**Quick Setup guide**

**Eureka**

Eureka is a REST based service primarily used for locating services (service registry) for the purpose of load balancing and fail-over of middle tier servers.

Download and setup eureka from the below location :

https://github.com/Netflix/eureka

Standing up an instance of the Eureka service registry is easy if we have org.springframework.boot:spring-cloud-starter-eureka-server on our classpath and we should have @EnableEurekaServer annotation on the main class of the microservice.

Eureka server by default runs on port 8761. We can change the default config by editing port in application.yml file like server:port:9761

Build the project with maven and run the jar. Once started we can view the Eureka homepage on http://localhost:8761

It displays all the services registered with Eureka

**Microservice registry with Eureka**

Each microservice registers itself with eureka service registry. A service with org.springframework.cloud:spring-cloud-starter-eureka on the classpath will be registered with the Eureka registry by its spring.application.name.

The src/main/resources/boostrap.yml file for each of my services looks like this, where my-service is the service name that changes from service to service:

spring:

application:

name: my-service

Also we need to specify the Eureka client's server url in the microservice config with which it will register like :

eureka:

instance:

client:

serviceUrl:

defaultZone: http://127.0.0.1:8761/eureka/

**Zuul**

Zuul is a gateway service that provides dynamic routing, monitoring, security etc.

Download zuul from the below location :

https://github.com/Netflix/zuul

Zuul configuration is defined in bootstrap.yml file. By default zuul runs on port 8087, if you want to change the zuul port of the eureka client's serverUrl then you can edit the config given in bootstrap.yml file

There are 2 ways to access microservices from zuul API gateway.

a. By defining the microservices url in bootstrap.yml under zuul : routes like :

zuul:

routes.product-be-rest:

path: /api/product/detail /\*\*

b. Define interfaces for your microservices and annotate it with @FeignClient like :

@FeignClient("product-service")

**public** **interface** ProductClient {

@RequestMapping(value ="api/product/detail", method = RequestMethod.GET)

String detail();

Once we build the zuul API Gateway and run the jar we can access the microservice API by giving call to zuul API gateway's URL.

**Ribbon**

Ribbon is a client side cloud based library for load balancing and fault tolerance.

Download and setup ribbon from the below location

https://github.com/Netflix/ribbon

To get ribbon binaries, we can include following maven dependency and build it :

<dependency>

<groupId>com.netflix.ribbon</groupId>

<artifactId>ribbon</artifactId>

<version>2.0-RC1</version>

</dependency>

For load balancing of microservices, we can configure the list of servers for microservices in the Zuul API gateway config file i.e bootstrap.yml like this :

product-service:

ribbon:

eureka:

enabled: false

listOfServers: localhost:9746,localhost:9748

ServerListRefreshInterval: 15000

With this configuration the Zuul server will make sure that even if one instance of product-service goes down it will automatically establish connection with the another server configured with ribbon.

From the Zuul API gateway if we are using RestTemplate to interact with the microservice then in the Controller class where we are using restTemplate we need to annotate that class with the below annotation :

@RibbonClient(name = "product-service", configuration = RibbonClientConfiguration.class)

public class RestController {

@Autowired

RestTemplate restTemplate;

.... }

where "product-service" is the name of microservice and the RibbonClientConfiguration is the configuration class where RibbonConfig bean is defined which will refer to the ribbon configuration specified in the bootstrap.yml file

**Hystrix**

Hystrix is a latency and fault tolerance librabry which also helps in real time monitoring and configuration changes.

Download and setup hystrix from the below location

https://github.com/Netflix/hystrix

You can build Hystrix either with maven or with gradle.

For building with maven we will need to add following dependency

<dependency>

<groupId>com.netflix.hystrix</groupId>

<artifactId>hystrix-core</artifactId>

<version>x.y.z</version>

</dependency>

or we can build using gradle with command

gradlew build

For enabling hystrix support in all the microservice main class we need to add the following annotation :

@EnableHystrix

And then in the main class of Zuul API gateway project from where we give call to different microservices we need to add annotations @EnableCircuitBreaker and @EnableHystrixDashboard like:

@EnableCircuitBreaker

@EnableHystrixDashboard

@SpringBootApplication

@EnableEurekaClient

public class App {

....

}

Then once we start this Zuul API gatweway server we can view the hysterix dashboard with the url : http://localhost:8087/hysterix

which will display the hysterix dashboard with which we can monitor the microservices associated with the zuul server.