

Pivotal.

# Cloud Native Architecture 300



# Cloud Native Architecture

---



Why

What

How

# Cloud Native Architecture

Why

What

How

- Cloud Computing
- New Demands
- Being Reactive

# Cloud Native Architecture

Why

- Cloud Computing
- New Demands
- Being Reactive

What

- Reactive Design
- 12 Factor Apps
- Microservices
- Design Patterns

How

# Cloud Native Architecture

Why

- Cloud Computing
- New Demands
- Being Reactive

What

- Reactive Design
- 12 Factor Apps
- Microservices
- Design Patterns

How

- Tools
- Platform
- Process and Culture

# Cloud Native Architecture

Why

What

How

- Cloud Computing
- New Demands
- Being Reactive

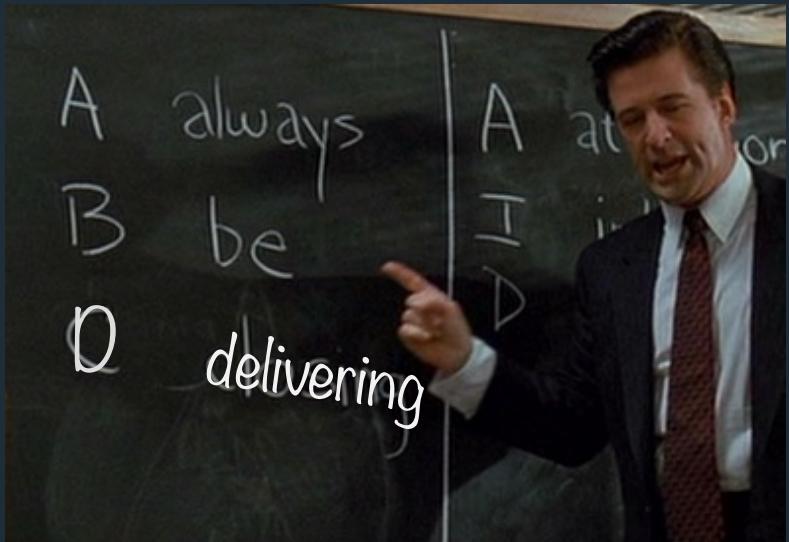
# Cloud Defined

---

- 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

# Speed & Safety

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



# Cloud Native Architecture

Why

What

How

- Reactive Design
- 12 Factor Apps
- Microservices
- Design Patterns

# Reactive Design

---

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

# 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

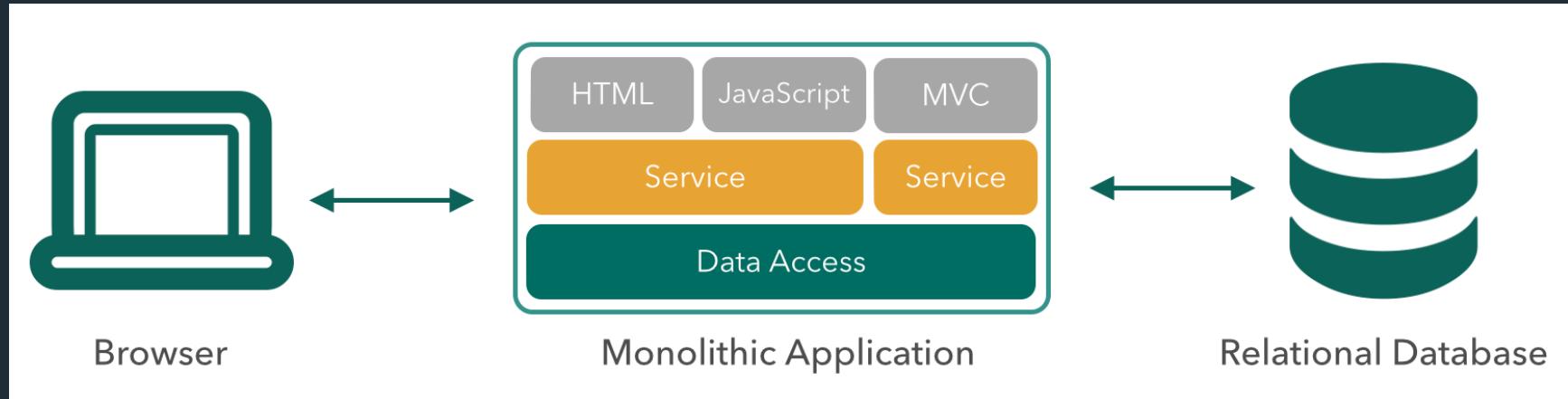
If services have to be updated together,  
they're not loosely coupled!

## Loosely coupled service oriented architecture with bounded contexts

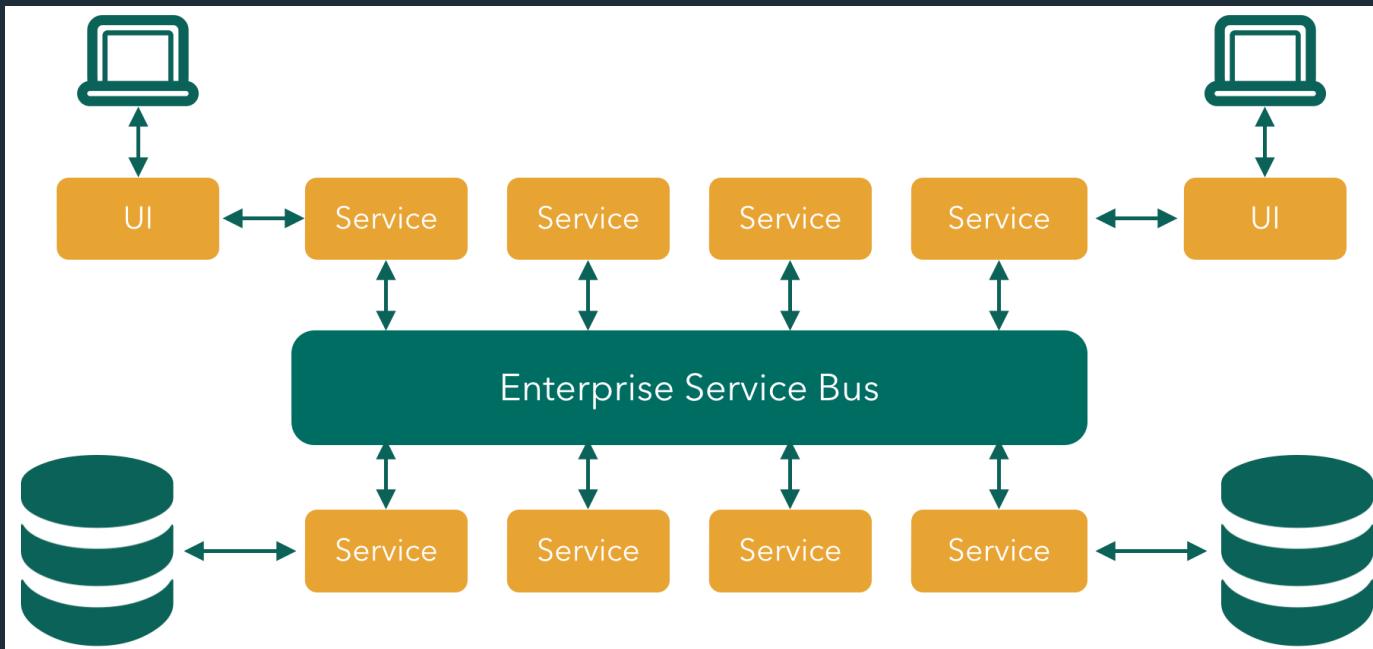
- Adrian Cockcroft

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

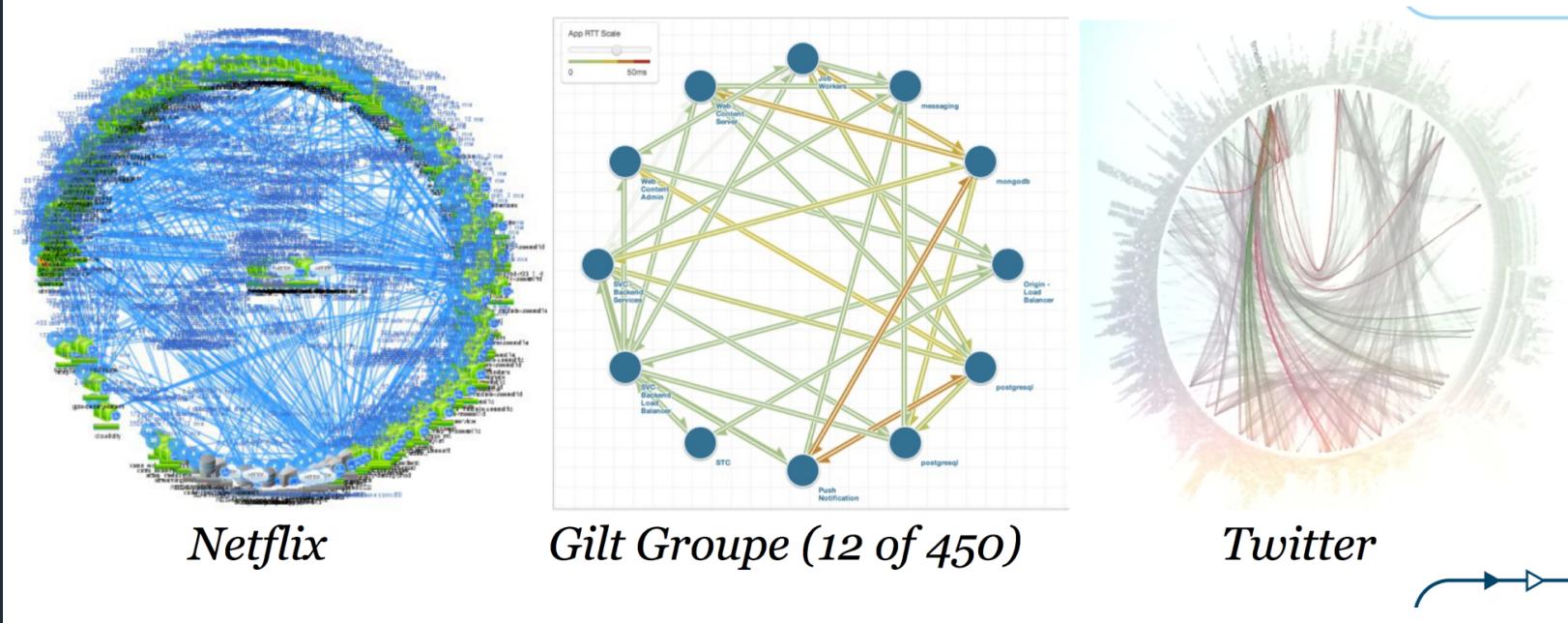
# Not Monoliths



# Not ESB-Centric SOA



# Microservice Pioneers



Pivotal™

# Who Moved My Complexity?

---

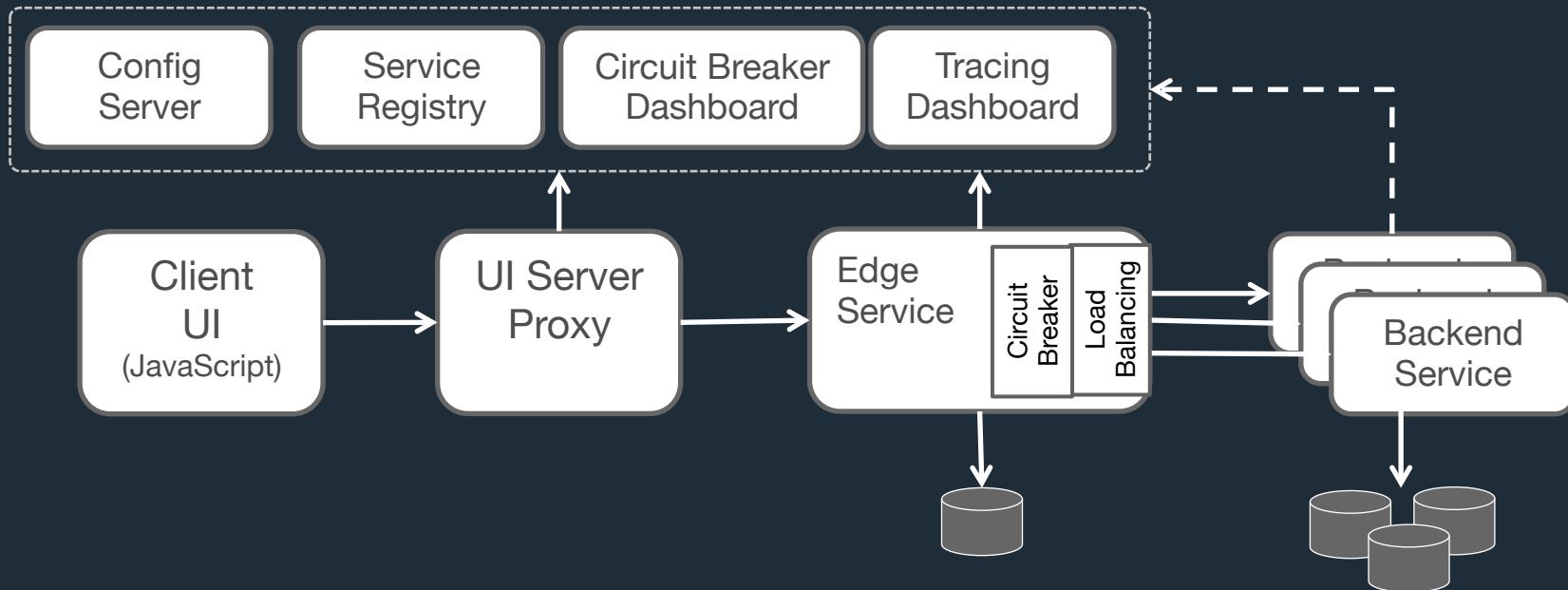
- Microservices are individually simple
- Complexity is transferred to the ecosystem

# Challenges in a Distributed System

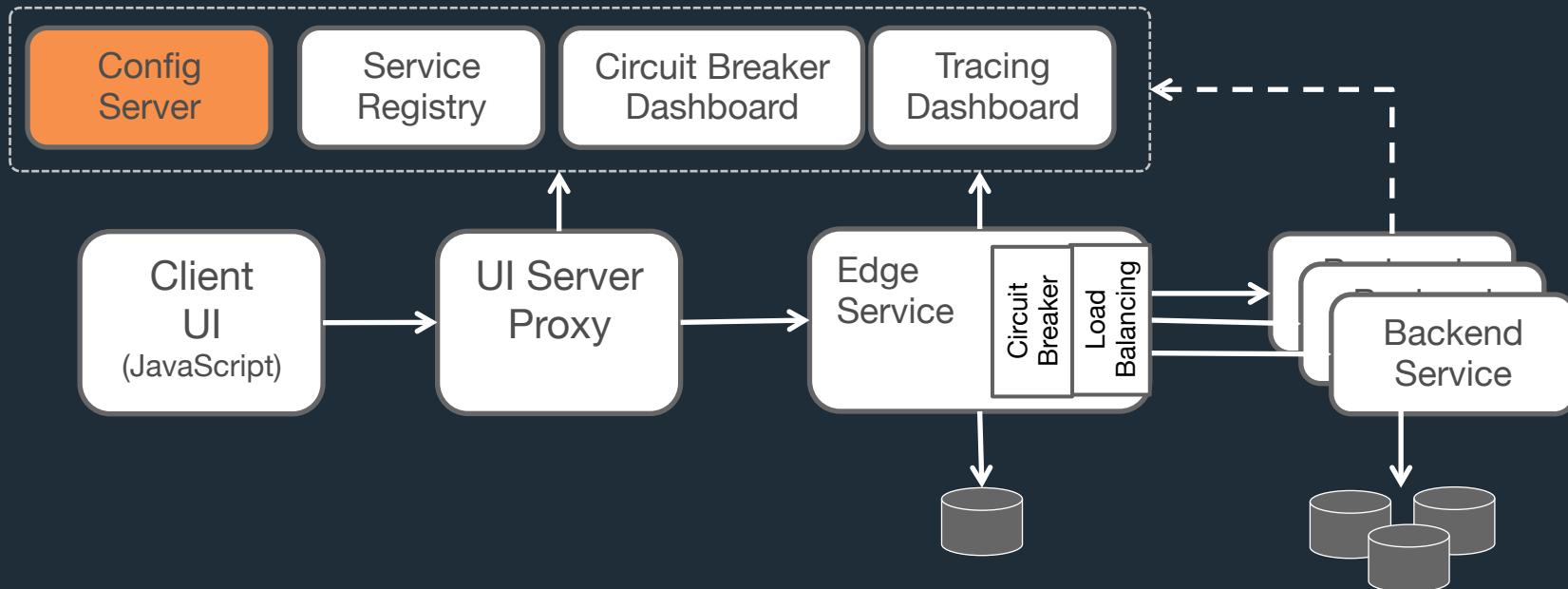
---

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

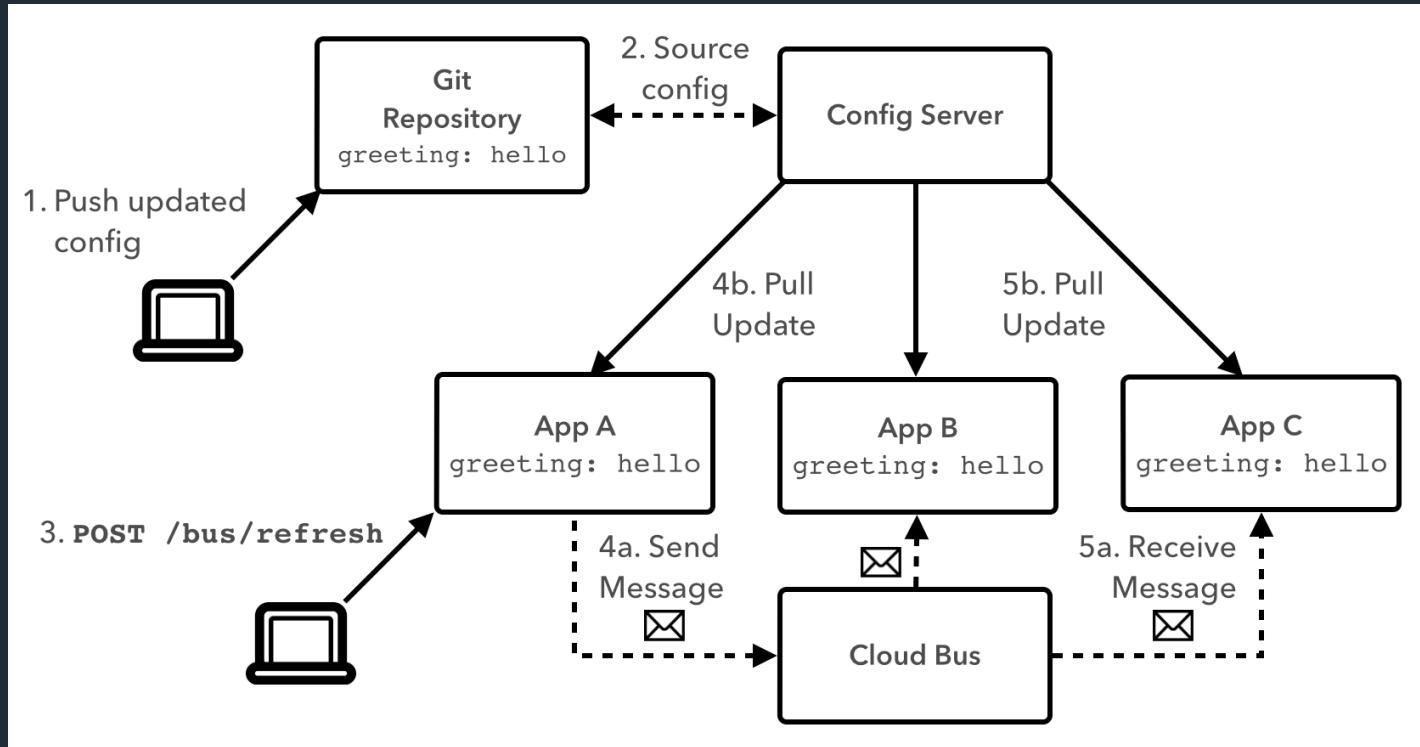
# Design Patterns & Automation to the Rescue!



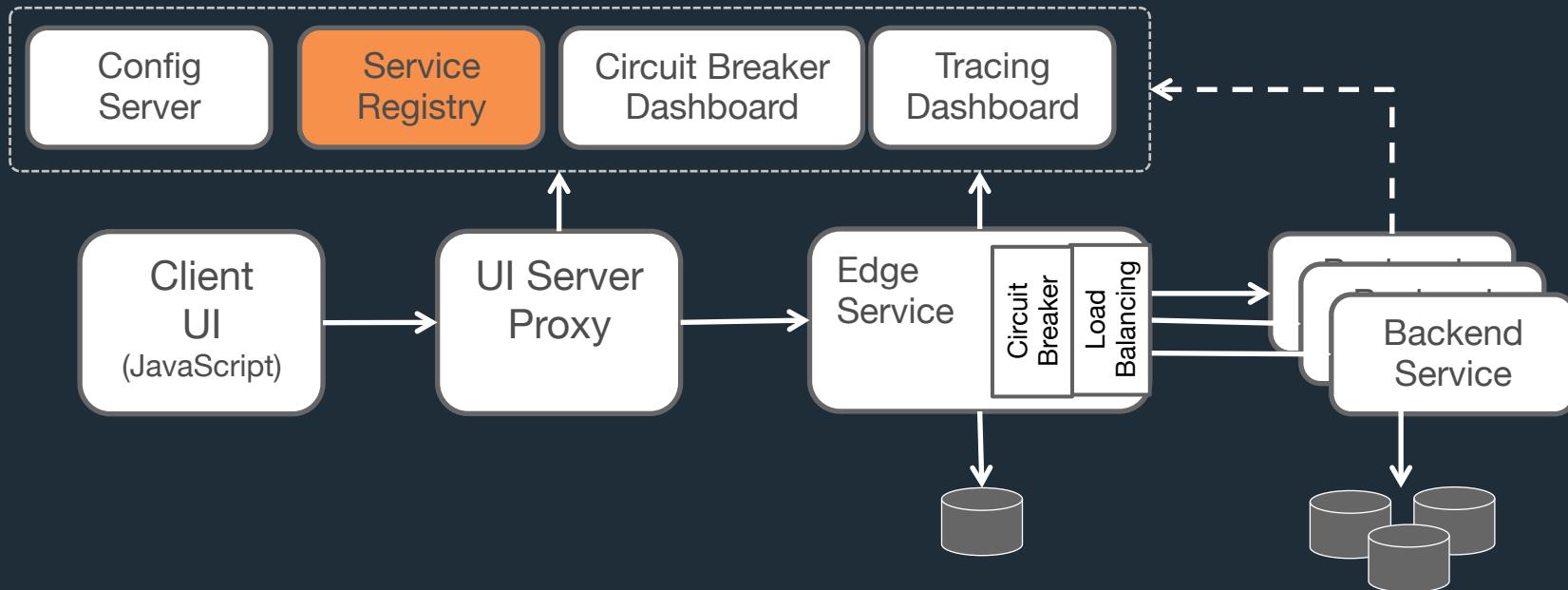
# Configuration Management



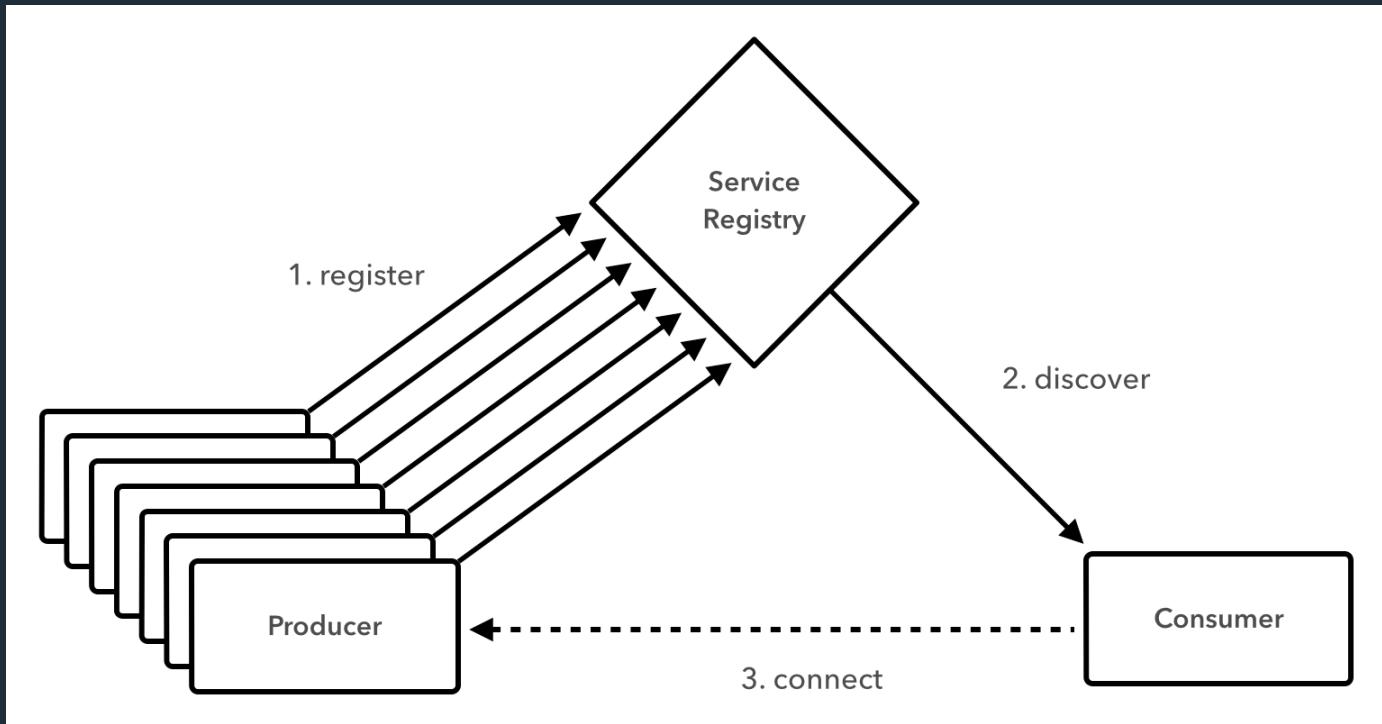
# Configuration Server



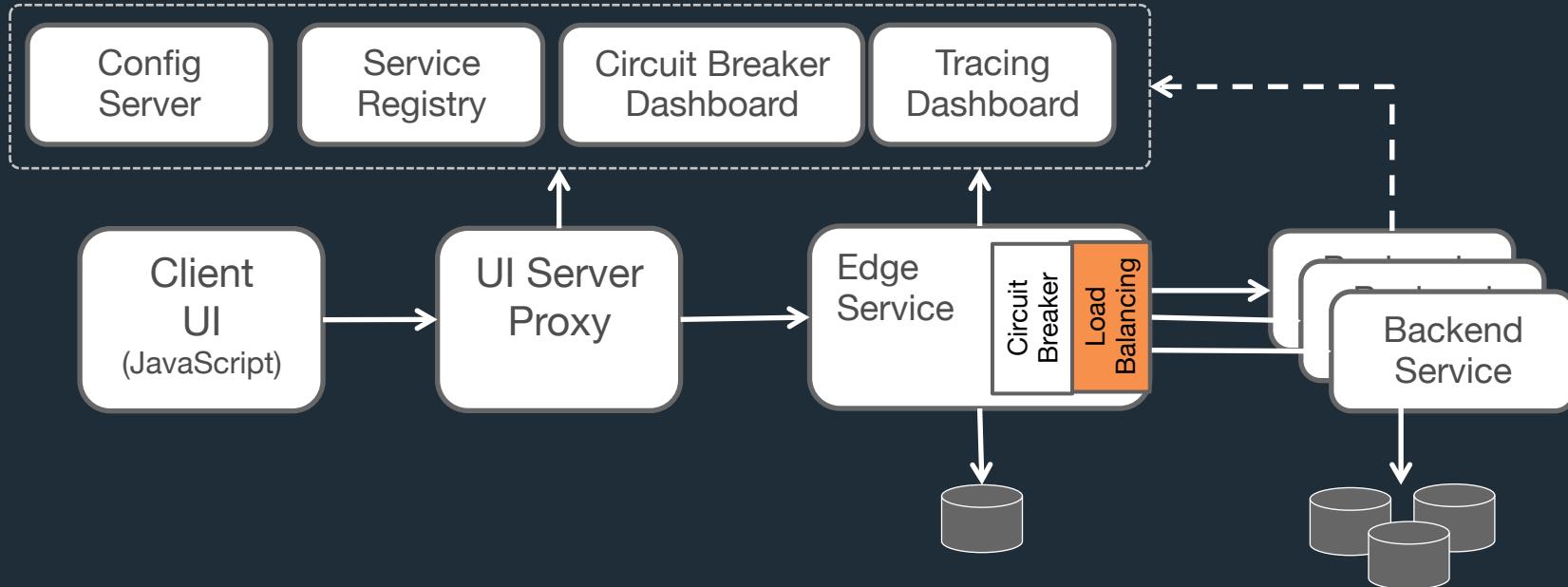
# Registration and Discovery



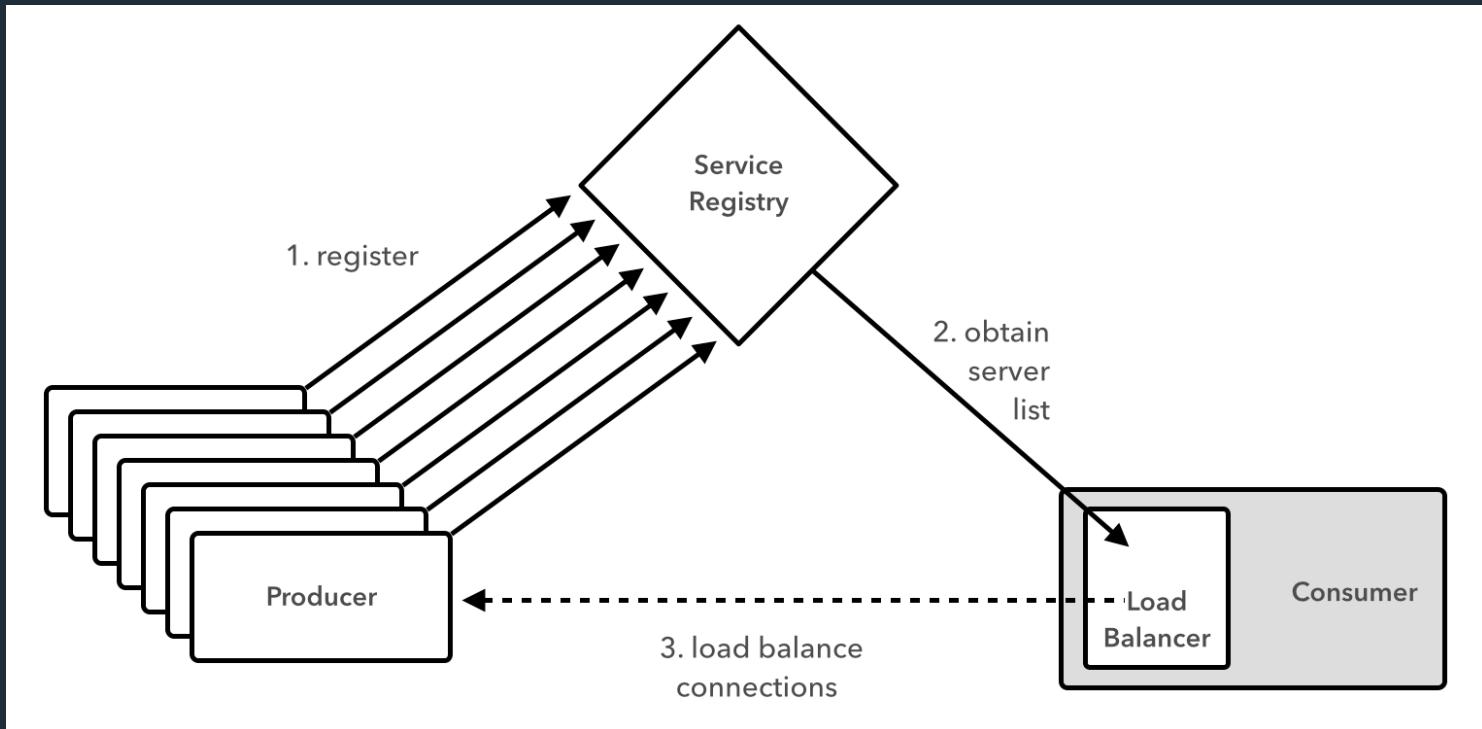
# Registration and Discovery Server



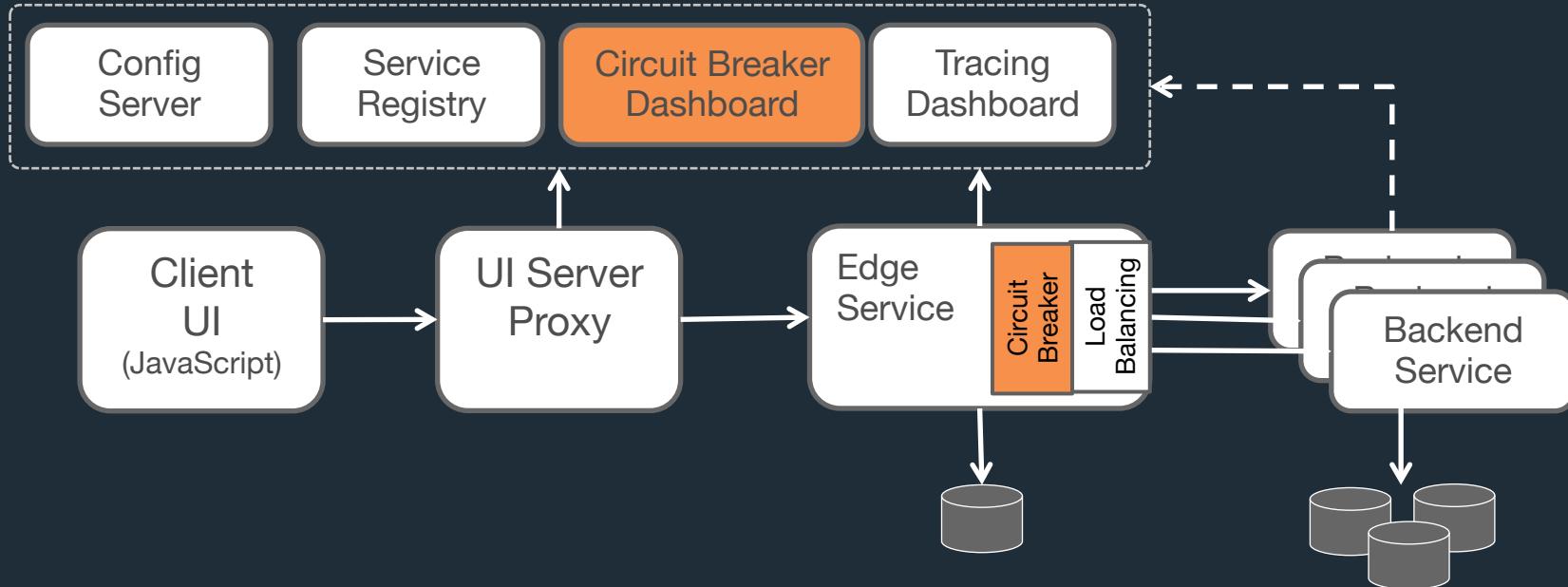
# Routing and Load Balancing



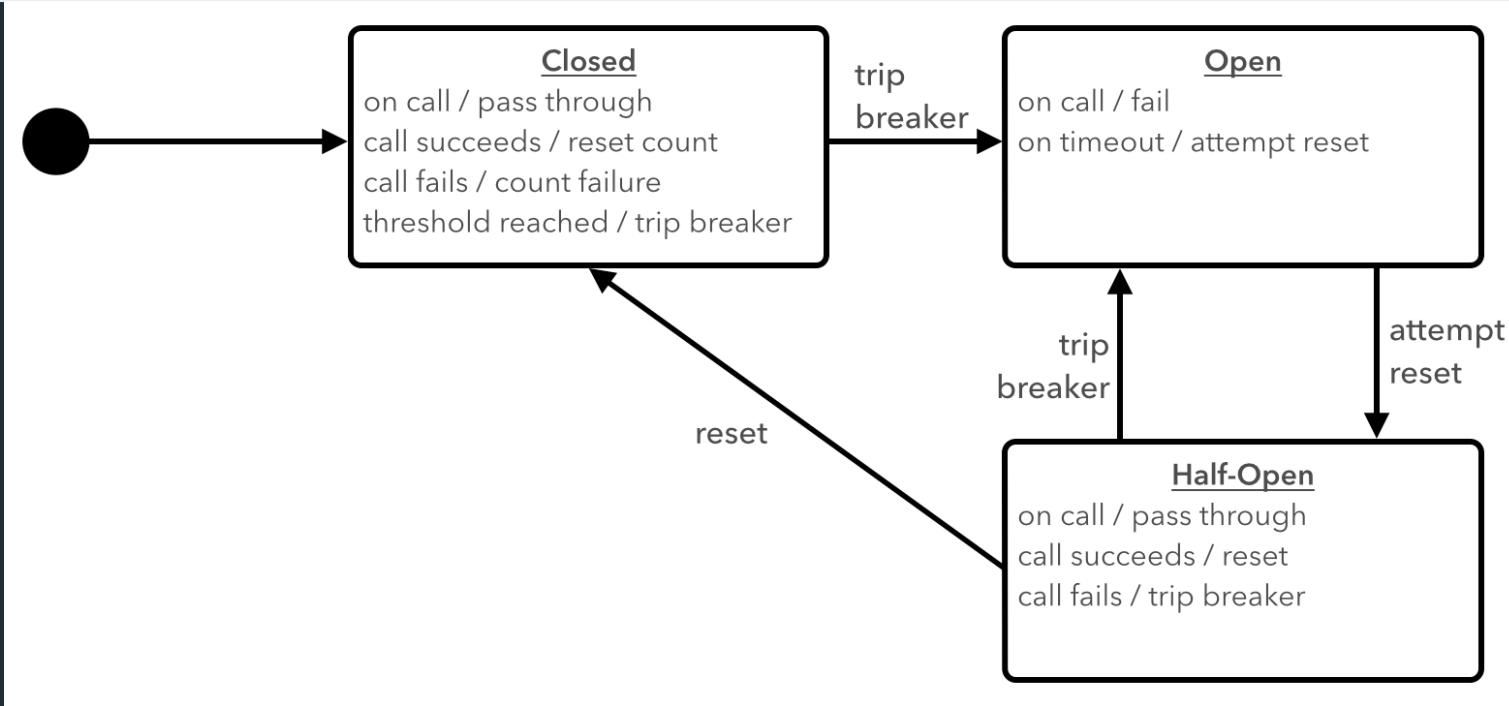
# Client-Side Load Balancing



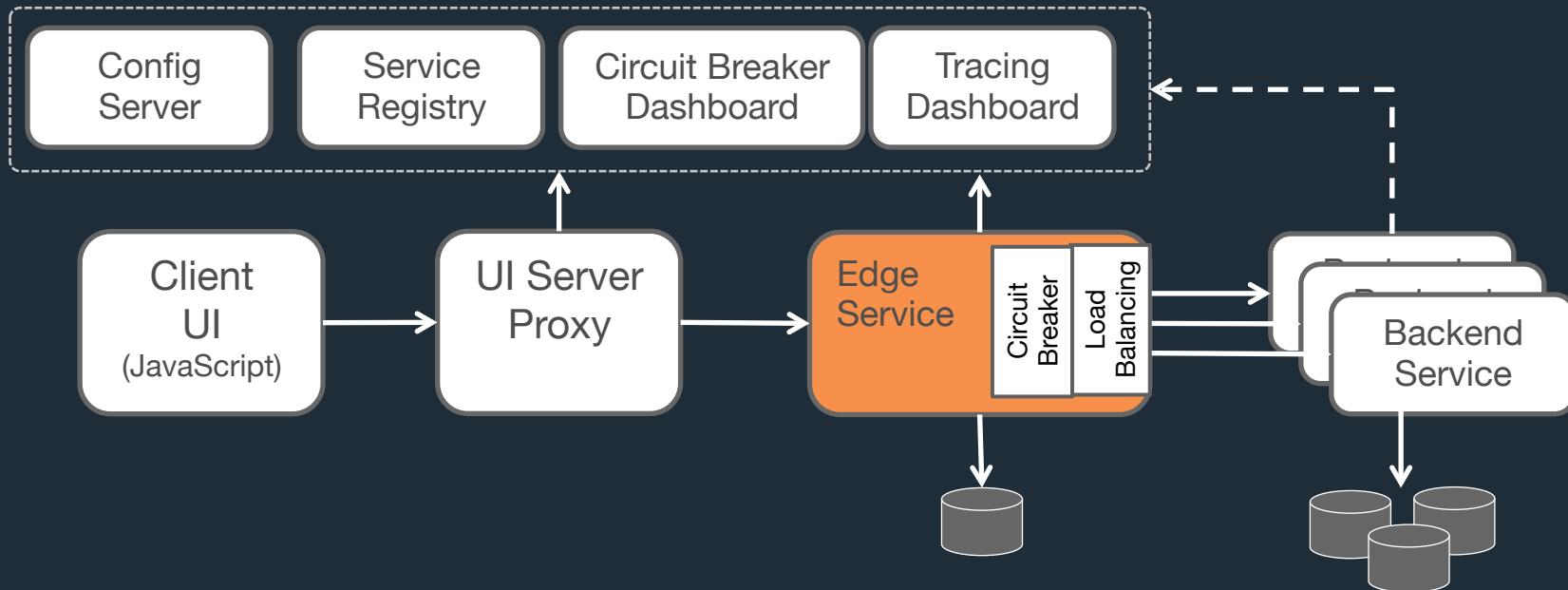
# Fault Tolerance and Isolation



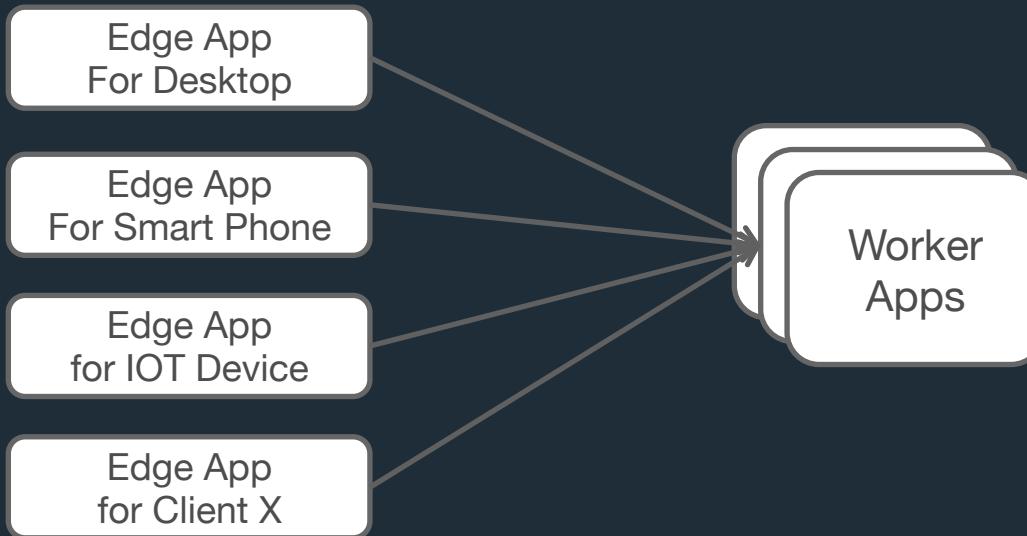
# Circuit Breaker



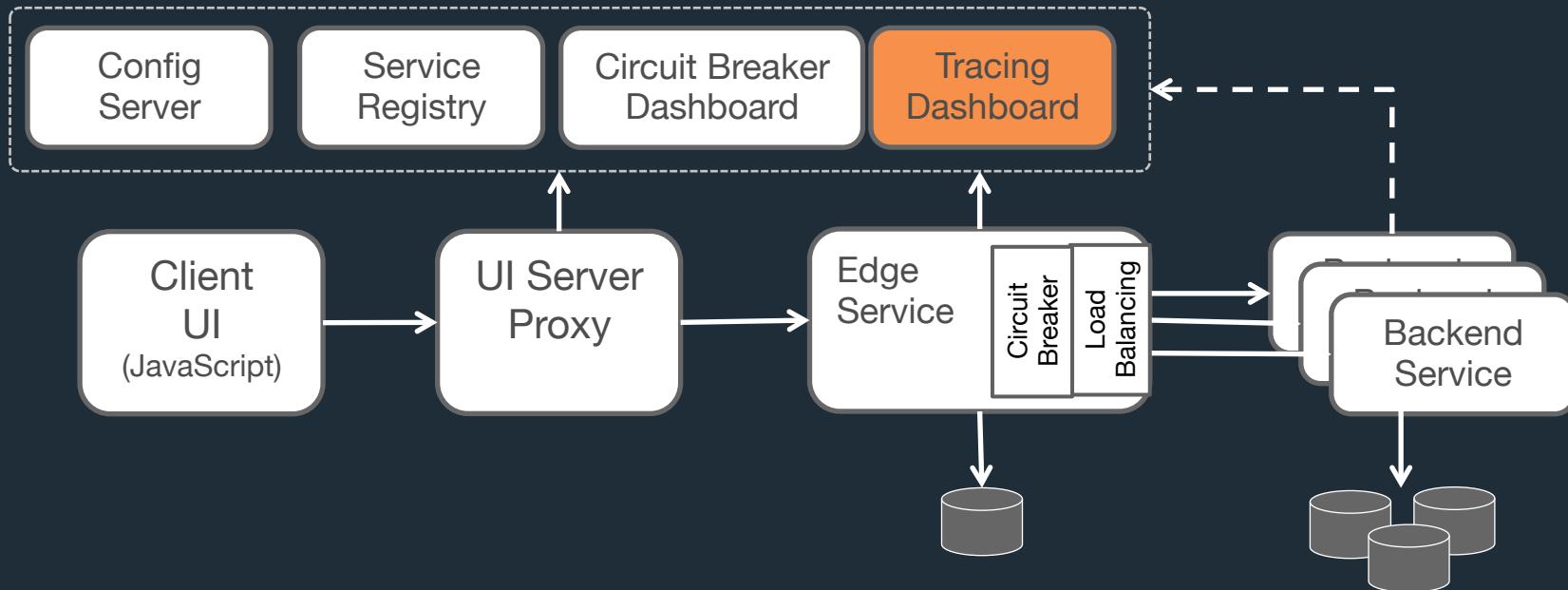
# Aggregation and Transformation



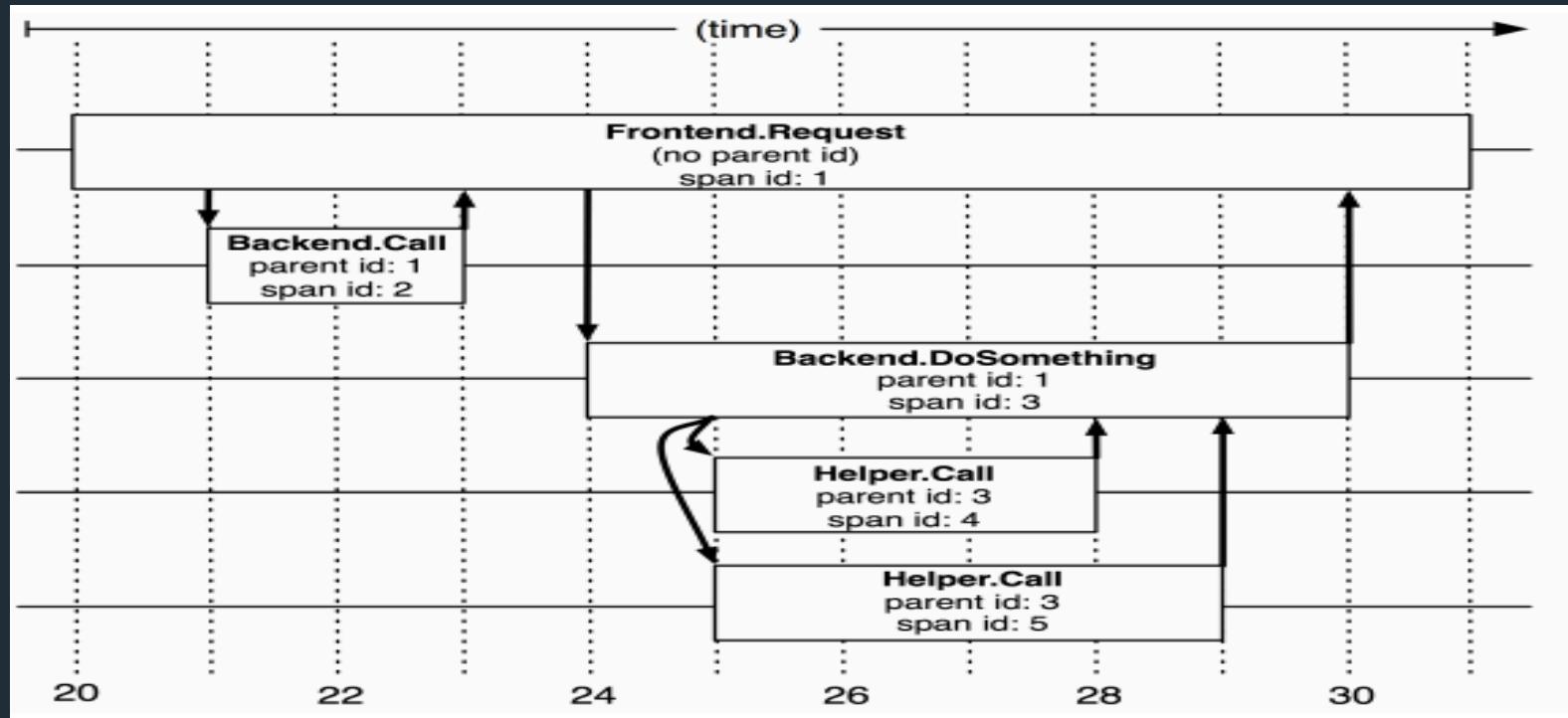
# API Gateway



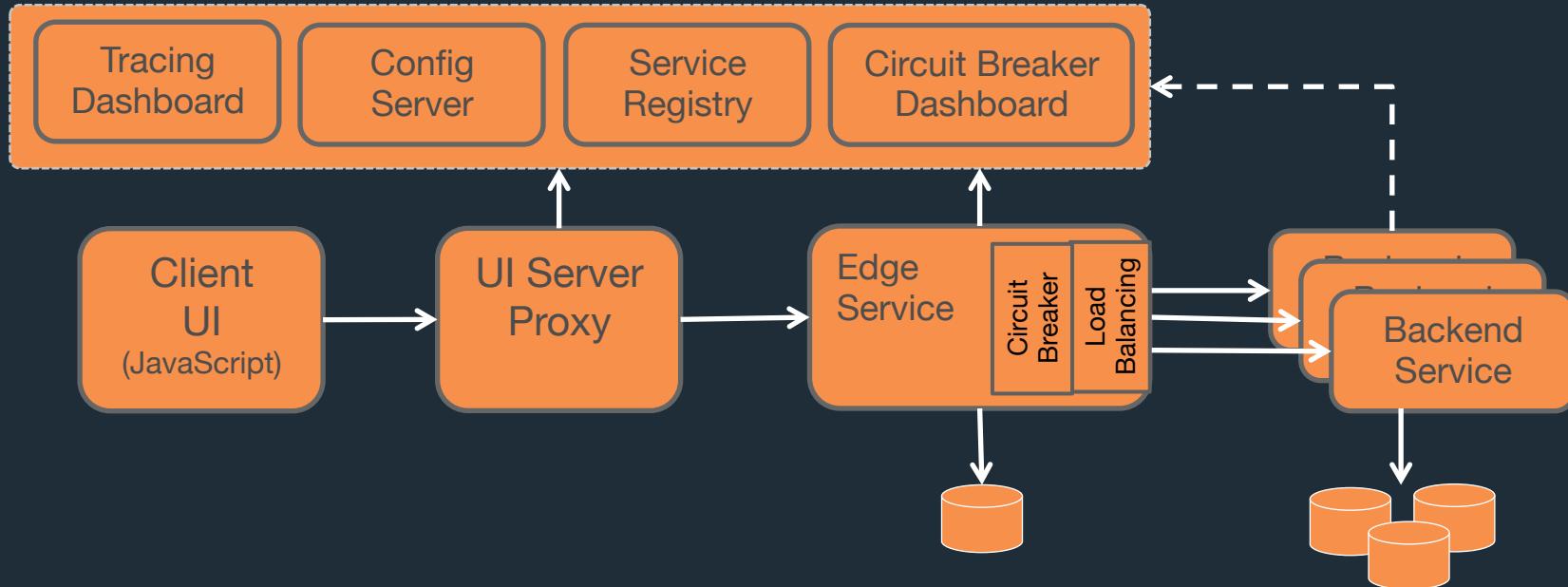
# Monitoring and Distributed Tracing



# 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

What

How

- Tools
- Platform
- Process and Culture

# Spring I/O

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



# Spring Boot

---

- “Microframework”
- Opinionated
- Convention over configuration
- Production ready
- Ops Friendly
  - Self contained
  - Health and metrics
  - Externally configurable

# Spring Cloud

---

- Distributed/versioned configuration
- Service registry and discovery
- Routing
- Service-to-service calls
- Load balancing
- Circuit Breakers
- More...

# Spring Cloud Netflix Components



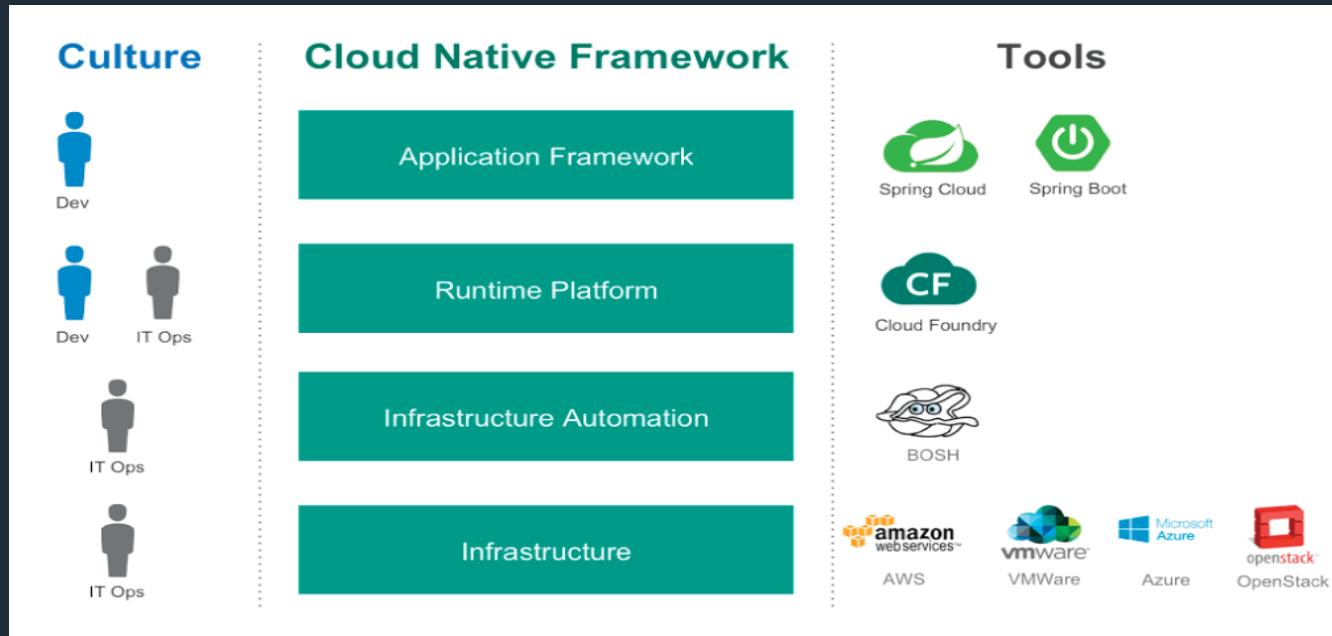
Client



Server

Pivotal™

# Pivotal Cloud Foundry



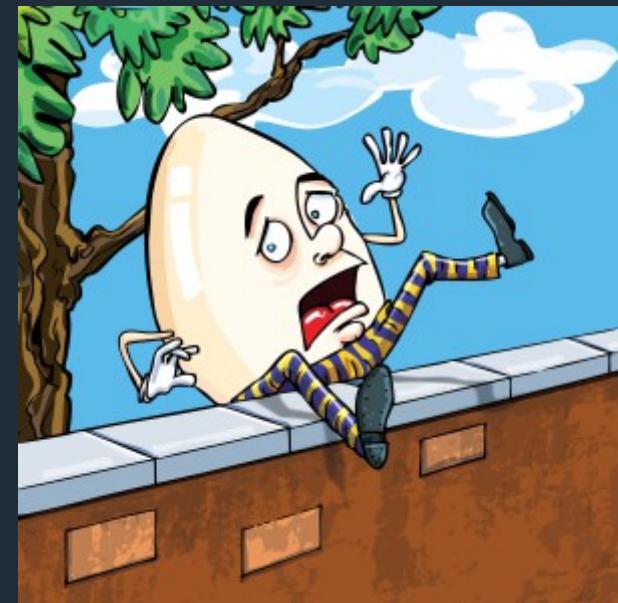
# The Journey: Technology

---

- 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



# Cloud Native Architecture

Why

- Cloud Computing
- New Demands
- Being Reactive

What

- Reactive Design
- 12 Factor Apps
- Microservices
- Design Patterns

How

- Tools
- Platform
- Process and Culture

# Pivotal.

# Thank You



# Open. Agile. Cloud-Ready.

Pivotal.