SpringCloud Netflix

一：微服务架构说明

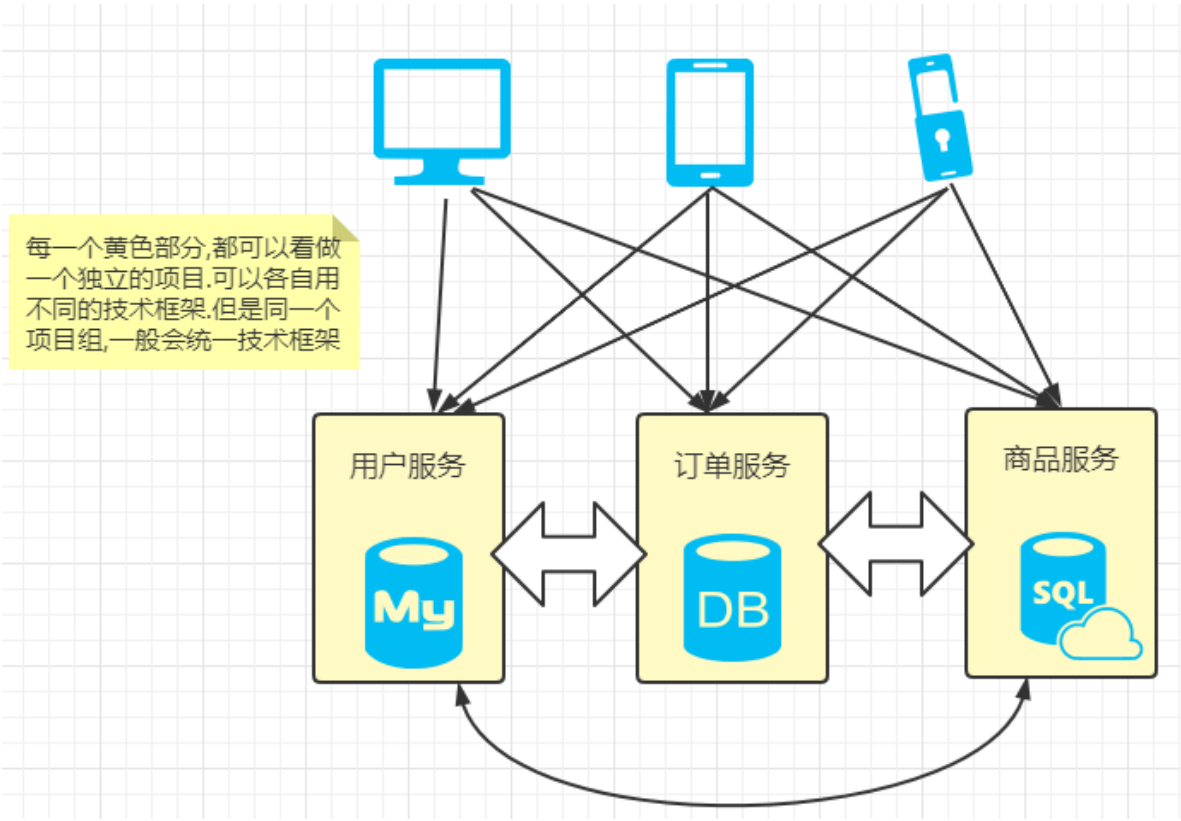
1.1 概念

> 把一个大型的单个应用程序和服务拆分为数个甚至数十个的支持微服务，它可扩展单个组件而不是整个的应用程序堆栈.

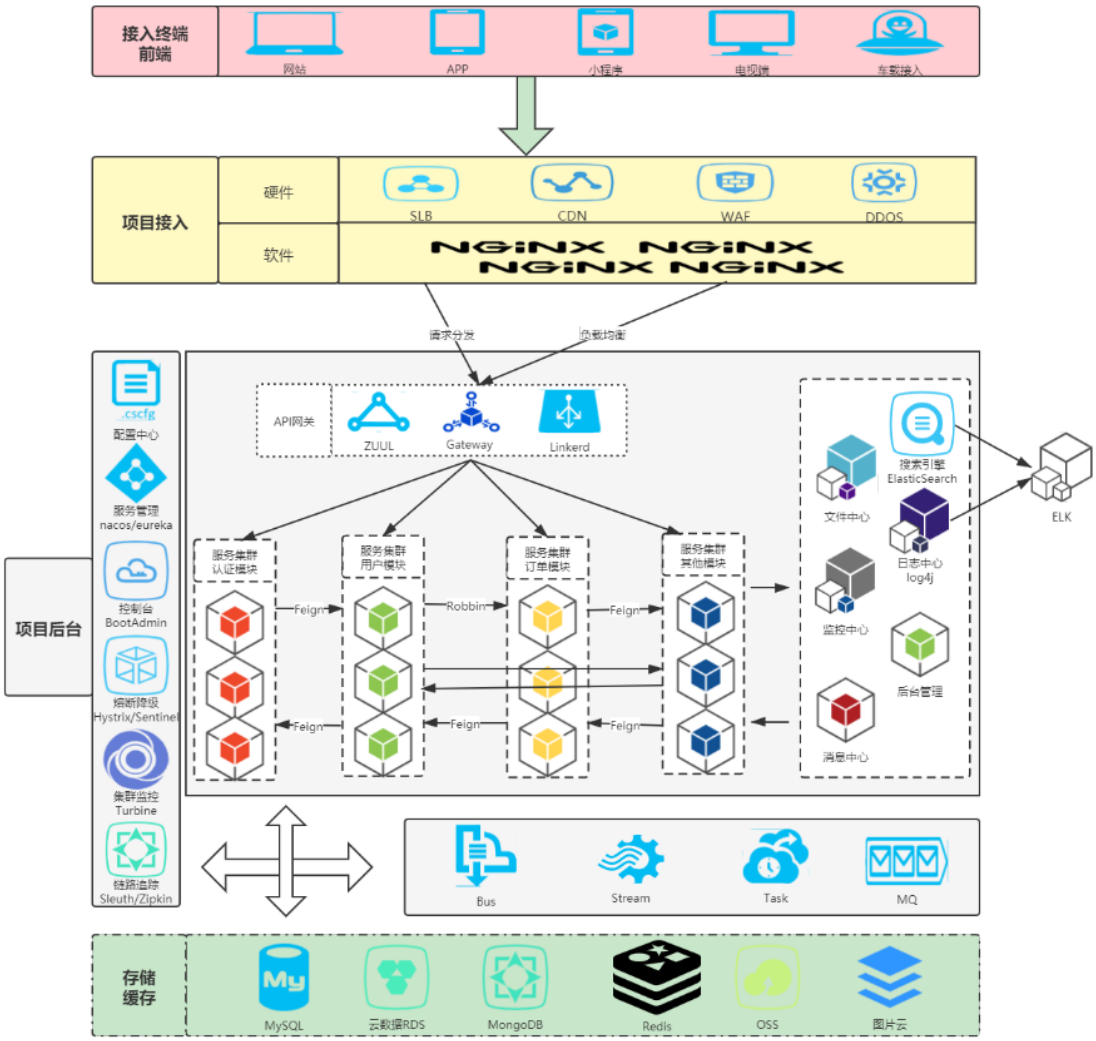
1.2 定义

> 围绕业务领域组件来创建应用，这些应用可独立地进行开发、管理和迭代。在分散的组件中使用云架构和平台式部署、管理和服务功能，使产品交付变得更加简单。

1.3 图形说明



> 实际完整架构



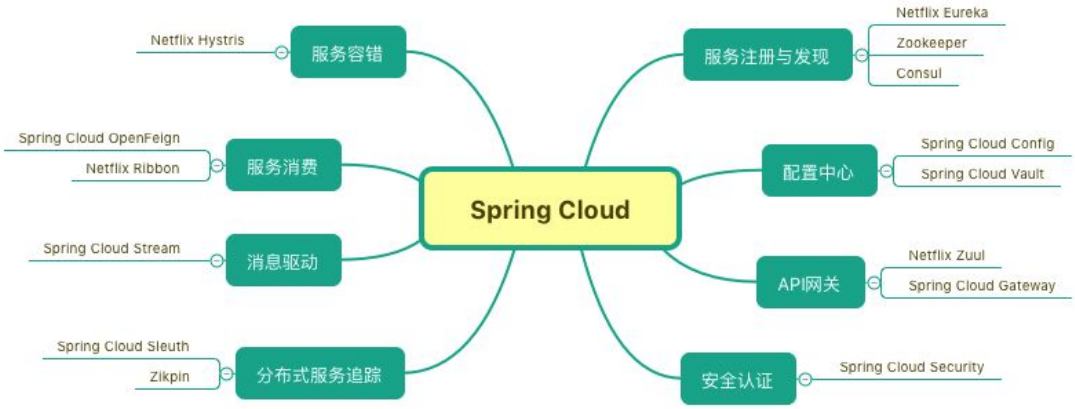
二：微服务主流框架

2.1 概念

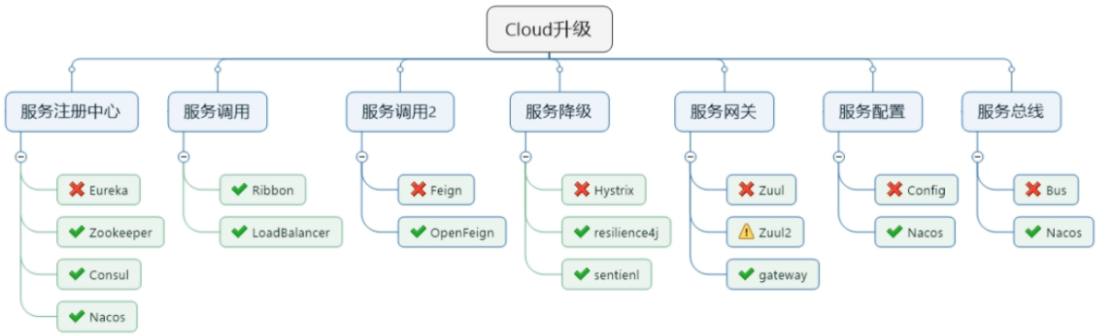
> 不同的技术公司,有自己的一套技术框架;

2.2 主要的框架介绍

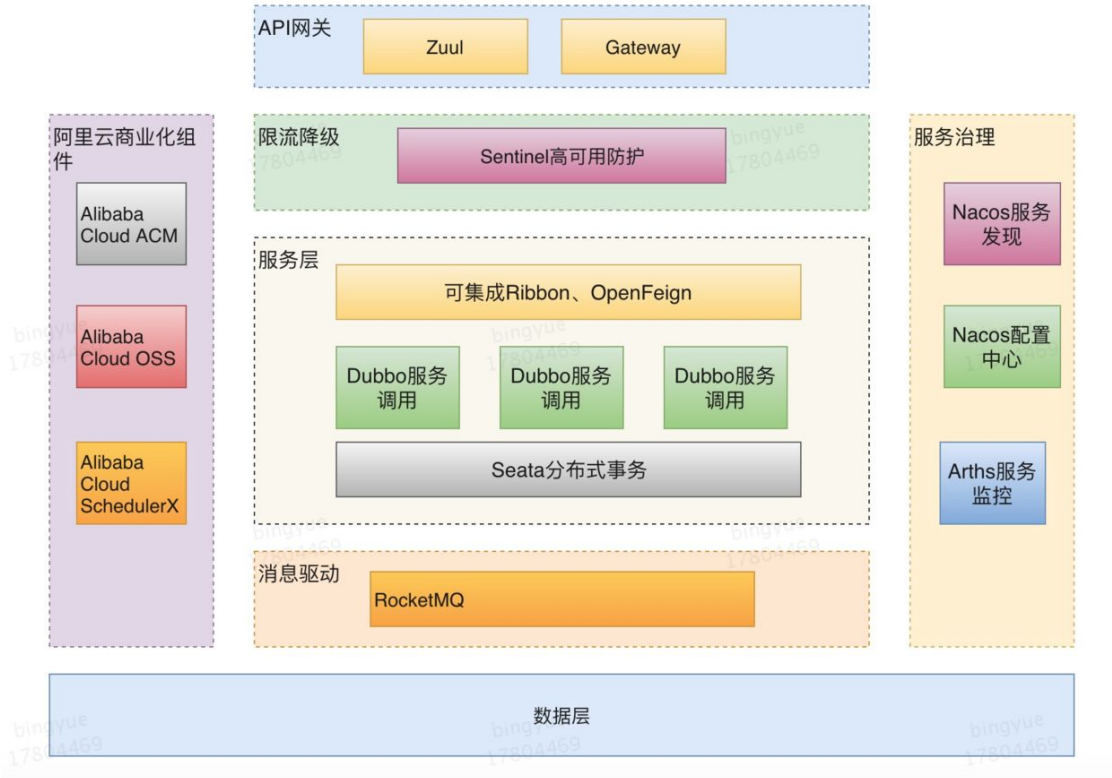
> Dubbo(阿里) : 目前开源于Apache ； 2012年推出;2014年停更;2015年恢复更新
>
> DubboX(当当基于Dubbo的更新)
>
> JD-hydra(京东基于Dubbo的更新)
>
> ServiceComb/CSE(华为2017)
>
> SpringCloud (Spring推出) 官网有自己的组件,但是部分没人用



SpringCloud Netfix (由Netflix开发,后面加入) 相对使用人数最多;但是现在已经停更



SpringCloud Alibaba(阿里推出 2018.10.31正式入驻官方孵化器)



版本对照

	Spring Cloud Netflix	Spring Cloud 官方	Spring Cloud Zookeeper	Spring Cloud Consul	Spring Cloud Kubernetes	Spring Cloud Alibaba
分布式配置	Archaius	Spring Cloud Config	Zookeeper	Consul	ConfigMap	Nacos
服务注册/发现	Eureka	–	Zookeeper	Consul	Api Server	Nacos
服务熔断	Hystrix	–	–	–	–	Sentinel
服务调用	Feign	OpenFeign RestTemplate	–	–	–	Dubbo RPC
服务路由	Zuul	Spring Cloud Gateway	–	–	–	Dubbo PROXY
分布式消息	–	SCS RabbitMQ	–	–	–	SCS RocketMQ
负载均衡	Ribbon	–	–	–	–	Dubbo LB
分布式事务	–	–	–	–	–	Seata

三 : SpringCloud Netflix

3.1 简介

> Spring Cloud是一系列框架的有序集合。它利用Spring Boot的开发便利性巧妙地简化了分布式系统基础设施的开发，如服务发现注册、配置中心、消息总线、负载均衡、断路器、数据监控等，都可以用Spring Boot的开发风格做到一键启动和部署。

3.2 版本介绍

> 采用英国伦敦地铁站来命名，并由地铁站名称字母A-Z依次类推的形式来迭代版本
>
> 在Hoxton版本后,采用传统数字版本的方式;可能是没有i开头的地铁站名字了;

3.3 版本兼容

> SpringCloud版本和Springboot版本兼容
>
> 不同的springboot版本,配套支持SpringCloud的版本;如果使用Hoxton版本,Boot版本必须要2.2.x或者以上.
>
> **官网对照图**
>
> **Table 1. Release train Spring Boot compatibility**

Release Train	Boot Version
2020.0.x aka Ilford	2.4.x, 2.5.x (Starting with 2020.0.3)
Hoxton	2.2.x, 2.3.x (Starting with SR5)
Greenwich	2.1.x
Finchley	2.0.x
Edgware	1.5.x
Dalston	1.5.x

3.4 最新版本

> **Documentation**
>
> Each **Spring project** has its own; it explains in great details how you can use **project features** and what you can achieve with them.
>
> | | |
> |---|---|
> | 2020.0.3 CURRENT GA | Reference Doc. |
> | 2020.0.4-SNAPSHOT SNAPSHOT | Reference Doc. |
> | Hoxton.SR11 GA | Reference Doc. |
> | Hoxton.BUILD-SNAPSHOT SNAPSHOT | Reference Doc. |

四 : 组件介绍

4.1 Eureka

> 服务注册和发现，它提供了一个服务注册中心、服务发现的客户端，还有一个方便的查看所有注册的服务的界面。 所有的服务使用Eureka的服务发现客户端来将自己注册到Eureka的服务器上。

4.2 Ribbon

> 负载均衡，一个请求发送给某一个服务的应用的时候，如果一个服务启动了多个实例，就会通过Ribbon来通过一定的负载均衡策略来发送给某一个服务实例。

4.3 Feign

> 服务客户端，服务之间如果需要相互访问，可以使用RestTemplate，也可以使用Feign客户端访问。它默认会使用Ribbon来实现负载均衡。

4.4 Hystrix

> 监控和熔断器。我们只需要在服务接口上添加Hystrix标签，就可以实现对这个接口的监控和断路器功能。
>
> Hystrix Dashboard，监控面板，他提供了一个界面，可以监控各个服务上的服务调用所消耗的时间等。
>
> Turbine，监控聚合，使用Hystrix监控，我们需要打开每一个服务实例的监控信息来查看。

4.5 Zuul

> 网关,所有的客户端请求通过这个网关访问后台的服务。他可以使用一定的路由配置来判断某一个URL由哪个服务来处理。并从Eureka获取注册的服务来转发请求。

五 : 项目规划

5.1 版本选择

> JDK : 1.8
>
> SpringBoot : 2.3.7.RELEASE
>
> SpringCloud : Hoxton.SR9

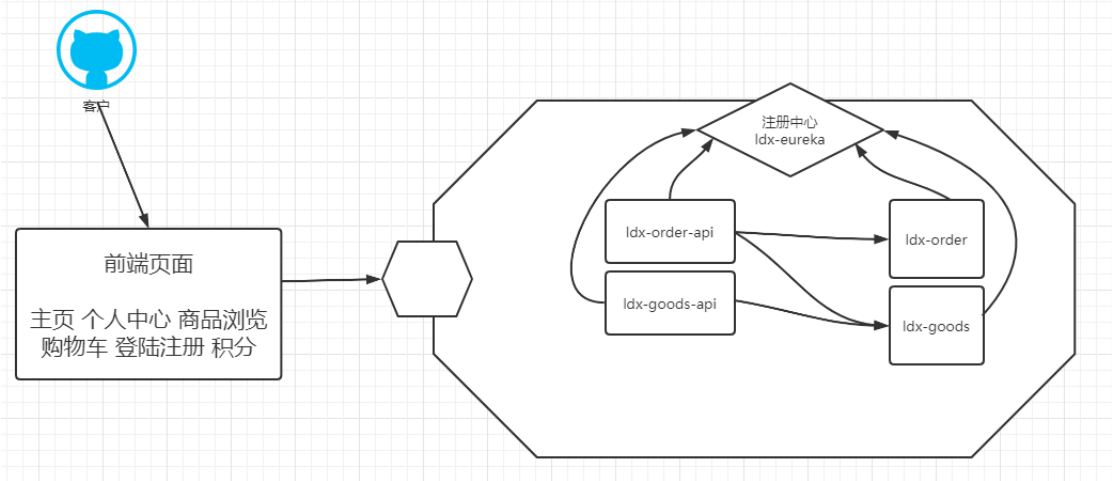
> **说明 : 请使用相同的版本配置,如果使用其他版本;大概率会造成某些不确定的问题;**

```
<properties>
    <java.version>1.8</java.version>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <spring-boot.version>2.3.7.RELEASE</spring-boot.version>
    <spring-cloud.version>Hoxton.SR9</spring-cloud.version>
</properties>
```

5.2 项目架构

> 项目名称 : ldx-demo
>
> 项目模块 :
>
> ldx-dependency
>
> pom方式;项目总依赖包;通用的依赖版本;总体父项目
>
> ldx-commons
>
> jar包方式;项目公共包;包含通用实体类,工具类,结果码等.
>
> ldx-eureka
>
> 项目注册中心;可做集群 ; 默认端口 8761;
>
> 应用名称(spring.application.name) : ldx-eureka
>
> 端口规划 : 7000 ; 集群规划 7000,7001,7002;
>
> ldx-order
>
> 订单服务接口 ; 可做集群 ; 默认端口 8000
>
> 应用名称(spring.application.name) : ldx-order
>
> 端口规划 : 8000 - 8199
>
> ldx-order-api
>
> 订单服务 ; 实际调用ldx-order ; 可做集群 ; 默认端口 8200
>
> 应用名称(spring.application.name) : ldx-order-api
>
> 端口规划 : 8200 - 8399
>
> ldx-goods
>
> 商品服务接口 ; 可做集群 ; 默认端口 8400
>
> 应用名称(spring.application.name) : ldx-goods
>
> 端口规划 : 8400 - 8599
>
> ldx-goods-api
>
> 商品服务 ; 实际调用ldx-goods; 可做集群 ; 默认端口 8600
>
> 应用名称(spring.application.name) : ldx-goods-api
>
> 端口规划 : 8600 - 8799

5.3 图片流程


客户
前端页面
主页 个人中心 商品浏览
购物车 登陆注册 积分
注册中心
ldx-eureka
ldx-order-api
ldx-goods-api
ldx-order
ldx-goods