
Pivotal
Cloud Native Design &
Development

Defining Cloud-Native *(AKA: 'Agility, Agility, Agility')*

Cloud Native is not about **where** you run apps, but **how**

- **'Twelve-Factor' Architecture**
- **Microservices Designs**
- **"Antifragility"**
- **Containers**
- **The Application "Dial Tone"**
- **Cultural Shift from Silo IT to DevOps**

Pivotal Cloud Foundry

Cloud Native App Development

Twelve-Factor Apps

I. Codebase
One codebase tracked in SCM, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Configuration
Store config in the environment

IV. Backing Services
Treat backing services as attached resources

V. Build, Release, Run
Strictly separate build and run stages

VI. Processes
Execute app as stateless processes

VII. Port binding
Export services via port binding

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity
Keep dev, staging, prod as similar as possible

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin / mgmt tasks as one-off processes

Architectural and development practices – http://12factor.net

Pivotal

Pivotal Cloud Foundry

Microservices

Microservice: Definition

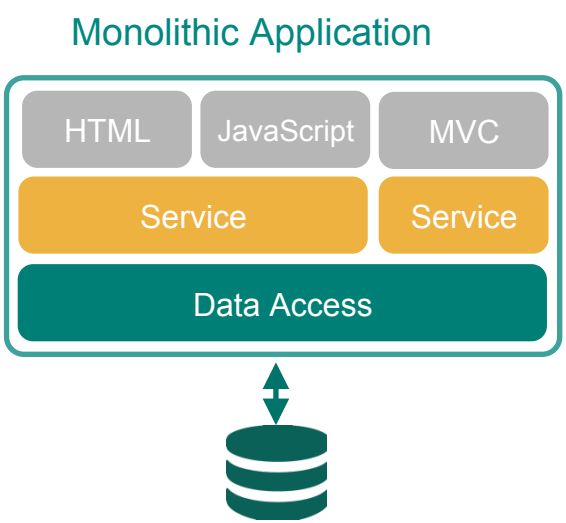
If every service has to be updated in concert,
it's not loosely coupled!

"Loosely coupled service oriented architecture with bounded contexts"

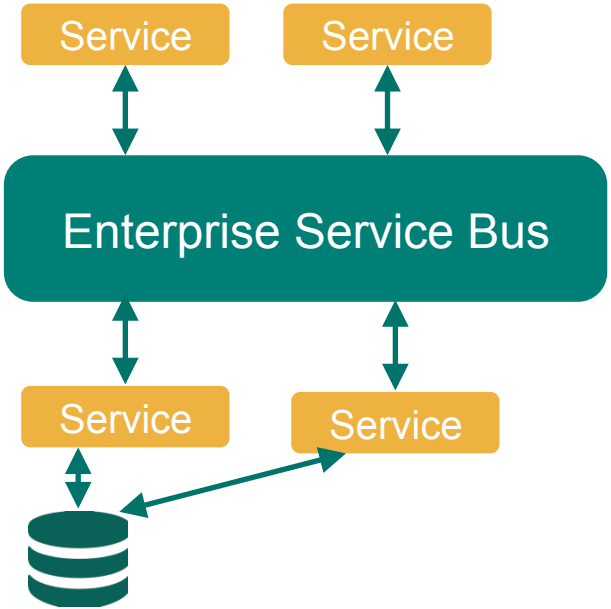
If you have to know about surrounding services you don't have a bounded context.

- Adrian Cockcroft

Microservices are NOT



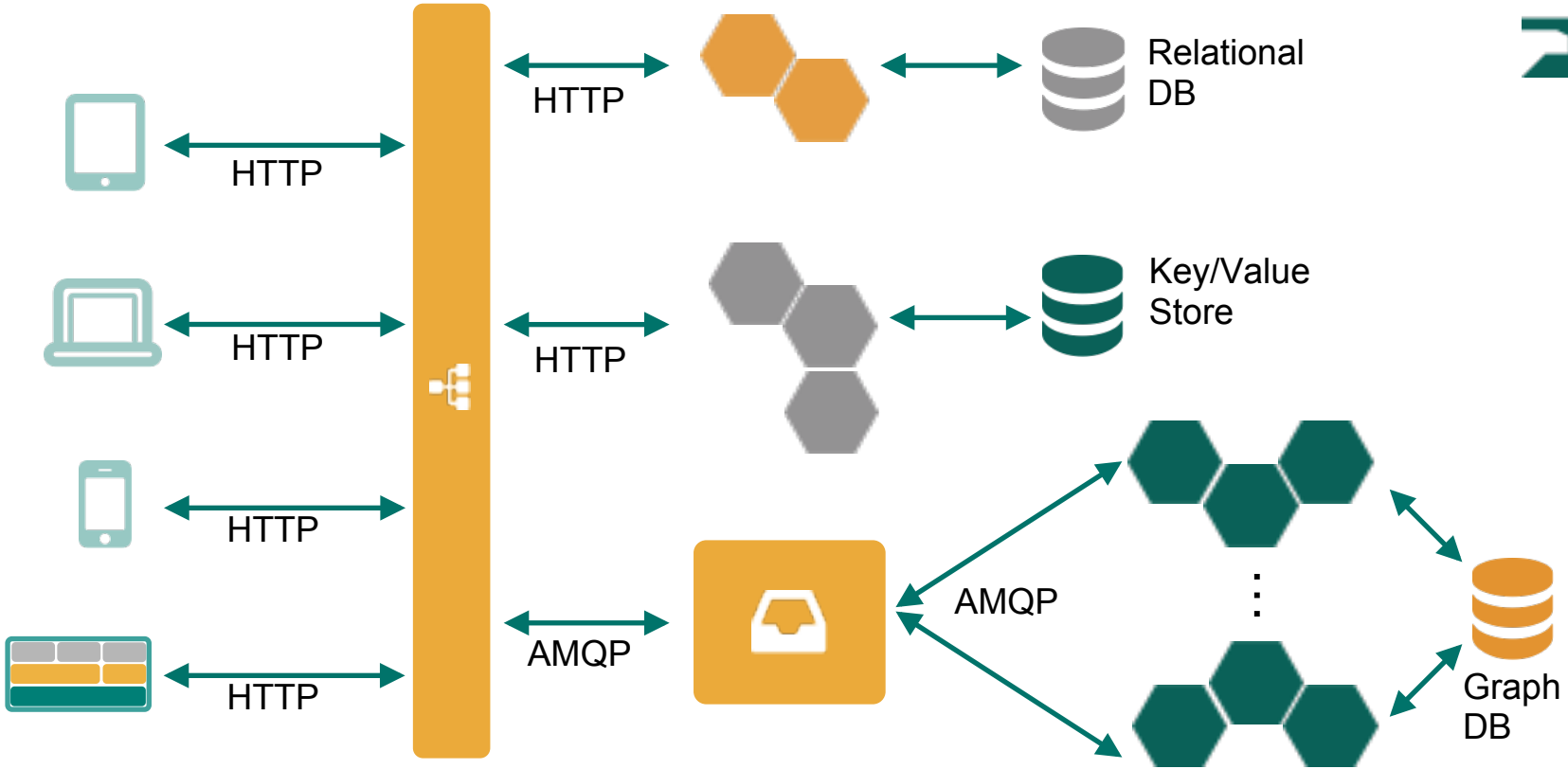
OR



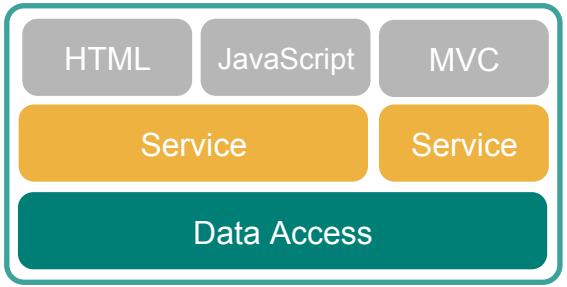
Tightly Coupled

Centralized

Microservice Architecture



Monoliths: challenges



- Traditional monolithic design patterns are not appropriate for the cloud.
- Monoliths couple change cycles together.
- Monoliths services can’t be scaled independently.
- Difficult coordination: too many developers in one code base.
- Developers struggle to understand a large codebase.
- Long term commitment to the tech stack.

Microservice: benefits

- Change cycles are decoupled: Enabling frequent deploys
- Allow for efficient and independent scaling
- Developers learn a smaller codebase faster
- Better coordination and scaling of development: Fewer developers in each code base
- Eliminate long-term commitment to technical stack

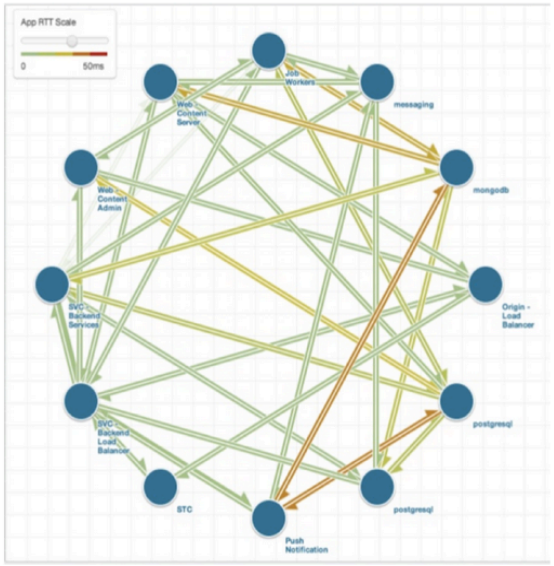
BUT ...

Challenges in a Distributed System

- Configuration management
- Registration and discovery
- Routing and load balancing
- Fault tolerance and isolation
- Aggregation and transformation
- Monitoring and distributed tracing
- Process management

Microservice pioneers

Netflix



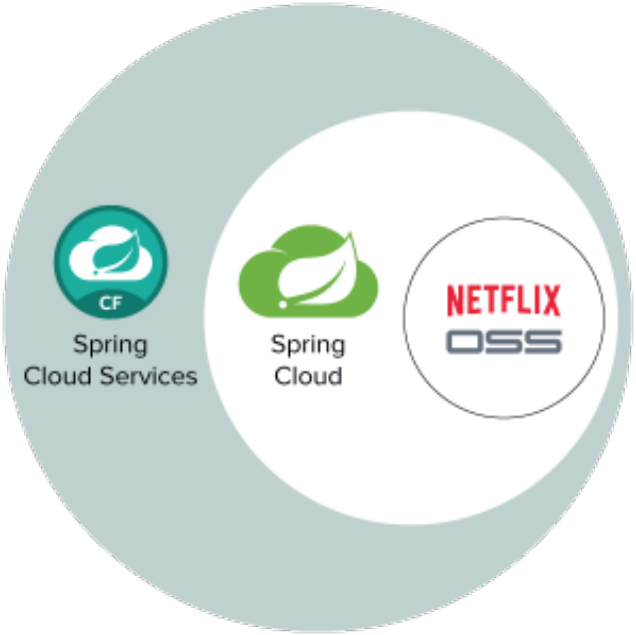
Gilt Groupe (12 of 450)

Twitter

Netflix Cloud Libraries

- Netflix needed to be faster to win / disrupt
- Pioneer and vocal proponent of microservices – the key to their speed and success
- Config Server, Eureka, Hystrix, Ribbon, Zuul ...
- Netflix OSS supplies parts, but it's not a solution

Spring Cloud - Spring Cloud Services



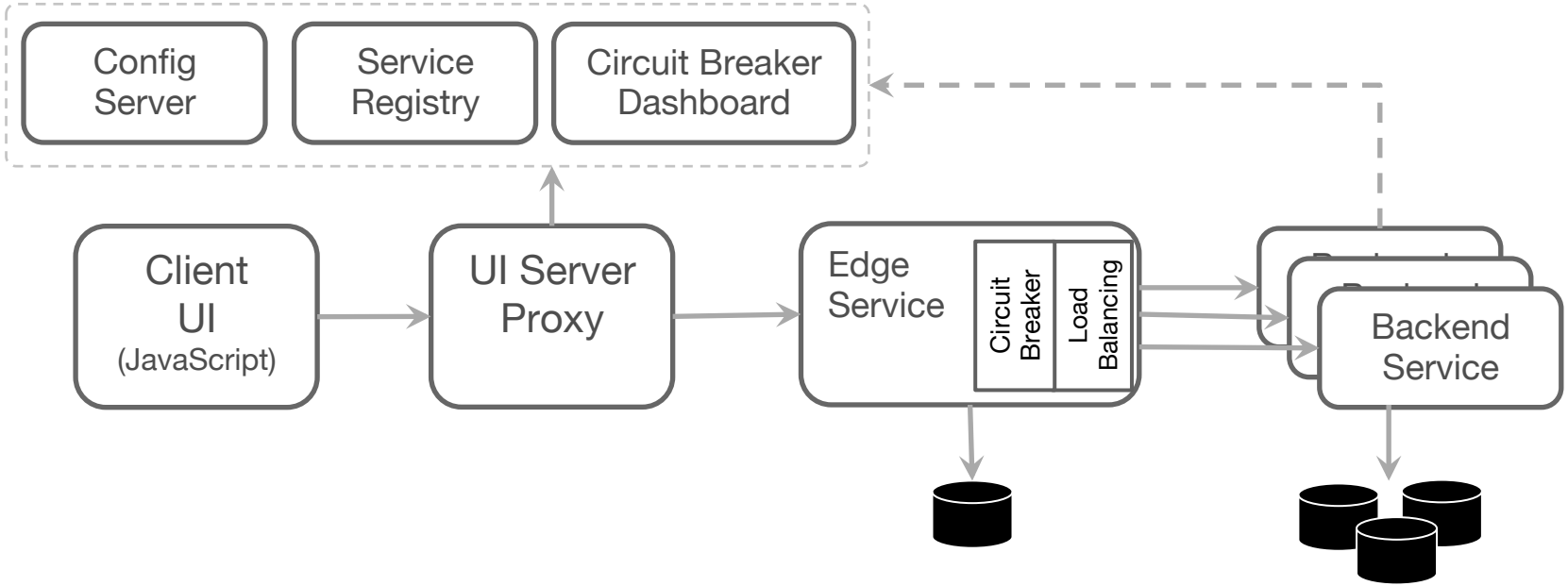
Service Registry

Config Server

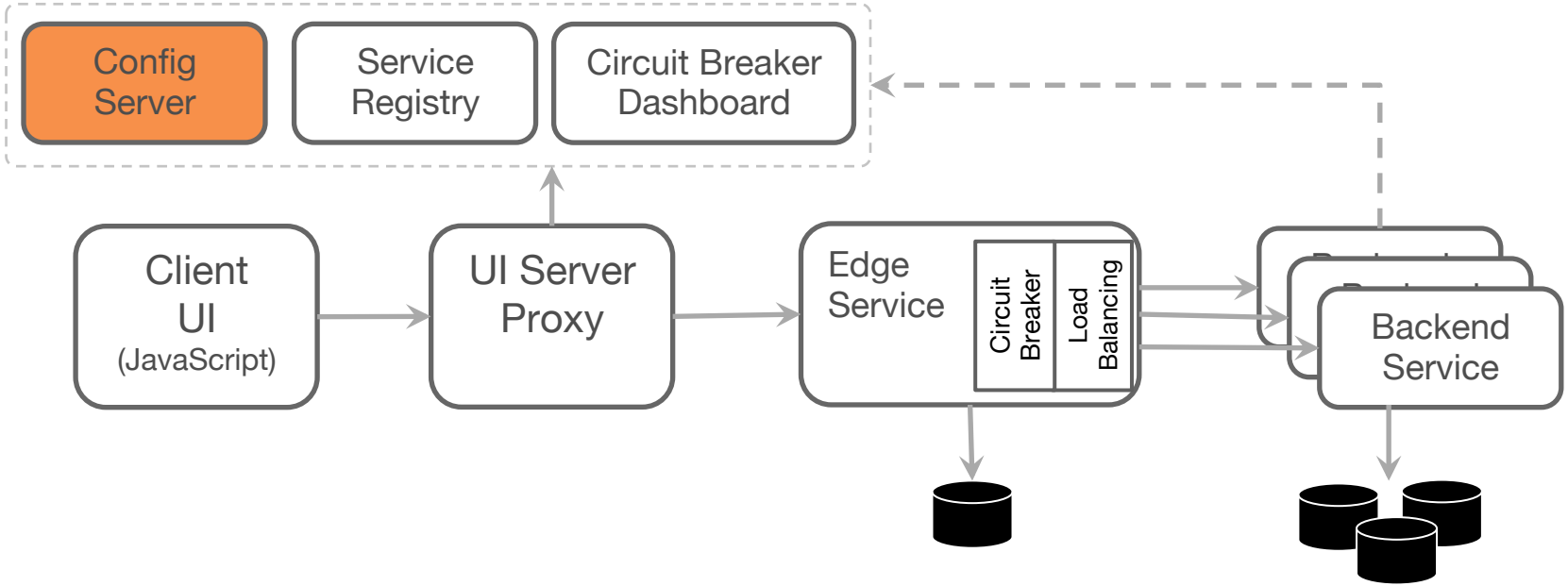
Circuit Breaker

- Operationalizes Spring Cloud in your deployment environment
- Provisions highly available infrastructure (messaging, databases) to support Spring Cloud deployments
- Adds key security features, including OAuth support, required for enterprise deployment of Spring Cloud

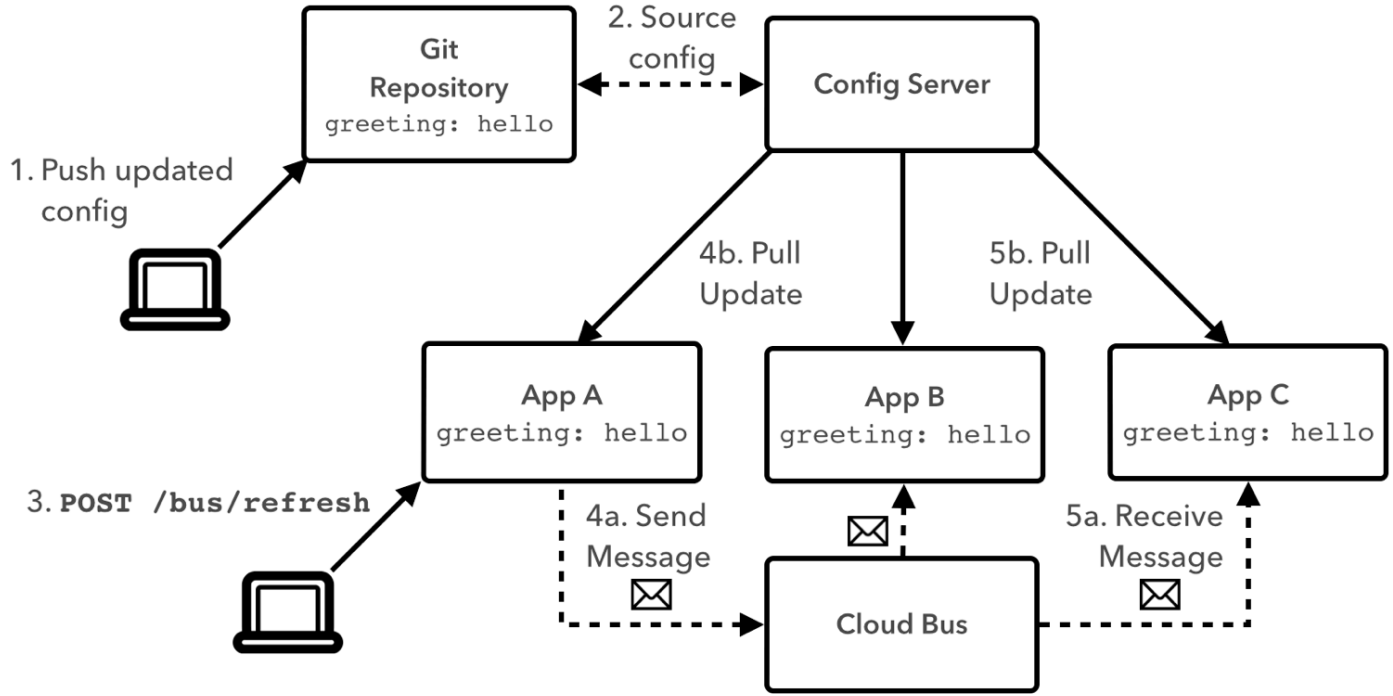
Design Patterns & Automation to the Rescue!



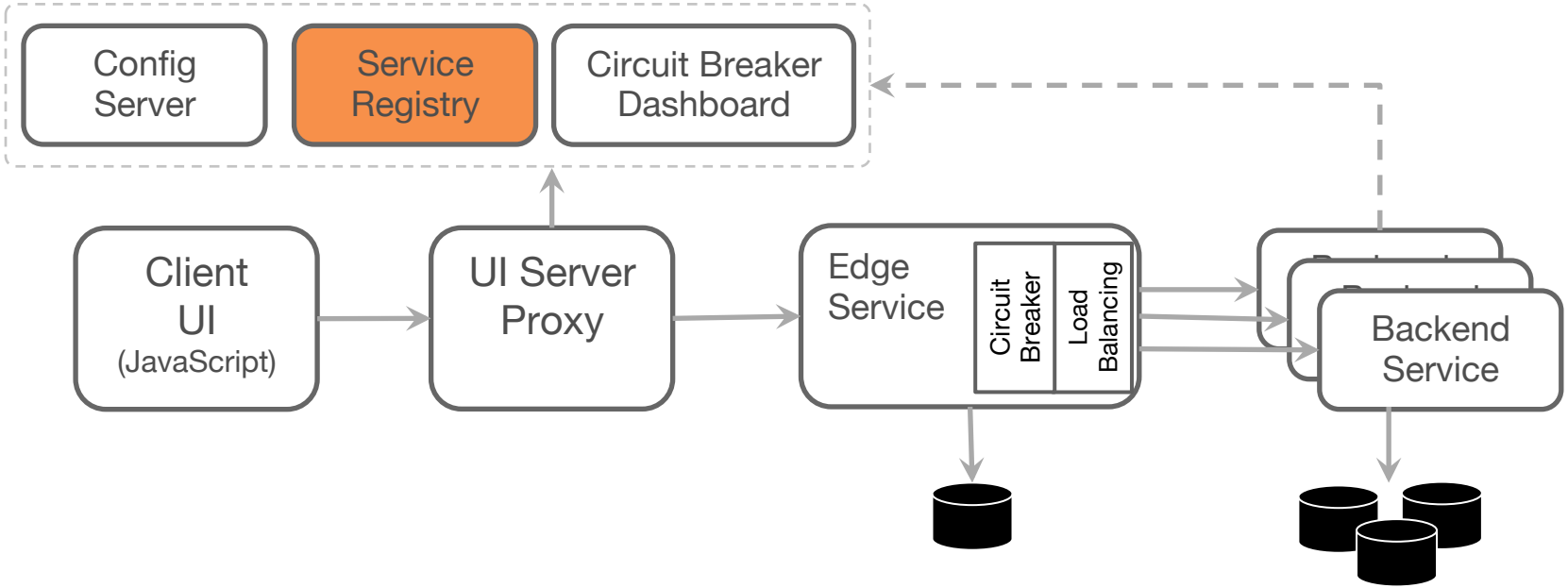
Configuration Management



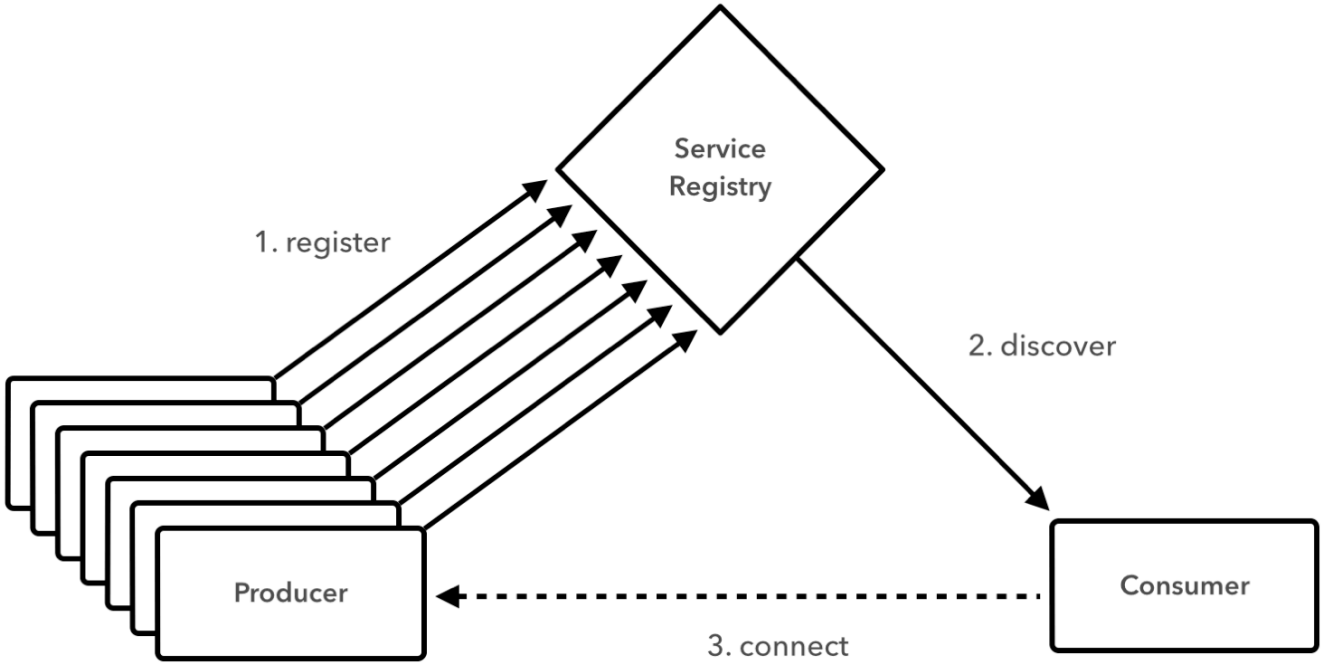
Configuration Server



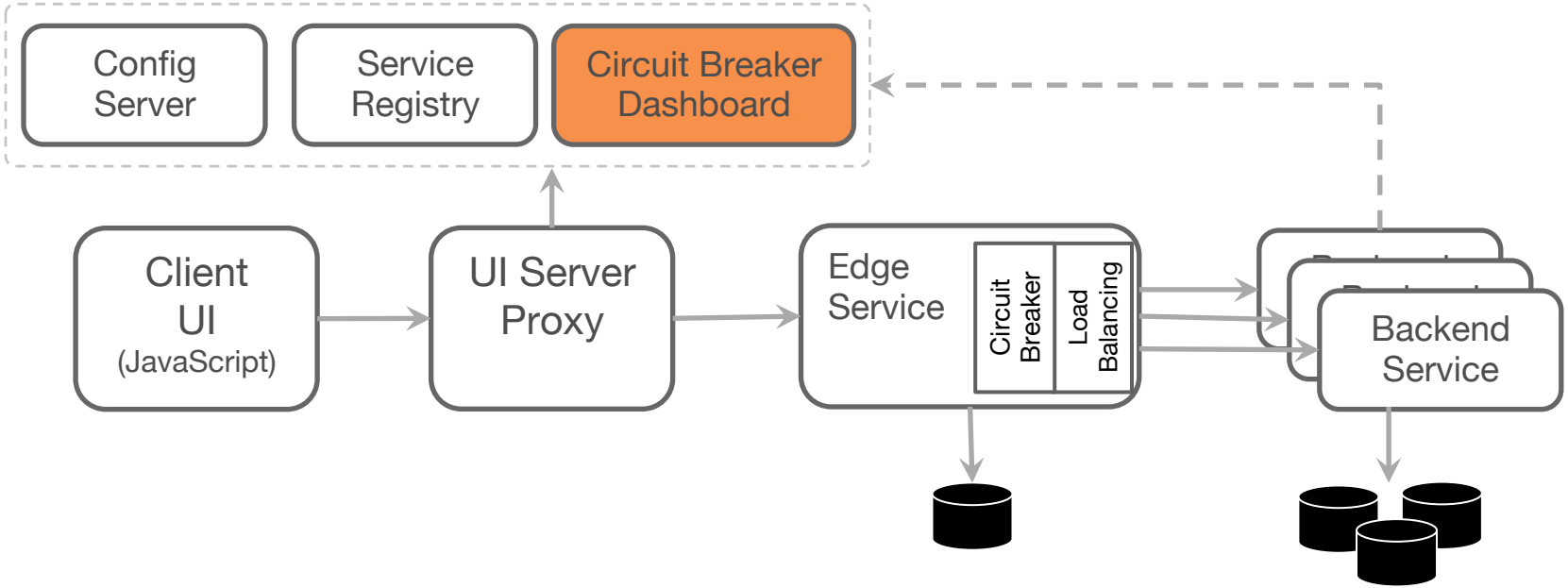
Registration and Discovery



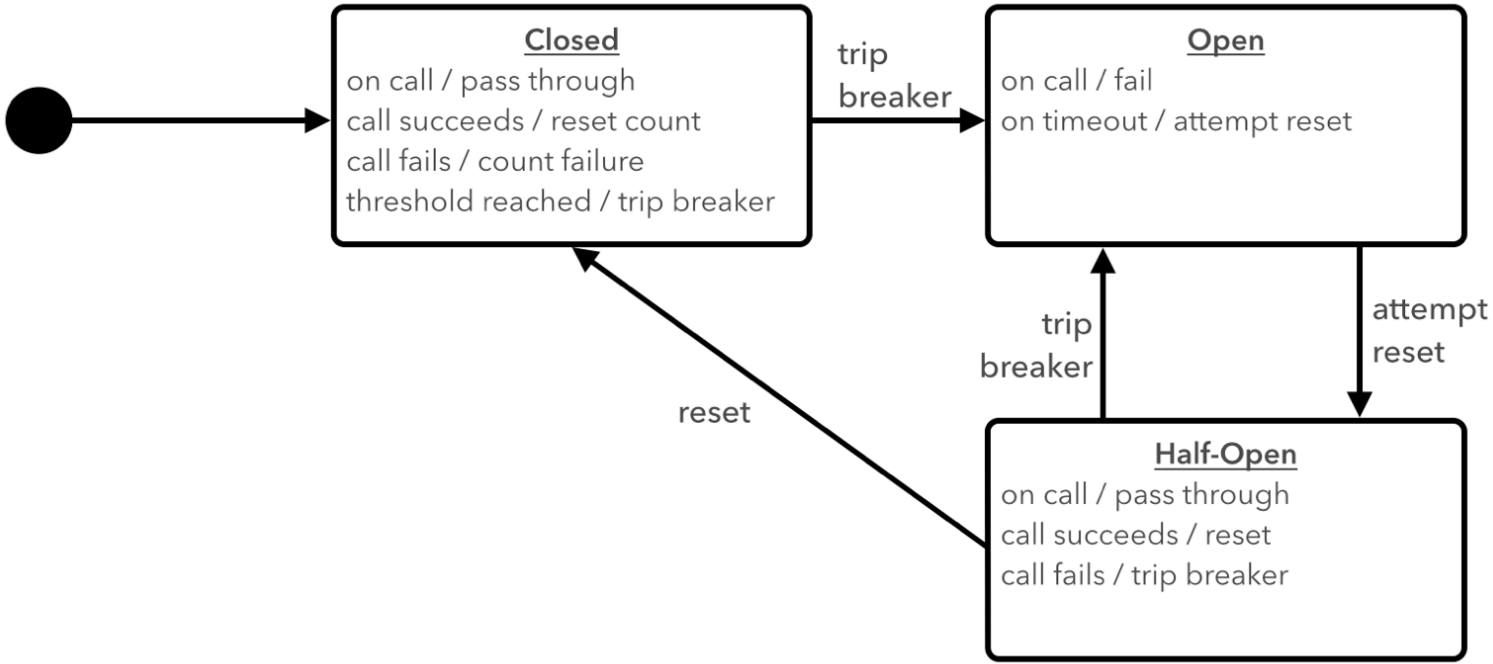
Registration and Discovery Server



Fault Tolerance and Isolation



Circuit Breaker




Pivotal
Thank You!
Pivotal