How to use Concourse CI to deliver BOSH releases

nominally

Amit Kumar Gupta, Pivotal
“nominally”, because...

caveat auditor

• BOSH Day attendees may actually want to see slides full of YAML

• if this is you, you will be disappointed
about me

• **product manager @ Pivotal**
  – cf-release
  – etcd-release
  – consul-release
  – bosh-bootloader

• **engineer @ Pivotal (previously)**
  – Diego
  – Ops Manager
  – BOSH
let’s talk about…

• why machines & software?
• complexity vs simplicity
• testable, discoverable contracts
• complexity in the BOSH ecosystem
• recommended practices around BOSH and Concourse
why machines & software?

automate physical labour

• speed
• consistency
• scale
• reproducibility
why machines & software?

**automate cognitive labour**

- speed
- consistency
- scale
- reproducibility
complexity vs simplicity

Complex
cognitive tasks
are hard to do well
complexity vs simplicity

Complex
many pieces, unclear interactions, requiring implicit knowledge

cognitive tasks
  e.g. do taxes, deploy software

are hard to do well
  fast, consistent, at scale, reproducible
testable, discoverable contracts

• provide some *interactions* /solutions to consumers, and no more

• ask consumers for *input* required for those interactions, and no more

• provide consumers with *output* /effects promised by solutions, and no more

• make these *contracts* testable and discoverable
testable, discoverable contracts

These problems of design apply to:

• user-facing products and services
• distributed system components
• processes
• modules, objects, and functions
complexity & BOSH ecosystem

These problems of design apply to:

- BOSH deployments
- BOSH instance groups
- BOSH releases (and their jobs)
- business logic source code
- and also the test pipelines delivering these things
recommended practices

- no snowflake environments
- separate per-environment config resources
recommended practices

• test your task scripts
• build your own task images
recommended practices

- manage BOSH jobs with real programs
- unit test & system test that logic
- keep release small enough to be used with hand-editable example manifests
- validate those manifests in CI
**recommended practices**

- cut final releases in CI
  - manually or automatically
- no snowflake buckets
recommended practices

- separate pipeline config from params
  - fly set-pipeline -c config.yml -l params.yml
- strive to make all jobs public
recommended practices

• separate create-release, deploy, and test
• use BOSH-specific Concourse resources
  – bosh-io-\{stemcell\,release\}
  – bosh-deployment
resources

• release repos
  – https://github.com/cloudfoundry/cf-release
  – https://github.com/cloudfoundry-incubator/consul-release
  – https://github.com/cloudfoundry-incubator/etcd-release

• pipelines
  – https://runtime.ci.cf-app.com
  – https://mega.ci.cf-app.com

• pipeline configs
  – https://github.com/cloudfoundry/runtime-ci
  – https://github.com/cloudfoundry/mega-ci
contact

- agupta@pivotal.io
- @amitkgupta84
- cf-dev & cf-bosh mailing lists

- http://akgupta.ca
- https://github.com/Amit-PivotalLabs/

- slack