Jawaban

2. External Service Reliability

a) Kondisi di mana retry tidak boleh dilakukan:
Retry tidak dilakukan pada request non-idempotent tanpa idempotency key, error client-side (4xx selain 429), kegagalan permanen (business validation), atau ketika retry berpotensi memperparah overload dan berdampak ke sistem lain.

b) Perbedaan retry dan circuit breaker:
Retry berfungsi untuk menangani transient failure dengan mencoba ulang request.
Circuit breaker berfungsi melindungi sistem dengan menghentikan request sementara saat failure rate tinggi, mencegah cascading failure.

c) Risiko retry tanpa jitter/backoff:
Tanpa jitter dan backoff, retry dapat menyebabkan thundering herd, meningkatkan beban secara bersamaan, memperburuk latency, dan berujung pada self-induced denial of service.

3. Observability & Debugging

a) Informasi minimum dalam log:
Setiap log wajib memiliki timestamp, log level, service name, traceId/correlationId, requestId, serta identifier bisnis (userId atau entityId).

b) Menelusuri request end-to-end:
Menggunakan distributed tracing dengan traceId yang dipropagasikan antar service melalui HTTP header, dikombinasikan dengan centralized logging dan metrics.

c) Dampak traceId tidak konsisten:
Tanpa traceId konsisten, alur request tidak dapat direkonstruksi, root cause analysis menjadi sulit, dan waktu debugging meningkat signifikan.

4. Event-Driven Architecture

a) Menjamin consumer idempotent:
Consumer harus mampu memproses event lebih dari sekali dengan aman menggunakan eventId unik, deduplication, dan operasi upsert.

b) Ordering dijamin oleh siapa:
Broker hanya menjamin ordering dalam scope terbatas (partition/queue). Consumer bertanggung jawab menjaga urutan pemrosesan jika ordering penting.

c) Dampak ordering tidak terjaga:
Ketidakterjagaan ordering dapat menyebabkan state tidak konsisten, data lama menimpa data baru, dan bug sulit direproduksi.

5. Caching Strategy

a) Kapan cache di-invalidate:
Cache di-invalidate saat data sumber berubah, setelah write/update/delete, atau berdasarkan TTL untuk mencegah stale data.

b) Cache sebagai source of truth:
Cache tidak boleh menjadi source of truth karena sifatnya volatile dan tidak konsisten.

c) Pencegahan cache stampede:
Menggunakan locking, request coalescing, staggered TTL, dan circuit breaker ke database.

6. Internal Service Security

a) Apakah API key cukup:
API key saja tidak cukup karena lemah secara identity dan sulit direvoke.

b) Mencegah service palsu:
Menggunakan mTLS, service mesh, network isolation, dan OAuth2 client credentials.

c) Risiko JWT tanpa expiry pendek:
JWT berdurasi panjang meningkatkan risiko token leakage, replay attack, dan menyulitkan revocation.