===============================================================

Steps to develop Service Registry Application (Eureka Server)

===============================================================

1) Create SpringBoot application with below dependency

- Eureka Server (spring-cloud-starter-netflix-eureka-server)

- devtools

2) Configure @EnableEurekaServer annotation in boot start class

3) Configure below properties in application.yml file

spring:

application:

name: 01\_Service\_Registry

server:

port: 8761

eureka:

client:

register-with-eureka: false

Note-1: If "Service-Registry" project port is 8761 then clients can discover service-registry and will register automatically with service-registry.

Note-2 : If service-registry project running on any other port number then we have to register clients with service-registry manually.

4) Once application started we can access Eureka Dashboard using below URL

URL : http://localhost:8761/

======================================

Steps to develop Spring Admin-Server

======================================

1) Create Boot application with "admin-server" dependency

(select it while creating the project)

2) Configure @EnableAdminServer annotation at start class

3) Change Port Number (Optional)

spring:

application:

name: 02\_Admin\_Server

server:

port: 1111

4) Run the boot application

5) Access application URL in browser (We can see Admin Server UI)

URL : http://localhost:1111/

======================================

Steps to work with Zipkin Server

======================================

1) Download Zipin Jar file

URL : https://zipkin.io/pages/quickstart.html

2) Run zipkin jar file

$ java -jar <jar-name>

3) Zipkin Server Runs on Port Number 9411

4) Access zipkin server dashboard

URL : http://localhost:9411/

#################################

Steps to develop WELCOME-API

#################################

1) Create Spring Boot application with below dependencies

- eureka-discovery-client

- admin-client

- zipkin

- starter-web

- devtools

- actuator

2) Configure @EnableDiscoveryClient annotation at boot start class.

3) Create RestController with required method

@RestController

public class WelcomeRestController {

@GetMapping("/welcome")

public String getWelcomeMsg() {

String msg = "Welcome To Ashok IT..!!";

return msg;

}

}

4) Configure below properties in application.yml file

-----------------------------------------------------

spring:

application:

name: 04\_Welcome\_Service

boot:

admin:

client:

url: http://localhost:1111/

server:

port: 8081

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka/

management:

endpoints:

web:

exposure:

include: '\*'

--------------------------------------------------------

5) Run the application and check in Eureka Dashboard (It should display in eureka dashboard)

6) Check Admin Server Dashboard (It should display) (we can access application details from here)

Ex: Beans, loggers, heap dump, thred dump, metrics, mappings etc...

7) Send Request to REST API method

8) Check Zipkin Server UI and click on Run Query button

(it will display trace-id with details)

#################################

Steps to develop GREET-API

#################################

1) Create Spring Boot application with below dependencies

- eureka-discovery-client

- admin-client

- zipkin

- starter-web

- devtools

- actuator

- openfeign

2) Configure @EnableDiscoveryClient annotation at boot start class.

3) Create RestController with required method

@RestController

public class GreetRestController {

@GetMapping("/greet")

public String getGreetMsg() {

String msg = "Good Morning";

return msg;

}

}

4) Configure below properties in application.yml file

-----------------------------------------------------

spring:

application:

name: 05\_Greet\_Service

boot:

admin:

client:

url: http://localhost:1111/

server:

port: 9091

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka/

management:

endpoints:

web:

exposure:

include: '\*'

--------------------------------------------------------

5) Run the application and check in Eureka Dashboard (It should display in eureka dashboard)

6) Check Admin Server Dashboard (It should display) (we can access application details from here)

Ex: Beans, loggers, heap dump, thred dump, metrics, mappings etc...

7) Send Request to REST API method

8) Check Zipkin Server UI and click on Run Query button

(it will display trace-id with details)

==============================

Interservice communication

==============================

=> Add @EnableFeignClients dependency in GREET-API boot start class

=> Create FeignClient interface like below

@FeignClient(name = "WELCOME-API")

public interface WelcomeApiClient {

@GetMapping("/welcome")

public String invokeWelcomeMsg();

}

=> Inject feign client into GreetRestController like below

@RestController

public class GreetRestController {

@Autowired

private WelcomeApiClient welcomeClient;

@GetMapping("/greet")

public String getGreetMsg() {

String welcomeMsg = welcomeClient.invokeWelcomeMsg();

String greetMsg = "Good Morning, ";

return greetMsg.concat(welcomeMsg);

}

}

=> Run the applications and access greet-api method

(It should give combined response)

==================

Load Balancing

==================

=> If we run our application in one server then burden will be increased on that server.

1) Single Server should handle all the load

2) Burden on server

3) Response delay

4) Server can crash

5) Single Point of failure

=> To overcome above problems we will run our application in multiple servers so that we can distribute the requests to multiple servers.

=> Load Balancer is used to distribute requests to multiple servers.

=> We have below advantages with load balancer.

1) Less burden on server

2) Quick Responses to clients

3) No Single point of failure

===============================

Load Balancing For Welcome API

===============================

1) Remove server port number from welcome api yml file

2) Make changes in rest controller to send port number in response.

@RestController

public class WelcomeRestController {

@Autowired

private Environment env;

@GetMapping("/welcome")

public String getWelcomeMsg() {

String port = env.getProperty("server.port");

String msg = "Welcome To Ashok IT..!! (" + port + ")";

return msg;

}

}

3) Right click => Run as => run configuration => select welcome-api => VM Arguments => -Dserver.port=8081 and apply and run it.

4) Right click => Run as => run configuration => select welcome-api => VM Arguments => -Dserver.port=8082 and apply and run it.

5) Right click => Run as => run configuration => select welcome-api => VM Arguments => -Dserver.port=8083 and apply and run it.

Note: With this our api will run in 3 servers with 3 diff port numbers.

6) Check Eureka Dashboard and observer 3 instances available for welcome-service or not.

7) Start Greet-Service and send request to Greet-Service and check Interservice communication.

========================================

Working with Spring Cloud API Gateway

========================================

1) Create Spring boot application with below dependencies

-> eureka-client

-> cloud-reactive-gateway

-> devtools

2) Configure @EnableDiscoveryClient annotation at boot start class

3) Configure API Gateway Routings in application.yml file like below

spring:

application:

name: API-Gateway

cloud:

gateway:

routes:

- id: api-1

uri: lb://WELCOME-SERVICE

predicates:

- Path=/welcome

- id: api-2

uri: lb://GREET-SERVICE

predicates:

- Path=/greet

server:

port: 3333

4) Create Filter to validate incoming request

if request contains below header then it is valid request so process it.

Secret=ashokit@123

if above header is not present then it is invalid request, don't process it.

@Component

public class MyFilter implements GlobalFilter {

@Override

public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {

System.out.println(" filter() - executed..... ");

// validate request

ServerHttpRequest request = exchange.getRequest();

HttpHeaders headers = request.getHeaders();

Set<String> keySet = headers.keySet();

if(!keySet.contains("Secret")) {

throw new RuntimeException("Invalid Request");

}

List<String> list = headers.get("Secret");

if(!list.get(0).equals("ashokit@123")) {

throw new RuntimeException("Invalid Request");

}

return chain.filter(exchange);

}

}