GProduct 操作说明

DaoCloud 产品团队

2024 年 5 月 23 日

目录

GProduct 如何对接全局管理

GProduct 是 DCE 5.0 中除全局管理外的所有其他模块的统称，这些模块需要与全局管理对接后才能加入到 DCE 5.0 中。

对接什么

- 对接导航栏

  入口统一放在左侧导航栏。

- 接入路由和 AuthN

  统一 IP 或域名，将路由入口统一走全局管理的 Istio Gateway。

- 统一登录 / 统一 AuthN 认证

  登录统一使用全局管理 (Keycloak) 登录页，API authn token 验证使用 Istio Gateway。 GProduct 对接全局管理后不需要关注如何实现登录和认证。

DCE 5.0 对接客户系统视频演示

对接导航栏

以容器管理（开发代号 kpanda ）为例，对接到导航栏。

对接后的预期效果如图：

对接方法

参照以下步骤对接 GProduct：

1. 通过 GProductNavigator CR 将容器管理的各功能项注册到导航栏菜单。

```yaml
apiVersion: ghippo.io/v1alpha1
kind: GProductNavigator
metadata:
  name: kpanda
spec:
  gproduct: kpanda
  name: 容器管理
  localizedName:
    zh-CN: 容器管理
    en-US: Container Management
  url: /kpanda
  category: 容器  # (1)
  iconUrl: /kpanda/nav-icon.png
  order: 10 # (2)
  menus:
  - name: 备份管理
    localizedName:
      zh-CN: 备份管理
      en-US: Backup Management
    iconUrl: /kpanda/bkup-icon.png
    url: /kpanda/backup
```

   1. 当前只支持概览、工作台、容器、微服务、数据服务、管理，六选一
   2. 数字越大排在越上面

   全局管理的导航栏 **category** 配置在 ConfigMap，暂时不能以注册方式增加，需要联系全局管理团队来添加。

2. kpanda 前端作为微前端接入到 DCE 5.0 父应用 Anakin 中

   前端使用 qiankun 来接入子应用 UI， 可以参考快速上手。

   在注册 GProductNavigator CR 后，接口会生成对应的注册信息，供前端父应用注册使用。 例如 kpanda 就会生成以下注册信息：

```jsonc
{
  "id": "kpanda",
  "title": "容器管理",
  "url": "/kpanda",
  "uiAssetsUrl": "/ui/kpanda/", // 结尾的/是必须的
  "needImportLicense": false
},
```

以上注册信息与 qiankun 子应用信息字段的对应关系是：

```
{
    name: id,
    entry: uiAssetsUrl,
    container: '#container',
    activeRule: url,
    loader,
    props: globalProps,
}
```

container 和 loader 由前端父应用提供，子应用无需关心。 props 会提供一个包含用户基本信息、子产品注册信息等的 pinia store。

qiankun 启动时会使用如下参数：

```
start({
  sandbox: {
    experimentalStyleIsolation: true,
  },
  // 去除子应用中的 favicon 防止在 Firefox 中覆盖父应用的 favicon
  getTemplate: (template) => template.replaceAll(/<link\s* rel="[
\w\s]*icon[\w\s]*"\s*( href=".*?")?\s*\/?>/g, ''),
});
```

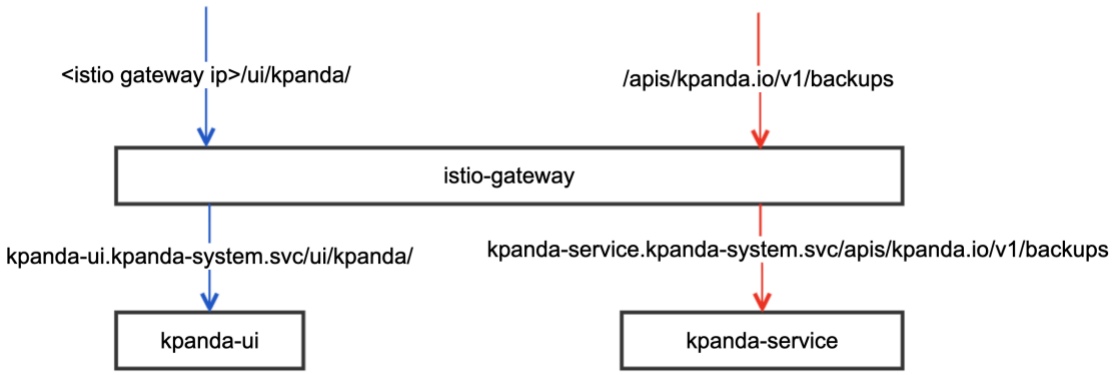
请参阅前端团队出具的 GProduct 对接 demo tar 包。

接入路由和登录认证

接入后统一登录和密码验证，效果如下图：

各个 GProduct 模块的 API bear token 验证都走 Istio Gateway。

接入后的路由映射图如下：



接入方法

以 kpanda 为例注册 GProductProxy CR。

```
# GProductProxy CR 示例，包含路由和登录认证

# spec.proxies: 后写的路由不能是先写的路由子集，反之可以
# spec.proxies.match.uri.prefix: 如果是后端 api，建议在 prefix 末尾添加 "/" 表述这段 path 结束（特殊需求可以不用加）
# spec.proxies.match.uri: 支持 prefix 和 exact 模式；Prefix 和 Exact 只能 2 选 1；Prefix 优先级大于 Exact

apiVersion: ghippo.io/v1alpha1
kind: GProductProxy
metadata:
  name: kpanda  # (1)
spec:
  gproduct: kpanda  # (2)
  proxies:
  - labels:
      kind: UIEntry
    match:
      uri:
        prefix: /kpanda # (3)
    rewrite:
      uri: /index.html
    destination:
      host: ghippo-anakin.ghippo-system.svc.cluster.local
      port: 80
    authnCheck: false  # (4)
  - labels:
```

```
    kind: UIAssets
  match:
    uri:
      prefix: /ui/kpanda/ # (5)
  destination:
    host: kpanda-ui.kpanda-system.svc.cluster.local
    port: 80
  authnCheck: false
- match:
    uri:
      prefix: /apis/kpanda.io/v1/a
  destination:
    host: kpanda-service.kpanda-system.svc.cluster.local
    port: 80
  authnCheck: false
- match:
    uri:
      prefix: /apis/kpanda.io/v1 # (6)
  destination:
    host: kpanda-service.kpanda-system.svc.cluster.local
    port: 80
  authnCheck: true
```

1. cluster 级别 CRD
2. 需要用小写指定 GProduct 名字
3. 还可支持 exact
4. 是否需要 istio-gateway 给该条路由 API 作 AuthN Token 认证, false 为跳过认证
5. UIAssets 建议末尾添加 / 表示结束（不然前端可能会出现问题）
6. 后写的路由不能是先写的路由的子集, 反之可以

如何将客户系统集成到 DCE 5.0（OEM IN）

OEM IN 是指合作伙伴的平台作为子模块嵌入 DCE 5.0，出现在 DCE 5.0 一级导航栏。用户通过 DCE 5.0 进行登录和统一管理。实现 OEM IN 共分为 5 步，分别是：

1. 统一域名
2. 打通用户体系
3. 对接导航栏
4. 定制外观
5. 打通权限体系（可选）

具体操作演示请参见 OEM IN 最佳实践视频教程。

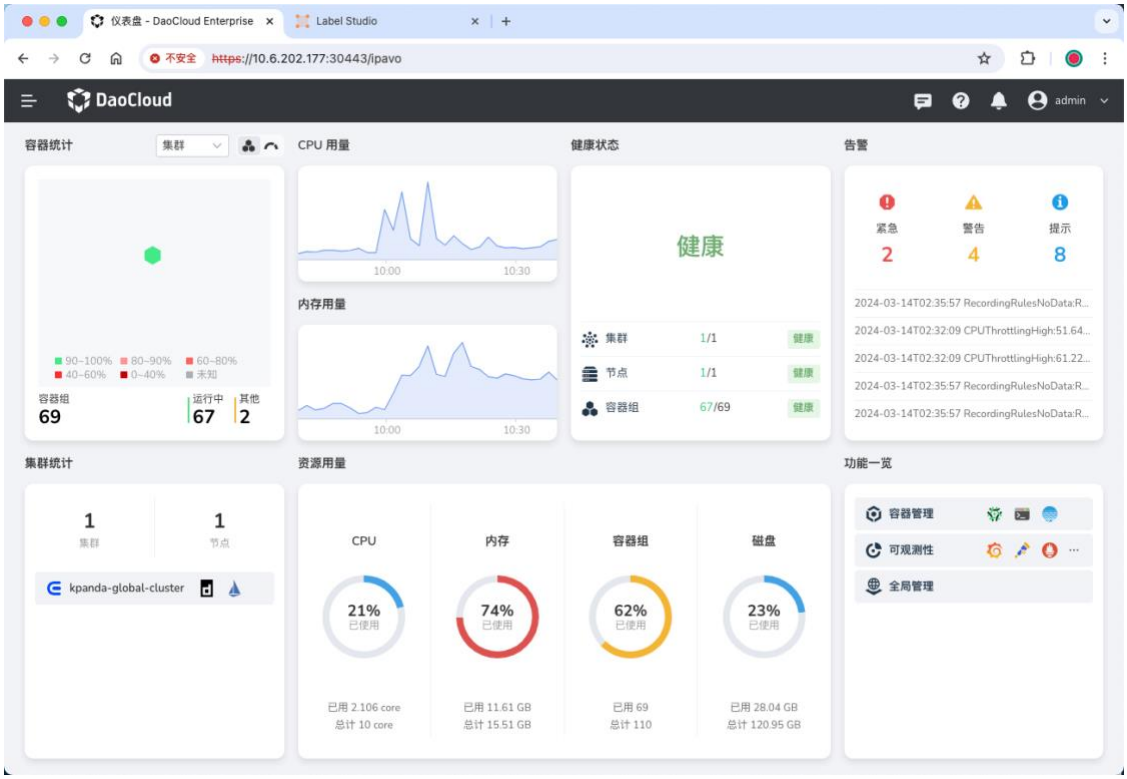
Note：以下使用开源软件 `Label Studio` 来做嵌套演示。实际场景需要自己解决客户系统的如下问题：

客户系统需要自己添加一个 `Subpath`，用于区分哪些是 `DCE 5.0` 的服务，哪些是客户系统的服务。

环境准备

1. 部署 DCE 5.0 环境：

   `https://10.6.202.177:30443` 作为 DCE 5.0

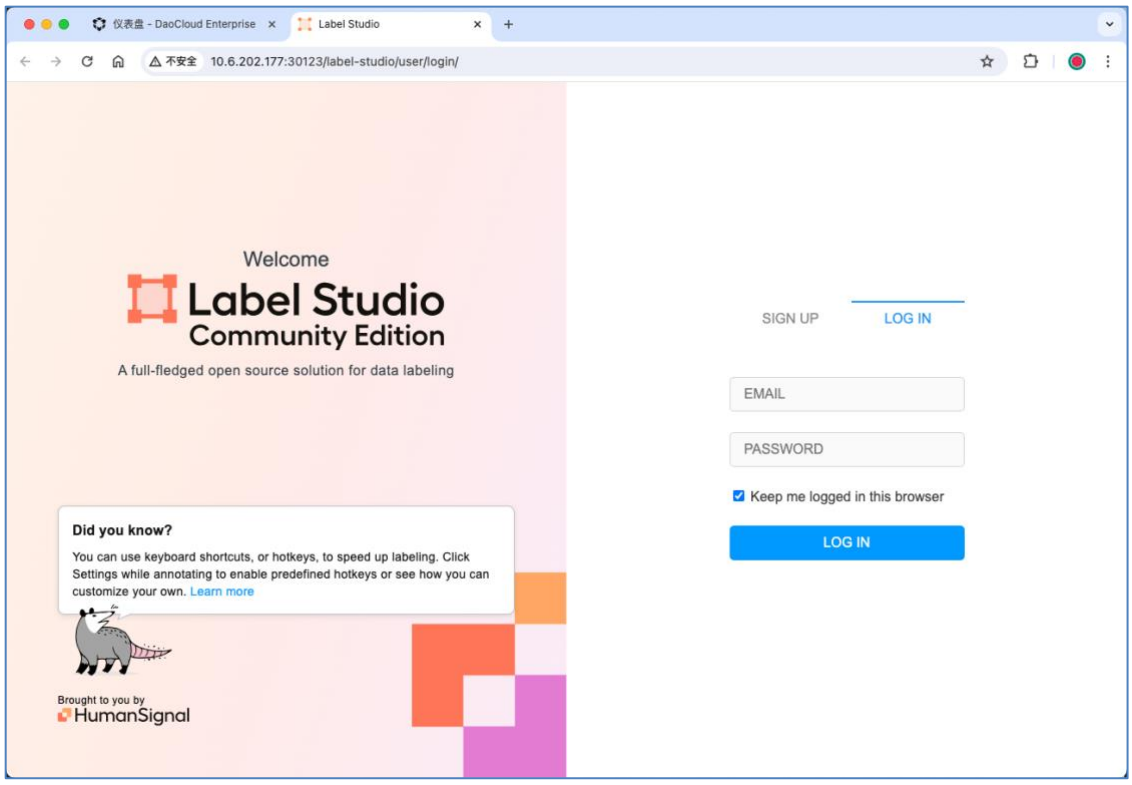


2. 部署客户系统环境：

   `http://10.6.202.177:30123` 作为客户系统

   应用过程中对客户系统的操作请根据实际情况进行调整。

3. 规划客户系统的 Subpath 路径：`http://10.6.202.177:30123/label-studio`（建议使用辨识度高的名称作为 Subpath，不能与主 DCE 5.0 的 HTTP router 发生冲突）。请确保用户通过 `http://10.6.202.177:30123/label-studio` 能够正常访问客户系统。



统一域名和端口

1. SSH 登录到 DCE 5.0 服务器。

   ```
   ssh root@10.6.202.177
   ```

2. 使用 vim 命令创建 **label-studio.yaml** 文件

   ```
   vim label-studio.yaml
   ```

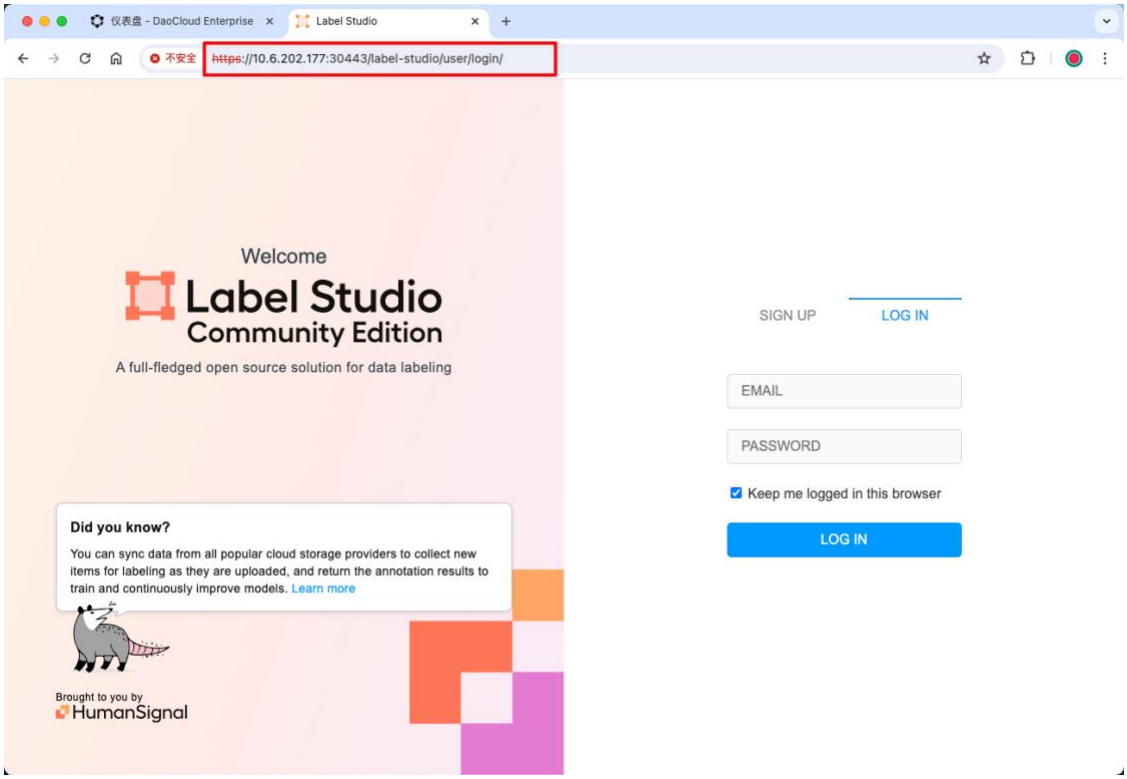
   ```yaml title="label-studio.yaml"
   apiVersion: networking.istio.io/v1beta1
   kind: ServiceEntry
   metadata:
     name: label-studio
     namespace: ghippo-system spec: exportTo:
     - "*" hosts:
     - label-studio.svc.external ports: # 添加虚拟端口
     - number: 80 name: http protocol: HTTP location: MESH_EXTERNAL resolution: STATIC endpoints: # 改为客户系统的域名（或 IP）
   ```

- address: 10.6.202.177 ports: # 改为客户系统的端口号 http: 30123 — apiVersion: networking.istio.io/v1alpha3 kind: VirtualService metadata: # 修改为客户系统的名字 name: label-studio namespace: ghippo-system spec: exportTo:
- “*” hosts:
- “*” gateways:
- ghippo-gateway http:
- match:
  - uri: exact: /label-studio # 修改为客户系统在 DCE5.0 Web UI 入口中的路由地址
  - uri: prefix: /label-studio/ # 修改为客户系统在 DCE5.0 Web UI 入口中的路由地址 route:
  - destination: # 修改为上文 ServiceEntry 中的 spec.hosts 的值 host: label-studio.svc.external port: # 修改为上文 ServiceEntry 中的 spec.ports 的值 number: 80 — apiVersion: security.istio.io/v1beta1 kind: AuthorizationPolicy metadata: # 修改为客户系统的名字 name: label-studio namespace: istio-system spec: action: ALLOW selector: matchLabels: app: istio-ingressgateway rules:
- from:
  - source: requestPrincipals:
    - ’*’
- to:
  - operation: paths:
    - /label-studio # 修改为 VirtualService 中的 spec.http.match.uri.prefix 的值
    - /label-studio/* # 修改为 VirtualService 中的 spec.http.match.uri.prefix 的值（注意，末尾需要添加 “*“） ```

3. 使用 kubectl 命令应用 label-studio.yaml：

```
kubectl apply -f label-studio.yaml
```

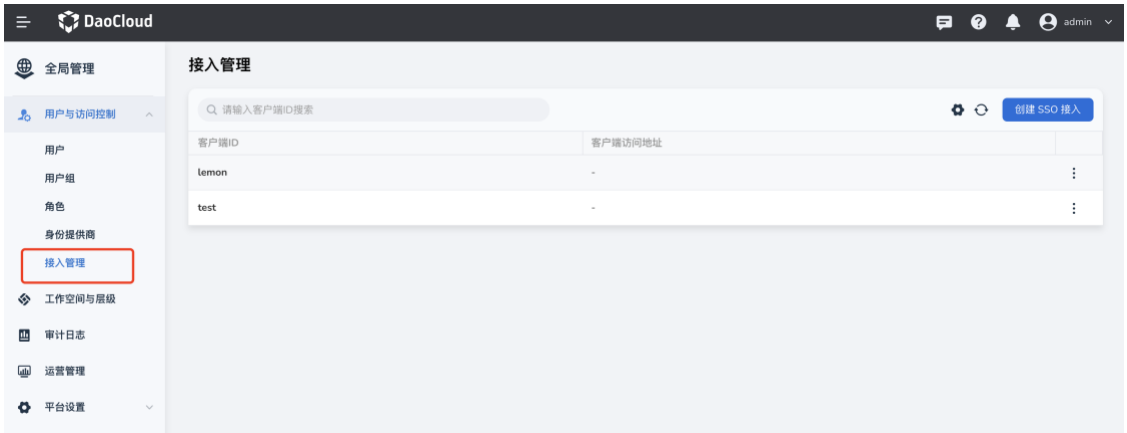
4. 验证 Label Studio UI 的 IP 和 端口是否一致：



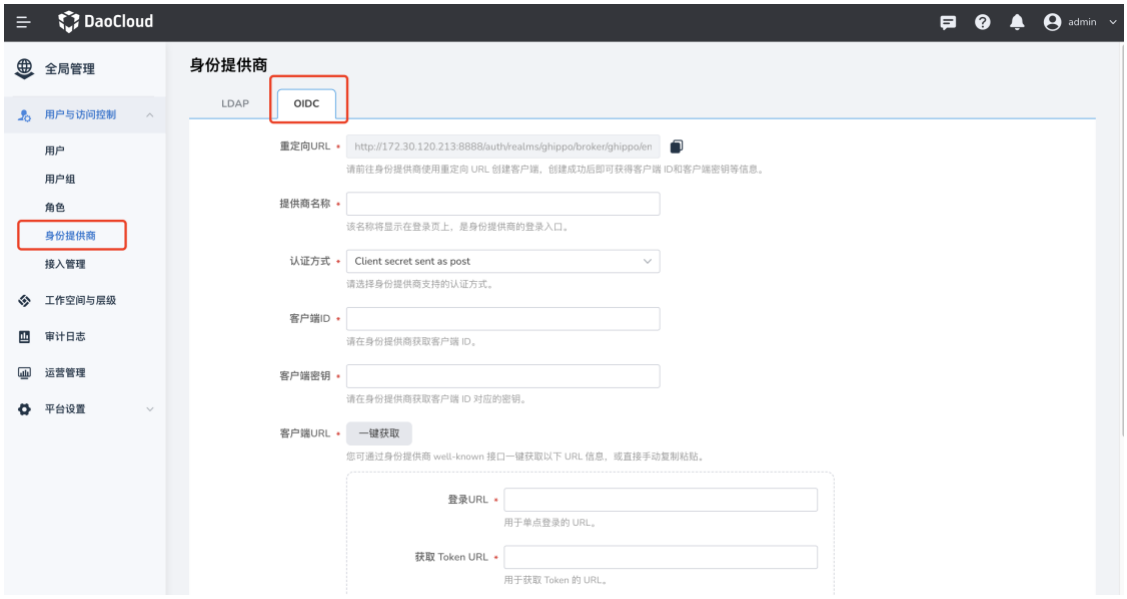
打通用户体系

将客户系统与 DCE 5.0 平台通过 OIDC/OAUTH 等协议对接，使用户登录 DCE 5.0 平台后进入客户系统时无需再次登录。

1. 在两套 DCE 5.0 的场景下，可以在 DCE 5.0 中通过 **全局管理 -> 用户与访问控制 -> 接入管理** 创建 SSO 接入。



2. 创建后将详情中的客户端 ID、密钥、单点登录 URL 等填写到客户系统的 **全局管理 -> 用户与访问控制 -> 身份提供商 -> OIDC** 中，完成用户对接。



3. 对接完成后，客户系统登录页面将出现 OIDC（自定义）选项，首次从 DCE 5.0 平台进入客户系统时选择通过 OIDC 登录， 后续将直接进入客户系统无需再次选择。

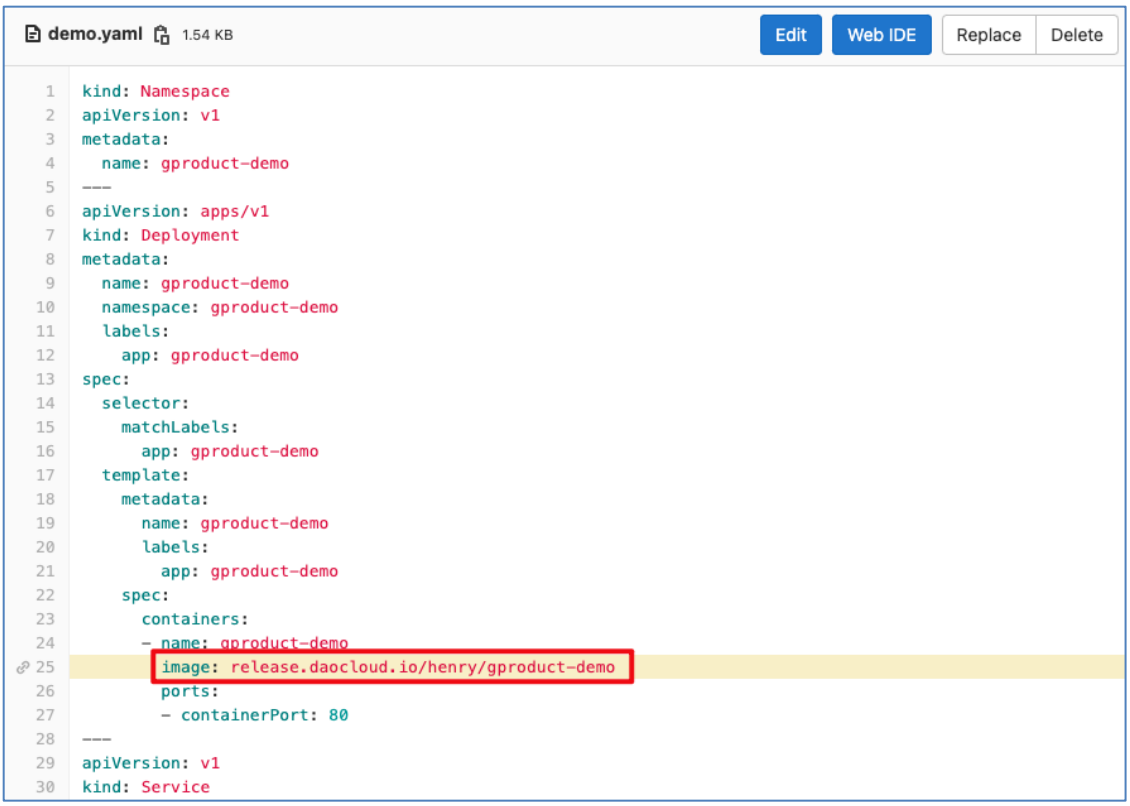
对接导航栏

参考文档下方的 tar 包来实现一个空壳的前端子应用，把客户系统以 iframe 的形式放进该空壳应用里。

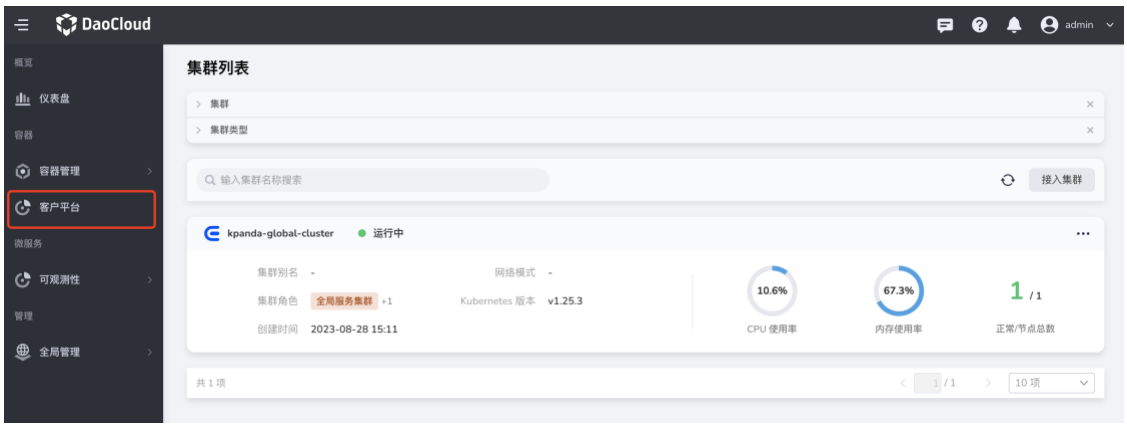
1. 下载 gproduct-demo-main.tar.gz 文件，将 src 文件夹下 App-iframe.vue 中的 src 属性值改为用户进入客户系统的绝对地址，如：**src="https://10.6.202.177:30443/label-studio" (DCE 5.0 地址 + Subpath)** 或相对地址，如： **src="./external-anyproduct/insight"**

```vue
<template>
  <iframe
    src="https://daocloud.io"
    title="demo"
    class="iframe-container"
  />
</template>

<style lang="scss">
html,
body {
  height: 100%;
}

#app {
  display: flex;
  height: 100%;
  .iframe-container {
    border: 0;
    flex: 1 1 0;
  }
}
</style>
```

2. 删除 src 文件夹下的 App.vue 和 main.ts 文件，同时将：
   - App-iframe.vue 重命名为 App.vue
   - main-iframe.ts 重命名为 main.ts
3. 按照 readme 步骤构建镜像（注意：执行最后一步前需要将 **demo.yaml** 中的镜像地址替换成构建出的镜像地址）



对接完成后，将在 DCE 5.0 的一级导航栏出现 **客户系统**，点击可进入客户系统。



定制外观

Note：DCE 5.0 支持通过写 CSS 的方式来实现外观定制。实际应用中客户系统如何实现外观定制需要根据实际情况处理。

登录客户系统，通过 **全局管理 -> 平台设置 -> 外观定制** 可以自定义平台背景颜色、logo、名称等，具体操作请参照外观定制。

打通权限体系（可选）

方案思路一：

定制化团队可实现一定制模块，DCE 5 将每一次的用户登录事件通过 Webhook 的方式通知到定制模块， 定制模块可自行调用 AnyProduct 和 DCE 5.0 的 OpenAPI 作该用户的权限信息同步。

方案思路二：

通过 Webhook 方式，将每一次的授权变化都通知到 AnyProduct (如有需求，后续可实现)。

AnyProduct 使用 DCE 5.0 的其他能力(可选)

方法为：调用 DCE 5.0 OpenAPI

参考资料

- 参考 OEM OUT 文档
- 参阅 gProduct-demo-main 对接 tar 包

如何将 DCE 5.0 集成到客户系统（OEM OUT）

OEM OUT 是指将 DCE 5.0 作为子模块接入其他产品，出现在其他产品的菜单中。用户登录其他产品后可直接跳转至 DCE 5.0 无需二次登录。实现 OEM OUT 共分为 5 步，分别是：

- 统一域名
- 打通用户体系
- 对接导航栏
- 定制外观
- 打通权限体系(可选)

具体操作演示请参见 OEM OUT 最佳实践视频教程。

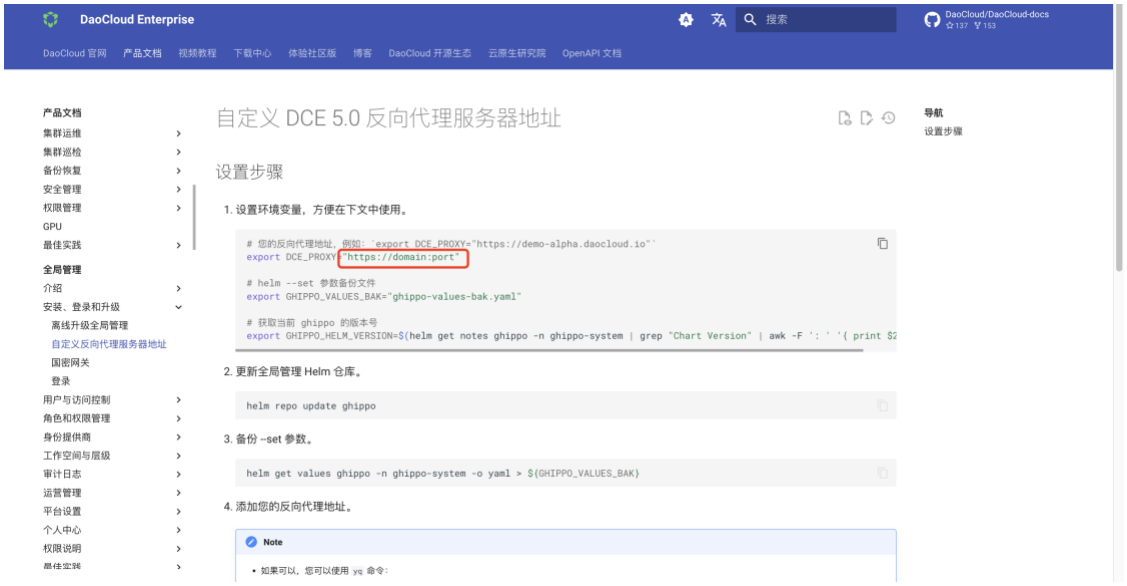
统一域名

1. 部署 DCE 5.0（假设部署完的访问地址为 `https://10.6.8.2:30343/`）
2. 客户系统和 DCE 5.0 前可以放一个 Nginx 反代来实现同域访问， **/** 路由到客户系统， **/dce5 (subpath)** 路由到 DCE 5.0 系统， **vi /etc/nginx/conf.d/default.conf** 示例如下：

```
server {
    listen       80;
```

```
    server_name  localhost;

    location /dce5/ {
      proxy_pass https://10.6.8.2:30343/;
      proxy_http_version 1.1;
      proxy_read_timeout 300s; # 如需要使用 kpanda cloudtty 功能需要这行，否则可以去掉
      proxy_send_timeout 300s; # 如需要使用 kpanda cloudtty 功能需要这行，否则可以去掉

      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

      proxy_set_header Upgrade $http_upgrade; # 如需要使用 kpanda cloudtty 功能需要这行，否则可以去掉
      proxy_set_header Connection $connection_upgrade; # 如需要使用 kpanda cloudtty 功能需要这行，否则可以去掉
    }

    location / {
        proxy_pass https://10.6.165.50:30443/; # 假设这是客户系统地址(如意云)
        proxy_http_version 1.1;

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

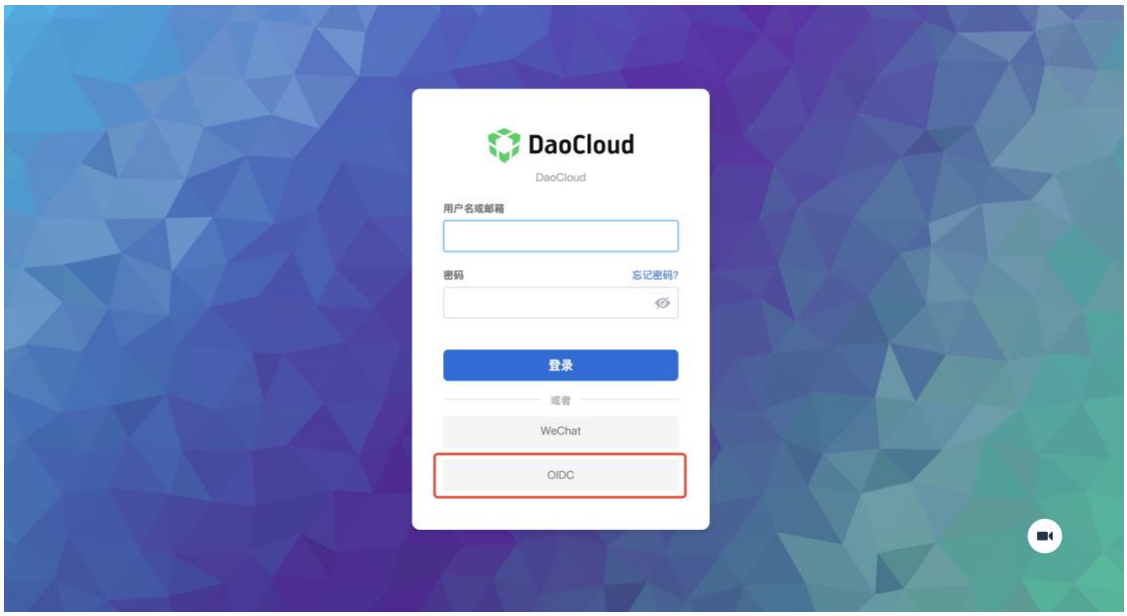
3. 假设 Nginx 入口地址为 10.6.165.50，按自定义 DCE 5.0 反向代理服务器地址把 DCE_PROXY 反代设为 `http://10.6.165.50/dce5`。确保能够通过 `http://10.6.165.50/dce5` 访问 DCE 5.0。 客户系统也需要进行反代设置，需要根据不同平台的情况进行处理。



打通用户体系

将客户系统与 DCE 5.0 平台通过 OIDC/OAUTH 等协议对接，使用户登录客户系统后进入 DCE 5.0 时无需再次登录。 在拿到客户系统的 OIDC 信息后填入 **全局管理** -> **用户与访问控制** -> **身份提供商** 中。

对接完成后，DCE 5.0 登录页面将出现 OIDC（自定义）选项，首次从客户系统进入 DCE 5.0 时选择通过 OIDC 登录， 后续将直接进入 DCE 5.0 无需再次选择。



对接导航栏

对接导航栏是指 DCE 5.0 出现在客户系统的菜单中，用户点击相应的菜单名称能够直接进入 DCE 5.0。 因此对接导航栏依赖于客户系统，不同平台需要按照具体情况进行处理。

定制外观

通过 **全局管理 -> 平台设置 -> 外观定制** 可以自定义平台背景颜色、logo、名称等，具体操作请参照外观定制。

打通权限体系（可选）

打通权限较为复杂，如有需求请联系全局管理团队。

定制 DCE 5.0 对接外部身份提供商 (IdP)

身份提供商（IdP, Identity Provider）： 当 DCE 5.0 需要使用客户系统作为用户源，使用客户系统登录界面来进行登录认证时，该客户系统被称为 DCE 5 的身份提供商

适用场景

如果客户对 Ghippo 登录 IdP 有高度定制需求，例如支持企业微信、微信等其他社会组织登录需求，请根据本文档实施。

支持版本

Ghippo 0.15.0 及以上版本。

具体方法

自定义 ghippo keycloak plugin

1. 定制 plugin

   参考 keycloak 官方文档和 keycloak 自定义 IdP 进行开发。

2. 构建镜像

```
# FROM scratch
FROM scratch

# plugin
COPY ./xxx-jar-with-dependencies.jar /plugins/
```

Note：如果需要两个定制化 IdP，需要复制两个 jar 包。

部署 Ghippo keycloak plugin 步骤

1. 把 Ghippo 升级到 0.15.0 或以上。 您也可以直接安装部署 Ghippo 0.15.0 版本，但需要把以下信息手动记录下来。

```
helm -n ghippo-system get values ghippo -o yaml

apiserver:
  image:
    repository: release.daocloud.io/ghippo-ci/ghippo-apiserver
    tag: v0.4.2-test-3-gaba5ec2
controllermanager:
  image:
    repository: release.daocloud.io/ghippo-ci/ghippo-apiserver
    tag: v0.4.2-test-3-gaba5ec2
global:
  database:
    builtIn: true
  reverseProxy: http://192.168.31.10:32628
```

2. 升级成功后，手工跑一个安装命令， --set 里设的参数值从上述保存的内容里得到，并且外加几个参数值：

   - global.idpPlugin.enabled：是否启用定制 plugin，默认已关闭
   - global.idpPlugin.image.repository：初始化自定义 plugin 的 initContainer 用的 image 地址

- global.idpPlugin.image.tag：初始化自定义 plugin 的 initContainer 用的 image tag
- global.idpPlugin.path：自定义 plugin 的目录文件在上述 image 里所在的位置

具体示例如下：

```
helm upgrade \
    ghippo \
    ghippo-release/ghippo \
    --version v0.4.2-test-3-gaba5ec2 \
    -n ghippo-system \
    --set apiserver.image.repository=release.daocloud.io/ghippo-c
i/ghippo-apiserver \
    --set apiserver.image.tag=v0.4.2-test-3-gaba5ec2 \
    --set controllermanager.image.repository=release.daocloud.io/
ghippo-ci/ghippo-apiserver \
    --set controllermanager.image.tag=v0.4.2-test-3-gaba5ec2 \
    --set global.reverseProxy=http://192.168.31.10:32628 \
    --set global.database.builtIn=true \
    --set global.idpPlugin.enabled=true \
    --set global.idpPlugin.image.repository=chenyang-idp \
    --set global.idpPlugin.image.tag=v0.0.1 \
    --set global.idpPlugin.path=/plugins/.
```

3. 在 keycloak 管理页面选择所要使用的插件。

Keycloak 自定义 IdP

要求：keycloak >= v20

已知问题 keycloak >= v21，删除了旧版 theme 的支持，可能会在 v22 修复。 参见 Issue #15344。

此次 demo 使用 Keycloak v20.0.5。

基于 source 开发

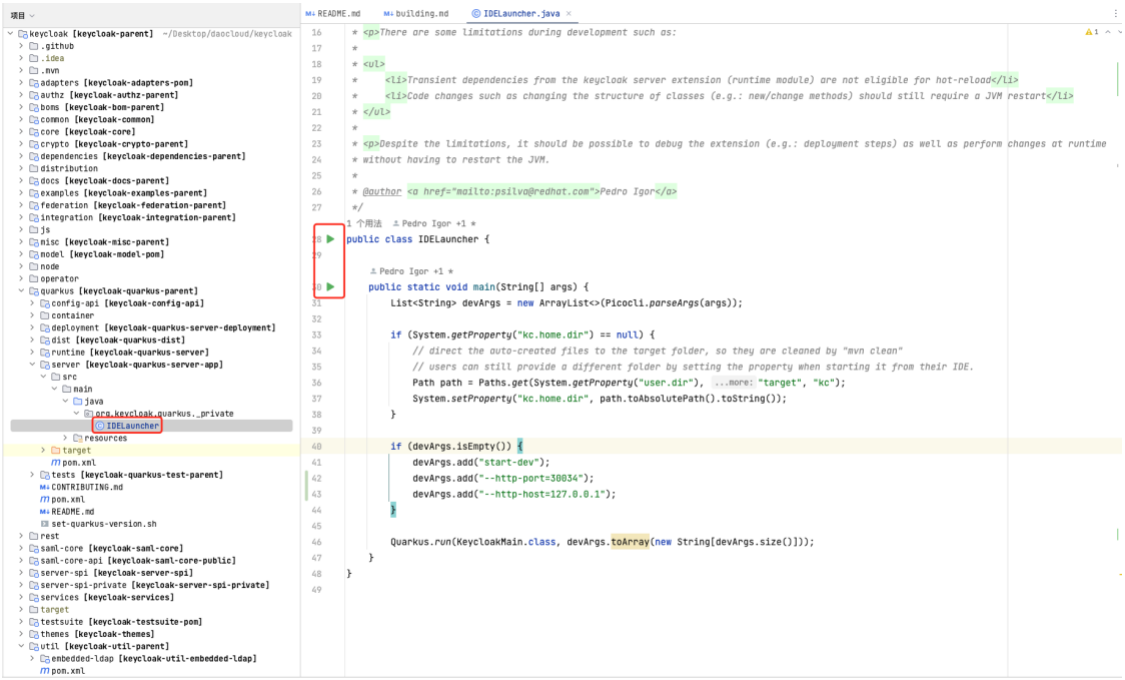
配置环境

参照 keycloak/building.md 配置环境。

参照 keycloak/README.md 运行以下命令：

```
cd quarkus
mvn -f ../pom.xml clean install -DskipTestsuite -DskipExamples -DskipTe
sts
```

从 IDE 运行

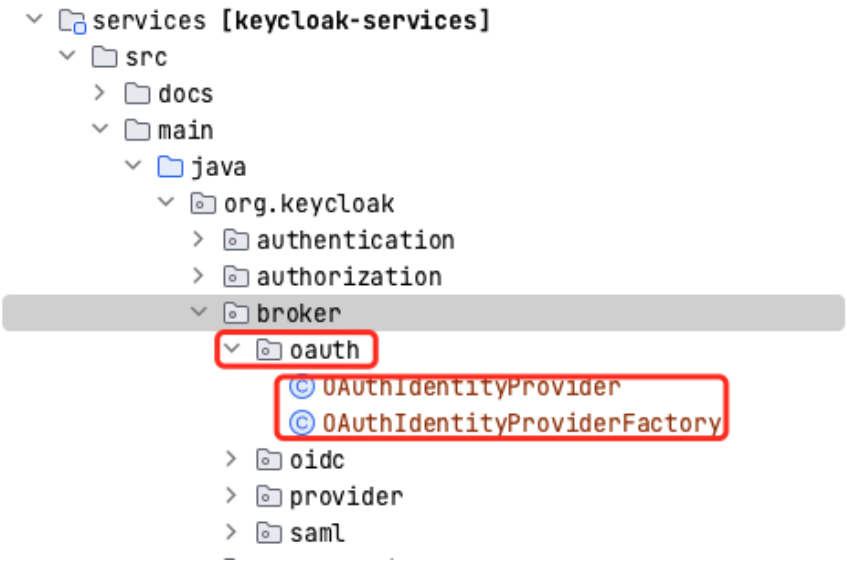


添加 service 代码

如果可从 keycloak 继承部分功能

在目录 **services/src/main/java/org/keycloak/broker** 下添加文件：

文件名需要是 **xxxProvider.java** 和 **xxxProviderFactory.java**



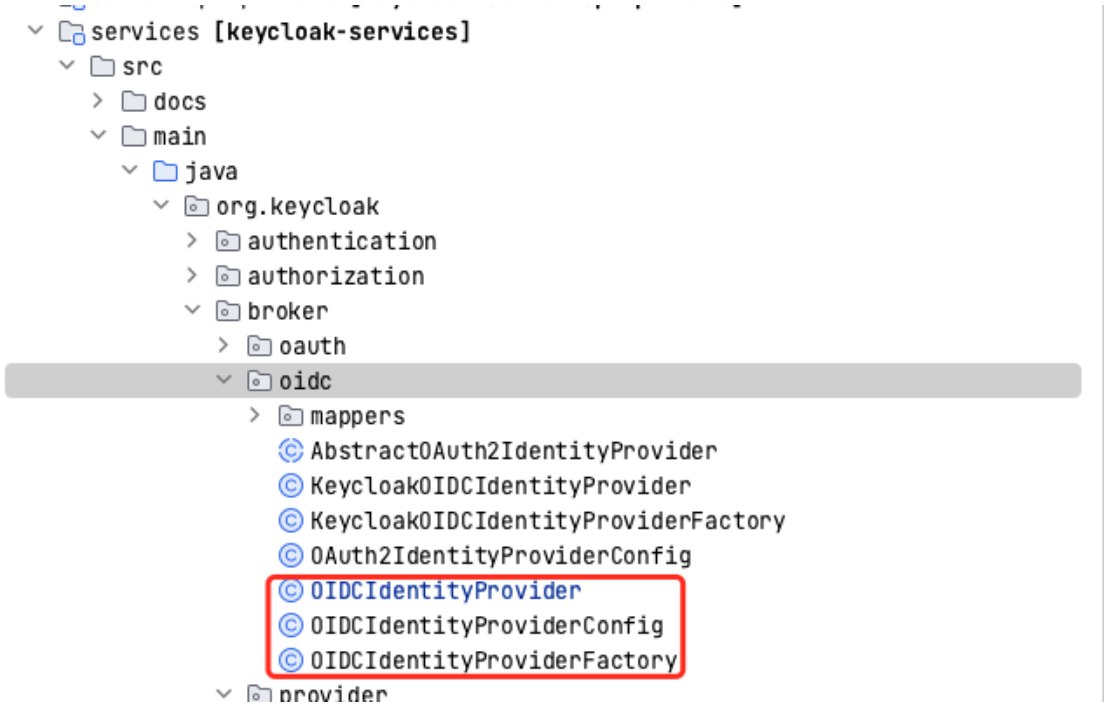
xxxProviderFactory.java 示例：

留意 **PROVIDER_ID = “oauth”;** 这个变量，后面定义 html 会用到。

xxxProvider.java 示例

如果不能从 keycloak 继承功能

参考下图中的三个文件编写你的代码：



添加 xxxProviderFactory 到 resource service

在 **services/src/main/resources/META-INF/services/org.keycloak.broker.provider.IdentityProviderFactory** 添加 xxxProviderFactory，这样刚刚编写的能工作了：

```
#
# Copyright 2016 Red Hat, Inc. and/or its affiliates
# and other contributors as indicated by the @author tags.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

org.keycloak.broker.oidc.OIDCIdentityProviderFactory
org.keycloak.broker.oidc.KeycloakOIDCIdentityProviderFactory
org.keycloak.broker.saml.SAMLIdentityProviderFactory
org.keycloak.broker.oauth.OAuthIdentityProviderFactory
```

添加 html 文件

复制 **themes/src/main/resources/theme/base/admin/resources/partials/realm-identity-provider-oidc.html** 文件到（改名为 **realm-identity-provider-oauth.html**，还记得上文中需要留意的变量吗）**themes/src/main/resources/theme/base/admin/resources/partials/realm-identity-provider-oauth.html**

到此所有的文件都添加完成了，开始调试功能。

打包成 jar 作为插件运行

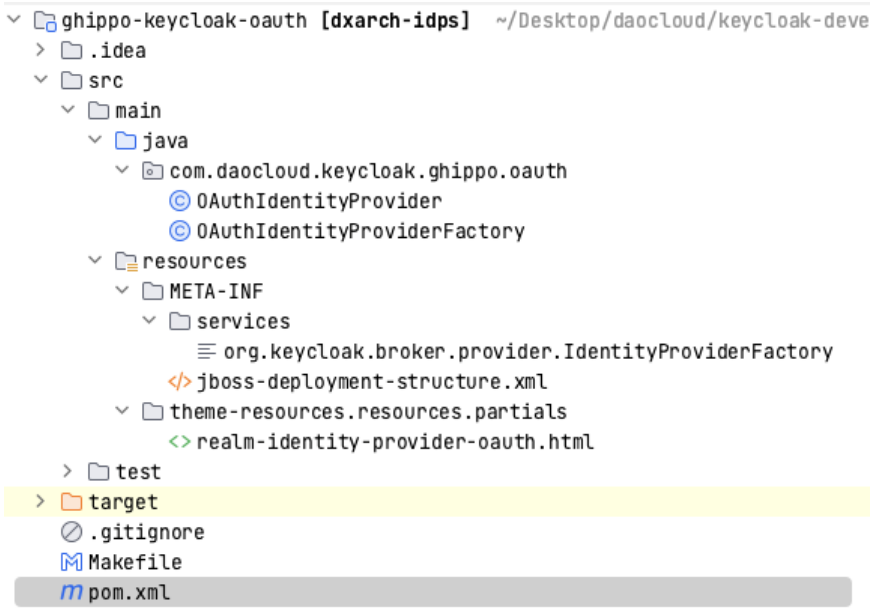
新建一个 java 项目，并将上面的代码复制到项目中，如下所示：

```
#
# Copyright 2016 Red Hat, Inc. and/or its affiliates
# and other contributors as indicated by the @author tags.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

org.keycloak.broker.oidc.OIDCIdentityProviderFactory
org.keycloak.broker.oidc.KeycloakOIDCIdentityProviderFactory
org.keycloak.broker.saml.SAMLIdentityProviderFactory
org.keycloak.broker.oauth.OAuthIdentityProviderFactory
```

参见 pom.xml。

运行 **mvn clean package**，打包完成得到 **xxx-jar-with-dependencies.jar** 文件。

下载 keycloak Release 20.0.5 zip 包并解压。



将 **xxx-jar-with-dependencies.jar** 复制到 **keycloak-20.0.5/providers** 目录中。

运行以下命令查看功能是否完整：

```
bin/kc.sh start-dev
```