# DevOps

Arun Gupta, @arungupta
Red Hat

# Delivering Software: Reality vs Goal

| BUDGET | TIME | VALUE |
| --- | --- | --- |
| 45% OVER | 7% OVER | 56% LESS |

Source: Delivering large-scale IT projects on time, on budget, and on value
McKinsey & Company, October 2012

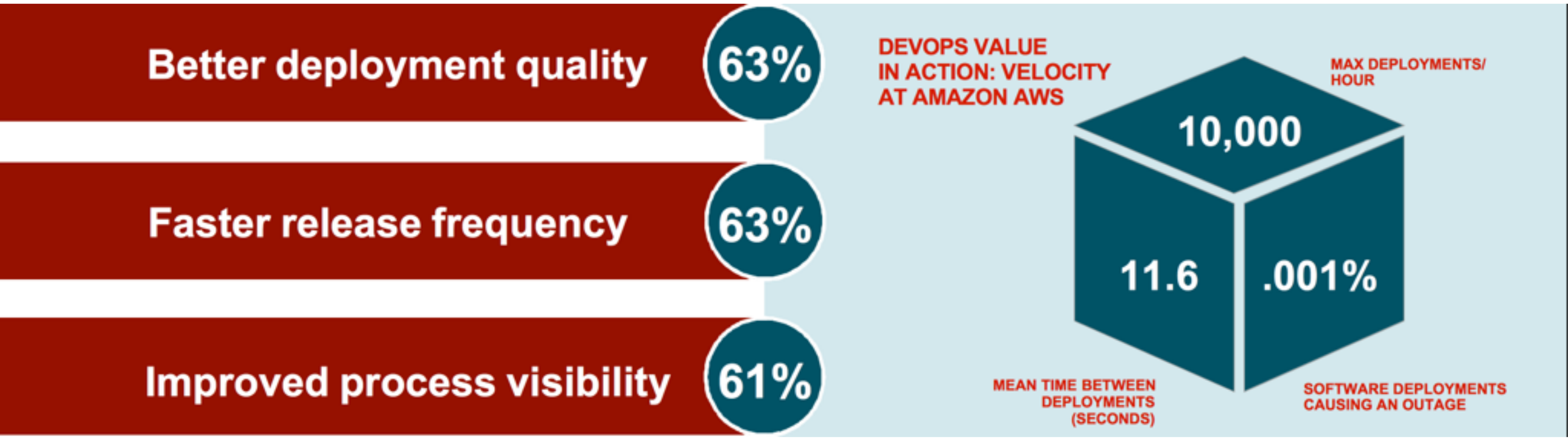
# Is DevOps for you?

**Ship code 30x faster**
and complete those deployments 8,000 times faster than their peers.

**Have 50% fewer failures**
and restore service 12 times faster than their peers.

https://puppetlabs.com/wp-content/uploads/2013/03/2013-state-of-devops-report.pdf

# Organizations implementing DevOps



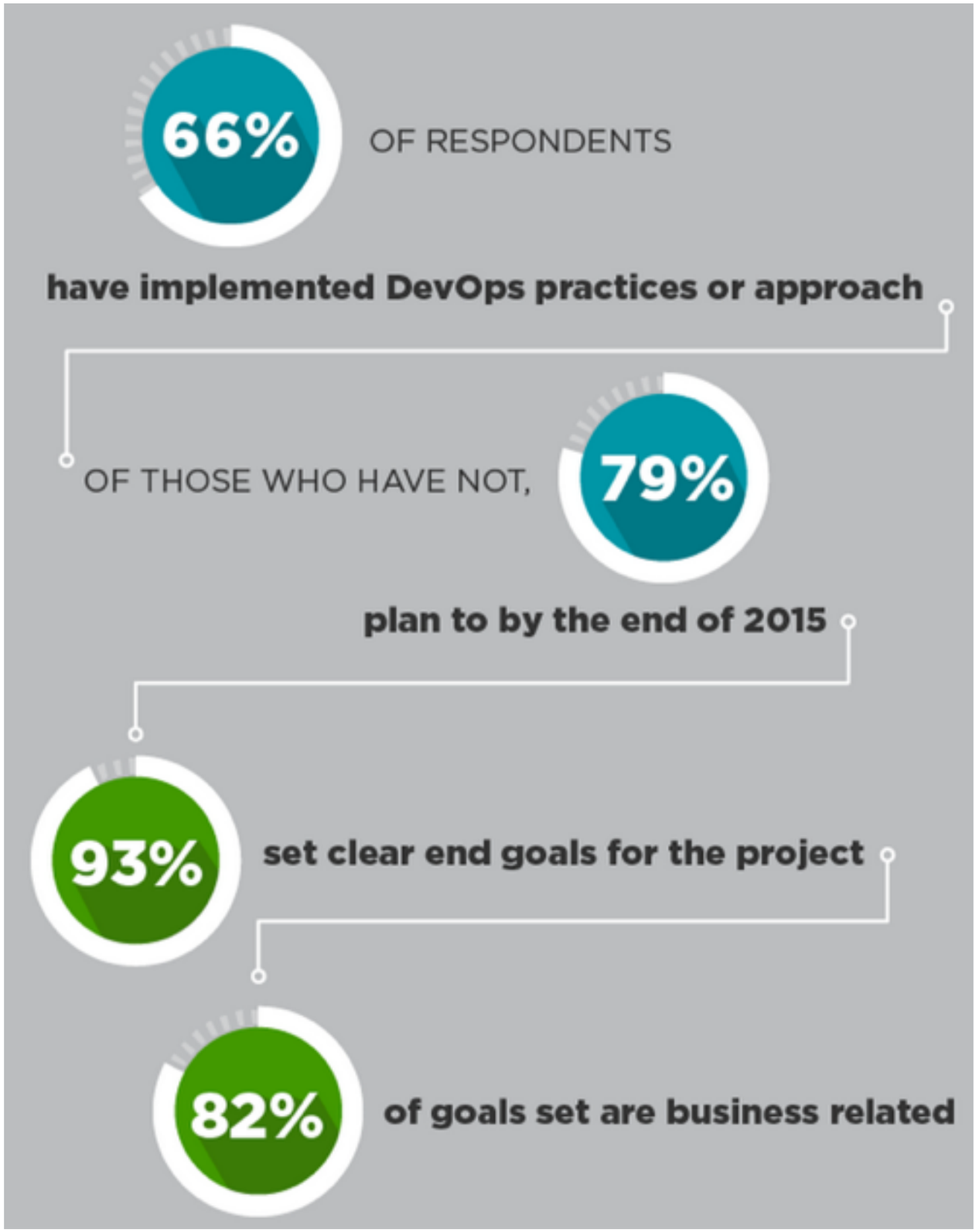

https://puppetlabs.com/wp-content/uploads/2013/03/2013-state-of-devops-report.pdf

# Organizations implementing DevOps

Numbers missing?

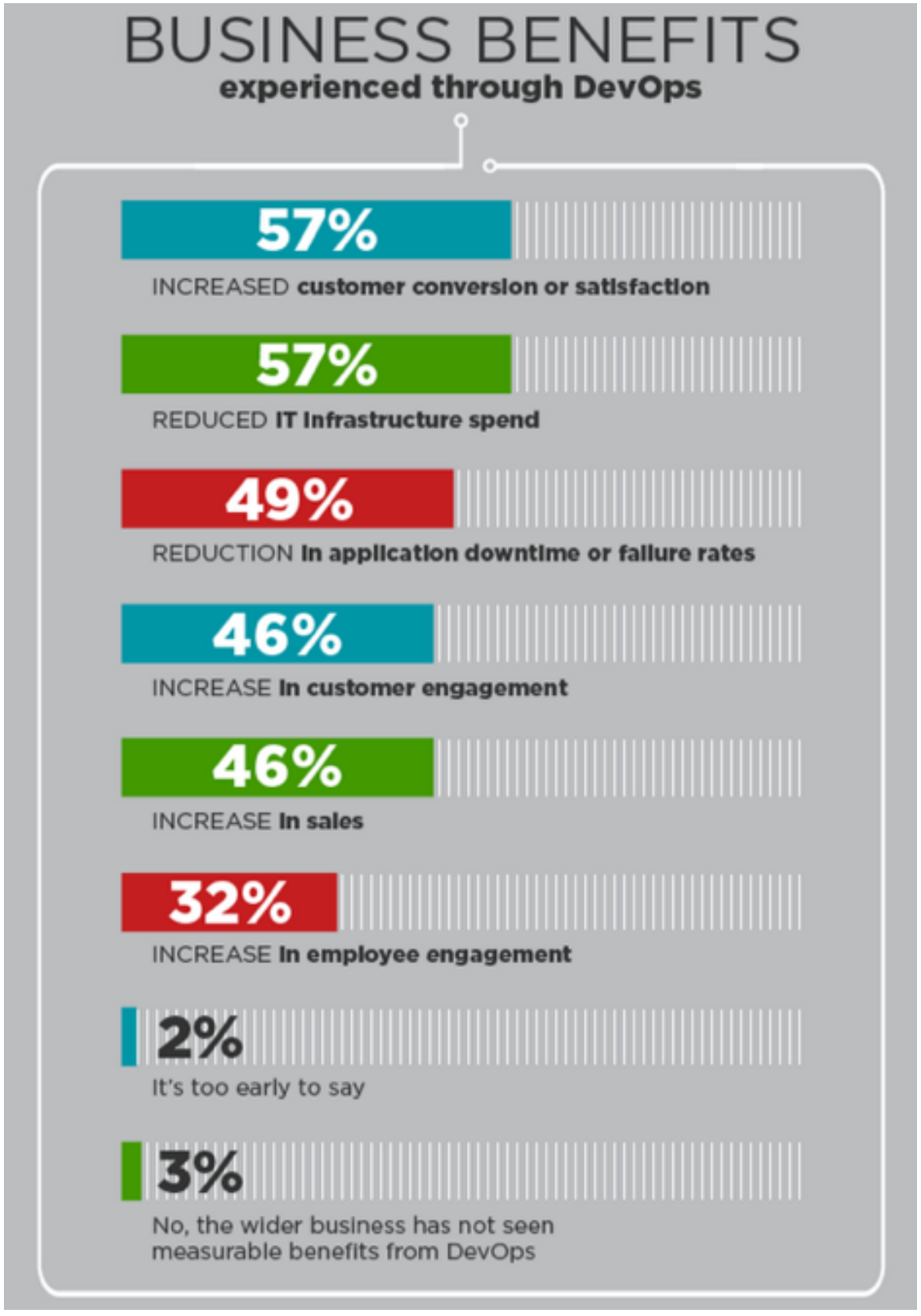
http://puppetlabs.com/sites/default/files/2014-state-of-devops-report.pdf

Oct 2014



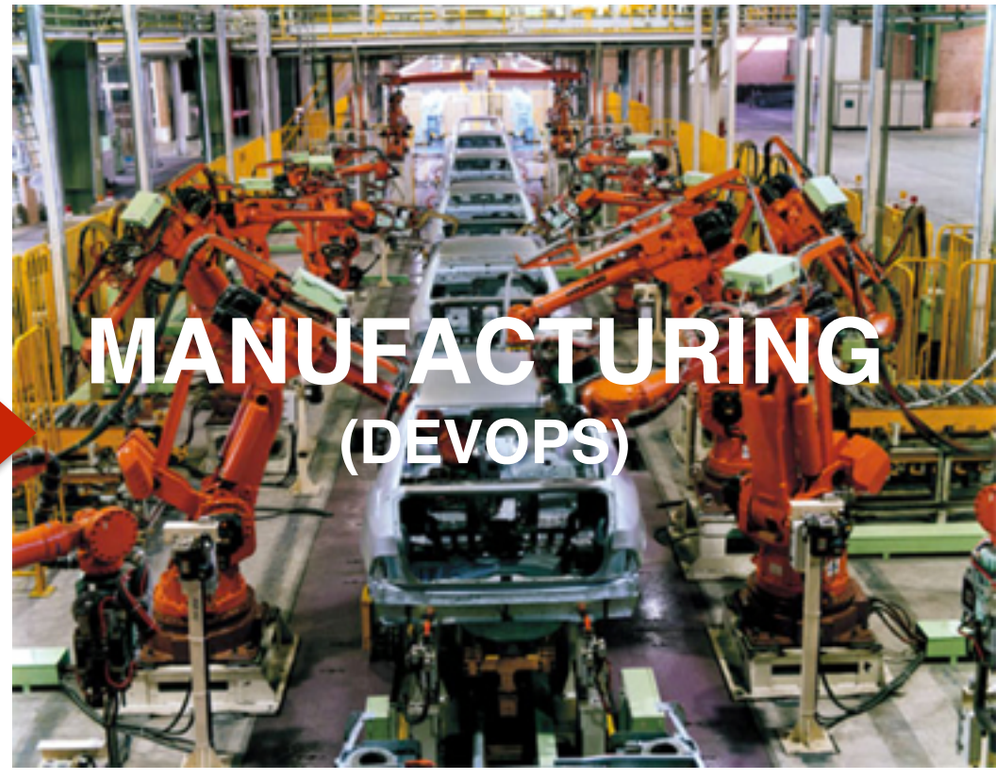

http://www.rackspace.com/blog/a-devops-snapshot-rackspace-survey-shows-businesses-benefits-of-devops-infographic/

Oct 2014




http://www.rackspace.com/blog/a-devops-snapshot-rackspace-survey-shows-businesses-benefits-of-devops-infographic/

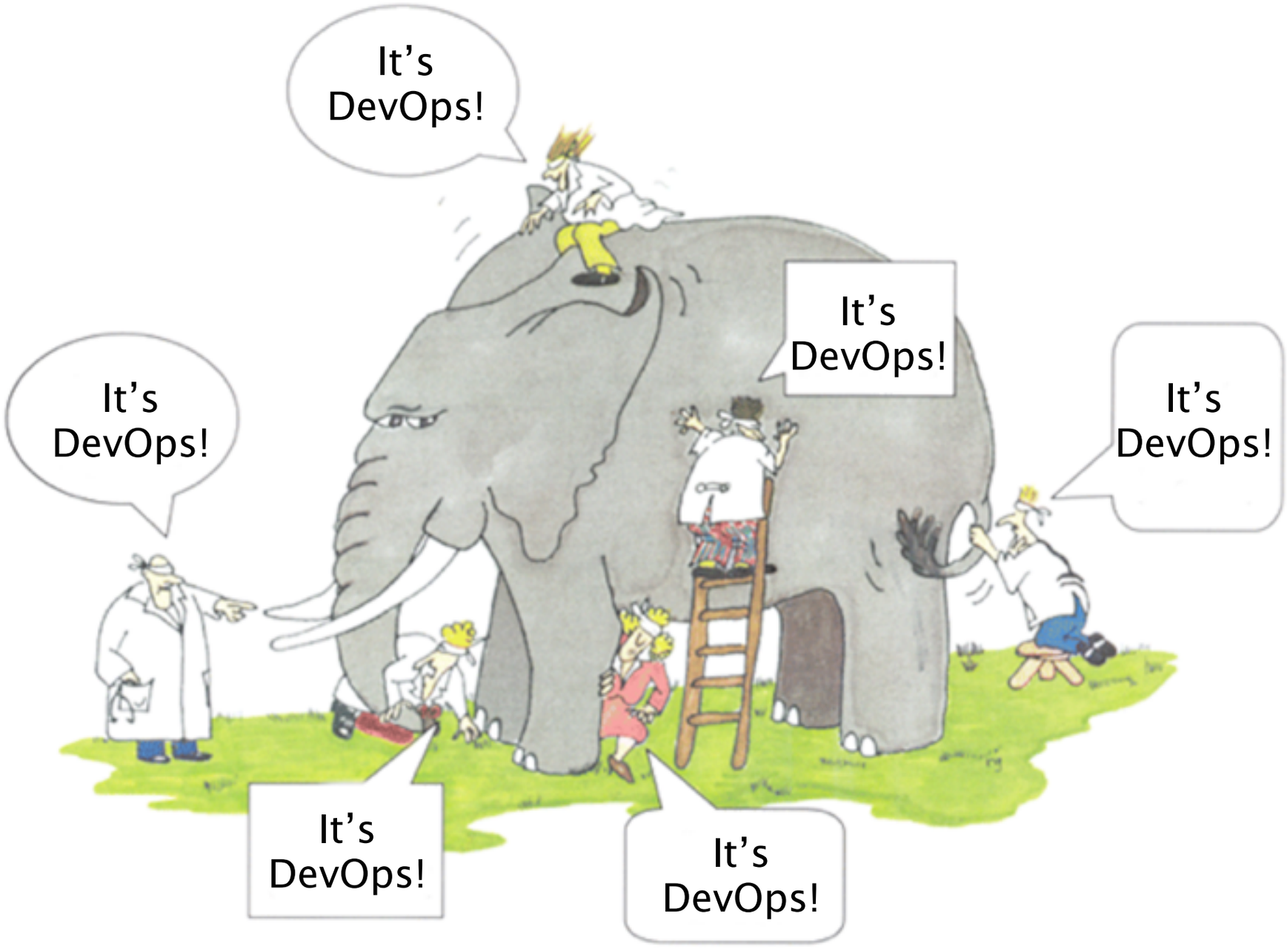

CRAFTWORK


MANUFACTURING
(DEVOPS)


WORKSHOP


FACTORY
(CLOUD)

# What is DevOps?



# DevOps is...

* A philosophy that starts with passion
* A cultural, professional movement with attitude and values
* A reaction to poor communication
* About creating visibility between dev and ops
* About the symbiotic relationship between dev and ops
* Cross-functional teams over organizational silos
* Products not projects
* Automation over documentation (and more automation... and more...)
* About creating self-service infrastructure for teams
* Knowing that good software doesn't end with development / release
* Software that doesn't require support
* Ensuring a continual feedback loop between development and operations
* Cross-functional teams over organizational silos
* Creating products that are owned by the delivery team
* Knowing that a project is only finished when it is retired from production
* Something you can do without doing agile

https://sites.google.com/a/jezhumble.net/devops-manifesto/

# Key Components

- People - “Dev” and “Ops”
- Processes - Continuous Delivery, Agile, …
- Technology - Jenkins, Chef, Puppet, Arquillian, …

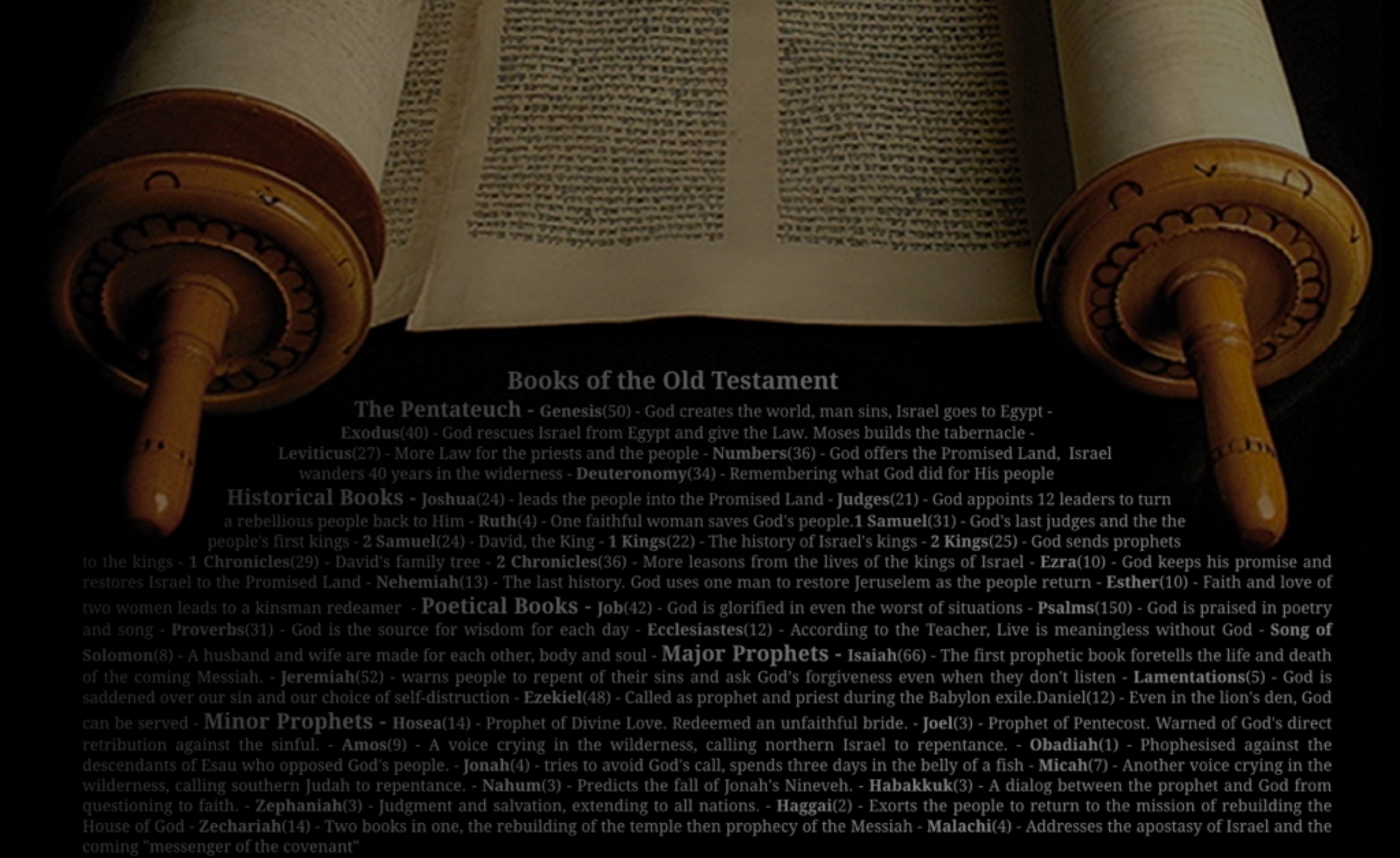
# Five “C”s of DevOps

- **C**ollaboration between “dev” and “ops”
- **C**ulture
- **C**ode everything - application and configuration
- **C**onsistency - automation over documentation
- **C**ontinuous delivery

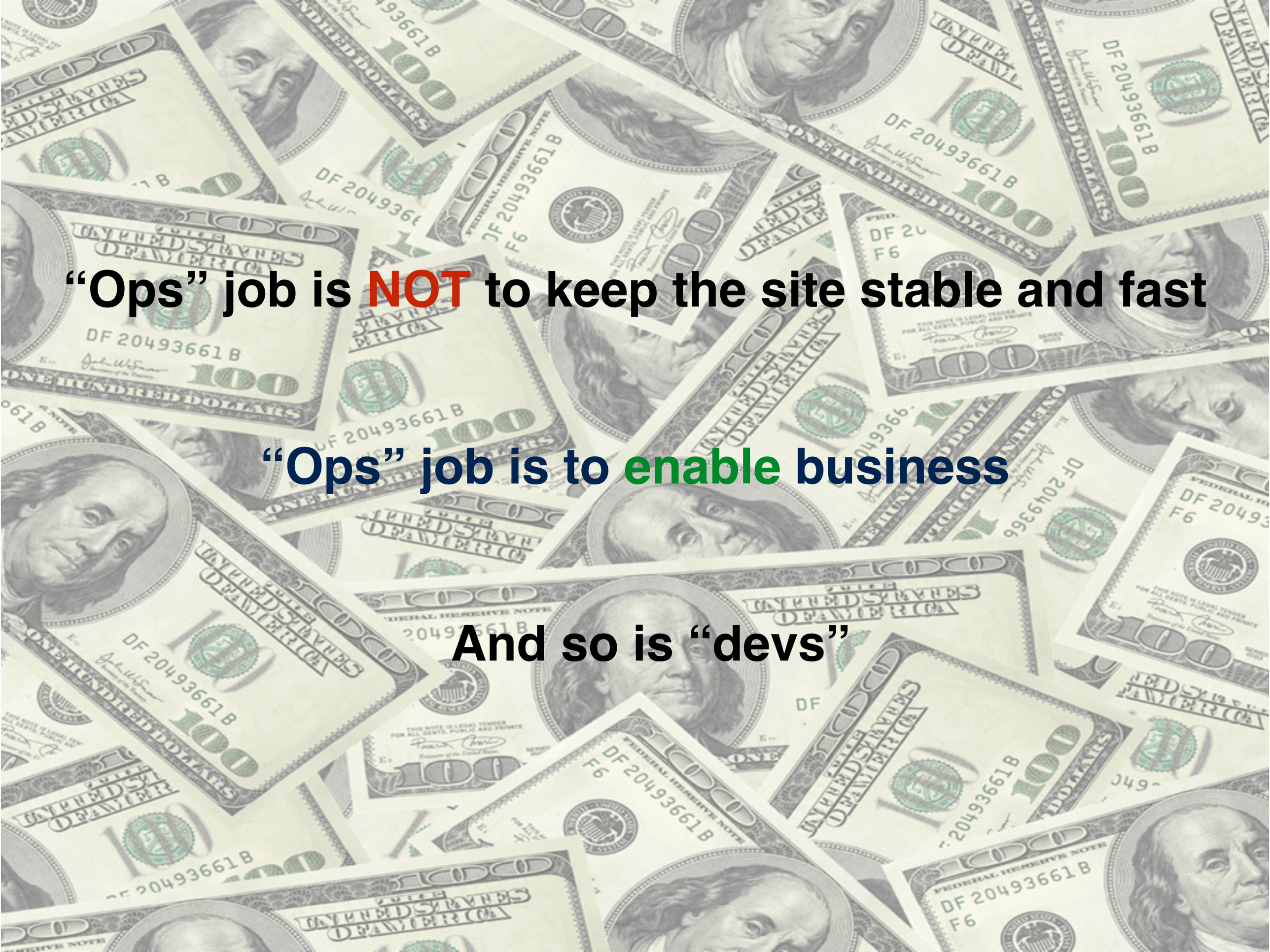
# **C**ollaboration

- "Dev"
  - Engineering
  - Test
  - Product management

- "Ops"
  - System administrators
  - Operations staff
  - DBAs
  - Network engineers
  - Security professionals



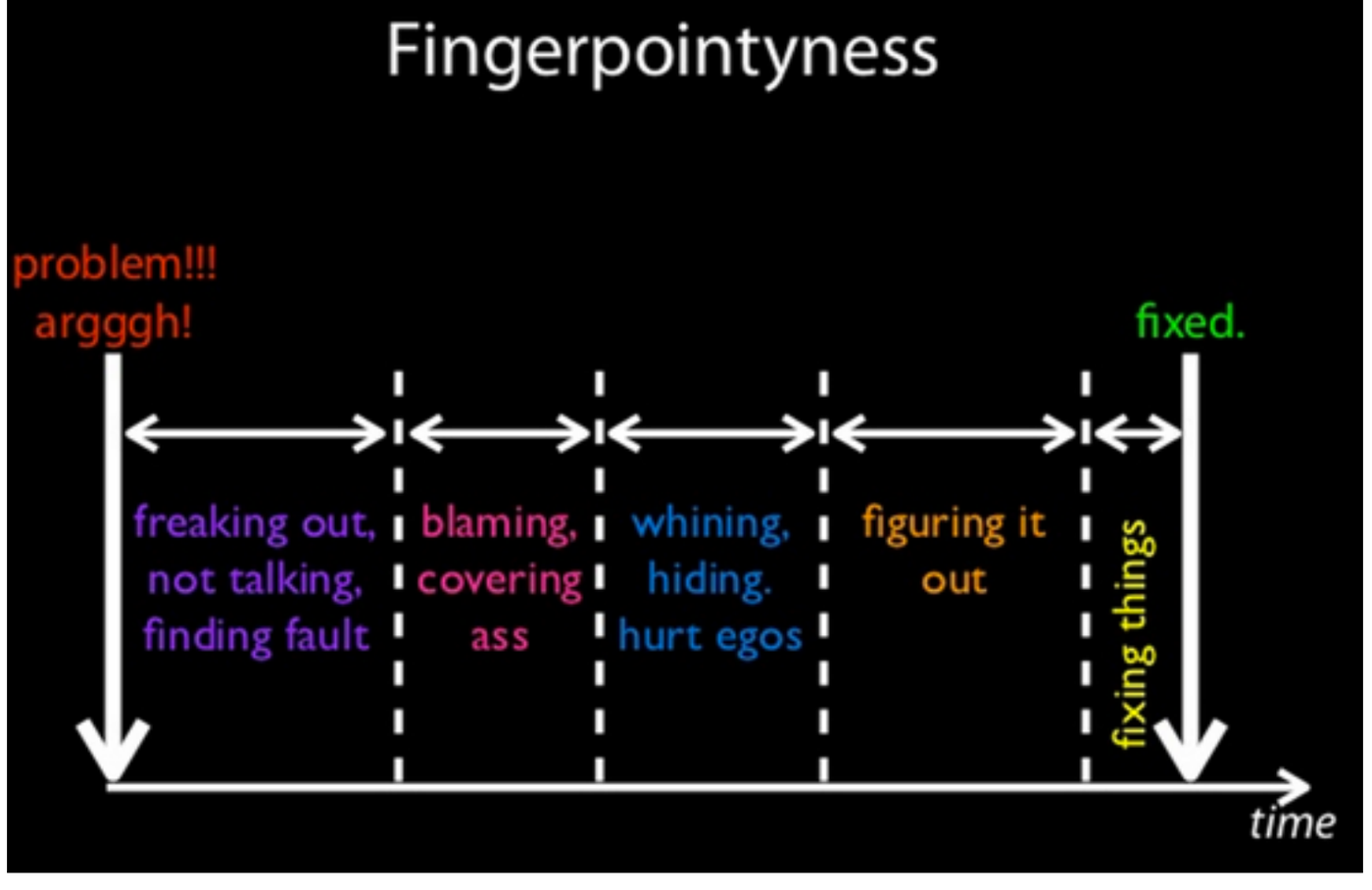

**“Devs” job is to add new features**
**“Ops” job is to keep the site stable and fast**

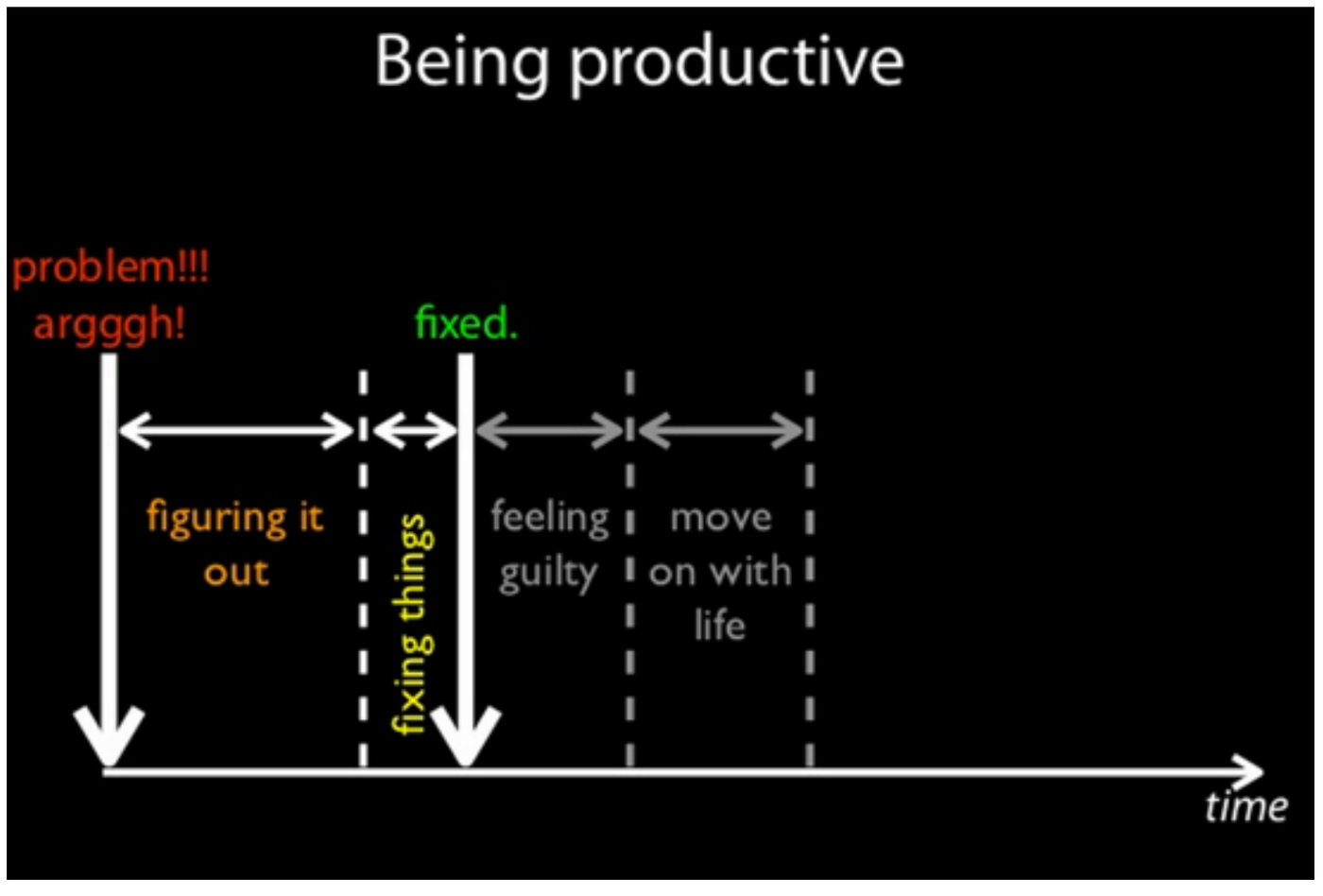

“Ops” job is NOT to keep the site stable and fast
“Ops” job is to enable business
And so is “devs”


“Dev” “Ops”

# **C**ulture

- Respect other's expertise, opinions, responsibilities
- Trust: Ops to think like devs, devs to think like ops
  - Leads to transparency
- Don't ignore failure, build joint recovery plans
- Amplify feedback loops



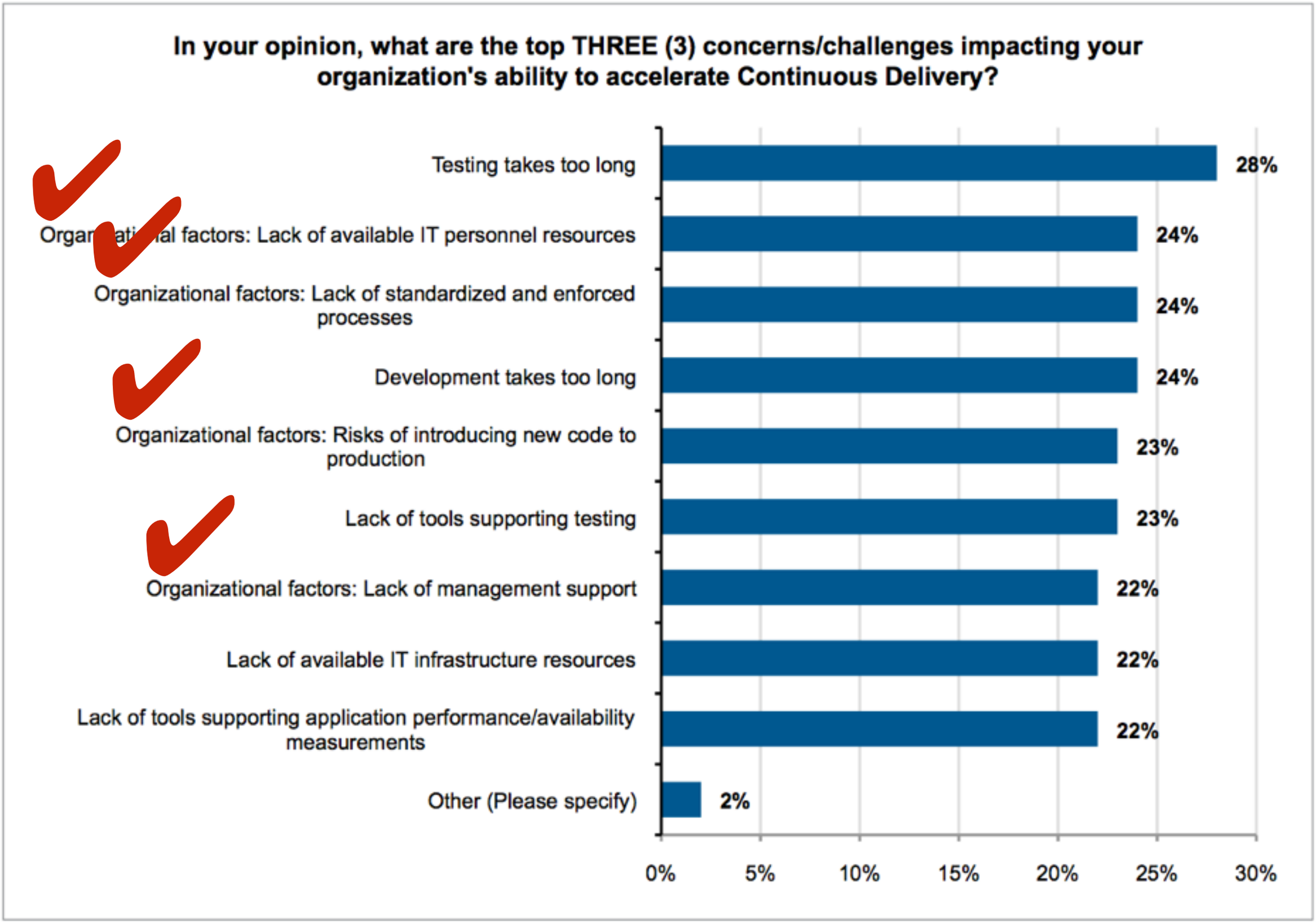




http://www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr

"Treat people warmly, issues coldly!"

With great power, comes great responsibility

"you build it, you run it!"




http://pages.cloudbees.com/rs/cloudbees/images/CloudBees-EMA-DevOps_2014.pdf

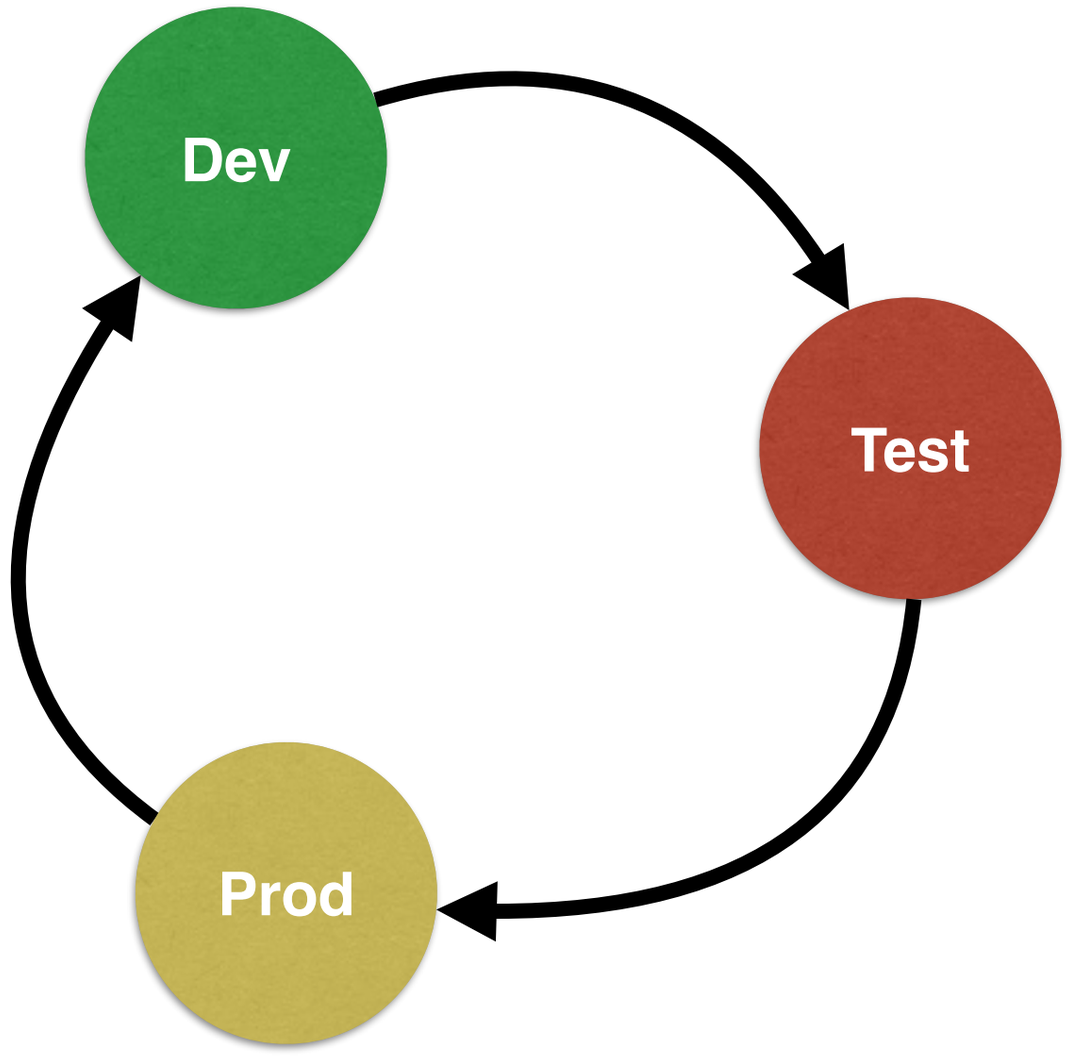
# **C**ode everything

- Application code
- Build scripts
- Database schema
- Configuration files
- IDE configurations
- Infrastructure
- Deployment scripts
- Test code and scripts
- Provisioning scripts
- Monitoring
- Logging
- …

*“Never send a human to do a machine’s job”*

# **C**onsistency

- Automation over documentation
  - Repeatability
  - Push-button deployments
  - Managing environments



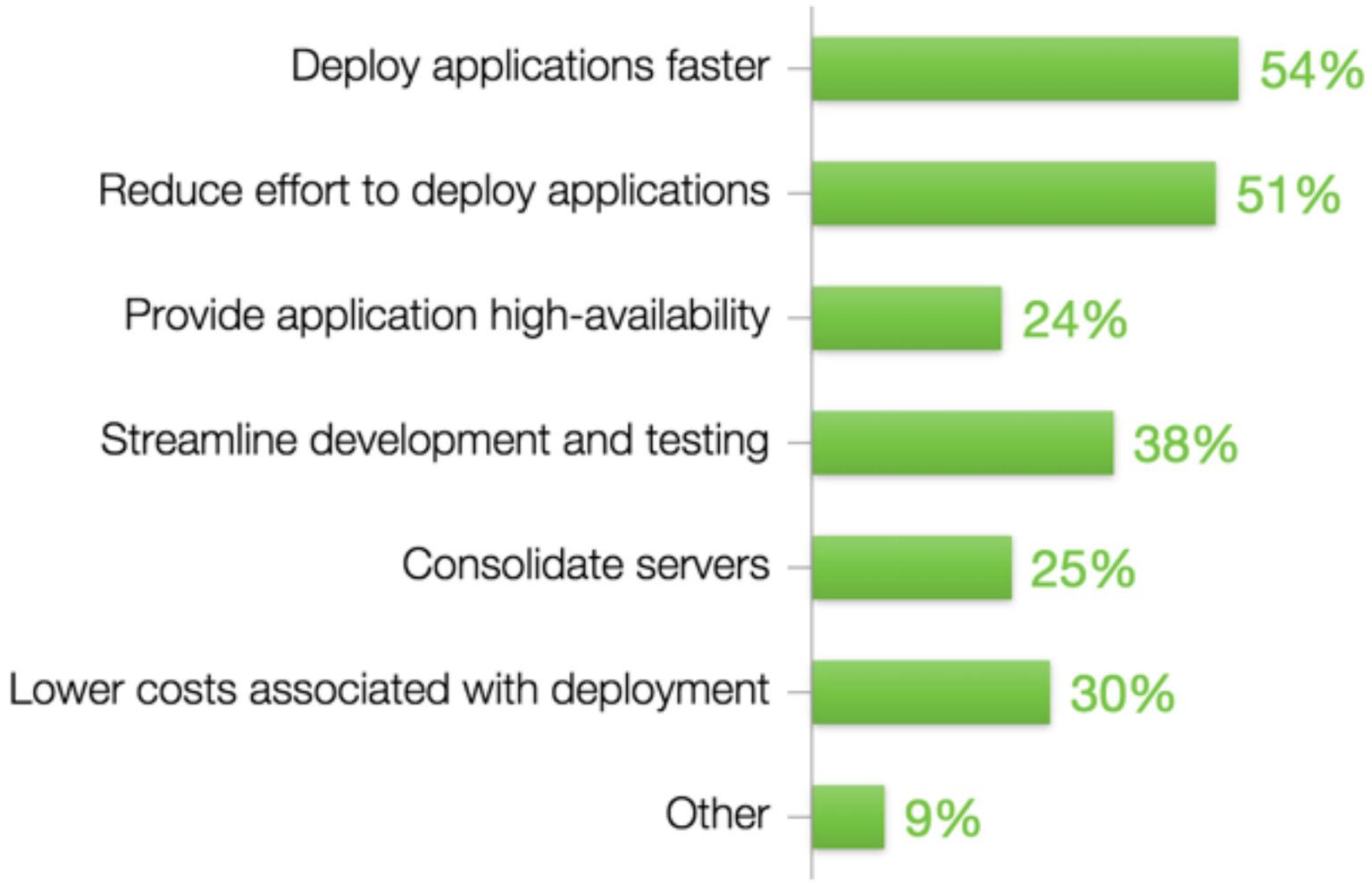
# Manage environments

- Cloud computing: PaaS, OpenShift, …
- Virtualization: Virtual Box, Vagrant, …
- Containers: Docker, Rocket, …
- App server: JBoss EAP, Tomcat, …
- Configuration tools: Puppet, Chef, Ansible, Salt
- Orchestration: Kubernetes, Swarm, …


- Simple and portable way to create VMs
- Works with VirtualBox, VMWare, Docker, …
- Environments packaged as “boxes”
- Commands: **`vagrant up`**, **`destroy`**, **`login`**, …
- Minimize impedance mismatch between Dev, QA, Prod

# Benefits of containers

What are the top benefits you see with containers?




Note: this is a multiple-choice question – response percentages may not add up to 100.

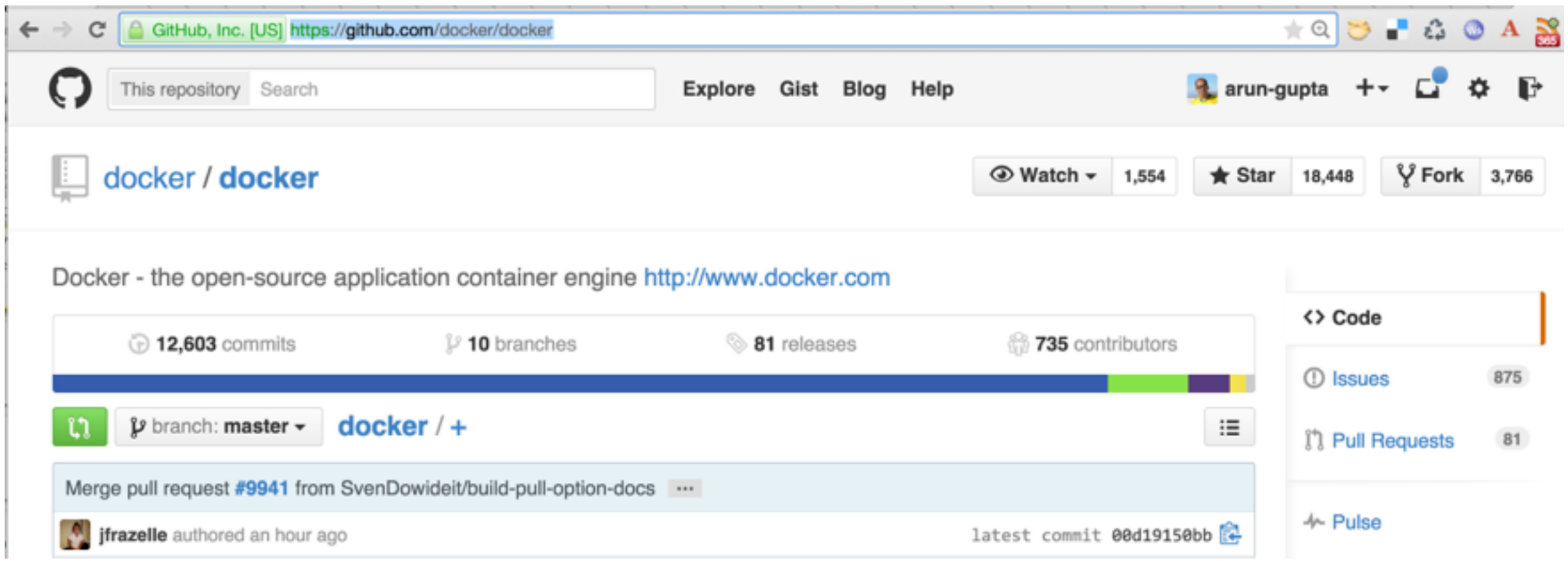
**Source:** TechValidate survey of 79 IT professionals

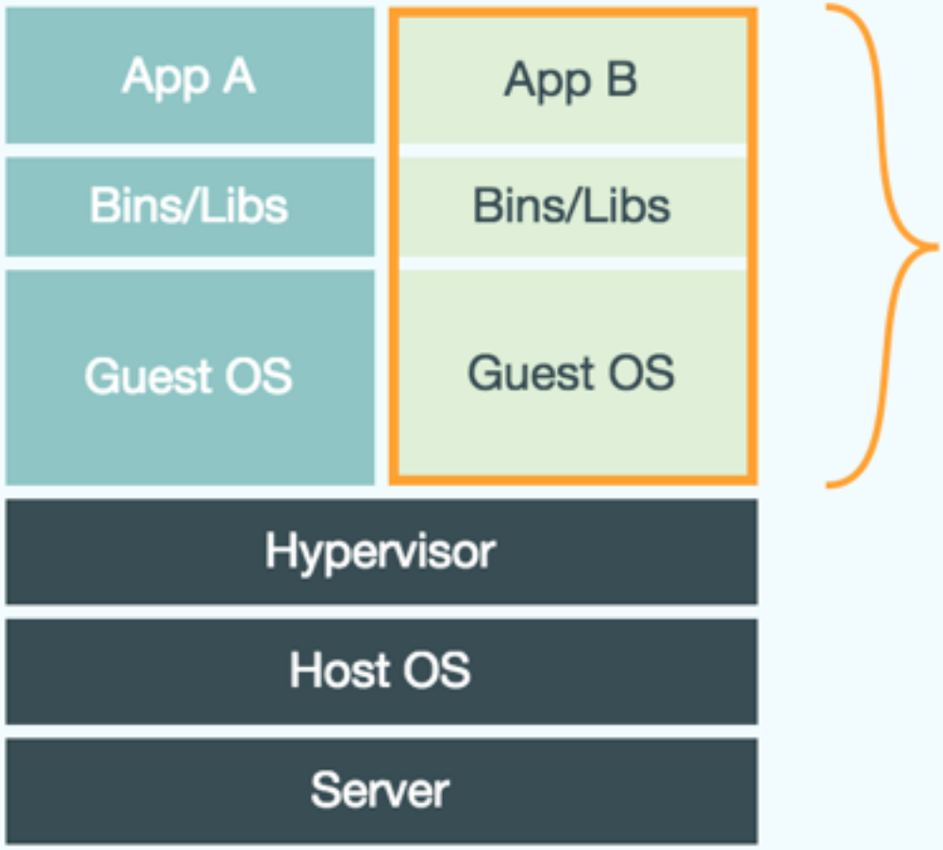
http://www.techvalidate.com/product-research/red-hat-enterprise-linux/charts/E21-1EE-BF2

# Containers

- Faster deployments
- Isolation
- Portability - “it works on my machine”
- Snapshotting
- Security sandbox
- Limit resource usage
- Simplified dependency
- Sharing

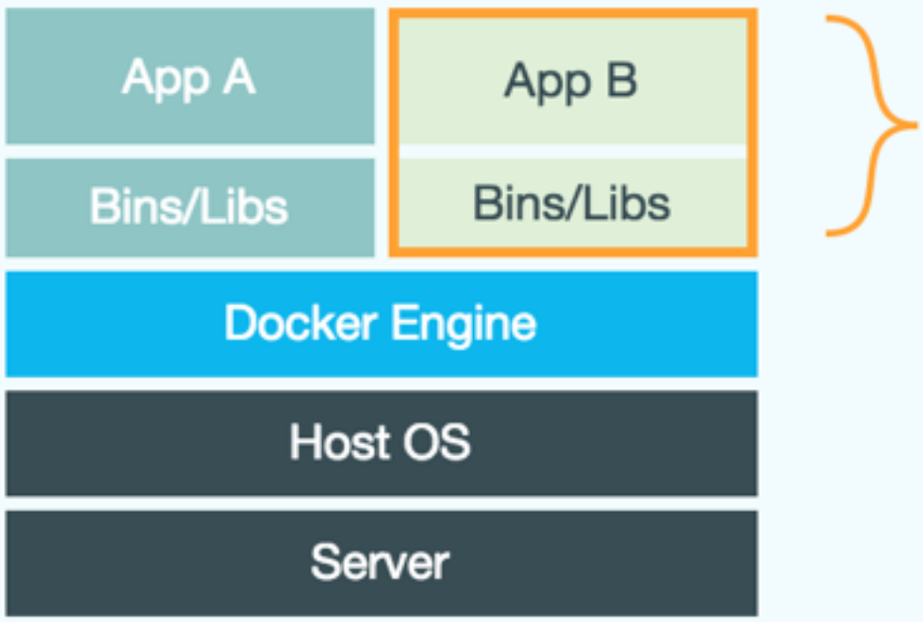

- Open source project and company
- Used to create containers for software applications
- Package Once Deploy Anywhere (PODA)






# Virtual Machines

Each virtualized application includes not only the application - which may be only 10s of MB - and the necessary binaries and libraries, but also an entire guest operating system - which may weigh 10s of GB.




# Docker

The Docker Engine container comprises just the application and its dependencies. It runs as an isolated process in userspace on the host operating system, sharing the kernel with other containers. Thus, it enjoys the resource isolation and allocation benefits of VMs but is much more portable and efficient.

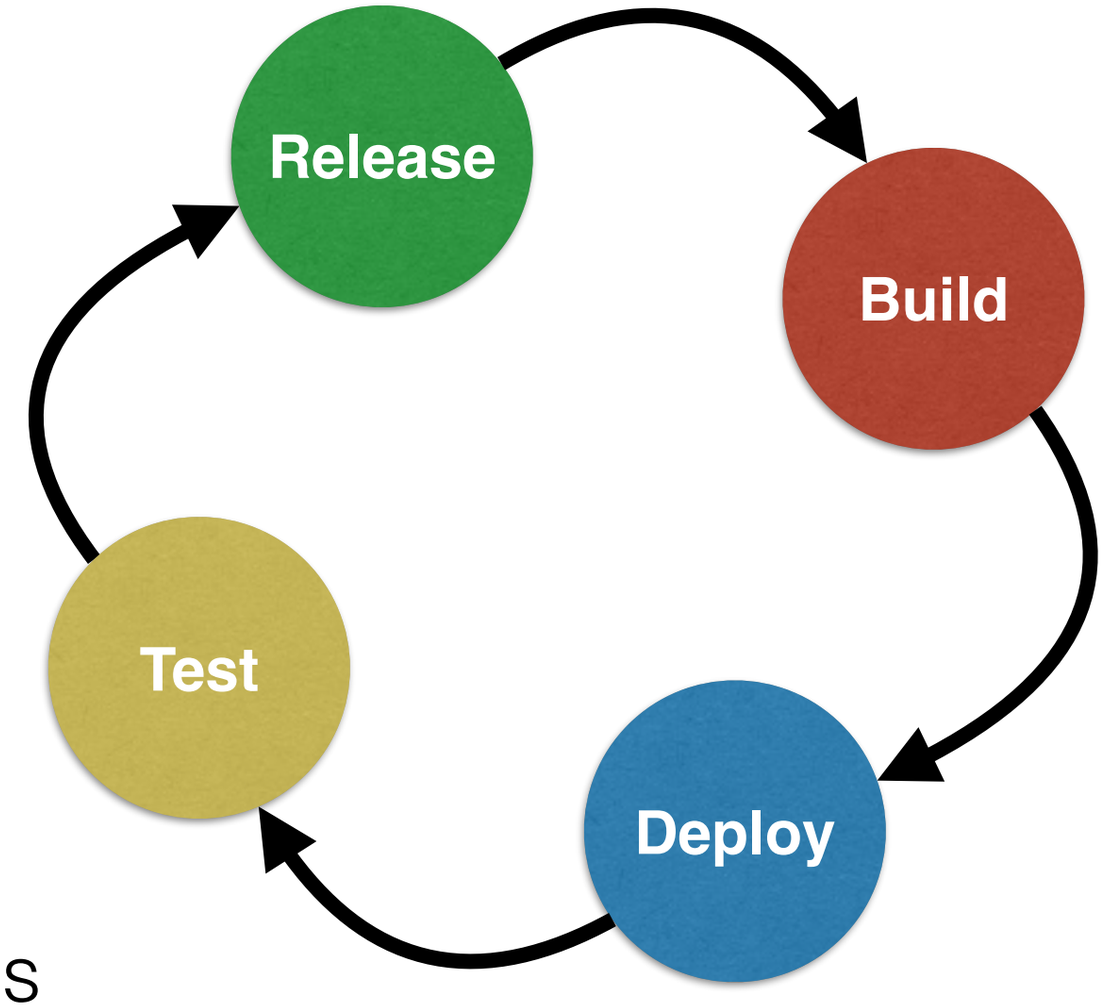
# Dashboards

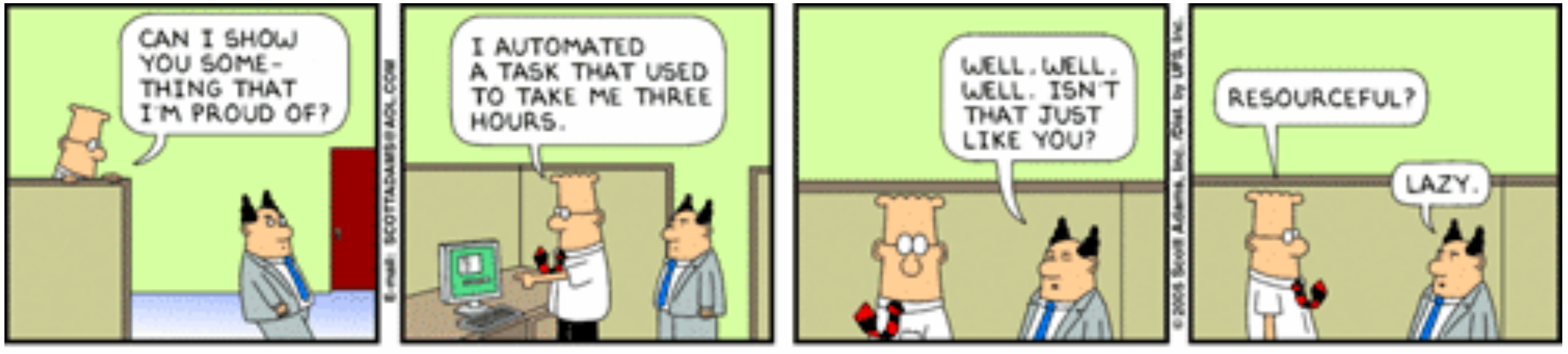
- Build dashboards, improve transparency

- stackexchange.com/performance

# Continuous Delivery

- Agile Manifesto
- Continuous Integration
- Fail fast and recover
- Self service
- 100% Automation
- Proactive monitoring and metrics
- Push to Prod




CAN I SHOW YOU SOME-THING THAT I'M PROUD OF?
I AUTOMATED A TASK THAT USED TO TAKE ME THREE HOURS.
WELL. WELL. WELL. ISN'T THAT JUST LIKE YOU?
RESOURCEFUL?
LAZY.
DID YOU JUST TURN MY BRILLIANT ACCOMPLISHMENT INTO A CHARACTER FLAW?
COMPLAINER.
LETS JUST FORGET THE WHOLE THING.
QUITTER.

Our highest priority is to satisfy the customer through **early and continuous delivery** of valuable ~~software~~ service.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

**Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

**Business people and developers must work together daily throughout the project.**

Build projects around motivated individuals. **Give them the environment and support they need**, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

**Working software is the primary measure of progress**.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

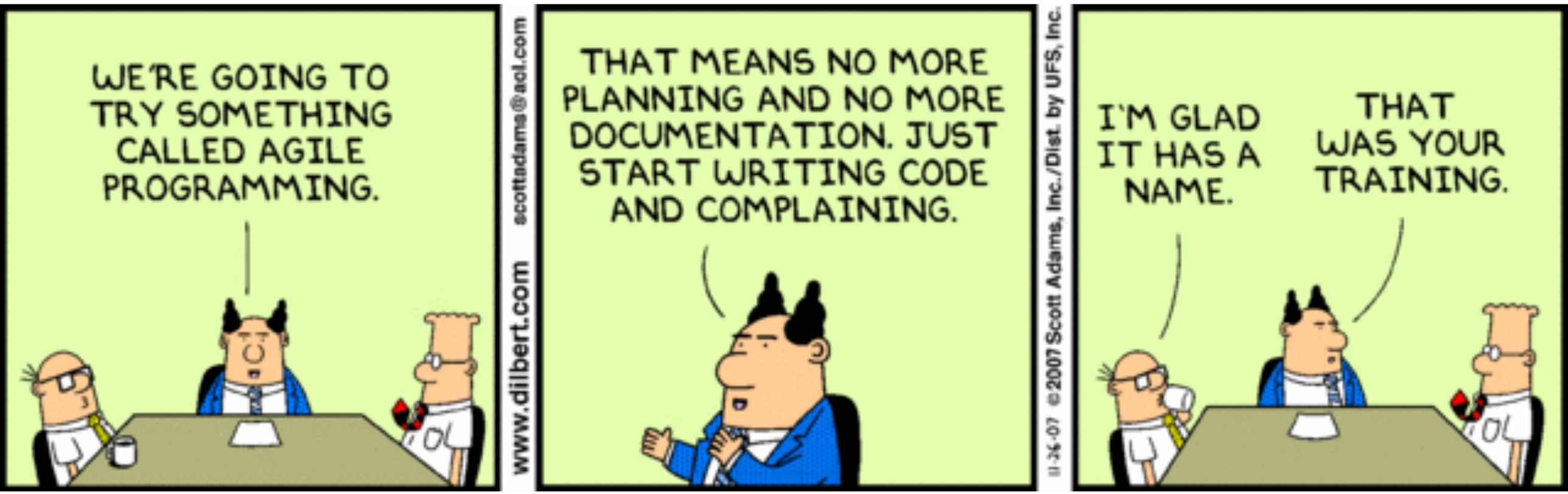
Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

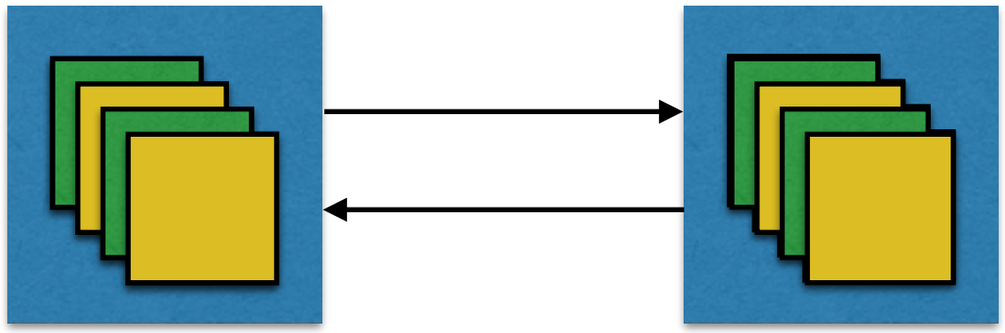

WE'RE GOING TO INTRODUCE AGILE METHODOLOGIES
www.agileweboperations.com
WILL WE NEED TO CHANGE THE WAY WE WORK?
2012 © Matthias Marschall
I DON'T THINK SO. LET'S JUST REPLACE „PHASED APPROACH" WITH „AGILE APPROACH" IN OUR PROCESS DOCUMENTATION


WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.
www.dilbert.com scottadams@aol.com
THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.
I'M GLAD IT HAS A NAME.
THAT WAS YOUR TRAINING.
11-06-07 ©2007 Scott Adams, Inc./Dist. by UFS, Inc.

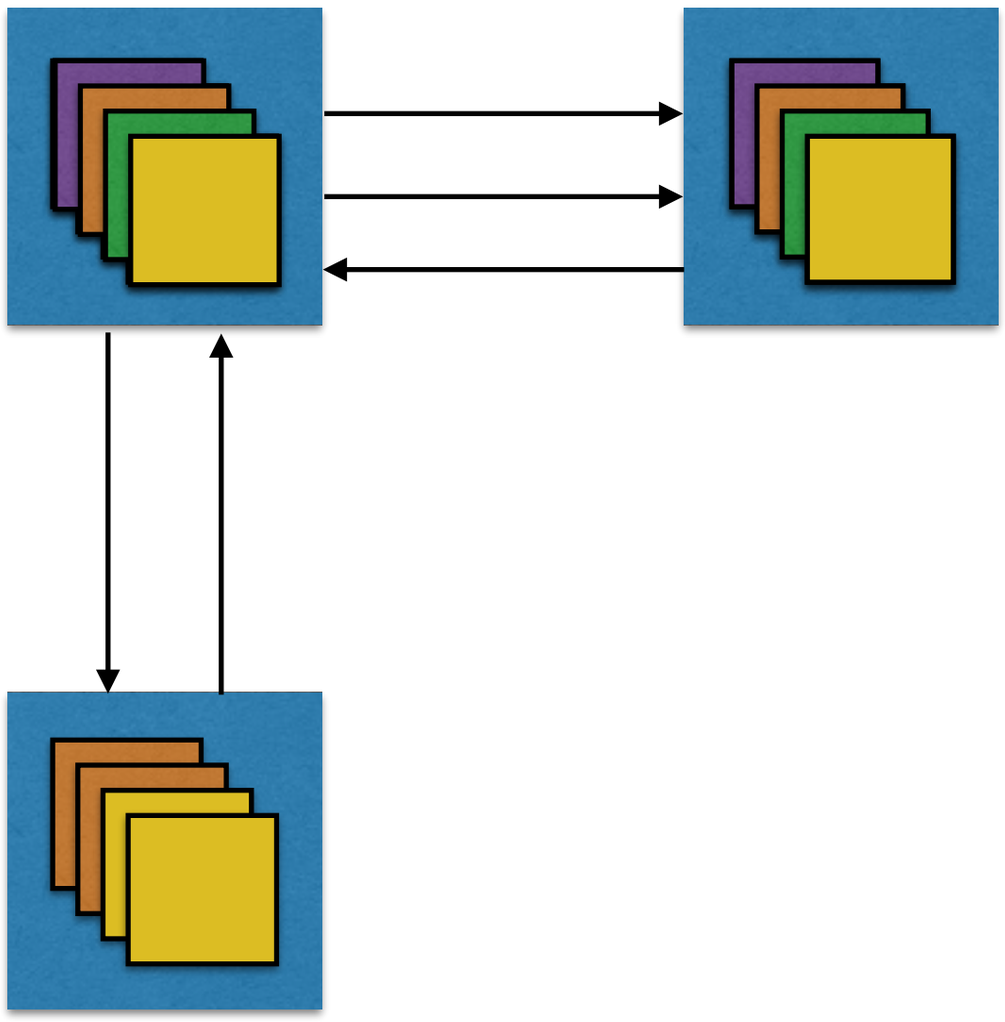
# Tools

- CI: Jenkins, Travis CI, Bambo, …
- Versioning: Git, SVN, Mercurial, …
- Monitoring: Nagios, NewRelic, Pingdom, StatsD, …
- Logging: Log stash, ??

# Continuous Integration - One Developer



# Continuous Integration - Multiple Developers



# Tenets of CI

- Source code repository is the “single source of truth”
  - All source code - application and test
  - Include build/test scripts, schemas, IDE configurations, docs, …
  - Tag your builds, define cadence (features?)

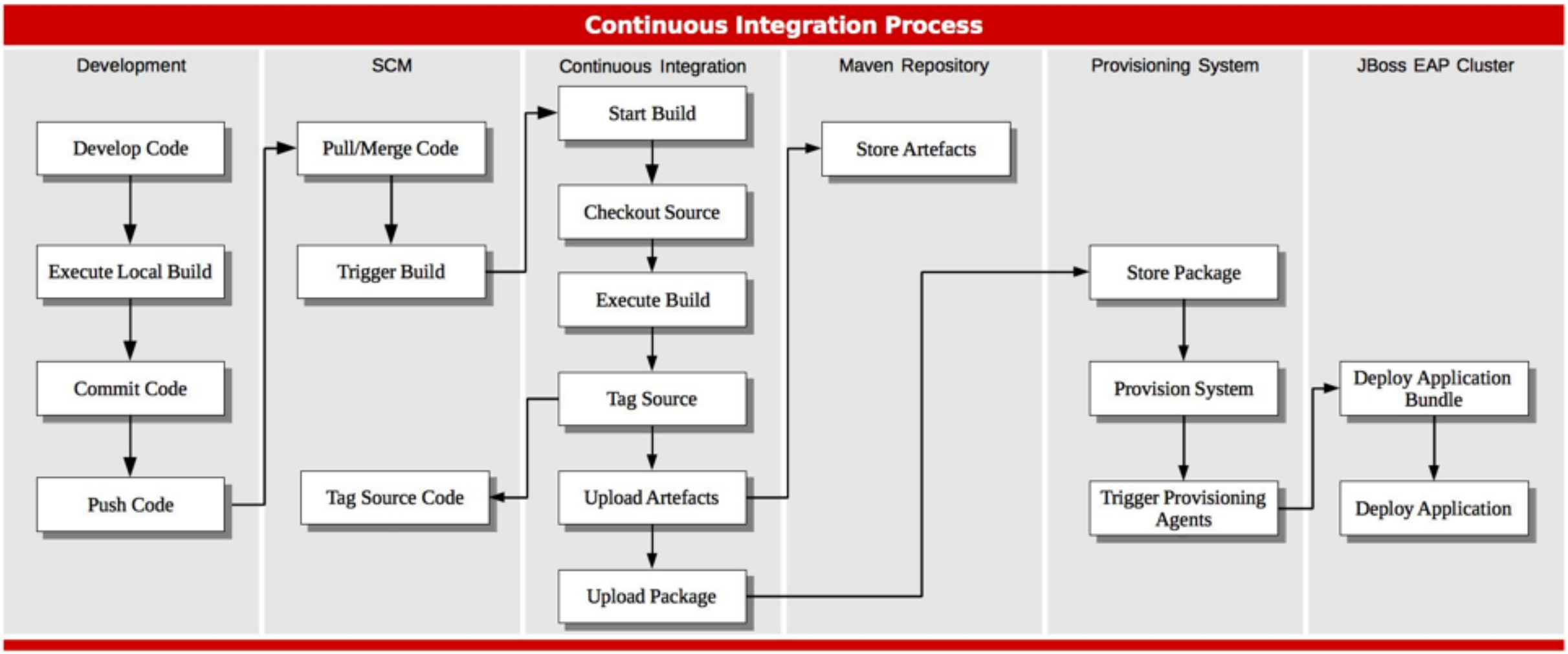
# Tenets of CI

- Automate the build - single command builds the entire project

- Track who? when? what?

# Tenets of CI

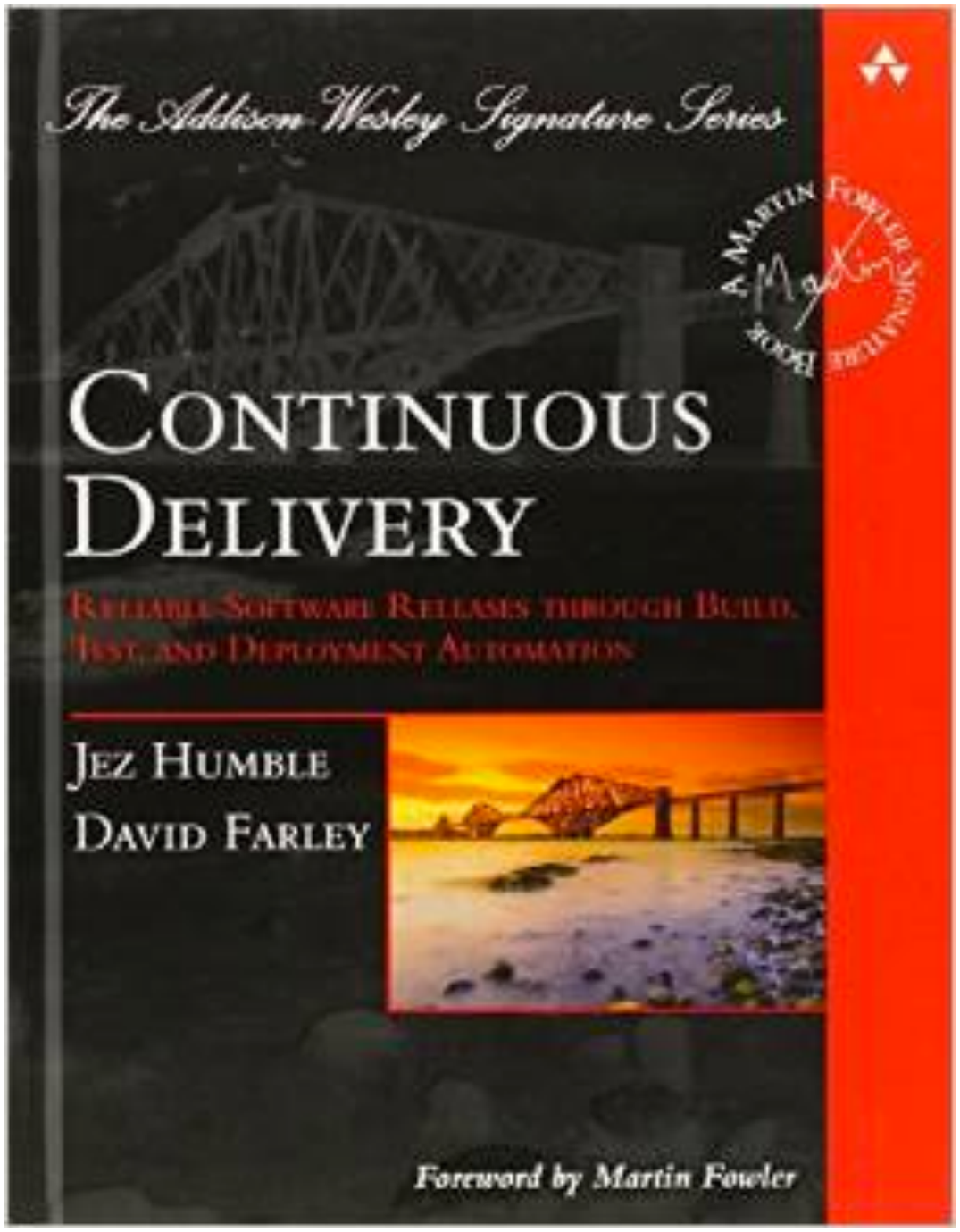
- Keep the build fast, rapid feedback (~10 mins)
- Commit early and often, once every few hours, at least once a day
- Check out, build, and test on a CI machine
- Nightly builds != CI


Continuous Integration Process
Development
SCM
Continuous Integration
Maven Repository
Provisioning System
JBoss EAP Cluster
Develop Code
Execute Local Build
Commit Code
Push Code
Pull/Merge Code
Trigger Build
Tag Source Code
Start Build
Checkout Source
Execute Build
Tag Source
Upload Artefacts
Upload Package
Store Artefacts
Store Package
Provision System
Trigger Provisioning Agents
Deploy Application Bundle
Deploy Application

*“Continuous Integrations doesn't get rid of bugs, but it does make them dramatically easier to find and remove”*

Martin Fowler

http://martinfowler.com/articles/continuousIntegration.html

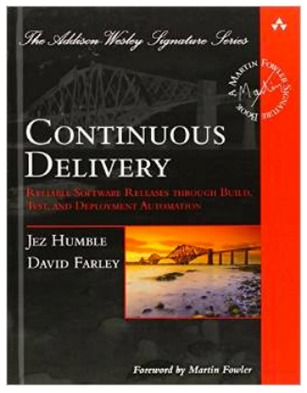



*"it is the practice of releasing every good build to users"*

*"continuous integration to its logical conclusion"*

# Deployment Pipeline

*“an automated implementation of your application’s build, deploy, test, and release process"*



# Aim of Deployment Pipeline

1. Make every part of the process visible to everybody involved, aiding collaboration

2. Improves feedback to identify/resolve problems, early in the process

3. Deploy/release any version of the software on demand, fully automated

# CD Maturity Model

- Application code in source code control?
- Automated builds?
- Tests? Are they automated? TDD?
- What is the criteria for committing into “master”?
- What does it take to mature?

# The Continuous Delivery Maturity Model

| | Base | Beginner | Intermediate | Advanced | Expert |
|---|---|---|---|---|---|
| Culture & Organization | • Prioritized work<br>• Defined and documented process<br>• Frequent commits | • One backlog per team<br>• Share the pain<br>• Stable teams<br>• Adopt basic Agile methods<br>• Remove boundary dev & test | • Extended team collaboration<br>• Component ownership<br>• Act on metrics<br>• Remove boundary dev & ops<br>• Common process for all changes<br>• Decentralize decisions | • Dedicated tools team<br>• Team responsible all the way to prod<br>• Deploy disconnected from Release<br>• Continuous improvement (Kaizen) | • Cross functional teams<br>• No rollbacks (always roll forward) |
| Design & Architecture | • Consolidated platform & technology | • Organize system into modules<br>• API management<br>• Library management<br>• Version control DB changes | • No (or minimal) branching<br>• Branch by abstraction<br>• Configuration as code<br>• Feature hiding<br>• Making components out of modules | • Full component based architecture<br>• Push business metrics | • Infrastructure as code |
| Build & Deploy | • Versioned code base<br>• Scripted builds<br>• Basic scheduled builds (CI)<br>• Dedicated build server<br>• Documented manual deploy<br>• Some deployment scripts exsists | • Polling builds<br>• Builds are stored<br>• Manual tag & versioning<br>• First step towards standardized deploys | • Auto triggered build (commit hooks)<br>• Automated tag & versioning<br>• Build once deploy anywhere<br>• Automated bulk of DB changes<br>• Basic pipeline with deploy to prod<br>• Scripted config changes (e.g. app server)<br>• Standard process for all environments | • Zero downtime deploys<br>• Multiple build machines<br>• Full automatic DB deploys | • Build bakery<br>• Zero touch continuous deployments |
| Test & Verification | • Automatic unit tests<br>• Separate test environment | • Automatic integration tests | • Automatic component tests (isolated)<br>• Some automatic acceptance tests | • Full automatic acceptance tests<br>• Automatic performance tests<br>• Automatic security tests<br>• Risk based manual testing | • Verify expected business value |
| Information & Reporting | • Baseline process metrics<br>• Manual reporting | • Measure the process<br>• Static code analysis<br>• Scheduled quality reports | • Common information model<br>• Traceability built into pipeline<br>• Report history is available | • Graphing as a service<br>• Dynamic test coverage analysis<br>• Report trend analysis | • Dynamic graphing and dashboards<br>• Cross silo analysis |

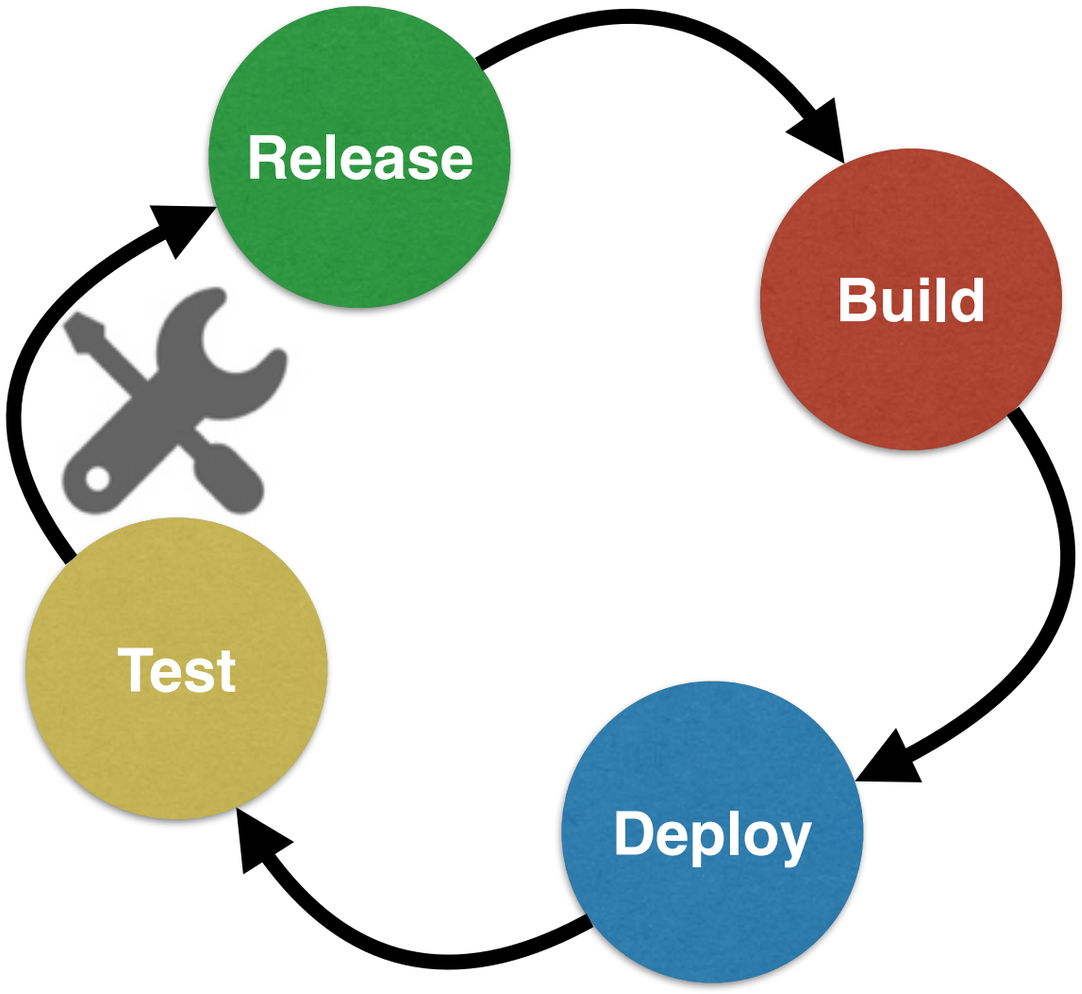
http://www.infoq.com/articles/Continuous-Delivery-Maturity-Model

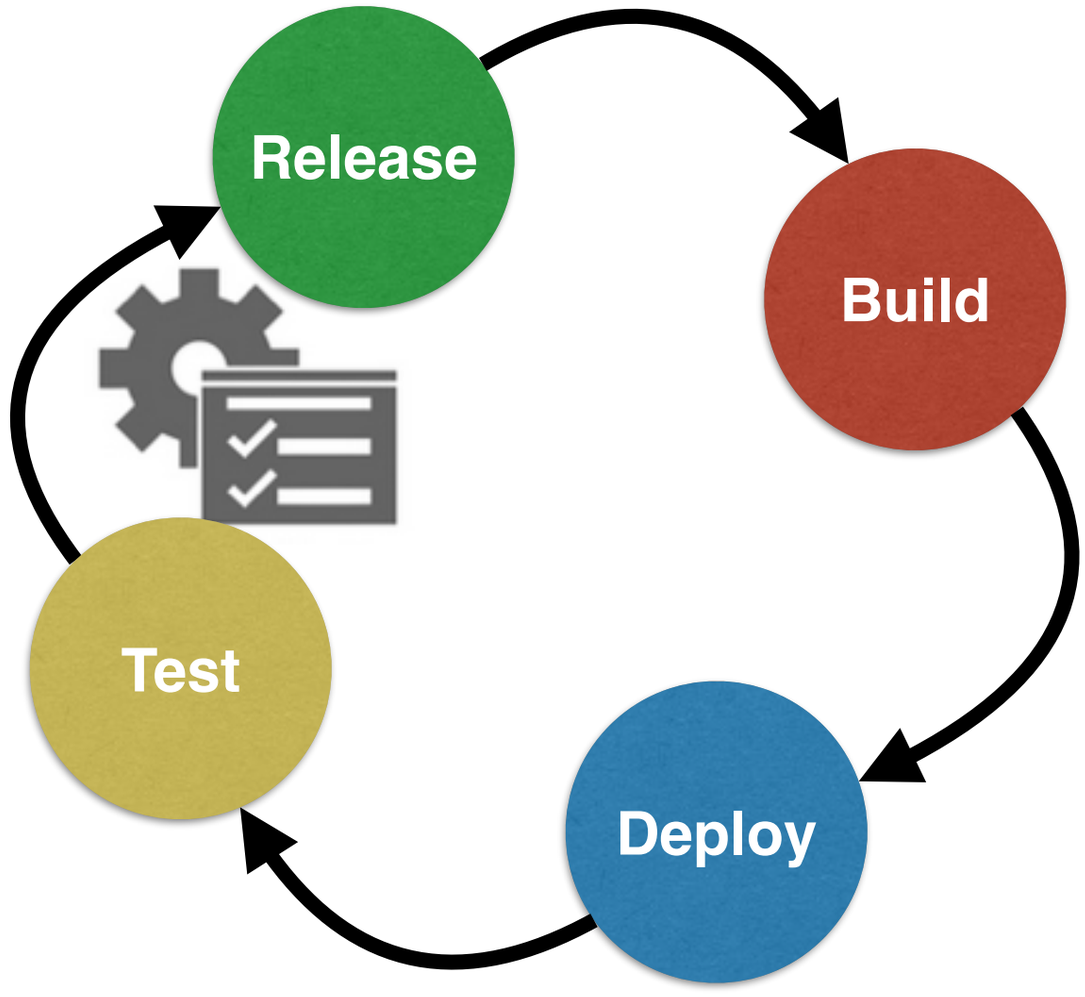
| | Initial | Managed | Defined | Quantitatively Managed | Optimizing |
|---|---|---|---|---|---|
| Culture & Organization | Teams organised based on platform/technology.<br>Defined and documented processes. | One backlog per team.<br>Adopt agile methodologies.<br>Remove team boundaries.<br>Share the pain. | Extended team collaboration.<br>Remove boundary dev/ops.<br>Common process for all changes. | Cross-team continuous improvement.<br>Teams responsible all the way to production. | Cross functional teams. |
| Build & Deploy | Centralized version control.<br>Automated scripts for building software.<br>Nightly builds.<br>No management of artifacts<br>Manual deployment.<br>Environments are manually provisioned. | Polling builds.<br>Any build can be re-created from source control.<br>Management of build artifacts.<br><br>Automated deployments<br>Automated provisioning of environments. | Triggered builds (commit hook).<br>Fail builds if quality is not met (code analysis, performance, etc.)<br><br>Push button deployment and release of any releasable artifact to any environment.<br>Standard deployment process for all environments | Team prioritises keeping codebase deployable over doing new work.<br>Builds are not left broken.<br><br>Orchestrated deployments.<br>Blue Green Deployments, | Zero touch Continuous Deployments. |
| Release | Infrequent and unreliable releases/ | Painful infrequent but reliable releases. | Infrequent but fully automated and reliable releases in any environment. | Frequent fully automated releases.<br>Deployment disconnected from release.<br>Canary releases. | No rollbacks, always roll forward. |
| Data Management | Data migrations unversioned and performed manually. | Automated and versioned changes to datastores. | Changes to datastores automatically performed as part of the deployment process. | Automatic datastore changes and rollbacks tested with every deployment. | |
| Test & Verification | Automated Unit tests on every build.<br>Separate test environment | Automatic Integration Tests.<br>Code analysis.<br>Test coverage analysis. | Automatic component tests.<br>(Some) automatic integration tests. | Full automatic acceptance tests.<br>Automatic performance tests.<br>Automatic security tests.<br>Manual exploratory testing based on risk based testing analysis. | Verify expected business value.<br>Defects found and fixed immediately (roll forward). |
| Information & Reporting | Baseline process metrics.<br>Manual reporting. | Measure the process.<br>Static code analysis.<br>Automatic reporting. | Automatic generation of release notes.<br>Pipeline traceability<br>Reporting history | Report trend analysis.<br>Real time graphs on deployment pipeline metrics. | Dynamic self-service of information.<br>Customizable dashboards.<br>Cross-reference across organizational boundaries. |

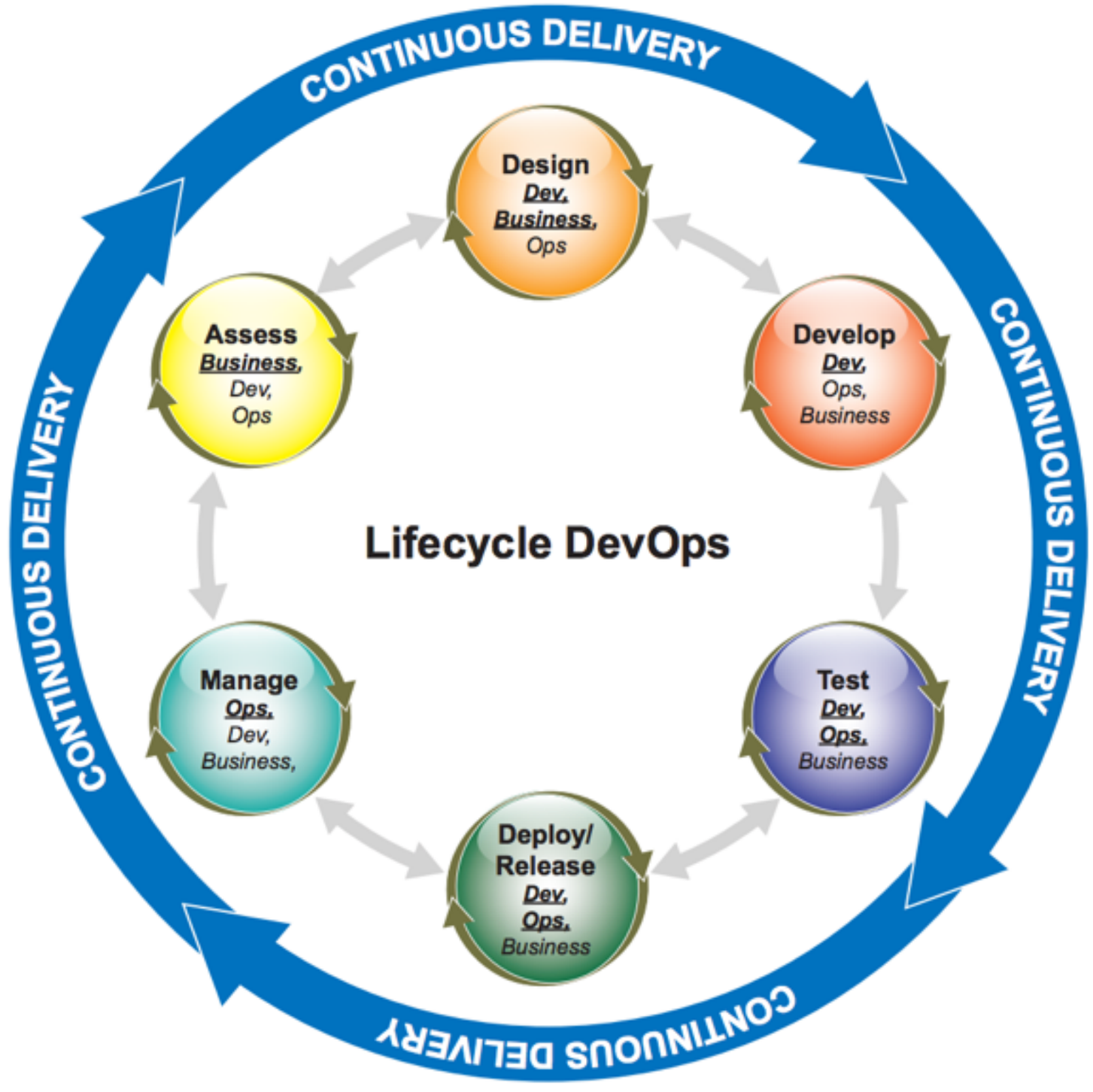
# CD Maturity

Deploy Frequency $\propto$ DevOps Maturity

# Continuous Deployment






http://pages.cloudbees.com/rs/cloudbees/images/CloudBees-EMA-DevOps_2014.pdf


- 1000 releases/day
- No Chef/Puppet or QA/Release team
- 70% instances changed in 2 weeks, 90% every month
- Deployment **fully** automated

https://www.youtube.com/watch?v=WvpmiGURFYQ

NETFLIX | OSS

- **Aminator**: Create AMI from base AMI, include code + configuration (can be extended for other clouds)
- **Asgard**: Deploys images to cloud, manages resources (~Kubernetes)
- Canary analysis with red/black push
- **Glisten**: Workflow for automated deployment
- **Hysterix**: Builds resiliency, graceful fallback
- **Simian Army**: test resilience at runtime
  - Chaos, Latency, Janitor, Conformity, …



http://netflix.github.io/

- Switched from self-hosted data center to AWS
- New update every 11.6 seconds on weekdays
  - Peak was 3x (1079 deployments in an hour)
  - Picked up by 10k different hosts
  - 0.0001% deployments resulted in outage
- Apollo - “secret sauce” for rolling updates
  - aws.amazon.com/codedeploy/

http://www.allthingsdistributed.com/2014/11/apollo-amazon-deployment-engine.html
http://servicevirtualization.com/profiles/blogs/making-the-transition-to-a-devops-culture


Graphite · cacti · New Relic. · Chartbeat · Nagios

node · python · php · Java · Ruby

ASGARD · amazon web services · ubuntu · puppet labs

Scott VanDenPlas
@scottvdp
Follow

4Gb/s, 10k req/s, 2k nodes, 3 datacenters, 180TB and 8.5 billion req. Design, deploy, dismantle in 583 days to elect the President. #madops

http://arstechnica.com/information-technology/2012/11/how-team-obamas-tech-efficiency-left-romney-it-in-dust/

# DevOps with Java EE





# WildFly

- Red Hat's open source Java EE 7 compliant application server

- 8.0 (Feb 2014), 8.1 released (May 2014), 8.2 (Nov 2014), 9.0 next year

- Commercial support coming in JBoss Enterprise Application Platform (EAP) 7

JBoss Tools

# JBoss Tools

- Eclipse plugins for JBoss technology









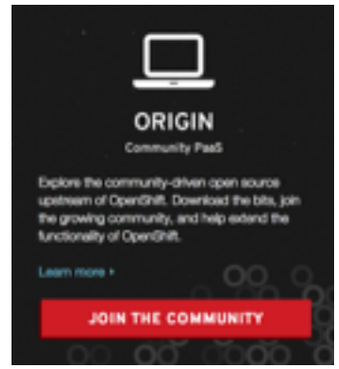


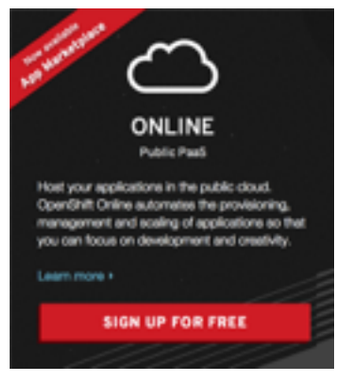
# Forge

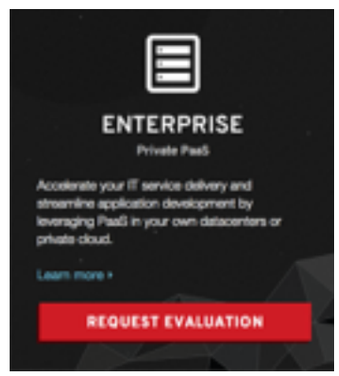
- Fastest way to build Maven-based Java EE projects
- WORAI for Java EE (using Maven)
  - Aids in automation of repetitive tasks
  - Generation of boilerplate code
  - In-built support for Java EE 6 and 7
- Extensible by design
- Test-driven development

# OpenShift

- OpenShift Origin, Online, Enterprise
- Pluggable cartridges
- Multiple languages, databases, app servers
- Gears: small, medium, large
  - FREE account: 3 gears (1 GB disk, 0.5 GB RAM)



# Arquillian

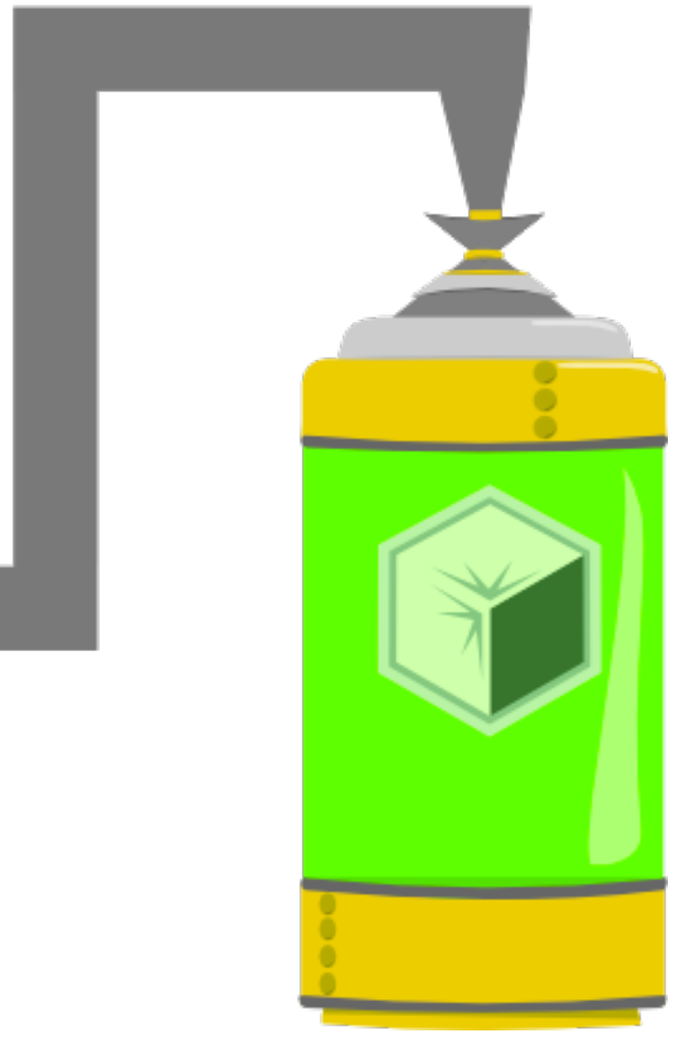
Highly **extensible testing platform** for the JVM that enables to easily create **automated tests**

# 1 Select a container

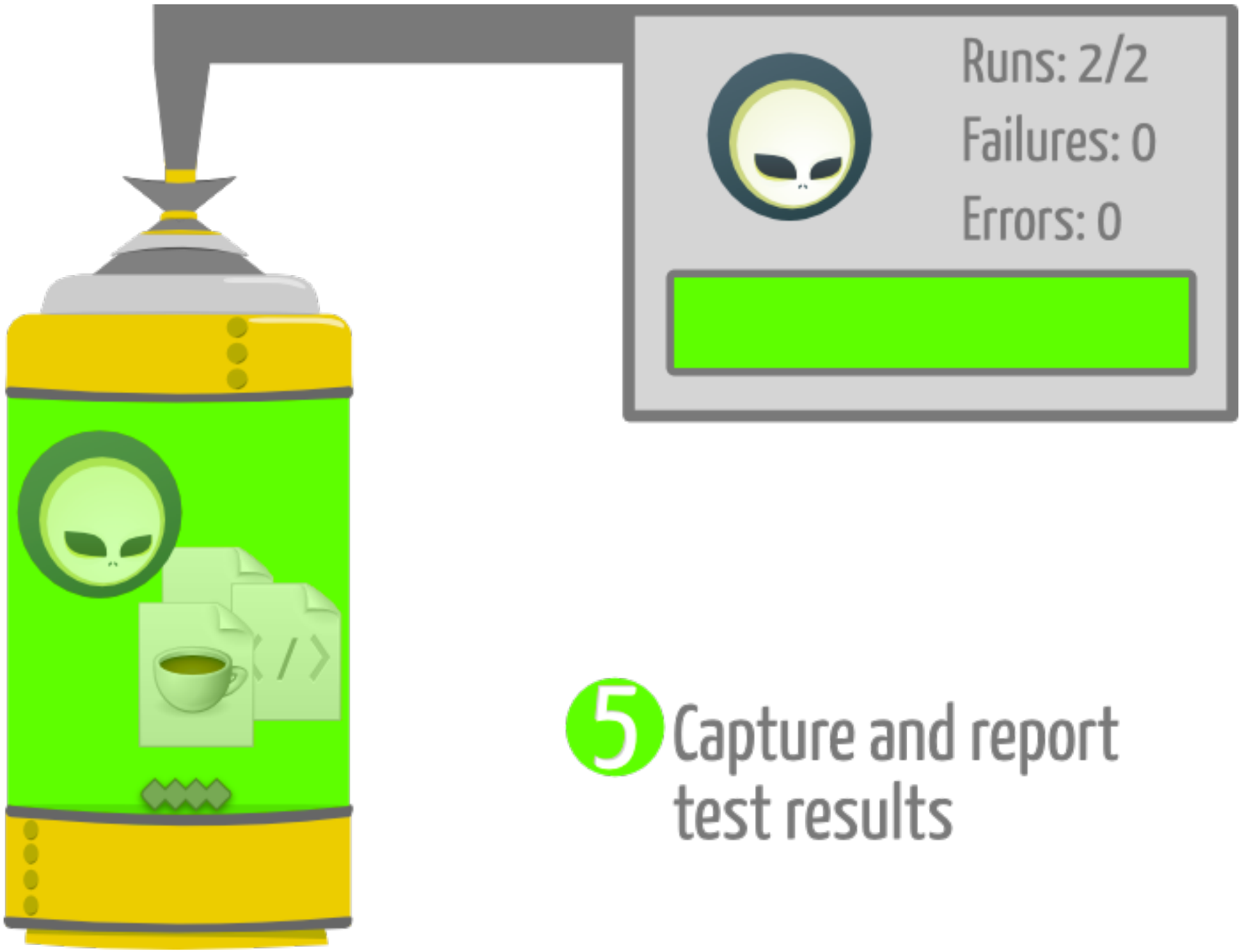
# 2 Start or connect to a container

Arquillian

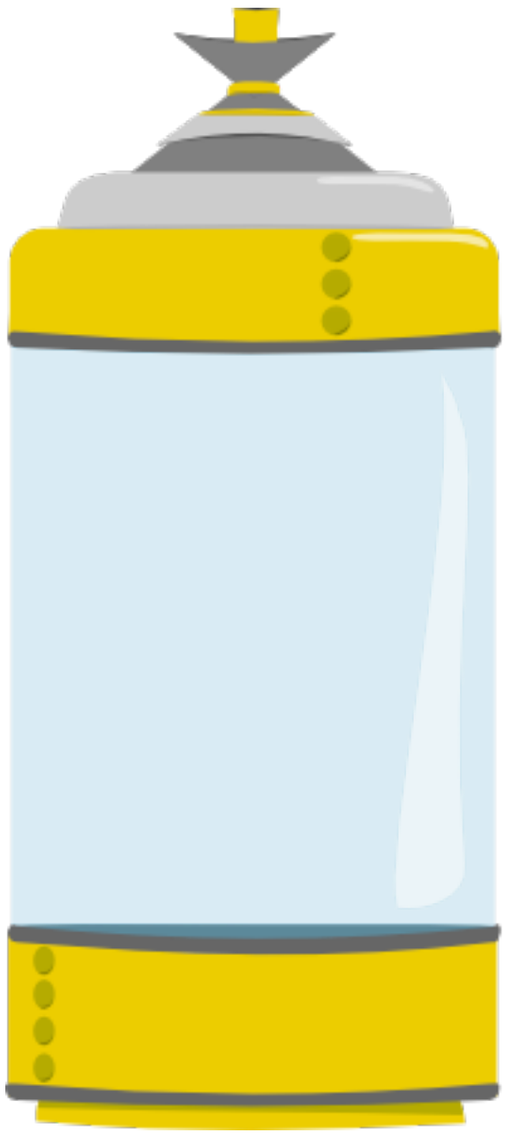
## 3 Package test archive and deploy to container



# 4 Run test in-container




## 5 Capture and report test results



6 Undeploy test archive and stop or disconnect from container

# Getting Started with Arquillian

- Testable Java EE 7 application

```
mvn
--batch-mode
archetype:generate
-DarchetypeGroupId=org.javaee-samples
-DarchetypeArtifactId=javaee7-arquillian-archetype
-DgroupId=org.samples.javaee7.arquillian
-DartifactId=arquillian
```

# Testable Java EE 7 App

- Generated profiles
  - wildfly-remote-arquillian
  - wildfly-managed-arquillian
  - glassfish-remote-arquillian
  - glassfish-embedded-arquillian

# Test on WildFly

- Download WildFly 8.1 from wildfly.org
- Unzip and start as `bin/standalone.sh`
- Run the test as:

```
mvn test -Pwildfly-remote-arquillian
```

# Enable Arquillian on Existing Java EE Projects

- Using Forge

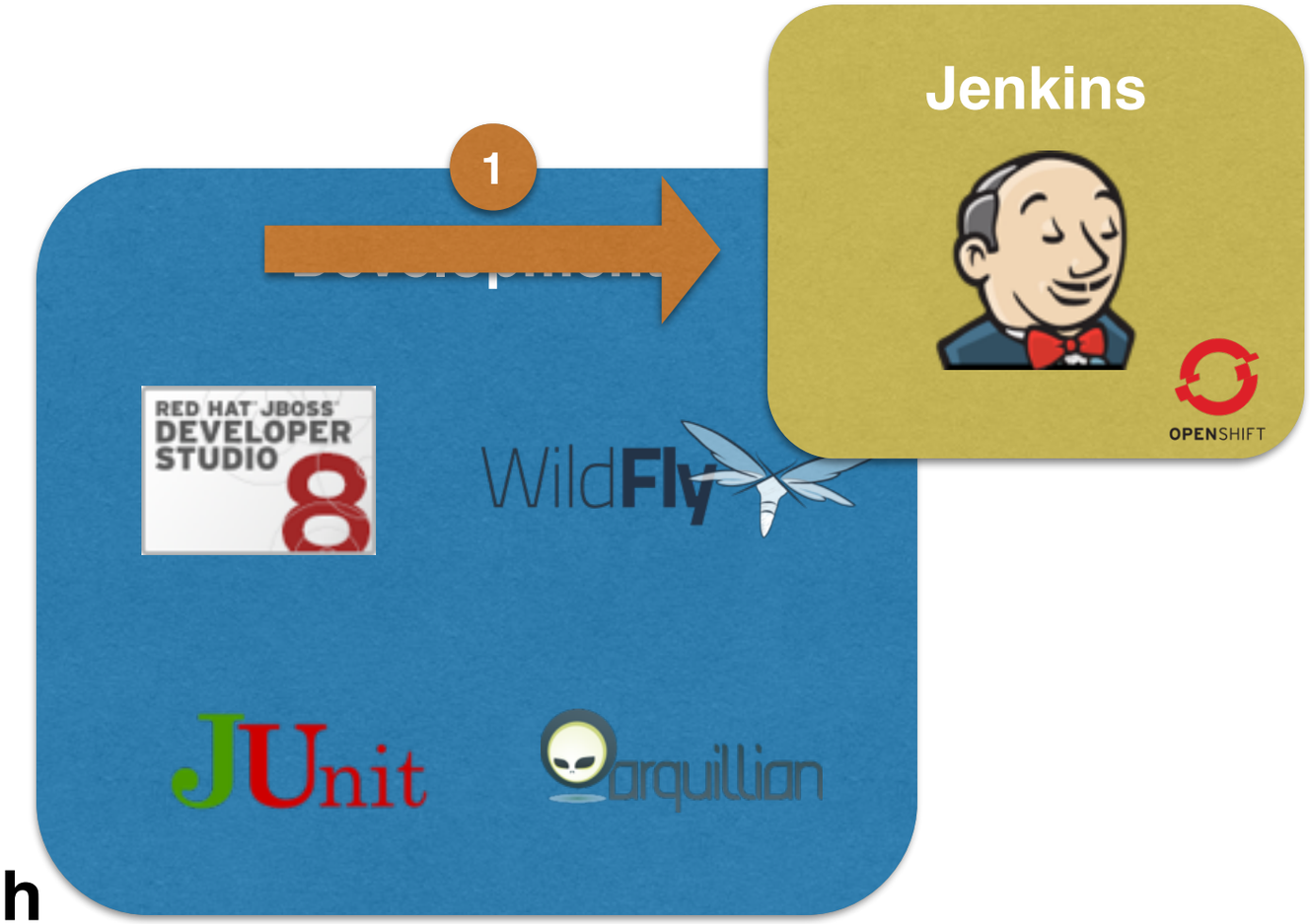
  - addon-install-from-git --url https://github.com/forge/addon-arquillian.git

  - arquillian-setup --testFramework junit --containerAdapter wildfly-remote

# Stage 1: Test Driven Development

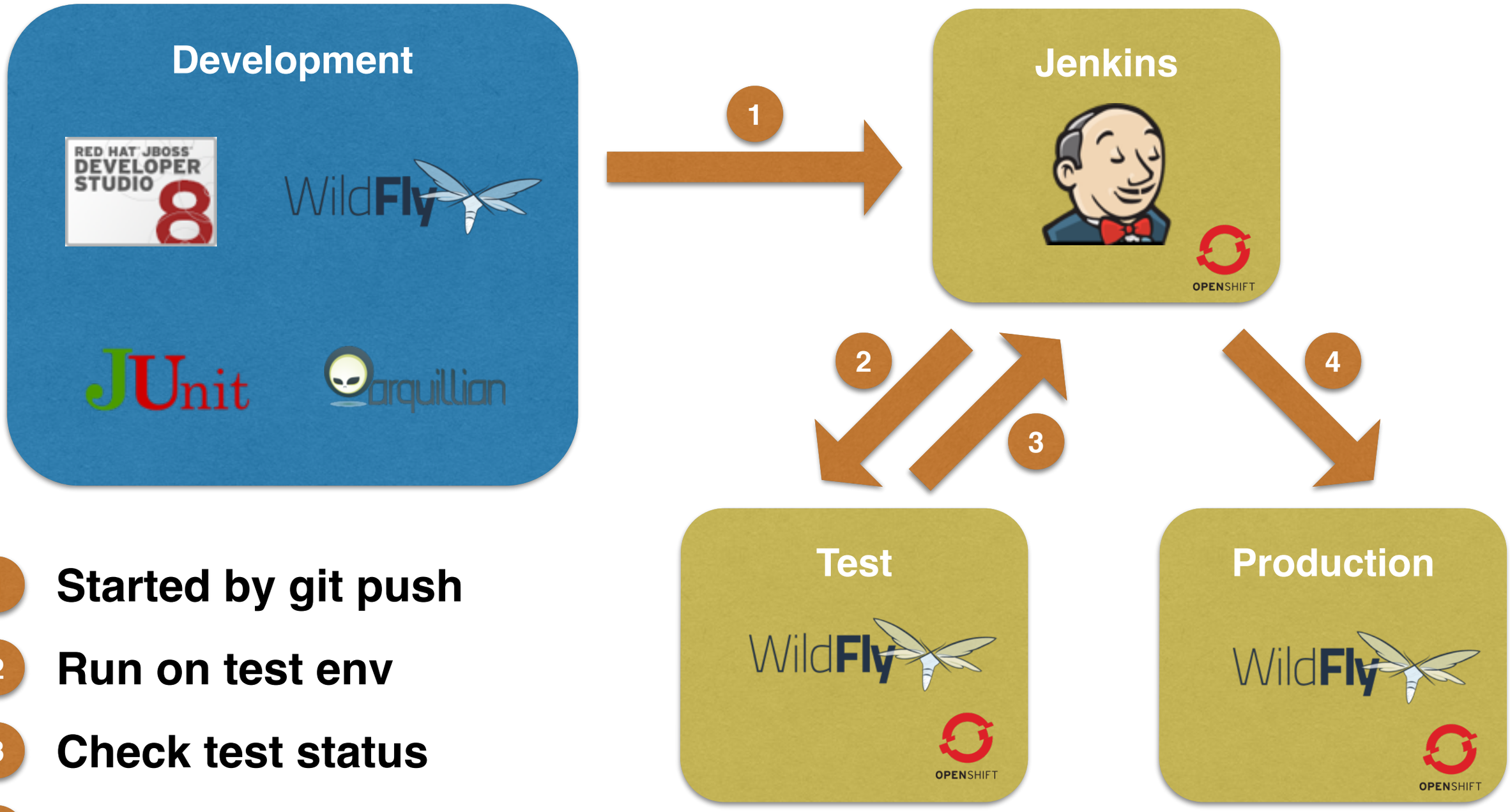

Code review
Code coverage
APM

# Stage 2: Continuous Integration

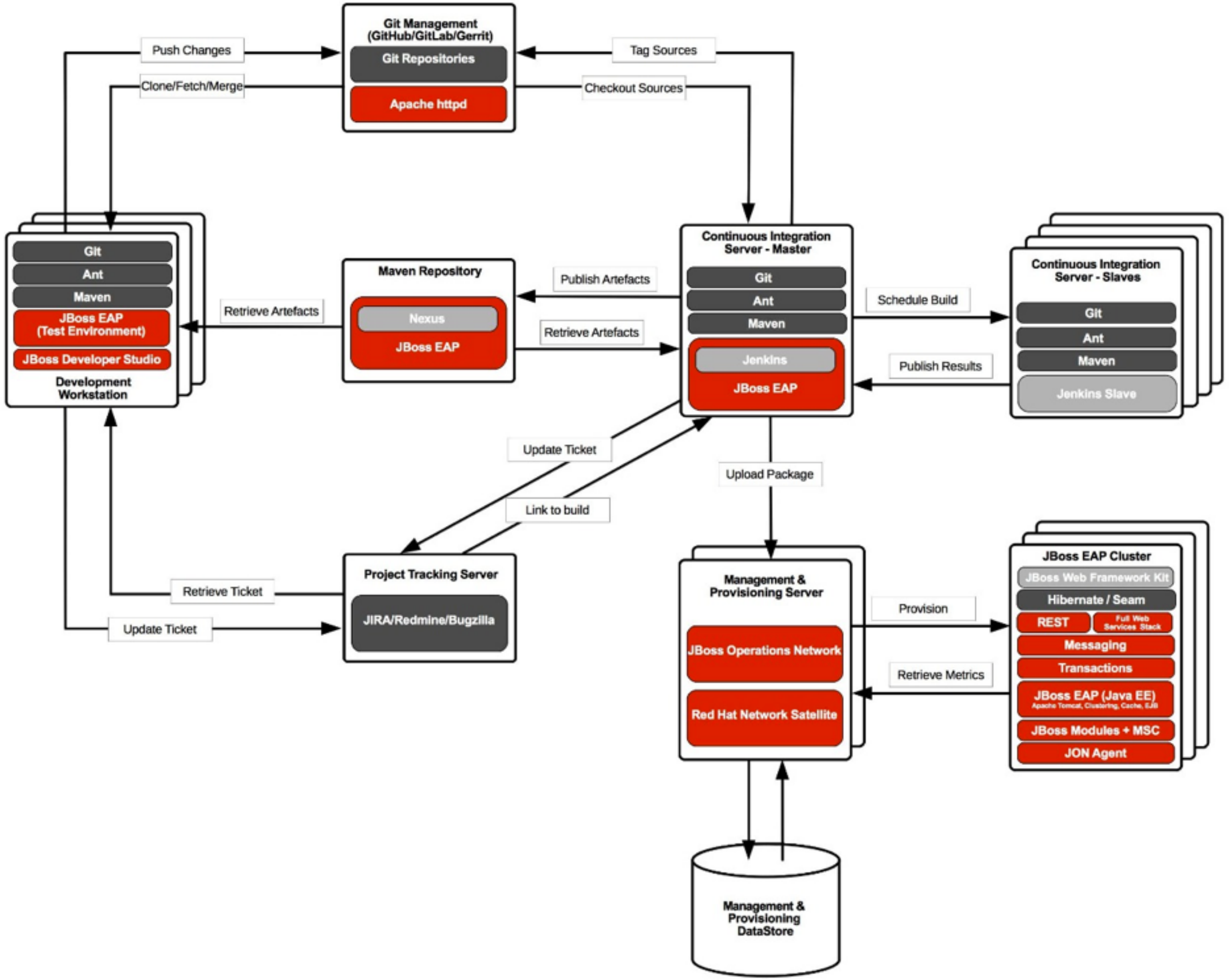



1 **Started by git push**

# Stage 3: Automated




1. **Started by git push**
2. **Run on test env**
3. **Check test status**
4. **Push changes to prod**


Git Management
(GitHub/GitLab/Gerrit)
Git Repositories
Apache httpd
Push Changes
Clone/Fetch/Merge
Tag Sources
Checkout Sources
Git
Ant
Maven
JBoss EAP
(Test Environment)
JBoss Developer Studio
Development
Workstation
Retrieve Artefacts
Maven Repository
Nexus
JBoss EAP
Publish Artefacts
Retrieve Artefacts
Continuous Integration
Server - Master
Git
Ant
Maven
Jenkins
JBoss EAP
Schedule Build
Publish Results
Continuous Integration
Server - Slaves
Git
Ant
Maven
Jenkins Slave
Update Ticket
Link to build
Upload Package
Project Tracking Server
JIRA/Redmine/Bugzilla
Retrieve Ticket
Update Ticket
Management &
Provisioning Server
JBoss Operations Network
Red Hat Network Satellite
Provision
Retrieve Metrics
JBoss EAP Cluster
JBoss Web Framework Kit
Hibernate / Seam
REST
Full Web
Services Stack
Messaging
Transactions
JBoss EAP (Java EE)
Apache Tomcat, Clustering, Cache, EJB
JBoss Modules + MSC
JON Agent
Management &
Provisioning
DataStore

# References

- github.com/arun-gupta/devops