Building a Terraform Pipeline and Establishing Integration with Source Control

Kyler Middleton
NETWORK AND DEVOPS ARCHITECT

@kymidd www.kylermiddleton.com

Overview

CI/CD pipelines

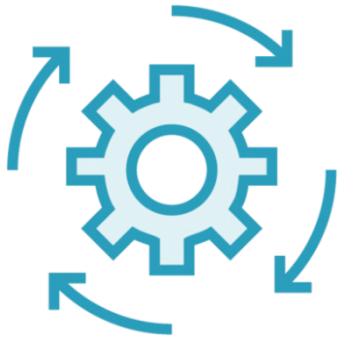
Automatic validation

Automatic deployment

Pipelines as Code

CI/CD Process: A Definition

CI/CD Pipelines

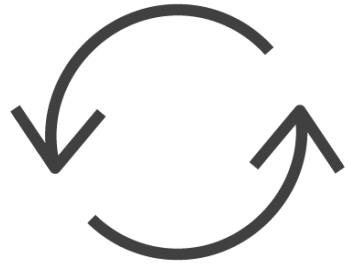


Continuous Integration

Testing code changes and human approvals

Continuous Deployment

Deploying approved code to environments



CI/CD Process

Whole process to test and approve code, build resources

Defining a Pipeline

Defining a Pipeline: YAML

.travis.yml

```yaml
# Define environment
os: linux  # The default OS
dist: xenial # Ubuntu 16.04

# Define language
language: shell  # aka, bash on Linux

# Only build PRs against master branch or pushes to master branch
branches:
  only:
    - master
```

Defining a Pipeline: Operating System

Ubuntu Bionic 18.04

Ubuntu Xenial 16.04

Ubuntu Trusty 14.04

Ubuntu Precise 12.04

macOS

Windows Server 1803 (Early Release)

Virtualization environments

Each build runs in one of the following virtual environments.

Linux

A sudo enabled, full virtual machine per build, that runs Linux, one of:

- Ubuntu Bionic 18.04
- Ubuntu Xenial 16.04 **default**
- Ubuntu Trusty 14.04
- Ubuntu Precise 12.04

LXD compliant OS images for arm64 are run in Packet. Have a look at Building on Multiple CPU Architectures for more information.

macOS

A macOS environment for Objective-C and other macOS specific projects

Windows

A Windows environment running Windows Server, version 1803.

https://docs.travis-ci.com/user/reference/overview/#virtualization-environments

Ubuntu Linux LXD Container (Early Release)

Virtualisation Environment vs Operating System

The following table summarizes the differences across virtual environments and operating systems:

	Ubuntu Linux (Bionic, Xenial , Trusty, Precise)	macOS	Windows	Ubuntu Linux / LXD container with (Bionic), Xenial
Name	Ubuntu	macOS	Windows	Ubuntu
Status	Current	Current	Early release	Early release
Infrastructure	Virtual machine on GCE	Virtual machine	Virtual machine on GCE	ARM: LXD container on Packet IBM Power: LXD container on IBM Cloud IBM Z: LXD container on IBM Cloud
CPU architecture	amd64	amd64	amd64	arm64 (armv8) ppc64le (IBM Power) s390x (IBM Z)
`.travis.yml`	`dist: bionic` or `dist: xenial` or `dist: trusty` or `dist: precise`	`os: osx`	`os: windows`	`os: linux arch: arm64 dist: bionic` or `os: linux arch: arm64 dist: xenial` or `os: linux arch: ppc64le dist: bionic` or `os: linux arch: ppc64le dist: xenial` or `os: linux arch: s390x dist: bionic` or `os: linux arch: s390x dist: xenial`

https://docs.travis-ci.com/user/reference/overview/#virtualization-environments

Programming Languages

Programming Languages

- Android
- C
- C#
- C++
- Clojure
- Crystal
- D
- Dart
- Elixir
- Elm
- Erlang
- F#
- Generic
- Go
- Groovy
- Haskell
- Haxe
- Java
- JavaScript (with Node.js)
- Julia
- Minimal
- Nix
- Objective-C
- Perl
- Perl6
- PHP
- Python
- R
- Ruby
- Rust
- Scala
- Smalltalk
- Swift
- Visual Basic
- Adding a language

So many I could barely fit them here

Each language supports custom configs

https://docs.travis-ci.com/user/languages/android/

Defining a Pipeline: Git Branch Filter

Environment-specific testing

Blocklist (except)

Safelist (only)

.travis.yml

```yaml
# blocklist
branches:
  except:
  - legacy
  - experimental

# safelist
branches:
  only:
  - master
  - stable
```

https://docs.travis-ci.com/user/customizing-the-build/#building-specific-branches

Defining a Pipeline: Variables

Defining a Pipeline: Why Variables?

Repeated Usage

Storage once, reference many times. Travis calls these Global variables

Secure Values

Store passwords, API keys as write-only secret values

Parallel Testing

Concurrent testing with multiple variable values. Travis calls these "environment" variables

https://docs.travis-ci.com/user/speeding-up-the-build

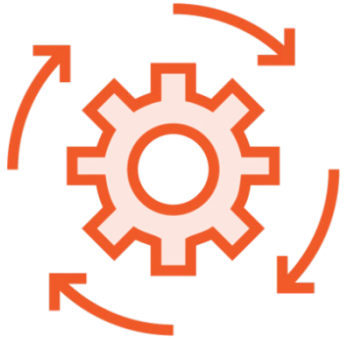
What Makes a Pipeline?

Running a Pipeline: Jobs, Builds, and Stages

Let's establish some order and hierarchy on our pipelines

Job

List of instructions, clones repo and runs instructions in order, has phases



Build

Group of jobs and stages, finishes when all jobs have completed

Stage

Group of jobs which can be run in parallel as part of a build

Running a Pipeline: Phases

Jobs are made up of phases

Implicit phases

Major: Install, Script

1. OPTIONAL Install `apt addons`
2. OPTIONAL Install `cache components`
3. `before_install`
4. `install`
5. `before_script`
6. `script`
7. OPTIONAL `before_cache` (for cleaning up cache)
8. `after_success` or `after_failure`
9. OPTIONAL `before_deploy`
10. OPTIONAL `deploy`
11. OPTIONAL `after_deploy`
12. `after_script`

https://docs.travis-ci.com/user/customizing-the-build/#building-specific-branches

Running a Pipeline: Logic

If/then

Conditions list

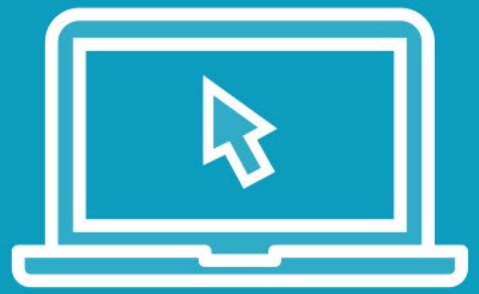
.travis.yml

```
stages:
  - name: deploy
    # require the branch name to be master (note for PRs this is the base branch name)
    if: branch = master
```

- `type` (the current event type, known event types are: `push`, `pull_request`, `api`, `cron`)
- `repo` (the current repository slug `owner_name/name`)
- `branch` (the current branch name; for pull requests: the base branch name)
- `tag` (the current tag name)
- `commit_message` (the current commit message)
- `sender` (the event sender's login name)
- `fork` (`true` or `false` depending if the repository is a fork)
- `head_repo` (for pull requests: the head repository slug `owner_name/name`)
- `head_branch` (for pull requests: the head repository branch name)
- `os` (the operating system)
- `language` (the build language)
- `sudo` (sudo access)
- `dist` (the distribution)
- `group` (the image group)

https://docs.travis-ci.com/user/conditions-v1

Demo



Build .travis.yml pipeline

- Define operating system
- Define language
- Branch targeting
- Environmental (global) variables
- Terraform validate on PR
- Terraform apply on master update

Set secure IAM variables in TravisCI

Create pull request to build resources