

The background is a vibrant, abstract graphic. It features a central bright white light source from which numerous colorful rays emanate, creating a sunburst or starburst effect. The rays transition through a spectrum of colors including yellow, orange, red, and various shades of blue and green. Overlaid on this are several large, semi-transparent, wavy shapes in similar color tones, giving the overall image a sense of motion and energy.

cisco *Live!*

Let's go

#CiscoLive



The bridge to possible

Build a Simple yet Powerful CI/CD Pipeline with Cisco ACI and Nexus Dashboard Insights

Alejandro de Alda, Technical Marketing Engineer

DEVNET-2473



#CiscoLive

Agenda

- What is a CI/CD Pipeline?
- Why do I need pre/post-change validations?
- How can I build a simple yet powerful CI/CD Pipeline?
- Conclusion

Cisco Webex App

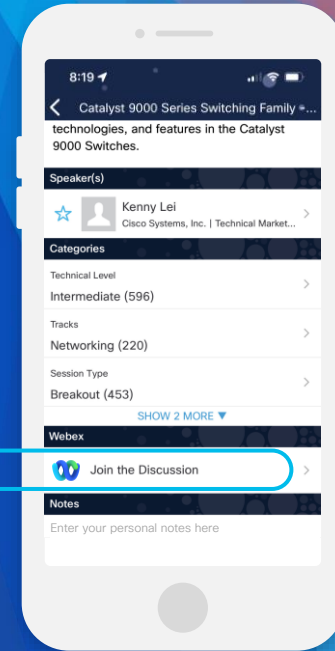
Questions?

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 June 9, 2023.



<https://ciscolive.ciscoevents.com/ciscolivebot/#DEVNET-2473>

What is a CI/CD Pipeline?



What is a CI/CD Pipeline?

- A networking CI/CD Pipeline is a process that deploys network infrastructure through a series of steps that include building, testing and deploying infrastructure as code.



Why do I need a CI/CD Pipeline?

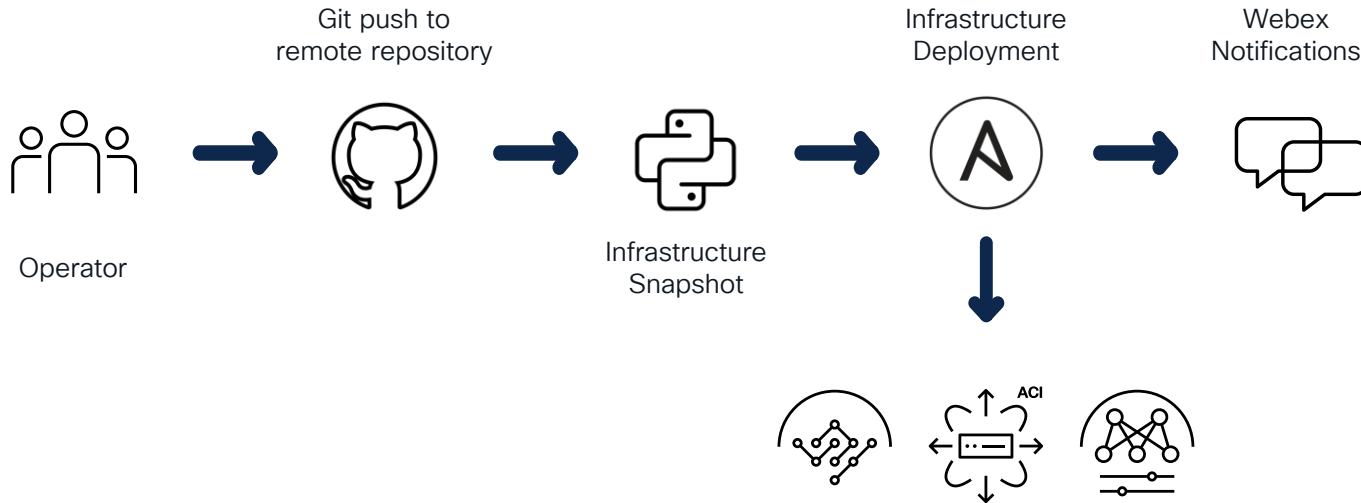
- Pipelines implement the process in a **consistent** and **automated** way
- Some of the benefits:
 - Increases efficiency and speed
 - Saves time, effort and cost
 - Minimizes human error
 - Maintains consistency
 - Reduces risk



Even simple pipelines will bring most of these benefits to your processes

A Simple CI/CD Pipeline

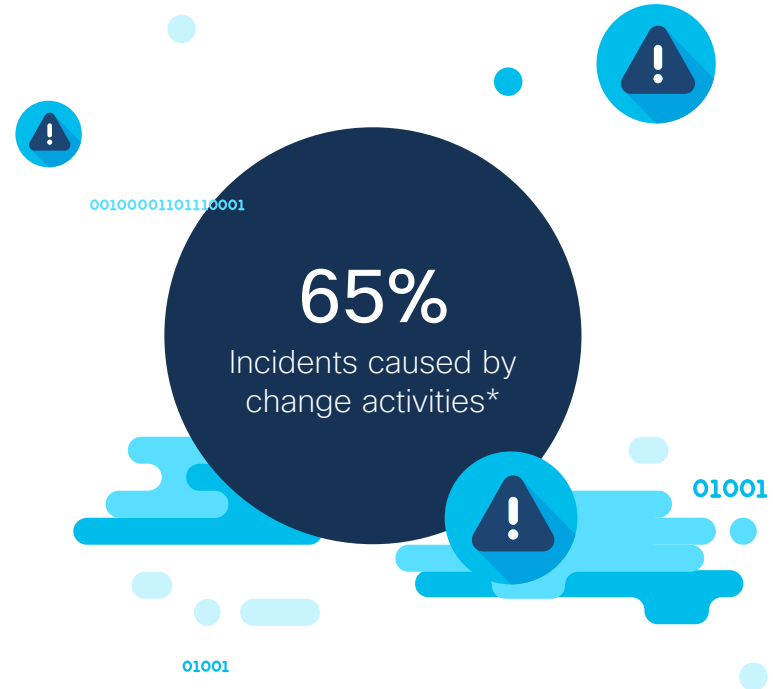
Example



Why do I need
pre/post-change
validations?

Most of incidents are caused by change activities

Verifying the impact of changes **before** and **after** deployment is critical to avoid incidents and increase the success rate of changes



* Source: [ITSM.tools](https://www.it-sm.com/), figures from Gartner and Forrester, 2017

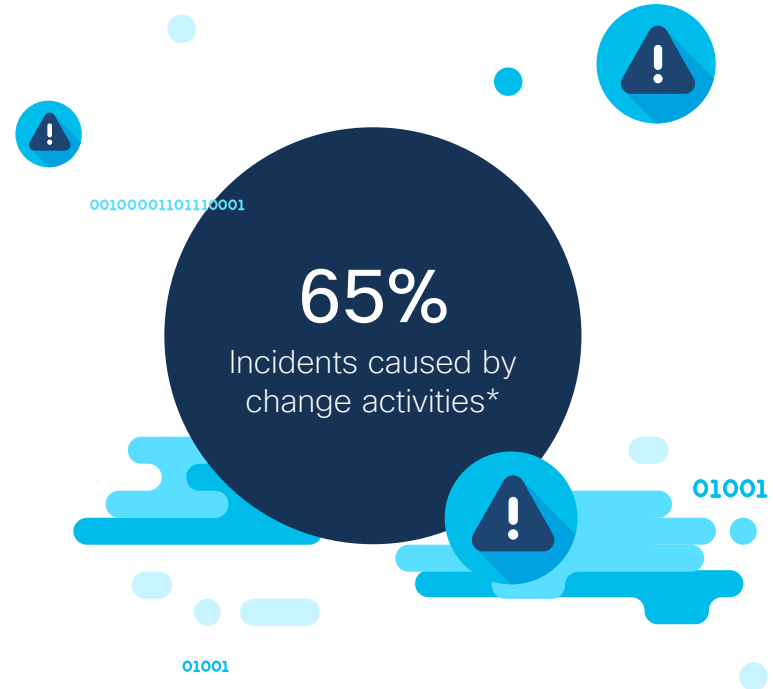
Most of incidents are caused by change activities

Verifying the impact of changes **before** and **after** deployment is critical to avoid incidents and increase the success rate of changes



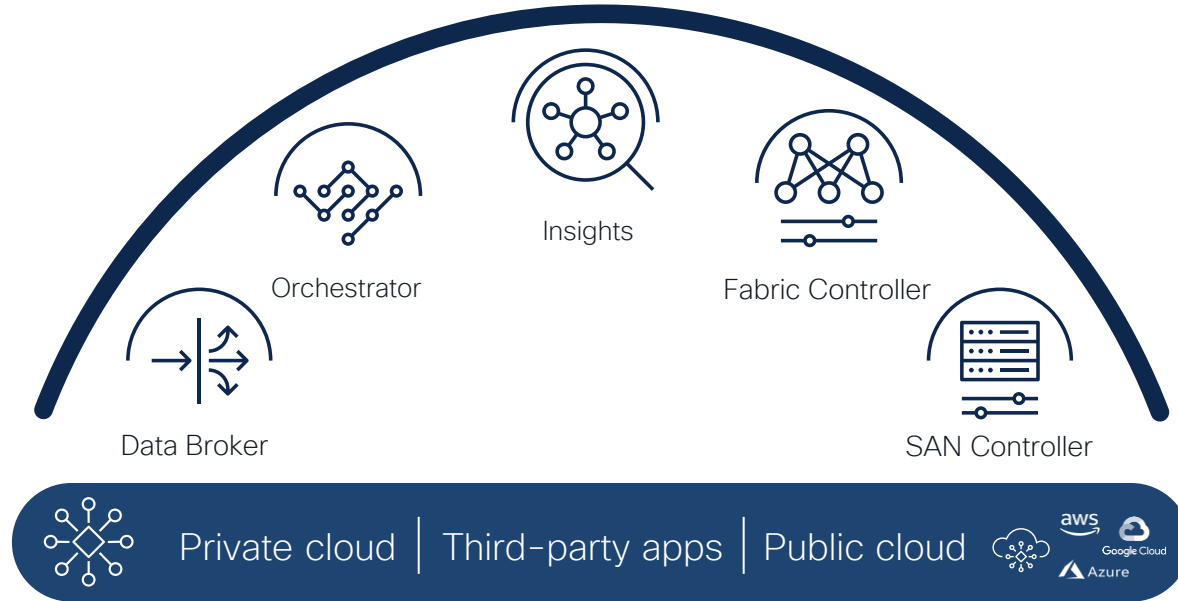
Nexus Dashboard Insights
can help us here! 😊

* Source: [ITSM.tools](https://www.it-sm.com/), figures from Gartner and Forrester, 2017



Cisco Nexus Dashboard

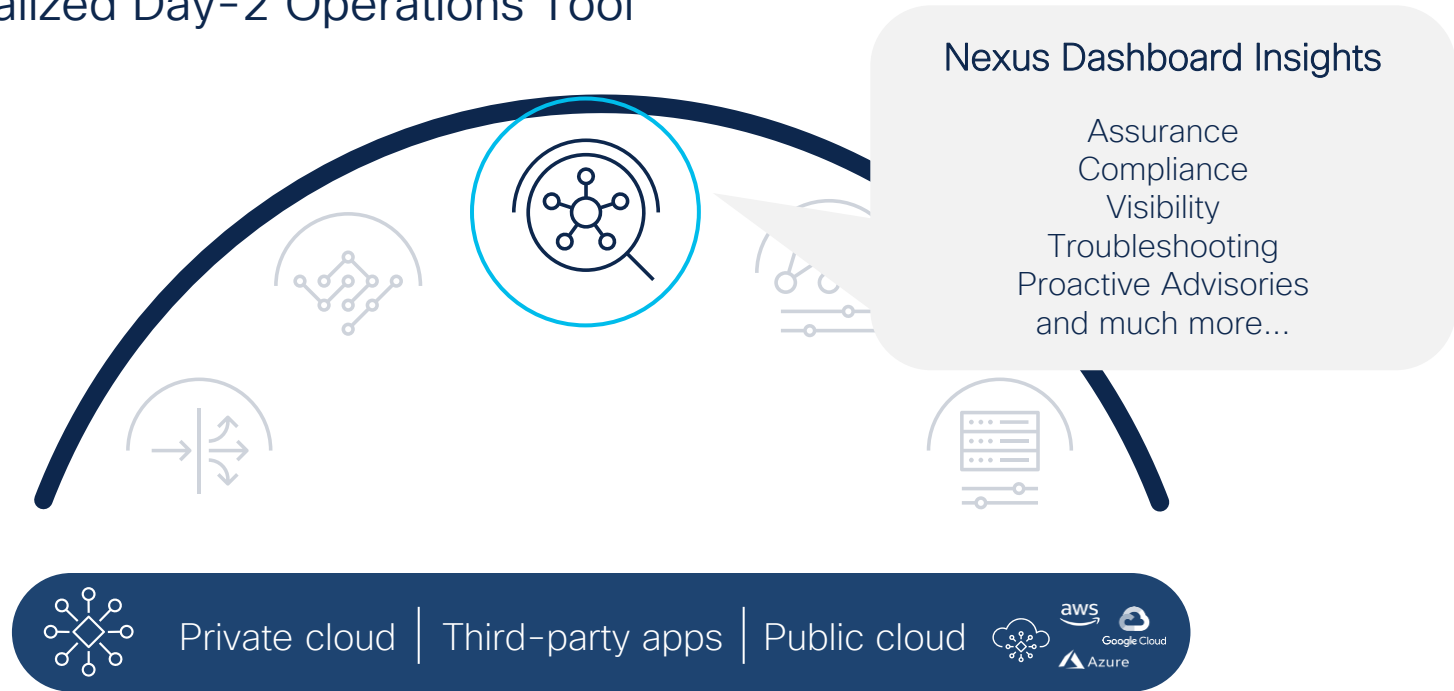
Simple to automate, simple to consume



Consume all services **in one place**

Cisco Nexus Dashboard **Insights**

Your Centralized Day-2 Operations Tool



Consume all services **in one place**

Cisco Nexus Dashboard Insights

Current Feature Set

Visibility and monitoring



Topology



Capacity planning



Control plane statistics



Microburst detection



Explorer queries



Interface statistics

Analytics and Correlation



Flow telemetry



Assurance



Endpoint analytics



Integrations



Delta and pre-change analysis



Anomalies

Advisories and tools



Conformance and lifecycle



PSIRT notification



TAC assist



Upgrade analysis



Field notices



Bug scan

Cisco Nexus Dashboard Insights

Current Feature Set

Visibility and monitoring



Topology



Capacity planning



Control plane statistics



Microburst detection



Explorer queries



Interface statistics

Analytics and Correlation



Flow telemetry



Integrations



Assurance



Delta and pre-change analysis



Endpoint analytics



Anomalies

Advisories and tools



Conformance and lifecycle



Upgrades



PSIRT notification



Field notices



TAC assist

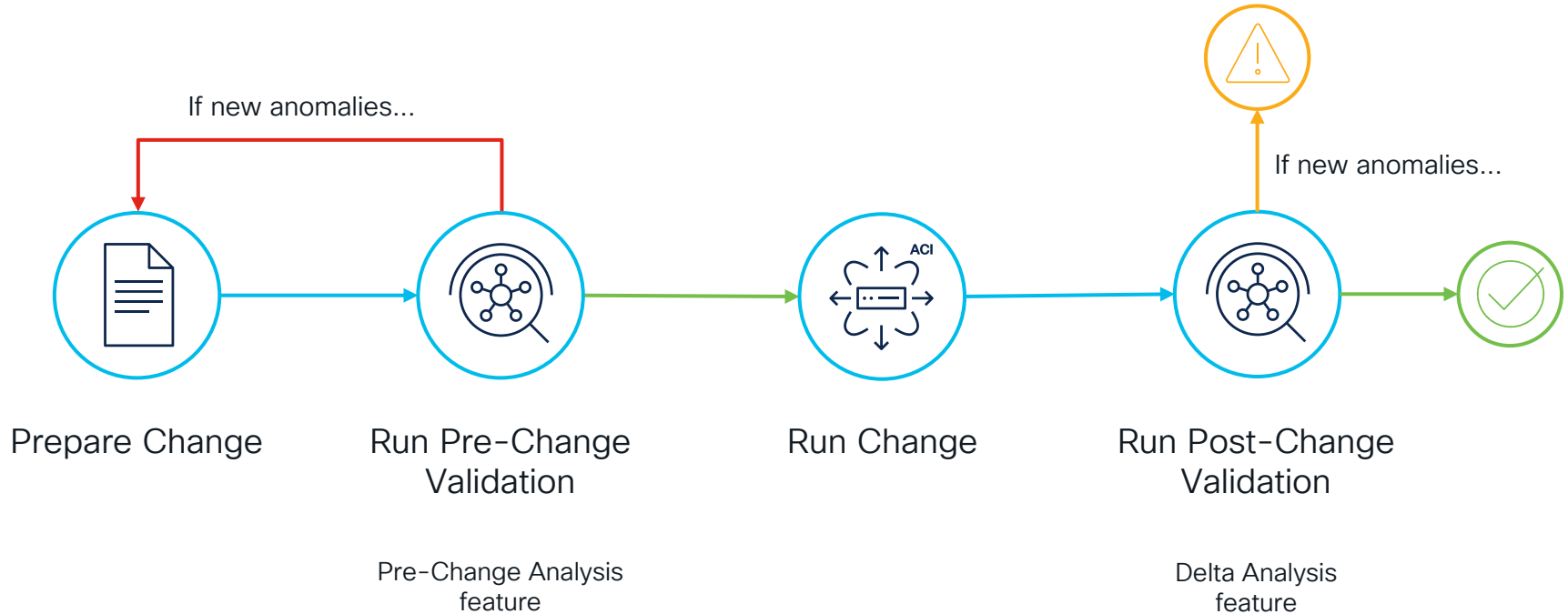


Bug scan

This is our focus today

Verify your changes before and after

With Nexus Dashboard Insights



How can I build
a simple yet
powerful CI/CD
Pipeline?

Our CI/CD toolset

The tools we will use for today's session



GitHub Actions

Version Control System

SaaS-based CI engine



Ansible

Deployment tool

Support for check-only mode



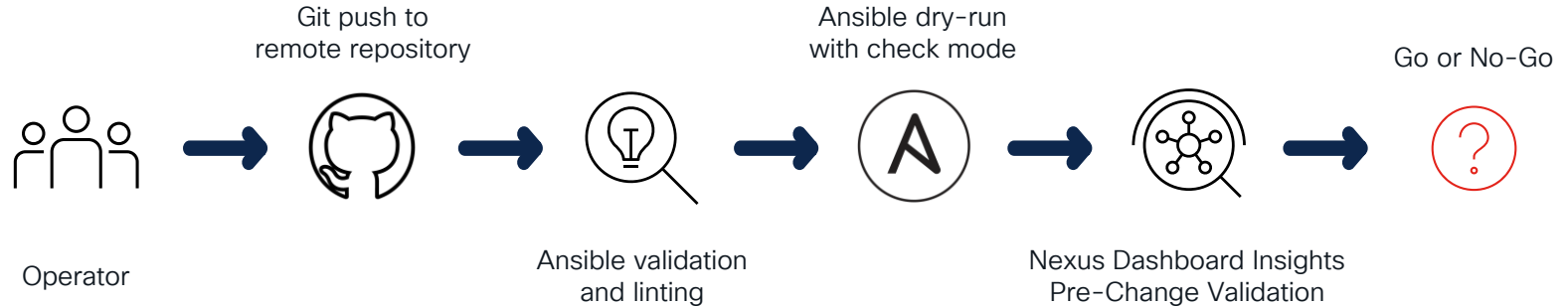
Nexus Dashboard Insights

Pre-change Validation

Post-Change validation

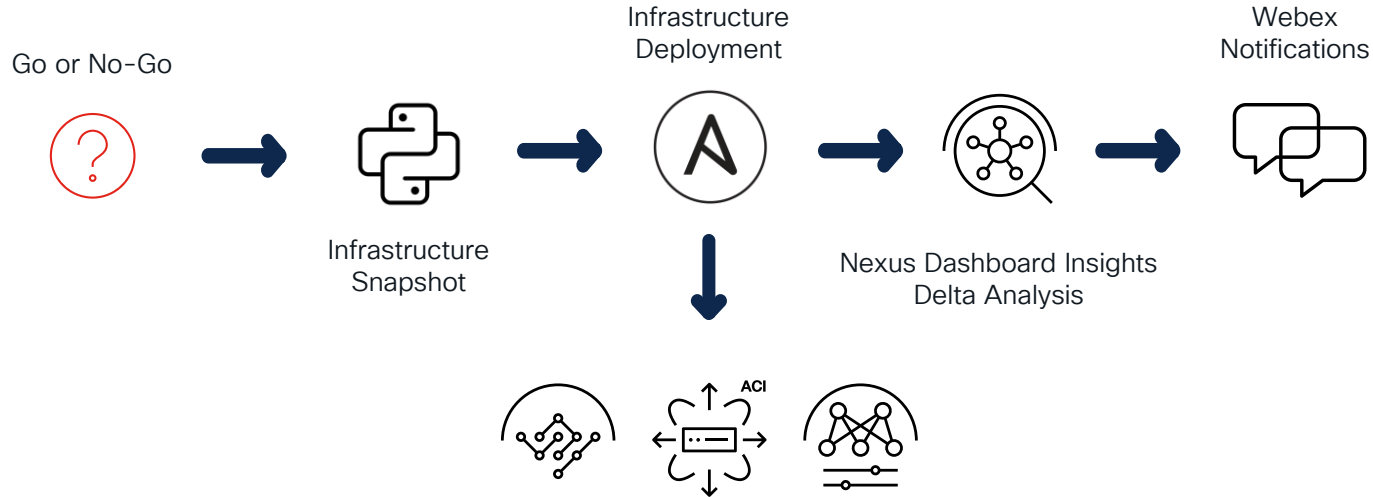
Our target CI/CD Pipeline

The goal for today's session



Our target CI/CD Pipeline

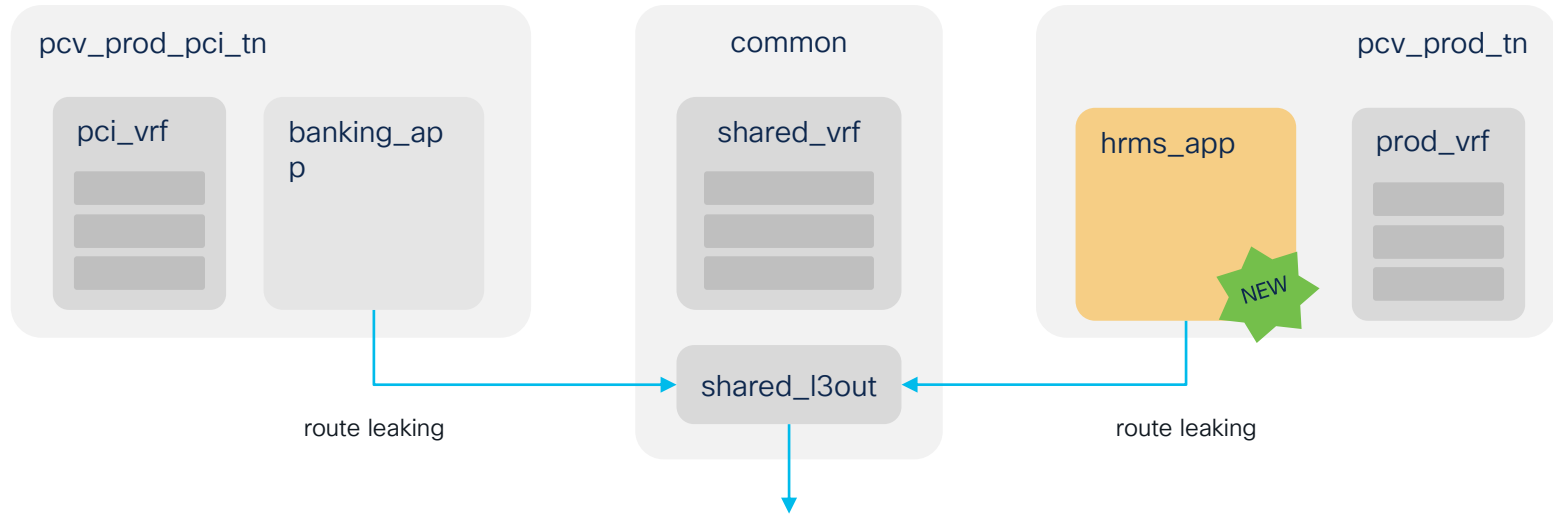
The goal for today's session





Demo

Demo Scenario



Conclusion

Key points to remember

- Networking CI/CD Pipelines are fundamental to implement the network provisioning process in a consistent and automated way
- A high percentage of incidents are caused due to change activities
 - Hence, including pre/post-change validations in your pipeline is strongly recommended
- Nexus Dashboard Insights can be integrated in your pipeline for pre-change and post-change validations in a very simple way

Key points to remember

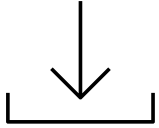
- Networking CI/CD Pipelines are fundamental to implement the network provisioning process in a consistent and automated way
- A high percentage of incidents are caused due to change activities
 - Hence, including pre/post-change validations in your pipeline is strongly recommended
- Nexus Dashboard Insights can be integrated in your pipeline for pre-change and post-change validations in a very simple way



Building a powerful CI/CD Pipeline is simple
if you have Nexus Dashboard Insights

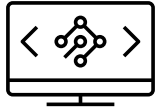
Next Steps

What to do during and after Cisco Live



Download this code from GitHub and get familiar with it

[Code in GitHub](#)



Test the code in your lab (or use DevNet Sandbox)

[DevNet ACI Sandboxes](#)



Explore more about Nexus Dashboard Insights in Cisco Live

[DEVNET-1369](#) | [BRKDCN-2673](#)

Fill out your session surveys!



Attendees who fill out a minimum of four session surveys and the overall event survey will get **Cisco Live-branded socks** (while supplies last)!



Attendees will also earn 100 points in the **Cisco Challenge** for every survey completed.



These points help you get on the leaderboard and increase your chances of winning daily and grand prizes

Continue your education



- Visit the Cisco Showcase for related demos
- Book your one-on-one Meet the Engineer 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

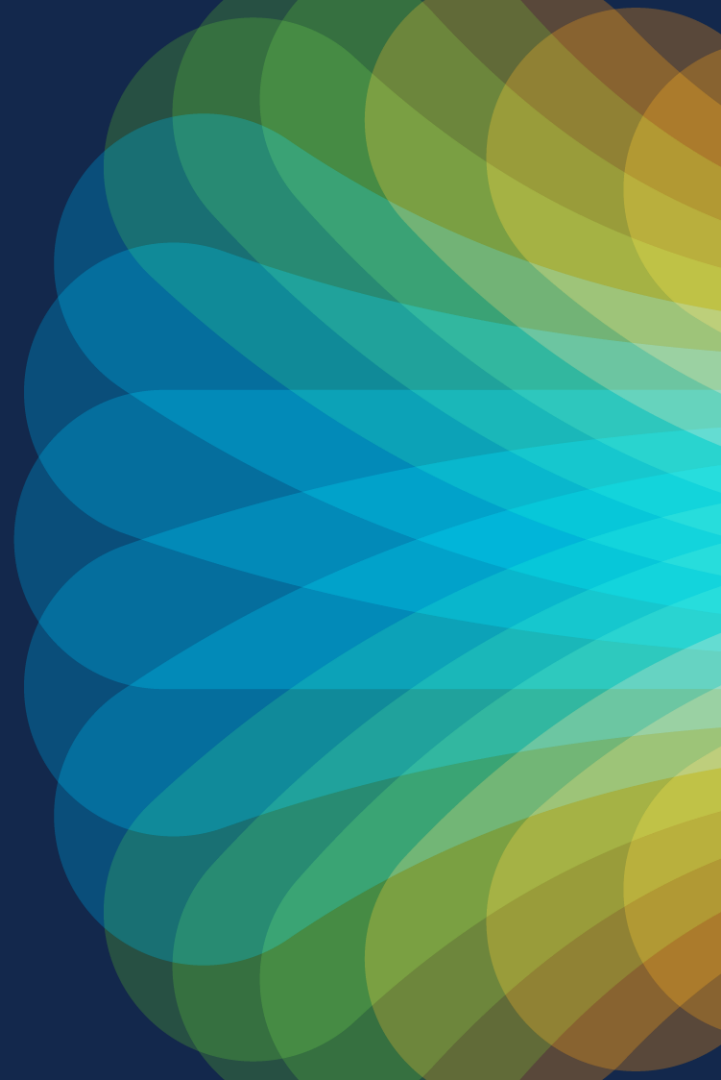


The bridge to possible

Thank you



#CiscoLive

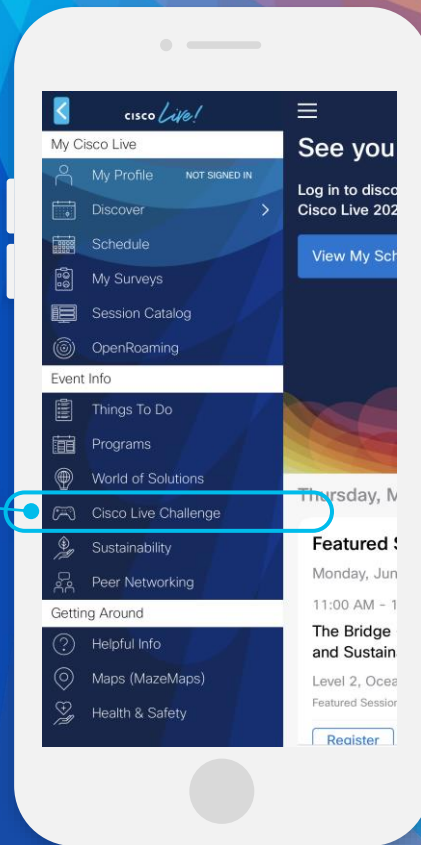


Cisco Live Challenge

Gamify your Cisco Live experience!
Get points for attending this session!

How:

- 1 Open the Cisco Events App.
- 2 Click on 'Cisco Live Challenge' in the side menu.
- 3 Click on View Your Badges at the top.
- 4 Click the + at the bottom of the screen and scan the QR code:



The background features a vibrant, multi-colored abstract design. On the left, there are overlapping, wavy, organic shapes in shades of red, orange, and yellow. On the right, a bright white light source emits a series of sharp, radiating lines in various colors, including blue, green, and yellow, creating a sunburst effect. The overall composition is dynamic and energetic.

cisco *Live!*

Let's go

#CiscoLive

Appendix I

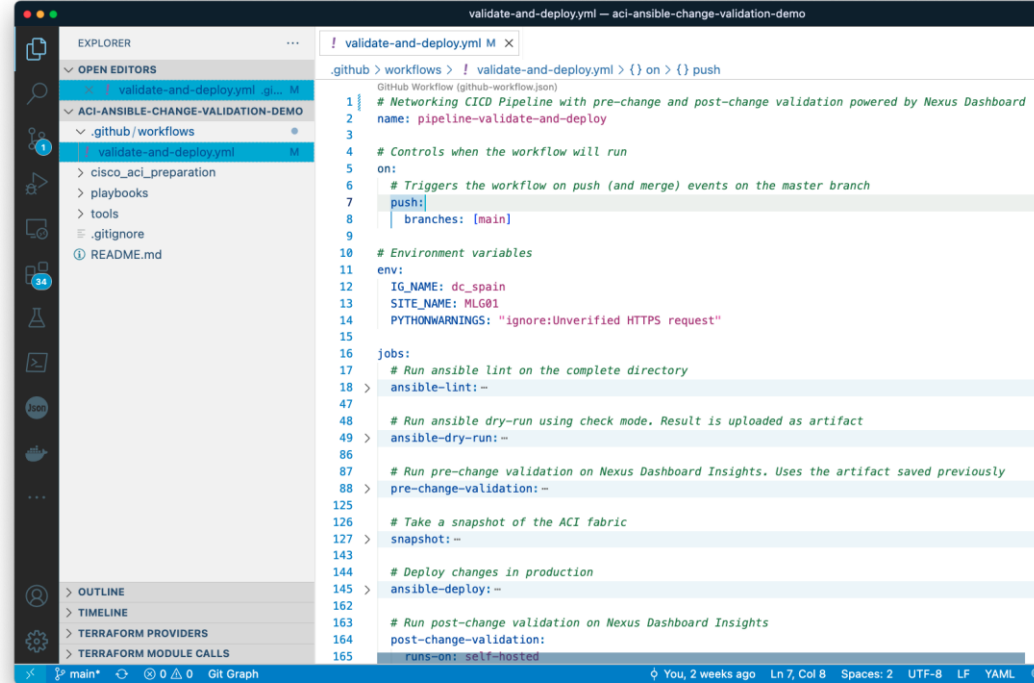
Building the Pipeline

Building the Pipeline

How to get started

- The pipeline is defined in a YAML file created in:

Project-X
 .github
 workflows



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a '.github/workflows' directory containing 'validate-and-deploy.yml'. The code editor displays the content of 'validate-and-deploy.yml', which is a GitHub Actions workflow. The workflow is triggered on push to the main branch and runs on a self-hosted runner. It includes jobs for linting, dry-run, pre-change validation, snapshotting, and deployment.

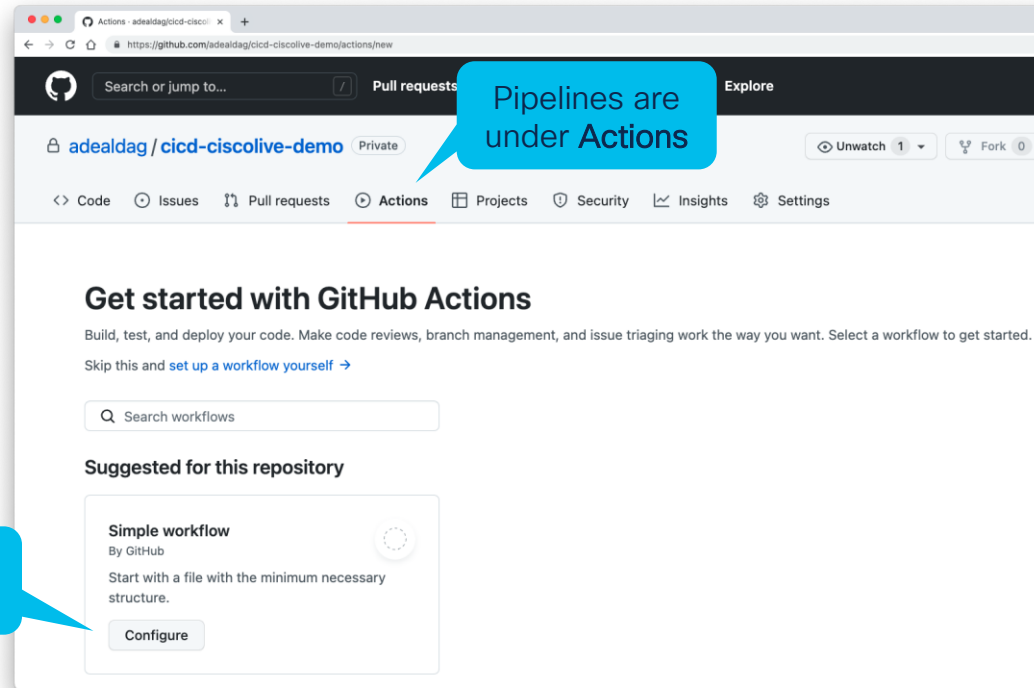
```
github > workflows > ! validate-and-deploy.yml > {} on > {} push

GitHub Workflow (github-workflow.json)
1 | # Networking CI/CD Pipeline with pre-change and post-change validation powered by Nexus Dashboard
2 | name: pipeline-validate-and-deploy
3 |
4 | # Controls when the workflow will run
5 | on:
6 |   # Triggers the workflow on push (and merge) events on the master branch
7 |   push:
8 |     branches: [main]
9 |
10 | # Environment variables
11 | env:
12 |   IG_NAME: dc_spain
13 |   SITE_NAME: MLG01
14 |   PYTHONWARNINGS: "ignore:Unverified HTTPS request"
15 |
16 | jobs:
17 |   # Run ansible lint on the complete directory
18 |   ansible-lint:--
19 |
20 |   # Run ansible dry-run using check mode, Result is uploaded as artifact
21 |   ansible-dry-run:--
22 |
23 |   # Run pre-change validation on Nexus Dashboard Insights. Uses the artifact saved previously
24 |   pre-change-validation:--
25 |
26 |   # Take a snapshot of the ACI fabric
27 |   snapshot:--
28 |
29 |   # Deploy changes in production
30 |   ansible-deploy:--
31 |
32 |   # Run post-change validation on Nexus Dashboard Insights
33 |   post-change-validation:
34 |     runs-on: self-hosted
```

Building the Pipeline

How to get started

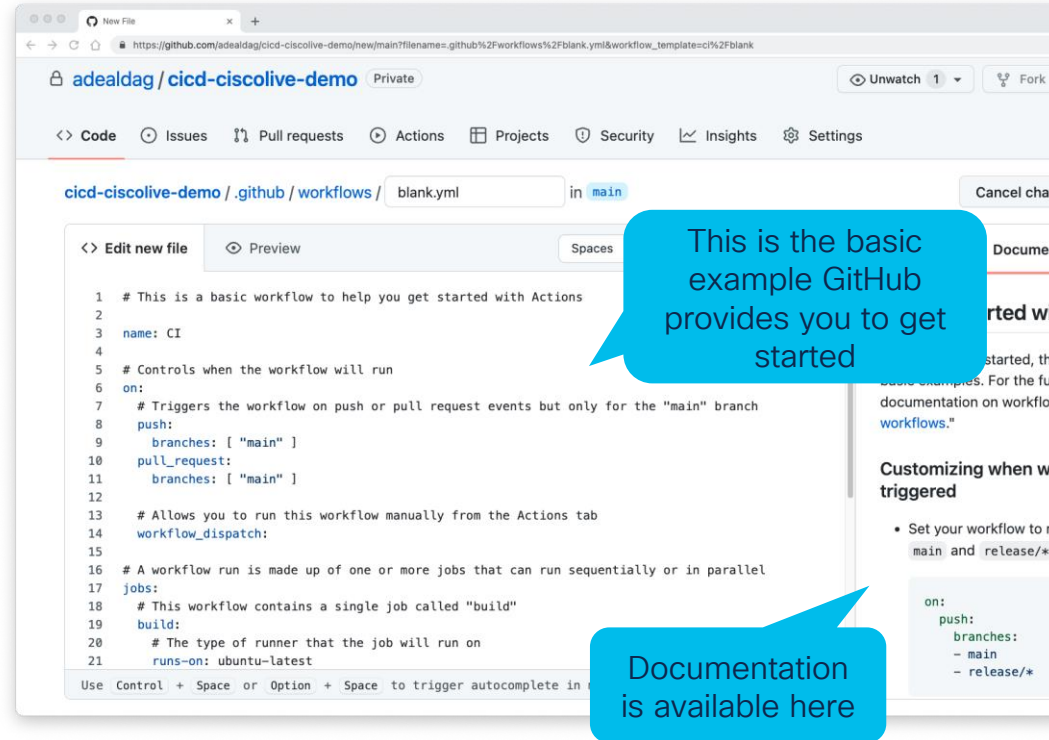
- The pipeline can also be created from GitHub
 - From there you can get a basic workflow template to get started
 - Documentation is presented in a side panel on the right



Building the Pipeline

How to get started

- The pipeline can also be created from GitHub
 - From there you can get a basic workflow template to get started
 - Documentation is presented in a side panel on the right



Building the Pipeline

Configuring our pipeline

```
# Networking CICD Pipeline with pre-change and post-change dashboard
name: pipeline-validate-and-deploy

# Controls when the workflow will run
on:
  # Triggers the workflow on push (and merge) events on the "main" branch
  push:
    branches: [main]

# Allows you to run this workflow manually from the Actions tab
# workflow_dispatch:

# Environment variables
env:
  PYTHONWARNINGS: "ignore:Unverified HTTPS request"

# A workflow run is made up of one or more jobs that can run sequentially or in parallel
jobs:
  [...]
```

Workflow (or pipeline)
name

Defines when the
workflow will get triggered

If you need to run the
workflow manually,
uncomment this

Include any required
environment variable here

Building the Pipeline

Configuring the “jobs”

```
jobs:
  # Run ansible lint on the complete directory
  ansible-lint:
    [...]
  # Run ansible dry-run using check mode. Result is uploaded as artifact
  ansible-dry-run:
    [...]
  # Run pre-change validation on Nexus Dashboard Insights. Uses the artifact saved previously
  pre-change-validation:
    [...]
  # Take a snapshot of the ACI fabric
  snapshot:
    [...]
  # Deploy changes in production
  ansible-deploy:
    [...]
  # Run post-change validation on Nexus Dashboard Insights
  post-change-validation:
    [...]
```

These are the different stages in our pipeline or workflow

Jobs can run in parallel or sequentially, based on dependencies configured

Building the Pipeline

Job: ansible-lint

Run ansible lint on the complete directory

ansible-lint:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2

- name: Set up Python

uses: actions/setup-python@v2

with:

python-version: 3.8

- name: Install yamllint

run: pip install yamllint

- name: Lint YAML files

run: yamllint ./playbooks

- name: Send Webex Notification

[...]

Runs in a cloud-hosted runner maintained by GitHub

Checks-out your repository under \$GITHUB_WORKSPACE, so your workflow can access it

Installs and sets-up a version of python, add it to the PATH and more

With “run” you can run any command on the runner. Here we install *yamllint* and then run against our playbooks

More about Webex notifications later!

Building the Pipeline

Job: ansible-dry-run

Run ansible dry-run using check mode. Result

ansible-dry-run:

runs-on: self-hosted

needs: ansible-lint

container: adealdag/ansible:latest

steps:

- uses: actions/checkout@v2

- name: Run ansible playbook dry-run

env:

VAULT_KEY: \${ secrets.VAULT_KEY }

run: /

ansible --version

echo \$VAULT_KEY > vault.key

ansible-playbook -i inventory.yaml --vault-password-file vault.key deploy.yaml --check

rm vault.key

working-directory: playbooks

[...]

Runs in a **self-hosted runner**, installed in our on-prem infrastructure

With **needs** dependencies can be defined.
This job will not run until 'ansible-lint' completes

If the self-hosted runner has docker installed, you can **run each job in a container**, that gets destroyed after execution (recommended)

Building the Pipeline

Job: ansible-dry-run (continuation)

steps:

- uses: actions/checkout@v2

- name: Run ansible playbook dry-run

env:

VAULT_KEY: \${ secrets.VAULT_KEY }

run: /

ansible --version

echo \$VAULT_KEY > vault.key

ansible-playbook -i inventory.yaml --vault-password-file vault.key deploy.yaml --check

rm vault.key

working-directory: playbooks

- name: Upload artifact

uses: actions/upload-artifact@v2

with:

name: config-dump

path: playbooks/dryrun_data.json

- name: Send Webex Notification

[...]

Inventory is encrypted using ansible vault.
Vault key is stored as an **action secret**.
More on secrets later!

Directory from where these commands are run

Saves output file from previous step in an artifact.
Artifacts are the way to share data between jobs.

Building the Pipeline

Dissecting ansible-playbook command

Inventory is encrypted using **Ansible Vault** not to disclose sensitive information in GitHub repo

Our playbook.
It contains a series of “import_playbook” statements to combine multiple playbooks in a single one

```
ansible-playbook -i inventory.yaml --vault-password-file vault.key deploy.yaml --check
```

Ansible Vault password is provided in a file.
This file contains the password stored in GitHub action secret

Runs ansible in ‘**check mode**’ *

In ‘check mode’, no changes are made on remote systems, it is just a simulation, or ‘dry-run’

* More info [here](#)

Building the Pipeline

Job: pre-change-validation

Run pre-change validation on Nexus Dashboard Insights. Uses the artifact saved previously

pre-change-validation:

runs-on: self-hosted

needs: ansible-dry-run

container: adealdag/ansible:latest

steps:

- uses: actions/checkout@v2

- name: Download artifact

uses: actions/download-artifact@v2

with:

name: config-dump

path: tools/change-validation

- name: Run pre-change analysis playbook
[...]

Runs in a **self-hosted runner**, in a **container**, and needs previous job to be completed

Downloads output file from previous step, saved on an artifact called 'config-dump', into folder specified in 'path'



adealdag/ansible container have required ansible collections pre-installed

Building the Pipeline

Job: pre-change-validation (continuation)

Run pre-change validation on Nexus Dashboard Insights. Uses the artifact saved previously

pre-change-validation:

[...]

steps:

[...]

- name: Run pre-change analysis playbook

env:

VAULT_KEY: \${ secrets.VAULT_KEY }

run: /

ansible --version

rm -rf \$HOME/.ansible/pc

echo \$VAULT_KEY > vault.key

ansible-playbook -i inventory.yaml --vault-password-file vault.key pre-change-validation.yaml

rm vault.key

working-directory: tools/change-validation

- name: Send Webex Notification

[...]

Runs pre-change-validation on Nexus Dashboard Insights using **cisco.nd Ansible Collection**

This is a workaround to avoid issues with ansible control plane sockets seen when running from a workflow runner

Building the Pipeline

Job: snapshot

Take a snapshot of the ACI fabric

snapshot:

runs-on: self-hosted

needs: pre-change-validation

container: adealdag/aci_cobra:5.2.4e

steps:

- uses: actions/checkout@v2

- name: Take an aci snapshot

env:

APIC_HOST: \${ secrets.APIC_HOST }

APIC_USERNAME: \${ secrets.APIC_USERNAME }

APIC_PASSWORD: \${ secrets.APIC_PASSWORD }

run: /

export no_proxy=\$APIC_HOST,\$no_proxy

python ./tools/trigger_backup/py_trigger_backup.py

Runs in a **self-hosted runner**, in a **container**, and needs previous job to be completed

Runs a python script to take an ACI snapshot
The script uses Cobra SDK



adealdag/cobra container have
Cobra SDK pre-installed

Building the Pipeline

Job: ansible-deploy

Deploy changes in production

ansible-deploy:

runs-on: self-hosted

needs: snapshot

container: adealdag/ansible:latest

steps:

- uses: actions/checkout@v2

- name: Run ansible playbook

env:

VAULT_KEY: \${ secrets.VAULT_KEY }

run: /

ansible --version

echo \$VAULT_KEY > vault.key

ansible-playbook -i inventory.yaml --vault-password-file vault.key deploy.yaml

rm vault.key

working-directory: playbooks

- name: Send Webex Notification

[...]

Runs in a **self-hosted runner**, in a **container**, and needs previous job to be completed

Runs Ansible playbook, now **without -check**

Building the Pipeline

Job: post-change-validation

Run post-change validation on Nexus Dashboard Insights

post-change-validation:

runs-on: self-hosted

needs: ansible-deploy

container: adealdag/ansible:latest

Runs in a **self-hosted runner**, in a **container**, and needs previous job to be completed

steps:

- uses: actions/checkout@v2

- name: Run post-change analysis playbook

env:

VAULT_KEY: \${ secrets.VAULT_KEY }

run: /

ansible --version

rm -rf \$HOME/.ansible/pc

echo \$VAULT_KEY > vault.key

ansible-playbook -i inventory.yaml --vault-password-file vault.key post-change-validation.yaml

rm vault.key

working-directory: tools/change-validation

Runs post-change-validation on Nexus Dashboard Insights using **cisco.nd Ansible Collection**

This is a workaround to avoid issues with ansible control plane sockets seen when running from a workflow runner

- name: Send Webex Notification

[...]

Building the Pipeline

Webex: Sending notifications in Webex

```
- name: Send Webex Notification
  uses: adealdag/action-webex-notification@python-v1
  if: always()
  with:
    webexToken: ${ secrets.WEBEX_TOKEN }
    roomID: ${ secrets.ROOM_ID }
    markdownMsg: /
    ### [${ github.job }] ${ github.repository } - ${ github.event.head_commit.message }
    * Trigger: ${ github.event_name }
    * Git SHA: ${ github.sha }
    * Status: ${ job.status }
    * Details URL:
      [Job Results](https://github.com/${ github.repository }}/actions/runs/${ github.run_id })
```

This uses a **custom action**.
It is publicly available, code can be checked [here](#)

Runs always, regardless if previous steps
succeeded or failed

It is recommended to use this action together with
a Webex Bot.

Building the Pipeline

Webex: Sending notifications in Webex

```
- name: Send Webex Notification
  uses: adealdag/action-webex-notification@python-v1
  if: always()
  with:
    webexToken: ${{ secrets.WEBEX_TOKEN }}
    roomID: ${{ secrets.ROOM_ID }}
    markdownMsg: /
      ### [${{ github.job }}] ${{ github.repository }} - ${{ github.event.head_commit.message }}
      * Trigger: ${{ github.event_name }}
      * Git SHA: ${{ github.sha }}
      * Status: ${{ job.status }}
      * Details URL:
      [Job P...
```

This uses a **custom action**.
It is publicly available, code can be checked [here](#)

Runs always, regardless if previous steps
succeeded or failed

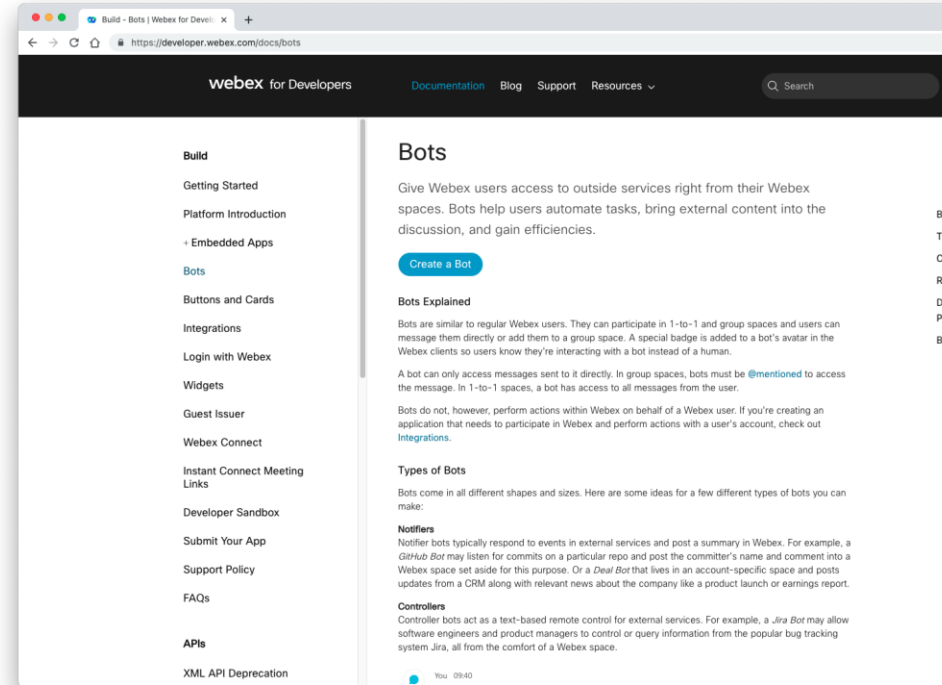


You can also create and publish your own actions

Building the Pipeline

Webex: Creating your bot

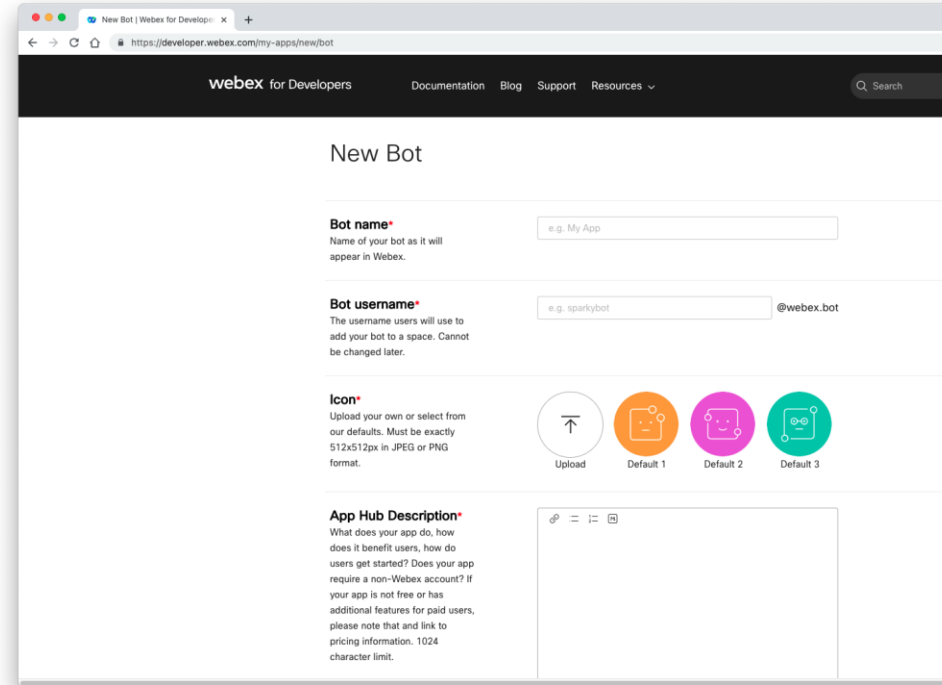
- Create your bot in:
 - <https://developer.webex.com/docs/bots>
- After creating your bot, you will get your Webex Token
 - Store this safely, we'll need it for authenticating REST API calls
- Now add your bot to a Webex Room, and you are ready to go



Building the Pipeline

Webex: Creating your bot

- Create your bot in:
 - <https://developer.webex.com/docs/bots>
- After creating your bot, you will get your Webex Token
 - Store this safely, we'll need it for authenticating REST API calls
- Now add your bot to a Webex Room, and you are ready to go



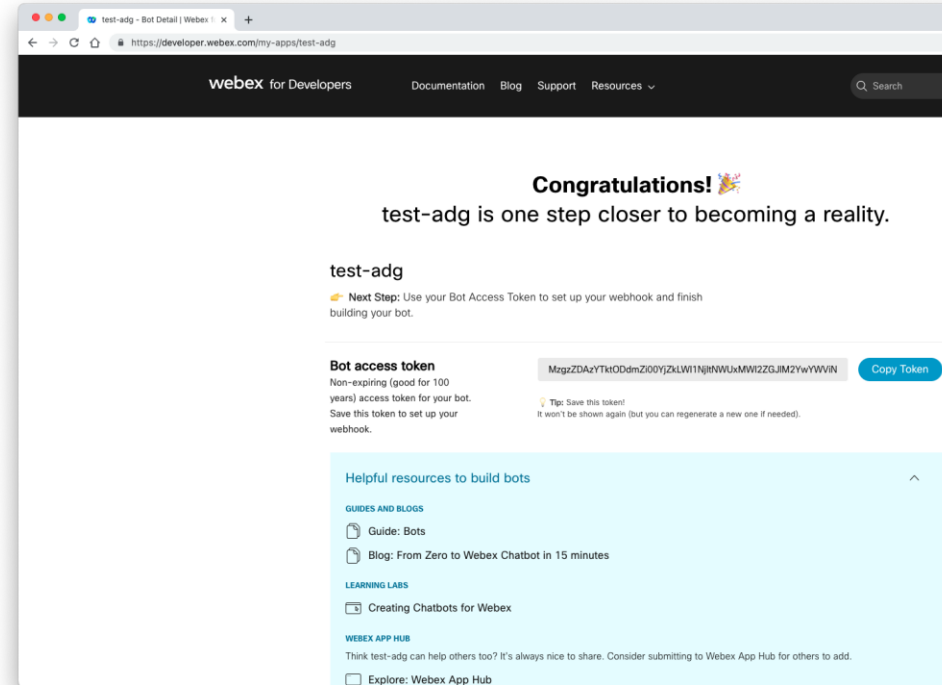
The screenshot shows the 'New Bot' page on the Webex Developer Portal. The page has a dark header with the 'webex for Developers' logo and navigation links for Documentation, Blog, Support, and Resources. A search bar is on the right. The main content area is titled 'New Bot' and contains several form fields:

- Bot name***: A text input field with the placeholder 'e.g. My App'. Below it, a note says 'Name of your bot as it will appear in Webex.'
- Bot username***: A text input field with the placeholder 'e.g. sparkybot' and a dropdown menu showing '@webex.bot'. Below it, a note says 'The username users will use to add your bot to a space. Cannot be changed later.'
- Icon***: A section for uploading an icon. It includes a note: 'Upload your own or select from our defaults. Must be exactly 512x512px in JPEG or PNG format.' Below this are four circular icons: 'Upload' (with an upward arrow), 'Default 1' (orange with a robot face), 'Default 2' (pink with a robot face), and 'Default 3' (teal with a robot face).
- App Hub Description***: A text area for a description. It includes a note: 'What does your app do, how does it benefit users, how do users get started? Does your app require a non-Webex account? If your app is not free or has additional features for paid users, please note that and link to pricing information. 1024 character limit.'

Building the Pipeline

Webex: Creating your bot

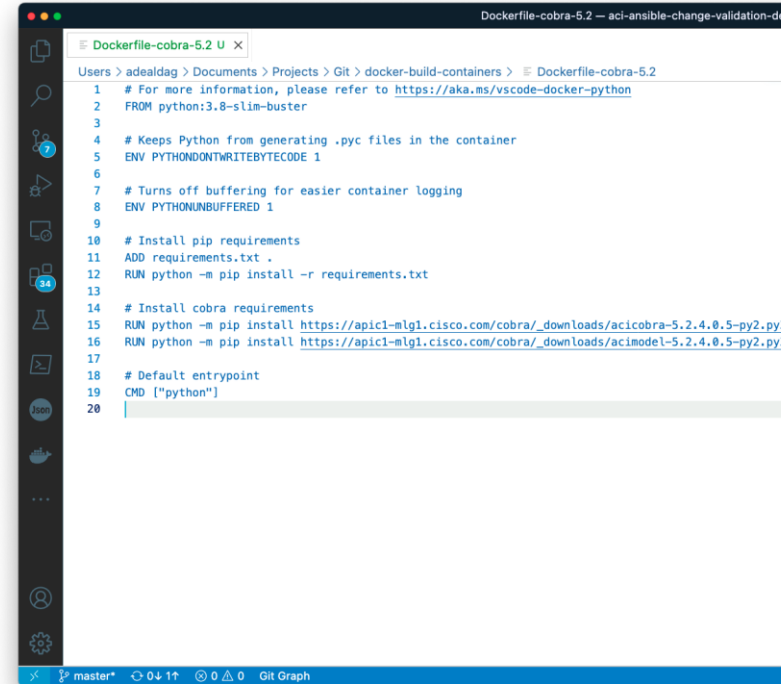
- Create your bot in:
 - <https://developer.webex.com/docs/bots>
- After creating your bot, you will get your Webex Token
 - Store this safely, we'll need it for authenticating REST API calls
- Now add your bot to a Webex Room, and you are ready to go



Building the Pipeline

A look at the containers used

- You can build your own containers for GitHub Actions workflows
- Steps:
 - Write your Dockerfile
 - Build your container locally
 - `docker build -t adealdag/ansible:v2 --platform linux/amd64 .`
 - Push the container to Docker Hub
 - `docker push adealdag/ansible:v2`
 - Your container is ready to use

A screenshot of a code editor window showing a Dockerfile. The title bar indicates the file is 'Dockerfile-cobra-5.2 U'. The breadcrumb navigation shows the path: 'Users > adealdag > Documents > Projects > Git > docker-build-containers > Dockerfile-cobra-5.2'. The code is a Dockerfile for a Python-based container. It starts with a comment and a link to a VS Code Docker Python tutorial. The base image is 'python:3.8-slim-buster'. It then sets environment variables to prevent Python from generating .pyc files and to turn off buffering for logging. Next, it installs pip requirements from a local file 'requirements.txt'. Then, it installs cobra requirements from two URLs. Finally, it sets the default entrypoint to 'python'. The line numbers 1 through 20 are visible on the left side of the code editor.

```
1 # For more information, please refer to https://aka.ms/vscode-docker-python
2 FROM python:3.8-slim-buster
3
4 # Keeps Python from generating .pyc files in the container
5 ENV PYTHONDONTWRITEBYTECODE 1
6
7 # Turns off buffering for easier container logging
8 ENV PYTHONUNBUFFERED 1
9
10 # Install pip requirements
11 ADD requirements.txt .
12 RUN python -m pip install -r requirements.txt
13
14 # Install cobra requirements
15 RUN python -m pip install https://apic1-mlg1.cisco.com/cobra/\_downloads/acicobra-5.2.4.0.5-py2.py
16 RUN python -m pip install https://apic1-mlg1.cisco.com/cobra/\_downloads/acimodel-5.2.4.0.5-py2.py
17
18 # Default entrypoint
19 CMD ["python"]
20
```

Building the Pipeline

A look at the containers used – aci_cobra:xxx

*# For more information, please refer to <https://aka.ms/vscode-docker-python>
FROM python:3.8-slim-buster*

*# Keeps Python from generating .pyc files in the container
ENV PYTHONDONTWRITEBYTECODE 1*

*# Turns off buffering for easier container logging
ENV PYTHONUNBUFFERED 1*

*# Install pip requirements
ADD requirements.txt .
RUN python -m pip install -r requirements.txt*

*# Install cobra requirements
RUN python -m pip install https://apic1-mlg1.cisco.com/cobra/_downloads/acicobra-5.2.4.0.5-py2.py3-none-any.whl --trusted-host apic1-mlg1.cisco.com
RUN python -m pip install https://apic1-mlg1.cisco.com/cobra/_downloads/acimodel-5.2.4.0.5-py2.py3-none-any.whl --trusted-host apic1-mlg1.cisco.com*

*# Default entrypoint
CMD ["python"]*

Replace with your cobra SDK version.

In this example, it gets downloaded directly from APIC

Building the Pipeline

A look at the containers used – ansible:xxx

```
# For more information, please refer to https://aka.ms/vscode-docker-python  
FROM python:3.8-slim-buster
```

```
# Keeps Python from generating .pyc files in the container  
ENV PYTHONDONTWRITEBYTECODE 1
```

```
# Turns off buffering for easier container logging  
ENV PYTHONUNBUFFERED 1
```

```
# Install packages  
RUN apt-get update  
RUN apt-get --yes --force-yes install build-essential
```

```
# Install pip requirements  
ADD requirements.txt .  
RUN python -m pip install -r requirements.txt
```

```
# Install collections  
RUN ansible-galaxy collection install cisco.aci -p /usr/share/ansible/collections  
RUN ansible-galaxy collection install cisco.mso -p /usr/share/ansible/collections  
RUN ansible-galaxy collection install cisco.nd -p /usr/share/ansible/collections
```

```
# Default endpoint  
CMD ["ansible-playbook", "--version"]
```

requirements.txt

```
setuptools  
ansible  
paramiko  
requests  
requests-toolbelt  
jsonpath_ng  
pathlib  
filelock  
lxml  
xmljson
```

Appendix II

Cisco ACI Playbooks

Cisco ACI Playbook

How to prepare them for dry-run

- Cisco ACI modules in Ansible support check mode
 - When using check mode...
 - Changes are not pushed to the fabric
 - Changes are populated in an output file

```
hrms_app_playbook.yml -- aci-ansible-change-validation-demo

EXPLORER
  OPEN EDITORS
    hrms_app_playbook.yml playbo...
  ACI-ANSIBLE-CHANGE-VALIDATION-DEMO
    .github/workflows
    ! validate-and-deploy.yml M
    cisco_aci_preparation
    playbooks
      deploy.yml
      hrms_app_playbook.yml
      hrms_app_vars.yml
      inventory.yml
      tn_prod_playbook.yml
      tn_prod_vars.yml
    tools
    .gitignore
    README.md
  OUTLINE
  TIMELINE
  TERRAFORM PROVIDERS
  TERRAFORM MODULE CALLS

hrms_app_playbook.yml
6 connection: local
7 gather_facts: no
8
9 vars:
10 aci_login: &aci_login
11 host: "{{ ansible_host }}"
12 username: "{{ aci_username }}"
13 password: "{{ aci_password | default(omit) }}"
14 private_key: "{{ aci_private_key | default(omit) }}"
15 certificate_name: "{{ aci_certificate_name | default(omit) }}"
16 validate_certs: "{{ aci_validate_certs }}"
17 annotation: "orchestrator:ansible"
18 state: present
19 output_path: dryrun_data.json
20
21 vars_files:
22 - ./hrms_app_vars.yml
23
24 tasks:
25 - name: Add Application Profile
26   cisco.aci.aci_ap:
27     <<: *aci_login
28     tenant: "{{ tenant_name }}"
29     ap: "{{ app_name }}"
30
31 - name: Add EPG
32   cisco.aci.aci_epg:
33     <<: *aci_login
34     tenant: "{{ tenant_name }}"
35     ap: "{{ app_name }}"
```


Cisco ACI Playbook

How to prepare them for dry-run

tasks:

- **name:** Add Application Profile

cisco.aci.aci_ap:

host: "{{ ansible_host }}"

username: "{{ aci_username }}"

password: "{{ aci_password }}"

validate_certs: "{{ aci_validate_certs }}"

tenant: "{{ tenant_name }}"

ap: "{{ app_name }}"

annotation: "orchestrator:ansible"

output_path: dryrun_data.json

state: present

Include **output_path**
attribute in every task

Value is the path to the
JSON file where changes
will be saved

Cisco ACI Playbook

How to prepare them for dry-run

- *name*: Create HRMS App Profile (prod)
[...]

vars:

aci_login: &aci_login
host: "{{ ansible_host }}"
username: "{{ aci_username }}"
password: "{{ aci_password }}"
validate_certs: "{{ aci_validate_certs }}"
annotation: "orchestrator:ansible"
output_path: dryrun_data.json

tasks:

- *name*: Add Application Profile
cisco.aci.aci_ap:
 <: *aci_login
 tenant: "{{ tenant_name }}"
 ap: "{{ app_name }}"
 state: present

Recommended

Use an **anchor** to apply this
to every task in one single
action

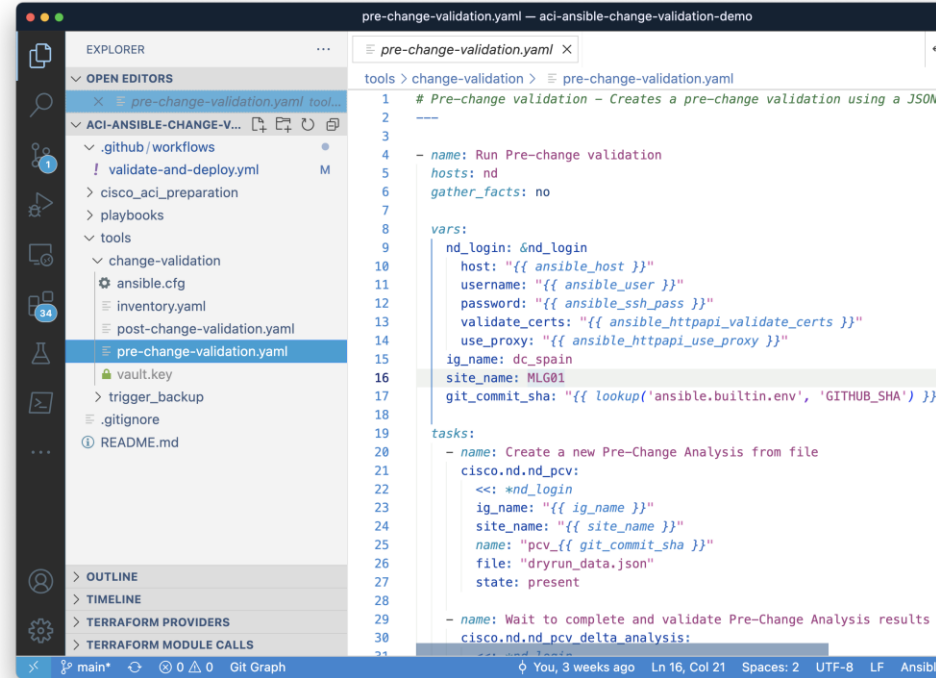
Appendix II

Cisco ND Playbooks

Cisco Nexus Dashboard Playbooks

Introducing cisco.nd collection

- Cisco Nexus Dashboard collection allows you automate Nexus Dashboard and Nexus Dashboard Insights
- More info in:
 - [Ansible Galaxy](#)
 - [GitHub](#)



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like .github/workflows, cisco_aci_preparation, playbooks, tools, change-validation, and files like ansible.cfg, inventory.yaml, post-change-validation.yaml, pre-change-validation.yaml, vault.key, trigger_backup, .gitignore, and README.md. The code editor displays the content of pre-change-validation.yaml, which is an Ansible playbook. The playbook has a name 'Run Pre-Change validation', hosts 'nd', and a task 'Create a new Pre-Change Analysis from file' that uses the cisco.nd.nd_pcv module. The task includes variables for ig_name, site_name, and git_commit_sha, and a file 'dryrun_data.json'.

```
pre-change-validation.yaml -- aci-ansible-change-validation-demo
pre-change-validation.yaml X
tools > change-validation > pre-change-validation.yaml
1 # Pre-change validation - Creates a pre-change validation using a JSON
2 ---
3
4 - name: Run Pre-Change validation
5   hosts: nd
6   gather_facts: no
7
8   vars:
9     nd_login: &nd_login
10    host: "{{ ansible_host }}"
11    username: "{{ ansible_user }}"
12    password: "{{ ansible_ssh_pass }}"
13    validate_certs: "{{ ansible_httpapi_validate_certs }}"
14    use_proxy: "{{ ansible_httpapi_use_proxy }}"
15    ig_name: dc_spain
16    site_name: MLG01
17    git_commit_sha: "{{ lookup('ansible.builtin.env', 'GITHUB_SHA') }}"
18
19   tasks:
20     - name: Create a new Pre-Change Analysis from file
21       cisco.nd.nd_pcv:
22         <<: *nd_login
23         ig_name: "{{ ig_name }}"
24         site_name: "{{ site_name }}"
25         name: "pcv_{{ git_commit_sha }}"
26         file: "dryrun_data.json"
27         state: present
28
29     - name: Wait to complete and validate Pre-Change Analysis results
30       cisco.nd.nd_pcv_delta_analysis:
31
```

Cisco Nexus Dashboard Playbooks

Pre-Change Validation Playbook

tasks:

- **name:** Create a new Pre-Change Analysis from file

cisco.nd.nd_pcv:

<<: *nd_login

ig_name: "{{ ig_name }}"

site_name: "{{ site_name }}"

name: "pcv_{{ git_commit_sha }}"

file: "dryrun_data.json"

state: present

Creates a new pre-change analysis using the file provided

- **name:** Wait to complete and validate Pre-Change Analysis results

cisco.nd.nd_pcv_delta_analysis:

<<: *nd_login

insights_group: "{{ ig_name }}"

site_name: "{{ site_name }}"

name: "pcv_{{ git_commit_sha }}"

state: validate

exclude_ack_anomalies: yes

epoch_choice: epoch2

register: pcv_result

Wait until the pre-change analysis finishes, pulls the number of new anomalies and validate no new anomalies has been raised

Acknowledged anomalies (using alert rules) can be ignored

Cisco Nexus Dashboard Playbooks

Post-Change Validation Playbook

tasks:

- **name:** Query Pre-Change Analysis performed before

cisco.nd.nd_pcv:

<<: *nd_login

ig_name: "{{ ig_name }}"

site_name: "{{ site_name }}"

name: "pcv_{{ git_commit_sha }}"

state: query

register: pre_change_validation_info

- **name:** Trigger instant assurance analysis job

cisco.nd.nd_instant_assurance_analysis:

<<: *nd_login

insights_group: "{{ ig_name }}"

site_name: "{{ site_name }}"

state: present

register: instant_analysis_triggered

[...]

Queries past pre-change validation to get base epoch

Triggers an assurance analysis (a.k.a. epoch collection)

Cisco Nexus Dashboard Playbooks

Post-Change Validation Playbook

```
tasks:
[...]
```

- **name:** Wait until instant assurance analysis is completed

```
  cisco.nd.nd_instant_assurance_analysis:
    <<: *nd_login
    insights_group: "{{ ig_name }}"
    site_name: "{{ site_name }}"
    job_id: "{{ instant_analysis_triggered.current.jobId }}"
    state: query
    register: instant_analysis_info
    until: instant_analysis_info.current.operSt == "COMPLETE"
    retries: 200
    delay: 6

[...]
```

Waits until the instant assurance analysis has completed

Cisco Nexus Dashboard Playbooks

Post-Change Validation Playbook

tasks:

[...]

- *name*: Trigger delta analysis

cisco.nd.nd_delta_analysis:

<<: *nd_login

insights_group: "{{ ig_name }}"

site_name: "{{ site_name }}"

name: "delta_{{ git_commit_sha }}"

earlier_epoch_id: "{{ pre_change_validation_info.current.baseEpochId }}"

later_epoch_id: "{{ instant_analysis_info.current.epochInfo.epochId }}"

state: present

register: delta_analysis_info

- *name*: Validate delta analysis

cisco.nd.nd_delta_analysis:

<<: *nd_login

insights_group: "{{ ig_name }}"

site_name: "{{ site_name }}"

name: "delta_{{ git_commit_sha }}"

state: validate

register: delta_analysis_results

Creates a delta analysis comparing the base epoch used in PCV (before) and the epoch just created from instant analysis (after)

Wait until completed, pulls the number of new anomalies and validate no new anomalies has been raised

The background is a vibrant, abstract graphic. It features a central bright white light source from which numerous colorful rays emanate, creating a sunburst or starburst effect. The rays transition through a spectrum of colors including yellow, orange, red, and various shades of blue and green. Overlaid on this are several large, semi-transparent, wavy shapes in similar color tones, giving the overall image a sense of motion and energy.

cisco *Live!*

Let's go

#CiscoLive