Spring Cloud - Practice Sheet

10 Q&A on service discovery, gateway, circuit breaking, and tracing

Q1. What problems does Spring Cloud solve in a microservices architecture?

A: Spring Cloud provides turnkey solutions for service discovery (Eureka), distributed configuration (Config Server), API gateway (Gateway), circuit breaking (Resilience4j), and distributed tracing (Micrometer + Zipkin) the core operational concerns of microservices.

Q2. How does Eureka service discovery work in Spring Cloud?

A: Services annotated @EnableEurekaClient register with the Eureka server on startup (heartbeat every 30 s). Clients use DiscoveryClient or a load-balanced RestTemplate/WebClient to resolve service names to running instances by querying Eureka.

Q3. What is Spring Cloud Gateway and how do you define a route?

A: Gateway is a reactive API gateway built on Spring WebFlux. Routes are defined by predicates (Path=/api/**) and filters (AddRequestHeader, CircuitBreaker, RateLimiter). Requests matching a predicate are forwarded to the configured URI.

Q4. How does Spring Cloud Config Server manage distributed configuration?

A: Config Server serves property files stored in a Git repo (or file system). Client applications fetch their config from Config Server at startup using spring.application.name and active profiles. Refresh without restart is possible via /actuator/refresh + @RefreshScope.

Q5. How does Resilience4j circuit breaking work in Spring Cloud?

A: A circuit breaker wraps a service call and counts failures. When failures exceed a threshold the circuit opens, and calls immediately return a fallback without hitting the downstream service. After a wait duration the circuit enters HALF_OPEN to test recovery.

Q6. What is distributed tracing in Spring Cloud and how does Micrometer help?

A: Micrometer Tracing injects a trace ID and span ID into every request, propagating them via HTTP headers (B3 or W3C). Spring Cloud integrates with Zipkin or Grafana Tempo. All spans for one user request share the same trace ID for correlation.

Q7. How does Spring Cloud LoadBalancer perform client-side load balancing?

A: A @LoadBalanced RestTemplate or WebClient intercepts requests, resolves the service name via DiscoveryClient (Eureka, Consul), and selects an instance using a RoundRobin strategy. It replaces the deprecated Netflix Ribbon library.

Q8. What is Spring Cloud Stream and how does it abstract message brokers?

A: Stream defines Supplier (publisher), Function (processor), and Consumer (subscriber) beans that are automatically bound to message broker topics. A binder (Kafka or RabbitMQ) translates between the Spring abstraction and the broker's API.

Q9. What is the Saga pattern and how does it apply to Spring Cloud microservices?

A: A Saga replaces a distributed ACID transaction with a sequence of local transactions, each publishing an event. If a step fails, compensating transactions undo prior steps. Orchestration (a central saga coordinator) or choreography (event-driven) are the two styles.

Q10. How do you propagate OAuth2 tokens between microservices in Spring Cloud?

A: Configure each microservice as a Resource Server. Use Spring Security's token relay filter or a WebClient filter (ServerBearerExchangeFilterFunction) to forward the incoming Bearer token from upstream to downstream service calls.