Microservices Guide

Development and Deployment

Contents

[Step 1: Create Eureka Server - Service 3](#_Toc510626104)

[Step 2: Create Zuul Service 4](#_Toc510626105)

[Step 3: Create Business Services 8](#_Toc510626106)

[Step 4: Create Turbine Service 17](#_Toc510626107)

[Step 5: Deployment on Docker Container 19](#_Toc510626108)

[Build Jar files for each spring boot application 19](#_Toc510626109)

[Create Dockerfile 19](#_Toc510626110)

[Build Image 20](#_Toc510626111)

[Run Container 20](#_Toc510626112)

[Important Notes: 20](#_Toc510626113)

[Current URL Configurations: 20](#_Toc510626114)

[Docker Executed Commands: 20](#_Toc510626115)

# Step 1: Create Eureka Server - Service

***Dependency: pom.xml***

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>kpit.poc</groupId>  <artifactId>ms-eureka-server</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>  <name>ms-eureka-server</name>  <description>Eureka server for POC</description>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.0.RELEASE</version>  <relativePath/> <!-- lookup parent from repository -->  </parent>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  <java.version>1.8</java.version>  <spring-cloud.version>Finchley.M8</spring-cloud.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  </dependencies>  <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>${spring-cloud.version}</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies>  </dependencyManagement>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  <repositories>  <repository>  <id>spring-milestones</id>  <name>Spring Milestones</name>  <url>https://repo.spring.io/milestone</url>  <snapshots>  <enabled>false</enabled>  </snapshots>  </repository>  </repositories>  </project> |

***Java: Application Class***

|  |
| --- |
| @SpringBootApplication  @EnableEurekaServer  public class EurekaServerApplication {  public static void main(String[] args) {  SpringApplication.run(EurekaServerApplication.class, args);  }  } |

***Properties: application.yml***

|  |
| --- |
| server:  address: 0.0.0.0  port: 9091  error:  whitelabel:  enabled: false    eureka:  client:  register-with-eureka: false  fetch-registry: false  service-url:  defaultZone: ${EUREKA\_URI:http://<HOST\_IP>:<PORT>/eureka}  instance:  prefer-ip-address: true |

***Create jar file***

Execute ‘*mvn clean install*’ on root directory of the project.

# Step 2: Create Zuul Service

***Dependency: pom.xml***

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>kpit.poc</groupId>  <artifactId>ms-zuul-service</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>  <name>ms-zuul-service</name>  <description>Zuul service for POC</description>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.0.RELEASE</version>  <relativePath /> <!-- lookup parent from repository -->  </parent>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  <java.version>1.8</java.version>  <spring-cloud.version>Finchley.M8</spring-cloud.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-zuul</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.retry</groupId>  <artifactId>spring-retry</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  </dependencies>  <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>${spring-cloud.version}</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies>  </dependencyManagement>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  <repositories>  <repository>  <id>spring-milestones</id>  <name>Spring Milestones</name>  <url>https://repo.spring.io/milestone</url>  <snapshots>  <enabled>false</enabled>  </snapshots>  </repository>  </repositories>  </project> |

***Java: Application Class***

|  |
| --- |
| @SpringBootApplication  @EnableDiscoveryClient  @EnableZuulProxy  @EnableRetry  public class ZuulServiceApplication {  public static void main(String[] args) {  SpringApplication.run(ZuulServiceApplication.class, args);  }    @Bean  public PostFilter postFilter() {  return new PostFilter();  }  @Bean  public PreFilter preFilter() {  return new PreFilter();  }  @Bean  public RouteFilter routeFilter() {  return new RouteFilter();  }  @Bean  public ErrorFilter errorFilter() {  return new ErrorFilter();  }  } |

***Java: Filter Class***

Do same for all four filters - Pre, Post, Route, and Error

(refer below – change filterType to ‘post’, ’route’, ’error’, and method run)

|  |
| --- |
| public class PreFilter extends ZuulFilter {  @Override  public Object run() {  System.out.println(“Pre Filter”);  RequestContext ctx = RequestContext.getCurrentContext();  HttpServletRequest request = ctx.getRequest();  ctx.addZuulRequestHeader(“Authorization”, request.getHeader(“Authorization”));  System.out.println(“Request Method : “ + request.getMethod() + “ Request URL : “ + request.getRequestURL().toString());  return null;  }  @Override  public boolean shouldFilter() {  return true;  }  @Override  public int filterOrder() {  return 0;  }  @Override  public String filterType() {  return “pre”;  }  } |

Authorization is optional – depending on services.

***Properties: application.yml***

|  |
| --- |
| spring:  application:  name: zuul-service  server:  address: 0.0.0.0  port: 8084  error:  whitelabel:  enabled: false  eureka:  client:  service-url:  defaultZone: ${EUREKA\_URI:http://<HOST\_IP>:<PORT>/eureka}  instance:  ip-address: true  zuul:  routes:  users:  path: /users/\*\*  serviceId: user-service  stripPrefix: true  orders:  path: /orders/\*\*  serviceId: order-service  stripPrefix: true  items:  path: /items/\*\*  serviceId: item-service  stripPrefix: true  host:  connect-timeout-millis: 30000  socket-timeout-millis: 30000  retryable: true  hystrix:  command:  default:  execution:  isolation:  thread:  timeoutInMilliseconds: 30000 |

Here, ‘retryable’ is optional as referred above.

# Step 3: Create Business Services

Here, we create our business services.

In this document, we are implementing Item service. We are creating a REST service of Item, where MySQL is used as database.

***Dependency: pom.xml***

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>kpit.poc</groupId>  <artifactId>ms-item-service</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>  <name>ms-item-service</name>  <description>Item service for POC</description>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.0.RELEASE</version>  <relativePath /> <!-- lookup parent from repository -->  </parent>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  <java.version>1.8</java.version>  <spring-cloud.version>Finchley.M8</spring-cloud.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>  </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-actuator</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>  spring-cloud-starter-hystrix-dashboard  </artifactId>  <version>1.4.3.RELEASE</version>  </dependency>  <dependency>  <groupId>com.netflix.hystrix</groupId>  <artifactId>hystrix-metrics-event-stream</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-data-jpa</artifactId>  </dependency>  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  </dependency>  <dependency>  <groupId>io.springfox</groupId>  <artifactId>springfox-swagger2</artifactId>  <version>2.6.1</version>  </dependency>  <dependency>  <groupId>io.springfox</groupId>  <artifactId>springfox-swagger-ui</artifactId>  <version>2.6.1</version>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-security</artifactId>  </dependency>  <dependency>  <groupId>io.swagger</groupId>  <artifactId>swagger-jaxrs</artifactId>  <version>1.5.18</version>  </dependency>  </dependencies>  <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>${spring-cloud.version}</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies>  </dependencyManagement>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  <repositories>  <repository>  <id>spring-milestones</id>  <name>Spring Milestones</name>  <url>https://repo.spring.io/milestone</url>  <snapshots>  <enabled>false</enabled>  </snapshots>  </repository>  </repositories>  </project> |

***Java: Application Class***

|  |
| --- |
| import org.springframework.boot.web.servlet.ServletRegistrationBean;  @SpringBootApplication  @EnableEurekaClient  @EnableCircuitBreaker  @EnableHystrixDashboard  public class ItemServiceApplication {  public static void main(String[] args) {  SpringApplication.run(ItemServiceApplication.class, args);  }    @Bean  public ServletRegistrationBean servletRegistration() {  ServletRegistrationBean registration = new ServletRegistrationBean(new HystrixMetricsStreamServlet(), "/hystrix.stream");  return registration;  }  } |

***Java: Model Class***

|  |
| --- |
| @Entity(name="ms\_items")  public class Item {  @Id  @Column(name="item\_id")  @ApiModelProperty(dataType="int", notes="Database generated item ID")  int itemID;    @Column(name="item\_name")  @ApiModelProperty(dataType="String", notes="Item Name")  String name;    @Column(name="price")  @ApiModelProperty(dataType="double", notes="Price of the item")  double price;    @Column(name="description")  @ApiModelProperty(dataType="String", notes="Item Description")  String description;    @Column(name="quantity")  @ApiModelProperty(dataType="String", notes="Quantity available in inventory")  int quantity;    public Item() {  }  public Item(int itemID, String name, double price, String description, int quantity) {  this.itemID = itemID;  this.name = name;  this.price = price;  this.description = description;  this.quantity = quantity;  }  public int getItemID() {  return itemID;  }  public void setItemID(int itemID) {  this.itemID = itemID;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public double getPrice() {  return price;  }  public void setPrice(double price) {  this.price = price;  }  public String getDescription() {  return description;  }  public void setDescription(String description) {  this.description = description;  }  public int getQuantity() {  return quantity;  }  public void setQuantity(int quantity) {  this.quantity = quantity;  }  } |

***Java: Repository Interface***

|  |
| --- |
| public interface ItemRepository extends CrudRepository<Item, Integer> { } |

***Java: DataBase Service Class***

|  |
| --- |
| @Service  public class ItemService {  @Autowired  ItemRepository itemRepository;    public List<Item> getAllItems() {  return (List<Item>) itemRepository.findAll();  }    public Item getItemByID(int id) {  return itemRepository.findById(id).get();  }    public void addItem(Item item) {  itemRepository.save(item);  }    public boolean updateItem(int id, Item item) {  itemRepository.save(item);  return true;  }    public void deleteItem(int id) {  itemRepository.deleteById(id);  }  } |

***Java: Controller Class***

|  |
| --- |
| @RestController  @Api(value="Items", description="operations pertaining to items in E-cart")  public class ItemController {  @Autowired  private ItemService itemService;    @RequestMapping(method=RequestMethod.GET, value="/items")  @HystrixCommand(fallbackMethod="getAllItemsFallBack")  @ApiOperation(value="View list of available items", response=Item.class, responseContainer="List", authorizations={@Authorization(value="basicAuth")})  @ApiResponses(value={  @ApiResponse(code=200, message="Successfully retrieved all items"),  @ApiResponse(code=400, message="Resource not found")  })  public List<Item> getAllItems() {  return itemService.getAllItems();  }  public List<Item> getAllItemsFallBack() {  List<Item> itemList = new ArrayList<>();  itemList.add(new Item(0, "Error", 0, "Error", 0));  return itemList;  }    @RequestMapping(method=RequestMethod.GET, value="/items/{id}")  @HystrixCommand(fallbackMethod="getItemFallBack")  @ApiOperation(value="Fetch one item with item id", response=Item.class, authorizations={@Authorization(value="basicAuth")})  public Item getItem(@PathVariable("id") int id) {  return itemService.getItemByID(id);  }  public Item getItemFallBack(@PathVariable("id") int id) {  return new Item(0, "Error", 0, "Error", 0);  }    @RequestMapping(method=RequestMethod.POST, value="/items")  @HystrixCommand(fallbackMethod="addItemFallBack")  @ApiOperation(value="Add an item", response=String.class, authorizations={@Authorization(value="basicAuth")})  public String addItem(@RequestBody Item item) {  itemService.addItem(item);  return "{\"message\":\"Item successfully added!\"}";  }  public String addItemFallBack(@RequestBody Item item) {  return "{\"message\":\"Item not added - fallback!\"}";  }    @RequestMapping(method=RequestMethod.PUT, value="/items/{id}")  @HystrixCommand(fallbackMethod="updateItemFallBack")  @ApiOperation(value="Update an item", response=String.class, authorizations={@Authorization(value="basicAuth")})  public String updateItem(@PathVariable("id") int id, @RequestBody Item item) {  if(itemService.updateItem(id, item))  return "{\"message\":\"Item successfully updated!\"}";  else  return "{\"message\":\"Item not updated!\"}";  }  public String updateItemFallBack(@PathVariable("id") int id, @RequestBody Item item) {  return "{\"message\":\"Item not updated - fallback!\"}";  }    @RequestMapping(method=RequestMethod.DELETE, value="/items/{id}")  @HystrixCommand(fallbackMethod="deleteItemFallBack")  @ApiOperation(value="Delete an item with given item id", response=String.class, authorizations={@Authorization(value="basicAuth")})  public String deleteItem(@PathVariable("id") int id) {  itemService.deleteItem(id);  return "{\"message\":\"Item successfully deleted!\"}";  }  public String deleteItemFallBack(@PathVariable("id") int id) {  return "{\"message\":\"Item not deleted - fallback!\"}";  }  } |

***Java: Swagger API Definition Configuration***

This file is mainly used for adding Authentication scheme in Swagger.

|  |
| --- |
| @SwaggerDefinition  public class ApiDefinition implements ReaderListener {  public static final String BASIC\_AUTH\_SCHEME = "basicAuth";    @Override  public void beforeScan(Reader reader, Swagger swagger) {  }    @Override  public void afterScan(Reader reader, Swagger swagger) {  BasicAuthDefinition basicAuthDefinition = new BasicAuthDefinition();  swagger.addSecurityDefinition(BASIC\_AUTH\_SCHEME, basicAuthDefinition);  }  } |

***Java: Swagger Configuration***

If security/authentication is not defined, we can remove *securitySchemes* in below class.

|  |
| --- |
| @Configuration  @EnableSwagger2  public class SwaggerConfig {  @Bean  public Docket itemApi() {  List<SecurityScheme> securitySchemes = new ArrayList<>();  securitySchemes.add(new BasicAuth("basicAuth"));  return new Docket(DocumentationType.SWAGGER\_2)  .select()  .apis(RequestHandlerSelectors.basePackage("kpit.poc.controller"))  .paths(PathSelectors.any())  .build()  .apiInfo(metaData())  .securitySchemes(securitySchemes);  }  private ApiInfo metaData() {  ApiInfo apiMeta = new ApiInfo("Item API",  "Performs all operations on Item resource",  "1.0",  "No terms of service",  new Contact("Prabhash", "http://www.kpit.com", "kumarprabhaa@kpit.com"),  "No license",  "www.google.com");  return apiMeta;  }  } |

***Java: Security Configuration***

This configuration file is required for setting authentication credentials. If you do not want to set this, no need to create this class.

|  |
| --- |
| @Configuration  @EnableWebSecurity  public class SecurityConfig extends WebSecurityConfigurerAdapter {  @Autowired  BasicAuthEntryPoint authEntryPoint;  @Override  protected void configure(HttpSecurity http) throws Exception {  http.csrf().disable().httpBasic().authenticationEntryPoint(authEntryPoint);  }  @Autowired  public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {  auth.inMemoryAuthentication().withUser("kpit").password("{noop}kpit123").roles("admin");  }  } |

***Java: Authentication Entry Point Component***

|  |
| --- |
| @Component  public class BasicAuthEntryPoint extends BasicAuthenticationEntryPoint {  @Override  public void commence(HttpServletRequest request, HttpServletResponse response,  AuthenticationException authException) throws IOException, ServletException {  response.addHeader("WWW-Authenticate", "Basic realm=" +getRealmName());  response.setStatus(HttpServletResponse.SC\_UNAUTHORIZED);  PrintWriter printWriter = response.getWriter();  printWriter.println("Http Status 401 : "+authException.getMessage());  System.out.println("Http Status 401 : "+authException.getMessage());  }  @Override  public void afterPropertiesSet() throws Exception {  setRealmName("Microservices");  super.afterPropertiesSet();  }  } |

***Java: CORS Filter Component***

|  |
| --- |
| import javax.servlet.Filter;  import javax.servlet.FilterChain;  import javax.servlet.FilterConfig;  import javax.servlet.ServletException;  import javax.servlet.ServletRequest;  import javax.servlet.ServletResponse;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Component;  @Component  public class CORSFilter implements Filter {  @Override  public void destroy() { }  @Override  public void init(FilterConfig filterConfig) throws ServletException { }  @Override  public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)  throws IOException, ServletException {  HttpServletResponse res = (HttpServletResponse) response;  HttpServletRequest req = (HttpServletRequest) request;  res.setHeader("Access-Control-Allow-Origin", "\*");  res.setHeader("Access-Control-Allow-Methods", "POST, PUT, GET, DELETE");  res.setHeader("Access-Control-Max-Age", "3600");  res.setHeader("Access-Control-Allow-Headers", "content-type, authorization");  res.setHeader("Access-Control-Allow-Credentials", "true");  if (!"OPTIONS".equalsIgnoreCase(req.getMethod())) {  chain.doFilter(req, res);  }  }  } |

***Properties: application.yml***

|  |
| --- |
| spring:  application:  name: item-service  jpa:  hibernate:  ddl-auto: none  datasource:  url: jdbc:mysql://<HOST\_IP>:<PORT>/<DB\_NAME>  username: root  password: password  server:  address: 0.0.0.0  port: 7073  error:  whitelabel:  enabled: false  eureka:  client:  service-url:  defaultZone: ${EUREKA\_URI:<HOST\_IP>:<PORT>/eureka}  instance:  ip-address: true |

# Step 4: Create Turbine Service

***Dependency: pom.xml***

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>kpit.poc</groupId>  <artifactId>ms-turbine-service</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>  <name>ms-turbine-service</name>  <description>Turbine service for POC</description>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.0.0.RELEASE</version>  <relativePath/> <!-- lookup parent from repository -->  </parent>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  <java.version>1.8</java.version>  <spring-cloud.version>Finchley.M8</spring-cloud.version>  </properties>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-actuator</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-hystrix-dashboard</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-turbine</artifactId>  </dependency>    <dependency>  <groupId>com.netflix.hystrix</groupId>  <artifactId>hystrix-metrics-event-stream</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  </dependencies>  <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>${spring-cloud.version}</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies>  </dependencyManagement>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  <repositories>  <repository>  <id>spring-milestones</id>  <name>Spring Milestones</name>  <url>https://repo.spring.io/milestone</url>  <snapshots>  <enabled>false</enabled>  </snapshots>  </repository>  </repositories>  </project> |

***Java: Application Class***

|  |
| --- |
| @SpringBootApplication  @EnableEurekaClient  @EnableHystrixDashboard  @EnableTurbine  public class TurbineServiceApplication {  public static void main(String[] args) {  SpringApplication.run(TurbineServiceApplication.class, args);  }    @Bean  public ServletRegistrationBean servletRegistration() {  ServletRegistrationBean registration = new ServletRegistrationBean(new HystrixMetricsStreamServlet(), "/hystrix.stream");  return registration;  }  } |

***Properties: application.yml***

|  |
| --- |
| spring:  application:  name: turbine-service  server:  address: 0.0.0.0  port: 8089  error:  whitelabel:  enabled: false  eureka:  client:  service-url:  defaultZone: ${EUREKA\_URI:<HOST\_IP>:<PORT>/eureka}  instance:  ip-address: true    turbine:  clusterNameExpression: new String("default")  appConfig: user-service, order-service, item-service |

# Step 5: Deployment on Docker Container

## Build Jar files for each spring boot application

We need to change the application configuration file as per our requirements. Server port, Eureka and MySQL configuration details should be updated before building jars.

***Building jar for each services:***

Go to maven project directory on command prompt and execute below command:

|  |
| --- |
| mvn clean install |

Jar file is created in *target* directory.

## Create Dockerfile

Once jar files are created we need to create *Dockerfile* for the same.

|  |
| --- |
| FROM openjdk:8-jdk-alpine  WORKDIR /  ADD ms-turbine-service-0.0.1-SNAPSHOT.jar app.jar  EXPOSE 8089  CMD ["java","-jar","app.jar"] |

Change the jar file name with your jar file name on 3rd line and save the file.

## Build Image

Execute below command for creating image for the *Dockerfile*.

|  |
| --- |
| sudo docker build -t eureka-server . |

Change file name of image as per requirement.

## Run Container

Execute below command for starting the container.

|  |
| --- |
| sudo docker run --net=host -d -p 8084:8084 turbine-services |

First port number represents Host port no and Second port number represents Docker port number.

***--net=host*** is used for binding container to host network directly.

## Important Notes:

***Order of Execution of containers:***

Mysql -> Eureka -> Zuul -> User -> Order -> Item -> Turbine (Optional)

***Basic Authentication Credentials for API:***

This value is set in java class.

kpit:kpit123

***Swagger API Documentation URL:***

User API: <http://localhost:7071/swagger-ui.html>

Order API: <http://localhost:7072/swagger-ui.html>

Item API: <http://localhost:7073/swagger-ui.html>

## Current URL Configurations:

|  |  |  |
| --- | --- | --- |
| Service Name | URL | Zuul-Router |
| Eureka Server | <http://localhost:9091> |  |
| Zuul Service | <http://localhost:8084> |  |
| User Service | <http://localhost:7071/users> | <http://localhost:8084/users/users> |
| Order Service | <http://localhost:7072/orders> | <http://localhost:8084/orders/orders> |
| Item Service | <http://localhost:7073/items> | <http://localhost:8084/items/items> |
| Turbine Service | <http://localhost:8089/hystrix> |  |

Add ‘*http://localhost:8089/turbine.stream*’ in the dashboard URL.

## Docker Executed Commands:

MySQL Service – It should have *Dockerfile* and *sql* file

|  |
| --- |
| cd mysql-dockerfile-directory  sudo docker build -t mysql-service .  sudo docker run --net=host -d -p 3306:3306 mysql-service |

Eureka Server – It should have *Dockerfile* and Jar file

|  |
| --- |
| cd eureka-dockerfile-directory  sudo docker build -t eureka-server .  sudo docker run --net=host -d -p 9091:9091 eureka-server |

Zuul Service – It should have *Dockerfile* and *Jar* file

|  |
| --- |
| cd zuul-dockerfile-directory  sudo docker build -t zuul-service .  sudo docker run --net=host -d -p 8084:8084 zuul-service |

User Service – It should have *Dockerfile* and *Jar* file

|  |
| --- |
| cd user-dockerfile-directory  sudo docker build -t user-service .  sudo docker run --net=host -d -p 7071:7071 user-service |

Order Service – It should have *Dockerfile* and *Jar* file

|  |
| --- |
| cd order-dockerfile-directory  sudo docker build -t order-service .  sudo docker run --net=host -d -p 7072:7072 order-service |

Item Service – It should have *Dockerfile* and *Jar* file

|  |
| --- |
| cd item-dockerfile-directory  sudo docker build -t item-service .  sudo docker run --net=host -d -p 7073:7073 item-service |

Turbine Service – It should have *Dockerfile* and *Jar* file

|  |
| --- |
| cd turbine-dockerfile-directory  sudo docker build -t turbine-service .  sudo docker run --net=host -d -p 8089:8089 turbine-service |