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

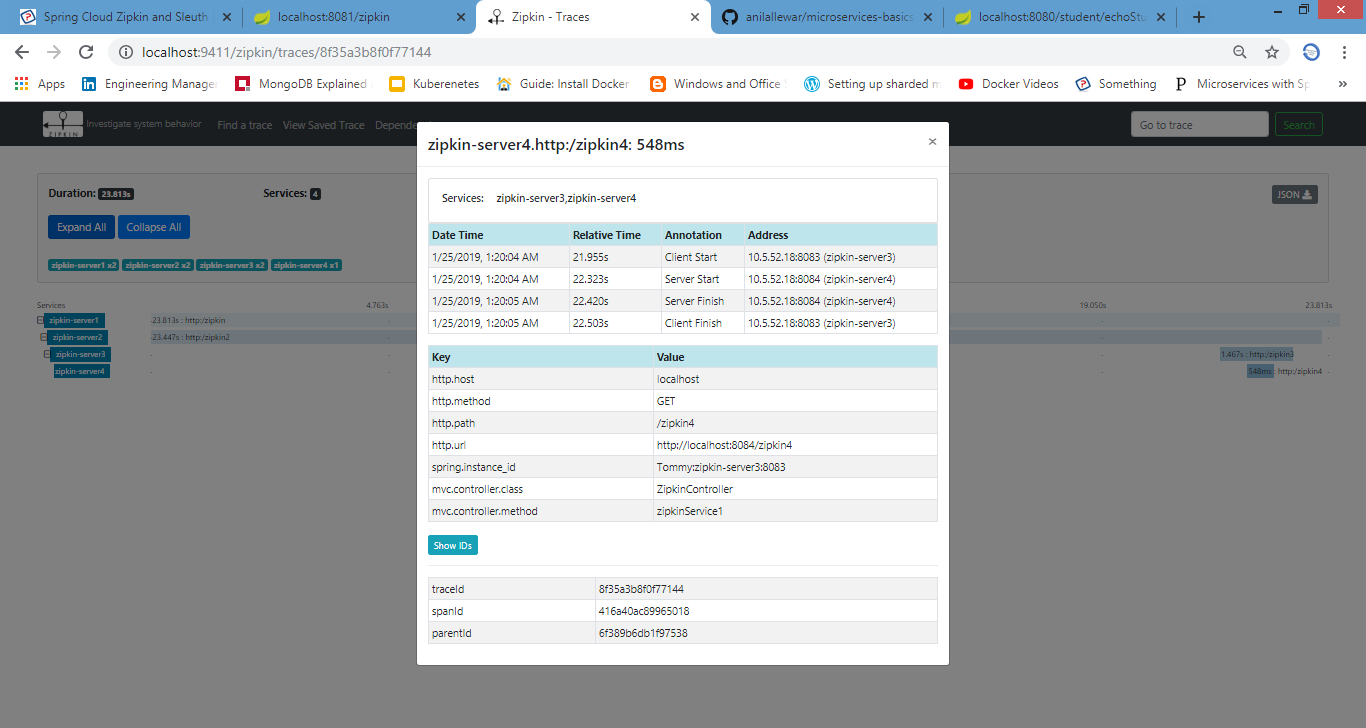
Go to <http://localhost:9411/zipkin/>

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

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

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