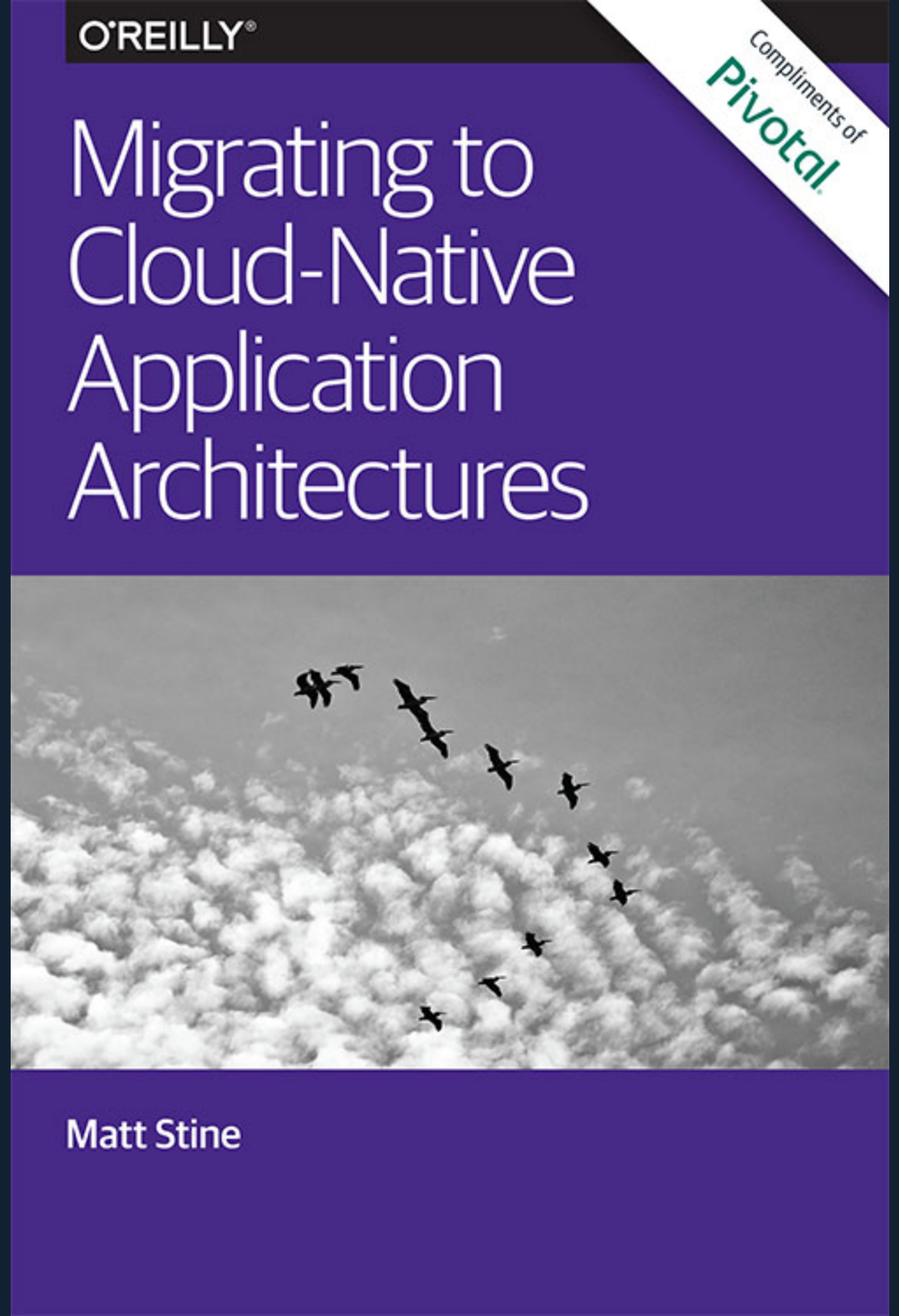
# CLOUD-NATIVE APPLICATION ARCHITECTURES

## WITH SPRING AND CLOUD FOUNDRY

# FREE

Compliments of Pivotal

http://bit.ly/cloud-native-book



# SESSION ONE

# CLOUD-NATIVE ARCHITECTURE OVERVIEW

# WHY?

# SPEED
# SAFETY
# SCALE
# MOBILITY

# WHAT?

# WHAT?

» Twelve Factor Apps (http://12factor.net)

» Microservices

» Self-Service Agile Infrastructure

» API-based Collaboration

» Antifragility


THERE SEEMS TO BE SOME HYPE...
http://www.etsy.com/shop/sharpwriter
9
HEUSER 2012

**Jessica Kerr** @jessitron · Jan 20

.@mstine If the distinguishing feature of microservices vs SOA is the way they connect, then is the name (focused on size) terrible?

3 12

**Matt Stine**
@mstine

@jessitron "microservices" is one of the worst naming disasters of our time

RETWEETS 12 FAVORITES 7

10:56 PM - 20 Jan 2015

# DEFINE: MICROSERVICE

“Loosely coupled service oriented architecture with bounded contexts...”
Adrian Cockcroft

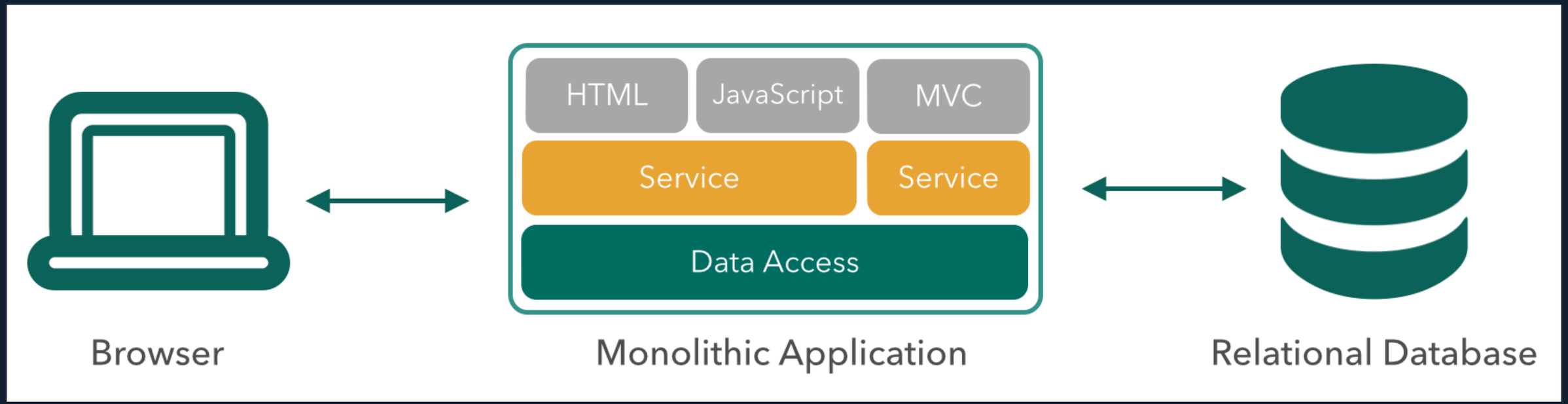
# LOOSELY COUPLED

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

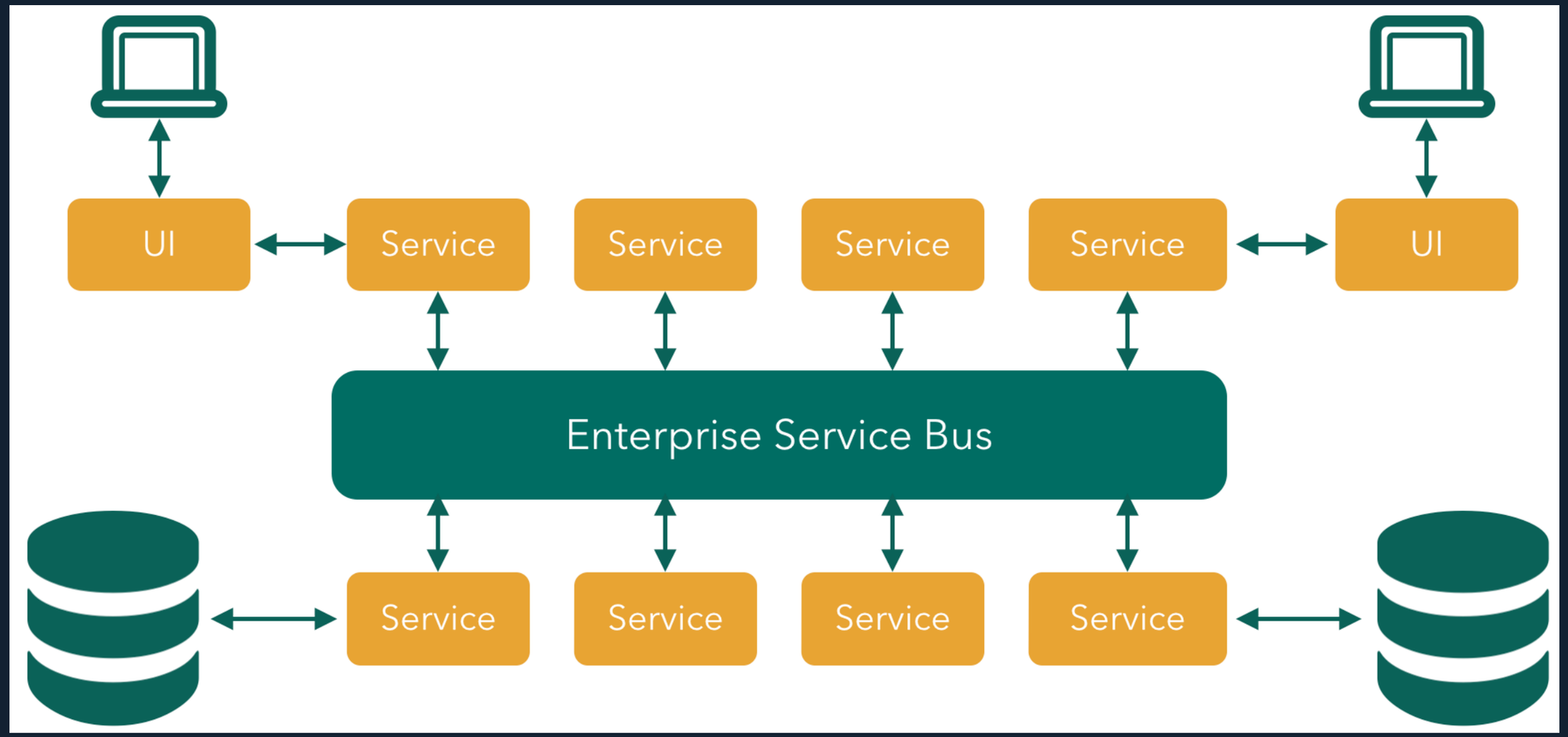
# BOUNDED CONTEXTS

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

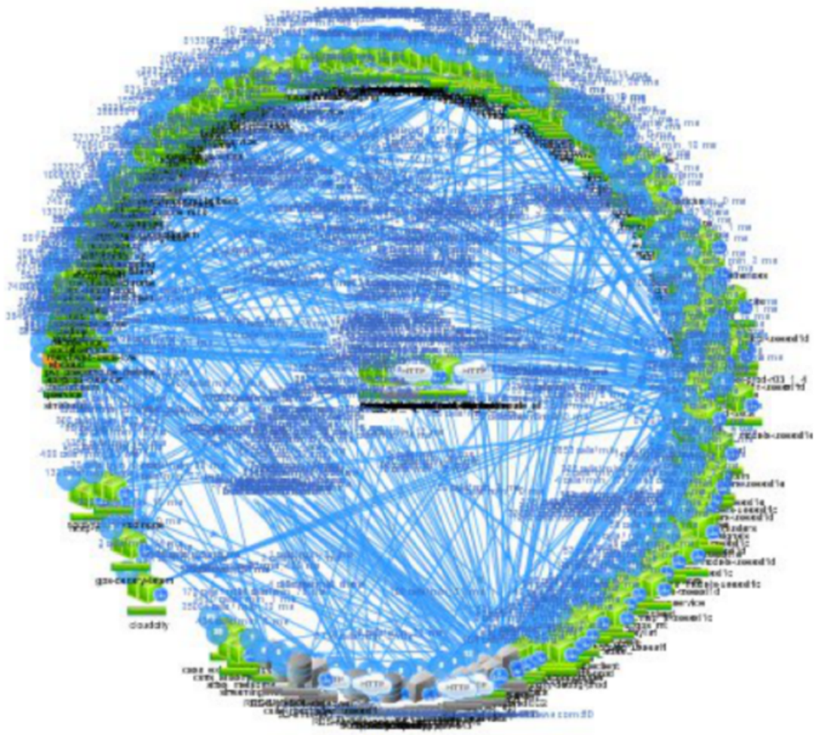
# NOT MONOLITHS...



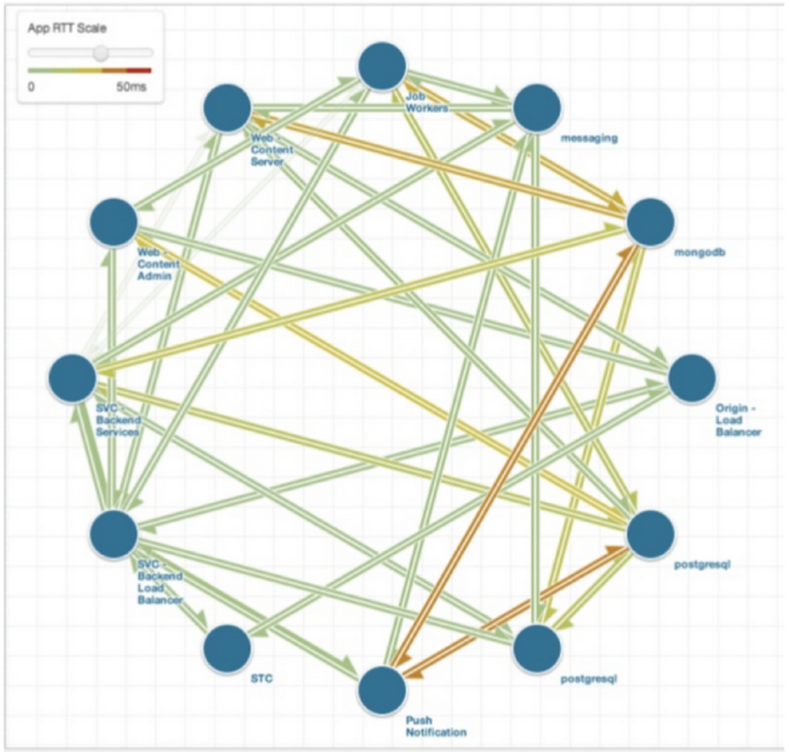
# NOT TRADITIONAL (ESB-CENTRIC) SOA...



# BUT MICROSERVICES!



*Netflix*

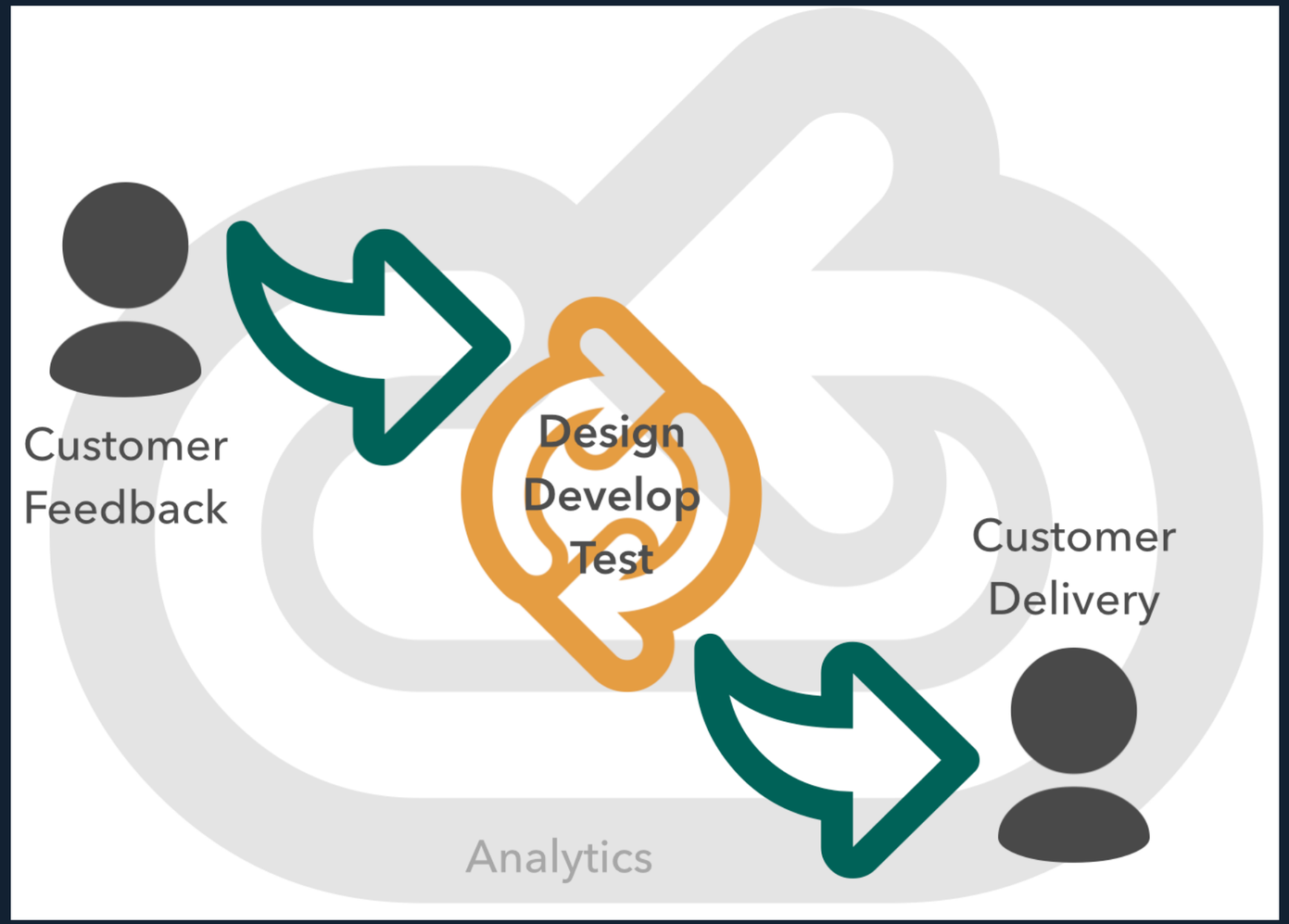



*Gilt Groupe (12 of 450)*

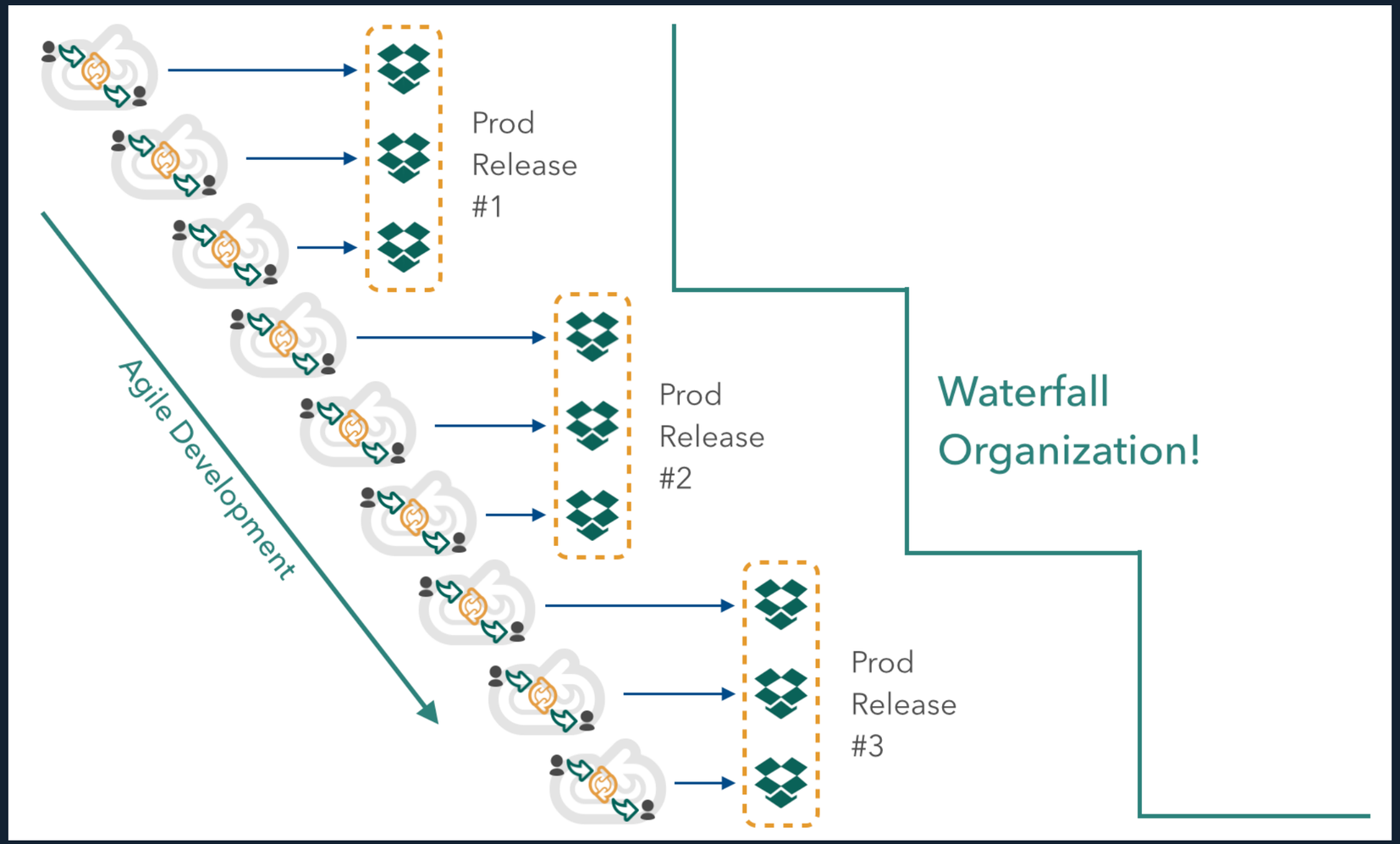
*Twitter*

# ENABLING CONTINUOUS DELIVERY

# AGILE: ITERATIVE FEEDBACK LOOPS



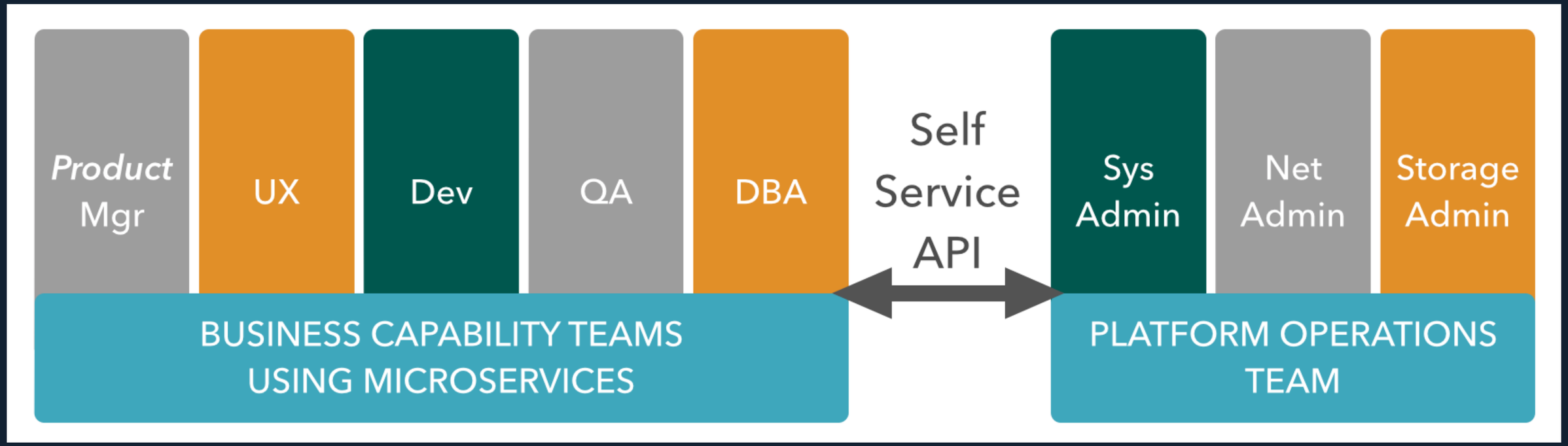
# WATERSCRUMFALL



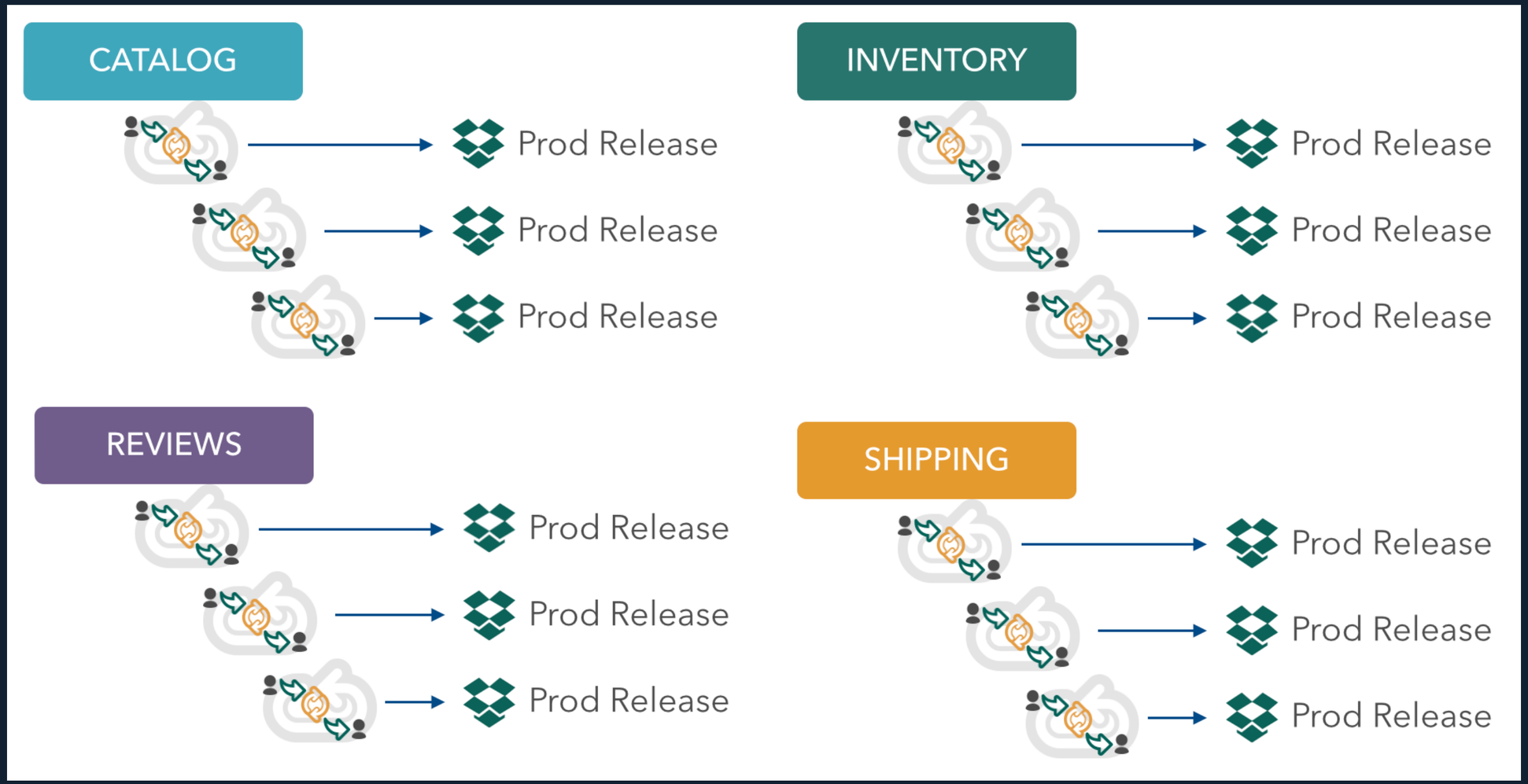
# CONWAY'S LAW

“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 MANUEVER



# CONTINUOUS DELIVERY



# BUILDING TWELVE-FACTOR APPS WITH SPRING BOOT

# The Twelve-Factor App



## In[illegible]DUCTION

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

# PATTERNS

» Cloud-native application architectures

» Optimized for speed, safety, & scale

» Declarative configuration

» Stateless/shared-nothing processes

» Loose coupling to application environment

# TWELVE 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

# TWELVE 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

# HTTP://HEROKU.COM


# HTTP://CLOUDFOUNDRY.ORG

# MICROFRAMEWORKS

» Dropwizard (http://www.dropwizard.io/)

» Spring Boot (http://projects.spring.io/spring-boot/)

# SPRING BOOT

» http://projects.spring.io/spring-boot

» Opinionated convention over configuration

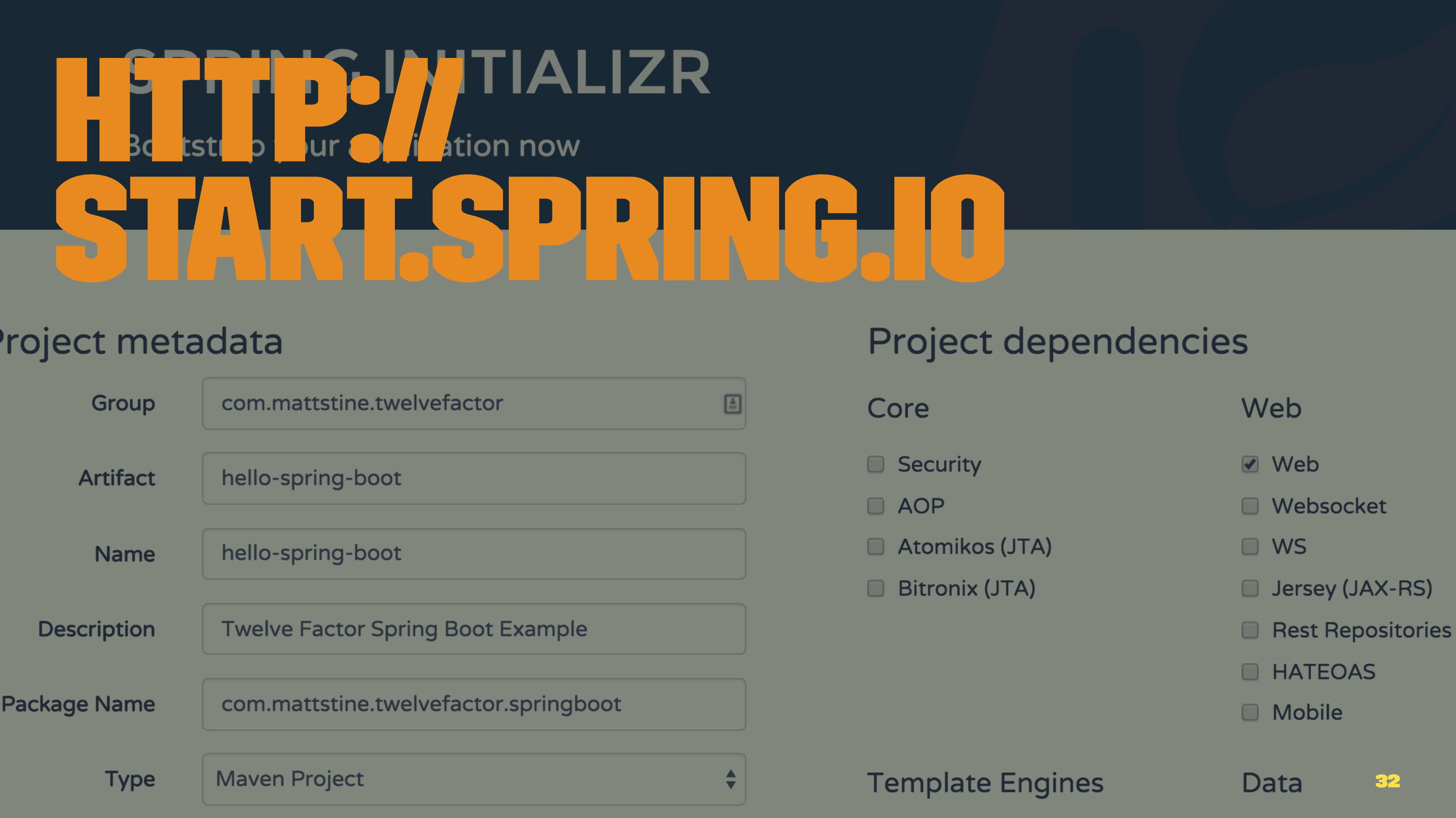
» Production-ready Spring applications

» Embed Tomcat, Jetty or Undertow

» STARTERS

» Actuator: Metrics, health checks, introspection

# HTTP://START.SPRING.IO



# TO THE LABS!