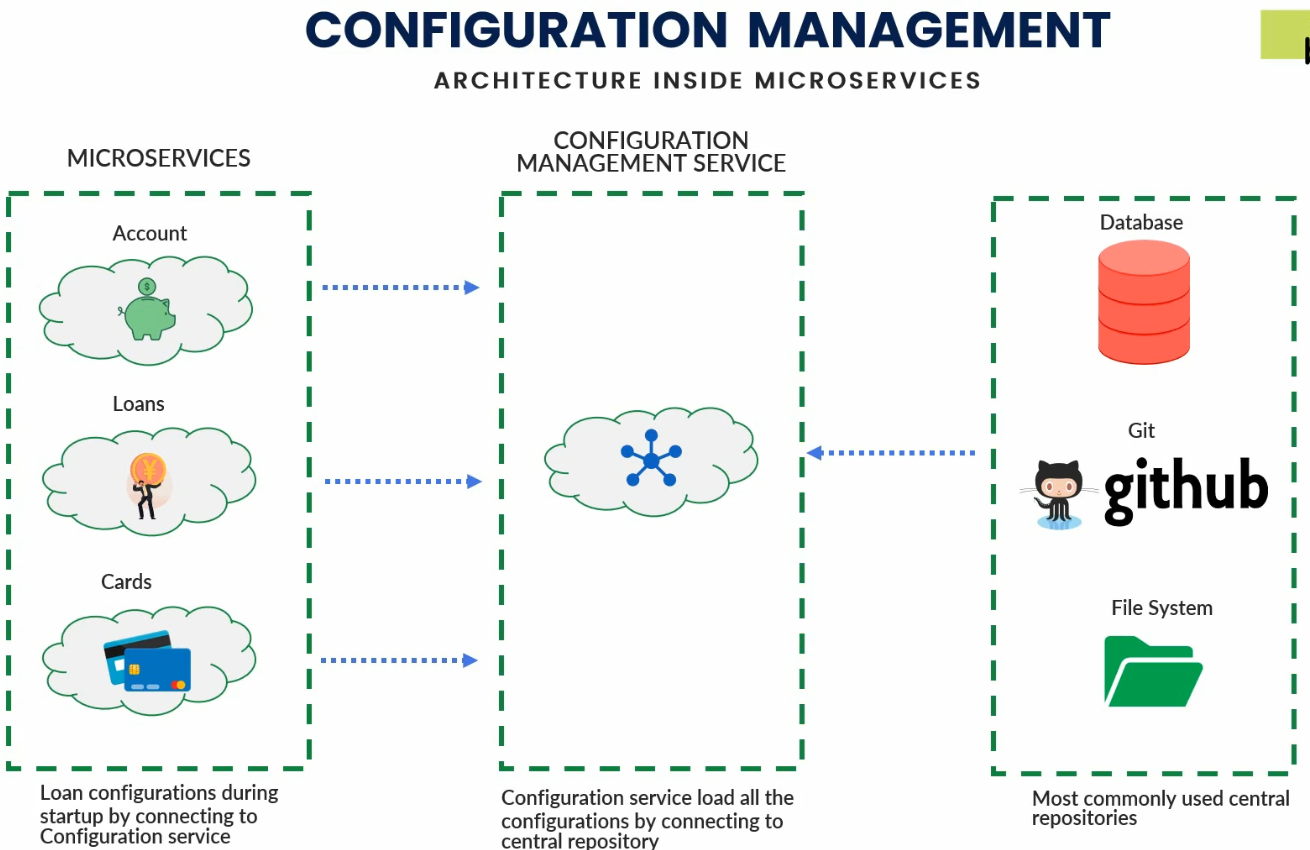
1. **In Last Lecture:**
   1. We studied the challenges we face in Microservice Architecture in terms of configuration management.
2. **Agenda**:
   1. Ideal architecture to be followed to resolve the challenges faced for configuration management.
3. 
4. Actually, we create a **configuration management service** in an isolated way similar to we create microservices around the business entities.
5. So, **Configuration Management Microservice loads** all the configurations for all Microservices and for all environments.
   1. **For example**:
      1. Suppose we have 3 microservices 🡺 Accounts, Loans, Cards 🡺 Total 3
      2. Suppose we have 3 environments 🡺 Development, Test, Production 🡺 Total 3
      3. Now, configuration management microservice will maintain 9 different kinds of configurations.
6. Many Options where we can store configurations:
   1. **Github**: Most commonly used central repository to store configurations is **Github**.
   2. **File System**: On your local computer’s file system or in local network.
   3. **Cloud Storage:** 
      1. **Amazon S3 Bucket**
      2. Or some other cloud storage.
   4. **DB**: Few projects use DB.
7. Brief Approach:
   1. We can maintain all our configurations in a centralized repo like Github, File System, DB.
   2. Configure the configuration management service to read from that centralized repo.
   3. Expose all those properties based on microservice’s name and environment name together through REST Endpoints to all our microservices (Account, Loan, Card).
   4. Do configuration inside your business microservice so that they read their respective configurations based on env from Config Management Service through the exposed endpoints.
8. We will also see how to make any change to the configurations effective in configuration management microservice and business microservices without restarting them.  
   Disadvantages if we have to restart
   1. If 50 microservices and each has 4 instances, then total 200 microservices which would take enough amount of time to restart.
   2. Users’ operations would be affected.
9. In next lecture, let’s see how Spring Cloud can help us to achieve this architecture through a sub project 🡺 **Spring Cloud Config**.