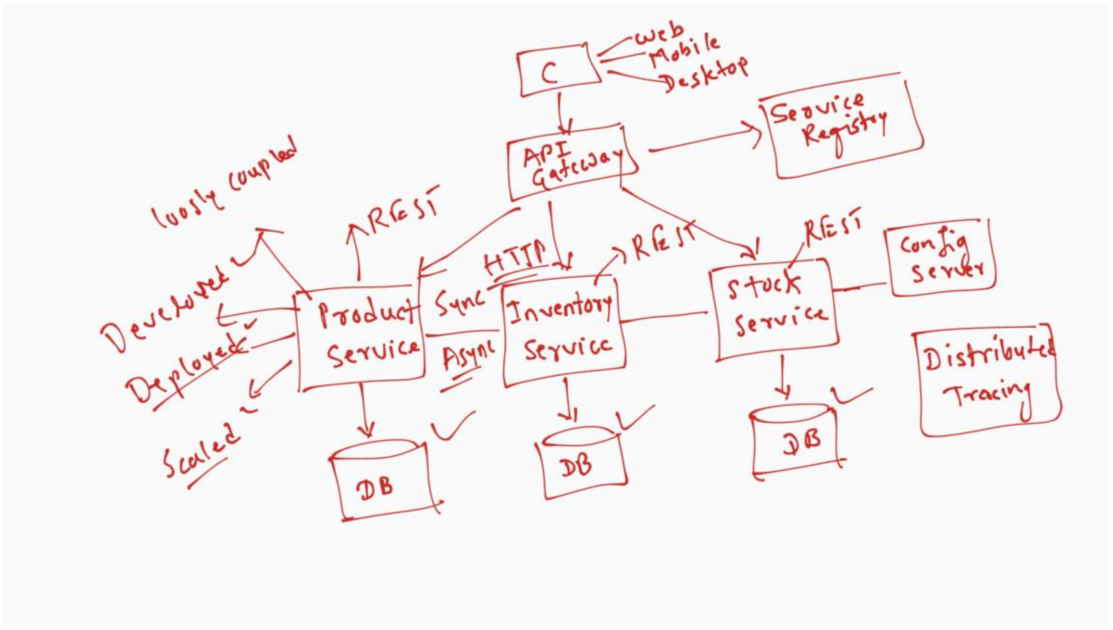
What are Microservices and How to Build Microservices in Java?

What are Microservices or Microservice Architecture?

Well, a microservice architecture enables large teams to build scalable applications that are composed of many loosely coupled services.

Here is what a typical microservice architecture looks like. For example, consider this microservice architecture for a simple shopping cart application. It has different services like product service, inventory service, and stock service, and these are the independent and loosely coupled services in the microservices projects.



Each microservice has its own database. For example, product service has its own database, inventory service has its own database, and stock service has its own database.

In the microservices project, all the microservices are loosely coupled. So loosely coupled, meaning all the services in a microservices project are independent of each other and each microservice should be developed independently and each microservice should be deployed independently and each microservice should be scaled independently.

So basically Microservice following characteristics:

- Each microservice can have its own database.
- Each microservice should be developed independently
- Each microservice should be deployed independently
- Each microservice should be scaled independently

In microservices projects, the services can communicate with each other. For example, product service can communicate with inventory service and inventory service can communicate with stock service. Microservice can communicate with multiple services as well.

Well, there are two types of communication styles. One is synchronous and another is asynchronous.

In the case of synchronous, we can use the HTTP protocol to make an HTTP request from one microservice to the microservice.

And in the case of asynchronous communication, we have to use a message broker for asynchronous communication between multiple microservices. For example, we can use a RabbitMQ or Apache Kafka as a message broker in order to make an asynchronous communication between multiple microservices and each microservice in a microservices project can expose REST API's.

Key Components in a Microservices Architecture

Now let's take a look into the key components in a typical microservices architecture.

Well, the key component is the API gateway. Well, whenever the client sends a request to the API gateway and then an API gateway will route that request to the relevant microservices All right.

The client can be a web application, a mobile application, or a desktop application and whenever a client wants to consume the REST API's of backend services, the client has to first send a request to the API gateway, and then the API gateway will route that request to the relevant microservice.

Here one more key component is a service registry. Well, all the microservices in our microservice project will register to the service registry, and the API gateway will discover the particular microservice hostname and port using the service registry so that the API gateway can allow that request to a particular microservice.

One more key component is the config server. So this config server component will basically externalize the configuration of microservices.

One more key component is distributed tracing. Well, in order to maintain the logs or complete log hierarchy for a particular HTTP call from start to end, we can use distributed tracing.

One more key component is Security. We can implement centralized security in API-Gateway.

So these are the few key components in a microservices architecture.