CompoZed Labs

Untangling Platform Complexity with Concourse CI

Summary

- Product mindset drives an automation heavy culture
- We iterated our way to success
- We have created engineering capacity through automation
- You can do it too!

Culture

Platform as a Product

- Customer
  - Developers that need to deploy operable software quickly and reliably without friction.
- Problem
  - Developers spend too much time on process heavy lifing and non-business value added work.
  - Developers expect access to the latest and greatest technolgy for use with their products.
- Solution
  - An integrated platform with connected services that make the right thing the easy thing.
  - Minimize the operational burdon of operating the overall platform.

The Starting Point

- ELK
- 6 Cloud Foundry Foundations in 3 environments (Need a pic)
- No Production
- No Automation
- Several tiles (Mobile push, mySQL, Redis)

Too Much Time Maintaining CF

- Basically 100% of engineering time
  - Tiles broke frequently (Upgrade order matters)
  - e.g. RabbitMQ must be upgraded before mobile push
  - Lack of consistency

(Operational Tax Rate was high)

Our Goal

- Reduce the operational burden of running and scaling the platform.
- No more #@$*^$ GUI's.
- Be able to reliably and quickly build new environments
- Ensure that our CF environments look consistent

Some Graphic about more of the same stuff

Solution Requirements

- Environments can be represented and versioned through source control
- Automated testing for all platform components
- Support the complex workflow of deploying CF
- Manage all the things

Simple gif of new stemcell trigging a deploy of a tile in Dev

- Inputs:
  - Component -> Tile
  - Image -> Stemcell
  - Configuration -> YAML
- JOB
- Outputs

Definitions

- Task
- Resource
- Pipeline

The Journey

Attempt Number One

Single pipeline to rule them all (pic)

- Let concourse manage platform components (concourse resource)
  - Tiles
  - Ops Manager
  - Stemcells
- Let concourse and git manage configurations (concourse resource)
- Bash to deploy ops manager appliance (concourse tasks)
- Bash to deploy using experimental ops manager api (concourse tasks)
- Needed to vendor packages into internal artifact repository

Benefits

- Codified tile dependencies
- Each environment was perfectly consistent
- New environment provisioning went from 1 week to 8 hours
- Engineering time for a deployment went from 40 hours to 3 hours

Problems

- Concourse pipeline definition file was large and repetitive (2-3K Lines)
- Managing multiple concurrent versions of Ops Manager was difficult with Bash
- Concourse missing proxy support

Attempt Number Two

- Generate single pipelines per product
  - Ops Manager
  - Tiles: Elastic Runtime, MySQL, Redis
- Build ruby gem to replace Bash Tasks
  - Integration Tests for Ops Manager API breaking changes
- Manage, release and deploy our own fork of concourse
  - Added proxy support

Benefits

- We were able to build only the components that changed
- Pipeline definitions were smaller and more and responsible for a single component
- Fail faster when there were breaking Ops Manager API changes
- Easier management of multiple versions of Ops Manager API
- Can pull resources directly from Pivotal Network

Problems

- Concourse pipeline definitions were repetitive
- We had forked Concourse, so we had to merge changes

Attempt Three

- Wrote a generation tool for Concourse Pipelines (Travel Agent)
- Moved to latest open source version of Concourse
- Fully integrated Pivotal Network resource

Benefits

- Reuced pipeline definitions from 400 line yaml to a template + 40 line config
- Reduced human error by not repeating ourselves in pipeline definitions
- Built in validation of generated pipelines
- We no longer have to maintain a fork of Concourse
- We no longer maintain Bash scripts to pull from Pivotal Network
- Engineering time for a deployment went from 3 hours to 2 hours

Results

- 6 Engineers
- 10 Foundations
- X Pipelines
- 2 Datacenters
- X servers
- Features we have deployed because we have not been patching

Lessons Learned

- Iteration and learning is key and part of the journey
- Start with triggers turned off
- Don't be afraid to code
- Focus on removing error from the system
- Smaller pipelines with a single responsibility reduce complexity
  - Faster feedback
  - easier to reason about

How to get started

- Awesome Stark and Wayne Tutorial
- Start with Ops Manager
- Evolve to BOSH Director
- Evolve to Cloud Foundry
- Evolve to the platform

We

Concourse