MICROSERVICES: THE ORGANISATIONAL AND PEOPLE IMPACT

DANIEL BRYANT

@DANIELBRYANTUK

@SPECTOLABS

SpectoLabs

TL;DR

MOST OF THE CURRENT PROBLEMS WE SEE WITH IMPLEMENTING MICROSERVICES ARE CONNECTED WITH PEOPLE AND ORGANISATIONAL SYSTEMS

THIS IS GOOD NEWS, AS WE ALREADY HAVE EXISTING SOLUTIONS!

@DANIELBRYANTUK

- TECHNICAL CONSULTANT, CTO AT SPECTOLABS
  - ARCHITECTURE, CI/CD, PROGRAMMABLE INFRASTRUCTURE
  - JAVA, GO, JS, MICROSERVICES, CLOUD, CONTAINERS
  - CONTINUOUS DELIVERY OF VALUE THROUGH EFFECTIVE TECHNOLOGY AND TEAMS

O'REILLY

Containerizing Continuous Delivery in Java

Docker Integration for Build Pipelines and Application Architecture

Daniel Bryant

O'REILLY

Continuous Delivery in Java

ESSENTIAL TOOLS AND BEST PRACTICES FOR DEPLOYING CODE TO PRODUCTION

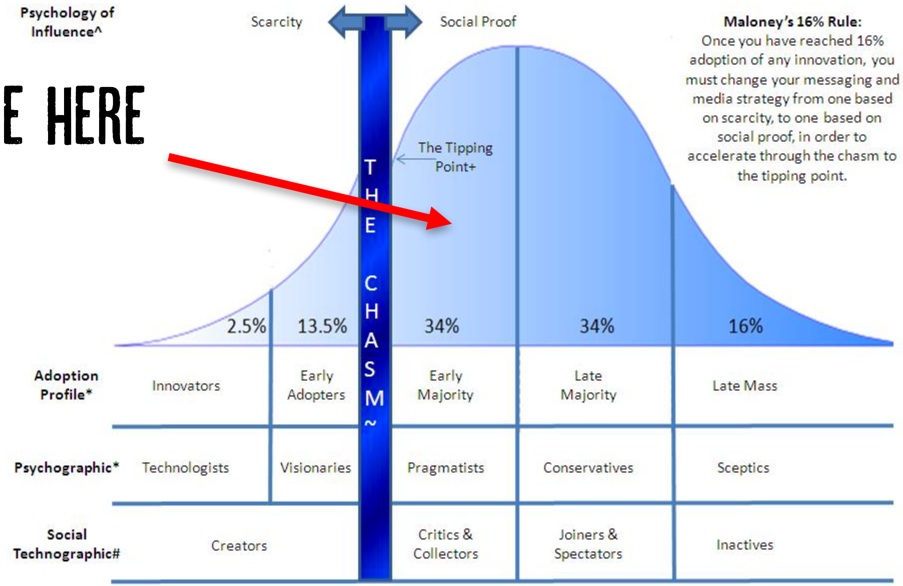
Daniel Bryant

bit.ly/2jWDSF7

PREAMBLE: MICROSERVICES ADOPTION

MICROSERVICES ARE HERE

Accelerating Diffusion of Innovation: Maloney's 16% Rule©



Maloney's 16% Rule: Once you have reached 16% adoption of any innovation, you must change your messaging and media strategy from one based on scarcity, to one based on social proof, in order to accelerate through the chasm to the tipping point.

Adoption Profile*	Innovators	Early Adopters	Early Majority	Late Majority	Late Mass
Psychographic*	Technologists	Visionaries	Pragmatists	Conservatives	Sceptics
Social Technographic#	Creators		Critics & Collectors	Joiners & Spectators	Inactives

^ Robert Cialdini *Everett Rogers #Forresters ~Geoffrey Moore + Malcolm Gladwell

innovateordie.com.au/2010/05/10/the-secret-to-accelerating-diffusion-of-innovation-the-16-rule-explained/

TODAY…

CREATE AND SHARE STRATEGY/TACTICS

OPTIMISE FOR FEEDBACK

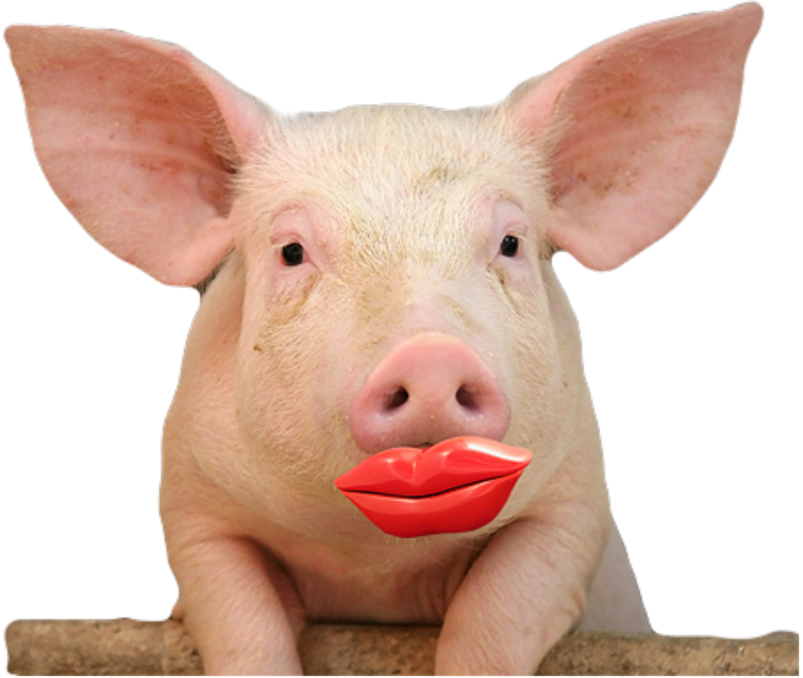
DEFINE RESPONSIBILITIES

TOM FISH BURNE
OUR MISSION
PRETTY MUCH WHATEVER OUR COMPETITION DOES, BUT SIX MONTHS LATER
© marketoonist.com

1. STRATEGY - SITUATIONAL AWARENESS, VISION & EVALUATION

STRATEGY: ARE MICROSERVICES A GOOD FIT?

- “OUR ’MODE TWO’ APPS ARE MICROSERVICES”
  - MIDDLE-MANAGEMENT LATCH ON TO BUZZWORD
  - LIPSTICK ON THE PIG
- NOT UNDERSTANDING ARCHITECTURE PRINCIPLES
  - NOT BUILDING AROUND BUSINESS FUNCTIONALITY
  - CREATING MINI-MONOLITHS (NO TWELVE FACTORS)
- NO WELL-DEFINED DEVOPS / SRE / OPS
  - DEPLOYMENT/OPERATIONAL FREE-FOR-ALL



ANTIPATTERN

ANTIPATTERN: (MICRO) SILVER BULLETS

- NO WELL-DEFINED GOALS/STRATEGY

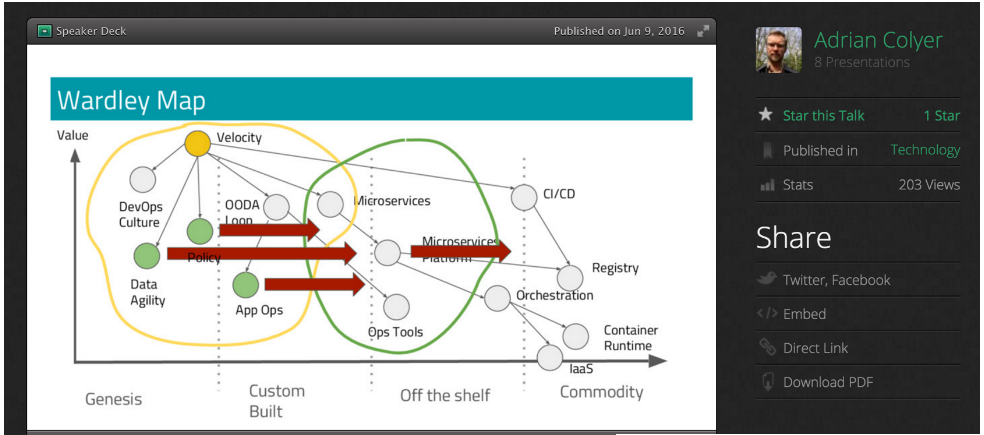
- BUT…
  - MICROSERVICES IS THE SOLUTION?

- DETERMINE BUSINESS GOALS, CREATE HYPOTHESIS, (CHOOSE TECH?), AND VALIDATE

WHAT ARE OUR GOALS?

- DELIVERY OF VALUE TO END USERS (CUSTOMERS)
- BUSINESS AGILITY
- SAFER, MORE RAPID CHANGES TO SOFTWARE SYSTEMS
  - BUT CONSIDER CI/CD, DEVOPS, AND VALUE STREAM BEFORE MICROSERVICES

SITUATIONAL AWARENESS


Speaker Deck

Published on Jun 9, 2016

Adrian Colyer
8 Presentations

★ Star this Talk 1 Star

Published in Technology

Stats 203 Views

Share

Twitter, Facebook

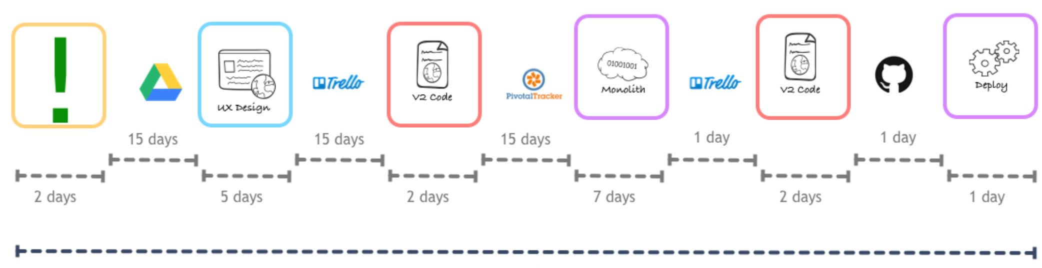
Embed

Direct Link

Download PDF

Making Sense of it All by Adrian Colyer
Published June 9, 2016 in Technology

speakerdeck.com/acolyer/making-sense-of-it-all



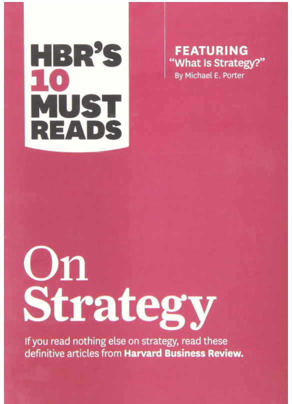
philcalcado.com/2015/09/08/how_we_ended_up_with_microservices.html

02/10/2017 @danielbryantuk

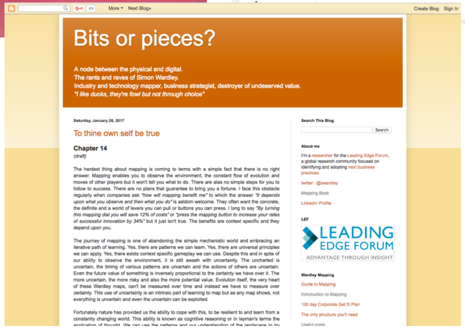
SpectoLabs

DEFINE GOALS AND STRATEGY

- CREATE S.M.A.R.T. GOALS
  - WITH ALL STAKEHOLDERS
- PICK YOUR (STRATEGIC) POISON
- SHARE THE OUTCOMES…


HBR'S 10 MUST READS
FEATURING "What Is Strategy?" By Michael E. Porter
On Strategy
If you read nothing else on strategy, read these definitive articles from Harvard Business Review.

GOTO 2016 • Strategy on Big Programmes • Jamie Dobson



Bits or pieces?

A node between the physical and digital.
The rants and raves of Simon Wardley.
Industry and technology mapper, business strategist, destroyer of undeserved value.
"I like ducks, they're fowl but not through choice"

Saturday, January 28, 2017

To thine own self be true

Chapter 14
(draft)

The hardest thing about mapping is coming to terms with a simple fact that there is no right answer. Mapping enables you to observe the environment, the constant flow of evolution and moves of other players but it won't tell you what to do. There are alas no simple steps for you to follow to success. There are no plans that guarantee to bring you a fortune. I face this obstacle regularly when companies ask *"how will mapping benefit me"* to which the answer *"it depends upon what you observe and then what you do"* is seldom welcome. They often want the concrete, the definite and a world of levers you can pull or buttons you can press. I long to say *"By turning this mapping dial you will save 12% of costs"* or *"press the mapping button to increase your rates of successful innovation by 34%"* but it just isn't true. The benefits are context specific and they depend upon you.

The journey of mapping is one of abandoning the simple mechanistic world and embracing an iterative path of learning. Yes, there are patterns we can learn. Yes, there are universal principles we can apply. Yes, there exists context specific gameplay we can use. Despite this and in spite of our ability to observe the environment, it is still awash with uncertainty. The uncharted is uncertain, the timing of various patterns are uncertain and the actions of others are uncertain. Even the future value of something is inversely proportional to the certainty we have over it. The more uncertain, the more risky and also the more potential value. Evolution itself, the very heart of these Wardley maps, can't be measured over time and instead we have to measure over certainty. This use of uncertainty is an intrinsic part of learning to map but as any map shows, not everything is uncertain and even the uncertain can be exploited.

Fortunately nature has provided us the ability to cope with this, to be resilient to and learn from a constantly changing world. This ability is known as cognitive reasoning or in layman's terms the application of thought. We can use the patterns and our understanding of the landscape to try

Search This Blog

About me

I'm a researcher for the Leading Edge Forum, a global research community focused on identifying and adopting next business practices.

twitter : @swardley

Mapping Book

LinkedIn Profile

LEF

LEADING EDGE FORUM
ADVANTAGE THROUGH INSIGHT

Wardley Mapping

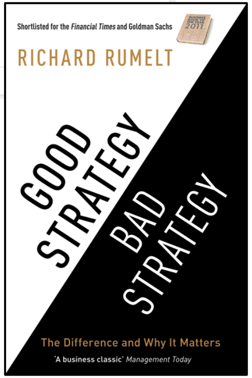
Guide to Mapping

Introduction to Mapping

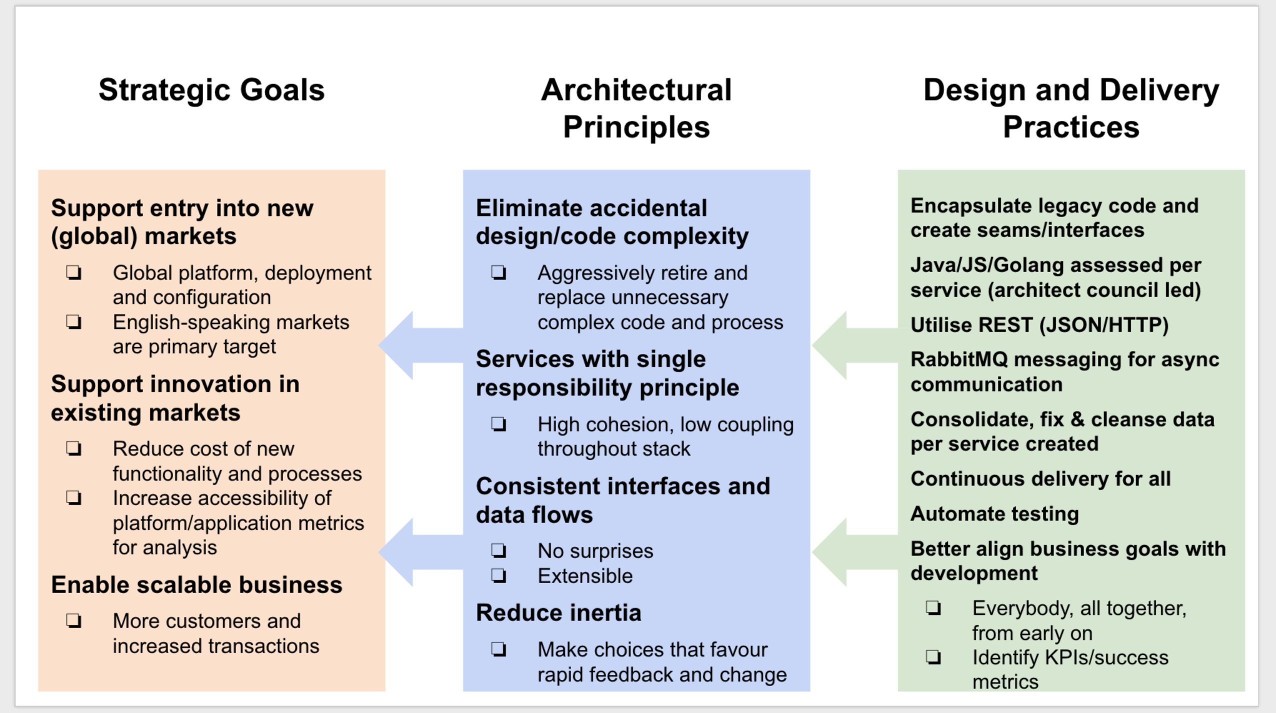
100 day Corporate Get fit Plan

The only structure you'll need

Useful posts


Shortlisted for the Financial Times and Goldman Sachs
RICHARD RUMELT
GOOD STRATEGY
BAD STRATEGY
The Difference and Why It Matters
'A business classic' Management Today

COMMUNICATE THE VISION



Strategic Goals

Support entry into new (global) markets

❏ Global platform, deployment and configuration

❏ English-speaking markets are primary target

Support innovation in existing markets

❏ Reduce cost of new functionality and processes

❏ Increase accessibility of platform/application metrics for analysis

Enable scalable business

❏ More customers and increased transactions

Architectural Principles

Eliminate accidental design/code complexity

❏ Aggressively retire and replace unnecessary complex code and process

Services with single responsibility principle

❏ High cohesion, low coupling throughout stack

Consistent interfaces and data flows

❏ No surprises

❏ Extensible

Reduce inertia

❏ Make choices that favour rapid feedback and change

Design and Delivery Practices

Encapsulate legacy code and create seams/interfaces

Java/JS/Golang assessed per service (architect council led)

Utilise REST (JSON/HTTP)

RabbitMQ messaging for async communication

Consolidate, fix & cleanse data per service created

Continuous delivery for all

Automate testing

Better align business goals with development

❏ Everybody, all together, from early on

❏ Identify KPIs/success metrics

TACTICS: TECHNICAL LEADERSHIP IS VITAL

- PROMOTE SHARED UNDERSTANDING
  - COMMUNICATION (BIT.LY/1IA3U80)
- RISK MANAGEMENT
- 'JUST ENOUGH' UP-FRONT DESIGN
  - MICROSERVICE BOUNDARIES AND 'GLUE'

O'REILLY
Microservice Architecture
ALIGNING PRINCIPLES, PRACTICES, AND CULTURE
Mike Amundsen, Matt McLarty, Ronnie Mitra & Irakli Nadareishvili
Talking with Tech Leads
From Novices to Practitioners
Patrick Kua
Foreword by Jim Webber
coding the architecture
Software Architecture for Developers
Technical leadership by coding, coaching, collaboration, architecture sketching and just enough up front design
Simon Brown

TACTICS: TECHNICAL LEADERSHIP IS VITAL

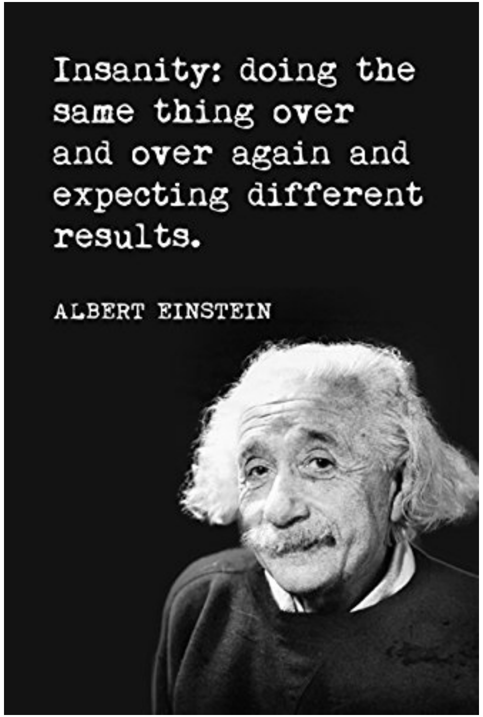
- CONWAY'S LAW IS WELL ACCEPTED
  - ALIGN TEAMS WITH SERVICE (VICE VERSA)
- NOT SO CLEAR WHERE 'ARCHITECTS' SIT
  - OVERARCHING, CONSULTING, OR PER TEAM?
- HIGHLY RECOMMENDED TO PAIR PRODUCT MANAGERS WITH TECH LEADS PER TEAM

O'REILLY
Microservice Architecture
ALIGNING PRINCIPLES, PRACTICES, AND CULTURE
Mike Amundsen, Matt McLarty, Ronnie Mitra & Irakli Nadareishvili
Talking with Tech Leads
From Novices to Practitioners
Patrick Kua
Foreword by Jim Webber
coding the architecture
Software Architecture for Developers
Technical leadership by coding, coaching, collaboration, architecture sketching and just enough up front design
Simon Brown

ANTIPATTERN

ANTIPATTERN: TECHNICAL INSANITY

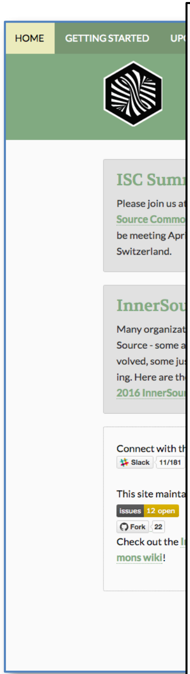
- WE CREATED A TECHNICAL MESS WITH A MONOLITH…
- BUT NO CHANGE REQUIRED WITH OUR APPROACH TO ARCHITECTURE WHEN IMPLEMENTING MICROSERVICES???
- WE NEED STRONG TECHNICAL LEADERSHIP



Insanity: doing the same thing over and over again and expecting different results.

ALBERT EINSTEIN

TEAMS AND TECHNOLOGY: INNERSOURCE



ISC Sum

Please join us a
Source Commo
be meeting Apr
Switzerland.

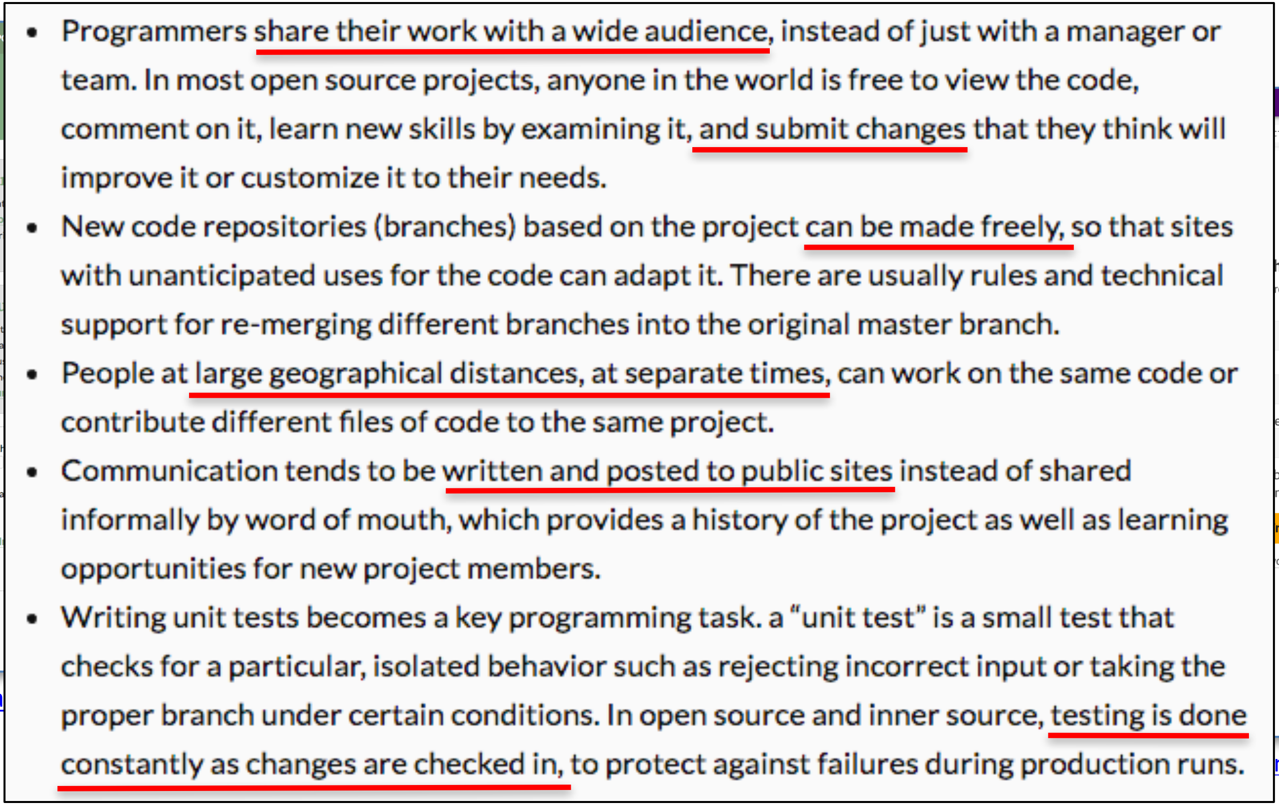
InnerSou

Many organizat
Source - some a
volved, some ju
ing. Here are th
2016 InnerSou

Connect with t
Slack 11/181

This site mainta
issues 12 open
Fork 22
Check out the I
mons wiki!

paypa



- Programmers share their work with a wide audience, instead of just with a manager or team. In most open source projects, anyone in the world is free to view the code, comment on it, learn new skills by examining it, and submit changes that they think will improve it or customize it to their needs.
- New code repositories (branches) based on the project can be made freely, so that sites with unanticipated uses for the code can adapt it. There are usually rules and technical support for re-merging different branches into the original master branch.
- People at large geographical distances, at separate times, can work on the same code or contribute different files of code to the same project.
- Communication tends to be written and posted to public sites instead of shared informally by word of mouth, which provides a history of the project as well as learning opportunities for new project members.
- Writing unit tests becomes a key programming task. a "unit test" is a small test that checks for a particular, isolated behavior such as rejecting incorrect input or taking the proper branch under certain conditions. In open source and inner source, testing is done constantly as changes are checked in, to protect against failures during production runs.

he form below
required.

ess

be to the O'Reilly
ng Newsletter

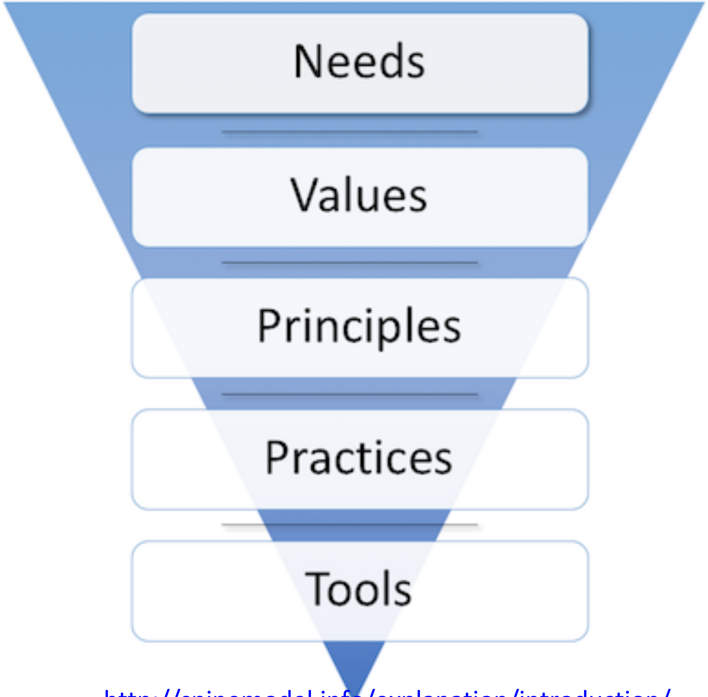
ree Report

our privacy.

nersource.csp

EVALUATING TOOLING - THE SPINE MODEL

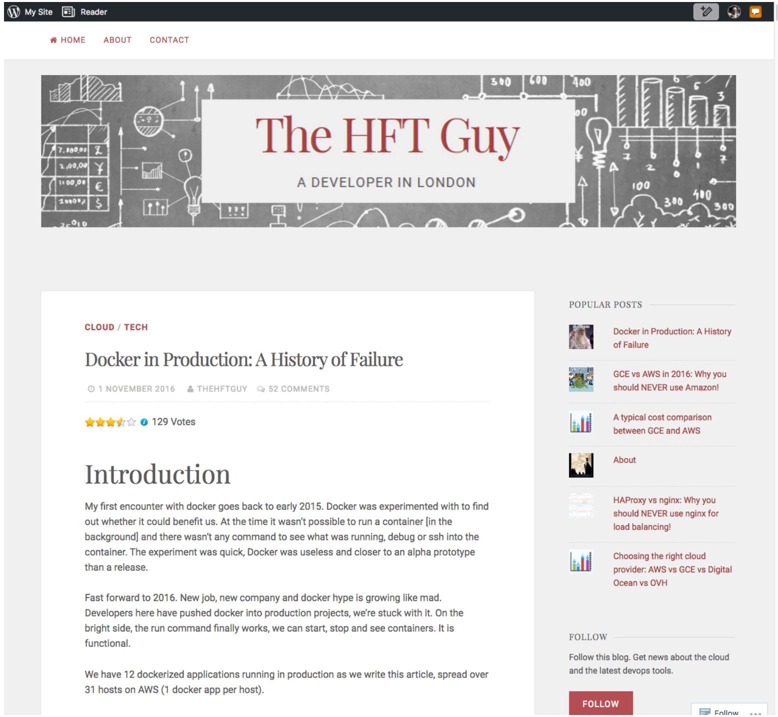
- EFFECTIVE CONVERSATIONS MAKE FOR EFFECTIVE COLLABORATION WITHIN A TEAM
  - KEVIN TRETHEWEY & DANIE ROUX, AGILE 2015
- IT'S A TOOL PROBLEM
  - AS A SPECIES, WE HAVE ALWAYS BEEN TOOL USERS AND MAKERS.
  - WE USE ______ TO GET OUR WORK DONE
- PEOPLE GET STUCK IN A DILEMMA WHERE EQUALLY PLAUSIBLE OPTIONS ARE AVAILABLE
  - "GOING UP THE SPINE" BREAKS DEADLOCK



http://spinemodel.info/explanation/introduction/

SpectoLabs

CHOICES: BEWARE OF CONFIRMATION BIAS



CLOUD / TECH

Docker in Production: A History of Failure

1 NOVEMBER 2016 THEHFTGUY 52 COMMENTS

★★★½☆ 129 Votes

Introduction

My first encounter with docker goes back to early 2015. Docker was experimented with to find out whether it could benefit us. At the time it wasn't possible to run a container [in the background] and there wasn't any command to see what was running, debug or ssh into the container. The experiment was quick, Docker was useless and closer to an alpha prototype than a release.

Fast forward to 2016. New job, new company and docker hype is growing like mad. Developers here have pushed docker into production projects, we're stuck with it. On the bright side, the run command finally works, we can start, stop and see containers. It is functional.

We have 12 dockerized applications running in production as we write this article, spread over 31 hosts on AWS (1 docker app per host).

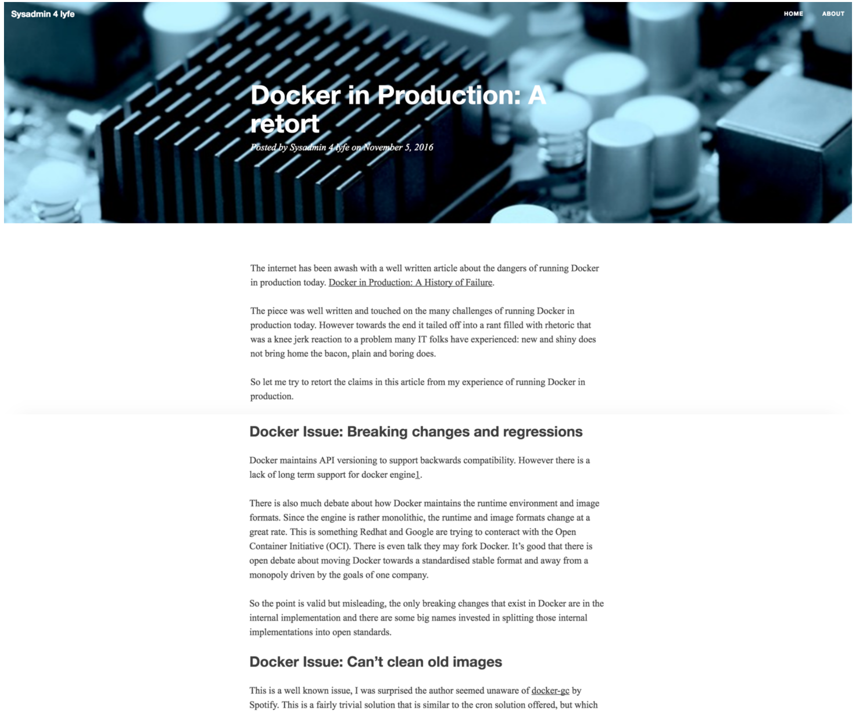
POPULAR POSTS

- Docker in Production: A History of Failure
- GCE vs AWS in 2016: Why you should NEVER use Amazon!
- A typical cost comparison between GCE and AWS
- About
- HAProxy vs nginx: Why you should NEVER use nginx for load balancing!
- Choosing the right cloud provider: AWS vs GCE vs Digital Ocean vs OVH

FOLLOW

Follow this blog. Get news about the cloud and the latest devops tools.

https://thehftguy.wordpress.com/2016/11/01/docker-in-production-an-history-of-failure/



Docker in Production: A retort

Posted by Sysadmin 4 lyfe on November 5, 2016

The internet has been awash with a well written article about the dangers of running Docker in production today. Docker in Production: A History of Failure.

The piece was well written and touched on the many challenges of running Docker in production today. However towards the end it tailed off into a rant filled with rhetoric that was a knee jerk reaction to a problem many IT folks have experienced: new and shiny does not bring home the bacon, plain and boring does.

So let me try to retort the claims in this article from my experience of running Docker in production.

Docker Issue: Breaking changes and regressions

Docker maintains API versioning to support backwards compatibility. However there is a lack of long term support for docker engine1.

There is also much debate about how Docker maintains the runtime environment and image formats. Since the engine is rather monolithic, the runtime and image formats change at a great rate. This is something Redhat and Google are trying to conteract with the Open Container Initiative (OCI). There is even talk they may fork Docker. It's good that there is open debate about moving Docker towards a standardised stable format and away from a monopoly driven by the goals of one company.

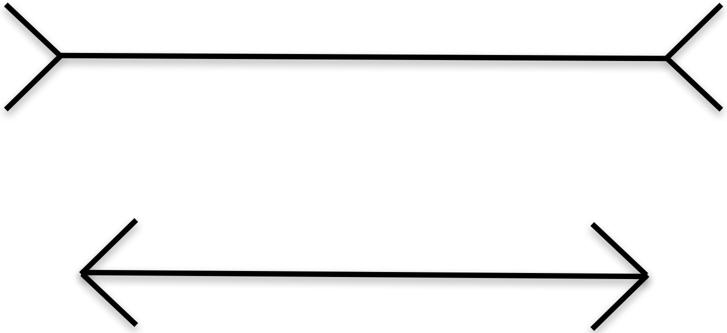
So the point is valid but misleading, the only breaking changes that exist in Docker are in the internal implementation and there are some big names invested in splitting those internal implementations into open standards.

Docker Issue: Can't clean old images

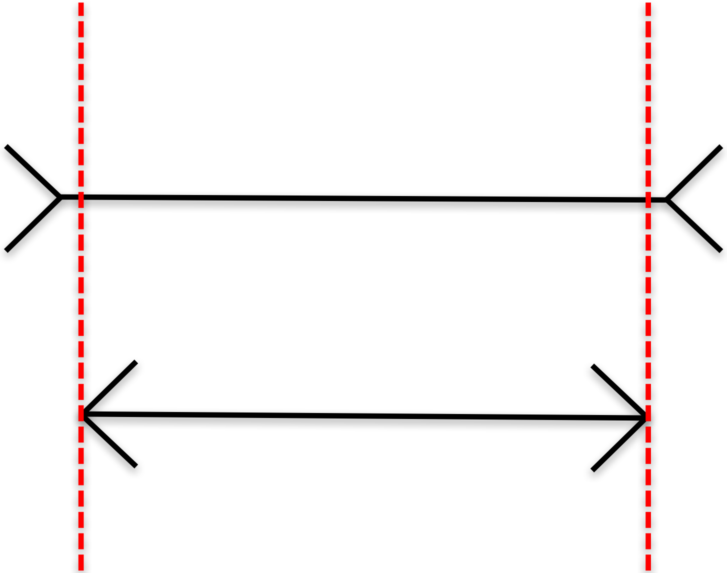
This is a well known issue, I was surprised the author seemed unaware of docker-gc by Spotify. This is a fairly trivial solution that is similar to the cron solution offered, but which

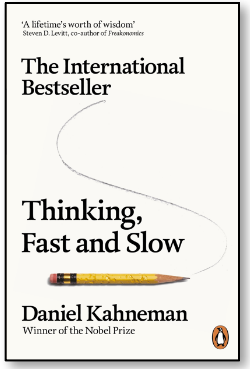
http://patrobinson.github.io/2016/11/05/docker-in-production/

EVALUATION - IT'S EASY TO BE TRICKED



EVALUATION - BEWARE OF BIAS AND HEURISTICS





'A lifetime's worth of wisdom'
Steven D. Levitt, co-author of *Freakonomics*

The International Bestseller

Thinking, Fast and Slow

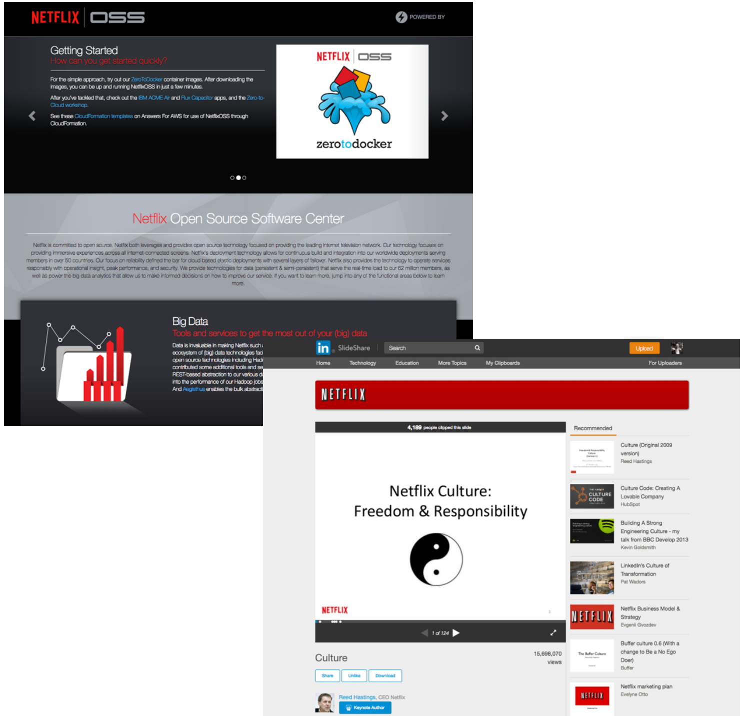
Daniel Kahneman
Winner of the Nobel Prize

ANTIPATTERN

ANTIPATTERN: NETFLIX* (TECHNICAL) ENVY

- BLINDLY COPYING TECH FROM *NETFLIX/GOOGLE/AMAZON/TWITTER

- LEARN ABOUT THE PRINCIPLES, PRACTICES AND CULTURE AS WELL



Getting Started

How can you get started quickly?

For the simple approach, try out our ZeroToDocker container images. After downloading the images, you can be up and running NetflixOSS in just a few minutes.

After you've tackled that, check out the IBM ACME Air and Flux Capacitor apps, and the Zero-to-Cloud workshop.

See these CloudFormation templates on Answers For AWS for use of NetflixOSS through CloudFormation.

Netflix Open Source Software Center

Netflix is committed to open source. Netflix both leverages and provides open source technology focused on providing the leading Internet television network. Our technology focuses on providing immersive experiences across all internet-connected screens. Netflix's deployment technology allows for continuous build and integration into our worldwide deployments serving members in over 50 countries. Our focus on reliability defined the bar for cloud based elastic deployments with several layers of failover. Netflix also provides the technology to operate services responsibly with operational insight, peak performance, and security. We provide technologies for data (persistent & semi-persistent) that serve the real-time load to our 62 million members, as well as power the big data analytics that allow us to make informed decisions on how to improve our service. If you want to learn more, jump into any of the functional areas below to learn more.

Big Data

Tools and services to get the most out of your (big) data

Data is invaluable in making Netflix such a
ecosystem of (big) data technologies faci
open source technologies including Had
contributed some additional tools and se
REST-based abstraction to our various d
into the performance of our Hadoop jobs
And Aegisthus enables the bulk abstracti

4,189 people clipped this slide

Netflix Culture:
Freedom & Responsibility

Culture

15,698,070 views

Reed Hastings, CEO Netflix

Culture (Original 2009 version)
Reed Hastings

Culture Code: Creating A Lovable Company
HubSpot

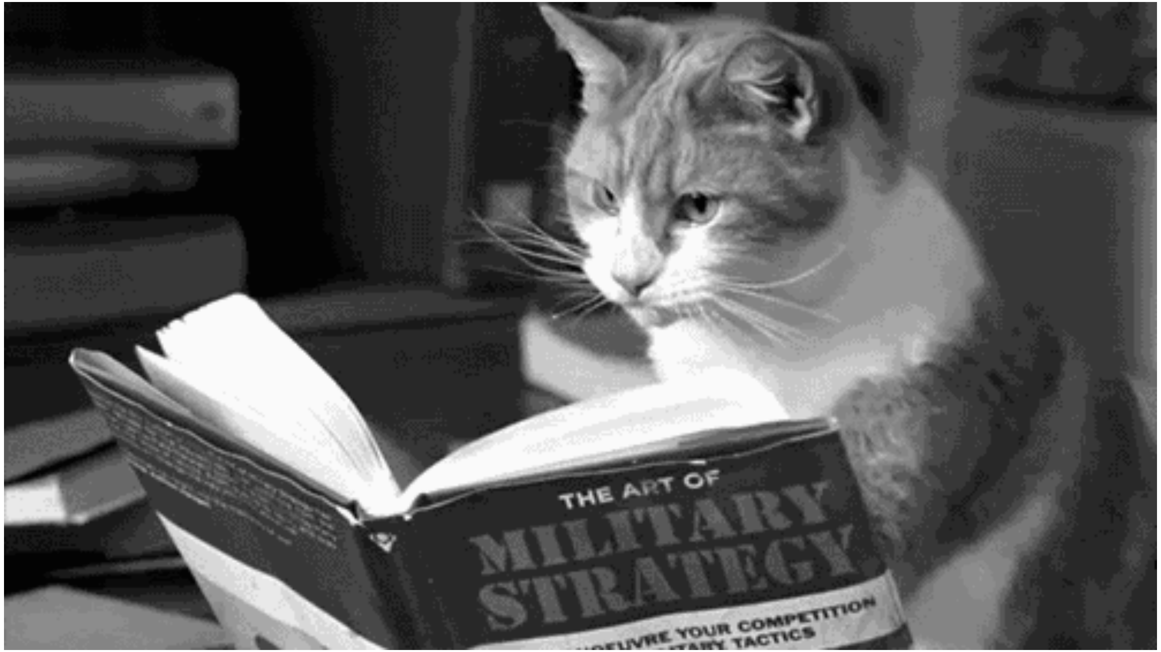
Building A Strong Engineering Culture - my talk from BBC Develop 2013
Kevin Goldsmith

LinkedIn's Culture of Transformation
Pat Wadors

Netflix Business Model & Strategy
Evgenii Gvozdev

Buffer culture 0.6 (With a change to Be a No Ego Doer)
Buffer

Netflix marketing plan
Evelyne Otto



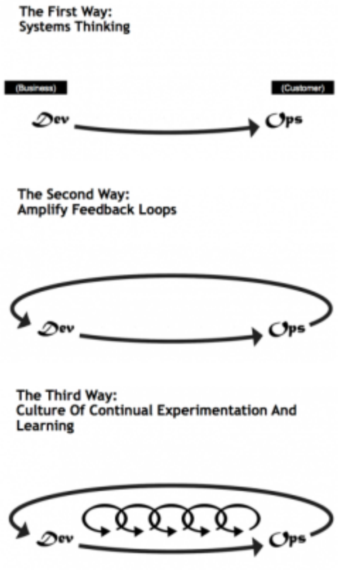
2. FEEDBACK – VISIBILITY AND CONSTANT LEARNING

FEEDBACK IS VITAL

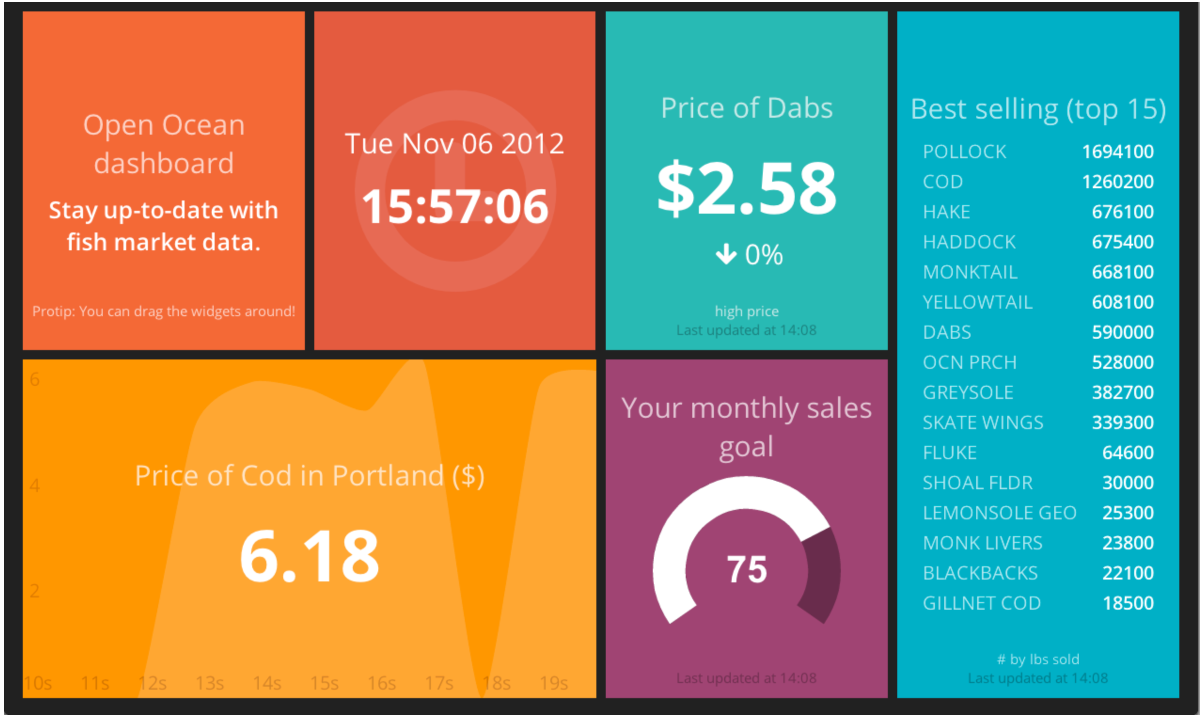
- BUSINESS
- ARCHITECTURE
- OPERATIONS

We assert that the Three Ways describe the values and philosophies that frame the processes, procedures, practices of DevOps, as well as the prescriptive steps.

Gene Kim



VISIBILITY FOR THE BUSINESS



Open Ocean dashboard

Stay up-to-date with fish market data.

Protip: You can drag the widgets around!

Tue Nov 06 2012

15:57:06

Price of Dabs

$2.58

↓ 0%

high price

Last updated at 14:08

Best selling (top 15)

POLLOCK	1694100
COD	1260200
HAKE	676100
HADDOCK	675400
MONKTAIL	668100
YELLOWTAIL	608100
DABS	590000
OCN PRCH	528000
GREYSOLE	382700
SKATE WINGS	339300
FLUKE	64600
SHOAL FLDR	30000
LEMONSOLE GEO	25300
MONK LIVERS	23800
BLACKBACKS	22100
GILLNET COD	18500

by lbs sold

Last updated at 14:08

Price of Cod in Portland ($)

6.18

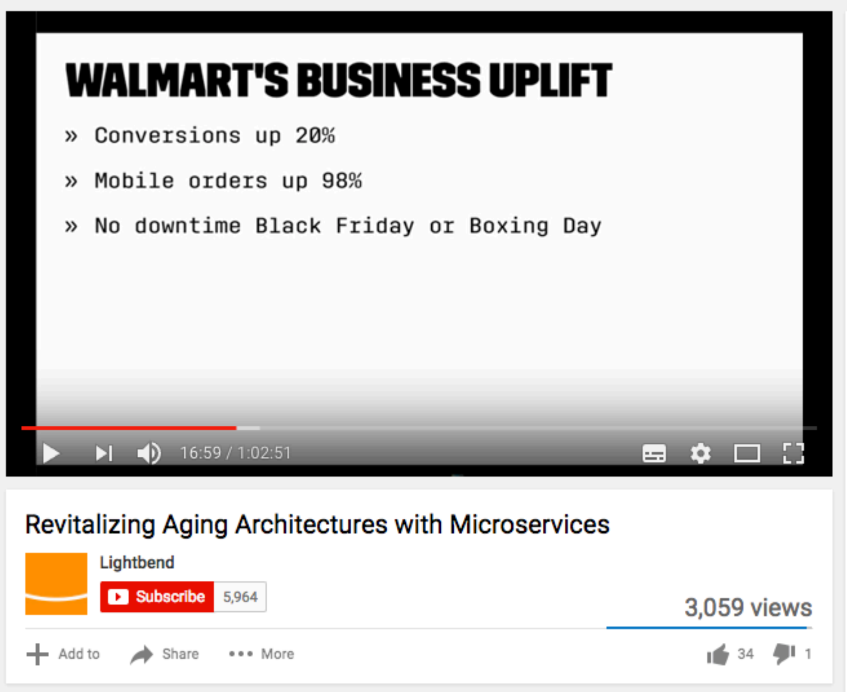
Your monthly sales goal

75

Last updated at 14:08

MICROSERVICES SHOULD BE BUSINESS-DRIVEN

- BAKE-IN METRICS AND SIGNALS WITHIN SERVICES AND PLATFORM
- ALLOWS VALIDATION OF HYPOTHESES
- SHARE REGULARLY THROUGHOUT ORG



WALMART'S BUSINESS UPLIFT

» Conversions up 20%

» Mobile orders up 98%

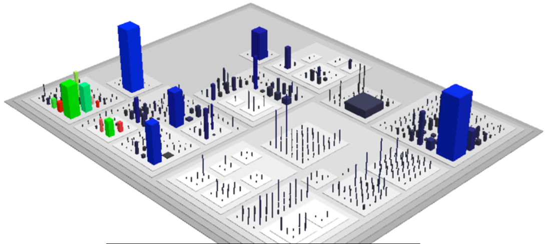
» No downtime Black Friday or Boxing Day

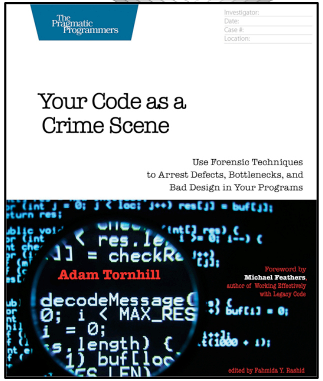
Revitalizing Aging Architectures with Microservices

Lightbend

www.youtube.com/watch?v=SPGCdziXlHU

ARCHITECTURAL FEEDBACK





The Pragmatic Programmers

Investigator:
Date:
Case #:
Location:

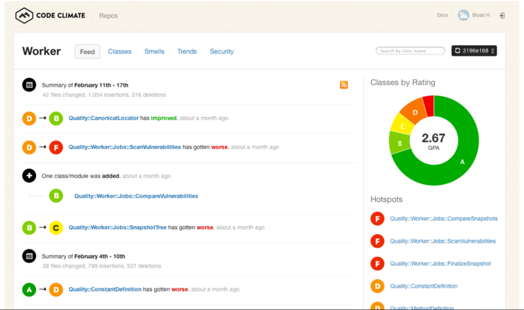
Your Code as a Crime Scene

Use Forensic Techniques to Arrest Defects, Bottlenecks, and Bad Design in Your Programs

Adam Tornhill

Foreword by Michael Feathers, author of Working Effectively with Legacy Code

edited by Fahmida Y. Rashid



Worker

Summary of **February 11th - 17th**
42 files changed, 1,054 insertions, 316 deletions

D → B Quality::CanonicalLocator has **improved**. about a month ago

D → F Quality::Worker::Jobs::ScanVulnerabilities has gotten **worse**. about a month ago

One class/module was **added**. about a month ago
B Quality::Worker::Jobs::CompareVulnerabilities

B → C Quality::Worker::Jobs::SnapshotTree has gotten **worse**. about a month ago

Summary of **February 4th - 10th**
38 files changed, 799 insertions, 527 deletions

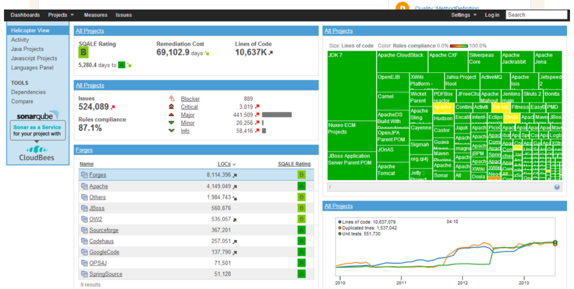
A → D Quality::ConstantDefinition has gotten **worse**. about a month ago

Classes by Rating

2.67 GPA

Hotspots

- F Quality::Worker::Jobs::CompareSnapshots
- F Quality::Worker::Jobs::ScanVulnerabilities
- F Quality::Worker::Jobs::FinalizeSnapshot
- D Quality::ConstantDefinition
- D Quality::MethodDefinition



All Projects

SQALE Rating B
5,280.4 days to A

Remediation Cost 69,102.9 days

Lines of Code 10,637K

All Projects

Issues 524,089

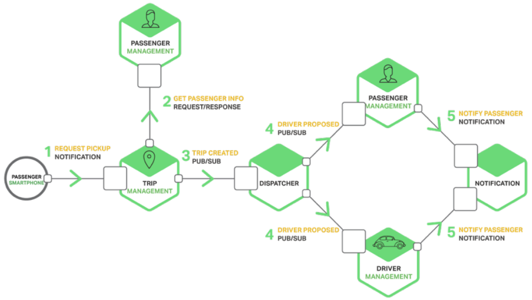
Rules compliance 87.1%

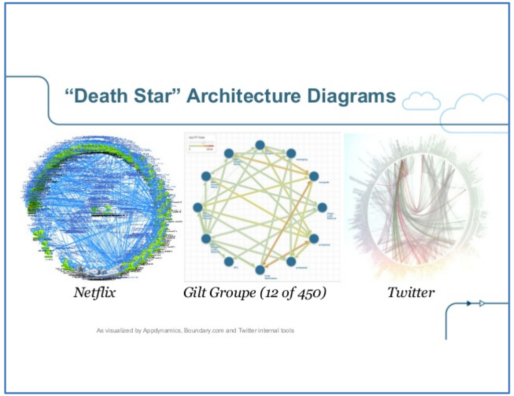
Blocker	889
Critical	3,019
Major	441,509
Minor	20,256
Info	58,416

Forges

Name	LOCs	SQALE Rating
Forges	8,114,396	B
Apache	4,149,049	A
Others	1,984,743	B
JBoss	560,876	B
OW2	535,057	B
Sourceforge	367,201	A
Codehaus	257,051	A
GoogleCode	137,790	A
OPS4J	71,501	A
SpringSource	51,128	A

9 results





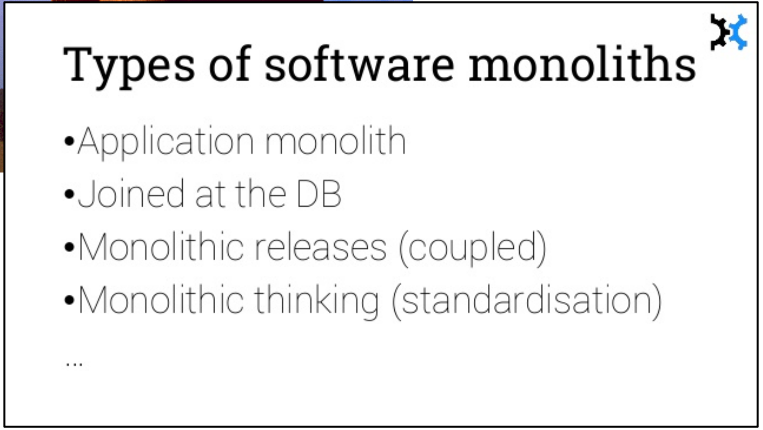
"Death Star" Architecture Diagrams

As visualized by Appdynamics, Boundary.com and Twitter internal tools

ANTIPATTERN

ANTIPATTERN: TROJAN MONOSERVICES

- ITS ALL TOO EASY FOR THE MONOLITH TO CREEP BACK IN (SOMEWHERE)...
- H/T MATTHEW SKELTON
- CONTINUALLY RETROSPECT ON TECHNICAL WORK USING SUPPORTING METRICS



Types of software monoliths

- Application monolith
- Joined at the DB
- Monolithic releases (coupled)
- Monolithic thinking (standardisation)

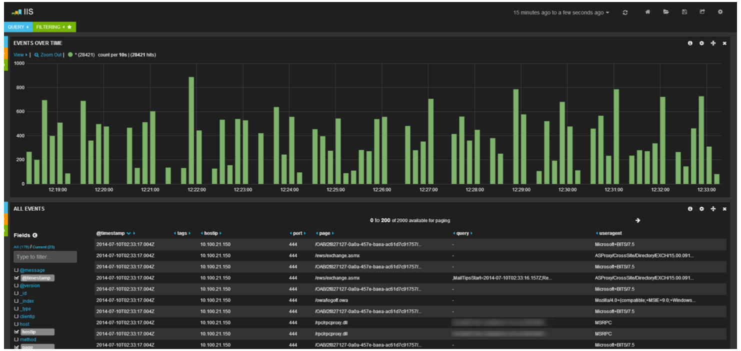
...

www.slideshare.net/SkeltonThatcher/teams-and-monoliths-matthew-skelton-londoncd-2016

OPERATIONAL VISIBILITY

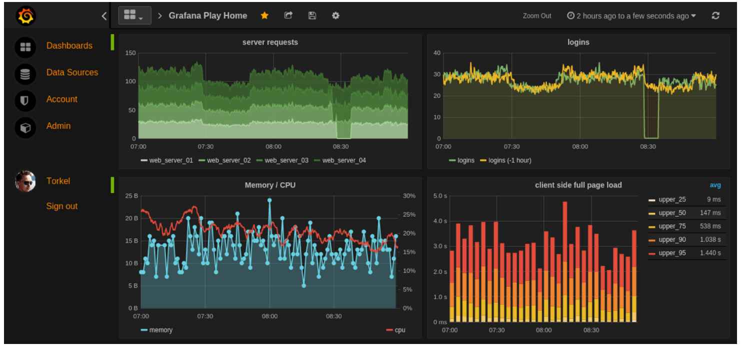
- LOGGING
  - THE 10 COMMANDMENTS OF LOGGING
  - THE LOG: WHAT EVERY SOFTWARE ENGINEER SHOULD KNOW

- MONITORING AND ALERTING
  - ROB EWASCHUK'S PHILOSOPHY ON ALERTING
  - BRENDAN GREGG'S USE METHOD



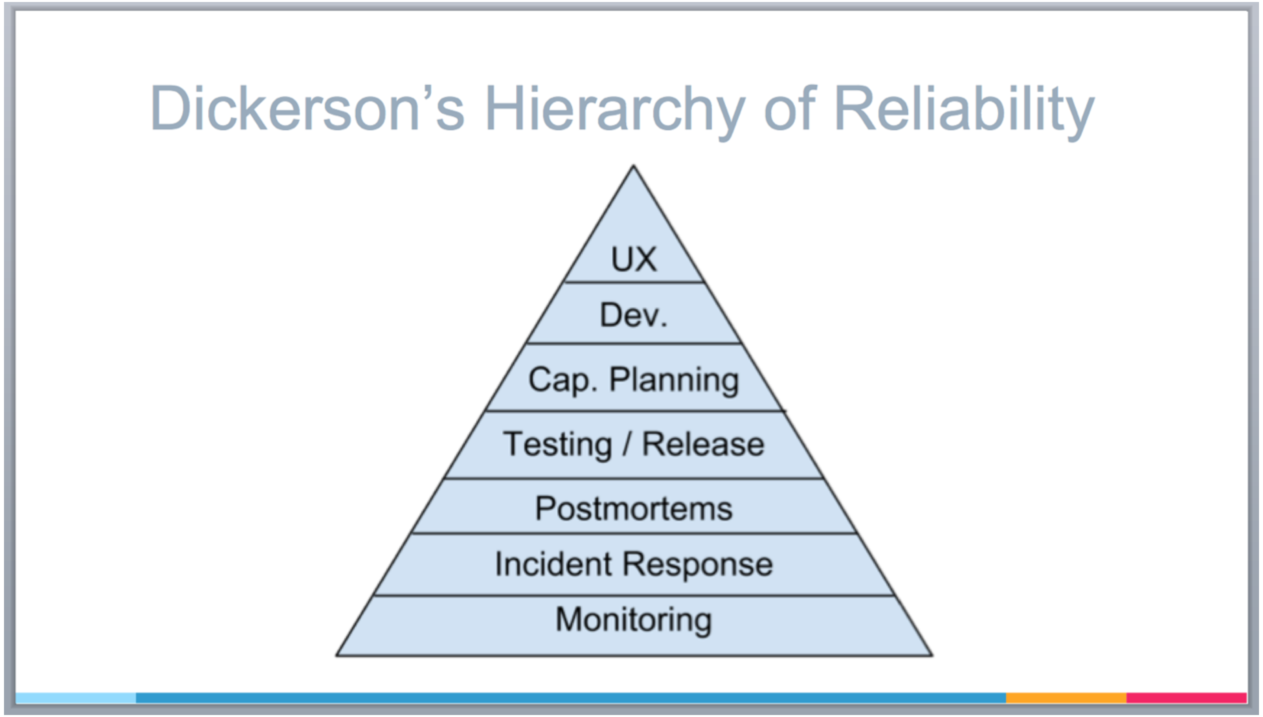
0 to 200 of 2000 available for paging

@timestamp	tags	hostip	port	page	query	useragent
2014-07-10T02:33:17.004Z		10.100.21.150	444	/OAB/2f827127-0a0a-457e-baea-ac61d7c91757/...	-	Microsoft+BITS/7.5
2014-07-10T02:33:17.004Z		10.100.21.150	444	/ews/exchange.asmx	-	ASProxy/CrossSite/Directory/EXCH/15.00.091...
2014-07-10T02:33:17.004Z		10.100.21.150	444	/OAB/2f827127-0a0a-457e-baea-ac61d7c91757/...	-	Microsoft+BITS/7.5
2014-07-10T02:33:17.004Z		10.100.21.150	444	/ews/exchange.asmx	,MailTipsStart=2014-07-10T02:33:16.157Z;Re...	ASProxy/CrossSite/Directory/EXCH/15.00.091...
2014-07-10T02:33:17.004Z		10.100.21.150	444	/OAB/2f827127-0a0a-457e-baea-ac61d7c91757/...	-	Microsoft+BITS/7.5
2014-07-10T02:33:17.004Z		10.100.21.150	444	/owa/logoff.owa	-	Mozilla/4.0+(compatible;+MSIE+9.0;+Windows...
2014-07-10T02:33:17.004Z		10.100.21.150	444	/OAB/2f827127-0a0a-457e-baea-ac61d7c91757/...	-	Microsoft+BITS/7.5
2014-07-10T02:33:17.004Z		10.100.21.150	444	/rpc/rpcproxy.dll	[redacted]	MSRPC
2014-07-10T02:33:17.004Z		10.100.21.150	444	/rpc/rpcproxy.dll	[redacted]	MSRPC
2014-07-10T02:33:17.004Z		10.100.21.150	444	/OAB/2f827127-0a0a-457e-baea-ac61d7c91757/...	-	Microsoft+BITS/7.5



WHEN BAD THINGS HAPPEN, PEOPLE ARE ALWAYS INVOLVED

MIKEY DICKERSON'S HIERARCHY OF RELIABILITY



Dickerson's Hierarchy of Reliability

UX

Dev.

Cap. Planning

Testing / Release

Postmortems

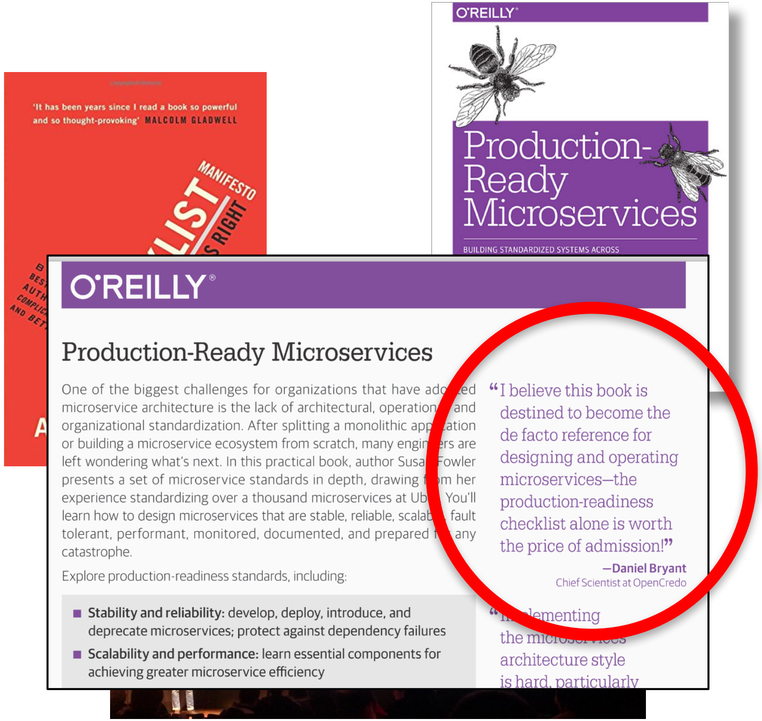
Incident Response

Monitoring

www.infoq.com/news/2015/06/too-big-to-fail

A LITTLE BIT OF STANDARDISATION GOES A LONG WAY

- AUTOMATION IS THE GOAL
  - ESSENTIAL AT SCALE WITH MICROSERVICES
- BUT WE HAVE TO UNDERSTAND PROBLEMS
  - RESPONSE, POSTMORTEMS & ROOT CAUSE
- CHECKLISTS PROVIDE STRUCTURE


'It has been years since I read a book so powerful and so thought-provoking' MALCOLM GLADWELL
MANIFESTO
LIST
S RIGHT
O'REILLY
Production-Ready Microservices
BUILDING STANDARDIZED SYSTEMS ACROSS

O'REILLY

Production-Ready Microservices

One of the biggest challenges for organizations that have ado ed microservice architecture is the lack of architectural, operation and organizational standardization. After splitting a monolithic app ation or building a microservice ecosystem from scratch, many engin rs are left wondering what's next. In this practical book, author Susa Fowler presents a set of microservice standards in depth, drawing om her experience standardizing over a thousand microservices at Ub You'll learn how to design microservices that are stable, reliable, scalab fault tolerant, performant, monitored, documented, and prepared f any catastrophe.

Explore production-readiness standards, including:

- **Stability and reliability:** develop, deploy, introduce, and deprecate microservices; protect against dependency failures
- **Scalability and performance:** learn essential components for achieving greater microservice efficiency

> "I believe this book is destined to become the de facto reference for designing and operating microservices—the production-readiness checklist alone is worth the price of admission!"
>
> —Daniel Bryant
> Chief Scientist at OpenCredo

"Implementing the microservices architecture style is hard, particularly

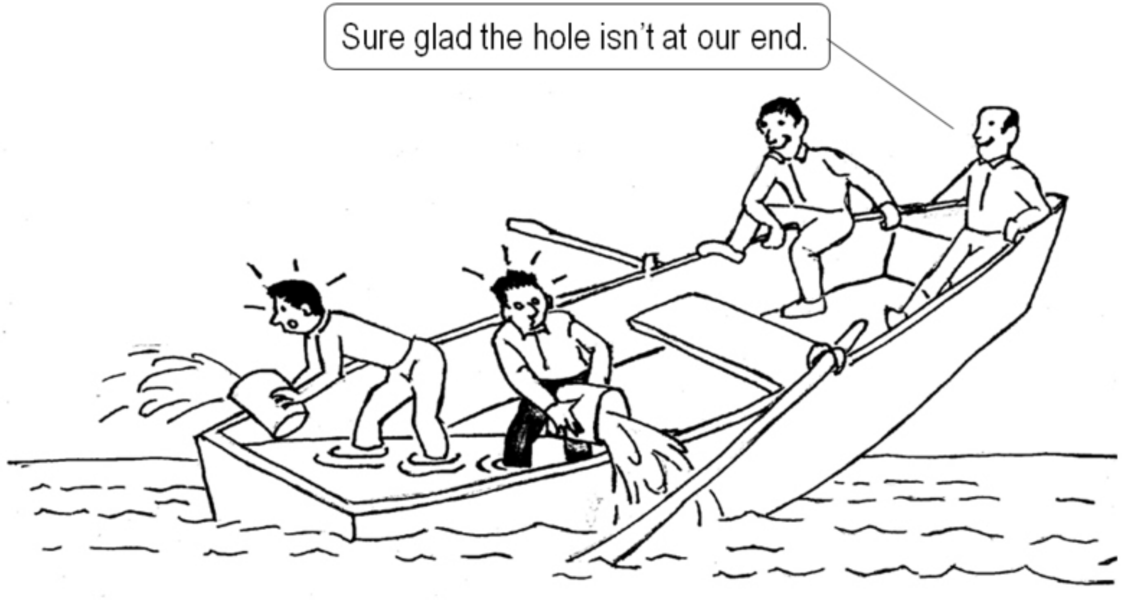
www.microservices.com/talks/microservices-standardization-susan-fowler/

MICROSERVICES ENABLE AGILITY

- WHEN DONE WELL…

- BUILD, MEASURE, LEARN
  - BUILD-IN SIGNALS AND METRICS
  - CREATE A CULTURE OF EXPERIMENTATION AND FAILING FAST

- IF YOU DON'T COLLECT THE DATA AND TAKE ACTION TO ADAPT…
  - THERE IS LIMITED BENEFIT WITH MICROSERVICES



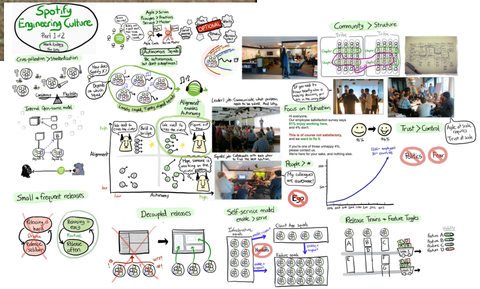
3. RESPONSIBILITIES - THE BUCK ALWAYS STOPS SOMEWHERE

WE HEAR THIS A LOT…

"WE'VE DECIDED TO REFORM OUR TEAMS AROUND SQUADS, CHAPTERS AND GUILDS"

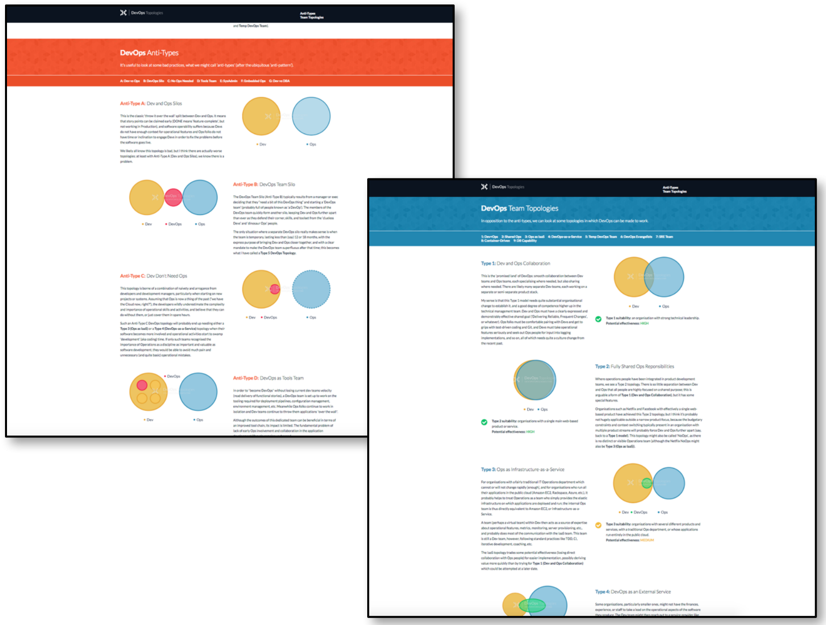
- BEWARE OF CARGO-CULTING
  - REPEAT THREE TIMES "WE ARE NOT SPOTIFY"

- UNDERSTAND THE PRACTICES, PRINCIPLES, VALUES ETC



PEOPLE PAIN POINT - HOW DOES DEVOPS FIT INTO THIS?

- DEVOPS TOPOLOGIES
  - HTTP://WEB.DEVOPSTOPOLOGIES.COM/
  - @MATTHEWPSKELTON



and Temp DevOps Team).

DevOps Anti-Types

It's useful to look at some bad practices, what we might call 'anti-types' (after the ubiquitous 'anti-pattern').

A: Dev vs Ops B: DevOps Silo C: No Ops Needed D: Tools Team E: SysAdmin F: Embedded Ops G: Dev vs DBA

Anti-Type A: Dev and Ops Silos

This is the classic 'throw it over the wall' split between Dev and Ops. It means that story points can be claimed early (DONE means 'feature-complete', but not working in Production), and software operability suffers because Devs do not have enough context for operational features and Ops folks do not have time or inclination to engage Devs in order to fix the problems before the software goes live.

We likely all know this topology is bad, but I think there are actually worse topologies; at least with Anti-Type A (Dev and Ops Silos), we know there is a problem.

Anti-Type B: DevOps Team Silo

The DevOps Team Silo (Anti-Type B) typically results from a manager or exec deciding that they "need a bit of this DevOps thing" and starting a 'DevOps team' (probably full of people known as 'a DevOp'). The members of the DevOps team quickly form another silo, keeping Dev and Ops further apart than ever as they defend their corner, skills, and toolset from the 'clueless Devs' and 'dinosaur Ops' people.

The only situation where a separate DevOps silo really makes sense is when the team is temporary, lasting less than (say) 12 or 18 months, with the express purpose of bringing Dev and Ops closer together, and with a clear mandate to make the DevOps team superfluous after that time; this becomes what I have called a **Type 5 DevOps Topology**.

Anti-Type C: Dev Don't Need Ops

This topology is borne of a combination of naivety and arrogance from developers and development managers, particularly when starting on new projects or systems. Assuming that Ops is now a thing of the past ("we have the Cloud now, right?"), the developers wildly underestimate the complexity and importance of operational skills and activities, and believe that they can do without them, or just cover them in spare hours.

Such an Anti-Type C DevOps topology will probably end up needing either a **Type 3 (Ops as IaaS)** or a **Type 4 (DevOps-as-a-Service)** topology when their software becomes more involved and operational activities start to swamp 'development' (aka coding) time. If only such teams recognised the importance of Operations as a discipline as important and valuable as software development, they would be able to avoid much pain and unnecessary (and quite basic) operational mistakes.

Anti-Type D: DevOps as Tools Team

In order to "become DevOps" without losing current dev teams velocity (read delivery of functional stories), a DevOps team is set up to work on the tooling required for deployment pipelines, configuration management, environment management, etc. Meanwhile Ops folks continue to work in isolation and Dev teams continue to throw them applications "over the wall".

Although the outcomes of this dedicated team can be beneficial in terms of an improved tool chain, its impact is limited. The fundamental problem of lack of early Ops involvement and collaboration in the application

DevOps Team Topologies

In opposition to the anti-types, we can look at some topologies in which DevOps can be made to work.

1: Dev+Ops 2: Shared Ops 3: Ops as IaaS 4: DevOps-as-a-Service 5: Temp DevOps Team 6: DevOps Evangelists 7: SRE Team 8: Container-Driven 9: DB Capability

Type 1: Dev and Ops Collaboration

This is the 'promised land' of DevOps: smooth collaboration between Dev teams and Ops teams, each specialising where needed, but also sharing where needed. There are likely many separate Dev teams, each working on a separate or semi-separate product stack.

My sense is that this Type 1 model needs quite substantial organisational change to establish it, and a good degree of competence higher up in the technical management team. Dev and Ops must have a clearly expressed and demonstrably effective shared goal ('Delivering Reliable, Frequent Changes', or whatever). Ops folks must be comfortable pairing with Devs and get to grips with test-driven coding and Git, and Devs must take operational features seriously and seek out Ops people for input into logging implementations, and so on, all of which needs quite a culture change from the recent past.

✓ **Type 1 suitability:** an organisation with strong technical leadership.
Potential effectiveness: HIGH

✓ **Type 2 suitability:** organisations with a single main web-based product or service.
Potential effectiveness: HIGH

Type 2: Fully Shared Ops Reponsibilities

Where operations people have been integrated in product development teams, we see a Type 2 topology. There is so little separation between Dev and Ops that all people are highly focused on a shared purpose; this is arguable a form of **Type 1 (Dev and Ops Collaboration)**, but it has some special features.

Organisations such as Netflix and Facebook with effectively a single web-based product have achieved this Type 2 topology, but I think it's probably not hugely applicable outside a narrow product focus, because the budgetary constraints and context-switching typically present in an organisation with multiple product streams will probably force Dev and Ops further apart (say, back to a **Type 1 model**). This topology might also be called 'NoOps', as there is no distinct or visible Operations team (although the Netflix NoOps might also be **Type 3 (Ops as IaaS)**).

Type 3: Ops as Infrastructure-as-a-Service

For organisations with a fairly traditional IT Operations department which cannot or will not change rapidly (enough), and for organisations who run all their applications in the public cloud (Amazon EC2, Rackspace, Azure, etc.), it probably helps to treat Operations as a team who simply provides the elastic infrastructure on which applications are deployed and run; the internal Ops team is thus directly equivalent to Amazon EC2, or Infrastructure-as-a-Service.

A team (perhaps a virtual team) within Dev then acts as a source of expertise about operational features, metrics, monitoring, server provisioning, etc., and probably does most of the communication with the IaaS team. This team is still a Dev team, however, following standard practices like TDD, CI, iterative development, coaching, etc.

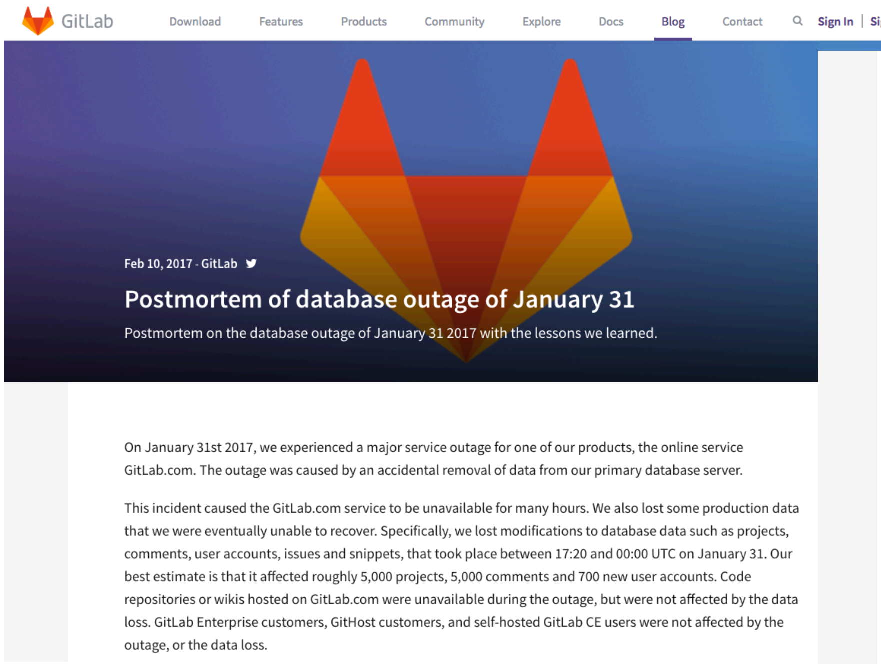
The IaaS topology trades some potential effectiveness (losing direct collaboration with Ops people) for easier implementation, possibly deriving value more quickly than by trying for **Type 1 (Dev and Ops Collaboration)** which could be attempted at a later date.

✓ **Type 3 suitability:** organisations with several different products and services, with a traditional Ops department, or whose applications run entirely in the public cloud.
Potential effectiveness: MEDIUM

Type 4: DevOps as an External Service

Some organisations, particularly smaller ones, might not have the finances, experience, or staff to take a lead on the operational aspects of the software they produce. The Dev team might then reach out to a service provider like

KUDOS TO GITLAB FOR THEIR POSTMORTEM



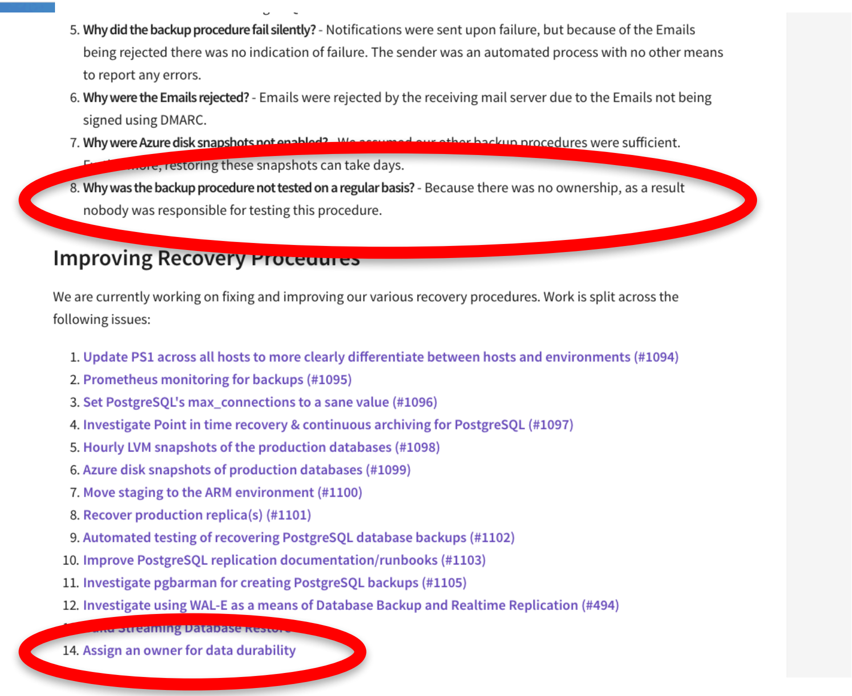
Feb 10, 2017 - GitLab

Postmortem of database outage of January 31

Postmortem on the database outage of January 31 2017 with the lessons we learned.

On January 31st 2017, we experienced a major service outage for one of our products, the online service GitLab.com. The outage was caused by an accidental removal of data from our primary database server.

This incident caused the GitLab.com service to be unavailable for many hours. We also lost some production data that we were eventually unable to recover. Specifically, we lost modifications to database data such as projects, comments, user accounts, issues and snippets, that took place between 17:20 and 00:00 UTC on January 31. Our best estimate is that it affected roughly 5,000 projects, 5,000 comments and 700 new user accounts. Code repositories or wikis hosted on GitLab.com were unavailable during the outage, but were not affected by the data loss. GitLab Enterprise customers, GitHost customers, and self-hosted GitLab CE users were not affected by the outage, or the data loss.



5. **Why did the backup procedure fail silently?** - Notifications were sent upon failure, but because of the Emails being rejected there was no indication of failure. The sender was an automated process with no other means to report any errors.
6. **Why were the Emails rejected?** - Emails were rejected by the receiving mail server due to the Emails not being signed using DMARC.
7. **Why were Azure disk snapshots not enabled?** - We assumed our other backup procedures were sufficient. Furthermore, restoring these snapshots can take days.
8. **Why was the backup procedure not tested on a regular basis?** - Because there was no ownership, as a result nobody was responsible for testing this procedure.

Improving Recovery Procedures

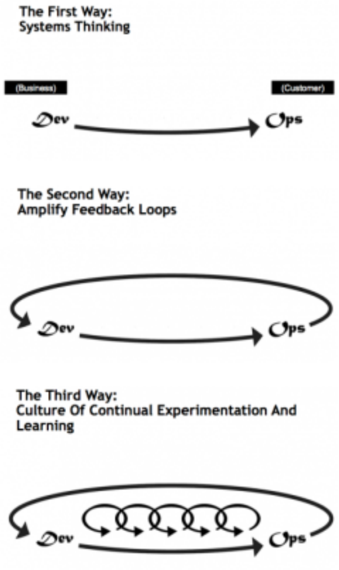
We are currently working on fixing and improving our various recovery procedures. Work is split across the following issues:

1. Update PS1 across all hosts to more clearly differentiate between hosts and environments (#1094)
2. Prometheus monitoring for backups (#1095)
3. Set PostgreSQL's max_connections to a sane value (#1096)
4. Investigate Point in time recovery & continuous archiving for PostgreSQL (#1097)
5. Hourly LVM snapshots of the production databases (#1098)
6. Azure disk snapshots of production databases (#1099)
7. Move staging to the ARM environment (#1100)
8. Recover production replica(s) (#1101)
9. Automated testing of recovering PostgreSQL database backups (#1102)
10. Improve PostgreSQL replication documentation/runbooks (#1103)
11. Investigate pgbarman for creating PostgreSQL backups (#1105)
12. Investigate using WAL-E as a means of Database Backup and Realtime Replication (#494)
13. ...and Streaming Database Restore
14. Assign an owner for data durability

DEVOPS - RESPONSIBILITIES

We assert that the Three Ways describe the values and philosophies that frame the processes, procedures, practices of DevOps, as well as the prescriptive steps.

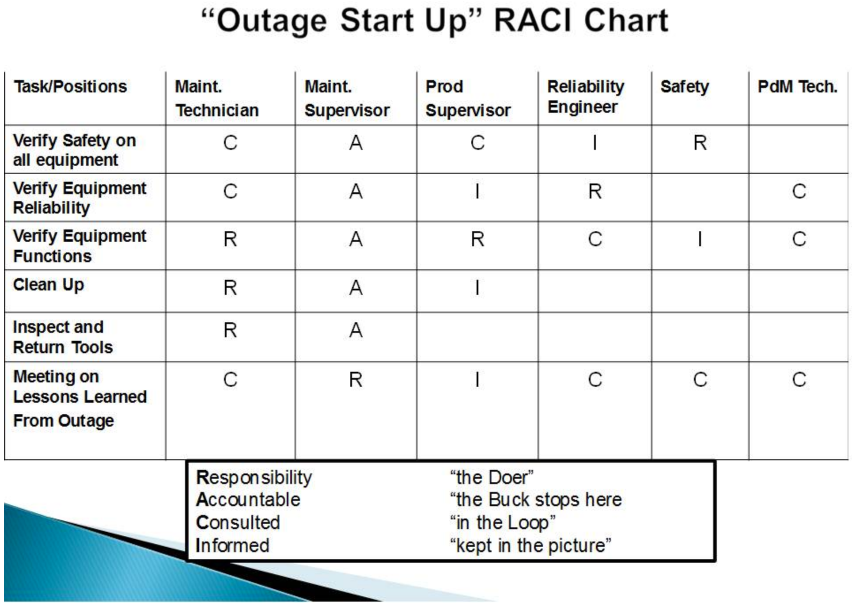
Gene Kim



The First Way:
Systems Thinking

The Second Way:
Amplify Feedback Loops

The Third Way:
Culture Of Continual Experimentation And Learning



“Outage Start Up” RACI Chart

Task/Positions	Maint. Technician	Maint. Supervisor	Prod Supervisor	Reliability Engineer	Safety	PdM Tech.
Verify Safety on all equipment	C	A	C	I	R	
Verify Equipment Reliability	C	A	I	R		C
Verify Equipment Functions	R	A	R	C	I	C
Clean Up	R	A	I			
Inspect and Return Tools	R	A				
Meeting on Lessons Learned From Outage	C	R	I	C	C	C

Responsibility	“the Doer”
Accountable	“the Buck stops here
Consulted	“in the Loop”
Informed	“kept in the picture”

DEVOPS - DEFINE RESPONSIBILITIES

- FOCUS ON WHAT MATTERS
  - CI/CD
  - MECHANICAL SYMPATHY
  - LOGGING
  - MONITORING
  - NFRS / CFRS
  - CHANGE MANAGEMENT
  - INCIDENT RESOLUTION

The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble
David Farley
Foreword by
O'REILLY
Site Reliability Engineering
HOW GOOGLE RUNS PRODUCTION SYSTEMS
Edited by Betsy Beyer, Chris Jones, Jennifer Petoff & Niall Richard Murphy
O'REILLY
Effective DevOps
BUILDING A CULTURE OF COLLABORATION, AFFINITY, AND TOOLING AT SCALE
Jennifer Davis & Katherine Daniels

HOW MUCH VALUE DOES NON-DEPLOYED CODE PROVIDE TO USERS?

0*

(*Universal Unit of Value)

ANTIPATTERN

ANTIPATTERN: WATER-MICRO-FALL

- CRAFTING PERFECT MICROSERVICE PLATFORMS / CHASSIS / DOCS
- BUILDING CONTINUALLY ON UNDEPLOYED (AND UNTESTED) ASSUMPTIONS
- CHANGE MINDSET TO CONTINUOUSLY DELIVER INCREMENTAL CHANGES TO PRODUCTION ASAP

IN CONCLUSION: MICROSERVICES WILL CREATE CHANGE...

CHANGE MANAGEMENT IS ESSENTIAL

- FAIR PROCESS (THREE 'E'S)
  - ENGAGEMENT
  - EXPLANATION
  - EXPECTATION

- LEADING CHANGE
  - TRANSFORMATION IS A PROCESS
  - COMMUNICATE, PLAN, EVALUATE, LEARN, EMPOWER
  - OBTAIN BUY-IN FROM THE TOP

Copyrighted Material

HBR'S 10 MUST READS

FEATURING "Leading Change" By John P. Kotter

On Change Management

If you read nothing else on change management, read these definitive articles from **Harvard Business Review.**

Copyrighted Material

HBR'S 10 MUST READS

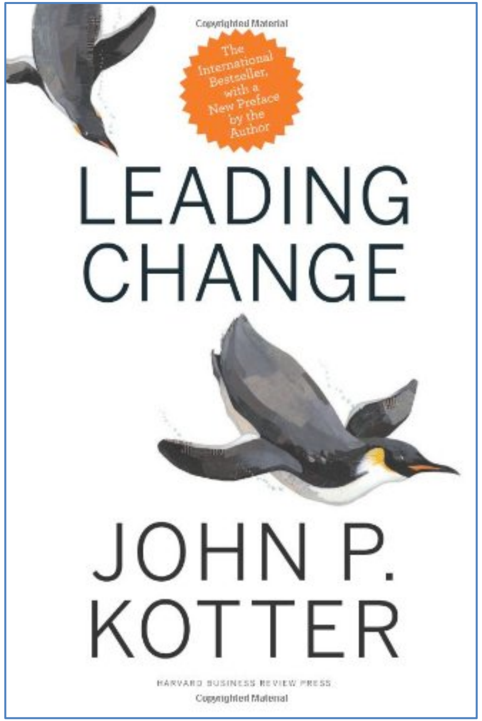
The Essentials

An introduction to the most enduring ideas on management from **Harvard Business Review.**

Copyrighted Material

LEADING CHANGE

1. ESTABLISH SENSE OF URGENCY
2. CREATE THE GUIDING COALITION
3. DEVELOP A VISION AND STRATEGY
4. COMMUNICATE THE VISION
5. EMPOWER EMPLOYEES FOR BROAD-BASED ACTION
6. GENERATE SHORT-TERM WINS
7. CONSOLIDATE GAINS AND PRODUCE MORE CHANGE
8. ANCHOR NEW APPROACHES IN THE CULTURE


Copyrighted Material
The International Bestseller, with a New Preface by the Author
LEADING CHANGE
JOHN P. KOTTER
HARVARD BUSINESS REVIEW PRESS
Copyrighted Material

WRAPPING UP - CONCLUSION

- STRATEGY / TACTICS
  - ENSURE SMART GOALS AND ACCOMPANYING STRATEGY ARE DEFINED AND COMMUNICATED
  - TECHNICAL LEADERSHIP (ARCHITECTURE) SKILLS ARE VITAL
  - CHOOSE TOOLING TO SUPPORT YOUR APPROACH (NOT THE OTHER WAY AROUND)
- FEEDBACK
  - OPTIMISE FOR VISIBILITY AND LEARNING (THROUGHOUT THE ORGANISATIONAL STACK)
- RESPONSIBILITIES
  - LEARN FROM CONWAY, NETFLIX AND SPOTIFY ET AL, BUT DO NOT CARGO CULT BLINDLY
  - DEVOPS (DONE RIGHT) IS A PREREQUISITE FOR MICROSERVICES

THANKS...

@DANIELBRYANTUK

DANIEL.BRYANT@SPECTO.IO

(CREDIT TO THE ENTIRE OPENCREDO & SPECTO TEAMS FOR INSPIRATION/GUIDANCE)

BEDTIME READING

O'REILLY
Building Microservices
DESIGNING FINE-GRAINED SYSTEMS
Sam Newman
Microservices
Flexible Software Architectures
Eberhard Wolff
O'REILLY
Designing Delivery
Rethinking IT in the Digital Service Economy
Jeff Sussna
Foreword by James Urquhart
The Addison-Wesley Signature Series
More Agile Testing
Learning Journeys for the Whole Team
Janet Gregory
Lisa Crispin
O'REILLY
Production-Ready Microservices
BUILDING STANDARDIZED SYSTEMS ACROSS AN ENGINEERING ORGANIZATION
Susan J. Fowler
O'REILLY
Microservice Architecture
ALIGNING PRINCIPLES, PRACTICES, AND CULTURE
Mike Amundsen, Matt McLarty, Ronnie Mitra & Irakli Nadareishvili
Designing web APIs that developers will love
Irresistible APIs
Kirsten L. Hunter
MANNING
THE ART OF BUSINESS VALUE
MARK SCHWARTZ
FOREWORD BY GENE KIM
coding the architecture
Software Architecture for Developers
Technical leadership by coding, coaching, collaboration, architecture sketching and just enough up front design
Simon Brown
O'REILLY
Containerizing Continuous Delivery in Java
Docker Integration for Build Pipelines and Application Architecture
Daniel Bryant