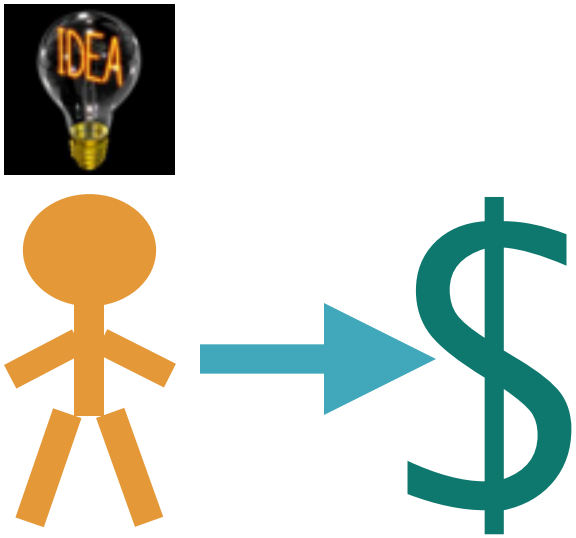
Architecting for Continuous Delivery

Cloud Foundry and Microservices

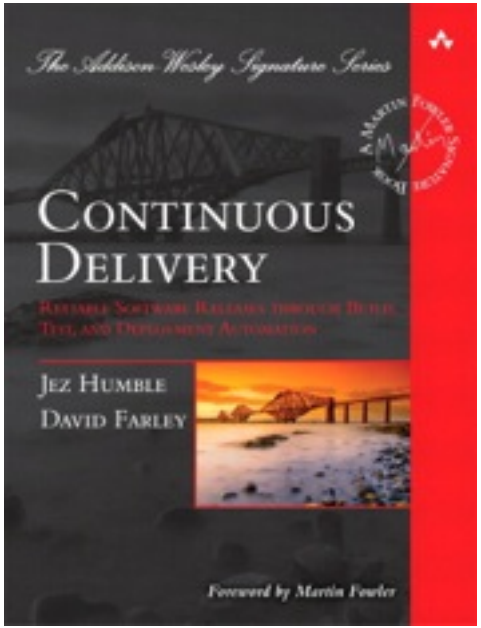
Pivotal

What is Continuous Delivery?

What is Continuous Delivery?



Continuous Delivery - How?



Pivotal

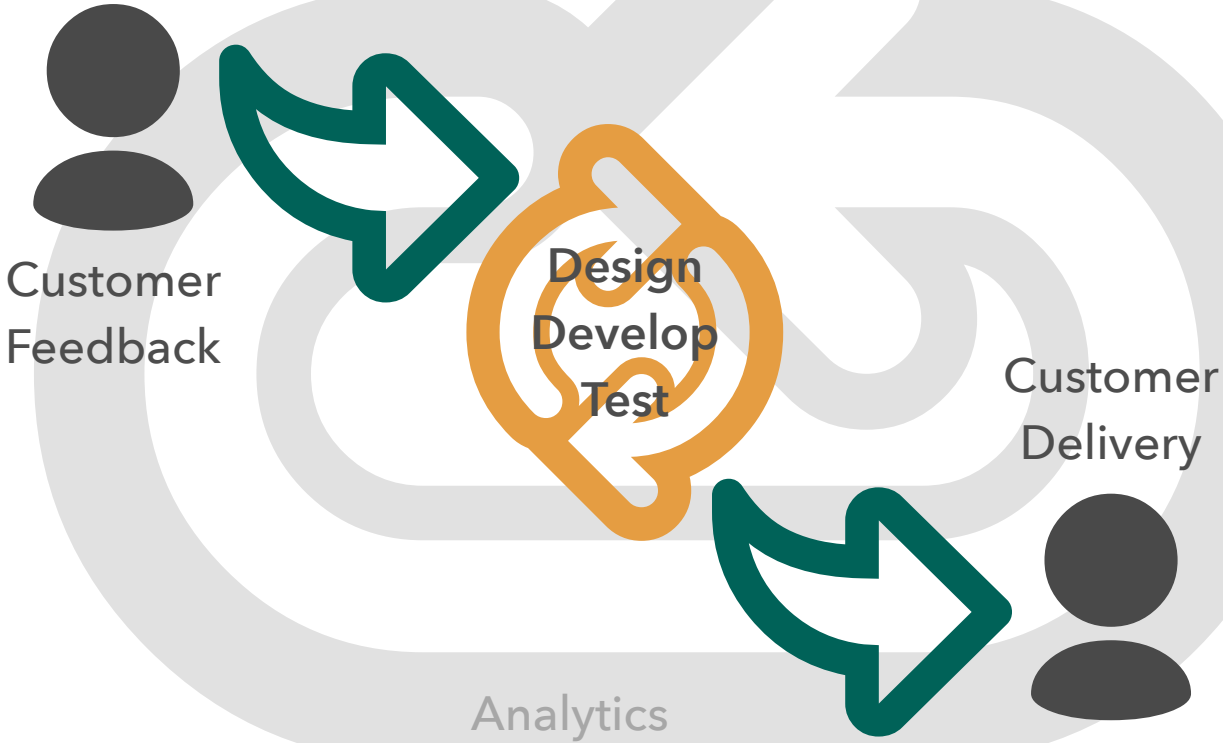
Warner Music: Software Factories

Warner Software Factory Platform

- New applications and major updates
  - **Before**: 6 months, team of 10 developers
  - **After**: 6 weeks, same team
  - **Speed/Agility**: 400% faster on new platform
  - **HR Hard Savings:** $1.1M per application update delivered

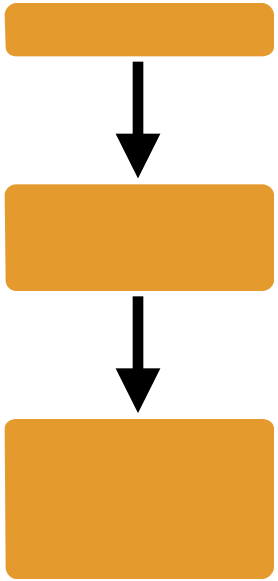
Pivotal

Iterative Development

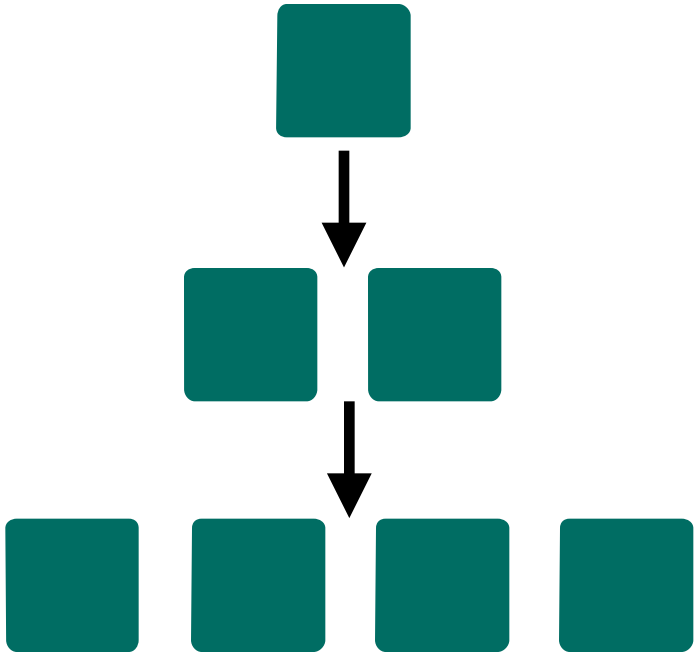


Pivotal

Horizontal Scale



Slow/Expensive



Fast/Cheap

Pivotal

Diversity of Clients

In January 2014, mobile devices accounted for 55% of Internet usage in the United States. Apps made up 47% of Internet traffic and 8% of traffic came from mobile browsers.

http://money.cnn.com/2014/02/28/technology/mobile/mobile-apps-internet/

Pivotal

Continuous Delivery

Software Factories

Feedback → Rapid Iteration

Horizontal Scale

Diversity of Clients

Physical/Virtual

Cloud Foundry

Infrastructure

Applications

Monoliths

Microservices

Pivotal

Continuous Delivery

Software Factories

Feedback → Rapid Iteration

Horizontal Scale

Diversity of Clients

Physical/Virtual

Cloud Foundry

Infrastructure

Applications

Monoliths

Microservices

Pivotal

New Architectural Constraints

- CF optimizes for 12 Factor Linux applications
- Microservices: a radical departure from traditional monolithic applications
- In both cases, the enterprise is forced to “think different.”

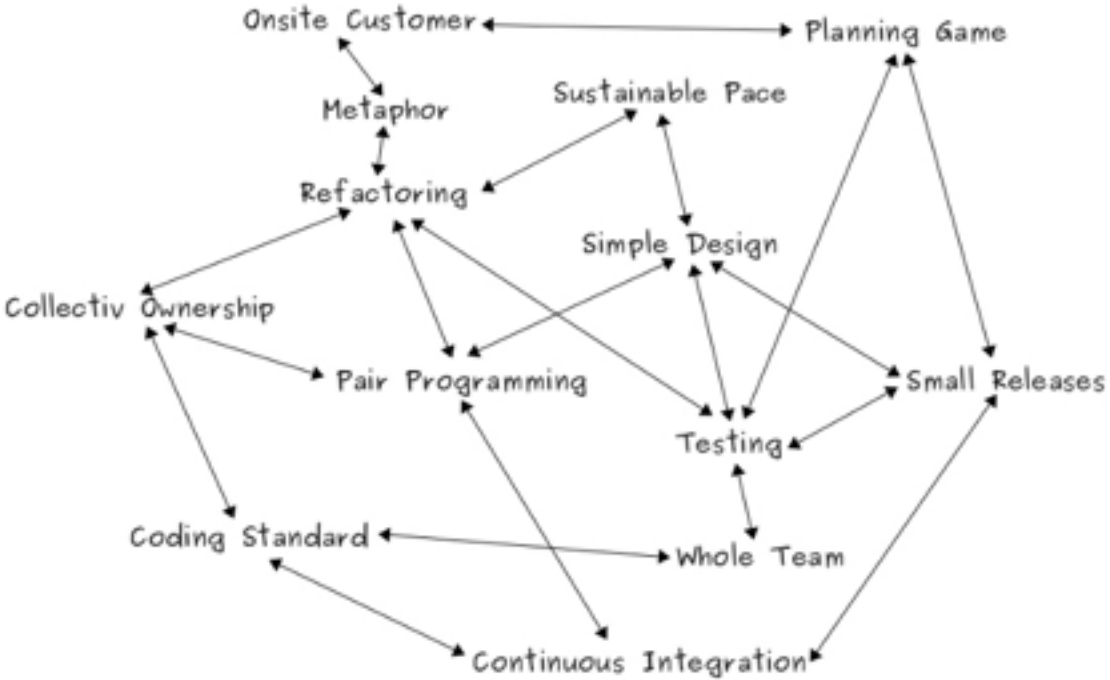
Pivotal

New Architectural Constraints

- CF optimizes for 12 Factor Linux applications
- Microservices: a radical departure from traditional monolithic applications
- In both cases, the enterprise is forced to “think different.”

Pivotal

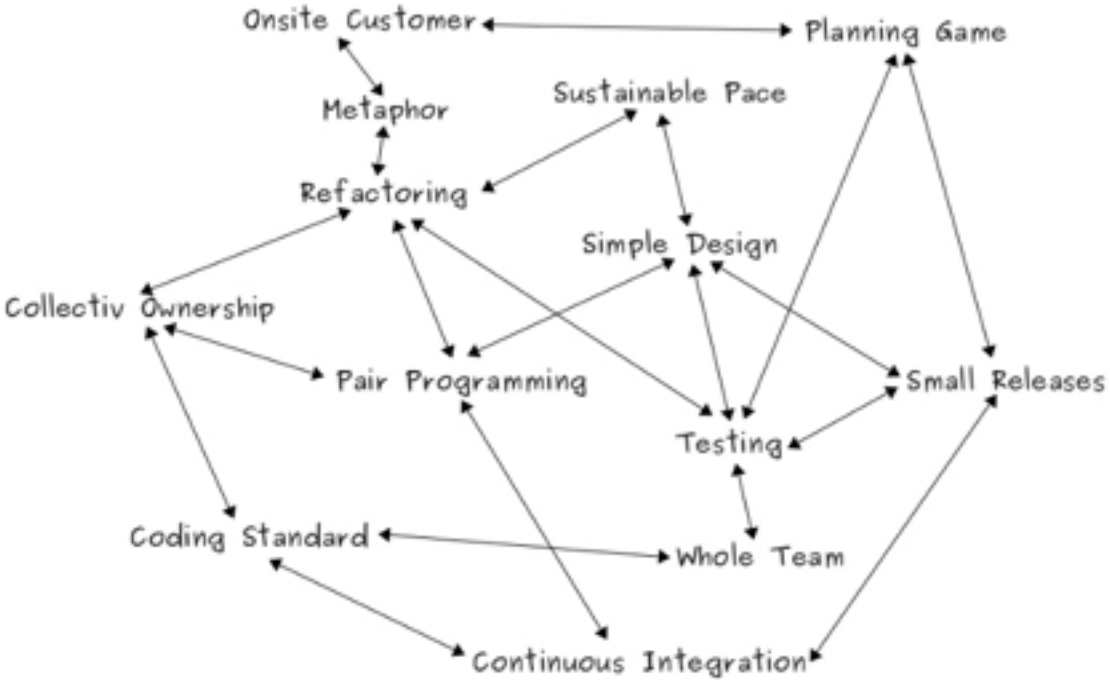
How XP Practices Support Each Other



http://www.kusmin.eu/wiki/index.php/How_XP_Practices_Support_Each_Other

Pivotal

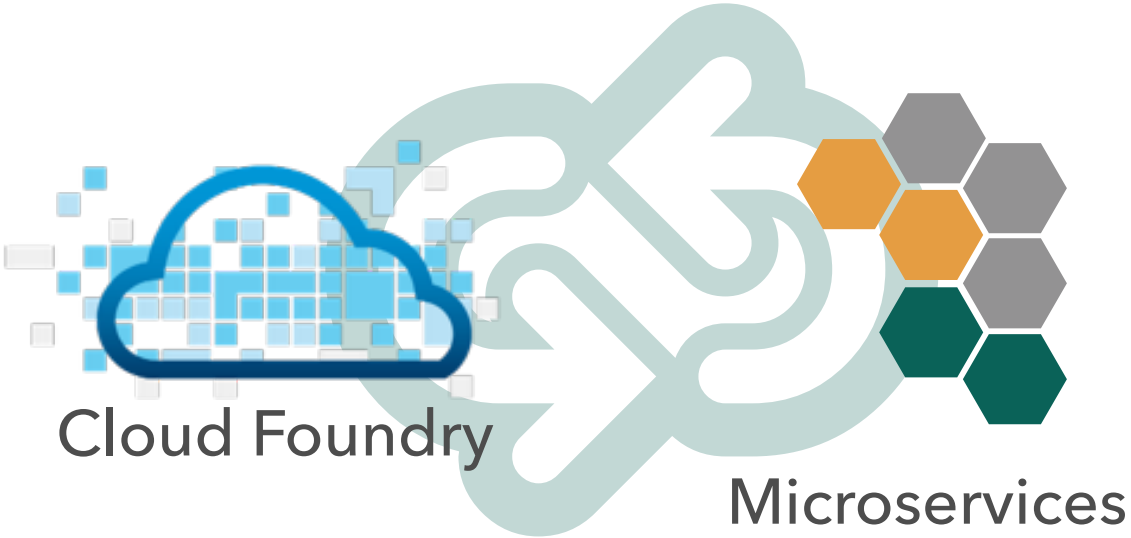
How XP Practices Support Each Other



http://www.kusmin.eu/wiki/index.php/How_XP_Practices_Support_Each_Other

Pivotal

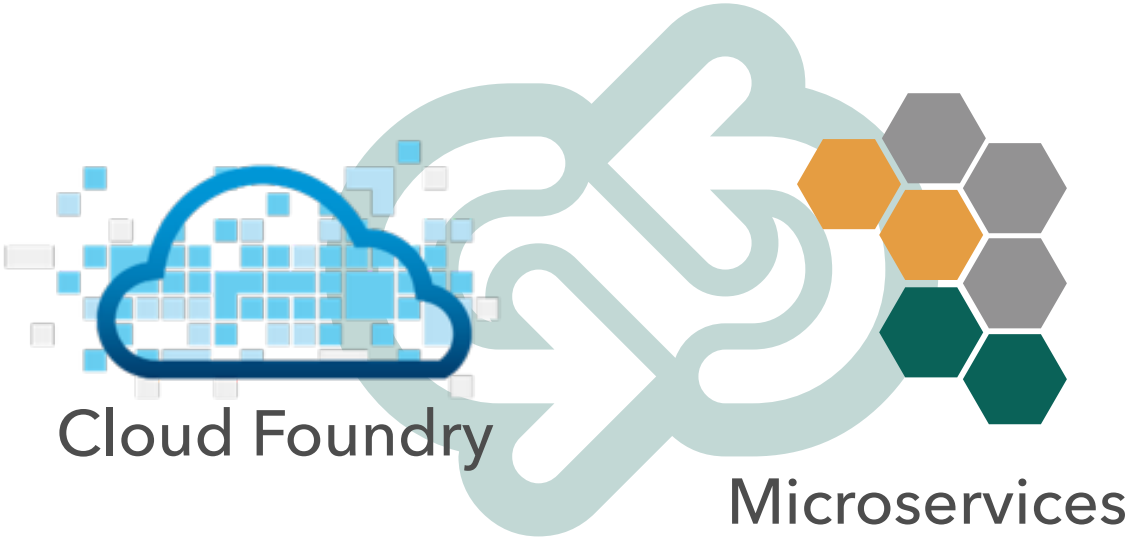
A Mutualistic Symbiotic Relationship…



http://en.wikipedia.org/wiki/Symbiosis#Mutualism

Pivotal

A Mutualistic Symbiotic Relationship…



http://en.wikipedia.org/wiki/Symbiosis#Mutualism

Pivotal

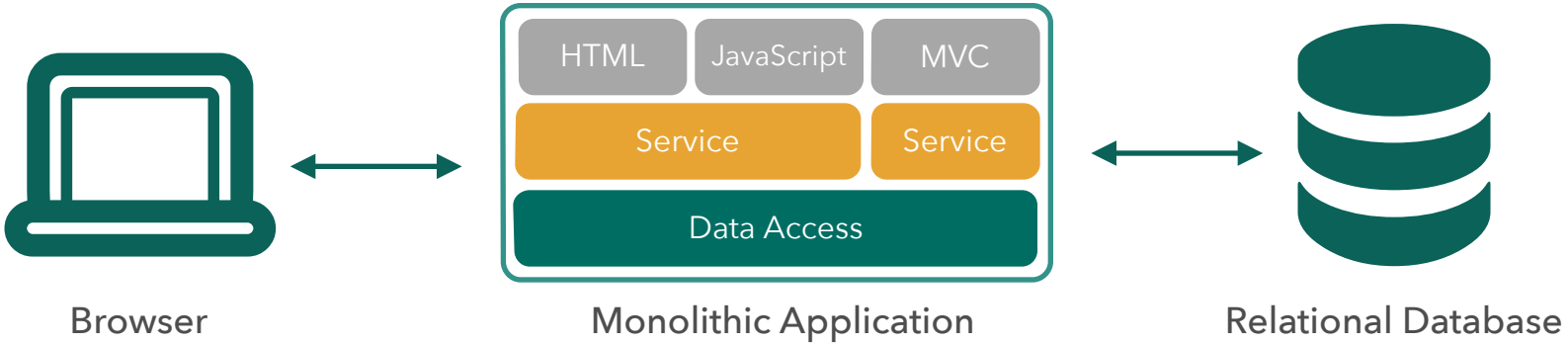
Microservices Overview

Simple vs. Easy

- **Simple**
  - *sim- plex*
  - one fold/braid
  - vs complex
- **Easy**
  - *ease < aise < adjacens*
  - lie near
  - vs hard

Pivotal

Monolithic Architecture

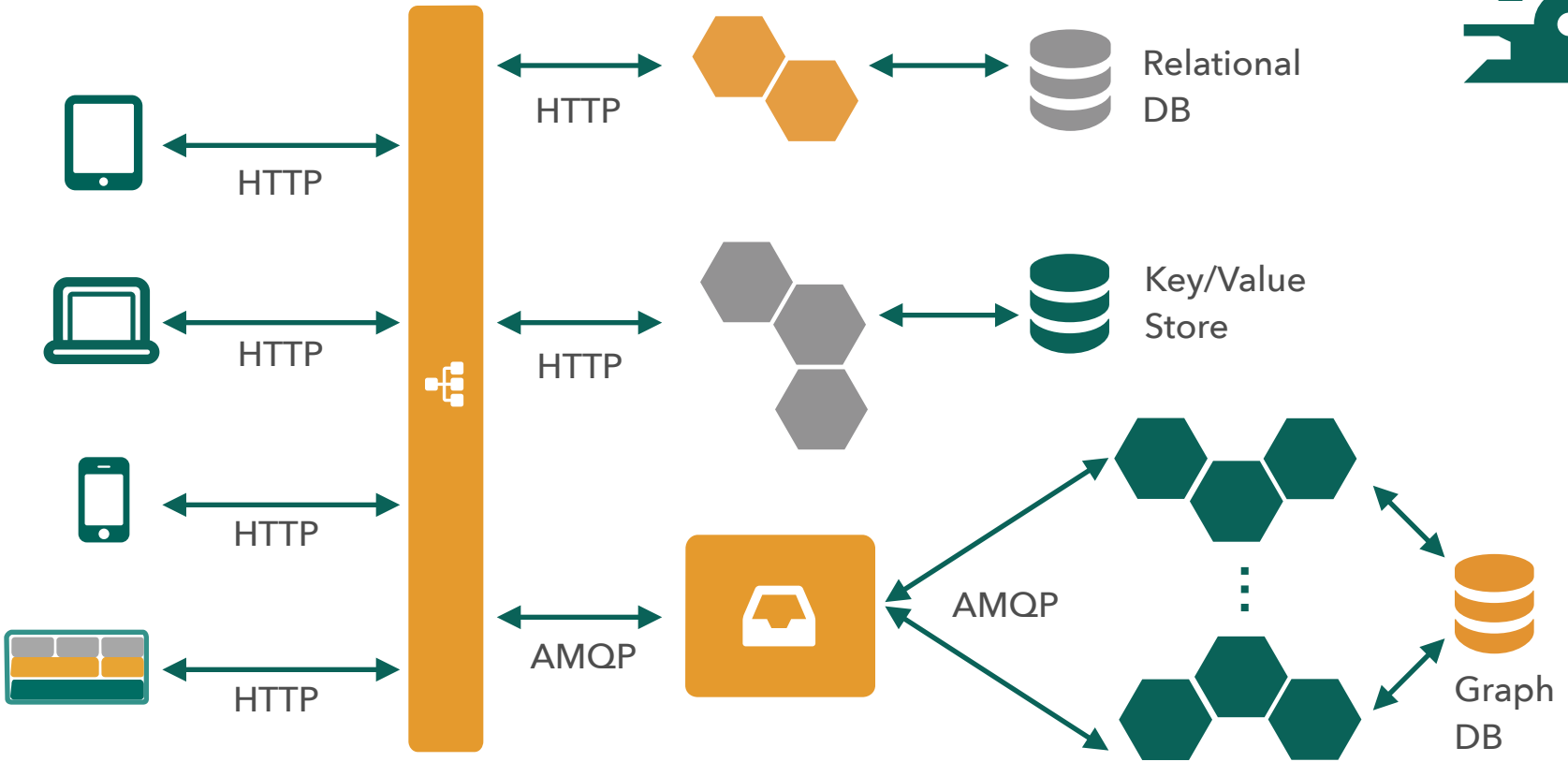


Monolithic Architectures

- Complex / Easy
- Modularity Dependent Upon Language / Frameworks
- Change Cycles Tightly Coupled / Obstacle to Frequent Deploys
- Inefficient Scaling
- Can Be Intimidating to New Developers
- Obstacle to Scaling Development
- Requires Long-Term Commitment to Technical Stack

Pivotal

Microservice Architecture



Pivotal

Microservice Architectures

- Simple / Hard
- Modularity Based on Component Services
- Change Cycles Decoupled / Enable Frequent Deploys
- Efficient Scaling
- Individual Components Less Intimidating to New Developers
- Enables Scaling of Development
- Eliminates Long-Term Commitment to Technical Stack

Pivotal

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

http://martinfowler.com/articles/microservices.html#OrganizedAroundBusinessCapabilities

Pivotal

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

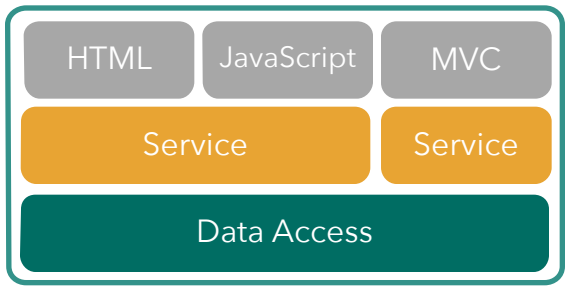
http://martinfowler.com/articles/microservices.html#OrganizedAroundBusinessCapabilities

Pivotal

Organize Around Business Capabilities

Siloed Functional Teams

Siloed Application Architectures



Cross-functional Teams

Microservice Architectures

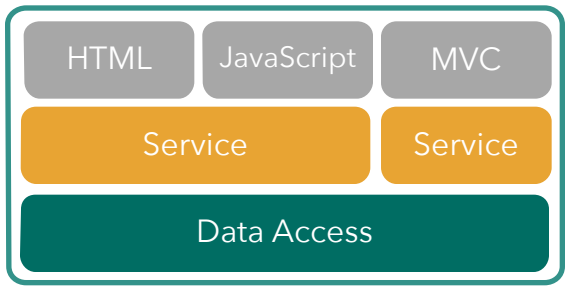
http://martinfowler.com/articles/microservices.html#OrganizedAroundBusinessCapabilities

Pivotal

Organize Around Business Capabilities

Siloed Functional Teams

Siloed Application Architectures



Cross-functional Teams

Microservice Architectures

http://martinfowler.com/articles/microservices.html#OrganizedAroundBusinessCapabilities

Pivotal

Partitioning Strategies

- By Noun (e.g. product info service)
- By Verb (e.g. shipping service)
- Single Responsibility Principle (http://programmer.97things.oreilly.com/wiki/index.php/The_Single_Responsibility_Principle)
- http://www.mattstine.com/2014/06/30/microservices-are-solid/

Pivotal

UNIX Pipes and Filters

```
cut -d" " -f1 < access.log | sort | uniq -c | sort -rn | less
```

Pivotal

Choreography over Orchestration

http://martinfowler.com/articles/microservices.html#SmartEndpointsAndDumbPipes

Pivotal

Choreography over Orchestration

http://martinfowler.com/articles/microservices.html#SmartEndpointsAndDumbPipes

Pivotal

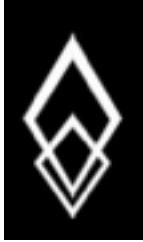
The Twelve-Factor App

Overview

http://12factor.net

Pivotal

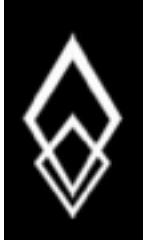
Twelve Factor + Cloud Foundry



- One codebase tracked in revision control, many deploys
  - Multiple = Distributed System
  - Consistent with CF application unit
- Explicitly declare and isolate dependencies
  - CF emphasis on deployable units (e.g. Java WAR)
  - CF Buildpacks provide runtime dependencies

Pivotal

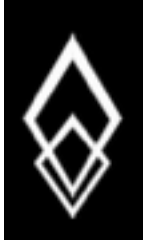
Twelve Factor + Cloud Foundry



- Store config in the environment
  - Nicely facilitated via CF

Pivotal

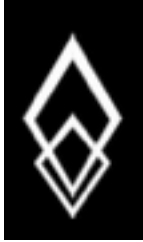
Twelve Factor + Cloud Foundry



- Store config in the environment
  - Nicely facilitated via CF
- Treat backing services as attached resources
  - cf create-service / cf bind-service

Pivotal

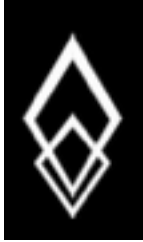
Twelve Factor + Cloud Foundry



- Store config in the environment
  - Nicely facilitated via CF
- Treat backing services as attached resources
  - cf create-service / cf bind-service
- Strictly separate build and run stages
  - CF Buildpacks + immutable Warden containers

Pivotal

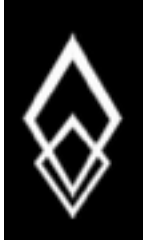
Twelve Factor + Cloud Foundry



- Execute the app as one or more stateless processes
  - CF Warden containers - no app server clustering, no shared FS.
  - Challenge for the monolith!

Pivotal

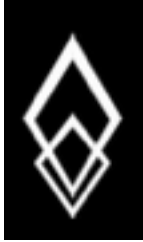
Twelve Factor + Cloud Foundry



- Execute the app as one or more stateless processes
  - CF Warden containers - no clustered memory, no shared FS.
  - Challenge for the monolith!
- Export services via port binding
  - CF provides HTTP/HTTPS today, more future (TCP?)

Pivotal

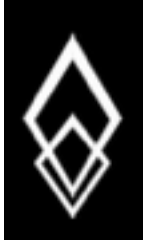
Twelve Factor + Cloud Foundry



- Execute the app as one or more stateless processes
  - CF Warden containers - no clustered memory, no shared FS.
  - Challenge for the monolith!
- Export services via port binding
  - CF provides HTTP/HTTPS today, more future (TCP?)
- Scale out via the process model
  - cf scale app -i 1000

Pivotal

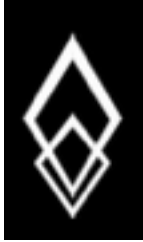
Twelve Factor + Cloud Foundry



- Maximize robustness with fast startup and graceful shutdown
  - CF scales quickly, but can only move as fast as your app can bootstrap (challenge for the monolith!

Pivotal

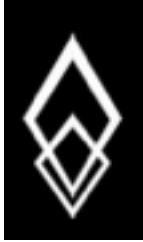
Twelve Factor + Cloud Foundry



- Maximize robustness with fast startup and graceful shutdown
  - CF scales quickly, but can only move as fast as your app can bootstrap (challenge for the monolith!)
- Keep development, staging, and production as similar as possible
  - CF is CF! Spaces provide separation of concerns without technical differences.

Pivotal

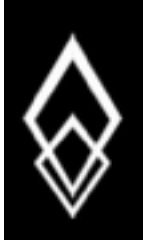
Twelve Factor + Cloud Foundry



- Treat logs as event streams
  - CF Loggregator!
- Run admin/management tasks as one-off processes
  - Still a challenge to be addressed…

Pivotal

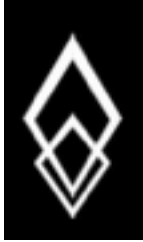
Twelve Factor + Microservices



- Fully compatible architectural style
- Frameworks tend to optimize around same ideas
- Examples:
  - Spring Boot + Cloud
    - http://projects.spring.io/spring-boot
    - http://projects.spring.io/spring-cloud
  - Dropwizard (https://dropwizard.github.io/dropwizard)

Pivotal

Twelve Factor + Microservices



- Fully compatible architectural style
- Frameworks tend to optimize around same ideas
- Examples:
  - Spring Boot + Cloud
    - http://projects.spring.io/spring-boot
    - http://projects.spring.io/spring-cloud
  - Dropwizard (https://dropwizard.github.io/dropwizard)

Pivotal

High Scalability

Building bigger, faster, more reliable websites.

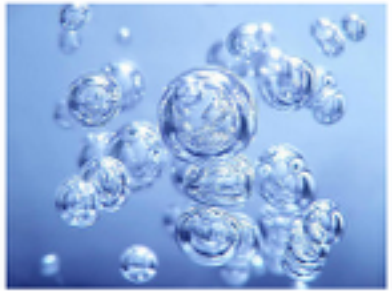
Home Real Life Architectures Strategies All Posts Advertising Book Store Start Here Contact
All Time Favorites RSS Twitter Facebook G+

« Paper: Scalable Atomic Visibility with RAMP Transactions - Scale Linearly to 100 Servers | Main | Google Finds: Centralized Control, Distributed Data Architectures Work Better than Fully Decentralized Architectures »

Microservices - Not A Free Lunch!

TUESDAY, APRIL 8, 2014 AT 8:54AM

This is a guest post by Benjamin Wootton, CTO of Contino, a London based consultancy specialising in applying DevOps and Continuous Delivery to software delivery projects.



Microservices are a style of software architecture that involves delivering systems as a set of very small, granular, independent collaborating services.

http://highscalability.com/blog/2014/4/8/microservices-not-a-free-lunch.html

Pivotal

Paying for your lunch…

- Significant Operations Overhead
- Substantial DevOps Skills Required
- Implicit Interfaces
- Duplication of Effort
- Distributed System Complexity
- Asynchronicity is Difficult!
- Testability Challenges

Pivotal

Paying for your lunch…

- **Significant Operations Overhead**
- **Substantial DevOps Skills Required**
- Implicit Interfaces
- Duplication of Effort
- **Distributed System Complexity**
- Asynchronicity is Difficult!
- **Testability Challenges**

Pivotal

Significant Operations Overhead

- Mitigate polyglot language/environment provisioning complexity via CF Buildpacks
- Mitigate failover and resilience concerns via CF Scale, CF Health Monitor, and future CF App AZ's (http://blog.gopivotal.com/cloud-foundry-pivotal/products/the-four-levels-of-ha-in-pivotal-cf)

Pivotal

Significant Operations Overhead

- Mitigate polyglot language/environment provisioning complexity via CF Buildpacks
- Mitigate failover and resilience concerns via CF Scale, CF Health Monitor, and future CF App AZ's (http://blog.gopivotal.com/cloud-foundry-pivotal/products/the-four-levels-of-ha-in-pivotal-cf)
- Mitigate routing/load balancing and plumbing concerns via CF Router and CF Services
- High quality monitoring = CF BP agent-based tooling, future CF metric streams
- High quality operations infrastructure = CF BOSH!

Pivotal

Significant Operations Overhead

- Mitigate polyglot language/environment provisioning complexity via CF Buildpacks
- Mitigate failover and resilience concerns via CF Scale, CF Health Monitor, and future CF App AZ's (http://blog.gopivotal.com/cloud-foundry-pivotal/products/the-four-levels-of-ha-in-pivotal-cf)
- Mitigate routing/load balancing and plumbing concerns via CF Router and CF Services
- High quality monitoring = CF BP agent-based tooling, future CF metric streams
- High quality operations infrastructure = CF BOSH!
- Robust release/deployment automation = CF API, scriptable CF CLI, Maven/Gradle Plugins, Strong Cloudbees/Jenkins partnerships

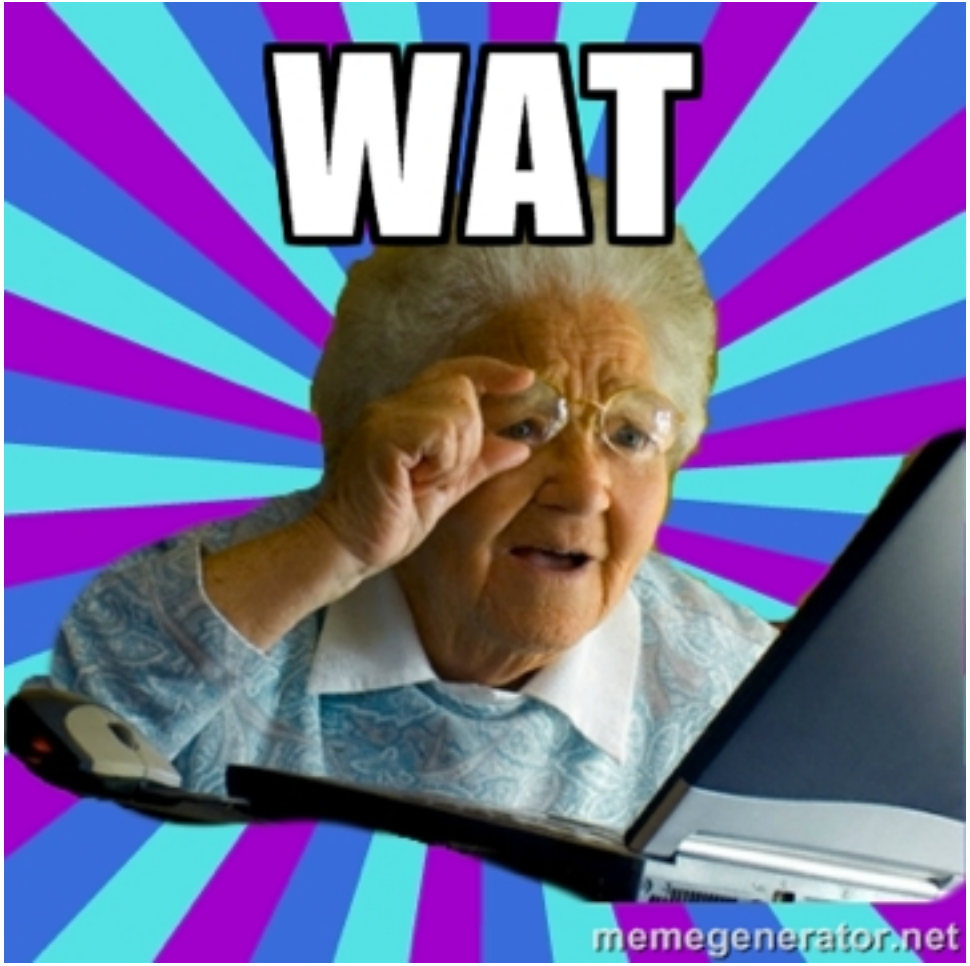
Pivotal

*Currently, **there is not much in terms of frameworks and open source tooling** to support this from an operational perspective. It's likely therefore that a team rolling out Microservices will need to make **significant investment** in custom scripting or development to manage these processes **before they write a line of code that delivers business value.***

Operations is the most obvious and commonly held objection towards the model, though it is too easily brushed aside by proponents of this architecture.

http://highscalability.com/blog/2014/4/8/microservices-not-a-free-lunch.html

Pivotal



No Open Source Tooling?

Pivotal

Substantial DevOps Skills Required

- This is a **Good Thing™** in any architecture!
- CF keeps your microservices up and available (and your monoliths too!)
- CF = development and production parity!
- Polyglot persistence without all the fuss: CF BOSH and Service Brokers

Pivotal

Substantial DevOps Skills Required

- This is a **Good Thing™** in any architecture!
- CF keeps your microservices up and available (and your monoliths too!)
- CF = development and production parity!
- Polyglot persistence without all the fuss: CF BOSH and Service Brokers

Pivotal

Distributed System Complexity

- Agreed: Microservices imply distributed systems.
- All of the CF platform features we've discussed help to mitigate these concerns:
  - latent/unreliable networks
  - fault tolerance
  - load variability

Pivotal

Testability Challenges

- With CF, it is ***NOT*** difficult to recreate environments in a consistent way for either manual or automated testing!
- Idiomatic Microservices involves placing less emphasis on testing and more on monitoring
  - Not sure where this idea comes from…
  - CF is an enabler of both!

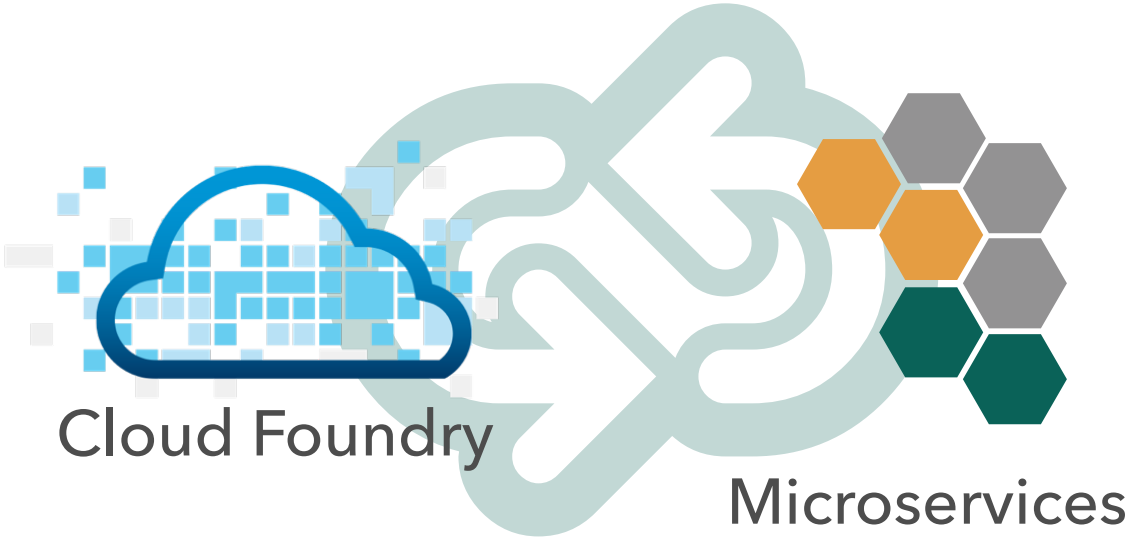
Pivotal

Testability Challenges

- With CF, it is ***NOT*** difficult to recreate environments in a consistent way for either manual or automated testing!
- Idiomatic Microservices involves placing less emphasis on testing and more on monitoring
  - Not sure where this idea comes from…
  - CF is an enabler of both!

Pivotal

A Mutualistic Symbiotic Relationship



http://en.wikipedia.org/wiki/Symbiosis#Mutualism

Pivotal

A NEW PLATFORM FOR A NEW ERA