State of the Art in Microservices

Adrian Cockcroft @adrianco
Technology Fellow - Battery Ventures
Microxchg Berlin - February 2015
Speeding up Development
Microservice Architectures
What’s Next
Why am I here?

By Simon Wardley http://enterpriseitadoption.com/
Why am I here?

 Adoption

 By Simon Wardley http://enterpriseitadoption.com/

2009

Ignore Ignore Ignore "No" "No" I said "No" dammit "Oh No" "Oh %*&!"

Rest of World

Enterprise IT

Time
Why am I here?

By Simon Wardley http://enterpriseitadoption.com/

2009

Ignore: Ignore: Ignore

“No” "No" "No"

I said "No" dammit

"Oh No" "Oh %* &!"

Adoption

Time

Rest of World

Enterprise IT
Why am I here?

@adrianco’s job at the intersection of cloud and Enterprise IT, looking for disruption and opportunities.

By Simon Wardley http://enterpriseitadoption.com/
Why am I here?

@adrianco’s job at the intersection of cloud and Enterprise IT, looking for disruption and opportunities.

Example: Docker wasn’t on anyone’s roadmap for 2014. It’s on everyone’s roadmap for 2015.

By Simon Wardley http://enterpriseitadoption.com/
2014 was the year that Enterprises finally embraced cloud and DevOps.
2014 was the year that Enterprises finally embraced cloud and DevOps.
2014 was the year that Enterprises finally embraced cloud and DevOps.
2014 was the year that Enterprises finally embraced cloud and DevOps.
What does @adrianco do?

- Presentations at Conferences
- Presentations at Companies
- Program Committee for Conferences
- Maintain Relationship with Cloud Vendors
- Technology Due Diligence on Deals
- Technical Advice for Portfolio Companies
- Networking with Interesting People
- Tinkering with Technologies
- Maintain Relationship with Cloud Vendors
Product Development Processes
Assumption: Process prevents problems
Organizations build up slow complex “Scar tissue” processes
Observe
Orient
Decide
Act
Continuous Delivery
Observe
Orient
Decide
Act

Land grab opportunity
Measure Customers
Competitive Move
Customer Pain Point

Continuous Delivery

Customers
Continuous Delivery
INNOVATION

- **Observe**
  - Land grab opportunity
  - Measure Customers
- **Act**
- **Orient**
  - Competitive Move
  - Customer Pain Point
- **Decide**
  - Continuous Delivery
Observe

Orient

Decide

Act

INNOVATION

Land grab opportunity

Competitive Move

Customer Pain Point

Measure Customers

Analysis

Model Hypotheses

Continuous Delivery

Customers
INNOVATION

Observe

Land grab opportunity
Measure Customers

Competitive Move
Customer Pain Point

Orient
Continuous Delivery

Act

Decide

BIG DATA
Analysis
Model Hypotheses

Measure Customers
INNOVATION

Observe

Land grab opportunity
Measure Customers

Orient

Competitive Move
Customer Pain Point

Decide

Continuous Delivery

Act

Share Plans

Plan Response
JFDI

BIG DATA

Analysis
Model Hypotheses
INNOVATION

Observe
- Land grab opportunity
- Measure Customers
- Competitive Move
- Customer Pain Point

Orient
- Continuous Delivery
- Analysis
- Model Hypotheses

Decide
- Plan Response
- JFDI

Act
- Share Plans

CULTURE

BIG DATA

INNOVATION

Observe
- Land grab opportunity
- Measure Customers
- Competitive Move
- Customer Pain Point

Orient
- Continuous Delivery
- Analysis
- Model Hypotheses

Decide
- Plan Response
- JFDI

Act
- Share Plans

CULTURE

BIG DATA
INNOVATION
- Land grab opportunity
- Competitive Move
- Customer Pain Point
- Measure Customers

CLOUD
- Launch AB Test
- Automatic Deploy
- Incremental Features
- Continuous Delivery

CULTURE
- Decide
- Share Plans
- Plan Response
- JFDI

BIG DATA
- Analysis
- Model Hypotheses

INNOVATION
- Observe
- Orient
- Act

INNOVATION
- INNOVATION
- CLOUD
- CULTURE

INNOVATION
- Innovation
- Cloud
- Culture

INNOVATION
- Land grab opportunity
- Competitive Move
- Customer Pain Point
- Measure Customers

CLOUD
- Launch AB Test
- Automatic Deploy
- Incremental Features
- Continuous Delivery

CULTURE
- Decide
- Share Plans
- Plan Response
- JFDI

BIG DATA
- Analysis
- Model Hypotheses
Observe

Orient

Decide

Act

Land grab opportunity

Competitive Move

Customer Pain Point

INNOVATION

Measure Customers

Analysis

Model Hypotheses

BIG DATA

CLOUD

Launch AB Test

Automatic Deploy

Incremental Features

Continuous Delivery

CULTURE

Share Plans

Plan Response

JFDI

INNOVATION

CULTURE

Cloud
Breaking Down the SILOs
Breaking Down the SILOs

- Prod Mgr
- UX
- Dev
- QA
- DBA
- Sys Adm
- Net Adm
- SAN Adm
Breaking Down the SILOs

Product Team Using Monolithic Delivery

Product Team Using Microservices

Product Team Using Microservices

Product Team Using Microservices
Breaking Down the SILOs

Product Team Using Monolithic Delivery

Product Team Using Microservices

Product Team Using Microservices

Product Team Using Microservices

Platform Team
Breaking Down the SILOs

Product Team Using Monolithic Delivery

Product Team Using Microservices

Product Team Using Microservices

Product Team Using Microservices

Platform Team
Breaking Down the SILOs

DevOps is a Re-Org!
Monolithic service updates

Works well with a small number of developers and a single language like php, java or ruby
Monolithic service updates

Works well with a small number of developers and a single language like php, java or ruby
Monolithic service updates

Works well with a small number of developers and a single language like php, java or ruby
Immutable microservice deployment scales, is faster with large teams and diverse platform components.
Immutable microservice deployment scales, is faster with large teams and diverse platform components.
Immutable microservice deployment scales, is faster with large teams and diverse platform components.
Immutable microservice deployment scales, is faster with large teams and diverse platform components.
Standardized portable container deployment saves time and effort

Configure

Deploy Standardized Services

https://hub.docker.com
Standardized portable container deployment saves time and effort

Configure

Deploy Standardized Services

Deploy Feature to Production

Deploy Feature to Production

Deploy Feature to Production

Bugs

https://hub.docker.com
Run What You Wrote

Developer

Developer

Developer

Developer
Run What You Wrote
Run What You Wrote

Monitoring Tools

Developer

Microservice

Developer

Microservice

Developer

Microservice

Developer

Microservice
Run What You Wrote
99.95% customer success rate

Run What You Wrote

Available Metrics

Site Reliability

Monitoring Tools

Developer

Microservice

Developer

Microservice

Developer

Microservice

Developer

Microservice

Developer

Microservice
Run What You Wrote

99.95% customer success rate

- Availability Metrics
- Site Reliability
- Monitoring Tools

Manager

Developer

Microservice

Microservice

Microservice

Microservice

Manager

Developer

Microservice

Microservice

Microservice

Microservice

Microservice

Microservice

Microservice
Run What You Wrote

99.95% customer success rate

Availabilty Metrics

Site Reliability

Monitoring Tools

VP Engineering

Manager

Developer

Microservice

Microservice

Microservice

Microservice

Microservice

Microservice

Microservice

Microservice

Microservice
What Happened?

Rate of change increased

Cost and size and risk of change reduced
Developing at the Speed of Docker

Developers
- Compile/Build
- Seconds

Extend container
- Package dependencies
- Seconds

PaaS deploy Container
- Docker startup
- Seconds
Developing at the Speed of Docker

- Speed is addictive, hard to go back to taking much longer to get things done

Developers
- Compile/Build
- Seconds

Extend container
- Package dependencies
- Seconds

PaaS deploy Container
- Docker startup
- Seconds
Disruptor: Continuous Delivery with Containerized Microservices
"This is the IT swamp draining manual for anyone who is neck deep in alligators."

1984

2014
A Microservice Definition

Loosely coupled service oriented architecture with bounded contexts
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.

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.
Coupling Concerns

- Conway’s Law - organizational coupling
- Centralized Database Schemas
- Enterprise Service Bus - centralized message queues
- Inflexible Protocol Versioning

http://en.wikipedia.org/wiki/Conway’s_law
Non-Destructive Production Updates

- “Immutable Code” Service Pattern
  - Existing services are unchanged, old code remains in service
  - New code deploys as a new service group
  - No impact to production until traffic routing changes
- A|B Tests, Feature Flags and Version Routing control traffic
  - First users in the test cell are the developer and test engineers
  - A cohort of users is added looking for measurable improvement
  - Finally make default for everyone, keeping old code for a while
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

- **Docker Containers**
  - 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

- **Docker Containers**
  - Deploy in seconds
  - Live for minutes/hours

- **AWS Lambda**
  - 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

- **Docker Containers**
  - Deploy in seconds
  - Live for minutes/hours

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

*Speed enables and encourages new microservice architectures*
With AWS Lambda compute resources are charged by the 100ms, not the hour

First 1M node.js executions/month are free
State of the Art in Web Scale Microservice Architectures

AWS Re:Invent : Asgard to Zuul [https://www.youtube.com/watch?v=p7ysHhs5hl0](https://www.youtube.com/watch?v=p7ysHhs5hl0)
Resiliency at Massive Scale [https://www.youtube.com/watch?v=ZfYJHtVL1_w](https://www.youtube.com/watch?v=ZfYJHtVL1_w)
Microservice Architecture [https://www.youtube.com/watch?v=CriDUYtfrjs](https://www.youtube.com/watch?v=CriDUYtfrjs)

[http://www.infoq.com/presentations/scale-gilt](http://www.infoq.com/presentations/scale-gilt)

[http://www.slideshare.net/mcculloughsean/itier-breaking-up-the-monolith-philly-ete](http://www.slideshare.net/mcculloughsean/itier-breaking-up-the-monolith-philly-ete)

[http://www.infoq.com/presentations/Twitter-Timeline-Scalability](http://www.infoq.com/presentations/Twitter-Timeline-Scalability)

Microservice Concerns

- Tooling
- Configuration
- Discovery
- Routing
- Observability

Datastores

Operational: Orchestration and Deployment Infrastructure

Development: Languages and Container
Microservices

- Asgard
  - Aminator
  - Tooling
- Edda
  - Archaius
  - Configuration
- Eureka
  - Prana
  - Discovery
- Denominator
  - Zuul, Netty
  - Ribbon 2.0
  - Routing
- Hystrix
  - Pytheus
  - SALP
  - Observability

Ephemeral datastores using Dynomite, Memcached, Astyanax, Staash, Priam, Cassandra

Manual Orchestration with Asgard and deployment on AWS or Eucalyptus

Java, Groovy, Scala, Clojure, Python, Node.js with AMI and Docker Containers
Microservices

- Asgard
  Tooling
- Edda
  Archaius
  Configuration
- Eureka
  Prana
  Discovery
- Denominator
  Zuul, Netty
  Ribbon 2.0
  Routing
- Hystrix
  Pytheus
  SALP
  Observability

Ephemeral datastores using Dynomite, Memcached, Astyanax, Staash, Priam, Cassandra

Manual Orchestration with Asgard and deployment on AWS or Eucalyptus

Java, Groovy, Scala, Clojure, Python, Node.js with AMI and Docker Containers

Focus on global distribution, high scale and availability
These companies are using and contributing to Netflix OSS Components.

[Logos of various companies]
Twitter Microservices

- Tooling
- Decider
- Configuration
- Finagle
- Zookeeper
- Discovery
- Finagle
- Netty
- Routing
- Zipkin
- Observability

Custom Cassandra-like datastore: Manhattan

Orchestration using Aurora deployment in datacenters using Mesos

Scala with JVM Container
Twitter Microservices

- Tooling
- Decider
- Configuration
- Finagle
- Zookeeper
- Discovery
- Finagle
- Netty
- Routing
- Zipkin
- Observability

Custom Cassandra-like datastore: Manhattan

Orchestration using Aurora deployment in datacenters using Mesos

Scala with JVM Container

Focus on efficient datacenter deployment at scale
Gilt Microservices

Ion Cannon
SBT
Rake
Tooling

Decider
Configuration

Finagle
Zookeeper
Discovery

Akka
Finagle
Netty
Routing

Zipkin
Observability

Datastores per Microservice using MongoDB, Postgres, Voldemort

Deployment on AWS

Scala and Ruby with Docker Containers
Gilt Microservices

- Ion Cannon
- SBT
- Rake
- Tooling

- Decider
- Configuration

- Finagle
- Zookeeper
- Discovery

- Akka
- Finagle
- Netty
- Routing

- Zipkin
- Observability

Datastores per Microservice using MongoDB, Postgres, Voldemort

Deployment on AWS

Scala and Ruby with Docker Containers

*Focus on fast development with Scala and Docker*
Hailo Microservices

Hubot Janky Jenkins
Tooling

Configuration

go-platform
Discovery

go-platform RabbitMQ
Routing

Request trace
Observability

Datastore based on Cassandra

Deployment on AWS

Go using Docker
Hailo Microservices

- Hubot
- Janky
- Jenkins

- Configuration
- Discovery
- Routing
- Request trace

- go-platform
- RabbitMQ

Focus on fast development at scale using Go

- Datastore based on Cassandra
- Deployment on AWS
- Go using Docker
Node.js Microservices

Several different approaches

Mostly small simple microservices

Focus on easy interface with presentation code in javascript

AWS Lambda - preview only

Groupon

@Walmart Labs

http://senecajs.org/

http://aws.amazon.com/lambda/
Adrian’s Tinkering Projects

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

See [github.com/adriancor/spigo](https://github.com/adriancor/spigo)
Simulate Protocol Interactions in Go
Visualize with D3
What is Spigo?

- Creates and animates microservices
- Single Go program on this laptop
- Generates 100,000+ instances
- About 250,000 messages/sec
- Uses Go channels rather than http
- Supports social network architecture
- Supports NetflixOSS architecture
- Simple code patterns to extend
Why Build Spigo?

*Generate test microservice configurations at scale*

*Stress monitoring tools display capabilities*

Eventually (i.e. not implemented yet)

*Dynamically vary configuration: autoscale, code push*

*Simulate microservice, zone, region failures*

*D3 websocket dynamic browser interface*

*Timescale: Monitorama Conference June 2015*
func Start(listener chan gotocol.Message) {
    ...
    for {
        select {
            case msg := <-listener:
                switch msg.Imposition {
                    case gotocol.Hello:
                        ...
                    case gotocol.NameDrop:
                        ...
                    case gotocol.Chat:
                        ...
                    case gotocol.GetResponse:
                        ...
                    case gotocol.Goodbye:
                        gotocol.Message{gotocol.Goodbye, nil, time.Now(), name}.GoSend(netflixoss)
                        return
                }
            case <-chatTicker.C:
                ...
        }
    }
}
What’s Next?
Web Scale Characteristics

- Brand new Microservices are deployed infrequently
- New versions deployed automatically/frequently
- No real need for general purpose orchestration
- Architectures use hundreds of microservices
- Each deployment is heavily customized
Orchestration for Applications

- Standard portable microservice based applications
- New versions deployed automatically/frequently
- Orchestration automated and standardized
- Architectures likely based on tens of microservices
- Opportunity: Docker Hub as the enterprise app store
Next Generation Applications

**Tooling**
- Docker?
- PaaS?
- ?

**Configuration**
- ?
- Eureka?
- Consul?

**Discovery**
- ?
- Ribbon?
- Finagle?

**Routing**
- ?
- Zipkin?
- Metrics?
- Hystrix?

**Observability**
- ?

**Datastores:** Distributed Ephemeral, Orchestrated or DBaaS

**Operational:** Many orchestration choices across public and private clouds

**Development:** Components assembled from Docker Hub as a composable “app store”
Next Generation Applications

- **Tooling**
  - Docker?
  - PaaS?
- **Configuration**
- **Discovery**
  - Eureka?
  - Consul?
- **Routing**
  - Ribbon?
  - Finagle?
- **Observability**
  - Zipkin?
  - Metrics?
  - Hystrix?

**Datastores:** Distributed Ephemeral, Orchestrated or DBaaS

**Operational:** Many orchestration choices across public and private clouds

**Development:** Components assembled from Docker Hub as a composable “app store”

*Fill in the gaps, rapidly evolving ecosystem choices*
Forward Thinking
Forward Thinking
Forward Thinking

LEAN ENTERPRISE
Adopting Continuous Delivery, DevOps, and Lean Startup at Scale
Forward Thinking

http://eugenedvorkin.com/seven-micro-services-architecture-advantages/
Any Questions?

- Battery Ventures [http://www.battery.com](http://www.battery.com)
- Adrian’s Tweets [@adrianco](http://twitter.com/adrianco) and Blog [http://perfcap.blogspot.com](http://perfcap.blogspot.com)
- Slideshare [http://slideshare.com/adriancockcroft](http://slideshare.com/adriancockcroft)

- Monitorama Opening Keynote Portland OR - May 7th, 2014
- GOTO Chicago Opening Keynote May 20th, 2014
- Qcon New York – Speed and Scale - June 11th, 2014
- Structure - Cloud Trends - San Francisco - June 19th, 2014
- GOTO Copenhagen/Aarhus – Fast Delivery - Denmark – Sept 25th, 2014
- DevOps Enterprise Summit - San Francisco - Oct 21-23rd, 2014 #DOES14
- GOTO Berlin - Migrating to Microservices - Germany - Nov 6th, 2014
- AWS Re:Invent - Cloud Native Cost Optimization - Las Vegas - November 14th, 2014
- O’Reilly Software Architecture Conference - Fast Delivery - Boston March 16th 2015

Disclosure: some of the companies mentioned may be Battery Ventures Portfolio Companies
See [www.battery.com](http://www.battery.com) for a list of portfolio investments