Microservices Workflow (E-Commerce Example) - Java Spring Boot

Product Service (8081)

```
--- Product Service (port 8081) - pom.xml (dependencies snippet) ---
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
    <version>3.1.4</version>
  </dependency>
</dependencies>

--- ProductApplication.java ---
package com.example.productservice;

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

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

--- Product.java (entity) ---
package com.example.productservice.model;

public class Product {
    private String id;
    private String name;
    private double price;
    private int stock;
    // constructors, getters, setters
}

--- ProductController.java ---
package com.example.productservice.controller;

import com.example.productservice.model.Product;
import org.springframework.web.bind.annotation.*;

import java.util.*;
import java.util.concurrent.ConcurrentHashMap;

@RestController
@RequestMapping("/products")
public class ProductController {
    private Map<String, Product> repo = new ConcurrentHashMap<>();

    @PostMapping
    public Product add(@RequestBody Product p) {
        repo.put(p.getId(), p);
        return p;
    }

    @GetMapping("/{id}")
    public Product get(@PathVariable String id) {
        return repo.get(id);
    }
    @GetMapping
```

```
    public Collection<Product> list() {
        return repo.values();
    }

    @PutMapping("/reduceStock")
    public Product reduceStock(@RequestParam String id, @RequestParam int qty) {
        Product p = repo.get(id);
        if (p == null) throw new RuntimeException("Not found");
        p.setStock(p.getStock() - qty);
        return p;
    }
}
```

Order Service (8082)

```
--- Order Service (port 8082) - pom.xml (dependencies snippet) ---
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-webflux</artifactId>
  </dependency>
</dependencies>

--- OrderApplication.java ---
package com.example.orderservice;

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

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

--- Order.java (entity) ---
package com.example.orderservice.model;

public class Order {
    private String orderId;
    private String productId;
    private int quantity;
    private String status; // PENDING_PAYMENT, CONFIRMED, FAILED
    private double totalAmount;
    // constructors, getters, setters
}

--- OrderController.java (simplified synchronous flow) ---
package com.example.orderservice.controller;

import com.example.orderservice.model.Order;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.client.RestTemplate;

import java.util.UUID;
import java.util.concurrent.ConcurrentHashMap;
import java.util.Map;

@RestController
@RequestMapping("/orders")
public class OrderController {
    private Map<String, Order> repo = new ConcurrentHashMap<>();
    private RestTemplate rest = new RestTemplate();

    @PostMapping
    public ResponseEntity<?> placeOrder(@RequestBody Map<String,Object> req) {
        String productId = (String) req.get("productId");
        int qty = (int) req.get("quantity");
        // 1. Check product
```

```
        Map product = rest.getForObject("http://localhost:8081/products/" + productId, Map.class);
        if (product == null) return ResponseEntity.badRequest().body("Product not found");
        int stock = (int) product.get("stock");
        double price = ((Number)product.get("price")).doubleValue();
        if (stock < qty) return ResponseEntity.badRequest().body("Insufficient stock");

        // 2. Create pending order
        String orderId = UUID.randomUUID().toString();
        Order order = new Order();
        order.setOrderId(orderId);
        order.setProductId(productId);
        order.setQuantity(qty);
        order.setStatus("PENDING_PAYMENT");
        order.setTotalAmount(price * qty);
        repo.put(orderId, order);

        // 3. Call payment service
        Map<String,Object> paymentReq = Map.of("orderId", orderId, "amount", order.getTotalAmount());
        Map paymentResp = rest.postForObject("http://localhost:8083/payments", paymentReq, Map.class);

        if ("SUCCESS".equals(paymentResp.get("status"))) {
            order.setStatus("CONFIRMED");
            // 4. Reduce stock
            rest.put("http://localhost:8081/products/reduceStock?id=" + productId + "&qty=" + qty, null);
        } else {
            order.setStatus("FAILED");
        }

        return ResponseEntity.ok(order);
    }
}
```

Payment Service (8083)

```
--- Payment Service (port 8083) - pom.xml (dependencies snippet) ---
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
  </dependency>
</dependencies>

--- PaymentApplication.java ---
package com.example.paymentservice;

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

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

--- PaymentController.java ---
package com.example.paymentservice.controller;

import org.springframework.web.bind.annotation.*;
import java.util.Map;

@RestController
@RequestMapping("/payments")
public class PaymentController {

    @PostMapping
    public Map<String,Object> pay(@RequestBody Map<String,Object> req) {
        // simulate payment processing
        String orderId = (String) req.get("orderId");
        Number amount = (Number) req.get("amount");
        // For demo, always succeed
        return Map.of("orderId", orderId, "status", "SUCCESS");
    }
}
```

Eureka Server (8761)

```
--- Eureka Server (port 8761) - pom.xml (dependencies snippet) ---
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
</dependencies>

--- EurekaApplication.java ---
package com.example.eureka;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

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