Simplifying Deployment Using Automation Tools

Sean Douglas
DATA CENTER ENGINEER

@ocdlearning

Overview

Examine how automation tools are transforming the way we manage IT infrastructure

- Ansible
- On-box Python
- Terraform
- Nexus Dashboard Fabric Controller

Ansible Automation

Imagine Nexus devices as a team of chefs in a large, bustling restaurant

Ansible as head chef who has main recipe book

When recipes are updated, all chefs automatically know what to do

- Playbook in Ansible
- Instructions (or 'recipes') for the network devices
- Automatically receive instructions and adjust configurations or tasks

Ansible Features

Managing and configuring network infrastructure:

- Does not require an agent
- Uses push-based model
- Ansible uses playbooks
- Can be used for configuration management, deployment, and orchestration of UCS servers, storage, fabric, Nexus switches

Ansible Advantages

Automation:

- Automate repetitive tasks and ensure

Consistency:

- Maintain consistent configurations

Scalability:

- Easily manage many devices

Ansible Components

Playbooks: automation blueprints in YAML format. Detailed tasks for configuring, deploying, and managing network devices and services

Tasks: core actions to be performed on Cisco devices, each task represents a single operation

Modules: execute specific tasks on Cisco device. Cisco-related modules are designed to interact directly with the operating system

Inventory: lists your Cisco devices, defining which devices to target for automation tasks. Can be static or dynamically generated

Ansible Components

Variables: allow customization of playbooks to suit different Cisco devices or environments without changing the playbook code

Templates: dynamically generate configurations for Cisco devices based on specific variables or conditions

Roles: Package tasks, templates, variables, and more into reusable units for network automation

Facts: gathered information about devices being managed, to inform and customize the automation tasks

```
- name: feature testing
    hosts: all
    connection: local
    gather_facts: no

    tasks:
# Ensure ospf is enabled
    - nxos_feature: feature=ospf
   state=enabled host={{ inventory_hostname}}
```

Ansible Playbook

The nxos-ansible Ansible library is used to convert the modules to CLI

CLIs are sent to switch via NX-API using pycsco Python module

No need for Python on switch—simply enable NX-API feature

Ansible with ACI and UCS

Ansible and Cisco ACI Integration

```
- name: Add a aaa role
  cisco.aci.aci_aaa_role:
    host: apic
    username: admin
    password: SomeSecretPassword
    name: anstest
    privileges: aaa
    state: present
  delegate_to: localhost
```

Ansible provides numerous modules tailored for managing APIC policies:

- Modules interact with the APIC REST API
- They cover most common workflows and objects
- Efficient to manage ACI environments

Cisco aci_rest Module

```yaml
---
- name: Create a new Tenant using the aci_rest module
  hosts: localhost
  gather_facts: no
  tasks:
    - name: Ensure the Tenant 'MyTenant' exists
      aci_rest:
        host: apic.example.com
        username: admin
        password: mypassword
        validate_certs: no
        method: post
        path: /api/mo/uni.json
        content:
          fvTenant:
            attributes:
              name: MyTenant
              descr: "Tenant created by Ansible"
```

If you need to do something different:

- Perform any task not covered by existing modules
- Create and send any API REST calls to the APIC

Must define the uri and HTTP method:

- APIC supports POST, GET, DELETE
- Optionally provide a payload

https://docs.ansible.com

```
-   aci_rest:
    method: post
    path: /api/mo/uni.xml
    content: |
      <fvTenant name="Support" descr="The Support Organization."/>
```

Cisco aci_rest Module

XML Format

```
- aci_rest:
    method: post
    path: /api/mo/uni.xml
    content: |
      <fvTenant name="Support" descr="The Support Organization."/>
```

Cisco aci_rest Module

XML Format

```
-   aci_rest:
    method: post
    path: /api/mo/uni.xml
    content: |
      <fvTenant name="Support" descr="The Support Organization."/>
```

Cisco aci_rest Module

XML Format

```
-   aci_rest:
    method: post
    path: /api/mo/uni.xml
    content: |
      <fvTenant name="Support" descr="The Support Organization."/>
```

Cisco aci_rest Module

XML Format

```
- aci_rest:
    method: post
    path: /api/mo/uni.xml
    content: |
      <fvTenant name="Support" descr="The Support Organization."/>
```

Cisco aci_rest Module

XML Format

```
-    aci_rest:
     method: post
     path: /api/mo/uni.json
     content: |
          {
            “fvTenant”: {
              “attributes”: {
                “name”: “Support”,“descr”: “The Support Organization.”
              }
            }
          }
```

Cisco aci_rest Module

JSON Format

Cisco ACI Ansible Example

```
git clone https://github.com/CiscoDevNet/aci_ansible_learning_labs_code_samples
cd aci_ansible_learning_labs_code_samples/intro_module
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

On-box Python with Cisco NX-OS

Python Scripting Capability

Perform tasks such as:

- Run a script to verify configuration on bootup
- Back up a configuration
- Avoid congestion, monitor data flow
- Integrate with POAP and EEM
- Schedule tasks
- Use CLI to perform various tasks

Python on Cisco NX-OS

No configuration needed to use Python on Nexus devices; it's enabled by default

- Interactive (using CLI commands)
- Non-interactive (running scripts) modes
- Can execute CLI commands through APIs available in the Python CLI module

```
Nexus# python3
<output omitted>
>>>
>>> import cisco
>>> dir(cisco)
['BGPSession', 'CiscoSecret', 'Feature', 'IPv4ACL', 'IPv6ACL',
'Interface', 'Key', 'LineParser', 'OSPFSession', ‘VRF]
```

Interactive Mode

Enter commands and get immediate feedback

To exit out of interactive mode, use exit command or Control-D

```
Nexus# dir bootflash:///scripts | include get_status.py
       1023    Jan 24 21:18:56 2024  get_status.py

Nexus# python3 get_status.py

['/bootflash/Hello_world.py'] hello world!
>>>
```

Noninteractive Mode

Can execute the Python script from the CLI by using the python3 <filename>

Check that the script is present on the switch

```
import cisco

switch = cisco.NexusSwitch("10.10.10.1")

hostname = switch.get_hostname()
```

Python API on NX-OS

To use the Python API on Nexus switches, you can import either the cisco or cli module

```
from cisco import cli

hostname = cli("show hostname")
```

Import CLI Module

To use the Python API on Nexus switches, you can import either the cisco or cli module

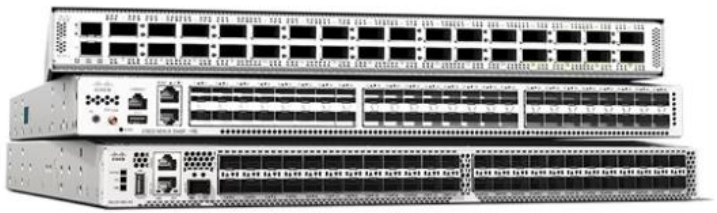
```
Nexus# python3
>>>

Nexus# source example_script.py
```

Importing the Cisco Module

Automatically imports functions from Cisco module when python command used to invoke Python interactive mode or

Use source command to invoke the Python non-interactive mode

Using CLI Command Python APIs



These APIs are like translators, allowing Python to understand and execute CLI commands:

cli()

clid()

clip()

cli() API

```
import cisco

switch = nxos.Device(host='192.168.1.1', username='admin', password='class')

# Get the output of the "show version" command
output = switch.cli("show version")

# Print the output
print(output)
```


clid() API

```
import nxos_cli

# Connect to the switch
switch = nxos_cli.Device("192.168.1.1", "admin", "password")

try:
    # Get the output of the "show version" command
    output = switch.clip("show interface")

    # Print the output to the console
    print(output)

finally:
    # Close the connection to the switch
    switch.close()
```


clid () Output


```
[
  {
    "interface": "Ethernet1/1",
    "ip_address": "192.168.1.1",
    "status": "up",
    "speed": "1000 Mbps",
    "duplex": "full"
  },
  {
    "interface": "Ethernet1/2",
    "ip_address": "192.168.1.2",
    "status": "up",
    "speed": "1000 Mbps",
    "duplex": "full"
}
]
```

```
from cli import *

# Get the output of the "show interface" command
output = clip("show interface")

# Print the output
print(output)
```

clip()

The clip() API executes CLI commands and prints the output directly to the standard output. It does not return any value to the Python script.

Results are more readable

clip() API Output

Interface	Admin St	Oper St	PortCh	Encap Type	Description
Ethernet1/1	up	up	-	trunk	Eth1
Ethernet1/2	up	up	-	trunk	Eth2
Ethernet1/3	up	up	-	trunk	Eth3
Ethernet1/4	up	up	-	trunk	Eth4
Ethernet1/5	up	up	-	trunk	Eth5
Ethernet1/6	up	up	-	trunk	Eth6
Ethernet1/7	up	up	-	trunk	Eth7

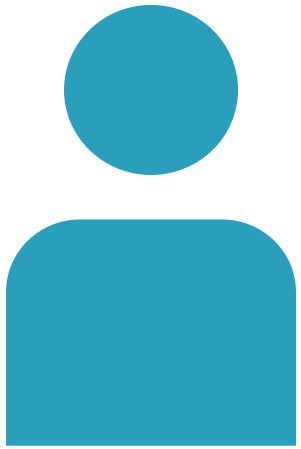
```
Nexus# source get_status.py

------------------- [Printing report] --------------
        Current time:         2024-01-22 13:35:43.187805
        Switch uptime:        6 day(s), 11 hour(s), 45 minute(s), 5 second(s)
        Switch name:          Nexus
        Kickstart version:    8.4(2)
        Software version:     8.4(2)
----------------------------------------------------
```

Run Script from NX-OS CLI

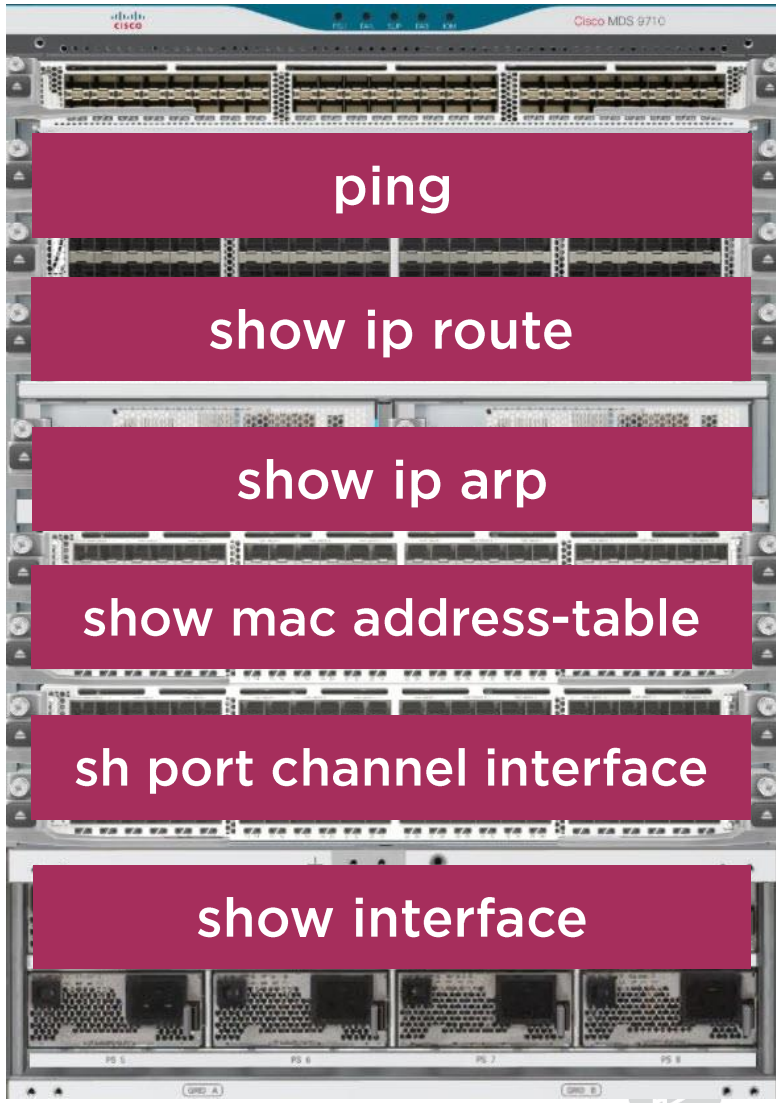
Use the source command to execute the script from the CLI

The source command searches for the scripts in the bootflash:///scripts folder

Python Scripting and Aliasing



Support Engineer




```
Nexus(config)# cli alias name resolve source script.py
Nexus(config)# exit
Nexus(config)# resolve
```

```
event manager applet call_python_script
event syslog pattern "INTERFACE_DOWN"  # This is just an example trigger
action 1.0 cli command "enable"
action 1.1 cli command "source bootflash:/my_script.py"
```

Call Python Script Inside of EEM

To call a Python script from inside Cisco's Embedded Event Manager, use the EEM action command

Cisco UCS Python SDK

```
git clone https://github.com/CiscoUcs/ucsmsdk.git
cd ucsmsdk
make install

pip install ucsmsdk
```

Cisco UCS can use Python SDK to allow automation of the tasks in Cisco UCS

- View and modify configurations
- Automate routine tasks
- Monitoring and reporting
- Handle alerts and events

HashiCorp Terraform

Terraform is like a blueprint for your infrastructure

Define and control resources using simple text files

Describe the deployment, and Terraform automatically builds it

IaC – create and manage VMs - whether in cloud or on-premises

Step 1

Download Terraform: Go to the Terraform website and download the Windows or Linux

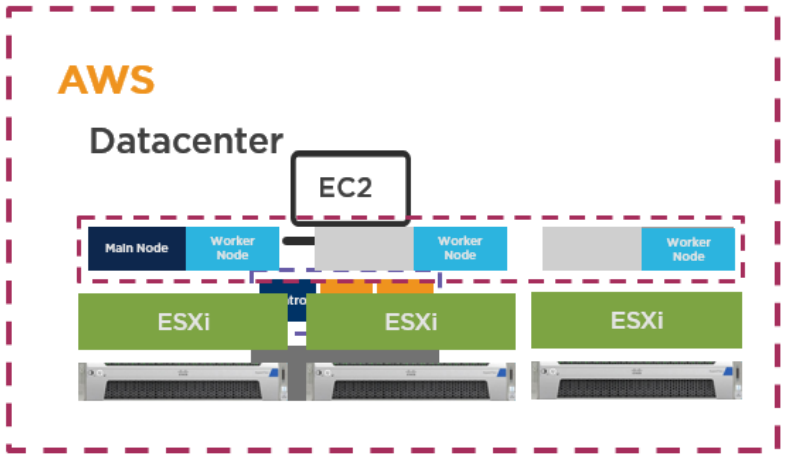
Step 2

Unzip the File: Once you've downloaded it, you'll have a zip file

Step 3

Add the Path: to the folder where your Terraform executable is located. Now you can just type "terraform" into your command line, and it will work

Terraform



HashiCorp Terraform Language (HCL)

- Code is stored in plain text files with the .tf file extension
- Files containing Terraform code called configuration files
- Define desired state of infrastructure, including resources, variables, and providers

```
init        Prepare your working directory for other commands
plan        What need to be changed by to match what's specified
apply       Make the changes to the infrastructure per the plan
destroy     Destroy previously-created infrastructure
```

Main Terraform Commands

Whether you want to create, update, or delete resources, Terraform commands are designed to simplify the process

Terraform Workflow

Terraform init

Scan the code and providers; download code for them

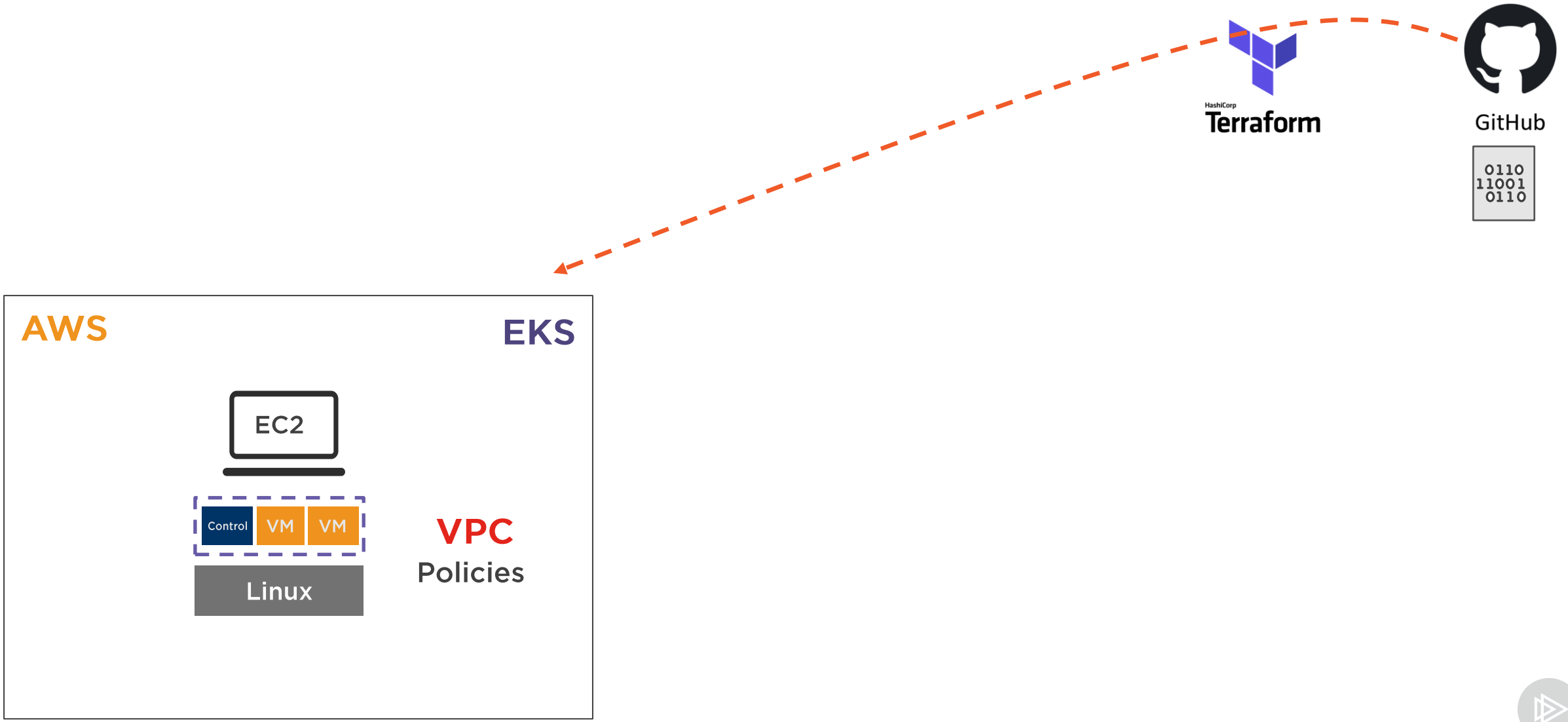
Terraform plan

Examine what Terraform will do before making changes

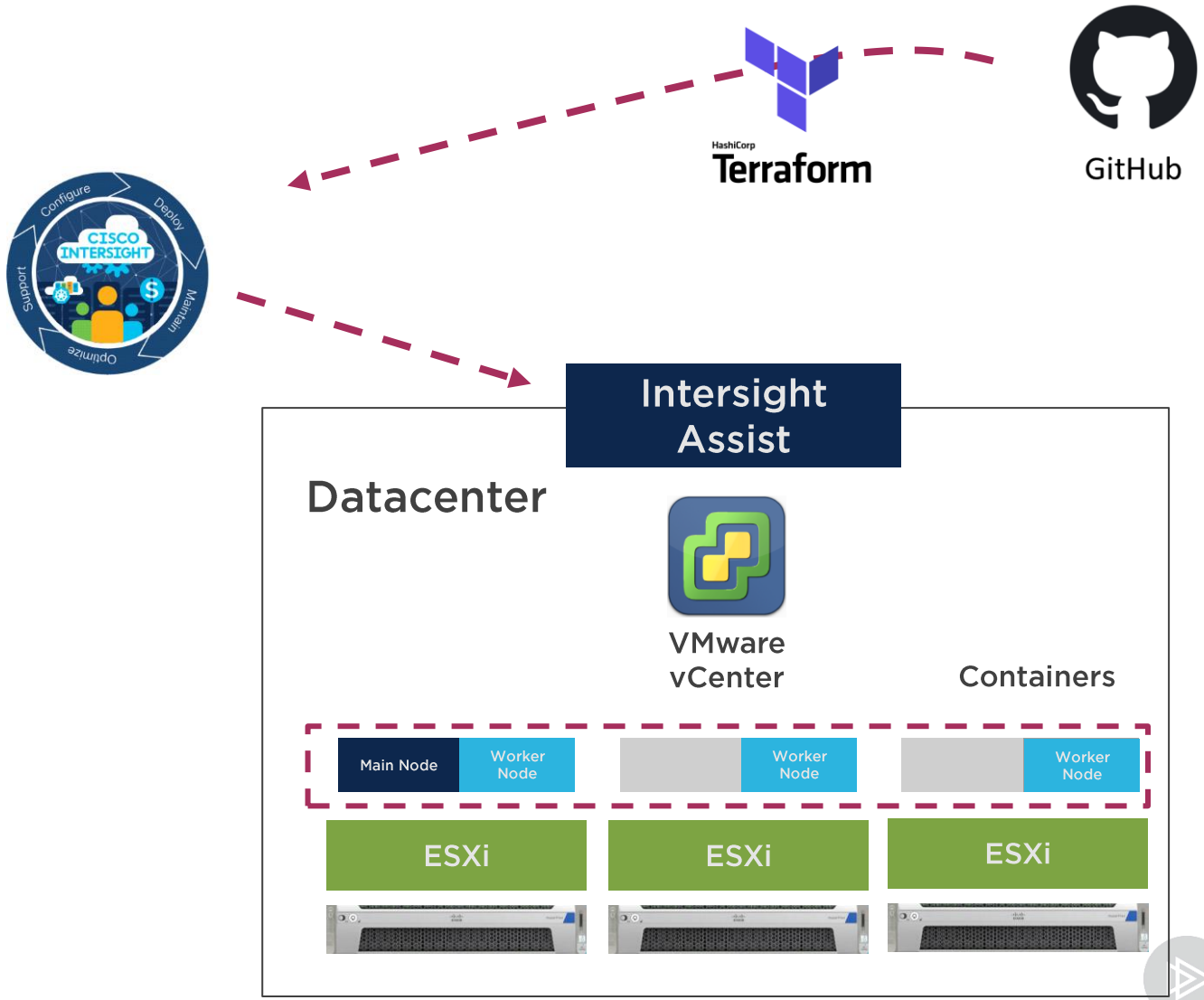
Terraform apply

Executes the actions proposed in Terraform plan

Terraform in Action



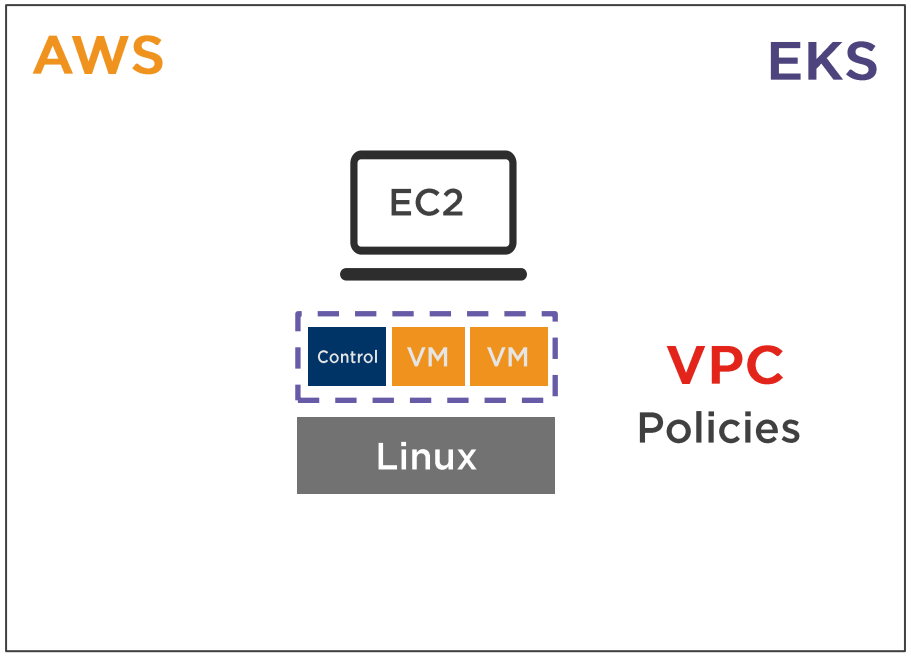
Terraform in Action

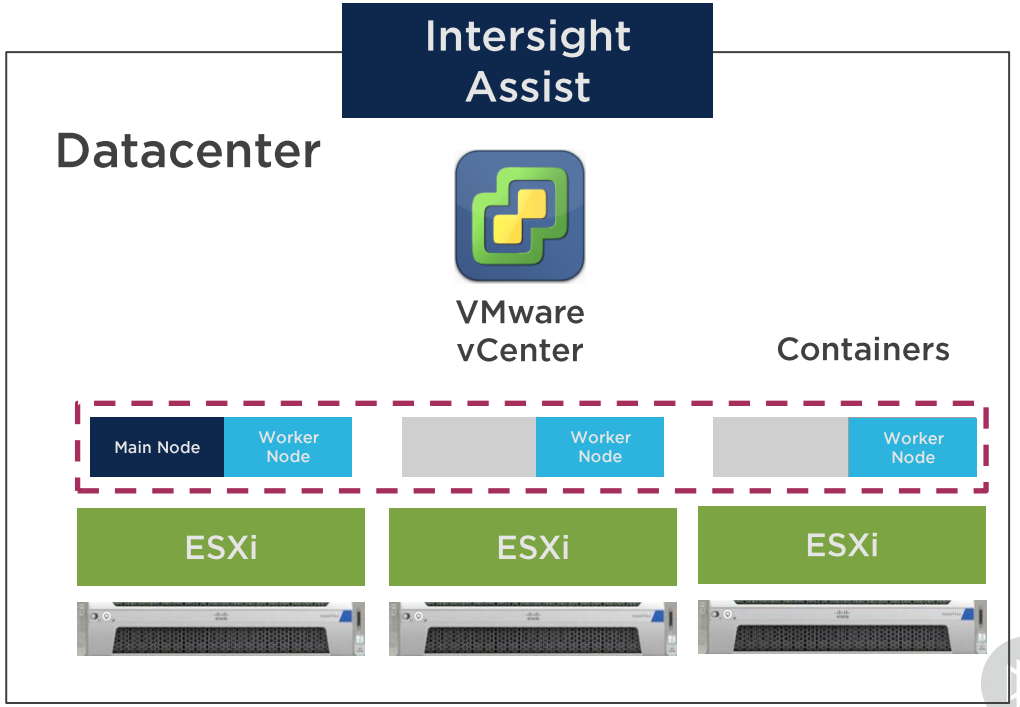


Terraform in Action

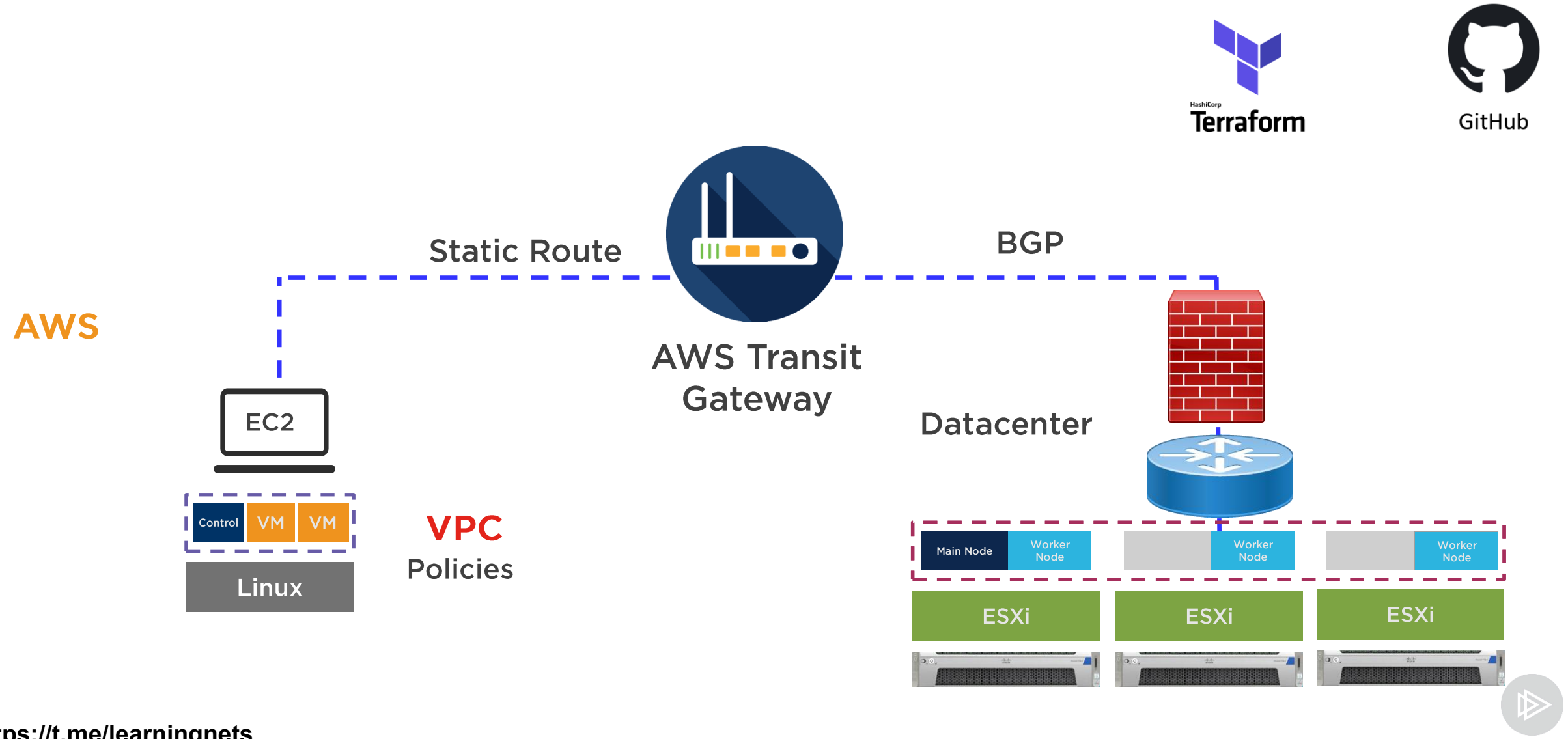
HashiCorp Terraform

GitHub





TerraForm in Action

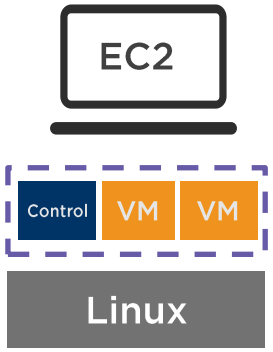


TerraForm in Action

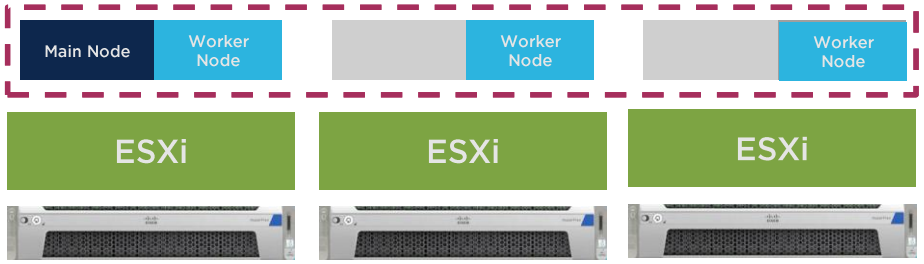
HashiCorp Terraform

GitHub

AWS



Datacenter



Terraform in Action

Applications

HashiCorp Terraform

GitHub

Hybrid Compute Environment

APPDYNAMICS

ThousandEyes

AWS

Datacenter

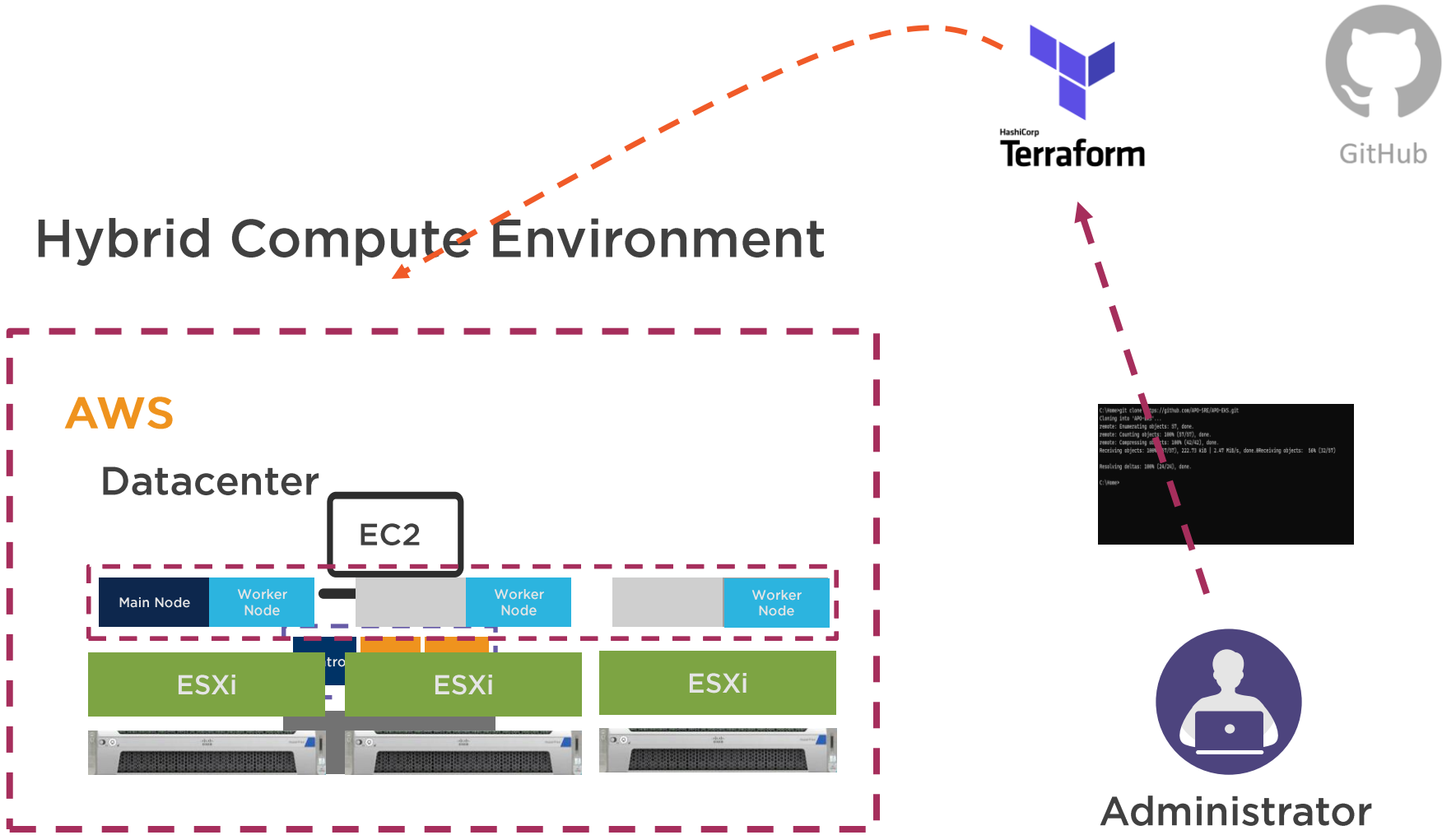
EC2

Main Node | Worker Node | Worker Node | Worker Node

ESXi | ESXi | ESXi

Administrator

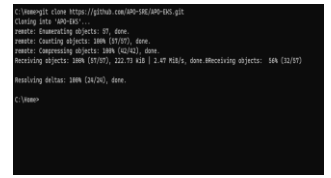
Terraform in Action



Terraform in Action

HashiCorp Terraform

GitHub



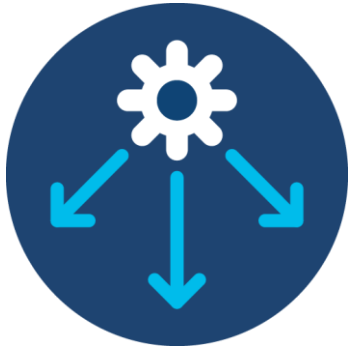
Administrator

Cisco Nexus Dashboard Fabric Controller

Nexus Dashboard Services

Fabric Controller

Deploy VXLAN fabrics on Cisco NX-OS



Fabric Discovery

Monitor NX-OS fabrics, provides inventory and automation tools

SAN Controller

Manages and observes SAN fabrics

Cisco Nexus Dashboard Fabric Controller

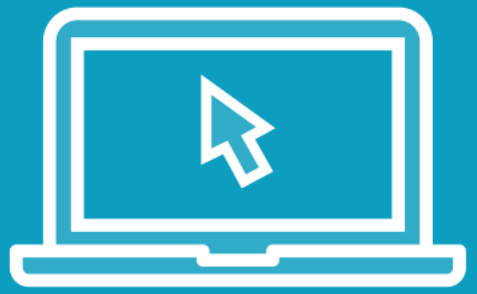
Designed for managing NX-OS and VXLAN-EVPN fabrics

- Works with all Nexus and MDS switches
- Automation, visibility, analytics

Do not require the ACI provided by APIC

Integrates with products like UCS Director, vSphere, and OpenStack

Demo



Complex and time-consuming process of configuring and managing a large-scale fabric

Utilize NDFC

- Reduce complexity, avoid errors
- Show you how to bring a VXLAN EVPN fabric to life using templates

NDFC Demo Recap

Initial Challenge

Managing and configuring a large-scale fabric network

NDFC

Streamline process, deploying and managing configurations

NDFC Recap

Initial Challenge

Managing and configuring a large-scale fabric network

Cisco NDFC

Streamlined process, deploying and managing configurations

Talking point three

Be concise and keep the text to four lines or fewer

Summary

Examined how automation tools are transforming the way we manage IT infrastructure

- Ansible
- Python
- Terraform
- Nexus Dashboard Fabric Controller