苏宁 Spring Cloud 微服务脚手架工具 vole 实践分享

作者：王一硼 , 黄小虎
阅读数：3494 2019 年 6 月 12 日

为应对传统单体架构的缺陷，微服务架构被企业广泛应用。Spring Cloud 为开发人员提供了快速构建微服务的系列工具，但是并没有进行相关整合， vole 是在其基础上搭建的一套可以快速实现微服务的基础脚手架工具。

1、传统单体架构的缺陷

传统单体应用将所有功能的表示层、业务逻辑层、数据访问层、包括静态资源等全部糅合在一个工程内，编译 打包 部署在单台服务器上线，比如打成 war 包放在 Tomcat 的 webapp 目录中部署。这样的开发部署流程适合小型项目，系统功能不复杂，访问量不大的情况下有绝对的优势，开发速度快且运维方便。但是，当业务越来越复杂，功能越来越多，参与的开发人员越来越多，该流程就暴露出如下问题：

- 业务复杂，代码量增大，代码可读性、可维护性、可扩展性下降。一旦要新同事接手代码，需要花很多时间理解；
- 测试难度增大；
- 单体应用并发能力有限，访问量高，用户体验差；
- 单体应用容错率低，一旦出错，可能导致整个项目崩亏；
- 将单体应用做集群部署，添加负载均衡服务器（例如 Nginx 反向代理转发请求）可略微缓解以上两条条缺点，但不能完美解决问题。

2、微服务是什么?

微服务架构：就是将原来的单体应用按义务范围来，划分为多个小 model，每个微服务运行在自己的进程中，相互不产生影响，完全自动化独立部署，并使用轻量级机制通信，通常是 HTTP RESTUFUL API，可对各微服务进行集中管理。这些小 model 可以使用不同的编程语言及存储技术，微服务架构是分布式架构。

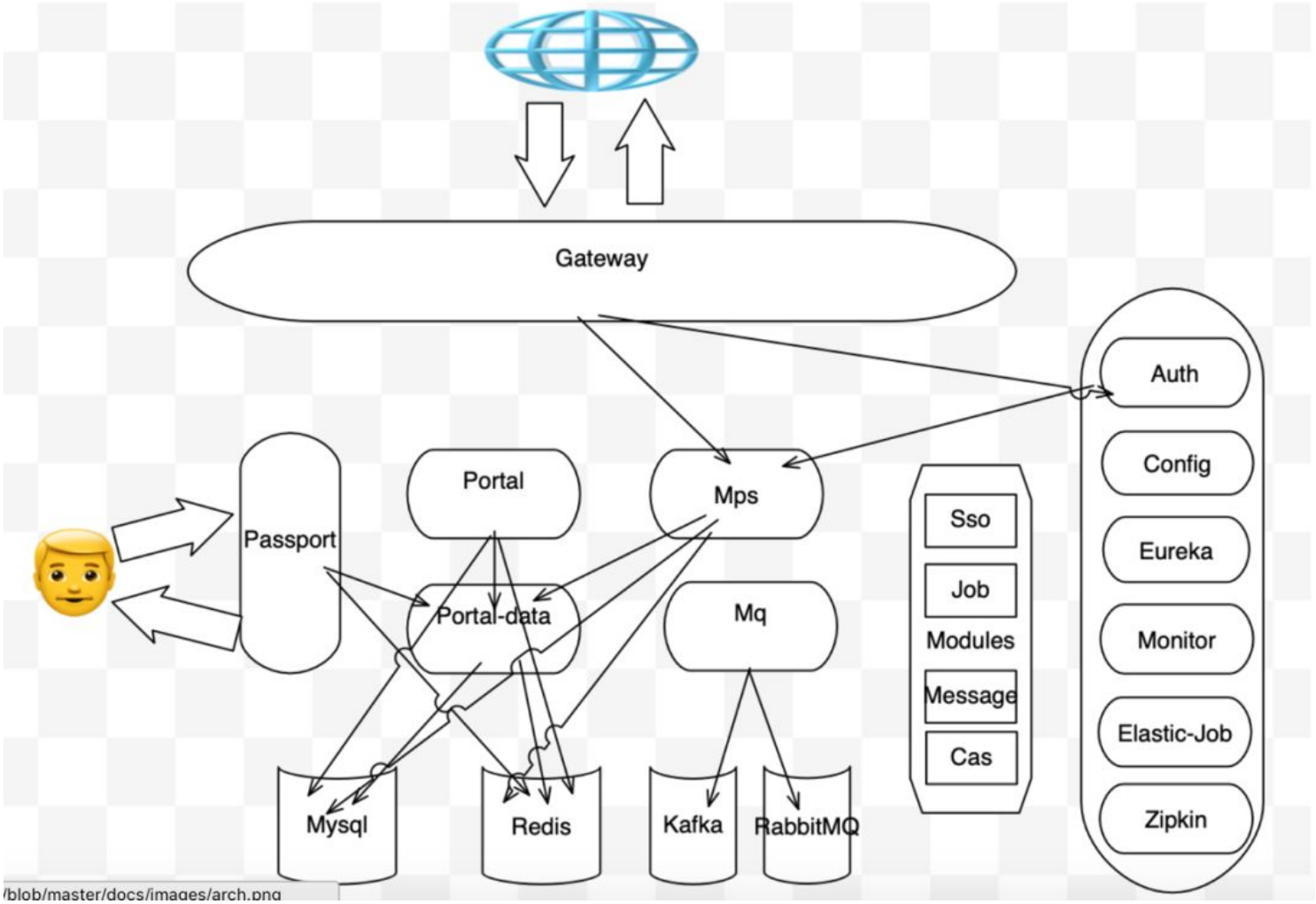
2.1、微服务架构的优点

- 按业务划分的微服务单元独立部署，运行在独立的进程中，服务之间没有任何耦合，具备良好的扩展性和复用性；
- 服务之间通常采用 HTTP 通信，该通信机制与平台和语言无关，可以使用不同的编程语言和存储方法。也可以采用轻量级消息总线通信，如 RabbitMQ、Kafaka 消息队列等，数据格式一般采用 JSON；
- 每个微服务都有自己的数据库，服务间数据库相互是独立；
- 微服务一般采用自动化工具部署。Docker 容器技术是微服务最佳部署容器；
- 服务集中化管理（服务注册与发现：Eureka、Zookeeper、Consul），监控（服务运行状况监控：Spring-Boot-Admin-Server）；
- 微服务架构是分布式架构。

3、微服务脚手架工具：vole

Spring Cloud 为开发人员提供了快速构建微服务系统的系列工具，包括配置管理、服务发现、断路器、路由、微代理、事件总线、分布式会话等相关功能，但是并没有进行相关整合， vole 是在 Spring Cloud 基础上搭建的一套可以快速实现微服务架构的基础脚手架工具，vole 基于 Spring Cloud Finchley 版本 的框架搭建，可以快速帮助项目组完成老系统微服务改造。苏宁新广告平台原来大单体应用的基础上使用 vole 对原来单体应用进行了快速改造，帮助业务系统快速搭建微服务化。

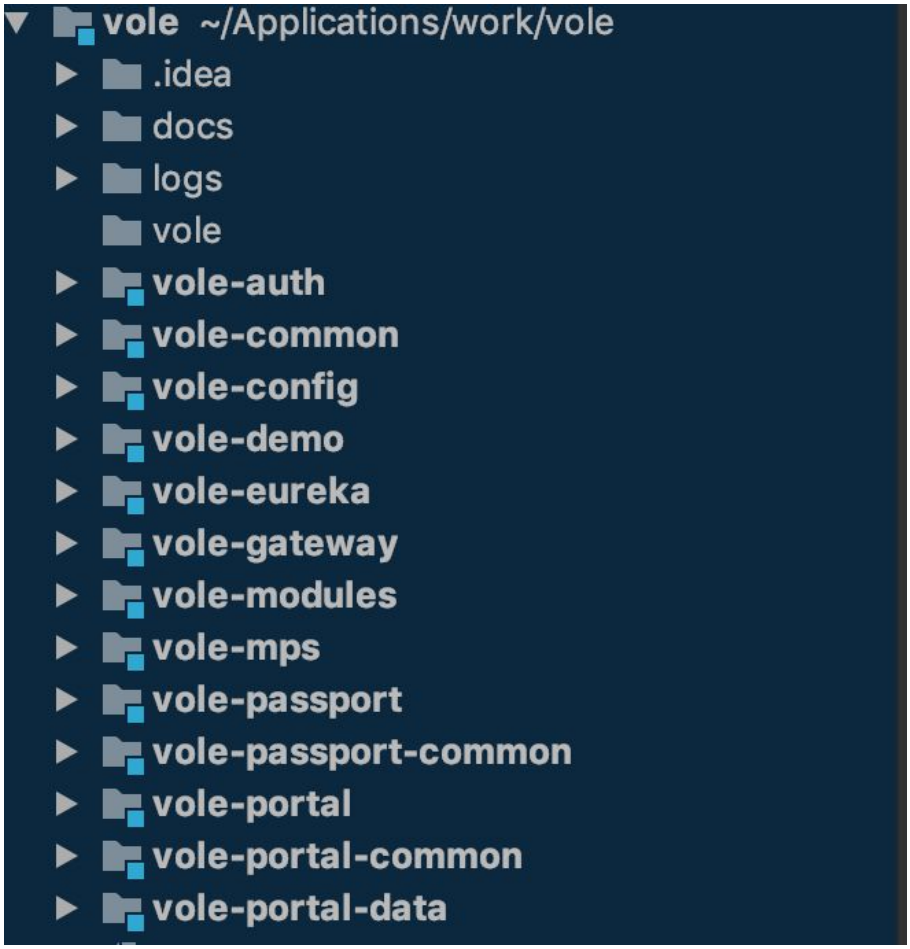
架构模型图如下所示：



Github 相关链接： https://github.com/gavenwangcn/vole

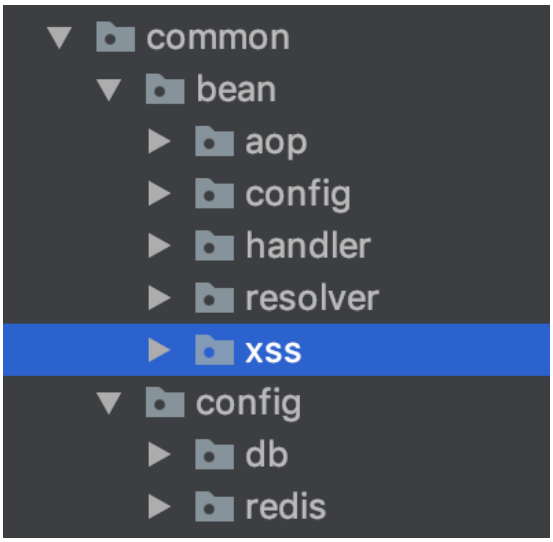
主要包括以下功能模块：

- 基础核心工具包 common
- 微服务服务中心 eureka
- 微服务配置中心 config
- 微服务鉴权中心 auth
- 微服务登陆统一中心 passport
- 微服务后台统一管理中心 portal
- 微服务边缘网关管理中心 gateway
- 微服务其他相关模块（监控，消息，job）



vole-common

vole-common 基础核心工具包，主要负责如 请求切面、服务配置、异常处理、参数格式化、请求防护、redis、Db、基础配置组件、以及其他相关如文件服务、基础工具类等组件。这里，重点要介绍 bean 包下组件 。



如上图 bean 包下主要是 aop Controller 增强切面，保证各微服务的接口返回值都基于基础格式类 ® 。

config 包主要负责 请求忽略配置插件， api 文档插件配置以及 - MVC 参数管理配置类。

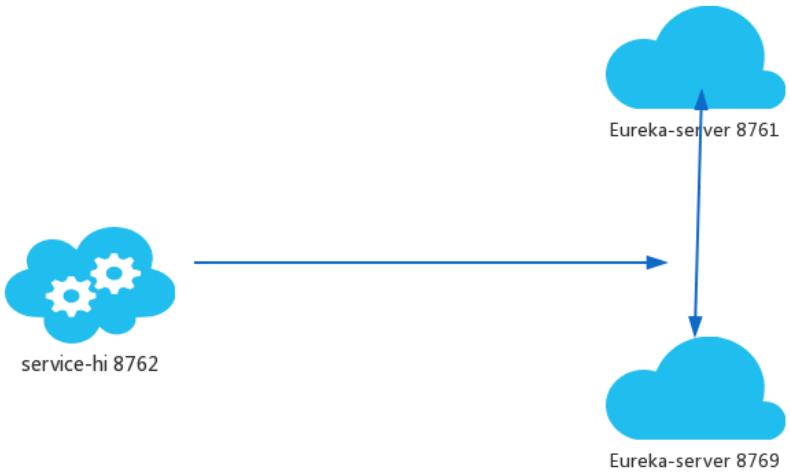
handler 和 resolver 包 分别处理全局异常和用户请求参数解析。

Xss 包处理请求攻击过滤的 比如 xss 攻击，sql 注入等。

基础工具类为整个脚手架提供了基本配置管理功能，为其他相关组件提供了基础功能。

vole- eureka

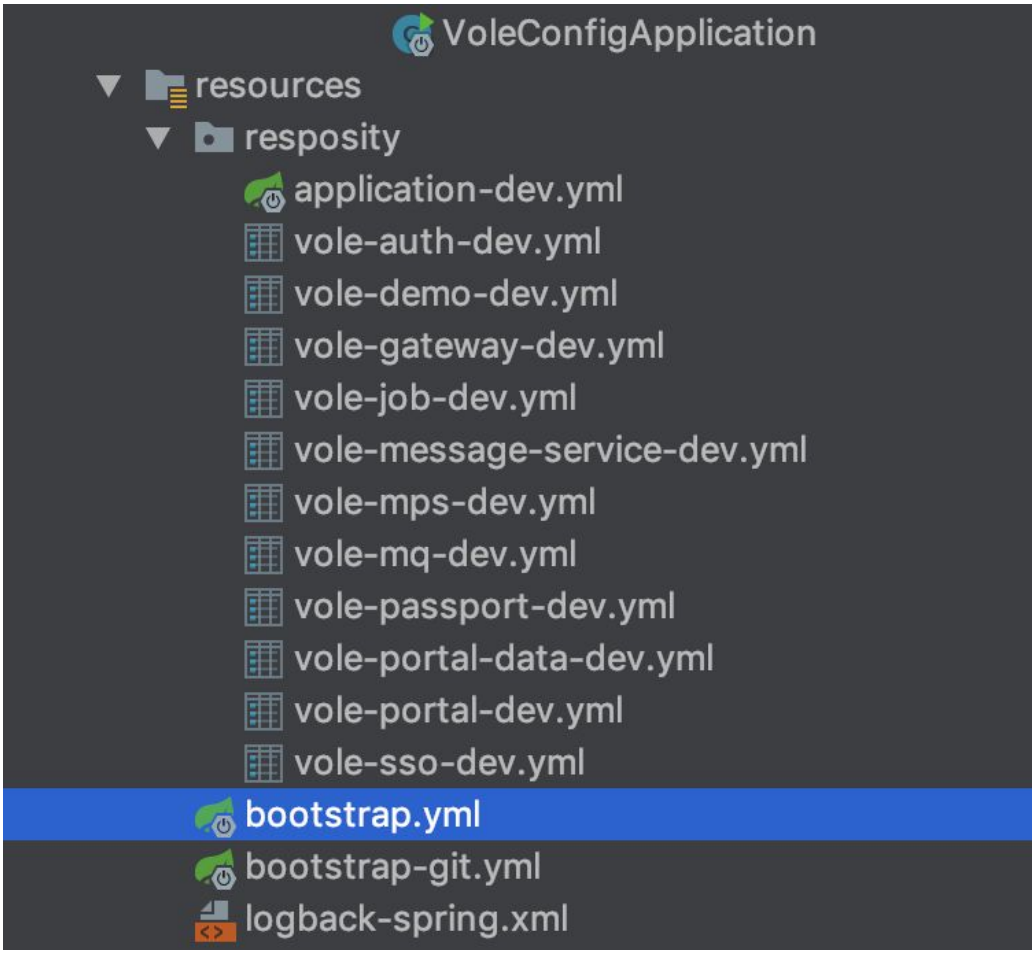
vole- eureka 微服务注册中心 eureka 高可用配置方案，Eureka 通过运行多个实例，使其更具高可用性。事实上，这是它的默认属性，用户需要做的就是给对等实例一个合法的关联 serviceUrl, 如下图所示：



eureka 互备两个节点 Eureka-eserver peer1 8761,Eureka-eserver peer2 8769 相互感应。当有服务注册时，两个 Eureka-eserver 是对等的，它们都存有相同的信息，这就是通过服务器的冗余来增加可靠性，当有一台服务器宕机了，服务并不会终止，因为另一台服务存有相同的数据。

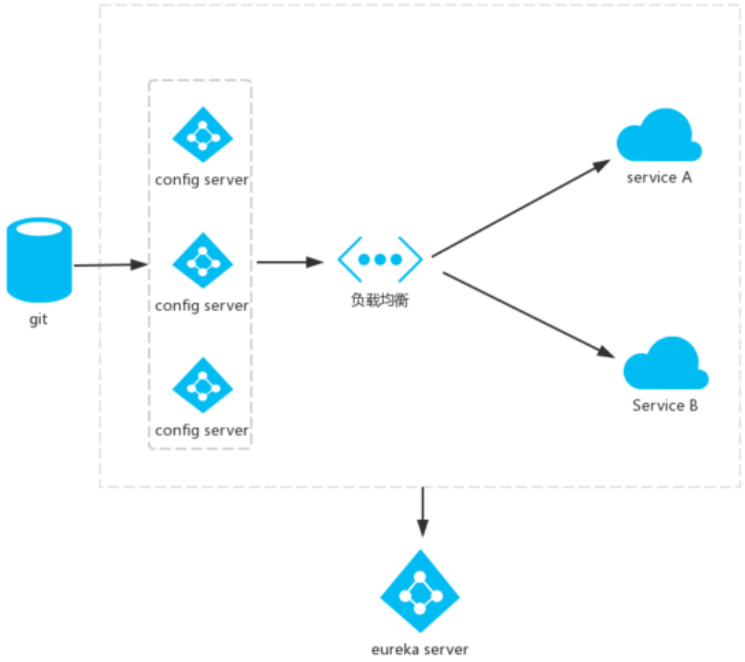
vole-config 微服务配置中心

在分布式系统中，由于服务数量巨多，为了方便服务配置文件统一管理，实时更新，所以需要分布式配置中心组件。在 Spring Cloud 中，有分布式配置中心组件 spring cloud config ，支持配置服务放在配置服务内存中（即本地），也支持放在远程 Git 仓库中。在 spring cloud config 组件中，分两个角色，一是 config server，二是 config client。



如上图 vole-config 默认使用本地方式，也可以配置到 git 服务器上去。

当服务实例很多时，都从配置中心读取文件，这时可以考虑将配置中心做成一个微服务，将其集群化，从而达到高可用，架构图如下：

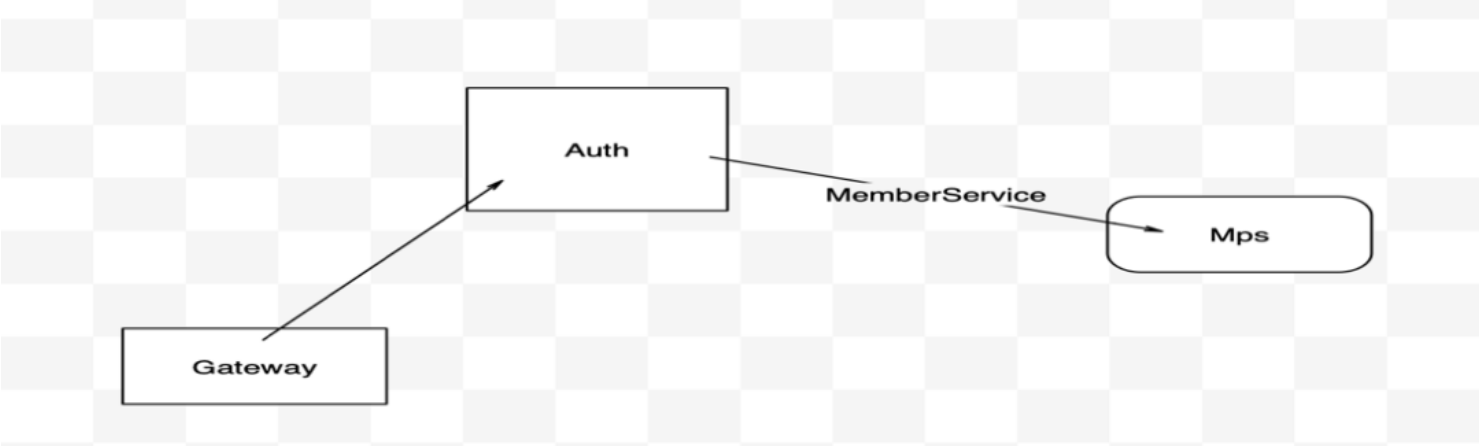


配置中心高可用架构图

vole-auth

vole-auth 是微服务鉴权中心。在微服务架构下，一个应用会被拆分成若干个微应用，每个微应用都需要对访问进行鉴权，每个微应用都需要明确当前访问用户以及权限。尤其当访问来源不只是浏览器，还包括其他服务调用时，单体应用架构下的鉴权方式就不是特别合适。在微服务架构下，要考虑外部应用接入的场景、用户 - 服务鉴权、服务 - 服务鉴权等多种鉴权场景。vole-auth 是基于 spring cloud auth2 基础上配置的鉴权中心，实现了服务与数据分离的方式，即用户相关数据 都是通过服务获取，vole-auth 负责相关鉴权功能。

如下图所示:



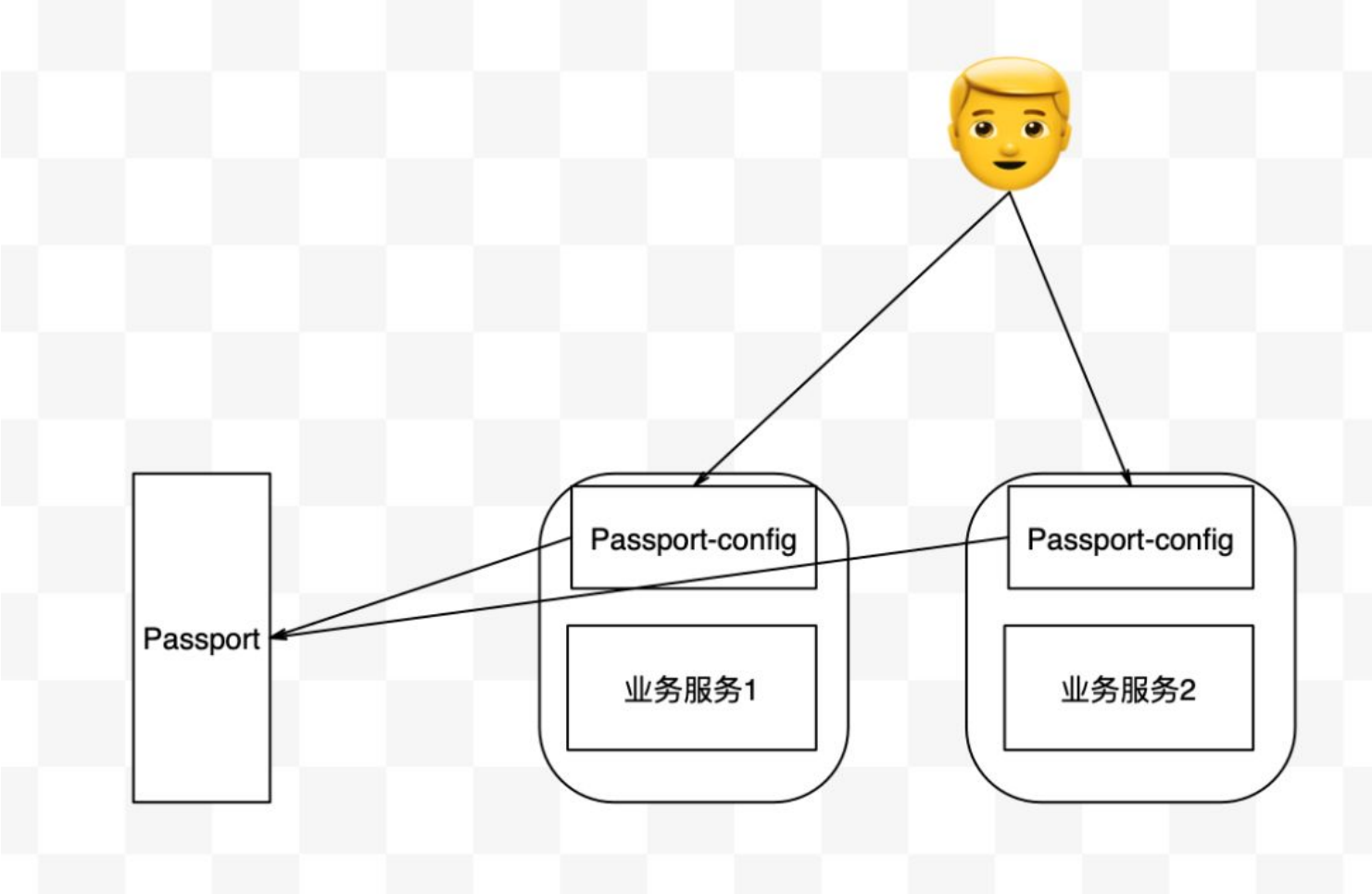
鉴权关系图

- auth 的 memberService 去掉用 mps 服务的相关会员基本信息来确认 用户权限是否正常;
- Auth 服务内部有两个重点服务包 :config 包 和 component 包;
- Config 包：主要负责 auth 服务的基本配置 ,- 包括安全配置方案，token 增强和存储方案以及 JWT 配置;
- Component 包 ： 主要负责自定义 token 使用方案，这里主要是自定义手机号 token 方案。

vole-passport

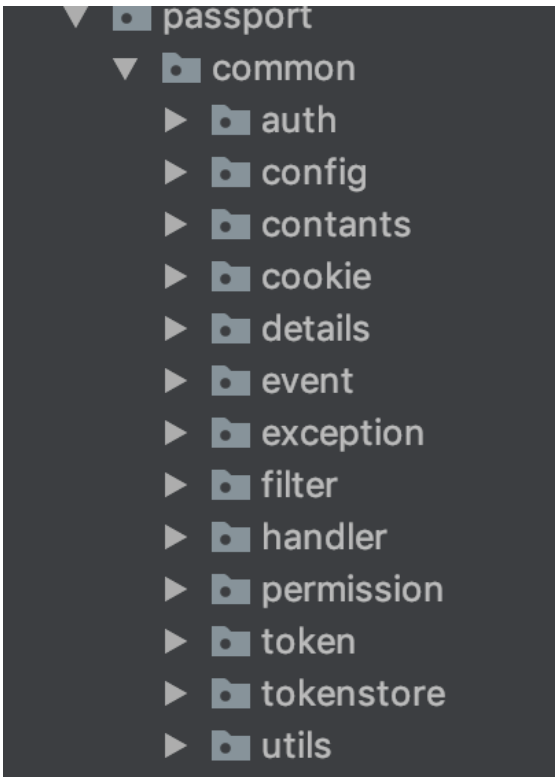
vole-passport 是微服务登陆统一中心主要负责各系统后台相关登陆服务管理，基于 springSecurity 改造使用 cookie 管理登陆信息的服务，vole-passport 支持各个服务注解式一键配置方式。

如下图所示:



统一登录配置中心图

业务人员需要通过登陆相关业务系统进行查看或者添加相关数据先要经过 passport 进行认证和鉴权。Vole-passport 采用集中式服务管理模式，各个子业务系统的介入只需系统上配置相关客户端信息。Passport 重点工模块是 passport-common 负责 passport 所有功能服务组件。



如上图所示包括：

- auth 用户认证基本信息管理；
- config 服务配置管理包括服务端配置和客户端配置；
- cookie 是负责写入和读取 cookie 基本信息；
- detail 是用户基本配置信息宝库用户的密码，登陆状态，id 等等；
- Filter 是基于 springSecurity 改造的相关的客户端和服务端的配置管理服务；
- Handler 是用户正常登陆等处的成功或失败处理；
- Permission 是负责用户登录后相关资源的鉴权服务；
- Token 用户基本 token 操作管理。

相关系统配置 passport-config 如下所示：使用配置继承 WebSecurityConfigurerAdapter 的配置类，并 @EnablePassportSso 开启 sso 配置

```
@Configuration
@EnablePassportSso
public class MpsWebSecurityConfiguration extends WebSecurityConfigurerAdapter {

    @Resource
    private FilterIgnorePropertiesConfig filterIgnorePropertiesConfig;


    @Override
    protected void configure(HttpSecurity http) throws Exception {
        String logout ="/logout";
        ExpressionUrlAuthorizationConfigurer<HttpSecurity>.ExpressionInterceptUrlRegistry registry =
                http
                        .authorizeRequests();
        registry.requestMatchers(CorsUtils::isPreFlightRequest).permitAll();
        filterIgnorePropertiesConfig.getUrls().forEach(url -> registry.antMatchers(url).permitAll());
        registry.anyRequest().authenticated()
                .and()
                .logout().logoutUrl(logout)
                .invalidateHttpSession(true)
                .clearAuthentication(true);
    }
```

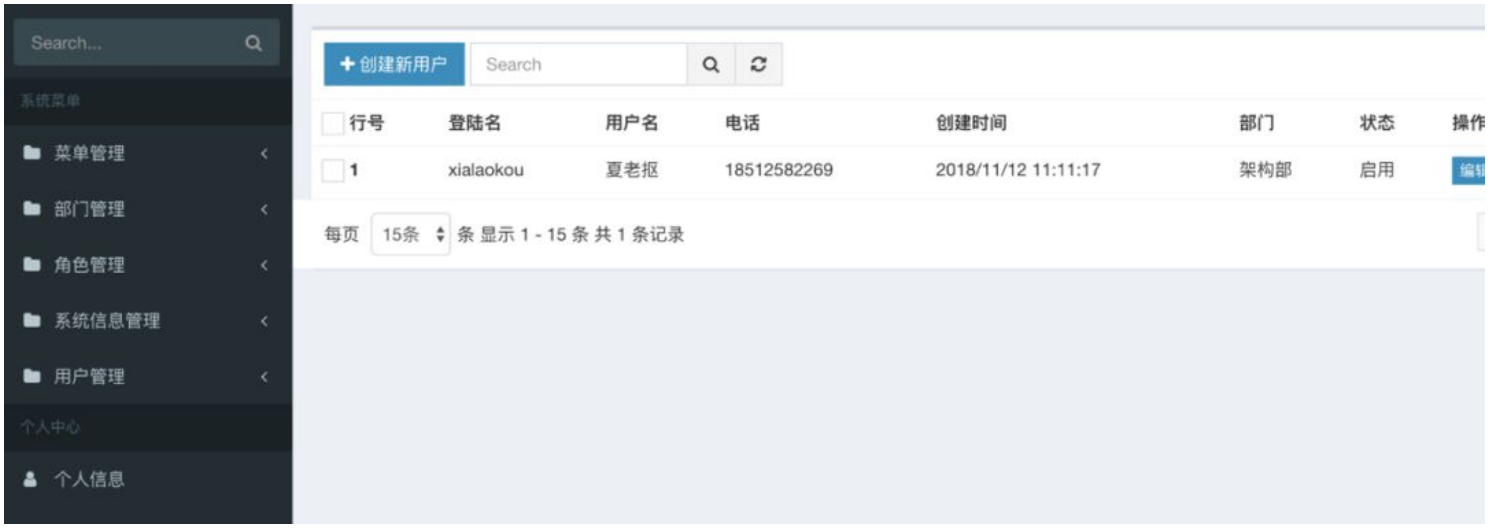
在 springboot 启动配置文件 (application.yml) 上配置相关统一登录服务链接：

```
security:
  passport:
    defaultRedirectUrl: http://localhost:7007/passport/login
```

vole-portal

vole-portals 是分布式统一管理后台门户，实现对各系统后台的权限统一管理，包括 - 用户管理、角色管理、菜单和权限管理以及系统管理，是相关系统后台管理的基本功能集合体。

如下图：

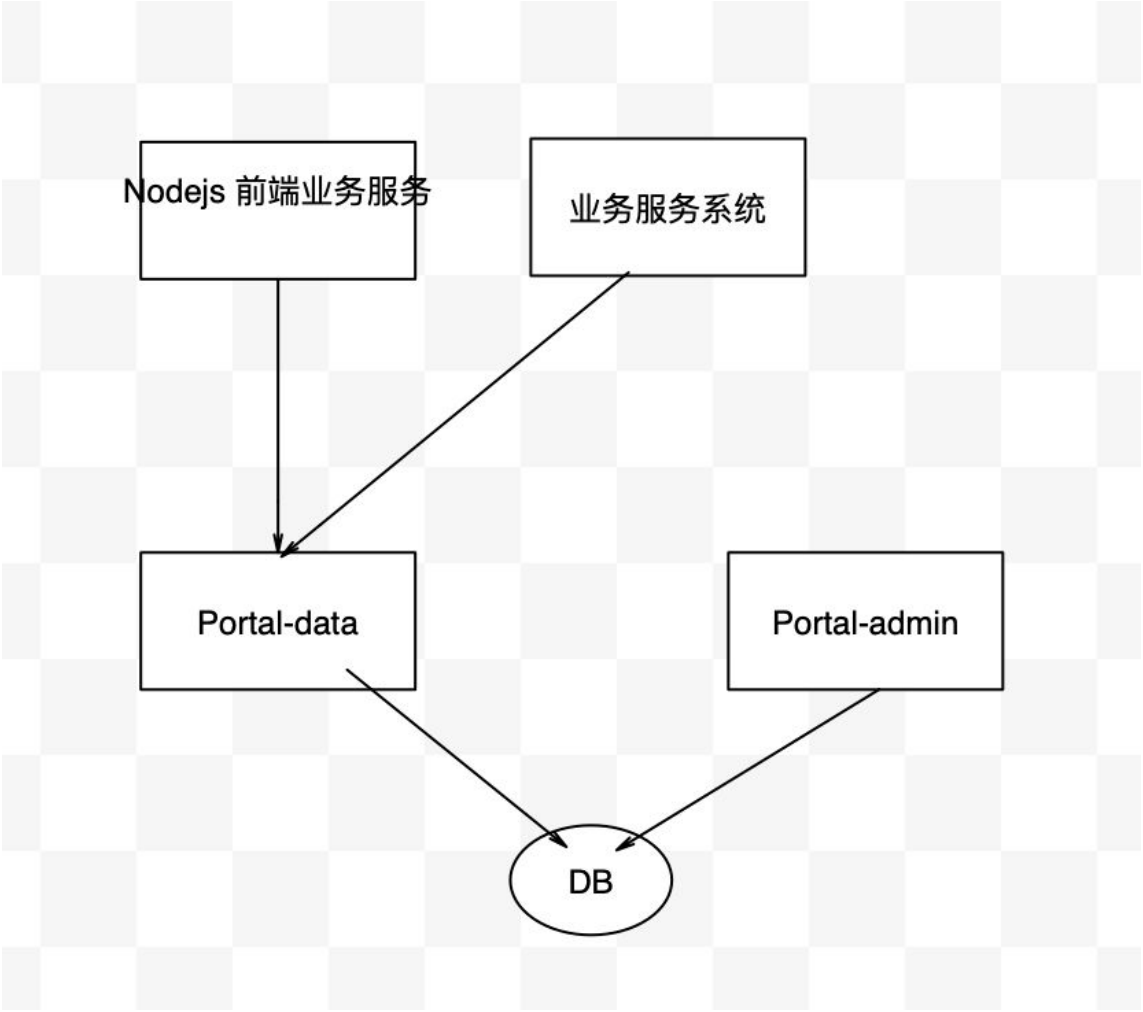


统一门户管理分三个组件：门户服务端 vole-portal、门户数据端 vole-portal-data、和门户基本组件 vole-portal-common，用微服务方式实现服务和数据分离。

- vole-portal: 主要负责门户的基本配置功能 如菜单，权限，系统，用户配置等；
- vole-portal-data: 负责对用户，权限，菜单的基本数据读取的管理。

服务和数据分离的好处是 - 对于任何前端程序都能接入相关 portal 服务，便于定制化管理界面。

如下图：

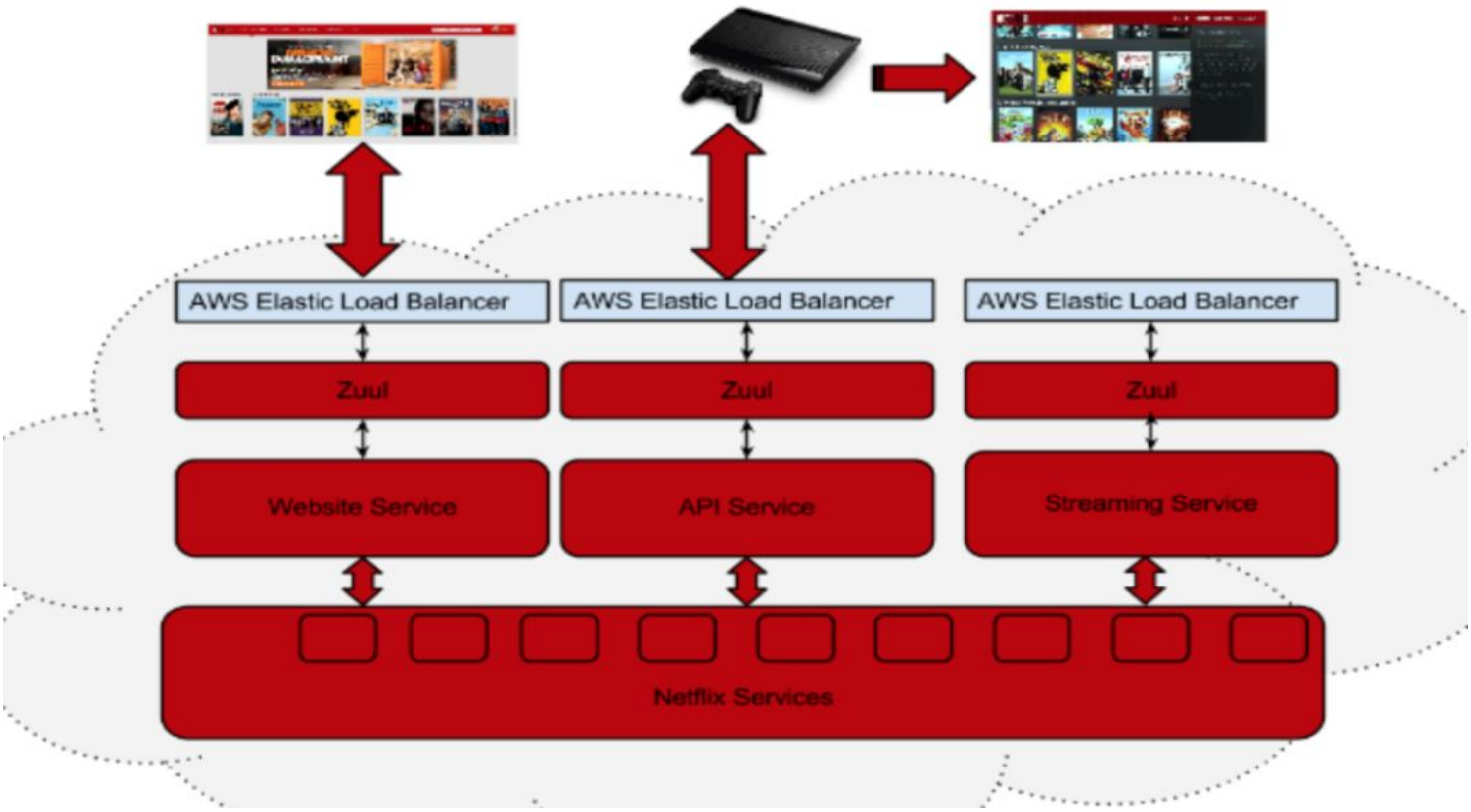


前端业务系统可以使用任何服务进行集成，便于前后端分离。

vole-gateway

微服务边缘网关 vole-gateway 是基于 zuul 改造的边缘网关服务，在 zuul 基础上添加了动态路由配置，服务鉴权，异常，安全等相关功能。

zuul 的服务方式如下图：



微服务网关 zuul 架构

为什么要使用边缘服务网关?

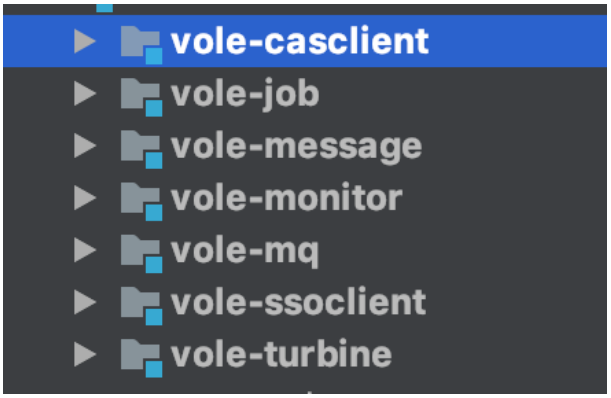
- 客户端直接和微服务交互，增加了复杂度;
- 某些场景下可能存在跨域;
- 如果一个功能点需要调用多个微服务，每个服务都需要身份认证，使得身份验证复杂冗余;
- 客户端直接和微服务交互，后期代码重构难度大。

基于以上问题，使用边缘网关使得整个微服务系统对外部服务的管理，尤其是移动端服务更加规范，简单，清晰。

vole-modules

vole-modules 包括了微服务的其他功能，比如任务、消息、监控等相关组件，可以帮助快速搭建微服务相关基础功能，提供基本骨架工程结合包。

如下图所示:



vole-job 使用的是 Elastic-Job-Lite；定时任务一般都是使用 quartz 或者 spring-task（ScheduledExecutorService），无论是使用 quartz 还是 spring-task，我们都会至少遇到两个痛点：

1. 不敢轻易跟着应用服务多节点部署，可能会重复多次执行而引发系统逻辑错误；
2. quartz 的集群仅仅只是用来 HA，节点数量的增加并不能给我们的每次执行效率带来提升，即不能实现水平扩展。

Elastic job 的主要功能有支持弹性扩容，通过 Zookepper 集中管理和监控 job，支持失效转移等。

vole-message 对钉钉，阿里云大鱼短信平台的基础服务能够快速的支持消息接入。

vole-mq 集成了对 kafka,rabbit,rocket 等主流 mq 客户端的集合包。

vole-monitor 和 vole-turbine 分别是对容器基础数据监控以及服务聚合熔断监控的组合。

结束语

通过精简或者重组 vole 微服务脚手架可以快速提炼出适应相关项目组的微服务脚手架工具，解决项目组大量的前期框架整合时间，提升开发效率和项目进度。

作者介绍

王一硼，苏宁科技集团消费者平台架构部技术专家，全面负责苏宁易购相关核心系统的优化及大促保障工作。 曾经在阿里，惠普等大型互联网和 IT 公司任职，有 10 多年的互联网一线研发经验，对高可用高并发及大规模分布式系统有深刻的认识和实践经验。

黄小虎，苏宁科技集团消费者平台购物流程架构负责人，全面负责苏宁易购商品详情页、购物车、大聚会等核心系统的优化及大促保障工作。对电商交易流程和业务有较深入的思考和研究，专注于高并发大型电商网站的架构设计、高可用的系统设计。曾主导和参与了 Commerce 系统拆分、商品详情页接入层优化、云信客服系统重构等重大技术攻关项目。现致力于打造苏宁易购新一代核心购物流程系统，希望将购物体验做到极致。