

Service Discovery with Netflix Eureka



Spring Cloud

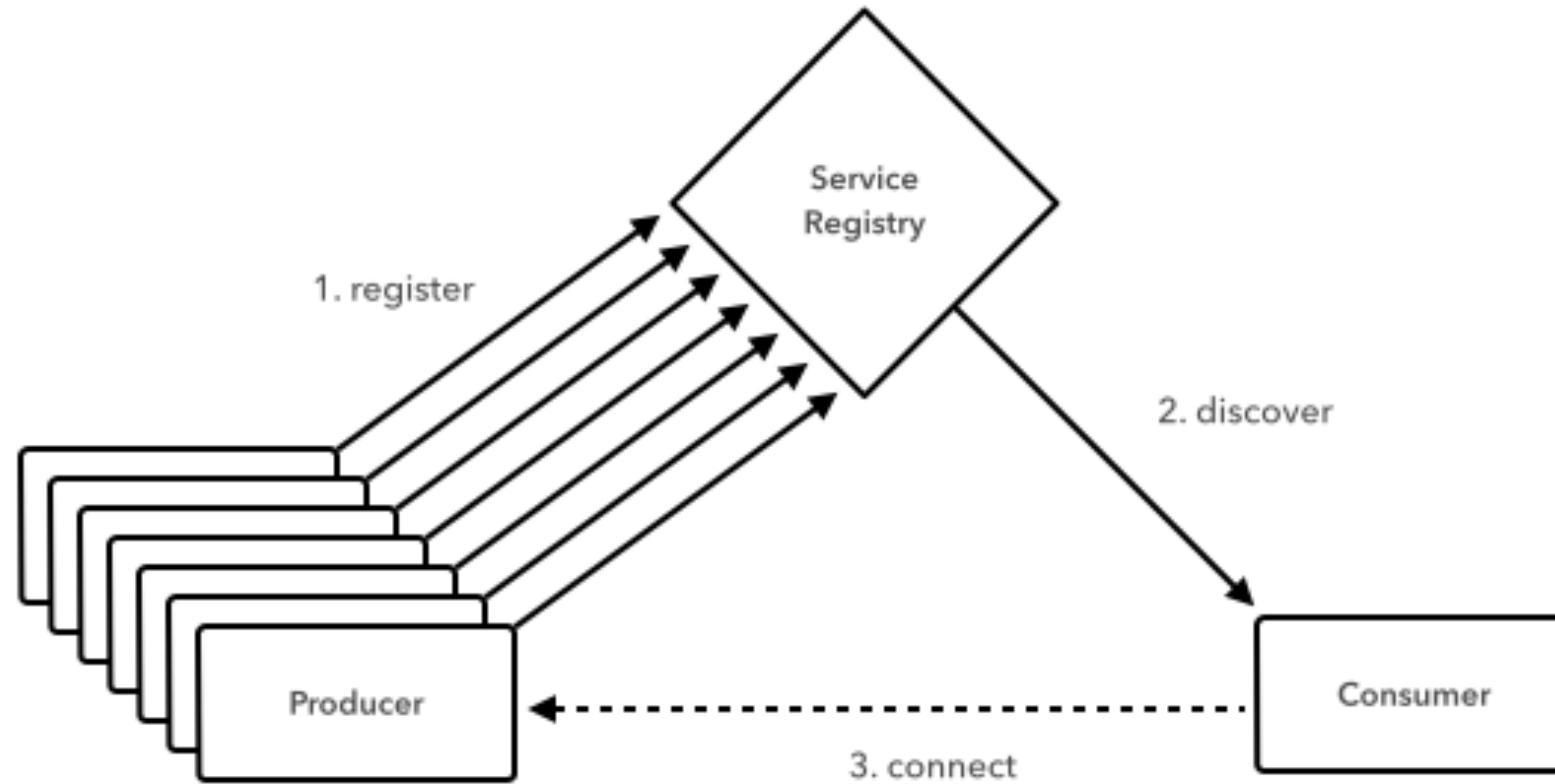
- **Spring Cloud Contract:** supports and facilitates contract testing
- **Spring Cloud Netflix:** umbrella project offering a number of Netflix services and libraries adapted for Spring:
 - Hystrix, **Eureka**, Ribbon, Feign, Zuul
- **Spring Cloud Config Server:** configuration as a service
- **Spring Cloud Sleuth:** distributed tracing



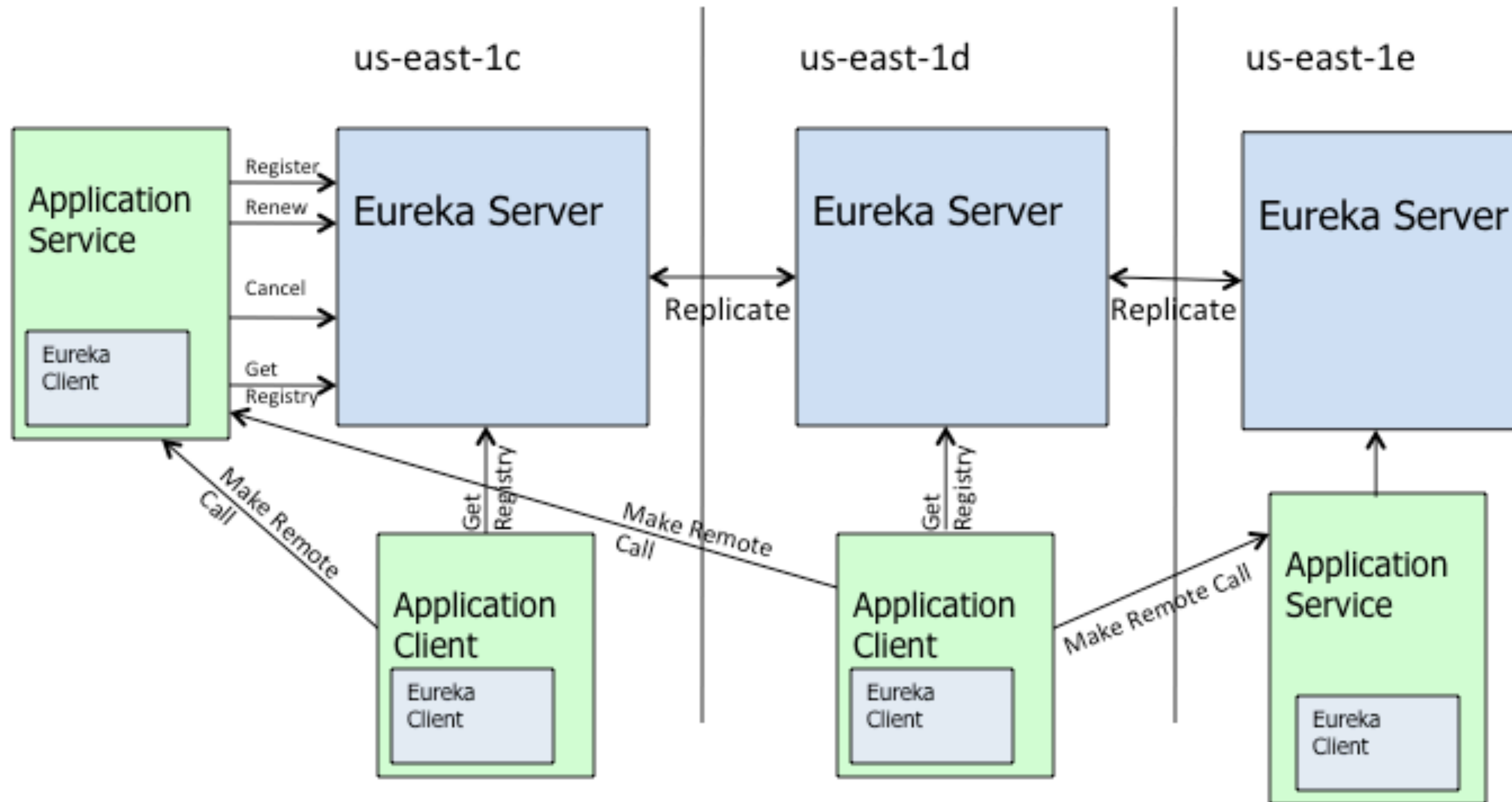
Service Registries

- A microservice architecture consists of many collaborating service instances that much know each others' address
- A cloud environment implies application instances that come and go, that are dynamically scaled
- Service registries provide dynamic application instance lookup capabilities
- Pattern prevalent in distributed systems: Service Locators, Membership Coordinators
- Examples: HashiCorp Consul, Apache ZooKeeper, *Netflix Eureka*

General Concept



Eureka Architecture



Endpoints

- Register (and de-register)
- Renew Registration
- Fetch Registry

See <https://github.com/Netflix/eureka/wiki/Eureka-REST-operations>

Register

- Service registers with eureka on startup
- With Spring Cloud, the `spring.application.name` property is used as the registration key (or virtual hostname)
- Registration can be turned off by setting configuration property `eureka.client.registerWithEureka` to false
- `eureka.client.serviceUrl.defaultZone` can be used to specify default url for contacting eureka


Renew Registration

- Services must periodically renew their registration, which would otherwise expire
- aka “Heartbeats”
- The configuration property `eureka.instance.leaseRenewalIntervalInSeconds` governs how often a service renews their registration

Fetch Registry

- Clients fetch a copy of the registry periodically
- An optimization, allows lookups to be performed directly against a cached copy
- `eureka.client.fetchRegistry` can be used to control whether to fetch the registry
- `eureka.client.registryFetchIntervalSeconds` controls how frequently to fetch a new copy

The Eureka Dashboard



HOME LAST 1000 SINCE STARTUP

System Status

Environment	test	Current time	2017-11-21T13:13:25 -0600
Data center	default	Uptime	00:00
		Lease expiration enabled	false
		Renews threshold	5
		Renews (last min)	0

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
FORTUNE	n/a (1)	(1)	UP (1) - eitans-mbp:fortune:8081
GREETING	n/a (1)	(1)	UP (1) - eitans-mbp:greeting

Configuring a eureka instance or client

1. Add build dependency: `spring-cloud-starter-eureka`
2. Configure service with `spring.application.name` property
3. Annotate Spring Boot Application class with `@EnableDiscoveryClient`
4. Clients auto-wire a `EurekaClient` instance

Eureka Lookup Example

```
29 String getFortune() {
30     String fortuneUrl = lookupUrlFor( appName: "FORTUNE");
31     Map map = restTemplate.getForObject(fortuneUrl, Map.class);
32     return (String) map.get("fortune");
33 }
34
35 private String lookupUrlFor(String appName) {
36     InstanceInfo instance = eurekaClient.getNextServerFromEureka(appName, secure: false);
37     return instance.getHomePageUrl();
38 }
39
```