### Ory Hydra 应用示例

## 1. 简介

最近从其他同事得知项目用到了OAuth2认证方式,使用的认证服务就是Hydra。出于对技术的提前积累特对该服务进行了简单的调研。

Ory Hydra是经过强化的、经过OpenID认证的OAuth2服务软件。其针对低延迟、高吞吐量和低资源消耗进行了优化。其不是身份提供者(也就是不提供用户注册、用户登录和密码重置等功能),而是通过登录和同意应用程序连接到现有身份提供者。

本文的目的是积累调研过程,防止后面重复踩坑。

## 2. 服务启动

为了方便的搭建起Hydra开发环境,可以使用docker-compose工具启动Hydra。

以下为docker-compose.yml文件的内容

```
version: "3.7"
services:
     hydra:
           image: oryd/hydra:v2.0.2
           ports:
                - "4444:4444" # Hydra对外服务端口。
                - "4445:4445" # Hydra管理端口。所有访问地址是admin开头的使用该端口。
                - "5555:5555"  # Port for hydra token user
           command: serve -c /etc/config/hydra/hydra.yml all --dev
           volumes:
                - type: bind
                      source: ./config
                      target: /etc/config/hydra
           environment:
                 DSN=postgres://hydra:secret@postgresd:5432/hydra?sslmode=disable&max_conns=20&max
           restart: unless-stopped
           depends on:
                 - hydra-migrate
           networks:
                - intranet
     hydra-migrate:
           image: oryd/hydra:v2.0.2
           environment:
                - DSN=postgres://hydra:secret@postgresd:5432/hydra?sslmode=disable&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&max_conns=20&
           command: migrate -c /etc/config/hydra/hydra.yml sql -e --yes
           volumes:
                - type: bind
                      source: ./config
                      target: /etc/config/hydra
           restart: on-failure
           networks:
                intranet
     consent:
           environment:
                 - HYDRA ADMIN URL=http://hydra:4445
           image: oryd/hydra-login-consent-node:v2.0.2
           ports:
                - "3001:3000"
           restart: unless-stopped
           networks:
                - intranet
     postgresd:
           image: postgres:11.8
           ports:
                - "5432:5432"
           environment:

    POSTGRES USER=hydra

                POSTGRES_PASSWORD=secret

    POSTGRES DB=hydra

           networks:
```

```
- intranet
networks:
  intranet:
```

还需要为Hydra服务创建一个配置文件hydra.yml,该文件放在docker-compose.yml文件同级的config目录下。

```
serve:
  cookies:
    same_site_mode: Lax
urls:
  self:
    issuer: http://172.18.3.200:4444
  consent: http://172.18.3.200:3001/consent
  login: http://172.18.3.200:3001/login
  logout: http://172.18.3.200:3001/logout
secrets:
  system:
    youReallyNeedToChangeThis
oidc:
  subject_identifiers:
    supported_types:
      pairwise
      - public
    pairwise:
      salt: youReallyNeedToChangeThis
```

配置文件编辑好以后就可以使用docker-compose命令把服务启动起来。

```
# docker-compose up -d
```

# 3. 令牌(token)颁发过程

### 3.1. 创建客户端

请求地址

```
POST http://172.18.3.200:4445/admin/clients
```

#### 请求体

```
{
   "client_name": "crm", //这里为应用程序起一个名字,该名称会在令牌中体现
   "token_endpoint_auth_method": "client_secret_post", //注意有的实例中使用client_secret_l
   "redirect_uris": [
       "http://127.0.0.1:5555/callback" //目标跳转地址。
   ],
   "scope": "openid offline",
   "grant_types": [
       "authorization_code",
       "refresh_token",
       "implicit",
       "client_credentials"
   ],
   "response_types": [
       "code",
       "id_token",
       "token"
   ]
}
```

返回内容

```
{
    "client_id": "19d6a916-6ac6-4f26-99ca-98534b075182", // 客户端id
    "client_name": "crm",
    "client_secret": "Y00FMUQ6kZ-aLg.X0p.1oWEH4T", //客户端密码
    "redirect uris": [
        "http://127.0.0.1:5555/callback"
    ],
    "grant_types": [
        "authorization_code",
        "refresh_token",
        "implicit",
        "client credentials"
    ],
    "response_types": [
        "code",
        "id_token",
        "token"
    ],
    "scope": "openid offline",
    "audience": [],
    "owner": "",
    "policy uri": "",
    "allowed_cors_origins": [],
    "tos_uri": "",
    "client uri": "",
    "logo_uri": "",
    "contacts": null,
    "client_secret_expires_at": 0,
    "subject_type": "public",
    "iwks": {}.
    "token_endpoint_auth_method": "client_secret_post",
    "userinfo signed response alg": "none",
    "created_at": "2023-08-09T05:56:34Z",
    "updated at": "2023-08-09T05:56:33.588728Z",
    "metadata": {},
    "registration_access_token": "ory_at_Jkk30mBb8SKzGroR6KYP7r92GZHrkSzdSHlM2fbEdk8.jr.
    "registration_client_uri": "http://172.18.3.200:4444/oauth2/register/19d6a916-6ac6-4
    "authorization_code_grant_access_token_lifespan": null,
    "authorization code grant id token lifespan": null,
    "authorization code grant refresh token lifespan": null,
    "client credentials grant access token lifespan": null,
    "implicit grant access token lifespan": null,
    "implicit_grant_id_token_lifespan": null,
    "jwt bearer grant access token lifespan": null,
    "refresh_token_grant_id_token_lifespan": null,
    "refresh_token_grant_access_token_lifespan": null,
    "refresh token grant refresh token lifespan": null
}
```

### 3.2. 请求授权

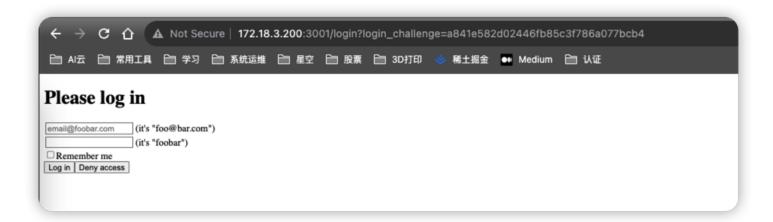
#### 请求地址

GET http://172.18.3.200:4444/oauth2/auth?response\_type=code&client\_id=19d6a916-6ac6-4f20

#### 请求参数解析

请求参数	请求参数类型	请求参数说明
response_type	String	请求回执类型。code表示授权码
client_id	String	请求的客户id
scope	String	请求的作用域方位
state	String	大于16位的随机值,可随便写非中文字符

建议该步骤在浏览器中访问,访问后会跳转到登录code连接



跳转地址中最重要的就是login\_challenge的值,该值需要传入到下一步骤用于登录验证。

### 3.3. 登录请求

上一步获取到登录的验证码login\_challenge后,需要访问如下地址进行登录验证.

#### 访问地址

PUT http://172.18.3.200:4445/admin/oauth2/auth/requests/login/accept?login\_challenge=4a

请求参数解析

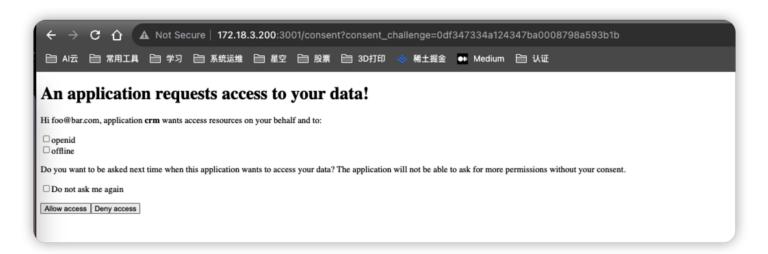
请求参数	请求参数类型	请求参数说明
login_challenge	String	登录验证码,从上一步获得

#### 返回内容

登录成功后会返回一个跳转地址

```
{
    "redirect_to": "http://172.18.3.200:4444/oauth2/auth?client_id=19d6a916-6ac6-4f26-9996}
```

将redirect to中的地址全文拷贝到浏览器访问就可以得到跳转后的信息



这里最终要的值为consent\_challenge,该值就是授权码。

### 3.4. 认证请求

上一步骤获取到授权码以后就可以发送认证请求。

#### 请求地址

http://172.18.3.200:4445/admin/oauth2/auth/requests/consent/accept?consent\_challenge=0d

该地址最后的consent\_challenge就是上一步骤获取到的授权码值。

#### 返回内容

```
{
    "redirect_to": "http://172.18.3.200:4444/oauth2/auth?client_id=19d6a916-6ac6-4f26-9996}
```

以上地址放入浏览器跳转后可获得令牌访问吗

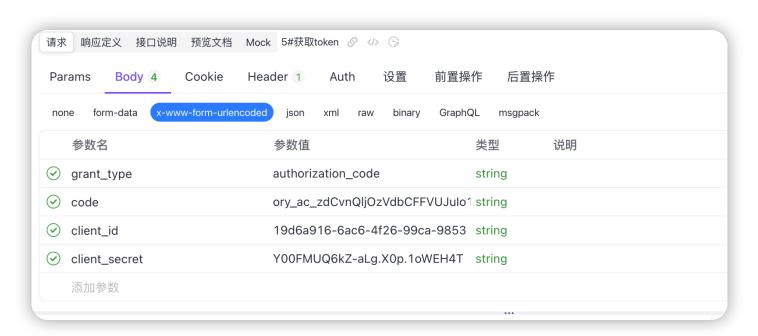
截图中的code后面的值就是令牌访问码,需要用此码调用获取令牌接口获取到最终的令牌信息.

### 3.5. 获取令牌

访问地址

http://172.18.3.200:4444/oauth2/token

该请求需要数据格式为x-www-form-urlencoded, 具体的请求参数为



请求参数	请求参数类型	请求参数说明
grant_type	String	请求授权类型。authorization_code表示授权码
code	String	授权码
client_id	String	客户端id,在第一步返回结果中获得
client_secret	String	客户端密码,在第一步返回结果中获得

#### 返回内容

```
Pretty Raw Preview Visualize JSON v utf8 v 5

⊕ □ □ Q

                                                                                                                                                                                                                                                                                                                                                                                                   200 85 ms
       1
       2
                                   "access_token": "ory_at_9Eex_tbDmvg_mfllAtI0FgKApDM6s-Udkg0tDA57eFY.
                                                                                                                                                                                                                                                                                                                                                                                                   ☑ 返回数据:
                                   _B-GwaGcDj9KLWLweh0WASjkku6oeRmR_Bl0_I4fK2U",
                                   "expires_in": 3600,
       3
                                   "id_token":
        4
                                   "eyJhbGci0iJSUzI1NiIsImtpZCI6ImMwY2ZiNWIzLTY2NmItNDVkZi04ZDNhLTE5MDIyNDA0MmEyMCIsInR5cCI6Ikp
                                  XVCJ9.
                                   {\tt eyJhY3Ii0iIxIiwiYXRfaGFzaCI6Im52WEtkdmxqYzdjR0haNTk1N19TbUEiLCJhdWQi0lsiMTlkNmE5MTYtNmFjNi00}
                                   \label{eq:control} ZjI2LTk5Y2Et0Tg1MzRiMDc1MTgyIl0sImF1dGhfdGltZSI6MTY5MTU2NDk1MSwiZXhwIjoxNjkxNTY5MTYxLCJpYXQiInformation and the control of the control 
                                  {\tt OjE20TE1NjU1NjEsImlzcy16Imh0dHA6Ly8xNzIuMTguMy4yMDA6NDQ0NCIsImp0aS16Ijg5YWU0YTIwLTVkMDMtNDFm} \\
                                  ODYxNDU5NmMyYThkIiwic3ViIjoiZm9vQGJhci5jb20iLCJ1c2VySWQi0iIxMTEifQ.\\
                                   fJ4ALhvoFf0iFVasQby-eU5Ld-Efoj_SC_NEnf5ziBg0N9SY0P2nwd28Mo-85PHzjwuxvvnXSEA26nL4-Tn0AYBEeTHN
                                  Rcea2\_r9IIkVBgLkCamKBcg7XaAGQxholmDkPhJzBrrZHjq5gXw1A42M2VCIK50Cj1mx3BhE7Hbsh0lE78MTzl9pFnII
                                   b\texttt{M\_J8} i \texttt{MBCdj} \texttt{II80QwriRdIvUdBYPC6} - \texttt{UjRk9199c6tizl27Zigz01oU1INIuTwnCEtC-q-j4vz6YbU3YUldFq5bVJg9} \_ i \texttt{MSCHMINGETC-q-j4vz6YbU3YUldFq5bVJg9} \_ i \texttt{MSCHMINGETC-q-j4vz6VbJg9} \_ i \texttt{MSCHM
                                  NH75rHGG_iVICeoYG98Dw00ZtNlcQu6TodlUHJlj82YwHPUPVIk0D1aFRUzJYDimYv_908wUiKskLeUrM5lSAs1JMV-5
                                   mItsIjRGTJT4gjsT38gT7pph_6y7Mz_0GkSsxEihsD4F9iWkq7GvGAPPzf5THsH4EVIGwMLaUrVp8D8skEh0alFiHglG
                                   603 d0 U79 PU dvBA\_ksvk0 a UuYHjruNfs B6RnirrKXVCCK lF6Eq6gyaGhwQgfNPfyTzrCQ2Uoh8bGnIIMaCbLuDIMzibp2
                                   8v6wR4IeAhKnjR9F84UMypTHtbQfU4UiW8YyUUc",
                                   "refresh token": "ory rt jCUV-MSDnZp9YoXN0Km2DD0NFFJ7GElEE8CATCP1Zoc.
       5
                                   YMU1IabnGw6fbPBn1xAK7BeAfgbxfTJJNQukuhUIKcg",
                                   "scope": "openid offline",
       6
                                   "token_type": "bearer"
       8
```

#### 返回内容解析

请求参数	请求参数类型	请求参数说明
access_token	String	访问授权码,与请求中的code值一致
expires_in	Number	令牌过期时间
id_token	String	最终返回的令牌信息
refresh_token	String	令牌刷新授权码
scope	String	获取到的令牌生效范围

请求参数	请求参数类型	请求参数说明
token_type	String	令牌类型.

# 4. 参考资料

#### 【官网】

https://www.ory.sh/docs/identities/native-browser

【阮一峰 OAuth 2.0 的四种方式】

https://www.ruanyifeng.com/blog/2019/04/oauth-grant-types.html

#### 【其他网络资源】

https://blog.csdn.net/u010381752/article/details/119328575

http://www.junyao.tech/posts/f8dc0074.html