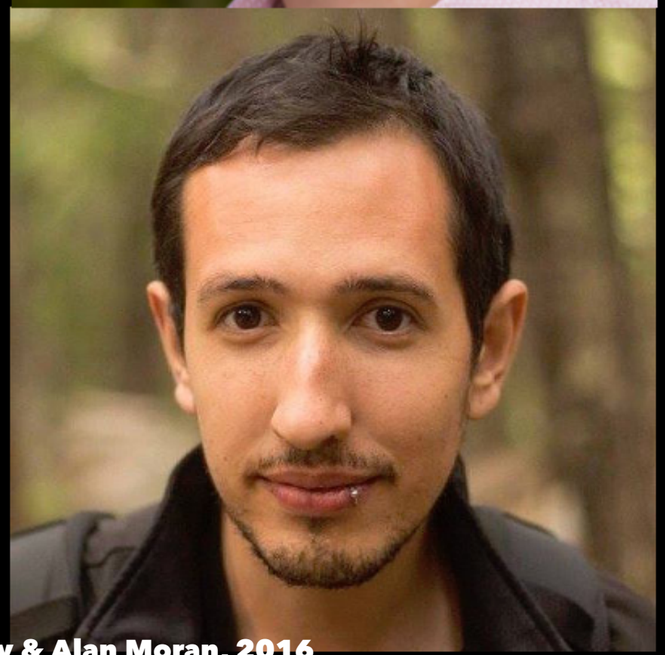
CompoZed Labs

Untangling Platform Complexity with Concourse CI



Matt Curry
Director of Cloud Engineering
Allstate
@mattjcurry

Alan Moran
Cloud Foundry Architect
Allstate
@bonzofenix

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!

Platform as a Product

- Customer
  - Developers
- Problem
  - Too much time on non-business value added work
- Solution
  - Connected Platform
  - Minimize the operational burden of running apps

The Starting Point

- 3 full time engineers
- 3 Cloud Foundry Foundations
- Several tiles (Mobile push, mySQL, Redis)
- No production environment
- Manual upgrades of the platform

Principles behind the Agile Manifesto

We follow these principles:

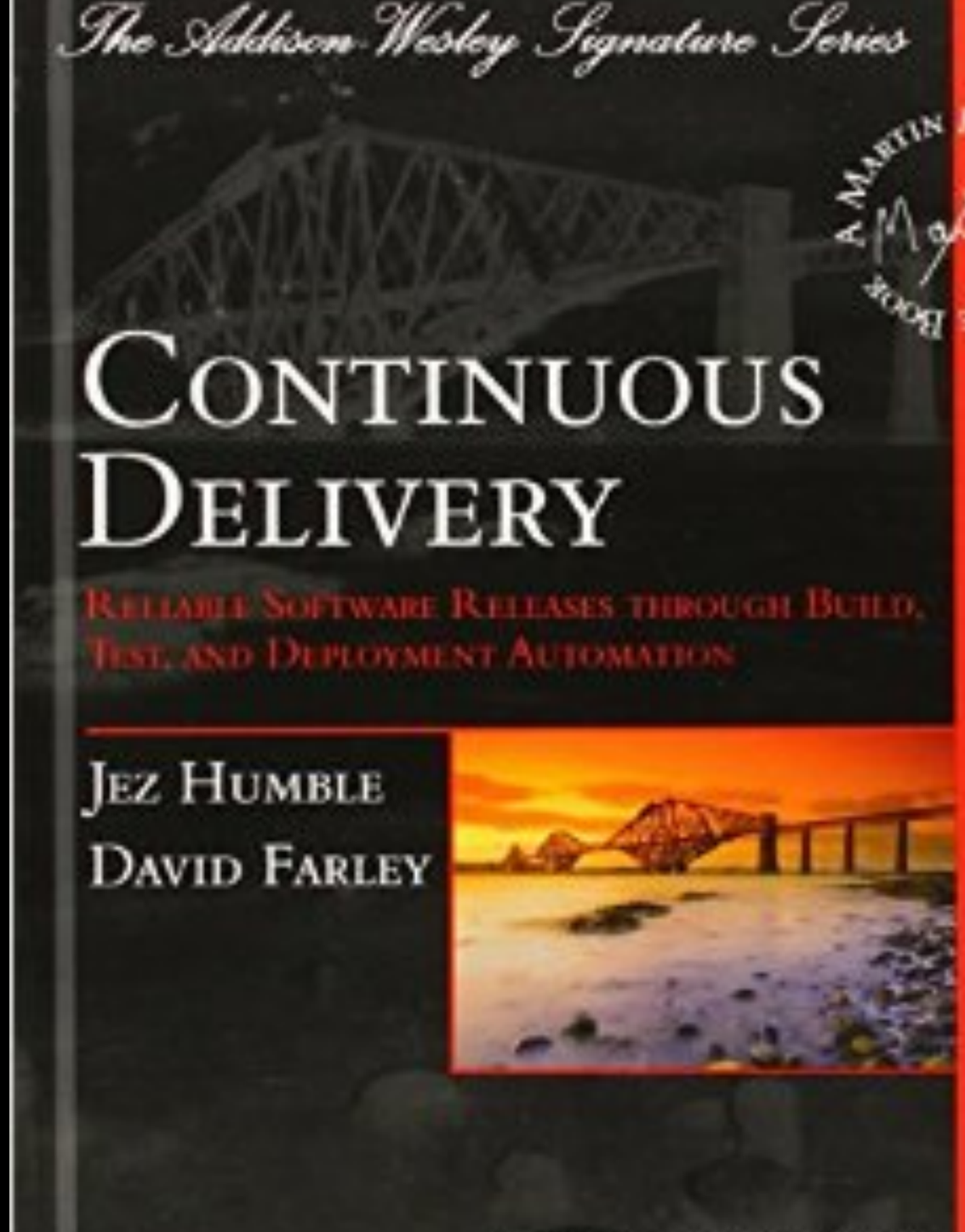
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Our Initial Goals

- Reduce the operational burden of running and scaling the platform.
  - Ensure that our CF environments look consistent
  - Automate testing of platform products
- Have a reliable way to recover in case of failure
  - Ensure environments can be reproduced from source control
- Automate Everything
  - No more #@$*^$ GUI's.

Releasing software should be easy. It should be easy because you have tested every single part of the release process hundreds of times before. It should be as simple as pressing a button. The repeatability and reliability derive from two principles: automate almost everything, and keep everything you need to build, deploy, test, and release your application in version control.

– David Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation



Pivotal Cloud Foundry Operations Manager

BOSH

The outer shell for cloud software

Pivotal Cloud Foundry Elastic Runtime

Push Notification for PCF

NFS

NETWORK FILE SYSTEM

Pivotal RabbitMQ

CloudBees Jenkins Platform for PCF

concourse

Pivotal Cloud Foundry JMX Bridge (Ops Metrics)

Pivotal Cloud Foundry Elastic Runtime

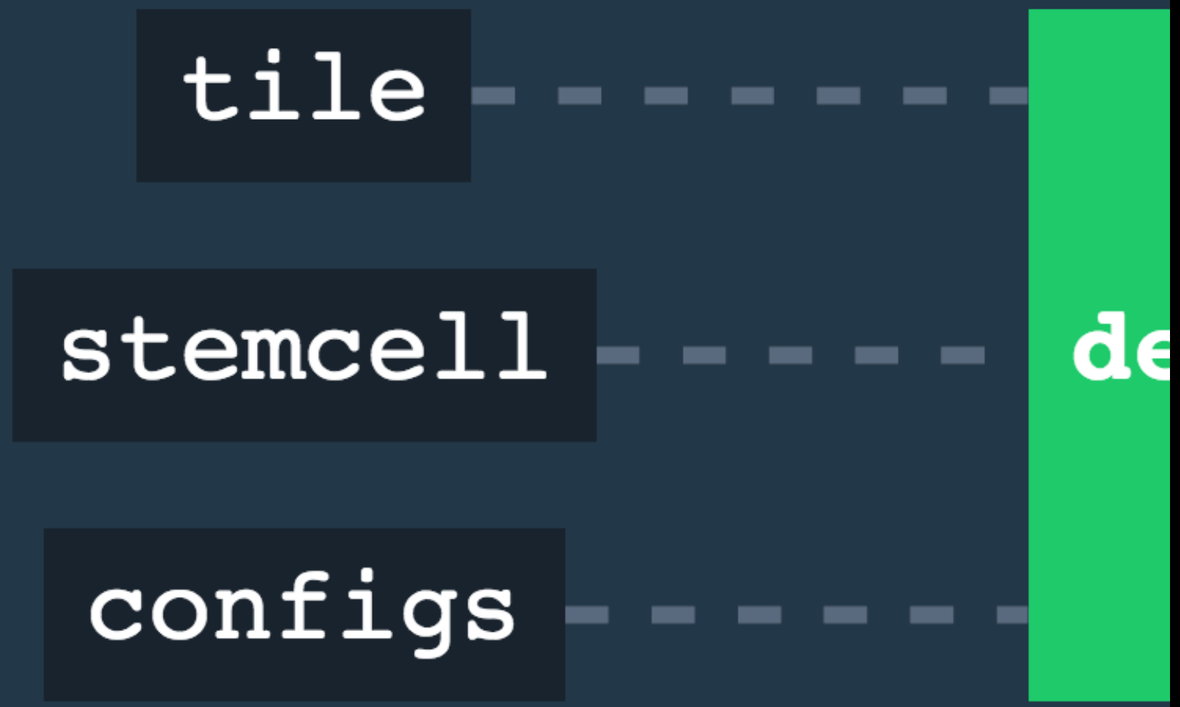
elastic + logstash + kibana

Redis for PCF

Pivotal Cloud Foundry Metrics

docker

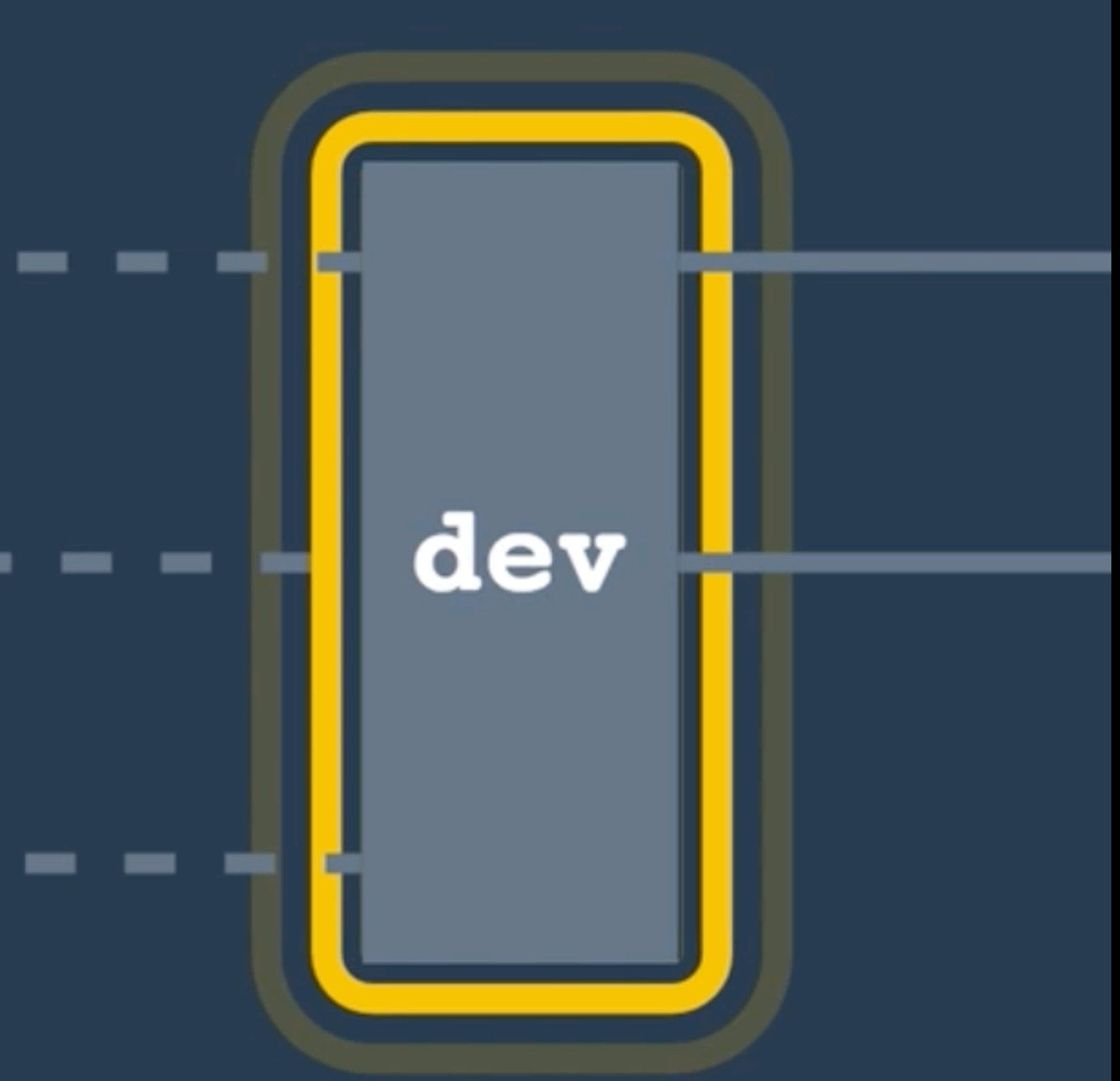
Ops Manager Director for vmware vSphere®



Concourse resources

any entity that can be checked for new versions, pulled down at a specific version, and/or pushed up to idempotently create new versions

– concourse.ci



Concourse jobs

Some actions to perform when dependent resources change (or when manually triggered)
– concourse.ci

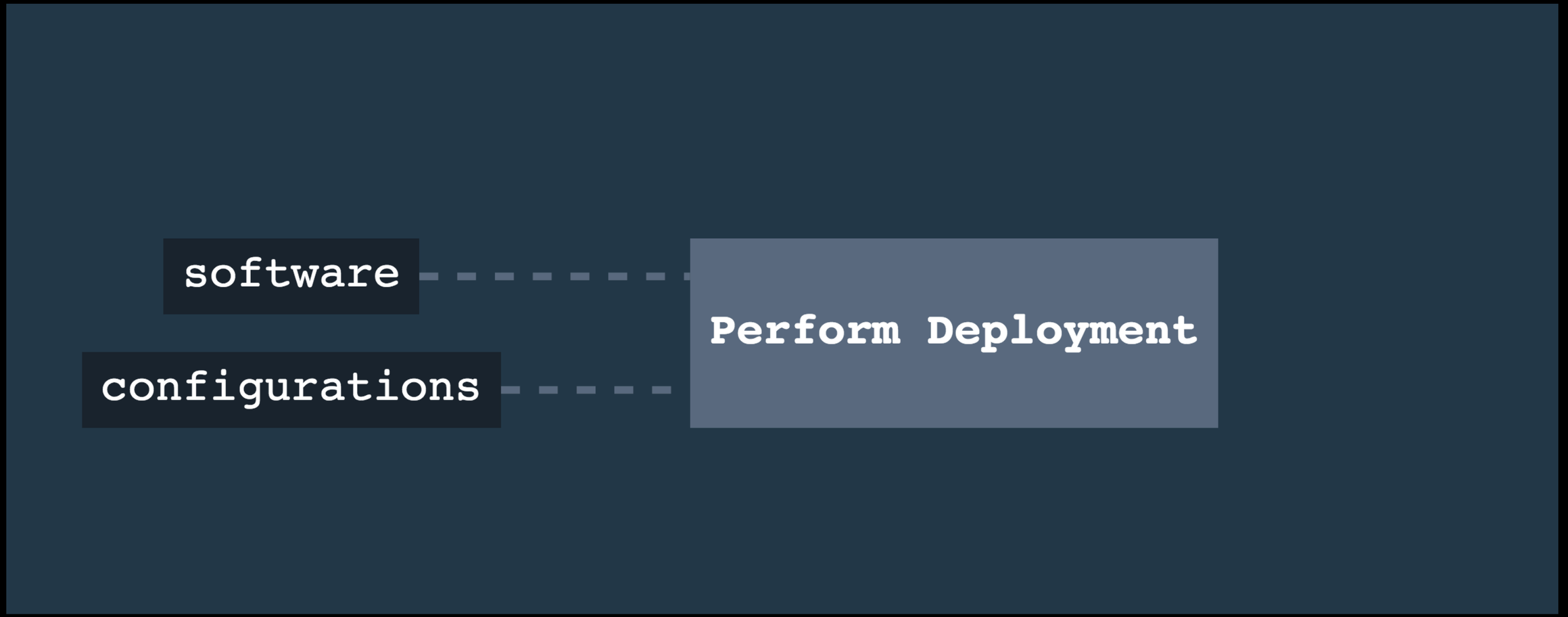
```
19
20 - name: prod
21   serial: true
22   public: true
23   plan:
24   - get: configs
25   - get: stemcell
26     passed: [ dev ]
27   - get: tile
28     passed: [ dev ]
29   - task: task1
30     config:
31       platform: linux
32       image_resource:
33         type: docker-image
34         source: {repository: busybox}
35       run:
36         path: sleep
37         args: [10]
38   - task: task2
39     config:
40       platform: linux
41       image_resource:
42         type: docker-image
43         source: {repository: busybox}
44       run:
45         path: echo
46         args: [ hello concourse ]
47
48
```

Concourse tasks

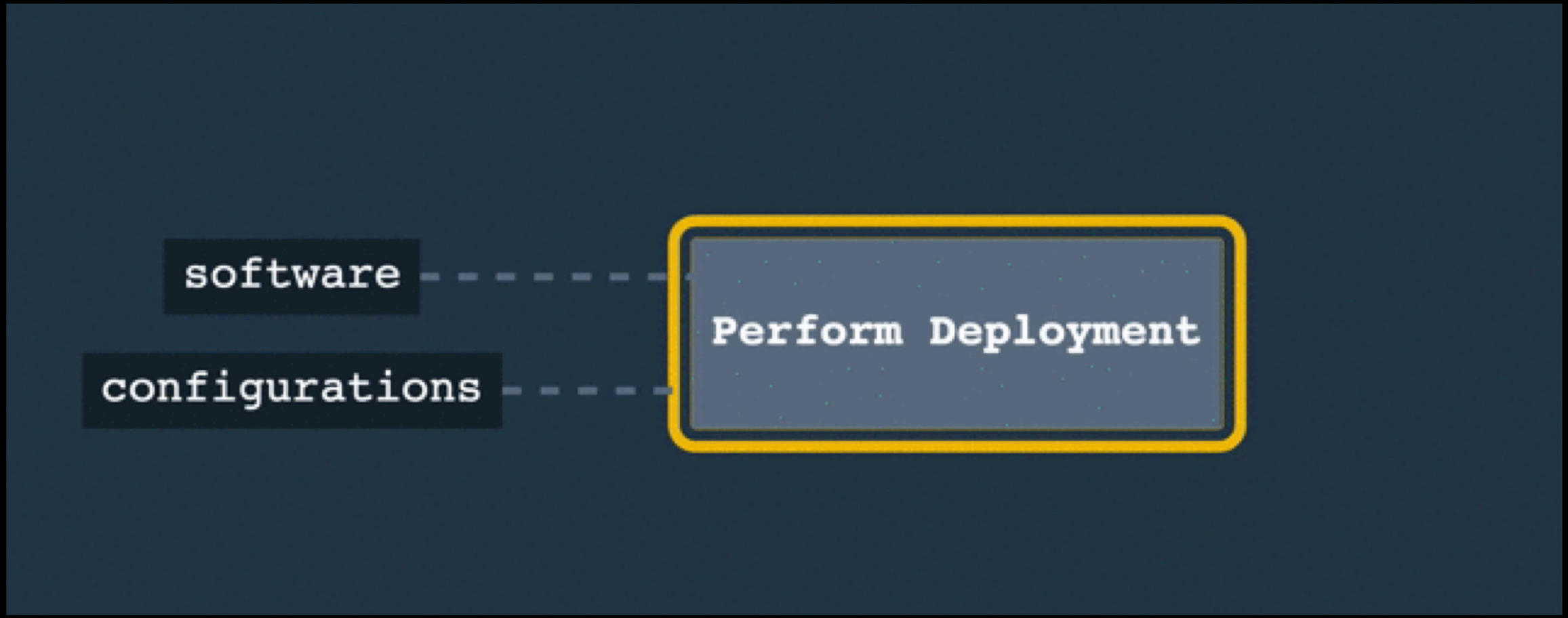
the execution of a script in an isolated environment with dependent resources available to it

– concourse.ci

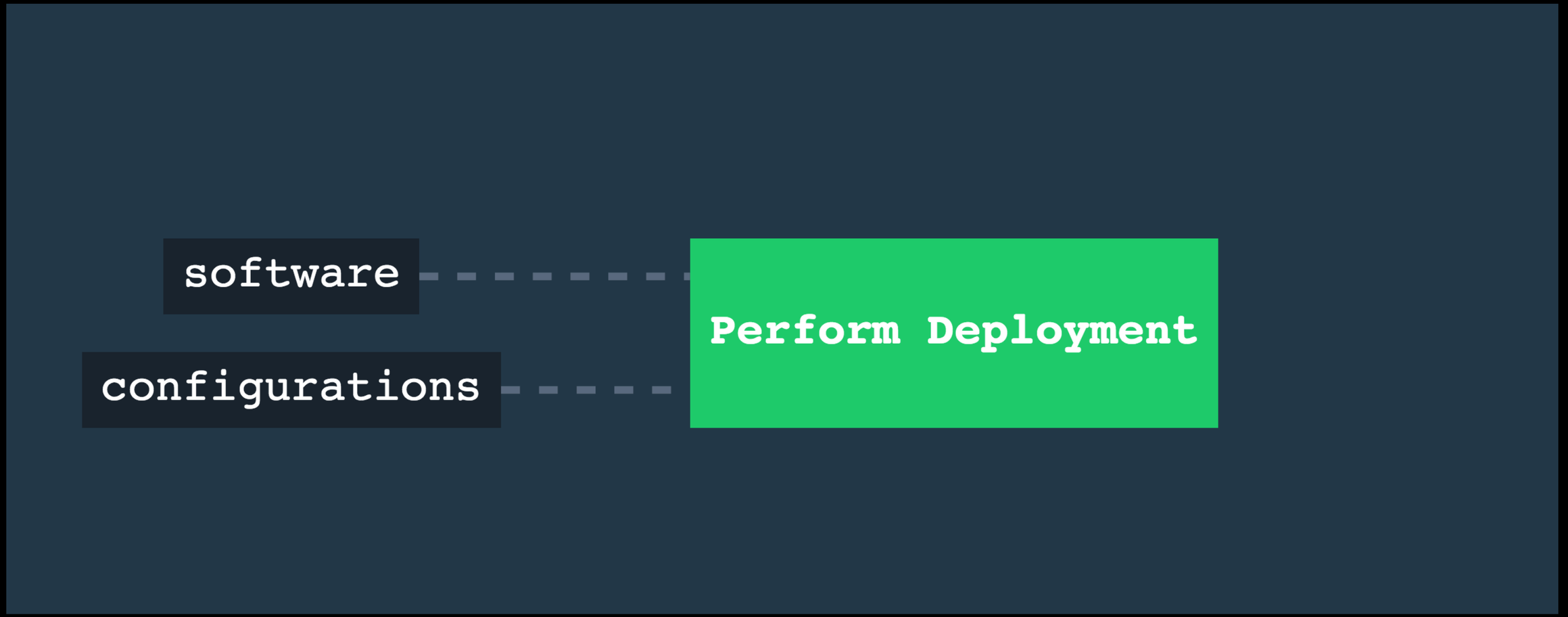
Basic pipeline pattern



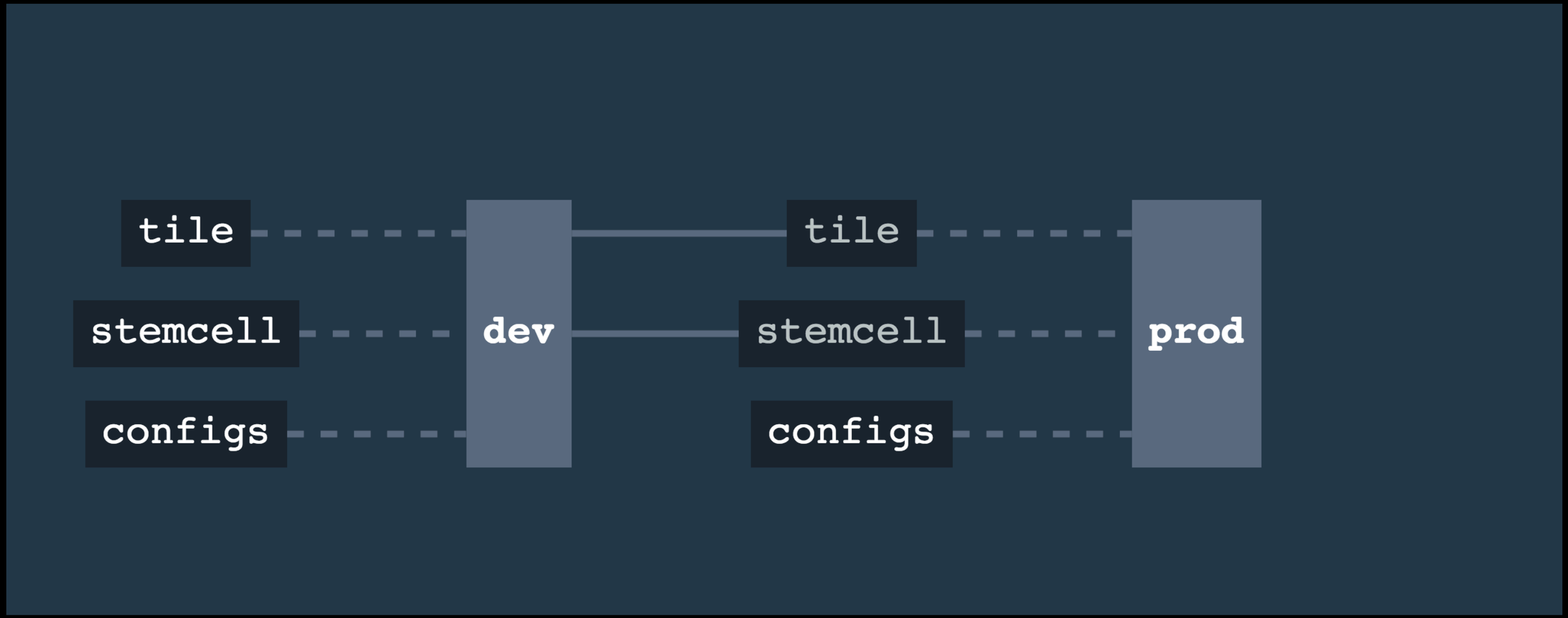
Basic pipeline pattern



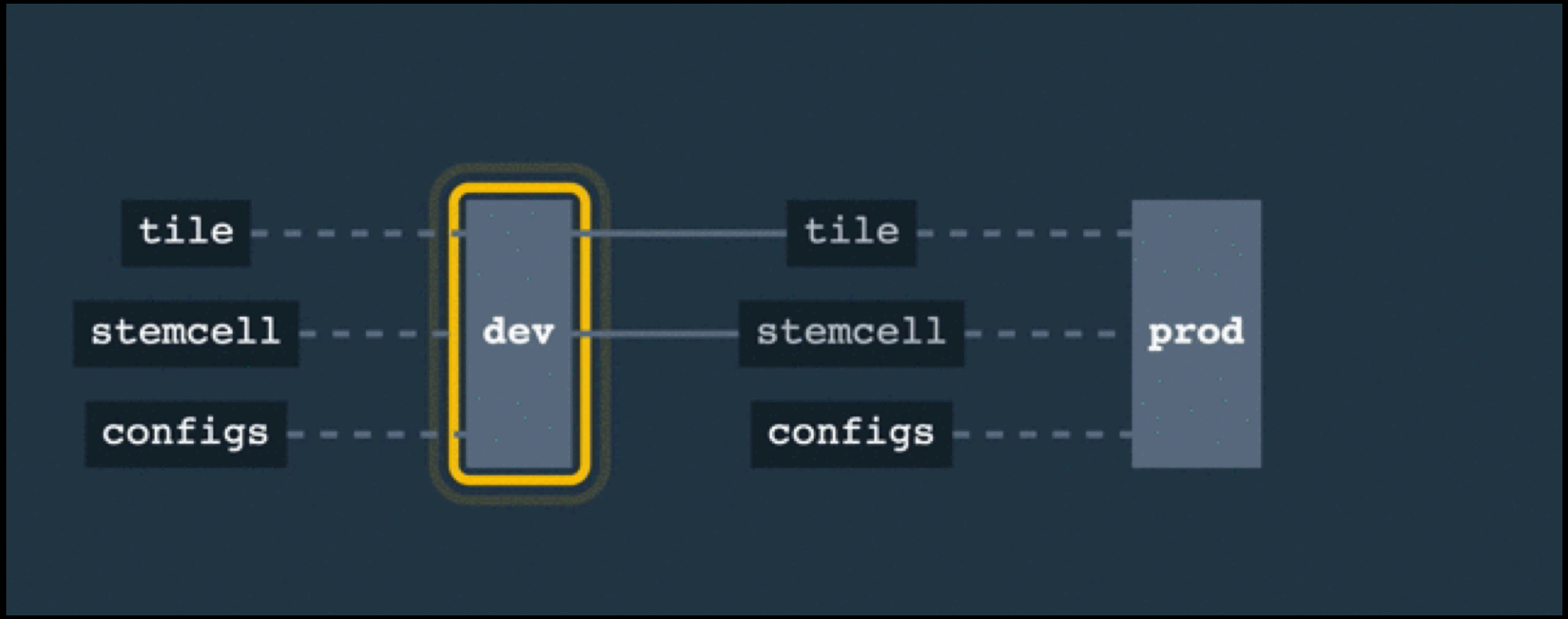
Basic pipeline pattern



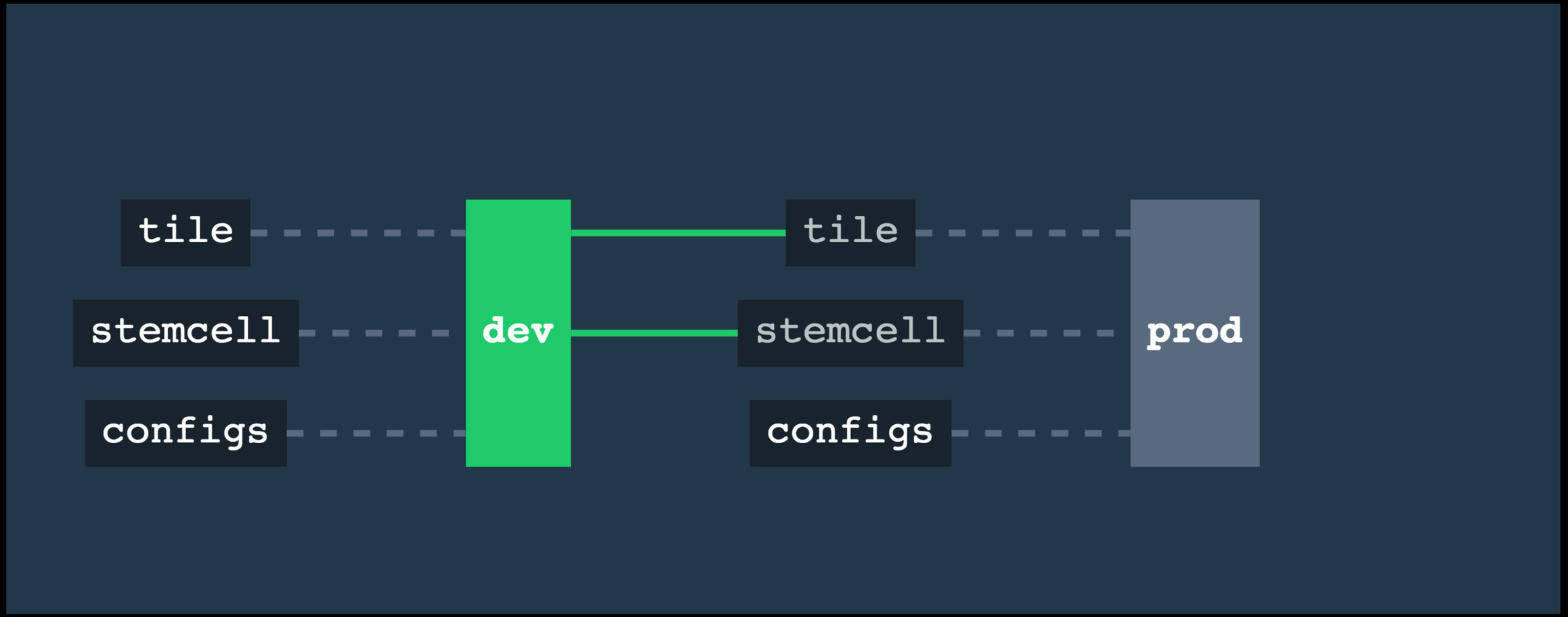
Multi-environment pipeline



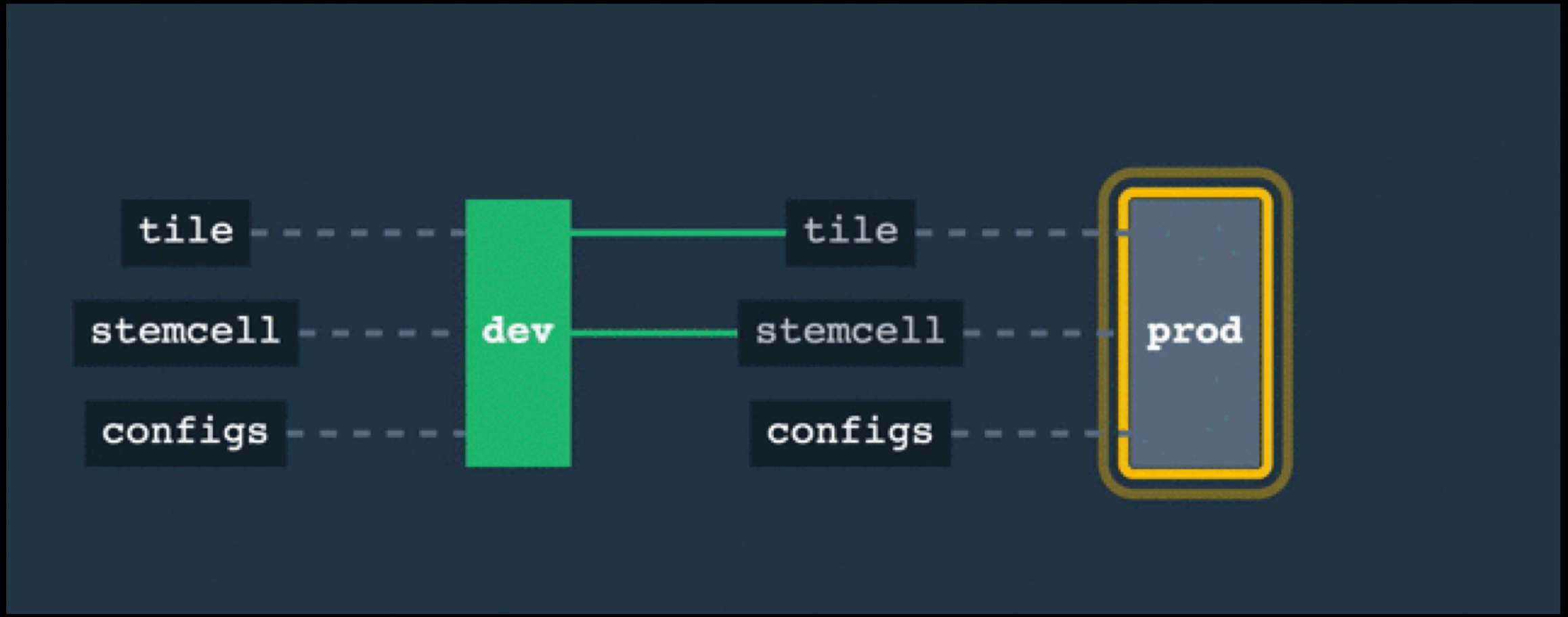
Multi-environment pipeline



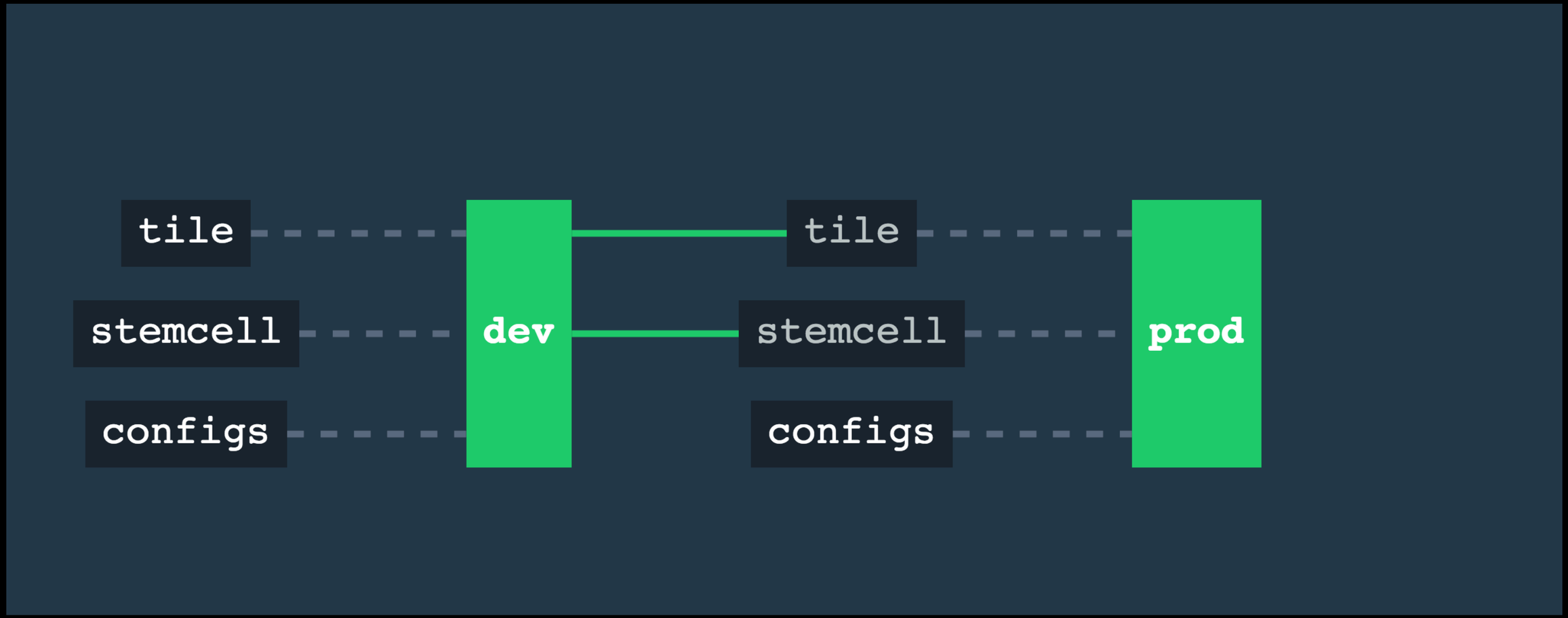
Multi-environment pipeline



Multi-environment pipeline



Multi-environment pipeline



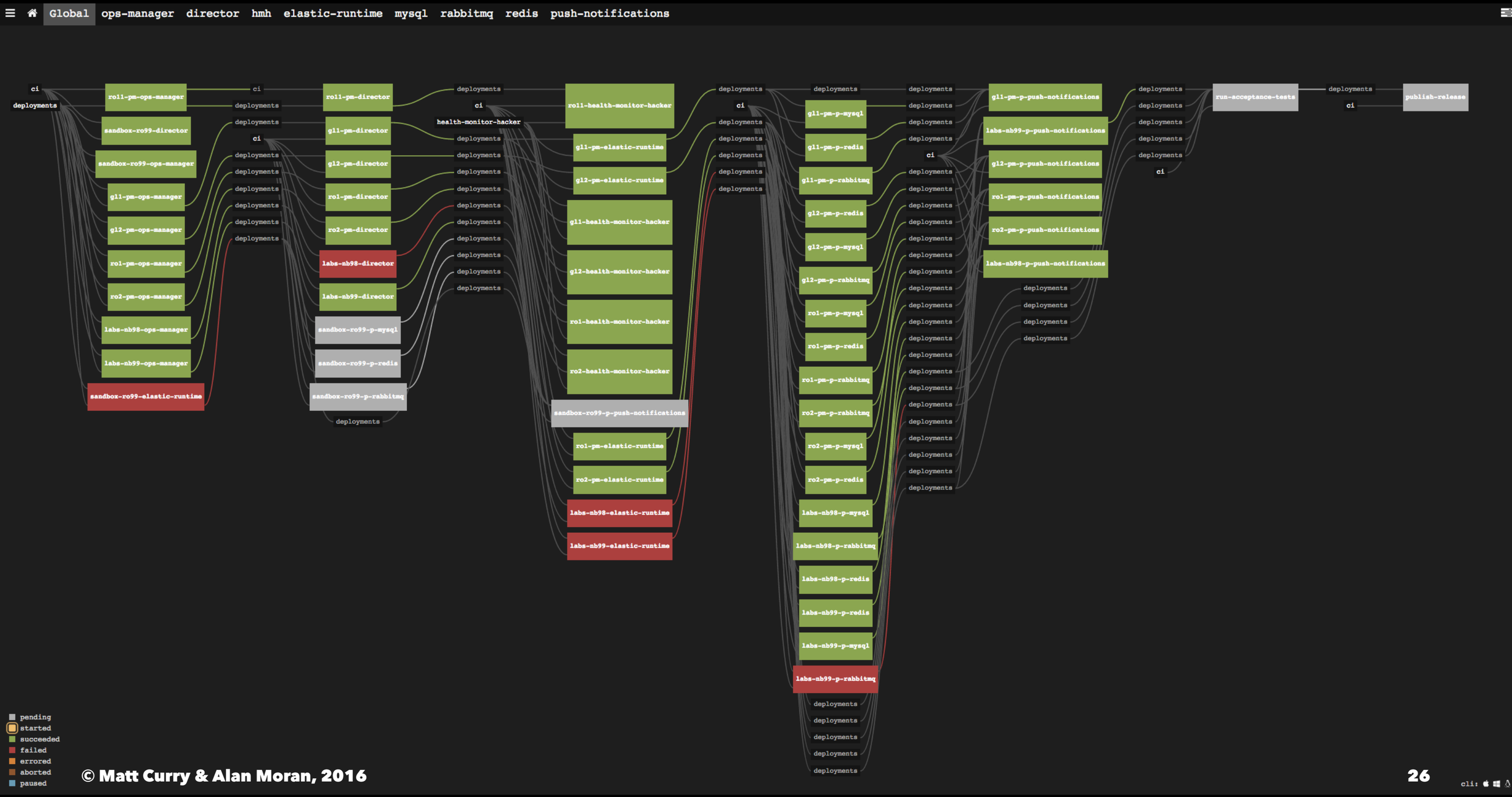
The Journey

First Iteration

- Concourse tasks instead of resources to manage:
  - Tiles
  - Ops Manager
  - Stemcells
- Vendor products dependencies into internal artifact repository
- Let Concourse git resource manage configurations
- Deploy ops manager appliance with Concourse Bash task
- Deploy PCF tiles though Concourse Bash tasks using experimental ops manager API

Benefits

- Codified tile dependencies
- Each environment was perfectly consistent
- New environment provisioning went from 1 week to 18 hours
- Engineering time for a deployment went from 40 hours to 4 hours



Opportunities

- Concourse pipeline definition file was large, complex and repetitive (2-3K Lines)
- Managing multiple concurrent versions of Ops Manager was difficult with Bash
- No official API for Ops Manager
- Vendoring dependencies

Second Iteration

- Generate single pipelines per product
  - Ops Manager
  - Tiles: Elastic Runtime, MySQL, Redis
- Manage, release and deploy our own fork of Concourse
  - Added proxy support
- Build command line application for Ops Manager
  - Integration Tests for Ops Manager API breaking changes

Benefits

- Pipeline definitions were granular and responsible for a single product
- Easier management of multiple versions of Ops Manager API
- Can pull dependencies directly from Pivotal Network through Concourse tasks

Opportunities

- Concourse pipeline definitions were repetitive
  - Spending lot of time writing boilerplate manifests to create pipelines
- We had forked Concourse, so we had to merge changes

Third Iteration

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

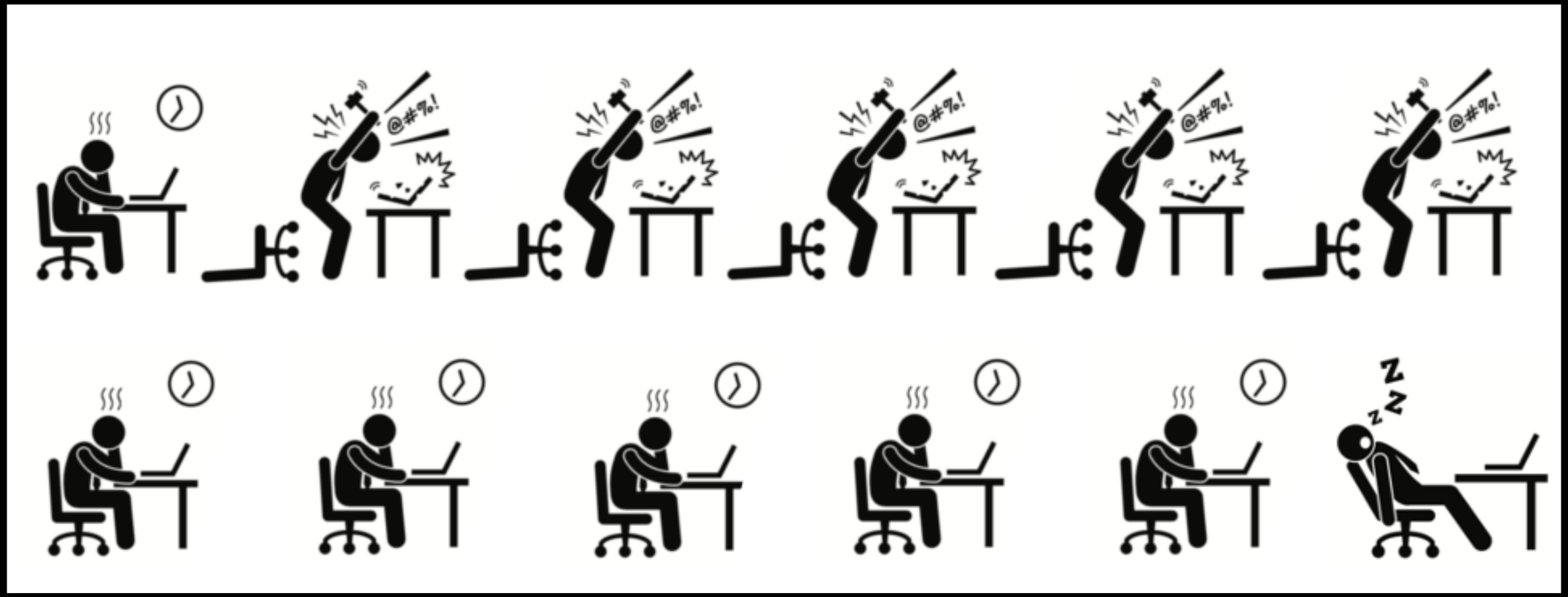
Benefits

- Reduced pipeline definitions from 400 line YAML to a template + config file
- Built in validation of generated pipelines
- Reduced human error by reducing repetition in pipeline definitions
- 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 new deployment went from 3 hours to under 2 hours

What we Manage

- 6 Engineers
- 2 Datacenters
- 10 Foundations
- 11 Products
- 15 Pipelines
- 150 Automated consistent deployments

From / To



Unleash the Freedom

Activity	Manual	Interation 1	Interation 2	Iteration 3
New Environment Prep Time	60 Hrs	4 Hrs	3 Hrs	.5 Hrs
Stemcell Upgrade	10 Hours	2 Hrs	1.5 Hrs	0 Hrs
Tile Upgrade	8 hours	2 Hrs	.5 Hrs	0 Hrs
Line of Config per Product	0 (in GUI)	500	500	40
Innovation Capacity	5%	50%	75%	80%

Lessons Learned

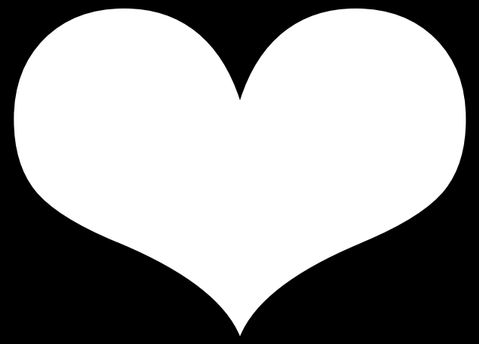
- Iteration and learning is key and part of the journey
- Start with triggers turned off
- Don't be afraid to code
- Focus on removing error from the system
- Smaller pipelines with a single responsibility reduce complexity
  - Faster feedback
  - Easier to reason about

Don't be afraid of change, embrace it and get good at it

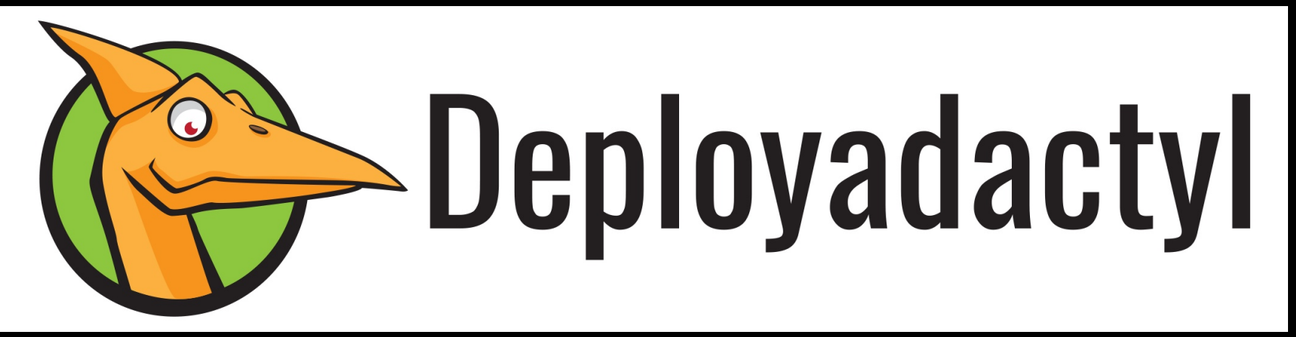
How to get started

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

We



Concourse



Announcing Deployadactly

An open source for deploying to multiple CF foundations

Deployadactyl is a Go library for deploying applications to multiple Cloud Foundry instances. Deployadactyl utilizes blue green deployments and if it's unable to push your application it will rollback to the previous version. Deployadactyl utilizes Gochannels for concurrent deployments across the multiple Cloud Foundry instances.

Star us on Github

Q&A