

CISCO *Live!*

GO BEYOND

#CiscoLiveAPJC



Transform your Network Automation Apps with GenAI

Gabi Zapodeanu
Principal Technical Marketing Engineer,
Cisco Systems
@zapodeanu
DEVNET-3419

Cisco Webex App

Questions?

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

How

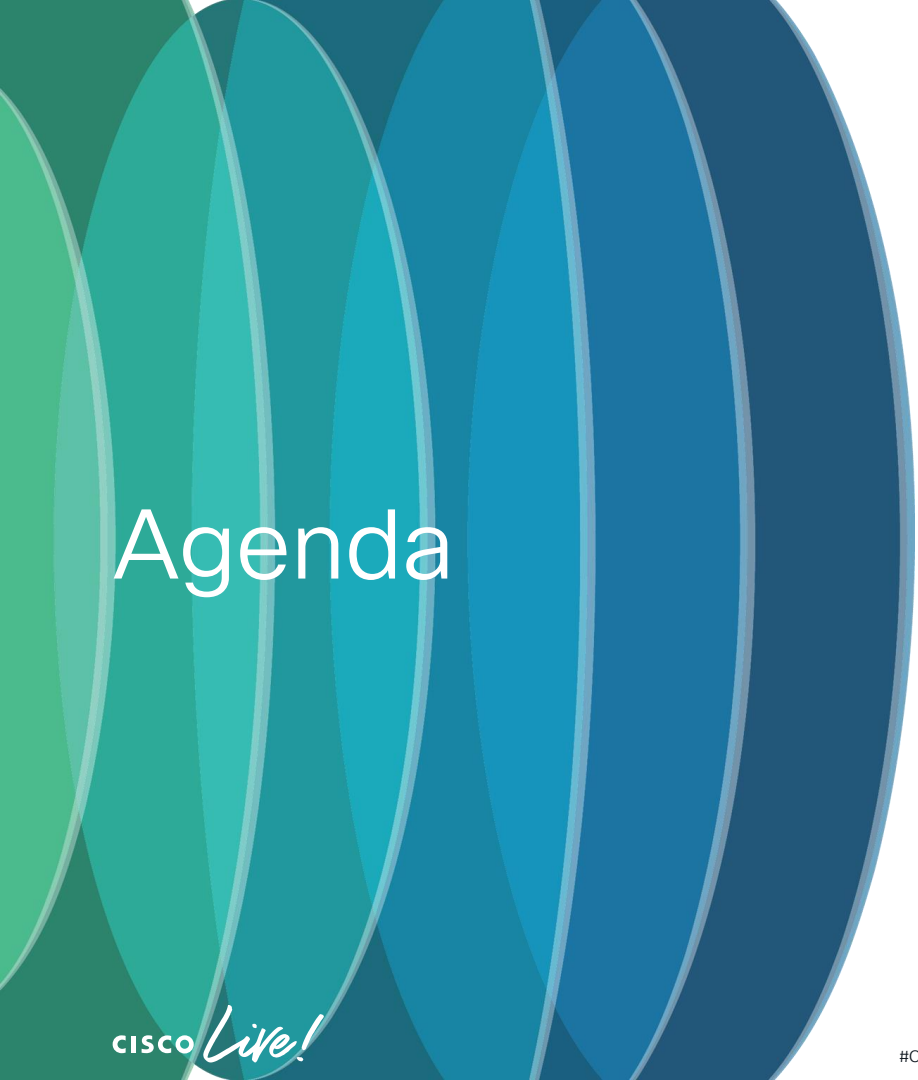
- 1 Find this session in the Cisco Live Mobile App
- 2 Click “Join the Discussion”
- 3 Install the Webex App or go directly to the Webex space
- 4 Enter messages/questions in the Webex space

Webex spaces will be moderated by the speaker until November 15, 2024.



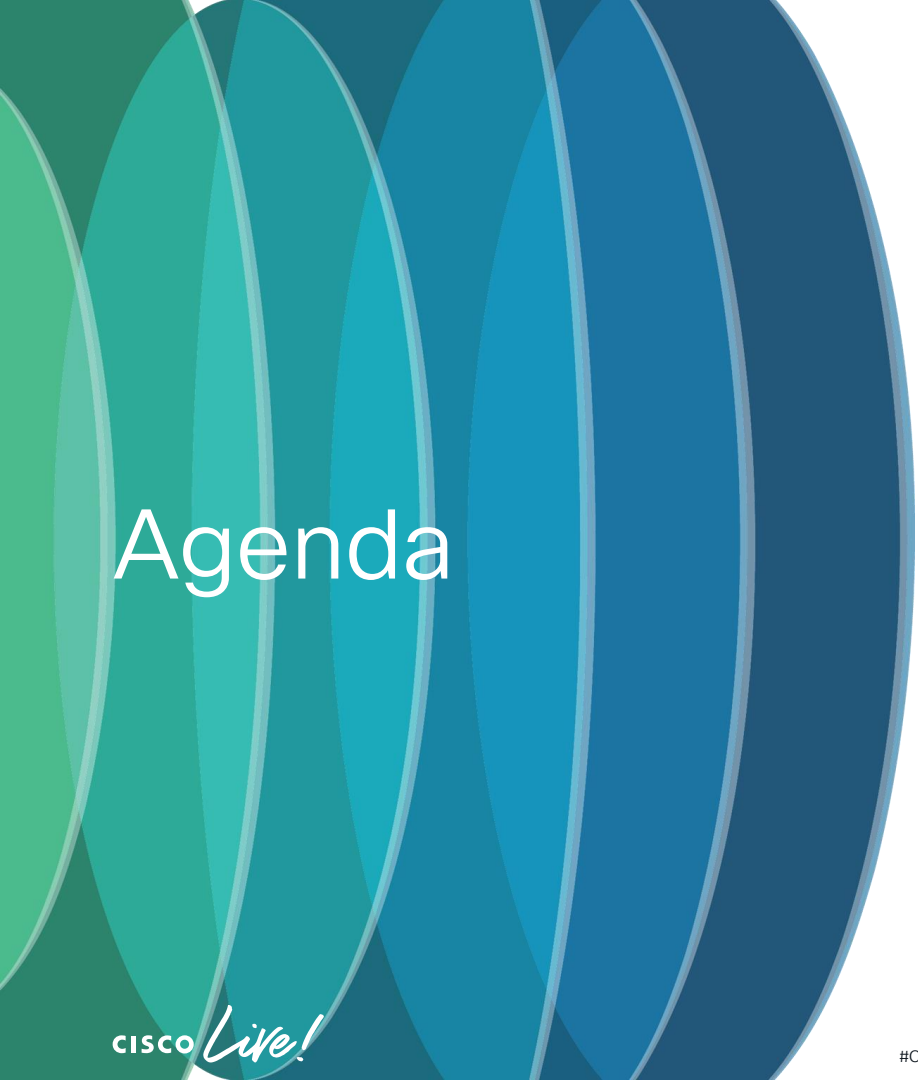
<https://ciscolive.ciscoevents.com/ciscolivebot/#DEVNET-3419>





Agenda

- Catalyst Center Platform Overview
- Integrations
- GenAI Essentials
- AI Code Generation
- GenAI Network Automation
- Developers Resources
- Summary



Agenda

- Catalyst Center Platform Overview
- Integrations
- GenAI Essentials
- AI Code Generation
- GenAI Network Automation
- Developers Resources
- Summary

Catalyst Center Platform Vision



Efficient

Managing infrastructure as code while lowering OPEX



Consistent

Repeatable and precise intent-based outcomes



Scalable

Able to reduce time and effort with large network automation requirements



Secure

Secure APIs and integrations



Sustainable

Integrate sustainable devices and systems



Smart

260+ real time event notifications and AI/ML capabilities

The Value of Catalyst Center Platform



Automate

Implement changes
from enterprise-sized
deployments to
minor adjustments



Integrate

Unlock new insights,
streamline business
workflows through
integration into Cisco
and 3rd Party Platforms

Alerting/notifications

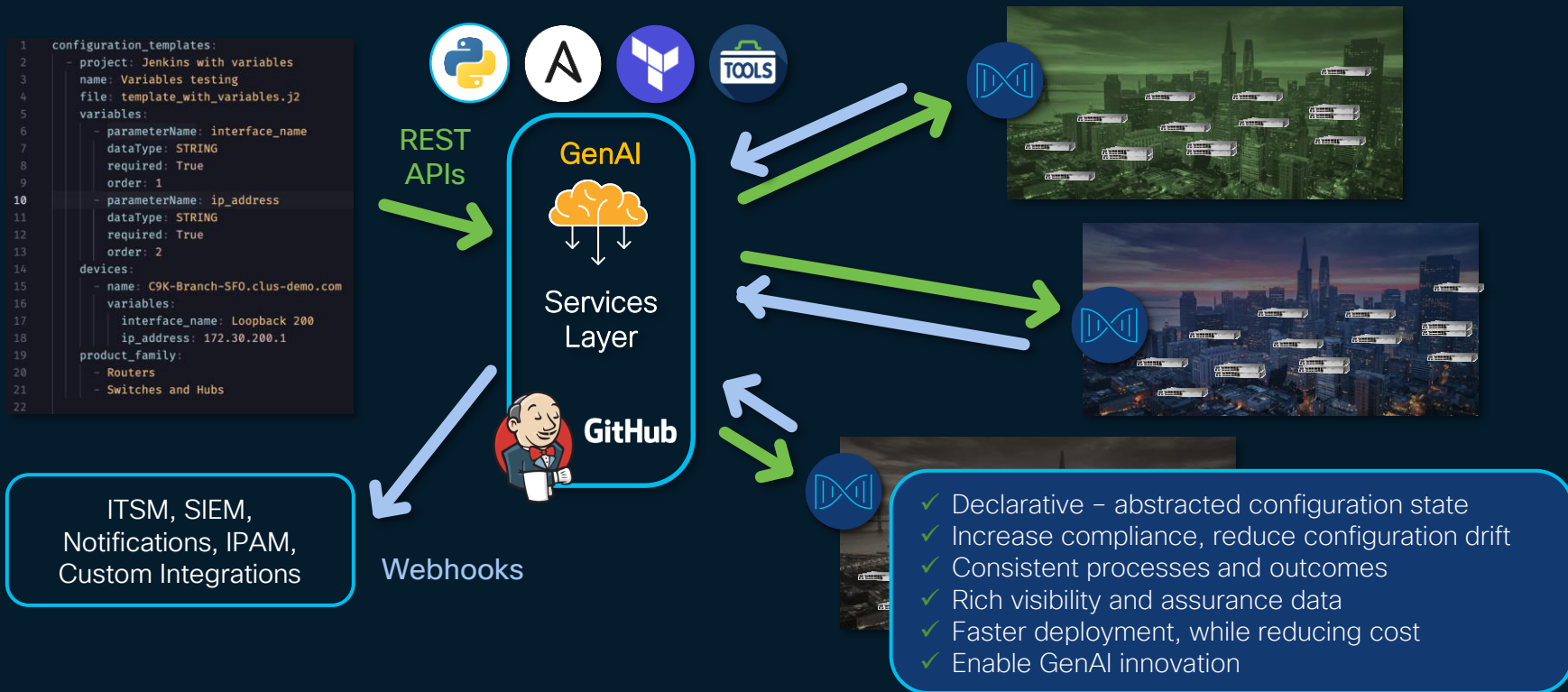
Real-time
communications providing
situational awareness and
enabling data-driven
decisions



Business process streamlining

Translate business process
into multiplatform
automation workflows

Multi Catalyst Center/Controller Managed Networks



Catalyst Center Platform

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

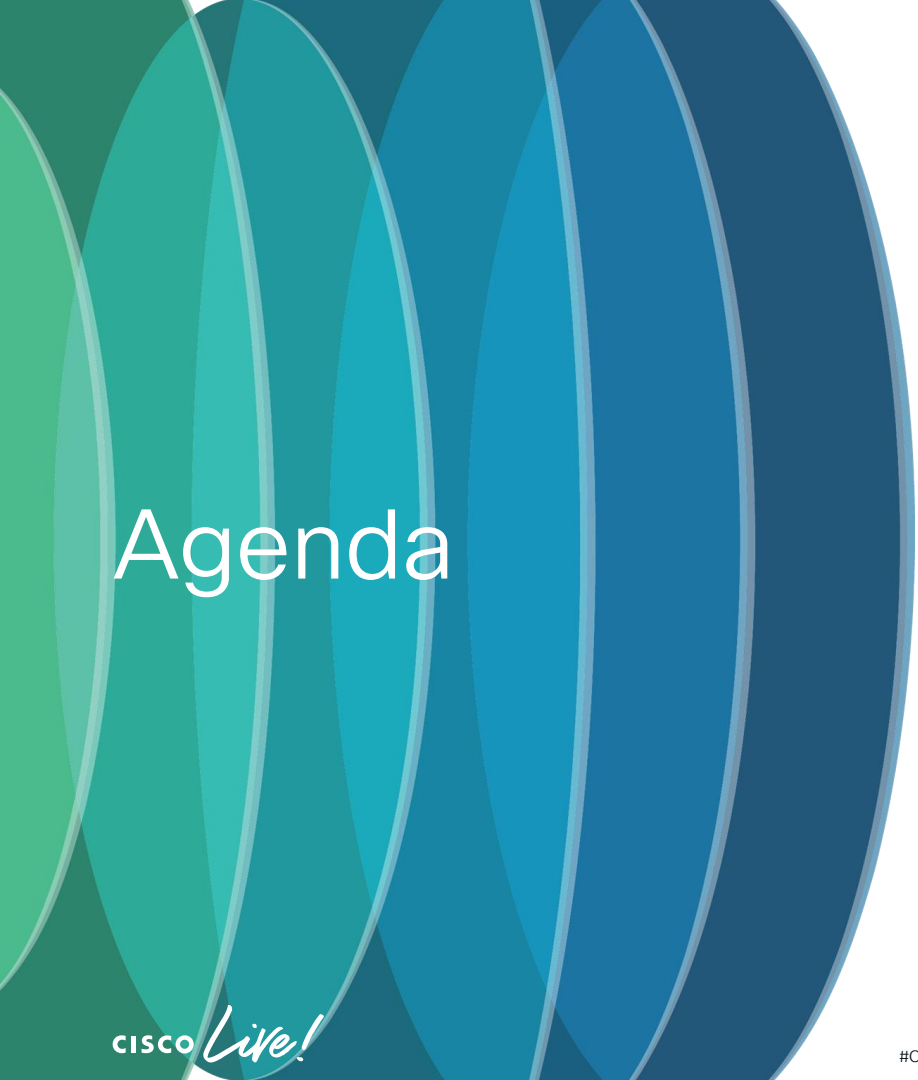
IT Ecosystem Integrations

- IT Service Management
- IP Address Management
- Reporting
- Wireless Planning
- Alerting
- Network Insights

Developer Resources

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

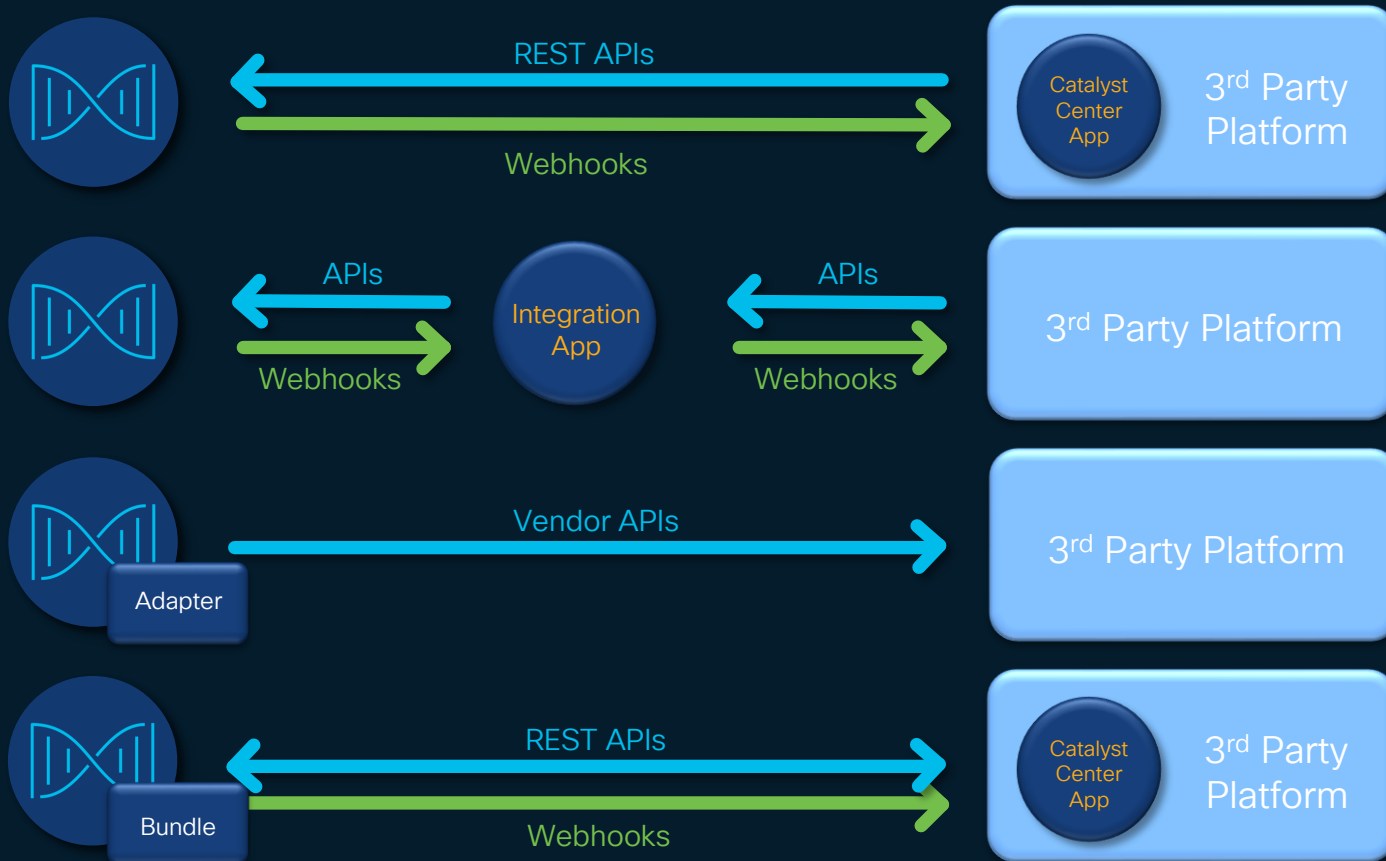




Agenda

- Catalyst Center Platform Overview
- Integrations
- GenAI Essentials
- AI Code Generation
- GenAI Network Automation
- Developers Resources
- Summary

Catalyst Center Integration Options



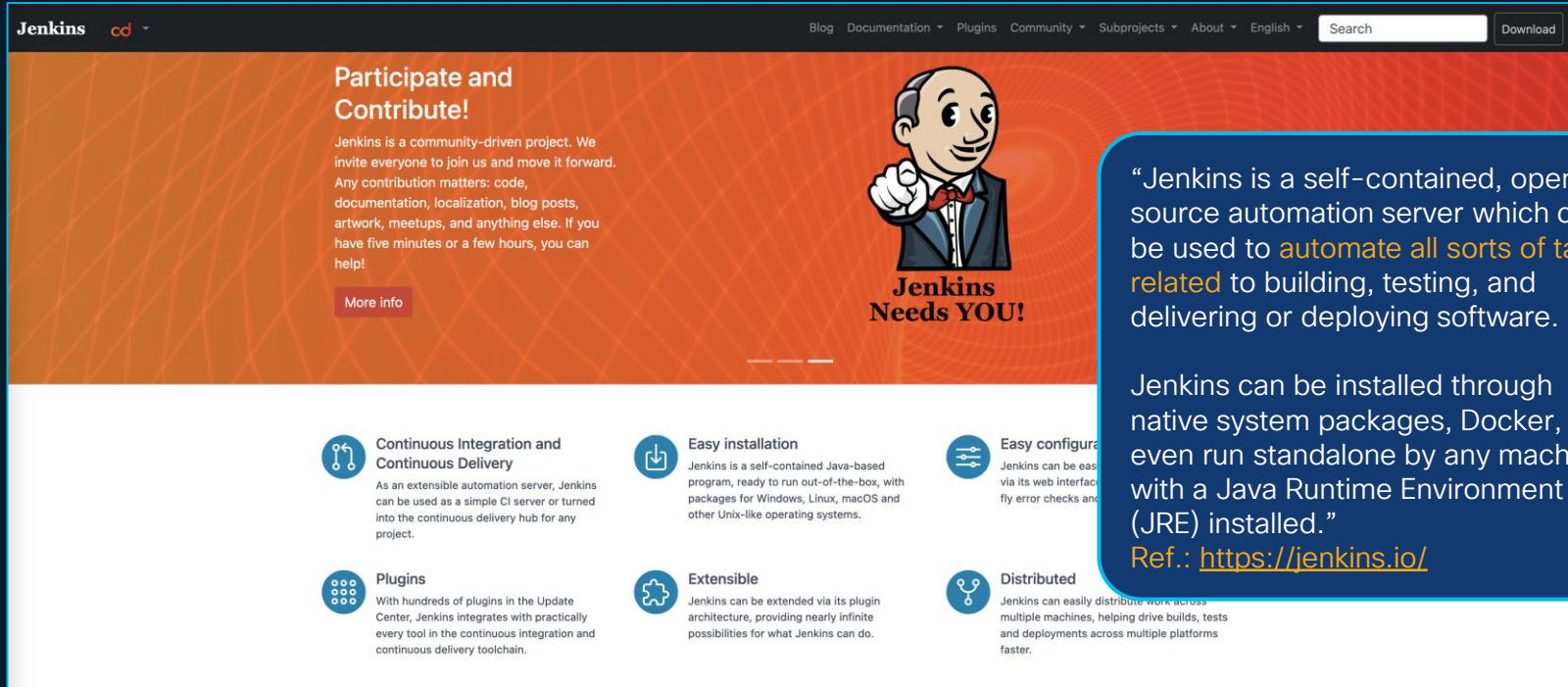
Integration Apps

- Rich features, innovation and maximum extensibility
- Enables GenAI use cases
- Supports non-matching API architectures, auth options, data encoding and data structures
- Ability to build custom integrations with any platforms
- May be required as a proxy between Catalyst Center and cloud platforms

The device state and configuration data, used for our GenAI Bot, is collected using API calls to Catalyst Center.



What is Jenkins?



The screenshot shows the Jenkins homepage. At the top, there's a navigation bar with links for Blog, Documentation, Plugins, Community, Subprojects, About, and English, along with a search bar and a Download button. The main header features the Jenkins logo and the text "Participate and Contribute!". Below this, a cartoon character of a man in a suit and bow tie points towards the text "Jenkins Needs YOU!". To the left of the character, there's a paragraph about Jenkins being a community-driven project and a "More info" button. Below the header, there are six feature cards: "Continuous Integration and Continuous Delivery", "Easy installation", "Easy configuration", "Plugins", "Extensible", and "Distributed". A blue callout box on the right contains a quote about Jenkins being a self-contained, open-source automation server and a reference link to <https://jenkins.io/>.

Participate and Contribute!

Jenkins is a community-driven project. We invite everyone to join us and move it forward. Any contribution matters: code, documentation, localization, blog posts, artwork, meetups, and anything else. If you have five minutes or a few hours, you can help!

[More info](#)

Jenkins Needs YOU!

Continuous Integration and Continuous Delivery

As an extensible automation server, Jenkins can be used as a simple CI server or turned into the continuous delivery hub for any project.

Easy installation

Jenkins is a self-contained Java-based program, ready to run out-of-the-box, with packages for Windows, Linux, macOS and other Unix-like operating systems.

Easy configuration

Jenkins can be easily configured via its web interface, with automatic error checks and feedback.

Plugins

With hundreds of plugins in the Update Center, Jenkins integrates with practically every tool in the continuous integration and continuous delivery toolchain.

Extensible

Jenkins can be extended via its plugin architecture, providing nearly infinite possibilities for what Jenkins can do.

Distributed

Jenkins can easily distribute work across multiple machines, helping drive builds, tests and deployments across multiple platforms faster.

“Jenkins is a self-contained, open-source automation server which can be used to **automate all sorts of tasks related** to building, testing, and delivering or deploying software.

Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed.”

Ref.: <https://jenkins.io/>

Why Catalyst Center and Jenkins?

Runbook Automation



+



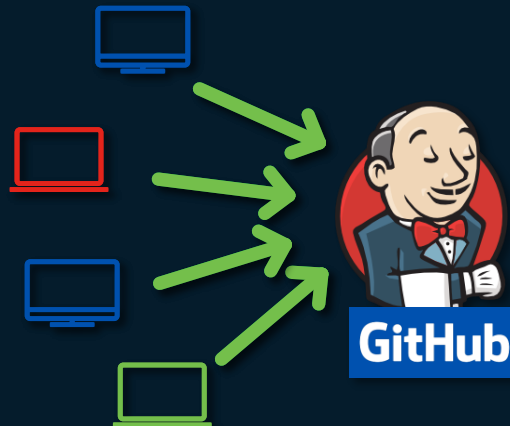
Jenkins is a self-contained, open-source **automation server** which may be used to:

- Support network management as Infrastructure-as-Code
- Integrate with GitHub for validated and approved apps
- Integrate with GitHub for intent configurations, profiles and policies
- Pipelines may include lab testing, approval and production deployment

Jenkins Pipeline Build

- Trigger Pipeline build using APIs
- Create a Docker container
- Clone, or pull, code from Version Control Platforms: GitHub or GitLab
- Install or upgrade to the **desired versions** of Python Libraries – Catalyst Center SDK, Ansible, Terraform
- Provide the required params, or pull them from Version Control
- Run the application

Note:
All steps are completely automated



Runbook environments identical, always using the latest code versions and libraries
Rich scheduling/triggering options

Jenkins Dashboard

- Jenkins preferences
- Group pipelines in views
- Build queue and execution status
- Pipeline stats – health, number of runs, success and failure, ...

Dashboard > OpsView >

Delete View

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

Open Blue Ocean

Build Queue

No builds in the queue.

Build Executor Status

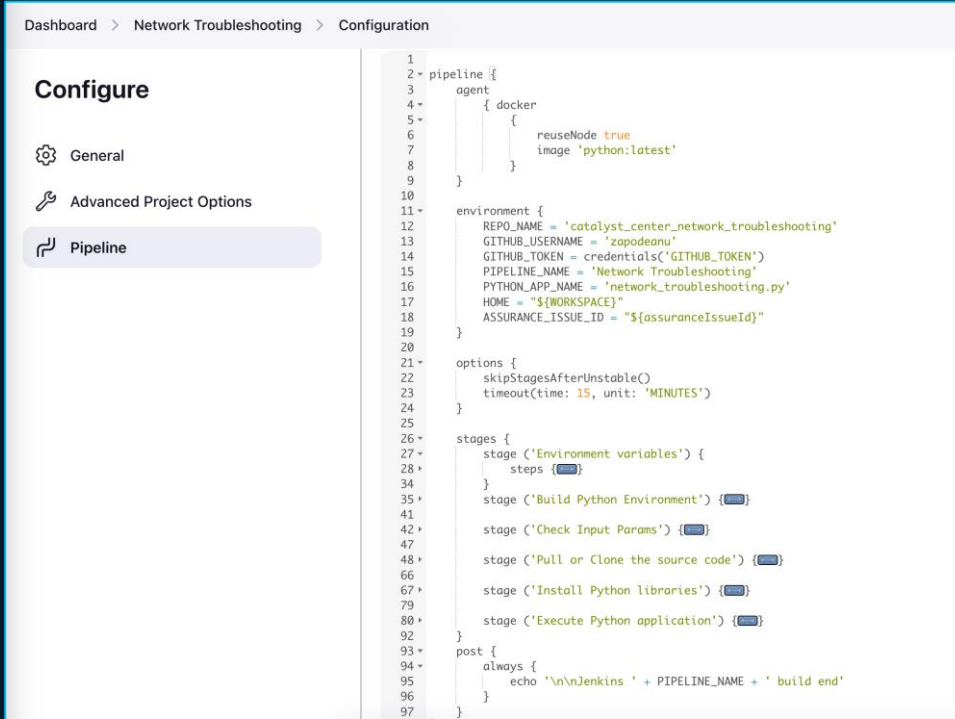
1 Idle

2 Idle

3 Idle

		Catalyst Center State Sync	18 hr #188	4 days 18 hr #184		
		CLI Templates Deployment	10 mo #18	1 yr 1 mo #13		
		Device Inventory	18 days #1841	17 hr #1859		
		GitHub Sync	17 hr #455	4 days 17 hr #451	31 sec	
		GitLab Sync	18 hr #409	4 days 18 hr #405	22 sec	
		Intent Device Compliance	19 hr #151	N/A	39 sec	
		ITSM PnP Workflow	3 mo 16 days #14	3 mo 16 days #12	3.2 sec	
		ITSM Provision Switch	3 days 0 hr #39	1 mo 19 days #38	41 sec	
		Network Troubleshooting	1 day 4 hr #52	1 day 5 hr #48	1 min 22 sec	
		PoE Sustainability	1 yr 1 mo #56	N/A	16 sec	
		SDA-as-Code	11 mo #8	N/A	25 sec	
		Trigger Assurance Issues	11 hr #628	4 days 11 hr #620	7 min 53 sec	

Jenkins Pipeline



```
1 pipeline {
2   agent
3     { docker
4       {
5         reuseNode true
6         image 'python:latest'
7       }
8     }
9
10  environment {
11    REPO_NAME = 'catalyst_center_network_troubleshooting'
12    GITHUB_USERNAME = 'zapodeanu'
13    GITHUB_TOKEN = credentials('GITHUB_TOKEN')
14    PIPELINE_NAME = 'Network Troubleshooting'
15    PYTHON_APP_NAME = 'network_troubleshooting.py'
16    HOME = "${WORKSPACE}"
17    ASSURANCE_ISSUE_ID = "${assuranceIssueId}"
18  }
19
20  options {
21    skipStagesAfterUnstable()
22    timeout(time: 15, unit: 'MINUTES')
23  }
24
25  stages {
26    stage ('Environment variables') {
27      steps {
28        [ ]
29      }
30    }
31    stage ('Build Python Environment') { [ ] }
32    stage ('Check Input Params') { [ ] }
33    stage ('Pull or Clone the source code') { [ ] }
34    stage ('Install Python libraries') { [ ] }
35    stage ('Execute Python application') { [ ] }
36  }
37
38  post {
39    always {
40      echo '\n\nJenkins ' + PIPELINE_NAME + ' build end'
41    }
42  }
43 }
```

- Jenkinsfile is a text file representing the definition of a pipeline. It includes multiple stages and steps.
- Steps may be constructed using:
 - Shell commands
 - Run code developed using the Python SDK, Ansible or Terraform
 - Jenkins Plugins
- Environment includes all params required by the pipeline

Details: Cisco Live US 2023- Jenkins Automations for Cisco DNA Center – DEVNET-2151

Pipeline Status

Dashboard > OpsView > Network Troubleshooting >

Status

Changes

Build with Parameters

Configure

Delete Pipeline

Full Stage View

Open Blue Ocean

Rename

Pipeline Syntax

Build History

trend

Filter builds...

#52 May 9, 2024, 10:47 AM

#51 May 9, 2024, 10:41 AM

#50 May 9, 2024, 10:36 AM

#49 May 9, 2024, 10:23 AM

Network Troubleshooting

This pipeline is for the Network Troubleshooting automation workflow, issue event notification received

Stage View

	Environment variables	Build Python Environment	Check Input Params	Pull or Clone the source code	Install Python libraries	Execute Python application	Declarative: Post Actions
Average stage times: (Average full run time: ~1min 0s)	89ms	67ms	64ms	1s	6s	50s	62ms
#52 May 09 10:47 No Changes	83ms	93ms	65ms	1s	6s	1min 11s	65ms
#51 May 09 10:41 No Changes	103ms	71ms	57ms	1s	6s	1min 11s	60ms
#50 May 09 10:36 No Changes	82ms	63ms	59ms	1s	6s	1min 11s	60ms

Stage view:

- create container
- validate input params
- pull code from GitHub
- install libraries
- execute troubleshooting app

Jenkins Server – Enable Remote API Access

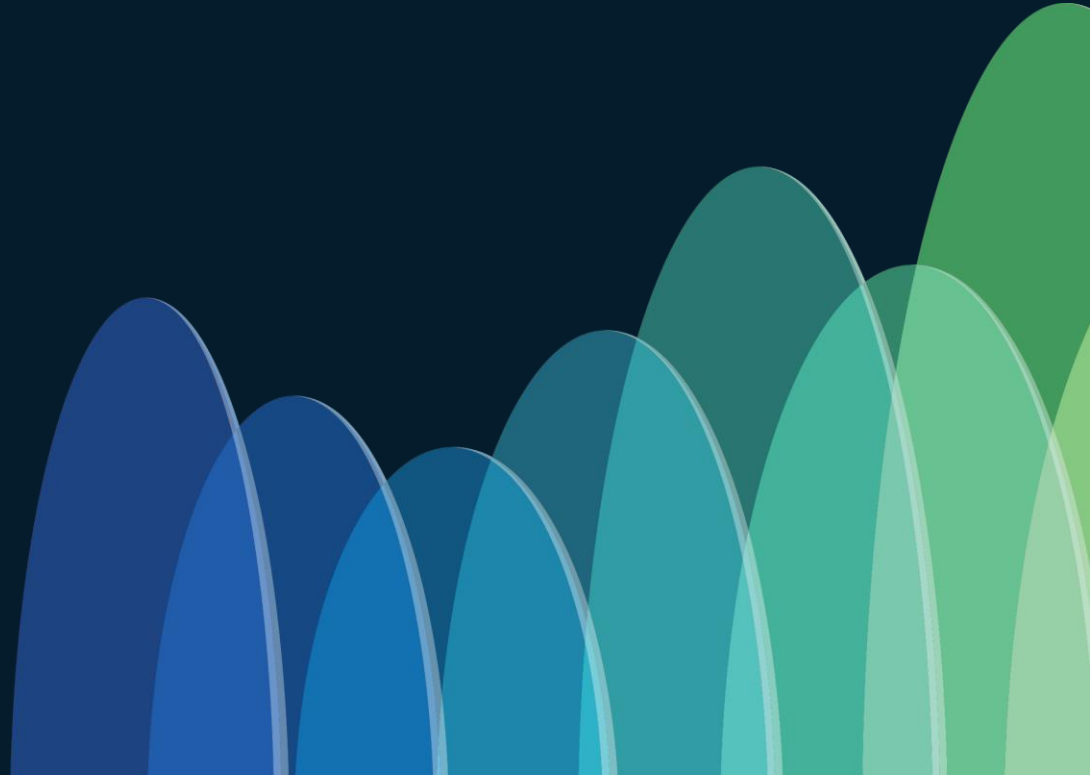
- Add new API Auth token
 - API Authentication is using username and the API token.
- Configure pipeline to use input params or input file
- Configure the build trigger for remote access API
- REST API URL call to Jenkins:

/Jenkins_server/job/pipeline_name/
buildWithParameters/{parameters}

The screenshot shows the Jenkins configuration page for a user named 'Gabriel Zapodeanu'. The page is divided into several sections:

- API Token:** A table showing current tokens. One token named 'API_TOKEN' is highlighted, showing it was created 3 days ago and used 33 times.
- String Parameter:** A section for defining parameters. A parameter named 'assuranceissueid' is shown with a default value of 'Assurance Issue Id'.
- Build Triggers:** A section for configuring build triggers. The 'Trigger builds remotely (e.g., from scripts)' option is checked, and the 'Authentication Token' is set to 'API_TOKEN'.

Demo Jenkins Pipelines



Camtasia 2023 File Edit Modify Text View Export Window Help

01:14:08:21 110% Mon May 27 12:51 PM

Cisco Catalyst Center Dashboard [jenkins]

https://10.93.141.47:8443

Jenkins

Search (⌘+K)

1 1 Gabriel Zapodeanu log out

Dashboard

+ New Item

People

Build History

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

Open Blue Ocean

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

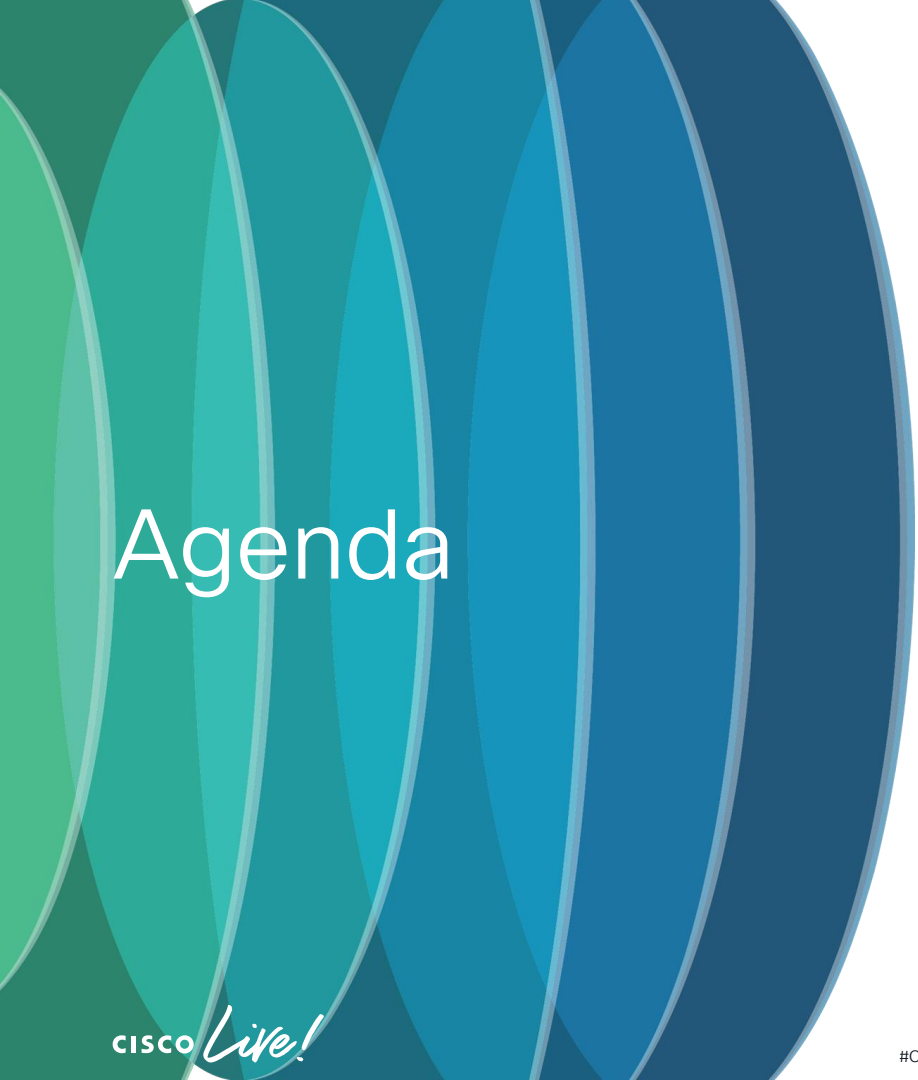
2 Idle

3 Idle

All LabView OpsView +

S	W	Name ↓	Last Success	Last Failure	Last Duration	Fav
✓	☀	Catalyst Center State Sync	16 hr #205	N/A	50 sec	▶ ☆
✓	☀	ChatGPT Plugin	11 mo #11	N/A	13 sec	▶ ☆
✓	☀	Cisco DNA Center Jenkins Sandbox	1 yr 0 mo #1	N/A	3.8 sec	▶ ☆
✓	☁	CLI Templates Deployment	11 mo #18	1 yr 1 mo #13	2 min 50 sec	▶ ☆
✓	☀	Dev Ready	8 mo 4 days #61	N/A	2 min 40 sec	▶ ☆
✓	☁	Device Inventory	14 hr #1876	1 day 14 hr #1875	51 sec	▶ ☆
✓	☀	GitHub Sync	15 hr #472	N/A	29 sec	▶ ☆
✗	☁	GitLab Sync	14 days #412	15 hr #426	21 sec	▶ ☆
✓	☀	Intent Device Compliance	16 hr #168	N/A	41 sec	▶ ☆

CISCO Live!



Agenda

- Catalyst Center Platform Overview
- Integrations
- GenAI Essentials
- AI Code Generation
- GenAI Network Automation
- Developers Resources
- Summary

AI, ML, LLM, Function Calling, GenAI, RAG, ... ?

- **Artificial Intelligence (AI)** – use of data and algorithms to allow systems to perform tasks similar with human intelligence
 - **Machine Learning (ML)** – AI subset, focused on performing steps without being explicitly programmed. Learn from data patterns and insights to create recommendations
 - **Large Language Model (LLM)** – neural network model designed to understand and generate human language
- **Function calling** – Use LLM models to interface with external platforms (usually with API calls), to identify and perform tasks or retrieve data
 - **Generative AI (GenAI)** – Use patterns in existing data to create new content. This content could be text, numbers or multimedia
 - **Retrieval-Augmented Generation (RAG)** – Retrieves matches in provided dataset and uses NLPs to create high-confidence content

Large Language Model (LLM) – Do I need one?

LLMs:

- GPT (OpenAI)
- Gemini (Google)
- Llama 3 (Meta)
- Ollama (framework)
- Cohere
- ...

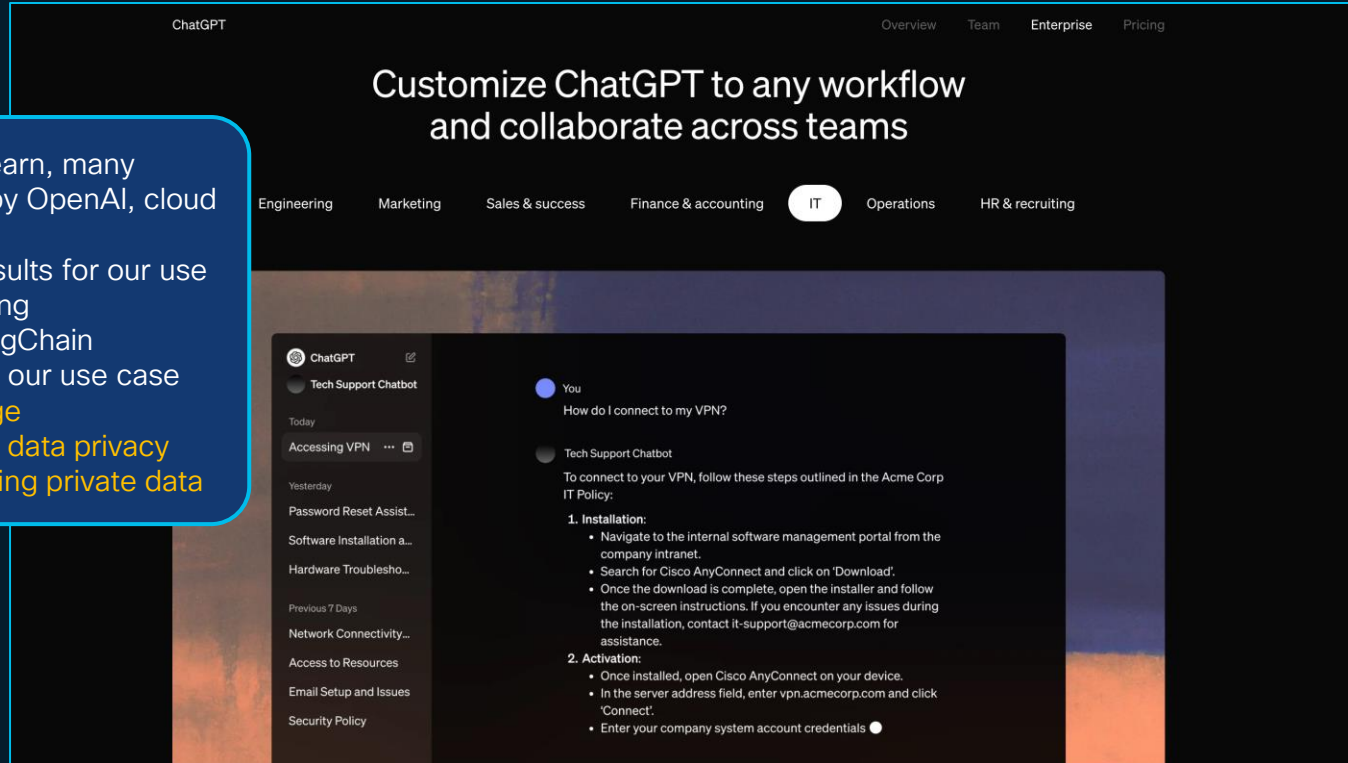
Consider cost and data privacy

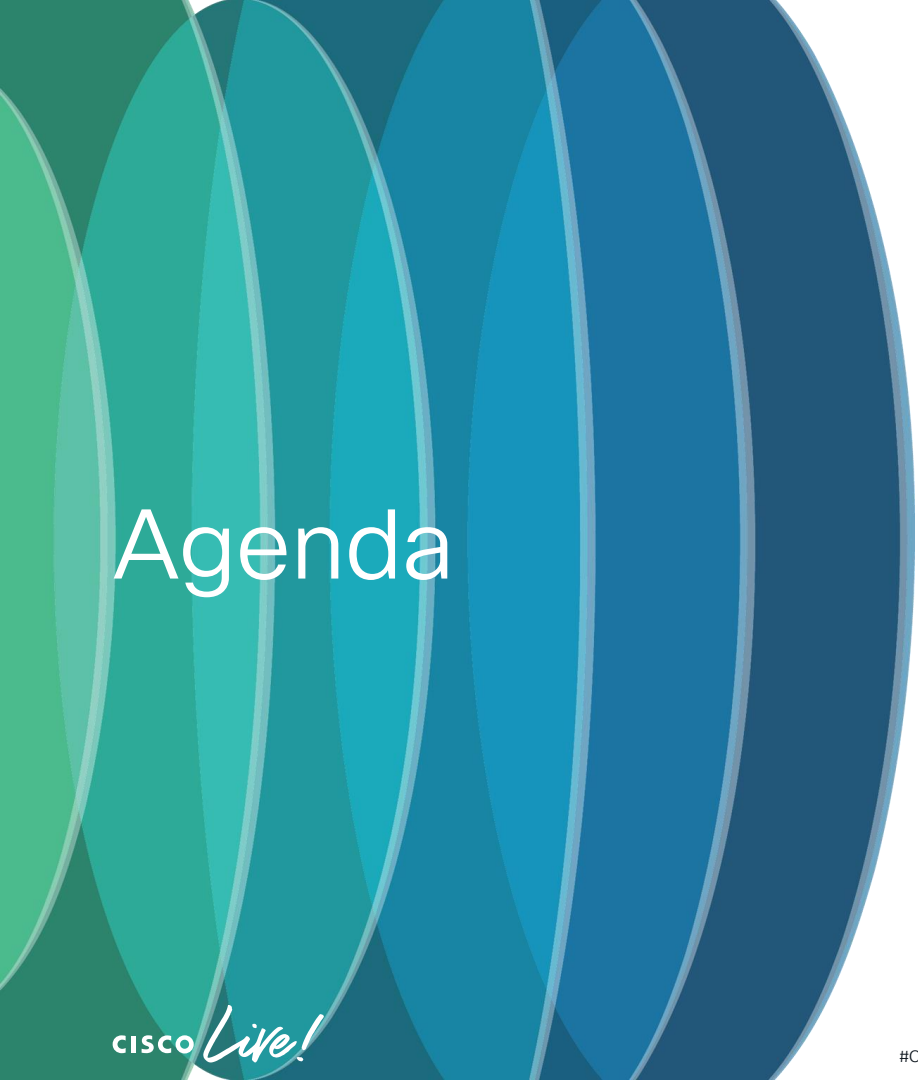
- Large Language Model is an AI Model designed to understand and generate human language
- Will answer our query, using natural language, based on the similarity search results
- Trained on massive data sets (billions of text chunks)
- Understand context and are **general-purpose**
- Provide other features, like function calling
- Support code development of network automation tasks

ChatGPT LLM (Generative Pre-trained Transformer)

<https://openai.com/chatgpt/>

- Very easy to use and learn, many examples, developed by OpenAI, cloud based
- Versatile, very good results for our use case, without any training
- Integration through LangChain
- Efficient, works well for our use case
- Cost may be a challenge
- Follow your policies for data privacy and be aware of exposing private data



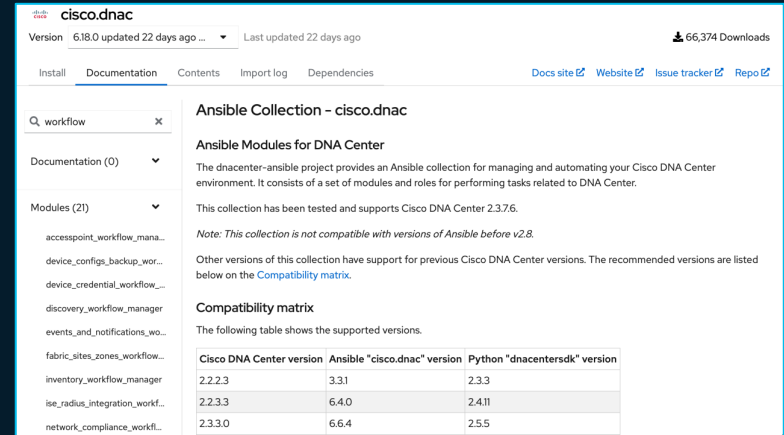


Agenda

- Catalyst Center Platform Overview
- Integrations
- GenAI Essentials
- AI Code Generation
- GenAI Network Automation
- Developers Resources
- Summary

Catalyst Center IaC Platforms

- Most customers use Python SDK
- Ansible Collection and Terraform Provider are very well adopted
- Customers developed network-as-code workflows in Go SDK, JavaScript, and other programming languages
- CI/CD and Runbook Platforms are very common



Ansible Collection - cisco.dnac

Version 6.18.0 updated 22 days ago ... Last updated 22 days ago 66,374 Downloads

Install Documentation Contents Import log Dependencies Docs site Website Issue tracker Repo

Q workflow x

Documentation (0) v

Modules (21) v

- accesspoint_workflow_mana...
- device_configs_backup_wor...
- device_credential_workflow...
- discovery_workflow_manager
- events_and_notifications_wo...
- fabric_sites_zones_workflow...
- inventory_workflow_manager
- ise_radius_integration_workf...
- network_compliance_workfl...

Ansible Modules for DNA Center

The dnacenter-ansible project provides an Ansible collection for managing and automating your Cisco DNA Center environment. It consists of a set of modules and roles for performing tasks related to DNA Center.

This collection has been tested and supports Cisco DNA Center 2.3.7.6.

Note: This collection is not compatible with versions of Ansible before v2.8.

Other versions of this collection have support for previous Cisco DNA Center versions. The recommended versions are listed below on the [Compatibility matrix](#).

Compatibility matrix

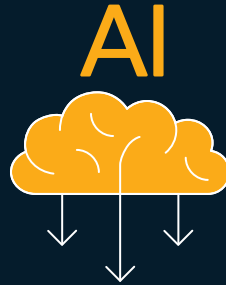
The following table shows the supported versions.

Cisco DNA Center version	Ansible "cisco.dnac" version	Python "dnacentersdk" version
2.2.2.3	3.31	2.3.3
2.2.3.3	6.4.0	2.4.11
2.3.3.0	6.6.4	2.5.5

Why AI Code Generation for Catalyst Center



+



AI may be able to:

- Suggest recommendations and able to search documentation
- Accelerate code development for your IaC Catalyst Center workflows
- Multi-language support
- Know or understand the available APIs or able to use swagger files

Always verify the recommended code, check the order of operations and test the recommendations before using them.

Python Code Generation

ChatGPT 4o ▾

Can you write a Python code to create a function that deploys a CLI template to a device using Catalyst Center REST APIs?



Here is an example Python function using `dnacentersdk` to deploy a CLI template to a device using Cisco Catalyst Center (DNA Center) REST APIs:

Steps the function performs:

1. **Authenticate with Catalyst Center** using `dnacentersdk`.
2. **Find the target device** by its IP or device ID.
3. **Find the CLI template** by its name or ID.
4. **Deploy the CLI template** to the target device using the API call.

Here's the code that does this:

- The ask is to write code in Python
- ChatGPT recommendation is using the “dnacentersdk” library
- The suggested code has an understanding of the workflow, and the APIs called to obtain the required parameters
- It is aware of the rebranding of Cisco DNA Center to Catalyst Center

Python Code Generation (continued)

Here's the code that does this:

python

```
from dnacentersdk import *
import time

def deploy_cli_template(
    """
    Deploy a CLI template to a device.

    :param api: Authenticated API object
    :param device_ip: IP address of the device
    :param template_name: Name of the template
    :param params: Dictionary of parameters for the template
    :return: Status of the deployment
    """

    # Step 1: Get the device list
    devices = api.device_programmer.get_device_list()
    if not devices['result']:
        raise Exception("No devices found")

    device_id = devices['result'][0]['id']
    print(f"Found device {device_id}")
```

```
# Step 2: Get the template
templates = api.template_programmer.get_templates_available()
template = next((tm for tm in templates if tm['name'] == template_name), None)

if not template:
    raise Exception("Template not found")

template_id = template['id']
print(f"Found template {template_id}")

# Step 3: Deploy the template
deployment_data = {
    "targetInfo": [
        {
            "id": device_id,
            "type": "device",
            "params": {}
        }
    ],
    "templateId": template_id
}
```

```
# Step 4: Check the deployment status until it completes
status = "IN_PROGRESS"
while status == "IN_PROGRESS":
    deployment_status = api.template_programmer.get_template_deployment_status(deployment_id)
    status = deployment_status['status']
    print(f"Deployment status: {status}")
    time.sleep(5)

return status

# Example usage:
# Define API credentials
api = DNACenterAPI(hostname, username, password)

# Call the function
device_ip = "10.1.1.1"
template_name = "BasicConfig"
params = {
    "interfaceName": "GigabitEthernet0/0/0",
    "description": "Configured by Python script"
}

status = deploy_cli_template(api, device_ip, template_name, params)
print(f"Deployment completed with status: {status}")

deployment_response = api.template_programmer.deploy_template(deployment_data)
deployment_id = deployment_response['result']['deploymentId']
print(f"Deployment started with ID: {deployment_id}")
```

Copy code

Key Points:

- **Device Lookup:** It searches for the device using the `get_device_list` API based on the provided IP address.
- **Template Lookup:** It retrieves all templates available using `gets_the_templates_available` and searches for the one by name.
- **Deployment:** The template is deployed to the specified device using the `deploy_template` API.
- **Polling Status:** The function polls the deployment status until it completes or fails.

Ensure the `dnacentersdk` is installed, and the API credentials are correctly set up. This function can be extended with error handling for production use.

Let me know if you need further customization!

🔊 📄 📌 🔄 ⌵

ChatGPT knows the order of operations: getting device Id, template Id, payload sample if params are required when deploying the template, and polling for status (async API).

Python Code Generation (end)

Can you do the same in Python, do not use the Catalyst Center SDK?



Certainly! If you prefer not to use the Catalyst Center SDK, you can use the `requests` module to interact with the DNA Center REST API via a Python script to deploy a CLI template.

Key steps:

1. Authenticate with Cisco DNA Center
2. Find the target device and template
3. Retrieve the CLI template
4. Deploy the CLI template
5. Monitor deployment status

```
def get_device_by_ip(self, device_ip):
    """
    Get device info by IP address.
    """
    url = f"{self.base_url}/dna/intent/api/v1/intent/{device_ip}"
    response = requests.get(url, headers=self.headers)
    response.raise_for_status()
    return response.json()['response']

def get_template_by_name(self, template_name):
    """
    Get template info by name.
    """
    url = f"{self.base_url}/dna/intent/api/v1/template/{template_name}"
    response = requests.get(url, headers=self.headers)
    response.raise_for_status()
    templates = response.json()
    for template in templates:
        if template['name'] == template_name:
            return template
    raise Exception(f"Template '{template_name}' not found.")
```

Example usage:

```
if __name__ == "__main__":
    # Set your DNA Center details here
    base_url = "https://dnacenter_url" # Replace with your DNA Center URL
    username = "your_username" # Replace with your DNA Center username
    password = "your_password" # Replace with your DNA Center password

    # Device and template details
    device_ip = "10.1.1.10" # Replace with the device IP
    template_name = "Basic_Configuration" # Replace with the template name
    params = {
        "interfaceName": "GigabitEthernet1", # Example parameter for template
        "description": "Configured by raw API"
    }

    # Deploy template and print the result
    status = deploy_cli_template(base_url, username, password, device_ip, template_name, params)
    print(f"Deployment completed with status: {status}")
```

Copy code

Explanation:

1. `get_token`: Authenticates to DNA Center and retrieves an access token.
2. `get_device_by_ip`: Fetches device information by IP address.

Upgrade Code to Current Library Version

ChatGPT 4o ▾

Can you rewrite the deploy template using the Python SDK located here: <https://github.com/cisco-en-programmability/dnacentersdk> and use version 2.3.7.6?

< 2/2 >



To rewrite the deployment template using the **Cisco DNA Center Python SDK** from the [GitHub repository](#) and using **version 2.3.7.6**, you need to ensure that you are working with the right version of the SDK that matches your Cisco DNA Center version.

Here's the updated Python script using `dnacentersdk 2.3.7.6` for template deployment:

Installation:

First, clone and install the SDK from GitHub.

bash

Copy code

```
git clone https://github.com/cisco-en-programmability/dnacentersdk
cd dnacentersdk
pip install .
```



Python Code for Deploying Template:

Does it Run Correctly? Almost

AI Generated

```
# Step 1: Get the device by IP address
devices = api.devices.get_device_list(ip_address=device_ip)
if not devices['response']:
    raise Exception(f"Device with IP {device_ip} not found.")
```

```
device_id = devices['response'][0]['id']
```

```
print(f"Found template with ID: {template_id}")
```

```
# Step 3: Deploy the CLI template to the device
```

```
deployment_data = {
```

```
    "targetInfo": [
```

```
        {
```

```
            "id": device_id,
```

```
            "type": "MANAGED_DEVICE_IP",
```

```
            "params": params
```

```
        }
```

```
    ],
```

```
    "templateId": template_id
```

```
}
```

```
deployment_response = api.configuration_templates.deploy_template_v2(
```

```
    deployment_data)
```

```
deployment_id = deployment_response['response']['taskId']
```

```
print(f"Deployment started with ID: {task_id}")
```

```
# Step 4: Check the deployment status until it completes
```

```
status = "IN_PROGRESS"
```

```
while status == "IN_PROGRESS":
```

```
    deployment_status =
```

```
        api.configuration_templates.get_template_deployment_status(
```

```
            deployment_id=deployment_id)
```

```
        status = deployment_status['status']
```

```
        print(f"Deployment status: {status}")
```

```
        time.sleep(5)
```

```
Found device with ID: 657008e6-6160-481e-aa7d-a5b3513b1f7c
```

```
Found template with ID: 73aab375-30ef-4a41-90de-a110a072ea12
```

```
Deployment started with ID: 01929cd6-f759-76c5-828e-9ac647ad971b
```

```
Deployment status: is error None
```

```
Deployment completed with status: success
```

Updated

```
# Step 1: Get the device by IP address
devices = api.devices.get_device_list(management_ip_address=device_ip)
```

```
if not devices['response']:
    raise Exception(f"Device with IP {device_ip} not found.")
```

```
device_id = devices['response'][0]['id']
```

```
print(f"Found template with ID: {template_id}")
```

```
# Step 3: Deploy the CLI template to the device
```

```
deployment_data = {
```

```
    "forcePushTemplate": True,
```

```
    "targetInfo": [
```

```
        {
```

```
            "id": device_id,
```

```
            "type": "MANAGED_DEVICE_UUID"
```

```
        }
```

```
    ],
```

```
    "templateId": template_id
```

```
}
```

```
deployment_response = api.configuration_templates.deploy_template_v2(
```

```
    deployment_data)
```

```
task_id = deployment_response['response']['taskId']
```

```
print(f"Deployment started with ID: {task_id}")
```

```
# Step 4: Check the deployment status until it completes
```

```
status = "IN_PROGRESS"
```

```
while status == "IN_PROGRESS":
```

```
    deployment_status = api.task.get_task_by_id(task_id=task_id)
```

```
    status = deployment_status['status']
```

```
    print(f"Deployment status: is error {status}")
```

```
    time.sleep(5)
```

```
return status
```

App Run Using Catalyst Center 2.3.7.6

```
template_id = template['templateId']
print(f"Found template with ID: {template_id}")

# Step 3: Deploy the CLI template to the device
deployment_data = {
    "forcePushTemplate": True,
    "targetInfo": [
        {
            "id": device_id,
            "type": "MANAGED_DEVICE_UUID"
        }
    ],
    "templateId": template_id
}

deployment_response = api.configuration_templates.deploy_template_v2(payload=deployment_data)
task_id = deployment_response['response']['taskId']
print(f"Deployment started with ID: {task_id}")

# Step 4: Check the deployment status until it completes
status = "IN_PROGRESS"
while status == "IN_PROGRESS":
    deployment_status = api.task.get_task_by_id(task_id=task_id)
    status = deployment_status['isError']
    print(f"Deployment status: is error {status}")
    time.sleep(5)

return status
```

```
/Users/gzapodea/PythonCode/DEVNET-3419/venv/bin/python /Users/gzapodea/PythonCode/DEVNET-3419/app.py
Found device with ID: 657008e6-6160-481e-aa7d-a5b3513b1f7c
Found template with ID: 73aab375-30ef-4a41-90de-a110a072ea12
Deployment started with ID: 01929cd6-f759-76c5-828e-9ac647ad971b
Deployment status: is error None
Deployment completed with status: success
```

Update GenAI with Corrected Code

I have an updated code that works for the Python SDK version 2.3.7.6. Yours had some errors

< 2/2 >



Thank you for letting me know. I would be happy to review the updated code that works for the **Python SDK version 2.3.7.6**. Please share your updated version of the code, and I will review it and troubleshoot any issues or further enhance it if needed.

```
from dnacentersdk import DNACenterAPI
import time
```

```
def deploy_cli_template(api, device_ip, template_name):
    """
    Deploy a CLI template to a device using Catalyst Center.
    :param api: Authenticated DNACenterAPI object.
    :param device_ip: IP address of the device where the template is to be deployed.
```



Thank you for sharing the updated code! It looks good, and it follows the proper use of the **Python SDK for Cisco DNA Center version 2.3.7.6**. Let me walk you through the key components and some minor optimizations for clarity:

Key Features:

1. **Device Retrieval:** The code correctly fetches the device information by **management IP address** using `api.devices.get_device_list`.
2. **Template Fetching:** The correct API method is used to retrieve available templates with `api.configuration_templates.gets_the_templates_available()`, and the template ID is selected based on the template name.
3. **Template Deployment:** You are using `deploy_template_v2` with the `forcePushTemplate` option and passing the device ID in the deployment data.
4. **Task Monitoring:** The code checks the deployment task status using `api.task.get_task_by_id` and continues to monitor it in a loop until completion.

Ansible Code Generation

- The ask is to write the same simple workflow in Ansible
- ChatGPT recommendation is using the DNA Center Ansible collection, and it knows the requirement to have the Catalyst Center Python SDK installed

ChatGPT 4o ▾

Can you do the same with Ansible?



Yes, you can deploy a CLI template to a device using **Ansible** and the **Cisco DNA Center Ansible Collection**. Below is an example playbook that performs the same function as the Python script, using Ansible to deploy a template to a device managed by Cisco DNA Center (Catalyst Center).

Prerequisites:

1. Install the **Cisco DNA Center Ansible Collection**:

```
bash
```

Copy code

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

2. Ensure you have the required Python libraries (`dnacentersdk` and others) and a working Ansible environment.

Ansible Code Generation (continued)

Ansible Playbook Example:

```
yaml
---
- name: Deploy CLI template
  hosts: localhost
  gather_facts: no
  tasks:

  - name: Get the device b
    cisco.dnac.device_info
    dnac_host: "{{ dnac_
    dnac_username: "{{ d
    dnac_password: "{{ d
    dnac_verify: no
    state: query
    ip_address: "{{ devi
    register: device_info

  - name: Fail if the devi
    fail:
      msg: "Device with IP
    when: device_info.resp
```

```
- name: Get the CLI template info
  cisco.dnac.template_info
  dnac_host: "{{ dnac_
  dnac_username: "{{ dn
  dnac_password: "{{ dn
  dnac_verify: no
  state: query
  name: "{{ templ
  register: template_info

- name: Wait for deployment
  cisco.dnac.template_deploy
  dnac_host: "{{ dnac_
  dnac_username: "{{ dn
  dnac_password: "{{ dn
  dnac_verify: no
  deployment_id: "{{ dep
  register: final_status
  until: final_status.stat
  retries: 10
  delay: 5

- name: Output final deployment status
  debug:
    msg: "Deployment completed"

- name: Deploy the CLI template
  cisco.dnac.template_deploy
  dnac_host: "{{ dnac_
  dnac_username: "{{ dn
  dnac_password: "{{ dn
  dnac_verify: no
  state: query
  target_info: "{{ device_info
  register: deployment_status
```

Variables and their Purpose:

1. `dnac_host`: The URL of the Cisco
2. `dnac_username` and `dnac_password`

ChatGPT knows the order of operations, and adds some validations and errors checking, if device or template do not exist.

Example Variables File (vars.yml):

```
yaml
dnac_host: "https://dnacenter_url"
dnac_username: "your_username"
dnac_password: "your_password"
device_ip: "10.1.1.10" # Replace with the device IP address
template_name: "Basic_Configuration" # Replace with the actual template name
interface_name: "GigabitEthernet1" # Example interface parameter for the template
description: "Configured by Ansible" # Example description parameter for the template
```

Playbook Execution:

You can run this playbook with the following command:

```
bash
ansible-playbook -i localhost, deploy_cli_template.yml -e @vars.yml
```

Explanation of the Playbook:

- **Retrieve the device info:** Uses the `cisco.dnac.device_info` module to find the device by its IP address.
- **Get the CLI template info:** Uses the `cisco.dnac.template_info` module to find the CLI template by name.
- **Deploy the CLI template:** The `cisco.dnac.template_deploy` module is used to deploy the template to the target device, passing necessary parameters.

Ansible Code Generation Challenges

What is the Catalyst Center Ansible collection version is compatible with the proposed playbook?



The proposed Ansible playbook uses modules from the [Cisco DNA Center \(Catalyst Center\)](#)

Ansible Collection. As of the time of writing, the compatible version of the collection for the playbook is `cisco.dnac` version 1.1.3 or later. This version includes the required modules such as:

1. `cisco.dnac.device_info`
2. `cisco.dnac.template_info`
3. `cisco.dnac.template_deploy`
4. `cisco.dnac.template_deployment_status`

To check or install the correct version of the collection:

1. Check the installed version:

```
bash
```

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

Good start, but not perfect.

Be aware of versions used for examples, sometimes outdated.

The suggested code is using version 1.1.3. The same is valid for Python, using an older version of the Python SDK.

What is the version of the Python SDK you used for the proposed code?



The code I provided is designed to work with `dnacentersdk` version 2.3.5. This version aligns with the API capabilities and functions of Cisco Catalyst Center (formerly Cisco DNA Center) as of that SDK release.

If you're using a different version of Catalyst Center or the SDK, you might need to adjust the API calls or the version in the code:

```
python
```

Copy code

```
dnac = DNACenterAPI(base_url="https://<your-dnac-url>", username="<username>", pas:
```

For other versions, you can find the correct API version by referencing the Catalyst Center API documentation or checking compatibility with your deployment.

Code Generation Update

Can you update the playbook using the current version, 2.3.7.6, <https://galaxy.ansible.com/ui/repo/published/cisco/dnac/> ?



Searched 6 sites ▾

To update the proposed Ansible playbook for **Cisco Catalyst Center Ansible Collection version 2.3.7.6**, we need to utilize the modules compatible with this version of the collection. Here's an updated playbook using the `cisco.dnac.configuration_template_deploy_v2` and `cisco.dnac.configuration_template_deploy_status_info` modules, which are part of this release and are used to deploy and track configuration templates.

Updated Ansible Playbook:

yaml

Copy code

```
---
- name: Deploy CLI template to a device using Cisco DNA Center (v2.3.7.6)
  hosts: localhost
  gather_facts: no
  tasks:

    - name: Get device information by IP address
      cisco.dnac.network_device_info:
        dnac_host: "{{ dnac_host }}"
        dnac_username: "{{ dnac_username }}"
        dnac_password: "{{ dnac_password }}"
        dnac_verify: false
        dnac_version: "2.3.7.6"
        state: query
        ip_address: "{{ device_ip }}"
        register: device_info
```

We may provide the resource to the current version.
The Playbook is rewritten using v2 APIs and modules, and adding the Catalyst Center version

```
- name: Deploy the CLI template to the device
  cisco.dnac.configuration_template_deploy_v2:
    dnac_host: "{{ dnac_host }}"
    dnac_username: "{{ dnac_username }}"
    dnac_password: "{{ dnac_password }}"
    dnac_verify: false
    dnac_version: "2.3.7.6"
    target_info:
      - id: "{{ device_info.response[0].id }}"
        type: "MANAGED_DEVICE_IP"
        params:
          interfaceName: "{{ interface_name }}"
          description: "{{ description }}"
        template_id: "{{ template_info.response[0].templateId }}"
    register: deployment_status

- name: Check deployment status until complete
  cisco.dnac.configuration_template_deploy_status_info:
    dnac_host: "{{ dnac_host }}"
    dnac_username: "{{ dnac_username }}"
    dnac_password: "{{ dnac_password }}"
    dnac_verify: false
    dnac_version: "2.3.7.6"
    deployment_status: "{{ deployment_status.deploymentId }}"
    status: "COMPLETED"
    msg: "Deployment completed with status: {{ final_status.status }}"
```

Copy code

Ansible Playbook for Catalyst Center 2.3.7.6

```
- name: Get device info by IP address
  cisco.dnac.network_device_info:
    dnac_host: "{{ dnac_host }}"
    dnac_username: "{{ dnac_username }}"
    dnac_password: "{{ dnac_password }}"
    dnac_verify: False
    managementIpAddress: "{{ device_ip }}"
  register: device_info

- name: Fail if device not found
  fail:
    msg: "Device with IP {{ device_ip }} not found."
  when: device_info.dnac_response.response | length == 0

- name: Get CLI template by name
  cisco.dnac.templates_details_info:
    dnac_host: "{{ dnac_host }}"
    dnac_username: "{{ dnac_username }}"
    dnac_password: "{{ dnac_password }}"
    dnac_verify: "{{ dnac_verify }}"
    name: "{{ template_name }}"
  register: template_info

- name: Fail if template not found
  fail:
    msg: "Template '{{ template_name }}' not found."
  when: template_info.dnac_response.response | length == 0

- name: Print device id
  ansible.builtin.debug:
    msg: "Device id is: {{ device_info.dnac_response.response[0].id }}"
```

```
PLAY [Deploy CLI template to device using Catalyst Center] *****

TASK [Device hostname] *****
ok: [localhost] => {
  "msg": "Configuring the device with the name: 10.93.141.22"
}

TASK [Get device info by IP address] *****
ok: [localhost]

TASK [Fail if device not found] *****
skipping: [localhost]

TASK [Get CLI template by name] *****
ok: [localhost]

TASK [Fail if template not found] *****
skipping: [localhost]

TASK [Print device id] *****
ok: [localhost] => {
  "msg": "Device id is: 657008e6-6160-481e-aa7d-a5b3513b1f7c"
}

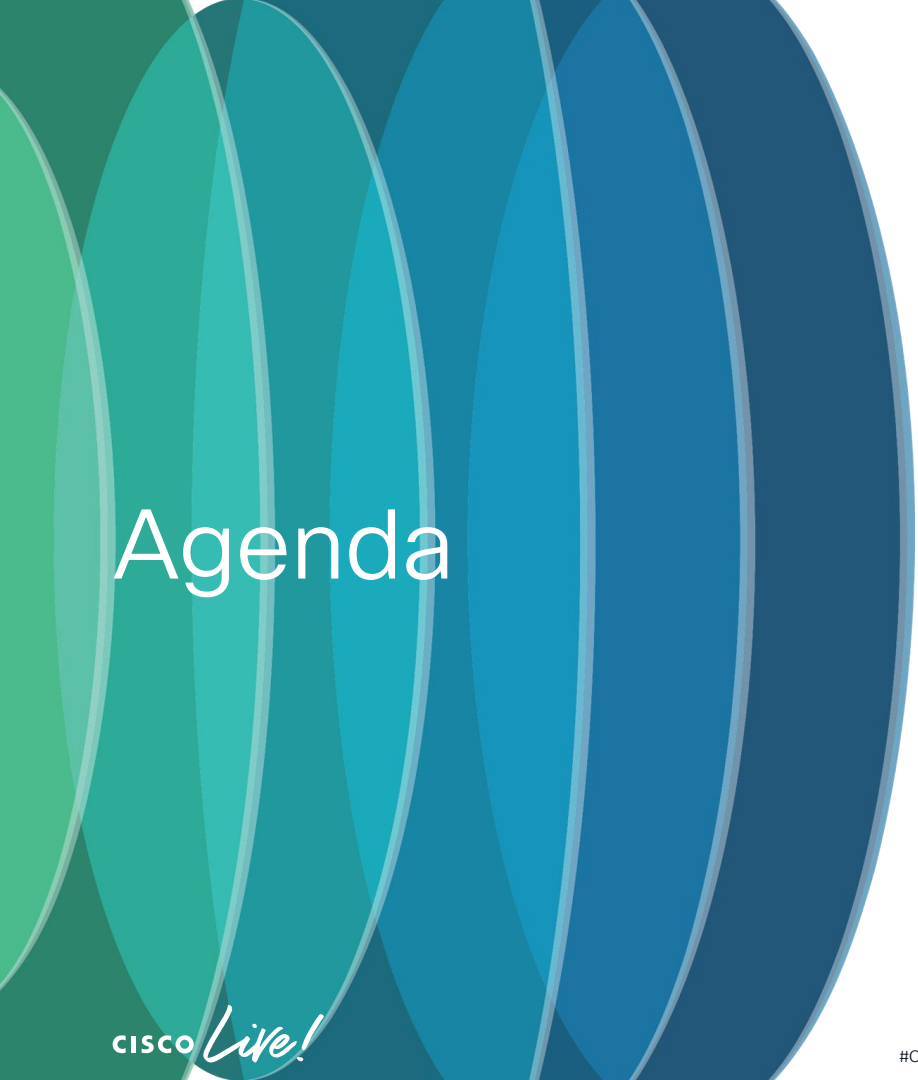
TASK [Print template id] *****
ok: [localhost] => {
  "msg": "Template id is: 657008e6-6160-481e-aa7d-a5b3513b1f7c"
}

TASK [Deploy CLI template to device] *****
ok: [localhost]

TASK [Wait for deployment completion] *****
ok: [localhost]

TASK [Output deployment result] *****
ok: [localhost] => {
  "msg": "Deployment completed with status: {'changed': False, 'dnac_response': {'response': {'progress': 'Deploym
a12.ApplicableTargets: [657008e6-6160-481e-aa7d-a5b3513b1f7c]Template Deployemnt Id: 6256950b-de5e-4ccc-b7bb-8b661cf
725201486, 'version': 1729725201486, 'data': '73aab375-30ef-4a41-90de-all0a072ea12', 'serviceType': 'NCTP', 'usern
': '5f7f7b4e22689d00c4c923e7', 'id': '0192bba7-fach-7e40-ad55-e4ff917f38e0'}, 'version': '1.0'}, 'result': '', 'faile
}

PLAY RECAP *****
localhost                : ok=8   changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
```

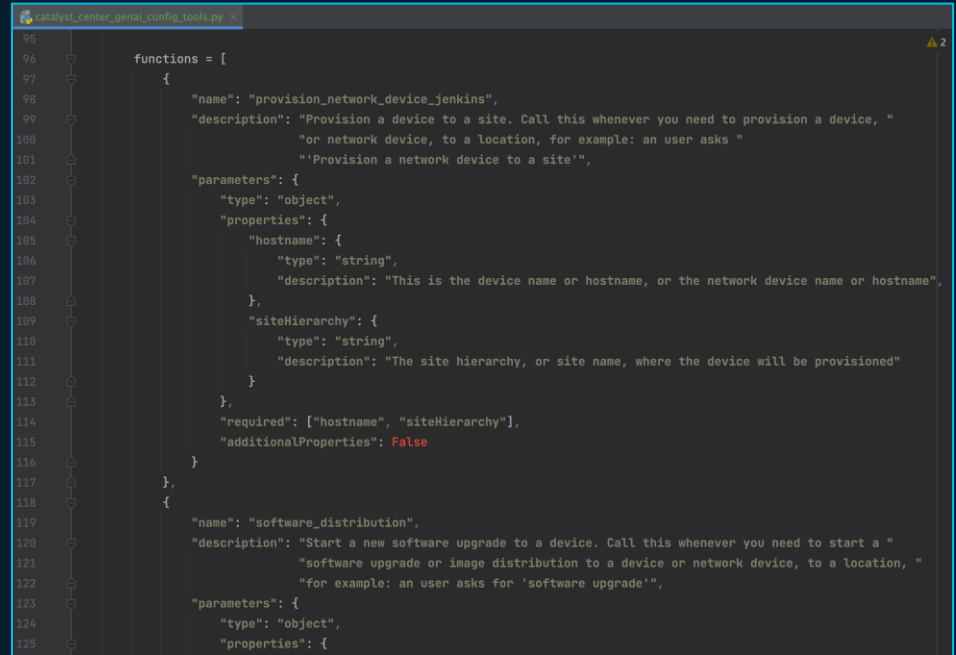


Agenda

- Catalyst Center Platform Overview
- Integrations
- GenAI Essentials
- AI Code Generation
- GenAI Network Automation
- Developers Resources
- Summary

AI Enable your Network Management Tools

- Re-use existing investment in:
 - Network automation workflows written in Python, Ansible, Terraform, ...
 - CI/CD pipelines and orchestration platforms
- Multi-controller support
- Multi-channel interface in your native language
- Accelerate workflow code development



```
catalyst_center_genai_config_tools.py
95
96     functions = [
97     {
98         "name": "provision_network_device_jenkins",
99         "description": "Provision a device to a site. Call this whenever you need to provision a device, "
100             "or network device, to a location, for example: an user asks "
101             "'Provision a network device to a site'",
102         "parameters": {
103             "type": "object",
104             "properties": {
105                 "hostname": {
106                     "type": "string",
107                     "description": "This is the device name or hostname, or the network device name or hostname",
108                 },
109                 "siteHierarchy": {
110                     "type": "string",
111                     "description": "The site hierarchy, or site name, where the device will be provisioned"
112                 }
113             },
114             "required": ["hostname", "siteHierarchy"],
115             "additionalProperties": False
116         },
117     },
118     {
119         "name": "software_distribution",
120         "description": "Start a new software upgrade to a device. Call this whenever you need to start a "
121             "software upgrade or image distribution to a device or network device, to a location, "
122             "for example: an user asks for 'software upgrade'",
123         "parameters": {
124             "type": "object",
125             "properties": {
```

LLM Function Calling

- Connects Large Language Models to other platforms to:
 - Retrieve data with API calls
 - Perform tasks or trigger processes
 - Create new workflows and AI assistants
- The LLM does not call the function, it identifies the function, and it returns function name and the required parameters

```
functions = [  
  {  
    "name": "provision_network_device_jenkins",  
    "description": "Provision a device to a site. Call this whenever you need to provision a device, "  
                  "or network device, to a location, for example: an user asks "  
                  "'Provision a network device to a site'",  
    "parameters": {  
      "type": "object",  
      "properties": {  
        "hostname": {  
          "type": "string",  
          "description": "This is the device name or hostname, or the network device name or hostname"  
        },  
        "siteHierarchy": {  
          "type": "string",  
          "description": "The site hierarchy, or site name, where the device will be provisioned"  
        }  
      },  
      "required": ["hostname", "siteHierarchy"],  
      "additionalProperties": False  
    }  
  },  
]
```

OpenAI Function Calling

<https://platform.openai.com/docs/guides/function-calling>

OpenAI Platform

Docs API reference Log in Sign up

Q Search 36 K

CAPABILITIES

Text generation

Image generation

Vision

Audio generation

Text to speech

Speech to text

Vector embeddings

Moderation

Reasoning

GUIDES

Function calling

Structured outputs

Evaluations

Fine-tuning

Distillation

</> Cookbook

🗯️ Forum

🔗 Help

Function calling

Learn how to connect large language models to external tools.

Introduction

Function calling allows you to connect models like `gpt-4o` to external tools and systems. This is useful for many things such as empowering AI assistants with capabilities, or building deep integrations between your applications and the models.

In August 2024, we launched Structured Outputs. When you turn it on by setting `strict: true`, in your function definition, Structured Outputs ensures that the arguments generated by the model for a function call exactly match the JSON Schema you provided in the function definition.

① As an alternative to function calling you can instead constrain the model's regular output to match a JSON Schema of your choosing. [Learn more](#) about when to use function calling vs when to control the model's normal output by using `response_format`.

Example use cases

Function calling is useful for a large number of use cases, such as:

1 **Enabling assistants to fetch data:** an AI assistant needs to fetch the latest customer data from an internal system when a user asks "what are my recent orders?" before it can generate the response to the user

2 **Enabling assistants to take actions:** an AI assistant needs to schedule meetings based on user preferences and calendar availability.

3 **Enabling assistants to perform computation:** a math tutor assistant needs to perform a math computation.

4 **Building rich workflows:** a data extraction pipeline that fetches raw text, then converts it to structured data and

If the LLM of your choice does not have a native function calling, we could develop one for our use case

CISCO Live!

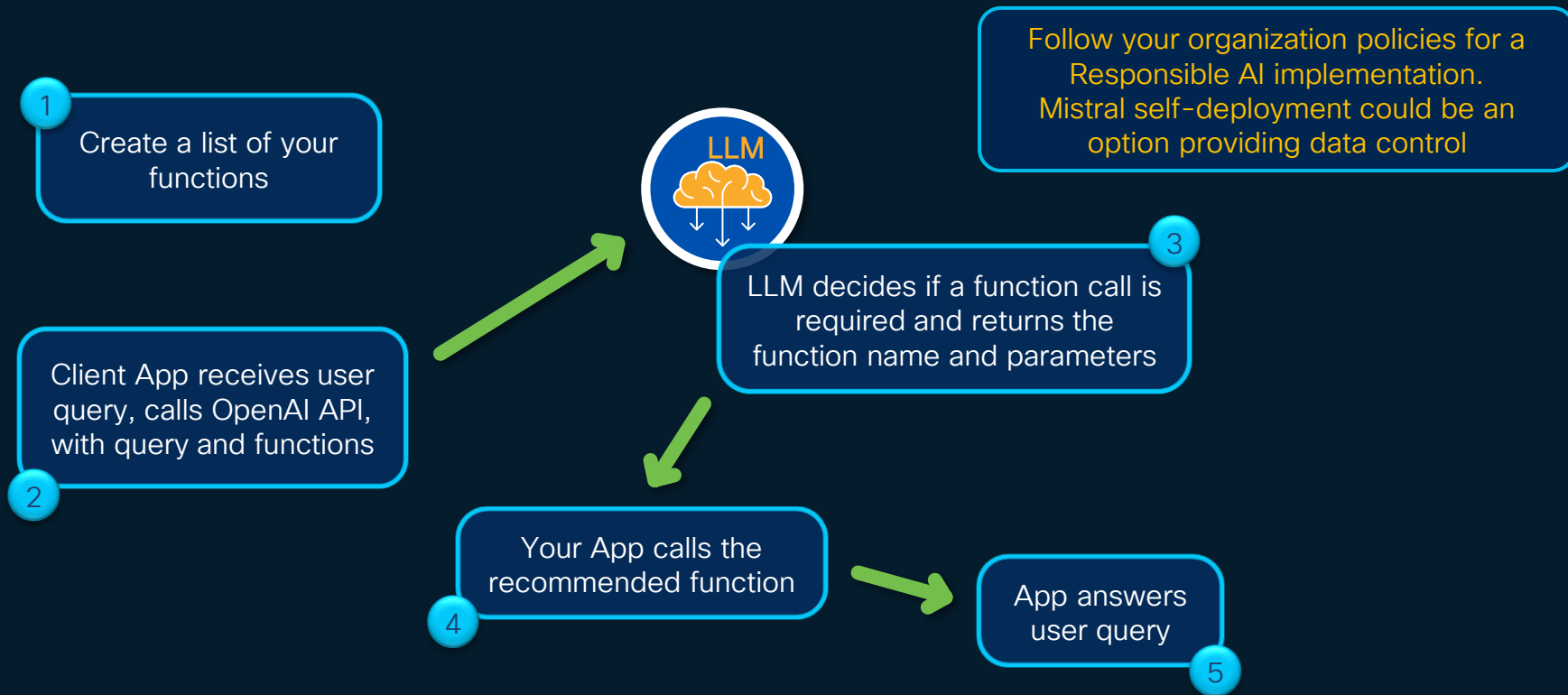
#CiscoLiveAPJC

DEVNET-3419

© 2024 Cisco and/or its affiliates. All rights reserved. Cisco Public

46

OpenAI Function Calling Workflow



Function Database

Functions to be called

```
def provision_network_device_jenkins(arguments):  
    """  
    This function will call REST APIs to trigger Jenkins pipeline to provision the network device with {hostname} to  
    site with the hierarchy {site_hierarchy}  
    :param arguments: Required arguments as dictionary: {'hostname': 'PDX-RN', 'site_hierarchy': 'Global/OR/PDX/Floor-2'}  
    :return: task status  
    """  
  
    url = JENKINS_SERVER + '/job/Provision%20Device/buildWithParameters'  
    response = requests.post(url, params=arguments, auth=JENKINS_AUTH)  
    if response.status_code == 201:  
        response_status = 'Device provisioning started, see status'  
    else:  
        response_status = 'Pipeline not started, something went wrong'  
    return response_status, response.status_code
```

Functions database or list,
that will be used by the LLM.
They map to functions
available in your code

```
functions = [  
    {  
        "name": "provision_network_device_jenkins",  
        "description": "Provision a device to a site. Call this whenever you need to provision a device, "  
        "or network device, to a location, for example: an user asks "  
        "Provision a network device to a site'",  
        "parameters": {  
            "type": "object",  
            "properties": {  
                "hostname": {  
                    "type": "string",  
                    "description": "This is the device name or hostname, or the network device name or hostname"  
                },  
                "siteHierarchy": {  
                    "type": "string",  
                    "description": "The site hierarchy, or site name, where the device will be provisioned"  
                }  
            }  
        },  
        "required": ["hostname", "siteHierarchy"],  
        "additionalProperties": False  
    }  
]
```

NetOps AI Assistant

I am a network assistant running network automation workflows. What network configuration task are you interested in? *Can you start the software upgrade for the device PDX-ACCESS?*

Workflow name: software_distribution
Params identified:
{ "hostname": "PDX-ACCESS" }

Do you want to continue or not (y/n)? *n*

I am a network assistant running network automation workflows. What network configuration task are you interested in? *Provision this switch BEAV-CN at this location Global/OR/BEAV/Floor-2*

Workflow name: provision_network_device_jenkins
Params identified:
{ "hostname": "BEAV-CN", "siteHierarchy": "Global/OR/BEAV/Floor-2" }

Do you want to continue or not (y/n)? *y*

Network Assistant: Device provisioning started, see status here: <https://10.93.141.47:8443/job/Provision%20Device/>



API call to trigger pipeline run

GitHub

Single source of truth, version controlled, data validation, compliant

Pull source-code and intent configs

Pipeline build will configure Catalyst Center and network devices

Pipelines may execute Python, Ansible or Terraform workflows



Multi controllers, lab and production

- Multi-channel support
- Webex Bots
- Web Portal
- ITSM – change management
- Voice interface



Function Database (List)

- Create the list of actions the Assistant will be able to take
- Each function requires:
 - Detailed name and description
 - Params type and descriptions, required or not
- The name and descriptions will be used by AI to match query with desired function

```
198 functions = [  
199   {  
200     "name": "provision_network_device_jenkins",  
201     "description": "Provision a device to a site. Call this whenever you need to provision a device, "  
202                   "or network device, to a location, for example: an user asks "  
203                   "'Provision a network device to a site'",  
204     "parameters": {  
205       "type": "object",  
206       "properties": {  
207         "hostname": {  
208           "type": "string",  
209           "description": "This is the device name or hostname, or the network device name or hostname"  
210         },  
211         "siteHierarchy": {  
212           "type": "string",  
213           "description": "The site hierarchy, or site name, where the device will be provisioned"  
214         }  
215       },  
216       "required": ["hostname", "siteHierarchy"],  
217       "additionalProperties": False  
218     },  
219   },  
220   {  
221     "name": "software_distribution",  
222     "description": "Start a new software upgrade to a device. Call this whenever you need to start a "  
223                   "software upgrade or image distribution to a device or network device, to a location, "  
224                   "for example: an user asks for 'software distribution'",  
225     "parameters": {  
226       "type": "object",  
227       "properties": {...},  
228       "required": ["hostname"],  
229       "additionalProperties": False  
230     },  
231   }  
232 ]
```

Chat Completion using Function Calling

```
138
139 # Start a chat that triggers the function
140 messages = [
141     {"role": "system", "content": "I am a network assistant running network automation workflows like devices "
142                                     "provisioning, PnP onboarding, software upgrades, device configuration, ..."},
143     {"role": "user",
144      "content": user_input}
145 ]
146
147 # Make a request to the OpenAI API with function calling
148 response = client.chat.completions.create(model="gpt-4o", messages=messages, functions=functions,
149                                           function_call="auto")
150 print(' Chat completion response: ' + str(response))
151
152 # Check if the assistant decided to make a function call
153 if response.choices[0].finish_reason == 'function_call':
154     function_call = response.choices[0].message.function_call # Access the function call
155     function_name = function_call.name # Get the function name
156     arguments = json.loads(function_call.arguments) # Load the arguments as a dictionary
157     print('\n Workflow name: ' + function_name)
158     print(' Params identified: \n      ' + json.dumps(arguments))
159     validation_input = input('\n Do you want to continue or not (y/n)? ')
160
```

- Collect the user input
- Call the chat completion API:
 - Specify model – “gpt-4o”
 - Message includes the user input
 - Functions list
- The LLM will use the message and identify the best match for the **function and the parameters provided**

Function Calling Response

I am a network assistant running network automation workflows. What network configuration task are you interested in? *Start the image distribution to the device PDX-RN*

Chat completion response: ChatCompletion(id='chatcmpl-ALfZVf1knvWdMvMiLvUkCQwpurGw', choices=[Choice(finish_reason='function_call', index=0, logprobs=None, message=ChatCompletionMessage(content=None, refusal=None, role='assistant', audio=None, function_call=FunctionCall(arguments='{\"hostname\": \"PDX-RN\"}', name='software_distribution'), tool_calls=None))], created=1729728053, model='gpt-4o-2024-08-06', object='chat.completion', service_tier=None, system_fingerprint='fp_9e15ccd6a4', usage=CompletionUsage(completion_tokens=18, prompt_tokens=230, total_tokens=248, completion_tokens_details=CompletionTokensDetails(audio_tokens=None, reasoning_tokens=0), prompt_tokens_details=PromptTokensDetails(audio_tokens=None, cached_tokens=0)))

Workflow name: software_distribution
Params identified:
{\"hostname\": \"PDX-RN\"}

Proposed action, and
param identified

Do you want to continue or not (y/n)?

```
{
  "id": "chatcmpl-ALfZVf1knvWdMvMiLvUkCQwpurGw",
  "model": "gpt-4o-2024-08-06",
  "created": 1729728053,
  "object": "chat.completion",
  "system_fingerprint": "fp_9e15ccd6a4",
  "service_tier": null,
  "choices": [
    {
      "index": 0,
      "role": "assistant",
      "finish_reason": "function_call",
      "message": {
        "content": null,
        "function_call": {
          "name": "software_distribution",
          "arguments": {
            "hostname": "PDX-RN"
          }
        }
      }
    }
  ],
}
```

Function Calling Response Error Handling

I am a network assistant running network automation workflows. What network configuration task are you interested in? *Can you create a fabric at this site: Global/OR/PDX/Floor-2?*

Something went wrong, not sure what you asked, Creating a fabric involves a complex process that involves tasks such as provisioning devices, configuring network settings, and possibly coordinating with site-specific infrastructure requirements. While I'm equipped to handle provisioning network devices to specific sites, the creation of a network fabric is beyond my direct capabilities.

However, if you have devices that need provisioning to this site, please provide their hostname, and I can assist you with that!

I am a network assistant running network automation workflows. What network configuration task are you interested in? |

No action
recommended

Missing required
parameter

I am a network assistant running network automation workflows. What network configuration task are you interested in? *Start the provisioning of this device PDX-R0*

Something went wrong, not sure what you asked, Could you please provide the site hierarchy or site name where the device PDX-R0 should be provisioned?

I am a network assistant running network automation workflows. What network configuration task are you interested in? |

Jenkins Pipeline

Workflow name: provision_network_device_jenkins
Params identified:
{ "hostname": "PDX-RN", "siteHierarchy": "Global/OR/PDX/Floor-2" }

Do you want to continue or not (y/n)? y

Network Assistant: Device provisioning started, see status here: https://10.10.10.10:8080/job/provision_network_device_jenkins/



Dashboard > OpsView >

- + New Item
- People
- Build History
- Edit View
- Delete View
- Project Relationship
- Check File Fingerprint
- Manage Jenkins
- My Views
- Open Blue Ocean

Build Queue

No builds in the queue.

Dashboard > OpsView > Provision Device >

Status

</> Changes

Build with Parameters

Configure

Delete Pipeline

Full Stage View

Open Blue Ocean

Rename

Pipeline Syntax

Build History

trend

Filter builds...

#71	Sep 10, 2024, 10:36 PM
#70	Sep 10, 2024, 8:20 PM
#69	Sep 9, 2024, 8:16 PM
#68	Sep 9, 2024, 8:09 PM
#67	Sep 9, 2024, 8:02 PM
#66	Sep 9, 2024, 6:01 PM
#65	Sep 9, 2024, 6:00 PM
#64	Sep 9, 2024, 5:58 PM
#63	Sep 9, 2024, 5:58 PM

Provision D

This pipeline is for the auto

Stage View

Average stage
(Average full run time)

#71	Sep 10, 2024, 10:36 PM	No Changes
#70	Sep 10, 2024, 8:20 PM	No Changes
#69	Sep 09, 2024, 8:16 PM	No Changes
#68	Sep 09, 2024, 8:09 PM	No Changes
#67	Sep 09, 2024, 8:02 PM	No Changes
#66	Sep 09, 2024, 6:01 PM	No Changes
#65	Sep 09, 2024, 6:00 PM	No Changes
#64	Sep 09, 2024, 5:58 PM	No Changes
#63	Sep 09, 2024, 5:58 PM	No Changes

Dashboard > OpsView > Provision Device > Configuration

☒ This project is parameterized ?

String Parameter ?

Name ?

hostname

Default Value ?

Description ?

device hostname

Plain text [Preview](#)

☒ Trim the string ?

String Parameter ?

Name ?

siteHierarchy

Default Value ?

Description ?

site hierarchy, example: Global/Area/B

93ms	69ms	84ms
94ms	72ms	66ms

Dashboard > OpsView > Provision Device > Configuration

Build Triggers

- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☐ GitHub Branches
- ☐ GitHub Pull Requests ?
- ☐ GitHub hook trigger for GITScm polling ?
- ☐ Poll SCM ?
- ☐ Quiet period ?
- ☒ Trigger builds remotely (e.g., from scripts) ?

Provision Device > #71

[Pipeline] withEnv
[Pipeline] {
[Pipeline] echo

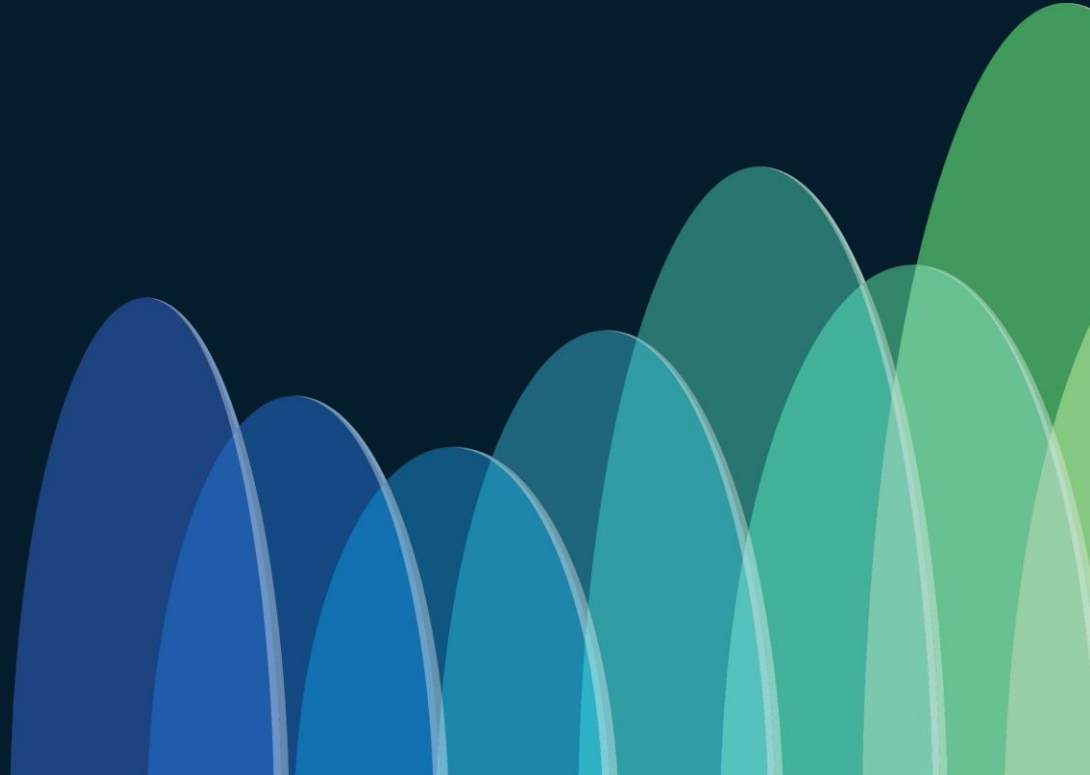
Verify Working Path:.....

[Pipeline] sh
+ pwd
/var/lib/jenkins/workspace/Provision Device
[Pipeline] dir

Running in /var/lib/jenkins/workspace/Provision Device/dnacentr_jenkins_automations

[Pipeline] {
[Pipeline] sh
+ python3 provision_wired_device.py PDX-RN Global/OR/PDX/Floor-2
INFO:root: App "provision_wired_device.py" run start, 2024-09-10 22:36:43
INFO:root: Device hostname: PDX-RN
INFO:root: Site hierarchy: Global/OR/PDX/Floor-2
INFO:root: Provisioning Task Status: pending
INFO:root: Provisioning Task URL: /dna/intent/api/v1/task/0191df97-26a1-7e8b-a995-5e119e90ef19
INFO:root: Device provisioned successfully to site
INFO:root: App "provision_wired_device.py" run end: 2024-09-10 22:37:02
[Pipeline] }
[Pipeline] // dir

Demo NetOps AI Assistant



Camtasia 2024 File Edit Modify Text View Export Window Help

Provision Device [Jenkins] x +

https://10.93.141.45/dna/provision/devices/inventory/list 120%

Cisco Systems, Inc. APIs Tools LAB Access Cisco Live Meraki Dashboard L... SFEH-NAS - Synolo... CDETS ONASAP SharePoint APIs and Ownp... JIRA Engineering S3... Other Bookmarks

Provision / Inventory

Global

✓ All Routers Switches Wireless Controllers Access Points Sensors

Take a tour Export

FILTERED BY

Unassigned x

DEVICE WORK ITEMS

- ☐ Unreachable
- ☒ Unassigned
- ☐ Untagged
- ☐ Failed Provision
- ☐ Non Compliant
- ☐ Outdated Software Image
- ☐ No Golden Image
- ☐ Failed Image Prechecks
- ☐ Under Maintenance
- ☐ Security Advisories
- ☐ Marked for Deployment

Devices (1) Focus: Select

0 Selected Tag Add Device Actions

As of: Sep 10, 2024 10:30 PM

☐	Tags	Device Name	IP Address	Device Family	Site	Reachability	Provisioning Status	Device Role	MAC Address
☐		PDX-RN	10.93.141.22	Routers	Assign	Reachable	Not Provisioned	BORDER ROUTER	00:1e:3d:8a:5d:00

1 Record(s)

Show Records: 25 1 - 1

Open-source Code

https://github.com/zapodeanu/catalyst_center_genai_ops

The screenshot displays the GitHub interface for the repository `zapodeanu / catalyst_center_genai_ops`. The repository is public and has 0 stars, 0 forks, and 1 watcher. The main content area shows a list of files and their commit history:

File	Commit Message	Time Ago
<code>.gitignore</code>	updated Readme, gitignore, app	4 minutes ago
<code>CODE_OF_CONDUCT.md</code>	initial commit	3 minutes ago
<code>CONTRIBUTING.md</code>	initial commit	3 minutes ago
<code>LICENSE</code>	initial commit	3 minutes ago
<code>NOTICE</code>	initial commit	3 minutes ago
<code>README.md</code>	updated README	2 minutes ago
<code>catalyst_center_genai_config_tools.py</code>	updated Readme, gitignore, app	4 minutes ago
<code>requirements.txt</code>	initial commit	3 minutes ago

The sidebar on the right contains the following information:

- About:** The repo includes the files for GenAI operations using function calling and Catalyst Center REST APIs. Links to Readme, View license, Code of conduct, Activity, 0 stars, 1 watching, and 0 forks.
- Releases:** No releases published. [Create a new release](#)
- Packages:** No packages published. [Publish your first package](#)

NLP Interface in Your Native Language

I am a network assistant running network automation workflows. What network configuration task are you interested in? *Ter beschikking stellen van AMS-R0 aan locatie Global/NL/AMS/Floor-2*

Workflow name: provision_network_device_jenkins

Params identified:

`{"hostname": "AMS-R0", "siteHierarchy": "Global/NL/AMS/Floor-2"}`

Do you want to continue or not (y/n)? *n*

I am a network assistant running network automation workflows. What network configuration task are you interested in? *Rozpocznij aktualizację oprogramowania przełącznika o nazwie hosta PL_WAR06*

Workflow name: software_distribution

Params identified:

`{"hostname": "PL_WAR06"}`

Do you want to continue or not (y/n)? *n*

I am a network assistant running network automation workflows. What network configuration task are you interested in? *Démarrez la mise à niveau logicielle du commutateur portant les noms d'hôte F-PAR-SW10*

Workflow name: software_distribution

Params identified:

`{"hostname": "F-PAR-SW10"}`

Do you want to continue or not (y/n)? *n*

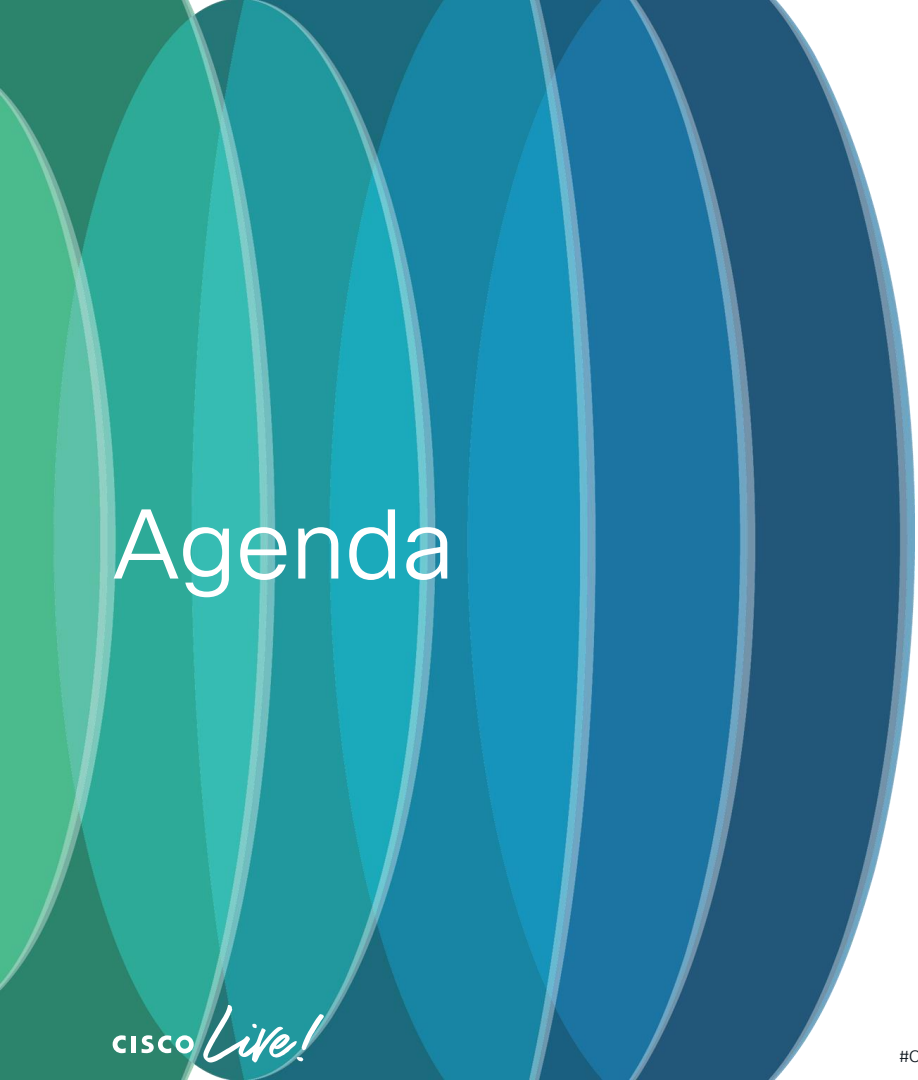
I am a network assistant running network automation workflows. What network configuration task are you interested in? *ホスト名 J-TOK-SW10 でスイッチのソフトウェア アップグレードを開始します。*

Workflow name: software_distribution

Params identified:

`{"hostname": "J-TOK-SW10"}`

Do you want to continue or not (y/n)? *n*



Agenda

- Catalyst Center Platform Overview
- Integrations
- GenAI Essentials
- AI Code Generation
- GenAI Network Automation
- Developers Resources
- Summary

Catalyst Center: API journey/progression



Cisco API
platform launch



Python SDK



Ansible
collection



Ansible
RedHat certified
Terraform provider
Jenkins open-
source



Ansible community
certification
Intent Ansible
modules introduced



Python SDK,
Ansible, Terraform
Support for
Catalyst Center
2.3.5.3



AI enabled use
cases
Support for
Catalyst Center
2.3.7.6

2018

2019

2020

2021

2022

2023

2024

API adoption



2 billion API calls per year, 45% customers



850+ APIs available (300+ new APIs in 2.3.7.6)



269 events and 65 reports

DevOps resources



Python SDK – 6,300 downloads/month



Ansible collection – 66,000
downloads



Terraform provider – 44,000 downloads

25 New Catalyst Center DevNet sandboxes

Catalyst Center API focus areas



Usability

Documentation

Ease of API use

Developer Resources



Completeness

Improve API coverage

Backward compatibility and
versioning

Better Prioritization



Scale

Rate Limit

Concurrence

Bulk Operations



Conformance

Cisco Standards

API governance,
Open API

API Best Practices

Quality

New Cisco EN Splunk integration apps

The screenshot shows the Splunkbase interface for the 'Cisco Catalyst Add-on for Splunk'. The header includes the 'splunkbase' logo, navigation links for 'Collections' and 'Apps', a search bar with the placeholder 'Find an app', and buttons for 'Submit an App' and 'Log In'. The main content area features the app title 'Cisco Catalyst Add-on for Splunk' with the Cisco logo. Below the title is a description: 'Cisco Catalyst Add-on for Splunk collects data for different Cisco Products - **Cisco Identity Services Engine**, **Cisco Catalyst SD-WAN**, and **Cisco Catalyst Center**. The add-on parses the data from these sources and stores them into the Splunk indexes. * Author - Cisco Systems *...'. It also mentions 'Built by Cisco EN Programmability'. A 'Login to Download' button is visible. At the bottom, there are sections for 'Latest Version 1.0.0' (released September 10, 2024), 'Compatibility' (Splunk Enterprise, Splunk Cloud, Platform Version: 9.3, 9.2, CIM Version: 5.X), and 'Rating' (0 stars, 0 reviews).

- Integrate with ISE, Catalyst SD-WAN and Catalyst Center
- Two apps – Visualizations and Add-on for data collection
- Apps released September 10, 2024
- Splunk Enterprise + Cloud certified

The screenshot shows the Splunkbase interface for the 'Cisco Enterprise Networking for Splunk Platform'. The header is identical to the previous screenshot. The main content area features the app title 'Cisco Enterprise Networking for Splunk Platform' with the Cisco logo. Below the title is a description: 'The **Cisco Enterprise Networking for Splunk Platform** presents visualizations in dashboards for different Cisco Products - **Cisco Identity Services Engine**, **Cisco Catalyst SD-WAN**, and **Cisco Catalyst Center**. The App uses the data collected by "Cisco Catalyst Add-on for...'. It also mentions 'Built by Cisco EN Programmability'. A 'Login to Download' button is visible. At the bottom, there are sections for 'Latest Version 1.0.0' (released September 10, 2024), 'Compatibility' (Splunk Enterprise, Splunk Cloud, Platform Version: 9.3, 9.2, CIM Version: 5.X), 'Rating' (0 stars, 0 reviews), 'Support' (Developer Supported App, with a 'Learn more' link), and 'Ranking' (#6 in Network Access Control, #26 in Firewall).

Catalyst Center Python SDK

- Covers all Cisco Catalyst Center REST APIs
- Support for Cisco Catalyst Center version 2.3.7.6
- First SDK version August 2019
- Average downloads/month 6,000+ (last 30 days 12,000+)

🏠 / dnacentersdk

dnacentersdk

Simple, lightweight, scalable Python API wrapper for the DNA Center APIs

Welcome to the docs! dnacentersdk is a *community developed* Pythonic wrapping of the DNA Center versions: 2.2.2.3, 2.2.3.3, 2.3.3.0, 2.3.5.3 and 2.3.7.6). The package represents all of the Cisco DNA interactions via native Python tools. Making working with the Cisco DNA Center APIs in Python a n experience.

dnacentersdk helps you get things done faster. We take care of the API semantics, and you can focus on your code.

With dnacentersdk, you can easily:

- Interact with the DNA Center APIs in an interactive Python session
- Quickly create code that enables you get something done in DNA Center
- Leverage the API wrapper to cleanly add DNA Center functionality to your project without having boilerplate code for working with the DNA Center APIs

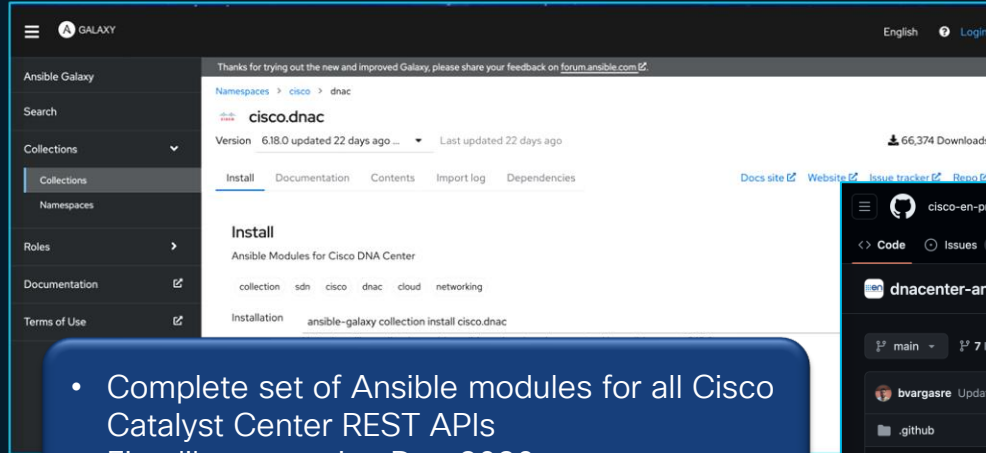
<https://dnacentersdk.readthedocs.io>

The screenshot shows the GitHub repository page for `cisco-en-programmability / dnacentersdk`. The repository is public and has 15 unwatched items, 31 forks, and 66 stars. The main content area displays a list of commits, including a recent merge pull request #154 by `bvgargse`. The commit list includes files like `.github/ISSUE_TEMPLATE`, `dnacentersdk`, `docs`, `script`, `tests`, `.flake8`, and `.gitattributes`, with their respective commit messages and timestamps.

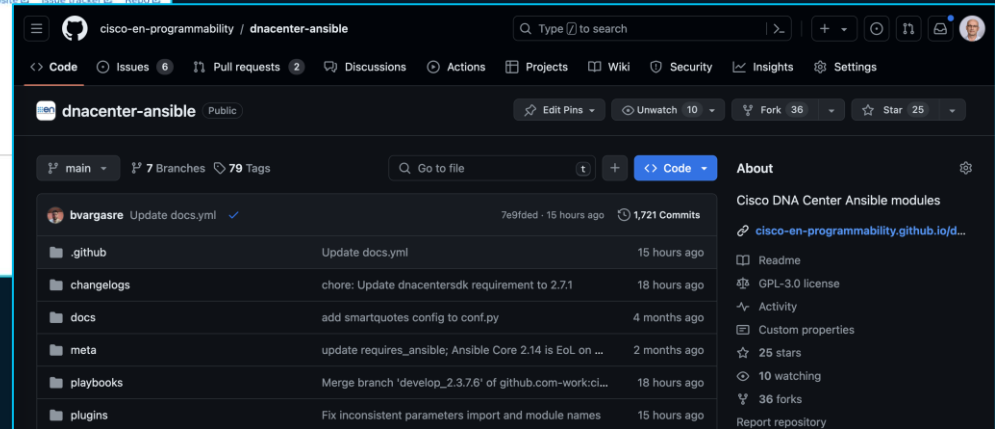
<https://github.com/cisco-en-programmability/dnacentersdk>

Cisco Catalyst Center Ansible Modules

<https://galaxy.ansible.com/ui/repo/published/cisco/dnac>



- Complete set of Ansible modules for all Cisco Catalyst Center REST APIs
- First library version Dec 2020
- Certified by RedHat v 2.9 – 04/2021
- Community certified – 05/2022
- Support for Cisco Catalyst Center version 2.3.7.6
- 66,300+ downloads
- 33% increase in last 3 Months



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

Cisco Developed, TAC/Engineering Supported

cisco.dnac
Version 6.18.0 updated 22 days ago ... Last updated 22 days ago 66,374 Downloads

Install Documentation Contents Import log Dependencies Docs site Website Issue tracker Repo

Q workflow x

Documentation (0) v

Modules (21) v

- accesspoint_workflow_mana...
- device_configs_backup_wor...
- device_credential_workflow_...
- discovery_workflow_manager
- events_and_notifications_wo...
- fabric_sites_zones_workflow...
- inventory_workflow_manager
- ise_radius_integration_workf...
- network_compliance_workfl...

Ansible Collection - cisco.dnac

Ansible Modules for DNA Center

The dnacenter-ansible project provides an Ansible collection for managing and automating your Cisco DNA Center environment. It consists of a set of modules and roles for performing tasks related to DNA Center.

This collection has been tested and supports Cisco DNA Center 2.3.7.6.

Note: This collection is not compatible with versions of Ansible before v2.8.

Other versions of this collection have support for previous Cisco DNA Center versions. The recommendation is to refer to the [Compatibility matrix](#).

Compatibility matrix

The following table shows the supported versions.

Cisco DNA Center version	Ansible "cisco.dnac" version	Python "dnacentersdk" version
2.2.2.3	3.3.1	2.3.3
2.2.3.3	6.4.0	2.4.11
2.3.3.0	6.6.4	2.5.5

- Support starting with Cisco Catalyst Center version 2.3.7.6
- Identified with the module name including "workflow_manager"
- Ansible playbooks for common operations for sites, PnP, SWIM, compliance, templates, network settings, ...

Terraform Provider and Go SDK

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

The screenshot shows the Terraform Registry page for the **dnacenter** provider. The page is titled "dnacenter" and includes a "USE PROVIDER" button. The left sidebar lists resources under "dnacenter provider". The main content area is titled "dnacenter Provider" and contains the following text:

The Cisco DNA Center provider is used to interact with Cisco DNA Center APIs (2.3.5.3). 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
}
```

- Extended coverage of Cisco Catalyst Center REST APIs
- Support for Catalyst Center version 2.3.7.6
- Released January 2021
- 44,000+ downloads

The screenshot shows the GitHub repository for the **terraform-provider-dnacenter** project. The repository is public and has 11 branches and 65 tags. The commit history shows a merge pull request #199 from cisco-en-programmability/... and a commit by cca4be - last month. The repository has 496 commits. The file list includes .github, dnacenter, docs, examples, scripts, tools, .gitignore, .goreleaser.yml, CHANGELOG.md, and LICENSE. The repository is linked to the Terraform Registry at <https://registry.terraform.io/providers/cisco-en-programmability/dnacenter>. The repository has 13 stars and 10 forks. The latest release is **v1.1.31-beta** (Latest) on Dec 7, 2023, with 60 releases.

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

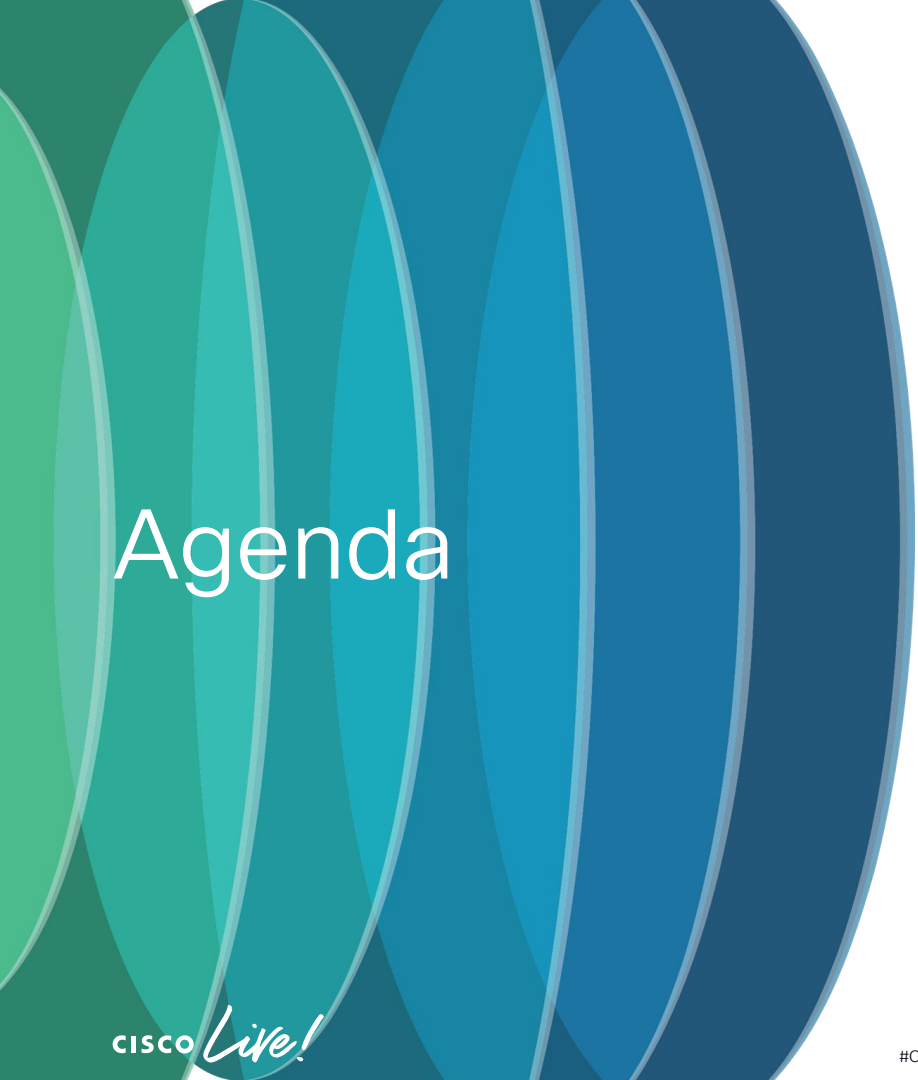
Cisco EN Programmability GitHub Org

The screenshot shows the GitHub organization page for Cisco EN Programmability. The organization has 53 followers and 41 repositories. The pinned repositories are:

- dnacentersdk** (Public): Cisco DNA Center Python SDK. 62 stars, 29 forks.
- dnacenter-ansible** (Public): Cisco DNA Center Ansible modules. 22 stars, 24 forks.
- terraform-provider-dnacenter** (Public): Cisco DNA Center Terraform Provider. 13 stars, 11 forks.
- splunk-apps** (Public): Splunk Add-On App for Cisco DNA Center. 2 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.
- dnacenter_webhook_receiver** (Public): Sample code for a Cisco DNA Center Webhook Receiver.

- DevOps Catalyst Center use cases
- Python SDK (support 2.3.7.6)
- Ansible modules (support 2.3.7.6)
- Go SDK and Terraform provider (support 2.3.5.x, pending support 2.3.7.6)
- Splunk Integration Apps
- GenAI use case
- SDA-as-Code
- Report Operations
- Day N use cases – Python SDK, Ansible, Terraform
- Jenkins open-source Integration and pipelines
- GitHub/GitLab open-source Integrations
- Compliance Use Cases
- Custom Integration App (Jira Service Desk)
- Webhook Receiver
- Other sample code

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

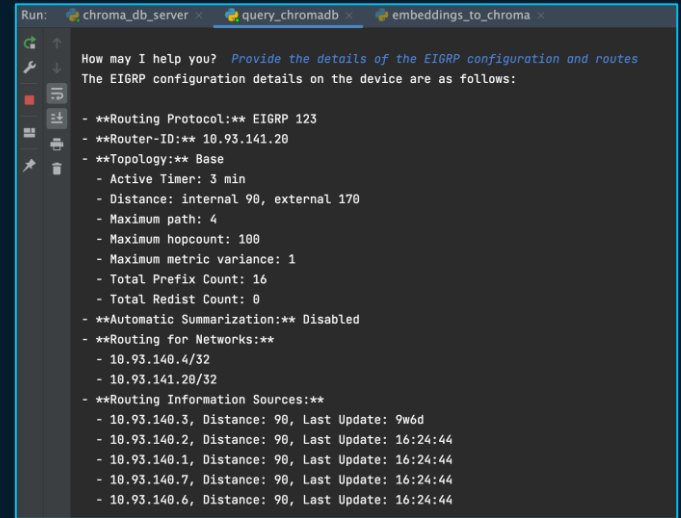


Agenda

- Catalyst Center Platform Overview
- Integrations
- GenAI Essentials
- AI Code Generation
- GenAI Network Automation
- Developers Resources
- Summary

Summary

- Support innovative use cases with data provided by Catalyst Center REST APIs or device APIs
- Massive scale for large data sets
- Training the existing models and add metadata, organize datasets, to reduce AI hallucination



The screenshot shows a web application with three tabs: 'chroma_db_server', 'query_chromadb', and 'embeddings_to_chroma'. The 'query_chromadb' tab is active, displaying a chat interface. The chat prompt is 'How may I help you? Provide the details of the EIGRP configuration and routes'. The response is 'The EIGRP configuration details on the device are as follows:'. Below the response, the configuration details are listed in a structured format:

```
- **Routing Protocol:** EIGRP 123
- **Router-ID:** 10.93.141.20
- **Topology:** Base
  - Active Timer: 3 min
  - Distance: internal 90, external 170
  - Maximum path: 4
  - Maximum hopcount: 100
  - Maximum metric variance: 1
  - Total Prefix Count: 16
  - Total Redist Count: 0
- **Automatic Summarization:** Disabled
- **Routing for Networks:**
  - 10.93.140.4/32
  - 10.93.141.20/32
- **Routing Information Sources:**
  - 10.93.140.3, Distance: 90, Last Update: 9w6d
  - 10.93.140.2, Distance: 90, Last Update: 16:24:44
  - 10.93.140.1, Distance: 90, Last Update: 16:24:44
  - 10.93.140.7, Distance: 90, Last Update: 16:24:44
  - 10.93.140.6, Distance: 90, Last Update: 16:24:44
```

Keynote Deep Dives

Wednesday
10:30am – 11:30am



Experiences Amplified:
How AI Can Fuel Better Employee and Customer Experiences

Level 1
Room 106



Smart, Secure, Seamless:
Transforming Experiences with Next-Generation Networking

Level 2
Room 204



Harness a Bold New Era:
Transform Data Centre and Service Provider Connectivity

Level 2
Room 203



Securing User to Application and Everything in Between

Level 2
Melbourne Room 2



Unlocking Digital Resilience through Unified Observability

The HUB
Centre Stage

Complete Your Session Evaluations



Complete a minimum of 4 session surveys and the Overall Event Survey to claim a **Cisco Live T-Shirt**.



Complete your surveys in the **Cisco Live mobile app**.



Continue your education



- Visit the Cisco Stand 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



Thank you

CISCO *Live!*

#CiscoLiveAPJC

CISCO *Live!*

GO BEYOND

#CiscoLiveAPJC