Monolith Application

SOA (focusing on web services)

Microservice (Design pattern)

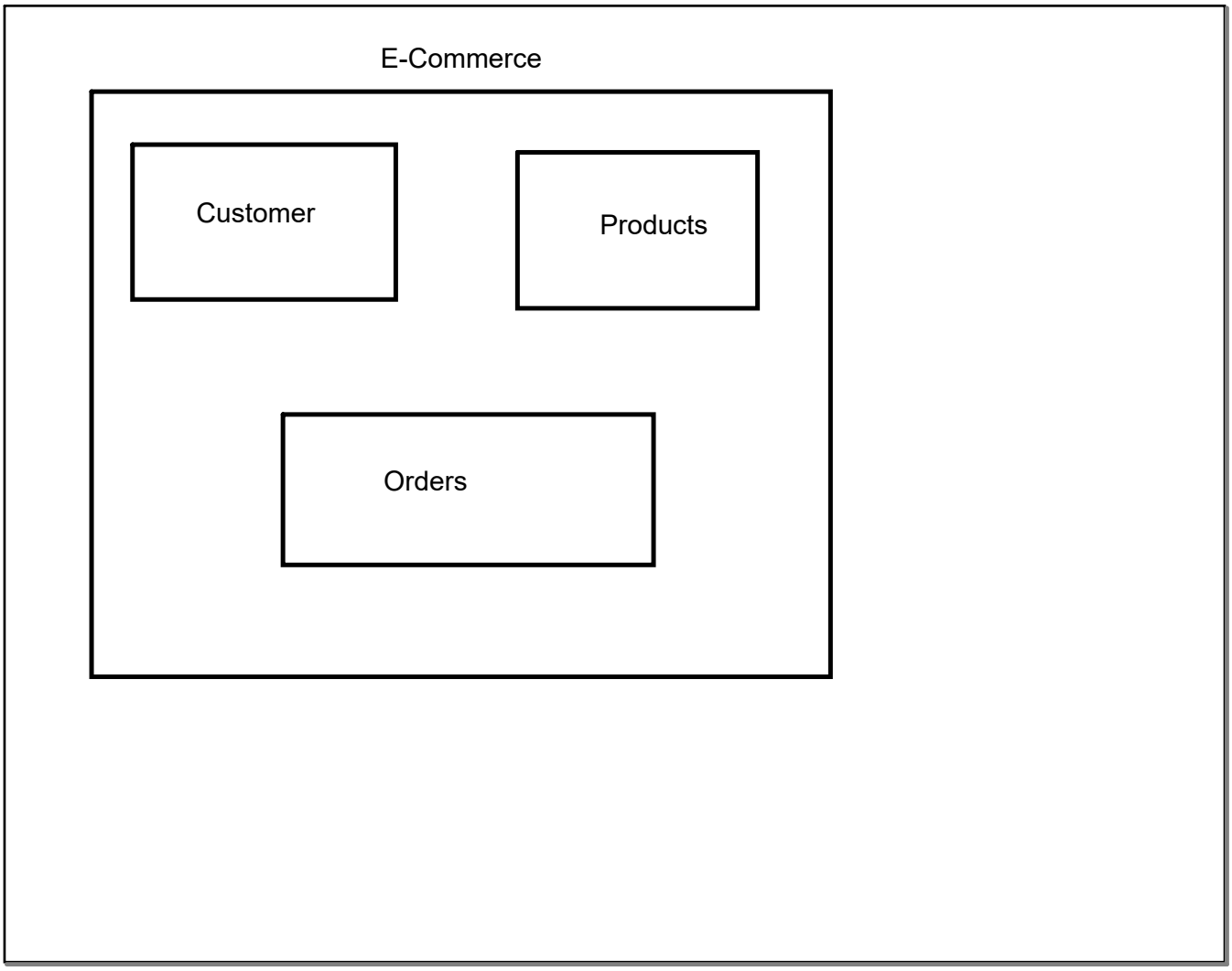
Monolith Applications

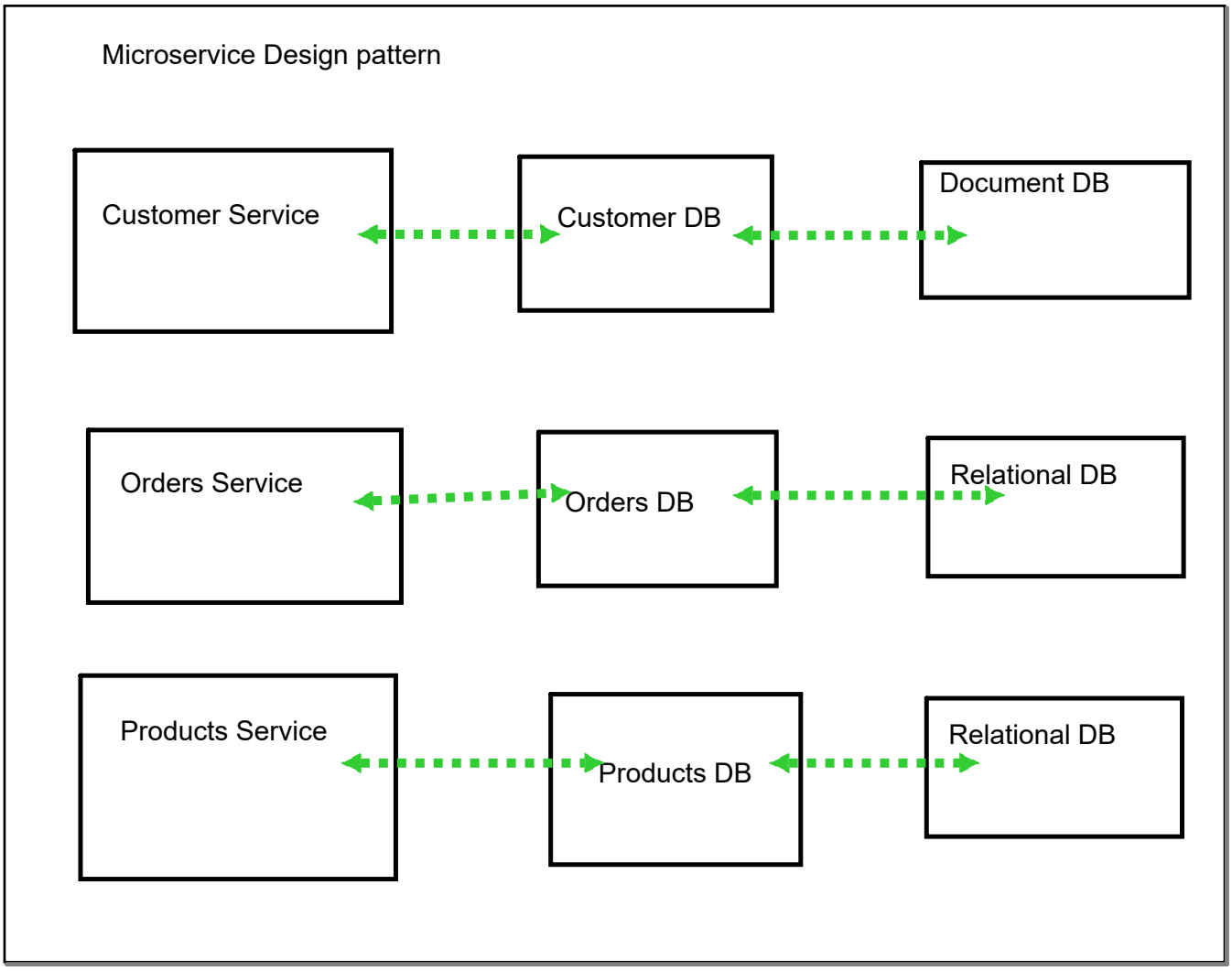
#One change could break entire app

#Need to rebuild on any changes

#responsive to changes

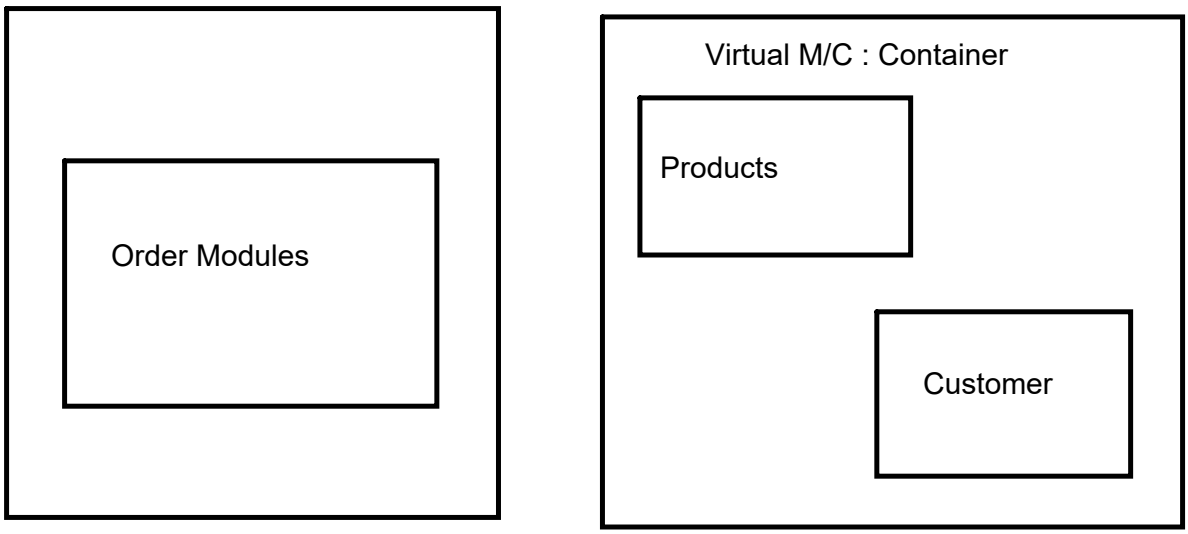
#no compatible to Agile methodologies


E-Commerce
Customer
Products
Orders


Microservice Design pattern
Customer Service
Customer DB
Document DB
Orders Service
Orders DB
Relational DB
Products Service
Products DB
Relational DB

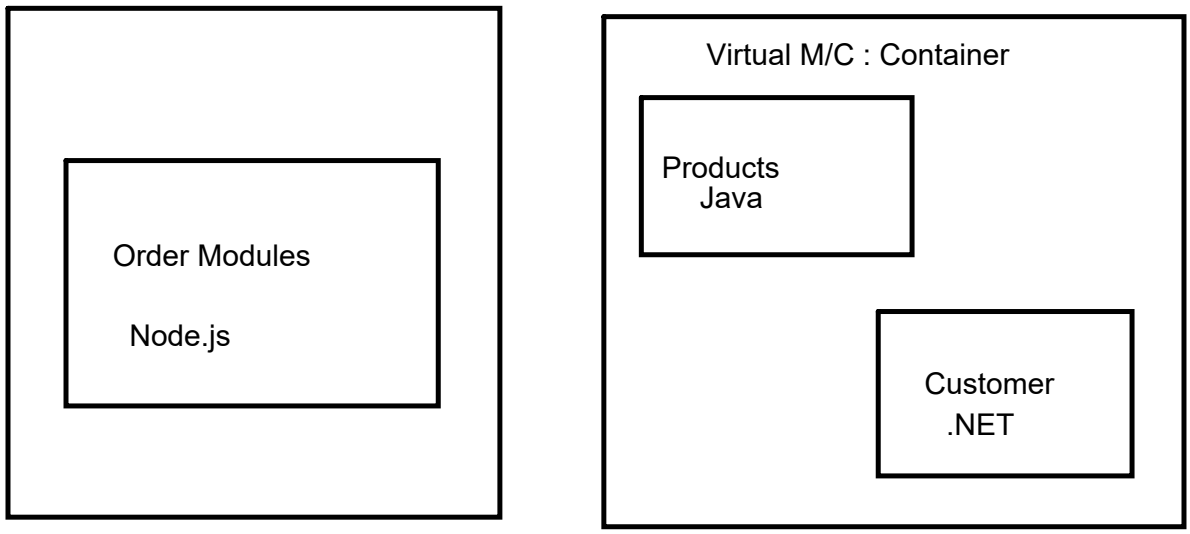
Deployment Advantage

Deploy service components on diff machine, load distribution



More choice of underlying technology

#max interaction approach among micro services are platform independent (REST)



Agile methodology : Services can be easily replaced/upgraded : continuous delivery

#maintain diff version of same service, other services can decide when to switch ...

#Even if a particular service break down , entire application does not suffer....

#Application can be scaled-up scaled -down on demand, as new in instances (containers) of a specific services can be spun up on demand

Key Design Elements :

#Keep services relatively small

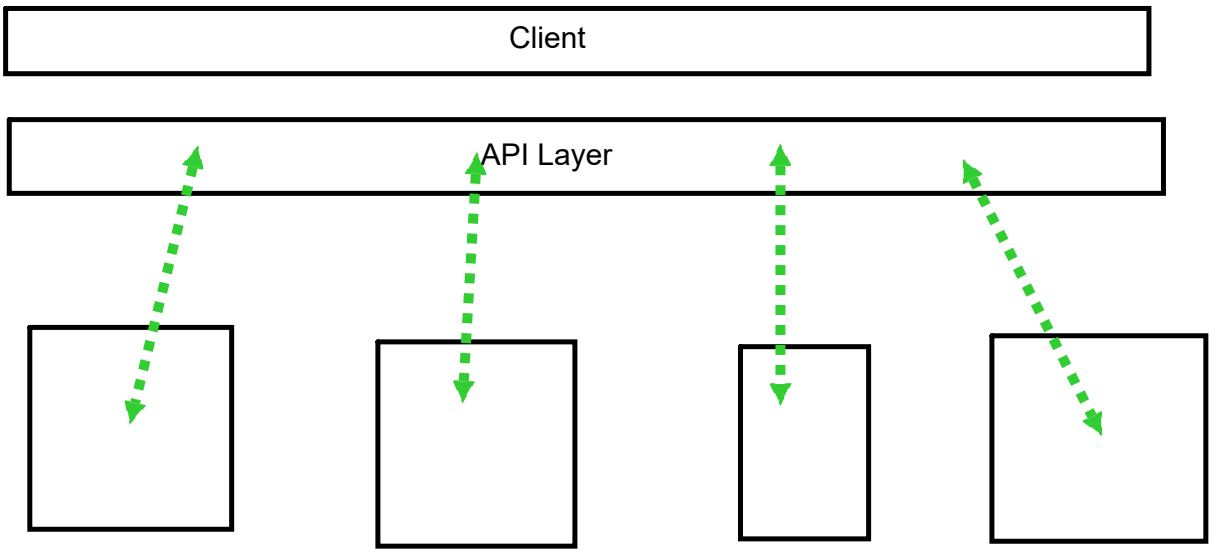
#Deployed independently

#Easy to monitor,maintain, able to isolate failure.

#keep db seperate if possible,

#try to develop service in reusable manner

Abstract layer in top of service (API Layer)

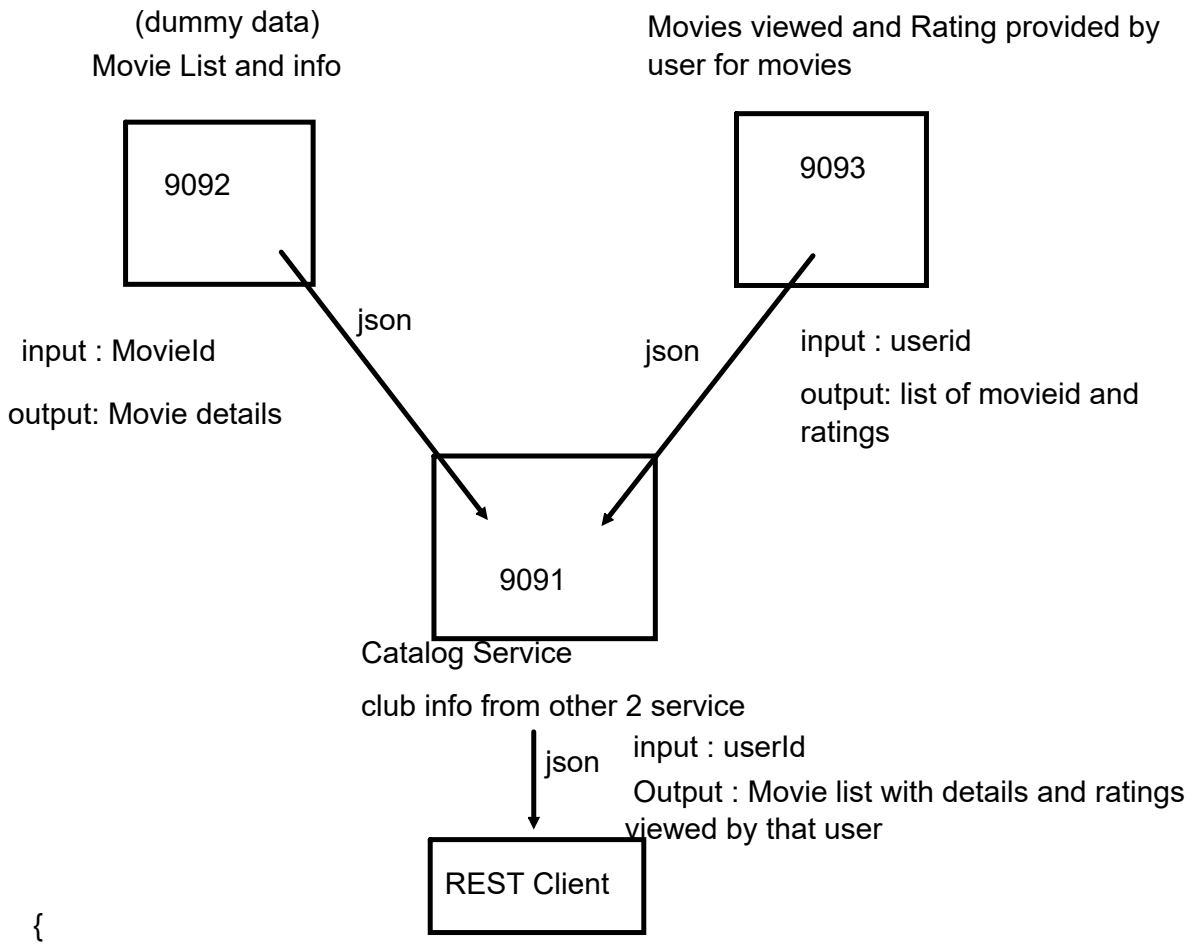


Technology : Spring boot

Use case:

#Movie Streaming platform

#lists the movie details viewed by a particular user along with ratings provided by that user




```
{
    movies : [
        {name:"", desc:"", rating : 1},
        {name:"", desc:"", rating : 5},
        {name:"", desc:"", rating : 2}
    ]
}
```

Go and make our CatalogService Active...

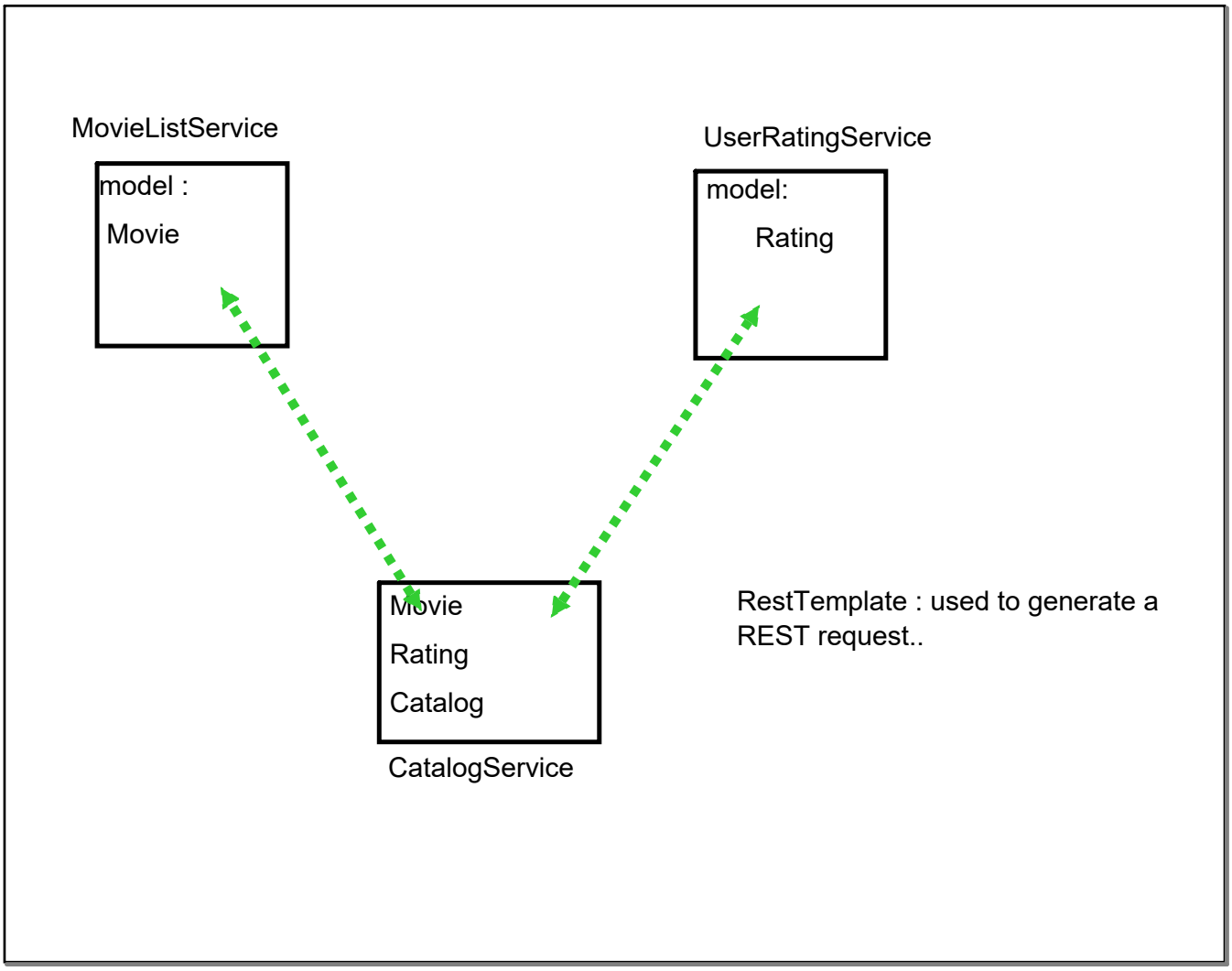
Go and make our MovieList Service Active

Go and make out Rating Service Active...

#Right now the REST service exposed by Rating Service

Input : movieId

Output : Rating for that movie (assume that movie id provided id viewed by user)


MovieListService
model :
Movie
UserRatingService
model:
Rating
Movie
Rating
Catalog
CatalogService
RestTemplate : used to generate a
REST request..

Bad Practice!!!

Rating Service : returning a list of records

Catalog Service: waiting for list to receive

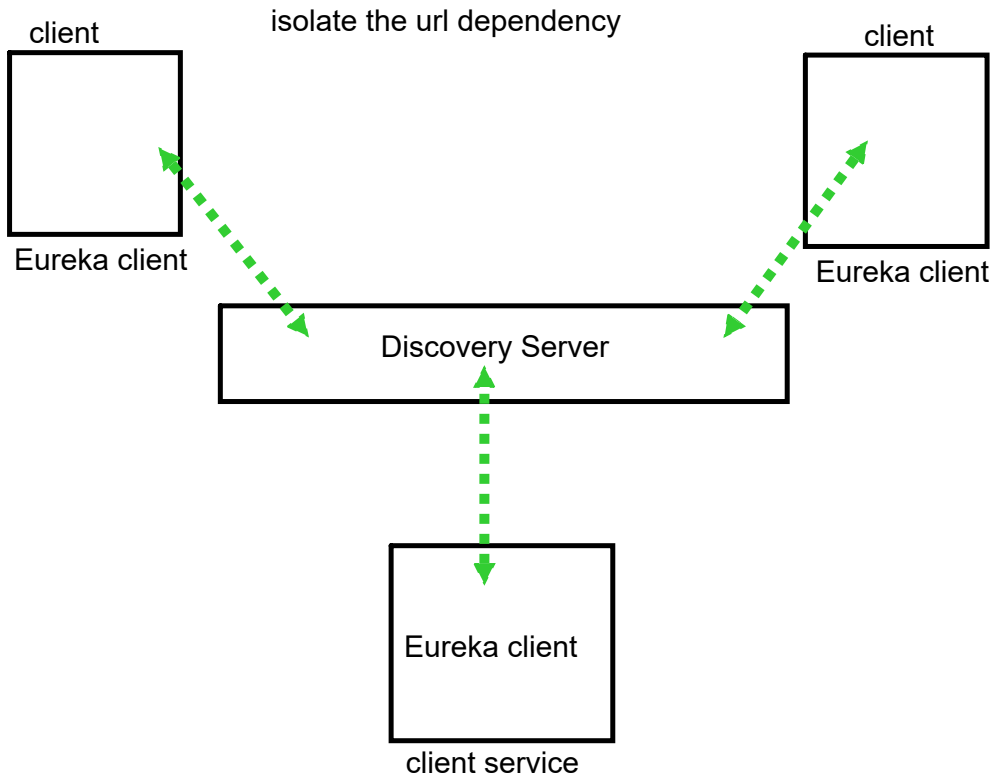
Transaction of Lists....

1. any modification of composition of data to be transacted is not going to incorporated in List
2. List can't be specified as a response type on dest

Solution : create a wrapper class around the list

#allow to change composition

#allow to specify it as response type



#hardcoded url in client(catalog) service :

1 .endpoint might change

2. these service might be launched on dyanamic ports

3. might be mapped ports

4. might be multiple instance are running of same service across multiple ports for load balancing

Solution :

Introduce a Discovery Server!!!

==> All microservice registers themselves with discovery server

==>Service ask for details about other service to interact with...

Discovery Server : Spring cloud tech

Eureka Server : netflix : exists as dependency (cloud dependency)

Hysteria

Zull

==>required to create a spring boot application add eureka server as dependency

==>port number for discovery server (8761 : default port) , eureka clients are going to auto find this server and registers themselves

==>change the absolute url with the registered service name

==>need to tell the restTemplate object to interact with Discovery server and fetch the services ..

aws : (amazon web services)

Cloud web hosting service:

=>flexible

=>reliable

=>scalable (efficient)

=>cost-effective

Cloud based web-hosting

#layers of abstraction

==>no need to plan for server and IT infra in advance

#hundred of server can be spin up on demand

#no upfront costing (pay per use)

#no long term commitments

#can be shared by mlt org for computing and storage

Cloud Types...

public : dependency on third - party provider

private : managed by org or dedicated service by third party

hybrid : combination

Cloud based service (abstraction)

IaaS : (Infra as a service)

provides the low level infra facilities : customer need to configure application level resources

PaaS (Platform as a Service) : we get various and we can applications around them

SaaS (Software as a Service) : end-user app with administrative capacity to customer

Technical improvisation :

CaaS : (Container as a Service) : Container based service

FaaS (Function as a Service) : Serverless Computing

spin up service using simple lambda function