**Master Microservices with Spring Boot and Spring Cloud**

**Section 1: Introduction**

**Section 2: Introduction to Web Services**

Web Services: S/W system designed to support interoperable machine-to-machine interaction over a network.

**Section 3: Restful Web Services with Spring boot**

**scenario: Basic Curd Operations**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/eb9809487afcbbb3a0155eeb464e19f8b11a388b>

**scenario: Validation Implemented**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/de4240a78dad2f3288a060f14135d25ae967036a>

**scenario: HATEOAS Implemented**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/da9b0f96359aa0cba6c93952bb9b73dbeb84ed29>

**scenario: Internationalization(i18) Implemented**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/32243f7e76a1ff1893a622b12ff6528811237ffb>

**scenario: Content Negotiation (xml support)**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/e9d044a3f11ab95098296e588687750a3b841f21>

**scenario: Swagger Implementation**

<http://localhost:8080/swagger-ui/index.html>

<http://localhost:8080/v3/api-docs>

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/168ded511dcaaee02bd668f0357af844fcb0692f>

**scenario: Actuator Implementation**

[localhost:8080/actuator](http://localhost:8080/actuator)

management.endpoints.web.exposure.include=\*

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/d4c681203633f3520680cbdf9c3f4c2671b486f8>

**scenario: Visualizing API with HAL Explore**

<http://localhost:8080/>

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/15e5c29254c3fceaa7960df546410f45fbf67ccf>

**scenario: Filtering Implementation**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/93a10db21e3fc83302d01062e2a239729e213fa5>

**scenario: Versioning Implementation**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/c2e7bef315659d6cd53e9dbdba09045d712c2568>

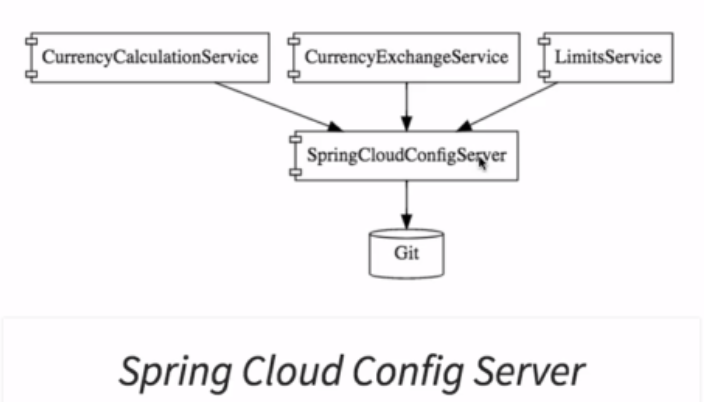
**Section 4: Quick introduction to Microservices**

**Section 5: Microservices with Spring cloud version 1**

|  |  |
| --- | --- |
| **Services** | **Port** |
| limits-service | 8080 |
| spring-cloud-config-server | 8888 |
| currency-exchange-service | 8000, 8001, 8002 … … … |
| currency-conversion-service | 8100, 8101, 8102 … … … |
| netflix-eureka-naming-server | 8761 |
| netflix-zuul-api-gateway-server | 8765 |
| Zipkin | 9411 |

**Use case: simple limit-service to read data from properties file**

**limits-service**



**scenario: simple limit service to read data from properties file**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/7ad41139e5ee50ba9fb7ca8085cc03848fc98dd2>

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/c762cfedb75aa753bf4b2a3ba38452808ae8ede4>

(Package name changed ws->ms)

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/486635df0aa86d9826640c0e1e1507a69382f03d>

(package name changed ws->ms)

**Use case: Spring Cloud Config Service**

**spring-cloud-config-server**

**Use case: create local git repository**

bibhu@DESKTOP-ISC6SEH MINGW64 /c/dev\_env/intellij\_projects/udemy\_master\_ms\_with\_spring\_boot\_cloud/git-localconfig-repo (master)

$ git init

Initialized empty Git repository in C:/dev\_env/intellij\_projects/udemy\_master\_ms\_with\_spring\_boot\_cloud/git-localconfig-repo/.git/

bibhu@DESKTOP-ISC6SEH MINGW64 /c/dev\_env/intellij\_projects/udemy\_master\_ms\_with\_spring\_boot\_cloud/git-localconfig-repo (master)

$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

limits-service.properties

nothing added to commit but untracked files present (use "git add" to track)

bibhu@DESKTOP-ISC6SEH MINGW64 /c/dev\_env/intellij\_projects/udemy\_master\_ms\_with\_spring\_boot\_cloud/git-localconfig-repo (master)

$ git add limits-service.properties

bibhu@DESKTOP-ISC6SEH MINGW64 /c/dev\_env/intellij\_projects/udemy\_master\_ms\_with\_spring\_boot\_cloud/git-localconfig-repo (master)

$ git commit -m "limit-service properties file added"

[master (root-commit) abd568c] limit-service properties file added

1 file changed, 2 insertions(+)

create mode 100644 limits-service.properties

**Use case: set up local git repository**

*git-localconfig-repo*

limits.service.proerties

limits-service.maximum=8888  
limits-service.minimum=88

*spring-cloud-config-server*

@SpringBootApplication  
@EnableConfigServer  
public class SpringCloudConfigServerApplication {  
  
 public static void main(final String[] args) {  
 SpringApplication.*run*(SpringCloudConfigServerApplication.class, args);  
 }  
  
}

server.port=8888  
spring.application.name=spring-cloud-config-server  
spring.cloud.config.server.git.uri=file:///C:/dev\_env/intellij\_projects/udemy\_master\_ms\_with\_spring\_boot\_cloud/git-localconfig-repo

<http://localhost:8888/limits-service/default>



**scenario: set up local git repository**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/5cb10cc70d36e46581345e6fb15bb3f1d72483fa>

**Use case: configure multiple environments in Git**

*git-localconfig-repo*

limits-service-dev.proerties

limits-service.maximum=1111  
limits-service.minimum=11

limits-service-qa.proerties

limits-service.maximum=2222  
limits-service.minimum=22

bibhu@DESKTOP-ISC6SEH MINGW64 /c/dev\_env/intellij\_projects/udemy\_master\_ms\_with\_spring\_boot\_cloud/git-localconfig-repo (master)

$ git status

On branch master

Untracked files:

(use "git add <file>..." to include in what will be committed)

limits-service-dev.properties

limits-service-qa.properties

nothing added to commit but untracked files present (use "git add" to track)

bibhu@DESKTOP-ISC6SEH MINGW64 /c/dev\_env/intellij\_projects/udemy\_master\_ms\_with\_spring\_boot\_cloud/git-localconfig-repo (master)

$ git add limits-service-dev.properties

bibhu@DESKTOP-ISC6SEH MINGW64 /c/dev\_env/intellij\_projects/udemy\_master\_ms\_with\_spring\_boot\_cloud/git-localconfig-repo (master)

$ git add limits-service-qa.properties

bibhu@DESKTOP-ISC6SEH MINGW64 /c/dev\_env/intellij\_projects/udemy\_master\_ms\_with\_spring\_boot\_cloud/git-localconfig-repo (master)

$ git commit -m "limit-service multiple env properties file added"

[master f932c96] limit-service multiple env properties file added

2 files changed, 4 insertions(+)

create mode 100644 limits-service-dev.properties

create mode 100644 limits-service-qa.properties

<http://localhost:8888/limits-service/dev>



**Use case: Connect limits-service to spring-cloud-config-server**

Rename the limits-service application.properties to bootstrap.properties

*limits-service*

bootstrap.proerties

server.port=8080  
spring.application.name=limits-service  
spring.cloud.config.url=http://localhost:8888

<http://localhost:8080/limits>

{"maximum":8888,"minimum":88}

**scenario: connect limits-service to spring-cloud-config-server**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/7dbaa180e8d057855ff6f9ef3a9aef99b48c6ceb>

**Use case: configure profiles for limits-service**

*limits-service*

bootstrap.proerties

server.port=8080  
spring.application.name=limits-service  
spring.cloud.config.url=http://localhost:8888

spring.profiles.active=qa

<http://localhost:8080/limits>

{"maximum":2222,"minimum":22}

spring.profiles.active=dev

<http://localhost:8080/limits>

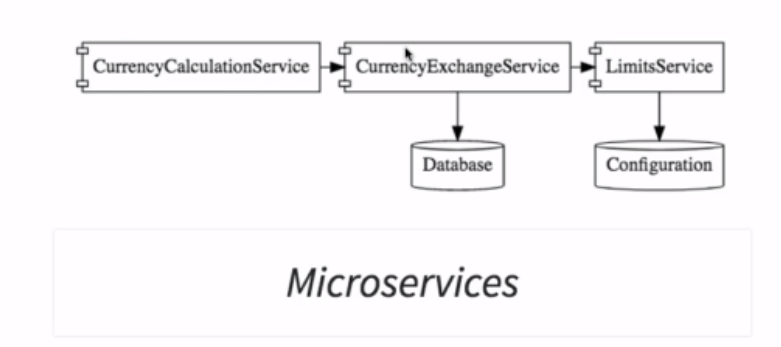
{"maximum":1111,"minimum":11}

**scenario: configure profiles for limits-service**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/2bfdcf48fe5645273912db400e8efb5b220d1416>

**Use case: Introduction to currency conversion and currency exchange service**

**currency-exchange-service**

****

**scenario: initial currency-exchange-service setup**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/9820d7b21fdc57c2973789aea260f61f586e07c1>

**Run in multiple instance (multiple port)**

Create duplicate instance of currency-exchange-service and add -Dserver.port=8001 as VM argument in IDEs.

**H2-console**

<http://localhost:8000/h2-console>

**scenario: Implementation of jpa and h2 database**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/052ccde26cb60b82e8b43196dd31ab8e46ae6ea0>

**currency-conversion-service**

**scenario: initial currency-conversion-service setup**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/11509cc33d112c6abda1292e066785f99c24db05>

**scenario: invoking currency-exchange-service from currency-conversion-service**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/2eb1b8fe9f6b2990568c7fb74f90dac6a2faca3f>

**Use case: Feign Rest client as Service invocation (to replace RestTemplate)**

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-openfeign</artifactId>  
</dependency>

@SpringBootApplication  
@EnableFeignClients("com.in28min.ms.currencyconversionservice")  
public class CurrencyConversionServiceApplication {  
 public static void main(final String[] args) {  
 SpringApplication.*run*(CurrencyConversionServiceApplication.class, args);  
 }  
}

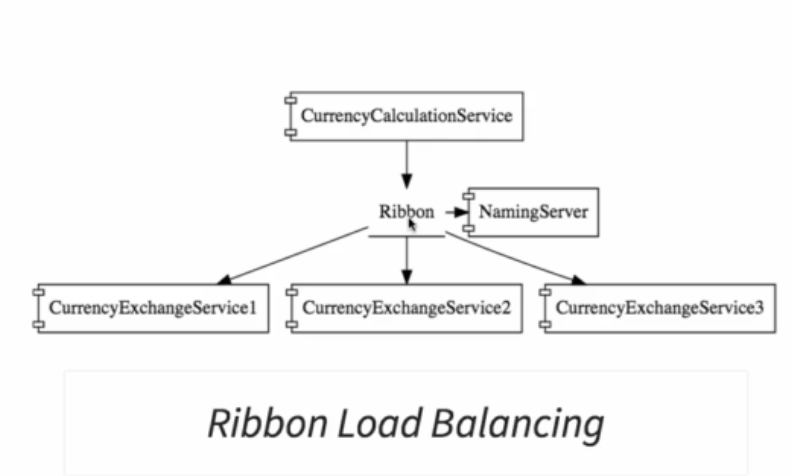
@FeignClient(name = "currency-exchange-service", url = "localhost:8000")  
public interface CurrencyExchangeServiceProxy {  
  
 @GetMapping("/currency-exchange/from/{from}/to/{to}")  
 public CurrencyConversionBean retrieveExchangeValue  
 (@PathVariable("from") final String from, @PathVariable("to") final String to);  
}

@GetMapping("/currency-converter-feign/from/{from}/to/{to}/quantity/{quantity}")  
public CurrencyConversionBean convertCurrencyFeign(@PathVariable final String from, @PathVariable final String to,  
 @PathVariable final BigDecimal quantity) {  
  
 final CurrencyConversionBean response = this.proxy.retrieveExchangeValue(from, to);

**scenario: Implementation of Feign Rest Client as service invocation**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/072221848f13f282ef5737eaa8024b1eca697188>

**Use case: Client-side Load balancing with Ribbon**



*<*dependency*>  
 <*groupId*>*org.springframework.cloud*</*groupId*>  
 <*artifactId*>*spring-cloud-starter-netflix-ribbon*</*artifactId*>  
</*dependency*>*

*//@FeignClient(name = "currency-exchange-service", url = "localhost:8000")*@FeignClient*(*name = "currency-exchange-service"*)*@RibbonClient*(*name = "currency-exchange-service"*)*public interface CurrencyExchangeServiceProxy *{* @GetMapping*(*"/currency-exchange/from/{from}/to/{to}"*)* public CurrencyConversionBean retrieveExchangeValue  
 *(*@PathVariable*(*"from"*)* final String from, @PathVariable*(*"to"*)* final String to*)*;  
*}*

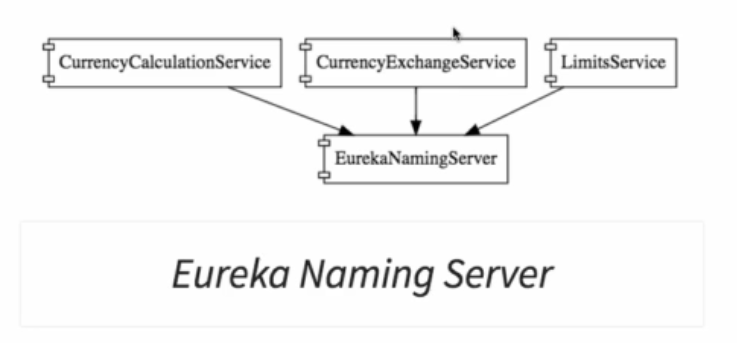
server.port=8100  
spring.application.name=currency-conversion-service  
currency-exchange-service.ribbon.listOfServers=http://localhost:8000, http://localhost:8001

**scenario: setting up client-side load balancing with Ribbon**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/22c3b8010fee403f980e5e7afaba3c5317325fc0>

**Use case: Understanding need of naming server**

**netflix-eureka-naming-server**



*<*dependency*>  
 <*groupId*>*org.springframework.cloud*</*groupId*>  
 <*artifactId*>*spring-cloud-starter-config*</*artifactId*>  
</*dependency*>*

*<*dependency*>  
 <*groupId*>*org.springframework.cloud*</*groupId*>  
 <*artifactId*>*spring-cloud-starter-netflix-eureka-server*</*artifactId*>  
</*dependency*>*

@SpringBootApplication  
@EnableEurekaServer  
public class NetflixEurekaNamingServerApplication *{* public static void main*(*final String*[]* args*) {* SpringApplication.*run(*NetflixEurekaNamingServerApplication.class, args*)*;  
 *}  
  
}*

server.port=8761  
spring.application.name=netflix-eureka-naming-server  
eureka.client.register-with-eureka=false  
eureka.client.fetch-registry=false

<http://localhost:8761>

**scenario: setting of Eureka Naming Server**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/4cf64c89224860518d0ae7601c49e803bcfe33ed>

**Use case: Connecting Currency services(both) to Eureka**

**currency-conversion-service**

*<*dependency*>  
 <*groupId*>*org.springframework.cloud*</*groupId*>  
 <*artifactId*>*spring-cloud-starter-netflix-eureka-client*</*artifactId*>  
</*dependency*>*

@SpringBootApplication  
@EnableFeignClients*(*"com.in28min.ms.currencyconversionservice"*)*@EnableDiscoveryClient  
public class CurrencyConversionServiceApplication *{* public static void main*(*final String*[]* args*) {* SpringApplication.*run(*CurrencyConversionServiceApplication.class, args*)*;  
 *}  
  
}*

eureka.client.service-url.default-zone=http://localhost:8761/eureka

**scenario: connect currency-conversion-service to Eureka**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/c61fb7242aaf385cac02a3cfa8a57e694f3e4c3b>

**currency-exchange-service**

*<*dependency*>  
 <*groupId*>*org.springframework.cloud*</*groupId*>  
 <*artifactId*>*spring-cloud-starter-netflix-eureka-client*</*artifactId*>  
</*dependency*>*

@SpringBootApplication  
@EnableDiscoveryClient  
public class CurrencyExchangeServiceApplication *{* public static void main*(*String*[]* args*) {* SpringApplication.*run(*CurrencyExchangeServiceApplication.class, args*)*;  
 *}  
  
}*

eureka.client.service-url.default-zone=http://localhost:8761/eureka

**scenario: connect currency-exchange-service to Eureka**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/5e85d8bee9af18c7c95d6389ef0d755f88289264>

**COURSE UPDATE: Exclude dependency on jackson-dataformat-xml**

*<*dependency*>  
 <*groupId*>*org.springframework.cloud*</*groupId*>  
 <*artifactId*>*spring-cloud-starter-netflix-eureka-client*</*artifactId*>  
 <*exclusions*>  
 <*exclusion*>  
 <*groupId*>*com.fasterxml.jackson.dataformat*</*groupId*>  
 <*artifactId*>*jackson-dataformat-xml*</*artifactId*>  
 </*exclusion*>  
 </*exclusions*>  
</*dependency*>*

**scenario: COURSE UPDATE : Exclude dependency on jackson-dataformat-xml (extra)**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/e27f05a01f29c3648b3996eb0ff49859a88251e4>

**Use case: Distributing calls using Eureka and Ribbon**

server.port=8100  
spring.application.name=currency-conversion-service  
*#currency-exchange-service.ribbon.listOfServers=http://localhost:8000, http://localhost:8001*eureka.client.service-url.default-zone=http://localhost:8761/eureka

**scenario: Distributing calls using Eureka and Ribbon**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/259d3f05df4f9283400625e0a9846b536de3f904>

**Use case: API Gateway**

**netflix-zuul-api-gateway-server**

*<*dependency*>  
 <*groupId*>*org.springframework.cloud*</*groupId*>  
 <*artifactId*>*spring-cloud-starter-netflix-zuul*</*artifactId*>  
</*dependency*>*

server.port=8765  
spring.application.name=netflix-zuul-api-gateway-server  
eureka.client.service-url.default-zone=http://localhost:8761/eureka

@SpringBootApplication  
@EnableZuulProxy  
@EnableDiscoveryClient  
public class NetflixZuulApiGatewayServerApplication *{* public static void main*(*final String*[]* args*) {* SpringApplication.*run(*NetflixZuulApiGatewayServerApplication.class, args*)*;  
 *}  
  
}*

**scenario: setting up Zuul API gateway**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/fd80117d2e01c3a312a452709c7b76cb7cb2f8cc>

**Use case: Implementing zuul logging filter**

@Component  
public class ZuulLoggingFilter extends ZuulFilter *{* private final Logger logger = LoggerFactory.*getLogger(*this.getClass*())*;  
  
 @Override  
 public String filterType*() {* return "pre";  
 *}* @Override  
 public int filterOrder*() {* return 1;  
 *}* @Override  
 public boolean shouldFilter*() {* return true;  
 *}* @Override  
 public Object run*()* throws ZuulException *{* final HttpServletRequest request =  
 RequestContext.*getCurrentContext()*.getRequest*()*;  
 this.logger.info*(*"request -> {} request uri -> {}",  
 request, request.getRequestURI*())*;  
 return null;  
 *}  
}*

<http://localhost:8765/currency-exchange-service/currency-exchange/from/USD/to/INR>

scenario: Implementing zuul logging filter

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/9f40eedd93718dfe7f4ab21054f3e87d8825f22d>

**Use case: Setting up Zuul API gateway between microservices**

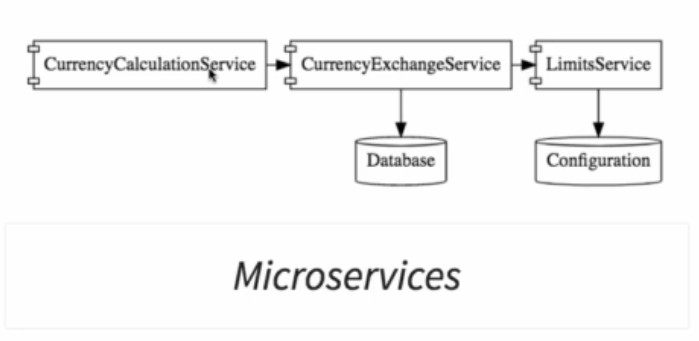
*//@FeignClient(name = "currency-exchange-service", url = "localhost:8000")  
//@FeignClient(name = "currency-exchange-service")*@FeignClient*(*name = "netflix-zuul-api-gateway-server"*)*@RibbonClient*(*name = "currency-exchange-service"*)*public interface CurrencyExchangeServiceProxy *{  
  
 //@GetMapping("/currency-exchange/from/{from}/to/{to}")* @GetMapping*(*"/currency-exchange-service/currency-exchange/from/{from}/to/{to}"*)* public CurrencyConversionBean retrieveExchangeValue  
 *(*@PathVariable*(*"from"*)* final String from, @PathVariable*(*"to"*)* final String to*)*;  
*}*

<http://localhost:8765/currency-conversion-service/currency-converter-feign/from/USD/to/INR/quantity/9>

**scenario: Setting up Zuul API gateway between microservices**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/094c567751109cea464344f39f0a655eece5064b>

**Use case: Introduction to Distributed Tracing**



*<*dependency*>  
 <*groupId*>*org.springframework.cloud*</*groupId*>  
 <*artifactId*>*spring-cloud-starter-sleuth*</*artifactId*>  
</*dependency*>*

@SpringBootApplication  
@EnableZuulProxy  
@EnableDiscoveryClient  
public class NetflixZuulApiGatewayServerApplication *{* public static void main*(*final String*[]* args*) {* SpringApplication.*run(*NetflixZuulApiGatewayServerApplication.class, args*)*;  
 *}* @Bean  
 public Sampler defaultSampler*(){* return Sampler.*ALWAYS\_SAMPLE*;  
 *}  
  
}*

@SpringBootApplication  
@EnableFeignClients*(*"com.in28min.ms.currencyconversionservice"*)*@EnableDiscoveryClient  
public class CurrencyConversionServiceApplication *{* public static void main*(*final String*[]* args*) {* SpringApplication.*run(*CurrencyConversionServiceApplication.class, args*)*;  
 *}* @Bean  
 public Sampler defaultSampler*(){* return Sampler.*ALWAYS\_SAMPLE*;  
 *}  
  
}*

@SpringBootApplication  
@EnableDiscoveryClient  
public class CurrencyExchangeServiceApplication *{* public static void main*(*final String*[]* args*) {* SpringApplication.*run(*CurrencyExchangeServiceApplication.class, args*)*;  
 *}* @Bean  
 public Sampler defaultSampler*(){* return Sampler.*ALWAYS\_SAMPLE*;  
 *}  
  
}*

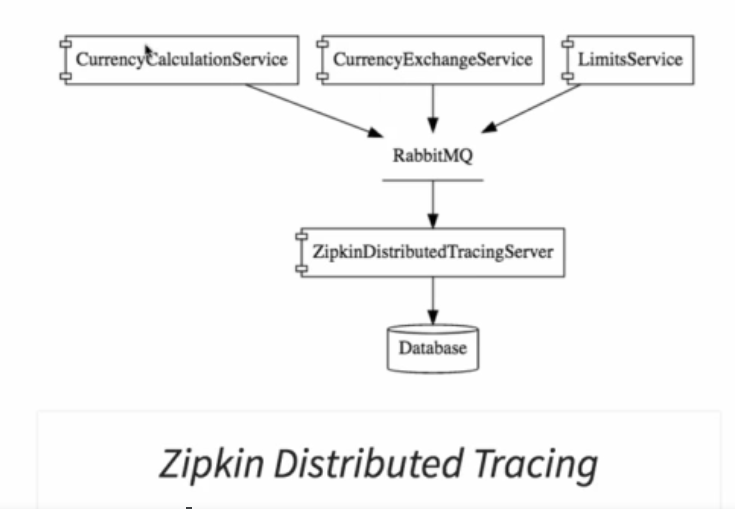
<http://localhost:8100/currency-converter-feign/from/USD/to/INR/quantity/9>

2022-01-20 20:42:09.458 INFO [netflix-zuul-api-gateway-server,c852661e8cc0e099,93bf5055fba54dfb,true] 7536 --- [nio-8765-exec-6] c.i.m.n.filter.ZuulLoggingFilter : request -> org.springframework.cloud.netflix.zuul.filters.pre.Servlet30RequestWrapper@5001ebb request uri -> /currency-exchange-service/currency-exchange/from/USD/to/INR

2022-01-20 20:42:09.473 INFO [currency-exchange-service,c852661e8cc0e099,a10ebcd073fa848e,true] 1244 --- [nio-8000-exec-2] c.i.m.c.c.CurrencyExchangeController : com.in28min.ms.currencyexchangeservice.bean.ExchangeValue@1119183a

2022-01-20 20:42:09.473 INFO [currency-conversion-service,c852661e8cc0e099,c852661e8cc0e099,true] 5048 --- [nio-8100-exec-7] c.i.m.c.c.CurrencyConversionController : com.in28min.ms.currencyconversionservice.bean.CurrencyConversionBean@5e01e2a7

**Introduction to Distributed Tracing with Zipkin**



**Installing Rabbit MQ**

1. Download Erlang/OTP
2. Download Rabbit MQ

**Setting up Distributed Tracing with Zipkin**

Download Zipkin Jar

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

**SET RABBIT\_URI=amqp://localhost**

**java -jar zipkin-server-2.23.16-exec.jar**

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

**connecting microservices to zipkin**

*<*dependency*>  
 <*groupId*>*org.springframework.cloud*</*groupId*>  
 <*artifactId*>*spring-cloud-sleuth-zipkin*</*artifactId*>  
</*dependency*>  
<*dependency*>  
 <*groupId*>*org.springframework.cloud*</*groupId*>  
 <*artifactId*>*spring-cloud-starter-bus-amqp*</*artifactId*>  
</*dependency*>*

**scenario: connecting microservices to zipkin**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/4695e71a18a0bd98666ff8428361208c156931ed>

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/acb5d975bba1064ca83e1396f9f1c47ebd2157b9>

**Understanding the need of Spring cloud BUS**

**Implementing spring cloud bus**

**scenario: Implementing Spring Cloud Bus**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/91ab04eb3352b66f95b7be86759fca504b9542d8>

**Fault Tolerance with Hystrix**

*<*dependency*>  
 <*groupId*>*org.springframework.cloud*</*groupId*>  
 <*artifactId*>*spring-cloud-starter-netflix-hystrix*</*artifactId*>  
</*dependency*>*

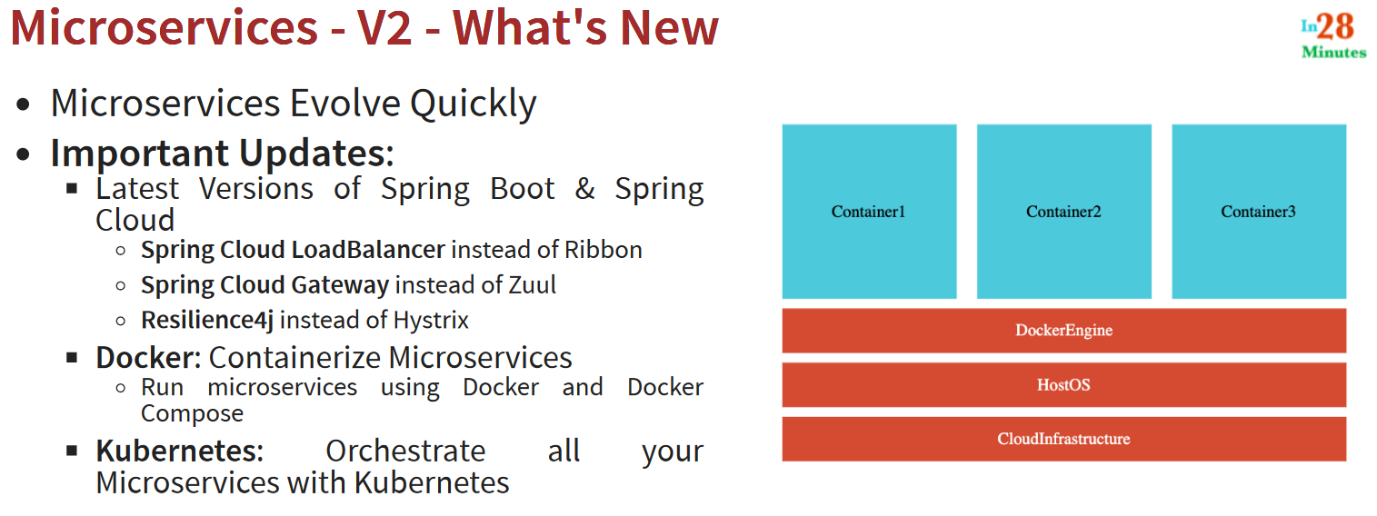
@SpringBootApplication  
@EnableHystrix  
public class LimitsServiceApplication *{* public static void main*(*final String*[]* args*) {* SpringApplication.*run(*LimitsServiceApplication.class, args*)*;  
 *}  
  
}*

@GetMapping*(*"/fault-tolerance-example"*)*@HystrixCommand*(*fallbackMethod="fallbackRetrieveConfiguration"*)*public LimitConfiguration retrieveConfigurations*(){* throw new RuntimeException*(*"Service Not available"*)*;  
*}*public LimitConfiguration fallbackRetrieveConfiguration*() {* return new LimitConfiguration*(*555, 555*)*;  
*}*

**scenario: Fault Tolerance with Hystrix**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/ba163fa4b2f95c21dc1a05b177bad231f1943d53>

**Section 6: Microservices with Spring cloud version 2**



|  |  |
| --- | --- |
| **Services** | **Port** |
| limits-service-v2 | 8080 |
| spring-cloud-config-server-v2 | 8888 |
| currency-exchange-service-v2 | 8000, 8001, 8002 … … … |
| currency-conversion-service-v2 | 8100, 8101, 8102 … … … |
| netflix-eureka-naming-server | 8761 |
| netflix-zuul-api-gateway-server | 8765 |
| Zipkin | 9411 |

**simple limits-service to read data from properties file**

**limits-service-V2**

*<*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-config*</*artifactId*>  
</*dependency*>*

server.port=8080  
spring.application.name=limits-service-v2  
spring.config.import=optional:configserver:http://localhost:8888  
limits-service-v2.maximum=997  
limits-service-v2.minimum=3

public class Limits *{* private int maximum;  
 private int minimum;

@Component  
@ConfigurationProperties*(*"limits-service-v2"*)*public class Configuration *{* private int maximum;  
 private int minimum;

@RestController  
public class LimitsController *{* @Autowired  
 private Configuration configuration;  
  
 @GetMapping*(*"/limits"*)* public Limits retrieveLimits*() {  
 //return new Limits(1000,1);* return new Limits*(*this.configuration.getMaximum*()*,  
 this.configuration.getMinimum*())*;  
 *}  
  
}*

**Use case: simple limits-service to read data from properties file v2**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/d8c6fc7017d321550feea8378a4403072fb270d8>

**Use case: Spring Cloud Config Service**

**spring-cloud-config-server-v2**

**Use case: create local git repository**

*<*dependency*>  
 <*groupId*>*org.springframework.boot*</*groupId*>  
 <*artifactId*>*spring-boot-starter-actuator*</*artifactId*>  
</*dependency*>  
<*dependency*>  
 <*groupId*>*org.springframework.cloud*</*groupId*>  
 <*artifactId*>*spring-cloud-config-server*</*artifactId*>  
</*dependency*>*

server.port=8888  
spring.application.name=spring-cloud-config-server-v2  
spring.cloud.config.server.git.uri=file:///C:/dev\_env/intellij\_projects/udemy\_master\_ms\_with\_spring\_boot\_cloud/git-localconfig-repo-v2

@SpringBootApplication  
@EnableConfigServer  
public class SpringCloudConfigServerV2Application *{* public static void main*(*final String*[]* args*) {* SpringApplication.*run(*SpringCloudConfigServerV2Application.class, args*)*;  
 *}  
  
}*

limits-service-v2.maximum=996  
limits-service-v2.minimum=4

**scenario: set up local git repository**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/991bce8fccd70534ff8508bf4422e9eac0d8906d>

**scenario: configure profiles for limits-service**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/98030504b380d3f5fff7ff3a6bb5ac3f03eecbde>

**Use case: Introduction to currency conversion and currency exchange service**

**currency-exchange-service-v2**

**scenario: Implementing currency-exchange-service-v2 with jpa and h2 database**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/2ba3da6ecd6ad22e3a6ad0b05fc971a930dd83e2>

**currency-conversion-service-v2**

**scenario: Initializing and invoking currency-exchange-service from currency-conversion-service**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/49c4976dbff939ce38b0a46cea9bee31d9809e4b>

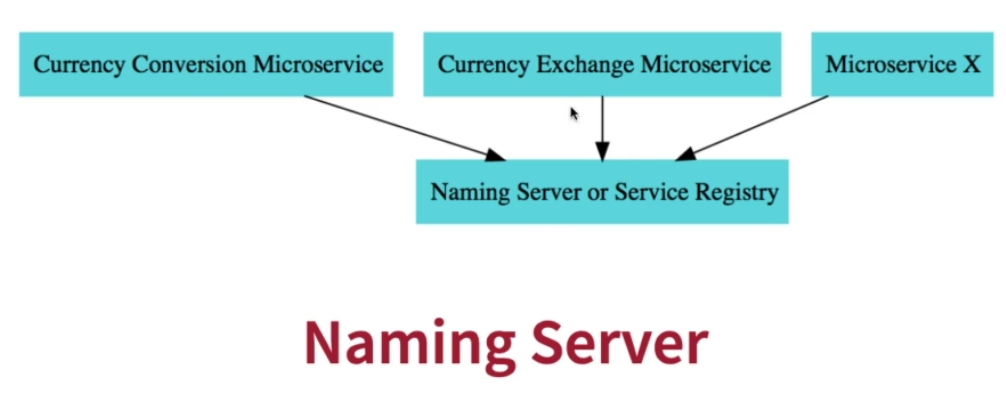
**Feign Rest Client as service invocation (to replace RestTemplate)**

**scenario: Feign Rest Client as service invocation**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/ca40a2fc7f4f7f1dd6709b3b383affed445d239e>

**Understanding need of naming server**

**netflix-eureka-naming-server-v2**



*<*dependency*>  
 <*groupId*>*org.springframework.boot*</*groupId*>  
 <*artifactId*>*spring-boot-starter-actuator*</*artifactId*>  
</*dependency*>  
<*dependency*>  
 <*groupId*>*org.springframework.cloud*</*groupId*>  
 <*artifactId*>*spring-cloud-starter-netflix-eureka-server*</*artifactId*>  
</*dependency*>*

server.port=8761  
spring.application.name=netflix-eureka-naming-server-v2  
eureka.client.register-with-eureka=false  
eureka.client.fetch-registry=false

@SpringBootApplication  
@EnableEurekaServer  
public class NetflixEurekaNamingServerV2Application *{* public static void main*(*final String*[]* args*) {* SpringApplication.*run(*NetflixEurekaNamingServerV2Application.class, args*)*;  
 *}  
  
}*

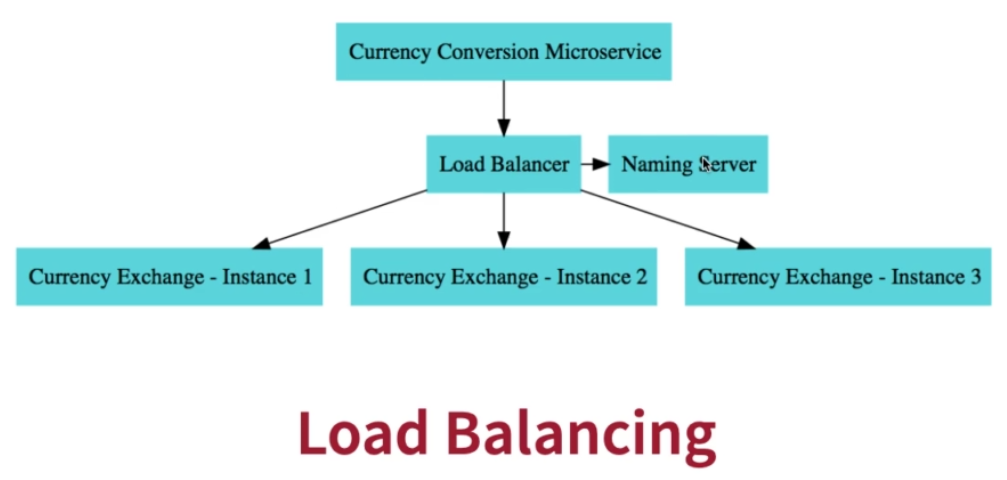
**scenario: setting of Eureka Naming Server**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/25c16e3f52cbc78f7c8322057a98e8c3dd86925b>

**scenario: Connect currency-exchange-service-v2 and currency-conversion-service-v2 with Eureka Naming Server**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/f28b4b24aaf08cc3a620b0a6ad71cd2d5ac60045>

**Load Balancing**



*//@FeignClient(name="currency-exchange-service-v2", url="localhost:8000")*@FeignClient*(*name="currency-exchange-service-v2"*)*public interface CurrencyExchangeProxy *{* @GetMapping*(*"/currency-exchange/from/{from}/to/{to}"*)* public CurrencyConversion retrieveExchangeValue*(* @PathVariable*(*"from"*)* String from,  
 @PathVariable*(*"to"*)* String to*)*;  
  
*}*

**scenario: Distributed / Load Balancing with Eureka, Feign, and Spring Cloud LoadBalancer**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/425bf9a4c7037ea497cb89604403a2f065143993>

**Spring Cloud API Gateway**

**spring-cloud-api-gateway-server-v2**

*<*dependency*>  
 <*groupId*>*org.springframework.cloud*</*groupId*>  
 <*artifactId*>*spring-cloud-starter-gateway*</*artifactId*>  
</*dependency*>  
<*dependency*>  
 <*groupId*>*org.springframework.cloud*</*groupId*>  
 <*artifactId*>*spring-cloud-starter-netflix-eureka-client*</*artifactId*>  
</*dependency*>*

server.port=8765  
spring.application.name=spring-cloud-api-gateway-server-v2  
eureka.client.service-url.default-zone=http://localhost:8761/eureka

**scenario: setting up Spring cloud API gateway**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/339ad2282c67ed84392fb8f05a72bbde73d28295>

**Enabling Discovery Locator with Eureka for Spring cloud gateway**

server.port=8765  
spring.application.name=spring-cloud-api-gateway-server-v2  
eureka.client.service-url.default-zone=http://localhost:8761/eureka  
spring.cloud.gateway.discovery.locator.enabled=true  
spring.cloud.gateway.discovery.locator.lowerCaseServiceId=true

**scenario: Enabling Discovery Locator with Eureka for Spring cloud gateway**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/90d4b2c6e402f5c33df4212576d7c36c2369faad>

**Exploring Routes with Spring Cloud Gateway**

import org.springframework.cloud.gateway.route.RouteLocator;  
import org.springframework.cloud.gateway.route.builder.RouteLocatorBuilder;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
  
@Configuration  
public class ApiGatewayConfiguration *{* @Bean  
 public RouteLocator gatewayRouter*(*final RouteLocatorBuilder builder*) {* return builder.routes*()* .route*(*p -> p  
 .path*(*"/get"*)* .filters*(*f -> f  
 .addRequestHeader*(*"MyHeader", "MyURI"*)* .addRequestParameter*(*"Param", "MyValue"*))* .uri*(*"http://httpbin.org:80"*))* .route*(*p -> p.path*(*"/currency-exchange/\*\*"*)* .uri*(*"lb://currency-exchange-service-v2"*))* .route*(*p -> p.path*(*"/currency-conversion/\*\*"*)* .uri*(*"lb://currency-conversion-service-v2"*))* .route*(*p -> p.path*(*"/currency-conversion-feign/\*\*"*)* .uri*(*"lb://currency-conversion-service-v2"*))* .route*(*p -> p.path*(*"/currency-conversion-new/\*\*"*)* .filters*(*f -> f.rewritePath*(* "/currency-conversion-new/(?<segment>.\*)",  
 "/currency-conversion-feign/${segment}"*))* .uri*(*"lb://currency-conversion-service-v2"*))* .build*()*;  
 *}  
}*

server.port=8765  
spring.application.name=spring-cloud-api-gateway-server-v2  
eureka.client.service-url.default-zone=http://localhost:8761/eureka  
*#spring.cloud.gateway.discovery.locator.enabled=true  
#spring.cloud.gateway.discovery.locator.lowerCaseServiceId=true*

<http://localhost:8765/currency-exchange-service-v2/currency-exchange/from/USD/to/INR> (Normal)

<http://localhost:8765/currency-exchange/from/USD/to/INR> (Route)

<http://localhost:8765/currency-conversion-service-v2/currency-conversion/from/USD/to/INR/quantity/9> (Normal)

<http://localhost:8765/currency-conversion/from/USD/to/INR/quantity/9> (Route)

<http://localhost:8765/currency-conversion-service-v2/currency-conversion-feign/from/USD/to/INR/quantity/9> (Normal)

<http://localhost:8765/currency-conversion-feign/from/USD/to/INR/quantity/9> (Route)

<http://localhost:8765/currency-conversion-new/from/USD/to/INR/quantity/9> (Route)

**scenario: Exploring Routes with Spring Cloud Gateway**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/dd0c721ab1fc443e2c43b54ef473436813339f4e>

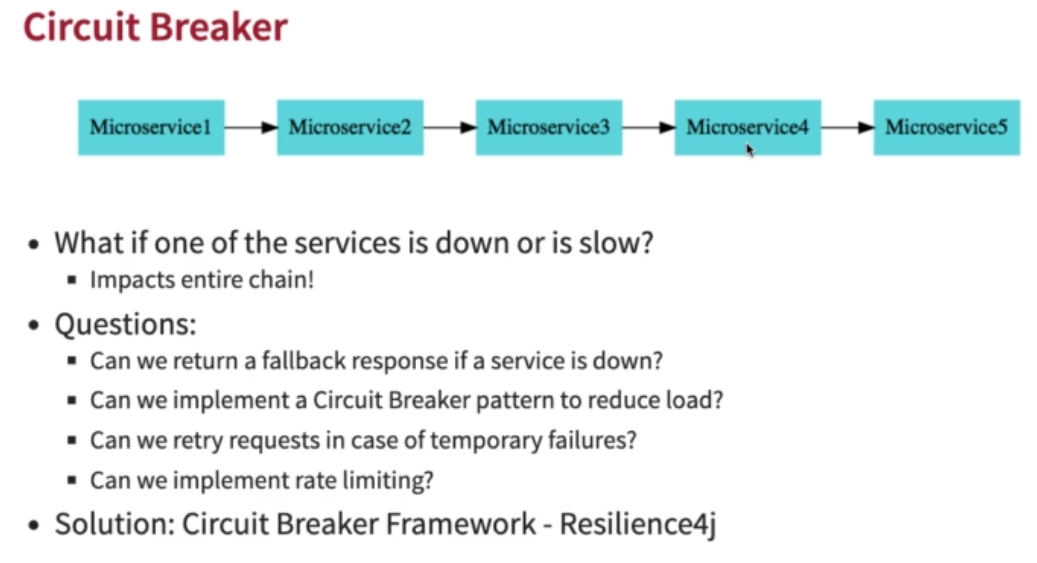
**Implementing Spring Cloud Gateway Logging Filtering**

import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.cloud.gateway.filter.GatewayFilterChain;  
import org.springframework.cloud.gateway.filter.GlobalFilter;  
import org.springframework.stereotype.Component;  
import org.springframework.web.server.ServerWebExchange;  
  
import reactor.core.publisher.Mono;  
  
@Component  
public class LoggingFilter implements GlobalFilter *{* private final Logger logger = LoggerFactory.*getLogger(*LoggingFilter.class*)*;  
  
 @Override  
 public Mono*<*Void*>* filter*(*final ServerWebExchange exchange,  
 final GatewayFilterChain chain*) {* this.logger.info*(*"Path of the request received -> {}",  
 exchange.getRequest*()*.getPath*())*;  
 return chain.filter*(*exchange*)*;  
 *}  
  
}*

**scenario: Implementing Spring Cloud Gateway Logging Filtering**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/ebc0b20618e21ac03f8c522e57b2299d89c212fb>

**Use case: Getting started with Circuit Breaker**



*<*dependency*>  
 <*groupId*>*org.springframework.boot*</*groupId*>  
 <*artifactId*>*spring-boot-starter-aop*</*artifactId*>  
</*dependency*>  
<*dependency*>  
 <*groupId*>*io.github.resilience4j*</*groupId*>  
 <*artifactId*>*resilience4j-spring-boot2*</*artifactId*>  
</*dependency*>*

@RestController  
public class CircuitBreakerController *{* private final Logger logger = LoggerFactory.*getLogger(*CircuitBreakerController.class*)*;  
  
 @GetMapping*(*"/sample-api"*)* @Retry*(*name="sample-api", fallbackMethod = "hardcodedResponseMethod"*)* public String sampleApi*() {* this.logger.info*(*"Sample api call received"*)*;  
 final ResponseEntity*<*String*>* forEntity = new RestTemplate*()*.getForEntity*(*"http://localhost:8080/some-dummy-url", String.class*)*;  
 return forEntity.getBody*()*;  
 *//return "sample-api";  
 }* public String hardcodedResponseMethod*(*final Exception ex*) {* return "fallback-response";  
 *}  
  
}*

server.port=8000  
spring.application.name=currency-exchange-service-v2  
spring.jpa.show-sql=true  
spring.datasource.url=jdbc:h2:mem:testdb  
spring.h2.console.enabled=true  
spring.config.import=optional:configserver:  
spring.jpa.defer-datasource-initialization=true  
eureka.client.service-url.default-zone=http://localhost:8761/eureka  
resilience4j.retry.instances.sample-api.maxAttempts=5  
resilience4j.retry.instances.sample-api.waitDuration=1s  
resilience4j.retry.instances.sample-api.enableExponentialBackoff=true

**scenario: Playing with Resilience4j - Retry and Fallback Methods**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/4faa65ca288e47562918660868a6230ebffc0715>

**Playing with Circuit Breaker of Resilience4j**

import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.client.RestTemplate;  
  
import io.github.resilience4j.circuitbreaker.annotation.CircuitBreaker;  
  
@RestController  
public class CircuitBreakerController *{* private final Logger logger = LoggerFactory.*getLogger(*CircuitBreakerController.class*)*;  
  
 @GetMapping*(*"/sample-api"*)  
 //@Retry(name="sample-api", fallbackMethod = "hardcodedResponseMethod")* @CircuitBreaker*(*name = "default", fallbackMethod = "hardcodedResponseMethod"*)* public String sampleApi*() {* this.logger.info*(*"Sample api call received"*)*;  
 final ResponseEntity*<*String*>* forEntity = new RestTemplate*()*.getForEntity*(*"http://localhost:8080/some-dummy-url", String.class*)*;  
 return forEntity.getBody*()*;  
 *//return "sample-api";  
 }* public String hardcodedResponseMethod*(*final Exception ex*) {* return "fallback-response";  
 *}  
  
}*

server.port=8000  
spring.application.name=currency-exchange-service-v2  
spring.jpa.show-sql=true  
spring.datasource.url=jdbc:h2:mem:testdb  
spring.h2.console.enabled=true  
spring.config.import=optional:configserver:  
spring.jpa.defer-datasource-initialization=true  
eureka.client.service-url.default-zone=http://localhost:8761/eureka  
resilience4j.retry.instances.sample-api.maxAttempts=5  
resilience4j.retry.instances.sample-api.waitDuration=1s  
resilience4j.retry.instances.sample-api.enableExponentialBackoff=true  
*#resilience4j.circuitbreaker.instances.default.failureRateThreshold=90*

**scenario: Playing with Circuit Breaker of Resilience4j**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/e1645284589b44eb8061106709e8009b7f08de71>

**Exploring Rate Limiting and BulkHead Features of Resilience4j**

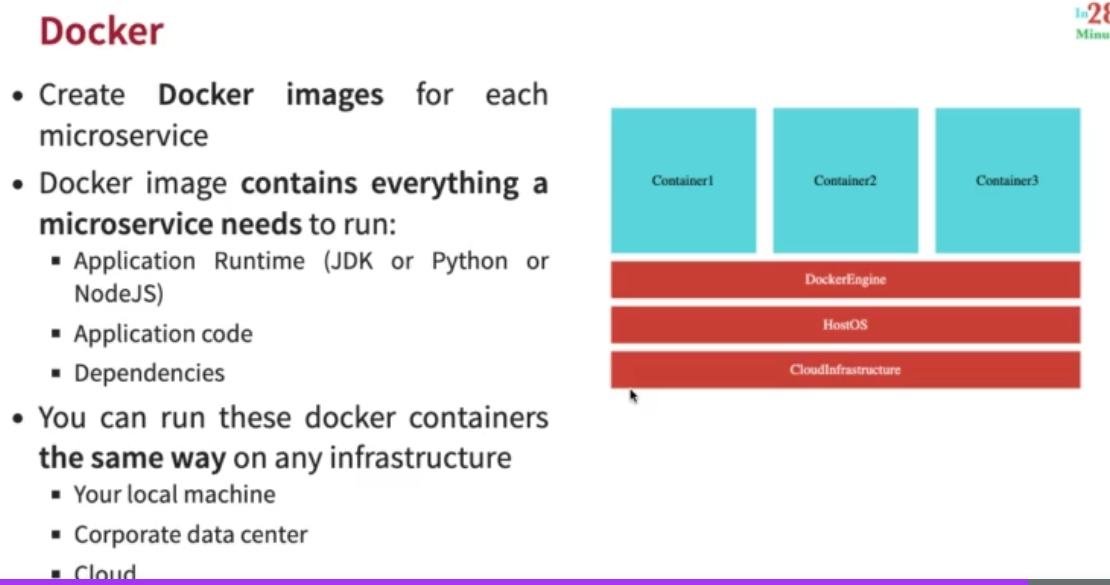
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
import io.github.resilience4j.bulkhead.annotation.Bulkhead;  
  
@RestController  
public class CircuitBreakerController *{* private final Logger logger = LoggerFactory.*getLogger(*CircuitBreakerController.class*)*;  
  
 @GetMapping*(*"/sample-api"*)  
 //@Retry(name="sample-api", fallbackMethod = "hardcodedResponseMethod")  
 //@CircuitBreaker(name = "default", fallbackMethod = "hardcodedResponseMethod")  
 //@RateLimiter(name="default")* @Bulkhead*(*name="default"*)* public String sampleApi*() {* this.logger.info*(*"Sample api call received"*)*;  
 *//final ResponseEntity<String> forEntity = new RestTemplate().getForEntity("http://localhost:8080/some-dummy-url", String.class);  
 //return forEntity.getBody();* return "sample-api";  
 *}* public String hardcodedResponseMethod*(*final Exception ex*) {* return "fallback-response";  
 *}  
  
}*

server.port=8000  
spring.application.name=currency-exchange-service-v2  
spring.jpa.show-sql=true  
spring.datasource.url=jdbc:h2:mem:testdb  
spring.h2.console.enabled=true  
spring.config.import=optional:configserver:  
spring.jpa.defer-datasource-initialization=true  
eureka.client.service-url.default-zone=http://localhost:8761/eureka  
resilience4j.retry.instances.sample-api.maxAttempts=5  
resilience4j.retry.instances.sample-api.waitDuration=1s  
resilience4j.retry.instances.sample-api.enableExponentialBackoff=true  
*#resilience4j.circuitbreaker.instances.default.failureRateThreshold=90*resilience4j.ratelimiter.instances.default.limitForPeriod=2  
resilience4j.ratelimiter.instances.default.limitRefreshPeriod=10s  
resilience4j.bulkhead.instances.default.maxConcurrentCalls=10  
resilience4j.bulkhead.instances.sample-api.maxConcurrentCalls=10

**scenario: Exploring Rate Limiting and BulkHead Features of Resilience4j**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/615f8db07eaae27ba4448663223e93cf843a0f27>

**Section 7: Docker and Microservices using Spring Boot and Spring Cloud version 2**



Install Docker

After that open powershell

>docket --version //to check the version

>docker run in28min/todo-rest-api-h2:1.0.0.RELEASE

// downloaded a new docker image from hub

[https://hub.docker.com/r/in28min/todo-rest-api-h2](https://hub.docker.com/r/in28min/todo-rest-api-h2:1.0.0.RELEASE)

>docker run -p 5000:5000 in28min/todo-rest-api-h2:1.0.0.RELEASE

//to expose the rest end point of the application on port 5000

>docker run -p 5000:5000 in28min/todo-rest-api-h2:1.0.0.RELEASE

// to expose the rest end point of the application on port 5001

[localhost:5000/hello-world-bean](http://localhost:5000/hello-world-bean)

>docker run -p 5000:5000 -d in28min/todo-rest-api-h2:1.0.0.RELEASE

>docker logs 92532cc2e65f4bc73cb43582a669a76f2bd3574a7643f1ad9be505cc39aa8956

//to see logs

>docker container ls

//to see all the running container

> docker images

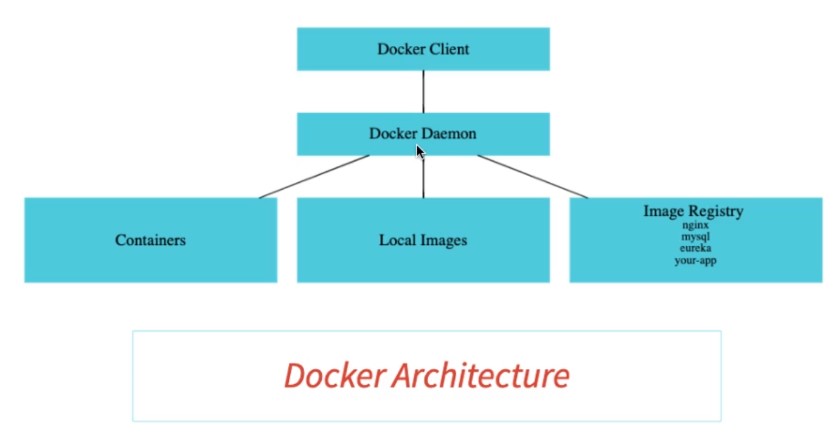
//show all the images present in local machine

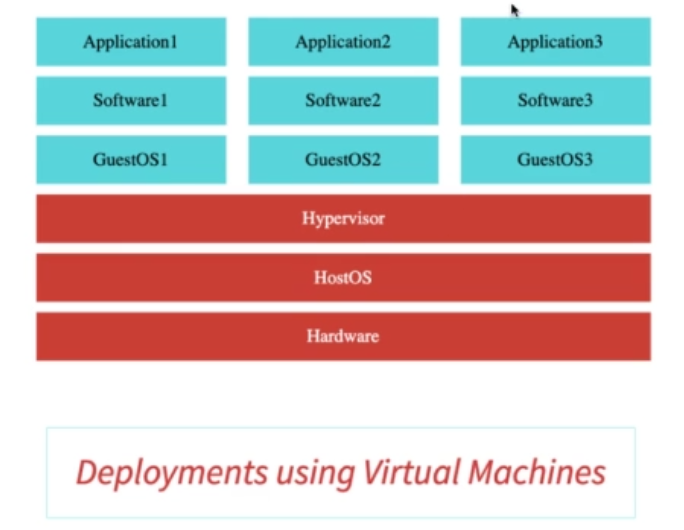
> docker container ls -a

//show all stop and running container

>docker container stop <docker container id>

//To stop the docker

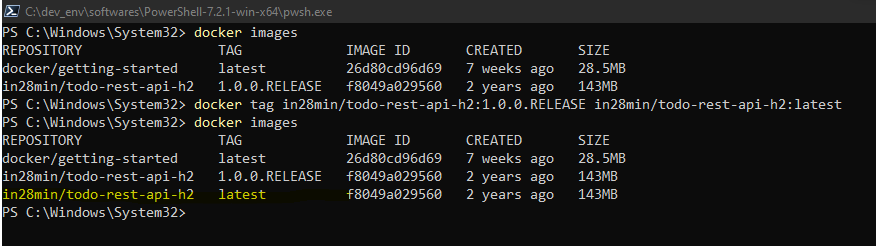




**playing with docker images**

>docker tag in28min/todo-rest-api-h2:1.0.0.RELEASE in28min/todo-rest-api-h2:latest

//create tag of “in28min/todo-rest-api-h2” with latest release



>docker pull mysql

//to pull latest version of mysql

> docker search mysql

//show all mysql images with all other database details

> docker image history f8049a029560

//show history of that image id

> docker image inspect f8049a029560

//inspect for that image id

> docker image remove f8049a029560

//to remove the image from local machine

**playing with docker container**

>docker container pause 92532cc2e65f4bc73cb43582a669a76f2bd3574a7643f1ad9be505cc39aa8956

>docker container pause 92532

//to pause a container

>docker container unpause 92532

//to unpause a container

>docker container stop 92532

//to stop(gracefully) a container

>docker container kill 92532

//to stop(immediately) a container

>docker container inspect 92532

//to inspect a container

>docker container prune

//to remove all stop container

>docker run -p 5000:5000 -d –restart=always in28min/todo-rest-api-h2:1.0.0.RELEASE

**playing with docker commands**

> docker events

//show all events happen inside docker

>docker top

//show the top processes which is running

>docker top <container id>

//show all the processes in that container

>docker stats

//show all the stats regarding the running container

//show memory cpu all details

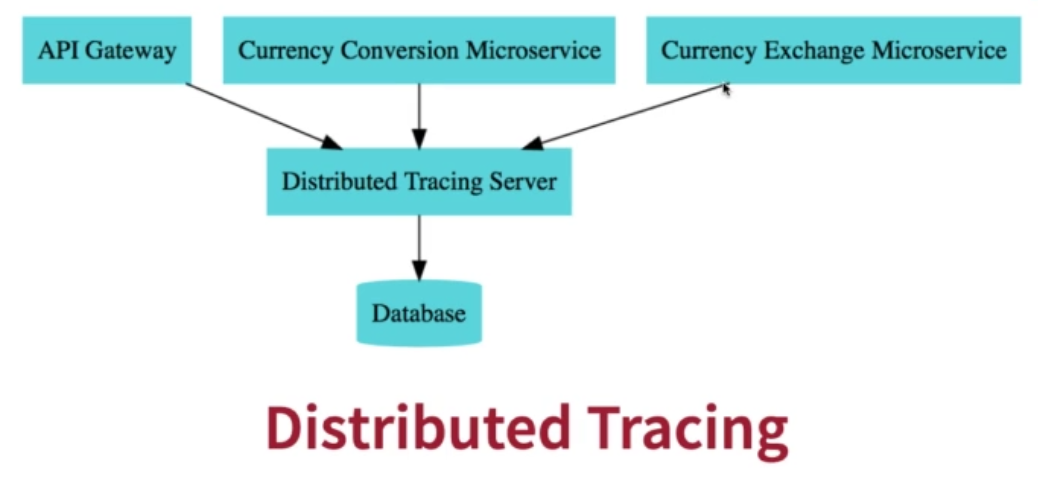
>docker system df

//show all details (volumes)

>docker run -m 512m --cpu-quota 50000

**scenario: Basic docker commands**

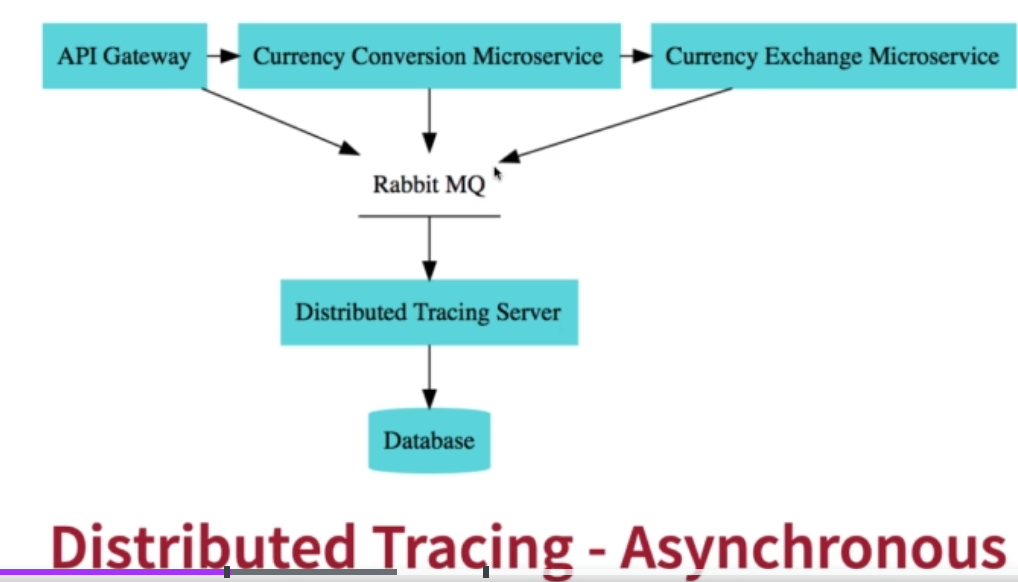
commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/62fff21af8d9acef36fe8470d60994ea7390e63a>



**Launching Zipkin Container Using Docker**

> docker run -p 9411:9411 openzipkin/zipkin:2.23.16

**Connecting currency-exchange-service-v2 with Docker**

****

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

**scenario: Launching Zipkin container using Docker**

**scenario: connecting our microservices with Docker**

**commit:** <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/b462b56dd1f53e8fbc653acf39c86b1eaefa828a>

**creating Docker container image for currency-exchange-service-v2**

*<*build*>  
 <*plugins*>  
 <*plugin*>  
 <*groupId*>*org.springframework.boot*</*groupId*>  
 <*artifactId*>*spring-boot-maven-plugin*</*artifactId*>  
 <*configuration*>  
 <*image*>  
 <*name*>*bibhu082/mmv2-${project.artifactId}:${project.version}*</*name*>  
 </*image*>  
 <*pullPolicy*>*IF\_NOT\_PRESENT*</*pullPolicy*>  
 </*configuration*>  
 </*plugin*>  
 </*plugins*>  
</*build*>*

//bibhu082 is my docker id

//To create a docker Image build command

>mvn spring-boot:build-image -DskipTests

//To run our custom docker image

docker run -p 8000:8000 bibhu082/mmv2-currency-exchange-service-v2:0.0.1-SNAPSHOT

**scenario: creating Docker container image for currency-exchange-service-v2**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/ee8d02d548805c288f0f1b807bfb6830ca7f3130>

**started with Docker compose for currency-exchange-service-v2**

>docker-compose –version

//check docker compose version

**docker-compose.yaml**

version: '3.7'  
  
services:  
 currency-exchange-service-v2:  
 image: bibhu082/mmv2-currency-exchange-service-v2:0.0.1-SNAPSHOT  
 mem\_limit: 700m  
 ports:  
 - "8000:8000"  
 networks:  
 - currency-exchange-network-v2  
  
networks:  
 currency-exchange-network-v2:

>docker-compose up

//command to run docker-compose file

**scenario: started with Docker compose for currency-exchange-service-v2**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/8c57d99a890b5a9f37580de9741edacc1992a10d>

**started with Docker compose for other microservices**

**netflix-eureka-naming-server-v2**

*<*build*>  
 <*plugins*>  
 <*plugin*>  
 <*groupId*>*org.springframework.boot*</*groupId*>  
 <*artifactId*>*spring-boot-maven-plugin*</*artifactId*>  
 <*configuration*>  
 <*image*>  
 <*name*>*bibhu082/mmv2-${project.artifactId}:${project.version}*</*name*>  
 </*image*>  
 <*pullPolicy*>*IF\_NOT\_PRESENT*</*pullPolicy*>  
 </*configuration*>  
 </*plugin*>  
 </*plugins*>  
</*build*>*

>mvn spring-boot:build-image -DskipTests

bibhu082/mmv2-netflix-eureka-naming-server-v2:0.0.1-SNAPSHOT

**docker-compose.yaml**

version: '3.7'  
services:  
 currency-exchange-service-v2:  
 image: bibhu082/mmv2-currency-exchange-service-v2:0.0.1-SNAPSHOT  
 mem\_limit: 700m  
 ports:  
 - "8000:8000"  
 networks:  
 - currency-network-v2  
 environment:  
 eureka.client.serviceUrl.defaultZone: http://netflix-eureka-naming-server-v2:8761/eureka/  
 depends\_on:  
 - netflix-eureka-naming-server-v2  
 netflix-eureka-naming-server-v2:  
 image: bibhu082/mmv2-netflix-eureka-naming-server-v2:0.0.1-SNAPSHOT  
 mem\_limit: 700m  
 ports:  
 - "8761:8761"  
 networks:  
 - currency-network-v2  
networks:  
 currency-network-v2:

**currency-conversion-service-v2**

*<*build*>  
 <*plugins*>  
 <*plugin*>  
 <*groupId*>*org.springframework.boot*</*groupId*>  
 <*artifactId*>*spring-boot-maven-plugin*</*artifactId*>  
 <*configuration*>  
 <*image*>  
 <*name*>*bibhu082/mmv2-${project.artifactId}:${project.version}*</*name*>  
 </*image*>  
 <*pullPolicy*>*IF\_NOT\_PRESENT*</*pullPolicy*>  
 </*configuration*>  
 </*plugin*>  
 </*plugins*>  
</*build*>*

>mvn spring-boot:build-image -DskipTests

bibhu082/mmv2-currency-conversion-service-v2:0.0.1-SNAPSHOT

**docker-compose.yaml**

version: '3.7'  
services:  
 currency-exchange-service-v2:  
 image: bibhu082/mmv2-currency-exchange-service-v2:0.0.1-SNAPSHOT  
 mem\_limit: 700m  
 ports:  
 - "8000:8000"  
 networks:  
 - currency-network-v2  
 environment:  
 eureka.client.serviceUrl.defaultZone: http://netflix-eureka-naming-server-v2:8761/eureka/  
 depends\_on:  
 - netflix-eureka-naming-server-v2  
 currency-conversion-service-v2:  
 image: bibhu082/mmv2-currency-conversion-service-v2:0.0.1-SNAPSHOT  
 mem\_limit: 700m  
 ports:  
 - "8100:8100"  
 networks:  
 - currency-network-v2  
 environment:  
 eureka.client.serviceUrl.defaultZone: http://netflix-eureka-naming-server-v2:8761/eureka/  
 depends\_on:  
 - netflix-eureka-naming-server-v2  
 netflix-eureka-naming-server-v2:  
 image: bibhu082/mmv2-netflix-eureka-naming-server-v2:0.0.1-SNAPSHOT  
 mem\_limit: 700m  
 ports:  
 - "8761:8761"  
 networks:  
 - currency-network-v2  
networks:  
 currency-network-v2:

**spring-cloud-api-gateway-server-v2**

*<*build*>  
 <*plugins*>  
 <*plugin*>  
 <*groupId*>*org.springframework.boot*</*groupId*>  
 <*artifactId*>*spring-boot-maven-plugin*</*artifactId*>  
 <*configuration*>  
 <*image*>  
 <*name*>*bibhu082/mmv2-${project.artifactId}:${project.version}*</*name*>  
 </*image*>  
 <*pullPolicy*>*IF\_NOT\_PRESENT*</*pullPolicy*>  
 </*configuration*>  
 </*plugin*>  
 </*plugins*>  
</*build*>*

>mvn spring-boot:build-image -DskipTests

bibhu082/mmv2-spring-cloud-api-gateway-server-v2:0.0.1-SNAPSHOT

**docker-compose.yaml**

version: '3.7'  
services:  
 currency-exchange-service-v2:  
 image: bibhu082/mmv2-currency-exchange-service-v2:0.0.1-SNAPSHOT  
 mem\_limit: 700m  
 ports:  
 - "8000:8000"  
 networks:  
 - currency-network-v2  
 environment:  
 eureka.client.serviceUrl.defaultZone: http://netflix-eureka-naming-server-v2:8761/eureka/  
 depends\_on:  
 - netflix-eureka-naming-server-v2  
 currency-conversion-service-v2:  
 image: bibhu082/mmv2-currency-conversion-service-v2:0.0.1-SNAPSHOT  
 mem\_limit: 700m  
 ports:  
 - "8100:8100"  
 networks:  
 - currency-network-v2  
 environment:  
 eureka.client.serviceUrl.defaultZone: http://netflix-eureka-naming-server-v2:8761/eureka/  
 depends\_on:  
 - netflix-eureka-naming-server-v2  
 spring-cloud-api-gateway-server-v2:  
 image: bibhu082/mmv2-spring-cloud-api-gateway-server-v2:0.0.1-SNAPSHOT  
 mem\_limit: 700m  
 ports:  
 - "8765:8765"  
 networks:  
 - currency-network-v2  
 environment:  
 eureka.client.serviceUrl.defaultZone: http://netflix-eureka-naming-server-v2:8761/eureka/  
 depends\_on:  
 - netflix-eureka-naming-server-v2  
 netflix-eureka-naming-server-v2:  
 image: bibhu082/mmv2-netflix-eureka-naming-server-v2:0.0.1-SNAPSHOT  
 mem\_limit: 700m  
 ports:  
 - "8761:8761"  
 networks:  
 - currency-network-v2  
networks:  
 currency-network-v2:

**scenario: started with Docker compose for other microservices**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/d30e0b084016e1aec510d5836795e53b29d40a88>

**Running Zipkin with docker composer**

**docker-compose.yaml**

version: '3.7'  
services:  
 currency-exchange-service-v2:  
 image: bibhu082/mmv2-currency-exchange-service-v2:0.0.1-SNAPSHOT  
 mem\_limit: 700m  
 ports:  
 - "8000:8000"  
 networks:  
 - currency-network-v2  
 environment:  
 eureka.client.serviceUrl.defaultZone: http://netflix-eureka-naming-server-v2:8761/eureka/  
 spring.zipkin.baseUrl: http://zipkin-server:9411/  
 depends\_on:  
 - netflix-eureka-naming-server-v2  
 currency-conversion-service-v2:  
 image: bibhu082/mmv2-currency-conversion-service-v2:0.0.1-SNAPSHOT  
 mem\_limit: 700m  
 ports:  
 - "8100:8100"  
 networks:  
 - currency-network-v2  
 environment:  
 eureka.client.serviceUrl.defaultZone: http://netflix-eureka-naming-server-v2:8761/eureka/  
 spring.zipkin.baseUrl: http://zipkin-server:9411/  
 depends\_on:  
 - netflix-eureka-naming-server-v2  
 spring-cloud-api-gateway-server-v2:  
 image: bibhu082/mmv2-spring-cloud-api-gateway-server-v2:0.0.1-SNAPSHOT  
 mem\_limit: 700m  
 ports:  
 - "8765:8765"  
 networks:  
 - currency-network-v2  
 environment:  
 eureka.client.serviceUrl.defaultZone: http://netflix-eureka-naming-server-v2:8761/eureka/  
 spring.zipkin.baseUrl: http://zipkin-server:9411/  
 depends\_on:  
 - netflix-eureka-naming-server-v2  
 netflix-eureka-naming-server-v2:  
 image: bibhu082/mmv2-netflix-eureka-naming-server-v2:0.0.1-SNAPSHOT  
 mem\_limit: 700m  
 ports:  
 - "8761:8761"  
 networks:  
 - currency-network-v2  
 zipkin-server:  
 image: openzipkin/zipkin:2.23.16  
 mem\_limit: 300m  
 ports:  
 - "9411:9411"  
 networks:  
 - currency-network-v2  
networks:  
 currency-network-v2:

**scenario: running zipkin with docker composer**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/eadfcce468c2b938b46ebcb52a1ebea3c1b9ab57>

**Running zipkin and RabbitMq with docker composer**

*<*dependency*>  
 <*groupId*>*org.springframework.amqp*</*groupId*>  
 <*artifactId*>*spring-rabbit*</*artifactId*>  
</*dependency*>*

**docker-compose.yaml**

version: '3.7'  
services:  
 currency-exchange-service-v2:  
 image: bibhu082/mmv2-currency-exchange-service-v2:0.0.1-SNAPSHOT  
 mem\_limit: 700m  
 ports:  
 - "8000:8000"  
 networks:  
 - currency-network-v2  
 environment:  
 eureka.client.serviceUrl.defaultZone: http://netflix-eureka-naming-server-v2:8761/eureka/  
 spring.zipkin.baseUrl: http://zipkin-server:9411/  
 RABBIT\_URI: amqp://guest:guest@rabbitmq:5672  
 spring.rabbitmq.host: rabbitmq  
 spring.zipkin.sender.type: rabbit  
 depends\_on:  
 - netflix-eureka-naming-server-v2  
 - rabbitmq  
 currency-conversion-service-v2:  
 image: bibhu082/mmv2-currency-conversion-service-v2:0.0.1-SNAPSHOT  
 mem\_limit: 700m  
 ports:  
 - "8100:8100"  
 networks:  
 - currency-network-v2  
 environment:  
 eureka.client.serviceUrl.defaultZone: http://netflix-eureka-naming-server-v2:8761/eureka/  
 spring.zipkin.baseUrl: http://zipkin-server:9411/  
 RABBIT\_URI: amqp://guest:guest@rabbitmq:5672  
 spring.rabbitmq.host: rabbitmq  
 spring.zipkin.sender.type: rabbit  
 depends\_on:  
 - netflix-eureka-naming-server-v2  
 - rabbitmq  
 spring-cloud-api-gateway-server-v2:  
 image: bibhu082/mmv2-spring-cloud-api-gateway-server-v2:0.0.1-SNAPSHOT  
 mem\_limit: 700m  
 ports:  
 - "8765:8765"  
 networks:  
 - currency-network-v2  
 environment:  
 eureka.client.serviceUrl.defaultZone: http://netflix-eureka-naming-server-v2:8761/eureka/  
 spring.zipkin.baseUrl: http://zipkin-server:9411/  
 RABBIT\_URI: amqp://guest:guest@rabbitmq:5672  
 spring.rabbitmq.host: rabbitmq  
 spring.zipkin.sender.type: rabbit  
 depends\_on:  
 - netflix-eureka-naming-server-v2  
 - rabbitmq  
 netflix-eureka-naming-server-v2:  
 image: bibhu082/mmv2-netflix-eureka-naming-server-v2:0.0.1-SNAPSHOT  
 mem\_limit: 700m  
 ports:  
 - "8761:8761"  
 networks:  
 - currency-network-v2  
 zipkin-server:  
 image: openzipkin/zipkin:2.23.16  
 mem\_limit: 300m  
 ports:  
 - "9411:9411"  
 networks:  
 - currency-network-v2  
 environment:  
 RABBIT\_URI: amqp://guest:guest@rabbitmq:5672  
 depends\_on:  
 - rabbitmq  
 restart: always *#Restart if there is a problem starting up* rabbitmq:  
 image: rabbitmq:3.9.13-management  
 mem\_limit: 300m  
 ports:  
 - "5672:5672"  
 - "15672:15672"  
 networks:  
 - currency-network-v2  
networks:  
 currency-network-v2:

**scenario: running zipkin and RabbitMq with docker composer**

commit: <https://github.com/bibhusprasad/017_Udemy_Master_Microservices_with_Spring_Boot_and_Spring_Cloud/commit/d63bb7214a458efea7cee2448ed74e9a50f06599>



**Section 8: Kubernetes with Microservices using Docker, Spring Boot and Spring Cloud version 2**



docker run -p 8080:8080 in28min/hello-world-rest-api:0.0.1.RELEASE

kubectl create deployment hello-world-rest-api --image=in28min/hello-world-rest-api:0.0.1.RELEASE

kubectl expose deployment hello-world-rest-api --type=LoadBalancer --port=8080

kubectl scale deployment hello-world-rest-api --replicas=3

kubectl delete pod hello-world-rest-api-58ff5dd898-62l9d

kubectl autoscale deployment hello-world-rest-api --max=10 --cpu-percent=70

kubectl edit deployment hello-world-rest-api #minReadySeconds: 15

kubectl set image deployment hello-world-rest-api hello-world-rest-api=in28min/hello-world-rest-api:0.0.2.RELEASE

gcloud container clusters get-credentials in28minutes-cluster --zone us-central1-a --project solid-course-258105

kubectl create deployment hello-world-rest-api --image=in28min/hello-world-rest-api:0.0.1.RELEASE

kubectl expose deployment hello-world-rest-api --type=LoadBalancer --port=8080

kubectl set image deployment hello-world-rest-api hello-world-rest-api=DUMMY\_IMAGE:TEST

kubectl get events --sort-by=.metadata.creationTimestamp

kubectl set image deployment hello-world-rest-api hello-world-rest-api=in28min/hello-world-rest-api:0.0.2.RELEASE

kubectl get events --sort-by=.metadata.creationTimestamp

kubectl get componentstatuses

kubectl get pods --all-namespaces

kubectl get events

kubectl get pods

kubectl get replicaset

kubectl get deployment

kubectl get service

kubectl get pods -o wide

kubectl explain pods

kubectl get pods -o wide

kubectl describe pod hello-world-rest-api-58ff5dd898-9trh2

kubectl get replicasets

kubectl get replicaset

kubectl scale deployment hello-world-rest-api --replicas=3

kubectl get pods

kubectl get replicaset

kubectl get events

kubectl get events --sort.by=.metadata.creationTimestamp

kubectl get rs

kubectl get rs -o wide

kubectl set image deployment hello-world-rest-api hello-world-rest-api=DUMMY\_IMAGE:TEST

kubectl get rs -o wide

kubectl get pods

kubectl describe pod hello-world-rest-api-85995ddd5c-msjsm

kubectl get events --sort-by=.metadata.creationTimestamp

kubectl set image deployment hello-world-rest-api hello-world-rest-api=in28min/hello-world-rest-api:0.0.2.RELEASE

kubectl get events --sort-by=.metadata.creationTimestamp

kubectl get pods -o wide

kubectl delete pod hello-world-rest-api-67c79fd44f-n6c7l

kubectl get pods -o wide

kubectl delete pod hello-world-rest-api-67c79fd44f-8bhdt

gcloud container clusters get-credentials in28minutes-cluster --zone us-central1-c --project solid-course-258105

docker login

docker push in28min/mmv2-currency-exchange-service:0.0.11-SNAPSHOT

docker push in28min/mmv2-currency-conversion-service:0.0.11-SNAPSHOT

kubectl create deployment currency-exchange --image=in28min/mmv2-currency-exchange-service:0.0.11-SNAPSHOT

kubectl expose deployment currency-exchange --type=LoadBalancer --port=8000

kubectl get svc

kubectl get services

kubectl get pods

kubectl get po

kubectl get replicaset

kubectl get rs

kubectl get all

kubectl create deployment currency-conversion --image=in28min/mmv2-currency-conversion-service:0.0.11-SNAPSHOT

kubectl expose deployment currency-conversion --type=LoadBalancer --port=8100

kubectl get svc --watch

kubectl get deployments

kubectl get deployment currency-exchange -o yaml >> deployment.yaml

kubectl get service currency-exchange -o yaml >> service.yaml

kubectl diff -f deployment.yaml

kubectl apply -f deployment.yaml

kubectl delete all -l app=currency-exchange

kubectl delete all -l app=currency-conversion

kubectl rollout history deployment currency-conversion

kubectl rollout history deployment currency-exchange

kubectl rollout undo deployment currency-exchange --to-revision=1

kubectl logs currency-exchange-9fc6f979b-2gmn8

kubectl logs -f currency-exchange-9fc6f979b-2gmn8

kubectl autoscale deployment currency-exchange --min=1 --max=3 --cpu-percent=5

kubectl get hpa

kubectl top pod

kubectl top nodes

kubectl get hpa

kubectl delete hpa currency-exchange

kubectl create configmap currency-conversion --from-literal=CURRENCY\_EXCHANGE\_URI=http://currency-exchange

kubectl get configmap

kubectl get configmap currency-conversion -o yaml >> configmap.yaml

watch -n 0.1 curl http://34.66.241.150:8100/currency-conversion-feign/from/USD/to/INR/quantity/10

docker push in28min/mmv2-currency-conversion-service:0.0.12-SNAPSHOT

docker push in28min/mmv2-currency-exchange-service:0.0.12-SNAPSHOT