71. What are the common ways services communicate in microservices?

1. Synchronous – REST APIs, gRPC
2. Asynchronous – Messaging (Kafka, RabbitMQ)

72. What is synchronous communication?

Services interact in real-time, waiting for a response. Example: REST API call from Order Service to Inventory Service.

73. What is asynchronous communication?

Caller sends a message and doesn't wait for a response. Improves performance and resilience. Tools: Kafka, RabbitMQ.

74. When should you use asynchronous communication?

Use when you don't need an immediate response, want decoupling, or use event-driven architecture.

75. What are the drawbacks of synchronous communication?

Tight coupling, higher latency, cascading failure risks, and poor resilience in network failures.

76. What is Feign in Spring Cloud?

A declarative REST client. Example:
@FeignClient(name = "inventory-service")
public interface InventoryClient {
@GetMapping("/inventory/check/{id}")
InventoryResponse check(@PathVariable String id);
}

77. What are the benefits of using Feign Client?

- Simplifies REST calls
- Integrates with Eureka
- Supports fallback
- Declarative and clean

78. What is RestTemplate?

A blocking HTTP client in Spring. Example:
RestTemplate rt = new RestTemplate();
ResponseEntity r = rt.getForEntity("http://user-service/users/1", User.class);

79. What is WebClient?

A non-blocking, reactive HTTP client introduced in Spring 5. Example:
WebClient wc = WebClient.create();
wc.get().uri("/products").retrieve().bodyToMono(Product.class);

80. Feign vs RestTemplate vs WebClient

Feign: Declarative, Eureka integration
RestTemplate: Blocking
WebClient: Reactive, non-blocking

81. What is messaging in microservices?

Services communicate using messages through brokers like Kafka or RabbitMQ. Enables asynchronous interaction.

82. What are some popular message brokers?

- Kafka
- RabbitMQ
- ActiveMQ
- Amazon SQS

83. What is the Producer and Consumer model?

Producer sends messages to a broker; Consumer reads them. Services can be both.

84. What are the benefits of message queues in microservices?

Loose coupling, retry handling, decoupled scaling, better async performance.

85. What is message durability?

Messages are persisted to avoid loss. Kafka persists and replicates messages.

86. What is idempotency in messaging?

Guarantees that processing the same message multiple times has the same effect. Helps with retries.

87. What is dead-letter queue (DLQ)?

Special queue for failed messages. Helps with troubleshooting and avoiding message loss.

88. What is the Outbox Pattern?

Ensures reliable message publishing. Events are written to DB, then published to broker by a separate job.

89. What is correlation ID and why is it important?

A unique ID used for tracing requests across services. Enables end-to-end logging and debugging.

90. How do you implement request tracing across services?

Use correlation IDs, Spring Sleuth, and Zipkin or Jaeger for distributed tracing visualization.