Mhd. Rizki Asyafiq

2301083006

TK2A-MICROSERVICE

1. Server-side Discovery

Definisi:

Server-side Discovery adalah pola dalam arsitektur mikroservis di mana komponen infrastruktur (seperti load balancer atau API Gateway) bertanggung jawab untuk menemukan lokasi layanan (service instance) yang tersedia dan meneruskan permintaan klien ke instance tersebut.

Cara Kerja:

1. Klien mengirim permintaan ke endpoint statis (misalnya, load balancer atau API Gateway).
2. Server-side component (misalnya, Service Registry) menyimpan daftar instance layanan yang aktif.
3. Komponen ini memilih instance yang sesuai (dengan algoritma round-robin, load-based, dll.) dan meneruskan permintaan klien.

Contoh:

* AWS Elastic Load Balancer (ELB) yang mengarahkan traffic ke instance EC2.
* Kubernetes menggunakan kube-proxy untuk service discovery.

2. Client-side Discovery

Definisi:

Client-side Discovery adalah pola di mana klien bertanggung jawab untuk menemukan lokasi layanan dengan meminta daftar instance yang tersedia dari Service Registry, lalu memilih instance secara mandiri.

Cara Kerja:

1. Klien mengakses Service Registry (seperti Eureka, Consul) untuk mendapatkan daftar instance layanan.
2. Klien menggunakan algoritma (seperti round-robin atau cache) untuk memilih instance.
3. Klien mengirim permintaan langsung ke instance yang dipilih.

Contoh:

* Netflix OSS menggunakan Eureka untuk Client-side Discovery.
* Aplikasi berbasis Spring Cloud dengan Ribbon untuk load balancing di sisi klien.

3. Kelebihan dan Kekurangan

|  |  |  |
| --- | --- | --- |
| Aspek | Server-side Discovery | Client-side Discovery |
| Kelebihan | - Klien sederhana (tidak perlu tahu lokasi service).  - Sentralisasi manajemen (load balancing di satu tempat).  - Cocok untuk lingkungan cloud (e.g., AWS ELB). | - Lebih cepat (tidak ada hop tambahan ke load balancer).  - Fleksibilitas algoritma routing (klien bisa custom).  - Tidak ada single point of failure (jika Registry high-availability). |
| Kekurangan | - Bottleneck jika traffic tinggi (load balancer jadi titik kritis).  - Tambahan latency (hop ekstra).  - Kurang fleksibel untuk algoritma khusus. | - Klien lebih kompleks (harus implementasi discovery logic).  - Ketergantungan pada Service Registry.  - Potensi inconsistency jika cache klien kadaluarsa. |

4. Service Registry

Definisi:

Service Registry adalah database terdistribusi yang menyimpan daftar instance layanan beserta metadata (e.g., alamat IP, port, health status). Ini menjadi sumber kebenaran untuk pola discovery.

Fitur Utama:

1. Pendaftaran Otomatis: Layanan mendaftarkan diri saat startup (e.g., via heartbeat).
2. Health Check: Memantau kesehatan instance (misalnya, dengan HTTP health endpoint).
3. Pencarian: Memungkinkan klien atau server-side component untuk query instance yang tersedia.

Contoh Tools:

* Eureka (Netflix): Cocok untuk ekosistem Spring Cloud.
* Consul (HashiCorp): Mendukung multi-datacenter dan fitur tambahan (KV store, ACL).
* Zookeeper: Digunakan di Apache Kafka untuk koordinasi layanan.
* Kubernetes Service: Memiliki built-in registry menggunakan DNS dan label selector.

Contoh Alur:

1. Service A startup → mendaftar ke Registry.
2. Service B ingin memanggil Service A → query Registry untuk mendapatkan daftar instance.
3. Registry mengembalikan alamat Service A yang sehat.

Kesimpulan

* Server-side Discovery: Cocok untuk arsitektur sederhana dengan ketergantungan pada infrastruktur cloud.
* Client-side Discovery: Ideal untuk performa tinggi dan kontrol rute di sisi klien.
* Service Registry: Komponen kunci yang memungkinkan kedua pola discovery bekerja.