

Neglecting Automated Testing

Or: How To Take Down Your Network In 3 Easy Steps

Anthony Miloslavsky
🐦 @permitanyany

Cumulus Networks SE

Define Some Common Terms

- Why Infrastructure As Code (IaC)?
- Why Testing?
- Why CI/CD Pipelines?
- How Do We Bridge The Gap Between Scripting And IaC?

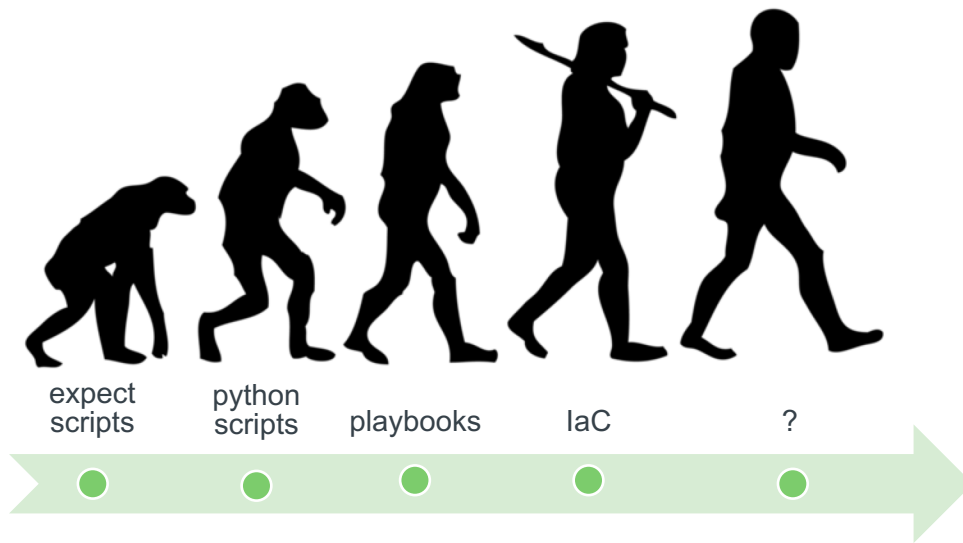
A Bit Of History

Why Is Networking So Far Behind?

- Shared infrastructure / blast radius
- If it ain't broken...don't fix it
- High risk, low reward culture
- Innovation overshadowed by fear

```
ROM: System Bootstrap, Version 11.1(17)AA  
  
        uptime is 15 years, 8 weeks,  
System restarted by power-on at 17:50:11
```

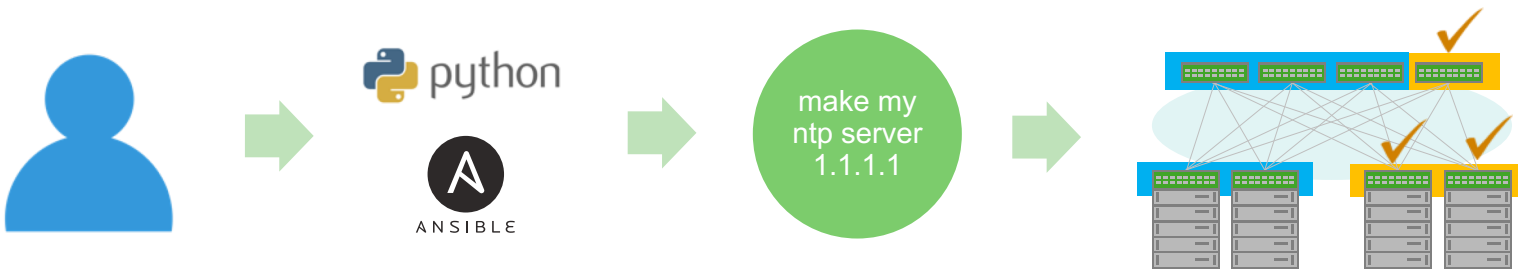
A Bit Of History



Why The Move Towards IaC?

Scripts/Playbooks

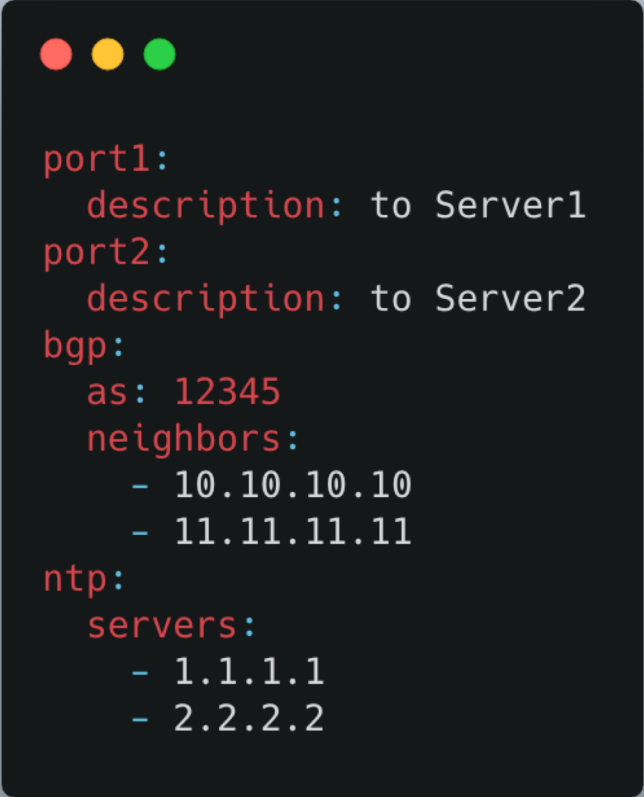
- Imperative Approach
- Minimal layers of abstraction



Why The Move Towards IaC?

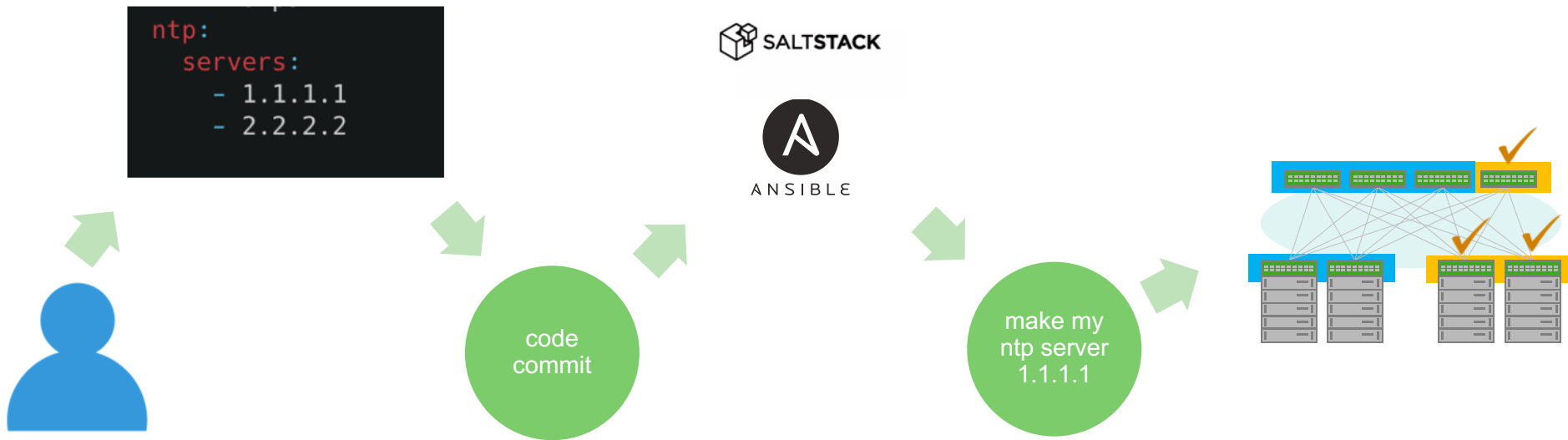
Infrastructure As Code

- Declarative Approach
- Source Of Truth
- Scalability
- Readability & Collaboration
- Reusability

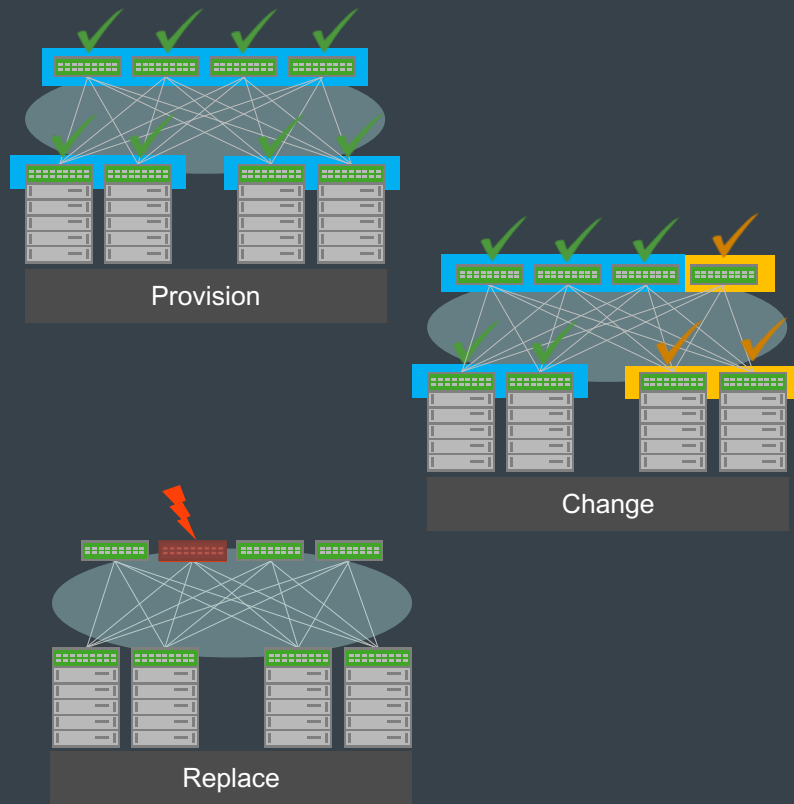
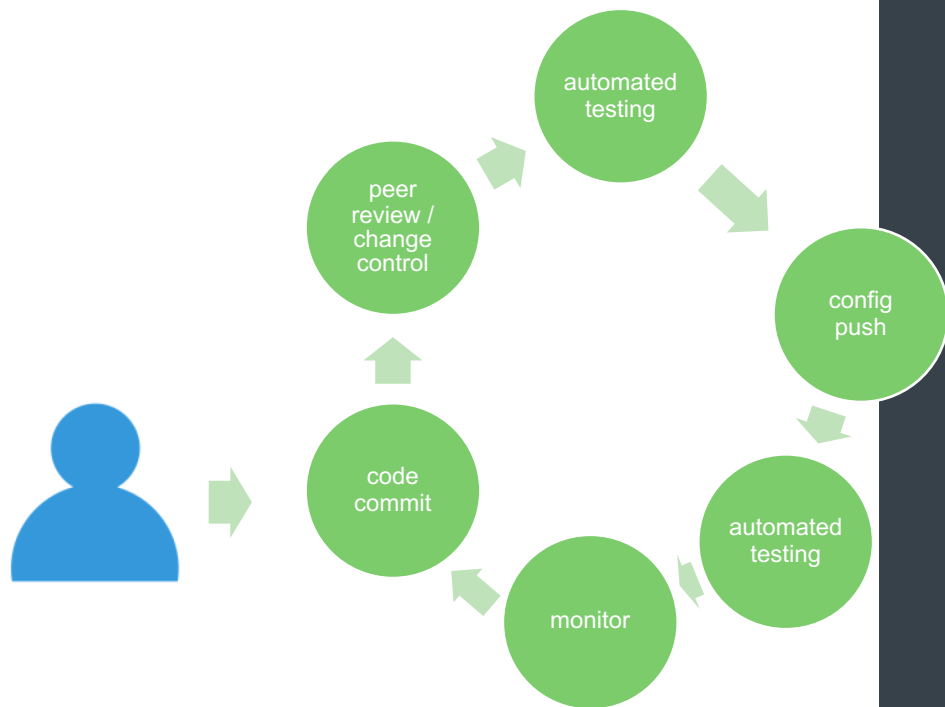


```
port1:
  description: to Server1
port2:
  description: to Server2
bgp:
  as: 12345
  neighbors:
    - 10.10.10.10
    - 11.11.11.11
ntp:
  servers:
    - 1.1.1.1
    - 2.2.2.2
```

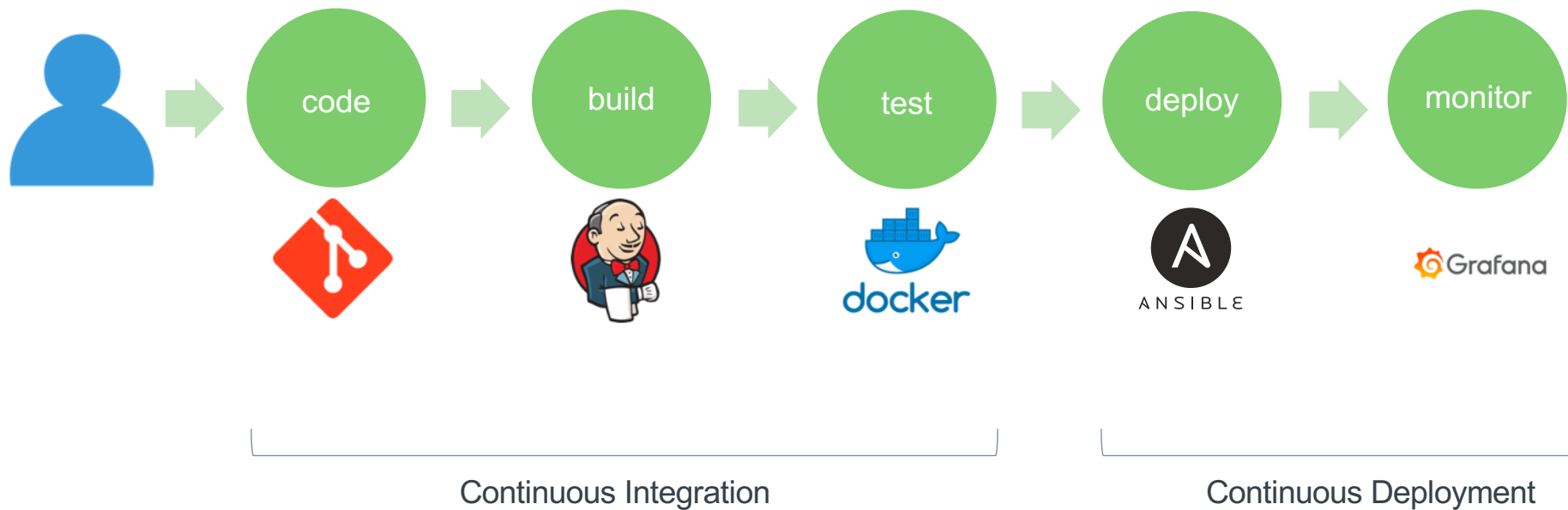
Why The Move Towards IaC?



End Goal

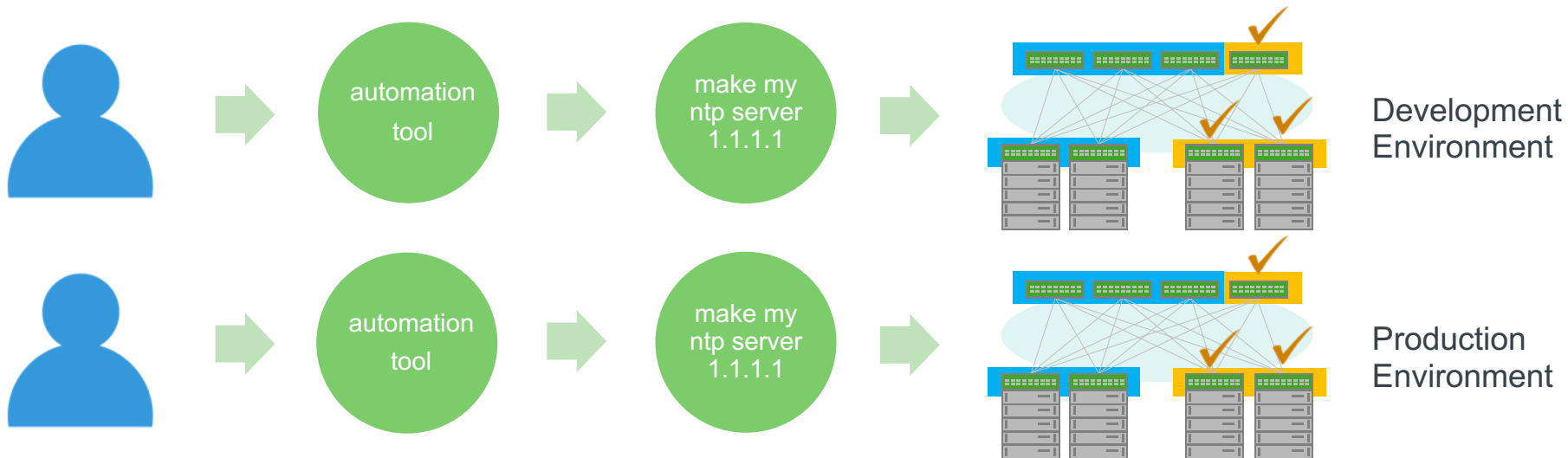


Us And Them



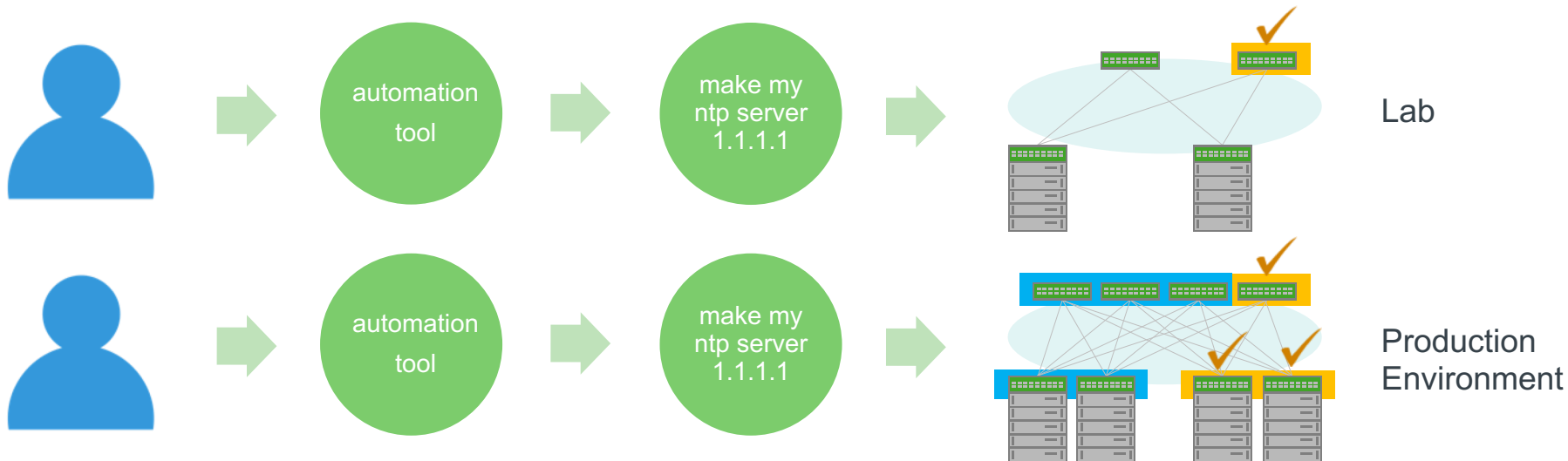
Challenges Of Testing

Does This Translate To Networking?



Challenges Of Testing

This Looks More Realistic



Levels Of Testing

Unit Testing

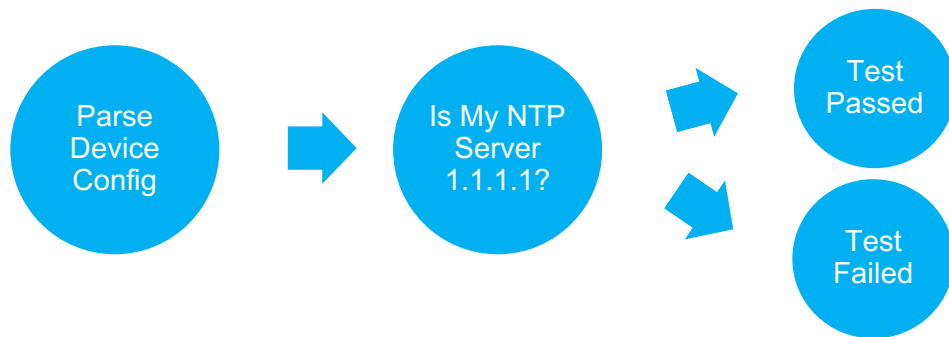
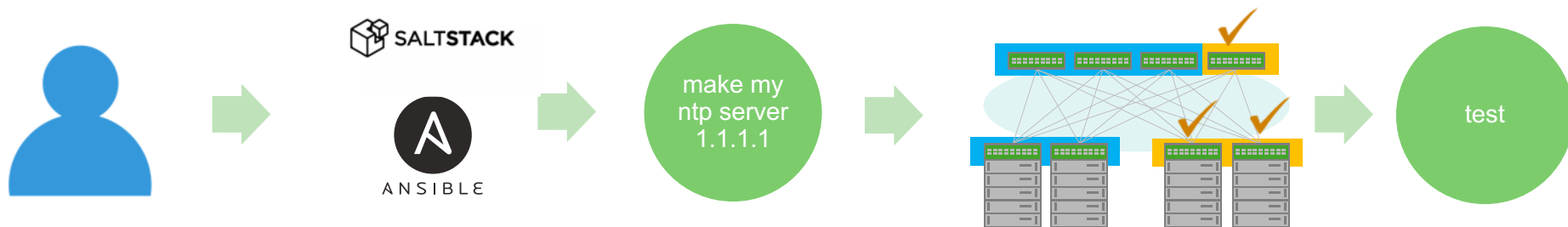
- Breaking down the problem into small pieces so that it be tested quickly
 - “I’m adding a vlan” – “Let’s confirm that the vlan was added successfully”
 - “I’ve added a new BGP prefix to my prefix list” – “Let’s confirm that I’m advertising/receiving it”

Levels Of Testing

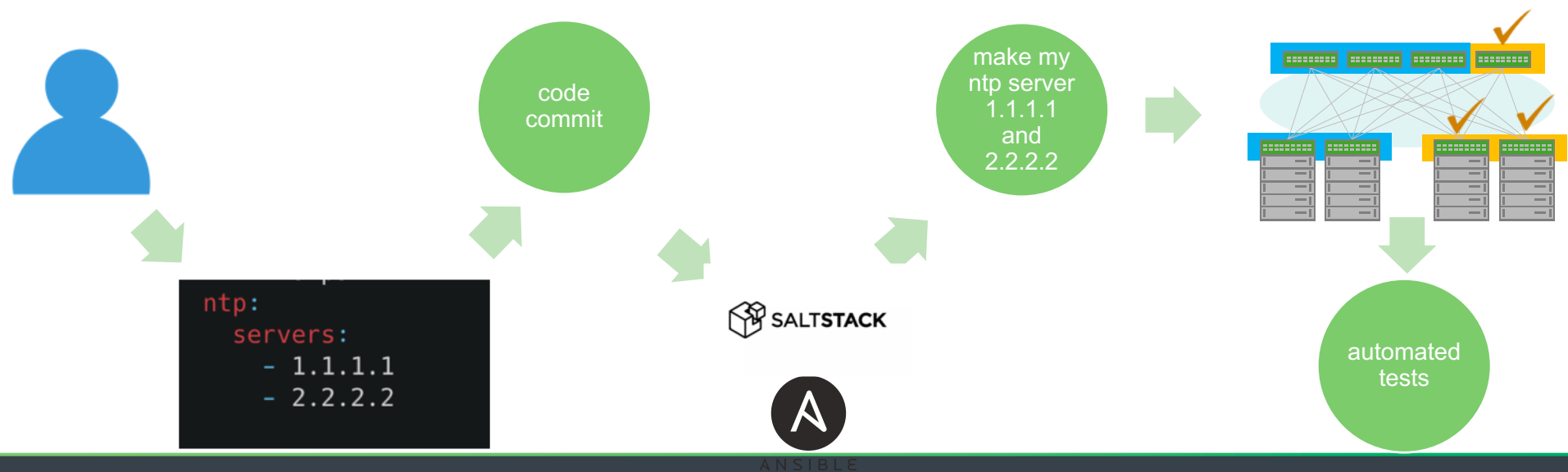
Integration Testing

- Verify how various components are interacting with each other
 - “I’m adding a new vlan” – “Let’s confirm that spanning tree looks healthy globally”
 - “I’ve added a new BGP prefix to my prefix list” – “Let’s confirm that routing looks healthy globally”

Unit Testing



Integration Testing

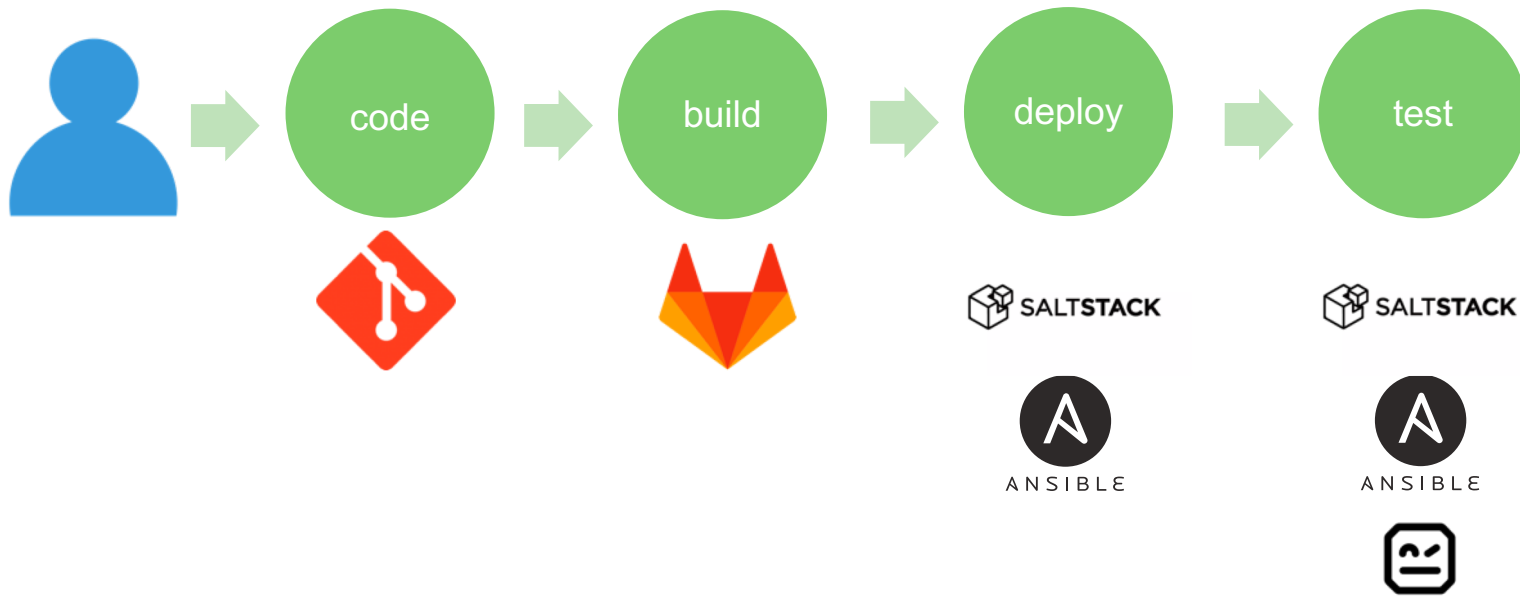


Integration Testing

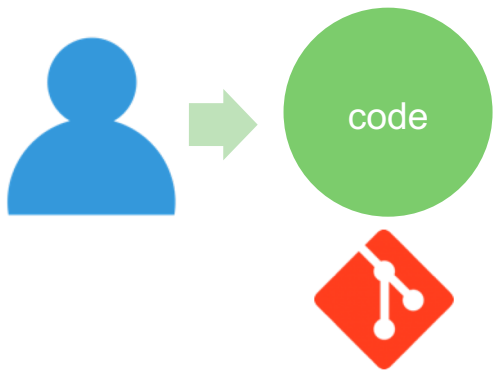
Generic & Reusable *environment* wide tests

- Overall L2/L3 Protocol Health
- MTU Mismatches
- # of routes/mroutes is in "normal" range
- NTP status
- ...Insert your favorite battle scar here

CI/CD Approach



CI/CD Approach



```
port1:
  description: to Server1
port2:
  description: to Server2
bgp:
  as: 12345
  neighbors:
    - 10.10.10.10
    - 11.11.11.11
ntp:
  servers:
    - 1.1.1.1
    - 2.2.2.2
```




```
port1:
  description: to Server1
port2:
  description: to Server2
bgp:
  as: 12345
  neighbors:
    - 10.10.10.10
    - 11.11.11.11
ntp:
  servers:
    - 1.1.1.1
    - 2.2.2.2
```



```
interface port1
description {{ port1.description }}

interface port2
description {{ port2.description }}

router bgp {{ bgp.as }}
  {% for ip in bgp.neighbors -%}
  neighbor {{ ip }} remote-as 4321
  {% endfor %}
```



```
stages:
  - deploy

deploy:
  tags:
    - deploy
  stage: deploy
  script:
    - ansible-playbook main.yml
```





```
interface port1
description {{ port1.description }}

interface port2
description {{ port2.description }}

router bgp {{ bgp.as }}
{% for ip in bgp.neighbors -%}
  neighbor {{ ip }} remote-as 4321
{% endfor %}
```



```
tasks:
  - name: Push Config
    vendor_config:
      src: leaf.j2
      transport: magicAPI
```



ANSIBLE

test



ANSIBLE

```
tasks:
  - name: Check BGP State
    shell: show bgp summary | grep "state" | grep -v "Established"
    register: bgp_check

  - name: Evaluate BGP State
    fail:
      msg: "BGP Is Currently In A Bad State"
    when: bgp_check.rc == 0
```



build

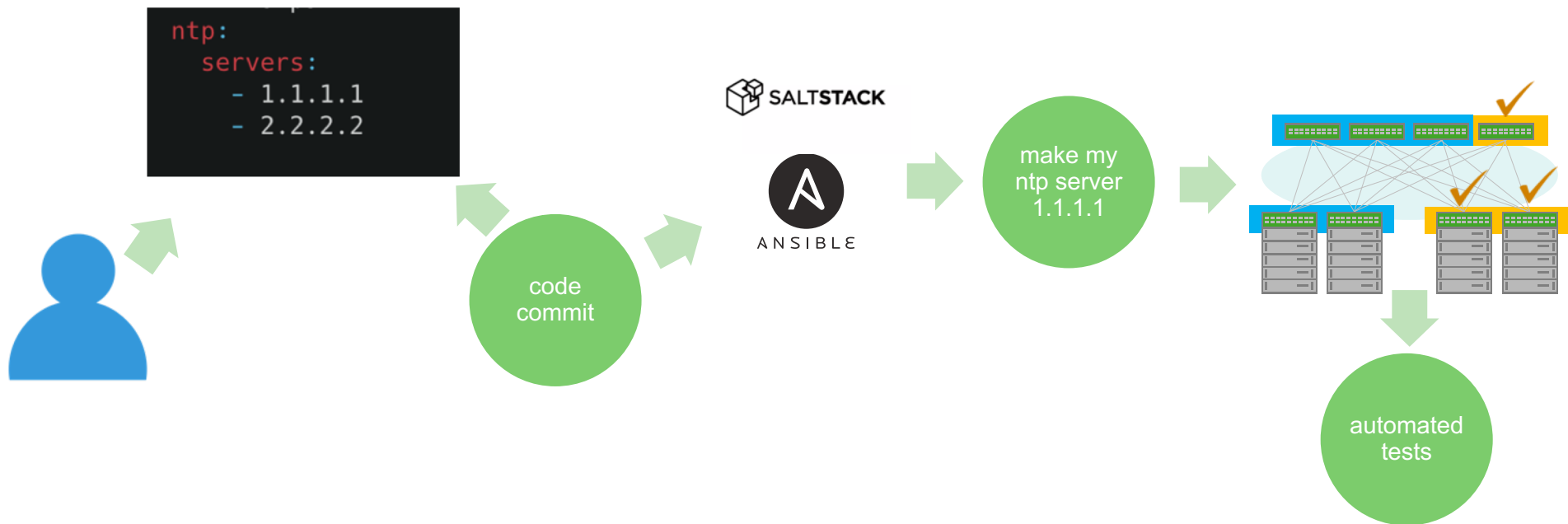


```
stages:
  - deploy
  - test

deploy:
  tags:
    - deploy
  stage: deploy
  script:
    - ansible-playbook main.yml

bgp_test:
  tags:
    - deploy
  stage: test
  script:
    - ansible-playbook test.yml
```

Is This Good Enough?



CI/CD Approach

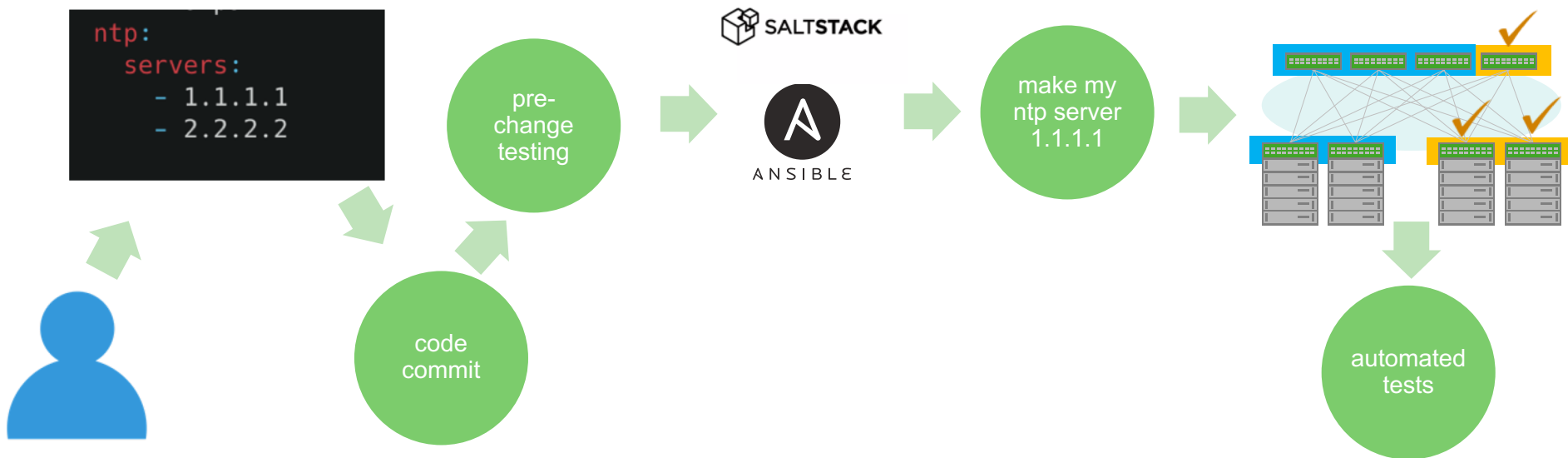
Pre-change testing

- Linting
- Pre-commit diff
- Ansible --check-mode
- Prediction Tools (Batfish, Veriflow, Forward Networks)
- Simulation



pre-
change
testing

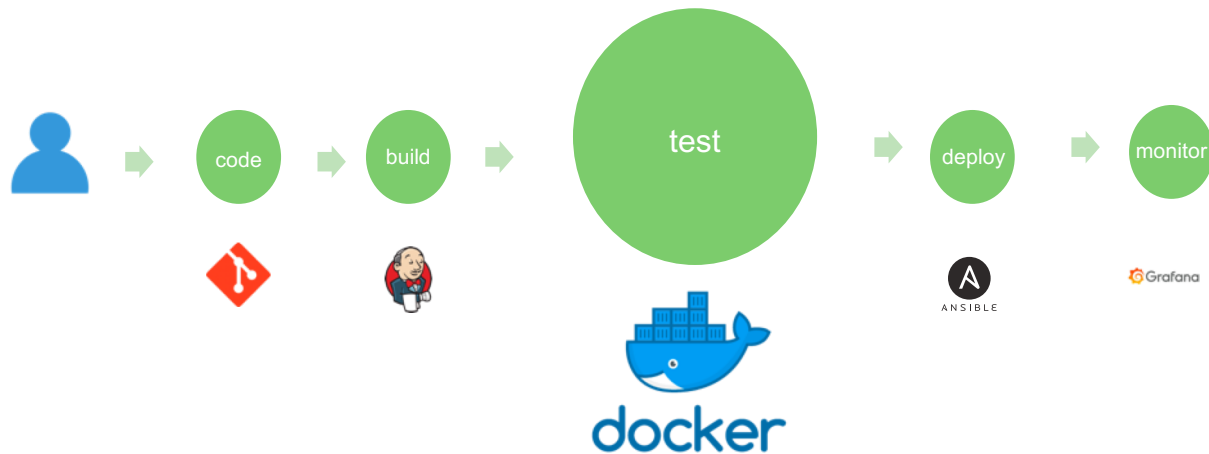
CI/CD Approach



CI/CD Approach

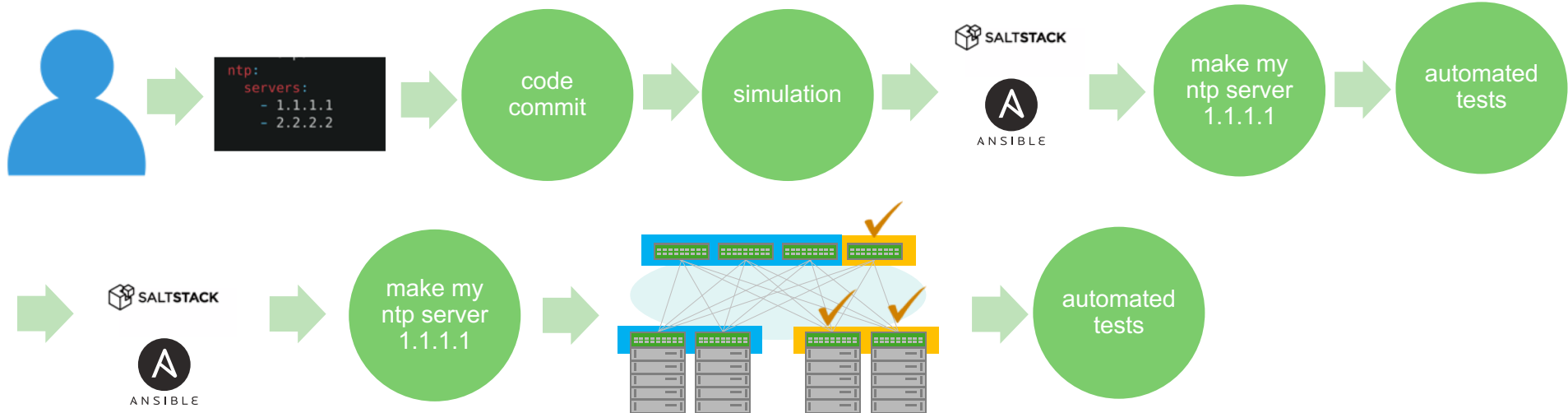
Pre-change testing

- Simulation



Levels Of Testing

Simulation



Simulation

Microsoft CrystalNet

Root Cause	Proportion	Examples	CrystalNet Coverage	Verification Coverage
Software Bugs	36%	bugs in routers, middleboxes, management tools	✓	X
Config. Bugs	27%	wrong ACL policies, traffic black holes, route leaking	✓	✓
Human Errors	6%	mis-typing, unexpected design flaws	✓	X
Hardware Failures	29%	ASIC driver failures, silent packet drops, fiber cuts, power failures	X	X
Unidentified	2%	transient failures	X	X

Table 1: Root causes of O(100) significant and customer-impacting incidents in our network (2015 - 2017).

Simulation

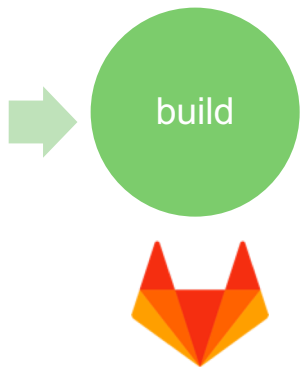
What's Required?

- Laptop/Hypervisor/Bare Metal/Public Cloud
- Multi-vendor orchestrator (Vagrant, EVE-NG, GNS3)

Is Network Simulation Ready For Primetime?

- All vendors support some version
- VMs and Containers
- Bloated Images
- Feature Parity
- Simulation Speed

CI/CD Recap



```
stages:
  - build
  - test
  - destroy
  - deploy

build:
  stage: build
  before_script:
    - cd cicd-simulate
  script:
    - vagrant up leaf01 leaf02

test:
  stage: test
  script:
    - ansible-playbook playbook.yml -i hosts.yml
    - ansible-playbook test.yml -i hosts.yml

destroy:
  stage: destroy
  before_script:
    - cd cicd-simulate
  script:
    - vagrant destroy leaf01 leaf02

deploy:
  stage: deploy
  script:
    - ansible-playbook playbook.yml -i hosts.yml
    - ansible-playbook test.yml -i hosts.yml
```

Observations/Lessons Learned

- Break the problem up into small chunks
 - Pod architecture helps here
 - Separate inventory files
- Are network engineers ready for automated CD?
- Separate branch for simulation
- New tests stem from battle scars
- Orchestration scripts – It's ok to write one-off testing
- Large/impactful changes – It's ok to write one-off testing