Nama : Dhea Safira

Kelas : TK 2A

Prodi : D3 Teknik Komputer

Mata Kuliah : Pemrograman Microservice

Ringkasan mengenai materi VIDIO 18 Service Registry:

Kesimpulan dari video yang dibahas adalah mengenai "service registry" dalam konteks microservices, dan bagaimana hal ini dapat mengatasi beberapa masalah yang muncul ketika menggunakan metode client-side discovery.

1. **Kekurangan Client-Side Discovery**: Salah satu masalah utama dalam client-side discovery adalah ketidakstabilan jumlah node (sering bertambah dan berkurang) yang menyulitkan konfigurasi layanan, terutama dalam lingkungan yang menggunakan autoscaling. Menggunakan logik sederhana seperti round-robin tidak cukup untuk menangani ini secara efisien.
2. **Pengenalan Service Registry**: Service registry adalah solusi untuk menyimpan informasi lokasi dari setiap service. Daripada menyimpan informasi endpoint di masing-masing service, semua informasi ini disimpan di satu tempat, yaitu service registry. Dengan demikian, ketika sebuah service baru dinyalakan atau dimatikan, informasi ini otomatis diperbarui di service registry tanpa perlu perubahan manual di konfigurasi setiap service.
3. **Proses Registrasi dan Discovery**:
    - Ketika sebuah service dinyalakan, ia akan mendaftarkan lokasinya ke service registry.
    - Ketika service membutuhkan komunikasi dengan service lain, ia akan bertanya ke service registry untuk mendapatkan informasi lokasi service yang dibutuhkan.
    - Jika ada perubahan dalam jumlah node, service registry akan mengupdate informasi tersebut dan menginformasikannya kepada service yang memintanya.
4. **Optimisasi Kecepatan**: Agar tidak terjadi delay karena terus menerus harus bertanya ke service registry, service bisa menyimpan informasi yang diterima untuk jangka waktu tertentu (misalnya setiap 10 detik). Sehingga, pertanyaan ke service registry tidak dilakukan setiap kali ada permintaan, tetapi secara berkala.
5. **Health Check**: Beberapa service registry dilengkapi dengan fitur health check, yang secara berkala memeriksa apakah setiap service yang terdaftar masih aktif dan berfungsi dengan baik. Jika sebuah service tidak merespon atau merespon dengan kesalahan, service registry akan menganggap service tersebut tidak sehat dan tidak akan memberitahukan lokasinya kepada service lain yang membutuhkannya.
6. **Contoh Service Registry**: Beberapa contoh populer dari service registry adalah HashiCorp Consul dan Netflix Eureka, yang keduanya bersifat free dan open source.

Dengan menggunakan service registry, manajemen lokasi dan kesehatan service dalam arsitektur microservices menjadi lebih efisien dan otomatis, sehingga mengurangi beban konfigurasi manual dan meningkatkan ketersediaan serta reliabilitas layanan.

Ringkasan mengenai materi VIDIO 19 Centralized Configuration :

Video tersebut membahas tentang pentingnya **centralized configuration** dalam microservice architecture. Berikut ini adalah kesimpulan dari isi video tersebut:

1. **Konfigurasi Penting**: Setiap aplikasi membutuhkan konfigurasi, seperti pengaturan database atau API keys. Konfigurasi ini biasanya disimpan dalam file, environment variables, atau database.
2. **Tantangan Penyimpanan Konfigurasi**: Setiap metode penyimpanan konfigurasi memiliki kelemahan. Misalnya, mengubah konfigurasi dalam file atau environment variables membutuhkan restart aplikasi atau akses langsung ke server.
3. **Centralized Configuration**: Untuk mengatasi masalah ini, digunakan konsep centralized configuration, di mana semua konfigurasi disimpan dalam satu service khusus yang disebut centralized configuration service.
4. **Keuntungan**:
    - **Kemudahan Pengelolaan**: Memungkinkan pengelolaan konfigurasi dari satu tempat.
    - **Keamanan**: Data konfigurasi dapat dienkripsi untuk melindungi informasi sensitif seperti username dan password.
    - **Fleksibilitas**: Memudahkan perubahan konfigurasi yang otomatis diterapkan ke semua microservices yang terkait tanpa perlu restart manual.
5. **Aplikasi untuk Centralized Configuration**: Beberapa tools yang dapat digunakan untuk centralized configuration antara lain:
    - **Consul**: Bisa digunakan untuk service registry dan konfigurasi, namun tidak mendukung enkripsi.
    - **HashiCorp Vault**: Menyediakan enkripsi yang lebih aman untuk data sensitif.
    - **Spring Cloud Config**: Umum digunakan dalam komunitas Java.
    - **Etcd** dan **Zookeeper**: Lainnya yang dapat digunakan tergantung kebutuhan.
6. **Implementasi**: Penggunaan centralized configuration memungkinkan perubahan konfigurasi secara real-time, meningkatkan keamanan dan efisiensi pengelolaan aplikasi, serta memudahkan pelacakan siapa yang melakukan perubahan.

Video ini menekankan bahwa centralized configuration adalah solusi praktis dan aman untuk mengelola konfigurasi dalam arsitektur microservices, serta memberikan beberapa opsi aplikasi yang dapat digunakan untuk mengimplementasikannya.