Pivotal

Cloud Native Journey – Mono2Micro

Srinivasa Vasu
@srinivasavasu
March 2018

Cloud Native Architecture

Why

What

How

Cloud Native Architecture

Why

- Cloud Computing
- New Demands
- Being Reactive

What

- Reactive Design
- 12Factor apps
- Microservices

How

- Tools
- Platform
- Process and Culture

Cloud Native Architecture

Why

What

- Reactive Design
- 12Factor apps
- Microservices

How

- Tools
- Platform
- Process and Culture

Cloud Defined

- Cloud Computing
- Cloud Native Applications
- Cloud Native Application Architecture

Cloud-Era Demands

- Competitive landscape
- Consumer expectations
- Device Diversity & Ubiquity
- Scale

Being Reactive

- React to Events
- React to Load
- React to Failure
- React to Users

http://www.reactivemanifesto.org

Speed & Safety

- Design for quick recovery
- Quick recovery increases risk tolerance
- Risk tolerance enables innovation

Cloud Native Architecture

Why

- Cloud Computing
- New Demands
- Being Reactive

What

How

- Tools
- Platform
- Process and Culture

Pivotal

Reactive Design

- React to Events → Event-Driven
- React to Load → Scalable
- React to Failure → Resilient
- React to Users → Responsive

http://www.reactivemanifesto.org

The Twelve Factor App

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
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 the app as one or more 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 development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

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

The Twelve Factor App

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
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 the app as one or more 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 development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

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

The Twelve Factor App

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
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 the app as one or more 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 development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

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

The Twelve Factor App

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
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 the app as one or more 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 development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

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

The Twelve Factor App

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
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 the app as one or more 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 development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

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

The Twelve Factor App

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
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 the app as one or more 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 development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

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

The Twelve Factor App

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
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 the app as one or more 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 development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

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

Microservices

If services have to be updated together, they're not loosely coupled!

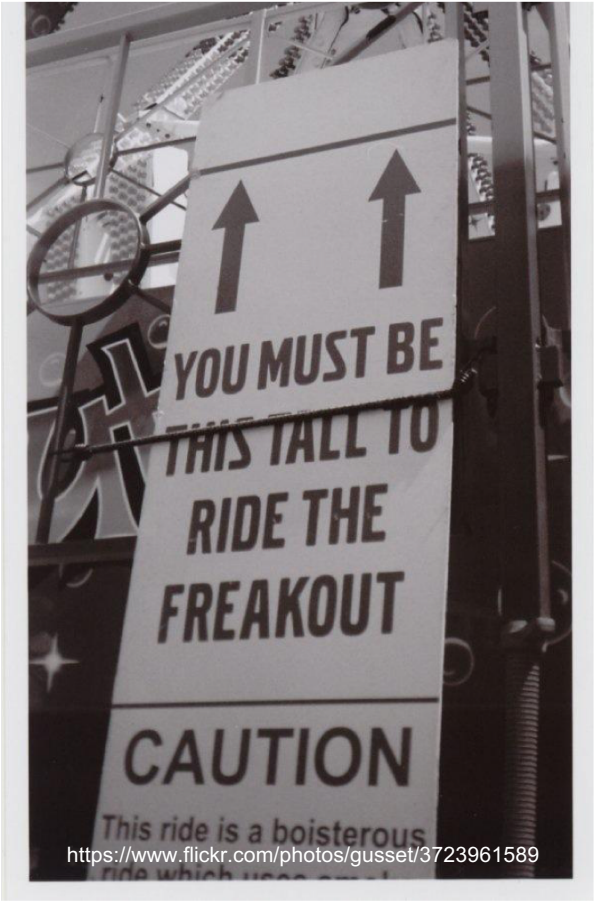
Loosely coupled service oriented architecture with bounded contexts

- Adrian Cockcroft

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

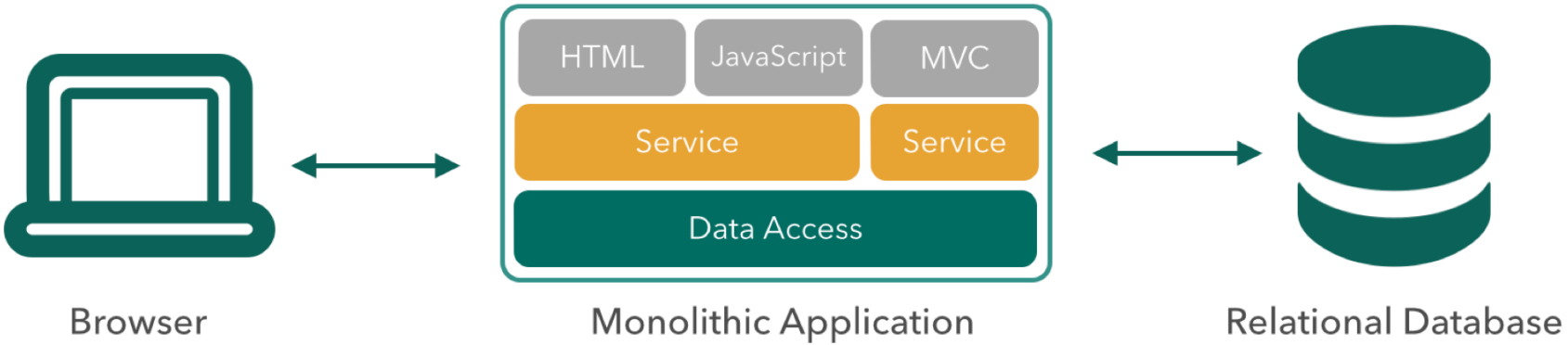
Min Prerequisites

- DevOps Culture
- Rapid Infrastructure Provisioning
- Rapid Application Development
- Basic Monitoring

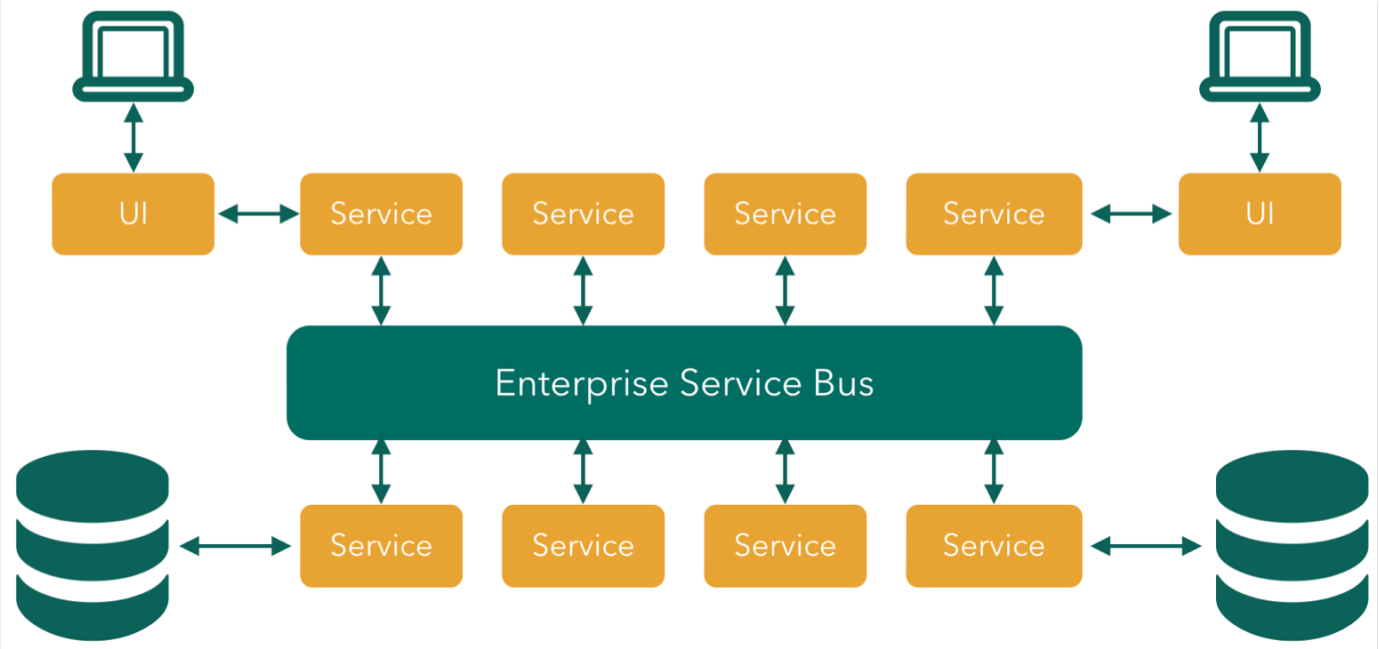


https://www.flickr.com/photos/gusset/3723961589

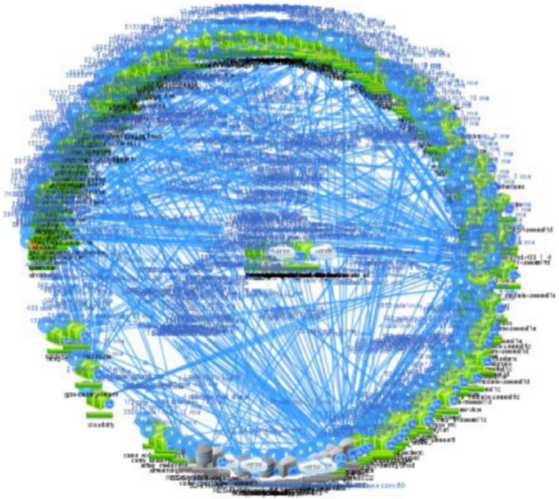
Not Monoliths



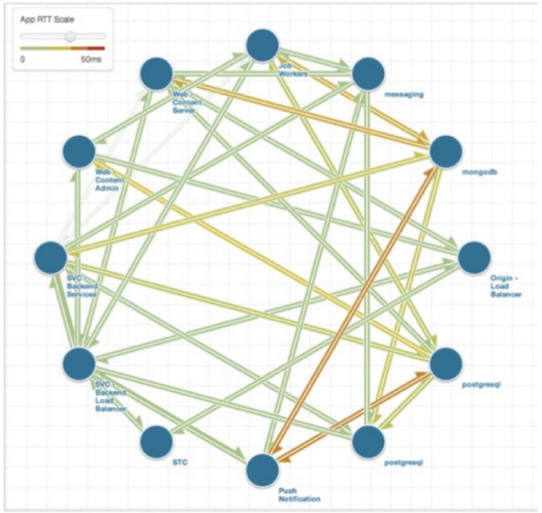
Not ESB-Centric SOA



Microservice Pioneers



Netflix



Gilt Groupe (12 of 450)

Twitter

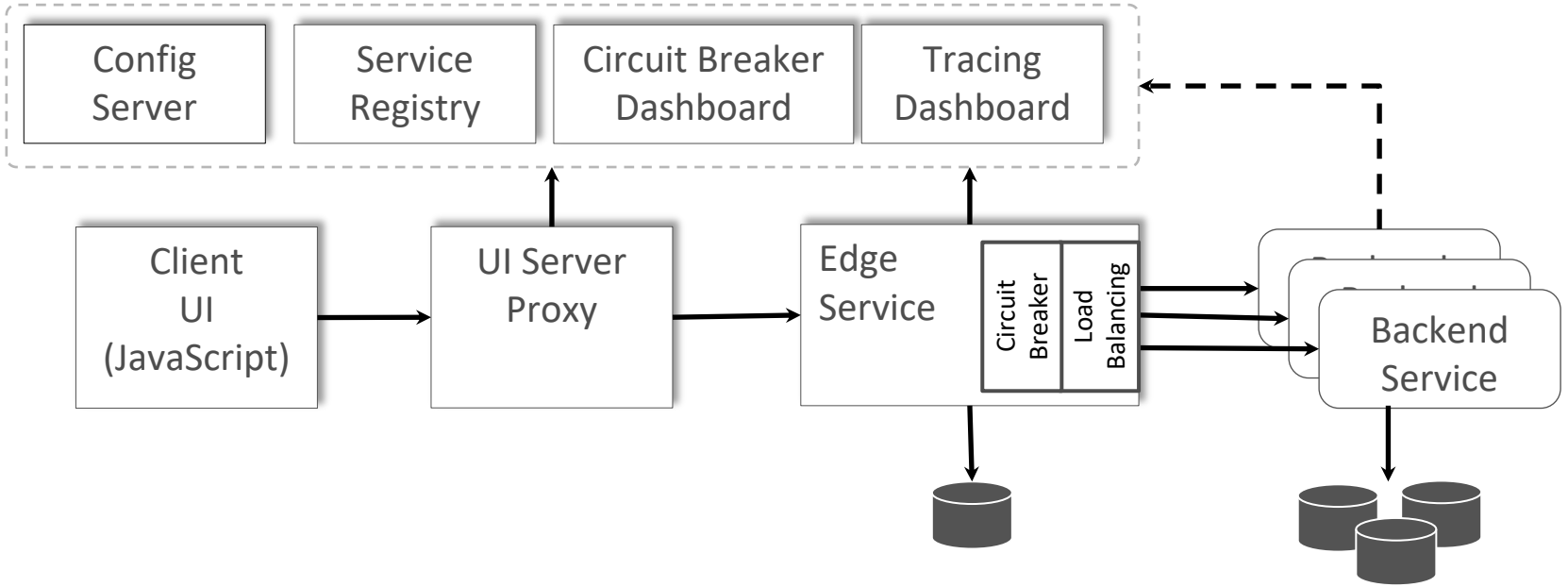
Who Moved My Complexity?

- Microservices are individually simple
- Complexity is transferred to the ecosystem

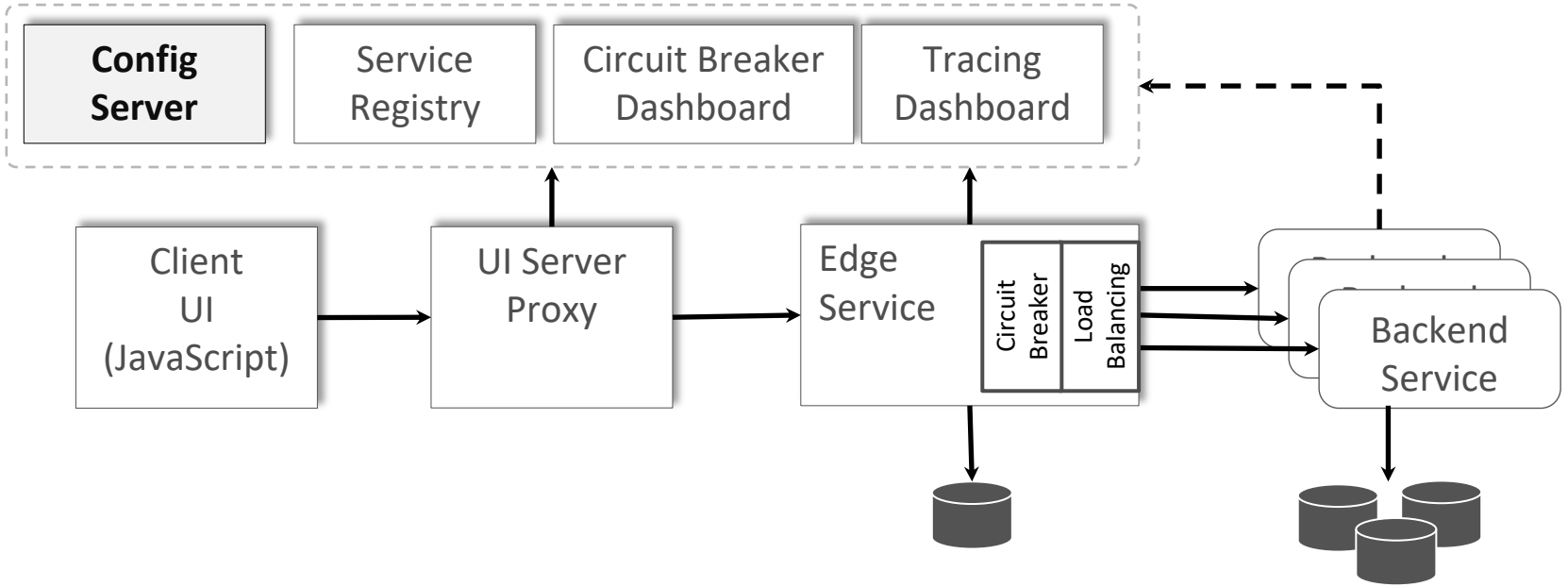
Challenges in a Distributed System

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

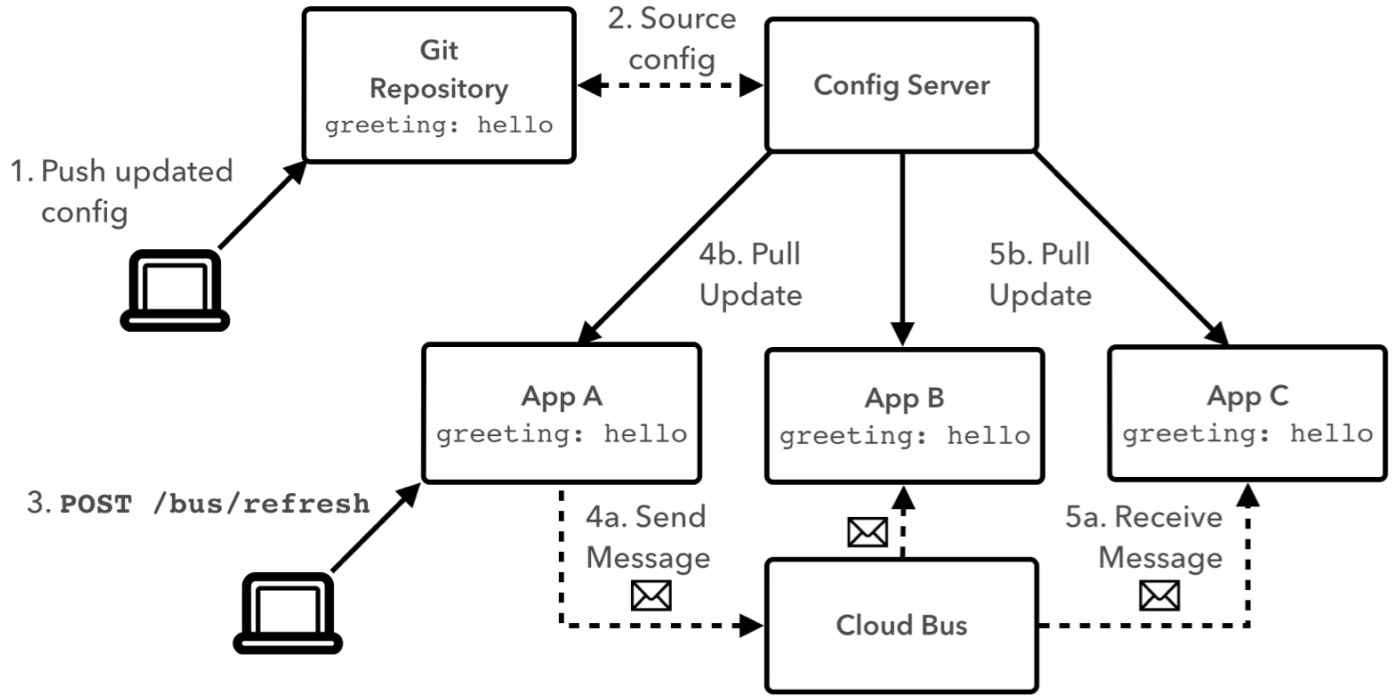
Design Patterns & Automation to the Rescue!



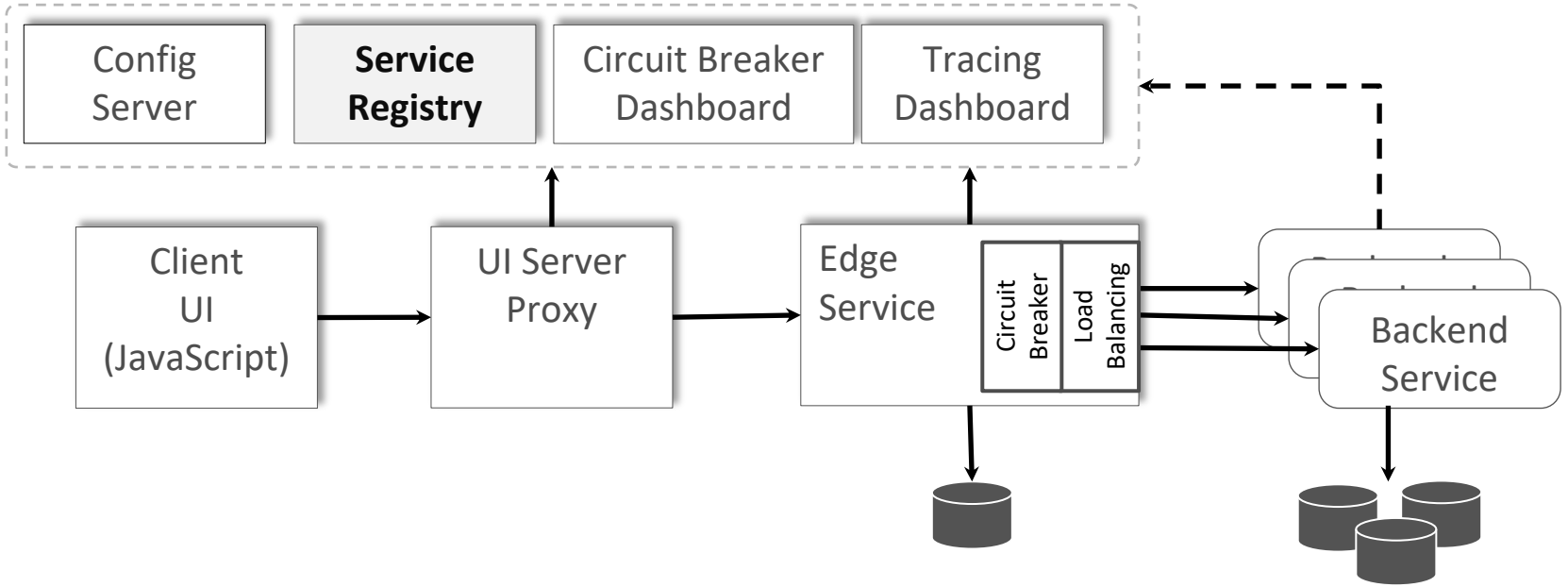
Configuration Management



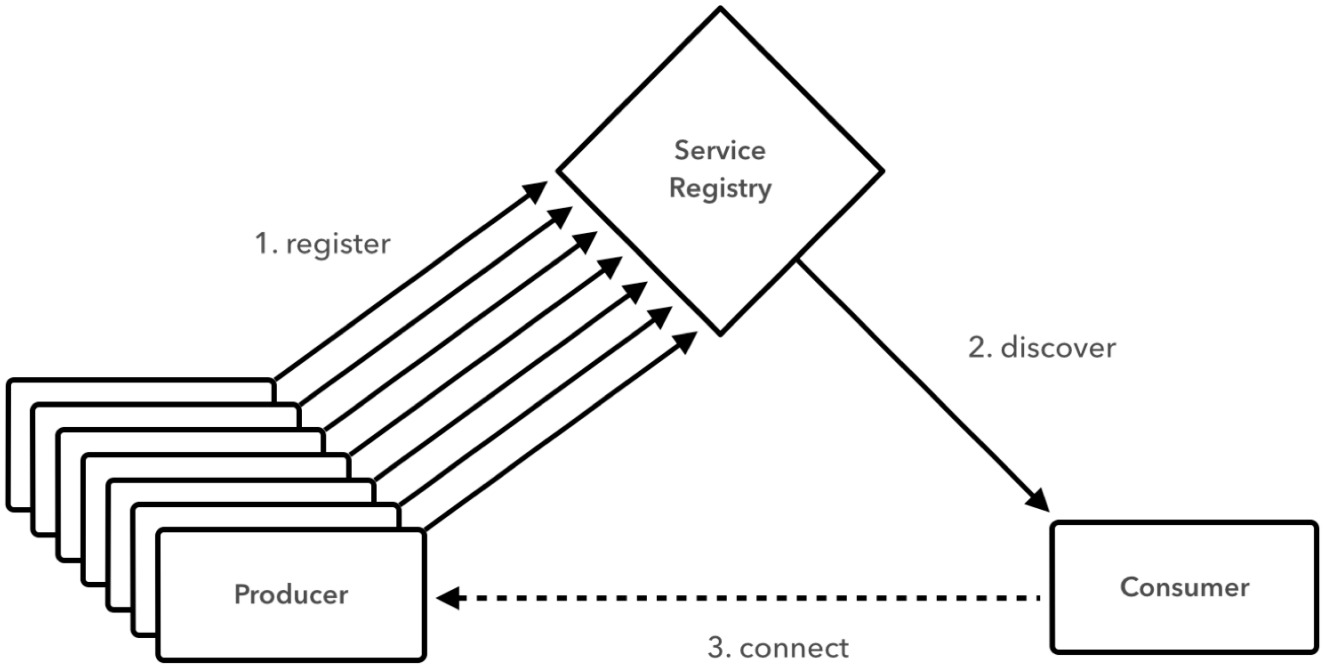
Config Server



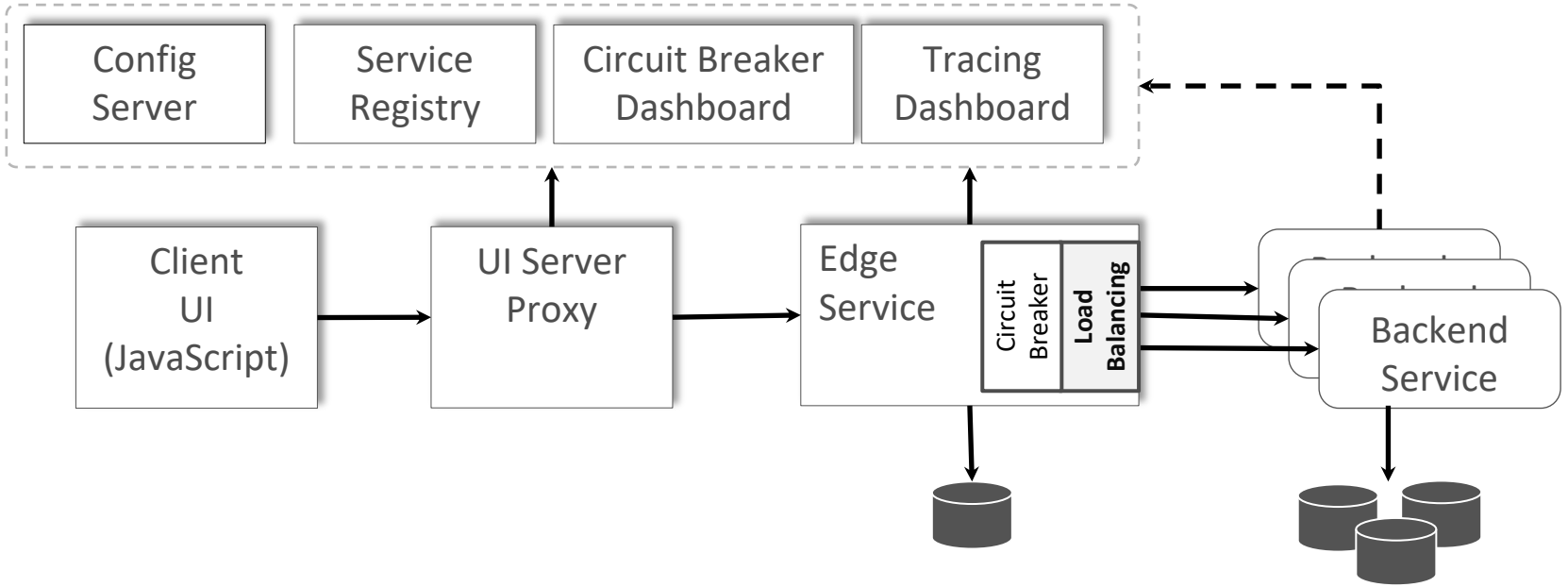
Registration and Discovery



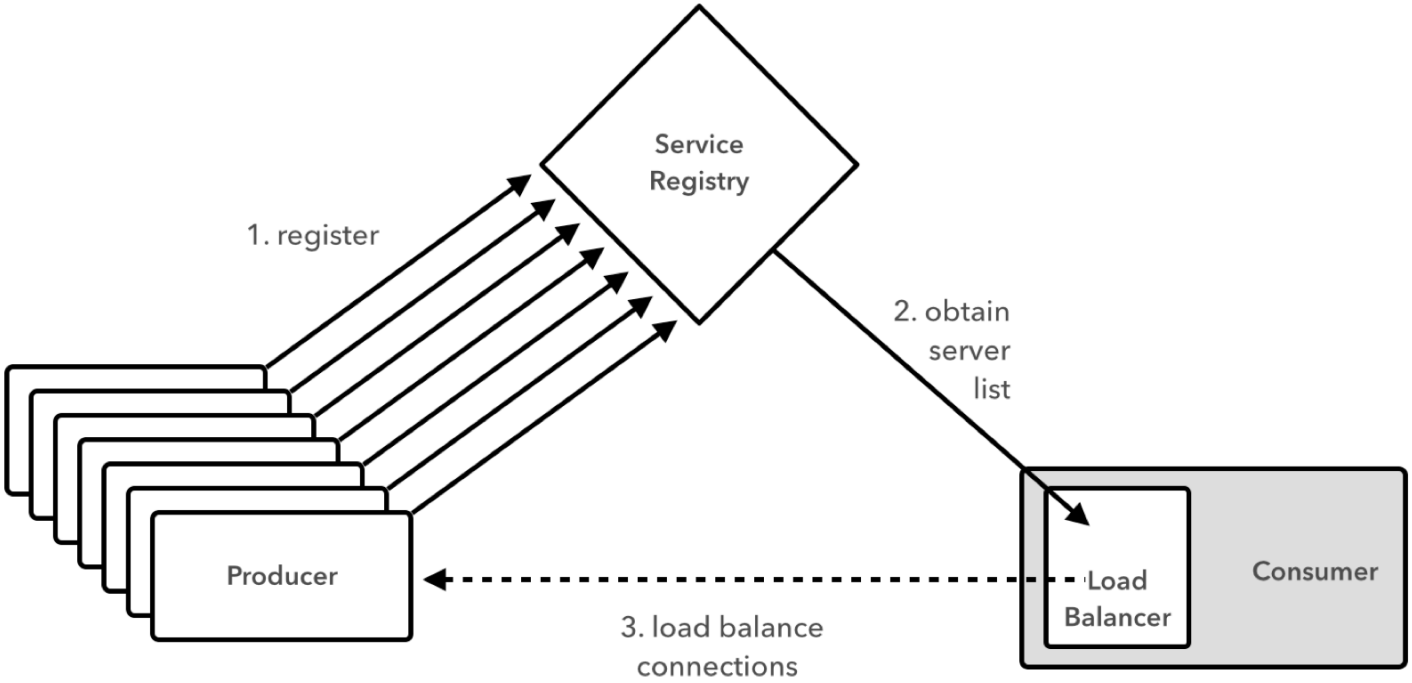
Service Registry



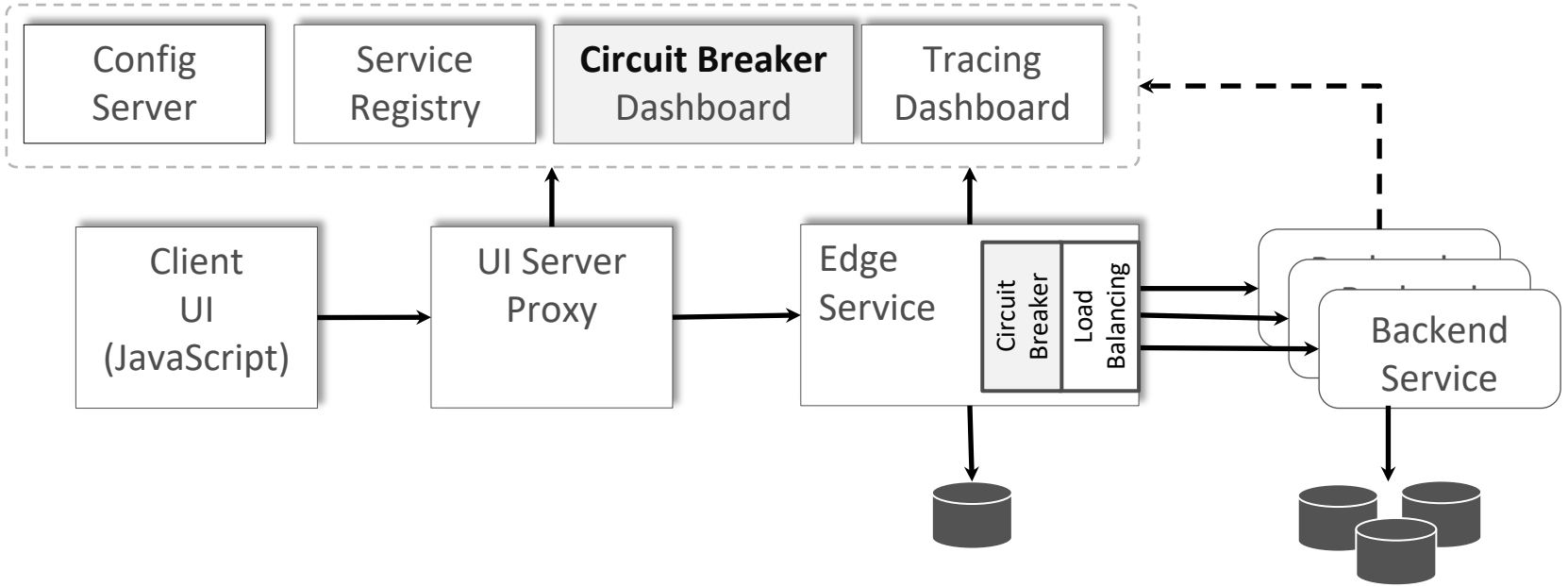
Routing and Load Balancing



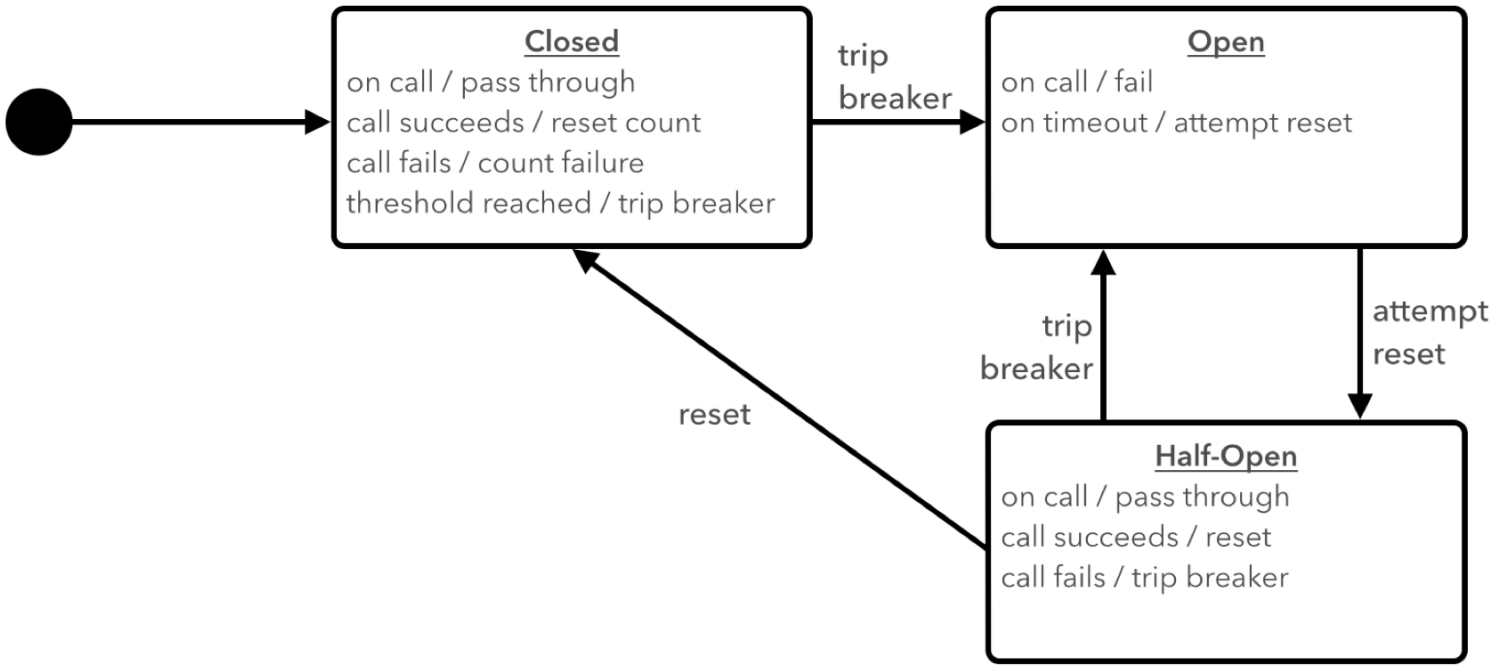
Load Balancing (client-side)



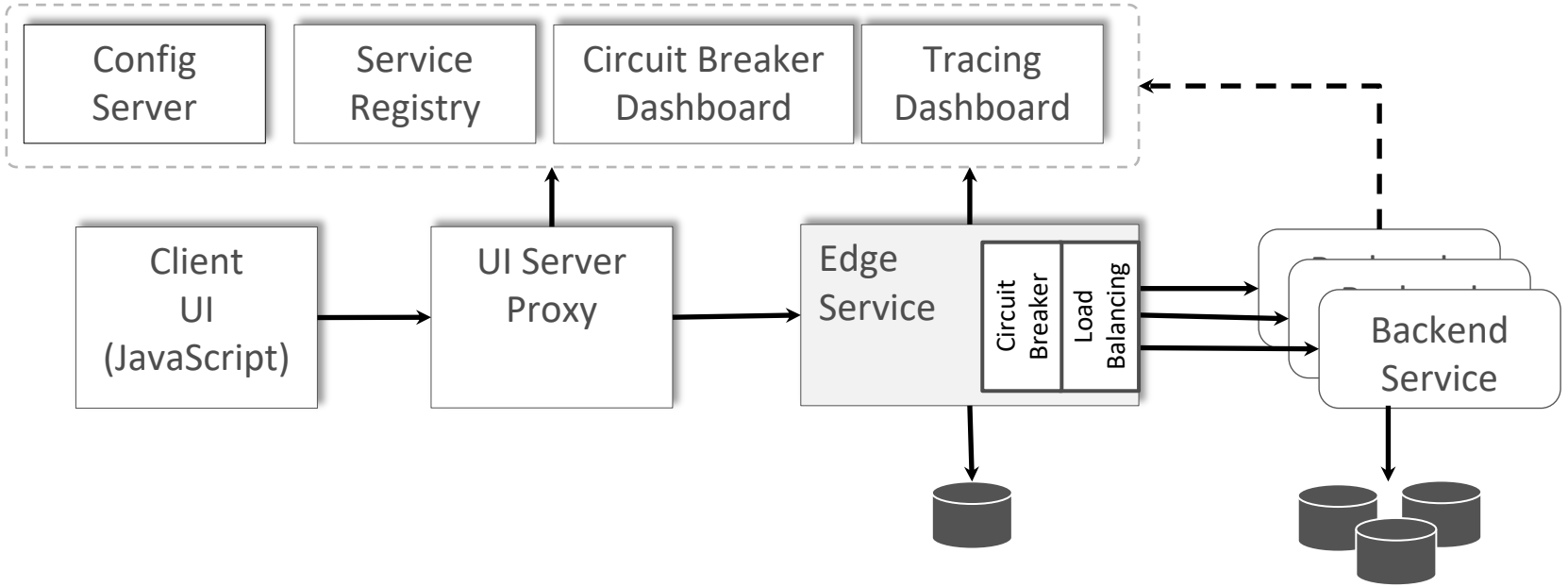
Fault Tolerance and Isolation



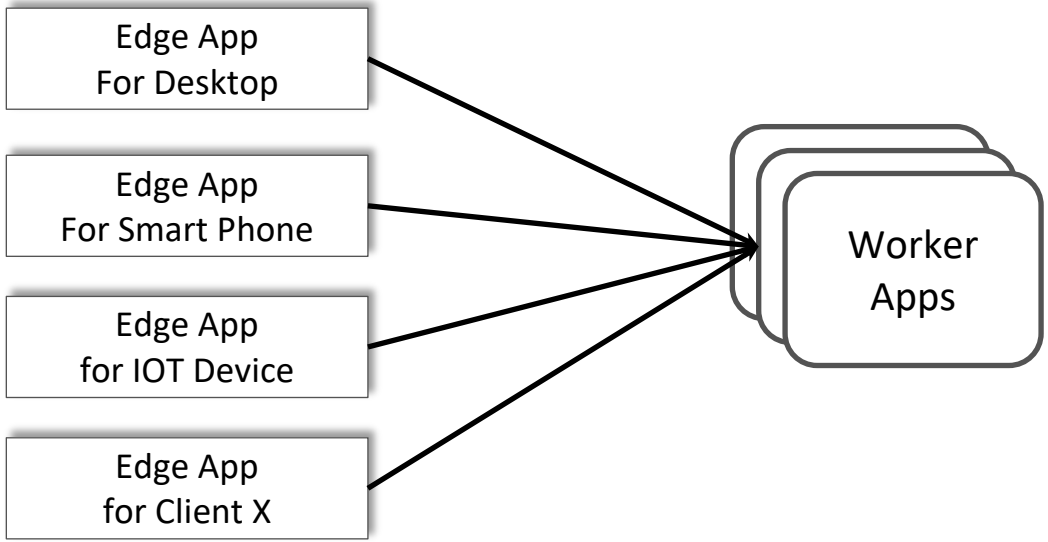
Circuit Breaker



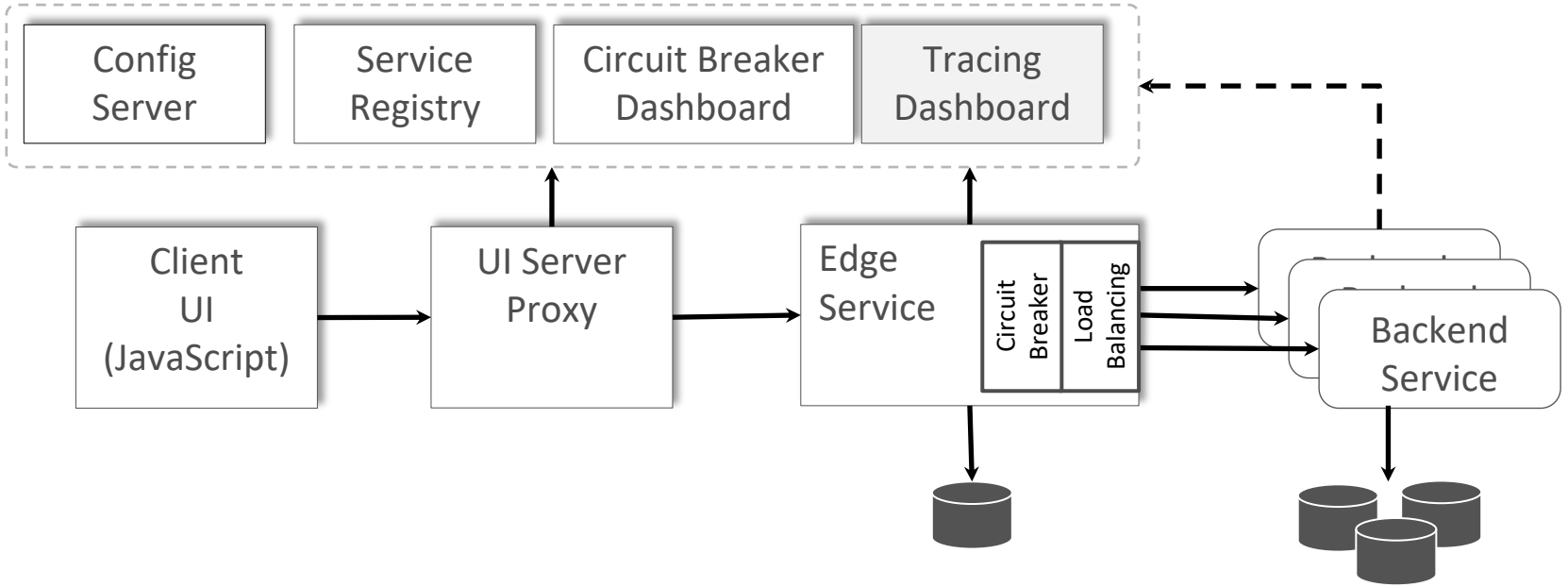
Aggregation and Transformation



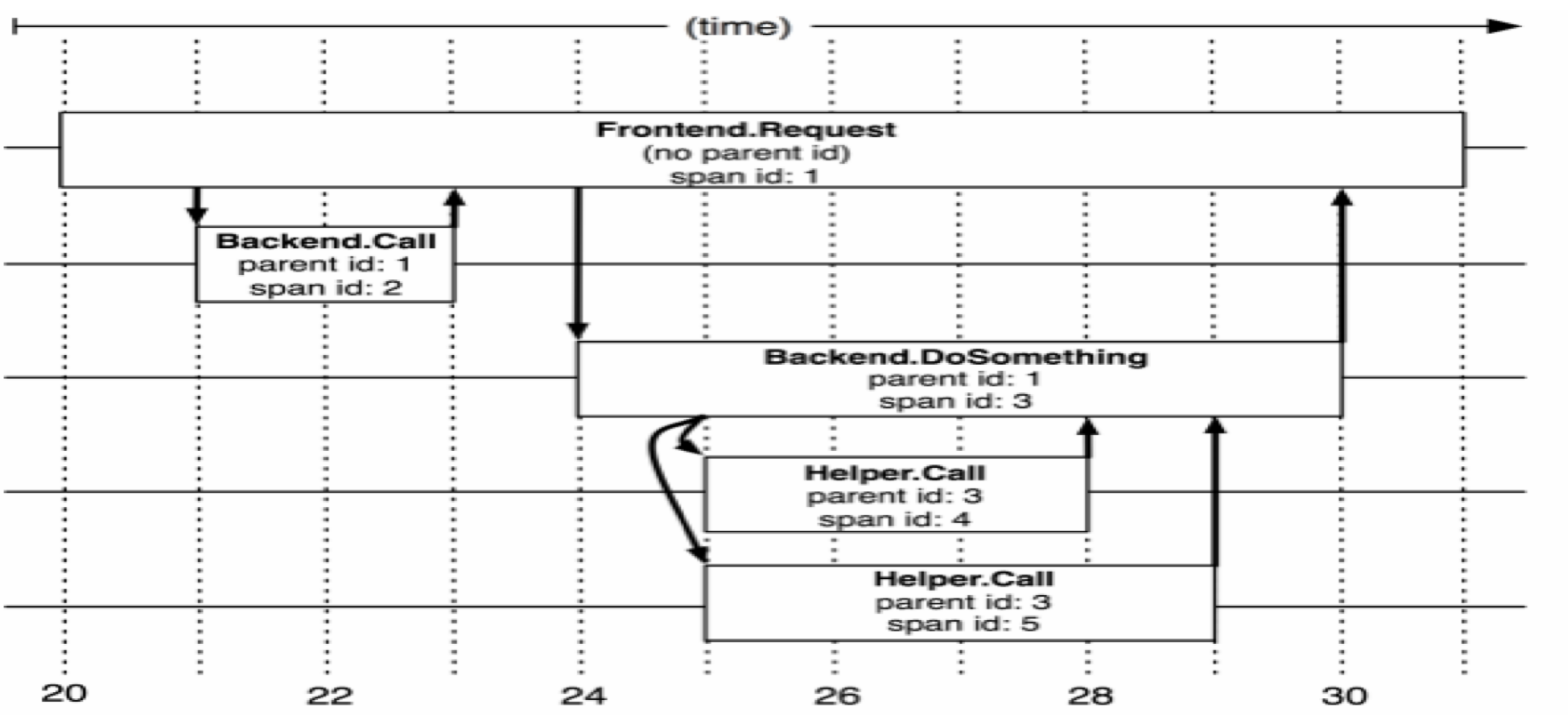
API Gateway


Edge App
For Desktop
Edge App
For Smart Phone
Edge App
for IOT Device
Edge App
for Client X
Worker
Apps

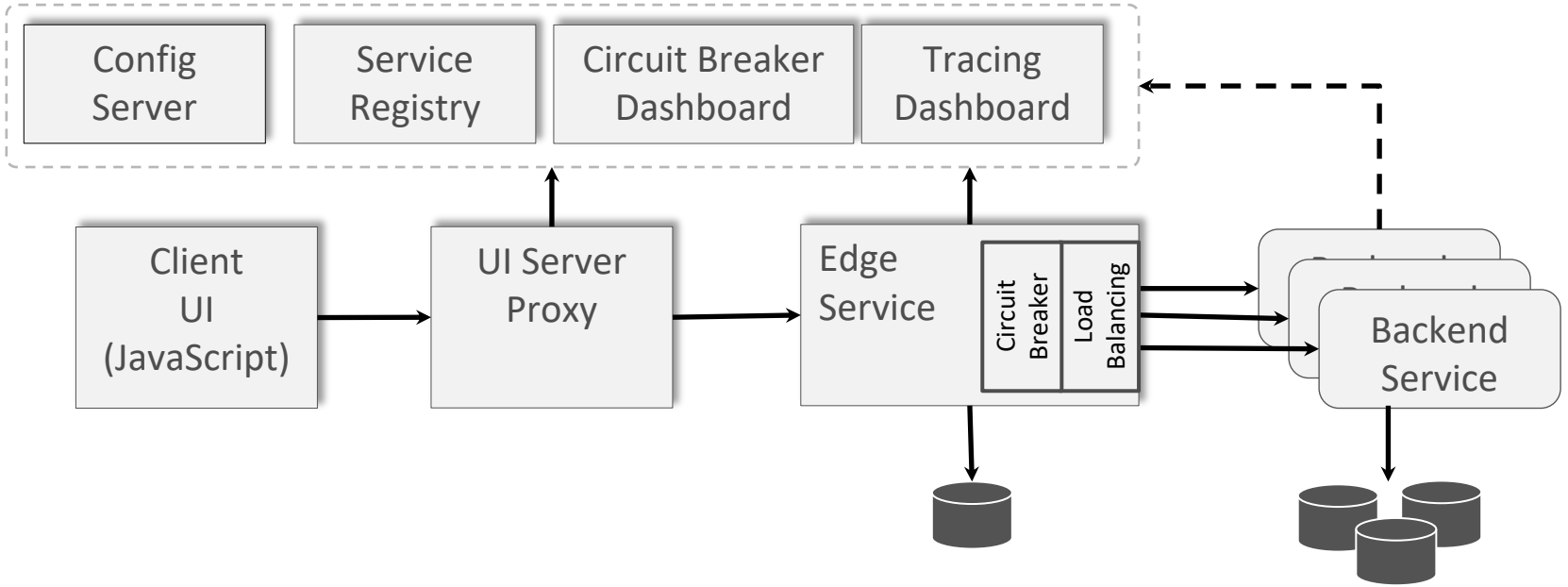
Monitoring and Distributed Tracing



Distributed Tracing



Process Management



Structured Automation

- Groundwork for devops
- Self-service
- Rapid, automated provisioning
- Predictability and consistency
  - not ad-hoc!
- Visibility (monitoring & metrics)
- Continuous delivery on Day 2, too!

Cloud Native Architecture

Why

- Cloud Computing
- New Demands
- Being Reactive

What

- Reactive Design
- 12Factor apps
- Microservices

How

Spring I/O

- Spring Boot
  - How you build an app
- Spring Cloud
  - Pieces for Cloud Native
- Spring Core

Enterprise Cloud Native Frameworks

Development

Spring Boot

Building cloud native software at a consistent speed and quality.

PRACTICES

- Opinionated
- Bootstrap
- Convention over Configuration
- Prod Ready
- Ops Friendly
- Most popular

Services

Spring Cloud

For managing cloud native distributed micro services

PRACTICES

- Service Discovery
- Configuration Management
- Latency & Fault tolerant
- Intelligent Routing
- Load Balancing
- and More

Spring Cloud Netflix Components

Ribbon

Archaius

Hystrix

Feign

Zuul

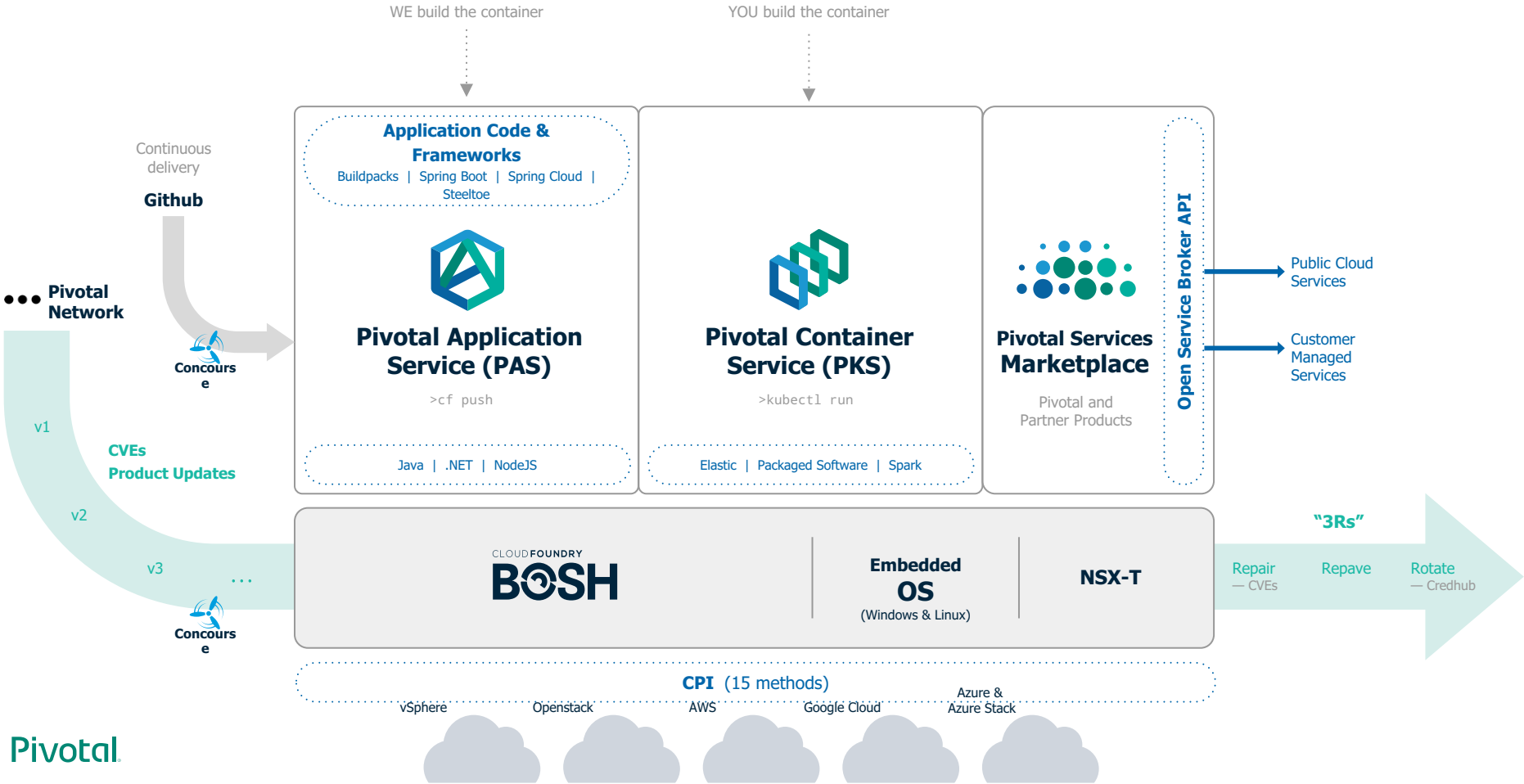
Client

Eureka

Turbine

Server

Pivotal Cloud Foundry



Pivotal

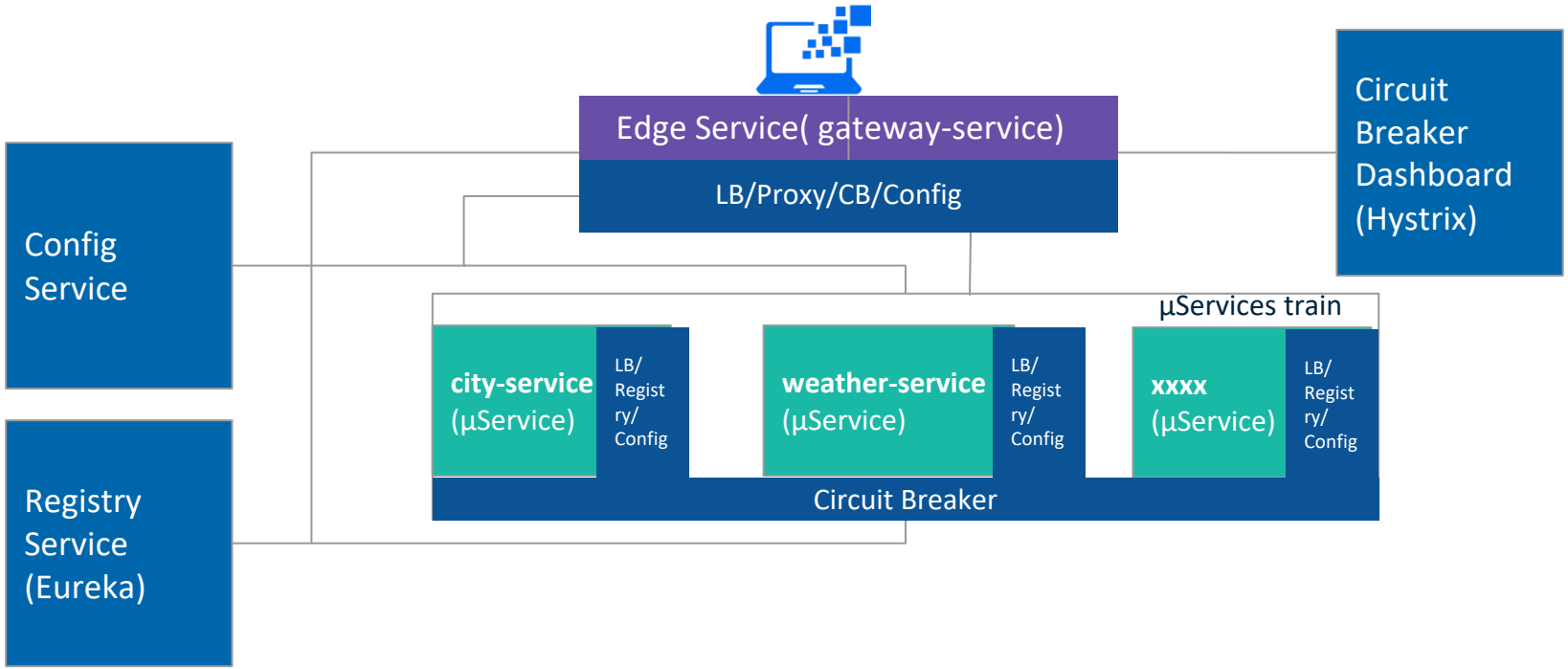
The Journey: Technology

- Provide the right tools
  - Platform, design pattern foundations, continuous delivery…
- Start small
  - Right-size for your needs
  - For migrations, don’t boil the ocean
- Target applications that:
  - Have high business/competitive impact
  - Need to scale or change frequently

The Journey: Culture & Organization

- Small, cross-functional teams
- Products, not projects
- Test-driven development
- Short iterations
- Transparency and visibility

Hi Cloud Native App!



Transforming How The World Builds Software