Agile & DevOps for Complex Enterprise Systems

Matt McKnight, VP Solutions Development

24 September 2015
Why Agile?

What does this buzzword really mean?

• The 2001 manifesto and principles
• Scrum, Lean, Kanban, SAFe, XP, Crystal
• Frequent, incremental deliveries to realize value
• Testing throughout
• A definition of “done”
• Respond to change over follow a plan
Agile on Complex Enterprise Systems

What complications arise?

• Architecture doesn’t support dividing up the work
• Interdependencies between systems and teams
• Teams working at different paces and places
• Teams working for different companies, on different contracts, or reporting to different organizations

Let’s look at DevOps as a way to deal with some of these concerns
Why DevOps?

Development Team

throw code

Operations Team
Combine functional Silos

• “Bad behavior arises when you abstract people away from the consequences of their actions.” –Jez Humble
Need practices that scale to large cluster deployments

• New cluster software stacks, such as Hadoop, require coordinated deployments across large numbers of machines
• The deployment itself is complex enough to require significant scripting
• Manual system administration process can’t keep up
Need practices that scale to reliable, fast releases

• Example: rapidly deploy patches for security incidents without reliability impacts

• High-performing IT organizations deploy 30 times more frequently, with 200 times shorter lead times*.  
  – 60x fewer failures and 168x faster

*Puppet Labs 2015 state of DevOps report
Development team needs operational team to:

• Development team needs from the operational team:
  – A reliable infrastructure
  – Self-service environments and deployments

• Operational team needs from the development team:
  – Automate the build process
  – Simplify deployment
The SAFe Deployment Pipeline
Automated Testing and QA

• Automate
  – Builds
  – Functional testing
  – Performance testing
  – Security testing
So Should you Create a DevOps Team?

- A team doesn’t have to be the answer, but we need tighter coordination between the functions.
- Integrated teams can be composed of individuals from different organizations, as a way of uniting towards a common objective.
What has the industry learned from the big guys

- Google: If you buy enough software, something is always breaking. You need processing models that are robust to machine failure (Google File System, BigTable, MapReduce)
- Netflix: You need to aggressively test your systems to be robust to failure. (ChaosMonkey)
- Amazon service testing
- Cloud enabled zero downtime deployments (A/B clusters)
- GSA 18F DevOps went with a container model to “move the vast majority of the security and review burden to the platform itself”
Tools: Deployment

• Automate the steps required to get a version of the application deployed—even when that deployment includes multiple servers
• Prevent local changes that create snowflake servers
• Configuration Management: Ansible, Puppet, Chef, SaltStack, etc.
• Cluster Management: Kubernetes, Mesos, Marathon
Tools: Containers

- Easy deployments assuming
  - Linux has won in the data center

- The best purpose of containers is **not** trying to run multiple machines on one machine- if you want to do that just use VMWare, Xen, etc.

- The purpose is having a pre-installed build of the software that can be immediately dropped on any Linux kernel running the container software

- You avoid the performance penalty of a VM

- The container is smaller than a VM, because it doesn’t include the OS, just the components needed by the application

- Look at: Docker, Rocket, LXD, Drawbridge, CoreOS
Real World Takeaways

• DevOps is a new kind of cross-functional role, and/or a new kind of team
  – Need people that understand systems and can script their automated management.

• Automation is a key to dealing with complexity and scale

• Standard containers can allow for verification work at multiple levels of scale

• Not all projects need to be hyper-agile, not all projects need to be hyper-reliable and scalable. Focus your process and energy at the right level.

• Focus on figuring out how to not be the bottleneck in the value stream