Consul - Completed

1. Download Consul
2. consul agent -server -bootstrap-expect=1 -data-dir=consul-data -ui -bind=192.168.99.1

Find out your IP address using ipconfig

1. <http://localhost:8500>

* Spring boot with

Actuator

Web

Rest Repositories

Consul Discovery

1. @EnableDiscoveryClient to main class
2. Add following to application.properties

server.port=9098

spring.application.name: student-service

management.security.enabled=false

1. Create School App
2. Clone Client App to clientapp2
3. <http://localhost:8098//getSchoolDetails/abcschool> and see in console - Load Balancingx`x`
4. Hystrix - Completed

Hystrix :

1. Add

<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>

1. Add @EnableCircuitBreaker annotation
2. Add @EnableHystrixDashboard annotation
3. Add annotation @HystrixCommand(fallbackMethod = "myFallbackMethod")

Add following to hystrixschoolapp properties file

management.endpoints.web.exposure.include=hystrix.stream

This is the endpoint

localhost:8098/actuator/hystrix.stream

Open <http://localhost:8098/hystrix> and add above endpoint

3> Consul-Config-Server

1. Spring-boot-starter-actuator
2. Spring-cloud-starter-config
3. Spring-boot-starter-data-rest
4. Spring-boot-starter-web & test

Spring-config-client

Spring-config-server

Config-server-repo

Test following after getting server up & running:

* http://localhost:8888/config-server-client/development
* <http://localhost:8888/config-server-client/production>

Change in bootstrap.properties of client & check ( from prod to dev or so)

<http://localhost:8080/msg>

make changes & push to git

post from postman <http://localhost:8080/refresh>

check msg again

Zuul Proxy

spring-boot-zuulgatwayproxy

spring-boot-zuulgatwayproxy-student-service

Start zool-proxy-student

<http://localhost:8090/echoStudentName/Sajal>

<http://localhost:8090/getStudentDetails/Sajal>

start zool-proxy

and visit

<http://localhost:8080/student/getStudentDetails/Sajal>

<http://localhost:8080/student/echoStudentName/Sajal>

Monitoring - Completed

Eureka-server – Service Discovery

Api-gateway

Employee-service

@EnableEurekaClient in EmployeeService

Application.yml

ApiGatway – Hystrix + Eureka Client

Application .yml in api-gateway

Start Eureka Server -> Api – Employee

Hit url

<http://localhost:8010/employeeDetails/111>

Hit url  <http://localhost:8010/hystrix>

Add <http://localhost:8010/hystrix.stream> for streaming

Hit <http://localhost:8761/admin>

Zipkin – To be completed

Download zipkin from <https://search.maven.org/remote_content?g=io.zipkin.java&a=zipkin-server&v=LATEST&c=exec>

Run as java -jar

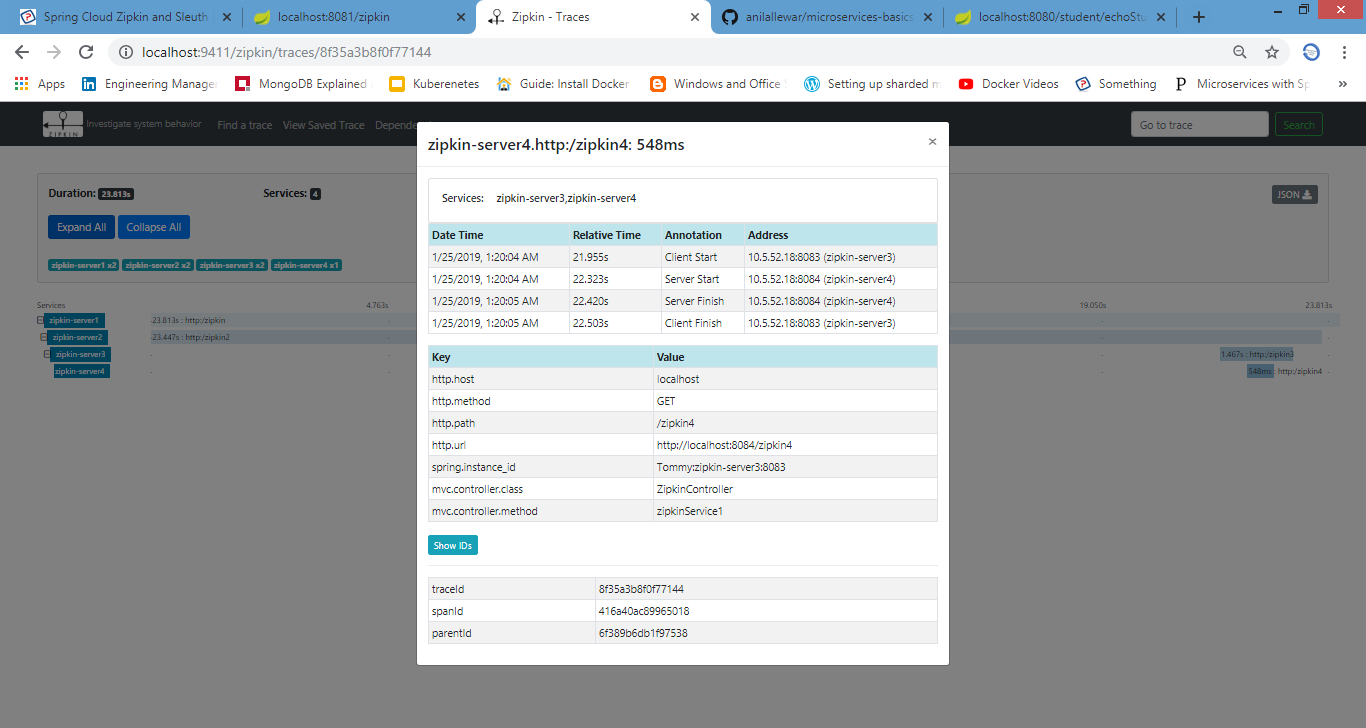
Go to <http://localhost:9411/zipkin/>

Collector & storage

[Sleuth](https://cloud.spring.io/spring-cloud-sleuth/) is a tool from Spring cloud family. It is used to generate the trace id, span id and add these information to the service calls in the headers and MDC, so that It can be used by tools like Zipkin and [ELK](https://howtodoinjava.com/microservices/elk-stack-tutorial-example/) etc. to store, index and process log files. As it is from spring cloud family, once added to the CLASSPATH, it automatically integrated to the common communication channels like –

* requests made with the [RestTemplate](https://howtodoinjava.com/spring/spring-restful/spring-restful-client-resttemplate-example/) etc.
* requests that pass through a [Netflix Zuul](https://howtodoinjava.com/spring/spring-cloud/spring-cloud-api-gateway-zuul/) microproxy
* HTTP headers received at [Spring MVC](https://howtodoinjava.com/spring-mvc-tutorial/) controllers
* requests over messaging technologies like Apache Kafka or RabbitMQ etc.

Build All



Hoverfly-actual-service

Hoverfly-actual-service-client

Download Hoverfly from <https://hoverfly.readthedocs.io/en/latest/pages/introduction/downloadinstallation.html>

And unzip

Start actual-service

Add vm parameter -Dmode=proxy to service-client and start it

Hoverctrl start

Hoverctrl mode capture

Open http://localhost:8888/dashboard and ensure mode is in Capture mode

Now it http://localhost:8080/invoke for multiple times

Ensure it is captured by looking back at Dashboard

hoverctl export simulations.json

( You can use hoverctl import simulations.json in future when you want to import)

Change mode to simulate

hoverctl mode simulate

http://localhost:8080/invoke and see the simulation count is increasing

Now stop the service

Now hit http://localhost:8080/invoke and see the magic 😊

Now change mode to capture

And now what happens if you refresh invoke endpoint.

https://play.grafana.org - Play Around MicroServices

Ribbon

**Ribbon-client**

**Ribbon-Server**

To do that we need to use different port for this, to start service in a specific port we need to pass the port in this way.  
java -jar -Dserver.port=XXXX target/YYYYY.jar. We will create 3 instances of this service in ports 9090, 9091 and 9092 ports.

Try with & without following :

|  |
| --- |
| application.properties |
| server.ribbon.listOfServers=localhost:9090,localhost:9091,localhost:9092  server.ribbon.eureka.enabled=false |

Security :

1. Import Spring-MicroServices-OAuth-Server

Start the Project

Go to localhost:9090/oauth/token ( No / ) from POSTMAN ( POST request)

Authorization : Basic Auth -> webapp/websecret

Params -> grant\_type= password, username=user1, password=password1

1. Import OAuth-Resource-Server

See Changes

Go to localhost:9091/resource/endpoint/

Auth Failed

Now get Token for User1/password1 using endpoint in #1

And add a parameter access\_token= <token>

Again Failed

Now get Token for admin/password2 using endpoint in #1

Now it should succeed

1. Import oAuth-Client

Go to localhost:8080/execute

Try login with user1/password1

Failed – Clear Cookies & again go to same url

Login with admin/password2

Resource Protected Message is seen

Cloud-Foundry

Spring-helloworld-cf

Download from <https://cli.run.pivotal.io/stable?release=windows64&source=github>

And unzip here

Install exe

Type cf from command line and see

Register yourself at <https://account.run.pivotal.io/z/uaa/sign-up>

Activate your account

Go to webservices

Login using

cf login -a api.run.pivotal.io

Build Project using mvn

F:\sts-workspace\microservices\spring-helloworld-cf>cf push myapptechnopreneur -d domainname.com -p target\sp

ring-helloworld-cf-0.0.1-SNAPSHOT.jar

Optional below :

cf create-domain technopreneur mydomaintechnopreneur.com

cf create-route development mydomaintechnopreneur.com -n www

Eureka

Spring-eureka-client-student

Spring-edureka-client-school

Spring-eureka-server

application.yml  in server

bootstrap.yml

check server <http://localhost:8761/>

enableedurekaclient

and application.yml

Spring Data flow

Sping-data-flow-source

processor

Sink

Run Data flow

<http://repo.spring.io/milestone/org/springframework/cloud/spring-cloud-dataflow-server-local/1.7.3.RELEASE/>

run as java -jar

Download

<http://repo.spring.io/milestone/org/springframework/cloud/spring-cloud-dataflow-shell/1.7.3.RELEASE/spring-cloud-dataflow-shell-1.7.3.RELEASE.jar>

run as java -jar

under this shell:

app register --name source-app --type source --uri maven://com.example:source:jar:0.0.1-SNAPSHOT

app register --name processor-app --type processor --uri maven://com.example:processor:jar:0.0.1-SNAPSHOT

app register --name sink-app --type sink --uri maven://com.example:sink:jar:0.0.1-SNAPSHOT

Go to <http://localhost:9393/dashboard>

stream create --name log-data2 --definition 'source-app | processor-app | sink-app'

stream deploy --name log-data2

Check <http://localhost:8098/getStudentDetailsForSchool/abcschool>

Create School Service

Consume Student

Can clone student and check for Load balancing

Go to <http://localhost:8761/> and see instances

Using Spring Cloud Bus

springcloudconfigrepo

Employee-config

Employee-producer-eureka

Employee-producer2

eureka-server

eureka -server2

Edit Application.properties in employee-config to point to your git repo

Start servers & visit localhost:8761 where you can see 2 servers ; while null in 8762

Now Edit the application.properties under springcloudconfigrepo to 8762

Post <http://localhost/bus/refresh>

Now all servers should move to 8762