1. Server-Side Discovery

Server-Side Discovery adalah pola dalam arsitektur mikroservis di mana klien tidak langsung menemukan lokasi layanan yang ingin diakses. Sebagai gantinya, permintaan dikirim ke load balancer atau gateway, yang kemudian mencari layanan tujuan menggunakan Service Registry dan meneruskan permintaan ke instance yang sesuai.

- **Cara Kerja:**

  - Klien mengirim permintaan ke load balancer atau API Gateway.
  - Load balancer melakukan pencarian pada Service Registry untuk menemukan instance layanan yang tersedia.
  - Load balancer meneruskan permintaan ke salah satu instance layanan berdasarkan kebijakan load balancing (misalnya, round-robin, least connections, dll.).

- **Contoh Implementasi:**

  - AWS Elastic Load Balancer (ELB)
  - Kubernetes Service dengan kube-proxy
  - Netflix Zuul atau Spring Cloud Gateway

2. Client-Side Discovery

Client-Side Discovery adalah pola di mana klien bertanggung jawab untuk menemukan layanan tujuan menggunakan Service Registry sebelum mengirim permintaan secara langsung ke instance layanan yang sesuai.

- **Cara Kerja:**

  - Klien pertama-tama meminta daftar instance layanan yang tersedia dari Service Registry.
  - Klien memilih salah satu instance berdasarkan kebijakan load balancing (biasanya dengan pustaka seperti Ribbon atau Eureka).
  - Klien langsung mengirim permintaan ke instance yang dipilih tanpa perantara.

- **Contoh Implementasi:**

  - Netflix Eureka + Ribbon
  - Consul
  - HashiCorp Consul + Envoy

3. Perbandingan Server-Side vs. Client-Side Discovery

Aspek	Server-Side Discovery	Client-Side Discovery
Kemudahan Implementasi	Lebih mudah bagi klien karena tidak perlu menangani discovery secara langsung.	Klien harus memiliki logika tambahan untuk mencari layanan dan melakukan load balancing.
Kinerja	Bisa lebih lambat karena adanya perantara (load balancer/gateway).	Lebih cepat karena komunikasi langsung antara klien dan layanan.
Skalabilitas	Cocok untuk skala besar karena mengurangi beban pada klien.	Bisa menjadi tantangan jika jumlah klien besar karena mereka harus menangani load balancing sendiri.
Ketergantungan	Bergantung pada load balancer atau gateway.	Bergantung pada pustaka load balancing di klien.
Contoh Penggunaan	API Gateway di Kubernetes, AWS ELB, Spring Cloud Gateway.	Netflix Eureka + Ribbon, HashiCorp Consul.

Kesimpulan:

- **Server-Side Discovery** lebih cocok jika arsitektur sudah menggunakan API Gateway atau Load Balancer.
- **Client-Side Discovery** lebih fleksibel untuk sistem terdistribusi dengan banyak instance layanan yang dinamis.

4. Service Registry

Service Registry adalah komponen yang berfungsi sebagai direktori pusat yang menyimpan daftar instance layanan yang tersedia dalam sistem terdistribusi. Layanan dapat mendaftarkan dirinya di Service Registry saat dijalankan dan menghapus pendaftarannya saat berhenti.

- **Fungsi Utama:**

1. **Pendaftaran Layanan** – Layanan yang baru dijalankan akan mendaftarkan dirinya ke Service Registry.
2. **Pencarian Layanan** – Klien atau Load Balancer dapat mencari layanan yang tersedia.
3. **Pemantauan Kesehatan (Health Check)** – Service Registry memeriksa status layanan dan menghapus layanan yang tidak sehat dari daftar.

- **Contoh Implementasi:**

  - **Netflix Eureka** (sering digunakan dalam ekosistem Spring Cloud)
  - **Consul** (dari HashiCorp)
  - **Zookeeper** (digunakan oleh Apache Kafka, Mesos, dll.)