From Microservices to Teraservices

Adrian Cockcroft @adrianco
Technology Fellow - Battery Ventures
September 2015

What does @adrianco do now?

Presentations at Conferences

Maintain Relationship with Cloud Vendors

Technology Due Diligence on Deals

Presentations at Companies

@adrianco

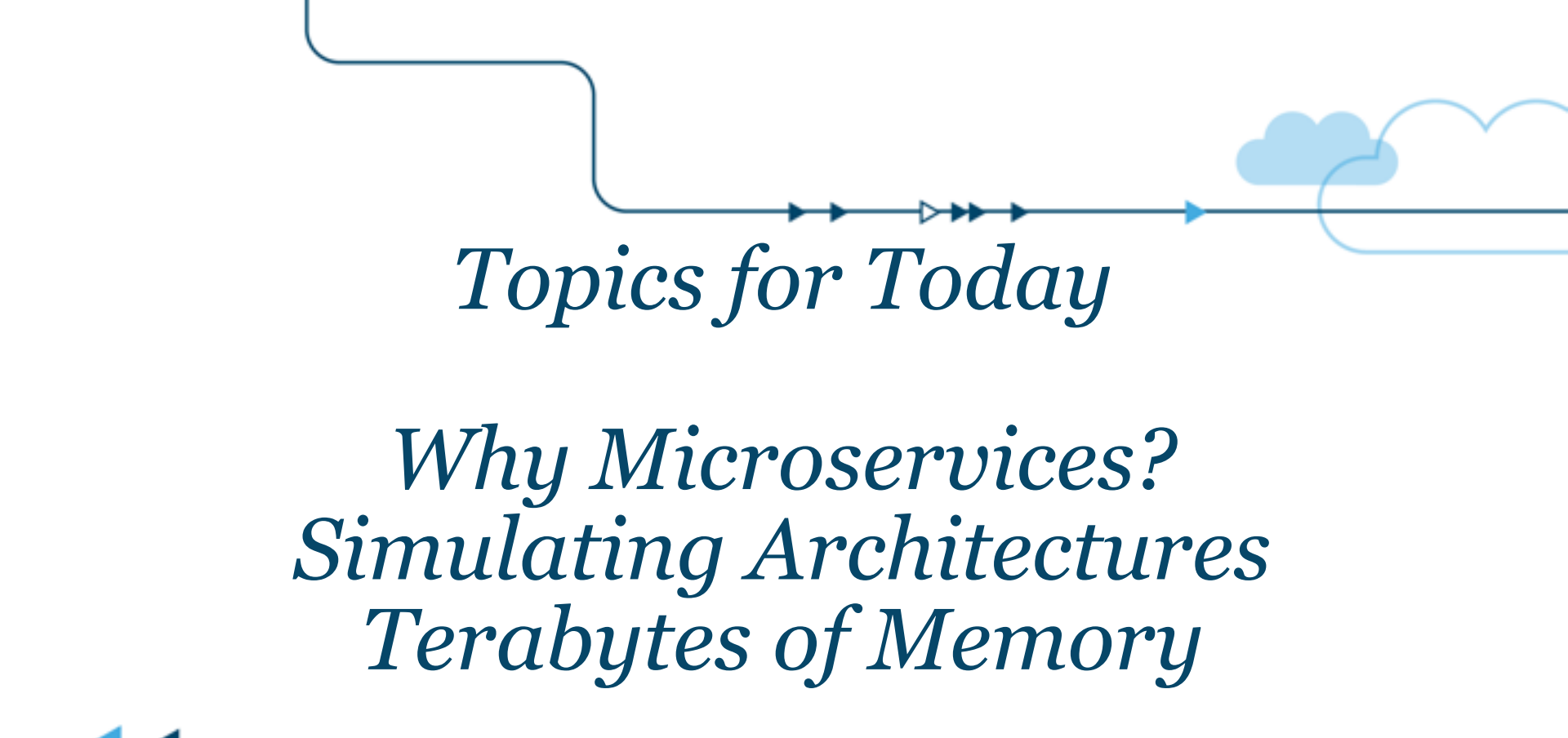
BV Battery Ventures

Technical Advice for Portfolio Companies

Program Committee for Conferences

Tinkering with Technologies

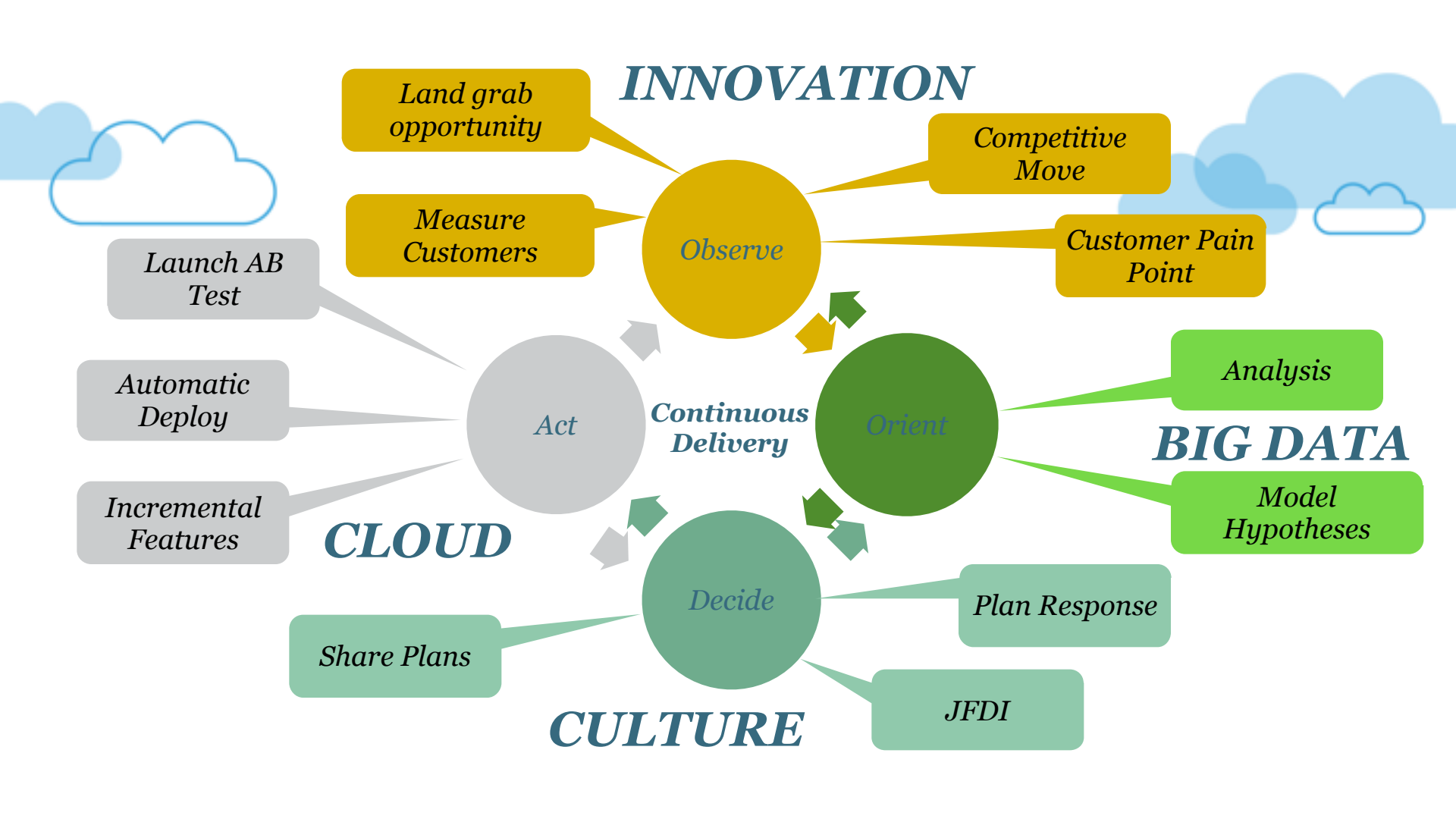
Networking with Interesting People

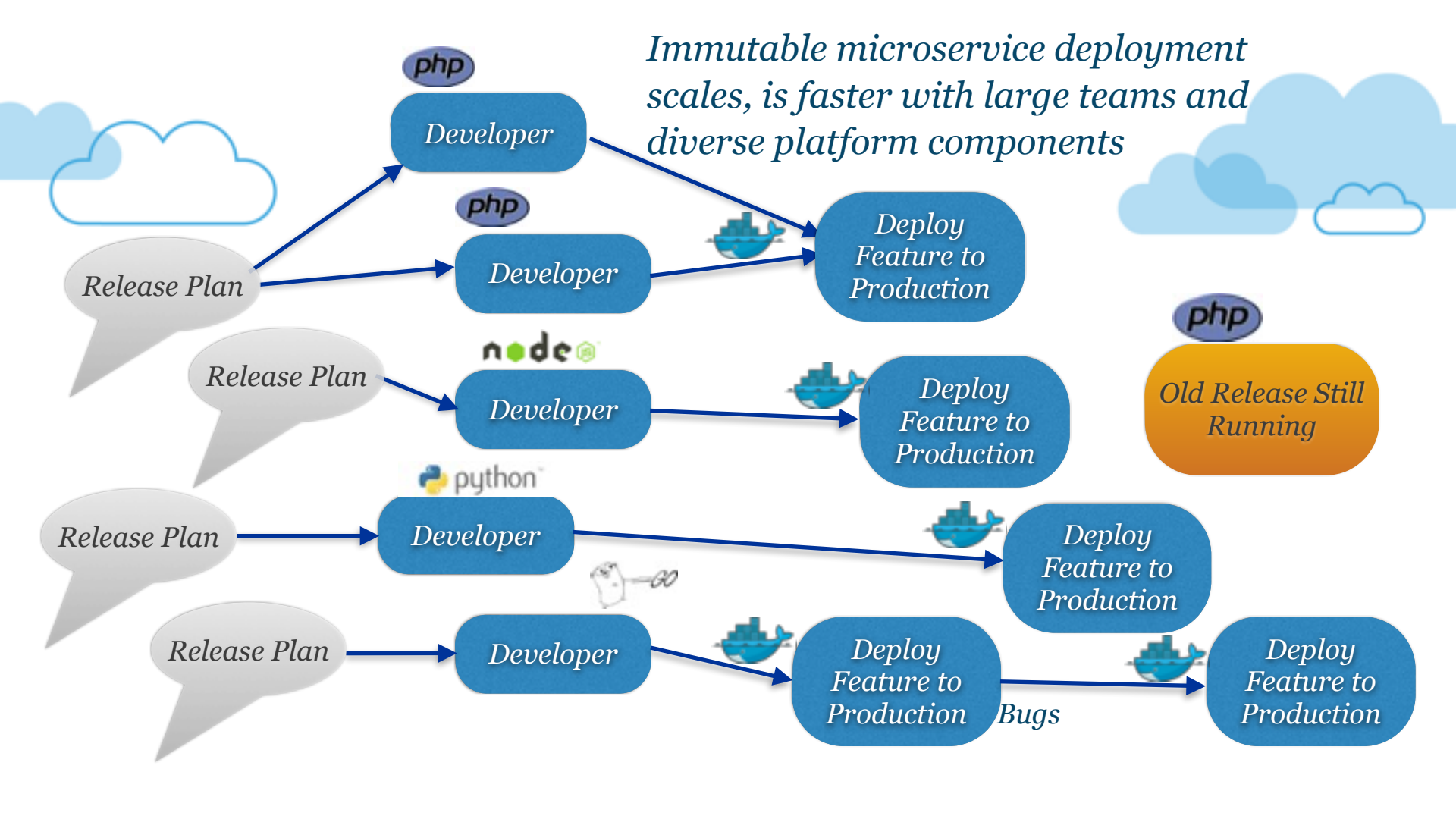


Topics for Today

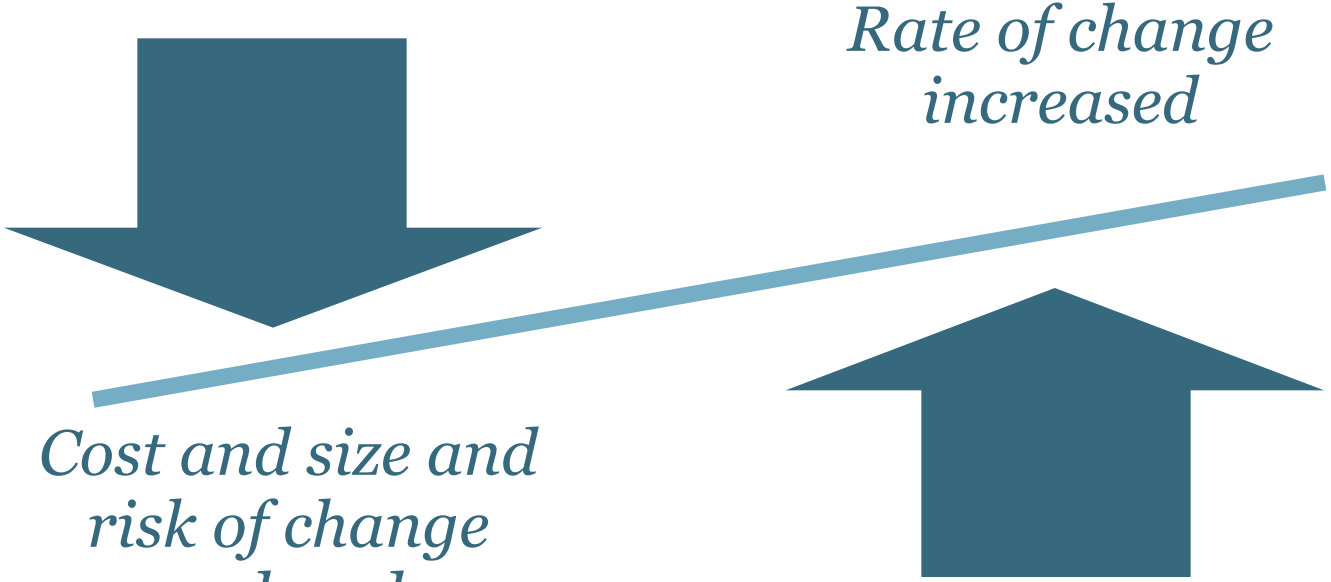
Why Microservices?
Simulating Architectures
Terabytes of Memory

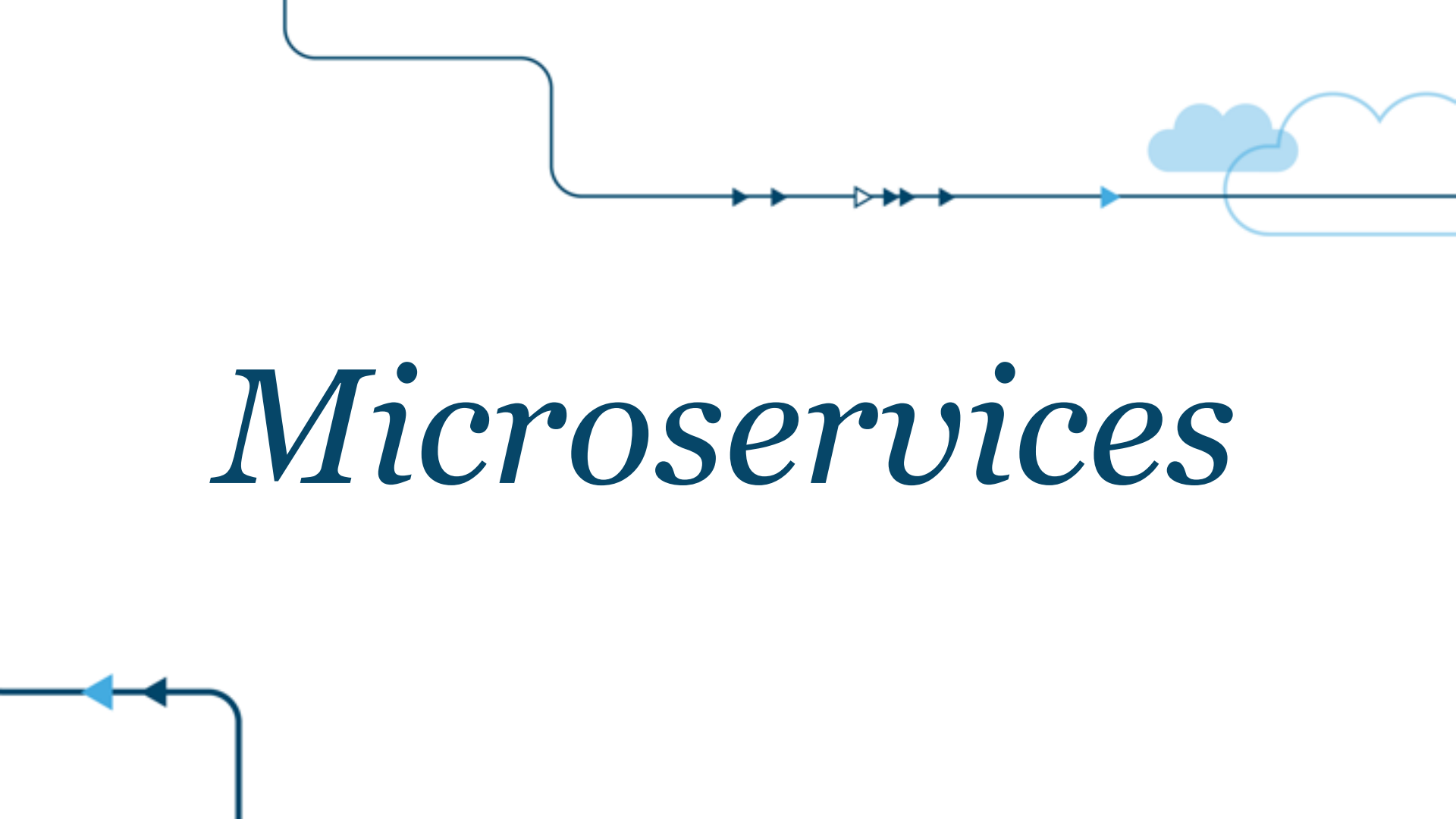

Key Goals of the CIO?
Align IT with the business
Develop products faster
Try not to get breached


INNOVATION
Land grab opportunity
Measure Customers
Observe
Competitive Move
Customer Pain Point
Launch AB Test
Automatic Deploy
Incremental Features
Act
Continuous Delivery
Orient
Analysis
BIG DATA
Model Hypotheses
CLOUD
Decide
Share Plans
Plan Response
JFDI
CULTURE


Immutable microservice deployment scales, is faster with large teams and diverse platform components
php
Developer
Release Plan
php
Developer
Deploy Feature to Production
Release Plan
node
Developer
Deploy Feature to Production
php
Old Release Still Running
Release Plan
python
Developer
Deploy Feature to Production
Release Plan
Developer
Deploy Feature to Production
Bugs
Deploy Feature to Production

What Happened?



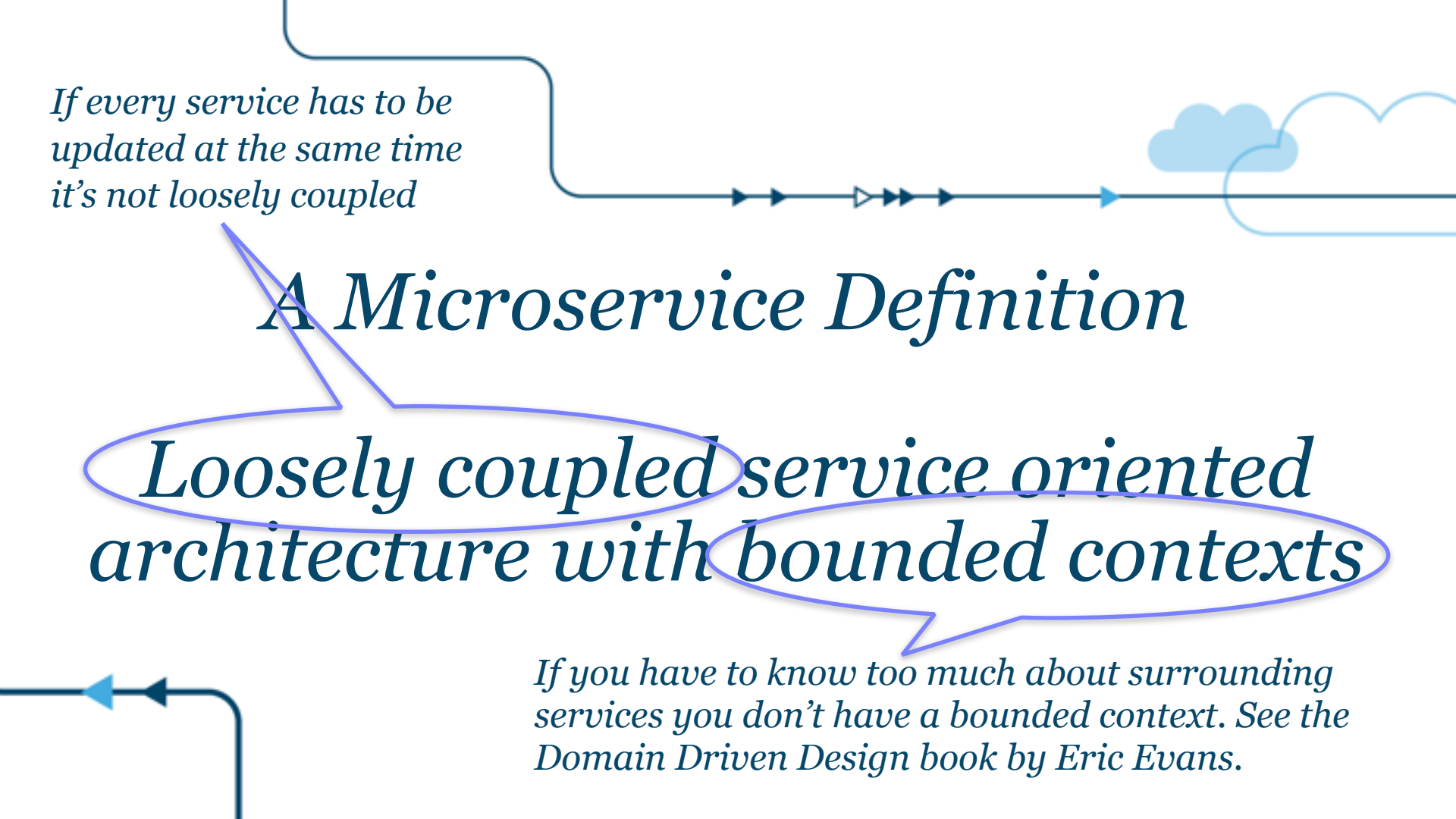

Microservices

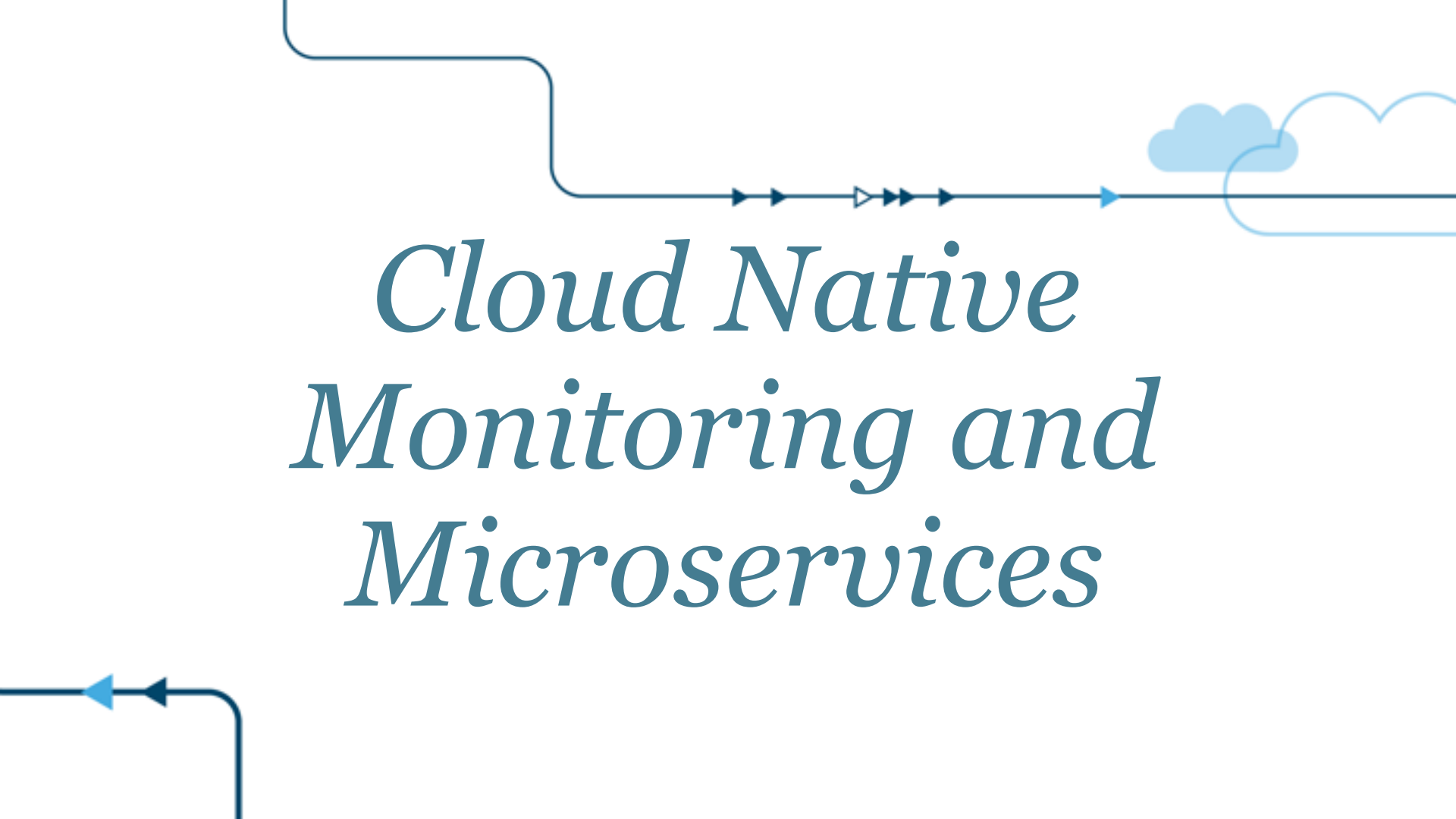
If every service has to be updated at the same time it's not loosely coupled

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.




Cloud Native
Monitoring and
Microservices

Cloud Native Microservices

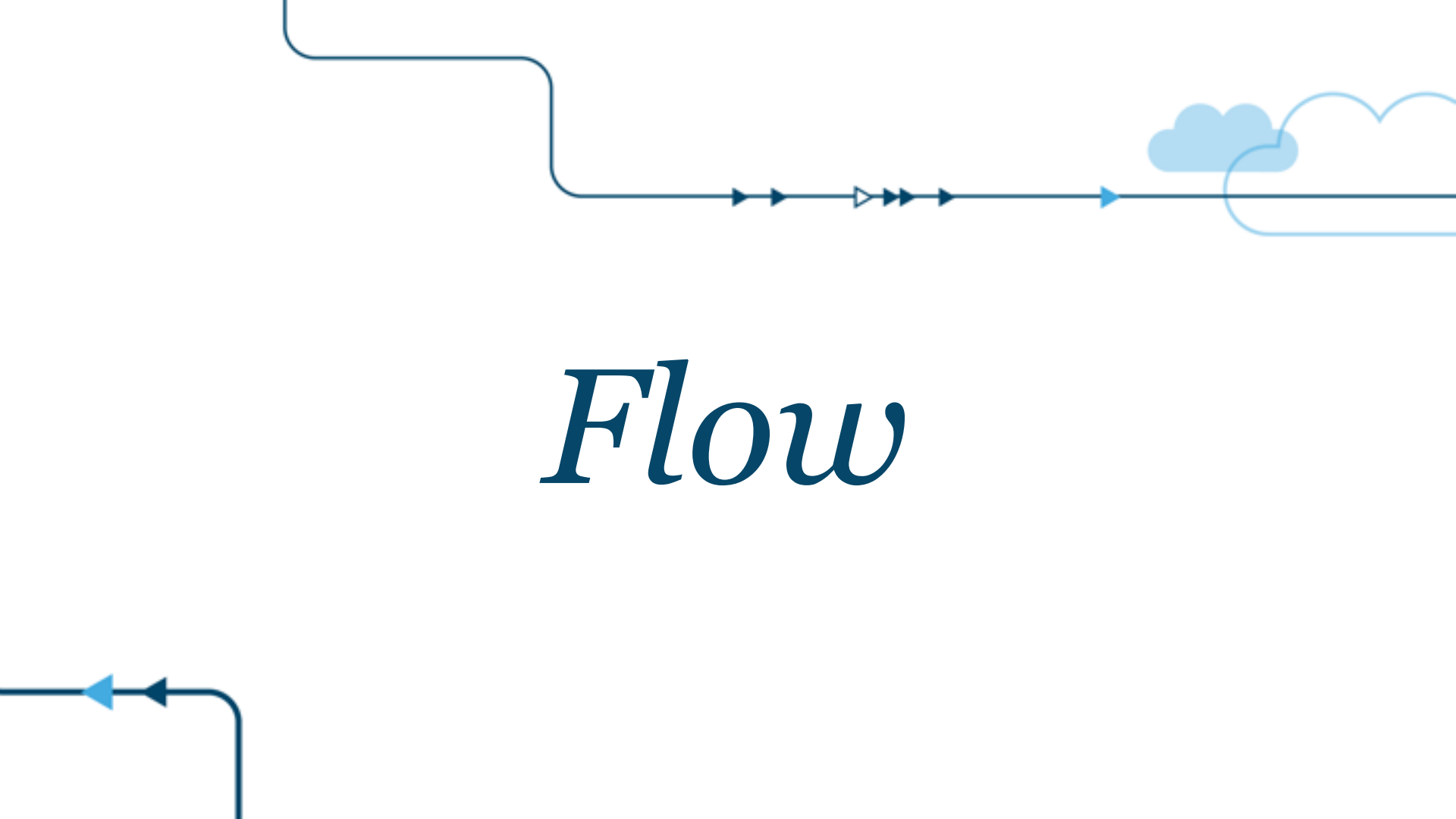
- *High rate of change*
  - *Code pushes can cause floods of new instances and metrics*
  - *Short baseline for alert threshold analysis – everything looks unusual*
- *Ephemeral Configurations*
  - *Short lifetimes make it hard to aggregate historical views*
  - *Hand tweaked monitoring tools take too much work to keep running*
- *Microservices with complex calling patterns*
  - *End-to-end request flow measurements are very important*
  - *Request flow visualizations get overwhelmed*


Challenges for
Microservice
Platforms

Managing Scale

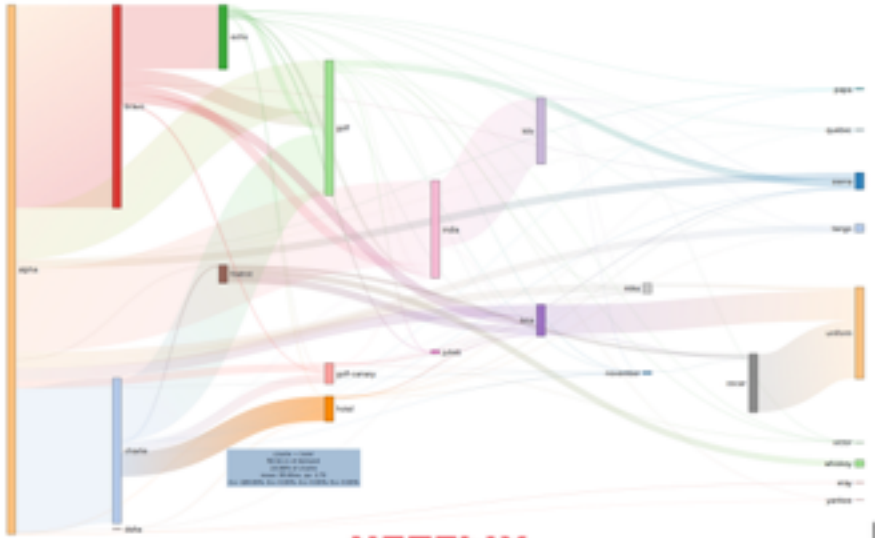
It's much more challenging than just a large number of machines

A Possible Hierarchy	***How Many?***
Continents	*3 to 5*
Regions	*2-4 per Continent*
Zones	*1-5 per Region*
Services	*100's per Zone*
Versions	*Many per Service*
Containers	*1000's per Version*
Instances	*10,000's*


Flow

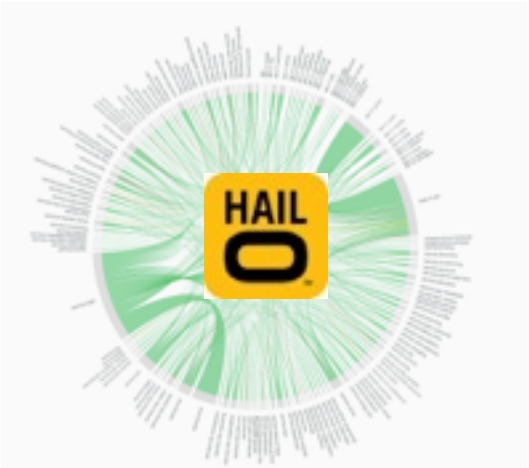
Some tools can show the request flow across a few services

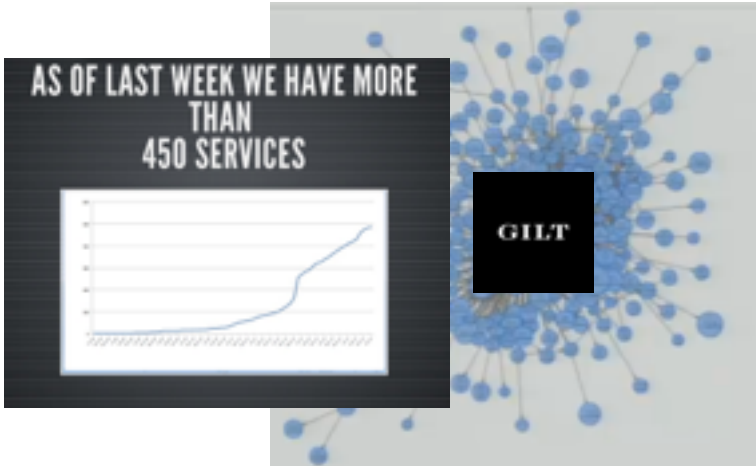
APPDYNAMICS



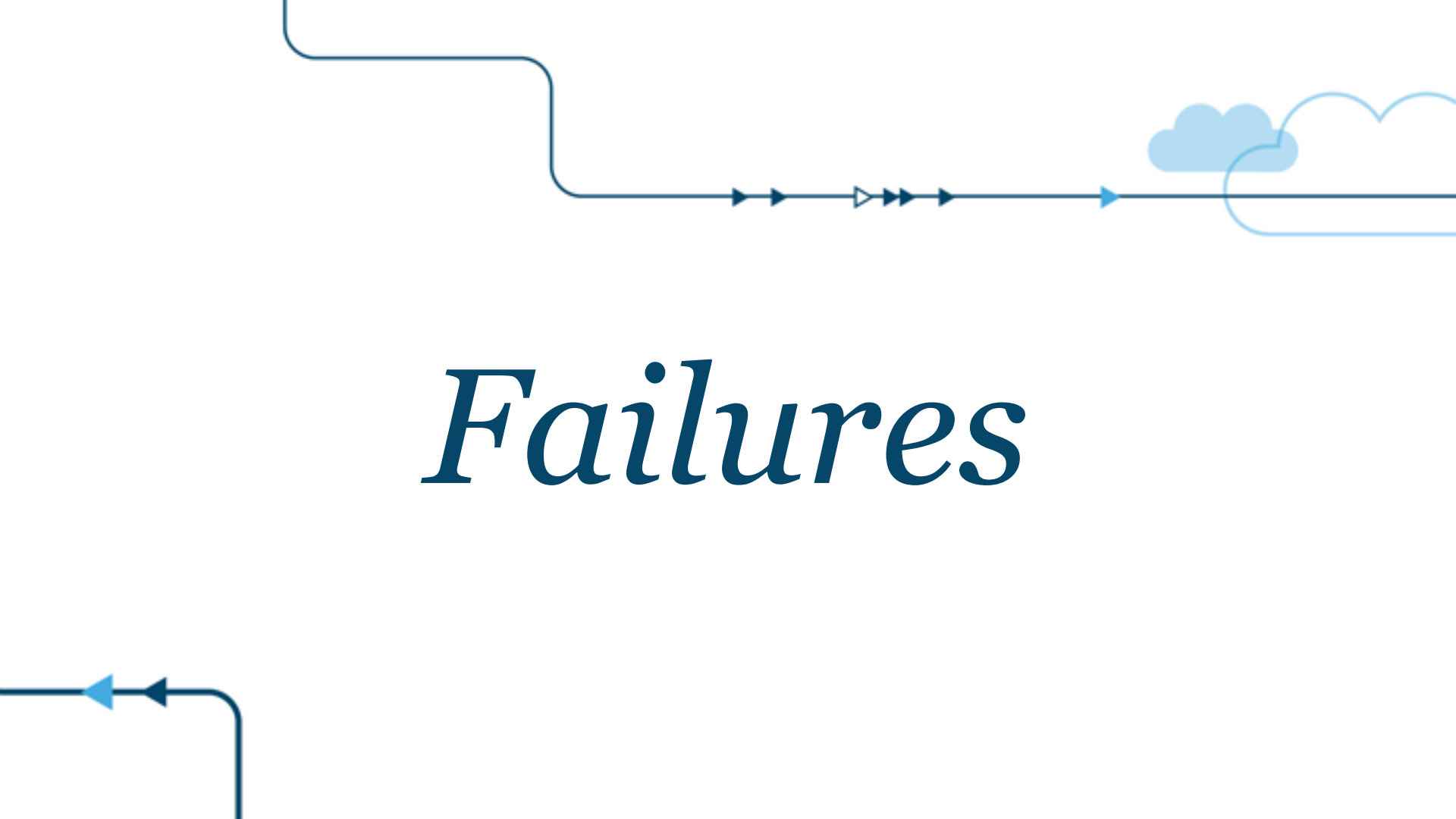
NETFLIX

But 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


Failures

Simple NetflixOSS style microservices architecture on three AWS Availability Zones

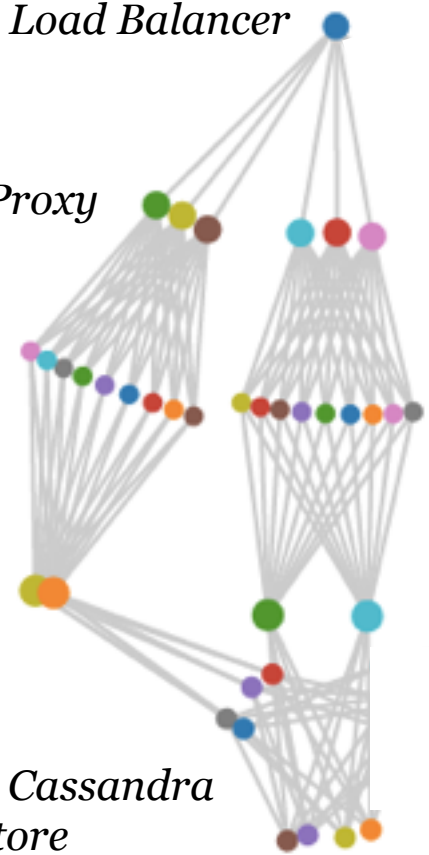
ELB Load Balancer

Zuul API Proxy

Karyon Business Logic

Staash Data Access Layer

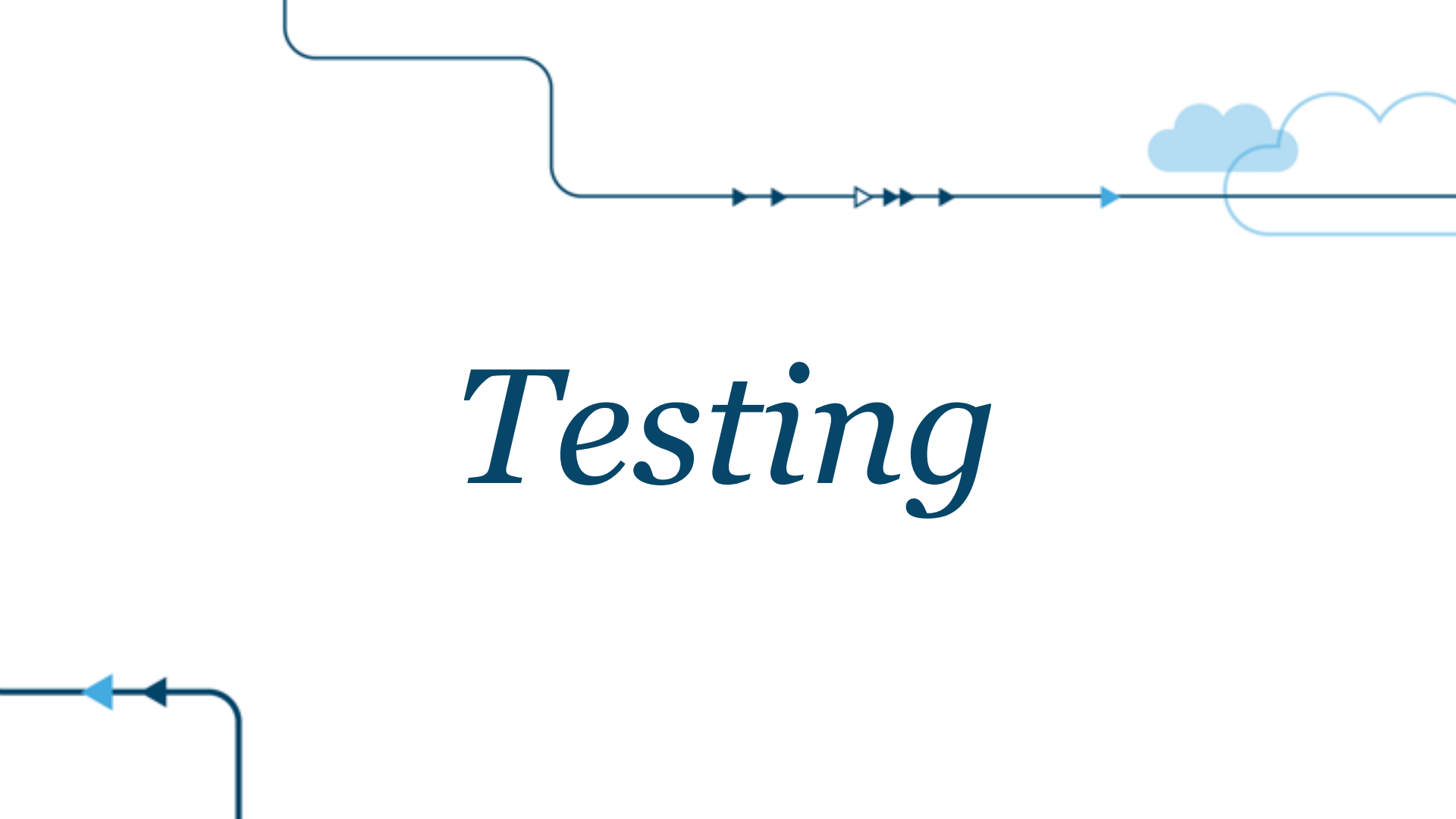
Priam Cassandra Datastore



Zone partition/failure
What should you do?
What should monitors show?

By design, everything works with 2 of 3 zones running.
This is not an outage, inform but don't touch anything!
Halt deployments perhaps?

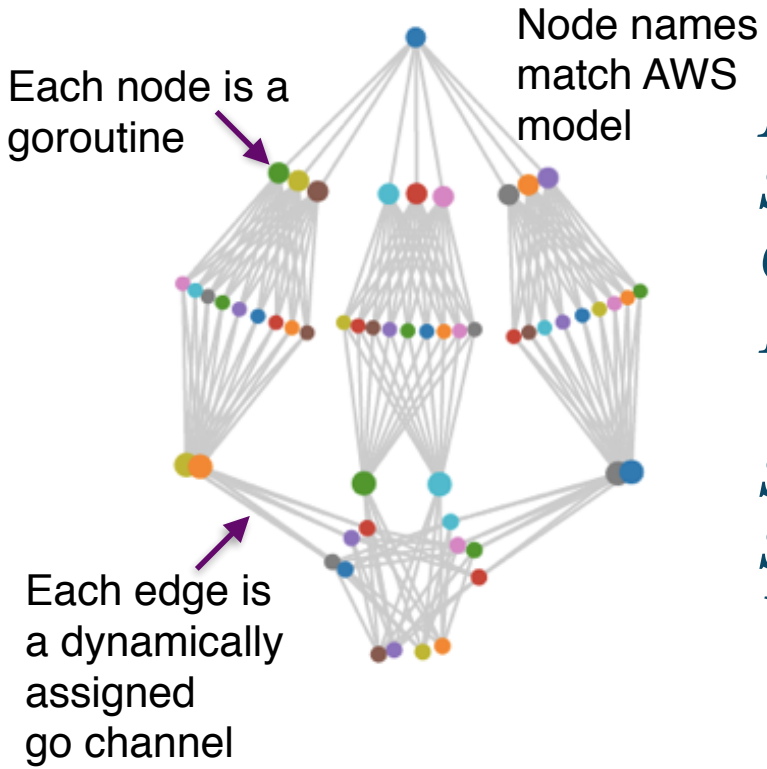
Challenge: understand and communicate common microservice failure patterns.


Testing

Testing monitoring tools at scale gets expensive quickly…

Simulation

Simulated Microservices



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

See github.com/adrianco/spigo
Simulate Protocol Interactions in Go
Visualize with D3

Definition of an architecture

Header can include chaos monkey victim

```
{
    "arch": "cassandra",
    "description":"Simple Cassandra model for the Cassandra Summit 2015 paper",
    "version": "arch-0.0",
    "victim": "",
    "services": [
        { "name": "cassandra",  "package": "priamCassandra", "count": 6, "regions": 1, "dependencies": ["cassandra", "eureka"]},
        { "name": "restdata",   "package": "staash",         "count": 6, "regions": 1, "dependencies": ["cassandra"]},
        { "name": "app",        "package": "karyon",        "count": 12, "regions": 1, "dependencies": ["restdata"]},
        { "name": "proxy",      "package": "zuul",           "count": 6, "regions": 1, "dependencies": ["app"]},
        { "name": "www-elb",    "package": "elb",            "count": 0, "regions": 1, "dependencies": ["proxy"]},
        { "name": "www",        "package": "denominator",    "count": 0, "regions": 0, "dependencies": ["www-elb"]}
    ]
}
```

New tier name

Tier package

Node count

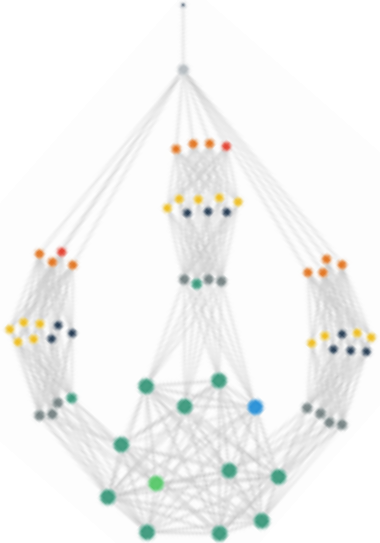
Region count: 1

List of tier dependencies

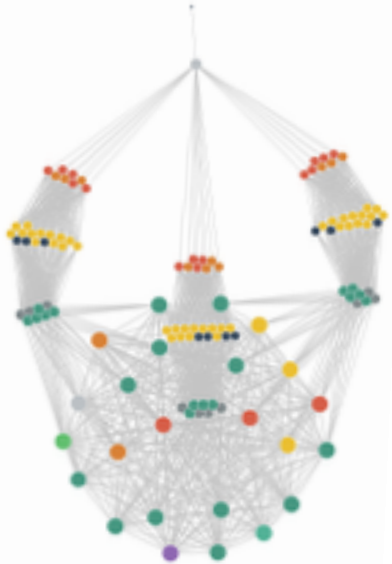
Single Region Architectures

Service only view
3 zones in 1 region

Instance level view
3 zones in 1 region



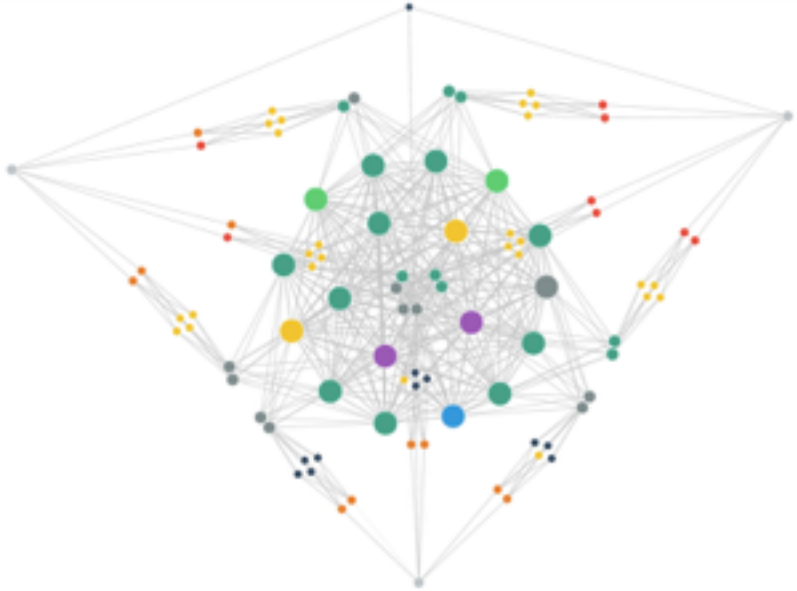
Instance level view
3 zones in 1 region
double scale



Instance level view
3 zones in 1 region
quadruple scale

2 and 3 Region Architectures

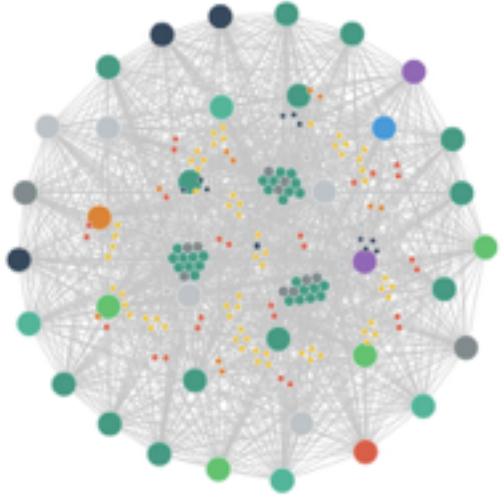
Instance level view
3 zones in 2 regions



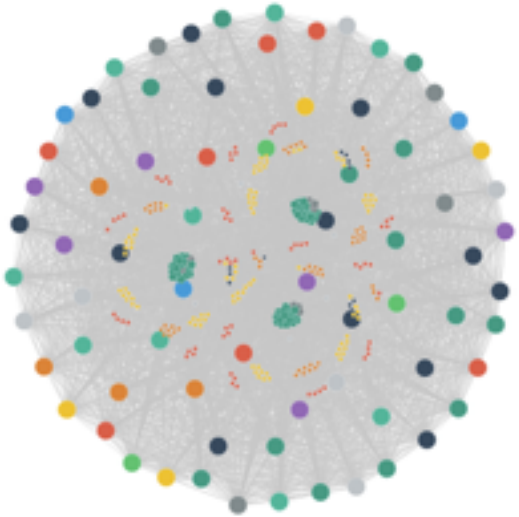
Instance level view
3 zones in 3 regions

4, 5 and 6 Region Cassandra

Instance level view
3 zones in 4 regions



Instance level view
3 zones in 5 regions



Instance level view
3 zones in 6 regions

Adding endpoints & clusters

More realistic architecture for a simple Netflix-like web service

Instance level view
3 zones in 1 region

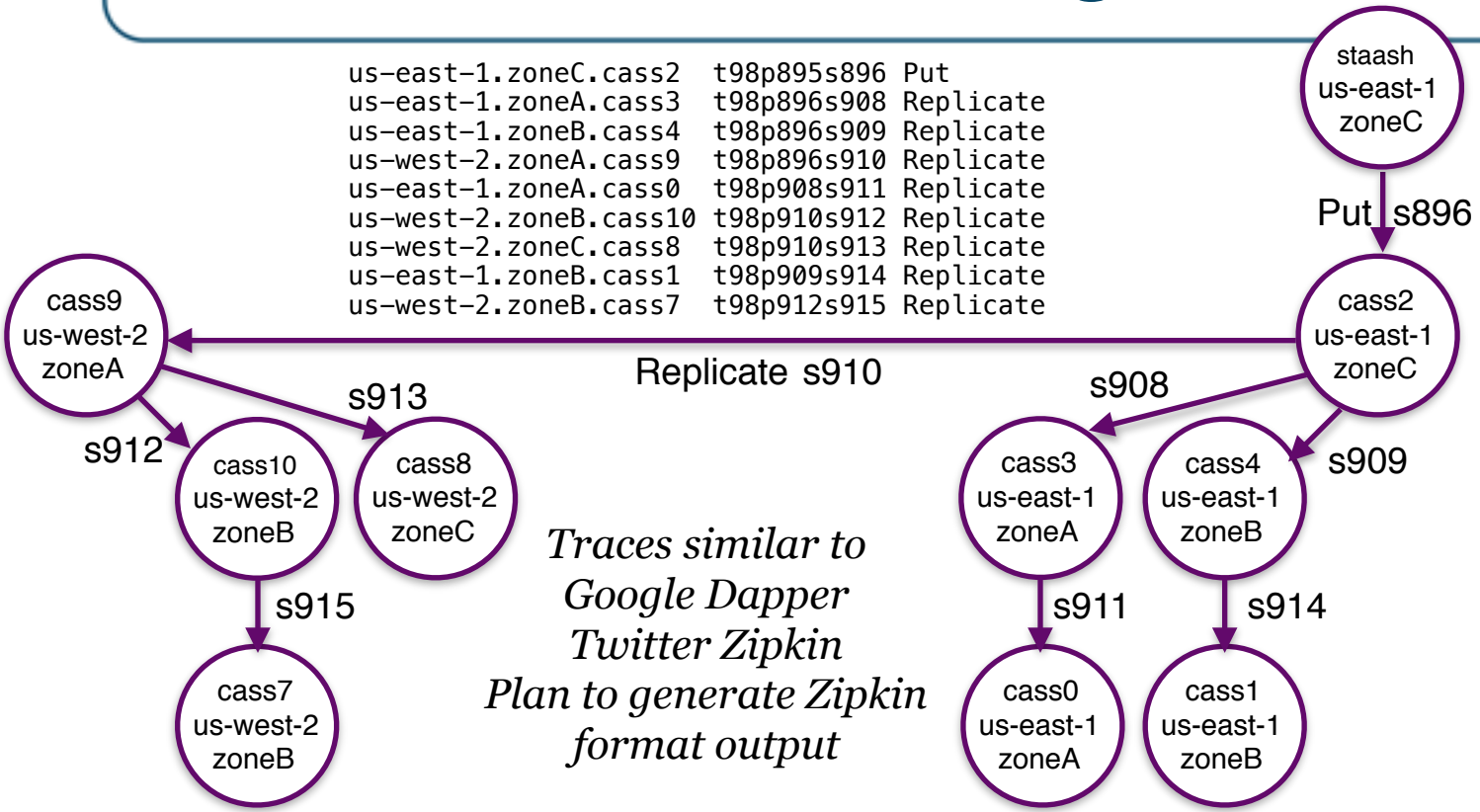
Separate endpoints for API and WWW
Three Cassandra clusters

Spigo Nanoservice Structure

```
func Start(listener chan gotocol.Message) {
        ...
        for {
                select {
                case msg := <-listener:
                        switch msg.Imposition {
                        case gotocol.Hello:      // get named by parent
                                ...
                        case gotocol.NameDrop:   // someone new to talk to
                                ...
                        case gotocol.GetRequest: // upstream request handler
                                ...
                        case gotocol.GetResponse:// downstream response handler
                                ...
                        case gotocol.Goodbye:    // tell parent I'm going away now
                                gotocol.Message{gotocol.Goodbye, nil, time.Now(), name}.GoSend(parent)
                                return
                        }
                case <-eurekaTicker.C:           // poll the service registry
                        ...
                }
        }
}
```

priamCassandra.go package total about 200 lines of Go

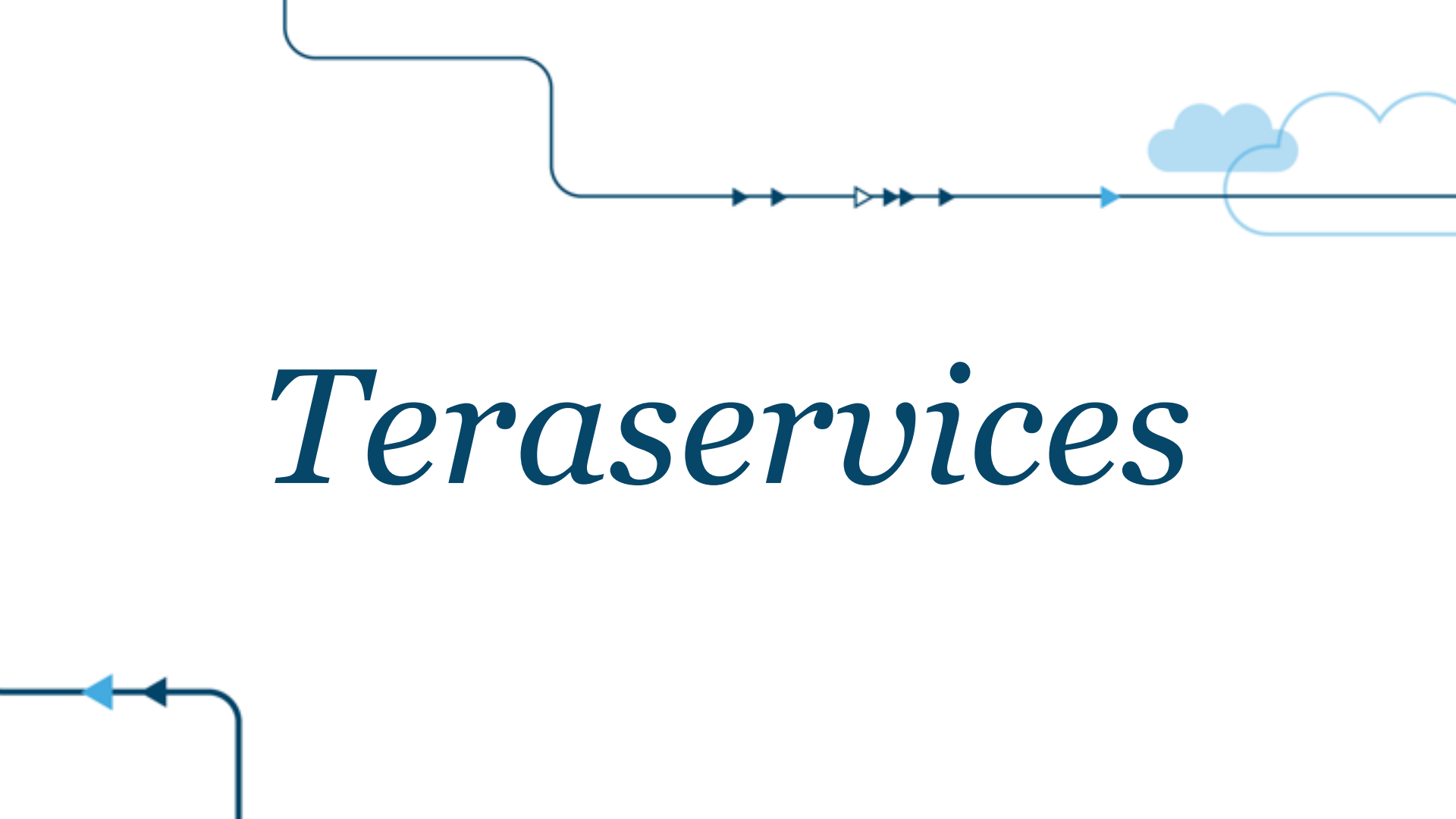
Flow Trace Recording



Why Build Spigo/Simianviz?

Generate test microservice configurations at scale
Stress monitoring tools display capabilities

Eventually (i.e. not implemented yet)
Dynamically vary configuration: autoscale, code push
Chaos monkey for microservice, zone, region failures
D3 websocket dynamic browser interface
github.com/adrianco/spigo


Teraservices

Terabyte Memory Directions

Engulf dataset in memory for analytics

Balanced config for memory intensive workloads

Replace high end systems at commodity cost point

Explore non-volatile memory implications

A Terabyte Memory Option

Diablo - a Battery Ventures portfolio company

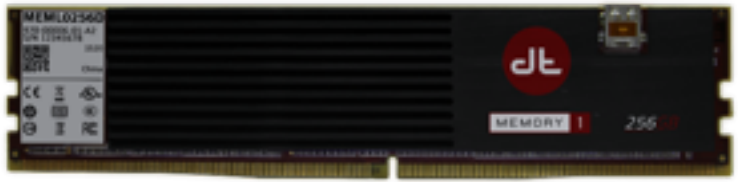
DDR4 DIMM containing flash 64/128/256GB

Migrates pages to/from companion DRAM DIMM

Shipping now as volatile memory, future non-volatile

Memory1: 1st All-Flash System

MEMORY 1™



- **NO CHANGES** to CPU or Server
- **NO CHANGES** to Operating System
- **NO CHANGES** to Applications

✓ **UP TO 256GB DDR4 MEMORY PER MODULE**

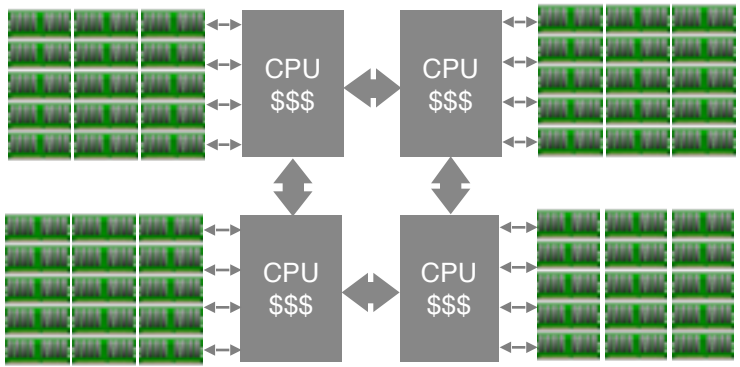
✓ **UP TO 4TB MEMORY IN 2 SOCKET SYSTEM**

In-Memory Database: *Minimize Infrastructure Cost*

Problem: IMDB memory requirements force purchase of additional unneeded CPUs

Solution: Increased memory-per-socket eliminates need for additional costly CPUs

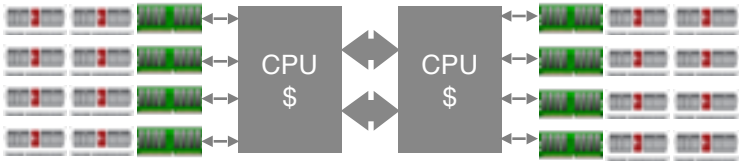
Quad-Processor Configuration
(With DRAM Only)



DRAM capacity limitations necessitate additional DIMM slots

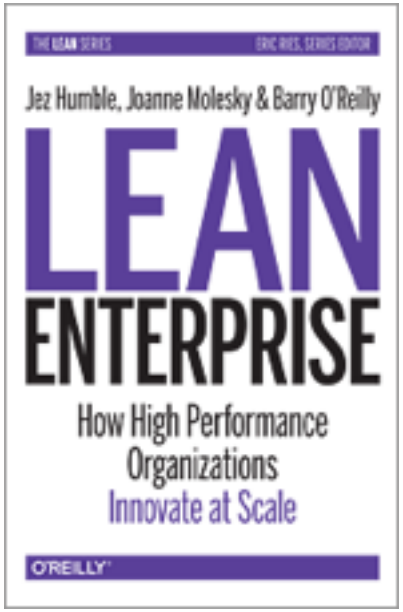
MEMORY 1™

Dual-Processor Configuration
(System Memory Expanded By Memory1)



With **more memory per socket**, IMDBs can be supported by **less expensive servers**

Takeaway



Teraservices moving to mainstream

MEMORY 1™

Q&A

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

July 2015

BV
Battery Ventures

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

BV
Battery Ventures

Enterprise IT

Compute

NUTANIX
STRATO SCALE
mx mendix

Networking

BlueJeans
fastly
cumulus

Storage

DELPHIX
diablo technologies
Pd Primary Data
Elastifile
Zerto

Data

platfora
CASK
SiSense
interana

Operations & Management

AppDynamics
catchpoint
Chef
VisidCortex

Security

LIGHTCYBER
RISKIQ
GuardiCore Securing SDN
VERA
CYVERA
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.