Cloud Native Application

Cloud Foundry and Microservices

Derrick Wong

Pivotal

Microservices Overview

Pivotal

DEFINE: Microservice

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

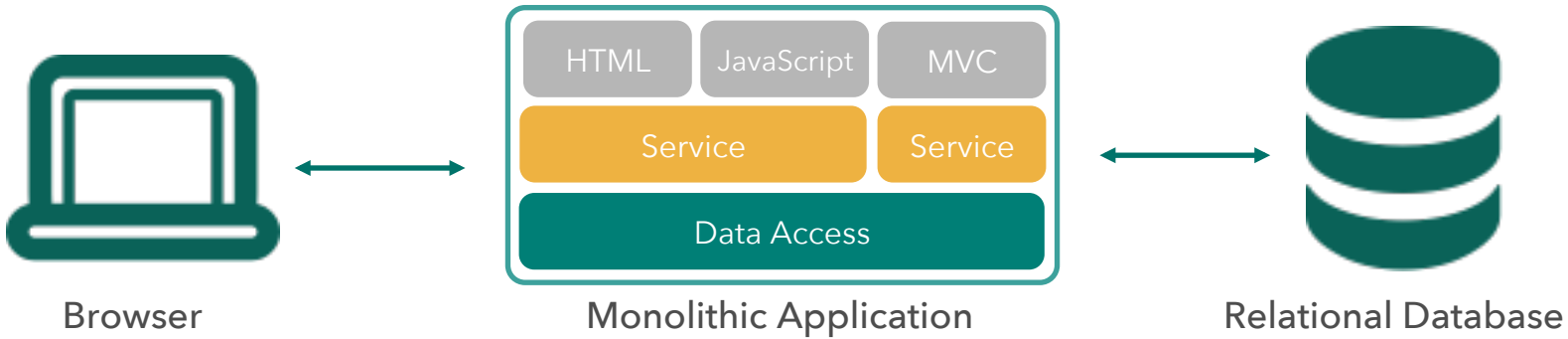
Loosely coupled service oriented architecture with bounded contexts

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

Adrian Cockcroft, Former Netflix Chief Cloud Architect

Pivotal

Monolithic Architecture

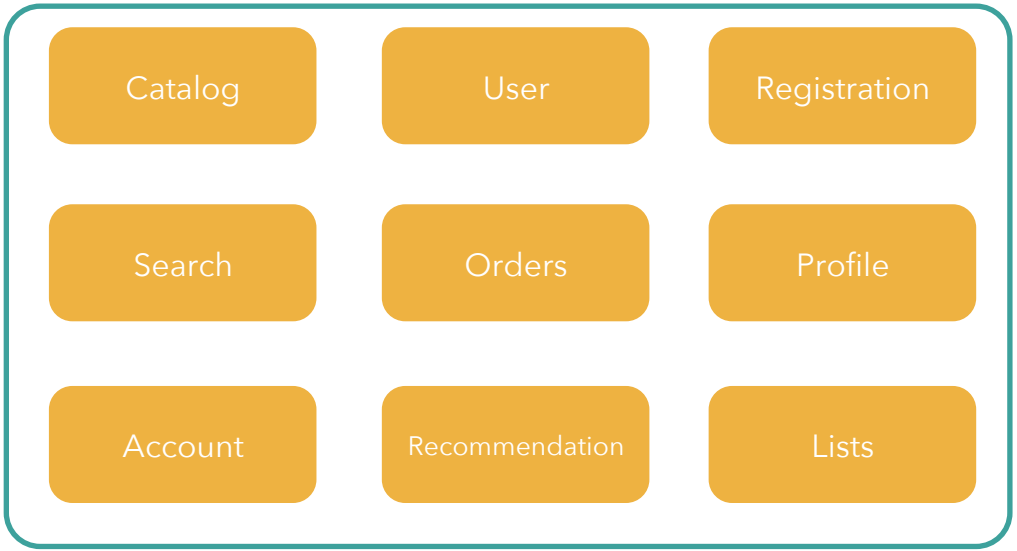


Monolithic Architectures

- Complex
- 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

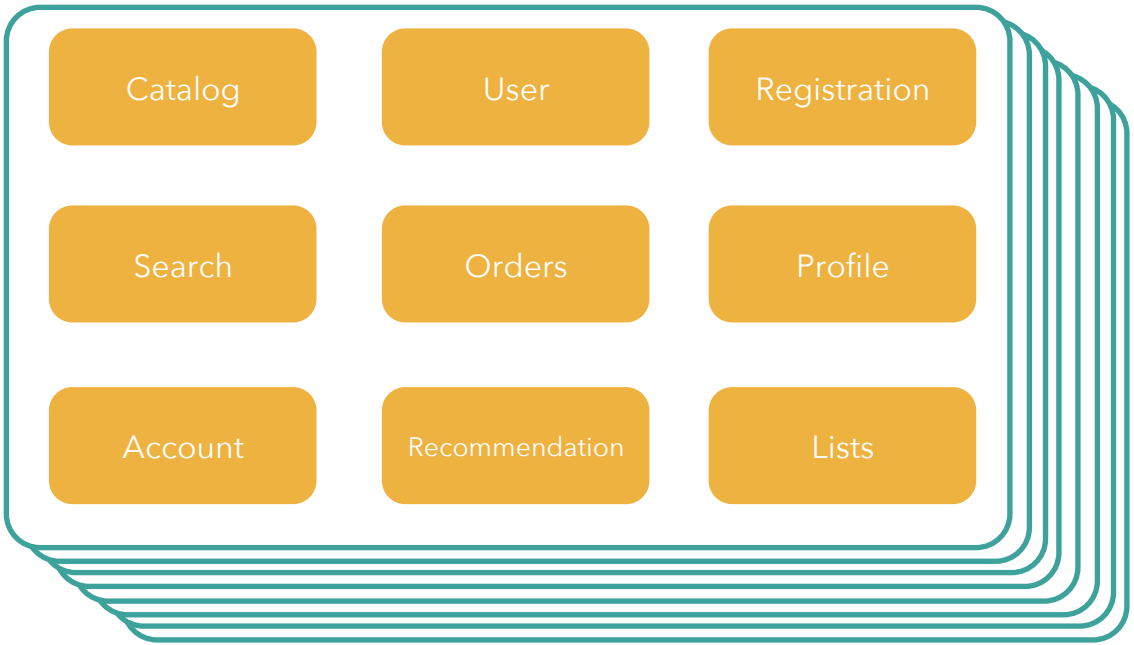
Pivotal

Monolithic Architectures



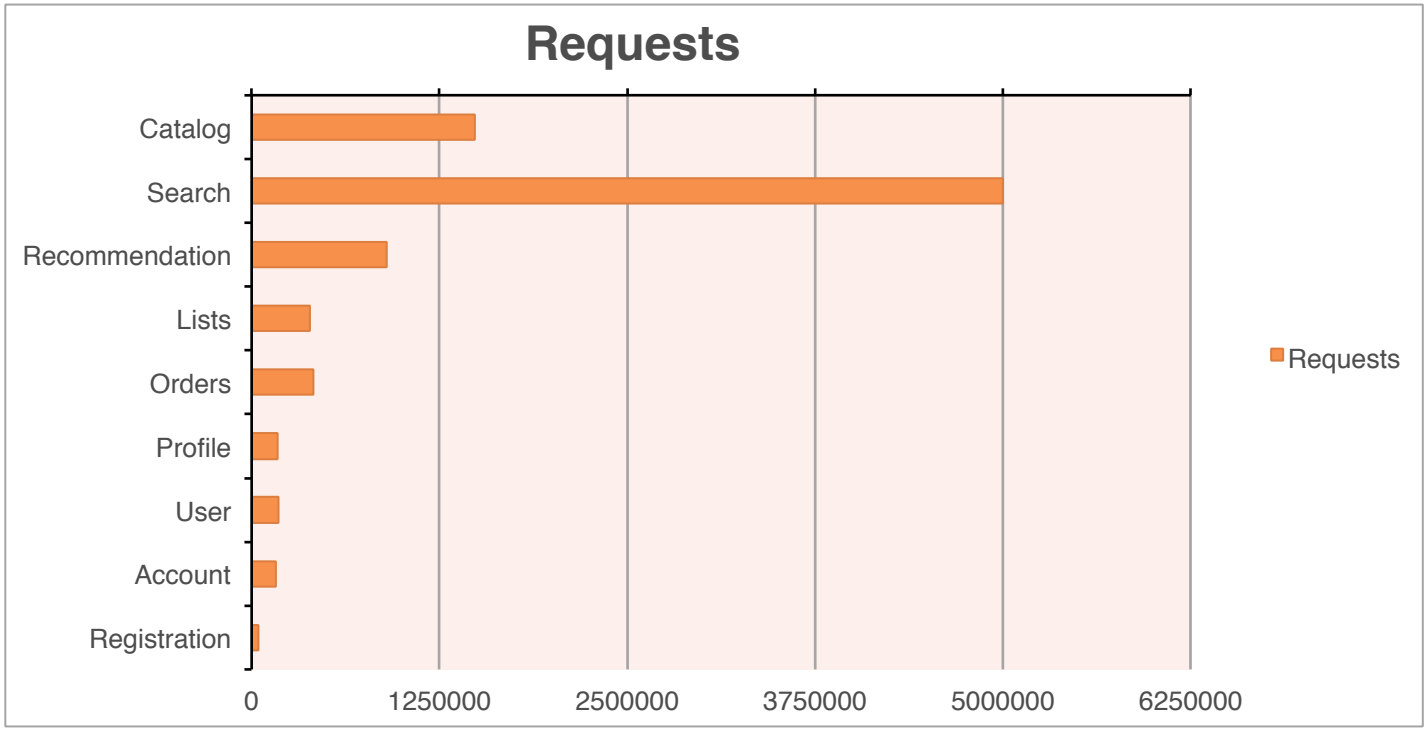
Pivotal

Scaling Monolithic Architectures



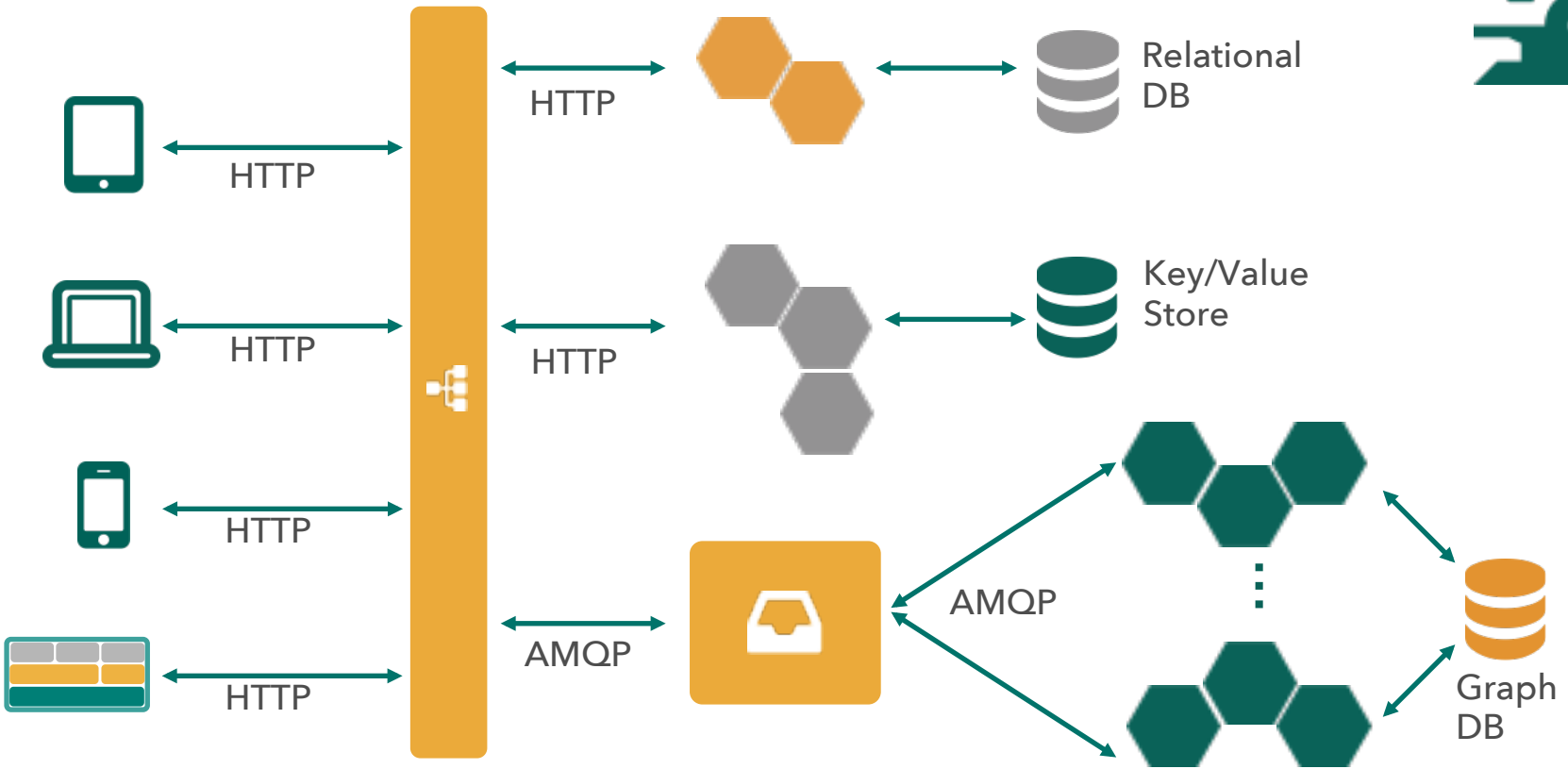
Pivotal

Linear scalability?



Pivotal

Microservice Architecture

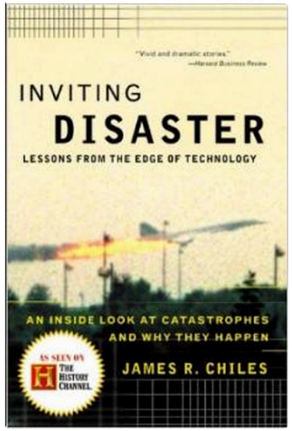


Pivotal

Microservice Architectures

- Simple
- 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



Flip to back

See all 2 images

Inviting Disaster: Lessons From the Edge of Technology Paperback

by James R. Chiles (Author)

★★★★½ 58 customer reviews

See all 7 formats and editions

Kindle	Hardcover	Paperback
$10.23	from $0.01	$12.08 Prime
	37 Used from $0.01 14 New from $23.59	50 Used from $0.71 19 New from $6.16 2 Collectible from $9.00

Combining captivating storytelling with eye-opening findings, *Inviting Disaster* delves inside some of history's worst catastrophes in order to show how increasingly "smart" systems leave us wide open to human tragedy.

Weaving a dramatic narrative that explains how breakdowns in these systems result in such disasters as the chain reaction crash of the Air France Concorde to the meltdown at the Chernobyl Nuclear Power Station, Chiles vividly demonstrates how the battle between man and machine may be escalating beyond manageable limits -- and why we all have a stake in its outcome.

Included in this edition is a special introduction providing a behind-the-scenes look at the World Trade Center catastrophe. Combining firsthand accounts of employees' escapes with an in-depth look at the

Read more

Share

Buy New Prime $12.08

Qty: 1

List Price: $15.99

Save: $3.91 (24%)

Only 16 left in stock (more on the way).

Ships from and sold by Amazon.com.

Gift-wrap available.

Add to Cart

Sign in to turn on 1-click ordering

Want it tomorrow, May 7? Order within **2 hrs 22 mins** and choose **One-Day Shipping** at checkout. Details

Buy Used Prime $9.22

Add to Wish List

Have one to sell? Sell on Amazon

Frequently Bought Together

Price for all three: $53.26

Add all three to Cart | Add all three to Wish List

Show availability and shipping details

- **This item:** Inviting Disaster: Lessons From the Edge of Technology by James R. Chiles Paperback $12.08
- The Logic Of Failure: Recognizing And Avoiding Error In Complex Situations by Dietrich Dorner Paperback $12.36
- Normal Accidents: Living with High-Risk Technologies by Charles Perrow Paperback $28.82

How many microservices?

ELEVEN

Pivotal

All teams will henceforth expose their data and functionality through service interfaces.

Teams must communicate with each other through these interfaces.

There will be no other form of inter-process communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.

It doesn't matter what technology they use.

All service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.

http://apievangelist.com/2012/01/12/the-secret-to-amazons-success-internal-apis/

Pivotal

Partitioning Strategies

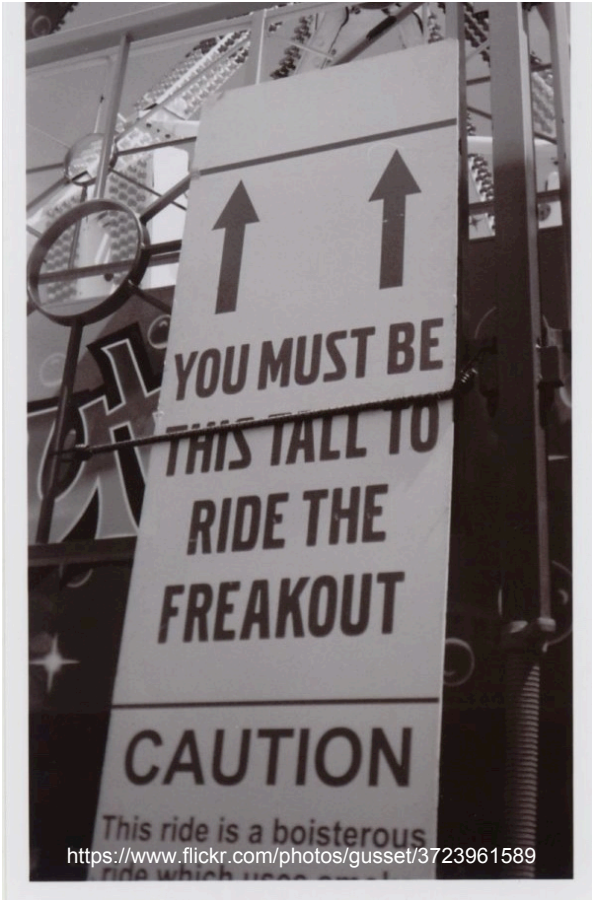
Challenges of Microservices

- Distributed System
- Remote Calls More Expensive Than In-process Calls
- Eventual Consistency
- Features Spanning Multiple Services
- Dependency Management / API Versioning
- Refactoring Module Boundaries

Pivotal

You must be this tall to use Microservices…

- Rapid provisioning
- Basic monitoring
- Rapid application deployment
- DevOps culture



https://www.flickr.com/photos/gusset/3723961589

Source: "Microservice Prerequisites," Martin Fowler, August 2014.

Pivotal

Platform Features

- Environment Provisioning
- On-Demand/Automatic Scaling
- Failover/Resilience
- Routing/Load Balancing
- Data Service Operations
- Monitoring

Pivotal

Pattern:

Configuration/Service Consumption

Pivotal

What is configuration?

- Resource handles to databases and other backing services
- Credentials to external sources (e.g. S3, Twitter, ...)
- Per-deploy values (e.g. canonical hostname for deploy)
- ANYTHING that's likely to vary between deploys (dev, test, stage, prod)

Pivotal

Where NOT to store it:

- In the **CODE** (Obvious)
- In **PROPERTIES FILES** (That's code...)
- In the **BUILD** (ONE build, MANY deploys)
- In the **APP SERVER** (e.g. JNDI datasources)

Pivotal

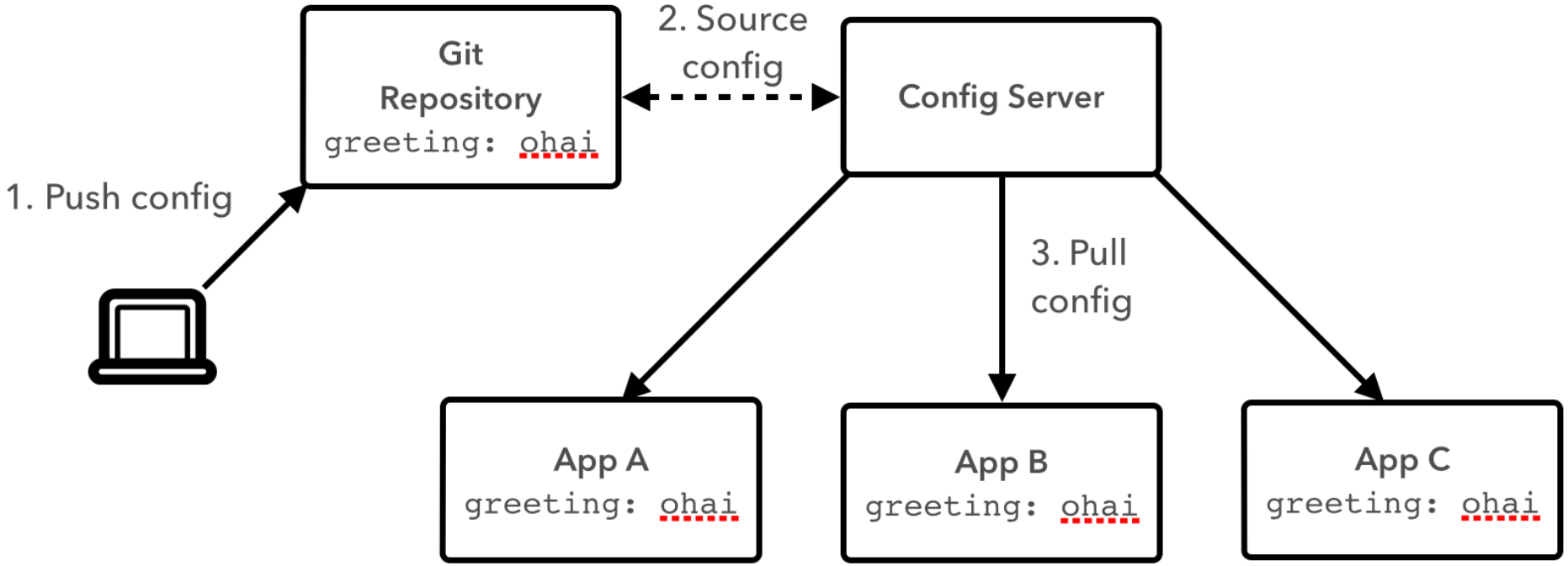
Why environment variables?

When "...the codebase could be made open source at any moment, without compromising any credentials."

http://12factor.net/config

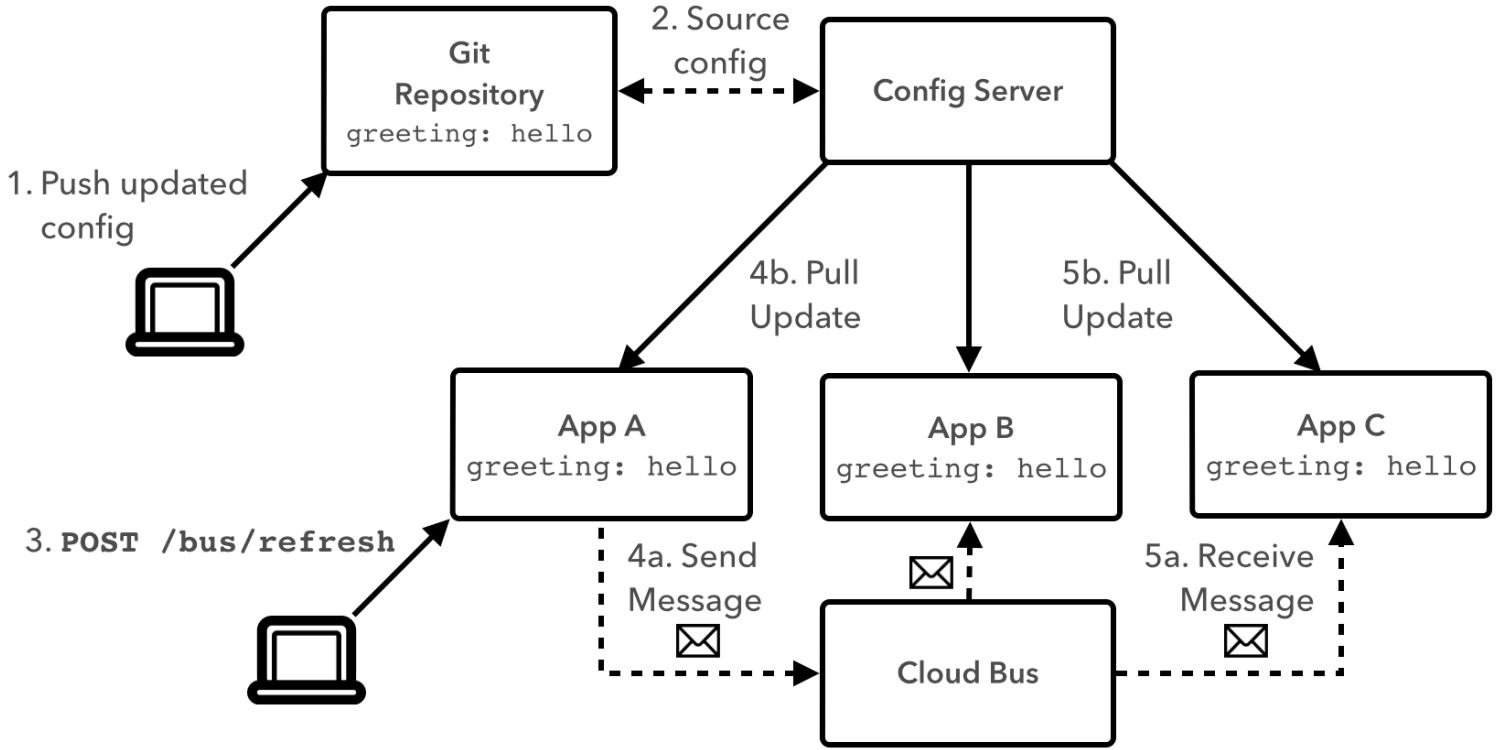
Pivotal

Config Server

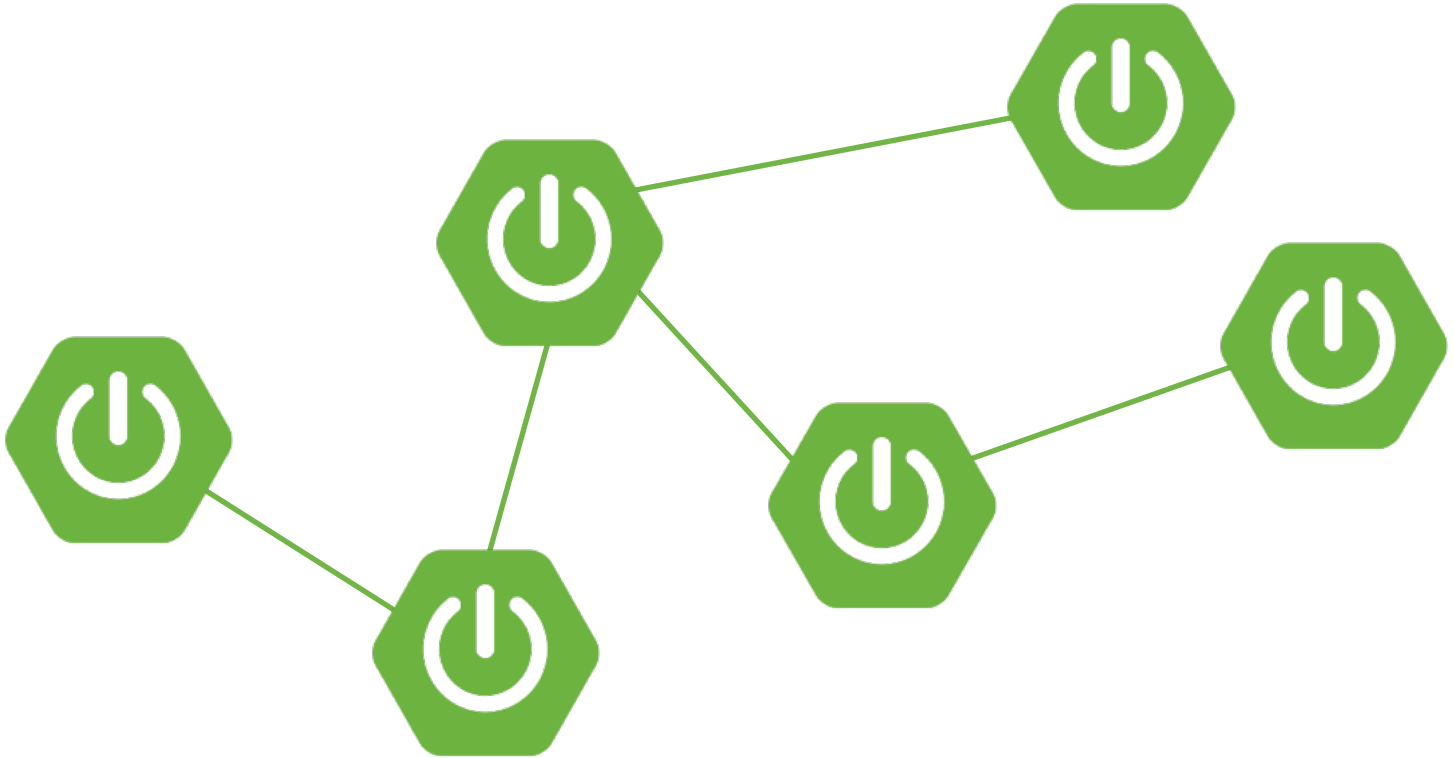


Pivotal

Config Server + Cloud Bus

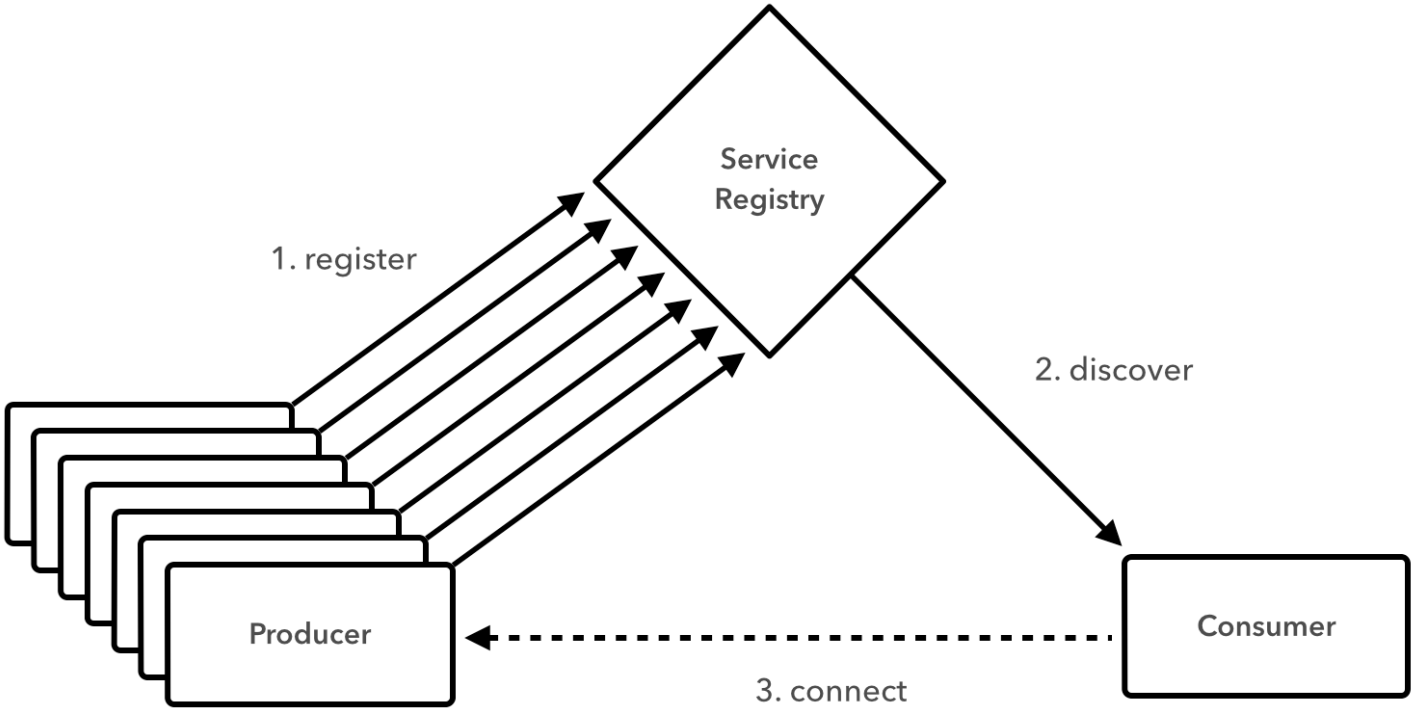


But no Microservice is an Island…



Pivotal

Service Registration/Discovery



Service Registration/Discovery

```
@SpringBootApplication
@EnableCircuitBreaker
@EnableDiscoveryClient
public class CustomerApp extends RepositoryRestMvcConfiguration {

    @Override
    protected void configureRepositoryRestConfiguration(RepositoryRestConfiguration
config) {
        config.exposeIdsFor(Customer.class);
    }

    public static void main(String[] args) {
        SpringApplication.run(CustomerApp.class, args);
    }

}
```

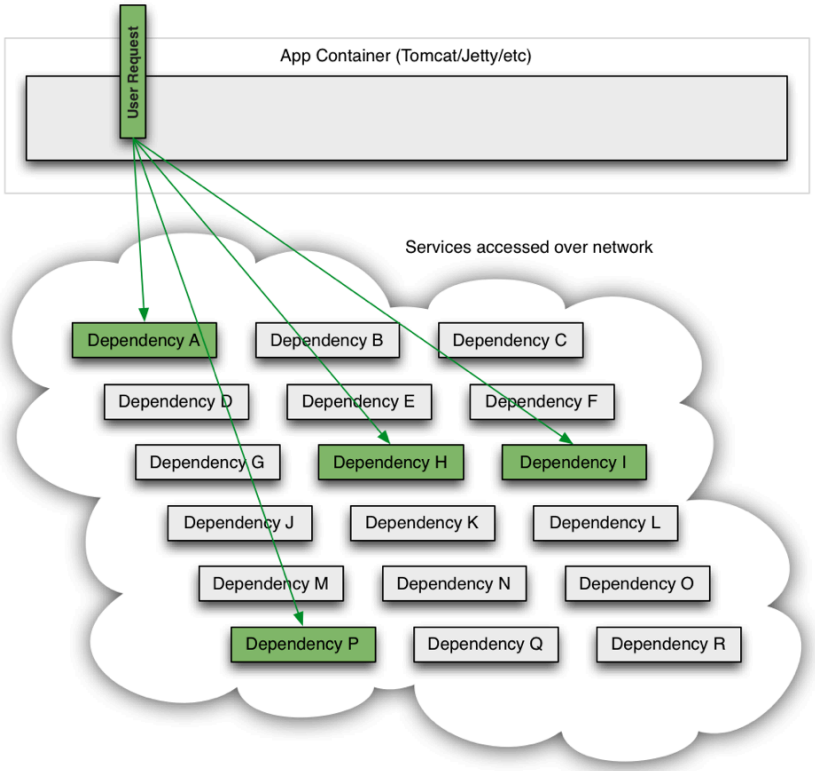
Pivotal

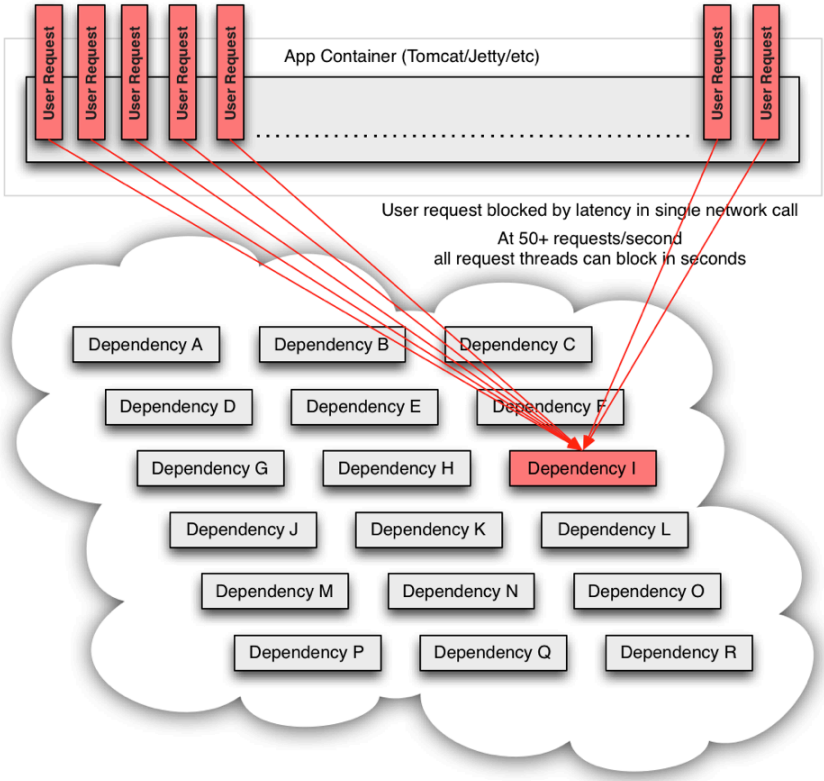
Pattern:

Fault Tolerance

Pivotal

Fault Tolerance at Netflix





http://techblog.netflix.com/2012/02/fault-tolerance-in-high-volume.html

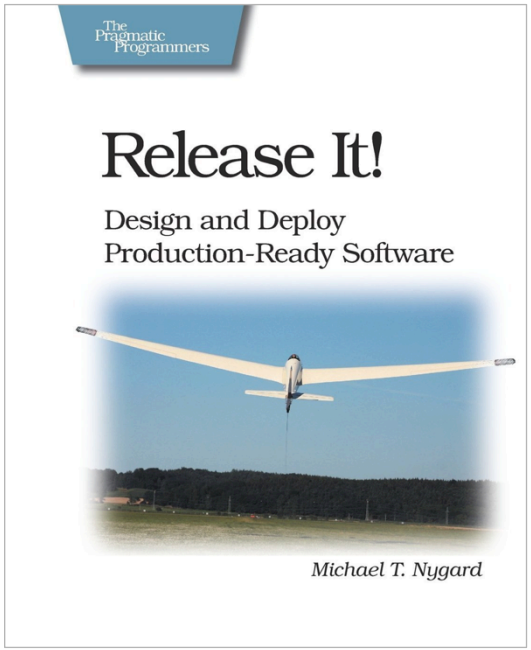
Pivotal

Without taking steps to ensure fault tolerance, 30 dependencies each with 99.99% uptime would result in 2+ hours downtime/month (99.99%^{30} = 99.7% uptime = 2+ hours downtime in a month).

http://techblog.netflix.com/2012/02/fault-tolerance-in-high-volume.html

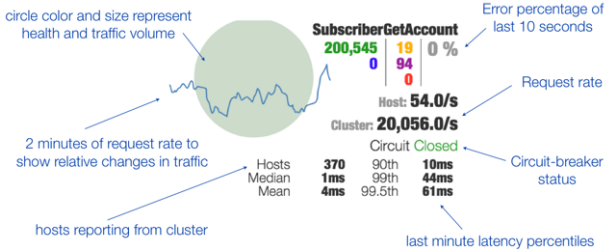
Pivotal

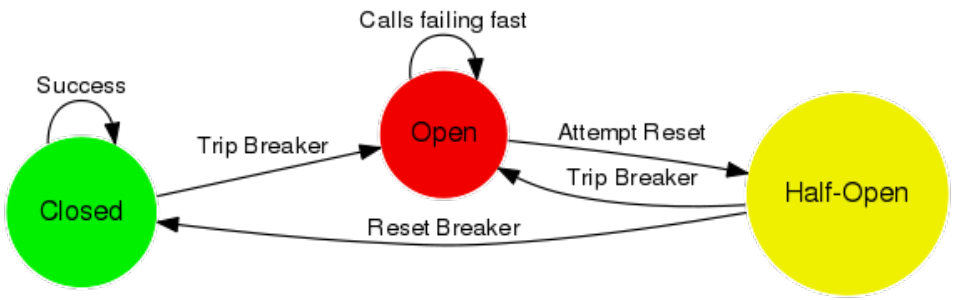
Circuit Breaker



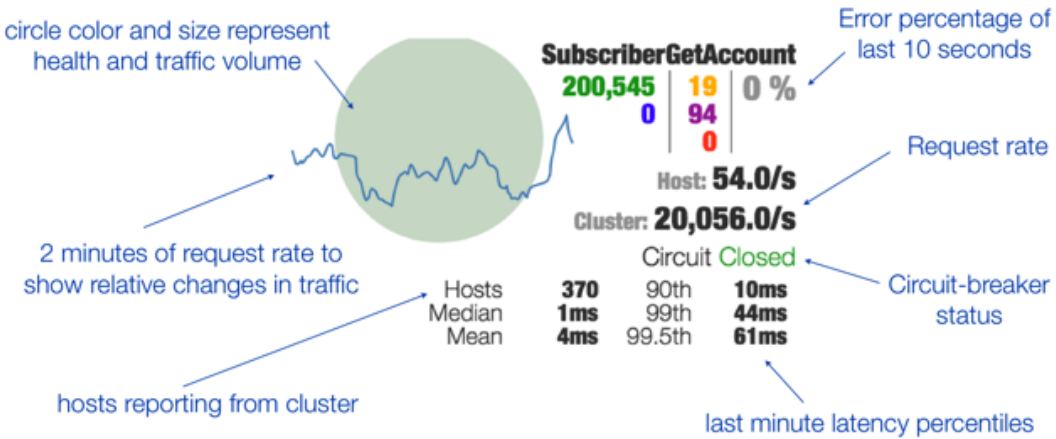
Resilience

- On large distributed systems, failures are a norm and not an exception, be ready for that.





Hystrix Dashboard



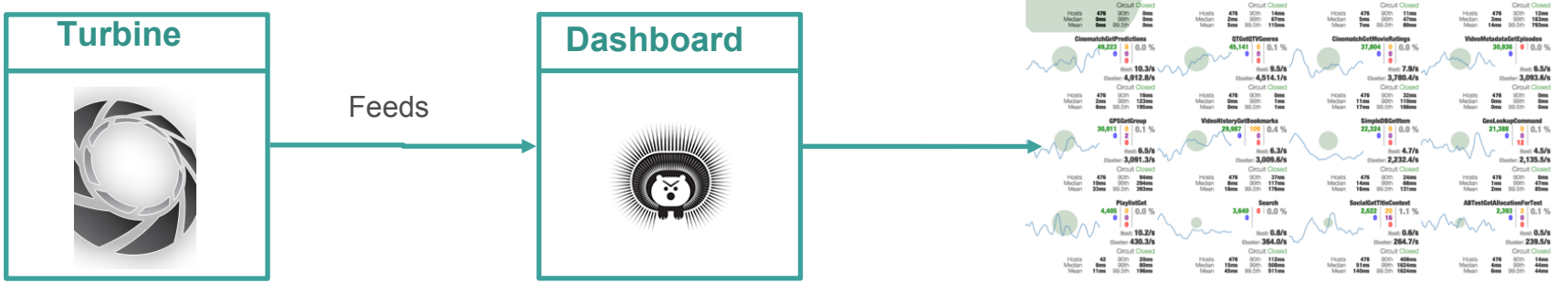
Rolling 10 second counters
with 1 second granularity

Successes	200,545	19	Thread timeouts
Short-circuited (rejected)	0	94	Thread-pool Rejections
		0	Failures/Exceptions

Pivotal

A Failing Circuit

Putting it all together



Spring Cloud Services Suite

Spring Cloud Services

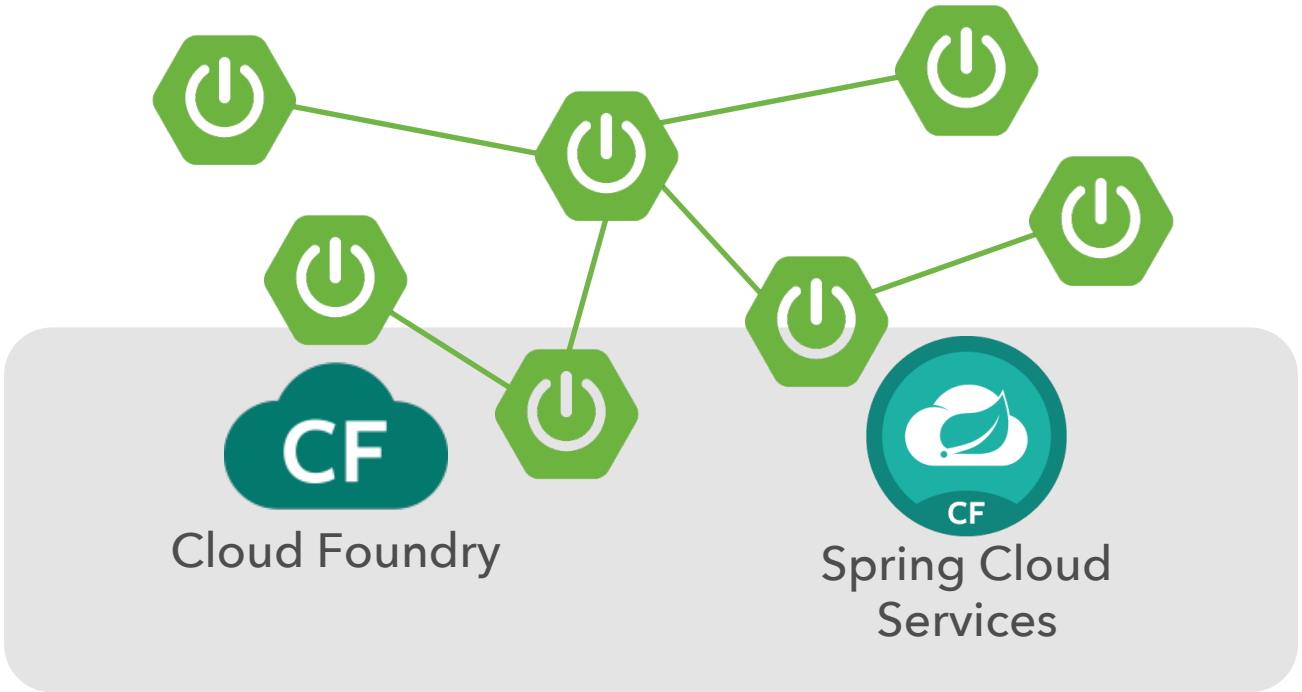
Config Server

Service Registry

Circuit Breaker Dashboard

Pivotal

It takes a platform…



Spring Cloud Services Suite

- Installed via Pivotal Ops Manager
- Adds all services to Pivotal Cloud Foundry Marketplace
- **Dependencies:**
  - MySQL for PCF
  - RabbitMQ for PCF
- Public Beta: May 2015

Pivotal

Spring Cloud Config Server

- Spring Cloud Config Server
- Service Binding via Spring Cloud Connector
- Git/SVN URL for Config Repo provided via Service Dashboard (post-provisioning)
- Single tenant, scoped to CF space
  (nothing prevents shared Git repo)

Spring Cloud Service Registry

- Service Registration and Discovery via Netflix OSS Eureka
- Service Binding via Spring Cloud Connector
- Single-tenant, scoped to CF space
- Registration via CF Route
- PCF 1.5: Support Direct Address (“promiscuous”) Mode

Pivotal

Spring Cloud Services Suite

Circuit Breaker Dashboard

- Netflix OSS Turbine + Hystrix Dashboard
- Aggregation via AMQP (RabbitMQ)
- Binding via Spring Cloud Connector
- Single-tenant, scoped to CF space

Pivotal

A NEW PLATFORM FOR A NEW ERA