***1.Server-Side Discovery***

Server-Side Discovery adalah salah satu pola (pattern) dalam service discovery di mana komponen load balancer atau service registry di sisi server bertanggung jawab untuk mengarahkan permintaan klien ke instance layanan (service instance) yang tersedia.

Dalam arsitektur mikroservis, layanan sering kali berjalan dalam beberapa instance yang dinamis (muncul dan hilang karena scaling, failure, atau pembaruan). Server-side discovery memungkinkan klien mengirim permintaan ke satu titik masuk (misalnya load balancer) tanpa perlu mengetahui lokasi sebenarnya dari service instance.

Cara Kerja Server-Side Discovery

1. Service Registry (misalnya Eureka, Consul, Zookeeper) menyimpan daftar service instance yang aktif.
2. Service Instance melakukan registrasi ke Service Registry saat startup dan melakukan deregistrasi saat shutdown.
3. Ketika klien mengirim permintaan ke layanan, permintaan tersebut tidak langsung menuju ke service instance, melainkan ke load balancer atau API Gateway (misalnya Nginx, AWS ALB, Spring Cloud Gateway).
4. Load Balancer/API Gateway bertanya ke Service Registry untuk mendapatkan daftar service instance yang tersedia.
5. Load Balancer memilih instance yang sesuai (menggunakan algoritma Round Robin, Least Connections, dll.) dan meneruskan permintaan klien ke sana.

***2. Client-Side Discovery***

**Client Side Discovery** adalah pola dalam arsitektur mikroservis di mana **klien (aplikasi atau service lain)** bertanggung jawab untuk menemukan lokasi (alamat IP/port) dari **service yang dibutuhkan** secara dinamis.

**Cara Kerja**

1. Klien meminta daftar **service yang tersedia** dari **Service Registry** (seperti Eureka, Consul, atau ZooKeeper).

2. Klien memilih instance service yang sesuai (misalnya dengan **load balancing** round-robin).

3. Klien mengirim permintaan langsung ke instance yang dipilih.

***3. Kelebihan dan Kekurangan Server Side Discovery dan Client Side Discovery***

Kelebihan Server Side Discovery

-Klien Sederhana

Klien tidak perlu tahu alamat service, cukup kirim permintaan ke load balancer.

- Load Balancing Terpusat

Strategi load balancing (round-robin, least connections) dikelola di satu tempat.

- Keamanan Lebih Baik

Klien tidak perlu akses langsung ke Service Registry.

Kekurangan Server Side Discovery

- Single Point Of Failure

Jika load balancer down, seluruh sistem terpengaruh.

- Latensi Tambahan

Setiap permintaan harus melewati load balancer terlebih dahulu.

- Kompleksitas Infrastruktur

Membutuhkan pengaturan load balancer dan Service Registry.

Kelebihan Client Side Discovery

- Performansi Lebih Cepat

Tidak ada hop tambahan ke load balancer, mengurangi latensi.

- Fleksibilitas Tinggi

Klien bisa memilih strategi routing sendiri (misal: prefer instance terdekat).

- Tidak ada Bottleneck Terpusat

Tidak bergantung pada satu load balancer.

Kekurangan Client Side Discovery

- Kompleksitas di Klien

Setiap klien harus mengimplementasikan logika discovery & load balancing.

- Tidak cocok untuk multi-bahasa

Jika klien gunakan berbagai bahasa, logika discovery harus diimplementasikan ulang.

- Service Registry jadi kritis

Jika registry down, klien tidak bisa menemukan service.

***4. Service Registry***

Service Registry adalah komponen kunci dalam arsitektur mikroservis dan sistem terdistribusi yang berfungsi sebagai **database real-time** untuk melacak lokasi (alamat IP, port, health status) dari semua service yang tersedia. Ini memungkinkan **service discovery**, di mana aplikasi atau service lain dapat menemukan dan berkomunikasi dengan instance service yang aktif.

Fungsi Service Registry

**a. Pendaftaran Service (Registration)**

- Saat sebuah service dimulai, ia mendaftarkan diri ke registry (misal: mengirimkan alamat IP + port).

- Contoh: Service A memberitahu registry bahwa ia berjalan di 10.0.0.1:8080.

**b. Penyimpanan Daftar Service (Service Catalog)**

- Menyimpan metadata service (nama, versi, health check, endpoint).

**c. Health Checking (Pemantauan Kesehatan)**

- Registry secara berkala memeriksa apakah service masih aktif (misal via HTTP health check /health).

- Jika service mati, registry otomatis menghapusnya dari daftar.

**d. Service Discovery (Pencarian Service)**

- Klien (atau load balancer) meminta registry untuk mendapatkan daftar instance yang tersedia.

- Digunakan dalam **Client-Side** dan **Server-Side Discovery**.