Microservice -Service Interaction

Steps for Customer Service Interact with Order Service

Step:1 We can interact the Customer service with Order service by using Rest Template

Rest Template is used to create applications that consume RESTful Web Services. You can use the exchange() method to consume the web services for all HTTP methods.

Add the below code in the application.java file

```
@Bean
	public RestTemplate restTemplate() {
		return new RestTemplate();
	}
```

Step:2 In the Controller file add the below code

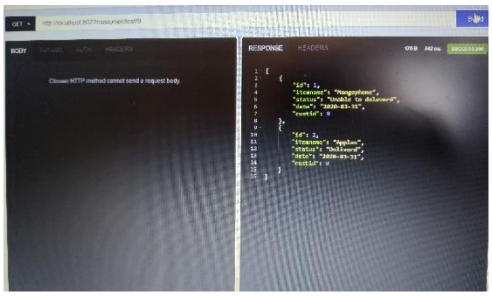
```
@Autowired
	private RestTemplate restTemplate;


@GetMapping("rest/{customerId}")
	public List<Orders_Entity> getOrdersFromOrderEntity(@PathVariable
int customerId){
		Orders_Entity obj[]
=restTemplate.getForObject("http://localhost:8021/Orders/"+customerId,
Orders_Entity[].class);
		return Arrays.asList(obj);
	}
```

Use the Autowired to inhert the Rest Template Properties.

The Above code We are interacting the Order service from the Customer service we are using the resttemplate.get forObject method to get the details from the Orders service.

For that we have given the Link of Order service(http://localhost:8021/Orders/"+customerId) and the Orders Entity Class to get the Objects and return in the List of Object.

Step:3 Strat the Order Service And Customer Service and run the Url we Get the Response from the Customer Service.



Chosen HTTP method cannot send a request body.

```
[
    {
        "id": 1,
        "itemname": "Mangophone",
        "status": "Unable to delevered",
        "date": "2020-03-31",
        "custid": 9
    },
    {
        "id": 2,
        "itemname": "Apples",
        "status": "Deliverd",
        "date": "2020-03-31",
        "custid": 9
    }
]
```

Steps to Implement the Circuit Breaker

We can implement the circuit-breaker in the spring boot by using Hystrix annotation. The main idea is to stop cascading failures by failing fast and recover as soon as possible.

Step:1 firstly we have to add the Hystrix and Spring-boot-actuator dependencies in the porm.xml file to add the properties to the microservices

The dependency file is given below

Spring-boot-actuator dependency

```
<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-actuator</artifactId>
		</dependency>
```

Hystrix Dependency

```
<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
			<version>2.2.7.RELEASE</version>
		</dependency>
```

Step:2 Enable the @Enable Hystrix Command in the application.java file so that we can use the hystrix properties.

Step:3 Add the @Enable Hystrix Command annotation and fall back method to which method we are going to implement the circuit-Breaker.the code is given below.

```
@GetMapping("rest/{customerId}")
```

```
	@HystrixCommand(fallbackMethod = 
"getOrdersFromOrderEntityFallBackMethod")
	public List<Orders_Entity> getOrdersFromOrderEntity(@PathVariable 
int customerId){
		Orders_Entity obj[] 
=restTemplate.getForObject("http://localhost:8021/Orders/"+customerId, 
Orders_Entity[].class);
		return Arrays.asList(obj);
	}
```

Step:4 write the fallback method to give the response in case of failure of request.

```
public List<Orders_Entity> 
getOrdersFromOrderEntityFallBackMethod(@PathVariable int customerId){
		List<Orders_Entity> list = new ArrayList<>();
		return list;
	}
```

Containerization- Docker Execution

Build and deploy using docker container

- **Add Docker Configuration file**
  Now create docker file in the root directory and add the below lines of code
  FROM openjdk:8-jdk-alpine
  VOLUME/tmp
  ADD target/demo-0.0.1snaoshot.jar app.jar// add the jar file to the docker image
  ENTRYPOINT [java" ,"jar","/app.jar]

1. The above line code is to add in the docker file.
2. Those lines indicates that we are building docker image from the base of jdk8.
3. Tmp is used to create a work setup(Generally for spring boot application).
4. And adding jar file to docker image.

- **Add Maven Docker plugins**
  Add to maven plugins in the porm.xml file so that we can use docker related maven commands while creating the instance those dockerfile-maven-plugins and maven-dependency-plugin.

- **Creating Docker Image**
  now run the in the docker file location as docker command as docker build file name.

  we can check the image is created or not by using command docker image ls.

- **Deploy &Run the docker image**
  We can run the docker image inside the docker container we use the below command.
  Docker run -p8080:8081 -t demo –namedemo-docker-image

  Now the -p8080:8081 is says that expose port 8080 for internal port 8081 and the application running port is 8081 inside the docker image and we use 8080 for outside the docker container.

- **Stop Docker Container**
  We stop the container by using the command docker stop<name>.

  We can restart the container by using the command docker container start <container id>.

- **Get status of available pods**

  We use this command Kubectl get pods –all.

Containerization -Docker Concepts

Docker File :- its analogous recipe it has specifying ingredients required for an application.

The Expose Command inform the Docker that the container listen on the specified port network port at Runtime. Expose does not make the port of the container accessible to the host.

- The Expose Instruction exposes the specified port and make it available only for inter-container communication.

Example

- Let think we have two containers a nodejs application and redis server our node app needs to communicate with redis for several times.
- For the node app to communicate with redis server we need to expose the port. For example the port is 6337.
- So Expose command make possible to connect the nodejs with redis server.
- To communicate with each other it is important to both the containers are running in the same network. In this way Expose command helps to maintain inter-container connection.
- One benefit of Expose command is we can know easily the whether ur are running the multi-container by simply seeing the docker file.

Difference Between Virtualization and Containerization

Containerization	**Virtualization**
Containers is to encapsulate an application and its dependencies within its own environment. they are using same operating system.	Virtualization provides existence of multiple smaller server to run on the physical environment. Basically the parent server provides the resource and its own operating system. its runs by software called hypervisor.
Since containers are using same operating system. Resource are not wasted and containers allow much quicker deployment of application.	In virtualization every server has own resource and operating system assigned by parent sever so resource are not efficiently used in virtualization.
Where coming to security the containers has less secure then virtualization since containers are using isolated data and applications.	In the security the virtualization keeps separated and isolated with each other so every application ha own security protocols.
Containers are used for short-term application	Virtual machine are for long term application because they use virtualized environment that is more robust.

Where coming to speed containers very fast because the operating system is already running so application run without any delay.	In virtual machines we have start whole operating system it need boot process to be done so it takes much time compare to containers.
Since containers are size are very small it is very easy to migrate and move.	Virtual machine has to copy whole operating system, including configuration files so it takes more size to copy.
Containers themselves don’t move instead an orchestrator can start or stop containers on cluster nodes to manage the load.	Virtual machine load balancing moves running server to other serer in failure of cluster.