主讲老师：Fox老师

```
1 文档：00 微服务架构介绍.note
2 链接：http://note.youdao.com/noteshare?
id=49caf5a7a6d11335b4217189c809770b&sub=5318E1B30CC24CDABCB36859CF8720F7
```

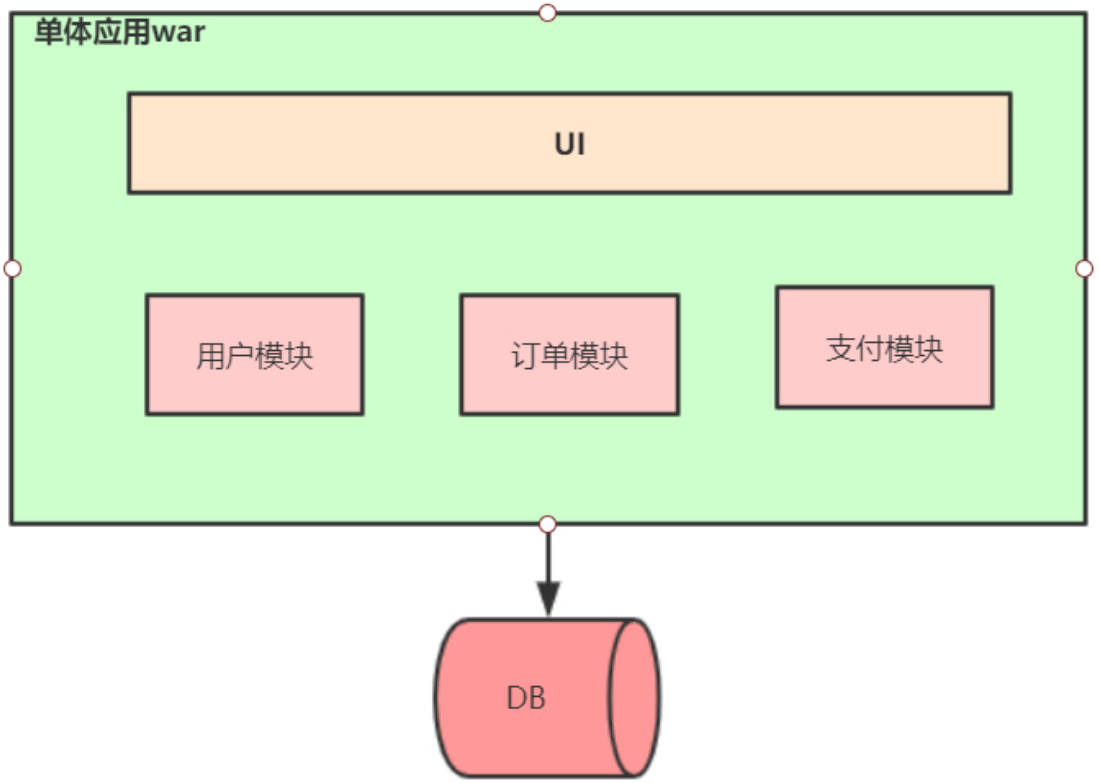
1. 单体架构vs微服务架构

1.1 单机架构

1.1.1 什么是单体架构

一个归档包（例如war格式）包含了应用所有功能的应用程序，我们通常称之为单体应用。架构单体应用的方法论，我们称之为单体应用架构。（就是一个war包打天下）

1.1.2 单体架构示意图



1.1.3 单体架构的优缺点

优点:

①: 架构简单明了，没有”花里胡哨“的问题需要解决。

②:开发，测试，部署简单（尤其是运维人员 睡着都会笑醒）

缺点:

①：随着业务扩展，代码越来越复杂，代码质量参差不齐(开发人员的水平不一),会让你每次提交代码 ， 修改每一个小bug都是心惊胆战的。

②：部署慢(由于单体架构，功能复杂) 能想像下一个来自200W+代码部署的速度(15分钟)

③: 扩展成本高，根据单体架构图 假设用户模块是一个CPU密集型的模块(涉及到大量的运算)，那么我们需要替换更加牛逼的CPU，而我们的订单模块是一个IO密集模块（涉及大量的读写磁盘）,那我们需要替换更加牛逼的内存以及高效的磁盘。但是我们的单体架构上 无法针对单个功能模块进行扩展，那么就需要替换更牛逼的CPU，更牛逼的内存，更牛逼的磁盘，价格蹭蹭的往上涨。

④: 阻碍了新技术的发展。。。。。。比如我们的web架构模块 从struts2迁移到springboot，那么就会成为灾难

1.2 微服务以及微服务架构

1.2.1 微服务的定义

①: 英文:https://martinfowler.com/articles/microservices.html

②: 中文:http://blog.cuicc.com/blog/2015/07/22/microservices

微服务核心就是把传统的单机应用，根据业务将单机应用拆分为一个一个的服务，彻底的解耦，每一个服务都是提供特定的功能，一个服务只做一件事,类似进程，每个服务都能够单独部署，甚至可以拥有自己的数据库。这样的一个一个的小服务就是微服务。

①: 比如传统的单机电商应用,有 订单/支付/库存/物流/积分等模块(理解为service)

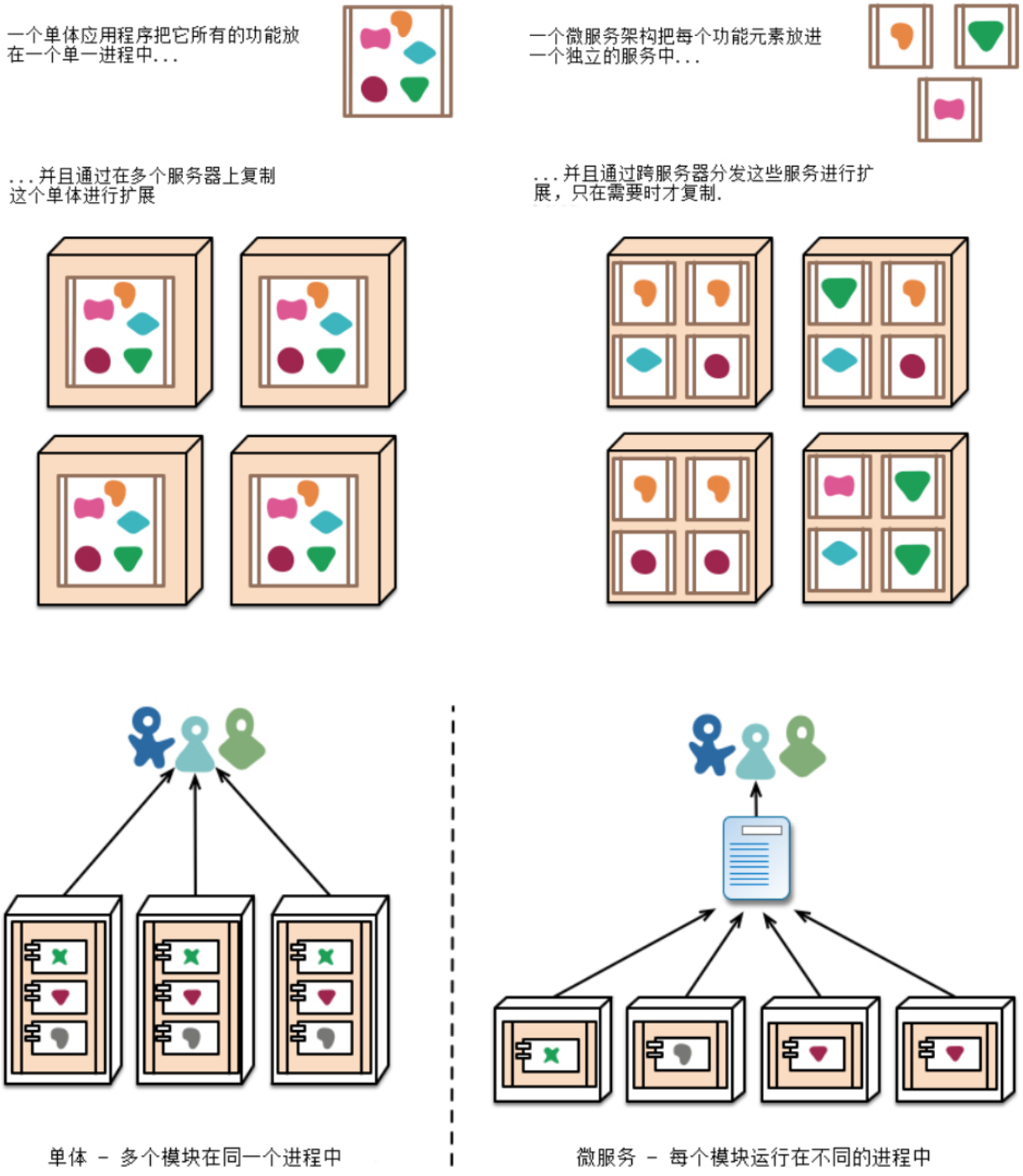
②: 我们根据 业务模型来拆分,可以拆分为 订单服务，支付服务，库存服务，物流服务，积分服务

③: 若不拆分的时候，我的非核心业务积分模块 出现了重大bug 导致系统内存溢出，导致整个服务宕机,若拆分之后，只是说我的积分微服务不可用，我的整个系统核心功能还是能使用

1.2.2 微服务的特点

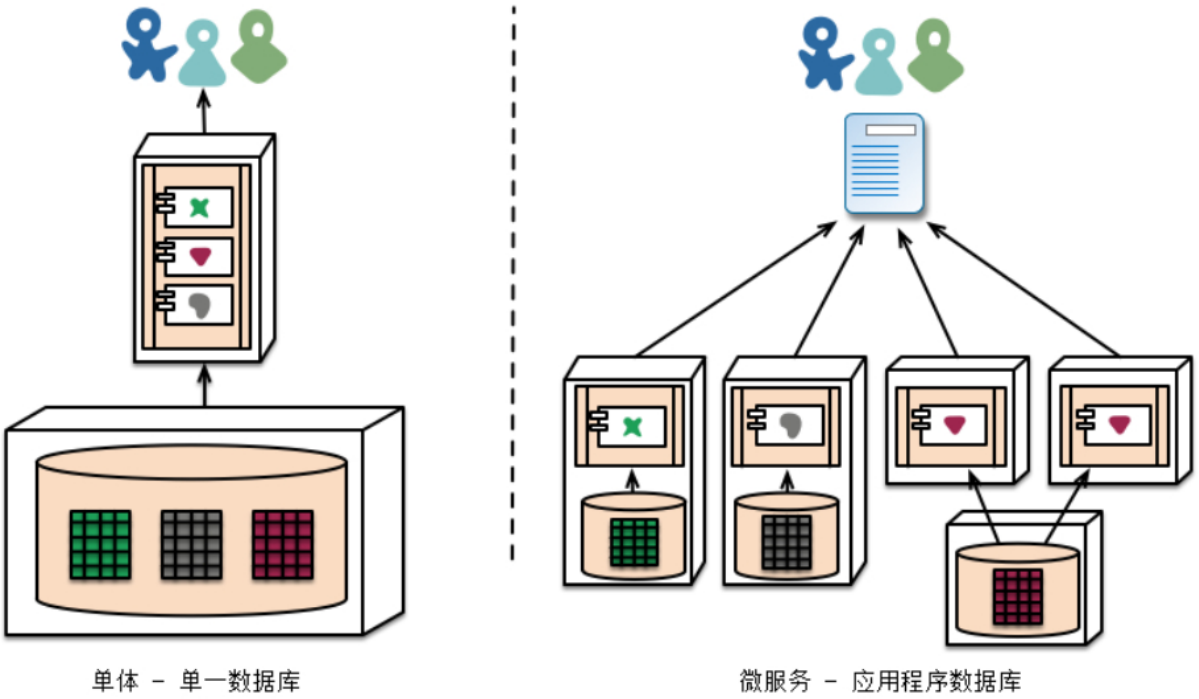
1）独立部署，灵活扩展

传统的单体架构是以整个系统为单位进行部署，而微服务则是以每一个独立组件（例如用户服务，商品服务）为单位进行部署。



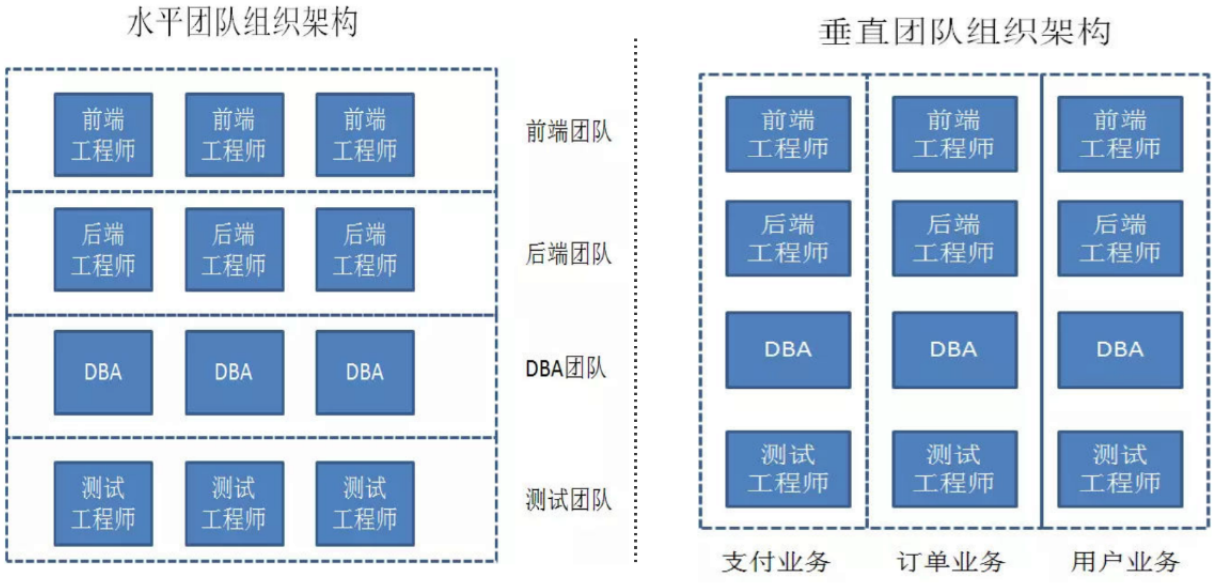
2）资源的有效隔离

微服务设计的原则之一，就是每一个微服务拥有独立的数据源，假如微服务A想要读写微服务B的数据库，只能调用微服务B对外暴露的接口来完成。这样有效避免了服务之间争用数据库和缓存资源所带来的问题。



3）团队组织架构的调整

微服务设计的思想也改变了原有的企业研发团队组织架构。传统的研发组织架构是水平架构，前端有前端的团队，后端有后端的团队，DBA有DBA的团队，测试有测试的团队。而微服务的设计思想对团队的划分有着一定的影响，使得团队组织架构的划分更倾向于垂直架构，比如用户业务是一个团队来负责，支付业务是一个团队来负责。



1.2.3 微服务架构是什么?

微服务架构风格是一种将单个应用程序作为一套小型服务开发的方法，每种应用程序都在自己的进程中运行，并与轻量级机制（通常是HTTP资源API）进行通信。 这些服务是围绕业务功能构建的，可以通过全自动部署机制独立部署。 这些服务的集中管理最少，可以用不同的编程语言编写，并使用不同的数据存储技术。

SOA架构强调的是异构系统之间的通信和解耦合，而微服务架构强调的是系统按业务边界做细粒度的拆分和部署

微服务架构是一个架构风格, 提倡:

①:将一个单一应用程序开发为一组小型服务.

②:每个服务运行在自己的进程中

③:服务之间通过轻量级的通信机制(http rest api)

④:每个服务都能够独立的部署

⑤:每个服务甚至可以拥有自己的数据库

微服务以及微服务架构的是二个完全不同的概念。微服务强调的是服务的大小和对外提供的单一功能，而微服务架构是指把 一个一个的微服务组合管理起来，对外提供一套完整的服务。

1.2.3 微服务的优缺点

优点:

①：每个服务足够小,足够内聚，代码更加容易理解,专注一个业务功能点(对比传统应用，可能改几行代码 需要了解整个系统)

②: 开发简单，一个服务只干一个事情。（加入你做支付服务，你只要了解支付相关代码就可以了）

③: 微服务能够被2-5个人的小团队开发，提高效率

④: 按需伸缩，服务松耦合，每个服务都能够开发部署

⑤: 前后端分离, 作为java开发人员，我们只要关系后端接口的安全性以及性能，不要去关注页面的人机交互(H5工程师)根据前后端接口协议，根据入参，返回json的回参。

⑥:一个服务可用拥有自己的数据库，也可以多个服务连接同一个数据库。

缺点:

①:增加了运维人员的工作量，以前只要部署一个war包，现在可能需要部署成百上千个war包 (k8s+docker+jenkins)

②: 服务之间相互调用，增加通信成本

③:数据一致性问题(分布式事务问题)

④:性能监控等,问题定位.....

1.2.4）微服务的适用场景

合适

①:大型复杂的项目.....(来自单体架构200W行代码的恐惧)

②:快速迭代的项目.....(来自一天一版的恐惧)

③:并发高的项目.....(考虑弹性伸缩扩容的恐惧)

不合适

①：业务稳定，就是修修bug ， 改改数据

②：迭代周期长 发版频率 一二个月一次.

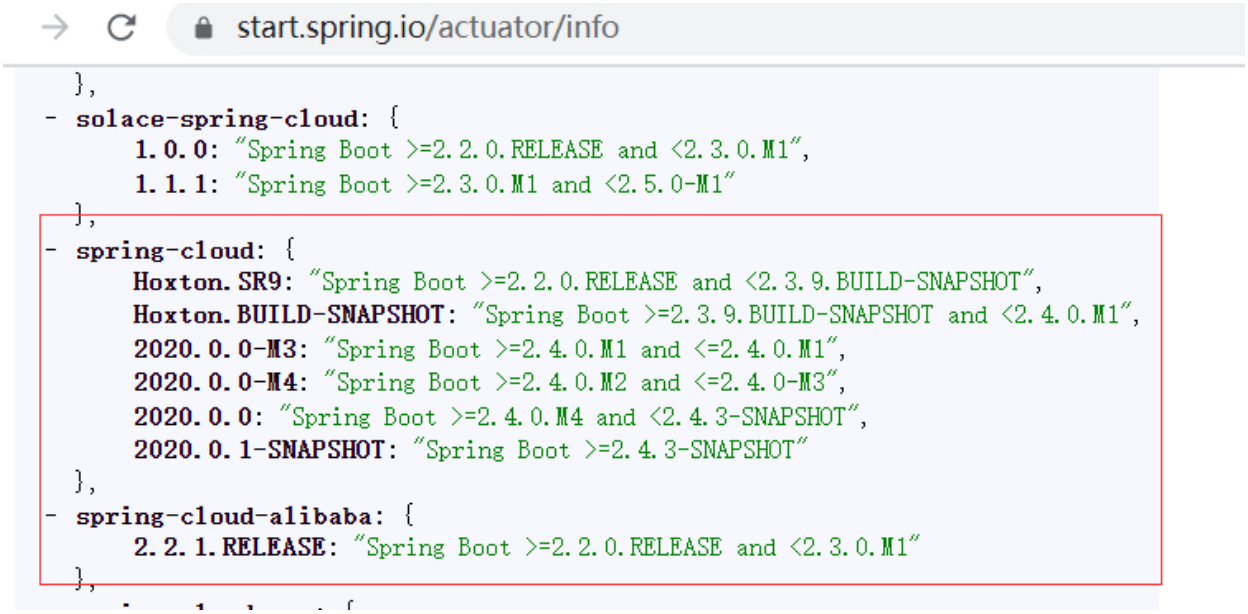
2.Spring Cloud 微服务技术栈

2.1）介绍

Spring Cloud是分布式微服务架构的一站式解决方案，是多种微服务架构落地技术的集合体，俗称微服务全家桶。

Spring Cloud为开发人员提供了快速构建分布式系统中的一些常见模式的工具(例如配置管理、服务发现、断路器、智能路由、微代理、控制总线、一次性令牌、全局锁、领导选举、分布式会话、集群状态)。

Release Train	Boot Version
Hoxton	2.2.x
Greenwich	2.1.x
Finchley	2.0.x
Edgware	1.5.x
Dalston	1.5.x

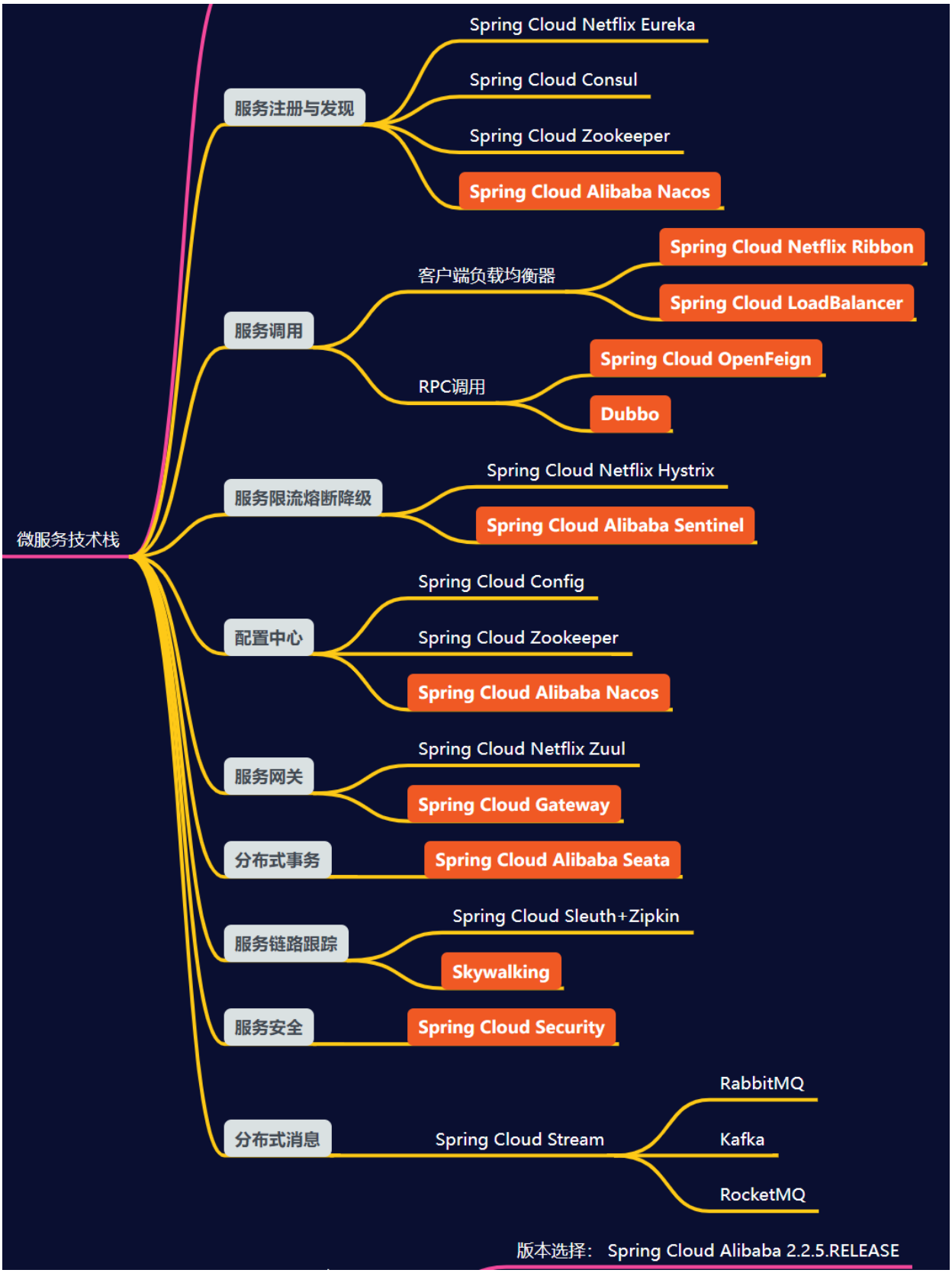


官网： https://spring.io/projects/spring-cloud

中文文档： https://www.springcloud.cc/

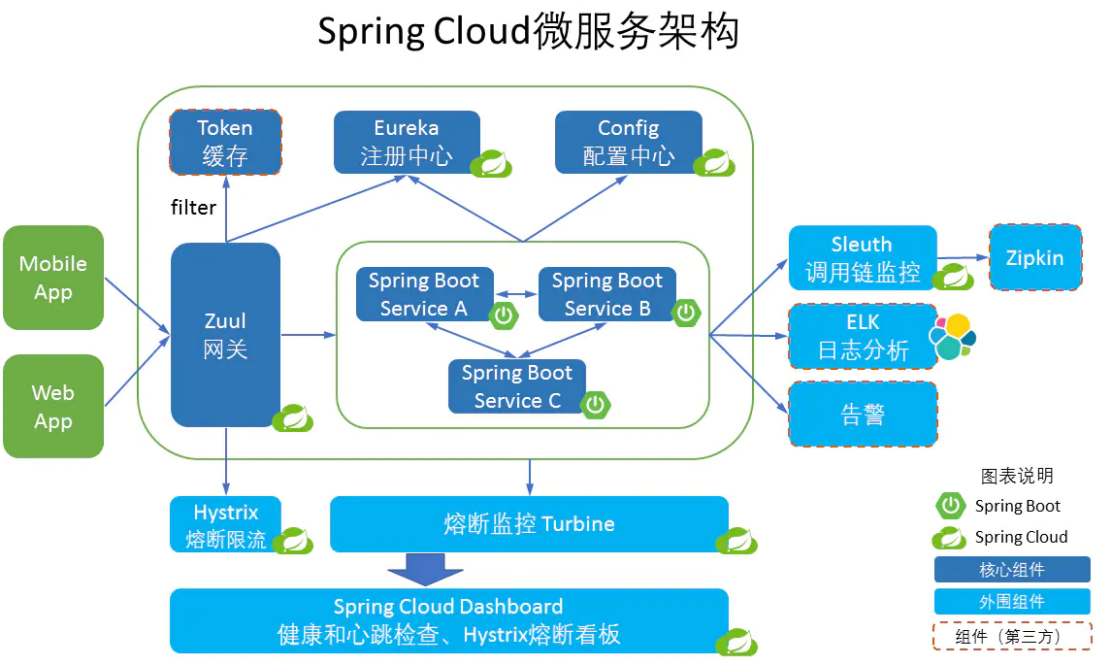
Spring Cloud中国社区：http://springcloud.cn/

2.2) SpringCloud微服务架构生态圈



2.3） Spring Cloud Netflix包含的组件：

- 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。
- 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。
- Ribbon，即负载均衡，Zuul网关将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。
- 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。
- Hystrix，监控和断路器。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。而Turbine可以帮助我们把所有的服务实例的监控信息聚合到一个地方统一查看。这样就不需要挨个打开一个个的页面一个个查看。



3. Spring Cloud Alibaba技术栈

同 Spring Cloud 一样，Spring Cloud Alibaba 也是一套微服务解决方案，包含开发分布式应用微服务的必需组件，方便开发者通过 Spring Cloud 编程模型轻松使用这些组件来开发分布式应用服务。

依托 Spring Cloud Alibaba，您只需要添加一些注解和少量配置，就可以将 Spring Cloud 应用接入阿里微服务解决方案，通过阿里中间件来迅速搭建分布式应用系统。

作为 Spring Cloud 体系下的新实现，Spring Cloud Alibaba 跟官方的组件或其它的第三方实现如 Netflix, Consul，Zookeeper 等对比，具备了更多的功能:

	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

2.1) Spring Cloud Alibaba 包含组件

阿里开源组件

Nacos：一个更易于构建云原生应用的动态服务发现、配置管理和服务管理平台。

Sentinel：把流量作为切入点，从流量控制、熔断降级、系统负载保护等多个维度保护服务的稳定性。

RocketMQ：开源的分布式消息系统，基于高可用分布式集群技术，提供低延时的、高可靠的消息发布与订阅服务。

Dubbo：在国内应用非常广泛的一款高性能 Java RPC 框架。

Seata：阿里巴巴开源产品，一个易于使用的高性能微服务分布式事务解决方案。

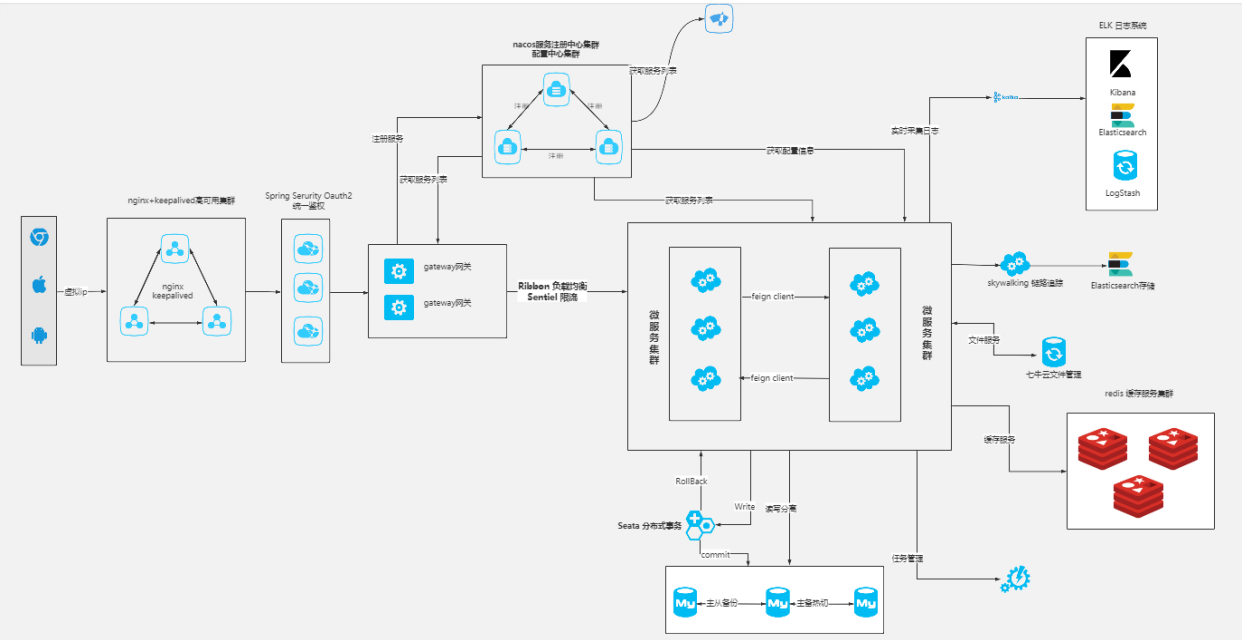
Arthas：开源的Java动态追踪工具，基于字节码增强技术，功能非常强大。

阿里商业化组件

作为一家商业公司，阿里巴巴推出 Spring Cloud Alibaba，很大程度上市希望通过抢占开发者生态，来帮助推广自家的云产品。

Alibaba Cloud ACM：一款在分布式架构环境中对应用配置进行集中管理和推送的应用配置中心产品。

Alibaba Cloud OSS：阿里云对象存储服务（Object Storage Service，简称 OSS），是阿里云提供的云存储服务。
Alibaba Cloud SchedulerX：阿里中间件团队开发的一款分布式任务调度产品，提供秒级、精准的定时（基于 Cron 表达式）任务调度服务。



2.2）Spring Cloud Alibaba版本选择

https://github.com/alibaba/spring-cloud-alibaba/wiki/%E7%89%88%E6%9C%AC%E8%AF%B4%E6%98%8E

版本选择： Spring Cloud Alibaba 2.2.8.RELEASE

父pom如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.3.12.RELEASE</version>
  <relativePath/> <!-- lookup parent from repository -->
 </parent>
  <groupId>com.tuling.mall</groupId>
  <artifactId>vip-spring-cloud-alibaba</artifactId>
```

```
  <version>0.0.1-SNAPSHOT</version>
  <name>vip-spring-cloud-alibaba</name>
  <packaging>pom</packaging>
  <description>Demo project for Spring Cloud Alibaba</description>

  <properties>
  <java.version>1.8</java.version>
  <spring-cloud.version>Hoxton.SR12</spring-cloud.version>
  <spring-cloud-alibaba.version>2.2.8.RELEASE</spring-cloud-alibaba.version>
  </properties>

  <dependencyManagement>
  <dependencies>
  <dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-dependencies</artifactId>
  <version>${spring-cloud.version}</version>
  <type>pom</type>
  <scope>import</scope>
  </dependency>
  <dependency>
  <groupId>com.alibaba.cloud</groupId>
  <artifactId>spring-cloud-alibaba-dependencies</artifactId>
  <version>${spring-cloud-alibaba.version}</version>
  <type>pom</type>
  <scope>import</scope>
  </dependency>
  </dependencies>
  </dependencyManagement>

</project>
```