**Definition:** Small autonomous services that work together.

**Advantages of Micro Services:**

* Smaller codebase is easy to maintain.
* Can independently scale up highly used services
* Each team can focus on one (or few) Micro Service(s)
* Technology updates/rewrites become simpler

1. **Configuration Management:**

* Spring cloud config server (Keeps configuration in one place, that makes easy to maintain configuration for all micro-services)

1. **Dynamic Scale Up and Scale down:**

* Naming Server (Eureka)
* Ribbon (Client Side Load Balancing by using Naming server)
* Feign (Easier REST Clients)

1. **Visibility and Monitoring:**

* Zipkin distributed tracing (to trace request across multiple components)
* Netflix API gateway.

Spring cloud slouth will assign id to request across multiple components.

1. **Fault Tolerance:**

* Hystrix

**Zuul API Gateway:**

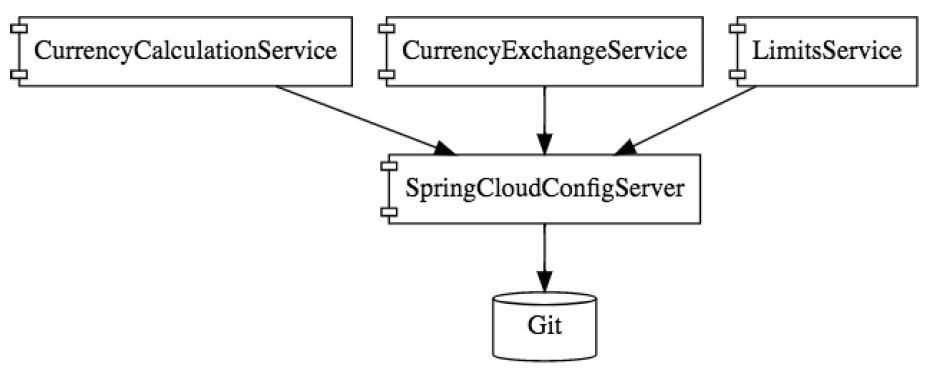
* Authentication, Authorization and Security
* Rate Limits
* Fault Toleration
* Service Aggregation

Feign is used as rest client

Ribbon is used as load balancing. (We are hard coding instance URL’s)

If you use both Feign and Ribbon then only load balancing works.

Eureka Naming Server is used to dynamically increase and decrease instances. (We no need to hard code URL’s)



**Development Notes:**

**limits-service:**

@SpringBootApplication

@EnableHystrix

**public** **class** LimitsServiceApplication {

}

**spring-cloud-starter-config:** is used as spring cloud config server server client.

**spring-cloud-starter-bus-amqp:** is used to refresh the configuration. Refresh URL is

http://localhost:2018/actuator/bus-refresh (POST method)

**spring-cloud-starter-netflix-hystrix:** is used to implement fault tolerance.

@HystrixCommand(fallbackMethod = "fallbackRetrieveConfiguration")

@EnableHystrix (at Application level)

**spring-boot-starter-actuator:** is used to monitor the application.

**spring-data-rest-hal-browser:** is UI for actuator.

**spring-boot-devtools:** Automatically pics up the changes.

* To get configuration from spring-cloud-config-server project to limits-service we need to rename application.properties to bootstrap.properties then add

spring.cloud.config.uri=http://localhost:8888/

spring.profiles.active=dev

management.endpoints.web.exposure.include=\* (This requires for bus refresh)

@ConfigurationProperties("limit-service") (Is used to read the properties files with prefix limit-service)

**spring-cloud-config-server:**

@SpringBootApplication

@EnableConfigServer

**public** **class** SpringCloudConfigServerApplication {

}

**spring-cloud-config-server:** it’s config server.

@EnableConfigServer (at Application level)

To connect spring-cloud-config-server to GIT repo add below property in application.properties.

spring.cloud.config.server.git.uri=file:\\E:\\GitRepo

If the active profile is DEV and key is not available in DEV properties file, then value will be picked up from default properties file.

**eureka-naming-server:**

@SpringBootApplication

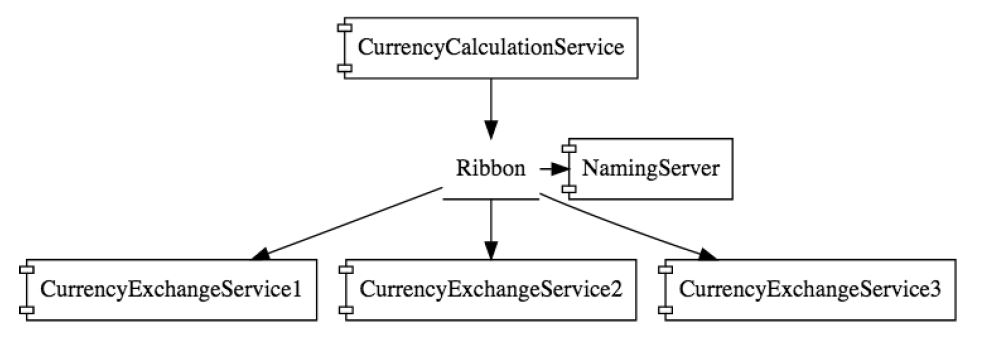
@EnableEurekaServer

**public** **class** EurekaNamingServerApplication {

}

**spring-cloud-starter-netflix-eureka-server:** is used for define eureka server.

@EnableEurekaServer (at Application level)



**currency-exchange-service:**

@SpringBootApplication

@EnableDiscoveryClient

**public** **class** CurrencyExchangeServiceApplication {

}

**spring-cloud-starter-netflix-eureka-client:** to register this project with eureka server.

@EnableDiscoveryClient (at Application level)

eureka.client.service-url.default-zone=http://localhost:8761/eureka (in properties file)

**spring-cloud-starter-sleuth:** is used to assign unique id to same request.

@Bean

**public** Sampler defaultSampler() {

**return** Sampler.***ALWAYS\_SAMPLE***;

}// To trace all the requests.

**spring-cloud-sleuth-zipkin:** zipkin server.

**spring-cloud-starter-bus-amqp:** rabbit MQ.

**currency-conversion-service:**

@SpringBootApplication

@EnableFeignClients("com.spring.microservices")

@EnableDiscoveryClient

**public** **class** CurrencyConversionServiceApplication {

}

**spring-cloud-starter-netflix-eureka-client:** to register this project with eureka server.

@EnableDiscoveryClient (at Application level)

eureka.client.service-url.default-zone=http://localhost:8761/eureka (in properties file)

**spring-cloud-starter-sleuth:** is used to assign unique id to same request.

@Bean

**public** Sampler defaultSampler() {

**return** Sampler.***ALWAYS\_SAMPLE***;

}// To trace all the requests.

**spring-cloud-sleuth-zipkin:** zipkin server.

**spring-cloud-starter-bus-amqp:** rabbit MQ.

**spring-cloud-starter-openfeign:** is used to write client code to invoke service feign make it easy.

@EnableFeignClients("com.spring.microservices") (at Application lever)

@FeignClient("zuul-api-gateway-server") (we used zuul api app name here so request go through the zuul api gateway)

**public** **interface** CurrencyExchangeServiceProxy {

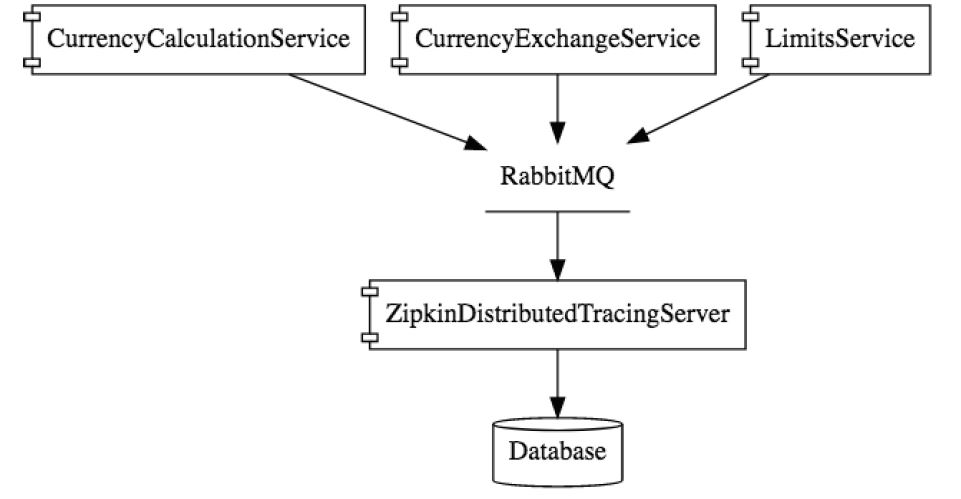
}

**spring-cloud-starter-netflix-ribbon:** it enables client side load balancing.

@RibbonClient("currency-exchange-service")

**public** **interface** CurrencyExchangeServiceProxy {

}



**zuul-api-gateway-server:**

@SpringBootApplication

@EnableZuulProxy

@EnableDiscoveryClient

**public** **class** ZuulApiGatewayServerApplication {

}

**spring-cloud-starter-netflix-zuul:** zuul api gateway server.

@EnableZuulProxy (at Application lever)

**spring-cloud-starter-netflix-eureka-client:** to register this project with eureka server.

@EnableDiscoveryClient (at Application level)

eureka.client.service-url.default-zone=http://localhost:8761/eureka (in properties file)

**spring-cloud-starter-sleuth:** is used to assign unique id to same request.

@Bean

**public** Sampler defaultSampler() {

**return** Sampler.***ALWAYS\_SAMPLE***;

}// To trace all the requests.

**spring-cloud-sleuth-zipkin:** zipkin server.

**spring-cloud-starter-bus-amqp:** rabbit MQ.

@Component

**public** **class** ZuulLoggingFilter **extends** ZuulFilter {

}