**Tugas Micro Kontroller**

Nama: Farhan Delonart A

NIM : 2301082006

**1. Server-side Discovery**

**Definisi**:  
Server-side Discovery adalah pola dalam arsitektur mikroservis di mana *client* mengirim permintaan ke *load balancer* atau komponen perantara (seperti API Gateway), yang bertanggung jawab untuk menemukan lokasi layanan (service) yang tersedia di *Service Registry* dan meneruskan permintaan ke instance layanan yang sesuai.

**Cara Kerja**:

1. Client mengirim permintaan ke endpoint statis (misalnya, API Gateway).
2. API Gateway menanyakan *Service Registry* (seperti Eureka atau Consul) untuk mendapatkan daftar instance layanan yang tersedia.
3. *Load balancer* pada API Gateway memilih instance dan meneruskan permintaan.

**Contoh Tools**:

* AWS Elastic Load Balancer (ELB)
* NGINX
* Spring Cloud Gateway (dengan Eureka)

**2. Client-side Discovery**

**Definisi**:  
Client-side Discovery adalah pola di mana *client* bertanggung jawab langsung untuk menemukan lokasi layanan dengan meminta daftar instance yang tersedia dari *Service Registry*, lalu memilih instance (biasanya dengan *load balancing* sederhana seperti Round Robin).

**Cara Kerja**:

1. Client mengakses *Service Registry* secara langsung.
2. Client memilih instance layanan dari daftar yang diberikan.
3. Permintaan dikirim langsung ke instance yang dipilih.

**Contoh Tools**:

* Netflix Ribbon (dengan Eureka)
* Consul Client

**3. Kelebihan dan Kekurangan Server-side dan Client-side Discovery**

**Server-side Discovery**

**Kelebihan**:

1. **Client lebih sederhana** karena tidak perlu mengimplementasikan logika discovery.
2. **Load balancing terpusat** ditangani oleh komponen seperti API Gateway atau load balancer, sehingga lebih mudah dikelola.
3. **Keamanan lebih baik** karena client tidak berinteraksi langsung dengan Service Registry.

**Kekurangan**:

1. **Single point of failure** jika load balancer atau API Gateway bermasalah, seluruh sistem bisa terganggu.
2. **Latensi tambahan** karena permintaan harus melewati perantara sebelum mencapai service.
3. **Kompleksitas infrastruktur** membutuhkan komponen tambahan seperti load balancer atau API Gateway.

**Client-side Discovery**

**Kelebihan**:

1. **Lebih cepat** karena client berkomunikasi langsung dengan service tanpa perantara.
2. **Lebih toleran terhadap kegagalan** karena tidak bergantung pada satu komponen pusat.
3. **Fleksibilitas tinggi** karena client bisa mengimplementasikan strategi load balancing sendiri.

**Kekurangan**:

1. **Client lebih kompleks** karena harus mengintegrasikan logika discovery dan load balancing.
2. **Load balancing terbatas** karena biasanya hanya menggunakan algoritma sederhana (seperti Round Robin).
3. **Tantangan keamanan** jika client mengakses Service Registry secara langsung.

**4. Service Registry**

**Definisi**:  
Service Registry adalah database yang menyimpan daftar instance layanan (service instances) beserta metadata (seperti alamat IP, port, status kesehatan) dalam arsitektur mikroservis. Berfungsi sebagai *temporary phonebook* untuk layanan yang dinamis.

**Fitur Utama**:

* **Pendaftaran Otomatis**: Layanan mendaftarkan diri saat startup (*self-registration*).
* **Health Check**: Memantau kesehatan instance (misal: heartbeat).
* **Deregistrasi**: Menghapus instance yang tidak sehat secara otomatis.

**Contoh Tools**:

* **Netflix Eureka**: Cocok untuk ekosistem Spring Cloud.
* **HashiCorp Consul**: Mendukung multi-datacenter dan fitur tambahan (KV store, ACL).
* **Apache Zookeeper**: Lebih rendah level, butuh konfigurasi manual.
* **Kubernetes**: Memiliki Service Registry bawaan (via kube-proxy dan DNS).

**Contoh Alur**:

1. Service A startup → mendaftar ke Eureka.
2. Service B ingin memanggil Service A → tanya Eureka untuk mendapatkan daftar instance.
3. Eureka mengembalikan alamat Service A yang aktif.