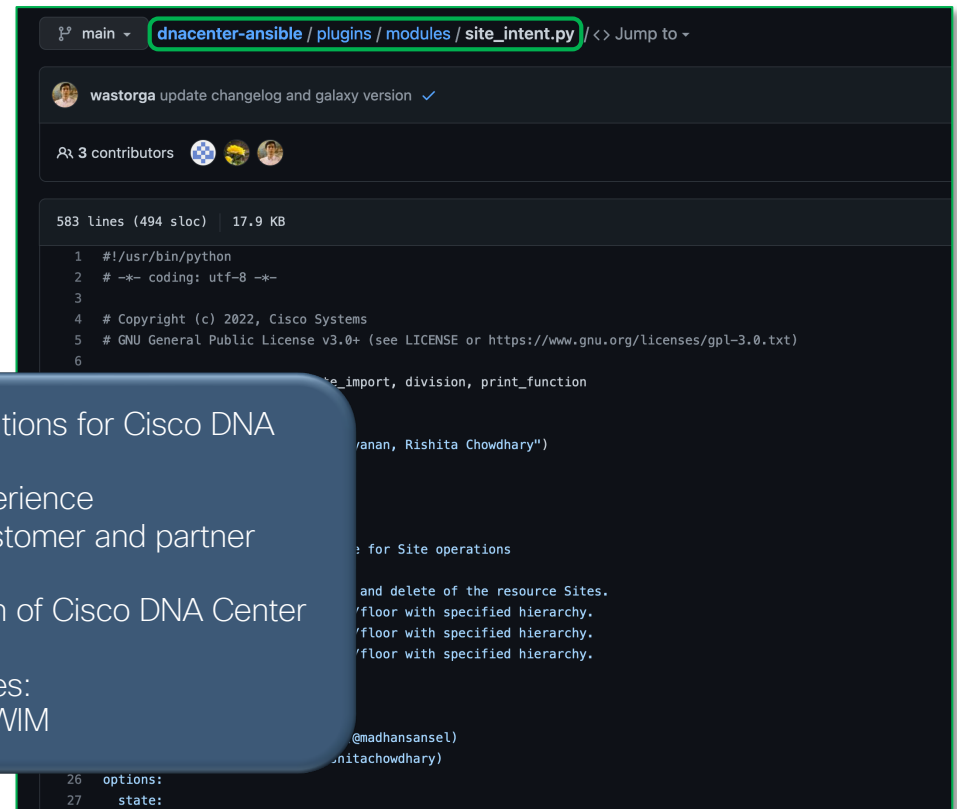


- Cisco DNA Center Collection Community Certified – 05/19/2022
- Included with every Ansible installation
- Does not require additional Cisco DNA Center collection installation steps

Intent Modules – Cisco Developed and Supported



The screenshot shows the GitHub repository for `dnacenter-ansible`, specifically the `plugins/modules/pnp_intent.py` file. The file is 963 lines long (889 sloc) and 30.5 KB. It is a Python module for Ansible, licensed under the GNU General Public License v3.0+. The module is named `pnp_intent` and is described as a resource module for Site and PnP related functions. It includes a `DOCUMENTATION` section with a description and a list of options. The module is authored by Madhan Sankaranarayanan and Rishita Chowdhary.



The screenshot shows the GitHub repository for `dnacenter-ansible`, specifically the `plugins/modules/site_intent.py` file. The file is 583 lines long (494 sloc) and 17.9 KB. It is a Python module for Ansible, licensed under the GNU General Public License v3.0+. The module is named `site_intent` and is described as a resource module for Site operations. It includes a `DOCUMENTATION` section with a description and a list of options. The module is authored by Madhan Sankaranarayanan and Rishita Chowdhary.

- Provide Ansible automations for Cisco DNA Center workflows
- Simplify developer experience
- Significantly reduce customer and partner development effort
- Accelerate the adoption of Cisco DNA Center Ansible collection
- Published Intent Modules: PnP, Site, Template, SWIM

Using the Ansible Modules

<https://galaxy.ansible.com/cisco/dnac>

The screenshot displays the Ansible Galaxy interface for the 'cisco.dnac' collection. The left sidebar shows the collection's details, including the Cisco logo, 'dnac' title, and 'Ansible Modules for Cisco DNA Center'. The main content area is titled 'Installing according to Compatibility Matrix' and provides instructions for installing the collection and its dependencies. It includes code snippets for installing 'dnacentersdk' and the 'cisco.dnac' collection using 'ansible-galaxy collection install'. A 'Requirements' section lists the necessary versions of Ansible, Python, and the DNA Center SDK. The 'Install' section provides commands for installing Ansible and the collection. The bottom of the page notes that the collection has been tested and supports Cisco DNA Center 2.3.3.0.

Community Authors > cisco > dnac

Details Read Me Content

Info

Installation

Install Version 6.6.1 r

Tags cisco

Ansible Collection - cisco.dnac

Ansible Modules for DNA Center

The dnacenter-ansible project provides a set of modules and roles for managing Cisco DNA Center.

This collection has been tested and supports Cisco DNA Center 2.3.3.0.

Installing according to Compatibility Matrix

For example, for Cisco DNA Center 2.2.2.3, it is recommended to use Ansible "cisco.dnac" v3.3.1 and Python "dnacentersdk" v2.3.3.

To get the Python DNA Center SDK v2.3.3 in a fresh development environment:

```
sudo pip install dnacentersdk==2.3.3
```

To get the Ansible collection v3.3.1 in a fresh development environment:

```
ansible-galaxy collection install cisco.dnac:3.3.1
```

Requirements

- Ansible >= 2.9
- Python DNA Center SDK v2.4.7 or newer
- Python >= 3.6, as the DNA Center SDK doesn't support Python version 2.x

Install

Ansible must be installed ([Install guide](#))

```
sudo pip install ansible
```

Python DNA Center SDK must be installed

```
sudo pip install dnacentersdk
```

Install the collection ([Galaxy link](#))

```
ansible-galaxy collection install cisco.dnac
```

Cisco DNA Center Ansible Collection Installation

Ansible Installation

Terminal: Local x +

(venv_temp) gzapodea@GZAPODEA-M-67G6 dnacenter_day_n % pip install ansible

Collecting ansible

Downloading ansible-5.6.0.tar.gz (35.5 MB)

35.5/35.5 MB 5.3 MB/s eta 0:00:00

Preparing metadata (setup.py) ... done

Collecting ansible-core==2.12.4

Downloading ansible-core-2.12.5.tar.gz (7.8 MB)

7.8/7.8 MB 5.3 MB/s eta 0:00:00

Preparing metadata (setup.py) ... done

Collecting jinja2

Using cached Jinja2-3.1.1-py3-none-any.whl (132 kB)

Collecting PyYAML

Using cached PyYAML-6.0-cp39-cp39-macosx_10_9_x86_64.whl (197 kB)

Collecting cryptography

Using cached cryptography-36.0.2-cp36-abi3-macosx_10_10_x86_64.whl (2.5 MB)

Collecting packaging

Using cached packaging-21.3-py3-none-any.whl (40 kB)

Collecting resolvelib<0.6.0,>=0.5.3

Using cached resolvelib-0.5.4-py2.py3-none-any.whl (12 kB)

Collecting cffi>=1.12

Using cached cffi-1.15.0-cp39-cp39-macosx_10_9_x86_64.whl (178 kB)

Collecting MarkupSafe>=2.0

Using cached MarkupSafe-2.1.1-cp39-cp39-macosx_10_9_x86_64.whl (13 kB)

Collecting pyparsing!=3.0.5,>=2.0.2

Downloading pyparsing-3.0.8-py3-none-any.whl (98 kB)

98.5/98.5 KB 3.6 MB/s eta 0:00:00

Collecting pycparser

Using cached pycparser-2.21-py2.py3-none-any.whl (118 kB)

Using legacy 'setup.py install' for ansible, since package 'wheel' is not installed.

Using legacy 'setup.py install' for ansible-core, since package 'wheel' is not installed.

Installing collected packages: resolvelib, PyYAML, pyparsing, pycparser, MarkupSafe, packaging, jinja2, cffi, cryptography, ansible-core, ansible

Install Ansible



Cisco DNA Center Ansible Collection Installation

Python SDK Installation

```
Terminal: Local x +
(venv_temp) gzapodea@GZAP0DEA-M-6766 dnacenter_day_n % pip install dnacentersdk
Collecting dnacentersdk
  Using cached dnacentersdk-2.4.9-py3-none-any.whl (2.4 MB)
Collecting fastjsonschema<3.0.0,>=2.14.5
  Using cached fastjsonschema-2.15.3-py3-none-any.whl (22 kB)
Collecting requests<3.0.0,>=2.25.1
  Using cached requests-2.27.1-py2.py3-none-any.whl (63 kB)
Collecting requests-toolbelt<0.10.0,>=0.9.1
  Using cached requests_toolbelt-0.9.1-py2.py3-none-any.whl (54 kB)
Collecting future<0.19.0,>=0.18.2
  Using cached future-0.18.2.tar.gz (829 kB)
  Preparing metadata (setup.py) ... done
Collecting urllib3<1.27,>=1.21.1
  Using cached urllib3-1.26.9-py2.py3-none-any.whl (138 kB)
Collecting charset-normalizer~=2.0.0
  Using cached charset-normalizer-2.0.12-py3-none-any.whl (39 kB)
Collecting idna<4,>=2.5
  Using cached idna-3.3-py3-none-any.whl (61 kB)
Collecting certifi>=2017.4.17
  Using cached certifi-2021.10.8-py2.py3-none-any.whl (149 kB)
Using legacy 'setup.py install' for future, since package 'wheel' is not installed.
Installing collected packages: fastjsonschema, certifi, urllib3, idna, future, charset-normalizer, requests, requests-toolbelt, dnacentersdk
  Running setup.py install for future ... done
Successfully installed certifi-2021.10.8 charset-normalizer-2.0.12 dnacentersdk-2.4.9 fastjsonschema-2.15.3 future-0.18.2 idna-3.3 requests-2.27.1 requests-toolbelt-0.9.1 urllib3-1.26.9
(venv_temp) gzapodea@GZAP0DEA-M-6766 dnacenter_day_n %
```

Install the Python SDK

Note: Instructions to get started with Cisco DNA Center Ansible modules
<https://github.com/cisco-en-programmability/dnacenter-ansible>

Cisco DNA Center Ansible Collection Installation

Cisco DNA Center Ansible Collection Installation

```
Terminal: Local x +
(venv_temp) gzapodea@GZAP0DEA-M-6766 dnacenter_day_n % ansible-galaxy collection install cisco.dnac --force
Starting galaxy collection install process
Process install dependency map
Starting collection install process
Downloading https://galaxy.ansible.com/download/cisco-dnac-6.4.0.tar.gz to /Users/gzapodea/.ansible/tmp/ansible-local-90046olztvg0q/tmpghgmXlr1u/cisco-dnac-6.4.0-nya15s_x
Installing 'cisco.dnac:6.4.0' to '/Users/gzapodea/.ansible/collections/ansible_collections/cisco/dnac'
cisco.dnac:6.4.0 was installed successfully
'ansible.utils:2.4.3' is already installed, skipping.
(venv_temp) gzapodea@GZAP0DEA-M-6766 dnacenter_day_n % ansible-galaxy collection list

# /Users/gzapodea/PythonCode/dnacenter_day_n/venv_temp/lib/python3.9/site-packages/ansible_collections
Collection      Version
-----
amazon.aws      2.2.0
ansible.netcommon 2.6.1
ansible.posix    1.3.0
ansible.utils    2.5.2
ansible.windows  1.9.0

vyos.vyos        2.8.0
wti.remote        1.0.3

# /Users/gzapodea/.ansible/collections/ansible_collections
Collection      Version
-----
ansible.utils    2.4.3
cisco.dnac       6.4.0
(venv_temp) gzapodea@GZAP0DEA-M-6766 dnacenter_day_n %
```

Install, or update, the Cisco DNA Center Collection

Verify the collection is installed

Modules version

Device List API + Ansible Module

<https://github.com/cisco-en-programmability/dnacenter-ansible/tree/main/plugins/modules>

The screenshot shows the Cisco DNA Center web interface. On the left is a navigation menu with categories like APIs, Integration Flows, and Multitenancy. The main content area is titled 'Get Device list'. It shows a GET request to the endpoint `https://10.93.141.45/dna/intent/api/v1/network-device`. Below this, there's a description of the API and a table of request query parameters.

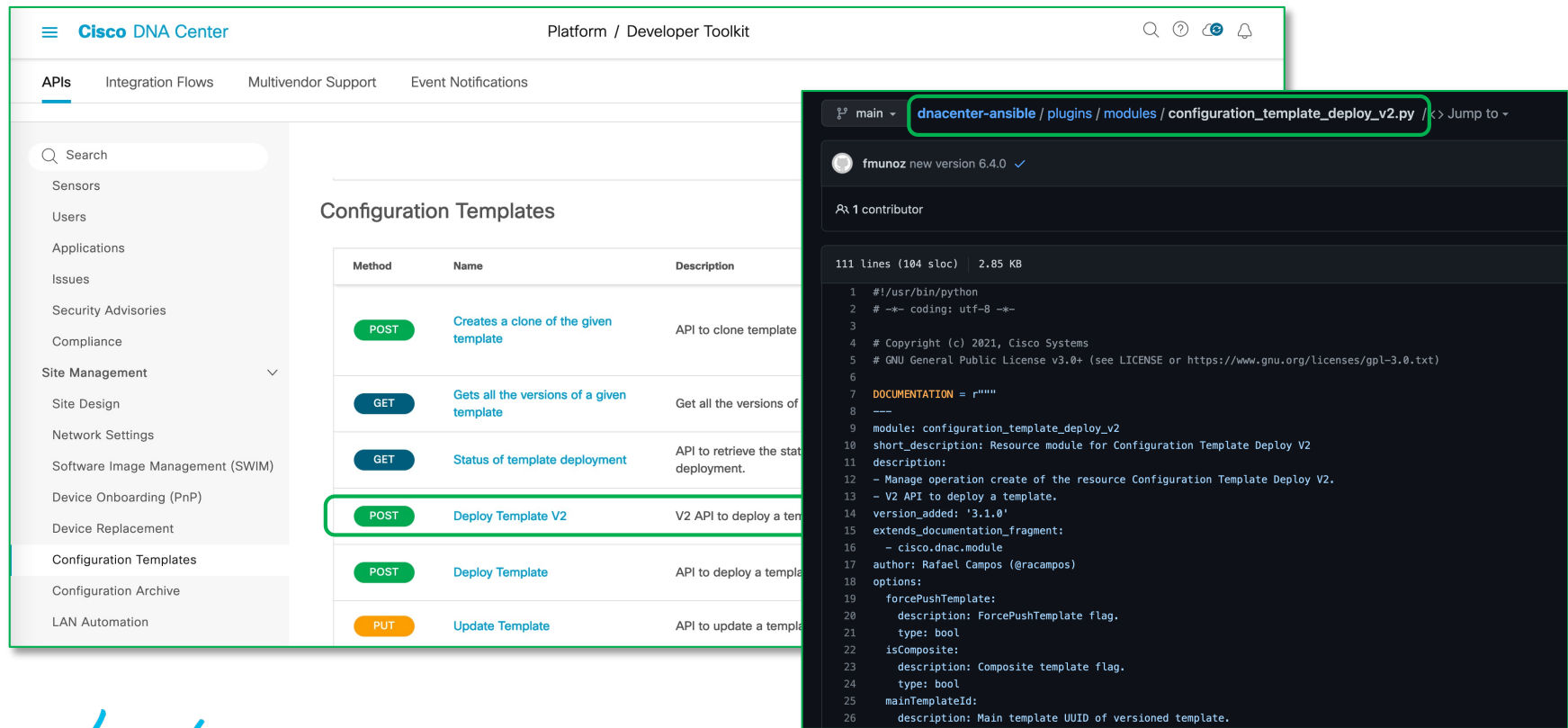
Name	Description
hostname	hostname
managementIpAddress	managementIpAddress
macAddress	macAddress
locationName	locationName
serialNumber	serialNumber
location	location
family	family

The screenshot shows the GitHub repository for `dnacenter-ansible`, specifically the `network_device_info.py` module. The file is 310 lines long and 7.33 KB. It includes a contributor `fmunoz` and a description of the module. A callout box highlights the `deviceSupportLevel` parameter and the API endpoints used.

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3
4  # Copyright (c) 2021, Cisco Systems
5  # GNU General Public License v3.0+ (see LICENSE or https://www.gnu.org/licenses/gpl-3.0.txt)
6
7  DOCUMENTATION = r"""
8  ---
9  module: network_device_info
10 short_description: Information module for Network Device
11 description:
12   - Get all Network Device.
13   - Get Network Device by id.
14   - Returns list of network devices based on filter.
15   - Returns the network device details for the given id.
16 version_added: '3.1.0'
17 extends_documentation_fragment:
18   - cisco.dnac.module_info
19 author: Rafael Campos (@racampos)
20 options:
21   headers:
22     description: Additional headers.
23     type: dict
24   hostname:
25     description:
26       - Hostname query parameter.
27     elements: str
28
29   deviceSupportLevel:
30     description:
31       - DeviceSupportLevel query parameter.
32     type: str
33   requirements:
34     - dnacentersdk >= 2.4.9
35     - python >= 3.5
36   notes:
37     - SDK Method used are
38       devices.Devices.get_device_by_id,
39       devices.Devices.get_device_list,
40     - Paths used are
41       get /dna/intent/api/v1/network-device,
42       get /dna/intent/api/v1/network-device/{id},
```


Deploy Template API + Ansible Module

<https://github.com/cisco-en-programmability/dnacenter-ansible/tree/main/plugins/modules>



The image shows two overlapping screenshots. The background screenshot is the Cisco DNA Center web interface, specifically the 'Configuration Templates' section. The foreground screenshot is a GitHub repository view for the 'dnacenter-ansible' project, showing the 'configuration_template_deploy_v2.py' file.

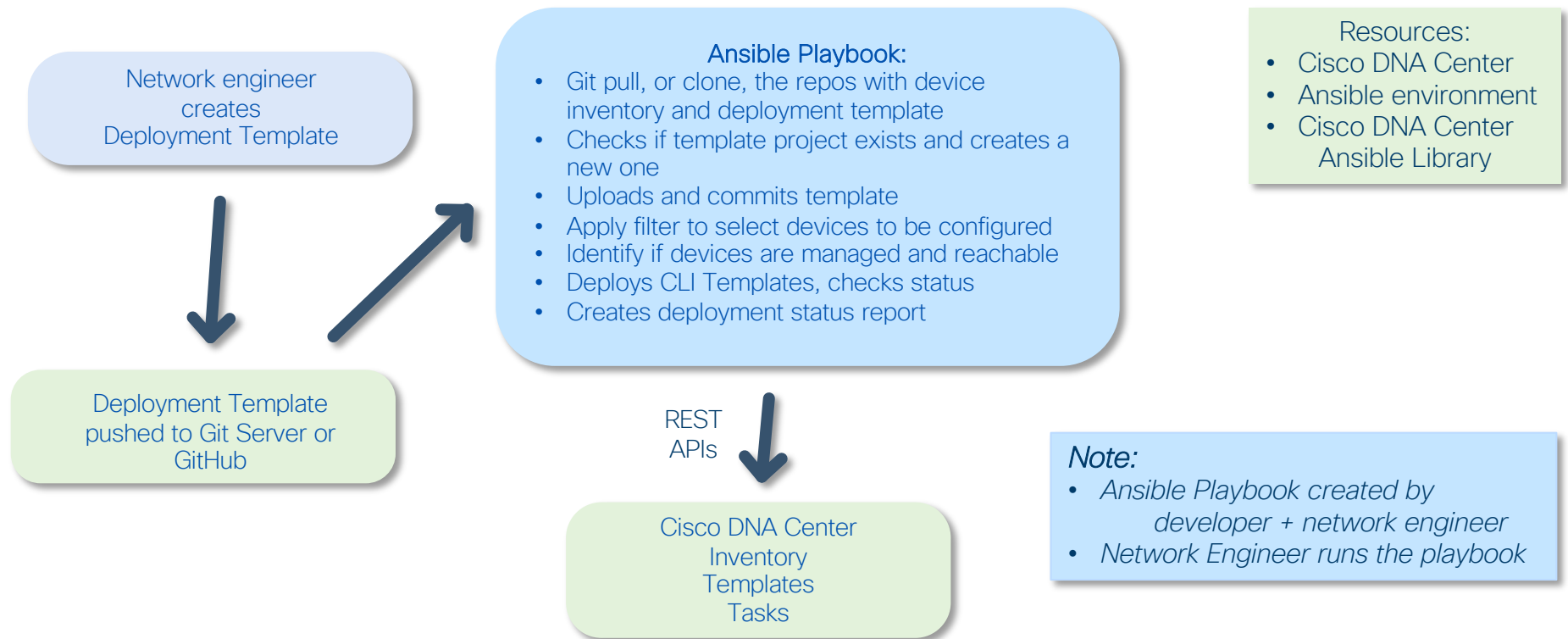
Cisco DNA Center Configuration Templates Table:

Method	Name	Description
POST	Creates a clone of the given template	API to clone template
GET	Gets all the versions of a given template	Get all the versions of
GET	Status of template deployment	API to retrieve the stat deployment.
POST	Deploy Template V2	V2 API to deploy a ten
POST	Deploy Template	API to deploy a templa
PUT	Update Template	API to update a templa

dnacenter-ansible / plugins / modules / configuration_template_deploy_v2.py

```
1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3
4 # Copyright (c) 2021, Cisco Systems
5 # GNU General Public License v3.0+ (see LICENSE or https://www.gnu.org/licenses/gpl-3.0.txt)
6
7 DOCUMENTATION = """
8 ---
9 module: configuration_template_deploy_v2
10 short_description: Resource module for Configuration Template Deploy V2
11 description:
12 - Manage operation create of the resource Configuration Template Deploy V2.
13 - V2 API to deploy a template.
14 version_added: '3.1.0'
15 extends_documentation_fragment:
16 - cisco.dnac.module
17 author: Rafael Campos (@racampos)
18 options:
19   forcePushTemplate:
20     description: ForcePushTemplate flag.
21     type: bool
22   isComposite:
23     description: Composite template flag.
24     type: bool
25   mainTemplateId:
26     description: Main template UUID of versioned template.
```

Template Deployment Ansible Playbook



Deployment Templates

Note:
Network Engineer created

```
dnacenter_day_n_templates > csr_updates.yaml
1 project_name: Ansible_Project
2 template_name: csrs_ntp_snmp
3
4 filter: device_family
5 filter_value: Cisco Cloud Services Router 1000V
6
7 cli_template: |
8     !
9     no snmp-server host 10.93.135.30 version 2c R0
10    no snmp-server host 10.93.130.50 version 2c RW
11    !
12    ntp server 171.68.48.78
13    no ntp server 171.68.48.80
```

Specify Project and Template Names

Specify Filter and Value

Specify CLI Templates

```
dnacenter_day_n_templates > access_switches_updates
1 project_name: Ansible_Project
2 template_name: access_switches_logging
3
4 filter: role
5 filter_value: ACCESS
6
7 cli_template: |
8     !
9     logging host 10.93.141.37
10    logging host 10.93.141.37 transport udp port 8514
11    !
12    service timestamps debug datetime localtime
13    service timestamps log datetime localtime
```

- Filter Options:
- hostname
 - device family
 - role
 - site
 - software version
 - device IP address
 - other params or and/or logic

Ansible Playbook

Note:

Developer + network engineer created

```
dnacenter_day_n > ansible > deploy_cli_templates_ansible.yaml
deploy_cli_templates_ansible.yaml x create_template_tasks.yaml x deploy_template_tasks.yaml x
49  - name: Include the new downloaded vars file
50      include_vars:
51          dir: /Users/gzapodea/PythonCode/dnacenter_day_n_templates/
52          files_matching: "{{ deployment_name }}"
53
54      # include tasks to create a new project, template
55      - include: create_template_tasks.yaml
56
57  - name: Print filter info to select devices to be configured with the CLI Template
58      ansible.builtin.debug:
59          msg: "Filter: {{ filter }}, value: {{ filter_value }}"
60
61  - name: Create an empty "device_hostname" list set_fact
62      set_fact:
63          device_hostname: []
64
65  - name: Append the device hostnames matching the filter
66      set_fact:
67          device_hostname: "{{ device_hostname }} + ['{{ item.hostname }}']"
68      loop: "{{ device_inventory }}"
69      when: item.filter == "{{ filter_value }}"
70
71  - name: Print devices hostnames matching the filter
72      ansible.builtin.debug:
73          msg: "The devices hostnames that match the filter: {{ device_hostname }}"
```

- Git pull, or clone, the repos with device inventory and deployment template
- Checks if template project exists and creates a new one
- Uploads and commits template
- Applies filter to select devices to be configured
- Verifies if devices are managed and reachable
- Deploys CLI Templates
- Retrieves deployment status

CISCO *Live!*

DevNet Zone

#CiscoLiveAPJC DEVNET-2739

© 2022 Cisco and/or its affiliates. All rights reserved. Cisco Public

40

Ansible Playbook – Use Task Lists

deploy_cli_templates_ansible.yaml

```
47 force: yes
48
49 - name: Include the new downloaded vars file
50   include_vars:
51     dir: /Users/gzapodea/PythonCode/dnacenter_day_n_templates/
52     files_matching: "{{ deployment_name }}"
53
54   # include tasks to create a new project, template
55   - include_tasks: create_template_tasks.yaml
56
57 - name: Print filter info to select devices to be configured with the CLI Template
58   ansible.builtin.debug:
59     msg: "Filter: {{ filter }}, value: {{ filter_value }}"
60
61 - name: Create an empty "device_hostname" list set_fact
62   set_fact:
63     device_hostname: []
64
65 - name: Append the device hostnames matching the filter
66   set_fact:
67     device_hostname: "{{ device_hostname }} + [ '{{ item.hostname }}' ]"
68   loop: "{{ device_inventory }}"
```

create_template_tasks.yaml

```
1 ---
2 - name: Print the CLI Template
3   ansible.builtin.debug:
4     msg: "CLI Template: {{ cli_template }}"
5
6 - name: "Verify if existing project with the name: {{ project_name }}"
7   cisco.dna.oject_info: <7 keys>
15  register: project_info
16
17 - name: Project not found
18   ansible.b.ltin.debug: <1 key>
20   when: project_info.dnac_response == []
21
22 - name: Create new project
23   cisco.dna.te_project: <7 keys>
31   register: create_project_info
32   when: project_info.dnac_response == []
33
34 - name: Print create project task id
35   ansible.b.ltin.debug: <1 key>
37   when: project_info.dnac_response == []
38
39 - name: Sleep for 10 seconds to create project and continue with play
40   wait_for: <1 key>
42   delegate_to: localhost
43   when: project_info.dnac_response == []
44
45 - name: Get the project id
```

Simplify Ansible Playbooks by using tasks lists
Re-use of task lists with future playbooks

Ansible Playbook – Use Blocks

The screenshot shows an Ansible playbook named `deploy_cli_templates_ansible.yml` with three tabs: `deploy_cli_templates_ansible.yml`, `deploy_template_tasks.yml`, and `create_template_tasks.yml`. The `deploy_template_tasks.yml` tab is active, showing a block of tasks. A green box highlights the block structure, and a blue box highlights the `when: device_managed_reachable` condition at the bottom.

```
53
54 - block:
55   - name: "Deploy template {{ template_name }} to device {{ device_name }}"
56     cisco.dna.deploy_v2: <10 keys>
57     register: template_deployment_task
58
59
60
61
62
63
64
65
66
67
68
69
70
71 - name: Parse the deployment task id
72   set_fact: <1 key>
73
74
75
76
77
78
79
80
81
82
83
84 - name: Verify the template deployment result
85   cisco.dnac.task_info: <7 keys>
86   register: deployment_task_info
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102 - name: Parse the deployment task result
103   set_fact: <1 key>
104
105
106
107
108
109
110
111
112
113
114
115
116
```

deploy_template_tasks.yml

Group several tasks using Ansible Blocks

Execute the Block when condition is True

Run the Ansible Playbook

Terminal: Local x + v

```
(venv) gzapode@GZAP0DEA-M-6766 ansible % ansible-playbook -e "deployment_name=csr_updates.yaml" deploy_cli_templates_ansi
```

PLAY [Ansible CLI Templates]

TASK [Get start times from the system] *****

changed: [localhost]

TASK [Print playbook structure] *****

ok: [localhost] => {

"msg": "---Deploy CLI Templates Ansible-- playbook start time: 2022-05-09T21-09-19"

}

TASK [Print user input] *****

ok: [localhost] => {

"msg": "---Deploy CLI Templates Ansible-- playbook start time: 2022-05-09T21-09-19"

}

TASK [Update template status] *****

ok: [localhost]

TASK [Include the new template] *****

ok: [localhost]

TASK [include_tasks] *****

skipping: [localhost] => (item={'hostname': 'NYC-ACCESS', 'device_ip': '10.93.141.26', 'device_id': '20937bce-b8d8-42c2-b3a8-88cda93fae1e', 'version': '17.3.5', 'device_family': 'Cisco Catalyst 9300 Switch', 'role': 'PDX-M', 'device_ip': 'd6c3d7df09d'})

skipping: [localhost] => (item={'hostname': 'PDX-CORE1', 'device_ip': '5', 'device_family': 'Cisco Catalyst38xx stack-able ethernet', 'role': 'PDX-RN', 'device_ip': 'd7df09d'})

ok: [localhost] => (item={'hostname': 'PDX-RN', 'device_ip': 'd7df09d', 'device_family': 'Cisco Cloud Services Router 1000V', 'role': 'PDX-R0', 'device_ip': 'd7df09d'})

ok: [localhost] => (item={'hostname': 'PDX-R0', 'device_ip': 'd7df09d', 'device_family': 'Cisco Cloud Services Router 1000V', 'role': 'PDX-R0', 'device_ip': 'd7df09d'})

skipping: [localhost] => (item={'hostname': 'PDX-STOCK', 'device_ip': '2a', 'device_family': 'Cisco Catalyst 9300 Switch', 'role': 'PDX-R0', 'device_ip': 'd7df09d'})

ok: [localhost] => (item={'hostname': 'SP', 'device_ip': 'd7df09d', 'device_family': 'Cisco Cloud Services Router 1000V', 'role': 'PDX-R0', 'device_ip': 'd7df09d'})

TASK [Print devices hostnames matching the filter] *****

ok: [localhost] => {

"msg": "The devices hostnames that match the filter: NYC-ACCESS, PDX-CORE1, PDX-M, PDX-RN, PDX-R0, PDX-STOCK, SP"

}

TASK [Print deployment status] *****

ok: [localhost] => {

"msg": "The deployment status is: {'NYC-R0': 'not reachable'}, {'PDX-RN': 'Template deployment successful'}, {'PDX-R0': 'Template deployment successful'}, {'SP': 'not reachable'}]"

}

TASK [Get end timestamp from the system] *****

changed: [localhost]

TASK [Print timestamp] *****

ok: [localhost] => {

"msg": "---Deploy CLI Templates Ansible-- playbook end time: 2022-05-09T21-09-19"

}

PLAY RECAP *****

localhost : ok=63 changed=2 unreachable=0 failed=0 skipped=44 rescued=0 ignored=0

(venv) gzapode@GZAP0DEA-M-6766 ansible %

Specify deployment name as a parameter

Demo Device Configuration



FirefoxFileEditViewHistoryBookmarksToolsWindowHelp

Template Editor - Cisco DNA Centerzapodeanu/dnacentr...https://10.93.141.45/templateEditor110%

Cisco Systems, IncAPIsToolsLAB_AccessCiscoLiveMeraki Dashboard L...Home - AsanaEssentials of Machi...Data & AnalyticsCisco BoxSFEH-NAS - Synolo...CDETSDNACaaPCisco.DNA.Center...SharePointOther Bookmarks

Cisco DNA Center

Template Editor

Find template...

> Cloud DayN Templates

> Compliance

> Custom Profiles

> Jenkins

> Jinja_Templates

> NDB Configurations

> Onboarding Configuration

> Sample Jinja Templates

> Sample Velocity Templates



Introduction

Templates are organized under a project folder structure.

Use the predefined Onboarding Configuration project to define templates for day-0 provisioning.

Click **Create Project** to create a new project.

Hover over the project, click the **gear** icon, and click **Add Template** to add a new template under that project.

As you work on templates, choose **Actions > Save** to save your in-progress work.

When you're ready to commit and deploy a template, choose **Actions > Commit**.

A regular template is a single template.

A composite sequence is a group of templates. Templates in a composite sequence are deployed in the order in which they are added.

You can add committed templates to a network profile on the Design page.

Use template tags only when you have to push different templates for the same device type based on the device tag.

Templates added to a network profile are available for deployment as part of the provisioning workflow.

You can only import templates that have been exported from Cisco DNA Center.



Agenda

- Cisco DNA Center Platform
- Inventory Collection – Python SDK
- Device Configuration – Ansible
- Software Upgrades – Terraform
- Developer Resources
- Lessons Learned

Automation Workflow: Software Upgrades

Software Non-Compliant Device Inventory

```
1- [
2- {
3-   "hostname": "LO-BN",
4-   "device_ip": "10.93.141.28",
5-   "device_id": "db05ad49-2c8d-4094-9df6-3a03f9b28069",
6-   "version": "17.6.1",
7-   "device_family": "Cisco Catalyst 9300 Switch",
8-   "role": "DISTRIBUTION",
9-   "site": "Global/OR/LO/Floor-3",
10-   "site_id": "3bb9e022-b230-40f2-a17e-f588376f31e3"
11- },
12- {
13-   "hostname": "LO-QN",
14-   "device_ip": "10.93.141.20",
15-   "device_id": "6a612b05-1c89-4d0a-a4d3-94ec5e5a7a57",
16-   "version": "17.6.1",
17-   "device_family": "Cisco Catalyst 9300 Switch",
18-   "role": "DISTRIBUTION",
19-   "site": "Global/OR/LO/Floor-3",
20-   "site_id": "3bb9e022-b230-40f2-a17e-f588376f31e3"
21- }
22 ]
```



Terraform Plan

```
48 output "non_compliant_devices_list" {
49   value     = data.github_repository_file.non_compliant_devices.content
50   description = "file_content"
51 }
52
53 # save file to local folder
54 resource "local_file" "file_name" {
55   filename = "non_compliant_devices.json"
56   content   = data.github_repository_file.non_compliant_devices.content
57 }
58
59 # Configure provider with your Cisco DNA Center SDK credentials
60 provider "dnacenter" {
61   username = var.DNAC_USERNAME
62   password = var.DNAC_PASSWORD
63   base_url = var.DNAC_URL
64   debug    = "false"
65   ssl_verify = "false"
66 }
67
68 # import the downloaded info, encoding in json
69 locals {
70   # get json
71   non_compliant_devices_list = jsondecode(data.github_repository_file.non_compliant_devices.content)
```

- Identify the devices that are software non-compliant using the inventory file from GitHub repo
- Automate the image distribution to network devices
- May run on-demand, or scheduled
- Terraform plans pulled from GitHub

Terraform Use Cases

The screenshot shows the Terraform website's 'Common use cases for Terraform' page. The page features a navigation bar with links to Overview, Use Cases, Editions, Registry, Tutorials, Docs, and Community. A 'Terraform Cloud' button and a 'Download' button are also present. The main content area displays eight use cases in a grid:

- Infrastructure as code**: Use infrastructure as code to automate the provisioning of your infrastructure including servers, databases, firewall policies, and almost every other aspect.
- Multi-cloud deployment**: Deploy serverless functions with AWS Lambda, manage Microsoft Azure Active Directory resources, provision a load balancer in Google Cloud, and more.
- Manage Kubernetes**: Provision and manage Kubernetes clusters on AWS, Microsoft Azure, or Google Cloud, and interact with your cluster using the Kubernetes Terraform provider.
- Manage network infrastructure**: Automate key networking tasks, like updating load balancer member pools or applying firewall policies. (This use case is highlighted with a green border and a blue callout box.)
- Manage virtual machine images**: Deploy and manage virtual machine images with Terraform and Packer.
- Integrate with existing workflows**: Automate infrastructure deployments through existing CI/CD workflows.
- Enforce policy as code**: Enforce policies before your users create infrastructure using Sentinel policy as code.
- Inject secrets into Terraform**: Automate using dynamically generated secrets and credentials within Terraform configurations.

A blue callout box with white text is overlaid on the 'Manage network infrastructure' use case, stating: 'This presentation will focus on a manage network infrastructure use case, calling the Cisco DNA Center REST APIs'.

Terraform One Slide

- **HCL** (HashiCorp Configuration Language) – Establishes the syntax Terraform uses for things like arguments, blocks, literal values, and expressions, and writing plans.
- **Provider** – Plugins responsible for understanding API interactions with other platforms and exposing resources based on their APIs.
- **Data Source** – Allows Terraform to use (read) information defined outside of Terraform. Example: providers, local-only
- **Resource** – Are the most important element in the Terraform language. Each resource block describes one or more infrastructure objects – devices, interfaces, operations.
- **Init** – The command is used to initialize a working directory containing Terraform configuration files. This is the first command that should be run, and it is safe to run this command multiple times. It will install the required providers and modules.
- **Plan** – compares the managed infrastructure state to the configuration, and it determines which changes are necessary. It presents a human-readable summary to the user.
- **Apply** – Makes changes to real infrastructure in order to make it match the desired state. It may use saved plans or creates a new plan and asks for approval.

Ref: <https://www.terraform.io/docs>

Terraform Provider and Go SDK

The screenshot shows the Terraform Registry page for the `dnacenter` provider. The breadcrumb navigation at the top reads: `Providers` / `cisco-en-programmability` / `dnacenter` / Version 1.0.13-beta. The page title is `dnacenter`. On the left, under 'DNACENTER DOCUMENTATION', there is a search bar and a list of links: `dnacenter provider`, `> Resources`, and `> Data Sources`. The main content area is titled `dnacenter Provider` and includes an 'Overview' tab and a 'USE PROVIDER' button. The text describes the provider's purpose and configuration requirements. A callout box on the right lists key features: extended REST API coverage, support for version 2.3.3.x, a January 2021 release, and over 16,600 downloads. The 'Example Usage' section shows a Terraform configuration snippet for setting up the provider with credentials and environment variables.

Providers / cisco-en-programmability / dnacenter / Version 1.0.13-beta ▾ Latest Version

dnacenter Overview Documentation [USE PROVIDER](#)

DNACENTER DOCUMENTATION

Filter

[dnacenter provider](#)

- > Resources
- > Data Sources

dnacenter Provider

The Cisco DNA Center provider is used to interact with Cisco DNA Center APIs (2.2.3.3). The provider needs to be configured with the proper credentials before it can be used.

Use the navigation to the left to read about the available resources.

Example Usage

The DNA Center SDK client could be generated with the following parameters:

```
# Configure provider with your Cisco DNA Center SDK credentials
provider "dnacenter" {
  # Cisco DNA Center user name
  username = "admin"
  # it can be set using the environment variable DNAC_BASE_URL
  # Cisco DNA Center password
}
```

- Extended coverage of Cisco DNA Center REST APIs
- Support for Cisco DNA Center version 2.3.3.x
- Released January 2021
- 16,600+ downloads

<https://registry.terraform.io/providers/cisco-en-programmability/dnacenter>

Cisco DNA Center Terraform Provider Docs

The screenshot displays the GitHub repository for the Cisco DNA Center Terraform Provider. The repository is named `cisco-en-programmability / terraform-provider-dnacenter` and is public. It has 9 forks, 12 stars, and 4 issues. The file `image_distribution.md` is open, showing the documentation for the `dnacenter_image_distribution` resource. The documentation includes a table with the resource details, a warning, and an example usage.

page_title	subcategory	description
dnacenter_image_distribution Resource - terraform-provider-dnacenter		It performs create operation on Software Image Management (SWIM). - Distributes a software image on a given device. Software image must be imported successfully into DNA Center before it can be distributed

dnacenter_image_distribution (Resource)

It performs create operation on Software Image Management (SWIM). - Distributes a software image on a given device. Software image must be imported successfully into DNA Center before it can be distributed

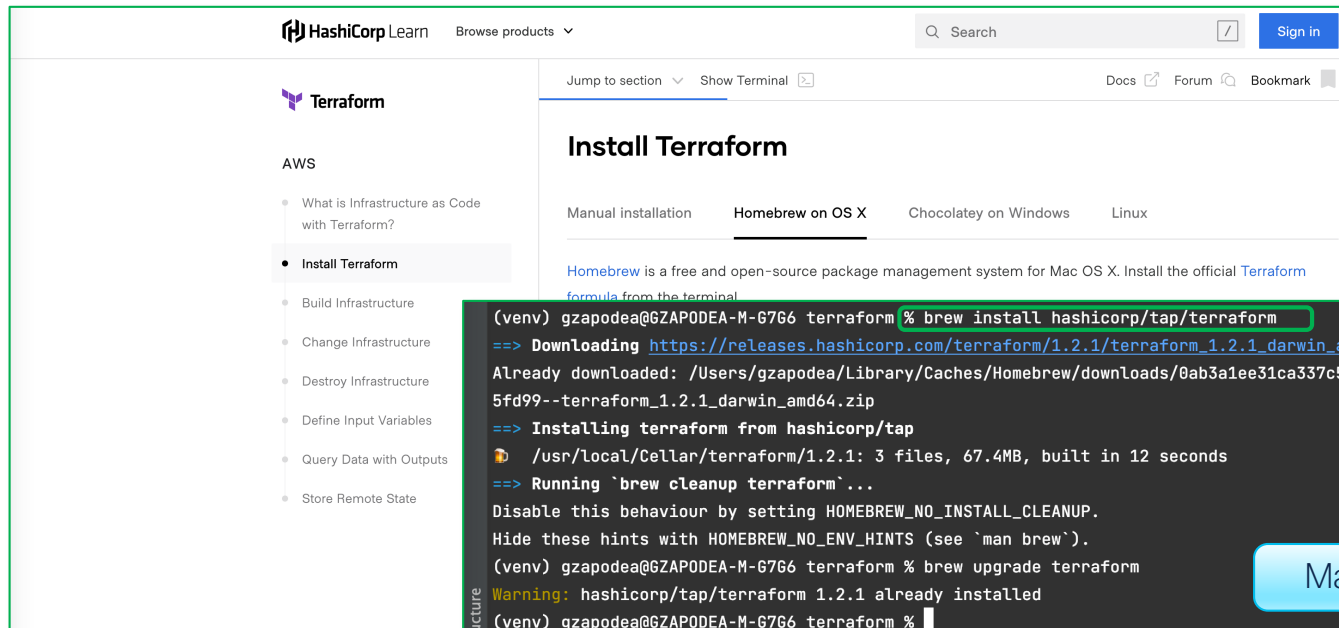
~>Warning: This resource does not represent a real-world entity in Cisco DNA Center, therefore changing or deleting this resource on its own has no immediate effect. Instead, it is a task part of a Cisco DNA Center workflow. It is executed in DNACenter without any additional verification. It does not check if it was executed before or if a similar configuration or action already existed previously.

Most commonly, this resource is used together with `swim_image_file` and `network_device` to distribute a software image.

Example Usage

```
provider "dnacenter" {
  debug = "true"
}
```

Terraform Installation of the Open-Source Edition



The screenshot shows the HashiCorp Learn website for Terraform installation. The sidebar on the left lists various topics under 'AWS', with 'Install Terraform' selected. The main content area has tabs for 'Manual installation', 'Homebrew on OS X', 'Chocolatey on Windows', and 'Linux'. The 'Homebrew on OS X' tab is active, displaying instructions for installing Terraform using Homebrew. A terminal window is overlaid on the page, showing the following commands and output:

```
(venv) gzapodea@GZAPODEA-M-67G6 terraform % brew install hashicorp/tap/terraform
==> Downloading https://releases.hashicorp.com/terraform/1.2.1/terraform_1.2.1_darwin_amd64.zip
Already downloaded: /Users/gzapodea/Library/Caches/Homebrew/downloads/0ab3a1ee31ca337c52e4c15b3f560e4fac08867e4fcf1ecaec10cc2a2f65fd99--terraform_1.2.1_darwin_amd64.zip
==> Installing terraform from hashicorp/tap
📦 /usr/local/Cellar/terraform/1.2.1: 3 files, 67.4MB, built in 12 seconds
==> Running `brew cleanup terraform`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
(venv) gzapodea@GZAPODEA-M-67G6 terraform % brew upgrade terraform
Warning: hashicorp/tap/terraform 1.2.1 already installed
(venv) gzapodea@GZAPODEA-M-67G6 terraform %
```

Mac OS X Terraform Installation

<https://learn.hashicorp.com/tutorials/terraform/install-cli>

Cisco DNA Center Terraform Provider Installation

To use this provider, copy and paste this code into your Terraform configuration. Then, run terraform init.

```
terraform {
  required_providers {
    dnacenter = {
      source = "hashicorp.com/edu/dnacenter"
      version = "0.3.0-beta"
    }
  }
}

# Configure provider with your Cisco DNA Center credentials
provider "dnacenter" {
  username = "admin"
  password = "admin123"
  base_url = "https://172.168.191.1"
  debug = "false"
  ssl_verify = "false"
}

# this plan will distribute the golden software image to the devices from the dnacenter
```

Start creating a plan

Terminal: Local

```
(venv) gzapodee@GZAPODEA-M-6766 terraform % terraform init
Initializing modules...

Initializing the backend...

Initializing provider plugins...
- Finding cisco-en-programmability/dnacenter versions matching "0.3.0-beta"...
- Installing cisco-en-programmability/dnacenter v0.3.0-beta...
- Installed cisco-en-programmability/dnacenter v0.3.0-beta (self-signed, key ID A3DE487CD358EC62)

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Run command:
"terraform init"
It will download the latest
Cisco DNA Center provider
version

Note: Instructions to get started with Cisco DNA Center Ansible modules

<https://github.com/cisco-en-programmability/terraform-provider-dnacenter/blob/main/README.md>

Software Image Distribution Plan

```
1
2 # this plan will distribute the golden software image to the devices from the "non_compliant_devices.json" file
3
4 terraform {
5   required_providers {
6     dnacenter = {
7       version = "0.3.0-beta"
8       source  = "cisco-en-programmability/dnacenter"
9     }
10    github = {
11      source = "integrations/github"
12      version = "~> 4.0"
13    }
14  }
15 }
16
17 provider "github" {
18   token = var.GITHUB_TOKEN
19 }
20
21 # git pull to download the image non compliant devices inventory
22 data "github_repository_file" "non_compliant_devices" {
23   repository = "dnacenter_day_n_inventory"
24   branch     = "main"
25   file       = "non_compliant_devices.json"
26 }
27
28 output "non_compliant_devices_list" {
29   value       = data.github_repository_file.non_compliant_devices.content
30   description = "file_content"
31 }
```

Specify the Cisco DNA Center and
GitHub Providers

Data Source:
Pull from GitHub the image non-
compliant devices inventory

CISCO *Live!*

DevNet Zone

#CiscoLiveAPJC DEVNET-2739

© 2022 Cisco and/or its affiliates. All rights reserved. Cisco Public

54

Software Image Distribution Plan – continued

```
38
39 # Configure provider with your Cisco DNA Center SDK credentials
40 provider "dnacenter" {
41   username = var.DNAC_USERNAME
42   password = var.DNAC_PASSWORD
43   base_url = var.DNAC_URL
44   debug    = "false"
45   ssl_verify = "false"
46 }
47
48 # import the downloaded info, encoding in json
49 locals {
50   # get json
51   non_compliant_devices_list = jsondecode(data.github_repository_file.non_compliant_devices.content)
52 }
53
54 # define the software distribution module that will be called for each device
55 module "swim_upgrade" {
56   source      = "./modules/distribution"
57   count       = length(local.non_compliant_devices_list)
58   device_info = local.non_compliant_devices_list[count.index]
59 }
60
61 # output the devices hostname requiring an
62 output "device_upgraded" {
63   description = "Device Hostname"
64   value       = [for device in local.non_compliant_devices_list : device.hostname]
65 }
66
```

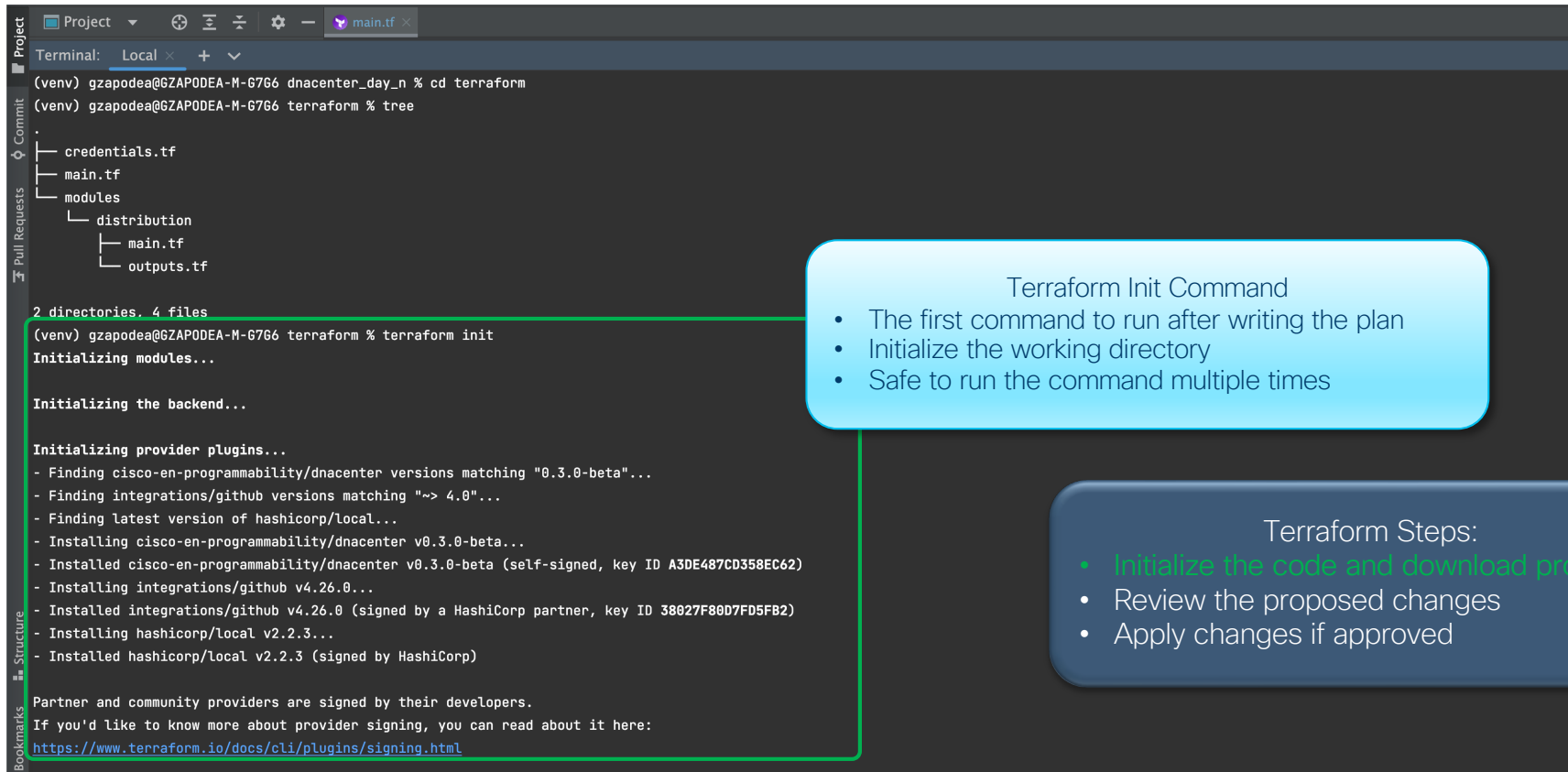
Specify the Cisco DNA Center credentials

Loop (count) through the list of all software non-compliant devices, calling the "swim_upgrade" module

```
13 variable "device_info" {
14   description = "Device UUIDs to be upgraded"
15   type        = map(string)
16 }
17
18 # use the image distribution resource, provide the device uuid
19 resource "dnacenter_image_distribution" "response" {
20   provider = dnacenter
21   lifecycle {
22     create_before_destroy = true
23   }
24   parameters {
25     payload {
26       device_uuid = var.device_info.device_id
27     }
28   }
29 }
```

Resource:
"swim_upgrade" module
Use the image distribution resource

Initialize the Terraform Working Directory



```
(venv) gzapodea@GZAP0DEA-M-6766 dnacenter_day_n % cd terraform
(venv) gzapodea@GZAP0DEA-M-6766 terraform % tree
.
├── credentials.tf
├── main.tf
├── modules
│   └── distribution
│       ├── main.tf
│       └── outputs.tf
└── 2 directories, 4 files

(venv) gzapodea@GZAP0DEA-M-6766 terraform % terraform init
Initializing modules...

Initializing the backend...

Initializing provider plugins...
- Finding cisco-en-programmability/dnacenter versions matching "0.3.0-beta"...
- Finding integrations/github versions matching "~> 4.0"...
- Finding latest version of hashicorp/local...
- Installing cisco-en-programmability/dnacenter v0.3.0-beta...
- Installed cisco-en-programmability/dnacenter v0.3.0-beta (self-signed, key ID A3DE487CD358EC62)
- Installing integrations/github v4.26.0...
- Installed integrations/github v4.26.0 (signed by a HashiCorp partner, key ID 38027F80D7FD5FB2)
- Installing hashicorp/local v2.2.3...
- Installed hashicorp/local v2.2.3 (signed by HashiCorp)

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html
```

Terraform Init Command

- The first command to run after writing the plan
- Initialize the working directory
- Safe to run the command multiple times

Terraform Steps:

- Initialize the code and download providers
- Review the proposed changes
- Apply changes if approved

Create Execution Plan

```
Project
  main.tf x
Terminal: Local x + v
(venv) gzapodea@G2APODEA-M-6766 terraform % terraform plan
data.github_repository_file.non_compliant_devices: Reading...
data.github_repository_file.non_compliant_devices: Read complete after 2s [id=dnacenter_day_n_inventory/non_compliant_devices.json]

Terraform used the selected providers to generate the following execution plan:
+ create
<= read (data resources)

Terraform will perform the following actions:

# local_file.file_name will be created
+ resource "local_file" "file_name" {
  + content      = jsonencode(
    [
      + {
        + device_family = "Cisco Catalyst 9300 Switch"
        + device_id     = "6a612b05-1c89-4d0a-a4d3-94ec5e5a7a57"
        + device_ip     = "10.93.141.28"
        + hostname      = "LO-CN"
        + role           = "DISTRIBUTION"
        + site           = "Global/OR/L0/Floor-3"
        + site_id       = "3bb9e022-b230-40f2-a17e-f588376f31e3"
        + version        = "17.6.1"
      },
      + {
        + device_family = "Cisco Catalyst 9300 Switch"
        + device_id     = "db05ad49-2c8d-4094-9df6-3a03f9b28069"
        + device_ip     = "10.93.141.28"
        + hostname      = "LO-BN"
        + role           = "DISTRIBUTION"
        + site           = "Global/OR/L0/Floor-3"
      },
    ],
  )
}
```

Terraform Plan Command

- Reads existing state
- Creates execution plan
- Allows the preview of changes

Outputs:

Devices to be upgraded, identified based on current state and software non-compliant devices inventory

Terraform Steps :

- Initialize the code and download provider
- **Review the proposed changes**
- Apply changes if approved

CISCO *Live!*

DevNet Zone

#CiscoLiveAPJC DEVNET-2739

© 2022 Cisco and/or its affiliates. All rights reserved. Cisco Public

57

Plan Execution

```
Terminal: Local x + v
(venv) gzapodea@GZAP0DEA-M-G766 terraform % terraform apply -auto-approve
data.github_repository_file.non_compliant_devices: Reading...
data.github_repository_file.non_compliant_devices: Read complete after 1s [id=dnacenter_day_n_inventory/non_compliant_devices.json]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create
<= read (data resources)

Terraform will perform the following actions:

# local_file.file_name will be created
+ resource "local_file" "file_name" {
  + content      = jsonencode(
    [
      + {
        + device_family = "Cisco Catalyst 9300 Switch"
        + device_id     = "6a612b05-1c89-4d0a-a4d3-94ec5e5a7a57"
        + device_ip     = "10.93.141.28"
        + hostname      = "LO-CN"
        + role           = "DISTRIBUTION"
        + site           = "GLobaL/OR/L0/Floor-3"
        + site_id        = "3bb9e022-b230-40f2-a17e-f588376f31e3"
        + version        = "17.6.1"
      },
      + {
        + device_family = "Cisco Catalyst 9300 Switch"
        + device_id     = "db05ad49-2c8d-4094-9df6-3a03f9b28069"
        + device_ip     = "10.93.141.28"
        + hostname      = "LO-BN"
        + role           = "DISTRIBUTION"
        + site           = "GLobaL/OR/L0/Floor-3"
      }
    ]
  }
```

Terraform Apply Command

- Creates execution plan
- May use saved plan, or use "auto-approve"
- Parameters could be provided
- Will return outputs with what was changed

Terraform Steps:

- Initialize the code and download providers
- Review the proposed changes
- **Apply changes if approved**

CISCO *Live!*

DevNet Zone

#CiscoLiveAPJC DEVNET-2739

© 2022 Cisco and/or its affiliates. All rights reserved. Cisco Public

58

Plan Execution End

```
Terminal: Local x + v
+ {
  + device_family = "Cisco Catalyst 9300 Switch"
  + device_id     = "6a612b05-1c89-4d0a-a4d3-94ec5e5a7a57"
  + device_ip    = "10.93.141.20"
  + hostname     = "LO-CN"
  + role         = "DISTRIBUTION"
  + site         = "Global/OR/LO/Floor-3"
  + site_id      = "3bb9e022-b230-40f2-a17e-f588376f31e3"
  + version      = "17.6.1"
},
+ {
  + device_family = "Cisco Catalyst 9300 Switch"
  + device_id     = "db05ad49-2c8d-4094-9df6-3a03f9b28069"
  + device_ip    = "10.93.141.28"
  + hostname     = "LO-BN"
  + role         = "DISTRIBUTION"
  + site         = "Global/OR/LO/Floor-3"
  + site_id      = "3bb9e022-b230-40f2-a17e-f588376f31e3"
  + version      = "17.6.1"
},
}

local_file.file_name: Creating...
local_file.file_name: Creation complete after 0s [id=9344cbb5d962282fb196747acf7851b5a6be5bb8]
module.swim_upgrade[1].dnacenter_image_distribution.response: Creating...
module.swim_upgrade[0].dnacenter_image_distribution.response: Creating...
module.swim_upgrade[0].dnacenter_image_distribution.response: Still creating... [10s elapsed]
module.swim_upgrade[1].dnacenter_image_distribution.response: Still creating... [10s elapsed]
module.swim_upgrade[1].dnacenter_image_distribution.response: Still creating... [20s elapsed]
module.swim_upgrade[0].dnacenter_image_distribution.response: Still creating... [20s elapsed]
```

```
Terminal: Local x + v
module.swim_upgrade[1].dnacenter_image_distribution.response: Still creating... [4m30s elapsed]
module.swim_upgrade[0].dnacenter_image_distribution.response: Still creating... [4m30s elapsed]
module.swim_upgrade[0].dnacenter_image_distribution.response: Still creating... [4m40s elapsed]
module.swim_upgrade[1].dnacenter_image_distribution.response: Still creating... [4m40s elapsed]
module.swim_upgrade[0].dnacenter_image_distribution.response: Still creating... [4m50s elapsed]
module.swim_upgrade[1].dnacenter_image_distribution.response: Still creating... [4m50s elapsed]
module.swim_upgrade[0].dnacenter_image_distribution.response: Still creating... [5m0s elapsed]
module.swim_upgrade[1].dnacenter_image_distribution.response: Still creating... [5m0s elapsed]
module.swim_upgrade[0].dnacenter_image_distribution.response: Still creating... [5m10s elapsed]
module.swim_upgrade[1].dnacenter_image_distribution.response: Still creating... [5m10s elapsed]
module.swim_upgrade[1].dnacenter_image_distribution.response: Creation complete after 5m16s [id=1654808554]
module.swim_upgrade[1].data.dnacenter_task.response: Reading...
module.swim_upgrade[1].data.dnacenter_task.response: Read complete after 0s [id=1654808554]
module.swim_upgrade[0].dnacenter_image_distribution.response: Still creating... [5m20s elapsed]
module.swim_upgrade[0].dnacenter_image_distribution.response: Creation complete after 5m21s [id=1654808559]
module.swim_upgrade[0].data.dnacenter_task.response: Reading...
module.swim_upgrade[0].data.dnacenter_task.response: Read complete after 0s [id=1654808559]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

Outputs:
device_upgraded = [
  "LO-CN",
  "LO-BN",
]

non_compliant_devices_list = [{"hostname": "LO-CN", "device_ip": "10.93.141.20", "device_id": "6a612b05-1c89-4d0a-a4d3-94ec5e5a7a57", "version": "17.6.1", "device_family": "Cisco Catalyst 9300 Switch", "role": "DISTRIBUTION", "site": "Global/OR/LO/Floor-3", "site_id": "3bb9e022-b230-40f2-a17e-f588376f31e3"}, {"hostname": "LO-BN", "device_ip": "10.93.141.28", "device_id": "db05ad49-2c8d-4094-9df6-3a03f9b28069", "version": "17.6.1", "device_family": "Cisco Catalyst 9300 Switch", "role": "DISTRIBUTION", "site": "Global/OR/LO/Floor-3", "site_id": "3bb9e022-b230-40f2-a17e-f588376f31e3"}]

(venv) gzapodea@GZAPPODEA-M-6706 terraform %
```

Execution Completed

Image distribution to devices started

Demo Software Upgrades



PyCharm File Edit View Navigate Code Refactor Run Tools Git Window Help

dnacenter_day_n - terraform/main.tf

inventory_collection_sdk

dnacenter_day_n > terraform > main.tf

Project

- dnacenter_day_n ~/PythonCode/dr
 - ansible
 - git
 - inventory
 - python_sdk
 - terraform
 - .terraform
 - modules
 - distribution
 - main.tf
 - outputs.tf
 - credentials.tf
 - main.tf
 - venv
 - .gitignore
 - CODE_OF_CONDUCT.md
 - CONTRIBUTING.md
 - environment.env
 - LICENSE
 - NOTICE
 - README.md
 - requirements.txt
 - External Libraries
 - Scratches and Consoles

Commit

Pull Requests

Structure

Bookmarks

Notifications

```
1 # this plan will distribute the golden software image to the devices from the "non_compliant_devices.json" file
2
3
4 terraform {
5   required_providers {
6     dnacenter = {
7       version = "0.3.0-beta"
8       source  = "cisco-en-programmability/dnacenter"
9     }
10    github = {
11      source = "integrations/github"
12      version = "~> 4.0"
13    }
14  }
15 }
16
17 provider "github" {
18   token = var.GITHUB_TOKEN
19 }
20
21 # git pull to download the image non compliant devices inventory
22 data "github_repository_file" "non_compliant_devices" {
23   repository = "dnacenter_day_n_inventory"
24   branch     = "main"
25   file       = "non_compliant_devices.json"
26 }
27
28 output "non_compliant_devices_list" {
29   value     = data.github_repository_file.non_compliant_devices.content
30   description = "file_content"
31 }
32
```

48:31 LF UTF-8 2 spaces Python 3.9 (dnacenter_day_n) main

GitHub Security

The screenshot displays the GitHub Security overview page for the repository `cisco-en-programmability/dnacentr_day_n`. The page is divided into a sidebar on the left and a main content area on the right. The sidebar contains links to Overview, Reporting, Policy, Advisories, Vulnerability alerts, Dependabot, and Code scanning. The main content area is titled "Security overview" and lists several security settings:

- Security policy** - Disabled: Define how users should report security vulnerabilities for this repository. A button "Set up a security policy" is available.
- Security advisories** - Enabled: View or disclose security advisories for this repository. A link "View security advisories" is provided.
- Private vulnerability reporting** - Enabled: Allow users to privately report potential security vulnerabilities. A link "See reported vulnerabilities" is provided.
- Dependabot alerts** - Enabled: Get notified when one of your dependencies has a vulnerability. A link "View Dependabot alerts" is provided.
- Code scanning alerts** - Enabled: Automatically detect common vulnerability and coding errors. A link "View alerts" is provided.

A green box highlights the sections for Security advisories, Private vulnerability reporting, Dependabot alerts, and Code scanning alerts.

GitHub Security – Code Scanning

Search or jump to... / Pull requests Issues Marketplace Explore

cisco-en-programmability / dnacentr_day_n Public Edit Pins Unwatch 3 Fork 2 Star 2

<> Code Issues Pull requests 2 Actions Projects Wiki Security Insights Settings

Overview

Reporting

- Policy
- Advisories
- Vulnerability alerts
- Dependabot
- Code scanning**

Code scanning

Latest scan	Branch	Workflow	Lines scanned	Duration
1 hour ago	main	CodeQL	153 / 226	5m 56s

is:open branch:main

0 Open 0 Closed

No code scanning alerts found. Keep up the good work!

Found 61 workflows

- Dependency Review** By GitHub Actions
Scans Pull Requests on each push for the introduction and/or resolution of vulnerable dependencies to the repository.
Configure Dependency review
- CodeQL Analysis** By GitHub
Security analysis from GitHub for C, C++, C#, Go, Java, JavaScript, TypeScript, Python, and Ruby developers.
Configure Code scanning
- APIsec Scan** By APIsec
APIsec provides the industry's only automated and continuous API testing platform that uncovers security vulnerabilities and logic flaws in APIs.
Configure Code scanning
- Checkmarx** By Checkmarx
Beat vulnerabilities with more secure code. Scan your code with Checkmarx One and see results in the GitHub code scanning.
Configure Code scanning
- CxSAST** By Checkmarx
Scan your code with Checkmarx CxSAST and see your results in the GitHub security tab.
Configure Code scanning
- Codacy Security Scan** By Codacy
Free, out-of-the-box, security analysis provided by multiple open source static analysis tools.
Configure Code scanning
- Red Hat CodeReady Dependency Analytics** By Red Hat
Scan your project's dependencies with CodeReady Dependency Analytics.
Configure Code scanning
- DevSkim** By Microsoft CST-E
DevSkim is a security linter that highlights common security issues in source code.
Configure Code scanning
- EthicalCheck** By APIsec
EthicalCheck provides the industry's only free & automated API security testing service that uncovers security vulnerabilities using OWASP API list.
Configure Code scanning
- Fortify on Demand Scan** By Micro Focus
Integrate Fortify's comprehensive static code analysis (SAST) for 27+ languages into your DevOps workflows to build secure software faster.
Configure Code scanning
- Frogbot Scan and Fix** By JFrog
Automatically creates pull requests with fixes for vulnerable project dependencies. Uses JFrog Xray to scan the project. Included as part of JFrog's free subscription.
Configure Code scanning
- Frogbot Scan Pull Request** By JFrog
Automatically scans new pull requests for security vulnerabilities. Uses JFrog Xray to scan the project. Included as part of JFrog's free subscription.
Configure Code scanning

GitHub Security – Dependencies Alerts

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security 6

Insights

Settings

Overview

Reporting

Policy

Advisories

Vulnerability alerts

Dependabot 6

Code scanning

Dependabot alerts

is:open

6 Open

0 Closed

Package

Ecosystem

Manifest

Severity

Sort

Ixml NULL Pointer Dereference allows attackers to cause a denial of service

High

#8 opened 4 months ago • Detected in Ixml (pip) • requirements.txt

Ixml's HTML Cleaner allows crafted and SVG embedded scripts to pass through

High

#6 opened 11 months ago • Detected in Ixml (pip) • requirements.txt

CRLF injection

Moderate

#5 opened 17 months ago • Detected in urllib3 (pip) • requirements.txt

Ixml before 4.6.3 vulnerable to Cross-Site Scripting

Moderate

#3 opened 2 years ago • Detected in Ixml (pip) • requirements.txt

Ixml vulnerable to Cross-site Scripting

Moderate

#2 opened 2 years ago • Detected in Ixml (pip) • requirements.txt

RSA decryption vulnerable to Bleichenbacher timing vulnerability

Moderate

#1 opened 2 years ago • Detected in cryptography (pip) • requirements.txt

Dependabot alerts

surface known security vulnerabilities in some dependency manifest files. [Dependabot security updates](#) automatically keep your application up-to-date by updating dependencies in response to these alerts. [Dependabot version updates](#) can also help keep dependencies updated.

*Note:
3 years old repo, not maintained*

Day N Operations

https://github.com/cisco-en-programmability/dnacenter_day_n

cisco-en-programmability / dnacenter_day_n Public

Edit Pins Unwatch 3 Fork 2 Star 3

<> Code Issues Pull requests 2 Actions Projects Wiki Security Insights Settings

main 2 branches 0 tags Go to file Add file <> Code

zapodeanu Merge remote-tracking branch 'origin/main' into... 39a803e 1 minute ago 26 commits

.github/workflows	Create codeql.yml	24 days ago
ansible	updated files, improve comments	1 minute ago
inventory	updated task list, inventory	6 months ago
python_sdk	updated files, improve comments	1 minute ago
terraform	updated files, improve comments	1 minute ago
.gitignore	updated gitignore	7 months ago
CODE_OF_CONDUCT.md	initial commit	7 months ago
CONTRIBUTING.md	initial commit	7 months ago

About

Applications that will automate Day N operations using Cisco DNA Center APIs and these libraries: Cisco DNA Center Python SDK, Cisco DNA Center Ansible Collection, Cisco DNA Center Terraform Provider

Readme View license Code of conduct 3 stars 3 watching 2 forks

CISCO *Live!*

DevNet Zone

#CiscoLiveAPJC DEVNET-2739

© 2022 Cisco and/or its affiliates. All rights reserved. Cisco Public

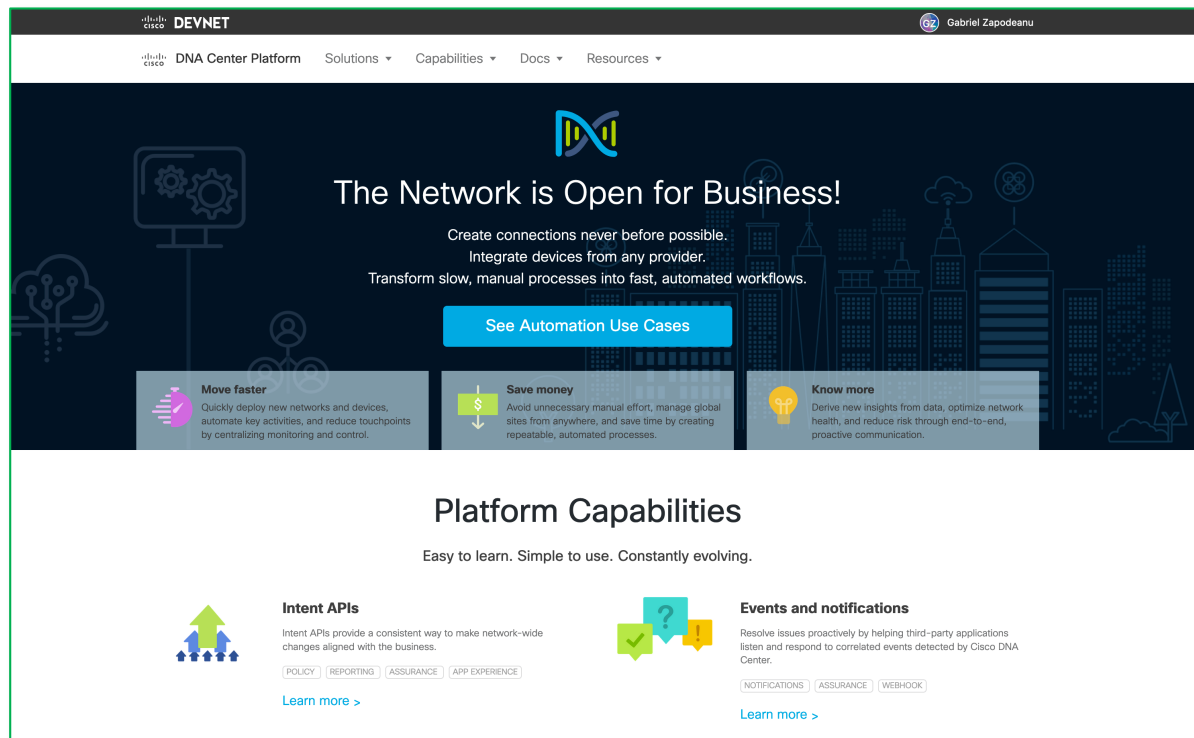
65



Agenda

- Cisco DNA Center Platform
- Inventory Collection – Python SDK
- Device Configuration – Ansible
- Software Upgrades – Terraform
- Developer Resources
- Lessons Learned

Cisco DNA Center – DevNet Resources



- API Documentation
- User Guides
- Learning Labs
- Sandboxes
- Code Exchange and Automation Exchange

Cisco EN Programmability GitHub Org

Cisco EN Programmability
Cisco ENB Programmability Technical Marketing Engineering sample code for Cisco DNA Center REST APIs and Integrations
23 followers | cisco-en-programmability@cisco.c...

Overview | Repositories (36) | Projects | Packages | Teams (1) | People (6) | Settings

Pinned

- dnacentersdk** (Public)
Cisco DNA Center Python SDK
Python | 42 stars | 18 forks
- dnacenter-ansible** (Public)
Cisco DNA Center Ansible modules
Python | 19 stars | 14 forks
- dnacenter_fabric_operations** (Public)
Repo for Cisco DNA Center Software Defined Access (SDA) operations
Python | 11 stars | 2 forks
- dnacenter_day_n** (Public)
Applications that will automate Day N operations using Cisco DNA Center APIs and these libraries: Cisco DNA Center Python SDK, Cisco DNA Center Ansible Collection, Cisco DNA Center Terraform Provider
Python | 3 stars | 2 forks
- splunk-apps** (Public)
Splunk Add-On App for Cisco DNA Center
Python | 1 star | 2 forks
- dnacenter_webhook_receiver** (Public)
Sample code for a Cisco DNA Center Webhook Receiver
Python | 9 stars | 7 forks

View as: **Public**
You are viewing the README and repositories as a public user.

You can...

Get started with this organization

People

Invite

Top languages

- Python
- Go

- Cisco DevNet DNA Center use cases
- Python SDK
- Ansible modules
- Go SDK and Terraform provider
- Splunk Integration
- Compliance Use Case
- Other sample code

YouTube Channel – EN Programmability

The screenshot displays the Cisco EN Programmability YouTube channel. At the top, there's a search bar and navigation links for Cisco DNA Center (DESIGN, POLICY, PROVISION, ASSURANCE, PLATFORM). The 'PLATFORM' link is highlighted. Below this, the 'Platform' section is visible, version 1.1.0, released 5/15/2019. It includes an 'Overview' tab and a welcome message. The channel's name 'Cisco EN Programmability' and subscriber count '294 subscribers' are shown. Navigation tabs for 'HOME', 'VIDEOS', 'PLAYLISTS', 'CHANNELS', and 'ABOUT' are present. The 'VIDEOS' tab is active, showing a list of uploads. The first video is 'NETCONF with YANG Suite' (10:55, 68 views, 1 month ago). Other videos include 'Cisco IOS XE Terraform provider introduction and...' (9:27, 117 views, 3 months ago), 'Cisco Catalyst 9300 IOS XE 17.5 GNMI GNOI OS.proto -...' (2:22, 139 views, 1 year ago), 'Cisco DNA Center Client Proximity API and Event' (13:26, 103 views, 1 year ago), 'Cisco DNA Center Real Time Notifications to Splunk...' (8:08, 2.1K views, 2 years ago), and 'Cisco DNA Center IP Address Conflict Prevention' (5:57, 130 views, 2 years ago).

Cisco DNA Center – Developer Resources



<https://developer.cisco.com/dnacenter/>



<https://github.com/cisco-en-programmability>



<http://cs.co/EN-Programmability-Videos>



<https://galaxy.ansible.com/cisco/dnac>



<https://registry.terraform.io/providers/cisco-en-programmability/dnacenter>



<https://dnacentersdk.readthedocs.io/en/latest/>



<https://github.com/cisco-en-programmability/dnacenter-go-sdk>



DevNet Zone

#CiscoLiveAPJC DEVNET-2739

© 2022 Cisco and/or its affiliates. All rights reserved. Cisco Public

70



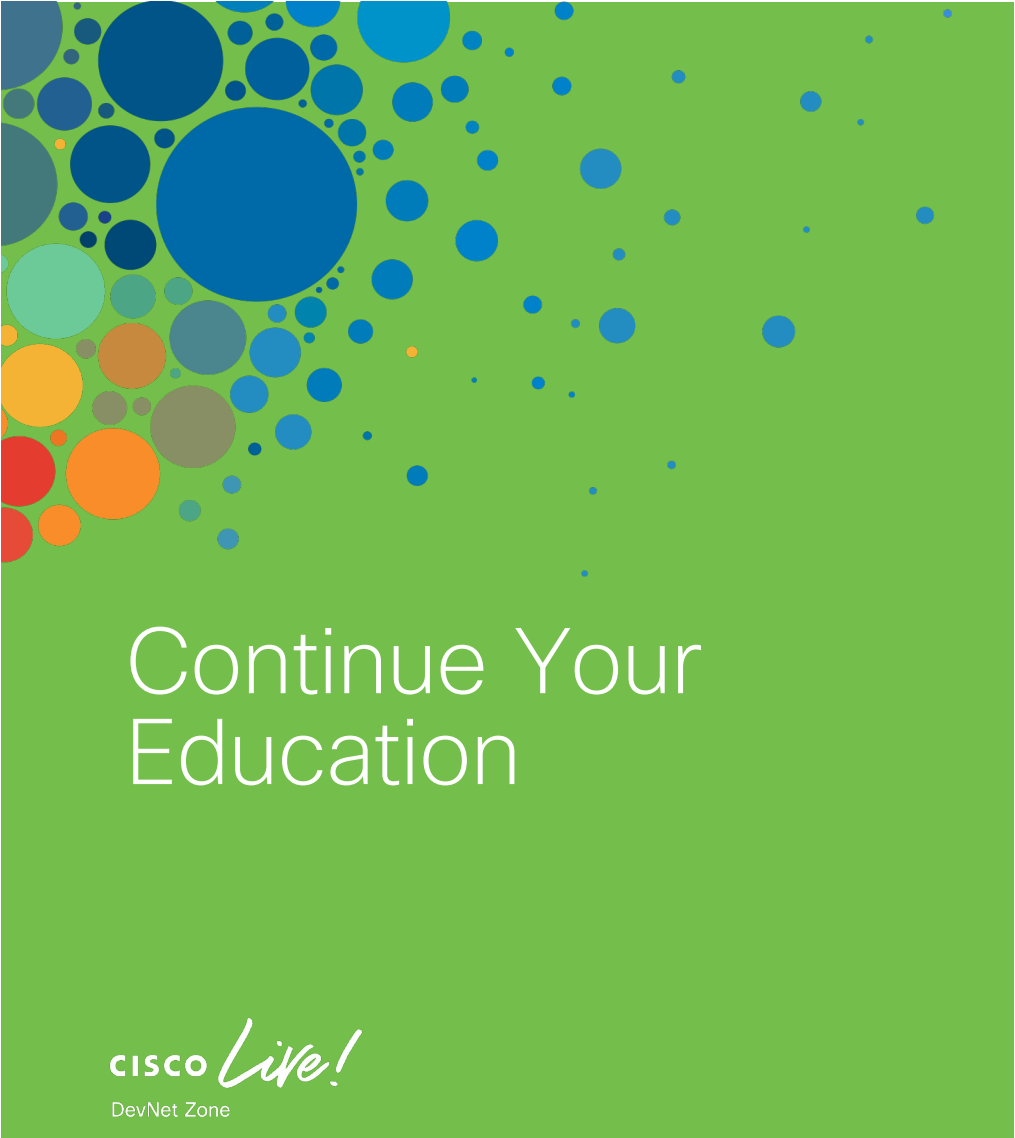
Agenda

- Cisco DNA Center Platform
- Inventory Collection – Python SDK
- Device Configuration – Ansible
- Software Upgrades – Terraform
- Developer Resources
- Lessons Learned

Cisco DNA Center Libraries



- Any network automation use case may be implemented with any Cisco DNA Center library
- Python SDK is very versatile, excellent choice for complex use cases, and integrations
- Ansible and Terraform are great options for:
 - Simple use cases, or end-to-end automations with Data Center and Cloud
 - When DevOps engineers use these automation tools
- Partner with developers to create your automations workflows
- Code re-use – build your team’s library: modules, tasks lists, plans



Continue Your
Education

CISCO *Live!*

DevNet Zone

DEVNET-2031 - Three Cisco DNA Center
Integrations: Splunk, Webex Notifications, and
PagerDuty

Wednesday, Dec 7th - 9:00 - 9:45 AM

DEVNET-2151

Jenkins Automations for Cisco DNA Center

Thursday, Dec 8th - 3:00 - 3:45 PM

World of Solutions - Cisco DNA Center Demos

Session Surveys

We would love to know your feedback on this session!

- Complete the session surveys in the Cisco Events mobile app. You'll earn some points in the Cisco Live Game and potentially win a prize.
- Complete a minimum of four session and the overall event surveys to claim a Cisco Live cable bag.

Continue your education



Visit the Cisco Showcase for related demos



Book your one-on-one Meet the Expert 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



Cisco Learning and Certifications

From technology training and team development to Cisco certifications and learning plans, let us help you empower your business and career. www.cisco.com/go/certs

Pay for Learning with Cisco Learning Credits

(CLCs) are prepaid training vouchers redeemed directly with Cisco.



Learn

Cisco U.

IT learning hub that guides teams and learners toward their goals

Cisco Digital Learning

Subscription-based product, technology, and certification training

Cisco Modeling Labs

Network simulation platform for design, testing, and troubleshooting

Cisco Learning Network

Resource community portal for certifications and learning



Train

Cisco Training Bootcamps

Intensive team & individual automation and technology training programs

Cisco Learning Partner Program

Authorized training partners supporting Cisco technology and career certifications

Cisco Instructor-led and Virtual Instructor-led training

Accelerated curriculum of product, technology, and certification courses



Certify

Cisco Certifications and Specialist Certifications

Award-winning certification program empowers students and IT Professionals to advance their technical careers

Cisco Guided Study Groups

180-day certification prep program with learning and support

Cisco Continuing Education Program

Recertification training options for Cisco certified individuals



DevNet Zone

#CiscoLiveAPJC DEVNET-2739

© 2022 Cisco and/or its affiliates. All rights reserved. Cisco Public

76



The bridge to possible

Thank you

CISCO *Live!*

DevNet Zone

#CiscoLiveAPJC

CISCO *Live!*

ALL IN

#CiscoLiveAPJC