

THE SEVEN (MORE) DEADLY SINS OF MICROSERVICES

@DANIELBRYANTUK

@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
don't just say it, make it

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

01/05/2017

@danielbryantuk

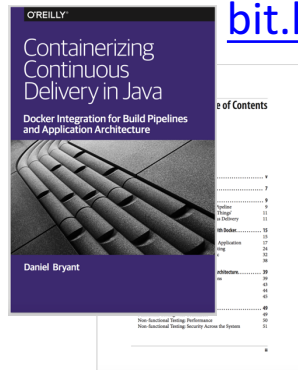
SpectoLabs

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 NFERS
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**



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



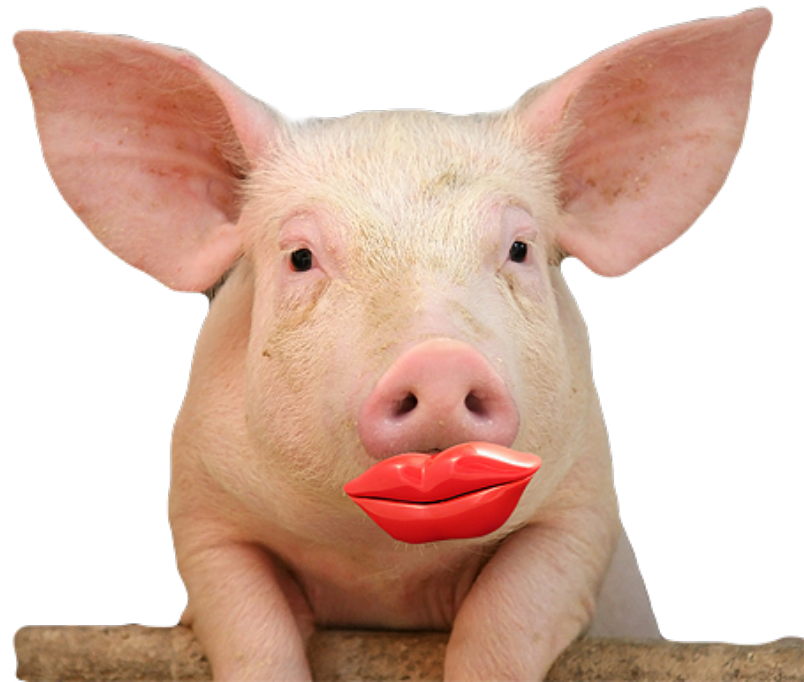
This has been me
many times!

F*CKING NEW TECHNOLOGY...

EVALUATION IS A KEY SKILL...

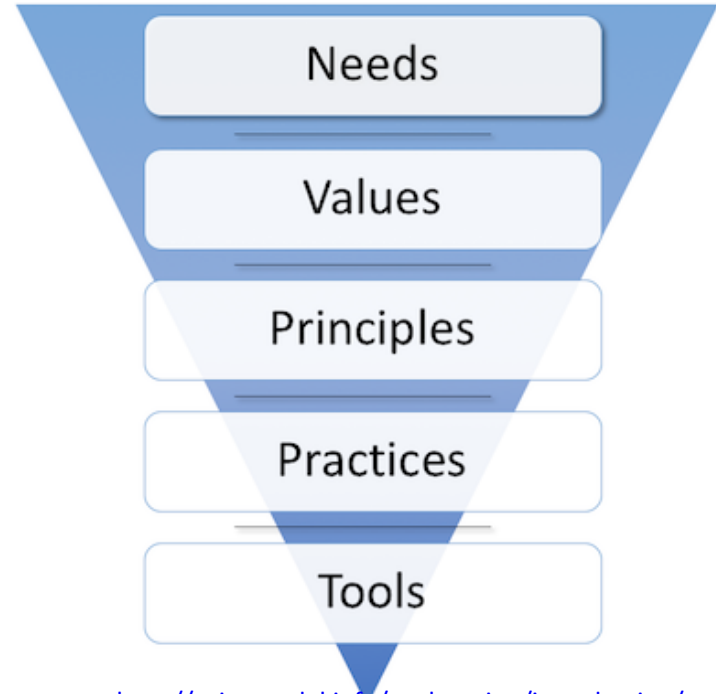
EVALUATION – ARE MICROSERVICES A **GOOD FIT?**

- “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**

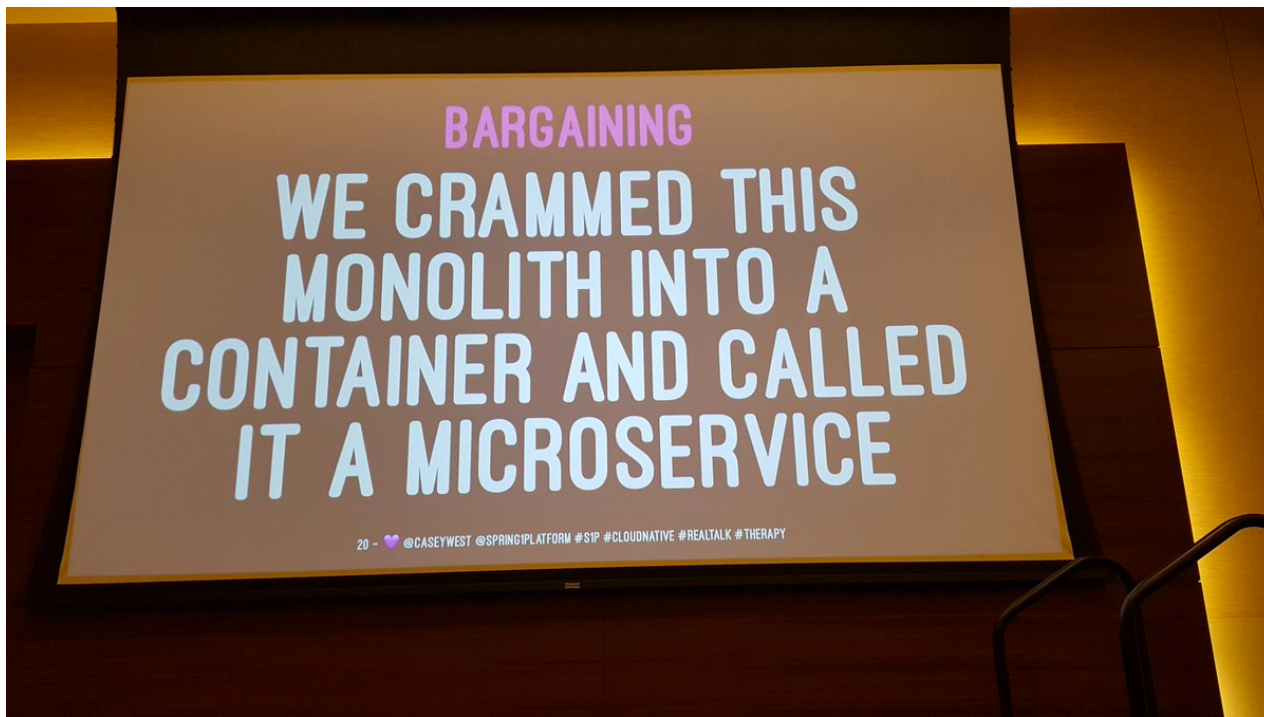


<http://spinemodel.info/explanation/introduction/>

AN EXAMPLE: TO CONTAINERISE, OR NOT TO CONTAINERISE?

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

STRATEGY #FAIL

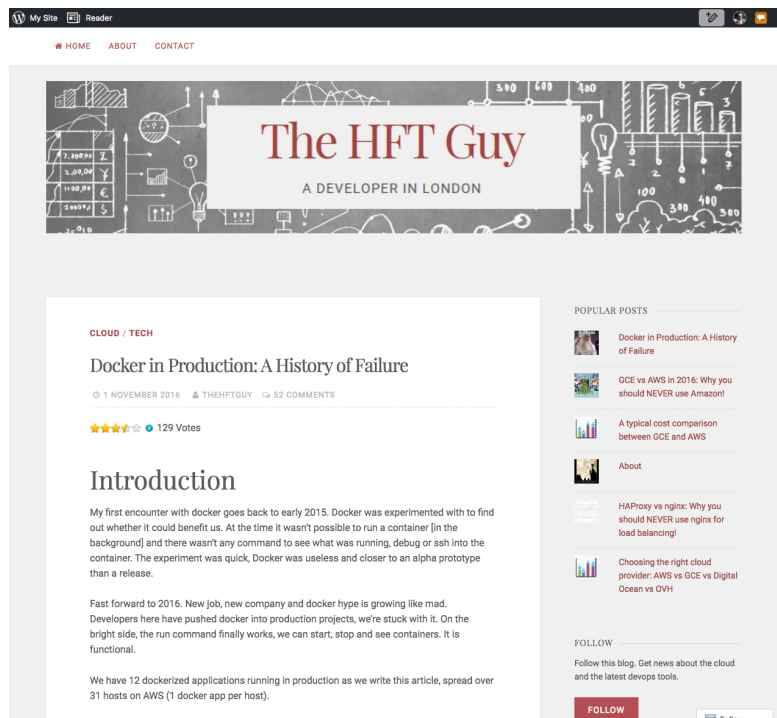


ARCHITECTURE/OPS: EXPECTATIONS VERSUS REALITY



“DevOps”

CHOICES: BEWARE OF CONFIRMATION BIAS



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



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 engine.

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 counteract 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

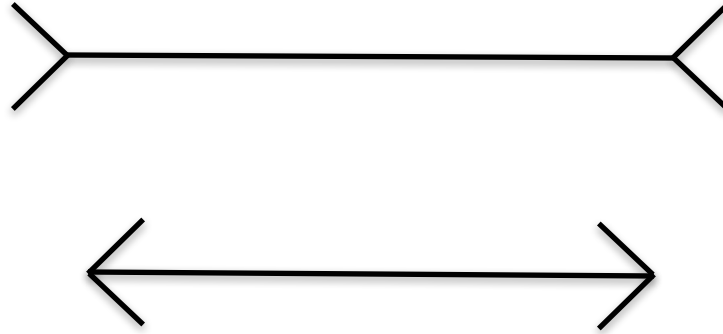
<http://patrobinson.github.io/2016/11/05/docker-in-production/>

01/05/2017

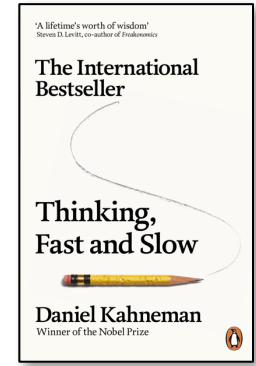
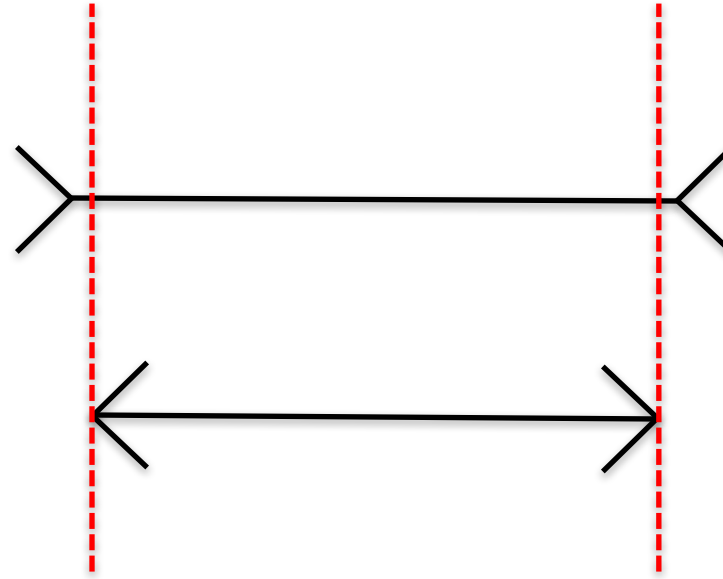
@danielbryantuk

SpectoLabs

EVALUATION - IT'S EASY TO BE TRICKED



EVALUATION - BEWARE OF BIAS AND HEURISTICS





2. GLUTTONY - COMMUNICATION LOCK-IN

RPC – NOT THE DEVIL IN DISGUISE

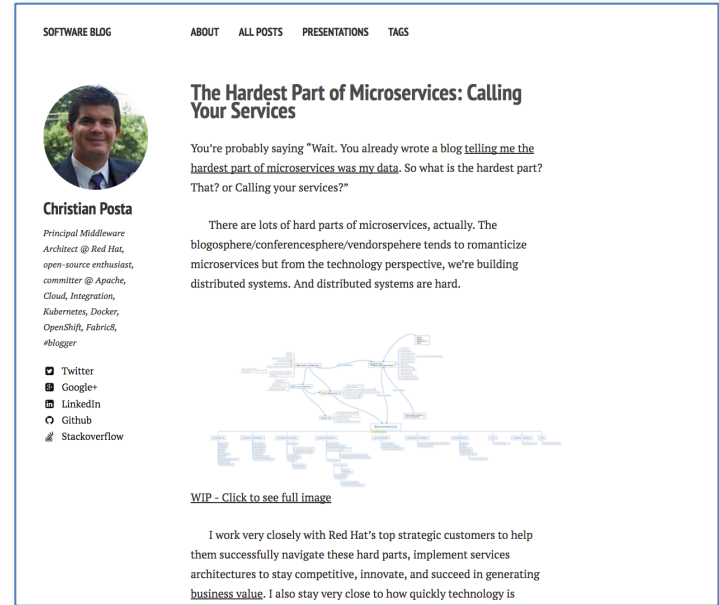
The logo for gRPC, featuring the letters 'gRPC' in a teal color. The 'g' is stylized with an upward-pointing arrow on its left side, and the 'P' has a downward-pointing arrow on its right side.

Apache Thrift™



- 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

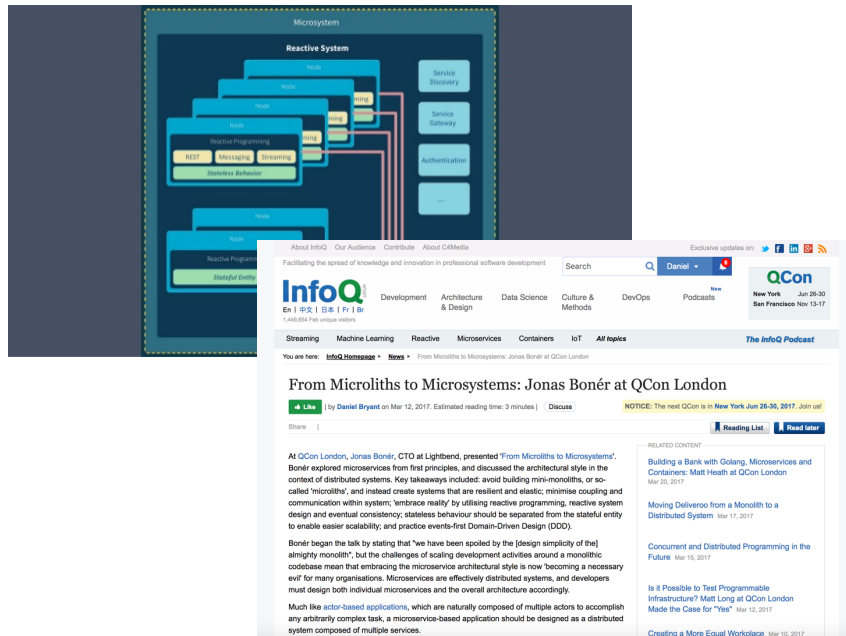
DELEGATION OF COMMS OPERABILITY



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

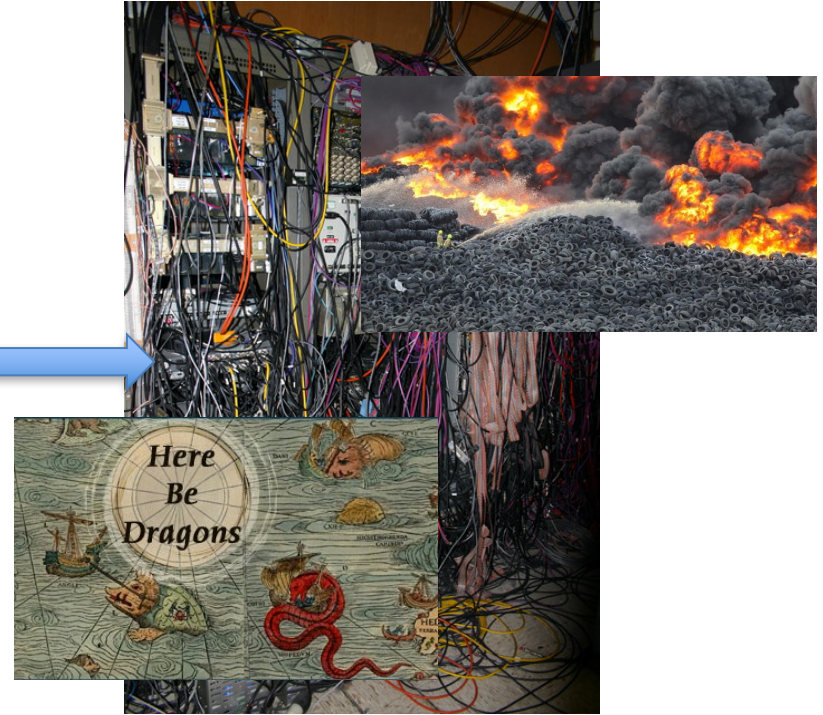
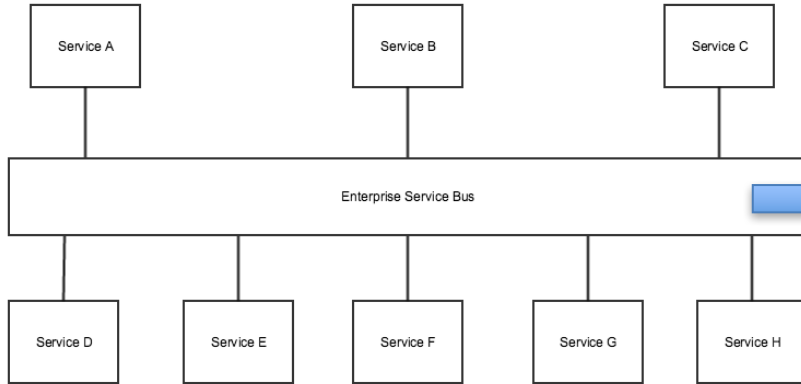
RPC - NOT THE DEVIL IN DISGUISE

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

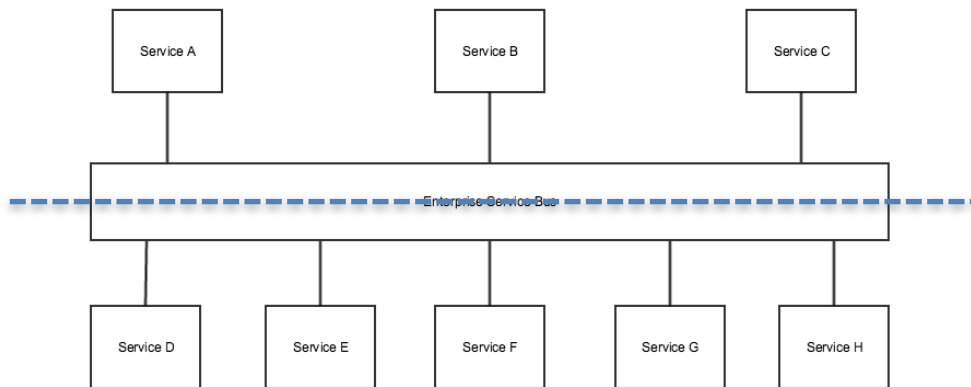


www.infoq.com/news/2017/03/microliths-microsystems

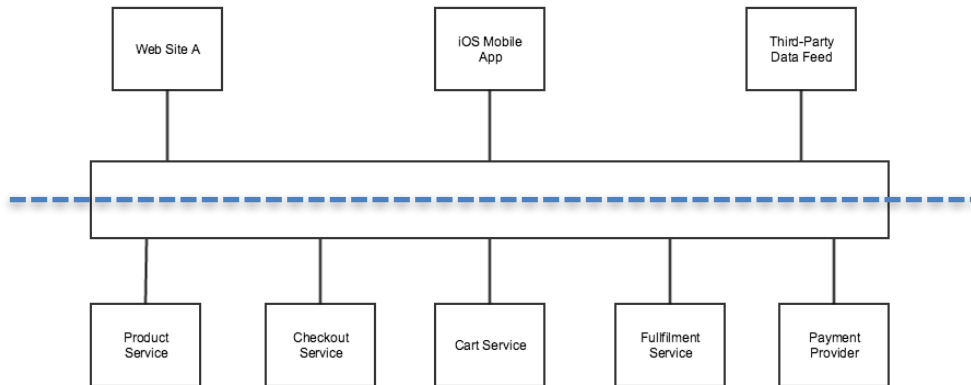
THE ESB IS DEAD - LONG LIVE THE ESB!



THE ESB IS DEAD - LONG LIVE THE ESB!

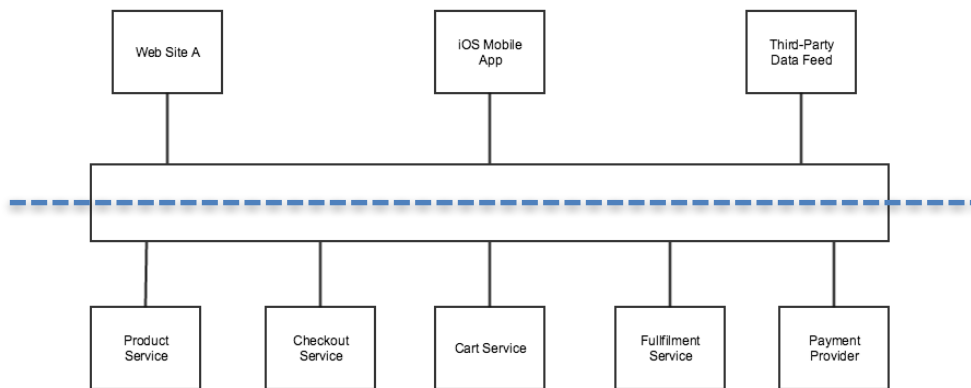


THE ESB IS DEAD - LONG LIVE THE ESB!



- 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)

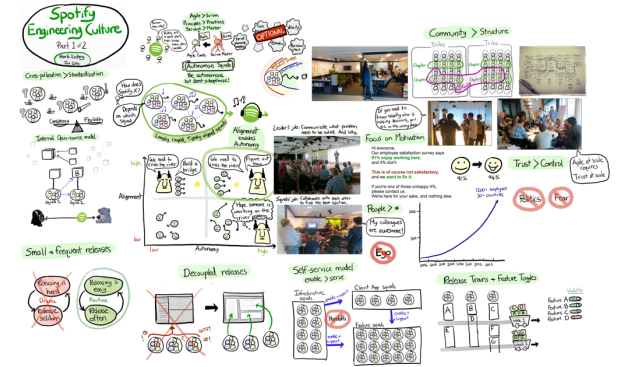
PREVIOUSLY...

- 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 NFIRS - SECURITY

Be the first to clip this slide

ThoughtWorks®

APPSEC & MICROSERVICES

Sam Newman
Velocity 2016

1 of 100

AppSec & Microservices - Velocity 2016 1,408 views

Share Like Download

Sam Newman, Consultant
+ Follow

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

Facilitating the spread of knowledge and innovation in professional software development

InfoQ® Development Architecture & Design Data Science Culture & Methods DevOps

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

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

RELATED CONTENT

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 - "same 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

Amir Chaudhry on Unikernels, Docker Aug 07, 2018

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

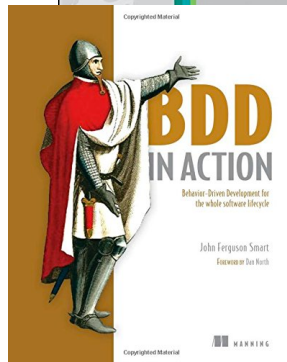
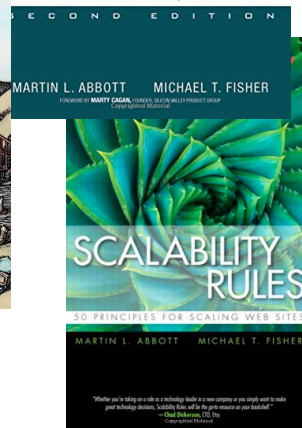
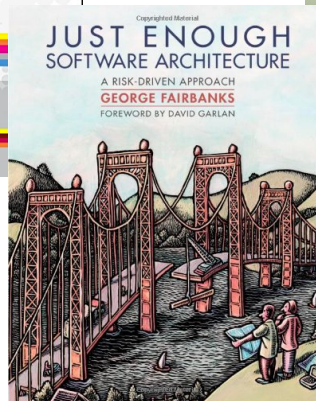
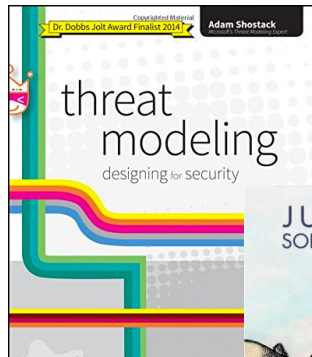
Is Mesos DC/OS a Better Way to Run Docker on AWS? Jul 24, 2016

Modern iOS Application Security

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

TESTING NFERS IN THE BUILD PIPELINE

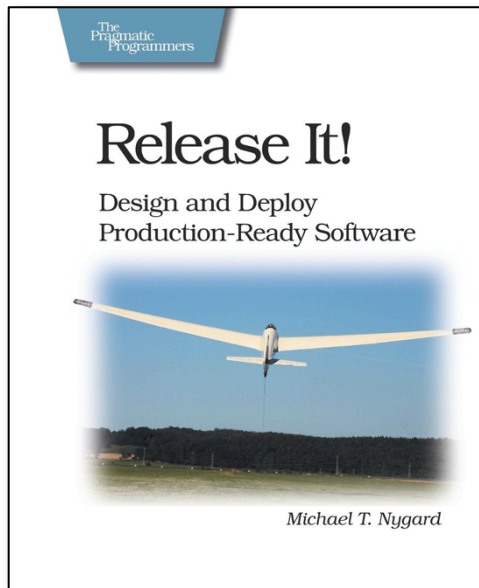
- PERFORMANCE AND LOAD TESTING
 - GATLING / JMETER
 - FLOOD.IO
- SECURITY TESTING
 - FINDSECBUGS / OWASP DEPENDENCY CHECK
 - BDD-SECURITY (OWASP ZAP) / ARACHNI
 - GAUNTLT / SERVERSPEC
 - DOCKER BENCH FOR SECURITY / CLAIR





5. WRATH – BLOWING UP WHEN **BAD THINGS** HAPPEN

PREVIOUSLY - BRING IN MICHAEL NYGARD (OR SOME MONKEYS)



WHEN **BAD THINGS** HAPPEN, **PEOPLE** ARE ALWAYS INVOLVED



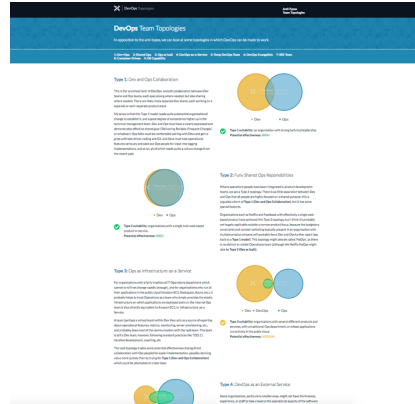
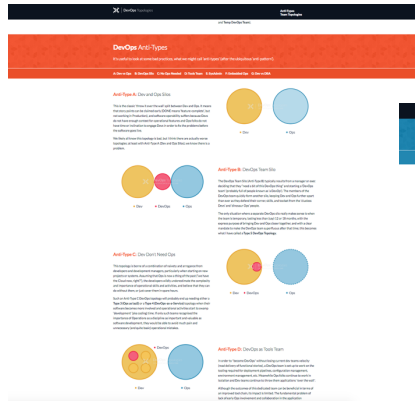
01/05/2017

@danielbryantuk | @oakinger

SpectoLabs

PEOPLE PAIN POINT – HOW DOES DEVOPS FIT INTO THIS?

- [HTTP://WEB.DEVOPSTOPOLOGIES.COM/](http://web.devopstopologies.com/)
- @ MATTHEWPSKELTON



BOOKS



DEVOPS – THE 'FULLSTACK ENGINEER' MYTH

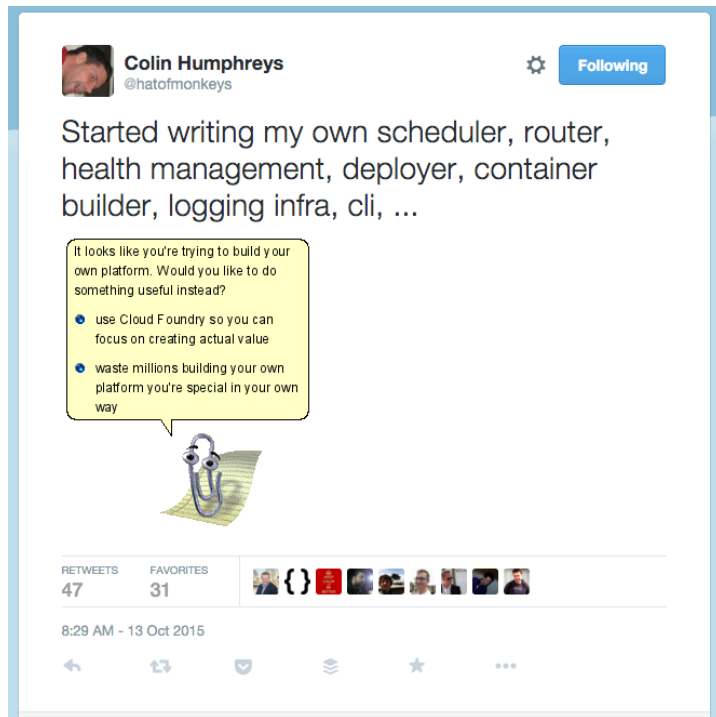
**“I’M SORRY, BUT IF YOU’RE NOT DESIGNING THE COMPUTER CHIPS AND
WRITING THE WEBSITE, THEN I DON’T WANNA HEAR FROM YOU”**

CHARITY MAJORS (@MIPSYTIPSY), CRAFTCONF 2016

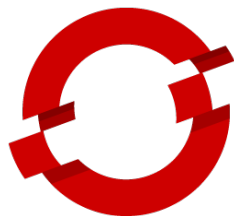
[HTTP://WWW.USTREAM.TV/RECORDED/86181845](http://www.ustream.tv/recorded/86181845)

DEVOPS – DEFINE RESPONSIBILITIES

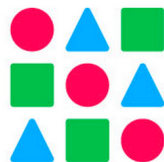
- DO YOU REALLY WANT TO BUILD AN **ENTIRE MICROSERVICES PLATFORM?**
- FOCUS ON **WHAT MATTERS**
 - CI/CD
 - MECHANICAL SYMPATHY
 - LOGGING
 - MONITORING



WORTH CONSIDERING: OPEN SOURCE PAAS/FAAS/DBAAS



OPENSIFT[®]
by Red Hat[®]



DEIS



CLOUD FOUNDRY



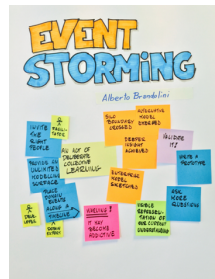
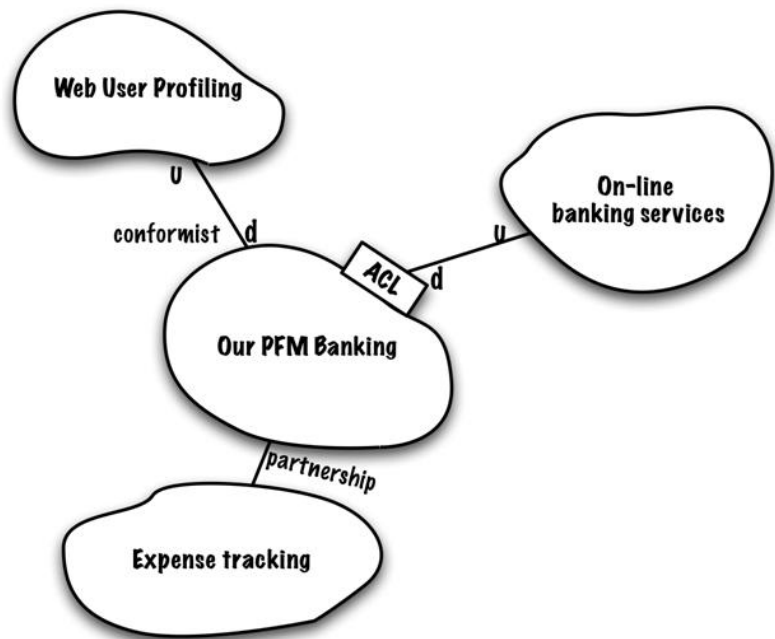
6. ENVY – THE **SHARED SINGLE DOMAIN (AND DATA STORE)** FALLACY

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)

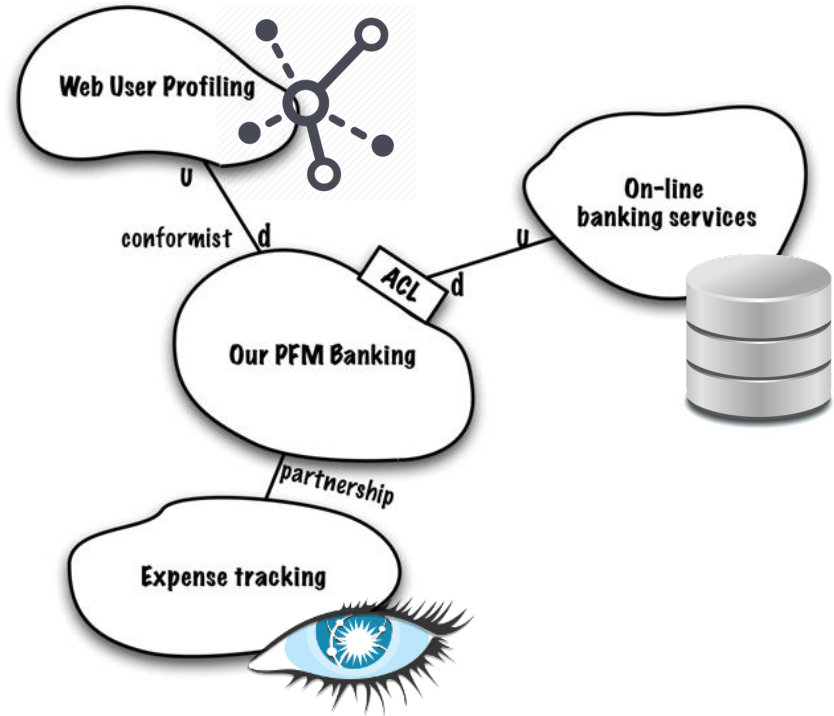


ziobrando.blogspot.co.uk/2013/11/introducing-event-storming.html

www.infoq.com/articles/ddd-contextmapping

CHOOSE (AND USE) DATA STORES APPROPRIATELY

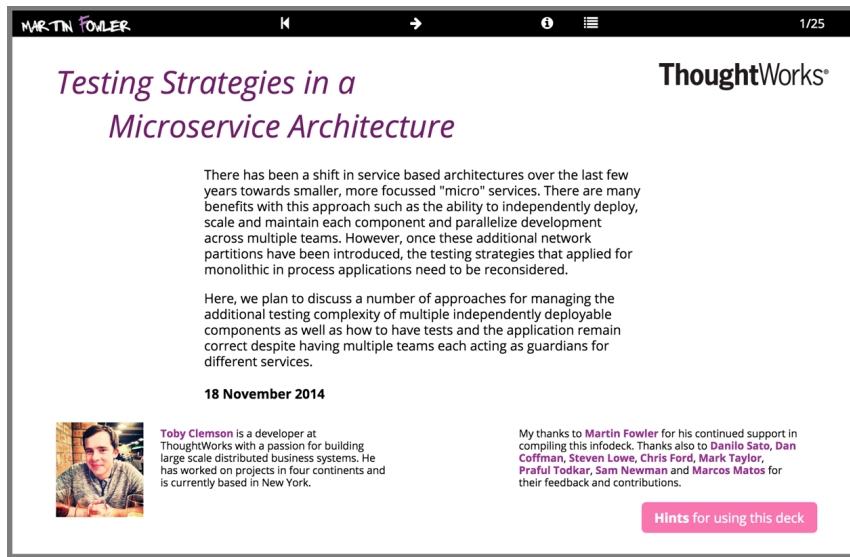
- RDBMS
 - VALUABLE FOR **STRUCTURED** DATA
- **CASSANDRA** IS AWESOME
 - BUT DON'T TREAT IT LIKE AN RDBMS!
- DON'T BUILD A GRAPH WITH RDBMS
 - USE **NEO4J, TITAN** ETC
- BEWARE OF OPERATIONAL OVERHEAD





7. PRIDE – TESTING IN THE **WORLD OF TRANSIENCE**

PREVIOUSLY...



MARTIN FOWLER


ThoughtWorks®

Testing Strategies in a Microservice Architecture

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

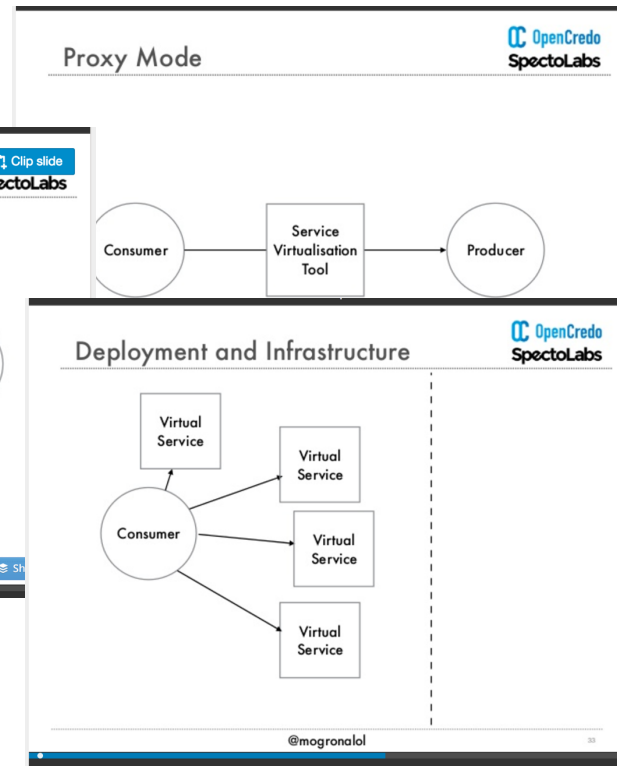
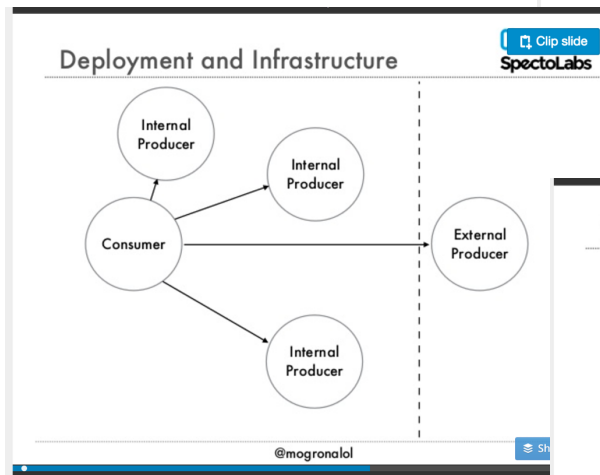
SERVICE VIRTUALISATION / API SIMULATION

- VIRTUALISE REQUEST/RESPONSE OF SERVICES

- UNAVAILABLE
- EXPENSIVE TO RUN
- FRAGILE/BRITTLE
- NON-DETERMINISTIC
- CANNOT SIMULATE FAILURES

[HTTPS://DZONE.COM/ARTICLES/CONTINUOUSLY-DELIVERING-SOA](https://dzone.com/articles/continuously-delivering-soa)

ANDREW MORGAN'S TALK [HTTP://BIT.LY/20VOECD](http://bit.ly/20VOECD)



SERVICE VIRTUALISATION

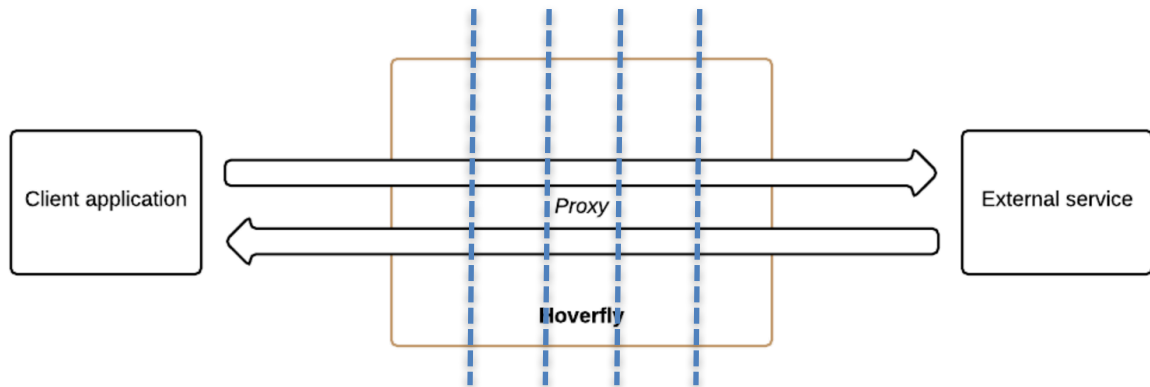
- **CLASSICS**
 - CA SERVICE VIRTUALIZATION
 - PARASOFT VIRTUALIZE
 - HPE SERVICE VIRTUALIZATION
 - IBM TEST VIRTUALIZATION SERVER
- **NEW (OPEN SOURCE) KIDS ON THE BLOCK**
 - HOVERFLY
 - WIREMOCK
 - VCR/BETAMAX
 - MOUNTEBANK
 - MIRAGE

HOVERFLY



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

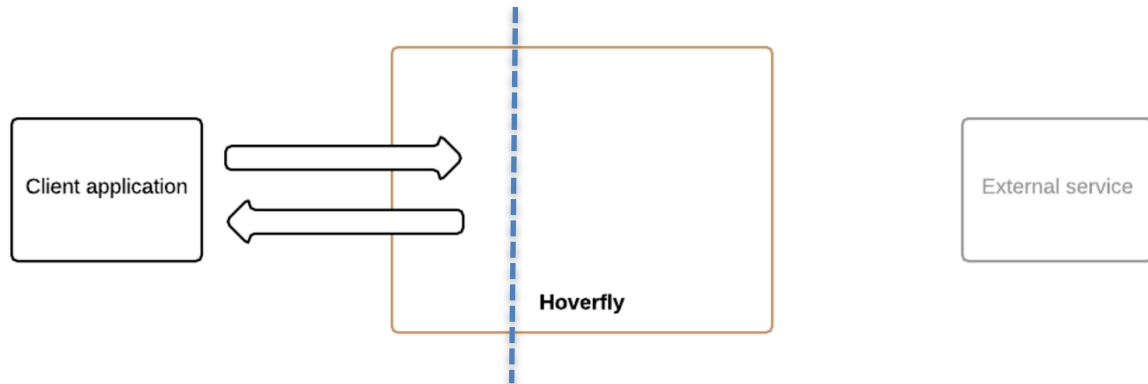
Capture mode



- 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 static void capture() {

    // After the capture
    @ClassRule
    public static void afterCapture() {

    // Or you can use the DSL
    @ClassRule
    public static void dsl() {
```

Create API simulation using DSL

```
@ClassRule
public static HoverflyRule rule =
    HoverflyRule.of()
        .service("www.my-test.com")
        .get("/api/bookings")
        .willReturn(succesfulResponse());
});

@Test
public void shouldBeAbleToBook() {
    // When
    final ResponseEntity<Book> response =
        restTemplate.getForEntity("http://www.my-test.com/api/bookings",
            Book.class);

    // Then
    assertEquals("book flight", response.getBody());
    assertEquals("book flight", response.getBody());
}
```

```
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())
```

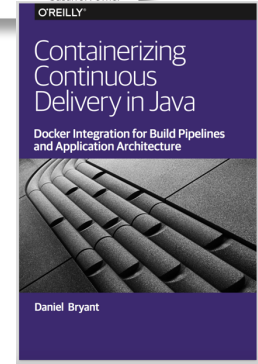
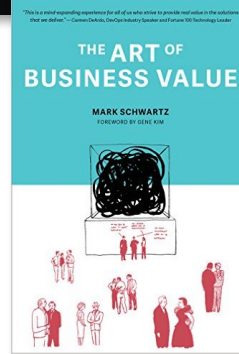
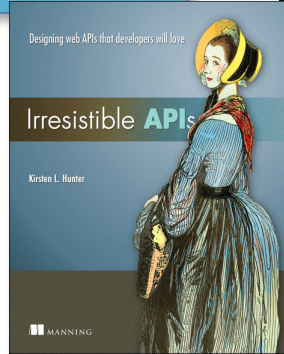
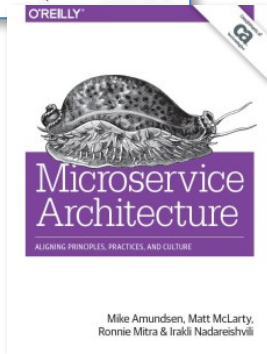
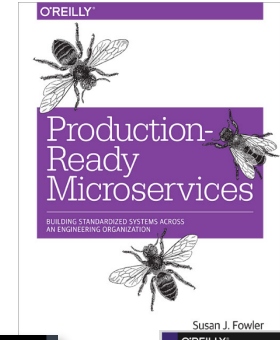
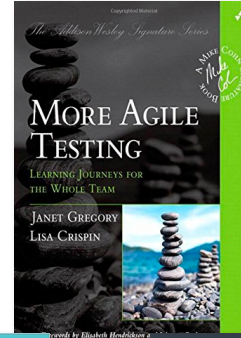
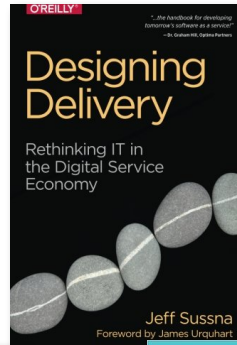
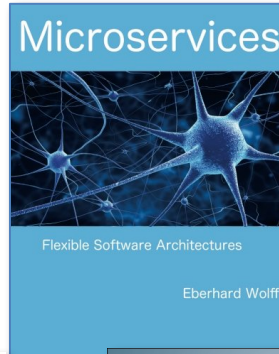
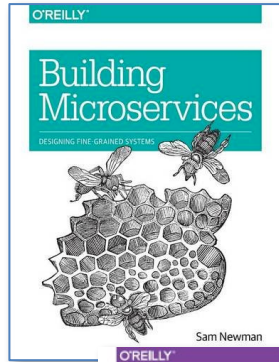
github.com/SpectoLabs/hoverfly-java

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 NFERS
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



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

[HTTP://SPECTO.IO](http://specto.io)

[MUSERVICESWEEKLY.COM](http://muservicesweekly.com)

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