

# Migrating A Monolith Application to *Cloud Native*

Carlos Queiroz - @cax  
[cqueiroz@pivotal.io](mailto:cqueiroz@pivotal.io)

# Agenda – Day 1

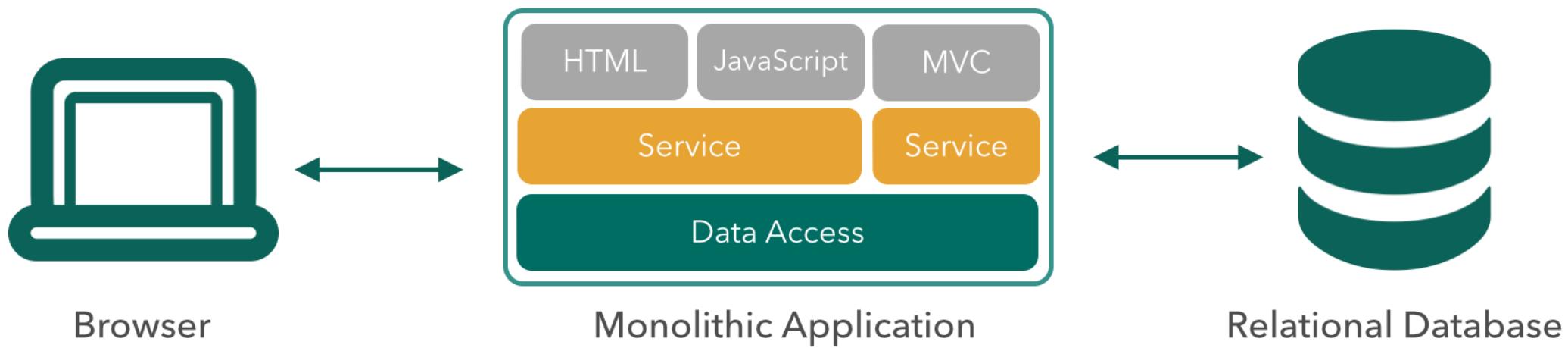
- 9:00 AM - 10:30 AM - **Session 1:** *Cloud-Native Architecture Overview and Building Twelve-Factor Apps with Spring Boot*
- 10:30 AM - 10:45 AM - Morning Break
- 10:45 AM - 12:30 PM - **Session 2:** *Operating Twelve-Factor Apps with Cloud Foundry*
- 12:30 PM - 1:30 PM - Lunch
- 1:30 PM - 2:30 PM - **Session 3:** *Spring Data JPA*
- 2:30 PM - 3:30 PM - **Session 4:** *Splitting the Monolith: Part One*
- 3:30 PM - 3:45 PM - Afternoon Break
- 3:45 PM - 5:00 PM - **Session 5:** *Microservices with Polyglot Persistence*

# Agenda – Day 2

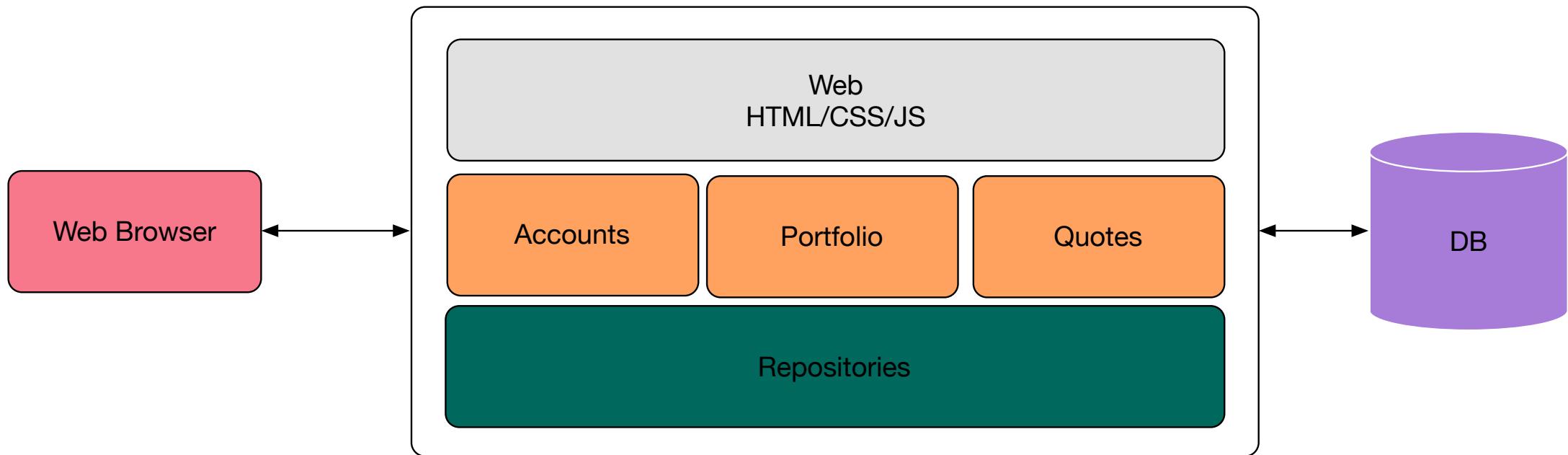
- 9:00 AM - 10:30 AM - **Session 6:** *Cloud-Native Architecture Patterns: Part One*
- 10:30 AM - 10:45 AM - Morning Break
- 10:45 AM - 12:30 AM - **Session 7:** *Cloud-Native Architecture Patterns: Part Two*
- 12:30 PM - 1:30 PM - Lunch
- 1:30 PM - 3:00 PM - **Session 8:** *Securing Cloud-Native Applications*
- 3:00 PM - 3:15 PM - Afternoon Break
- 3:15 PM - 5:00 PM - **Session 9:** *The API Gateway Pattern*

```
$ git clone https://github.com/caxqueiroz/MigratingAMonolithApplicationToCloudNative  
$ git fetch --all
```

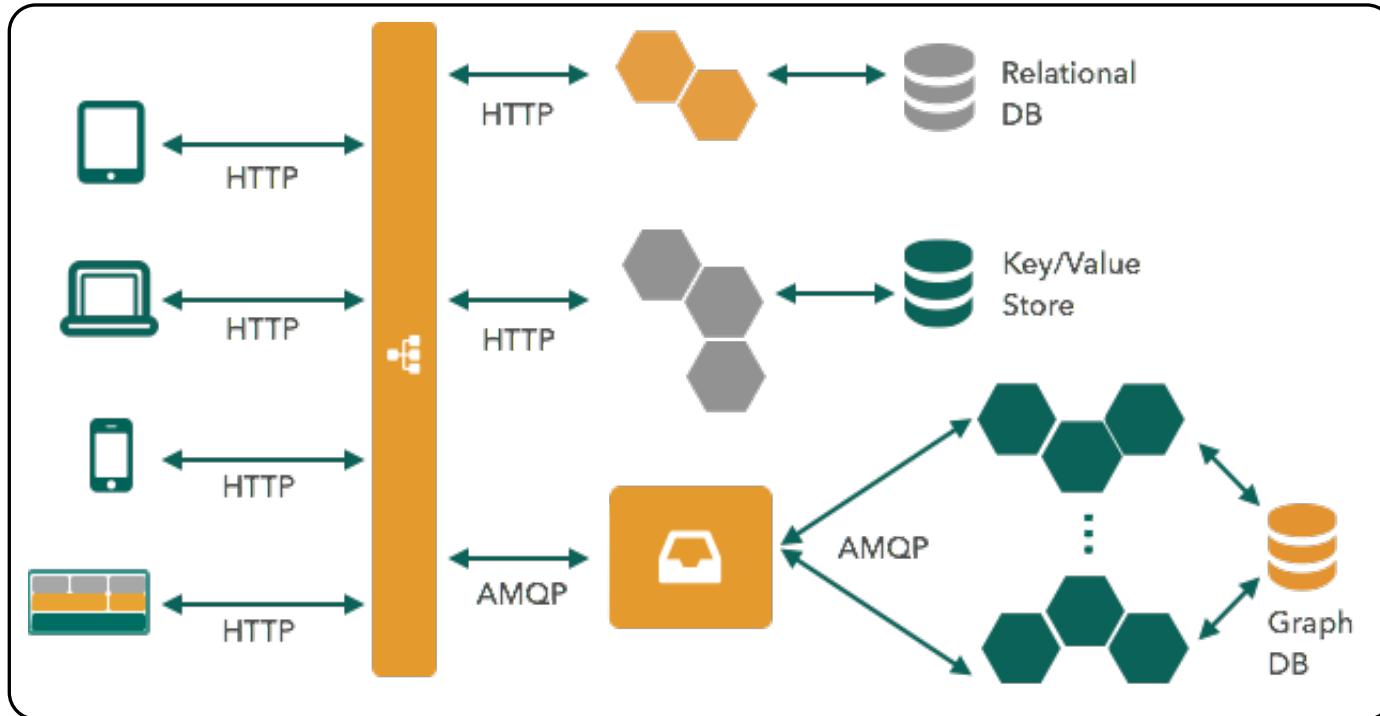
# A Monolith Application



# SpringTrader Application



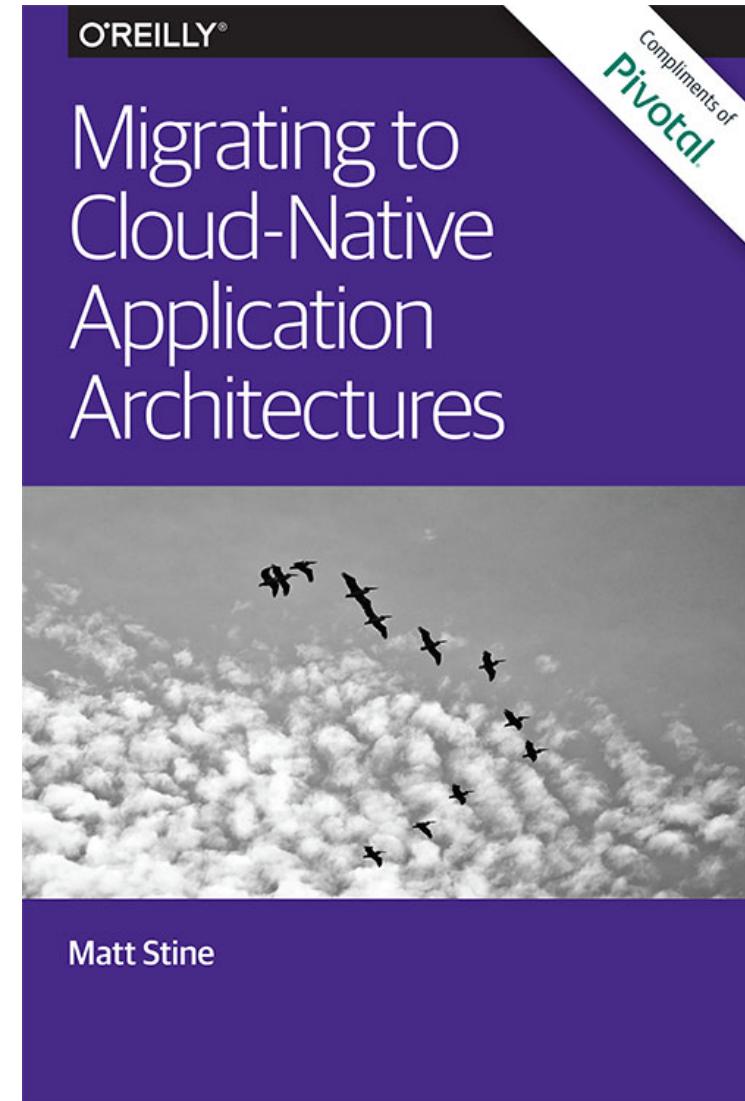
# Goal



# How?

Pivotal<sup>®</sup>

# Cloud Native Applications

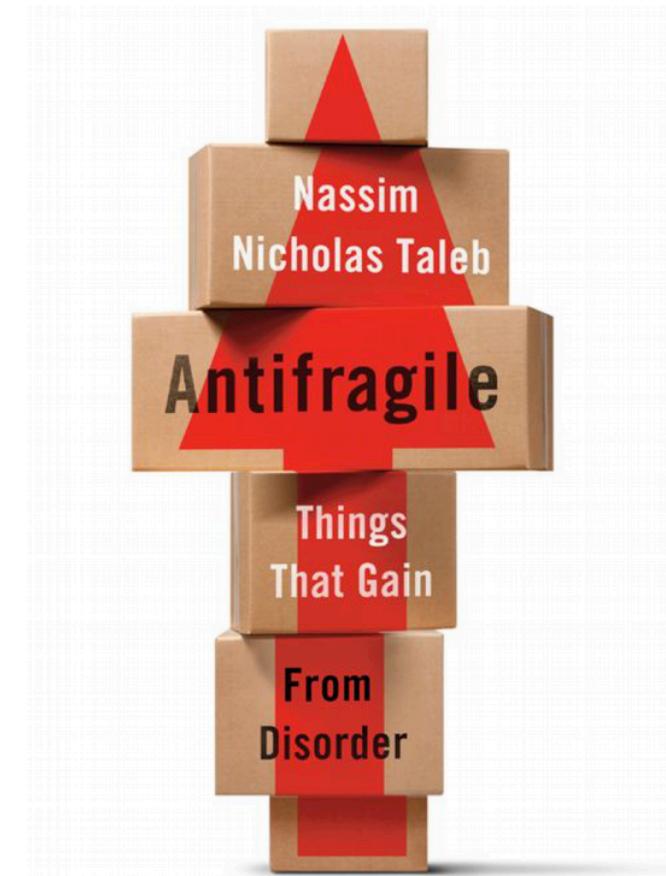


# Why?

- SPEED
- SAFETY
- SCALE
- MOBILITY

# Cloud Native Applications

- Twelve Factor Apps
- **Microservices**
- Self-Service Agile Infrastructure
- API-Based Collaboration
- Antifragility



# Microservice

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

-- Adrian Cockcroft

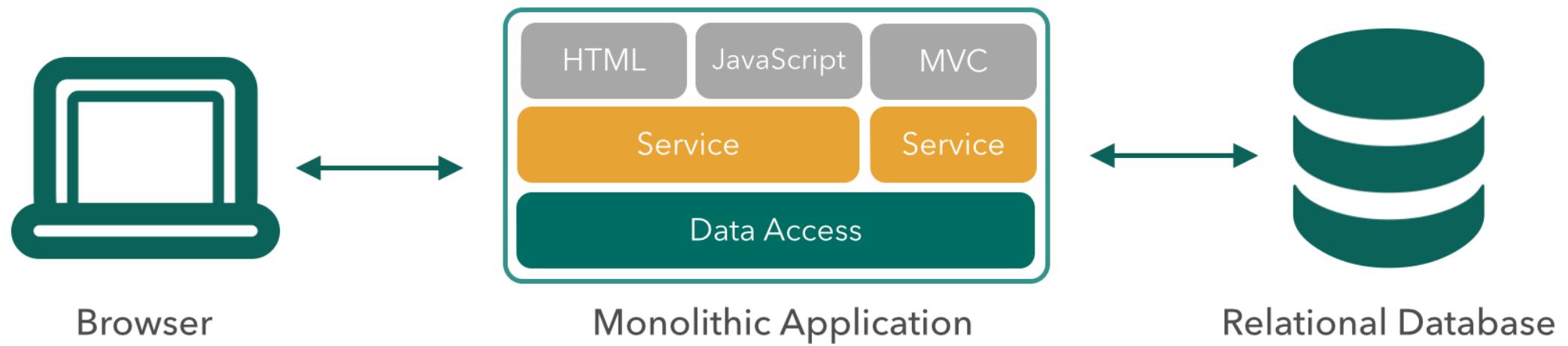
# Loosely Coupled

*If every service has to be updated in concert,  
it's not loosely coupled!*

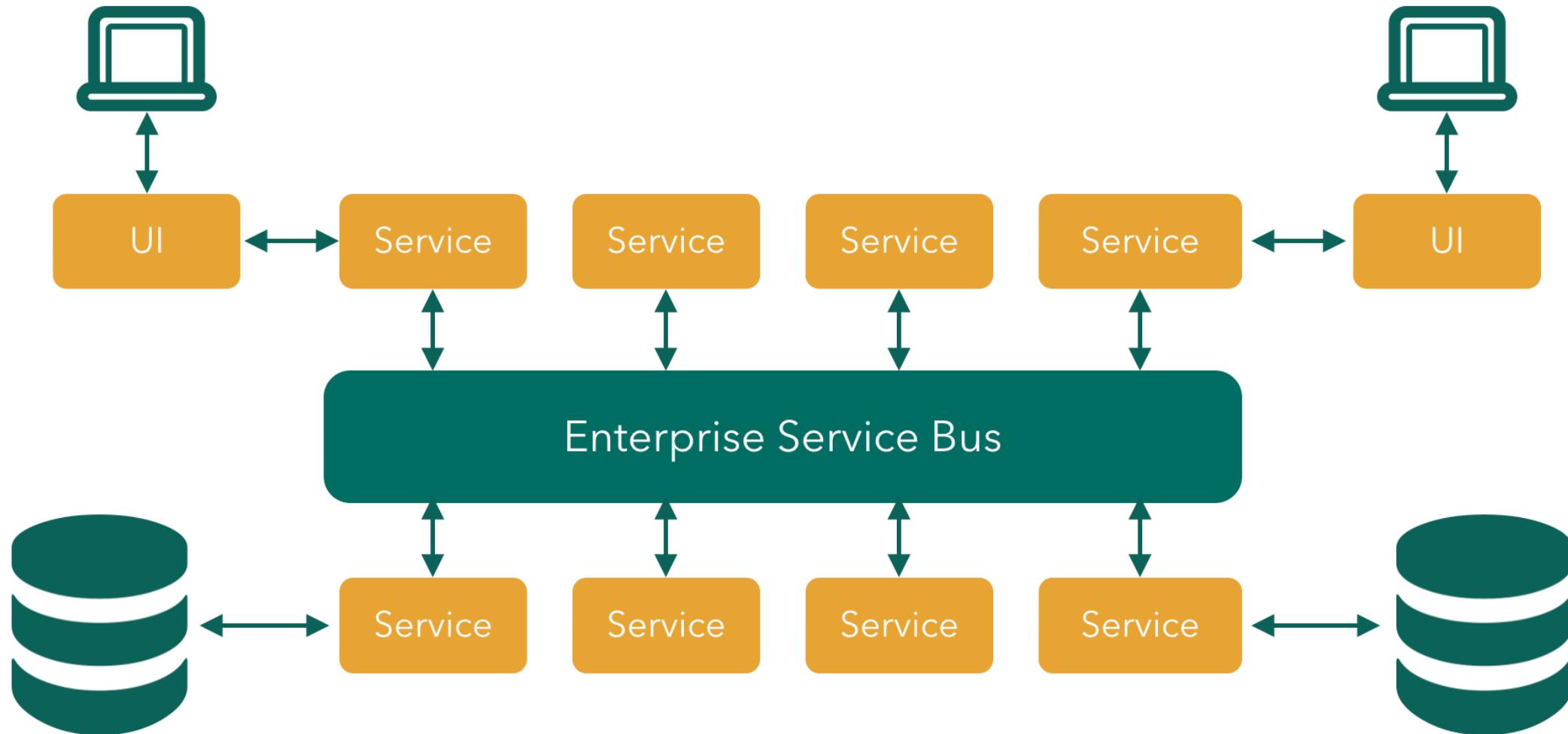
# Bounded Contexts

*If you have to know about surrounding services you don't have a bounded context.*

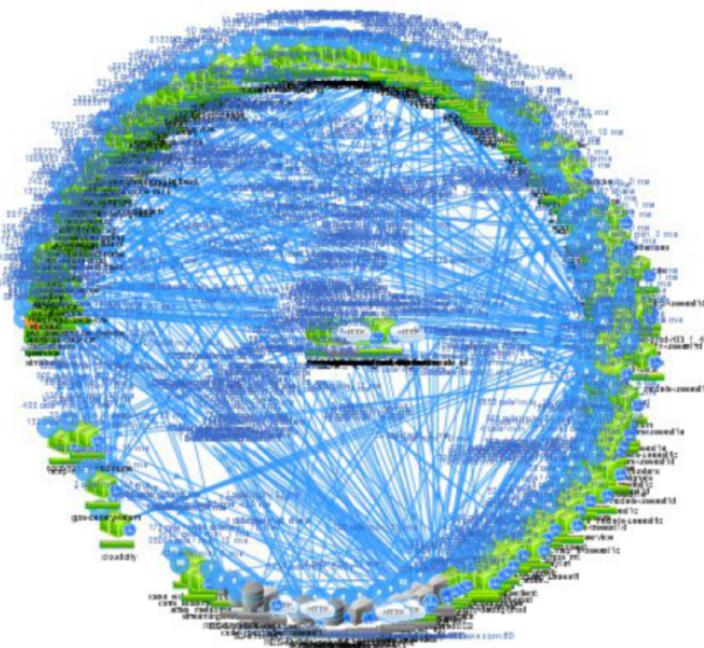
# Not monoliths...



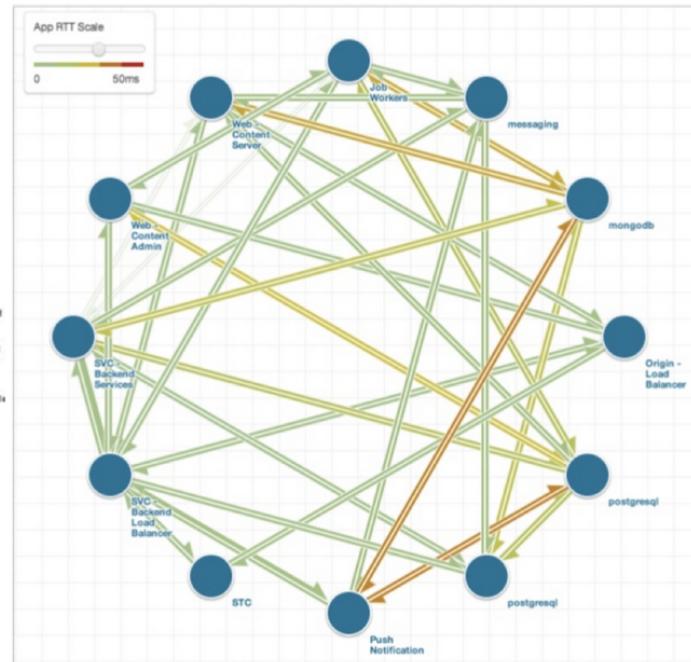
# Not traditional SOA (esb-centric)



# But Microservices...



*Netflix*



*Gilt Groupe (12 of 450)*



*Twitter*

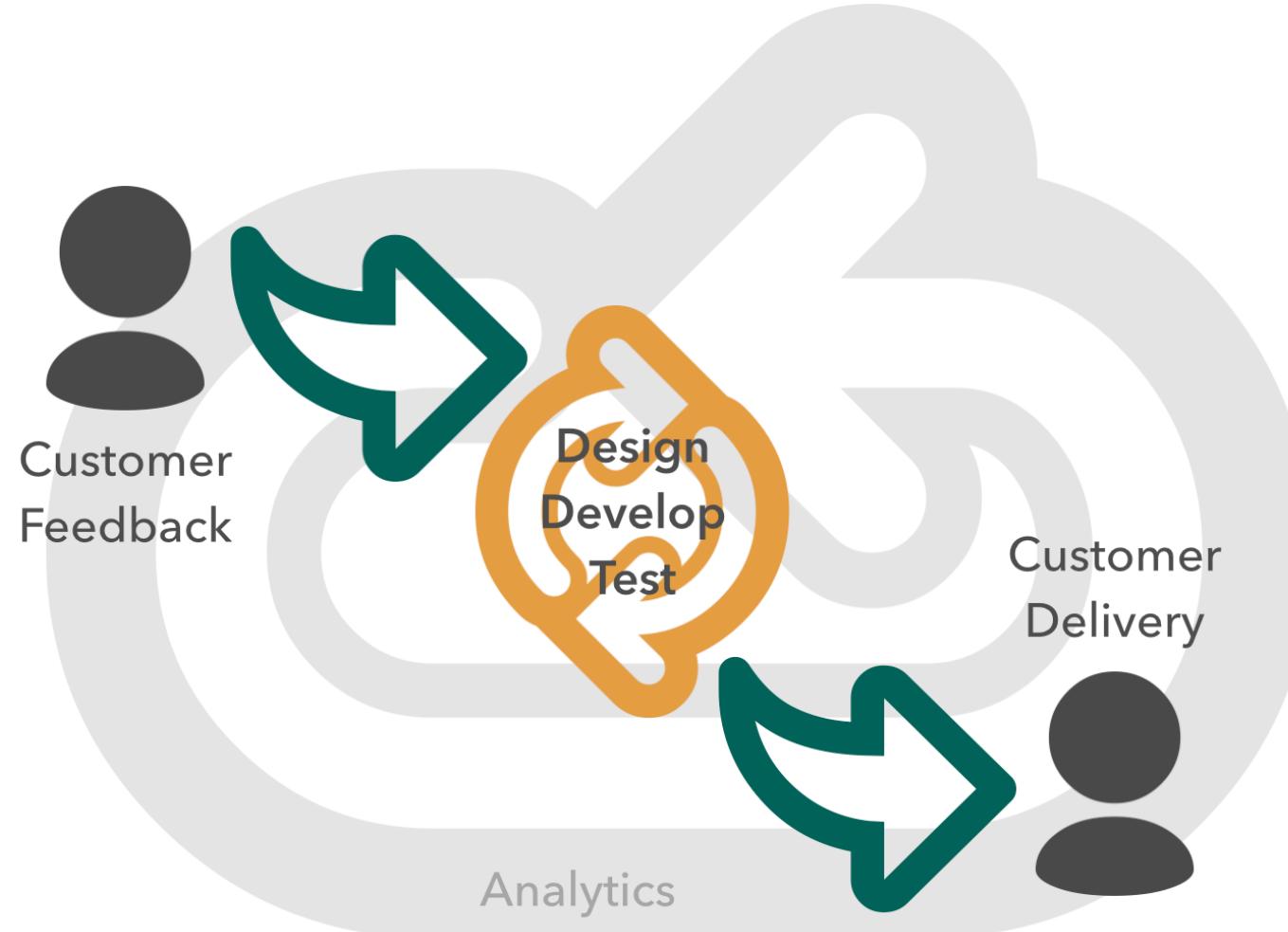


Pivotal®

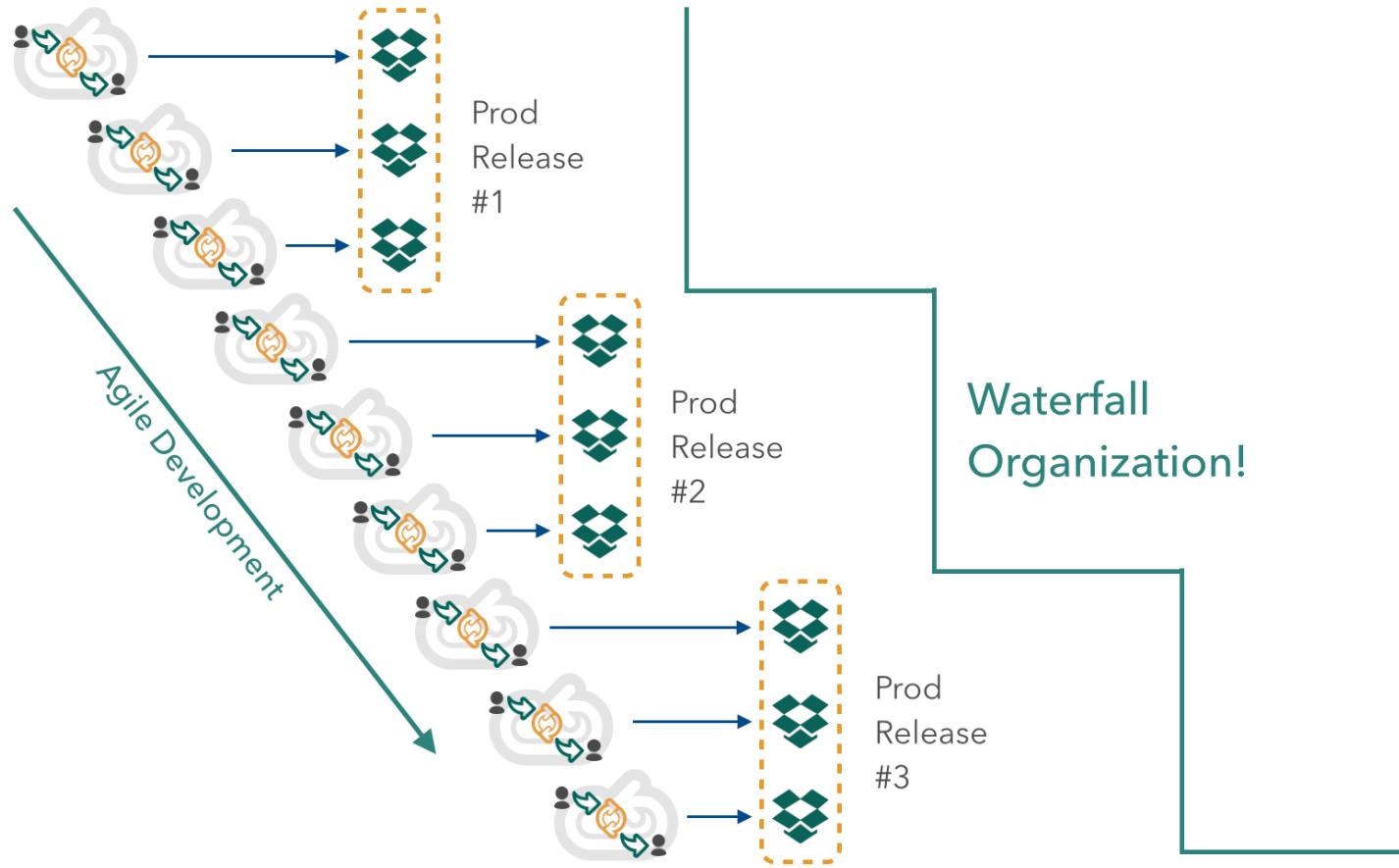
# Enabling Continuous Delivery



# Agile - Iterative Feedback loops



# Waterscrumfall

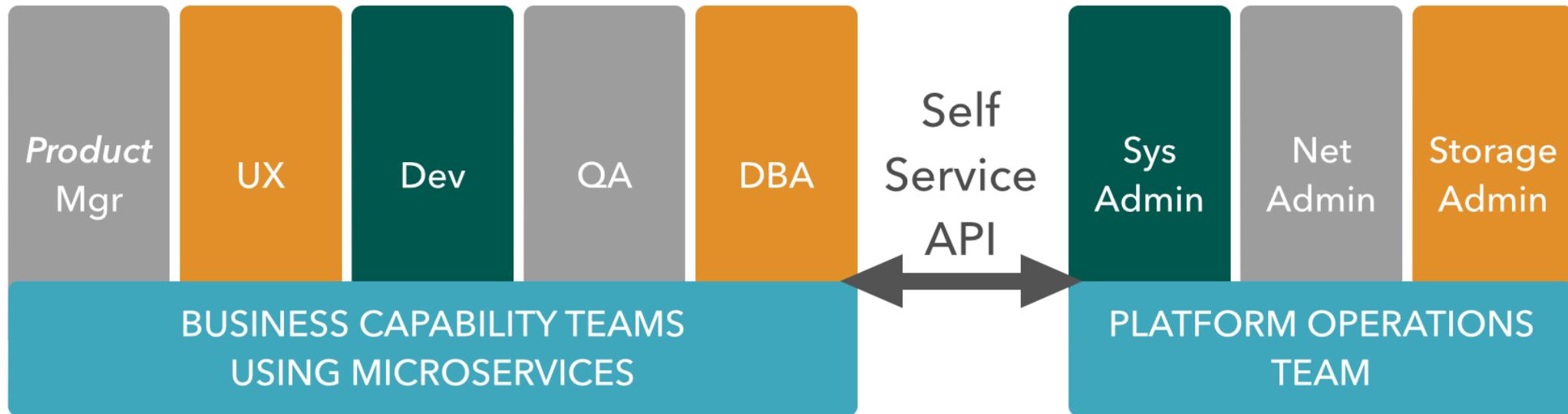


# Conway's Law

*“Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure.”*

-- Melvyn Conway, 1967

# The Inverse Conway Manoeuvre



# Continuous Delivery



# Building Twelve-Factor Apps with Spring Boot

Pivotal®



# THE TWELVE-FACTOR APP

## INTRODUCTION

In the modern era, software is commonly delivered as a service: called *web apps*, or *software-as-a-service*. The twelve-factor app is a methodology for building software-as-a-service apps that:

- Use **declarative** formats for setup automation, to minimize time and cost for new developers joining the project;
- Have a **clean contract** with the underlying operating system, offering **maximum portability** between execution environments;
- Are suitable for **deployment** on modern **cloud platforms**, obviating the need for servers and systems administration;
- **Minimize divergence** between development and production, enabling **continuous deployment** for maximum agility;
- And can **scale up** without significant changes to tooling, architecture, or development practices.

The twelve-factor methodology can be applied to apps written in any programming language, and which use any combination of backing services (database, queue, memory cache, etc).

# Patterns

- Cloud-native application architectures
- Optimized for speed, safety, & scale
- Declarative configuration
- Stateless/shared-nothing processes
- Loose coupling to application environment

# 12 Factors (1/2)

- One Codebase in Version Control
- Explicit Dependencies
- Externalized Config
- Attached Backing Services
- Separate Build, Release, and Run Stages
- Stateless, Shared-Nothing Processes

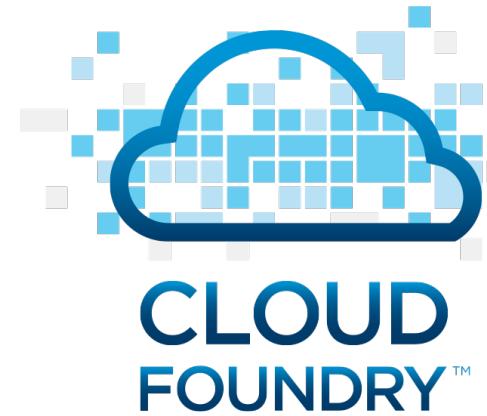
# 12 Factors (2/2)

- Export Services via Port binding
- Scale Out Horizontally for Concurrency
- Instances Should Be Disposable
- Dev/Prod Parity
- Logs Are Event Streams
- Admin Processes

# Platforms

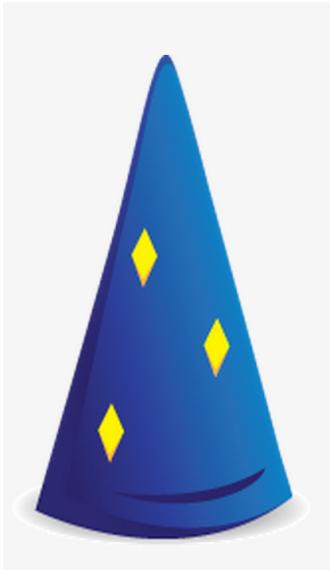


<http://heroku.com>



<http://cloudfoundry.org>

# Frameworks



<http://www.dropwizard.io>



<http://projects.spring.io/spring-boot>

# Spring Boot

- <http://projects.spring.io/spring-boot>
- Opinionated convention over configuration
- Production-ready Spring applications
- Embed Tomcat, Jetty or Undertow
- *STARTERS*
- Actuator: Metrics, health checks, introspection

The screenshot shows the Spring Initializr web application. At the top, it says "SPRING INITIALIZR" and "Bootstrap your application now". Below this, there are two main sections: "Project metadata" and "Project dependencies".

**Project metadata:**

- Group: com.mattstine.twelvefactor
- Artifact: hello-spring-boot
- Name: hello-spring-boot
- Description: Twelve Factor Spring Boot Example
- Package Name: com.mattstine.twelvefactor.springboot
- Type: Maven Project

**Project dependencies:**

- Core:** Security, AOP, Atomikos (JTA), Bitronix (JTA)
- Web:** Jersey (JAX-RS), Rest Repositories, HATEOAS, Mobile

Checkmarks indicate that "Web" and "Jersey (JAX-RS)" are selected.

# Labs!!

Pivotal®