Cisco DevOps 300-910: CI/CD Pipelines

ESTABLISHING BUILDING BLOCKS FOR INFRASTRUCTURE AS CODE

Kyler Middleton
NETWORK AND DEVOPS ARCHITECT

@kymidd www.kylermiddleton.com

Overview

The time before DevOps

How to git

Automating the automation

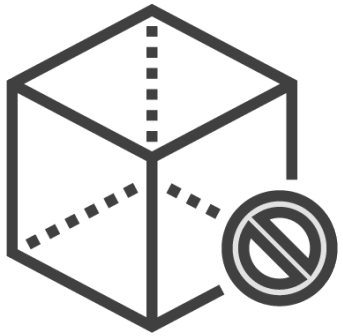
Cloudy authorization

The world eater

- More effective learning
- Multiply impact

The Non-DevOps Model

Non-DevOps Hallmarks



Change Management

Changes are bundled into massive periodic change windows

Automation

Where automation exists, it is old, complex, and difficult

Auditability

Finding out who changes what, when is difficult or impossible

Managing Code with Git

Git Vocabulary

Repo

Collections of code tracked by git server

Clone

Copy remote repo to your computer

Commit

Collection of changes to repo, saved into git

Branch

A workspace to update code in

Pull Request

Proposed changes to repo, can be approved

DevOps Tooling: What's out There?

Infrastructure Provisioning: CloudFormation, Terraform

Establishes resources

Builds resources like virtual machines, subnets, and routing

Limited Post-build Configuration

Doesn't install software, updates, antivirus

Endpoint Management: Chef, Puppet, Ansible

Post-Build Configuration

Takes over configuration of built resources – updates, antivirus, triggered actions

OS Integration

Deep hooks into operating system for ongoing management

Fleet Management

Often used to manage many machines at a time with consistent policies

Procedural vs. Declarative

Procedural

Procedural tools aren't aware of past runs. Admins define actions which are executed the same way each time the tool runs.

Examples: Ansible, Chef

Declarative

Declarative tools allow an admin to define a target configuration for a resource. The tool is aware of past runs and will figure out how to get you there.

Examples: CloudFormation, Terraform, Puppet

Terraform Overview

Terraform Essentials

Terraform differs from other provisioning tools in a few key ways.

- Support for many providers
- State file required

How does Terraform work?

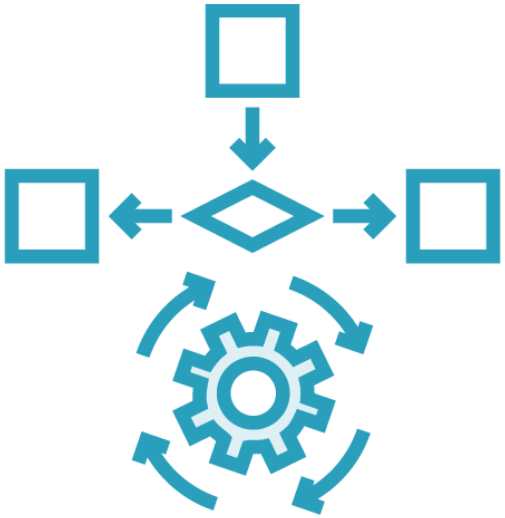
Sync actual state of environment to state file

Compare terraform configuration of environment with state file

True up differences discovered so environment matches configuration

Centralizing Automation with CI/CDs

Automating the Automation



Pipelines

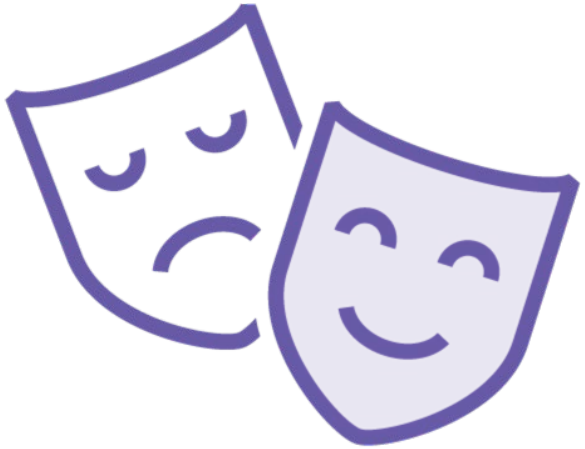
Validation, testing, executing automation jobs

Approvals

Admin permissions for all only when allowed

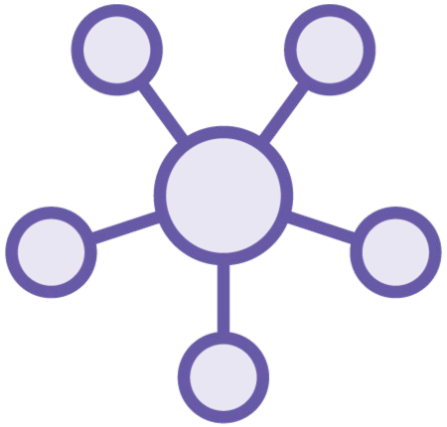
AWS IAM Permissions Primer

AWS Users, Roles, and Policies



IAM User

A profile intended for a single user, linked to policies



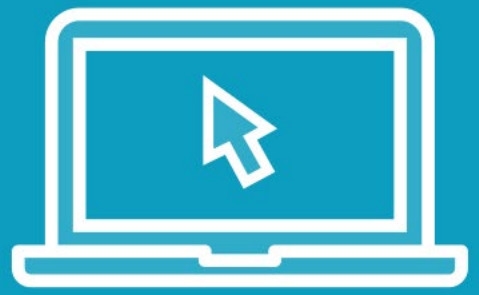
IAM Role

A profile to be assumed by any services that needs it

IAM Policy

List of permissions to be granted

Demo



Build AWS resources for Terraform

- IAM user and secret
- S3 bucket to store state file
- DynamoDB for concurrency locking

Pivot Terraform state file to S3 bucket

Create GitHub repo for terraform code

Integrate GitHub repo with TravisCI