**What is Microservices?**

It is a kind of architectural style which structures an application as a collection of Services

Advantage-

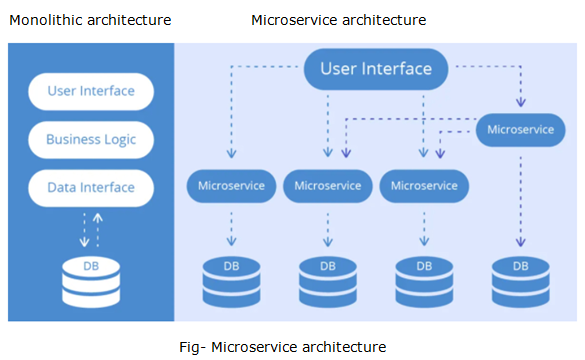
They are loosely coupled.

Independently deployable.

What is the difference between Monolithic application & microservices application?

In Case of Monolithic we consider whole project as a single code-base & hence we deploy the whole app as a single war file.

But when it comes to Microservices application, each of the service is considered as a separate code-base & hence we deploy a separate war file for each code-base (service).



Note

1. We need to create a separate spring boot project for every microservice in the project
2. We need the separate tomcat instances for running each of the application independently.
3. For microservices to communicate with each other , we will use Spring-cloud [EurekaServer].
4. EurekaServer can be considered as a Discovery server.
5. For Microservices to communicate with each other, we have to register them on this EurekaServer.
6. Here each Microservice will be the Client for the EurekaServer.

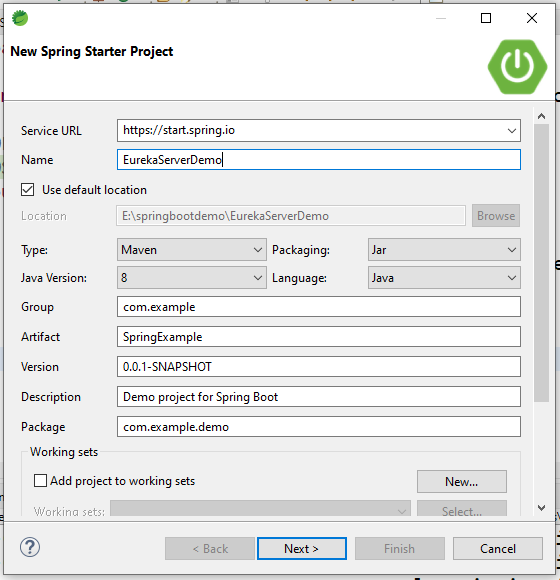
Example-

It is Kind of site which will display the entire Product that End-User has watched.

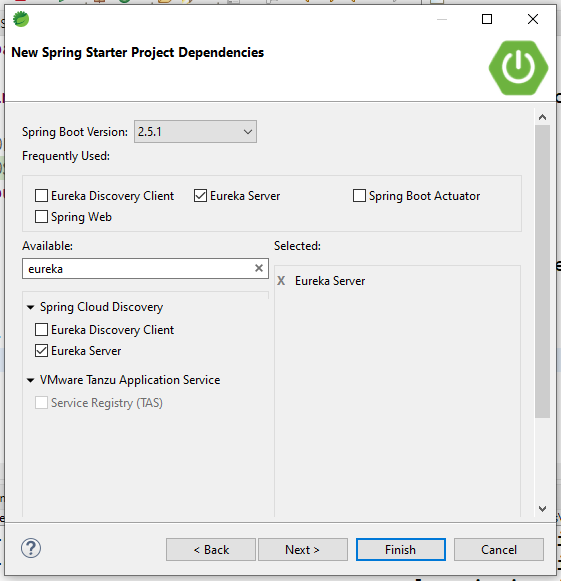
Microservice example-

**Eureka Server**

File->New->Spring Starter Project



Click on Next button



Click on Next button then click on finish button.

Step-1

Go to main method and put @EnableEurekaServer

On above @SpringBootApplication

EurekaServerDemoApplication.Java

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@EnableEurekaServer

@SpringBootApplication

public class EurekaServerDemoApplication {

public static void main(String[] args) {

SpringApplication.run(EurekaServerDemoApplication.class, args);

}

}

Step-2

application.properties

server.port=8761

eureka.client.register-with-eureka=false

Step-3

Run as Spring Boot Application then display below message on screen

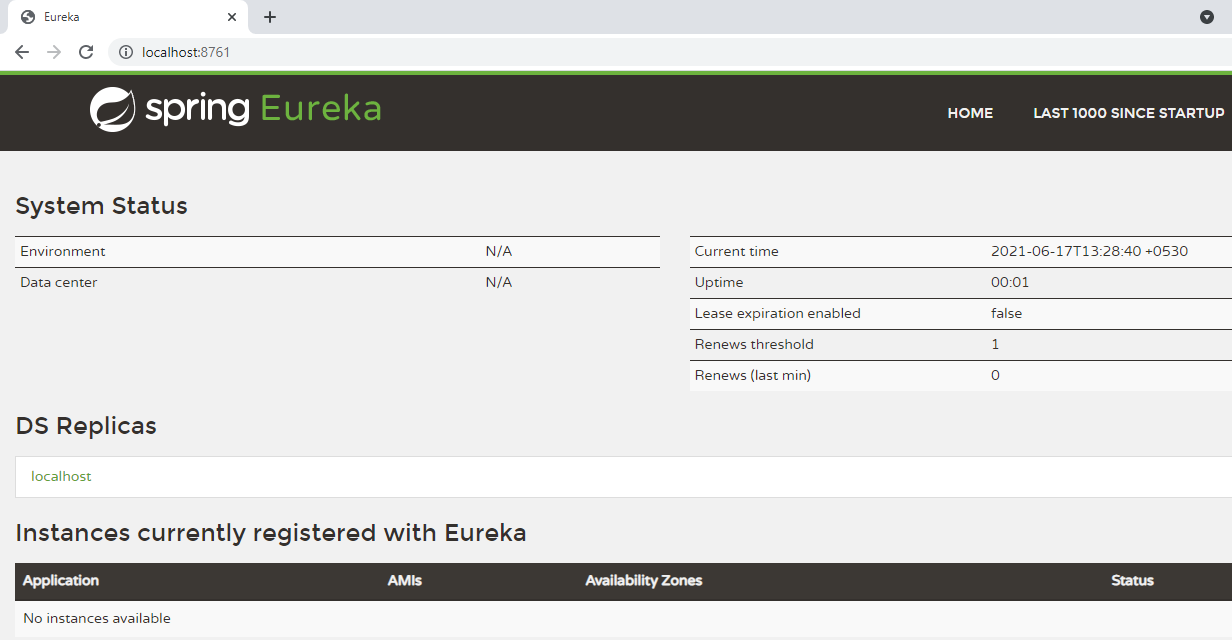
2021-06-17 13:27:48.697 INFO 3336 --- [ main] .s.c.n.e.s.EurekaAutoServiceRegistration : Updating port to 8761

2021-06-17 13:27:49.432 INFO 3336 --- [ Thread-9] e.s.EurekaServerInitializerConfiguration : Started Eureka Server

2021-06-17 13:27:50.585 INFO 3336 --- [ main] c.e.demo.EurekaServerDemoApplication : Started EurekaServerDemoApplication in 37.8 seconds (JVM running for 42.349)

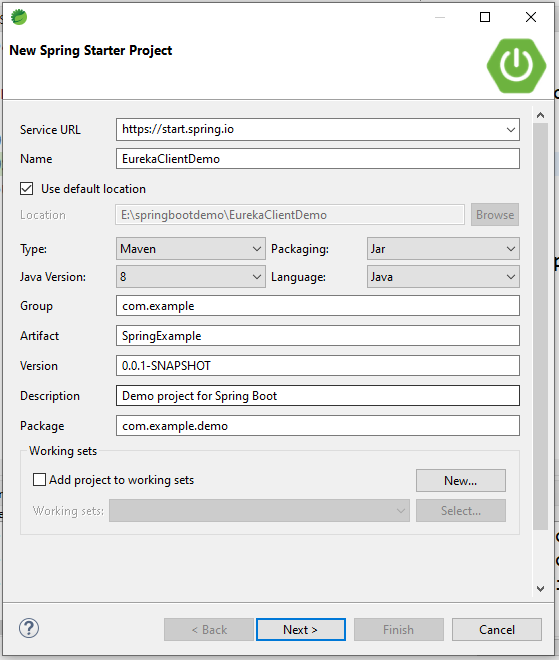
Step-4

Go to browser <http://localhost:8761/> and press enter

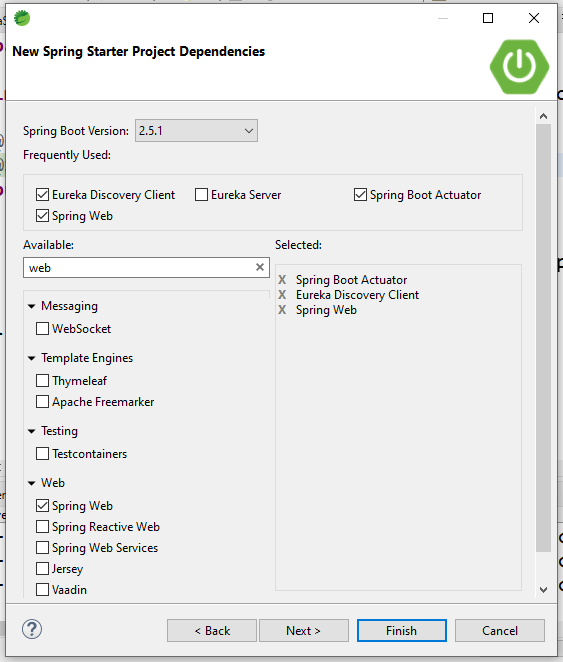


**Eureka Client**

File->New->Spring Starter Project



Click on Next button and add below three dependencies into it.



Click on Next and finish button

Step-1

Go to main method and put @EnableEurekaClient

On above @SpringBootApplication

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

@EnableEurekaClient

@SpringBootApplication

public class ProductTestServiceApplication {

public static void main(String[] args) {

SpringApplication.run(ProductTestServiceApplication.class, args);

}

}

Step-2 application.properties

spring.application.name=producttestservice

server.port=8083

Step-3

Create the Rest controller as per below

**package** com.example.demo;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/product")

**public** **class** ProductController {

@GetMapping("/list")

**public** List<String> getProductList() {

List<String> list = **new** ArrayList<>();

list.add("mobile");

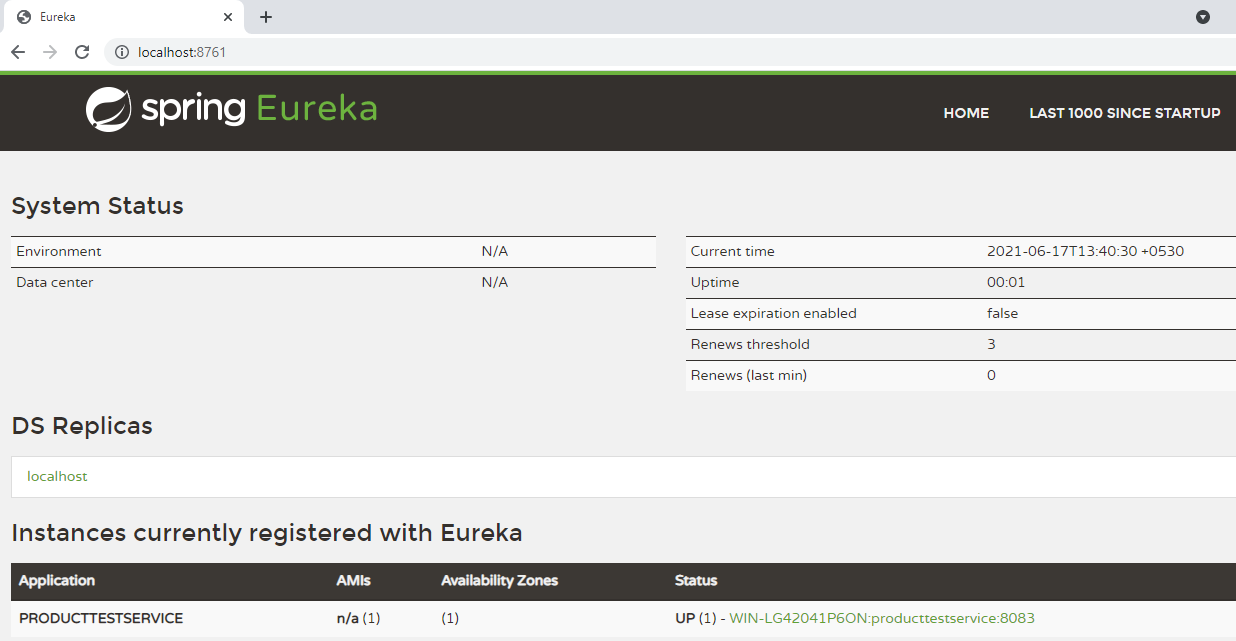
list.add("laptop");

**return** list;

}

}

Step-4 Run as Spring Boot Application and hit the eureka server



Here, service will be displayed on screen