From Fix to Final Release

Continuous Integration and Testing for Diego

Jen Spinney (HPE) & Andrew Edgar (IBM)
Your Presenters

Jen Spinney - Hewlett Packard Enterprise
Seattle, WA
jen.spinney@hpe.com

Andrew Edgar - IBM
Regina, Saskatchewan
aedgar@ca.ibm.com
@AndrewEdgar67
Diego Team

- Multi-company, pair-programming
- Partially remote
- Responsible for the new runtime of Cloud Foundry, Diego
- Andrew and Jen both “Dojo”d in June 2015
  - Been on Diego ever since
What this talk is about

- Levels of testing
- Continuous integration
- Release versioning
What this talk is *not* about

- Implementation and features of Diego
- Concourse-specific configuration
So what is this?
The Life of a Fix

Step 1: Pick up the bug or feature
The Life of a Fix

Step 2: Pair on the fix

Test locally
The Life of a Fix

Step 3: Push to develop branch
The Life of a Fix

Step 4: Let pipeline test and publish
Test Driven Development

- All code is written using Test Driven Development (TDD).
  - Write a failing test first, then write code until the test goes green.
- We run unit tests constantly while developing
  - Requires fast unit tests
- TDD is used on all Cloud Foundry projects.
Local Dev & Test Cycle

1. Write unit tests to validate implementation in isolation.
2. Develop the code to make the tests pass.
3. Write component, cross component, and acceptance tests if required.
4. Ensure the integration tests pass.
5. Run ALL the unit and component tests.
6. Run cross-component integration tests.
7. Run acceptance tests against a locally deployed bosh-lite env.
8. git push. Now it gets interesting!
Acceptance

- During the CI Pipeline the code is deployed into an acceptance environment.

- Once the code reaches this environment, the Product Manager (PM) determines acceptance.
  - Acceptance criteria is written into the story up front.

- The PM can validate story completion by experimenting with that environment.

- If needed, the PM will also add a test to the PM acceptance test suite.
Run all the unit, component, and cross component integration tests (Inigo)
Units

- The bedrock of our development.
- One unit test file per .go file and a set of component integration tests.
- Component level integration test builds and runs the exe being tested.
  - Usually just confirms “happy path” base functionality.
  - Test that smaller pieces are wired up correct in main.go
Inigo

- Cross component integration level tests.
- Test features that cut across multiple components.
- This suite of tests run against all diego components spun up in a single container with non conflicting ports.
- Because these components will be running on various ports, we run them in a container to have a sandboxed environment.
  - We use concourse to run this suite in a container.
  - The suite can be easily run on a developer’s local machine or in a remote concourse instance.
Create BOSH Releases

Before more tests are run we need to create both a CF and Diego releases for deployment.
Wait for Bosh Environment.

Wait until the acceptance environment is available for deployment.
Deploy the updated CF and Diego releases to the acceptance environment.
Smoke Tests

A standard set of simple tests to ensure the minimal functionality (cf push).
Acceptance Tests

Run CF Acceptance Test Suite (CATS).
Run Windows Acceptance Test Suite (WATS).
Acceptance Tests

- These tests are written from the perspective of an end user using CF CLI.

- Testing aspects of the system that can be verified solely with the CF APIs.

- CATS can be run with either a DEA or Diego backend. Obviously we use Diego!

- There are some tests that only work with Diego (i.e. SSH testing) and also some that are DEA specific and so these can be skipped during a test against Diego.
Diego Upgrade Stability Tests are run in parallel with the main acceptance tests.
DUSTS

- Diego Upgrade Stability Tests.

- Ensures upgradability from the initial Diego release to the current version.

- The tests run in CI and entail deploying the base version of Diego and performing a piecewise upgrade to the version being tested.

- There is a continuous polling of an application to ensure routability throughout the process.
Deliver

Once all standard tests have passed the deliver phase auto-delivers all stories for acceptance by the PM.
The PM can mark a build to be shipped with the Shiplt manual task.
A draft release is created in GitHub which the PM can edit before finalizing.
All the git work has been done on the develop branch. The final step is to merge the release on Master back to Develop.
Vizzini in our CI can be found in the “periodic” set of tests and runs every 30 minutes against Ketchup.
Vizzini

- PM acceptance tests.

- A suite of tests that interact with Diego through the Diego API.

- In CI we run those against the acceptance environment.

- A script is available to run this against a bosh-lite env.
The Benchmark Suite runs against a delivered successful build to ensure performance has not regressed. It deploys into a separate environment.
Benchmarks

- We isolated the known performance bottleneck in our system: communication to and from the BBS
  
  - NSYNC bulker, route-emitter bulk loop, lrp convergence gathering, rep bulk fetching, bbs event watching
  
- These tests record performance benchmarks for a test suite running high load directly against this known bottleneck.

- Fill the backend with data and run simulated extreme load.

- Having this suite drove the move towards a SQL backend instead of ETCD

  - Increasing the load saw unacceptable performance degradation and even data loss.
How much is too much?

- 51 total jobs in our pipeline
- 6 bosh environments
- Flakes happen
  - Some jobs rely on external services
    - AWS
    - github.com
    - Docker Hub
  - Our build environment can be overloaded
  - Etc.
- Build czar is now a full time job
- Some jobs take a long time
  - (1.5 hours for DUSTs)
Conclusions

- A top priority to ship known good builds of Diego.
- Testing and continuous integration is a balancing act between what is quick and what is beneficial.
- We hope to have shown that the Diego team dedicates significant resources to ensuring the quality of our final releases.