Microservices, Eureka Discovery & API Gateway – Step-by-Step Lab Solution

# Part A — Account Microservice

Objective: Create a Spring Boot REST microservice for Accounts.

Step-by-step execution:

1) Generate project from start.spring.io → Group: com.cognizant, Artifact: account; add Spring Web & DevTools.

2) Extract project, run: mvn clean package, then import into IDE.

3) Implement the controller to return dummy account data for a given account number.

4) Run application (default port 8080). Test in Postman.

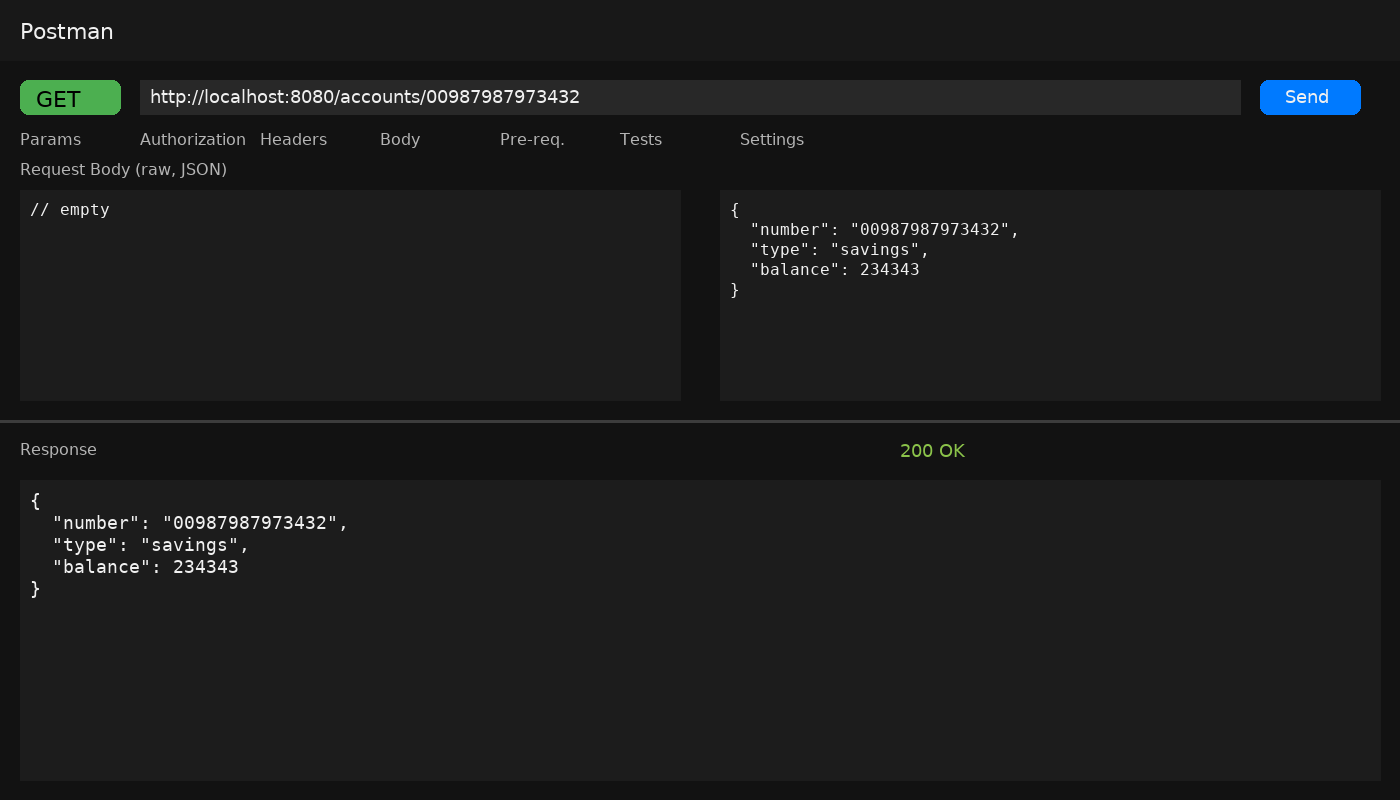
Full code:

// src/main/java/com/cognizant/account/AccountApplication.java  
@SpringBootApplication  
public class AccountApplication {  
 public static void main(String[] args) { SpringApplication.run(AccountApplication.class, args); }  
}

// src/main/java/com/cognizant/account/model/Account.java  
@Data @AllArgsConstructor @NoArgsConstructor  
public class Account {  
 private String number;  
 private String type;  
 private long balance;  
}

// src/main/java/com/cognizant/account/controller/AccountController.java  
@RestController  
@RequestMapping("/accounts")  
public class AccountController {  
 @GetMapping("/{number}")  
 public Account get(@PathVariable String number){  
 return new Account(number, "savings", 234343);  
 }  
}

Postman (request + response):



# Part B — Loan Microservice

Objective: Create another microservice for loan details and run on port 8081.

Step-by-step execution:

1) Generate project: Group: com.cognizant, Artifact: loan; add Spring Web & DevTools.

2) Set server.port=8081 in application.properties to avoid port clash with account service.

3) Implement controller to return dummy loan details.

4) Run application and test in Postman.

Full code:

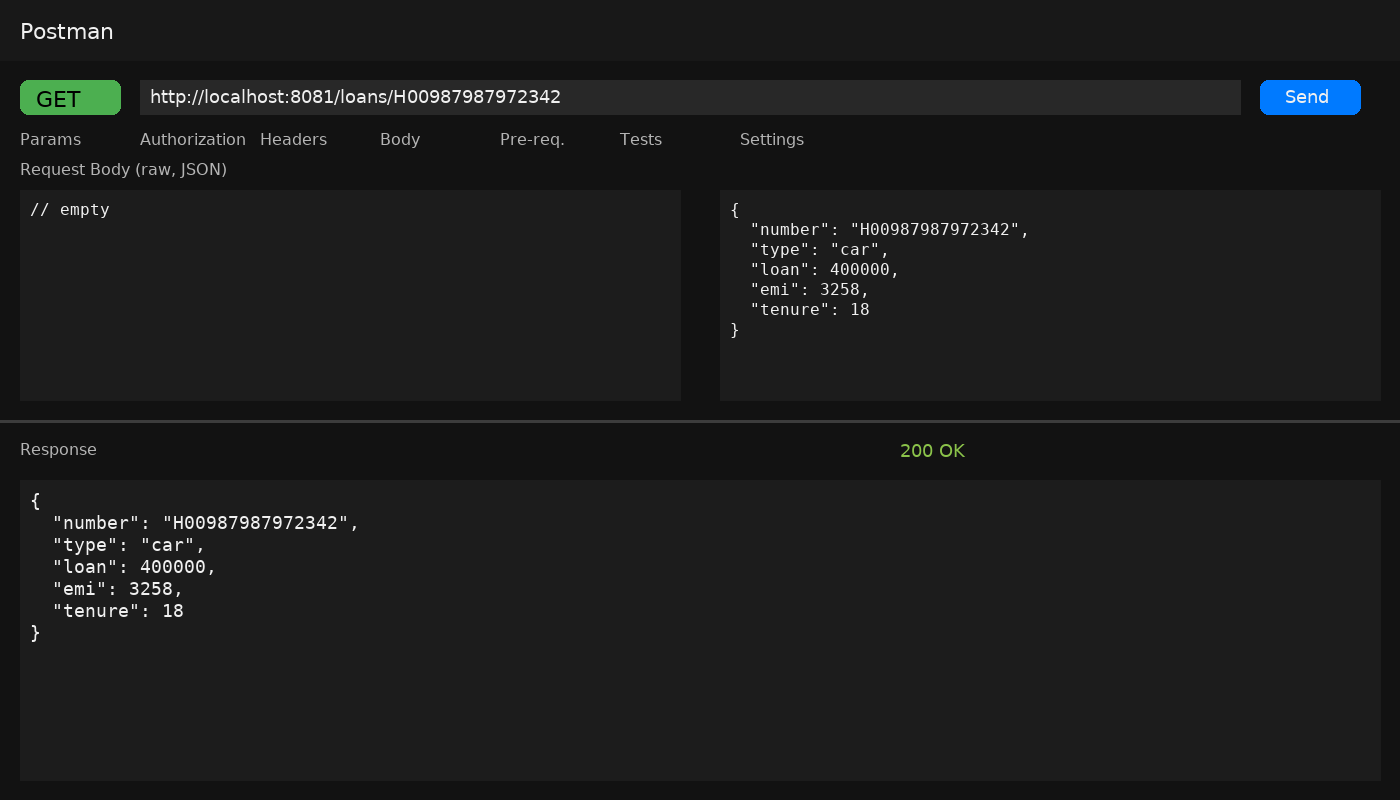
// src/main/java/com/cognizant/loan/LoanApplication.java  
@SpringBootApplication  
public class LoanApplication {  
 public static void main(String[] args) { SpringApplication.run(LoanApplication.class, args); }  
}

// src/main/java/com/cognizant/loan/model/Loan.java  
@Data @AllArgsConstructor @NoArgsConstructor  
public class Loan {  
 private String number;  
 private String type;  
 private long loan;  
 private int emi;  
 private int tenure;  
}

// src/main/java/com/cognizant/loan/controller/LoanController.java  
@RestController  
@RequestMapping("/loans")  
public class LoanController {  
 @GetMapping("/{number}")  
 public Loan get(@PathVariable String number){  
 return new Loan(number, "car", 400000, 3258, 18);  
 }  
}

# src/main/resources/application.properties  
server.port=8081

Postman (request + response):



# Part C — Eureka Discovery Server

Objective: Create Eureka server and register microservices.

Step-by-step execution:

1) Generate project: Group: com.cognizant, Artifact: eureka-discovery-server; add Spring Cloud Discovery → Eureka Server.

2) Add @EnableEurekaServer on main class.

3) Configure application.properties and run on port 8761.

4) Start server and open http://localhost:8761 to verify.

Full code & config:

<!-- pom.xml (relevant Spring Cloud setup) -->  
<dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-dependencies</artifactId>  
 <version>${spring-cloud.version}</version>  
 <type>pom</type>  
 <scope>import</scope>  
 </dependency>  
 </dependencies>  
</dependencyManagement>  
  
<dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>  
 </dependency>  
</dependencies>

// src/main/java/com/cognizant/eureka/EurekaDiscoveryServerApplication.java  
@EnableEurekaServer  
@SpringBootApplication  
public class EurekaDiscoveryServerApplication {  
 public static void main(String[] args) { SpringApplication.run(EurekaDiscoveryServerApplication.class, args); }  
}

# src/main/resources/application.properties  
server.port=8761  
eureka.client.register-with-eureka=false  
eureka.client.fetch-registry=false  
logging.level.com.netflix.eureka=OFF  
logging.level.com.netflix.discovery=OFF

# Part D — Register Account & Loan with Eureka

Objective: Make both microservices register with Eureka.

Step-by-step execution:

1) Add Eureka client dependency to each microservice (account, loan).

2) Add @EnableDiscoveryClient to each main application class.

3) Set spring.application.name and Eureka client URL in application.properties.

4) Start eureka-discovery-server first, then start account and loan; verify they appear in the registry.

Code & config:

<!-- add to account/loan pom.xml -->  
<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
</dependency>

// AccountApplication.java (or LoanApplication.java)  
@EnableDiscoveryClient  
@SpringBootApplication  
public class AccountApplication { ... }

# account/src/main/resources/application.properties  
spring.application.name=account-service  
server.port=8080  
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka

# loan/src/main/resources/application.properties  
spring.application.name=loan-service  
server.port=8081  
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka

# Part E — Spring Cloud API Gateway (with Global Log Filter)

Objective: Route requests to greet-service via API Gateway and log each inbound request.

Step-by-step execution:

1) Create greet-service microservice that returns “Hello World”.

2) Create eureka-server and register greet-service.

3) Create api-gateway with Spring Cloud Gateway and enable DiscoveryClient route locator.

4) Access greet-service via gateway: first uppercase, then configure lower-case service IDs.

5) Implement GlobalFilter to log every request.

Code — greet-service:

// greet-service/src/main/java/com/cognizant/greet/GreetApplication.java  
@EnableDiscoveryClient  
@SpringBootApplication  
public class GreetApplication {  
 public static void main(String[] args){ SpringApplication.run(GreetApplication.class, args); }  
}

// greet-service/src/main/java/com/cognizant/greet/GreetController.java  
@RestController  
public class GreetController {  
 @GetMapping("/greet")  
 public Map<String,String> greet(){ return Map.of("message","Hello World"); }  
}

# greet-service/src/main/resources/application.properties  
spring.application.name=greet-service  
server.port=8082  
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka

Code — api-gateway:

<!-- api-gateway/pom.xml -->  
<dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-gateway</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
 </dependency>  
</dependencies>  
  
<dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-dependencies</artifactId>  
 <version>${spring-cloud.version}</version>  
 <type>pom</type>  
 <scope>import</scope>  
 </dependency>  
 </dependencies>  
</dependencyManagement>

// api-gateway/src/main/java/com/cognizant/gateway/ApiGatewayApplication.java  
@EnableDiscoveryClient  
@SpringBootApplication  
public class ApiGatewayApplication {  
 public static void main(String[] args){ SpringApplication.run(ApiGatewayApplication.class, args); }  
}

# api-gateway/src/main/resources/application.properties  
spring.application.name=api-gateway  
server.port=9090  
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka  
  
# Enable discovery-based routes  
spring.cloud.gateway.discovery.locator.enabled=true  
  
# (initial behavior) ServiceId is used as registered (uppercase by Eureka UI)  
# To access: http://localhost:9090/GREET-SERVICE/greet

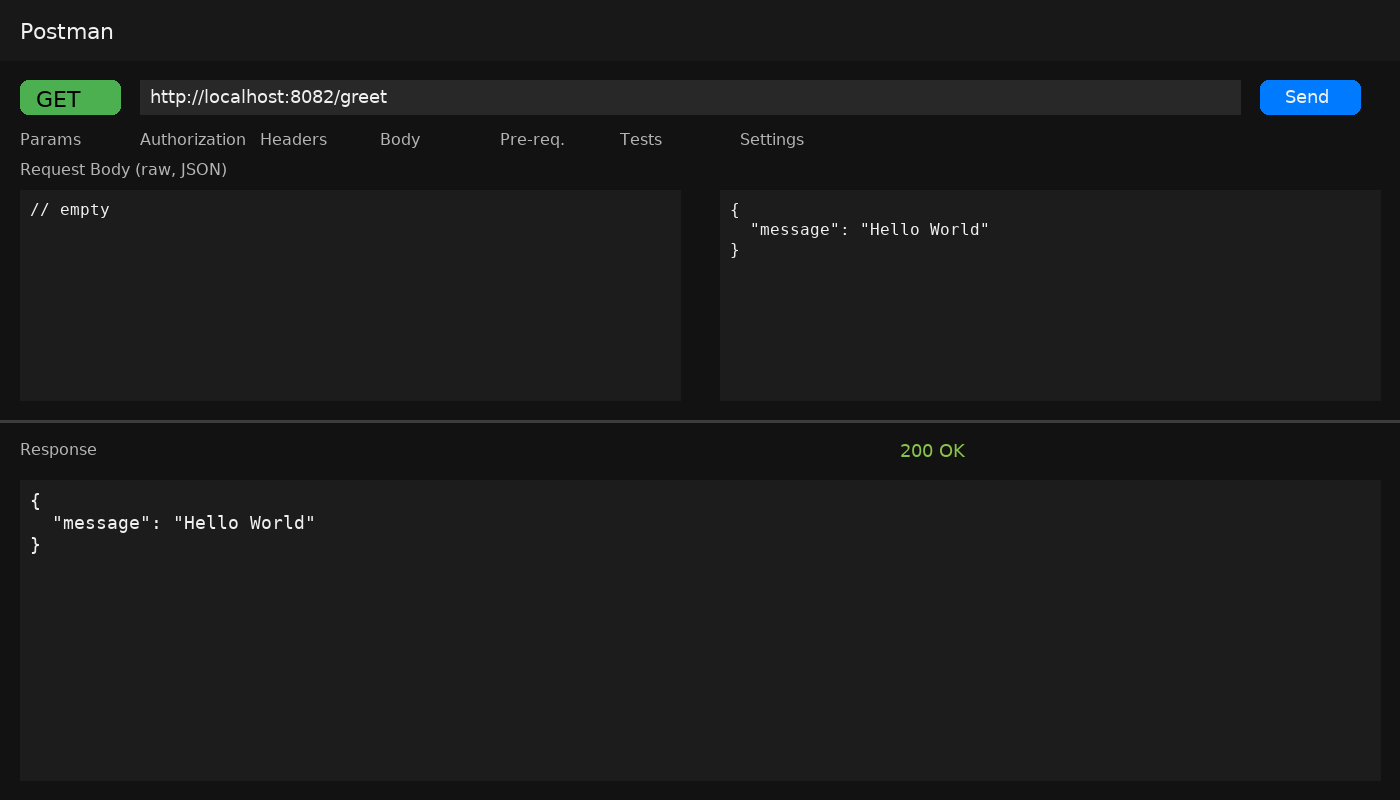
Global Log Filter:

// api-gateway/src/main/java/com/cognizant/gateway/LogFilter.java  
@Component  
public class LogFilter implements GlobalFilter, Ordered {  
 private static final Logger log = LoggerFactory.getLogger(LogFilter.class);  
  
 @Override  
 public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {  
 ServerHttpRequest req = exchange.getRequest();  
 log.info("Incoming request: {} {}", req.getMethod(), req.getURI());  
 return chain.filter(exchange).then(Mono.fromRunnable(() -> {  
 ServerHttpResponse res = exchange.getResponse();  
 log.info("Response status: {}", res.getStatusCode());  
 }));  
 }  
 @Override public int getOrder() { return -1; }  
}

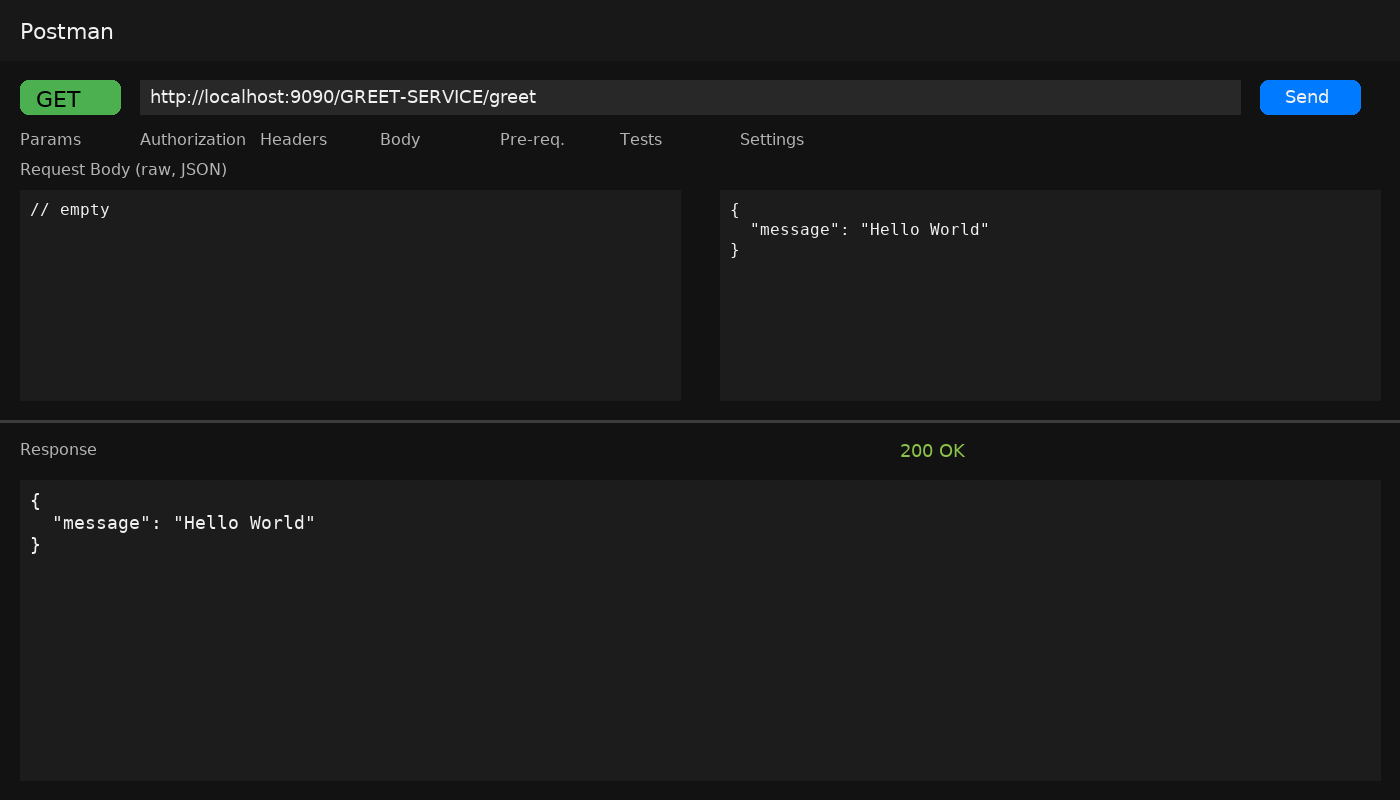
Lower-case route config (so you can access /greet-service/greet):

# api-gateway/src/main/resources/application.properties (add)  
spring.cloud.gateway.discovery.locator.lower-case-service-id=true  
# Now you can access: http://localhost:9090/greet-service/greet

Postman (greet-service direct):



Postman (via API Gateway — uppercase route):



Postman (via API Gateway — lowercase route):

