Moving From Agile Towards DevOps With a Small Team

Chris Lauer
NOAA Space Weather Prediction Center (SWPC)

Space Weather Prediction Center

- Buried deep in the agency Bureaucracy:
    - Commerce, NOAA, NWS, NCEP, SWPC
- Different subject area, not in common with parent agencies
- FISMA high, “no cloud!” they said
- Cool Mission

First: Agile

We started our move to Agile in 2012

- Sent all our developers to scrum master training
- Found a project that required building team and organization consensus: the new www.spaceweather.gov
- 2-weeks sprints for 8 years now, with several evolutions of team membership and structure.
- Always trying to work closely with our scientists and forecasters

We saw big improvements in morale and capability. It was time to improve delivery

The Challenge

SWPC’s architecture:

- Giant single MS SQL production database for all kinds of data
- Views and stored procedures reaching across schemas to combine disparate data sources into something tidy; lots of business logic in the database
- No service layer - vast majority of applications hit the database directly
- Crons everywhere

Deep coupling was killing our agility.

How do you get started with CI with this?

Seeking out a new way

1. Took some Jenkins/CI training, our first exposure to Docker
2. Labeled what we were doing: shared database integration
3. Read a book on a new way: Sam Newman’s Building Microservices
4. Presented to developers, got buy-in on trying the new way
5. Found a small but high value project (Verification Service)
6. Split our too-large team in two, one to work the database and one to work the new way
7. Got Slack
8. Developed the Verification Service with new rules

New Rules

- Only one thing (service) talks to a database
- Service Oriented: A collection of loosely coupled, highly cohesive services that each solve part of the problem
- CI friendly: Jenkins can stand up all components and dependencies, drop test data in the ingest and query the results. Fully automated end-to-end testing.
- Event driven, low latency
- 12factor.net: especially “store config in the environment”
- Oh, and let’s try a bunch of new stuff:
    - NoSQL (MongoDB) for a more developer friendly database
    - Python Flask for RESTful gets/posts
    - Asynchronous Messaging (RabbitMQ) for loose coupling
    - Docker containers for everything

How did it go?

Amazing.

- Developers were excited to learn contemporary technologies. Pace of learning was rapid!
- Iterative changes were much easier
- Customer was thrilled with the capabilities we built, and how quickly we delivered
- It was faster to learn all this new stuff than to deliver in the old system

But..

- Microservices team got way out ahead of the organization very quickly
    - Other software developers
    - Ops team
- CI tests were quickly ignored (whoops!)
- Difficult to build our containers on secure networks
- Next steps became.. difficult
    - No mature container orchestration: we were stuck using docker compose, container sprawl
    - Our tests were breaking our agility, developers were upset with Jenkins
    - Security failures in pen testing: Nagios monitoring of container status resulted in privilege escalation, Jenkins was also compromised

Keep Iterating on what you can control

- Locked Jenkins down
- No more nagios monitoring through docker daemon, use container endpoints
- Got an on-prem docker image repository (Harbor), no longer building on staging or production (huge improvement!)
- Socialized the benefits of containers and CI to scientists, developers, managers, ops team (formally and informally)
- Containers became the new way: great for managing complicated/conflicting dependencies, developing locally, and almost push-button deployments

Signs of hope

- We’ve delivered some complicated systems to production very quickly (space weather models, the new GOES satellite processing)
- Ops team lead asked if we’re using Kubernetes!
- Linux admin is starting to use containerized versions of some of his tools
- We might get some cloud in a year or two - who knows?

Next Steps

- New rule: all images on staging/production must be built by CI
- Centralize container logging (ELK?)
- Get an orchestrator! Experimenting with Docker EE, OPs team might be leaning towards Pivotal/VMWare. Swarm would be great, but it’s probably on the way out. We’re finding this to be a difficult choice.
- Try to containerize as much legacy as we can with the RHEL 6 end of life
- Cloud, cloud, cloud - keep pushing: developers are frustrated with having to build/maintain so much DevOps tooling on-prem

Questions and Feedback?

Thanks!