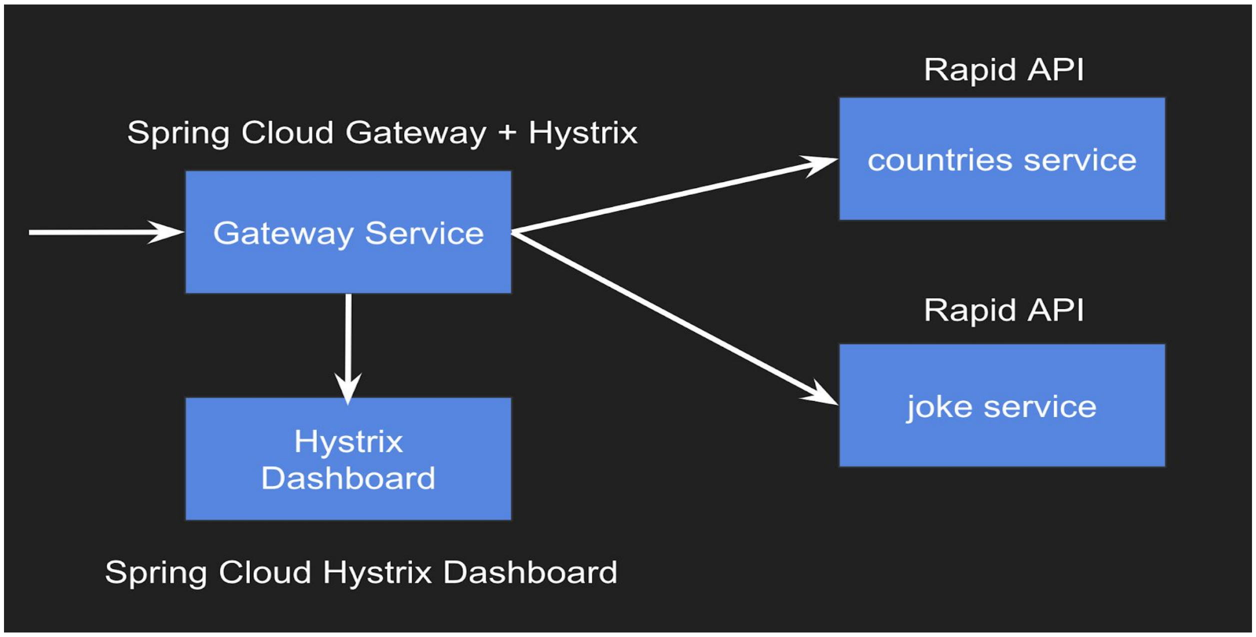
Spring Cloud Gateway Example with Hystrix 1

Architecture



URLs

- `http://localhost:8080/all` - Countries Service
- `http://localhost:8080/v1/joke` - Joke Service
- `http://localhost:8080/actuator/hystrix.stream` - Hystrix Stream endpoint
- `http://localhost:8081/hystrix` - Hystrix Dashboard

References

- Countries API
- Joke API
- Spring Guide
- Spring Cloud Gateway Documentation

Spring Cloud Gateway Example with Hystrix 2

Spring-cloud-gateway-service:

Step 1: Create a spring starter project using following dependencies
(spring-cloud-gateway-service)

```
#Actuator is mainly used to expose operational information about the running
application – #health, metrics, info, dump, env, etc. It uses HTTP#
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
#Spring Cloud Gateway makes use of the Actuator API, a well-known Spring-Boot
library that #provides several out-of-the-box services for monitoring the application.
<dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-gateway</artifactId>
</dependency>
#Hystrix is a library from Netflix. Hystrix isolates the points of access between the
#services, stops cascading failures across them and provides the fallback options.
<dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
</dependency>
```

Step2: create a application properties file(application.yml)

```
management:
  endpoints:
    web:
      exposure:
        include: hystrix.stream

hystrix:
  command:
    fallbackcmd:
      execution:
        isolation:
          thread:
            timeoutInMilliseconds: 3000
```

Step 3: Create some configurations purpose GatewayConfig.java

```
package com.chandra.springcloudgatewayservice;

import org.springframework.cloud.gateway.route.RouteLocator;
import org.springframework.cloud.gateway.route.builder.RouteLocatorBuilder;
import org.springframework.cloud.netflix.hystrix.EnableHystrix;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@EnableHystrix
@Configuration
public class GatewayConfig {

    @Bean
    public RouteLocator myRoutes(RouteLocatorBuilder builder) {
        return builder.routes()

                .route(p -> p
                        .path("/all")
                        .filters(f ->
                                f.addRequestHeader("x-rapidapi-host",
"restcountries-v1.p.rapidapi.com")
                                        .addRequestHeader("x-rapidapi-key",
"1cfbdceb89msh5ae0c25f8a27b7ap17353djsn03ed743b1d4f")
                                        .hystrix(config -> config.setName("countries-service")
                                                .setFallbackUri("forward:/countriesfallback"))
                        )
                        .uri("https://restcountries-v1.p.rapidapi.com")
                )
                .route(p -> p
                        .path("/v1/joke")
                        .filters(f ->
                                f.addRequestHeader("x-rapidapi-host", "joke3.p.rapidapi.com")
                                        .addRequestHeader("x-rapidapi-key",
"1cfbdceb89msh5ae0c25f8a27b7ap17353djsn03ed743b1d4f")
                                        .hystrix(config -> config.setName("joke-service")
                                    .setFallbackUri("forward:/jokefallback"))
                        )
                        .uri("https://joke3.p.rapidapi.com")
                )
                .build();
    }
}
```

Step 4: Create a Gateway fallback controller if you're not getting response from the endpoint this controller fallback method executed.

```
package com.chandra.springcloudgatewayservice;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import reactor.core.publisher.Mono;

@RestController
public class GatewayController {

    @RequestMapping("/countriesfallback")
    public Mono<String> countries() {
        return Mono.just("Countries API is taking too long to respond or is down. Please try
again later");
    }
    @RequestMapping("/jokefallback")
    public Mono<String> joke() {
        return Mono.just("Joke API is taking too long to respond or is down. Please try
again later");
    }
}
```

Step 5: by default configuration class created.

```
package com.chandra.springcloudgatewayservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringCloudGatewayServiceApplication {

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

}
```

Hystrix-Dashboard:

Step1 : create another Hystrix-Dashboard application using only one dependency

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

Step2: add the port number in to the application.porperties file

```
server.port=8081
```

Step 3 package com.chandra.hystrixdashboard;

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.hystrix.dashboard.EnableHystrixDashboard;
```

```
@EnableHystrixDashboard
@SpringBootApplication
public class HystrixDashboardApplication {

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

}
```

To Test Rapid Api URLs

http://localhost:8080/countries
http://localhost:8080/all
http://localhost:8080/v1/joke
http://localhost:8080/actuator/hystrixstream

http://localhost:8080/hystrix