DEVOPS: WHAT IS THAT, ANYWAY?

BY: LAURA STONE

WHO AM I?

- "Cloud DevOps Engineer"
- Background in QA
- Background in Operations

Twitter: @tyrostone

GitHub: laura-stone

WHAT'S THE AGENDA?

1. DevOps Confusion
2. History of DevOps
3. Definitions
4. How to DevOps
5. Apply concepts to Python environment
6. Further resources
7. Questions

WHAT IS DEVOPS?

WHAT IS DEVOPS?

Ops all the things?!

Continuous integration

Agile

Operations

Scale

DevOps teams

Microservices

Development

Business growth

Reliability

Automation

Lean

DevOps tools

Synergy

Moving faster

Silos

Technology Organizations Create Impenetrable Silos

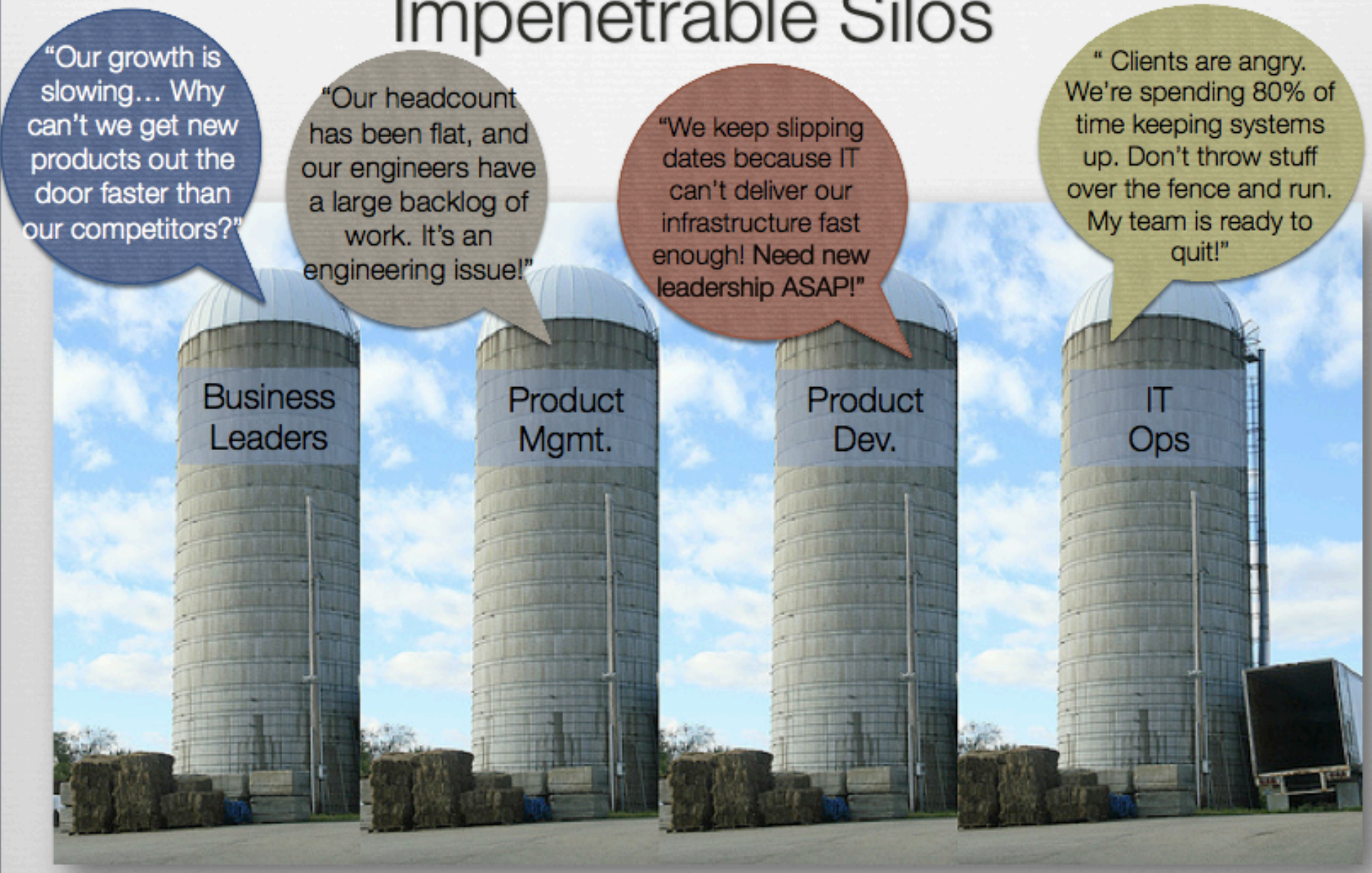
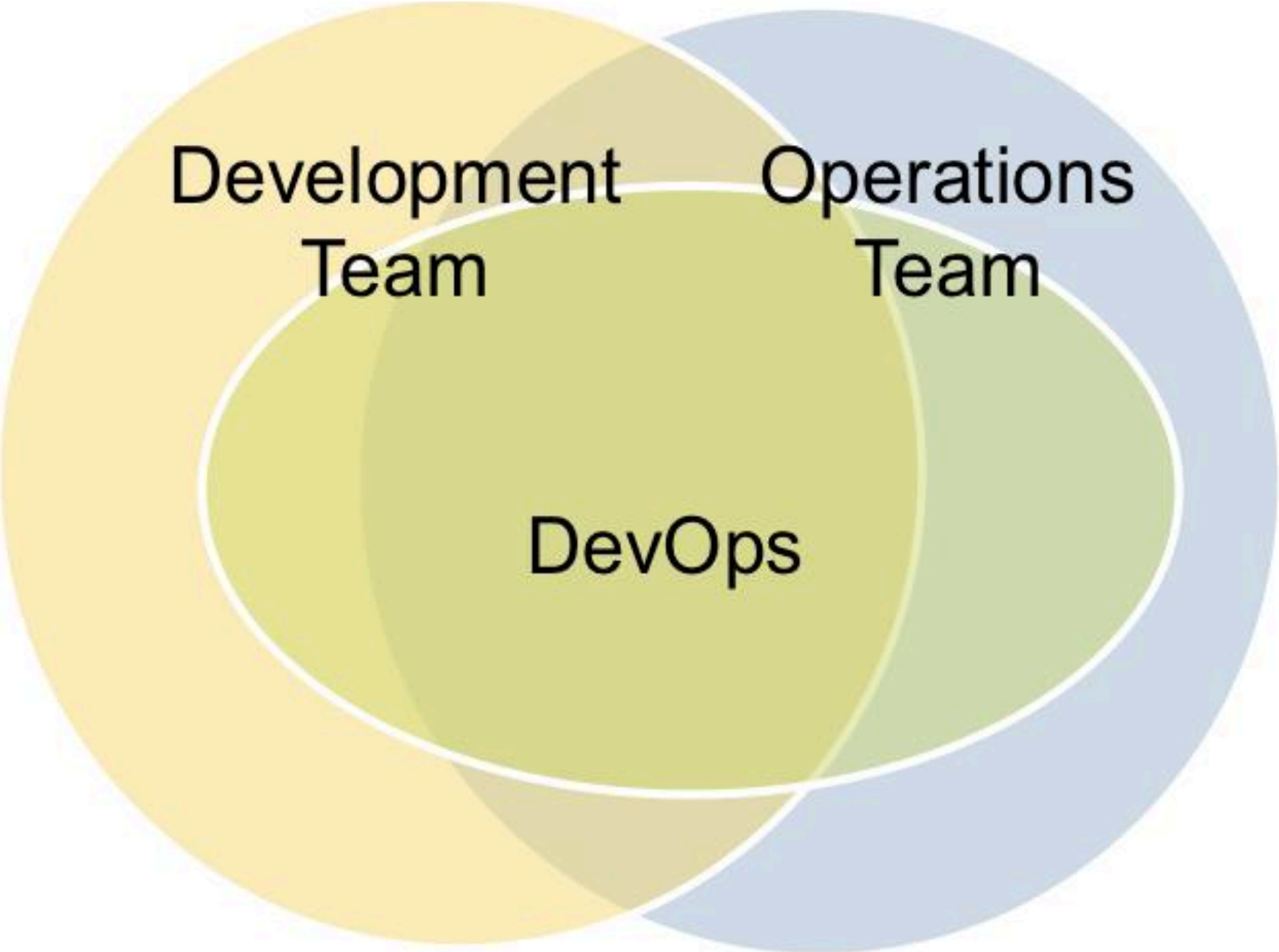


Image Source: http://www.wired.com/insights/wp-content/uploads/2012/05/silos.jpg


Development Team
Operations Team
DevOps


STOP

WHAT IS DEVOPS?

DEVOPS IS…

#DEVOPS(*DAYS*)

DEVOPS: THE DEFINITION*

"a cross-disciplinary movement dedicated to the praxis of building and delivering quality systems in order to solve business problems and provide value to customers"

*sort of

BUT…
WHAT DOES THAT MEAN?

CROSS-DISCIPLINARY

MOVEMENT

PRAXIS

SOLVE BUSINESS PROBLEMS

DELIVER VALUE TO CUSTOMERS

HOW TO DEVOPS

HOW TO DEVOPS

- CAMS
  - Culture
  - Automation
  - Measurement
  - Sharing
- Etc.

*sort of

CULTURE

Learning

Collaboration

Empathy

Breaking down silos

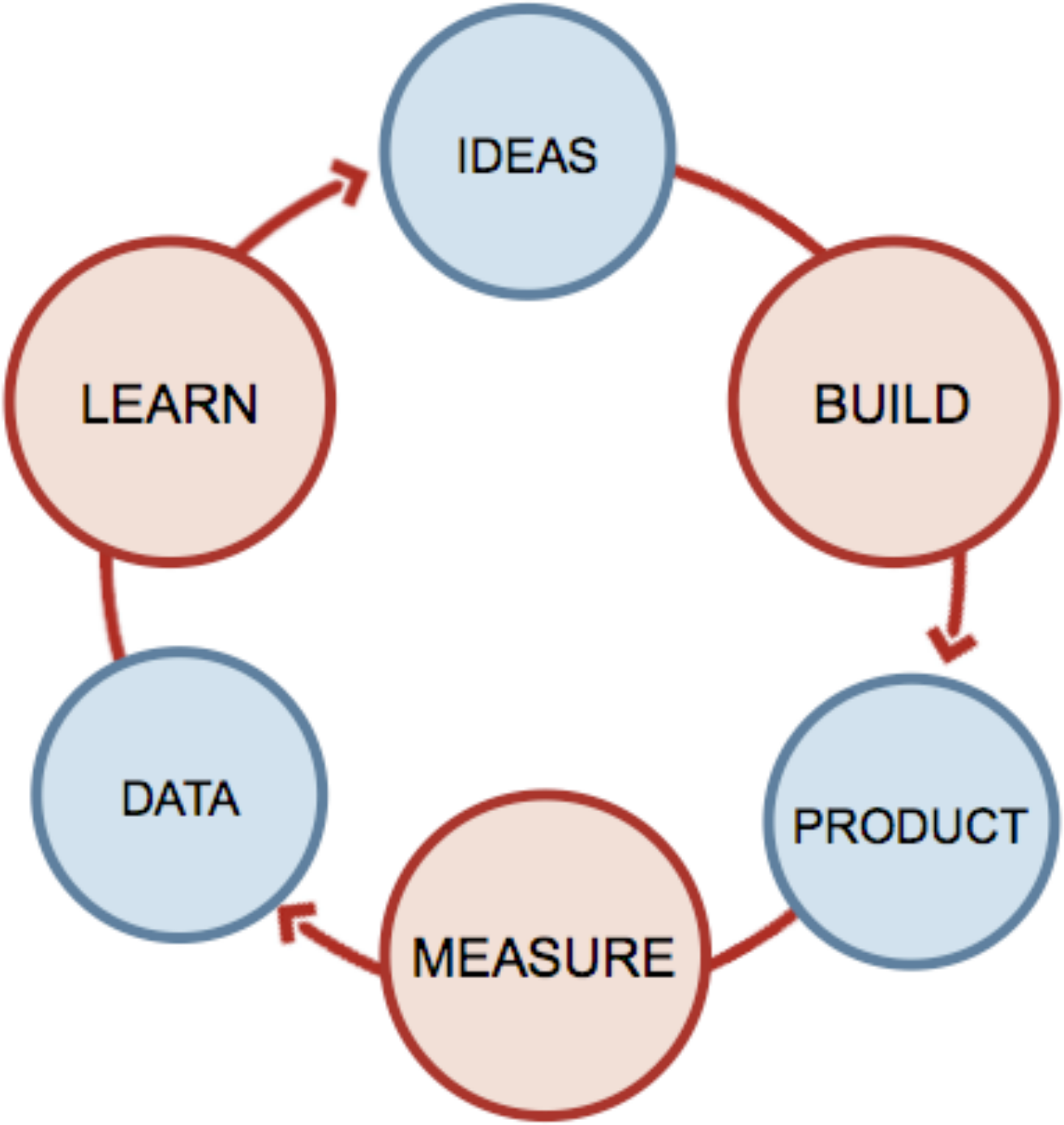
Diversity

Teamwork

Experimentation

Communication

LET'S COME BACK TO THIS…


IDEAS
BUILD
PRODUCT
MEASURE
DATA
LEARN

PEOPLE

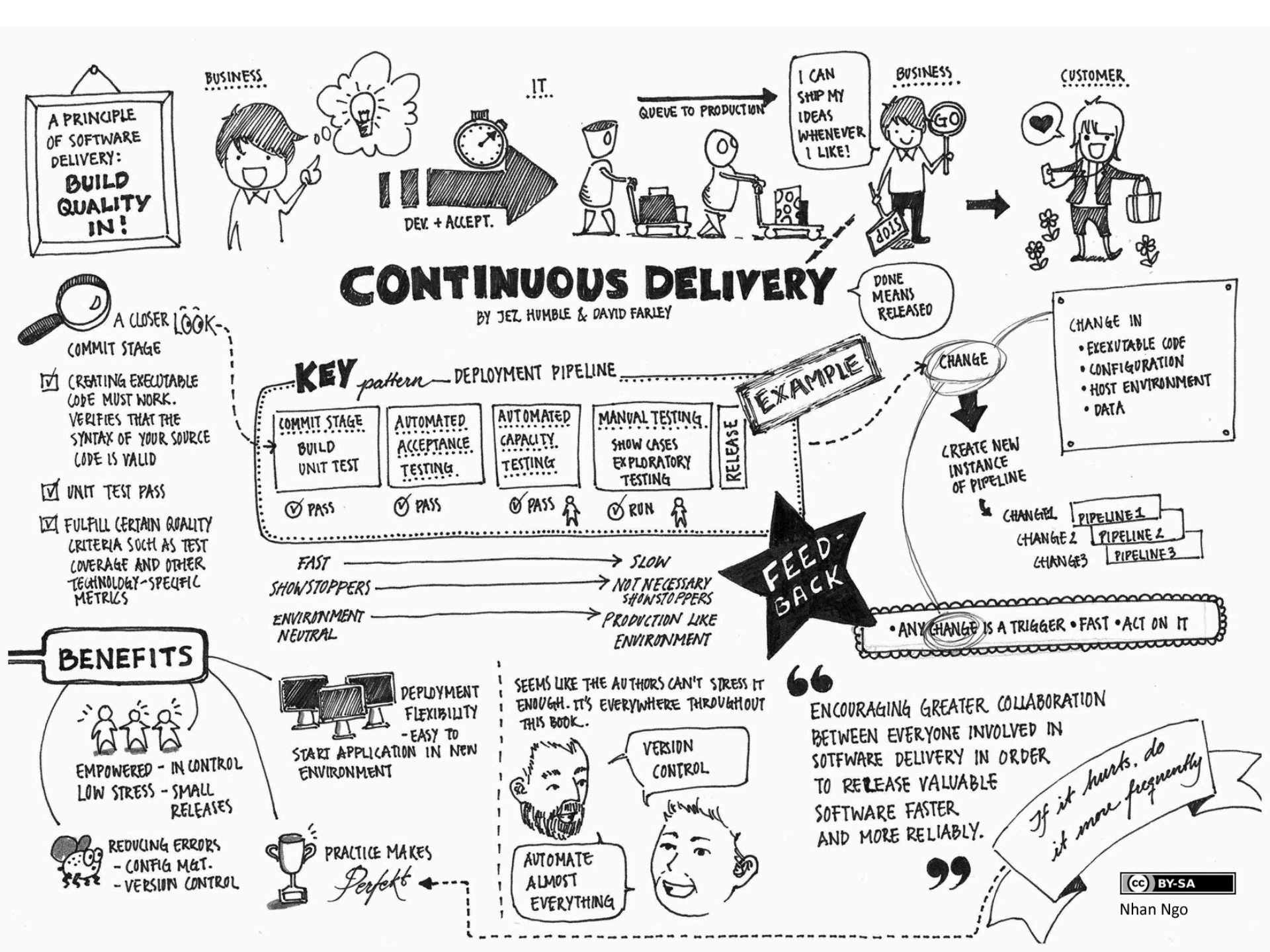
PROCESSES

TOOLS

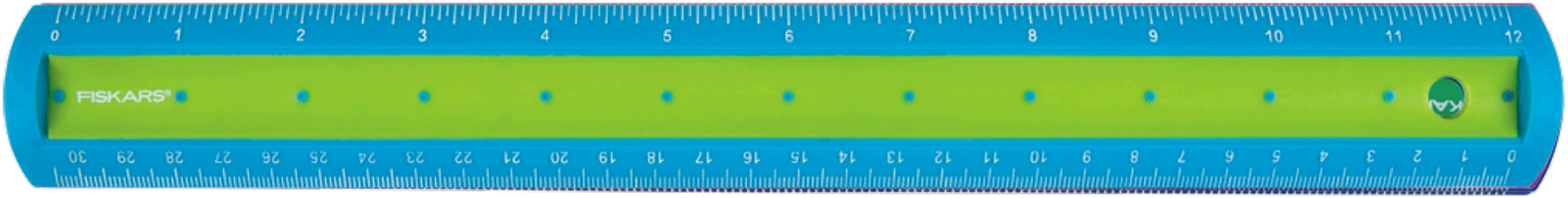
AUTOMATION

- Continuous Delivery
- Infrastructure as code
- Version control
- Value stream mapping

*sort of


A PRINCIPLE OF SOFTWARE DELIVERY: BUILD QUALITY IN!
BUSINESS
IT
DEV. + ACCEPT.
QUEUE TO PRODUCTION
I CAN SHIP MY IDEAS WHENEVER I LIKE!
BUSINESS
GO
STOP
CUSTOMER
CONTINUOUS DELIVERY
BY JEZ HUMBLE & DAVID FARLEY
DONE MEANS RELEASED
A CLOSER LOOK
COMMIT STAGE
CREATING EXECUTABLE CODE MUST WORK. VERIFIES THAT THE SYNTAX OF YOUR SOURCE CODE IS VALID
UNIT TEST PASS
FULFILL CERTAIN QUALITY CRITERIA SUCH AS TEST COVERAGE AND OTHER TECHNOLOGY-SPECIFIC METRICS
KEY pattern — DEPLOYMENT PIPELINE
EXAMPLE
COMMIT STAGE
BUILD
UNIT TEST
AUTOMATED ACCEPTANCE TESTING
AUTOMATED CAPACITY TESTING
MANUAL TESTING
SHOW CASES
EXPLORATORY TESTING
RELEASE
PASS
PASS
PASS
RUN
FAST → SLOW
SHOWSTOPPERS → NOT NECESSARY SHOWSTOPPERS
ENVIRONMENT NEUTRAL → PRODUCTION LIKE ENVIRONMENT
FEED-BACK
CHANGE
CHANGE IN
• EXEXUTABLE CODE
• CONFIGURATION
• HOST ENVIRONMENT
• DATA
CREATE NEW INSTANCE OF PIPELINE
CHANGE1 PIPELINE 1
CHANGE 2 PIPELINE 2
CHANGE3 PIPELINE 3
• ANY CHANGE IS A TRIGGER • FAST • ACT ON IT
BENEFITS
EMPOWERED - IN CONTROL
LOW STRESS - SMALL RELEASES
DEPLOYMENT FLEXIBILITY - EASY TO START APPLICATION IN NEW ENVIRONMENT
REDUCING ERRORS
- CONFIG MGT.
- VERSION CONTROL
PRACTICE MAKES Perfect
SEEMS LIKE THE AUTHORS CAN'T STRESS IT ENOUGH. IT'S EVERYWHERE THROUGHOUT THIS BOOK.
VERSION CONTROL
AUTOMATE ALMOST EVERYTHING
ENCOURAGING GREATER COLLABORATION BETWEEN EVERYONE INVOLVED IN SOFTWARE DELIVERY IN ORDER TO RELEASE VALUABLE SOFTWARE FASTER AND MORE RELIABLY.
If it hurts, do it more frequently
CC BY-SA
Nhan Ngo

MEASUREMENT



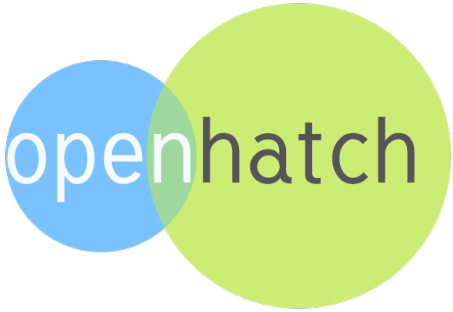
Why?

- Expect failure!
- Failure as learning opportunity!
- Continuous improvement!

How?

- Logging
- Monitoring
- Alerting
- Anonymous Surveys
- Retrospectives

SHARING



code is fun

- Learning Lunches
- Hackathons
- Open Source
- Meetups
  - Boston Devops
  - Boston Amazon Web Services
  - Boston Infrastructure Coders
  - Ansible Boston
- Conferences
  - DevOpsDays Boston

*sort of

HOW TO DEVOPS (PYTHON STYLE)

IT DEPENDS

Maturity Model Table

	Base	Beginner	Intermediate	Advanced	Expert
Source control	• No source control	• Code is under source control • One project/repository • Uncontrolled branching process	• New components go into separate projects/repositories • Well-defined branching process, including code reviews	• Trunk/master is always kept up to date with latest code from developers – continuous integration • Branching process designed with CI in mind	• One artifact per repository • Release off of master • Automated tagging
Environments	• Inconsistent environments between dev and production	• Environment setup for development and testing is well understood and documented	• Primary testing environment (qa) is similar to production • Penultimate promotion environment (staging) is a production replica	• Environment setup is automated for various promotion environments • Environment provisioning code is treated like any other code	• Environments are consistent across all promotion levels
Build & Test	• Inconsistent build processes • Testing is done entirely by QA • Developers are not developing with testing in mind	• Builds are automated on a non-development system • Build process is documented for new users • Some unit tests are being written by developers	• Build once, deploy everywhere • Increasing unit test code coverage is an established goal and priority • Unit test suite is run consistently by developers before changes are committed • "You break it, you fix it" – hard promotion gates	• Automated deployment to promotion environments – continuous delivery • Automated testing provides accurate data to determine code quality • Manual testing fills any remaining gaps in release candidates • Coding standards tools (checkstyle, pmd, etc) are raised to the same level as functional tests	• Manual testing is needed only in (rare) edge cases
Deployment	• Code is deployed directly from source control by a manual process	• Code is deployed in packages	• Configuration is managed separately from application installation • Deployment can be done in a way that is invisible to users	• Push-button deployments • Deployment code is treated like any other code	• Automated deploys to production – continuous deployment
Reporting	• Any reports that exist are put together manually	• Reports on code health (test results, coverage, etc.) are being generated when tests are being run	• Code health metrics can be graphically represented • Trending metrics are kept across consecutive iterations of the same build	• Automatic integration between various information sources (eg. git, jira, Jenkins)	• "Single point of truth" about the quality of code and where it is installed

DEMO

PYTHON FOR GREAT DEVOPS GOOD

Deployment

- Fabric

Provisioning

- Boto (AWS)
- Troposphere

Configuration Management

- Ansible
- Salt(Stack)

Continuous Integration

- Buildbot

Monitoring

- AmonOne
- ServerDensity
- Glances

Etc.

https://wiki.python.org/moin/ConfigurationAndBuildTools

RESOURCES

- Books
  - The Lean Startup – Eric Ries
  - The Phoenix Project - Gene Kim, Kevin Behr, George Spafford
  - The Goal - Eliyahu M. Goldratt, Jeff Cox
  - Continuous Delivery – Jez Humble
  - Web Operations – John Allspaw

RESOURCES

- Videos
  - Chef Style DevOps Kung Fu
  - 10+ Deploys Per Day: Dev and Ops Cooperation at Flickr
  - The (Short) History of DevOps
  - Continuously Deploying Culture

RESOURCES

- Tutorials/Online Resources
  - Full Stack Python
  - How to Containerize Python Applications
  - Infrastructure with Python
  - Udacity - Intro to DevOps
  - DevOps: A Crash Course

“CULTURE”

QUESTIONS?

Twitter: @tyrostone
GitHub: laura-stone