|  |  |  |
| --- | --- | --- |
| **Nama** | **:** | **Andini Dwiputri Basril** |
| **Nim** | **:** | **2301083011** |
| **Kelas** | **:** | **2A Teknik Komputer** |

**Tugas Microservice (Teori)**

1. **Jelaskan Tentang Server side Discovery**

Server-side Discovery adalah mekanisme di mana permintaan dari klien (misalnya aplikasi atau pengguna) tidak langsung dikirim ke layanan yang dituju, melainkan melalui sebuah perantara (biasanya load balancer atau API gateway). Perantara ini bertugas mencari tahu di mana letak layanan yang tersedia dengan memeriksa Service Registry (daftar layanan), lalu meneruskan permintaan klien ke sana.

**Cara kerja:**

* Klien mengirim permintaan ke service registry atau load balancer.
* Load balancer mengambil daftar instance layanan dari service registry.
* Load balancer meneruskan permintaan ke instance layanan yang tersedia.

**Contoh implementasi:**

* AWS Elastic Load Balancing (ELB)
* Kubernetes Service (ClusterIP, NodePort, LoadBalancer)
* Netflix Ribbon (bersama Eureka)

1. **Jelaskan Tentang Cleint side Discovery**

Dalam Client-side Discovery, klien langsung bertanya ke Service Registry untuk mengetahui di mana layanan yang dibutuhkan berada. Setelah mendapatkan daftar layanan yang tersedia, klien memilih sendiri server mana yang akan digunakan (biasanya dengan algoritma load balancing sederhana seperti round-robin).

**Cara kerja:**

* Klien melakukan query ke service registry untuk mendapatkan daftar instance layanan yang tersedia.
* Klien memilih salah satu instance dan mengirimkan permintaan langsung ke instance tersebut.

**Contoh implementasi :**

* Netflix Eureka (bersama Spring Cloud Load Balancer)
* Consul
* ZooKeeper

1. **Jelaskan kekurangan dan kelebihan dari no 1 \dan 2 diatas**

|  |  |  |
| --- | --- | --- |
| **Metode** | **Kelebihan** | **Kekurangan** |
| **Server side Discovery** | * Klien lebih sederhana karena tidak perlu menangani discovery sendiri. * Load balancing terpusat (konsisten dan mudah dikelola). * Cocok untuk arsitektur dengan banyak klien "dumb" (e.g., IoT). | * Menambah overhead di sisi server. * Performa bisa berkurang jika load balancer menjadi bottleneck. |
| **Cleint side Discovery** | * Mengurangi overhead server karena klien yang menangani load balancing. * Tidak memerlukan load balancer tambahan. * Fleksibel (klien bisa implementasi algoritma load balancing sendiri). * Scalability lebih baik (tidak bergantung pada bottleneck load balancer). | * Klien harus memiliki logika tambahan untuk menemukan layanan. * Perlu pembaruan daftar layanan jika ada instance baru. * Kurang cocok untuk klien sederhana |

1. **Jelaskan tentang Service Registry**

Service Registry adalah basis data yang menyimpan informasi tentang layanan-layanan yang tersedia dalam sebuah sistem terdistribusi. Setiap kali sebuah layanan (misalnya layanan pengguna atau layanan produk) dijalankan, ia akan mendaftarkan diri ke Service Registry.

**Fungsi Utama:**

* **Pendaftaran Layanan (Registration):**

Layanan baru melapor ke registry saat startup.

Contoh: Layanan Auth mendaftarkan diri dengan alamat auth-service:8080.

* **Pencarian Layanan (Discovery):**

Ketika klien membutuhkan sebuah layanan, ia bertanya ke registry.

Contoh: Klien meminta daftar server "Layanan Produk" yang aktif.

* **Pemantauan Kesehatan (Health Check):**

Registry secara berkala memeriksa apakah layanan masih hidup.

Jika layanan tidak merespons, registry menganggapnya mati dan menghapusnya.

**Contoh Tools:**

* Eureka (Netflix)
* Consul (HashiCorp)
* Zookeeper (Apache)