THEAGILEADMIN.COM

INTRO TO DEVOPS

ALIEN VAULT

@ERNESTMUELLER

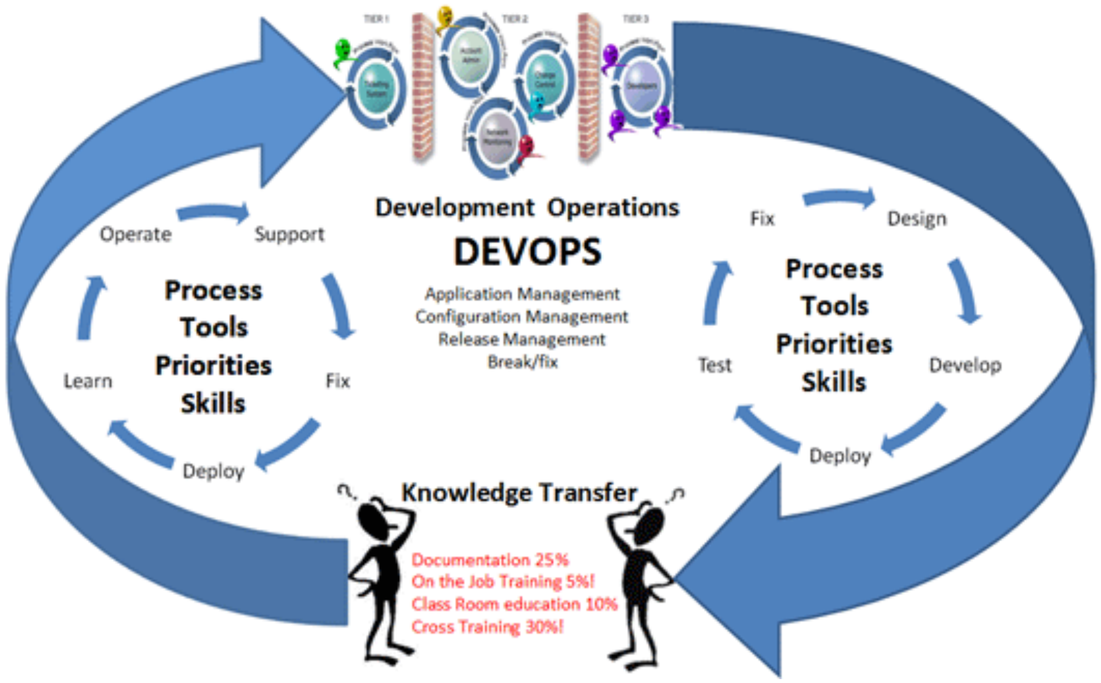
ERNEST MUELLER

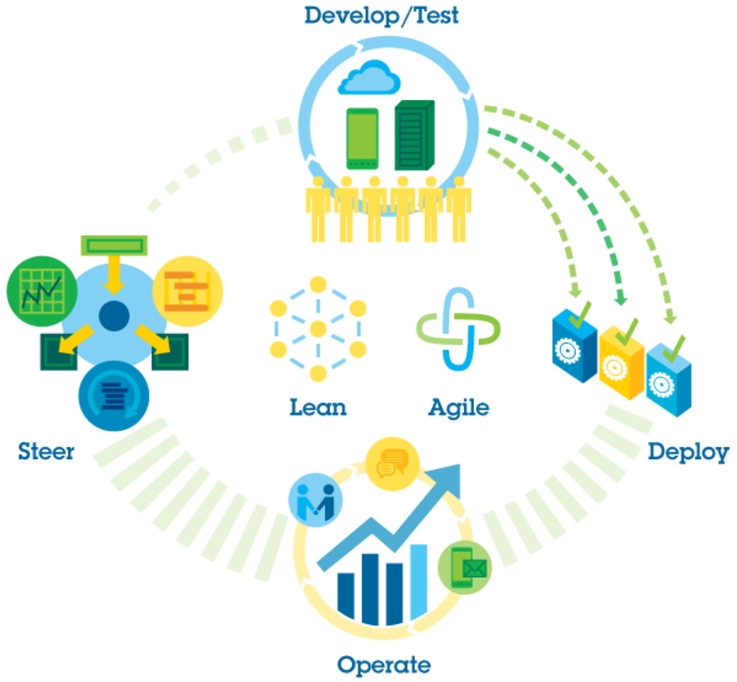

DevOps
AUSTIN
Days

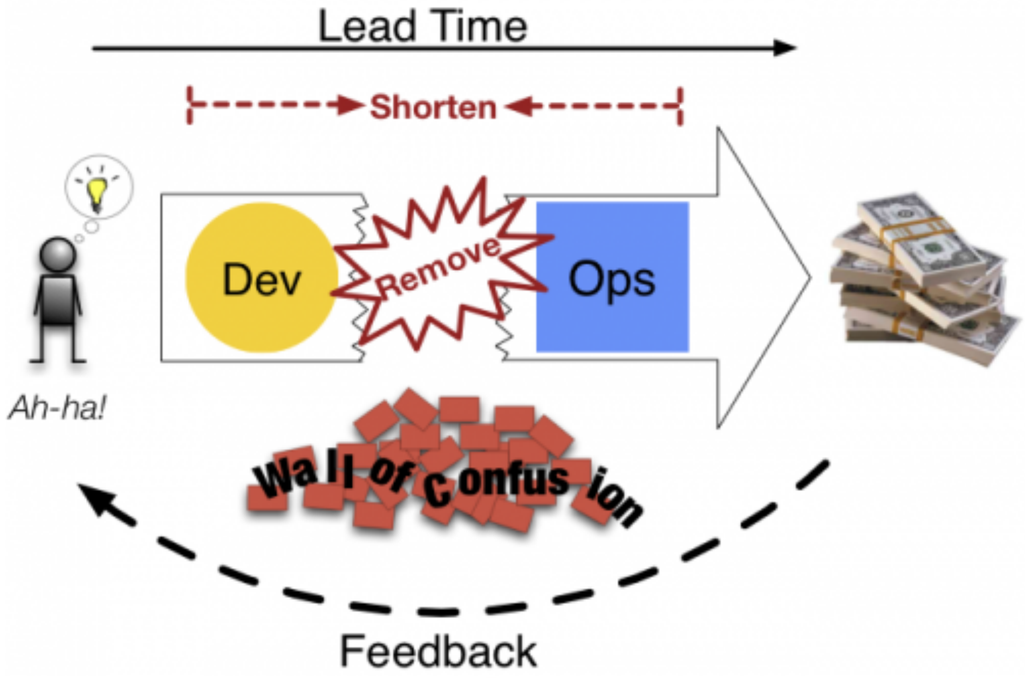
lynda.com

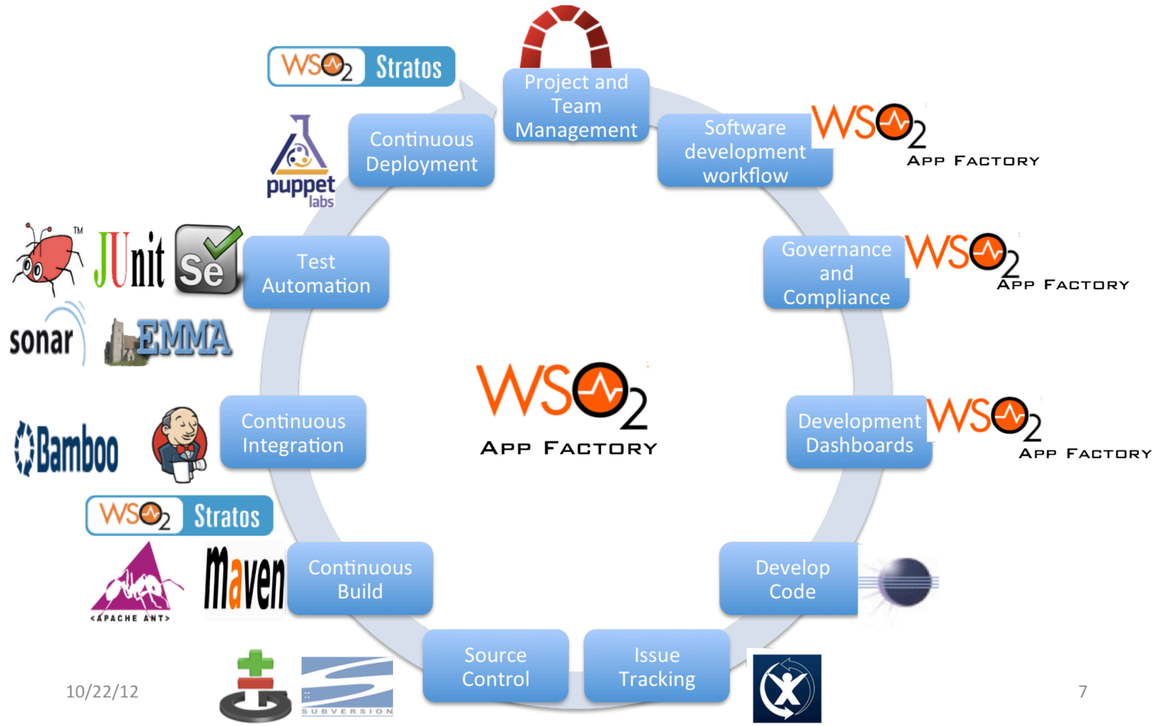
WHAT WE'LL COVER

- What is DevOps?
- What problems does DevOps address?
- How does DevOps relate to Agile?
- What can DevOps do for you?


Development Operations
DEVOPS
Application Management
Configuration Management
Release Management
Break/fix
Operate
Support
Process
Tools
Priorities
Skills
Learn
Fix
Deploy
Fix
Design
Process
Tools
Priorities
Skills
Test
Develop
Deploy
Knowledge Transfer
Documentation 25%
On the Job Training 5%!
Class Room education 10%
Cross Training 30%!


Develop/Test
Steer
Lean
Agile
Deploy
Operate


Lead Time
Shorten
Dev
Remove
Ops
Ah-ha!
Wall of Confusion
Feedback


WSO2 Stratos
Project and Team Management
Continuous Deployment
Software development workflow
WSO2 App Factory
puppet labs
JUnit
Test Automation
Governance and Compliance
WSO2 App Factory
sonar
EMMA
WSO2 App Factory
Bamboo
Continuous Integration
Development Dashboards
WSO2 App Factory
WSO2 Stratos
Maven
Continuous Build
Develop Code
Source Control
Issue Tracking

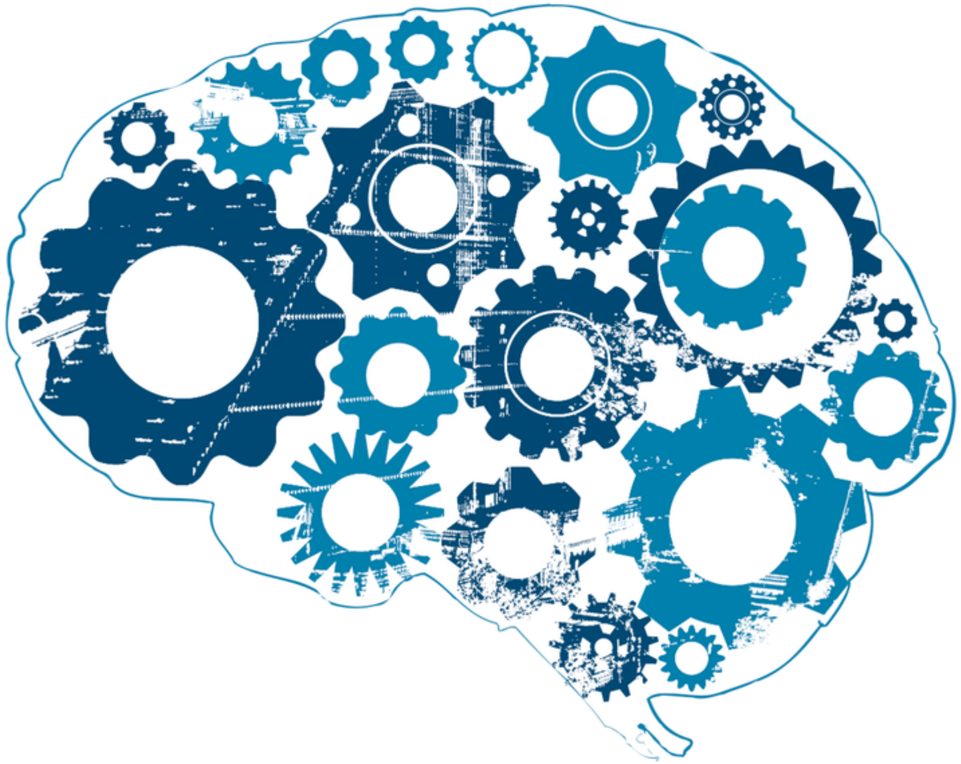
THE PROBLEM WE FACE

- Everything needs software nowadays.
- Software has to run on a server to become a service.
- Delivering a service from inception to its users is too slow and error-prone.
- There are internal friction points that make this the case.
- This loses you money. (delay = loss)
- IT is frequently the bottleneck in the transition of “concept to cash.”

SYMPTOMS

- Defects are released into production, causing outages
- Inability to diagnose production issues quickly
- Problems appear in some environments only
- Blame shifting/finger pointing
- Long delays while dev, QA, or another team waits on resource or response from other teams
- “Manual error” is a commonly cited root cause
- Releases slip/fail
- Quality of life issues in IT


WORKED FINE IN DEV...
...OPS PROBLEM NOW
memegenerator.net



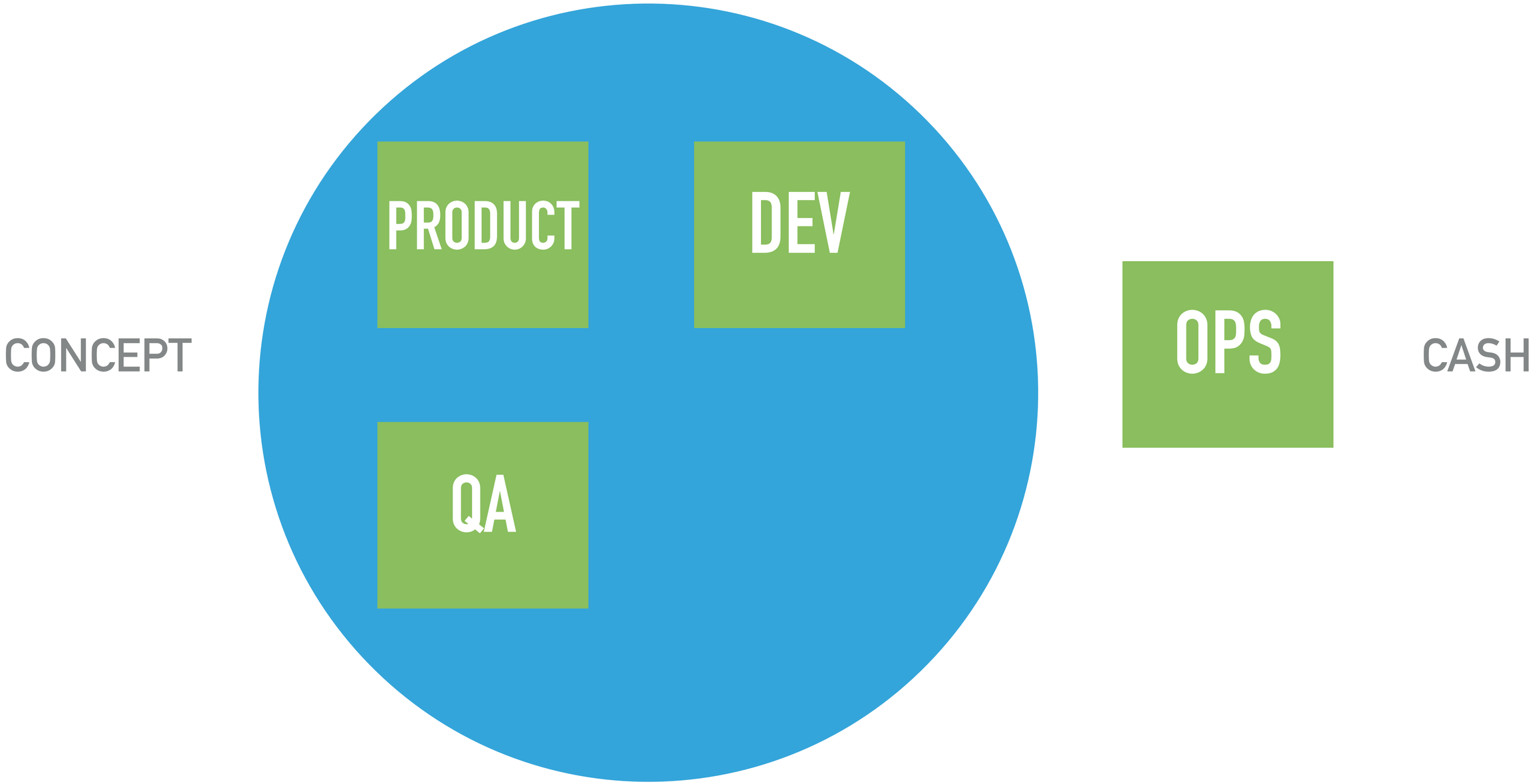
WHAT IS DEVOPS?

DEVOPS IS THE PRACTICE OF OPERATIONS AND DEVELOPMENT ENGINEERS PARTICIPATING TOGETHER IN THE ENTIRE SERVICE LIFECYCLE, FROM DESIGN THROUGH THE DEVELOPMENT PROCESS TO PRODUCTION SUPPORT.

DEVOPS IS ALSO CHARACTERIZED BY OPERATIONS STAFF MAKING USE OF MANY OF THE SAME TECHNIQUES AS DEVELOPERS FOR THEIR SYSTEMS WORK.

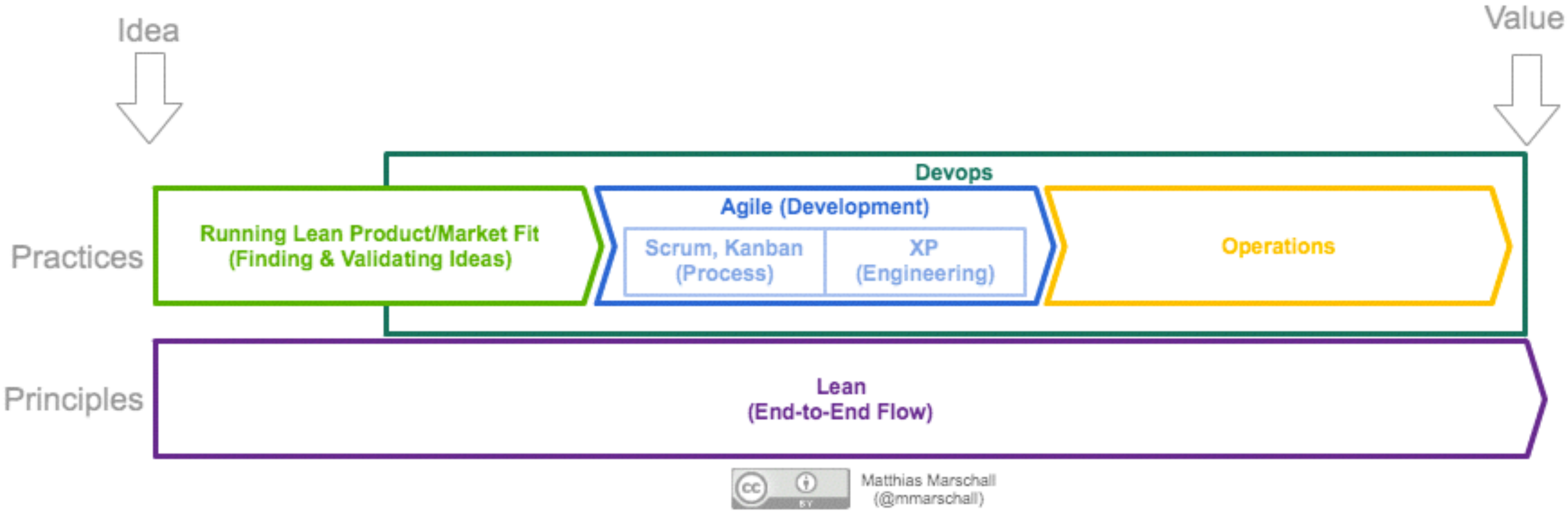
– theagileadmin.com

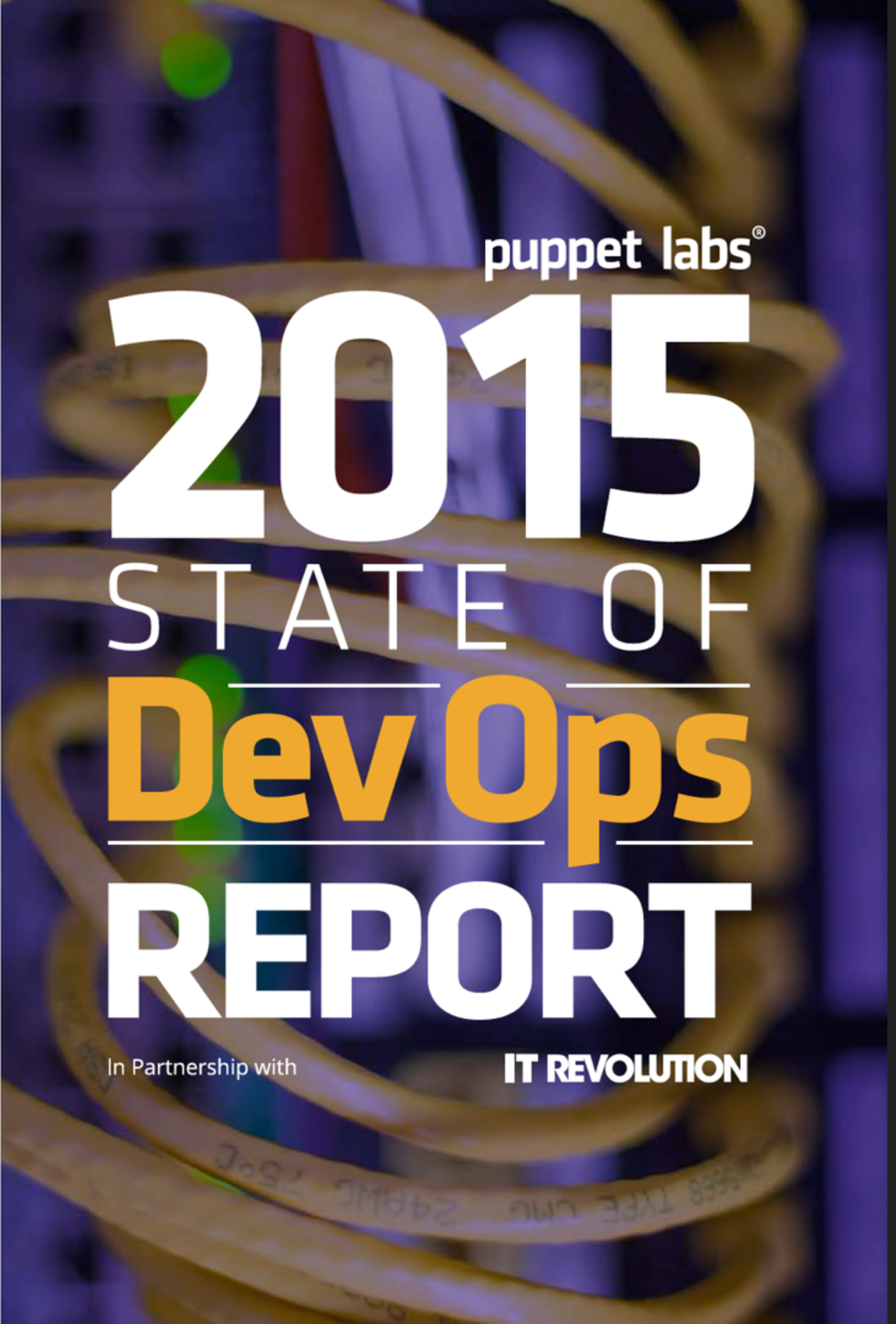
ADDING OPS TO THE AGILE TEAM



"DEVOPS IS THE APPLICATION OF AGILE METHODOLOGY TO SYSTEM ADMINISTRATION."

– Tom Limoncelli, The Practice of Cloud System Administration





- High-performing IT organizations deploy 30x more frequently with 200x shorter lead times; they have 60x fewer failures and recover from issues 168x faster.
- Lean management and continuous delivery practices create the conditions for delivering value faster, sustainably.
- High performance is achievable whether your apps are greenfield, brownfield or legacy.

DEVOPS IN DEPTH

CAMS

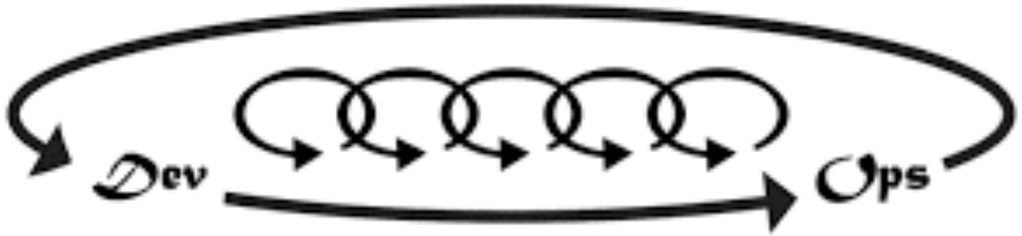
- Culture – People > Process > Tools
- Automation – Infrastructure as Code
- Measurement – Measure Everything
- Sharing – Collaboration/Feedback

THE THREE WAYS

The First Way:
Systems Thinking

The Second Way:
Amplify Feedback Loops

The Third Way:
Culture Of Continual Experimentation And Learning



THE THREE PILLARS OF DEVOPS

- Infrastructure Automation
- Continuous Delivery
- Reliability Engineering

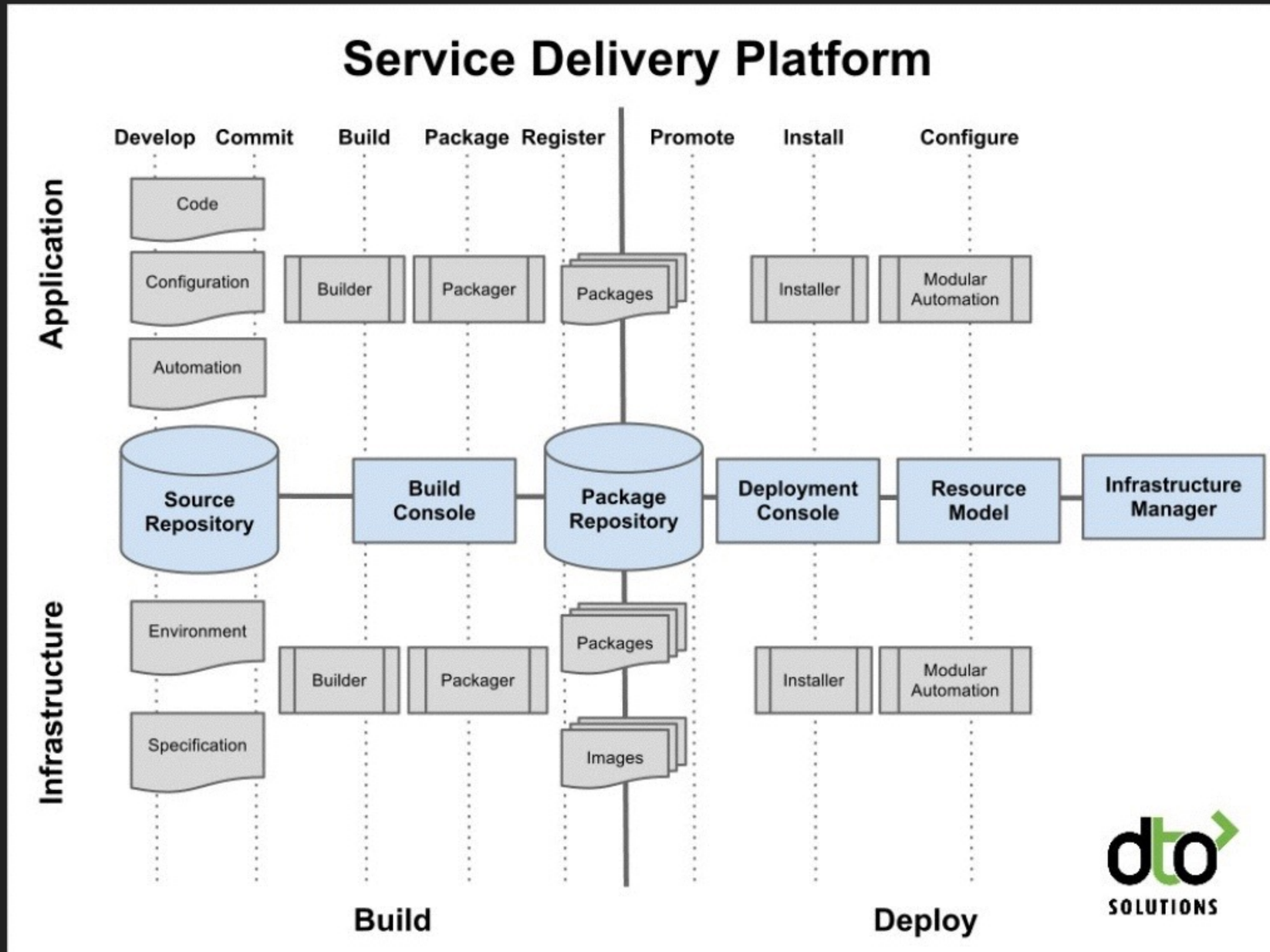
INFRASTRUCTURE AUTOMATION

INFRASTRUCTURE AS CODE

- Automate everything
  - Infrastructure provisioning
  - Application deployment
  - Runtime orchestration
- Model driven automation

INFRASTRUCTURE AS CODE

- Dev workflow
  - Write it in code
  - Validate the code
  - Unit test the code
  - Built it into an artifact
  - Deploy artifact to test
  - Integration test it
  - Deploy artifact to prod



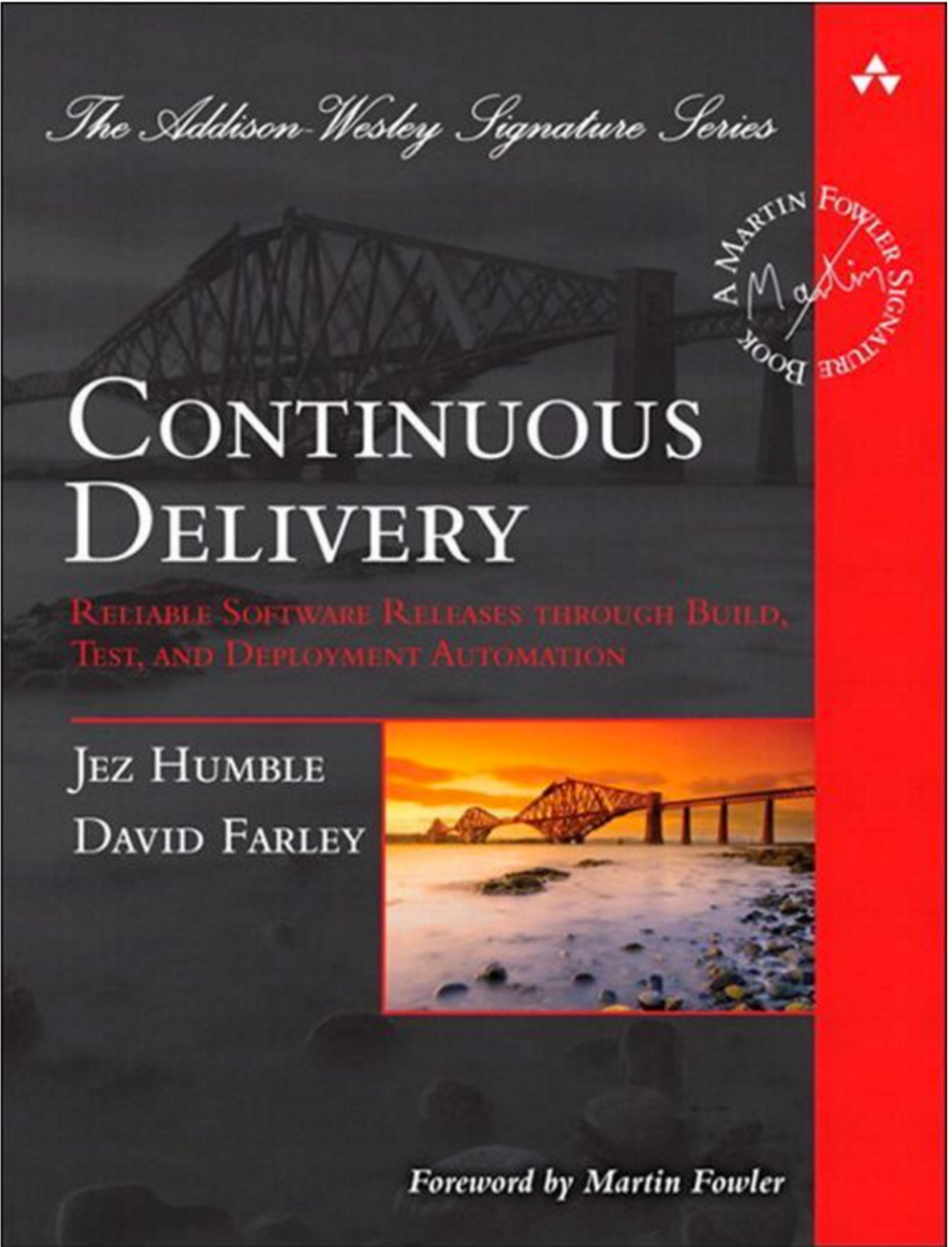
INFRASTRUCTURE AUTOMATION TOOLING

- Infrastructure Models - AWS Cloudformation, Terraform, Azure ARM Templates, Ubuntu Juju
- Hardware Provisioning - Packer, Foreman, MaaS, Cobbler, Crowbar, Digital Rebar
- Configuration Management - Puppet, Chef, Ansible, Salt, CFEngine
- Integration Testing - rspec, serverspec
- Orchestration - Rundeck, Ansible, Kubernetes (for docker)

SPECIAL TOPICS

- Immutable deployment with docker
- Serverless with AWS Lambda/Azure Functions/Google Cloud Functions
- Single model for infra and apps is best (Juju, Kubernetes)

CONTINUOUS DELIVERY



<- All You Need To Know

- Integration (CI): Build and test
- Deployment (CD): Deploy and integration test
- Delivery: All the way to production, baby

"CEASE DEPENDENCE ON INSPECTION TO ACHIEVE QUALITY. ELIMINATE THE NEED FOR INSPECTION ON A MASS BASIS BY BUILDING QUALITY INTO THE PRODUCT IN THE FIRST PLACE."

W. Edwards Deming

	High IT Performers	Medium IT Performers	Low IT Performers
Deployment frequency *For the primary application or service you work on, how often does your organization deploy code?*	On demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every 6 months
Lead time for changes *For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code commit to code successfully running in production)?*	Less than one hour	Between one week and one month	Between one month and 6 months
Mean time to recover (MTTR) *For the primary application or service you work on, how long does it generally take to restore service when a service incident occurs (e.g., unplanned outage, service impairment)?*	Less than one hour	Less than one day	Less than one day*
Change failure rate *For the primary application or service you work on, what percentage of the changes either result in degraded service or subsequently require remediation (e.g., lead to service impairment, service outage, require a hotfix, rollback, fix forward, patch)?*	0-15%	31-45%	16-30%

WORKING SOFTWARE, ALL THE TIME

- Builds should pass the coffee test (< 5 minutes)
- Commit really small bits
- Don’t leave the build broken
- Use a trunk/master based development flow (branch less than a day - use feature flags to branch by abstraction)
- Don't allow flaky tests, fix them!
- The build should return a status, a log, and an artifact.

THE CD PIPELINE

- Only build artifacts once
- Artifacts should be immutable
- Deployment should go to a copy of production before going into production
- Stop deploys if it a previous step fails
- Deployments should be idempotent

TEST FOR SUCCESS

- Automated testing is CI table stakes
- Unit tests
- Integration tests
- Crossbrowser, performance, security, etc.
- Test driven development, ideally (TDD/BDD/ATDD)
- Do it
- Stop being a crybaby
- Really, do it

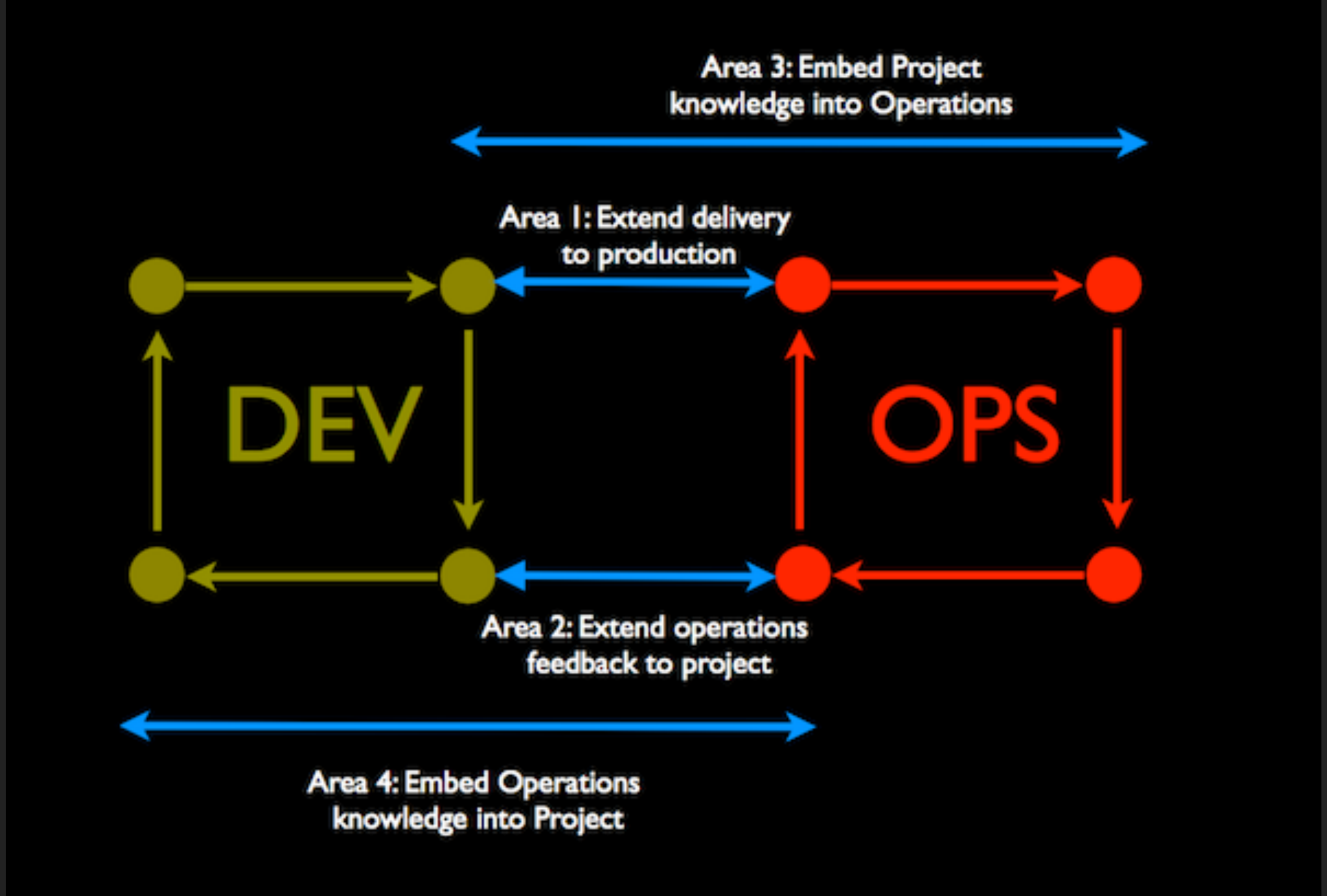
NO PULL REQUESTS

WITHOUT TESTS

TOOLING

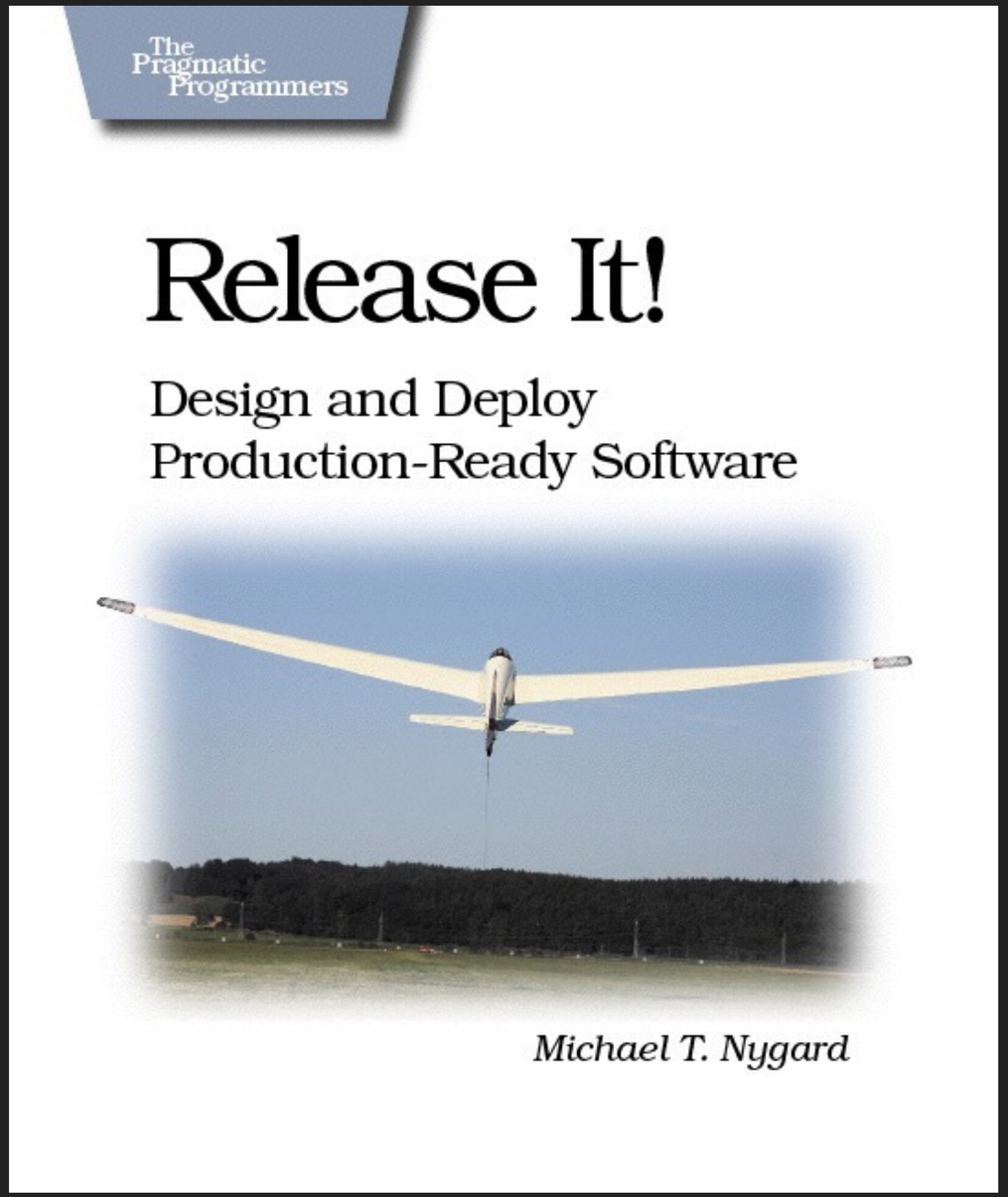
- Version Control (git, others, doesn’t matter as long as you use it)
- CI System (jenkins, bamboo, circleCI, travisCI)
- Build (whatever your platform needs - make/rake, maven, gulp, packer)
- Test (*unit, robot/protractor, cucumber)
- Artifact Repository (artifactory, nexus, dockerhub, S3)
- Deployment (rundeck, ansible, your CM system)

RELIABILITY ENGINEERING


Area 3: Embed Project
knowledge into Operations
Area 1: Extend delivery
to production
DEV
OPS
Area 2: Extend operations
feedback to project
Area 4: Embed Operations
knowledge into Project

DESIGN FOR OPERATION

- Design patterns exist for creating resilient systems.
- If your app sucks, there's only so much an ops team can do about it once it's deployed.
- Devs need to take responsibility for their app through deployment.

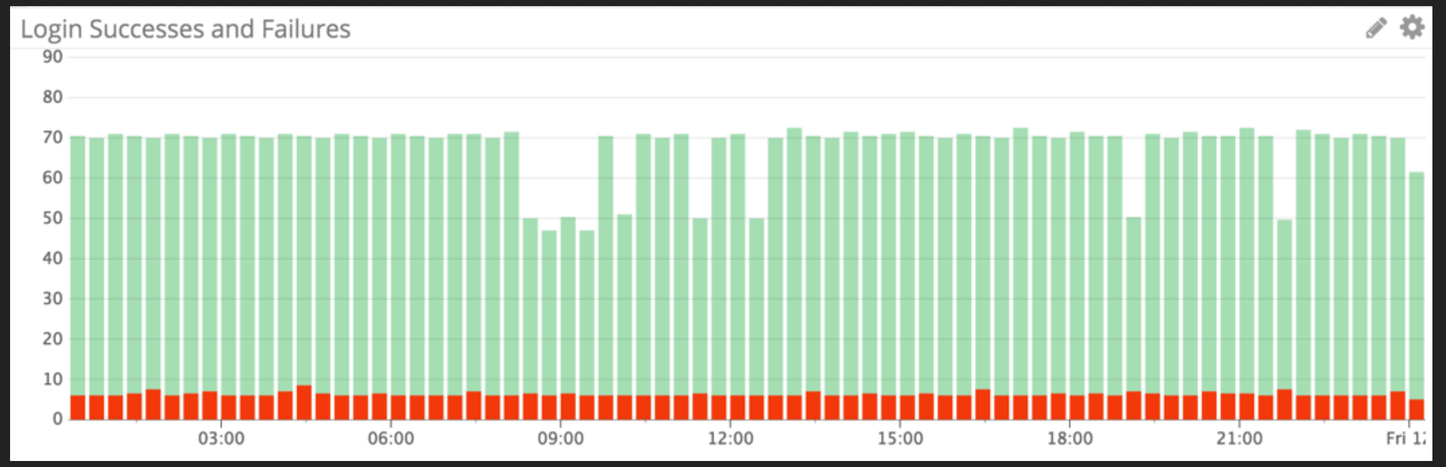


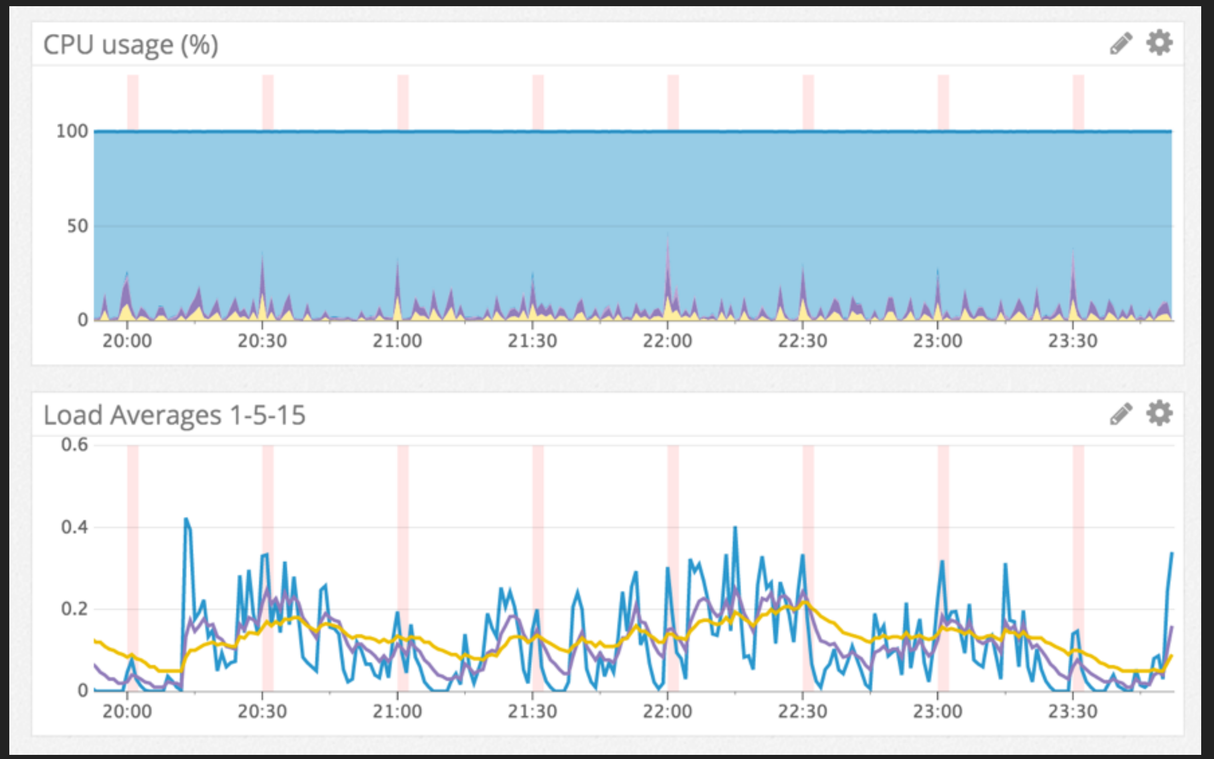
ADDING OPS INTO DEV

- Enhance Service Design With Operational Knowledge
  - Reliability
  - Performance
  - Security
  - Test These
  - Design Patterns (Release It!, 12-factor apps)
- Foster a Culture of Responsibility
  - Whether your code passes test, gets deployed, and stays up for users is your responsibility – not someone else's
- Make Development Better With Ops
  - Productionlike environments
  - Power tooling - vagrant, etc.

OPERATE FOR DESIGN

- Build
- Measure
- Learn
- Repeat





ADDING DEV INTO OPS

- Don’t do tasks for people. Build tools so they can do their own work. Ops is now a dev team and the platform is their product.
- Automate. This means code, and using coder practices (from source control on)
- Build Feedback Paths Back To Dev From Production
  - Monitoring and metrics - Ensure that monitoring/logging/metrics feed back into dev (and the business)
  - Blameless Incident Postmortems
- Developers Do On Call Production Support
- SRE model - devs support till it gets big and ops accepts support

TO LEARN MORE

- Gene Kim, John Willis, Jez Humble, and Damon Edwards' The DevOps Handbook
- Gene Kim's The Phoenix Project
- Jez Humble's Continuous Delivery
- Michael Nygard's Release It!
- Tom Limoncelli's The Practice Of Cloud System Administration
- Mary and Tom Poppendieck's Lean Software Development
- Velocity Conference (velocityconf.com)
- DevOpsDays Conferences (devopsdays.org)
- DevOps Weekly newsletter (devopsweekly.com)
- DevOps Café Podcast (devopscafe.com)
- The Twelve Factor App (12factor.net)
- The Agile Admin (theagileadmin.com)

GETTING STARTED

- Automated infrastructure and automated testing give you safety
- Making it all continuous gives you speed and safety
- Engineering for reliability gives you more safety
- Accelerating flow makes you money

BONUS PRO TIP:
BIMODAL IT

BIMODAL HAS MOVED FROM 38% OF COMPANIES ADOPTING THIS STRATEGY IN 2016 TO 43% IN 2017.

Gartner

AM I THE ONLY ONE WHO READS THIS AS “43% OF COMPANIES STILL CAN’T FIGURE OUT WHY THEIR IT ORGANIZATIONS ARE DYSFUNCTIONAL”?

Damon Edwards

THANKS!

THEAGILEADMIN.COM