MSA – MicroService Architecture

SOA – Service Oriented Architecture (Web Service Based Applications)

Shopping Site

Amazon.com

Entities (Tables) Associated with Amazon Shopping Site

1. Users Service (Create User, Update details, Delete Users )
2. Products Service ( View Products, Add to Cart )
3. Order Service ( Place the Order )
4. Rating Service ( Ratings, Feedback )
5. Shipping Service (Track Shipment )
6. Payment Service

Single SpringBoot Application with Multiple Entities. – Monolithic Application

Is very hard to manage.

Problem/Error in one Service will affect the entire application.

Load balancing. (Scaling Up/Down)

Monolith Applications are heavy weight

To Resolve the above challenges (MSA- Micro Service Architecture)

Single Spring Boot Application ----- Multiple separate Spring Boot Application ( user microservice, product microservice, order microservice, rating microservice, shipping microservice, payment microservice)

* Many small team (Individually working)
* Easy to maintain the application
* Different Implementation ( Java, Kotlin, Groovy, .Net, Python, Php etc.,)
* Problem in one microservice will not affect other microservice.
* microservice applications are light weight
* Microservices are language and platform independent. (URI)

MicroService --- Microservice Architecture describes a way of designing software applications as suites of independently deployable services.

MicroService Example

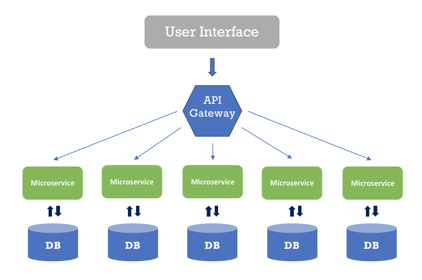
MovieInfo

WeatherInfo

CurrencyConversionRate

GoldRate

BitCoinRate



DebitCard to Witthdraw money from ATM (100 bucks)

ABC Bank DebitCard --🡪 ABC Bank ATM [ Deposit, Withdraw, Print Last 5 Transaction, Update Mobile, Update Email, Update Pin, Check Account Balance] [JAVA]

1. Debit Card Number, Pin number, Operation, Amount

ABC Bank Debit Card -🡪 XYZ Bank ATM [Deposit, WithDraw, Print Last 5 Transaction, Check Account Balance] [.Net Platform]

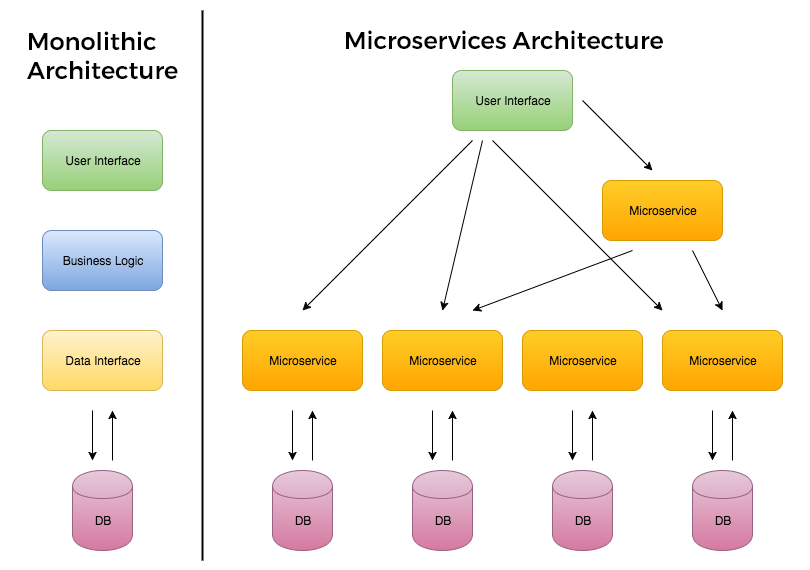
1. Debit Card Number, Pin, Operation, Amount

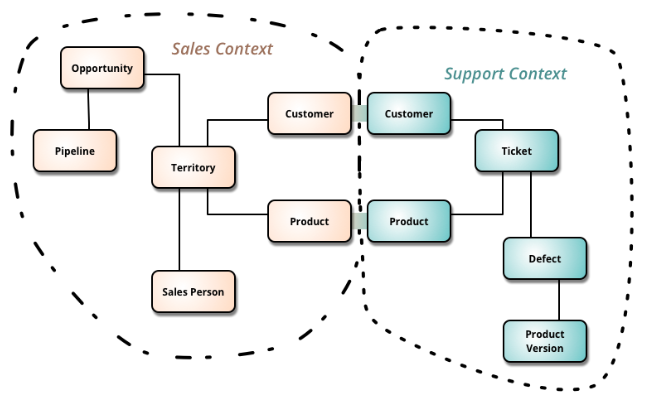
Cons – MicroService

1. Need additional Discovery Service / API Gateway (Eureka Server)

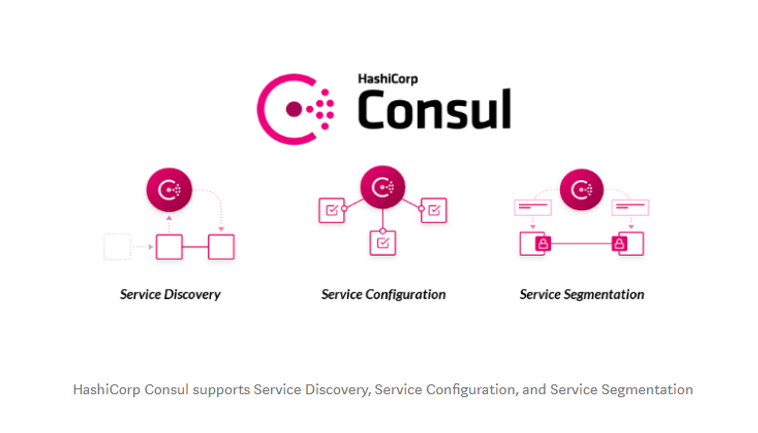
Inter-Service Communication --- RestTemplate

Application.name =





Consule – Is a Open Source Tool to manage MSA



Service Discovery – Listing all the available service ( Netflix-Eureka, HashiCrop-Consul)

List of Public APIs

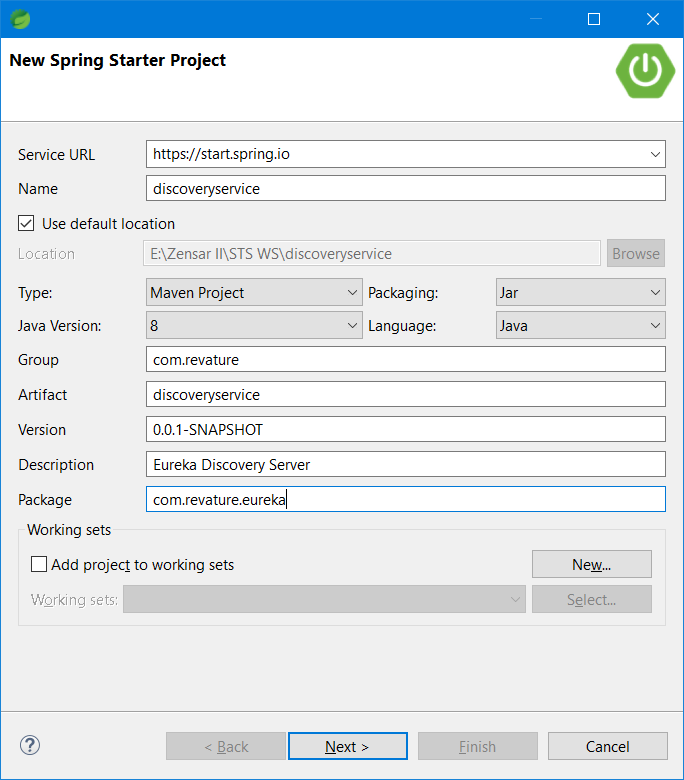
<https://rapidapi.com/collection/list-of-free-apis>

**7a0327f448804fda92d160110220704**

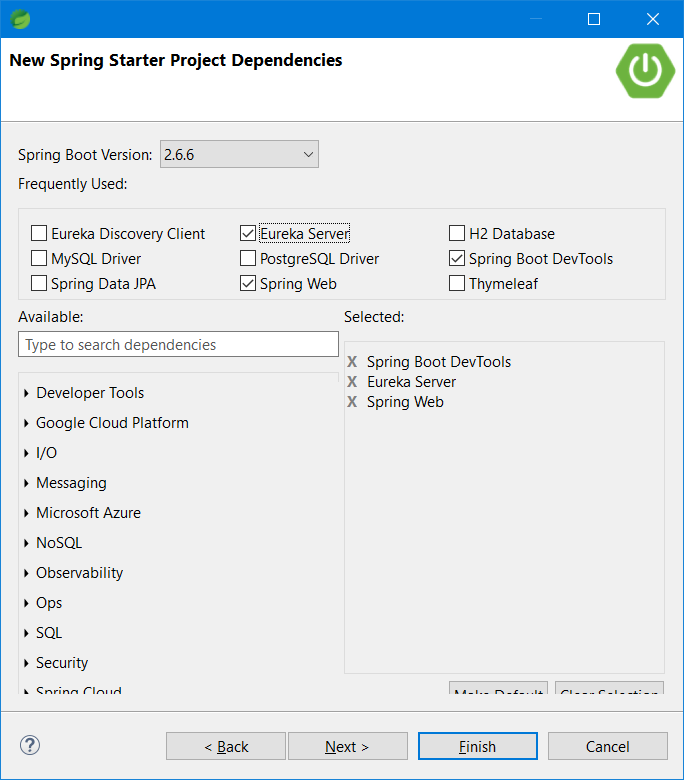
https://api.themoviedb.org/3/movie/550?api\_key=a9a201c957a52859916ab1a66cb3d5e7

Creating MicroService Application

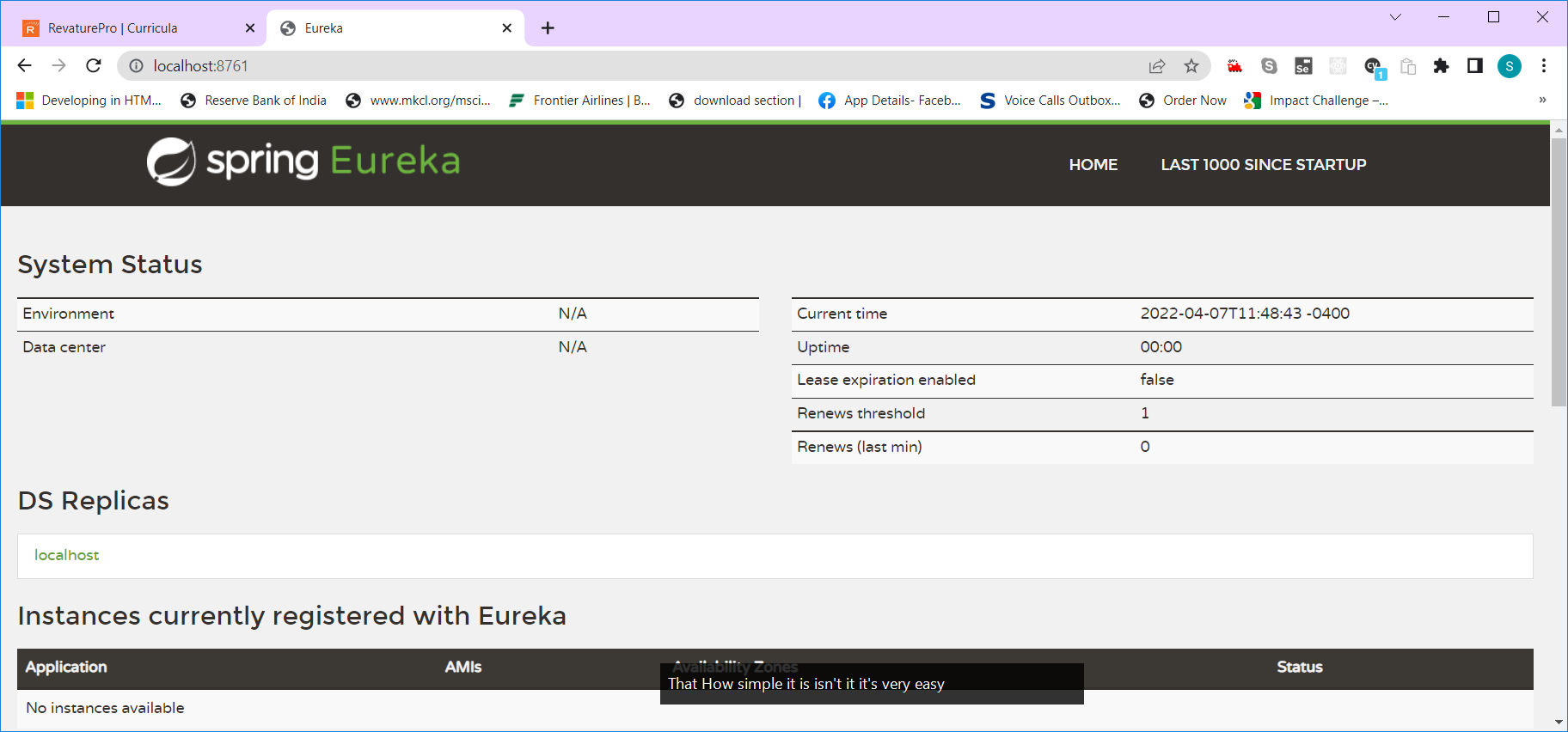
1. Create API Gateway/ Discovery Server [ Eureka Server/ Consul Server]
2. Creating Eureka Server
3. Open STS
4. Create a Spring Starter Project



1. Add the dependency (Eureka Server, Spring Web, Swagger, Spring DevTools, springdoc-apenapi-ui)



1. Update application.properties ( )
2. Add @EnableEurekaServer & @OpenAPIDefinition annotation to the Starter class
3. Start the Application
4. Open localhost:8761



1. Creating Consul Server
2. Creating movie-detail service
   1. Open STS
   2. Create New Spring Starter Project
   3. Add dependencies [spring web, springboot devTools, EurekaClient, Lombok, springdoc-openapi-ui]
   4. Add @EnableEurekaClient & @OpenAPIDefinition annotation to the starter class
   5. Update application.properties ()
3. Creating movie-rating service
   1. Open STS
   2. Create New Spring Starter Project
   3. Add dependencies [spring web, springboot devTools, EurekaClient,Lombok, springdoc-openapi-ui]
   4. Add @EnableEurekaClient & @OpenAPIDefinition annotation to the starter class
   5. Update application.properties ()
4. Creating movie-catalog service
   1. Open STS
   2. Create New Spring Starter Project
   3. Add dependencies [spring web, springboot devTools, EurekaClient, Lombok, springdoc-openapi-ui]
   4. Add @EnableEurekaClient & @OpenAPIDefinition annotation to the starter class
   5. Update application.properties ()