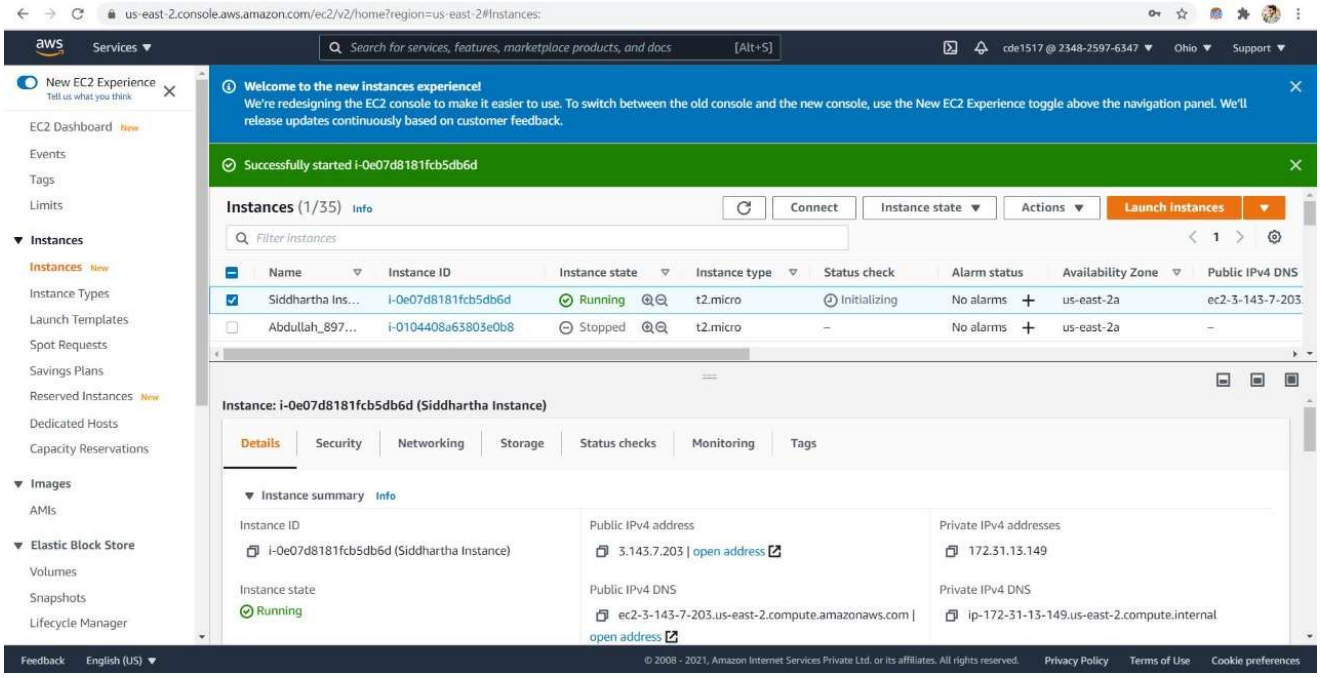
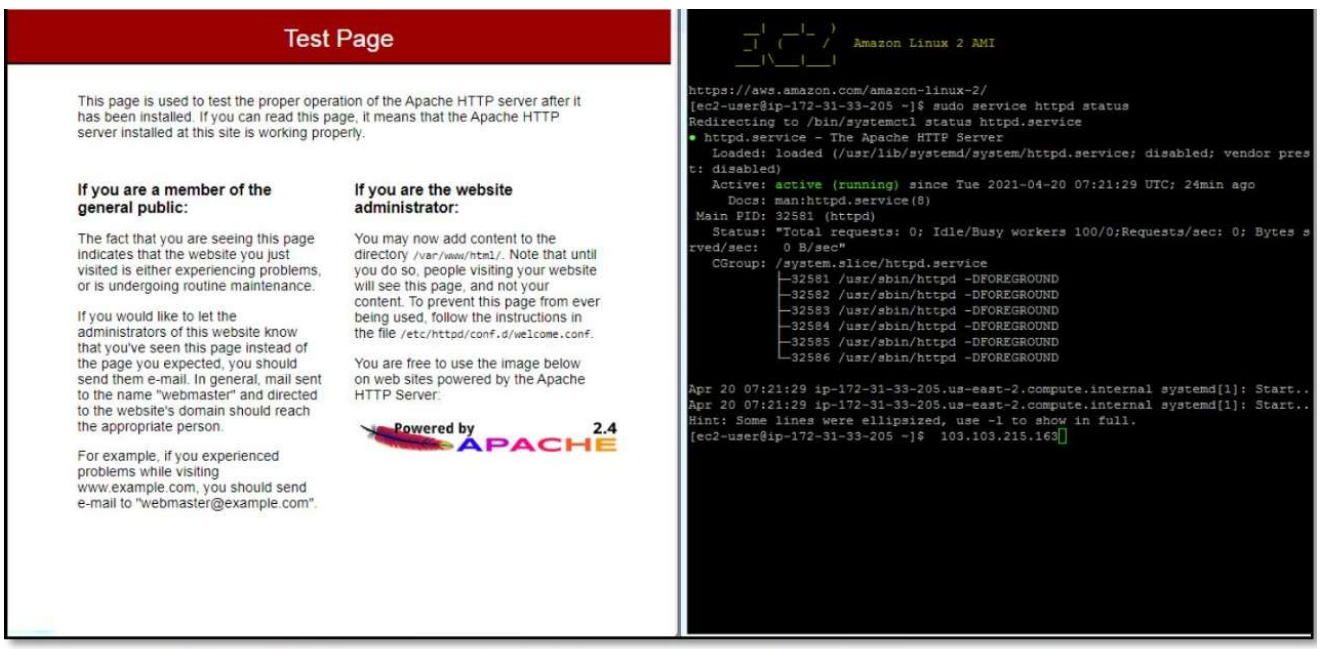
AWS HandsOn

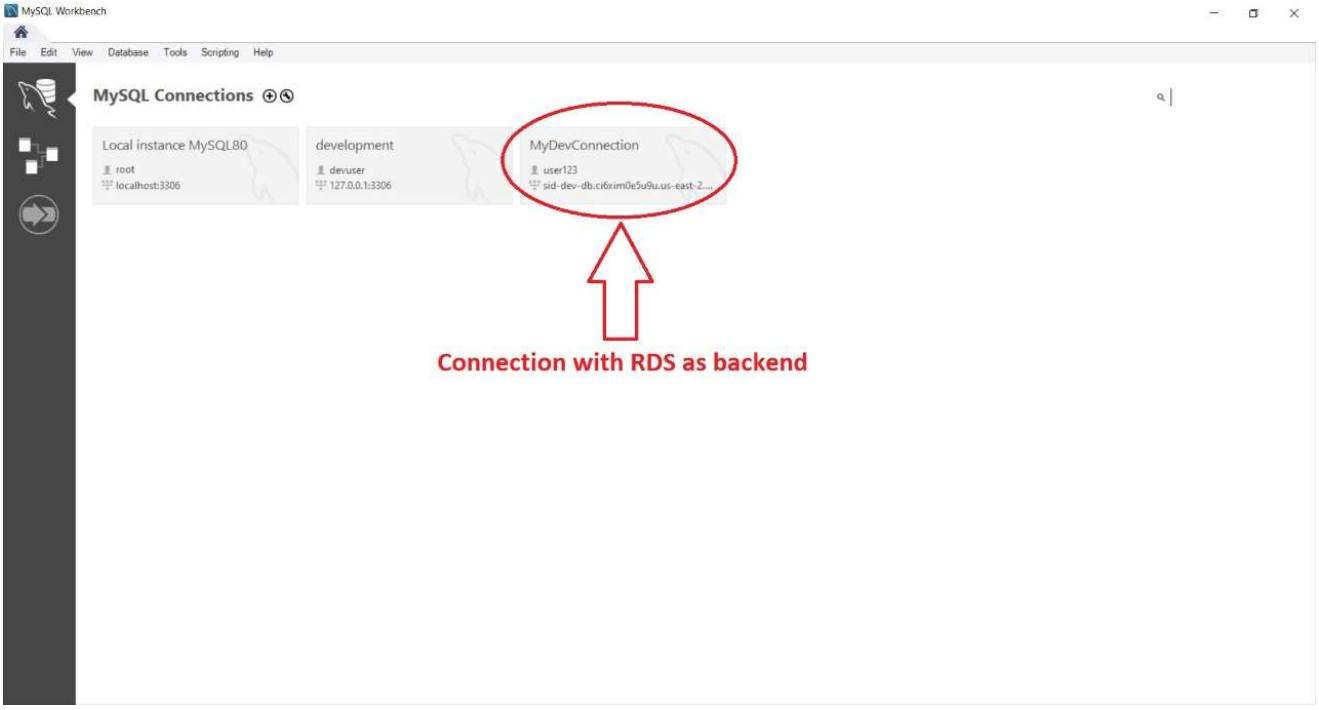
EC2 HandsOn





Type the public IPV4 address on the browser url bar and we should get the above shown screen

RDS HandsOn

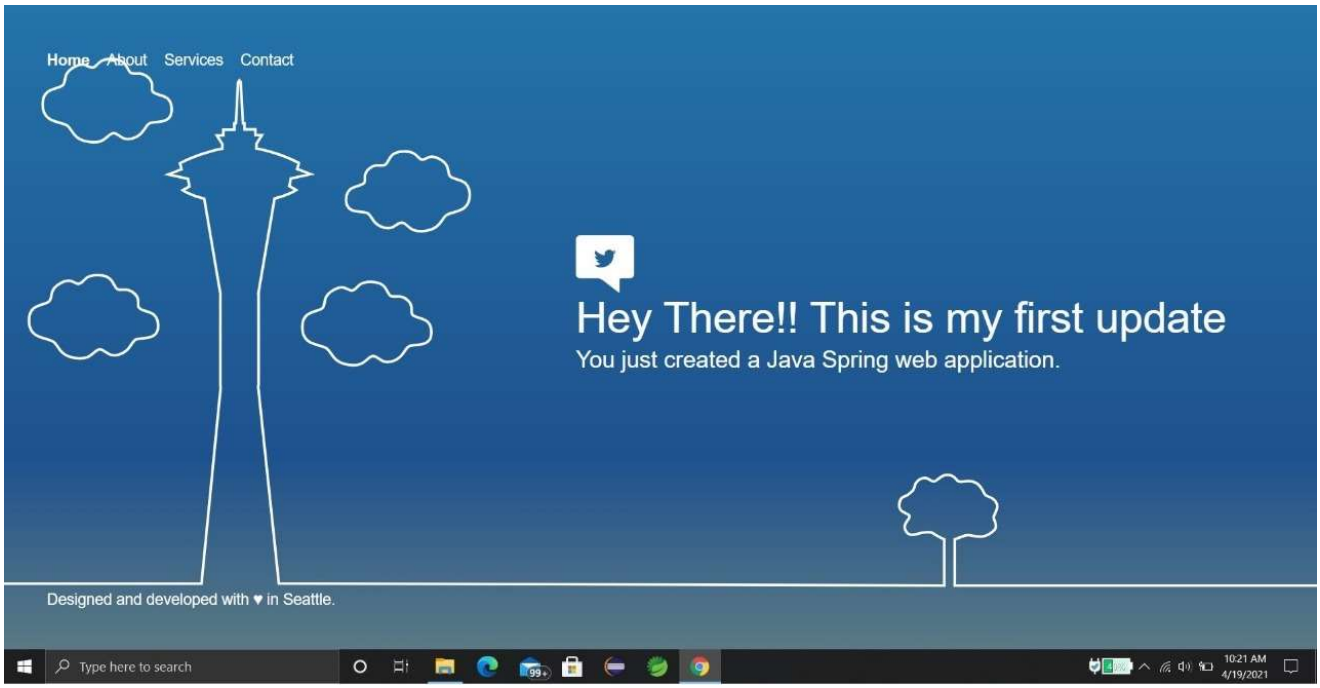


Data in RDS

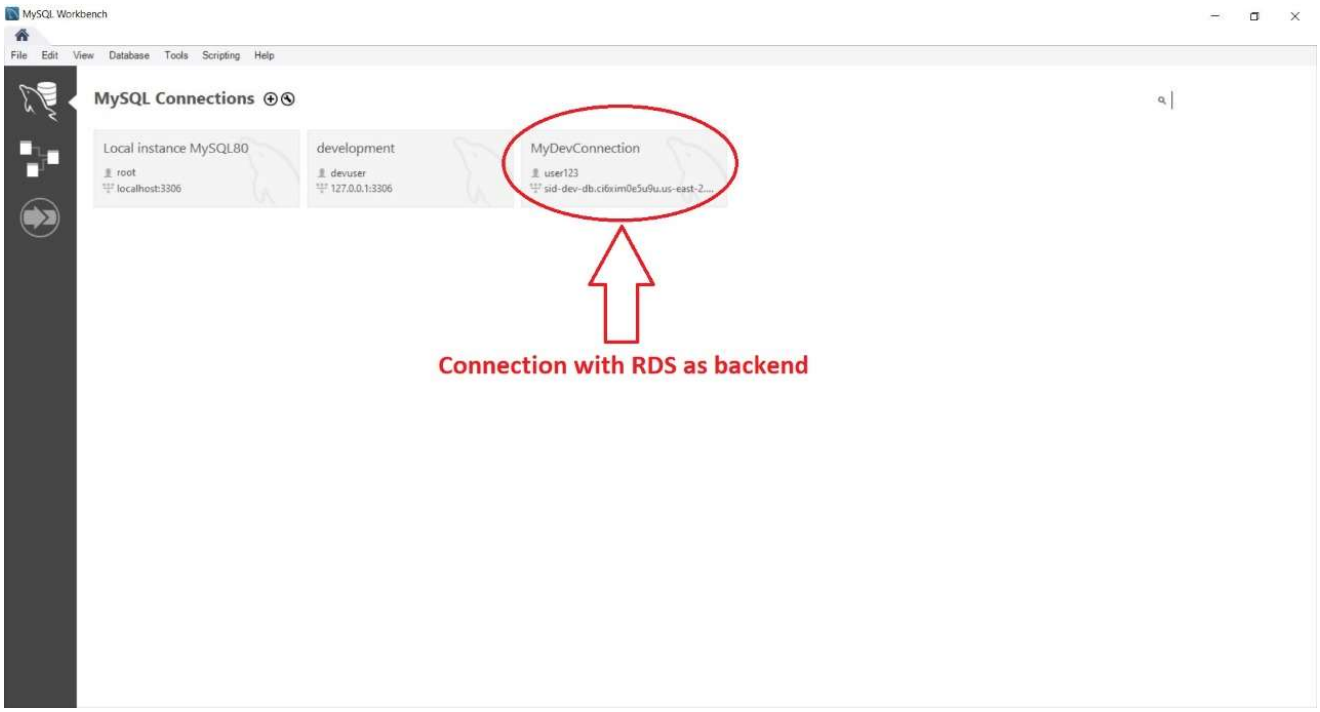
```
create database empdb;
use empdb;
create table Employee(id int primary Key,
                      name varchar(50),
                      gender varchar(50),
                      age int,
                      salary double);
insert into Employee values(1,'Manu','Male',23,34000);
insert into Employee values(2,'Chitra','Female',33,40000);
```

CI/CD HandsOn

Output:-



Spring-Rest-with-RDS-Backend



```
create database empdb;
use empdb;
create table Employee(id int primary Key,
                      name varchar(50),
                      gender varchar(50),
                      age int,
                      salary double);
insert into Employee values(1,'Manu','Male',23,34000);
insert into Employee values(2,'Chitra','Female',33,40000);
```

```
package com.cognizant.controller;

import java.util.List;

@RestController
@RequestMapping("/employees")
public class EmployeeController {

    @Autowired
    EmployeeService employeeService;

    @GetMapping
    public List<Employee> getAllEmployee(){
        return employeeService.getAllEmployee();
    }

    @PostMapping
    public String addEmployee(@RequestBody Employee employee) {
        return employeeService.addEmployee(employee);
    }

    @DeleteMapping("/{id}")
    public String deleteEmployee(@PathVariable int id) {
        return employeeService.deleteEmployee(id);
    }

    @PutMapping
    public String updateEmployee(@RequestBody Employee employee) {
        return employeeService.updateEmployee(employee);
    }

    @GetMapping("/{id}")
    public Employee getEmployeeBy(@PathVariable int id) {
        return employeeService.getEmployeeBy(id);
    }
```

```
2021-04-23 11:57:48.223 DEBUG 7112 --- [nio-8080-exec-3] org.hibernate.SQL                          : select employee0_.id as id1_0_0_, employee0_.age as age
2021-04-23 11:57:48.228 TRACE 7112 --- [nio-8080-exec-3] o.h.type.descriptor.sql.BasicBinder         : binding parameter [1] as [INTEGER] - [4]
2021-04-23 11:57:48.247 TRACE 7112 --- [nio-8080-exec-3] o.h.type.descriptor.sql.BasicExtractor       : extracted value ([age2_0_0_] : [INTEGER]) - [32]
```

We have created a “employee” microservice to test the RDS Database.

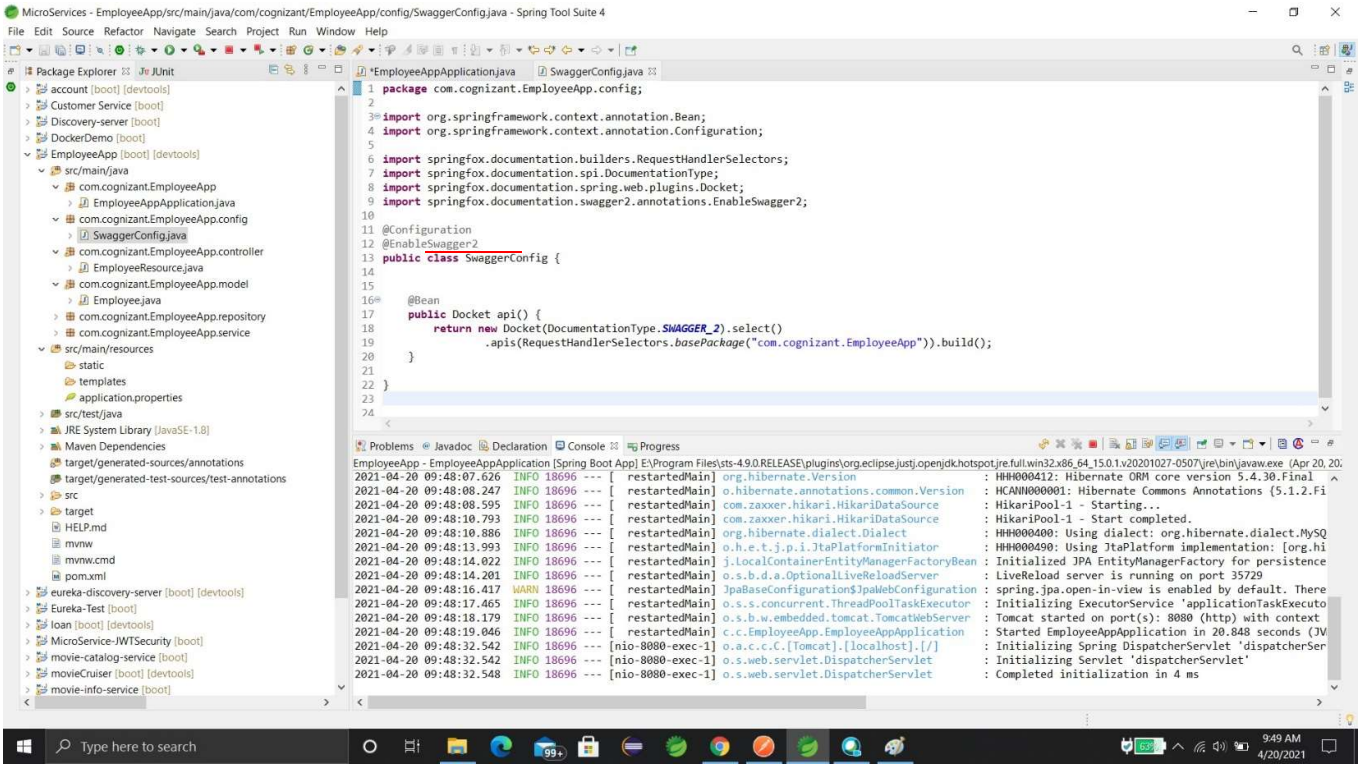
Swagger HandsOn

Step-1:- Add Dependencies

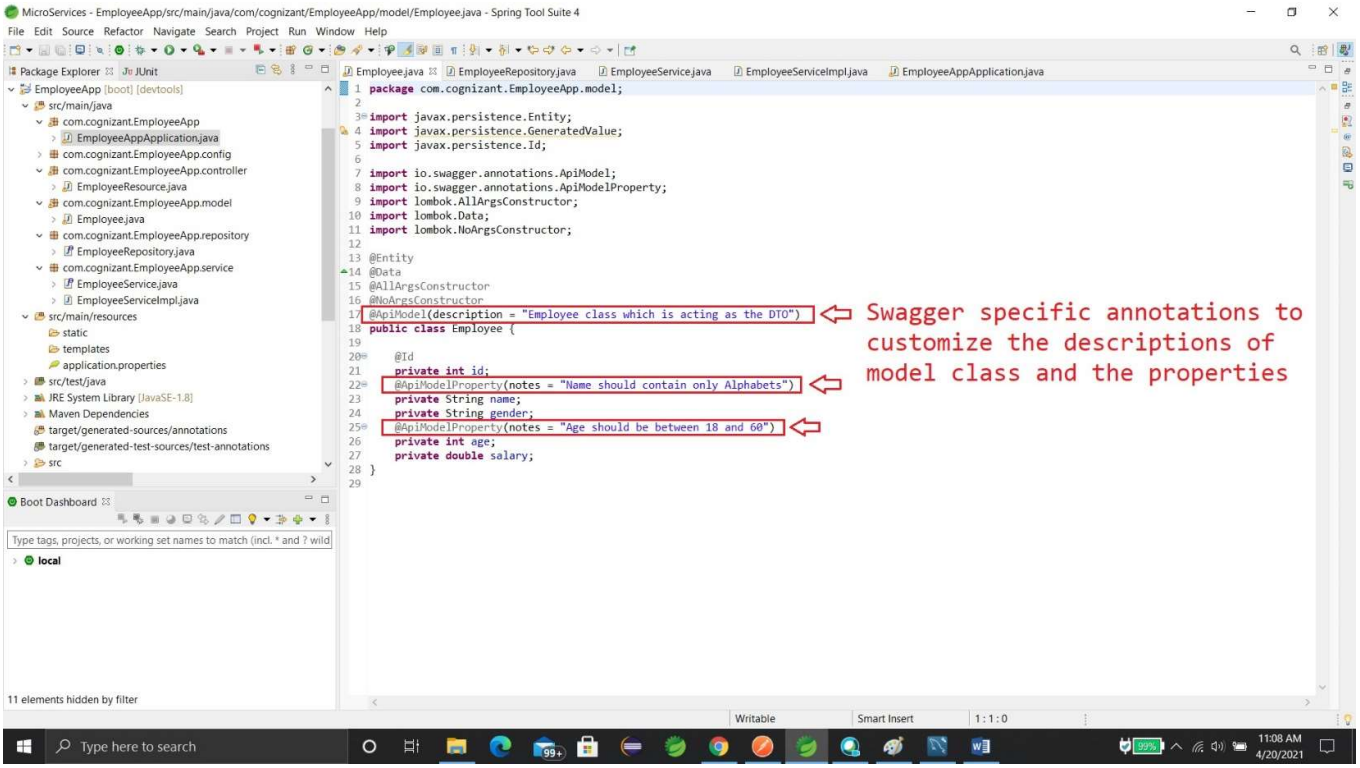
```
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>io.springfox</groupId>
        <artifactId>springfox-swagger-ui</artifactId>
        <version>3.0.0</version>
    </dependency>

    <dependency>
        <groupId>io.springfox</groupId>
        <artifactId>springfox-swagger2</artifactId>
        <version>3.0.0</version>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
```

Swagger Dependencies

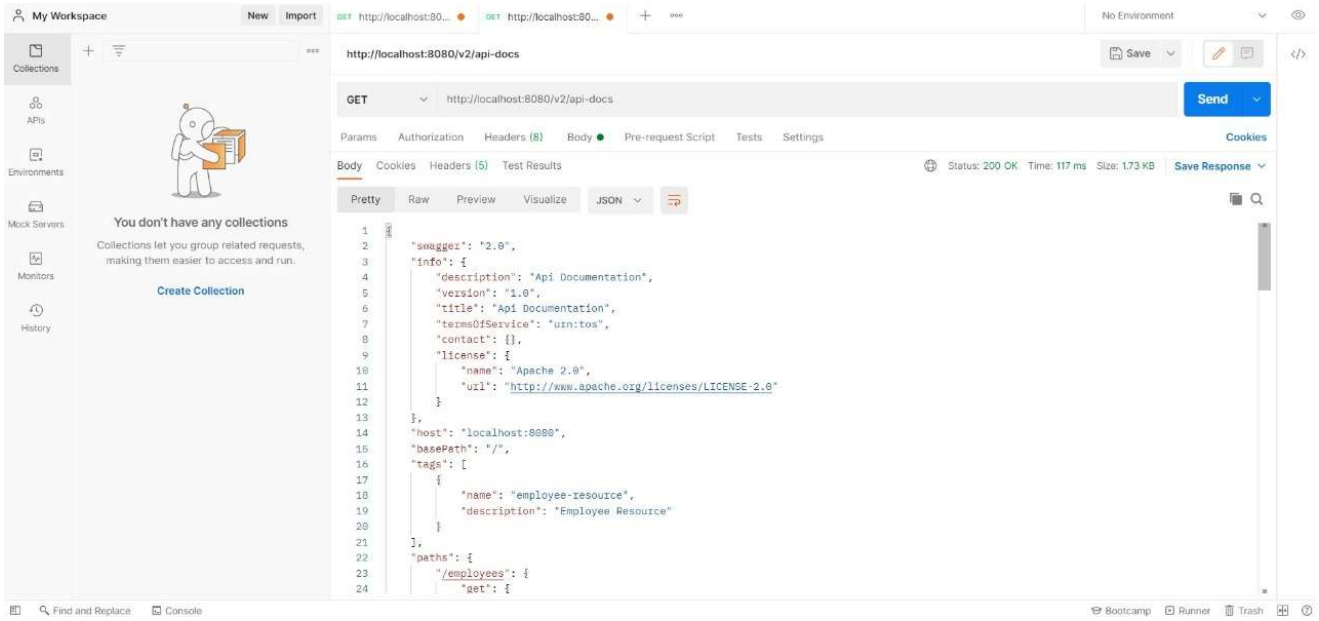
Step 2:- Create a Swagger configuration class



Step 3:- use Swagger specific annotations to customize the descriptions of model class and the properties.

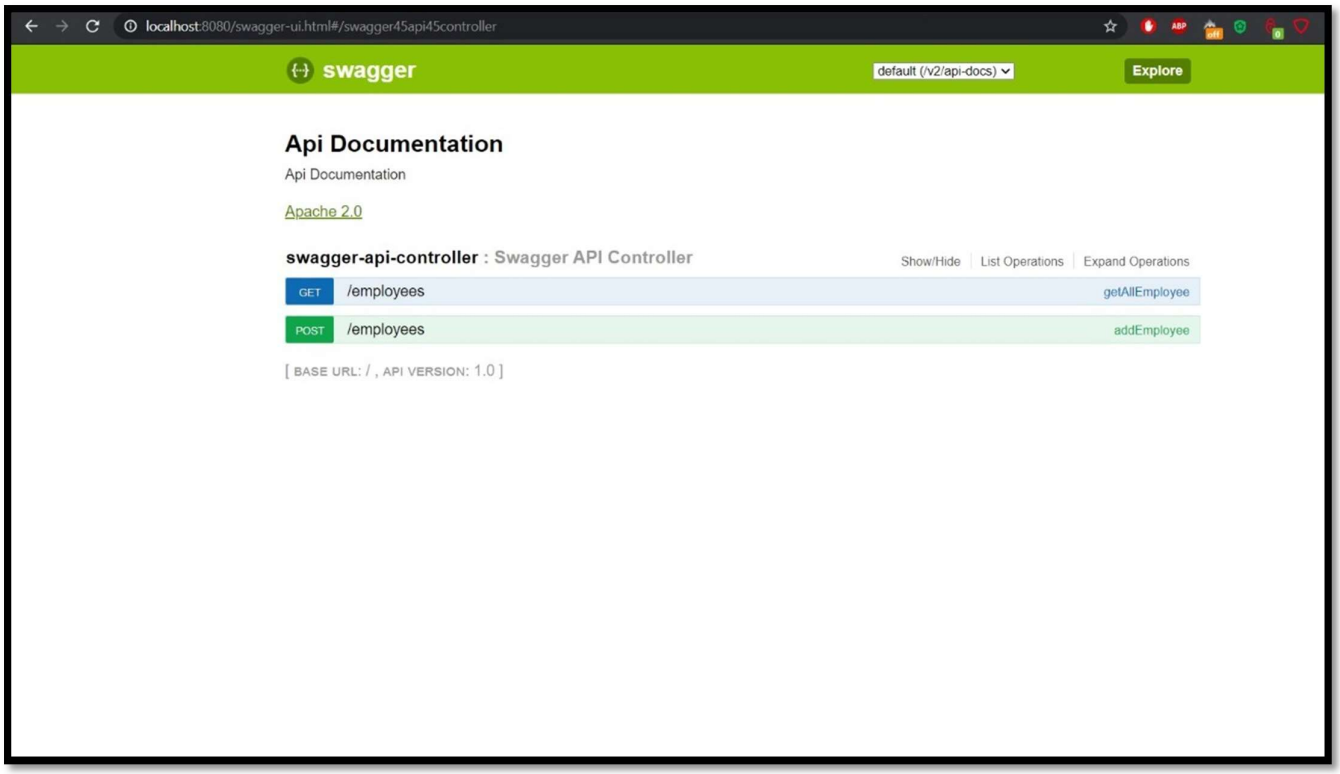


"**localhost:8080/v2/api-docs**" and you can see the complete API documentation of your service.



Now, hit the URL in your web browser and see the Swagger API functionalities.

http://localhost:8080/swagger-ui.html



Spring MVC Client For Spring REST Service

Note:- We have already created a microservice(employee) in our local System. Now, we are just creating another microservice which will consume the rest service of our previous employee microservice.

First we have to add “openfeign” dependency

```
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-openfeign</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
```

Feign Client dependecy

Note:-

The *value* argument passed in the *@FeignClient* annotation is a mandatory, arbitrary client name, while with the *url* argument, we specify the API base URL.

Furthermore, since this interface is a Feign client, we can use the Spring Web annotations to declare the APIs that we want to reach out to.

EmployeeRepository.java | EmployeeFeign.java | MvcController.java | application.properties

```java
package com.cts.feign;

import java.util.List;

@FeignClient(name = "emp-spring", url = "${feign.url-employee-spring}")
public interface EmployeeFeign {

    @GetMapping
    public List<Employee> getAllEmployee();

    @GetMapping("/{id}")
    public Employee getEmployeeByID(@PathVariable int id);

    @DeleteMapping("/{id}")
    public String deleteEmployee(@PathVariable int id);

    @PostMapping
    public String addEmployee(@RequestBody Employee employee);

    @PutMapping
    public String updateEmployee(@RequestBody Employee employee);
}
```

Declaring this interface as Feign Client

EmployeeRepository.java | EmployeeFeign.java | MvcController.java | application.properties | EmployeeMvcApplication.java

```java
package com.cts;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
@EnableFeignClients
public class EmployeeMvcApplication {

    public static void main(String[] args) {
        SpringApplication.run(EmployeeMvcApplication.class, args);
    }

}
```

With this annotation, we enable component scanning for interfaces that declare they are Feign clients.

With this annotation, we enable component scanning for interfaces that declare they are Feign clients.

Output:-

output:-