

目录

目录	1
简介	2
概述	2
系统拓扑图	2
代码结构	2
数据模型	3
技术选型	5
依赖环境	5
后端技术	5
前端技术	5
后端部署	5
前端部署	5
快速开始	6
快速开始	6
常见问题	8
集成开发	9
资源服务器,提供restful接口服务	9
接口调试	11
OAuth2	13
OAuth2 获取 AccessToken 并调用接口	13
OAuth2 获取 AccessToken	14
密码模式(password)	14
授权码模式(authorization_code)	15
客户端模式(client_credentials)	17
多用户认证中心	20
多用户认证中心	20
项目示例：	20
Jenkins持续集成	21
安装jenkins服务	21
构建后端服务	21
构建前端项目	22
Nginx部署前端项目	23

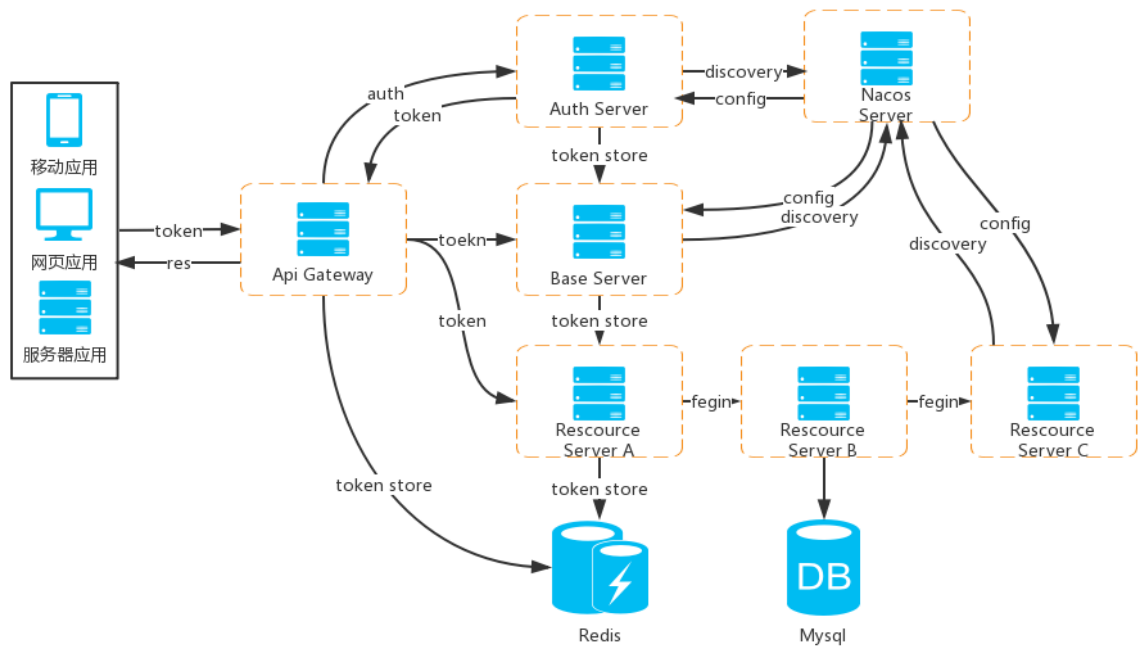
# 简介

## 概述

搭建基于OAuth2的开放平台、为APP端提供统一接口管控平台、为第三方合作伙伴的业务对接提供授信可控的技术对接平台。

- 统一API网关、访问鉴权、参数验签、外部调用更安全.
- 分布式架构,基于服务发现,Fegin(伪RPC)方式内部调用,更便捷.
- 深度整合SpringCloud+SpringSecurity+Oauth2,更细粒度、灵活的ABAC权限控制.
- 前后端分离方式开发应用，分工合作更高效!
- 代码合理封装、简单易懂、

## 系统拓扑图

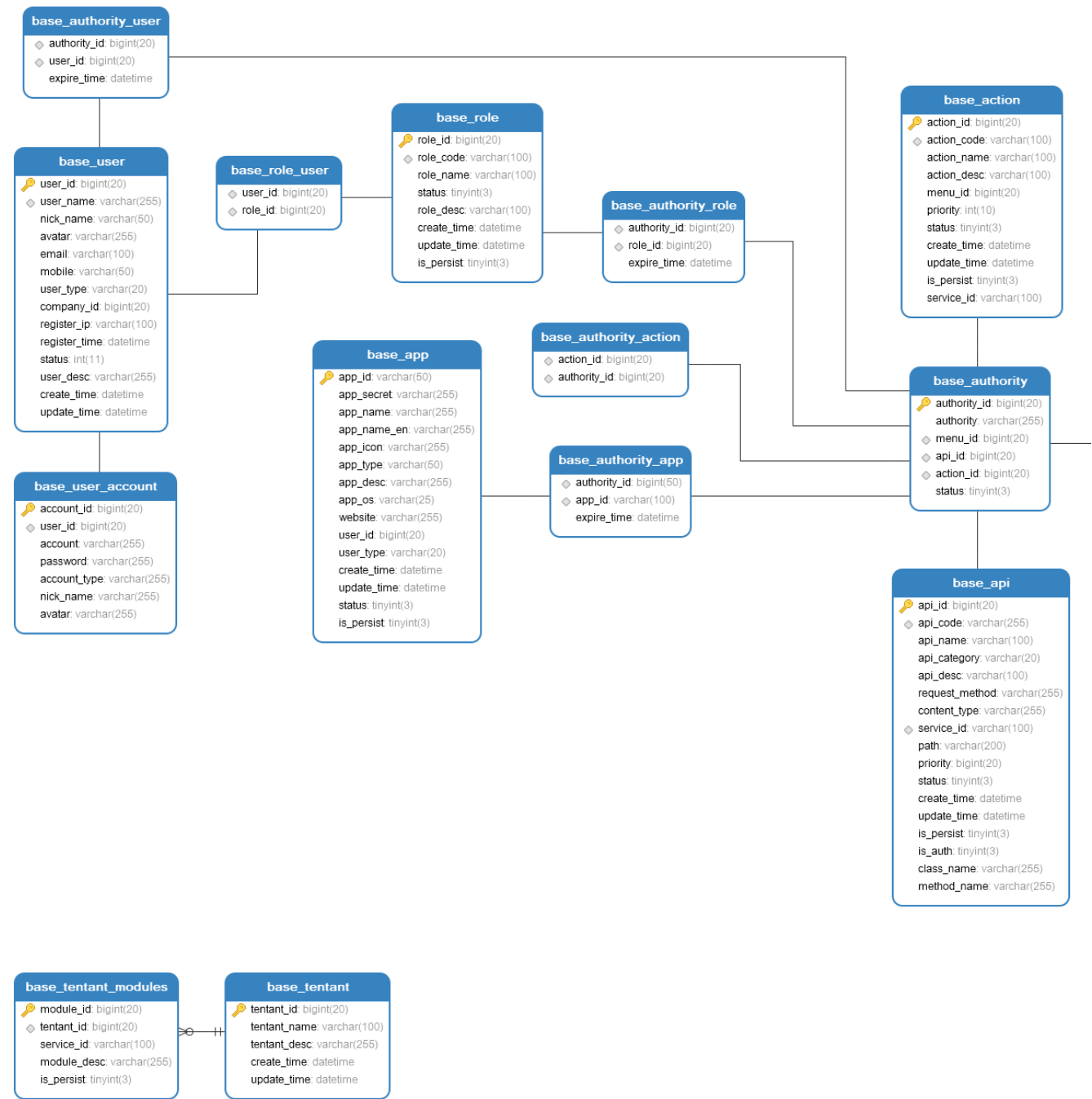


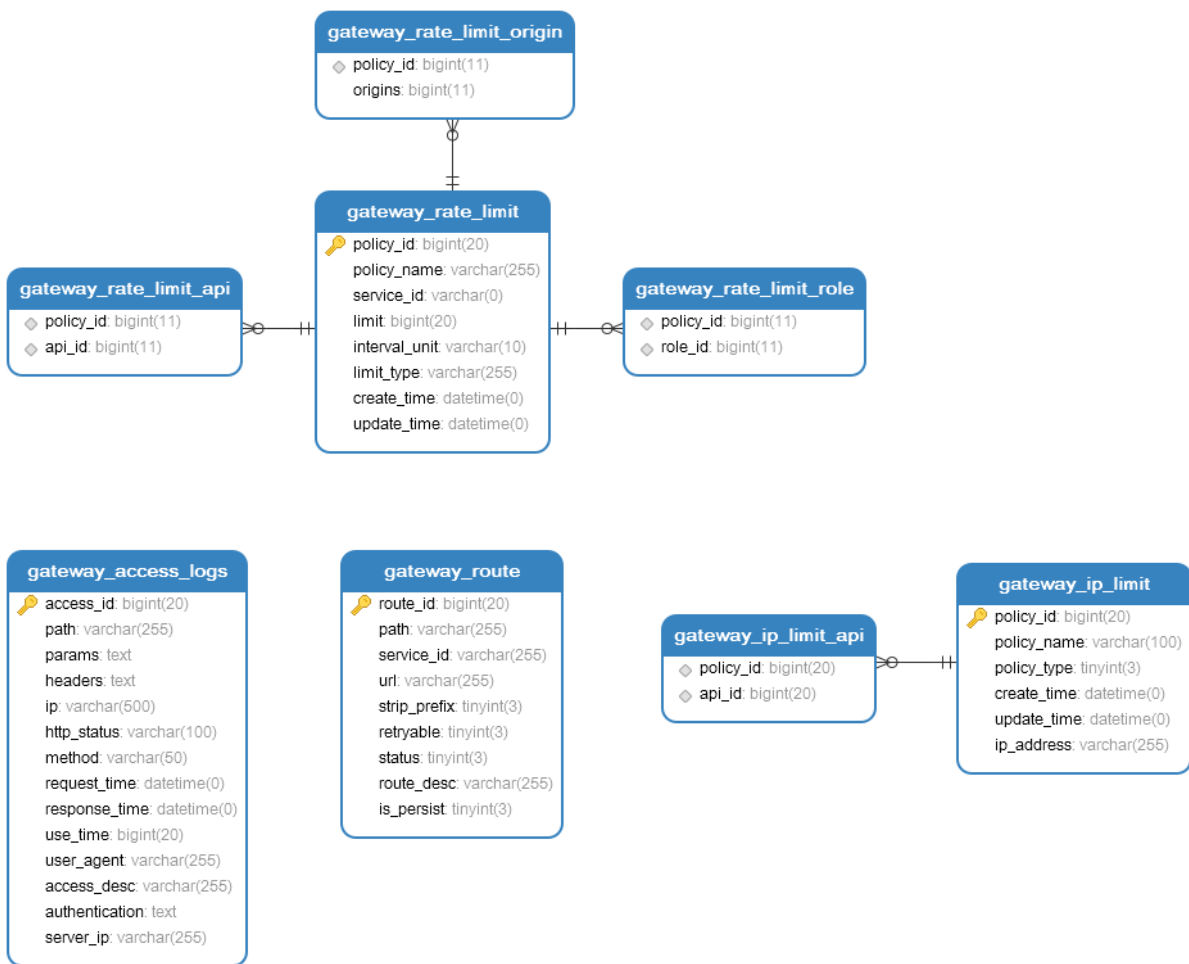
## 代码结构

```
open-cloud
├── docs
│   ├── bin          -- 执行脚本
│   ├── config       -- 公共配置,用于导入到nacos配置中心
│   ├── generator     -- mapper生成器
│   └── sql           -- sql文件
├── opencloud-app     -- 应用服务模块
│   └── app-opensite-provider -- 平台门户网站服务(port = 7211)
├── opencloud-common  -- 公共类和jar包依赖
│   ├── opencloud-common-core -- 提供微服务相关依赖包、工具类、全局异常解析等...
│   └── opencloud-common-starter -- SpringBoot自动扫描
├── opencloud-gateway -- 开放API服务模块
│   ├── opencloud-api-gateway -- API开放网关-基于SpringCloudGateway-(port = 8888)
│   └── opencloud-api-gateway-zuul -- (较为稳定推荐使用) API开放网关-基于Zuul-(port = 8888)
├── opencloud-platform -- 平台服务模块
│   ├── opencloud-base-client -- 平台基础服务接口
│   └── opencloud-base-provider -- 平台基础服务(port = 8233)
```

| opencloud-auth-client -- 平台认证服务接口  
| opencloud-auth-provider -- 平台认证服务(port = 8211)  
| opencloud-msg-client -- 消息服务接口  
| opencloud-msg-provider -- 消息服务(port = 8266)  
| opencloud-scheduler-client -- 任务调度接口  
| opencloud-scheduler-provider -- 任务调度服务(port = 8501)  
| opencloud-bpm-client -- 工作流接口  
| opencloud-bpm-provider -- 工作流服务(port = 8255)

数据模型





# 技术选型

## 依赖环境

- nodejs
- redis(缓存服务)
- rabbitMq(消息服务)
- nacos(阿里巴巴服务注册和配置管理)
- mysql(数据库)

## 后端技术

- maven
- springcloud
- spring-security-oauth2
- mybatis
- mybatis-plus
- swagger2
- ...

## 前端技术

- vue.js
- iview (前端框架)

## 后端部署

- jar包形式部署,使用undertow代替原有tomcat
- jenkins持续集成

## 前端部署

- nginx

# 快速开始

快速开始

上手难度：★★★★

本项目基于springCloud打造的分布式快速开发框架. 需要了解SpringCloud, SpringBoot开发, 分布式原理。

1. 准备环境

- Java1.8
- 阿里巴巴Nacos服务发现和注册中心 [nacos.io](https://nacos.io)
- Redis
- RabbitMq （需安装rabbitmq\_delayed\_message\_exchange插件 [下载地址](#)）
- Mysql
- Maven
- Nodejs

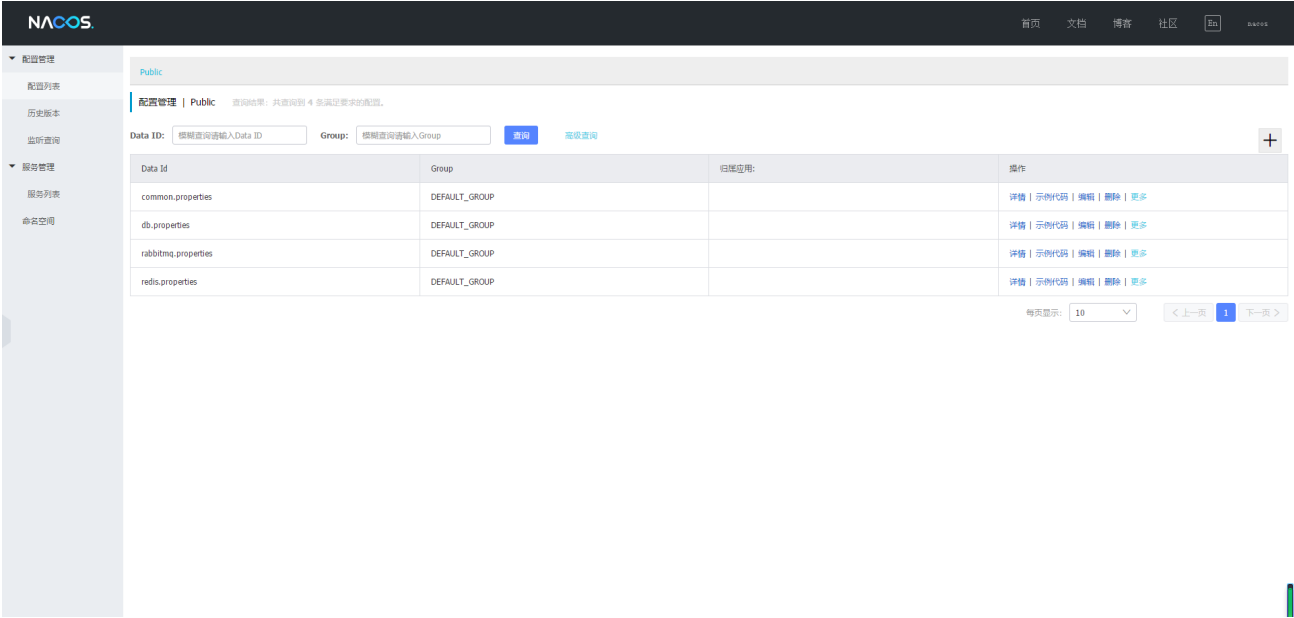
2. 执行创建数据库open-platform并执行sql脚本

- docs/sql/oauth2.sql
- docs/sql/base.sql
- docs/sql/gateway.sql
- docs/sql/quartz.sql && scheduler.sql

3. 启动nacos服务发现&配置中心,新建公共配置文件

- 访问 <http://localhost:8848/nacos/index.html>
- 新建配置文件 ```
  - 项目目录/docs/config/db.properties > db.properties
  - 项目目录/docs/config/rabbitmq.properties > rabbitmq.properties
  - 项目目录/docs/config/redis.properties > redis.properties
  - 项目目录/docs/config/common.properties > common.properties

如图:



4. 修改主pom.xml

初始化maven项目

```
maven clean install
```

本地启动,默认不用修改

```
<!--Nacos配置中心地址-->
<config.server-addr>127.0.0.1:8848</config.server-addr>
<!--Nacos配置中心命名空间,用于支持多环境.这里必须使用ID,不能使用名称,默认为空-->
<config.namespace></config.namespace>
<!--Nacos服务发现地址-->
<discovery.server-addr>127.0.0.1:8848</discovery.server-addr>
```

## 5. 本地启动(顺序启动)

- i. BaseApplication
- ii. AuthApplication
- iii. ZuulGatewayApplication(推荐) 或 ApiGatewayApplication

访问 <http://localhost:8888>

## 6. 前端启动

```
npm install
npm run dev
```

访问 <http://localhost:8080>

## 7. 项目打包部署

maven多环境打包,并替换相关变量

```
mvn clean install package -P {dev|test|online}
```

项目启动

```
./docs/bin/startup.sh {start|stop|restart|status} opencloud-base-provider.jar
./docs/bin/startup.sh {start|stop|restart|status} opencloud-auth-provider.jar
./docs/bin/startup.sh {start|stop|restart|status} opencloud-api-gateway-zuul.jar
```

## 常见问题

- 项目启动失败？

1. 先检查nacos服务注册发现是否启动成功。
2. 检查相关公共配置是否完全导入到nacos中。
3. 新建项目启动失败, 检查是否引入jar冲突,并排除相关依赖.

- 提示访问权限不足,后台设置步骤？

1. 后台权限 1.1 系统管理 > 菜单资源 > 功能权限 > 接口授权 > 勾选当前操作涉及到的相关接口 > 保存 1.2 系统管理 > 角色信息 > 分配菜单 > 勾选菜单和功能按钮 > 保存 1.3 系统管理 > 系统用户 > 分配角色 > 保存
2. 应用权限(客户端模式) 应用管理 > 开发配置 > 勾选客户端模式 > 更多-接口授权 > 绑定相关接口资源

- 资源服务器解析token？

1. (默认)远程校验解析token. 缺点:需创建客户端信息,http方式,性能较差.

```
ResourceServerTokenServices tokenService = OpenHelper.buildRemoteTokenServices(OpenCommonProperties properties)
```

2. (默认)jwt校验解析token. 缺点:需提供jwt签名或密钥,生存token过长(cookie有时存放不下),且base64加密无法存放敏感数据,不方便拓展.

```
ResourceServerTokenServices tokenService = OpenHelper.buildJwtTokenServices(OpenCommonProperties properties)
```

3. (推荐使用)redis校验解析token. 无需创建任何客户端和密钥,读取性能优,可存放复杂的认证信息. 缺点:必须使用同一个redisDbIndex.

```
ResourceServerTokenServices tokenService = OpenHelper.buildRedisTokenServices(RedisConnectionFactory redisConnectionFactory)
```



# 集成开发

资源服务器,提供restful接口服务

## 1.创建新maven项目

```
<!-- 引入公共包,需发布到本地仓库或私服 -->
<dependency>
  <artifactId>opencloud-common-starter</artifactId>
  <groupId>com.opencloud</groupId>
  <version>${opencloud.common.version}</version>
</dependency>
```

## 2.配置 bootstrap.yml

```
server:
  port: 8266
spring:
  application:
    name: ${artifactId}
  cloud:
    nacos:
      config:
        namespace: ${config.namespace}
        refreshable-dataids: common.properties
        server-addr: ${config.server-addr}
        shared-dataids: common.properties,db.properties,redis.properties,rabbitmq.properties
      discovery:
        server-addr: ${discovery.server-addr}
  main:
    allow-bean-definition-overriding: true
  mvc:
    throw-exception-if-no-handler-found: true
  resources:
    add-mappings: false
  profiles:
    active: ${profile.name}

management:
  endpoints:
    web:
      exposure:
        include: refresh,health
opencloud:
  swagger2:
    description: 平台消息服务
    enabled: true
    title: 平台消息服务
```

## 3. 创建MyServiceApplication.java

```
//开启feign RPC远程调用
@EnableFeignClients
// 开启服务发现
@EnableDiscoveryClient
@SpringBootApplication
public class MyServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(MyServiceApplication.class, args);
    }
}
```

## 4.创建ResourceServerConfiguration.java 资源服务配置

```

@Configuration
@EnableResourceServer
public class ResourceServerConfiguration extends ResourceServerConfigurerAdapter {
    @Autowired
    private RedisConnectionFactory redisConnectionFactory;

    @Override
    public void configure(ResourceServerSecurityConfigurer resources) throws Exception {
        // 构建redis获取token,这里是为了支持自定义用户信息转换器
        resources.tokenServices(OpenHelper.buildRedisTokenServices(redisConnectionFactory));
    }

    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.IF_REQUIRED)
            .and()
            .authorizeRequests()
            // 内部访问直接放行
            .antMatchers("/v1/**").permitAll()
            // 只有拥有actuator权限可执行远程端点
            .requestMatchers(EndpointRequest.toAnyEndpoint()).hasAnyAuthority(CommonConstants.AUTHORITY_ACTUATOR)
            .anyRequest().authenticated()
            .and()
            //认证鉴权错误处理,为了统一异常处理。每个资源服务器都应该加上。
            .exceptionHandling()
            .accessDeniedHandler(new OpenAccessDeniedHandler())
            .authenticationEntryPoint(new OpenAuthenticationEntryPoint())
            .and()
            .csrf().disable();
    }
}

```

## 5.启动项目

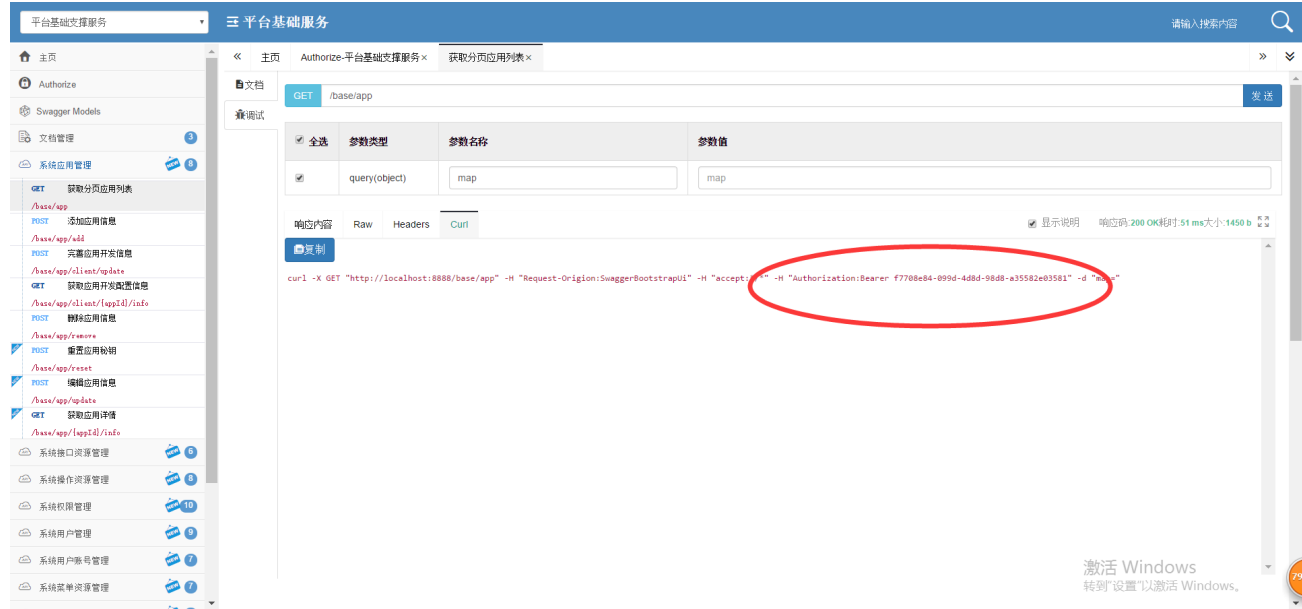
# 接口调试

基于swagger2+swagger-bootstrap-ui 官方文档

1. 设置全局认证token 输入值: Bearer {access\_token}



2. 访问接口



平台基础支撑服务

三 平台基础服务

请输入搜索内容

Authorize

Swagger Models

文档管理

系统应用管理

获取分页应用列表

添加应用信息

完善应用开发信息

获取应用开发配置信息

删除应用信息

重置应用密钥

编辑应用信息

获取应用详情

系统接口资源管理

系统操作资源管理

系统权限管理

系统用户管理

系统用户账号管理

系统菜单资源管理

Authorize-平台基础支撑服务 x

获取分页应用列表 x

GET /base/app

发送

参数类型

参数名称

参数值

query(object)

map

map

响应内容

Raw

Headers

Curl

1 {

2 "code": 0,

3 "message": "请求成功",

4 "path": "",

5 "data": {

6 "records": [

7 {

8 "appDesc": "",

9 "appIcon": "https://obwK8Baw.qnssl.com/7eb99af9b5f317c812f08b5112fd669a/avatar",

10 "appId": "1553588682729",

11 "appName": "测试应用",

12 "appSecret": "",

13 "appType": "Server",

14 "appSecret": "1a616ba3f91141efalc44a1ce725edc",

15 "createTime": "2019-03-27 00:23:49",

16 "isPersist": 0,

17 "status": 1,

18 "updateTime": "2019-03-27 00:23:49",

19 "userId": "521077655146233856",

20 "userName": "admin",

21 "userType": "platform",

22 "website": "http://www.test.com"

23 },

24 },

25 {

26 "appDesc": "资源服务器",

27 "appIcon": "",

28 "appId": "155323783265",

29 "appName": "资源服务器",

30 "appSecret": "",

31 "appSecret": "2cde1ea60fe4af1987f94caa13f29a2",

32 "createTime": "2018-11-13 01:48:45",

33 "isPersist": 1,

34 "status": 1,

35 }

36 }

外应码: 0-请求处理成功

请求消息

请求路径

响应数据

显示说明

响应码 200 OK耗时: 51 ms大小: 1450 B

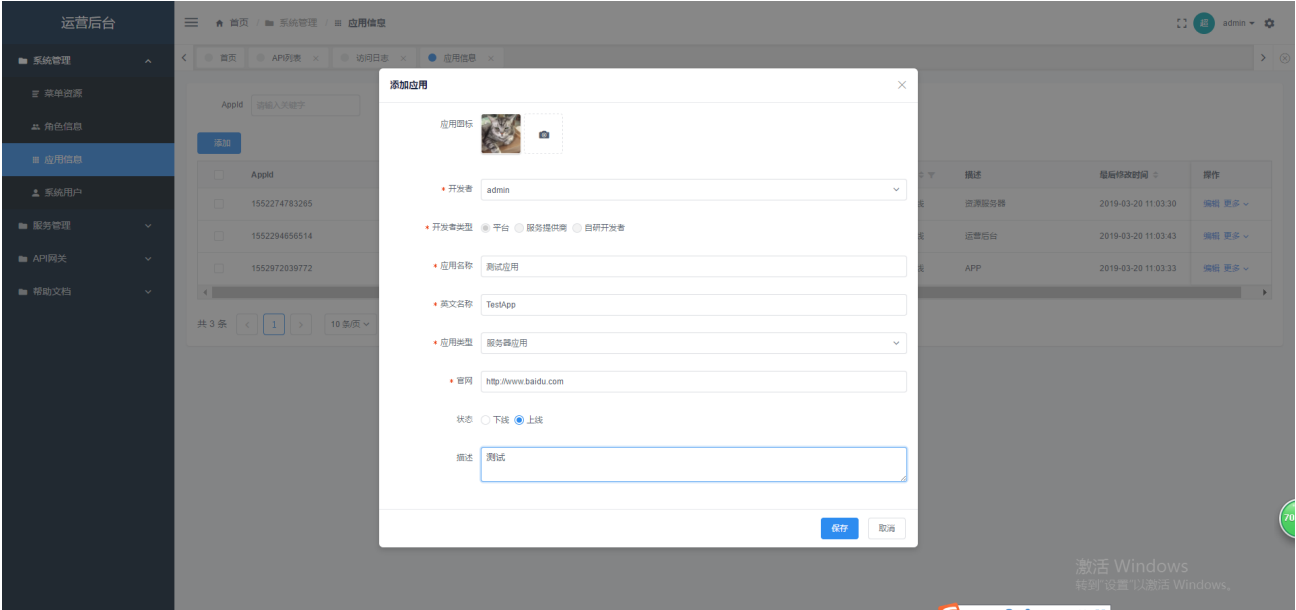
激活 Windows

转到“设置”以激活 Windows。

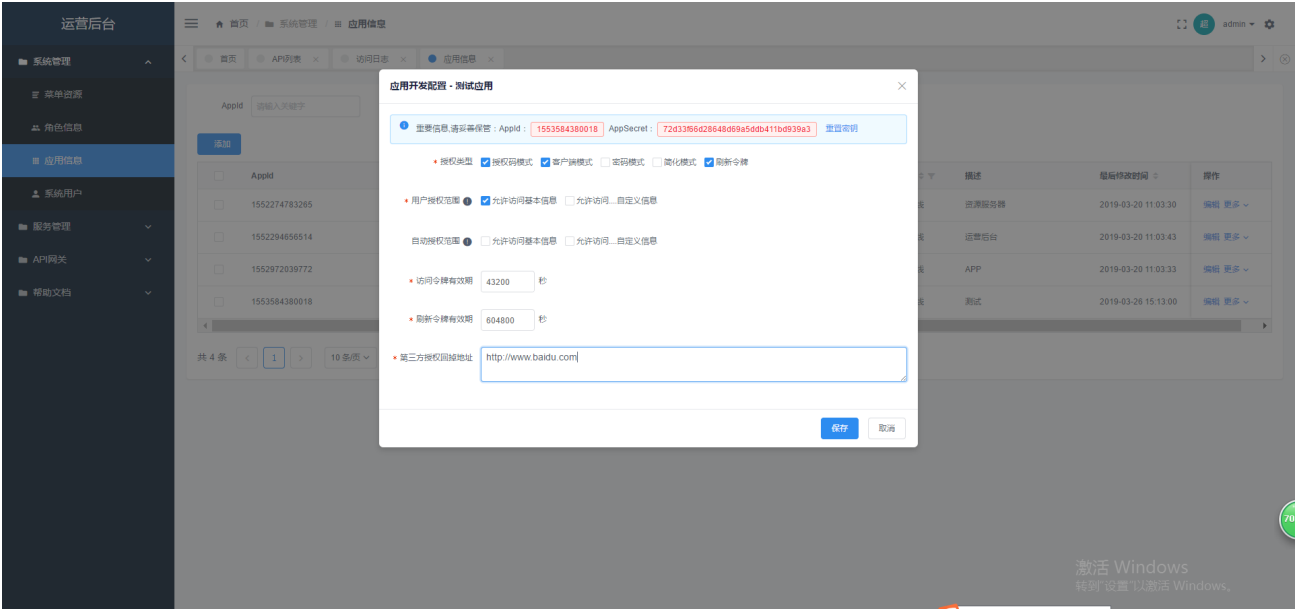
# OAuth2

## OAuth2 获取 AccessToken 并调用接口

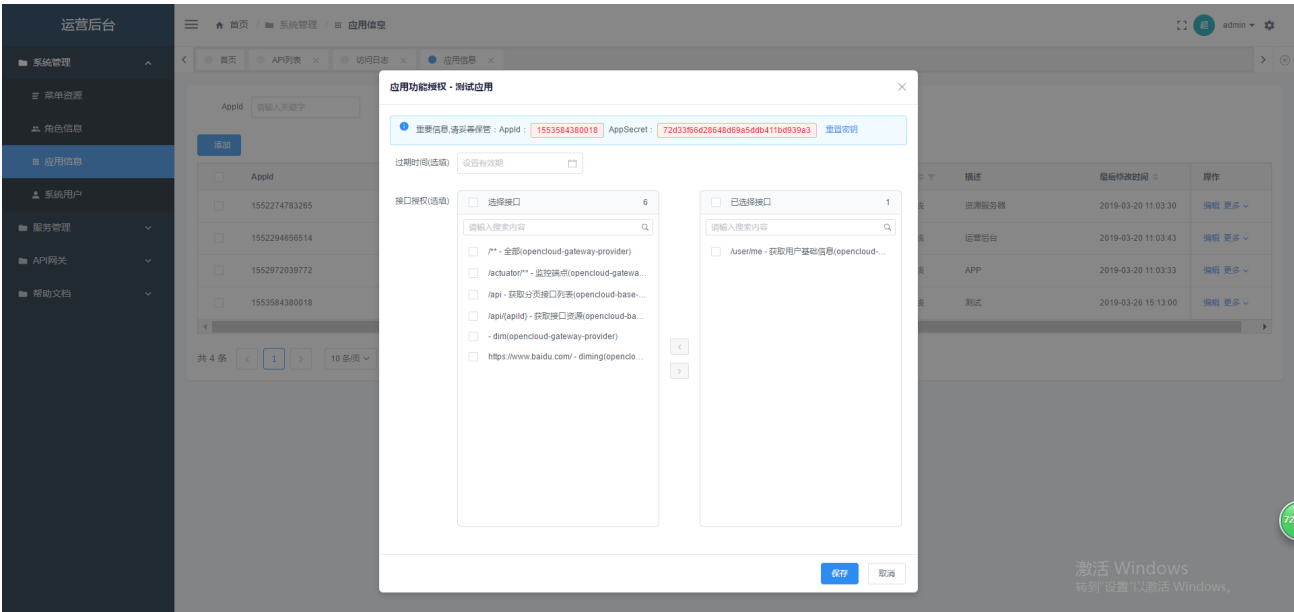
- 创建一个新应用



- 配置开发信息 根据实例需求勾选,授权方式



- 接口授权,勾选需要授权的接口 访问api网关时,将验证该权限。 否则提示权限不足



OAuth2 获取 AccessToken

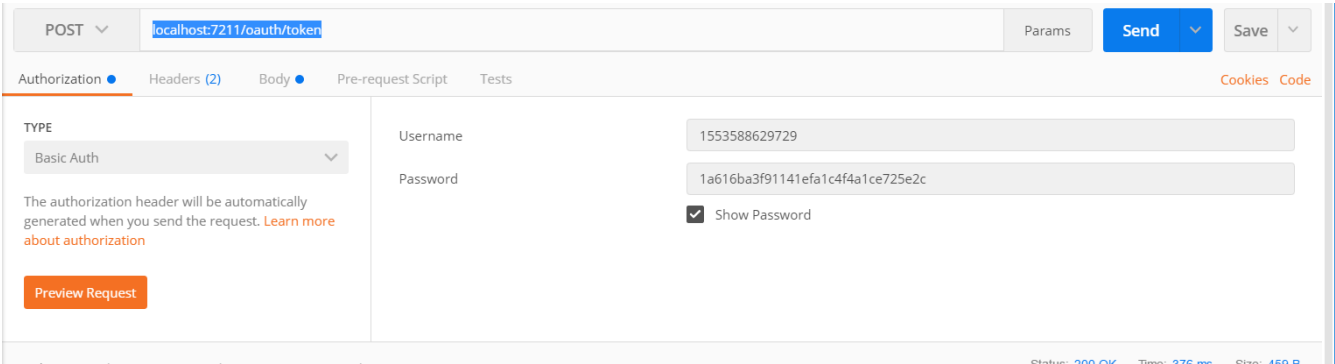
- 使用postman测试

应用信息  
AppId(client\_id) : 1553588629729  
AppSecret(client\_secret) : 1a616ba3f91141efa1c4f4a1ce725e2c

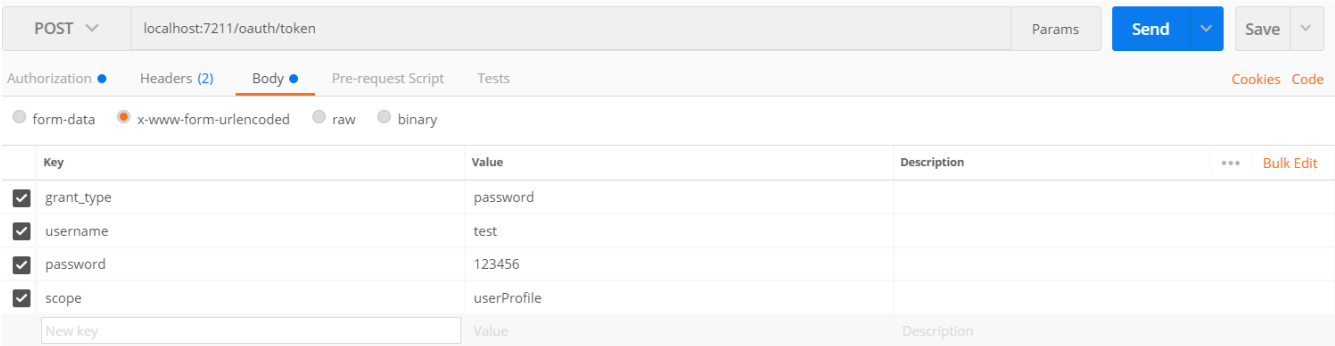
密码模式(password)

http://localhost:8211/oauth/token

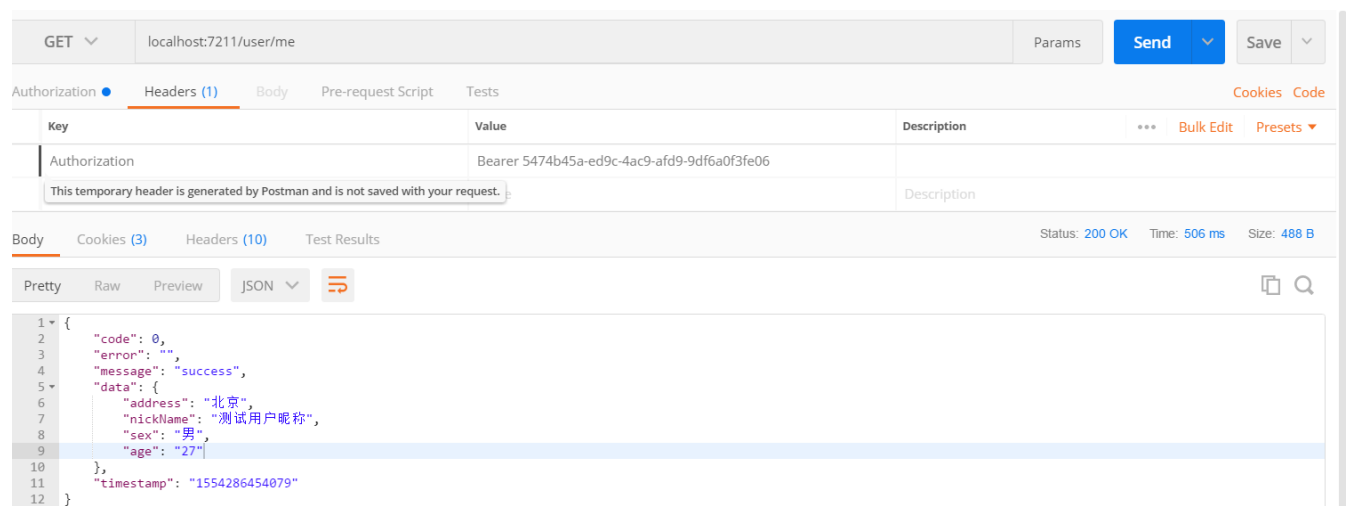
1. 首先设置请求头basic方式配置client\_id和client\_secret



2. 输入用户username和password



2. 获取使用access\_token用户信息

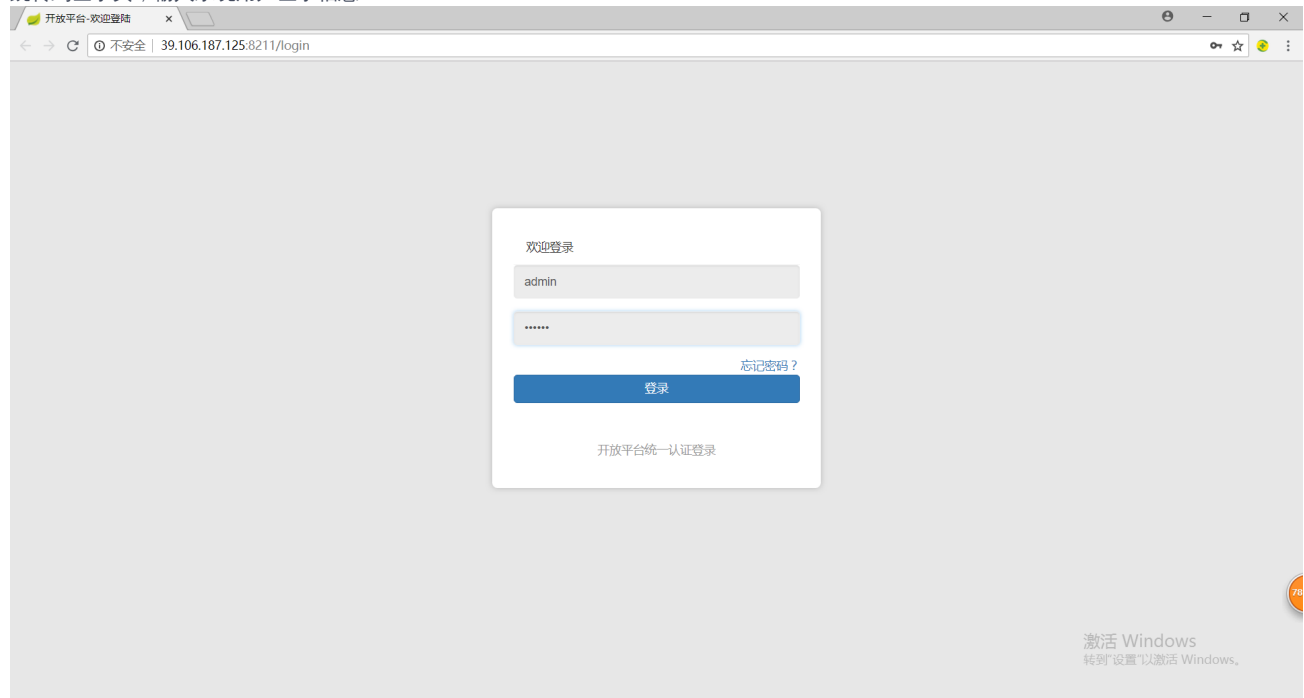


授权码模式(authorization\_code)

- 浏览器访问，引导用户跳转到登录页(适合sso单点登录系统接入)

[http://localhost:8211/oauth/authorize?response\\_type=code&client\\_id=1553588629729&redirect\\_uri=http://www.baidu.com](http://localhost:8211/oauth/authorize?response_type=code&client_id=1553588629729&redirect_uri=http://www.baidu.com)

- 跳转到登录页，输入系统用户登录信息



- 用户确认授权信息



测试应用 将获得以下权限：

☒ 获得您的昵称、头像、等基础信息

授权后表明你已同意 服务协议

授权

- 重定向到回调地址,获得code



Baidu 百度

百度一下



百度

把百度设为主页 关于百度 About Baidu 百度推广  
©2019 Baidu 使用百度前必读 意见反馈 京ICP证030173号 京公网安备11000002000001号



激活 Windows  
转到“设置”以激活 Windows。

- 使用postman通过code获取access\_token ,



POST localhost:8211/oauth/token

Authorization Headers Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary

Key	Value	Description
grant_type	authorization_code	
client_id	1553588629729	
client_secret	1a616ba3f91141efa1c4f4a1ce725e2c	
redirect_uri	http://www.baidu.com	
code	5ZCobU	

Body Cookies (2) Headers (9) Test Results

Status: 200 OK Time: 442 ms Size: 459 B

```

1 {
2   "access_token": "80e3b44a-807f-47e6-bb37-44a50dc62be2",
3   "token_type": "bearer",
4   "refresh_token": "34b43926-c79d-4d81-bce0-4724b576d5ed",
5   "expires_in": 43199,
6   "scope": "userProfile"
7 }
  
```

- 使用access\_token获取已授权资源

GET localhost:8888/base/api/all

Authorization Headers Body Pre-request Script Tests

TYPE OAuth 2.0

Access Token

80e3b44a-807f-47e6-bb37-44a50dc62be2

Get New Access Token

The authorization data will be automatically generated when you send the request. [Learn more about authorization](#)

Add authorization data to Request Headers

Preview Request

Body Cookies (2) Headers (11) Test Results

Status: 200 OK Time: 850 ms Size: 40.51

```

136 {
137   "updateTime": "2019-05-26 14:40:42",
138   "isPersist": 1,
139   "isOpen": 0,
140   "isAuth": 1,
141   "requestMethod": "post"
142 },
143 {
144   "apiId": "555567987034161152",
145   "apiCode": "com.github.lyd.base.provider.controller.BaseAppController.getAppDevInfo",
146   "apiName": "获取应用开发配置信息",
147   "serviceId": "opencloud-base-provider",
148   "apiCategory": "default",
  
```

### 客户端模式(client\_credentials)

[http://localhost:8211/oauth/token?grant\\_type=client\\_credentials&client\\_id=1553588629729&client\\_secret=1a616ba3f91141efa1c4f4a1ce725e2c](http://localhost:8211/oauth/token?grant_type=client_credentials&client_id=1553588629729&client_secret=1a616ba3f91141efa1c4f4a1ce725e2c)

- 获取客户端token

localhost:8080/access localhost:8080/test localhost:8080/test localhost:8888/base/a + ... No Environment

POST localhost:8211/oauth/token Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Cookies

form-data x-www-form-urlencoded raw binary

Key	Value	Description
grant_type	client_credentials	
client_id	1553588629729	
client_secret	1a616ba3f91141efa1c4f4a1ce725e2c	

New key Value Description

Body Cookies (2) Headers (9) Test Results Status: 200 OK Time: 298 ms Size: ...

Pretty Raw Preview JSON

```
1 {
2   "access_token": "70fe31b3-88fc-4ab6-8940-24fc1a756a1b",
3   "token_type": "bearer",
4   "expires_in": 43188,
5   "scope": "userProfile"
6 }
```

- 访问未授权资源提示权限不足!

GET localhost:8888/base/api/all Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Cookies

TYPE OAuth 2.0

The authorization data will be automatically generated when you send the request. [Learn more about authorization](#)

Add authorization data to Request Headers

Preview Request

Access Token 70fe31b3-88fc-4ab6-8940-24fc1a756a1b Available Tokens

Get New Access Token

Body Cookies (2) Headers (11) Test Results Status: 403 Forbidden Time: 579 ms Size: 504

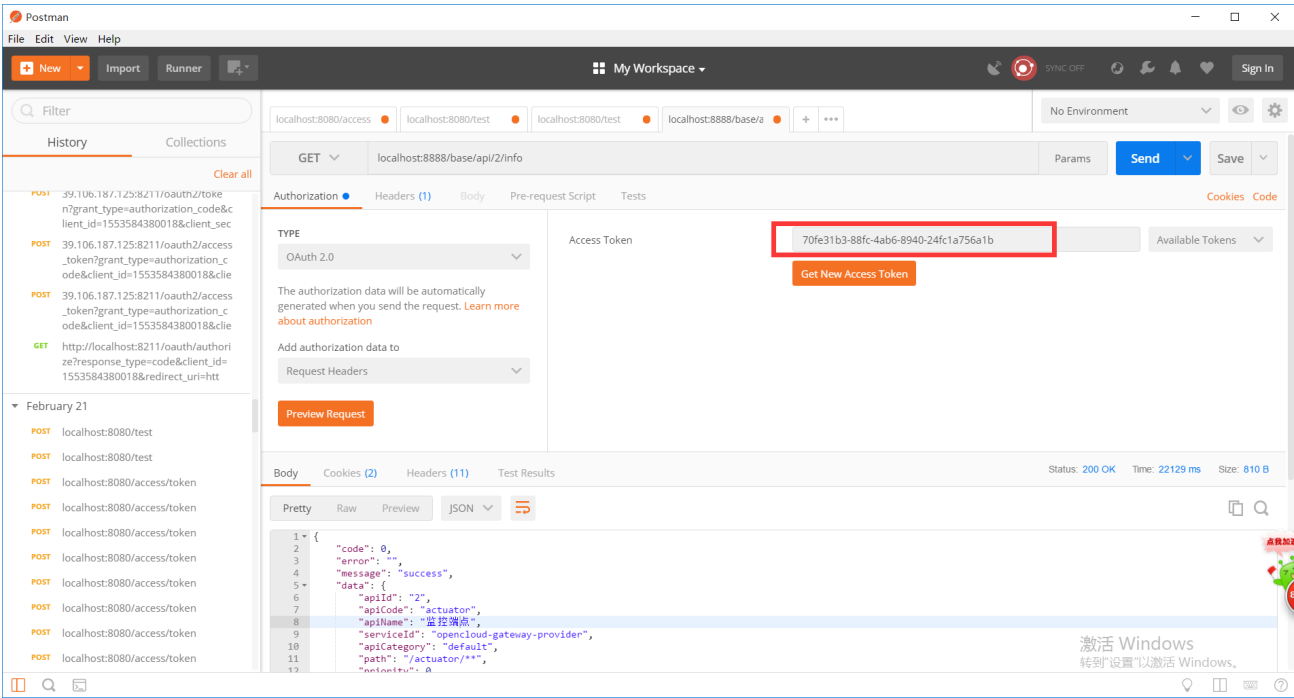
Pretty Raw Preview JSON

```
1 {
2   "code": 4030,
3   "error": "access_denied",
4   "message": "权限不足, 拒绝访问!",
5   "path": "/base/api/all",
6   "timestamp": 1553591162926
7 }
```

未授权资源

- 访问已授权资源,正常返回数据

激活 Windows



# 多用户认证中心

## 多用户认证中心

- 不同点：用户表都是完全独立的,登录方式也是多种多样(密码登录,手机验证码登录,等等...)
- 共同点:每个认证中心,共享oauth\_client\_details(客户端信息)
- 优点:
  1. 从项目层隔离开,统一token协议,在向下游微服务传递时,使用同一方式读取token信息.
  2. app应用直接具备第三方授权登录能力, 例如：微信,qq等第三方登录. 方便后期拓展。

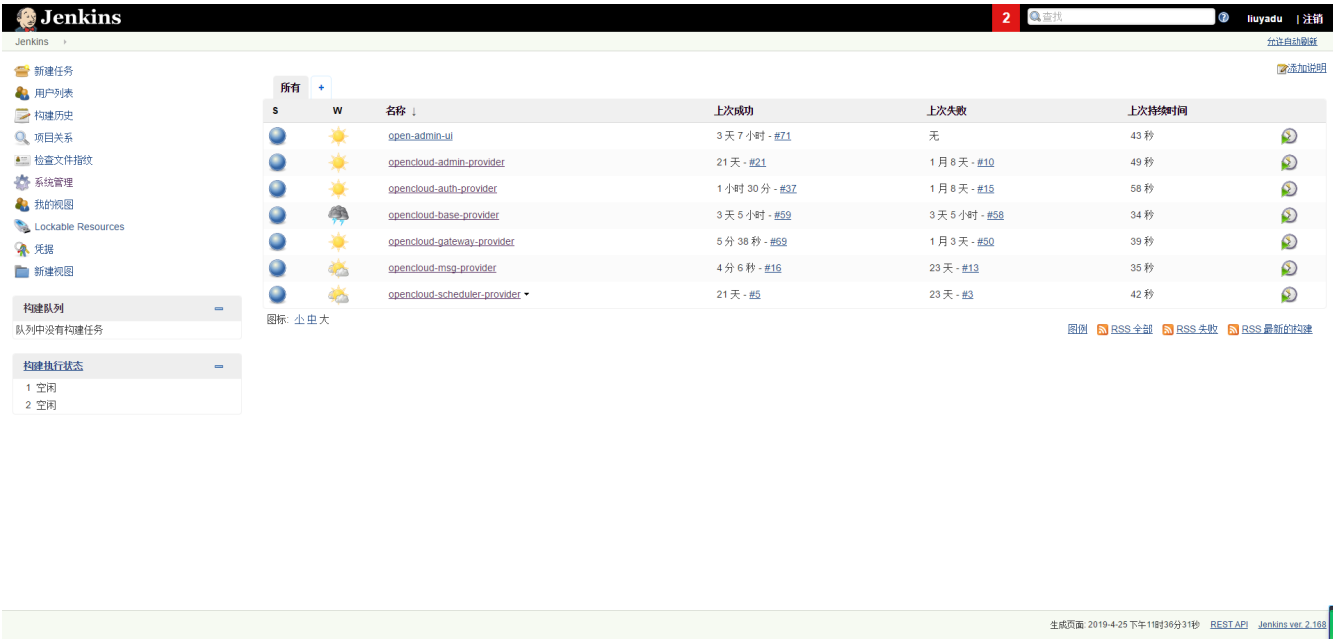
## 项目示例：

- opencloud-auth-provider 平台管理员认证服务和资源服务
- app-opensite-provider 门户开发者认证服务和资源服务

# Jenkins持续集成

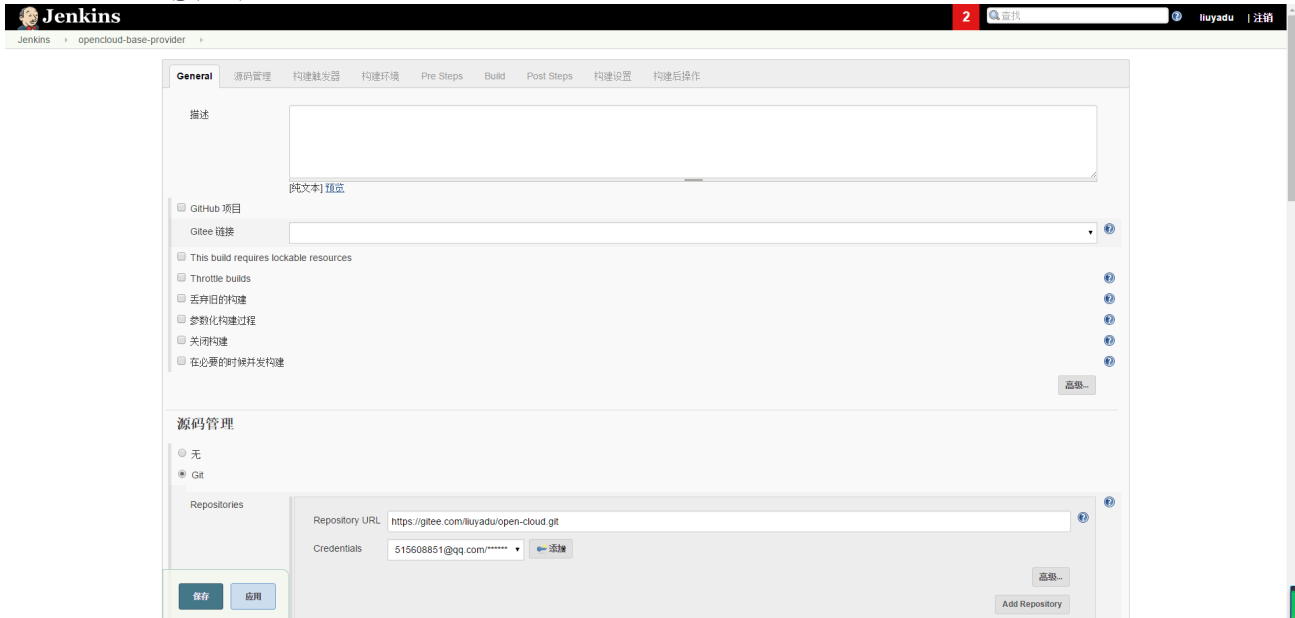
## 安装jenkins服务

### 1. 安装相关插件



## 构建后端服务

- 安装maven插件
- 构建一个maven项目
- 项目名作为任务名称
- 配置代码拉取地址(git/svn)



- 配置项目构建路径

**Pre Steps**

Add pre-build step

**Build**

Root POM:

Goals and options:

**Post Steps**

☒ Run only if build succeeds
 ☐ Run only if build succeeds or is unstable
 ☐ Run regardless of build result

- 配置构建完成后,执行命令 (这里执行的本机命令,可使用ssh管道执行远程脚本)

**Post Steps**

☒ Run only if build succeeds
 ☐ Run only if build succeeds or is unstable
 ☐ Run regardless of build result

Should the post-build steps run only for successful builds, etc.

**执行 shell**

命令

```
cp -rf $WORKSPACE/opencloud-upms/$JOB_NAME/target/$JOB_NAME-1.0.0.jar /root/deploy
cd /root/deploy
./startup.sh restart $JOB_NAME-1.0.0.jar
```

[查看 可用的环境变量列表](#)

高级...

## 构建前端项目

- 安装nodejs插件
- 构建一个自由风格的软件项目
- 项目名作为任务名称
- 配置代码拉取地址(git/svn)
- 配置执行命令

Jenkins > open-admin-ui

General 源码管理 构建触发器 构建环境 构建 构建后操作

☐ Send files or execute commands over SSH before the build starts  
☐ Send files or execute commands over SSH after the build runs  
☐ Abort the build if it's stuck  
☐ Add timestamps to the Console Output  
☒ Provide Node & npm bin/ folder to PATH

NodeJS Installation:

Specify needed nodejs installation where npm installed packages will be provided to the PATH

npmrc file:

☐ With Ant

**构建**

**执行 shell**

命令

```
npm install
npm run build
cp -rf dist/* /root/www/admin
```

[查看 可用的环境变量列表](#)

高级...

增加构建步骤

**构建后操作**

增加构建后操作步骤

保存 应用

# Nginx部署前端项目

- 安装nginx服务
- 配置nginx.conf

```
user root;
worker_processes 1;

#error_log logs/error.log;
#error_log logs/error.log notice;
#error_log logs/error.log info;

#pid logs/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include mime.types;
    default_type application/octet-stream;

    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
        '$status $body_bytes_sent "$http_referer" '
        '"$http_user_agent" "$http_x_forwarded_for"';

    access_log logs/access.log main;

    sendfile on;
    #tcp_nopush on;

    #keepalive_timeout 0;
    keepalive_timeout 65;

    #开启或关闭gzip on off
    gzip on;

    #不使用gzip IE6
    gzip_disable "msie6";

    #gzip压缩最小文件大小，超出进行压缩（自行调节）
    gzip_min_length 100k;

    #buffer 不用修改
    gzip_buffers 4 16k;

    #压缩级别:1-10，数字越大压缩的越好，时间也越长
    gzip_comp_level 3;

    # 压缩文件类型
    gzip_types text/plain application/javascript text/css application/xml text/javascript application/x-httpd-php image/jpeg image/gif image/png;

    #跟Squid等缓存服务有关，on的话会在Header里增加 "Vary: Accept-Encoding"
    gzip_vary off;

    underscores_in_headers on;

    server {
        listen 80;
        server_name localhost;

        #charset koi8-r;

        access_log logs/host.access.log main;

        #默认目录
        location / {
```

```
    root html;
    index index.html;
}

#vue二级目录代理
location /admin {
    alias /root/www/admin;
    index index.html;
    try_files $uri $uri/ /index.html last;
}

#api网关代理
location /api {
    rewrite ^/api/(.*)$ /$1 break;
    proxy_pass http://127.0.0.1:8888;
    proxy_set_header Host $host;
    proxy_set_header User-Agent $http_user_agent;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $remote_addr;
    proxy_set_header authorization $http_authorization;
}

#error_page 404 /404.html;

# redirect server error pages to the static page /50x.html
error_page 500 502 503 504 /50x.html;
location = /50x.html {
    root html;
}

}
```



