Cloud Native Java

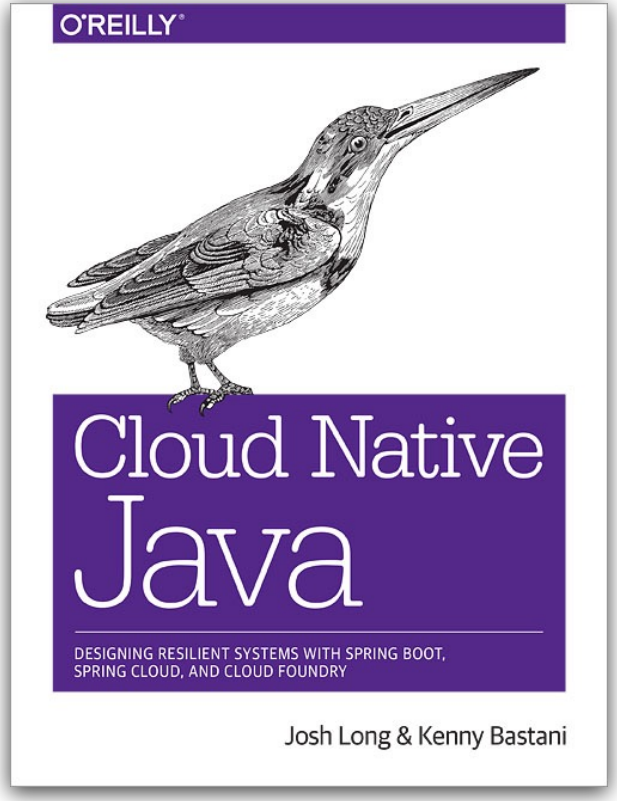
Kenny Bastani
Spring Developer Advocate @ **Pivotal**

Kenny Bastani

Spring Developer Advocate

spring io

Pivotal



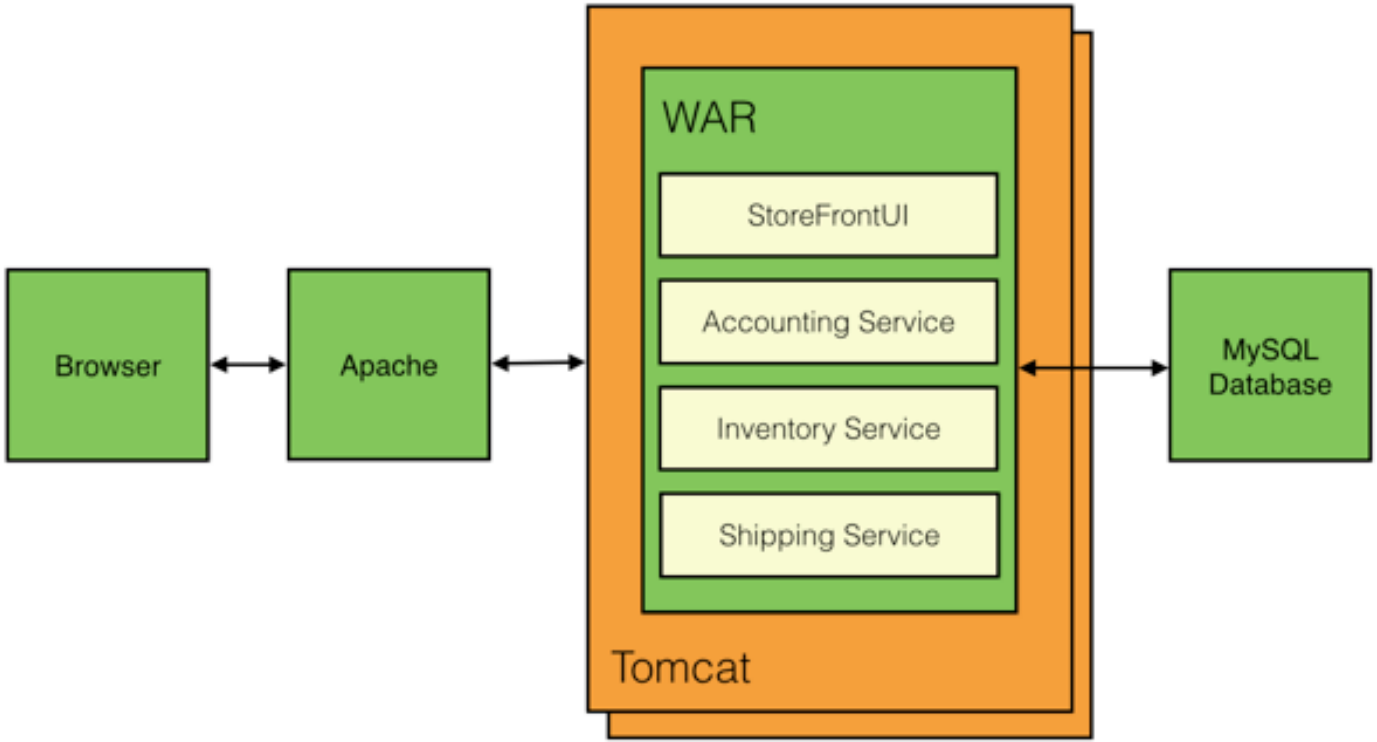
Agenda

	Agenda
1	Monolith to Microservices
2	Microservice Reference Architecture
3	Spring Boot & Spring Cloud Examples

Microservices

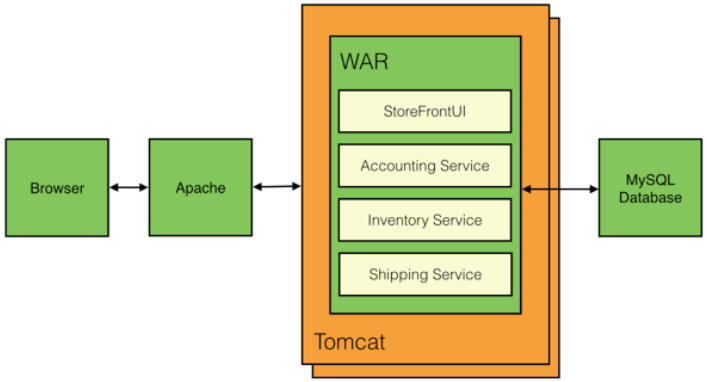
How did we get there?

We started with the monolith…



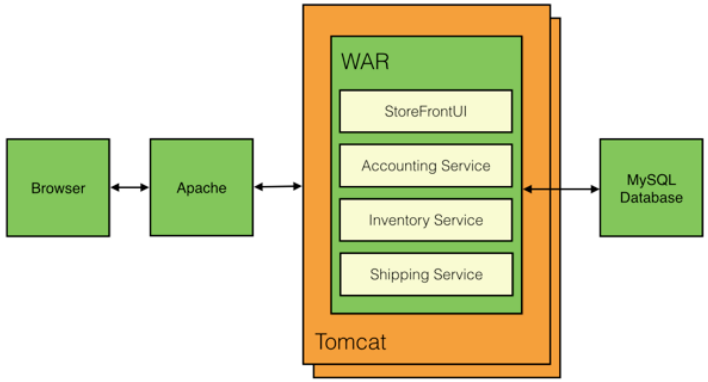
Cultural problems with monoliths

- Slows our velocity getting into production
- It takes too long to ramp up new engineers
- The code base is just too large for any one person to fully comprehend
- Centralized authority and change management slows progress (DBA, Ops)

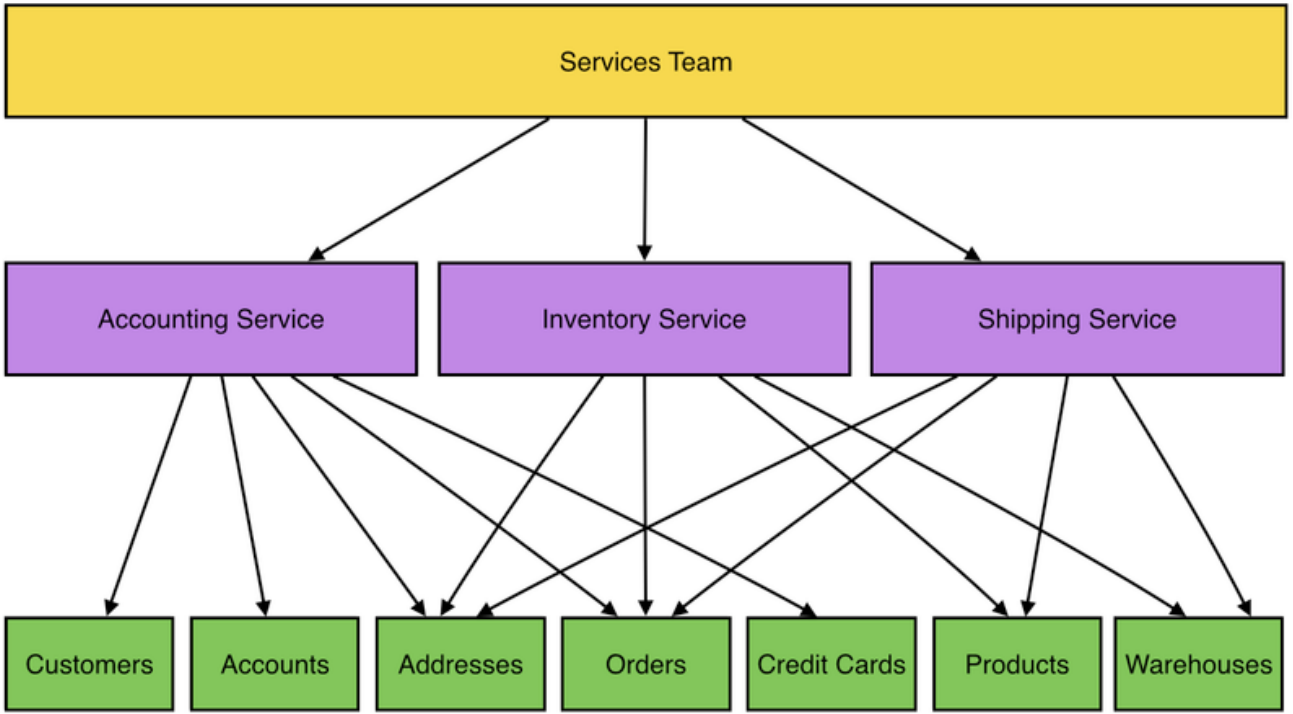


Operational problems with monoliths

- Deployments batch many changes together from different teams
- Operations drives the runtime environment of applications
- Operators take on all operational responsibility (including VM upgrades)
- *Deploy everything at once or nothing at all*

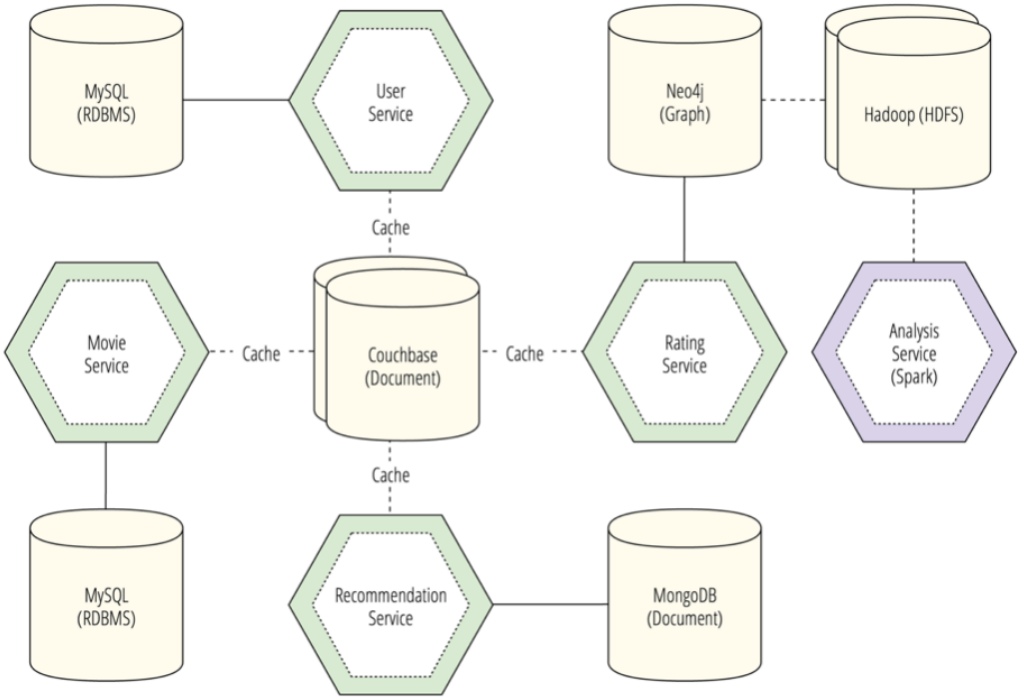


We then moved towards SOA…



We have now arrived at microservices…

- Each team provisions self-service infrastructure, like a database, and builds a single application
- Centralized resources are provided as backing services that are shared for data consistency, caching



Online Store Example

Cloud native application as microservices

Cloud Native Outfitters

online-store-web.cfapps.io/index.html#/products/SKU-34563

Search

Cloud Native Outfitters

Welcome, John Doe! Orders Settings Cart Logout

Home / Products / Like a BOSH (T-Shirt, Women's Medium)

Summary

Product name:

Like a BOSH (T-Shirt, Women's Medium)

Price:

14.99

Product Details

The BOSH Outer Shell (**BOSH**) is an elegant release engineering tool for a more virtualized cloud-native age. The feeling of spinning up a highly available distributed system of VMs is second only to the feeling of frequently pushing code to production. Show the cloud *who's BOSH* with this stylish cloud native ops tee.

Cloud Native Tee Collection
99% YAML, 11% CLI
BOSH CCK *recommended*
4 nines of *re-washability*

Price: $14.99

Product number: SKU-34563

0 Add to Cart

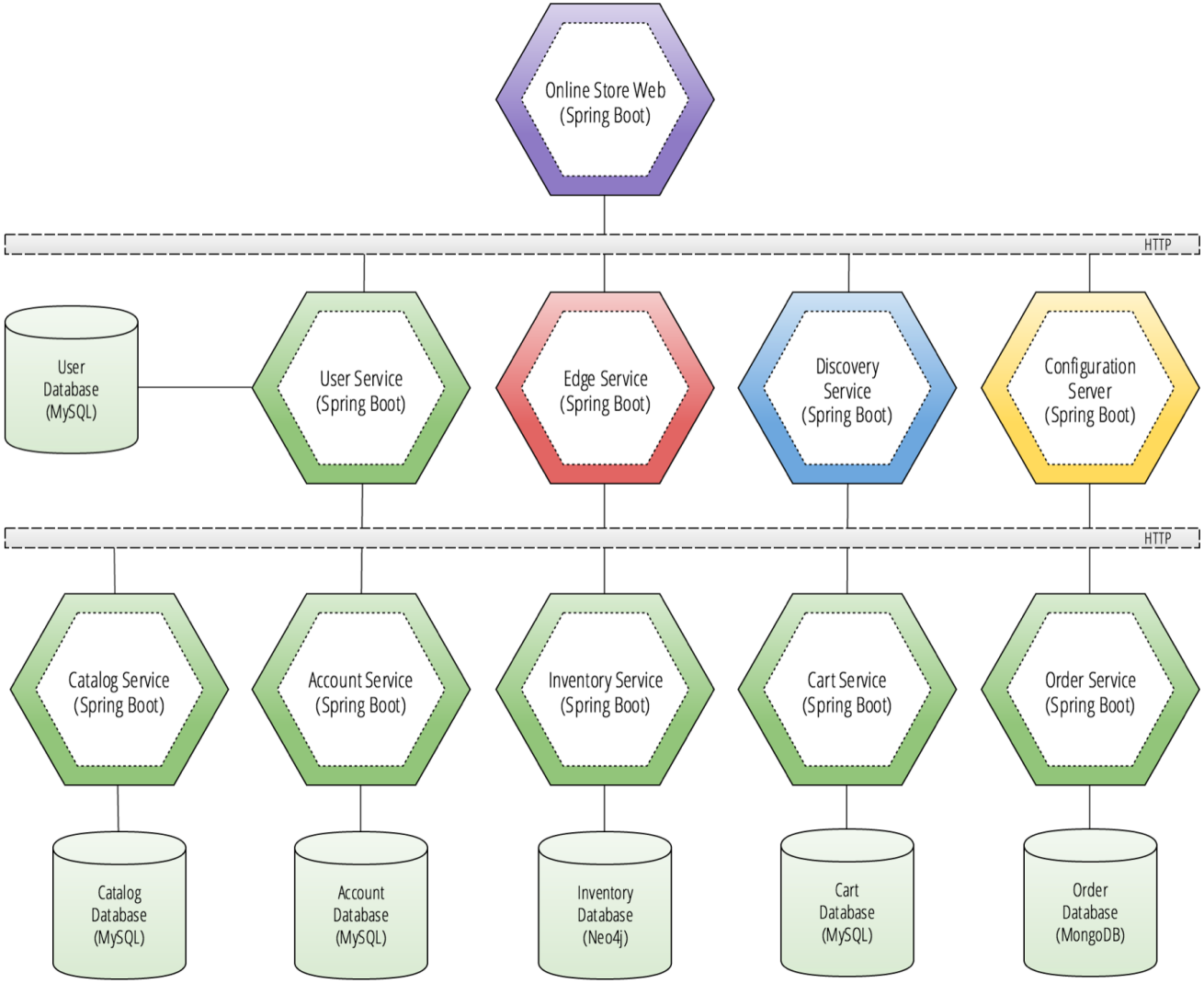
Related

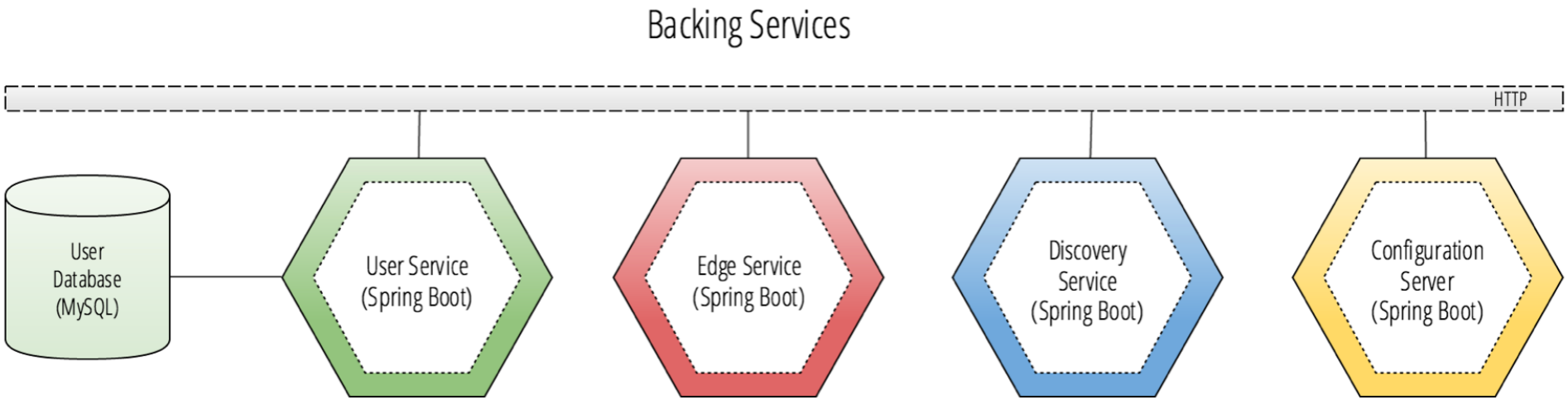
Best. Cloud. Ever. (T-Shirt, Men's

We're gonna need a bigger VM

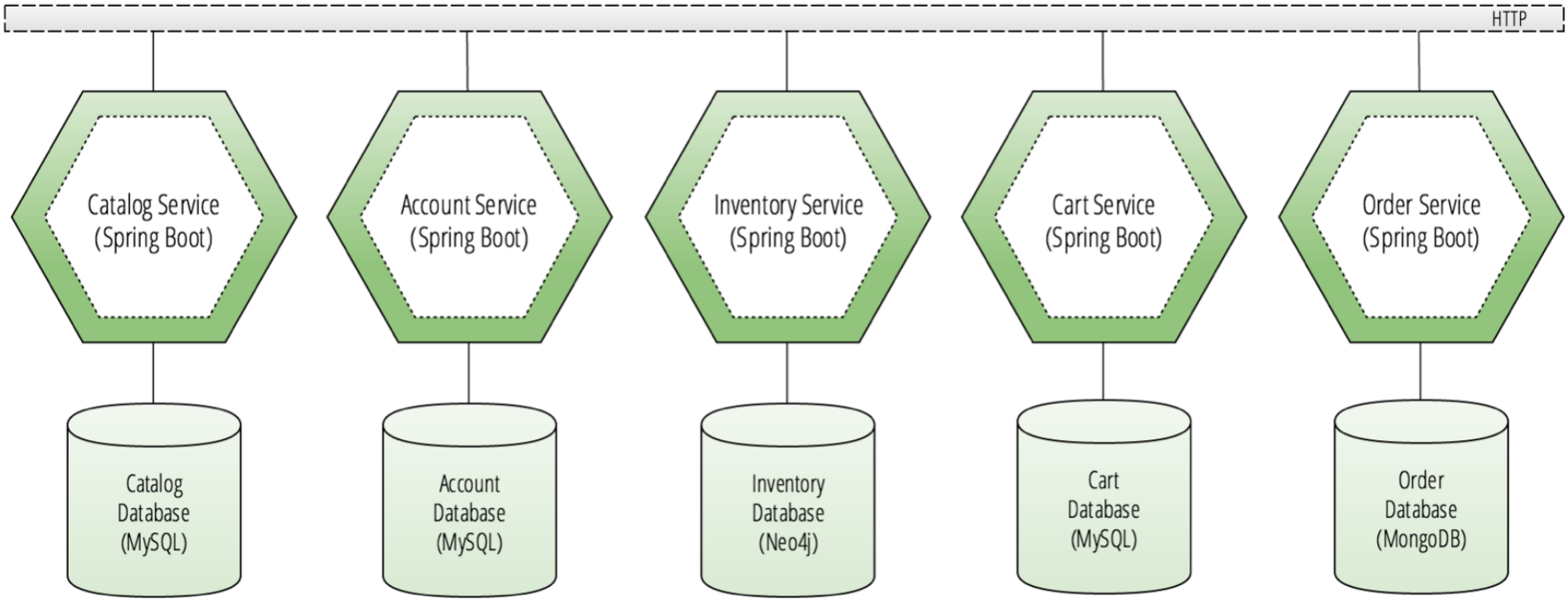
Like a BOSH (T-Shirt, Women's

cf push awesome (Hoodie, Men's


Online Store Web
(Spring Boot)
HTTP
User
Database
(MySQL)
User Service
(Spring Boot)
Edge Service
(Spring Boot)
Discovery
Service
(Spring Boot)
Configuration
Server
(Spring Boot)
HTTP
Catalog Service
(Spring Boot)
Account Service
(Spring Boot)
Inventory Service
(Spring Boot)
Cart Service
(Spring Boot)
Order Service
(Spring Boot)
Catalog
Database
(MySQL)
Account
Database
(MySQL)
Inventory
Database
(Neo4j)
Cart
Database
(MySQL)
Order
Database
(MongoDB)

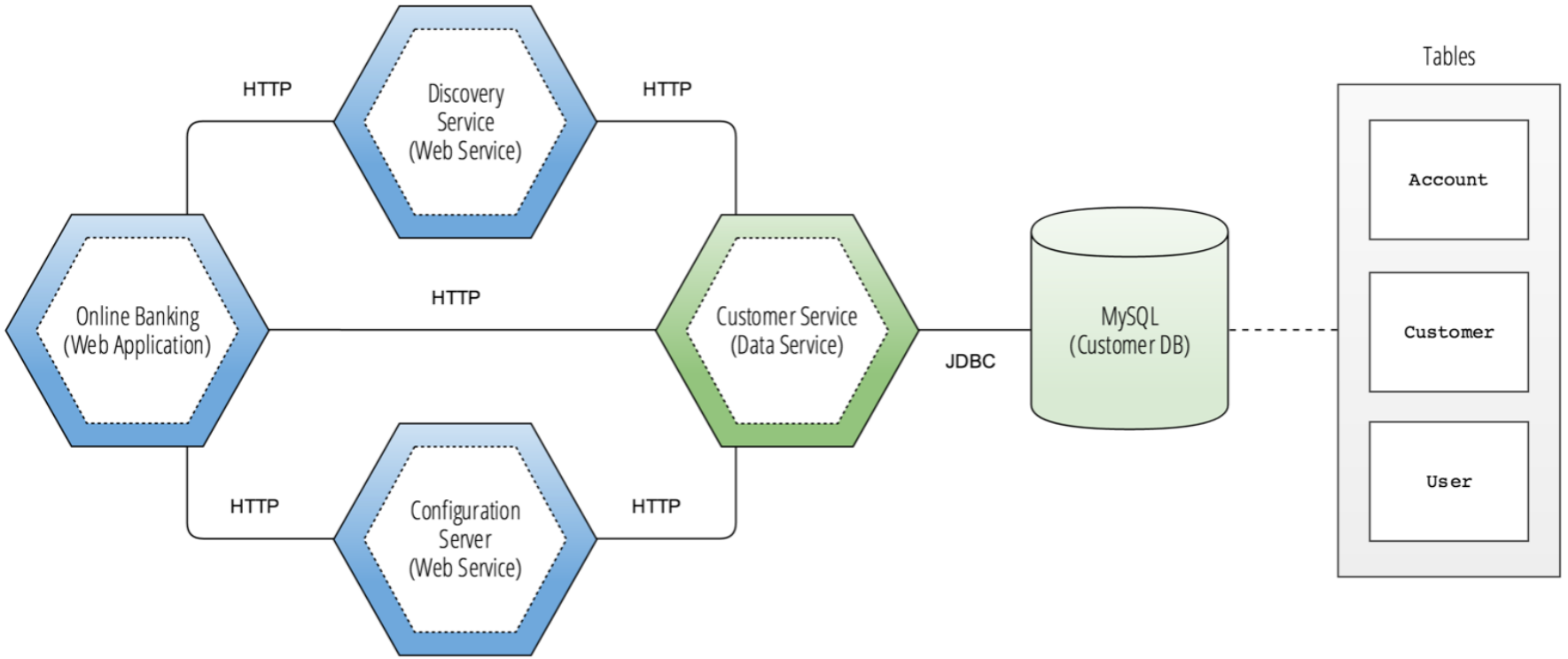

Backing Services
HTTP
User
Database
(MySQL)
User Service
(Spring Boot)
Edge Service
(Spring Boot)
Discovery
Service
(Spring Boot)
Configuration
Server
(Spring Boot)

Backend Microservices

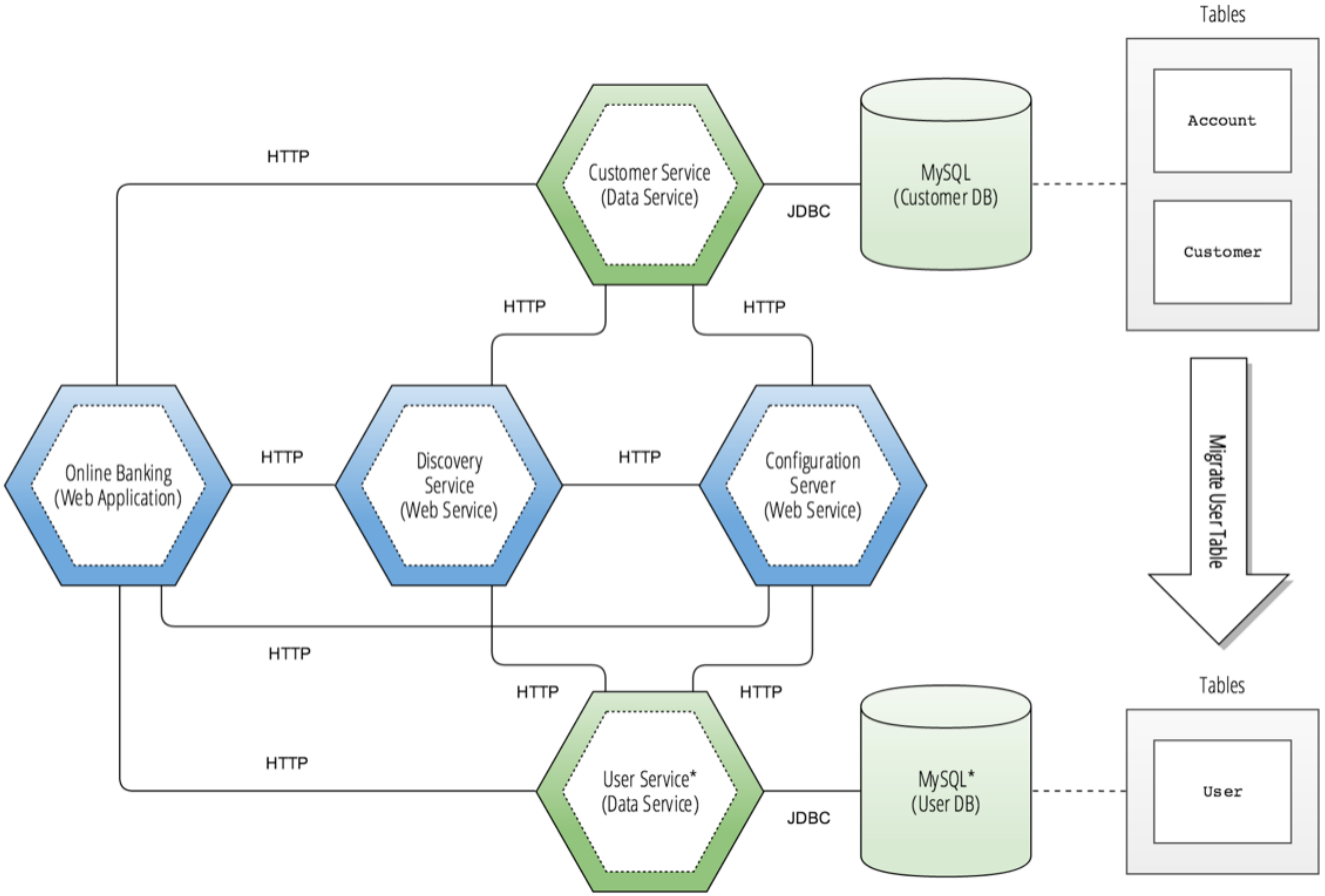


Migration strategies

Monolith to Microservice



Splitting the Monolith: User Service Migration



Strangler Pattern

“Gradually create a new system around the edges of the old, letting it grow slowly over several years until the old system is strangled.”

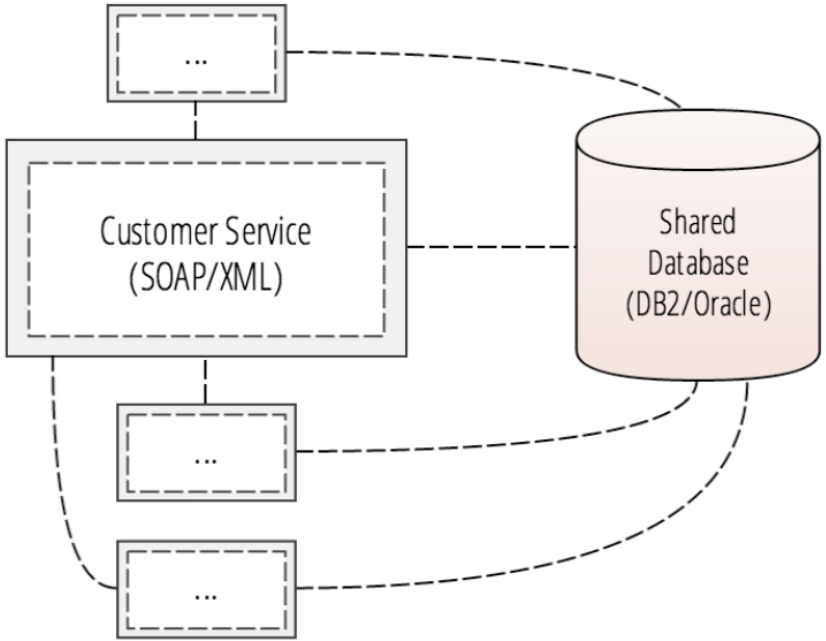
— Martin Fowler

Strangler vines—seed in the upper branches of a fig tree and gradually work their way down to the soil—strangling and eventually and killing the tree

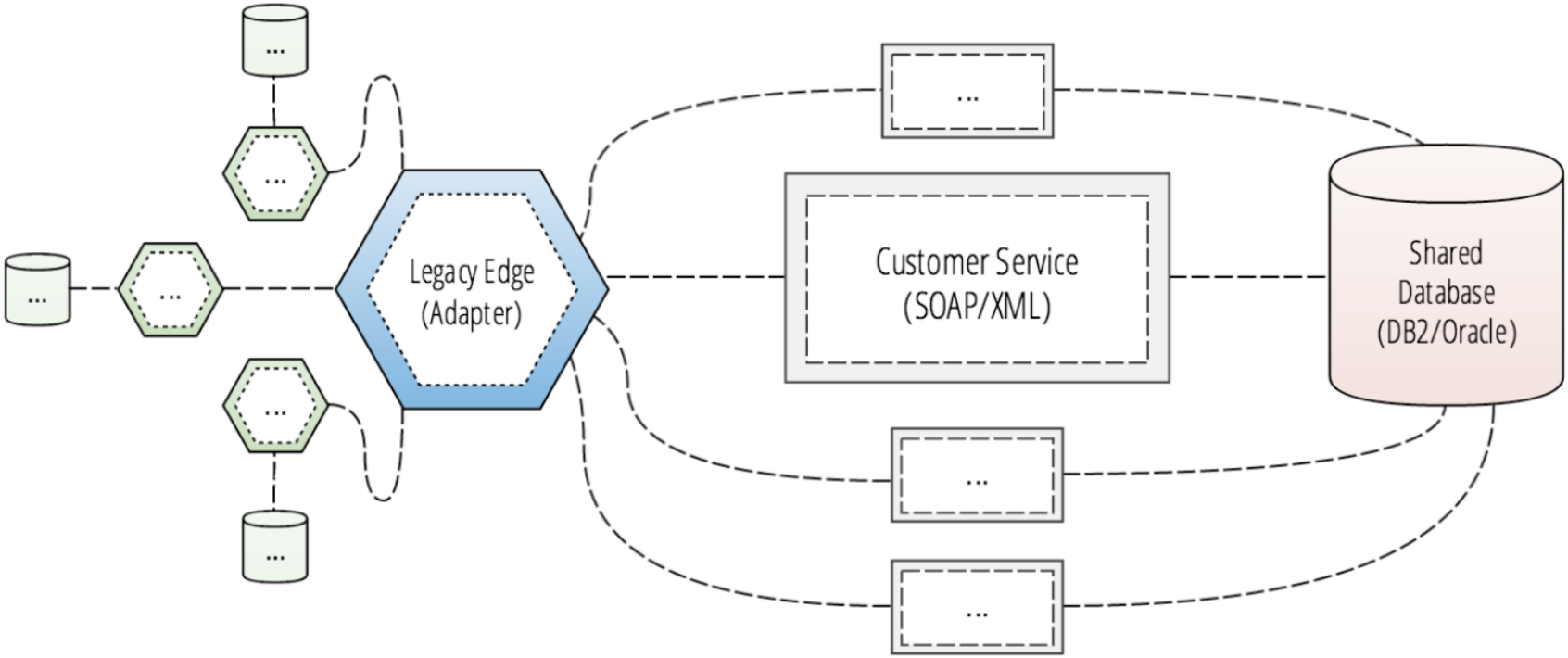
Strangling legacy using microservices

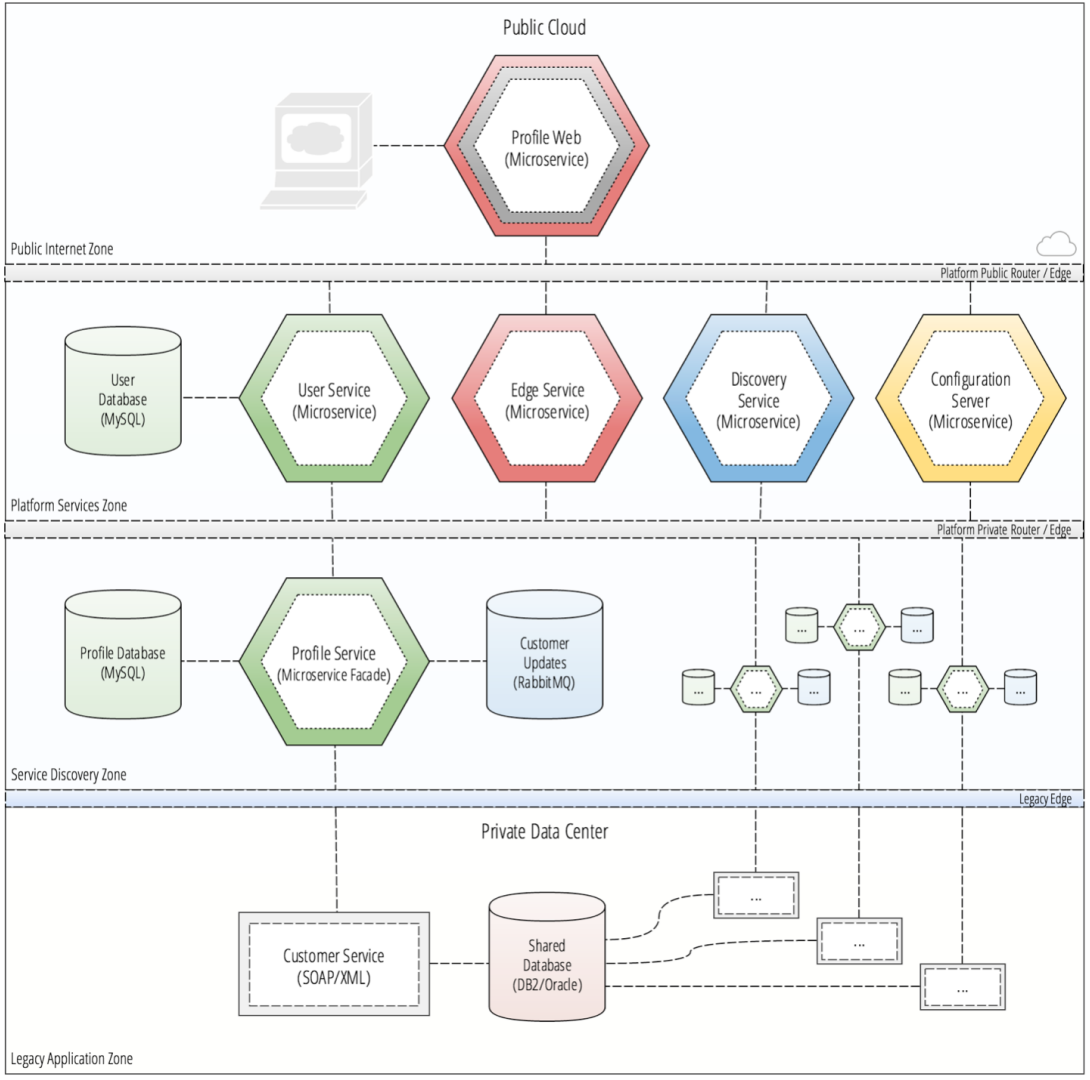
How can we take advantage of building microservices that also strangle data from the edge of legacy systems?

Point-to-point Connections



Legacy Indirection Layer




Public Cloud
Profile Web
(Microservice)
Public Internet Zone
Platform Public Router / Edge
User
Database
(MySQL)
User Service
(Microservice)
Edge Service
(Microservice)
Discovery
Service
(Microservice)
Configuration
Server
(Microservice)
Platform Services Zone
Platform Private Router / Edge
Profile Database
(MySQL)
Profile Service
(Microservice Facade)
Customer
Updates
(RabbitMQ)
Service Discovery Zone
Legacy Edge
Private Data Center
Customer Service
(SOAP/XML)
Shared
Database
(DB2/Oracle)
Legacy Application Zone

Spring Boot

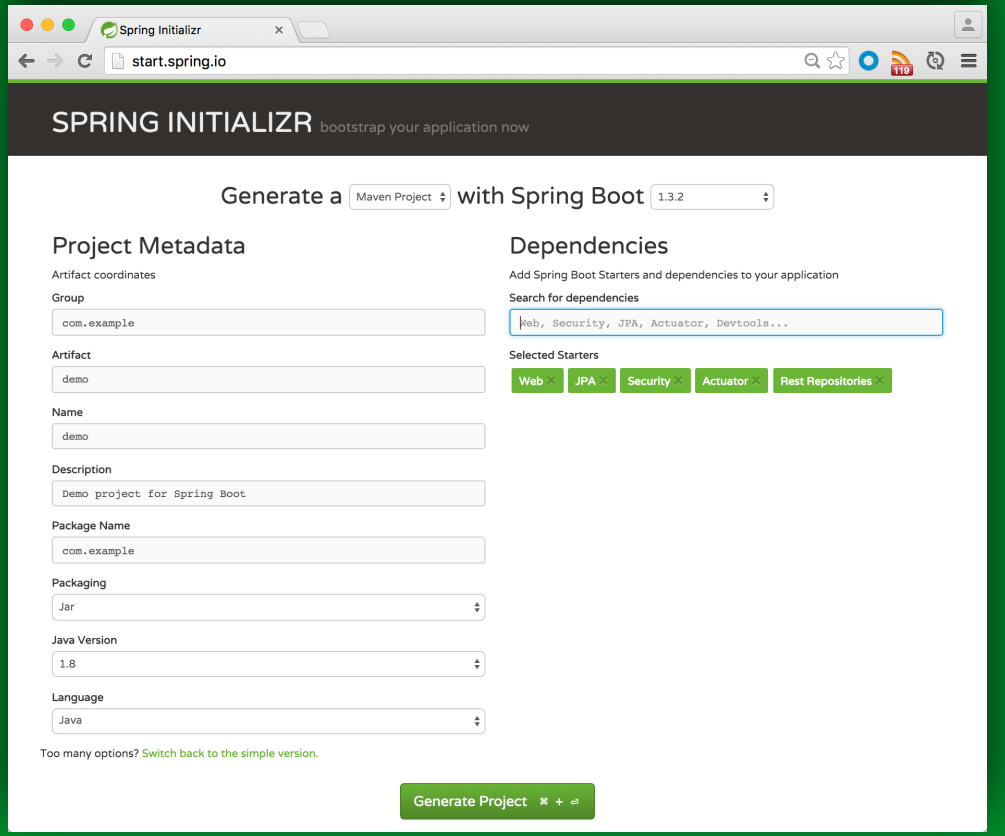
A JVM micro-framework for building microservices

What is Spring Boot?

Spring Initializr for bootstrapping your applications

spring boot

supports rapid development of production-ready applications and services

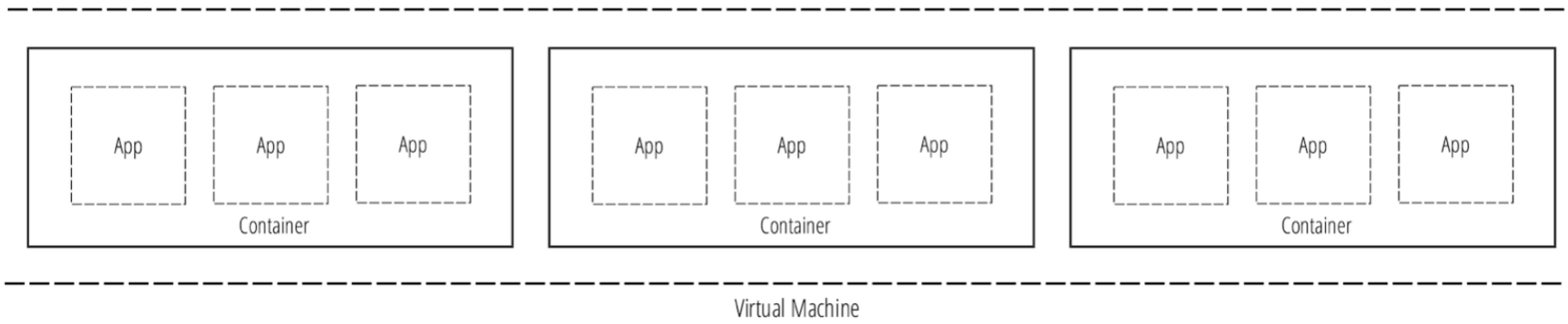


Cloud Platforms

Shipping applications in containers instead of shared application servers

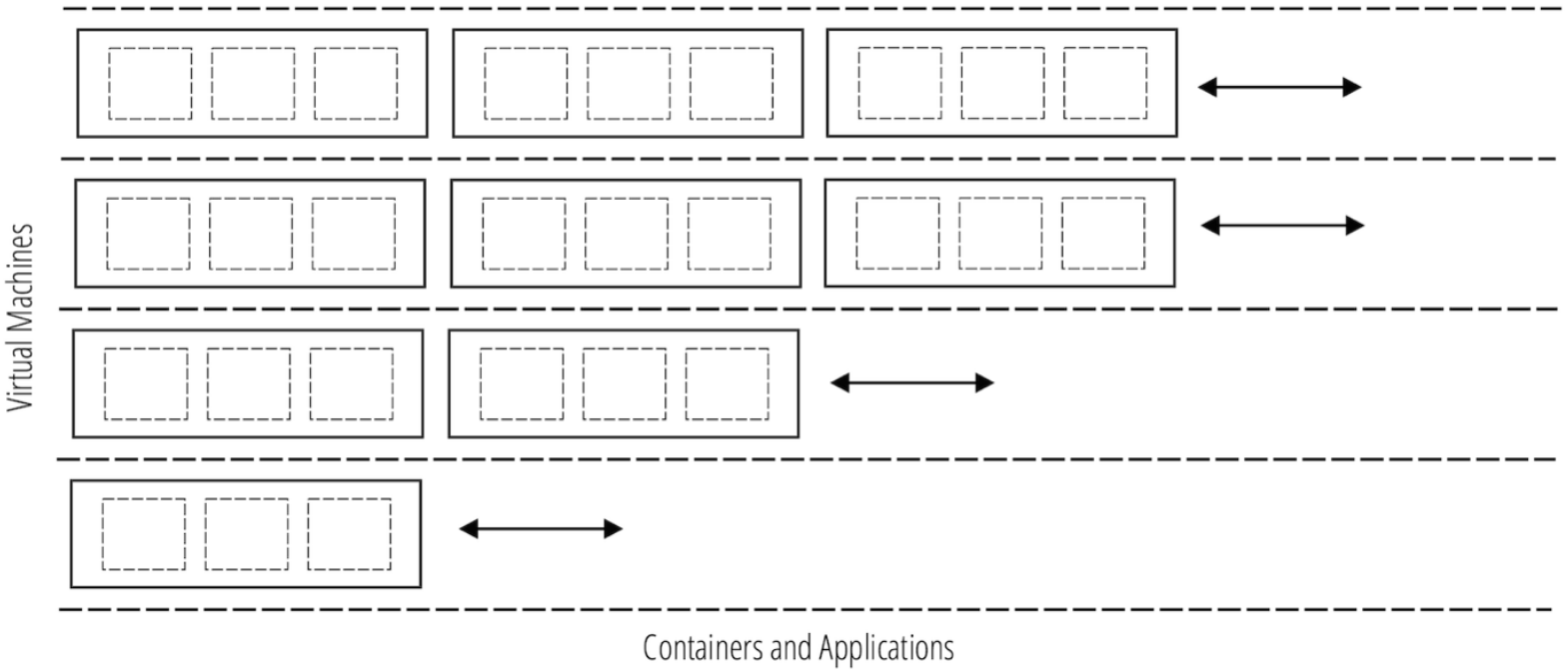
Microservices: Container Deployment

- Each microservice can be containerized with their application dependencies
- Containers get scheduled on virtual machines with an allotted resource policy



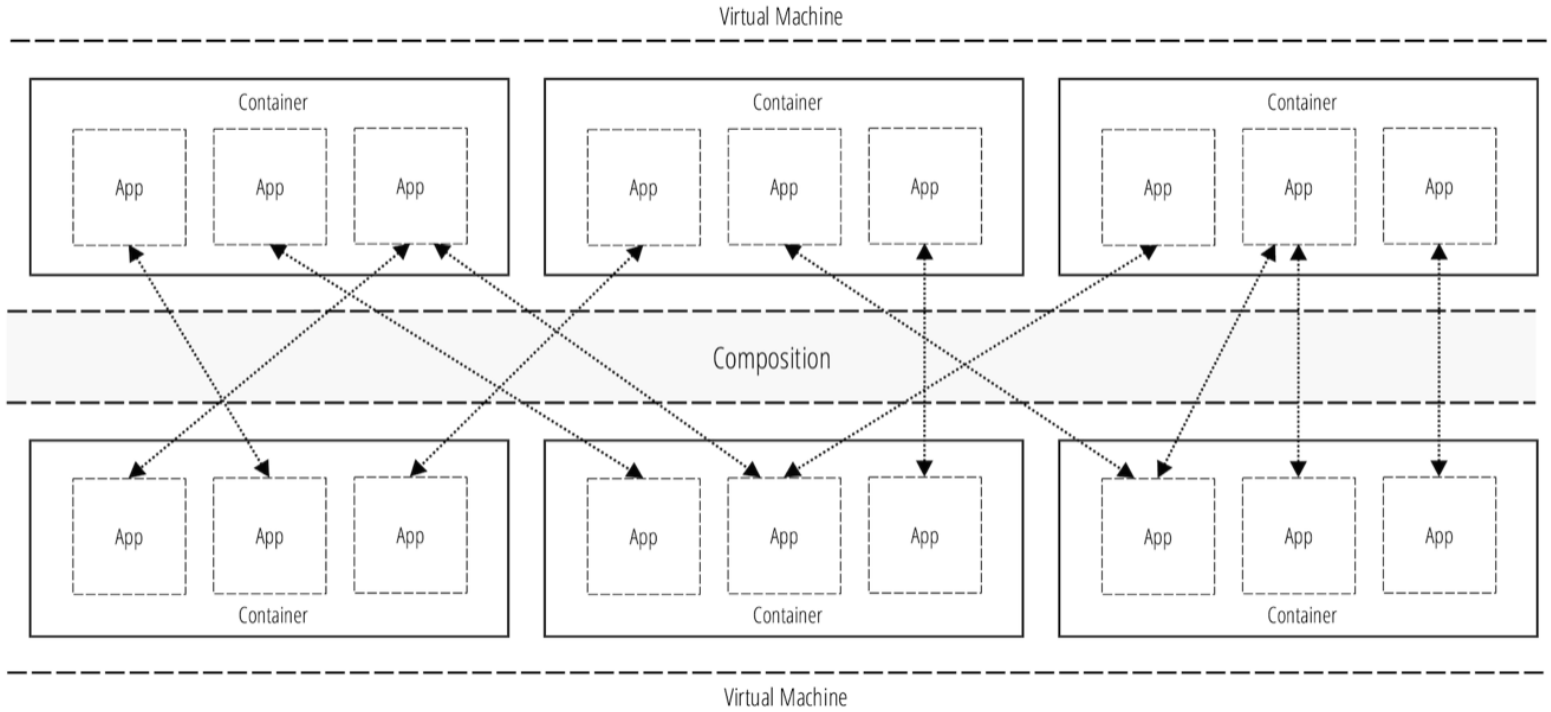
Container scheduling and auto-scaling

- Minutes to provision and start a VM, but seconds to schedule and start a container
- Auto-scaling becomes a feature of the cloud platform by scheduling on pool of VMs



Distributed Applications

- Each microservice will likely need to communicate with other containers
- Service discovery automates how distributed apps find service dependencies



Spring Cloud

A toolset designed for building distributed systems

What is Spring Cloud?

- Spring Cloud provides a way to turn Spring Boot microservices into distributed applications

@kennybastani

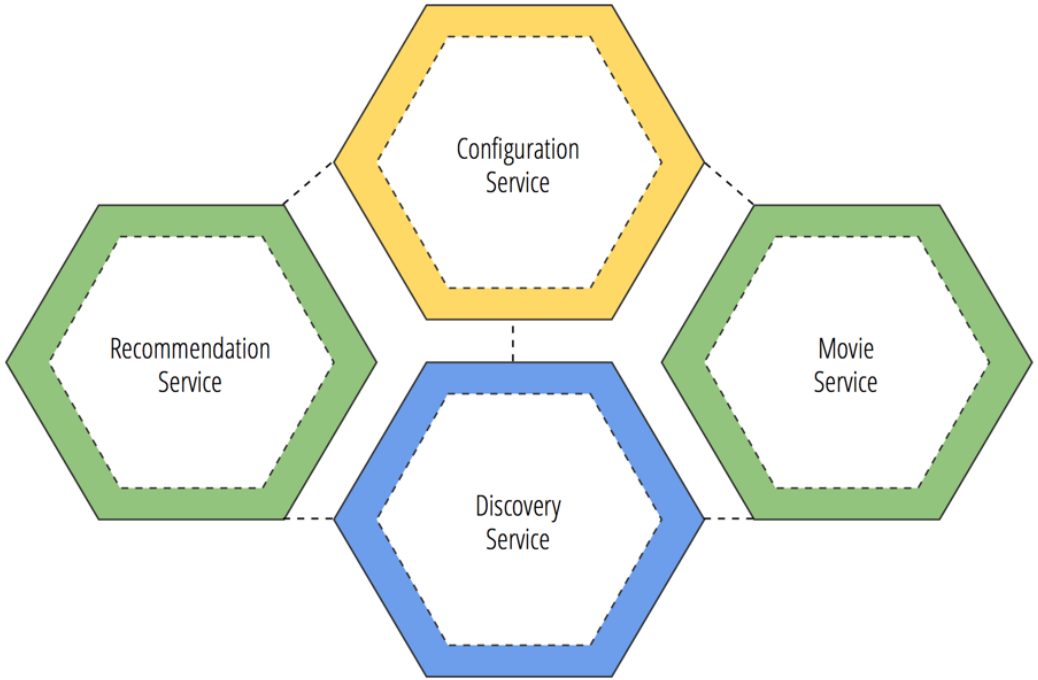
Spring Cloud Demos

Applying the patterns of cloud-native architectures
with **Spring Boot** and **Spring Cloud**

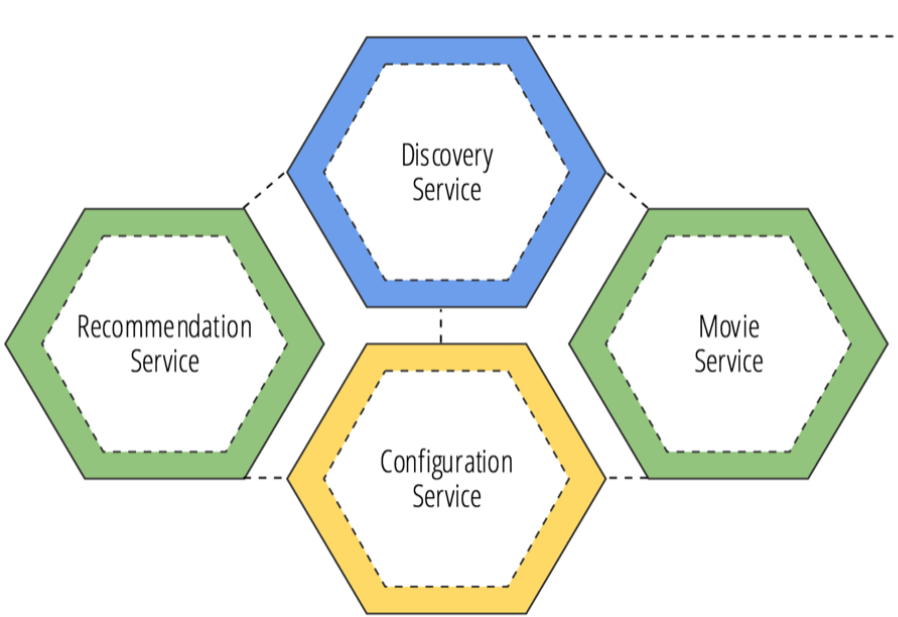
Service Discovery

- Allows applications to find each other in an environment
- An essential component when using dynamic container scheduling
- Each application handles its own routing
- Developers only need the **name** of a dependency, *not the URL*
- A *service registry* is distributed to all subscriber applications

Service Discovery



Service Registry



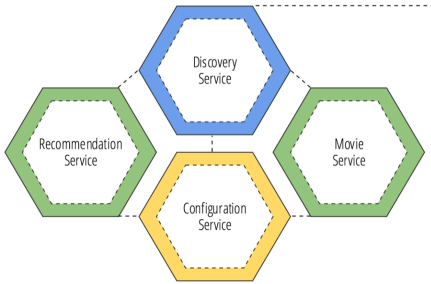
Service Registry

```
<application>
   <name>MOVIE</name>
   <instance>
      <instanceId>192.168.99.1:movie:9004</instanceId>
      <hostName>192.168.99.1</hostName>
      <app>MOVIE</app>
      <ipAddr>192.168.99.1</ipAddr>
      <status>UP</status>
      <port enabled="true">9004</port>
      <securePort enabled="false">443</securePort>
   </instance>
</application>
<application>
   <name>RECOMMENDATION</name>
   <instance>
      <instanceId>192.168.99.2:recommendation:9712</instanceId>
      <hostName>192.168.99.2</hostName>
      <app>RECOMMENDATION</app>
      <ipAddr>192.168.99.2</ipAddr>
      <status>UP</status>
      <port enabled="true">9712</port>
      <securePort enabled="false">443</securePort>
   </instance>
</application>
```

Client-side Load Balancing

User Service

User Client



Service Registry

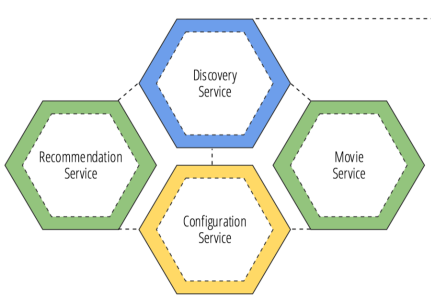
```
<application>
   <name>MOVIE</name>
   <instance>
      <instanceId>192.168.99.1:movie:9004</instanceId>
      <hostName>192.168.99.1</hostName>
      <app>MOVIE</app>
      <ipAddr>192.168.99.1</ipAddr>
      <status>UP</status>
      <port enabled="true">9004</port>
      <securePort enabled="false">443</securePort>
   </instance>
</application>
<application>
   <name>RECOMMENDATION</name>
   <instance>
      <instanceId>192.168.99.2:recommendation:9712</instanceId>
      <hostName>192.168.99.2</hostName>
      <app>RECOMMENDATION</app>
      <ipAddr>192.168.99.2</ipAddr>
      <status>UP</status>
      <port enabled="true">9712</port>
      <securePort enabled="false">443</securePort>
   </instance>
</application>
```

Client-side Load Balancing

User Service

User Service

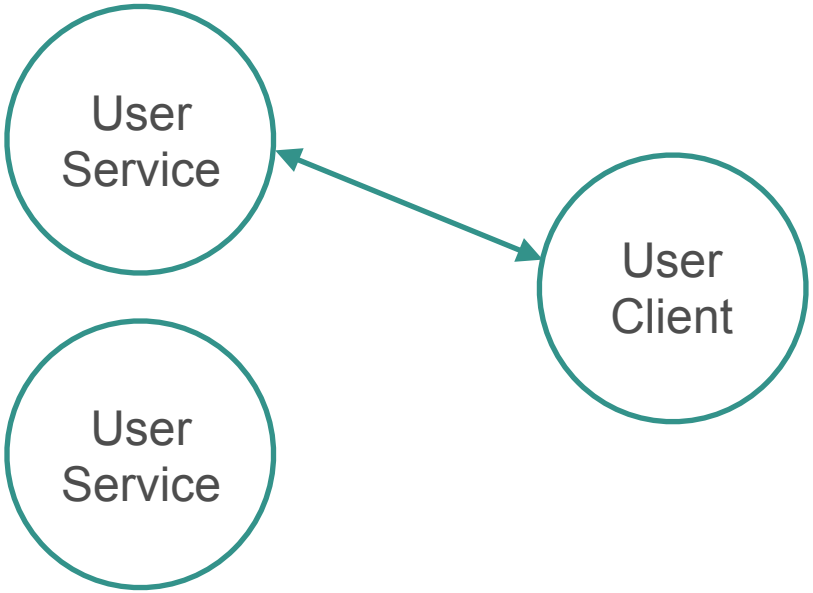
User Client

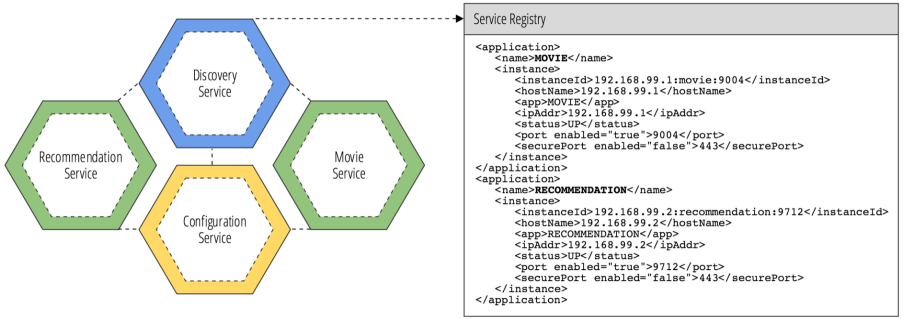


Service Registry

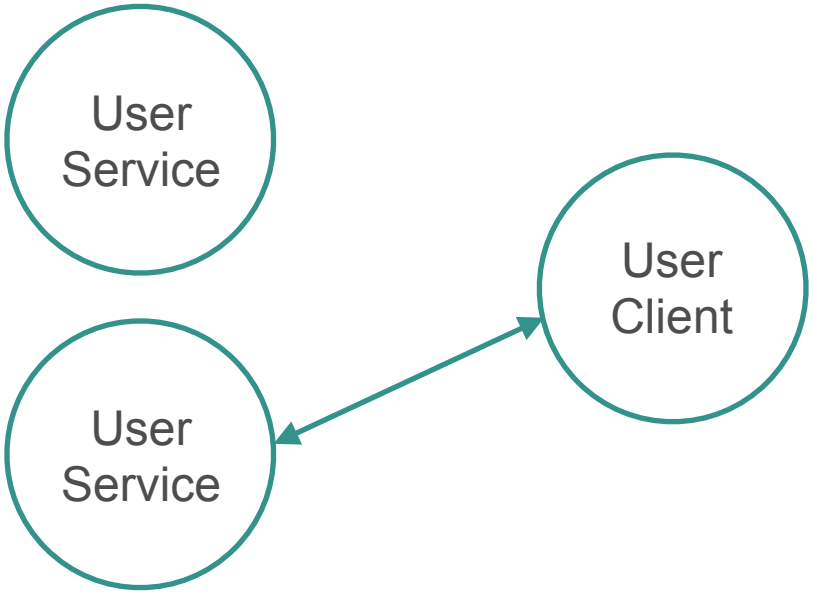
```
<application>
   <name>MOVIE</name>
   <instance>
      <instanceId>192.168.99.1:movie:9004</instanceId>
      <hostName>192.168.99.1</hostName>
      <app>MOVIE</app>
      <ipAddr>192.168.99.1</ipAddr>
      <status>UP</status>
      <port enabled="true">9004</port>
      <securePort enabled="false">443</securePort>
   </instance>
</application>
<application>
   <name>RECOMMENDATION</name>
   <instance>
      <instanceId>192.168.99.2:recommendation:9712</instanceId>
      <hostName>192.168.99.2</hostName>
      <app>RECOMMENDATION</app>
      <ipAddr>192.168.99.2</ipAddr>
      <status>UP</status>
      <port enabled="true">9712</port>
      <securePort enabled="false">443</securePort>
   </instance>
</application>
```

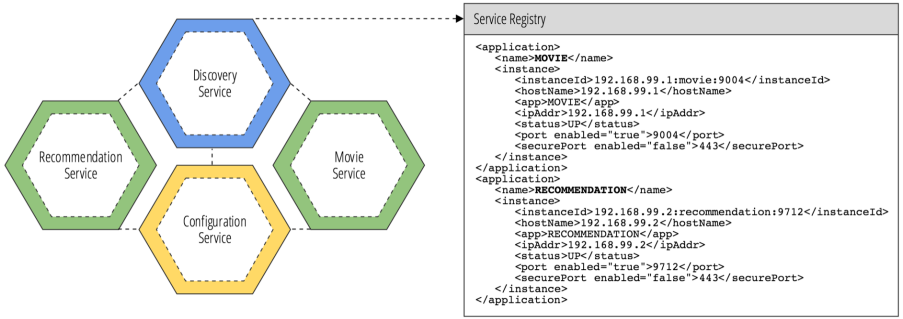
Client-side Load Balancing

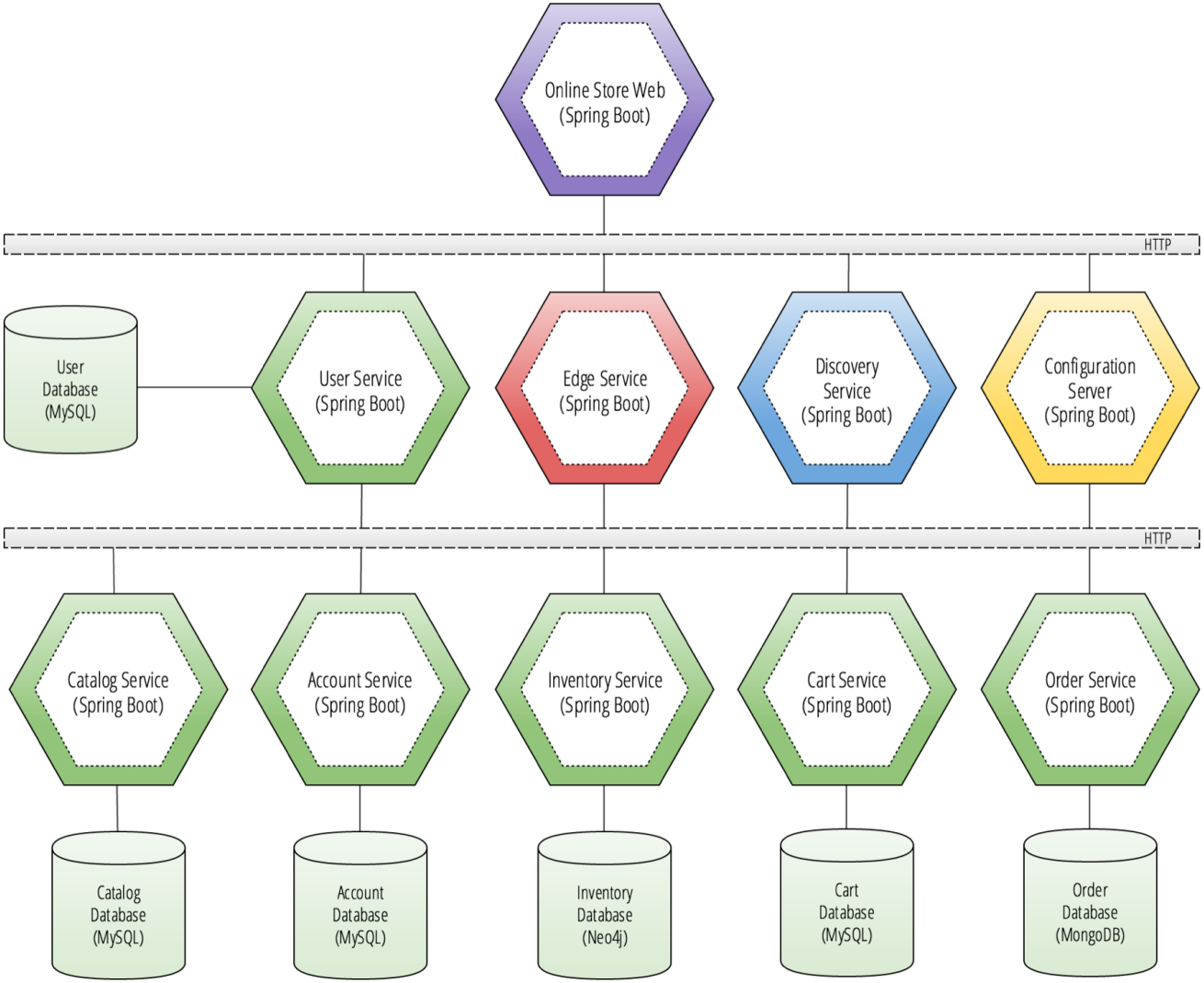




Client-side Load Balancing






Online Store Web
(Spring Boot)
HTTP
User
Database
(MySQL)
User Service
(Spring Boot)
Edge Service
(Spring Boot)
Discovery
Service
(Spring Boot)
Configuration
Server
(Spring Boot)
HTTP
Catalog Service
(Spring Boot)
Account Service
(Spring Boot)
Inventory Service
(Spring Boot)
Cart Service
(Spring Boot)
Order Service
(Spring Boot)
Catalog
Database
(MySQL)
Account
Database
(MySQL)
Inventory
Database
(Neo4j)
Cart
Database
(MySQL)
Order
Database
(MongoDB)

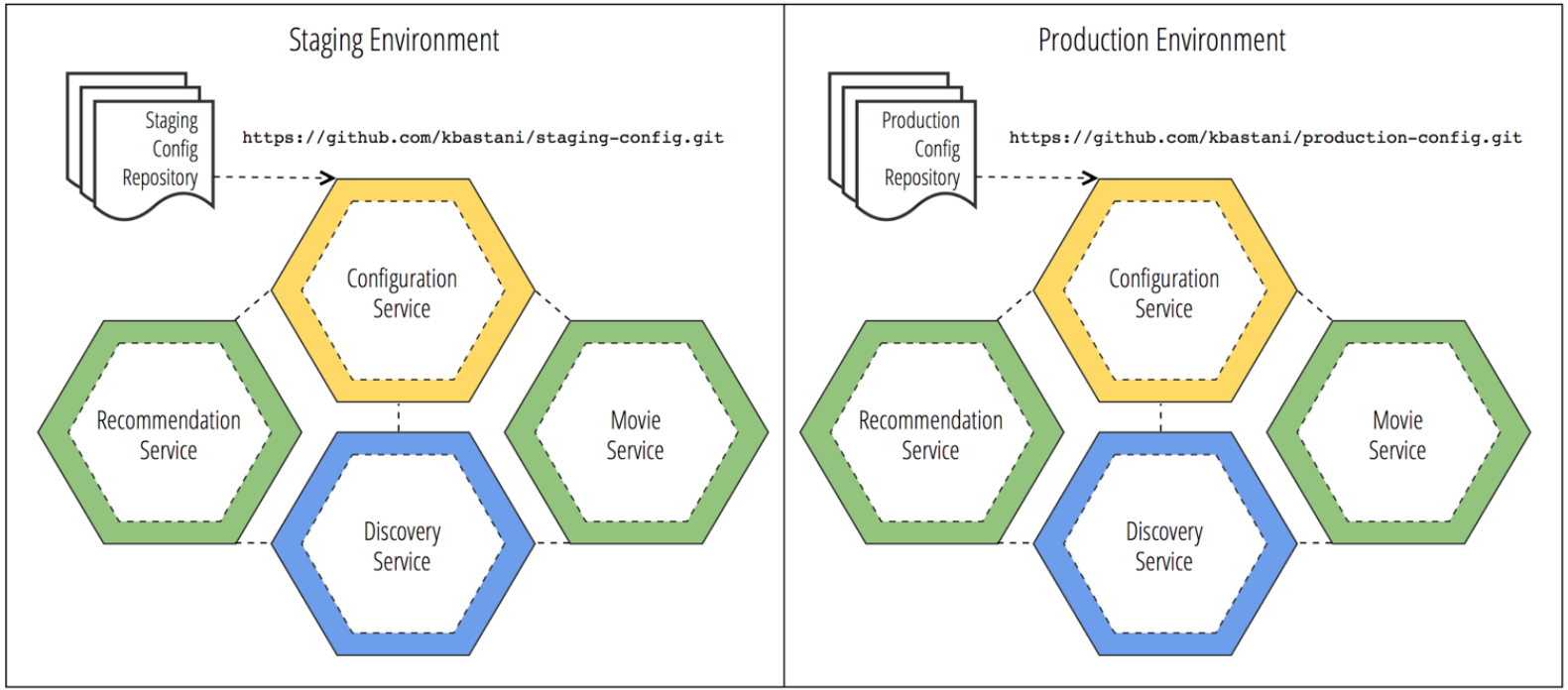
Config Server

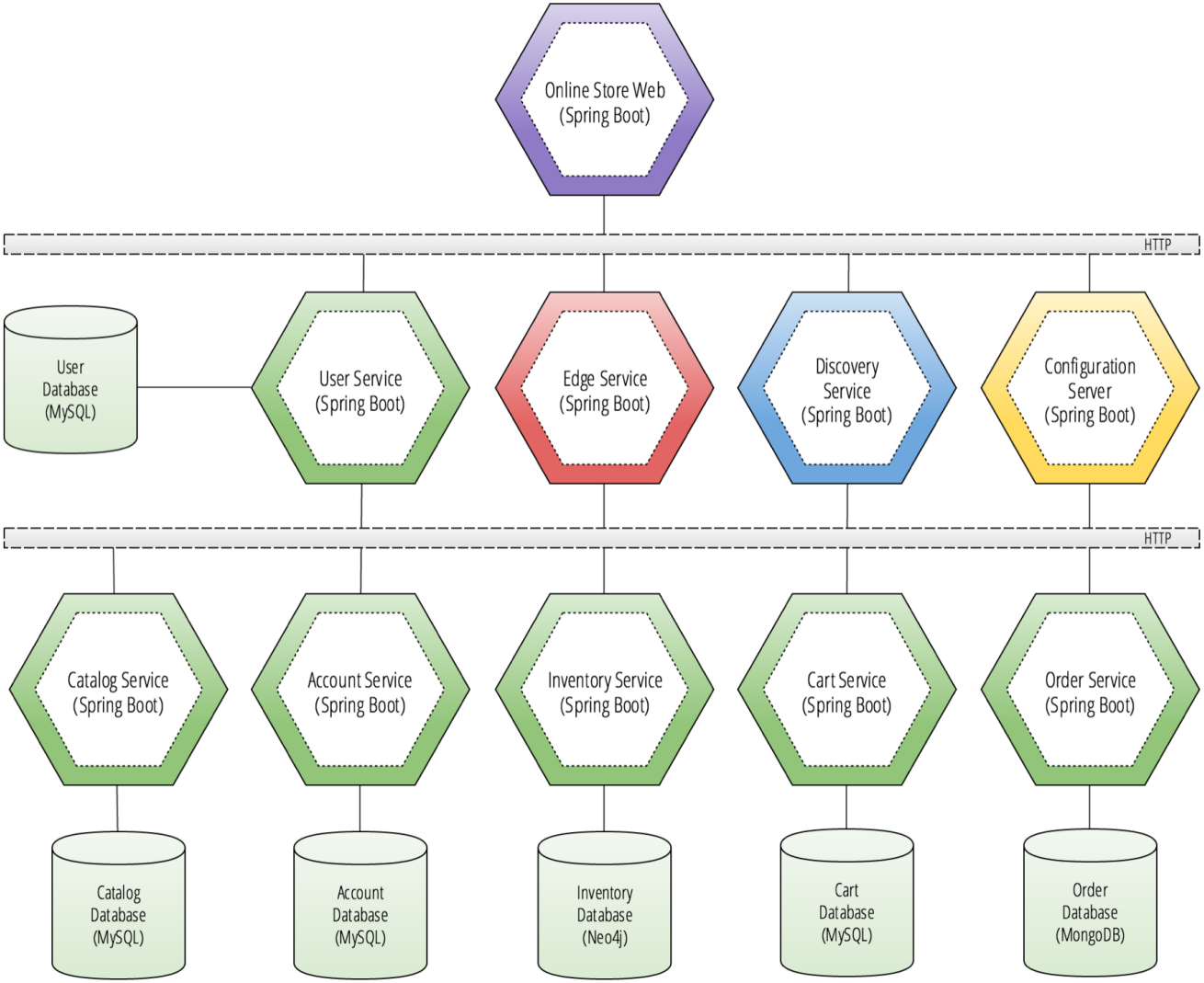
Centralizing config management for microservices with
Spring Cloud Config Server

Config Server

- Allows applications to source configuration from central service
- Application configuration can be changed without deployment
- Cascading configuration files for all apps and individual apps
- Uses Git repository as file system, providing audit log of changes
- Add **Spring Cloud Bus** to automate config refresh for many instances

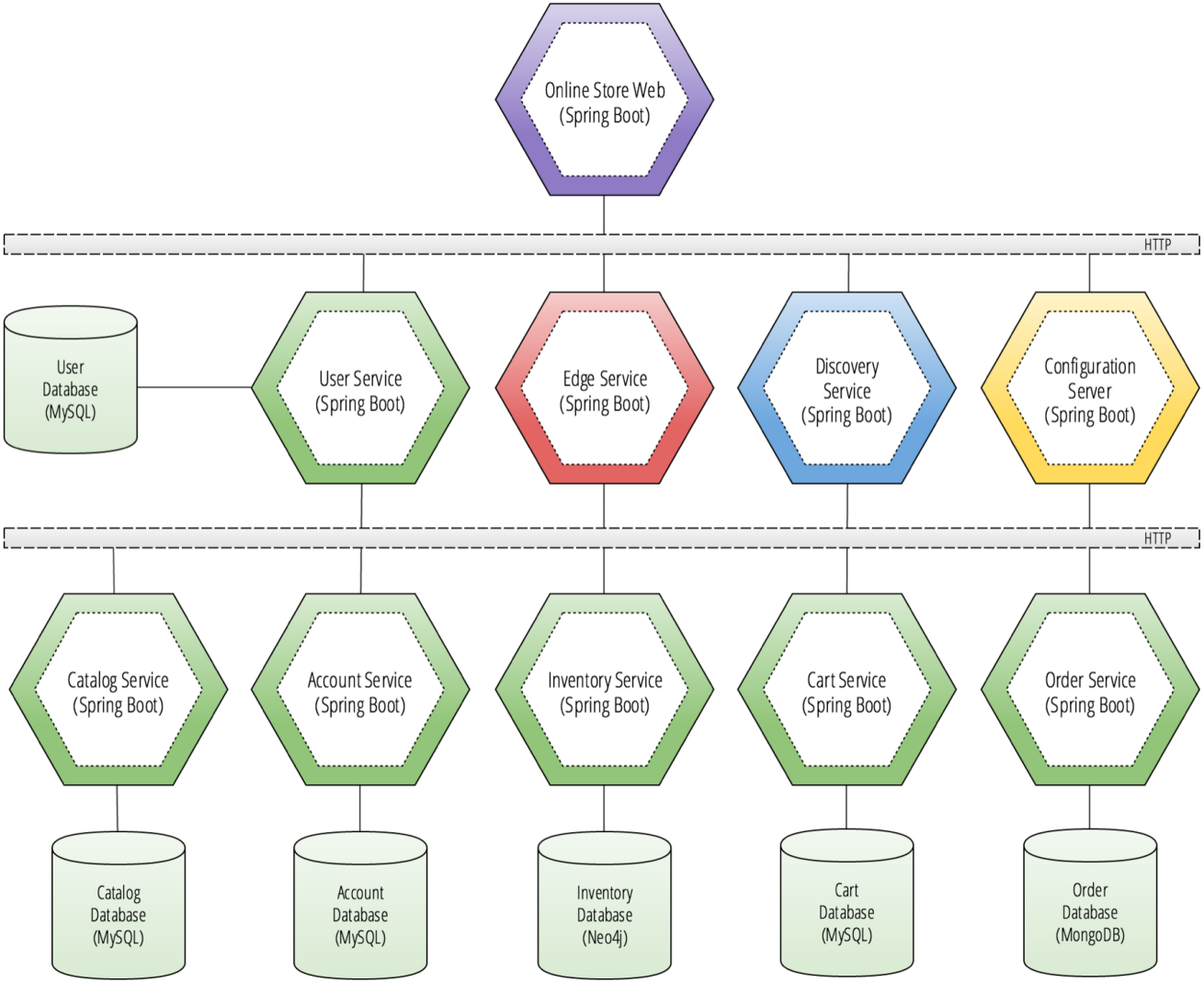
Configuration Server



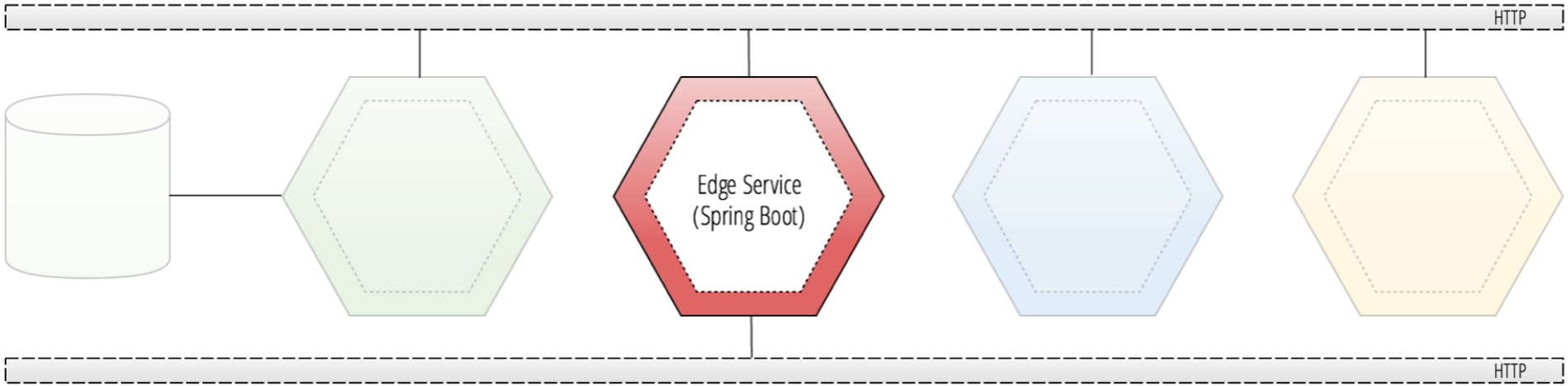

Online Store Web
(Spring Boot)
HTTP
User
Database
(MySQL)
User Service
(Spring Boot)
Edge Service
(Spring Boot)
Discovery
Service
(Spring Boot)
Configuration
Server
(Spring Boot)
HTTP
Catalog Service
(Spring Boot)
Account Service
(Spring Boot)
Inventory Service
(Spring Boot)
Cart Service
(Spring Boot)
Order Service
(Spring Boot)
Catalog
Database
(MySQL)
Account
Database
(MySQL)
Inventory
Database
(Neo4j)
Cart
Database
(MySQL)
Order
Database
(MongoDB)

API Gateway

Building edge services that route requests to backend microservices


Online Store Web
(Spring Boot)
HTTP
User
Database
(MySQL)
User Service
(Spring Boot)
Edge Service
(Spring Boot)
Discovery
Service
(Spring Boot)
Configuration
Server
(Spring Boot)
HTTP
Catalog Service
(Spring Boot)
Account Service
(Spring Boot)
Inventory Service
(Spring Boot)
Cart Service
(Spring Boot)
Order Service
(Spring Boot)
Catalog
Database
(MySQL)
Account
Database
(MySQL)
Inventory
Database
(Neo4j)
Cart
Database
(MySQL)
Order
Database
(MongoDB)

Edge Service

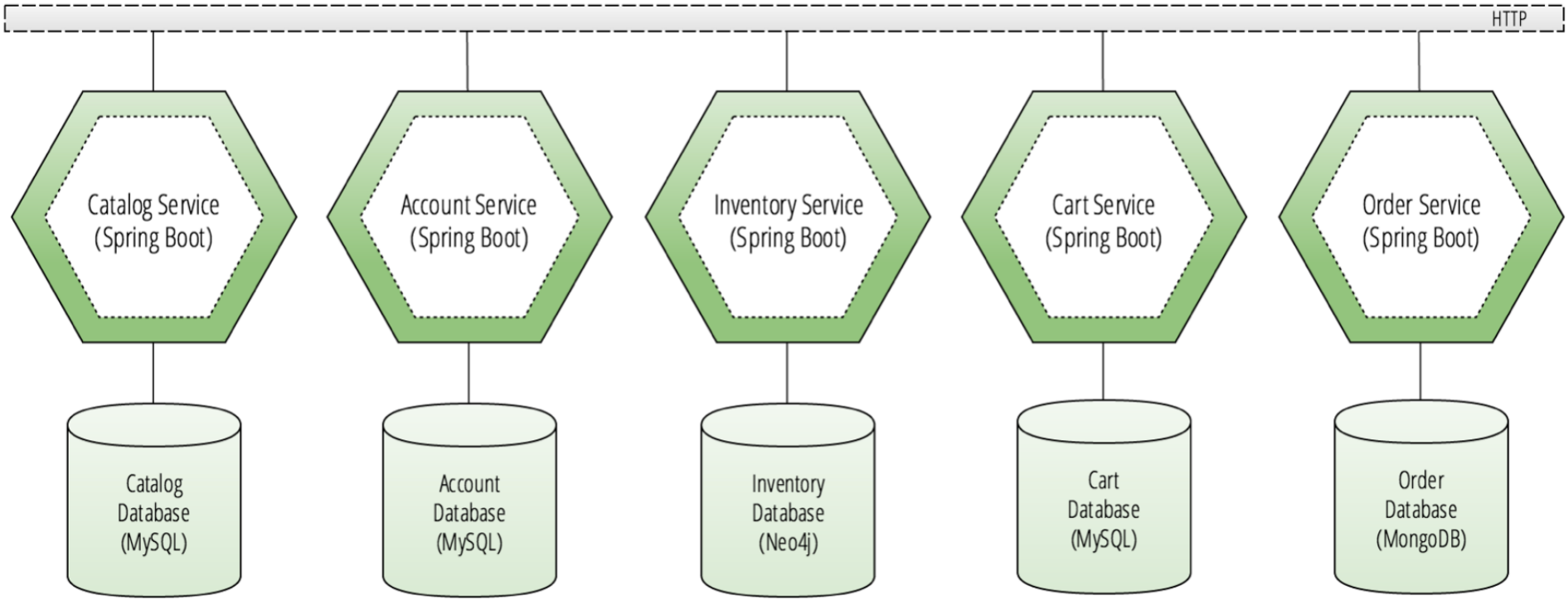


Exposes a secure reverse-proxy to back-end microservices

Edge Service

- API gateway pattern using **Spring Cloud Netflix Zuul** starter project
- Embeds relative routes from other services registered with Eureka
- Automatic method for reverse proxying to other services
- Routes are displayed at `/routes` of the Spring Boot app

Backend Microservices



Distributed Tracing

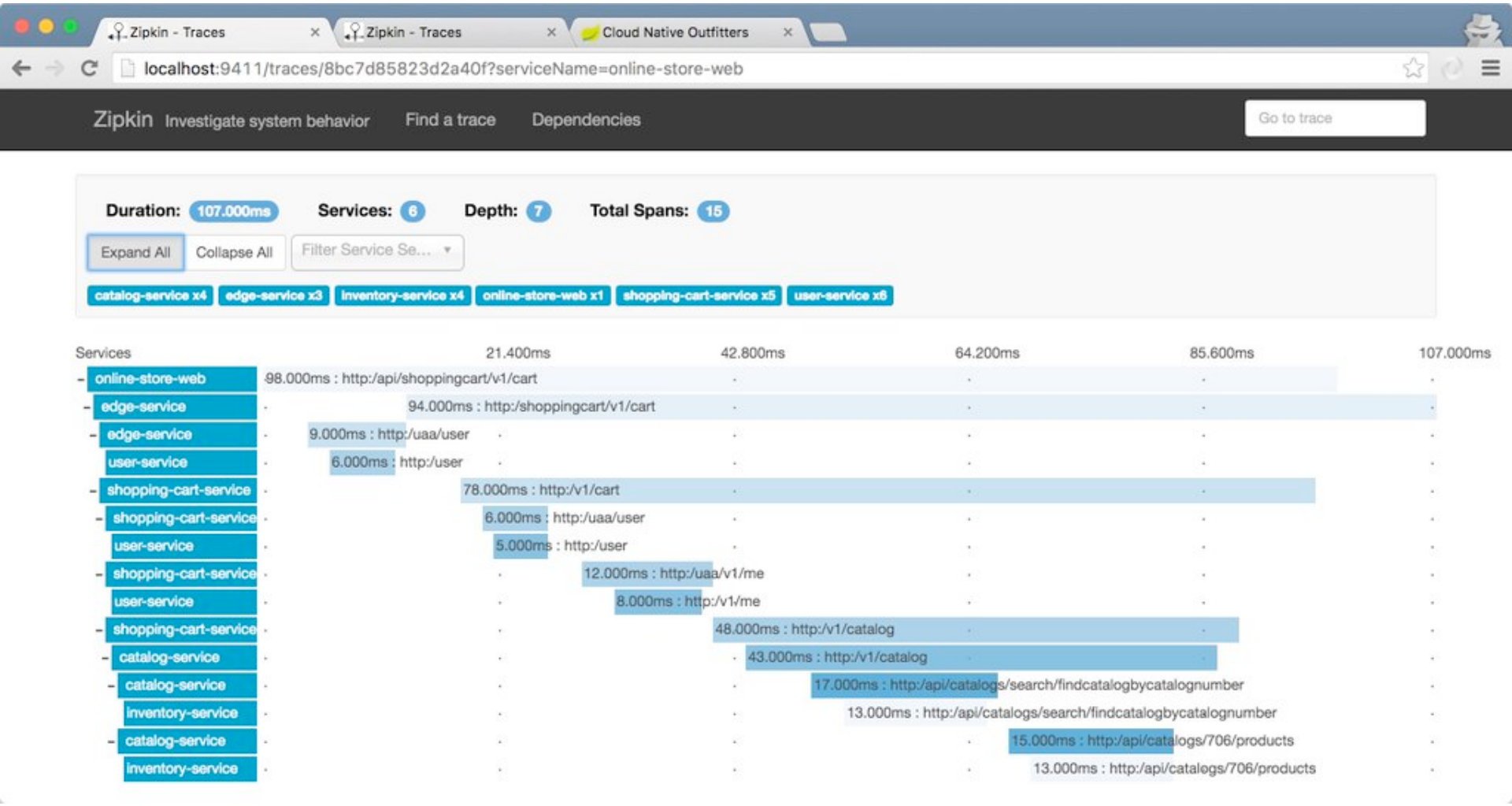
Tracing the Online Store

Spring Cloud Sleuth

- Spring Cloud Sleuth is a project that adds distributed tracing capabilities to your distributed Spring Boot applications

- https://cloud.spring.io/spring-cloud-sleuth/
- http://zipkin.io/

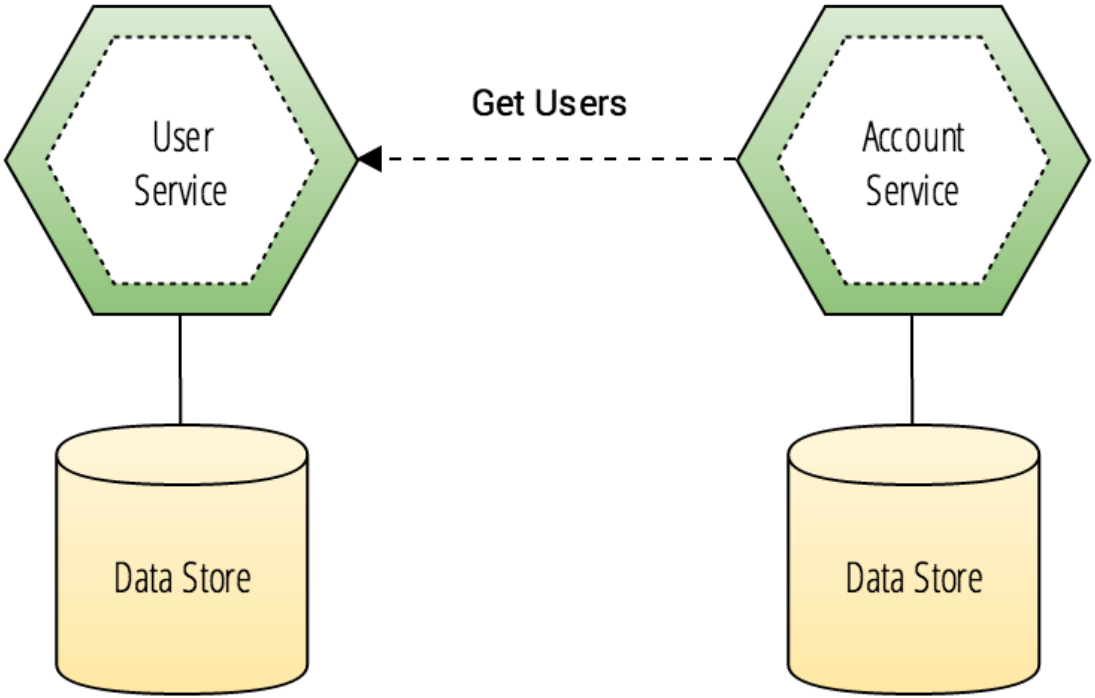
Zipkin Dashboard


Zipkin - Traces
Zipkin - Traces
Cloud Native Outfitters
localhost:9411/traces/8bc7d85823d2a40f?serviceName=online-store-web
Zipkin
Investigate system behavior
Find a trace
Dependencies
Go to trace
Duration: 107.000ms
Services: 6
Depth: 7
Total Spans: 15
Expand All
Collapse All
Filter Service Se...
catalog-service x4
edge-service x3
inventory-service x4
online-store-web x1
shopping-cart-service x5
user-service x6
Services
21.400ms
42.800ms
64.200ms
85.600ms
107.000ms
online-store-web
98.000ms : http:/api/shoppingcart/v1/cart
edge-service
94.000ms : http:/shoppingcart/v1/cart
edge-service
9.000ms : http:/uaa/user
user-service
6.000ms : http:/user
shopping-cart-service
78.000ms : http:/v1/cart
shopping-cart-service
6.000ms : http:/uaa/user
user-service
5.000ms : http:/user
shopping-cart-service
12.000ms : http:/uaa/v1/me
user-service
8.000ms : http:/v1/me
shopping-cart-service
48.000ms : http:/v1/catalog
catalog-service
43.000ms : http:/v1/catalog
catalog-service
17.000ms : http:/api/catalogs/search/findcatalogbycatalognumber
inventory-service
13.000ms : http:/api/catalogs/search/findcatalogbycatalognumber
catalog-service
15.000ms : http:/api/catalogs/706/products
inventory-service
13.000ms : http:/api/catalogs/706/products

Spring Cloud Contract

Consumer-driven contract testing

Consumer-driven Contracts



Thanks!