Spring Cloud

Written by Rony Keren
John Bryce Training Center

JOHN BRYCE
Leading in IT Education
a matrix company

Introduction

JOHN BRYCE
Leading in IT Education
a matrix company

- provides tools for out-of-the-box Microservice platform solutions
- Built on Spring-boot & Spring MVC
- Provides both abstractions & implementations
- Allows rapidly wrap a micro-service with all / most important capabilities in order to integrate into Microservice ecosystems
- Done mostly via annotations

Spring Cloud provides Cloud Native application style

Features

JOHN BRYCE
Leading in IT Education
a matrix company

- Distributed/versioned configuration
- Service registration and discovery
- Routing
- Service-to-service calls
- Load balancing
- Circuit Breakers
- Global locks
- Leadership election and cluster state
- Distributed messaging

Projects

JOHN BRYCE
Leading in IT Education
a matrix company

- Configuration server with GIT
- Netflix (Eureka, Hystrix, Zuul…)
- Cloud Bus
- Cloud Foundry (SSH & oAuth2 support)
- Cloud Cluster (Zookeeper, Redis, Hazelcast, Consul)
- Cloud Data Flow (data composing)
- Cloud Stream (external resources – Kafka & RabbitMQ)
- Cloud Zookeeper (service discovery)
- Cloud Gateway (based on Project Reactor programmatic routing)
- Load balancing - Ribbon

Spring Cloud Contexts

JOHN BRYCE
Leading in IT Education
a matrix company

- Bootstrap application context
  - Is the absolute root context
  - Is the parent of the main application
  - Populates and shares environmental information (Environment)
    - Bootstrap properties cannot be overridden by application
    - Holds configuration info
    - Configuration files: bootstrap.properties / bootstrap.yml (instead of 'application')
  - Usually, loads remote configurations from Configuration Server
- ApplicationContext
  - Usually a direct sibling of Bootstrap Application Context
    - Use parent() to obtain bootstrap from application context
  - By default, application configuration is used when not found in bootstrap

Spring Cloud Contexts

JOHN BRYCE
Leading in IT Education
a matrix company

- Generally, client Microservices use bootstrap.yml / bootstrap.properties
  - Bootstrap configuration allows getting conf. info from a remote repository
  - Bootstrap context will help in generating service application context

- Bootstrap configuration holds local configuration as well
  - Used in a case where remote configuration is unavailable

- The process:

Client-Microservice→configuration-server→GIT

if config server not available:

Client-Microservice→local configuration

Spring Cloud + Netflix

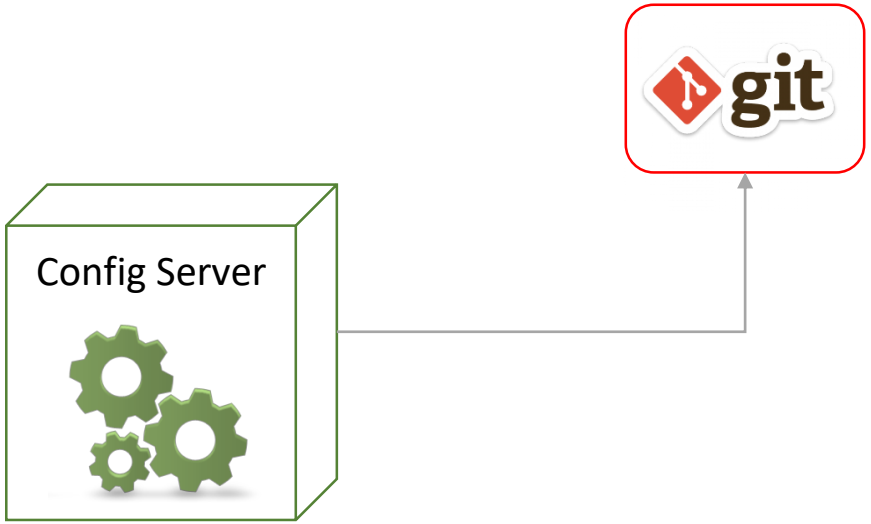
JOHN BRYCE
Leading in IT Education
a matrix company

Common out-of-the-box solutions:

- Configuration server
- Service discovery - Eureka
- API gateway - Zuul
- Circuit breaker (load-balancing) – Hystrix
- Monitoring – Hystrix
- oAuth2 support Zuul
- Client-side Load balancing - Ribbon

Configuration Server

- Serves externalized configuration in a distributed system
- Provides server and client-side support
- Default implementation uses GIT to download configuration info



Configuration Server

JOHN BRYCE
Leading in IT Education
a matrix company

- Config Server

  Maven dependencies:

pom.xml

```
...
<dependencies>
        <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-config-server</artifactId>
        </dependency>
 </dependencies>
 <dependencyManagement>
        <dependencies>
                <dependency>
                        <groupId>org.springframework.cloud</groupId>
                        <artifactId>spring-cloud-dependencies</artifactId>
                        <version>Edgware.SR2</version>
                        <type>pom</type>
                        <scope>import</scope>
                </dependency>
        </dependencies>
 </dependencyManagement>
...
```

dependency management takes care for the jar dependency chain

Configuration Server

JOHN BRYCE
Leading in IT Education
a matrix company

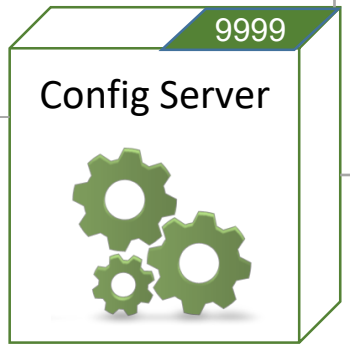
application.yml

```
spring:
  cloud:
    config:
      server:
          git :
            uri: https://github.com/spring-cloud-repo
server:
  port: 9999
```

ConfigServer.java

```
@SpringBootApplication
@EnableConfigServer
 public class ConfigServer {
   public static void main(String[] args) {
       SpringApplication.run(ConfigServer.class, args);
   }
}
```

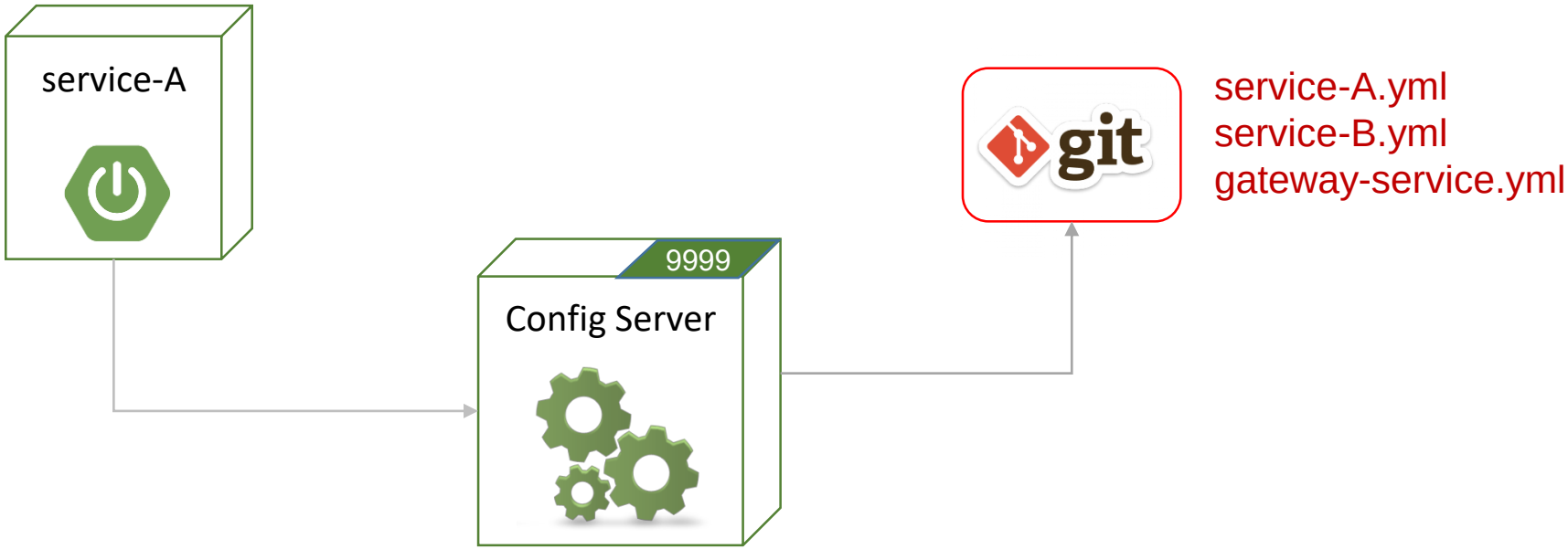
service-A.yml
service-B.yml
gateway-service.yml



Configuration Server

Any service that uses Config-server will:

- Try to get *.yml / *.properties loaded from GIT
- If fails to connect to the Config-Server, uses its own local configuration



Configuration Server

Microservices are set to connect to config server

or use local configuration

bootstrap.yml

```
spring:
  application:
    name: service-A
  cloud:
    config:
      uri: http://localhost:9999
server:
  host: localhost
  port: 7001
```

service-A

8001/7001

service-A.yml (port:8001)

MicroserviceA.java

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

9999

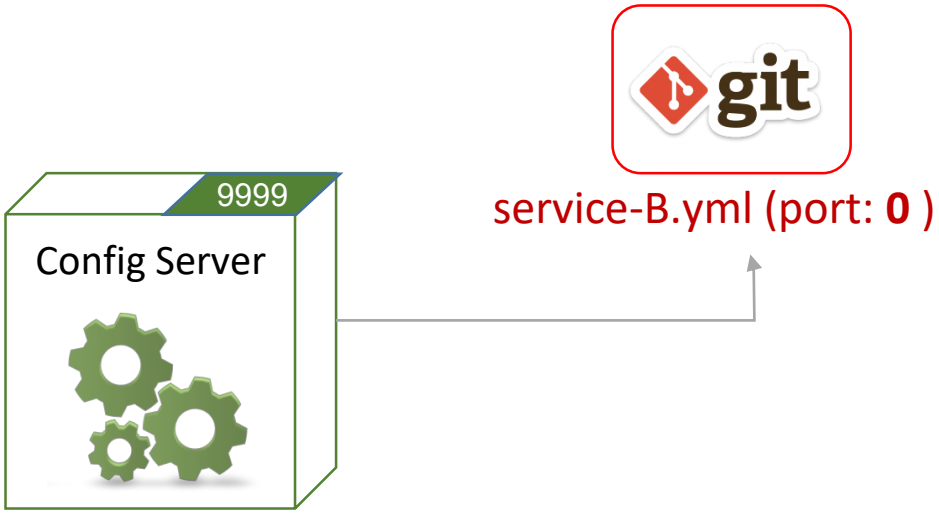
Config Server

Configuration Server

Running multiple Microservice instances

Assign zero value to port

Configuration server randomizes values



Configuration Server

JOHN BRYCE
Leading in IT Education
a matrix company

- Client Microservice

Maven devendencies:

pom.xml

```
...
<dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-config</artifactId>
</dependency>
 ...
```

Configuration Server

JOHN BRYCE
Leading in IT Education
a matrix company

- Config Server

test server configuration:

http://localhost:9999/service-A/master

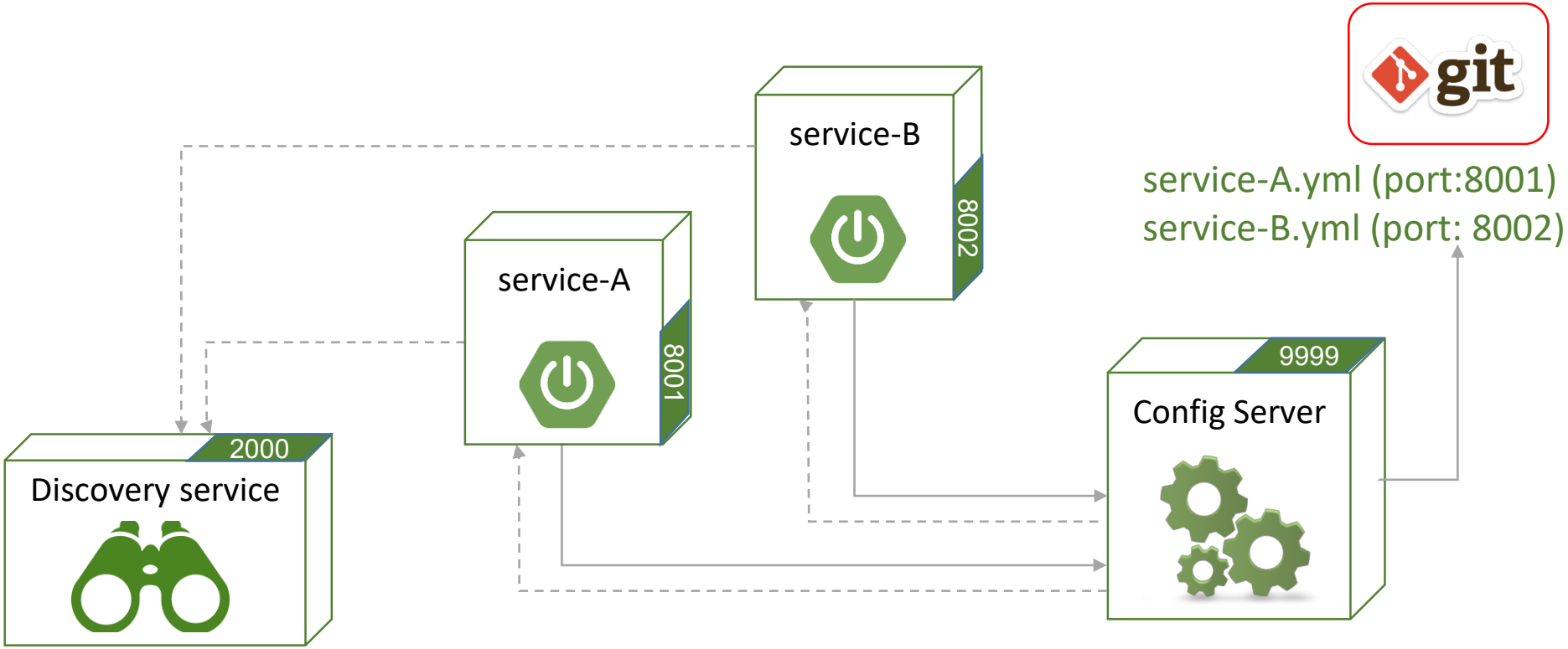
http://localhost:9999/service-B/master

- Information is returned in JSON format

Service Discovery - Eureka

JOHN BRYCE
Leading in IT Education
a matrix company

- Service discovery receives notifications from other services
- Maintains health & state



Service Discovery - Eureka

JOHN BRYCE
Leading in IT Education
a matrix company

Enabling Eureka discovery service in Spring Boot

DiscoverServerApplication.java

```
@EnableEurekaServer
@SpringBootApplication
public class DiscoverServerApplication {
    public static void main(String[] args) {
        SpringApplication.run(DiscoverServerApplication.class, args);
    }
}
```

application.yml

```
server:
  port: 2000
```

Service Discovery - Eureka

JOHN BRYCE
Leading in IT Education
a matrix company

- Eureka clients are
  - Our Microservices
  - Eureka client – embedded client (requires URL configuration)

- Discovery server may be configured as:
  - Instance & Client
    - default
    - Runs continues intensive checks with embedded client
  - Instance –maintains available Microservices client list

Configuration:

application.yml

```
eureka:
  client:
    registerWithEureka: false
    fetchRegistry: false
    service-url:
      defaultZone: http://localhost:2000
server:
  port: 2000
```

Service Discovery - Eureka

JOHN BRYCE
Leading in IT Education
a matrix company

- Discovery server

Maven dependencies:

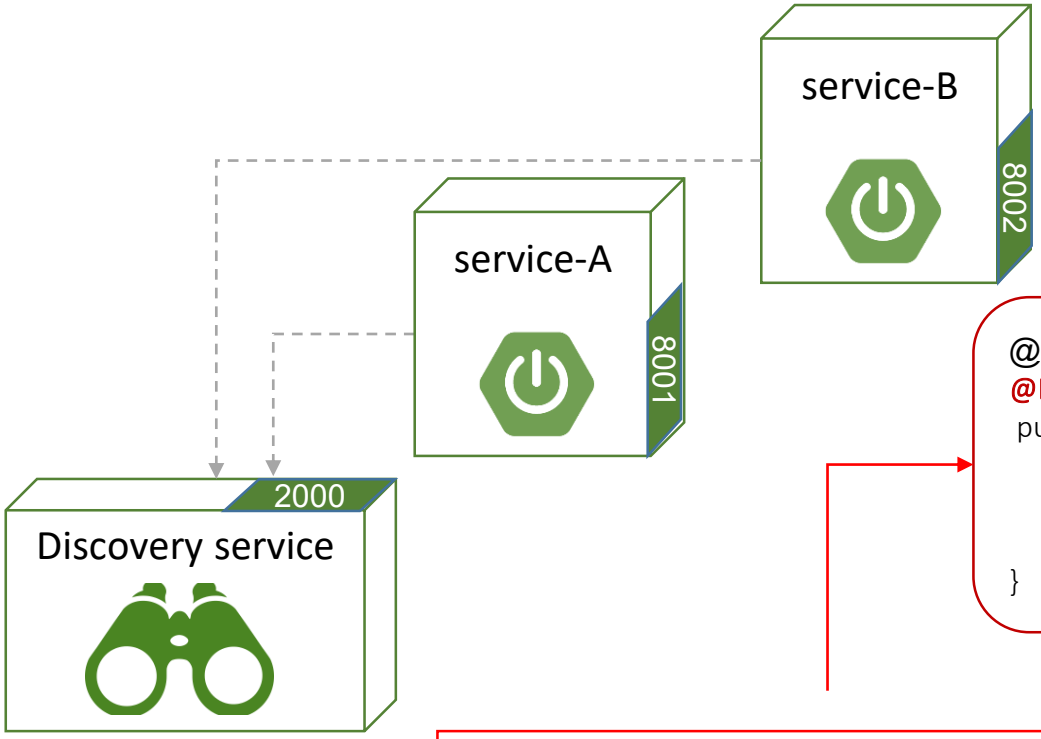
pom.xml

```
...
<dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-netflix-eureka-server</artifactId>
</dependency>
 ...
```

Service Discovery - Eureka

JOHN BRYCE
Leading in IT Education
a matrix company

- Microservices register to Discovery service
  - Configuration (local and/or on GIT):



service-A.yml & service-B.yml

```
server:
   port: ....
eureka:
   client:
      serviceUrl:
         defaultZone: http://localhost:2000/eureka
```

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

Note:

@EnableDiscoveryService is more generic and supported by Eureka
@EnableEurekaClient does the same but is supported only by Eureka

Service Discovery - Eureka

JOHN BRYCE
Leading in IT Education
a matrix company

- Client Microservice discovery enabling Maven dependencies:

pom.xml

```
...
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-eureka</artifactId>
</dependency>
...
```

Service Discovery - Eureka

JOHN BRYCE
Leading in IT Education
a matrix company

After starting discovery server we can monitor running services

Eureka

localhost:2000

spring Eureka

HOME LAST 1000 SINCE STARTUP

System Status

Environment	test
Data center	default

Current time	2018-04-18T17:25:28 +0300
Uptime	00:02
Lease expiration enabled	false
Renews threshold	5
Renews (last min)	0

DS Replicas

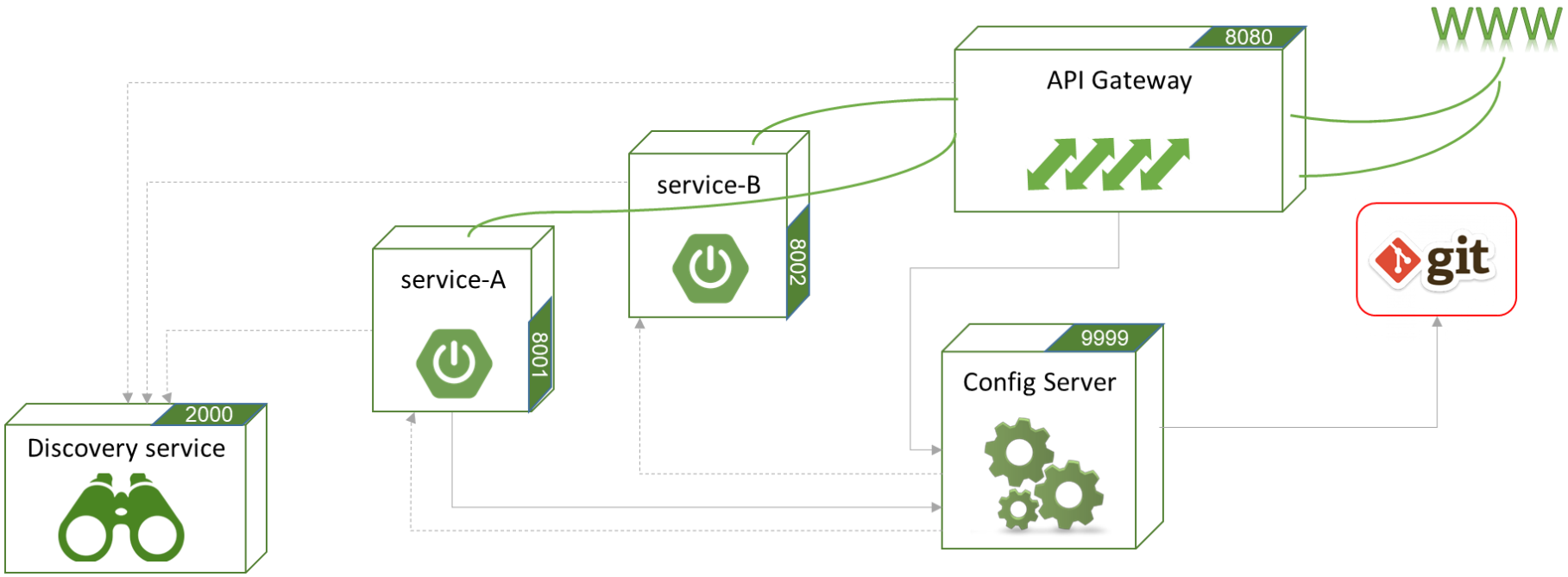
localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status	
MICROSERVICE-A	n/a (1)	(1)	UP (1) -	Admin-PC Microservice-A7001
MICROSERVICE-B	n/a (1)	(1)	UP (1) -	Admin-PC Microservice-B:7002

API Gateway - Zuul

- Spring Cloud provides an Embedded Zuul proxy
  - Acts as API Gateway



API Gateway - Zuul

- Gateway server Zuul

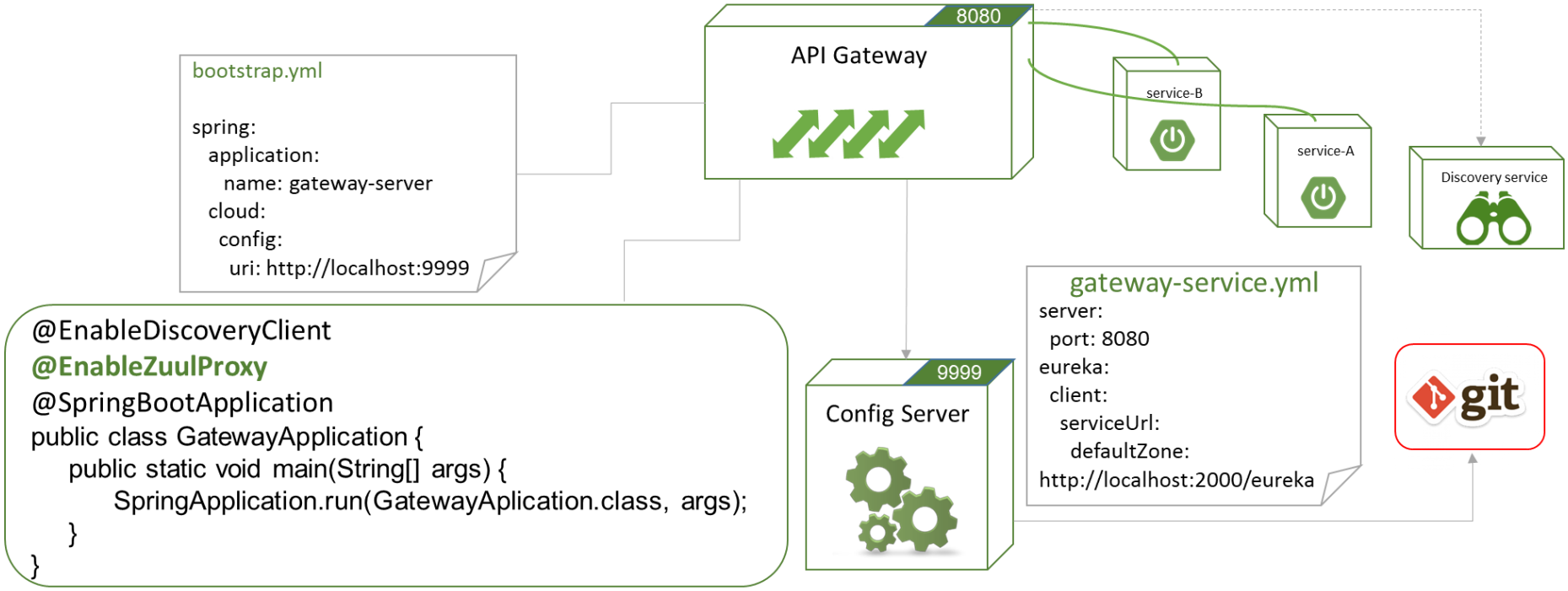
Maven dependencies:

pom.xml

```
…
<dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-zuul</artifactId>
</dependency>
<dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-eureka</artifactId>
  </dependency>
  <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-config</artifactId>
  </dependency>
 …
```

API Gateway - Zuul

- Spring Cloud provides an Embedded Zuul proxy
  - Acts as API Gateway



API Gateway - Zuul

JOHN BRYCE
Leading in IT Education
a matrix company

- When starting Gateway server it:
  - downloads configuration from Config Server (port 8080)
    - Port (8080)
    - discovery server registration info
  - Queries Discovery Server for all running services
  - Before running Gateway:
    - http://localhost:8001/..microserviceA
    - http://localhost:####/..microserviceB
  - After running Gateway – services also available on:
    - http://localhost:8080/service-a/..microserviceA
    - http://localhost:8080/service-b/..microserviceB

Note:

Zuul uses LOWER-CASE proxy names regardless actual application name as specified in bootstrap.yml e.g: 'service-A' is tracked via 'service-a'

service-A.yml
service-B.yml

API Gateway - Zuul

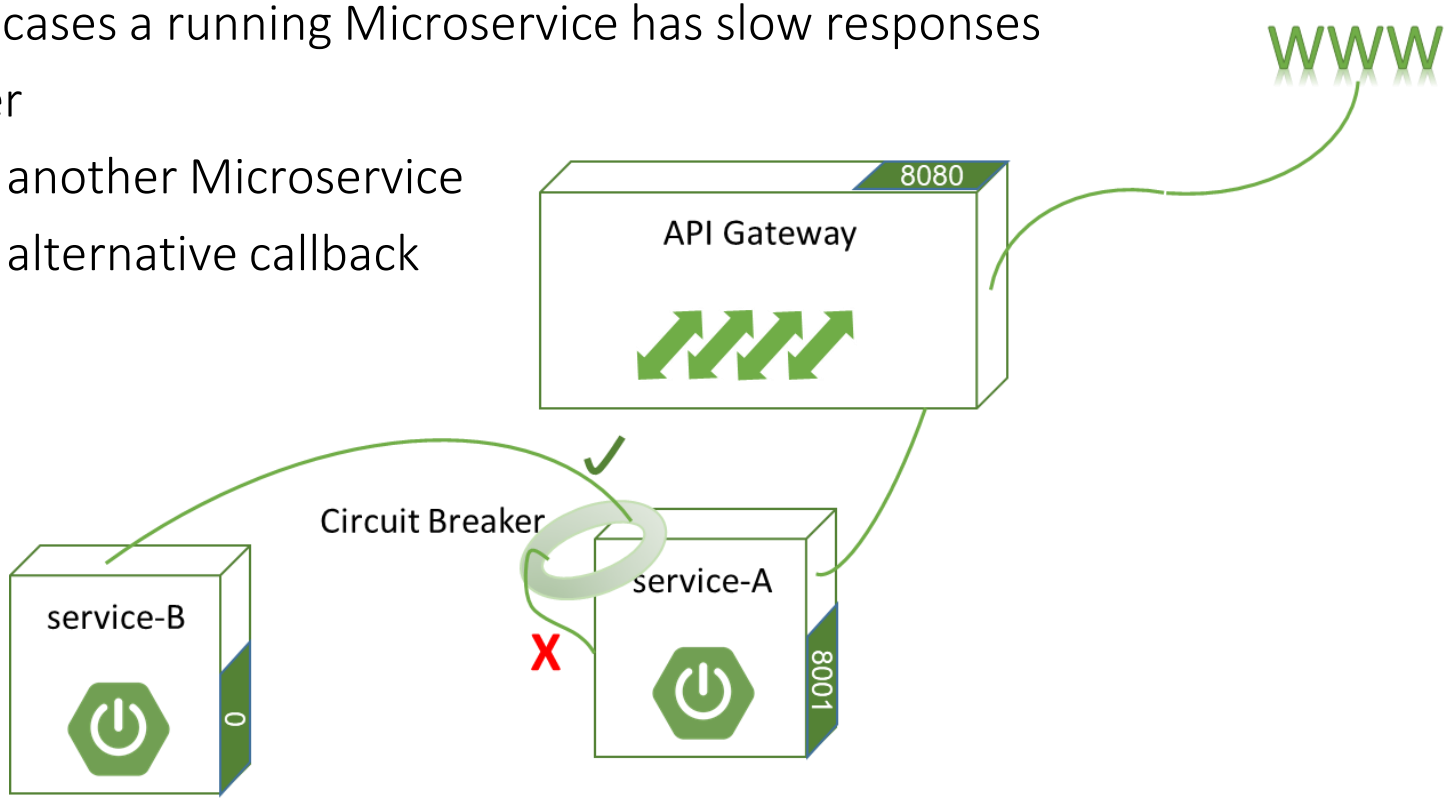
JOHN BRYCE
Leading in IT Education
a matrix company

- Zuul uses Ribbon implementation
  - Ribbon is used for load-balancing when multiple Microservice instances exists
  - Default load-balance algorithm is Round-Robin

- Currently, we can run multiple instance but only one will be discovered by Eureka
  - This is because our instances registers to Eureka with the same instance Id
  - Eureka keeps track of the last instance and ignores all others….
  - Later we will solve this issue by assigning different instance Id to Eureka

Circuit Breaker – Hystrix

- Hystrix implements Circuit breaker DP
  - Tracks requests
  - Cancel policy (5 failures in 20 sec.)
    - In cases a running Microservice has slow responses
  - Failover
    - X to another Microservice
    - X to alternative callback

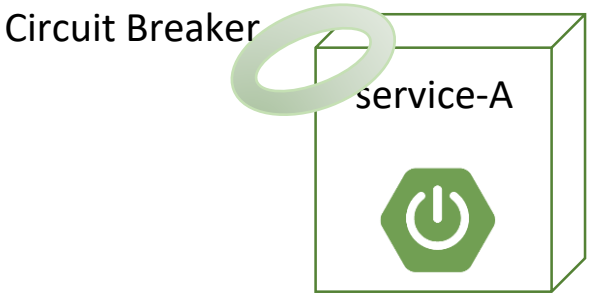


Circuit Breaker – Hystrix

- Circuit Breaker is configured on client Microservice:

```
@SpringBootApplication
@EnableDiscoveryClient
@EnableCircuitBreaker
 public class MicroserviceA }
    public static void main(String[] args) {
        SpringApplication.run(MicroserviceA.class, args);
    }

    @Configuration
    class Config {
            @LoadBalanced
            @Bean
            public RestTemplate testTemplate(){
                    return new RestTemplate();
            }
     }
}
```



Note:

@LoadBalanced puts all resource activity on a Ribbon
Circuit Breaker works on load balanced resources only (but not vice versa)

Circuit Breaker – Hystrix

- Client Microservice Hystrix circuit breaker Maven dependencies:

pom.xml

```
…
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-hystrix</artifactId>
</dependency>
 …
```

Circuit Breaker – Hystrix

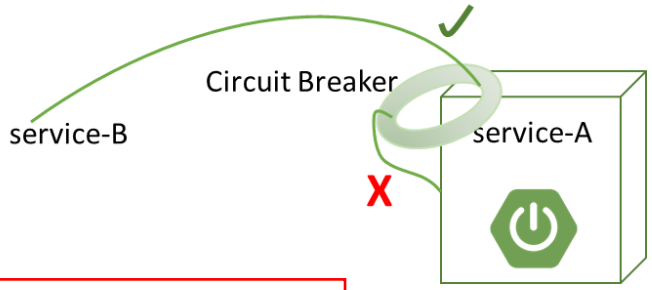
JOHN BRYCE
Leading in IT Education
a matrix company

- Using load-balanced resources:
  - @HystrixCommand – defines failover callback & meta data
  - URL is resolved by Gateway Server ("/service-B/")

```
@Autowired
  private RestTemplate restTemplate;

@HystrixCommand(fallbackMethod = "fallback", groupKey = "srvA",
                commandKey = "srvA", threadPoolKey = "srvAThread")
@GetMapping("/serviceA")
public String method() {
    String url = "http://service-B/some/url";
    return restTemplate.getForObject(url, String.class);
}

public String fallback(Throwable hystrixCommand) {
    return "Fall Back Message";
}
```



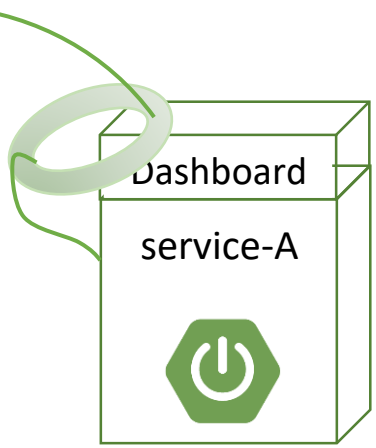
Note:

Pay attention to the fact that we use a URL without mentioning its host:port

Hystrix counts on information taken from **Config & Discovery** and maintains a valid server-list

Monitoring – Hystrix Dashboard

- Hystrix commands are gathered
- Command can be monitored via Hystrix Dashboard
- In most cases Dashboard is enabled on Gateway Server
- Denote service with @EnableHystrixDashboard to enable
- Tracks all endpoints in a Microservice
  - Success, failure, short-circuited, timed-out
  - Execution time
  - Traffic




```
@EnableDiscoveryClient
@EnableZuulProxy
@EnableHystrixDashboard
@SpringBootApplication
public class GatewayApplication {
    public static void main(String[] args) {
        SpringApplication.run(GatewayAplication.class, args);
    }
}
```

Monitoring – Hystrix Dashboard

JOHN BRYCE
Leading in IT Education
a matrix company

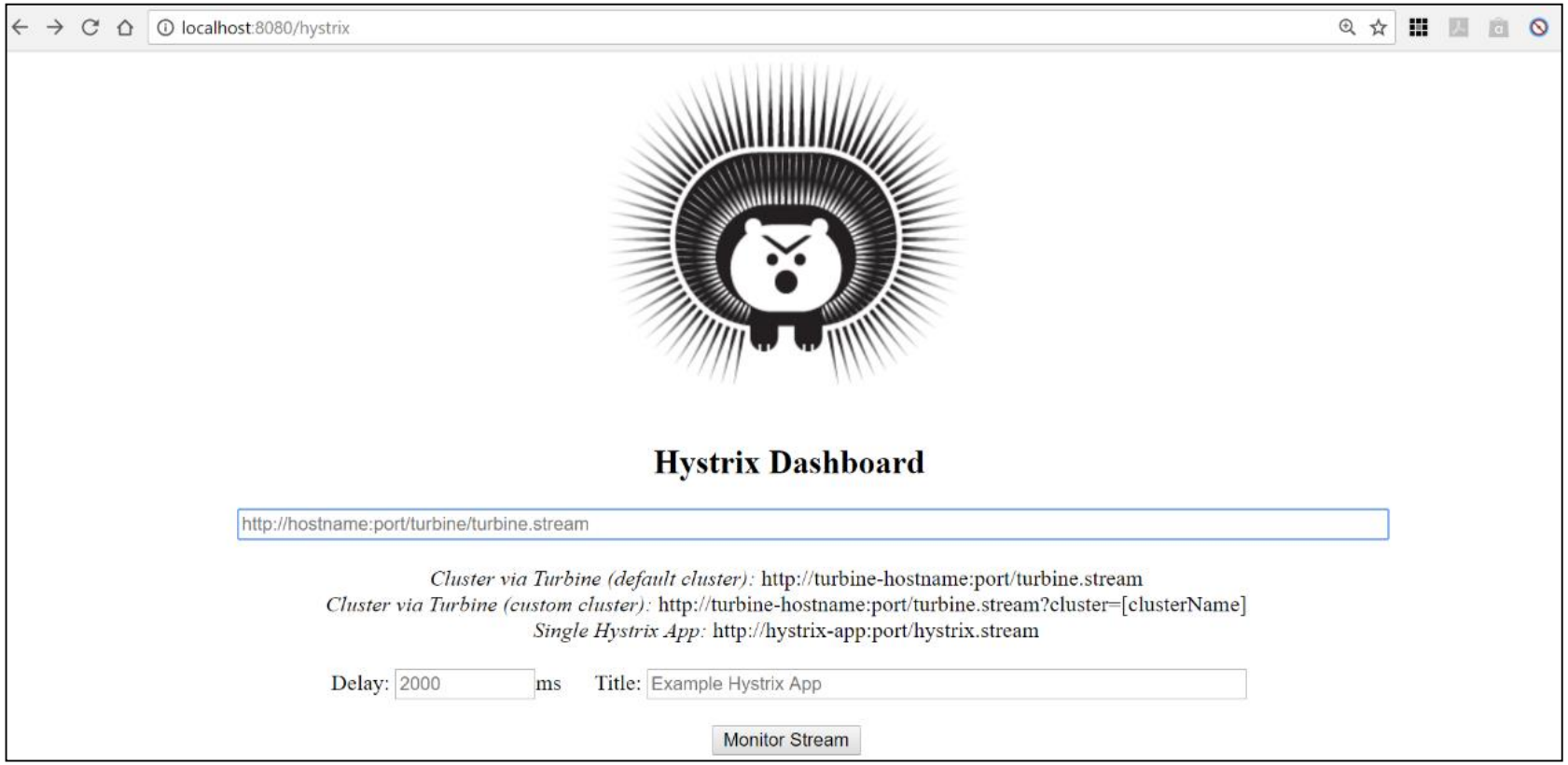
Gateway / client Microservice

Hystrix dashboard Maven dependencies:

```
pom.xml
…
<dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-hystrix-
dashboard</artifactId>
</dependency>
 …
```

Monitoring – Hystrix Dashboard

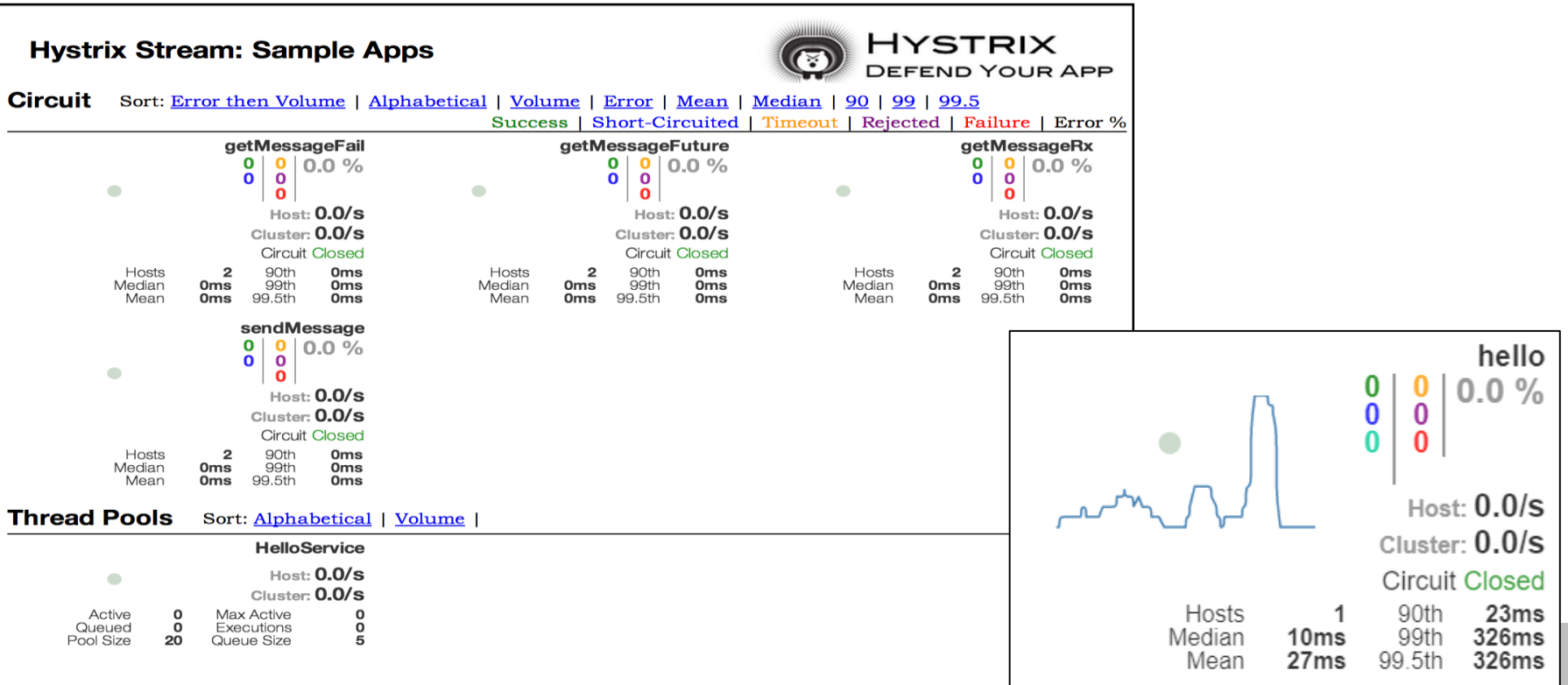
- Logging to Hystrix Dashboard on running service
  - On Gateway Server : http://localhost:8080/hystrix



Monitoring – Hystrix Dashboard

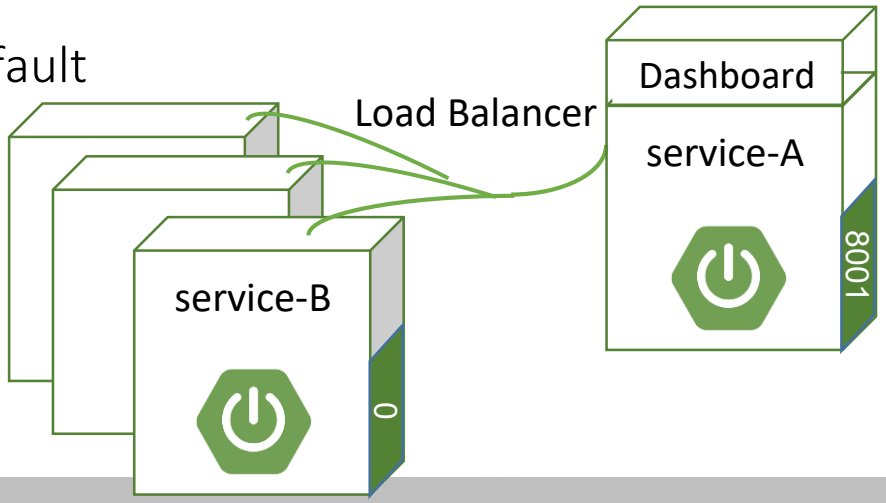
JOHN BRYCE
Leading in IT Education
a matrix company

- In order to monitor all Microservice endpoints specify its location and press 'Start Monitor'
  - Location ends with /hystrix.stream
  - http://localhost:8080/service-A/hystrix.stream (Gateway URL)



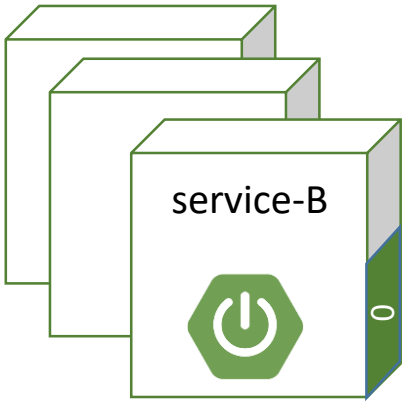
Client Side Load Balancing

- Spring Cloud Ribbon
  - Ribbon maintains load-balancing for domain-intra-communication
    - Tracks living instances & ignores failed instances
    - Maintains valid available server list
    - Can be fully configured both programmatically & via configuration files
    - Can be easily wrap any RestTemplate activity done from one Microservice to another
    - Round-robin is used by default



Client Side Load Balancing

- For load-balancing multiple service-B instances should be running
  - Random ports by assigning zero value allows multiple instances on the same host
  - Problem is that Eureka uses <application-name>:<port> as default instance ID
    - These values are the same for every service-B instance: 'service-B:0'
  - In order to provide each instance a unique name – edit service-B.yml on GIT:



service-B.yml

```
server:
   port: ….
eureka:
   instance:
      instanceId: ${spring.application.name}:${spring.application.instance_id:${random.value}}

   client:
      serviceUrl:
         defaultZone: http://localhost:2000/eureka
```

Client Side Load Balancing

JOHN BRYCE
Leading in IT Education
a matrix company

- Monitoring instances in Eureka:

Eureka

localhost:2000

spring Eureka

HOME LAST 1000 SINCE STARTUP

System Status

Environment	test
Data center	default

Current time	2018-04-18T17:25:28 +0300
Uptime	00:02
Lease expiration enabled	false
Renews threshold	5
Renews (last min)	0

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status	
MICROSERVICE-A	n/a (1)	(1)	UP (1) -	Admin-PC Microservice-A:7001
MICROSERVICE-B	n/a (1)	(1)	UP (2) -	Admin-PC Microservice-B:79b14c9dad18a6f421e75f48091691b9, Admin-PC Microservice-B:cb516a35330cddcc41debe4b317190cb

Client Side Load Balancing

JOHN BRYCE
Leading in IT Education
a matrix company

- Spring Cloud Ribbon Load Balancer & Hystrix Circuit Breaker
  - Ribbon is for load balancing
  - Hystrix is for circuit breaker

- Gateway & client load balancing uses ribbon
- Any load balanced call may use circuit breaker as well
- How this combination behaves?

Client Side Load Balancing

JOHN BRYCE
Leading in IT Education
a matrix company

- We can already enjoy load-balancing by denoting restTemplate as @LoadBalanced

- A much cleaner & rapid way is to use Feign Clients which load-balance natively
  - @FeignClient – declarative rest client generation
    - Generates RestTemplate based implementation
    - Wraps RestTemplate with client load-balancing ribbon
    - Uses URL or logical (proxy) names as base client URL

  - Feign Clients can be reused by different Microservices

  - Note: Zuul already uses ribbon. Add @HystrixCommand in addition to ribbon-load-balancer for adding Circuit-breaking capabilities

Client Side Load Balancing

- Client Microservice Ribbon Feign Starter Maven dependencies:

pom.xml

```
…
<dependency>
   <groupId>org.springframework.cloud</groupId>
   <artifactId>spring-cloud-starter-feign</artifactId>
</dependency> …
```

Client Side Load Balancing

JOHN BRYCE
Leading in IT Education
a matrix company

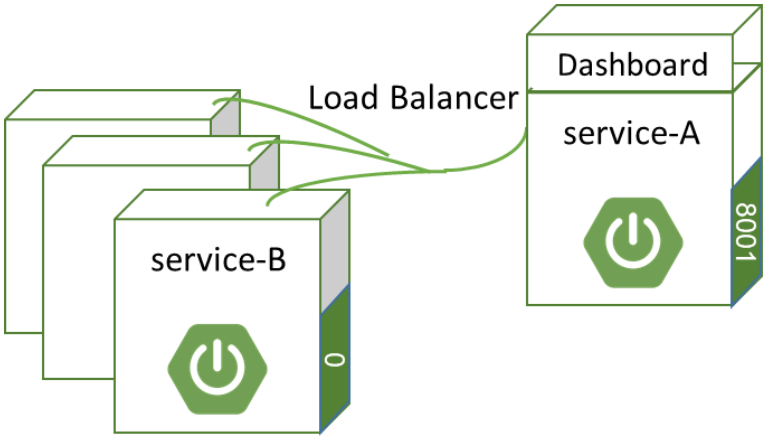
- Spring Cloud Ribbon

```
@FeignClient(name="service-B")
public interface ServiceBFeignClient {
    @GetMapping("/some/url")
    String method();
}
```

```
@RestController
public class ServiceAController
    @Autowired
     private ServiceBFeignClient client;

    @GetMapping("/call/service")
    public  String callServiceB(){
        return client.method();
    }
}
```

```
@EnableCircuitBreaker
@EnableDiscoveryClient
@SpringBootApplication
@EnableFeignClients
public class MicroserviceA{ …
```



- When calling http://service-A/call/service, microservice-B calls (http://service-B/some/uri) are load-balanced

Client Side Load Balancing

JOHN BRYCE
Leading in IT Education
a matrix company

- Ribbon can be totally configured by creating Ribbon-Configurations
- What can be set?

Bean Type	Bean Name	Class Name
IClientConfig	ribbonClientConfig	DefaultClientConfigImpl
IRule	ribbonRule	ZoneAvoidanceRule
IPing	ribbonPing	DummyPing
ServerList<Server>	ribbonServerList	ConfigurationBasedServerList
ServerListFilter<Server>	ribbonServerListFilter	ZonePreferenceServerListFilter
ILoadBalancer	ribbonLoadBalancer	ZoneAwareLoadBalancer
ServerListUpdater	ribbonServerListUpdater	PollingServerListUpdater

Client Side Load Balancing

JOHN BRYCE
Leading in IT Education
a matrix company

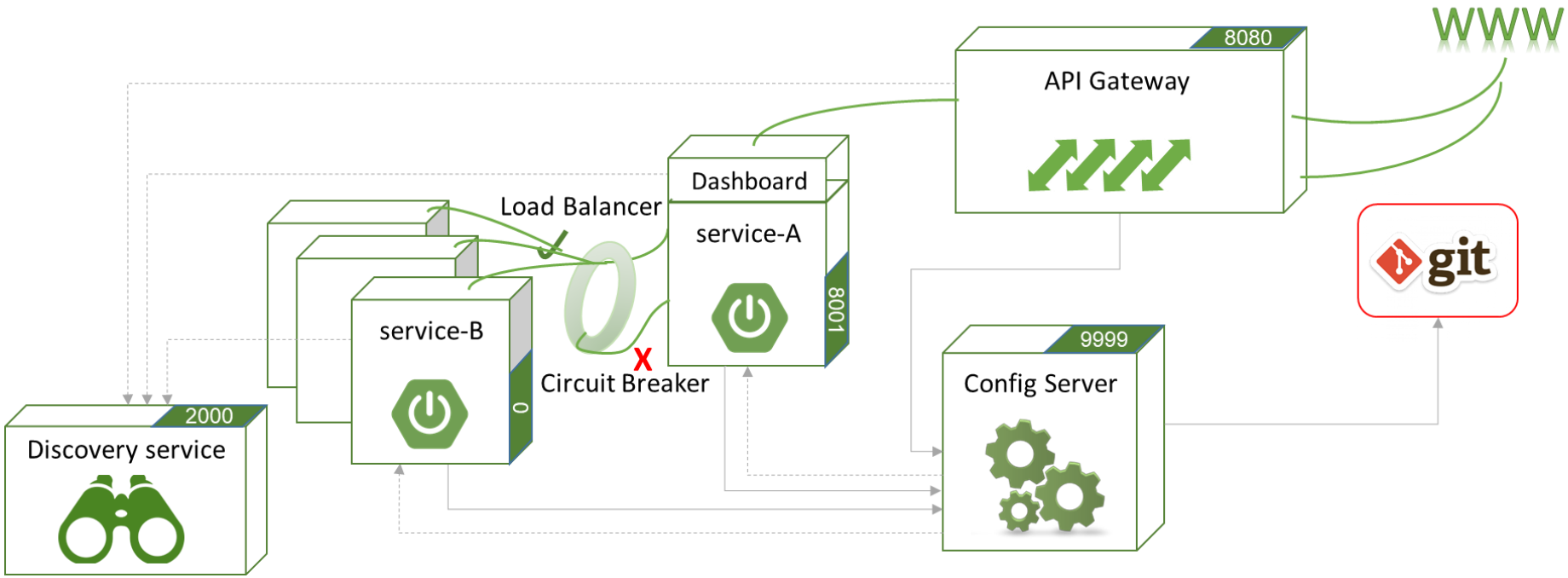
- Creating Ribbon Configuration:

```
@Configuration
protected static class FooConfiguration {

    @Bean
    public ZonePreferenceServerListFilter serverListFilter() {
            ZonePreferenceServerListFilter filter = new ZonePreferenceServerListFilter();
            filter.setZone("….");
            return filter;
    }

    @Bean
    public IPing ribbonPing() {
          return new CustomPingUrl();
    }
}
```

Spring Cloud Microservice Ecosystem



JOHN BRYCE
Leading in IT Education
a matrix company

Thank You!