1. **Agenda**:
   1. Instead of writing code to achieve the **“Service Discovery and Registration Pattern”**, we’re going to spend some time on how Spring Cloud is going to support us to implement this **“Service Discovery & Registration Pattern”**.
2. Spring Cloud makes the “Service Discovery & Registration” setup very easy to undertake with the help of **Components** that you’re seeing on the screen.
3. **Components**: Using these 3 components, we’re going to implement “**Service Discovery and Registration Pattern**”.
   1. **Netflix’s Eureka Service**:
      1. It acts as **“Service Discovery agent”** (You can say **Service Registry**)
      2. People call it as **“Eureka Server”.**
      3. It maintains all the **registry details** about all the microservices inside your application in a centralized location.
      4. Why called “**Spring Cloud Netflix Eureka**”?
         1. This component is coming from Netflix which was later on assigned to Spring Cloud so that other organizations and developer communities can use it.
         2. It is **production** **battle tested** which handles all the scenarios that we discussed in Service Discovery and Registration.
      5. This is very stable and a perfect solution that you can use to implement **“Service Discovery and Registration Pattern”.**
      6. **Other famous “Service Discovery and Registry Service” agents are**
         1. Etcd
         2. Consule
         3. Apache Zookeeper
   2. **Spring Cloud Load Balancer**: Will be automatically added if Netflix Eureka Server or client dependency is added.
      1. It is a library which is provided by Spring Cloud Project itself.
      2. This is for client-side load balancing.
   3. **Netflix Feign**:
      1. Using this component, one microservice first fetches the details of another microservice from the Eureka Server so that the former can invoke the later directly by using fetched registration details.
4. 