ALIBABA CLOUD

阿里云

应用身份服务 IDaaS

单点登录配置

文档版本：20230215

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.最佳实践

1.1. Gitlab对接（SAML）

GitLab支持SAML协议的参考官方文档配置：https://docs.gitlab.com/ee/integration/saml.html

Gitlab本地部署可以参考 https://www.cnblogs.com/straycats/p/7637373.html 版本号：gitlab-ce-12.2.1-ce.0.el7.x86_64.rpm

Gitlab常用命令：

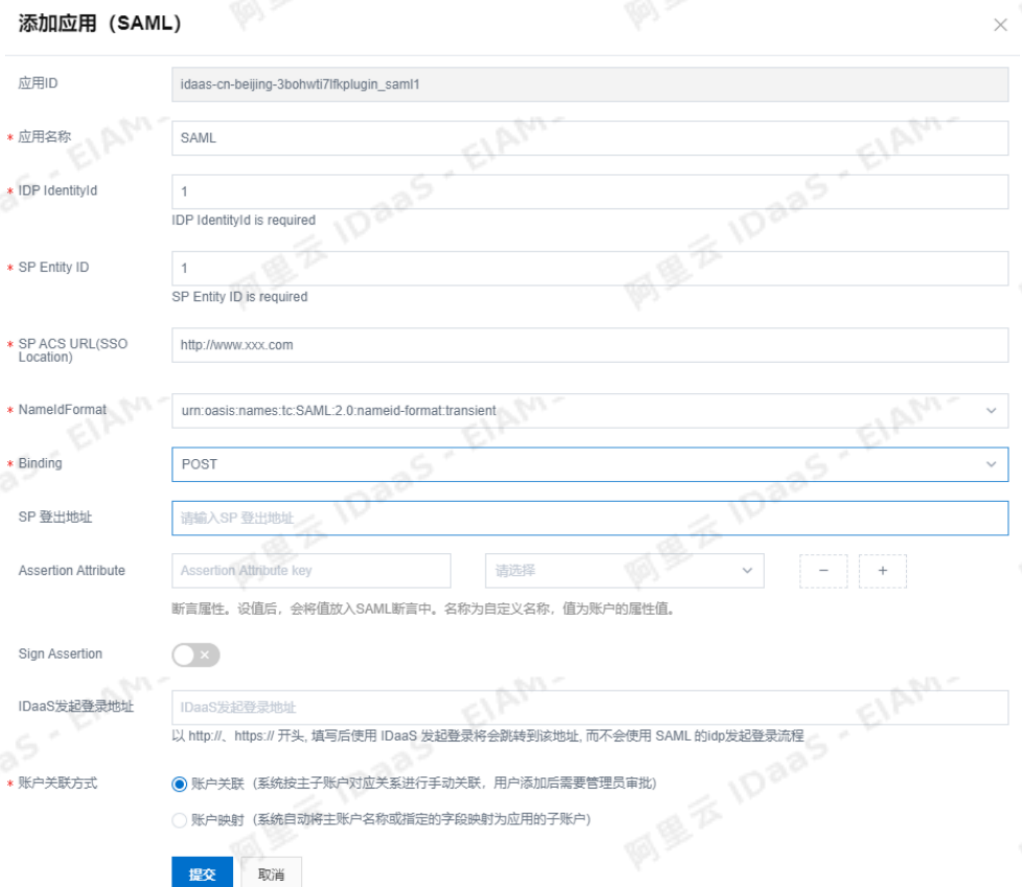
```
# 启动Gitlab
gitlab-ctl start
# 停止Gitlab
gitlab-ctl stop
# 重启Gitlab
gitlab-ctl restart
# 重新加载Gitlab配置
gitlab-ctl reconfigure
# 查看状态
gitlab-ctl status
# 查看所有的logs
gitlab-ctl tail
```

一、在IDaaS中创建一个SAML应用

1. 在添加应用页面，选择saml应用点击添加

2. 添加SigningKey

3. 创建SAML应用配置参数可以先随意填写，之后还要进行修改



二、修改gitlab配置文件

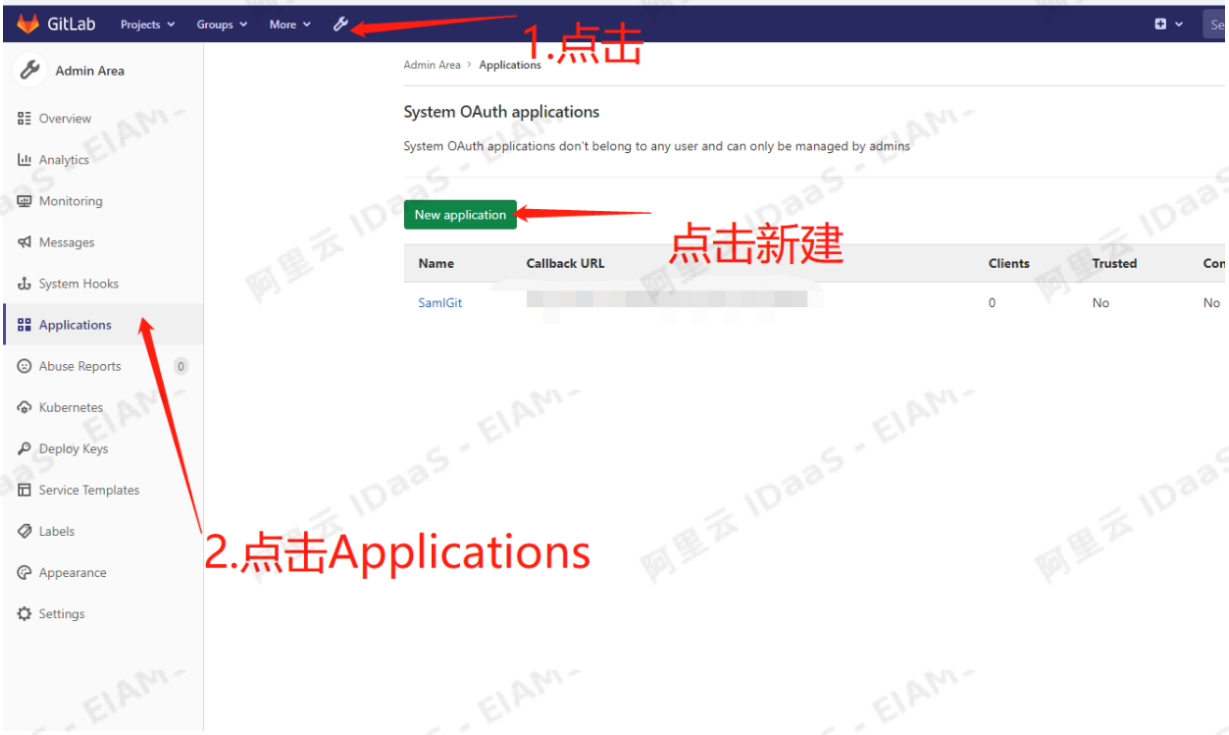
> 重要
>
> 在修改配置时请将下面的注释删除，避免gitlab配置格式的影响

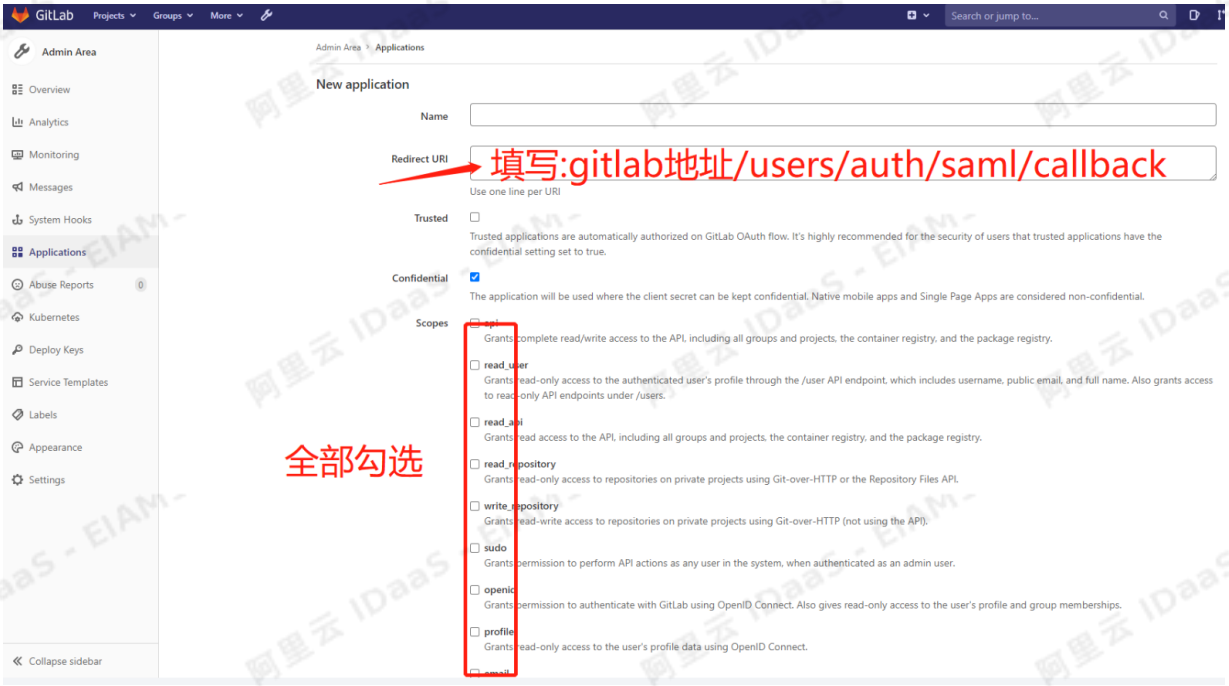
vim /etc/gitlab/gitlab.rb

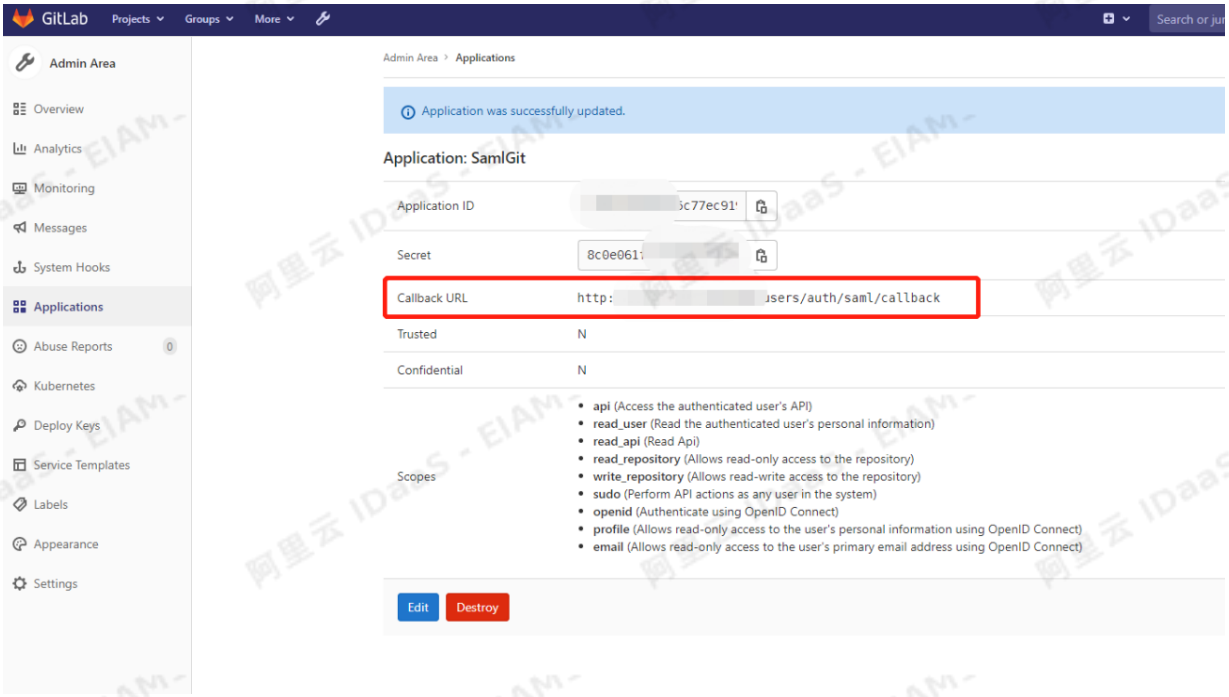
```
#允许用户使用SAML进行注册而无需手动创建账户
gitlab_rails['omniauth_enabled'] = true
gitlab_rails['omniauth_allow_single_sign_on'] = ['saml']
gitlab_rails['omniauth_block_auto_created_users'] = false
#设置自动将SAML用户与现有的GitLab用户连接
gitlab_rails['omniauth_auto_link_saml_user'] = true
#添加提供程序配置
gitlab_rails['omniauth_providers']=[
{
        name: 'saml',
        args: {
                #gitlab的断言地址，标绿部分替换为gitlab实际的地址
                  assertion_consumer_service_url:'http://192.168.20.178/users/auth/saml/callback',
                #证书的指纹信息，在IDaaS配置SAML应用后获取
                idp_cert_fingerprint:'a9:68:15:e2:35:3f:1e:de:ea:ac:26:62:a3:88:aa:9c:62:4e:e7:8a',
                #固定格式,http://192.168.20.173:8080为IDaaS域名地址，lin1121saml IDaaS中对应的saml应用ID,请根据实际信息进行调整
                idp_sso_target_url:'http://192.168.20.173:8080/enduser/api/application/plugin_saml/lin1121saml/sp_sso',
                #一般为固定格式，可以在gitlab登录页获取
                issuer:'http://192.168.20.178/users/auth/saml',
                name_identifier_format:'urn:oasis:names:tc:SAML:2.0:nameid-format:persistent',
              },
        #标签，可以根据需要随意更改
        label:'IDaaS'
}]
```

各参数具体获取方法如下

- assertion_consumer_service_url获取方式如下：


GitLab
Projects
Groups
More
1.点击
Admin Area
Overview
Analytics
Monitoring
Messages
System Hooks
Applications
Abuse Reports
Kubernetes
Deploy Keys
Service Templates
Labels
Appearance
Settings
Admin Area › Applications
System OAuth applications
System OAuth applications don't belong to any user and can only be managed by admins
New application
点击新建
Name
Callback URL
Clients
Trusted
SamlGit
0
No
No
2.点击Applications

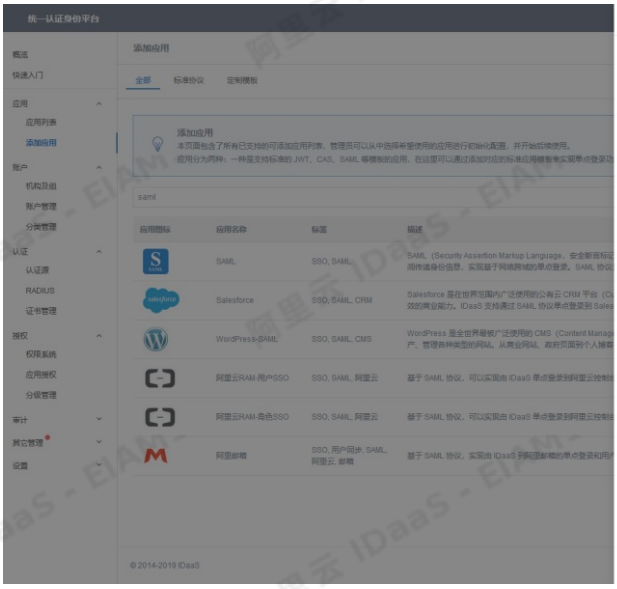

GitLab
Projects
Groups
More
Search or jump to...
Admin Area
Overview
Analytics
Monitoring
Messages
System Hooks
Applications
Abuse Reports
Kubernetes
Deploy Keys
Service Templates
Labels
Appearance
Settings
Collapse sidebar
Admin Area › Applications
New application
Name
Redirect URI
填写:gitlab地址/users/auth/saml/callback
Use one line per URI
Trusted
Trusted applications are automatically authorized on GitLab OAuth flow. It's highly recommended for the security of users that trusted applications have the confidential setting set to true.
Confidential
The application will be used where the client secret can be kept confidential. Native mobile apps and Single Page Apps are considered non-confidential.
Scopes
全部勾选
api
Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry.
read_user
Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.
read_api
Grants read access to the API, including all groups and projects, the container registry, and the package registry.
read_repository
Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.
write_repository
Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).
sudo
Grants permission to perform API actions as any user in the system, when authenticated as an admin user.
openid
Grants permission to authenticate with GitLab using OpenID Connect. Also gives read-only access to the user's profile and group memberships.
profile
Grants read-only access to the user's profile data using OpenID Connect.



- idp_cert_fingerprint获取方式如下：

1、导出证书

在添加应用界面，选择saml应用模板，点击添加应用。选择之前创建的SigningKey，点击导出

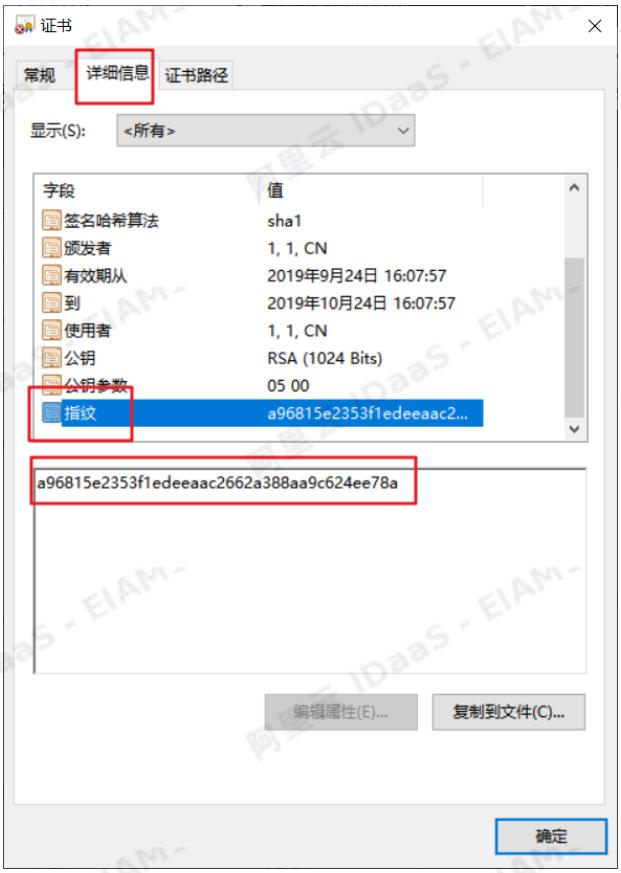


2、查看指纹。

打开证书，点击详细信息-指纹获取指纹。**每两个数字后面加上一个冒号：**，跟文档的格式保持一致。

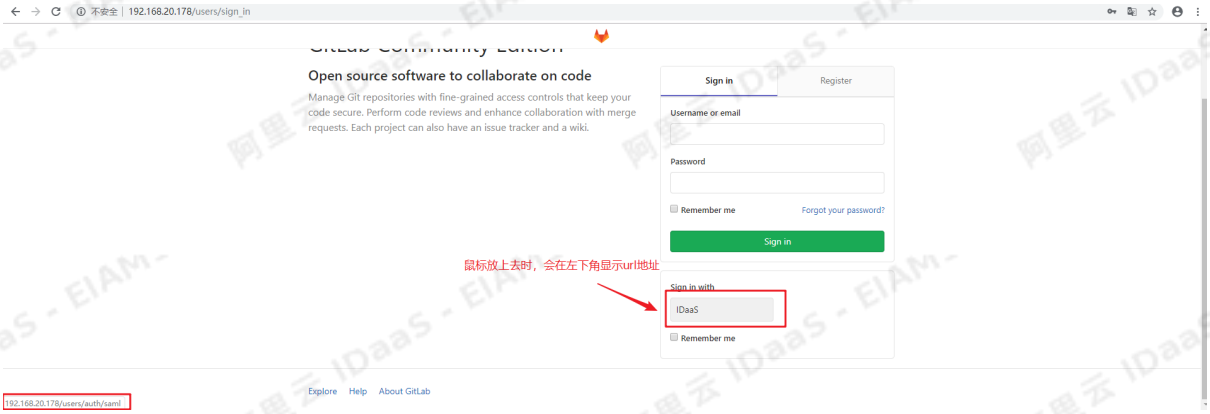
警告

每两个数字后面加上一个冒号“:”，否则将报错。



- idp_sso_target_url获取方式

- issuer获取方式如下：



修改完配置之后，在命令行中输入以下命令重启gitlab，刷新配置信息

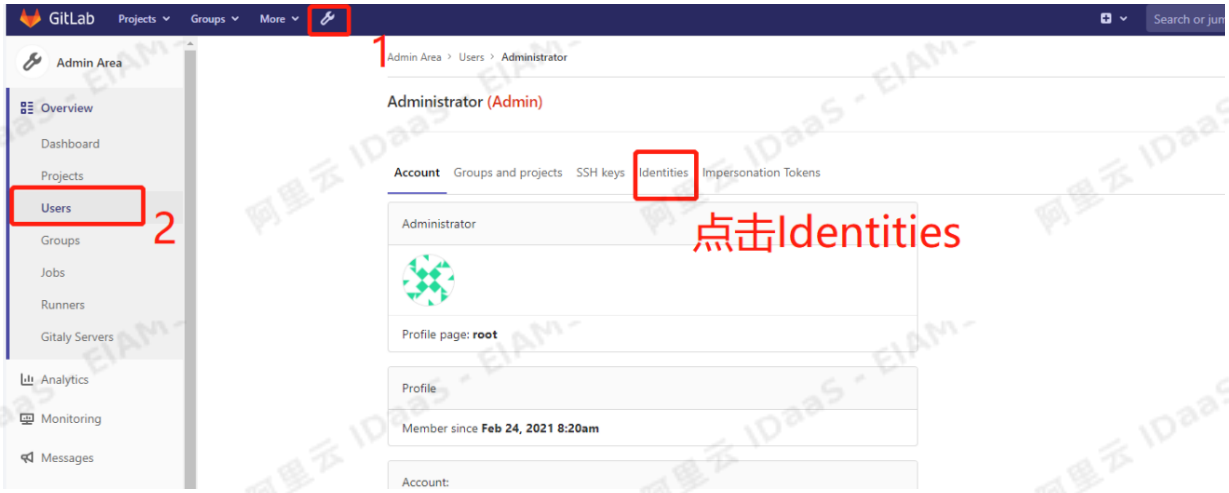
```
gitlab-ctl stop
gitlab-ctl reconfigure
gitlab-ctl start
```

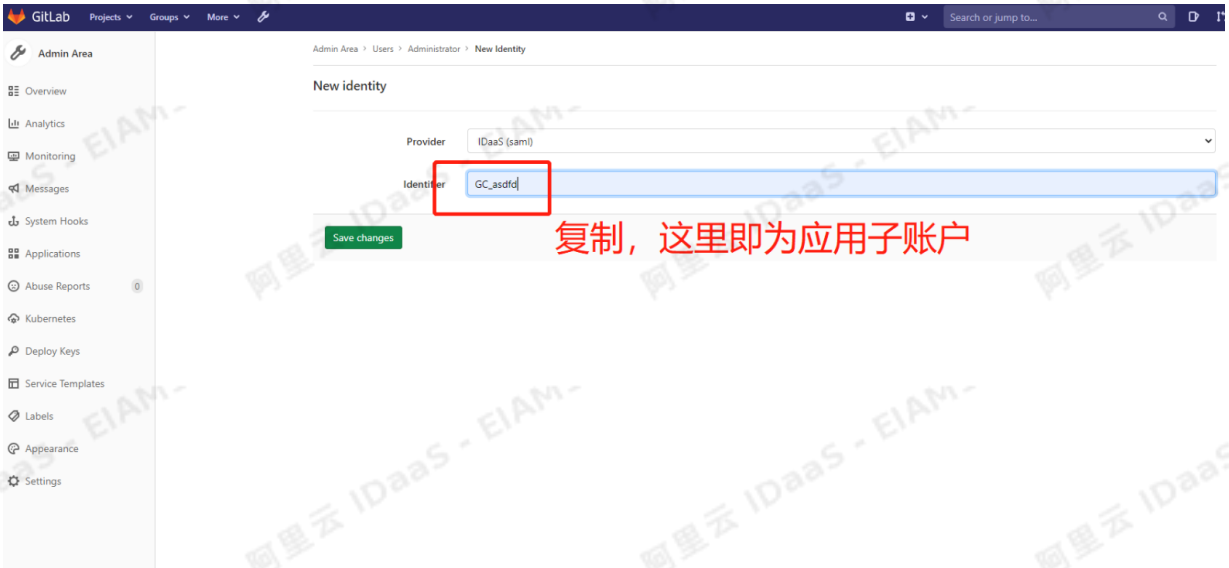
三、修改saml应用配置

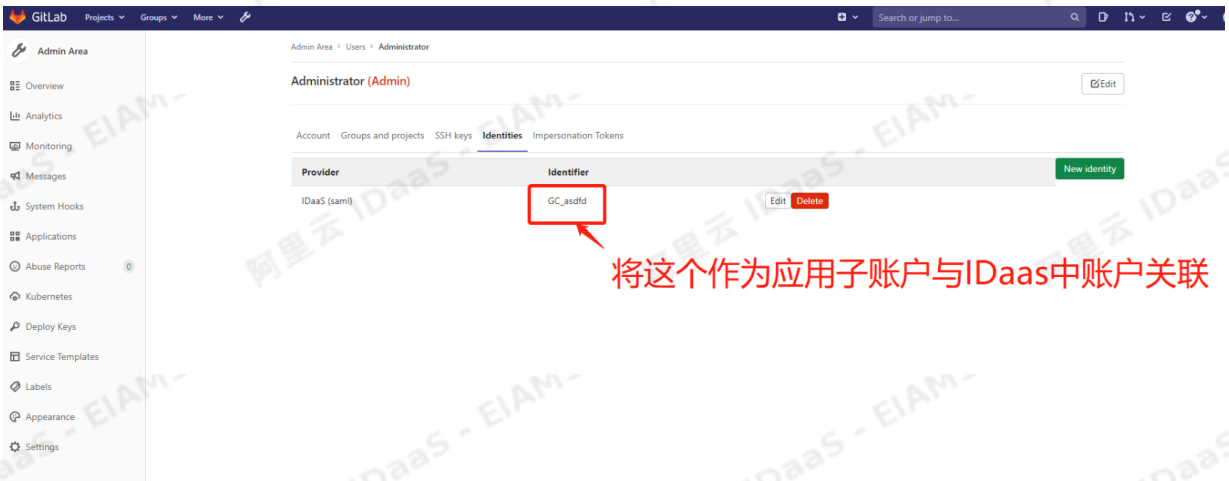
修改saml应用配置如下

四、设置Gitlab对应账户标识（IDaaS中的子账户）

使用root账户登录，为账户添加账户标识

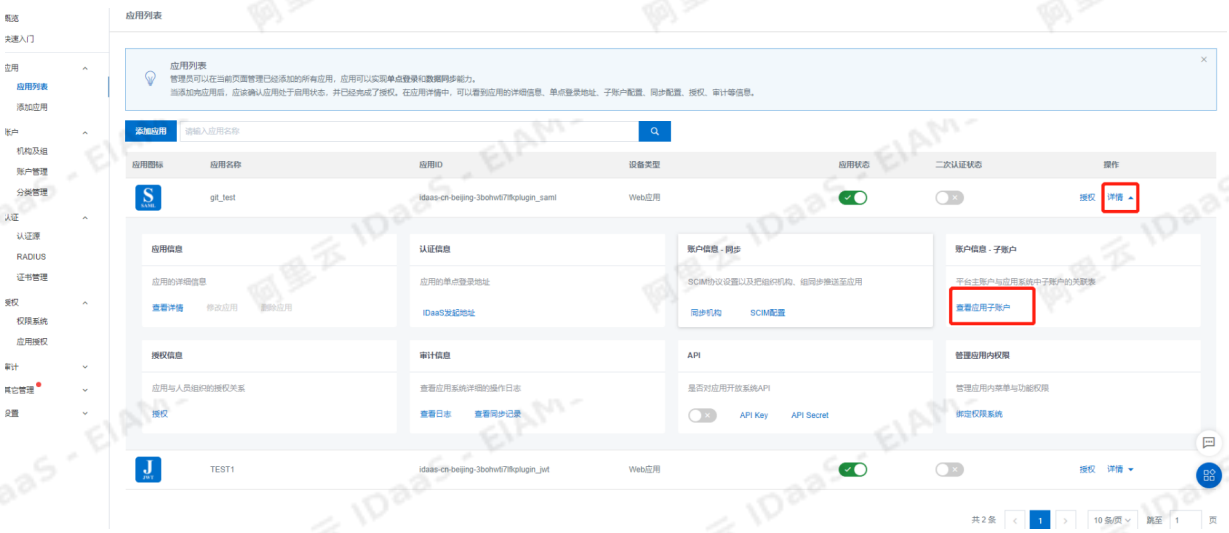






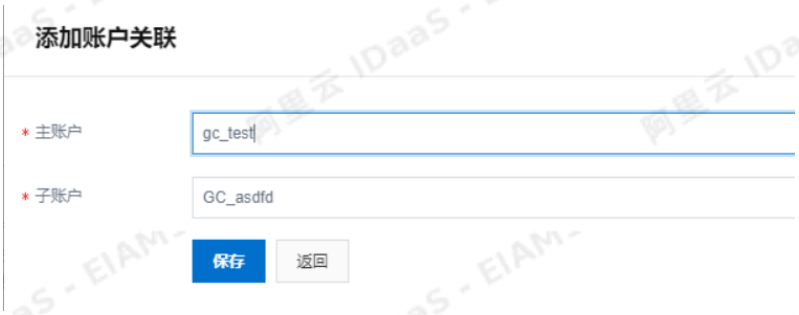
五、IDaaS添加子账户

点击应用详情，查看应用子账户。



点击添加账户关联

主账户为IDaas中的账户，子账户为刚刚Gitlab中设置的 identity.



完成以上步骤，即可单点登录到gitlab。

FAQ

1. 显示下图报错

gitLab的identifier和应用下设置的子账户需要一致。

2. 使用同步进入gitlab的账户进行登录，提示邮箱不能为空。

方法1：绑定主子账户时，子账户直接使用用户名，不使用邮箱；

方法2：在IDaaS中，修改SAML应用添加邮箱参数

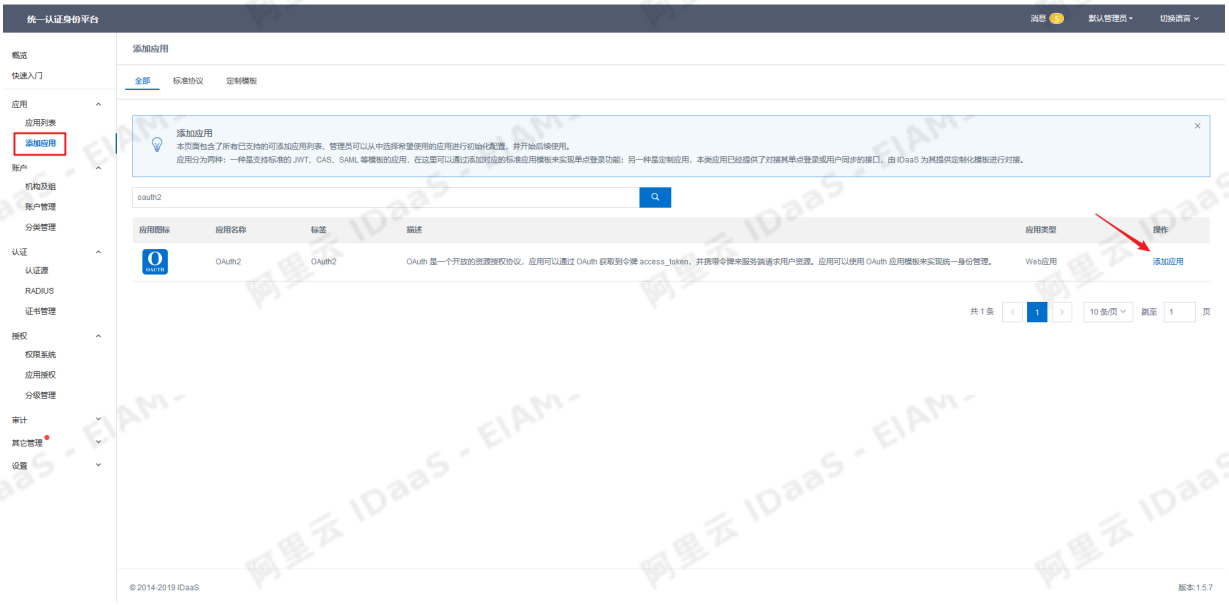
1.2. Gitlab对接（OAuth2）

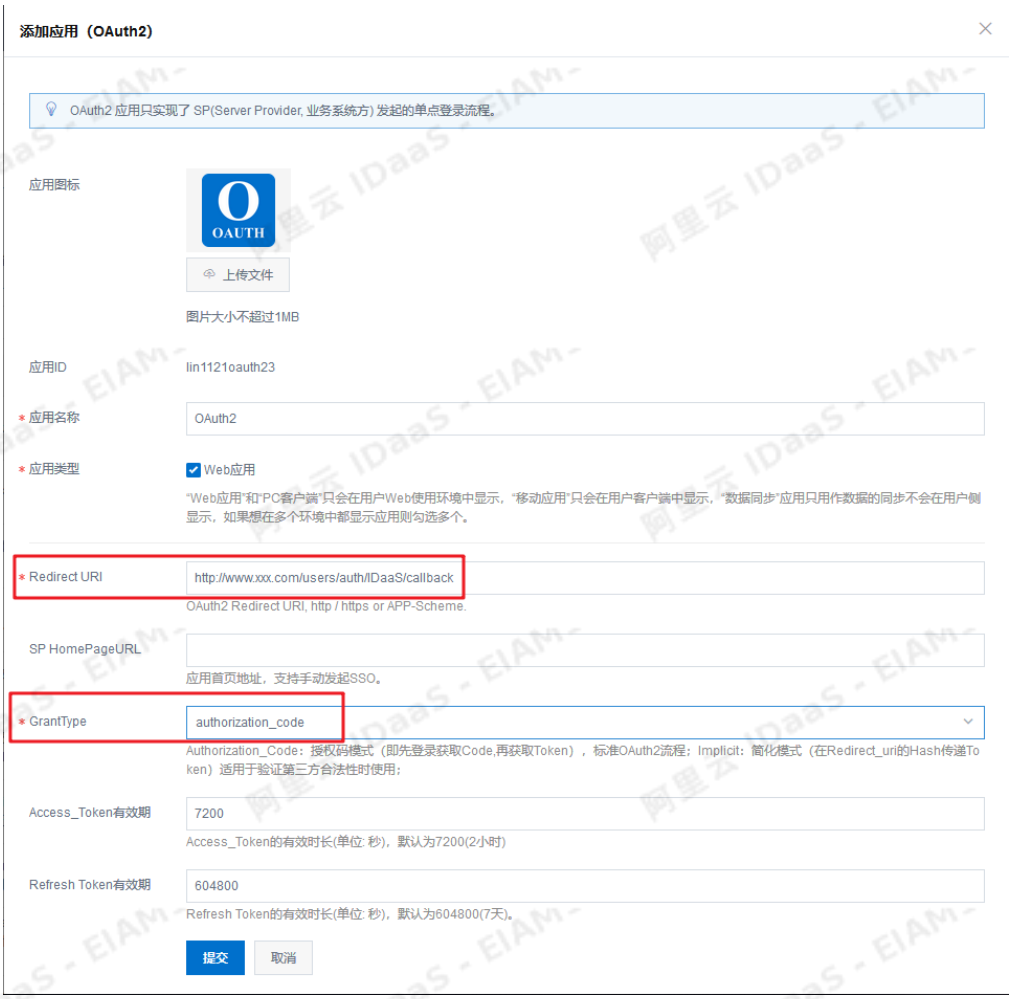
Gitlab常用命令：

```
# 启动Gitlab
gitlab-ctl start
# 停止Gitlab
gitlab-ctl stop
# 重启Gitlab
gitlab-ctl restart
# 重新加载Gitlab配置
gitlab-ctl reconfigure
# 查看状态
gitlab-ctl status
# 查看所有的logs
gitlab-ctl tail
```

一、在IDaaS中创建一个OAuth2应用

点击导航栏中点击添加应用，选择OAuth2应用模板





添加应用需要填写两个参数，Redirect URI和GrantType

Redirect URI填写格式如下：Gitlab_url/users/auth/IDaaS/callback。

其中Gitlab_url为gitlab服务器的地址，IDaaS为在Gitlab服务器配置文件里配置的标识。

GrantType选择Authorization_Code（授权码模式）

二、修改gitlab配置文件

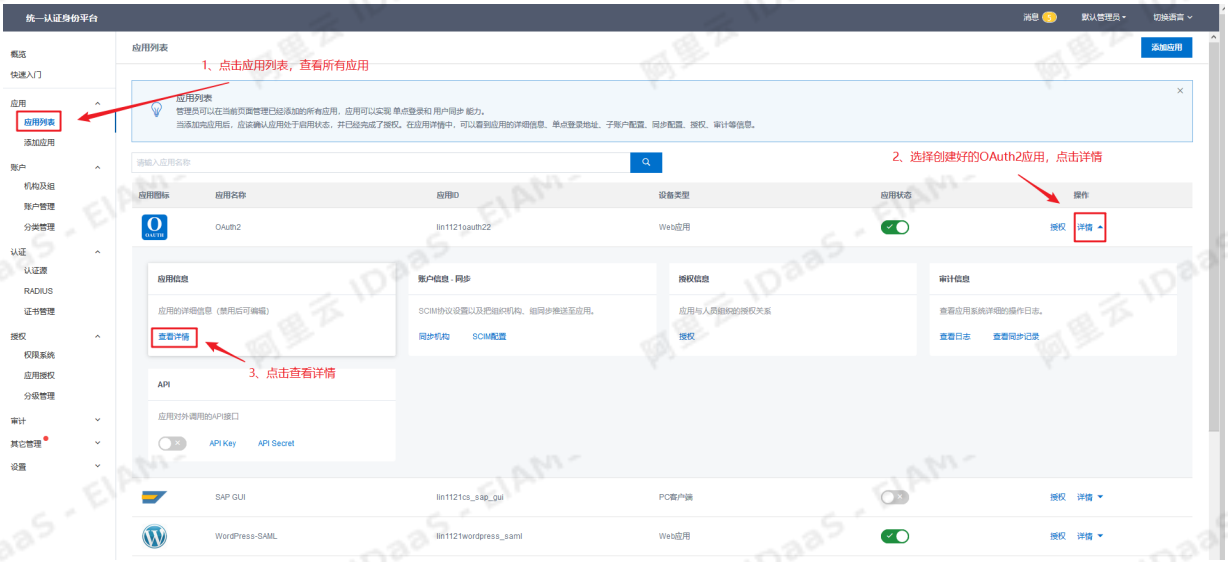
（在修改配置时请将下面的注释删除，避免gitlab配置格式的影响）

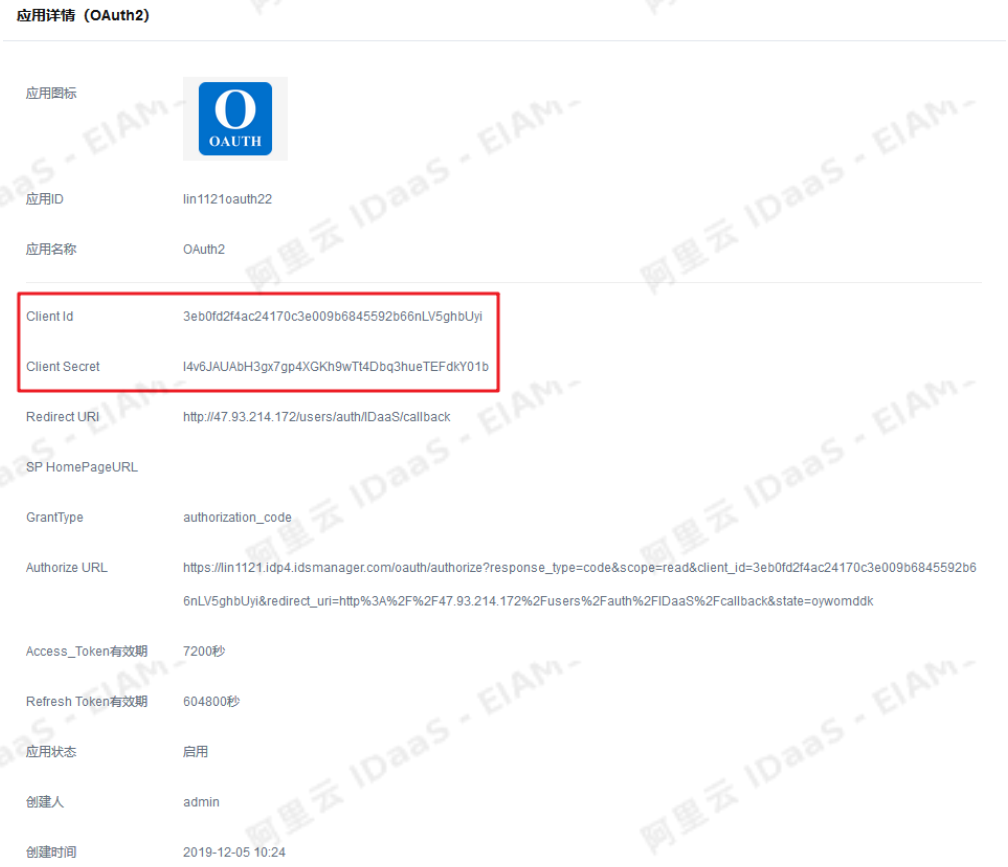
vim /etc/gitlab/gitlab.rb

```
#允许用户使用oauth2进行单点登录
gitlab_rails['omniauth_enabled'] = true
gitlab_rails['omniauth_allow_single_sign_on'] = ['oauth2_generic']
gitlab_rails['omniauth_block_auto_created_users'] = false
#添加提供程序配置
gitlab_rails['omniauth_providers'] = [
 {
       'name' => 'oauth2_generic',
       #IDaaS中OAuth2应用的client_id和client_secret
        'app_id' => 'oauth_client_app_id',
        'app_secret' => 'oauth_client_app_secret',
        'args' => {
              client_options: {
                #IDaaS服务器的地址
                'site' => 'https://your_oauth_server',
                #IDaaS提供的获取用户信息的接口
                 'user_info_url' => '/api/bff/v1.2/commons/user_details',
                #IDaaS提供的获取token的地址
                'token_url' => '/oauth/token'
              },
          user_response_structure: {
             root_path: ['data', 'udAccountInformation'],
             attributes: { nickname: 'username' }
           },
          #标签名，可以随意填写，但需要和创建OAuth2时填写的标识统一
          name: 'IDaaS',
           strategy_class: "OmniAuth::Strategies::OAuth2Generic"
      }
 }]
```

oauth_client_app_id和oauth_client_app_secret获取方式如下：

选择步骤一创建的OAuth2应用，点击查看详情

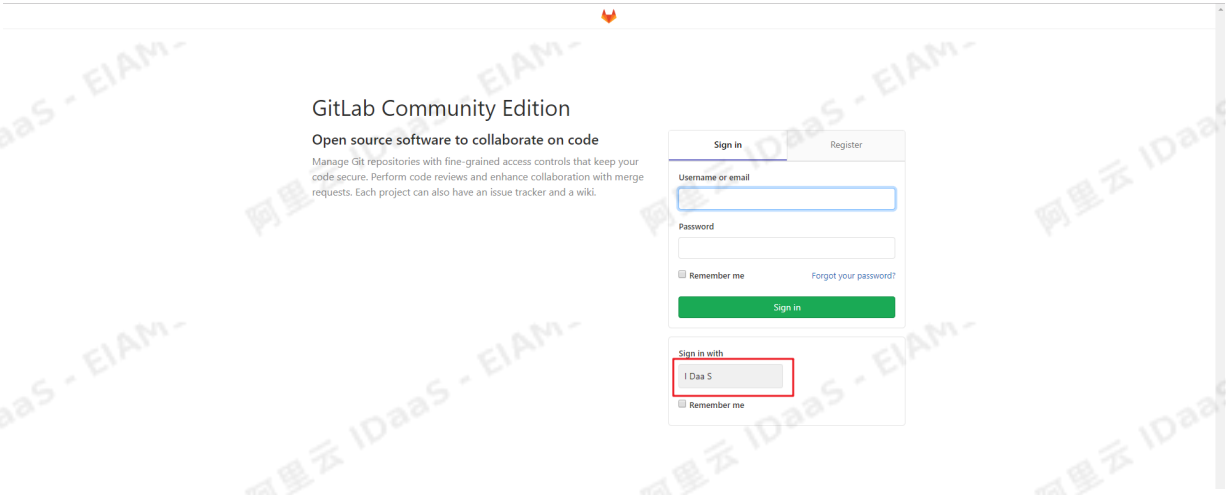




修改完配置之后，在命令行中输入以下命令重启gitlab，刷新配置信息。

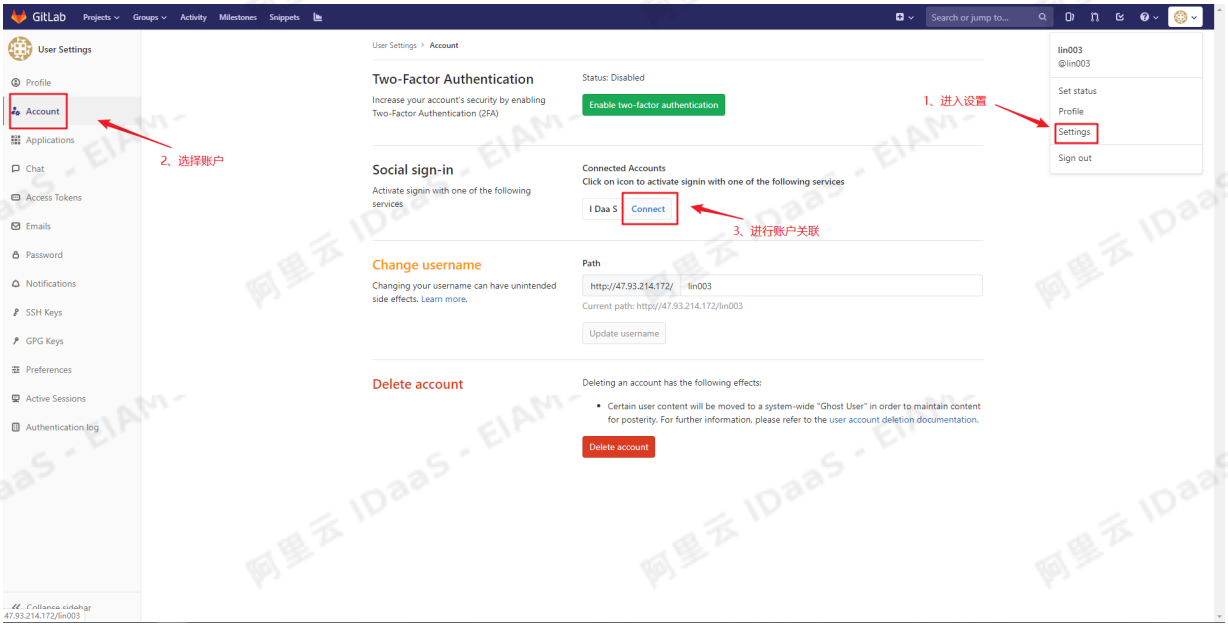
```
gitlab-ctl stop
gitlab-ctl reconfigure
gitlab-ctl start
```

重启完成之后，可以在gitlab登录界面如下：



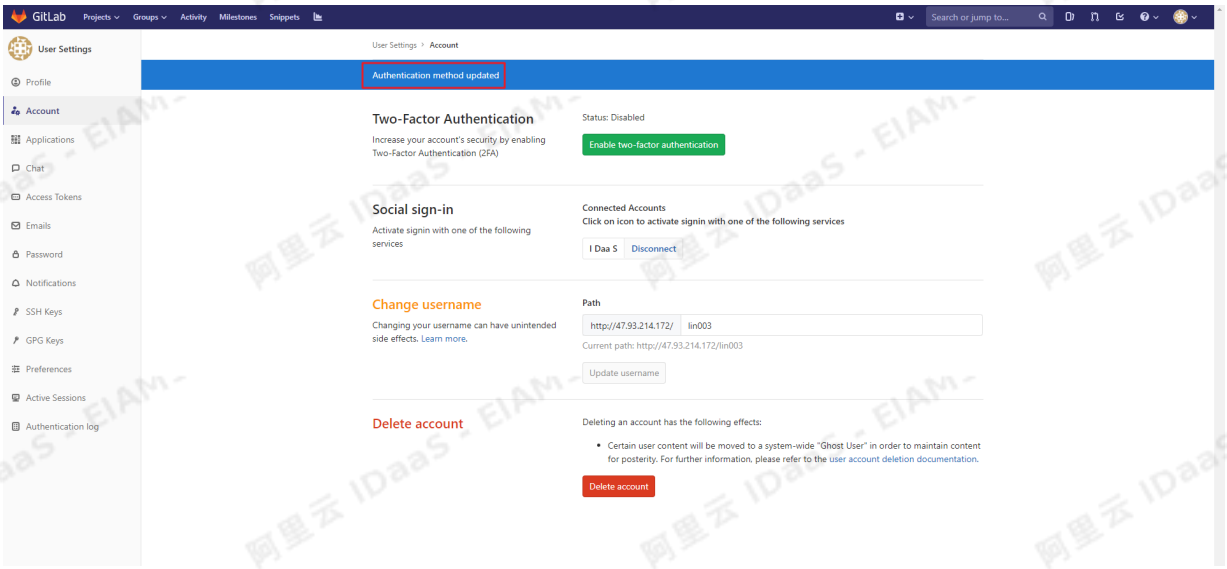
三、账户关联

用户登录gitlab之后，在setting-Account 中点击Connect进行账户关联



跳转到IDaaS的登录界面，使用IDaaS的账户进行登录

登录成功之后页面会切换回Gitlab的页面，并在页面上方出现一条提示信息



通过以上步骤，完成使用OAuth2应用模板实现Gitlab的单点登录。

1.3. JIRA、Confluence、bitbucket对接-使用miniOrange Single Sign On

本文为您介绍如何通过SAML协议单点登录到JIRA，Confluence或者bitbucket，实现应用的快捷登录，提升员工办公体验。我们此处演示的是单点登录到阿里云控制台

背景信息

某些企业员工日常办公需访问JIRA，Confluence 或者bitbucket，每次都需要访问应用的登录地址，输入账户名，密码登认证方式进行登录，如果有多个类似应用，就需要记录多套密码，使用应用时繁琐和耗时。

解决方案

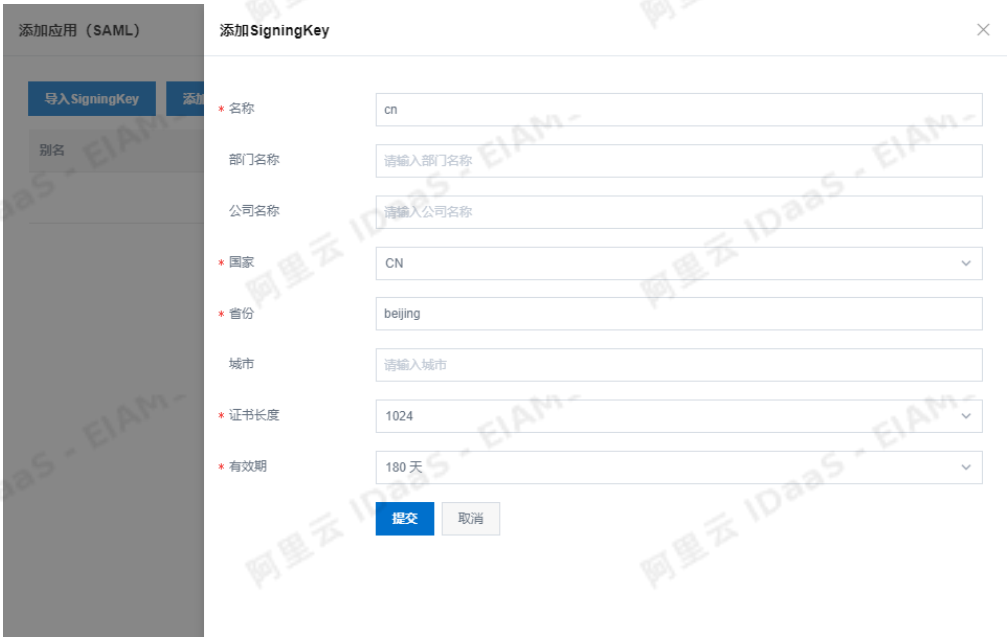
IDaaS应用身份服务通过账户单点登录到JIRA，Confluence或者bitbucket，只需要登录一次，就可以看到所有有权限访问的应用，并对应用进行单点登录。

JIRA，Confluence 和 bitbucket配置步骤一样，都使用下面的操作步骤进行配置。

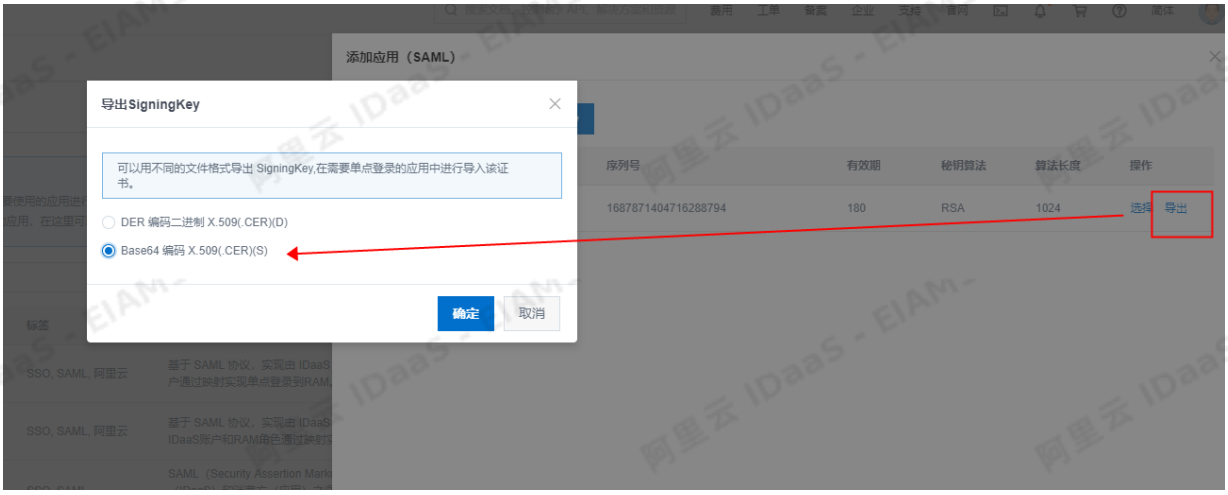
操作步骤

1. 以IT管理员账号登录云盾IDaaS管理平台。具体操作请参考 IT管理员指南-登录。
2. 在左侧导航栏，点击 **应用 > 添加应用**，选择SAML应用模板，点击**添加应用**。

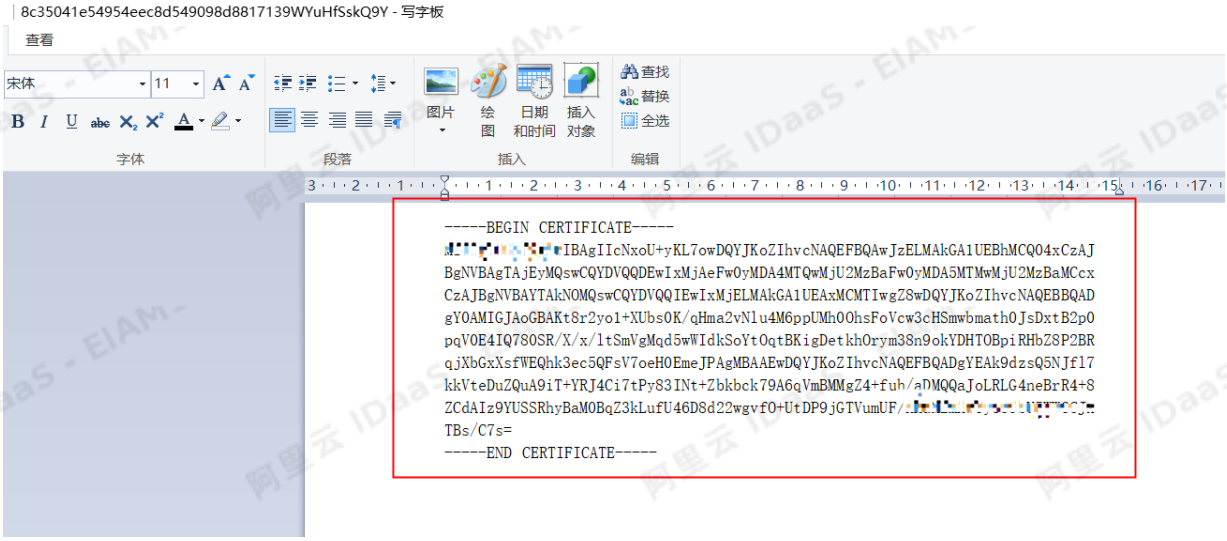
添加SigningKey



3. 导出SigningKey文件



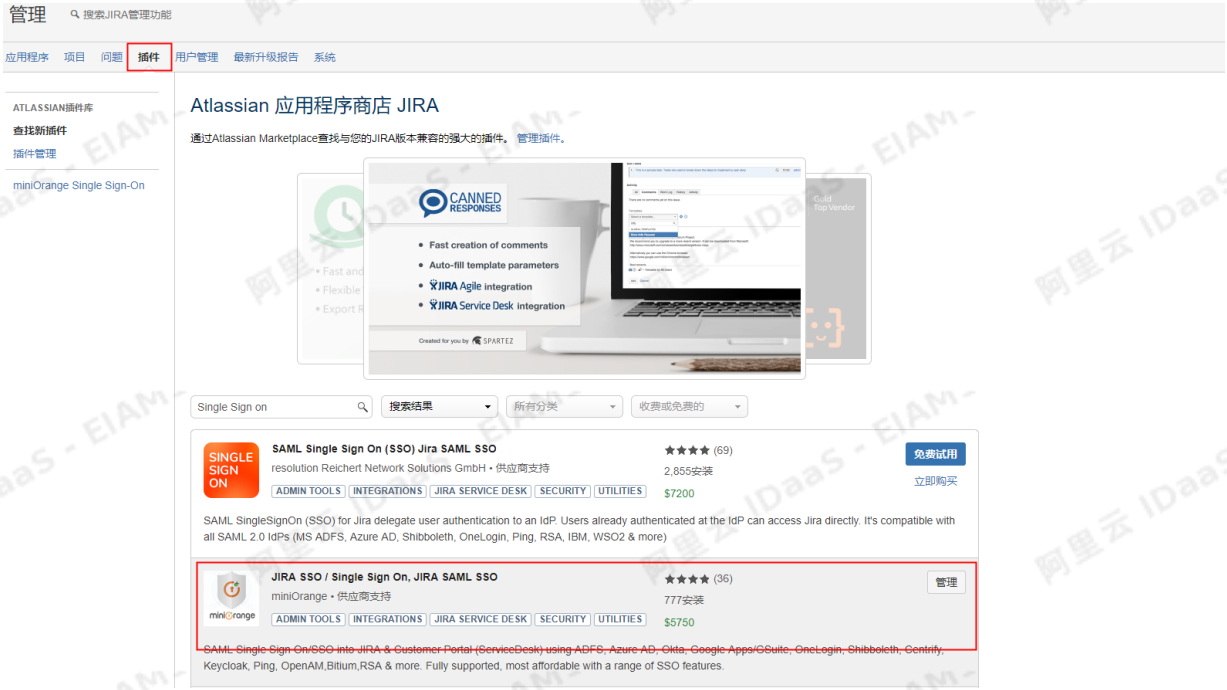
采用文本编辑器打开，获取到——BEGIN CERTIFICATE—- ——END CERTIFICATE——证书信息。 后面操作将用到该值



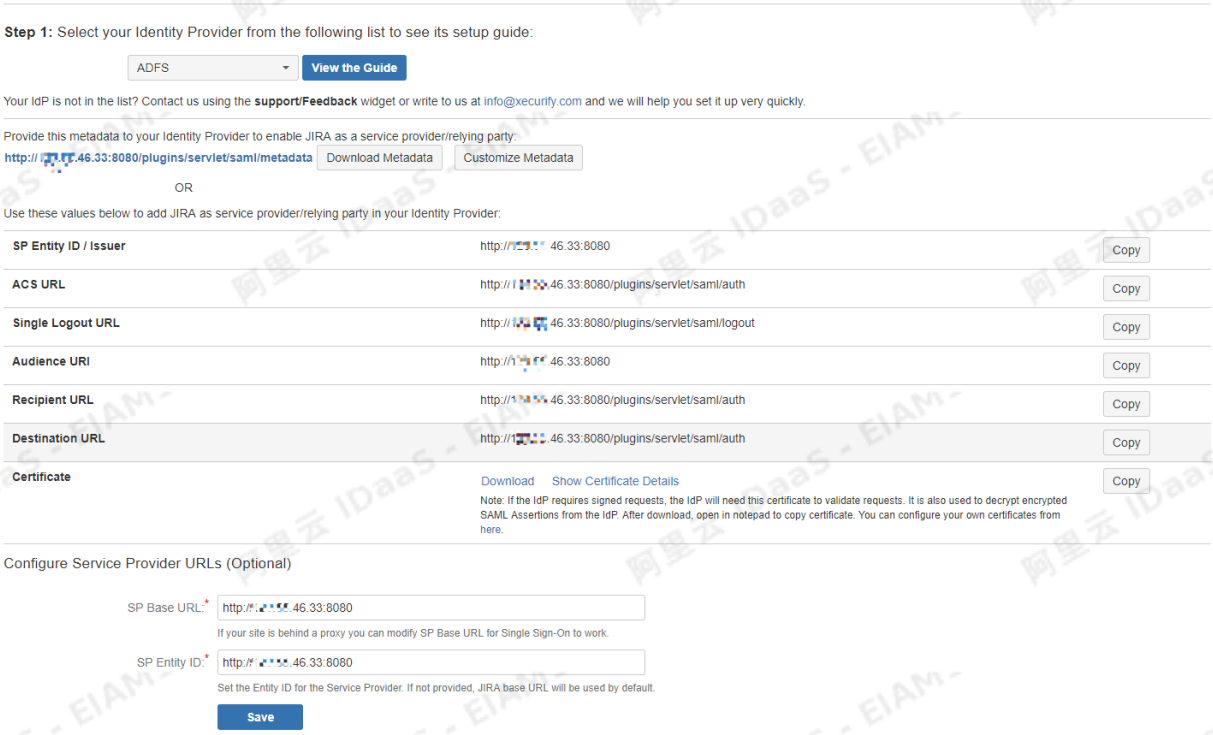
4. JIRA/Confluence页面配置

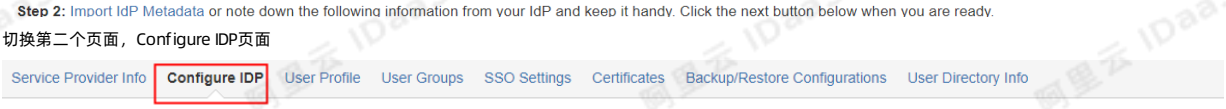
访问系统

访问插件页面，安装miniOrange Single Sign On插件， 该插件需要付费购买，是JIRA/Confluence 标准单点登录插件。

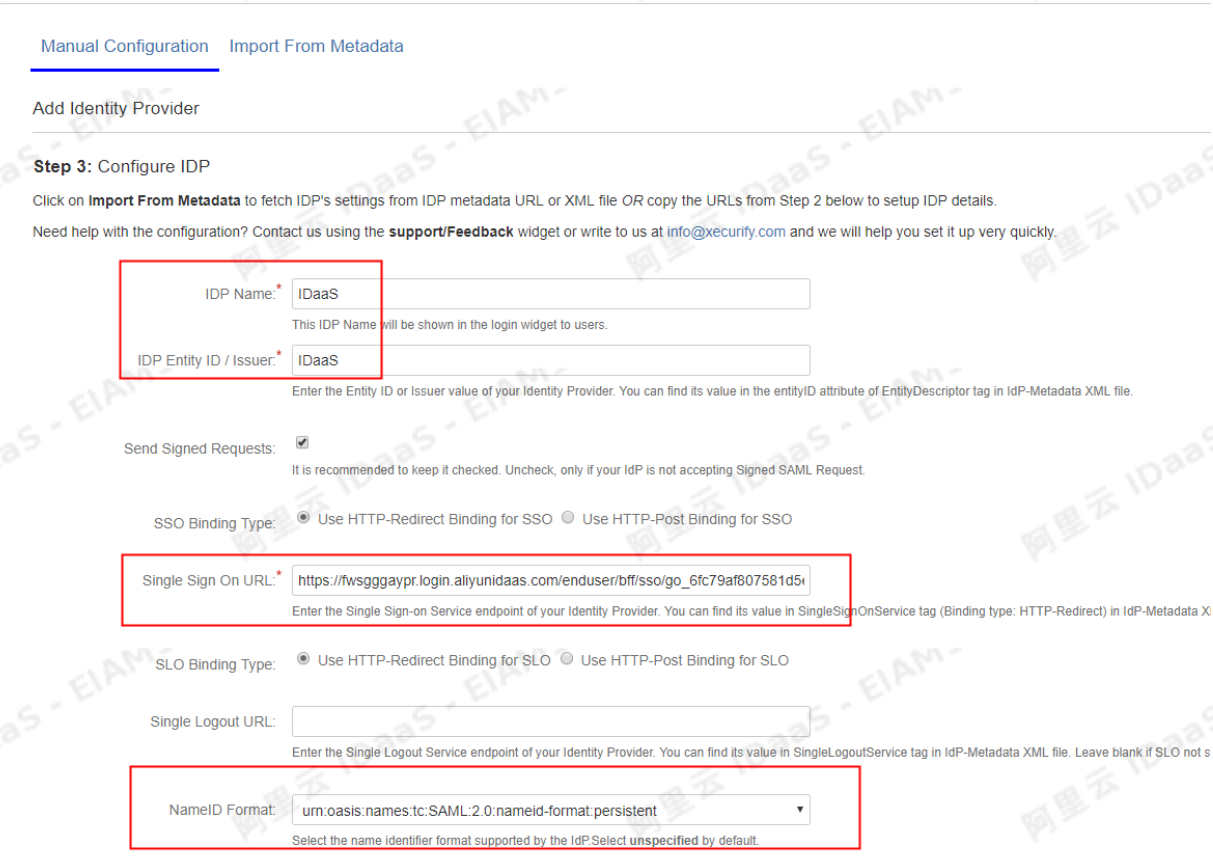


访问miniOrange Single Sign On插件配置页面，参数全部默认生成，只需要调整SP Base URL和SP Entity ID,填JIRA/Confluence的访问地址





切换第二个页面，Configure IDP页面



参数说明

- IDP Name：可以随意填写；

- IDP Entity ID/Issuer：需要和IDaaS的IDaaS IdentityId值一致，建议统一写成：IDaaS；
- Send Signed Requests 需要勾选；
- SSO Binding Type：勾选第一项；
- Single Sign On URL: 填IDaaS的发起地址，下面步骤中将介绍获取方式
- NameID Format：选择SAML:2.0 nameid-format persistent；（IDaaS应用中需配置一致）；
- IDP Signing Certificate: 此处填步骤3中获取的证书信息。

5. 返回IDaaS控制台，继续创建SMAL应用

选择SigningKey

填写应用信息

- 应用名称： 可以随意填写；
- IDaaS IdentityId： 建议统一写成：IDaaS；
- SP Entity ID： 填写JIRA/Confluence 基础访问地址
- SP ACS URL(SSO Location)： 填写JIRA/Confluence Service Provider Info 页面的ACS URL 参数
- NameIdFormat： 选择SMAL:2.0 nameid-format persistent
- 选择一种账户关联方式进行提交，应用创建成功

在机构及组页面创建一个IDaaS账户

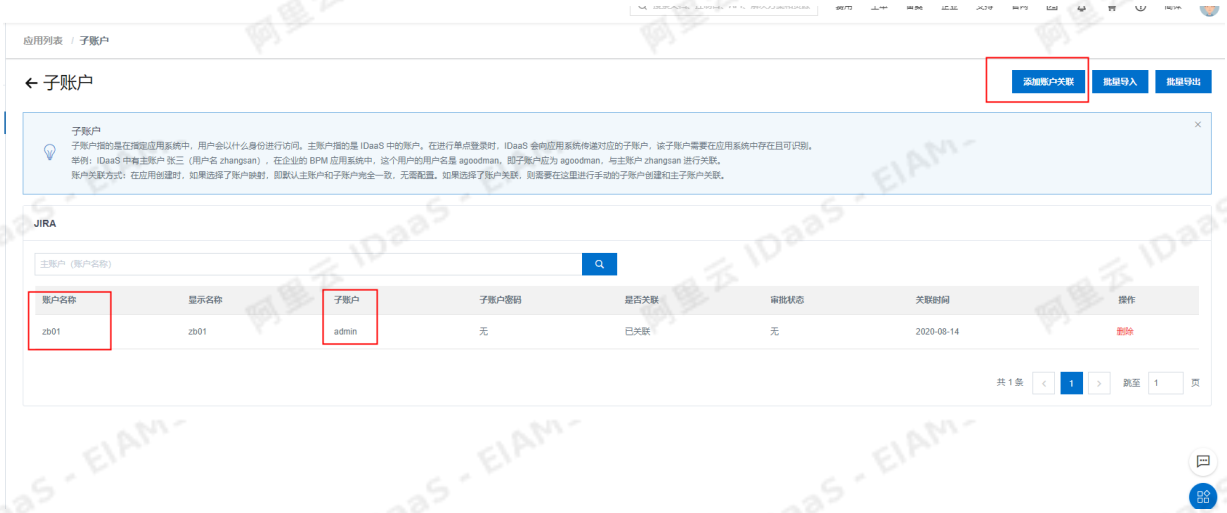
在应用授权页面，把应用授权给新创建的账户，并保存

访问应用列表，点开应用详情，点击查看应用子账户

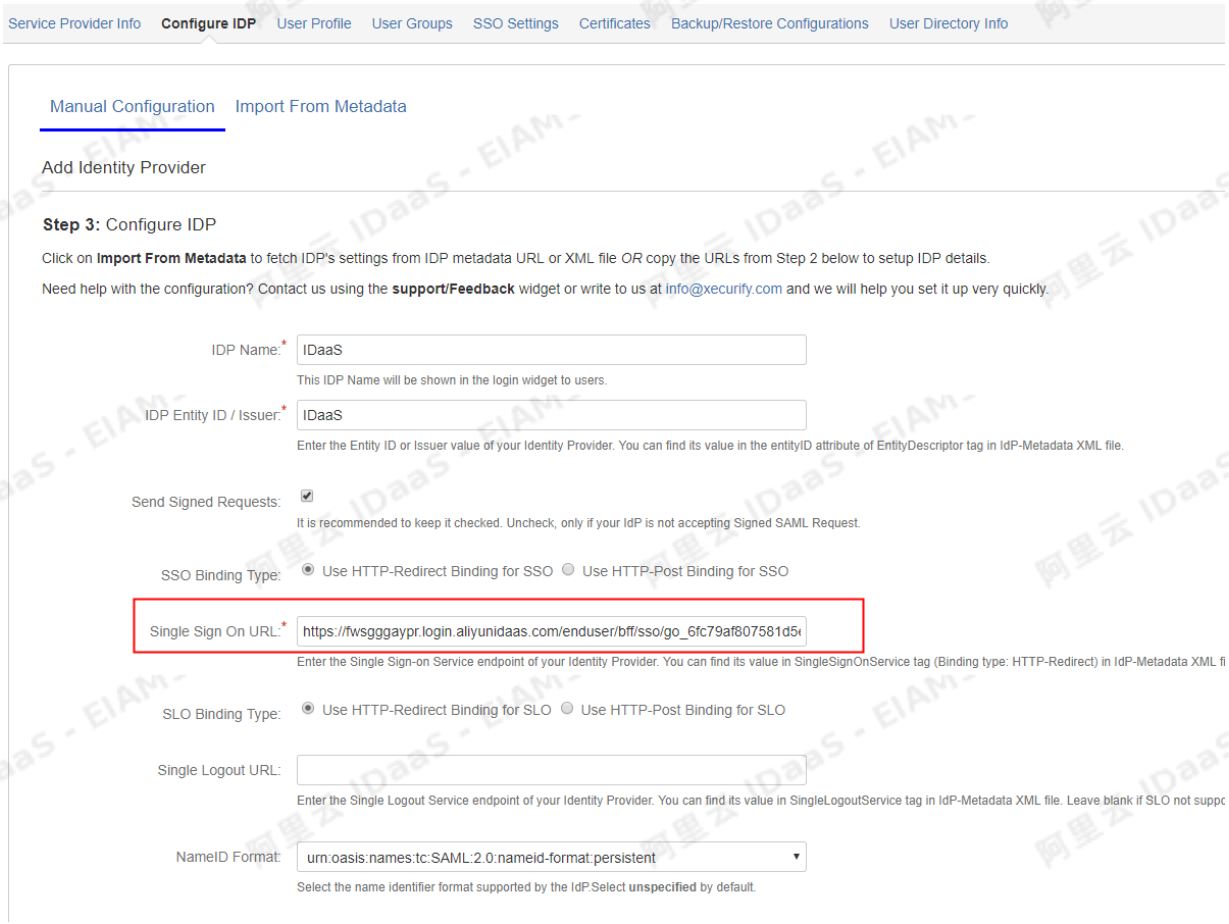
添加账户关联

主账户：IDaaS中的账户，上面在账户及组页面创建的账户

子账户：JIRA/Confluence 中的账户，如登录JIRA/Confluence账户名是admin，子账户填写admin



返回应用详情，复制 IDaaS发起地址， 粘贴到JIRA/Confluence的 Configure IDP页面Single Sign On URL 参数中，并进行保存。

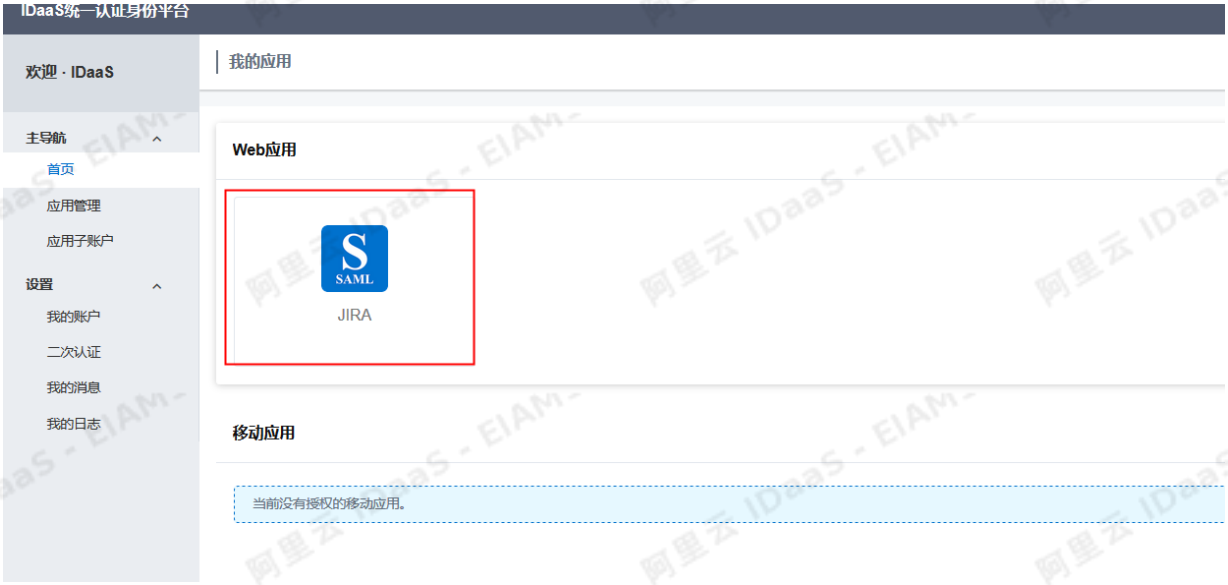


6. 单点登录JIRA/Confluence

访问IDaaS 用户登录地址

输入上文中创建的IDaaS账户进行登录

点击应用图标进行单点登录



FAQ

IDaaS是否支持同步数据到JIRA/Confluence

IDaaS不支持直接和JIRA, Confluence 进行数据同步，因为JIRA, Confluence是标准产品不支持改造， 只能适应JIRA, Confluence支持的LDAP同步方式。 建议同步流程： IDaaS账户数据变动可以同步到LDAP, JIRA, Confluence 再拉取LDAP中的账户变动信息实现同步。IDaaS同步数据到LDAP，请参考LDAP账户同步配置。

IDaaS不支持直接和JIRA, Confluence 进行数据同步，因为JIRA, Confluence是标准产品不支持改造， 只能适应JIRA, Confluence支持的LDAP同步方式。 建议同步流程： IDaaS账户数据变动可以同步到LDAP, JIRA, Confluence 再拉取LDAP中的账户变动信息实现同步IDaaS同步数据到LDAP，请参考LDAP账户同步配置。

1.4. JIRA对接-使用SSO 2.0

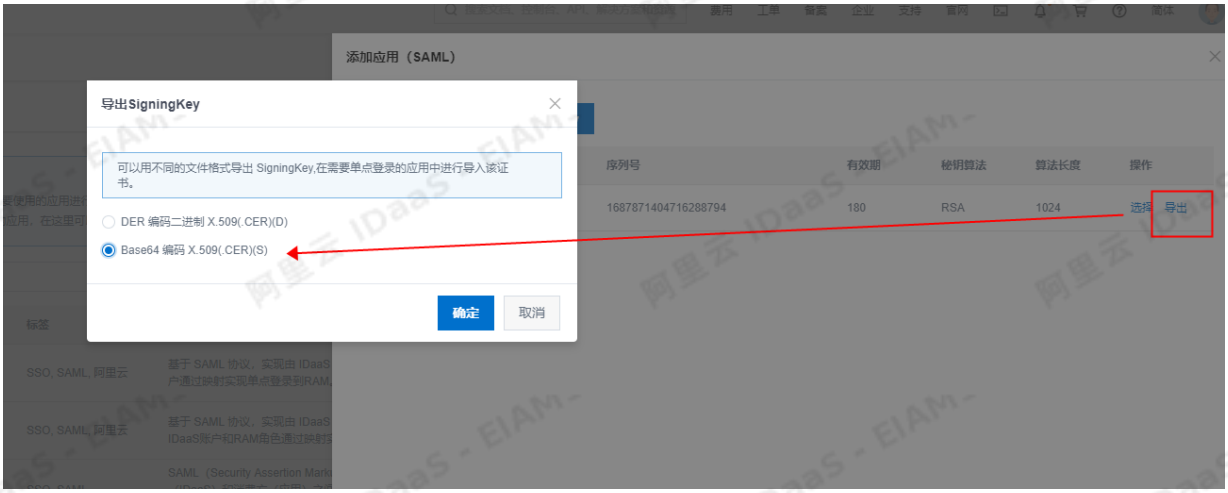
本文为您介绍如何通过Jira的SSO 2.0配置单点登录。

操作步骤

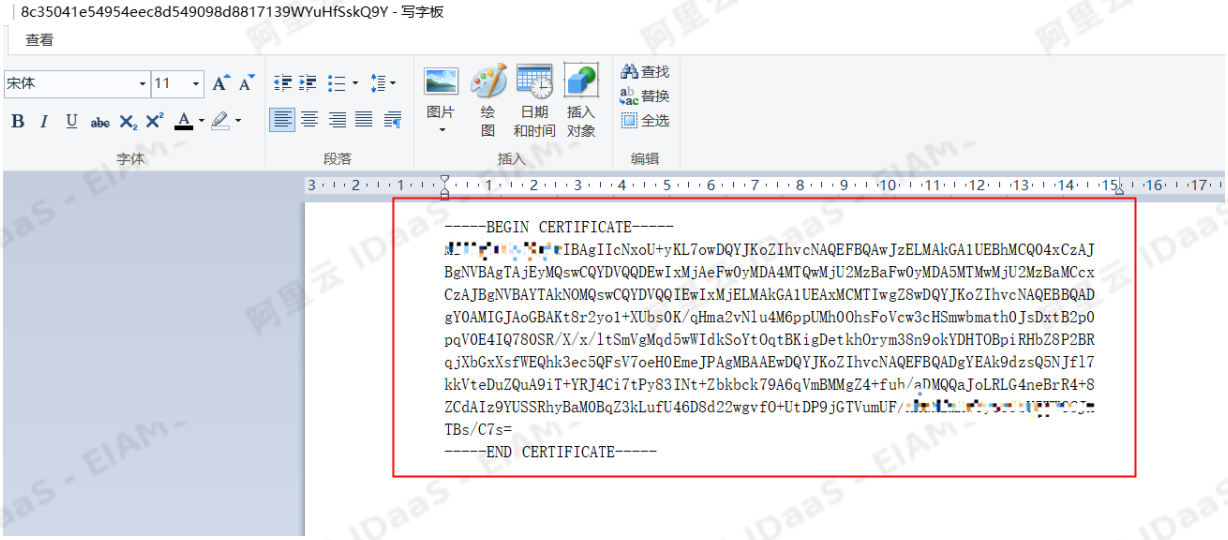
1. 以IT管理员账号登录云盾IDaaS管理平台。具体操作请参考 IT管理员指南-登录。
2. 在左侧导航栏，点击 **应用 > 添加应用**，选择SAML应用模板，点击**添加应用**。

添加SigningKey

3. 导出SigningKey文件



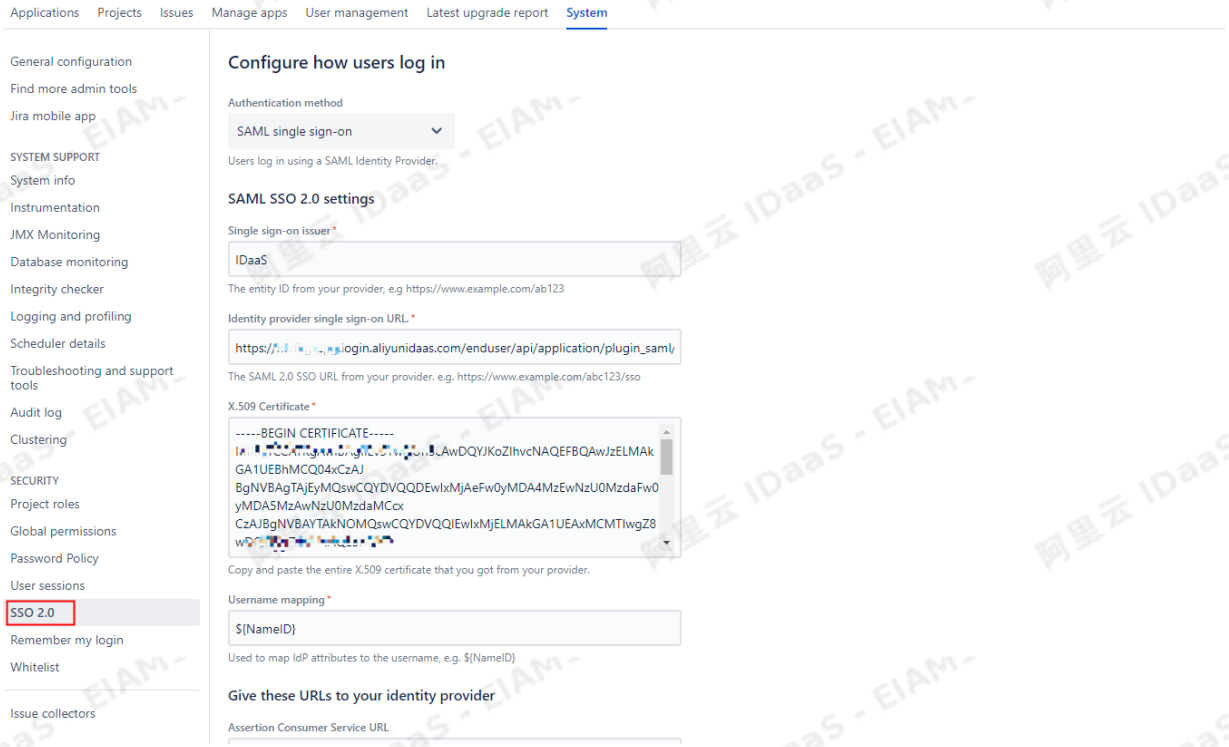
采用文本编辑器打开，获取到—-BEGIN CERTIFICATE—- —-END CERTIFICATE—-证书信息。 后面操作将用到该值



4. JIRA页面配置

访问system

选择SSO 2.0



参数说明

- Authentication method：选择SAML single sign-on
- Single sign-on issuer：建议统一写成 IDaaS
- X.509 Certificate：此处填步骤3中获取的证书信息。
- Identity provider single sign-on URL：格式为https://xxxxxx.login.aliyunidaas.com/enduser/api/application/plugin_saml/idaas-cn-shanghai-xxxxxx/sp_sso

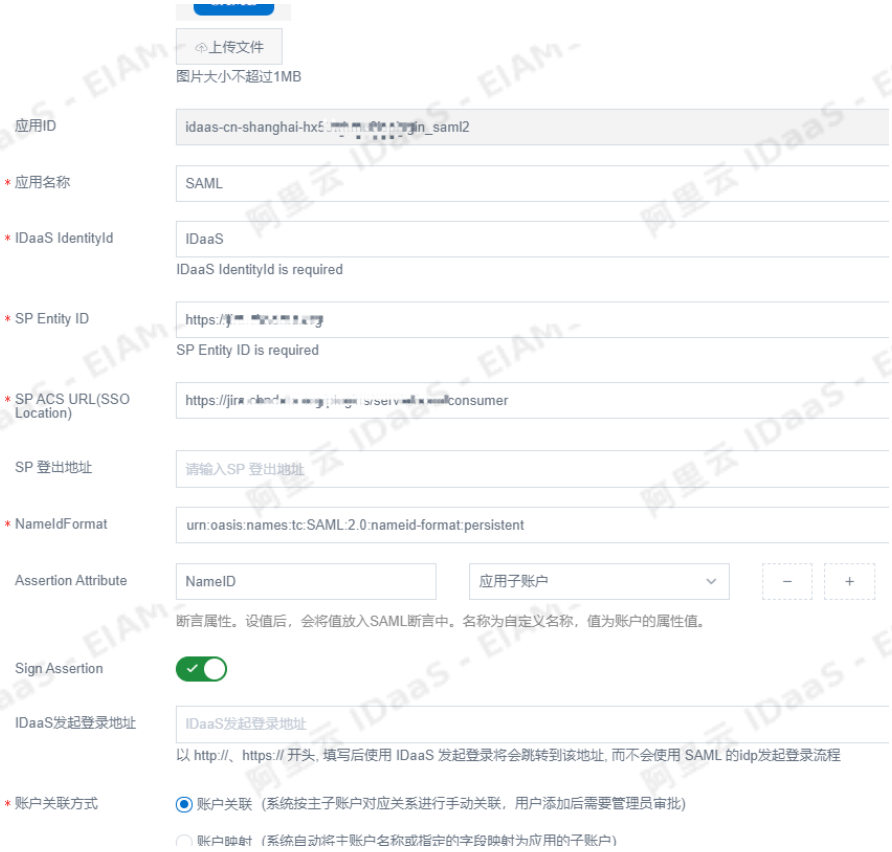
https://xxxxxx.login.aliyunidaas.com获取方式查看下图

idaas-cn-shanghai-xxxxxx获取方式查看下图

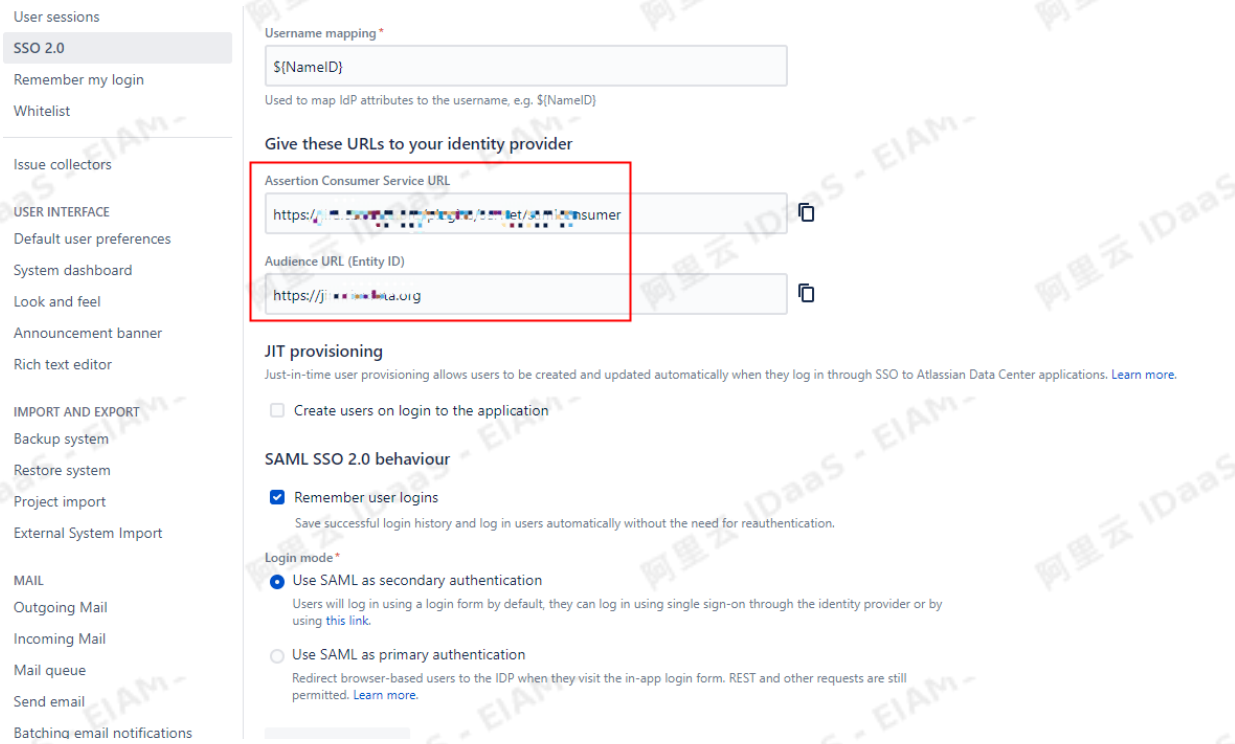
5. 返回IDaaS控制台，继续创建SMAL应用

选择SigningKey

填写应用信息

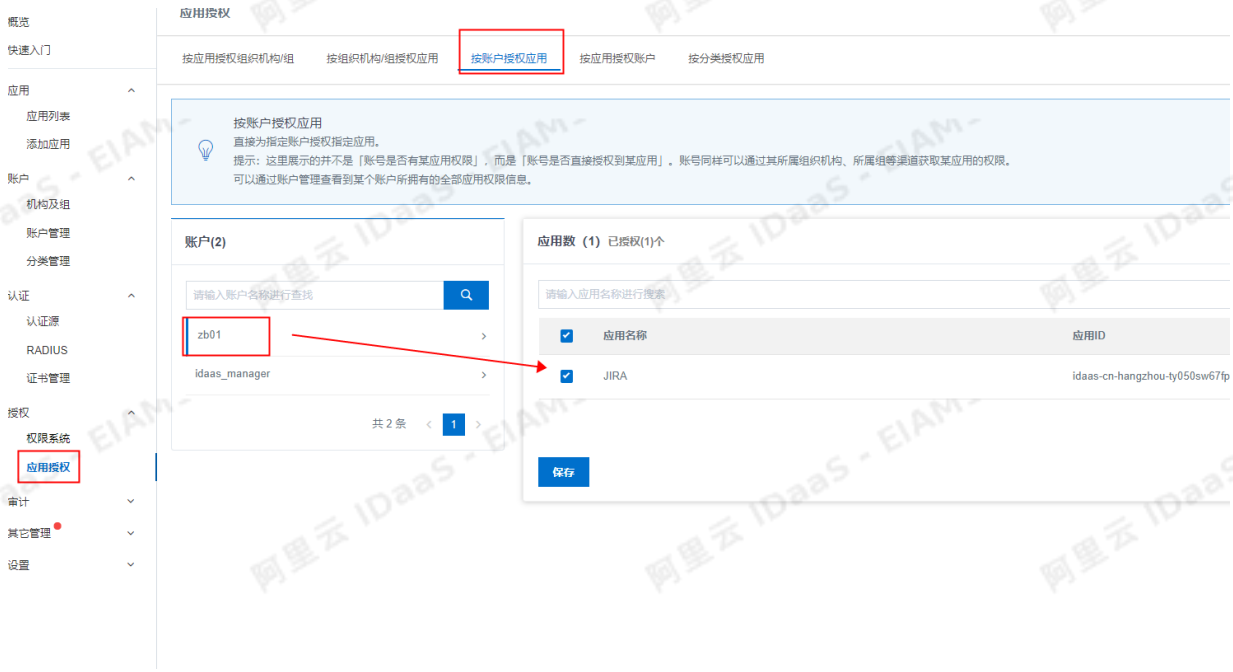


- 应用名称： 请填写一个应用名称；
- IDaaS IdentityId： 建议统一写成 IDaaS；
- SP Entity ID： 填写JIRA中的Audience URL (Entity ID)，见下图
- SP ACS URL(SSO Location)： 填写Jira中的Assertion Consumer Service URL，见下图
- NameIdFormat： 选择SMAL:2.0 nameid-format persistent
- Assertion Attribute：输入NameID, 选择应用子账户
- 开启Sign Assertion
- 选择一种账户关联方式进行提交，应用创建成功



在机构及组页面创建一个IDaaS账户

在应用授权页面，把应用授权给新创建的账户，并保存

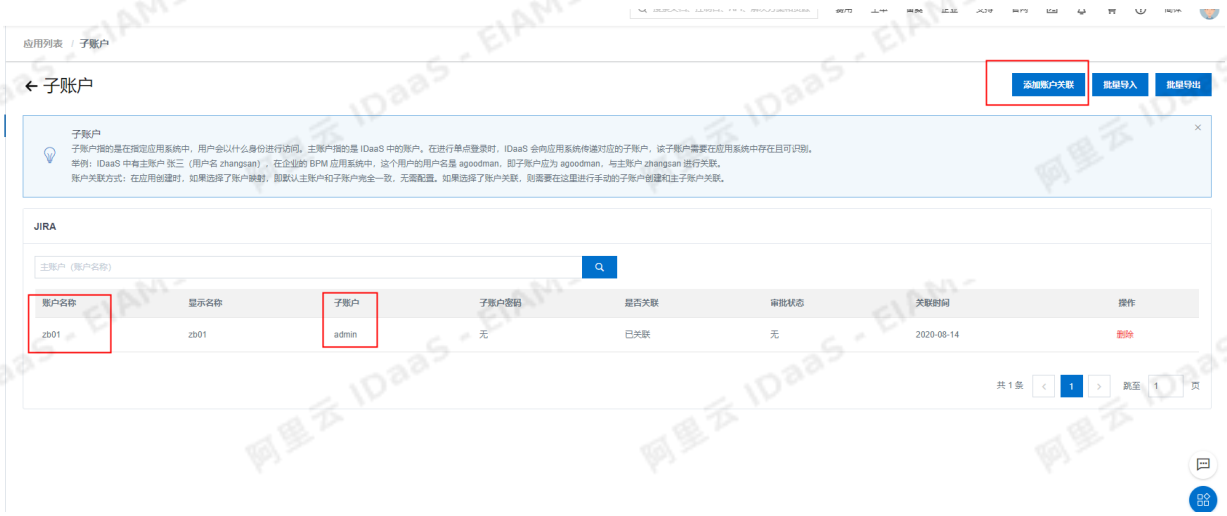


访问应用列表，点开应用详情，点击查看应用子账户

添加账户关联

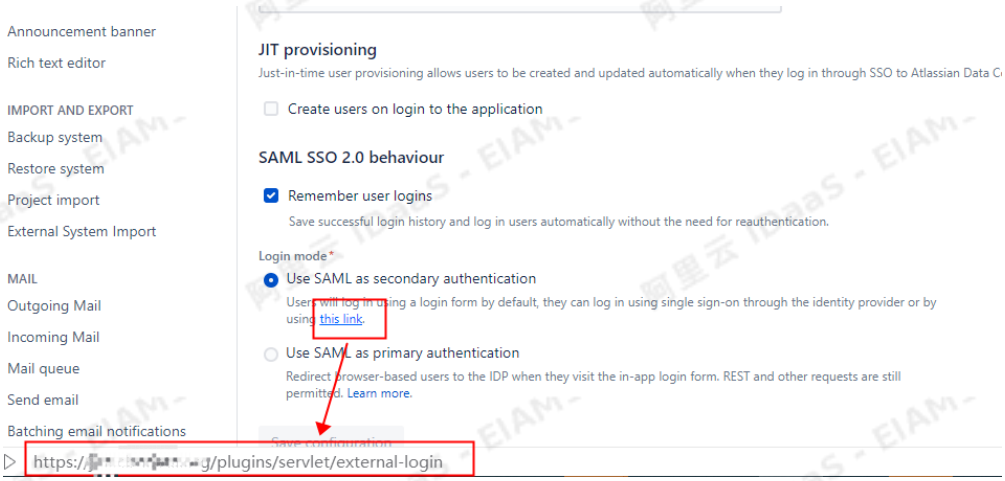
主账户：IDaaS中的账户，上面在账户及组页面创建的账户

子账户：JIRA 中的账户，如登录JIRA账户名是admin，子账户填写admin



6. 单点登录JIRA

复制jira中的link地址进行访问



输入上文中创建的IDaaS账户进行登录，登录成功后直接访问到Jira。

FAQ

是否支持通过IDaaS门户单点登录到Jira。

支持。具体信息请查看Jira/Confluence对接。

1.5. SAP GUI对接

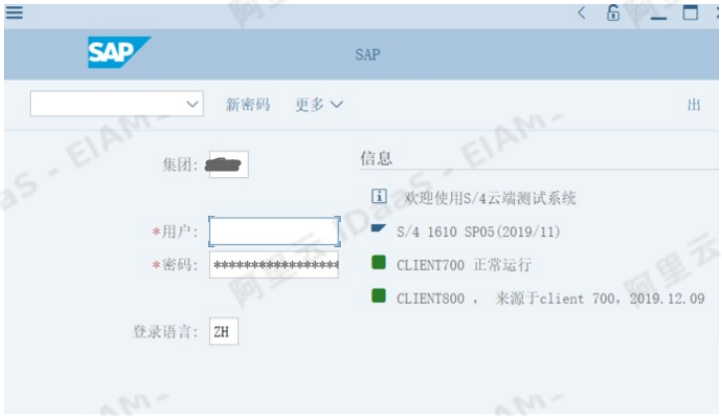
本文为您介绍如何配置实现SAP GUI的单点登录

操作步骤

1. 在SAP客户端添加新条目

应用服务器，实例编号，系统标识由客户提供。服务器版本：S/4 HANA 1610

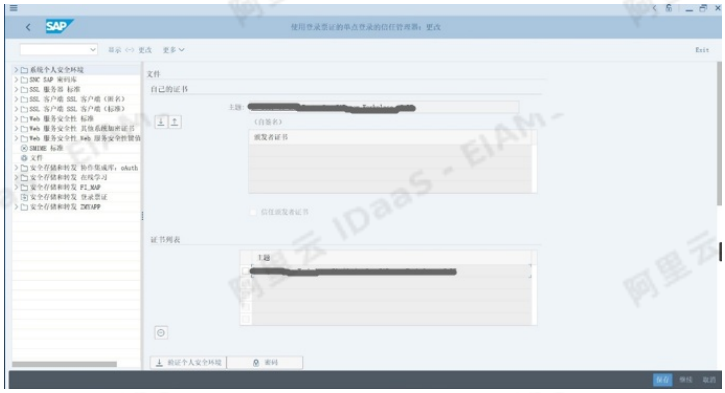
2. 登录SAP



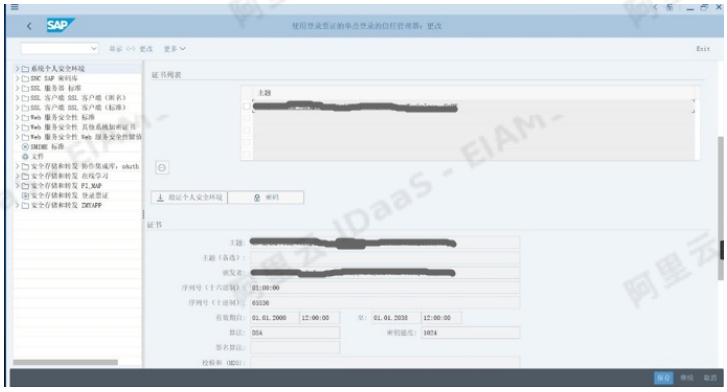
输入用户名密码进行登录。

3. 添加证书，左上角输入 STRUSTSSO2 回车

依次点击 “显示 ↔ 更改” → “更多” → “个人安全环境” → “导入”， 选择system.pse 文件, 完成后如图所示



双击证书列表中 “CN=mySAP.com Workplace CA (dsa), O=mySAP.com Workplace, C=DE”， 证书列表中证书将会在证书栏显示详细信息



点击证书下方添加到访问控制列表，系统标识输入“S4C”，集团输入“700”

依次点击 “更多” → “个人安全环境” → “另存为” → "系统个人安全环境" → “继续” → “是”

点击右上角 “Exit” → 询问是否保存更改时点击 “是”

4. 在IDaas中添加SAPGUI应用

配置名称	填写值	说明
authSchema	basicauthentication	固定填写该值, SAP未给出文档解释该值作用
mySysid	S4C	登录项中系统标识
myClient	700	登录时登录页显示的集团
extClient	700	登录时登录页显示的集团
extSysid	S4C	登录项中系统标识
host		应用服务器地址

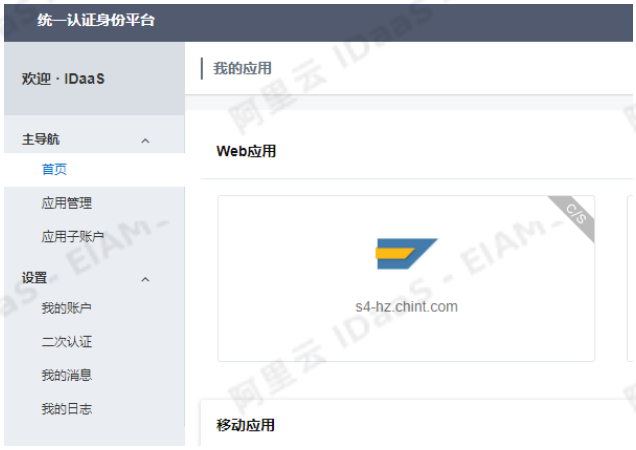
SAPRouter		使用到路由时填写该值，阿里给的环境中不需要设置该值
language	E	固定填写该值, SAP未给出文档解释该值作用
portalUser	PORTALUSER	固定填写该值, SAP未给出文档解释该值作用
pseKeyFile	选择证书system.pse	示例给出的 system.pse 由阿里提供；可使用 sapgenpse.exe 生成，尽量让客户提供
sapGuiLanguage	ZH	与SAP GUI登录时登录页下方登录语言一直
sysnr	00	实例ID

5. 给应用添加子账户并审批

i. 普通用户添加子账户：

ii. 管理员审批通过：

6. 用户进行SAP的单点登录，若提示没有安装插件，则安装插件，若有安装插件则可直接单点登录成功



1.6. OAuth2对接grafana最佳实践

概述

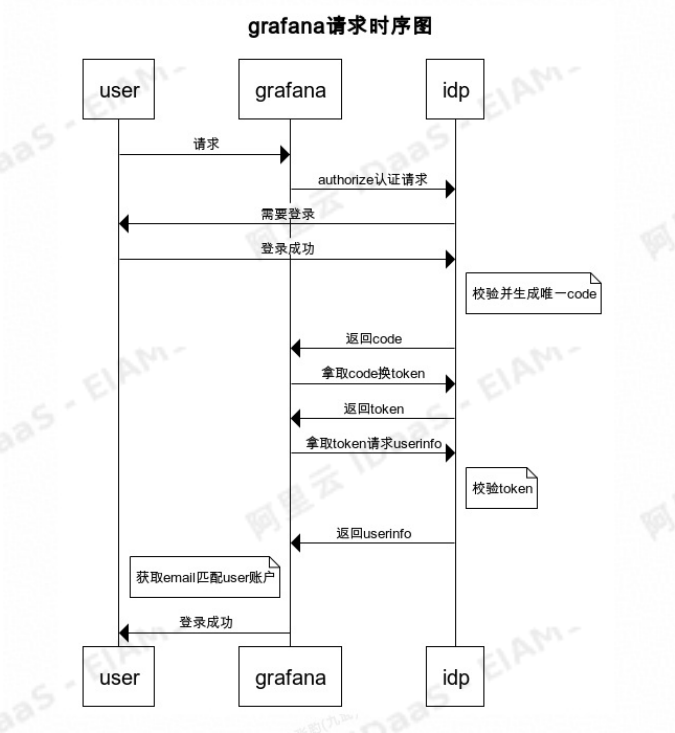
作为IDaaS平台，IDaaS支持基于标准OAuth2协议，实现从IDaaS到grafana单点登录

本文主要包含以下内容：

- 时序说明 - 时序图说明，以及交互
- 主要流程 - OAuth2配置grafana主要流程
- 操作步骤 - 详细配置说明
- FAQ - 常见问题以及其对策

时序说明

场景：用户从grafana发起单点登录时序



主要流程

Step1 创建OAuth2应用

Step2 授权OAuth2应用

Step3 子账户关系配置

Step4 获取应用信息

Step5 grafana配置

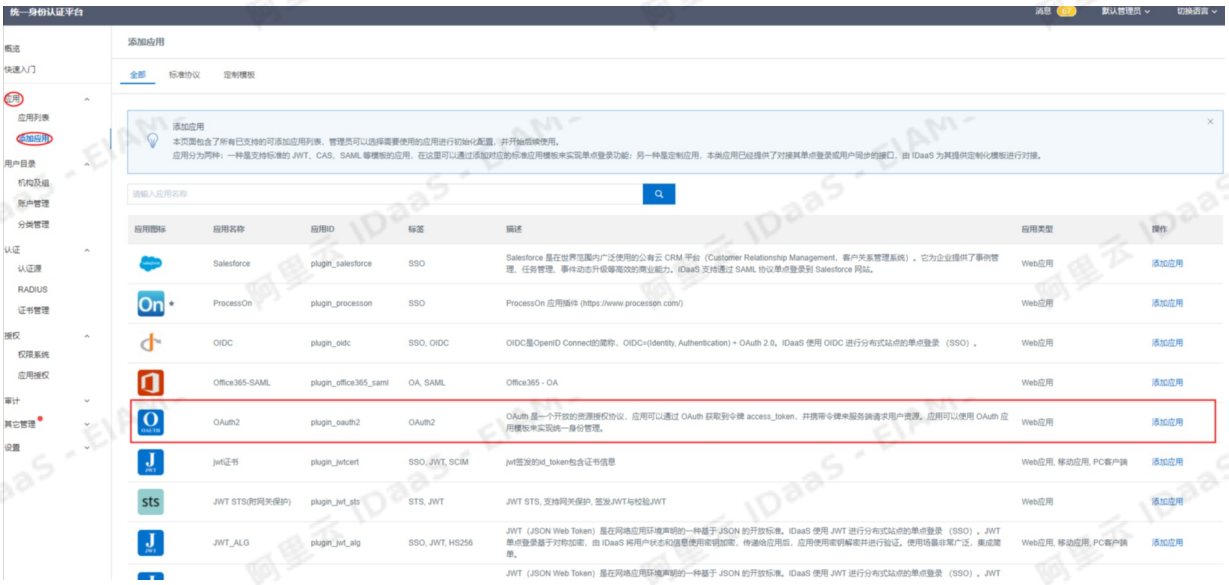
Step6 发起登录

操作步骤

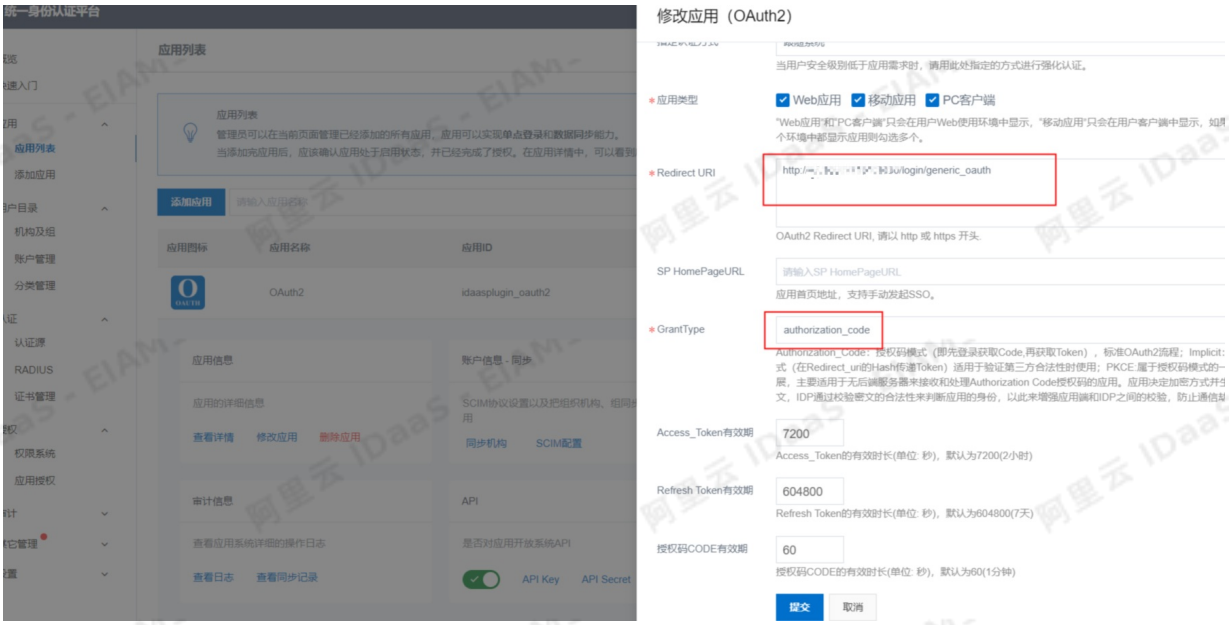
Step1 创建OAuth2应用：

1、首先以IT管理员账号登录云盾IDaaS管理平台。具体操作请参考 IT管理员指南-登录 。

2、点击左侧导航栏**应用>添加应用**选择右侧OAuth



3、选择OAuth2应用模板点击**添加应用**。



4、Redirect URI :http://{grafana domin}/login/generic_oauth{}替换成grafana的域名地址

GrantType：选择authorization_code

其他参数默认即可，有需要也可按照实际需要修改

Step2 OAuth2应用授权

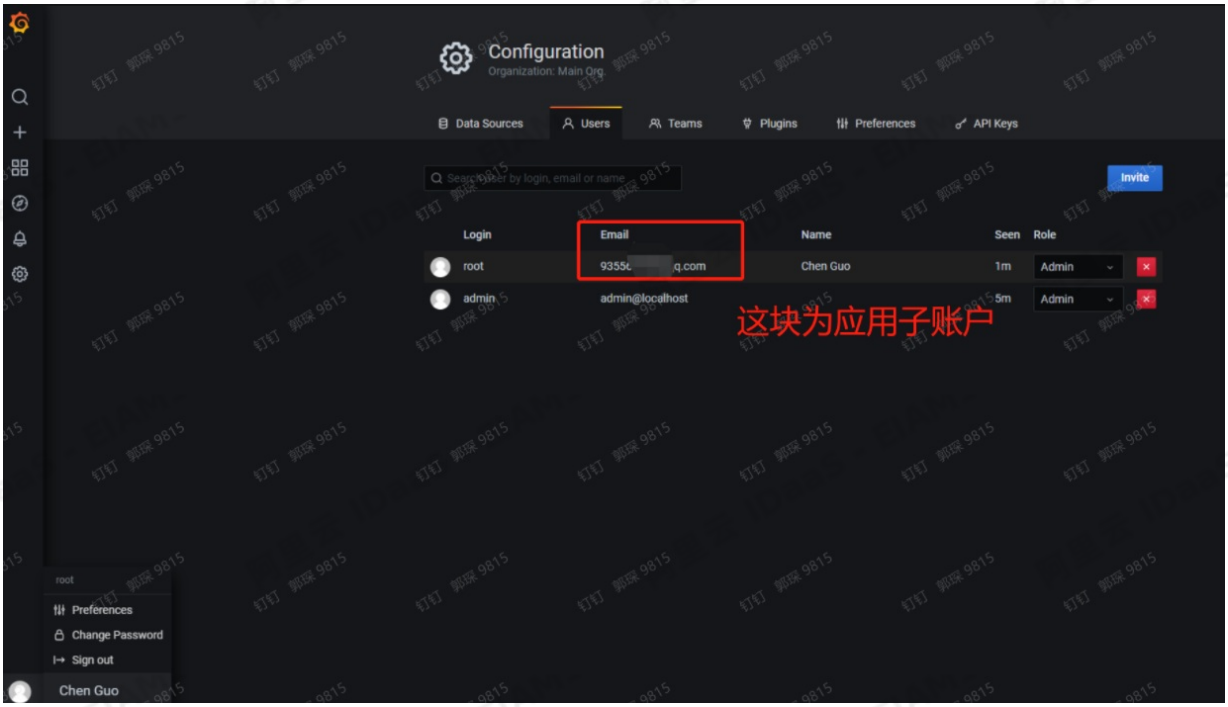
应用授权：选择应用（搜索应用）、选择组织机构（搜索组织机构）、勾选授权即可

Step3 子账户设置

点击左侧导航栏**应用>应用列表>应用子账户** 查看 OAuth2，添加子账户对应关系。

主账户添加当前已经授权应用的IDaaS账户，子账户对应grafana的邮箱用户，需要一一对应。

> 说明
>
> 子账户一定需要填写邮箱用户。



Step4 获取应用信息

点击左侧导航栏**应用>应用列表** 查看 OAuth2 应用详情, 获得Client Id、Client Secret、Authorize URL.

Step5 grafana配置

1.grafana配置文件增加generic_oauth配置

vim /etc/grafana/grafana.ini

#generic_oauth配置

[auth.generic_oauth]

enabled = true

#IDaaS应用Client Id

client_id =57064exxxxxxxxxx

#IDaaS应用Client Secret

client_secret =PTsclAxxxxxxxxxx

scopes = read

#IDaaS授权url，需要替换为自己的实际的IDaaS的域名

auth_url =http://xxxxxx.login.aliyunidaas.com/oauth/authorize

#IDaaS获取token url，需要替换为自己的实际的IDaaS的域名

token_url =http://xxxxxx.login.aliyunidaas.com/oauth/token

#IDaaS获取 user info url，需要替换为自己的实际的IDaaS的域名

api_url =http://xxxxxx.login.aliyunidaas.com/api/bff/v1.2/oauth2/userinfo

allow_sign_up = true

#获取邮箱节点JMEpath

email_attribute_path=data.email

[server]

#这里的root_url需要修改为真实的地址，因为授权回调的redirect_uri使用该地址

root_url=http://xxxxxxx:3000

2. 重启grafana

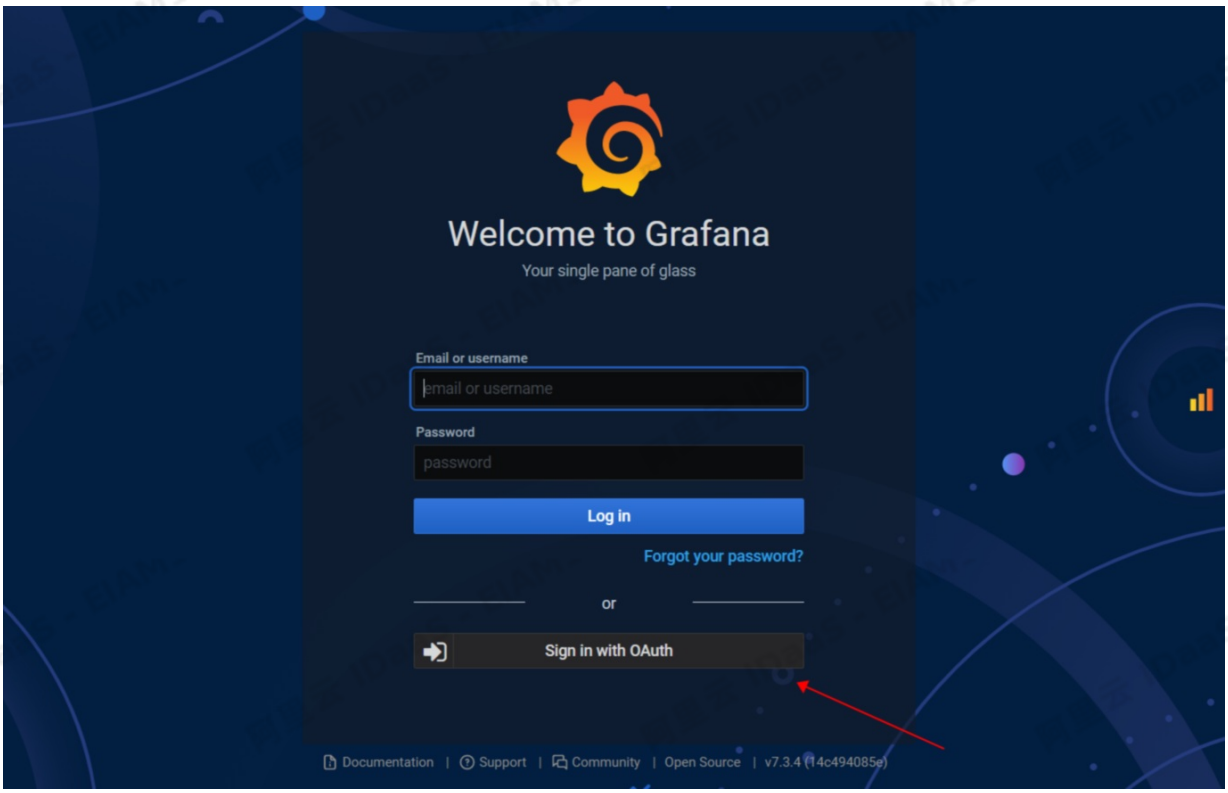
示例：/etc/init.d/grafana-server restart

根据自己实际安装路径来

Step6 grafana发起OAuth登录

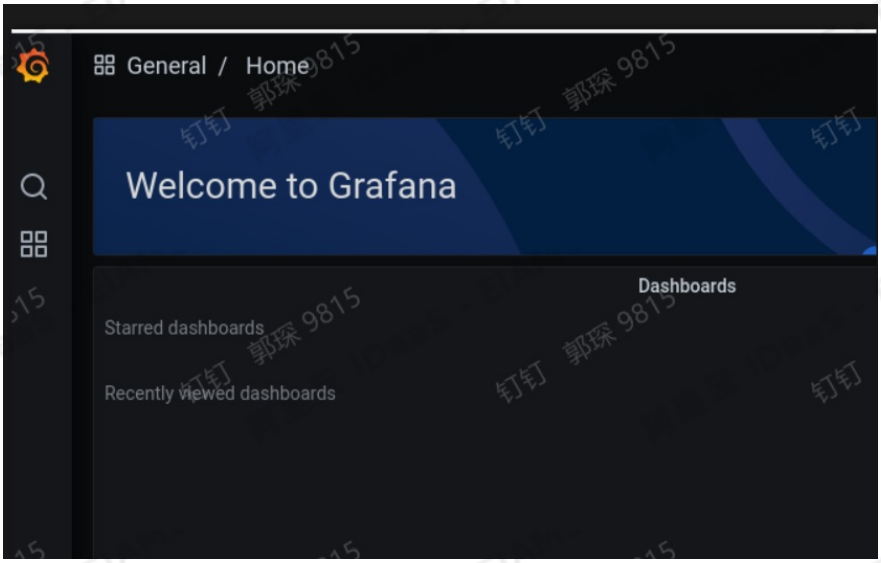
> 说明
>
> grafana使用OAuth2对接，只支持**grafana页面发起登录**，不支持 IDaaS发起

在grafana登录页面点击“Sing in with OAuth”发起认证登录



输入IDaas账号进行登录

认证成功



FAQ

1. 代理模式下报错如下

OAuth认证错误:
error="invalid_grant", error_description="Invalid redirect:http://localhost:3000/login/generic_oauthdoes not match one of the registered values: [http(https)://域名/login/generic_oauth]"。

代理模式（nginx或者apache），需要在grafana.ini中修改 redirect URI。因为代理模式下，不认识CALLBACK

Redirect URI :http://{grafana domin}/login/generic_oauth{}替换成grafana的域名地址

2. 子账户对应关系是否需要一对一

是的。

3. Grafana 支持的版本

需要grafana7.2+版本才能兼容此OAuth2的配置。

1.7. Jenkins对接（SAML）

通过 IDaaS 提供的单点登录能力，快速实现Jenkins 单点登录的目的。

操作步骤

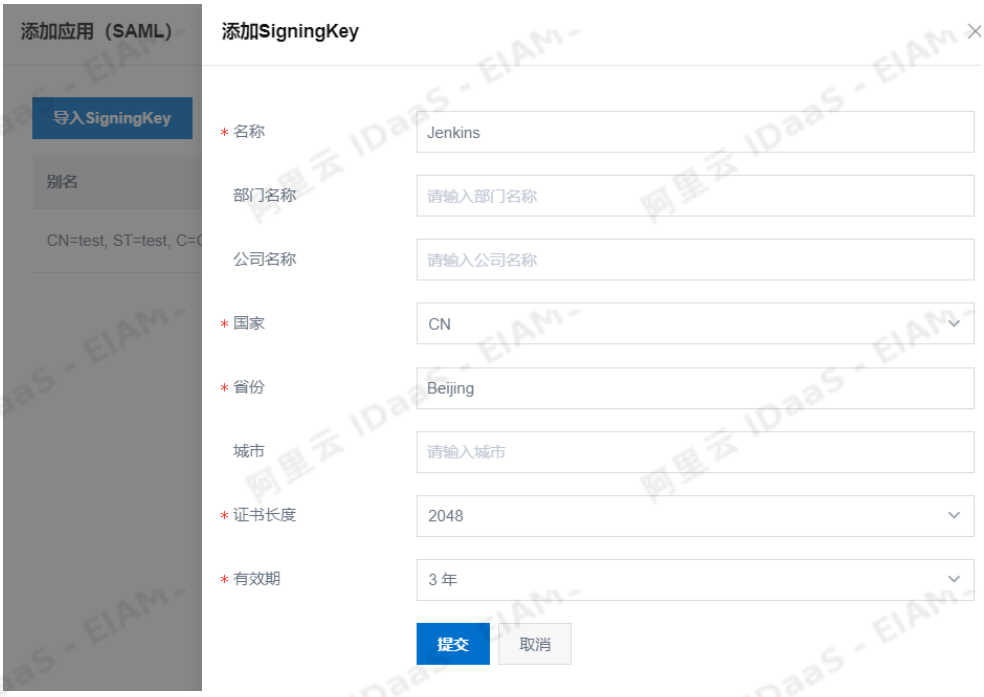
1、在 Jenkins 插件管理中安装 saml 插件。

2、在 Jenkins 中进入“Configure Global Security”，在“Authentication”中选择“SAML 2.0”。

3、以IT管理员账号登录云盾IDaaS管理平台。具体操作请参考 IT管理员指南-登录。

4、在云盾IDaaS管理平台中左侧菜单中点击添加应用，找到“SAML”应用，点击“添加应用”。

5、在证书界面点击“添加SigningKey”。在名称中输入一个便于标识的证书名称，如“Jenkins”；国家选择“CN”；省份任意填写，如“Beijing”；证书长度选择“2048”；有效期选择“3年”。



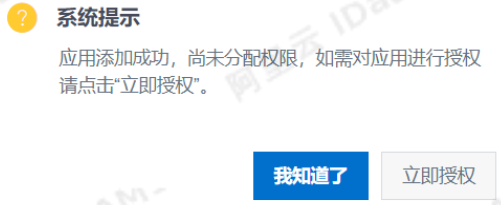
6、添加完成后会自动回到证书列表界面，在刚才添加的证书右边，点击“选择”。

7、添加应用页面参数

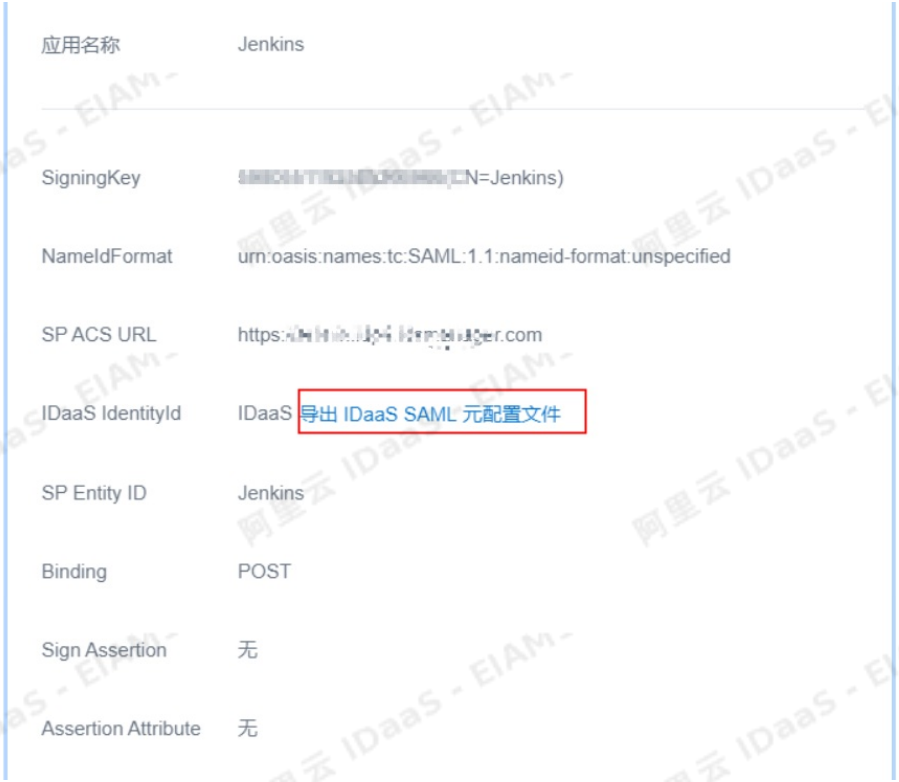
- 应用名称填写便于识别的名称，如“Jenkins”。
- IDaaS IdentityId 填写任意文字，如“IDaaS”。
- SP Entity ID 填写任意文字，如“Jenkins”；SP ACS URL(SSO Location) 填写任意地址；NameIdFormat 选择第一个，即“urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified”。
- 若Jenkins中员工用户名与IDaaS系统中员工用户名一致，则选择账户映射，否则选择账户关联。

单击“提交”添加成功。

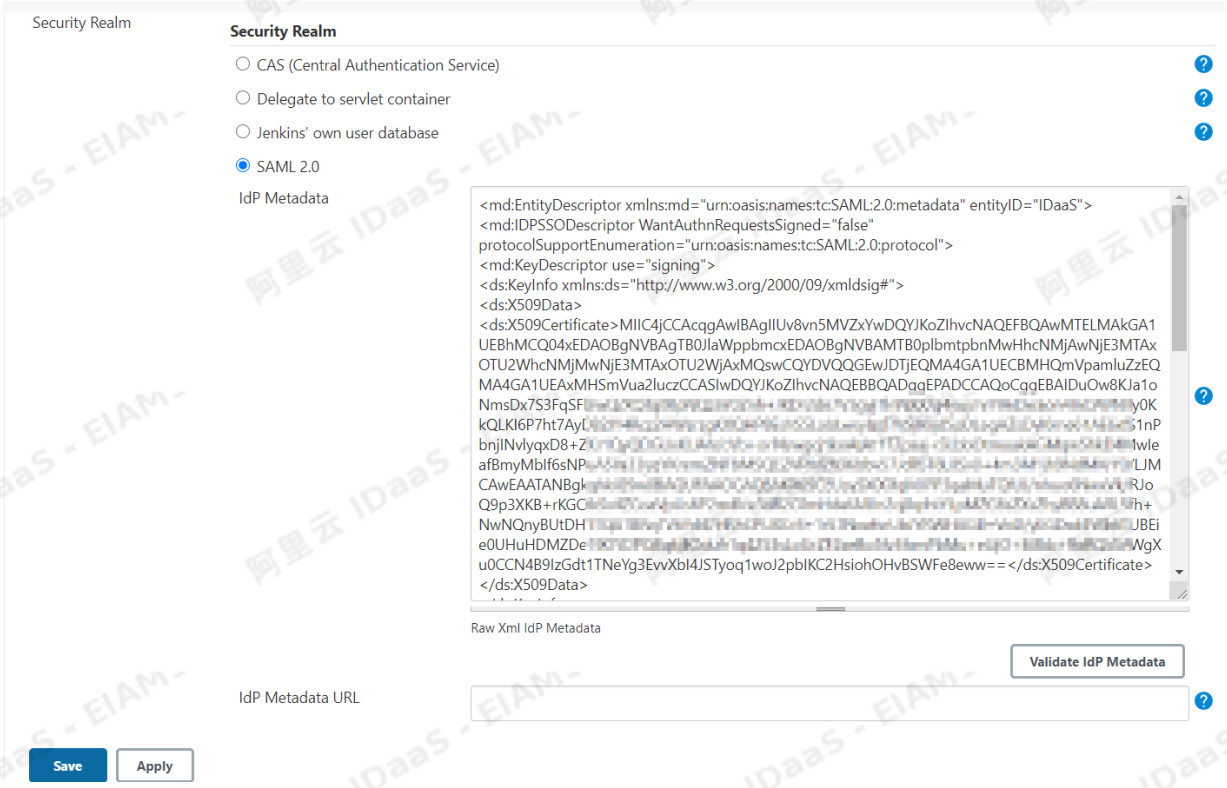
8、添加完成后将会提示对应用进行授权。点击“立即授权”，进入应用授权界面，可根据需求按组织机构、账户等对应用进行授权，授权后的组织机构/账户才能够登录该应用。勾选需要授权的组织机构/账户后，在页面最下方点击保存，并在弹出的确认窗口确认授权。



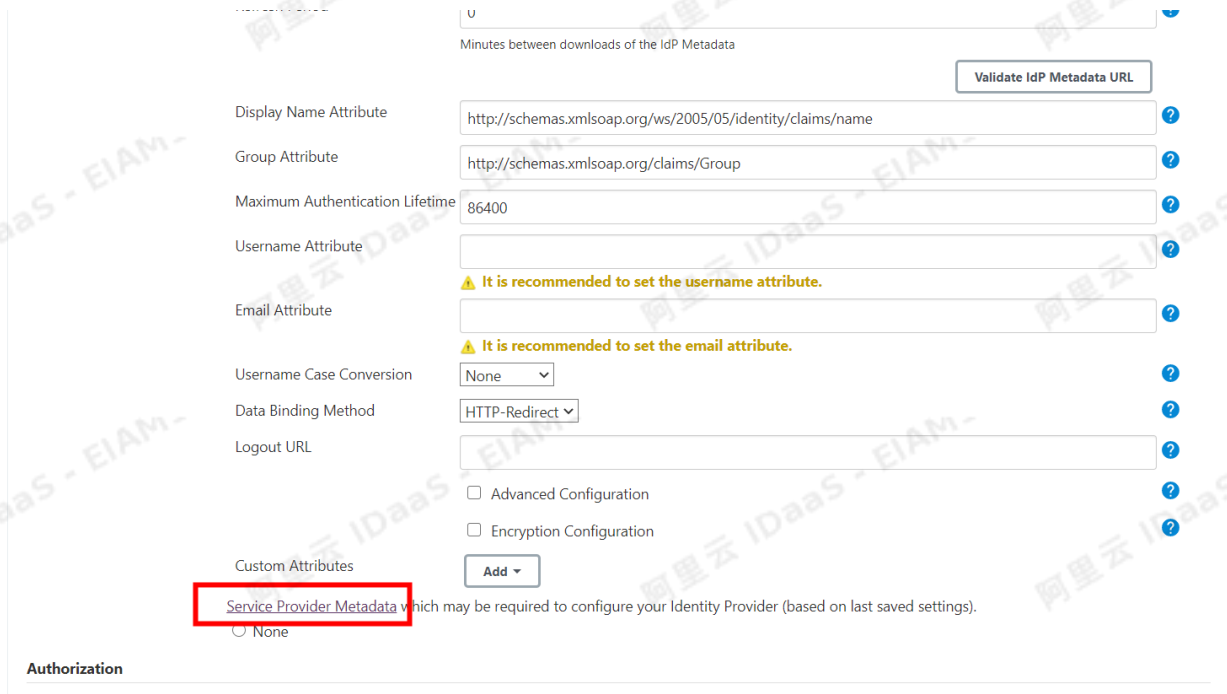
9、在IDaaS中进入应用列表页面，可看到刚才添加的应用。点击右侧详情，点击查看详情，可查看需要在Jenkins中配置的信息，点击“导出 IDaaS SAML 元配置文件”下载 IDaaS 元配置文件，使用记事本打开该文件，复制所有内容待用。



10、使用具有管理员权限的用户登录到 Jenkins 系统中，依次点击“Manage Jenkins”、“Configure Global Security”，进入“Configure Global Security”。在“Security Realm”中，选择“SAML 2.0”，在“IdP Metadata”中粘贴第 9 步复制的 IDaaS 元配置文件内容，点击“Save”。

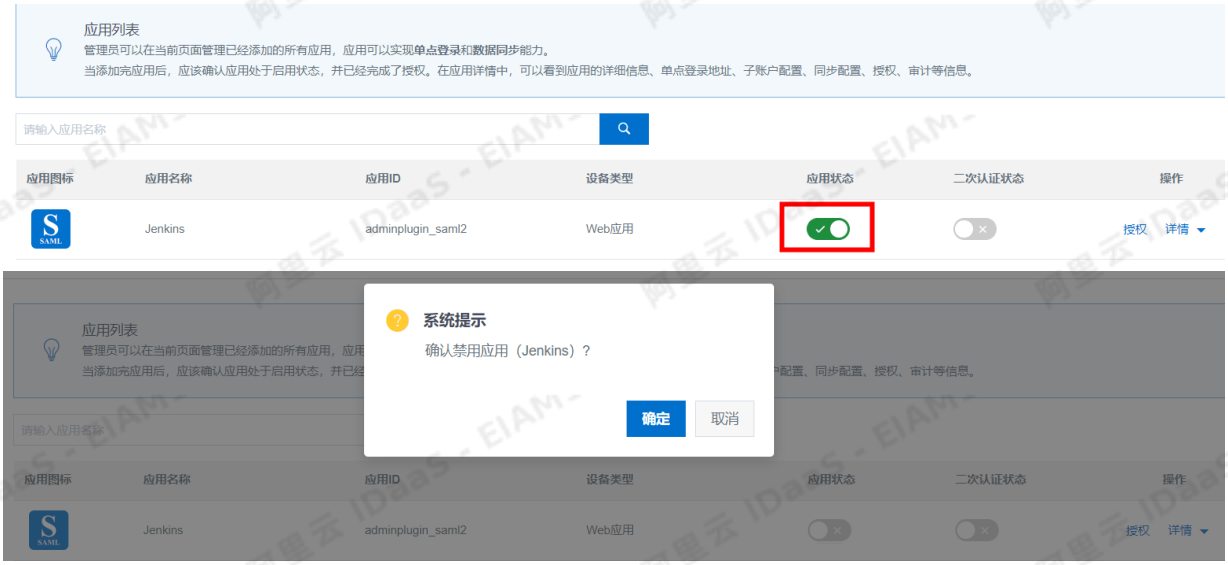


11、重新进入“Configure Global Security”。点击“SAML 2.0”下方“Service Provider Metadata”，打开 Jenkins SAML元配置文件。



12、在“Service Provider Metadata”数据中找到“entityID”，将后面引号中的值，如“http://11.167.XX.XX:808/securityRealm/finishLogin”，复制待用；
在“Service Provider Metadata”数据中找到“AssertionConsumerService”，将后面“Location=”后引号中的值，
如“http://11.167.XX.XX:8080/securityRealm/finishLogin”，复制待用。

13、在 IDaaS 中进入应用列表页面，找到添加的应用。点击应用状态下方按钮，禁用应用，在弹出的确认框中点击“确定”。



14、点击应用右侧详情按钮，点击“修改应用”。

15、在“SP Entity ID”中填写第12步复制的“entityID”的值；在“SP ACS URL(SSO Location)”中填写第 12 步复制的“Location=”的值，点击提交。

16、回到应用列表界面，点击应用状态下方按钮，启用应用。

17、使用新的浏览器打开 Jenkins 地址将会跳转到 IDaaS 进行登录，在 IDaaS 登录成功后，会跳转回 Jenkins。

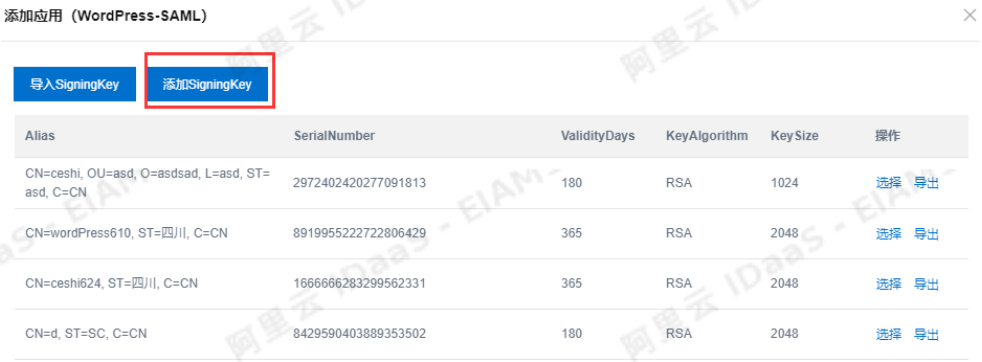
1.8. WordPress对接

一、WordPress-SAML应用

WordPress-SAML应用主要实现支撑单点登录流程的功能。

操作步骤：

1. 以IT管理员账号登录云盾IDaaS管理平台。具体操作请参考 IT管理员指南-登录。
2. 点击左侧导航 **应用 > 添加应用**，选择WordPress-SAML应用模板点击**添加应用**.。
3. 点击 **添加SigningKey**。



4. 点击 **选择** 进入添加应用页面

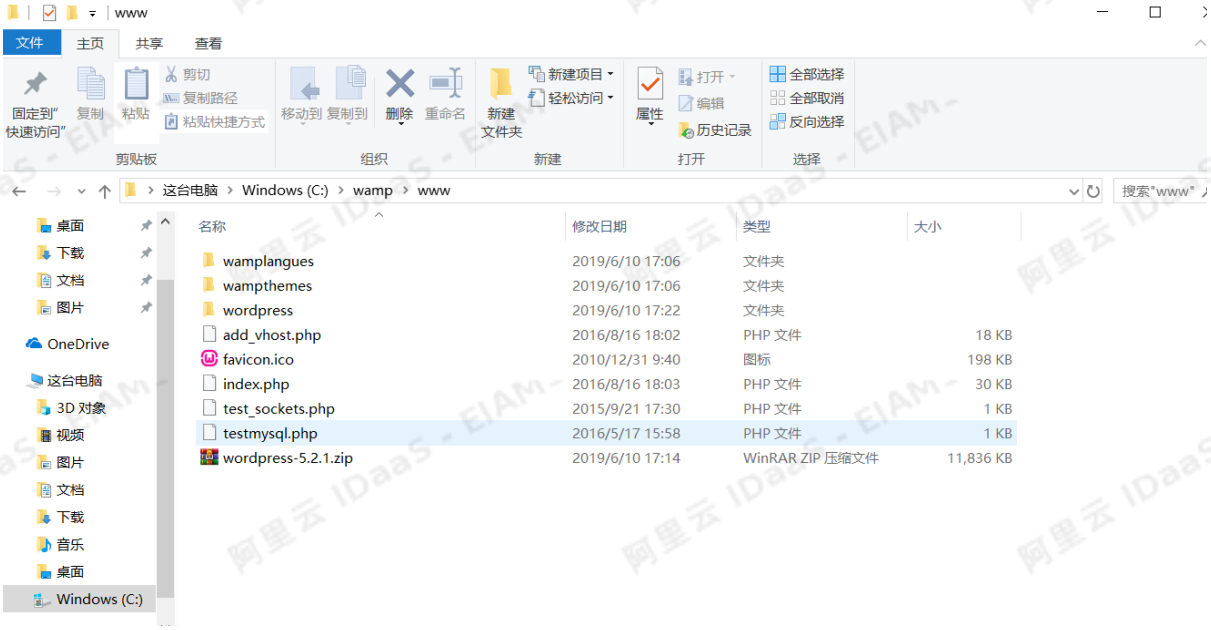
说明

其中IDaaS Identity Id任意填写，填写后的任意值同步到WordPress里，SP Entity ID、SP ACS URL、Name Id Format所对应填写参数的值要在WordPress里面获取；

二、WordPress-SAML配置流程

WordPress-SAML在php环境中运行的，因此需要搭建一套php的环境，操作步骤如下：

1. 官网下载WampServer并进行解压；
2. 官网下载WordPress解压后放置WampServe的www目录下

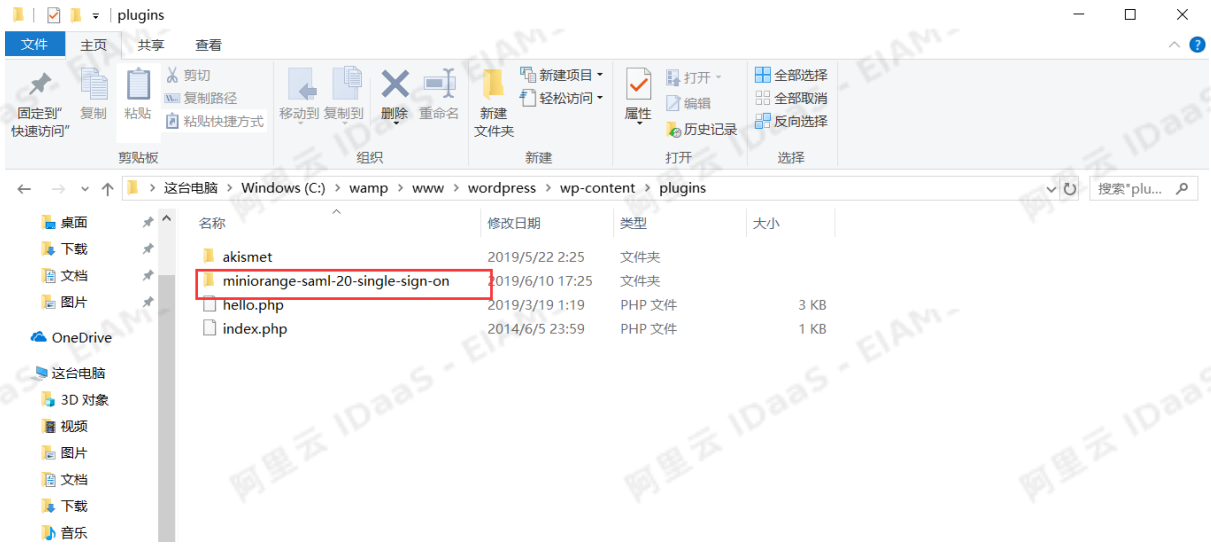


3. 在MySQL中创建一个WordPress账户；

4. 在WordPress欢迎页面，设置用户名和密码

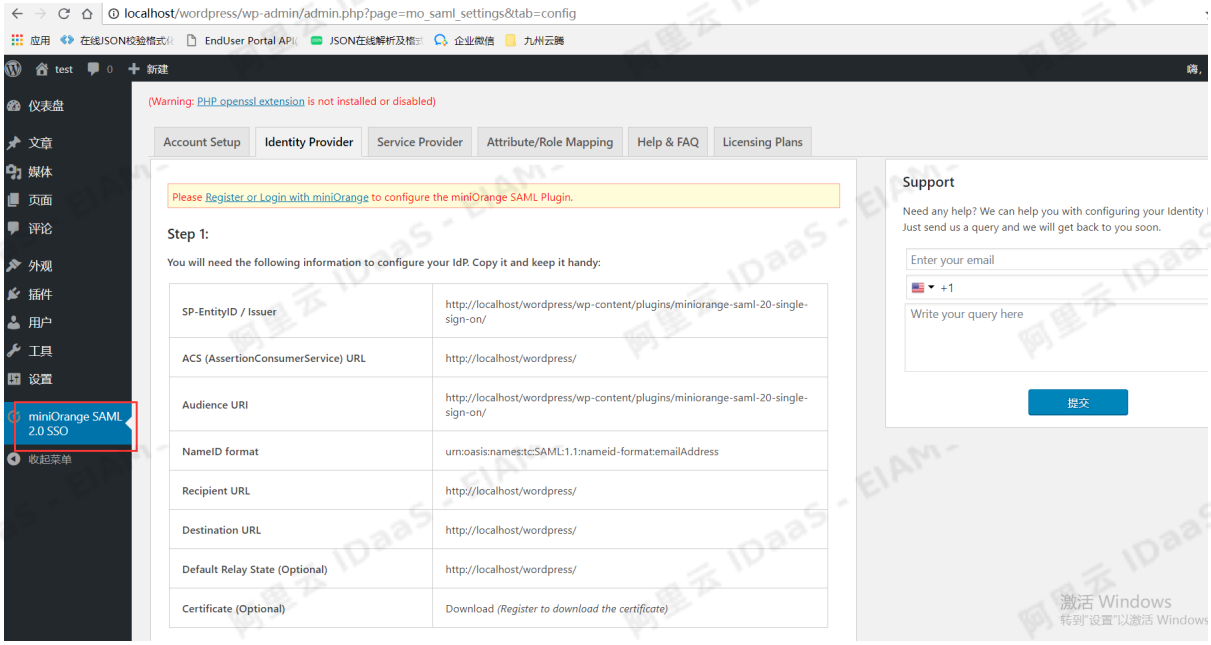
5. 点击左下角“安装WordPress”按钮；

6. 下载miniorange-saml-20-single-sign-on.4.8.23，将其解压后放置C:\wamp\www\WordPress\wp-content\plugins路径下面



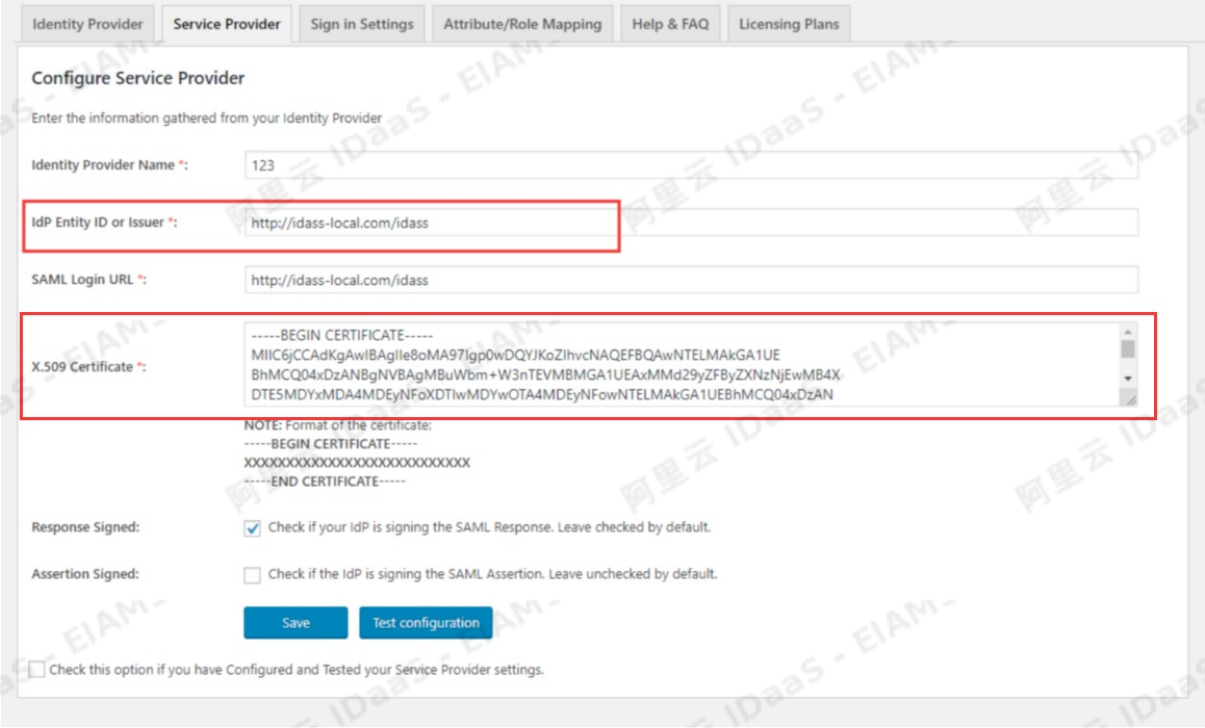
7. 重启php环境“restart all services”,然后安装miniorange并启用，首先登录WordPress进入页面点击“插件”进入页面，点击miniorange下面的“启用”按钮，然后刷新一下页面。

8. 页面会弹出miniorange saml选项



9. 将Identity Provider页面中SP-EntityID/Issuer、ACS（Assertion Consumer Service）URL、Name ID format的值填写到IDP4“添加WordPress-SAML”页面

0. 其中IDaaS Identity Id的值任意填写，填写后的值同步到WordPress的Service Provider页面进行对应，最后将IDP4中的cer文件导入到WordPress中的X.509 Certificate中



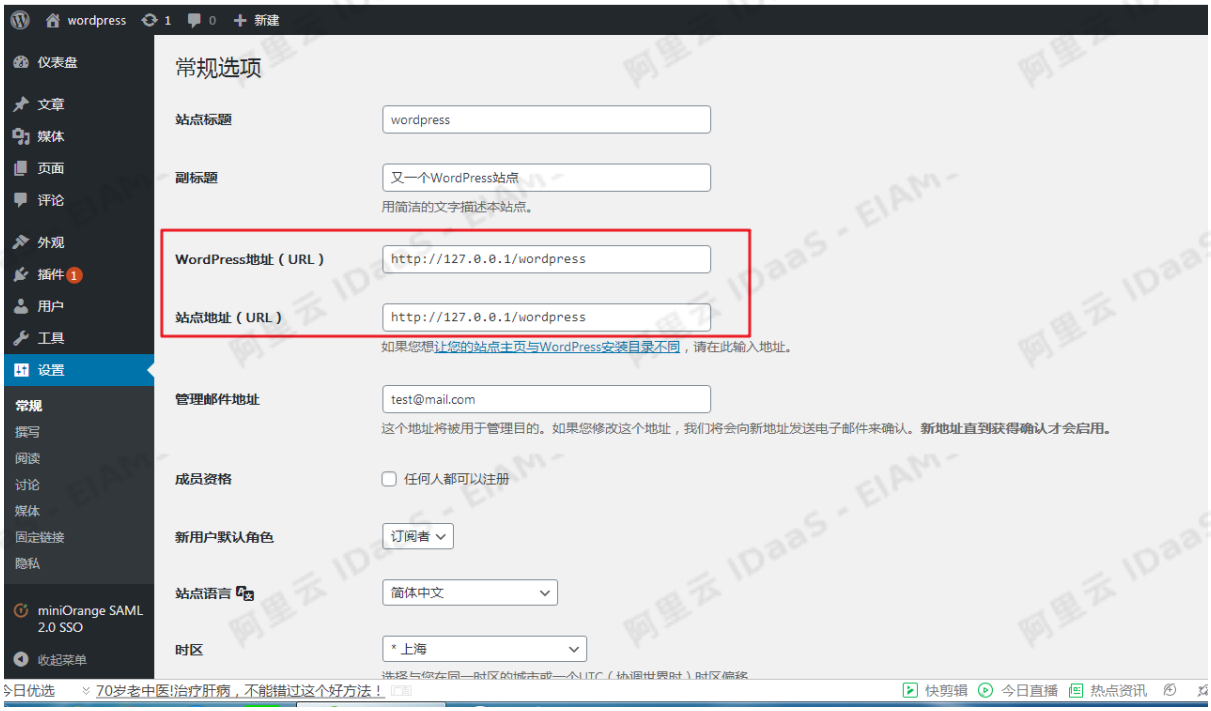
1. 配置完成之后,选择WordPress-SAML应用添加应用子账户，子账户是WordPress中账号的邮箱，即可单点登录WordPress。

补充

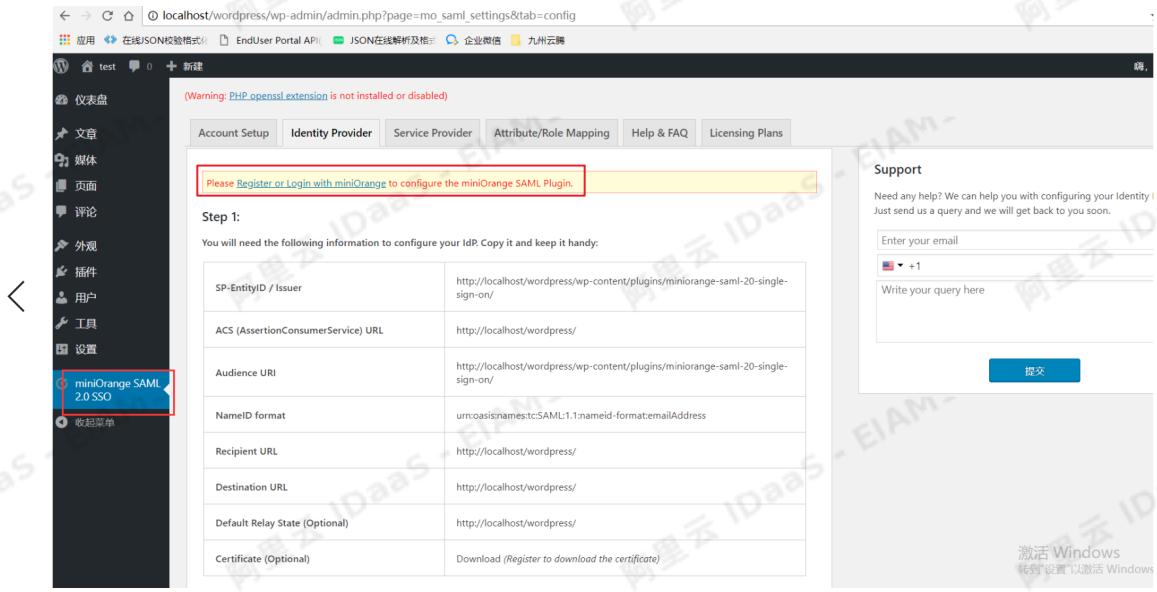
1. 本地部署的wordpress的URL地址为127.0.0.1或localhost，可以在**设置**中修改wordpress的url地址。

You will need the following information to configure your IdP. Copy it and keep it handy:

SP-EntityID / Issuer	http://127.0.0.1/wordpress/wp-content/plugins/miniorange-saml-20-single-sign-on/
ACS (AssertionConsumerService) URL	http://127.0.0.1/wordpress/
Audience URI	http://127.0.0.1/wordpress/wp-content/plugins/miniorange-saml-20-single-sign-on/
NameID format	urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
Recipient URL	http://127.0.0.1/wordpress/
Destination URL	http://127.0.0.1/wordpress/
Default Relay State (Optional)	Available in the **premium** version
Certificate (Optional)	Available in the **premium** version



2. miniOrange若不允许修改配置，原因是用户没有登录miniOrange，点击下图红框中的超链接进行登录。

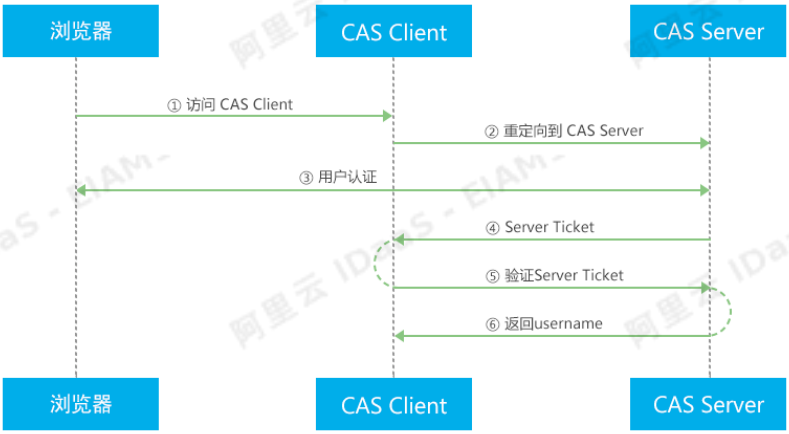


1.9. Jumpserver对接-CAS协议

原理和协议

从结构上看，CAS 包含两个部分： **CAS Server** 和 **CAS Client** 。CAS Server 需要独立部署，主要负责对用户的认证工作； CAS Client 负责处理对客户端受保护资源的访问请求，需要登录时，重定向到 CAS Server。

下图是标准 CAS 基本的请求过程：



CAS Client 与受保护的客户端应用部署在一起，以 Filter 方式保护受保护的资源。对于访问受保护资源的每个 Web 请求， CAS Client 会分析该请求的 HTTP 请求中是否包含 Service Ticket。如果没有，则说明当前用户尚未登录，于是将请求重定向到指定好的 CAS Server 登录地址，并传递 Service （也就是要访问的目的资源地址），以便登录成功过后转回该地址。

用户在上图流程中的**第 3 步**输入认证信息，如果登录成功，CAS Server 随机产生一个相当长度、唯一、不可伪造的 Service Ticket，并缓存以待将来验证。之后系统自动重定向到 Service 所在地址，并为客户端浏览器设置一个 Ticket Granted Cookie（TGC），CAS Client 在拿到 Service 和新产生的 Ticket 过后，在第 5，6 步中与 CAS Server 进行身份核实，以确保 Service Ticket 的合法性。

在 IDaaS 中， CAS （标准）应用模板实现了标准的 CAS 流程。它充当一个 CAS Server的角色。当 CAS Cient 决定使用IDaaS作为 CAS Server 时。在登录认证时需要使用 IDaaS 系统中公司的主账号，密码进行认证。

操作步骤

说明 CAS 标准应用目前只能由 IT 管理员在应用添加菜单中添加，下面是 IT 管理员的应用添加流程配置说明。

1. 以IT管理员身份登录 IDaaS，点击添加应用。找到 CAS （标准），点击添加应用

2. 配置CAS 应用CAS Client 也就是业务系统需要提供的两个参数：

ServiceNames：CAS客户端发起认证的URL地址，一般使用固定格式：jumpserver登录地址+/core/auth/cas/login/?next=%2F

TargetUrl：IDaaS发起单点登录地址一般格式为：jumpserver登录地址+/core/auth/cas/login/

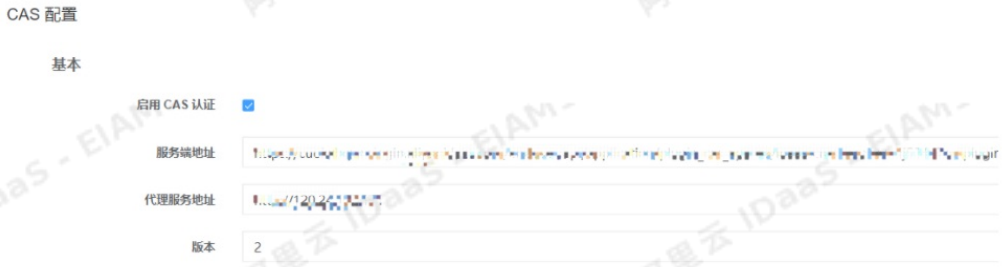
账户关联方式：根据实际情况选择，请查看主子账户介绍。

3. 点开应用详情，复制 **CAS Server URL Prefix**参数以便接下来修改 JumpServer 配置文件时使用。

3. 修改jumpserver配置文件

```
##是否启用CAS认证
AUTH_CAS=True
##CAS客户端认证地址对应之前复制的CAS Server URL Prefix参数
CAS_SERVER_URL=https://xxxxxx.login.aliyunidaas.com/enduser/api/application/plugin_cas_apereo/xxxxxxxxxxxxxxplugin_cas_apereo/
##Jumpserver登录地址
CAS_ROOT_PROXIED_AS=https://jumpserver.com
##要使用的CAS协议版本
CAS_VERSION=3
```

或者在jumpserver设置页面找到CAS配置（根据版本不同配置方式也会有不同，请以实际使用的jumpserver版本为准），地址同上



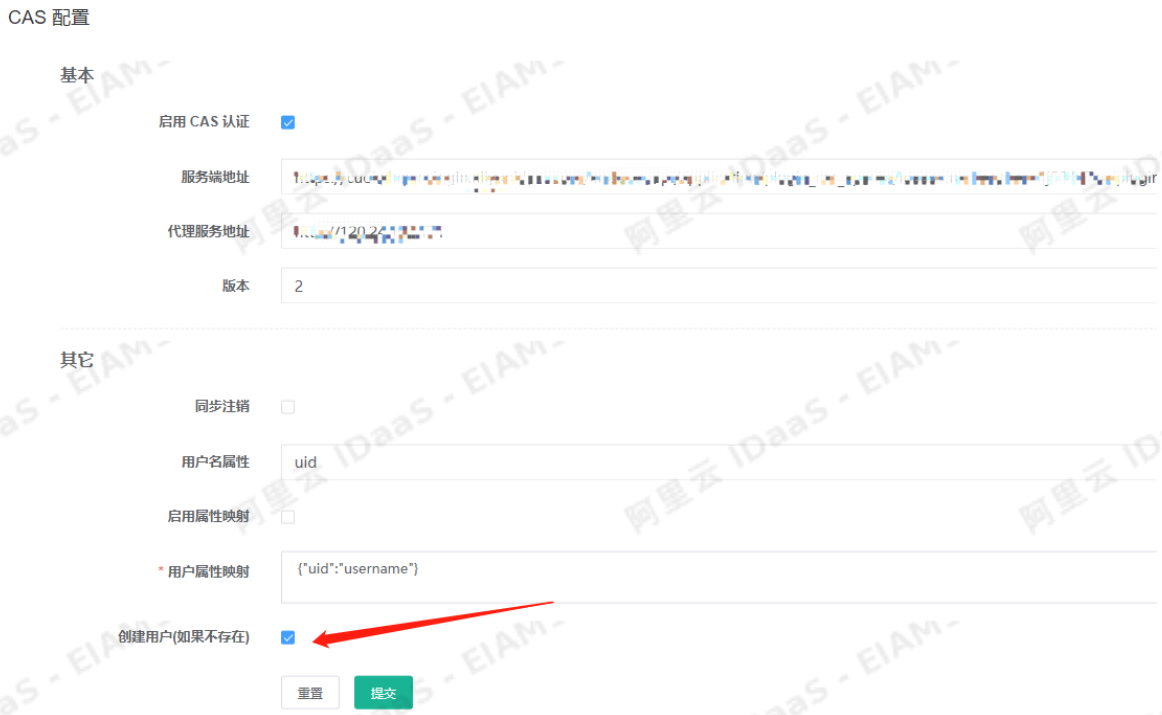
5.主子账户绑定，详情可参考主子账户介绍。

完成以上步骤，就可以使用CAS协议单点登录到Jumpserver.

FAQ

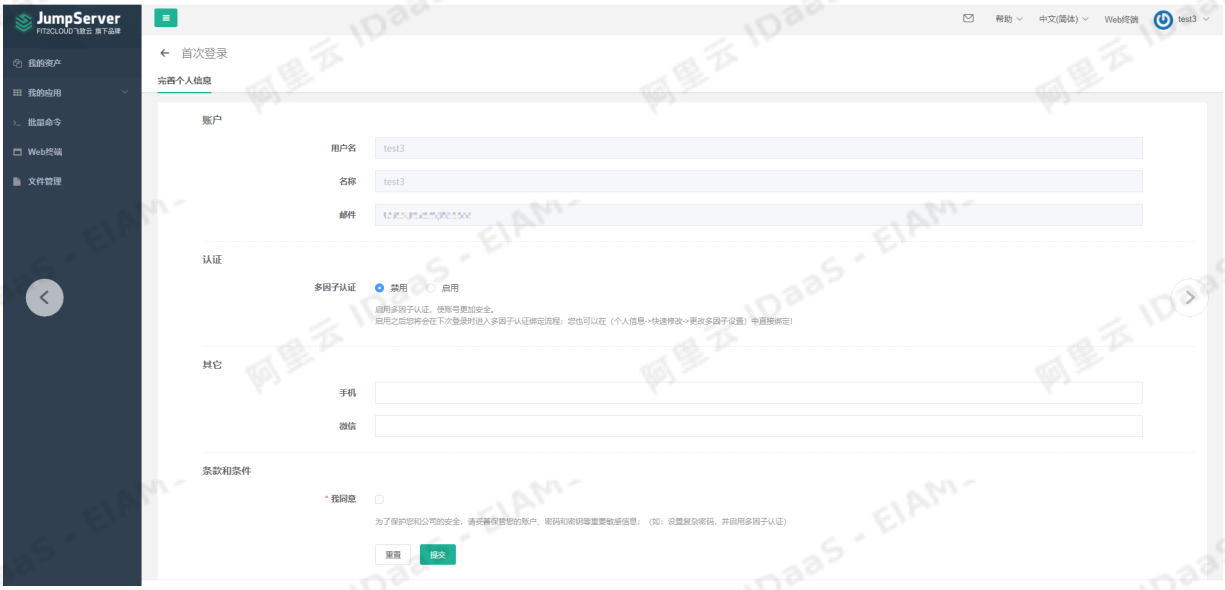
如果jumpserver这边没有账户是否可以登录后自动创建账户？

在jumpserver配置中开启该选项



IDaaS登录时如果未绑定子账户，将在登录时提示绑定子账户，如绑定的账户在jumpserver中存在则在审核通过后重新登录至该账户。

如绑定的子账户名在jumpserver中无匹配项，则会创建账户跳转至该创建用户页面。

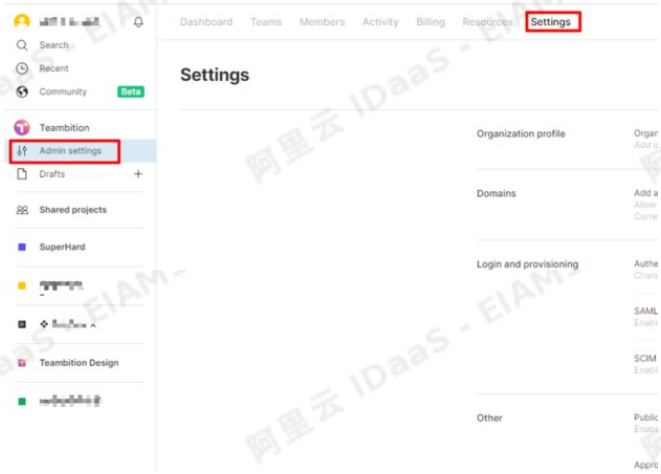


1.10. IDaaS对接Figma实践

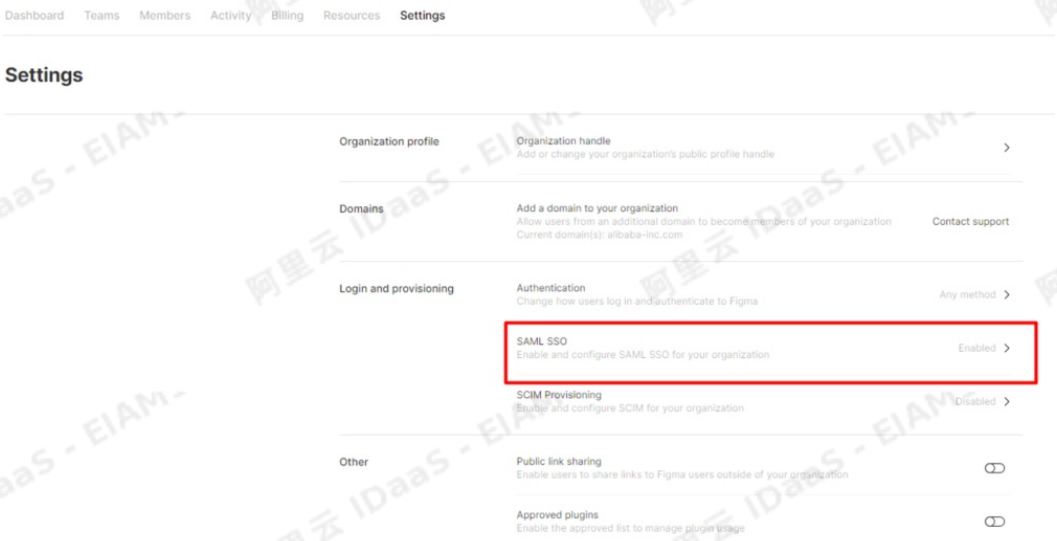
本文是Figma对接文档， 配置SAML SSO 进行单点登录。

一、获取Figma中SAML SSO元数据信息

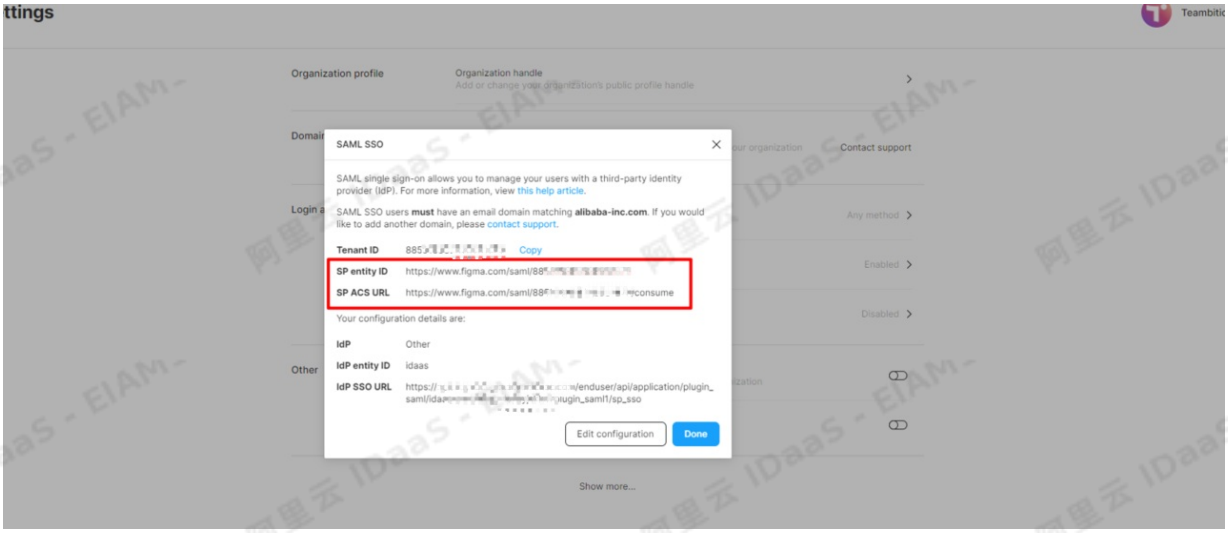
管理员登录Figma控制台，左侧菜单选择“Admin settings”，右侧展示区选择“Settings”



在“Login and provisioning”部分中，选择“SMAL SSO”



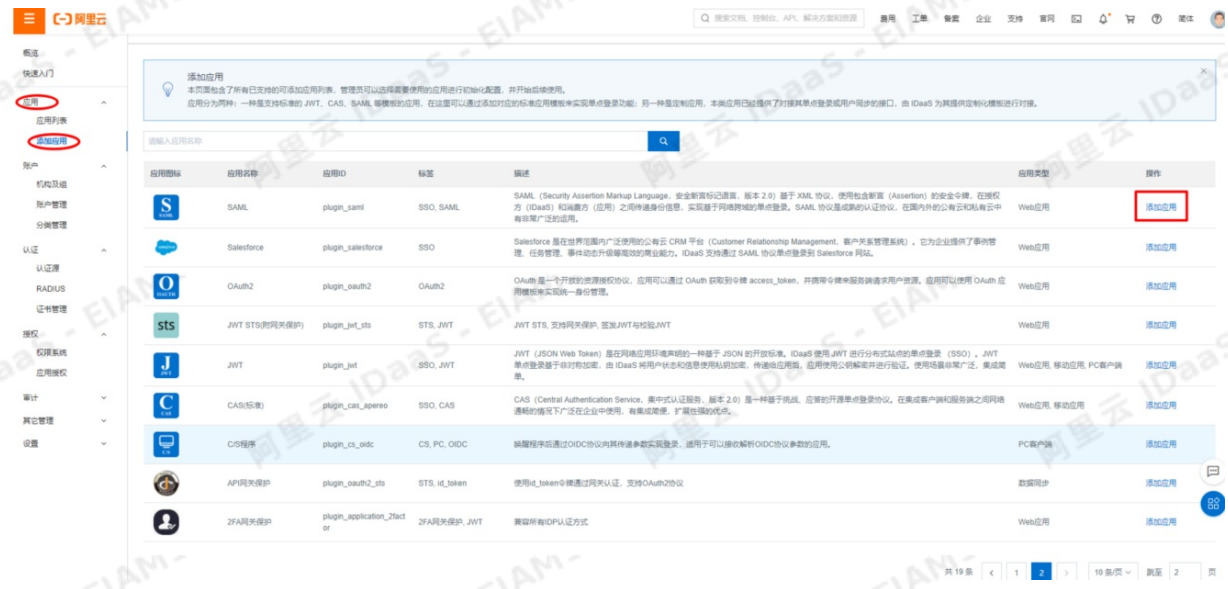
获取“SP entity ID”和“SP ACS URL”，供IDaaS配置SMAL应用时使用



二、IDaas中配置Figma的元数据信息

2.1、添加SAML应用

以IT管理员登录云盾IDaaS管理平台，点击左侧导航栏**应用** > **添加应用**在右侧选择一个SAML应用，点击添加应用。

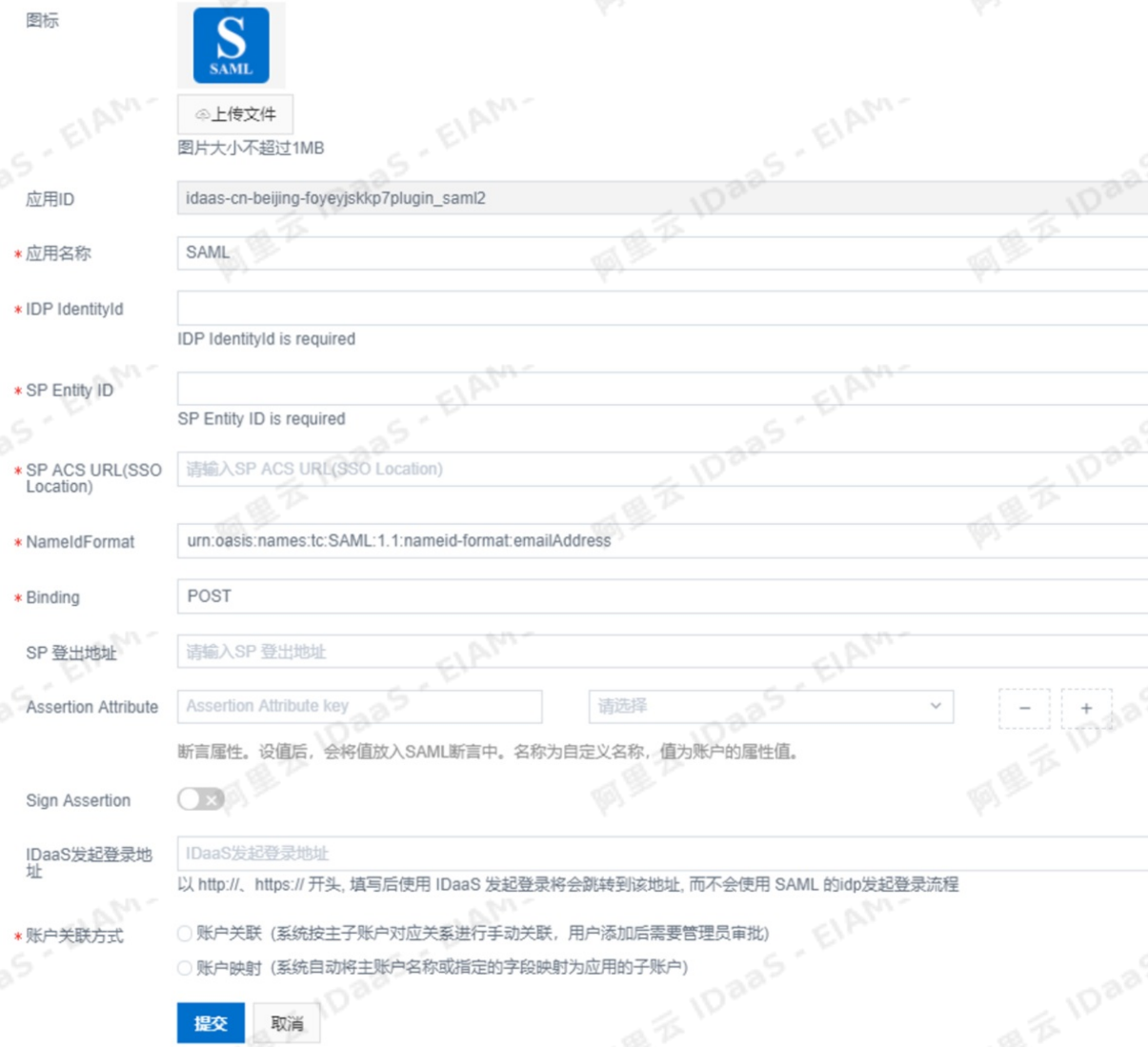


点击添加SigningKey按钮，输入名称等信息，系统会据此生成应用的证书，私钥保留在IDaaS，公钥导出到SP，用于IDaaS与SP应用通信的签名验签。

如果没有现成的证书可以选择， 则填写以下信息生成一个，其中的名称信息最好是和这个应用比如Figma关联的，方便将来识别。

无论是选择已有的还是刚添加的，找到对应的SigningKey，点击“选择”按钮。

接下来要填写更多的应用信息，名称等信息可以自定义，SP Entity ID、SP ACS URL(SSO Location)等信息从“**一、获取Figma中SAML SSO元数据**”中复制过来。



需要填写的主要信息如下：

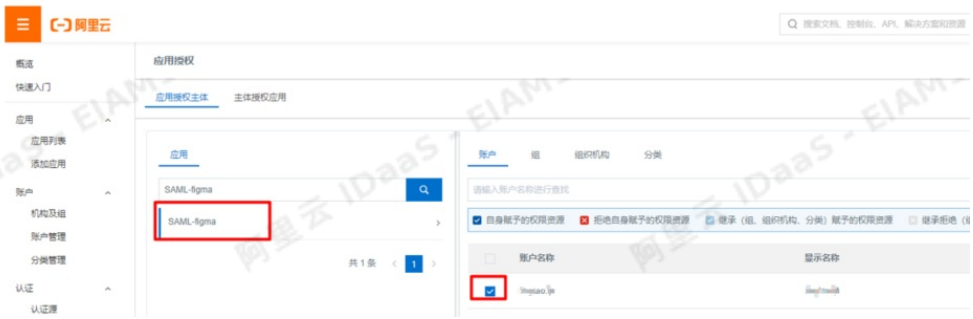
参数名称	说明
应用名称	所添加应用的名称，可以为任意值，但最好和应用相关。
IDP IdentityId	在IDaaS中设置的认证参数，需要将此参数配置到SP中，可设置为：**idaas**
SP Entity ID	在SP中设置的Entity ID，需要复制到IDaaS的配置中， 可以在Figma的SMAL SSO中获取
SP ACS URL（SSO Location）	单点登录地址，需要复制到IDaaS的配置中， 可以在Figma的SMAL SSO中获取
NameIdFormat	名称标识格式类型，这里以Figma为例 urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
Binding	默认POST方式发送消息到阿里云控制台

填写完成后提交保存， 如果应用是禁用状态， 可以继续修改重新提交。

2.2、启用应用并且授权

应用配置好以后需要先启用应用，并且将服务授权给一个账户，点击左侧导航栏**应用 > 应用列表**启用该应用并授权给账户。

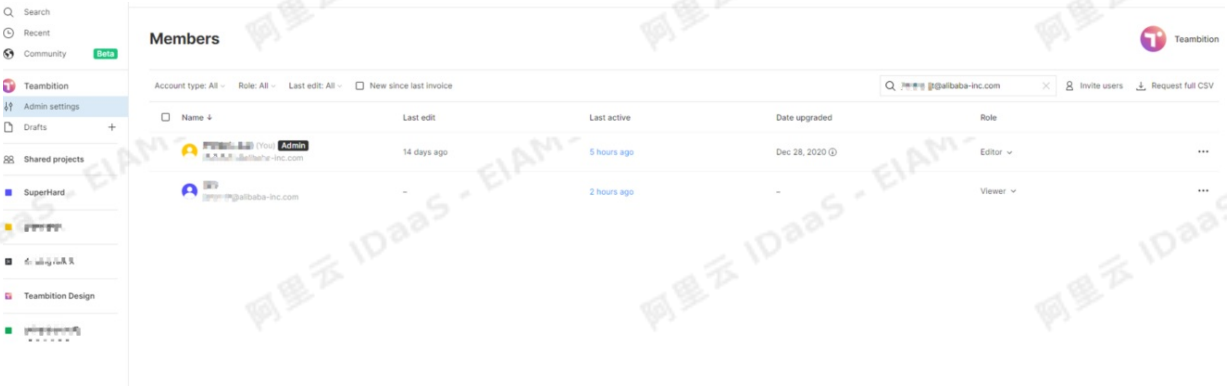
IDaaS支持多种方式进行授权，这里以按应用授权账户为例。



保存后， 这个用户登录就可以看到这个应用了。

2.3、IDaaS关联子账户

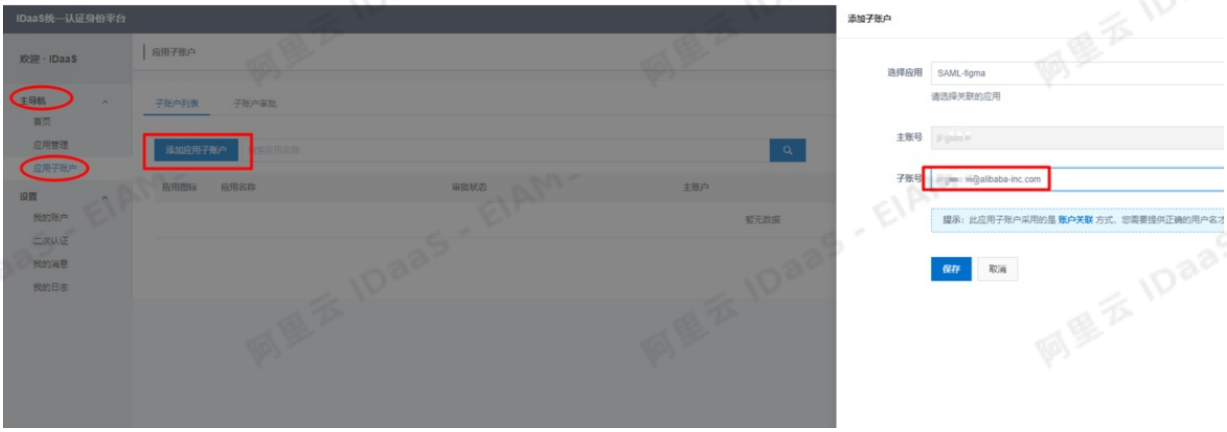
一个系统要SSO到另外一个系统，需要使用对方能够识别的子账号进行认证，往往登录到IDaaS的主账户和应用SP的子账户是不一样的，可以使用账号同步（两套系统中的账号信息相同）或者新建子账户进行账号映射的方法。账号映射是指给IDP的账户建立一个SP中已经存在的账户作为子账户，身份认证的时候通过子账户进行认证。例如SP系统中有个账户“test@alibaba-inc.com”，我们想用IDP系统中的“test.sp”账号SSO到SP，则需要给账号“test.sp”新建一个对应的子账户“test@alibaba-inc.com”。这里以Figma演示新建子账户的功能，如下图，Figma中有账户test@alibaba-inc.com。



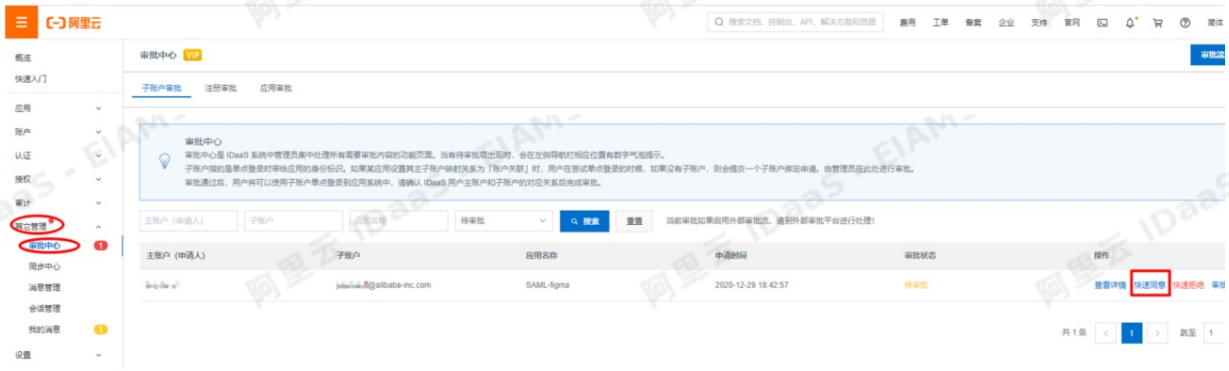
IDaaS中新建子账户有两种方式，操作如下：

2.3.1、普通账户申请关联子账户

普通用户登录，点击左侧导航栏 **主导航** > **应用子账户** 添加应用子账户功能中提交新建子账户申请。由于上一步Figma账户是jingtao.ljt@alibaba-inc.com，所以这里子账户的名称应该填“jingtao.ljt@alibaba-inc.com”。



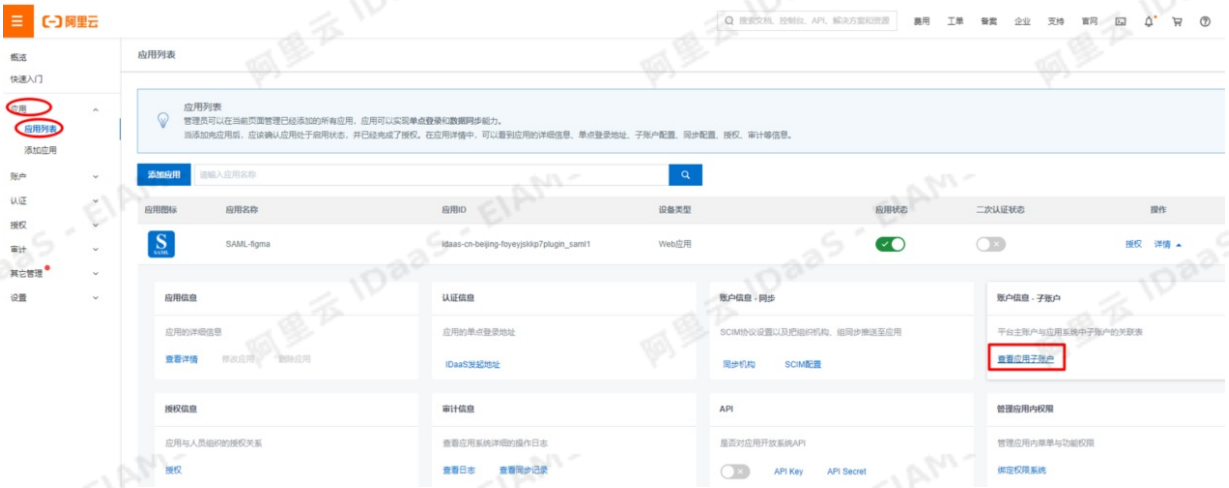
登录管理员账户，点击左侧导航栏 **其它管理** > **审批中心** 审核通过该应用子账户的添加。



2.3.2、管理员关联子账户

管理员新建子账户不需要审核过程，具体操作为：

登录管理员账户，点击左侧导航栏 **应用** > **应用列表** 找到添加的应用，点击详情中的查看应用子账户。



点击添加账户关联，添加子账户。

输入授权账户（主账户）和子账户，点击保存完成子账户添加。

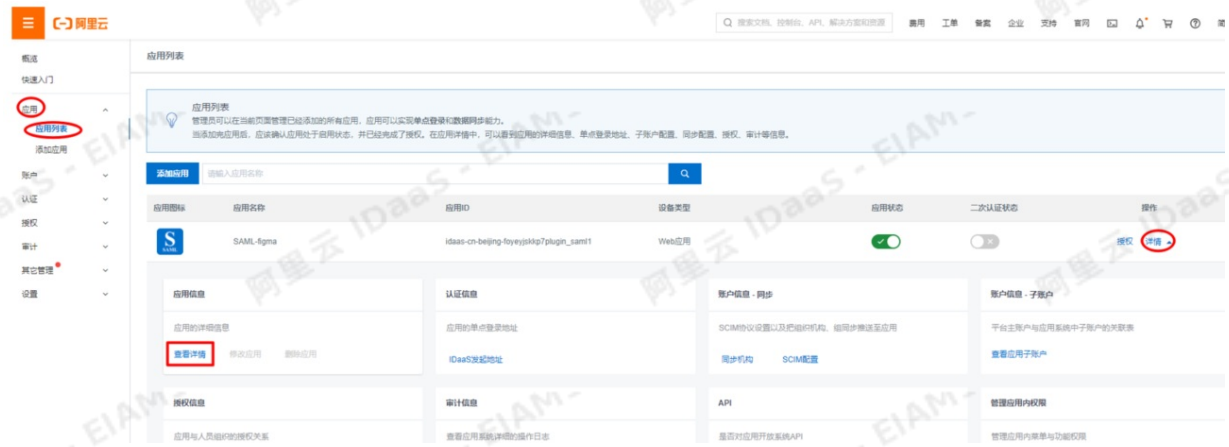
主账户：登录IDaaS使用账户。

子账户：SP应用中的子账户

三、Figma中配置IDaaS的元数据信息

3.1、获取IDaaS的元数据信息

以IT管理员账号登录云盾IDaaS管理平台，点击左侧导航栏**应用** > **应用列表** 选择刚才添加的应用，点击查看详情，如下图：



点击导出SAML元配置文件，将IDaaS的元数据文件保存到本地电脑。

IDaaS元配置文件示例如下：

```
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" entityID="idaas">
<md:IDPSSODescriptor WantAuthnRequestsSigned="false" protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
<md:KeyDescriptor use="signing">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
<ds:X509Certificate>
MIIB9zCCAWCgAwIBAgIIGpblwuj49s4wDQYJKoZIhvcNAQEFBQAwPjELMAkGA1UEBhMCQ04xDzANBgNVBAgMBuaxn+iLjzEPMA0GA1UEBwwG5Y2X5LqsMQ0wCwYDVQQDEwRTQU1MMB
MAkGA1UEBhMCQ04xDzANBgNVBAgMBuaxn+iLjzEPMA0GA1UEBwwG5Y2X5LqsMQ0wCwYDVQQDEwRTQU1MMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCY5IaMlj0e2AsN73hEfz
vs+VbTRZuQ0Lo3tFl6870DsVEe77CbrsMPdidFAbDYn5r1D1iSoo4H5ifwTxKTQ9Lc49D80B3j5XsfiKt20Pnlvzh9866e+qnL9qqcL9yAmDOaQIDAQABMA0GCSqGSIb3DQEBBQUAA4GBAI
aZYUNQD30RBZcQxkZuGiphcxShWO4SDB/T4rnXVS1YOdaeswn34u5fyWeR5zfLlPg3S/U+2J1JyPQ
</ds:X509Certi
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:KeyDescriptor use="encryption">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
<ds:X509Certificate>
MIIB9zCCAWCgAwIBAgIIGpblwuj49s4wDQYJKoZIhvcNAQEFBQAwPjELMAkGA1UEBhMCQ04xDzANBgNVBAgMBuaxn+iLjzEPMA0GA1UEBwwG5Y2X5LqsMQ0wCwYDVQQDEwRTQU1MMB
MAkGA1UEBhMCQ04xDzANBgNVBAgMBuaxn+iLjzEPMA0GA1UEBwwG5Y2X5LqsMQ0wCwYDVQQDEwRTQU1MMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCY5IaMlj0e2AsN73hEfz
vs+VbTRZuQ0Lo3tFl6870DsVEe77CbrsMPdidFAbDYn5r1D1iSoo4H5ifwTxKTQ9Lc49D80B3j5XsfiKt20Pnlvzh9866e+qnL9qqcL9yAmDOaQIDAQABMA0GCSqGSIb3DQEBBQUAA4GBAI
aZYUNQD30RBZcQxkZuGiphcxShWO4SDB/T4rnXVS1YOdaeswn34u5fyWeR5zfLlPg3S/U+2J1JyPQ
</ds:X509Certi
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location=
"https://...login.aliyunidaas.com/enduser/api/application/plugin_saml/idaas-cn-beijing-foye.../plugin_saml1/sp_sso"/>
<md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location=
"https://...login.aliyunidaas.com/enduser/api/application/plugin_saml/idaas-cn-beijing-foyey.../plugin_saml1/sp_sso_post"/>
</md:IDPSSODescriptor>
</md:EntityDescriptor>
```


获取上图位置URL地址。（**Figma配置IdP SSO target URL使用**）

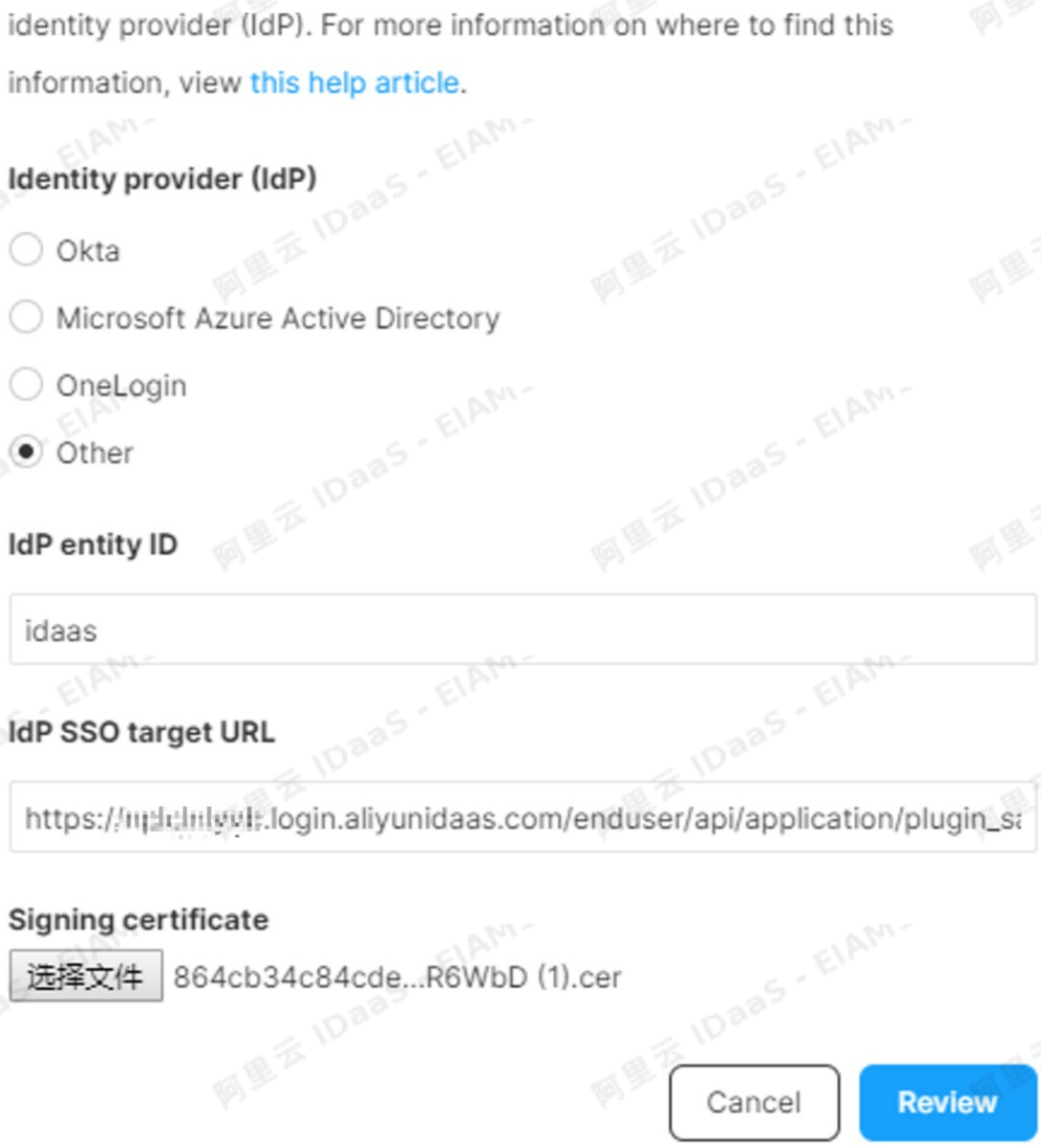
导出证书（Figma配置Signing certificate使用）：

点击左侧导航栏应用 > 添加应用在右侧选择一个SAML应用，点击添加应用。

在导出SigningKey页面，勾选“Base64 编码 X.509(.CER)(S)”,保存至本地电脑。

3.2、Figma中配置元数据信息

管理员登录Figma控制台，左侧菜单选择“Admin settings”，右侧展示区选择“Settings”，在“Login and provisioning”部分中，选择“SMAL SSO”，在配置页面点击“Edit configuration”，编辑配置页面，选择“Other”.



参数	说明
IDP IdentityId	与IDaaS添加应用时IDP IdentityId保持一致，这里以“idaas”为例
IdP SSO target URL	IdaaS单点登录地址，3.1中获取的元数据信息中的地址
Signing certificate	单点登录的证书，3.1中导入的SigningKey

四、功能演示

4.1 IDP发起SSO

配置完成后， 就可以检查结果了。 授权用户登录IDaaS，点击左侧导航栏 **主导航 > 首页** 在我的应用中点击该应用进行单点登录，点击应用的图标进行单点登录。

首次登录，Figma会要求进行一次账户认证，认证完成后进入登录。（只限首次认证登录）

认证成功后登录Figma，然后就可以看到Figma作为SP提供的资源了。

4.2 Figma发起SSO

同样，正确配置后，也支持SP发起，首先找到Figma登录地址，选择“Log in with SAML SSO”。

在SAML SSO页面输入Figma账户的邮箱地址

跳转到IDP进行用户认证，只有IDaaS中添加的账户进行登录

IDP认证通过后，然后就可以看到Figma提供的资源了。

1.11. Salesforce对接

本文为您介绍如何在IDaaS中使用SAML协议单点登录到Salesforce。

背景信息

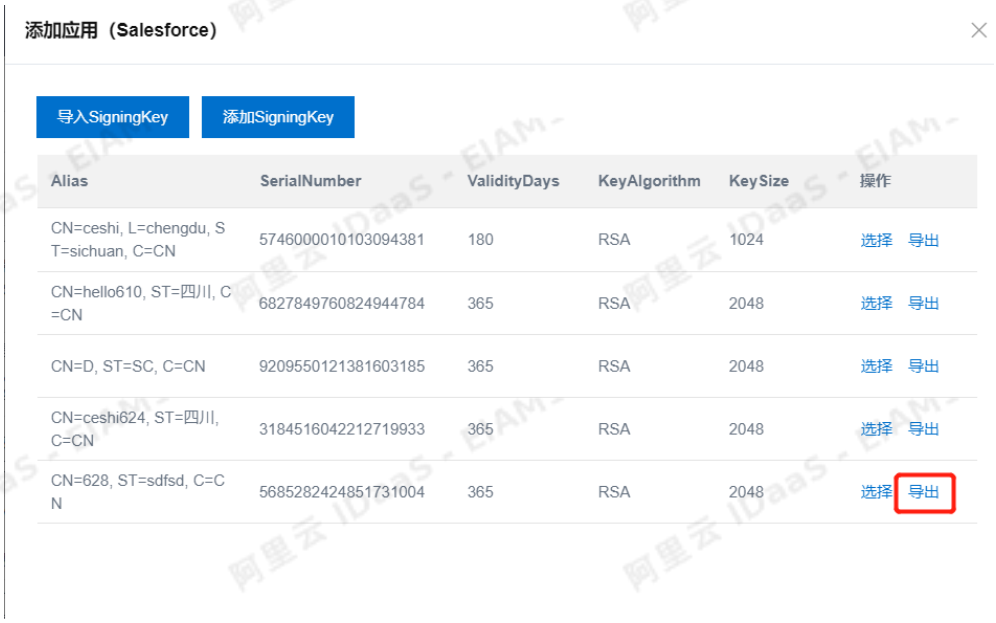
Salesforce是一家创建于1999年的客户关系管理（CRM）软件服务提供商，总部设于美国旧金山，可提供随需应用的客户关系管理平台。Salesforce支持SAML协议实现单点登录，本文将说明如何在IDaaS中使用SAML协议单点登录到Salesforce。

操作步骤

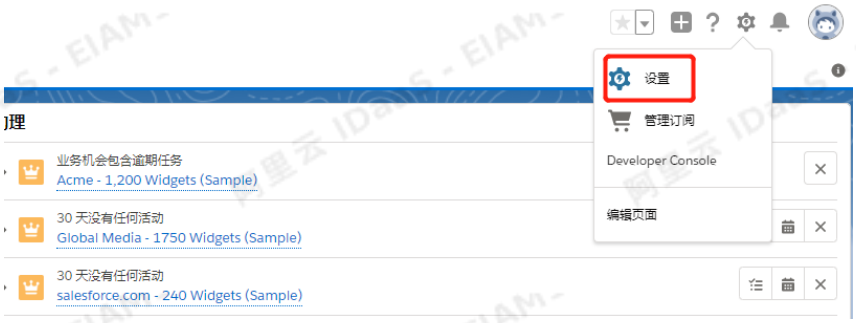
1. 以IT管理员账户登录云盾IDaaS管理平台。具体操作请参考 IT管理员指南-登录。

2. 在左侧导航栏，点击 **应用>添加应用** ，选择Salesforce应用模板,点击”添加应用”按钮。

3. 选择一个 SigningKey（如没有，可先添加一个SigningKey），并选择导出，此步骤需要会载一个cer证书到本地。



4. 以管理员账户登录 Salesforce，点击右上角“设置”按钮。



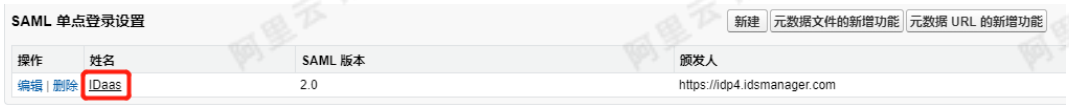
5. 进入设置主页，找到设置处，依次点击左侧菜单栏： 身份-单点登录设置，找到 SAML单点登录设置，点击“新建”按钮。

6. 进入Salesforce SAML单点登录设置页面。

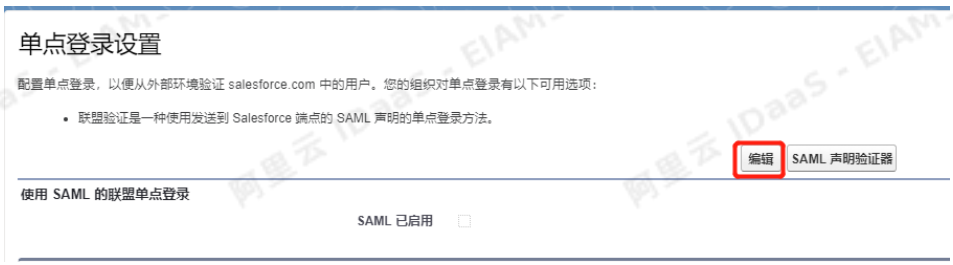
- 姓名：该SAML单点登录设置名字，随意输入；
- **颁发人**：注意此值应该与下面我们在IDP2中配置Saleforce SAML中**IDP Identity Id**一致；
- **实体ID**：https://SAML.Salesforce.com;
- **身份提供商证书**：选择我们刚刚在IDaaS导出到本地的证书文件；
- 请求签名证书：默认即可；
- 请求签名方法：RSA-SHA1；
- 声明解密证书：选择 “声明未加密”；
- SAML 身份类型：选择 “声明包含用户的Salesforce用户名”；
- SAML身份位置： 选择 “身份在“主题”声明的NameIdentifier 元素中”；
- 身份提供商登录URL，注销URL，自定义错误URL留白即可，点击保存。

7. 添加成功，会显示该SAML名称设置的详细信息，注意将Salesfore 登录URL复制出来，以备后用。

备注，也可点击该SAML设置的名称进入以上页面查看Salesforce 登录 URL。



8. 找到单点登录设置，联盟验证处，点击“编辑”按钮 。



9. 勾选 “SAML已启动 ”，点击保存 。

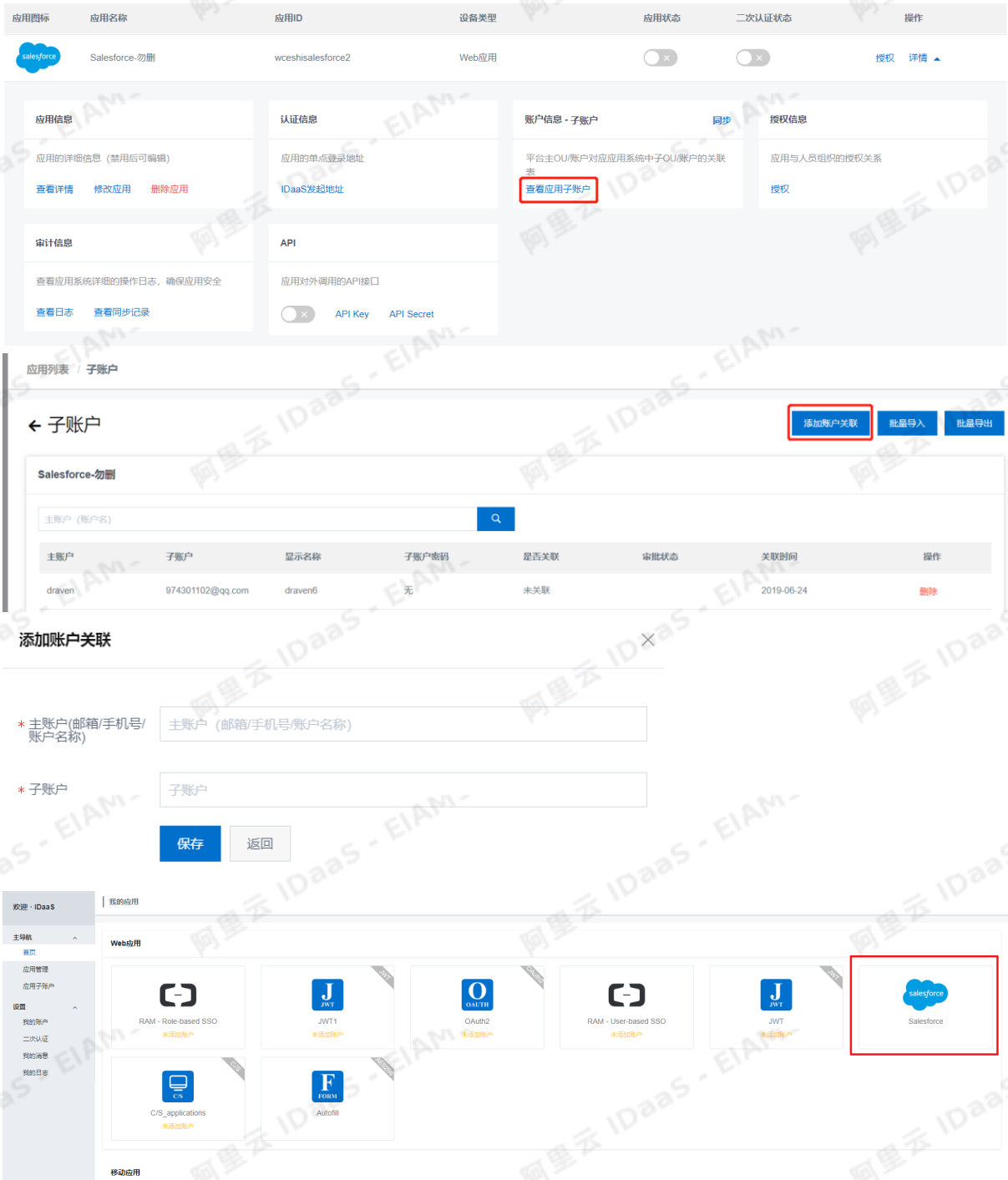
0. 回到IDaaS中添加Salesfore 页面,点击“选择”按钮，进入Salesforce 的SAML配置页面。

- IDP Identity Id为在Salesforce填写的颁发人值；
- SP ACS URL（SSO Location）为 Salesforce 登录 URL,为我们在上面复制的Salesforce 登录 URL值。

> 说明
> 该URL格式为 https://login.Salesforce.com?so=<您的组织ID> 的形式。 如果您不确定Salesforce组织ID，请转到Salesforce中的公司简介>公司信息来查找。

1. 启用应用并授权 。

2. 添加子账户并进行单点登录 。



通过以上步骤，完成了单点登录到Salesforce的功能。

2.标准协议模板使用指南

2.1. JWT 模板使用指南

一、概述

IDaaS平台提供了基于JWT标准协议实现的应用插件，使用该插件，业务系统可以快速的接入IDaaS平台，从而完成单点登录。并且JWT应用插件支持从SP（业务系统）发起单点登录请求，跳转到IDaaS平台，进行登录，再跳转回业务系统完成JWT令牌认证和业务系统的登录。同时，也支持从IDaaS平台直接发起单点登录请求，传递JWT令牌后，在业务系统进行验证，完成登录。

本文档主要为JWT应用配置人员或开发人员提供完整的JWT应用配置过程或开发流程，并提供相应的SDK下载。

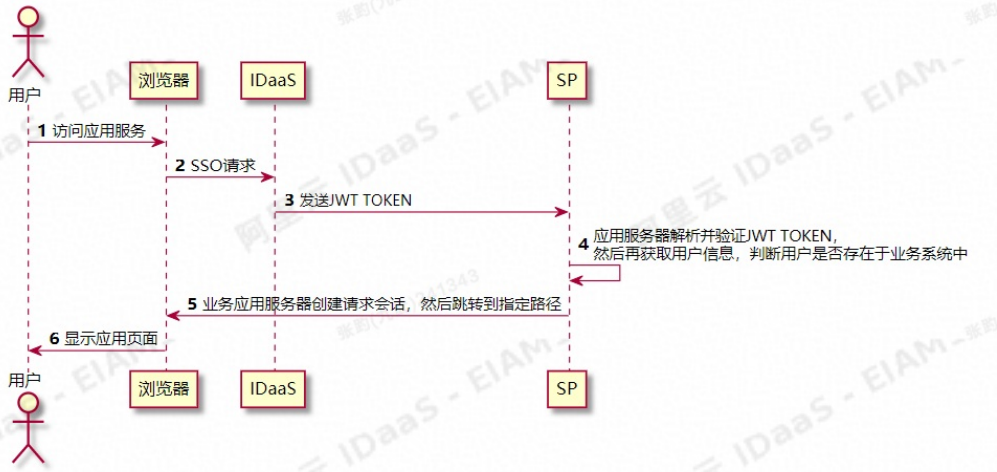
1.1 IDP/SP 发起单点登录的区别

Json web token （ JWT ），是一种用于双方之间传递安全信息的简洁的表述性声明规范。JWT作为一个开放的标准（RFC 7519），定义了一种简洁的方法用于通信双方之间以 JSON 对象的形式安全的传递信息，该 token被设计为紧凑且安全的，特别适用于分布式站点的单点登录（SSO）场景。

同样， IDaaS平台提供的JWT单点登录应用插件支持IDP发起和SP发起，二者主要共同点在于整个JWT的认证流程（后半截）是相同的，都需要业务系统开发JWT令牌验证和解析的接口，并且需要根据解析出来的用户子账户信息，判断用户是否为该业务系统用户。如果需要跳转到特定页面， 用户可以通过在IDaaS平台填写target_url（填写地方请参考操作步骤 Step1 创建JWT应用），或者在SP发起地址后面拼接target_url请求参数（注：个别SP开发的针对线上老版本IDaaS的SSO，target_url参数当初是以redirect_url参数来接收的），以实现页面二次跳转，达到deep-linking的目的， 上述功能和SAML中的RelayState是一致的。

IDP发起和SP发起二者的不同点在于从IDaaS平台发起单点登录，用户可以通过点击IDaaS平台首页的JWT应用，就能完成JWT令牌认证和业务系统的登录。从SP（业务系统）发起单点登录，系统不一定已经完成了IDaaS平台的登录，或者登录信息过期失效，这时候IDaaS系统会跳转到登录页面，登录完成后，再继续完成JWT令牌认证和业务系统的登录。

二、实现原理



上述时序图阐述了基于JWT发起SSO登录请求时的基本流程，该流程主要分为以下6个步骤：

1）用户通过浏览器访问 IDaaS应用服务 。

2）浏览器向IDaaS发起单点登录请求。

3）IDaaS 生成 JWT token 令牌发送到业务系统。

4）业务系统获取到 token 令牌，用提供的插件或方法解析验证 JWT token 令牌，解析成功获取到用户信息并验证用户是否存在于业务系统中。

5）业务应用服务器创建自己系统的请求会话，然后跳转到指定路径。

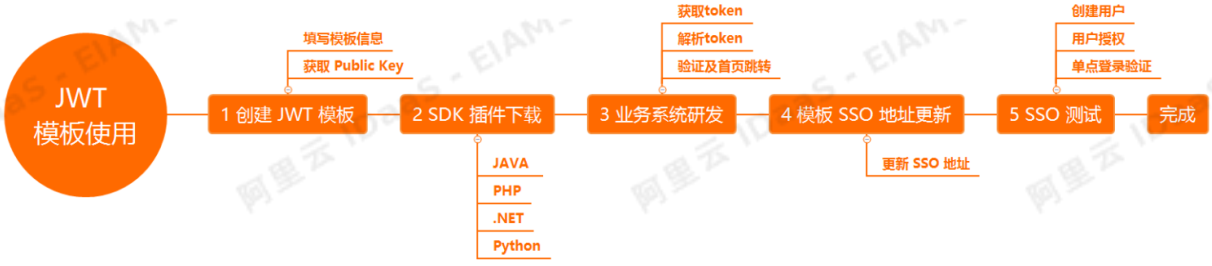
6）浏览器显示应用页面，完成sso登录。

验证通过：业务系统重定向到用户首页，或指定的二级页面。

验证失败：业务系统拒绝登录并页面提示错误信息。

三、对接流程图

下面是开发对接一个新的应用支持JWT协议SSO的过程。



3.1 主要流程

该流程主要帮助开发者理解和集成一个SP应用支持JWT单点登录，从而完成整个JWT应用模版的使用。流程主要分为以下6步：

Step1 创建JWT应用

该步骤主要帮助开发者在IDaaS平台创建一个JWT应用，让SP（业务系统）系统能接入IDaaS平台的JWT SSO登录，在该步骤中，主要是一些必要字段的填写，以便于完成后面的整个流程。

Step2 SDK下载

该步骤主要帮助开发者更加便捷的开发JWT token验证接口。使用IDaaS提供的JWT 验签SDK包，便于开发后续的JWT 验签接口。

Step3 业务系统研发

该步骤主要帮助开发者开发自己业务系统的JWT 验签接口，完成业务系统的用户验证和登录。

Step4 IDaaS上更新SSO地址

该步骤主要帮助开发者在完成JWT验签接口开发工作后，进行必要的接口验证工作。如果第一次在IDaaS平台创建JWT应用时，填写的JWT 验签接口不准确（redirect_uri）,可以再次更新该地址。

Step5 单点登录效果验证

该步骤主要帮助开发者验证JWT应用配置的完整性和自己开发的JWT验签接口是否正确，验证SSO登录流程是否能正确完成。

Step6 完成

四、操作步骤

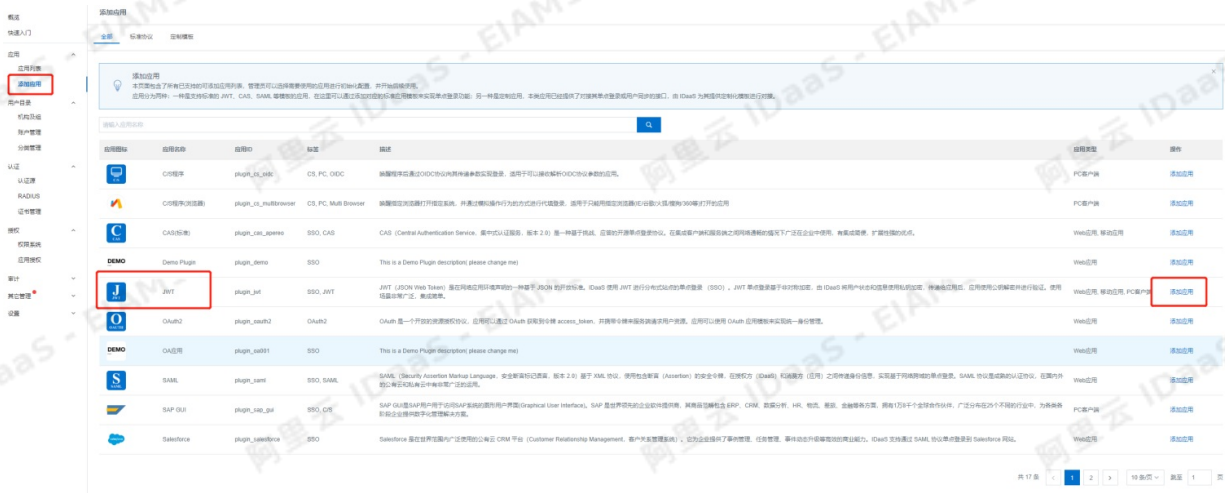
4.1 Step1创建JWT应用

4.1.1 登录 IDaaS 管理员平台。

使用 IT 管理员账号登录云盾 IDaaS 管理平台。具体操作请参考 IT管理员指南-登录。

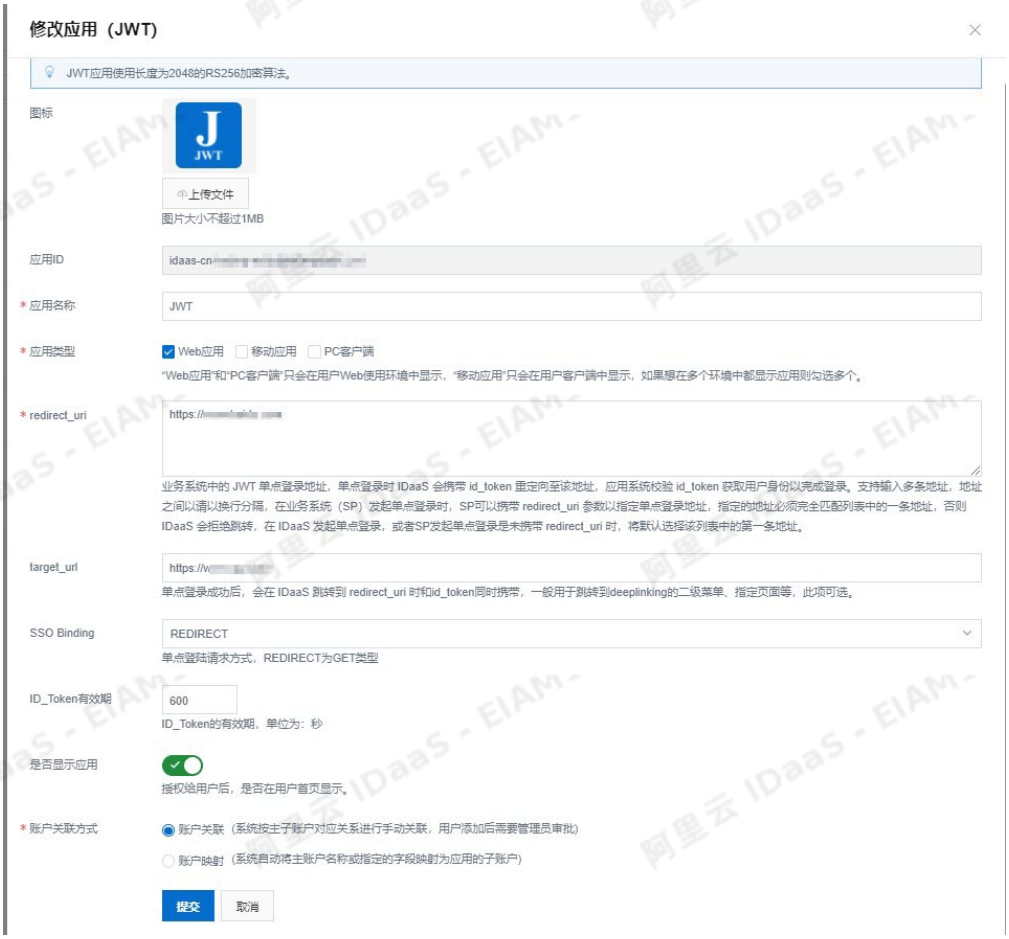
4.1.2 添加 JWT 应用。

在【应用】-【添加应用】中，找到应用名称为： JWT，点击右边【添加应用】按钮。（注意：请不要选择成jwt证书）。



4.1.3 填写信息并保存

根据需要填写如下信息：



IDaaS平台提供的参数， 具体如下：

1. 图标：业务应用的 logo 图片。

2. 应用 ID：IDaaS自动生成的应用 ID，不允许修改，且唯一。

3. 应用名称：填写创建应用的名称。

4. 应用类型：代表该服务支持的设备类型，标记使用。

5.SSO Binding：单点登录请求方式，REDIRECT 为 GET 类型，也可选择 POST。

7.ID_Token有效期：单位秒。

8.是否显示应用：授权给用户后，是否在用户的IDaaS平台首页显示，默认开启。若关闭，用户登录IDaaS平台首页，将看不到该应用。

9.账户关联方式：

a.账户关联（系统按主子账户对应关系进行手动关联，用户添加后需要管理员审批）。

b.账户映射（系统自动将主账户名称或指定的字段映射为应用的子账户）。

SP（业务系统）需要考虑的参数：

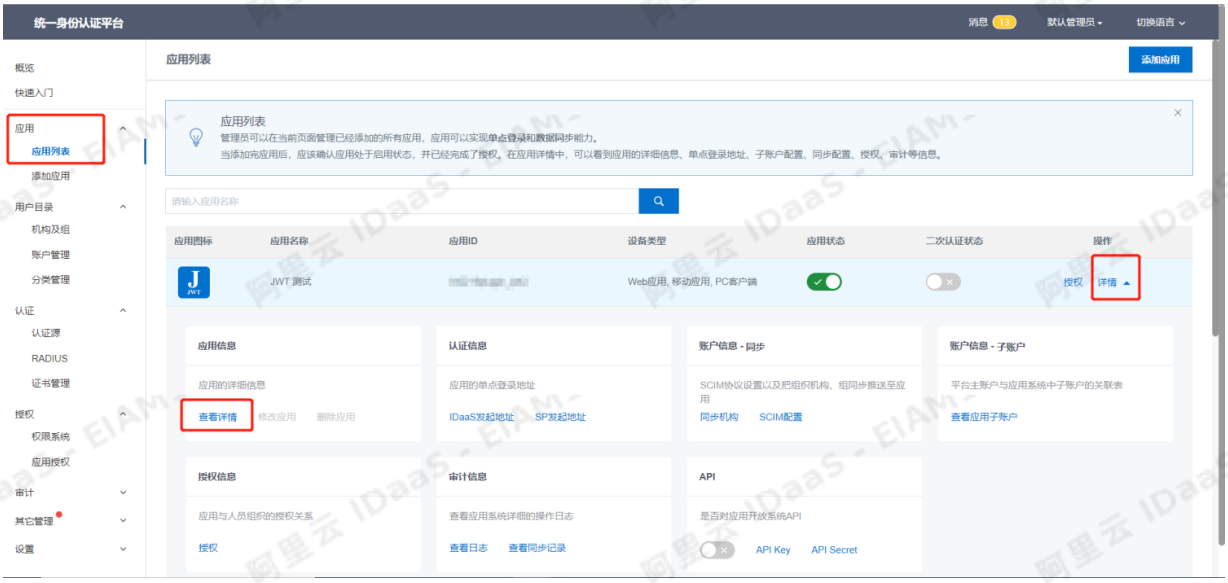
一个全新的应用从不支持JWT， 到支持， 需要开发几个URL，以下为两个重要参数：

1. redirect_uri：业务系统中（或 PC 程序）的 JWT SSO 地址，在单点登录时IDaaS将向该地址用[GET]方式发送 ID_Token 信息，参数名为id_Token，业务系统通过 ID_Token 与 Public Key 可获取JWT token中的用户信息。

2. target_url: 业务系统中在通过JWT系统完成身份认证成功后，重定向的 URI。一般是一个HTTP开头的URI，用于跳转到二级页面等。若设置了该 URI，在IDaaS平台在完成JWT协议身份认证成功时，会以参数 target_url传递该值，若未设置该值，若此时SP发起的SSO请求中有参数target_url，则会按照请求参数传递该值，此项可选。如果target_url为空，由SP决定跳转到哪个页面， 一般是默认的门户页面。

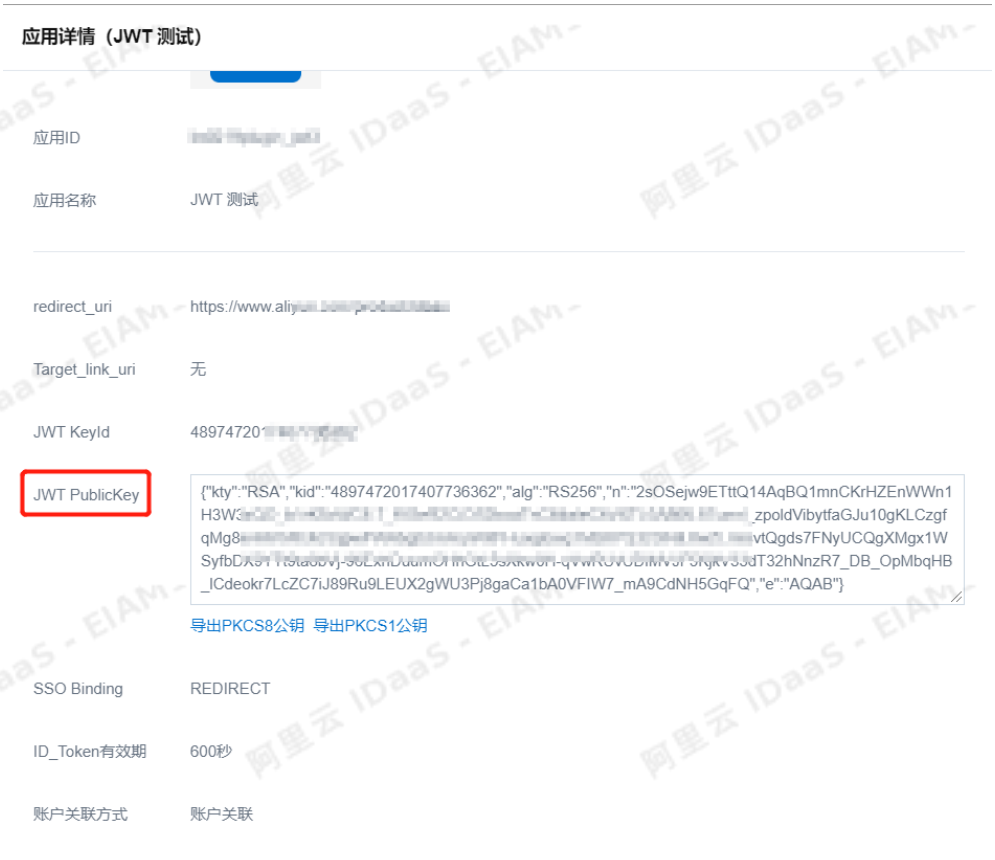
4.1.4 导出公钥

基于JWT的非对称签名/验签机制， 完成上面应用创建，私钥保存在IDaaS 后台，作为签名使用， 公钥需要导出传递给SP验签使用。 在【应用列表】中，就可以找到新创建的应用。点击【详情】按钮，点击【查看详情】。



找到 JWT PublicKey

复制粘贴到文本txt，或者使用下方导出，都可以将JWT PublicKey导出。将其交给需要接入JWT协议认证的SP业务系统，用作 ID_Token 的验签解析。



4.2 Step2 SDK下载

在这里，IDP配置好了， 第三方业务系统SP就要开始研发的准备工作了。所有需要接入JWT应用源的SP都需要开发相对应的JWT id_token的SSO接收接口，该接口主要用于id_token的解析，签名验证，和用户信息的抽取。然后在业务系统中对比获取到的用户信息，对比成功后，创建业务系统自己的登录会话，完成SSO登录。

IDaaS提供 4 种语言的 SDK 集成方式：JAVA/PHP/.NET/Python：

JAVA SDK下载

JAVA SDK - JDK 1.6

JAVA SDK - JDK 1.7

JAVA SDK - JDK 1.8

PHP SDK下载

PHP-JWT-SDK

.NET SDK下载

.NET-JWT-SDK

Python SDK下载

Python-JWT-SDK

当然，如果您的业务系统是除此之外其他语言也可以进行对接，需要您自行编写解析 ID_Token 的代码，需要可以参考

JWT 官网

4.3 Step3 业务系统研发

如上所述， SP研发核心需要考虑的：

1）能够接收到令牌

2）能够成功验签解析令牌，拿到用户Sub信息

3）匹配用户信息是否与当前自己的子账号一致，完成匹配之后，创建业务系统自己的会话

4）跳转至用户首页

4.3.1 JAVA 插件式集成

配置环境

根据java JDK版本，选择相对应的SDK版本，如常用的java JDK版本为1.8，请选择

JAVA SDK - JDK 1.8。

接收令牌

假设IDaaS通过POST或Redirect 将id_token 传递到SP，SP首先需要提供一个SSO的URL。

接收示例：

```
url 示例:
https://localhost/JWT/sso/login?id_token=eyJhbGciOiJSU****IsImtpZCI6IjEwMjQxNjE0NzI2Nzg2MjI0NjgifQ
```

```
// id_token 是 IDaaS 请求时带来的，在 requestParam 里获取，PublicKey是在 IDaaS 里注册应用时生成的，注册完可见，此示例代码是获取用户信息。
// JWT SSO
@RequestMapping(value = "/JWT/sso/login")
public String SSO Url(@RequestParam String id_token, String target_url, Model model, HttpServletRequest request){
    //1.接收方法为GET方式,参数名为 id_token
    //2.<解析令牌>为解析 id_token 并验证代码
}
```

解析令牌

拿到id_token后，为了保证不是重放或中间人攻击，需要对其进行验签，前面导出的PublicKey主要被用于这个目的。注意，不同语言的SDK可能用到的PublicKey格式是不同的。

```
//1.使用公钥，解析 id_token
// 使用PublicKey解析上一步获取的 id_token 令牌，并验证id_token
DingdangUserRetriever retriever = new DingdangUserRetriever( id_token, PublicKey);
DingdangUserRetriever.User user = null;
try {
    //2.获取用户信息
    user = retriever.retrieve();} catch (Exception e) {
    LOG.warn("Retrieve SSO user failed", e);
    return "error";
}
//3.判断用户名是否在自己系统存在,isExistedUsername()方法为业务系统自行判断数据库中是否存在
if (isExistedUsername(user.getUsername())) {
    //4.如果用户存在,则登录成功，然后创建业务系统自己的会话（如session的更新），具体操作，根据各自业务系统的需要进行开发，以下只做示例
    User SPUser = userService.updateLoginTimes(user.getUsername());
    request.getSession ().setAttribute(HttpSessionSecurityContextRepository. SPRING_SECURITY_CONTEXT_KEY, saveSecurity( SP User));
    //5.如果请求参数中带有target_url（注：线上有些版本参数名为redirect_url），那么返回此指定的url页面
  if (StringUtils.isNotEmpty(target_url)) {
      return "redirect:" + target_url;
    }
    //6.否则返回SP自定义的默认操作页面
    return "redirect:../../index";
} else {
    //7.如果用户不存在,返回登录失败页面,提示用户不存在
    model.addAttribute("error", "username { " + user.getUsername() + " } not exist");
    return "error";
}
```

4.3.2 PHP插件式集成

配置环境

在本例中，使用composer管理一个第三方JWT库（可选）。

同样， 假设IDaaS通过POST或Redirect 将id_token 传递到SP，SP首先需要提供一个SSO的URL。

接收示例：

```
url 示例：
https://localhost/JWT/sso/login?id_token=eyJhbGciOiJSUzI1****mtpZCI6IjEwMjQxNjE0NzI2Nzg2MjI0NjgifQ
```

接收令牌

如上， JWT 的 id_token 将会以URL参数的方式传进callback页面（同Java的），直接将其读取出来：

```
/* 使用composer 载入 php-JWT第三方库
* 命令行 composer require firebase/php-JWT
* 库链接：https://github.com/firebase/php-JWT
* 使用Firebase的这个第三方库来实现对JWT的解密，如果不用composer的话，请自行添加源文件
* 你也可以使用其他能对 JWT token 进行RS256解码的工具或库
*/
// 在这里将 JWT 库引入，在这里为了便捷demo直接使用
// 推荐使用
require 'vendor/autoload.php';
use \Firebase\ JWT \ JWT ;
// 本地存储public key公钥的位置
$public_key_location = "LOCATION/TO/YOUR/PUBLIC-KEY/XXX.pem";
// 读取公钥信息，公钥在这里存储在一个.pem文件内
$public_key = file_get_contents($public_key_location);
// 从url的参数中读取 id_token ，即令牌
if (!empty($_GET[" id_token "])) {
    $JWT = $_GET[" id_token "];
    // 这里继续第二步：解析令牌
}
```

获取到id_token之后，接下来就是对令牌的解析和验签步骤。

解析令牌

拿到id_token 后， 为了保证不是重放或中间人攻击， 需要对其进行验签，前面导出的PublicKey主要被用于这个目的。利用第三方库 php-JWT进行验签，获取到用户信息。验证通过后创建业务系统自己的会话，然后再跳转到SP登录后页面，失败则拒绝， 返回SSO失败页面：

```
phptry{
    /**
    * You can add a leeway to account for when there is a clock skew times between
    * the signing and verifying servers. It is recommended that this leeway should
    * not be bigger than a few minutes.
    *
    * Source: http://self-issued.info/docs/draft-ietf-oauth-json-web-token.html#nbfDef
    */
    // Firebase的 JWT 库的一个参数，不出问题的话可以忽略
    // （可选）当服务器时间与本地时间不符时，可以通过这个leeway参数来调整容错
    JWT::$leeway = 60; // $leeway in seconds
    // 使用公钥、使用RS256算法对 JWT （即第一步传进来的 id_token ）进行解密
    $decoded = JWT::decode($JWT, $public_key, array('RS256'));
    // 将解密的结果从class转化成PHP array
    $decoded_array = (array) $decoded;
    // 打印出解密的结果，成功!
    print("解密结果:<br>");
    foreach ($decoded_array as $key => $value) {
        print $key . ": " . $value . "<br>";
    }
    // 获取到用户信息后，判断该用户是否存在于你的系统内
    if (userExistsInSystem()) {
        // 如果用户存在，那么登录成功

         //登录成功后，创建业务系统自己的会话,略

         //会话创建完成后，如果有target_url参数，跳转到该地址
    } else {
        // 如果用户不存在，那么登录失败，跳转到显示错误页面
    }
catch(Exception $e) {
  print "错误：" . $e->getMessage();
}
```

4.3.3 .NET插件式集成

配置环境

.NET Framework 4及以上。

同样， 假设IDaaS通过POST或Redirect将id_token 传递到SP，SP首先需要提供一个SSO的URL。

接收示例：

```
url 示例：
https://localhost/JWT/sso/login?id_token=eyJhbGciOiJSUzI1NiI****pZCI6IjEwMjQxNjE0NzI2Nzg2MjI0NjgifQ
```

接收令牌

如上，JWT 的 id_token会以url参数的方式传进callback页面（同JAVA的）。

```
// id_token 是 IDaaS 请求时带来的，在body里获取，PublicKey是在 IDaaS 里注册应用时生成的，注册完可见,此示例代码是获取用户信息。
// JWT SSO
[Route("jwt/sso/login")]
public ActionResult ssoUrl(String id_token ){
   //1.接收方法为GET方式,参数名为 id_token
   //2.<解析令牌>为解析 id_token 并验证代码
}
```

解析令牌

拿到id_token 后， 为了保证不是重放或中间人攻击， 需要对其进行验签，前面导出的PublicKey主要被用于这个目的。如果成功，获取用户信息，然后创建业务系统自己的会话，最后跳转进入SP用户登录后的页面， 否则返回SSO失败页面。

```
//1. 使用公钥，解析 id_token
string username;
DingdangSDK.DingdangUserRetriever retriever = new DingdangSDK.DingdangUserRetriever( id_token， PublicKey);
DingdangSDK.User user = null;
//2. 获取用户信息
user = retriever.retrieve();
username = user.sub;
//3. 判断用户名是否在自己系统存在
//4. 如果用户存在，则登录成功，创建业务系统自己的会话，返回登录成功后的页面，略
//5. 如果参数中有target_url，那么返回此SP指定的url页面
//6. 否则返回系统默认操作页面
//7. 如果用户不存在，返回登录失败页面，提示用户不存在
```

4.3.4 python插件式集成

下载资源库

本Python JWT 示例使用Py JWT 库来进行 JWT 的解析。

假设IDaaS通过POST或Redirect 将id_token 传递到SP，SP首先需要提供一个SSO的URL。

接收示例：

```
url 示例:
https://172.168.XX.XX/JWT/sso/login?id_token=eyJhbGciO****I1NiIsImtpZCI6IjEwMjQxNjE0NzI2Nzg2MjI0NjgifQ
```

```
// 库的 github 链接 https://github.com/jpadilla/pyJWT
pip install Py JWT
// 注：CentOS系统如果使用时无法导入算法 RSAAlgorthm时需要下载pyJWT的2个依赖包
yum install ibffi-devel
pip install cryptography
```

接收令牌

如上，JWT 的 id_token 将会以URL参数的方式传进callback页面（同JAVA的）。

```
def get_id token ( token ):
    if not token .strip():
        print(' token 信息不能为空')
    else:
        //这里继续第二步：解析令牌
get_id token (my_id token ); // 运行程序
```

解析令牌

拿到id_token 后，为了保证不是重放或中间人攻击， 需要对其进行验签，获取用户信息，前面导出的PublicKey主要被用于这个目的。验证通过后创建业务系统自己的会话，然后再跳转到SP登录后页面，失败则拒绝，返回SSO失败页面。注意，Python库需要的PublicKey格式不一定是和其它一致的。开发前需要在IT管理员权限下前往应用->详细->导出PKCS8公钥来获取解密JWT用的公钥，并安全地放置在能访问到的目录内。

```
## 2.解析令牌
通过JWT解密库，使用公钥对传入的 id_token 进行解密。将公钥以字符串的形式从文件中读取出来，并作为key进行解密：
// 引入用到的包文件
import JWT
import json
from JWT.algorithms import RSAAlgorithm
from JWT.utils import force_bytes
from utils import key_path
// 本例中key_path辅助方法是写在utils工具类中的
def get_user_ifon( id_token ):
    try:
        algo = RSAAlgorithm(RSAAlgorithm.SHA256)
        pem_key = open(key_path('D:\pythonDemo\key\public_key_pkc8.pem'), 'r')
        public_key = algo.prepare_key(pem_key.read())
        token_info = JWT.decode(force_bytes( id_token ),key=public_key,verify=True)
        user_info = json.loads(json.dumps( token_info))
        username = user_info['sub']
        print(username)
        # 3.判断用户名是否在自己系统存在
        # 4.如果用户存在，则登录成功，创建业务系统自己的会话，返回登录成功后的页面，略
        # 5.如果参数中有target_url，那么返回此指定url页面
        # 6.否则返回系统默认操作页面
        # 7. 如果用户不存在，返回登录失败页面，提示用户不存在
    except Exception as e:
        print(e)
```

上面用到的key_path方法是用来获取存放在硬盘上的public key位置的辅助方法，具体如下：

```
def key_path (key_name):
 return os.path.join(os.path.dirname(os.path.realpath(__file__)), 'keys', key_name)
```

总之，在完成JWT令牌的接收，验签和解析之后，业务系统需要根据JWT令牌解析出来的用户信息，判断用户是否存在于当前业务系统中。如果用户存在，业务系统创建自己系统的会话请求（如存放session，生成cookie等，请根据业务系统各自要求去实现），以保证用户的登录状态。然后判断是否有target_url参数，如果有该参数，那么返回该参数指定的url页面，从而完成SSO的登录。

4.4 Step4 IDaaS上更新SSO地址

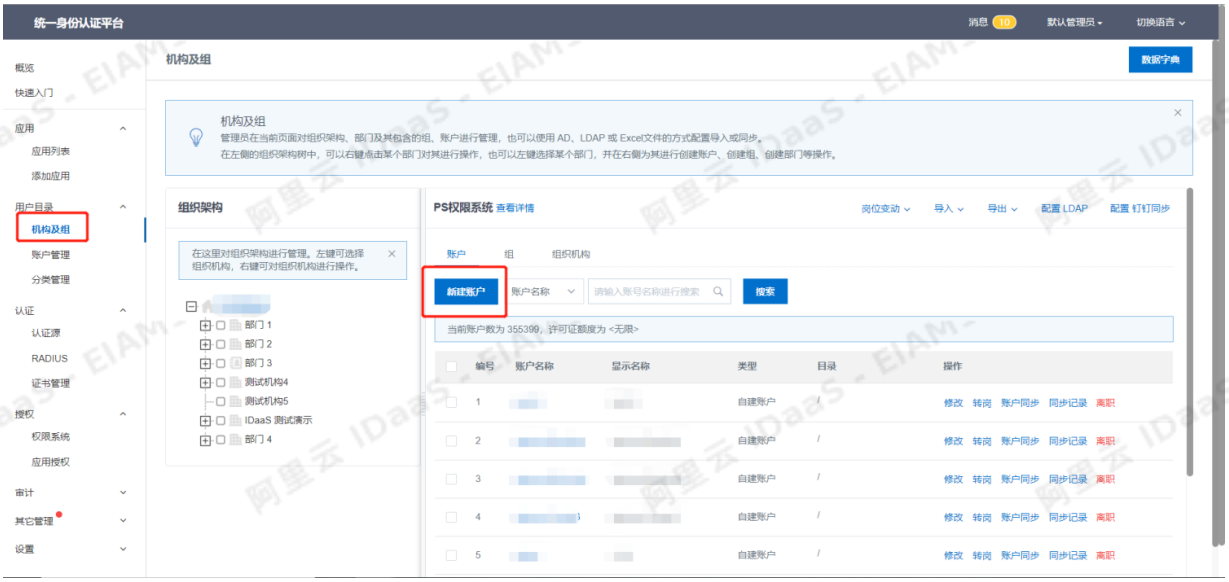
由于在第一步创建应用时，SSO 单点登录地址（即redirect_uri）不一定是SP开发完成后正确的URL。 所以，当业务系统开发工作结束后，有可能需要再返回 IDaaS 平台，以IT管理员身份，将这个地址进行更新，后续才能进行下一步的联调测试工作。

4.5 Step5 单点登录效果验证

4.5.1 从IDaaS发起单点登录

在完成IDaaS平台的redirect_uri更新之后，开发者可以新建一个测试账号进行单点登录效果验证，以确保JWT的应用源接入成功。

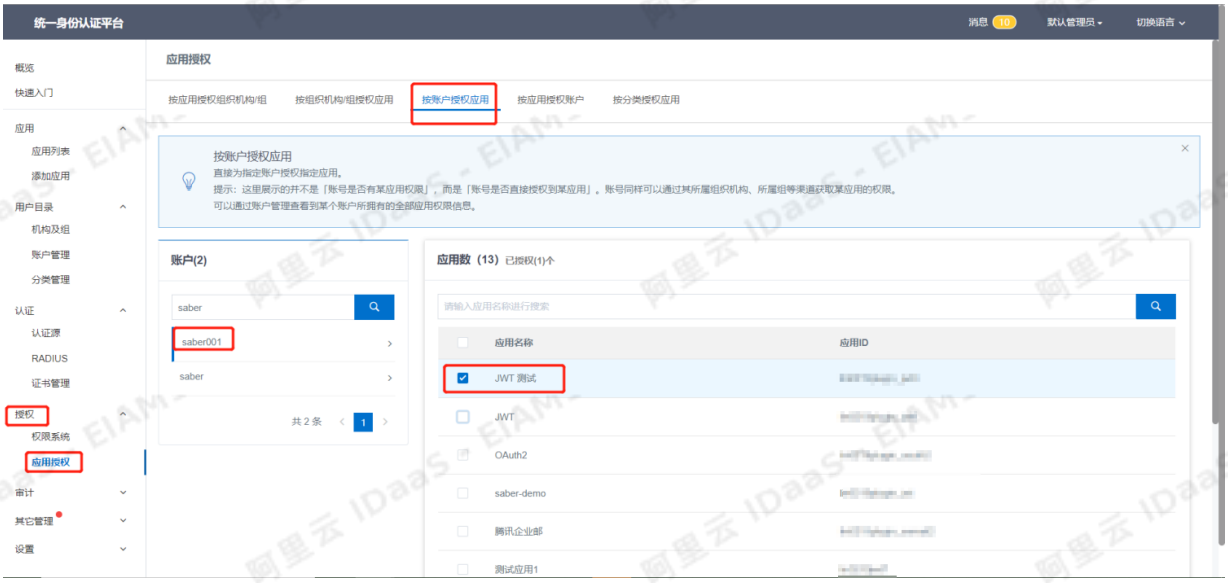
4.5.1.1 新建一个普通账号



4.5.1.2 账号授权

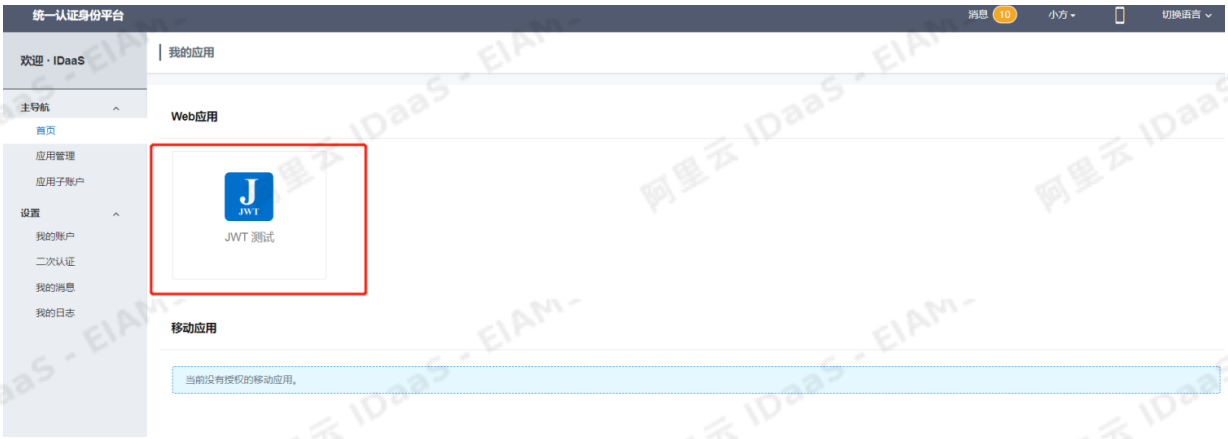
在新建账号时，会自动授权，比如可以看到给授权的应用，然后点击下一步，完成账号授权。

或者用另外一个方式，使用【授权】-【应用授权】-【按账户授权应用】功能，给账号分配需要测试的 JWT 应用。



4.5.1.3 使用测试账号登录

使用测试账号登录后，即可看到创建的JWT测试应用 logo图标。



注：若此时无法看到图标，请检查

（1）应用是否开启？（【应用列表】查看及开启）

（2）账号是否被授予应用权限？（【应用授权】查看及授权）

4.5.1.4 业务系统检查是否能获取到 id_Token

下一步， 点击这个图标后，会触发一个SSO URL， 通过检查IDaaS系统发送的id_token 和业务系统（SP）收到的id_token是否一致，从而确保业务系统能正确获取id_token。

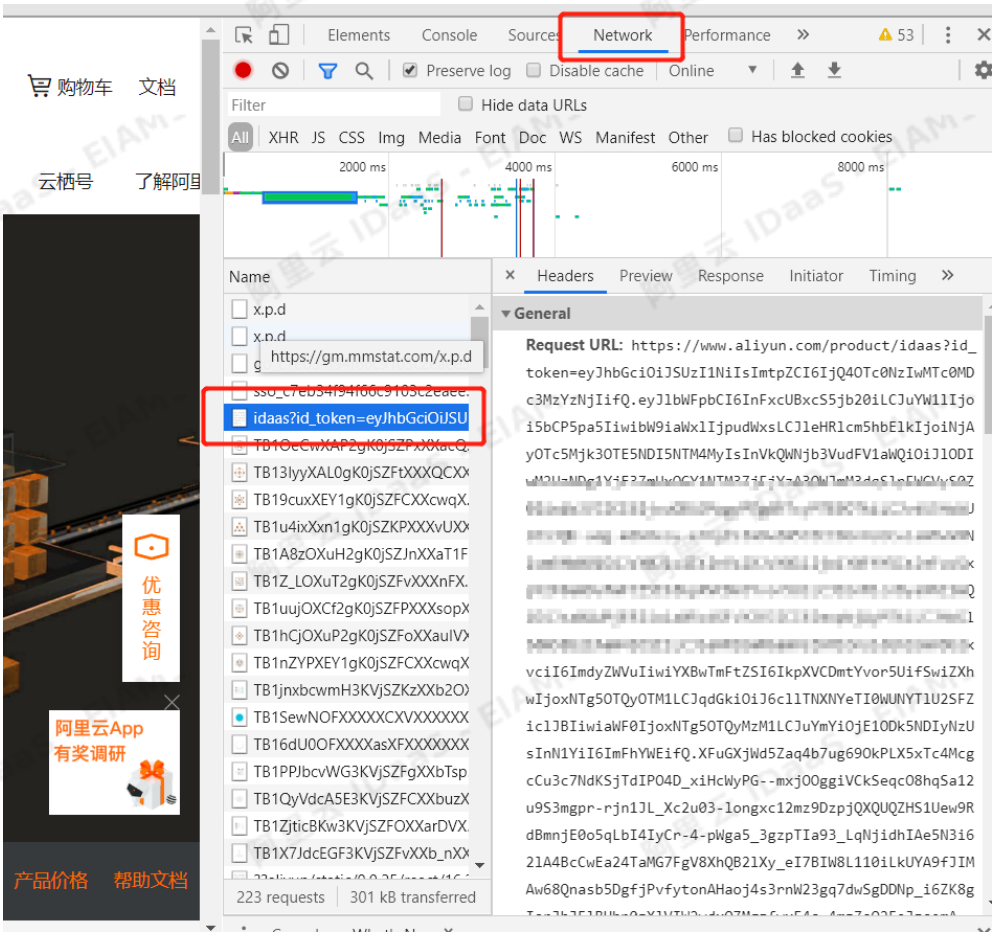
```
SSO URL 示例:
https://www.example.com/sso/login?id_token=xxxx
```

（1）首先登录IDaaS系统

（2）点击应用 logo

（3）可以在浏览器地址栏中看到 id_token 及SP二级页面的target_url信息，例如：

（4）也可以使用浏览器的 [开发者工具] 看到 id_token 信息，



（5）如果id_token 正常，此时业务系统就需要检查是否能够收到 IDaaS 发送的 id_token 并且成功解析验签，获取用户信息。

4.5.1.5 检查是否能正确跳转到业务系统指定页面

如业务系统成功获取并解析 id_token 正常，就应该能成功创建业务系统自己的会话（如session更新，生成cookie等），然后跳转到业务系统指定的页面。

4.5.2 从业务系统（SP）发起单点登录

上面介绍完了IDP发起， 下面介绍SP发起的SSO过程。

SP发起的URL地址由IDaaS平台在创建JWT应用时，自动创建，用户无法修改。

```
URL示例：
https://<idaas>/enduser/sp/sso/{应用ID}?enterpriseId={公司ID}
业务系统（SP）需要把SSO单点登录请求发送到该地址。
```

从SP发起的SSO单点登录请求，一般情况下，都是用户在业务系统的页面首先进行操作，如用户点击邮箱地址（htttps://www.example.com/?email=xxxx），此时，用户在未登录的情况下，邮箱服务需要把用户页面重定向到SP发起地址。然后用户在IDaaS完成登录，最后再次跳转到登录前的业务系统页面，即用户邮箱地址（htttps://www.example.com/?email=xxxx）。

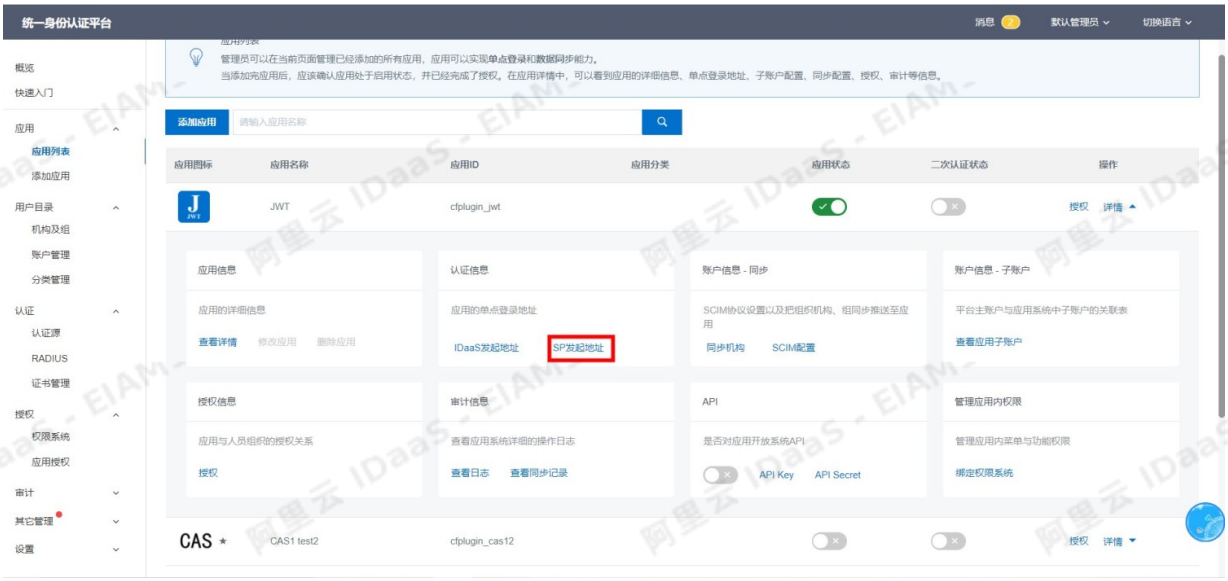
4.5.2.1 业务系统配置SP发起地址

一般情况下，SP的开发者需要开发相应的代码来完成整个SP发起的 SSO流程。

首先，开发者需要在业务系统开发中，添加SP发起地址（如业务系统首页的第三方登录按钮，或者在业务系统的登录拦截器），当用户需要登录业务系统时，页面需要重定向到SP发起地址， 同时SP系统应该把当前的URL， 生成一个随机数关联保存在本地， 把这个随机数作为target_url 传递给IDaaS。

SP发起地址获取方式：

请在IT管理员权限下前往应用->应用信息->点击【SP发起地址】进行复制。



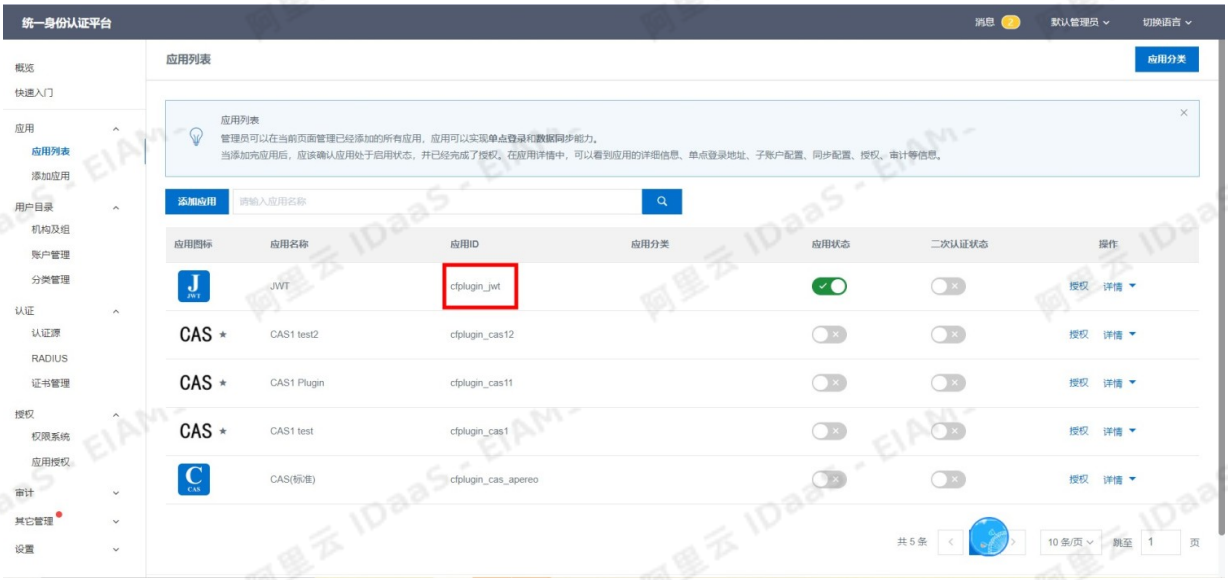
SP发起地址示例：

```
https://<idaas>/enduser/sp/sso/{应用ID}?enterpriseId={公司ID}&target_url=XXX
```

其中：

target_url：该参数为SP系统开发者可选参数，如果添加了该参数，IDaaS在完成登录后，会优先使用该值，覆盖在创建JWT应用时，填写的target_url，并把该值返回给SP。

应用ID：在IT管理员权限下前往应用列表 -> JWT应用源的应用ID。



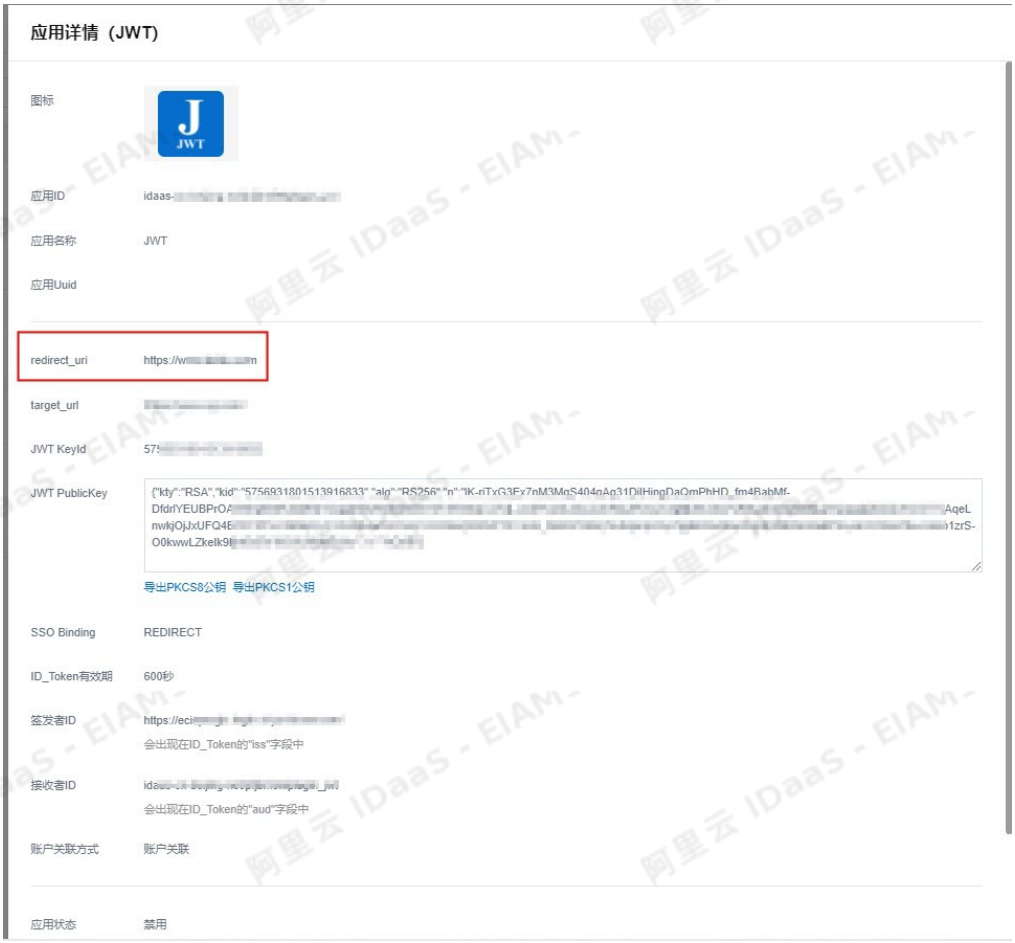
公司ID：在IT管理员权限下前往设置 -> 个性化设置 -> 公司信息查看。

4.5.2.2 从业务系统（SP）发起登录

在浏览器输入业务系统（SP）的地址，比如用户现在需要登录邮箱，此时用户首先点击的就是邮箱地址（如 https://www.example.com/?email=xxxx），然后邮箱服务需要把浏览器重定向到业务系统配置好的SP发起地址。

```
SP发起地址示例：
https://<idaas>/enduser/sp/sso/{应用ID}?enterpriseId={公司ID}&target_url=XXX
```

如果用户未登录，会跳转到IDaaS统一登录页面，进行登录。登录完成后，IDaaS系统会将浏览器重定向到redirect_uri地址（redirect_uri地址请在IDaaS平台 ->应用列表->应用信息里面查看）。

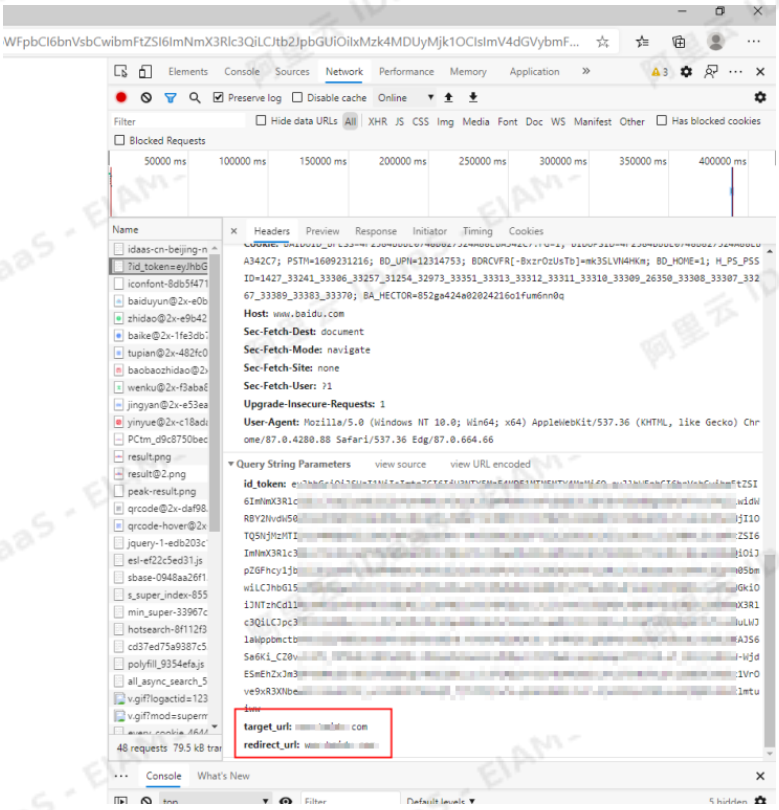


并且带上id_token及target_url参数。为了兼容老版本IDaaS平台，早期使用redirect_url来起到target_url相同的效果的， 注意redirect_url 和 redirect_uri 的区别， 所以现在请求参数中会返回target_url和redirect_url两个相同值的参数，参数的值都为需要跳转的二级页面地址，并且优先使用业务系统在SP发起地址里拼接的target_url。当业务系统没有在SP发起地址里拼接具体的target_url参数时，IDaaS系统会采用在创建JWT应用时，填写默认的target_url。

```
redirect_uri地址示例：

https://www.example.com/sso/login?id_token=xxxx&redirect_url=yyyy&target_url=yyyy
```

如下图所示， 会有3个参数拼接到地址后面。



最后业务系统（SP）完成JWT令牌的验签和用户信息匹配的工作。验签成功后，业务系统将创建本系统的请求会话，然后把浏览器重定向到target_url地址。

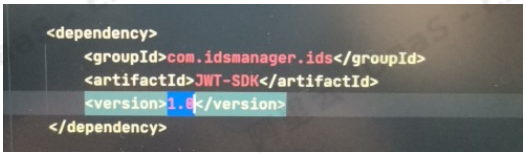
4.6 Step6 完成

当所有测试工作结束后，就在 IDaaS 上完成了 SSO 单点登录的对接工作。如果IDaaS系统和SP（业务系统）都是使用的测试环境，那么在正式上线切换时，需要再次将JWT模板中的地址修改为正式生产环境的地址。

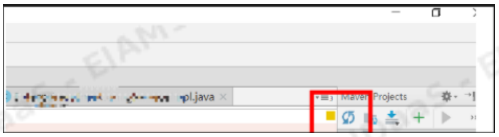
五、FAQ

5.1 提示Maven库错误

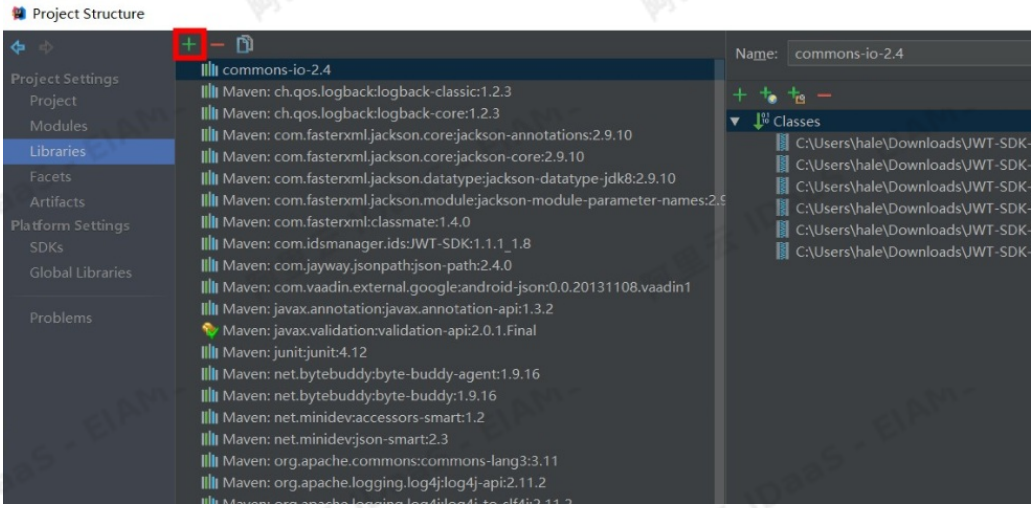
如果是找不到这个maven库依赖：



该SDK是由IDaaS提供不是从官方下载，没有官方的maven仓库，可以手动将Step2里面下载的SDK，安装到引用的maven库中。用idea的话，刷新下图位置，就能引入到本地maven中。



或者通过IDEA的Module Settings，手动添加JWT-SDK 本地jar包。



2.2. SAML 模板使用指南

一、概述

IDaaS平台支持基于标准SAML协议的SSO（Single Sign On 单点登录），IDaaS作为SAML协议中的IDP（Identity Provider身份提供方）角色，提供用户的身份认证服务，用户可以登录一次就直接使用多个SP（Service Provider 业务提供方）的服务，免去了每个应用都要登录的烦恼。

二、IDP发起和SP发起

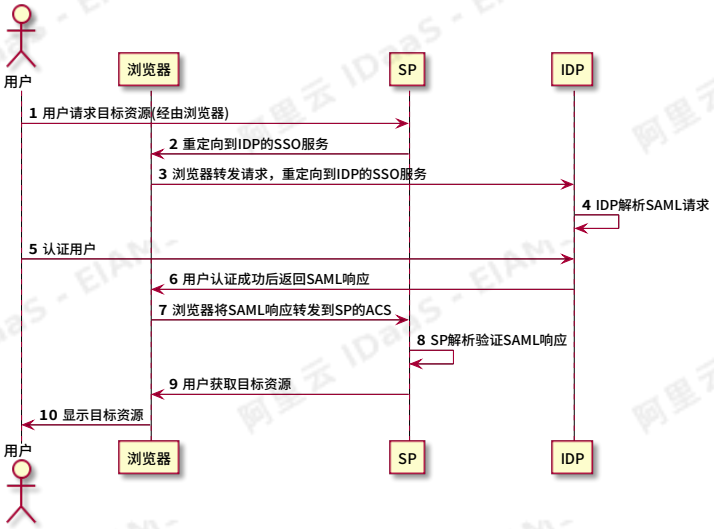
SAML（Security Assertion Markup Language 安全断言标记语言）是一个基于XML的开源标准数据格式，为在安全域间交换身份认证和授权数据，尤其是在IDP和SP之间。SAML是OASIS（Organization for the Advancement of Structured Information Standards 安全服务技术委员会）制定的标准，始于2001年，其最新主要版本SAML 2.0于2005年发布。

作为一种流行的SSO协议， SAML同时支持IDP发起和SP发起， 也就是可以在登录门户后，跳转到任意一个应用， 也可以从一个应用发起，跳转到IDP， 登录认证后，再跳转回这个应用， 继续SSO。 二者都是SSO， 流程的前半部分参数不同， 后半部分是很相似的。

2.1、SAML的流程

2.1.1、SP发起SSO

用户请求SP资源，SP生成SAML请求，IDP接收并解析SAML请求并进行用户认证后返回SAML响应，SP接收并解析SAML响应后，提取其中的令牌Assertion, 提供被请求的资源给用户使用。



具体流程如下：

2.1.1.1、用户请求目标资源

用户向SP请求目标资源，例如目标资源为：

https://sp.example.com/myresource

SP会进行安全检查，如果SP已经存在有效的IDP安全会话上下文，则认为已经登录过， 跳过步骤2~8。

2.1.1.2、重定向到IDP的SSO服务

SP会生成SAMLRequest，同时会把SP当前发起的URL生成一个随机数opaque， 临时存放， 同时把它作为RelayState，然后使用标准的HTTP 302重定向redirect到IDP的SSO服务，例如：

302 Redirect

Location: http://idp4/enduser/api/application/plugin_saml/<application_id>/sp_sso?SAMLRequest=xxx&RelayState=opaque

RelayState是SP的发起URL的不透明引用，SAMLRequest是Base64编码以后的<samlp:AuthnRequest>元素，<samlp:AuthnRequest>示例：

```
<samlp:AuthnRequest
  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  ID="identifier_1"
  Version="2.0"
  IssueInstant="2004-12-05T09:21:59Z"
  AssertionConsumerServiceIndex="0">
  <saml:Issuer>https://sp.example.com/SAML2</saml:Issuer>
  <samlp:NameIDPolicy
    AllowCreate="true"
    Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"/>
</samlp:AuthnRequest>
```

如果需要的话，SAMLRequest还可以使用SigningKey进行签名。

2.1.1.3、浏览器转发SAML请求，重定向到IDP的SSO服务

浏览器将SP的SAMLRequest和RelayState通过一个GET请求转发到IDP的SSO服务：

GET /SAML2/SSO/Redirect?SAMLRequest=request&RelayState=opaque HTTP/1.1

Host: idp.example.org

2.1.1.4、IDP解析SAML请求

IDP解析SAML请求，通过Base64解码得到<samlp:AuthnRequest>元素。IDP会验证用户是否已经登录，如果已经登录则跳过步骤5。

2.1.1.5、认证用户

IDP认证用户身份，常用的方法是IDP返回登录页面给用户，IDP可以配置自己需要的认证方式， 比如用户使用账号和密码进行登录认证。

2.1.1.6、用户认证成功后返回SAML响应

IDP认证用户身份以后会返回SAMLResponse响应，响应中包含如下表单：

```
<form method="post" action="https://sp.example.com/SAML2/SSO/POST" ...>
  <input type="hidden" name="SAMLResponse" value="response" />
  <input type="hidden" name="RelayState" value="opaque" />
  ...
  <input type="submit" value="Submit" />
</form>
```

表单中的RelayState参数值就是步骤2中生成的RelayState，IDP会将其原封不动的返回。表单中的SAMLResponse是Base64编码以后的<samlp:Response>元素，<samlp:Response>示例：

```
<samlp:Response
   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
   ID="identifier_2"
   InResponseTo="identifier_1"
   Version="2.0"
   IssueInstant="2004-12-05T09:22:05Z"
   Destination="https://sp.example.com/SAML2/SSO/POST">
   <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
   <samlp:Status>
     <samlp:StatusCode
       Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
   </samlp:Status>
   <saml:Assertion
     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
     ID="identifier_3"
     Version="2.0"
     IssueInstant="2004-12-05T09:22:05Z">
     <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
     <!-- a POSTed assertion MUST be signed -->
     <ds:Signature
       xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
     <saml:Subject>
       <saml:NameID
         Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">
         3f7b3dcf-1674-4ecd-92c8-1544f346baf8
       </saml:NameID>
       <saml:SubjectConfirmation
         Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
         <saml:SubjectConfirmationData
           InResponseTo="identifier_1"
           Recipient="https://sp.example.com/SAML2/SSO/POST"
           NotOnOrAfter="2004-12-05T09:27:05Z"/>
       </saml:SubjectConfirmation>
     </saml:Subject>
     <saml:Conditions
       NotBefore="2004-12-05T09:17:05Z"
       NotOnOrAfter="2004-12-05T09:27:05Z">
       <saml:AudienceRestriction>
         <saml:Audience>https://sp.example.com/SAML2</saml:Audience>
       </saml:AudienceRestriction>
     </saml:Conditions>
     <saml:AuthnStatement
       AuthnInstant="2004-12-05T09:22:00Z"
       SessionIndex="identifier_3">
       <saml:AuthnContext>
         <saml:AuthnContextClassRef>
           urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
        </saml:AuthnContextClassRef>
       </saml:AuthnContext>
     </saml:AuthnStatement>
   </saml:Assertion>
 </samlp:Response>
```

这里重要的是Assertion部分， 包含有用户的Subject 身份信息。默认一般用IDP的私钥对整个SAMLResponse 签名， 也可以是对Assertion 签名， 或是二者兼而有之，取决于IDP和SP的协商。

2.1.1.7、浏览器将SAML响应转发到SP的ACS

浏览器将SAMLResponse和RelayState以POST的方式转发到SP的ACS URL， SP继续解析令牌。

POST /SAML2/SSO/POST HTTP/1.1

Host: sp.example.com

Content-Type: application/x-www-form-urlencoded

Content-Length: nnn

SAMLResponse=response&RelayState=opaque

2.1.1.8、SP解析验证SAML响应

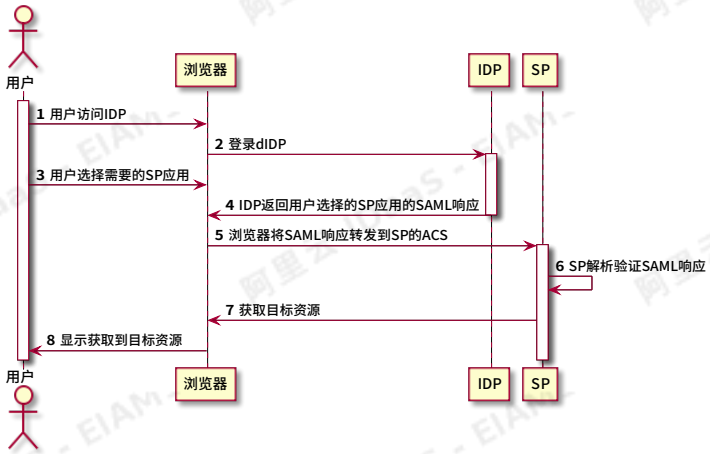
SP处理SAMLResponse响应，Base64解码得到<samlp:Response>元素，最重要的是要用SP中的公钥， 来检查签名的合法性， 如果合法 ，则抽取其中包含的用户信息Subject，找到对应的SP应用子账户， 生成SP安全会话上下文。

2.1.1.9、用户获取目标资源

用户成功获取SP提供的目标资源。如果SP发现RelayState中有对应的URL， 则提取这个URL， 跳转到对应的URL。

2.1.2、IDP发起SSO

同上面的SP发起SSO不同， IDP发起可以实现用户登录IDP，在IDP中选择某个SP应用，IDP跳转到SP，用户使用SP的资源。



具体的流程如下：

2.1.2.1、用户访问IDP

用户打开IDP的登录页面。

2.1.2.2、用户登录IDP

使用配置好的如账号密码等方式登录到IDP。

2.1.2.3、用户选择需要的SP应用

用户在IDP中选择需要使用的SP应用，背后会触发https://xxxx.login.aliyunidaas.com/api/bff/v1.2/enduser/portal/sso/go_0fbd26xxx? access_token=9a2e8d41-cde9-4ba9-b09b-yyyy，继续流程。

2.1.2.4、IDP返回用户选择的SP应用的SAML响应

IDP生成用户选择的SP应用的SAMLResponse响应（前文已介绍），返回给用户的浏览器。

2.1.2.5、浏览器将SAML响应转发到SP的ACS

浏览器将SAMLResponse和RelayState以POST的方式转发到SP的ACS URL。

2.1.2.6、SP解析验证SAML响应

SP处理SAMLResponse响应，Base64解码得到<samlp:Response>元素，最重要的是要用SP中的公钥， 来检查签名的合法性， 如果合法 ，则抽取其中包含的用户信息Subject，找到对应的SP应用子账户， 生成SP安全会话上下文。

注： 可以看到， 这一步和SP发起中的第8步非常类似， 包括下一步。

2.1.2.7、用户获取目标资源

自此，SSO结束，用户成功获取SP提供的目标资源。如果SP发现RelayState中有对应的URL， 则提取这个URL， 跳转到对应的URL。

2.1.2.8、显示目标资源

用户看到对应的应用目标资源。

2.2、SAML的Metadata

SAML协议中规定，IDP或SP的配置信息通过元数据（Metadata）信息实现，配置过程只要交换IDP和SP的元数据配置信息就可以快速实现SSO配置。

2.2.1、IDP的Metadata

IDP的Metada是<md:EntityDescriptor>元素，示例如下：

```
<md:EntityDescriptor entityID="https://idp.example.org/SAML2" validUntil="2013-03-22T23:00:00Z"
  xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <!-- insert ds:Signature element (omitted) -->
  <!-- insert md:IDPSSODescriptor element (below) -->
  <md:Organization>
    <md:OrganizationName xml:lang="en">Some Non-profit Organization of New York</md:OrganizationName>
    <md:OrganizationDisplayName xml:lang="en">Some Non-profit Organization</md:OrganizationDisplayName>
    <md:OrganizationURL xml:lang="en">https://www.example.org/</md:OrganizationURL>
  </md:Organization>
  <md:ContactPerson contactType="technical">
    <md:SurName>SAML Technical Support</md:SurName>
    <md:EmailAddress>mailto:a***@example.net</md:EmailAddress>
  </md:ContactPerson>
</md:EntityDescriptor>
```

主要元素信息为：

标签	说明

entityID	IDP的唯一标识。
validUtil	元数据的过期时间。
ds:Signature	包含数字签名，以确保元数据的真实性和完整性。
md:Organization	组织信息。
md:ContactPerson	联系人信息。

IDP的SSO相关Metadata是<md:IDPSSODescriptor>元素，示例如下：

```
<md:IDPSSODescriptor
  protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
  <md:KeyDescriptor use="signing">
    <ds:KeyInfo>...</ds:KeyInfo>
  </md:KeyDescriptor>
  <md:ArtifactResolutionService isDefault="true" index="0"
    Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
    Location="https://idp.example.org/SAML2/ArtifactResolution"/>
  <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</md:NameIDFormat>
  <md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient</md:NameIDFormat>
  <md:SingleSignOnService
    Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
    Location="https://idp.example.org/SAML2/SSO/Redirect"/>
  <md:SingleSignOnService
    Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
    Location="https://idp.example.org/SAML2/SSO/POST"/>
  <md:SingleSignOnService
    Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
    Location="https://idp.example.org/SAML2/Artifact"/>
  <saml:Attribute
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
    Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
    FriendlyName="eduPersonAffiliation">
    <saml:AttributeValue>member</saml:AttributeValue>
    <saml:AttributeValue>student</saml:AttributeValue>
    <saml:AttributeValue>faculty</saml:AttributeValue>
    <saml:AttributeValue>employee</saml:AttributeValue>
    <saml:AttributeValue>staff</saml:AttributeValue>
  </saml:Attribute>
</md:IDPSSODescriptor>
```

主要元素信息为：

标签	说明
<md:KeyDescriptor use="signing">	IDP配置的一个私有SAML签名密钥和/或一个私有后端通道TLS密钥。
<md:ArtifactResolutionService>下的Binding	SAML绑定信息。
<md:NameIDFormat>	SSO支持的SAML名称标识格式。
<md:SingleSignOnService>	单点登录信息。
<saml:Attribute>	IDP提供的断言的属性。

2.2.2、SP的Metadata

SP的Metada是<md:EntityDescriptor>元素，示例如下：

```
<md:EntityDescriptor entityID="https://sp.example.com/SAML2" validUntil="2013-03-22T23:00:00Z"
  xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <!-- insert ds:Signature element (omitted) -->
  <!-- insert md:SPSSODescriptor element (see below) -->
  <md:Organization>
    <md:OrganizationName xml:lang="en">Some Commercial Vendor of California</md:OrganizationName>
    <md:OrganizationDisplayName xml:lang="en">Some Commercial Vendor</md:OrganizationDisplayName>
    <md:OrganizationURL xml:lang="en">https://www.example.com/</md:OrganizationURL>
  </md:Organization>
  <md:ContactPerson contactType="technical">
    <md:SurName>SAML Technical Support</md:SurName>
    <md:EmailAddress>mailto:s****@example.com</md:EmailAddress>
  </md:ContactPerson>
</md:EntityDescriptor>
```

主要元素信息为：

标签	说明
entityID	SP的唯一标识。
validUtil	元数据的过期时间。
ds:Signature	包含数字签名，以确保元数据的真实性和完整性。
md:Organization	组织信息。
md:ContactPerson	联系人信息。

SP的ACS相关Metadata是<md:SPSSODescriptor>元素，示例如下：

```
<md:SPSSODescriptor
  protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
  <md:KeyDescriptor use="signing">
    <ds:KeyInfo>...</ds:KeyInfo>
  </md:KeyDescriptor>
  <md:KeyDescriptor use="encryption">
    <ds:KeyInfo>...</ds:KeyInfo>
  </md:KeyDescriptor>
  <md:ArtifactResolutionService isDefault="true" index="0"
    Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
    Location="https://sp.example.com/SAML2/ArtifactResolution"/>
  <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</md:NameIDFormat>
  <md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient</md:NameIDFormat>
  <md:AssertionConsumerService isDefault="true" index="0"
    Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
    Location="https://sp.example.com/SAML2/SSO/POST"/>
  <md:AssertionConsumerService index="1"
    Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
    Location="https://sp.example.com/SAML2/Artifact"/>
  <md:AttributeConsumingService isDefault="true" index="1">
    <md:ServiceName xml:lang="en">Service Provider Portal</md:ServiceName>
    <md:RequestedAttribute
      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
      Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
      FriendlyName="eduPersonAffiliation">
    </md:RequestedAttribute>
  </md:AttributeConsumingService>
</md:SPSSODescriptor>
```

标签	说明
<md:KeyDescriptor use="signing">	SP配置的一个私有SAML签名密钥和/或一个私有后端通道TLS密钥。
<md:KeyDescriptor use="encryption">	SP公共SAML加密密钥。
<md:AssertionConsumerService>下的index	<samlp:AuthnRequest>元素中的AssertionConsumerServiceIndex属性的值。

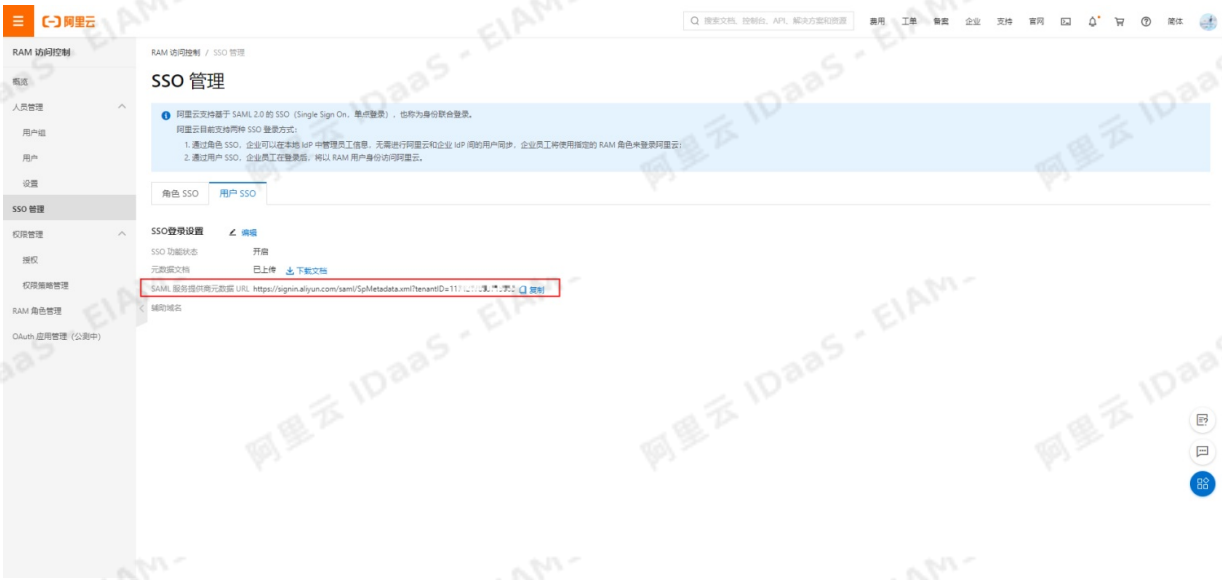
<md:AssertionConsumerService>下的Binding	SAML的绑定信息。
<md:AttributeConsumingService>	IDP用来构造一个<saml:AttributeStatement>元素，该元素与Web浏览器SSO一起推送到SP。
<md:AttributeConsumingService>下的index	SP在SSO时生成<samlp:AuthnRequest>元素中AttributeConsumingServiceIndex属性的值。

三、IDaaS中配置SAML应用示例

IDaaS平台支持基于标准SAML协议的SSO，并提供来一系列的应用模版来方便配置，这里以阿里云RAM为例演示如何配置。

3.1、获取SP的元数据信息

SP会提供自己的元数据信息，以阿里云RAM为例，登录控制台后找到元数据URL。



浏览器访问该URL得到元数据信息，显示如下：

将上述内容可以导出， 放在下一步使用。

3.2、IDaaS中配置SP的元数据信息

3.2.1、添加SP应用

准备好后， 接下来， 在IDaaS 中添加一个RAM应用。

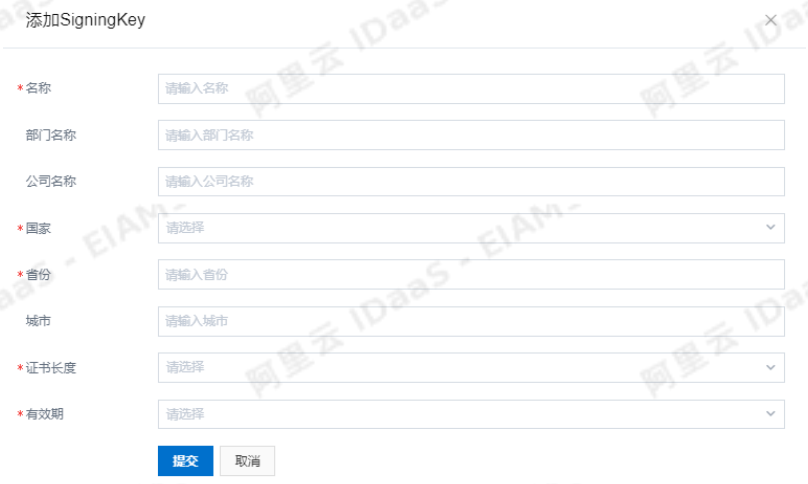
以IT管理员账号登录云盾IDaaS管理平台，具体操作请参考 IT管理员指南-登录 。

点击左侧导航栏 **应用 > 添加应用**

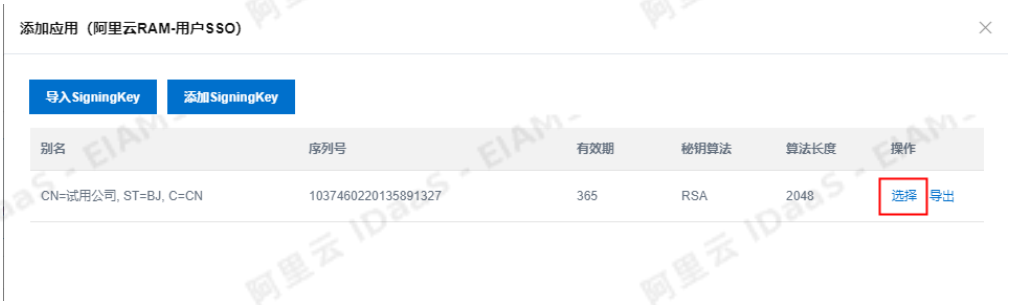
在右侧选择一个SAML应用，点击添加应用。IDaaS支持多种SAML应用，这里以添加阿里云RAM-用户SSO为例进行展示。

点击添加SigningKey按钮，输入名称等信息，系统会据此生成应用的证书，私钥保留在IDP，公钥导出到SP， 用于IDP和SP通信的签名验签。

如果没有现成的证书可以选择， 则填写以下信息生成一个，其中的名称信息最好是和这个应用比如RAM关联的， 方便将来识别。



无论是选择已有的还是刚添加的，找到对应的SigningKey，选择它。



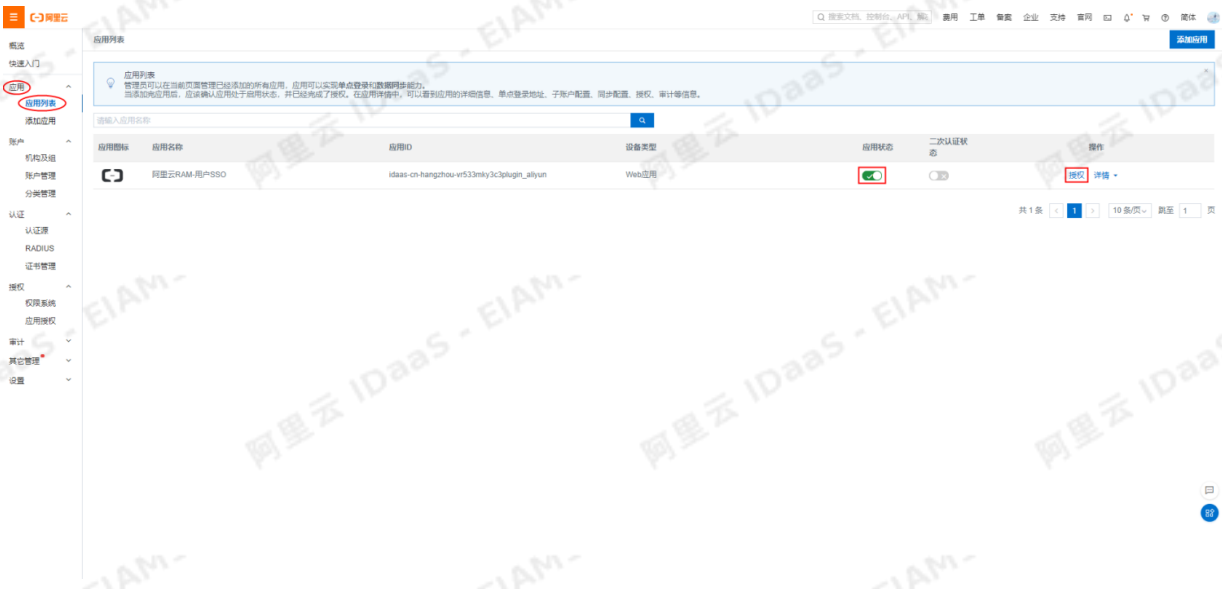
接下来要填写更多的应用信息，名称等信息可以自定义，EntityId、ACS URL等信息从步骤1中的到的SP的元数据中复制过来，需要填写的主要信息如下：

参数名称	说明
应用名称	所添加应用的名称，可以为任意值，但最好和应用相关。
应用类型	引用的类型，只有选中的应用类型才会在用户对应客户端中显示。
IDaaS EntityId	在IDaaS中设置的认证参数，需要将此参数配置到SP中，在IDaaS导出的 metadata 里可以获取，例如 https://signin.aliyun.com/117xxxxxxxxxx63/saml/SSO。
SP Entity ID	在SP中设置的Entity ID，需要复制到IDaaS的配置中， 可以在RAM的metadata中获取，例如https://signin.aliyun.com/117yyyyyyyyyy63/saml/SSO。
SP ACS URL（SSO Location）	单点登录地址，这里以阿里云RAM为例：https://signin.aliyun.com/saml/SSO。
NameIdFormat	名称标识格式类型，这里以阿里云RAM为例，选择urn:oasis:names:tc:SAML:2.0:nameid-format:persistent。

填写完成后提交保存， 如果应用是禁用状态， 可以继续修改重新提交。

3.2.2、启用应用并且授权

应用配置好以后需要先启用应用，并且将服务授权给一个账户，点击左侧导航栏 **应用 > 应用列表** 启用该应用并授权给账户。



IDaaS支持多种方式进行授权，这里以按应用授权账户为例。

保存后， 这个用户登录就可以看到这个应用了。

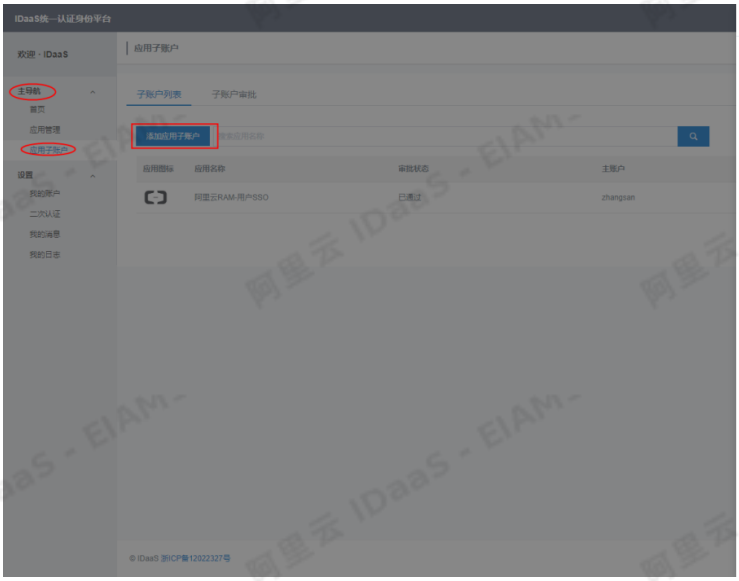
3.2.3、IDP新建子账户（非必须步骤）

一个系统要SSO到另外一个系统，需要使用对方能够识别的子账号进行认证，往往登录到IDP的主账户和应用SP的子账户是不一样的，可以使用账号同步（两套系统中的账号信息相同）或者新建子账户进行账号映射的方法。账号映射是指给IDP的账户建立一个SP中已经存在的账户作为子账户，身份认证的时候通过子账户进行认证。例如SP系统中有个账户“demo”，我们想用IDP系统中的“zhangsan”账号SSO到SP，则需要给账号“zhangsan”新建一个对应的子账户“demo”。这里以阿里云RAM演示新建子账户的功能，如下图，阿里云RAM中有账户demo@117yyyyy****yy63.onaliyun.com。

IDaaS中新建子账户有两种方式，操作如下：

3.2.3.1、授权账号新建子账户

登录授权账户，点击左侧导航栏 **主导航** > **应用子账户** 添加应用子账户功能中提交新建子账户申请。由于上一步阿里云RAM中的账户是demo@117yyyyy****yy63.onaliyun.com，所以这里子账户的名称应该填demo。IDaaS在SSO的时候，会将子账户（demo）和步骤3.2.1中配置的阿里云个人域名（117yyyyy****yy63.onaliyun.com）进行拼接映射到阿里云RAM的账户。

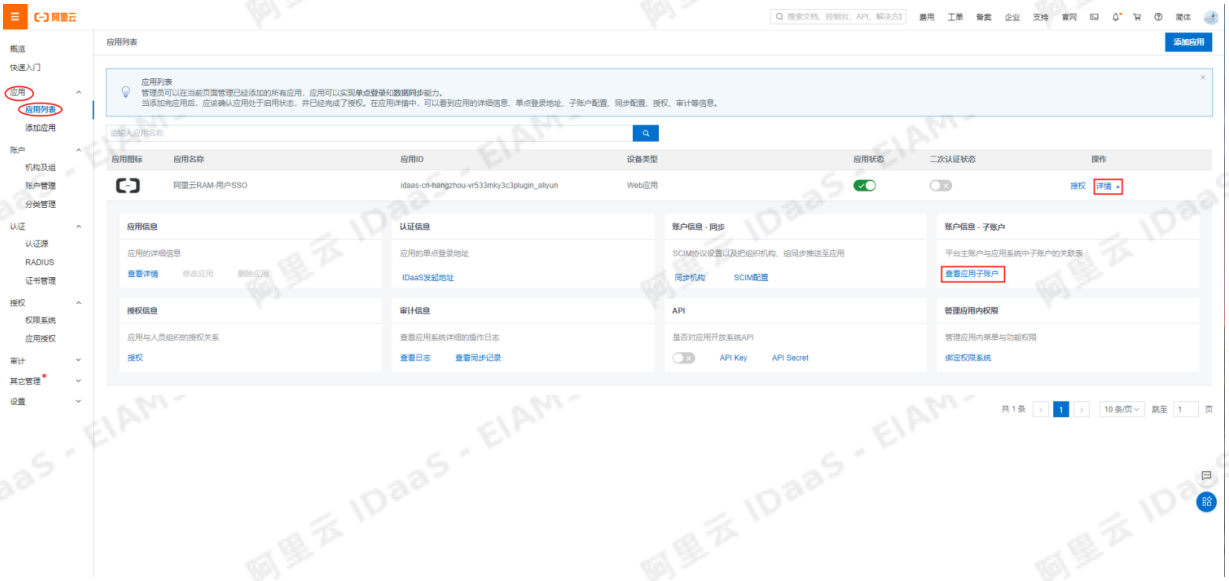


登录管理员账户，点击左侧导航栏 **其它管理** > **审批中心** 审核通过该应用子账户的添加。

3.2.3.2、管理员新建子账户

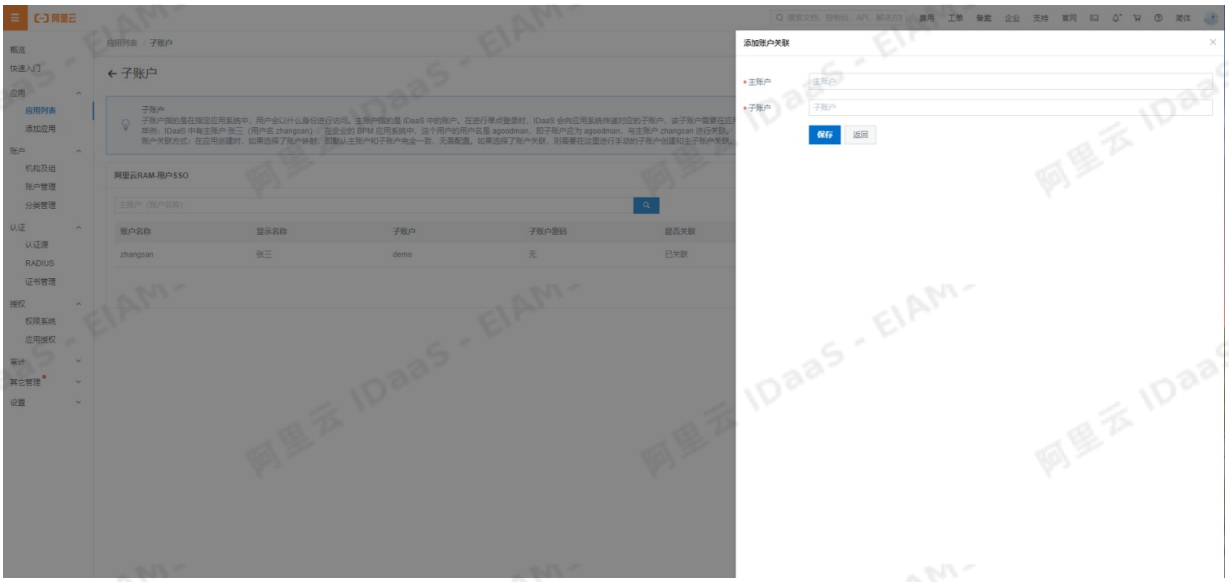
管理员新建子账户不需要审核过程，具体操作为：

登录管理员账户，点击左侧导航栏 **应用** > **应用列表** 找到添加的应用，点击详情中的查看应用子账户。



点击添加账户关联，添加子账户。

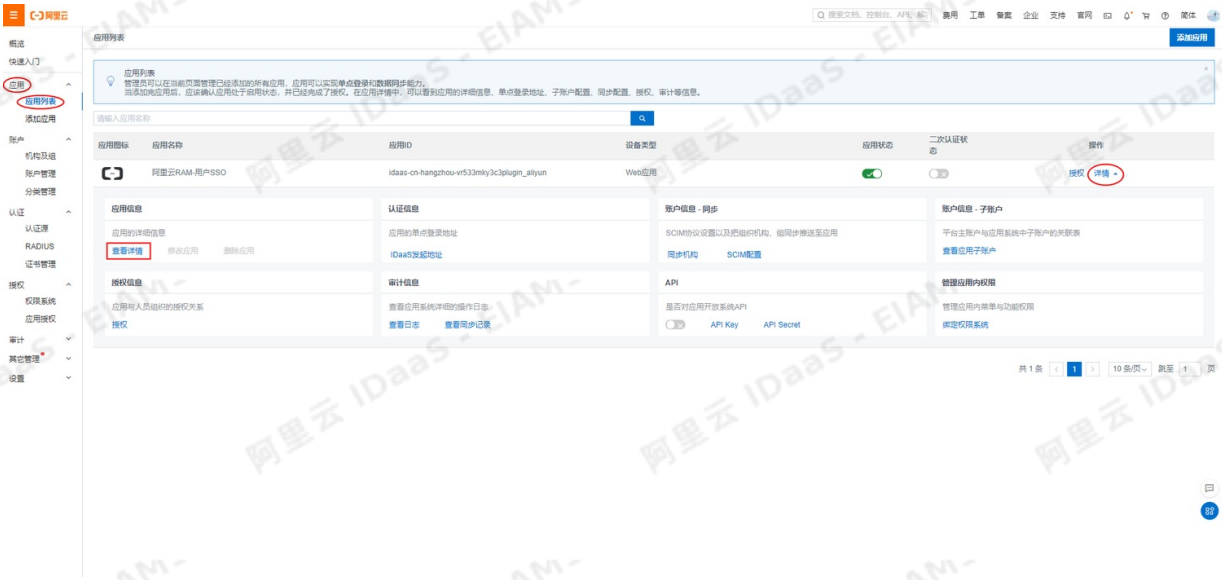
输入授权账户（主账户）和子账户，点击保存完成子账户添加。



3.3、SP中配置IDaaS的元数据信息

3.3.1、获取IDaaS的元数据信息

以IT管理员账号登录云盾IDaaS管理平台，点击左侧导航栏 **应用** > **应用列表** 选择刚才添加的应用，点击查看详情，如下图：



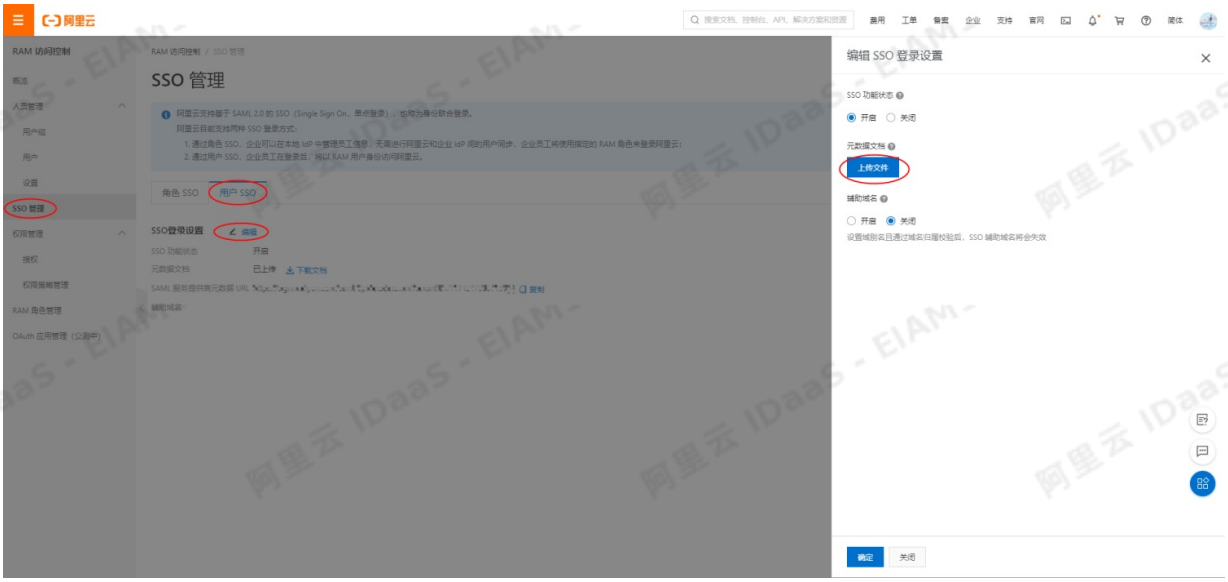
点击导出SAML元配置文件，将IDaaS的元数据文件保存到本地电脑。

IDaaS元配置文件示例如下：

```
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" entityID="https://signin.aliyun.com/117xxxxxxxxxx63/saml/SSO">
<md:IDPSSODescriptor WantAuthnRequestsSigned="false" protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
<md:KeyDescriptor use="signing">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
<ds:X509Certificate>TIIC4jCCAc****IBAgIIDmXMktHMYX8wDQYJKoZIhvcNAQEFBQAwMTELMAkGA1UEBhMCQ04xCzAJBgNVBAgTAkJKMRUwEwYDVQQDDAzor5XnlKjlhazlj7gwHhc
NMjAxMjA3MDMwNTU3WhcNMjExMjA3MDMwNTU3WjAxMQswCQYDVQQGEwJDTjELMAkGA1UECBMCQkoxFTATBgNVBAMMDOivleeUqOWFrOWPuDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQ
oCggEBALiuwcn+sbMaVRT2Byb3GzBV1P0eOoK326fQS9rdeaGaIykSMqCMMKOZ+/QfdMWh+9Fr59A5pIEbCN7aP3P+cV1ClqhfKD4DTbsmGikSiUYgYf4tWztZx9NFWyuoucm8LOKKpKlPb
jUyLudzLlQGOCrX/4Be0md4mIVZMK96J41jRuJXUTxFepE0cTEi15SXbEsXrnJ1wueFylNKl9JerbCJlEDayktAYvkMrmn2d2R2etiVR4Una9pBqtPvCElPKCNesWAE/3AcWTgSj+u8ocnT
gnknIfVO65QRxaNrDAyTOpkquXFshs+DtlILEdk2p9UUxkUCNySbIIM/gVgL0TkCAwEAATANBgkqhkiG9w0BAQUFAAOCAQEAeDdH7BYcalDYfpNKvJrHA9rKZ2vBTi4uy2WoXclE0EdzBGU
C4loPL5g3ictNA+4S8tnCkzl8aQr79tjUmcL/0Uzv4sdOggwglmkgw3kek9Yq44i/ycMN8HVeF/vtyVxhlvqBeXU2P5n6jFqatG+VkeVGyiJQHwuPlUHokXWwyukcjr35CQQX5WALFNJ+F6
8ICKT9Ulqb5GtQgrdlJoRQBlEb//IjxlZJAvZ6CxLnVCgVUSOI4xYEb8ATZPbzLIMIyXN4U6r6VxvBJuW/eMcqogYSYssbngSgpHmZFV9+MrDSjJLLtsVRuzmF+cBisojvo53z3EiNu/c4F
GlUuKozPA==</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:KeyDescriptor use="encryption">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
<ds:X509Certificate>TIIC4jCCAc****IBAgIIDmXMktHMYX8wDQYJKoZIhvcNAQEFBQAwMTELMAkGA1UEBhMCQ04xCzAJBgNVBAgTAkJKMRUwEwYDVQQDDAzor5XnlKjlhazlj7gwHhc
NMjAxMjA3MDMwNTU3WhcNMjExMjA3MDMwNTU3WjAxMQswCQYDVQQGEwJDTjELMAkGA1UECBMCQkoxFTATBgNVBAMMDOivleeUqOWFrOWPuDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQ
oCggEBALiuwcn+sbMaVRT2Byb3GzBV1P0eOoK326fQS9rdeaGaIykSMqCMMKOZ+/QfdMWh+9Fr59A5pIEbCN7aP3P+cV1ClqhfKD4DTbsmGikSiUYgYf4tWztZx9NFWyuoucm8LOKKpKlPb
jUyLudzLlQGOCrX/4Be0md4mIVZMK96J41jRuJXUTxFepE0cTEi15SXbEsXrnJ1wueFylNKl9JerbCJlEDayktAYvkMrmn2d2R2etiVR4Una9pBqtPvCElPKCNesWAE/3AcWTgSj+u8ocnT
gnknIfVO65QRxaNrDAyTOpkquXFshs+DtlILEdk2p9UUxkUCNySbIIM/gVgL0TkCAwEAATANBgkqhkiG9w0BAQUFAAOCAQEAeDdH7BYcalDYfpNKvJrHA9rKZ2vBTi4uy2WoXclE0EdzBGU
C4loPL5g3ictNA+4S8tnCkzl8aQr79tjUmcL/0Uzv4sdOggwglmkgw3kek9Yq44i/ycMN8HVeF/vtyVxhlvqBeXU2P5n6jFqatG+VkeVGyiJQHwuPlUHokXWwyukcjr35CQQX5WALFNJ+F6
8ICKT9Ulqb5GtQgrdlJoRQBlEb//IjxlZJAvZ6CxLnVCgVUSOI4xYEb8ATZPbzLIMIyXN4U6r6VxvBJuW/eMcqogYSYssbngSgpHmZFV9+MrDSjJLLtsVRuzmF+cBisojvo53z3EiNu/c4F
GlUuKozPA==</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location="https://lidcfkpjfb.login.aliyunidaas.com/enduser
/api/application/plugin_aliyun/idaas-cn-hangzhou-vr533mky3c3plugin_aliyun/sp_sso"/>
<md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="https://lidcfkpjfb.login.aliyunidaas.com/enduser/api
/application/plugin_aliyun/idaas-cn-hangzhou-vr533mky3c3plugin_aliyun/sp_sso_post"/>
</md:IDPSSODescriptor>
</md:EntityDescriptor>
```


3.3.2、SP中配置元数据信息

不同的SP配置IDP的元数据方式不同，有的需要填入参数，有的可以直接上传元数据文件。以阿里云RAM为例，在阿里云RAM选择开启SSO功能，并且上传刚刚下载的元配置文件，完成SP中的IDP元数据信息配置。



3.4、功能演示

3.4.1、IDP发起SSO

配置完成后， 就可以检查结果了。 授权用户登录IDaaS，点击左侧导航栏 **主导航 > 首页** 在我的应用中点击该应用进行单点登录，点击应用的图标进行单点登录。

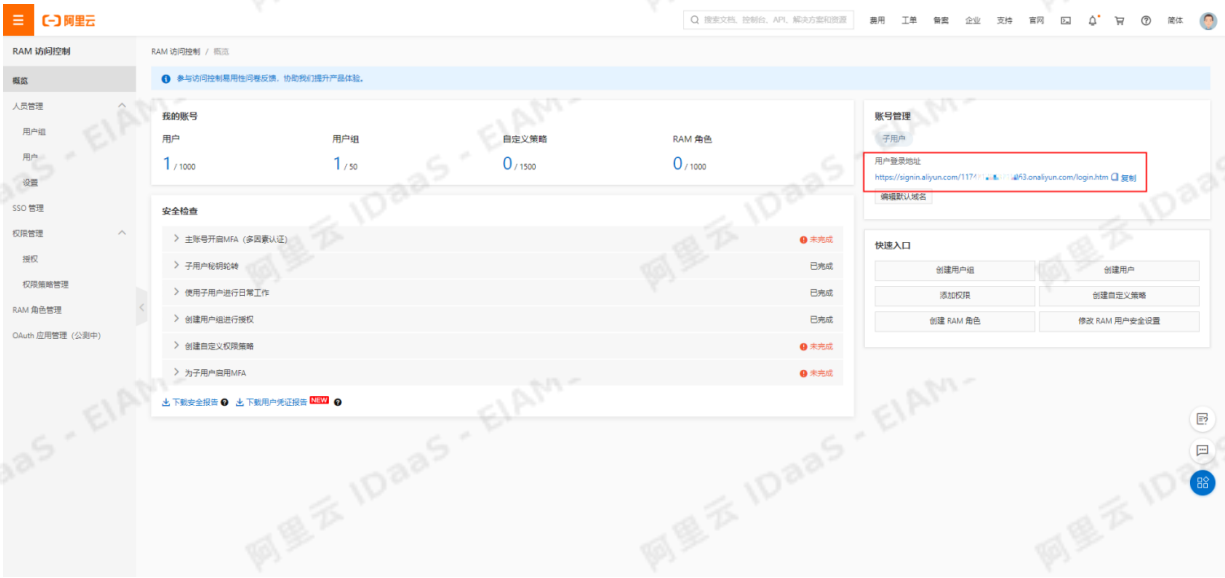
选择子账户demo进行单点登录。

成功登录阿里云RAM控制台，然后就可以看到阿里云作为SP提供的资源了。

如果账号配置错误或者选择的登录账号不是阿里云RAM中的账户，则会提示账户不存在。

3.4.2、SP发起SSO

同样， 正确配置后， 也支持SP发起， 首先找到阿里云RAM子账户登录地址。



贴到浏览器跳转后， 登录界面上可以看到“主账号登录”和“使用企业账号登录”两种选择。主账号登录是使用阿里云RAM自己的账号和密码进行登录，点击使用企业账号登录，则开始进行IDaaS的SSO过程。

浏览器会自动跳转到IDP的登录界面，登录IDaaS授权账号，例如zhangsan，然后IDaaS认证完成以后，找到对应的子账号，生成SAMLResponse， 就会跳转到阿里云RAM控制台。

自此， IDP发起和SP发起全部工作正常！

四、FAQ

4.1、代码中如何解析SAMLRequest

SP发起SSO的时候会生成SAMLRequest，SAMLRequest是Base64编码后的内容，我们需要解析以后才能得到需要的内容，如下代码可以解析SAMLRequest，然后就可以拿到AuthnRequest进行认证。

```
import java.io.*;
import org.opensaml.xml.util.Base64;
import java.util.zip.InflaterInputStream;
import java.util.zip.Inflater;
public class SamlRequestTest {
    public static void main(String[] args) throws Exception {
        // 接收到的原始SAMLRequest
        String samlRequest = "fZJNT%2BMwEIbv%2Bysi3%2FNhd9umVhPUXYQWiRUVCRy4IMedFIMzzmacavvvCQll4bAcfLD0fnjm8frsb2ODA3RkHGaMRwkLALXbGdxn7La8CFN2ln9bk2qsaOWm9494A396IB9siKDzg%2B%2BnQ%2Bob6AroDkbD7c1Vxh69b0nGMZk9GoyUNcceI%2B2a%2BDUqLoprFpwPKQaVH6tPBmyttmiPhyqybv9uNTulaPQD7vqhOFatGU5rjR4T4tb2g%2FxhksejPtQYVmCehmHC2h3h%2BETPz%2B3yk5LH1D4QORZcuE7DOGHGamUJWHB5njElaljCTOxTriuuVlzVS6iTNIVK8F01iGiriMwB%2FtmIerhE8gp9xkQikpCLMElLzuV8JvkqWqTf71mw7Zx32tkfBqeF9x1Kp8iQRNUASa9lsfl9JUWUyGoSkfxVlttwe12ULLg7gROv4AaUSHJC9XVW%2B1bM8omsHF%2FcfUz4OkCd2LP8%2F6R5ukgWq8V8NpsnKyEW7%2BjX8cfW%2FO36%2BXvlLw%3D%3D&RelayState=https%3A%2F%2Fhomenew.console.aliyun.com%2Fhome%2Fscene%2FOperation";
        // base64解码
        byte[] decodedBytes = Base64.decode(java.net.URLDecoder.decode(samlRequest, "utf-8"));
        // 获取输入流
        ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(decodedBytes);
        InflaterInputStream inflaterInputStream = new InflaterInputStream(byteArrayInputStream, new Inflater(true));
        byte[] buffer = new byte[decodedBytes.length];
        ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
        // 信息写到输出流
        for (int i = 0; i != -1; i = inflaterInputStream.read(buffer)) {
            byteArrayOutputStream.write(buffer, 0, i);
        }
        String result = new String(byteArrayOutputStream.toByteArray(), "UTF-8");
        // 输出解析后结果
        System.out.println(result);
    }
}
```

解析后的结果为

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:AuthnRequest AssertionConsumerServiceURL="https://signin.aliyun.com/saml/SSO" Destination="https://nplclnlyvb.login.aliyunidaas.com/enduser/api/application/plugin_aliyun/idaas-cn-beijing-foyeyjskkp7plugin_aliyun1/sp_sso" ForceAuthn="false" ID="a2fe7e32g81cbla91af7ef088eb21db" IsPassive="false" IssueInstant="2020-12-08T11:53:19.684Z" ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Version="2.0" xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"><saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">https://signin.aliyun.com/1860696533509226/saml/SSO</saml2:Issuer></saml2p:AuthnRequest>
```

2.3. SAML 模板使用指南

概述

IDaaS平台支持基于标准SAML协议的SSO（Single Sign On 单点登录），IDaaS作为SAML协议中的IDP（Identity Provider身份提供方）角色，提供用户的身份认证服务，用户可以登录一次就直接使用多个SP（Service Provider 业务提供方）的服务，免去了每个应用都要登录的烦恼。

SAML介绍

SAML（Security Assertion Markup Language 安全断言标记语言）是一个基于XML的开源标准数据格式，为在安全域间交换身份认证和授权数据，尤其是在IDP和SP之间。SAML是OASIS（Organization for the Advancement of Structured Information Standards 安全服务技术委员会）制定的标准，始于2001年，其最新主要版本SAML 2.0于2005年发布。

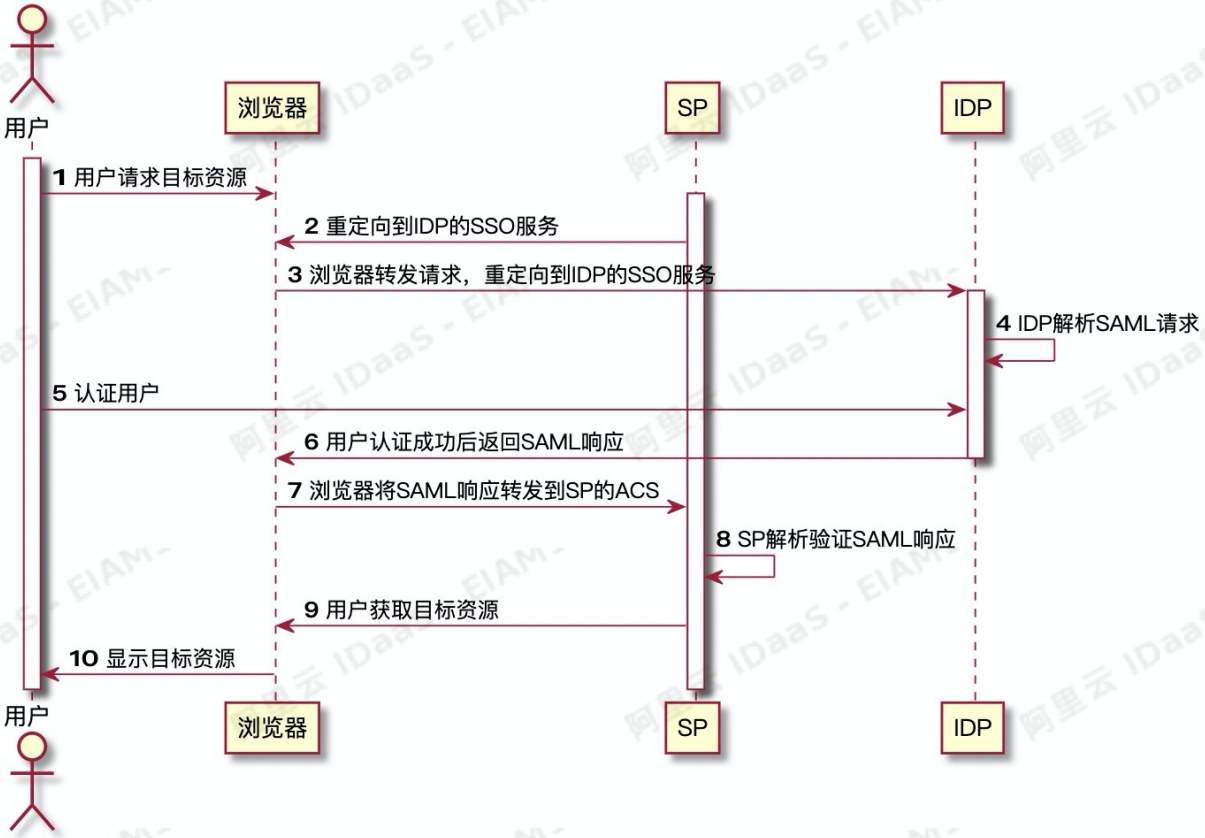
IDP发起和SP发起

作为一种流行的SSO协议， SAML同时支持IDP发起和SP发起， 也就是可以在登录门户后，跳转到任意一个应用， 也可以从一个应用发起，跳转到IDP， 登录认证后，再跳转回这个应用， 继续SSO。 二者都是SSO， 流程的前半部分参数不同， 后半部分是很相似的。

SAML的流程

SP发起SSO

用户请求SP资源，SP生成SAML请求，IDP接收并解析SAML请求并进行用户认证后返回SAML响应，SP接收并解析SAML响应后提供被请求的资源给用户使用。



1、用户请求目标资源

2、重定向到IDP的SSO服务

用户向SP请求目标资源，例如目标资源为：

https://sp.example.com/myresource

SP会进行安全检查，如果SP已经存在有效的安全上下文，则跳过步骤2~8。

SP会生成SAMLRequest，如果需要还会携带RelayState，然后使用标准的HTTP 302重定向到IDP的SSO服务，例如：

302 Redirect

Location: https://idp.example.org/SAML2/SSO/Redirect?SAMLRequest=request&RelayState=token

RelayState是SP的对状态信息的不透明引用，SAMLRequest是Base64编码以后的<samlp:AuthnRequest>元素，<samlp:AuthnRequest>示例：

```
<samlp:AuthnRequest
    xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    ID="identifier_1"
    Version="2.0"
    IssueInstant="2004-12-05T09:21:59Z"
    AssertionConsumerServiceIndex="0">
    <saml:Issuer>https://sp.example.com/SAML2</saml:Issuer>
    <samlp:NameIDPolicy
      AllowCreate="true"
      Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"/>
  </samlp:AuthnRequest>
```

如果需要的话，SAMLRequest还可以使用SigningKey进行签名。

3、浏览器转发SAML请求，重定向到IDP的SSO服务

浏览器将SP的SMALRequest和RelayState通过一个GET请求转发到IDP的SSO服务：

GET /SAML2/SSO/Redirect?SAMLRequest=request&RelayState=token HTTP/1.1

Host: idp.example.org

4、IDP解析SAML请求

IDP解析SAML请求，通过Base64解码得到<samlp:AuthnRequest>元素。IDP会验证用户是否已经登录，如果已经登录则跳过步骤5。

5、认证用户

IDP认证用户身份，常用的方法是IDP返回登录页面给用户，用户使用账号和密码进行登录认证。

6、用户认证成功后返回SAML响应

IDP认证用户身份以后会返回SMAL响应，响应中包含如下表单：

```
<form method="post" action="https://sp.example.com/SAML2/SSO/POST" ...>
    <input type="hidden" name="SAMLResponse" value="response" />
    <input type="hidden" name="RelayState" value="token" />
    ...
    <input type="submit" value="Submit" />
  </form>
```

表单中的RelayState参数值就是步骤2中生成的RelayState，IDP会将其原封不动的返回。表单中的SAMLResponse是Base64编码以后的<samlp:Response>元素，<samlp:Response>示例：

```
<samlp:Response
   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
   ID="identifier_2"
   InResponseTo="identifier_1"
   Version="2.0"
   IssueInstant="2004-12-05T09:22:05Z"
   Destination="https://sp.example.com/SAML2/SSO/POST">
   <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
   <samlp:Status>
     <samlp:StatusCode
       Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
   </samlp:Status>
   <saml:Assertion
     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
     ID="identifier_3"
     Version="2.0"
     IssueInstant="2004-12-05T09:22:05Z">
     <saml:Issuer>https://idp.example.org/SAML2</saml:Issuer>
     <!-- a POSTed assertion MUST be signed -->
     <ds:Signature
       xmlns:ds="http://www.w3.org/2000/09/xmldsig#">...</ds:Signature>
     <saml:Subject>
       <saml:NameID
         Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient">
         3f7b3dcf-1674-4ecd-92c8-1544f346baf8
       </saml:NameID>
       <saml:SubjectConfirmation
         Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
         <saml:SubjectConfirmationData
           InResponseTo="identifier_1"
           Recipient="https://sp.example.com/SAML2/SSO/POST"
           NotOnOrAfter="2004-12-05T09:27:05Z"/>
       </saml:SubjectConfirmation>
     </saml:Subject>
     <saml:Conditions
       NotBefore="2004-12-05T09:17:05Z"
       NotOnOrAfter="2004-12-05T09:27:05Z">
       <saml:AudienceRestriction>
         <saml:Audience>https://sp.example.com/SAML2</saml:Audience>
       </saml:AudienceRestriction>
     </saml:Conditions>
     <saml:AuthnStatement
       AuthnInstant="2004-12-05T09:22:00Z"
       SessionIndex="identifier_3">
       <saml:AuthnContext>
         <saml:AuthnContextClassRef>
           urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
        </saml:AuthnContextClassRef>
       </saml:AuthnContext>
     </saml:AuthnStatement>
   </saml:Assertion>
 </samlp:Response>
```

7、浏览器将SAML响应转发到SP的ACS

浏览器将SAMLResponse和RelayState以POST的方式转发到SP的ACS

POST /SAML2/SSO/POST HTTP/1.1

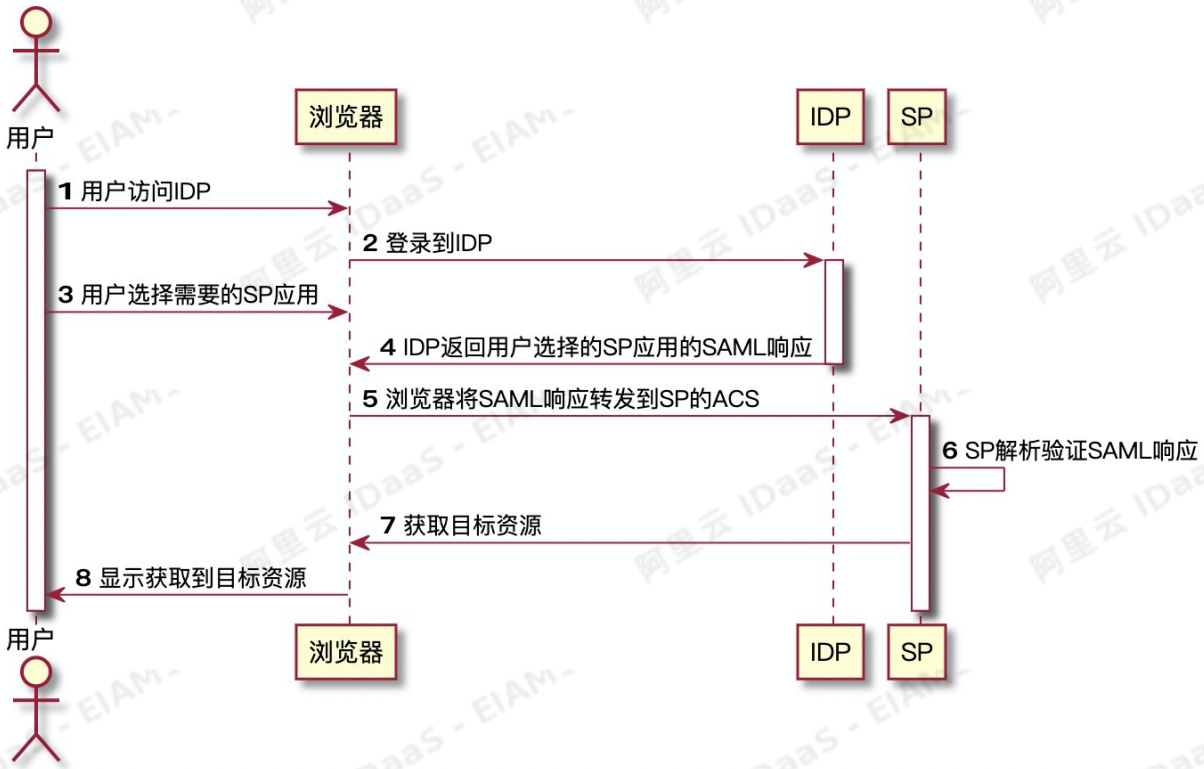
Host: sp.example.com

Content-Type: application/x-www-form-urlencoded

Content-Length: nnn

SAMLResponse=response&RelayState=token

8、SP解析验证SAML响应
SP处理SAML响应，Base64解码得到<samlp:Response>元素，生成安全上下文。
9、用户获取目标资源
用户成功获取SP提供的目标资源。
IDP发起SSO
用户登录IDP，在IDP中选择SP应用，IDP跳转到SP，用户使用SP的资源。



具体的流程如下：

1、用户访问IDP

用户打开IDP的登录页面。

2、用户登录IDP

使用配置好的如账号密码等方式登录到IDP。

3、用户选择需要的SP应用

用户在IDP中选择需要使用的SP应用，背后会触发https://xxxx.login.aliyunidaas.com/api/bff/v1.2/enduser/portal/sso/go_0fbd26xxx?access_token=9a2e8d41-cde9-4ba9-b09b-yyyy，继续流程

4、IDP返回用户选择的SP应用的SAML响应

IDP生成用户选择的SP应用的SAMLResponse响应（前文已介绍），返回给用户的浏览器。

5、浏览器将SAML响应转发到SP的ACS

浏览器将SAMLResponse和RelayState以POST的方式转发到SP的ACS URL。

6、SP解析验证SAML响应

SP处理SAMLResponse响应，Base64解码得到<samlp:Response>元素，最重要的是要用SP中的公钥， 来检查签名的合法性， 如果合法 ，则抽取其中包含的用户信息Subject，找到对应的SP应用子账户， 生成SP安全会话上下文。

注： 可以看到， 这一步和SP发起中的第8步非常类似， 包括下一步。

7、用户获取目标资源

自此，SSO结束，用户成功获取SP提供的目标资源。如果SP发现RelayState中有对应的URL， 则提取这个URL， 跳转到对应的URL。

8、显示目标资源

用户看到对应的应用目标资源。

SAML的Metadata

SAML协议中规定，配置信息通过元数据（Metadata）信息实现，配置过程只要交换IDP和SP的元数据配置信息就可以实现SSO功能。

IDP的Metadata

IDP的Metada是<md:EntityDescriptor>元素，示例如下：

```
<md:EntityDescriptor entityID="https://idp.example.org/SAML2" validUntil="2013-03-22T23:00:00Z"
  xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <!-- insert ds:Signature element (omitted) -->
  <!-- insert md:IDPSSODescriptor element (below) -->
  <md:Organization>
    <md:OrganizationName xml:lang="en">Some Non-profit Organization of New York</md:OrganizationName>
    <md:OrganizationDisplayName xml:lang="en">Some Non-profit Organization</md:OrganizationDisplayName>
    <md:OrganizationURL xml:lang="en">https://www.example.org/</md:OrganizationURL>
  </md:Organization>
  <md:ContactPerson contactType="technical">
    <md:SurName>SAML Technical Support</md:SurName>
    <md:EmailAddress>mailto:saml-support@example.org</md:EmailAddress>
  </md:ContactPerson>
</md:EntityDescriptor>
```

主要元素信息为：

标签	说明
entityID	IDP的唯一标识
validUtil	元数据的过期时间
ds:Signature	包含数字签名，以确保元数据的真实性和完整性
md:Organization	组织信息
md:ContactPerson	联系人信息

IDP的SSO相关Metadata是<md:IDPSSODescriptor>元素，示例如下：

```
<md:IDPSSODescriptor
  protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
  <md:KeyDescriptor use="signing">
    <ds:KeyInfo>...</ds:KeyInfo>
  </md:KeyDescriptor>
  <md:ArtifactResolutionService isDefault="true" index="0"
    Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
    Location="https://idp.example.org/SAML2/ArtifactResolution"/>
  <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</md:NameIDFormat>
  <md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient</md:NameIDFormat>
  <md:SingleSignOnService
    Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
    Location="https://idp.example.org/SAML2/SSO/Redirect"/>
  <md:SingleSignOnService
    Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
    Location="https://idp.example.org/SAML2/SSO/POST"/>
  <md:SingleSignOnService
    Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
    Location="https://idp.example.org/SAML2/Artifact"/>
  <saml:Attribute
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
    Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
    FriendlyName="eduPersonAffiliation">
    <saml:AttributeValue>member</saml:AttributeValue>
    <saml:AttributeValue>student</saml:AttributeValue>
    <saml:AttributeValue>faculty</saml:AttributeValue>
    <saml:AttributeValue>employee</saml:AttributeValue>
    <saml:AttributeValue>staff</saml:AttributeValue>
  </saml:Attribute>
</md:IDPSSODescriptor>
```

主要元素信息为：

标签	说明
<md:KeyDescriptor use="signing">	IDP配置的一个私有SAML签名密钥和/或一个私有后端通道TLS密钥
<md:ArtifactResolutionService>下的Binding	SAML绑定信息

标签	说明
<md:NameIDFormat>	SSO支持的SAML名称标识格式
<md:SingleSignOnService>	单点登录信息
<saml:Attribute>	IDP提供的断言的属性

SP的Metadata

IDP的Metada是<md:EntityDescriptor>元素，示例如下：

```
<md:EntityDescriptor entityID="https://sp.example.com/SAML2" validUntil="2013-03-22T23:00:00Z"
  xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
  xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <!-- insert ds:Signature element (omitted) -->
  <!-- insert md:SPSSODescriptor element (see below) -->
  <md:Organization>
    <md:OrganizationName xml:lang="en">Some Commercial Vendor of California</md:OrganizationName>
    <md:OrganizationDisplayName xml:lang="en">Some Commercial Vendor</md:OrganizationDisplayName>
    <md:OrganizationURL xml:lang="en">https://www.example.com/</md:OrganizationURL>
  </md:Organization>
  <md:ContactPerson contactType="technical">
    <md:SurName>SAML Technical Support</md:SurName>
    <md:EmailAddress>mailto:saml-support@example.com</md:EmailAddress>
  </md:ContactPerson>
</md:EntityDescriptor>
```

主要元素信息为：

标签	说明
entityID	SP的唯一标识
validUtil	元数据的过期时间
ds:Signature	包含数字签名，以确保元数据的真实性和完整性
md:Organization	组织信息
md:ContactPerson	联系人信息

SP的ACS相关Metadata是<md:SPSSODescriptor>元素，示例如下：

```
<md:SPSSODescriptor
  protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
  <md:KeyDescriptor use="signing">
    <ds:KeyInfo>...</ds:KeyInfo>
  </md:KeyDescriptor>
  <md:KeyDescriptor use="encryption">
    <ds:KeyInfo>...</ds:KeyInfo>
  </md:KeyDescriptor>
  <md:ArtifactResolutionService isDefault="true" index="0"
    Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
    Location="https://sp.example.com/SAML2/ArtifactResolution"/>
  <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</md:NameIDFormat>
  <md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient</md:NameIDFormat>
  <md:AssertionConsumerService isDefault="true" index="0"
    Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
    Location="https://sp.example.com/SAML2/SSO/POST"/>
  <md:AssertionConsumerService index="1"
    Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
    Location="https://sp.example.com/SAML2/Artifact"/>
  <md:AttributeConsumingService isDefault="true" index="1">
    <md:ServiceName xml:lang="en">Service Provider Portal</md:ServiceName>
    <md:RequestedAttribute
      NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
      Name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
      FriendlyName="eduPersonAffiliation">
    </md:RequestedAttribute>
  </md:AttributeConsumingService>
</md:SPSSODescriptor>
```

标签	说明
<md:KeyDescriptor use="signing">	SP配置的一个私有SAML签名密钥和/或一个私有后端通道TLS密钥
<md:KeyDescriptor use="encryption">	SP公共SAML加密密钥
<md:AssertionConsumerService>下的index	<samlp:AuthnRequest>元素中的AssertionConsumerServiceIndex属性的值
<md:AssertionConsumerService>下的Binding	SAML的绑定信息
<md:AttributeConsumingService>	IDP用来构造一个<saml:AttributeStatement>元素，该元素与Web浏览器SSO一起推送到SP
<md:AttributeConsumingService>下的index	SP在SSO时生成<samlp:AuthnRequest>元素中AttributeConsumingServiceIndex属性的值

IDaaS中配置SMAL应用示例

IDaaS平台支持基于标准SAML协议的SSO，这里以配置阿里云RAM为例演示如何配置。

1、获取SP的元数据信息

SP会提供自己的元数据信息，以阿里云RAM为例，找到元数据URL。

浏览器访问该URL得到元数据信息。

2、IDaaS中配置SP的元数据信息

① 添加SP应用

以IT管理员账号登录云盾IDaaS管理平台，具体操作请参考 IT管理员指南-登录 。

点击左侧导航栏 **应用** > **添加应用** 在右侧选择一个SAML应用，点击添加应用。IDaaS支持多种SAML应用，这里以添加阿里云RAM-用户SSO为例进行展示。

点击添加SigningKey按钮，输入名称等信息，系统会据此生成应用的证书，用于IDP和SP通信的认证。

选择刚才添加的SigningKey，填写应用的相关信息。

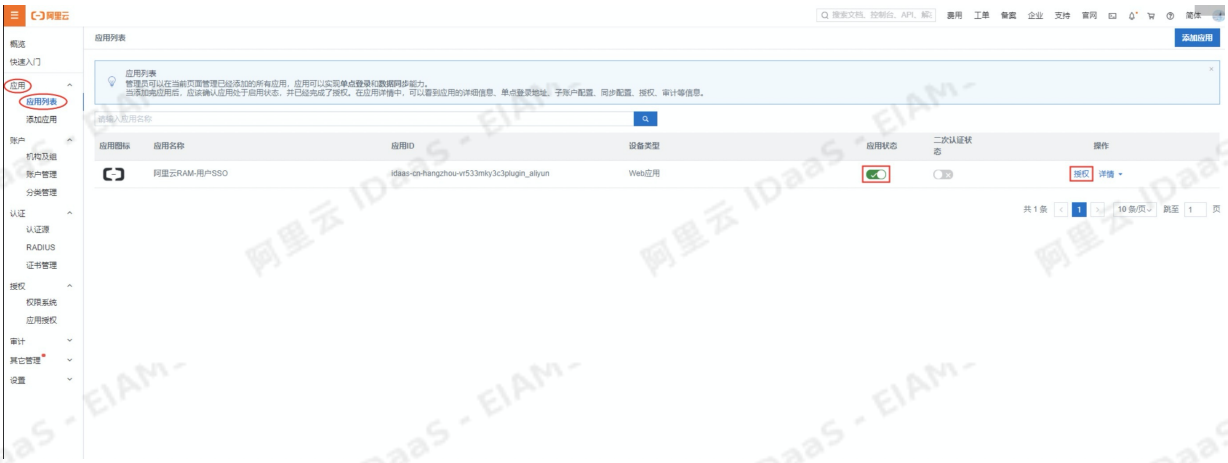
应用名称等信息可以自定义，EntityId、ACS URL等信息从步骤1中的到的SP的元数据中复制过来，需要填写的主要信息如下：

参数名称	说明
应用名称	所添加应用的名称，可以为任意值
应用类型	引用的类型，只有选中的应用类型才会在用户对应客户端中显示

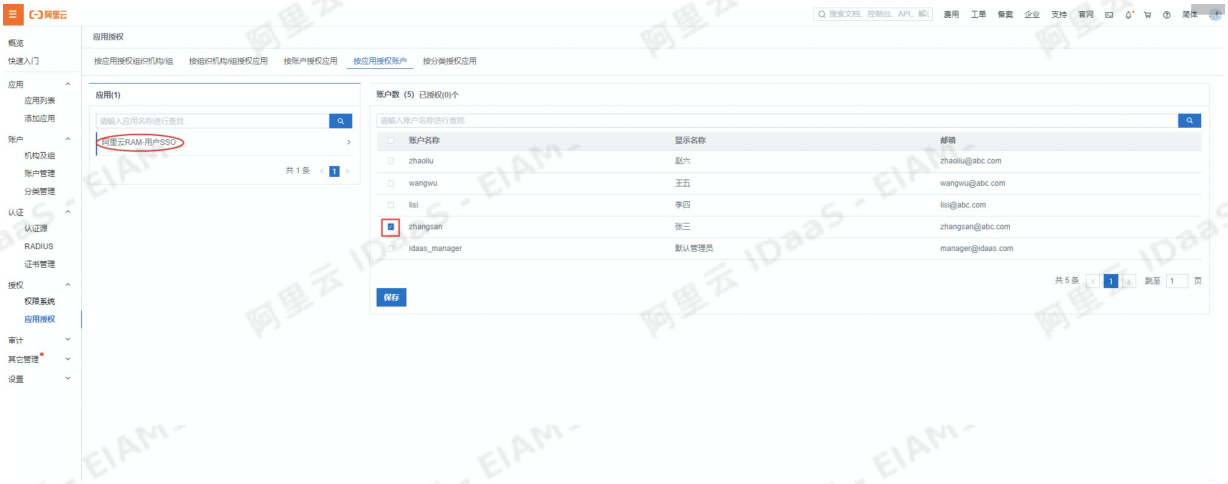
参数名称	说明
IDaas EntityId	IDaaS实体Id，这里以阿里云RAM为 例：https://signin.aliyun.com/11xxxxxxxxxxx63/saml/SSO，11xxxxxxxxxxx63为阿里云的账户ID
SP Entity ID	可以在控制台SAML服务提供方元数据中查看，默认与IDaaS IdentityId相同
SP ACS URL（SSO Location）	单点登录地址，这里以阿里云RAM为例：https://signin.aliyun.com/saml/SSO
NameIdFormat	名称标识格式类型，这里以阿里云RAM为例，选择 urn:oasis:names:tc:SAML:2.0:nameid-format:persistent

② 启用应用并且授权

应用配置好以后需要先启用应用，并且将服务授权给账户，点击左侧导航栏**应用>应用列表**启用该应用并授权给账户。



IDaaS支持多种方式进行授权，这里以按应用授权账户为例。

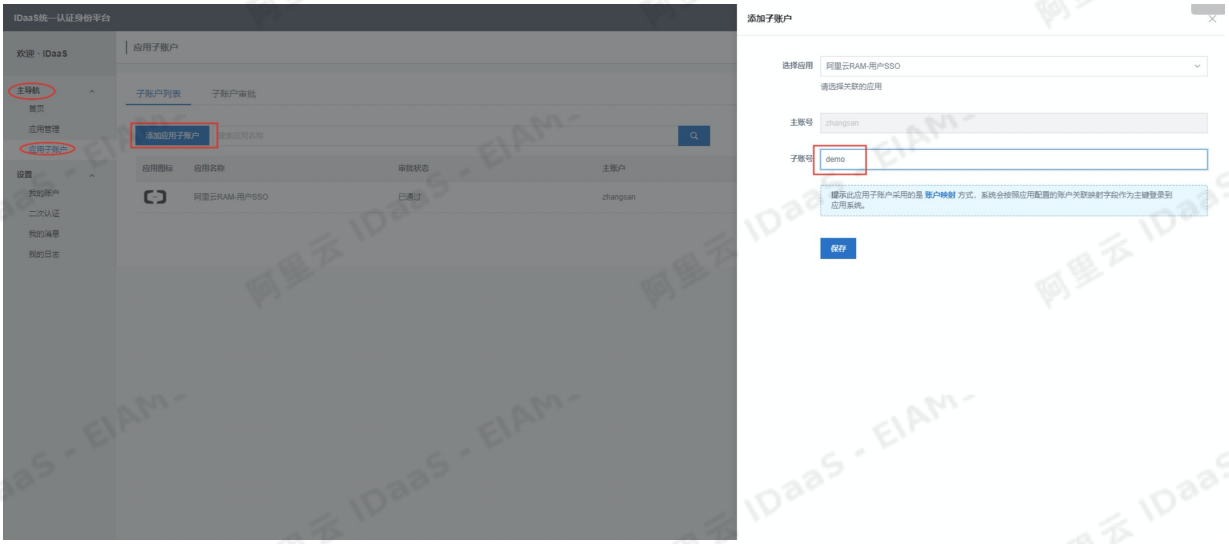


③ IDP新建子账户（非必须步骤）

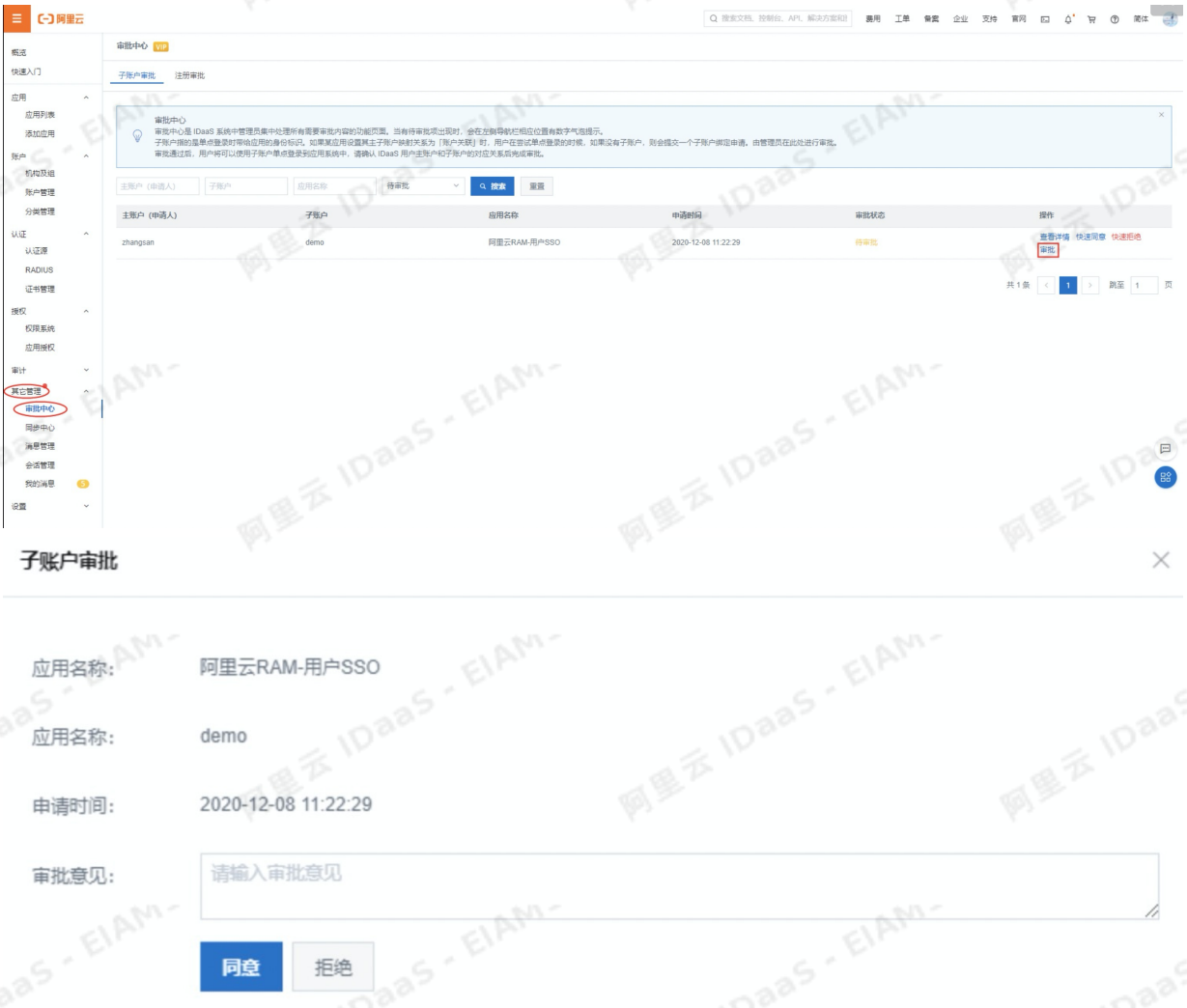
一个系统要SSO到另外一个系统，需要使用对方能够识别的账号进行认证，可以使用账号同步（两套系统中的账号信息相同）或者新建子账户进行账号映射的方法。账号映射是指给IDP的账户建立一个SP中已经存在的账户作为子账户，身份认证的时候通过子账户进行认证。例如SP系统中有个账户“demo”，我们想用IDP系统中的“zhangsan”账号SSO到SP，则需要给账号“zhangsan”新建一个子账户“demo”。

这里以阿里云RAM演示新建子账户的功能，如下图，阿里云RAM中有账户demo@117xxxxxxxxxx63.onaliyun.com。

IDaaS登录授权账户新建子账户，子账户必须和阿里云RAM中的账户名相同（不需要@符号后面的内容），否则在SSO的时候会因为找不到对应用户而报错。点击左侧导航栏**主导航>应用子账户**新建子账户。由于上一步我们新建的账号是demo@117xxxxxxxxxx63.onaliyun.com，为了映射账号，那这里需要建的子账号名称是demo。



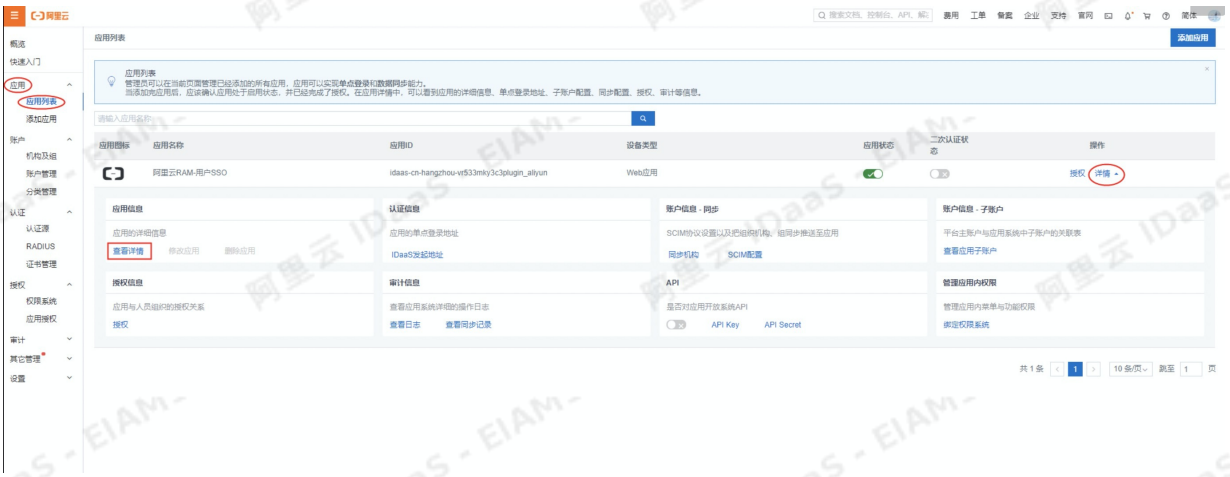
登录管理员账户，点击左侧导航栏 **其它管理 > 审批中心** 审核通过该应用子账户的添加。



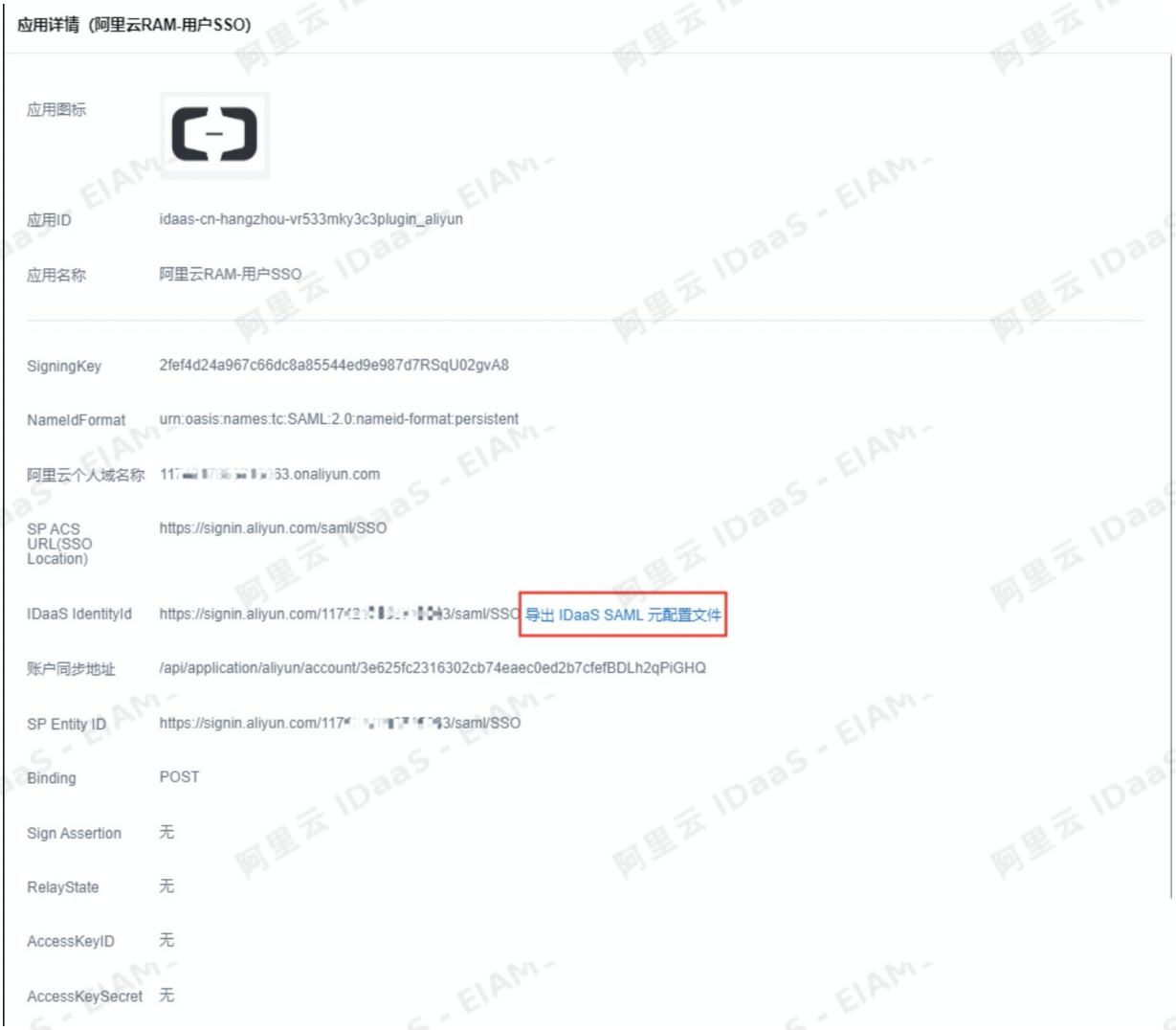
3、SP中配置IDaaS的元数据信息

① 获取IDaaS的元数据信息

以IT管理员账号登录云盾IDaaS管理平台，点击左侧导航栏**应用>应用列表**选择刚才添加的应用，点击查看详情，如下图：



点击导出SAML元配置文件，将IDaaS的元数据文件保存到本地电脑。



IDaaS元配置文件示例如下：

```
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" entityID="https://signin.aliyun.com/117xxxxxxxxxx63/saml/SSO">
<md:IDPSSODescriptor WantAuthnRequestsSigned="false" protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
<md:KeyDescriptor use="signing">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
<ds:X509Certificate>TIIC4jCCAcqgAwIBAgIIDmXMktHMYX8wDQYJKoZIhvcNAQEFBQAwMTELMAkGA1UEBhMCQ04xCzAJBgNVBAgTAkJKMRUwEwYDVQQDDAzor5XnlKjlhazlj7gwHhc
NMjAxMjA3MDMwNTU3WhcNMjExMjA3MDMwNTU3WjAxMQswCQYDVQQGEwJDTjELMAkGA1UECBMCQkoxFTATBgNVBAMMDOivleeUqOWFrOWPuDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQ
oCggEBALiuwcn+sbMaVRT2Byb3GzBV1P0eOoK326fQS9rdeaGaIykSMqCMMKOZ+/QfdMWh+9Fr59A5pIEbCN7aP3P+cV1ClqhfKD4DTbsmGikSiUYgYf4tWztZx9NFWyuoucm8LOKKpKlPb
jUyLudzLlQGOCrX/4Be0md4mIVZMK96J41jRuJXUTxFepE0cTEi15SXbEsXrnJ1wueFylNKl9JerbCJlEDayktAYvkMrmn2d2R2etiVR4Una9pBqtPvCElPKCNesWAE/3AcWTgSj+u8ocnT
gnknIfVO65QRxaNrDAyTOpkquXFshs+DtlILEdk2p9UUxkUCNySbIIM/gVgL0TkCAwEAATANBgkqhkiG9w0BAQUFAAOCAQEAeDdH7BYcalDYfpNKvJrHA9rKZ2vBTi4uy2WoXclE0EdzBGU
C4loPL5g3ictNA+4S8tnCkzl8aQr79tjUmcL/0Uzv4sdOggwglmkgw3kek9Yq44i/ycMN8HVeF/vtyVxhlvqBeXU2P5n6jFqatG+VkeVGyiJQHwuPlUHokXWwyukcjr35CQQX5WALFNJ+F6
8ICKT9Ulqb5GtQgrdlJoRQBlEb//IjxlZJAvZ6CxLnVCgVUSOI4xYEb8ATZPbzLIMIyXN4U6r6VxvBJuW/eMcqogYSYssbngSgpHmZFV9+MrDSjJLLtsVRuzmF+cBisojvo53z3EiNu/c4F
GlUuKozPA==</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:KeyDescriptor use="encryption">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
<ds:X509Certificate>TIIC4jCCAcqgAwIBAgIIDmXMktHMYX8wDQYJKoZIhvcNAQEFBQAwMTELMAkGA1UEBhMCQ04xCzAJBgNVBAgTAkJKMRUwEwYDVQQDDAzor5XnlKjlhazlj7gwHhc
NMjAxMjA3MDMwNTU3WhcNMjExMjA3MDMwNTU3WjAxMQswCQYDVQQGEwJDTjELMAkGA1UECBMCQkoxFTATBgNVBAMMDOivleeUqOWFrOWPuDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQ
oCggEBALiuwcn+sbMaVRT2Byb3GzBV1P0eOoK326fQS9rdeaGaIykSMqCMMKOZ+/QfdMWh+9Fr59A5pIEbCN7aP3P+cV1ClqhfKD4DTbsmGikSiUYgYf4tWztZx9NFWyuoucm8LOKKpKlPb
jUyLudzLlQGOCrX/4Be0md4mIVZMK96J41jRuJXUTxFepE0cTEi15SXbEsXrnJ1wueFylNKl9JerbCJlEDayktAYvkMrmn2d2R2etiVR4Una9pBqtPvCElPKCNesWAE/3AcWTgSj+u8ocnT
gnknIfVO65QRxaNrDAyTOpkquXFshs+DtlILEdk2p9UUxkUCNySbIIM/gVgL0TkCAwEAATANBgkqhkiG9w0BAQUFAAOCAQEAeDdH7BYcalDYfpNKvJrHA9rKZ2vBTi4uy2WoXclE0EdzBGU
C4loPL5g3ictNA+4S8tnCkzl8aQr79tjUmcL/0Uzv4sdOggwglmkgw3kek9Yq44i/ycMN8HVeF/vtyVxhlvqBeXU2P5n6jFqatG+VkeVGyiJQHwuPlUHokXWwyukcjr35CQQX5WALFNJ+F6
8ICKT9Ulqb5GtQgrdlJoRQBlEb//IjxlZJAvZ6CxLnVCgVUSOI4xYEb8ATZPbzLIMIyXN4U6r6VxvBJuW/eMcqogYSYssbngSgpHmZFV9+MrDSjJLLtsVRuzmF+cBisojvo53z3EiNu/c4F
GlUuKozPA==</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location="https://lidcfkpjfb.login.aliyunidaas.com/enduser
/api/application/plugin_aliyun/idaas-cn-hangzhou-vr533mky3c3plugin_aliyun/sp_sso"/>
<md:SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="https://lidcfkpjfb.login.aliyunidaas.com/enduser/api
/application/plugin_aliyun/idaas-cn-hangzhou-vr533mky3c3plugin_aliyun/sp_sso_post"/>
</md:IDPSSODescriptor>
</md:EntityDescriptor>
```


② SP中配置元数据信息

不同的SP配置IDP的元数据方式不同，有的需要填入参数，有的可以直接上传元数据文件。以阿里云RAM为例，在阿里云RAM选择开启SSO功能，并且上传刚刚下载的元配置文件，完成SP中的IDP元数据信息配置。

4、功能演示

① IDP发起SSO

授权用户登录IDaaS，点击左侧导航栏 **主导航>首页**在我的应用中点击该应用进行单点登录，点击应用的图标进行单点登录。

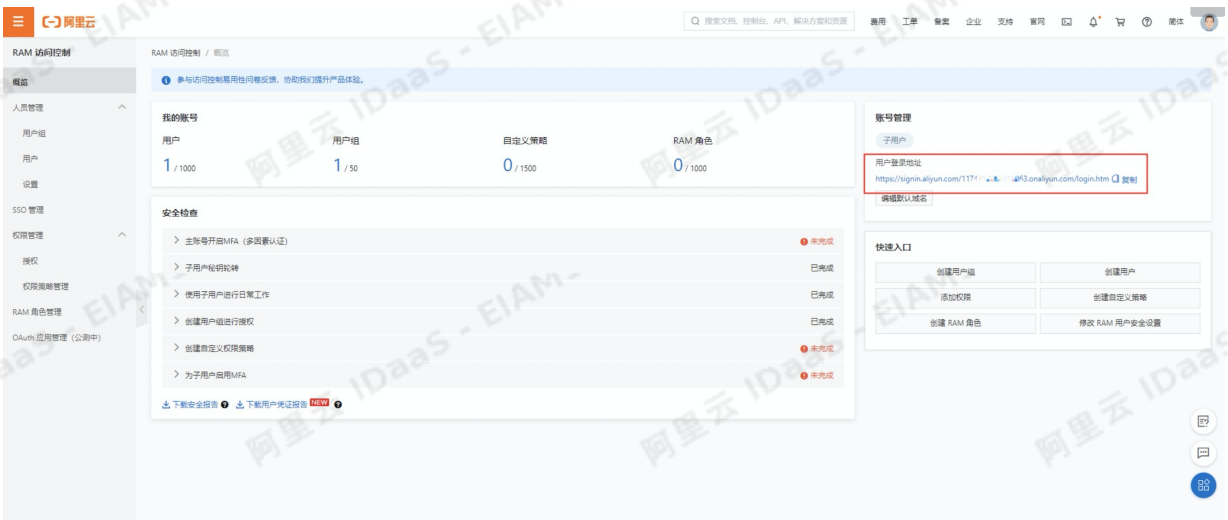
选择子账户demo进行单点登录。

成功登录阿里云RAM控制台，然后就可以SP提供的资源了。

如果账号配置错误或者选择的登录账号不是阿里云RAM中的账户，则会提示账户不存在。

② SP发起SSO

访问阿里云RAM子账户登录地址。



登录界面上可以看到“主账号登录”和“使用企业账号登录”两种选择，点击使用企业账号登录。

浏览器会自动跳转到IDP的登录界面，登录IDaaS授权账号，例如zhangsan，然后IDaaS认证完成以后就会跳转到阿里云RAM控制台。

FAQ

1. 代码中如何解析SAMLRequest

SP发起SSO的时候会生成SAMLRequest，SAMLRequest是Base64编码后的内容，我们需要解析以后才能得到需要的内容，如下代码可以解析SAMLRequest，然后就可以拿到AuthnRequest进行认证。

```
import java.io.*;
import org.opensaml.xml.util.Base64;
import java.util.zip.InflaterInputStream;
import java.util.zip.Inflater;
public class SamlRequestTest {
    public static void main(String[] args) throws Exception {
        // 接收到的原始SAMLRequest
        String samlRequest = "fZJNT%2BMwEIbv%2Bysi3%2FNhd9umVhPUXYQWiRUVCRy4IMedFIMzzmacavvvCQll4bAcfLD0fnjm8frsb2ODA3RkHGaMRwkLALXbGdxn7La8CFN2ln9bk2qsaOWm9494A396IB9siKDzg%2B%2BnQ%2Bob6AroDkbD7clVxh69b0nGMZk9GoyUNcceI%2B2a%2BDUqLoprFpwPKQaVH6tPBmyttmiPhyqybv9uNTulaPQD7vqhOFatGU5rjR4T4tb2g%2FxhksejPtQYVmCehmHC2h3h%2BETPz%2B3yk5LH1D4QORZcuE7DOGHGamUJWHB5njElaljCTOxTriuuVlzVS6iTNIVK8F01iGiriMwB%2FtmIerhE8gp9xkQikpCLMElLzuV8Jvkqwqtf71mw7Zx32tkfBqeF9x1Kp8iQRNUASa91sfl9JUWUyGoSkfxVlttwe12ULLg7gROv4AaUSHJC9XVW%2BlbM8omsHF%2FcfUz4OkCd2LP8%2F6R5ukgWq8V8NpsnKyEW7%2BjX8cfW%2FO36%2BXvlLw%3D%3D&RelayState=https%3A%2F%2Fhomenew.console.aliyun.com%2Fhome%2Fscene%2FOperation";
        // base64解码
        byte[] decodedBytes = Base64.decode(java.net.URLDecoder.decode(samlRequest, "utf-8"));
        // 获取输入流
        ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(decodedBytes);
        InflaterInputStream inflaterInputStream = new InflaterInputStream(byteArrayInputStream, new Inflater(true));
        byte[] buffer = new byte[decodedBytes.length];
        ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
        // 信息写到输出流
        for (int i = 0; i != -1; i = inflaterInputStream.read(buffer)) {
            byteArrayOutputStream.write(buffer, 0, i);
        }
        String result = new String(byteArrayOutputStream.toByteArray(), "UTF-8");
        // 输出解析后结果
        System.out.println(result);
    }
}
```

解析后的结果为：

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:AuthnRequest AssertionConsumerServiceURL="https://signin.aliyun.com/saml/SSO" Destination="https://nplclnlyvb.login.aliyunidaas.com/enduser/api/application/plugin_aliyun/idaas-cn-beijing-foyeyjskkp7plugin_aliyun1/sp_sso" ForceAuthn="false" ID="a2fe7e32g81cb1a91af7ef088eb21db" IsPassive="false" IssueInstant="2020-12-08T11:53:19.684Z" ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Version="2.0" xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"><saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">https://signin.aliyun.com/1860696533509226/saml/SSO</saml2:Issuer></saml2p:AuthnRequest>
```

2.4. OAuth2.0模板使用指南

概述

IDaaS支持基于标准OAuth2协议，实现从IDaaS到业务应用的单点登录功能。

本文主要包含以下内容：

- 时序说明 - OAuth2协议的简单时序图说明，以及交互参数
- 主要流程 - OAuth2.0模板使用主要流程
- 操作步骤 - 从新建开始配置一个OAuth2应用，以及如何在客户端中开发，包含具体API请求、响应和错误提示
- FAQ - 常见问题以及其对策

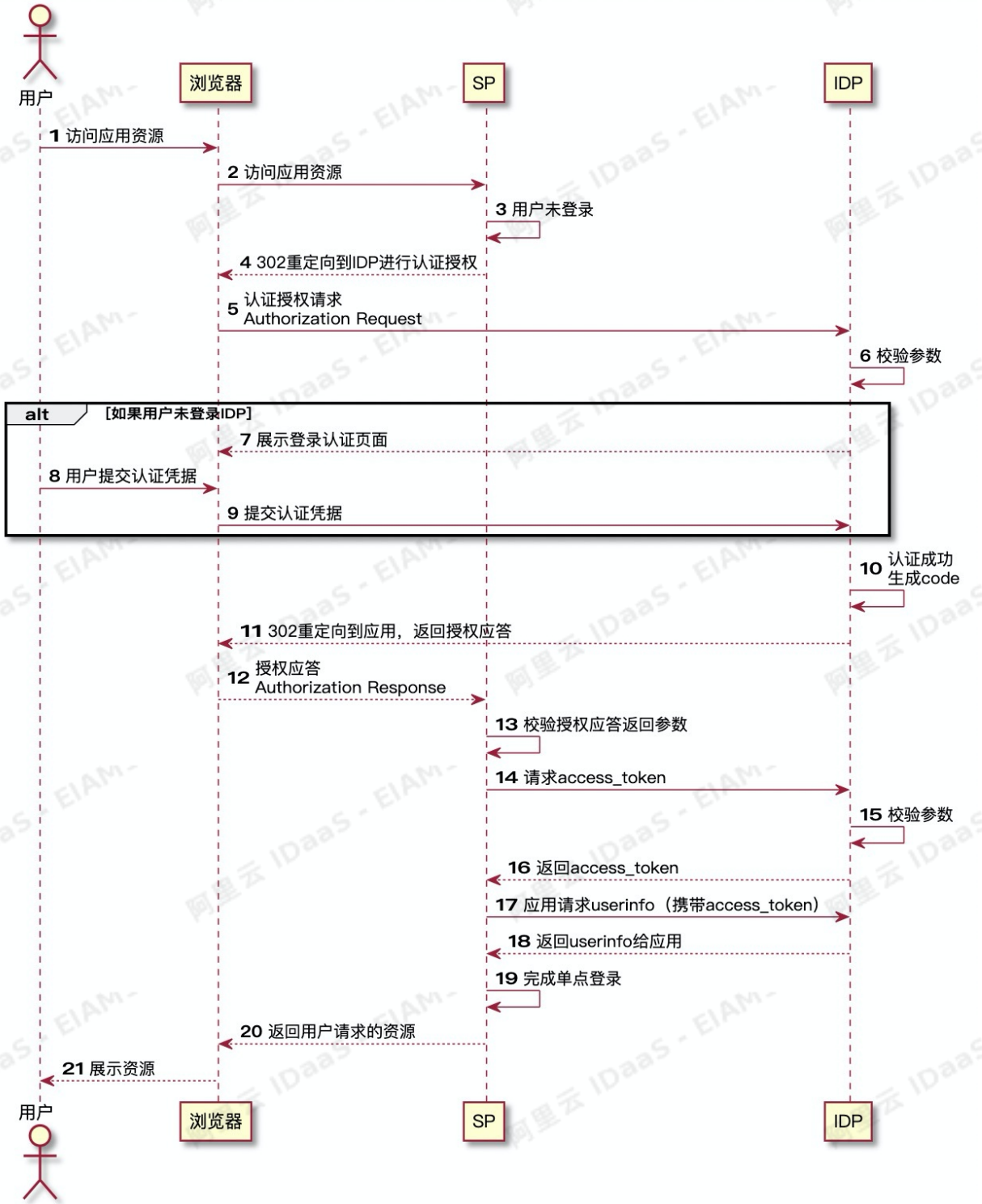
时序说明

场景：SP发起单点登录时序

OAuth 2.0的草案是在2010年5月初在IETF发布的。OAuth 2.0是OAuth协议的下一版本，但不向后兼容OAuth 1.0。 OAuth 2.0关注客户端开发者的简易性，同时为Web应用，PC应用和手机，和IoT设备提供专门的认证流程。规范在IETF OAuth工作组的主导下，OAuth标准于2010年末完成。

OAuth2是一个授权协议， 主要用来作为API的保护， 我们称之为STS（安全令牌服务， Security Token Service）。但是在某些情况下， 也可以被用来实现WEB SSO单点登录。一般的流程是用户把发起页面的URL和state参数关联上， 并保存在SP本地，用户登录后， 可以获取一个Code， 利用Code拿到AT（Access Token）后， 可以利用这个AT获取用户信息userinfo, 进而从state 中， 获取到对应的原始URL，并跳转到这个URL， 从而实现登录到一个业务应用SP的效果。 本文档详细描述了这个SSO过程。

详细时序图（以授权码模式为例）：



说明：

第[5]步参数要求

- response_type：必选、值固定为"code"
- client_id：必选、第三方应用的标识ID
- state：推荐、Client提供的一个字符串，服务器会原样返回给Client，它既能防止CSRF、XSRF， 同时也可以用来对应SP初始发起的状态。
- redirect_uri：必选、授权成功后的重定向地址
- scope：可选、表示授权范围
- prompt：可选

第[6]步校验内容

- a.client_id是否合法
- b.prompt：
  i. 若应用请求IDP时不带prompt参数，则逻辑为用户没登录就跳转到登录页
  ii. 若应用请求IDP时带参数prompt=none,则默认用户已经登录验证，如果IDP发现用户未登录验证，则直接报interaction_required错误
  iii. 若应用请求IDP时带参数prompt=login，则不论用户是否已经登录认证，都重新走一次认证流程

第[11]步返回参数

- 跳转到[5]中指定redirect_uri，并返回：code：授权码 state：步骤[5]中客户端提供的state参数原样返回

第[13]步校验参数

- state是否和自己发送出的一致

第[14]步请求参数

- grant_type：必选、固定值"authorization_code"
- code：必选、Authorization Response中响应的code
- redirect_uri：必选、必须和Authorization Request中提供的redirect_uri相同
- client_id：必选、必须和AuthorizationRequest中提供的client_id相同
- client secret：client的secret，用于授权服务器校验client的合法身份

第[15]步校验参数

- a.client_id、client_secret（若有）是否合法
- b.redirect_uri是否和步骤[A]中的redirect_uri一致
- c.code是否合法：
- >是否过期
- >是否被重复使用，若是就视为一次攻击，加入日志审计，并将之前为code生成的access token撤销
- >比较code和应用的client id是否匹配
- d.server必须在http server头部返回：Cache-Control:no-store and Pragma:no-cache,确保client不会被缓存

第[16]步返回参数

- access_token：访问令牌
- refresh_token：刷新令牌
- expires in：过期时间

第[19]步完成单点登录

- 完成了这一步，就获取到access_token和用户信息，可以展示当前登录用户信息，基于此保存的session会话，用户可以不用再频繁登录，实现点击图标即可跳转应用的过程

主要流程

Step1 创建OAuth2应用，基于OAuth2模板快速创建应用

Step2 授权OAuth2应用，对OAuth2应用授予访问权限

Step3 获取应用信息，基于配置应用信息主要为获取授权码Code

Step4 访问授权URL获取Code，通过相关应用配置，跳转应用地址

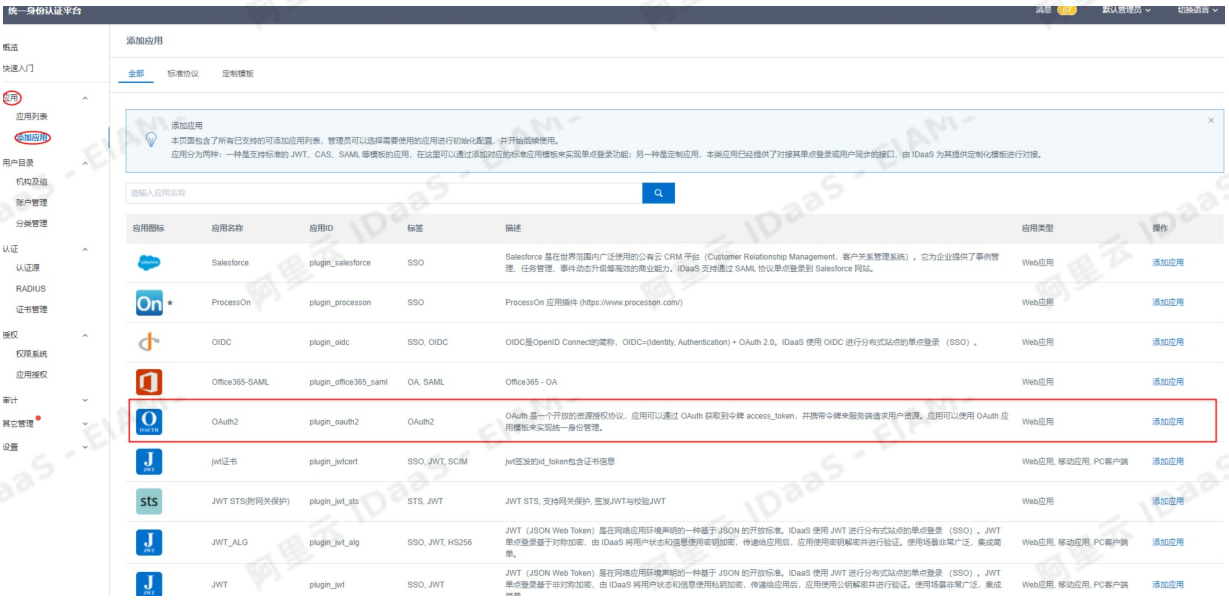
Step5 完成应用侧的开发/配置，就可以实现业务应用单点登录功能

操作步骤

Step1 创建OAuth2应用：

1、首先以IT管理员账号登录云盾IDaaS管理平台。具体操作请参考 IT管理员指南-登录 。

2、点击左侧导航栏**应用>添加应用**选择右侧OAuth。



3、选择OAuth2应用模板点击**添加应用**。

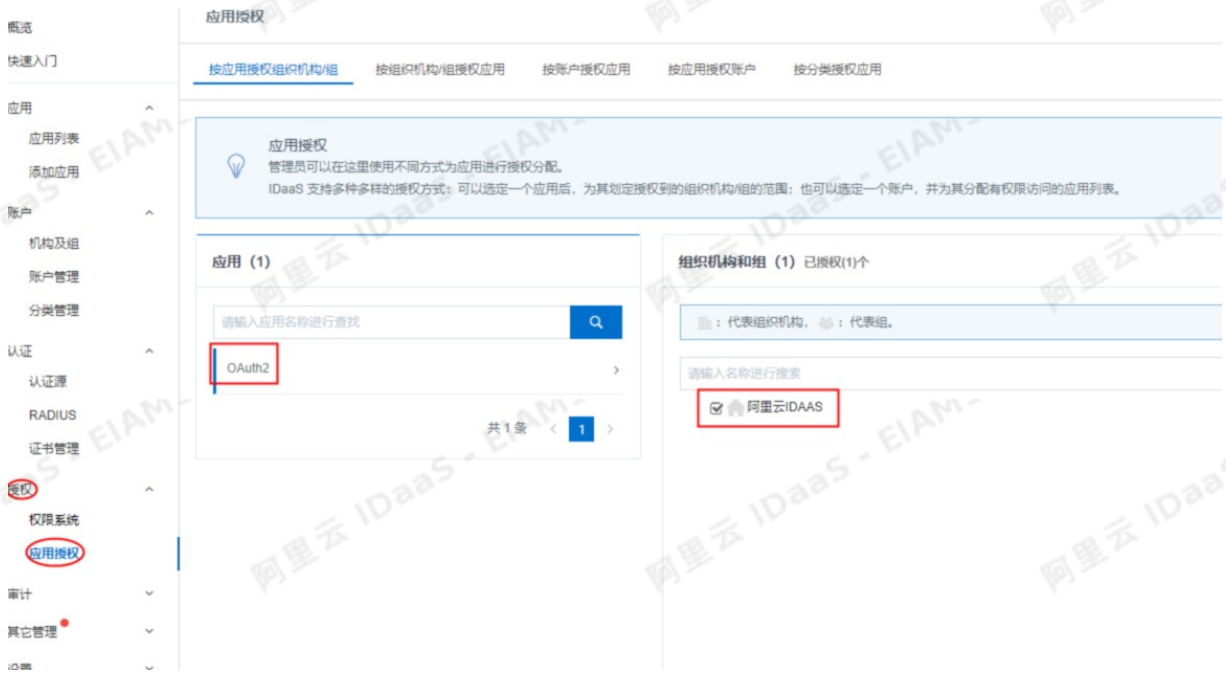
4、Redirect URI：填写需要使用OAuth2单点登录应用的URL

GrantType：选择authorization_code

其他参数默认即可，有需要也可按照实际需要修改

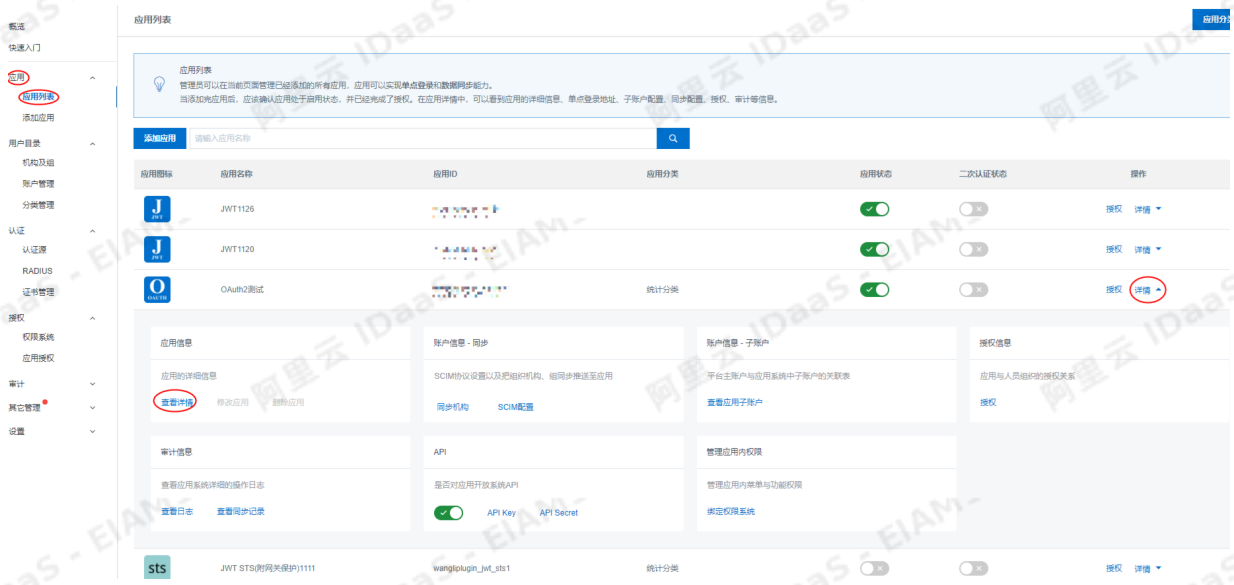
Step2 OAuth2应用授权

应用授权：选择应用（搜索应用）、选择组织机构（搜索组织机构）、勾选授权即可



Step3 获取应用信息

点击左侧导航栏**应用>应用列表**查看 OAuth2 应用详情，获得Client Id、Client Secret、Authorize URL.



Step4 登录获取Code

参考SP发起单点登录时序：**认证成功生成code**

应用通过一个IDP登录按钮等或其它方式， 触发浏览器打开 AuthorizeURL，使用**授权的账户**进行登录，登录成功后跳转到回调地址redirect_uri，并把Code参数一同转发过去。

Step5 完成应用侧的开发/配置

5.1、利用Code从服务器获取AT(Access Token)

参考SP发起单点登录时序：**请求access_token**

无论是JAVA， PHP， 还是.NET应用， 接下来要做的是，应用通过URL参数拿到这个Code 后， 紧接着构建一个应用Token 换AT（Access Token）的过程。

Request URI: https://{IDaaS_server}/oauth/token?grant_type=authorization_code&code={code}&client_id={client_id}&client_secret={client_secret}&redirect_uri={redirect_uri}

IDaaS_server：为实例用户登录页地址（推荐使用）、实例开放接口域名也可以使用

注： OAuth支持多种grant_type 这里使用的是authorization_code模式。

接口说明：获得 access_token

请求方式：POST

请求参数

参数	类型	是否必选	示例值	描述
code	String	是	vuQ3n6	用户登录成功后回调传递的code值
client_id	String	是	oauth2 client_id	OAuth2 client_id
client_secret	String	是	oauth2 client_secret	OAuth2 client_secret
redirect_uri	String	是	http://example.com	重定向 url

- 返回参数示例：

```
{
    "access_token": "b833ca9f-82f7-485e-a18a-4e3e2422f808",
    "token_type": "bearer",
    "refresh_token": "073283f5-6263-4130-86bd-64cbafbaae94",
    "expires_in": 7199,
    "scope": "read"
}
```

参数	类型	示例值	描述
access_token	String	333ab704-abc0-48b3-8af0-496eedd15383	Access Token
token_type	String	bearer	Token 类型
refresh_token	String	073283f5-6263-4130-86bd-64cbafbaae94	刷新token
expires_in	String	7199	Access Token 过期时间
scope	String	read	申请的权限范围

- 错误码说明

HttpCode	错误码	错误信息	描述
400	invalid_grant	Invalid authorization code: "code".	无效的授权码
400	invalid_grant	Redirect URI mismatch.	重定向 URI 不匹配
401	Unauthorized	Unauthorized	未授权的访问
403	Forbidden	Forbidden	无权限访问
404	ResourceNotFound	ResourceNotFound	访问的资源不存在
415	UnsupportedMediaType	UnsupportedMediaType	不支持的媒体类型
500	InternalError	The request processing has failed due to some unknown error, exception or failure.	发生未知错误

①{code}需要替换为授权**应答Authorization Response**中提取到的 code 参数的值。

注意 Code 的值只能用一次

②{client_id}、{client_secret}需要替换为**认证成功生成code**中获得的值

③{redirect_uri} 需要替换为**302重定向到IDP进行认证授权**添加 OAuth2 应用时输入的跳转值

以上完成后你将获得AT(Access Token),此AT将作为你访问的凭证。

5.2、获取用户信息userinfo

参考SP发起单点登录时序：**应用请求userinfo（携带access_token）**

在获取到AT（Access Token）后，应用可以接着向IDaaS平台发送进一步的请求， 以获取到用户信息，实现登录到一个业务应用SP的效果。

①发送GET请求到https://{IDaaS_server}/api/bff/v1.2/oauth2/userinfo?access_token={access_token}

{access_token}替换为前一步获取到的AT(Access Token)

②从返回参数即可获取userinfo信息

Request URI: https://{IDaaS_server}/api/bff/v1.2/oauth2/userinfo

- 接口说明：获取用户详细信息
- **请求方式：GET**
- 请求参数

参数	类型	是否必选	示例值	描述

access_token	String	是	333ab704-abc0-48b3-8af0-496eedd15383	Access Tok

■ 返回参数响应示例

```
{
    "success": true,
    "code": "200",
    "message": null,
    "requestId": "149DA248-8F49-4820-B87A-5EA36D932354",
    "data": {
        "sub": "823071756087671783",
        "ou_id": "2079225187122667069",
        "nickname": "test",
        "phone_number": 176****8971,
        "ou_name": "阿里云IDAAS",
        "email": "test@test.com",
        "username": "test"
    }
}
```

■ 参数说明

参数	类型	示例值	描述
success	boolean	true	是否成功
code	String	200	状态码
message	String	null	返回消息
requestId	String	B3776BB1-930F-4581-B4C3-18F2D7D136CA	请求ID
data	Object	响应数据	
sub	String	823071756087671783	子编号
ouid	String	2079225187122667069	父组织ID
nickname	String	test	昵称
phone_number	String	176****8971	手机号
ou_name	String	阿里云IDAAS	父组织名称
email	String	test@test.com	邮箱
username	String	test	用户名

■ 错误码说明

HttpCode	错误码	错误信息	描述
401	Unauthorized	Unauthorized	未授权的访问
403	Forbidden	Forbidden	无权限访问
404	ResourceNotFound	ResourceNotFound	访问的资源不存在
415	UnsupportedMediaType	UnsupportedMediaType	不支持的媒体类型

500	InternalError	The request processing has failed due to some unknown error, exception or failure.	发生未知错误

这样， 用户登录成功后， 浏览器有了主会话， 一个SP应用利用它获取一个令牌AT（AccessToken），应用拿到AT令牌后去IDaaS认证中心校验令牌是否有效，同时到/userinfo接口去拉取更多的用户信息， 获取到具体的子账户UserId， 有了UserId 就可以创建SP的子会话。 从而在子会话有效期都不用再登录，实现从IDaaS单点登录到应用的全过程。

FAQ

1. 如何强制用户登录认证？

在登录接口增加prompt参数， 当prompt=login则强制跳转登录页，也就是在下图 Authorize URL后面增加"&prompt=login"则不论用户是否已经登录认证，都会展示登录页，用户必须进行一次认证，才可继续单点登录流程。

```
https://cjl.idaas.test.com/oauth/authorize?response_type=code&scope=read&client_id=825d9cffb88ae45d023ae08cd5eb4ca6yk5YBPePbqV&redirect_uri=http%3A%2F%2Flocalhost%3A8082%2Fasd&state=38e78b11072df978a89138144e6e0933zxm3GeFnjLi_idp&prompt=login
```

2. 如何保存初始发起页面？

在SP发起一个SSO请求的时候， SP需要能够把对应的URL， 保存在内存中， 并和OAuth 中的State 参数关联起来。 这样， 在IDaaS返回State 后， 可以找到当初的URL， 并跳转到这个URL， 实现DeepLinking。比如使用了Java的Spring框架的话， 可以用SavedRequest来完成。

2.5. C/S（程序）模板使用指南

唤醒程序后通过OIDC协议向其传递参数实现登录，适用于可以接收解析OIDC协议参数的应用。

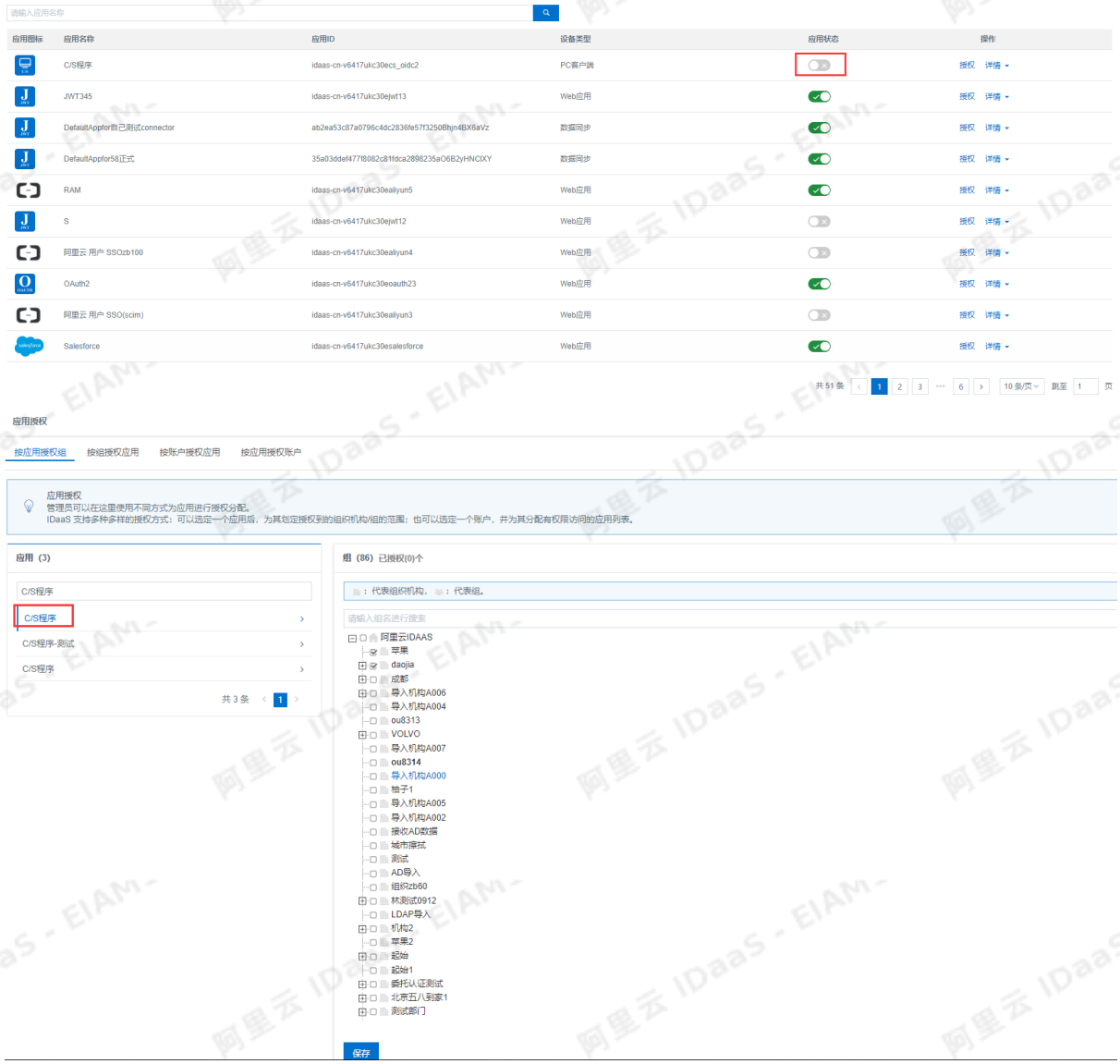
操作步骤

1. 在左侧导航栏，点击**应用 > 添加应用**，选择C/S程序应用模板，点击**添加应用**。

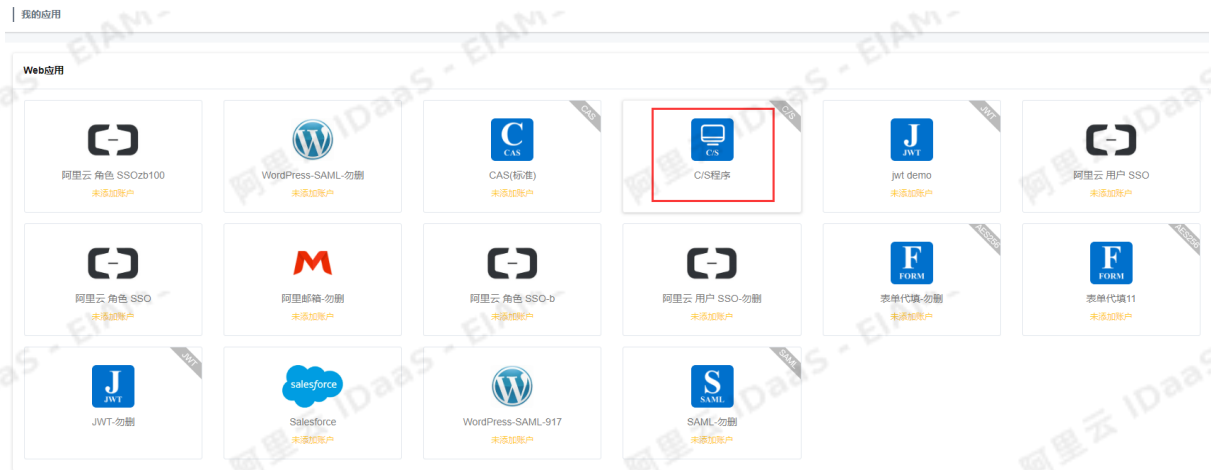
> **说明** C/S应用如果要完成单点登录，需要本地安装IDP-Agent插件。

- 应用名称：根据实际情况进行填写，为必填项；
- 可执行文件：需要填写C/S应用启动的执行文件名称
- 可执行文件的路径：C/S应用文件在本地计算机的位置

2. 开启应用并**授权**，默认是按应用授权组进行授权。



3. 使用普通用户账号登录云盾IDaaS控制台。具体操作请参考普通用户指南-登录。
4. 在用户首页点击C/S应用图标



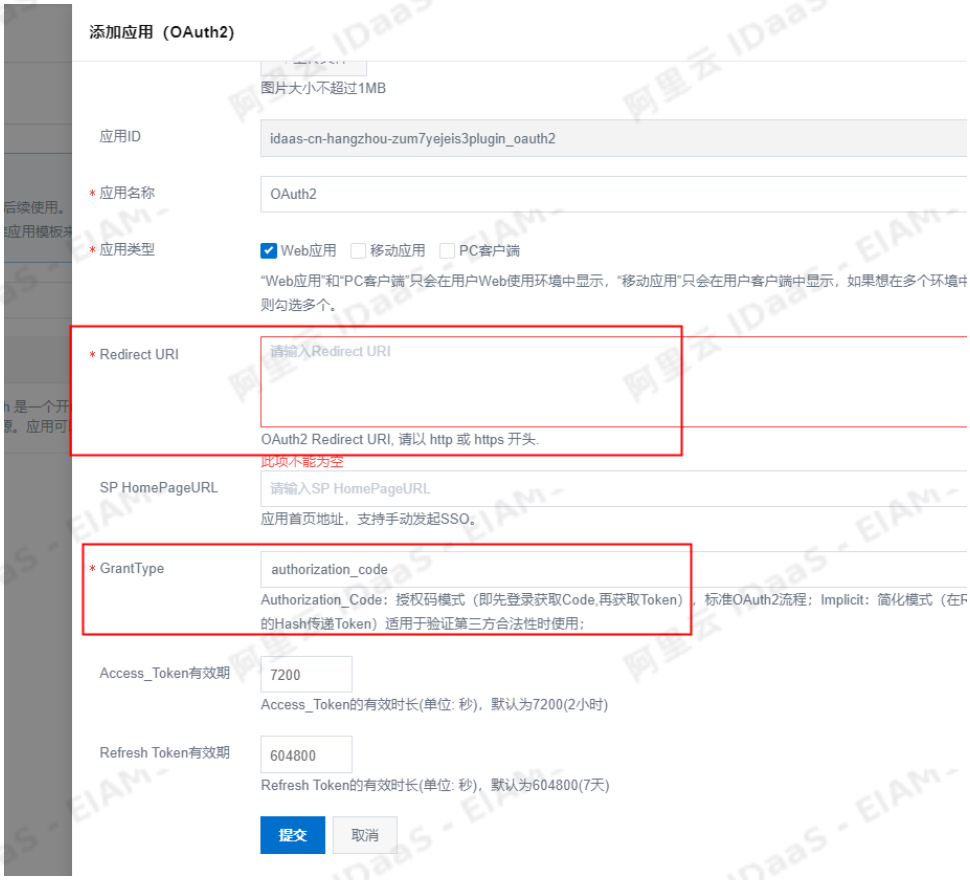
根据页面提示，添加应用的子账户

添加完子账户以后，在用户页面可以点击C/S图标进行单点登录。

2.6. OAuth2.0 模板使用指南

OAuth2是一个开放的资源授权协议，应用可以通过 OAuth 获取到令牌 access_token，并携带令牌来服务端请求用户资源。应用可以使用 OAuth 应用模板来实现统一身份管理。

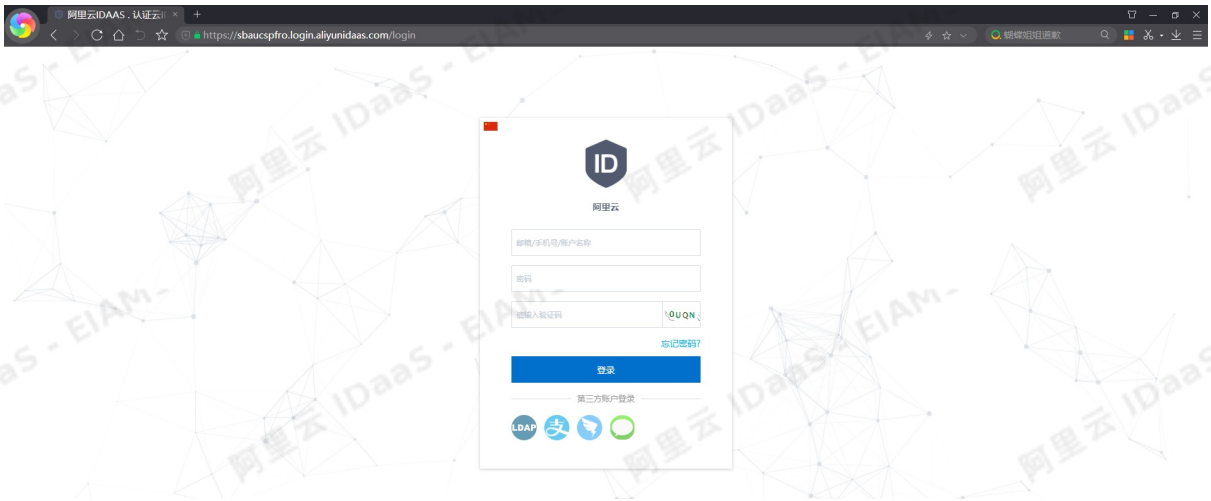
操作步骤

1. 以IT管理员账号登录云盾IDaaS管理平台。具体操作请参考 IT管理员指南-登录。
2. 点击左侧导航栏**应用 > 添加应用**。
3. 选择OAuth2应用模板点击**添加应用**。
4. 

   - Redirect URI：填写需要使用OAuth2单点登录应用的URL
   - GrantType：选择authorization_code
5. 查看 OAuth2 应用详情, 获得 Client Id、Client Secret、Authorize URL.

> **说明** 如果您希望应用系统每次登录时，都需要强制到IDaaS进行认证，在调用的 Authorize URL 后加上 prompt=login 参数即可。

6. 使用浏览器打开 Authorize URL, 使用授权的账户进行登录，登录成功后跳转到回调地址,从浏览器地址栏提取 code 参数的值.



7. 使用 POSTMAN 发送 POST 请求到 https://{IDaaS_server}/oauth/token?grant_type=authorization_code&code={code}&client_id={client_id}&client_secret={client_secret}&redirect_uri={redirect_uri}
   - {IDaaS_server}需要替换为真实IDaaS服务器地址

> **说明** IDaaS服务器地址获取方式：访问云盾IDaaS管理控制台，使用用户访问的Portal的地址

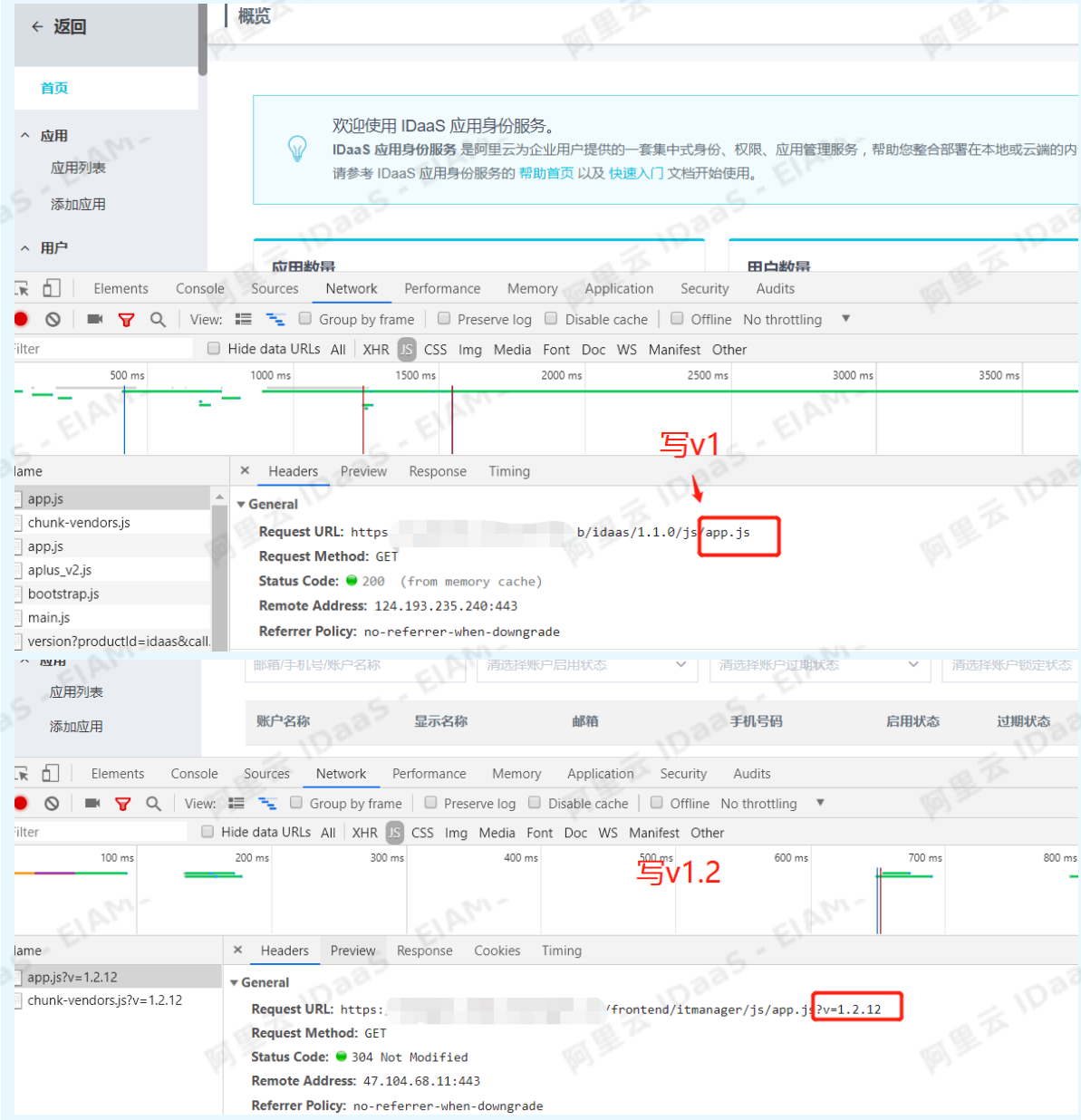
   - {code}需要替换为第 6 步中提取到的 code 参数的值。

重要 Code 的值只能用一次

- {client_id}、{client_secret}需要替换为第 5 步中获得的值
- {redirect_uri} 需要替换为第 2 步添加 OAuth2 应用时输入的跳转值

8. IDaaS服务器将会响应 access_token, 使用该值可访问 IDaaS服务器资源
9. 使用 POSTMAN 发送 GET 请求到 https://{IDaaS_server} /api/bff/v1.2/oauth2/userinfo?access_token={access_token}

说明 其中v1.2是版本，根据实际版本填写。如果js中没有写版本号，那么此处写v1；如果写了版本号，就写对应的版本号



具体接口

1. Request URI: /oauth/token
   - 接口说明:获得 access_token
   - 请求参数

参数	类型	是否必选	示例值	描述
code	string	是	vuQ3n6	用户登录成功后回调传递的code值
client_id	string	是	oauth2 client_id	OAuth2 client_id

参数	类型	是否必选	示例值	描述
client_secret	string	是	oauth2 client_secret	OAuth2 client_secret
redirect_uri	string	是	http://example.com	重定向 url

- 返回参数

参数	类型	示例值	描述
access_token	string	333ab704-abc0-48b3-8af0-496eedd15383	Access Token
token_type	string	bearer	Token 类型
expires_in	string	7199	Access Token 过期时间
scope	string	read	申请的权限范围

- 错误码说明

HttpCode	错误码	错误信息	描述
400	invalid_grant	Invalid authorization code: "code".	无效的授权码
400	invalid_grant	Redirect URI mismatch.	重定向 URI 不匹配
401	Unauthorized	Unauthorized	未授权的访问
403	Forbidden	Forbidden	无权限访问
404	ResourceNotFound	ResourceNotFound	访问的资源不存在
415	UnsupportedMediaType	UnsupportedMediaType	不支持的媒体类型
500	InternalError	The request processing has failed due to some unknown error, exception or failure.	发生未知错误

2. Request URI: /api/bff/v1.2/oauth2/userinfo
   - 接口说：获取用户详细信息
   - 请求参数

参数	类型	是否必选	示例值	描述
access_token	string	是	333ab704-abc0-48b3-8af0-496eedd15383	Access Token

   - 返回参数

     响应示例

```
{
"success": true,
"code": "200",
"message": null,
"requestId": "59C5766B-C7F9-4DF6-B5E4-0F2A89942749",
"data": {
    "sub": "4982789226325725762",
    "ou_id": "592041743949215346l",
    "nickname": "admin",
    "phone_number": null,
    "ou_name": "PG China",
    "email": "sz@xxxx.com",
    "username": "admin_wli"
  }
}
```

     参数说明

参数	类型	示例值	描述
sub	string	4982789226325725762	账户的外部ID
username	string	admin_wli	用户名
nickname	string	admin	显示名称
email	string	sz@xxxx.com	邮箱
phone_number	string	null	手机号

参数	类型	示例值	描述
ou_name	string	PG China	账户所属组织机构名称
ou_id	string	5920417439492153461	账户所属组织机构外部ID

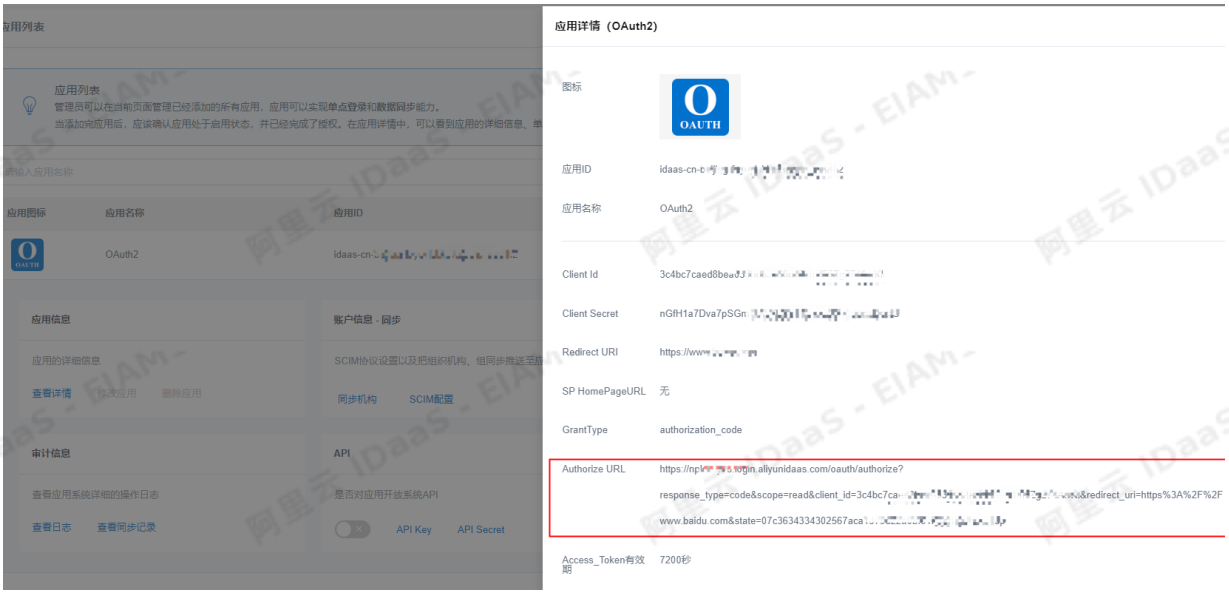
- 错误码说明

HttpCode	错误码	错误信息	描述
401	Unauthorized	Unauthorized	未授权的访问
403	Forbidden	Forbidden	无权限访问
404	ResourceNotFound	ResourceNotFound	访问的资源不存在
415	UnsupportedMediaType	UnsupportedMediaType	不支持的媒体类型
500	InternalError	The request processing has failed due to some unknown error, exception or failure.	发生未知错误

FAQ

1. 是否支持OAuth2退出

在登录接口增加proxmy参数， 当prompt=login则强制跳转登录页，也就是在下图 Authorize URL后面增加 “&prompt=login ”



2.7. 表单代填模板使用指南

本文为您介绍IDaaS通过表单代填，实现应用的单点登录的功能。

背景信息

假设公司将某应用A作为企业的网站，日常访问频繁。传统的访问方式便捷性差且存在安全隐患。

- 应用A日常办公使用频次高，登录繁琐且耗时长；
- 应用A登录未通过验证码进行身份鉴别，存在安全隐患；

解决方案

通过应用身份服务的应用管控（Application）功能，使用其中的表单代填应用模板实现对A应用的单点登录以及身份的鉴别。

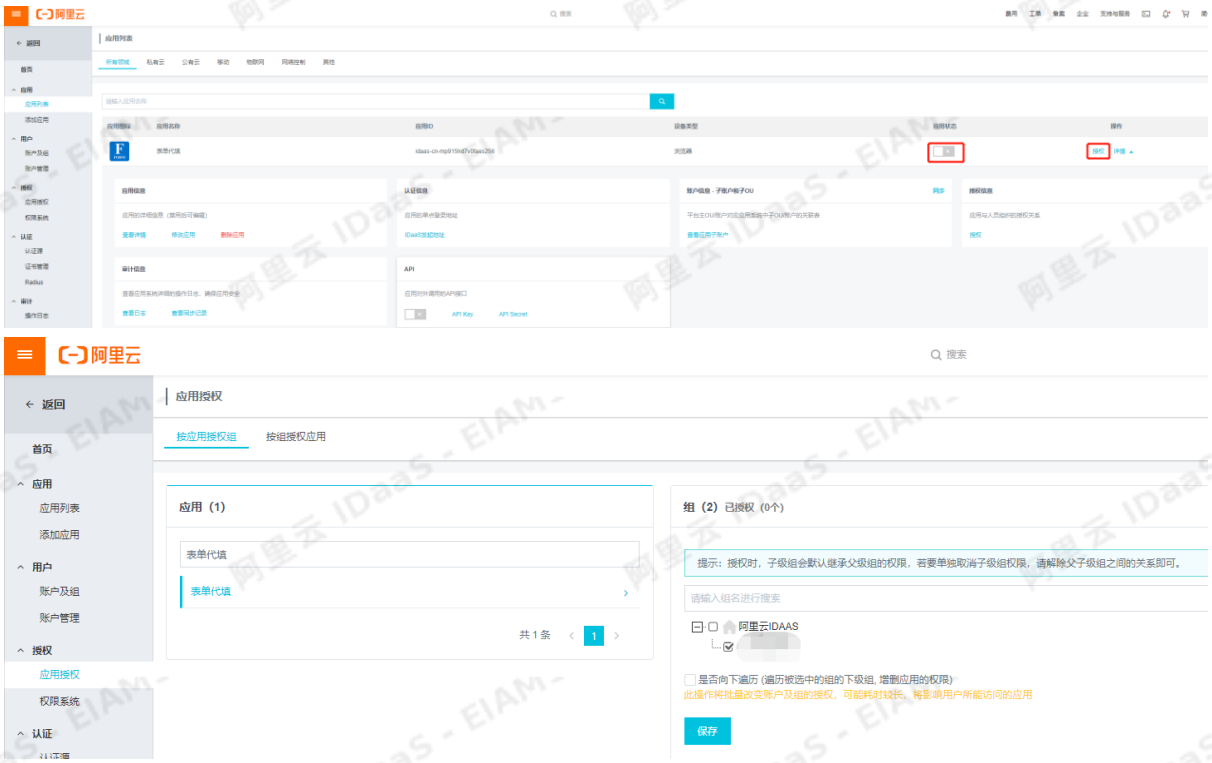
操作步骤

1. 以IT管理员账号登录云盾IDaaS管理平台。具体操作请参考 IT管理员指南-登录。
2. 在左侧导航栏，单击应用 > 添加应用，选择表单代填应用模板点击添加应用

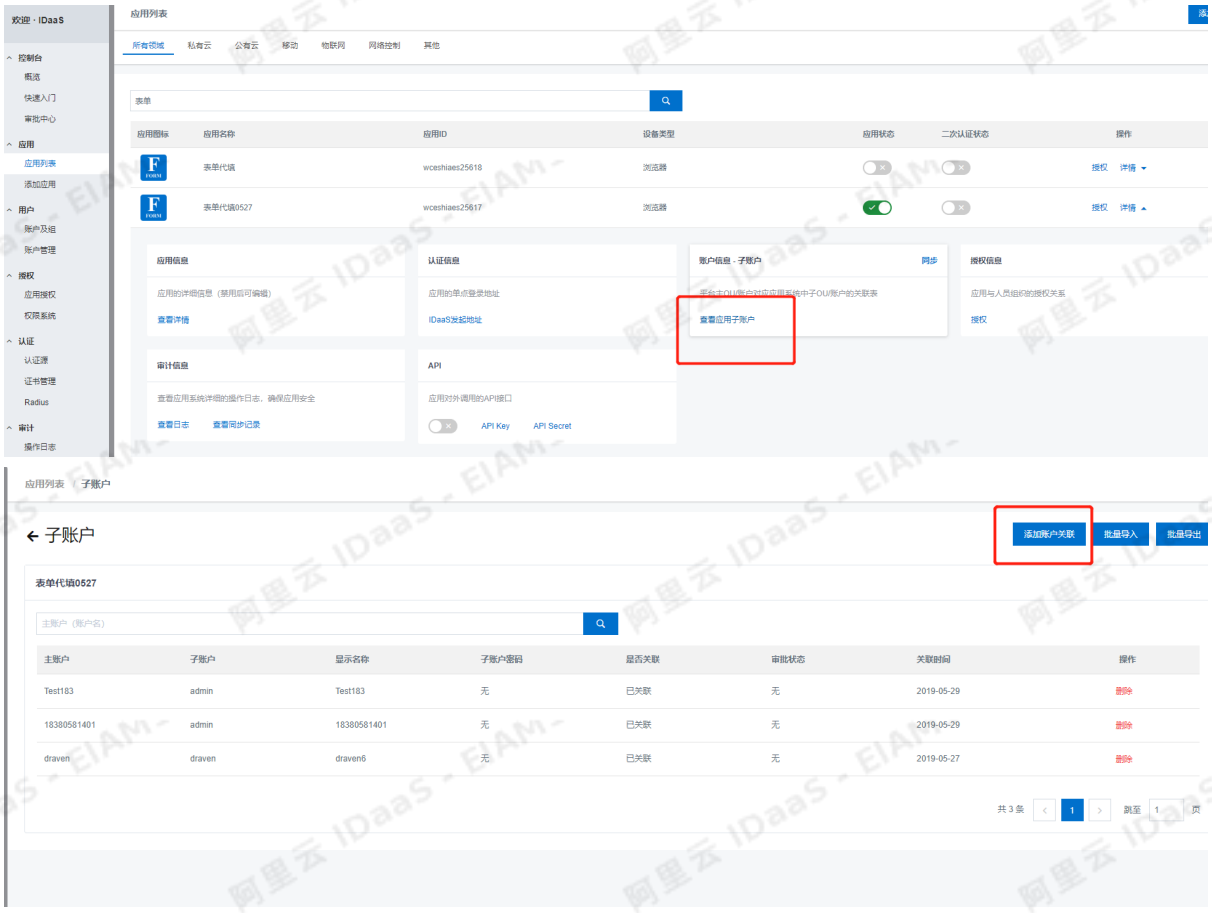
3. 在添加应用对话框中，填写应用的信息

- 提交登录URL：应用A的登录接口
- 登录名属性名称：登录接口的登录名
- 参数登录密码属性名称：登录接口的登录密码
- 参数登录提交方式：登录接口的请求方式
- 账户关联方式：账户和密码

4. 启用应用并授权



5. 绑定主子账户。其中主账户是IDaaS平台的账户，子账户是应用A中的账户



6. 登录已授权该应用的普通用户，点击图标进行单点登录

若以上步骤全部成功完成，即可实现使用表单代填单点到应用A。

FAQ

1. http应用是否可以使用表单代填

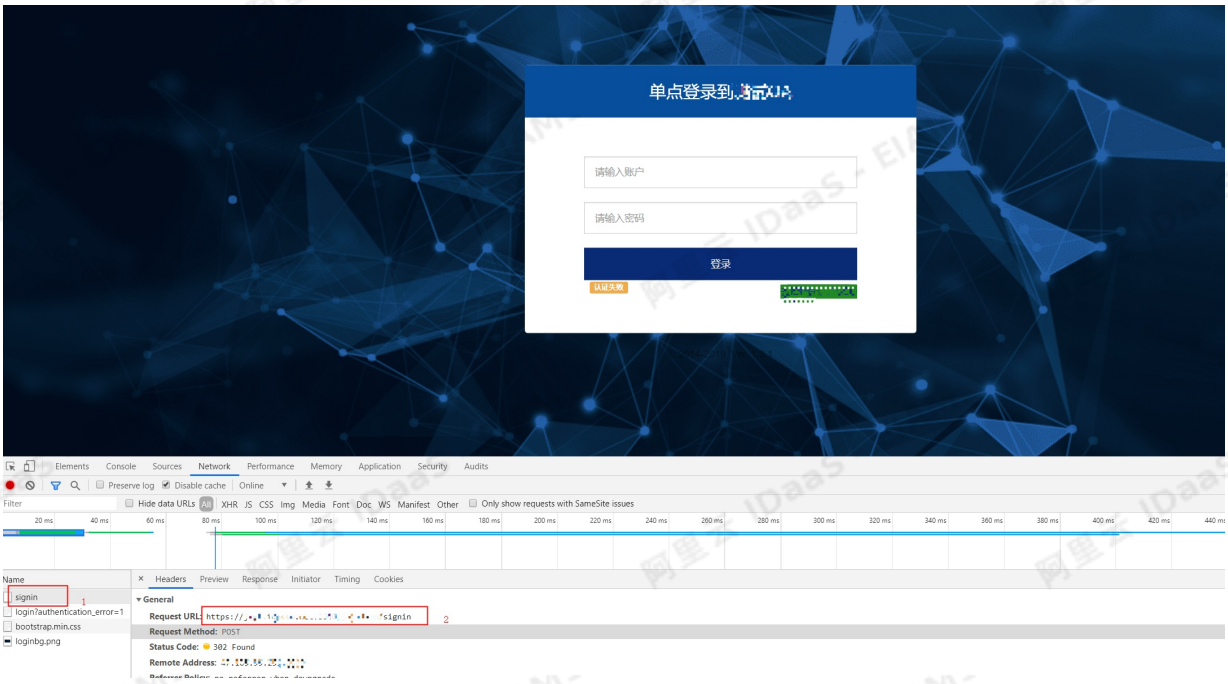
不可以。IDaaS是HTTPS请求，SP是http，从https往HTTP请求会被浏览器的安全机制给限制， 需要SP支持https才行。

2. 前后端分离的应用是否支持表单代填

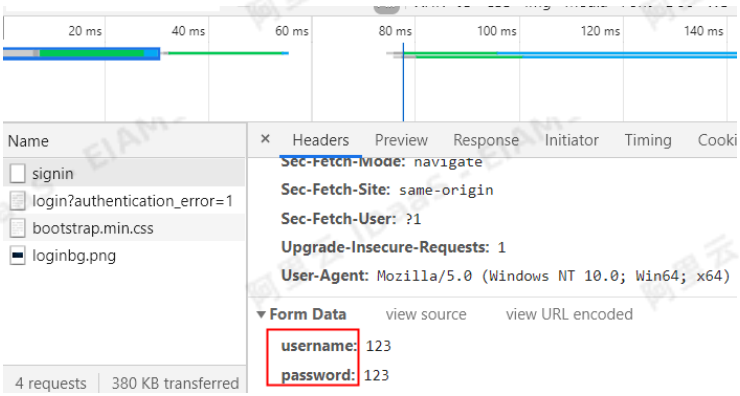
不支持。有图片验证码或者前后端分离的应用不支持表单代填。

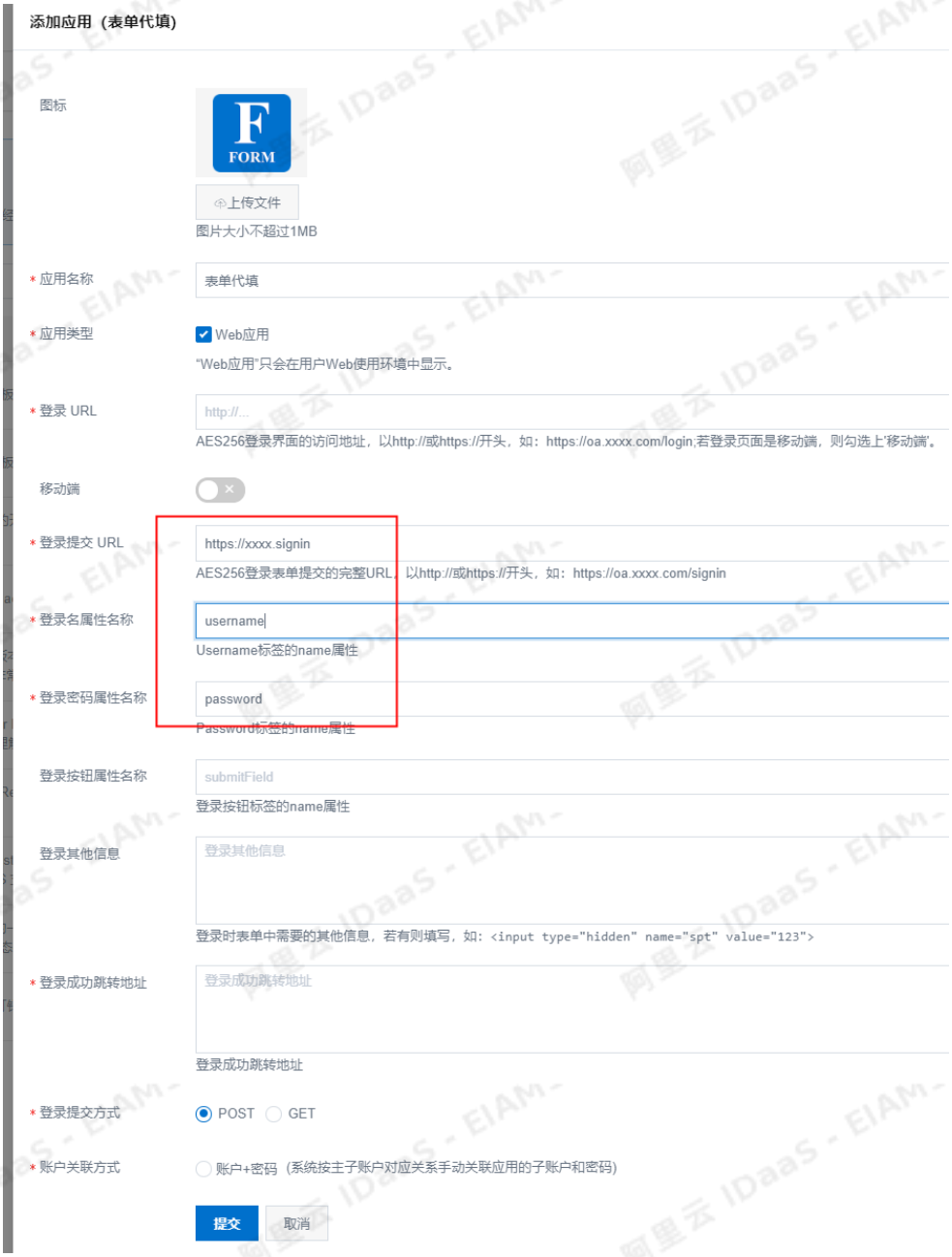
3. 如何获取登录的请求连接和登录参数

在SP登录页面，通过F12打开network，获取到登录请求的url。



并且获取到登录的参数，添加到IDaaS的表单代填模板中。





4. 应用不支持表单代填，如何进行单点登录对接

如果是支持标准协议的，可以使用SAML等协议对接； 如果是自建应用，支持改造的，可以使用JWT进行单点登录，或者使用OAuth2方式对接，IDaaS建议使用JWT方式对接。

3.Gitlab对接单点登录（CAS）

本文为您介绍如何使用CAS应用模板，实现Gitlab单点登录的对接

```
Gitlab常用命令：
# 启动Gitlab
gitlab-ctl start
# 停止Gitlab
gitlab-ctl stop
# 重启Gitlab
gitlab-ctl restart
# 重新加载Gitlab配置
gitlab-ctl reconfigure
# 查看状态
gitlab-ctl status
# 查看所有的logs
gitlab-ctl tail
```

操作步骤

一、IDaaS管理员创建CAS应用

1. 以IT管理员账号登录云盾IDaaS管理平台。具体操作请参考 IT管理员指南-登录。
2. 点击左侧导航栏**应用 > 添加应用**。
3. 选择CAS应用模板点击**添加应用**。

4. 配置添加应用参数

- ServerNames：

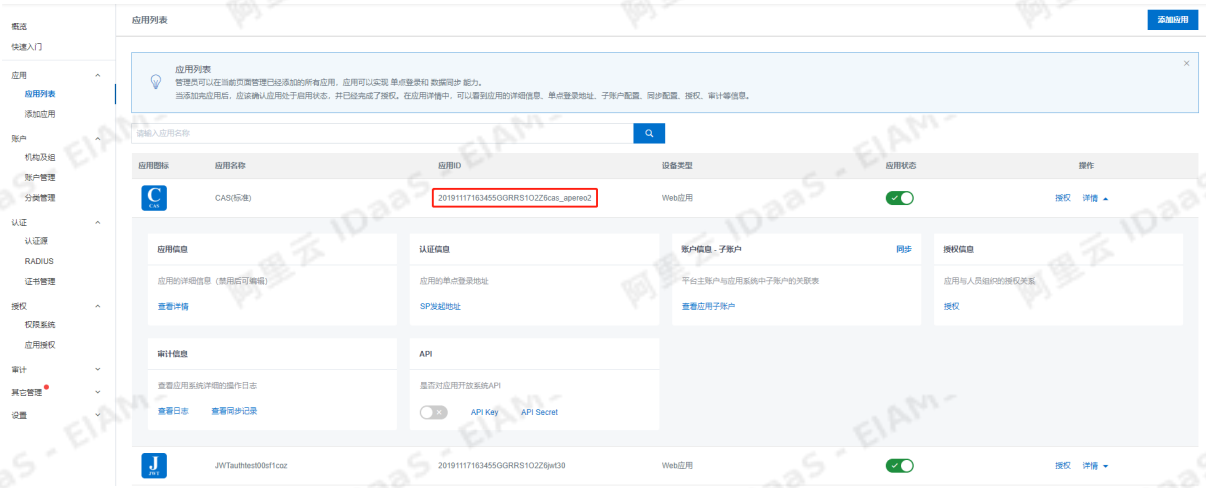
```
Gitlab_url/users/auth/cas3/callback?url=http%3A%2F%2Fgitlab_url%2Fusers%2Fsign_in
Gitlab_url/users/auth/cas3/callback?url=http%3A%2F%2Fgitlab_url%2Fprofile%2Faccount
Gitlab_url/users/auth/cas3/callback?url=Gitlab_url/users/sign_in
Gitlab_url/users/auth/cas3/callback?url=Gitlab_url/profile/Faccount
```

- TargetUrls：

```
Gitlab_url/users/auth/cas3/callback?url=Gitlab_url/users/sign_in
```

> **说明** Gitlab_url (http://xxx.xxx.xxx.xxx)、gitlab_url (xxx.xxx.xxx.xxx) 由Gitlab管理员提供
> %3A表示的是符号：
> %2F表示的是符号 /

5. 创建完成之后,记录下**应用的ID**，需要给Gitlab管理员进行配置



二、Gitlab管理员配置

1. 修改Gitlab配置文件

```
vim /etc/gitlab/gitlab.rb
```

> **说明** gitlab配置文件默认路径为/etc/gitlab/gitlab.rb。

在配置文件中加入以下内容，位置在340行左右

```
#     bind_dn: '_the_full_dn_of_the_user_you_will_bind_with'
#     password: '_the_password_of_the_bind_user'
#     encryption: 'plain' # "start_tls" or "simple_tls" or "plain"
#     verify_certificates: true
#     smartcard_auth: false
#     active_directory: true
#     allow_username_or_email_login: false
#     lowercase_usernames: false
#     block_auto_created_users: false
#     base: ''
#     user_filter: ''
#     ## EE only
#     group_base: ''
#     admin_group: ''
#     sync_ssh_keys: false
# EOS

### Smartcard authentication settings
###! Docs: https://docs.gitlab.com/ee/administration/auth/smartcard.html
# gitlab_rails['smartcard_enabled'] = false
# gitlab_rails['smartcard_ca_file'] = "/etc/gitlab/ssl/CA.pem"
# gitlab_rails['smartcard_client_certificate_required_port'] = 3444
# gitlab_rails['smartcard_required_for_git_access'] = false

### OmniAuth Settings
###! Docs: https://docs.gitlab.com/ce/integration/omniauth.html
# gitlab_rails['omniauth_enabled'] = nil
# gitlab_rails['omniauth_allow_single_sign_on'] = ['saml']
# gitlab_rails['omniauth_sync_email_from_provider'] = 'saml'
# gitlab_rails['omniauth_sync_profile_from_provider'] = ['saml']
# gitlab_rails['omniauth_sync_profile_attributes'] = ['email']
# gitlab_rails['omniauth_auto_sign_in_with_provider'] = 'saml'
# gitlab_rails['omniauth_block_auto_created_users'] = true
# gitlab_rails['omniauth_auto_link_ldap_user'] = false
# gitlab_rails['omniauth_auto_link_saml_user'] = false
# gitlab_rails['omniauth_external_providers'] = ['twitter', 'google_oauth2']
# gitlab_rails['omniauth_providers'] = [
#   {
#     "name" => "google_oauth2",
#     "app_id" => "YOUR APP ID",
#     "app_secret" => "YOUR APP SECRET",
#     "args" => { "access_type" => "offline", "approval_prompt" => "" }
#   }
# ]

### Backup Settings
###! Docs: https://docs.gitlab.com/omnibus/settings/backups.html
                                                                339,1         14%
```

> **说明** 在修改配置时请将下面的注释删除，避免gitlab配置格式的影响

```
# 允许进行单点登录
gitlab_rails['omniauth_allow_single_sign_on'] =true
# 阻止单点登录自动创建账户（默认单点登录会自动创建账户）
gitlab_rails['omniauth_block_auto_created_users'] =true
gitlab_rails['omniauth_providers'] = [
 {
  "name"=>"cas3",
  "label"=>"IDP统一身份认证",
  "args"=> {
   #Cas Server Host 地址
   "url"=>'IDP_url',
   #Cas Server 登录地址
   "login_url"=>'/enduser/api/application/cas_apereo/cas_applicationid/login',
   # Cas Server 校验票据地址
   "service_validate_url"=>'/public/api/application/cas_apereo/cas_applicationid/serviceValidate',
   # Cas Server 登出地址
   "logout_url"=>'/enduser/api/application/cas_apereo/cas_applicationid/logout'
   }
 }
]
```

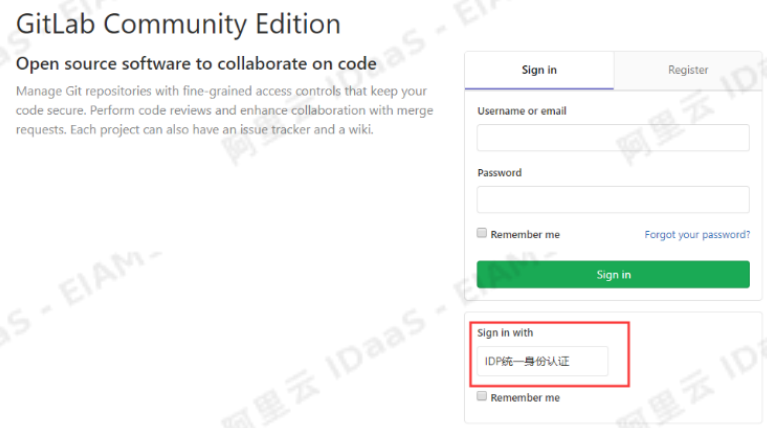
IDP_url:由IDP管理员提供

cas_applicationid:由IDP管理员提供

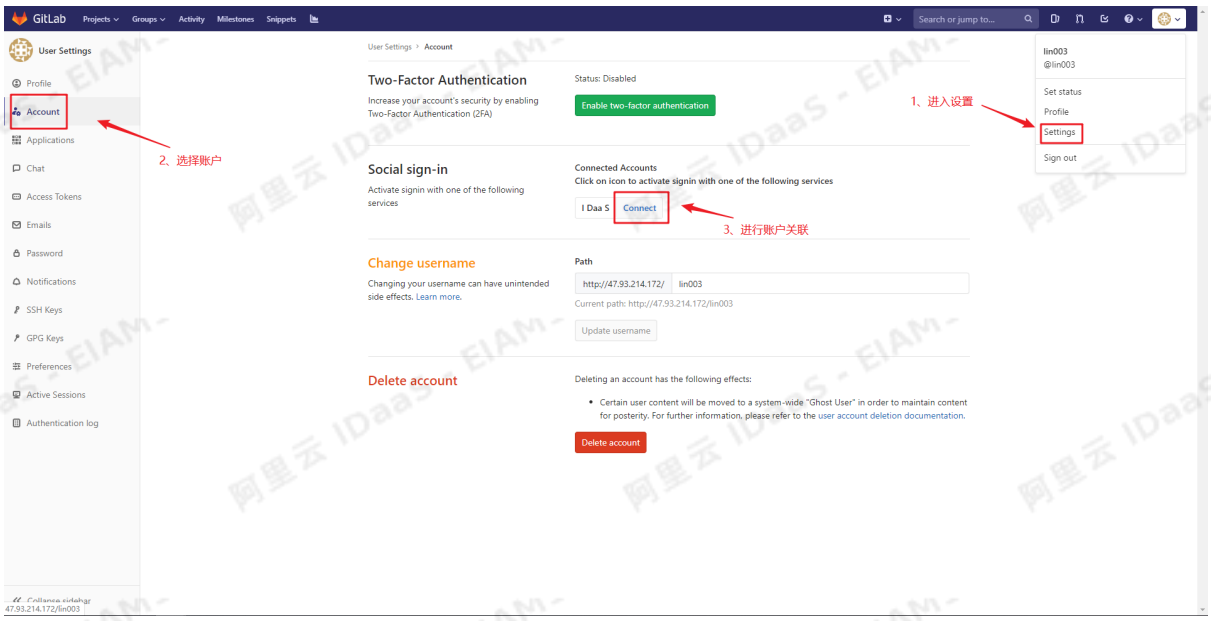
2. 配置好Gitlab后，重启Gitlab,刷新配置信息

```
[root@bogon etc]# gitlab-ctl stop
[root@bogon etc]# gitlab-ctl reconfigure
[root@bogon etc]# gitlab-ctl start
```

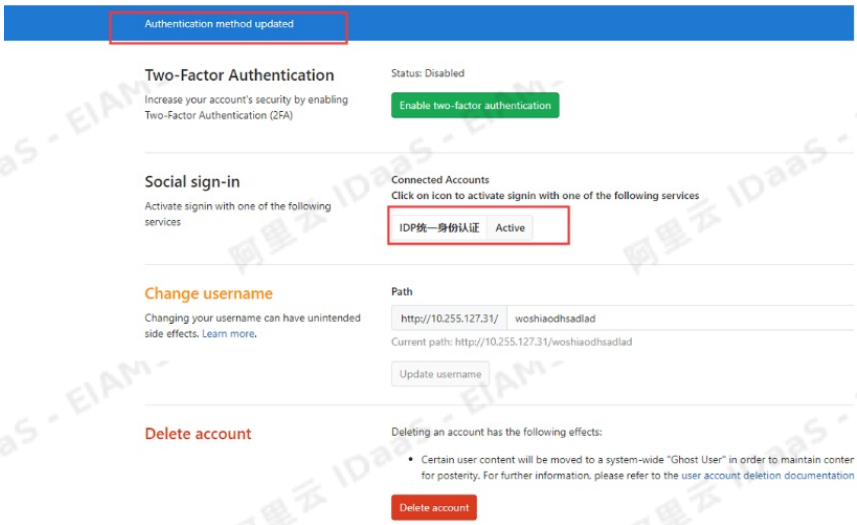
3. 刷新配置后，启动Gitlab，访问登录界面，出现红框中所展示内容表示配置成功



4. 登录Gitlab账号（随便一个普通用户），点击右上角个人头像 - - >Settings - - > Account - - > 点击Connect，进行账号关联



如下图所示则表示配置Gitlab单点登录成功



4.单点登录相关问题

CAS应用问题

CAS（标准）支持SP发起和用户侧的单点登录，但是应用列表处的IDaaS发起地址是不支持的。而我们的CAS(标准)是标准的。

5.主子账户介绍

本文介绍如何通过应用管理功能添加应用子账户。通过添加应用子账户，用户可以通过IDaaS单点登录到其他应用。

背景信息

传统应用的登录方式通过输入用户名和密码，随着日常办公软件数量的不断增加、用户需要记忆多套用户名和密码，给用户带来负担；统一所有用户名和密码固然方便，却会令企业账户体系面临严重的安全隐患。

解决方案

IDaas提供单点登录功能，只需使用单点登录协议和应用对接，并完成主子账户的绑定，即可实现一次登录访问所有授权应用的目的。应用之间后续进行切换时，无需再次输入用户名和密码，全面提升用户办公效率。

主子账户介绍

在进行单点登录时，IDaaS 会向应用系统传递对应的子账户，该子账户需要在应用系统中存在且可识别。

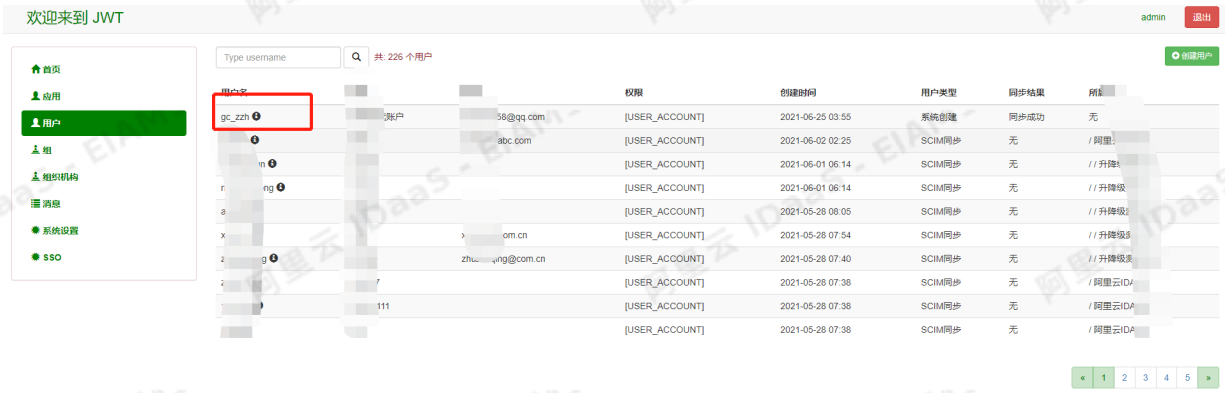
主账户：主账户指的是 IDaaS 中的账户；

子账户：子账户指的是在指定应用系统中，用户会以什么身份进行访问，是单点登录时带给应用的身份标识，存在于对接的第三方业务系统中。

示例：

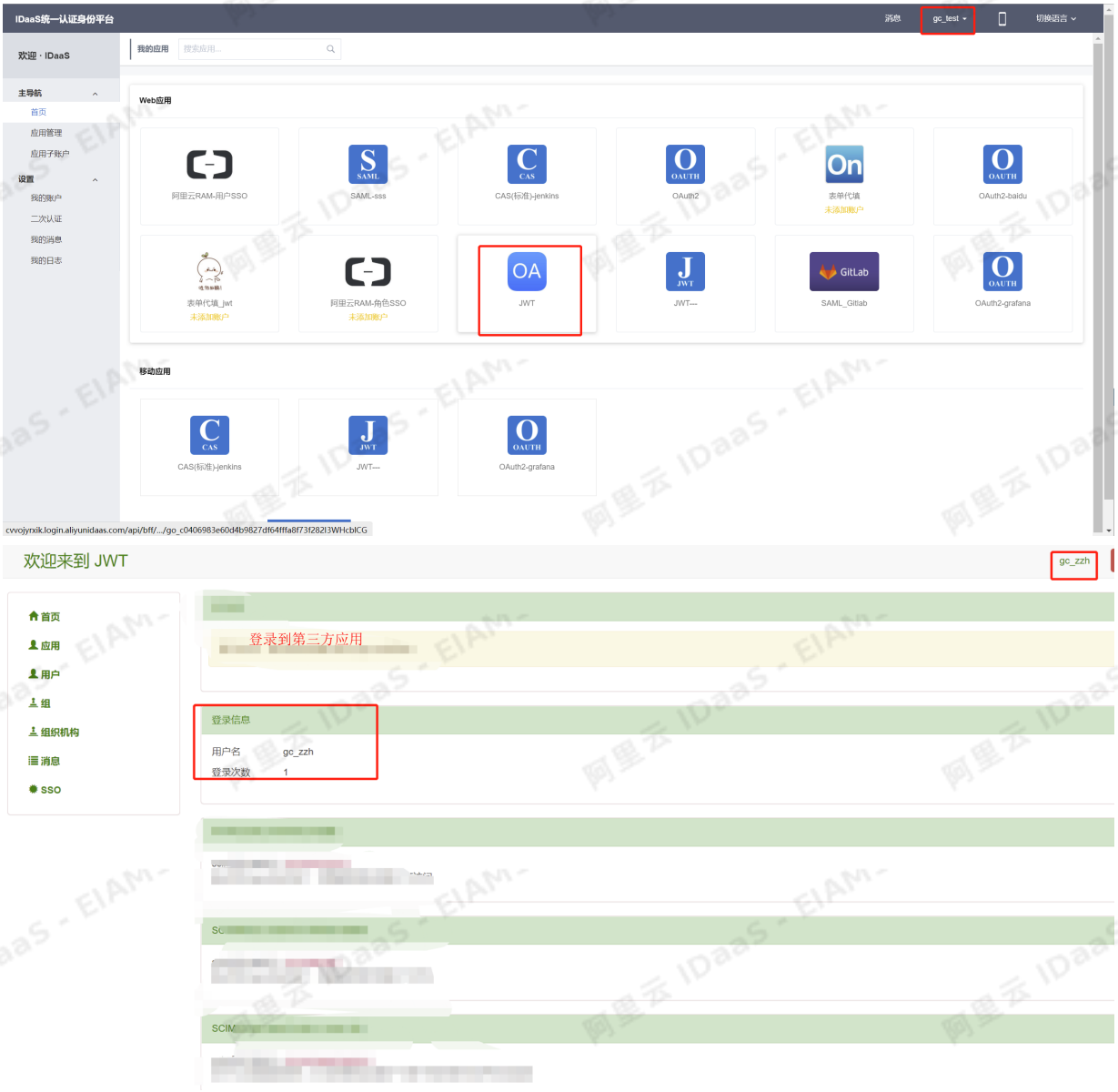
在IDaas的机构及组中创建了gc_test这个账户，该账户在绑定主子账户的时候是作为主账户存在。

在对接的第三方业务系统中，存在gc_zzh这个账户，该账户在绑定主子账户时是作为子账户存在。



从应用管理页面，点击查看应用子账户，可以进行主子账户绑定。

该主子账户绑定后，以gc_test这个账户登录IDaas用户端，点击对应的应用进行单点登录，IDaaS 会向应用系统传递对应的子账户gc_zzh,最终以gc_zzh的身份登录到第三方业务系统。

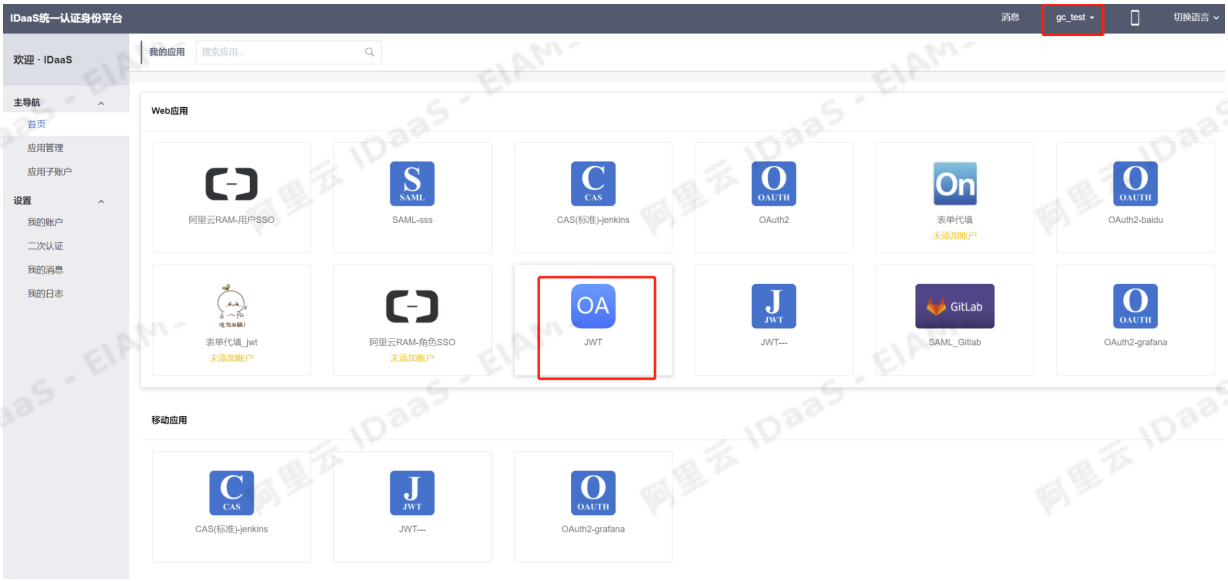


账户关联方式介绍

管理员在添加应用的时候，可以选择账户关联的方式，账户关联方式分为两种：账户关联和账户映射。

- 账户关联：系统按照子账户对应关系进行手动关联，适合主账户和子账户名称不同的情况；
- 账户映射：指系统自动将主账户名称作为应用的子账户，适合主账户和子账户名称相同的情况。

当选择账户映射时， 如下图登录IDaaS门户后，不需要手动给用户关联子账户，会自动根据主账户生成同名的子账户，并自动进行主子账户绑定。



账户关联的方式，如何进行绑定主子账户请查看手动绑定主子账户。

应用授权

创建好应用后，需要确认应用是开启状态， 并点击授权，将应用授权到IDaaS账户或者组织机构。

手动绑定主子账户

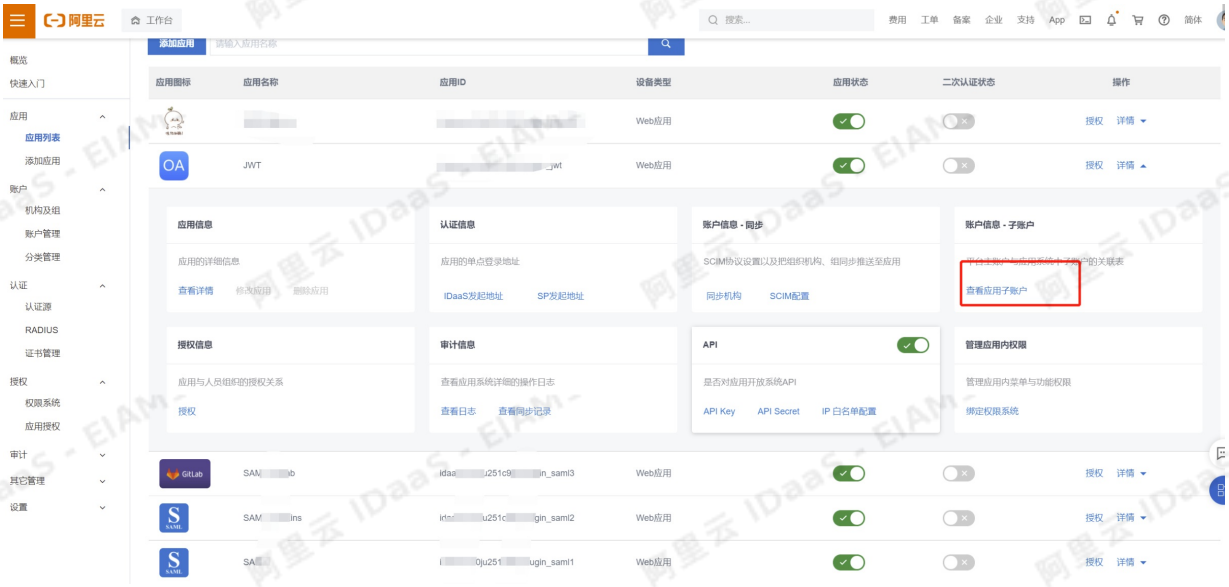
手动关联子账户可以分为两种方式：

- 由管理员直接操作。
- 由用户本人进行申请，管理员对用户的申请进行审批

1. 由管理员直接操作

1.1 手动1对1绑定

管理员点击应用的“详情”按钮，点击“查看应用子账户”即可对应用的所用子账户进管理，包括添加应用子账户操作。

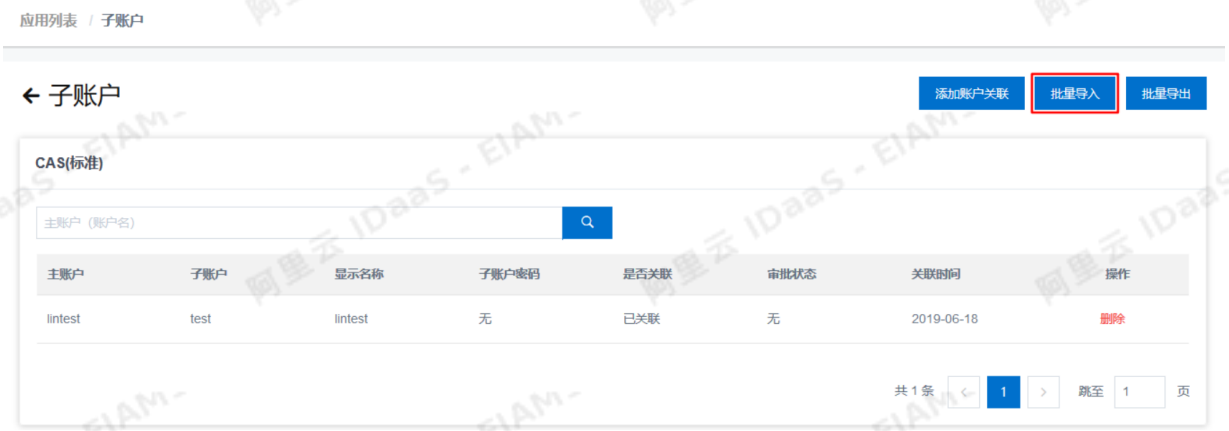


管理员可以点击“添加账户关联”按钮手动为该应用添加一个关联子账户。

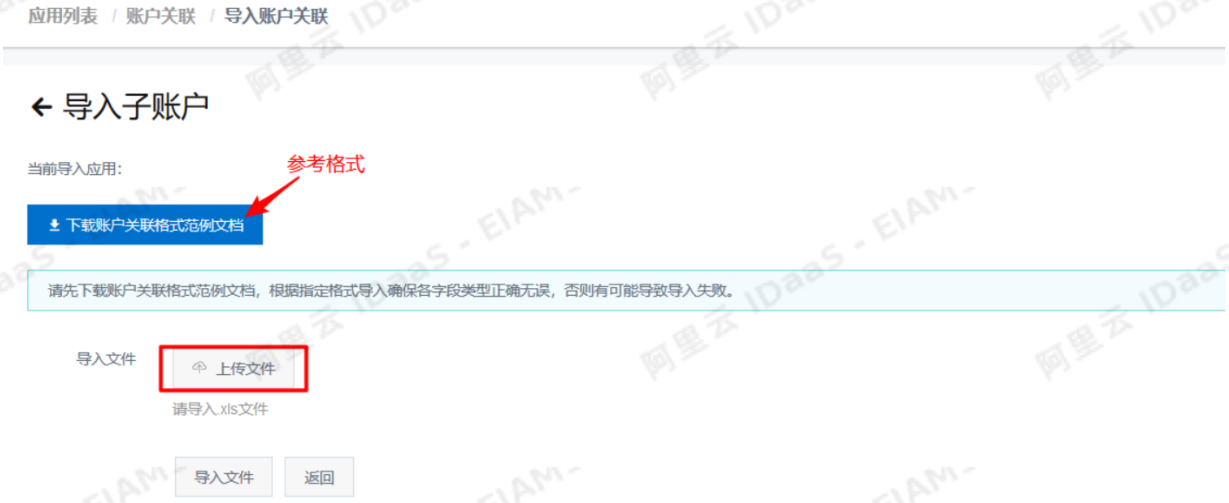
输入主账户的邮箱/手机号/账户名称以及子账户信息，点击保存按钮，即可成功添加一条账户关联。

1.2 手动批量绑定

管理员可以点击右上角“批量导入”按钮，从文件批量导入关联子账户。



点击“批量导入”按钮后，跳转到导入账户关联页面，点击上传文件，选择需要上传的文件。（上传文件的格式可以参考下载的“账户关联格式范例文档”）。



A列是主账户名称，B列是子账户名称，两个账户相互对应。

	A	B
1	主账户（IDP账户）	子账户（业务系统账户）

上传成功后，点击“导入文件”按钮。

系统会自动检测上传文件的内容，并返回每一条记录的检测结果。管理员可以查看检测结果，并根据结果修改文件，或移除某一条导入数据。确认无误之后，点击右上角“确定上传导入”即可实现批量添加应用子账户。

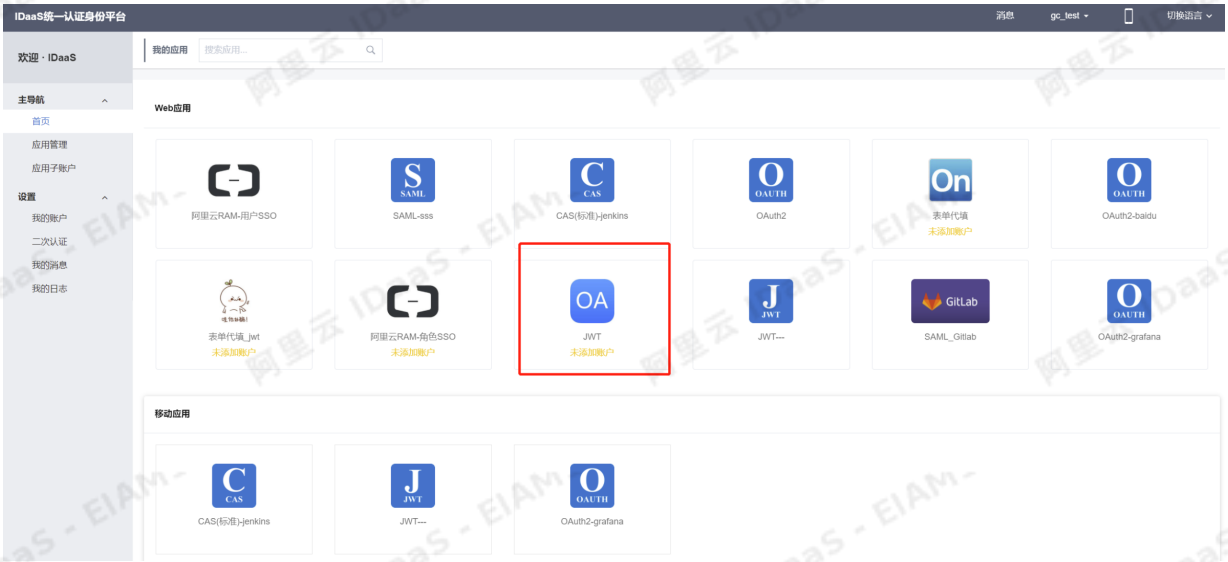
2. 由用户申请绑定主子账户

首先使用普用户登录IDaas用户端， 登录方式请参考用户登录。

- 支持在用户门户首页申请绑定主子账户
- 支持在查看子账户页面进行绑定申请

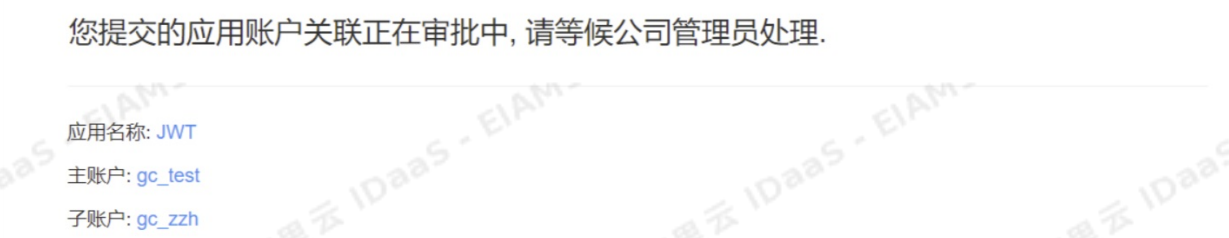
2.1 在用户门户首页申请绑定

可以在首页的免登应用栏中直接点击对应应用



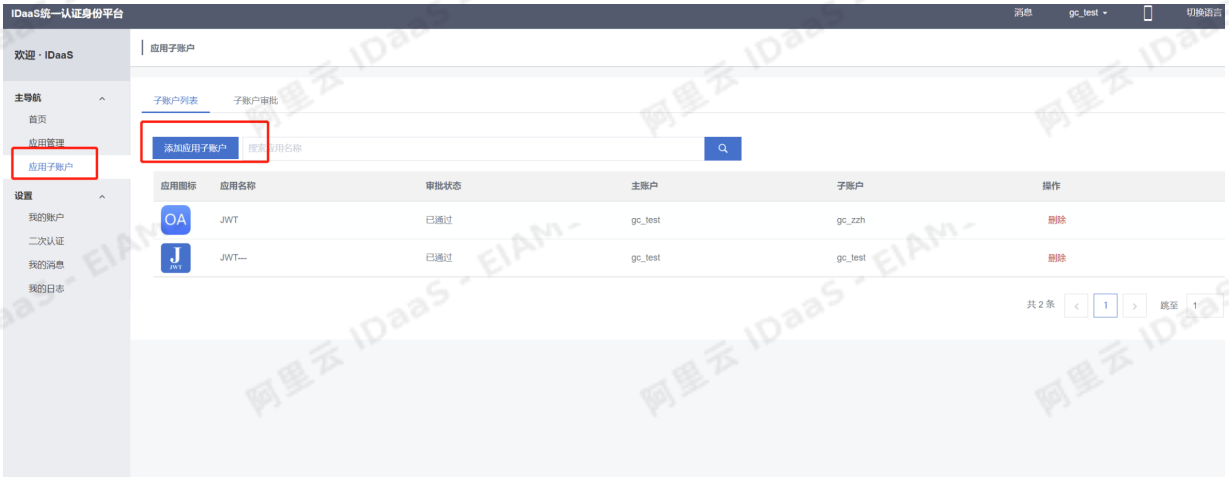
提示用户进行子账户的添加，用户输入子账户，等待管理员审批通过后即可添加该应用的子账户。

点击提交账户关联。

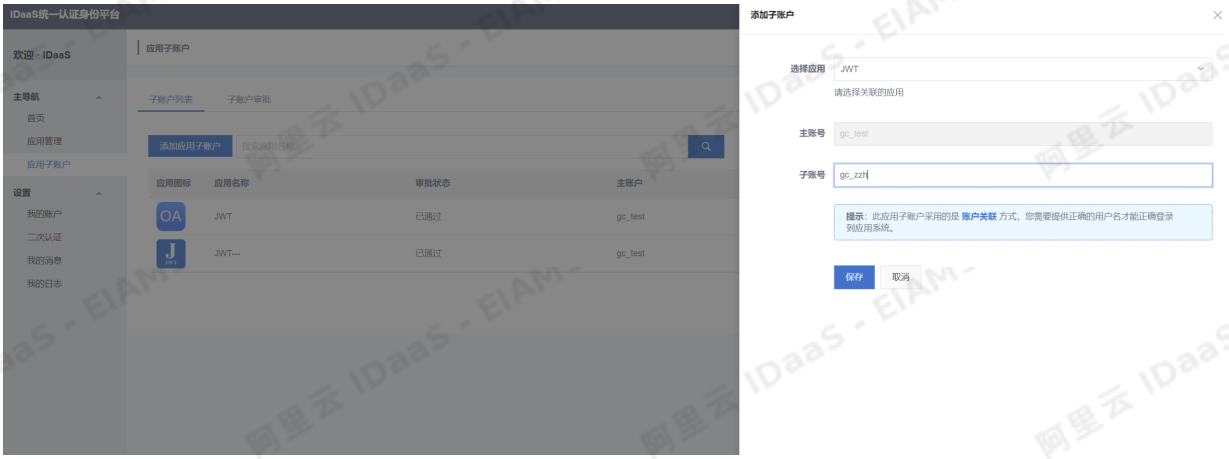


2.2 在查看子账户页面申请绑定

也可以在导航栏中选择应用子账户，点击右上角的“添加应用子账户”进行子账户的添加。



用户选择添加子账户的应用，输入子账户，点击保存按钮。等待管理员审批通过，即完成了添加子账户。



2.3 管理员进行审批

用户发出添加子账户的申请之后，管理员会收到添加子账户的申请。
管理员可以在审批中心下的子账户审批中对该用户添加子账户操作进行审批，同意申请后，用户即可成功添加应用子账户。

若以上步骤全部成功完成，即完成添加应用子账户的功能，可以使用IDaaS账户进行单点登录应用。