Using OpenShift & PaaS to accelerate DevOps & Continuous Delivery

Andrea Morena, @andreamorena5
Arun Gupta, @arungupta

Andrea Morena

Senior Solution Architect

@andreamorena5

amorena@redhat.com

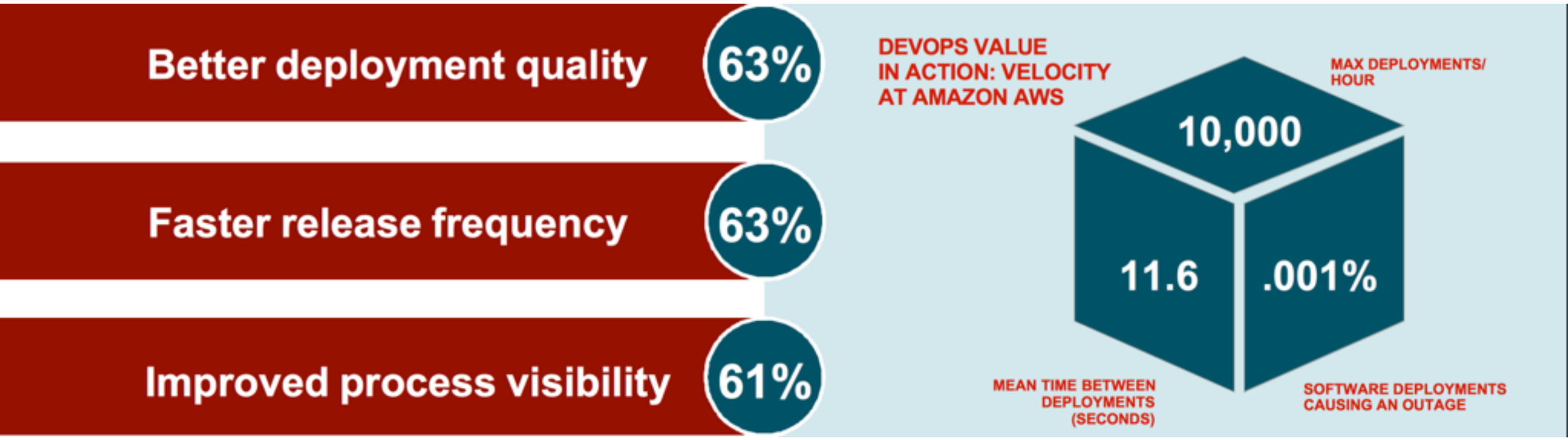
Arun Gupta

Director, Developer Advocacy & Technical Marketing

@arungupta
blog.arungupta.me
arungupta@redhat.com

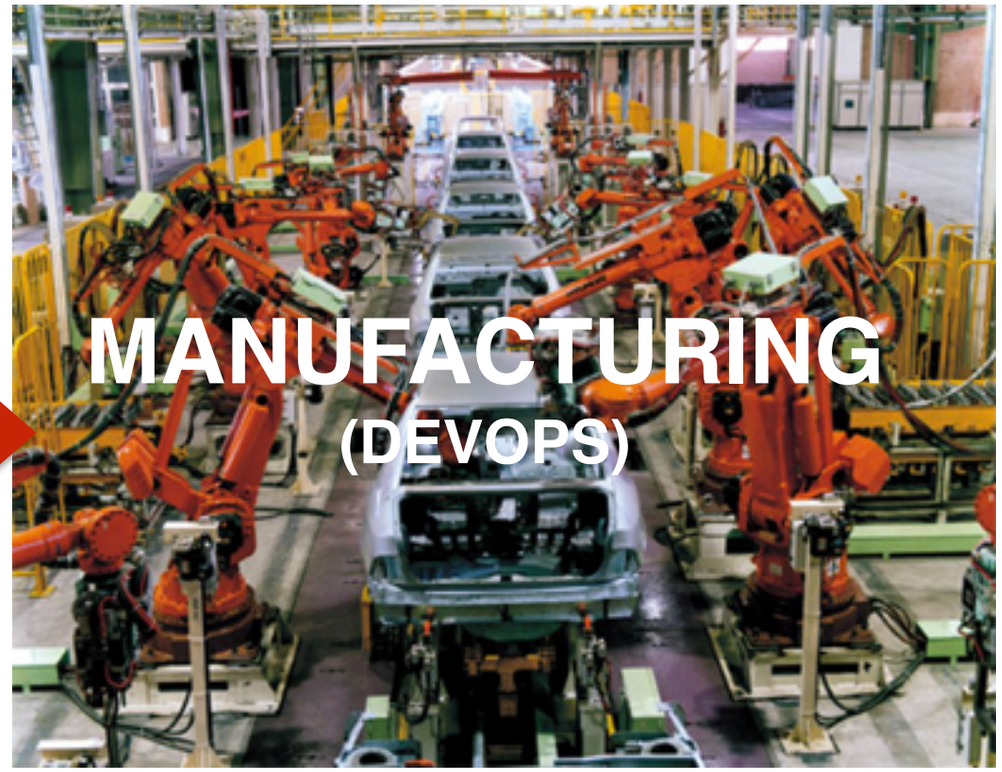

WORKED FINE IN DEV
OPS PROBLEM NOW
memegenerator.net

Organizations implementing DevOps



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

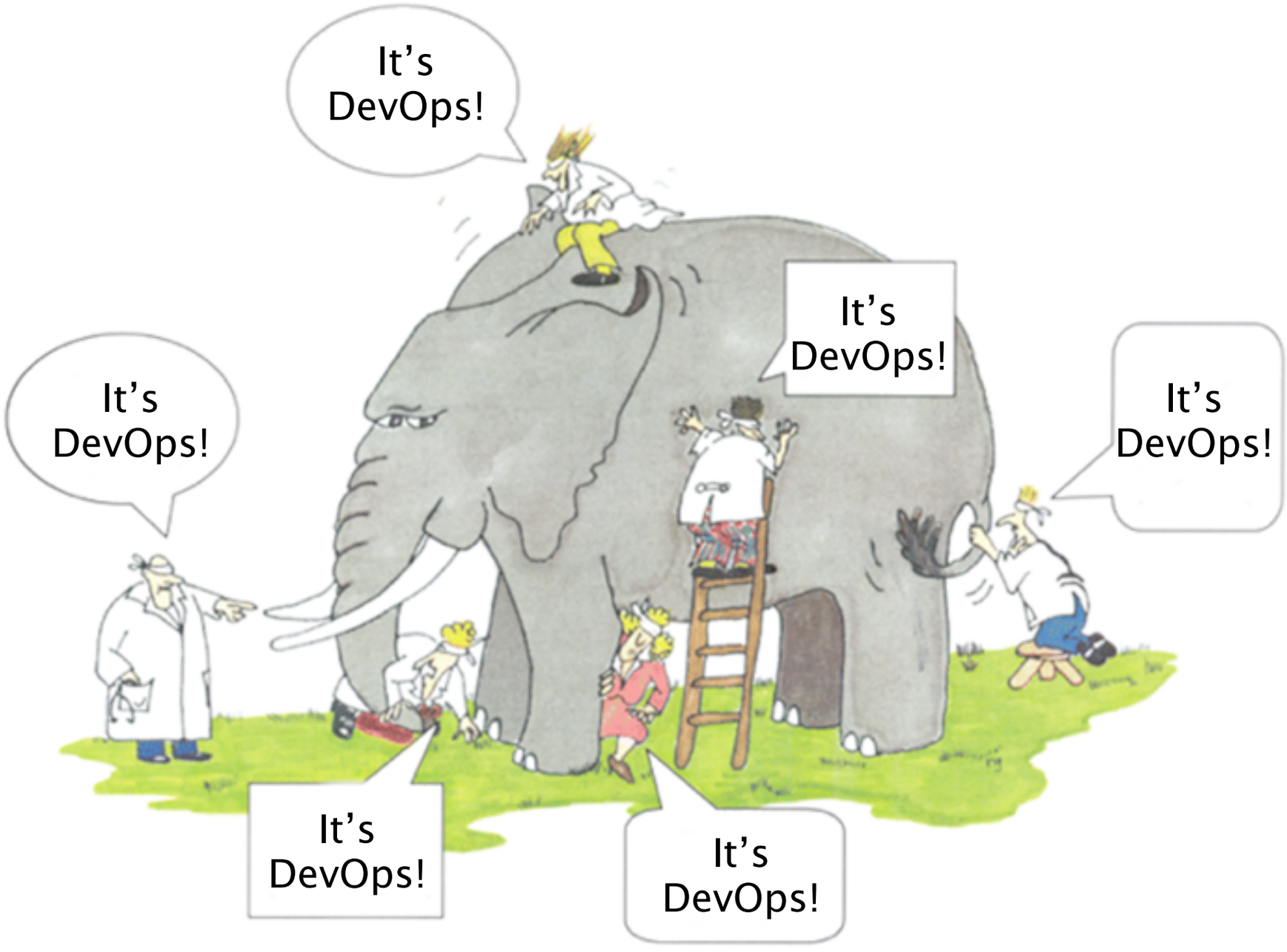

CRAFTWORK


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/


STRIKE
GRAD YEAR
TRAVEL
MARRIAGE
KIDS
DIVORCE
I'm BACK!
DIRTY LAUNDRY

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

Collaboration

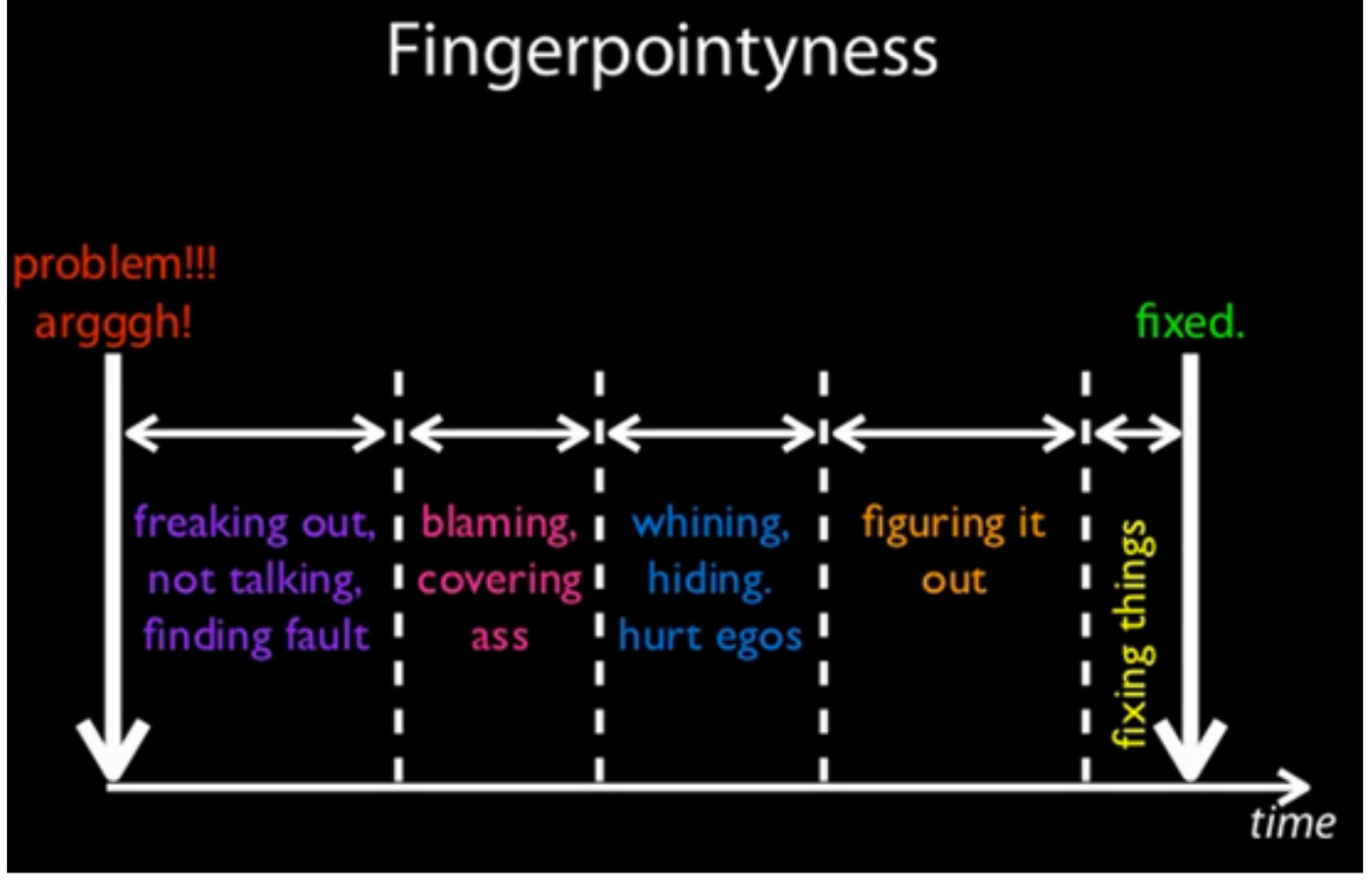
- "Dev"
  - Engineering
  - Test
  - Product management

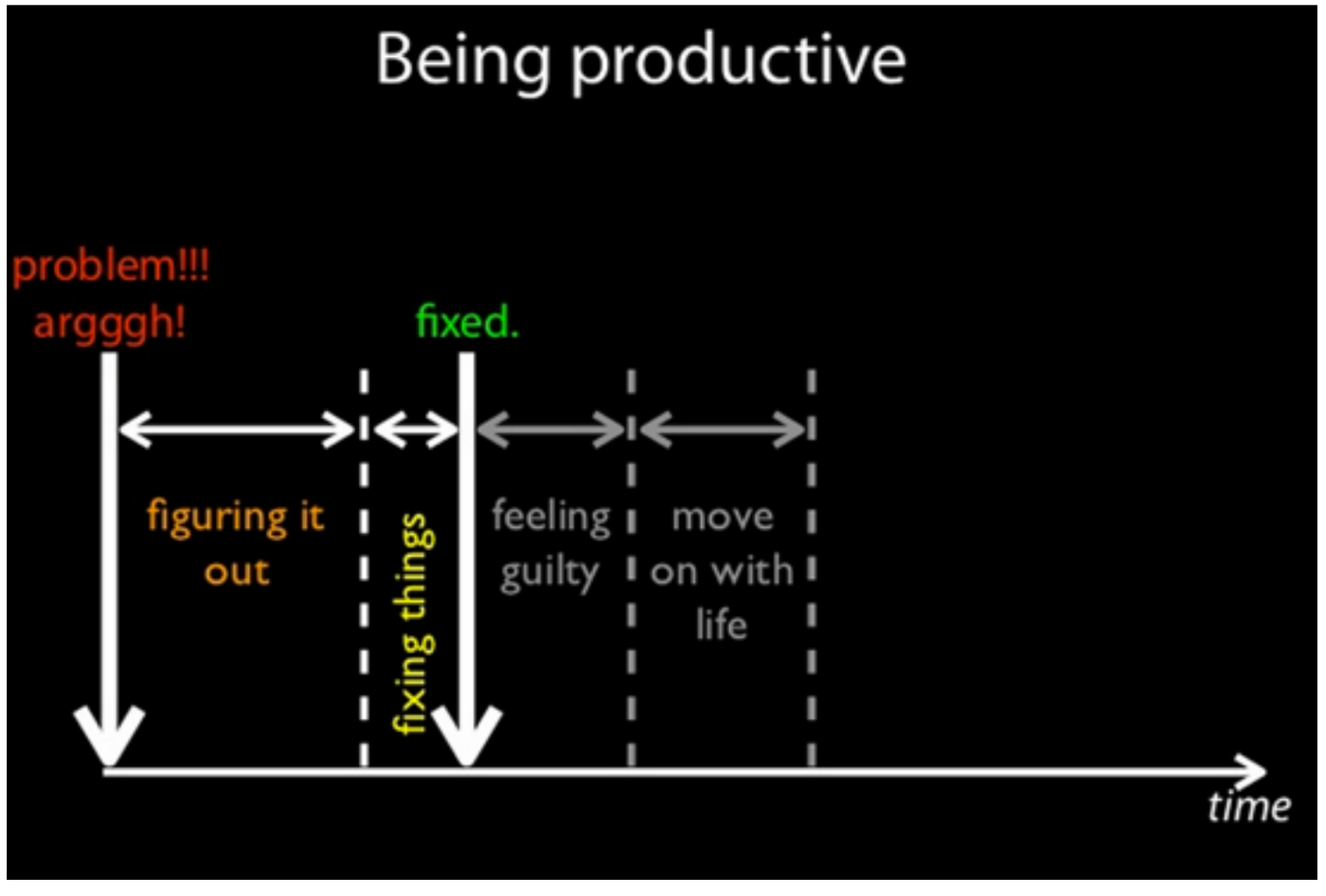
- "Ops"
  - System administrators
  - Operations staff
  - DBAs
  - Network engineers
  - Security professionals


“Dev” “Ops”

Culture

- Respect other's expertise, opinions, responsibilities
- Trust: Ops to think like devs, vice versa
  - 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!”

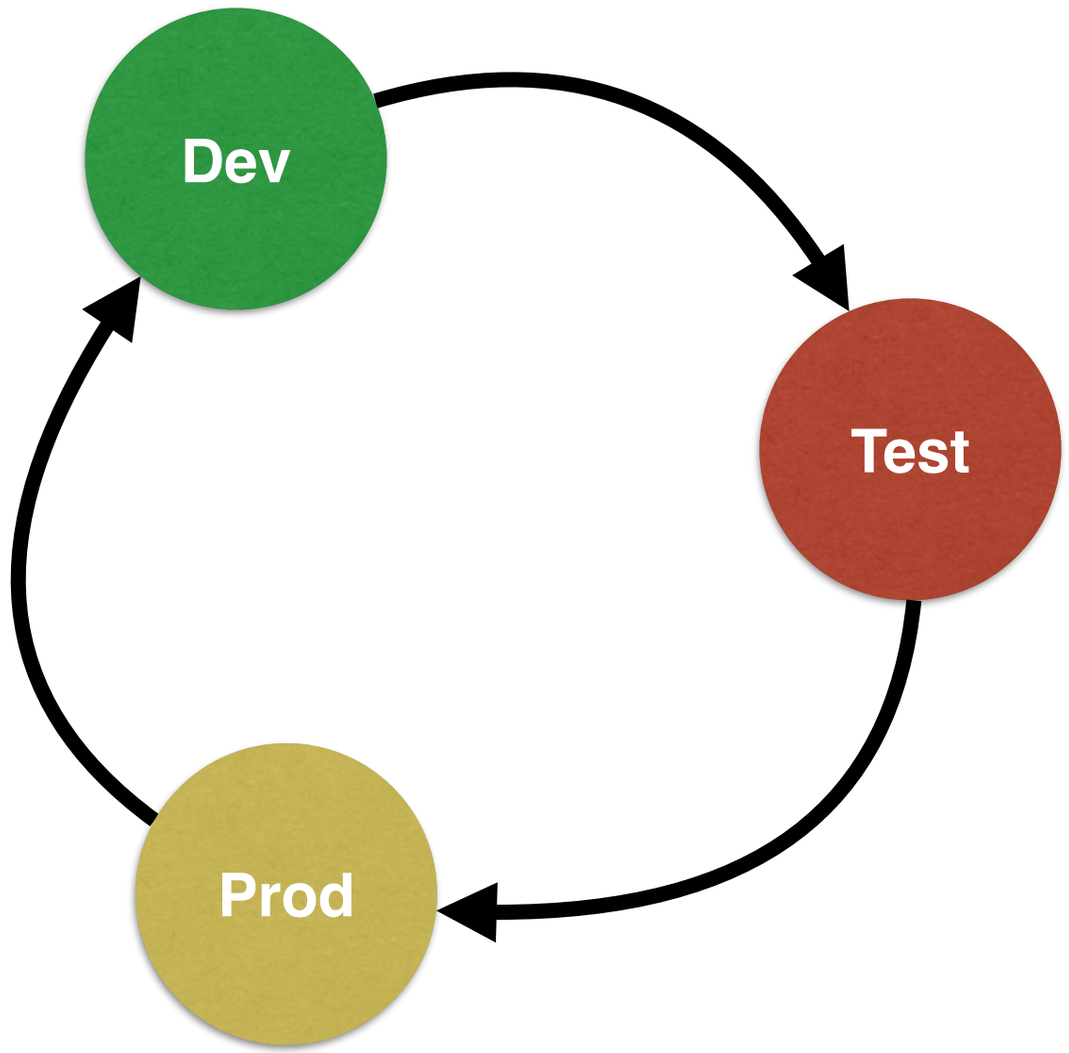
Code everything

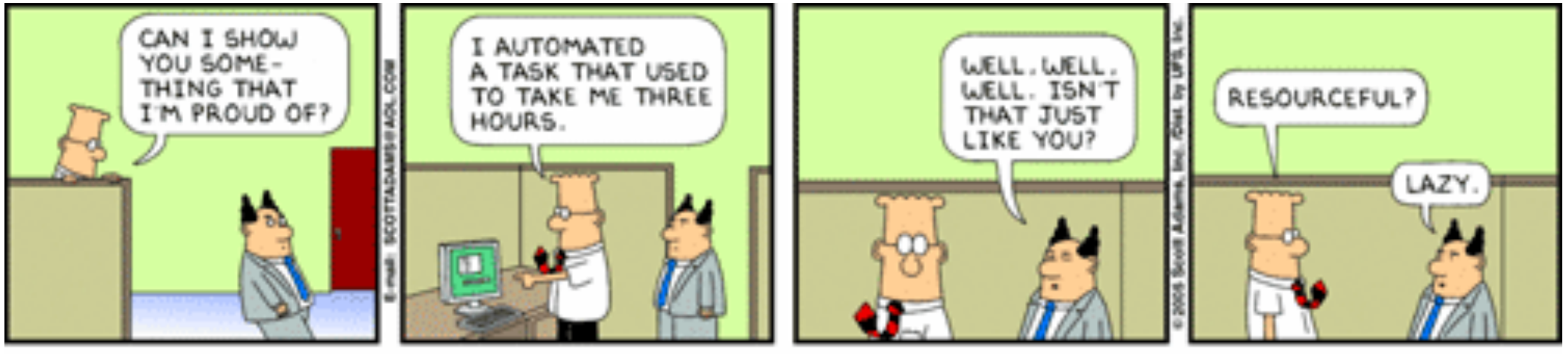
- Application code
- Build scripts
- Database schema
- Configuration files
- IDE configurations
- Infrastructure
- Deployment scripts
- Test code and scripts
- Provisioning scripts
- Monitoring
- Logging
- …

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

Consistency

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



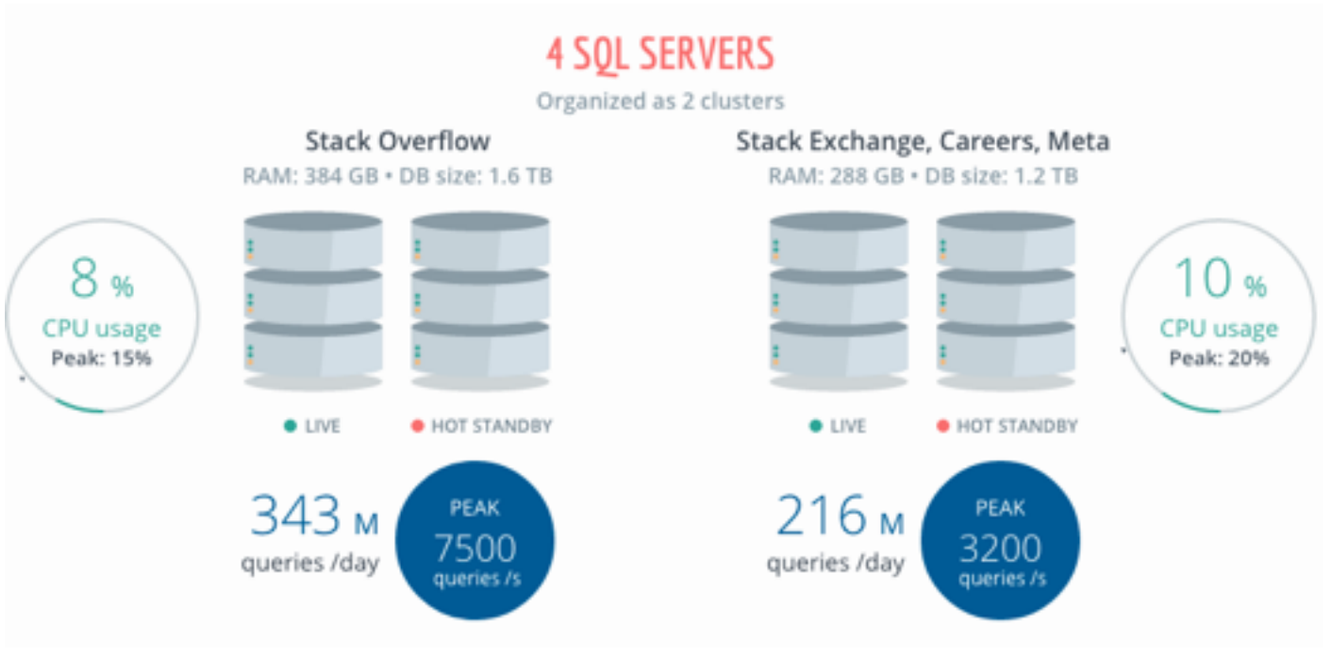

CAN I SHOW YOU SOME-THING THAT I'M PROUD OF?
I AUTOMATED A TASK THAT USED TO TAKE ME THREE HOURS.
WELL. WELL. WELL. ISN'T THAT JUST LIKE YOU?
RESOURCEFUL?
LAZY.
DID YOU JUST TURN MY BRILLIANT ACCOMPLISHMENT INTO A CHARACTER FLAW?
COMPLAINER.
LETS JUST FORGET THE WHOLE THING.
QUITTER.

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, …

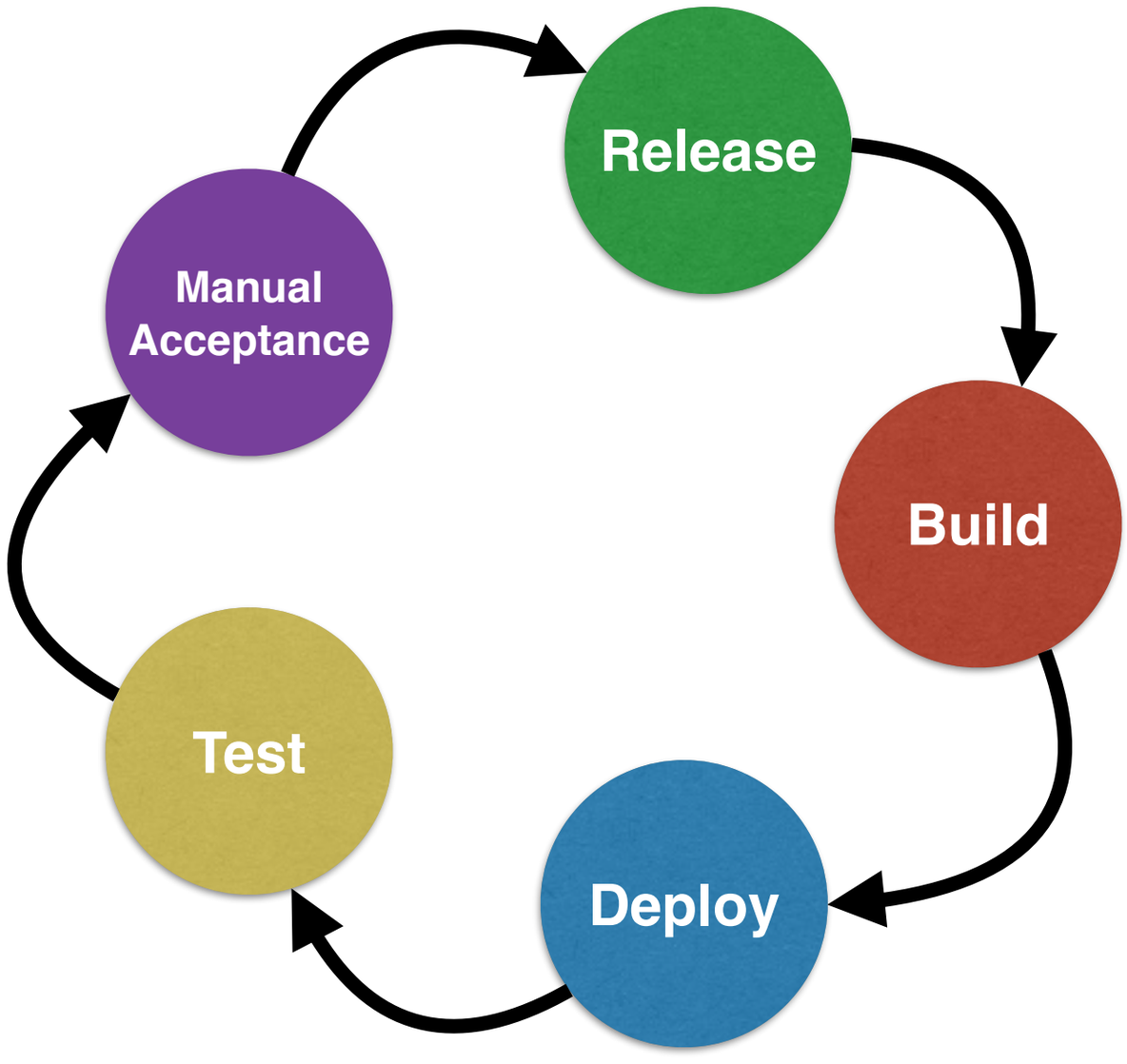
Dashboards

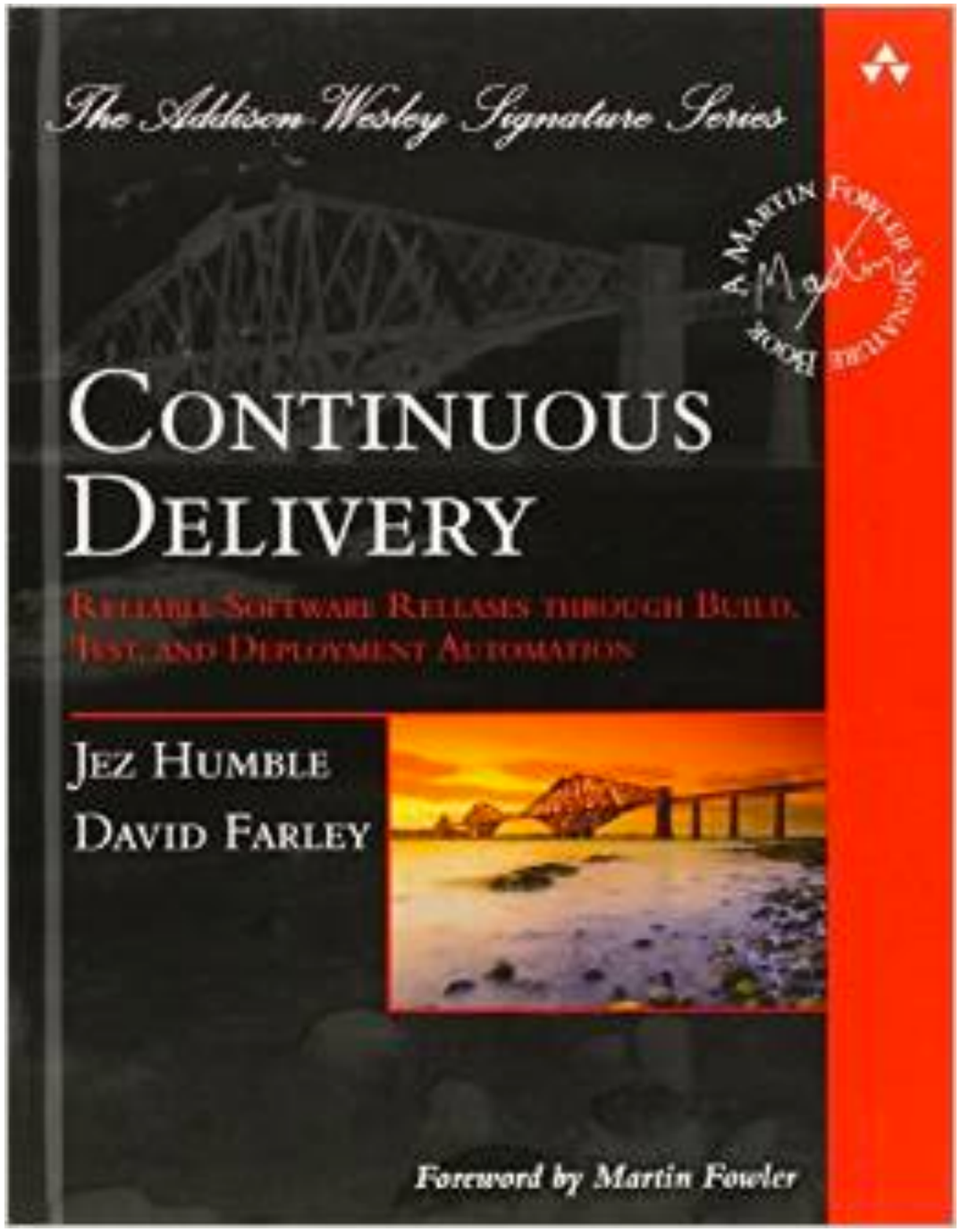
- Build dashboards, improve transparency
- stackexchange.com/performance



Continuous Delivery

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



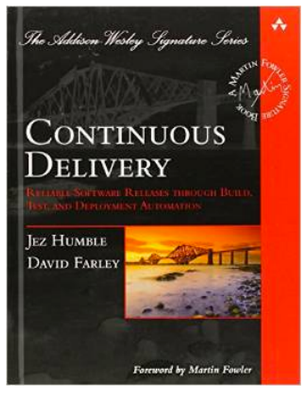


"it is the practice of releasing every good build to users"

"continuous integration to its logical conclusion"

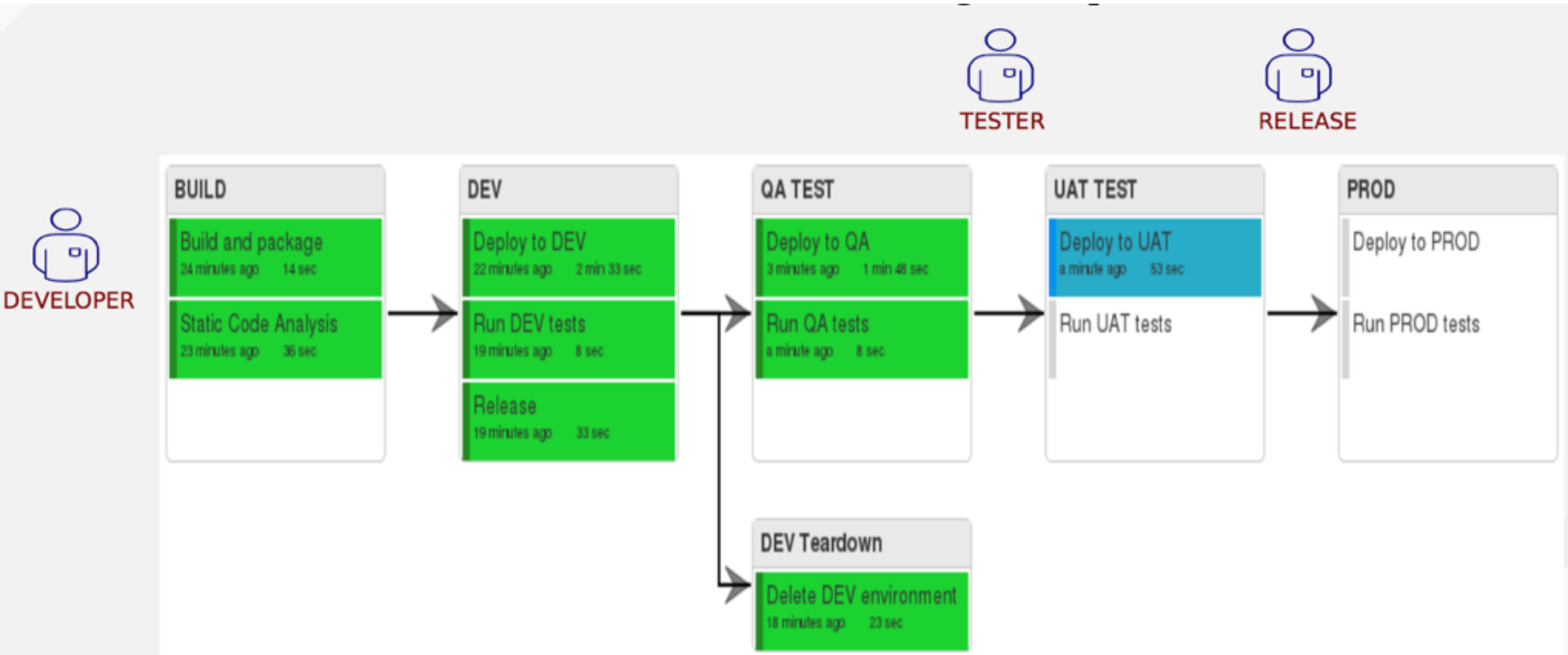
Deployment Pipeline

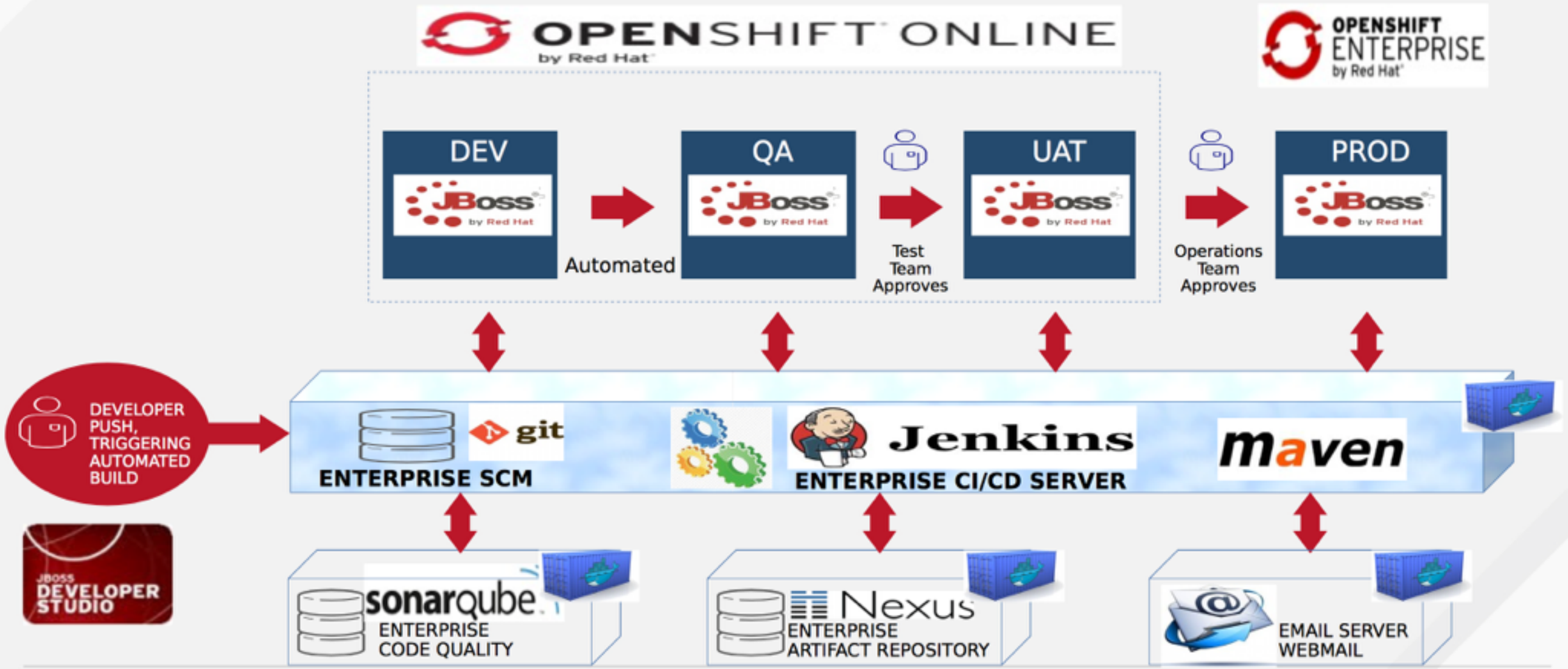
“an automated implementation of your application’s build, deploy, test, and release process"



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

http://blog.arungupta.me/continuous-integration-delivery-deployment-maturity-model/


TESTER
RELEASE
DEVELOPER
BUILD
Build and package
24 minutes ago 14 sec
Static Code Analysis
23 minutes ago 36 sec
DEV
Deploy to DEV
22 minutes ago 2 min 33 sec
Run DEV tests
19 minutes ago 8 sec
Release
19 minutes ago 33 sec
QA TEST
Deploy to QA
3 minutes ago 1 min 48 sec
Run QA tests
a minute ago 8 sec
UAT TEST
Deploy to UAT
a minute ago 53 sec
Run UAT tests
PROD
Deploy to PROD
Run PROD tests
DEV Teardown
Delete DEV environment
18 minutes ago 23 sec


OPENSHIFT ONLINE
by Red Hat
OPENSHIFT ENTERPRISE
by Red Hat
DEV
QA
UAT
PROD
JBoss
by Red Hat
Automated
Test Team Approves
Operations Team Approves
DEVELOPER PUSH, TRIGGERING AUTOMATED BUILD
JBOSS DEVELOPER STUDIO
git
ENTERPRISE SCM
Jenkins
ENTERPRISE CI/CD SERVER
maven
sonarqube
ENTERPRISE CODE QUALITY
Nexus
ENTERPRISE ARTIFACT REPOSITORY
EMAIL SERVER WEBMAIL


redhat.
RED HAT
SUMMIT
SUMMIT BY DAY
PARTY BY NIGHT
JOIN OUR JBOSS,
OPENSHIFT,
AND MOBILE TEAMS
FOR A NIGHT OF GAMES, DANCING,
AND OPEN CONTAINERS
Visit the Red Hat booth
in Hall D for location
and invitation.

References

- github.com/javaee-samples/devops