

CISCO *Live!*

GO BEYOND

#CiscoLiveAPJC



5 Tools You Should Learn to Level Up Your Network Automation Game

Kareem Iskander | Lead Technical Advocate | @kareem_isk
Quinn Snyder | Senior Technical Advocate | @qsnyder
DEVNET-2149

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-2149>



Agenda

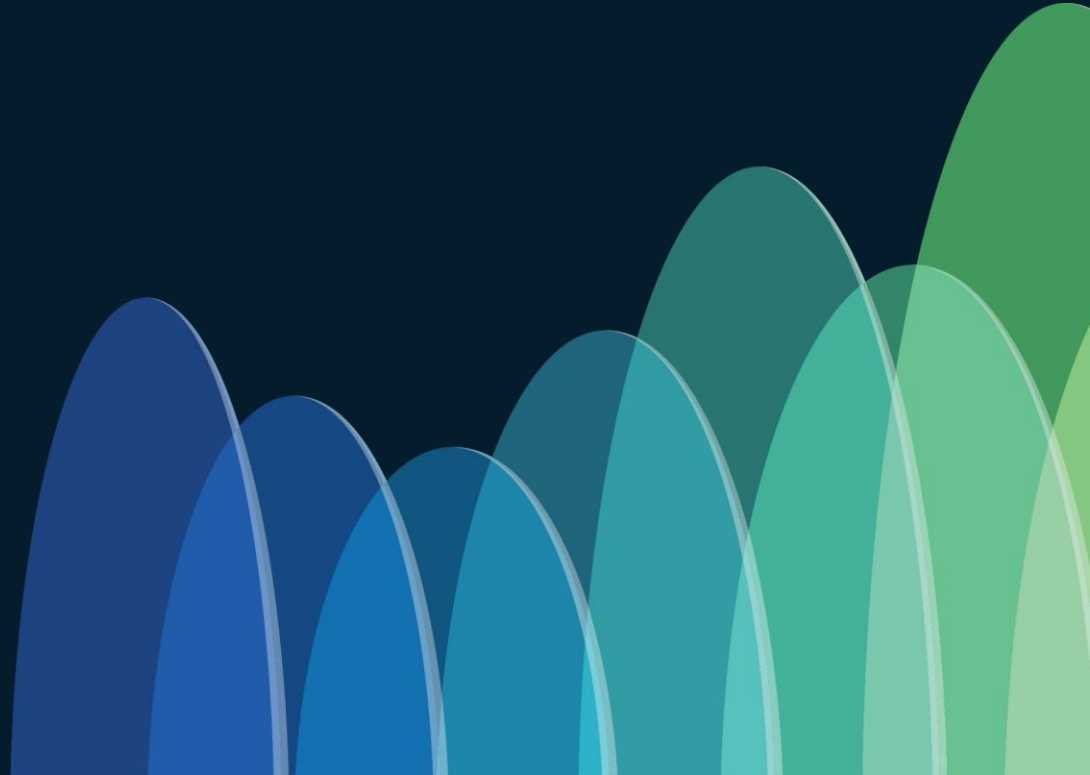
- Introduction
- Git'in it right
- Its OK to be `bash`ful
- Locking the Vault
- Codifying your infra
- Containerize this!
- Conclusion

Introduction

So.. I learned the
building blocks of
network
automation, but
what happens in
practice in the real
world?

Let's Level Up!

Git'in it right



What we've
been talking to
you about

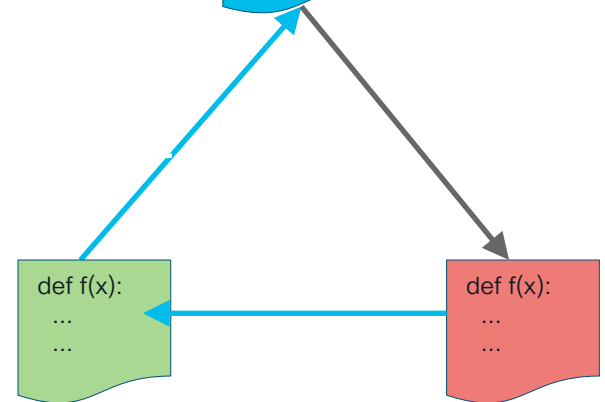


GIT - Basics

What we've been talking to you about



Step	Action	Git Command
1.	Clone the Remote Repository	<code>git clone url</code>
2.	Create and Checkout a Local Branch	<code>git checkout -b new-branch-name</code>
3.	Incrementally Commit Changes	<code>git add filename</code> <code>git commit -m "Commit message"</code>



GIT - Basics

What we've been talking to you about

Pull
Requests

Merge
Conflict

Commits

Branches

Logs

HEAD

Push

Real World



Interactive Rebase



Interactive Rebase

A Tool for Optimizing & Cleaning up Commit History

- Change a commit's message
- Delete a commit
- Reorder commits
- Combine multiple commits into one
- Edit/Split an existing commit into multiple new ones

 **DO NOT** use Interactive Rebase on commits that you've already pushed/shared on remote repo!

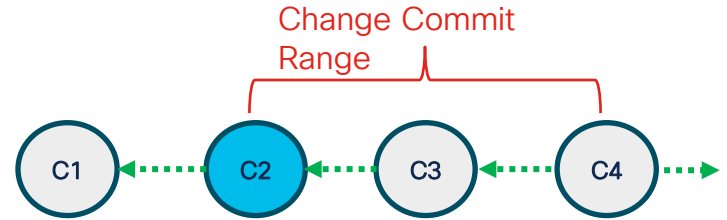
Interactive Rebase

Use Case

- You are working on the **Feature** branch to expand Meraki automation
- You have been making commits ever function you write
- You are ready to merge into **Main** branch
- You have realized:
 1. “Over Commit-ted” – get it?
 2. Commit messages aren’t cutting it

Interactive Rebase - Steps

1. How far back do you want to go?



2. `git rebase -i HEAD~2`

3. Determine action to apply to your commits

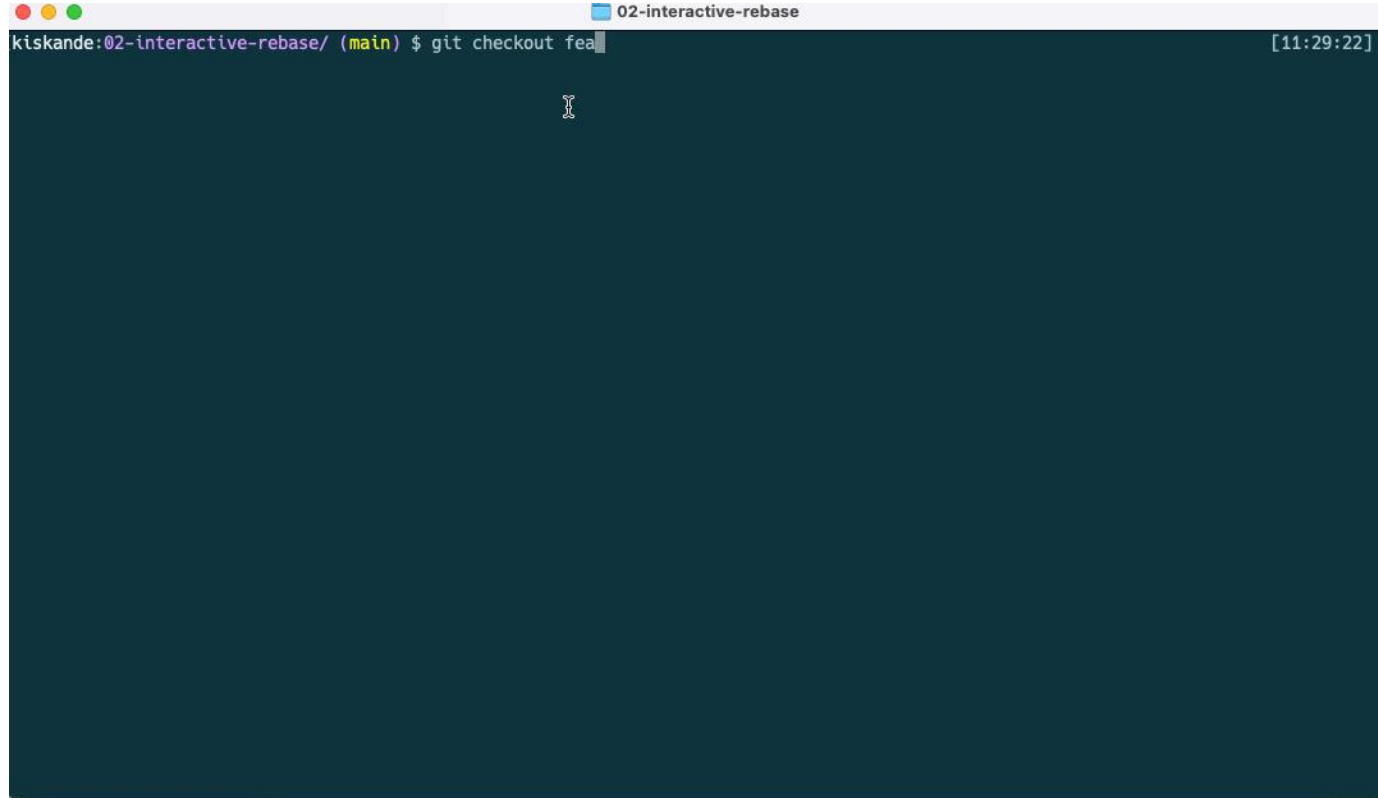
4. Make changes & save

5. `git log --oneline`

Let's try it!

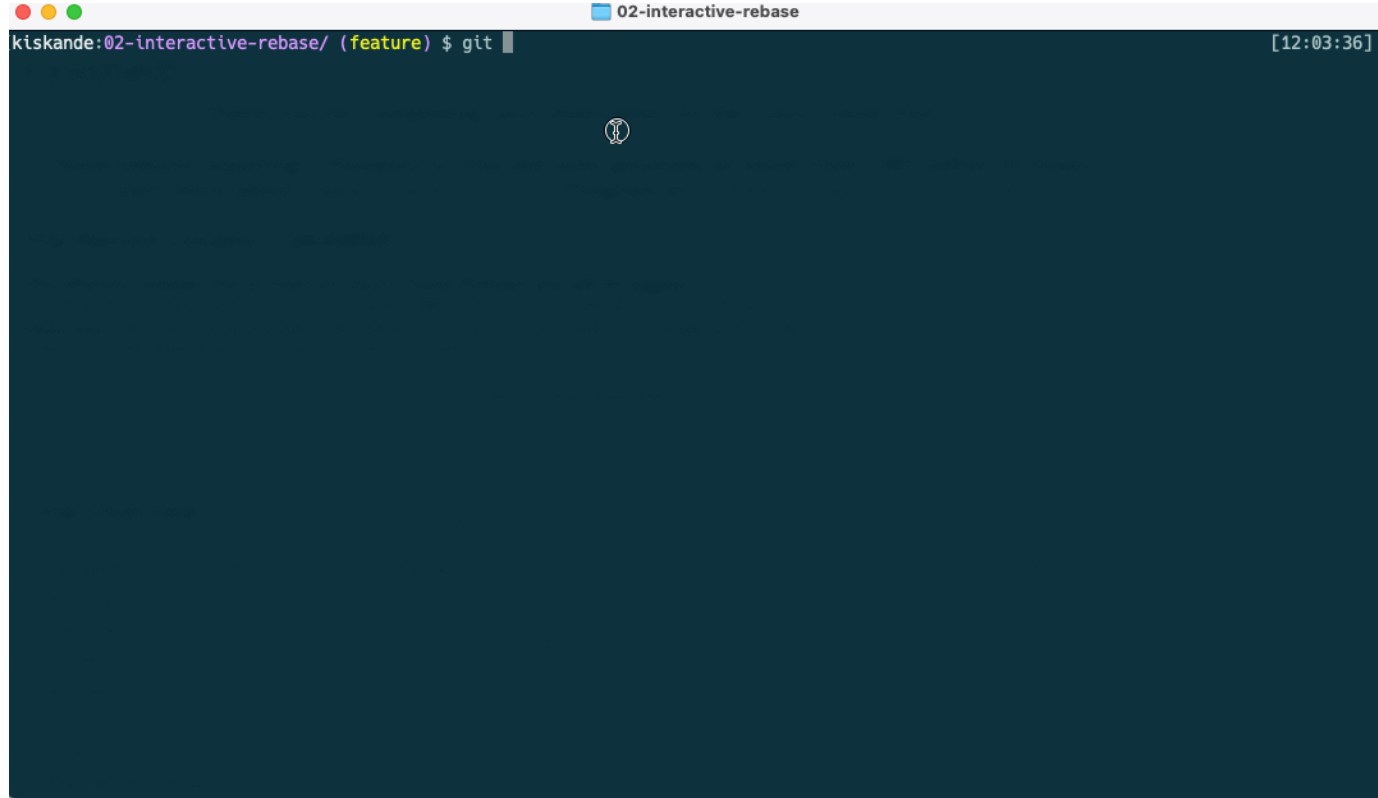


Interactive Rebase – Reword

A terminal window titled "02-interactive-rebase" with a dark blue background. The prompt is "kiskande:02-interactive-rebase/ (main) \$". The command "git checkout fea" is entered, with the cursor at the end. The time "[11:29:22]" is in the top right corner.

```
kiskande:02-interactive-rebase/ (main) $ git checkout fea [11:29:22]
```

Interactive Rebase - Squash



Cherry Picking 🍒

Cherry Picking

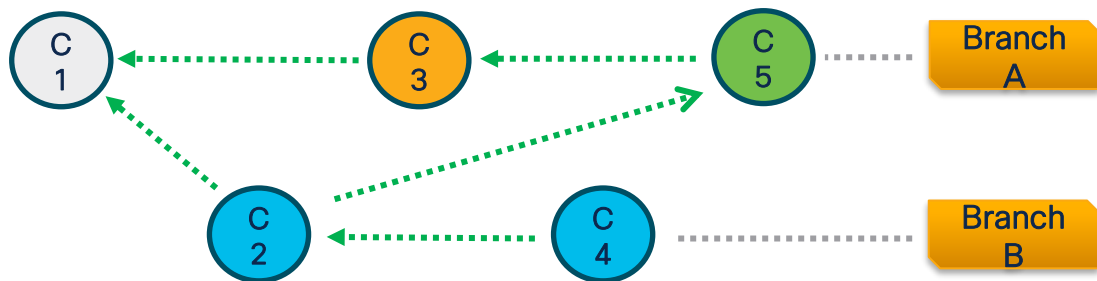
Use Case

- You are working on the **Feature** branch to expand your Meraki site automation
- You have made a commit on the **Feature** Branch
- You noticed you are working in the **Main** Branch
- The commit does not belong in **Main** .. Yet
- What to do??

Cherry Picking 🍒

Integrate Single, Specific commit

- Cherry Picking allows you to select individual commits
- Cherry Picking integrate specific commits to Branch
- Only Commit C2 from Branch B to integrate into Branch A



Cherry Picking 🍒 - Steps

1. While on **Main** Grab the commit hash

```
git log --oneline
```

2. Checkout the **Feature** branch

```
git checkout Feature
```

3. Cherry Pick the commit

```
git cherry-pick a827df1
```

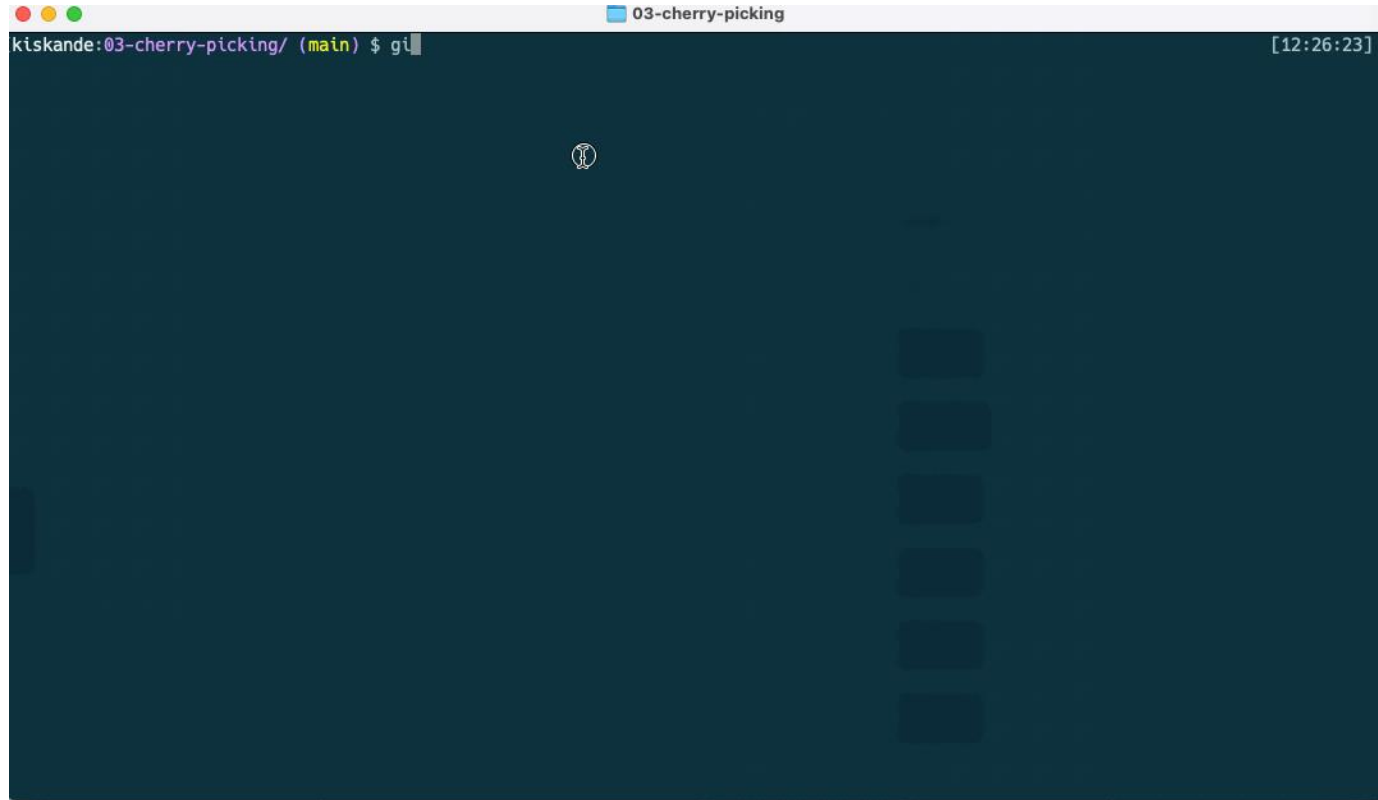
4. Clean up **Main**

```
git checkout main  
git reset --hard HEAD~1
```

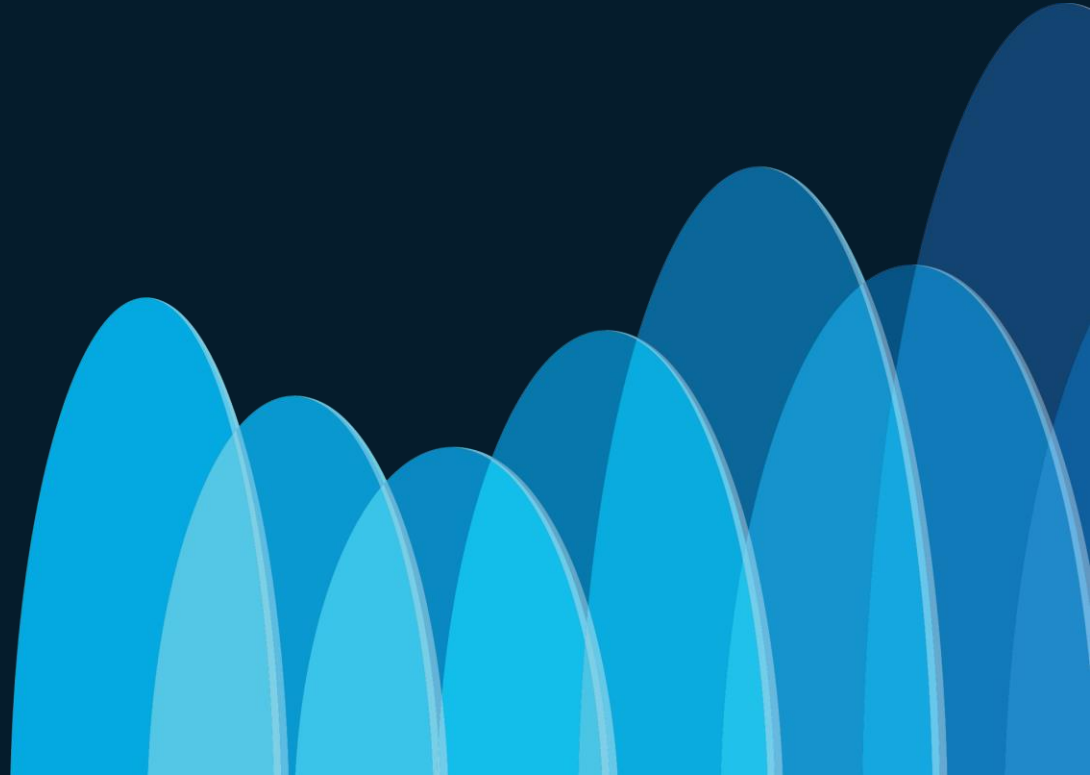
Let's try it!



Cherry Picking



Its OK to be
`bash`ful



What we've demonstrated



Dockerfiles, GNU `make`, SCM-based CI/CD

```
1 FROM python:3.8.19-alpine3.18
2 MAINTAINER Quinn Snyder <qsnyder@cisco.com>
3
4 COPY requirements.txt .
5
6 ENV info "LTRC1100: Python 3.8.19, Ansible-Core 2.13.13, Openconnect,
  VIRLUtils"
7
8 RUN echo 'PS1="\[\e[36m\]\u\[\e[m\]\[\e[37m\]@\[\e[m\]\[\e[32m\]\ltrcrt1100\
  \[\e[m\]:\[\e[33m\]\w\[\e[m\]\[\e[33m\]\$\[\e[m\] "' >> /root/.bashrc
9
10 RUN echo "http://dl-cdn.alpinelinux.org/alpine/edge/testing/" >>
  /etc/apk/repositories && \
11   apk update && \
12   apk add --no-cache make curl bash git openssh gcc linux-headers musl-dev
  libffi-dev openssl-dev libxml2-dev libxslt-dev wget ope
13   python -m ensurepip && \
14   pip install --upgrade pip setuptools && \
15   rm -r /root/.cache && \
16   pip install -r requirements.txt && \
17   rm -rf /var/cache/apk/*
18 WORKDIR "/mycode"
19 CMD ["/bin/bash"]

SHELL := /usr/bin/env bash
.PHONY: build-python-3.7 build-python-3.10 build-ansible-2.9 build-ansible-2.10 python-3.7 python-3.10 ansible-2.9
ansible-2.10

build-python-3.7:
  docker build -t qsnyder/python-3.7 ./python-3.7/

build-python-3.10:
  docker build -t qsnyder/python-3.10 ./python-3.10/

build-ansible-2.9:
  docker build -t qsnyder/ansible-2.9 ./ansible-2.9/

build-ansible-2.10:
  docker build -t qsnyder/ansible-2.10 ./ansible-2.10/

python-3.7:
  docker run -it --rm --privileged -e "TERM=xterm-256color" -v $(pwd)/python-3.7:/root/mycode
  qsnyder/python-3.7

python-3.10:
  docker run -it --rm --privileged -e "TERM=xterm-256color" -v $(pwd)/python-3.10:/root/mycode
  qsnyder/python-3.10

ansible-2.9:
  docker run -it --rm --privileged -e "TERM=xterm-256color" -v $(pwd)/ansible-2.9:/root/mycode
  qsnyder/ansible2.9

ansible-2.10:
  docker run -it --rm --privileged -e "TERM=xterm-256color" -v $(pwd)/ansible-2.10:/root/mycode
  qsnyder/ansible2.10
```

```
# Grab the current directory of the pull request.
- name: Grab the current directory
  us: pathname-gather
  run: |
    URL1="https://api.github.com/repos/${{ github.repository }}/pulls/${{ github.event.pull_request.number
  }}/files?page=100"
    URL2="https://api.github.com/repos/${{ github.repository }}/pulls/${{ github.event.pull_request.number
  }}/files?page=200&per_page=100"
    URL3="https://api.github.com/repos/${{ github.repository }}/pulls/${{ github.event.pull_request.number
  }}/files?page=300&per_page=100"
    if FILENAME=$(curl -sL -H "Authorization: Bearer ${{ secrets.GITHUB_TOKEN }}" $URL1 | jq -r '.[] |
  .filename' | grep sidecar.json) ; then
      echo "FILENAME found in the first page"
    elif FILENAME=$(curl -sL -H "Authorization: Bearer ${{ secrets.GITHUB_TOKEN }}" $URL2 | jq -r '.[] |
  .filename' | grep sidecar.json) ; then
      echo "FILENAME found in the second page"
    else FILENAME=$(curl -sL -H "Authorization: Bearer ${{ secrets.GITHUB_TOKEN }}" $URL3 | jq -r '.[] |
  .filename' | grep sidecar.json)
      echo "FILENAME found in the third page"
    fi
    echo $FILENAME
    MIDDLENAME="${dirname $FILENAME}"
    echo "PATHNAME=MIDDLENAME" >> $GITHUB_OUTPUT

# Collect the conversion and linting tools
- name: Install the tutorial conversion tool, set execution permissions
  run: |
    curl -sLJO \
    -H 'Accept: application/octet-stream' \
    -H 'Authorization: token ${{ secrets.LEARNING_TOKEN }}' \
    'https://api.github.com/repos/learningatcisco/tutorial-md-to-xml/releases/assets/${{
  v.TUTORIAL_MD_RELEASE_ASSET }}'
    mv tutorial_md2xml_v1.0.0_linux_amd64 tutorial_md2xml
    chmod +x ./tutorial_md2xml

- name: Install the cli tool, set execution permissions
  run: |
    curl -sLJO \
    -H 'Accept: application/octet-stream' \
    -H 'Authorization: token ${{ secrets.LEARNING_TOKEN }}' \
    'https://api.github.com/repos/learningatcisco/cli/releases/assets/${{ env.CLI_RELEASE_ASSET }}'
    mv sol_v1.0.0_linux_amd64 sol
    chmod +x ./sol
```

What's old is new again

Who had this book at Uni?



I use it more often than any book I've saved from my college education

`bash` | 36 years and still going strong!

- Everything entered at the Linux CLI* can become part of a shell script or CI/CD pipeline
 - Yes, its all bash scripts and YAML
 - This is “imperative language” which is still a thing in 2024
- Scripts can still execute other higher order processes
 - Python, Ansible, Terraform, etc
- Dockerfiles, CI/CD pipeline declarations (Gitlab, GitHub Actions)
 - May have specific runners or actions to hide abstraction
 - Everything is running some sort of Linux* container

*Assuming POSIX-shells and so on...

Practical Applications



Bash can build locally

A terminal window with a dark background. The title bar shows three colored dots (red, yellow, green) and the text '~ /d/s/task10'. The prompt is '[I] task10 »'. The command entered is 'docker build -t ltrcr1100:0.1 .' followed by a cursor. The output is '~ /d/s/task10 [16:12:10]'.

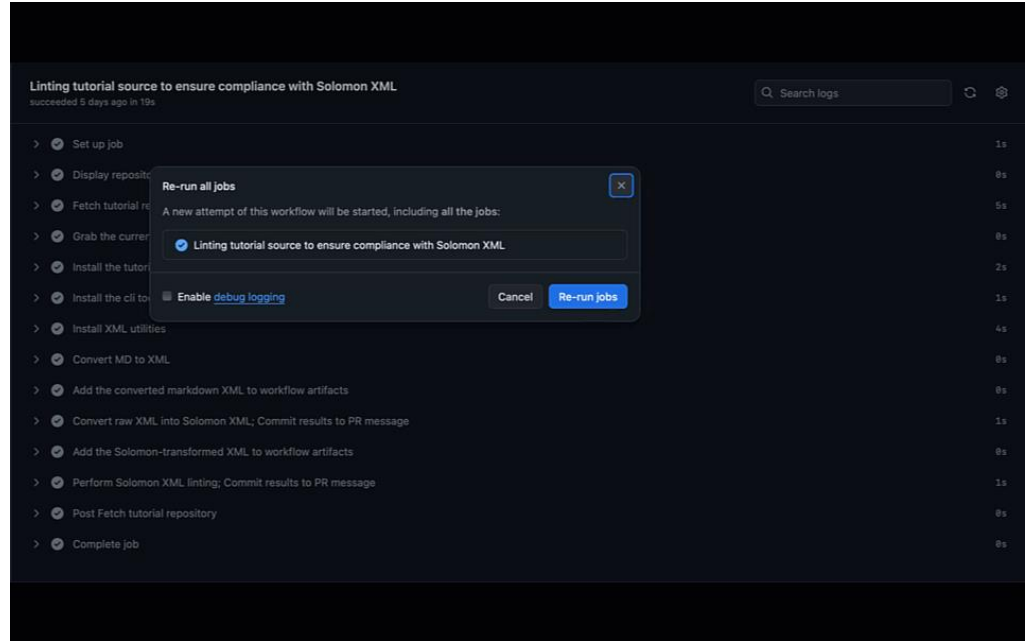
```
~/d/s/task10  
[I] task10 » docker build -t ltrcr1100:0.1 .  
~/d/s/task10 [16:12:10]
```

Whether using native Docker CLI, or `makefile` with targets, commands run within container are based on Linux CLI

Bash can build in the cloud

GitHub Actions are container runners of various OS-types.

Some pre-built functions exist, but bespoke scripts may be needed, even if calling Python/Go/etc.



“When I die, I don’t care about clearing my browser history. I want someone to delete my ~/dev folder to hide the heinous things I did with bash scripts”

~ Some Bearded Jabroni

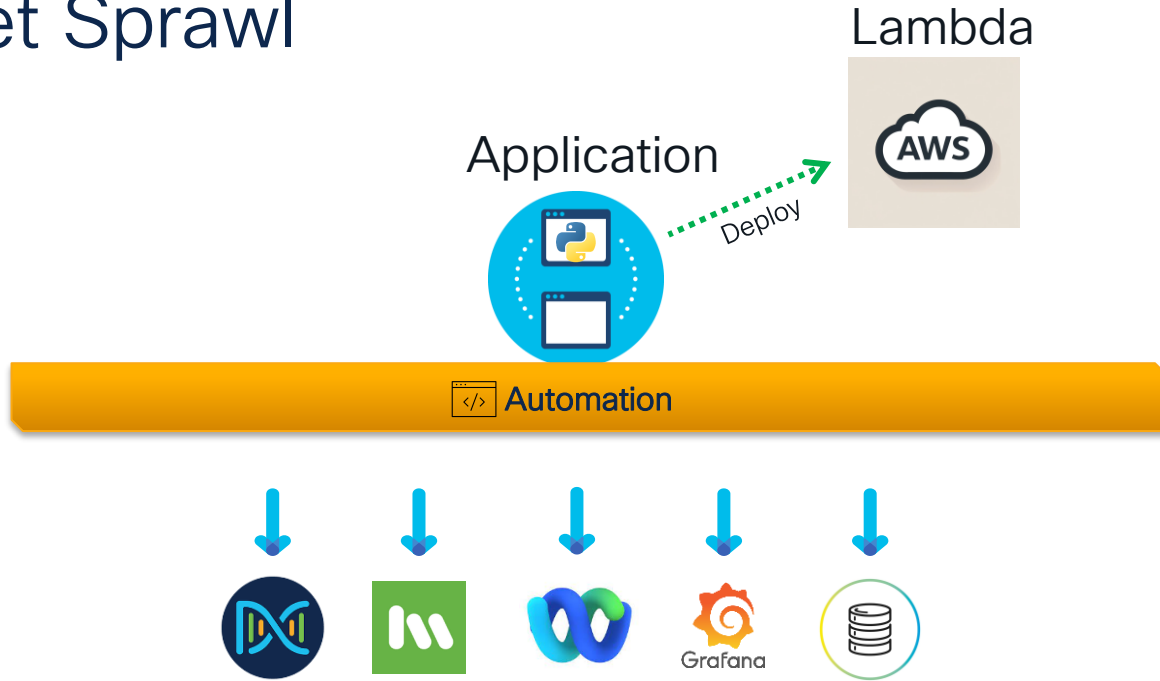
Locking the Vault



What we've
been talking to
you about



Secret Sprawl



~7 Different API Tokens | SSH Keys | Auth Methods

Don't be *that* person

HashiCorp Vault – Secrets SSOT

Single Source of Truth for all you Secretes

- API Tokens
- SSH Keys
- Basic Auth – Username/Password
- DB Creds

Granular Access Control (ACLs)

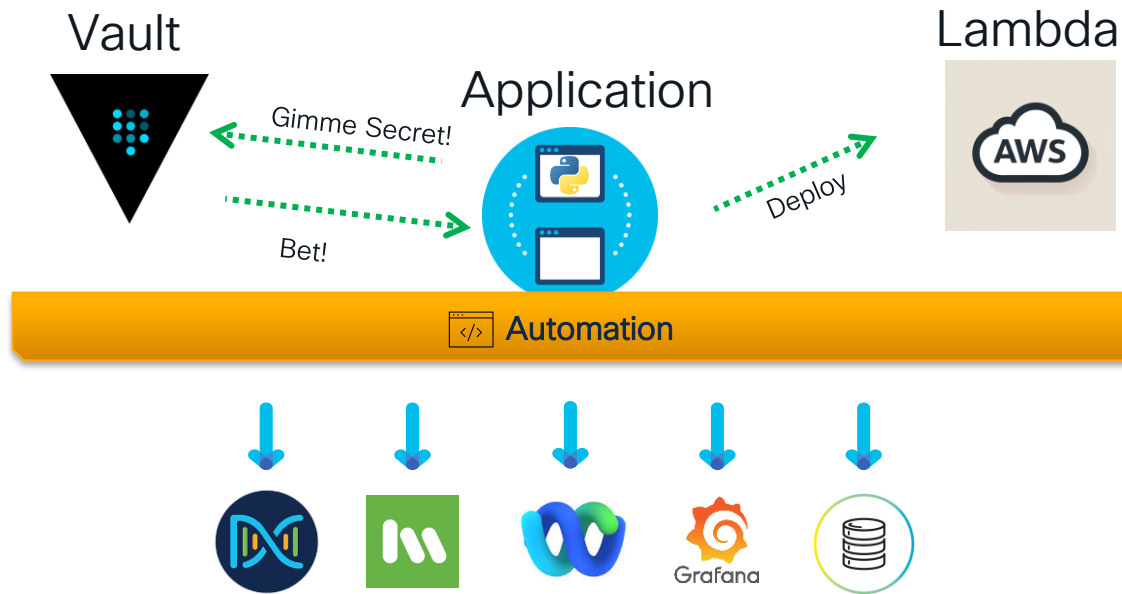
- Define App Rules
- Apps granted access to Mount/Path
- Paths Contain Creds

Vault APIs FTW

- API access to your creds
- Seamless integration with your App
- Bonus Points – HVAC SDK!!

Real World





Let's try it!



VAULT HVAC SDK + DNACSDK = <3

1. Instantiate Vault

```
def vault_auth():  
    """  
    This function will check if the vault is sealed, unseal it and authenticate against vault  
    """  
    # Check if vault is sealed  
    if client.sys.is_sealed() == True:  
        # if the vault is SEALED, UNSEAL IT using the unseal_key  
        unseal_response = client.sys.submit_unseal_key(vault_unseal_key)  
        client.token = vault_client_token
```

2. Unseal Vault

```
def vault_auth():  
    """  
    This function will check if the vault is sealed, unseal it and authenticate against vault  
    """  
    # Check if vault is sealed  
    if client.sys.is_sealed() == True:  
        # if the vault is SEALED, UNSEAL IT using the unseal_key  
        unseal_response = client.sys.submit_unseal_key(vault_unseal_key)  
        client.token = vault_client_token
```

3. Read the Secrets

```
def vault_r_secret(mount, path):  
    """  
    This function will read secret from the MOUNT you've created in VAULT and return the secret  
    """  
    read_secret_result = client.secrets.kv.v1.read_secret(path=vault_path, mount_point=vault_mount_point)  
    print(read_secret_result)  
    return read_secret_result
```

4. Start

```
def get_dnac_token(env_creds):  
    """  
    This function will Authenticate against Cisco DNA Center server and print out a list of all managed devices  
    """  
    dnac = DNACenterAPI(username=env_creds['data']['username'],  
                        password=env_creds['data']['password'],  
                        base_url=env_creds['data']['url'], verify=False)  
    print("DNAC API Authenticated ...")  
    print("Gathering Device Info ... \n")  
    devices = dnac.devices.get_device_list()  
    for device in devices.response:  
        print("Device Management IP {} for {}".format(device.managementIpAddress, device.hostname))
```

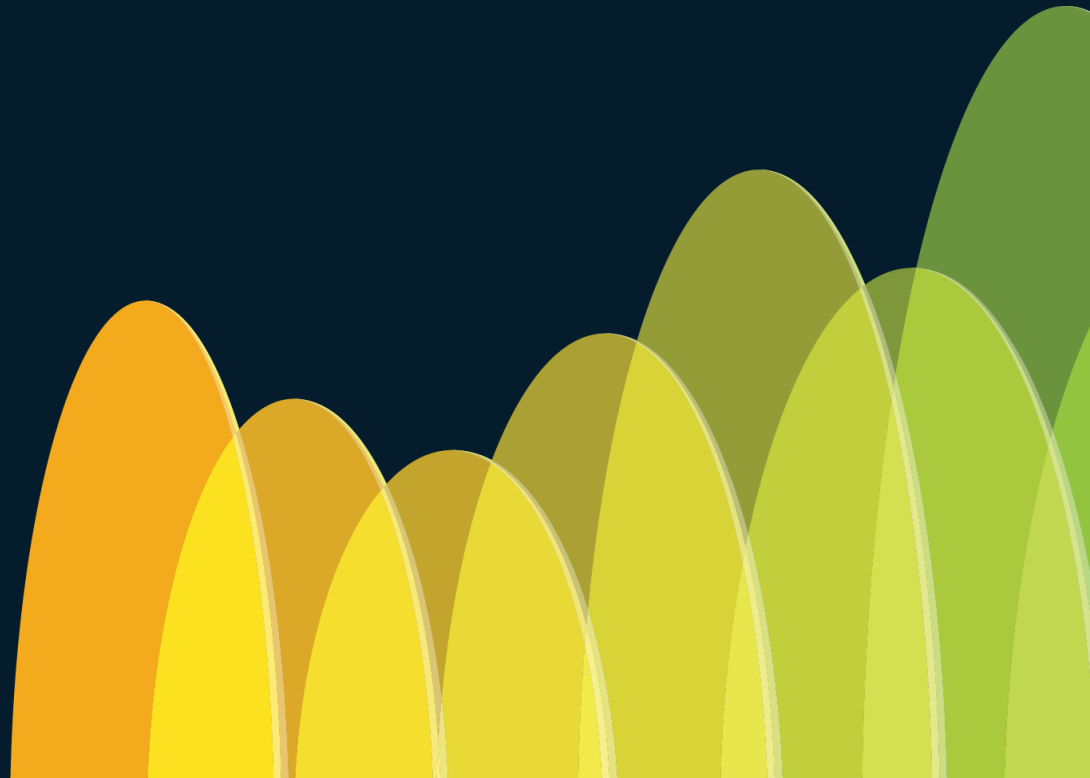
VAULT HVAC SDK + DNACSDK = <3

```
kiskande:secure-apis-vault/ (main*) $ python3
```

The screenshot shows the HashiCorp Vault web interface. The breadcrumb navigation is < kv-v1 < devnet < dnac < sb1. The secret is named 'devnet/dnac/sb1' and is of type 'Secret'. The format is set to 'JSON'. The secret contains three entries: 'password' with value 'Cisco123!', 'url' with value 'https://sandboxdnac.cisco.com/', and 'username' with value 'devnetuser'. The footer of the interface shows '© 2024 HashiCorp', 'Vault 1.13.2', and links to 'Upgrade to Vault Enterprise' and 'Documentation'.

Key	Value	Version created
password	Cisco123!	
url	https://sandboxdnac.cisco.com/	
username	devnetuser	

Codifying your Infra



What I've
droned on about



CISCO *Live!*



Free bridge to possible

Prepare for the Enterprise Automation (ENAUTO) Certification with Real-Life Applications

George Koukis, Leader Exam Progr.
Quinn Snyder, Senior Technical Adv.
BRKRCIT-2014

```
- name: Testing our fact-finding mission
hosts: all
gather_facts: no

tasks:
```

```
- name: Testing our fact-finding mission
  hosts: all
  gather_facts: no

  tasks:

    - name: Gather IOSXE Facts
      cisco.ios.ios_facts:
        gather_subset:
          - config
      when: ansible_network_os == 'cisco.ios.ios'

    - name: Gather NXOS Facts
      cisco.nxos.nxos_facts:
        gather_subset:
          - config
      when: ansible_network_os == 'cisco.nxos.nxos'
```

Typical Terraform Workflow

```
graph LR; A[Binary Unzip/install  
Create the configuration for the endpoint resource along with any variables] --> B[Terraform Init  
Reads the configuration, downloads and initializes the provider for this project]; B --> C[Terraform Plan  
Determines the delta and steps to apply desired configuration to the resource]; C --> D[Terraform Apply  
Applies the steps created from running the plan command]; D --> E[Terraform Destroy  
Removes the configured state]
```

HCL Basics

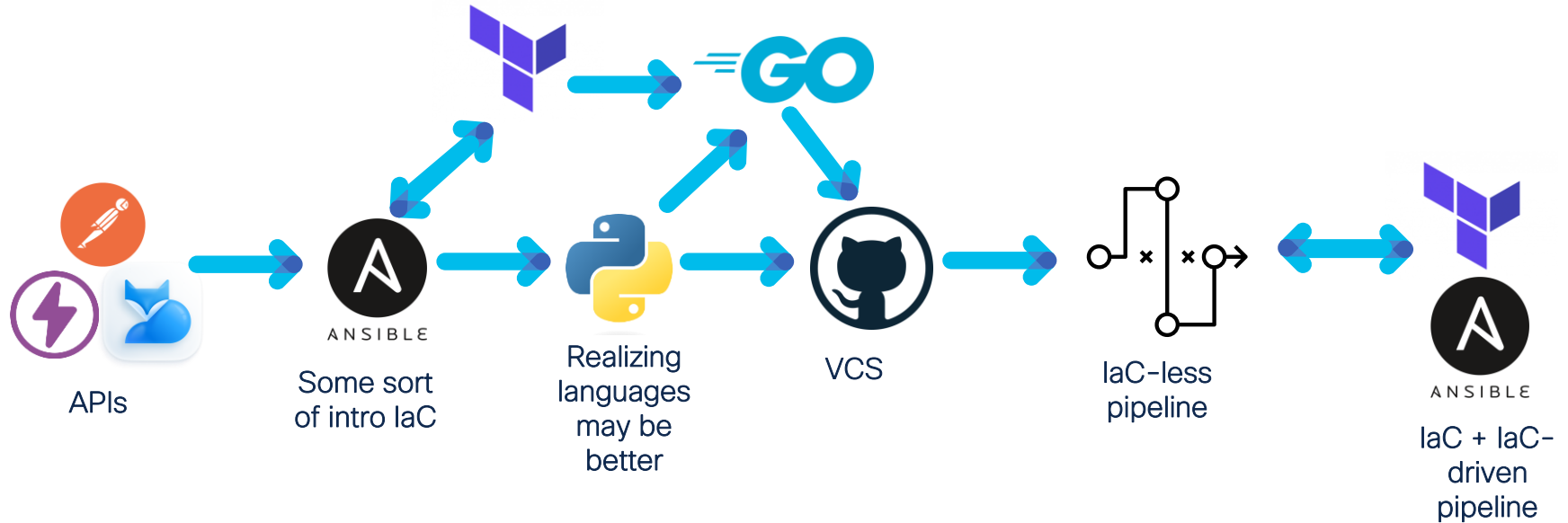
- All HCL plans of similar structure
- Possible to "read-in" for reuse without config
- Note the dotted notation nesting for relationships
- Sometimes variables are exposed without being declared as part of the provider

```
1 # Define an AWS Tenant Resource.  
2 data "aws_tenant" "tenant_obj" {  
3   name = var.tenant  
4   display_name = var.tenant  
5 }  
6  
7 # Define an AWS Schema Resource.  
8 resource "aws_schema" "schema_obj" {  
9   template_name = template_id  
10  tenant_id = data.aws_tenant.tenant_obj.id  
11 }  
12  
13 # Define an AWS Schema VRF Resource.  
14 resource "aws_schema_template_vrf" "vrf_obj" {  
15   template = aws_schema.schema_obj.template_name  
16   display_name = var.vrf  
17 }  
18  
19 # Define an AWS Schema BD Resource.  
20 resource "aws_schema_template_bd" "bd_obj" {  
21   template_name = aws_schema.schema_obj.template_name  
22   name = var.bd  
23   display_name = var.bd  
24   vrf_id = aws_schema_template_vrf.vrf_obj.id  
25   aws_schema_template_vrf = aws_schema_template_vrf.vrf_obj.name  
26   layer2_encapsulation = "prism"  
27   layer2_address = true  
28 }
```

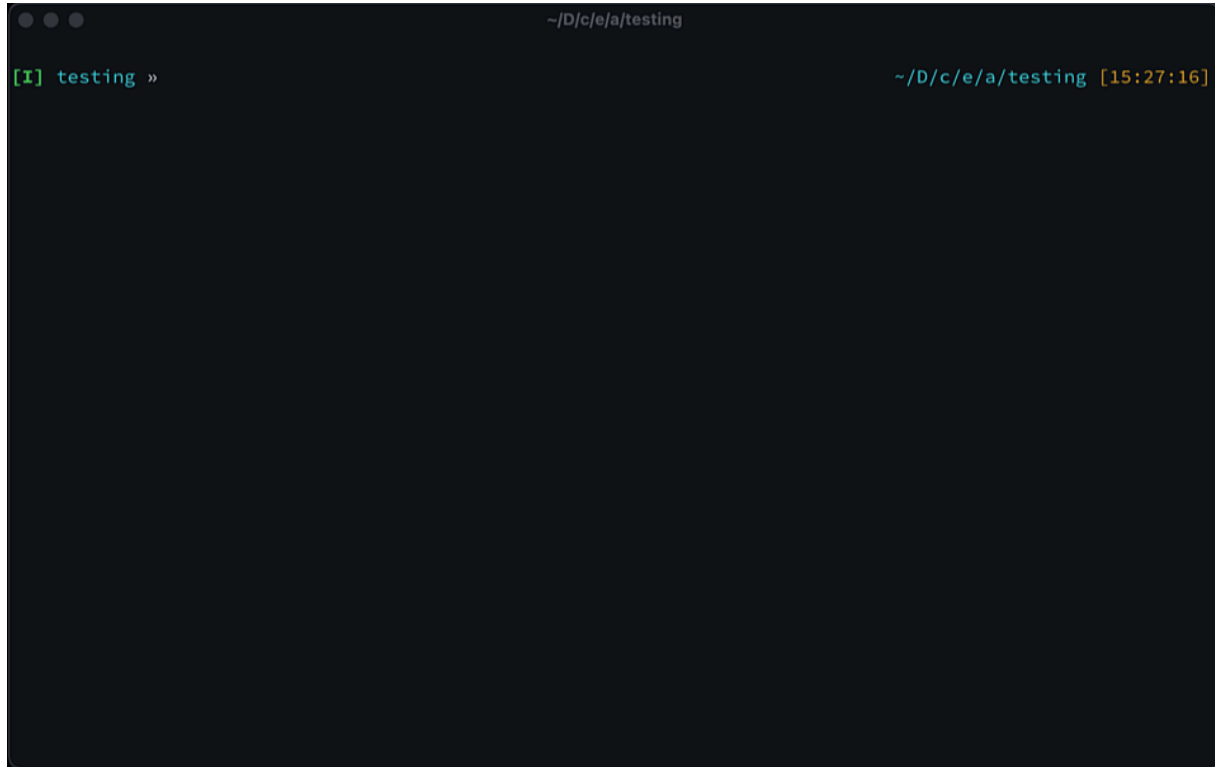
CISCO Live!

MCCOLLAB DevOps 2019 © 2022 Cisco and/or its affiliates. All rights reserved. Cisco Public

And I talk about an automation journey like this...



And IaC is GREAT for Cloud



A terminal window with a dark background. The title bar at the top shows three window control buttons on the left and the path `~/D/c/e/a/testing` on the right. The terminal content shows a green prompt `[I] testing »` on the left and a yellow timestamp `~/D/c/e/a/testing [15:27:16]` on the right.

But how does
that apply in
practice?



We can also convert click-ops into IaC



Terraformer:

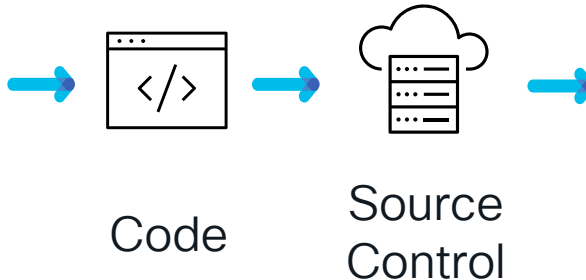
- <https://github.com/GoogleCloudPlatform/terraformer>

Azure Export for Terraform:

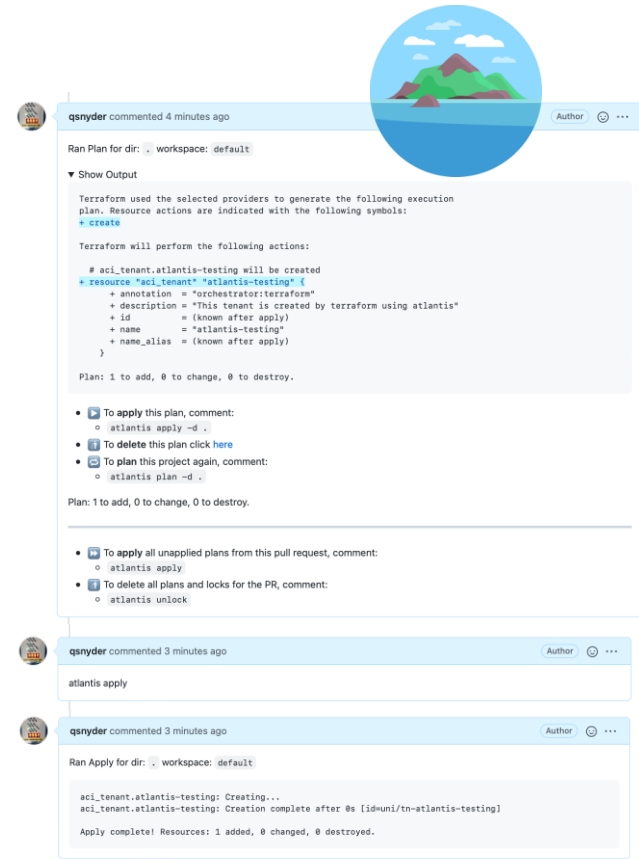
- <https://github.com/Azure/aztfexport>

laC enables easier pipelines*

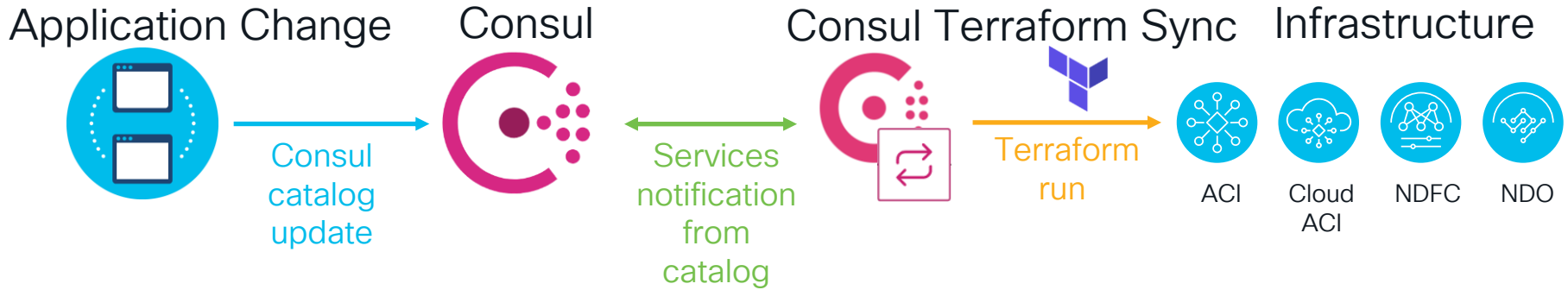
```
atlantis server \  
--atlantis-url="$URL" \  
--gh-user="$GH_USER" \  
--gh-token="$TOKEN" \  
--gh-webhook-secret="$SECRET" \  
--repo-allowlist="$GH_REPO"
```



*Please don't tell `bash` I'm cheating...



IaC enables VCS-less pipelines*



*Again, don't tell `bash` I'm cheating...

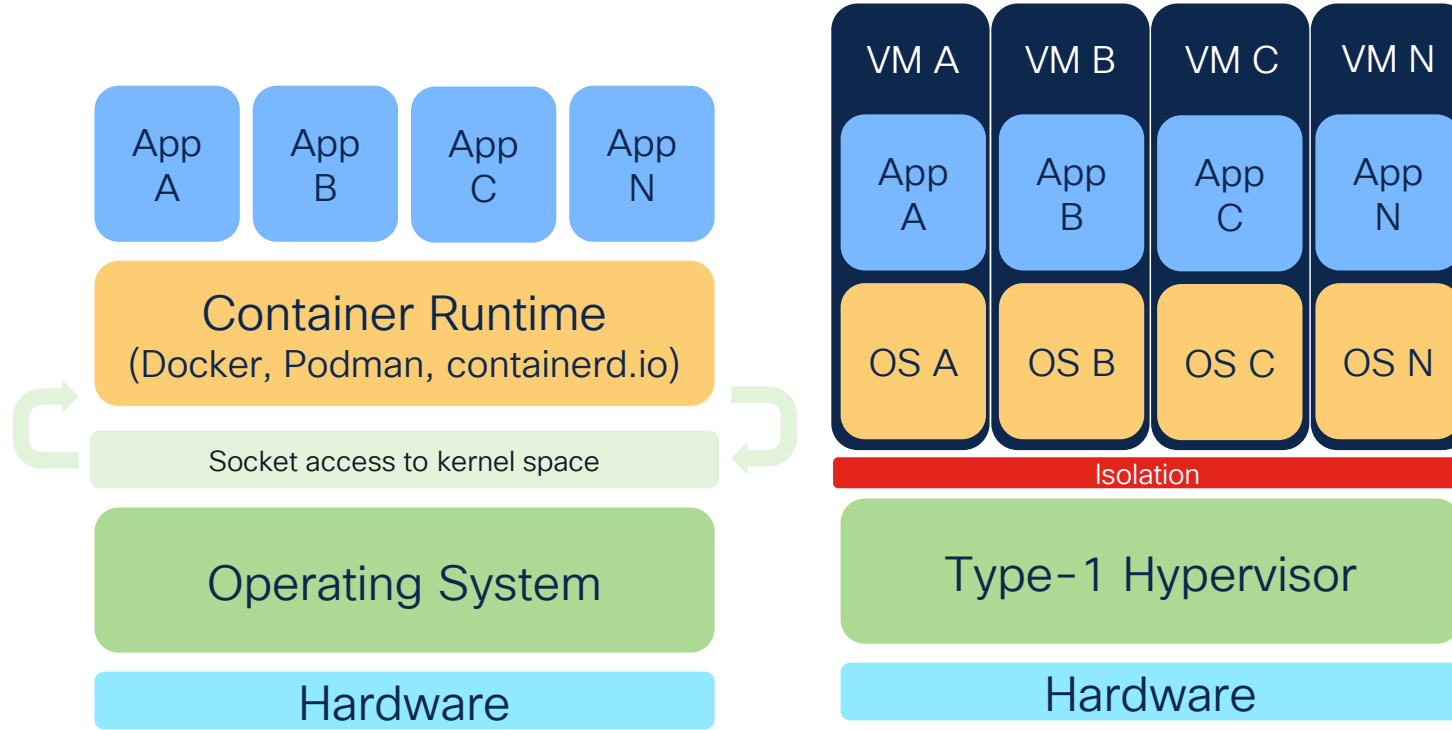
Containerize This!



I've said this
previously



What makes containers unique



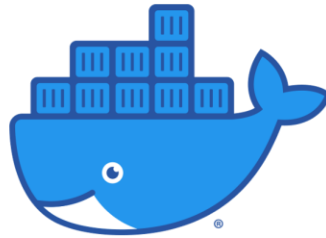
Package everything up and ship it



ANSIBLE



HashiCorp
Terraform



Containers create uniform
package runtime
environment (like Java was
supposed to do, but better)

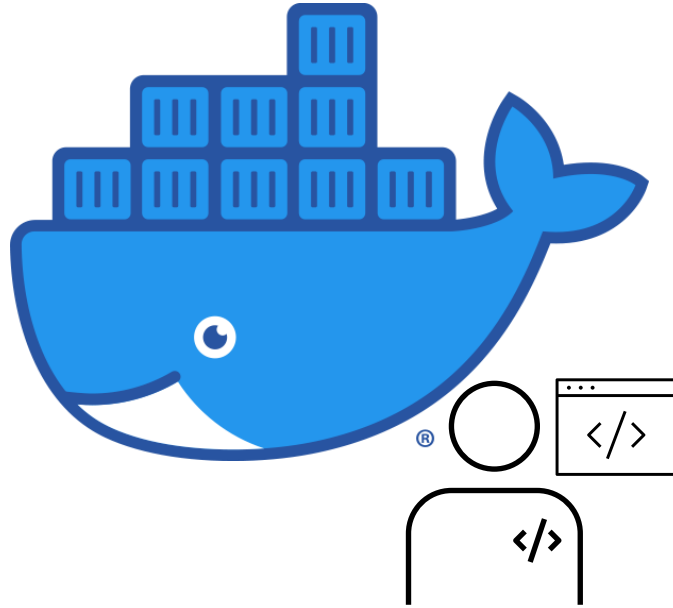
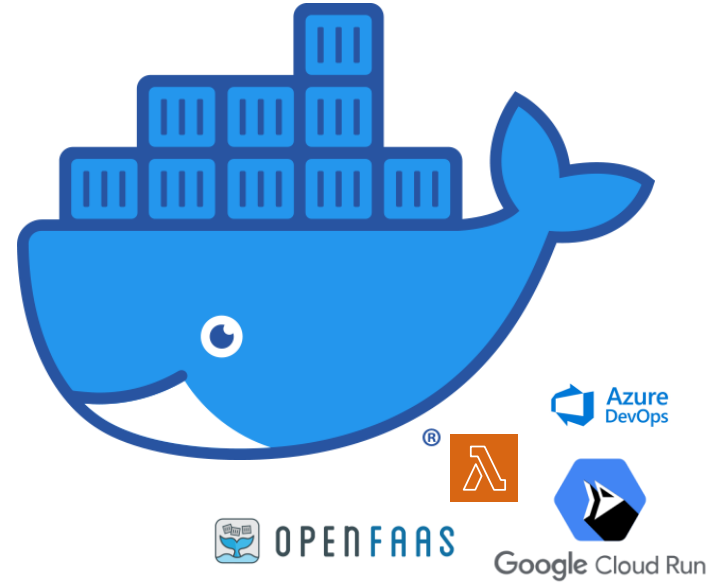
Most* abstractions survive
across OS and cloud
providers

Practically Speaking Though...

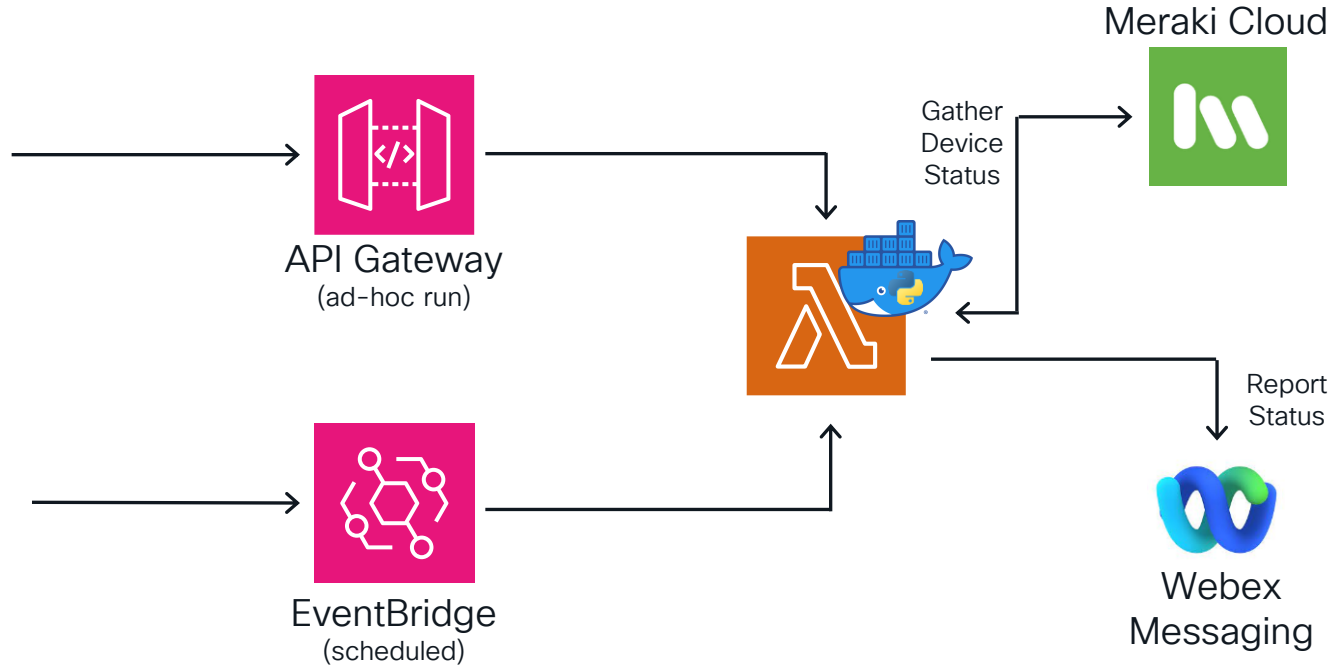
CISCO *Live!*



Containers can do both!

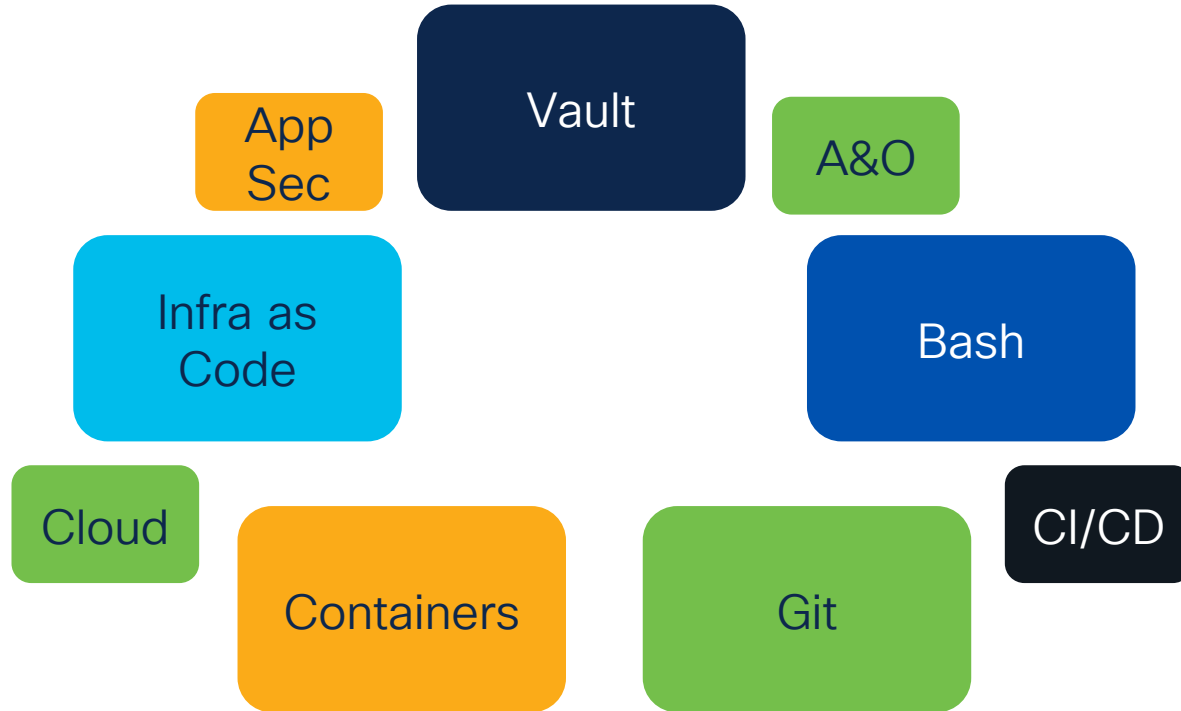


Level up containers in the Cloud!

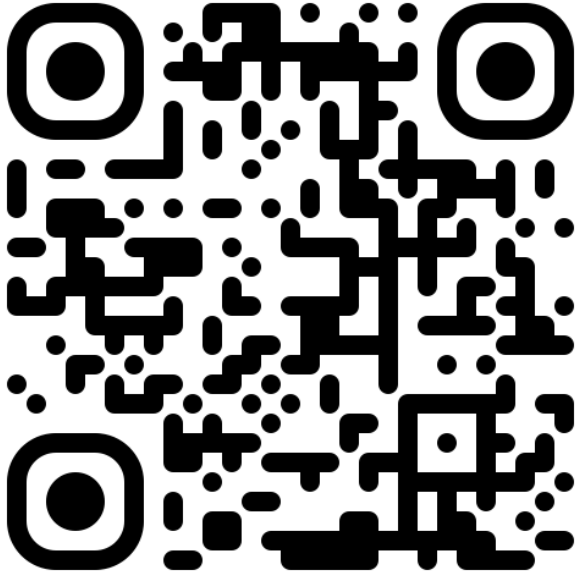


Wrapping it all up

Its all connected



We've Been Talking on These Topics for a While



<http://cs.co/devnet2149-search>

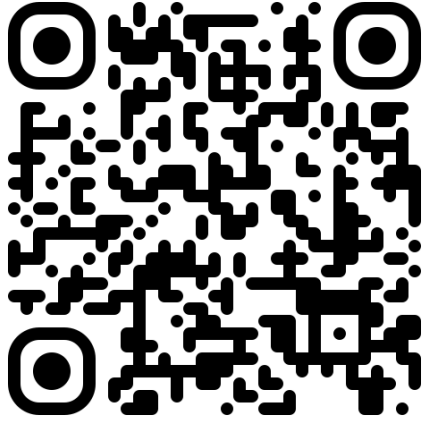
Kareem:

- DEVNET-2123
- DEVNET-1221
- BRKDEV-2003
- DEVLIT-4030

Quinn:

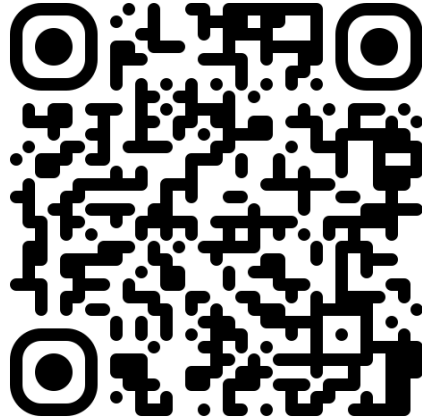
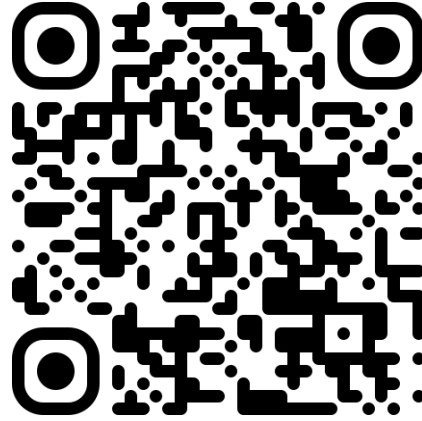
- DEVNET-2423
- DEVNET-3008
- CISCOU-2010
- CISCOU-1004
- CISCOU-2001
- DEVNET-2252

And We Have Hands-On Tutorials



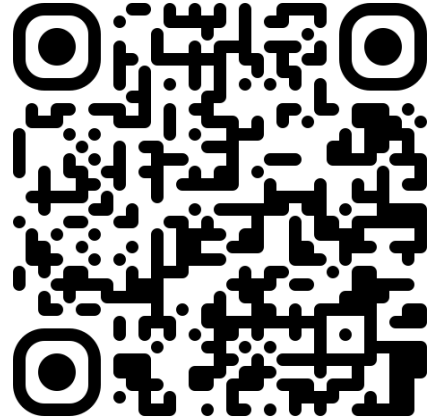
*Securing Your API Token with
Vault*

*Reverse Engineer the Cloud:
From Click-Ops to HCL*



*Deploy a Kubernetes
Cluster*

*Terraform Kubernetes
Resource Deployment*



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!*

GO BEYOND

#CiscoLiveAPJC