

# Microservices

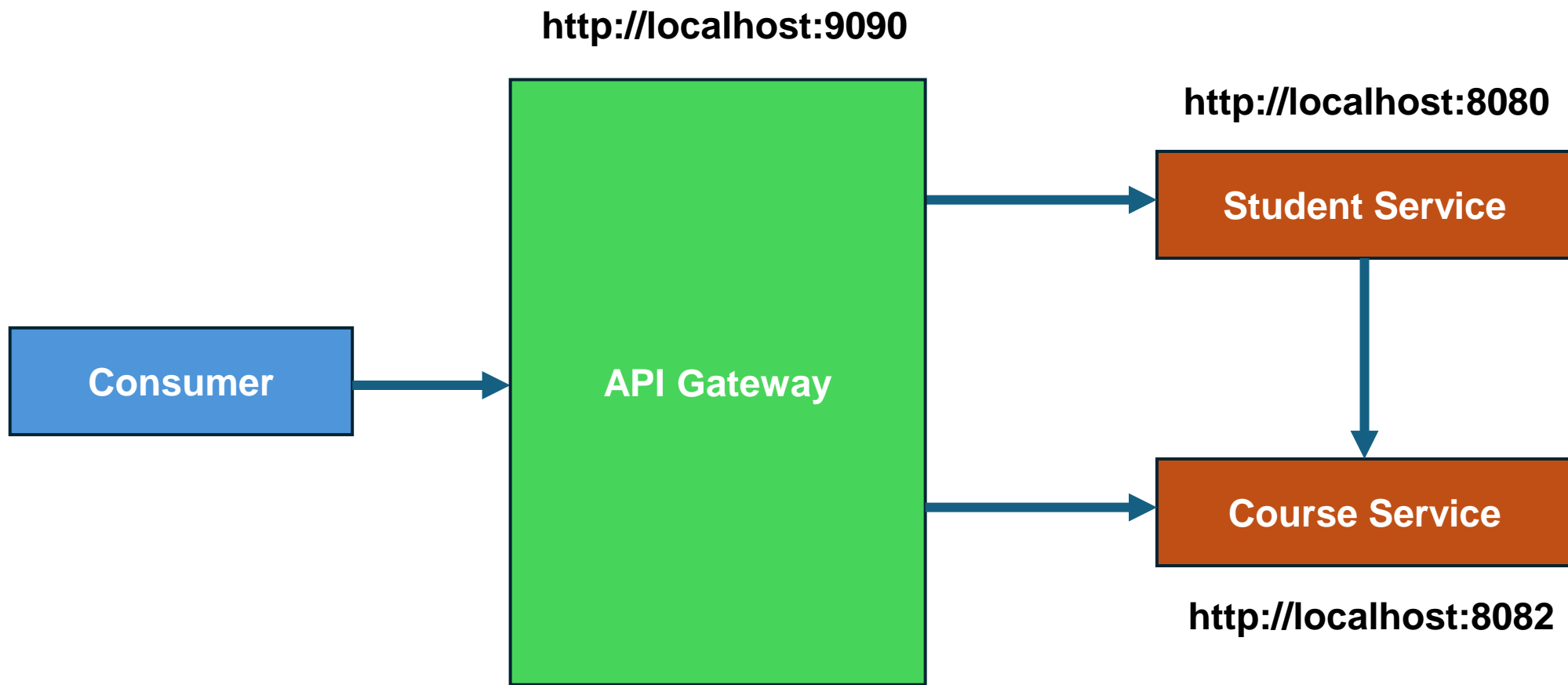
Spring Cloud Gateway

# Spring Cloud API Gateway

Alternate to Zuul

# Spring Cloud API Gateway

- Spring Cloud API Gateway is a part of the Spring Cloud ecosystem that acts as a gateway for routing requests to backend services.
- It provides a simple, effective way to route APIs, handle cross-cutting concerns like security, rate limiting, and monitoring, and work well in microservices architectures.



Service URL	<input type="text" value="https://start.spring.io"/>		
Name	<input type="text" value="api-gateway"/>		
<input checked="" type="checkbox"/> Use default location			
Location	<input type="text" value="E:\Java\MicroServices\Workspace\api-gateway"/>		<input type="button" value="Browse..."/>
Type:	<input type="text" value="Maven"/>	Packaging:	<input type="text" value="Jar"/>
Java Version:	<input type="text" value="17"/>	Language:	<input type="text" value="Java"/>
Group	<input type="text" value="com.rit"/>		
Artifact	<input type="text" value="api-gateway"/>		
Version	<input type="text" value="0.0.1-SNAPSHOT"/>		
Description	<input type="text" value="API Gateway"/>		
Package	<input type="text" value="com.rit"/>		
Working sets			
<input type="checkbox"/> Add project to working sets			<input type="button" value="New..."/>

#### Frequently Used:

- |   |  |   |
|---|--|---|
| <input checked="" type="checkbox"/> Eureka Discovery Client | <input type="checkbox"/> Eureka Server | <input checked="" type="checkbox"/> Gateway |
| <input type="checkbox"/> MySQL Driver                       | <input type="checkbox"/> OpenFeign     | <input type="checkbox"/> Spring Data JPA    |
| <input type="checkbox"/> Spring Web                         |  |   |

#### Available:

- ▶ AI
- ▶ Developer Tools
- ▶ Google Cloud
- ▶ I/O
- ▶ Messaging
- ▶ Microsoft Azure

#### Selected:

- X Eureka Discovery Client
- X Gateway

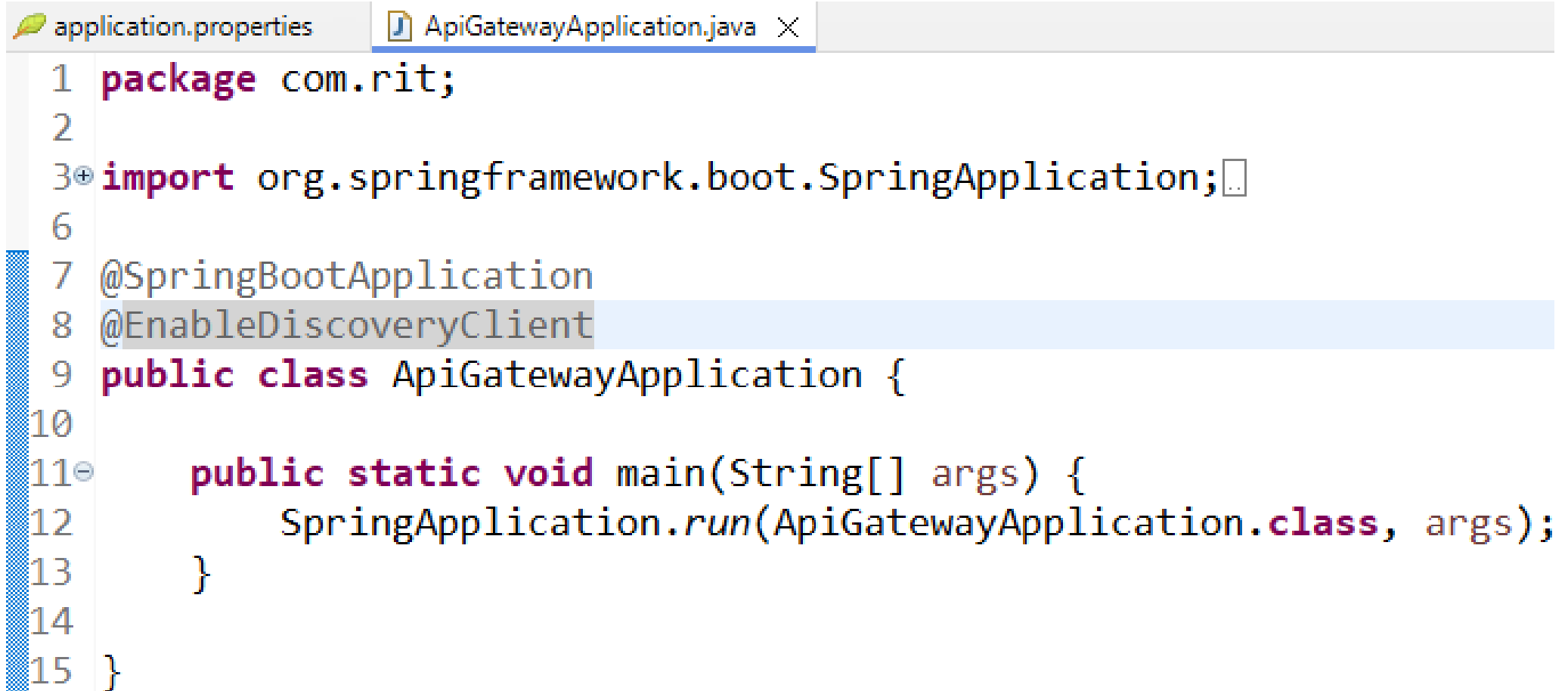
# Update the dependency to webflux

```
<properties>
  <java.version>17</java.version>
  <spring-cloud.version>2025.0.0</spring-cloud.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <!--|<artifactId>spring-cloud-starter-gateway-server-webmvc</artifactId>-->
    <artifactId>spring-cloud-starter-gateway-server-webflux</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
  </dependency>
```

# Properties File

```
*application.properties ×
1 spring.application.name=ApiGateway
2
3 #ApiGateway Port
4 server.port=9090
5
6 #Register with eureka server
7 eureka.client.service-url.defaultZone=http://localhost:8761/eureka
8
9 #Eureka server hostname
10 eureka.instance.hostname=localhost
11
12 # This allows the application to register itself with the discovery server
13 # and discover other registered services.
14 spring.cloud.discovery.enabled=true
15
16 # Enables the Discovery Locator feature in Spring Cloud Gateway (WebFlux)
17 # This allows the Gateway to automatically create routes based on services
18 spring.cloud.gateway.server.webflux.discovery.locator.enabled=true
19
20 # Converts all service IDs to lowercase when creating routes.
21 spring.cloud.gateway.server.webflux.discovery.locator.lower-case-service-id=true
22
```

# Enable Eureka Client in main class



```
application.properties  ApiGatewayApplication.java X
1 package com.rit;
2
3+ import org.springframework.boot.SpringApplication;
6
7 @SpringBootApplication
8 @EnableDiscoveryClient
9 public class ApiGatewayApplication {
10
11+     public static void main(String[] args) {
12         SpringApplication.run(ApiGatewayApplication.class, args);
13     }
14
15 }
```



# In case of error, Include this in all microservices

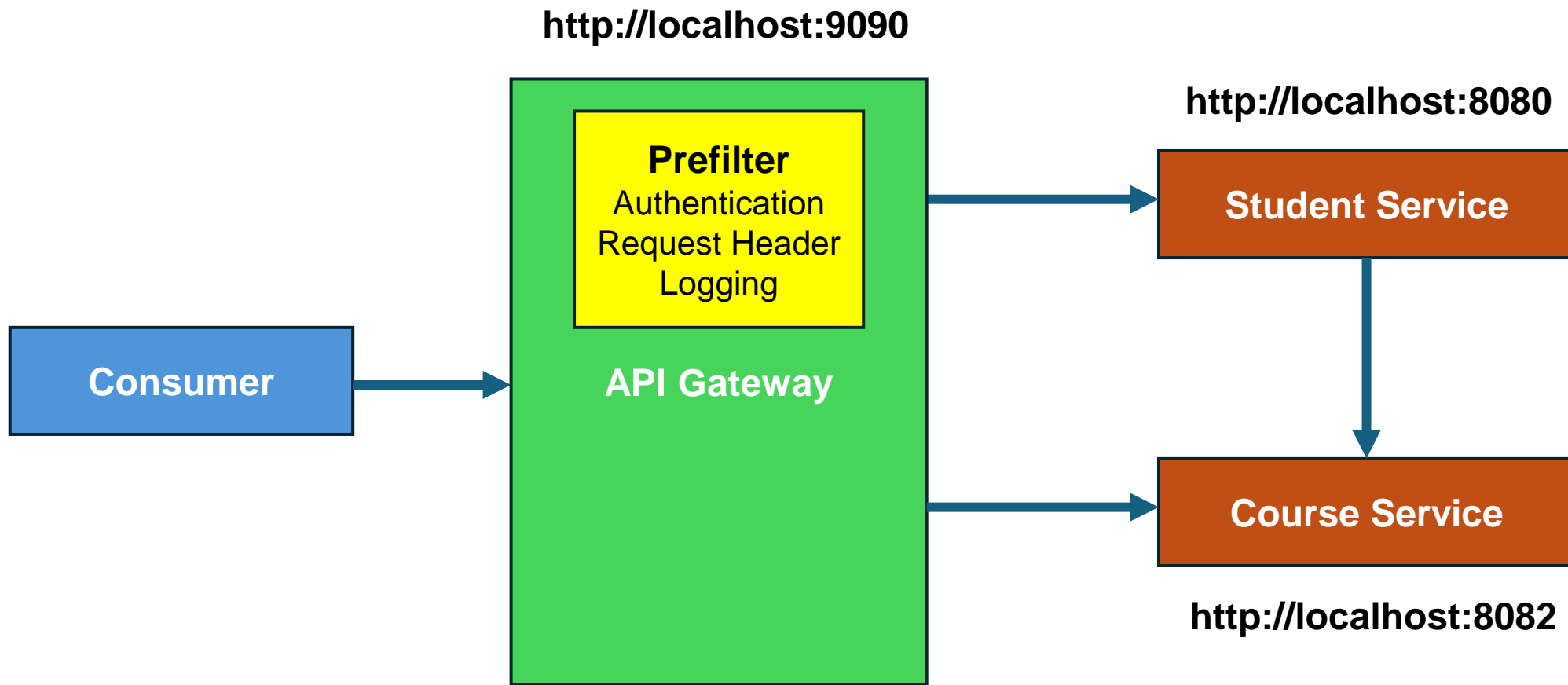
```
*application.properties ×
1 spring.application.name=course-service
2
3 server.port=8082
4
5 spring.datasource.url=jdbc:mysql://localhost:3306/mar24javafsd
6 spring.datasource.username=root
7 spring.datasource.password=root
8
9 spring.jpa.hibernate.ddl-auto=update
10 spring.jpa.show-sql=true
11
12
13 eureka.client.service-url.defaultZone=http://localhost:8761/eureka
14
15 eureka.instance.hostname=localhost
16
```

# API Gateway in Action

- Start the Eureka Server
  - API Gateway
  - Then Other service
  - Leave for a minutes for APIgateway to register all microservices
  - Then test it with postman using
  - `http://<api-gateway-port>/<service-name>/<endpointy>`
- <http://localhost:9090/student-service/api/student/getbyid/2>

# Prefilter (Cross cutting concern)

For Authentication, Request Header, Logging...



# Create a class in the API Gateway

```
*CustomFilter.java X
9  import reactor.core.publisher.Mono;
10 import org.springframework.http.server.reactive.ServerHttpRequest;
11
12 @Configuration
13 public class CustomFilter implements GlobalFilter {
14
15     Logger logger = LoggerFactory.getLogger(CustomFilter.class);
16     @Override
17     public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {
18
19         ServerHttpRequest request = exchange.getRequest();
20         logger.info("Authorization : "+request.getHeaders().getFirst("Authorization"));
21         //proceed to microservice
22         return chain.filter(exchange);
23     }
24 }
```



Send



...



1516 ms

271 B

6 hidden

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	Authorization	Marlabs Auth	
	Key	Value	



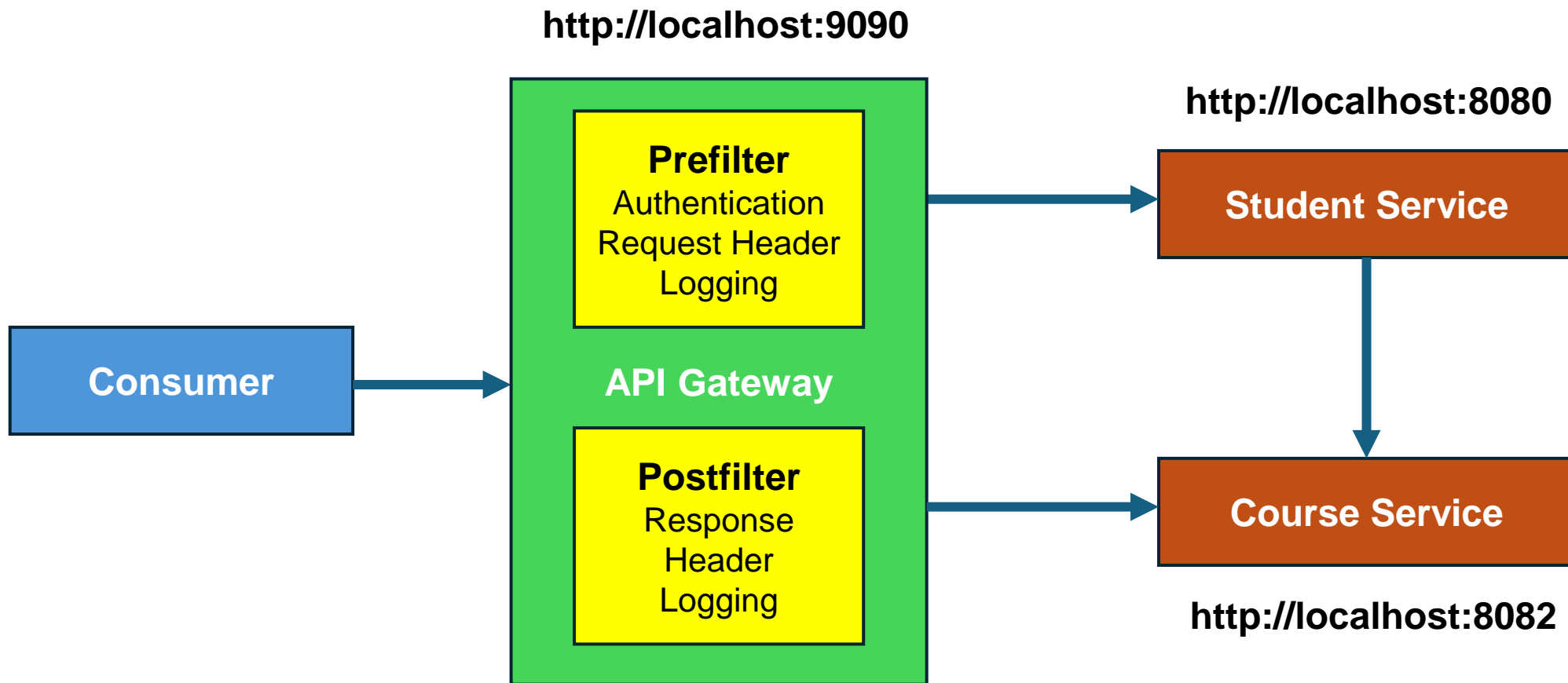
3

```
"studentId": 2,  
"firstName": "Vijay",  
"lastName": "Kumar",  
"email": "Vijay@gmail.com",  
"course": {  
  "courseId": 3,  
  "courseName": "PHP Full Stack",  
  "courseFees": 28000.0
```

```
4.24.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre
iscoveryClient
iscoveryClient
: Getting all instance registry info
: The response status is 200
: Authorization : Marlabs Auth
```

# Postfilter (Cross cutting concern)

For Response Header, Logging...





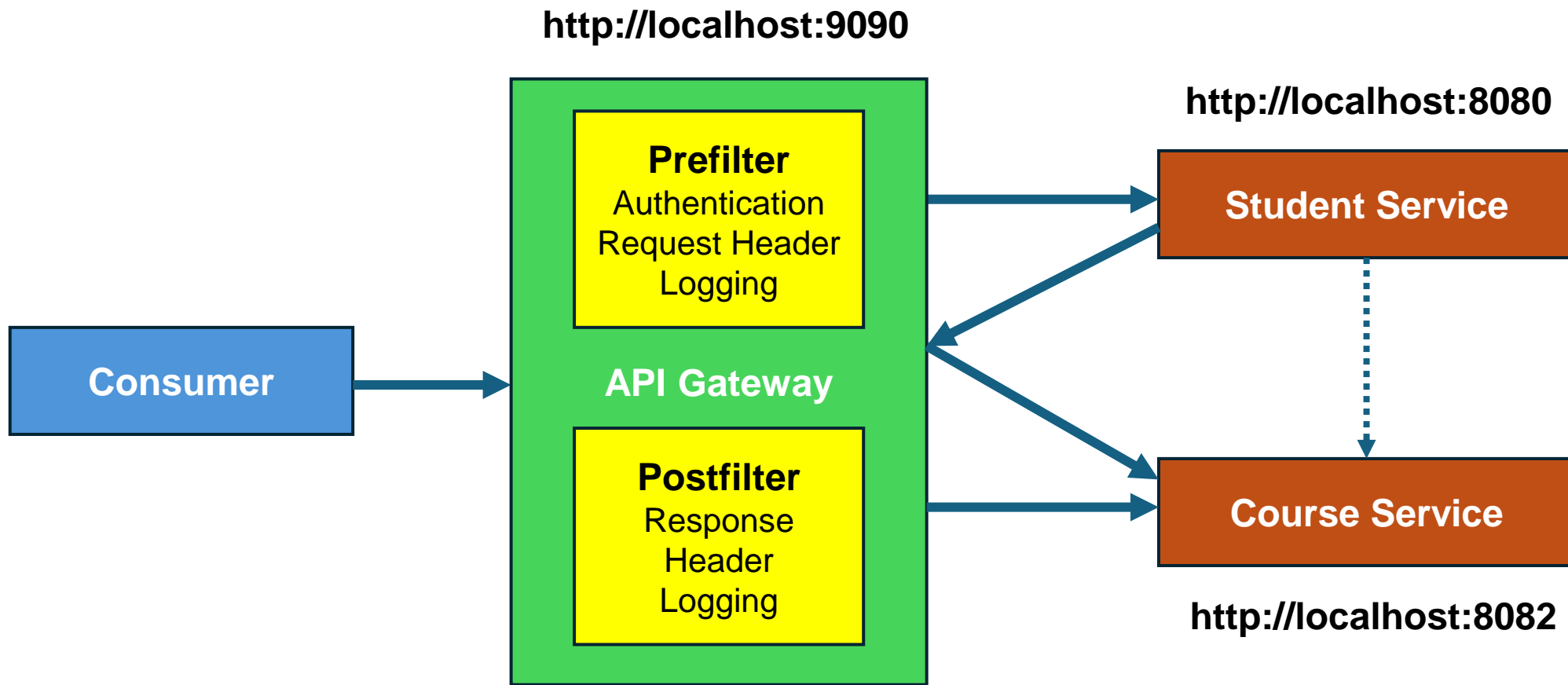
CustomFilter.java X

```
14 public class CustomFilter implements GlobalFilter {  
15  
16     Logger logger = LoggerFactory.getLogger(CustomFilter.class);  
17     @Override  
18     public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {  
19  
20         ServerHttpRequest request = exchange.getRequest();  
21         logger.info("Authorization : "+request.getHeaders().getFirst("Authorization"))  
22         //proceed to microservice  
23         return chain.filter(exchange).then(Mono.fromRunnable(()->{  
24             //while returning from microservice  
25             ServerHttpResponse response = exchange.getResponse();  
26             logger.info("Post Filter : "+response.getStatusCode());  
27         }));  
28     }  
29 }
```

jins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_21.0.3.v  
: Started ApiGatewayApplication  
: Authorization : Marlabs Auth  
: Post Filter : 200 OK

# Internal Calls through API Gateway

In Feign Client



We can create only one feign client  
with value="api-gateway"

Hence, we change the Feign Client for all microservices  
and not just Course Service

```
CourseFeignClient.java X
9
10 @FeignClient(name="api-gateway")
11 //No longer specific to Course Service
12 public interface CourseFeignClient {
13
14     @GetMapping("/course-service/api/course/getById/{id}")
15     public CourseResponse getById(@PathVariable long id);
16 }
17
18
```

# In api-gateway

```
: Authorization : Marlabs Auth  
: Authorization : null  
: Post Filter : 200 OK  
: Post Filter : 200 OK
```

# Load Balancing with API Gateway

Load Balancer not required

# We can delete / comment Load balancer

```
CourseServiceLoadBalancerConfig.java ×
1 package com.rit.feignclients;
2
3 // @LoadBalancerClient(name="course-service")
4 public class CourseServiceLoadBalancerConfig {
5     //
6     // @LoadBalanced
7     // @Bean
8     // public Feign.Builder feignBuilder(){
9     //     return Feign.builder();
10    // }
11 }
```