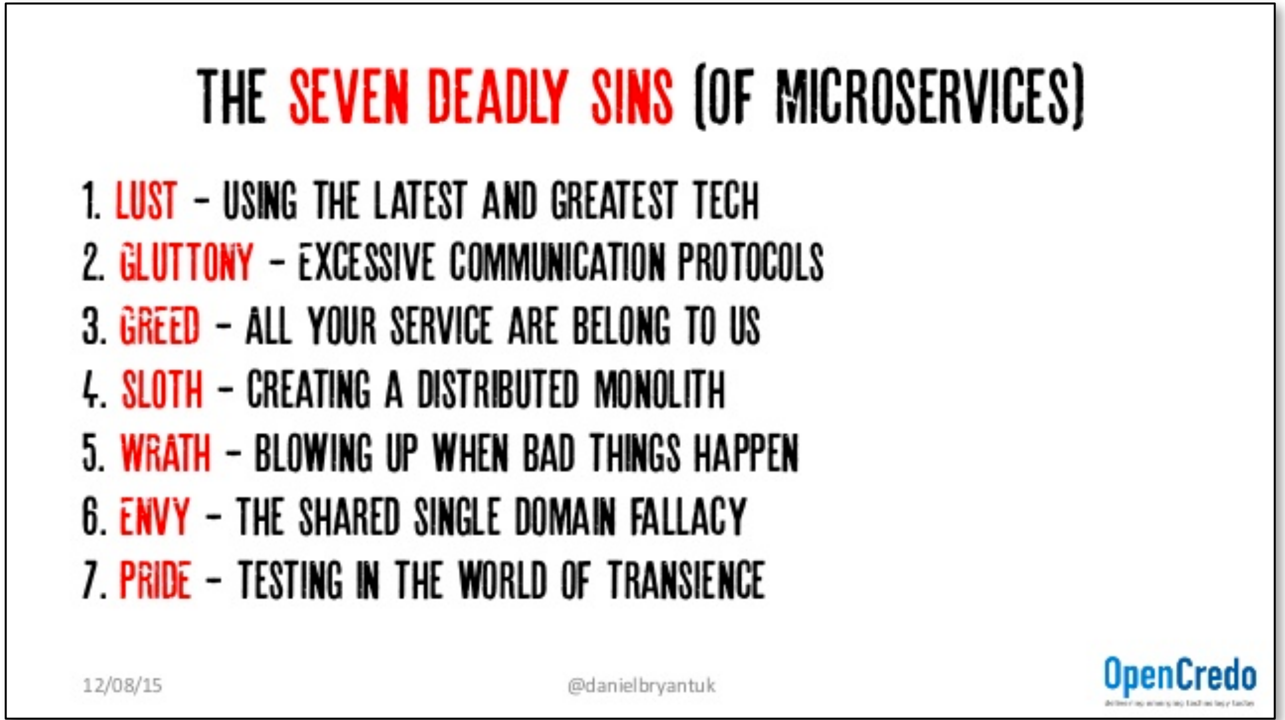
THE SEVEN (MORE) DEADLY SINS OF MICROSERVICES

@DANIELBRYANTUK

@SPECTOLABS

SpectoLabs

PREVIOUSLY, AT DEVOXX UK & QCON NYC 2015…



THE SEVEN DEADLY SINS (OF MICROSERVICES)

1. LUST - USING THE LATEST AND GREATEST TECH
2. GLUTTONY - EXCESSIVE COMMUNICATION PROTOCOLS
3. GREED - ALL YOUR SERVICE ARE BELONG TO US
4. SLOTH - CREATING A DISTRIBUTED MONOLITH
5. WRATH - BLOWING UP WHEN BAD THINGS HAPPEN
6. ENVY - THE SHARED SINGLE DOMAIN FALLACY
7. PRIDE - TESTING IN THE WORLD OF TRANSIENCE

12/08/15 @danielbryantuk OpenCredo

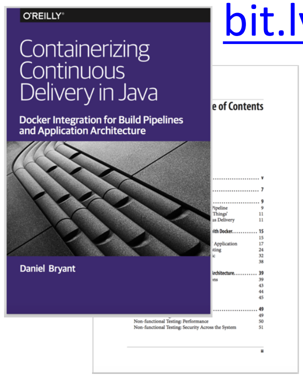
https://www.infoq.com/presentations/7-sins-microservices

THE SEVEN (MORE) DEADLY SINS OF MICROSERVICES

1. LUST - USING THE (UNEVALUATED) LATEST AND GREATEST TECH
2. GLUTTONY - COMMUNICATION LOCK-IN
3. GREED - WHAT'S MINE IS MINE (WITHIN THE ORGANISATION)
4. SLOTH - GETTING LAZY WITH NFRS
5. WRATH - BLOWING UP WHEN BAD THINGS HAPPEN
6. ENVY - THE SHARED SINGLE DOMAIN (AND DATA STORE) FALLACY
7. PRIDE - TESTING IN THE WORLD OF TRANSIENCE

@DANIELBRYANTUK

- SOFTWARE DEVELOPER, CTO AT SPECTOLABS
  - AGILE, 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
e of Contents

bit.ly/2jWDSF7

1. LUST - USING THE LATEST AND GREATEST TECH

NEW TECHNOLOGY IS GREAT… UNTIL IT ISN'T

DEVELOPERS WITH NEW TECH BE LIKE

This has been me many times!

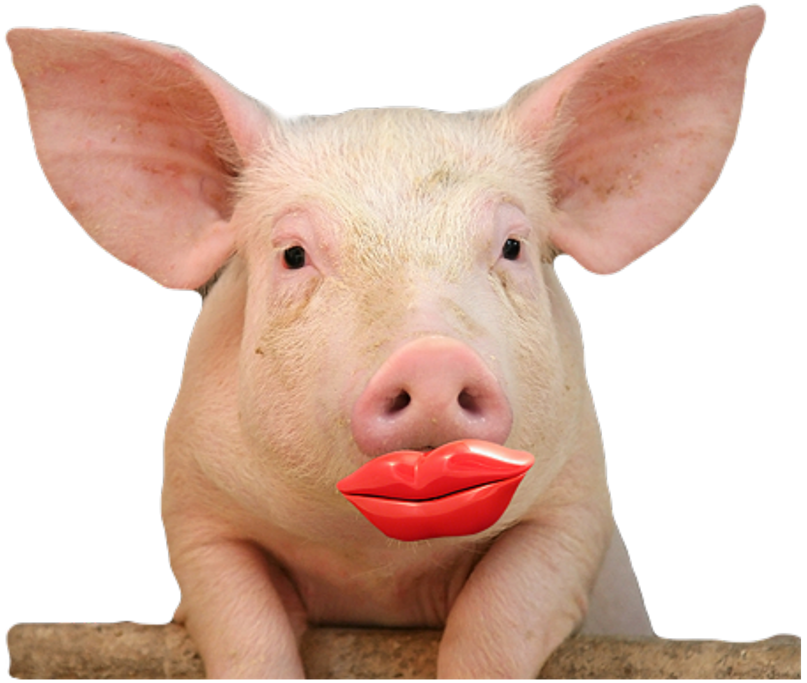
F*CKING NEW TECHNOLOGY…

CREDIT TO MICHAEL HAUSENBLAS

EVALUATION IS A KEY SKILL...

EVALUATION – ARE MICROSERVICES A GOOD FIT?

- "OUR 'MODE TWO' APPS ARE MICROSERVICES"
  - MIDDLE-MANAGEMENT LATCH ON TO BUZZWORD
  - NEW APP EVOLUTION LIMITED BY EXISTING SYSTEM
  - 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/OPS FREE-FOR-ALL



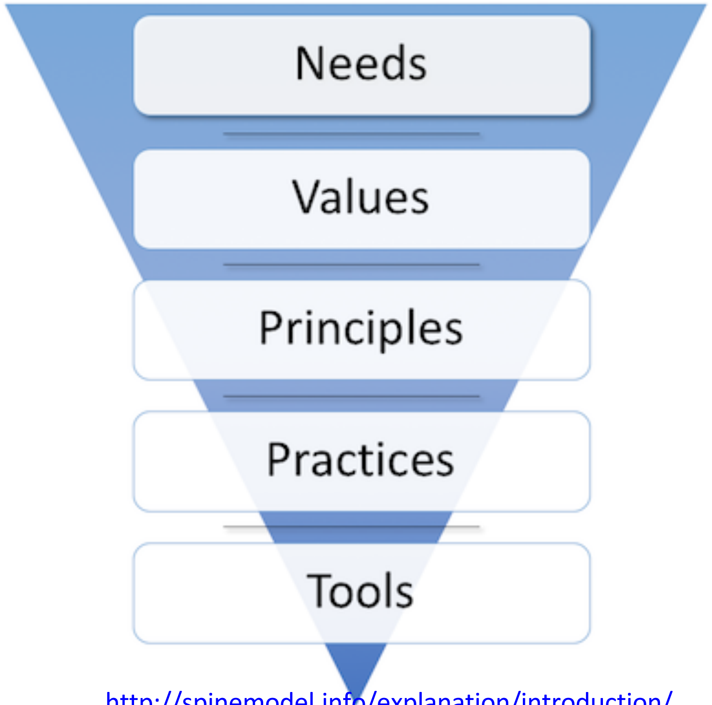
EVALUATION OF TECH – THE SPINE MODEL

- EFFECTIVE CONVERSATIONS MAKE FOR EFFECTIVE COLLABORATION

- 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



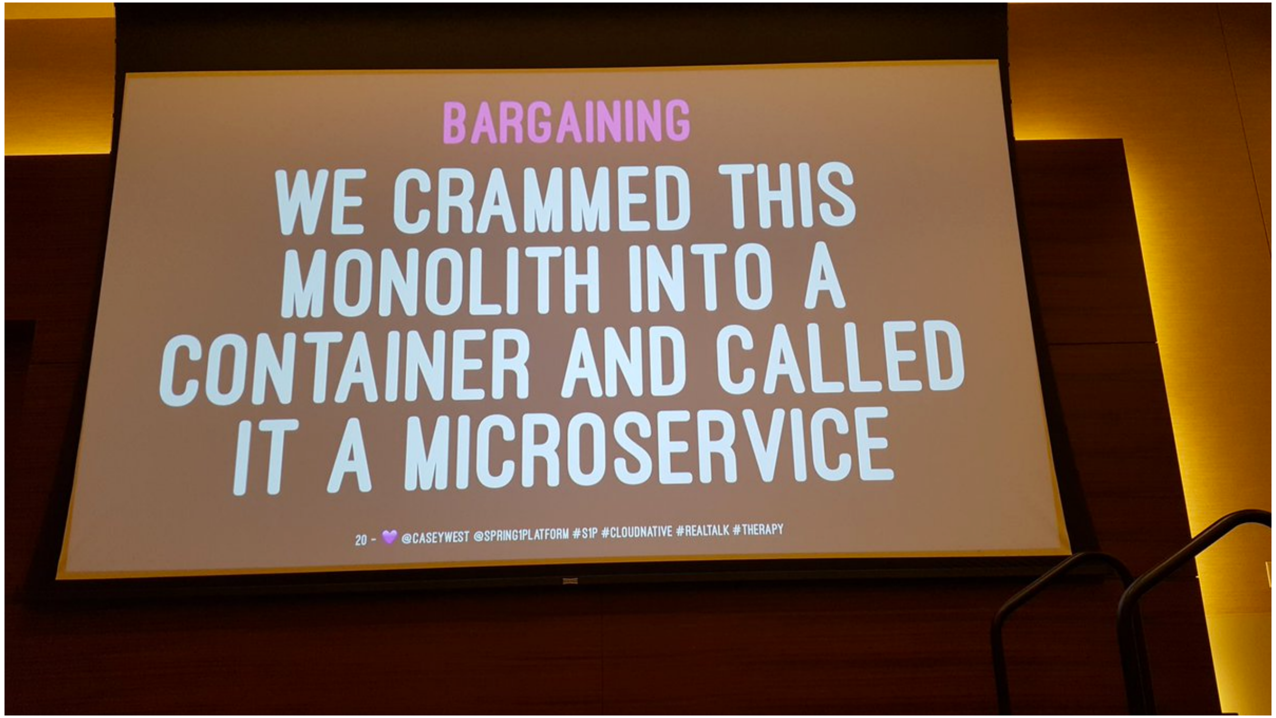
http://spinemodel.info/explanation/introduction/

SpectoLabs

AN EXAMPLE: TO CONTAINERISE, OR NOT TO CONTAINERISE?

(DOCKAH, DOCKAH, DOCKAH... DOCKAH?)

STRATEGY #FAIL



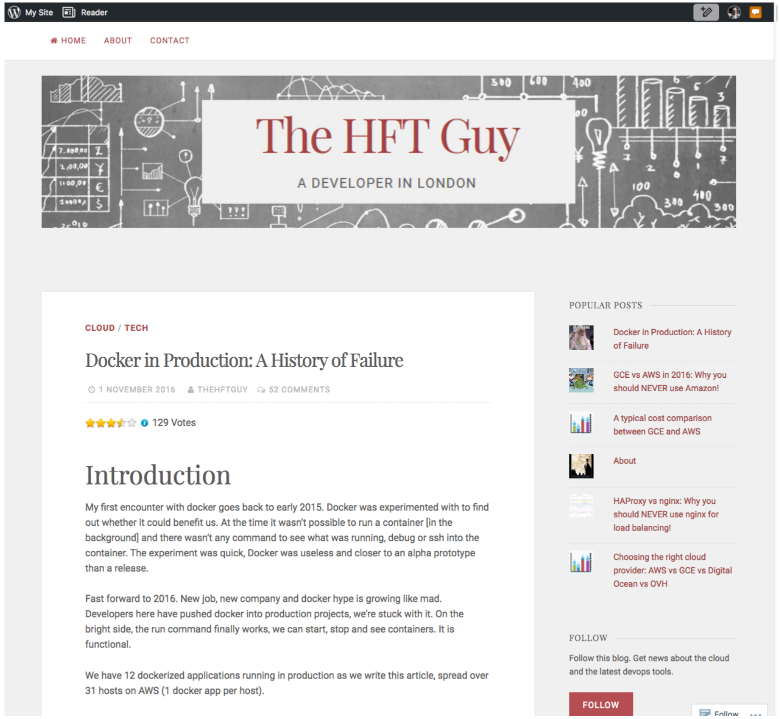
BARGAINING

WE CRAMMED THIS MONOLITH INTO A CONTAINER AND CALLED IT A MICROSERVICE

20 - 💜 @CASEYWEST @SPRING1PLATFORM #S1P #CLOUDNATIVE #REALTALK #THERAPY

ARCHITECTURE/OPS: EXPECTATIONS VERSUS REALITY

CHOICES: BEWARE OF CONFIRMATION BIAS



My Site Reader

HOME ABOUT CONTACT

The HFT Guy

A DEVELOPER IN LONDON

CLOUD / TECH

Docker in Production: A History of Failure

1 NOVEMBER 2016 THEHFTGUY 52 COMMENTS

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

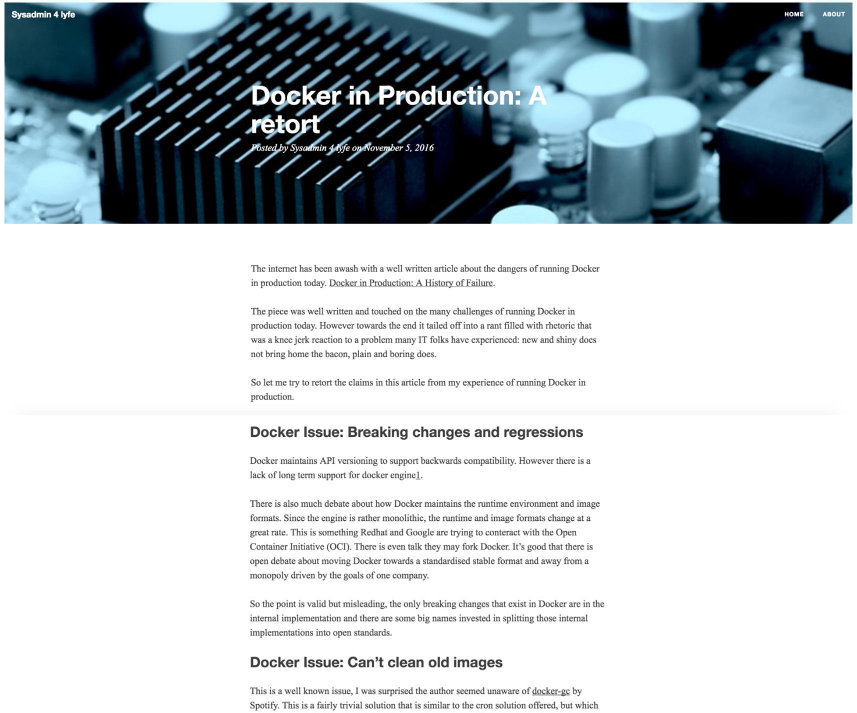
- Docker in Production: A History of Failure
- GCE vs AWS in 2016: Why you should NEVER use Amazon!
- A typical cost comparison between GCE and AWS
- About
- HAProxy vs nginx: Why you should NEVER use nginx for load balancing!
- Choosing the right cloud provider: AWS vs GCE vs Digital Ocean vs OVH

FOLLOW

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

FOLLOW

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



Sysadmin 4 lyfe HOME ABOUT

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.

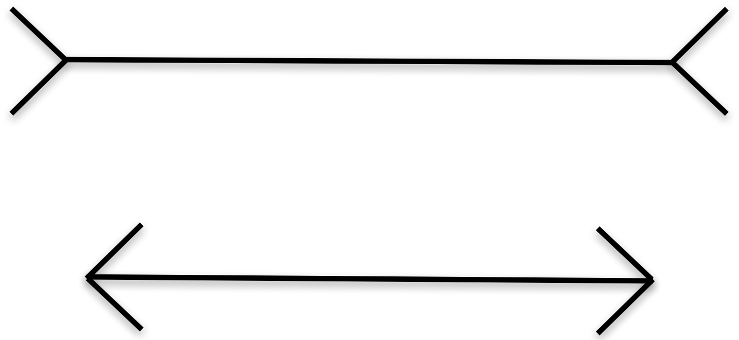
So the point is valid but misleading, the only breaking changes that exist in Docker are in the internal implementation and there are some big names invested in splitting those internal implementations into open standards.

Docker Issue: Can't clean old images

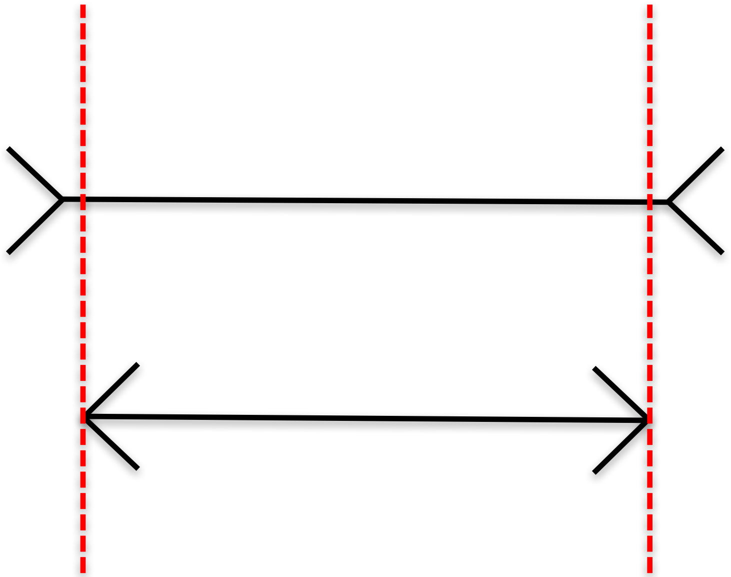
This is a well known issue, I was surprised the author seemed unaware of docker-gc by Spotify. This is a fairly trivial solution that is similar to the cron solution offered, but which

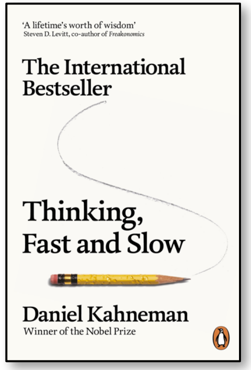
http://patrobinson.github.io/2016/11/05/docker-in-production/

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

2. GLUTTONY - COMMUNICATION LOCK-IN

RPC - NOT THE DEVIL IN DISGUISE

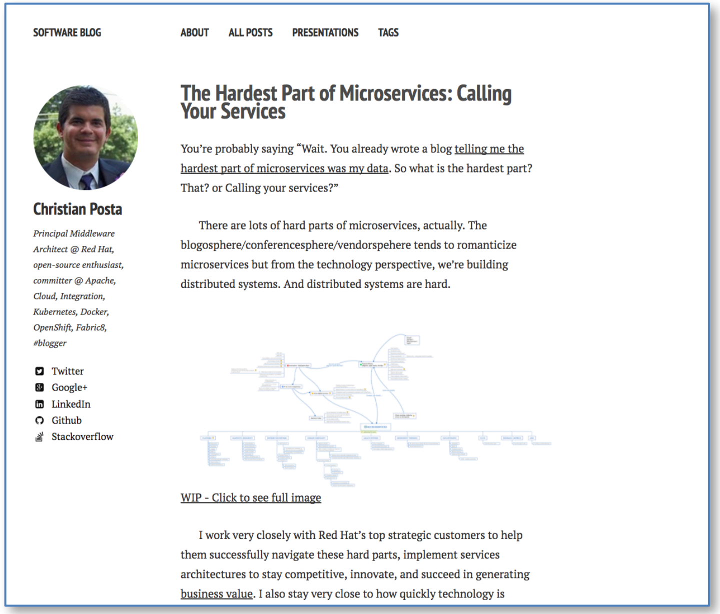
- WE ALL LIKE REST AND JSON, BUT…
- DON'T RULE OUT RPC (E.G. GRPC)
  - THE CONTRACT (AND SPEED) CAN BE BENEFICIAL
  - HUMAN READABILITY OF JSON IS OVER-RATED

DELEGATION OF COMMS OPERABILITY

linkerd

træfik

envoy by lyft



SOFTWARE BLOG ABOUT ALL POSTS PRESENTATIONS TAGS

Christian Posta

Principal Middleware Architect @ Red Hat, open-source enthusiast, committer @ Apache, Cloud, Integration, Kubernetes, Docker, OpenShift, Fabric8, #blogger

- Twitter
- Google+
- LinkedIn
- Github
- Stackoverflow

The Hardest Part of Microservices: Calling Your Services

You're probably saying "Wait. You already wrote a blog telling me the hardest part of microservices was my data. So what is the hardest part? That? or Calling your services?"

There are lots of hard parts of microservices, actually. The blogosphere/conferencesphere/vendorsphere tends to romanticize microservices but from the technology perspective, we're building distributed systems. And distributed systems are hard.

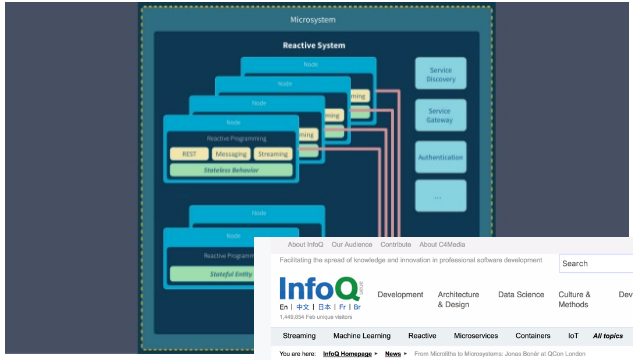
WIP - Click to see full image

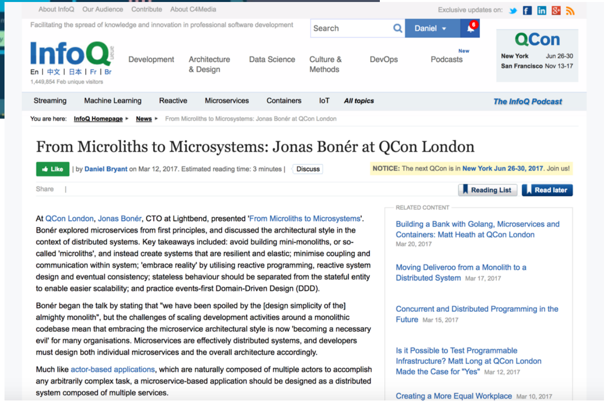
I work very closely with Red Hat's top strategic customers to help them successfully navigate these hard parts, implement services architectures to stay competitive, innovate, and succeed in generating business value. I also stay very close to how quickly technology is

blog.christianposta.com/microservices/the-hardest-part-of-microservices-calling-your-services/

RPC – NOT THE DEVIL IN DISGUISE

- SOMETIME EVENTS ARE BETTER
  - ASYNCHRONOUS (AP VS CP)
  - EVENT-SOURCING, CQRS ETC
- REACTIVE IS EVERYWHERE
  - AND ONLY GETTING HOTTER…




About InfoQ Our Audience Contribute About C4Media

Exclusive updates on:

Facilitating the spread of knowledge and innovation in professional software development

Search | Daniel

InfoQ | En | 中文 | 日本 | Fr | Br | 1,449,854 Feb unique visitors

Development | Architecture & Design | Data Science | Culture & Methods | DevOps | New Podcasts

QCon New York Jun 26-30 San Francisco Nov 13-17

Streaming | Machine Learning | Reactive | Microservices | Containers | IoT | All topics | The InfoQ Podcast

You are here: InfoQ Homepage ▸ News ▸ From Microliths to Microsystems: Jonas Bonér at QCon London

From Microliths to Microsystems: Jonas Bonér at QCon London

Like | by Daniel Bryant on Mar 12, 2017. Estimated reading time: 3 minutes | Discuss

NOTICE: The next QCon is in New York Jun 26-30, 2017. Join us!

Share | Reading List | Read later

At QCon London, Jonas Bonér, CTO at Lightbend, presented 'From Microliths to Microsystems'. Bonér explored microservices from first principles, and discussed the architectural style in the context of distributed systems. Key takeaways included: avoid building mini-monoliths, or so-called 'microliths', and instead create systems that are resilient and elastic; minimise coupling and communication within system; 'embrace reality' by utilising reactive programming, reactive system design and eventual consistency; stateless behaviour should be separated from the stateful entity to enable easier scalability; and practice events-first Domain-Driven Design (DDD).

Bonér began the talk by stating that "we have been spoiled by the [design simplicity of the] almighty monolith", but the challenges of scaling development activities around a monolithic codebase mean that embracing the microservice architectural style is now 'becoming a necessary evil' for many organisations. Microservices are effectively distributed systems, and developers must design both individual microservices and the overall architecture accordingly.

Much like actor-based applications, which are naturally composed of multiple actors to accomplish any arbitrarily complex task, a microservice-based application should be designed as a distributed system composed of multiple services.

RELATED CONTENT

Building a Bank with Golang, Microservices and Containers: Matt Heath at QCon London Mar 20, 2017

Moving Deliveroo from a Monolith to a Distributed System Mar 17, 2017

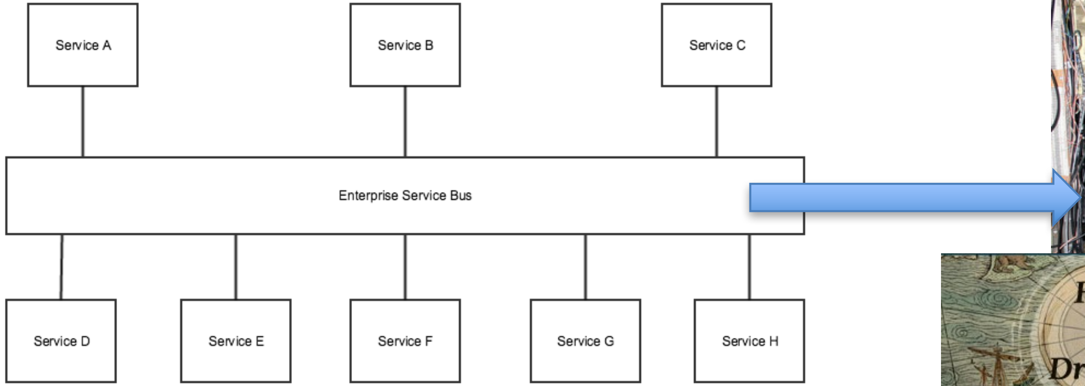
Concurrent and Distributed Programming in the Future Mar 15, 2017

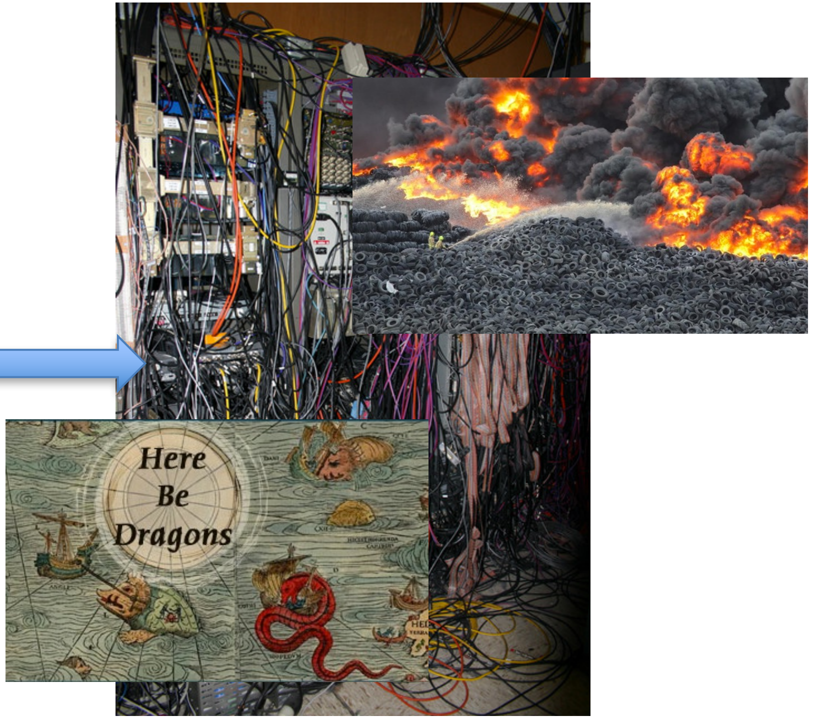
Is it Possible to Test Programmable Infrastructure? Matt Long at QCon London Made the Case for "Yes" Mar 12, 2017

Creating a More Equal Workplace Mar 10, 2017

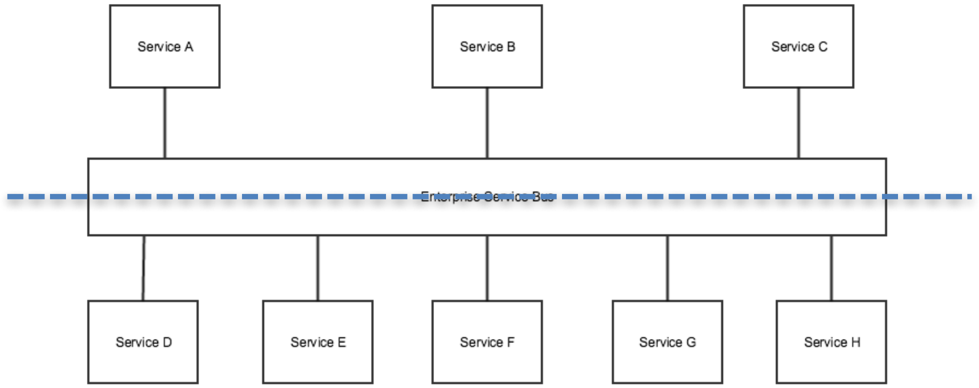
www.infoq.com/news/2017/03/microliths-microsystems

THE ESB IS DEAD – LONG LIVE THE ESB!

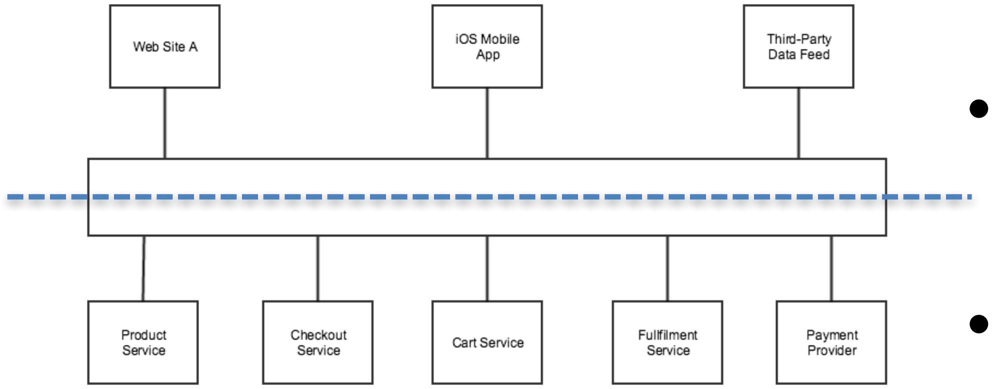




THE ESB IS DEAD - LONG LIVE THE ESB!

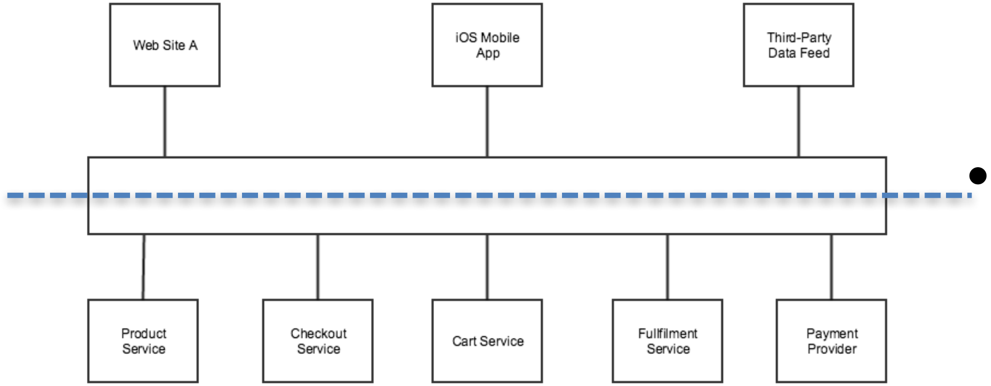


THE ESB IS DEAD – LONG LIVE THE ESB!



- IS THIS AN ESB?
- OR AN API GATEWAY?

THE ESB IS DEAD – LONG LIVE THE API GATEWAY!



- WATCH FOR THE API GATEWAY MORPHING INTO AN ENTERPRISE SERVICE BUS
  - LOOSE COUPLING IS VITAL
- BUT LET ME BE CLEAR…
  - THE API GATEWAY PATTERN IS AWESOME
  - CENTRALISE CROSS-CUTTING CONCERNS
  - PREVENT WHEEL-REINVENTION (PLUGINS)
  - CHECK OUT KONG, APIGEE, MULESOFT ETC

3. GREED - WHAT'S MINE IS MINE... (WITHIN THE ORGANISATION)

PREVIOUSLY...

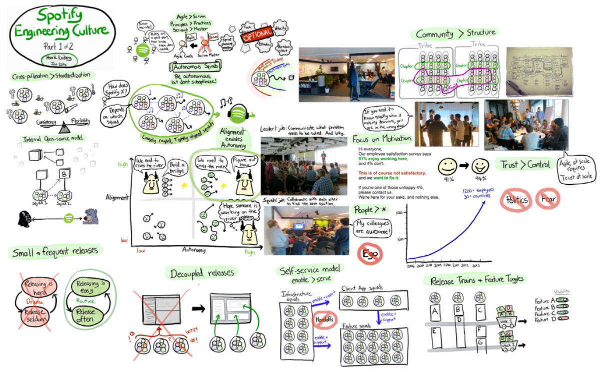
- CONWAY'S LAW

- MICROSERVICES ARE ABOUT PEOPLE, AS MUCH AS THEY ARE TECH
  - MAYBE MORE
  - PARTICULARLY IN A MIGRATION / TRANSFORMATION

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

4. SLOTH - GETTING LAZY WITH NFRS

GETTING LAZY WITH NON-FUNCTIONAL REQUIREMENTS

"THE DRIVING TECHNICAL REQUIREMENTS FOR A SYSTEM SHOULD BE IDENTIFIED EARLY TO ENSURE THEY ARE PROPERLY HANDLED IN SUBSEQUENT DESIGN"

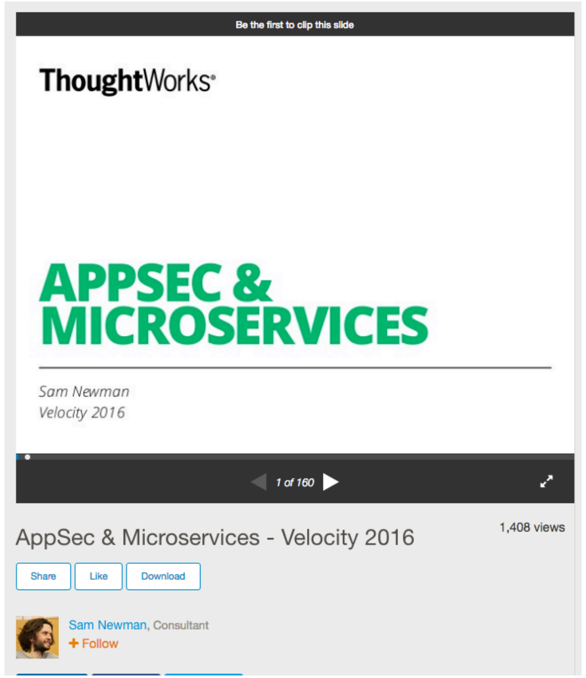
AIDAN CASEY

GUIDING PRINCIPLES FOR EVOLUTIONARY ARCHITECTURE

GETTING LAZY WITH NON-FUNCTIONAL REQUIREMENTS

- THE 'ILITIES' CAN BE (OFTEN) BE AN AFTERTHOUGHT
  - AVAILABILITY, SCALABILITY, AUDITABILITY, TESTABILITY ETC
- AGILE/LEAN: DELAY DECISIONS TO THE 'LAST RESPONSIBLE MOMENT'
  - NEWSFLASH - SOMETIMES THIS IS UP-FRONT
- IT CAN BE COSTLY (OR PROHIBITIVE) TO ADAPT LATE IN THE PROJECT
  - MICROSERVICES DON'T MAKE THIS EASIER (SOMETIMES MORE DIFFICULT)

GETTING LAZY WITH NFRS - SECURITY


Be the first to clip this slide

ThoughtWorks®

APPSEC & MICROSERVICES

Sam Newman
Velocity 2016

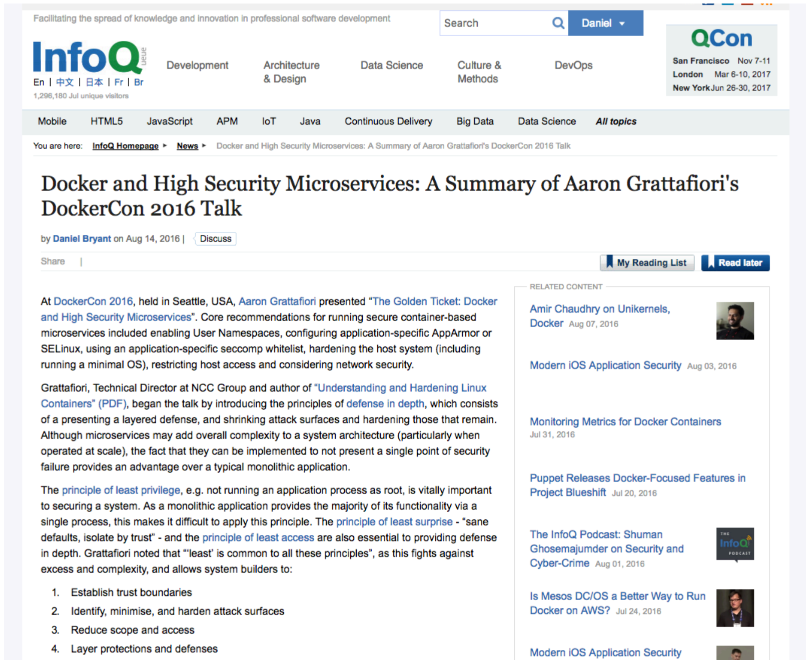
1 of 160

AppSec & Microservices - Velocity 2016

1,408 views

Share Like Download

Sam Newman, Consultant
+ Follow


Facilitating the spread of knowledge and innovation in professional software development

Search Daniel

InfoQ
En | 中文 | 日本 | Fr | Br
1,296,180 Jul unique visitors

Development Architecture & Design Data Science Culture & Methods DevOps

QCon
San Francisco Nov 7-11
London Mar 6-10, 2017
New York Jun 26-30, 2017

Mobile HTML5 JavaScript APM IoT Java Continuous Delivery Big Data Data Science *All topics*

You are here: InfoQ Homepage ▸ News ▸ Docker and High Security Microservices: A Summary of Aaron Grattafiori's DockerCon 2016 Talk

Docker and High Security Microservices: A Summary of Aaron Grattafiori's DockerCon 2016 Talk

by Daniel Bryant on Aug 14, 2016 | Discuss

Share | My Reading List Read later

At DockerCon 2016, held in Seattle, USA, Aaron Grattafiori presented "The Golden Ticket: Docker and High Security Microservices". Core recommendations for running secure container-based microservices included enabling User Namespaces, configuring application-specific AppArmor or SELinux, using an application-specific seccomp whitelist, hardening the host system (including running a minimal OS), restricting host access and considering network security.

Grattafiori, Technical Director at NCC Group and author of "Understanding and Hardening Linux Containers" (PDF), began the talk by introducing the principles of defense in depth, which consists of a presenting a layered defense, and shrinking attack surfaces and hardening those that remain. Although microservices may add overall complexity to a system architecture (particularly when operated at scale), the fact that they can be implemented to not present a single point of security failure provides an advantage over a typical monolithic application.

The principle of least privilege, e.g. not running an application process as root, is vitally important to securing a system. As a monolithic application provides the majority of its functionality via a single process, this makes it difficult to apply this principle. The principle of least surprise - "sane defaults, isolate by trust" - and the principle of least access are also essential to providing defense in depth. Grattafiori noted that "'least' is common to all these principles", as this fights against excess and complexity, and allows system builders to:

1. Establish trust boundaries
2. Identify, minimise, and harden attack surfaces
3. Reduce scope and access
4. Layer protections and defenses

RELATED CONTENT

Amir Chaudhry on Unikernels, Docker Aug 07, 2016

Modern iOS Application Security Aug 03, 2016

Monitoring Metrics for Docker Containers Jul 31, 2016

Puppet Releases Docker-Focused Features in Project Blueshift Jul 20, 2016

The InfoQ Podcast: Shuman Ghosemajumder on Security and Cyber-Crime Aug 01, 2016

Is Mesos DC/OS a Better Way to Run Docker on AWS? Jul 24, 2016

Modern iOS Application Security

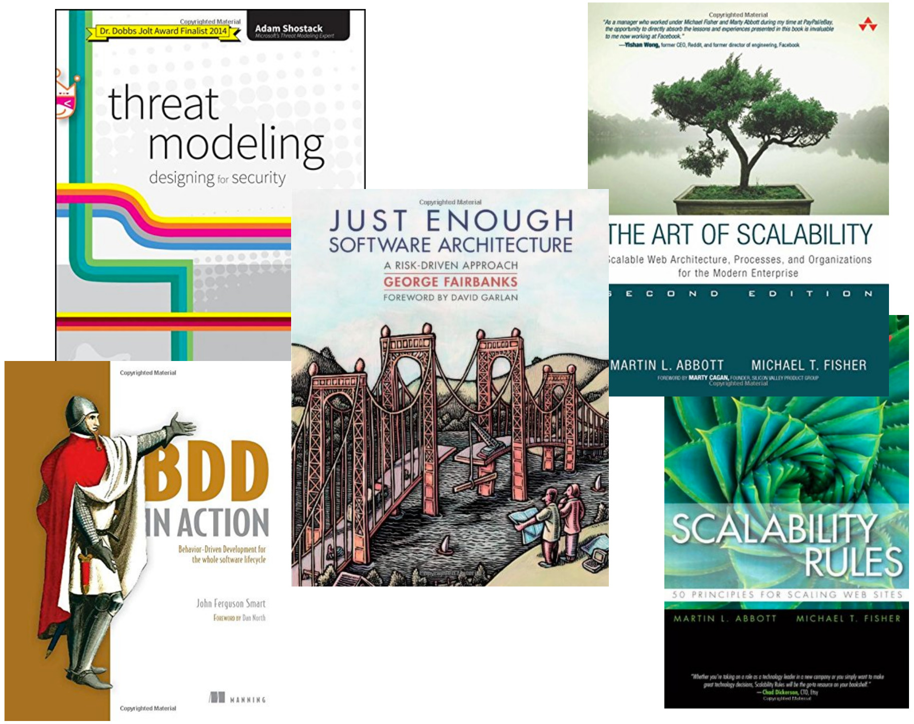
www.slideshare.net/spnewman/appsec-microservices-velocity-2016

www.infoq.com/news/2016/08/secure-docker-microservices

TESTING NFRS IN THE BUILD PIPELINE

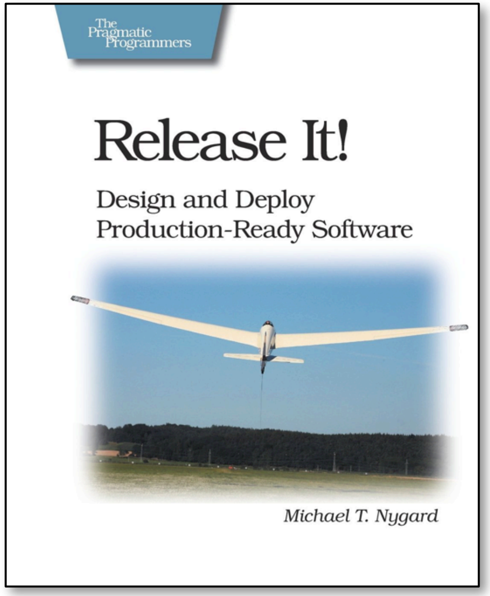
- PERFORMANCE AND LOAD TESTING
  - GATLING / JMETER
  - FLOOD.IO

- SECURITY TESTING
  - FINDSECBUGS / OWASP DEPENDENCY CHECK
  - BDD-SECURITY (OWASP ZAP) / ARACHNI
  - GAUNTLT / SERVERSPEC
  - DOCKER BENCH FOR SECURITY / CLAIR



5. WRATH - BLOWING UP WHEN BAD THINGS HAPPEN

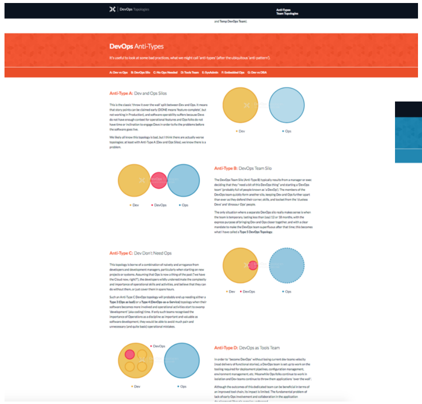
PREVIOUSLY – BRING IN MICHAEL NYGARD (OR SOME MONKEYS)

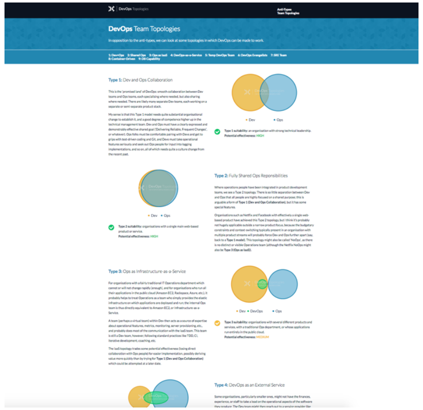

The Pragmatic Programmers
Release It!
Design and Deploy Production-Ready Software
Michael T. Nygard

WHEN BAD THINGS HAPPEN, PEOPLE ARE ALWAYS INVOLVED

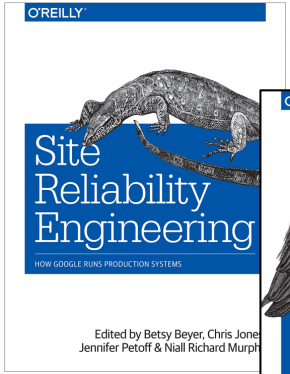
PEOPLE PAIN POINT – HOW DOES DEVOPS FIT INTO THIS?

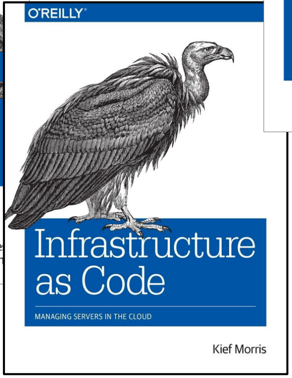
- HTTP://WEB.DEVOPSTOPOLOGIES.COM/
- @ MATTHEWPSKELTON

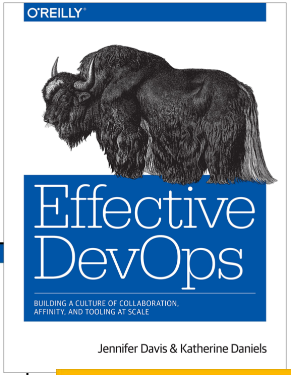

DevOps Anti-Types
Anti-Type A: Dev and Ops Silos
Anti-Type B: DevOps Team Silo
Anti-Type C: Dev Don't Need Ops
Anti-Type D: DevOps as Tools Team

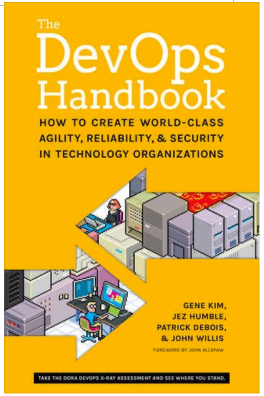

DevOps Team Topologies
Type 1: Dev and Ops Collaboration
Type 2: Fully Shared Ops Responsibilities
Type 3: Ops as Infrastructure-as-a-Service
Type 4: DevOps as an External Service

- BOOKS


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


The DevOps Handbook
HOW TO CREATE WORLD-CLASS AGILITY, RELIABILITY, & SECURITY IN TECHNOLOGY ORGANIZATIONS
GENE KIM, JEZ HUMBLE, PATRICK DEBOIS, & JOHN WILLIS

DEVOPS - THE 'FULLSTACK ENGINEER' MYTH

"I'M SORRY, BUT IF YOU'RE NOT DESIGNING THE COMPUTER CHIPS AND WRITING THE WEBSITE, THEN I DON'T WANNA HEAR FROM YOU"

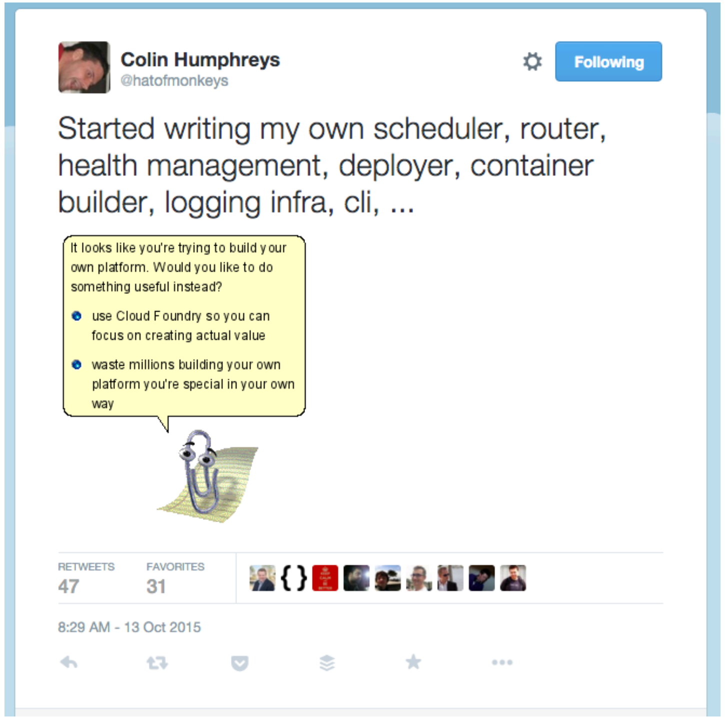
CHARITY MAJORS (@MIPSYTIPSY), CRAFTCONF 2016

HTTP://WWW.USTREAM.TV/RECORDED/86181845

DEVOPS - DEFINE RESPONSIBILITIES

- DO YOU REALLY WANT TO BUILD AN ENTIRE MICROSERVICES PLATFORM?

- FOCUS ON WHAT MATTERS
  - CI/CD
  - MECHANICAL SYMPATHY
  - LOGGING
  - MONITORING



Colin Humphreys
@hatofmonkeys

Started writing my own scheduler, router, health management, deployer, container builder, logging infra, cli, ...

It looks like you're trying to build your own platform. Would you like to do something useful instead?

- use Cloud Foundry so you can focus on creating actual value
- waste millions building your own platform you're special in your own way

RETWEETS 47 FAVORITES 31

8:29 AM - 13 Oct 2015

WORTH CONSIDERING: OPEN SOURCE PAAS/FAAS/DBAAS

OPENSHIFT®
by Red Hat®

DEIS

CLOUDFOUNDRY

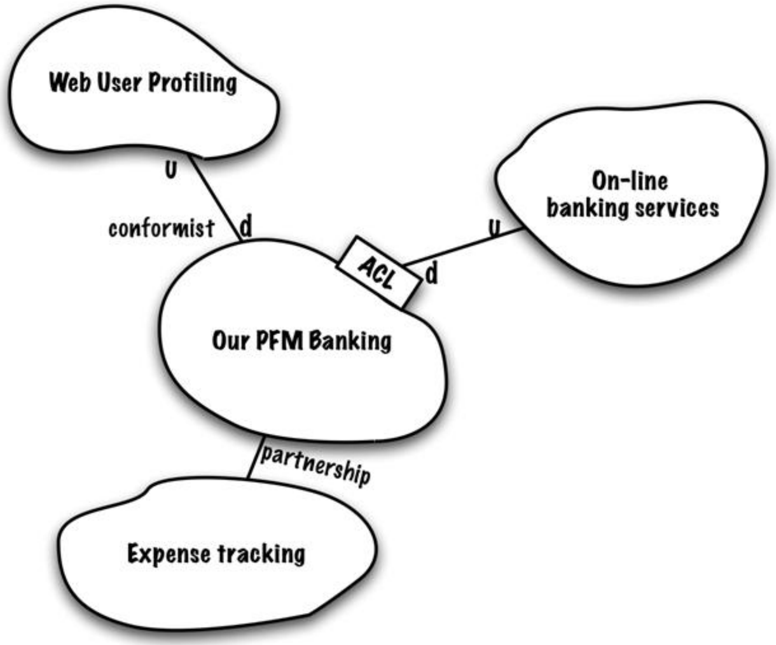
6. ENVY – THE SHARED SINGLE DOMAIN (AND DATA STORE) FALLACY

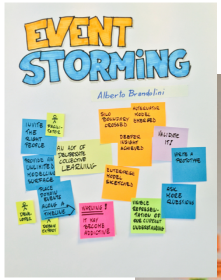
PREVIOUSLY – ONE MODEL TO RULE THEM ALL…

- ONE MODEL
  - BREAKS ENCAPSULATION
  - INTRODUCES COUPLING

- KNOW YOUR DDD
  - ENTITIES
  - VALUE OBJECTS
  - AGGREGATES AND ROOTS

CONTEXT MAPPING (STATIC) & EVENT STORMING (DYNAMIC)



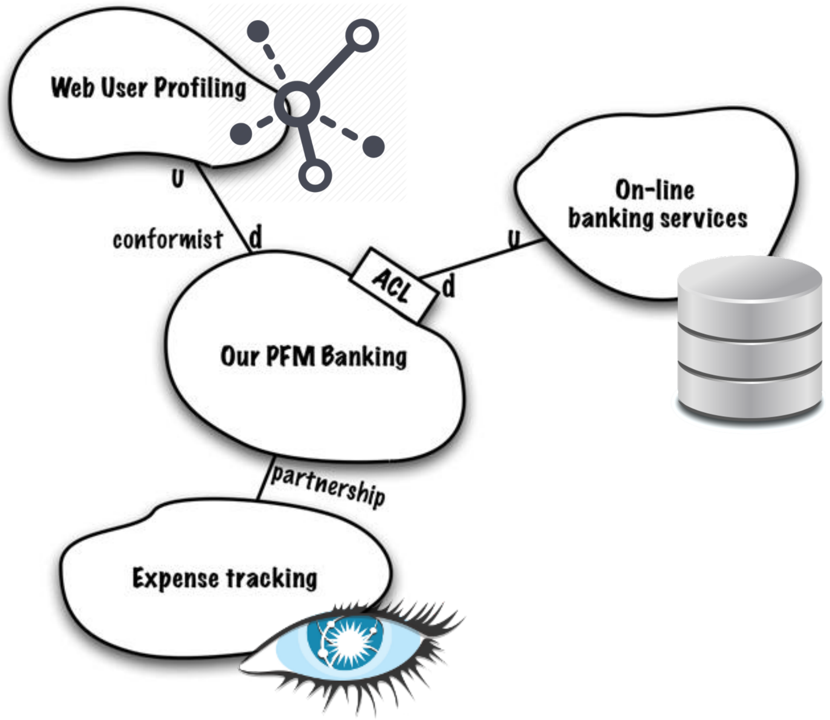


ziobrando.blogspot.co.uk/2013/11/introducing-event-storming.html

www.infoq.com/articles/ddd-contextmapping

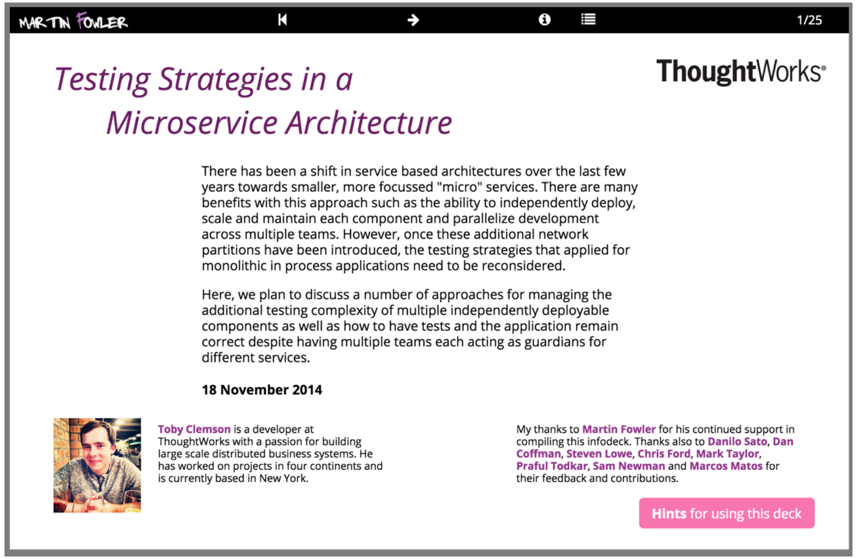
CHOOSE (AND USE) DATA STORES APPROPRIATELY

- RDBMS
  - VALUABLE FOR STRUCTURED DATA
- CASSANDRA IS AWESOME
  - BUT DON'T TREAT IT LIKE AN RDBMS!
- DON'T BUILD A GRAPH WITH RDBMS
  - USE NEO4J, TITAN ETC
- BEWARE OF OPERATIONAL OVERHEAD



7. PRIDE - TESTING IN THE WORLD OF TRANSIENCE

PREVIOUSLY...


MARTIN FOWLER 1/25

Testing Strategies in a Microservice Architecture

ThoughtWorks

There has been a shift in service based architectures over the last few years towards smaller, more focussed "micro" services. There are many benefits with this approach such as the ability to independently deploy, scale and maintain each component and parallelize development across multiple teams. However, once these additional network partitions have been introduced, the testing strategies that applied for monolithic in process applications need to be reconsidered.

Here, we plan to discuss a number of approaches for managing the additional testing complexity of multiple independently deployable components as well as how to have tests and the application remain correct despite having multiple teams each acting as guardians for different services.

18 November 2014

Toby Clemson is a developer at ThoughtWorks with a passion for building large scale distributed business systems. He has worked on projects in four continents and is currently based in New York.

My thanks to Martin Fowler for his continued support in compiling this infodeck. Thanks also to Danilo Sato, Dan Coffman, Steven Lowe, Chris Ford, Mark Taylor, Praful Todkar, Sam Newman and Marcos Matos for their feedback and contributions.

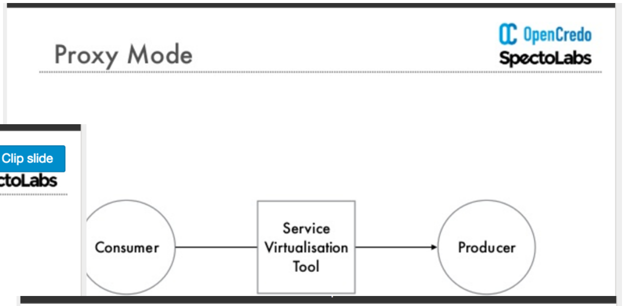
Hints for using this deck

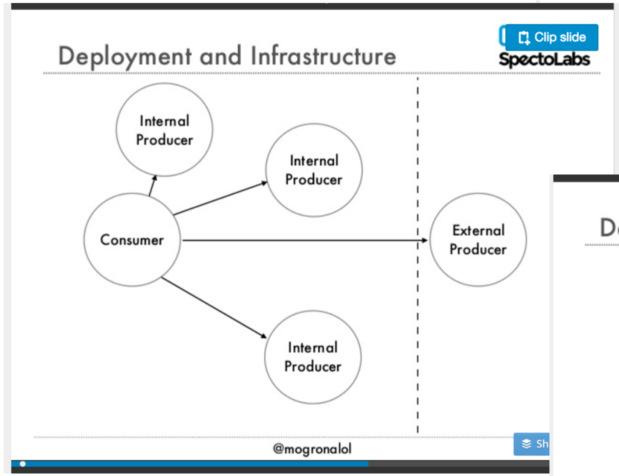
martinfowler.com/articles/microservice-testing/

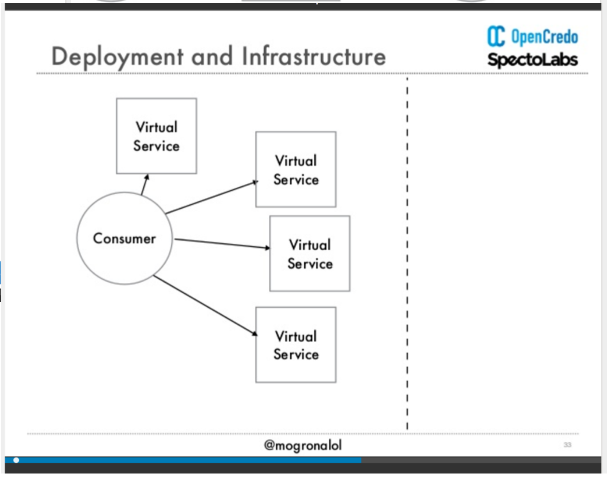
- LOCAL VERIFICATION
  - CONSUMER-DRIVEN CONTRACTS
- END-TO-END
  - BDD-STYLE CRITICAL PATH
- REMEMBER THE TEST PYRAMID

SERVICE VIRTUALISATION / API SIMULATION

- VIRTUALISE REQUEST/RESPONSE OF SERVICES
  - UNAVAILABLE
  - EXPENSIVE TO RUN
  - FRAGILE/BRITTLE
  - NON-DETERMINISTIC
  - CANNOT SIMULATE FAILURES







HTTPS://DZONE.COM/ARTICLES/CONTINUOUSLY-DELIVERING-SOA

ANDREW MORGAN'S TALK HTTP://BIT.LY/2OV0ECD

SERVICE VIRTUALISATION

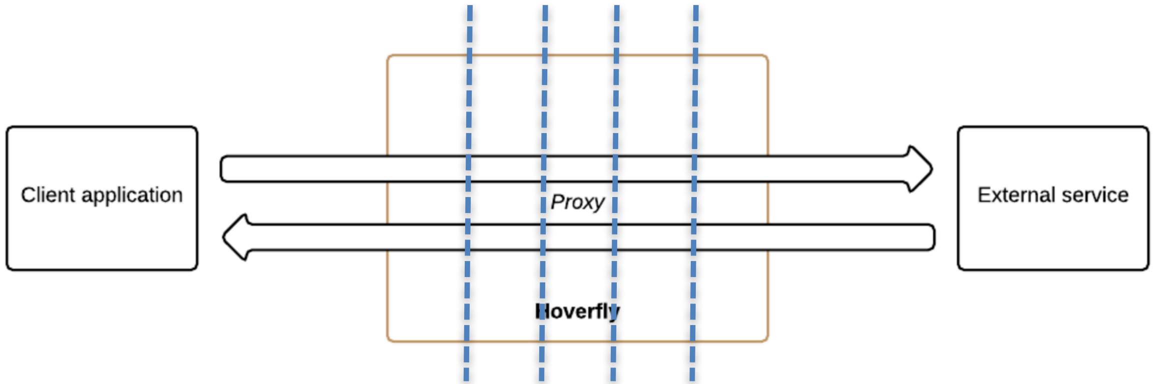
- CLASSICS
  - CA SERVICE VIRTUALIZATION
  - PARASOFT VIRTUALIZE
  - HPE SERVICE VIRTUALIZATION
  - IBM TEST VIRTUALIZATION SERVER
- NEW (OPEN SOURCE) KIDS ON THE BLOCK
  - HOVERFLY
  - WIREMOCK
  - VCR/BETAMAX
  - MOUNTEBANK
  - MIRAGE

HOVERFLY

Hoverfly

- LIGHTWEIGHT SERVICE VIRTUALISATION
  - OPEN SOURCE (APACHE 2.0)
  - GO-BASED / SINGLE BINARY
  - WRITTEN BY @SPECTOLABS
- FLEXIBLE API SIMULATION
  - HTTP / HTTPS
  - HIGHLY PERFORMANT

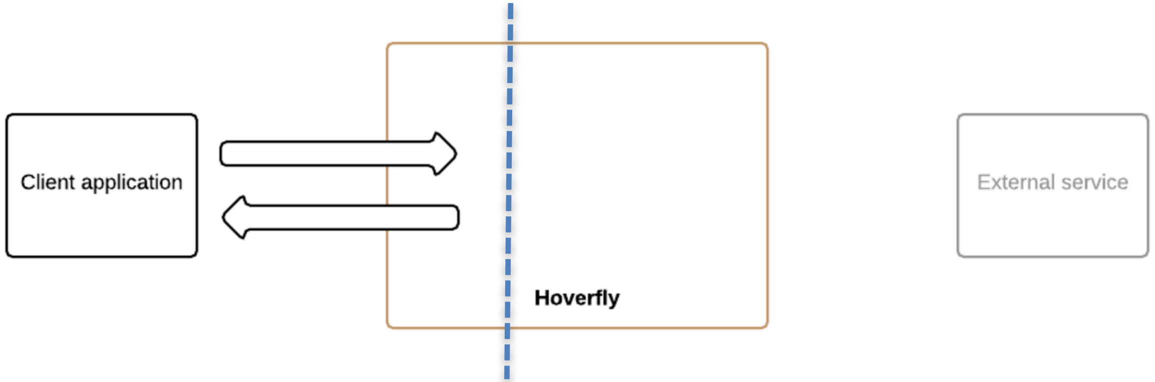
Capture mode



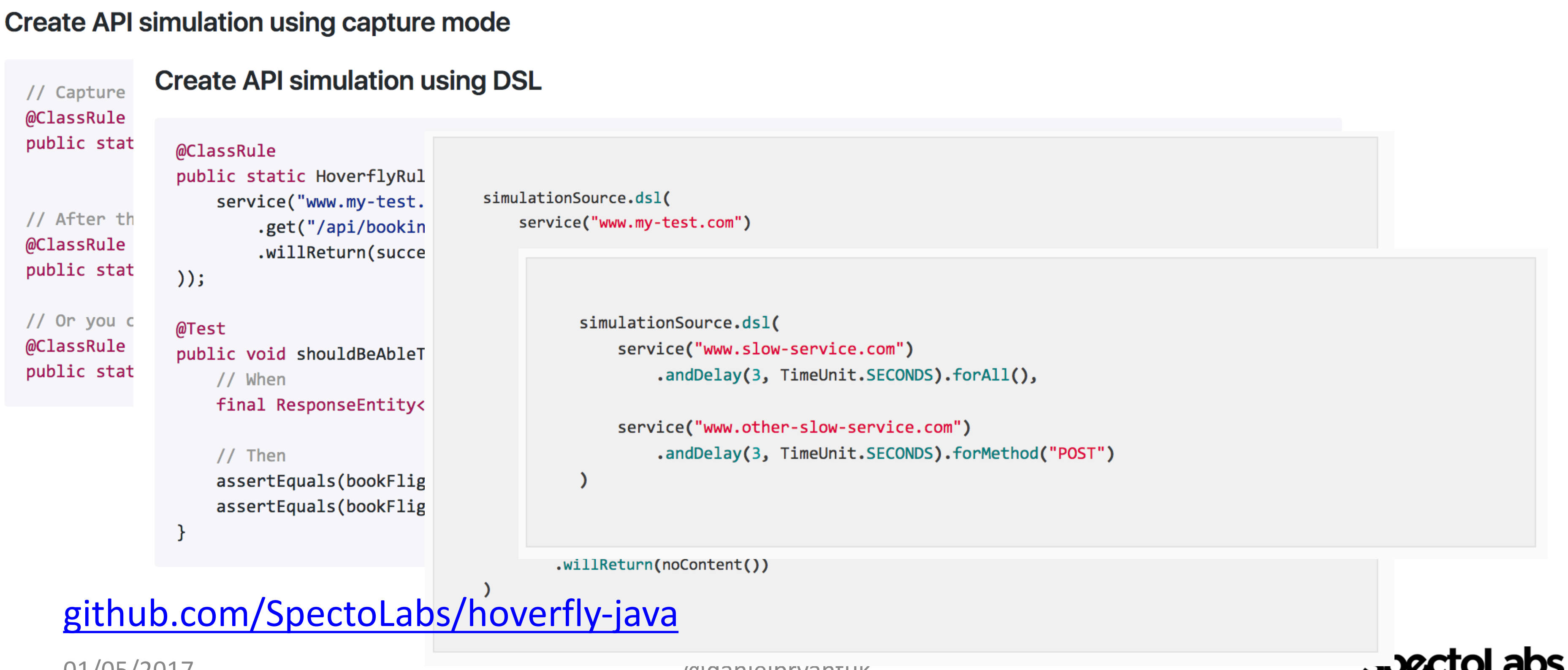
- Middleware
  - Remove PII
  - Rate limit
  - Add headers

Simulate mode

- Middleware
  - Fault injection
  - Chaos monkey



HOVERFLY-JAVA (JUNIT SUPPORT)



Create API simulation using capture mode

```
// Capture
@ClassRule
public stat


// After th
@ClassRule
public stat

// Or you c
@ClassRule
public stat
```

Create API simulation using DSL

```
@ClassRule
public static HoverflyRul
    service("www.my-test.
        .get("/api/bookin
        .willReturn(succe
));

@Test
public void shouldBeAbleT
    // When
    final ResponseEntity<

    // Then
    assertEquals(bookFlig
    assertEquals(bookFlig
}
```

```
simulationSource.dsl(
    service("www.my-test.com")
```

```
simulationSource.dsl(
    service("www.slow-service.com")
        .andDelay(3, TimeUnit.SECONDS).forAll(),

    service("www.other-slow-service.com")
        .andDelay(3, TimeUnit.SECONDS).forMethod("POST")
)
```

```
        .willReturn(noContent())
)
```

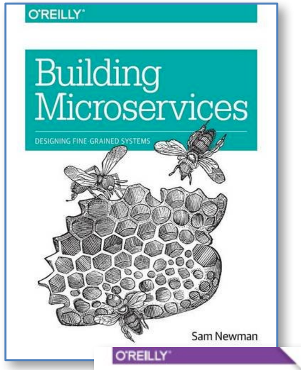
github.com/SpectoLabs/hoverfly-java

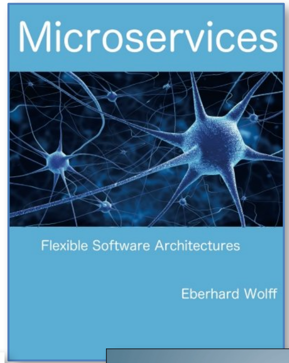
RIGHT, LET'S WRAP THIS UP...

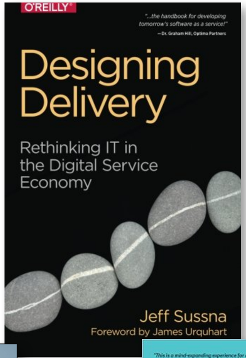
THE SEVEN (MORE) DEADLY SINS OF MICROSERVICES

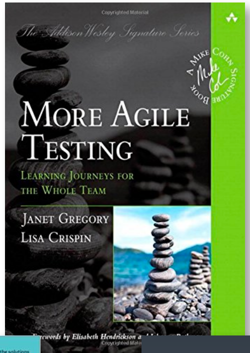
1. LUST - USING THE (UNEVALUATED) LATEST AND GREATEST TECH
2. GLUTTONY - COMMUNICATION LOCK-IN
3. GREED - WHAT'S MINE IS MINE (WITHIN THE ORGANISATION)
4. SLOTH - GETTING LAZY WITH NFRS
5. WRATH - BLOWING UP WHEN BAD THINGS HAPPEN
6. ENVY - THE SHARED SINGLE DOMAIN (AND DATA STORE) FALLACY
7. PRIDE - TESTING IN THE WORLD OF TRANSIENCE

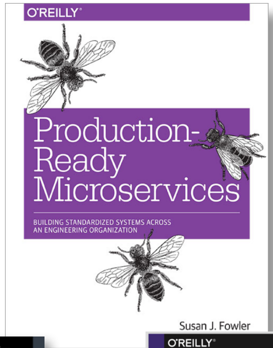
BEDTIME READING

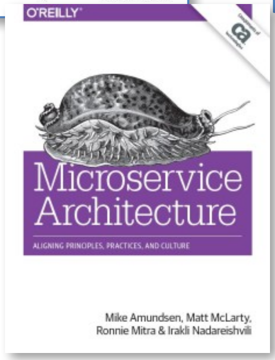


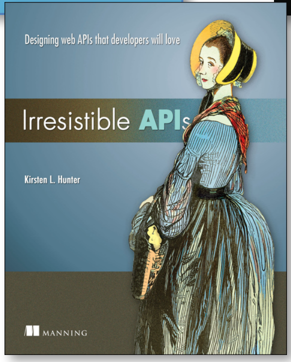


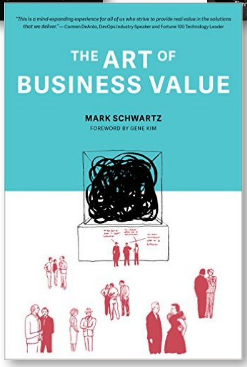

O'REILLY
Designing Delivery
Rethinking IT in the Digital Service Economy
Jeff Sussna
Foreword by James Urquhart


The Addison-Wesley Signature Series
More Agile Testing
Learning Journeys for the Whole Team
Janet Gregory
Lisa Crispin

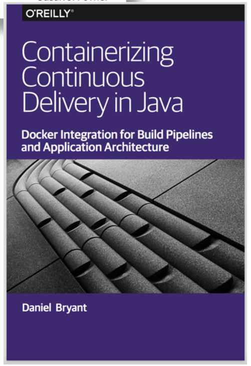

O'REILLY
Production-Ready Microservices
BUILDING STANDARDIZED SYSTEMS ACROSS AN ENGINEERING ORGANIZATION
Susan J. Fowler


O'REILLY
Microservice Architecture
ALIGNING PRINCIPLES, PRACTICES, AND CULTURE
Mike Amundsen, Matt McLarty, Ronnie Mitra & Irakli Nadareishvili





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

THANKS... (DON'T FORGET TO RATE THE TALK!)

HTTP://SPECTO.IO

MUSERVICESWEEKLY.COM

(CREDIT TO TAREQ ABEDRABBO, OPENCREDO FOR INSPIRATION/GUIDANCE)