Building a typical Spring Cloud architecture application

Timo Salm

Specialist Solution Engineer VMware Tanzu

February 2021

Agenda

- About me
- A typical Spring Cloud architecture application
- A typical Spring Cloud architecture application on Kubernetes

About me

Timo Salm

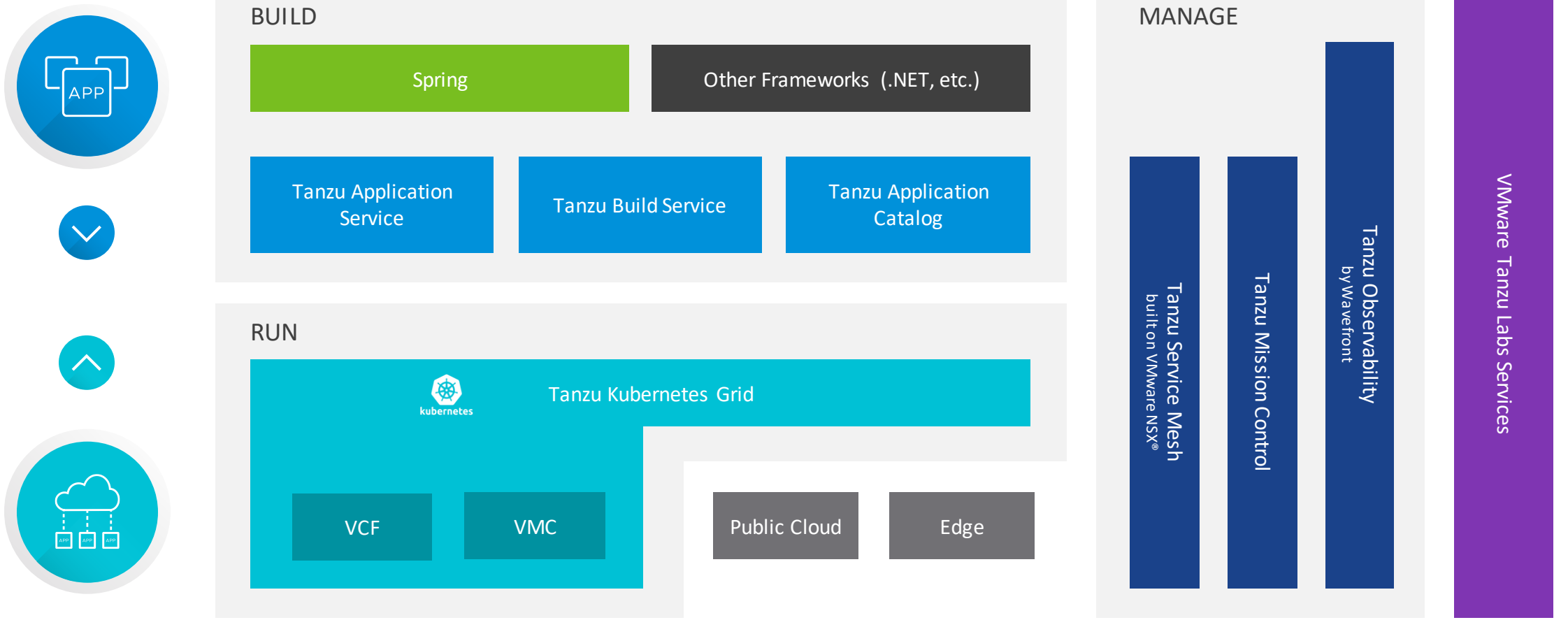
Specialist Solution Engineer for VMware Tanzu

- https://tanzu.vmware.com/
- Twitter: @salmto
- GitHub: https://github.com/tsalm-pivotal

VMware Tanzu

Comprehensive stack to modernize your applications

VMware Tanzu Including VMware Tanzu Labs



Spring by VMware

The Standard for Cloud Native Java

Spring Boot

Build Anything

Spring Cloud

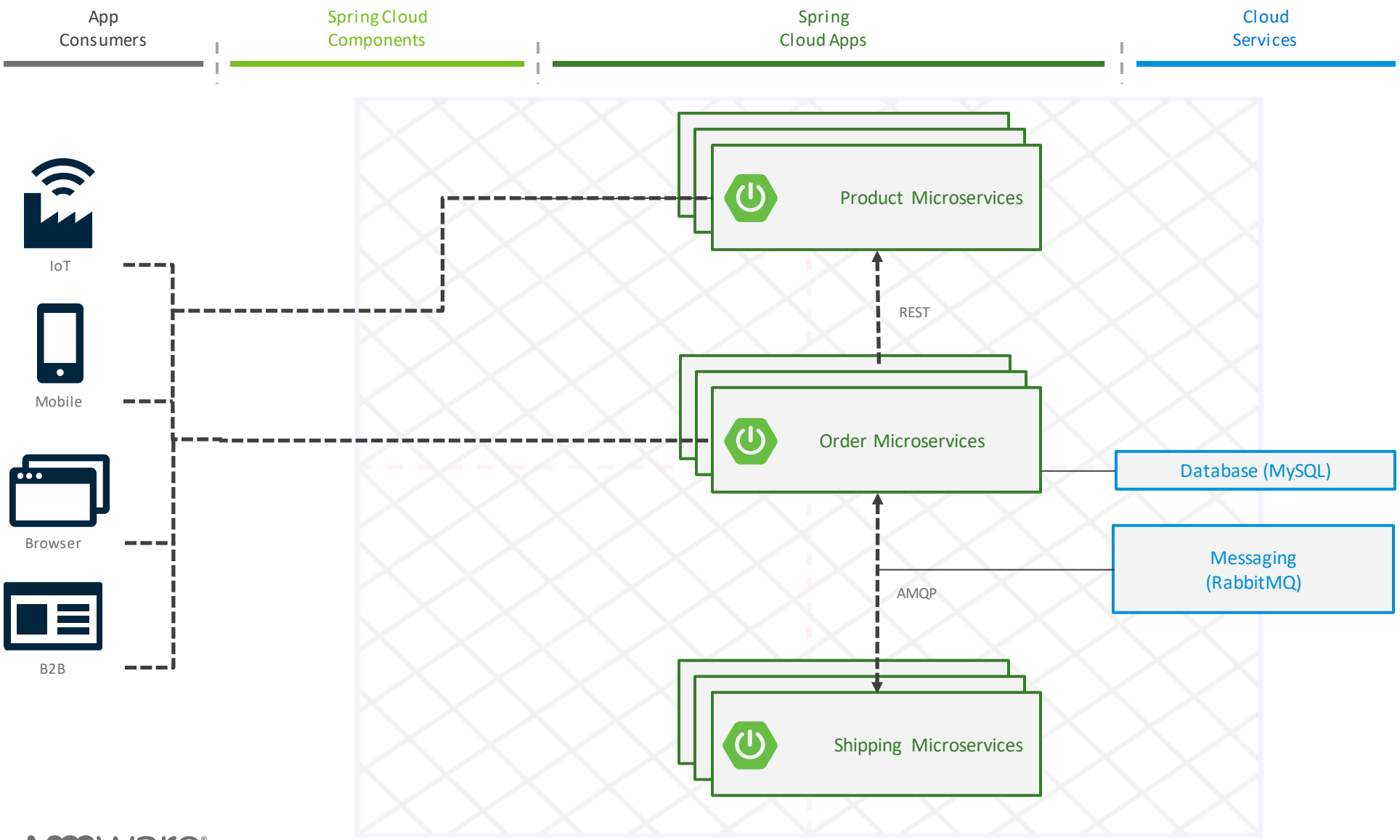
Coordinate Anything

Spring Cloud

Data Flow
Connect Anything

Code Clarity | Lower Complexity | Less Tech Debt |
Focus on Business Logic | Better Test Coverage | Faster Code Completion

A typical Spring Cloud architecture



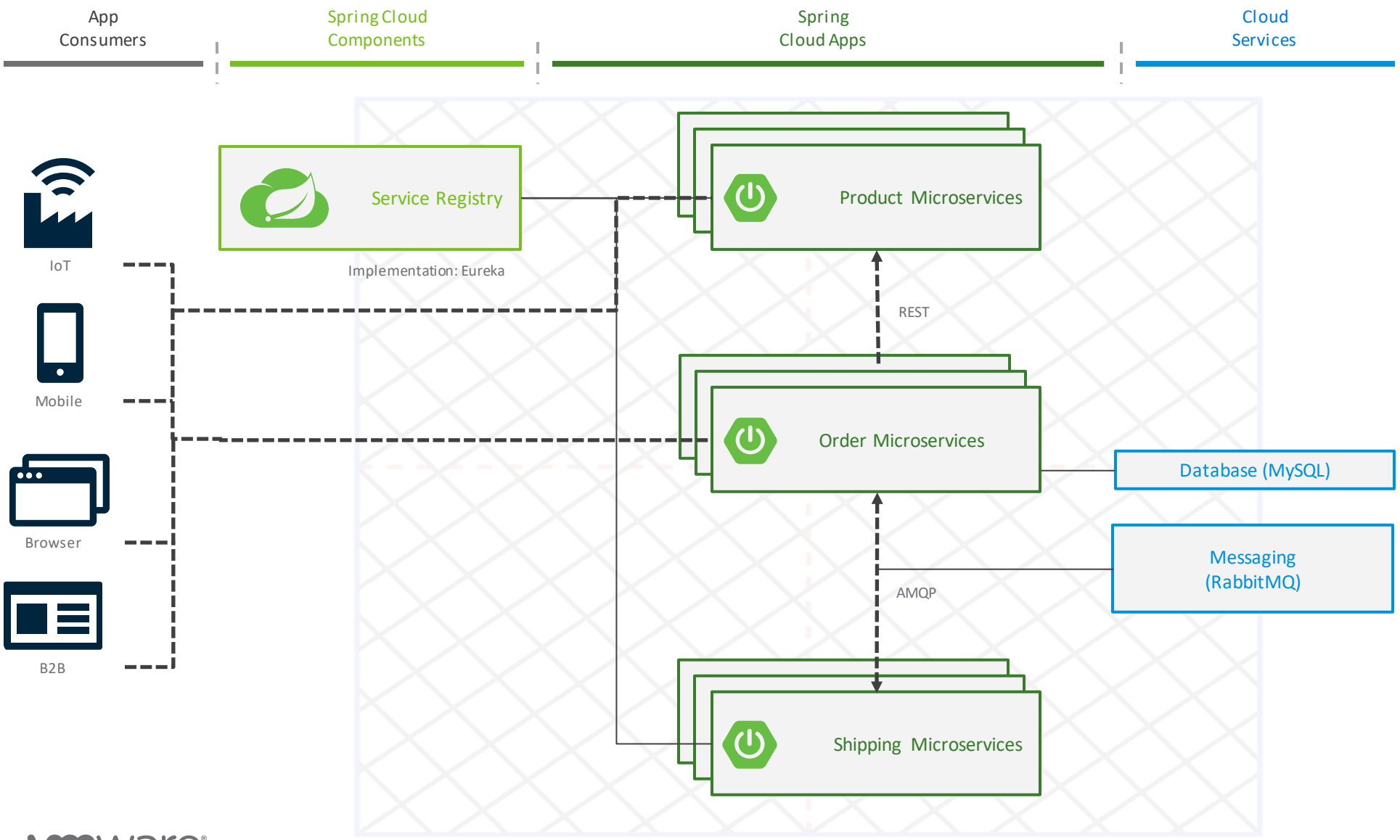
Service Registry

In the cloud, your services move around - their addresses aren't fixed, they will change over time ...

Spring Cloud's Registry interface solves this problem using popular service registry implementations such as Consul, Zookeeper, or Eureka.

Spring Cloud also provides client-side libraries for popular registries like Consul, Zookeeper, Eureka, or Kubernetes

A typical Spring Cloud architecture



All Microservices:
Spring Boot
Spring Cloud Discovery Client

Load Balancer

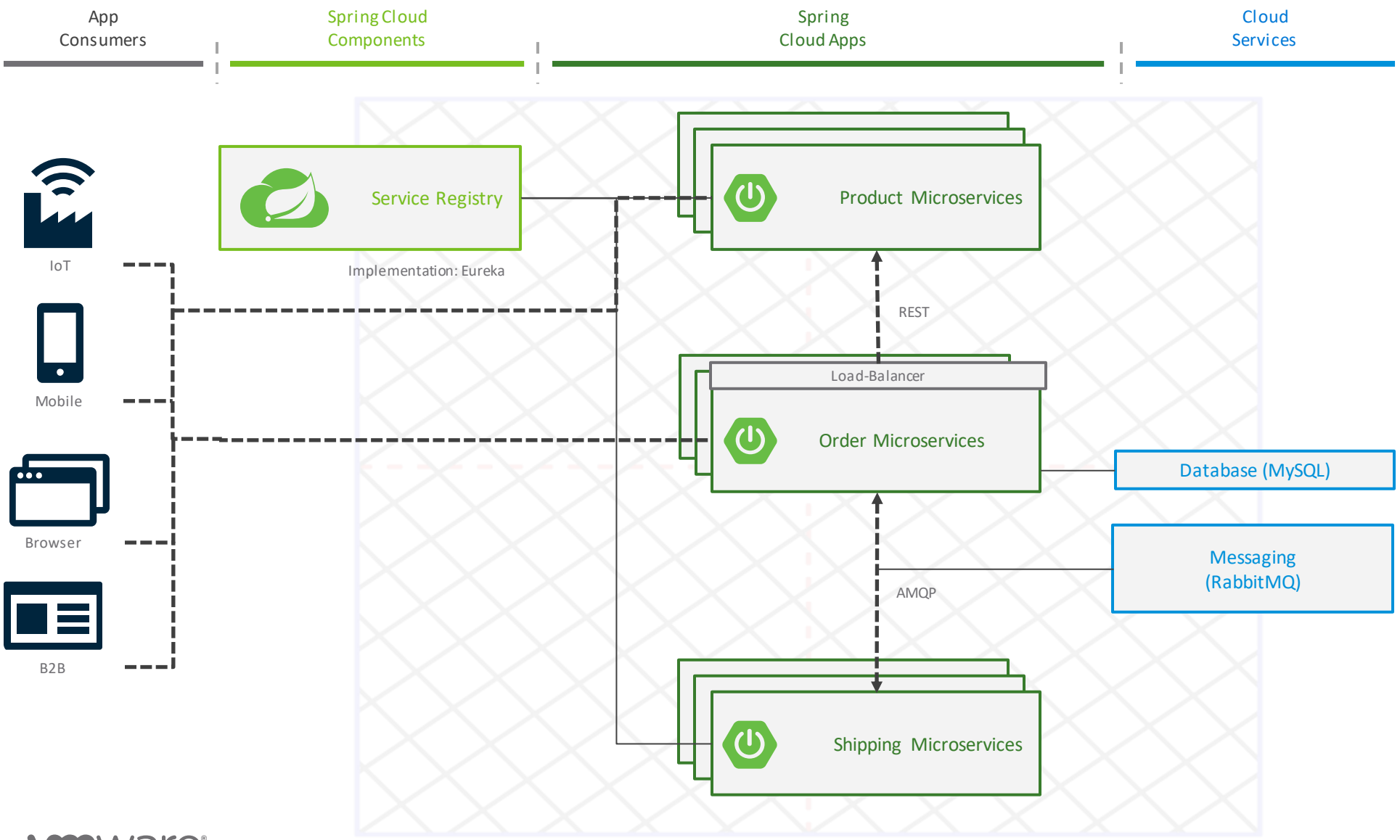
Great UXs need great responses. When your apps are under pressure, spreading the load helps smooth things out!

Spring Cloud Load Balancer can help your clients spread requests across multiple service instances

Integrates with Rest clients and WebFlux clients.

Supports health checks, multiple caching options and Zone based balancing.

A typical Spring Cloud architecture



All Microservices:
Spring Boot
Spring Cloud Discovery Client

Order Microservices:
+ Spring Cloud LoadBalancer

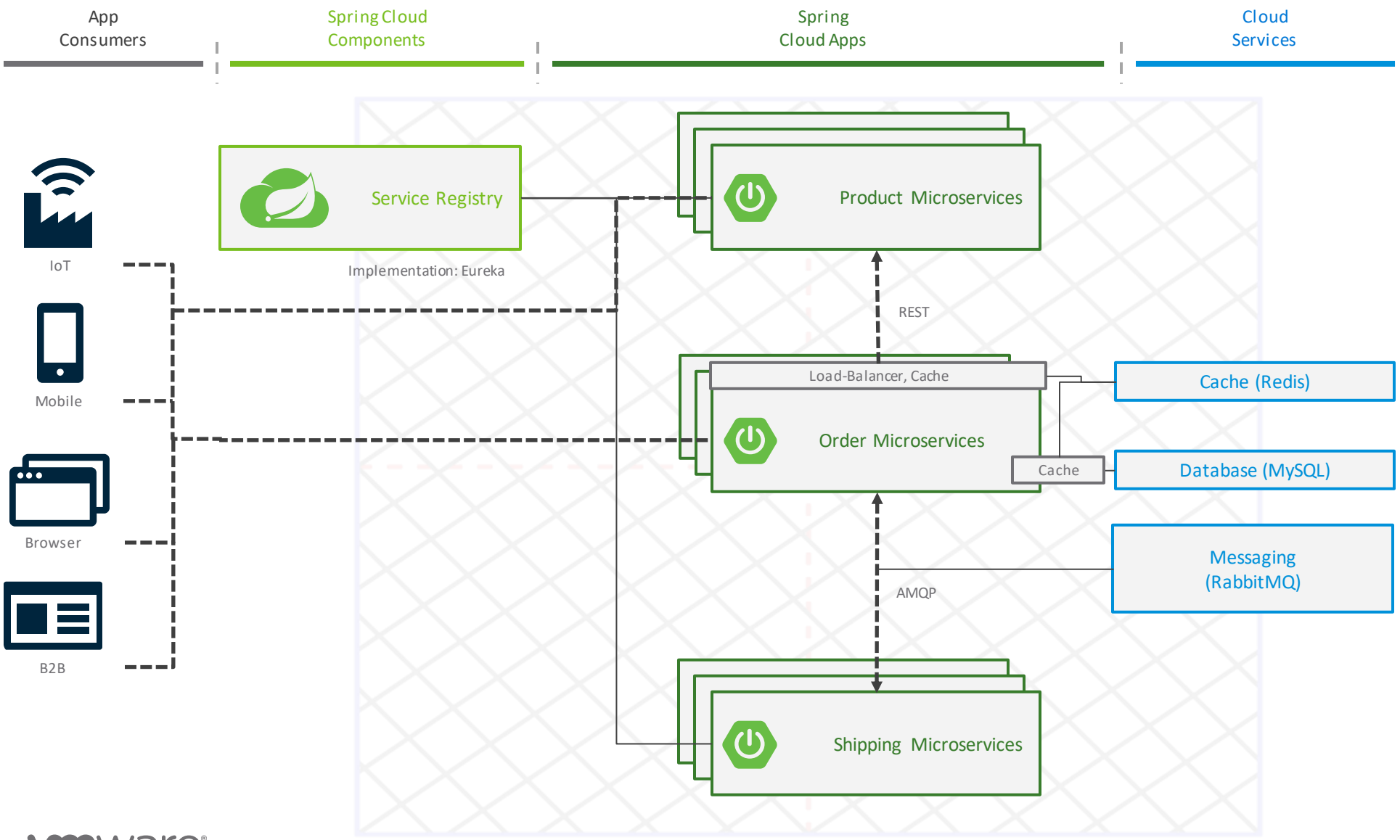
Caching

For example traditional databases are often too brittle or unreliable for use with microservices. That's why every modern distributed architecture needs a cache!

The Spring Framework provides support for transparently adding caching to an application.

The cache abstraction does not provide an actual store. Examples for Cache providers that are supported out of the box are EhCache, Hazelcast, Couchbase, Redis and Caffeine. Other providers like VMware Tanzu GemFire can also be used with minimal configuration.

A typical Spring Cloud architecture



All Microservices:
Spring Boot
Spring Cloud Discovery Client

Order Microservices:
+ Spring Cloud LoadBalancer

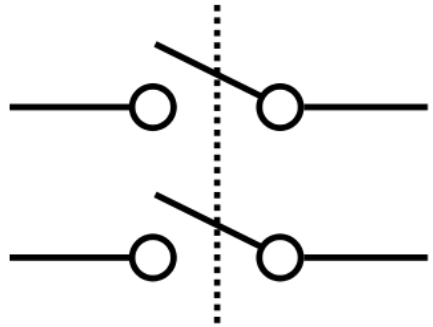
Circuit Breaker

Distributed systems can be unreliable - requests might timeout or fail completely ...

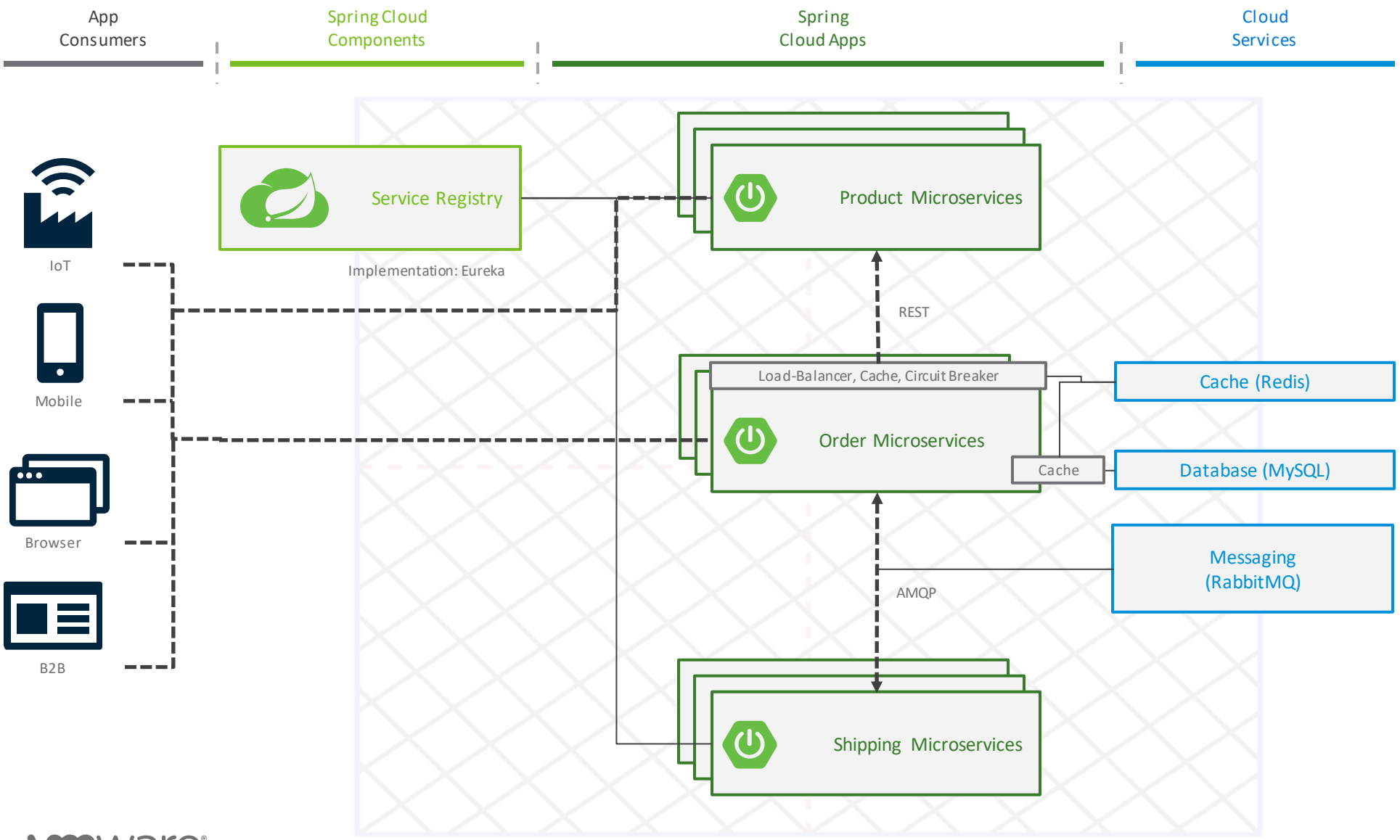
Circuit breakers mitigate this problem using sensible defaults and reliable fallbacks in case of emergency.

Spring Cloud Circuit Breaker gives you the choice of three popular open-source options:

- Resilience4J
- Sentinel
- Hystrix



A typical Spring Cloud architecture



All Microservices:
Spring Boot
Spring Cloud Discovery Client

Order Microservices:
+ Spring Cloud LoadBalancer
+ Spring Cloud Circuit Breaker

Configuration

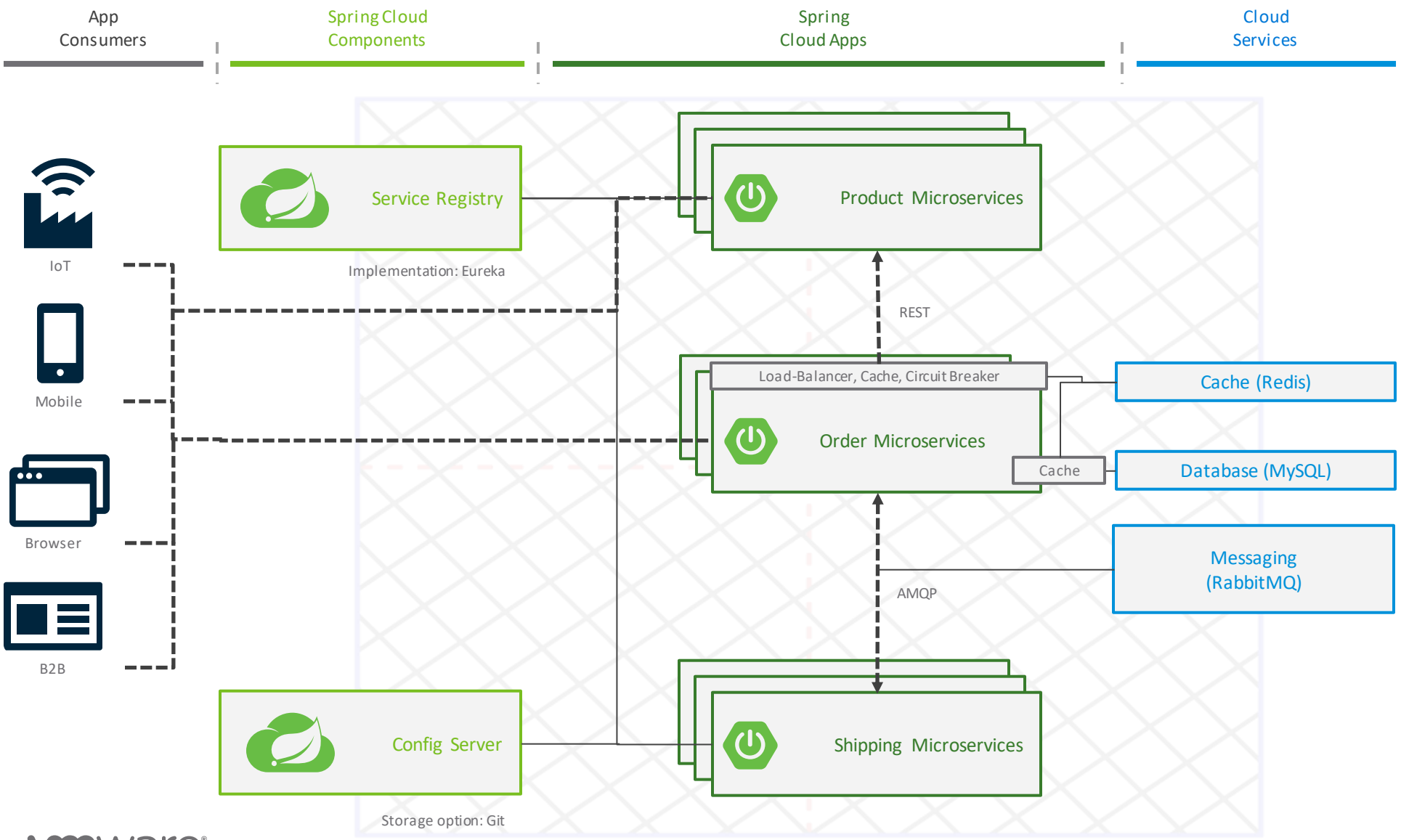
In cloud-native applications, configuration shouldn’t be bundled with code!

In the cloud, you have multiple applications, environments, and service instances — so configuration has to be flexible.

Spring Cloud Config is designed to ease this burden.

It delivers config straight to your apps and offers integration with multiple version control systems to keep your config safe.

A typical Spring Cloud architecture



All Microservices:
Spring Boot
Spring Cloud Discovery Client
Spring Cloud Config Client

Order Microservices:
+ Spring Cloud LoadBalancer
+ Spring Cloud Circuit Breaker

Gateway

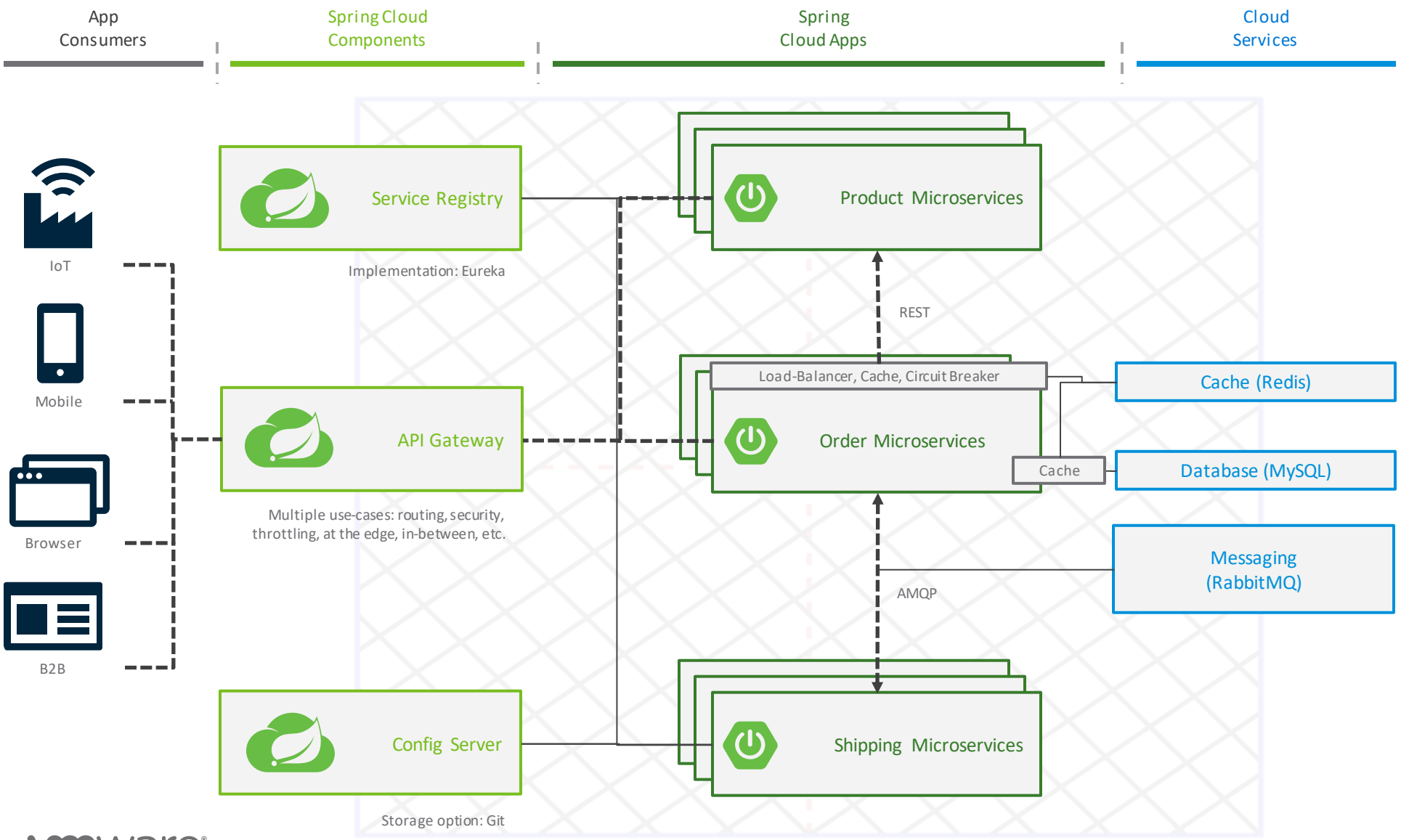
With so many APIs in play, developers need an
API Gateway that they can control!

Spring Cloud Gateway puts developers in control of APIs:

- Securing and hiding services
- Routing and filtering messages
- Handling load
- And much more …

Manage your config in regular version-control.
Roll-out your changes instantly - no tickets, no downtime!

A typical Spring Cloud architecture



All Microservices:
Spring Boot
Spring Cloud Discovery Client
Spring Cloud Config Client
Spring Cloud Sleuth

Order Microservices:
+ Spring Cloud LoadBalancer
+ Spring Cloud Circuit Breaker

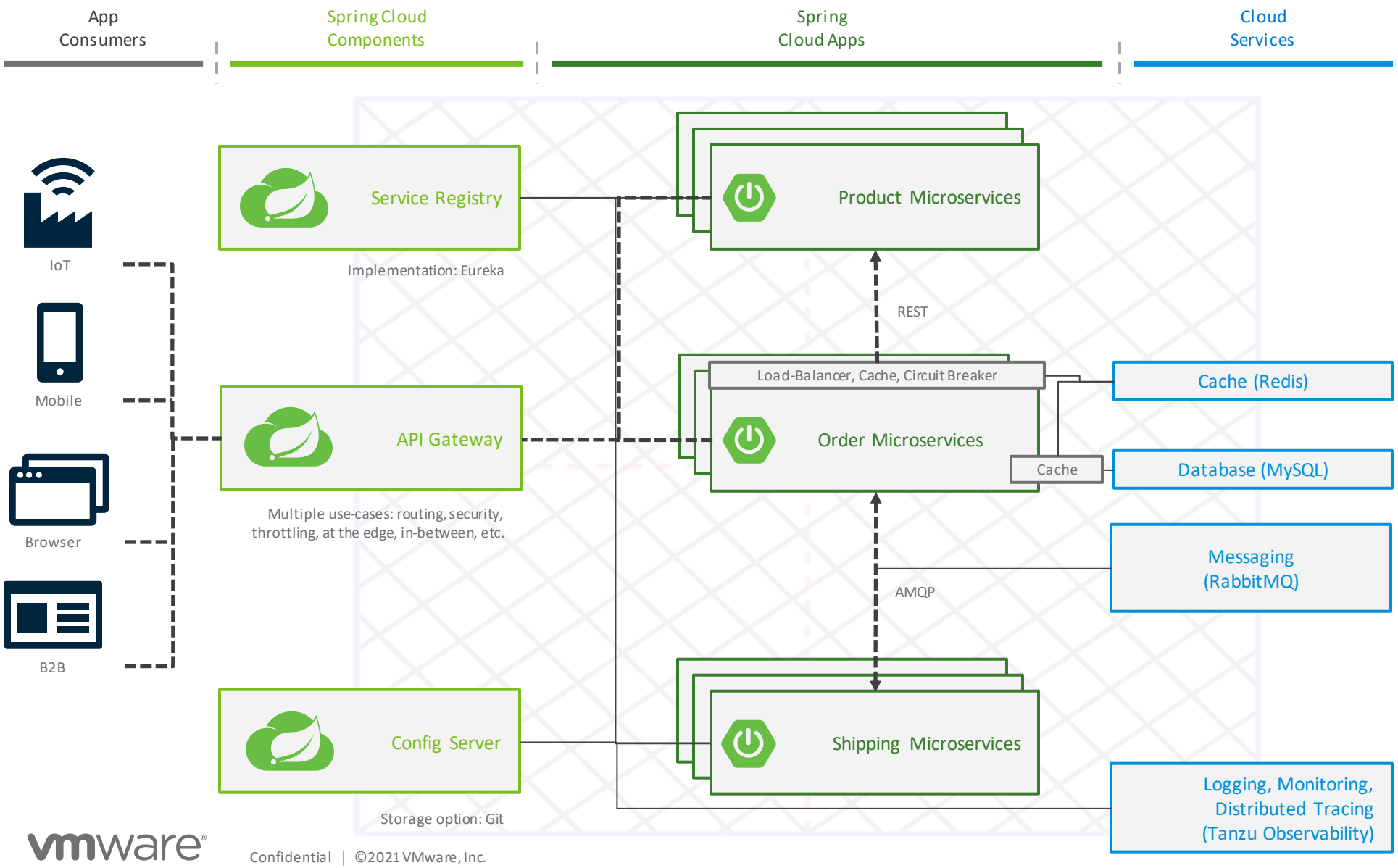
Distributed Tracing

Debugging distributed applications is a complex and time consuming chore!

For any given failure or poor experience, you'll need to piece together traces from multiple independent microservices.

Spring Cloud Sleuth instruments your applications using predictable industry-standard patterns.
When combined with Zipkin, you can zero in on latency problems fast.

A typical Spring Cloud architecture



All Microservices:
Spring Boot
Spring Cloud Discovery Client
Spring Cloud Config Client
Spring Cloud Sleuth

Order Microservices:
+ Spring Cloud LoadBalancer
+ Spring Cloud Circuit Breaker

Streaming Data

When you're working with streaming data, you need three key abstractions to simplify your code:

- Binders to integrate messaging systems (Kafka, RabbitMQ, SQS, etc.)
- Bindings to bridge the gap between messaging systems and code
- Messages to provide structure for data

Spring Cloud Stream delivers them all. Spring Cloud Stream also handles provisioning, content conversion, error handling, config management, consumer groups, partitioning, monitoring, and more

Serverless & Functions

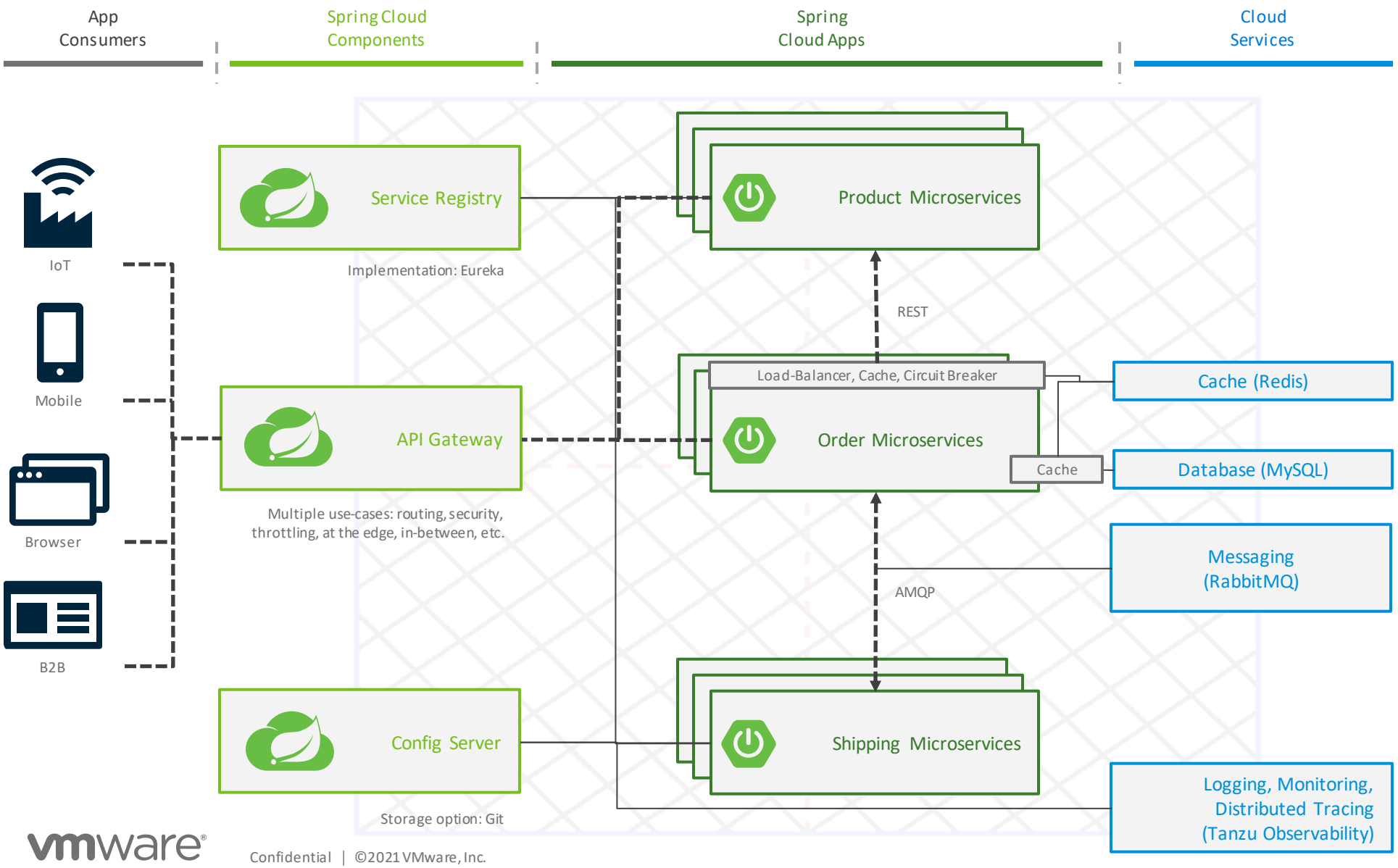
Vendor lock-in is a concern for many, so why not decouple your functions from your provider?

Spring Cloud Function lets you write functions once and run anywhere with familiar Spring APIs.

Function chaining lets you create sophisticated capabilities with ease

Multiple function inputs and outputs allow you to merge, join, and build other advanced use cases

A typical Spring Cloud architecture



All Microservices:
Spring Boot
Spring Cloud Discovery Client
Spring Cloud Config Client
Spring Cloud Sleuth

Order Microservices:
+ Spring Cloud LoadBalancer
+ Spring Cloud Circuit Breaker

Shipping Microsevices:
+ Spring Cloud Stream
+ Spring Cloud Function

Spring Cloud Netflix projects

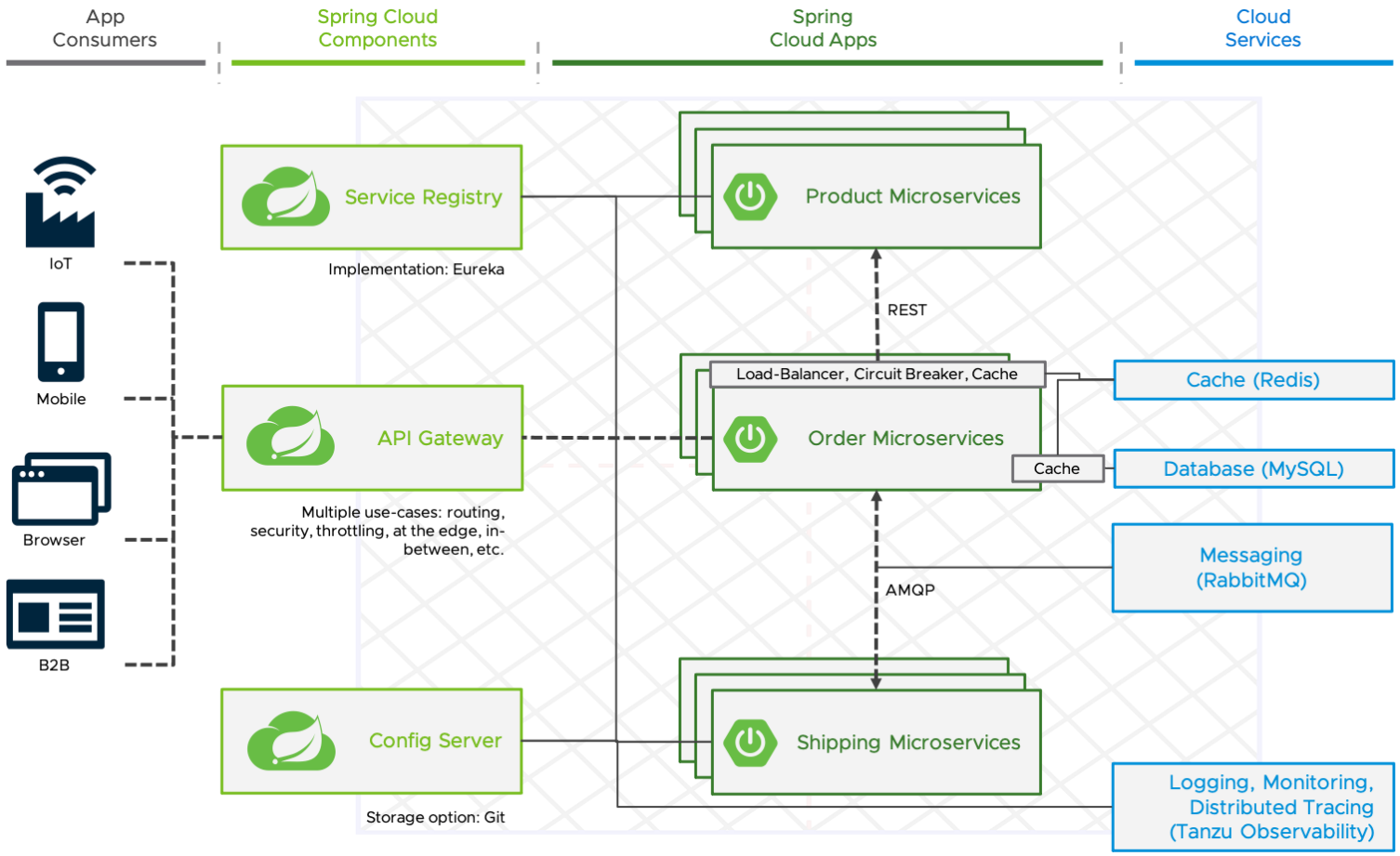
Since most of the Spring Cloud Netflix projects are in maintenance mode (see https://via.vmw.com/EN6F and https://via.vmw.com/EN6E) I used available replacements:

- Circuit breaker: Hystrix → Resilience4j
- Client-side LB: Ribbon → Spring Cloud LoadBalancer
- Application Gateway: Zuul → Spring Cloud Gateway

The maintenance mode does not include the Eureka module(service registry).

Common challenges

- High effort required to manage cloud infrastructure for microservices
- Application lifecycle is difficult to manage
- Painful to troubleshoot application issues

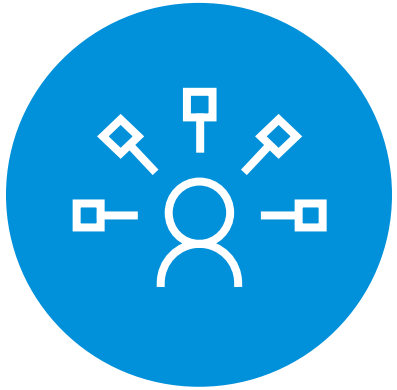


Azure Spring Cloud

A fully managed service for Spring Boot (and .NET*) microservices

More choices and full integration into Azure's ecosystem and services

Fully managed infrastructure



Built-in app lifecycle management

Ease of monitoring

Enterprise ready

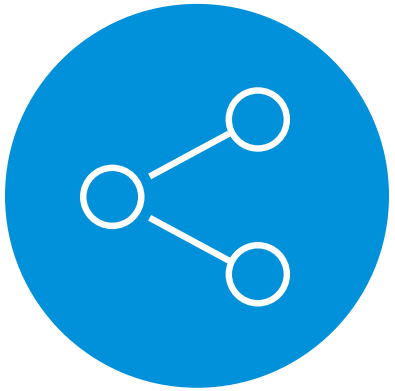
* Now in public preview: Run .NET Applications in Azure Spring Cloud using Steeltoe

VMware Tanzu Application Service

A Transformational Runtime for Apps

Best runtime for Spring and Spring Boot

A native Windows and .NET experience



Microservices Made Easy

Designed for Apps

Ready for Containers

Spring Cloud Kubernetes

https://github.com/tsalm-pivotal/spring-cloud-demo-k8s

Spring Boot > 2.3 Kubernetes Features

Cloud Native Buildpacks

Package up your Spring Boot application into containers with automated best practices using the paketo.io CNBs

```
mvn spring-boot:build-image
gradle bootBuildImage
```

Graceful shutdown support

Graceful shutdown allows a Spring Boot application to stop accepting new requests and finish currently executing requests.

application.yaml

```
server.shutdown=graceful
spring.lifecycle.timeout-per-shutdown-phase=20s
```

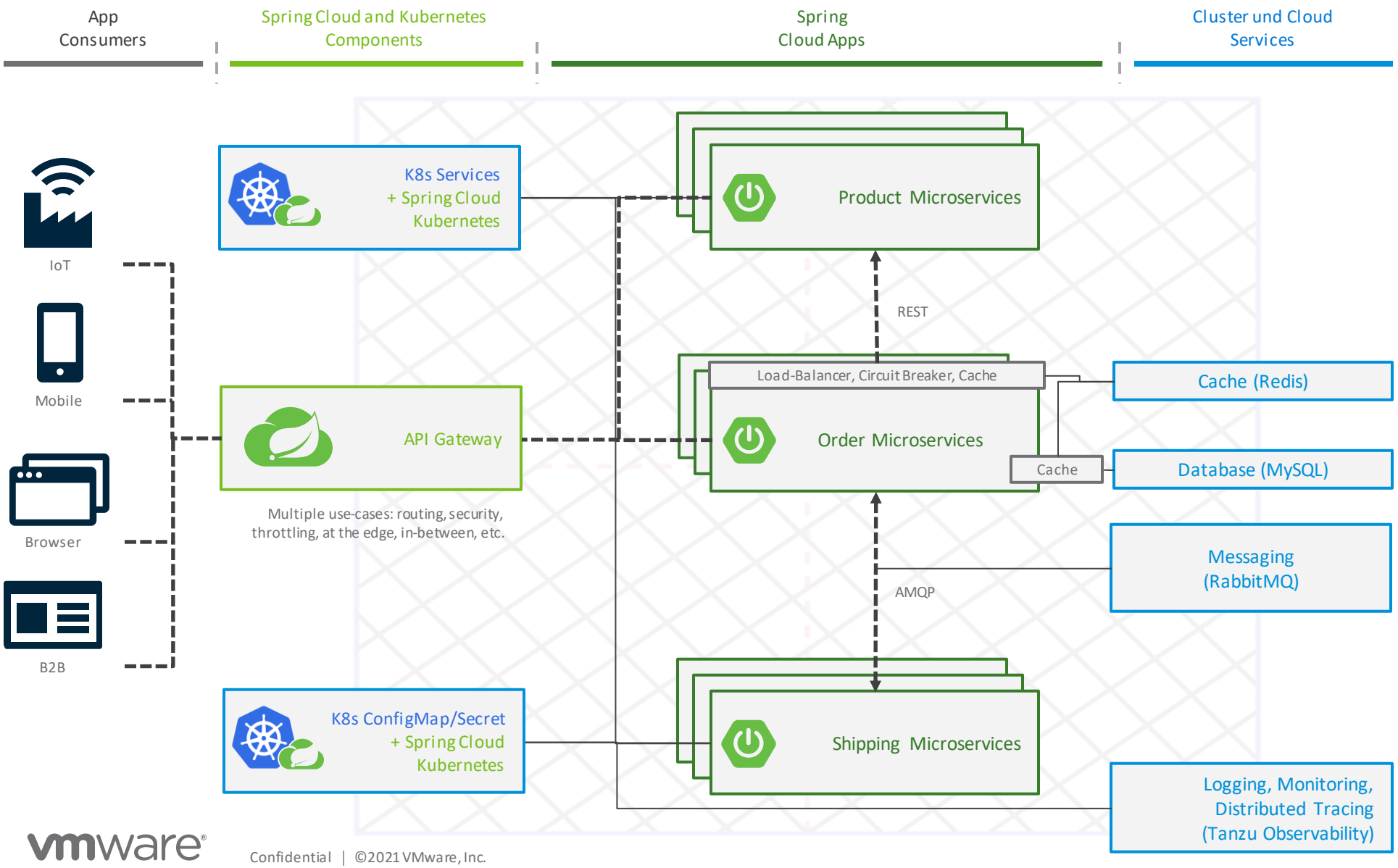
Kubernetes Life and Readiness Probes

Spring Boot 2.3 added actuators groups for Kubernetes life and readiness probes.

my-deployment.yaml

```
livenessProbe:
    httpGet:
        path: /actuator/health/liveness
        port: <actuator-port>
    failureThreshold: ...
    periodSeconds: ...
readinessProbe:
    httpGet:
        path: /actuator/health/readiness
        port: <actuator-port>
    failureThreshold: ...
    periodSeconds: ...
```

A typical Spring Cloud architecture on Kubernetes



All Microservices:
Spring Boot
Spring Cloud Kubernetes
Spring Cloud Sleuth

Order Microservices:
+ Spring Cloud Circuit Breaker

Shipping Microsevices:
+ Spring Cloud Stream
+ Spring Cloud Function

Spring Cloud Kubernetes Discovery Client & Load Balancer

DiscoveryClient for Kubernetes

Let‘s you query Kubernetes endpoints by name.

A service is typically exposed by the Kubernetes API server as a collection of endpoints that a client can access from a Spring Boot application running as a pod.

Load Balancer for Kubernetes

Spread requests across multiple service instances via load balancing based on Kubernetes Services or Kubernetes Endpoints(via Spring Cloud Loadbalancer).

Kubernetes ConfigMaps and Secrets

Spring Cloud Kubernetes Config makes ConfigMap and Secret instances available during application bootstrapping.

It‘s also able to trigger hot reloading of beans or Spring context when changes are detected on observed ConfigMap instances.

The reload feature is by default disabled.

application.yaml

```
spring.cloud.kubernetes.reload.enabled=true
spring.cloud.kubernetes.reload.strategy=refresh #other options: restart_context, shutdown
```

Thank You

- Twitter: @salmto
- GitHub: https://github.com/tsalm-pivotal