

GOTO CHICAGO 2017

Reining in Chaos

Colleen Stock, Uptake Engineering



colleen.stock@uptake.com



[@uptake](https://twitter.com/uptake)

A close-up, low-angle shot of a jet engine turbine. The image shows the complex, curved blades of the turbine, which are dark and metallic. The lighting is dramatic, with strong highlights and deep shadows, emphasizing the texture and geometry of the engine components. The text is overlaid on the left side of the image.

OUR MISSION

Uptake will be the actionable insights platform that makes the world more productive, safe, and reliable.

How Uptake Grew

- Uptake is 33 months old (2.75 years!)
- There are roughly 700 employed
- Hire approximately 25 people a week
- Currently about 400 developers
- Started with one vertical, expanded to 7



What Was So Broken?

- Uptake's deployment process (or lack of process)
- From 10 devs to a 200+ with no change in the process
- From 2 applications to 100+
- Without a unified deployment process, there was no way to keep track of what got deployed when and where.



How Does This Apply To Me?

- Could your team's deployment process use some work?
- If not, what about some other process in place that isn't working?
 - Tickets taking forever to get closed
 - Code review by people who don't know what they're reviewing
 - Weekly meetings that serve no purpose
- Ways to manage chaos and where to start



The Beginning

- We decided on Marathon/Mesos — Why?
 - Rest API
 - Easy app configuration
 - Docker artifact support
 - Load-balanced deployments
- Environment specific config
- No service discovery (which is why you need an environment specific configuration)



The Mess

- Application configuration: Mixture of Spring properties files and environment variables
- Marathon JSON: Template + sed script with environment variables defined in Jenkins jobs
 - Worked okay for 2 apps and 3 environments = 6 jobs
 - Apps went from 2 to 20 and the envs went from 3 to 6 = 120 jobs (Completely unmanageable!)
- No contract testing
- Regression testing against a specific set of app versions and configuration



Application Deployments

HOW THEY USED TO WORK:

- Jenkins jobs per application and environment
 - uptake-core-QA
 - uptake-core-RC
 - uptake-core-STAGING
 -
- Deploy apps one at a time to an environment

6:23 AM [redacted] yes
looks like version 96 was in prod, for like one day
2 weeks ago or so
we've been running 95 on prod since 3/17

6:24 AM [redacted] that would have been around the revert, I thought 1.0 FE was at v 100 for some reason

6:24 AM [redacted] it was
100 is what it was this morning
before i rolled it back to 95

6:24 AM [redacted] right

6:24 AM [redacted] we ran 96 in prod for one day
in april
then rolled back to 95

6:24 AM [redacted] Why did we go from 96 to 95?

6:25 AM [redacted] 4/5 to 4/6
i dont know

6:25 AM [redacted] would have been the roll back I believe
from when sorting first broke



How To Recognize Chaos

WHAT INDICATED THAT THERE WAS A PROBLEM?

- No change in how things were done
- Confusion around what was being deployed
- Lots of time lost to putting out fires
- Deployments started to take forever
- Human-managed process, making it error-prone



How To Communicate The Problem

- Talk to people about it
 - Devs who have to deal with it
 - Managers who don't know why things aren't getting shipped out faster
 - Mention the obvious
- Diagram the current structure to show how ugly it is
 - But have a proposed solution in place



The Plan

- Synthetic YAML processing (Environment, group, and application level configs)
- All configuration and versions are promoted in a monolithic manifest
- Resolve the issues of having hundreds of configurations dispersed and configured differently and hard to parse
- Auditing and approval tracked as git PRs



How Did We Fix It Without Breaking Everybody Else's Stuff?

- Single Jenkins job per environment (All apps are in there)
- Hard work of gathering all the configs was done by my team
- Wrote small ruby app to do this
- Fully owned by the team (with stakeholder by-in)
- Ability to deploy sets of applications to an environment based on what changed (No longer human-managed!)



How To Fix Chaos

- How to communicate change
 - Nemawashi
 - Multiple modes
 - Education!
- Distributing pain
- Sunset the old process/application visibly



The Next Step

- We've only covered the first step!
- 170+ applications
- Modularized deployments
- Releasing customer-specific code separate from the overall Uptake Platform
- Release schedules differ across customers



Thank you.

Colleen Stock, Uptake Engineering



colleen.stock@uptake.com



[@uptake](https://twitter.com/uptake)