Introduction to Microservices and Spring-Boot

January 6, 2016

David D Lucas

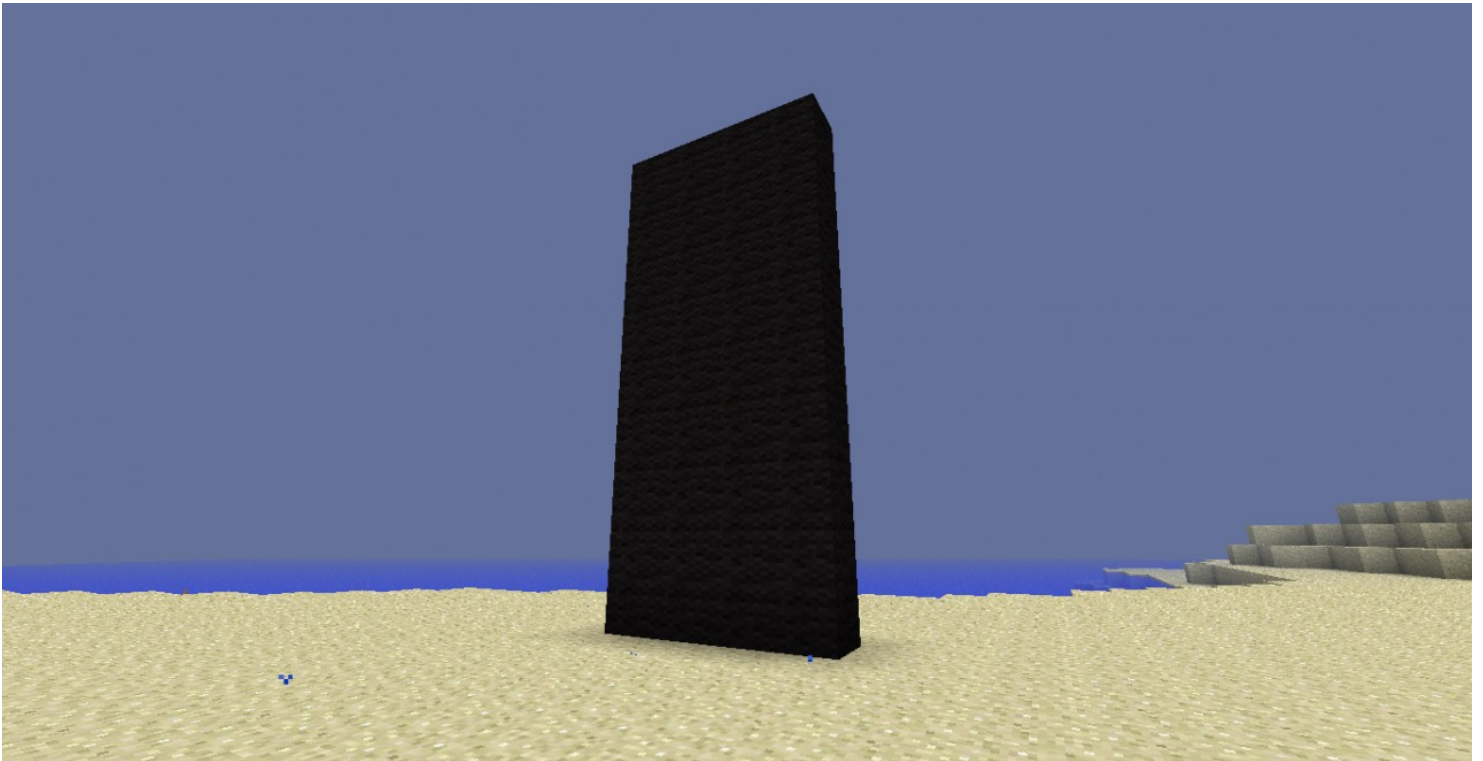Lucas Software Engineering, Inc.

ddlucas@lse.com

Agenda

- Introduction
- Microservices
- Spring-Boot
- Spring-Cloud
- Conslusion

Introduction

- Who am I ?
- What to expect ?
  - mix of discussion and demo
  - ask questions as we go

- Thanks to AIC for sponsoring !!!

Microservices

- Definitions
- Monolith vs Microservices
- How DevOPS fits ?
- Who is doing it ?

– 2001 Space Odyssey Monolith in Minecraft

Monolith : single artifact that may use multiple libraries to build a complete solution that is *deployed into a larger server with other processes* where scalability is dependent on replicating external processes along with application
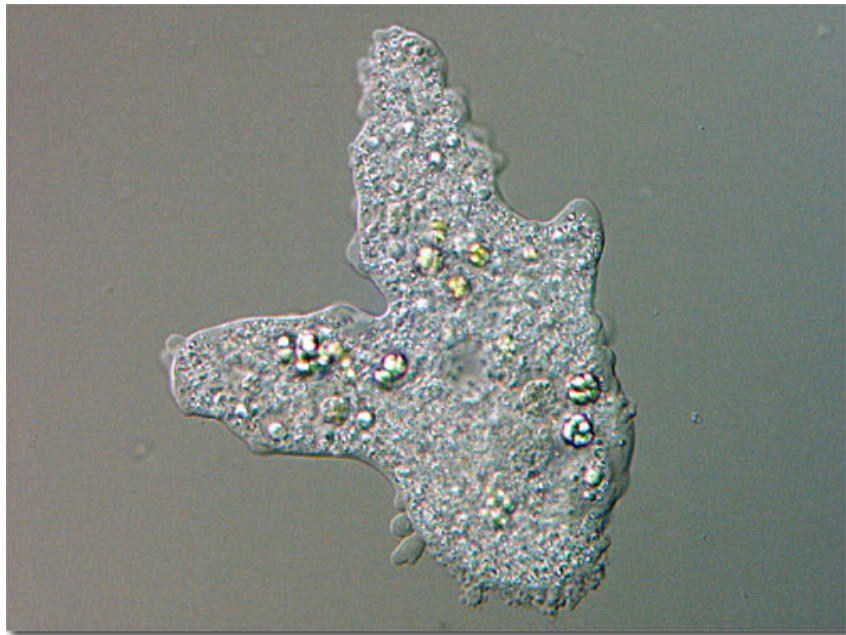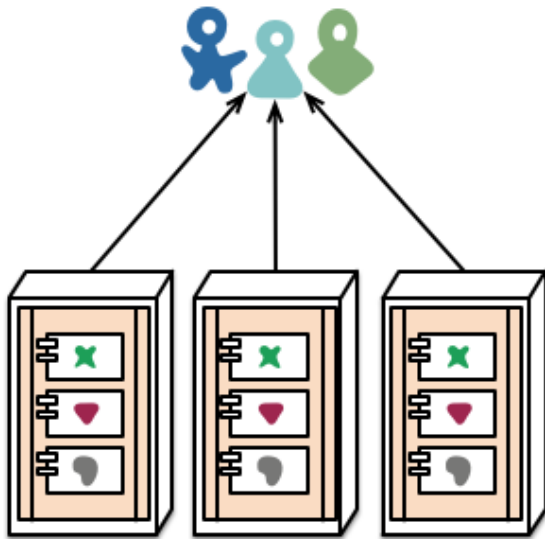

– Amoeba

Microservice: single artifact that may use multiple libraries to build a complete solution that is *deployed as a standalone process* and scalability is dependent on just replicating standalone application
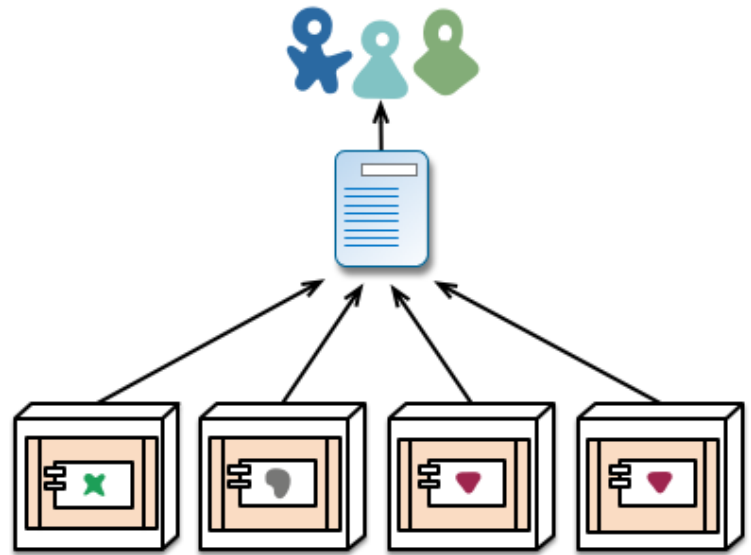
– Fed Ex Stolen Idea

https://www.youtube.com/embed/zNCrMEOqHpc

monolith - multiple modules in the same process

microservices - modules running in different processes

– Martin Fowler

They sound the same ?!?!

- Yep, they do, but they are slightly different
- How they are built is different
- How they scale is different
- How developers build them is different
- How they are managed is different
- How much code to build the service is different
- Networked Colonies versus Networked Individuals

Head to Head

Item	Monolith	Microservice
Code Base	LARGE	SMALL
Features	Gold Plated (Vendor encouraged)	Minimalist
SDLC	Waterfall	Agile
Testing	Full System	Focused Service
Releases	Large System Dependency Driven	Product Line Driven
Technology	Enterprise Driven	Team Driven
Feedback Loop	SLOW	QUICK
Automation	Some testing	almost all automation (CI / CD)
Production Access	Production Managment and Support	Developers: Trust but Verify

"Microservice definition: Loosely coupled service oriented architecture with bounded contexts" – Adrian Cockcroft

Conway's Law

"Any organizaiton that designs a system will inevitably produce a design whose structure is a copy of the organization's communications structure."
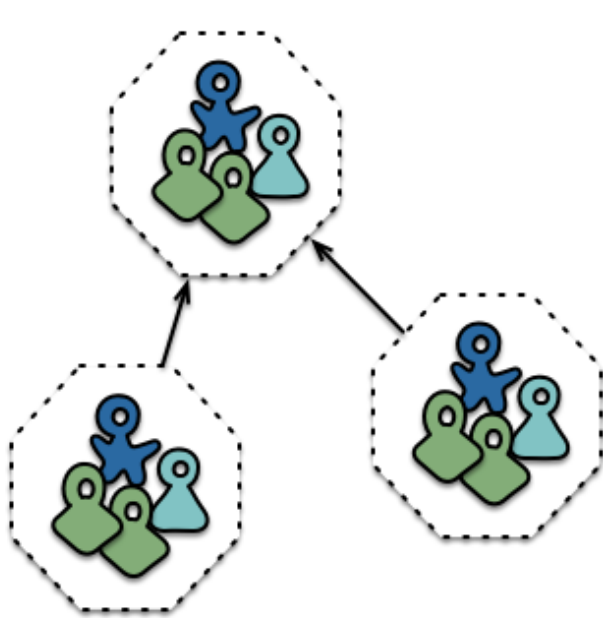
– Melvin Conway (1968)

Siloed functional teams...   ... lead to silod application architectures.
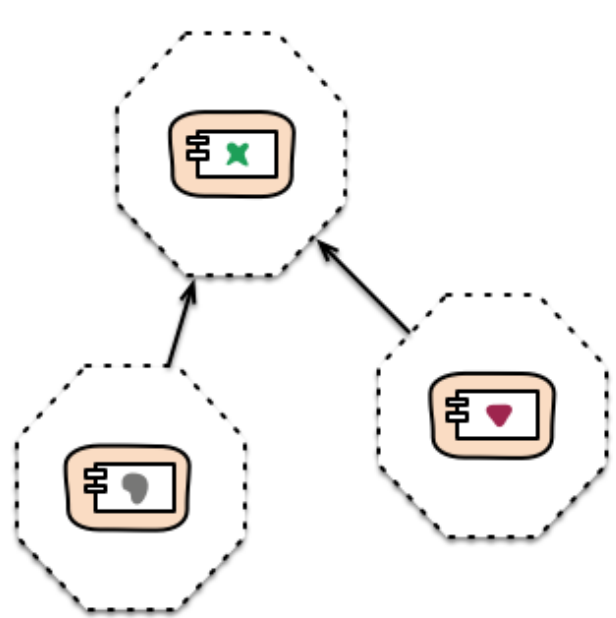   Because Conway's Law
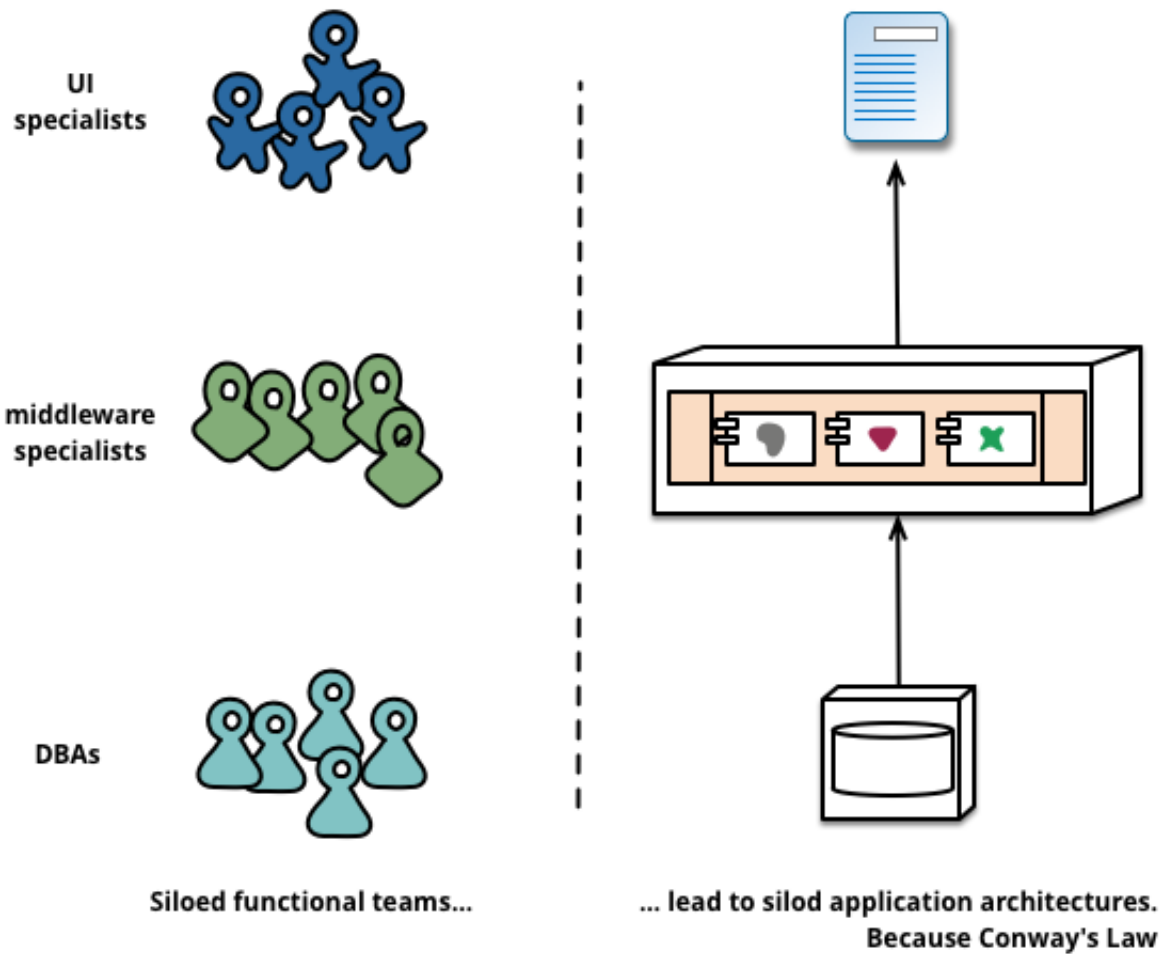
– Martin Fowler

Microservices attributes

- organize around business capabilities
- components versus services
- PRODUCTS not Projects !!!
- smart end points and dumb pipes
- decentralized control and management
- infrastructure automation

Cross-functional teams...

**... organised around capabilities
Because Conway's Law**

– Martin Fowler

DevOPS !

- DevOPS is a culture, not a title nor role
- Continuous Integration, Testing, Delivery
- Development Team takes more responsibility for operations and support
- Roles shift toward PRODUCT, away from single stack specialists
- Agile product teams will gravitate to microservices
- It is more about eliminating external dependences *(remove middle person)*

Requirements For Microservices

- Provision machines within hours
- Alot of Monitoring !!!
- Automatic deploy and rollback in prodcution (CI / CD)
- Culture change, developers own support of product

- 2 pizza product teams, direct customer access
- must be cheap to re-write
- Failure will happen
- Automatic scalability and failover

What tools are needed ?

- Tooling for building and deploying
- Configuration (env properties)
- Service Discovery Locator
- Routing
- Observability / Monitoring
- Datastores
- Orchestration and Deployment Infrastructure
- Development: Languages and Container

Companies doing Microservices

- Guilt
- Hailo
- Twitter
- Netflix

Gilt Microservices (online shopping)

- Ion Cannon / SBT / Rake (Tooling)
- Decider (Configuration)
- Finagle / Zookeeper (Discovery)
- Akka / Finagle / Netty (Routing)
- Zipkin (Observability / Monitoring)
- MongoDB / Postgres / Voldemort (Datastore)
- AWS (Deployment)
- Scala / Ruby in Docker Containers (processes)

Hailo Microservices (UK like Uber)

- Hubot / Janky / Jenkins (Tooling)
- ??? (Configuration)
- go-platform (Discovery)
- go-platform / RabbitMQ (Routing)
- Request trace (Monitoring)
- Cassandra based (Datastore)
- AWS (Deployment)
- Go using Docker Containers (processes)

Twitter Microservices

- Decider (Configuration)
- Finagle / Zookeeper (Discovery)
- Finagle / Netty (Routing)
- Zipkin (Observability / Monitoring)
- Manhattan (Cassandra like Datastore)
- Aurora deployment using Mesos (Orchestration)
- Scala in Docker Containers (processes)

Netflix OSS

- Asgard / Animator (Tooling)
- Edda / Archaius (Configuration)
- Eureka / Prana (Discovery)
- Denominator / Zull / Netty / Ribbon 2.0 (Routing)
- Hystrix / Pytheus / SALP (Observability / Monitoring)
- Ephemeral / Dynomite / Memcached / Astyanax / Staash / Priam / Cassandra (Datastore)
- Asgard / AWS / Eucalptus (Orchestration)
- Java / Groovy / Scala / Clojure / Python / Node.js with AMI / Docker Containers (processes)

How does Spring fit ?

- Spring is about easy use of frameworks
- Spring provides abstraction and wiring
- Spring wiring can be configuration driven
- Inverson of Control / Dependency Injection (IOC / DI)
- New kid on the block: Spring-Boot !!!

"Spring Boot lets you pair-program with the Spring team."
–Josh Long, @starbuxman

Spring-Boot

- Builds upon Spring Framework success
  - http://docs.spring.io/spring/docs/4.2.x/spring-framework-reference/htmlsingle
  - http://docs.spring.io/spring-boot/docs/1.3.x/reference/htmlsingle/

- Create stand-alone, production-grade applications that just run
- Opinioned view of Spring, best of bread setup
- Auto-configurations
- Easy to get running !!!
- Alot of free features by using starter framework and templates
- NO XML Required !!!

Spring-Boot (cont)

- Java 7 or 8 (8 recommended)
- Maven, Gradle, Ant support
- Parent starter dependencies with default implementations
- Annonations of Best Practices
- External Configurations, Profiles, Logging
- Web MVC, Embedded Containers
- Datasource SQL + NO-SQL
- Messaging / JMS
- Testing Utils and Boot Applications
- @Conditions (conditional Bean setup)
- Auto-configurations
- Containers: HTTP, JMX, SSH

Spring-Boot (cont)

- Embedded Web Containers: Tomcat, Jetty, Undertow
- Groovy Environment Manager (GVM : http://gvmtool.net) Spring CLI

  curl -s get.gvmtool.net l bash

- DEMO

  - app.groovy Hello World ! (examples/01*hello*groovy)

Spring-Boot (cont)

- Fast Track Setup

  - Starter Project http://start.spring.io
  - STS Starter http://spring.io/tools/sts

- DEMO
  - DEMOUI Java Hello World ! (examples/02_hello)

- Actuator
- CraSH

Spring-Cloud

- Spring Boot starters specialized for the cloud offereings
- Spring Cloud uses NetflixOSS as first integration
- Spring Cloud now has many more starter components
- http://projects.spring.io/spring-cloud
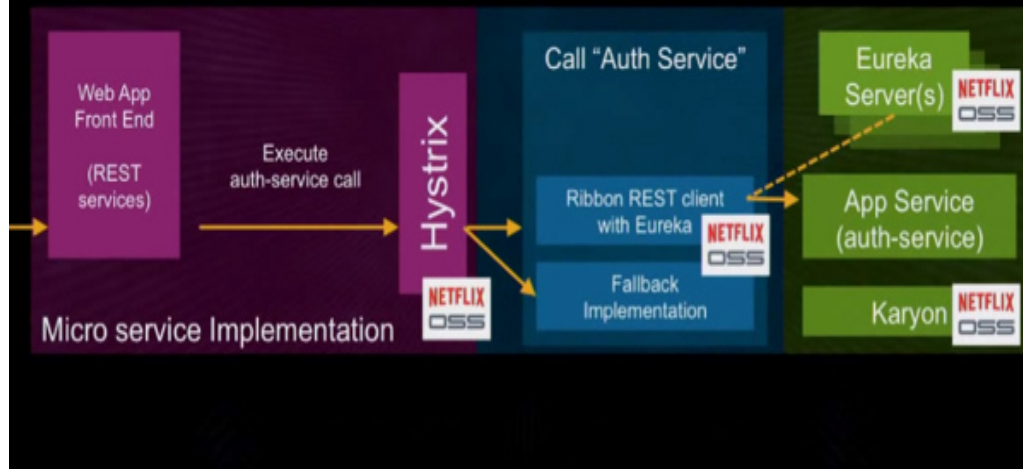
Spring Cloud Components

Spring-Cloud Features

- Distributed/versioned configuration
- Service registration and discovery
- Routing
- Service-to-service calls
- Load balancing
- Circuit Breakers
- Global locks
- Leadership election and cluster state
- Distributed messaging

Core Components of a Highly Available Service

Microservices & Spring

- Spring Boot gives organizations working templates to build on
- Spring Cloud gives integration to the tools we need to run our microservices
- DEMO (examples/03_microservice)
  - Zuul / Eureka / Hystrix / Actuator
  - Proxy and Load Balancing

Conclusions

- Spring

  - delivers again on ease of use
  - delivers the tools for microservices

- Microservices

  - may not be for everyone
  - requires commitment at all levels
  - hard to break up big program in a big company into small teams

- Must version everything: properties, source, configurations
- Must use service discovery locator pattern

Additional Questions?

THANKS !

Resources

- Example Code and Copy of Slides
  - https://github.com/lseinc/spring-boot

- Books
  - Spring in Action, 4th Edition, 2015, Walls
  - Learning Spring Boot, 2014, Turnquist

Resources (cont)

- Links
  - Spring Boot Reference http://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle
  - Martin Fowler http://martinfowler.com/articles/microservices.html http://martinfowler.com/bliki/CircuitBreaker.html
  - Gilt http://www.infoq.com/presentations/scale-gilt
  - Groupon http://www.slideshare.net/mcculloughsean/itier-breaking-up-the-monolith-philly-ete
  - HailO http://speakerdeck.com/mattheath/scaling-micro-services-in-go-highload-plus-plus-2014
  - Twitter
    - http://www.infoq.com/presentations/Twitter-Timeline-Scalability
    - http://www.infoq.com/presentations/twitter-soa
    - http://www.infoq.com/presentations/Zipkin

  - NetflixOSS
    - http://www.slideshare.net/SpringCentral/syer-gibbcloud
    - https://www.youtube.com/watch?v=X5DRXCKJH_M
    - http://www.youtube.com/watch?v=p7ysHhs5hl0

- http://www.youtube.com/watch?v=ZfYJHtVL1_w
- http://www.youtube.com/watch?v=CriDUYtfrjs
- http://www.youtube.com/watch?v=aEuNBk1b5OE

Resources (cont)

- Links (cont)
  - ACME Air Web on Docker:
    http://ispyker.blogspot.com/2014/05/cloud-services-fabric-and-netflixoss-on.html
    https://github.com/EmergingTechnologyInstitute/acmeair-netflixoss-dockerlocal
    http://ispyker.blogspot.com/2014/06/open-source-release-of-ibm-acme-air.html