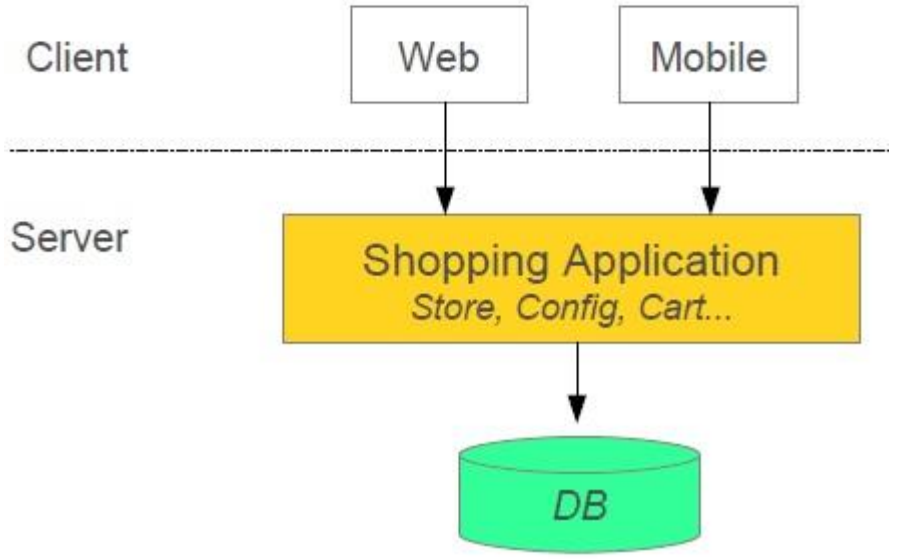
MICROSERVICES

2020

OLD TIMES



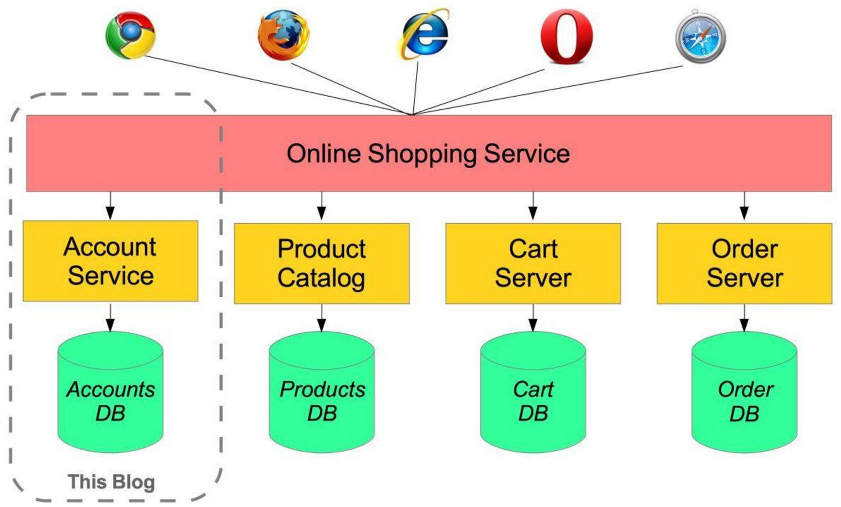
Shopping system without Microservices (Monolith architecture). In this architecture we are using Monolith architecture i.e. all collaborating components combine all in one application

WHAT IS MICROSERVICES ARCHITECTURE?

Microservices architecture allows to avoid monolith application for large system. It provide loose coupling between collaborating processes which running independently in different environments with tight cohesion.

Microservices allows us to break our large system into number of independent collaborating processes.

For example imagine an online shop with separate microservices for user-accounts, product-catalog order-processing and shopping carts



MICROSERVICE'S CHARACTERISTIC

Loose coupling

- application build from collaboration services or processes, so any process change without effecting another processes
- effect of changes isolated

Tight cohesion

- an individual service or process that deals with a single view of data
- code perform a single well defined task

MICROSERVICE BENEFITS

- Smaller code base is easy to maintain
- Easy to scale as individual component
- Technology diversity i.e. mixing libraries, databases, frameworks etc.
- Fault isolation i.e. a process failure should not bring whole system down
- Better support for smaller and parallel team
- Independent deployment
- Deployment time reduce

MICROSERVICE CHALLENGES

- Difficult to achieve strong consistency across services
- ACID transactions do not span multiple processes
- Distributed System so hard to debug and trace the issues
- Greater need for end to end testing
- Required cultural changes in across teams like Dev and Ops working together even in same team

SPRING CLOUD

Spring Cloud and Discovery server

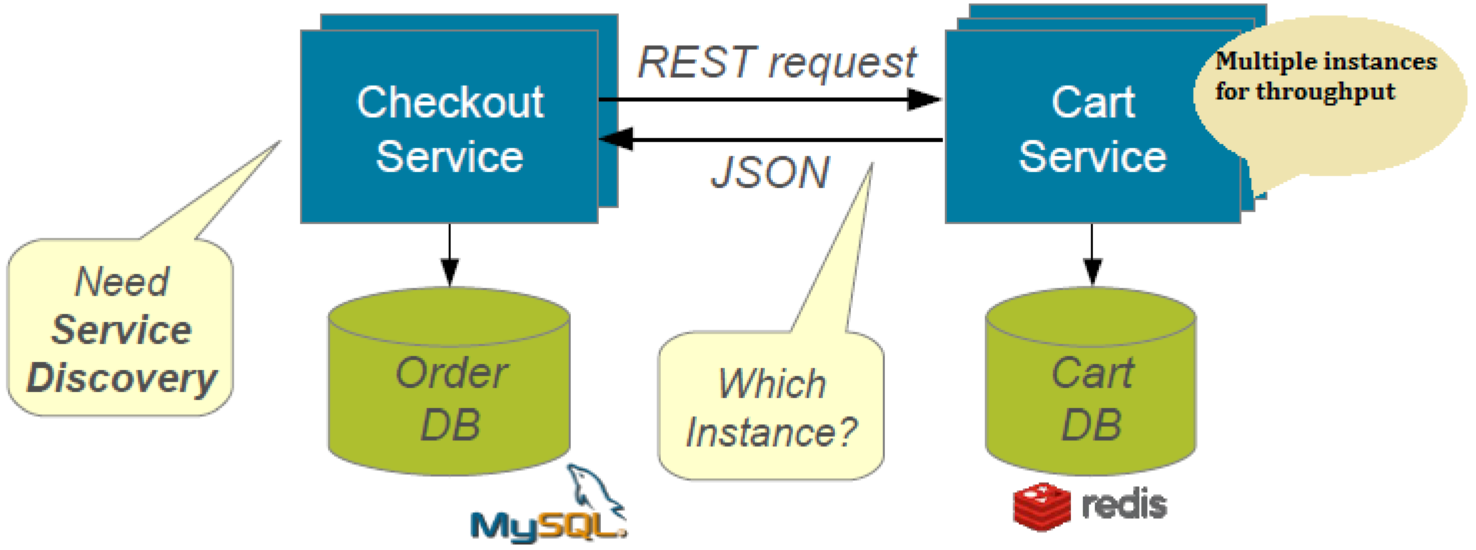
- building blocks for Cloud and Microservices
- provides microservices infrastructure: i.e provides use services such as Service Discovery, Configuration server and Monitoring.
- provides several other open source projects like Netflix OSS.
- uses Spring Boot style starters
- provides Platform as a Service like AWS

Spring Cloud supports

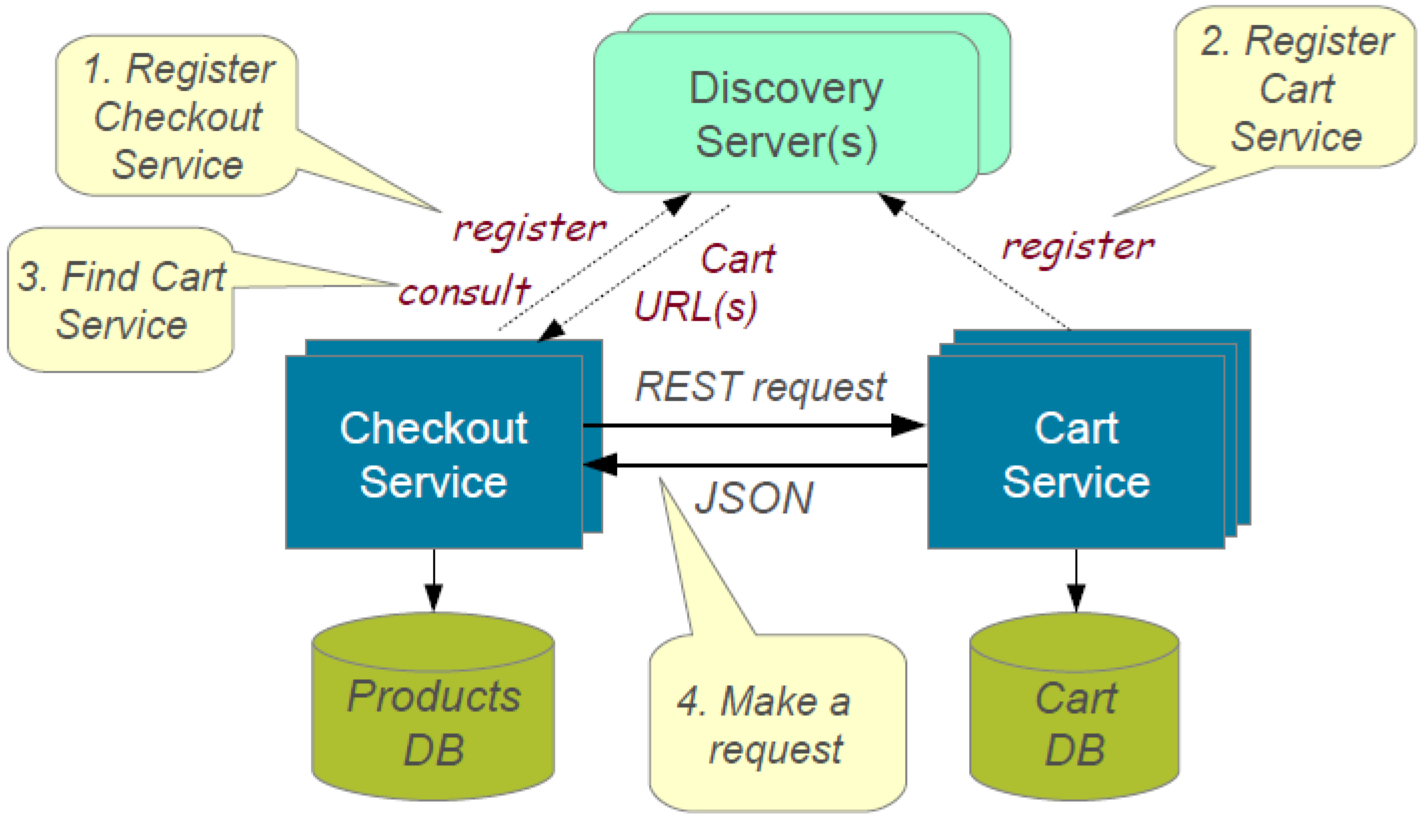
- Cloud Integration
- Dynamic Reconfiguration
- Service Discovery *(How do services find each other?)*
- Security
- Client-side Load Balancing *(How do we decide which service instance to use?)*

SERVICE DISCOVERY

- Problem without discovery
  - Finding right services
  - Running multiple instances for a service



IMPLEMENTING SERVICE DISCOVERY



RIBBON FOR LOAD BALANCING

Ribbon primarily provides client-side load balancing algorithms.

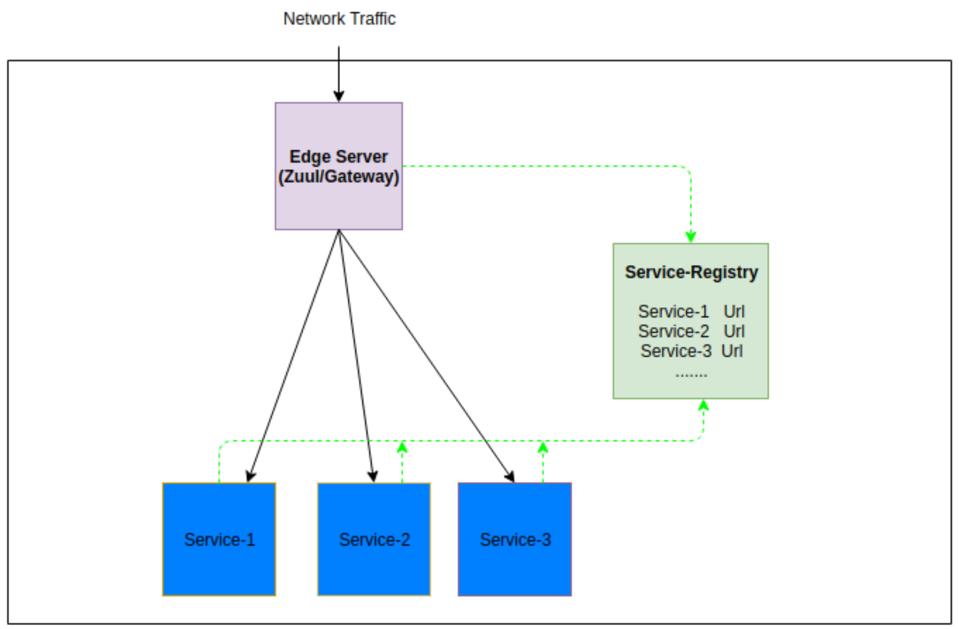
Apart from the client-side load balancing algorithms, Ribbon provides also other features:

- **Service Discovery Integration** – Ribbon load balancers provide service discovery in dynamic environments like a cloud. Integration with Eureka and Netflix service discovery component is included in the ribbon library
- **Fault Tolerance** – the Ribbon API can dynamically determine whether the servers are up and running in a live environment and can detect those servers that are down
- **Configurable load-balancing rules** – Ribbon supports *RoundRobinRule, AvailabilityFilteringRule, WeightedResponseTimeRule* out of the box and also supports defining custom rules

Ribbon API works based on the concept called "Named Client". While configuring Ribbon in our application configuration file we provide a name for the list of servers included for the load balancing.

ZUUL FOR REVERSE PROXY

- Used for routing an application
- **Problem statement**: many small applications running on different hosts and ports therefore how clients (Web Applications in browsers, Mobile apps, Third-party apps making a web service call, etc.) can access these end microservices without knowing their hosts and ports?
- **Zuul** creates a common entry point to our microservices, not only free the clients from knowing deployment details about all of the backend services, but also reduce development effort on the server side. At the same time, if an end microservice has multiple instances running, we can do load balancing at this entry point.


Network Traffic
Edge Server (Zuul/Gateway)
Service-Registry
Service-1 Url
Service-2 Url
Service-3 Url
....
Service-1
Service-2
Service-3