Pivotal

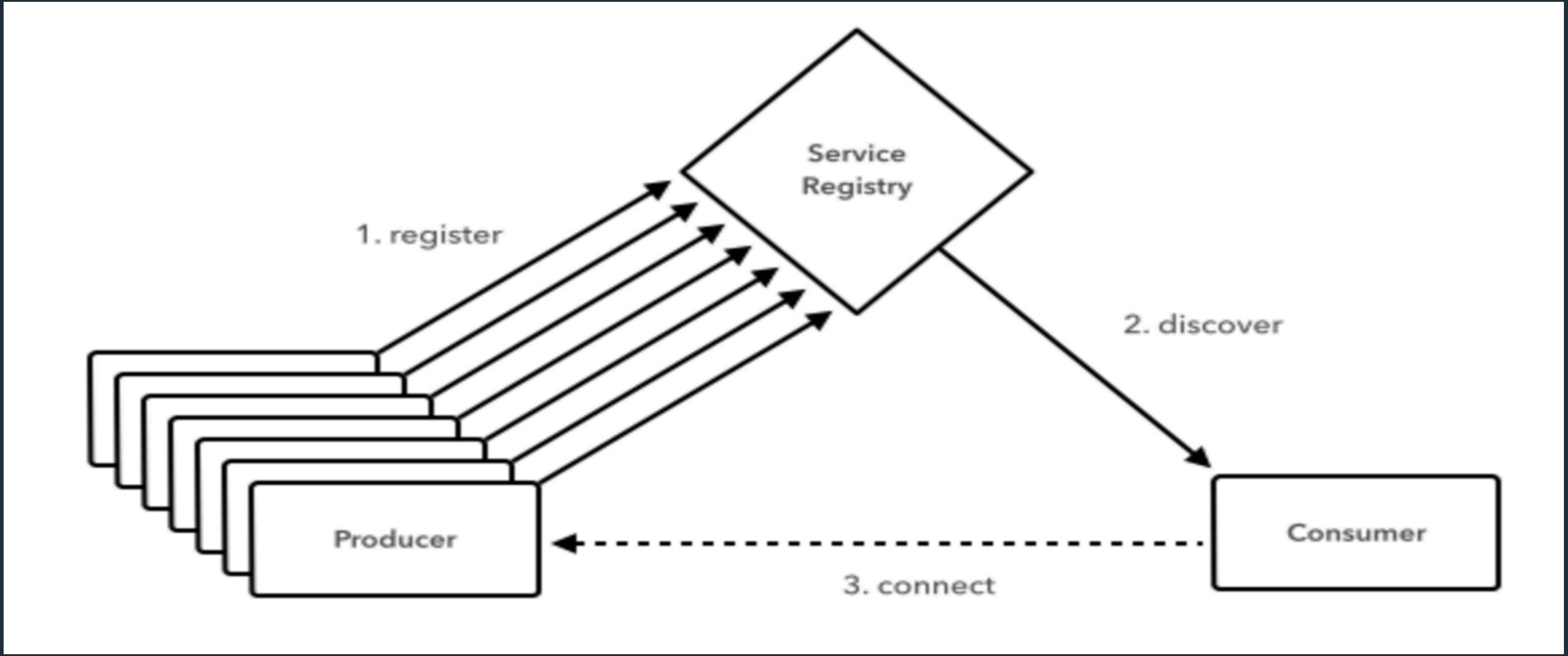
Cloud Native Applications

Spring Cloud Netflix – Service Discovery and Load Balancing

Distributed Service Challenges

- Service Discovery is one of the key tenets of a microservice based architecture.
- In distributed systems, ***dependencies != inter-process*** method call
- Trying to hand configure each client or use some form of convention can be very difficult to do and can be very brittle.
- How have we discovered services in the past?
  1. Service Locators
  2. Dependency Injection
  3. Service Registries

Service Discovery with Spring Cloud

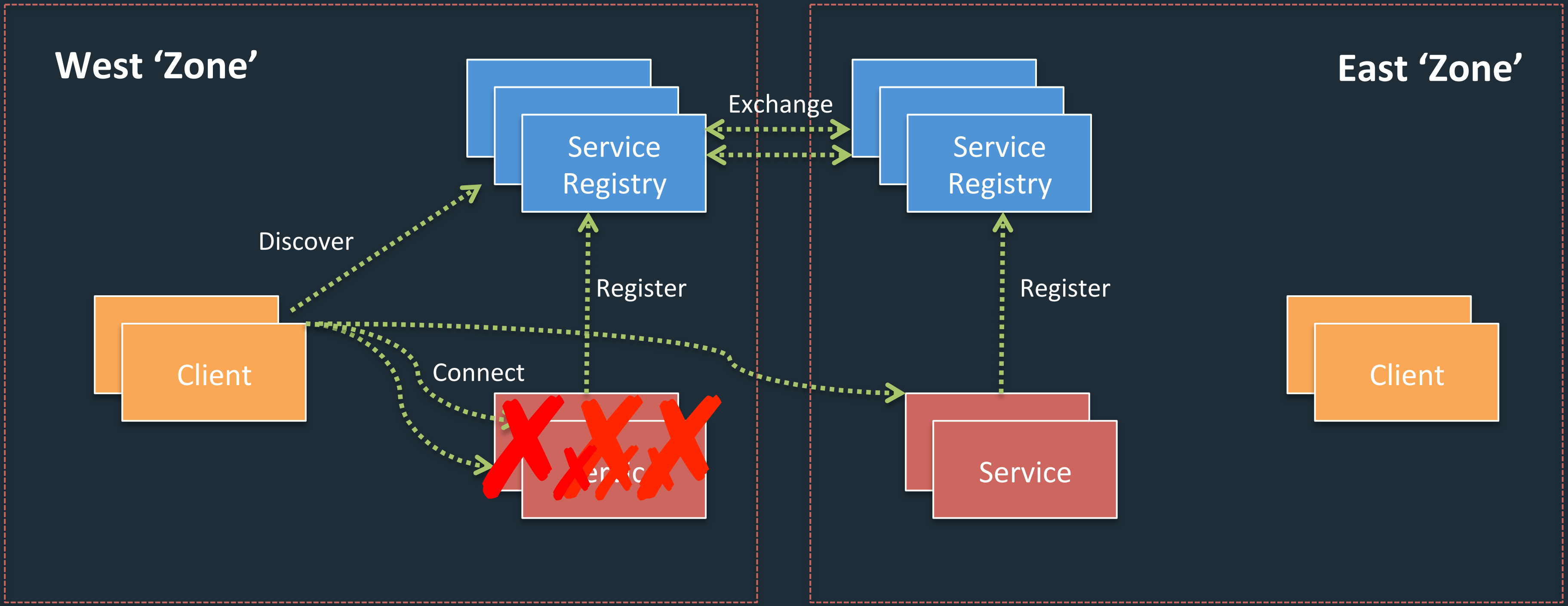


Pivotal

Spring Cloud Service Registry

- **Provides an HTTP interface + client libs for client registry/discovery**
- **Registry server collects heartbeats, maintains registry of available services/instances, exchanges registries with local peers + other “zones”**
- **Registry contains detailed information about each service**
  - **Service name, Host & port of each instance**
  - **Health indicator, URLs (health, homepage, etc)**
- **Embeddable easily in a Spring Boot application using @EnableEurekaServer and @EnableDiscoveryClient**

Service Registry – Availability Zones



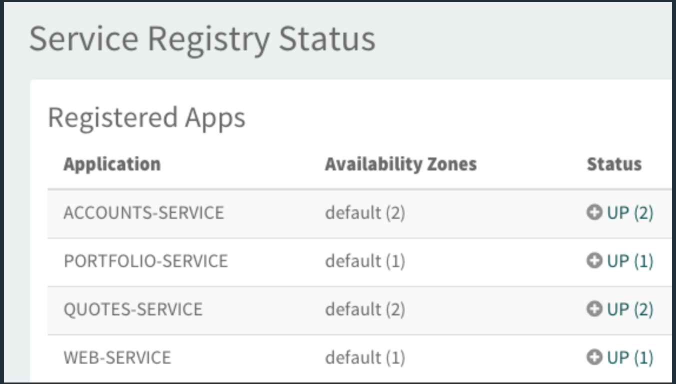
Pivotal

Client Service Discovery

```
@SpringBootApplication
@EnableDiscoveryClient  ← MAGIC!!
public class MyClientApp{
   public static void main(String[] args) {
     SpringApplication.run(MyClientApp.class, args);
   }
}
```

```
   public Portfolio accountLookup(String[acctId) {
      Portfolio p = restTemplate.getForObject(
         "http://portfolio-service/portfolio/{accId}",
         Portfolio.class
         acctId);
      return p;
   }
```

MAGIC!!


Service Registry Status

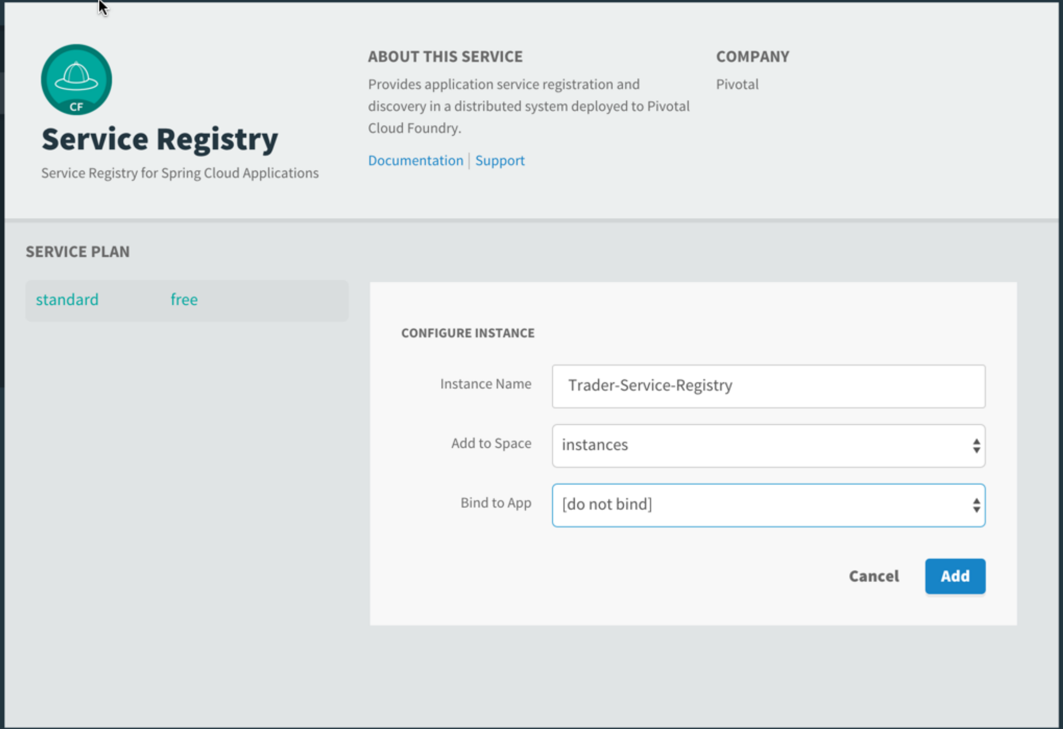
Registered Apps

Application	Availability Zones	Status
ACCOUNTS-SERVICE	default (2)	UP (2)
PORTFOLIO-SERVICE	default (1)	UP (1)
QUOTES-SERVICE	default (2)	UP (2)
WEB-SERVICE	default (1)	UP (1)

Pivotal

Spring Cloud Services: Service Registry

- Automated deployment of server component
- Security-optimized Eureka service instance using Oauth2
- Bind into CF client application(s)
- Cloud Connectors for auto-reconfiguration



Service Registry

Service Registry for Spring Cloud Applications

ABOUT THIS SERVICE

Provides application service registration and discovery in a distributed system deployed to Pivotal Cloud Foundry.

Documentation | Support

COMPANY

Pivotal

SERVICE PLAN

standard	free

CONFIGURE INSTANCE

Instance Name	Trader-Service-Registry
Add to Space	instances
Bind to App	[do not bind]

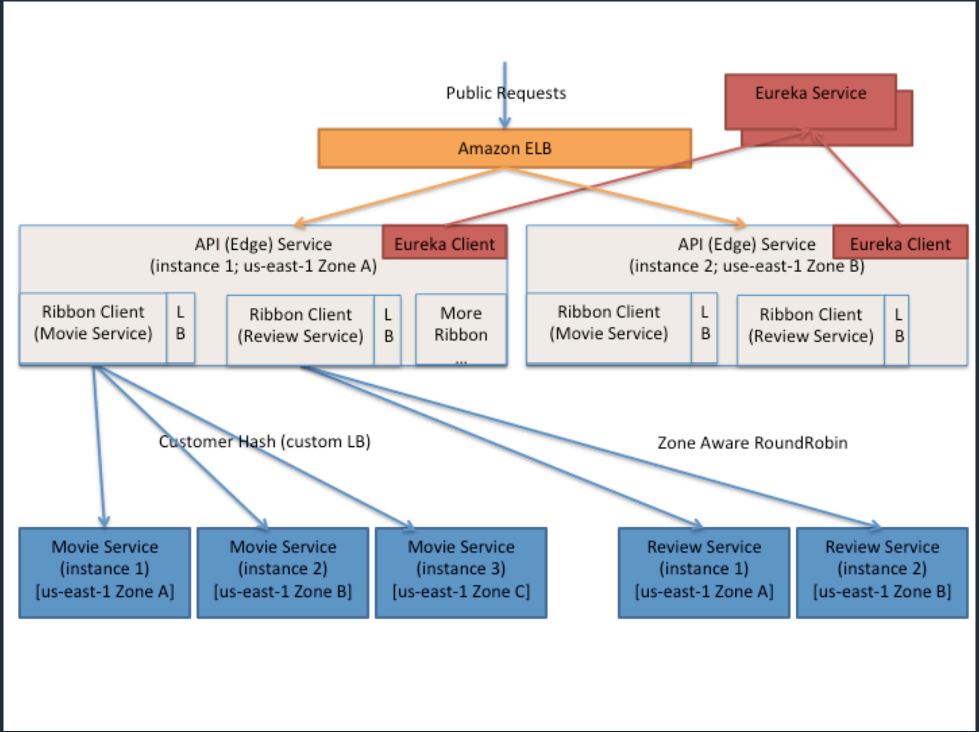
Spring Cloud: Client-side Load Balancing

- **Eureka *only* provides registry + discovery**
- **Ribbon is a *client side LB* providing control over the behavior of HTTP and TCP clients**
  - Pick right LB algorithm for client application + extensible algorithms
  - At least 1 less hop for client requests
  - Cloud-aware patterns (zones, circuit breakers, etc.)
  - No additional setup, just deploy apps
- **Zuul is JVM-based router and proxy commonly paired with Ribbon to create API gateways and reverse proxies**

Microservice API Gateways

Netflix uses Zuul and Ribbon for

- Authentication
- Stress Testing
- Canary Testing
- Dynamic Routing
- Service Migration
- Load Shedding
- Security
- Static Response handling
- Active/Active management


Public Requests
Eureka Service
Amazon ELB
API (Edge) Service (instance 1; us-east-1 Zone A)
Eureka Client
API (Edge) Service (instance 2; use-east-1 Zone B)
Eureka Client
Ribbon Client (Movie Service)
LB
Ribbon Client (Review Service)
LB
More Ribbon ...
Ribbon Client (Movie Service)
LB
Ribbon Client (Review Service)
LB
Customer Hash (custom LB)
Zone Aware RoundRobin
Movie Service (instance 1) [us-east-1 Zone A]
Movie Service (instance 2) [us-east-1 Zone B]
Movie Service (instance 3) [us-east-1 Zone C]
Review Service (instance 1) [us-east-1 Zone A]
Review Service (instance 2) [us-east-1 Zone B]

HOW??

```
@Autowired LoadBalancerClient loadBalancer;

public void doStuff() {
   ServiceInstance instance = loadBalancer.choose("stores");
   URI storesUri = URI.create(String.format("http://%s:%s",
                        instance.getHost(), instance.getPort()));
   // Do some stuff…
}
```

```
 public Portfolio accountLookup(String[acctId) {
    Portfolio p = restTemplate.getForObject(
       "http://portfolio-service/portfolio/{accId}",
       Portfolio.class
       acctId);
    return p;
 }
```

MAGIC!!

HOW??

```java
@SpringBootApplication
@EnableZuulProxy
@EnableDiscoveryClient      ← MAGIC!!
public class MyAPIGateway {

   public static void main(String[] args) {
     SpringApplication.run(MyAPIGateway.class, args);
   }
}
```

application.yml

```yaml
zuul:
   routes:
     users:
       path: /myusers/**
       serviceId: users_service
```

- **API proxy will be created at /myusers**
- **Ribbon/Zuul creates load balancer for Eureka service “users_service”**
- **All requests are executed in a Hystrix command**

Pivotal

LAB