

Microservices

@Steve_Upton

Who am I?

- Steve Upton
- English / Welsh / British / Irish (soon)
- BSc. Computer Science (Cardiff University)
- Physics & Astronomy (Open University)
- IBM (6 years)
 - Messaging
 - OASIS MQTT TC member
 - Working with clients on Microservice systems
 - London μ Service user group
- HERE Berlin (2 years)
 - Microservices and robots
- ThoughtWorks
 - Lead QA Consultant



Who are you?

Microservices

The histories of Microservices

Why you shouldn't do Microservices

What Microservices actually look like

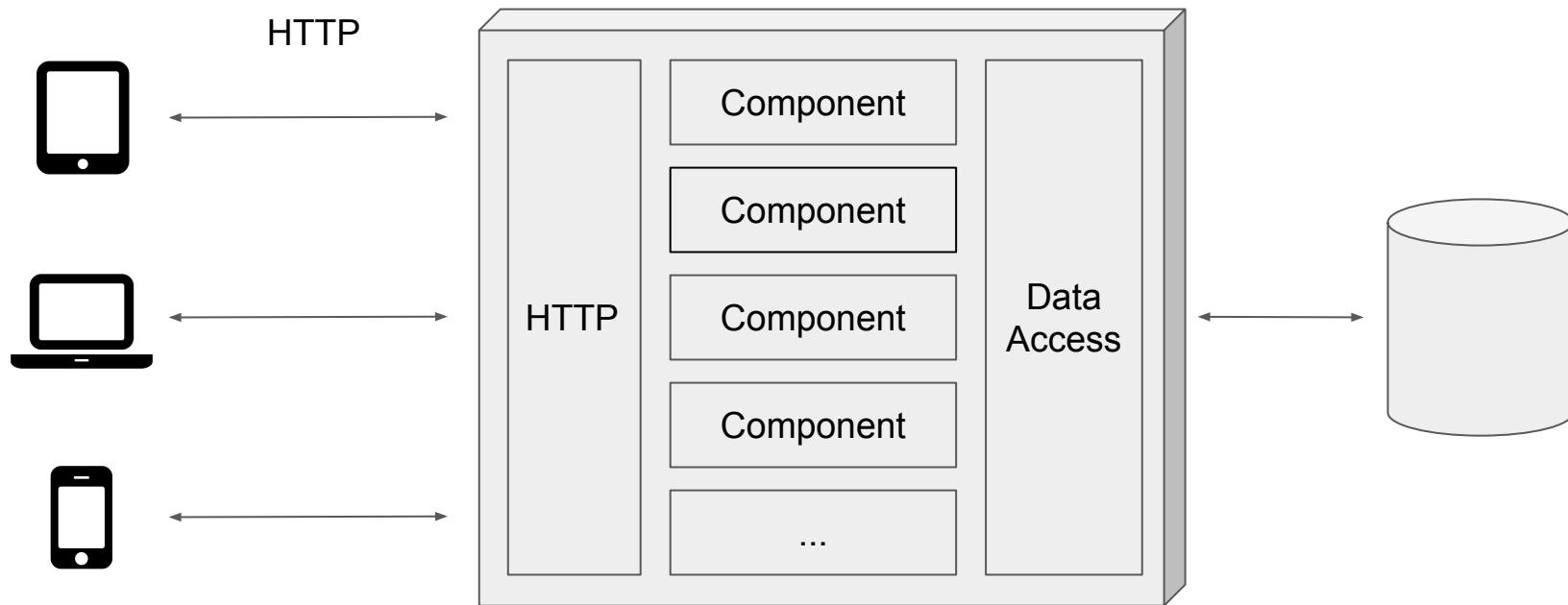
The future of Microservices

The histories of Microservices

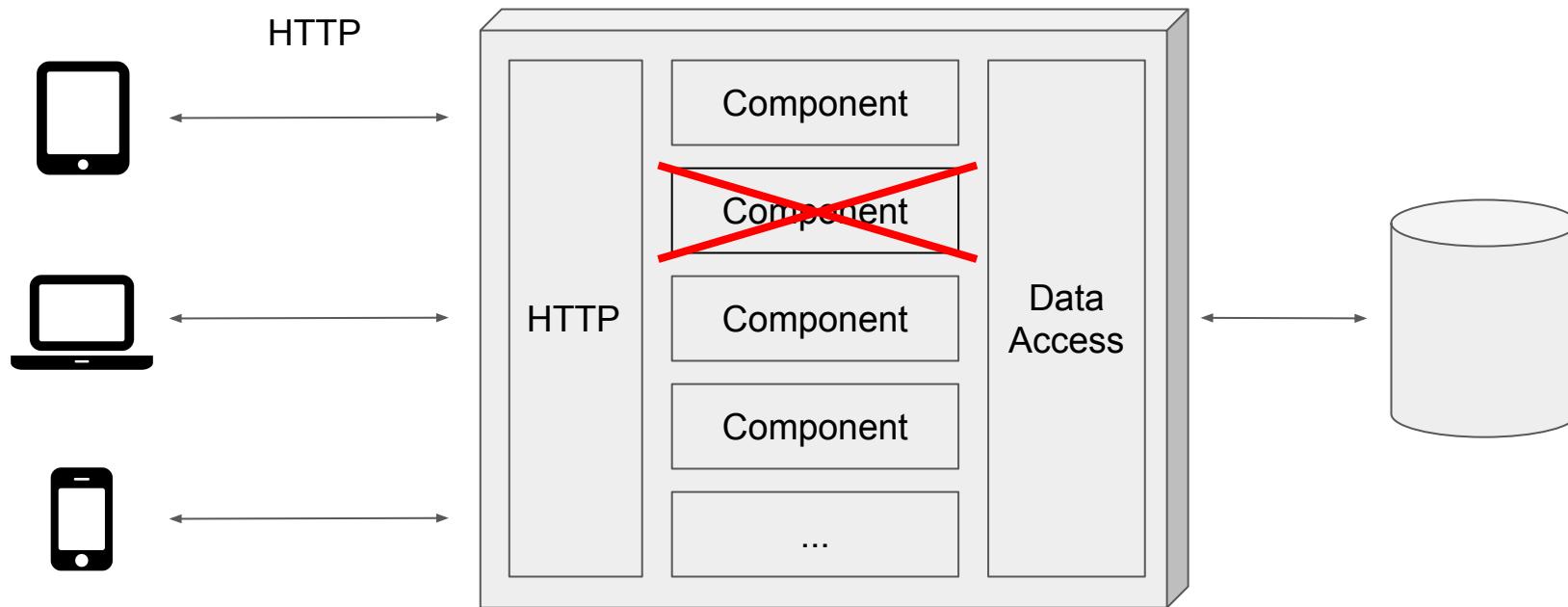
The standard Microservices creation story

NETFLIX

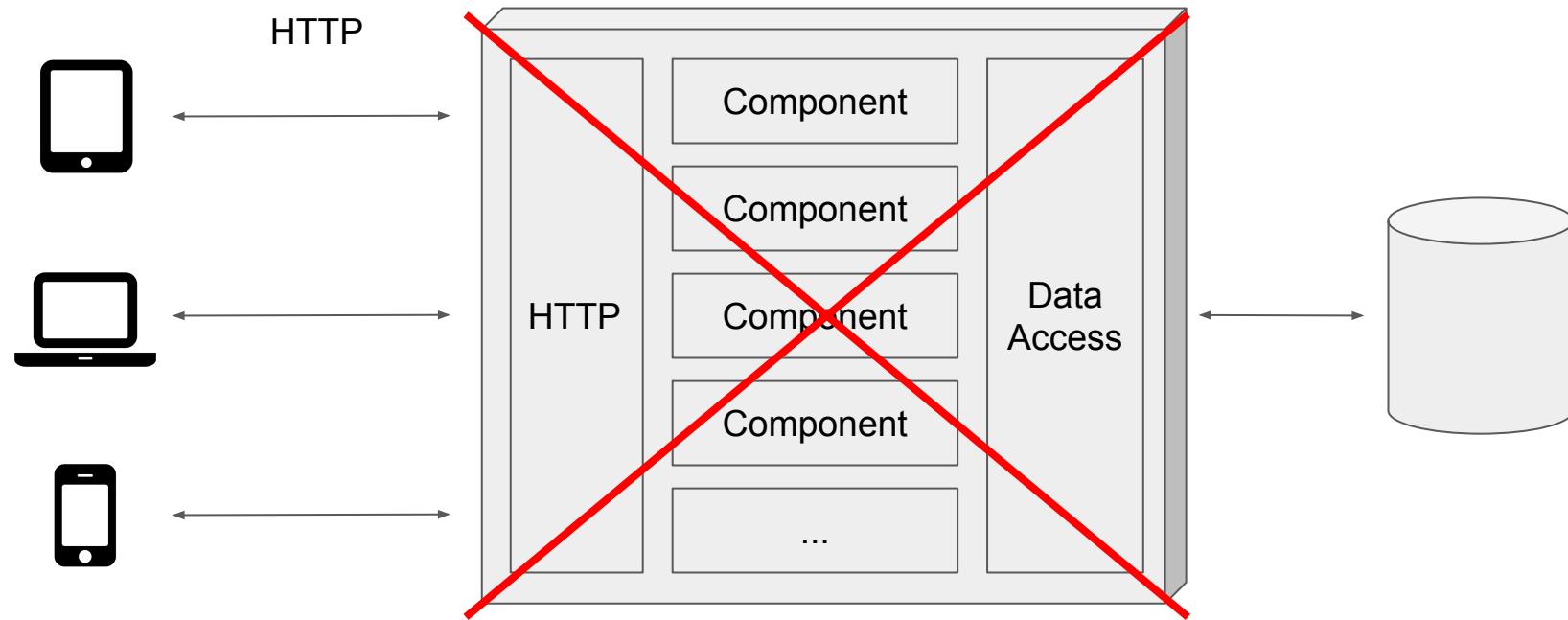
Monolith



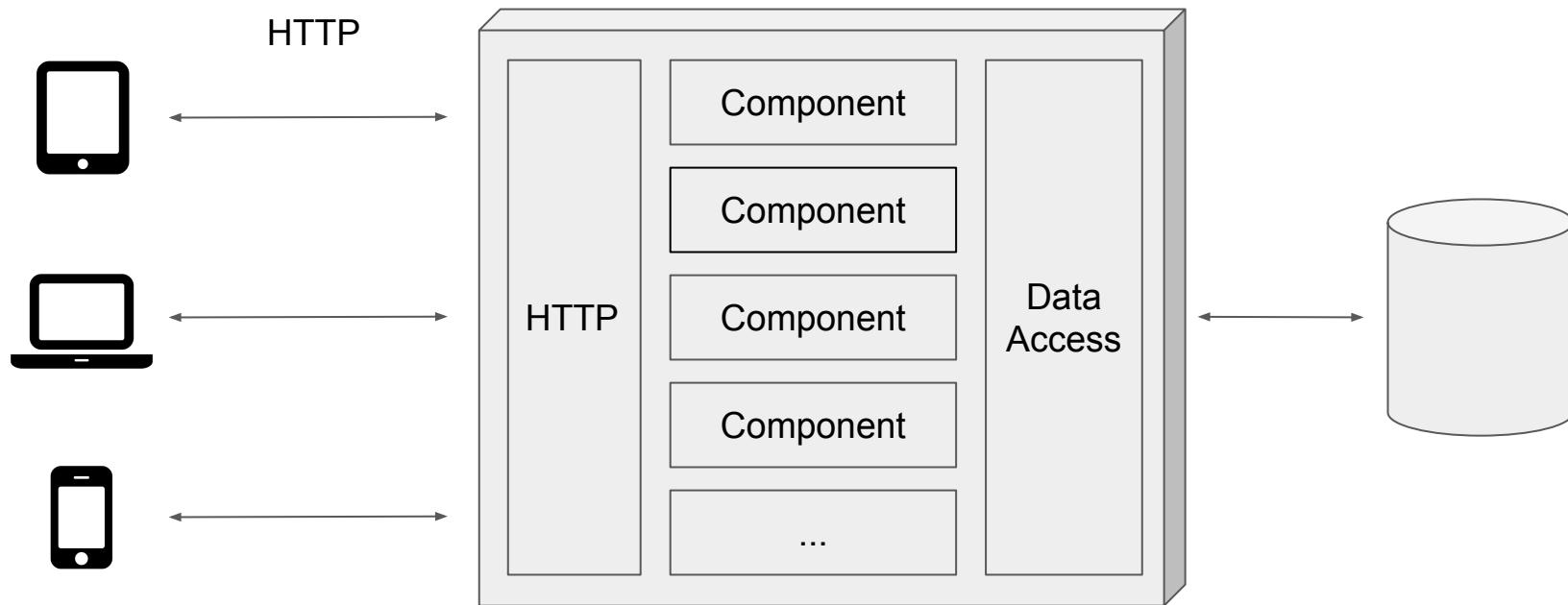
Monolith



Monolith



Monolith



Hard to scale

Hard to scale

RDBMS was also a **SPOF**

Hard to scale

RDBMS was also a **SPOF** and **bottleneck**

Hard to scale

RDBMS was also a **SPOF** and **bottleneck**

Vertical scaling easy, but **limited**

Hard to scale

RDBMS was also a **SPOF** and **bottleneck**

Vertical scaling easy, but **limited**

Horizontal scaling helpful, but **slow and expensive**

Hard to scale

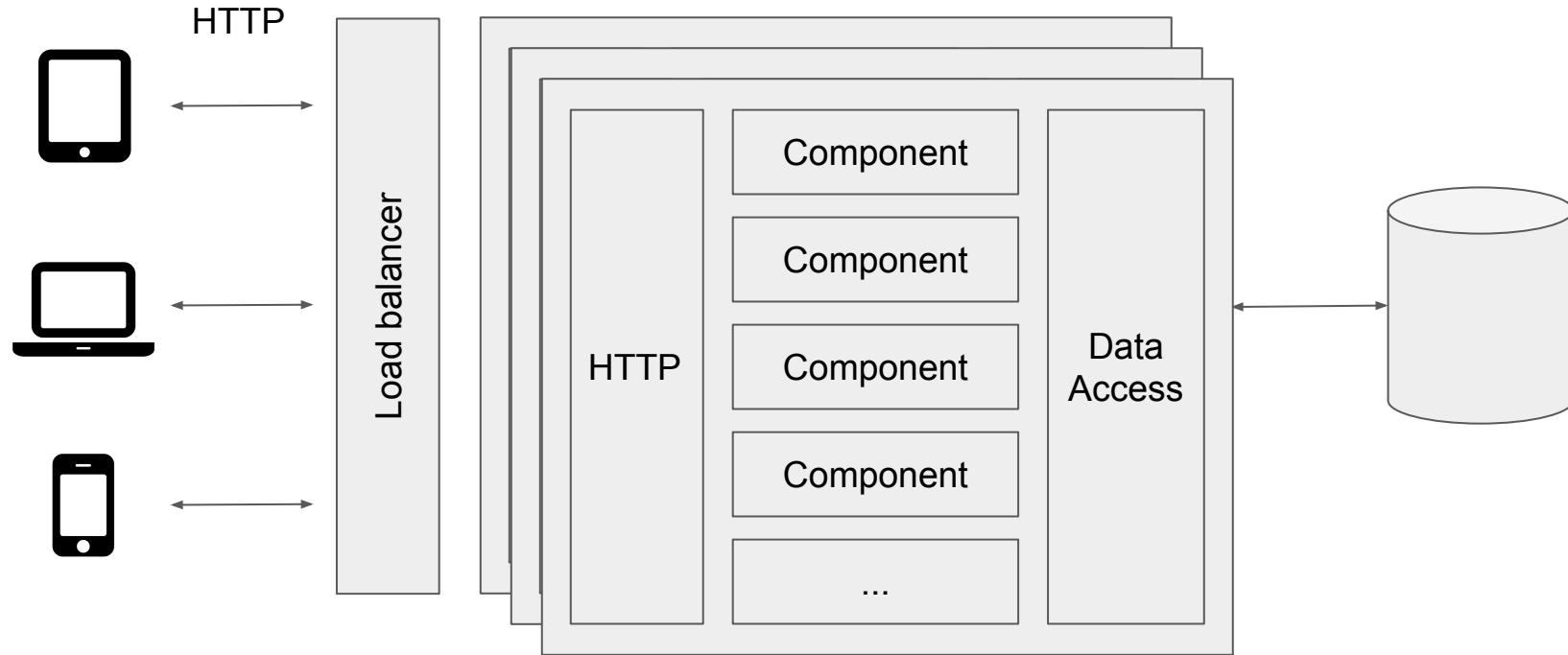
RDBMS was also a **SPOF** and **bottleneck**

Vertical scaling easy, but **limited**

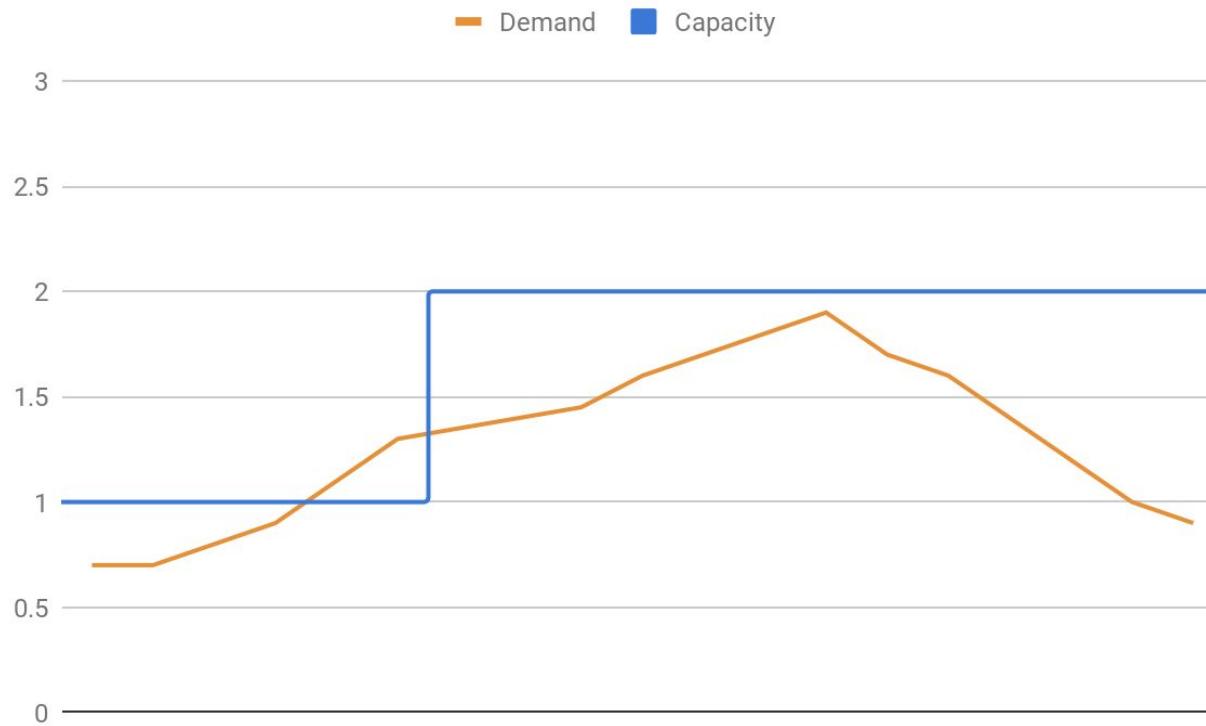
Horizontal scaling helpful, but **slow and expensive**

Scaling the whole monolith was **wasteful**

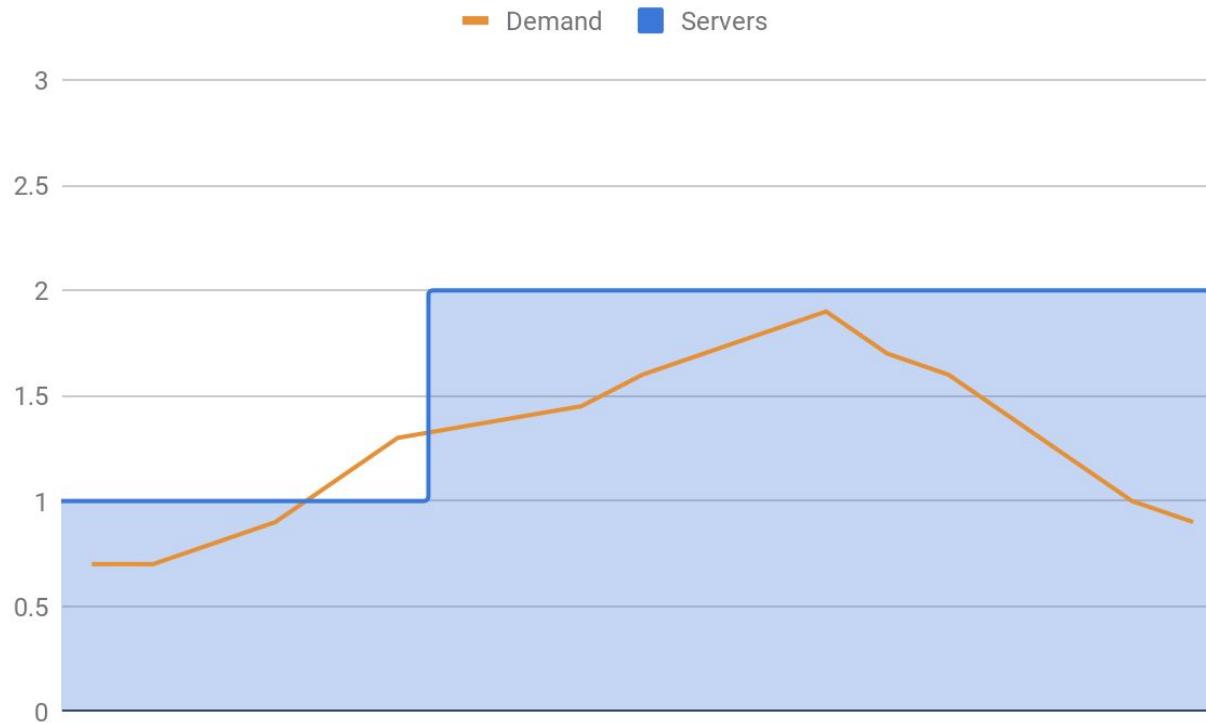
Scaling the monolith



Scaling the monolith



Scaling the monolith



“...the average server operates at no more than
12 to 18 percent of its capacity”

Natural Resources Defense Council

Tightly coupled, slow to change

Tightly coupled, slow to change

Sharing knowledge of components

Difficult to upgrade without affecting other services

Tightly coupled, slow to change

Sharing knowledge of components

Difficult to upgrade without affecting other services

Sharing libraries, platform, OS etc.

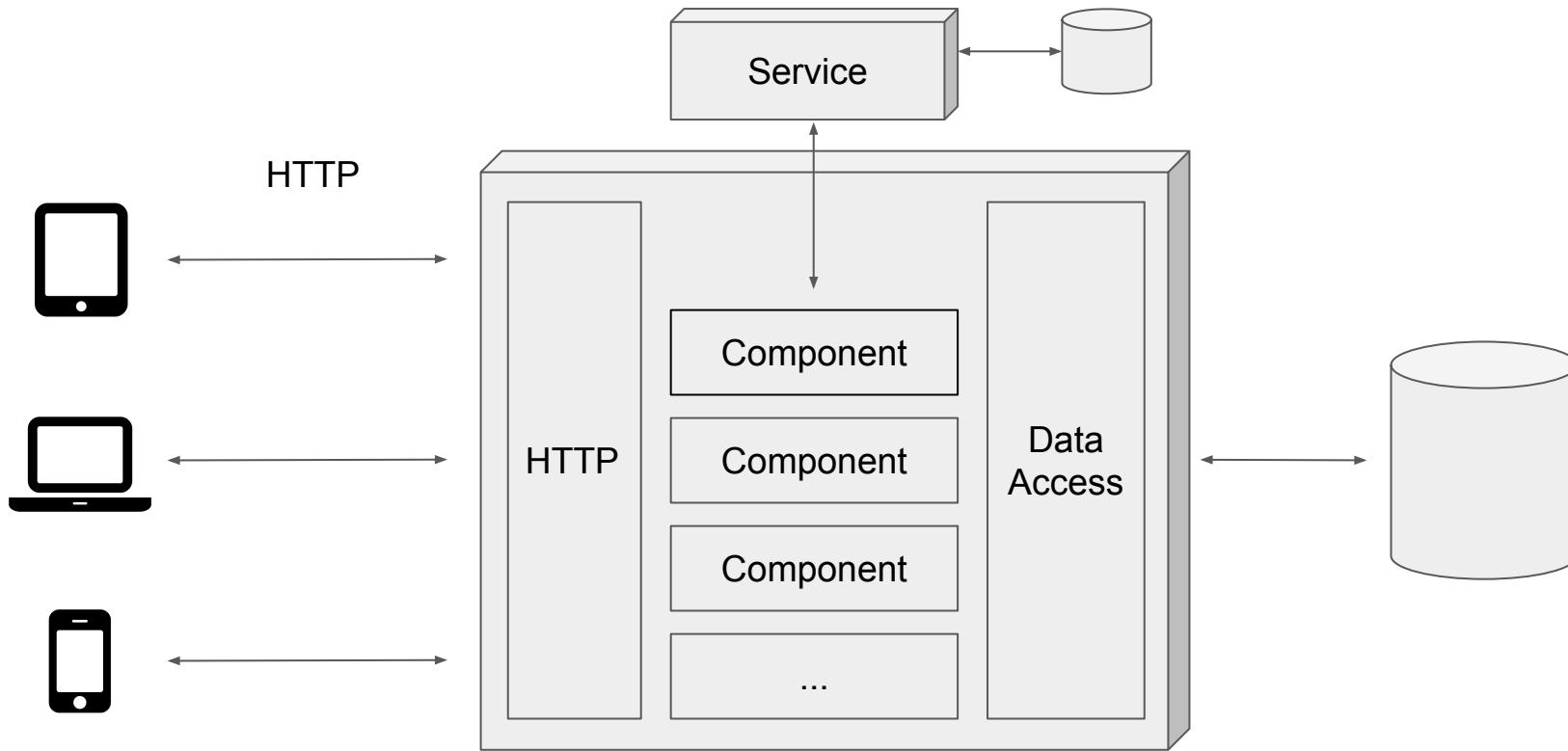
Teams have **little choice** in setup



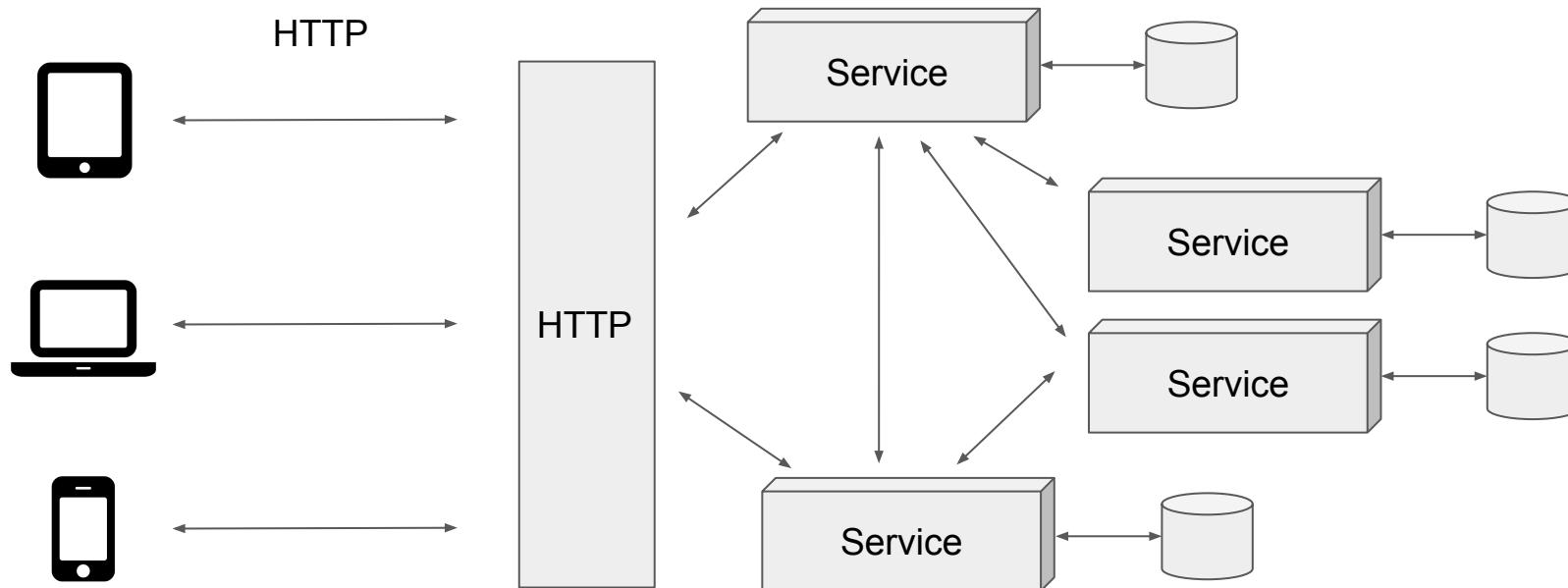




Moving to the cloud



Microservices



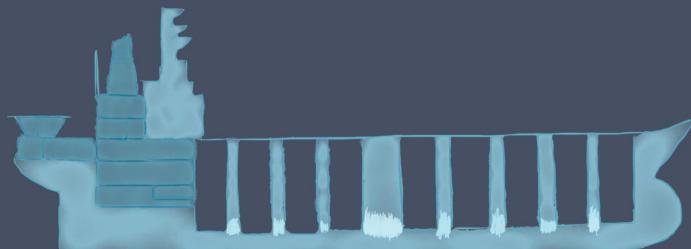
Resilience to failure

Able to **isolate** failures

Resilience to failure

Able to **isolate** failures

BULKHEADING

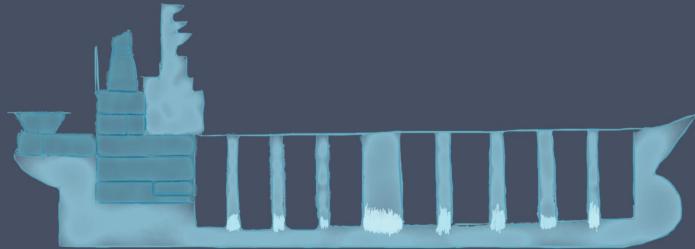


Resilience to failure

Able to **isolate** failures

Gracefully adapt to failures

BULKHEADING

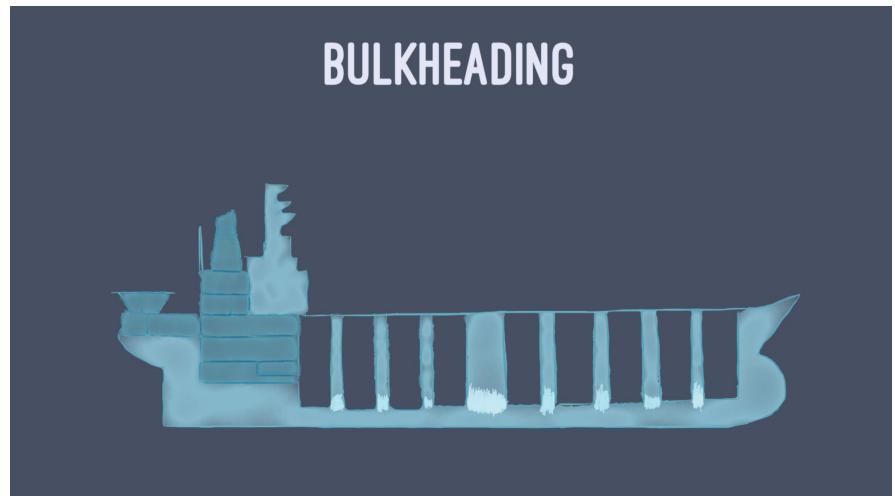


Resilience to failure

Able to **isolate** failures

Gracefully adapt to failures

Quick **recovery**



Loosely coupled, easy to change

Share **contracts**, not internals

Little to no knowledge of other services needed

Loosely coupled, easy to change

Share **contracts**, not internals

Little to no knowledge of other services needed

Teams **free to choose** OS, language, libraries etc.

Free to **upgrade** and **experiment independently**

Easy to scale

Vertical scaling easy and **more effective**

Easy to scale

Vertical scaling easy and **more effective**

Horizontal scaling also easy!

Easy to scale

Vertical scaling easy and **more effective**

Horizontal scaling also easy!

More **efficient** use of resources

Easy to scale

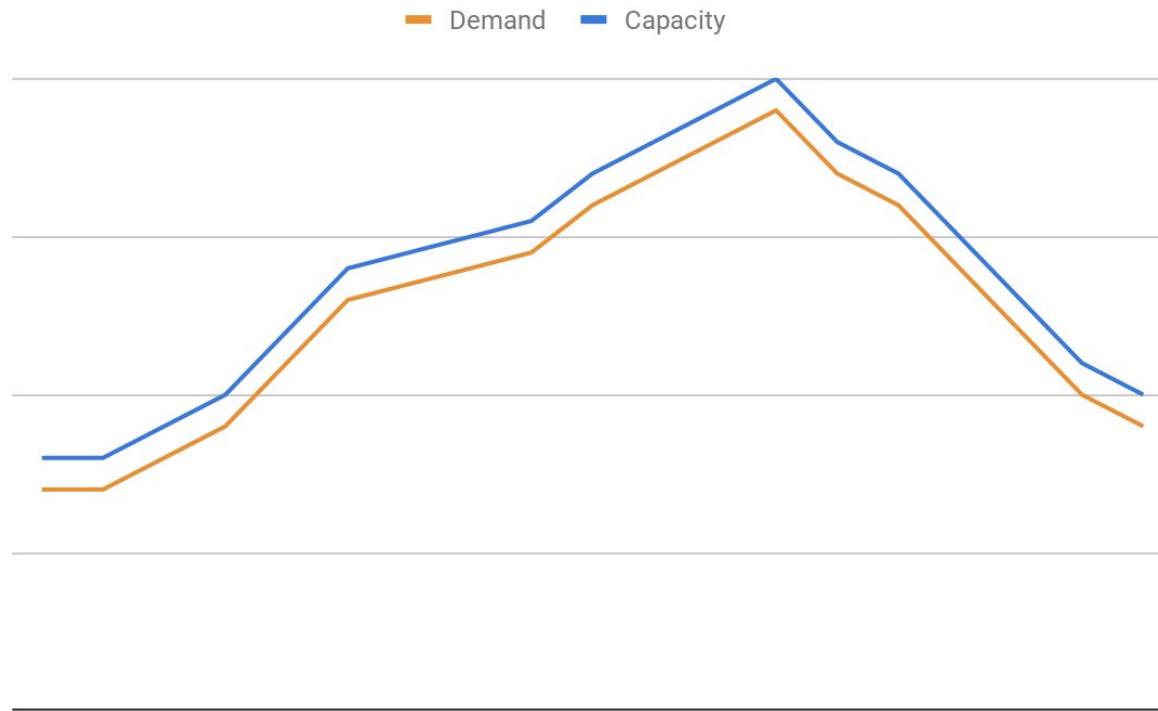
Vertical scaling easy and **more effective**

Horizontal scaling also easy!

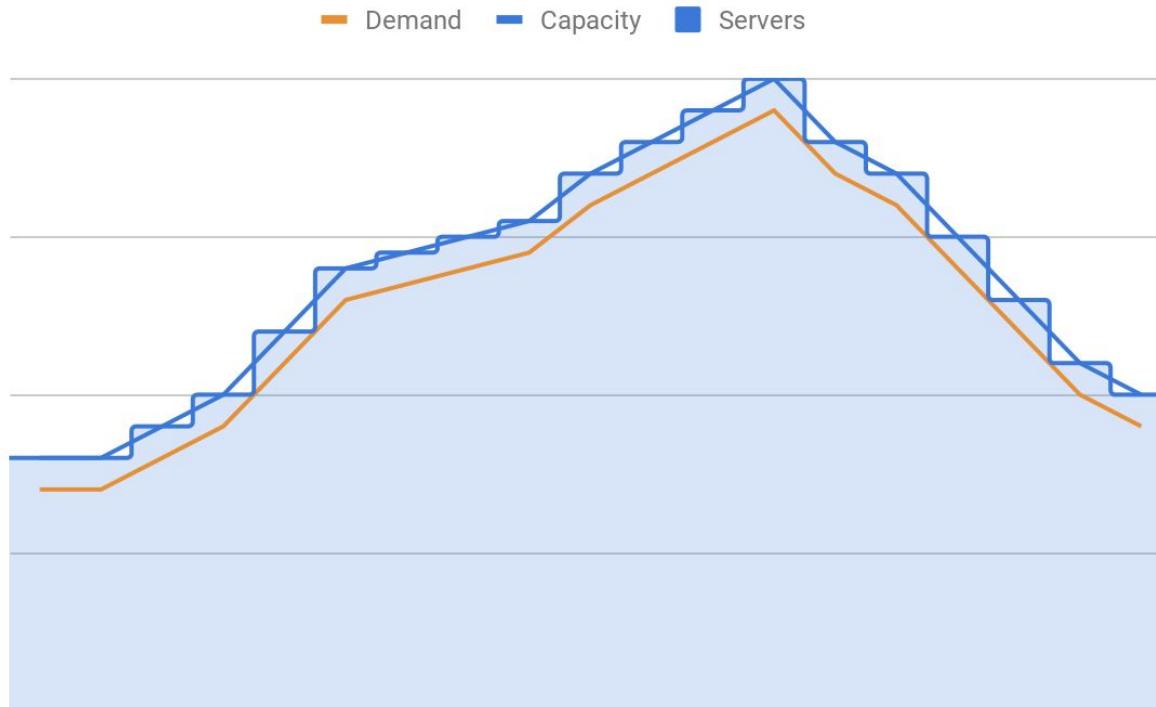
More **efficient** use of resources

Auto scaling of individual services possible

Scaling microservices

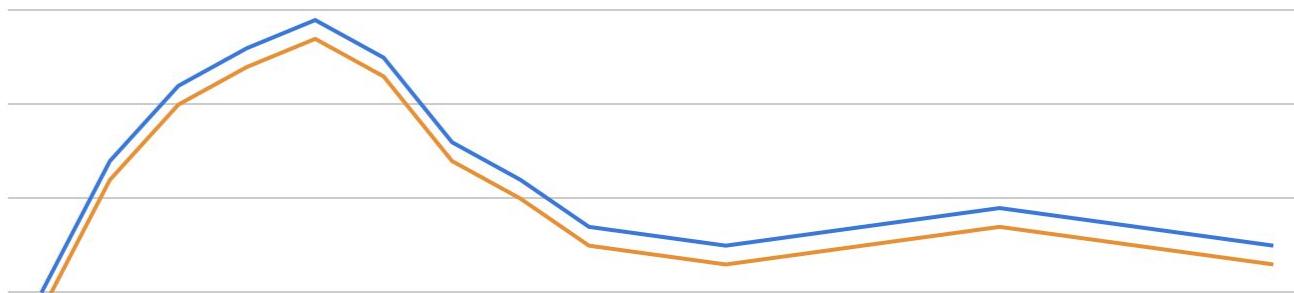


Scaling microservices



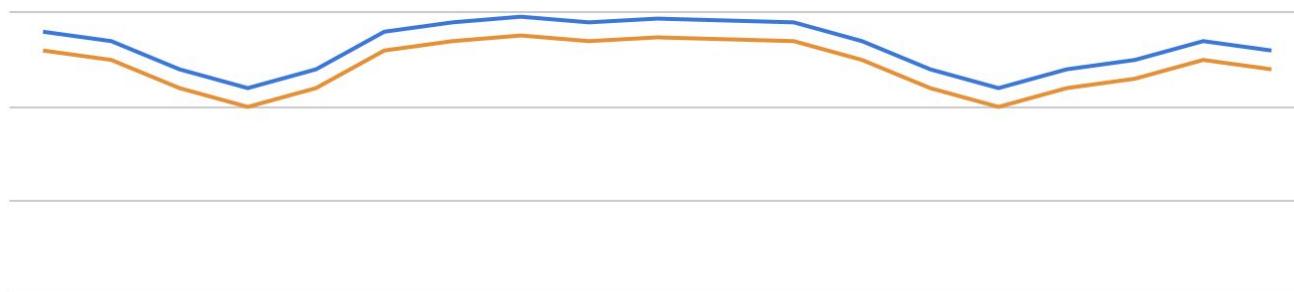
Login service

Demand Capacity



Search service

Demand Capacity



Auto scaling



Amazon EC2

The screenshot shows the AWS Auto Scaling Rule configuration interface. A form is displayed with various fields and dropdown menus. Yellow callout boxes point to specific parts of the configuration:

- Rule name:** Points to the "Rule name" field containing "MyScalingRule".
- Evaluation period:** Points to the "Add 1 Instances" section.
- CloudWatch metric:** Points to the "if" dropdown menu set to "YARNMemoryAvailablePercentage".
- Scaling adjustment:** Points to the "is less than or equal to 15 percent" section.
- Cooldown period:** Points to the "for 1 five-minute periods" section.
- Comparison operator:** Points to the "1 five-minute periods" section.
- Threshold:** Points to the "15 percent" value.

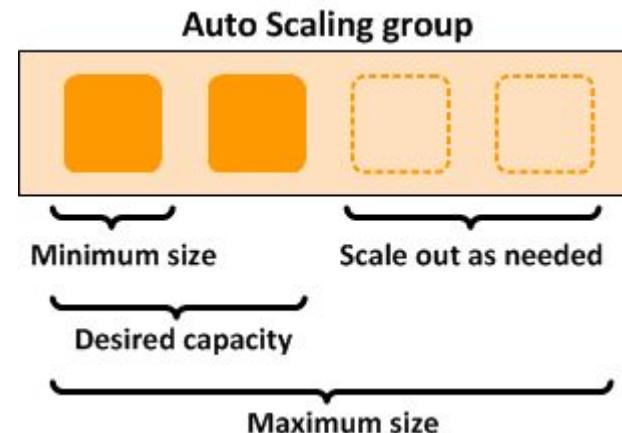
The configuration details are as follows:

- Rule name:** MyScalingRule
- Add:** 1 Instances
- if:** YARNMemoryAvailablePercentage
- is:** less than or equal to 15 percent
- for:** 1 five-minute periods
- Cooldown:** 1 five-minute periods

Auto scaling



Amazon EC2



“Cattle not pets”

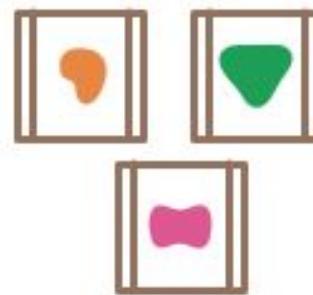
Gavin McCance, CERN

“In short, the microservice architectural style is an approach to developing a single application as a **suite of small services**, each **running in its own process** and communicating with lightweight mechanisms, often an HTTP resource API. These services are **built around business capabilities** and **independently deployable** by fully automated deployment machinery. There is a **bare minimum of centralized management** of these services, which may be written in different programming languages and use different data storage technologies.”

James Lewis and Martin Fowler, ThoughtWorks

**“Loosely coupled service oriented
architecture with bounded contexts”**

Adrian Cockcroft, Architect @ Netflix



EFFICIENCY

(of delivery)

~~EFFICIENCY~~

(of delivery)

SPEED

(of delivery)



<https://netflix.github.io/>

<http://techblog.netflix.com/>



nebula



restify



Atlas



NETFLIX

NETFLIX



ebay



CLOUD FOUNDRY

amazon

Groupon®



zalando

the guardian

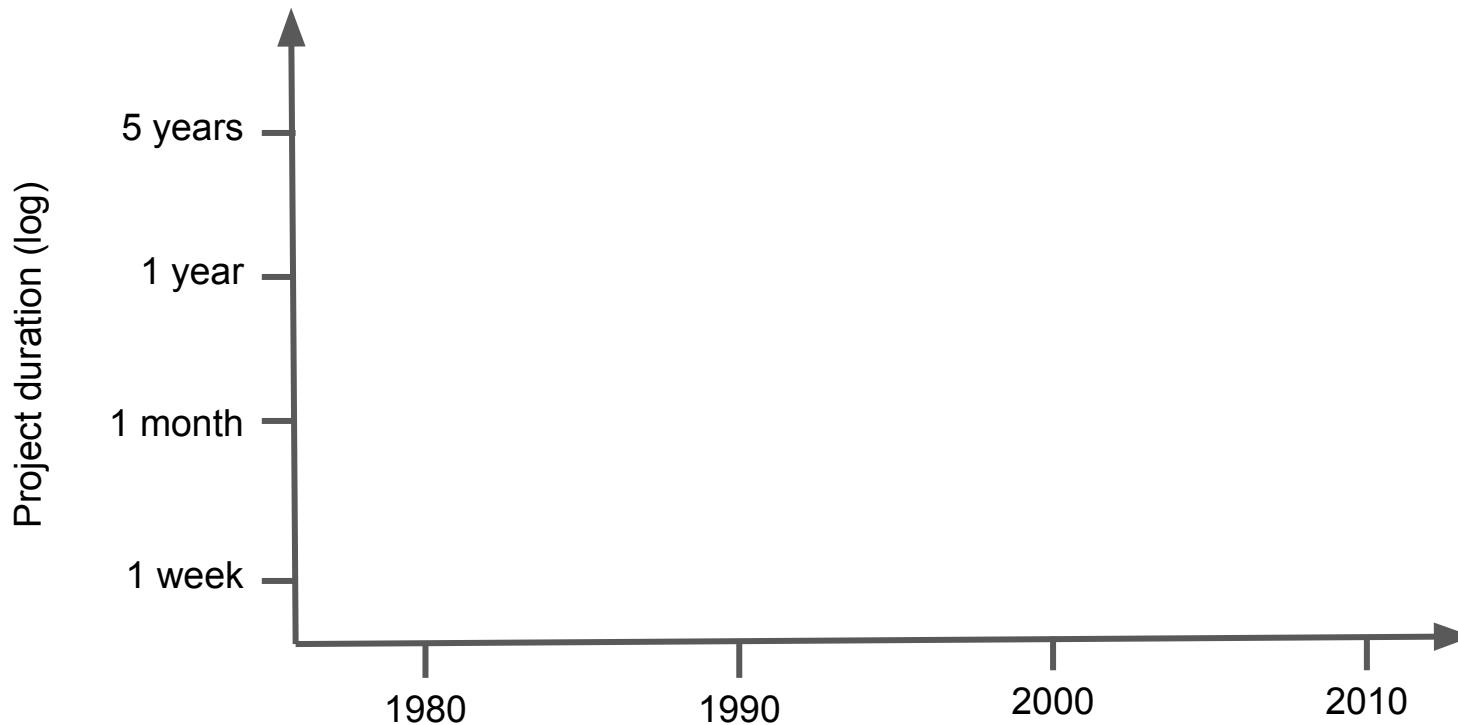
The other story

The other story

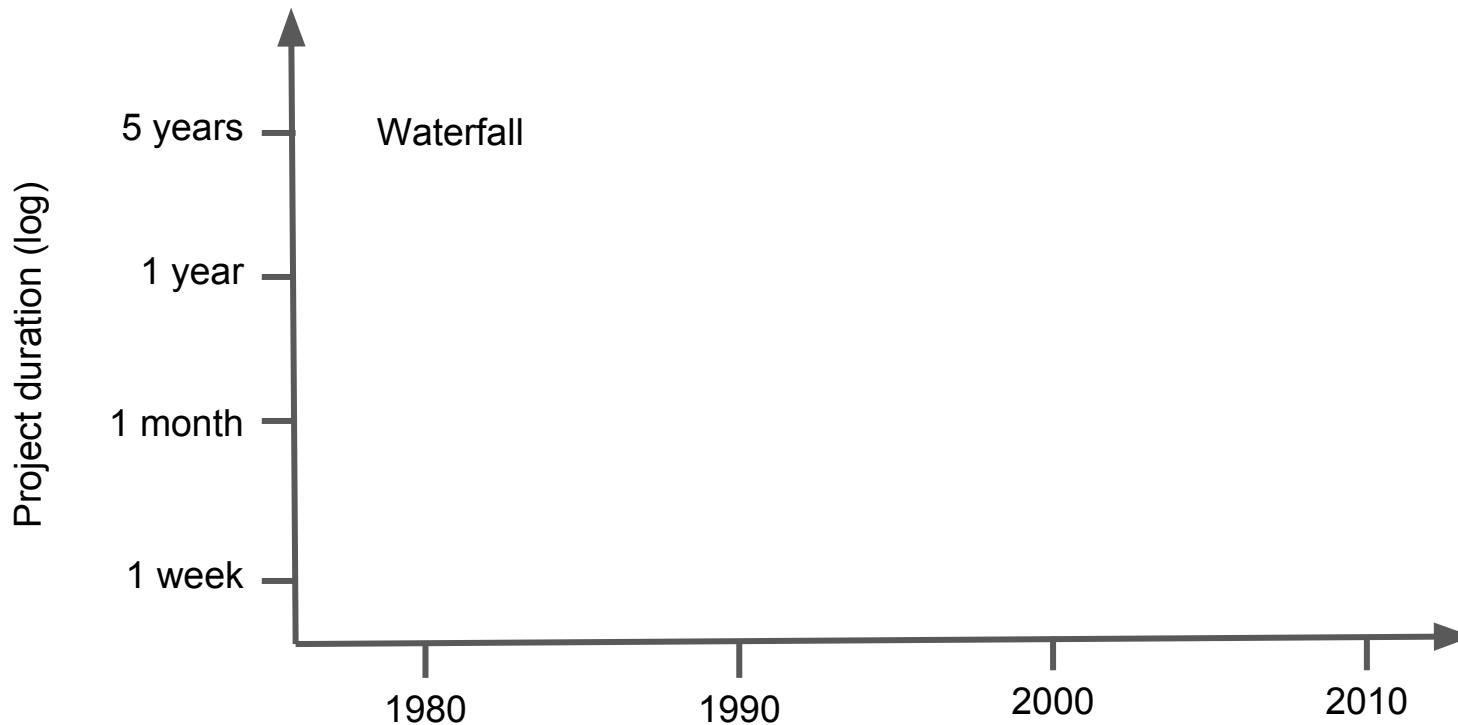
credit to Fred George



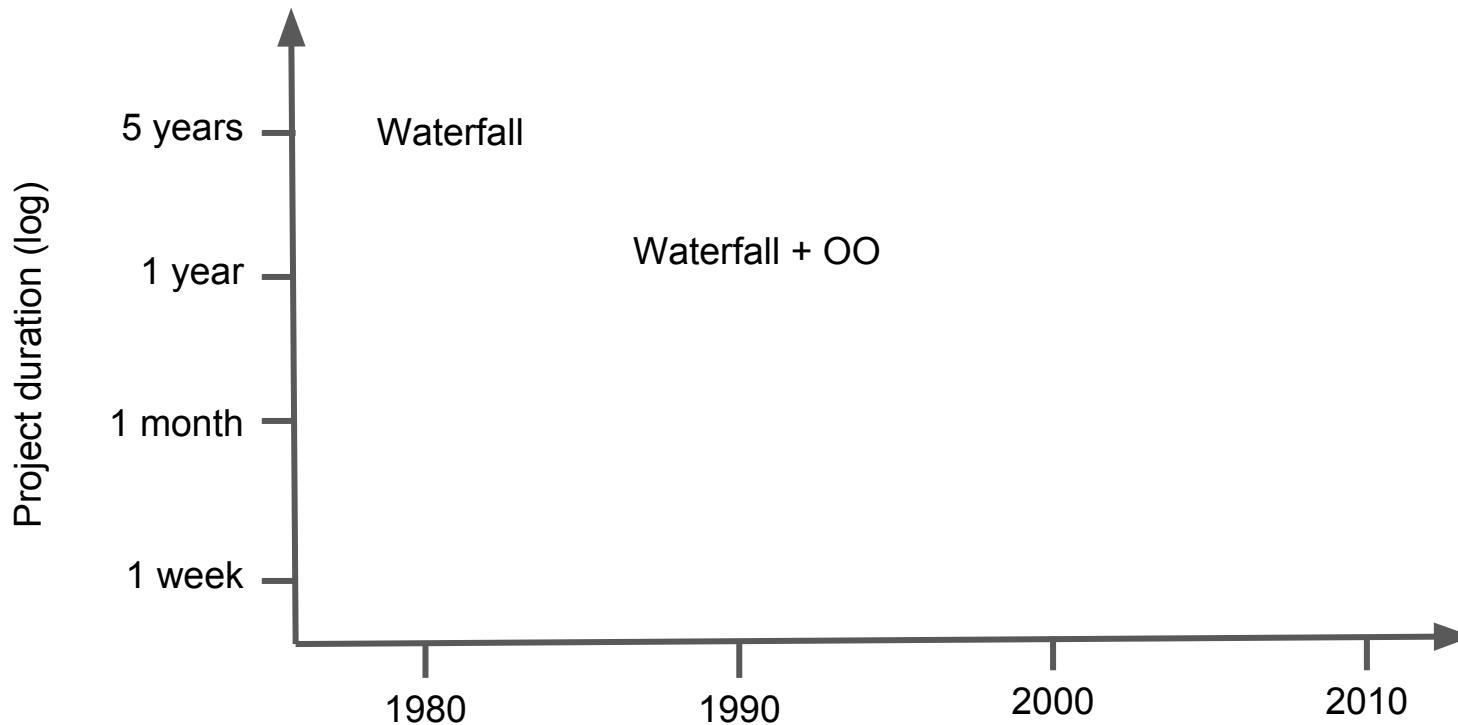
Project delivery cycles



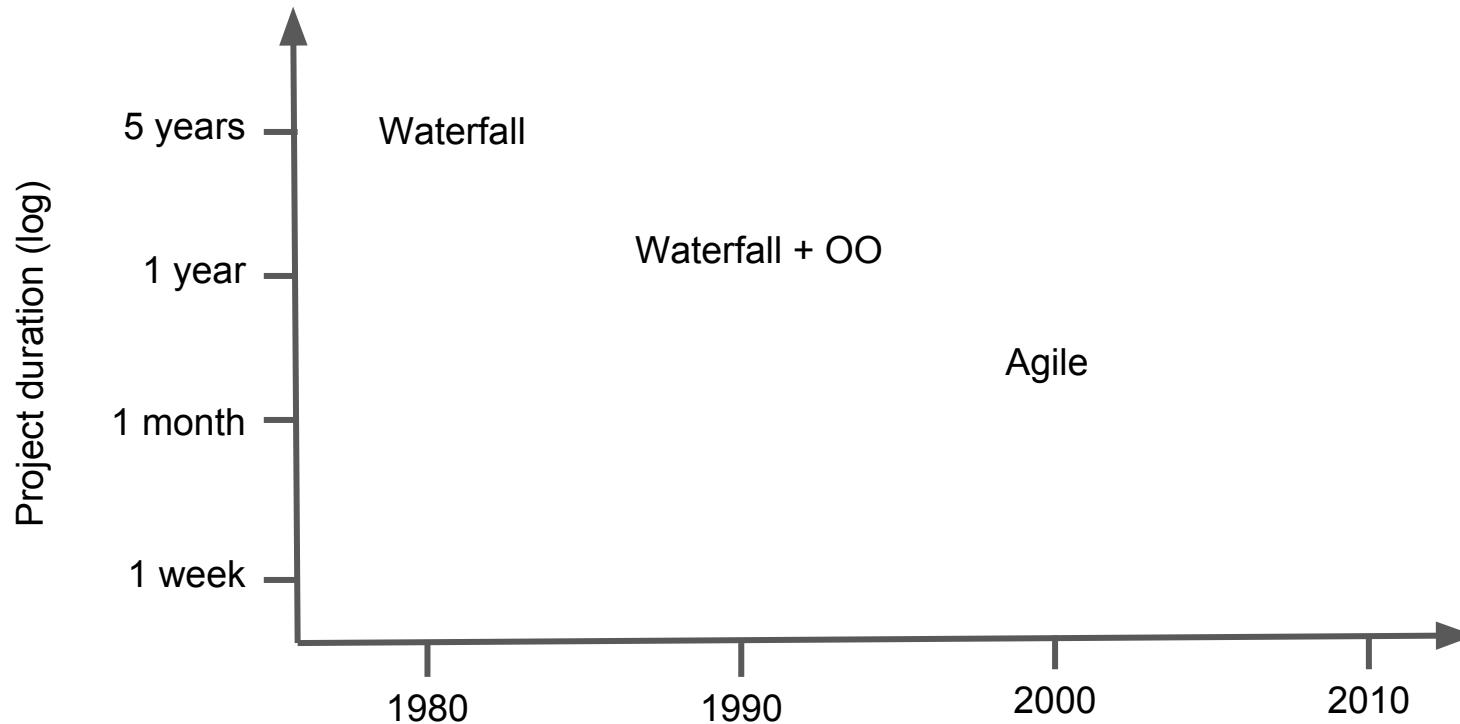
Project delivery cycles



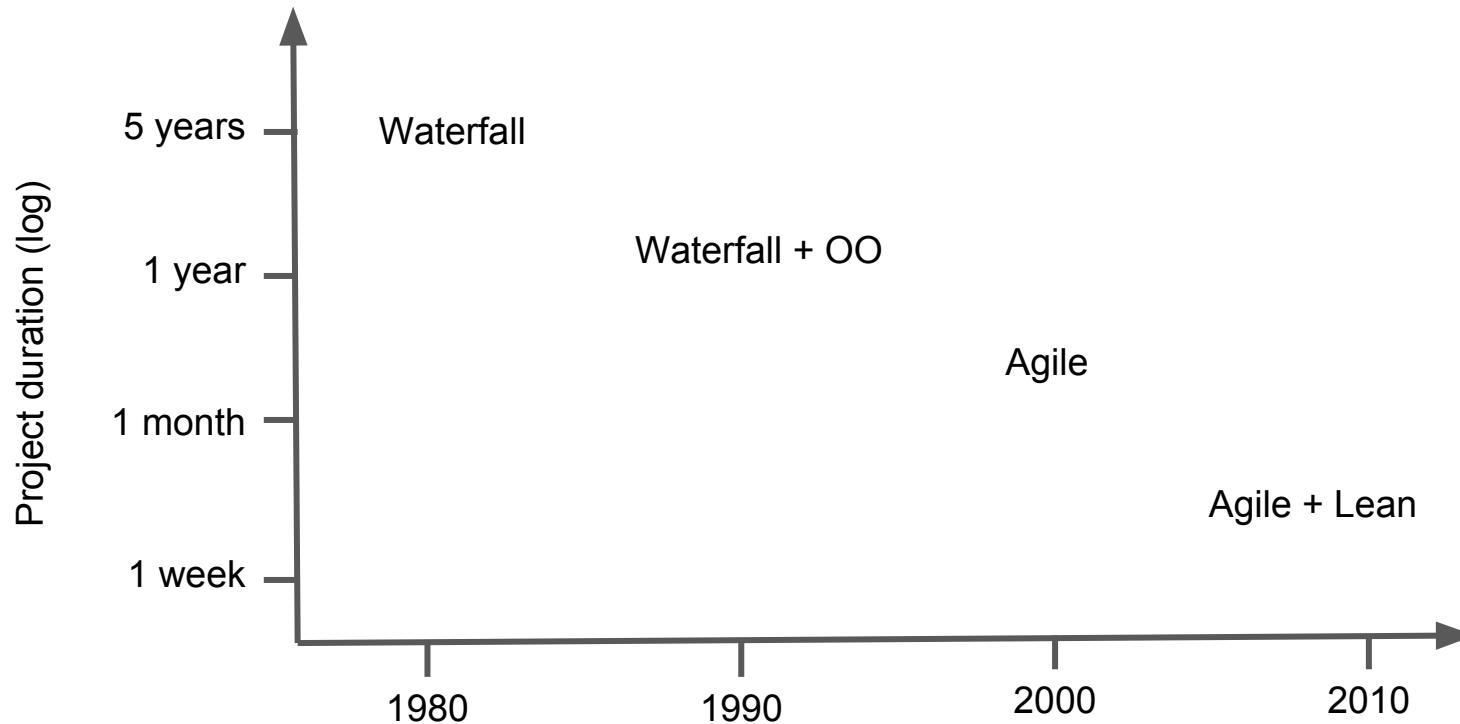
Project delivery cycles



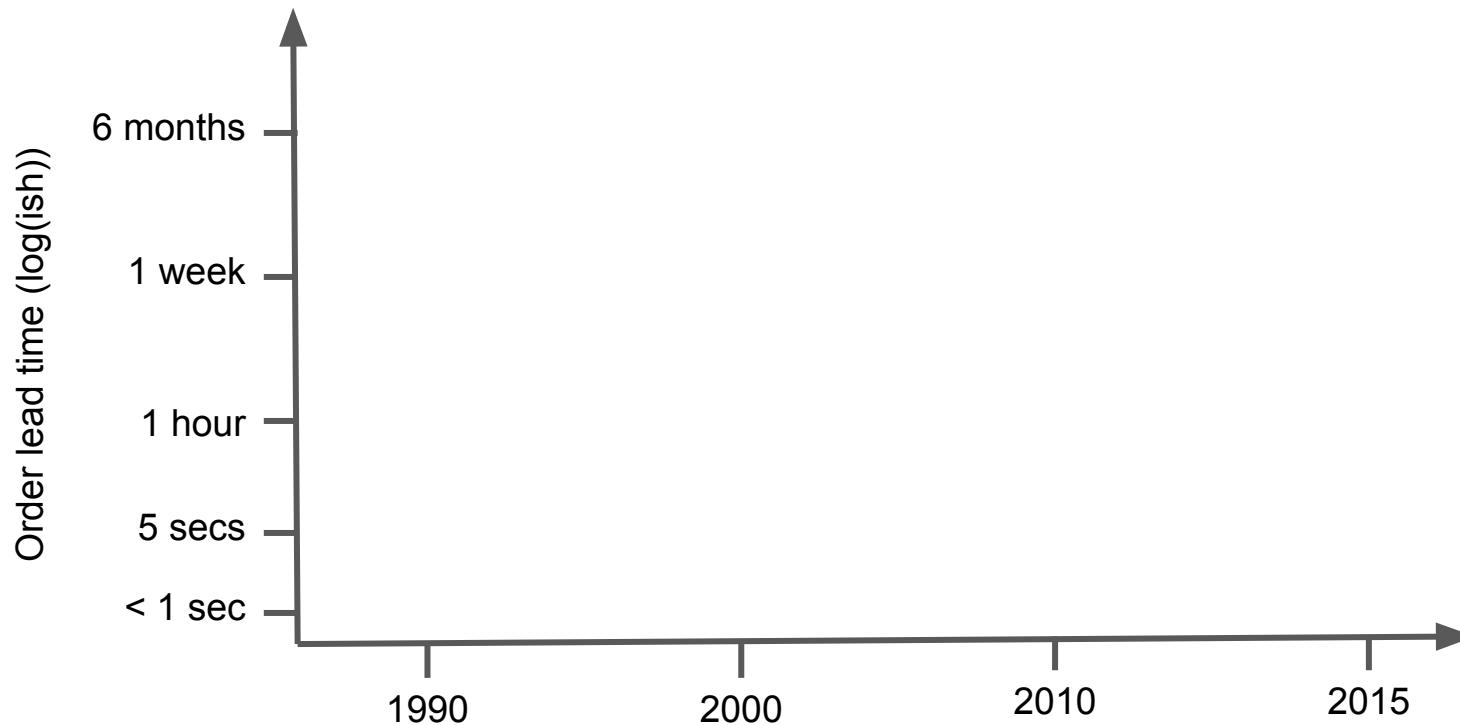
Project delivery cycles



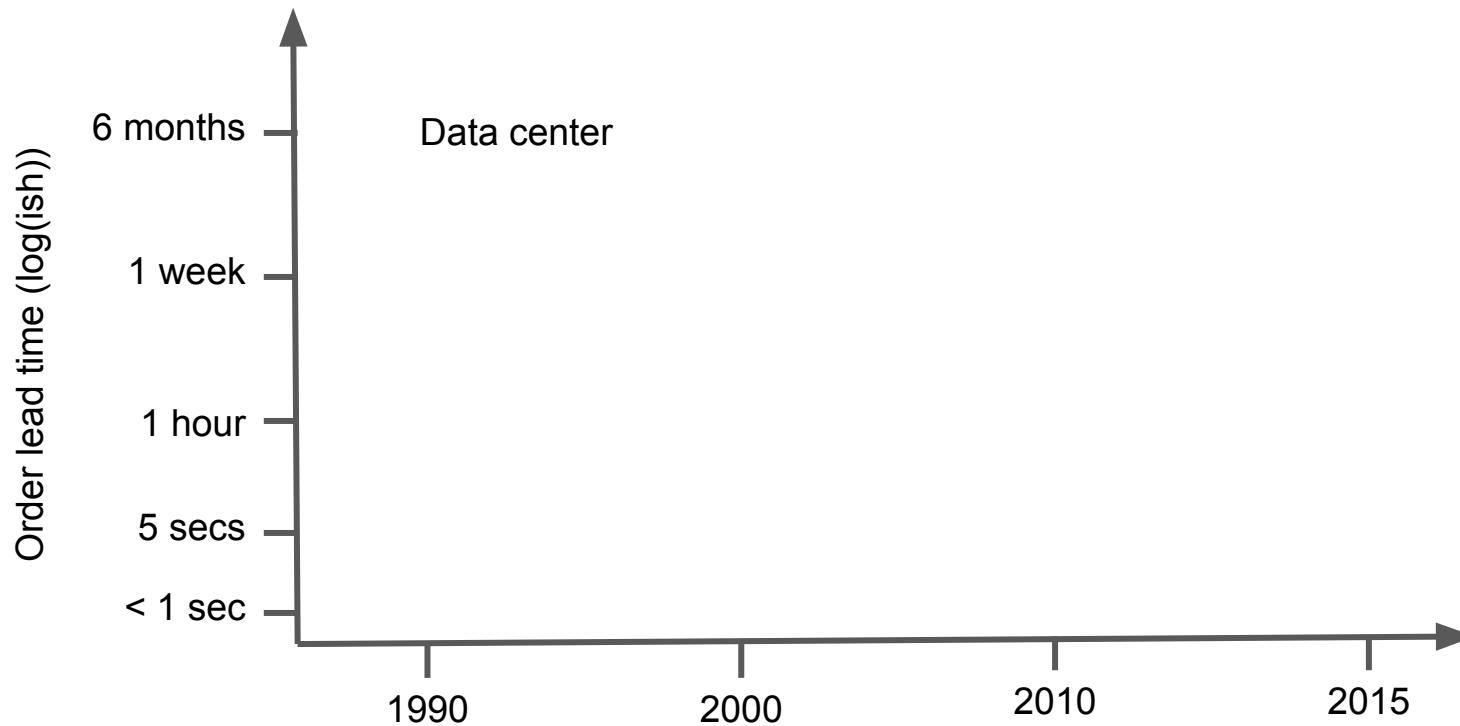
Project delivery cycles



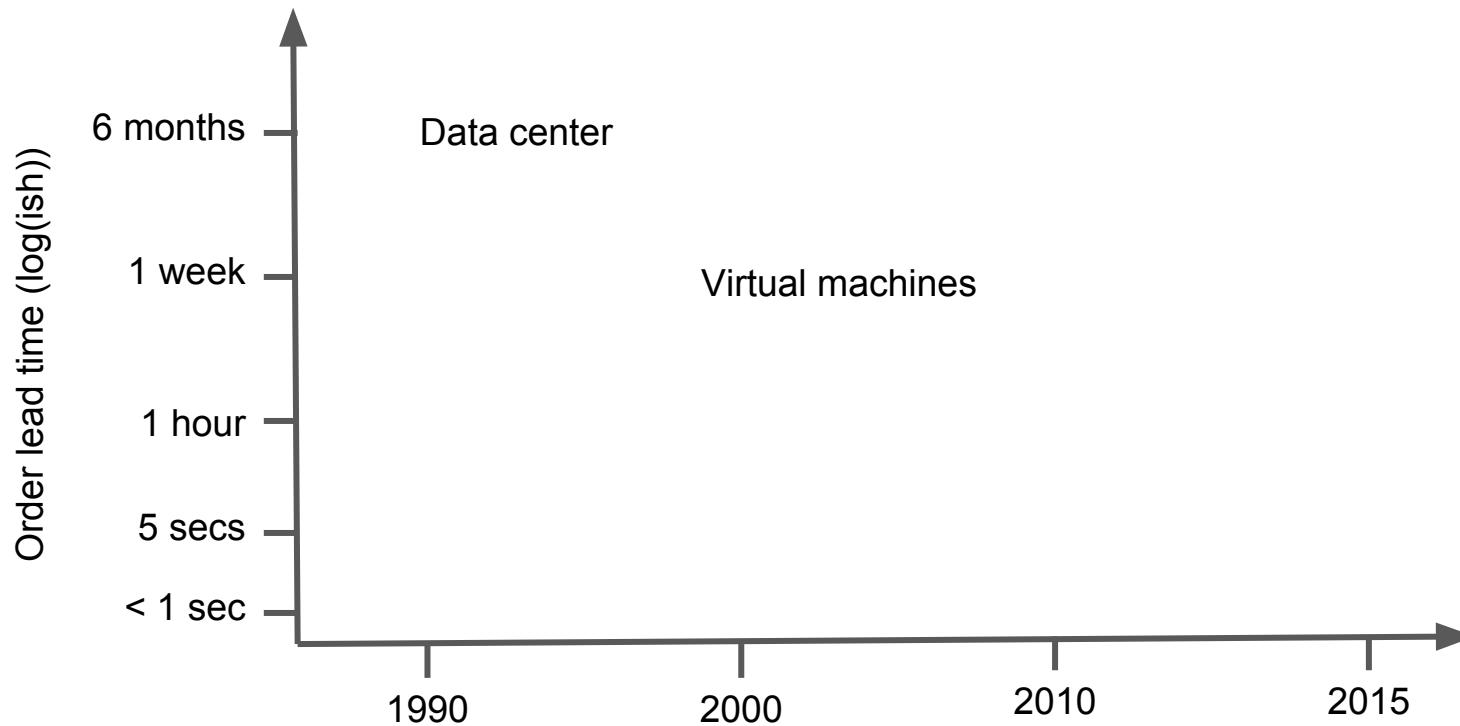
Hardware lead times



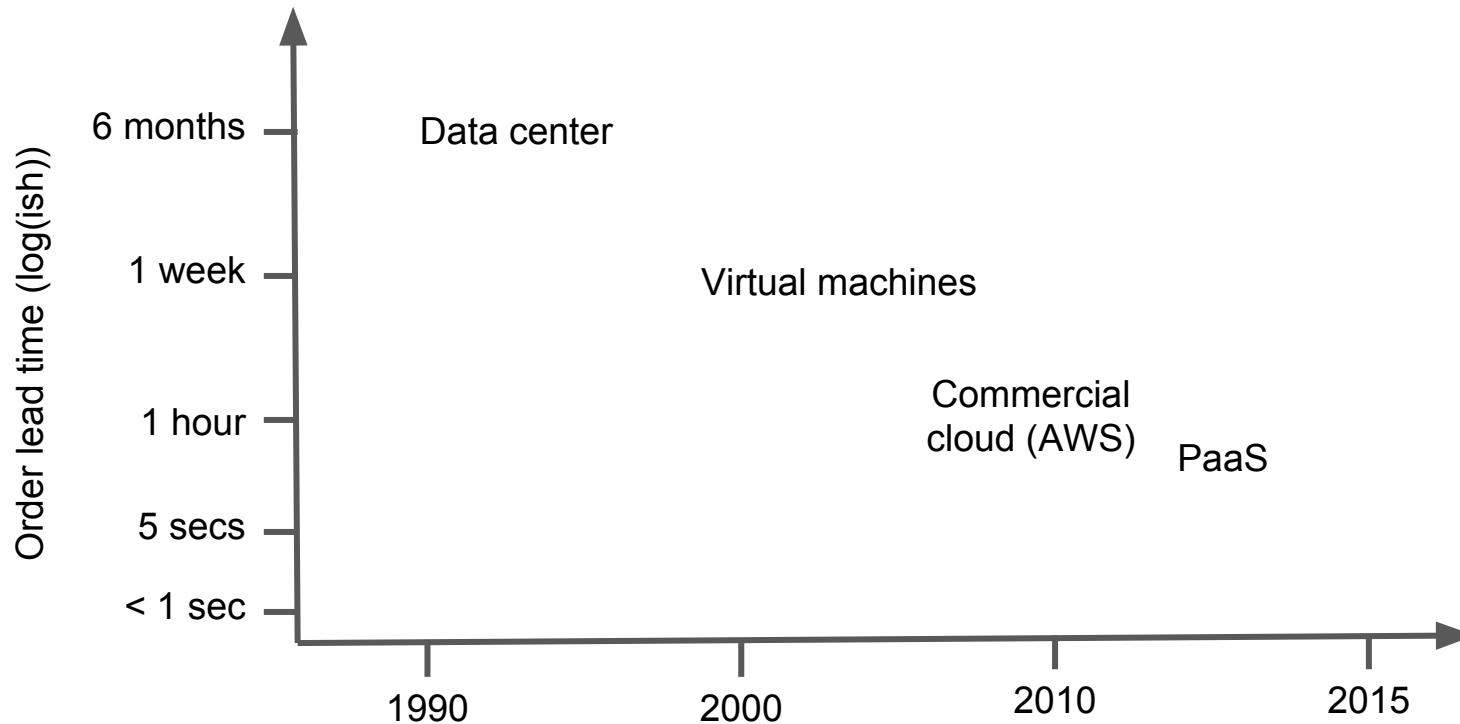
Hardware lead times



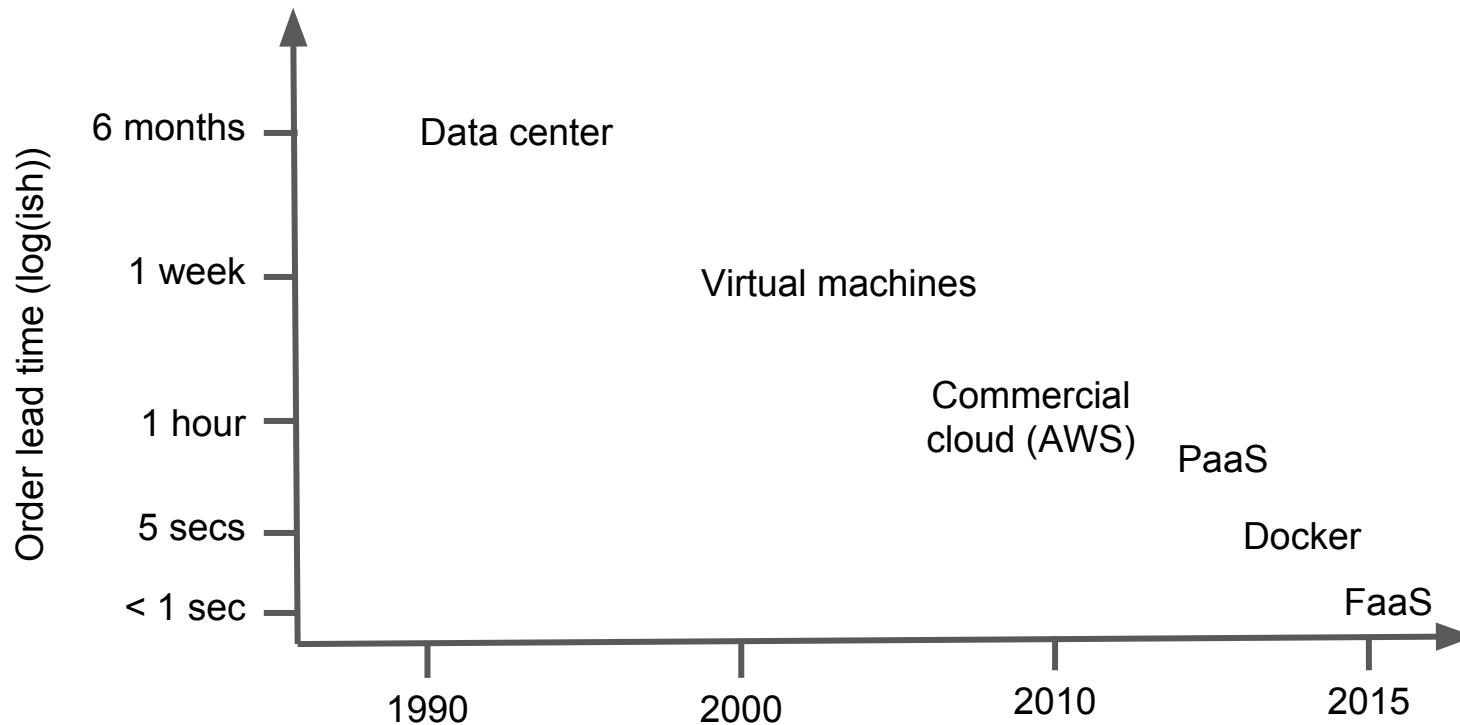
Hardware lead times



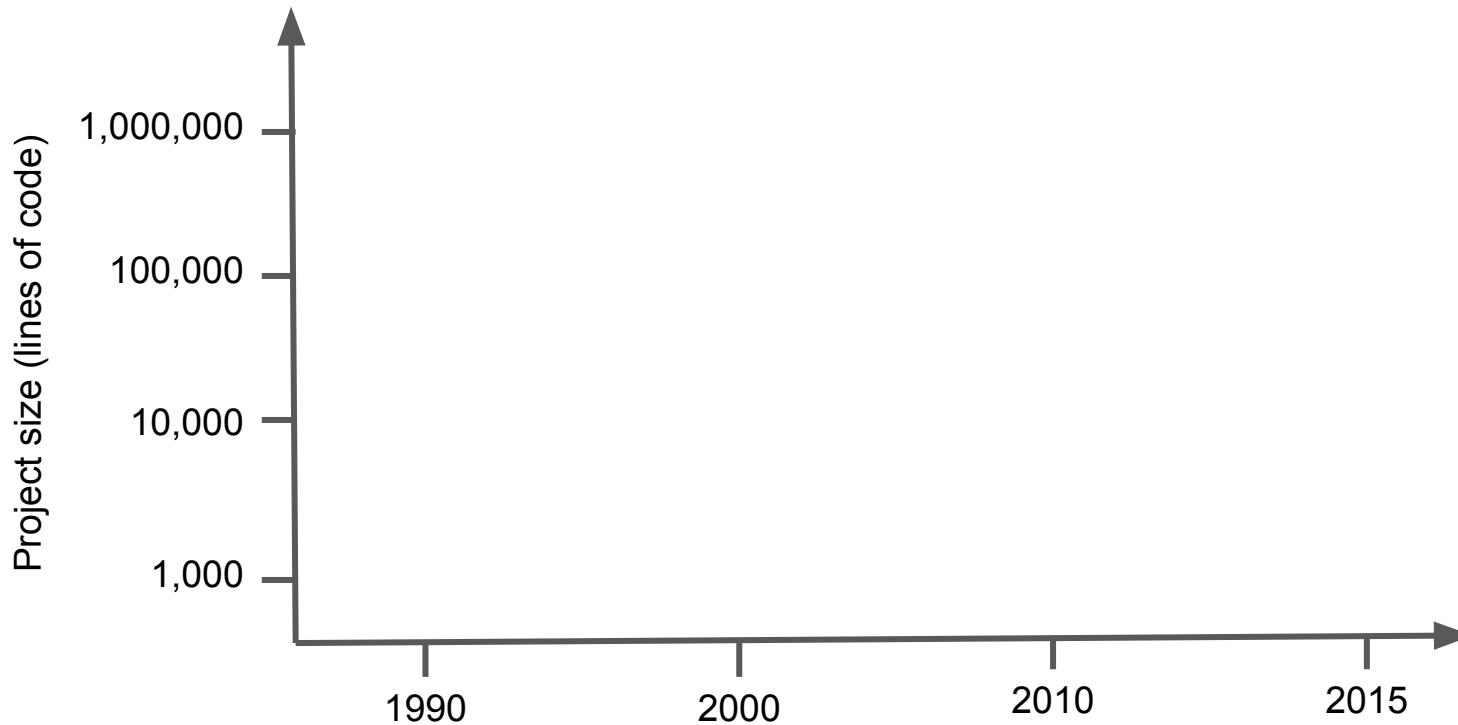
Hardware lead times



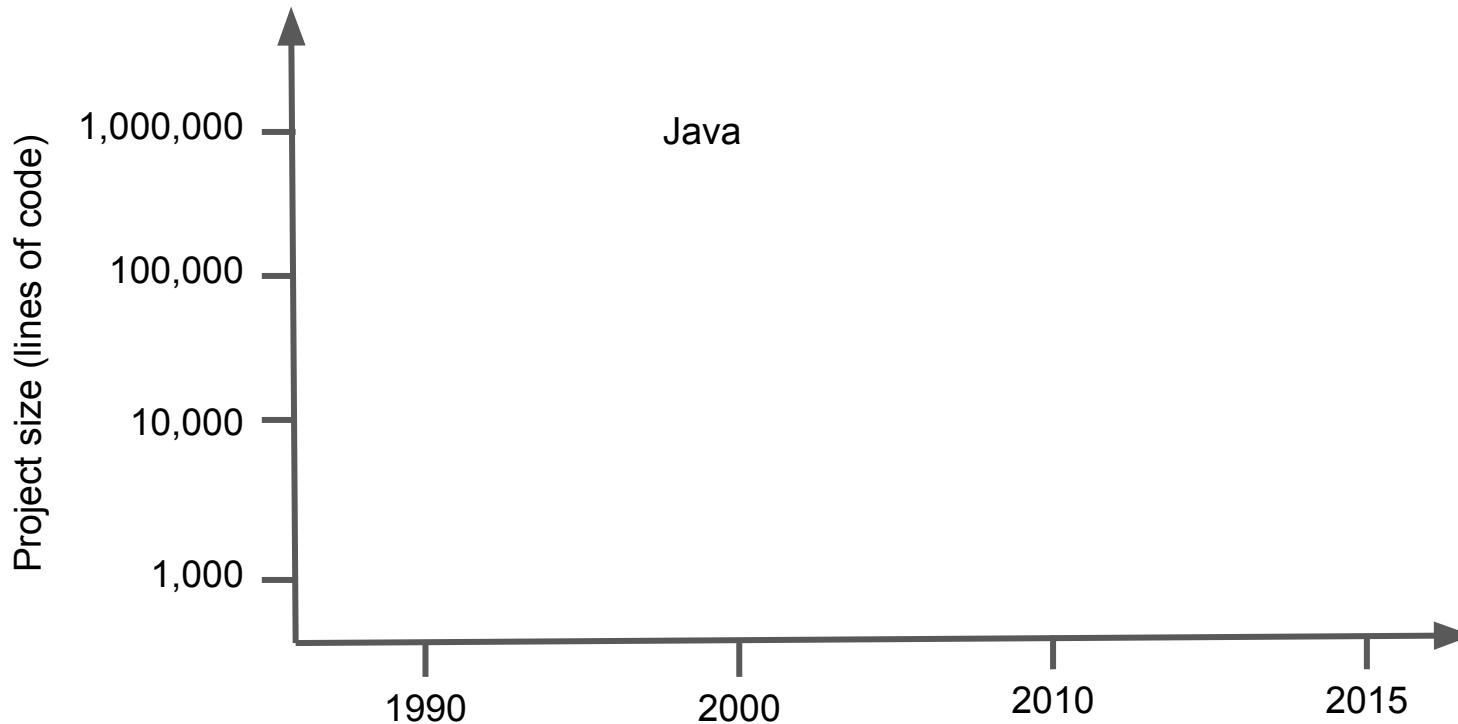
Hardware lead times



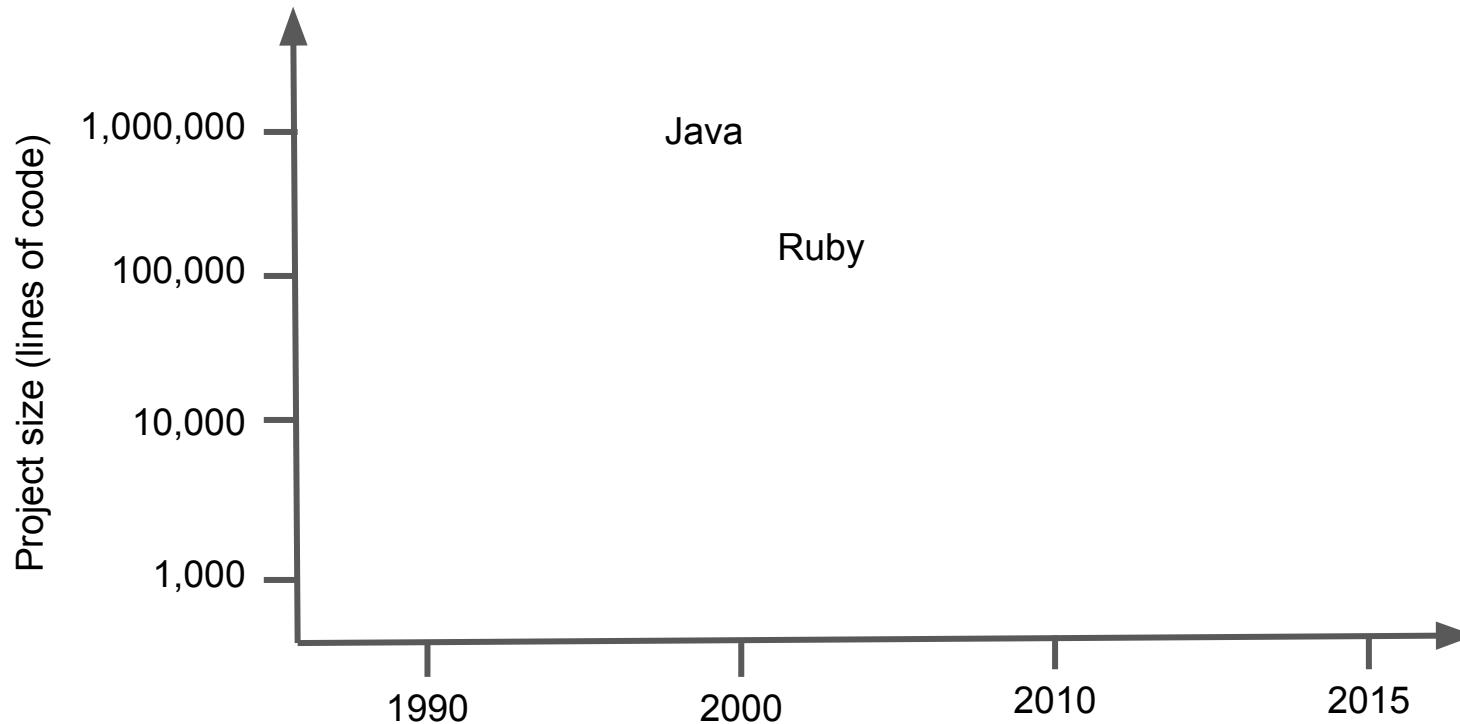
Project size



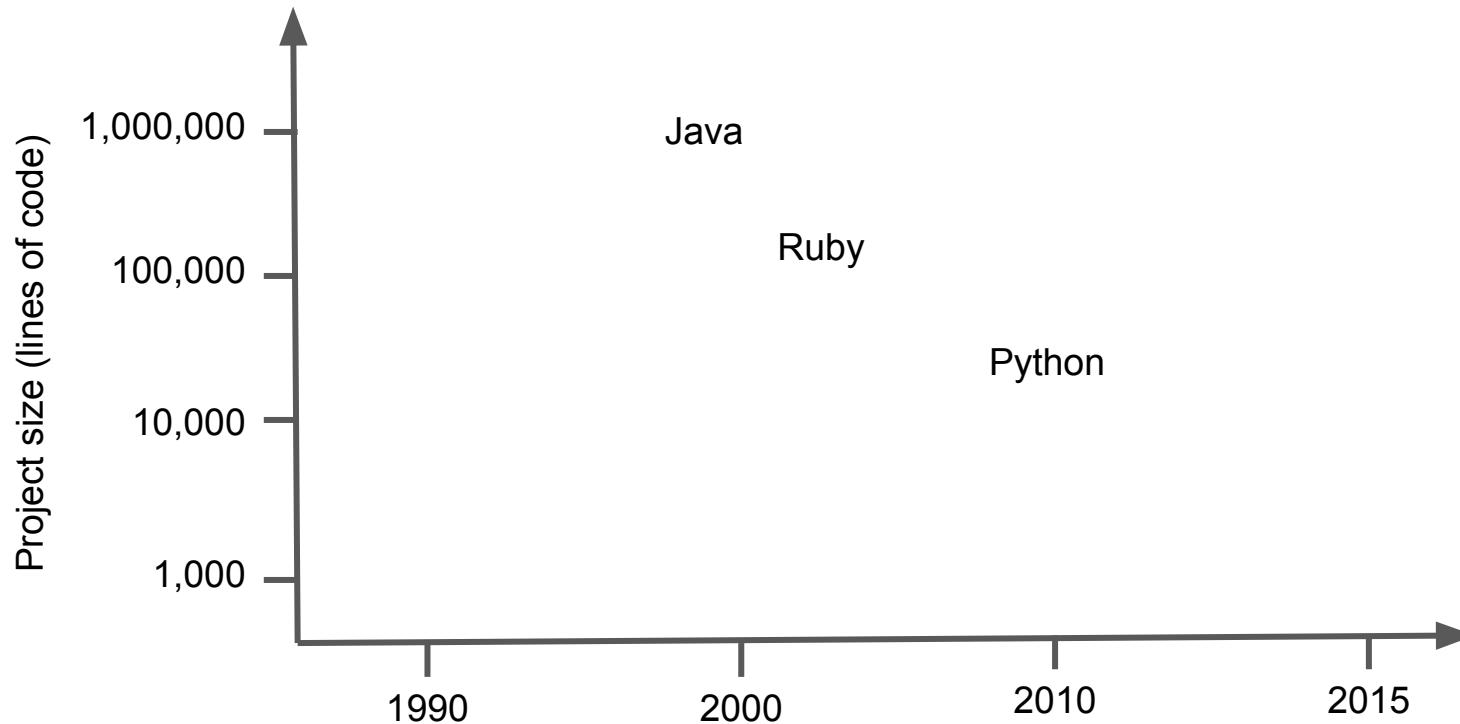
Project size



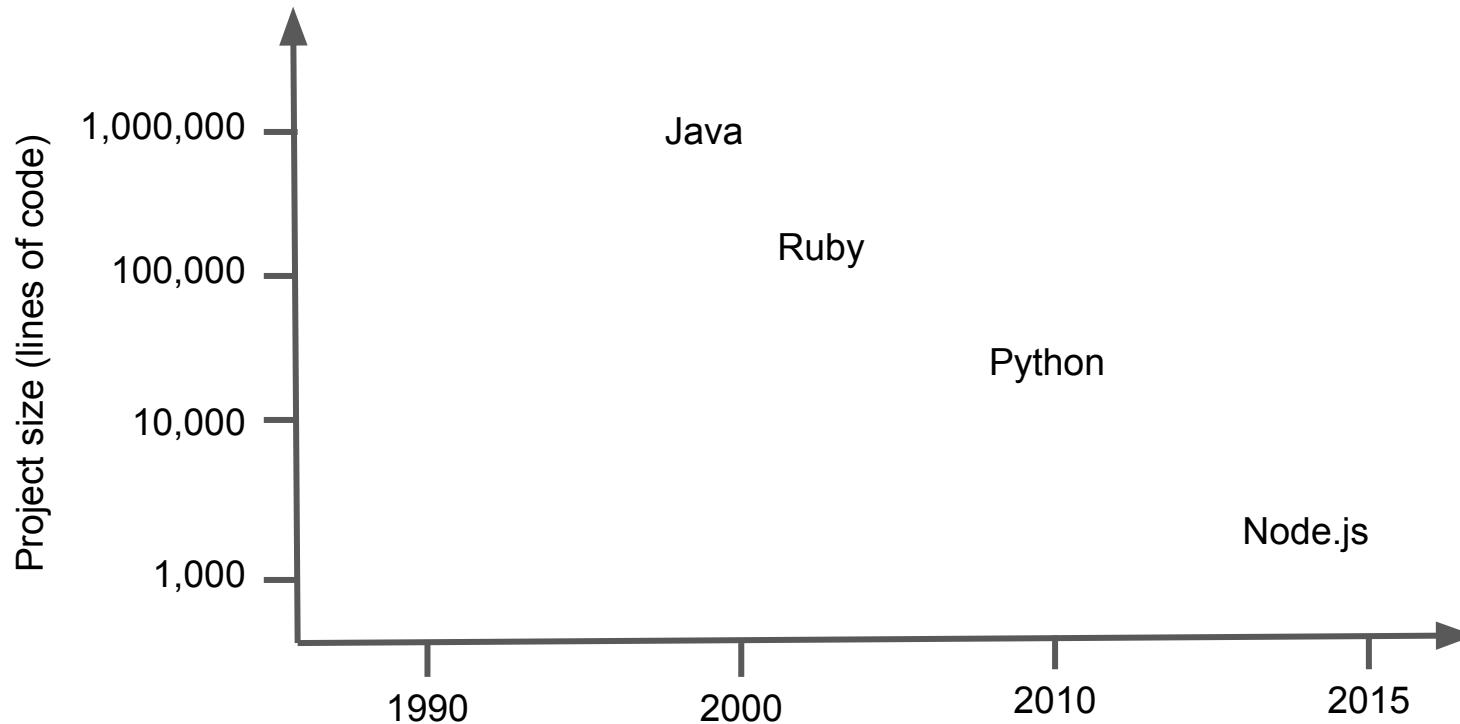
Project size



Project size



Project size



Projects
are being
delivered
faster

Projects
are being
delivered
faster

Hardware is
quicker to
provision

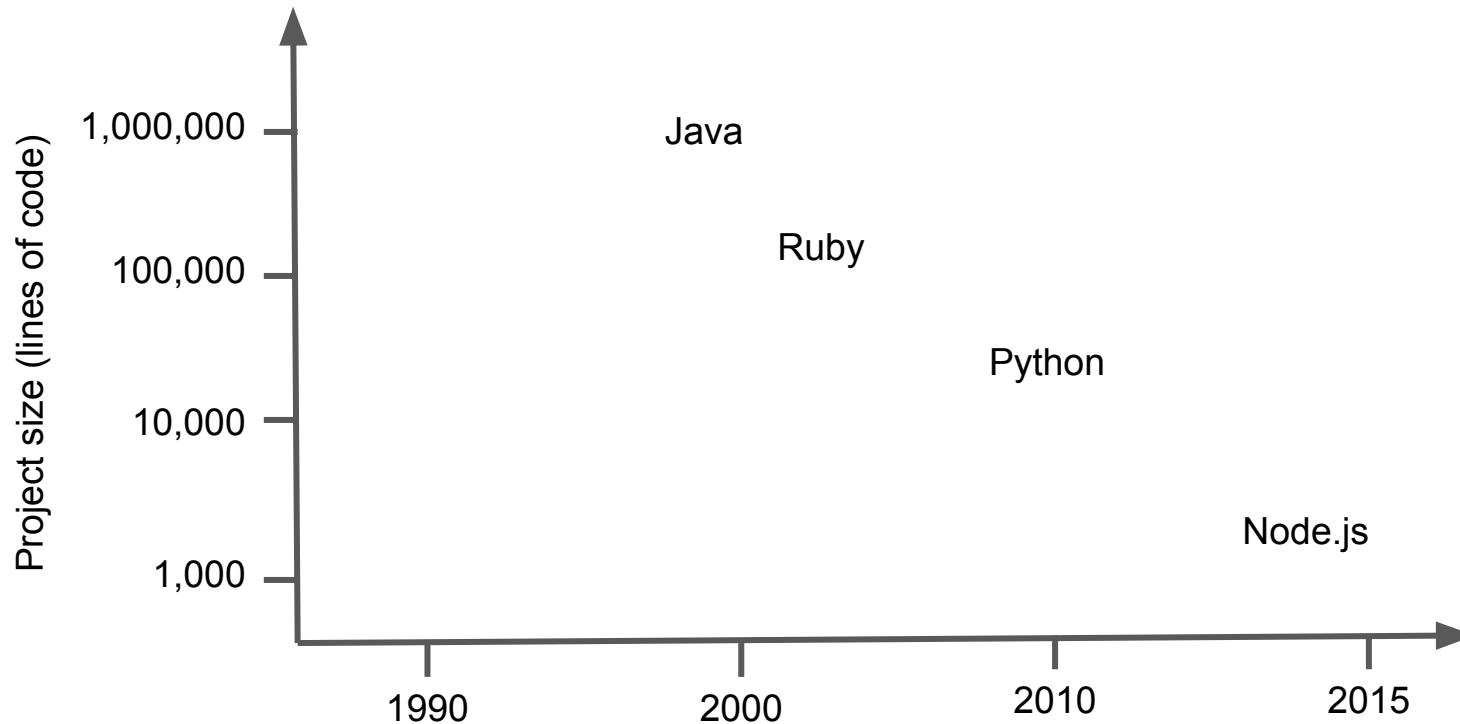
Projects
are being
delivered
faster

Hardware is
quicker to
provision

Projects
are getting
smaller

Microservices are an emergent architecture

Project size



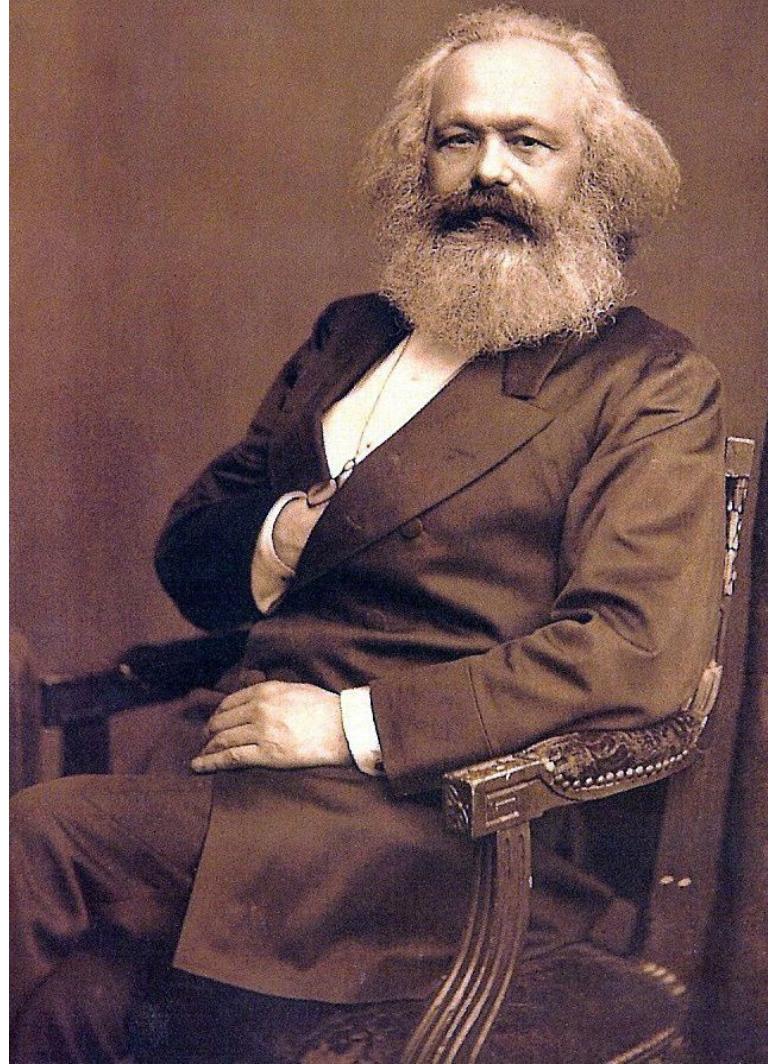
Onboarding the new dev...



credit: <https://twitter.com/moonpolysoft/status/781868129269919744>

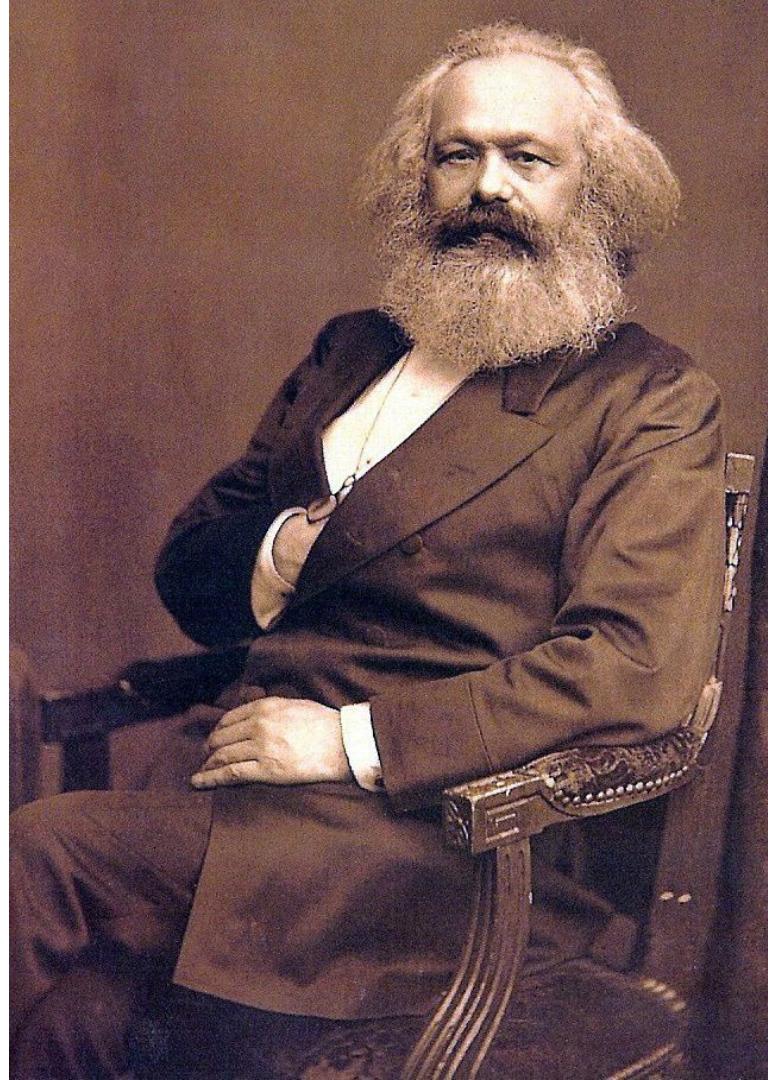
Story time

Marx on Microservices



Marx on Microservices

Economic and Philosophic Manuscripts of 1844

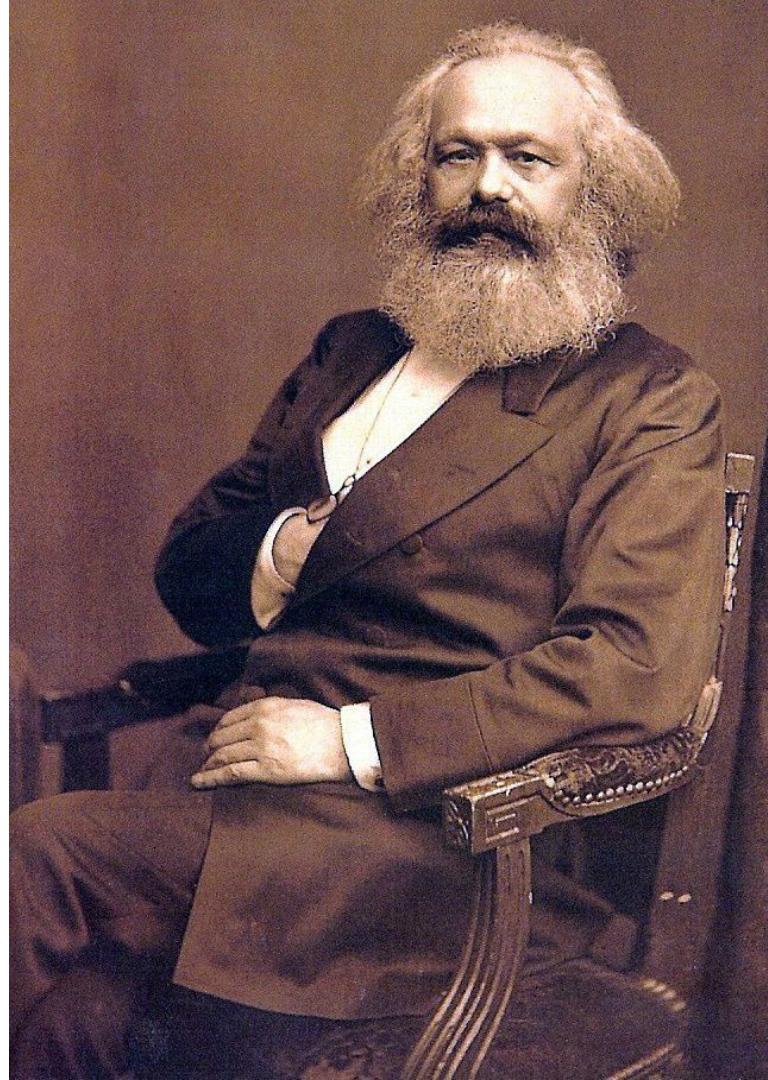


Marx on Microservices

Economic and Philosophic Manuscripts of 1844

Gattungswesen (species-being)

Workers feeling a connection to their work



Marx on Microservices

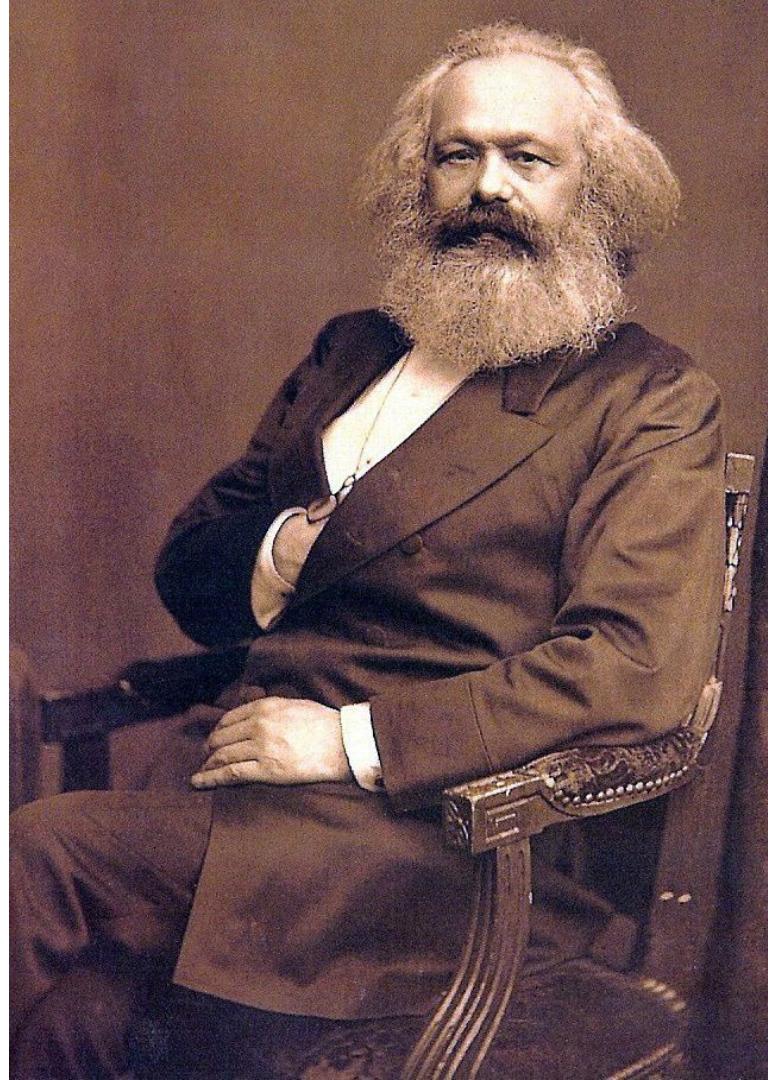
Economic and Philosophic Manuscripts of 1844

Gattungswesen (species-being)

Workers feeling a connection to their work

Entfremdung (alienation)

Workers feel estranged from their work



Products not Projects

“Any piece of software reflects the organizational
structure that produced it”

Conway's Law

Inverse Conway Maneuver?

What sort of organisation do you want to work in?

The other, other story

SOA

Service Oriented Architecture

Service Oriented Architecture

Boundaries are explicit

Service Oriented Architecture

Boundaries are explicit

Services are autonomous

Service Oriented Architecture

Boundaries are explicit

Services are autonomous

Services share schema and contract, not class

Service Oriented Architecture

Boundaries are explicit

Services are autonomous

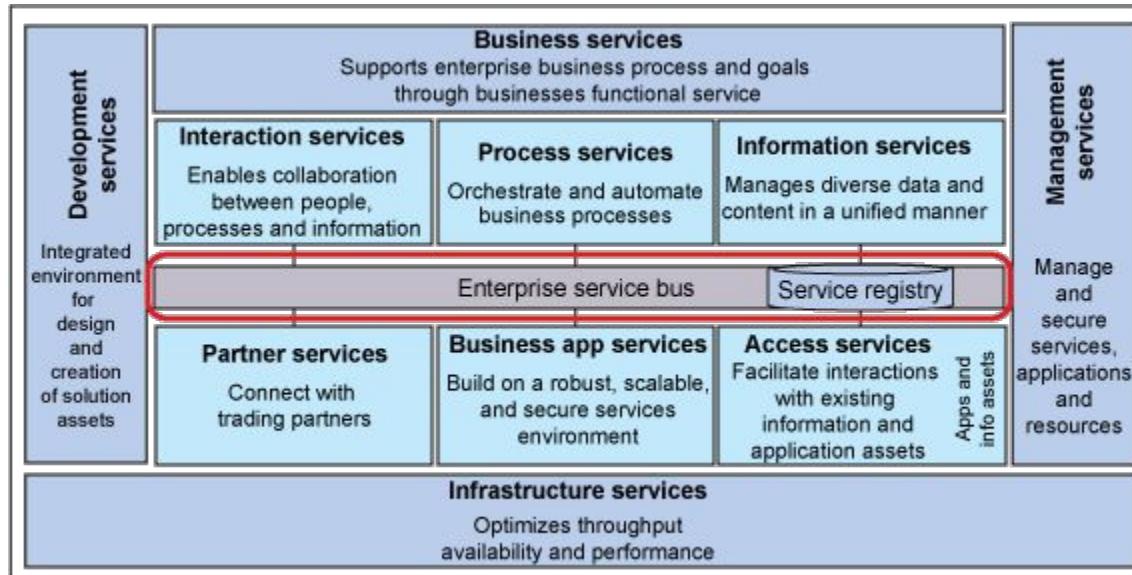
Services share schema and contract, not class

Service compatibility is based on policy

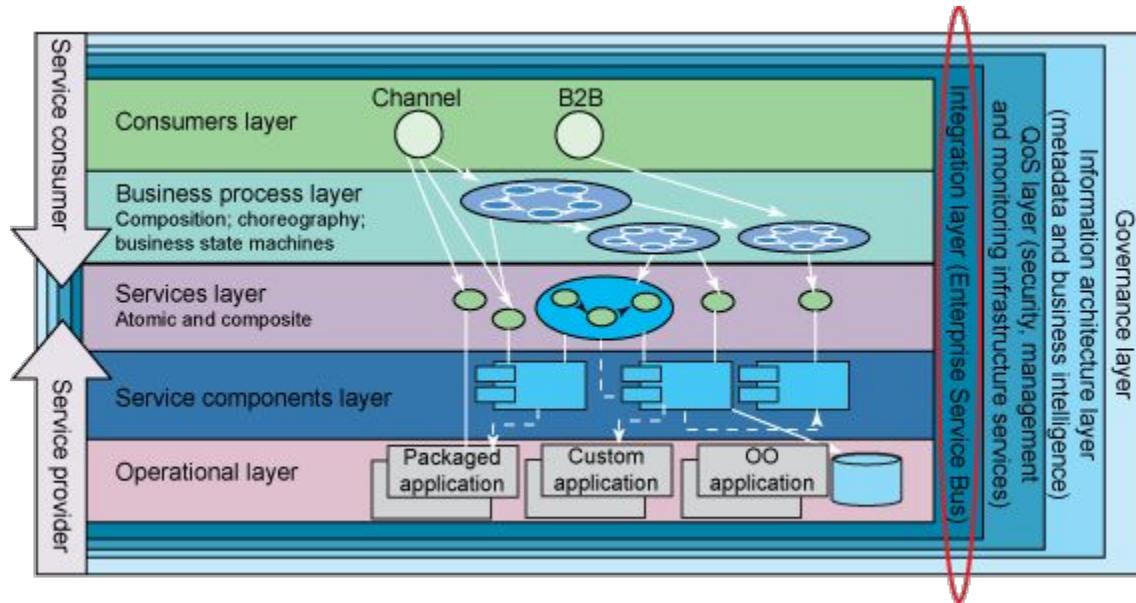
IBM SOA Foundation

IBM SOA Foundation Reference Architecture

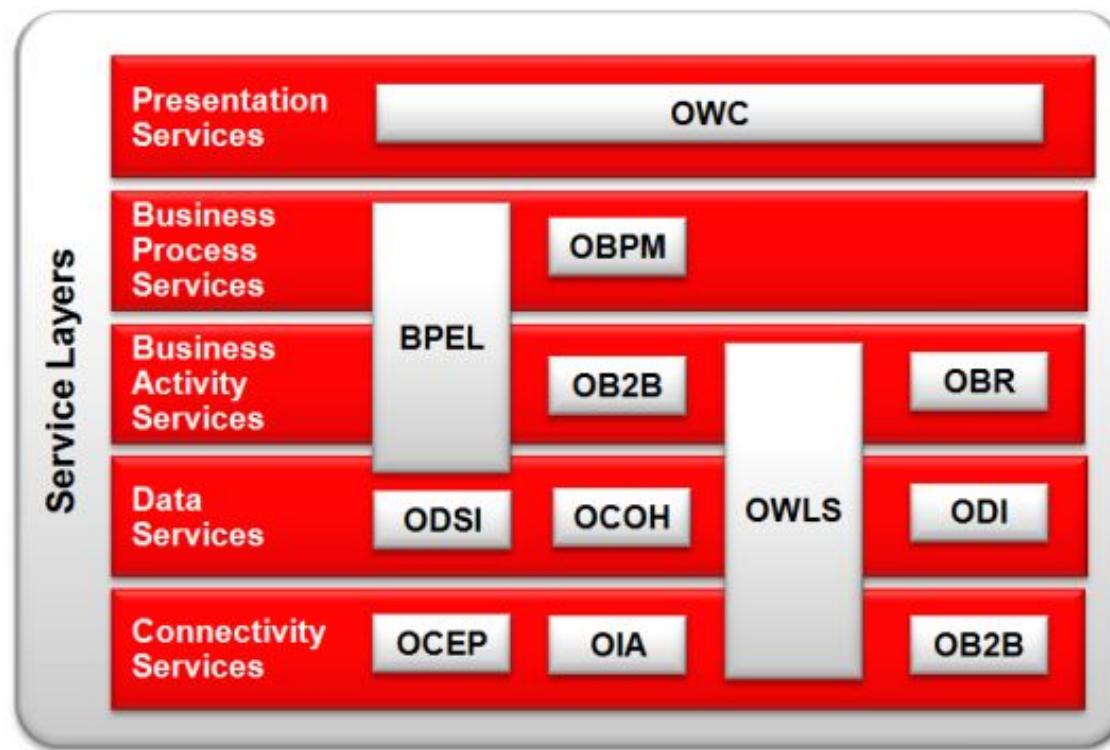
IBM SOA Foundation Reference Architecture logical model



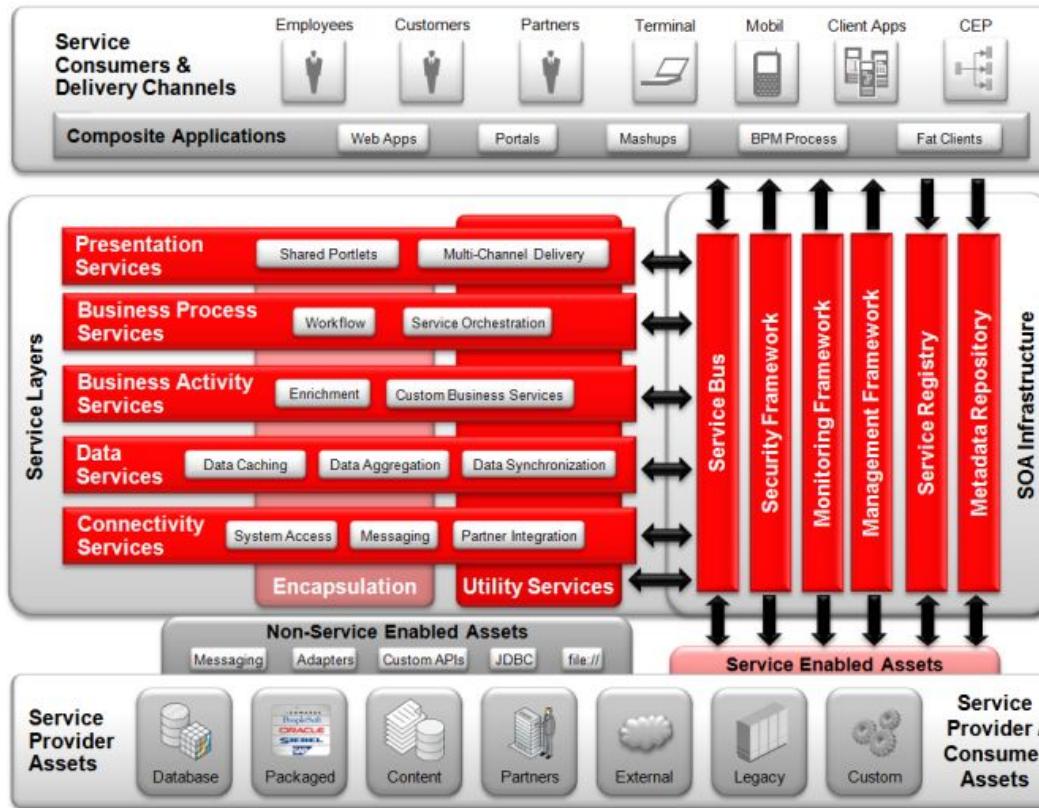
IBM SOA Foundation Reference Architecture solution



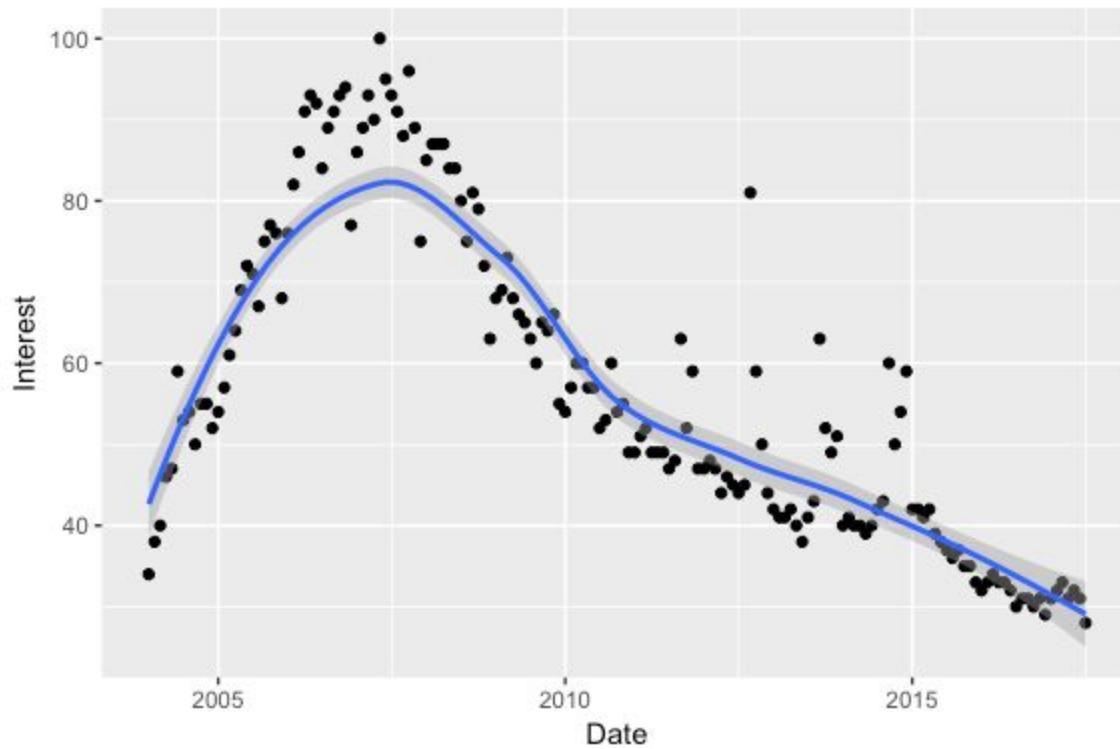
Oracle SOA Foundation Reference Architecture (Product Mapping)



Oracle SOA Foundation Reference Architecture



Google Trends: SOA



“The latest shiny new technology will not make things better... If you want **spectacular gains**, then you need to make a **spectacular commitment to change.**”

Anne Thomas Manes, SOA is Dead; Long Live Services

“... and that’s where we need to concentrate from
this point forward: **Services**”

Anne Thomas Manes, SOA is Dead; Long Live Services

Why you shouldn't do Microservices

CAP Theorem

Dr. Eric Brewer (2000)

CAP Theorem

Consistency

Availability

Partition tolerance

CAP Theorem

Consistency

(all requests return the correct results)

Availability

Partition tolerance

CAP Theorem

Consistency

(all requests return the correct results)

Availability

(all requests complete)

Partition tolerance

CAP Theorem

Consistency

(all requests return the correct results)

Availability

(all requests complete)

Partition tolerance

(the network may fail)

CAP Theorem

Consistency

(all requests return the correct results)

Availability

(all requests complete)

Partition tolerance

(the network may fail)

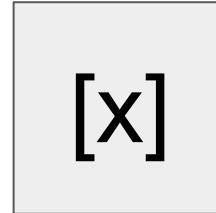
Pick 2

CAP Theorem - Example 1

User 1



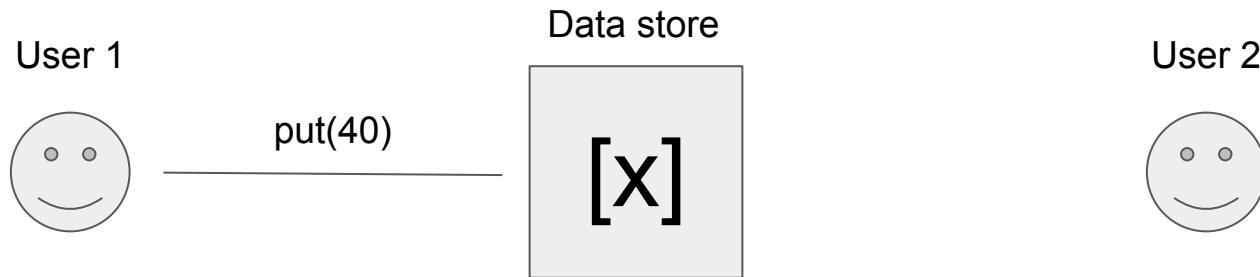
Data store



User 2



CAP Theorem - Example 1



CAP Theorem - Example 1

User 1



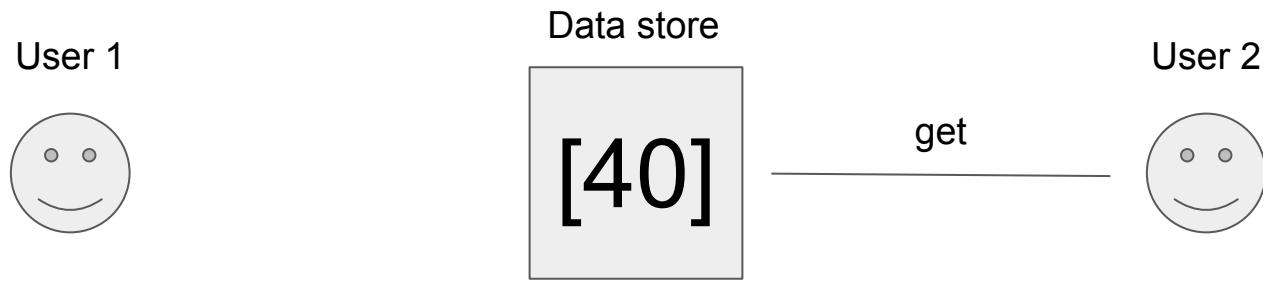
Data store

[40]

User 2



CAP Theorem - Example 1



CAP Theorem - Example 1

User 1



Data store



User 2



40

CAP Theorem - Example 1



- ✓ Consistent (correct)
- ✓ Available (answered)
- ✗ Partition tolerant (no network partitions)

CAP Theorem - Example 1



- ✓ Consistent (correct)
- ✓ Available (answered)
- ✗ Partition tolerant (no network partitions)

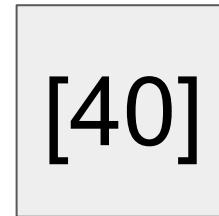
[PROBLEM] No real world system looks like this

CAP Theorem - Example 2

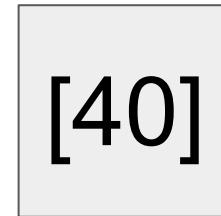
User 1



Data store (US)



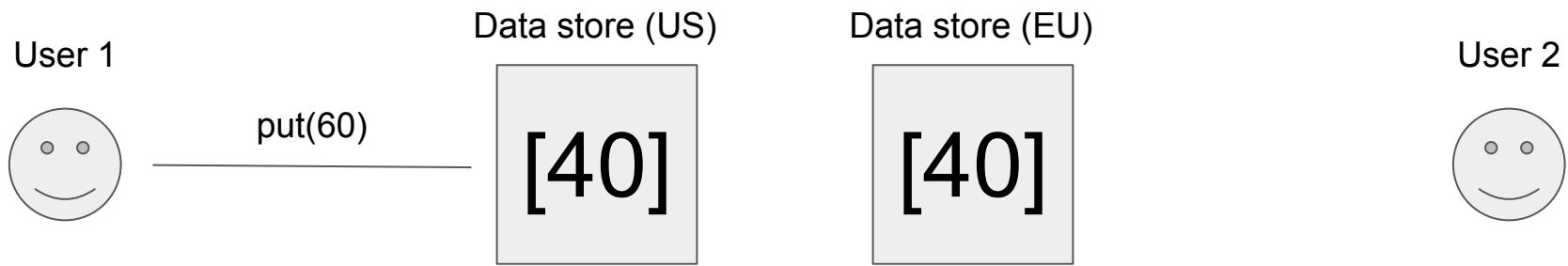
Data store (EU)



User 2



CAP Theorem - Example 2

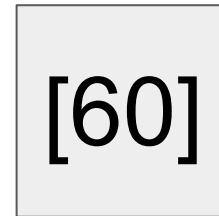


CAP Theorem - Example 2

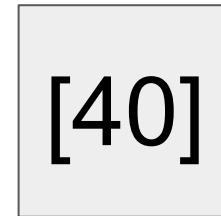
User 1



Data store (US)



Data store (EU)



User 2

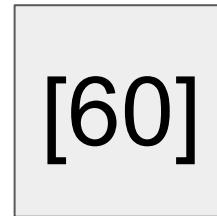


CAP Theorem - Example 2

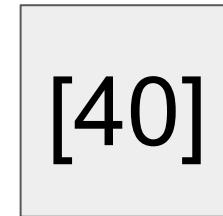
User 1



Data store (US)



Data store (EU)



User 2

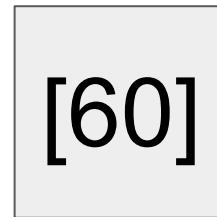


CAP Theorem - Example 2

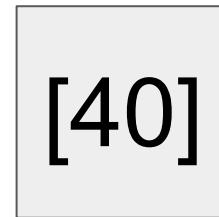


User 1

Data store (US)



Data store (EU)



get



User 2

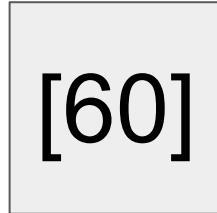
What now?

CAP Theorem - Example 2

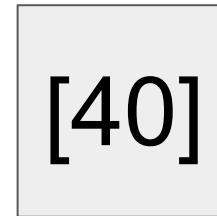
User 1



Data store (US)

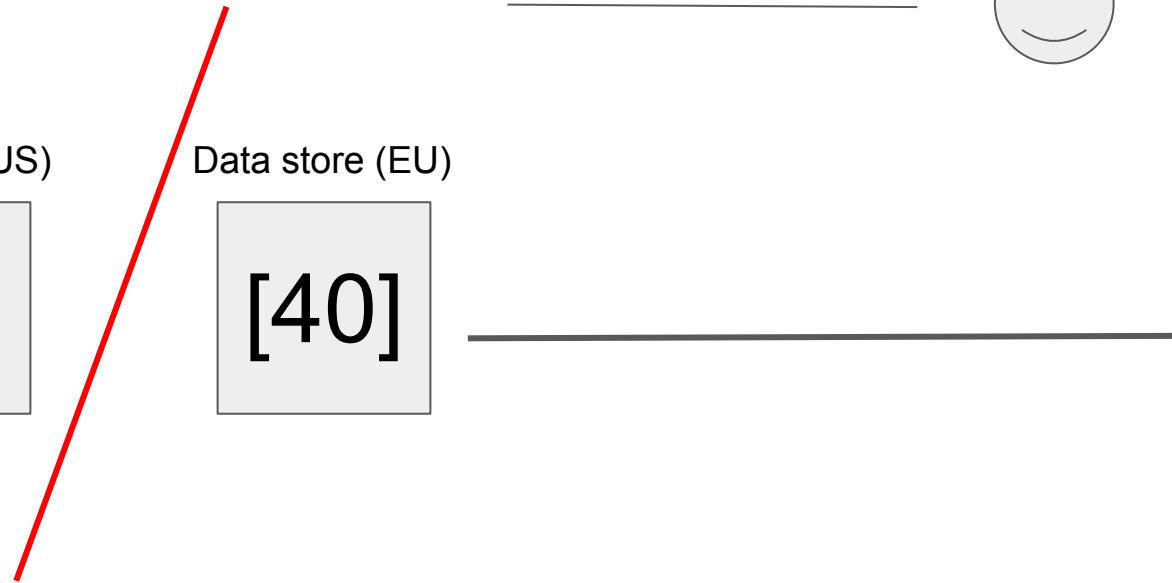


Data store (EU)



get → 40

User 2

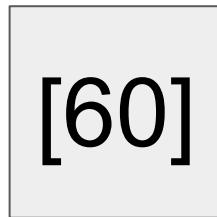


CAP Theorem - Example 2

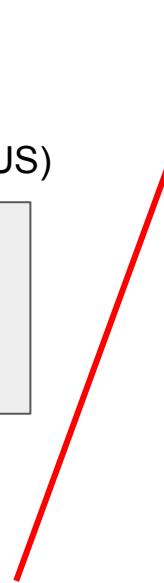
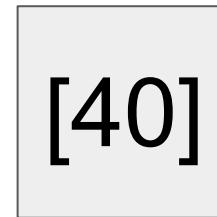
User 1



Data store (US)



Data store (EU)



get → 40

User 2



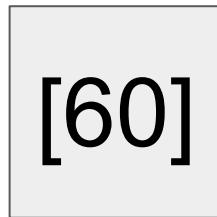
- ✗ Consistent (incorrect)
- ✓ Available (answered)
- ✓ Partition tolerant

CAP Theorem - Example 2

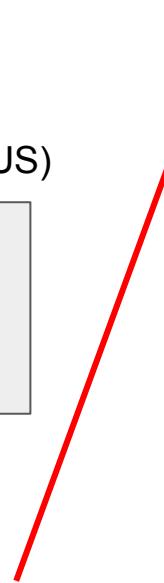
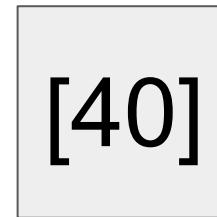
User 1



Data store (US)



Data store (EU)



get → 40

User 2



- ✗ Consistent (incorrect)
- ✓ Available (answered)
- ✓ Partition tolerant

User 2

wait

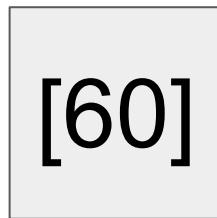


CAP Theorem - Example 2

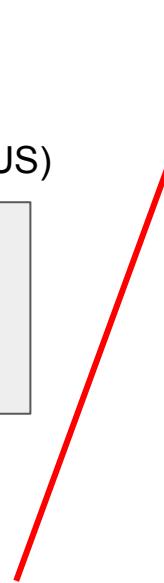
User 1



Data store (US)



Data store (EU)



get → 40

User 2



- ✗ Consistent (incorrect)
- ✓ Available (answered)
- ✓ Partition tolerant

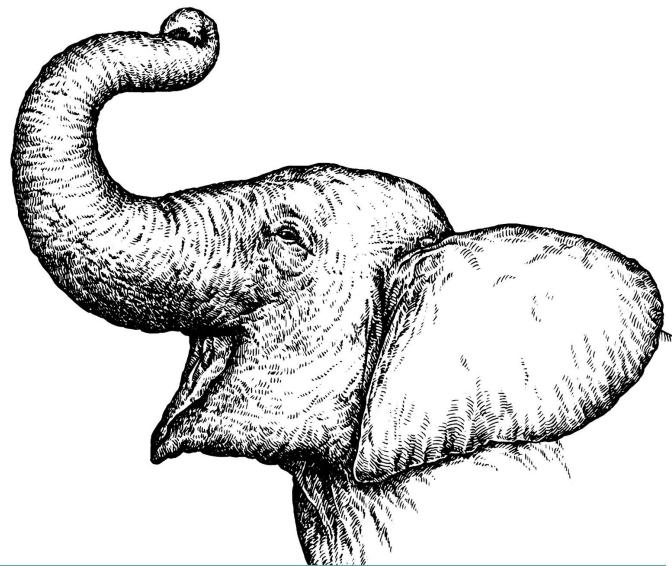
User 2

wait



- ✓ Consistent (correct)
- ✗ Available (not answered)
- ✓ Partition tolerant

The answer to every programming question ever conceived



It Depends

The Definitive Guide

O RLY?

@ThePracticalDev

Profile picture?

Profile picture?

Bank balance?

Profile picture?

ATM withdrawal?

Bank balance?

Profile picture?

Football scores?

ATM withdrawal?

Bank balance?

Profile picture?

Football scores?

ATM withdrawal?

Bank balance?

Parcel tracker?

Pick 2

Pick 2

Understand the tradeoffs

Pick 2

All distributed systems have to deal with CAP

All distributed systems have to deal with CAP

Microservices add CAP **where it wasn't before**

“Microservices are great for turning method calls in
to distributed computing problems”

Aaron Patterson

“A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable”

Leslie Lamport

The fallacies of distributed computing

The network is reliable

Latency is zero

Bandwidth is infinite

The network is secure

Topology doesn't change

There is one administrator

Transport cost is zero

The network is homogeneous

The network is reliable

~~The network is reliable~~

~~The network is reliable~~

Infrastructure is attacked

~~The network is reliable~~

Infrastructure is attacked



~~The network is reliable~~

Infrastructure is attacked

Networks are attacked



~~The network is reliable~~

Infrastructure is attacked

Networks are attacked

Machines fail



~~The network is reliable~~

Infrastructure is attacked

Networks are attacked

Machines fail



```
$traceroute wikipedia.org
traceroute to wikipedia.org (66.230.200.100), 64 hops max, 44 byte packets
 1 124.ae0.xr1.3d12.xs4all.net (194.109.21.1) 0.305 ms 0.360 ms 0.405 ms
 2 0.so-6-0-0.xr1.tc2.xs4all.net (194.109.5.10) 0.634 ms 0.716 ms 0.673 ms
 3 ams-ix-c00.wvfiber.net (195.69.145.58) 0.638 ms 0.601 ms 0.551 ms
 4 lon-c00-pos-4-0.OC48-ams-pos11-0.wvfiber.net (63.223.28.201) 7.512 ms 7.427 ms 7.494 ms
 5 nyc60-pos-1-0.OC48-lon-c00-pos-3-0.wvfiber.net (63.223.28.145) 84.108 ms 83.804 ms 83.995 ms
 6 66.216.1.181 (66.216.1.181) 83.435 ms 83.278 ms 83.348 ms
 7 ash-c01-tge-3-3.TG-nyc-c01-1-1.wvfiber.net (66.216.1.161) 89.563 ms 89.554 ms 89.551 ms
 8 atl-c01-tge-3-1.TG-ash-c01-3-1.wvfiber.net (66.216.1.157) 103.701 ms 103.606 ms 103.596 ms
 9 cpp-hostway.wvfiber.net (63.223.8.26) 103.678 ms 103.609 ms 103.630 ms
10 e1-12.co2.as30217.net (64.156.25.105) 113.014 ms 113.044 ms 113.084 ms
11 10ge5-1.csv5-pmptpa.wikimedia.org (84.40.25.102) 113.153 ms 113.251 ms 113.180 ms
12 rr.pmptpa.wikimedia.org (66.230.200.100) 113.069 ms 113.172 ms 113.003 ms
```

The latency is zero

~~The latency is zero~~

~~The latency is zero~~

US East to west (+~65ms)

~~The latency is zero~~

US East to west (+~65ms)

Atlantic crossing (+~90ms)

London to Auckland (+~270ms)

~~The latency is zero~~

US East to west (+~65ms)

Atlantic crossing (+~90ms)

London to Auckland (+~270ms)



~~The latency is zero~~

US East to west (+~65ms)

Atlantic crossing (+~90ms)

London to Auckland (+~270ms)

The case of the 500-mile email



Bandwidth is infinite

~~Bandwidth is infinite~~

~~Bandwidth is finite~~



~~Bandwidth is finite~~

VOIP, streaming video/audio...



~~Bandwidth is finite~~

VOIP, streaming video/audio...

1000s of services sharing a line?



~~Bandwidth is infinite~~

VOIP, streaming video/audio...

1000s of services sharing a line?

Even T-1 lines get saturated

Bottlenecks?



The network is secure

~~The network is secure~~

~~The network is secure~~

Heartbleed?



~~The network is secure~~

Heartbleed?

DynDNS?



“In a relatively short time we've taken a system built to resist destruction by nuclear weapons and made it vulnerable to toasters.”

Jeff Jarmoc

~~The network is secure~~

Heartbleed?

DynDNS?

Network border isn't good enough



“In a relatively short time we've taken a system built to resist destruction by nuclear weapons and made it vulnerable to toasters.”

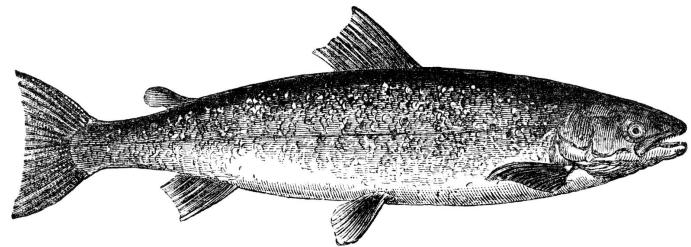
Jeff Jarmoc

~~The network is secure~~

Heartbleed?

DynDNS?

Network border isn't good enough



Expert

Hoping Nobody
Hacks You

Topology doesn't change

~~Topology doesn't change~~

~~Topology doesn't change~~

Topology is dynamic!

~~Topology doesn't change~~

Topology is dynamic! (Cattle not pets)

~~Topology doesn't change~~

Topology is dynamic! (Cattle not pets)

Can't rely on specific endpoints...

~~Topology doesn't change~~

Topology is dynamic! (Cattle not pets)

Can't rely on specific endpoints...

...or performance, routing etc.

~~Topology~~ doesn't change

Topology is dynamic! (Cattle not pets)

Can't rely on specific endpoints...

...or performance, routing etc.

Bits a network can be unreachable

There is one administrator

~~There is one administrator~~

~~There is one administrator~~

Who grants access to database?

~~There is one administrator~~

Who grants access to database?

...and to another service?

~~There is one administrator~~

Who grants access to database?

...and to another service?



~~There is one administrator~~

Who grants access to database?

...and to another service?

Who do you complain to?



~~There is one administrator~~

Who grants access to database?

...and to another service?

Who do you complain to?

Coordinated updates?



Transport cost is zero

~~Transport cost is zero~~

~~Transport cost is zero~~

Literally not free

Data Transfer OUT From Amazon S3 To Internet

First 1 GB / month	\$0.000 per GB
Up to 10 TB / month	\$0.090 per GB
Next 40 TB / month	\$0.085 per GB
Next 100 TB / month	\$0.070 per GB
Next 150 TB / month	\$0.050 per GB

~~Transport cost is zero~~

Literally not free

Who's paying?

Data Transfer OUT From Amazon S3 To Internet

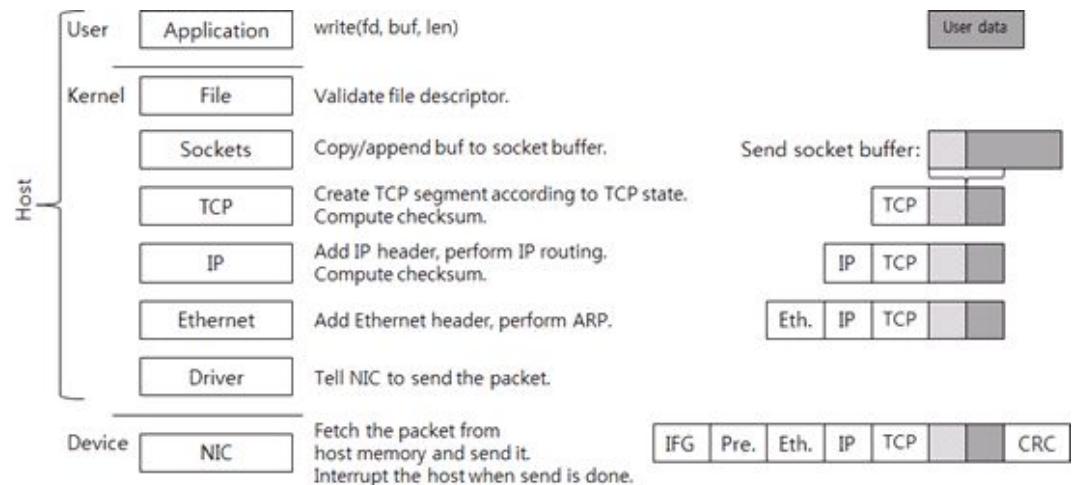
First 1 GB / month	\$0.000 per GB
Up to 10 TB / month	\$0.090 per GB
Next 40 TB / month	\$0.085 per GB
Next 100 TB / month	\$0.070 per GB
Next 150 TB / month	\$0.050 per GB

~~Transport cost is zero~~

Literally not free

Who's paying?

Marshalling isn't free either



The network is homogeneous

~~The network is homogeneous~~

~~The network is homogeneous~~

My pockets aren't homogeneous!



~~The network is homogeneous~~

My pockets aren't homogeneous!

Can't assume you're talking to UNIX etc.

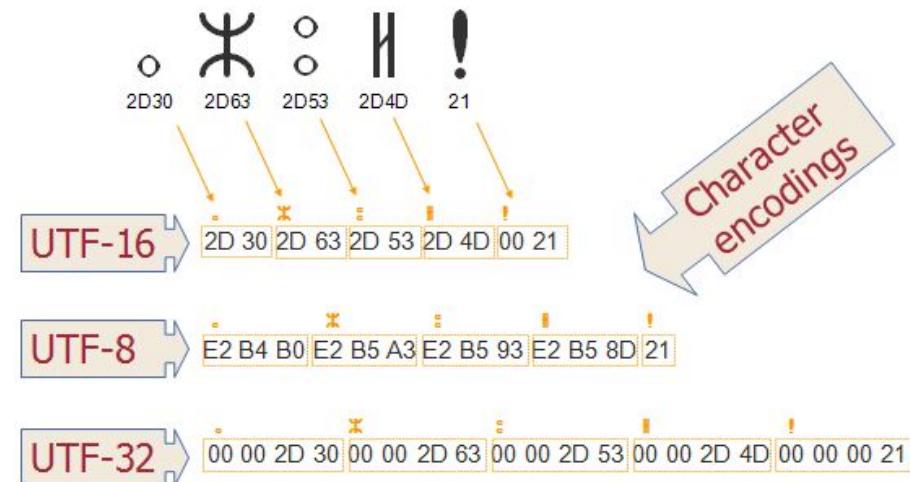


~~The network is homogeneous~~

My pockets aren't homogeneous!

Can't assume you're talking to UNIX etc.

Can't assume character encodings



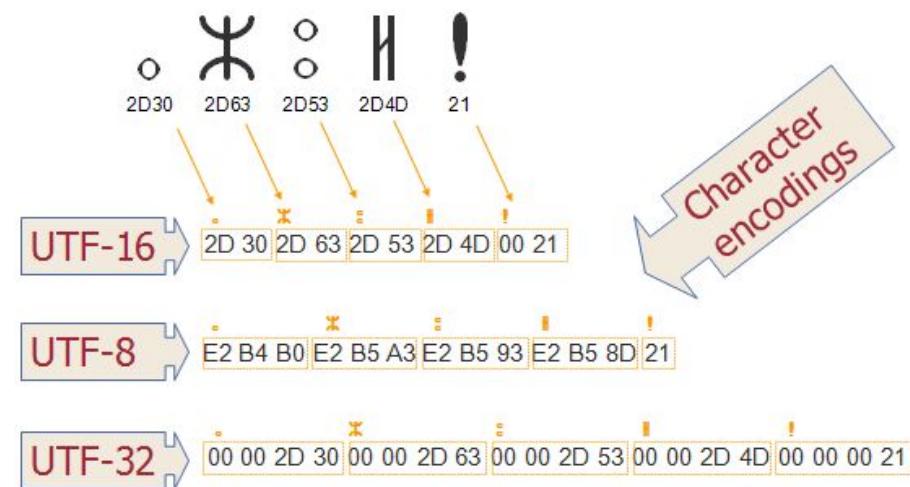
~~The network is homogeneous~~

My pockets aren't homogeneous!

Can't assume you're talking to UNIX etc.

Can't assume character encodings

Have to agree on exchange formats



The fallacies of distributed computing

The network is reliable

Latency is zero

Bandwidth is infinite

The network is secure

Topology doesn't change

There is one administrator

Transport cost is zero

The network is homogeneous

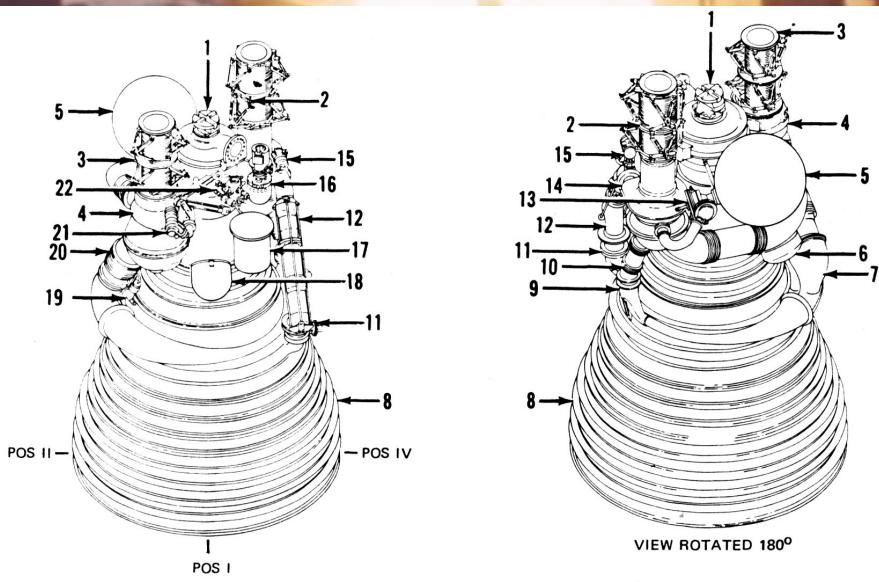
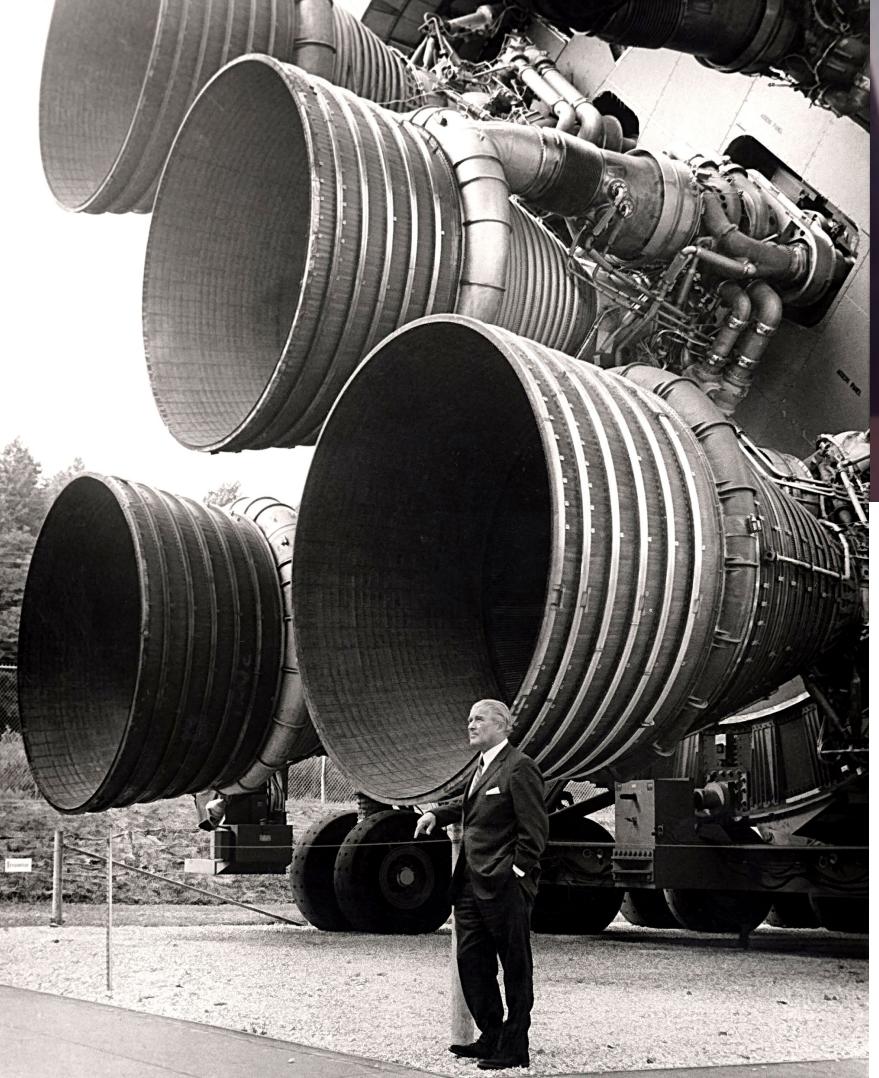


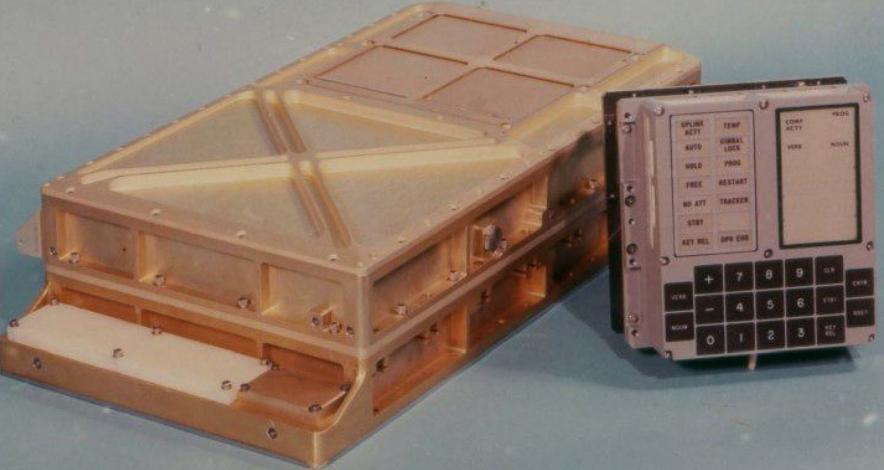
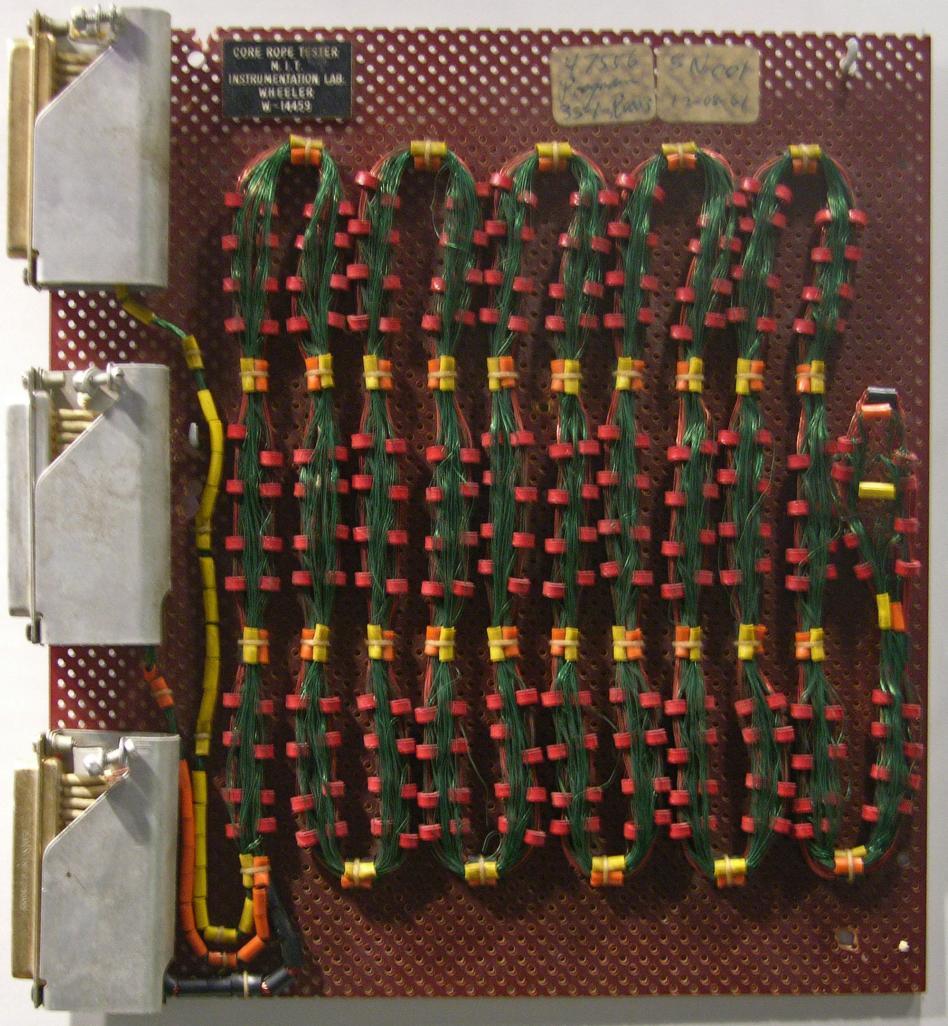
Aaron Cohen



Aaron Cohen
NASA









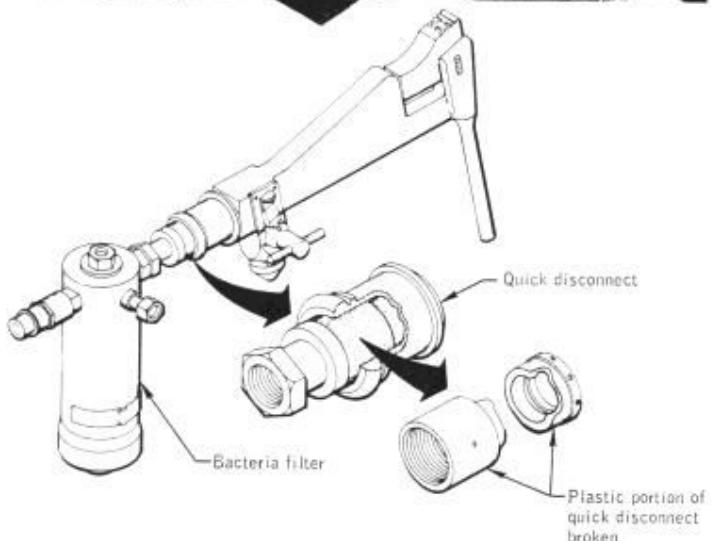
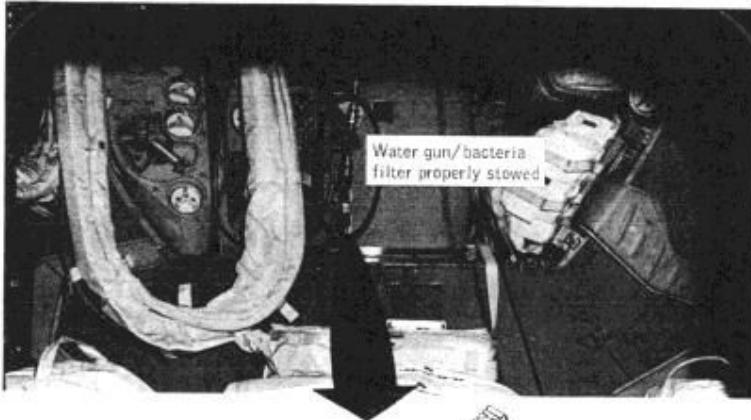
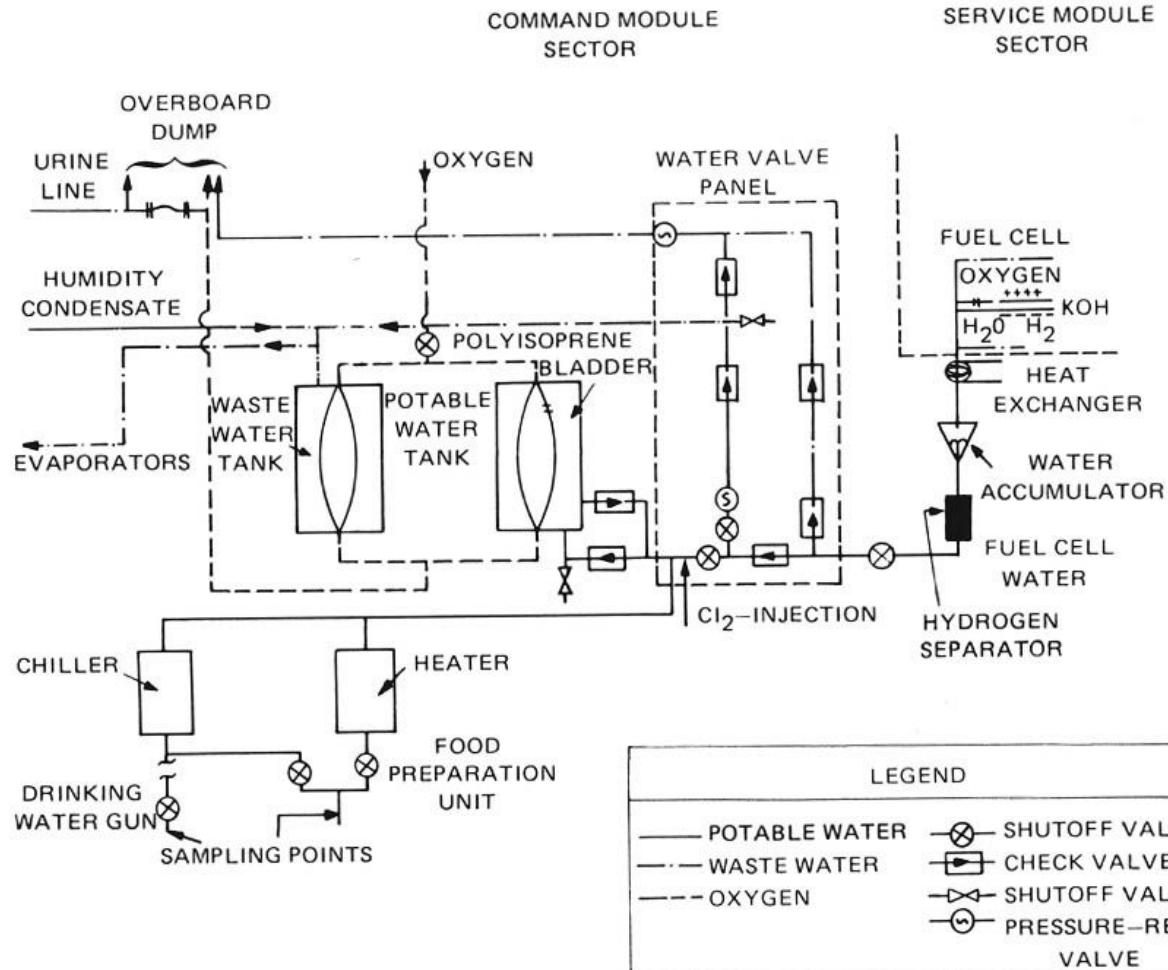
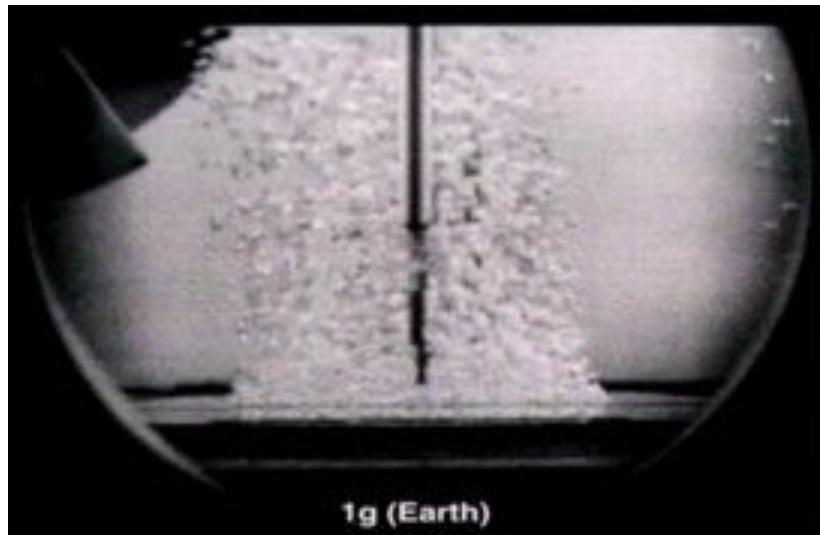
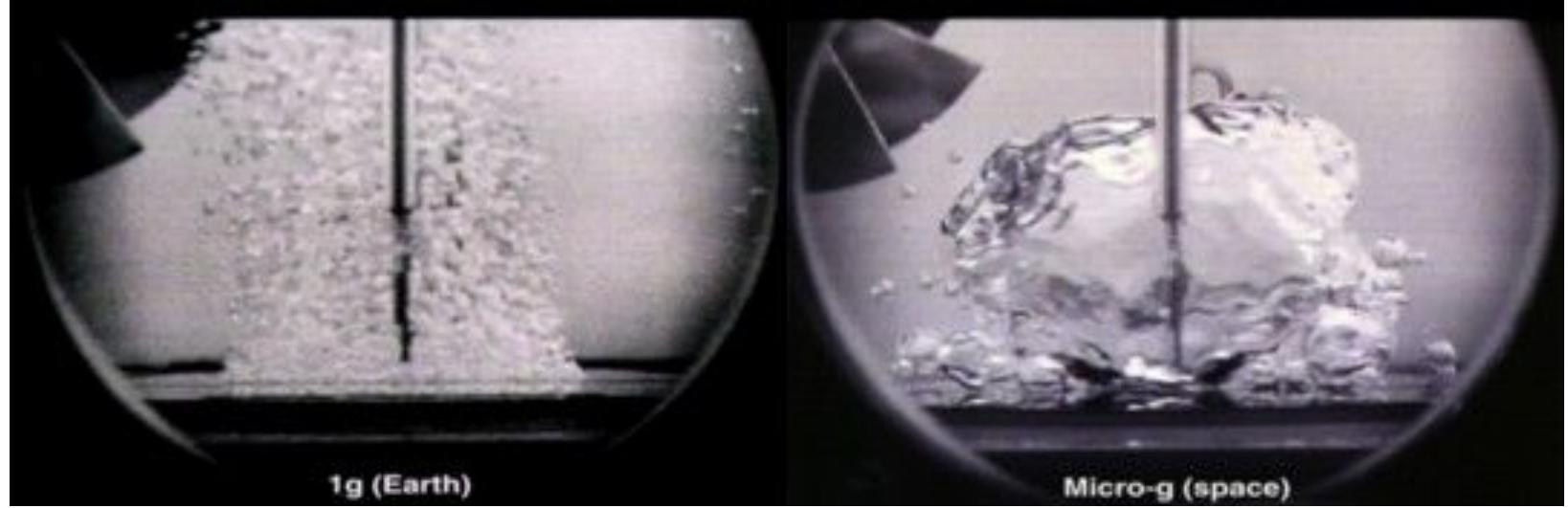


Figure 14-28.- Water gun/bacteria filter.





1g (Earth)









What Microservices actually look like

DevOps

DevOps

From Wikipedia, the free encyclopedia



Some of this article's [listed sources](#) **may not be reliable**. Please help this article by looking for better, more reliable sources. Unreliable citations may be challenged or deleted. *(December 2018)*
[*\(Learn how and when to remove this template message\)*](#)

DevOps (a clipped compound of "development" and "operations") is a set of software development practices^[failed verification] that combines software development (*Dev*) with information technology operations (*Ops*) to shorten the systems development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives.^[1]

Software development

Core activities

Processes · Requirements · Design · Engineering · Construction · Testing · Debugging · Deployment · Maintenance

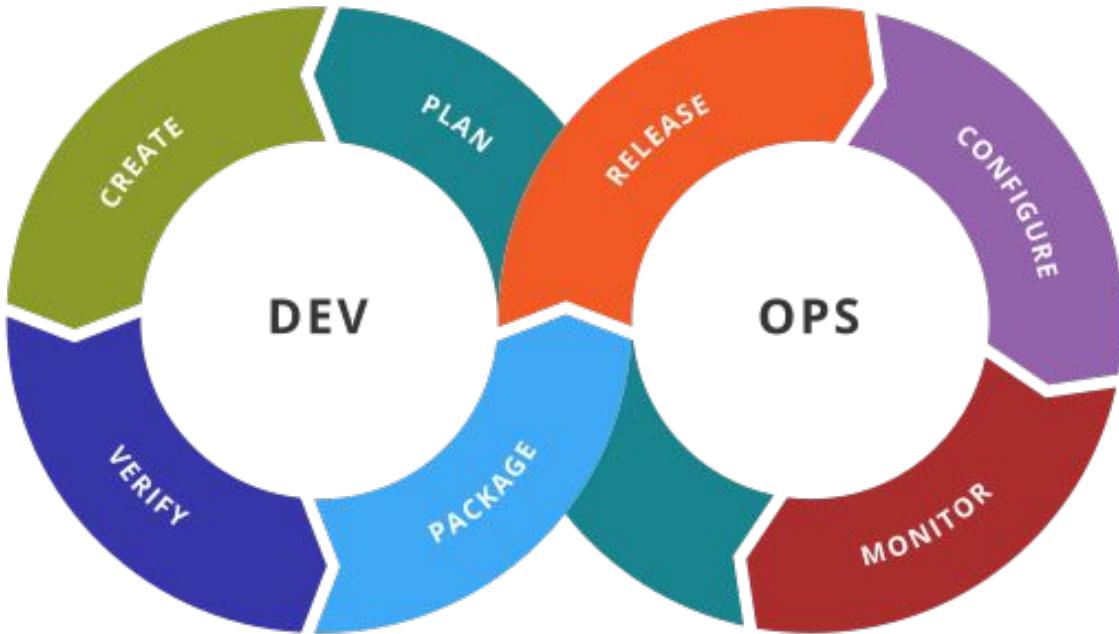
Paradigms and models

“10+ Deploys Per Day: Dev and Ops Cooperation at Flickr”

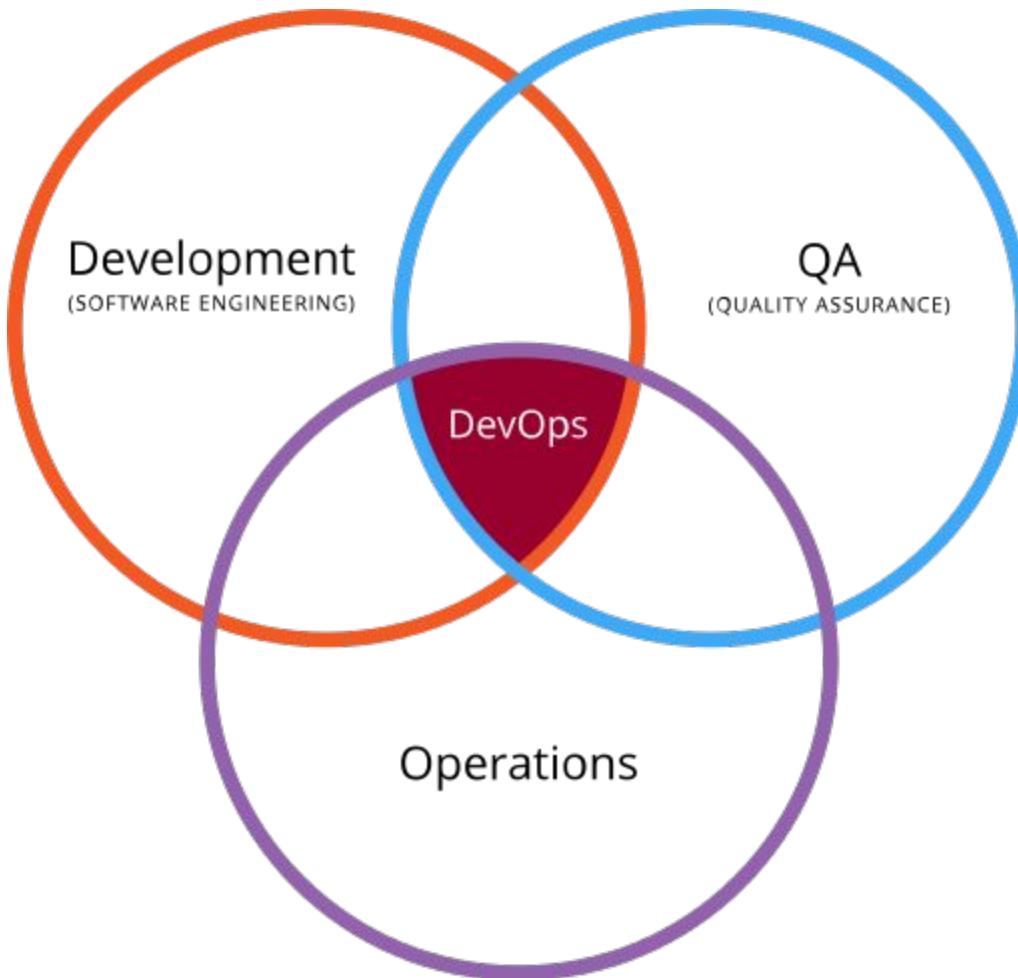
John Allspaw, Velocity 2009

“...a cross-disciplinary community of practice dedicated to the study of building, evolving and operating rapidly-changing resilient systems at scale.”

Jez Humble







**WORKED FINE IN
DEV**

OPS PROBLEM NOW

memegenerator.net

Continuous Delivery

Configuration Management

Monitoring

Continuous Delivery

“If you have more meetings than releases, you’re
an enterprise company”

Clifton Cunningham, CTO @ TES Global

“Un[released/merged] code makes me nervous”

Clifton Cunningham, CTO @ TES Global

Automation



Jenkins



circleci



Containers

Containers

credit to Anne Currie



Bare Metal

Bare Metal

- ✓ Powerful
- ✓ Simple

Bare Metal

✓ Powerful

✓ Simple

✗ Brittle

✗ Inflexible

Bare Metal

- ✓ Powerful
- ✓ Simple
- ✗ Brittle
- ✗ Inflexible



Virtual Machines

Virtual Machines

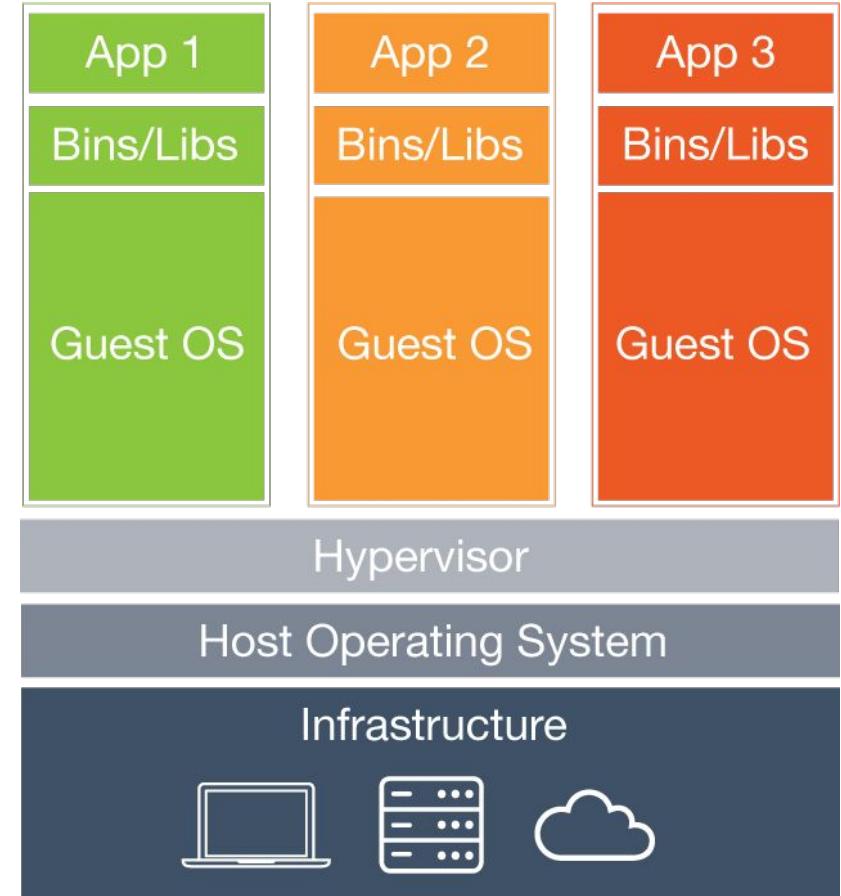
- ✓ Flexible
- ✓ Networking
- ✓ Security

Virtual Machines

- ✓ Flexible
- ✓ Networking
- ✓ Security
- ✗ Overweight

Virtual Machines

- ✓ Flexible
- ✓ Networking
- ✓ Security
- ✗ Overweight



Virtual Machines

- ✓ Flexible
- ✓ Networking
- ✓ Security
- ✗ Overweight



Containers

Containers

- ✓ Lightweight
- ✓ Agile

Containers

✓ Lightweight

✓ Agile

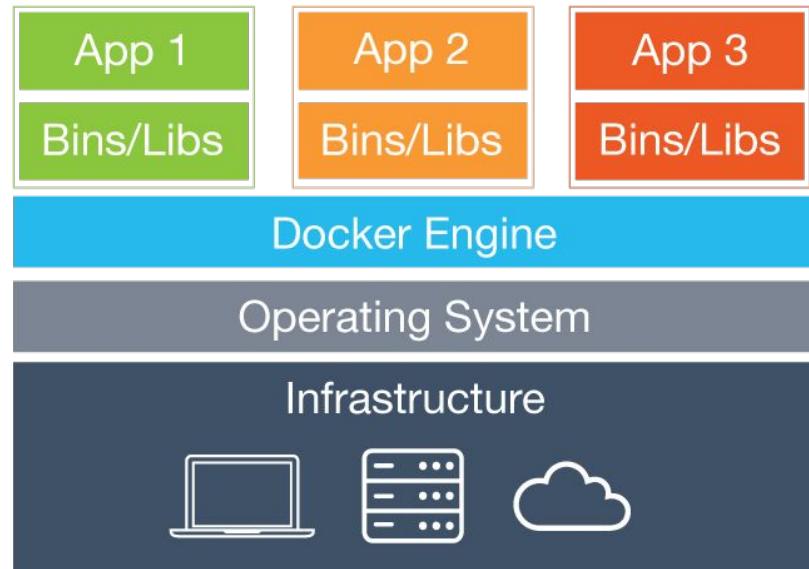
✗ Untested

✗ Networking

✗ Security

Containers

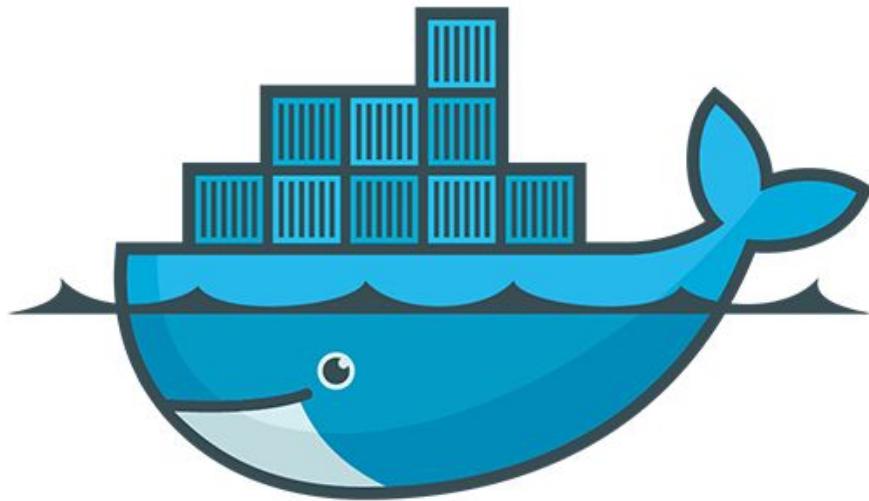
- ✓ Lightweight
- ✓ Agile
- ✗ Untested
- ✗ Networking
- ✗ Security



Containers

- ✓ Lightweight
- ✓ Agile
- ✗ Untested
- ✗ Networking
- ✗ Security





docker

Exercise

Configuration Management



Immutable Infrastructure

Don't change, replace

Requires automation

Immutable Infrastructure

Don't change, replace

Requires automation

Blue-Green

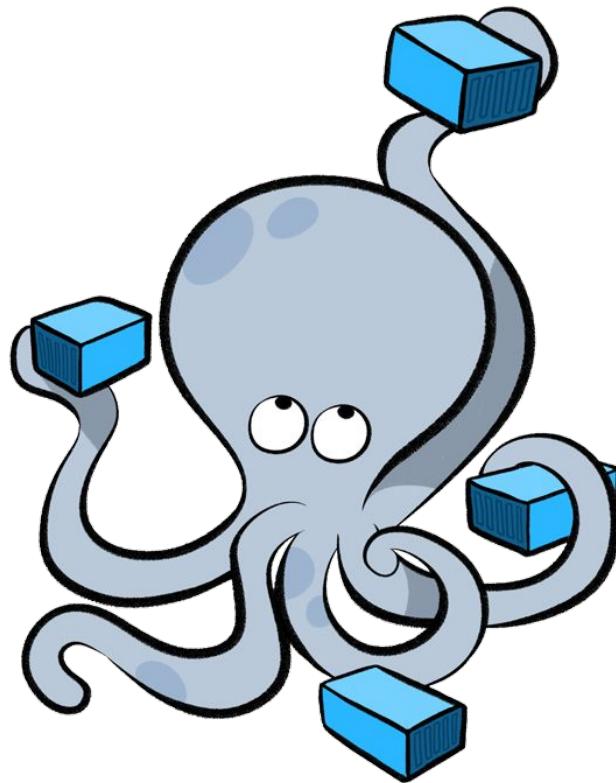
Mutable Server

Immutable Server



Docker Compose

Manage multi-container apps



Docker Compose

Manage multi-container apps

Collection of dockerfiles → services

```
version: '2'  
services:  
  web:  
    build: .  
    ports:  
      - "5000:5000"  
    volumes:  
      - ./code  
      - logvolume01:/var/log  
    links:  
      - redis  
  redis:  
    image: redis  
volumes:  
  logvolume01: {}
```

Is Docker production ready?

Is Docker production ready?

is docker production ready?



All

Videos

Images

News

Shopping

More

Settings

Tools

About 168.000 results (0,61 seconds)

Docker, not production-ready? Not so, says Docker | InfoWorld

www.infoworld.com/article/.../docker-not-production-ready-not-so-says-docker.html ▾

Mar 22, 2016 - Docker has taken dev-and-test by storm, but production deployments have awaited more mature security and management. Some customers ...

Containers reality check: They're still not production-ready - TechBeacon

<https://techbeacon.com/containers-reality-check-why-theyre-still-not-production-ready> ▾

Just because containers are increasingly popular, however, doesn't necessarily mean they're ready for prime time. Docker and other container platform vendors ...

Docker in Production: A History of Failure – The HFT Guy

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

Nov 1, 2016 - The issue with Docker is that it doesn't do any of that. It's just a dumb container system. It has the drawbacks of containers without the benefits. There are currently no good, battle tested, production ready orchestration system in existence.

Docker not ready for primetime | Hacker News

<https://news.ycombinator.com/item?id=12377457> ▾

Aug 29, 2016 - I have run docker in production at past employers, and am getting ready to do so again at my current employer. However I don't run it as a ...

Is Docker production ready?

is docker production ready?

All Videos Images News

About 168.000 results (0,61 seconds)

companies using docker

All News Videos Images

About 597.000 results (0,68 seconds)

Is Docker production ready?

Not great at backwards compatibility...

Requires extra tooling



kubernetes

is docker production ready?

All Videos Images News

About 168.000 results (0,61 seconds)

companies using docker

All News Videos Images

About 597.000 results (0,68 seconds)





“Concurrent Production”

“The development contracts are being performed concurrent with the production contracts”

2015 Lockheed Martin Annual Report

Release It!

Design and Deploy
Production-Ready Software



Michael T. Nygard

Ship It!

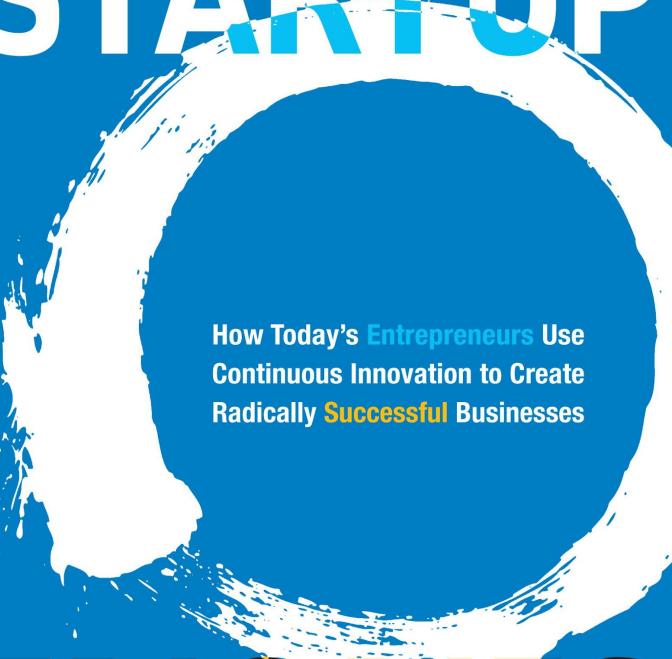
*A Practical Guide to
Successful Software Projects*



Jared Richardson William Gwaltney, Jr.

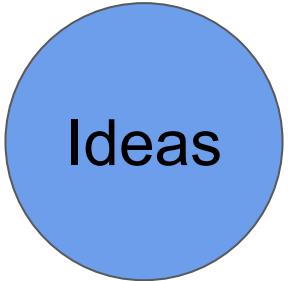
THE NEW YORK TIMES BESTSELLER

THE LEAN STARTUP

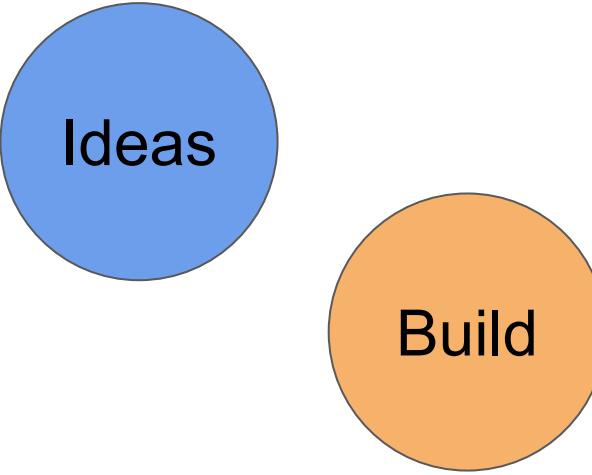


How Today's Entrepreneurs Use
Continuous Innovation to Create
Radically **Successful** Businesses

ERIC RIES

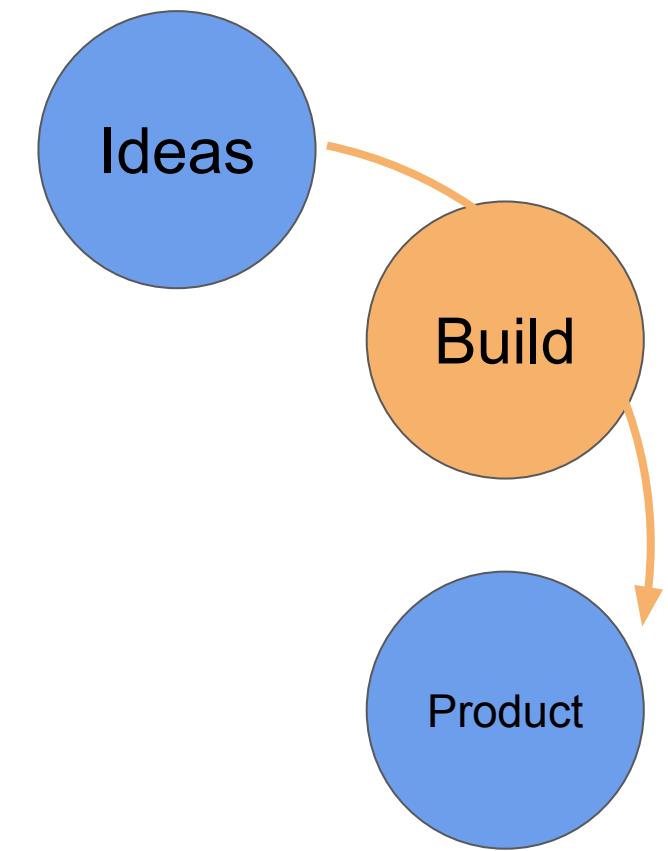


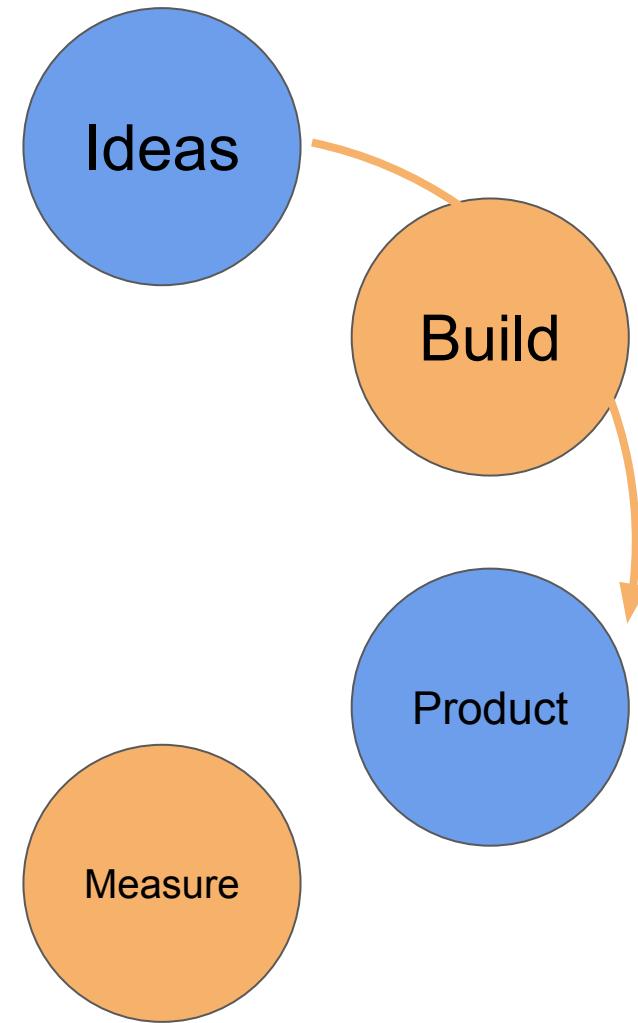
Ideas

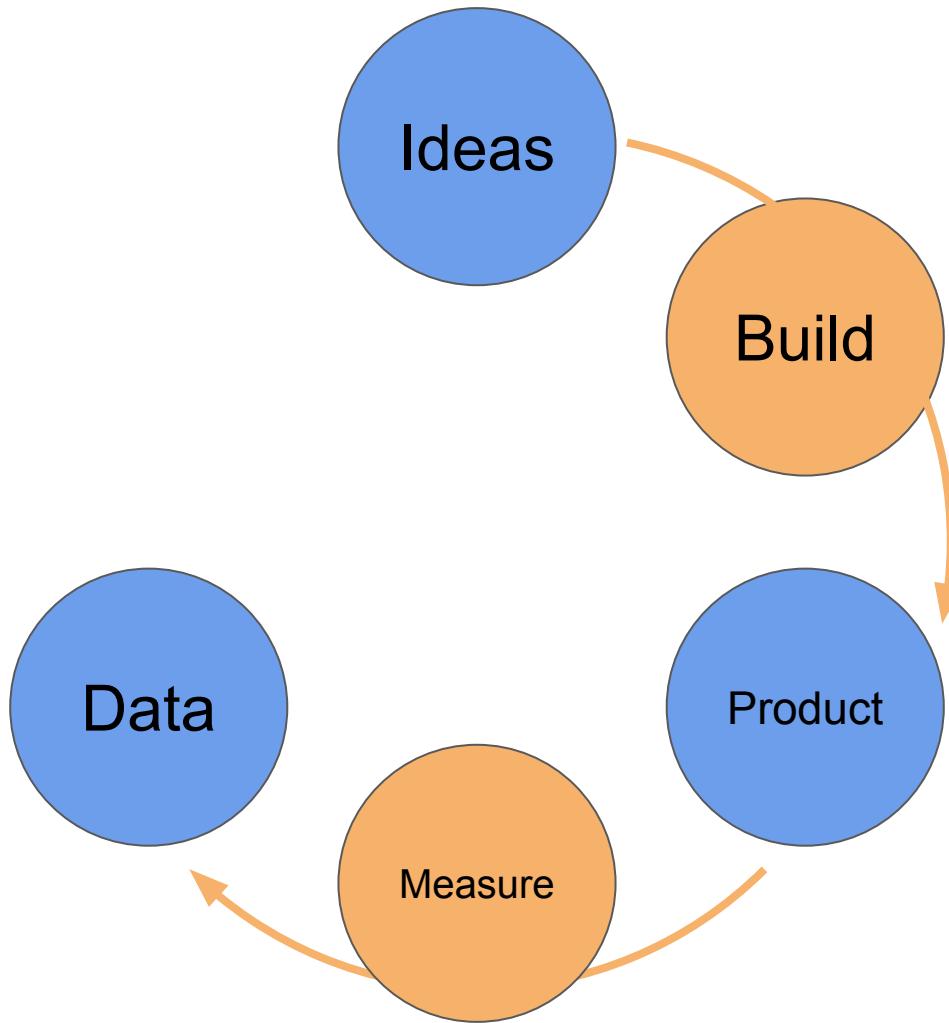


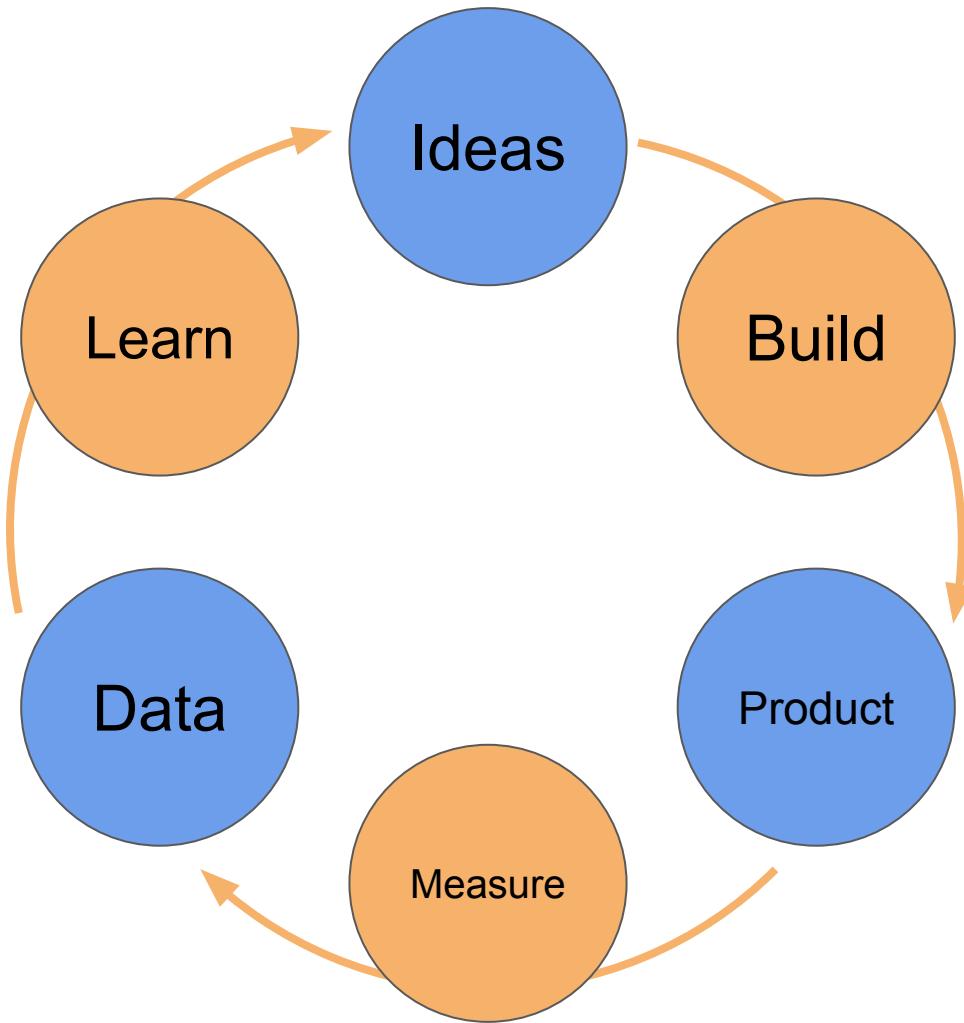
Ideas

Build











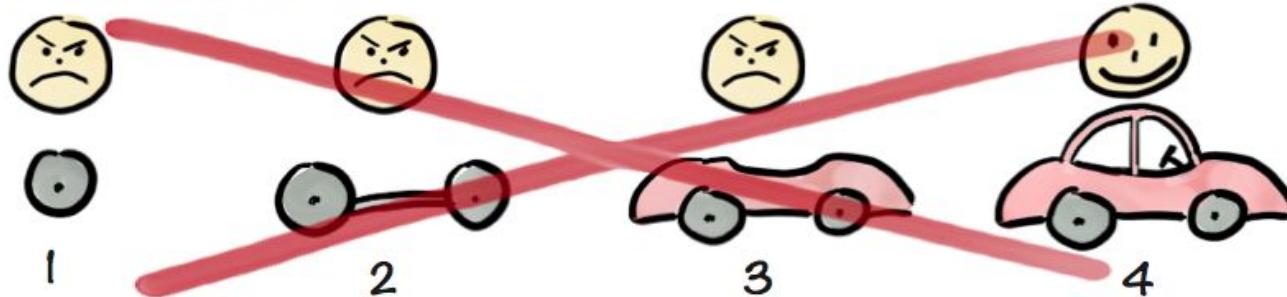
Some of the oldest F-35s on the globe are forced to sit and wait for upgrades
trib.al/CR0omrP



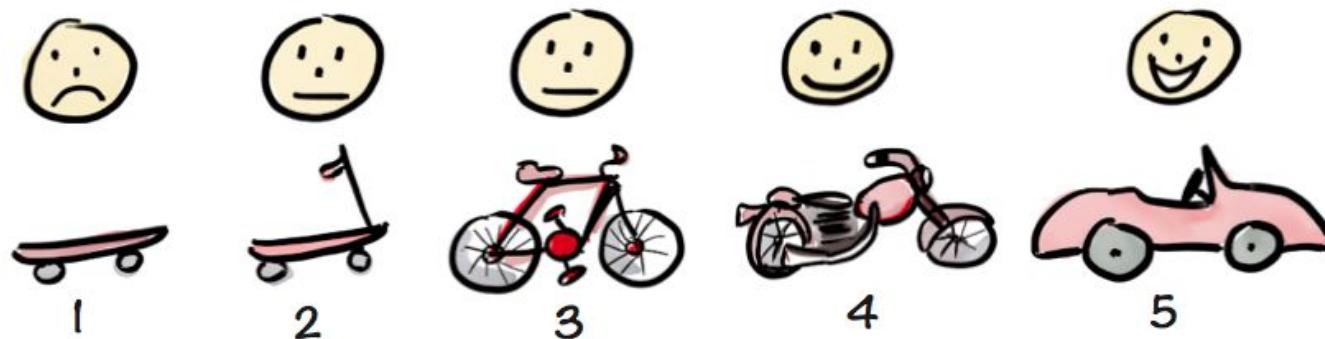
© Sightline Media Group

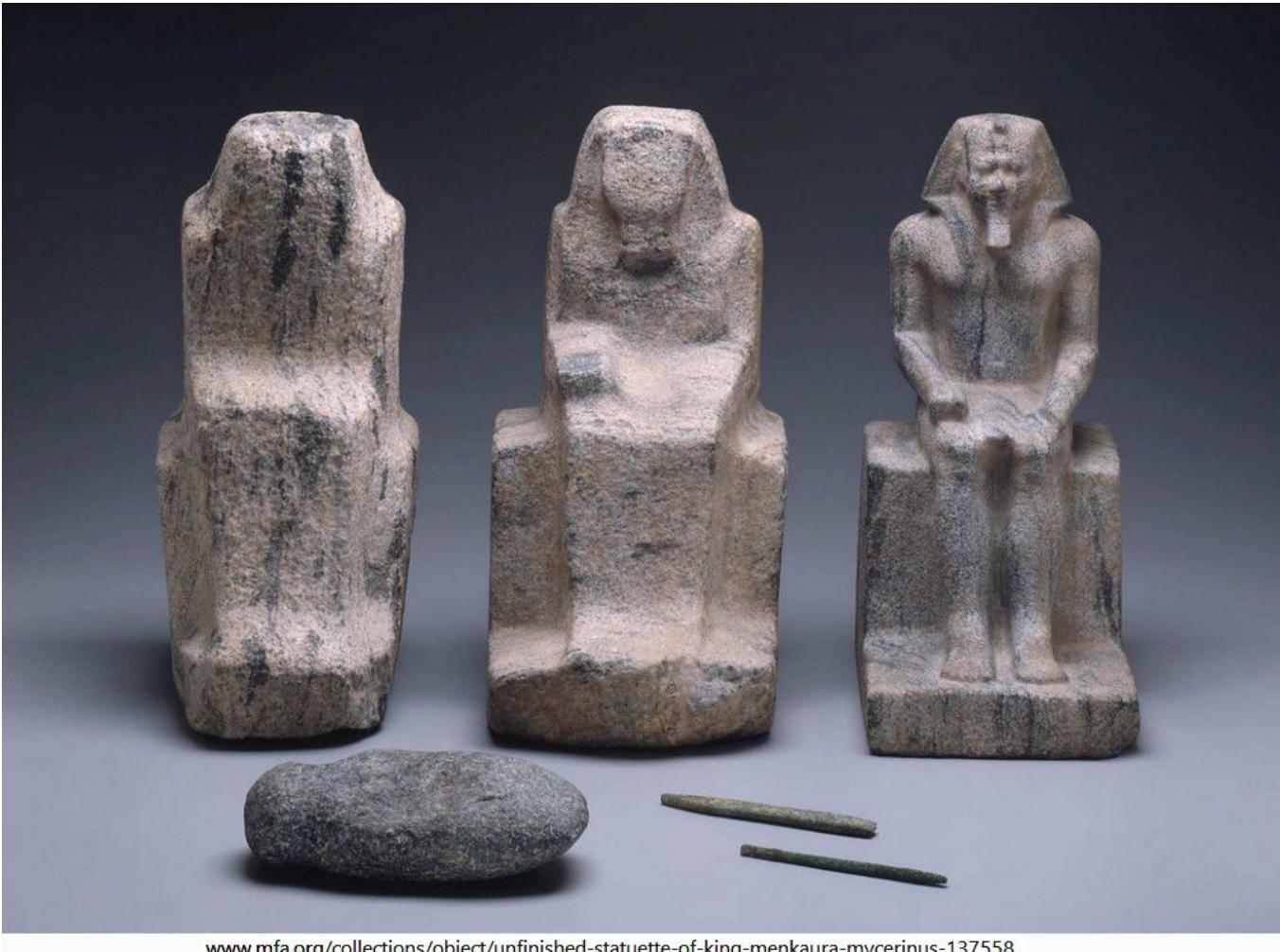
10:02 AM - 1 Jun 2018

Not like this....



Like this!



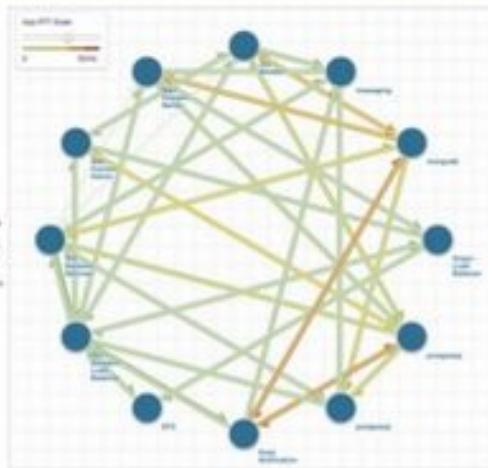


www.mfa.org/collections/object/unfinished-statuette-of-king-menkaura-mycerinus-137558

“Death Star” Architecture Diagrams



Netflix



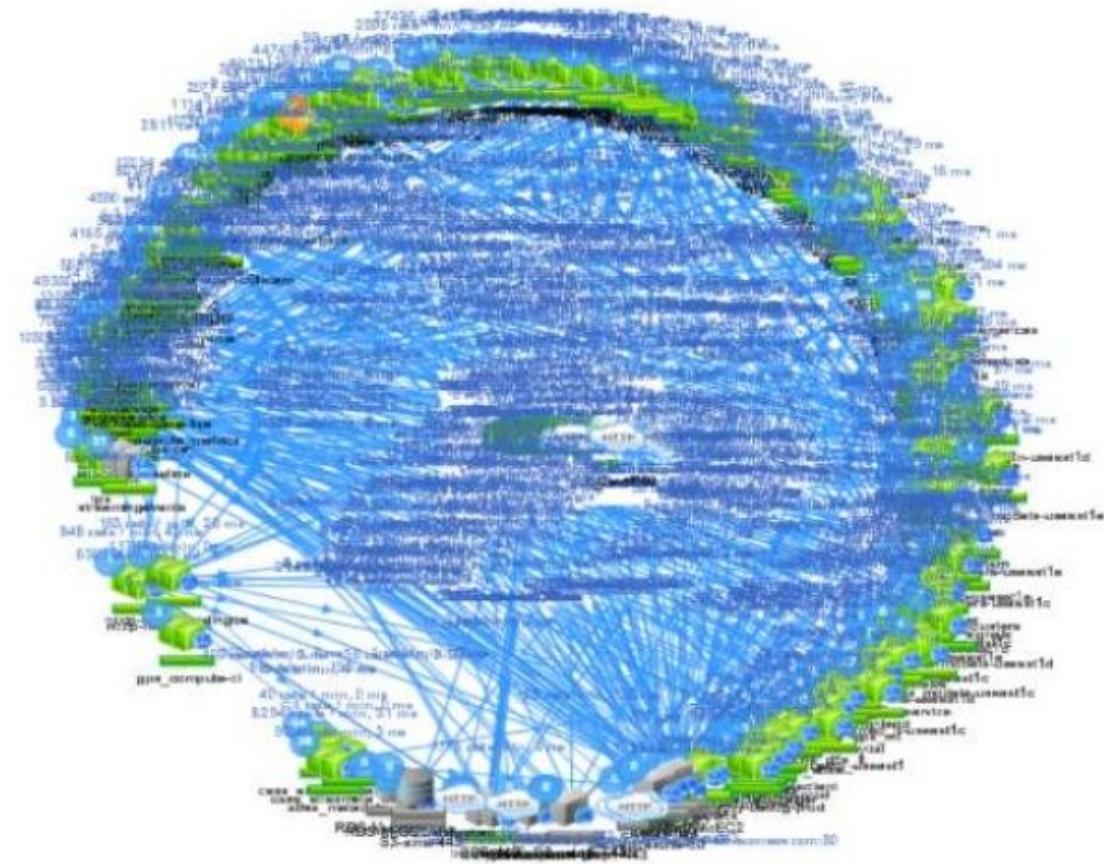
Gilt Groupe (12 of 450)



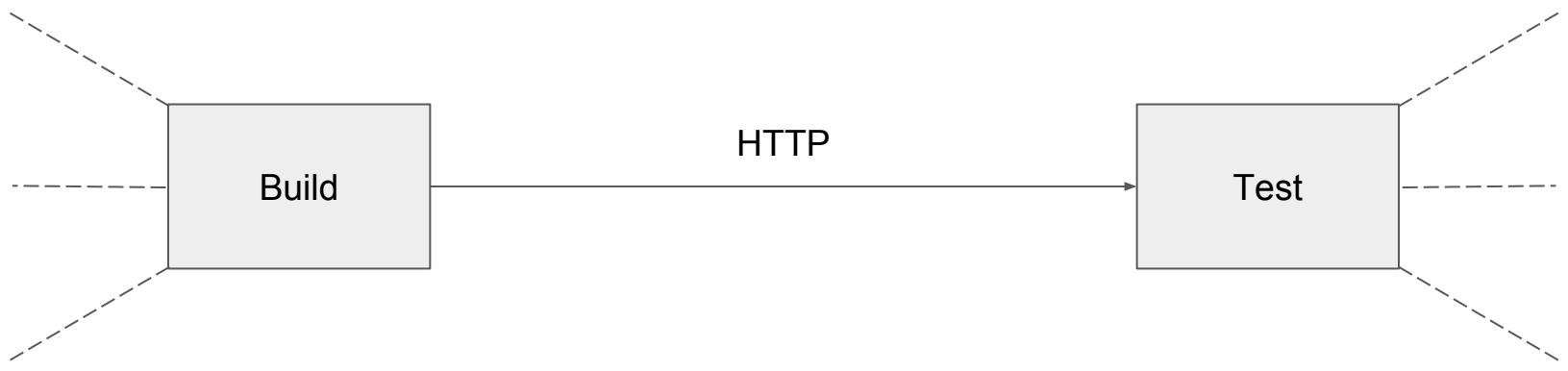
Twitter

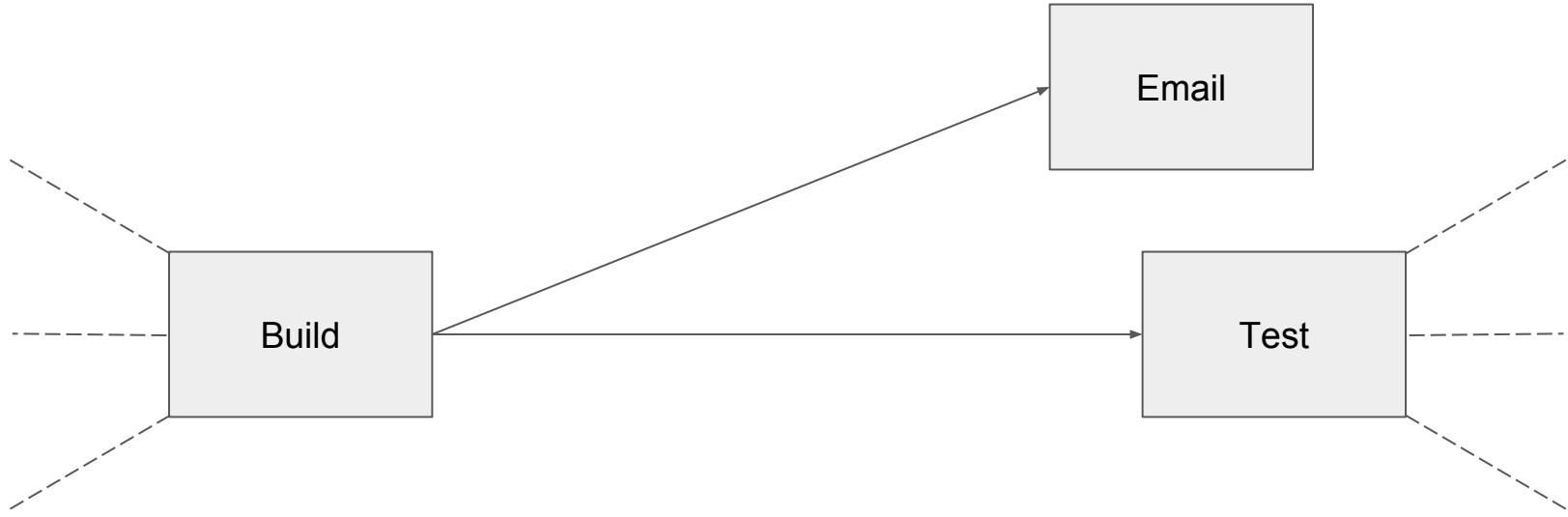
As visualized by Appdynamics, Boundary.com and Twitter internal tools

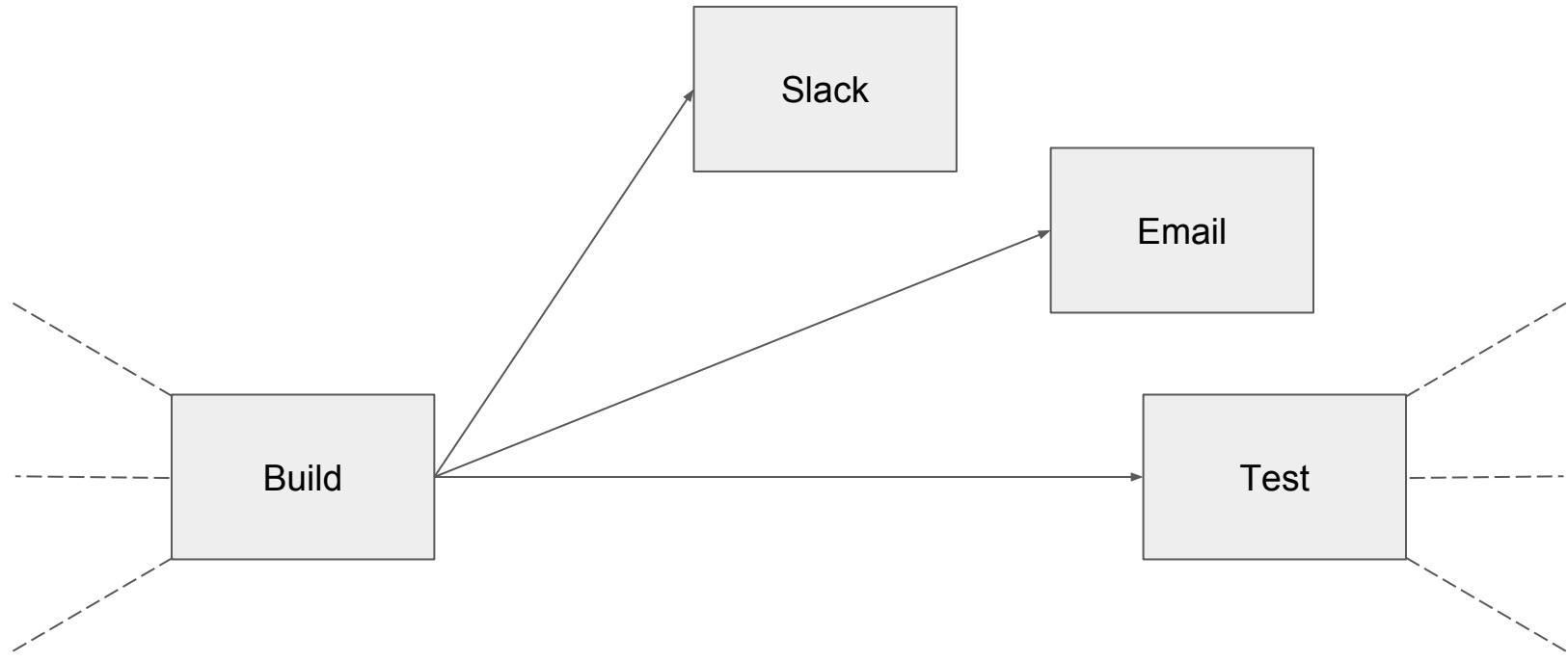
Netflix & Micro-Services

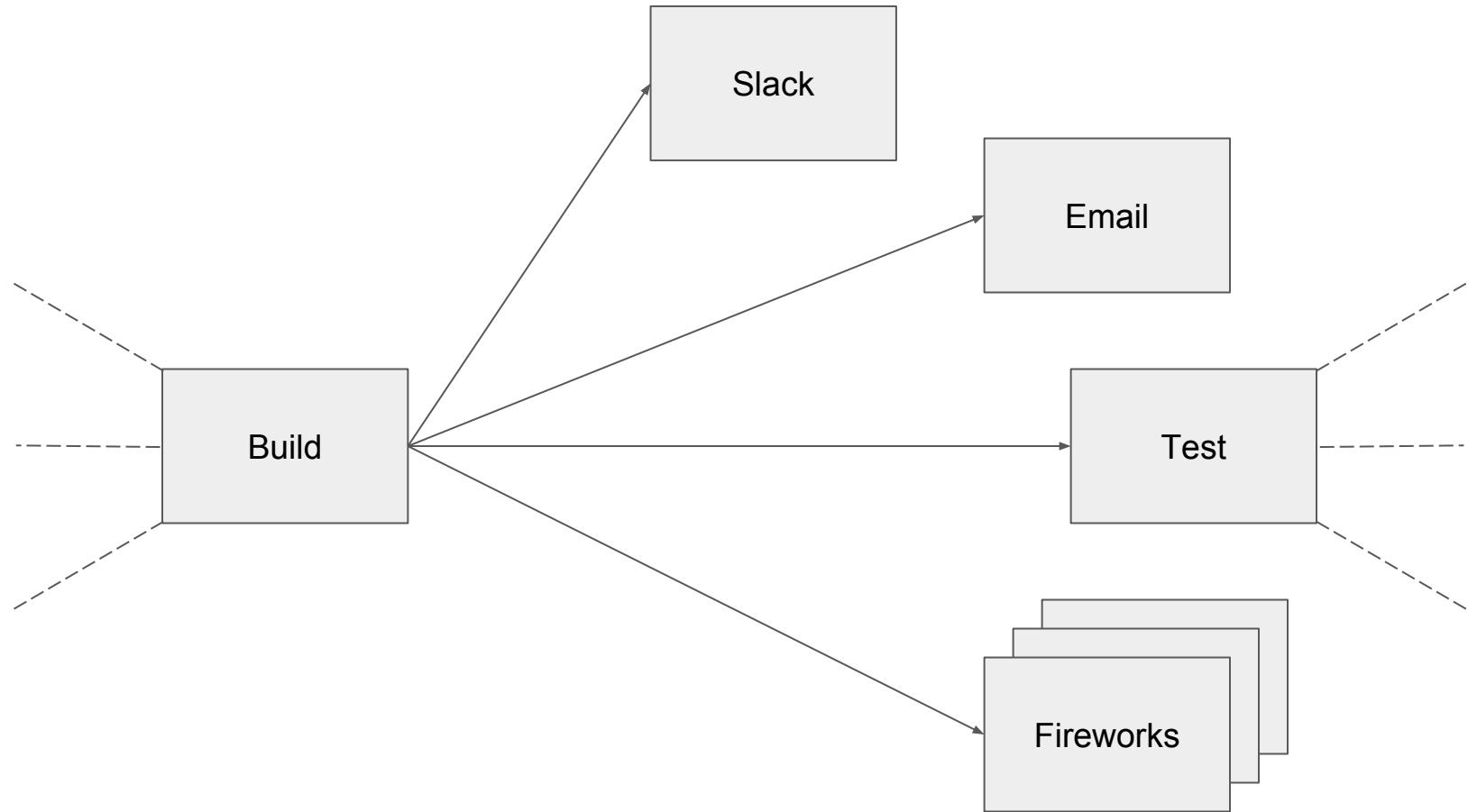


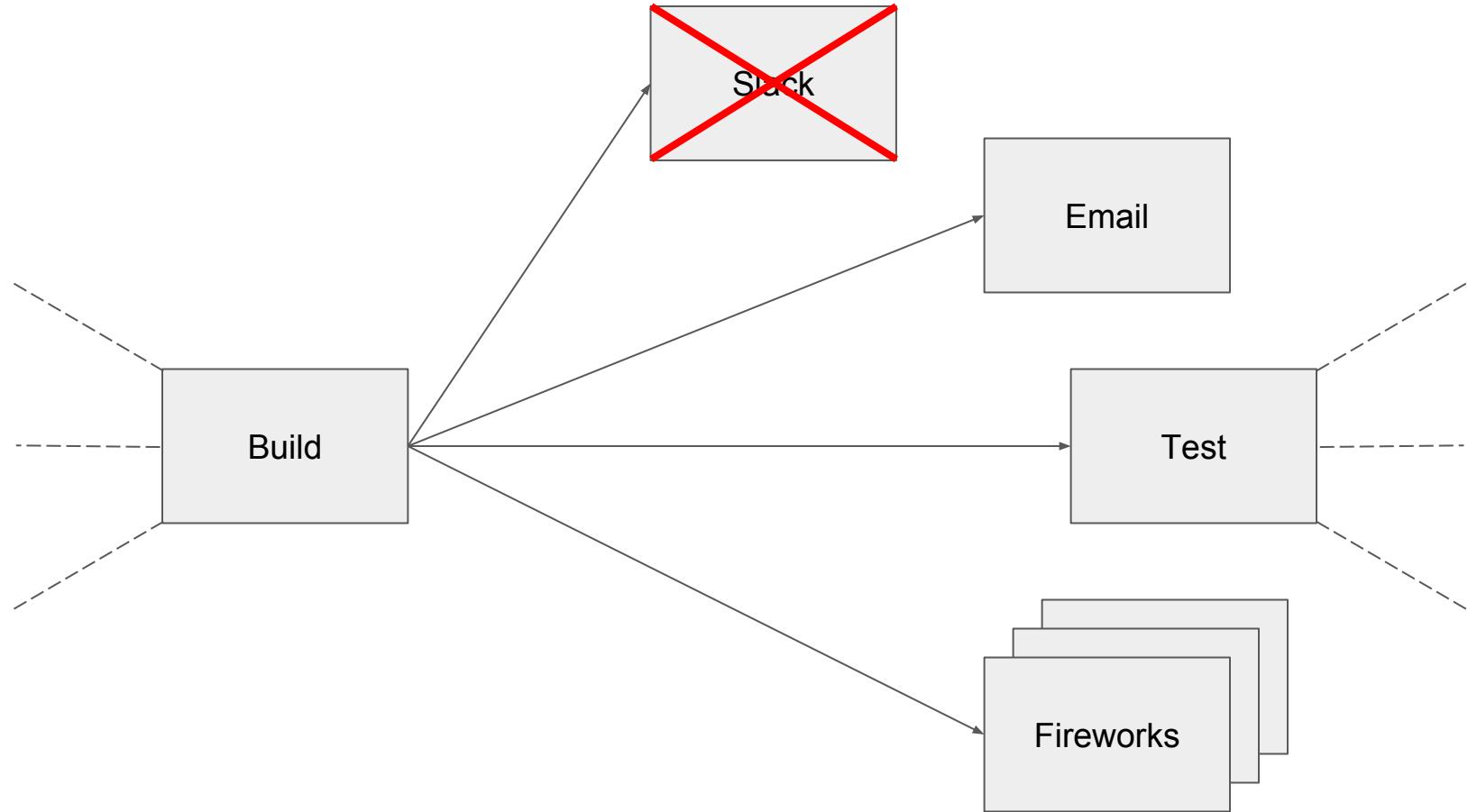
Inter-service communication

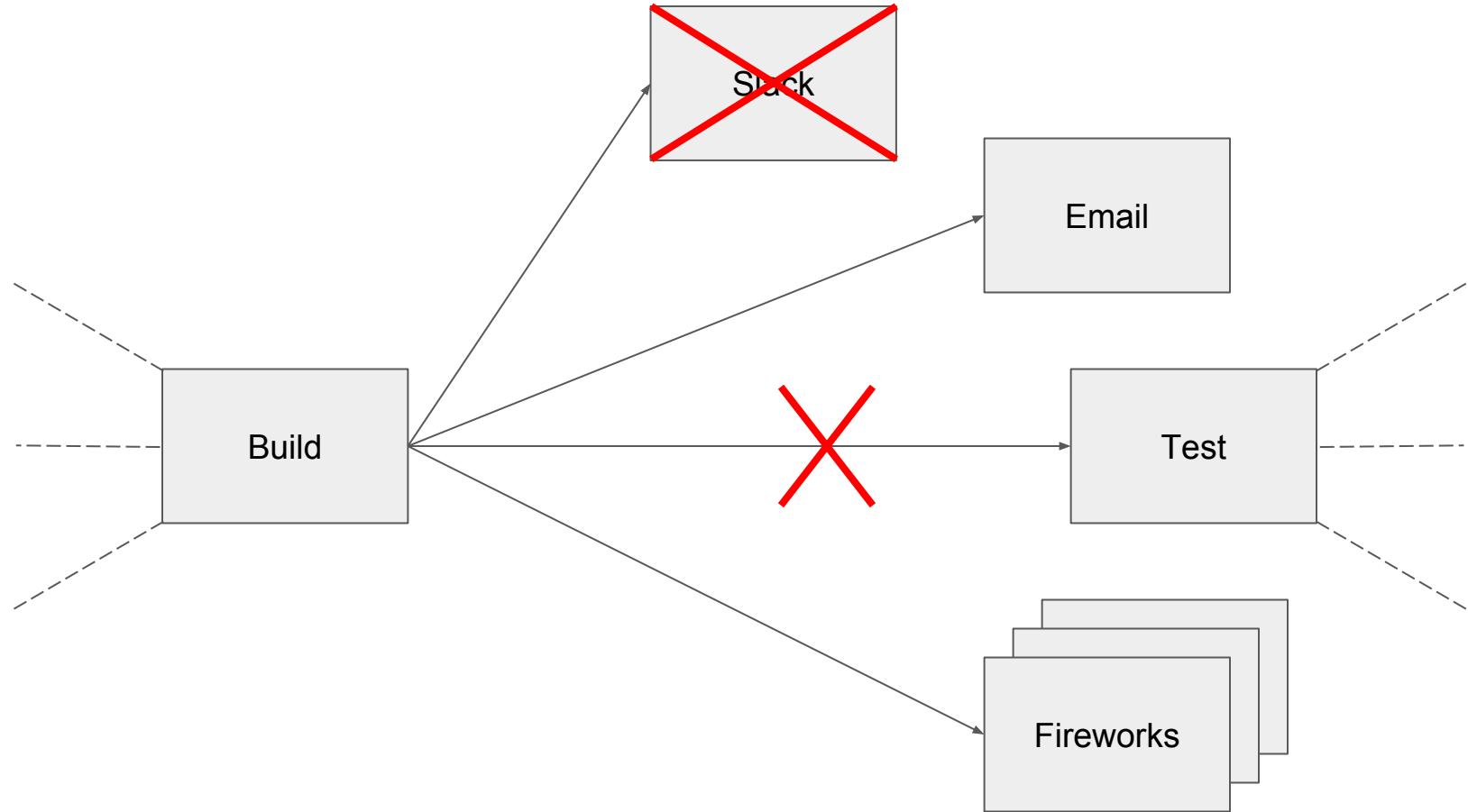














Service Discovery

“Service Discovery
is an anti-pattern”

Richard Rodger, CTO @ nearForm

Messaging!



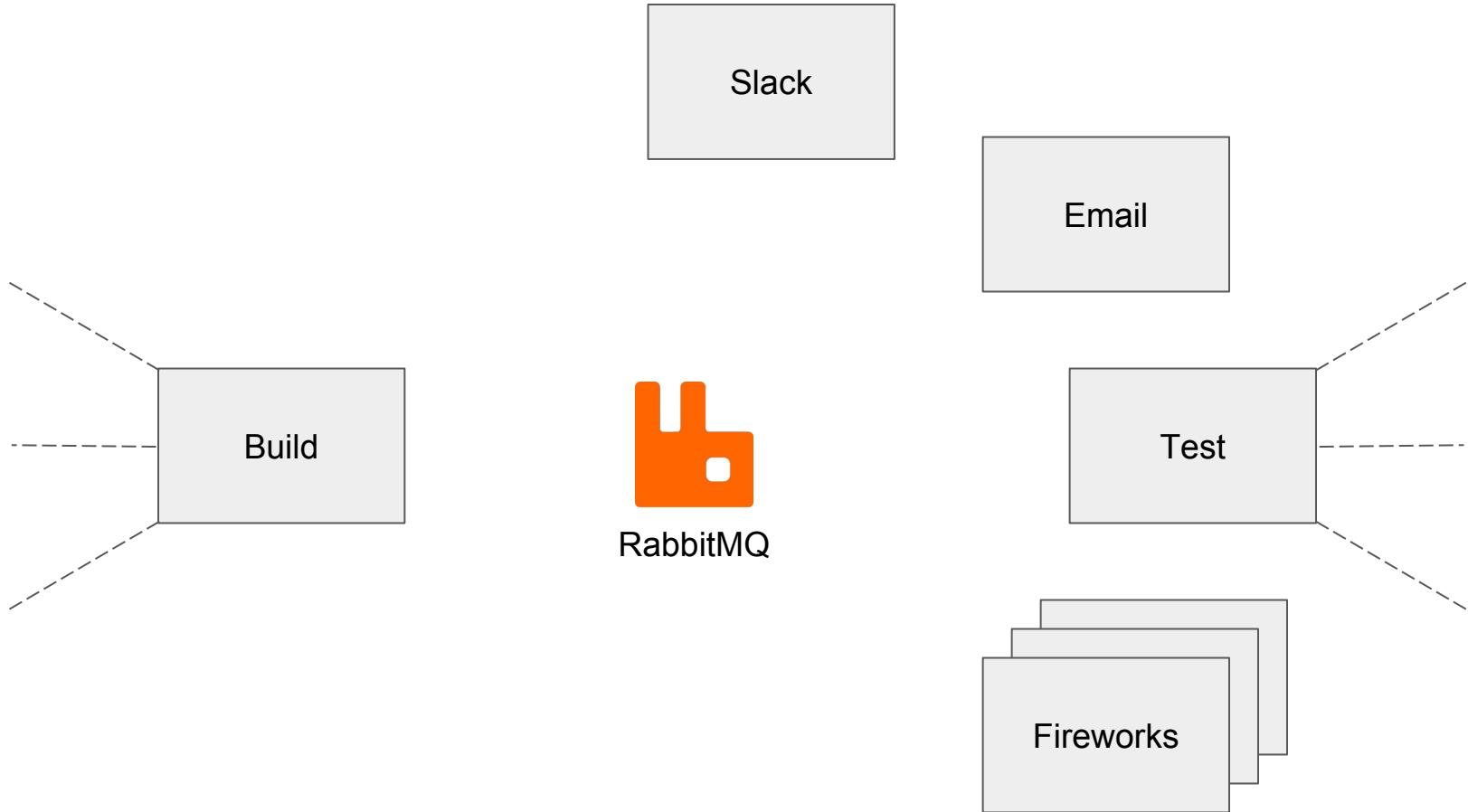
MQ Light

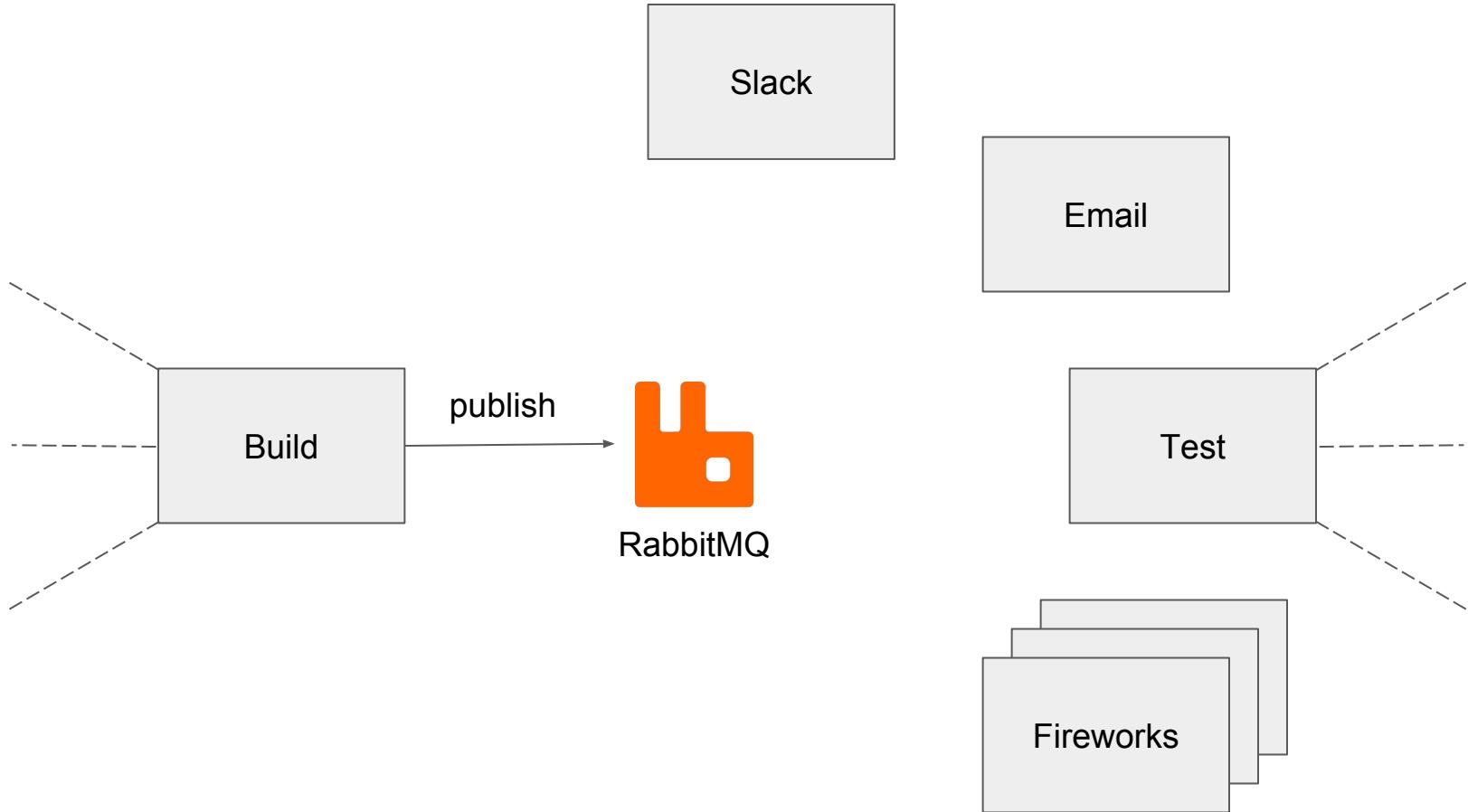


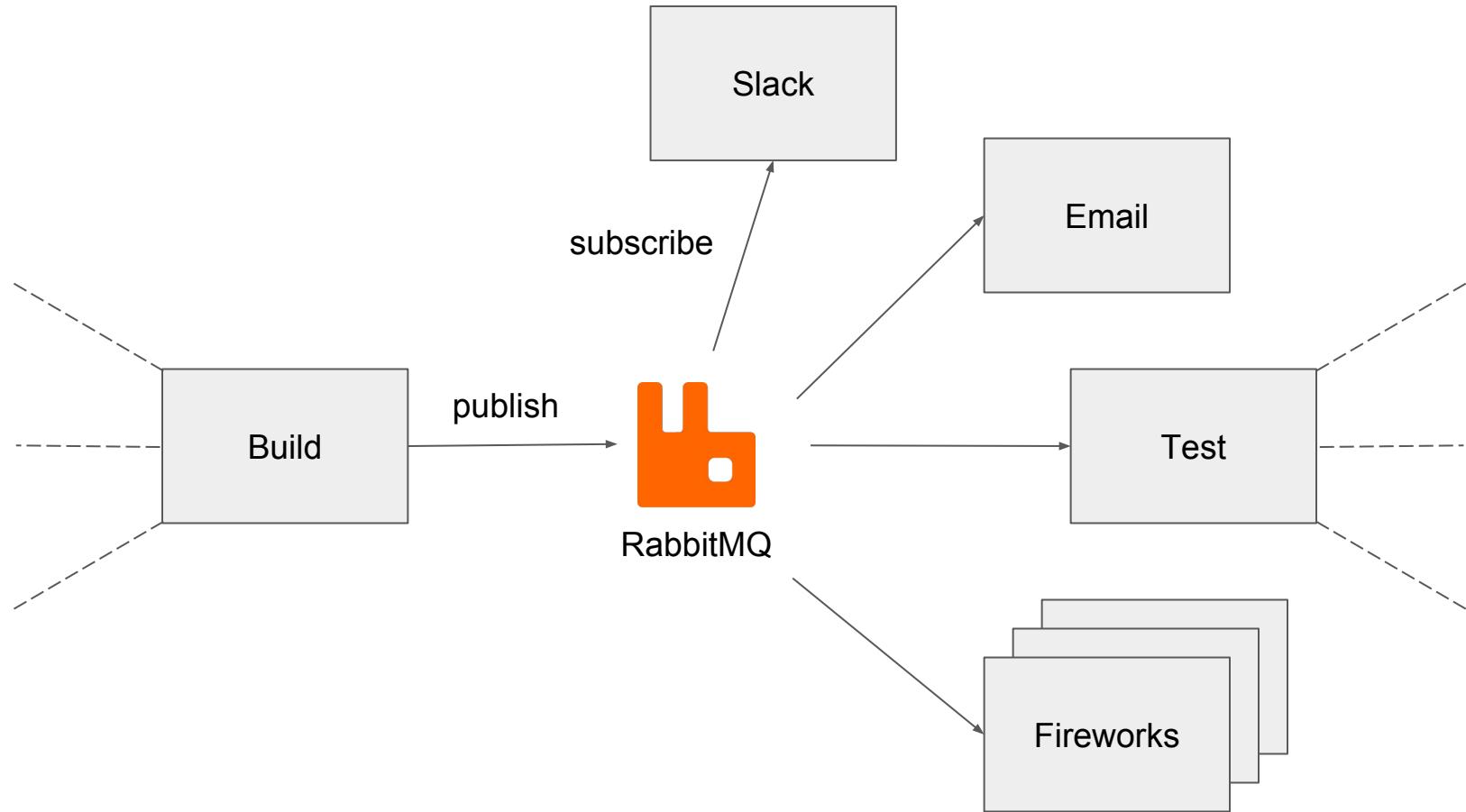
ActiveMQ



ØMQ







Time To Live

Time To Live

```
mq.publish(msg, {ttl: 10});
```

Time To Live

```
function put(msg, startTime) {  
    response = http.put(msg);  
    if (response.code != 200) {  
        if (getTime() - startTime < timeout) {  
            put(msg, startTime);  
        }  
    }  
  
    mq.publish(msg, {ttl: 10});  
}  
  
startTime = getTime();  
put(msg, startTime);
```

Time To Live

```
function put(msg, startTime) {  
    response = http.put(msg);  
    if (response.code == 404) {  
        // ???  
    } else if (response.code != 200) {  
        if (getTime() - startTime < timeout) {  
            put(msg, startTime);  
        }  
    }  
}  
  
mq.publish(msg, {ttl: 10});  
  
startTime = getTime();  
put(msg, startTime);
```

Time To Live

```
function put(msg, startTime) {  
    response = http.put(msg);  
    if (response.code == 404) {  
        // ???  
    } else if (response.code == 429) {  
        // ???  
    } else if (response.code == 500) {  
        // ???  
    } else if (response.code != 200) {  
        if (getTime() - startTime < timeout) {  
            put(msg, startTime);  
        }  
    }  
}  
  
startTime = getTime();  
put(msg, startTime);
```

Time To Live

```
function put(msg, startTime) {  
    response = http.put(msg);  
    if (response.code == 404) {  
        // ???  
    } else if (response.code == 429) {  
        // ???  
    } else if (response.code == 500) {  
        // ???  
    } else if (response.code != 200) {  
        if (getTime() - startTime < timeout) {  
            timeSpent = getTime() - startTime;  
            delay = Math.pow(timeSpent, 2);  
            setTimeout(put(msg, startTime), delay);  
        }  
    }  
}  
  
startTime = getTime();  
put(msg, startTime);
```

Time To Live

```
function put(msg, startTime) {  
    response = http.put(msg);  
    if (response.code == 404) {  
        // ???  
    } else if (response.code == 429) {  
        // ???  
    } else if (response.code == 500) {  
        // ???  
    } else if (response.code != 200) {  
        if (getTime() - startTime < timeout) {  
            timeSpent = getTime() - startTime;  
            delay = Math.pow(timeSpent, 2);  
            remainingTime = timeout - timeSpent;  
            delay = Math.min(delay, remainingTime);  
            setTimeout(put(msg, startTime), delay);  
        }  
    }  
}  
  
startTime = getTime();  
put(msg, startTime);
```

Time To Live

```
function put(msg, startTime) {
    response = http.put(msg);
    if (response.code == 404) {
        // ???
    } else if (response.code == 429) {
        // ???
    } else if (response.code == 500) {
        // ???
    } else if (response.code != 200) {
        if (getTime() - startTime < timeout) {
            timeSpent = getTime() - startTime;
            fuzz = Math.floor(Math.random() * 10);
            delay = Math.pow(timeSpent, 2) + fuzz;
            remainingTime = timeout - timeSpent;
            delay = Math.min(delay, remainingTime);
            setTimeout(put(msg, startTime), delay);
        }
    }
}

startTime = getTime();
put(msg, startTime);
```

Quality of Service

Quality of Service

```
// Best effort, fire and forget
mq.publish(unimportantMessage, {qos: 0});
```

Quality of Service

```
// Best effort, fire and forget
mq.publish(unimportantMessage, {qos: 0});
```

```
// At least once delivery
mq.publish(idempotentMessage, {qos: 1});
```

Quality of Service

```
// Best effort, fire and forget
mq.publish(unimportantMessage, {qos: 0});
```

```
// At least once delivery
mq.publish(idempotentMessage, {qos: 1});
```

```
// Exactly once delivery
mq.publish(importantMessage, {qos: 2});
```

Concurrency

Concurrency

Upload text → Generate video → Post video

Concurrency

Fast

Fast

Upload text → Generate video → Post video

Concurrency

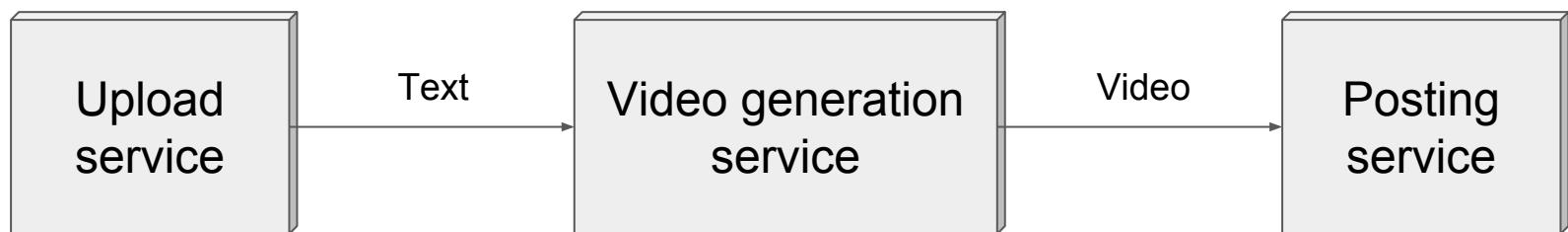
Fast

Slow

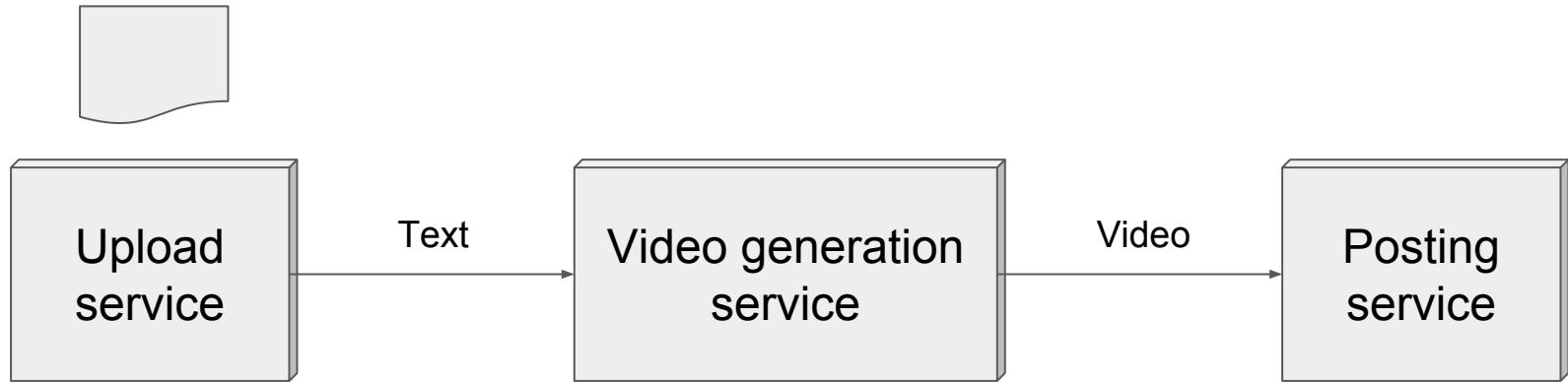
Fast

Upload text → Generate video → Post video

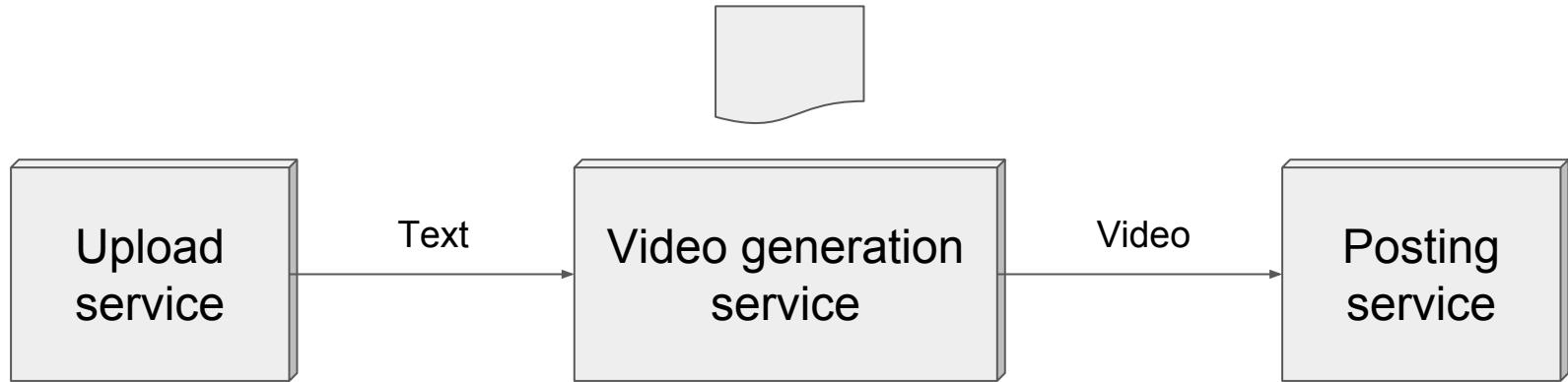
Concurrency



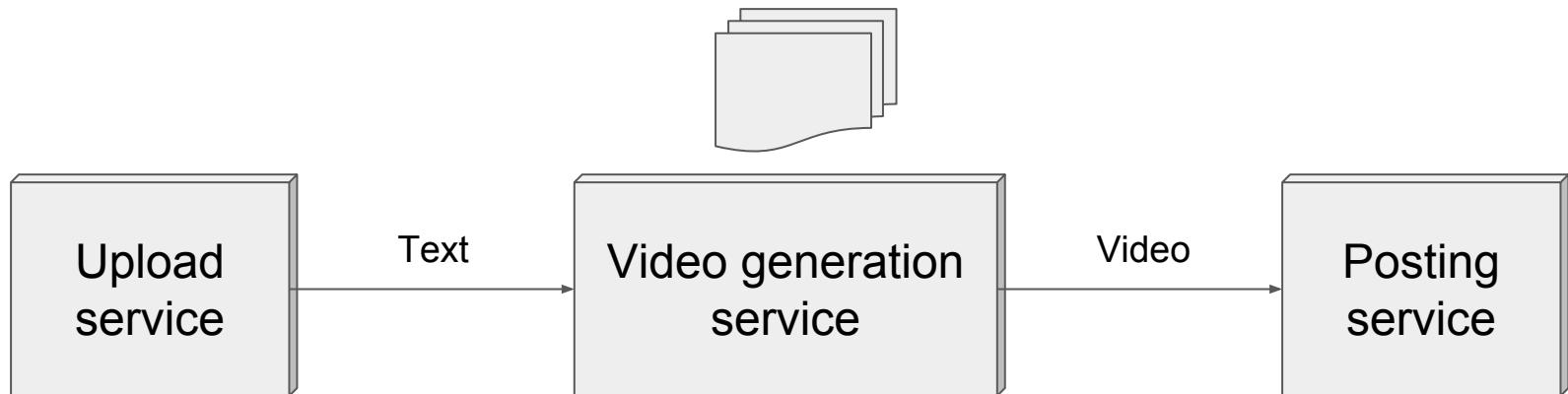
Concurrency



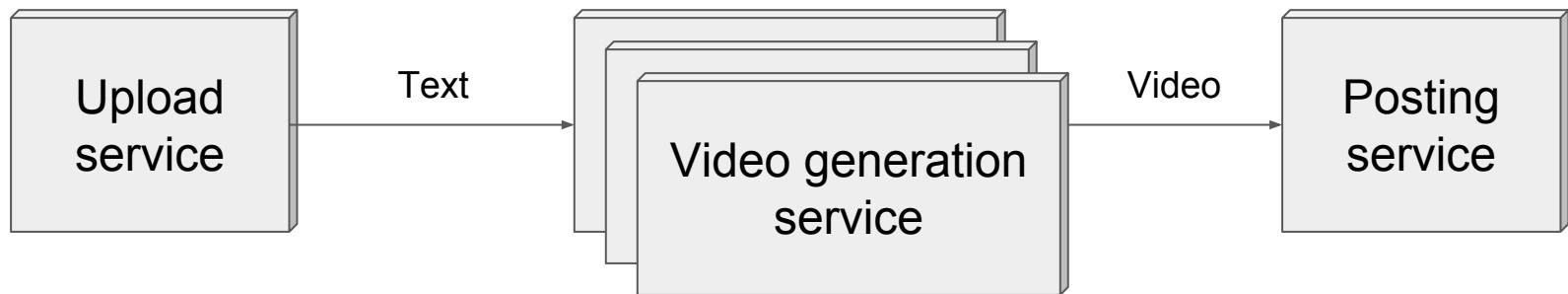
Concurrency



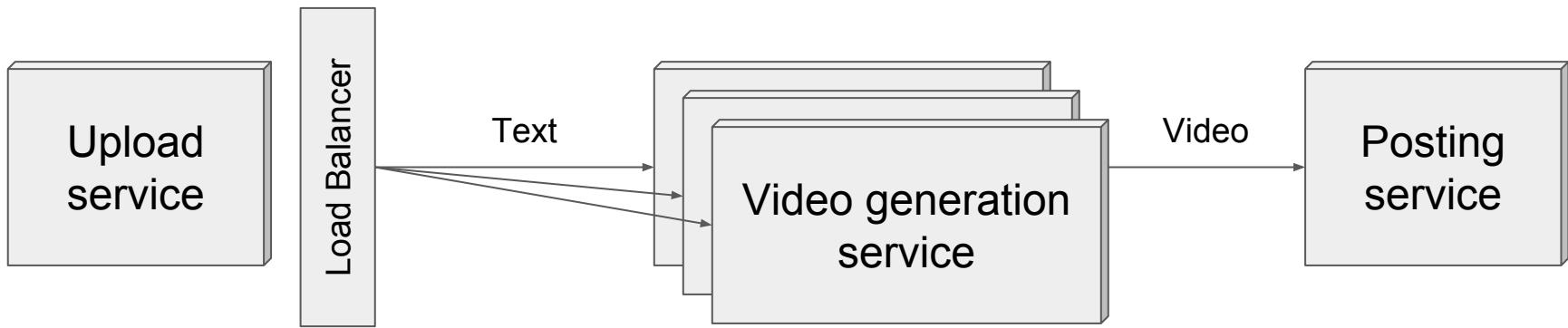
Concurrency



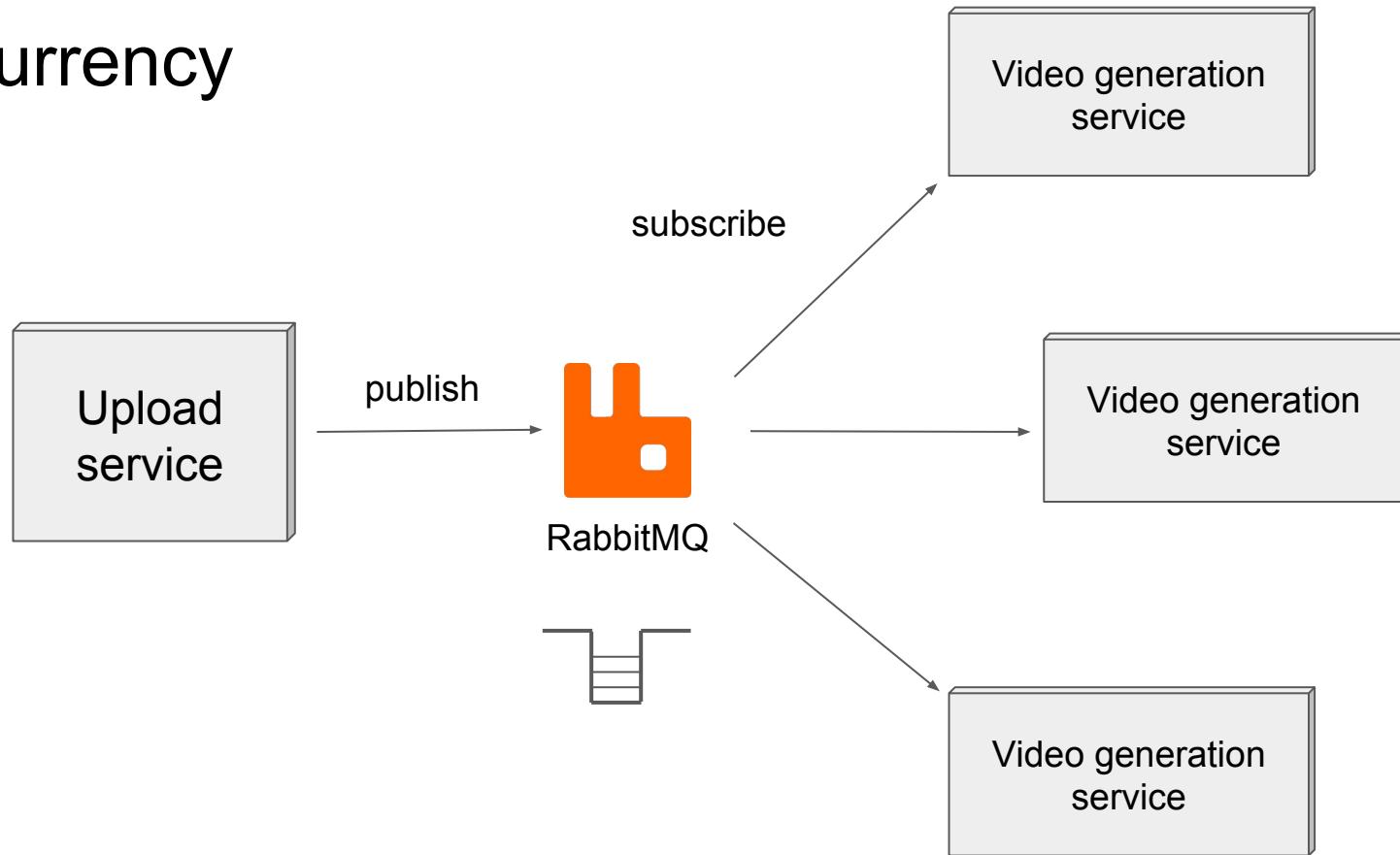
Concurrency



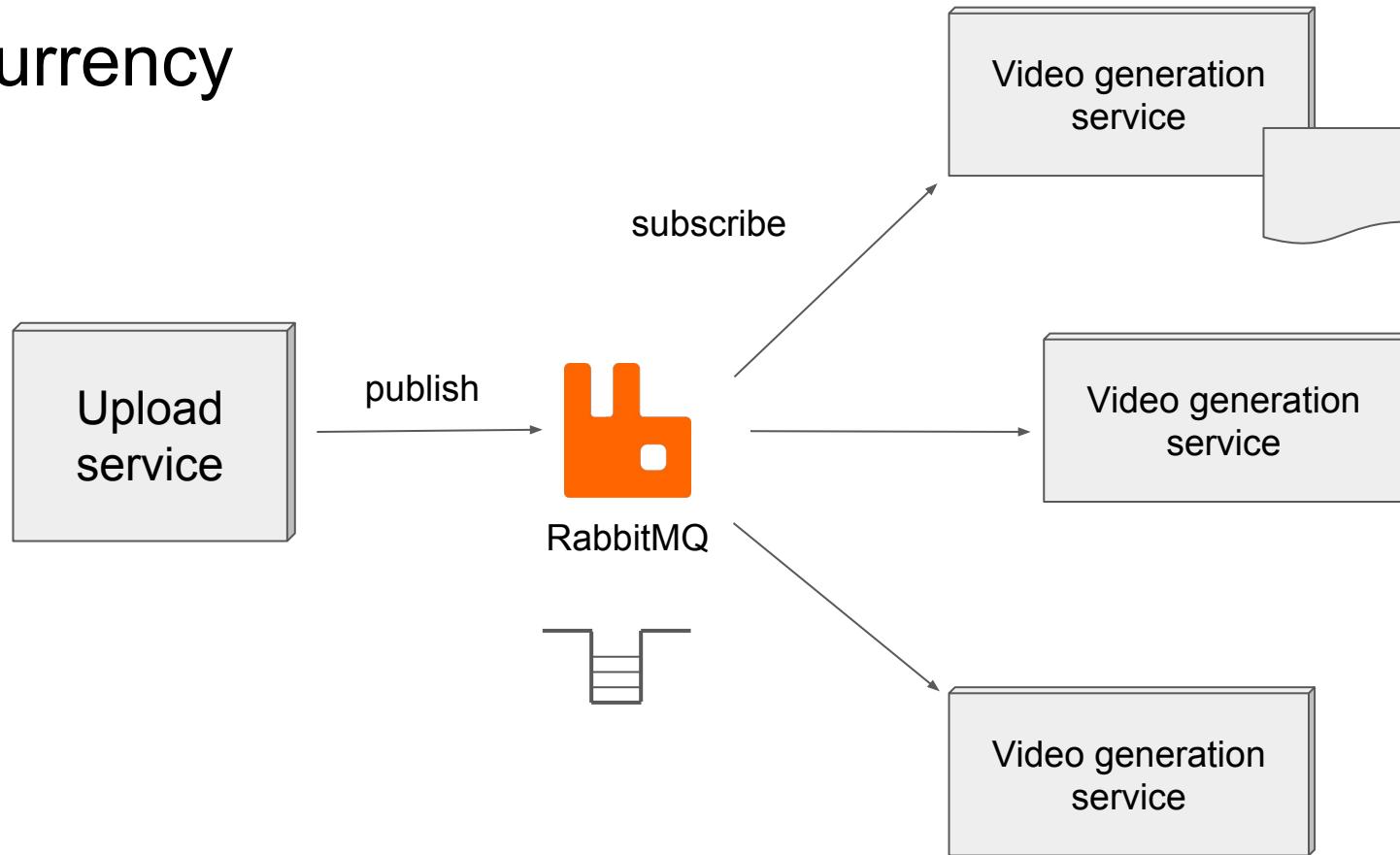
Concurrency



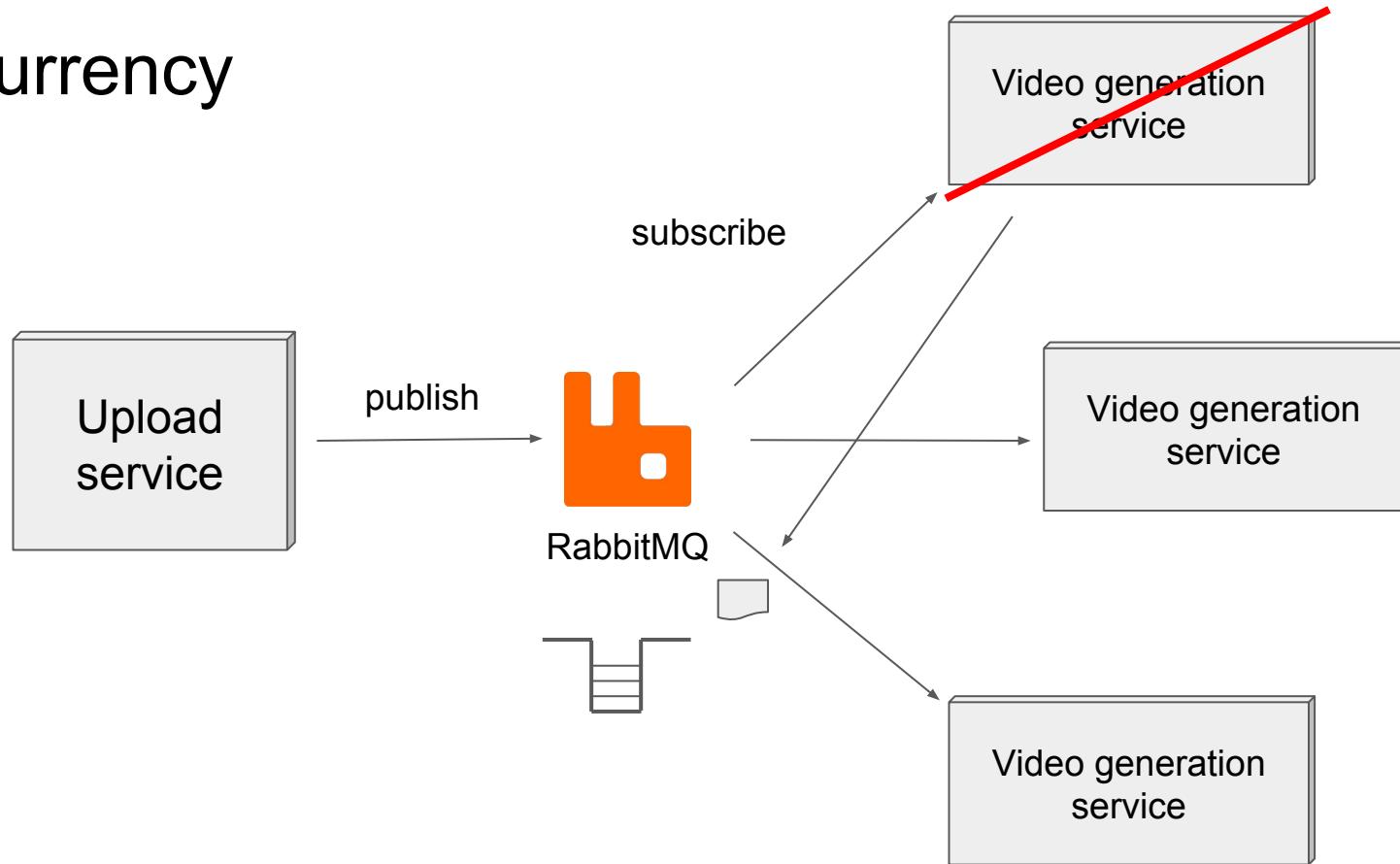
Concurrency



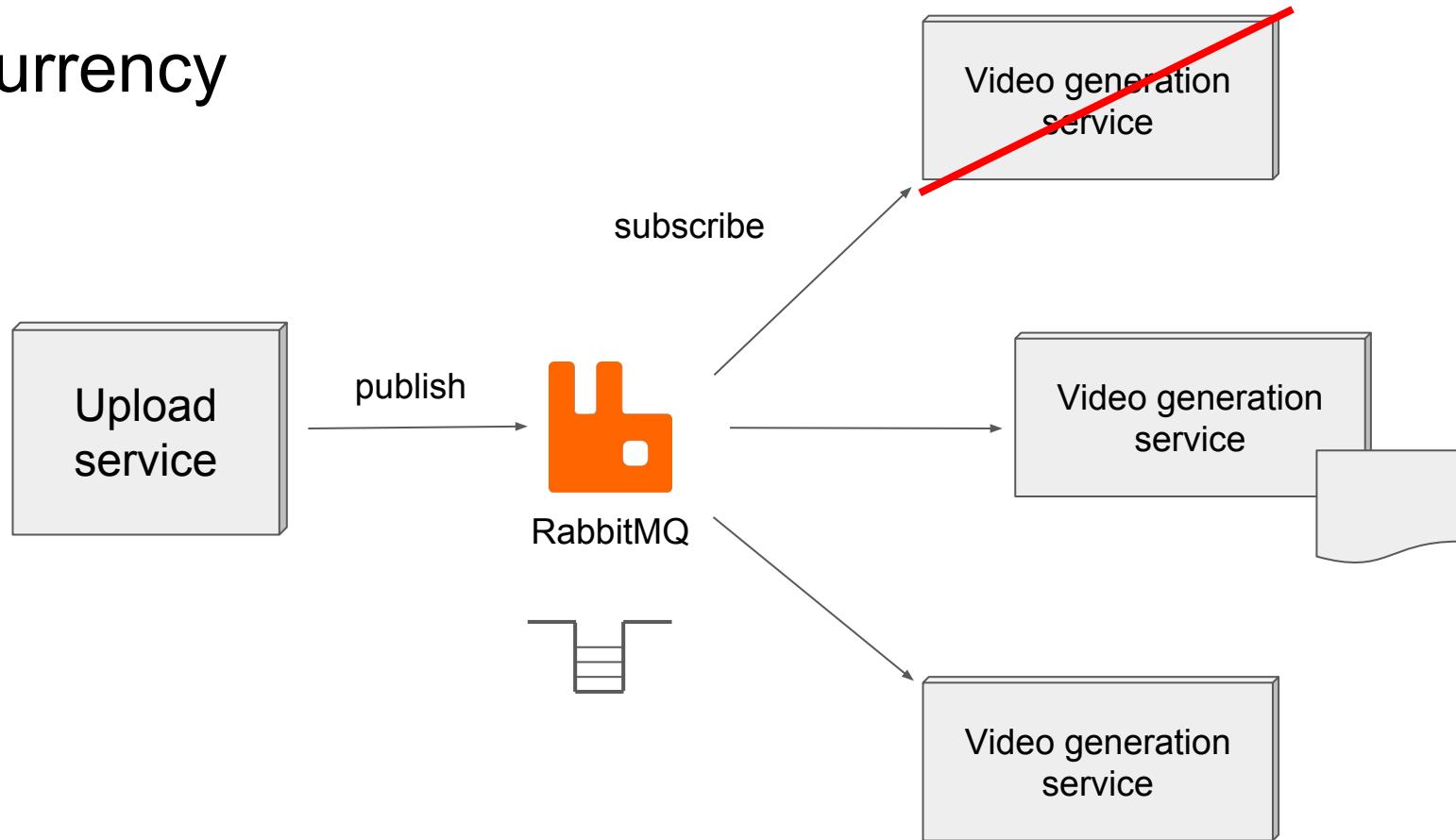
Concurrency



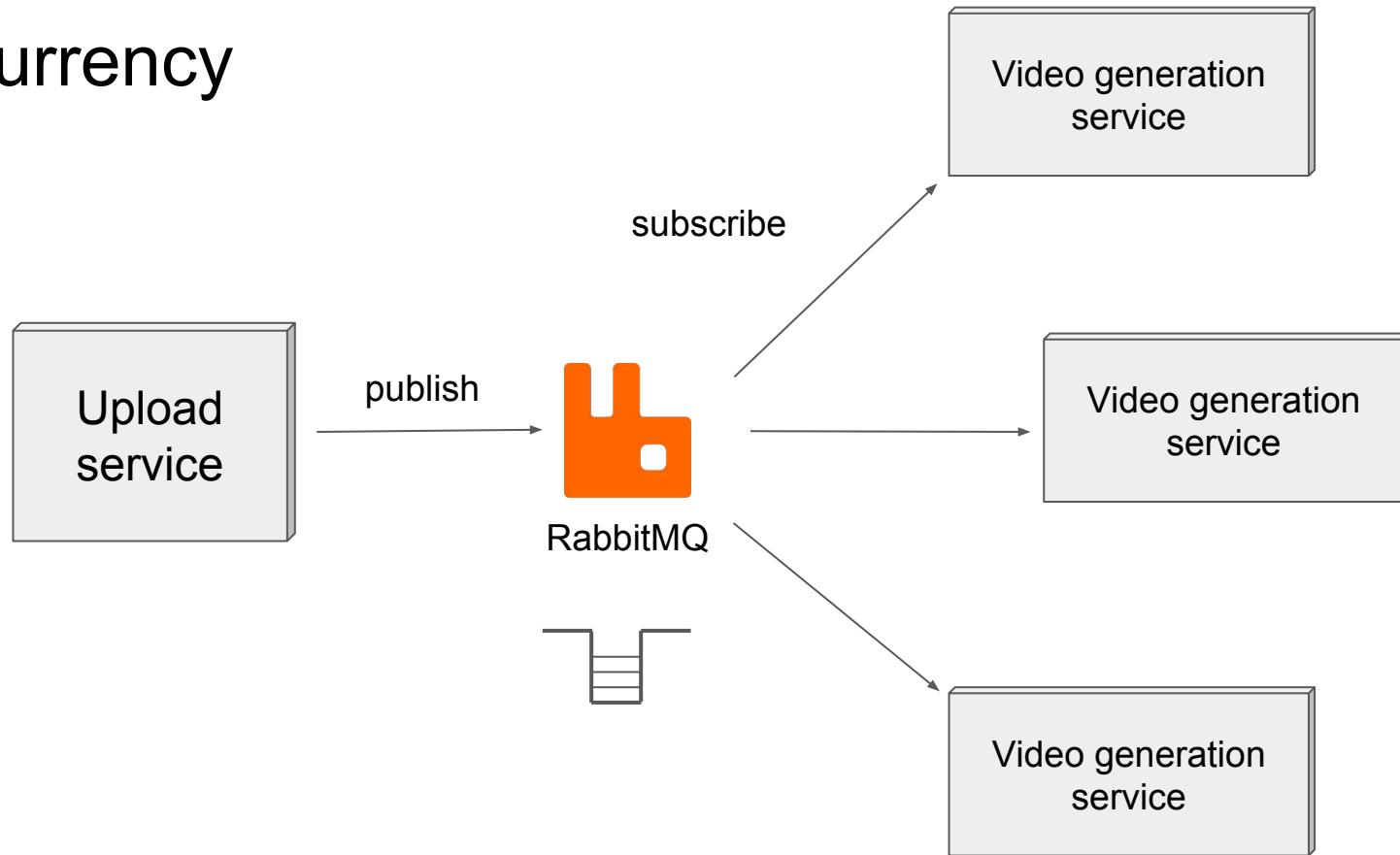
Concurrency



Concurrency



Concurrency



“Publish everything!”
Tom Livesey, Droplet

Build FAILED
Buildosaurus ANGRY

MQ Buildosaurus

Exercise



Lots of data sources



Lots of data sources

Lots of data consumers



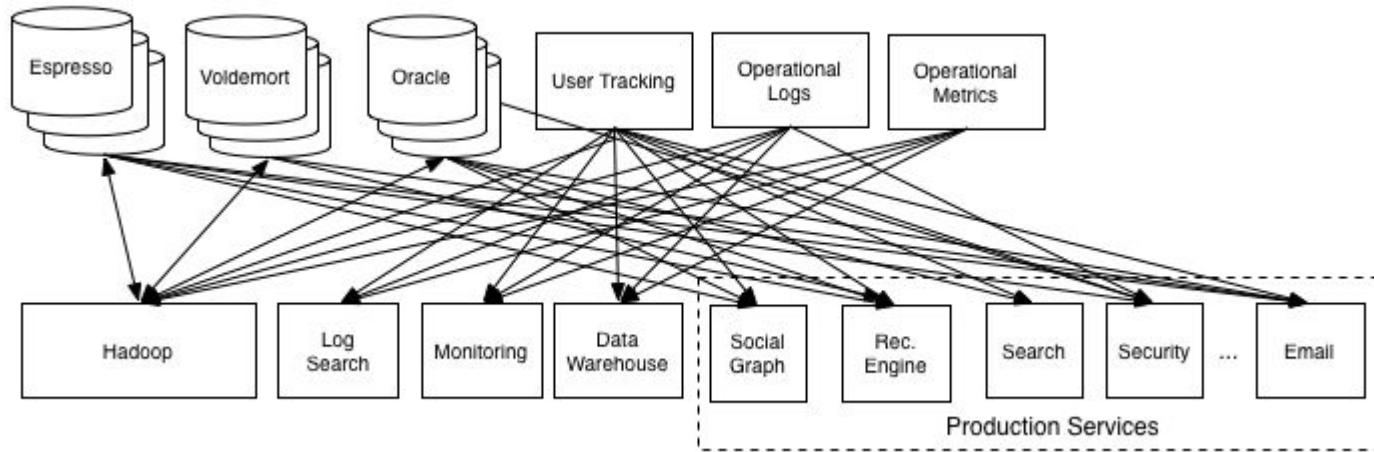
Lots of data sources

Lots of data consumers

Lots of integrations

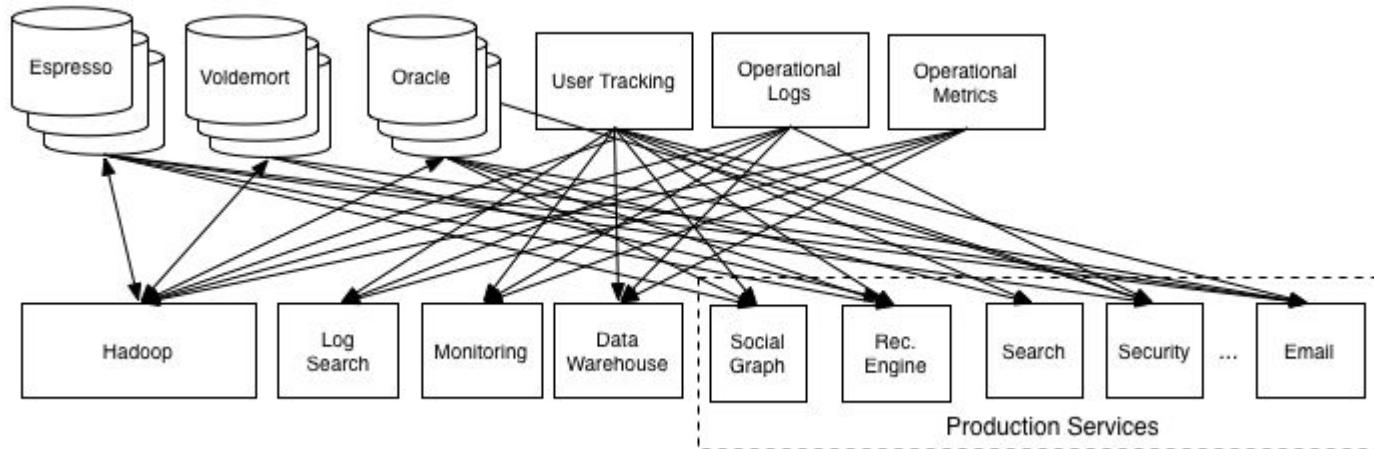


LinkedIn before

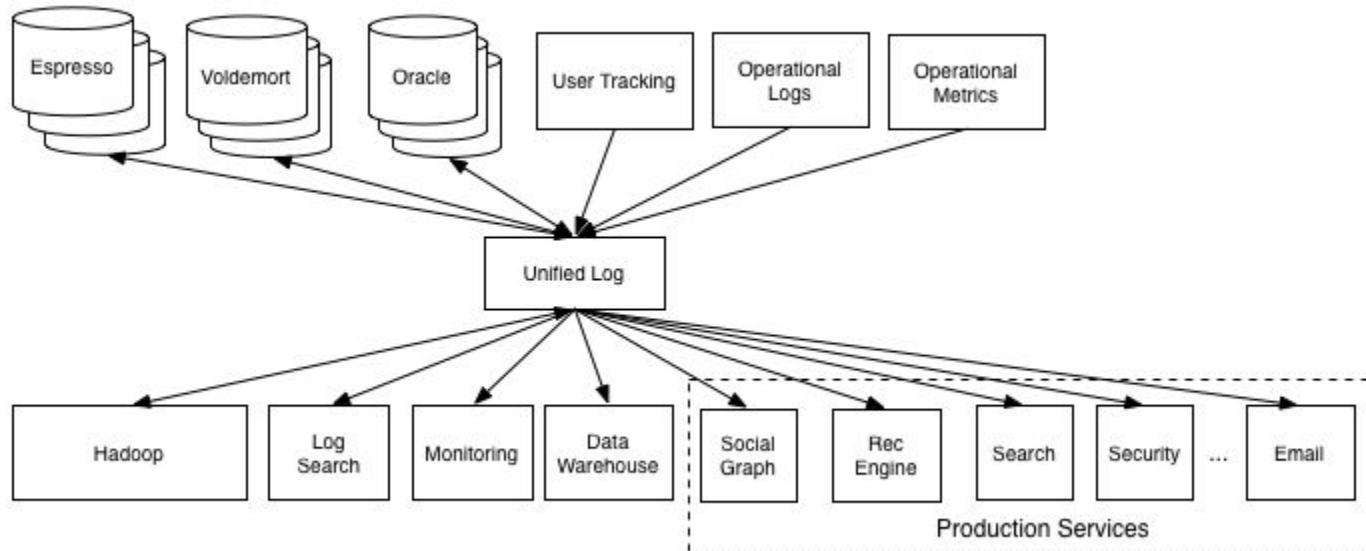


LinkedIn before

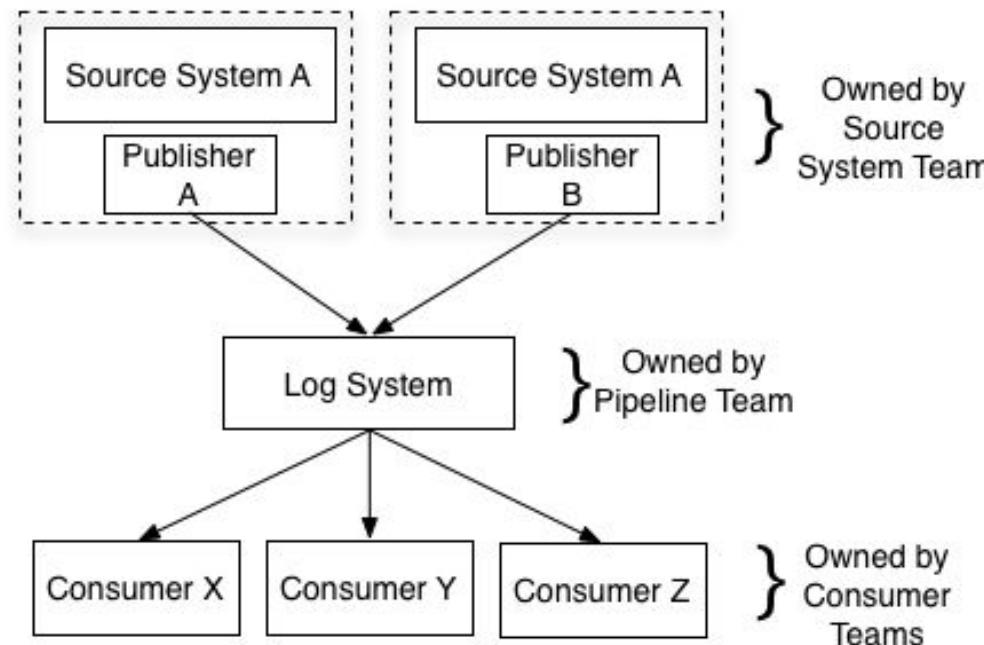
$O(n^2)$



LinkedIn after



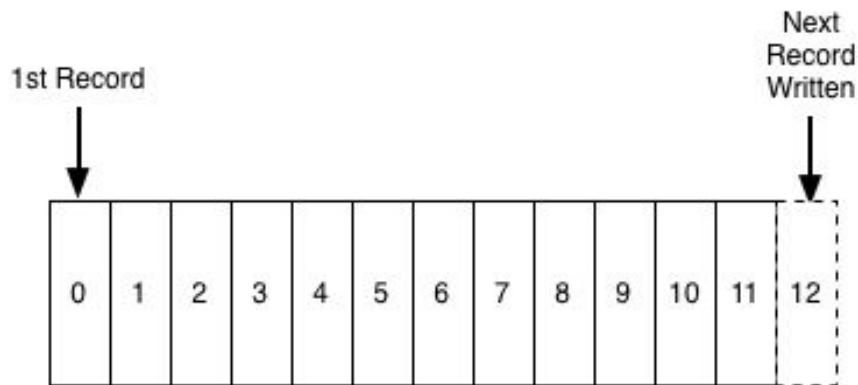
Up close



The log

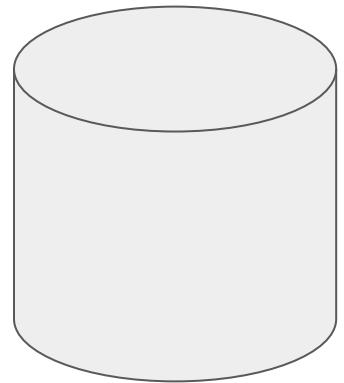
Append only

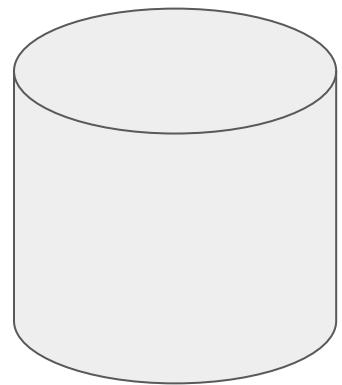
Time ordered



Distributed logging





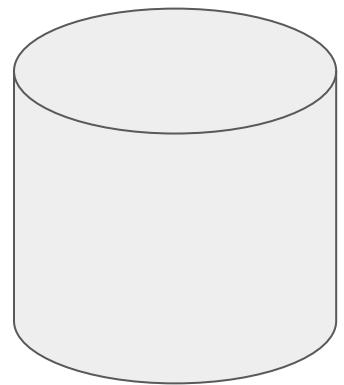


add(20)

add(5)

subtract(10)

add(15)



add(20)



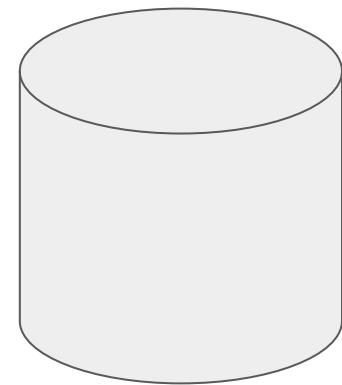
add(5)

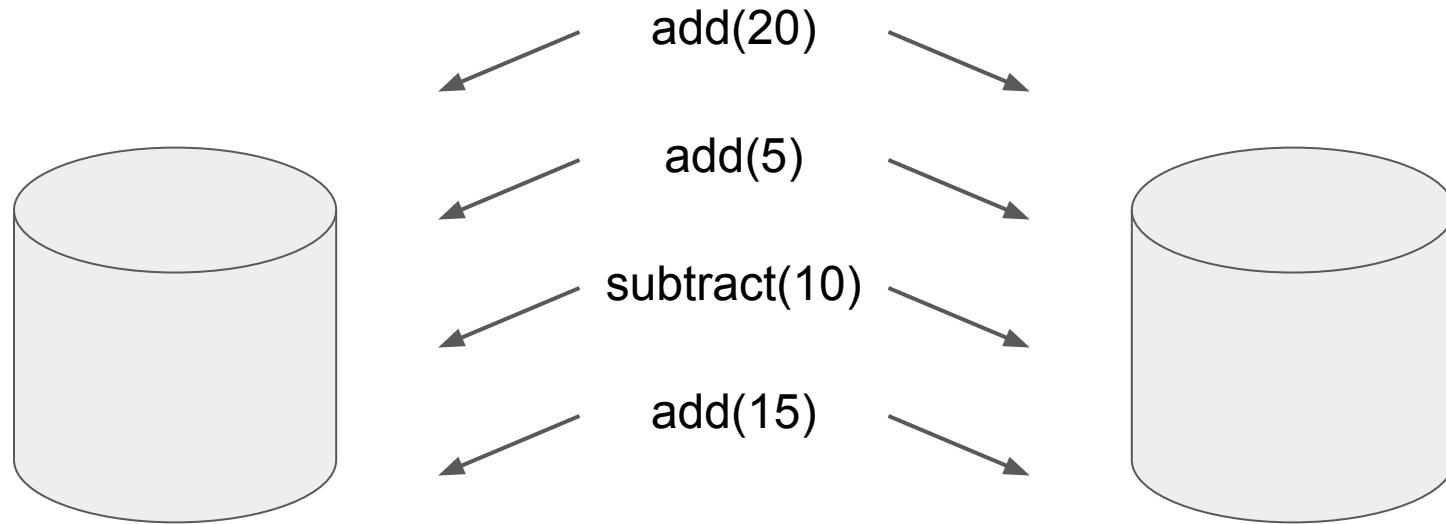


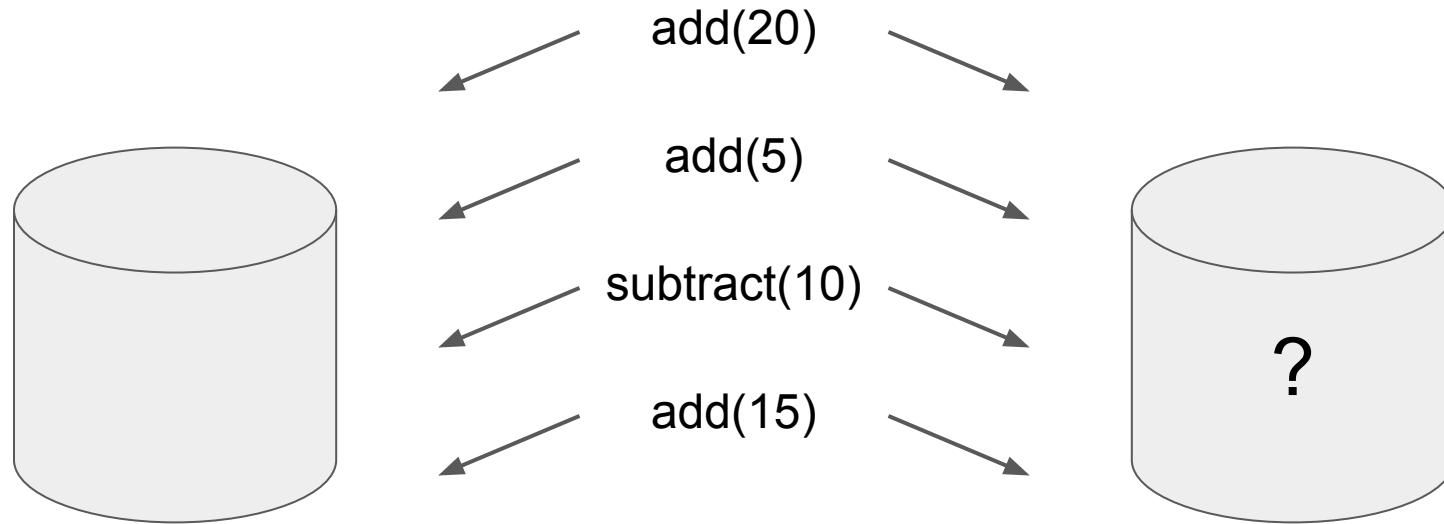
subtract(10)



add(15)





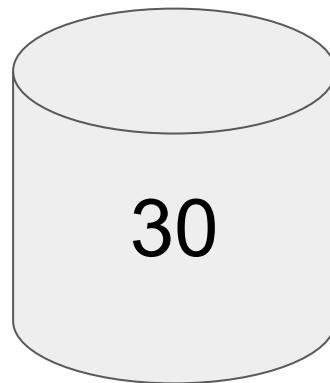


“If two identical, deterministic processes begin in the same state and get the same inputs in the same order, they will produce the same output and end in the same state.”

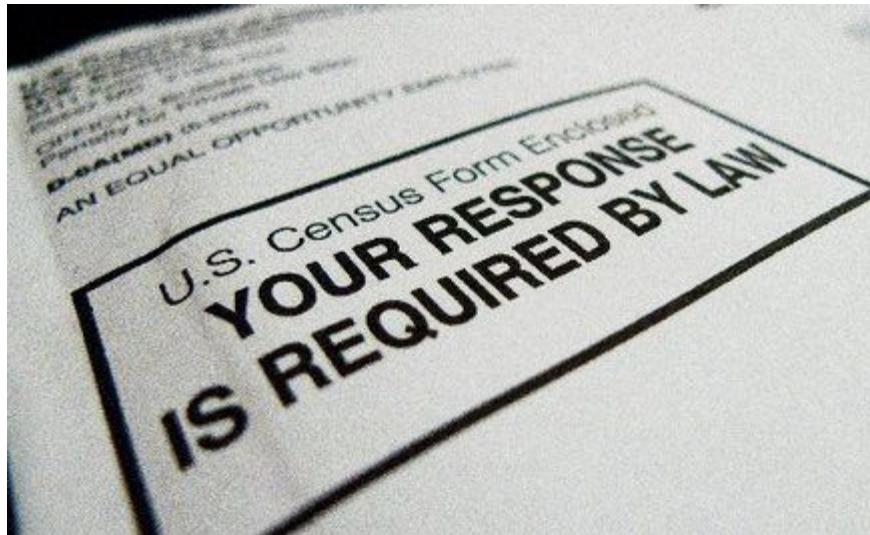
State Machine Replication Principle, Jay Kreps, LinkedIn

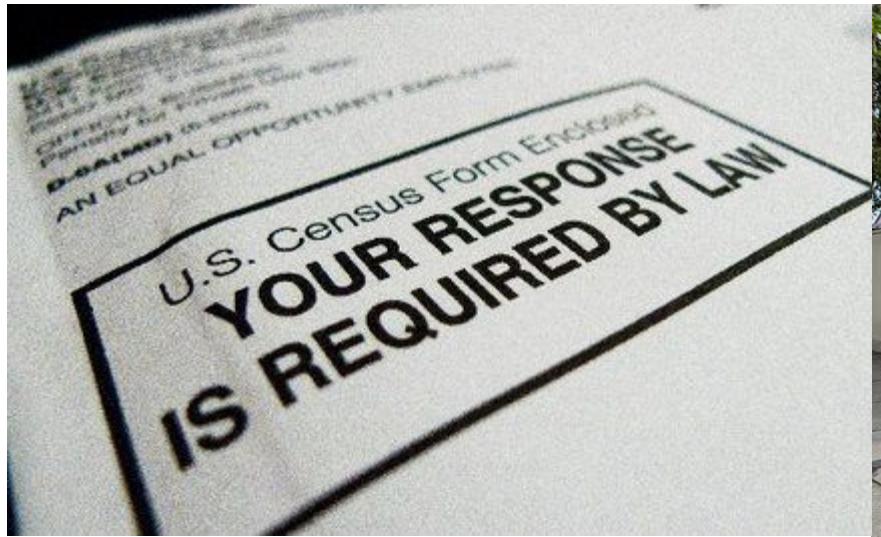
add(20)
add(5)
subtract(10)
add(15)

?



Log as source of Truth

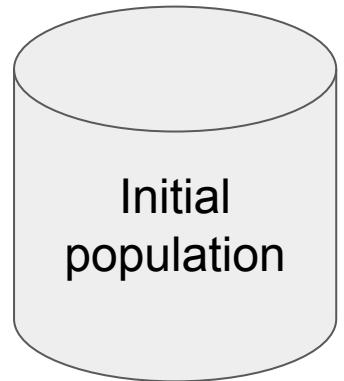




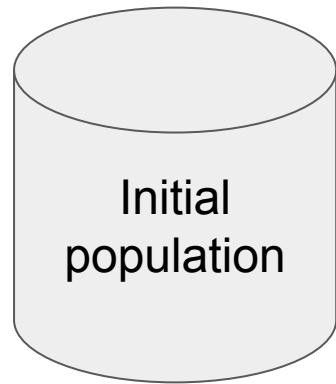
2010 RESIDENT POPULATION



308,745,538



Initial
population

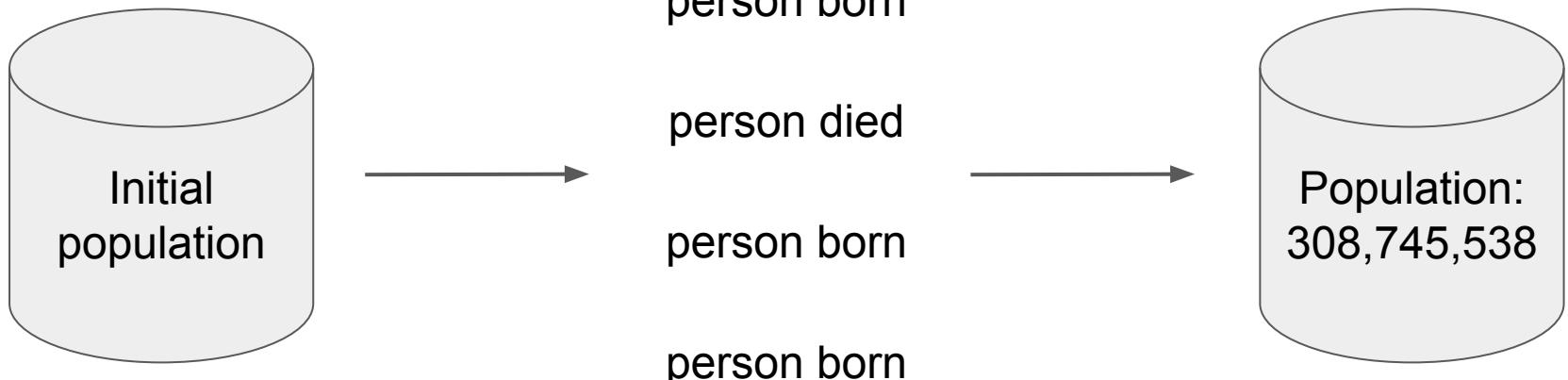


person born

person died

person born

person born





Summa de Arithmetica geo

metria. Proportioni: et proportionalita:
Monumentale impressa In Lofcolano su la riva vlt Benacense et
vnuco carpionito Laco: Amensimmo Sitos de li antique: &
evidenti ruine ol la nobil cit Benaco vita illustra:
to: Cum numerosta de Empatoris epiphaphis
di antique e perfette littere scalpiti vo:
tato: e cui finissimi e mirabil co:
lone marmozzi numeri
fragimenti di alaba:
stro porphyri e serpentini. Loco certo
letto suo sijgno occultato:
de mirata digne fote. Fil. Dr. OISTAR BENESTRO:
terra se ritro: BIBLIOTEK
HARO. STOCKHOLMS HOGSKOLA.

Continuentia de tutta lopera:

De numeri e misure in tutti modi
occurenti.
Proportioni e proportionalita notissima
vol 5: de Euclide e de tutti li altri
sui libri.
Quia unico euidente numero. 13. per
le quantita continue proportionali del
6: e 7: de Euclide extracta.
Tutte le parti de l'algoismo: cioè relesa:
re partem multiplicare: summare: e so:
trarre con tutte sue pag in santi e rotti
e radici e progreffioni.
De le regole mercantefca vita del. 3: e
sui fondamenti co' casi creplari p' m:
8: e guadagni per difteri transpotatio:
ni: e inuestite.
Partiri: multiplicare: summare: e sottrarre
le proportioni: e de tutte sotti radici.
De le tre regole del Catay ditta posi:
tione: e sua origine.
Quidientie generali: ouer conclusioni nu:
mero. 66. ab soluere ogni caso che per
regole ordinarie non si podesse.
Tutte sorte binomii: e rectifici: e altre linee
irrationali del decimo de Euclide.
Tutte regole de Algebra ditte de la colla

e lo fabrichi e fondamenti.
E spoglie in tutti modi: e le parti.
Socde de bestiam: e le parti.
Fatti: pecioni: cotti: melli: logazioni:
e godimenti.
Varatti in tutti modi semplici: compo:
site col tempo.
Lambi: real: sechi: fritti: e cipollini:
ouer communi.
Termini
Adritti semplici: e a capo: vanno: e altri
Rifletti: adi: contide: tempo: e venars: e de:
recare a via di piu partire.
Quargentis: ellos: affirmare: e corattare.
Abolti: caff: e ragioni: straordinarie: va:
rice: diuersitate: a tutto occurrerete: come
nella sequente taula appare: ordinamente
de tutte.
Ordine a faser tener ogni cosa scriptu:
re del quaderno in vinegia.
Tutte sorti: vianze: e costumi mer:
cantefci in tutto il mondo.
Pecanica: e theoria de geometria: e de li
cinque corpi regulari: e altri dependenti
E molte altre cose de grandissimi piace:
rie frutto: comino costitualmente per
la sequente taula appare.



BLOCKCHAIN





2 million writes s⁻¹
(on 3 cheap machines)

10s of millions of writes s⁻¹
(LinkedIn production)



2 million writes s⁻¹
(on 3 cheap machines)

10s of millions of writes s⁻¹
(LinkedIn production)

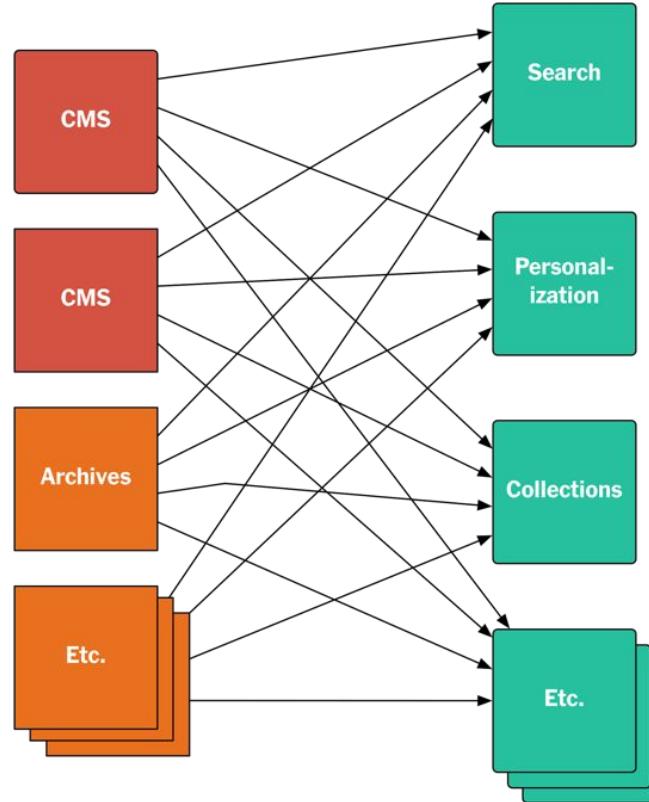
7 transactions s⁻¹
(Bitcoin network)

Log as source of Truth

Event Sourcing

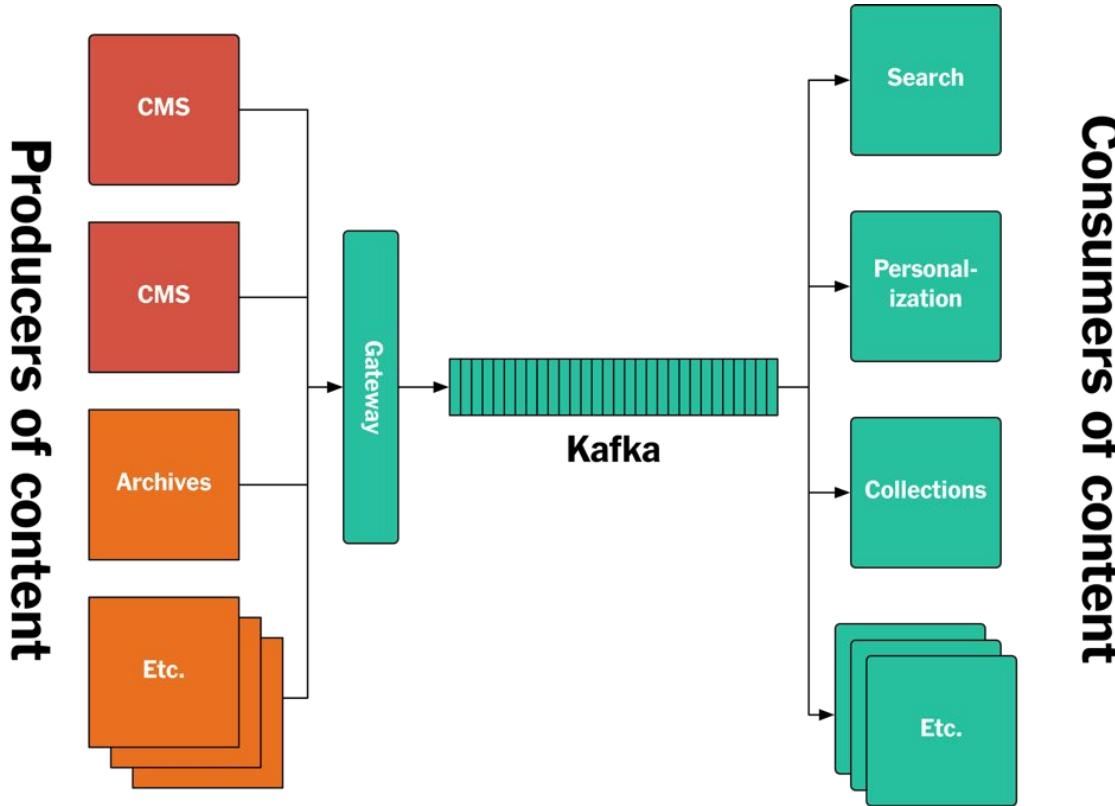
New York Times

Producers of content



Consumers of content

New York Times



New York Times Monolog

Section 1

Byline 1

Tag 1

Tag 2

Image 1

Image 2

Image 3

Section 2

Article 2

Article 1

Image 2, version 2

Tag 2, version 2

“We don’t care if our production system crashes.”

Valerii Vasylkov, William Hill



Full-fledged stream
processing framework

Build an app that consumes
from Kafka directly

Full-fledged stream
processing framework



Build an app that consumes
from Kafka directly



Data science. **Analytics**
about the business

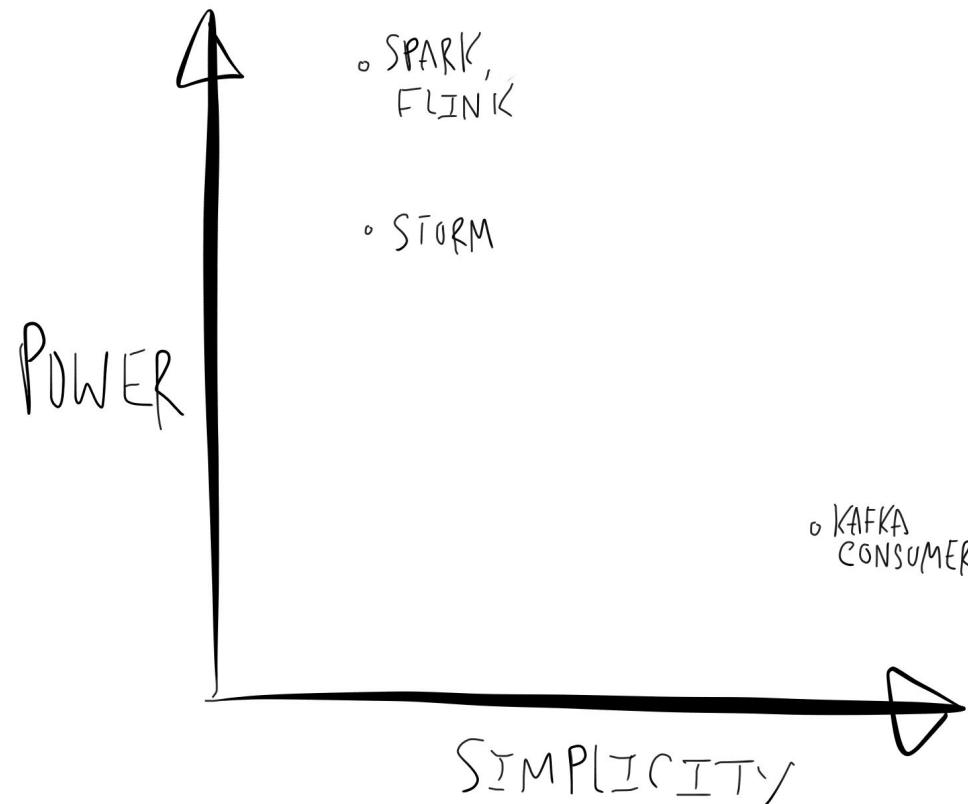
Full-fledged stream
processing framework

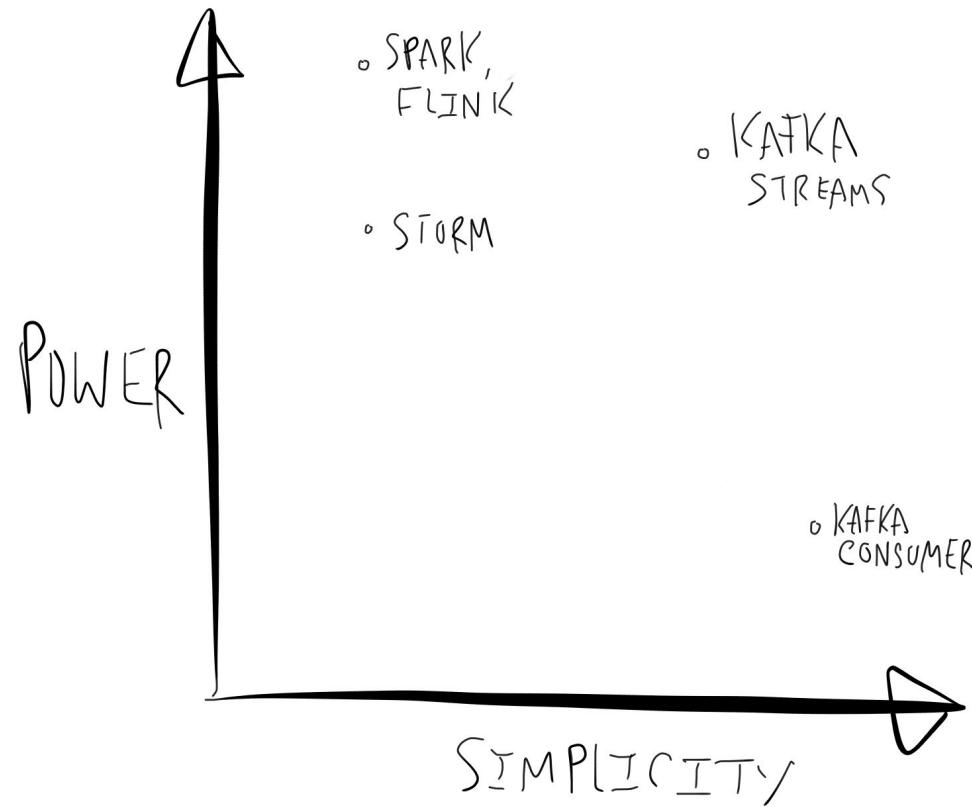


Microservice like. **Core**
business functions

Build an app that consumes
from Kafka directly



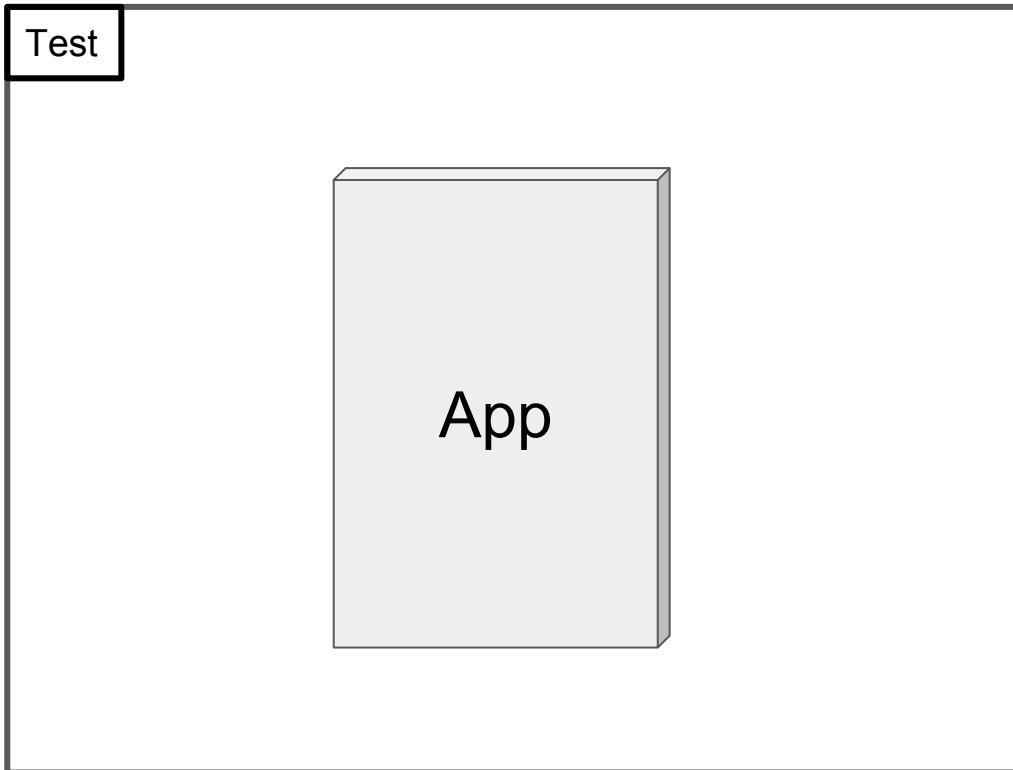




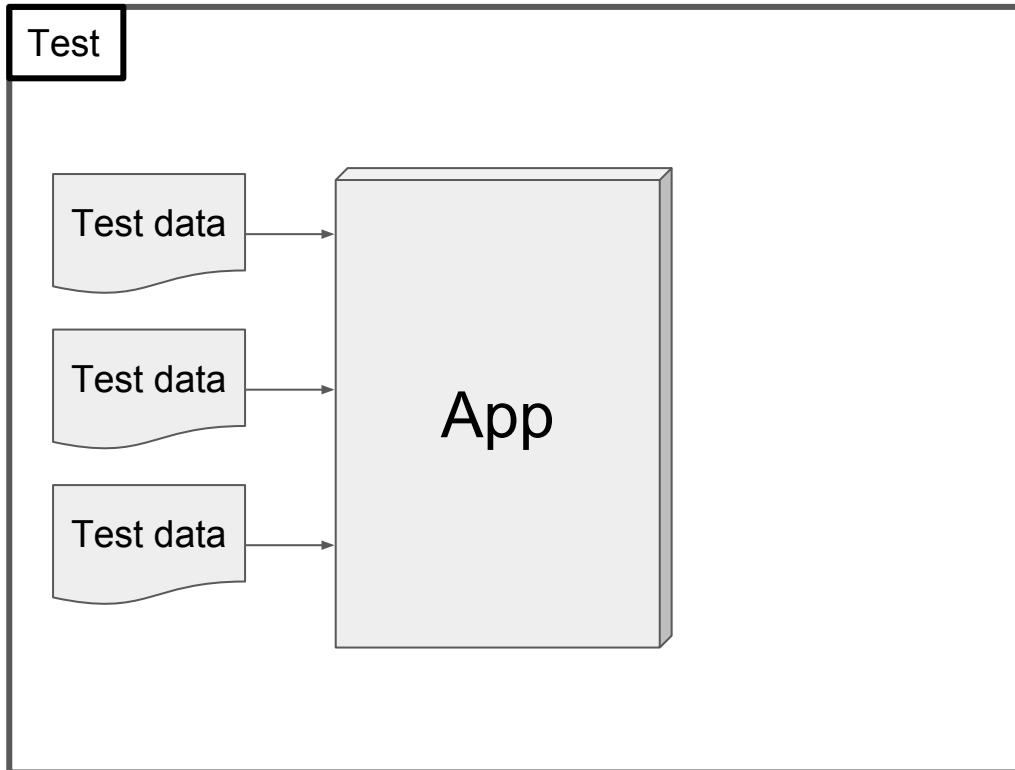
Testing Microservices

Monolithic testing

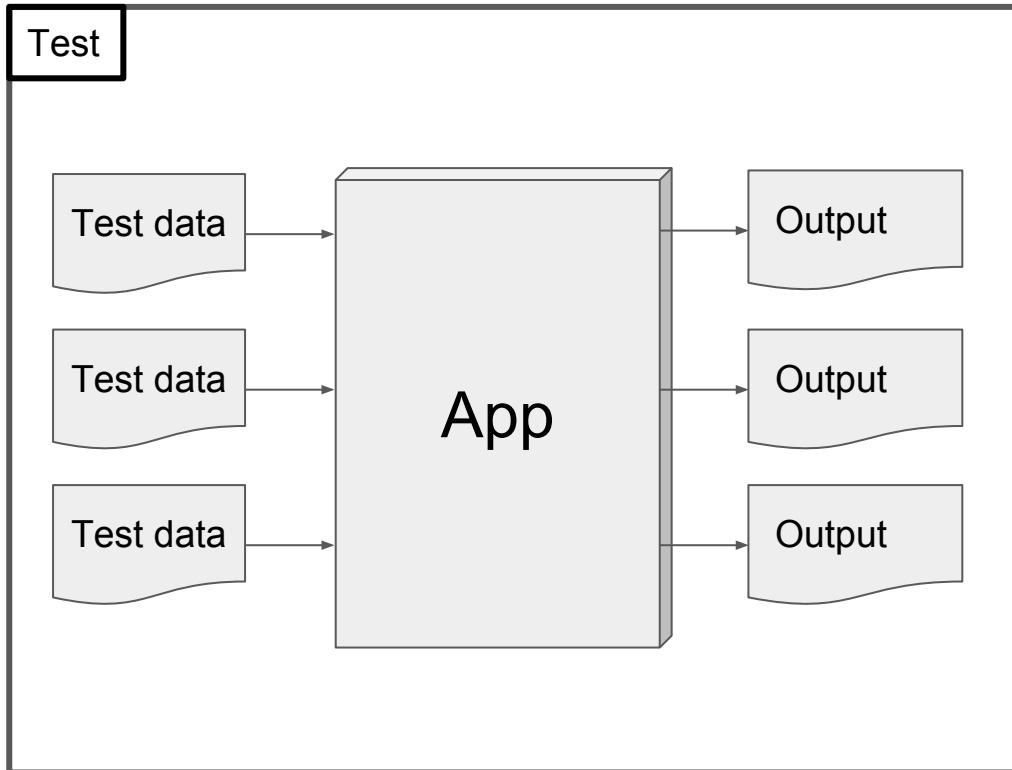
Monolithic testing



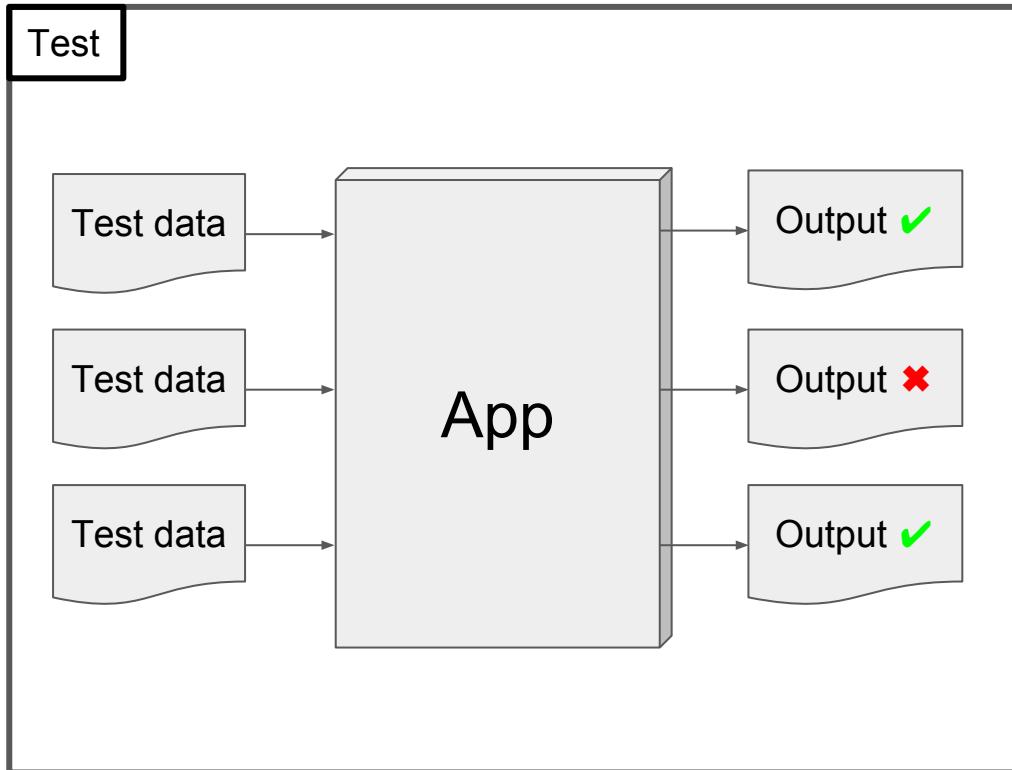
Monolithic testing



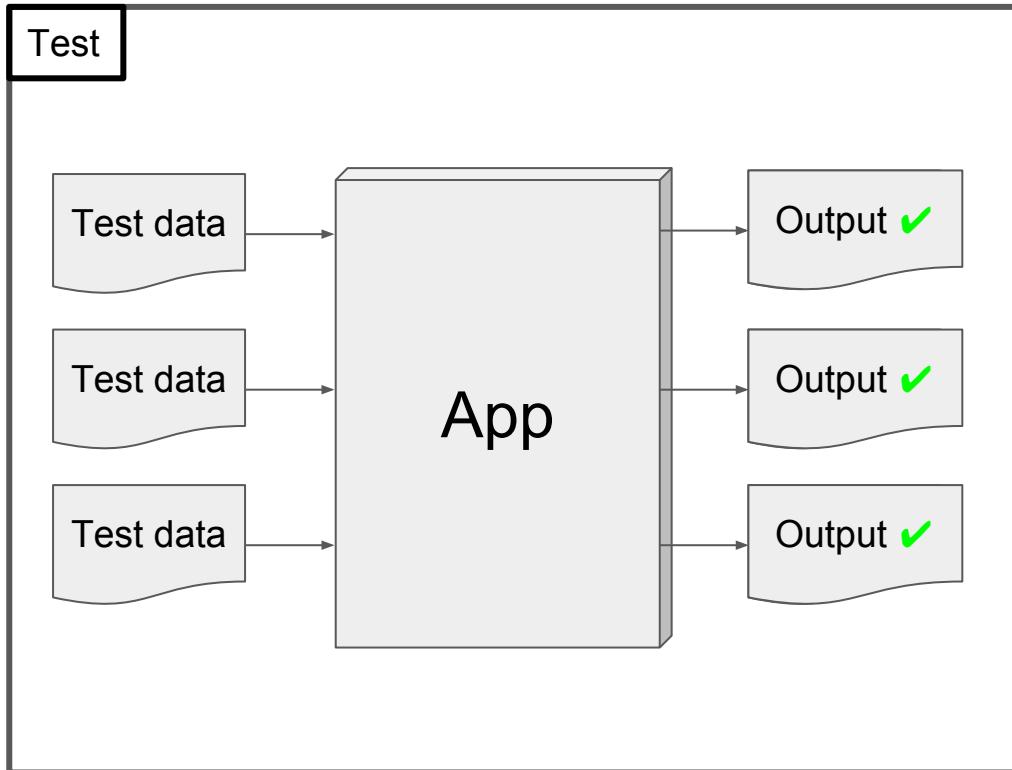
Monolithic testing



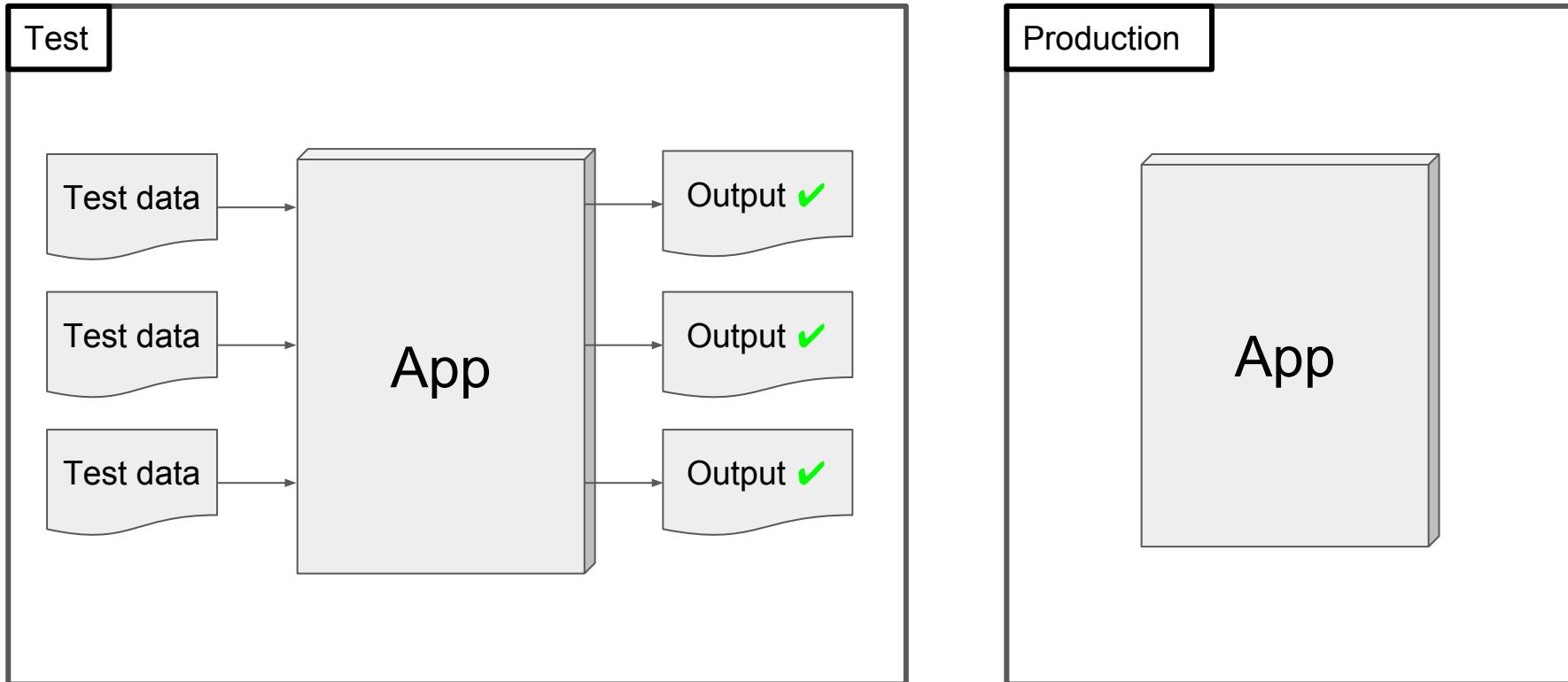
Monolithic testing



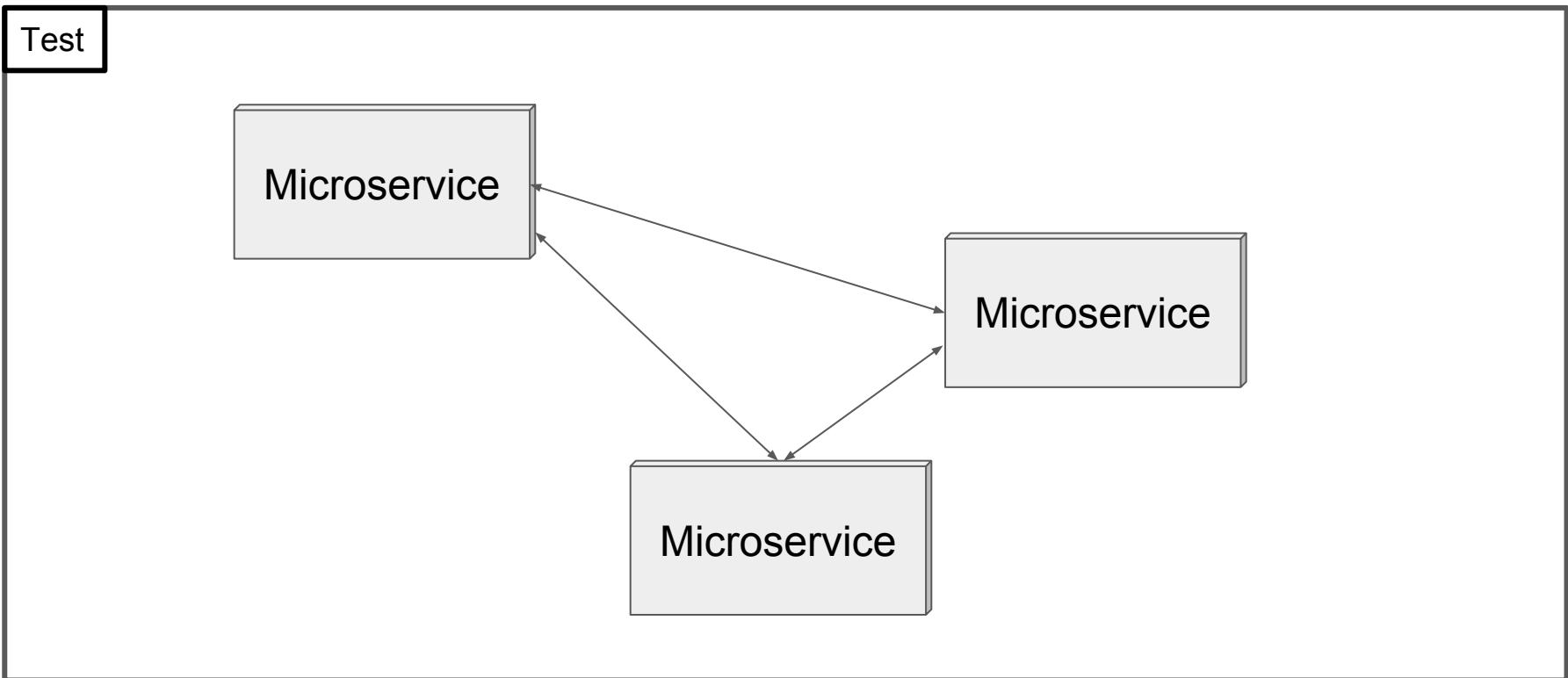
Monolithic testing



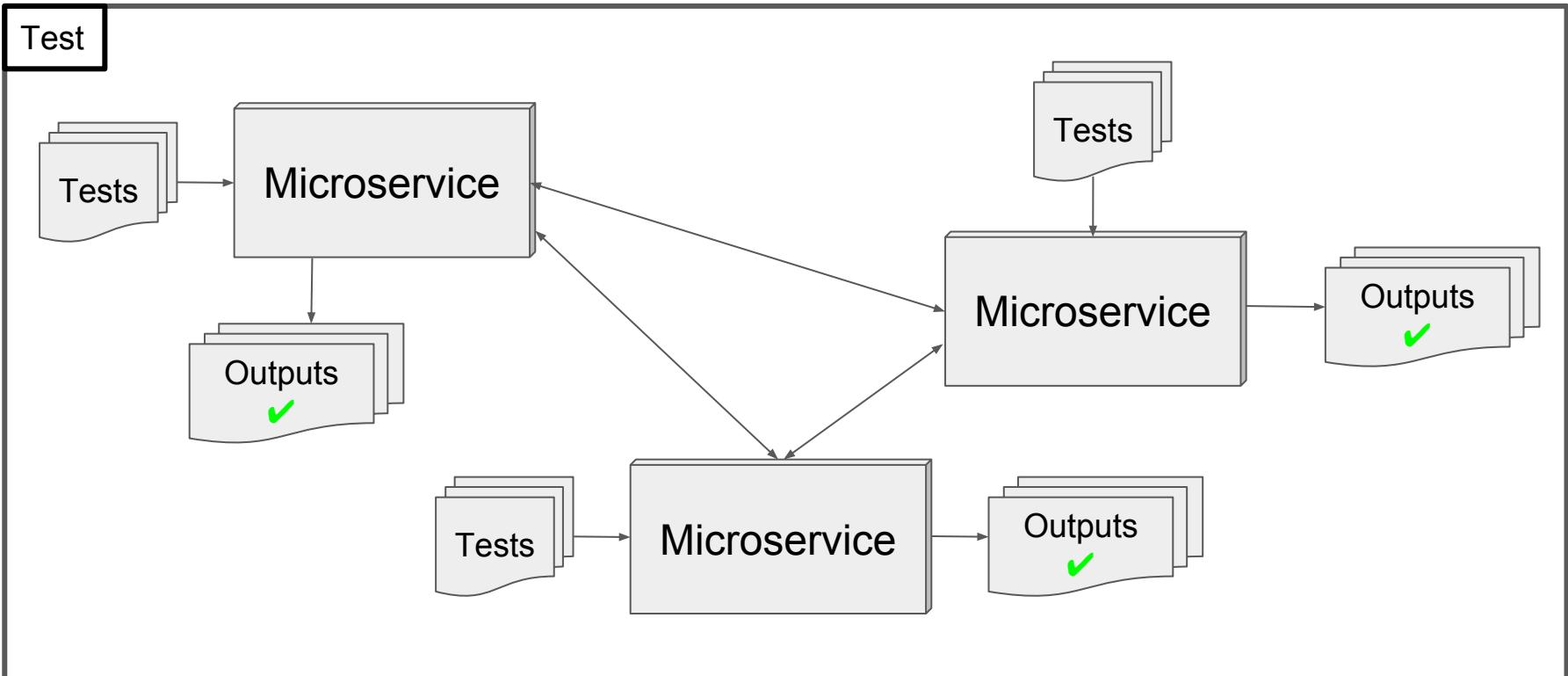
Monolithic testing



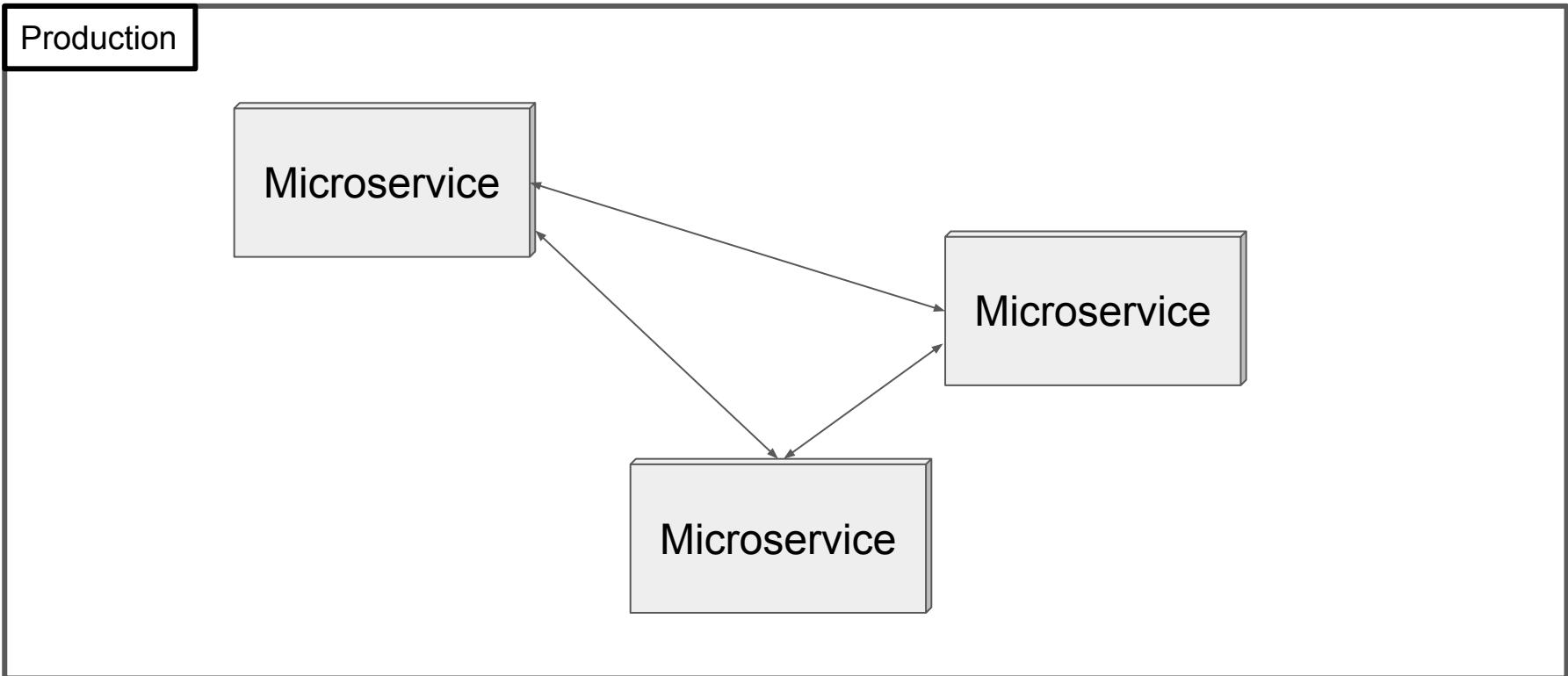
Microservice testing



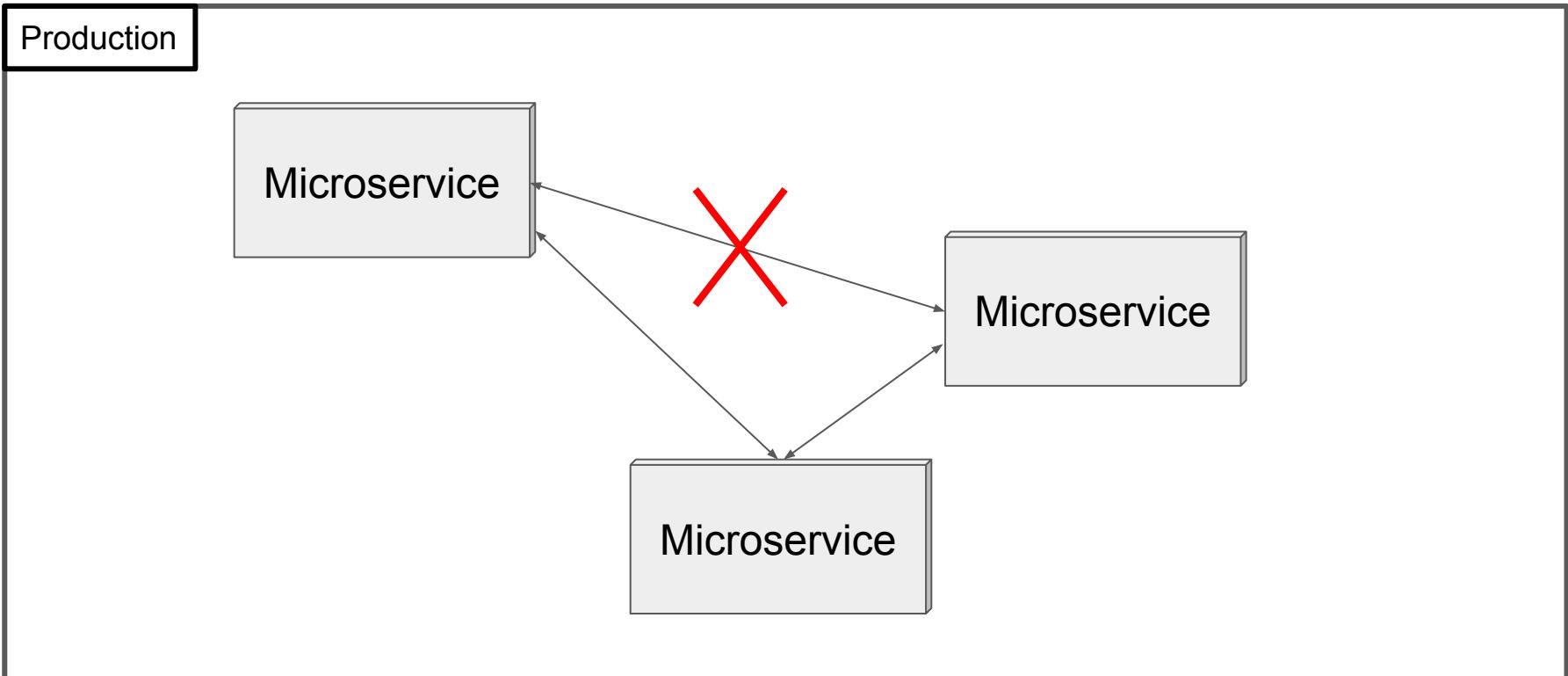
Microservice testing



Microservice testing



Microservice testing



Monitoring = Testing



DATADOG



SENTRY

application_ID: application_1443599818521_3111 -



I DON'T ALWAYS TEST MY
CODE

A photograph of the character "The Most Interesting Man in the World" from the TV show "Borat". He is an older man with a full, grey beard and mustache, wearing a dark pinstripe suit jacket over a white button-down shirt. He is leaning forward, resting his chin on his hand, with a bottle of Dos Equis beer on a wooden table in front of him.

BUT WHEN I DO I DO IT IN
PRODUCTION

NETFLIX



SIMIAN ARMY

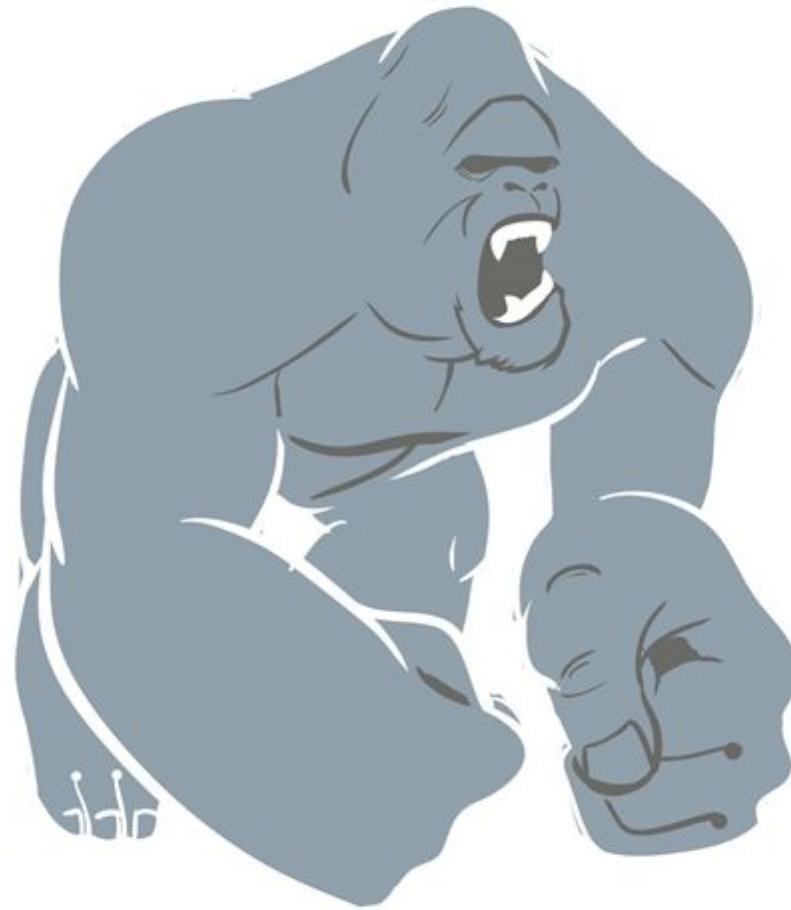


amazon
web services™

Chaos Gorilla



Chaos Kong



Conformity Monkey



Security Monkey



Janitor Monkey



Latency Monkey



“Everything fails all the time”

Werner Vogels, VP + CTO @ Amazon

Design for Failure

“It is far better to be in a constant state of minor failure”

Richard Rodger, CEO @ nearForm

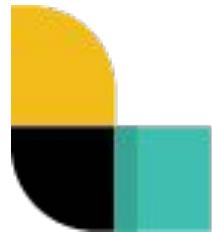
Self healing systems

Debugging Microservices

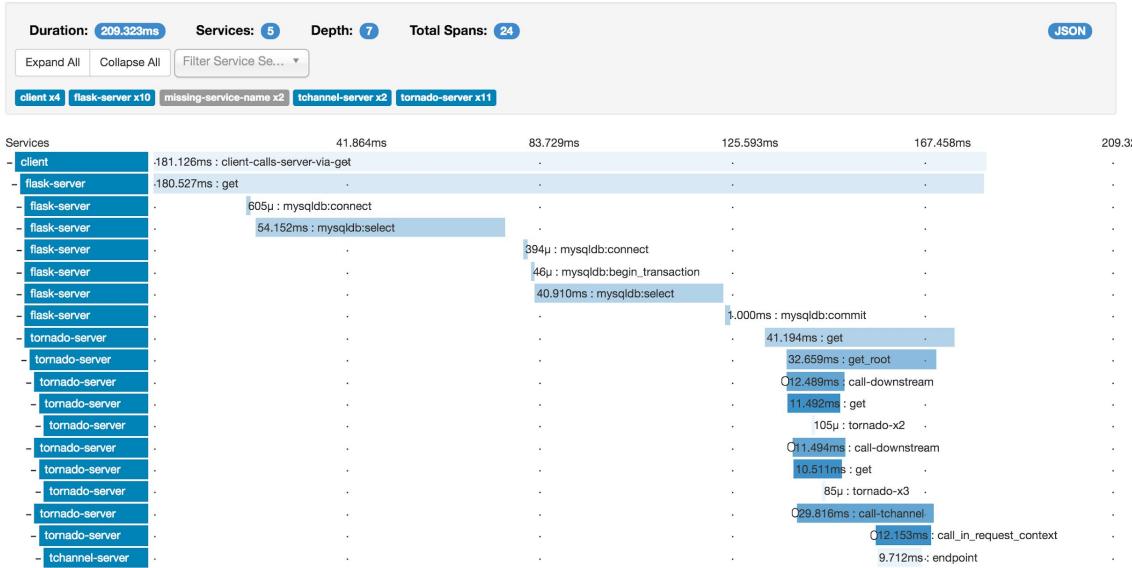
splunk®>



elastic



logstash



“How big should my Microservice be?”
Everyone

“Not all of a large system will
be well designed”

Eric Evans

Domain Driven Design (DDD)

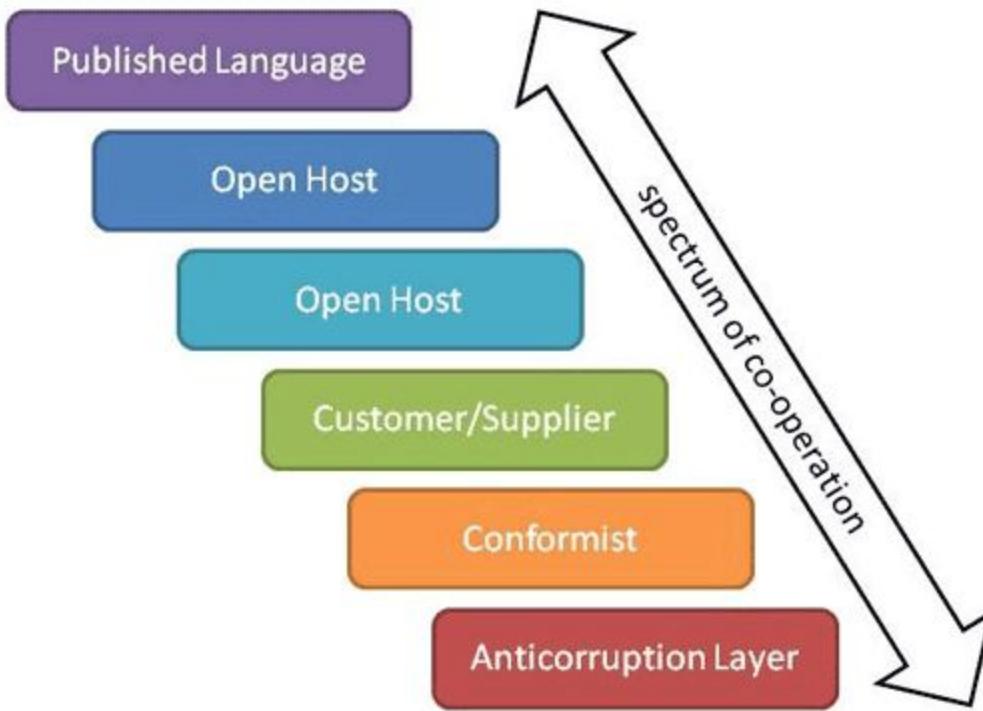
Domain-Driven DESIGN

Tackling Complexity in the Heart of Software



Eric Evans

Foreword by Martin Fowler

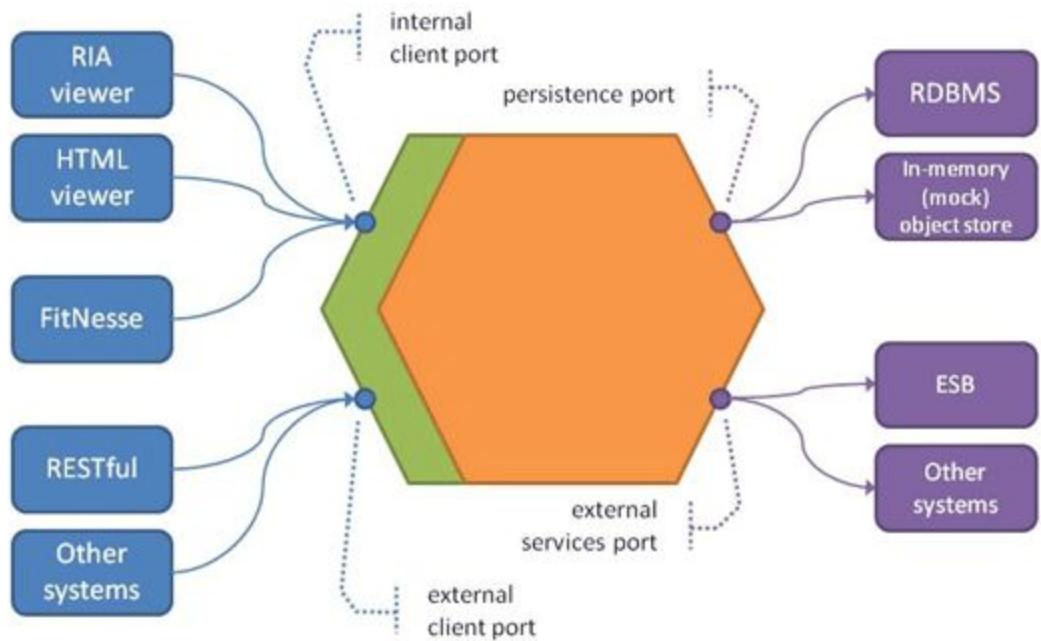


Presentation
Layer

Application
Layer

Domain
Layer

Infrastructure
Layer



“... big enough to fit in your head”

Dan North



“Replaceable component architecture”

Dan North

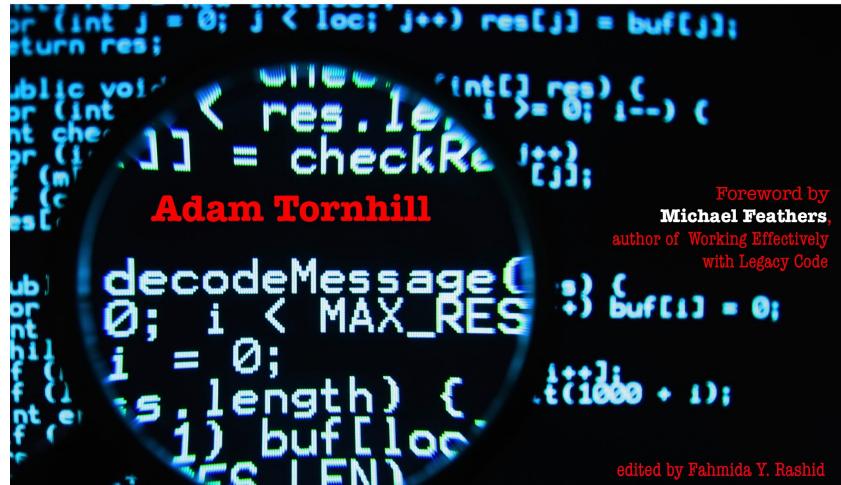


“The first draft of anything is shit”
Ernest Hemingway

Monolith first?

Your Code as a Crime Scene

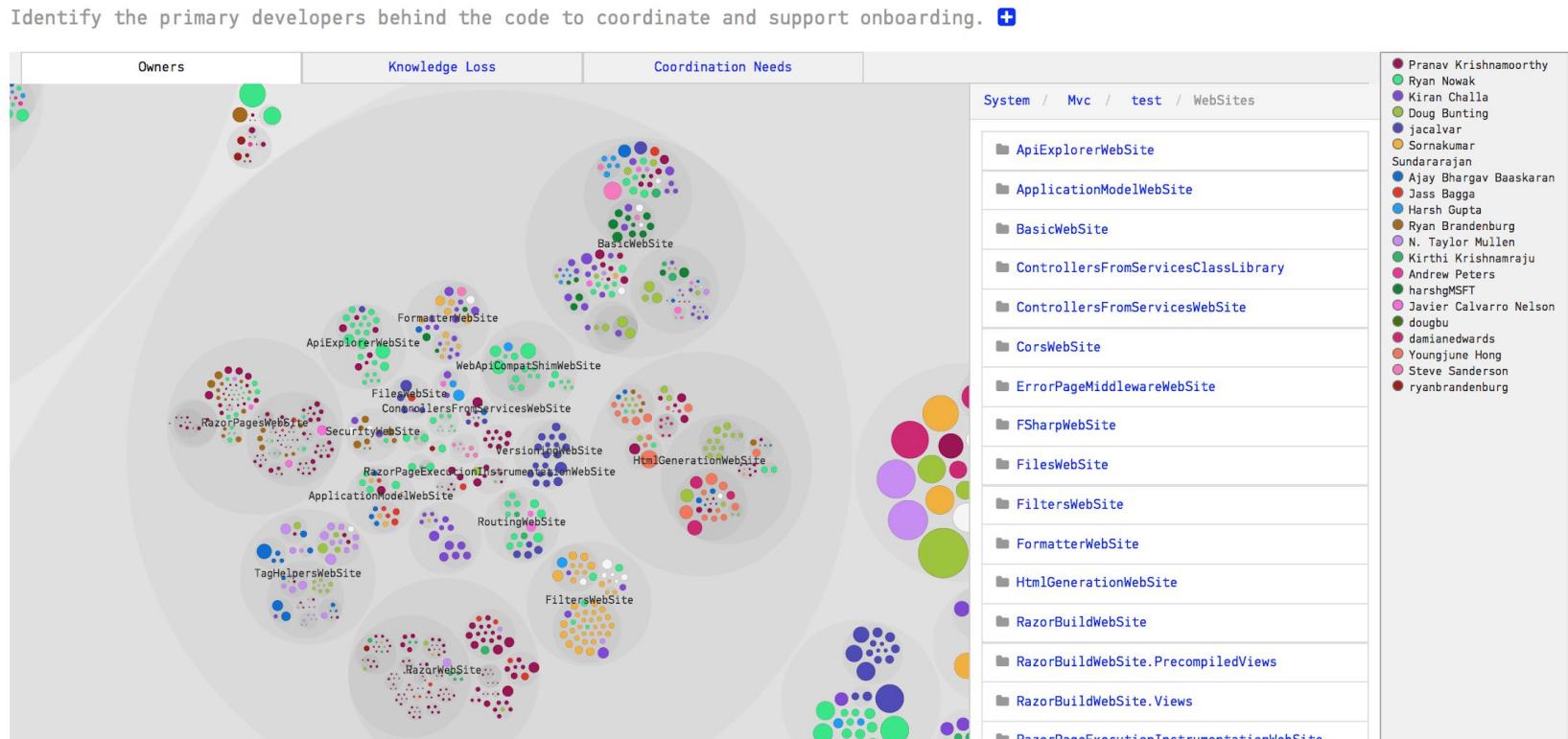
Use Forensic Techniques
to Arrest Defects, Bottlenecks, and
Bad Design in Your Programs

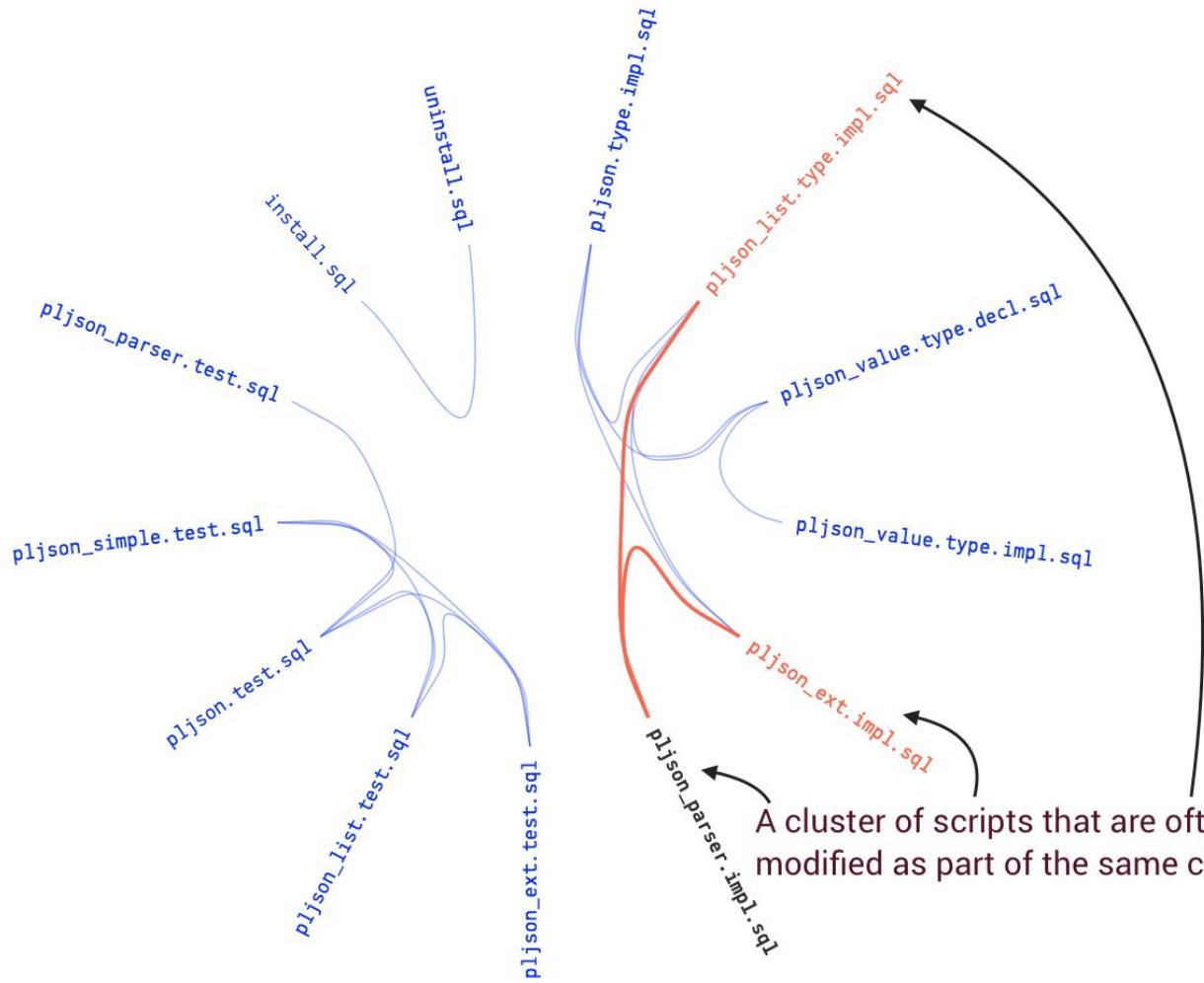


CØDESCENE

ANALYSIS RESULTS
ASP.NET MVC

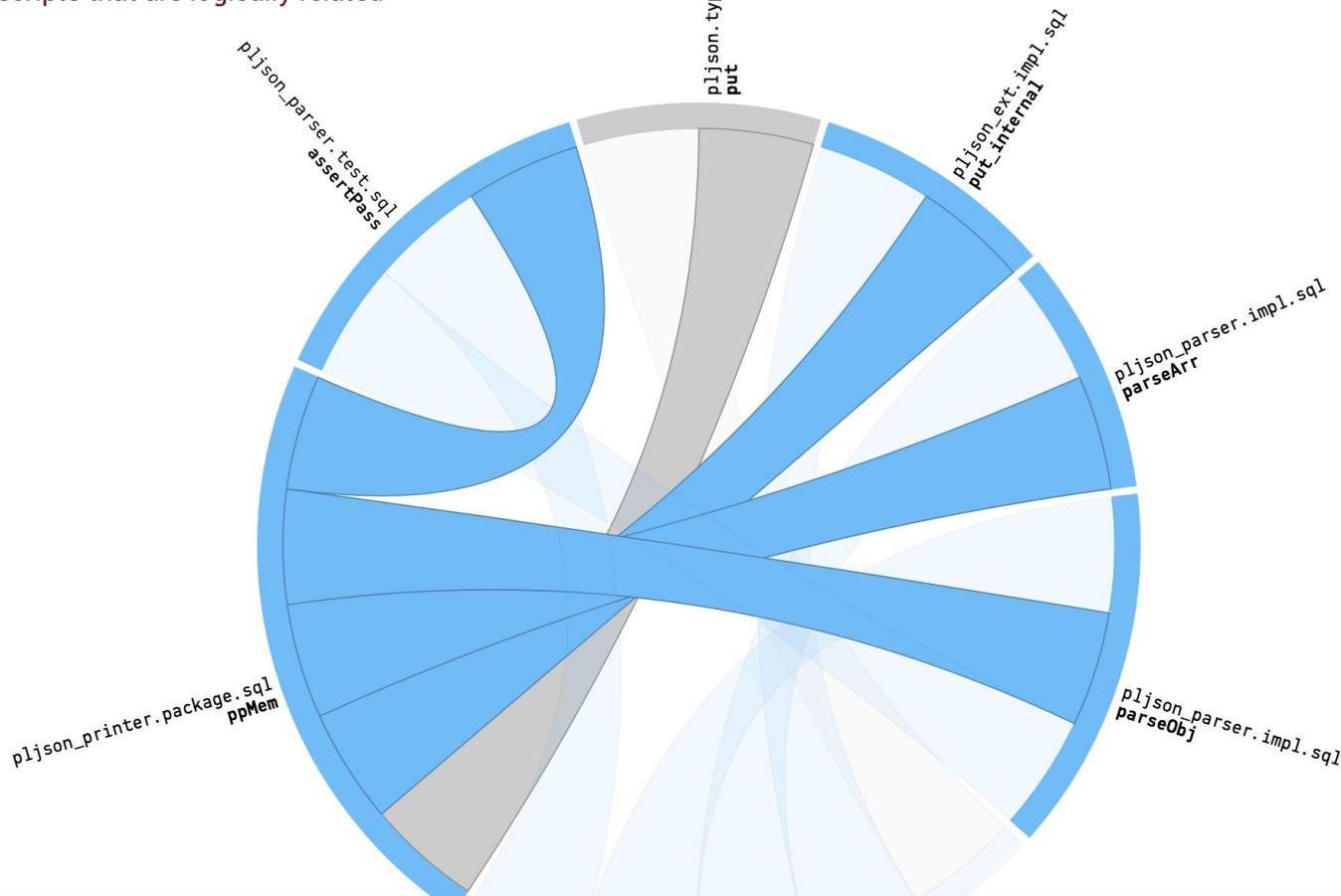
- Dashboard
- Scope
- Technical Debt
- Architecture
- Social Analyses
 - Social Networks
 - Individuals
 - Teams
 - Modus Operandi
 - Authors
- Project Management





A cluster of scripts that are often modified as part of the same changes.

Identify procedures in different SQL scripts that are logically related





THE TWELVE-FACTOR APP

12factor.net

I. Codebase

One codebase tracked in revision control, many deploys

I. Codebase

One codebase tracked in revision control, many deploys

Always use version control!

I. Codebase

One codebase tracked in revision control, many deploys

Always use version control!

One app (microservice) per repo

Codebase



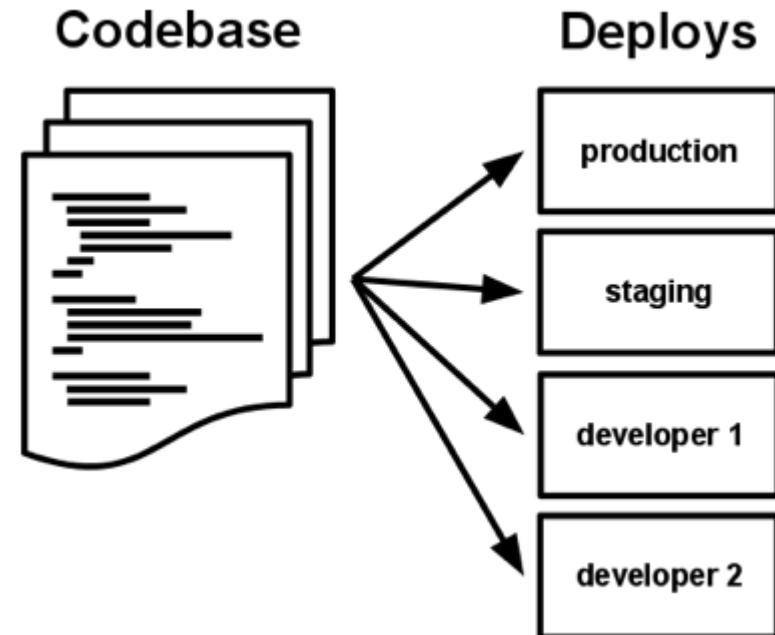
I. Codebase

One codebase tracked in revision control, many deploys

Always use version control!

One app (microservice) per repo

Multiple deploys per app (local, dev, prod, etc.)



II. Dependencies

Explicitly declare and isolate dependencies

II. Dependencies

Explicitly declare and isolate dependencies

Never assume something is pre-installed

II. Dependencies

Explicitly declare and isolate dependencies

Never assume something is pre-installed

Package managers/repos make this easy!

```
$ pip install -r requirements.txt
```

```
$ npm install
```

```
$ bundle install
```

II. Dependencies

Explicitly declare and isolate dependencies

Never assume something is pre-installed

Package managers/repos make this easy!

Isolate dependencies to prevent ‘leaking’

```
$ pip install -r requirements.txt
```

```
$ npm install
```

```
$ bundle install
```

II. Dependencies

Explicitly declare and isolate dependencies

Never assume something is pre-installed

Package managers/repos make this easy!

Isolate dependencies to prevent 'leaking'

```
"dependencies": {  
  "back": "^0.1.5",  
  "ringbufferjs": "0.0.1",  
  "xtend": "^4.0.0"  
},
```

```
$ npm install
```



node_modules

II. Dependencies

Explicitly declare and isolate dependencies

Never assume something is pre-installed

Package managers/repos make this easy!

Isolate dependencies to prevent ‘leaking’

(Aims to) prevent “it works on my machine”

```
"dependencies": {  
  "back": "^0.1.5",  
  "ringbufferjs": "0.0.1",  
  "xtend": "^4.0.0"  
},
```

```
$ npm install
```



node_modules

III. Config

Store config in the environment

III. Config

Store config in the environment

Config changes between deploys (dev, prod, etc.)

III. Config

Store config in the environment

Config changes between deploys (dev, prod, etc.)

Store config in environment not codebase

III. Config

Store config in the environment

Config changes between deploys (dev, prod, etc.)

Store config in environment not codebase



III. Config

Store config in the environment

Config changes between deploys (dev, prod, etc.)

Store config in environment not codebase

Suggest environment vars, but other options...



III. Config

Store config in the environment

Config changes between deploys (dev, prod, etc.)

Store config in environment not codebase

Suggest environment vars, but other options...

Supports multiple deploys from one codebase



III. Config

Store config in the environment

Config changes between deploys (dev, prod, etc.)

Store config in environment not codebase

Suggest environment vars, but other options...

Supports multiple deploys from one codebase

Makes Open Sourcing easier



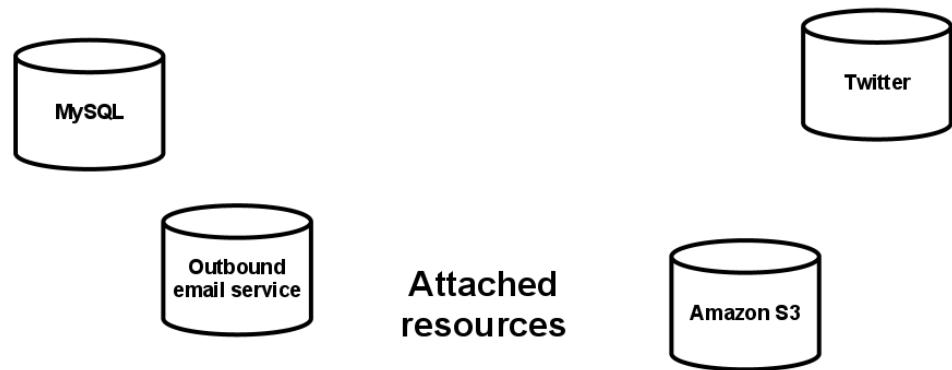
IV. Backing services

Treat backing services as attached resources

IV. Backing services

Treat backing services as attached resources

Backing services are any remote services

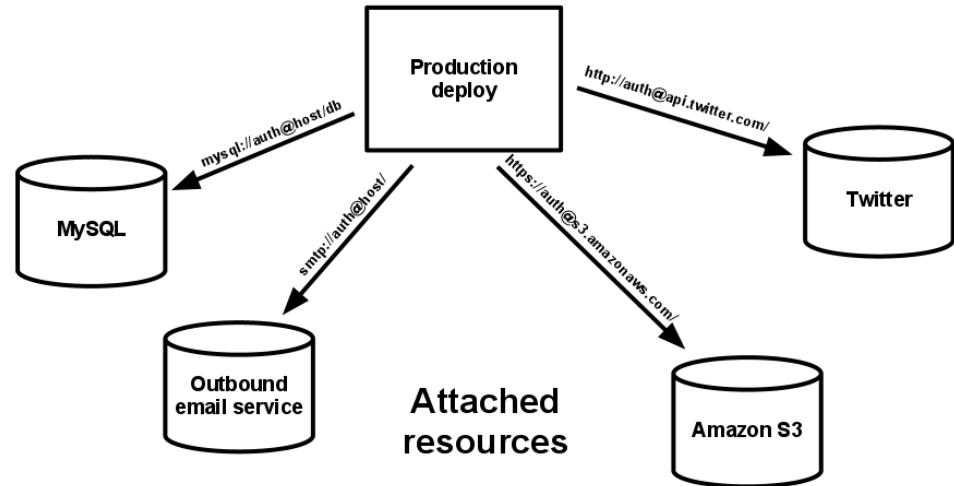


IV. Backing services

Treat backing services as attached resources

Backing services are any remote services

Loosely coupled to a deploy



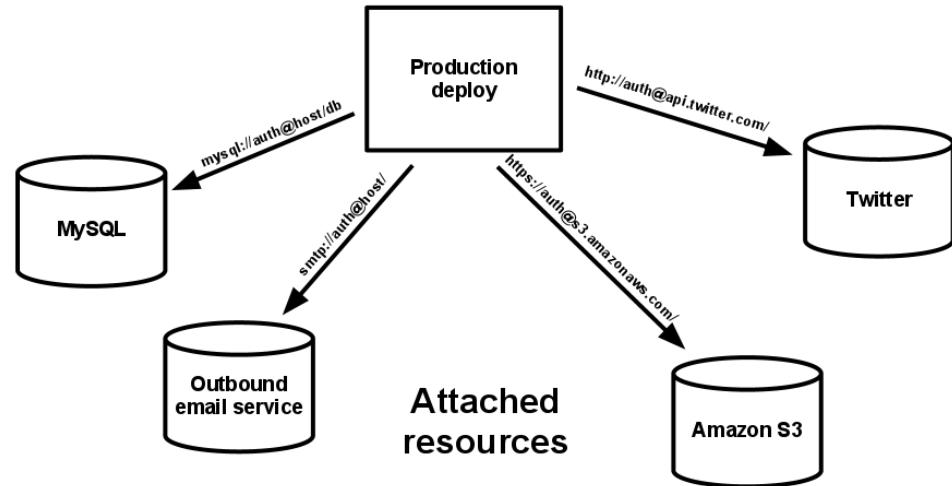
IV. Backing services

Treat backing services as attached resources

Backing services are any remote services

Loosely coupled to a deploy

Treated as local resources



IV. Backing services

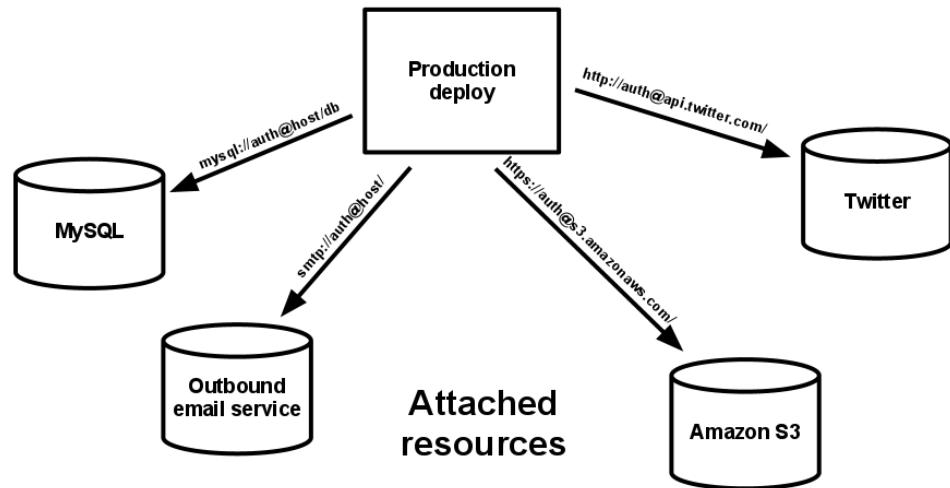
Treat backing services as attached resources

Backing services are any remote services

Loosely coupled to a deploy

Treated as local resources

Allows easy swapping out of services



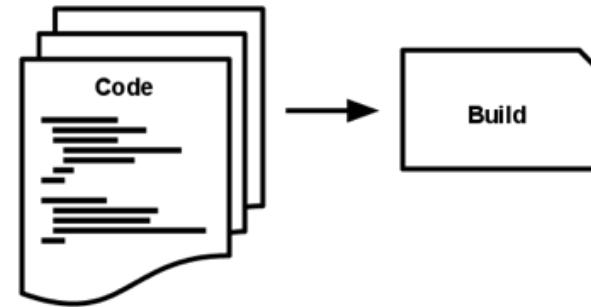
V. Build, release, run

Strictly separate build and run stages

V. Build, release, run

Strictly separate build and run stages

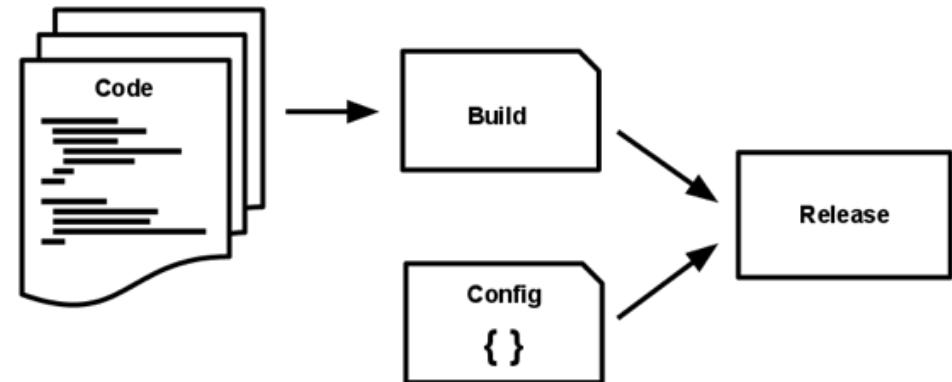
Build



V. Build, release, run

Strictly separate build and run stages

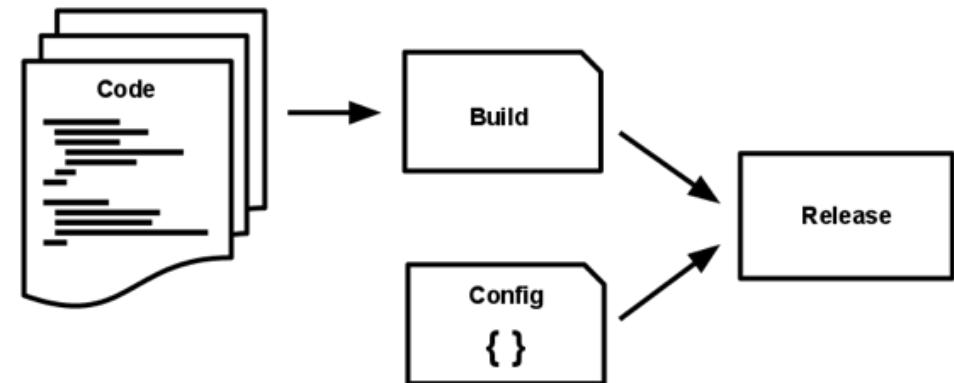
Build → Release



V. Build, release, run

Strictly separate build and run stages

Build → Release → Run

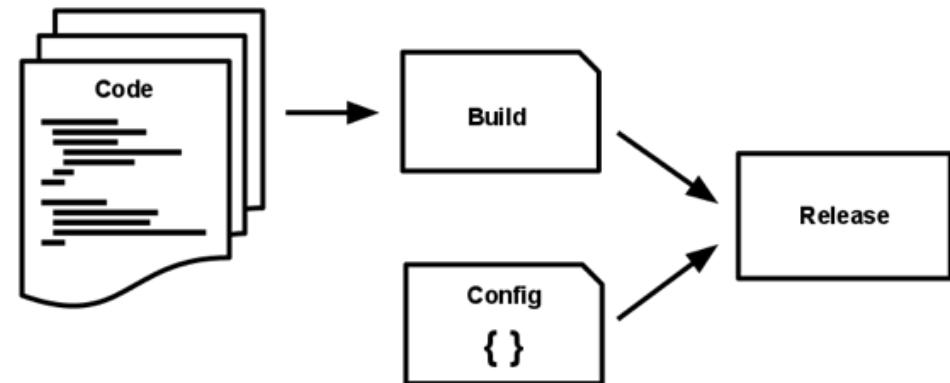


V. Build, release, run

Strictly separate build and run stages

Build → Release → Run

Strict separation between the stages



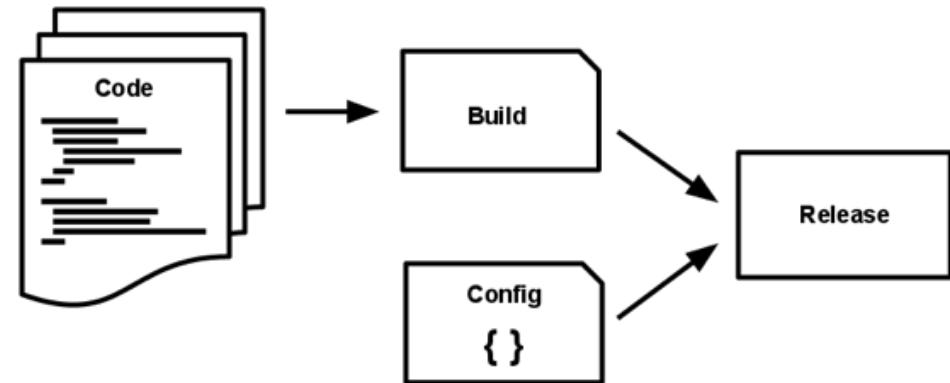
V. Build, release, run

Strictly separate build and run stages

Build → Release → Run

Strict separation between the stages

One way (no tweaking the live server!)



VI. Processes

Execute the app as one or more stateless processes

VI. Processes

Execute the app as one or more stateless processes

Apps run as processes (duh)

VI. Processes

Execute the app as one or more stateless processes

Apps run as processes (duh)

No state: every node is independent

VI. Processes

Execute the app as one or more stateless processes

Apps run as processes (duh)

No state: every node is independent

Caching is fine, but not for future use

VI. Processes

Execute the app as one or more stateless processes

Apps run as processes (duh)

No state: every node is independent

Caching is fine, but not for future use

State must be stored in backing services

VII. Port binding

Export services via port binding

VII. Port binding

Export services via port binding

Expose endpoints

`http://localhost:5000/`

`test.mosquitto.org:1883`

`redis://redisbox:6379/0`

VII. Port binding

Export services via port binding

Expose endpoints

No runtime injection or native APIs

`http://localhost:5000/`

`test.mosquitto.org:1883`

`redis://redisbox:6379/0`

VII. Port binding

Export services via port binding

Expose endpoints

No runtime injection or native APIs

`http://localhost:5000/`

HTTP or other protocols make things easy

`test.mosquitto.org:1883`

`redis://redisbox:6379/0`

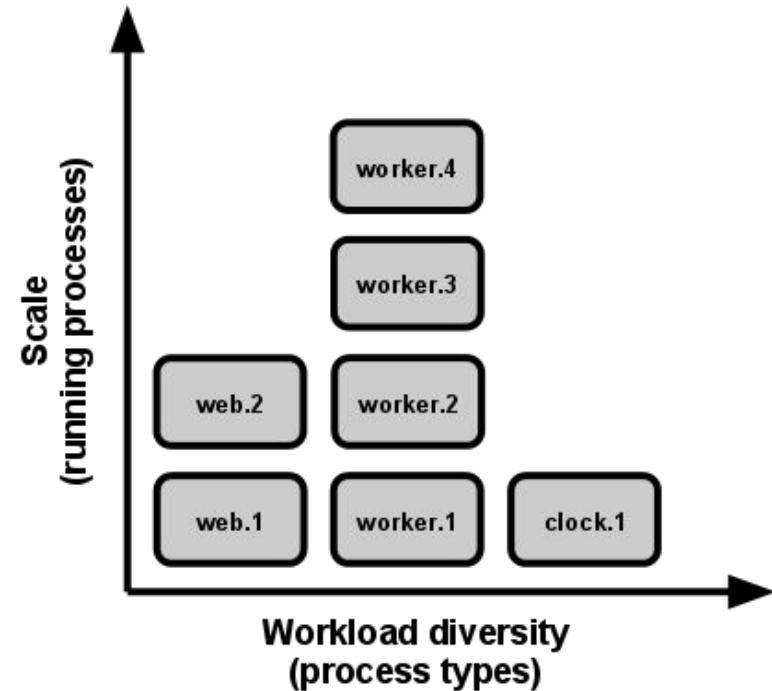
VIII. Concurrency

Scale out via the process model

VIII. Concurrency

Scale out via the process model

Processes are the unit of scaling

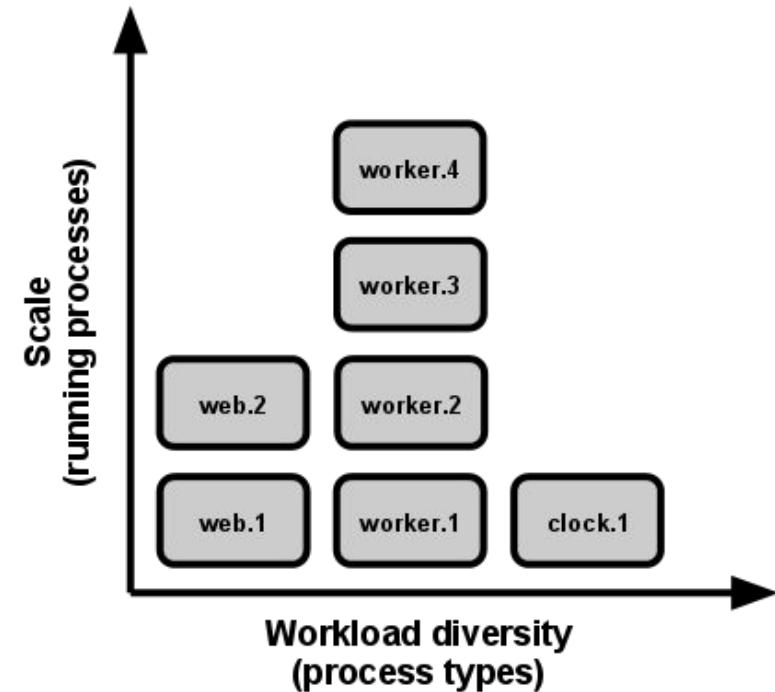


VIII. Concurrency

Scale out via the process model

Processes are the unit of scaling

Don't daemonize!



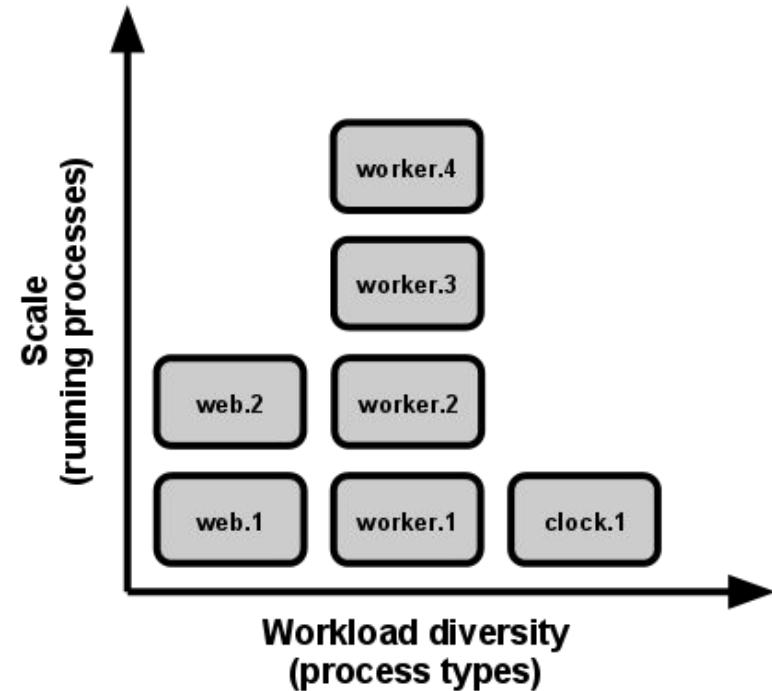
VIII. Concurrency

Scale out via the process model

Processes are the unit of scaling

Don't daemonize!

Take advantage of OS resource scaling



IX. Disposability

Maximize robustness with fast startup and graceful shutdown

IX. Disposability

Maximize robustness with fast startup and graceful shutdown

Can be started or stopped quickly (Cattle not pets)

IX. Disposability

Maximize robustness with fast startup and graceful shutdown

Can be started or stopped quickly (Cattle not pets)

Quick startup allows easy scaling up

Graceful shutdown allows easy scaling down

IX. Disposability

Maximize robustness with fast startup and graceful shutdown

Can be started or stopped quickly (Cattle not pets)

Quick startup allows easy scaling up

Graceful shutdown allows easy scaling down

Easier if you scale with independent processes...

X. Dev/prod parity

Keep development, staging, and production as similar as possible

X. Dev/prod parity

Keep development, staging, and production as similar as possible

Keep time between environment deploys low

X. Dev/prod parity

Keep development, staging, and production as similar as possible

Keep time between environment deploys low

Devs also handle deployment and maintenance (DevOps)

Keep tooling between environments as similar as possible

X. Dev/prod parity

Keep development, staging, and production as similar as possible

Keep time between environment deploys low

Devs also handle deployment and maintenance (DevOps)

Keep tooling between environments as similar as possible

If you can, try one environment...

XI. Logs

Treat logs as event streams

XI. Logs

Treat logs as event streams

Don't mess around writing to files, just use `stdout`

XI. Logs

Treat logs as event streams

Don't mess around writing to files, just use `stdout`

In dev, you can watch `stdout`

In production, it can be routed to other stores (Splunk etc.)

XI. Logs

Treat logs as event streams

Don't mess around writing to files, just use stdout

In dev, you can watch stdout

In production, it can be routed to other stores (Splunk etc.)

Keeps aggregated logs time ordered (like Kafka)



XII. Admin processes

Run admin/management tasks as one-off processes

XII. Admin processes

Run admin/management tasks as one-off processes

One-off admin tasks (DB migrations etc.) should be run from production environment

XII. Admin processes

Run admin/management tasks as one-off processes

One-off admin tasks (DB migrations etc.) should be run from production environment

Don't run scripts from local machine or directly on the DB

XII. Admin processes

Run admin/management tasks as one-off processes

One-off admin tasks (DB migrations etc.) should be run from production environment

Don't run scripts from local machine or directly on the DB

Check in scripts to app repos to keep things in sync

XII. Admin processes

Run admin/management tasks as one-off processes

One-off admin tasks (DB migrations etc.) should be run from production environment

Don't run scripts from local machine or directly on the DB

Check in scripts to app repos to keep things in sync

ssh into production box and run the task

XII. Admin processes

Run admin/management tasks as one-off processes

One-off admin tasks (DB migrations etc.) should be run from production environment

Don't run scripts from local machine or directly on the DB

Check in scripts to app repos to keep things in sync

ssh into production box and run the task

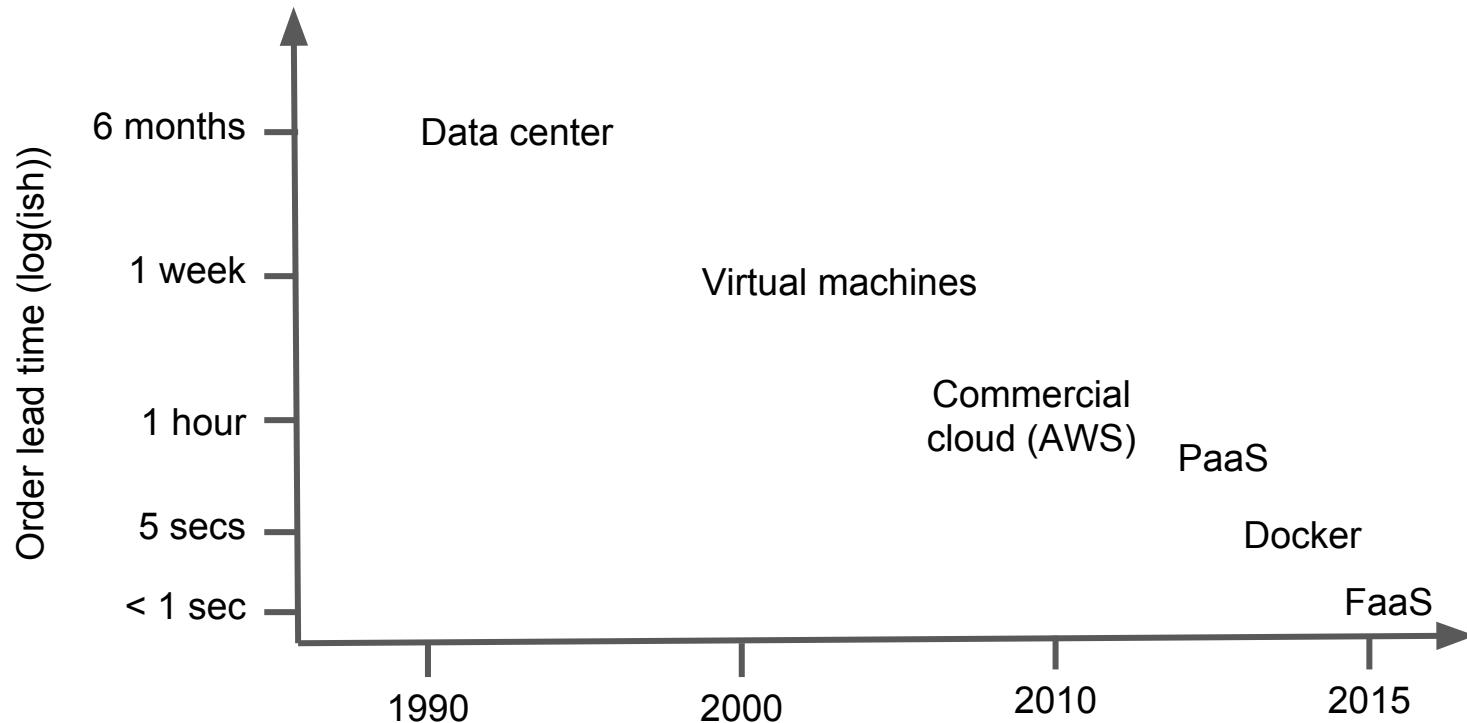
...or don't. Immutable Infrastructure!



CHEF™



ANSIBLE



Microservices or distributed systems?

The future of Microservices

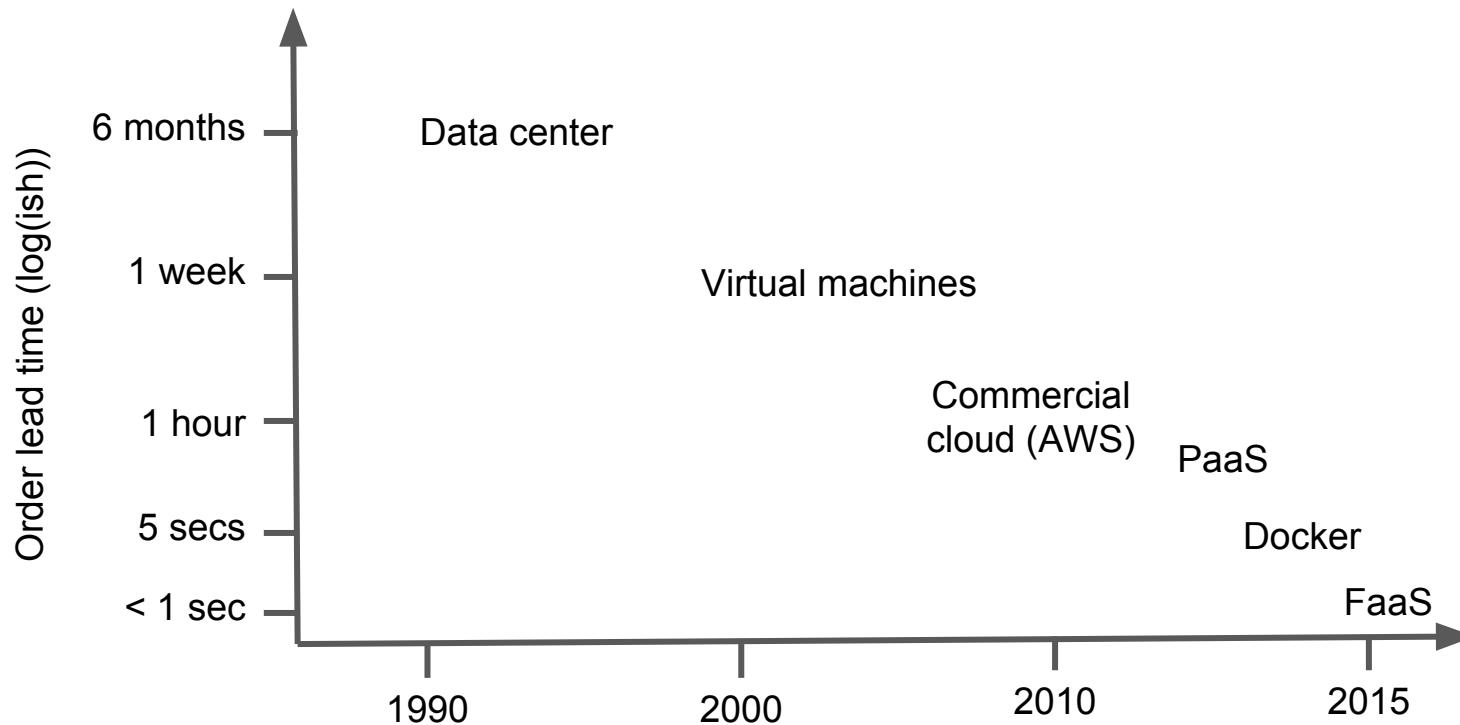
Saying something so people feel better about giving you money



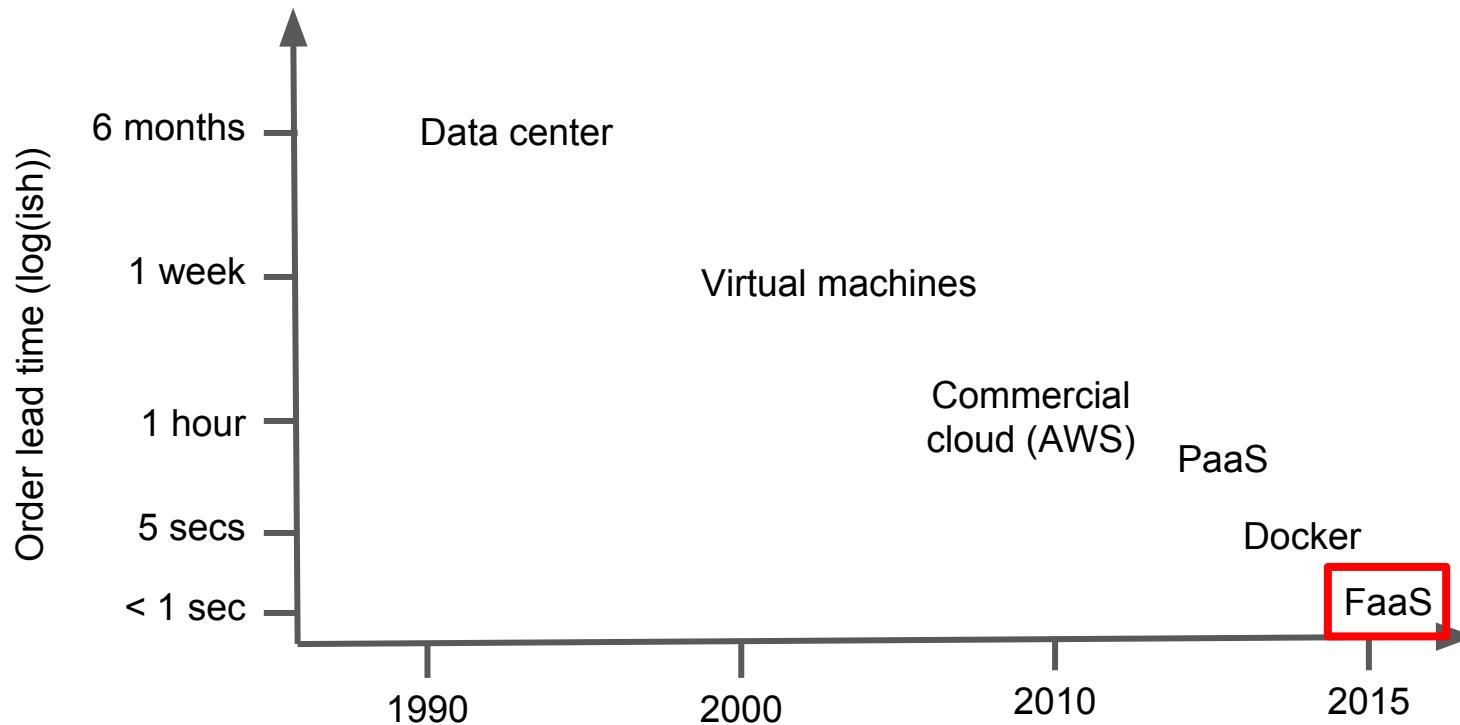
Making

Arbitrary Forecasts

Hardware lead times



Hardware lead times



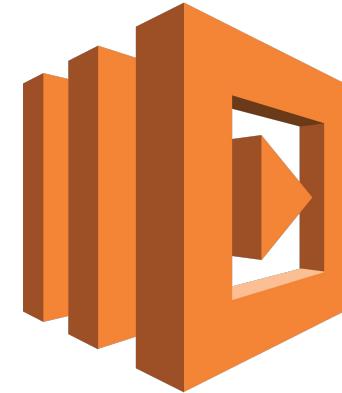
Serverless

‘Serverless’ (not really)

Function as a Service (FaaS)

No persistence, no upkeep costs

200 ms startup time (Node.js)



Amazon Lambda

What is the role of a manager?

Managers

Managers

Provide guidance

Managers

Provide guidance

Keep things ‘safe’

Managers

Provide guidance

Keep things ‘safe’

Organise the team

Managers

Provide guidance

Keep things ‘safe’

Organise the team

Enable team to work

Managers

Provide guidance

Keep things ‘safe’

Organise the team

Enable team to work

Microservices

Managers

Provide guidance

Keep things ‘safe’

Organise the team

Enable team to work

Microservices

Explore new tech

Managers

Provide guidance

Keep things ‘safe’

Organise the team

Enable team to work

Microservices

Explore new tech

Experiment

Managers

Provide guidance

Keep things ‘safe’

Organise the team

Enable team to work

Microservices

Explore new tech

Experiment

Enables self-organisation

Managers

Provide guidance

Keep things ‘safe’

Organise the team

Enable team to work

Microservices

Explore new tech

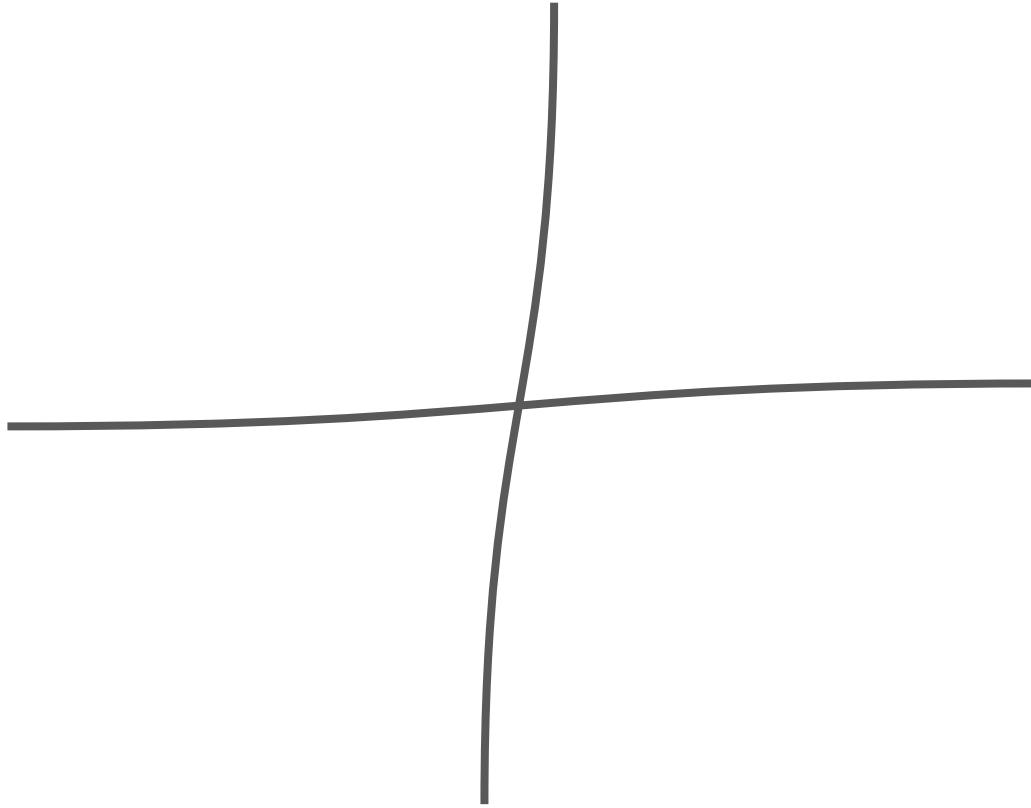
Experiment

Enables self-organisation

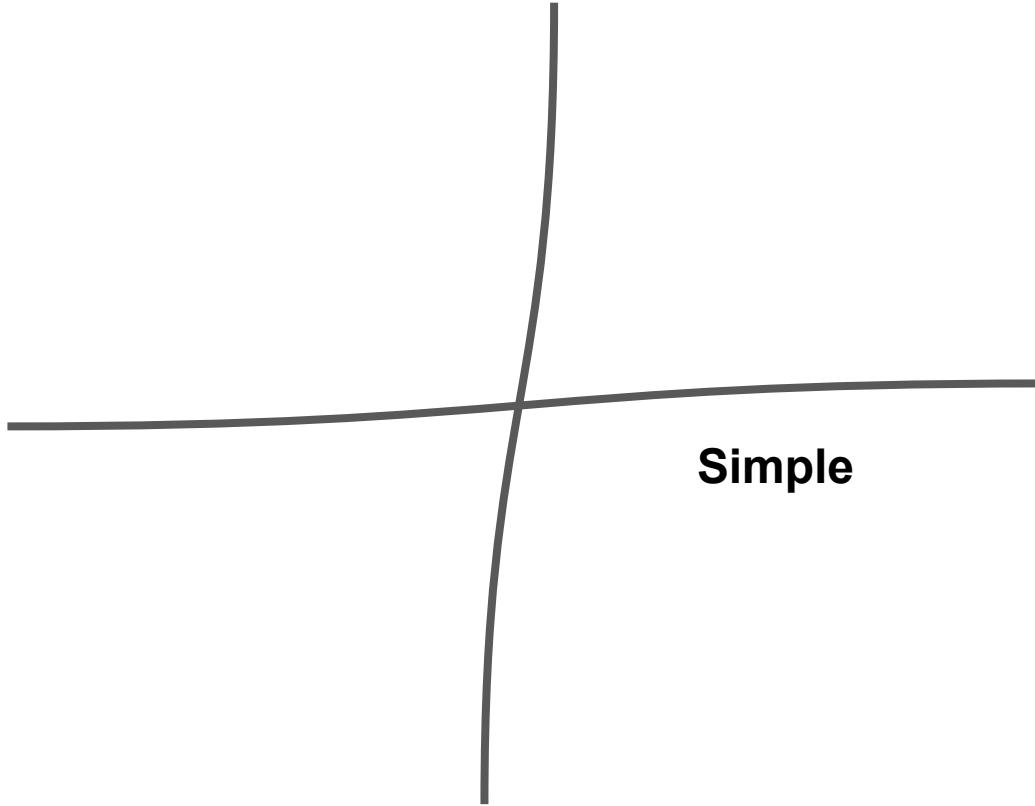
Tech empowers team

Cynefin Framework

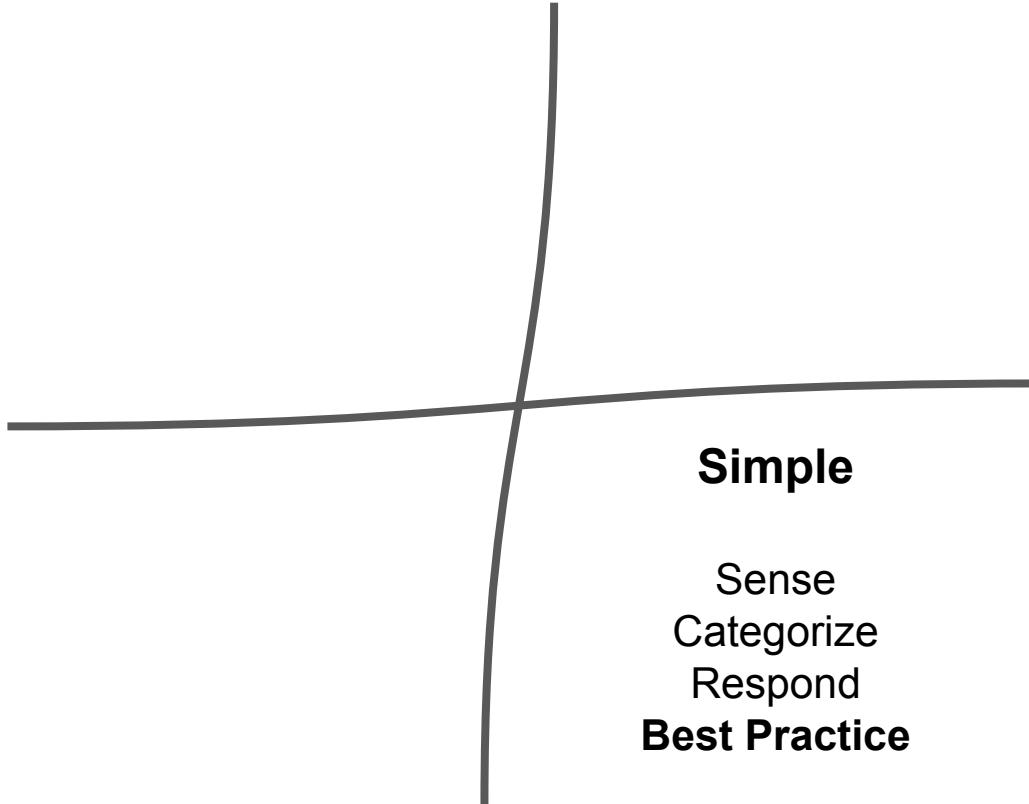
Cynefin Framework



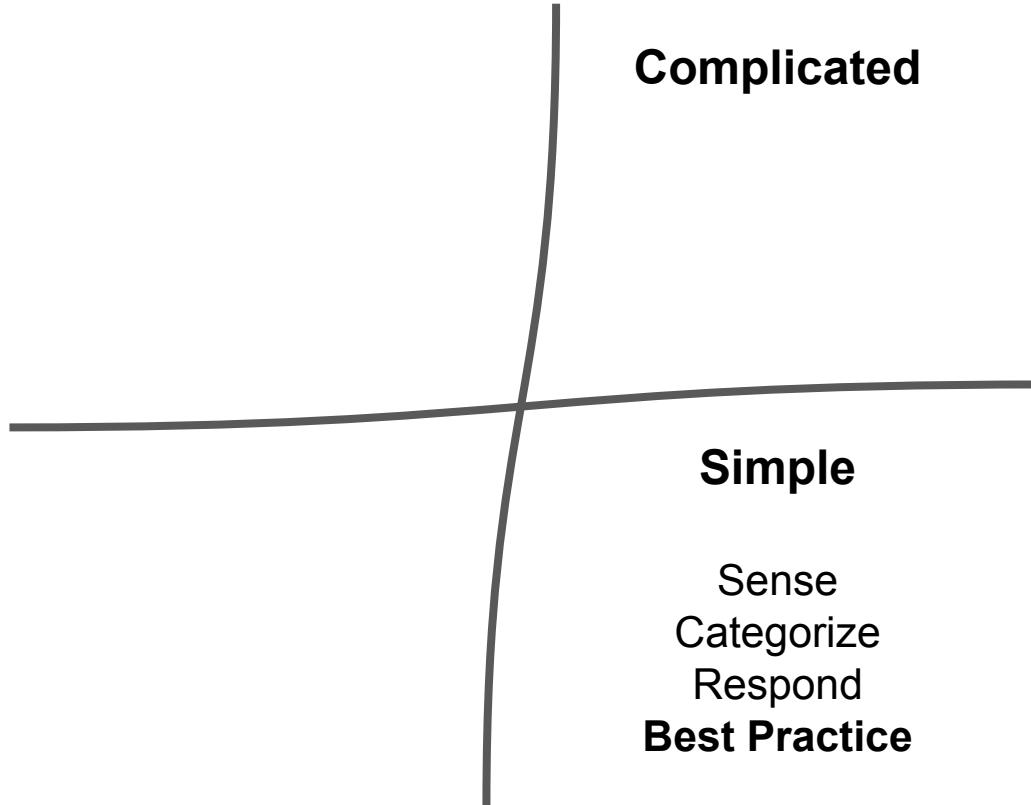
Cynefin Framework



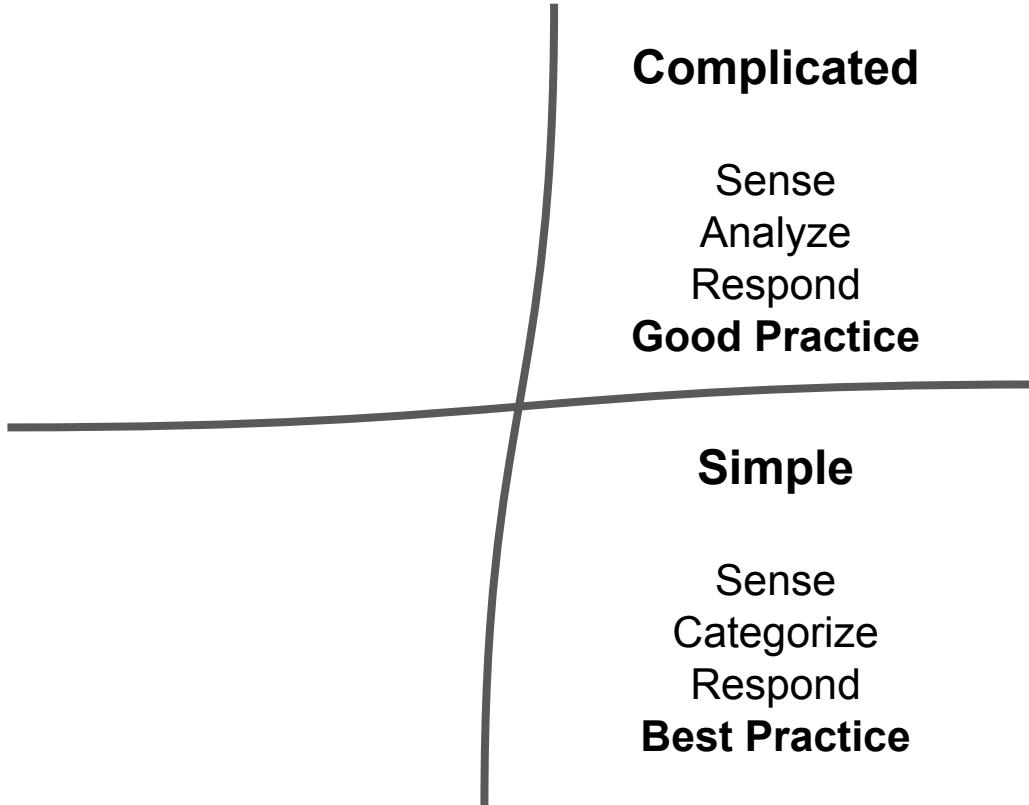
Cynefin Framework



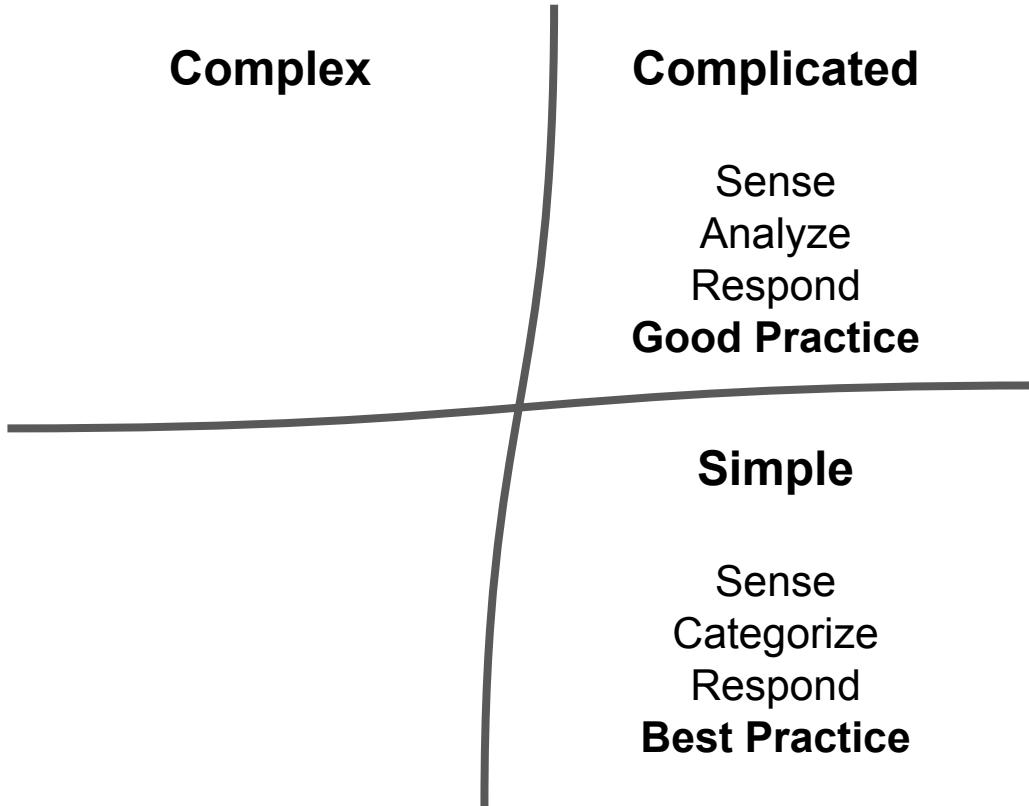
Cynefin Framework



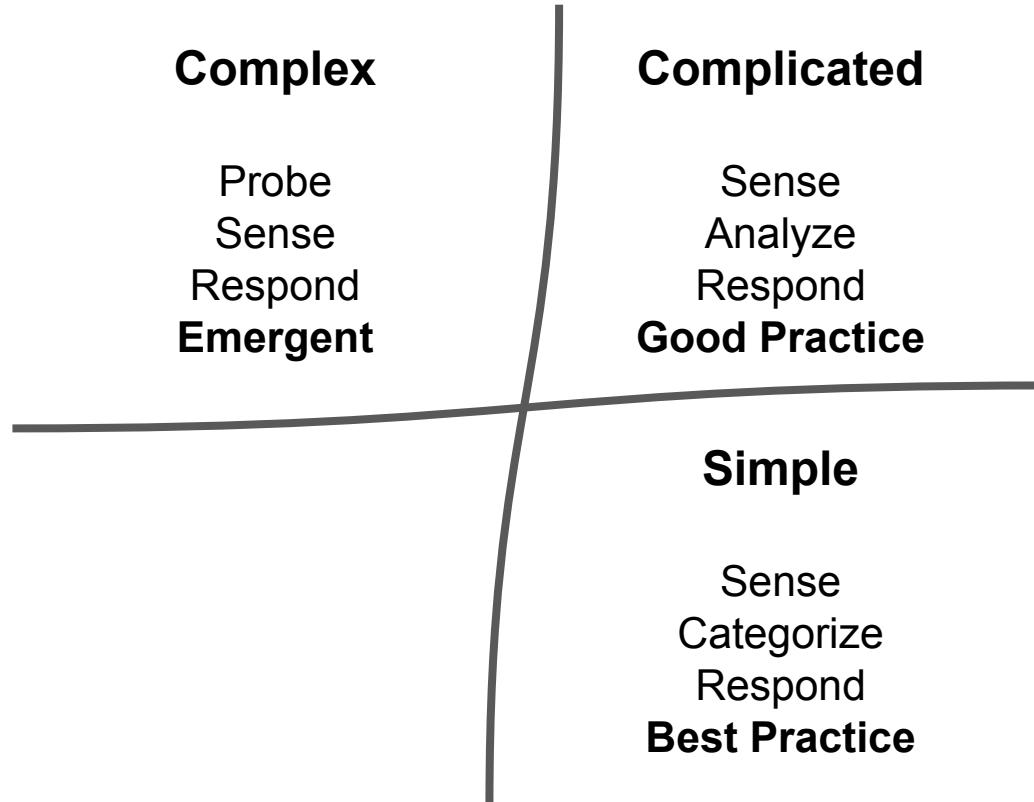
Cynefin Framework



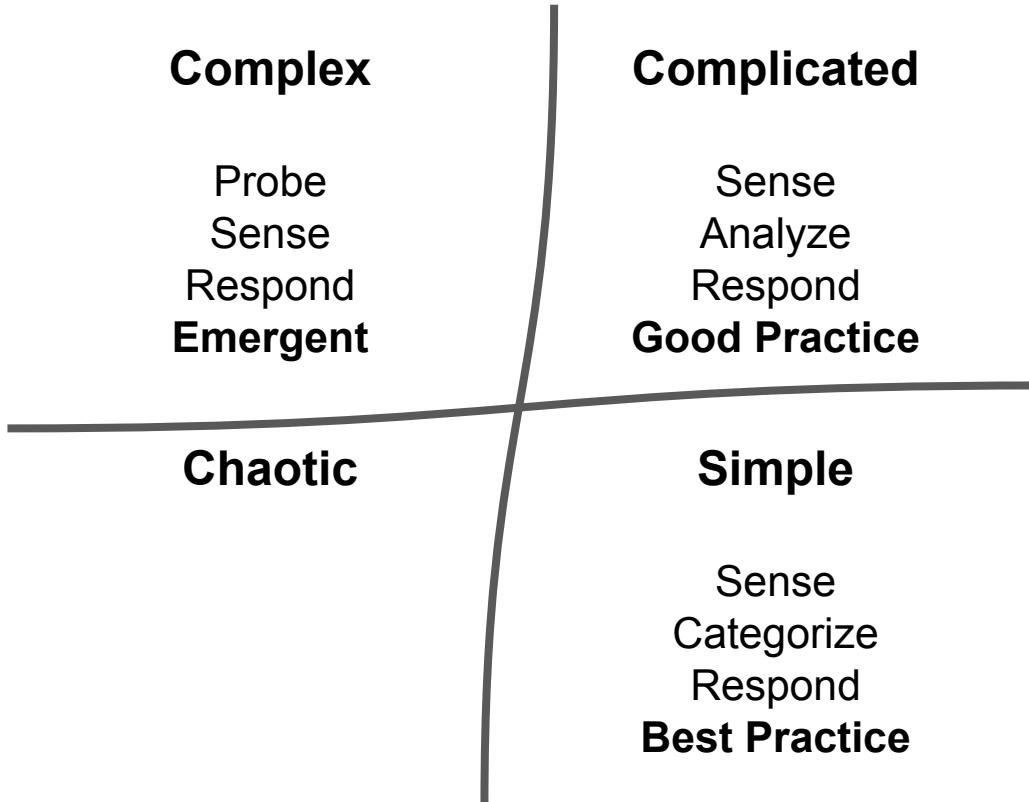
Cynefin Framework



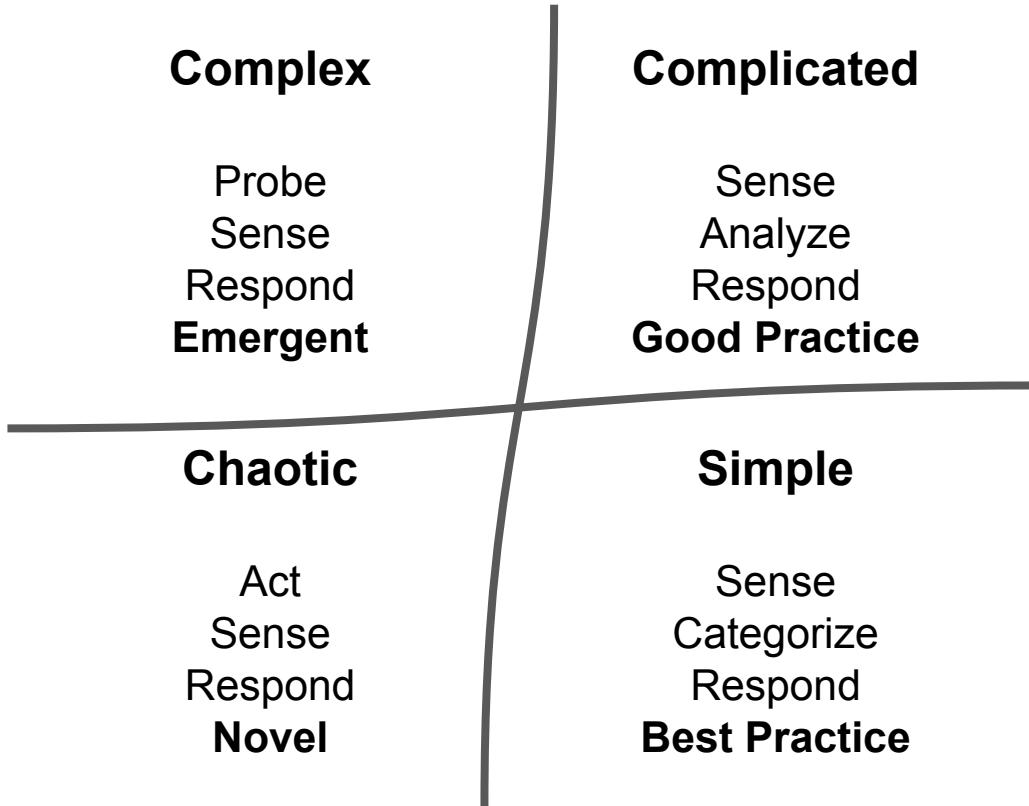
Cynefin Framework



Cynefin Framework



Cynefin Framework

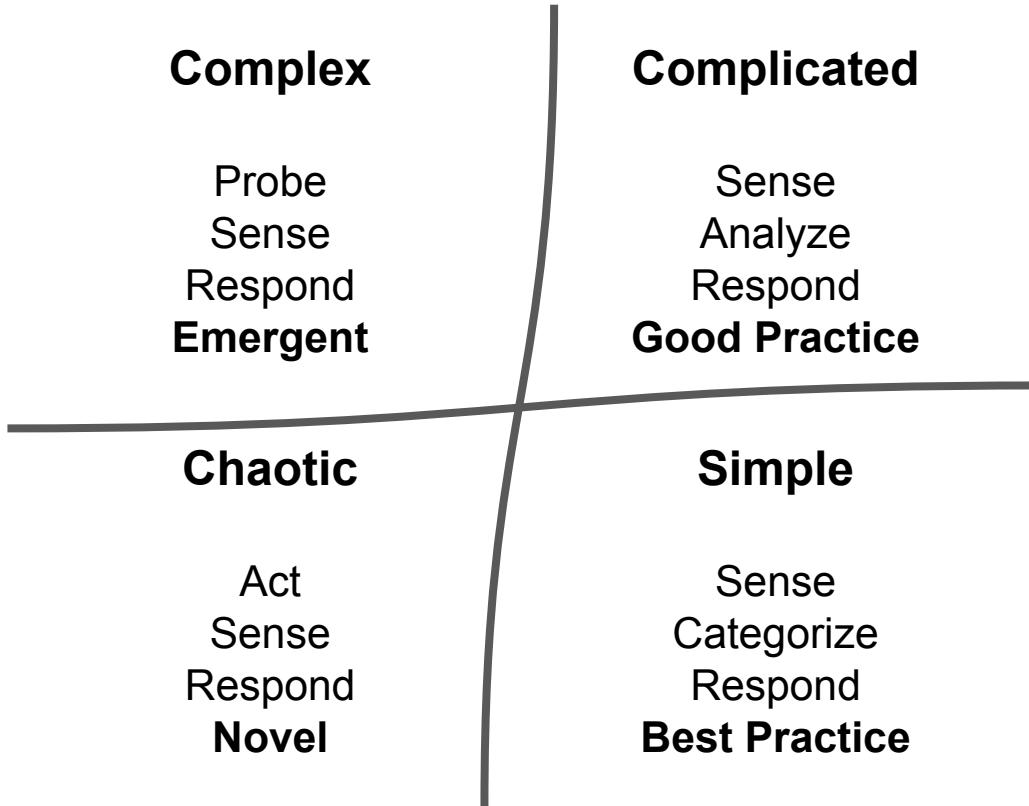


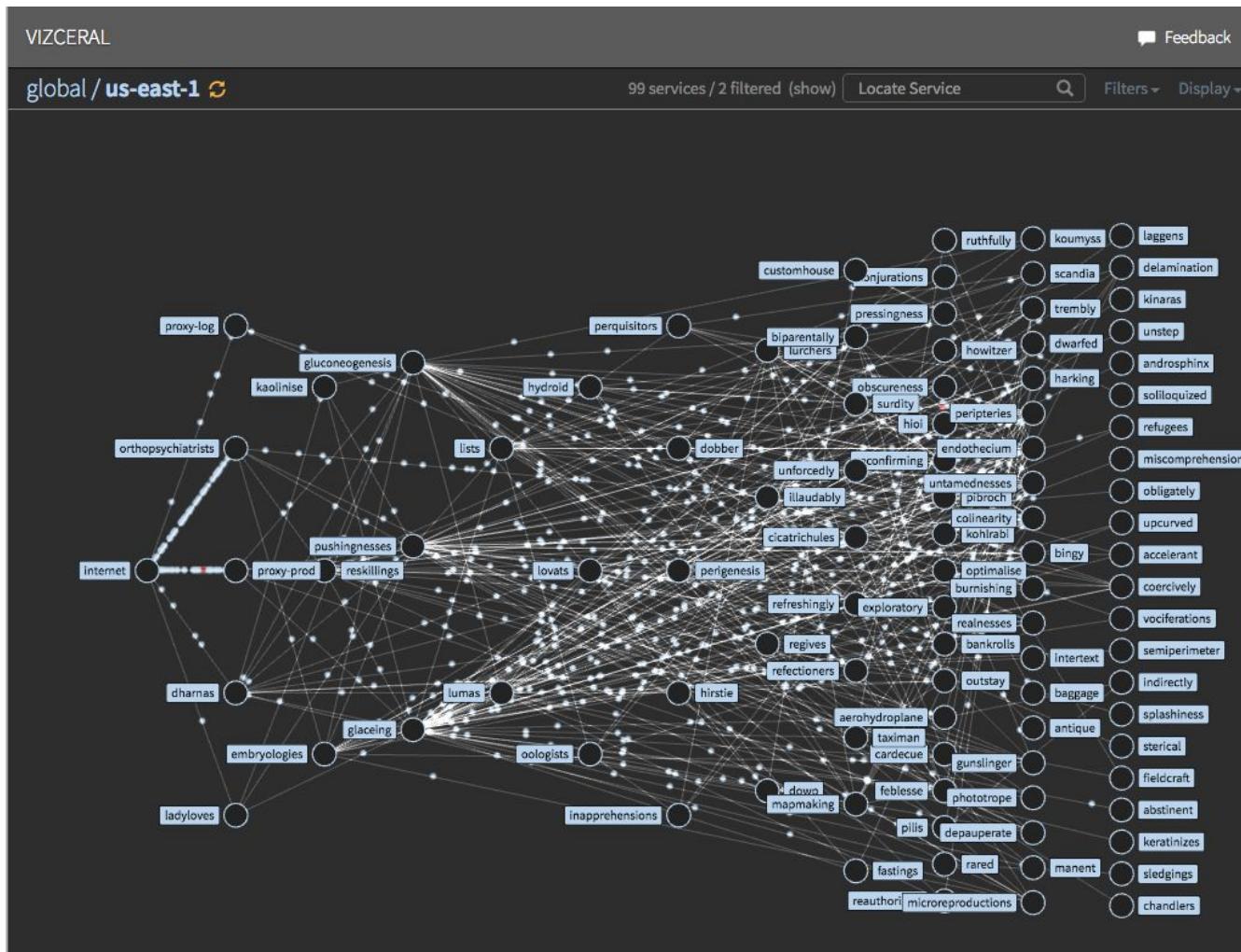
Catfin Framework



credit: @johnCutlefish

Cynefin Framework





VIZCERAL

Service Traffic Map / us-east-1

200 services / 110 filtered (show)

Locate Service



Filter +

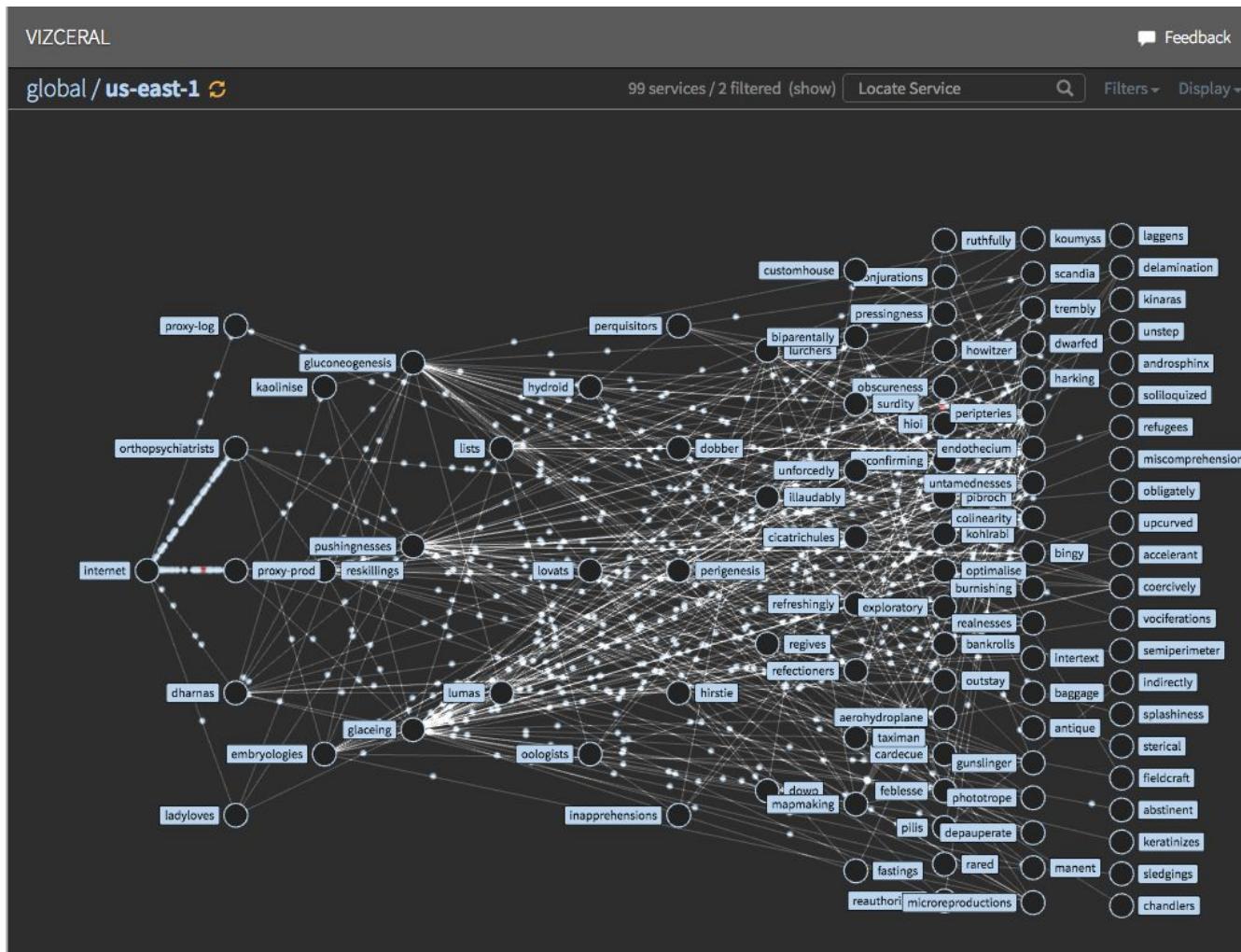
Display +



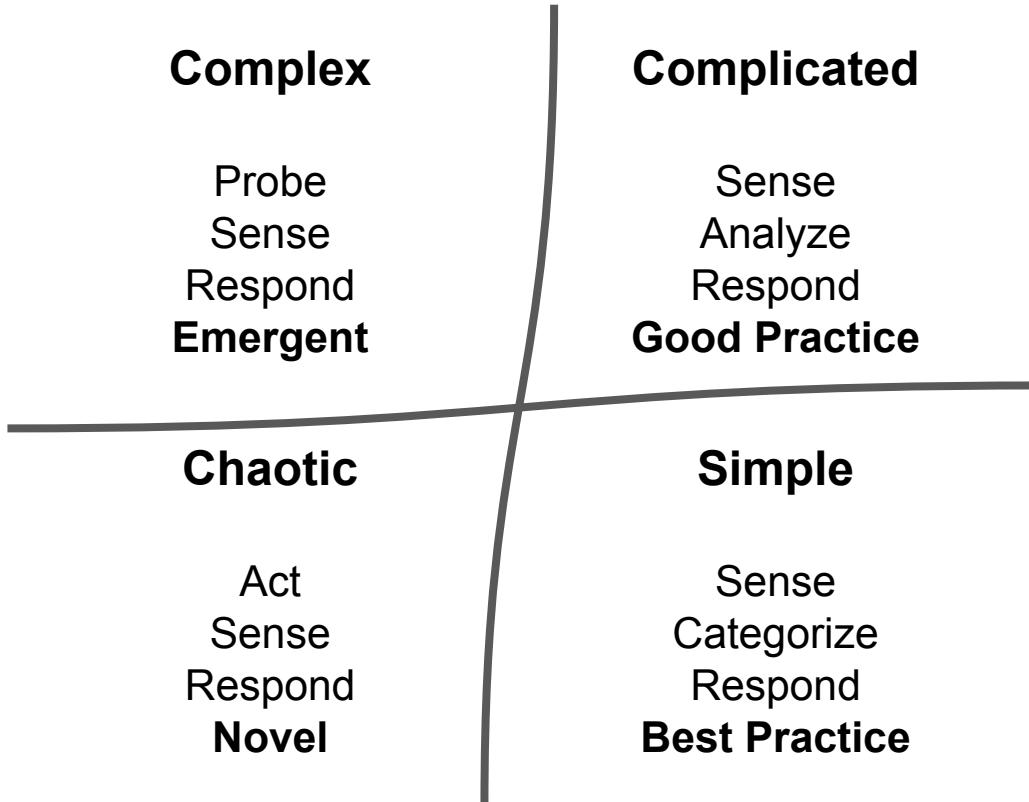
global / us-east-1 

99 services / 2 filtered (show)

Locate Service



Cynefin Framework



PROGRAMMER ANARCHY

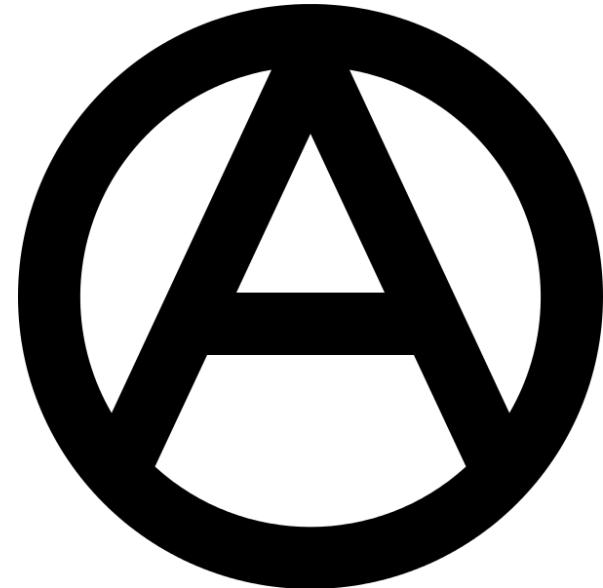
Programmer Anarchy

No managers (or other titles)

Programmers take responsibility

Expect/allow failure (remove fear)

Work directly with customers



Conclusion

Speed > efficiency

Speed > efficiency

DevOps

Speed > efficiency

DevOps

Empower teams

Speed > efficiency

DevOps

Empower teams

Embrace the distributed world

Microservices are **hard**.

Microservices are **hard**. Some things you get for
free

Microservices are **hard**. Some things you get for
free, but you have to **work for the good stuff**.

Microservices are **hard**. Some things you get for **free**, but you have to **work for the good stuff**. If you won't put in the work, you shouldn't be doing Microservices.

“Microservices are hard. Some things you get for free, but you have to work for the good stuff. If you won’t put in the work, you shouldn’t be doing Microservices. (You should be doing that stuff anyway!)”

“Microservices are hard. Some things you get for free, but you have to work for the good stuff. If you won’t put in the work, you shouldn’t be doing Microservices. (You should be doing that stuff anyway!)”

Steve Upton

Questions?

@Steve_Upton
Steveupton.io

This work is licensed under a [Creative Commons Attribution 4.0 International License](#).

Books!

A Novel About IT,
DevOps, and Helping
Your Business Win

The Phoenix Project

Gene Kim, Kevin Behr,
and George Spafford



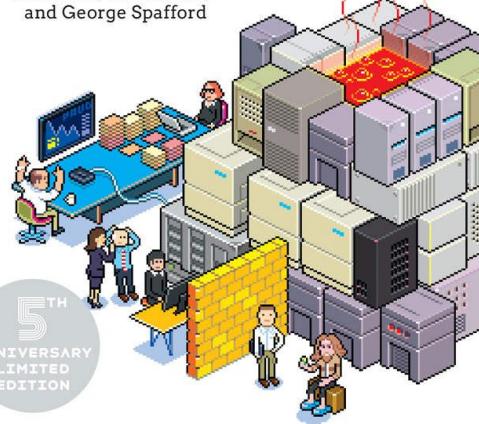
5TH
ANNIVERSARY
LIMITED
EDITION

Books!

A Novel About IT,
DevOps, and Helping
Your Business Win

The Phoenix Project

Gene Kim, Kevin Behr,
and George Spafford



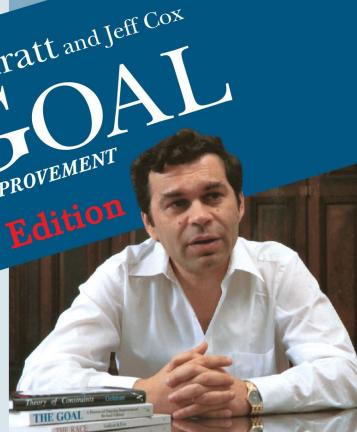
Includes
bonus
presentation
by the
author

Eliyahu M. Goldratt and Jeff Cox
THE GOAL
A PROCESS OF ONGOING IMPROVEMENT
30th Anniversary Edition

Eli Goldratt has been described by *Fortune* as a “guru to industry” and by *Business Week* as a “genius”. His book, *The Goal*, is a gripping fast-paced business novel.

“*Goal* readers are now doing the best work of their lives.”
Success Magazine

“A factory may be an unlikely setting for a novel, but the book has been wildly effective...”
Tom Peters



THE BEST-SELLING BUSINESS
NOVEL THAT INTRODUCED THE
**THEORY OF
CONSTRAINTS**
AND CHANGED HOW
AMERICA DOES BUSINESS

**OVER 6 MILLION
BOOKS SOLD!**

Books!

A Novel About IT,
DevOps, and Helping
Your Business Win

The Phoenix Project

Gene Kim, Kevin Behr,
and George Spafford



Includes
bonus
presentation
by the
author

Eliyahu M. Goldratt and Jeff Cox
THE GOAL
A PROCESS OF ONGOING IMPROVEMENT
30th Anniversary Edition

Eli Goldratt has been described by *Fortune* as a "guru to industry" and by *Business Week* as a "genius". His book, *The Goal*, is a gripping fast-paced business novel.

"*Goal* readers are now doing the best work of their lives." *Success Magazine*

"A factory may be an unlikely setting for a novel, but the book has been wildly effective..." *Tom Peters*



THE BEST-SELLING BUSINESS NOVEL THAT INTRODUCED THE
THEORY OF CONSTRAINTS
AND CHANGED HOW AMERICA DOES BUSINESS

OVER 6 MILLION BOOKS SOLD!

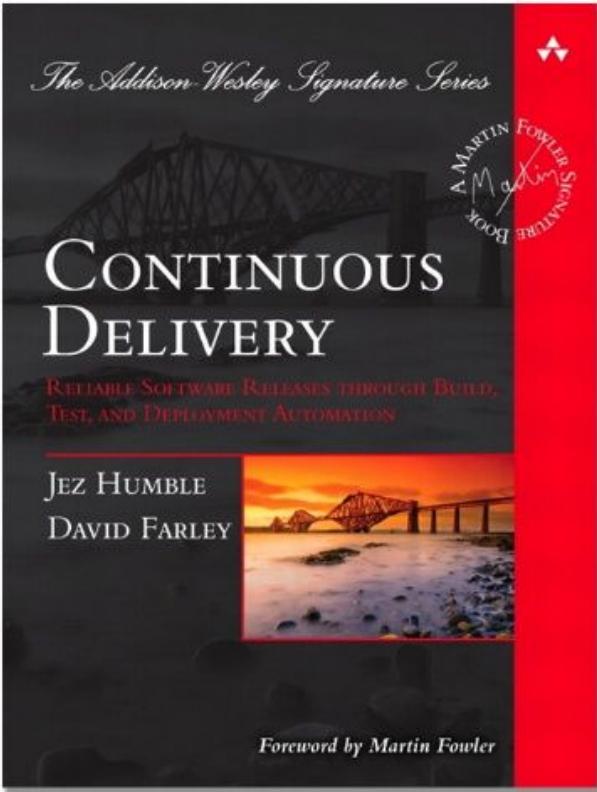
THE NEW YORK TIMES BESTSELLER

THE LEAN STARTUP

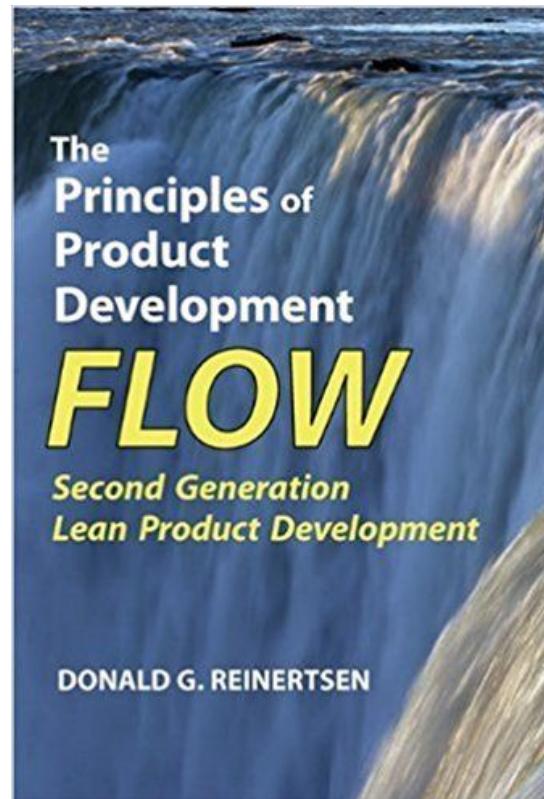
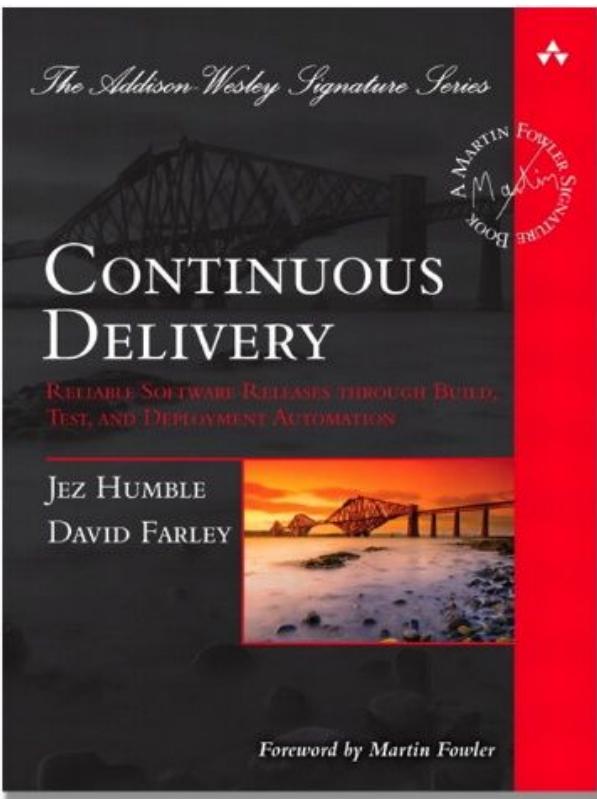
How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses

ERIC RIES

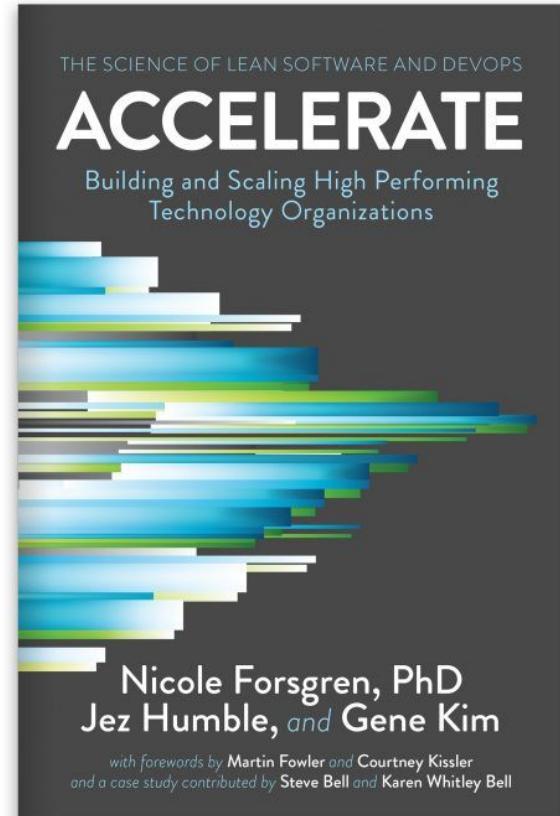
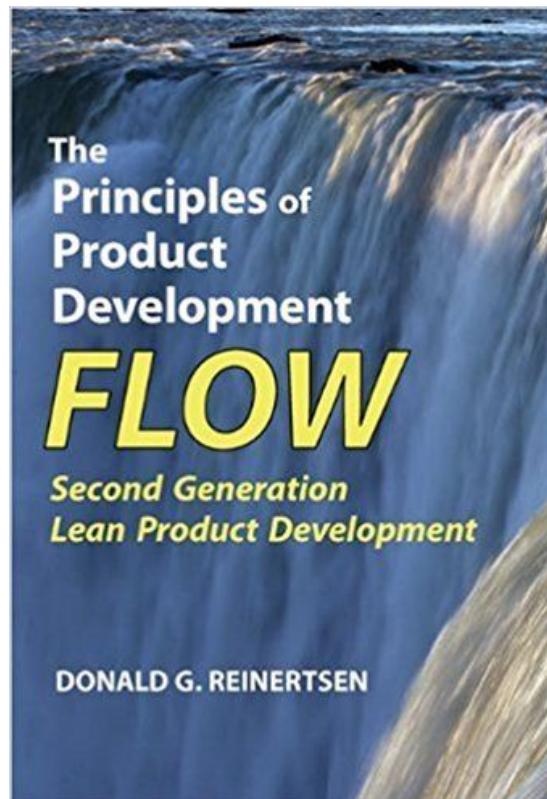
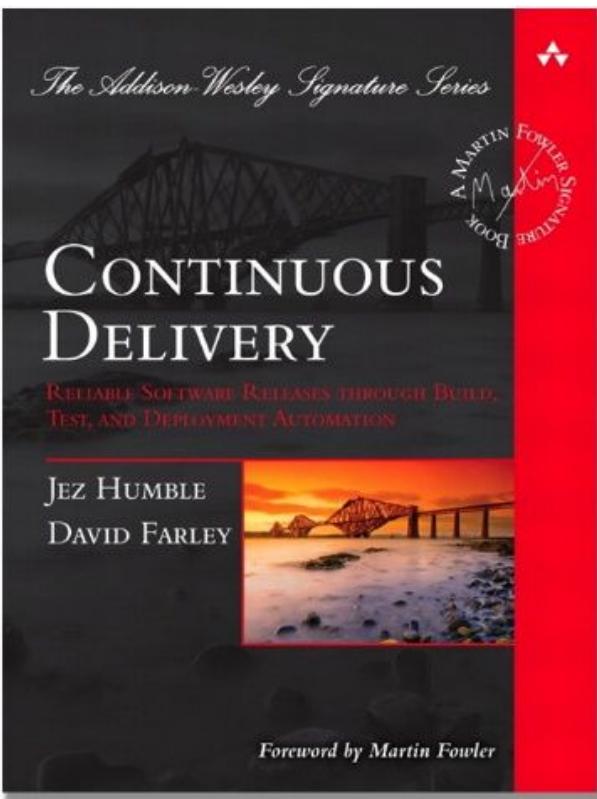
Books!



Books!



Books!



Essential reading

<https://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed>

<https://plus.google.com/+RipRowan/posts/eVeouesvaVX>

<https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying>

<http://martinfowler.com/articles/microservices.html>

<http://jonasboner.com/bla-bla-microservices-bla-bla/>

https://www.youtube.com/watch?v=_EWUeZoyB0k

Good reading

<http://highscalability.com/blog/2014/10/27/microservices-in-production-the-good-the-bad-the-it-works.html>

<http://highscalability.com/blog/2014/4/8/microservices-not-a-free-lunch.html>

<http://www.nearform.com/nodecrunch/microservices-series-4-beware-monolith/>

<https://hbr.org/2007/11/a-leaders-framework-for-decision-making>

<https://www.confluent.io/blog/publishing-apache-kafka-new-york-times/>

<http://milinda.pathirage.org/kappa-architecture.com/>

References

<http://www.slideshare.net/BruceWong3/the-case-for-chaos>

<http://www.slideshare.net/fredgeorge>

<http://www.slideshare.net/dataloop/anne-currie-force12io-game-of-hosts-containers-vs-vms>

<https://www.nrdc.org/sites/default/files/data-center-efficiency-assessment-IB.pdf>

<https://martinjeeblog.com/2012/11/20/what-is-programmer-anarchy-and-does-it-have-a-future/>

<http://static.googleusercontent.com/media/research.google.com/en//people/jeff/Berkeley-Latency-Mar2012.pdf>

<http://apsblog.burtongroup.com/2009/01/soa-is-dead-long-live-services.html>

References cont.

<http://www.ibm.com/developerworks/webservices/library/ar-esbpat1/>

<https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-885j-aircraft-systems-engineering-fall-2005/>

<https://www.fogcreek.com/blog/eight-fallacies-of-distributed-computing-tech-talk/>

<http://www.ibiblio.org/harris/500milemail.html>

<http://www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/2015-Annual-Report.pdf>

<http://www.counterpunch.org/2013/03/04/when-money-is-no-object-the-strange-saga-of-the-f-35/>

References cont.

<https://www.youtube.com/watch?v=dxk8b9rSKOo>

<http://assets.en.oreilly.com/1/event/60/Velocity%20Culture%20Presentation.pdf>

<http://www.cubrid.org/blog/dev-platform/understanding-tcp-ip-network-stack/>

<https://www.oreilly.com/ideas/an-introduction-to-immutable-infrastructure>

<http://www.methodsandtools.com/archive/archive.php?id=97>

<http://redmonk.com/sogrady/2017/07/20/soa-microservices/>

Bad things

<http://www.ibm.com/developerworks/webservices/library/ar-esbp1/>

<http://www.networkworld.com/article/3114195/system-management/the-8-fallacies-of-distributed-computing-are-becoming-irrelevant.html>

Image credits

<http://fontawesome.io/>

<https://github.com/thepracticaldev/orly-full-res>

<http://dimaip.github.io/slides/docker101.html>

<http://www.clipartlord.com/category/military-clip-art/bomb-clip-art/explosion-clip-art/>

https://en.wikipedia.org/wiki/File:S-IC_engines_and_Von_Braun.jpg

http://www.nutsvolts.com/magazine/article/the_computer_that_took_man_to_the_moon

<https://spaceflightnow.com/2014/11/05/engineers-recommend-changes-to-orion-heat-shield/>

<https://www.hq.nasa.gov/alsj/alsj-DrinkFood.html>

Image credits cont.

<https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19720013196.pdf>

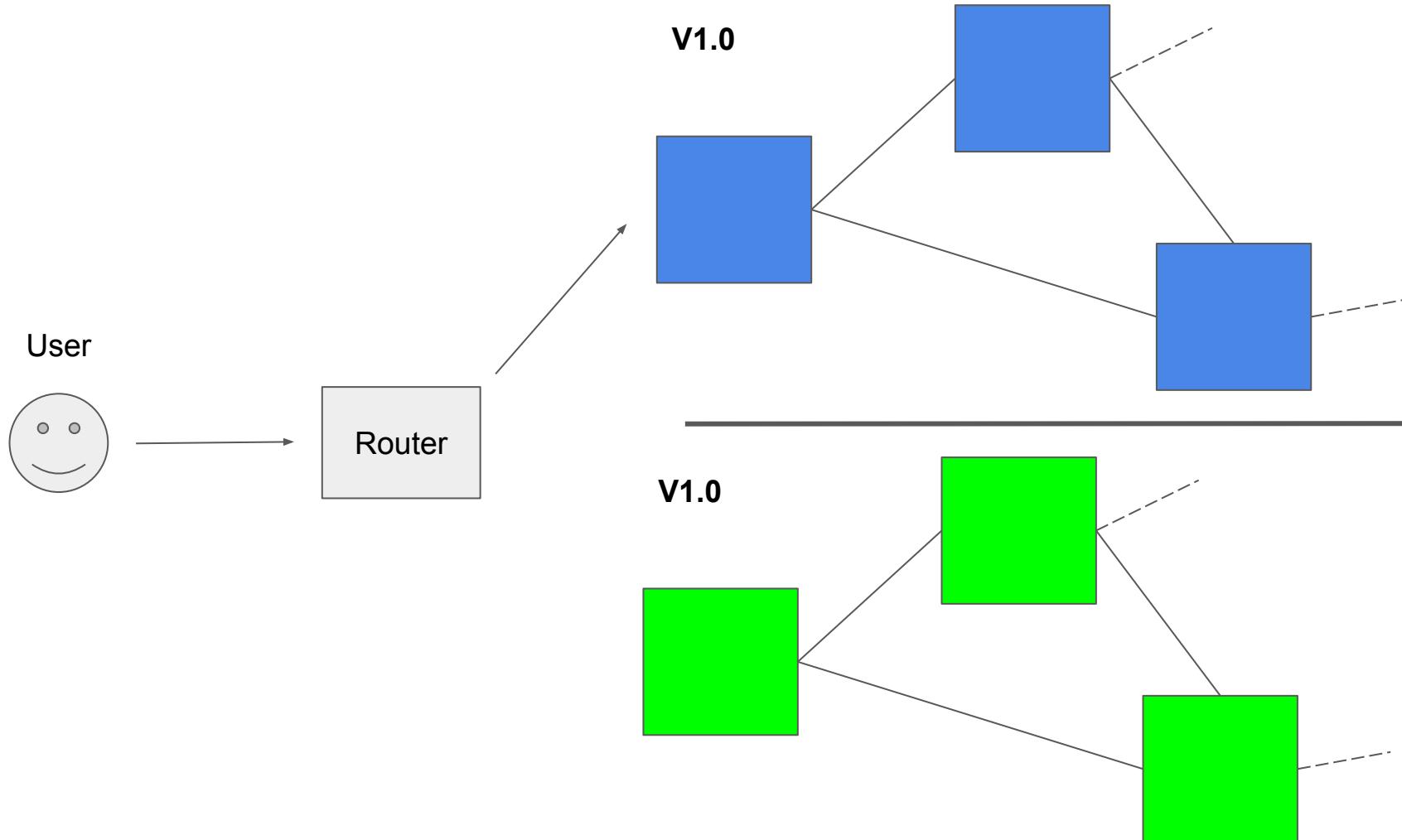
Extra material

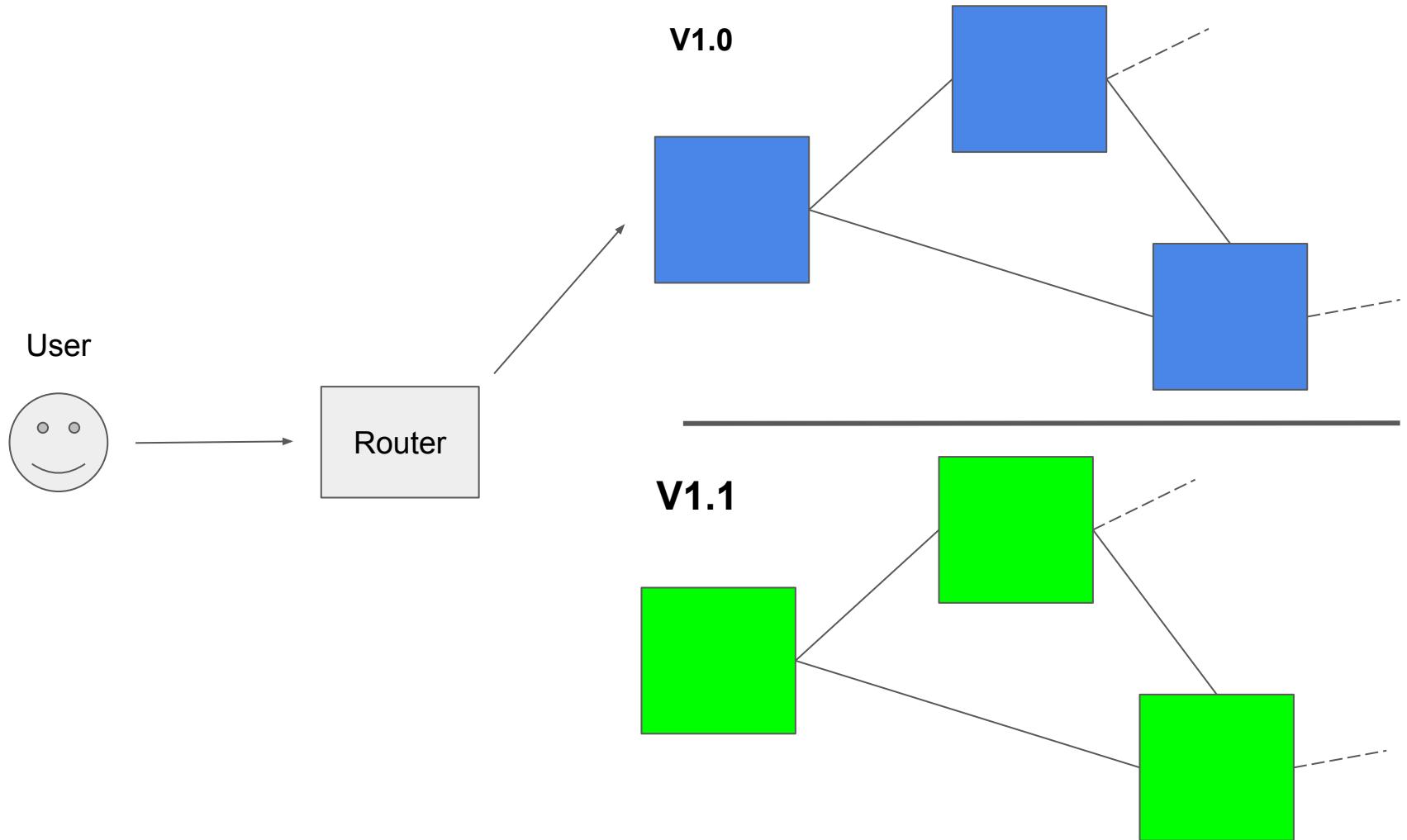
Blue-Green Deployment

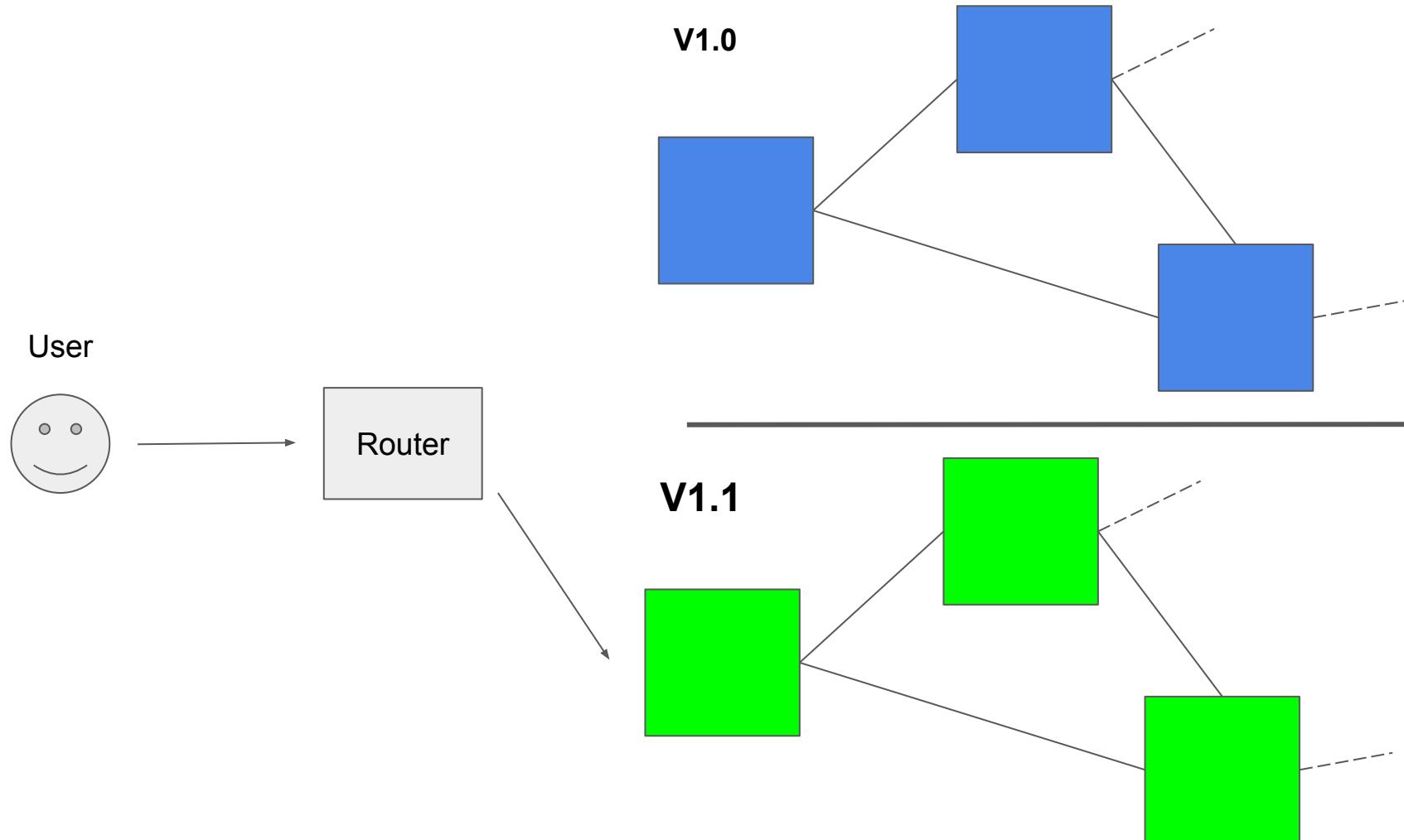
ESBs are shit

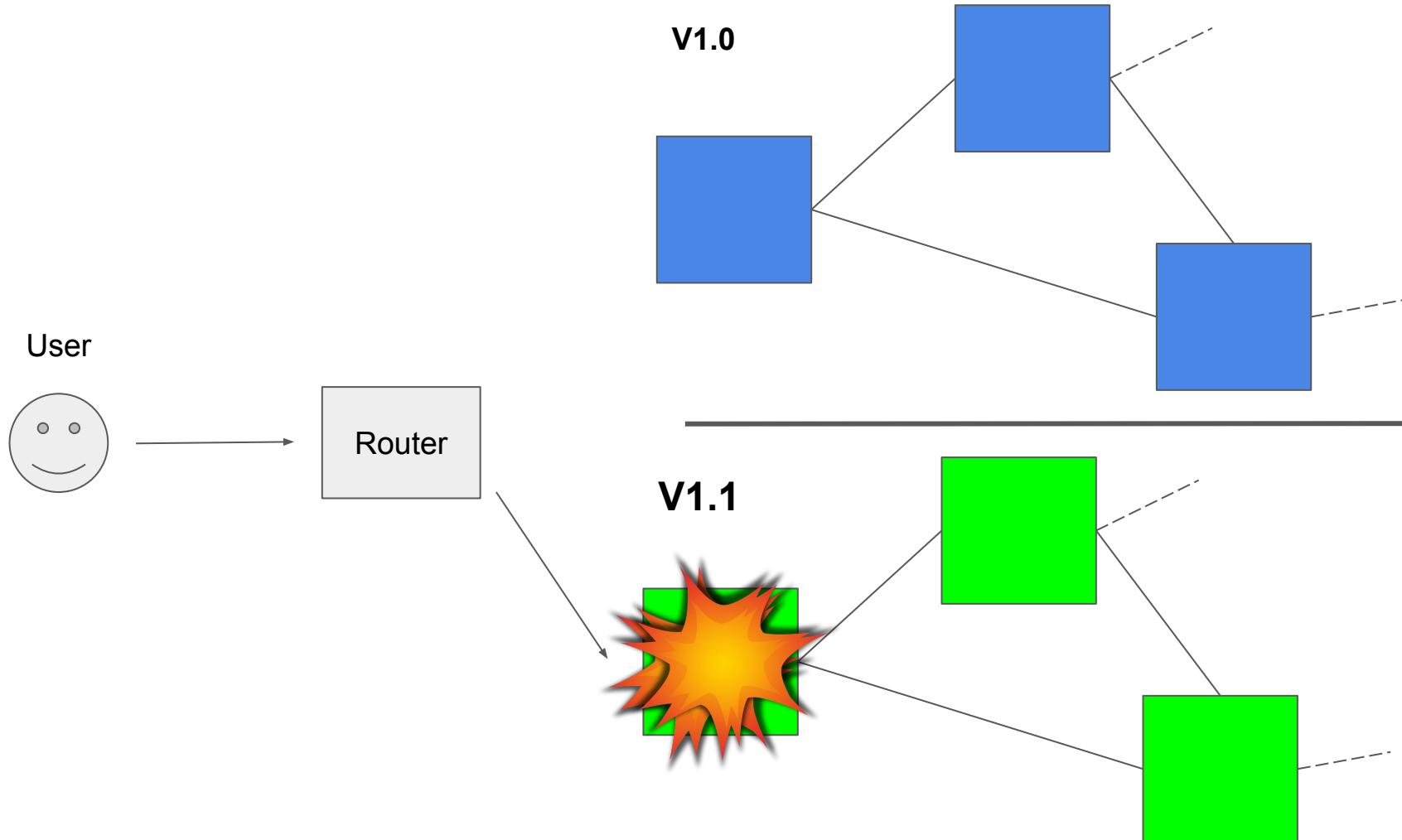
Language choice for Microservices

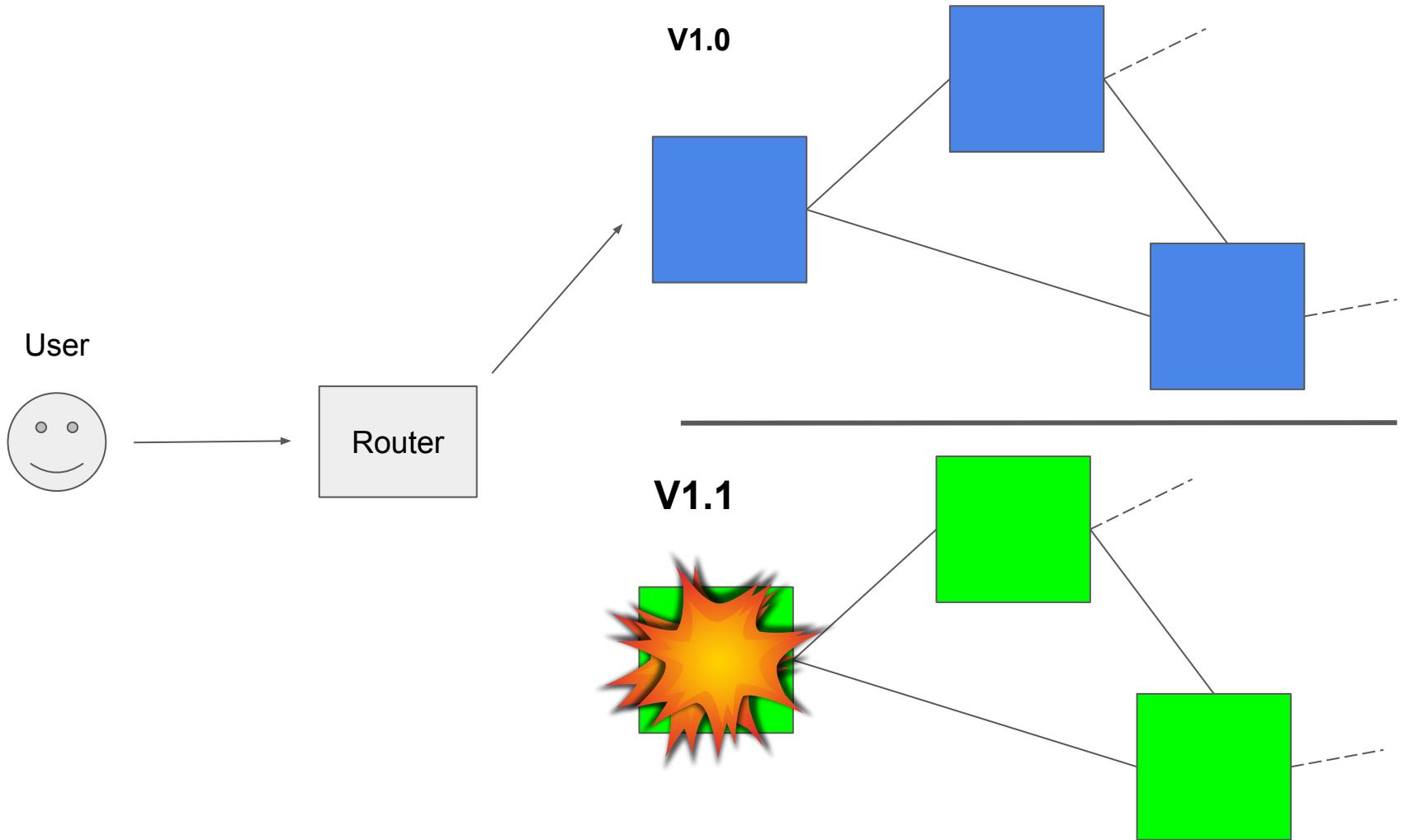
Blue-Green deployment









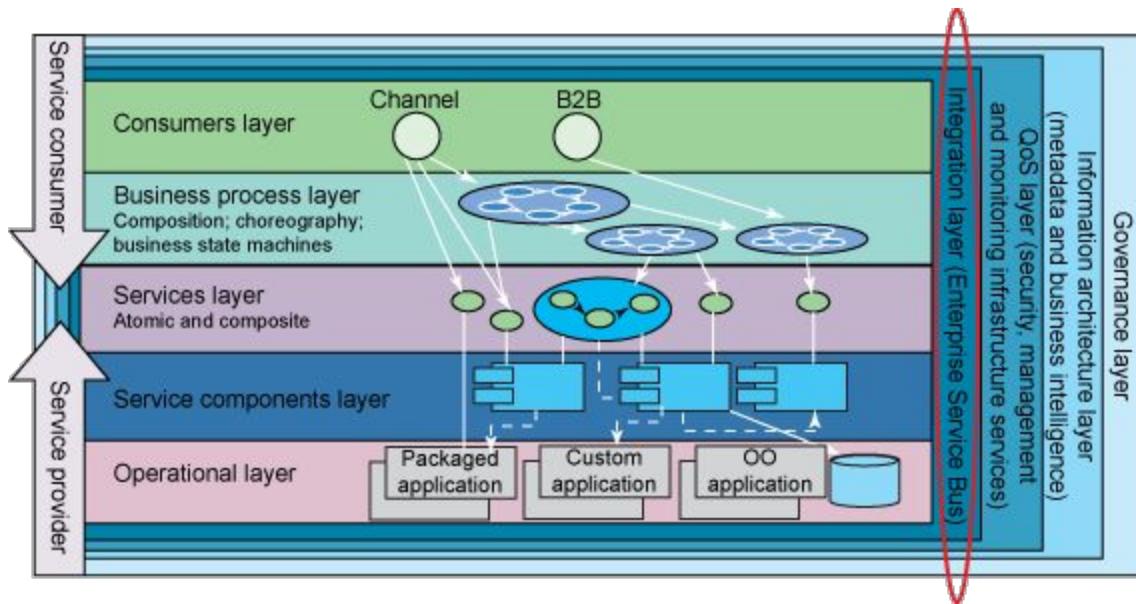


Possible with monoliths, just **harder!**

ESBs are shit

ESBs are shit
credit to Russ Miles





Service

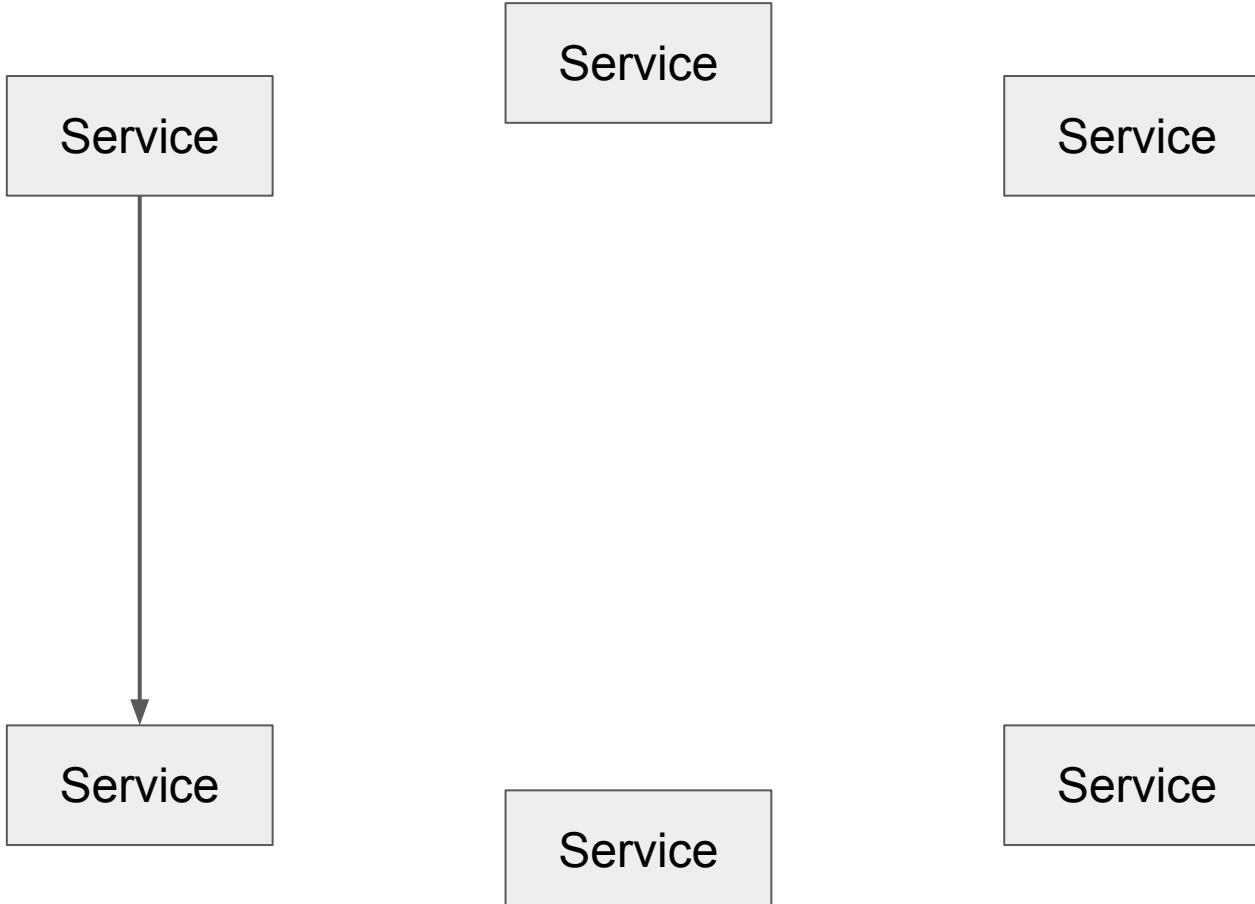
Service

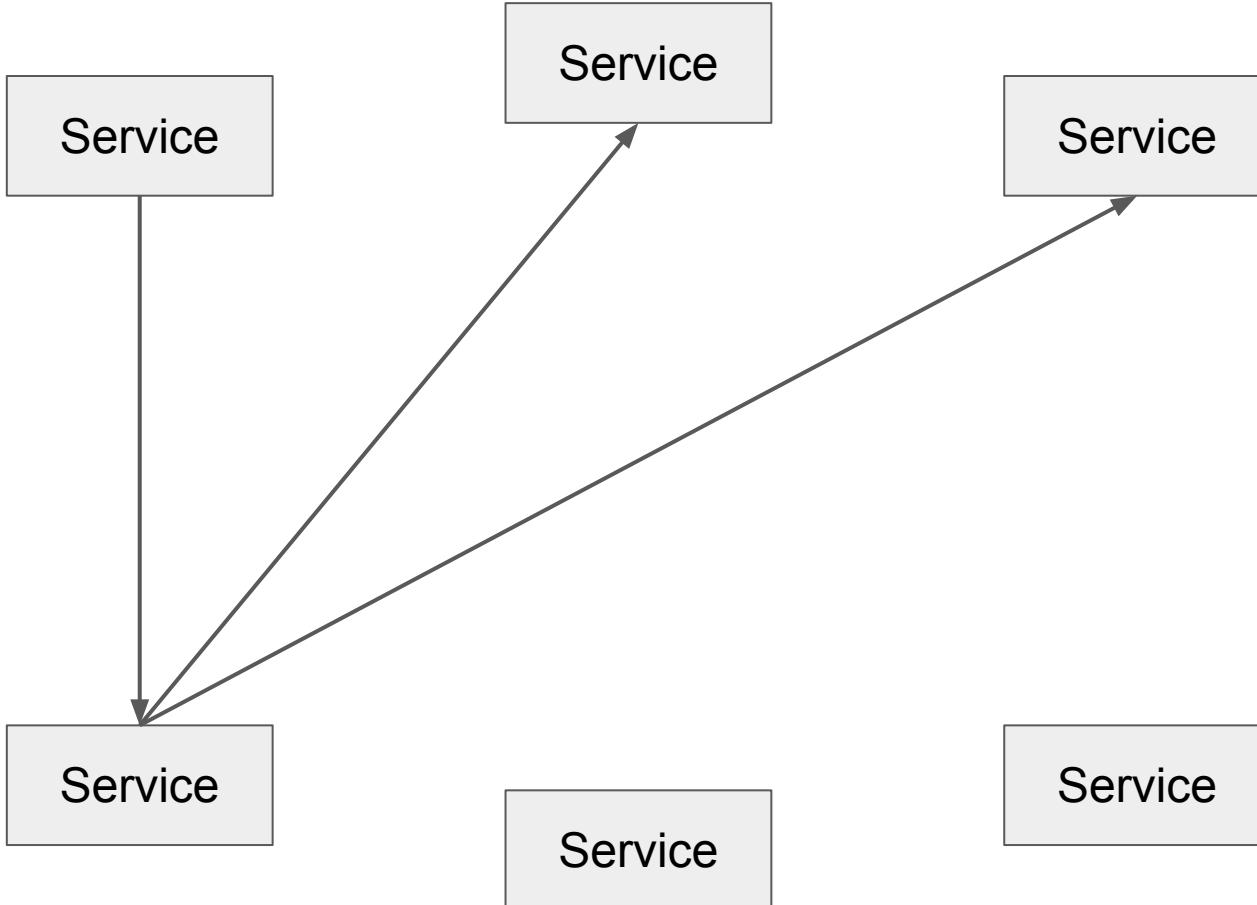
Service

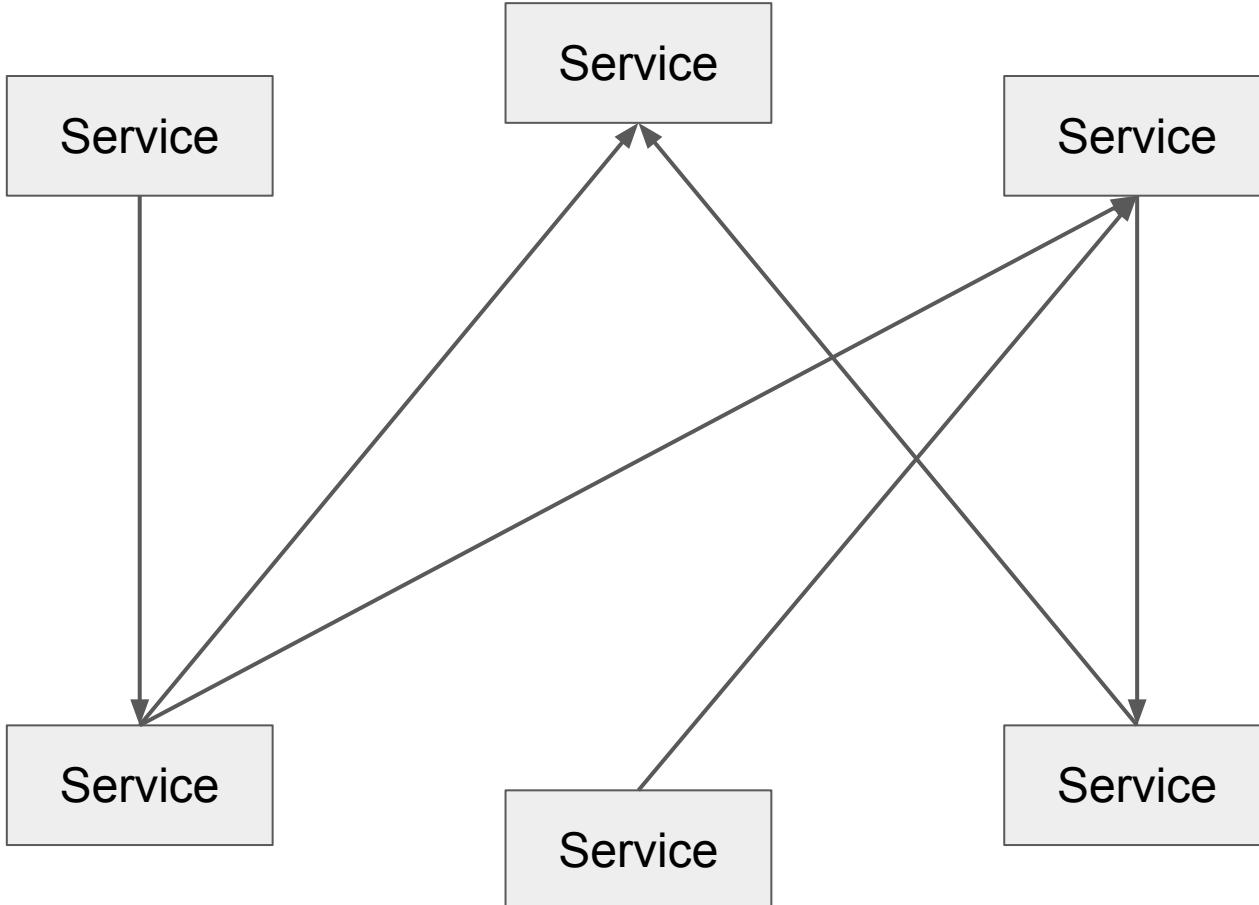
Service

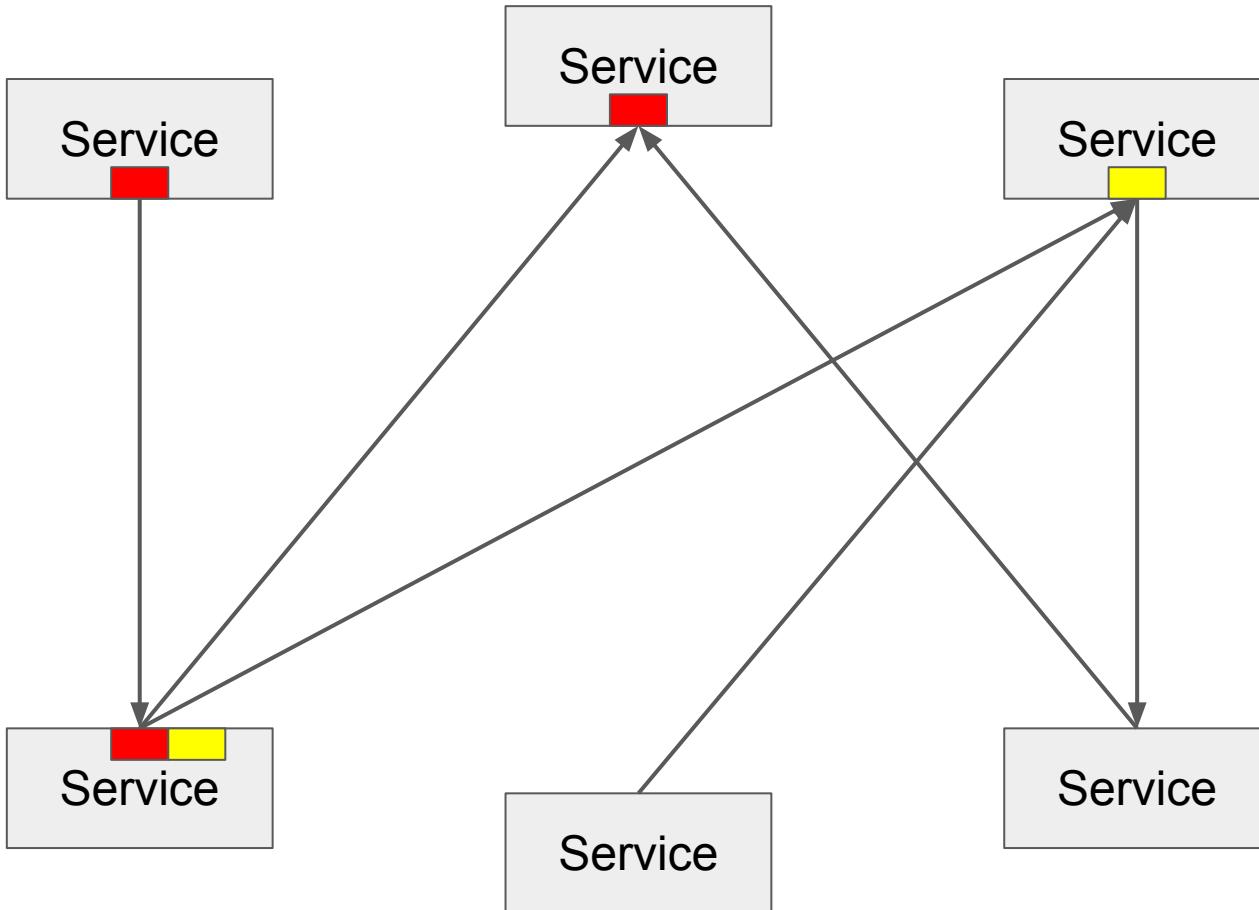
Service

Service









Smart endpoints and dumb pipes

Dumb Diagrams

