Nama : Hanif Istyansyah

NIM : 2301082010

Kelas : TK 2 B

MataKuliah : Microservice

1. Server-Side Discovery  
Server-side discovery adalah pola desain dalam arsitektur mikroservis di mana load balancer atau service proxy bertanggung jawab untuk menemukan layanan yang tersedia dan mengarahkan permintaan klien ke salah satu instance layanan tersebut. Dalam pola ini, klien tidak perlu mengetahui lokasi fisik dari layanan (misalnya, IP address atau port). Sebaliknya, klien hanya mengirimkan permintaan ke load balancer atau service proxy, yang kemudian akan meneruskan permintaan tersebut ke layanan yang sesuai.

Contoh implementasi: AWS Elastic Load Balancer (ELB), Nginx sebagai reverse proxy, Traefik.

Proses:

1. Klien mengirimkan permintaan ke load balancer.
2. Load balancer memeriksa daftar layanan aktif dari Service Registry.
3. Load balancer mengarahkan permintaan ke salah satu instance layanan yang tersedia.

2. Client-Side Discovery  
Client-side discovery adalah pola desain di mana klien bertanggung jawab untuk menemukan layanan yang tersedia. Dalam pola ini, klien langsung berkomunikasi dengan Service Registry untuk mendapatkan daftar layanan yang aktif. Setelah menerima daftar tersebut, klien memilih salah satu instance layanan dan mengirimkan permintaan langsung ke instance tersebut.

Contoh implementasi: Netflix Eureka (digunakan bersama Netflix Ribbon untuk load balancing), Consul.

Proses:

1. Klien menghubungi Service Registry untuk mendapatkan daftar layanan aktif.
2. Klien memilih salah satu instance layanan dari daftar tersebut.
3. Klien mengirimkan permintaan langsung ke instance layanan yang dipilih.

3. Kekurangan dan Kelebihan Server-Side Discovery vs Client-Side Discovery

Kelebihan Server-Side Discovery:

1. Mengurangi kompleksitas klien: Klien tidak perlu mengetahui detail implementasi layanan atau cara menemukannya. Semua logika routing ditangani oleh load balancer.
2. Pemeliharaan lebih mudah: Jika ada perubahan pada layanan (misalnya, penambahan atau penghapusan instance), hanya load balancer yang perlu diperbarui, bukan semua klien.
3. Kinerja lebih stabil: Load balancer dapat mengimplementasikan algoritma load balancing yang canggih (misalnya round-robin, least connections) untuk distribusi beban yang optimal.

Kekurangan Server-Side Discovery:

1. Tergantung pada load balancer: Jika load balancer gagal atau mengalami downtime, seluruh sistem dapat terganggu.
2. Latensi tambahan: Permintaan harus melewati load balancer sebelum mencapai layanan, yang dapat menambah sedikit latensi.

Kelebihan Client-Side Discovery:

1. Lebih fleksibel: Klien memiliki kendali penuh atas pemilihan layanan dan dapat menerapkan logika khusus (misalnya, prioritas geografis).
2. Mengurangi single point of failure: Tidak ada komponen pusat seperti load balancer yang menjadi titik kegagalan tunggal.
3. Performa lebih cepat: Klien dapat langsung mengakses layanan tanpa melalui perantara.

Kekurangan Client-Side Discovery:

1. Kompleksitas pada klien: Klien harus memiliki logika untuk berinteraksi dengan Service Registry dan memilih layanan.
2. Pemeliharaan lebih sulit: Jika ada perubahan pada layanan, semua klien harus diperbarui atau dikonfigurasi ulang.
3. Potensi kesalahan pada klien: Jika klien tidak mengimplementasikan logika pemilihan layanan dengan benar, dapat menyebabkan masalah seperti overloading pada satu instance layanan.

4. Service Registry  
Service Registry adalah komponen inti dalam arsitektur mikroservis yang bertanggung jawab untuk mencatat dan menyimpan informasi tentang layanan yang tersedia, seperti alamat IP, port, status kesehatan, dan metadata lainnya. Layanan yang baru dimulai (atau dihentikan) harus mendaftarkan dirinya (atau mencabut dirinya) di Service Registry agar dapat ditemukan oleh klien atau load balancer.

Fungsi utama Service Registry:

1. Registrasi Layanan: Layanan yang baru dimulai secara otomatis mendaftarkan dirinya di Service Registry.
2. Penemuan Layanan: Klien atau load balancer dapat meminta daftar layanan aktif dari Service Registry.
3. Health Check: Service Registry secara berkala memeriksa status kesehatan layanan untuk memastikan bahwa hanya layanan yang aktif yang dicantumkan.
4. Deregistrasi Layanan: Layanan yang dihentikan atau gagal health check akan dicabut dari daftar layanan aktif.

Contoh implementasi Service Registry: Eureka (dari Netflix), Consul (dari HashiCorp), Zookeeper (dari Apache), etcd (dari CoreOS).

Proses kerja:

1. Ketika sebuah layanan dimulai, ia mendaftarkan dirinya di Service Registry dengan memberikan informasi seperti alamat IP dan port.
2. Service Registry memperbarui daftar layanan aktif.
3. Klien atau load balancer meminta daftar layanan aktif dari Service Registry untuk menemukan layanan yang dibutuhkan.
4. Jika layanan mati atau tidak sehat, Service Registry akan menghapusnya dari daftar.

**Kesimpulan**

1. Server-side discovery cocok untuk sistem dengan banyak klien karena mengurangi kompleksitas klien dan memusatkan logika routing di load balancer.
2. Client-side discovery lebih fleksibel tetapi membutuhkan logika tambahan di sisi klien.
3. Service Registry adalah komponen penting dalam kedua pola di atas karena menyediakan informasi real-time tentang layanan yang tersedia.