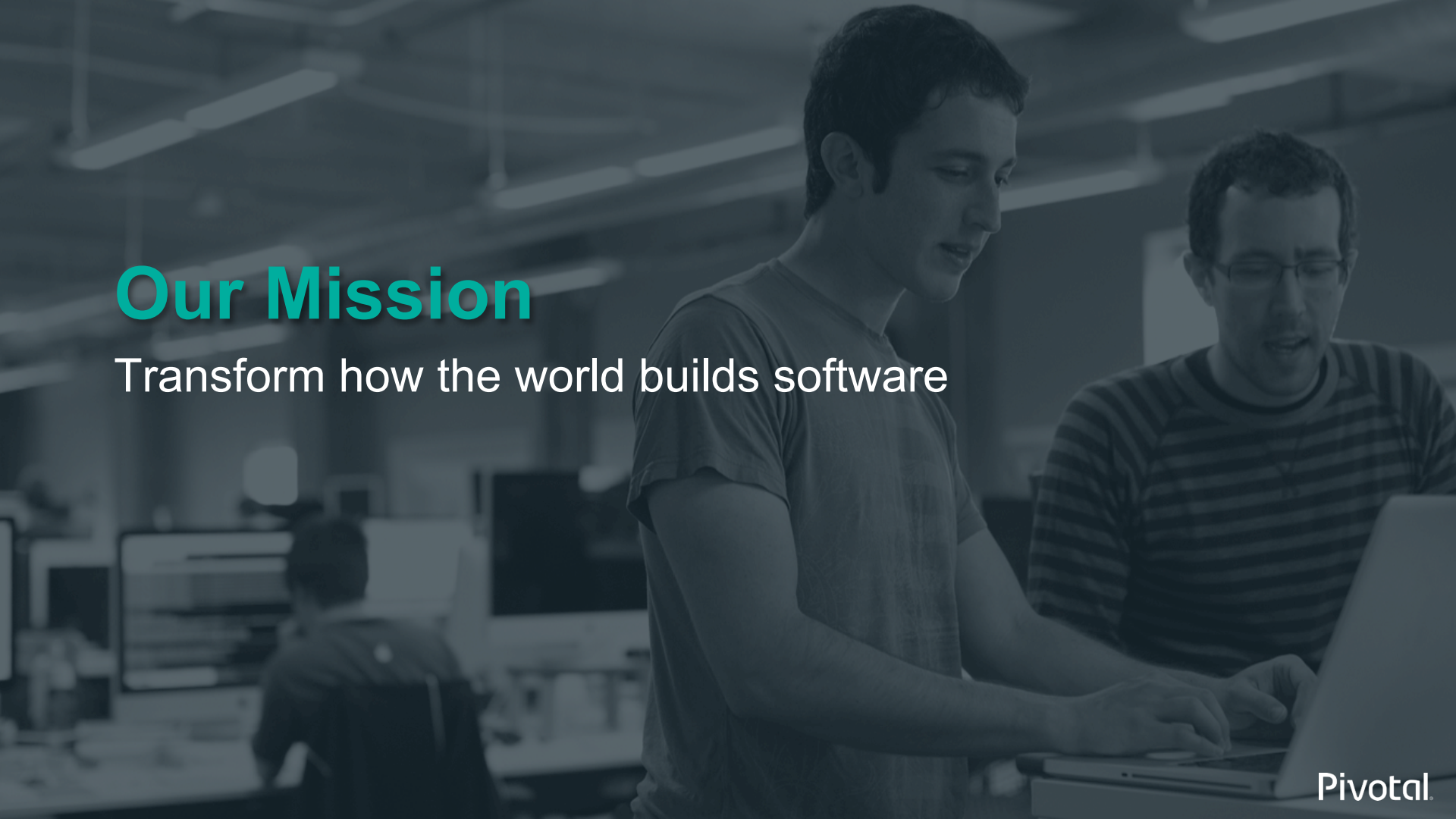
Pivotal

Cloud Native Applications

Development and Operations


Our Mission
Transform how the world builds software
Pivotal

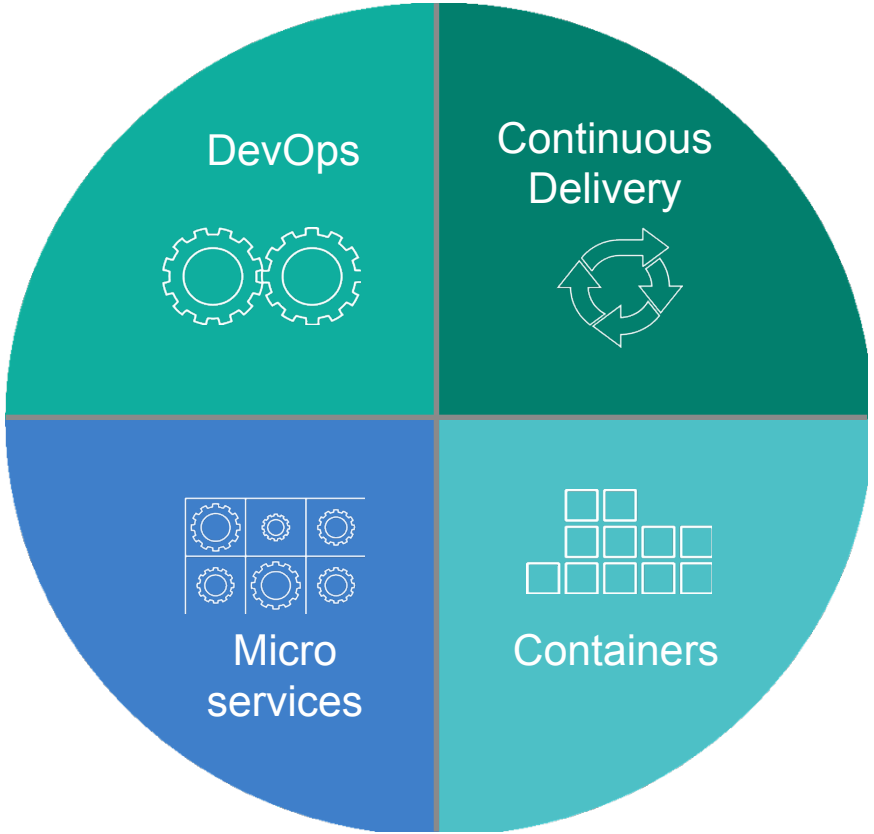
Agenda

- Cloud Native
  - What, Why?
- Cloud Platform
- Cloud Native Apps
  - 12 Factors, 8 Principles, etc.

Pivotal

Cloud Native

- **Cloud Computing:**
  - On demand, network available, distributed, ephemeral
- **Cloud Native:**
  - Applications, platforms, operations that assume a cloud environment.
- **Cloud Native Architecture:**
  - Patterns we implement that take advantage cloud.



Pivotal

Strategic Goals for Cloud Apps

- Speed
  - Innovate, change, deliver value
- Safety
  - Have speed, but do it without breaking anything
- Scalability
  - Apps can be big or small, and their needs will change
  - “right-sizing”
- Ubiquity
  - Anybody anywhere can interact with your services any time with any device
- Simplify Organization

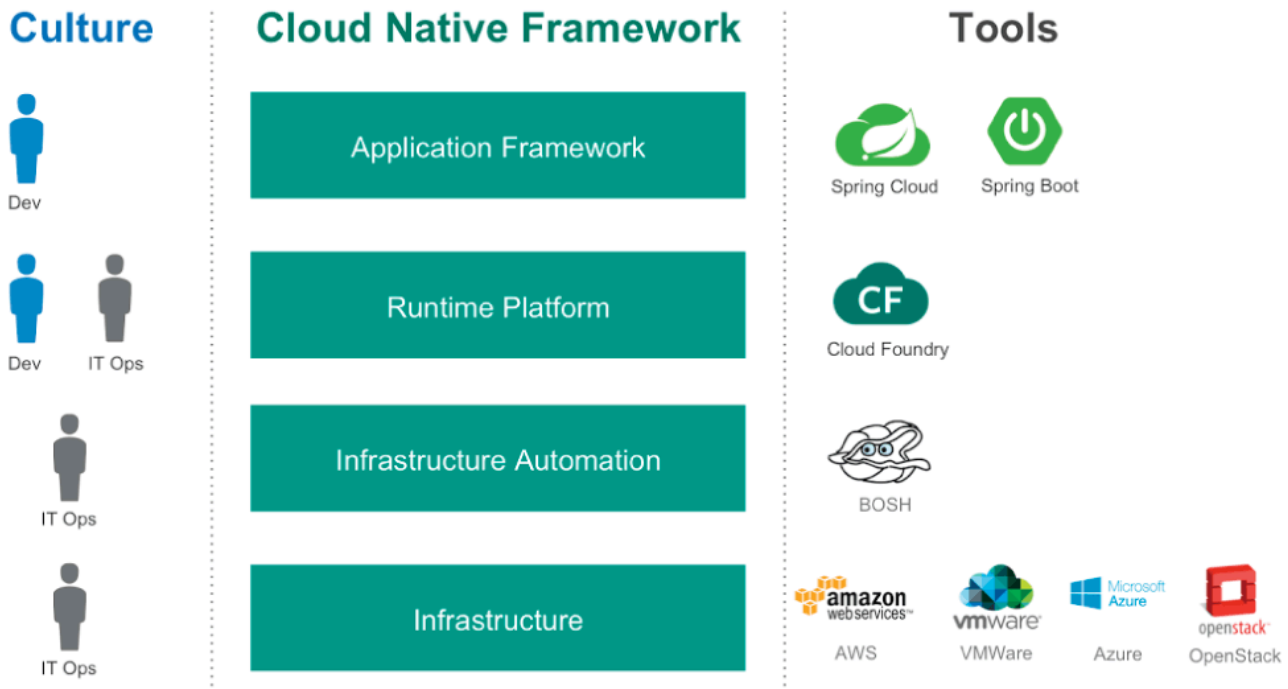
Pivotal

Cross cutting concerns

- The needs and solutions apply to everyone who builds apps
  - They cut across domain/industry
- Undifferentiated functionality has no direct business value
  - Let platforms provide it

Pivotal

Cloud Native Platform



Pivotal


Pivotal Cloud Foundry
Cloud Application Platform (PaaS)

Rapid Innovation Requires a New Approach

DEVELOPERS

- Dramatically improve developer experience
- Agile teams, rapid iteration
- Ingest and incubate open source advancements and new data services

OPERATORS

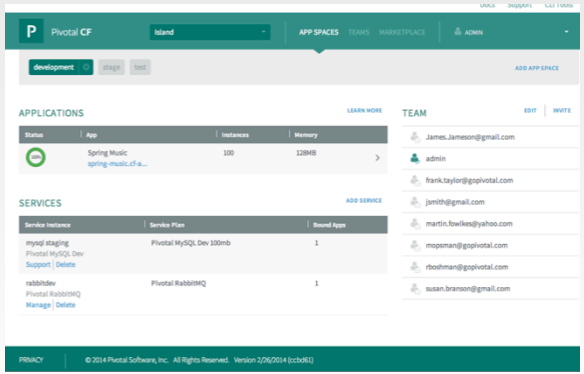
- Continuous delivery, no planned downtime
- Instant scaling of apps and data services
- Automation and deployment consistency at every step

Pivotal

Go From Hours to Minutes

DEVELOPER

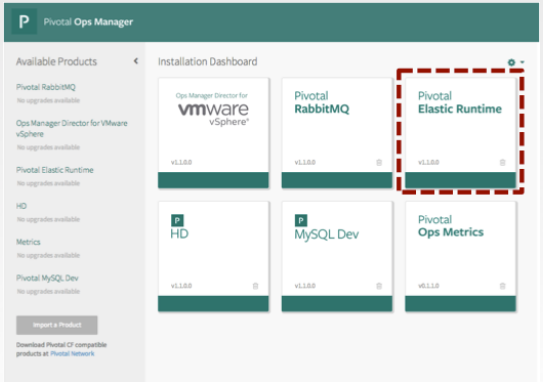
Auto-select runtimes
Deploy app
Select and bind middleware
Scale app



App Deployment: 30-90 seconds

OPERATOR

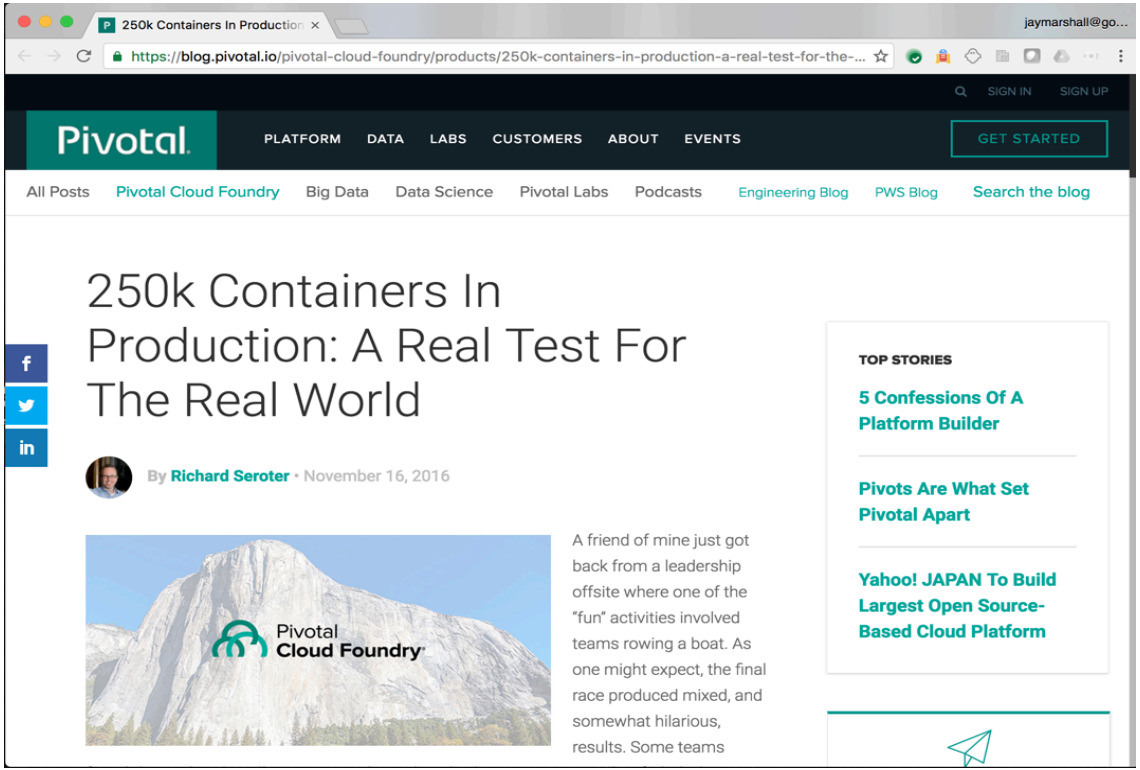
Establish infrastructure
Provision
Add capacity
Control Auth, Policies, HA



Cloud Deployment: 2-4 hours

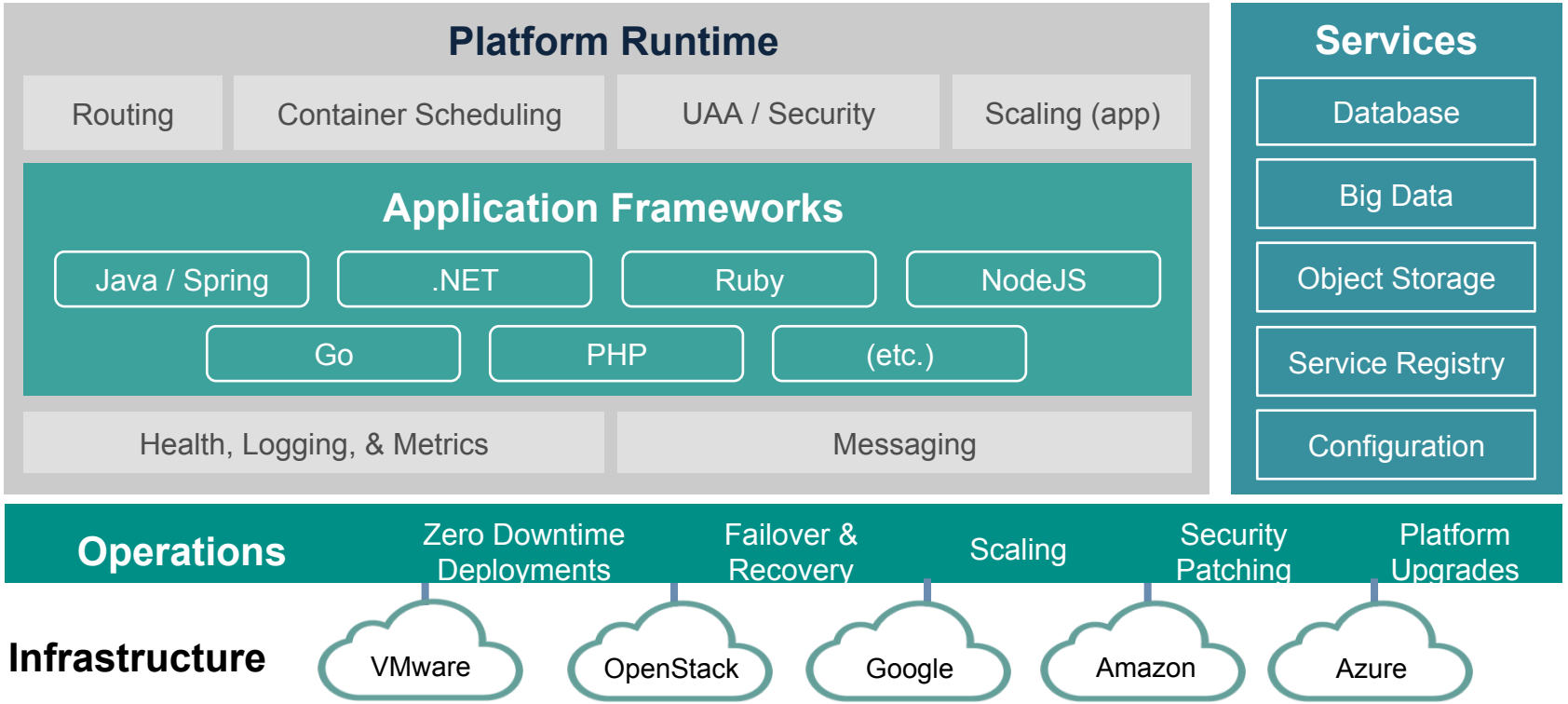
Pivotal

On an Extremely Scalable Platform



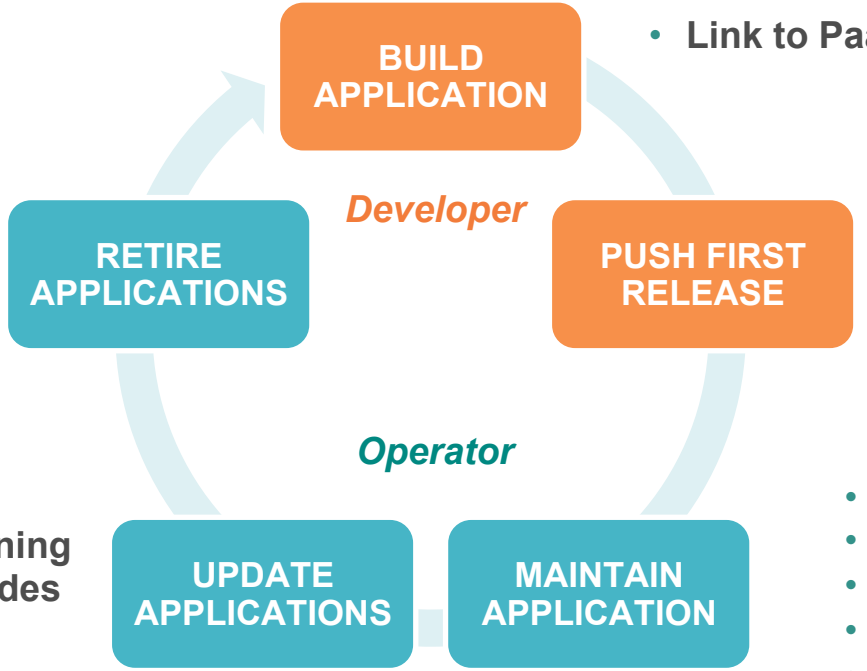
Pivotal

CloudFoundry



Pivotal

Removing Developer and Operator Constraints



- Link to PaaS

- Self-service deploy
- Autowired operational benefits
- Bind to ready Services

- Elastic scale
- Integrated HA
- APM with autoscale
- Log aggregation
- Policy and Auth

- A/B versioning
- Live upgrades

- Self-service removal

Pivotal

PaaS is the Foundation for DevOps

- Dev Ops shifts control of app deployment and maintenance closer to the development team
- It is enabled by automation (fast, repeatable, auditable)
- CFs declarative CLI interface is ideally suited to Dev Ops tools.
  - Heavily reduce need for custom deploy scripts as part of CI/CD
- Create complex upgrade patterns
  - Blue/Green, Canary, Feature toggle
- Buildpacks eliminate the need to create and maintain containers
- CVEs remediated via platform not by building tools

Pivotal

Cloud Platforms Improve Security

- Security is difficult, and the traditional approach of “lockdown and monitor” is an innovation killer.
- They dynamic nature of Pivotal CloudFoundry enables security
- The 3 R’s
  - Rotate credentials
  - Repave environments
  - Repair systems quickly

Go Fast to Stay Safe!

Pivotal

Cloud Native Applications

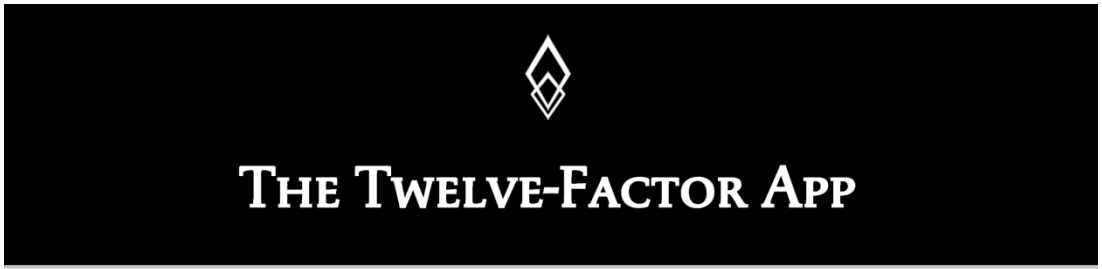
Diving Deeper

What Cloud Native Isn't

- Containers
  - Implementation detail related to the platform
- Maximum decomposition
  - Overdoing anything is generally a bad idea
- Container Scheduling
- Web/Stateless Only

Pivotal

The 12 Factor's



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).

1. One code base
2. Declare dependencies
3. Store config in environment
4. Services as attached resources
5. Build, release, run stages
6. Stateless processes
7. Interface via port binding
8. Scale out via process
9. Disposability
10. Dev/prod parity
11. Logs as event streams
12. Admin via one off processes

Pivotal

Sam Newman's 8 Principles

- Model around Business Domain
- Culture of Automation (you need a platform)
- Hiding implementation details (hide your DBs)
  - Independent evolution
- Decentralize
- Deploy independently
- Consumer first (consumer driven contracts)
- Isolating failure
- Highly Observable

Pivotal

Practices

- Compose applications from small units
  - Easy to change, quick to deploy
- Eliminate dependencies
- Build reliable systems from unreliable components
  - Bulkhead downstream connectors (because failure will happen)
  - Fail fast … slow is broken
- Streamline operations
  - The secret sauce for speed, safety

Pivotal


M!CR0S3RV!C3$!!!!!
IT SEEMS THERE'S SOME HYPE...
http://www.etsy.com/shop/sharpwriter
HEUSER 2012

What are Microservices?

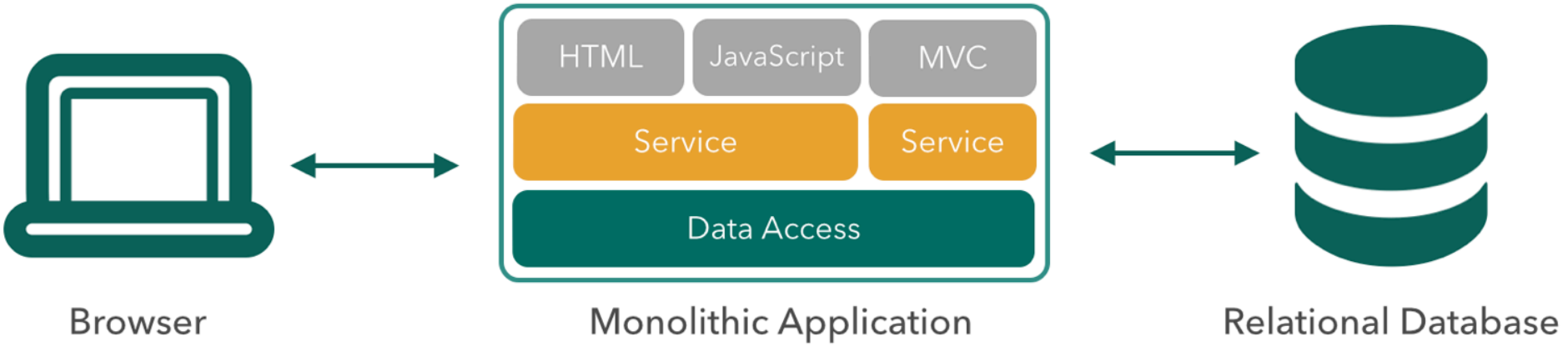
If every service has to be updated in concert, it's not loosely coupled!

Loosely coupled service oriented architecture with bounded contexts

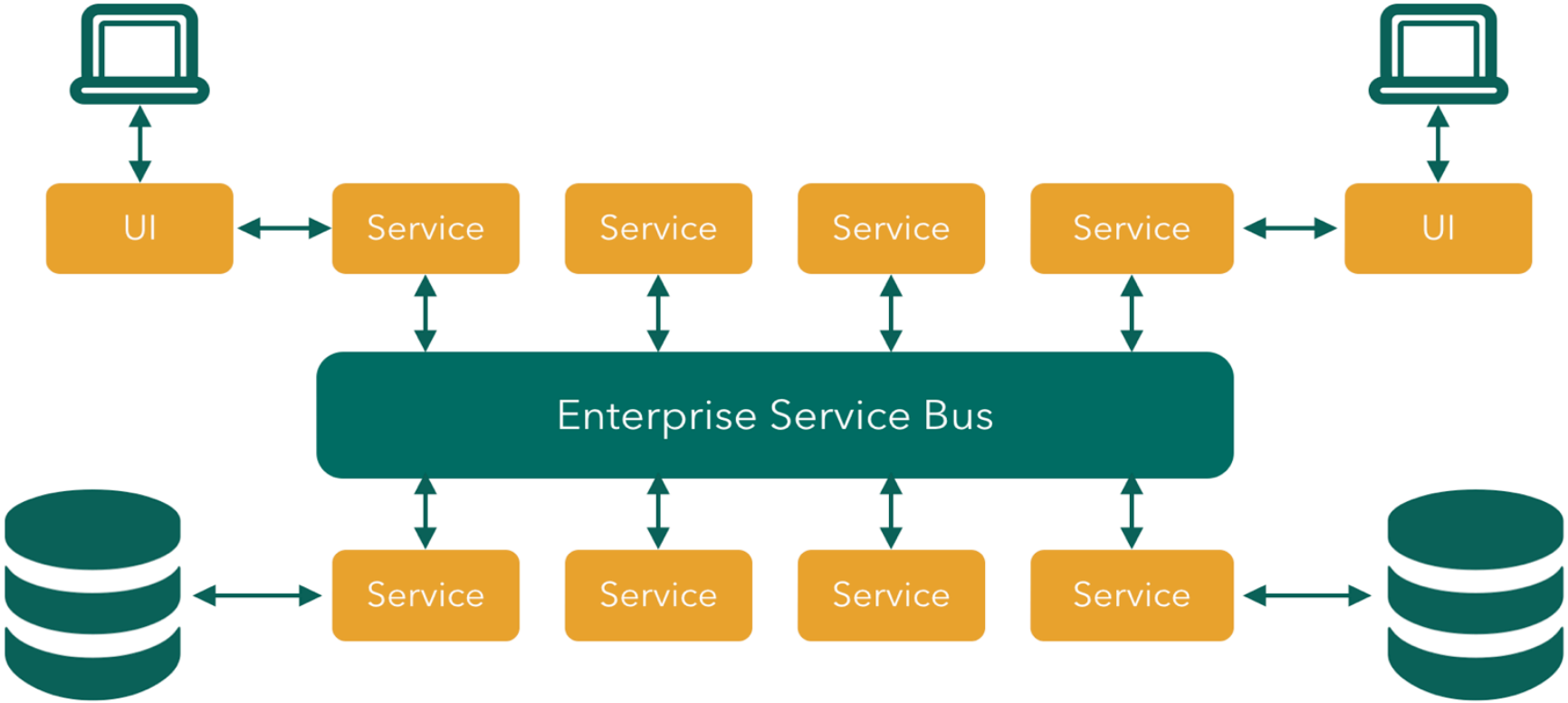
If you have to know about surrounding services you don't have a bounded context.

Pivotal

Not Traditional Architecture



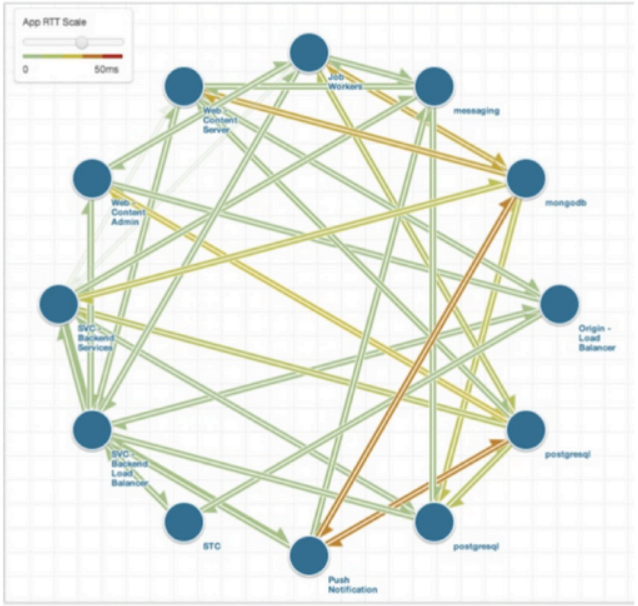
More than ESBs



Pivotal

Microservices!!!

Netflix



Gilt Groupe (12 of 450)

Twitter

Pivotal

Congrats! You have a distributed system

- Individual microservices are relatively simple
  - BUT, they interact with other services and app components
  - Complexity transferred to the ecosystem
  - It's probably even more complex ☹
- Challenges
  - Configuration Management
  - Registration and Discovery
  - Routing and Load Balancing
  - Monitoring/Debugging
  - Aggregation/Transformation
  - System consistency
  - Contract breaking changes
  - Security!!
  - Poorly behaved clients (which is all of them)

Pivotal

What Do We Gain?

- Independently
  - Deployable
  - Upgradable
  - Replaceable
  - Scalable
- Teams focused on a well defined component
  - In charge of their own destiny
  - Do what's right for them
- Technology independance

Pivotal

Defining the Bounded Context

- A business concept that exists in my app
  - —Domain driven design
- Services implement a contract
- Services use models matching their local view of the domain
  - Where models diverge is a good clue to a boundary
- This is where domain knowledge has the biggest impact
  - Spend your resources here

Pivotal

Application Journey

- Three primary levels for apps on a platform
  - Runs
  - Runs well
  - Cloud Native
- Brown Field
  - Apps with little to no active development
  - Won’t run on a cloud platform
  - Rewrite is usual remedy

Pivotal

Common Monolith to Microservice Goals

- Migrate monolith that is in production, making money
  - Want more sustainable solution for the future
- Architect for scalability
- Allow multiple teams to deliver business value quickly
- Allow for addition of resources without overcrowding

Sound familiar?

Pivotal

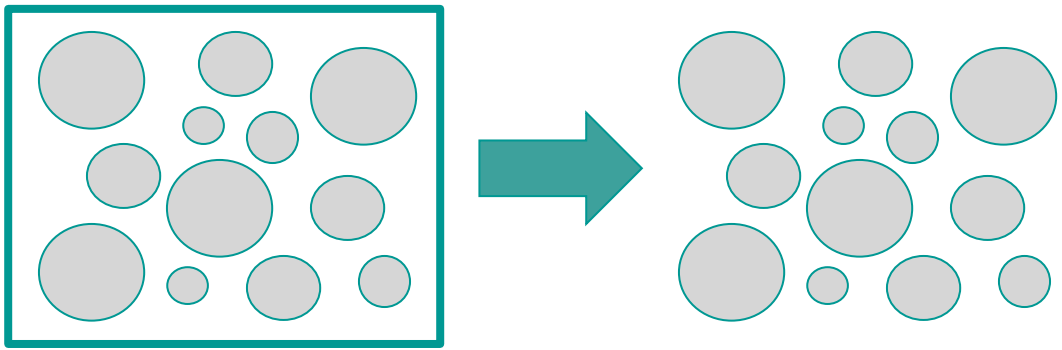
From Monolith to Microservice Nirvana

- It’s easy to get decomposition wrong
- TDD: be able to validate your app is doing what you want
- The “Well Structured” monolith
  - Use the monolith to experiment with domain boundaries.
  - Evolve it to be friendly to distributed function
- Break out components and new extensions with
- Continue strangling

Pivotal

Well Structured Monolith

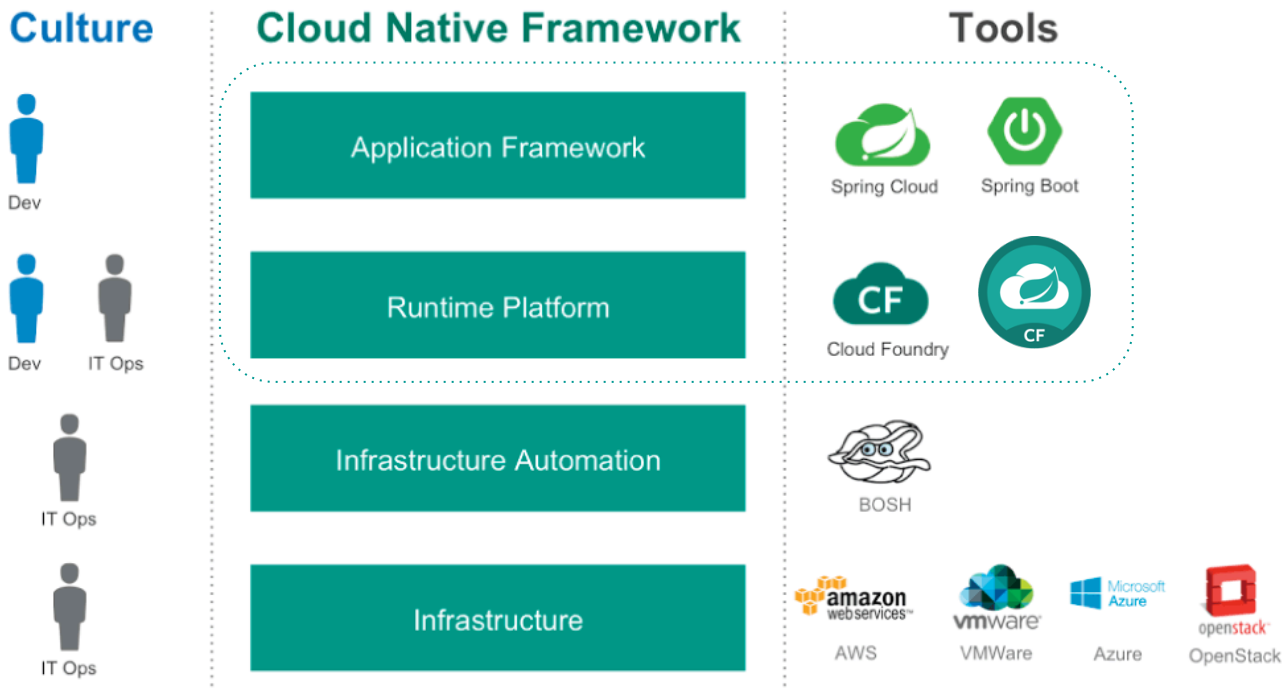
- Modularized on Business Capability (Bounded Context)
- High Cohesion
- Low Coupling
- Encapsulated Data
- Substitutable
- Composable
- Decomposable



Pivotal


Cloud Native Architecture
Patterns for Cloud Native with examples in Java/Spring

Cloud Native Platform



Pivotal

More

- Demo code
  - https://github.com/sdeeg-pivotal/cna-demo-greeting
- @Pivotal
  - http://pivotal.io/cloud-native
- Spring Cloud
  - http://projects.spring.io/spring-cloud/
- Josh Long Video
  - https://www.youtube.com/watch?v=SFDYdslOvu8

Pivotal

Pivotal

Open.
Agile.
Cloud-Ready.