Understanding CI/CD in Networking Context

Josh VanDeraa

Network to Code, LLC

Agenda

- Introduction
- What is CI/CD?
- Introduce the tooling and concepts
- Demos

Introduction

- 16+ years working on networks
- Large Enterprise, SMB, MSP
- Have been automating networks for 5+ years with Python, Ansible, & CI/CD

Cisco DEVNET 500

CISCO CERTIFIED DEVNET Associate

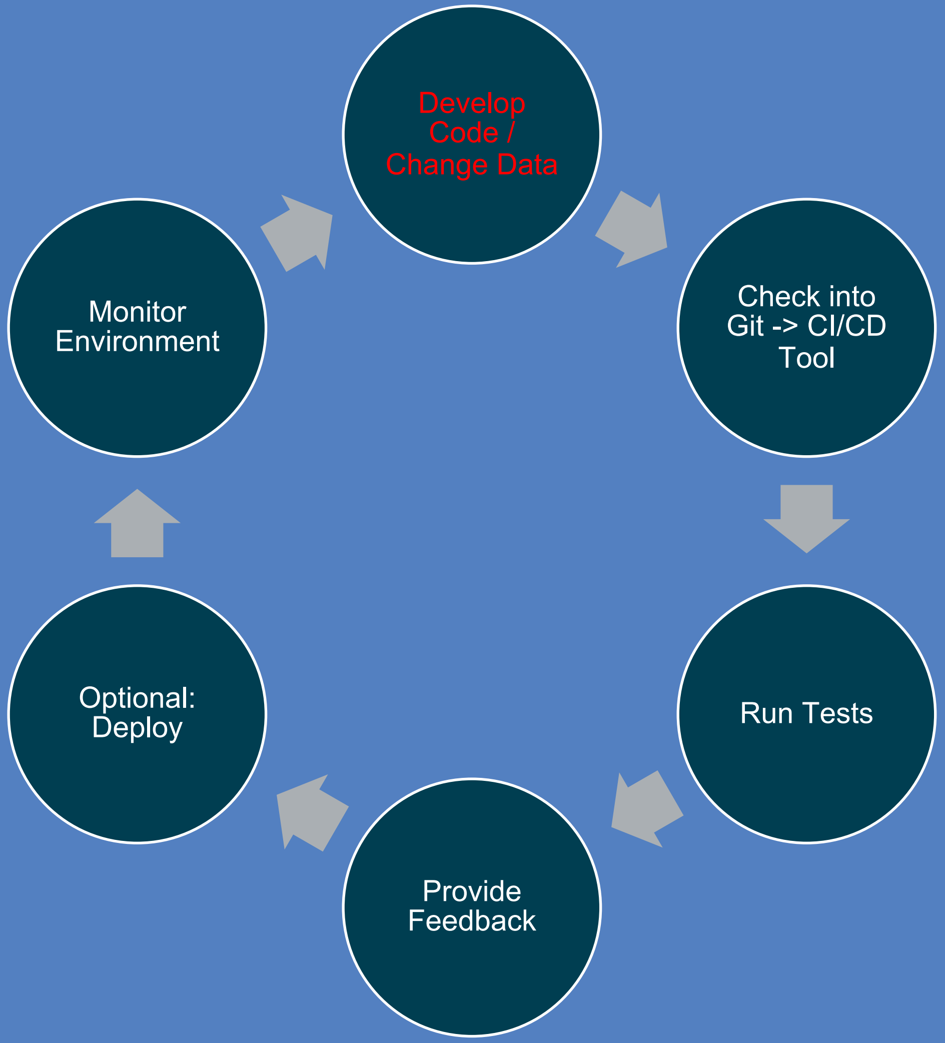
CISCO CERTIFIED DEVNET Professional

What is CI/CD?

- Continuous Integration / Continuous Deployment/Delivery
- Continuous **Automated** Testing
  - Does the data/code match organizational standards and guidelines
  - Will this generate the configuration that is expected
  - Catch the simple stuff
  - Catch the difficult to spot
  - Validate IP addressing (when setup right)
- Some have automated test suites,
  or create your own testing application

Img src: https://dzone.com/articles/what-is-cicd

Workflow



Develop Code / Change Data
Check into Git -> CI/CD Tool
Run Tests
Provide Feedback
Optional: Deploy
Monitor Environment

Testing Concepts

- Linting
  - Line length standards
  - Variable names
  - Verify modules are used
  - Doc Strings are of appropriate format
- Style Formating
  - Should you use single quotes or double quotes?
  - How many new lines between functions and methods?
- Data Validation
  - Is that an IP Address?
  - Is that a proper DNS server for the environment?
- Test Code
  - Functional programming
    - Send in controlled data, test that you get the expected result
  - Helps prevent future bugs introduced from changes to code/configuration changes

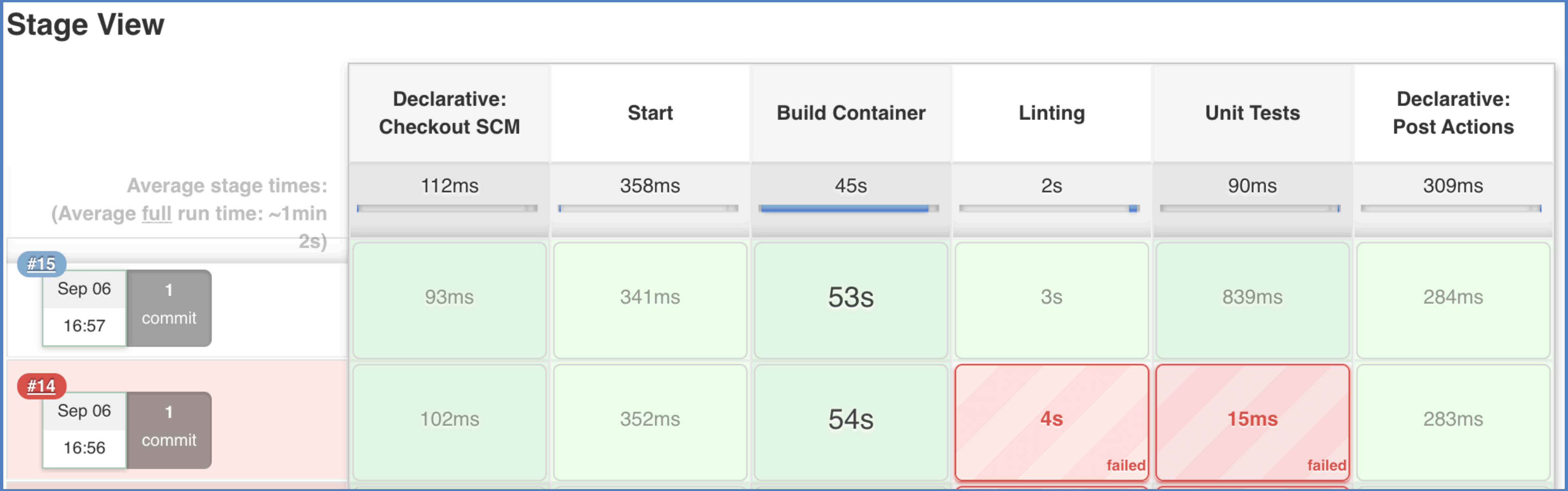
Tooling

- Python
  - Pylint
  - Bandit
  - Pytest
- Ansible
  - Ansible Lint
- Data
  - YAMLLint
  - JSON Schema
- Makefile
- Docker containers

Tooling

- Brief Search > 50 platforms
- Examples
  - Jenkins
  - Travis-CI
  - GitHub Actions
  - Gitlab CI
  - Drone.io
  - Circle CI
  - Microsoft TFS
  - Cloud platforms

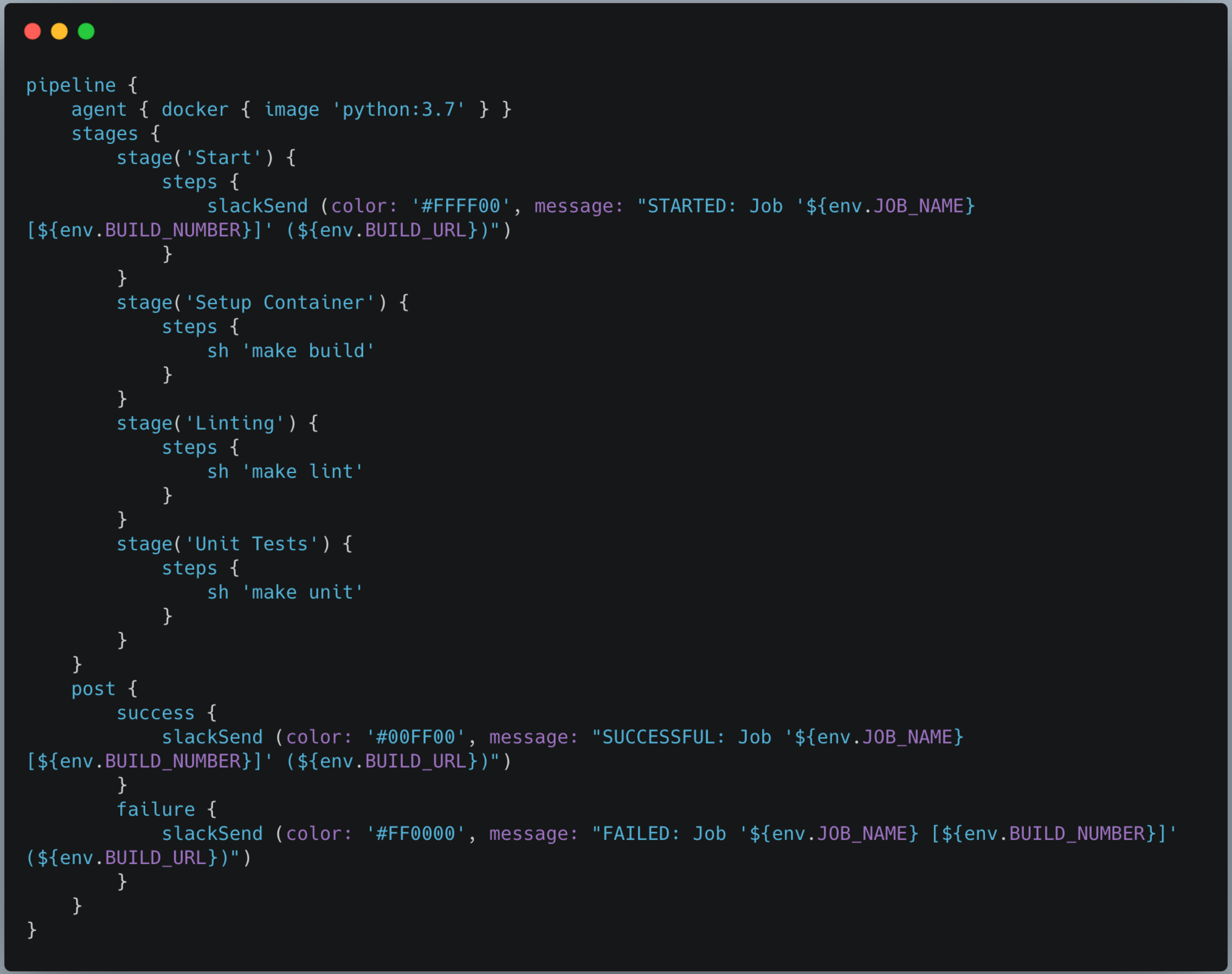
Jenkins Example



Stage View

	Declarative: Checkout SCM	Start	Build Container	Linting	Unit Tests	Declarative: Post Actions
Average stage times: (Average full run time: ~1min 2s)	112ms	358ms	45s	2s	90ms	309ms
#15 Sep 06 16:57 1 commit	93ms	341ms	53s	3s	839ms	284ms
#14 Sep 06 16:56 1 commit	102ms	352ms	54s	4s failed	15ms failed	283ms

Jenkinsfile



```
pipeline {
    agent { docker { image 'python:3.7' } }
    stages {
        stage('Start') {
            steps {
                slackSend (color: '#FFFF00', message: "STARTED: Job '${env.JOB_NAME}
[${env.BUILD_NUMBER}]' (${env.BUILD_URL})")
            }
        }
        stage('Setup Container') {
            steps {
                sh 'make build'
            }
        }
        stage('Linting') {
            steps {
                sh 'make lint'
            }
        }
        stage('Unit Tests') {
            steps {
                sh 'make unit'
            }
        }
    }
    post {
        success {
            slackSend (color: '#00FF00', message: "SUCCESSFUL: Job '${env.JOB_NAME}
[${env.BUILD_NUMBER}]' (${env.BUILD_URL})")
        }
        failure {
            slackSend (color: '#FF0000', message: "FAILED: Job '${env.JOB_NAME} [${env.BUILD_NUMBER}]'
(${env.BUILD_URL})")
        }
    }
}
```

Demos Overview

- Walk thru Linting
- Introduce pytest and testing via Python
- All done in containers
  - Consistent development experience
  - Portable to the CI/CD system
  - Simplified with Make in this example, Python Invoke is an option

Demos

Subtitle

Contact Me

networktocode.slack.com: jvanderaa

@vanderaaj

jvanderaa

https://www.linkedin.com/in/josh-vanderaa/