2017 JFrog swampUP

DevOps for Mere Mortals

How do smaller organizations run fast & smart?

Copyright © 2017 JFrog. All Rights Reserved

Agenda

- Introductions
- Devops at Yahoo!
- Devops at SADA
- Devops for you

Copyright © 2017 JFrog. All Rights Reserved

Introduction

Allen Reese, Yahoo Inc.

YAHOO!

- JNI engineer at Yahoo!
- Moved Yahoo from
  - RHEL4 -> RHEL6 -> RHEL7
- Builds everyone else's packages.

Jack Weaver, SADA Systems

SADA systems

- Tech Lead at SADA
- Worked with Allen at Yahoo prior to SADA.
- Took SADA from "zero" to: devops, automated, tested, CI/CD, etc.

Copyright © 2017 JFrog. All Rights Reserved

Links

JFrog 2017 swampUP

- All code, demos, documented details, are:
  - Deck: (Download to follow along, more readable).
    - http://bit.ly/2qMobqx
  - Github Repo:
  - https://github.com/SwampUpMereMortals
    - All code examples, as well as directions and setup notes.
  - Artifactory: https://meremortal.jfrog.io/meremortal
  - Travis: https://travis-ci.org/SwampUpMereMortals

Copyright © 2017 JFrog. All Rights Reserved

“DevOps in the Small”: SADA Systems

- Company Size about 150 employees.
  - About 25 engineers.
  - Very successful company, traditionally consulting/IT services.
  - First production application “Atom”.

- Originally **zero** DevOps.
  - Engineers literally deploying to production from their personal laptop:
    - “What’s a runbook?”
    - No testing. No automation.
  - Outages went unnoticed…. Until the customer called us. No monitoring.
  - “Testing” was a human being clicking on the screen endlessly, day after day, week after week.
    - It’s amazing this person is still alive today. Medical science can’t explain it.
- Things are different now! Significant maturity in 12 months. *Serious* levels of automation.

Copyright © 2017 JFrog. All Rights Reserved

“DevOps in the Small”: SADA Systems

First principle: *software development is an economic activity*.

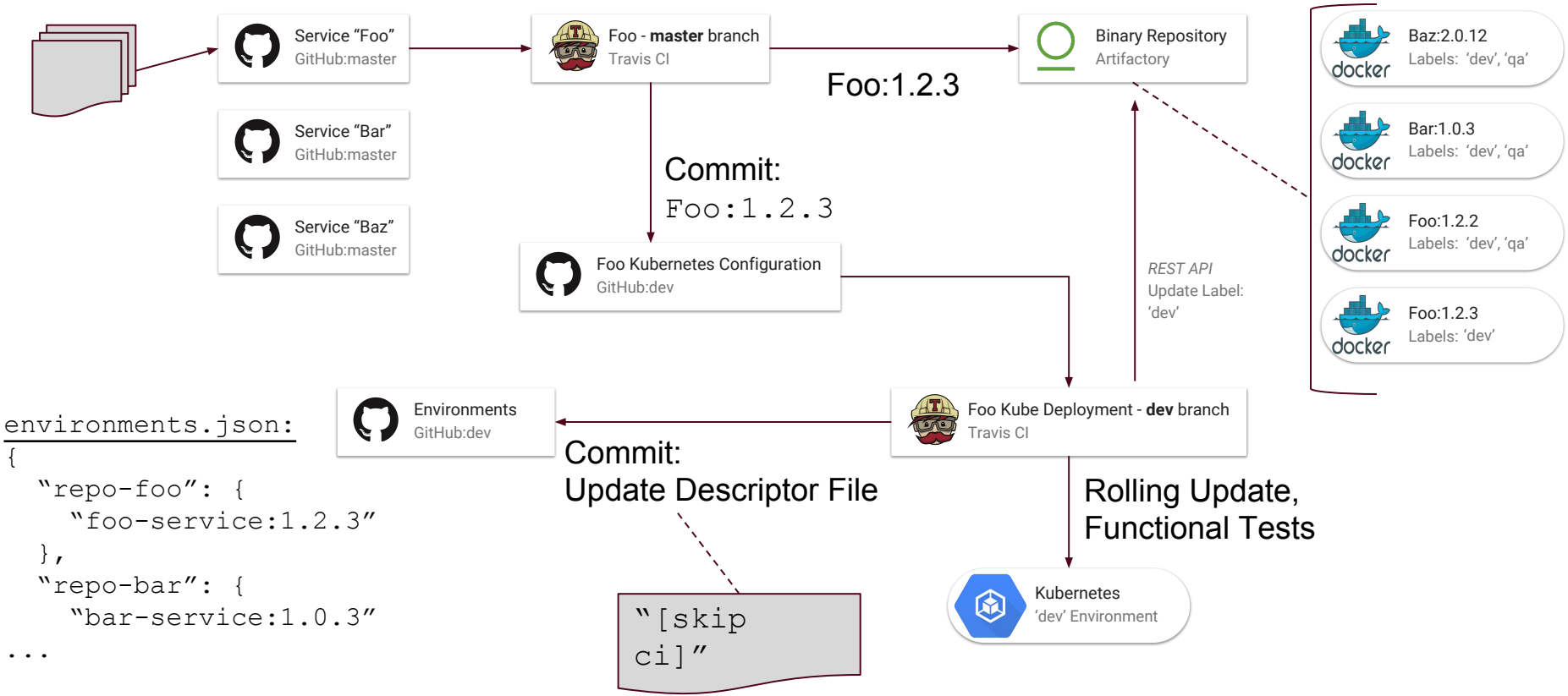
- Time spent on DevOps, Automation, CI/CD, etc, is an *investment* and thus must have an ROI.
- ***You therefore only invest in the ROIs that are greatest at your point in time***.

With a team size of 6 engineers, we started small and progressed outwards:

1. Setup Travis for all repos. Built, Unit Tested, Versioned & Published.
2. Infrastructure as Code from the start: code reviewed infrastructure changes, dedicated repos.
3. Started to want travis to deploy packages as well, setup CI commits.
4. Then wanted functional & integration tests, after deploy is done.
5. Finally, needed a way to coordinate deployment of multiple artifacts.

Copyright © 2017 JFrog. All Rights Reserved

From Commit to Deployment



Copyright © 2017 JFrog. All Rights Reserved |

What does this do? Why?

JFrog 2017 swampUP

- Deploy a single service, or an entire system (dev → qa, qa → prod).
- `environments.json` shows us exactly what is where.
- Single canonical version string, and everything is tagged.
- Different points in the pipeline correspond to different layers of testing: Unit Testing, Functional Testing, Integration Testing.
- Different points in the pipeline allow for interesting "articulation" points.
  - Need to know what was release into prod, from qa? Automate a CHANGELOG.

Copyright © 2017 JFrog. All Rights Reserved

What the Big players do (Yahoo!)

JFrog 2017 swampUP

- Large devtools teams to set everything up
  - 80+ people just to maintain build farms and tools to build it.
- Own the data centers
  - No clouds, no cost controls, no restrictions
  - Data is kept forever, until it causes a problem
- Control every aspect of what is run and deployed
  - Political and tool enforcement results in similar looking builds.
  - It's easier to build everything the way the other engineers understand
- Custom monitoring and CI tools built around Jenkins
  - Some of the choices around how to use Jenkins and sizing has resulted in poor scalability.
  - One box fits it all mentality.
- Internal SaaS platforms to push code through pipelines to production

Copyright © 2017 JFrog. All Rights Reserved

What the Big players do (Yahoo!)

JFrog 2017 swampUP

- Product teams make their own choices about builds.
  - On Jenkins every team has their own way of doing things.
  - With shared “build platforms” builds look similar.
    - Except when they don’t.
- Shared infrastructure
  - Shared outages.
  - Shared security holes
  - All source is effectively public
- Resource constraints to prevent monopolization of executors
  - Tried using docker, but cgroups broke the JDK.
  - JDK 8u131 has an option to behave better under docker
    - But you have to enable special flags to do it.
    - Which means modifying every build to know about the flags.
    - And the jvms that build tools launch

Copyright © 2017 JFrog. All Rights Reserved

What can the little players do? (YOU!)

JFrog 2017 swampUP

- Rely on cloud providers for infrastructure
  - No hardware to buy/maintain, no datacenters to put the hardware in
  - Fine grained cost controls.

- Use SaaS solutions to meet your needs.
  - Bintray, Artifactory SaaS, Travis CI/Circle/Drone/Semaphore/etc, GCP/Google Container Engine, StackDriver, Codacy
  - Sometimes buggy, or limited in ways that are hard to work around
  - Their downtime means you are down. It's a risk.

Copyright © 2017 JFrog. All Rights Reserved

What can the little players do? (YOU!)

JFrog 2017 swampUP

- Limit the amount of engineering time required to push code to production
  - Leverage travis or circle.
  - Can require lots of glue code. Everything is an ROI, measure it before you put it in a sprint.
  - Avoid unknown outages and large manual QA effort.
    - Smaller teams necessitate the need for automation.

- Your Advantage: new development, no legacy debt. Can design software and architecture using modern development practices.
  - Lower ramp up times for new engineers as they may already know these.

Copyright © 2017 JFrog. All Rights Reserved

What can the little players do? (YOU!)

JFrog 2017 swampUP

- Understand what tools you are using and why
  - If you don't know why you are using a tool, or how others use it, you might be using the wrong tool.
- Understand how the parts fit together.
  - Sometimes builds get so complicated, you can only guess if it works by committing the code.
  - This prevents debugging, and frustrates your developers.
  - Complicated build systems break in interesting ways. The more the people using the build can read and understand about how things fit together, the faster code goes to production.
- Understand how the solution fits your situation
  - For a devops system to work correctly, you need to understand how the system influences your end goal of increased reliability and decreased downtime.

Copyright © 2017 JFrog. All Rights Reserved

Why change what you are doing?

JFrog 2017 swampUP

- The root cause of a large number of incidents was found to be human error.
- People accidently mis-key or skip steps and tests in the runbooks.
- Each engineer deploys code differently.
- Allows traceback of what code is running in production.
  - A site was running 39 different jars in a 40 node cluster for a single service.
  - 38 of those jars were built on developer's boxes from code that was never committed

Copyright © 2017 JFrog. All Rights Reserved

LIVE DEMO

Let’s hope this works.

Artifactory SaaS on GCP

JFrog 2017 swampUP

- Setup Artifactory SaaS in GCP
  - We'll be deploying our software to artifactory.
  - http://meremortal.jfrog.io/meremortal
- Free 30 day trial
- *Protip*: use same GCP region as GKE.
- Gradle/maven/docker work seamlessly with artifactory SaaS or on-prem.
  - https://github.com/JFrogDev/project-examples

JFrog Artifactory

Copyright © 2017 JFrog. All Rights Reserved

Travis CI Setup

- Setup TravisCI to deploy to Artifactory
  - We'll use Travis as our CI provider.
- Installs common-ci (debian package)
- Configure for Service Accounts (GCP) & github GH_TOKEN (for auth/commits back)
- Room for optimization (caching).

Multiple Repos
Travis CI

https://travis-ci.org/SwampUpMereMortals

NOTE: You can easily use Circle, Semaphore, Drone, etc. We chose Travis randomly.

Copyright © 2017 JFrog. All Rights Reserved

Kubernetes on GCP/GKE

- Setup Kubernetes within GCP
  - We'll be deploying to the cloud as well.
- Rules of the road:
  - We separate out application code from deployment code. Those are two different concerns.
    - Application code repo & Kubernetes Pod/Deployment repo
  - Immutable Server Pattern
    - https://martinfowler.com/bliki/ImmutableServer.html
  - CI/CD does everything for us. Minimize the opportunity for fat fingers and human mistakes.

Copyright © 2017 JFrog. All Rights Reserved

The Top 4 Takeaways for Mere Mortals

What can You learn from this?

Automate: Our most active contributor

Atom CI
atom-headless-ci

http://pastebin.com/raw/NADFxbGX

Unfollow

Block or report user

@sadasystems
~/

Overview | Repositories 0 | Stars | Followers 2 | Following 0

Popular repositories

atom-headless-ci doesn't have any repositories you can view.

674 contributions in the last year

May Jun Jul Aug Sep Oct Nov Dec Jan Feb Mar Apr
Mon
Wed
Fri

Learn how we count contributions.

Less More

Contribution activity

Jump to ▾ 2017 2016

May 2017

Created 121 commits in 8 repositories

Copyright © 2017 JFrog. All Rights Reserved

Infrastructure as Code

- Immutable Server Pattern.
- Cattle, not pets.
- No snowflakes in your system.
- Don’t let data and state dictate your systems.
  - Stateful sets or pet sets in Kubernetes.

Copyright © 2017 JFrog. All Rights Reserved

Keep everything simple and similar

JFrog 2017 swampUP

- The simpler the setup is, the easier it is to describe and modify
- Keep things similar without copying too much boilerplate
- Share where it makes sense, the more you share the more you are forced to keep things simple.
- Simple things are easy to debug by sight.

Copyright © 2017 JFrog. All Rights Reserved

DevOps for ROI

JFrog 2017 swampUP

- Only invest your time in the biggest payoffs for your organization at the time.
  - Don’t build the world. Start **small**. Receive immediate returns on that time.
  - For example, in progression:
    - Infrastructure as Code (review it). Know your hardware & configurations.
    - CI Builds, Unit Test everything at build time.
    - Start versioning, packaging & publishing from CI builds.
    - Start having CI conduct deployments.
    - Start running Functional/Integration tests after deployment.
- Use Monitoring & Alerting Systems (sometimes you get this free).
- Use automation where it makes sense for you, this is not one size fits all.

Copyright © 2017 JFrog. All Rights Reserved

JFrog 2017 swampUP

Please submit feedback forms!

Questions?

Copyright © 2017 JFrog. All Rights Reserved