NAMA : HANIF AL ZIKRI

NIM : 2301082009

MATKUL : Microservices

**1. Server-side Discovery**

1. **Definisi**
   1. Mekanisme di mana **load balancer** atau **router** bertindak sebagai perantara untuk mengarahkan permintaan klien ke instance layanan (service) yang tersedia.
   2. Klien tidak perlu mengetahui lokasi pasti service; cukup mengirim permintaan ke endpoint yang telah ditentukan.
2. **Cara Kerja**
   1. Klien mengirim permintaan ke **load balancer** (contoh: API Gateway).
   2. Load balancer menanyakan **Service Registry** untuk mendapatkan daftar instance layanan yang tersedia.
   3. Load balancer meneruskan permintaan ke instance yang sesuai (berdasarkan algoritma seperti round-robin atau least connections).
3. **Contoh Tools**
   1. AWS Elastic Load Balancer (ELB), NGINX, Kubernetes Service.

**2. Client-side Discovery**

1. **Definisi**
   1. Klien bertanggung jawab untuk menemukan lokasi service dengan langsung berkomunikasi dengan **Service Registry**.
   2. Klien memilih instance service yang tersedia dan mengirim permintaan langsung ke instance tersebut.
2. **Cara Kerja**
   1. Klien meminta daftar instance service dari **Service Registry**.
   2. Klien menggunakan algoritma (seperti load balancing sederhana) untuk memilih instance.
   3. Klien mengirim permintaan langsung ke instance yang dipilih.
3. **Contoh Tools**
   1. Netflix Eureka, Consul, Zookeeper.

**3. Kelebihan dan Kekurangan**

**Server-side Discovery**

1. **Kelebihan**:
   1. Klien sederhana (tidak perlu mengimplementasi logika discovery).
   2. Sentralisasi manajemen traffic (load balancer mengatur distribusi).
   3. Keamanan lebih baik (klien tidak mengakses Service Registry langsung).
2. **Kekurangan**:
   1. Load balancer menjadi **single point of failure**.
   2. Latensi tambahan karena permintaan melewati perantara.
   3. Konfigurasi load balancer yang kompleks.

**Client-side Discovery**

1. **Kelebihan**:
   1. Latensi lebih rendah (komunikasi langsung ke service).
   2. Tidak ada bottleneck di load balancer.
   3. Fleksibilitas tinggi (klien bisa memilih algoritma discovery).
2. **Kekurangan**:
   1. Klien lebih kompleks (harus mengimplementasi logika discovery).
   2. Tight coupling dengan Service Registry.
   3. Potensi masalah konsistensi (cache daftar service mungkin kadaluarsa).

**4. Service Registry**

1. **Definisi**:
   1. Database terdistribusi yang menyimpan lokasi dan metadata semua instance service dalam sistem mikroservis.
   2. Bertindak sebagai **sumber kebenaran** untuk komponen yang membutuhkan service discovery.
2. **Fungsi Utama**:
   1. **Pendaftaran (Registration)**: Service instance mendaftarkan diri ke registry saat startup.
   2. **Deregistrasi**: Menghapus instance yang tidak aktif (misal: saat crash).
   3. **Health Check**: Memantau kesehatan service (contoh: heartbeat mechanism).
   4. **Pencarian (Lookup)**: Memberikan daftar instance service yang tersedia saat diminta.
3. **Contoh Implementasi**:
   1. **Netflix Eureka**: Open-source, dirancang untuk cloud.
   2. **Consul**: Mendukung multi-datacenter dan fitur tambahan (KV store, ACL).
   3. **Zookeeper**: Apache project, digunakan oleh Kafka dan Hadoop.
4. **Pentingnya**:
   1. Memungkinkan **dynamic scaling** (service bisa ditambah/dihapus tanpa konfigurasi manual).
   2. Mendukung **resilience** (failover otomatis jika instance mati).