# *The Evolution of Microservices*

**Adrian Cockcroft @adrianco**
**Technology Fellow - Battery Ventures**
June 2016

Battery Ventures

# What does @adrianco do?

- Presentations at Conferences
- Maintain Relationship with Cloud Vendors
- Technology Due Diligence on Deals
- Presentations at Companies
- @adrianco
  Battery Ventures
- Technical Advice for Portfolio Companies
- Program Committee for Conferences
- Tinkering with Technologies
- Networking with Interesting People

*Previously: Netflix, eBay, Sun Microsystems, CCL, TCU London BSc Applied Physics*

## Agenda

*Why now?*

*Microservice Architectures*

*What's Missing?*

*Migration and Simulation*

*What's Next?*

*Key Goals of the CIO?*
*Align IT with the business*
*Develop products faster*
*Try not to get breached*

## Security Blanket Failure

*Insecure applications hidden behind firewalls make you feel safe until the breach happens…*

http://peanuts.wikia.com/wiki/Linus'_security_blanket

## “Web scale” vs. “Enterprise”

*“Webscale”*

*Freedom and responsibility*
*High trust*

## “Enterprise”

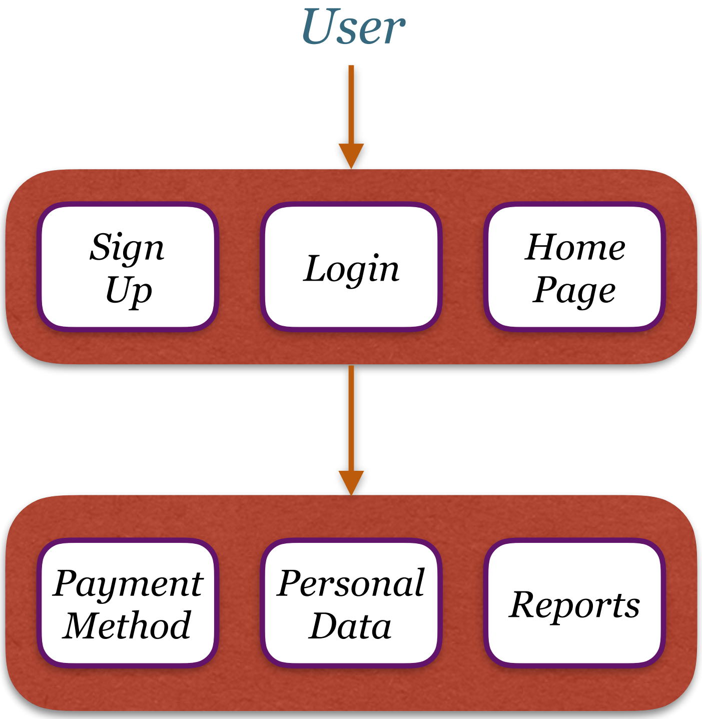
Bureaucracy and blame

Low trust

## *How can everyone get speed, low cost, and better usability?*

## *Mixed methods: Disaggregation into microservices helps!*

# Example Monolith:



*Monolithic application*

*Complex mix of queries*

*Monolithic "kitchen sink" database*

*Because one part of the monolithic application and database holds sensitive data all of it is subject to the most rigorous policies*

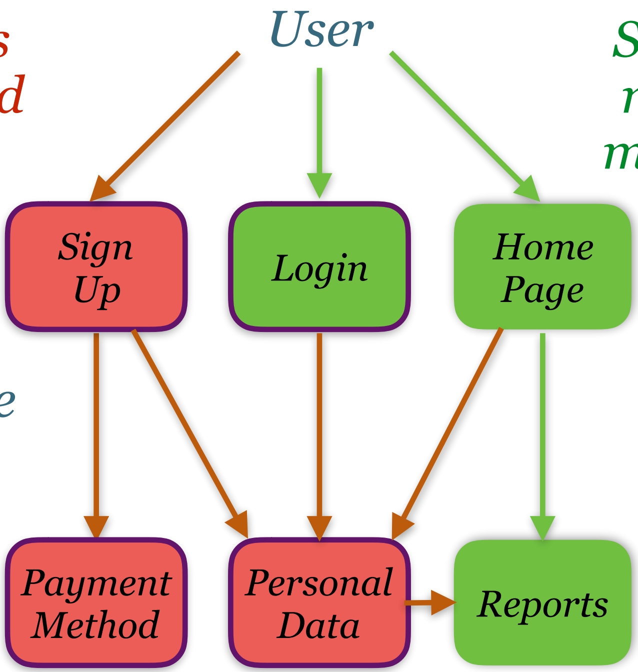
# Microservices version:

*Segregated team owns secure data sources and infrequent updates*

*Microservices separation of concerns*

*Isolated single purpose connections*

*Optimized datastores*



*Segregated team owns rapid improvement of most common use cases*

*Because each microservice can conform to the appropriate policy, demands for agility can be separated from requirements for security*

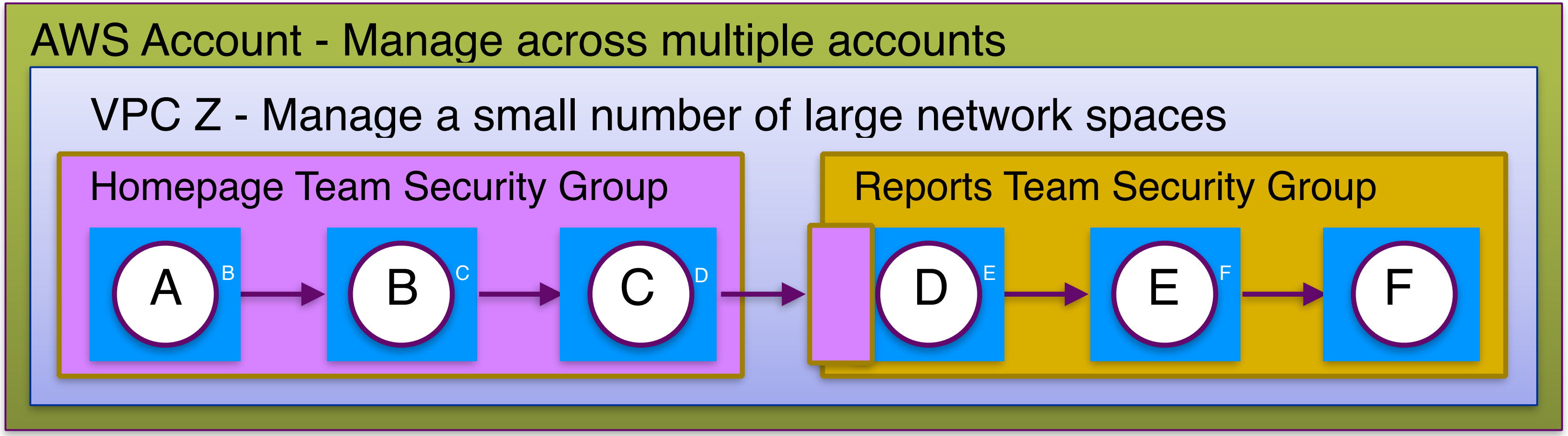
## In Search of Segmentation

Ops

|  |  |
|---|---|
| Datacenters | AWS Accounts |
| AD/LDAP Roles | IAM Roles |
| VLAN Networks | VPC |
| Hypervisor | Security Groups |
| IPtables | Calico Policy |
| Docker Links | Docker Net/Weave |

Dev

## *Hierarchical Segmentation*



AWS Account - Manage across multiple accounts
VPC Z - Manage a small number of large network spaces
Homepage Team Security Group
Reports Team Security Group
A
B
B
C
C
D
D
E
E
F
F

*An AWS oriented example…*

@adrianco

BV Battery Ventures

*“You build it, you run it.”*

*Werner Vogels 2006*

## *Developer responsibilities: Faster, cheaper, safer*

## Speeding Up The Platform

**Datacenter Snowflakes**

- Deploy in months
- Live for years

## Speeding Up The Platform

**Datacenter Snowflakes**
- Deploy in months
- Live for years

**Virtualized and Cloud**
- Deploy in minutes
- Live for weeks

# Speeding Up The Platform

**Datacenter Snowflakes**
- Deploy in months
- Live for years

**Virtualized and Cloud**
- Deploy in minutes
- Live for weeks

**Container Deployments**
- Deploy in seconds
- Live for minutes/hours

## Speeding Up The Platform

**Datacenter Snowflakes**
- Deploy in months
- Live for years

**Virtualized and Cloud**
- Deploy in minutes
- Live for weeks

**Container Deployments**
- Deploy in seconds
- Live for minutes/hours

**Lambda Deployments**
- Deploy in milliseconds
- Live for seconds

## Speeding Up The Platform

**Datacenter Snowflakes**
- Deploy in months
- Live for years

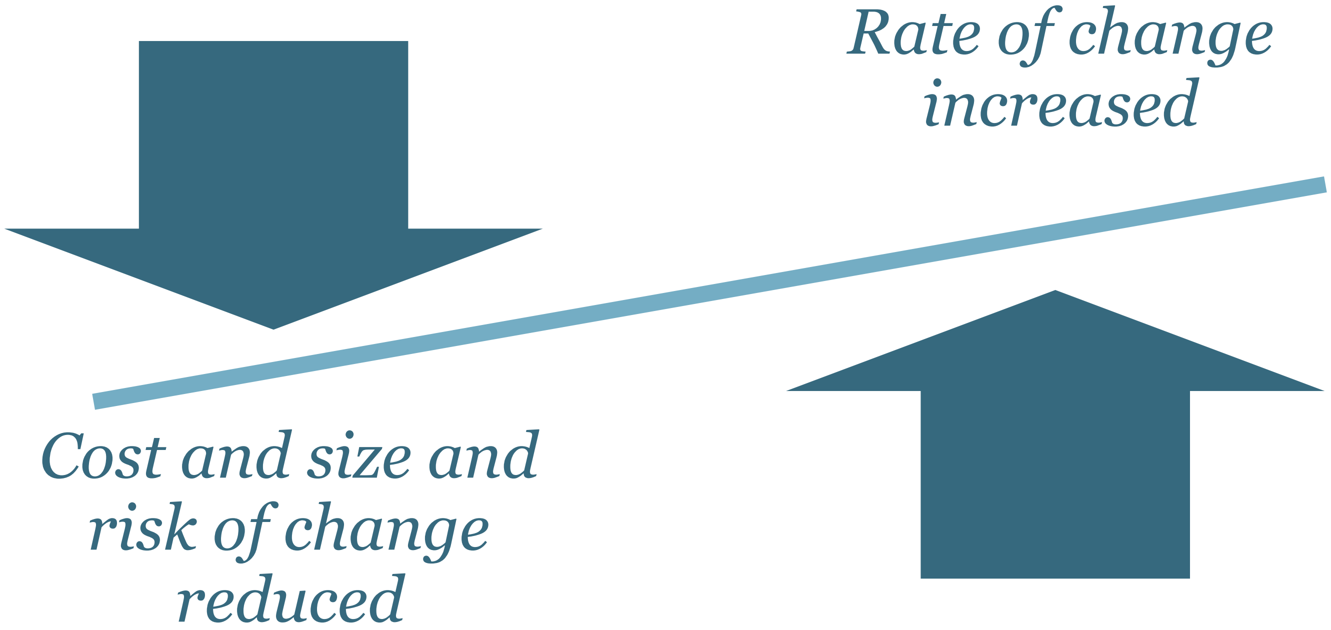
**Virtualized and Cloud**
- Deploy in minutes
- Live for weeks

**Container Deployments**
- Deploy in seconds
- Live for minutes/hours

**Lambda Deployments**
- Deploy in milliseconds
- Live for seconds

*AWS Lambda is leading exploration of serverless architectures in 2016*

## What Happened?


*Rate of change increased*
*Cost and size and risk of change reduced*

## *Microservices*

## *A Microservice Definition*

*Loosely coupled service oriented architecture with bounded contexts*

*If every service has to be updated at the same time it's not loosely coupled*

## *A Microservice Definition*

*Loosely coupled service oriented architecture with bounded contexts*

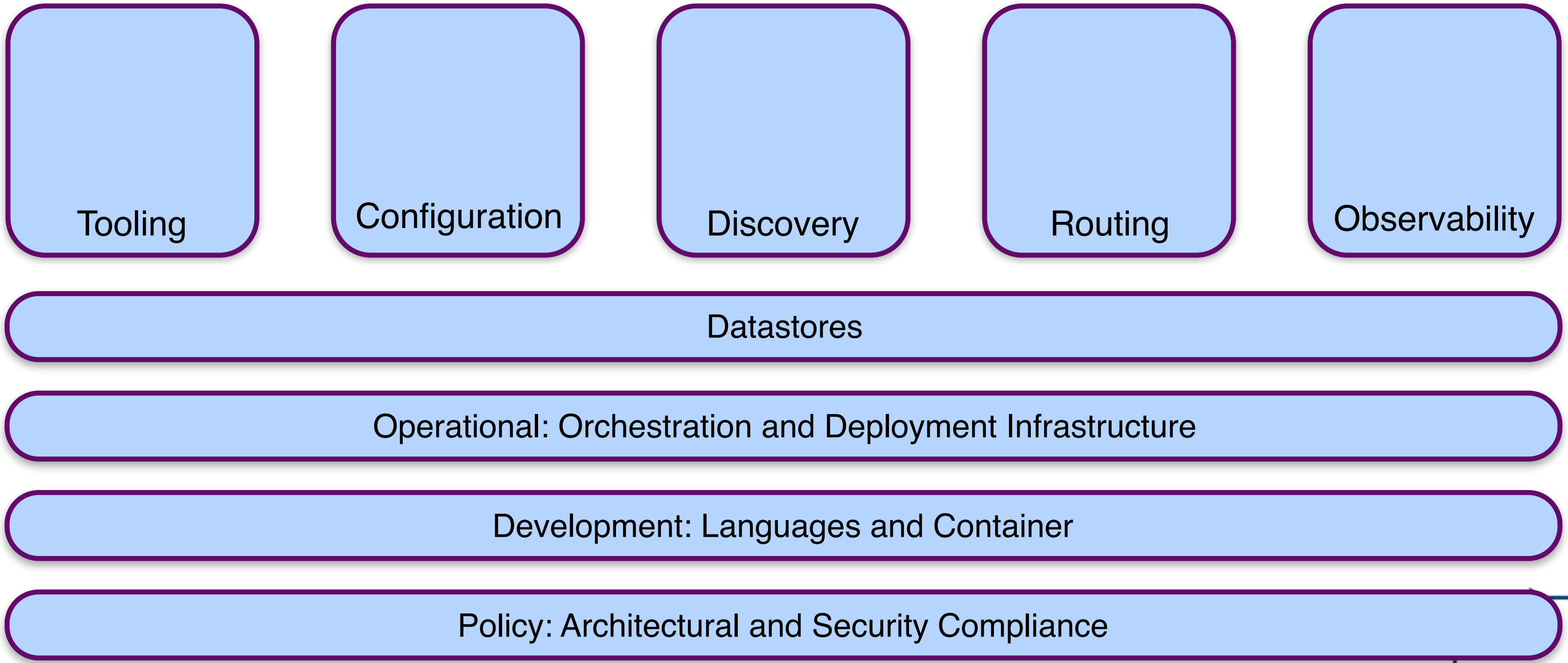
*If every service has to be updated at the same time it's not loosely coupled*

## *A Microservice Definition*

*Loosely coupled service oriented architecture with bounded contexts*

*If you have to know too much about surrounding services you don't have a bounded context. See the Domain Driven Design book by Eric Evans.*

# Microservice Architectures


Tooling
Configuration
Discovery
Routing
Observability
Datastores
Operational: Orchestration and Deployment Infrastructure
Development: Languages and Container
Policy: Architectural and Security Compliance

# Next Generation Applications

*Fill in the gaps, rapidly evolving ecosystem choices*

Lambda
Docker
Spinnaker

Tooling

Archaius
LaunchDarkly
Habitat

Configuration

Etcd
Eureka
Consul

Discovery

Compose
Linkerd
Weave

Routing

Zipkin
Prometheus
Hystrix

Observability

Datastores: Orchestrated, Distributed Ephemeral e.g. Cassandra, or DBaaS e.g. DynamoDB

Operational: Mesos, Kubernetes, Swarm, Nomad for private clouds. ECS, Mesos, GKS for public

Development: components interfaces languages e.g. Docker Hub, Artifactory, Datawire Quark, Go, Rust

Policy: Security compliance e.g. Docker Content Trust. Architecture compliance e.g. Cloud Foundry

## *What could go wrong?*

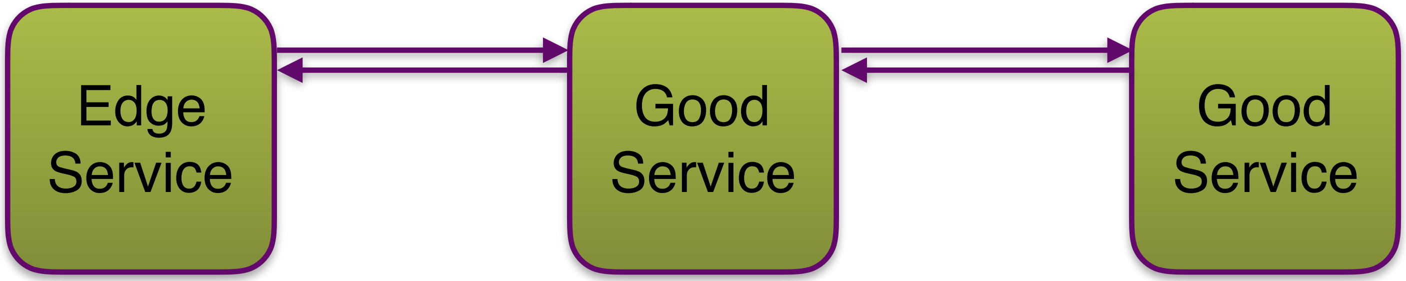
# Timeouts and Retries

*Bad config: Every service defaults to 2 second timeout, two retries*

Edge Service
Good Service
Good Service

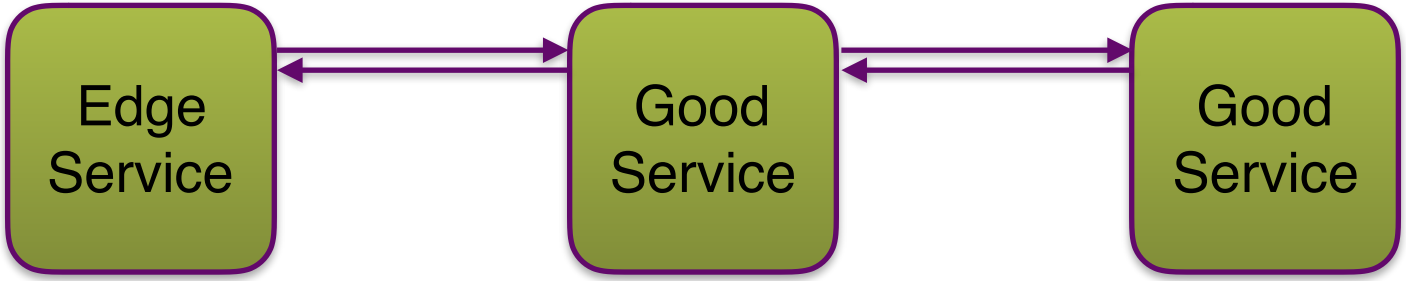
# Timeouts and Retries

*Bad config: Every service defaults to 2 second timeout, two retries*

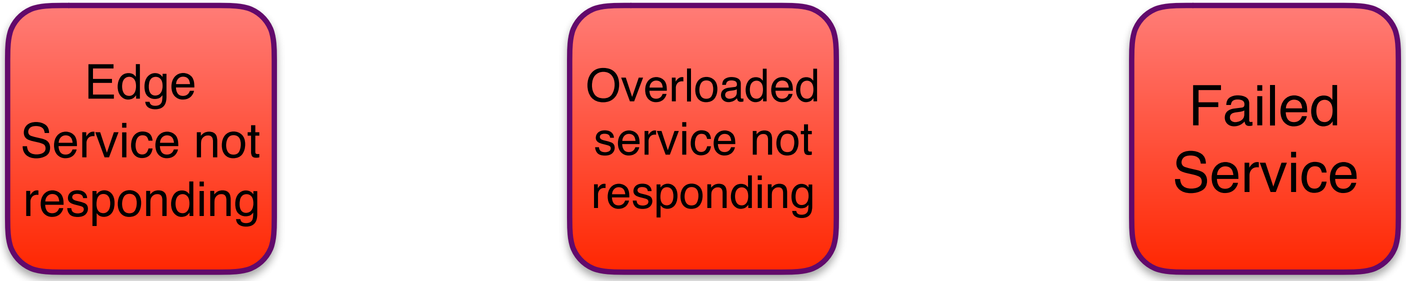


# Timeouts and Retries

*Bad config: Every service defaults to 2 second timeout, two retries*



*If anything breaks, everything upstream stops responding*

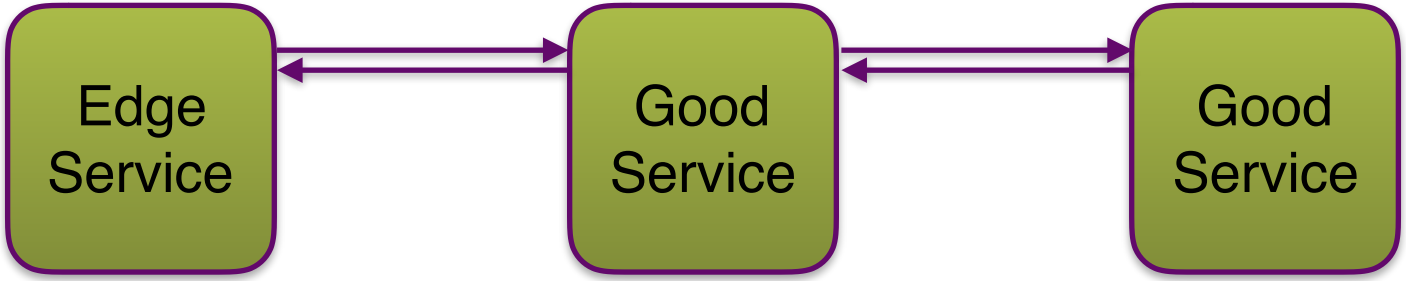

Edge Service not responding
Overloaded service not responding
Failed Service

@adrianco

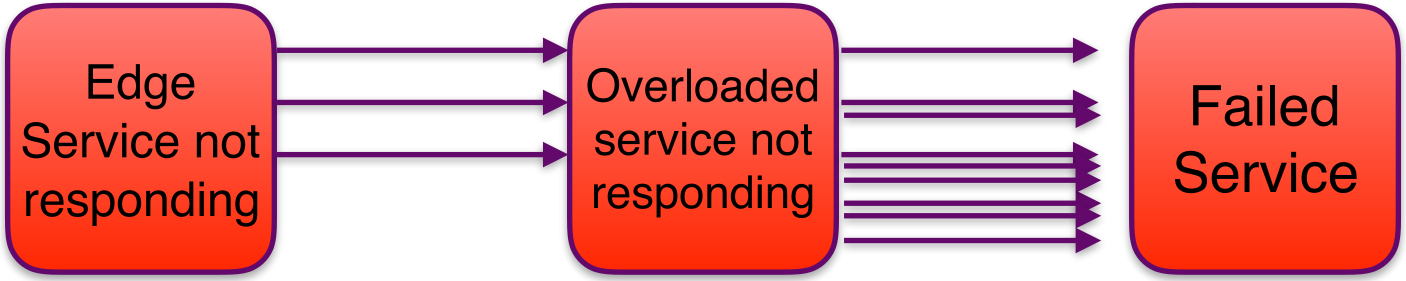
BV Battery Ventures

# Timeouts and Retries

*Bad config: Every service defaults to 2 second timeout, two retries*



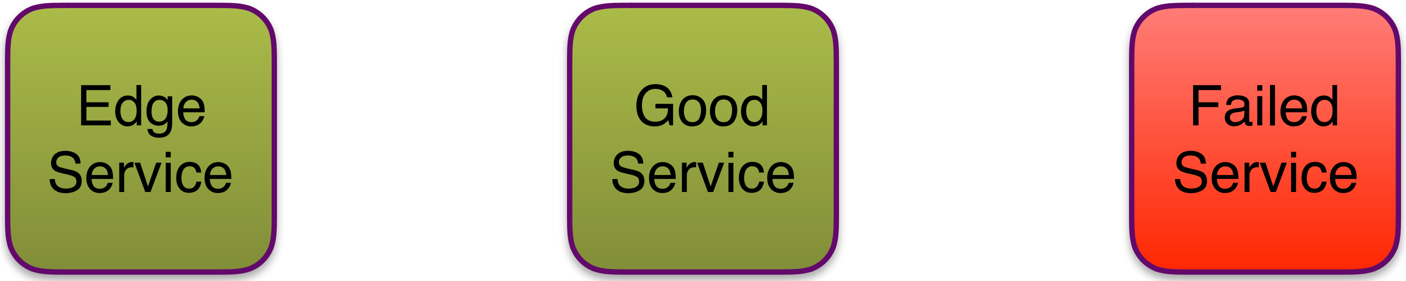
*If anything breaks, everything upstream stops responding*


Edge Service not responding
Overloaded service not responding
Failed Service

*Retries add unproductive work*

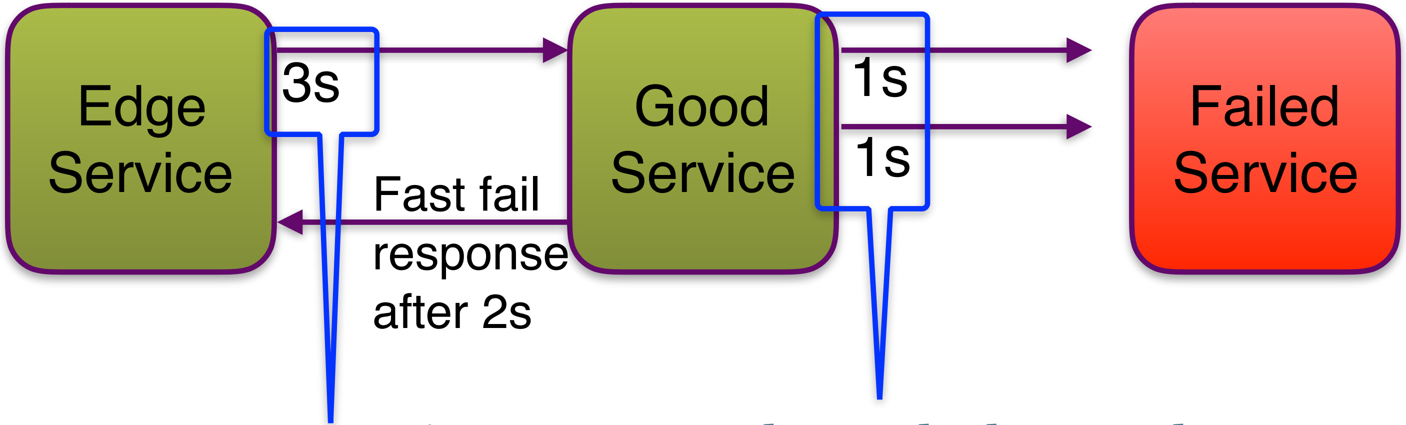
# Timeouts and Retries

*Budgeted timeout, one retry*



# Timeouts and Retries

*Budgeted timeout, one retry*

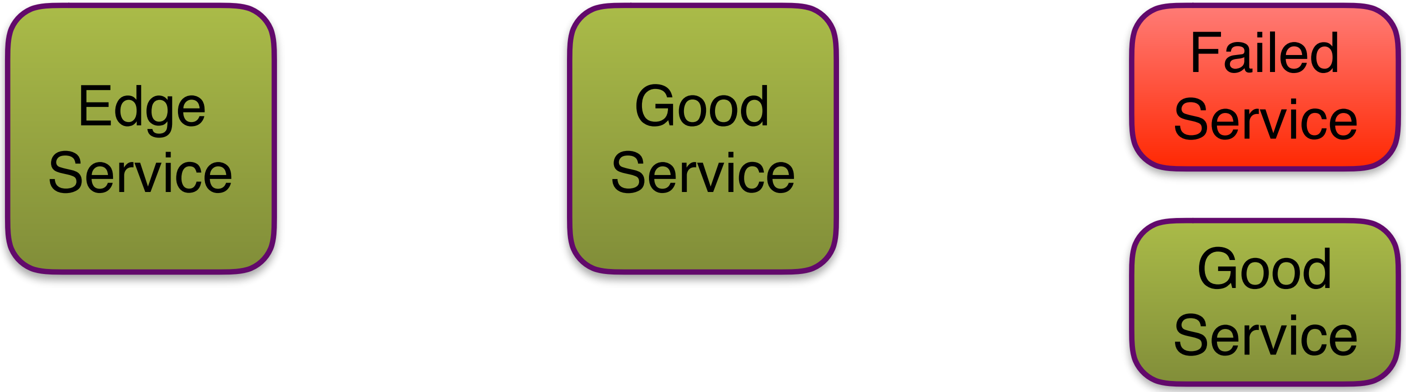


*Upstream timeout must always be longer than total downstream timeout * retries delay*

*No unproductive work while fast failing*

## Timeouts and Retries

*Budgeted timeout, failover retry*

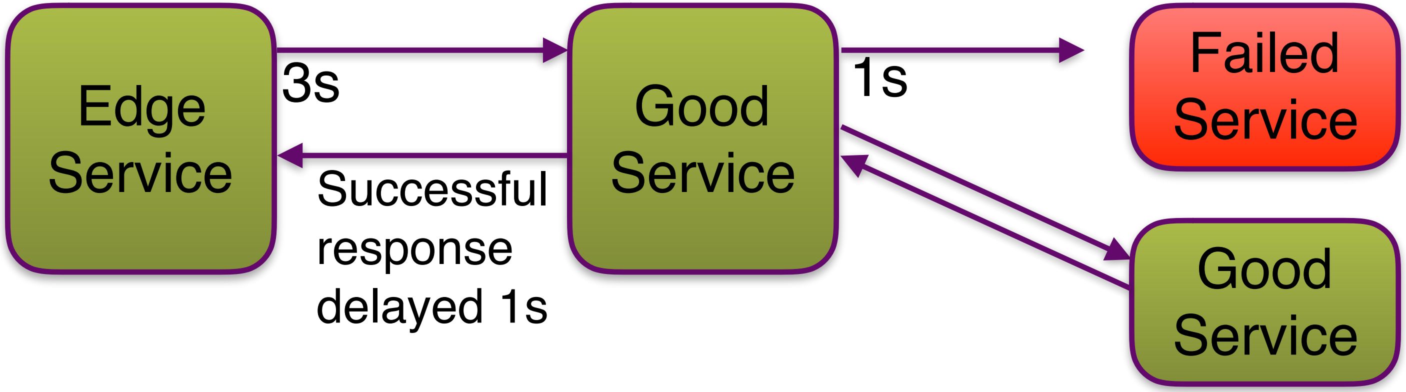


*For replicated services with multiple instances never retry against a failed instance*

*No extra retries or unproductive work*

# Timeouts and Retries

*Budgeted timeout, failover retry*

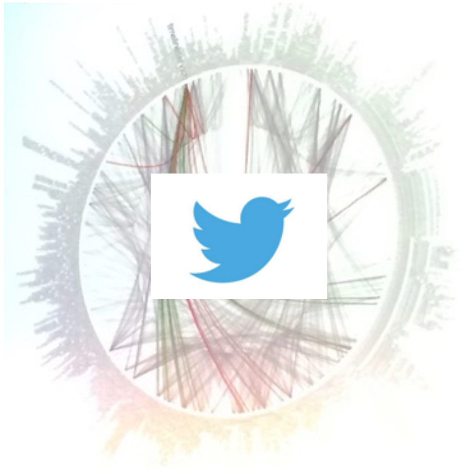


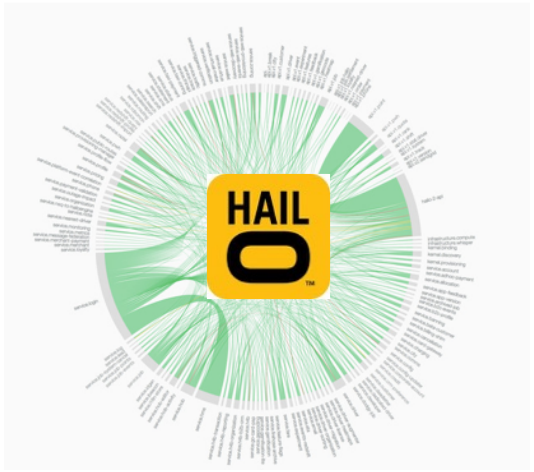
*For replicated services with multiple instances never retry against a failed instance*

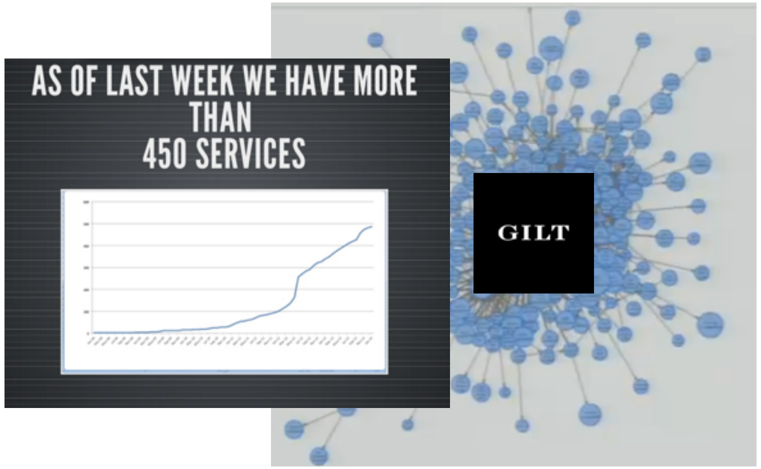
*No extra retries or unproductive work*

# Cloud Native Monitoring and Microservices

## *Interesting architectures have a lot of microservices! Flow visualization is a big challenge.*

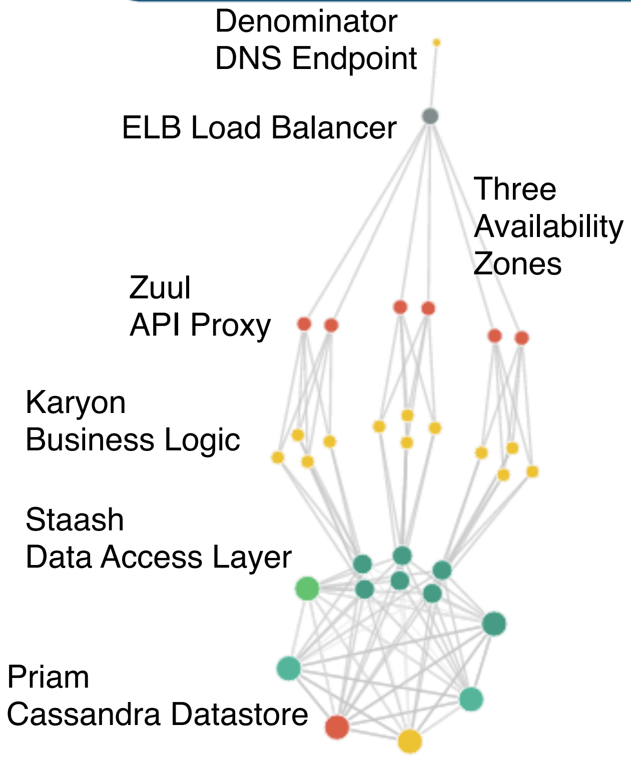






See http://www.slideshare.net/LappleApple/gilt-from-monolith-ruby-app-to-micro-service-scala-service-architecture

# Simulated Microservices



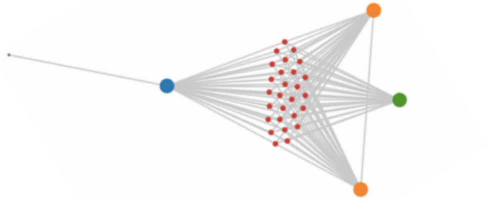
*Model and visualize microservices*
*Simulate interesting architectures*
*Generate large scale configurations*
*Eventually stress test real tools*

*Code: [github.com/adrianco/spigo](github.com/adrianco/spigo)*
*Simulate Protocol Interactions in Go*
*Visualize with D3*
*See for yourself: [http://simianviz.surge.sh](http://simianviz.surge.sh)*
*Follow @simianviz for updates*

# Definition of an architecture

Header includes chaos monkey victim

*See for yourself:* http://simianviz.surge.sh/lamp



```
{
    "arch": "lamp",
    "description":"Simple LAMP stack",
    "version": "arch-0.0",
    "victim": "webserver",
    "services": [
        { "name": "rds-mysql",     "package": "store",       "count": 2,  "regions": 1, "dependencies": [] },
        { "name": "memcache",      "package": "store",       "count": 1,  "regions": 1, "dependencies": [] },
        { "name": "webserver",     "package": "monolith",    "count": 18, "regions": 1, "dependencies": ["memcache", "rds-mysql"] },
        { "name": "webserver-elb", "package": "elb",         "count": 0,  "regions": 1, "dependencies": ["webserver"] },
        { "name": "www",           "package": "denominator", "count": 0,  "regions": 0, "dependencies": ["webserver-elb"] }
    ]
}
```

New tier name

Tier package

Node count

0 = non Regional

List of tier dependencies

# Running Spigo

-a  architecture lamp
-j   graph json/lamp.json
-d  run for 2 seconds

```
$ ./spigo -a lamp -j -d 2
2016/01/26 23:04:05 Loading architecture from json_arch/lamp_arch.json
2016/01/26 23:04:05 lamp.edda: starting
2016/01/26 23:04:05 Architecture: lamp Simple LAMP stack
2016/01/26 23:04:05 architecture: scaling to 100%
2016/01/26 23:04:05 lamp.us-east-1.zoneB.eureka01....eureka.eureka: starting
2016/01/26 23:04:05 lamp.us-east-1.zoneA.eureka00....eureka.eureka: starting
2016/01/26 23:04:05 lamp.us-east-1.zoneC.eureka02....eureka.eureka: starting
2016/01/26 23:04:05 Starting: {rds-mysql     store 1 2 []}
2016/01/26 23:04:05 Starting: {memcache     store 1 1 []}
2016/01/26 23:04:05 Starting: {webserver     monolith 1 18 [memcache rds-mysql]}
2016/01/26 23:04:05 Starting: {webserver-elb     elb 1 0 [webserver]}
2016/01/26 23:04:05 Starting: {www     denominator 0 0 [webserver-elb]}
2016/01/26 23:04:05 lamp.*.*.www00....www.denominator activity rate  10ms
2016/01/26 23:04:06 chaosmonkey delete: lamp.us-east-1.zoneC.webserver02....webserver.monolith
2016/01/26 23:04:07 asgard: Shutdown
2016/01/26 23:04:07 lamp.us-east-1.zoneB.eureka01....eureka.eureka: closing
2016/01/26 23:04:07 lamp.us-east-1.zoneA.eureka00....eureka.eureka: closing
2016/01/26 23:04:07 lamp.us-east-1.zoneC.eureka02....eureka.eureka: closing
2016/01/26 23:04:07 spigo: complete
2016/01/26 23:04:07 lamp.edda: closing
```

## Open Zipkin

*A common format for trace annotations*

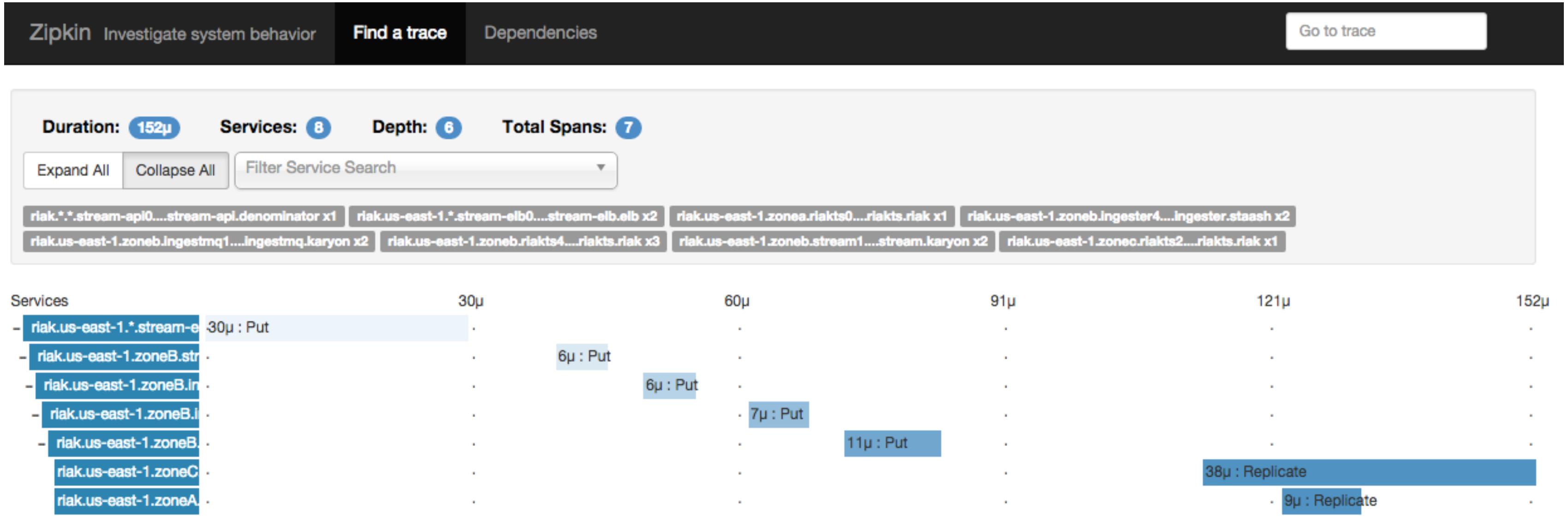
*A Java tool for visualizing traces*

*Standardization effort to fold in other formats*

*Driven by Adrian Cole (currently at Pivotal)*

*Extended to load Spigo generated trace files*

# Trace for one Spigo Flow



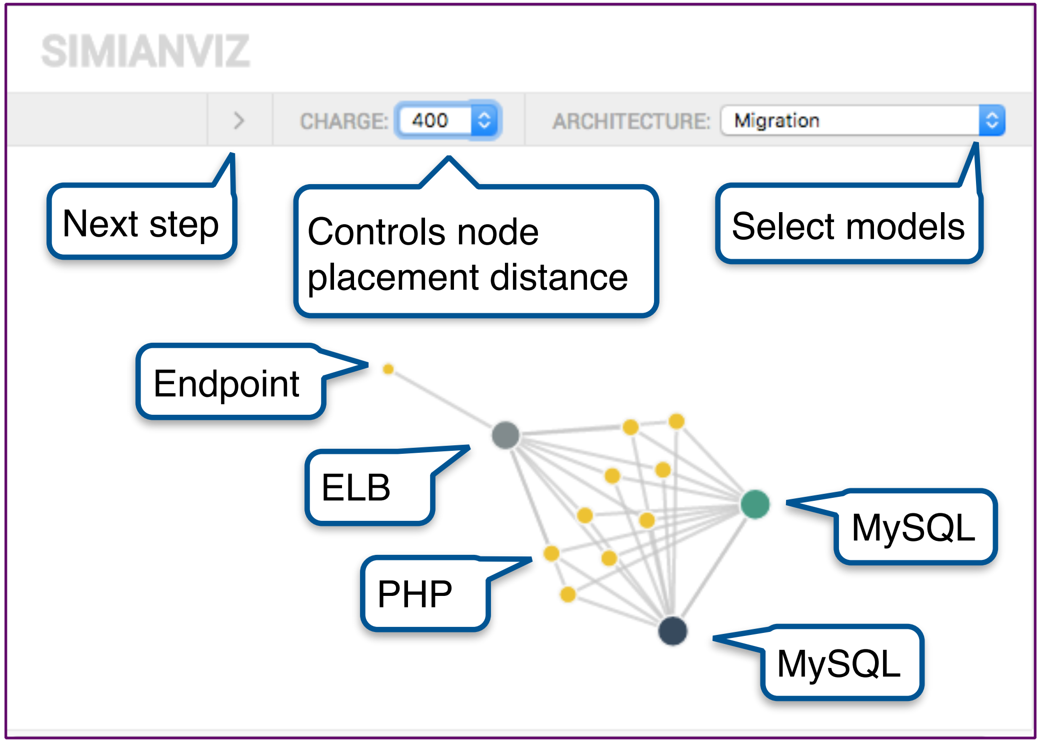
**Duration:** 152μ **Services:** 8 **Depth:** 6 **Total Spans:** 7

riak.*.*.stream-api0....stream-api.denominator x1 | riak.us-east-1.*.stream-elb0....stream-elb.elb x2 | riak.us-east-1.zonea.riakts0....riakts.riak x1 | riak.us-east-1.zoneb.ingester4....ingester.staash x2 | riak.us-east-1.zoneb.ingestmq1....ingestmq.karyon x2 | riak.us-east-1.zoneb.riakts4....riakts.riak x3 | riak.us-east-1.zoneb.stream1....stream.karyon x2 | riak.us-east-1.zonec.riakts2....riakts.riak x1

| Services | 30μ | 60μ | 91μ | 121μ | 152μ |
|---|---|---|---|---|---|
| - riak.us-east-1.*.stream-e | 30μ : Put | | | | |
| - riak.us-east-1.zoneB.str | 6μ : Put | | | | |
| - riak.us-east-1.zoneB.in | 6μ : Put | | | | |
| - riak.us-east-1.zoneB.i | | 7μ : Put | | | |
| - riak.us-east-1.zoneB. | | 11μ : Put | | | |
| riak.us-east-1.zoneC | | | | 38μ : Replicate | |
| riak.us-east-1.zoneA. | | | | 9μ : Replicate | |

# Migrating to Microservices

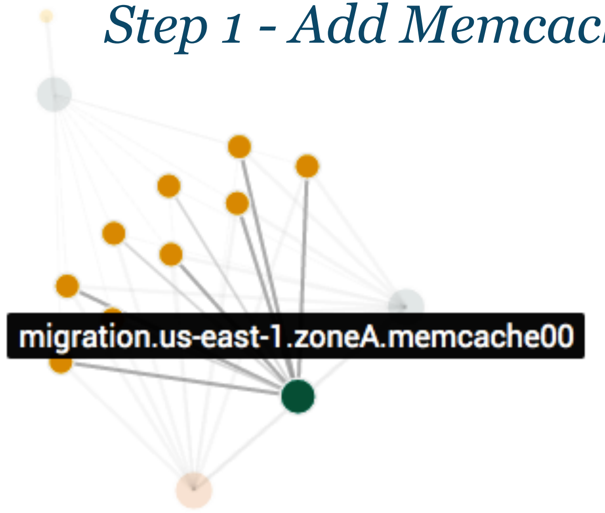
*See for yourself:* http://simianviz.surge.sh/migration



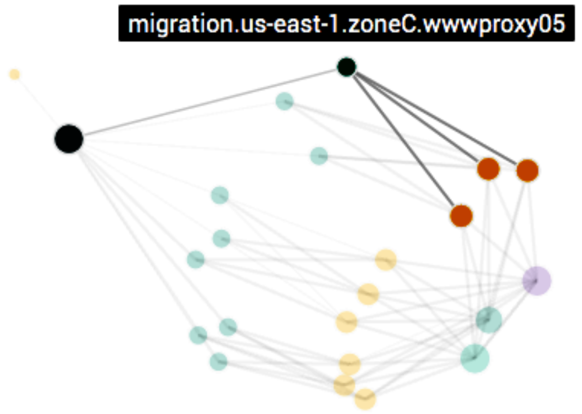
## Migrating to Microservices

*See for yourself:* http://simianviz.surge.sh/migration

*Step 1 - Add Memcache*



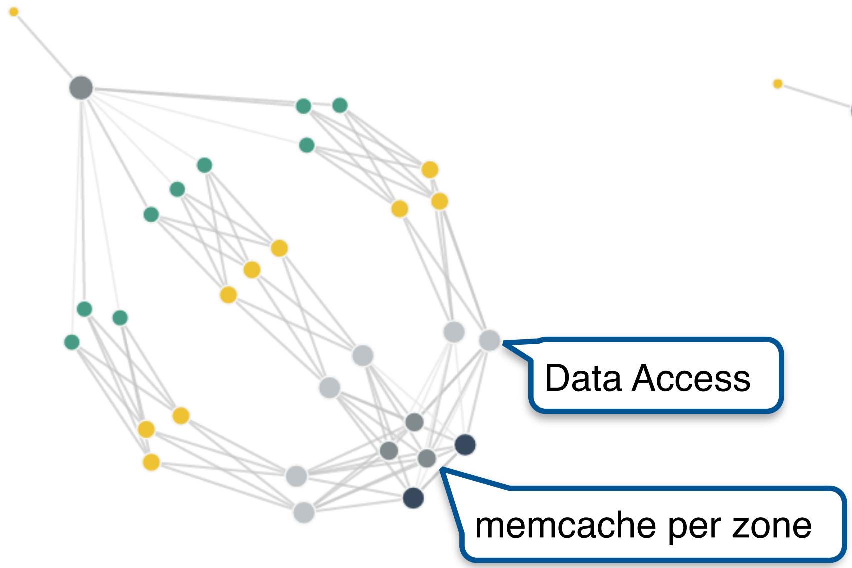
*Step 2 - Add Web Proxy Service*



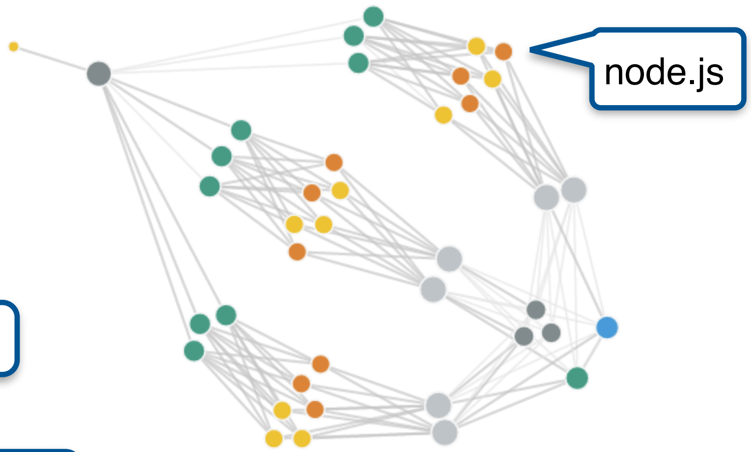
# Migrating to Microservices

*See for yourself:* http://simianviz.surge.sh/migration

*Step 3 - Add Data Access Layer*

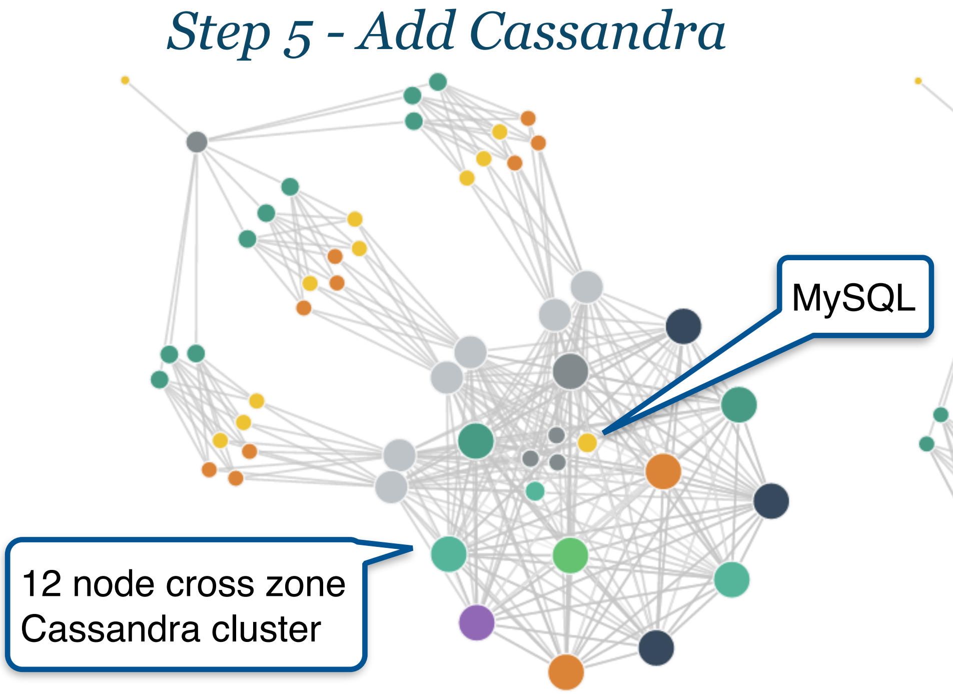


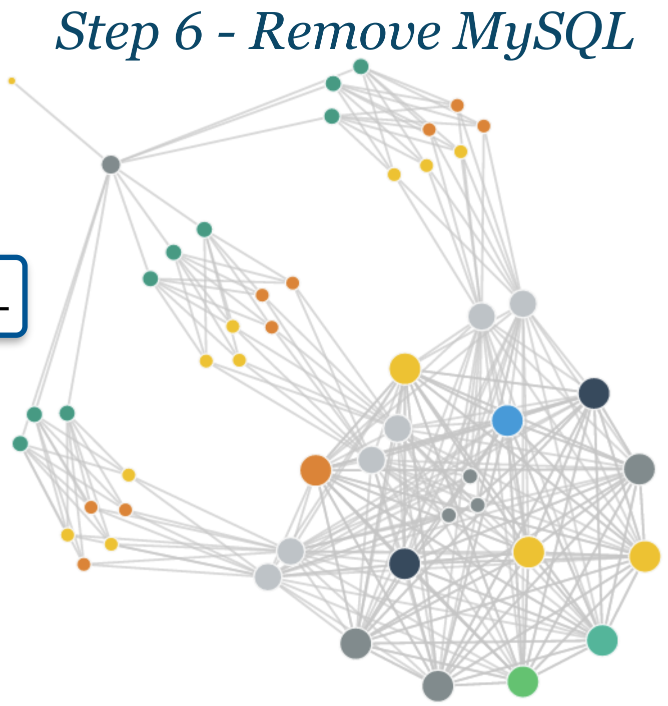
*Step 4 - Add Microservices*



# Migrating to Microservices

*See for yourself:* http://simianviz.surge.sh/migration

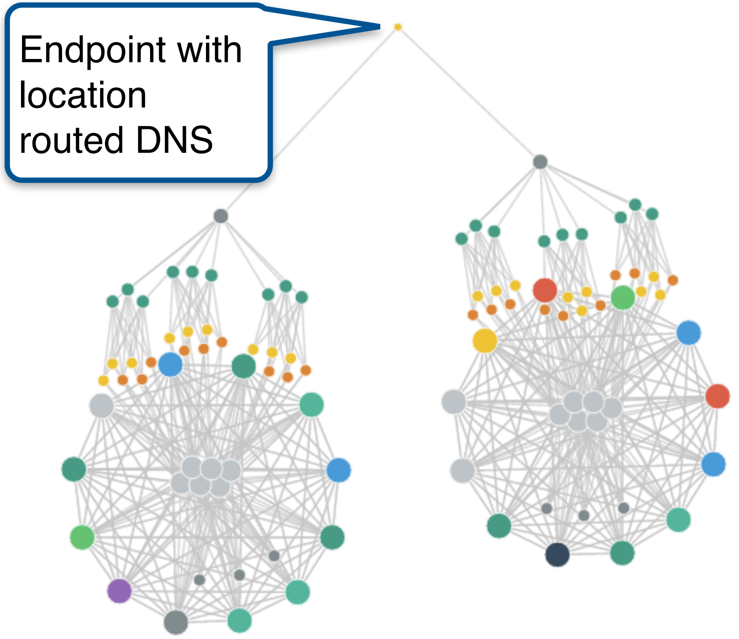

*Step 5 - Add Cassandra*


*Step 6 - Remove MySQL*

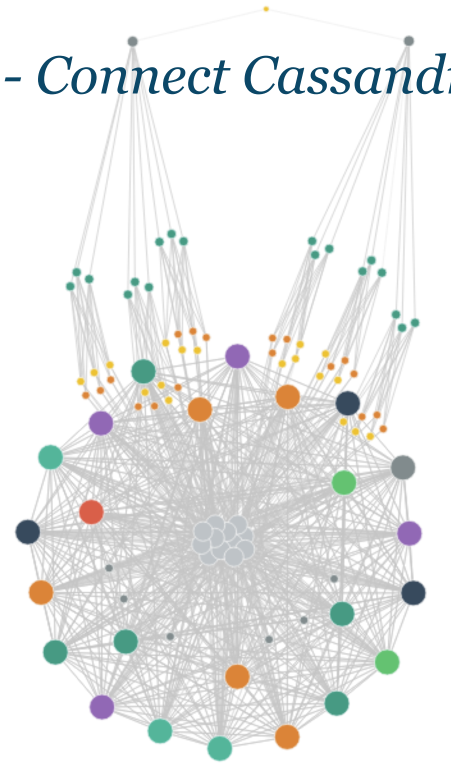
# Migrating to Microservices

*See for yourself:* [http://simianviz.surge.sh/migration](http://simianviz.surge.sh/migration)

*Step 7 - Add Second Region*



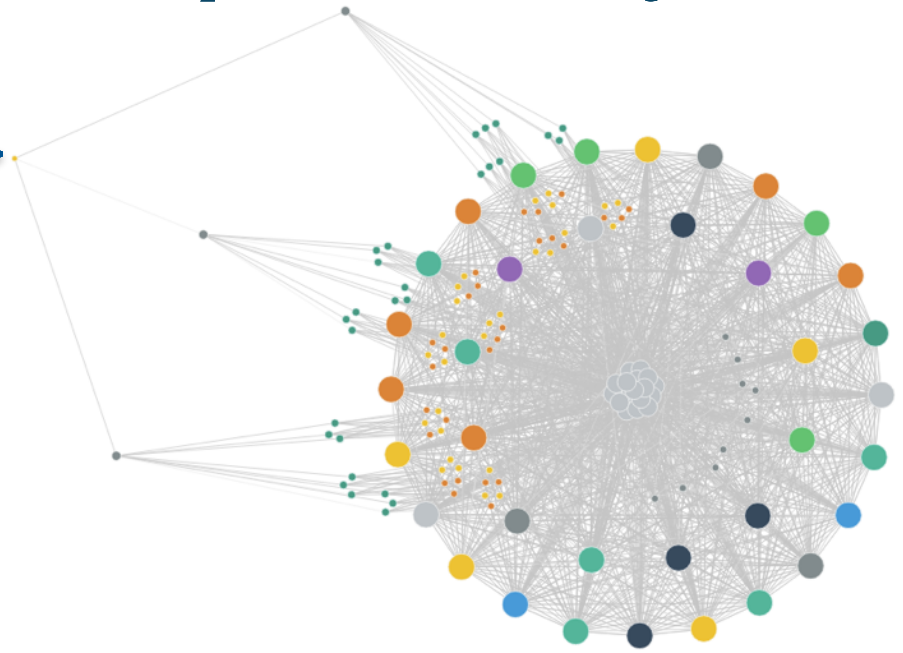
*Step 8 - Connect Cassandra Regions*



## Migrating to Microservices

*See for yourself:* [http://simianviz.surge.sh/migration](http://simianviz.surge.sh/migration)

*Step 9 - Add Third Region*



Endpoint with location routed DNS

## *Simple Architecture Principles*

Symmetry
Invariants
Stable assertions
No special cases

## *What’s Next?*

## *Serverless*

## Serverless Architectures

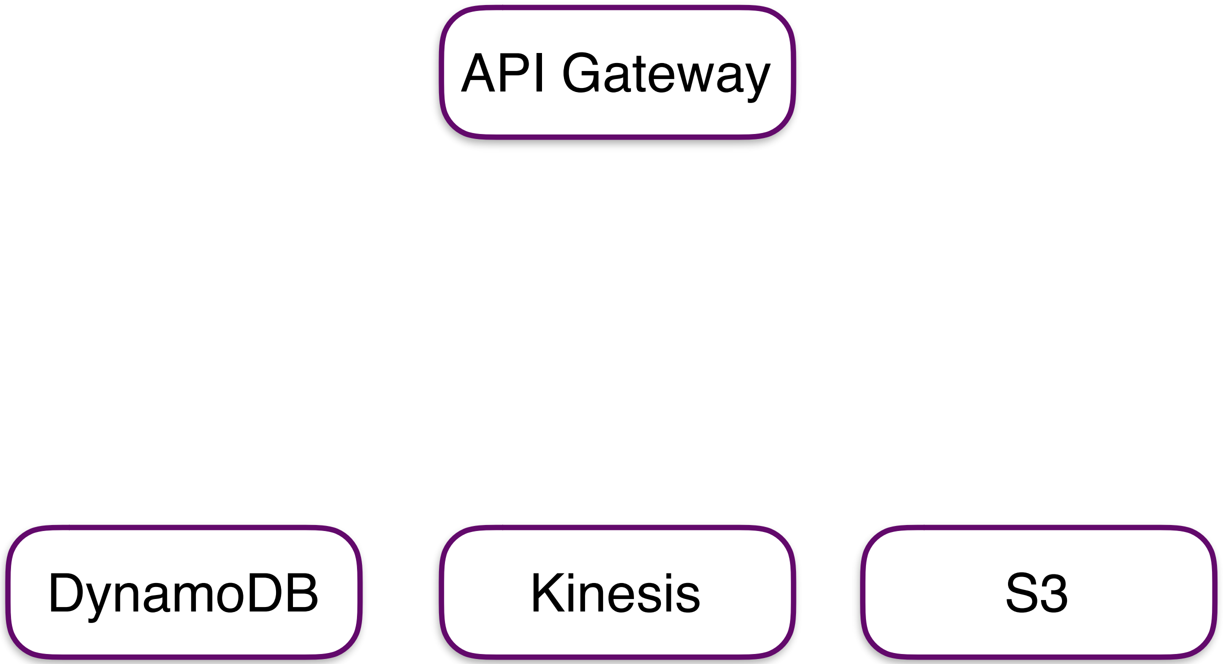
*AWS Lambda getting some early wins*

*Google Cloud Functions, Azure Functions alpha launched*

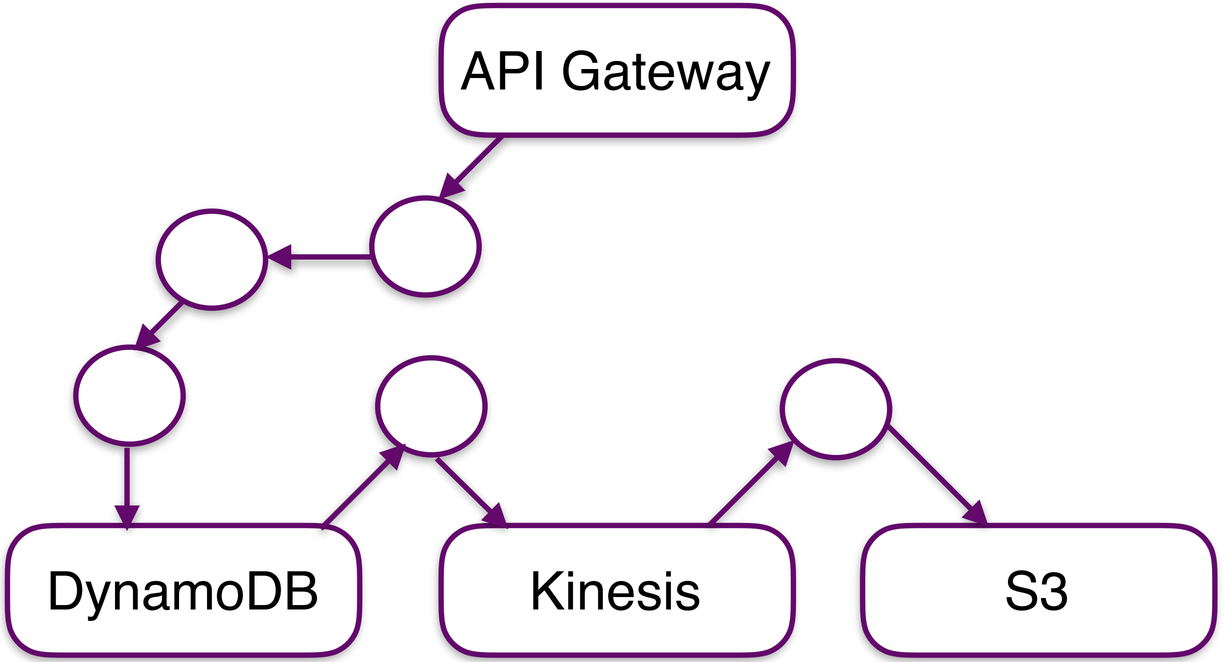
*IBM OpenWhisk - open sourced*

*Startup activity: iron.io , serverless.com, apex.run toolkit*

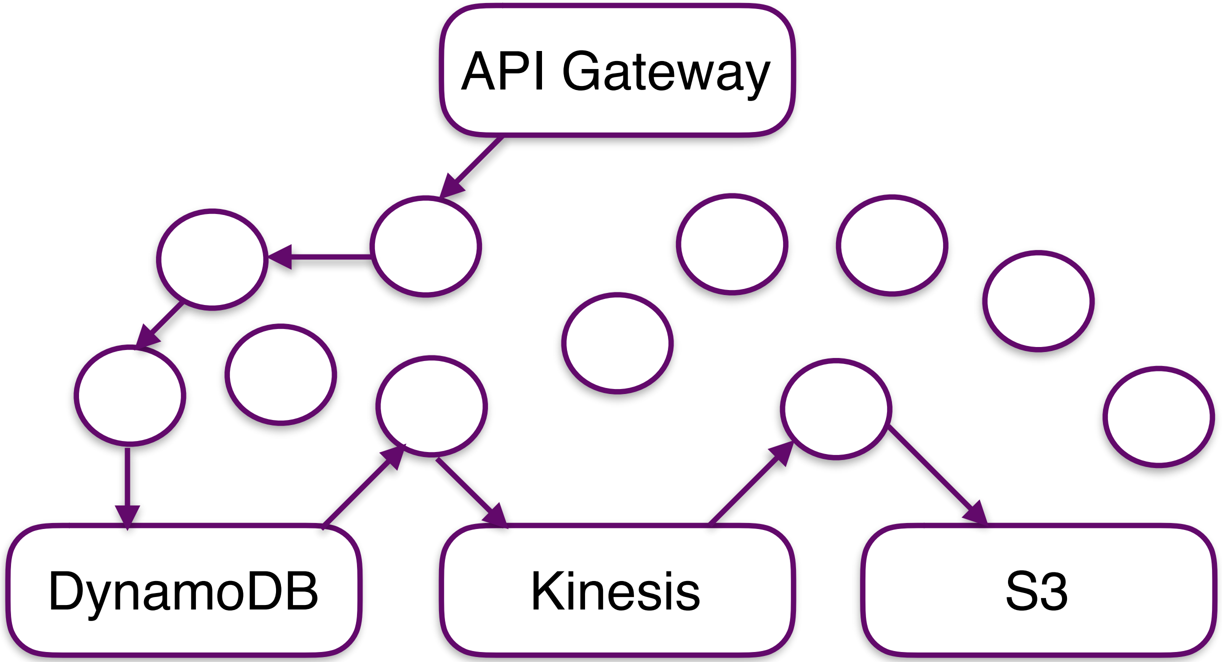
# *Serverless Architecture*



## *Serverless Architecture*

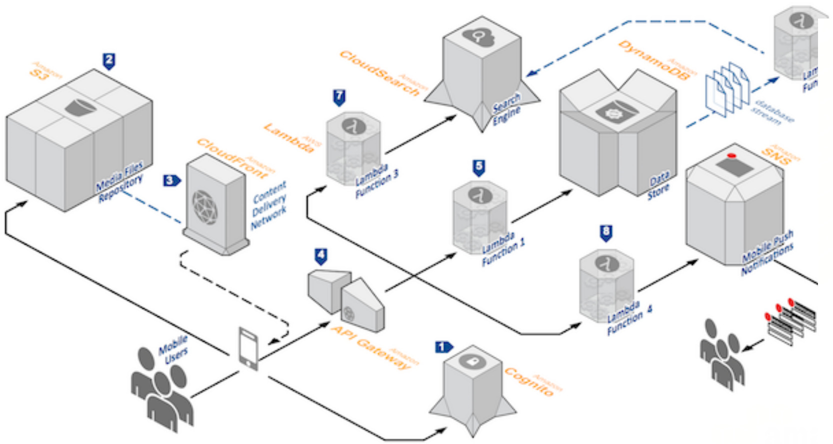


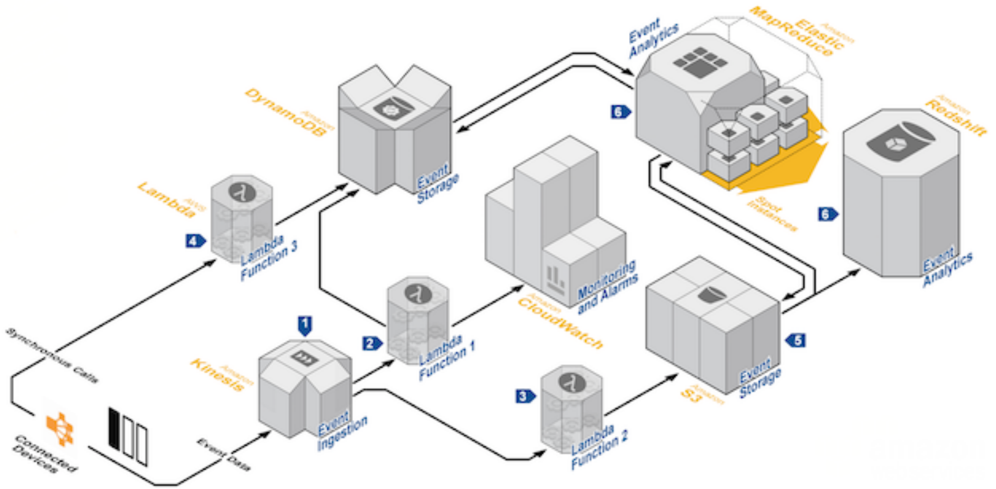
# Serverless Architecture



# AWS Lambda Reference Arch

http://www.allthingsdistributed.com/2016/05/aws-lambda-serverless-reference-architectures.html





## *Serverless Programming Model*

*Event driven functions*
*Role based permissions*
*Whitelisted API based security*
*Good for simple single threaded code*

## Serverless Cost Efficiencies

100% useful work, no agents, overheads
100% utilization, no charge between requests
No need to size capacity for peak traffic
Anecdotal costs ~1% of conventional system
Ideal for low traffic, Corp IT, spiky workloads

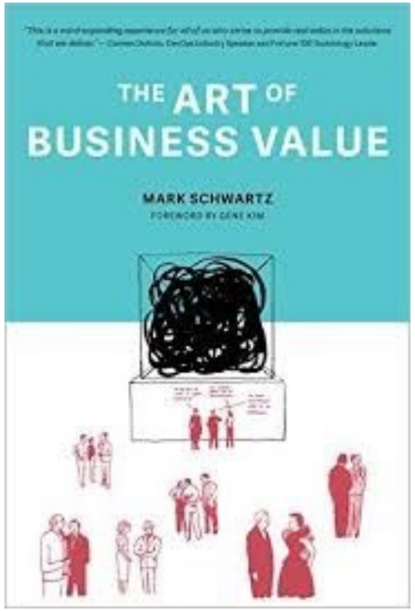
## *Serverless Work in Progress*

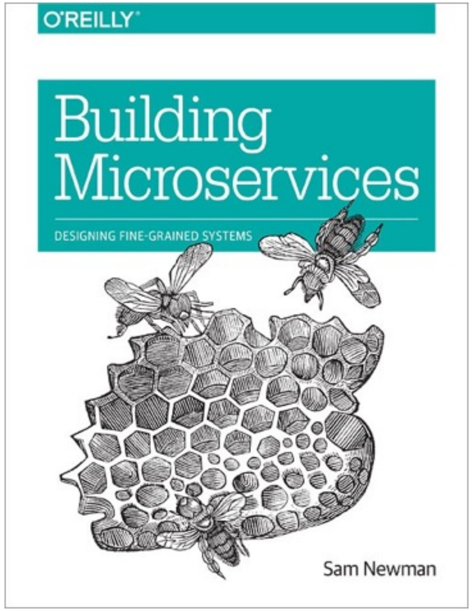
*Tooling for ease of use*
*Multi-region HA/DR patterns*
*Debugging and testing frameworks*
*Monitoring, end to end tracing*

## *DIY Serverless Operating Challenges*

*Startup latency*
*Execution overhead*
*Charging model*
*Capacity planning*

## Learn More…


THE ART OF
BUSINESS VALUE
MARK SCHWARTZ
FOREWORD BY GENE KIM


O'REILLY
Building Microservices
DESIGNING FINE-GRAINED SYSTEMS
Sam Newman

AWS Lambda
A Guide to Serverless Microservices
By. Matthew Fuller

“We see the world as increasingly more complex and chaotic because we use inadequate concepts to explain it. When we understand something, we no longer see it as chaotic or complex.”

*Jamshid Gharajedaghi - 2011*
*Systems Thinking: Managing Chaos and Complexity: A Platform for Designing Business Architecture*

# Q&A

**Adrian Cockcroft @adrianco**
**http://slideshare.com/adriancockcroft**
**Technology Fellow - Battery Ventures**

BV
Battery Ventures

See www.battery.com for a list of portfolio investments

BV Battery Ventures

# Enterprise IT

## Compute

NUTANIX

STRATO SCALE

mx mendix the app platform

## Networking

BlueJeans

fastly

cumulus

## Storage

DELPHIX

diablo technologies

Pd Primary Data

Elastifile Scale-out Performance Manageable

Zerto

## Data

platfora

CASK

bigpanda

SiSense

inter ana

## Operations & Management

AppDynamics

catchpoint

JFrog

Chef

VividCortex

## Security

LIGHTCYBER CLOSING THE BREACH DETECTION GAP

RISKIQ

GuardiCore Securing SDN

VERA

CYVERA CYBER DEFENSE SOLUTIONS

Palo Alto Networks

Visit http://www.battery.com/our-companies/ for a full list of all portfolio companies in which all Battery Funds have invested.