



# 概述

Casdoor是一个基于OAuth 2.0、OIDC、SAML 和 CAS 的，UI-first的身份和访问管理(IAM)/单点登录(SSO)平台。

You need to enable JavaScript to run this app.

Casdoor 可为网页UI和应用程序用户的登录请求提供服务。

# Casdoor 的特性：

1. 前后端分离的架构，使用 Go 语言开发，Casdoor 支持高并发，提供基于Web的用户管理 UI，并支持中、英等多种语言。
2. Casdoor 支持第三方应用登录，如 GitHub、谷歌、QQ、微信等，并支持通过插件扩展第三方登录。
3. 使用 [Casbin](#) 基于授权管理，Casdoor 支持 ACL, RBAC, ABAC, RESTful 等访问控制模型。
4. Casdoor 可通过手机验证码、邮箱验证码登录，还有找回密码功能。
5. 审查和记录访问日志。
6. 使用阿里云、腾讯云、七牛云提供的图片 CDN 云存储。
7. 个性化的注册、登录和找回密码页面。
8. Casdoor 支持通过数据库同步的方法来与现有系统集成，用户可以顺利过渡到 Casdoor。
9. Casdoor 支持主流数据库：MySQL、PostgreSQL、SQL Server 等，并支持扩展插件的新数据库。

# 如何操作?



## 步骤0（前置知识）

1. Casdoor 的授权程序建立在 OAuth 2 的基础上。因此，强烈建议简单了解 OAuth 2.0 的工作原理。详见[OAuth 2.0 简介](#)。

## Abstract Protocol Flow



### 步骤 1 (授权请求)

您的应用（或网站等）应该以 `endpoint/login/oauth/authorize?client_id=xxx&response_type=code&redirect_uri=xxx&scope=read&state=xxx` 这种格式编写 URL。在此 URL 中，使用 Casdoor 的主机 URL 替换 `endpoint`，并用您的信息替换 `xxx`。

#### ⓘ 提示

对于 `xxx` 的部分需要写上什么？

- 对于 `client_id`: 您可以在每个应用程序里找到它
- 对于 `redirect_uri`: 您应该将此设置为自己的应用程序回调URL, 通过这个信息, Casdoor 可以知道在授权后向哪里发送信息
- 对于 `state`: 您应该用您的应用程序名称填写这个内容

应用程序对用户说：“嘿，现在我需要一些资源，我需要您的许可才能拿这些资源。您愿意跳转到这个URL，填写您的用户名和密码吗？”

使用正确的URL，您的应用程序将会让用户向此 URL 发起请求，`授权请求` 已完成。

## 步骤 2（授权认证）

此步骤比较直接：用户会被重定向到上面的URL，看到来自Casdoor的登录页面。通过在登录页面输入正确的用户名和认证信息，Casdoor现在能成功识别用户，将发送`代码`和`状态`返回第1步中设置的回调URL。

用户打开网址并向Casdoor提供凭据。Casdoor说：“好～这是我在数据库中知道的用户（授权应用获取`code`和`state`）。然后我将使用回调URL发送`code`和`state`返回应用`(redirect_uri)`”

这两个关键词发到您的应用后，应用便得到授权，至此就完成了`授权`环节。



提示

Casdoor也提供第三方登录。在这种情况下，您将不会看到输入验证信息的页面，只会看到第三方提供商列表。您可以使用这些提供商登录到您的应用，而Casdoor是中间层（中间件）。

## 步骤 3（授权认证）

在这一步，您的应用程序已经有了来自第2步的代码，它将会告诉Casdoor：“嘿，现在用户同意给我`code`，你想检查这个`code`并给我`access_token`吗？

## 步骤 4（访问令牌）

这一步骤中，Casdoor会通知应用程序：“这个`code`应该是合法的，你一定就是那个正确的应用程序。这是`access_token`。”

有了`code`，Casdoor知道它是一个已授权的应用（第二步中用户给予的授权），试图获取`access_token`（稍后将会用来获取更有用的信息）。

## 步骤 5（访问令牌）

在这个步骤中，您的应用程序说：“很好，刚刚获得了最新的`access_token`我现在可以使用它从资源服务器获得更多宝贵的东西！”

您的应用程序转向 资源服务器：“嗨朋友，想检查这个 `access_token` 吗？我从 Casdoor 得到了访问令牌，你想看看这是否与 Casdoor 的一致吗？”

## 步骤 6 (受保护资源)

Resource Server 又告知您的应用程序：“不错~ 它似乎就像我在 Casdoor 中的那个一样。Casdoor 说谁拥有这个 `access_token` 谁就可以拥有这些 受保护的资源 现在，你可以自取这些资源了！”

这就是Casdoor 如何与您的应用程序一起工作。

### ⓘ 提示

Casdoor 可以同时运行 认证服务器 和 资源服务器 配件，也就是说：Casdoor 授权我们的应用程序从Casdoor的数据库获取资源（例如通常是当前登录用户的信息）。

## 在线演示

### Casdoor

这里是一个由Casbin部署的在线演示。

- [Casdoor官方演示](#)

全局管理员登录：

- 用户名：`admin`
- 密码：`123`

### Casbin-OA

Casbin-OA是Casbin 的web 应用程序之一。它使用 Casdoor 作为身份验证。

- [Casbin-OA](#)
- 源代码：<https://github.com/casbin/casbinoa>

## Casnnode

Casnnode 是Casbin社区开发的官方论坛。

它使用 Casdoor 作为认证平台并管理成员。

- [Casnnode](#)
- 源代码: <https://github.com/casbin/casnnode>

## 结构

Casdoor包含两个部分：

名称	描述	语言	源代码
前端	Casdoor的前端 Web界面	JavaScript + React	<a href="https://github.com/casdoor/casdoor/tree/master/web">https://github.com/casdoor/casdoor/tree/master/web</a>
后端	Casdoor后端 RESTful API	Golang + Beego + SQL	<a href="https://github.com/casdoor/casdoor">https://github.com/casdoor/casdoor</a>

# 核心概念

作为Casdoor的管理员，您至少应该熟悉4个核心概念：组织(Organization)，用户(User)，应用(Application) 和 提供商(Provider)。

```
flowchart LR; subgraph Organization-1; Applications-1; Users-1; end; subgraph Organization-2; Applications-2; Users-2; end; subgraph Users-1; Resources-1; Permissions-1; end; subgraph Users-2; Resources-2; Permissions-2; end; subgraph Providers; SMS; OAuth; SAML; Email; end; subgraph SMS; AWS\nAliyunCloud; end; subgraph OAuth; Github\nGoogle\nFacebook\nWeChat; end; subgraph SAML; Keycloak\nAliyunIDaaS; end; subgraph Applications-1; Forum; CMS; end; subgraph Applications-2; OA; end; Organization-1 --> Applications-1; Applications-1-->Providers; Applications-2-->Providers;
```



接下来，我们将使用演示站点：<https://door.casdoor.com>

## 组织

在Casdoor中，组织是用户和应用程序的容器。例如，一个公司的所有雇员或一个企业的所有客户都可以抽象成为一个组织。组织的类别定义如下：

```
type Organization struct {
    Owner      string `xorm:"varchar(100) notnull pk" json:"owner"`
    Name       string `xorm:"varchar(100) notnull pk" json:"name"`
    CreatedTime string `xorm:"varchar(100)" json:"createdTime"`

    DisplayName     string `xorm:"varchar(100)" json:"displayName"`
    WebsiteUrl     string `xorm:"varchar(100)" json:"websiteUrl"`
    Favicon        string `xorm:"varchar(100)" json:"favicon"`
    PasswordType   string `xorm:"varchar(100)" json:"passwordType"`
    PasswordSalt   string `xorm:"varchar(100)" json:"passwordSalt"`
    PhonePrefix    string `xorm:"varchar(10)" json:"phonePrefix"`
    DefaultAvatar  string `xorm:"varchar(100)" json:"defaultAvatar"`
    Tags           []string `xorm:"mediumtext" json:"tags"`
    MasterPassword string `xorm:"varchar(100)" json:"masterPassword"`
    EnableSoftDeletion bool `json:"enableSoftDeletion"`
    IsProfilePublic bool `json:"isProfilePublic"`

    AccountItems []*AccountItem `xorm:"varchar(2000)" json:"accountItems"`
}
```

## 用户

Casdoor 中的用户可以登录到一个应用程序。一个用户只能属于一个组织，但可以登录到该组织拥有的多个应用程序。Casdoor目前有两种类型的用户：

- 内置用户(built-in 组织下的所有用户)，例如 built-in/admin：全局管理员，在Casdoor平台上拥有完整的管理员权限。
- 其他组织下的用户，如 my-company/Alice：普通用户，只能注册、登录、登出、更改他/她自己的个人资料等。

在 Casdoor API 中，用户通常被定义为 <organization\_name>/<username>，e.g., Casdoor的默认管理员被定义为 built-in/admin。用户名权限中有一个名为 id 的属性，这是一个类似 d835a48f-2e88-4c1f-b907-60ac6b6c1b40 的 UUID，它也可以是用户通过一个应用程序选择的 ID。



如果您的应用程序仅应用于一个组织，您可以使用<username> instead of <organization\_name>/<username> 作为您整个应用程序的用户ID。

用户的类别定义如下：

```

type User struct {
    Owner      string `xorm:"varchar(100) notnull pk" json:"owner"`
    Name       string `xorm:"varchar(100) notnull pk" json:"name"`
    CreatedTime string `xorm:"varchar(100)" json:"createdTime"`
    UpdatedTime string `xorm:"varchar(100)" json:"updatedTime"`

    Id          string `xorm:"varchar(100)" json:"id"`
    Type        string `xorm:"varchar(100)" json:"type"`
    Password    string `xorm:"varchar(100)" json:"password"`
    PasswordSalt string `xorm:"varchar(100)" json:"passwordSalt"`
    DisplayName  string `xorm:"varchar(100)" json:"displayName"`
    Avatar      string `xorm:"varchar(500)" json:"avatar"`
    PermanentAvatar string `xorm:"varchar(500)" json:"permanentAvatar"`
    Email       string `xorm:"varchar(100) index" json:"email"`
    Phone       string `xorm:"varchar(100) index" json:"phone"`
    Location    string `xorm:"varchar(100)" json:"location"`
    Address     []string `json:"address"`
    Affiliation string `xorm:"varchar(100)" json:"affiliation"`
    Title       string `xorm:"varchar(100)" json:"title"`
    IdCardType  string `xorm:"varchar(100)" json:"idCardType"`
    IdCard      string `xorm:"varchar(100) index" json:"idCard"`
    Homepage   string `xorm:"varchar(100)" json:"homepage"`
    Bio         string `xorm:"varchar(100)" json:"bio"`
    Tag         string `xorm:"varchar(100)" json:"tag"`
    Region      string `xorm:"varchar(100)" json:"region"`
    Language    string `xorm:"varchar(100)" json:"language"`
    Gender      string `xorm:"varchar(100)" json:"gender"`
    Birthday    string `xorm:"varchar(100)" json:"birthday"`
    Education   string `xorm:"varchar(100)" json:"education"`
    Score       int    `json:"score"`
    Ranking     int    `json:"ranking"`
    IsDefaultAvatar bool  `json:"isDefaultAvatar"`
    IsOnline    bool  `json:"isOnline"`
    isAdmin     bool  `json:"isAdmin"`
    IsGlobalAdmin bool  `json:"isGlobalAdmin"`
    IsForbidden bool  `json:"isForbidden"`
    IsDeleted   bool  `json:"isDeleted"`
    SignupApplication string `xorm:"varchar(100)" json:"signupApplication"`
    Hash        string `xorm:"varchar(100)" json:"hash"`
    PreHash     string `xorm:"varchar(100)" json:"preHash"`

    CreatedIp    string `xorm:"varchar(100)" json:"createdIp"`
    LastSigninTime string `xorm:"varchar(100)" json:"lastSigninTime"`
    LastSigninIp  string `xorm:"varchar(100)" json:"lastSigninIp"`

    Github      string `xorm:"varchar(100)" json:"github"`
    Google      string `xorm:"varchar(100)" json:"google"`
    QQ          string `xorm:"qq varchar(100)" json:"qq"`
    WeChat      string `xorm:"wechat varchar(100)" json:"wechat"`
    Facebook    string `xorm:"facebook varchar(100)" json:"facebook"`
    DingTalk    string `xorm:"dingtalk varchar(100)" json:"dingtalk"`
    Weibo       string `xorm:"weibo varchar(100)" json:"weibo"`
    Gitee       string `xorm:"gitee varchar(100)" json:"gitee"`
    LinkedIn    string `xorm:"linkedin varchar(100)" json:"linkedin"`
    Wecom       string `xorm:"wecom varchar(100)" json:"wecom"`
    Lark        string `xorm:"lark varchar(100)" json:"lark"`
    Gitlab      string `xorm:"gitlab varchar(100)" json:"gitlab"`
    Apple       string `xorm:"apple varchar(100)" json:"apple"`
    AzureAD    string `xorm:"azuread varchar(100)" json:"azuread"`
    Slack       string `xorm:"slack varchar(100)" json:"slack"`

    Ldap        string `xorm:"ldap varchar(100)" json:"ldap"`
    Properties  map[string]string `json:"properties"`
}

```

# 应用程序

应用程序是指需要受Casdoor保护的网络服务。例如，论坛网站、OA系统、CRM系统都属于应用程序。

```
type Application struct {
    Owner      string `xorm:"varchar(100) notnull pk" json:"owner"`
    Name       string `xorm:"varchar(100) notnull pk" json:"name"`
    CreatedTime string `xorm:"varchar(100)" json:"createdTime"`

    DisplayName     string      `xorm:"varchar(100)" json:"displayName"`
    Logo           string      `xorm:"varchar(100)" json:"logo"`
    HomepageUrl   string      `xorm:"varchar(100)" json:"homepageUrl"`
    Description    string      `xorm:"varchar(100)" json:"description"`
    Organization   string      `xorm:"varchar(100)" json:"organization"`
    Cert          string      `xorm:"varchar(100)" json:"cert"`
    EnablePassword bool        `json:"enablePassword"`
    EnableSignUp   bool        `json:"enableSignUp"`
    EnableSigninSession bool      `json:"enableSigninSession"`
    EnableCodeSignin  bool      `json:"enableCodeSignin"`
    Providers     []*ProviderItem `xorm:"mediumtext" json:"providers"`
    SignupItems   []*SignupItem  `xorm:"varchar(1000)" json:"signupItems"`
    OrganizationObj *Organization `xorm:"-" json:"organizationObj"`

    ClientId      string      `xorm:"varchar(100)" json:"clientId"`
    ClientSecret   string      `xorm:"varchar(100)" json:"clientSecret"`
    RedirectUris  []string    `xorm:"varchar(1000)" json:"redirectUris"`
    TokenFormat    string      `xorm:"varchar(100)" json:"tokenFormat"`
    ExpireInHours int         `json:"expireInHours"`
    RefreshExpireInHours int      `json:"refreshExpireInHours"`
    SignupUrl     string      `xorm:"varchar(200)" json:"signupUrl"`
    SigninUrl     string      `xorm:"varchar(200)" json:"signinUrl"`
    ForgetUrl     string      `xorm:"varchar(200)" json:"forgetUrl"`
    AffiliationUrl string     `xorm:"varchar(100)" json:"affiliationUrl"`
    TermsOfUse    string      `xorm:"varchar(100)" json:"termsOfUse"`
    SignupHtml    string      `xorm:"mediumtext" json:"signupHtml"`
    SigninHtml    string      `xorm:"mediumtext" json:"signinHtml"`
}
```

每个应用程序都可以有自己的自定义注册页面，登录页等。例如，根登录页面 [/登录](#) (例如: <https://door.casdoor.com/login>) 是只针对Casdoor的内置应用程序，在页面上的标志是: [app-build-in](#)。

应用程序是用户登录到Casdoor的“入口”或“界面”。用户需要通过一个应用程序的登录页面才能登录到Casdoor。

应用程序	注册页面网址	登录页面网址
内置应用程序	<a href="https://door.casdoor.com/signup">https://door.casdoor.com/signup</a>	<a href="https://door.casdoor.com/login">https://door.casdoor.com/login</a>
casnode 论坛系统	<a href="https://door.casdoor.com/signup/app-casnode">https://door.casdoor.com/signup/app-casnode</a>	<a href="https://door.casdoor.com/login/oauth/authorize?client_id=014ae4bd048734ca2dea&amp;response_type=code&amp;redirect_uri=http://localhost:9000/callback&amp;scope=read&amp;state=casdoor">https://door.casdoor.com/login/oauth/authorize?client_id=014ae4bd048734ca2dea&amp;response_type=code&amp;redirect_uri=http://localhost:9000/callback&amp;scope=read&amp;state=casdoor</a>
casbin 的OA系统	<a href="https://door.casdoor.com/signup/app-casbin-oa">https://door.casdoor.com/signup/app-casbin-oa</a>	<a href="https://door.casdoor.com/login/oauth/authorize?client_id=0ba528121ea87b3eb54d&amp;response_type=code&amp;redirect_uri=http://localhost:9000/callback&amp;scope=read&amp;state=casdoor">https://door.casdoor.com/login/oauth/authorize?client_id=0ba528121ea87b3eb54d&amp;response_type=code&amp;redirect_uri=http://localhost:9000/callback&amp;scope=read&amp;state=casdoor</a>

## 登录 URL

通过Casdoor的内置应用程序登录到Casdoor非常容易，只需访问Casdoor服务器的主页(例如示例站点: <https://door.casdoor.com>)然后将会自动重定向您为 `/login`。但如何在前端和后端代码中为其他应用程序获取这些URL？您可以将您自己的字符串连接起来，也可以调用 Casdoor SDK 提供的一些实用功能来获取 URL：

### 1. 手动连接字符串：

- 注册页面URL
  - 注册指定的应用程序: `<your-casdoor-hostname>/signup/<your-application-name>`
  - 通过OAuth注册: `<your-casdoor-hostname>/signup/oauth/authorize?client_id=<client-id-for-your-application>&response_type=code&redirect_uri=<redirect-uri-for-your-application>&&scope=read&state=casdoor`
  - 自动注册: `<your-casdoor-hostname>/auto-signup/oauth/authorize?client_id=<client-id-for-your-application>&response_type=code&redirect_uri=<redirect-uri-for-your-application>&&scope=read&state=casdoor`
- 登录页面URL
  - 登录指定的组织: `<your-casdoor-hostname>/login/<your-organization-name>`
  - 通过OAuth登录: `<your-casdoor-hostname>/login/oauth/authorize?client_id=<client-id-for-your-application>&response_type=code&redirect_uri=<redirect-uri-for-your-application>&&scope=read&state=casdoor`

### 2. 使用前端 SDK (用于使用 React、Vue 或 Angular 的 Javascript 代码)：

`getSignupUrl()` 和 `getSigninUrl()`: <https://github.com/casdoor/casdoor-js-sdk/blob/3d08d726bcd5f62d6444b820596e2d8472f67d97/src/sdk.ts#L50-L63>

### 3. 使用后端 SDK (使用Go Java 等后端代码)：

`GetSignupUrl()` 和 `GetSigninUrl()`: <https://github.com/casdoor/casdoor-go-sdk/blob/f3ef1adff792e9a06af5682e0a3af9436ed24ed3/auth/url.go#L23-L39>

## 提供商

Casdoor是一个联合单点登录系统，通过OIDC、OAuth 和SAML支持多个身份提供者。 Casdoor 也可以通过电子邮件或短信(快速消息服务) 向用户发送验证码或其他通知。 Casdoor 使用这个概念: `Provider` 来管理所有这些第三方连接器。

目前，Casto支持的所有Provider都可以在这里找到: </docs/provider/overview>

```
type Provider struct {
    Owner      string `xorm:"varchar(100) notnull pk" json:"owner"`
    Name       string `xorm:"varchar(100) notnull pk" json:"name"`
    CreatedTime string `xorm:"varchar(100)" json:"createdTime"`

    DisplayName  string `xorm:"varchar(100)" json:"displayName"`
    Category    string `xorm:"varchar(100)" json:"category"`
    Type        string `xorm:"varchar(100)" json:"type"`
    Method      string `xorm:"varchar(100)" json:"method"`
    ClientId    string `xorm:"varchar(100)" json:"clientId"`
    ClientSecret string `xorm:"varchar(100)" json:"clientSecret"`
    ClientId2   string `xorm:"varchar(100)" json:"clientId2"`
    ClientSecret2 string `xorm:"varchar(100)" json:"clientSecret2"`

    Host     string `xorm:"varchar(100)" json:"host"`
    Port     int     `json:"port"`
    Title    string `xorm:"varchar(100)" json:"title"`
    Content  string `xorm:"varchar(1000)" json:"content"`

    RegionId  string `xorm:"varchar(100)" json:"regionId"`
    SignName  string `xorm:"varchar(100)" json:"signName"`
    TemplateCode string `xorm:"varchar(100)" json:"templateCode"`
}
```

## Casdoor是如何自我管理的?

当您首次运行Casdoor时， Casdoor将创建一些内置的对象来帮助管理员管理Casdoor本身：

- 一个内置的命名为 `built-in` 的组织。
- `built-in` 组织下用户名为 `admin` 的用户。
- 一个内置的应用程序名为 `app-built-in`, 由 `built-in` 组织所拥有, 代表Casdoor 本身(实际上也是一个应用程序)。

`built-in` 组织中的所有用户，包括 `admin` 默认情况下将在Casdoor平台上拥有完整的管理员权限。所以，如果你有多个管理员，可在 `built-in` 机构下创建新帐户。否则，请记住关闭 `app-built-in` 应用程序的注册功能。

### ⚠ 注意事项

内置对象已被禁止在网页UI或 RESTful API中重命名或删除。 Casdoor在许多地方硬编码了这些保留的名称。不要试图重命名或删除它们，比如修改数据库，否则整个系统可能崩溃。

# 服务器安装

## 安装要求

### 操作系统

支持所有主流的操作系统，包括Windows、Linux和macOS。

### 环境

- Go 1.6+
- Node.js LTS (16或14)
- Yarn 1.x

#### 信息

我们强烈建议您使用 [Yarn 1.x](#) 运行 & Casdoor 前端，使用 NPM 可能会导致UI 风格问题。更多详细信息见：[Casdoor#294](#)

#### 注意事项

对于中国大陆用户，为了成功下载依赖关系包，您需要通过配置 GOPROXY 环境变量来使用Go 代理。我们强烈建议使用：<https://goproxy.cn/>

## 数据库

Casdoor 使用 [XORM](#) 与数据库进行交互。基于 [Xorm Drivers Support](#)，当前支持的数据包括：

- MySQL
- MariaDB
- PostgreSQL
- SQL Server
- Oracle
- SQLite 3
- TiDB

## 下载

Casdoor的源代码托管在 GitHub: <https://github.com/casdoor/casdoor>。 Go 后端代码和 React 前端代码都在单个仓库中。

名称	描述	语言	源代码
前端	Casdoor的网页前端界面	JavaScript + React	<a href="https://github.com/casdoor/casdoor/tree/master/web">https://github.com/casdoor/casdoor/tree/master/web</a>
后端	Casdoor的ResTful API 后端	Golang + Beego + XORM	<a href="https://github.com/casdoor/casdoor">https://github.com/casdoor/casdoor</a>

Casdoor支持 Go Modules。要下载代码，您直接通过git克隆仓库就可以了：

```
cd /文件夹路径/  
git clone https://github.com/casdoor/casdoor
```

# 配置

## 配置数据库

Casdoor支持MySQL, msSQL, Sqlite3, PostgreSQL等数据库。默认使用MySQL。如果您想使用支持以外的数据库, 请自行修改object/adapter包

### MySQL:

Casdoor将会把users, nodes和topics信息存储在一个命名为casdoor的MySQL数据库中。如果数据库不存在, 则需手动创建。数据库配置信息可以在<https://github.com/casdoor/casdoor/blob/master/conf/app.conf> 中修改

```
driverName = mysql
dataSourceName = root:123456@tcp(localhost:3306)-
dbName = casdoor
```

### PostgreSQL:

因为在使用xorm打开Postgres时我们必须选择一个数据库, 所以您应该在运行Casdoor前手动准备一个数据库

假设您已经准备了一个名为casdoor的数据库, 您应该这样编写app.conf:

```
driverName = postgres
dataSourceName = "user=postgres password=postgres host=localhost
port=5432 sslmode=disable dbname=casdoor"
dbName =
```

### ① 信息

对于PostgreSQL, 请确保 `dataSourceName` 中的 `dbName` 不是空值, 并且与上面例子一样将单独 `dbName` 配置留空

## Sqlite3

首先你要将在 `object/adapter.go` 中导入MySQL包的内容注释掉, 改为导入`sqlite3`包。像这样:

```
//_ "github.com/go-sql-driver/mysql" // db = mysql  
_ "github.com/mattn/go-sqlite3" // db = sqlite3
```

接着您应该像这样修改 `app.conf`:

```
driverName = sqlite3  
dataSourceName = "file:casdoor.db?cache=shared&mode=memory"  
dbName =
```

## 通过 Ini 文件配置

可以使用一个名 `conf/app.conf` 的文件配置Casdoor。文件默认内容如下:

```
appname = casdoor  
httpport = 8000  
runmode = dev  
SessionOn = true  
copyrequestbody = true  
driverName = mysql  
dataSourceName = root:123456@tcp(localhost:3306)/  
dbName = casdoor
```

- `appname` 是应用程序名称，目前没有实际使用
- `httpport` 是您后端应用程序正在监听的端口
- `runmode` 是 `dev` 或 `prod`
- `SessionOn` 控制是否启用会话，默认为使用。
- `driverName`, `dataSourceName` 和 `dbName` 之前已经介绍，请参阅 [Configure Database](#).
- `verificationCodeTimeout` 设置验证码到期时间。超时之后用户需要再次获取验证码。

尽管有很多可配置的字段，作为初学者，您只需要基于您使用的数据库修改两个字段：`driverName` 和 `dataSourceName`。此数据库将被Casdoor用于存储所有数据，包括用户、组织、应用程序等等。

- `tableNamePrefix` 是使用适配器时表格的前缀。
- `showSql`：如果日志级别大于INFO 则显示 SQL 语句或不在 Logger 上。
- `redisEndpoint` 后填写Redis地址，被Beego用于session存储 如果此参数为空，session数据将被储存在 `./tmp` 文件夹中 使用Redis作为Beego的session存储，此参数的示例为：`redis.example.com:6379`。如果Redis启用了密码，请填写 `redis.example.com:6379, db, password`。详情请参阅 <https://beego.vip/docs/module/session.md#saving-provider-config>
- `defaultStorageProvider` 是默认的文件存储服务名称。如果您需要使用文件存储服务，例如 `头像上传`，您需要设置存储提供商，并在您的 `应用` 中应用它。详情请参阅 [存储](#)
- `isCloudIntranet` 用于确定您的提供者端是否是内联网端点。
- `authstate` 是授权应用程序名称。登录时将检查该参数。
- `socks5Proxy` 是 SOCKS 代理服务器 IP 地址。设置代理端口，因为我们有与谷歌相关的服务或使用 `Google` `GitHub` `Facebook` `LinkedIn` `Steam` 作为OAuth Provider，在某些领域将受到网络限制。
- `initscore` 是每个用户的最初分数。每个用户都有一个得分属性。得分被

`Casnode` 所使用。 分数无法控制 Casdoor 中的任何东西。

- `logPostOnly` 用于识别是否只使用帖子方法添加记录。
- `origin` 是origin形式的后端域名
- `staticBaseUrl` 是系统初始化数据库时静态图像的地址。

## 通过环境变量配置

以上提到的在ini文件里所有被Casdoor使用的参数也能通过环境变量配置，包括一些beego 配置项(`httpport,appname`)。

例如，当您尝试启动Casdoor时，您可以像如下所示通过环境变量传递配置

```
appname=casbin go run main.go
```

此外，`export` 命令也是一种可行的方法。 环境变量的名称应该与您在ini 文件中使用的名称完全相同。

*Note:* 环境变量的配置将会 覆盖 ini文件中的配置.

## 运行

当前有两种启动方法，您可以根据自己的情况选择一个。

### 开发模式

#### 后端

Casdoor后端默认端口是 8000。 您可以通过以下命令启动后端：

```
go run main.go
```

在服务器成功运行后，我们可以开始前端部分。

## 前端

Casdoor的前端是一个非常典型的 [Create-React-App \(CRA\)](#) 项目。默认情况下运行在 [7001](#) 端口。使用以下命令启动前端

```
cd web  
yarn install  
yarn start
```

在浏览器中访问<http://localhost:7001> 使用默认的全局管理帐户登录 Casdoor 控制面板：[built-in/admin](#)

```
admin  
123
```

## 生产模式

### 后端

将 Casdoor Go 后端代码构建为可执行文件并启动它。

#### Linux 环境

```
去构建  
. ./casdoor
```

#### Windows 环境

```
go build  
casdoor.exe
```

## 前端

构建Casdoor前端代码为静态资源 (.html、.js, .css 文件):

```
cd web  
yarn install  
yarn build
```

使用您的浏览器访问<http://localhost:8000> 使用默认的全局管理账户登录 Casdoor 控制面板: `built-in/admin`

```
admin  
123
```



提示

若想要使用另一个端口, 请修改 `conf/app.conf` 中的 `httpport`, 之后重启后端

### ❗ CASDOOR 端口详细信息

在 `dev` 环境中, 前端通过 `yarn run` 运行在 7001 端口, 所以如果您想要进入 Casdoor 登录页面, 需要访问 <http://localhost:7001>。

在 `prod` 环境中, 前端文件首先由 `yarn build` 构建, 服务端口号 8000 , 如果您想要打开 Casdoor 登录页面, 您需要访问的地址为 [http://SERVER\\_IP:8000](http://SERVER_IP:8000) (如果您正在使用反向代理, 您需要将链接设置为 [您的域名](#))

以我们的官方论坛 Casnode 为例：

Casnnode 使用 Casdoor 进行身份认证。

当我们在 `dev` 环境中测试 Casnode 时，我们将 `serverUrl` 设置为 `http://localhost:7001`，所以当我们使用 Casdoor 测试登录和注册功能时，它将会转到本地主机的 7001 端口，7001 为 Casdoor 的端口

当我们将 Casnode 部署到 `prod` 环境时，我们将 `serverUrl` 设置为 `https://door.casdoor.com`，用户就可以使用 Casdoor 登录或注册了。

```
|4 import * as ConfBackend from "./backend/ConfBackend.js"
|5
|6 export const AuthConfig = {
|7   // serverUrl: "https://door.casbin.com",
|8   serverUrl: "http://localhost:7001",
|9   clientId: "014ae4bd048734ca2dea",
|10 }
```



> 基础知识

> (可选) 使用 Docker 运行

# (可选) 使用 Docker 运行

## 安装要求

### 硬件

如果您想要自己构建Docker镜像, 请确保您的机器至少有**2GB** 的内存。Cassdoor的前端是React的一个NPM项目。构建前端至少需要 **2GB** 的内存。低于 **2GB** 的内存可能导致前端构建失败。

如果您只需要运行预编译的镜像, 请确保您的机器至少有**100MB** 的内存。

### 操作系统

支持所有操作系统 (Linux, Windows 和 macOS)

### Docker

在Linux系统中, 您可以使用**docker** (要求**docker-engine** 版本不低于 17.05), 在 Windows和mac系统中可以使用 **Docker Desktop**。

- [Docker](#)

所有操作系统的用户必须确保 **docker-engine** 版本不低于17.05。这是因为我们在 **docker-compose.yml** 中使用多阶段构建功能, 这个功能在17.05及以上版本中得到支持。更多信息请参阅 <https://docs.docker.com/develop/develop-images/multistage-build/>

如果你使用**docker-compose**, 请确保 **docker-compose** 版本不低于2.2 对于Linux用

户，考虑到docker-compose与docker-engine相分离，你还需要确保已安装docker-compose。

## 获取镜像

我们提供了两个DockerHub 镜像：

名称	描述	建议
casdoor-all-in-one	Casdoor 和 MySQL 数据库都在镜像内	已经包含示例数据库，仅用于测试
casdoor	只有casdoor在镜像里	可以连接到您自己的数据库并用于生产

1. 在casbin/casdoor-all-in-one中，casdoor二进制文件、mysql数据库和所有必要的配置都打包在一起。这个镜像是为了让新用户能够快速试用Casdoor。通过这个镜像，您可以使用一两个命令而不是复杂的配置即可启动Casdoor。注意：我们不建议您在生产环境中使用此镜像。

### 选项 1: 使用示例数据库

使用端口 8000 运行容器。如果本地host中不存在，它将自动下载镜像。

```
docker run -p 8000:8000 casbin/casdoor-all-in-one
```



注意事项

中国等地区的一些用户通常使用 Docker 镜像服务，例如 [Alibaba Cloud Image Booster \(English\)](#) 实现比DockerHub 更高的下载速度。然而，有一个问题，那些服务提供的 `latest` 标签不是最新的。它可能会通过获取 `latest` 标签来产生一个非常旧的镜像。为了缓解这个问题，您可以使用以下命令明确指定镜像版本号：

```
docker pull casbin/casdoor-all-in-one:$(`curl -ss "https://hub.docker.com/v2/repositories/casbin/casdoor-all-in-one/tags/?page_size=1&page=2" | sed 's/,/,\\n/g' | grep '"name"' | awk -F '"' '{print $4}'`)
```

注意：上面的命令使用的 Linux 工具，例如 `curl`, `ed`, `grep`, `awk`. 如果您正在使用 Windows，请确保您在 Linux 样式外壳中运行，如 `Git Shell` 或 `Cygwin`。`CMD` 或 `PowerShell` 将无法工作。

在您的浏览器中访问：<http://localhost:8000>。 使用默认的全局管理员帐户登录 Casdoor 仪表板：`built-in/admin`

admin

123

## 选项 2: 使用 docker-compose

### ⚠ 注意事项

中国等地区的一些用户通常使用 Docker 镜像服务，例如 [Alibaba Cloud Image Booster \(English\)](#) 实现比DockerHub 更高的下载速度。然而，有一个问题，那些服务提供的 `latest` 标签不是最新的。它可能会通过获取 `latest` 标签来产生一个非常古老的图像。为了缓解这个问题，您可以使用以下命令明确指定图像

版本号:

```
docker pull casbin/casdoor:$(  
curl -sS  
"https://hub.docker.com/v2/repositories/casbin/casdoor/  
tags/?page_size=1&page=2" | sed 's/,/,\\n/g' | grep '"name"'  
| awk -F '"' '{print $4}' )
```

注意: 上面的命令使用的 Linux 工具, 例如 curl, ed, grep, awk。如果您正在使用 Windows, 请确保您在 Linux 样式外壳中运行, 如 Git Shell 或 Cygwin。CMD 或 PowerShell 将无法工作。

在与 docker-compose.yml 同级目录下创建 conf/app.conf 文件, 然后从 Casdoor 复制 app.conf。您可以从 [Via Ini file](#) 找到 app.conf 的详细信息。

通过 docker-compose 创建一个独立的数据库:

```
docker-compose up
```

这样就完成了! 🎉

在您的浏览器中访问 <http://localhost:8000>。 使用默认的全局管理账户登录 Casdoor 控制面板: built-in/admin

```
admin  
123
```

注意: 如果你深入了解 docker-compose.yml, 你可能会对我们创建的称为 "RUNNING\_IN\_DOCKER" 的环境变量感到困惑。当数据库 'db' 是通过 docker-compose 创建时, 它可以在您的 pc 的本地主机上使用, 而不是连带容器的本地主机上使用。这是为了防止你因修改应用程序而引起问题。我们提供这个环境变量, 并且在 docker-

`compose.yml`中预先分配了它。当此环境变量为 `true` 时，本地主机将被替换为 `host.docker.internal`，以便您可以访问 `db`。

## 选项3 直接尝试使用标准镜像

### ⚠ 注意事项

中国等地区的一些用户通常使用 Docker 镜像服务，例如 [Alibaba Cloud Image Booster \(English\)](#) 实现比 DockerHub 更高的下载速度。然而，有一个问题，那些服务提供的 `latest` 标签不是最新的。它可能会通过获取 `latest` 标签来产生一个非常旧的镜像。为了缓解这个问题，您可以使用以下命令明确指定镜像版本号：

```
docker pull casbin/casdoor:$(curl -sS "https://hub.docker.com/v2/repositories/casbin/casdoor/tags/?page_size=1&page=2" | sed 's/,/,\\n/g' | grep '"name"' | awk -F '"' '{print $4}')
```

注意：上面的命令使用的 Linux 工具，例如 `curl`, `sed`, `grep`, `awk`。如果您正在使用 Windows，请确保您在 Linux 样式外壳中运行，如 `Git Shell` 或 `Cygwin`。`CMD` 或 `PowerShell` 将无法工作。

### 💡 提示

如果不方便挂载配置文件，使用环境变量也是一种可能的解决方法。

#### example

```
docker run \
-e driverName=mysql \
```

创建 `conf/app.conf` 文件，您可以从 `conf/app.conf` 复制。关于 `app.conf` 的更多信息，您可以访问 [Via Ini file](#).

然后运行

```
docker run -p 8000:8000 -v /folder/of/app.conf:/conf casbin/casdoor:latest
```

将 `app.conf` 挂载到 `/conf/app.conf` 并启动它。

使用您的浏览器访问 `http://localhost:8000` 使用默认的全局管理账户登录 Casdoor 控制面板： [built-in/admin](#)

```
admin  
123
```

# Casdoor公共API

Casdoor采用前后端分离的模式开发（与JSP或PHP不同）。只有通过 RESTful API才能显示其功能。React前端代码通过调用RESTful API来渲染Web UI并执行操作。这个 RESTful API接口被称为 `Casdoor Public API`。这个API被用在以下几个地方：

- Casdoor前端页面
- Casdoor Client SDK
- 应用方自定义的任何代码

`Casdoor Public API`的完整参考文档可以在 <https://door.casdoor.com/swagger> 中查看。这个Swagger文档是由Beego的Bee工具自动生成的。



&gt;

Deployment

# Deployment



## 数据初始化

如何从文件初始化Casdoor 数据



## 将静态文件托管到CDN

在CDN中托管前端静态文件



## 将静态文件托管在局域网内

如何部署Casdoor静态资源

# 数据初始化

如果您将Casdoor和其他服务作为一个应用整体进行部署，您可能想要为用户提供开箱即用的功能(用户不需要任何配置就可以直接使用应用程序)。

在这种情况下，您可以使用数据初始化功能，通过一个配置文件将您的服务注册到Casdoor。此文件可以由服务所有者预定义或动态生成。

## 使用方式

如果在 Casdoor 的根目录下有一个名为 `init_data.json` 的配置文件，它将被用于初始化 Casdoor 中的数据。您要做的只是将此文件放到运行 Casdoor 的根目录下。

如果您使用 Casdoor 的官方 `docker` 镜像，以下脚本可以帮助您将 `init_data.json` 挂载到容器里。

## Docker

如果您使用 `docker` 部署 Casdoor，您可以使用 `卷` 将 `init_data.json` 挂载到容器中。

```
docker run ... -v /path/to/init_data.json:/init_data.json
```

## Kubernetes

如果您使用 Kubernetes 部署了 Casdoor，您可以使用 `configmap` 来存放 `init_data.json`。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: casdoor-init-data
data:
  init_data.json:
```

您可以通过挂载 configmap 将数据挂载到Casdoor的 pods 中。 您可以按照如下例子修改 deployment 的定义文件：

```
apiVersion: apps/v1
kind: Deployment
...
spec:
  template:
    ...
      spec:
        containers:
          ...
            volumeMounts:
              - mountPath: /init_data.json
                name: casdoor-init-data-volume
                subPath: init_data.json
        volumes:
          - configMap:
              name: casdoor-init-data
              name: casdoor-init-data-volume
```

## 文件内容

Casto版本库的根目录中已经有一个名为 init\_data.json.template 的模板文件。 您可以参考此文件来自定义您的数据初始化文件。

以下是每个对象对应的Go结构体和文档链接：

对象	Go 结构体	文档
组织机构	<a href="#">stuct</a>	<a href="#">doc</a>
应用	<a href="#">stuct</a>	<a href="#">doc</a>
用户	<a href="#">stuct</a>	<a href="#">doc</a>
提供商	<a href="#">stuct</a>	<a href="#">doc</a>
证书	<a href="#">stuct</a>	
Idaps	<a href="#">stuct</a>	<a href="#">doc</a>

如果您对填写此模板仍然感到困惑， 您可以调用 restful api或使用浏览器的调试模式来查看这些对象的 `GetXXX` 的响应。 这些响应和 `init_data.json` 的内容是相同的。

# 将静态文件托管到CDN

前端静态资源 (.js, .css 文件) 在 `web/build/static/` 目录下。如果您想要将它部署在公共云服务CDN中 Casdoor 提供了一个脚本，让您可以轻松地部署前端静态文件 请按照以下步骤操作：

## ① 备注

我们假定您已经构建过了Casdoor的前端代码。如果还没有，请参照：[文档](#)。

## 准备工作

首先，您需要在 Casdoor UI 中创建一个有效的 [存储提供商](#)。您可以参考 [示例](#)。

### ⚠ 注意事项

当您填写 `域` 字段时，请以 '/' 结束

Domain [?](#) :

<https://cdn.casbin.com/casdoor/>

## 使用说明

脚本放在文件[deployment/deploy\\_test.go](#)中。

你需要修改 `deploy_test.go` 中传入 `GetProvider()` 方法的参数 `id` 的值。提供商 `id` 的格式是 `<owner>/<name>`

```
func TestDeployStaticFiles(t *testing.T) {
    provider := object.GetProvider("admin/
provider_storage_aliyun_oss")
    deployStaticFiles(provider)
}
```

然后使用以下命令来运行脚本：

```
cd deployment
go test
```

如果执行成功，您将看到：

```
PASS
ok      github.com/casdoor/casdoor/deployment  2.951s
```

## 工作原理

脚本的功能：

- 它将会上传文件夹： `css/` and `js/` 中所有文件到指定的存储提供商的 CDN 服务上。
- 替换 `web/build/index.html` 中所有 `.css` 和 `.js` 的URL为托管CDN的URL。

您仍然需要保留 `index.html`。静态文件上传到CDN后，用户通过 Casdoor Go 后端请求 `index.html` 这些托管在CDN中的静态文件将通过 `index.html` 的URL被请求加载。

# 将静态文件托管在局域网内

如果您在 **内网** 上部署了 Casdoor，您可能无法直接通过互联网访问静态资源。您需要将静态资源部署在能够访问到它们的地方，然后在 Casdoor 中 3 处地方的配置。

## 部署静态资源

Casdoor 的所有静态资源，包括图像、标志、css 等，都存储在 [casbin/static repository](#) 仓库中。

克隆仓库，在网页服务器上部署静态资源。请确保您可以访问该资源。

## 配置 Casdoor

您可以简单地修改配置文件，将静态资源地址设置为您部署的地址。转至 [conf/app.conf](#)，修改 `staticBaseUrl`。

```
staticBaseUrl = "https://cdn.casbin.org"
```



> 如何连接到Casdoor

# 如何连接到Casdoor

## 概览

将您的应用连接到Casdoor

## 标准OIDC 客户端

使用 OIDC 发现迁移到Casdoor

## Casdoor SDK

使用 Casdoor SDK 代替标准的 OIDC 协议

## 如何启用单点登录

启用单点登录



## Vue SDK

Casdoor Vue SDK



## 桌面 SDK

3 个项目



## Casdoor插件

在 Spring Boot, WordPress, Odoo 等其他框架中使用 Casdoor 插件或中间件。



## OAuth 2.0

使用AccessToken验证客户端



## CAS

使用 Casdoor 作为 CAS 服务器

 **SAML**

使用 Casdoor 作为 SAML IdP

# 概览

在本节中，我们将显示如何将您的应用程序连接到Casdoor。

作为服务提供商(SP)，Casdoor 支持两项认证协议：

- OAuth 2.0 (OIDC)
- SAML

As Identity Provider (IdP), Casdoor supports 4 authentication protocols:

- OAuth 2.0
- OIDC
- SAML
- CAS 1.0, 2.0, 3.0

## OAuth 2.0 (OIDC)

What is OAuth 2.0?

OAuth 2 is an authorization framework that enables applications — such as Facebook, GitHub, and Casdoor — to obtain limited access to user accounts on an HTTP service. It works by delegating user authentication to the service that hosts a user account and authorizing third-party applications to access that user account. OAuth 2 provides authorization flows for web and desktop applications, as well as mobile devices.

Casdoor's authorization process is built upon the OAuth 2.0 protocol. We recommend using the OAuth 2.0 protocol:

1. The protocol is simple and easy to implement, and can solve many scenarios.
2. High maturity and extensive community support

Therefore, your application will talk to Casdoor via OAuth 2.0 (OIDC). Specifically, there are three ways for connecting to Casdoor:

## Standard OIDC client

**Standard OIDC client**: use a standard OIDC client implementation, which is usually widely provided in any programming language or framework.

What is OIDC?

OpenID Connect (OIDC) is an open authentication protocol that works on top of the OAuth 2.0 framework. Targeted toward consumers, OIDC allows individuals to use single sign-on (SSO) to access relying party sites using OpenID Providers (OPs), such as an email provider or social network, to authenticate their identities. It provides the application or service with information about the user, the context of their authentication, and access to their profile information.

Casdoor has fulfilled the OIDC protocol completely. If your application is already running against another OAuth 2.0 (OIDC) identity provider via a **standard OIDC client library**, and you want to migrate to Casdoor, using OIDC discovery will be very easy for you to switch to Casdoor.

## Casdoor SDK

[Casdoor SDK](#): For most programming languages, Casdoor will provide easy-to-use SDK library on top of OIDC, with supporting extended functionality which are only available in Casdoor.

Compared to the standard OIDC protocol, Casdoor provides more functionalities in its SDK, like user management, resource uploading, etc. Connecting to Casdoor via Casdoor SDK costs more time than using a standard OIDC client library but will provide the best flexibility and the most powerful API.

## Casdoor plugin

[Casdoor plugin](#): if your application is built on top of a popular platform (like Spring Boot, WordPress, etc.) and Casdoor (or a third-party) has already provided a plugin or middleware for it, then use it. It will be much easier to use a plugin than manually invoking Casdoor SDK because the former is specially made for the platform.

plugin:

- [Jenkins plugin](#)
- [APISIX plugin](#)

Middleware:

- [Spring Boot plugin](#)
- [Django plugin](#)

# SAML

## What is SAML?

Security Assertion Markup Language (SAML) is an open standard that allows identity providers (IdP) to pass authorization credentials to service providers (SP). What that jargon means is that you can use one set of credentials to log into many different websites. It's much simpler to manage one login per user than it is to manage separate logins to email, customer relationship management (CRM) software, Active Directory, etc.

SAML transactions use Extensible Markup Language (XML) for standardized communications between the identity provider and service providers. SAML is the link between the authentication of a user's identity and the authorization to use a service.

Casdoor can be used as SAML IdP. Up to now the Casdoor has supported the main features of SAML 2.0. More details see [SAML](#).

Example:

[Casdoor as a SAML IdP in keycloak](#)

Suggestion:

1. The protocol is **powerful** and covers many scenarios, which can be said to be one of the most comprehensive SSO protocols.
2. The protocol is **too large**, and there are many optional parameters, so it is difficult to cover all application scenarios 100% in the actual implementation.
3. If the application is **newly developed**, SAML is **not recommended** because of

- ▶ its high technical complexity.

## CAS

### What is CAS?

The Central Authentication Service (CAS) is a single sign-on protocol for the web. Its purpose is to permit a user to access multiple applications while providing their credentials (such as user ID and password) only once. It also allows web applications to authenticate users without gaining access to a user's security credentials, such as a password.

Casdoor has implemented CAS 1.0, 2.0, 3.0 features. More details see [CAS](#).

### Suggestion:

1. The protocol itself is relatively lightweight and easy to implement, but it can solve a single scenario.
2. The mutual trust between the CAS Client and the CAS Server is established through interface invocation without any encryption or signature mechanism to ensure further security.
3. CAS protocol has no advantage over other protocols.

## Integrations table

Some applications already have examples that connect to Casdoor. You can follow the documentation to quickly connect to Casdoor. You can see all applications in [Integrations table](#).

# 标准OIDC 客户端

## OIDC discovery

Casdoor 完全实现了OIDC协议。如果您的应用程序已经运行了另一个 OAuth 2，那么 (OIDC) 身份提供商一般会通过标准的 OIDC 客户端库提供服务，如果您想要迁移到 Casdoor，使用 OIDC discovery 会帮助您非常容易地切换到Casdoor。Casdoor's OIDC discovery URL 是：

```
<your-casdoor-backend-host>/.well-known/openid-configuration
```

例如，演示站点的 OIDC discovery URL是：<https://door.casdoor.com/.well-known/openid-configuration>，具有以下内容：

```
{  
  "issuer": "https://door.casdoor.com",  
  "authorization_endpoint": "https://door.casdoor.com/login/oauth/authorize",  
  "token_endpoint": "https://door.casdoor.com/api/login/oauth/access_token",  
  "userinfo_endpoint": "https://door.casdoor.com/api/userinfo",  
  "jwks_uri": "https://door.casdoor.com/.well-known/jwks",  
  "introspection_endpoint": "https://door.casdoor.com/api/login/oauth/introspect",  
  "response_types_supported": [  
    "code",  
    "token",  
    "id_token",  
    "code token",  
    "code id_token",  
  ]  
}
```

# OIDC 客户端库列表

这里我们列出了一些OIDC 客户端库，如Go 和 Java 等语言：

OIDC 客户端库	语言	链接
go-oidc	Go	<a href="https://github.com/coreos/go-oidc">https://github.com/coreos/go-oidc</a>
pac4j-oidc	Java	<a href="https://www.pac4j.org/docs/clients/openid-connect.html">https://www.pac4j.org/docs/clients/openid-connect.html</a>

上表远远没有完成。 OIDC 客户端库的完整列表请查看更多详情：

1. <https://oauth.net/code/>
2. <https://openid.net/>
  - i. 认证的 OpenID Connect 实现
  - ii. 未认证的 OpenID Connect 实现





# Casdoor SDK

## 简介

与标准的OIDC协议相比，Casdoor在SDK中提供了更多的功能，如用户管理、资源上传等。通过Casdoor SDK连接到Casdoor的成本比使用OIDC标准客户端库更低，并将提供灵活性最佳和最强大的API。

Casdoor SDK可分为两类：

1. 前端SDK：用于网站的Javascript SDK和Vue SDK，用于应用的Android或iOS SDK。Casdoor支持为网站和移动应用程序提供身份验证。
2. 后端SDK：Go, Java, Node.js, Python, PHP等后端语言的SDK。



如果您的网站是采用后端分离的方式开发，您可以使用Javascript SDK：[casdoor-js-sdk](#) 或 Vue SDK：[casdoor-vue-sdk](#) 将Casdoor整合到前端。如果您的网页应用程序是由JSP或PHP开发的传统网站，那您就只能使用后端SDK。示例：[casdoor-Python-vue-sdk示例](#)

前端 SDK	描述	SDK 代码库	示例
Javascript SDK	适用于网页	<a href="#">casdoor-js-sdk</a>	<a href="#">Casnode</a> , <a href="#">Casbin-OA</a> , <a href="#">Confita</a>
Android SDK	适用于Android应用程序	<a href="#">casdoor-android-sdk</a>	<a href="#">casdoor-android-example</a>
iOS SDK	适用于iOS应用程序	<a href="#">casdoor-ios-sdk</a>	<a href="#">casdoor-ios-example</a>
Electron SDK	适用于Electron应用程序	<a href="#">casdoor-js-sdk</a>	<a href="#">casdoor-electron-example</a>
.NET Desktop SDK	适用于.NET桌面应用		WPF: <a href="#">casdoor-dotnet-desktop-example</a> , WinForms: <a href="#">casdoor-dotnet-winform-example</a>
React SDK	适用于React网站	<a href="#">casdoor-react-sdk</a>	
Vue SDK	适用于Vue网站	<a href="#">casdoor-vue-sdk</a>	<a href="#">casdoor-python-vue-sdk-example</a>
Angular SDK	适用于angular 1.0, 2.0网站	<a href="#">casdoor-angular-sdk</a>	
Flutter SDK	适用于Flutter应用程序	<a href="#">casdoor-flutter-sdk</a>	<a href="#">casdoor-flutter-example</a>
uni-app SDK	适用于uni-app应用程序	<a href="#">casdoor-uniapp-sdk</a>	<a href="#">casdoor-uniapp-example</a>

接下来，根据您后端的语言，可以选择使用下面的后端SDK之一：

后端 SDK	描述	SDK 码	示例
Go SDK	适用于Go后端	<a href="#">casdoor-go-sdk</a>	<a href="#">Casnode</a> , <a href="#">Casbin-OA</a> , <a href="#">Confita</a>

后端 SDK	描述	SDK 码	示例
Java SDK	适用于Java后端	casdoor-java-sdk	casdoor-spring-boot-starter, casdoor-spring-boot-example
Node.js SDK	适用于Node.js后端	casdoor-nodejs-sdk	
Python SDK	适用于Python后端	casdoor-python-sdk	casdoor-python-vue-sdk-example
PHP SDK	适用于PHP后端	casdoor-php-sdk	wordpress-casdoor-plugin
.NET SDK	适用于ASP.NET后端	casdoor-dotnet-sdk	casdoor-dotnet-sdk-example
Rust SDK	适用于Rust 后端	casdoor-rust-sdk	casdoor-rust-example
Dart SDK	适用于Dart后端	casdoor-dart-sdk	
Cpp SDK	适用于Cpp后端	casdoor-cpp-sdk	casdoor-cpp-qt-example

官方的 Casdoor SDK 完整列表请查看: <https://github.com/casdoor?q=sdk&type=all&language=&sort=>

## 如何使用 Casdoor SDK ?

### 1. 后端 SDK 配置

当您的应用程序启动时，您需要调用 `InitConfig()` 函数来初始化Casdoor SDK 配置。例举Casdoor-go-sdk 为示例: <https://github.com/casbin/casnode/blob/6d4c55f5c9a3c4bd8c85f2493abad3553b9c7ac0/controllers/account.go#L51-L64>

```
var CasdoorEndpoint = "https://door.casdoor.com"
var ClientId = "541738959670d221d59d"
var ClientSecret = "66863369a64a5863827cf949bab70ed560ba24bf"
var CasdoorOrganization = "casbin"
var CasdoorApplication = "app-casnode"

//go:embed token_jwt_key.pem
var JwtPublicKey string

func init() {
    auth.InitConfig(CasdoorEndpoint, ClientId, ClientSecret, JwtPublicKey, CasdoorOrganization, CasdoorApplication)
}
```

`InitConfig()` 的所有参数解释为:

参数	是否必须	描述
endpoint	是	Casdoor 的服务URL, 例如: <code>https://door.casdoor.com</code> or <code>http://localhost:8000</code>
clientId	是	Casdoor 应用程序的客户端 ID
clientSecret	是	Casdoor 应用程序的客户端密钥
jwtPublicKey	是	Casdoor 应用程序证书的公钥
organizationName	是	Casdoor 组织的名称

参数	是否必须	描述
applicationName	否	Casdoor 应用程序的名称

### 💡 提示

`jwtPublicKey` 可以在 `Certs` 页面中进行管理。

Certs <a href="#">Add</a>								
Name	Created time	Display name	Scope	Type	Crypto algorithm	Bit size	Expire in years	Action
cert_rjeegc	2022-02-16 11:04:10	New Cert - rjeegc		JWT	x509	RSA	4096	20
cert-built-in	2022-02-15 12:31:46	Built-in Cert		JWT	x509	RSA	4096	20

2 in total < 1 > 10 / page

您可以在证书编辑页面中找到公钥，复制或下载它以供 sdk 使用。

[Copy public key](#) [Download public key](#)

```
-----BEGIN CERTIFICATE-----
MIIEtTCCAuGgAwIBAgDAlcAMA0GCSqGSIb3DQEBCwUAMDAyHxHTAbBgNVAoTFEnh
c2Rvb3lgT3InWSpemf0aW9uMjRUiwEwDVQDExwDXNkb29yENlcnQwHcNMlEx
MDxEIMDpMTUyWhNCDexMDxE1MDgxMjUwYjA2MROwGwYDVQKExeDXNkb29yE9y
Z2fXpXphdGlvbJVMlBmGA1UEAxMCQFZg9vcBDZX0MlICjAnBqkjhkG9w0B
AQEEAOCAgBAMIIlCgKCAgEaInpbSE1/yml0f1RSDSSE8IR7y+lw+RJj74e5ejqgB8zMY
W8+0rkEr2zCKTR+9VB3jaeBz/zQoPFPVn79bfZate/hUrPK0Gg9P1gOwloC1A
3sarHTP4Qm/LQR0tHqZfybdySpvWaQvNnADEFm7mTrSB0/vUjNCUBDPSTLvjCo
4WlISf6NkfkoZ7kmPstj+b1vcsqRAWgd89h62Kp1js1Yn7Gauo3qf4zo
Kb1URYxkQjXivwCQsfUkU5ew5zuPSIDRlLoByQTlbx0jLAfnNW3/gz5Djd/
60d6H7mvbLs45mjdyfHxCD81Kn7N+xojinfaNkweP2REV-RNcf0x4GuhrsnLsmk
mtUDeqy29aBLs9j1YEQM212Eq+RVtU+vB8y8K/d1b1cy+IfrG5/bpw
OwlvC1A3saarHTP4Qm/LQR0tHqZfybdySpvWaQvNnADEFm7mTrSB0/vUjNCUBDPSTLvjCo
PTSLvJQ4WlISf6NkfkoZ7kmPstj+b1vcsqRAWgd89h62Kp1js1Yn7Gauo
I3qf4zoXkb1URYxkQjXivwCQsfUkU5ew5zuPSIDRlLoByQTlbx0jLAfnNW3/g
pzSDjgd/60d6H7mvbLs45mjdyfHxCD81Kn7N+xojinfaNkweP2REV-RNcf0x4GuhrsnLsmk
hRsnSmkmU/DeyJ29aBL9q11YEQIM2/JEq+RVtU+vB8y8K/d1b1cy+IfrG5/bpw
IDpS262b0q4RSvbZ7tbbDw2zvOfI/QLvOrfjblf0hfr/AeZMhPlK0xVf24
yE+hq268w#fD0V9xY/RbSAf73230sYnjifgUlrOhnRgCpjlk/MZ2K84Kb0
wnbCAwEAaaMQMA4wD4YDVR0TAQh/BAlwADA0BgkjhkG9w0BAQsFAAOCAgEAn2f
DKkLX+1vKRo/5gj+PlR8P5NKuQkmuH97b8C52gS1phDyNg/c4Lsdzu4Awe6ve
C06fWdWSls8UPUPdjmT2uMPsNjwLxG3QsismMUNRNvFLfTRem/heJeo2gu9j1M
8hawMaidsjjH2Rgnf0DeE2z8tNvRfhbR8knC01dtTku1S1N0/irh2t1W4jt4rxzCvI
2nR42Fybap3/g2XMHNNR0wZmNjggsF7XVENCsuFo1]jywLaguCg54l7XVLG
omKNNNcc8h1FcEkj/nmbGMhodnfFWKDtscbImcOPNHof6izqzMy/Hqr+nWYy7maAG
Jtevs3ggM28F9Qzr3HpUc6R3ZYYWDY/xPisukfOpZgtH979XC4ndf0WPn0BLql
2D1taBmjijGolvb7XNVKcUDIXXYw85ZT2Z5b9cl4e+6bmwyWqJlhtwt+At/uEV
XzC7084AL6xau1kEpfV9o1GERzYR25P9NUA7Ko5AVMp9wDD0Tkt+Lbxn2E
HHnWky8xhQKZ9sR7YBPGls/c6cviv5Ua15Qg/8dlRZ/veyfGo2yZs+hKVUS
nCIIhBcAyFnn1hddwEdH3jD BjN6ciotlZr/f/3VyalWsaLAdosHAgMwfXuWP+h
8XKXnlxruHbTMQY1ZPDgsp5a9+54Q9wbRRAY=
```

-----END CERTIFICATE-----

[Copy private key](#) [Download private key](#)

```
-----BEGIN PRIVATE KEY-----
MIUKQjBAKCAgEAslnpbSE1/yml0f1RSDSSE8IR7y+lw+RJj74e5ejqgB8zMY
k7HeHcYzr/hmNeWtVXnhXu1P0mBeQ5yp/CGoBvgEmjAE7NmzklNjOQCjYwUr
sOf1Mn1C0j1vx3mV1k7jSrsKmHYY1vaXEP3+vB8Hjg3MHWrb07uvfMCj5
W8+0rkEr2zCKTR+9VB3jaeBz/zQoPFPVn79bfZate/hUrPK0Gg9P1gOwloC1A
3sarHTP4Qm/LQR0tHqZfybdySpvWaQvNnADEFm7mTrSB0/vUjNCUBDPSTLvjCo
4WlISf6NkfkoZ7kmPstj+b1vcsqRAWgd89h62Kp1js1Yn7Gauo3qf4zo
Kb1URYxkQjXivwCQsfUkU5ew5zuPSIDRlLoByQTlbx0jLAfnNW3/gz5Djd/
60d6H7mvbLs45mjdyfHxCD81Kn7N+xojinfaNkweP2REV-RNcf0x4GuhrsnLsmk
mtUDeqy29aBLs9j1YEQM212Eq+RVtU+vB8y8K/d1b1cy+IfrG5/bpw
OwlvC1A3saarHTP4Qm/LQR0tHqZfybdySpvWaQvNnADEFm7mTrSB0/vUjNCUBDPSTLvjCo
PTSLvJQ4WlISf6NkfkoZ7kmPstj+b1vcsqRAWgd89h62Kp1js1Yn7Gauo
I3qf4zoXkb1URYxkQjXivwCQsfUkU5ew5zuPSIDRlLoByQTlbx0jLAfnNW3/g
pzSDjgd/60d6H7mvbLs45mjdyfHxCD81Kn7N+xojinfaNkweP2REV-RNcf0x4GuhrsnLsmk
hRsnSmkmU/DeyJ29aBL9q11YEQIM2/JEq+RVtU+vB8y8K/d1b1cy+IfrG5/bpw
IDpS262b0q4RSvbZ7tbbDw2zvOfI/QLvOrfjblf0hfr/AeZMhPlK0xVf24
yE+hq268w#fD0V9xY/RbSAf73230sYnjifgUlrOhnRgCpjlk/MZ2K84Kb0
wnbCAwEAaaMQMA4wD4YDVR0TAQh/BAlwADA0BgkjhkG9w0BAQsFAAOCAgEAn2f
DKkLX+1vKRo/5gj+PlR8P5NKuQkmuH97b8C52gS1phDyNg/c4Lsdzu4Awe6ve
C06fWdWSls8UPUPdjmT2uMPsNjwLxG3QsismMUNRNvFLfTRem/heJeo2gu9j1M
8hawMaidsjjH2Rgnf0DeE2z8tNvRfhbR8knC01dtTku1S1N0/irh2t1W4jt4rxzCvI
2nR42Fybap3/g2XMHNNR0wZmNjggsF7XVENCsuFo1]jywLaguCg54l7XVLG
omKNNNcc8h1FcEkj/nmbGMhodnfFWKDtscbImcOPNHof6izqzMy/Hqr+nWYy7maAG
Jtevs3ggM28F9Qzr3HpUc6R3ZYYWDY/xPisukfOpZgtH979XC4ndf0WPn0BLql
2D1taBmjijGolvb7XNVKcUDIXXYw85ZT2Z5b9cl4e+6bmwyWqJlhtwt+At/uEV
XzC7084AL6xau1kEpfV9o1GERzYR25P9NUA7Ko5AVMp9wDD0Tkt+Lbxn2E
HHnWky8xhQKZ9sR7YBPGls/c6cviv5Ua15Qg/8dlRZ/veyfGo2yZs+hKVUS
nCIIhBcAyFnn1hddwEdH3jD BjN6ciotlZr/f/3VyalWsaLAdosHAgMwfXuWP+h
8XKXnlxruHbTMQY1ZPDgsp5a9+54Q9wbRRAY=
```

-----END PRIVATE KEY-----

[Save](#) [Save & Exit](#)

之后，您可以在应用编辑页面选择证书。

The screenshot shows the Casdoor application configuration interface. The 'Edit Application' screen has fields for Name (app-built-in), Display name (Casdoor), Logo (URL: https://cdn.casbin.com/logo/logo\_1024x256.png), Home (https://casdoor.org), Description, Organization (built-in), Client ID (c2ab05e8460fd3ff9d), Client secret (c9199f102508f08f9d253638f1a72b4c3e926d05), and Cert (dropdown menu). A red arrow points to the 'cert-built-in' option in the dropdown menu.

## 2. 前端配置

首先，通过 NPM 或 Yarn 安装 `casdoor-js-sdk`

```
npm install casdoor-js-sdk
```

或者：

```
yarn add casdoor-js-sdk
```

然后定义以下实用功能(在全局JS文件中更好，比如 `Setting.js`)：

```
import Sdk from "casdoor-js-sdk";

export function initCasdoorSdk(config) {
  CasdoorSdk = new Sdk(config);
}

export function getSignupUrl() {
  return CasdoorSdk.getSignupUrl();
}

export function getSigninUrl() {
  return CasdoorSdk.getSigninUrl();
}

export function getUserProfileUrl(userName, account) {
  return CasdoorSdk.getUserProfileUrl(userName, account);
}

export function getMyProfileUrl(account) {
  return CasdoorSdk.getMyProfileUrl(account);
}

export function getMyResourcesUrl(account) {
  return CasdoorSdk.getMyProfileUrl(account).replace("/account?", "/resources?");
}
```

在您前端代码的入口文件(如 `index.js` 或 `app.js` 在React中), 您需要通过调用 `InitConfig()` 函数来初始化 `casdoor-js-sdk` 前4个参数应该使用与 Casdoor 后端SDK 相同的值。最后一个参数 `重定向路径` 是从Casdoor的登录页面返回的重定向URL的相对路径。

```
const config = {
  serverUrl: "https://door.casdoor.com",
  clientId: "014ae4bd048734ca2dea",
  organizationName: "casbin",
  appName: "app-casnode",
  redirectPath: "/callback",
};

xxx.initCasdoorSdk(config);
```

(可选) 因为我们正在使用React作为示例, 我们的 `/callback` 路径正在撞击React路由。我们使用以下React组件接收 `/回调` 调用并发送到后端。如果您直接重定向到后端(如JSP 或 PHP), 您可以忽略此步骤。

```
import React from "react";
import {Button, Result, Spin} from "antd";
import {withRouter} from "react-router-dom";
import * as Setting from "./Setting";

class AuthCallback extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      classes: props,
      msg: null,
    };
  }

  componentWillMount() {
    this.login();
  }

  login() {
    Setting.signin().then((res) => {
      if (res.status === "ok") {
        Setting.showMessage("success", `Logged in successfully`);
        Setting.goToLink("/");
      } else {
        this.setState({
          msg: res.msg,
        });
      }
    });
  }

  render() {
    return (
      <div style={{textAlign: "center"}}>
        {this.state.msg === null ? (
          <Spin
            size="large"
            tip="Signing in..."
            style={{paddingTop: "10%"}}
          />
        ) : (
          <div style={{display: "inline"}}>
            <Result
              status="error"
              title="Login Error"
              subTitle={this.state.msg}
              extra={[
                <Button type="primary" key="details">
                  Details
                </Button>
              ]}
            />
        )}
      </div>
    );
  }
}
```

### 3. 获取登录 URL

接下来，您可以显示“注册”和“登录”按钮或链接到您的用户。可以在前端或后端检索URL。详细信息见：[/docs/basic/core-concepts#login-urls](#)

### 4. 获取并验证token

步骤如下：

1. 用户点击登录URL并重定向到Casdoor的登录页面，如 `https://door.casbin.com/login/oauth/authorize?client_id=014ae4bd048734ca2dea&response_type=code&redirect_uri=https%3A%2F%2Fforum.casbin.com%2Fcallback&scope=read&state=app-casnode`
2. 用户输入用户名和密码，并点击登录（或者选择第三方登录，例如通过GitHub进行登录）
3. 该用户被重定向到您的应用，使用Casto发行的授权码(例如：`https://forum.casbin.com?code=xxx&state=yyy`)，您的应用程序的后端需要将授权码与访问令牌交换，并验证访问令牌是否有效和由Casdoor签发。函数`GetOAuthToken()` 和`ParseJwtToken()`由Casdoor后端SDK提供。

以下代码显示如何获取并验证访问令牌。Casnode (一个Go编写的论坛网站)，见：<https://github.com/casbin/casnode/blob/6d4c55f5c9a3c4bd8c85f2493abad3553b9c7ac0/controllers/account.go#L51-L64>

```
// 从重定向 URL 的 GET 参数中获取代码和状态
code := c.Input().Get("code")
state := c.Input().Get("state")

// 用代码和状态交换token
token, err := auth.GetOAuthToken(code, state)
if err != nil {
    panic(err)
}

// 验证访问令牌
claims, err := auth.ParseJwtToken(token.AccessToken)
if err != nil {
    panic(err)
}
```

如果`ParseJwtToken()`结束时没有错误，那么用户已成功登录到应用程序。返回的`claims`可以稍后用来识别用户。

### 4. 用token识别用户



信息

这一部分实际上是您的应用程序本身的业务逻辑，而不是OIDC、OAuth 或Casdoor的一部分。我们只是提供正确做法，因为许多人不知道该怎么做。

在Casdoor中，访问令牌通常与ID令牌相同。他们是一样的。因此，访问令牌包含登录用户的所有信息。

由`ParseJwtToken()`返回的变量`claims`被定义为：

```
Type Claims struct
    User
    AccessToken string `json:"accessToken"
    jwt.RegisteredClaims
}
```

1. `User`: User 对象，包含登录用户的所有信息，请参见定义：[/docs/basic/core-concepts#user](#)
2. `AccessToken`: token信息
3. `jwt.RegisteredClaim`: JWT需要一些其他值。

这时，应用程序通常有两种方法记住用户会话： `session` and `JWT`。

## Session

设置Session的方法因语言和框架而大不相同。例如，Casnode 使用 [Beego web 框架](#) 并通过调用设置会话：`c.SetSessionUser()`。

```
token, err := auth.GetOAuthToken(code, state)
if err != nil {
    panic(err)
}

claims, err := auth.ParseJwtToken(token.AccessToken)
if err != nil {
    panic(err)
}

claims.AccessToken = token.AccessToken
c.SessionUser(claims) // 设置会话
```

## JWT

从 Casdoor 返回的 `accessToken` 实际上是一个 JWT。因此，如果您的应用程序使用 JWT 来保持用户 `session`，只需直接为它使用访问令牌：

1. 将访问令牌发送到前端，在本地存储浏览器等地方保存。
2. 让浏览器为每一个请求发送访问令牌到后端。
3. 调用 `ParseJwtToken()` 或使用您自己的函数来验证 `token`，通过后端提供的已登录用户信息。

## 5. (可选) 与用户表的互动



信息

这一部分是由 [Castor Public API](#) 提供的，而不是OIDC 或 OAuth 的一部分。

Casdoor Backend SDK 提供了许多辅助功能，不仅限于：

- `GetUser(name string)`: 通过用户名获取用户。
- `GetUsers()`: 获取所有用户。
- `AddUser()`: 添加一个用户。
- `UpdateUser()`: 更新一个用户。
- `DeleteUser()`: 删除一个用户。
- `CheckUserPassword(auth.User)`: 检查用户的密码。

这些函数是通过对 [Castor Public API](#) 调用 RESTful API 实现的。如果 Casdoor Backend SDK 中没有提供功能，您可以自己调用 RESTful API。

# 如何启用单点登录

## 简介

您已连接了 Casdoor，并在组织中配置了多个应用程序。您希望用户登录一次到组织中的任何应用程序，然后在他们转到另一个应用程序时能够登录，而无需额外单击。

我们提供这个单点登录，您只需要：

- 启用自动登录按钮。
- 填写首页的 URL。
- 在应用程序主页中添加 `Silent Signin` 函数。

### 备注

Casdoor 提供的基本登录过程允许用户通过选择当前登录的用户或使用另一个账户登录到组织中的其他应用程序。

启用自动登录后，选择框将不显示，登录用户将直接登录。

## 配置

1. 填充字段 **首页**。它可以是应用程序的主页或登录页面。

Casdoor Home Organizations Users Roles Permissions Models Adapters Providers Applications

Edit Application Save Save & Exit

Name : app-casbin-oa

Display name : Casbin OA

Logo : URL : https://cdn.casbin.org/img/casbin\_logo\_1024x256.png

Preview: 

Home : https://oa.casbin.com

Description : OA system for Casbin

2. 启用自动登录按钮。

Password ON :

Enable signup :

Signin session :

Auto signin :

Enable code signin :

Enable WebAuthn signin :

# 添加静音签名

事实上，我们通过在URL上传参数来实现自动登录。所以您的应用程序需要有一种方法来在跳转到URL后触发 登录。 We provide [casdoor-react-sdk](#) for you to quickly implement the feature. 详细信息请参阅[use-in-react](#).

## ① 信息

工作原理

1. 在应用程序主页的 URL 中，我们将携带 `silentSignin` 参数。
2. 在您的主页中确定您是否需要通过参数 静音签名 来静音登录。如果 `silentSignin === 1`，函数返回 `SilentSignin` 组件，这将帮助您启动登录请求。既然您已启用自动登录，用户将自动登录无需点击。

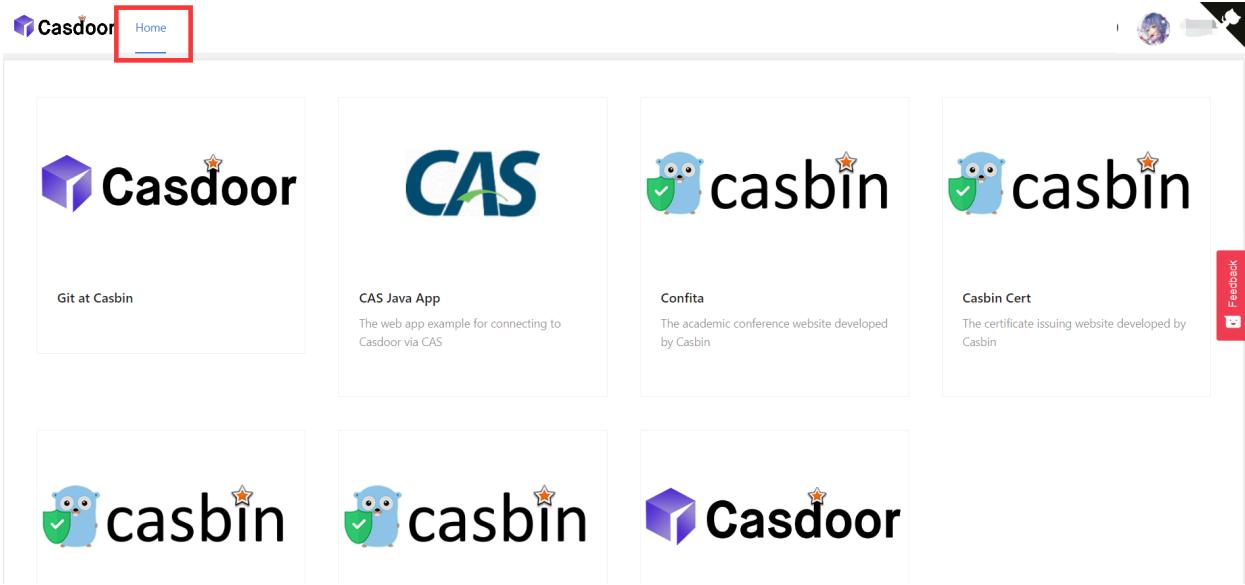
# 使用 SSO

配置已完成，下面将展示如何使用自动登录。

## ① 信息

请确保您的应用程序可以重定向到用户的个人资料页面。 API [getMyProfileUrl\(帐户, 返回 Url\)](#) 是在我们的SDK中为每种语言提供的。

打开个人资料页面并转到主页。您将看到组织中的其他应用。



点击应用程序面板，将跳转到配置中设置的网址并自动登录到应用程序。

# Vue SDK

Casdoor Vue SDK 是专为Vue2 和 Vue3 设计的，使用非常方便。

Vue SDK 基于 `casdoor-js-sdk`, 您也可以直接使用 `casdoor-js-sdk` , 这将更易定制使用。

这个插件还在开发中！ 如果您有任何问题和建议，请联系我们 [issue](#)

我们将向您展示以下步骤。

如果在阅读README后您仍然不知道如何使用它, 您可以访问这个链接: [casdoor-python-vue-sdk-example](#)来获取更多信息。

示例的前端是用`casdoor-vue-sdk`构建的, 后端是用`casdoor-Python-sdk`构建的, 您可以看到源代码的例子。

## 安装

```
# NPM  
npm i casdoor-vue-sdk  
  
# Yarn  
yarn add casdoor-vue-sdk
```

## Init SDK

初始化需要 5 个参数，它们都是字符串类型：

名称(按顺序排列)	必选项	描述信息
服务器Url	是	您的Casdoor服务器URL
客户端 Id:	是	您Casdoor应用程序的客户端 ID
应用程序名称	是	您的Casdoor应用程序的名称
组织名称	是	与您的Casdoor应用程序相关联的Casdoor组织名称
重定向路径	否	您的 Casdoor 应用程序的重定向URL 路径将是 /callback

install:

对于Vue3:

```
// in main.js
import Casdoor from 'casdoor-vue-sdk'
const config = {
  serverUrl: "http://localhost:8000",
  clientId: "4262bea2b293539fe45e",
  organizationName: "casbin",
  appName: "app-casnnode",
  redirectPath: "/callback",
};
const app = createApp(App)
app.use(Casdoor, config)
```

对于Vue2:

```
// in main.js
import Casdoor from 'casdoor-vue-sdk'
import VueCompositionAPI from '@vue/composition-api'
const config = {
  serverUrl: "http://localhost:8000",
  clientId: "4262bea2b293539fe45e",
  organizationName: "casbin",
  appName: "app-casnnode",
  redirectPath: "/callback",
};
Vue.use(VueCompositionAPI)
Vue.use(Casdoor, config)
new Vue({
  render: h => h(App),
}).$mount('#app')
```

## 示例：

```
// in app.vue
<script>
export default {
  name: 'App',
  methods: {
    login() {
      window.location.href = this.getSigninUrl();
    },
    signup() {
      window.location.href = this.getSignupUrl();
    }
  }
}
</script>
```

自动修复

如果 没有触发 `postinstall` hook, 或者您更新了 Vue 版本, 尝试运行以下命令解决重定向问题。

```
npx vue-demi-fix
```

关于Vue 版本的更多信息: 请参见 [vue-demi 文档](#)

# 桌面 SDK

## Electron 应用

一个 Electron 桌面应用程序用于Casdoor的示例

## Dotnet桌面应用程序

一个 Dotnet 桌面应用程序用于Casdoor的示例

## Qt 桌面应用程序

一个 Qt 桌面应用程序用于Casdoor的示例

# Electron 应用

一个为 Casdoor 的 Electron 桌面应用程序示例。

## 如何运行示例

### 初始化

您需要初始化6个参数，它们都是字符串类型：

名称	描述	路径
serverUrl	您的Casdoor服务器URL	src/App.js
clientId	您Casdoor应用程序的客户端 ID	src/App.js
appName	您的Casdoor应用程序的名称	src/App.js
redirectPath	您的 Casdoor 应用程序的重定向URL 路径将是 /callback 如果没有提供	src/App.js
clientSecret	您的Casdoor应用程序的客户端密钥	src/App.js
casdoorServiceDomain	您的Casdoor服务器URL	public/electron.js

如果您没有设置这些参数，此项目将使用 [Casdoor 在线演示](#) 作为默认的Casdoor 服务器，并使用 [Casnode](#) 作为默认的 Casdoor 应用程序。

### 可用命令

在项目目录中，您可以运行：

`npm run dev` 或者 `yarn dev`

构建electron应用并运行此应用。

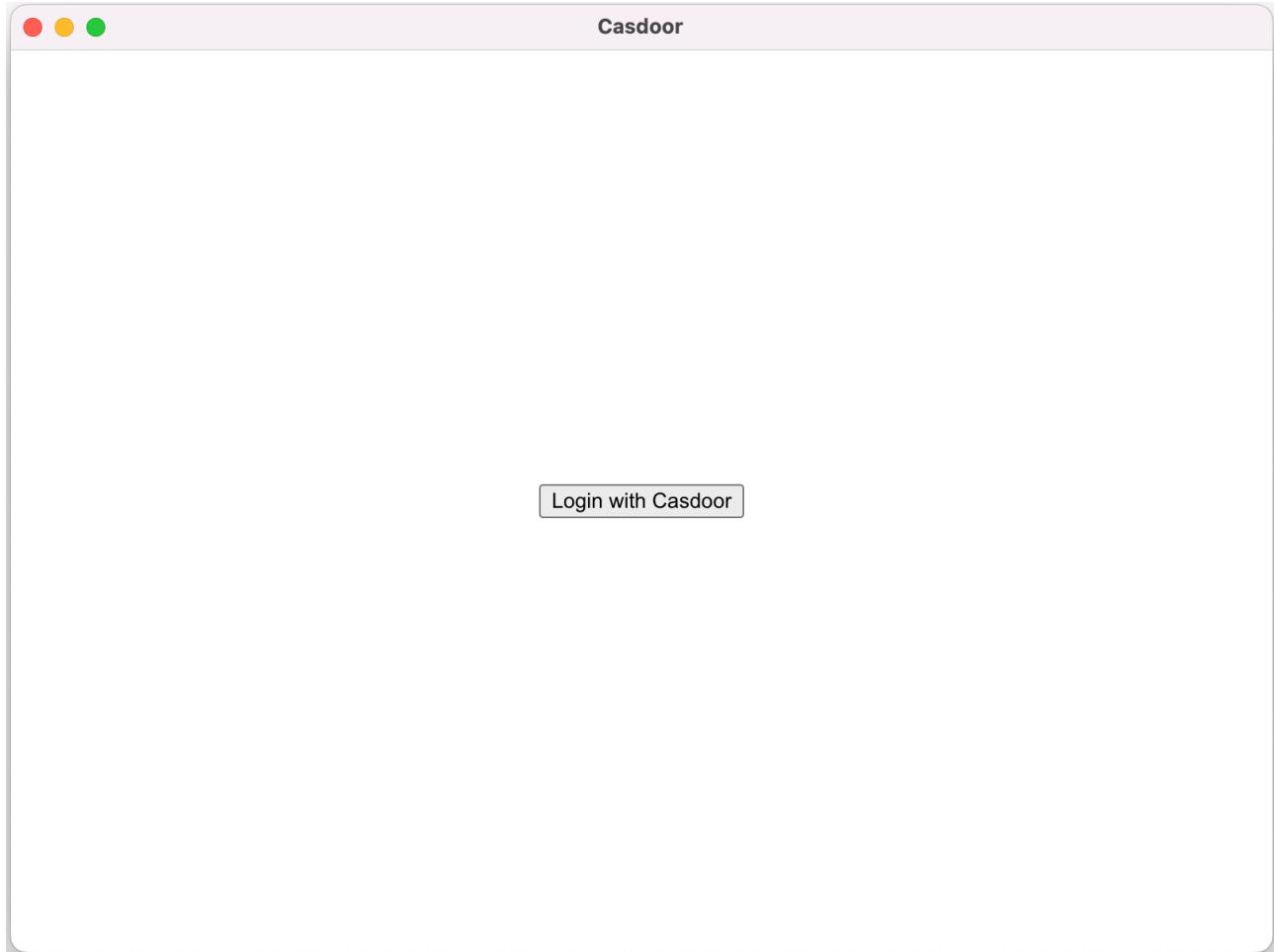
`npm run make` 或者 `yarn make`

打包和分发您的应用程序。它将创建 `out` 文件夹，您的软件包将被定位：

```
// macOS下的out/示例
├── out/make/zip/darwin/x64/casdoor-electron-example-darwin-x64-1.0.0.zip
├── ...
└── out/casdoor-electron-example-darwin-x64/casdoor-electron-example.app/Contents/MacOS/casdoor-electron-example
```

### 预览

在您运行此electron应用程序后，将在您的桌面上显示一个新的窗口。



如果您点击 [使用 Casdoor 登陆] 按钮，您的默认浏览器将被自动打开并显示登录页面。



admin

...

Auto sign in      [Forgot password?](#)

[Sign In](#)

No account? [sign up now](#)

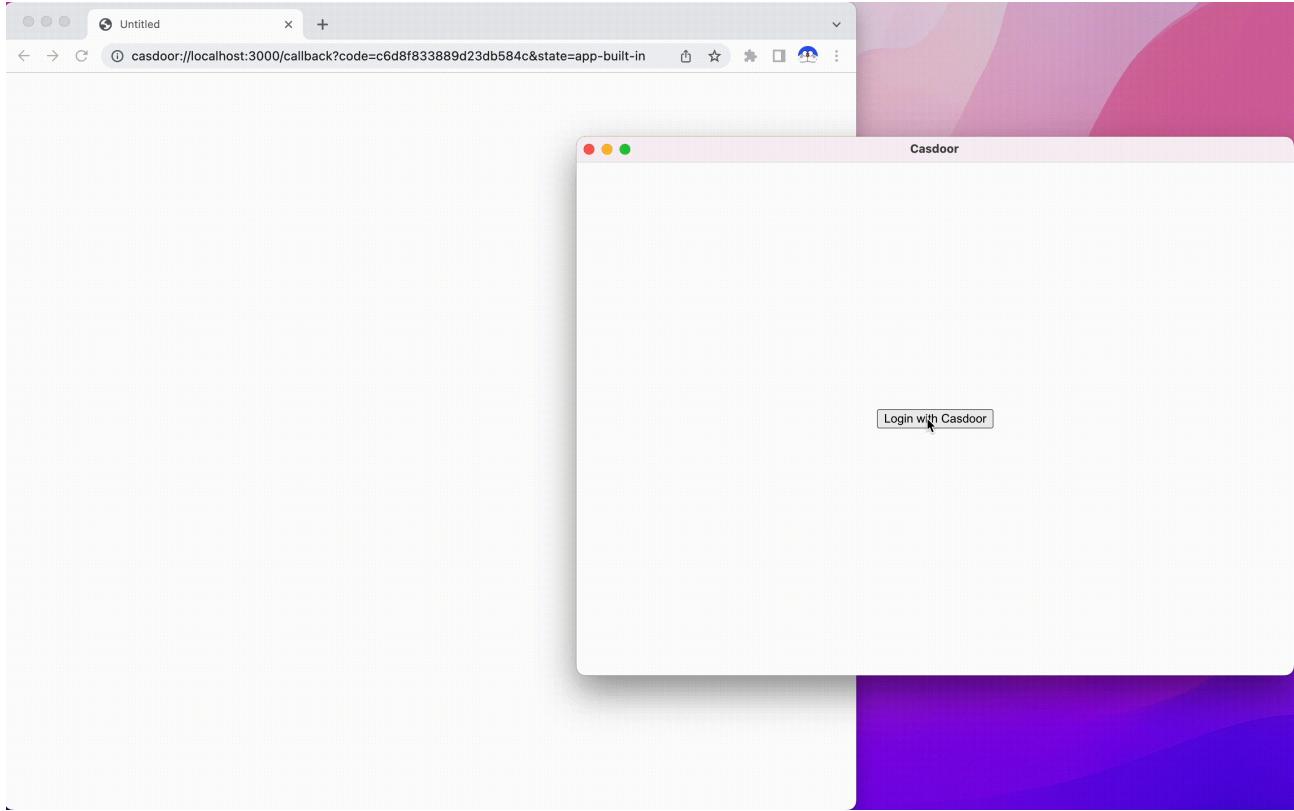


Made with ❤ by Casdoor

登录成功后，将会打开您的electron应用程序，您的用户名将会显示在您的应用程序中。



您可以通过下面的Gif图像预览整个过程。



## 如何整合?

### 设置自定义协议

首先，您需要设置自定义协议名为 `casdoor`。

```
const protocol = "casdoor";

if (process.defaultApp) {
  if (process.argv.length >= 2) {
    app.setAsDefaultProtocolClient(protocol, process.execPath, [
      path.resolve(process.argv[1]),
    ]);
  }
} else {
  app.setAsDefaultProtocolClient(protocol);
}
```

这将帮助浏览器打开您的electron应用程序并将登录信息发送到electron应用程序中。

### 通过浏览器打开登录网址

```
const serverUrl = "https://door.casdoor.com";
const appName = "app-casnode";
const redirectPath = "/callback";
const clientId = "014ae4bd048734ca2dea";
const clientSecret = "f26a4115725867b7bb7b668c81e1f8f7fae1544d";

const redirectUrl = "casdoor://localhost:3000" + redirectPath;
```

您可以更改前5个参数。

## 监听开启的应用程序事件

在您在浏览器成功登录后，浏览器将打开您的electron应用程序。您需要监听打开的应用程序事件。

```
const gotTheLock = app.requestSingleInstanceLock();
const ProtocolRegExp = new RegExp(`^${protocol}://`);

if (!gotTheLock) {
    app.quit();
} else {
    app.on("second-instance", (event, commandLine, workingDirectory) => {
        if (mainWindow) {
            if (mainWindow.isMinimized()) mainWindow.restore();
            mainWindow.focus();
            commandLine.forEach((str) => {
                if (ProtocolRegExp.test(str)) {
                    const params = url.parse(str, true).query;
                    if (params && params.code) {
                        store.set("casdoor_code", params.code);
                        mainWindow.webContents.send("receiveCode", params.code);
                    }
                }
            });
        }
    });
    app.whenReady().then(createWindow);

    app.on("open-url", (event, openUrl) => {
        const isProtocol = ProtocolRegExp.test(openUrl);
        if (isProtocol) {
            const params = url.parse(openUrl, true).query;
            if (params && params.code) {
                store.set("casdoor_code", params.code);
                mainWindow.webContents.send("receiveCode", params.code);
            }
        }
    });
}
}
```

您可以从broswer获取代码，即`casdoor_code`或`params.code`。

## 解析代码并获取用户信息

```
async function getUserInfo(clientId, clientSecret, code) {
    const { data } = await axios({
        method: "post",
        url: authCodeUrl,
        headers: {
            "content-type": "application/json",
        },
        data: JSON.stringify({
            grant_type: "authorization_code",
            client_id: clientId,
            client_secret: clientSecret,
            code: code,
        }),
    });
    const resp = await axios({
        method: "get",
        url: `${userInfoUrl}?accessToken=${data.access_token}`,
    });
    return resp.data;
}
```

最后，您可以解析代码，并按照[OAuth文档](#)页面获取用户信息。

# Dotnet桌面应用程序

一个为 Casdoor 的 [Dotnet 桌面应用程序示例](#)。

## 如何运行示例

### 前置要求

[dotnet6 sdk](#)

[webview2 runtime](#) (已经在您的窗口中预装了)

### 初始化

初始化需要 5 个参数，它们都是字符串类型：

名称	描述	文件
Domain	您的 Casdoor 服务器主机/域	<a href="#">CasdoorVariables.cs</a>
Clientid	您的 Casdoor 应用程序的客户端 ID	<a href="#">CasdoorVariables.cs</a>
AppName	您的 Casdoor 应用程序的名称	<a href="#">CasdoorVariables.cs</a>
CallbackURL	您的 Casdoor 应用程序的回调 URL 路径 将是 <code>casdoor://callback</code> 如果没有	<a href="#">CasdoorVariables.cs</a>

名称	描述	文件
	提供	
ClientSecret	您的Casdoor应用程序的客户端密钥	CasdoorVariables.cs

如果您没有设置这些参数，此项目将使用 [Casdoor 在线演示](#) 作为默认的Casdoor 服务器，并使用 [Casnode](#) 作为默认的 Casdoor 应用程序。

# 运行

## Visual Studio

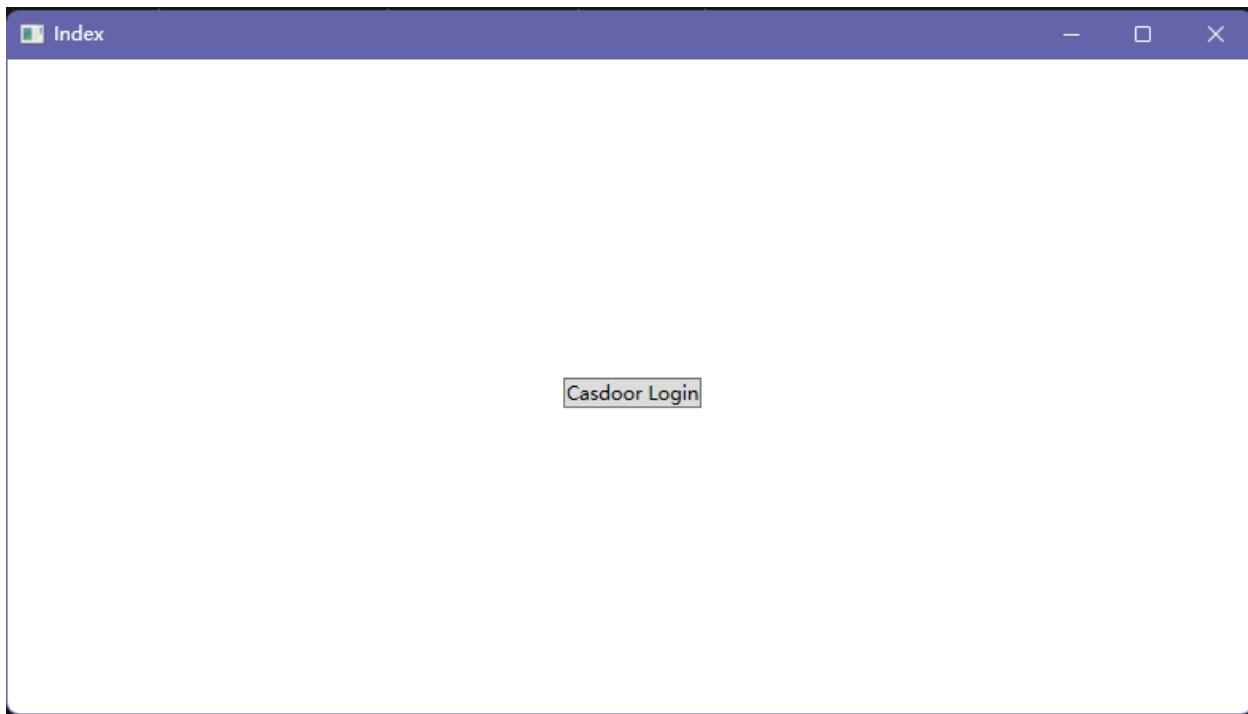
1. 打开casdoor-dotnet-desktop-example.sln
2. 按 Ctrl + F5 开始

## 命令行

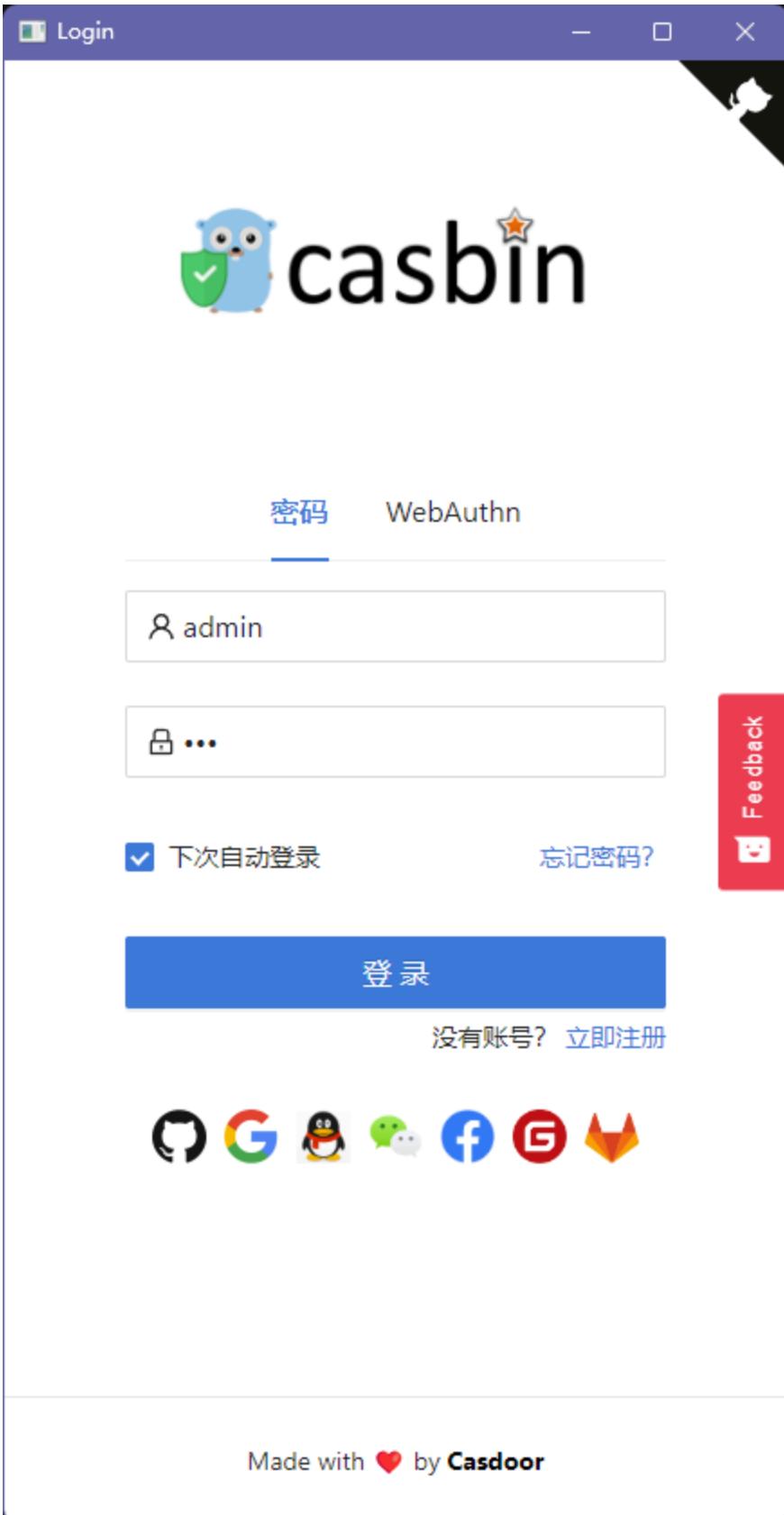
1. cd src/DesktopApp
2. dotnet run

# 效果预览

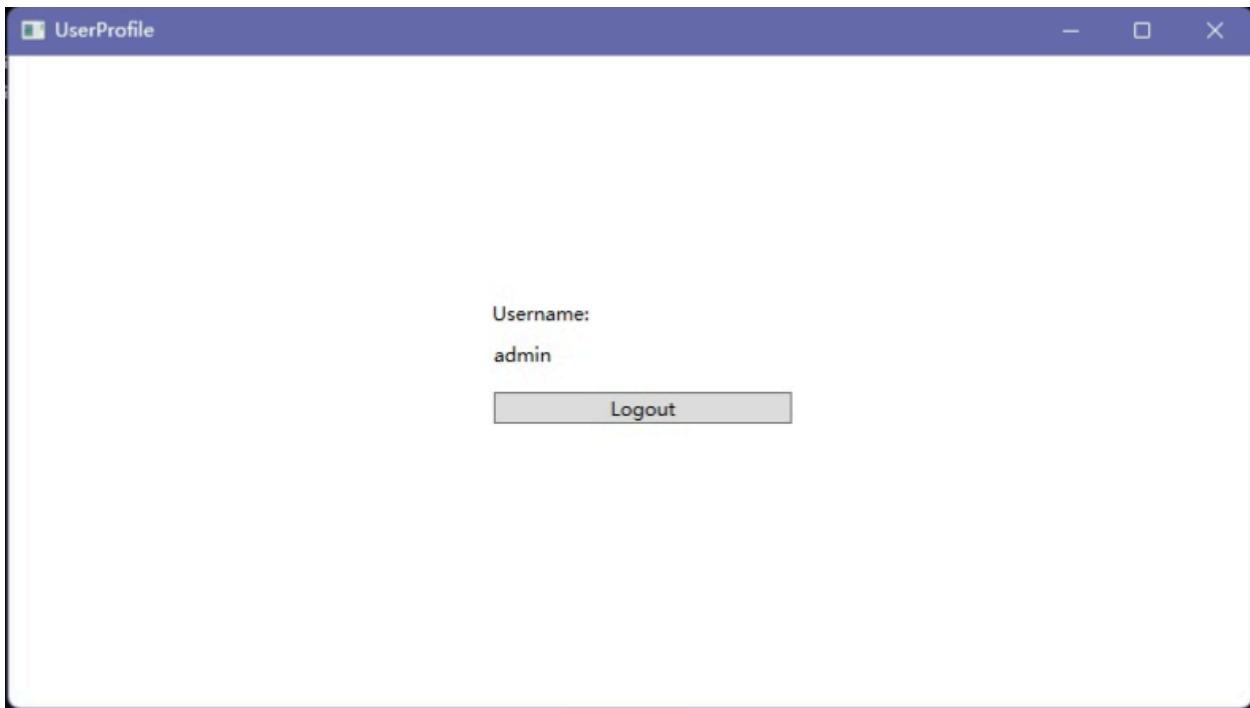
在运行此dotnet桌面应用程序后，您的桌面将显示一个新窗口。



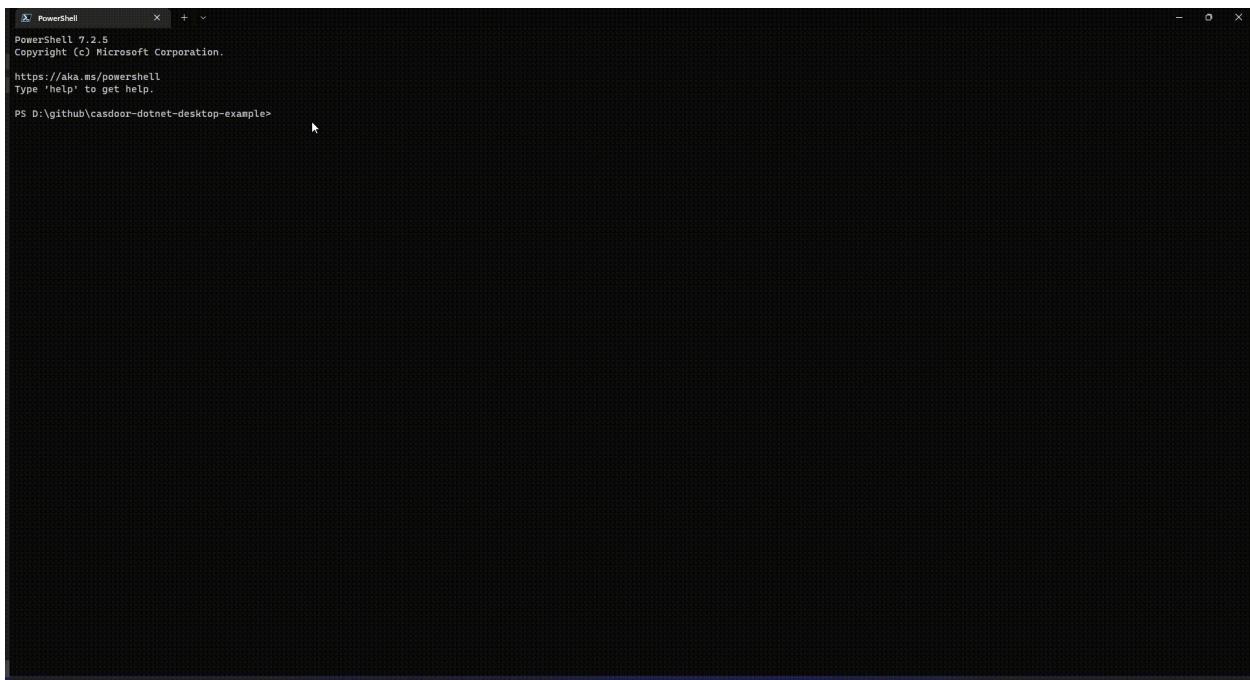
如果您点击 `Casdoor Login` 按钮，您的桌面将显示登录窗口。



成功登录后，桌面上将显示一个用户配置文件窗口。它显示您的用户名。



您可以通过下面的 gif 图像预览整个过程。



# 如何整合?

## 打开登录窗口

```
var login = new Login();
// 登录成功时触发，您将在事件处理程序中收到身份验证代码
login.CodeReceived += Login_CodeReceived;
login.ShowDialog();
```

## 使用认证码获取用户信息

```
public async Task<string?> RequestToken(string clientId, string
clientSecret, string code)
{
    var body = new
    {
        grant_type = "authorization_code",
        client_id = clientId,
        client_secret = clientSecret,
        code
    };

    var req = new RestRequest(_requestTokenUrl).AddJsonBody(body);
    var token = await _client.PostAsync<TokenDto>(req);

    return token?.AccessToken;
}

public async Task<UserDto?> GetUserInfo(string token)
{
    var req = new
    RestRequest(_getUserInfoUrl).AddQueryParameter("accessToken",
    token);
```



# Qt 桌面应用程序

一个为 Casdoor 的 [Qt 桌面应用程序示例](#)。

## 如何运行示例

### 前置要求

[Qt6 sdk](#)

[OpenSSL 工具包](#)

### 初始化

您需要初始化7个参数，它们都是字符串类型：

名称	描述	文件
endpoint	您的 Casdoor 服务器主机/域	<a href="#">mainwindow.h</a>
client_id	您的 Casdoor 应用程序的客户端 ID	<a href="#">mainwindow.h</a>
client_secret	您的 Casdoor 应用程序的客户端密钥	<a href="#">mainwindow.h</a>
certificate	Casdoor 应用程序证书的公钥	<a href="#">mainwindow.h</a>
org_name	您的 Casdoor 应用程序的名称	<a href="#">mainwindow.h</a>

名称	描述	文件
app_name	您的Casdoor应用程序的名称	mainwindow.h
redirect_url	您的Casdoor 应用程序的回调URL路径将是 casdoor://callback 如果没有提供	mainwindow.h

如果您没有设置参数 端点，此项目将使用 <http://localhost:8000> 作为默认Casdoor服务器。

## 运行

### Qt Creator

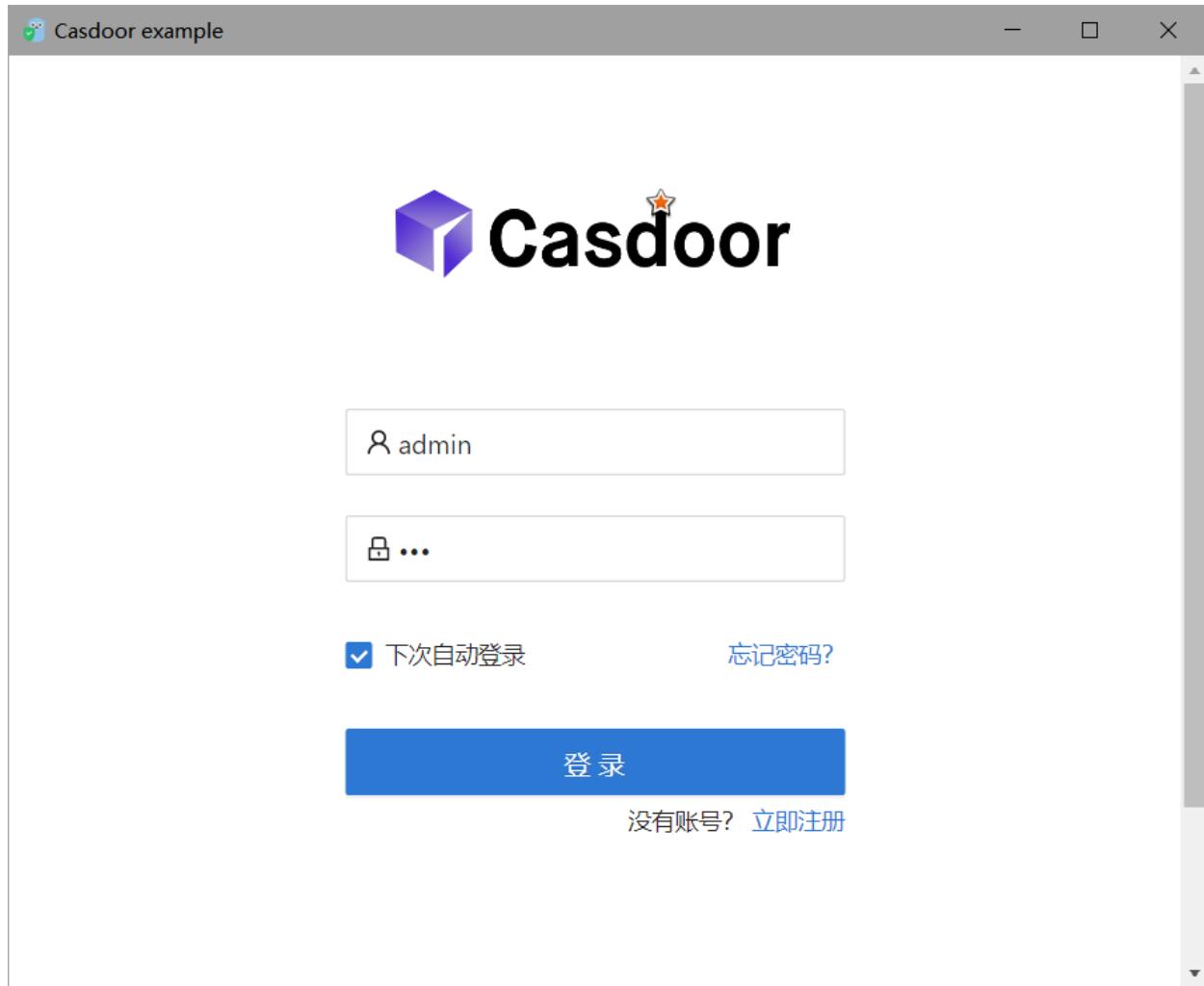
1. 打开casdoor-cpp-qt-example.pro
2. 在casdoor-cpp-qt-example.pro中设置 OpenSSL 的 INCLUDEPATH
3. 按 Ctrl + R 开始

## 效果预览

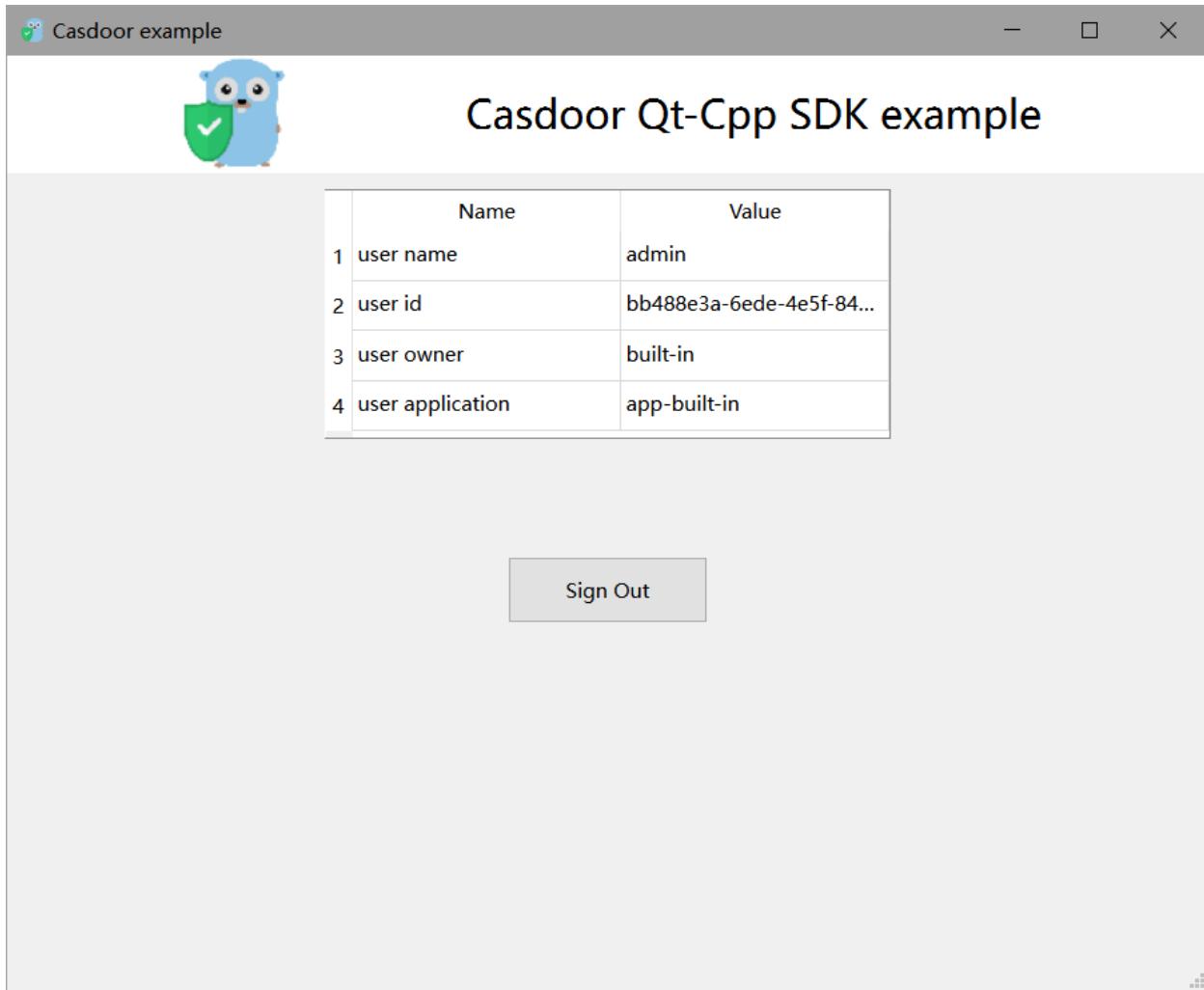
在运行此Qt桌面应用程序后，您的桌面将显示一个新窗口。



如果您点击 `Sign In` 按钮，您的桌面将显示登录窗口。



成功登录后，桌面上将显示一个用户配置文件窗口，它会显示您的信息。



您可以通过下面的 gif 图像预览整个过程。



## 如何整合?

### 打开登录窗口

```
// Load and display the login page of Casdoor
m_webview->page()->load(*m_signin_url);
m_webview->show();
```

## 监听开启的应用程序事件

```
// Initialize the TcpServer object and listen on port 8080
m_tcpserver = new QTcpServer(this);
if(!m_tcpserver->listen(QHostAddress::LocalHost, 8080)) {
    qDebug() << m_tcpserver->errorString();
    close();
}
connect(m_tcpserver, SIGNAL(newConnection()), this,
SLOT(on_tcp_connected()));
```

## 使用认证码获取用户信息

```
// Get token and parse it with the JWT library
std::string token = m_casdoor->GetOAuthToken(code.toStdString());
auto decoded = m_casdoor->ParseJwtToken(token);
```



如何连接到Casdoor

Casdoor插件

# Casdoor插件

Casdoor还为一些热门平台提供插件或中间件，例如Java的SpringBoot、PHP的WordPress、Python的Odoo等。

Casdoor 插件	语言	源代码
Spring Boot插件	Java	<a href="https://github.com/casdoor/casdoor-spring-boot-starter">https://github.com/casdoor/casdoor-spring-boot-starter</a>
Spring Boot示例	Java	<a href="https://github.com/casdoor/casdoor-spring-boot-example">https://github.com/casdoor/casdoor-spring-boot-example</a>
WordPress 插件	PHP	<a href="https://github.com/casdoor/wordpress-casdoor-plugin">https://github.com/casdoor/wordpress-casdoor-plugin</a>
Odoo 插件	Python	<a href="https://github.com/casdoor/odoo-casdoor-oauth">https://github.com/casdoor/odoo-casdoor-oauth</a>
Django 插件	Python	<a href="https://github.com/casdoor/django-casdoor-auth">https://github.com/casdoor/django-casdoor-auth</a>

完整的Casdoor官方插件列表请查看：[Casdoor repositories](#)。

# OAuth 2.0

## 介绍

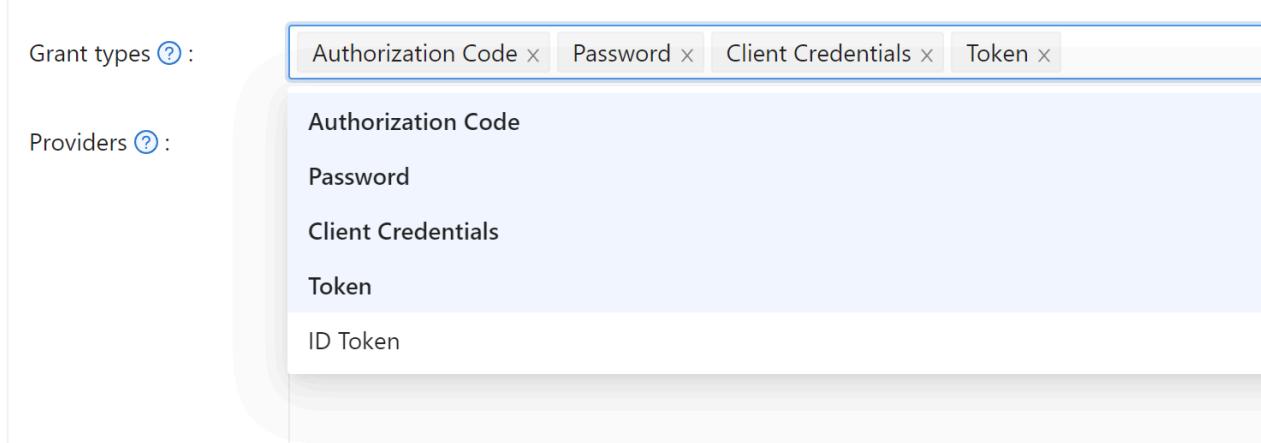
Casdoor 支持AccessToken验证客户端。在本节中，我们将向您展示如何获取AccessToken，如何验证AccessToken，以及如何使用AccessToken。

## 如何获取AccessToken

您有两种方法获得AccessToken：您可以使用 [Casdoor SDK](#)，欲了解详情，请参阅SDK文档，在此我们将主要向您展示如何使用 API 获取AccessToken。

Casdoor支持四种OAuth 授予类型: [Authorization Code Grant](#), [Implicit Grant](#), [Resource Owner Password Credentials Grant](#), 和 [Client Credentials Grant](#).

出于安全考虑，Casto应用默认已开启授权码模式。如果您需要使用其他模式，请前往相应的应用程序来设置它。



## 获取授权码

首先重定向您的用户请求到：

```
https://<CASDOOR_HOST>/login/oauth/authorize?  
client_id=CLIENT_ID&  
redirect_uri=REDIRECT_URI&  
response_type=code&  
scope=openid&  
state=STATE
```

## 可用的作用域 (scope)

名称	描述
openid (no scope)	sub (用户ID), iss (发行人) 和 aud (受众)
profile	用户资料信息，包括名称、显示名称、头像
email	用户的电子邮件地址
address	用户地址
phone	用户的电话号码

### ① 信息

您的 OAuth 应用程序可以在首次重定向时带上请求的作用域。您可以指定多个作用域并使用空格（转义后为%20）分隔：

```
https://<CASDOOR_HOST>/login/oauth/authorize?  
client_id=...&  
scope=openid%20email
```

更多详情，请参阅 [OIDC 标准](#)

当您的用户通过casdoor身份验证后，他的请求会被casdoor重定向到：

```
https://REDIRECT_URI?code=CODE&state=STATE
```

现在您已经获得授权码，在你的后端应用发送 POST 请求：

```
https://<CASDOOR_HOST>/api/login/oauth/access_token
```

在你的后端应用

```
{  
  "grant_type": "authorization_code",  
  "client_id": ClientId,  
  "client_secret": ClientSecret,  
  "code": Code,  
}
```

您将得到以下响应：

```
{  
    "access_token": "eyJhb...","  
    "id_token": "eyJhb...","  
    "refresh_token": "eyJhb...","  
    "token_type": "Bearer",  
    "expires_in": 10080,  
    "scope": "openid"  
}
```

#### ⓘ 备注

Casdoor也支持 [PKCE](#) 功能。当发送获取授权码请求时，您可以通过添加两个参数来启用 PKCE。

```
&code_challenge_method=S256&code_challenge=YOUR_CHALLENGE
```

获取令牌时，您需要通过 `code_verifier` 参数来验证 PKCE。值得一提的是，启用PKCE 后，`Client_secret`并不是必需的，但如果要发送这个参数，它的值就必须是正确的。

## 隐式授权

如果您的应用程序没有后端，您需要使用隐式授权。首先，您需要确保您启用了隐式授权，然后将您的用户请求重定向到：

```
https://<CASDOOR_HOST>/login/oauth/  
authorize?client_id=CLIENT_ID&redirect_uri=REDIRECT_URI&response_type=token&scope=openid&state=STATE
```

当您的用户通过casdoor身份验证后，他的请求会被casdoor重定向到：

```
https://REDIRECT_URI/#token=ACCESS_TOKEN
```

Casdoor还支持 `id_token` 作为参数 `response_type` 的值，这是OpenID的一个功能。

## 使用资源拥有者的密码凭据授权

如果您的应用程序没有前端来重定向用户到Casdoor，那么您可能需要这个功能。

首先，您需要确保您已启用密码凭证授权，并发送一个 POST 请求：

```
https://<CASDOOR_HOST>/api/login/oauth/access_token
```

```
{  
    "grant_type": "password",  
    "client_id": ClientId,  
    "client_secret": ClientSecret,  
    "username": Username,  
    "password": Password,
```

您将得到以下响应：

```
{  
    "access_token": "eyJhb... ",  
    "id_token": "eyJhb... ",  
    "refresh_token": "eyJhb... ",  
    "token_type": "Bearer",  
    "expires_in": 10080,  
    "scope": "openid"  
}
```

## 使用客户端凭据授权

当应用程序没有前端时，您也可以使用客户端凭据授权。

首先，您需要确保您已启用客户端凭据授权，并发送一个 POST 请求到 [https://<CASDOOR\\_HOST>/api/login/oauth/access\\_token](https://<CASDOOR_HOST>/api/login/oauth/access_token)：

```
{  
    "grant_type": "client_credentials",  
    "client_id": ClientId,  
    "client_secret": ClientSecret,  
}
```

您将得到以下响应：

```
{  
    "access_token": "eyJhb... ",  
    "id_token": "eyJhb... ",  
    "refresh_token": "eyJhb... ",  
    "token_type": "Bearer",  
    "expires_in": 10080,  
    "scope": "openid"  
}
```

必须指出，以这种方式获得的AccessToken 不同于前三个，因为它与应用程序相对应，而不是与用户相对应。

## 更新访问令牌

如果您想要更新访问令牌，您可以使用上面的 `refreshToken`。

首先您需要在应用程序中设置refreshToken的到期时间(默认为0小时)，发送一个 POST 请求到 [https://<CASDOOR\\_HOST>/api/login/oauth/refresh\\_token](https://<CASDOOR_HOST>/api/login/oauth/refresh_token)

```
{  
    "grant_type": "refresh_token",  
    "refresh_token": REFRESH_TOKEN,  
    "scope": SCOPE,
```

您将得到响应:

```
{  
    "access_token": "eyJhb... ",  
    "id_token": "eyJhb... ",  
    "refresh_token": "eyJhb... ",  
    "token_type": "Bearer",  
    "expires_in": 10080,  
    "scope": "openid"  
}
```

## 如何验证访问令牌

目前Casdoor有支持 [token 校验](#) 的API。 目前，接口使用Basic 方式认证(ClientId:ClientSecret)：

```
POST /api/login/oauth/introspect HTTP/1.1  
Host: CASDOOR_HOST  
Accept: application/json  
Content-Type: application/x-www-form-urlencoded  
Authorization: Basic Y2xpZW50X2lkOmNsawVudF9zzWNyZXQ=  
  
token=ACCESS_TOKEN&token_type_hint=access_token
```

您将得到以下响应:

```
{  
    "active": true,  
    "client_id": "c58c... ",  
    "username": "admin",  
    "token_type": "Bearer",  
    "exp": 1647138242,  
    "iat": 1646533442,  
    "nbf": 1646533442,  
    "sub": "7a6b4a8a-b731-48da-bc44-36ae27338817",  
    "aud": [  
        "c58c... "  
    ],  
    "iss": "http://localhost:8000"  
}
```

## 如何使用访问令牌

您可以使用AccessToken访问需要认证的 Casdoor API。

例如，请求 [/api/userinfo](#) 的两种不同方法。

方法 1 查询参数:

```
https://<CASDOOR_HOST>/api/userinfo?accessToken=<your_access_token>
```

方法 2 HTTP Bearer token

```
https://<CASDOOR_HOST>/api/userinfo with the header: "Authorization: Bearer <your_access_token>"
```

Casdoor 将解析 access\_token，根据 scope 作用域返回对应的用户信息。您将得到响应：

```
{  
  "sub": "7a6b4a8a-b731-48da-bc44-36ae27338817",  
  "iss": "http://localhost:8000",  
  "aud": "c58c..."  
}
```

如果您需要更多用户信息，在申请访问令牌[获取授权码](#)这一步添加更多 scope。

## userinfo 和 get-account API 之间的差异

- `/api/userinfo`: 返回用户信息是OIDC 协议的一部分。返回少量信息，只包含OIDC标准中的基本信息。请查看 Casdoor 支持的[可用的作用域（scope）](#)。
- `/api/get-account`: 获取当前登录帐户的用户对象。这是一个只适用于Casdoor 的API，用于获取Casdoor中的[用户](#)的所有信息。

# CAS

## 使用 Casdoor 作为 CAS 服务器

Cassdoor 现在可以用作CAS 服务器。 目前Casdoor支持CAS3.0 的功能。

### 简介

Casdoor中CAS终端的前缀是 `<Endpoint of casdoor>/cas/<organization name>/<application name>`, 这意味着

假设Cassdoor 的端点是 `https://door.cassdoor.com`, 其中包含一个名为 `cas-java-app` 属于一个名为 `casbin`的组织, 如果我们试图让用户通过CAS登录, 那么

- `/login` endpoint: `https://door.casdoor.com/cas/casbin/cas-java-app/login`
- `/logout` endpoint: `https://door.casdoor.com/cas/casbin/cas-java-app/logout`
- `/serviceValidate` endpoint: `https://door.casdoor.com/cas/casbin/cas-java-app/serviceValidate`
- `/proxyValidate` endpoint: `https://door.casdoor.com/cas/casbin/cas-java-app/proxyValidate`
- `/proxy` endpoint: `https://door.casdoor.com/cas/casbin/cas-java-app/proxy`
- `/validate` endpoint: `https://door.casdoor.com/cas/casbin/cas-java-app/validate`
- `/p3/serviceValidate` endpoint: `https://door.casdoor.com/cas/casbin/cas-java-app/p3/serviceValidate`
- `/p3/proxyValidate` endpoint: `https://door.casdoor.com/cas/casbin/cas-java-app/p3/proxyValidate`
- `/samlValidate` endpoint: `https://door.casdoor.com/cas/casbin/cas-java-app/samlValidate`

更多关于 CAS 及其不同版本的信息, 以及这些处理程序的参数, 请参阅 <https://aperegu.github.io/cas/6.0.x/protocol/CAS-Protocol-Specification.html>。

### 一个示例

这里是一个官方示例 <https://github.com/aperezce/cas-sample-java-webapp>, 其中包含一个使用正式的CAS java 客户端的网络应用实例 <https://github.com/aperezce/java-cas-client> 我们将通过这个例子来说明如何通过CAS 连接到Casdoor。

 备注

注意：Cassdoor 目前仅支持所有三个版本的 CAS 1.0 & 2.0 & CAS 3.0

Cas 配置位于 `src/main/webapp/WEB-INF/web.yml`。

默认情况下，此应用使用 CAS 3.0，由以下配置指定。

```
<filter-name>CAS Validation Filter</filter-name>
<filter-
class>org.jasig.cas.client.validation.Cas30ProxyReceivingTicketValidationFilter</filter-
class>
```

假定您想要通过 CAS 2.0 保护此网络应用，您应该将 CAS 验证过滤器更改为以下内容。

```
<filter-name>CAS Validation Filter</filter-name>
<filter-
class>org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter</filter-
class>
```

如果您想要使用 CAS 1.0，请使用

```
<filter-name>CAS Validation Filter</filter-name>
<filter-class>org.jasig.cas.client.validation.Cas10TicketValidationFilter</filter-class>
```

对于参数“casServerUrlPrefix”，将其更改为

```
<param-name>casServerUrlPrefix</param-name>
<param-value>http://door.casdoor.com/cas/casbin/cas-java-app</param-value>
```

对于参数“casServerLoginUrl”，将其更改为

```
<param-name>casServerLoginUrl</param-name>
<param-value>http://door.casdoor.com/cas/casbin/cas-java-app/login</param-value>
```

如果您需要自定义更多的配置，请参阅 <https://github.com/apetrue/java-cas-client> 获取详细信息。

 备注

实际上我们已经有这个演示应用程序在 <https://cas-java-app.casdoor.com> 上运行。 您可以通过CAS访问此处体验Casdoor。

# SAML

## 使用 Casdoor 作为 SAML IdP

Casdoor 现在可以用作SAML的IdP。 目前Casdoor支持SAML2.0 的主要功能。

### 简介

Casdoor 的 SAML 端点的元数据是 `<Endpoint of casdoor>/api/saml/metadata?application=<organization name>/<application name>`。 您也可以在应用程序编辑页面中找到元数据。

The screenshot shows the Casdoor application configuration interface. In the 'Providers' tab, there is a single provider entry for 'email' which is of type 'Email'. The 'SAML Metadata' field contains a large XML string representing the SAML2.0 metadata for the 'email' provider. This XML includes elements like EntityDescriptor, IDPSSODescriptor, KeyDescriptor, and X509Data.

假设Casdoor 的端点是 `https://door.casdoor.com`，其中包含一个名为 `app-built-in` 属于一个名为 `built-in` 的应用程序。

更多关于 SAML 及其不同版本的信息，请访问 [https://en.wikipedia.org/wiki/SAML\\_2.0](https://en.wikipedia.org/wiki/SAML_2.0)

### 一个示例

[gosaml2](#) 是针对服务提供商的 SAML 2.0实现，基于etree和goxmldsig，这个XML数字

签名是基于 Go 实现。 我们使用这个库在 Casdoor 中测试 SAML 2.0，如下所示。

假设Casdoor 的端点是 `https://door.casdoor.com`，其中包含一个名为 `app-built-in` 属于一个名为 `built-in` 的应用程序。 `http://localhost:6900/acs/example` 和 `http://localhost:6900/saml/acs/example`，应添加到 `app-built-in` 的重定向URL中。

```
import (
    "crypto/x509"
    "fmt"
    "net/http"

    "io/ioutil"

    "encoding/base64"
    "encoding/xml"

    saml2 "github.com/russellhaering/gosaml2"
    "github.com/russellhaering/gosaml2/types"
    dsig "github.com/russellhaering/goxmldsig"
)

func main() {
    res, err := http.Get("http://localhost:7001/api/saml/
metadata?application=admin/app-built-in")
    if err != nil {
        panic(err)
    }

    rawMetadata, err := ioutil.ReadAll(res.Body)
    if err != nil {
        panic(err)
    }

    metadata := &types.EntityDescriptor{}
    err = xml.Unmarshal(rawMetadata, metadata)
```

运行上述代码，控制台将显示以下消息。

```
Visit this URL To Authenticate:  
http://localhost:7001/login/saml/authorize/admin/app-built-in?SAMLRequest=lFVbk6K8Fv0rFvNo2QR...  
Supply:  
SP ACS URL : http://localhost:6900/v1/_saml_callback
```

点击 URL 进行身份验证，将显示 Casdoor 的登录页面。





Continue with :



Or sign in with another account :

Auto sign in      [Forgot password?](#)

[Sign In](#)

[Sign in with code](#)    No account? [sign up now](#)



验证后，您将获得如下响应消息。

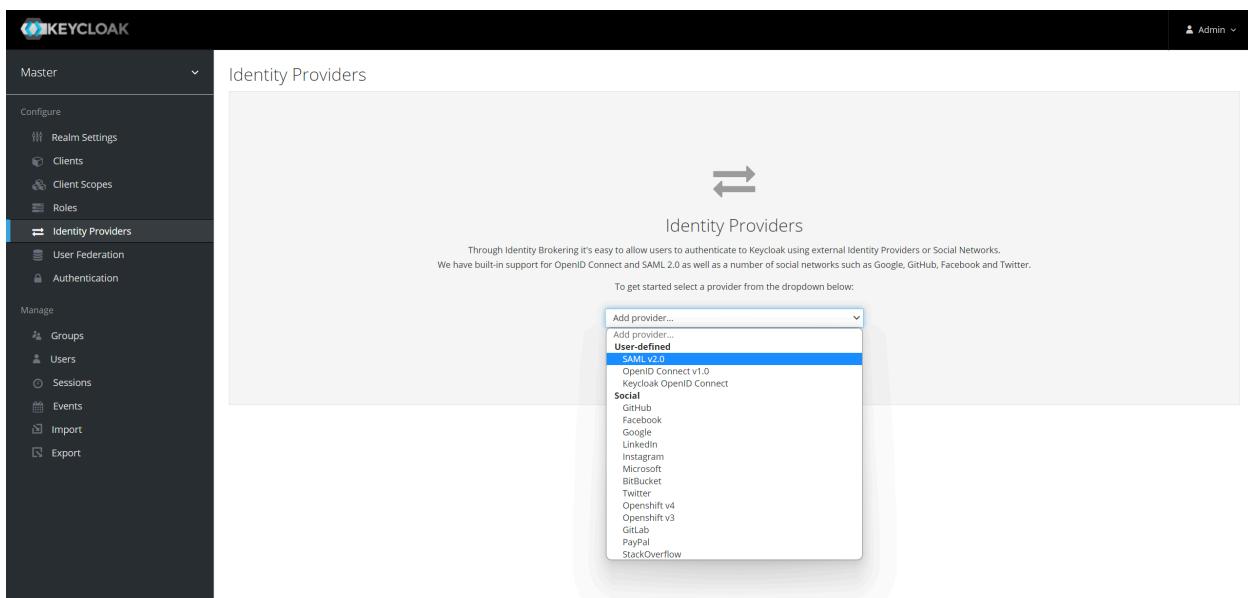
```
NameID: admin@example.com  
Assertion:  
  Email: [Space:urn:oasis:names:tc:SAML:2.0:assertion Local:Attribute] FriendlyName: Name>Email NameFormat:urn:oasis:names:tc:SAML:2.0:attribute-format:basic Values:[ [XMLName: [Space:urn:oasis:names:tc:SAML:2.0:assertion Local:AttributeValue] Type: Value:admin@example.com]]  
  Name: [XMLName: [Space:urn:oasis:names:tc:SAML:2.0:assertion Local:Attribute] FriendlyName: Name>Name NameFormat:urn:oasis:names:tc:SAML:2.0:attribute-format:basic Values:[ [XMLName: [Space:urn:oasis:names:tc:SAML:2.0:assertion Local:AttributeValue] Type: Value:admin]]]  
  DisplayName: [XMLName: [Space:urn:oasis:names:tc:SAML:2.0:assertion Local:Attribute] FriendlyName: Name>DisplayName NameFormat:urn:oasis:names:tc:SAML:2.0:attribute-format:basic Values:[ [XMLName: [Space:urn:oasis:names:tc:SAML:2.0:assertion Local:AttributeValue] Type: Value:Admin]]]  
Warnings:  
& (OneTimeUse:false ProxyRestriction:<nil> NotInAudience:false InvalidTime:false)
```

# Casdoor 作为一个 SAML IdP in Keycloak

本指南将向您展示如何配置 Casdoor 和 Keycloak 在Keycloak中将Casdoor 添加为 SAML IdP。

## 在 Keycloak 中添加 SAML IdP

打开Keycloak 管理页面, 点击 身份提供商 并从提供商列表中选择 SAML v2.0。



### ① 信息

您可以访问 [Keycloak SAML 身份提供商 文档](#) 获取更多详细信息。

输入 别名 和 [从 URL 导入](#) 在 Keycloak IdP 编辑页面中。 Import from URL 的内容可以在Casdoor应用程序编辑页面找到。 点击 导入 和 SAML 配置将自动填充。

## Import External IDP Config

Import from URL 

Import from file

您应该记住 **服务供应商实体 ID**，然后保存配置。

## 配置 Casdoor 的 SAML 应用程序

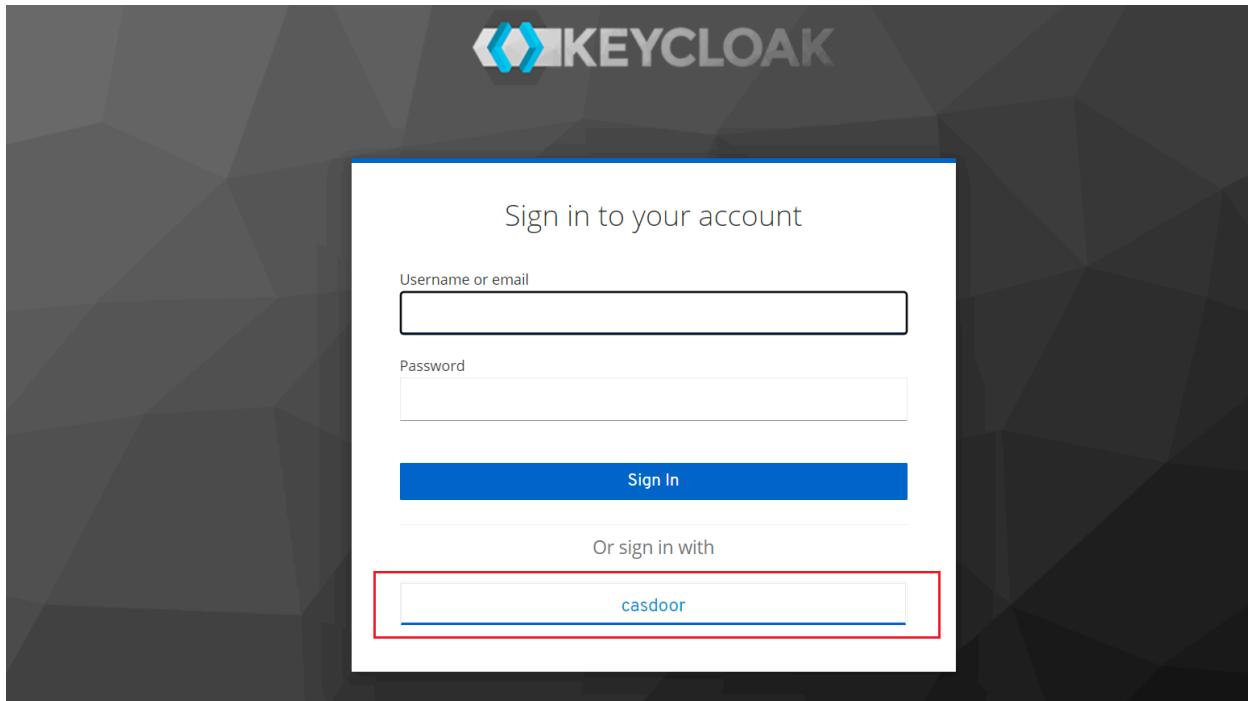
在应用程序编辑页面中添加一个重定向URL，其内容是 **服务供应商实体 ID**。您应该为 Keycloak 启用 SAML 压缩。

Enable SAML   
compress 

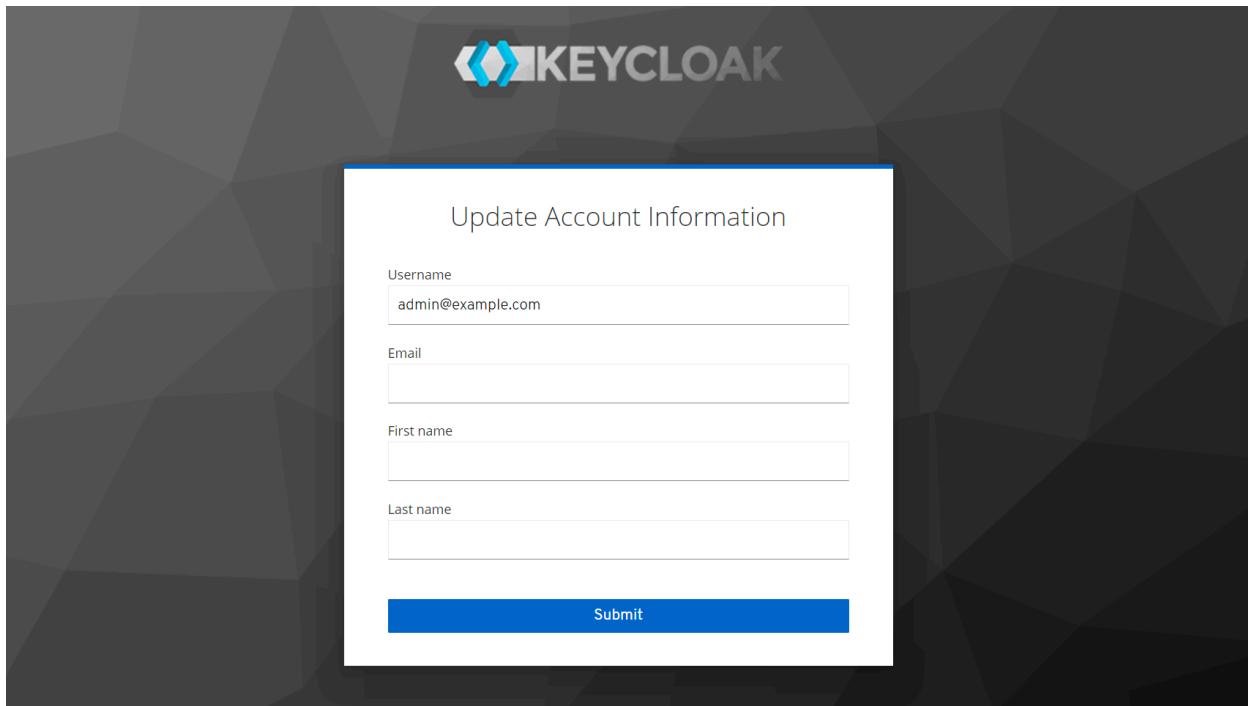
```
SAML metadata :
<EntityDescriptor xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns="urn:oasis:names:tc:SAML:2.0:metadata" xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" entityID="http://localhost:8000">
  <IDPSSODescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata" protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <KeyDescriptor use="signing">
      <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        <X509Data xmlns="http://www.w3.org/2000/09/xmldsig#">
          <X509Certificate xmlns="http://www.w3.org/2000/09/xmldsig#">MIIE+TCIAuGgAwIBAgIDAeJAMA0GCSqGSIb3DQEBCwUAMDYxHTAhBgNVBAoTFENhc2Rvb3IgT3JnYW5pemF0aW9uMRUwEwYDVQQDEwxDXNNkb29yIENlcnQwHhcNMjExMDExMDgxM
        </X509Data>
      </KeyInfo>
    </KeyDescriptor>
  <NameIDFormat urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</NameIDFormat>
  <NameIDFormat urn:oasis:names:tc:SAML:2.0:nameid-format:persistent</NameIDFormat>
  <NameIDFormat urn:oasis:names:tc:SAML:2.0:nameid-format:transient</NameIDFormat>
  <SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0-bindings:HTTP-Redirect" Location="http://localhost:7001/login/saml/authorize/admin/app-built-in"></SingleSignOnService>
```

## 使用 Casdoor SAML 登录

打开 Keycloak 登录页面，并且您可以找到额外的按钮，允许您使用 Casdoor SAML 提供商登录到 Keycloak。



点击按钮，您将被重定向到 Casdoor SAML 提供商进行身份验证。验证成功后，您将被重定向到Keycloak。然后您需要将用户分配到应用程序。



我们还提供一个演示录像，以展示整个进程，我们希望这将对你们有所帮助。





&gt;

开发指南

# 开发指南

## 前端

Casdoor 前端发展指南

## 生成 Swagger 文件

生成 Swagger 文件

# 前端

Casdoor的前端源代码在 `/web` 文件夹内: <https://github.com/casdoor/casdoor/tree/master/web>

这是一个 **Create-React-App (CRA)** 项目，其经典的 CRA 文件夹结构如下：

文件/目录	描述
public	React的 HTML 根文件
src	源代码
craco.config.js	Craco 配置文件可以在此更改主题颜色 (默认情况下为蓝色)
crowdin.yml	Crowdin i18n 配置文件
package.json	NPM/Yarn 依赖文件
yarn.lock	Yarn lockfile

在 `/src` 中，有以下几个重要文件或文件夹：

文件/目录	描述
account	已登录用户的“个人资料”页面
auth	所有与身份验证相关的代码，如OAuth、SAML、注册

文件/目录	描述
	页面、登录页面、忘记密码页等。
backend	调用 Go 后端 API 的 SDK 包含所有 <code>fetch()</code> 调用
basic	Casdoor的主页(控制面板页面) 包含几个卡片小部件
common	共享界面小部件
locales	JSON中的i18n 翻译文件与我们的 Crowdin 项目同步： <a href="https://crowdin.com/project/casdoor-site">https://crowdin.com/project/casdoor-site</a>
App.js	导入JS文件，包含所有路由
Setting.js	其他代码使用的实用函数
OrganizationListPage.js	组织列表的页面，类似于所有其他“XXXListPage.js”格式的页面
OrganizationEditPage.js	组织编辑的页面，类似于所有其他“XXXEditePage.js”格式的页面

# 生成 Swagger 文件

## 概述

我们知道，beego框架为生成交换文件提供支持，以便通过称为“bee”的命令行工具清除api。Casdoor也以beego为基础。但我们发现bee生成的swagger文件未能将api分类为“@Tag”标签，我们修改了原bee以执行功能。

## 如何写comment

大多数规则与原bee comment格式完全相同，唯一的差异是api必须按照“@Tag”标签分成不同的组别，因此，开发者有义务确保正确添加此标签。下面是一个示例：

```
// @Title Login
// @Tag Login API
// @Description login
// @Param oAuthParams     query    string  true      "oAuth
parameters"
// @Param body    body    RequestForm  true      "Login
information"
// @Success 200 {object} controllers.api_controller.Response The
Response object
// @router /login [post]
func (c *ApiController) Login() {
```

具有相同“@Tag”标签的api将会被放入同一个组。

# 如何生成swagger文件

0. 以正确的格式写入 api
1. 获取资源库 <https://github.com/casbin/bee>
2. 构建修改过的bee，例如在casbin/bee的根目录中运行

```
buid -o mybee
```

3. 复制mybe到casdoor的基础目录
4. 在该目录中运行

```
mybee generate docs
```

之后你会发现生成新的swagger文件。



&gt;

组织

# 组织

## 概述

Casdoor的基本单位 — 组织

## 账户自定义

自定义用户帐户项目

# 概述

组织是Casdoor的基本单位，它管理用户和应用。如果用户已登录过一个组织，那么此后他可以访问所有归属于该组织的应用，无需登录。

在 [应用程序](#) 和 [提供商](#) 中，选择一个组织很重要，它决定一个用户是否能够使用特定的提供商访问该应用程序。

我们还可以在Casdoor建立LDAP。欲了解更多详情，请参阅 [文档](#)。

Casdoor提供了多种密码存储算法，可在组织编辑页面中选择。

名称	算法	描述	场景
plain	-	密码将以明文形式存储。(默认)	-
salt	SHA256	SHA-256 是一个获得专利的加密哈希函数，输出256位长的值。	-
md5-salt	MD5	MD5 message-digest algorithm是一种加密中断但仍广泛使用的哈希函数，可产生128位哈希值。	Discuz!
bcrypt	bcrypt	bcrypt 是一个密码哈希函数，用于安全地对密码进行哈希和加密。	Spring Boot, WordPress
pbkdf2-salt	SHA256 和 PBKDF2	PBKDF2 是一个简单的加密密钥函数，能够抗拒 dictionary attacks 和 rainbow table attacks。它最初在 Casdoor 里用	Keycloak

名称	算法	描述	场景
		于Keycloak 语法。 如果您通过Keycloak 同步方式导入用户，请选择此选项。	

# 账户自定义

## 介绍

在组织中，您可以自定义用户的 **账户项**。这包括每个项目是否是 **可见**。如果可见，其 **查看规则** 和 **修改规则**。

当您自定义一个组织中的账户项目时，此配置 将对该组织所有成员的主页生效。

## 如何定制？

账户项目有四个属性：

属性	可选值	描述
Name	-	账户名称。
Visible	<input type="checkbox"/> 是 / <input type="checkbox"/> 否	选择此账户项是否在用户主页上可见。
ViewRule	<input type="checkbox"/> 规则项	选择用于查看帐户项目的规则。
ModifyRule	<input type="checkbox"/> 规则项	选择用于修改帐户项的规则。

输入组织编辑页面，您可以找到以下内容：

Name	visible	viewRule	modifyRule	Action
Organization	<input checked="" type="checkbox"/>	Public	Admin	  
ID	<input checked="" type="checkbox"/>	Public	Immutable	  
Name	<input checked="" type="checkbox"/>	Public	Admin	  
Display name	<input checked="" type="checkbox"/>	Public	Self	  
Avatar	<input checked="" type="checkbox"/>	Public	Self	  
User type	<input checked="" type="checkbox"/>	Public	Admin	  
Password	<input checked="" type="checkbox"/>	Self	Self	  
Email	<input checked="" type="checkbox"/>	Public	Self	  
Phone	<input checked="" type="checkbox"/>	Public	Self	  
Country/Region	<input checked="" type="checkbox"/>	Public	Self	  
Location	<input checked="" type="checkbox"/>	Public	Self	  
Affiliation	<input checked="" type="checkbox"/>	Public	Self	  
Title	<input checked="" type="checkbox"/>	Public	Self	  
Homepage	<input checked="" type="checkbox"/>	Public	Self	  
Bio	<input checked="" type="checkbox"/>	Public	Self	  
Tag	<input checked="" type="checkbox"/>	Public	Admin	  
Signup application	<input checked="" type="checkbox"/>	Public	Admin	  
3rd-party logins	<input checked="" type="checkbox"/>	Self	Self	  

Casdoor 提供非常简单的操作来配置：

- 将项目设置为可见或不可见

Name	visible	viewRule	modifyRule
Organization	<input checked="" type="checkbox"/>	Public	Admin
ID	<input type="checkbox"/>		
Name	<input checked="" type="checkbox"/>	Public	Admin
Display name	<input checked="" type="checkbox"/>	Public	Self
Avatar	<input checked="" type="checkbox"/>	Public	Self
User type	<input checked="" type="checkbox"/>	Public	Admin

- 设置查看和修改规则

visible	viewRule	modifyRule	Action
<input checked="" type="checkbox"/>	Public	Admin	<input type="button" value="^"/> <input type="button" value="v"/> <input type="button" value="Delete"/>
<input type="checkbox"/>	Public		<input type="button" value="^"/> <input type="button" value="v"/> <input type="button" value="Delete"/>
<input checked="" type="checkbox"/>	Self	Admin	<input type="button" value="^"/> <input type="button" value="v"/> <input type="button" value="Delete"/>
<input checked="" type="checkbox"/>	Admin	Self	<input type="button" value="^"/> <input type="button" value="v"/> <input type="button" value="Delete"/>

有 3 条规则：

- 公开：每个人都有权限
- 自我：用户有自己的权限
- 管理员：管理员拥有权限

## 账户列表

以下是帐户项目中的所有字段。 关于描述详情，您可以看 [用户](#)。

- 组织
- ID
- 姓名
- 显示名称
- 头像
- 用户类型
- 密码
- 邮件
- 手机号
- 国家/地区

- 城市
- 附属机构
- 职务
- 个人主页
- 自我介绍
- 标签
- 注册应用
- 第三方登录
- 属性
- 是否为管理员
- 是否为全局管理员
- 是否被禁用
- 是否已删除



&gt;

应用

# 应用

## 概览

Casdoor应用概述

## 应用程序配置

配置您的应用程序身份验证

## 注册项目表

配置注册项表以创建自定义注册页面

## 术语参考

术语参考

# 概览

Casdoor的每个应用程序都是一个application，它们之间没有关联并且不会相互影响，这意味着，你可以根据需要单独部署或删掉其中任何一个应用程序。

如果您想要使用 Casdoor 为您的页面应用提供登录服务，您可以将其添加为Casdoor 应用程序。

这样用户在访问组织中的所有应用程序时就无需重复登录。

应用程序配置非常灵活和简单。 您可以设置是否允许密码登录或第三方登录，配置您想要用户用以登录的第三方应用程序，您甚至可以自定义应用程序的注册条目等。

在本章中，您将学到如何从零开始将Casdoor配置进您自己的程序。

让我们一起探索吧！

# 应用程序配置

当您在您的服务器上部署Casdoor并设置您的组织后，您就可以部署您的应用程序了！

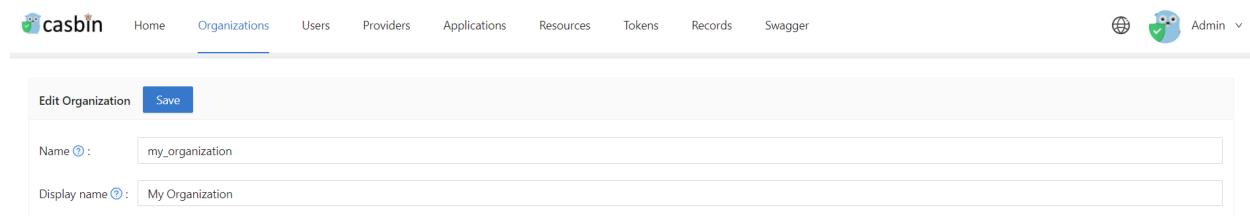
让我们看看如何使用Casdoor配置您的应用程序的身份认证系统吧！

## ⓘ 备注

例如，在这里，我想要使用 [Casnode](#) 设置我的论坛

创建我的应用程序并填写一些必要的配置。

选择我创建的组织，来使这个组织中的用户可以使用此应用程序。



The screenshot shows the Casbin web application's organization management page. At the top, there is a navigation bar with links for Home, Organizations (which is highlighted in blue), Users, Providers, Applications, Resources, Tokens, Records, and Swagger. On the far right, there is a user profile icon and the text "Admin". Below the navigation bar, there is a form titled "Edit Organization". The "Name" field contains the value "my\_organization". The "Display name" field contains the value "My Organization". There is a "Save" button at the top right of the form.

这个组织名为 `my_organization`，所以我在下拉菜单中选择它。

Edit Application Save

Name ? : my\_forum

Display name ? : My Forum

Logo ? : URL: [https://cdn.casbin.com/logo/logo\\_1024x256.png](https://cdn.casbin.com/logo/logo_1024x256.png)

Preview: 

Home ? :

Description ? :

Organization ? : built-in

Client ID ? : my\_organization  
built-in

然后我希望我的用户在注册时可以使用Casdoor来完成身份认证。 所以我在这里填写 redirect url为 <https://my-site-url.com/callback>

### ⚠ 注意事项

所以，我们需要记住提供商应用程序中的 **callback URL** 是 Casdoor的回调url，和Casdoor中的 **Redirect URL** 是 您网站的callback url。

### 更深层次的理解

如果我想要认证进度正常运行，详细步骤如下：

用户发送请求到 Casdoor, Casdoor 使用 `Client ID` (客户端ID) 和 `Client Secret` (客户端密钥) 获取GitHub、谷歌或其他供应商的身份验证。

如果身份验证成功, GitHub 回调到 Casdoor 告诉Casdoor验证成功, 因此 GitHub 授权callback URL应该是我的Casdoor callback URL, 它是 <http://your-casdoor-url.com/callback>, 然后Casdoor 告诉应用程序身份验证成功, 意味着Casdoor callback URL应该变成了我的应用程序的cassback URL, 也就是是 <http://your-site-url.com/callback>。

然后你可以通过添加提供商和设置它的属性来添加第三方应用程序。

Providers	Add	Name	canSignUp	canSignIn	canUnlink	prompted	Action
provider_casbin_email	<input type="button" value="^"/>	<input type="button" value="v"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="^"/> <input type="button" value="v"/> <input type="button" value="o"/>
provider_casbin_sms	<input type="button" value="^"/>	<input type="button" value="v"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="^"/> <input type="button" value="v"/> <input type="button" value="o"/>
provider_storage_aliyun_oss	<input type="button" value="^"/>	<input type="button" value="v"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="^"/> <input type="button" value="v"/> <input type="button" value="o"/>
provider_casdoor_github_localhost	<input type="button" value="^"/>	<input type="button" value="v"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="^"/> <input type="button" value="v"/> <input type="button" value="o"/>
provider_casdoor_github	<input type="button" value="^"/>	<input type="button" value="v"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="^"/> <input type="button" value="v"/> <input type="button" value="o"/>
provider_casdoor_google	<input type="button" value="^"/>	<input type="button" value="v"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="^"/> <input type="button" value="v"/> <input type="button" value="o"/>
provider_casdoor_qq	<input type="button" value="^"/>	<input type="button" value="v"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="^"/> <input type="button" value="v"/> <input type="button" value="o"/>
provider_casdoor_wechat	<input type="button" value="^"/>	<input type="button" value="v"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="^"/> <input type="button" value="v"/> <input type="button" value="o"/>
provider_casdoor_facebook	<input type="button" value="^"/>	<input type="button" value="v"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="^"/> <input type="button" value="v"/> <input type="button" value="o"/>
provider_casdoor_gitee	<input type="button" value="^"/>	<input type="button" value="v"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="^"/> <input type="button" value="v"/> <input type="button" value="o"/>
provider_casdoor_gitlab	<input type="button" value="^"/>	<input type="button" value="v"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="^"/> <input type="button" value="v"/> <input type="button" value="o"/>

You need to enable JavaScript to run this app.

### 提示

注意，如果您不希望用户使用**用户名/密码**访问您的应用， 您可以关闭 **Password On** 按钮，这样用户就只能使用第三方服务访问应用：

Token expire [?](#) :  Hours

Password ON [?](#) :

Enable signup [?](#) :

# 注册项目表

在应用程序配置页面上，我们可以配置注册项表来创建一个自定义注册页面。我们可以在此注册项表上添加或删除任何注册项。



The screenshot shows a configuration interface for 'Signup items'. It lists several fields: Name, ID, Username, Display name, Password, ID card, Email, and Agreement. Each field has columns for 'visible' (checkbox), 'required' (checkbox), 'prompted' (checkbox), 'rule' (dropdown menu with options: Random, First, last, Normal), and 'Action' (button). The 'ID' field is currently selected.

关于每个注册项的详细说明，请见下表。

参数	可选值	描述
Name	-	注册项名称
visible	True / False	选择此注册项是否在注册页面上可见
required	True / False	选择此注册项是否必须要填写
prompted	True / False	选择当用户忘记填写此注册项时是否给出提示
rule	Rule Items	规则。规则可以为这个注册项添加一些自定义要求。详细规则见下表。

参数	可选值	描述
Action	-	用户可以将这个注册项向上移动、向下移动，或删除这个注册项。

到目前为止，支持配置注册项的规则包括 `ID`, `Display name` 和 `Email`。

规则	可选规则	描述
ID	<code>Random</code> / <code>Incremental</code>	选择用户ID是随机生成还是递增生成。
Display name	<code>None</code> / <code>Real name</code> / <code>First, last</code>	选择名称的展示方式。选择 <code>None</code> 将展示 <code>Display name</code> ，选择 <code>Real name</code> 将展示 <code>Real name</code> ，选择 <code>First, last</code> 将展示 <code>First name</code> 和 <code>last name</code> 。
Email	<code>Normal</code> / <code>No verification</code>	选择是否验证邮箱验证码。选择 <code>Normal</code> 将验证邮箱验证码。选择 <code>No verification</code> 则不验证邮箱验证码。

### ⓘ 备注

例如，我想设置需要注册邮箱的注册页面，但不需要邮件验证码来验证此邮箱。

首先，我添加了一些注册所需的注册项目，例如ID、用户名、密码、电子邮件。

Signup items	Add	visible	required	prompted	rule	Action
Name		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Incremental	<input checked="" type="checkbox"/> ▲ ▼ ⌂
ID		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/> ▲ ▼ ⌂
Username		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/> ▲ ▼ ⌂
Password		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/> ▲ ▼ ⌂
Email		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	No verification	<input checked="" type="checkbox"/> ▲ ▼ ⌂

然后设置电子邮件的验证规则为 `No verification`, 然后生成的注册页面就可以实现该效果。



\* Username:

\* Password:  

\* Email:

[Sign Up](#) Have account? [sign in now](#)

# 术语参考

- `Name` 被创建的应用程序的名字
- `CreatedTime` 应用程序被创建的时间
- `DisplayName` 应用程序向公众显示的名称
- `Logo` 应用程序Logo将显示在登录和注册页面
- `HomepageUrl` 应用程序主页的 url
- `Description` 描述应用程序
- `Organization` 应用程序所属的组织
- `EnablePassword` 用户是否可以通过密码登录
- `EnableSignUp` 用户是否可以注册。如果没开启，应用程序不能注册新用户
- `SignupItems` 用户注册时需要填写的条目
- `Providers` 为应用程序提供的服务(例如OAuth、电子邮件、短信服务)
- `ClientId` OAuth客户端ID
- `ClientSecret` OAuth客户端密码
- `RedirectUries` 如果用户成功登录，Casdoor将导航到uries中的一个uri
- `ExpireInHours` 登录将在几小时后过期
- `SigninUrl`
- `SignupUrl` 如果您在Casdoor之外独立提供注册服务，请在此填写url
- `ForgotUrl` 与 `SignupUrl` 相同
- 联盟网址



&gt;

权限

# 权限

## 概述

使用 Casbin 管理用户在组织中的访问权限

## Casbin运行

使用开放的Casbin API管理用户在组织中的访问权限

# 概述

## 介绍

Casdoor 中每个组织的应用程序都共享该组织中的所有用户，因此这些用户都可以访问这些应用程序。要限制这些用户访问其中的某些应用程序，您可以使用由 [Casbin](#) 实现的 [Permission](#) 机制。

每个权限的内部都有 [Sub users](#) 和 [Resources](#) 属性，它们可用于检查用户使用哪个应用程序登录。此外，它们还支持配置自定义模型，以满足用户的多种需求。

请参阅下面的示例以更清楚地了解 Casdoor 的适用于应用程序的权限控制。

## 应用程序权限

在使用 [Permission](#) 之前，您需要创建一个基于 PERM 元模型并被抽象成一个 CONF 文件的 [Model](#)。您可以访问 [Casbin 文档](#) 以获取更多信息。建议您使用[Casbin在线编辑器](#) 来设计模型和检查语法。

点击 [Model](#) 标签并添加一个新模型。在编辑页面中，您可以设置自定义模型，如 [Model text](#) 中的 ACL 模型。

```
[request_definition]
r = sub, obj, act
```

```
[policy_definition]
p = sub, obj, act
```

Home   Organizations   Users   Roles   Permissions   **Models**   Providers   Applications   Resources   Tokens   Records   Webhooks   Syncers   Certs   ...

Admin

Edit Model

Organization [?](#): built-in

Name [?](#): model-built-in

Display name [?](#): Built-in Model

Model text [?](#):

```
[request_definition]
r = sub, obj, act

[policy_definition]
p = sub, obj, act

[policy_effect]
e = some(where (p.eft == allow))

[matchers]
m = r.sub == p.sub && r.obj == p.obj && r.act == p.act
```

Is enabled [?](#):

Made with ❤ by Casdoor

点击 **Permissions** 标签并添加一个新的权限。在编辑页面中，您需要像下面一样选择模型、适配器、子用户、资源和操作。

Home   Organizations   Users   Roles   **Permissions**   Models   Providers   Applications   Resources   Tokens   Records   Webhooks   Syncers   Certs   ...

Admin

Edit Permission

Organization [?](#): built-in

Name [?](#): permission-built-in

Display name [?](#): Built-in Permission

Model [?](#): model-built-in

Adapter [?](#): permission\_rule\_builtin

Sub users [?](#): built-in/admin ✕ built-in/seriouszyx ✕ built-in/test ✕

Sub roles [?](#):

Sub domains [?](#):

Resource type [?](#): Application

Resources [?](#): app-built-in ✕

Actions [?](#): Read ✕ Write ✕ Admin ✕

Effect [?](#): Allow

Is enabled [?](#):

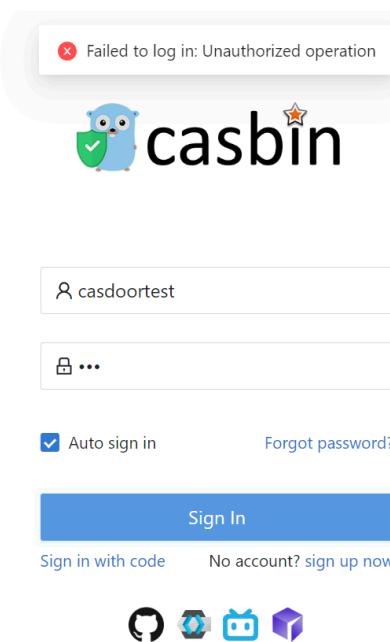
Submitter [?](#):

Approver [?](#):

## ① 信息

`Adapter` 字段用于指定保存策略的表的名称。如果此字段为空，策略将存储在 `权限规则` 表中。我们强烈建议 **为不同模型指定不同的适配器**，因为将所有的策略保存在同一个表格中可能导致冲突。

保存后，用户 `test`、`seriouszyx` 和 `admin` 就可以登录应用程序 `app-built-in` 了。其他用户，如 `casdoortest` 则不能。



# Casbin运行

## 开放的Casbin API

在casdoor的底层，每个许可对应一个casbin运行器。 我们提供了一些API来直接调用这些执行器。 您可以使用它们来获得更灵活的权限管理功能。

我们将展示一些这些API的使用示例，权限如下。

The screenshot shows the Casdoor web interface under the 'Permissions' tab. The form is titled 'Edit Permission' with a 'Save' button and a 'Save & Exit' button. The fields are as follows:

- Organization: built-in
- Name: permission-built-in
- Display name: Built-in Permission
- Model: model-built-in
- Adapter: permission\_rule\_builtin
- Sub users: built-in/admin, built-in/seriouszyx, built-in/test
- Sub roles: (empty)
- Sub domains: (empty)
- Resource type: Application
- Resources: app-built-in
- Actions: Read, Write, Admin
- Effect: Allow
- Is enabled:
- Submitter: (empty)
- Approver: (empty)



您应该登录到 Casdoor 并从 cookie 中获取 `casdoor_session_id`。

## 运行

请求：

```
curl --location --request POST 'http://localhost:8000/api/enforce' \
--header 'Content-Type: text/plain' \
--header 'Cookie:
casdoor_session_id=b1888c74a7903f1813bf8c34269b0118' \
--data-raw '{"id":"built-in/permission-built-in", "v1":"app-built-in", "v2":"write"}'
```

响应：

```
true
```

## 批量运行

请求：

```
curl --location --request POST 'http://localhost:8000/api/batch-enforce' \
--header 'Content-Type: text/plain' \
--header 'Cookie:
casdoor_session_id=b1888c74a7903f1813bf8c34269b0118' \
--data-raw '[{"id":"built-in/permission-built-in", "v1":"app-built-in", "v2":"write"}, {"id":"built-in/permission-built-in", "v1":"app-built-in", "v2":"read"}, {"id":"built-in/permission-built-in", "v1":"app-casnnode", "v2":"write"}]'
```

响应：

```
[  
    true,  
    true,  
    false  
]
```

## GetAllObjects

请求：

```
curl --location --request GET 'http://localhost:8000/api/get-all-  
objects' \  
--header 'Cookie:  
casdoor_session_id=b1888c74a7903f1813bf8c34269b0118'
```

响应：

```
[  
    "app-built-in"  
]
```

## GetAllActions

请求：

```
curl --location --request GET 'http://localhost:8000/api/get-all-  
actions' \  
--header 'Cookie:  
casdoor_session_id=b1888c74a7903f1813bf8c34269b0118'
```

响应：

```
[  
  "read",  
  "write",  
  "admin"  
]
```

## GetAllRoles

请求:

```
curl --location --request GET 'http://localhost:8000/api/get-all-  
roles' \  
--header 'Cookie:  
casdoor_session_id=b1888c74a7903f1813bf8c34269b0118'
```

响应:

```
[  
  "role_kcx661"  
]
```



&gt;

提供商

# 提供商

## 概述

添加第三方服务到您的应用程序

## OAuth

19 个项目

## 电子邮箱

使用电子邮件来完成身份验证

## 短信

使用短信来完成身份验证



2 个项目



3 个项目



1 个项目



5 个项目

# 概述

Casdoor 使用提供商为平台提供第三方服务。 在本章中，您将学习如何为Casdoor添加提供商。

Casdoor 将只使用组织所有者在收到组织用户的请求时添加的提供商。

现在，我们有5种提供商：

- **OAuth 提供商**

允许用户通过其他 OAuth 应用程序登录。 您可以将 GitHub、Google、QQ 和其他许多OAuth 应用程序添加到Casdoor。 欲了解更多详情，请参阅 [OAuth](#)。

- **短信提供商**

当用户想要验证他们的电话号码时，Casdoor将发送短信给他们。 短信提供者被用来发送Casdoor短信。

- **电子邮件提供商**

电子邮件提供者与短信提供者类似。

- **存储提供商**

Casdoor允许用户使用本地文件系统或云端服务存储文件。

- **支付服务提供商**

Casdoor 可以添加付款提供者，用于在产品页面上添加付款方法。 目前，受支持的付款提供者包括支付宝、微信支付、PayPal和GC。

- **验证码提供商**

Casdoor支持在用户流程中配置验证码。 目前，支持的验证码包括默认验证码、reCAPTCHA、hCaptcha和Aliyun 验证码。

# OAuth

## 概述

将 OAuth 提供商添加到您的应用程序

## 自定义提供商

添加您自己的自定义OAuth 提供商

## Twitter

添加Twitter OAuth 提供商到您的应用程序

## 微博

将 Weibo OAuth 提供商添加到您的应用程序

 **微信**

将Wechat OAuth 提供商添加到您的应用程序

 **企业微信**

将 WeCom OAuth 提供商添加到您的应用程序

 **腾讯 QQ**

添加腾讯QQ OAuth提供商到您的应用程序。

 **钉钉**

添加钉钉 OAuth 到您的应用程序

 **Steam**

将 Steam OAuth 添加到您的应用程序



## GitHub

添加Github OAuth 提供商到您的应用程序



## Gitee

添加Gitee OAuth 提供商到您的应用程序



## 领英 (LinkedIn)

添加Linkedin OAuth 提供商到您的应用程序



## Facebook

将 Facebook OAuth 提供商添加到您的应用程序



## 谷歌

添加Google OAuth 提供商到您的应用程序



## 百度

向您的应用程序添加 Baidu OAuth 提供商



## AD FS

添加AD FS 作为第三方服务来完成身份验证



## AzureAD

添加 AzureAD 作为第三方服务来完成身份验证



## Infoflow

在应用程序中添加 Infoflow OAuth 提供商



## Okta

将 Octa OAuth 提供商添加到您的应用程序

# 概述

Cadoor 可以使用其他 OAuth 应用程序登录。

现在，Casdoor 支持许多 OAuth 应用程序提供者。提供商的图标将在添加到 Casdoor 后显示在登录和注册页面中。以下是 Casdoor 支持的提供商：

Google	GitHub	Facebook	Twitter	LinkedIn	微博	微信	腾讯 QQ	钉钉	百度	Infoflow	Gitee	Steam	Okta	电子邮箱	短信
✓	✓	✓	🚧	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

我们将向您展示如何申请第三方服务并将其添加到 Casdoor。

## 申请成为开发者

在此之前，你需要理解一些概念。

- **RedirectUrl**, 认证后重定向地址, 填写您的应用程序地址, 例如 <https://forum.casbin.com/>
- **Scope**, 用户授予您的权限, 如基本个人资料, 电子邮件地址和帖子及其他。
- **ClientId/AppId, ClientKey/AppSecret**, 这是最重要的信息而且这是您在申请开发者帐户后需要得到的信息。您无法与任何人共享 的密钥。

## 添加 OAuth 提供商

1. 导航到您的 Casdoor 索引页面
2. 点击顶部栏中的 **提供商**
3. 点击 **添加**, 然后您可以在列表顶部看到一个新的提供商
4. 点击新的提供商修改它
5. 选择 **OAuth** 在 **类别** 中
6. 在 **类型** 中选择您需要的 OAuth 提供程序
7. 填写最重要的导入信息, **client ID** 和 **Client Secret**

## 应用中

1. 单击顶部栏中的 **应用程序** 并选择一个应用程序, 编辑
2. 点击提供商添加按钮, 选择您刚刚添加的提供商
3. 修改提供商的权限, 例如允许注册、登录和取消绑定
4. 完成!

# 自定义提供商

## ① 备注

Casdoor 支持自定义提供商，但自定义提供商必须遵循 3-legged OAuth 的标准流程。和 `Token URL` 和 `Userinfo URL` 的返回值必须遵循 Casdoor 指定的格式。

首先，前往 Casdoor 的供应商页面并创建一个新的供应商。在类型项中选择“自定义”。除了 `Client ID` 和 `Client Secret` 您需要填写 `Auth URL`, `Scope`, `Token URL`, `Userinfo URL` 和 `Favicon`

Type ② :

Custom

Auth URL ②

<https://door.casdoor.com/login/oauth/authorize>

Scope ②

openid profile email

Token URL ②

[https://door.casdoor.com/api/login/oauth/access\\_token](https://door.casdoor.com/api/login/oauth/access_token)

Userinfo URL ②

<https://door.casdoor.com/api/userinfo>

Favicon ② :

URL ② :



Preview:

Client ID ②



Client secret ②



- `Auth URL` 是自定义提供商的 OAuth 登录页面地址。

假定我们填写 <https://door.cassdoor.com/login/oauth/auth/authorization> at `Auth URL`, 然后当用户登

录到此自定义提供商时，浏览器将先跳转到

```
https://door.casdoor.com/login/oauth/
authorize?client_id={ClientID}&redirect_uri=https://{{your-casdoor-
hostname}}/callback&state={State_generated_by_Casdoor}&response_type=code&scope={Scope}`
```

授权完成后，自定义提供商应该重定向到

```
https://{{your-casdoor-hostname}}/callback?code={code}
```

然后，此 URL 中的代码参数将会被 Casdoor 识别。

- **Scope** 是访问 **Auth URL** 时携带的范围参数 它是根据自定义提供商的要求填写的。
- **Token URL** 是获取 accessToken 的 API 地址。

在上一步获取代码后，Casdoor 需要使用此代码获取 accessToken。

假定我们在 **https://door.casdoor.com/api/login/oauth/access\_token** 在 **Token URL** 中填写，然后 Casdoor 将访问 Token URL 如下所示：

```
curl -X POST -u "{ClientID}:{ClientSecret}" --data-binary
"code={code}&grant_type=authorization_code&redirect_uri=https://{{your-casdoor-
hostname}}/callback" https://door.casdoor.com/api/login/oauth/access_token
```

自定义提供商至少应该返回以下内容

```
{
  "access_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6Ixxxxxxxxxxxxxx",
  "refresh_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6xxxxxxxxxxxxxx",
  "token_type": "Bearer",
  "expires_in": 10080,
  "scope": "openid profile email"
}
```

- **Token URL** 是获取 accessToken 的 API 地址。

假定我们在 **https://door.casdoor.com/api/userinfo** 在 **Token URL** 中填写，然后 Casdoor 将访问 Token URL 如下所示：

```
curl -X GET -H "Authorization: Bearer {accessToken}" https://door.casdoor.com/api/
userinfo
```

自定义提供商至少应该返回以下内容

```
{  
  "name": "admin",  
  "preferred_username": "Admin",  
  "email": "admin@example.com",  
  "picture": "https://casbin.org/img/casbin.svg"  
}
```

- **Favicon** 是自定义提供商的标识URL。

这个标识将与其他第三方登录供应商一起显示在Casdoor的登录页面上。

# Twitter

## Twitter(仍在开发中🚧)

Twitter的应用步骤有点麻烦，官方限制有点严格，因此申请开发者帐户可能比其他第三方平台更困难。

访问[开发者门户](#)，如果您没有帐户，请注册。Twitter需要知道您正在申请哪些开发者帐户。你必须仔细填写它，否则它将不会通过。

申请获批后，创建申请书，填写回调地址和其他信息。您需要做两件事，将被设置为**认证设置**部分。

- 手动打开**3-legged OAuth**，用Twitter登录，代表其他账户发布Tweets等。
- 启用**Request email address from users**请求电子邮件地址以获取用户电子邮件地址。

# 微博

## 微博✓

申请微博的开发者账户并不困难，但速度相对较慢。大约需要2-3天。

访问 [开发者网站](#)，填写基本信息，等待审核通过。

审核通过后，您将得到 Client Id 和 Client Secret。

# 微信

## 微信:注意事项:

访问 [微信开放平台](#) 并注册成为开发者。在你的网站应用或移动应用获得批准后，您将得到您的 App ID 和 App Secret。

Home    Organizations    Users    Roles    Permissions    **Providers**    Applications    Resources    Token

---

Edit Provider Save Save & Exit

Name ? : provider\_79p43d

Display name ? : New Provider - 79p43d

Category ? : OAuth

Type ? : WeChat

Client ID ?  

Client secret ?  

Client ID 2 ?  

Client secret 2 ?  

Provider URL ? : <https://github.com/organizations/xxx/settings/applications/1234567>

---

Save Save & Exit

WeChat 提供商提供两套不同的密钥:

- 第一个密钥设置为 `WeChat Open Plat(assum-gending)`, 仅适用于 PC 场景。它可以在 PC 中显示二维码, 用户可以使用手机扫描代码。PC 浏览器允许使用 WeChat 登录。
- 第二个密钥设置为 `WeChat Media Platform (conction-classic difference)`, 它只适用于移动场景。它允许用户使用 WeChat 在 WeChat 移动 APP 内登录 它会跳转到您的 `WeChat 官方帐户 (微信公众号)` 以登录。

:::tip

我们建议同时设置两套钥匙。并链接您的 `WeChat 开放平台(即时扫描)` 账户和 `WeChat 媒体平台(即时连接)` 账户一起在 `WeChat 开放平台(即时连接)` 中 所以通过 PC 登录的 WeChat 用户和手机可以被识别为 Cassdoor 中的同一用户。

:::

① 备注

由于 WeChat OAuth 的限制, 目前没有办法通过 WeChat 登录到第三方移动 APP 或 WeChat APP 以外的移动浏览器。必须在 WeChat 中移动登录。

欲了解更多详情, 请访问 [微信开放平台](#)。

# 企业微信

## 介绍

企业微信 提供 OAuth 授权的登录方法。 它可以从 企业微信 终端打开的网页获取会员身份信息，从而消除登录需求。

有两种不同的应用程序类型： **内部** 应用程序和 **第三方** 应用程序

## 基本设置

要配置 企业微信 提供商，下面的表格描述了所需参数。

**参数描述:**

参数	描述
Sub type	内部或第三方
Method	静默或正常模式
Client ID	企业CorpID
Client secret	企业密钥
Agent ID	应用程序Agentid

## ① 信息

企业微信有两种授权方法。 **静默** 授权和 **正常** 授权。

**静默授权:** 在用户点击链接后，页面是 重定向\_URI? code=CODE&state=STATE

**正常授权:** 在用户点击链接之后，一个中间页面将显示给用户以选择是否为 授权。

用户确认授权后，转到 重定向uri?code=CODE&state=STATE

欲了解更多详情，请参阅 [文档](#)。

## 更多

欲了解更多关于内部应用程序的信息，请参阅 [内部应用程序](#)。

关于第三方应用程序，请参阅 [第三方应用](#)。

# 腾讯 QQ

## 腾讯QQ✓

访问QQ认证平台 - [连接 QQ](#)。

首先你需要应用 [成为开发者](#)。 审核通过后，遵循平台的指示，并获取您的客户ID和客户端密钥。

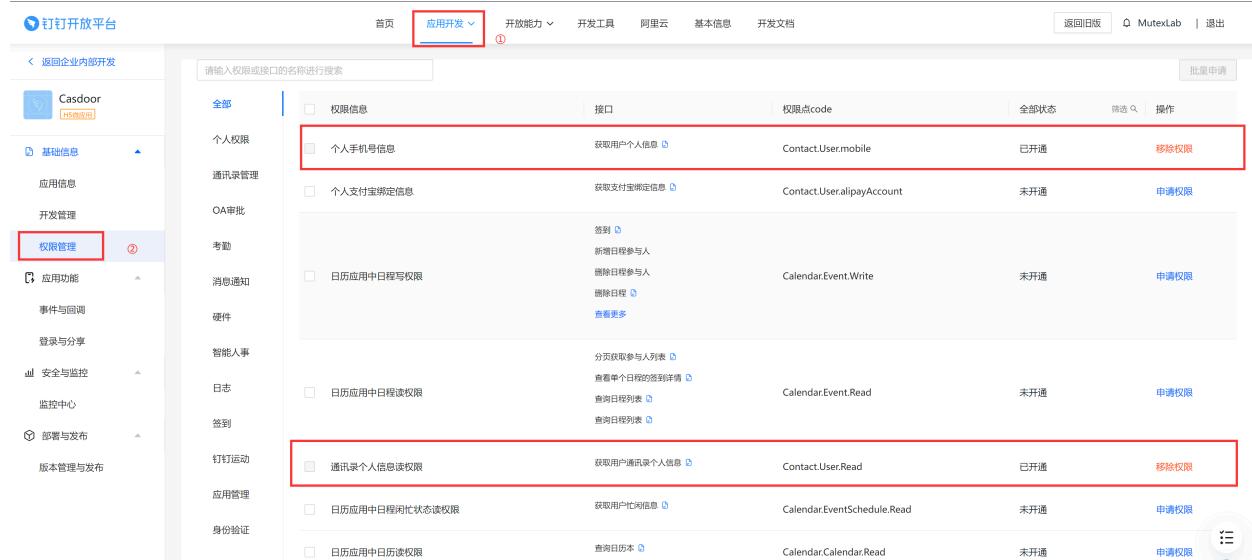
# 钉钉

## 钉钉✓

访问 [钉钉开放平台](#) 并使用您的钉钉账户登陆，成功登陆后，遵循平台的指导，您将会得到您的 Client Id 和 Client Secret。进入平台后，遵循平台的指示，您将会得到您的客户端ID和客户端的密钥。

欲了解更多信息，请访问 [钉钉开放平台](#)。

此外，您需要将以下权限添加到钉钉中：



The screenshot shows the DingTalk Open Platform's permission management interface. The left sidebar lists various application categories like Basic Information, Application Information, Development Management, and DingTalk Functions. The 'Permissions Management' section is selected and highlighted with a red box. The main content area displays a list of permissions categorized by type (All, Personal Permissions, Communication Record Management, OA Audit, Attendance, Message Notifications, Hardware, Intelligent Human Resources, Logs, Check-in, DingTalk Functions, Application Management, and Identity Verification). Several permissions are highlighted with red boxes:

- 个人权限: 个人手机号信息 (Contact.User.mobile) - 已开通, 移除权限
- 通讯录管理: 个人支付宝绑定信息 (Contact.User.alipayAccount) - 未开通, 申请权限
- 考勤: 日历应用中日程写权限 (Calendar.Event.Write) - 未开通, 申请权限
- 消息通知: 日历应用中日程读权限 (Calendar.Event.Read) - 未开通, 申请权限
- 钉钉运动: 通讯录个人信息读权限 (Contact.User.Read) - 已开通, 移除权限
- 应用管理: 日历应用中日程闲忙状态读权限 (Calendar.EventSchedule.Read) - 未开通, 申请权限
- 身份验证: 日历应用中日历读权限 (Calendar.Calendar.Read) - 未开通, 申请权限

# Steam

## Steam✓

访问 [Steam WebAPI platform](#) 并通过您的 Steam 账户登陆，之后为您的 Casdoor 域名或ip申请一个 API 密钥，最终将您的 API 密钥作为 Client Secret 填到 Casdoor 中。  
(ClientID 不需要被填写，您的 Steam 账户需要有游戏去申请 API)

欲了解更多详情，请访问 [Steam WebAPI doc](#)。

# GitHub

GitHub OAuth 支持网络应用程序流和设备流量。请继续阅读以获取 OAuth 凭据。

首先，请访问 [GitHub 开发者设置](#) 注册一个新的 GitHub 应用。

## ⚠ 注意事项

**Tricks:** 我们建议您使用 GitHub 应用程序替换 OAuth 应用程序 因为GitHub 应用程序可以添加多个重定向uri，可以在部署测试和生产环境时带来方便。 [GitHub](#) 官方也建议使用 GitHub 应用程序而不是 OAuth 应用。

## Settings / Developer settings

### GitHub Apps

### OAuth Apps

### Personal access tokens

然后填写 **应用名称**, **主页网址**, **描述** 和 **授权回调URL**.

GitHub App name \*

Casdoor

The name of your GitHub App.

Write

Preview

Markdown supported

A UI-first centralized authentication / Single-Sign-On (SSO) platform supporting OAuth 2.0, OIDC and SAML, integrated with Casbin RBAC and ABAC permission management

Homepage URL \*

http://door.casdoor.com

The full URL to your GitHub App's website.

Add Callback URL

Callback URL

http://localhost:7001/callback

Delete

Callback URL

https://door.casdoor.com/callback

Delete

## ❗ 正确设置授权回调URL

在 Github OAuth 配置中，**应用回调地址** 必须是 **您的 Casdoor 的回调 URL** 并且 Casdoor 中的 **Redirect URL** 应该为 **您的应用回调地址**

更多详情请阅读 [应用配置](#)

注册您的 GitHub 应用程序后，您现在可以生成您的 **客户端密钥**！

## About

Owned by: [REDACTED]

App ID: [REDACTED]

Client ID: lv1 [REDACTED] d2e

[Revoke all user tokens](#)

GitHub Apps can use OAuth credentials to identify users. Learn more about identifying users by reading our [integration developer documentation](#).

## Client secrets

[Generate a new client secret](#)



\*\*\*\*\*dba81954

Added 5 minutes ago by [REDACTED]

[Client secret](#)

[Delete](#)

Last used within the last week



\*\*\*\*\*15822f89

Added on 15 Feb by [REDACTED]

[Client secret](#)

[Delete](#)

添加一个 GitHub OAuth 提供商并填写 [客户端ID](#) and [客户端密钥](#)

Edit Provider

[Save](#) [Save & Exit](#)

Name <a href="#">?</a> :	provider_github_localhost
Display name <a href="#">?</a> :	provider_github_localhost
Category <a href="#">?</a> :	OAuth
Type <a href="#">?</a> :	GitHub
Client ID <a href="#">?</a>	lv` [REDACTED] .2e
Client secret <a href="#">?</a>	***
Provider URL <a href="#">?</a> :	<a href="https://github.com/organizations/xxx/settings/applications/1234567">https://github.com/organizations/xxx/settings/applications/1234567</a>

[Save](#)

[Save & Exit](#)

现在您可以使用 GitHub 作为第三方服务来完成认证。

# Gitee

要设置 Gitee OAuth 提供者, 请到 [Gitee 设置](#), 如果您之前没有创建过应用程序, gitee 工作台会是这样:



The screenshot shows the Gitee Workbench interface. At the top, there is a navigation bar with links for '特惠', '企业版', '高校版', '私有云', '博客', and '我的'. On the right side of the top bar are icons for search ('搜开源'), notifications ('通知'), location ('位置'), a plus sign ('+'), and a user profile ('个人中心'). Below the top bar, there are two tabs: '我的应用' (selected) and '已授权应用'. To the right of these tabs is a red button labeled '+ 创建应用'. The main content area is titled '无数据' with a magnifying glass icon.

然后您可以创建您的Gitee应用程序。

## 创建第三方应用

应用名称 \*

应用名称

应用描述

应用描述

应用主页 \*

你的应用主页

应用回调地址 \* +

用户授权后, 重定向的地址, 例如: <https://gitee.com/login>

填写 **应用名称**, **应用描述**, **应用主页** 和 **应用回调地址** 并仔细选择 **权限**

### ① 正确设置授权应用回调地址

在 Gitee OAuth 配置中，**应用回调地址** 必须是 **您的 Casdoor 的回调 URL** 并且 Casdoor 中的 **Redirect URL** 应该为 **您的应用回调地址**

更多详情请阅读 [应用配置](#)

然后您可以创建您的 gitee 应用并立即获得 **Client ID** 和 **Client Secrets**！

### Casdoor (今日请求次数: 0 次)

**应用名称 \***

Casdoor

**Client ID**

300ff94d994a7597850bbafb2d5dc67929676dd8e7176b029e067dc6966ef9c4

**Client Secret**

60be2e4e0f3fb8286cfe9f129ab0c3d6b40718a964dade150a8095eb2748730c

[重置 Client Secret](#)

[移除已授权用户的有效 Token](#)

添加一个 Gitee OAuth 提供商并在您的 Casdoor 中填写 **Client ID** 和 **Client Secrets**。

Edit Provider

Save

Name ③ :

my\_gitee\_provider

Display name ③ :

Gitee provider

Category ③ :

OAuth

Type ③ :

Gitee

Client ID ③

300ff94d994a7597850bbafb2d5dc67929676dd8e7176b029e067dc6966ef9c4

Client secret ③

\*\*\*\*\*

现在您可以使用 Gitee 作为第三方服务来完成身份验证！

### ⚠ 注意事项

由于 Casdoor 需要获取用户的电子邮件，必须选中电子邮件选项，否则会导致授权错误。

Permissions (Be careful to select scopes, users might deny authorization when there are too many scopes.)

All

- |                                     |               |  |
|-------------------------------------|---------------|--|
| <input checked="" type="checkbox"/> | user_info     | Access and update user data, activities, etc |
| <input type="checkbox"/>            | projects      | Full control of user projects                |
| <input type="checkbox"/>            | pull_requests | Full control of user pull requests           |
| <input type="checkbox"/>            | issues        | Full control of user issues                  |
| <input type="checkbox"/>            | notes         | Access, create and edit user comments        |
| <input type="checkbox"/>            | keys          | Full control of user public keys             |
| <input type="checkbox"/>            | hook          | Full control of user webhook                 |
| <input type="checkbox"/>            | groups        | Full control of user orgs and teams          |
| <input type="checkbox"/>            | gists         | Access, create and update user gists         |
| <input type="checkbox"/>            | enterprises   | Full control of user enterprises and teams   |
| <input checked="" type="checkbox"/> | emails        | Access user emails data                      |

Submit

Delete

# 领英 (LinkedIn)

要设置 Linkedin OAuth 提供商, 请前往 [Linkedin 开发者](#) 创建一个新的应用。

**DEVELOPERS** Products Docs and tools ▾ Resources ▾ My apps ▾

## Create an app

\* indicates required

**App name\***

**LinkedIn Page\***  
① This action can't be undone once the app is saved.  
The LinkedIn Company Page you select will be associated with your app. Verification can be done by a Page Admin. Please note this cannot be a member profile page. [Learn more](#)

[+ Create a new LinkedIn Page](#)

**Privacy policy URL**

**App logo\***  
This is the logo displayed to users when they authorize with your app  
 [Upload a logo](#)

填写上面的表单并创建您的应用后, 您需要验证与应用相关联的 LinkedIn 页面



## Identity Cloud Login

Client ID: 860t47n8b4jh7w | Created: Sep 4, 2020

**Settings**

Auth

Products

Analytics

Team members

### App settings

[Delete app](#)

Company:



**Identity Cloud Documentation**

Computer Software; 1-10 employees

[Verify](#)



This app is not verified as being associated with this company.

[Learn more](#)

① 备注

只有公司页面管理员可以验证您的应用，并允许您的应用

在您的应用验证后，您可以继续：

 Identity Cloud Login

Client ID: 860t47n8b4jh7w | Created: Sep 4, 2020

Settings Auth **Products** Analytics Team members

**Products**

Additional available products

 **Marketing Developer Platform**  
Build marketing experiences to reach the right audiences  
[View docs ↗](#) Select

 **Share on LinkedIn**  
Amplify your content by sharing it on LinkedIn  
[View docs ↗](#) Select

 **Sign In with LinkedIn**  
Let users easily sign in with their professional identity  
[View docs ↗](#) Select

为您的应用添加授权重定向URL作为 您的 Casdoor 回调URL。

**Authorized redirect URLs for your app**

*No redirect URLs added*

**+ Add redirect URL**

 正确设置授权的重定向 URL

在 Linkedin OAuth 配置中，**应用回调URL** 必须是 **您的Casdoor的回调URL** 并且 Casdoor 中的 **Redirect URL** 应该为 **您的应用回调地址**

更多详情请阅读 [应用配置](#)

然后你就可以获得你的 **Client ID** 和 **Client Secret**

## Application credentials

### Authentication keys

**Client ID:**

860t47n8b4jh7w

**Client Secret:**

.....



添加一个 Linkedin OAuth 提供商并在您的 Casdoor 中填写 **Client ID** 和 **Client Secret**。

Edit Provider

Save

Name [?](#) : my\_linkedin\_provider

Display name [?](#) : Linkedin provider

Category [?](#) : OAuth

Type [?](#) : LinkedIn

Client ID [?](#) : 860t47n8b4jh7w

Client secret [?](#) : \*\*\*\*

现在您可以使用 LinkedIn 作为第三方服务来完成身份验证！

# Facebook

要设置 Facebook OAuth 提供商, 请前往 [Facebook开发人员](#) 创建一个新的应用程序。

选择您要创建的应用类型。

## Select an app type

X

The app type can't be changed after your app is created.



Create or manage business assets like Pages, Events, Groups, Ads, Messenger and Instagram Graph API using the available business permissions, features and products.



### Consumer

Connect consumer products, and permissions, like Facebook Login and Instagram Basic Display to your app.



### Instant Games

Create an HTML5 game hosted on Facebook.



### Gaming

Connect an off-platform game to Facebook Login.



### Workplace

Create enterprise tools for Workplace from Facebook.



### None

Create an app with combinations of consumer and business permissions and products.

[Learn More About App Types](#)

Cancel

Continue

填写姓名并联系电子邮件后，您可以进入 facebook 开发者面板。

FACEBOOK for Developers

Docs Tools Support My Apps Search developer documentation

Casdoor App ID: 1231340483981478 In development

Dashboard Settings Roles Alerts App Review Products Add Product

Add a Product

**Facebook Login**  
The world's number one social login product.  
Read Docs Set Up

**Audience Network**  
Monetize your app and grow revenue with ads from Facebook advertisers.  
Read Docs Set Up

**App Events**  
Understand how people engage with your business across apps, devices, platforms and websites.  
Read Docs Set Up

**Messenger**  
Customize the way you interact with people on Facebook.  
Read Docs Set Up

**Webhooks**  
Subscribe to changes and receive updates in real time.  
Read Docs Set Up

**Instant Games**  
Create a cross-platform HTML 5 game hosted on Facebook.  
Read Docs Set Up

然后设置 Facebook 登录：



## Facebook Login

The world's number one social login product.

Read Docs

Set Up

选择此应用的 Web 平台：

Use the Quickstart to add Facebook Login to your app. To get started, select the platform for this app.



iOS



Android



Web



Other

填写网站URL后，您可以访问 [Facebook Login > 设置](#) 并填写有效的 OAuth Redirect URI

#### Client OAuth Settings

Yes

##### Client OAuth Login

Enables the standard OAuth client token flow. Secure your application and prevent abuse by locking down which token redirect URIs are allowed with the options below. Disable globally if not used. [?]

Yes

##### Web OAuth Login

Enables web-based Client OAuth Login. [?]

Yes

##### Enforce HTTPS

Enforce the use of HTTPS for Redirect URIs and the JavaScript SDK. Strongly recommended. [?]

No

##### Force Web OAuth Reauthentication

When on, prompts people to enter their Facebook password in order to log in on the web. [?]

No

##### Embedded Browser OAuth Login

Enable webview Redirect URIs for Client OAuth Login. [?]

Yes

##### Use Strict Mode for Redirect URIs

Only allow redirects that exactly match the Valid OAuth Redirect URIs. Strongly recommended. [?]

#### Valid OAuth Redirect URIs

A manually specified redirect\_uri used with Login on the web must exactly match one of the URIs listed here. This list is also used by the JavaScript SDK for in-app browsers that suppress popups. [?]

Valid OAuth redirect URIs.

### ❗ 正确设置授权的 REDIRECT URL

在 Facebook OAuth 配置中，有效的 OAuth Redirect URIs 必须是您的 Casdoor 回调 url, and the Redirect URL in Casdoor should be your application callback url

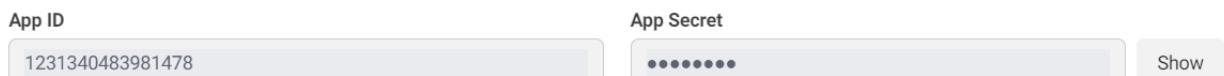
更多详情请阅读 [应用配置](#)

基本应用程序配置即将完成！

在仪表板顶部栏中切换模式从 **In development** 到 **Live**



然后您的 **App ID** 和 **App secrets** 可以在 Casdoor 中使用。



添加 Facebook OAuth 提供商并在您的 Casdoor 填写 **Client ID** 和 **Client Secrets** 使用 **App ID** 和 **App Secrets**。

A screenshot of the Casdoor OAuth provider configuration form for Facebook. The form includes fields for "名称" (Name) set to "my\_facebook\_provider", "显示名称" (Display Name) set to "Facebook provider", "分类" (Category) set to "OAuth", "类型" (Type) set to "Facebook", "Client ID" set to "1231340483981478", and "Client secret" set to "\*\*\*\*\*". A "修改提供商" (Edit Provider) button and a large blue "保存" (Save) button are visible at the top left of the form.

现在你可以使用 Facebook 作为第三方服务来完成身份验证！

# 谷歌

若要设置 Google OAuth 提供商, 请前往 [Google API 控制台](#) 并登录使用您的 Google 帐户。

Project name \*

My Casdoor



Project ID: my-casdoor. It cannot be changed later. [EDIT](#)

Location \*

 No organization

[BROWSE](#)

Parent organization or folder

[CREATE](#)

[CANCEL](#)

然后导航到 **OAuth 同意屏幕** 标签来配置 OAuth 同意屏幕。

**API** APIs & Services

## OAuth consent screen

 Dashboard

Choose how you want to configure and register your app, including your target users. You can only associate one app with your project.

 Library Credentials OAuth consent screen Domain verification Page usage agreements

## User Type

 Internal 

Only available to users within your organization. You will not need to submit your app for verification. [Learn more about user type](#)

 External 

Available to any test user with a Google Account. Your app will start in testing mode and will only be available to users you add to the list of test users. Once your app is ready to push to production, you may need to verify your app. [Learn more about user type](#)

**CREATE**

并注册您的 Google 应用。

## Edit app registration

1 OAuth consent screen — 2 Scopes — 3 Test users — 4 Summary

### App information

This shows in the consent screen, and helps end users know who you are and contact you

App name \*

The name of the app asking for consent

User support email \*

For users to contact you with questions about their consent

App logo

BROWSE

Upload an image, not larger than 1MB on the consent screen that will help users recognize your app. Allowed image formats are JPG, PNG, and BMP. Logos should be square and 120px by 120px for the best results.

### App domain

To protect you and your users, Google only allows apps using OAuth to use Authorized Domains. The following information will be shown to your users on the consent screen.

Application home page

然后导航到 Credential 选项卡。

## Credentials

[+ CREATE CREDENTIALS](#) [DELETE](#)

Create credentials to access your enabled APIs. [Learn more](#)

### API Keys

<input type="checkbox"/>	Name	Creation date
No API keys to display		

### OAuth 2.0 Client IDs

<input type="checkbox"/>	Name	Creation date
No OAuth clients to display		

### Service Accounts

<input type="checkbox"/>	Email	Name
No service accounts to display		

并为您的应用创建凭据:

[Create OAuth client ID](#)

A client ID is used to identify a single app to Google's OAuth servers. If your app runs on multiple platforms, each will need its own client ID. See [Setting up OAuth 2.0](#) for more information. [Learn more](#) about OAuth client types.

Application type \*



### ① 正确设置授权重定向 URI

在 Google OAuth 配置中，`应用回调URL` 必须是 您的Casdoor的回调URL并且 Casdoor 中的 `Redirect URL` 应该为 您的应用回调地址

更多详情请阅读 [应用配置](#)

创建Client ID并获取 `client ID` 和 `Client Secrets` 后

## OAuth client created

The client ID and secret can always be accessed from Credentials in APIs & Services



OAuth access is restricted to the [test users](#) listed on your [OAuth consent screen](#)

Your Client ID

487708653175-11oih9gfqb2u3tvfp6684qaes5ujjdca.apps.googleusercontent.com



Your Client Secret

HbxoqxxkGSs1lCVRuMTVvK57



DOWNLOAD JSON

OK

添加 Google OAuth 提供程序并在您的 Casdoor 中填写 `client ID` 和 `Client`

Secret

Edit Provider

Save

Name [?](#) :

my\_google\_provider

Display name [?](#) :

Google provider

Category [?](#) :

OAuth

Type [?](#) :

Google

Client ID [?](#)

487708653175-11oih9gfqb2u3tvfp6684qaes5ujjdca.apps.googleusercontent.com

Client secret [?](#)

\*\*\*\*\*

Provider URL [?](#) :

<https://console.cloud.google.com/apis/credentials/oauthclient/498643462012-46>

现在你可以使用 Google 作为第三方服务来完成身份验证！

# 百度

要设置 Baidu OAuth 提供商, 请阅读 [百度的文档](#) 并按其步骤完成 [应用程序创建](#)。

**开发者服务管理**

📍 提示:  
轻应用平台不再支持创建直达号, 如需开通直达号请登录<http://zhida.baidu.com>

**创建工作**

\* 应用名称: CasdoorTest 11/32

传统接入扩展:  合作网站

解决方案:  使用BAE

**创建**

在创建您的应用后, 重定向URL设置在以下位置:

**Casdoor**

**基本信息**

接入类型 —————

其他应用

开发者服务 ————— ✎

Oauth2.0

**安全设置**

**基本信息**

名称: Casdoor

Icon: 

ID: 25547043

API Key: Hn'...yQmAp61

在以下位置添加您的 Casdoor 域名:

Casdoor

ID API Key Secret Key

基本信息

接入类型

其他应用

开发者服务

Oauth2.0

安全设置

安全设置

Implicit Grant授权方式  启用  禁用

授权回调页:

不配置OAuth授权回调地址，会存在用户授权信息被窃取风险，强烈建议配置该项。授权回调地址的校验规则请参考：[帮助文档](#)

根域名绑定: door.casbin.com

应用服务器IP地址:

应用在访问OpenAPI时须带有Referer信息，且其域名被限制在“根域名绑定”的设置项中

限制访问OpenAPI的Referer

同时绑定访问OpenAPI服务器IP

确定 取消



## ⚠ 注意事项

这部分与百度给出的文档中的实际情况有很大的不同：

1. 将URL添加到回调URL设置中的话很可能验证URL失败，导致登录失败，所以我们要将我们的域名添加到域名设置中。
2. 只能添加一个URL或域名，这与文档有很大不同。

然后您现在就可以获得 `Client ID` 和 `Client Secrets` 了！

The screenshot shows the Casdoor interface for managing OAuth2.0 providers. On the left sidebar, there are links for '基本信息' (Basic Information), '接入类型' (Access Type), '其他应用' (Other Applications), '开发者服务' (Developer Services), 'OAuth2.0' (OAuth2.0), and '安全设置' (Security Settings). The main content area is titled '基本信息' (Basic Information) and displays the configuration for the Baidu provider. It includes fields for '名称' (Name: Casdoor), 'Icon' (Icon: Baidu Developer logo), 'ID' (ID: [redacted]), 'Client ID' (Client ID: HnhK7...QmAp61), 'API Key' (API Key: [redacted]), 'Client Secret' (Client Secret: DTgBZ...ls1bLm1Gha), 'Secret Key' (Secret Key: [redacted]), and two timestamp fields: '创建时间' (Created At: 2022-01-22 16:20:05) and '更新时间' (Updated At: 2022-01-23 15:45:06). A '重置' (Reset) button is also visible.

添加一个百度 OAuth 提供商并在您的 Casdoor 中填写 `client_id` 和 `client_secrets`。

The screenshot shows the Casbin provider configuration interface. The top navigation bar includes links for Home, Organizations, Users, Roles, Permissions, Providers (which is the active tab), Applications, Resources, and Tokens. The main form for editing the Baidu provider has the following fields: 'Name' (Baidu), 'Display name' (Baidu), 'Category' (OAuth), 'Type' (Baidu), 'Client ID' (HsM...nWT), and 'Client secret' (\*\*\*). The 'Client ID' and 'Client secret' fields are highlighted with red boxes. At the bottom of the form are 'Save' and 'Save & Exit' buttons.

现在你可以使用百度作为第三方服务来完成身份验证！

### ① 一般故障排除

故障排除如果您遇到百度提示您重定向URL不正确， 这里是你可能可以修复错误的一些方式：

1. 将你的域名添加到合适的位置，然后重置Secret(百度重置Secret有bug，会提示错误，但是刷新页面后Secret已经刷新)
2. 如果以上方法都不能解决问题，我们建议您删除应用程序并创建一个新的应用程序，并先设置您的域名。

另一个问题是百度返回的用户名被屏蔽了，不像它的文档显示用户名和显示的名称，所以我们目前只能使用被屏蔽的名称作为用户名。

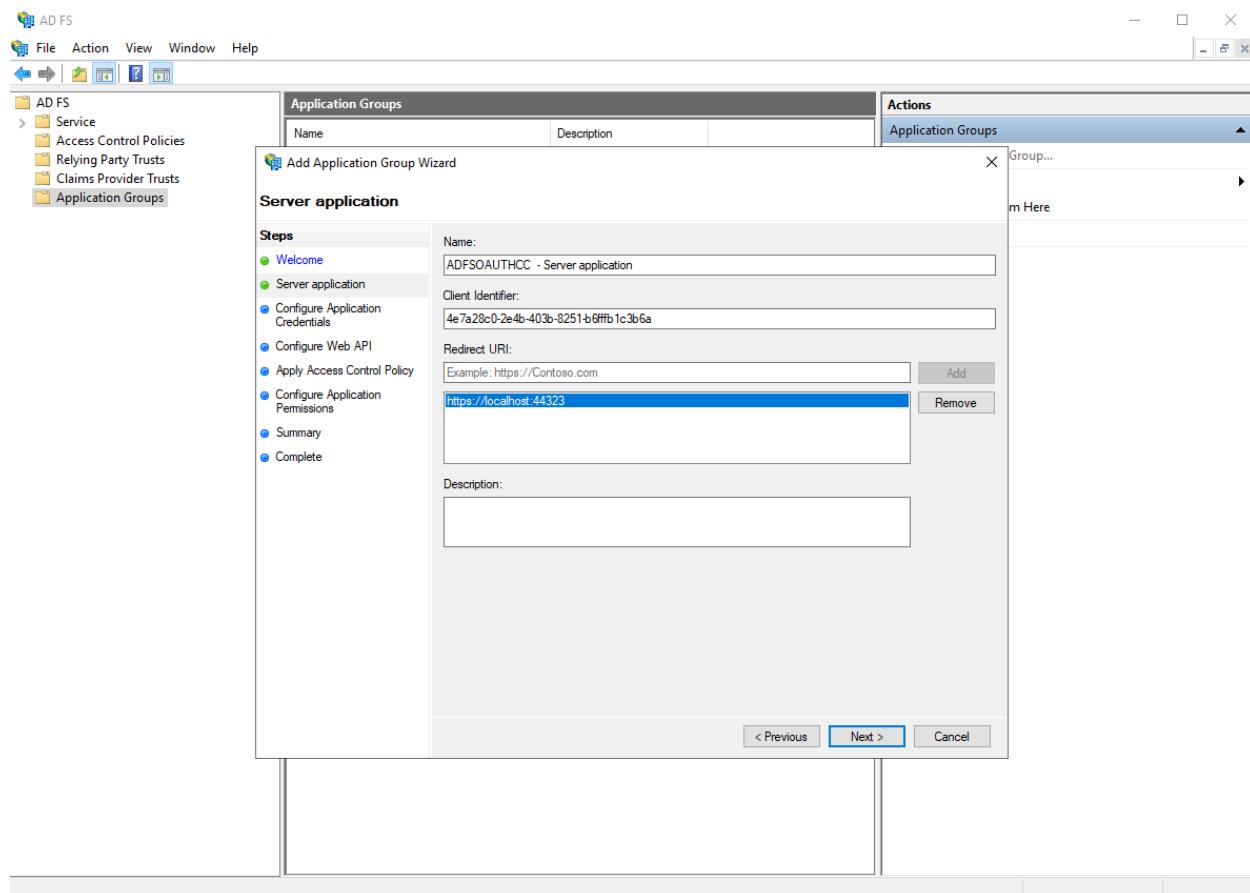
# AD FS

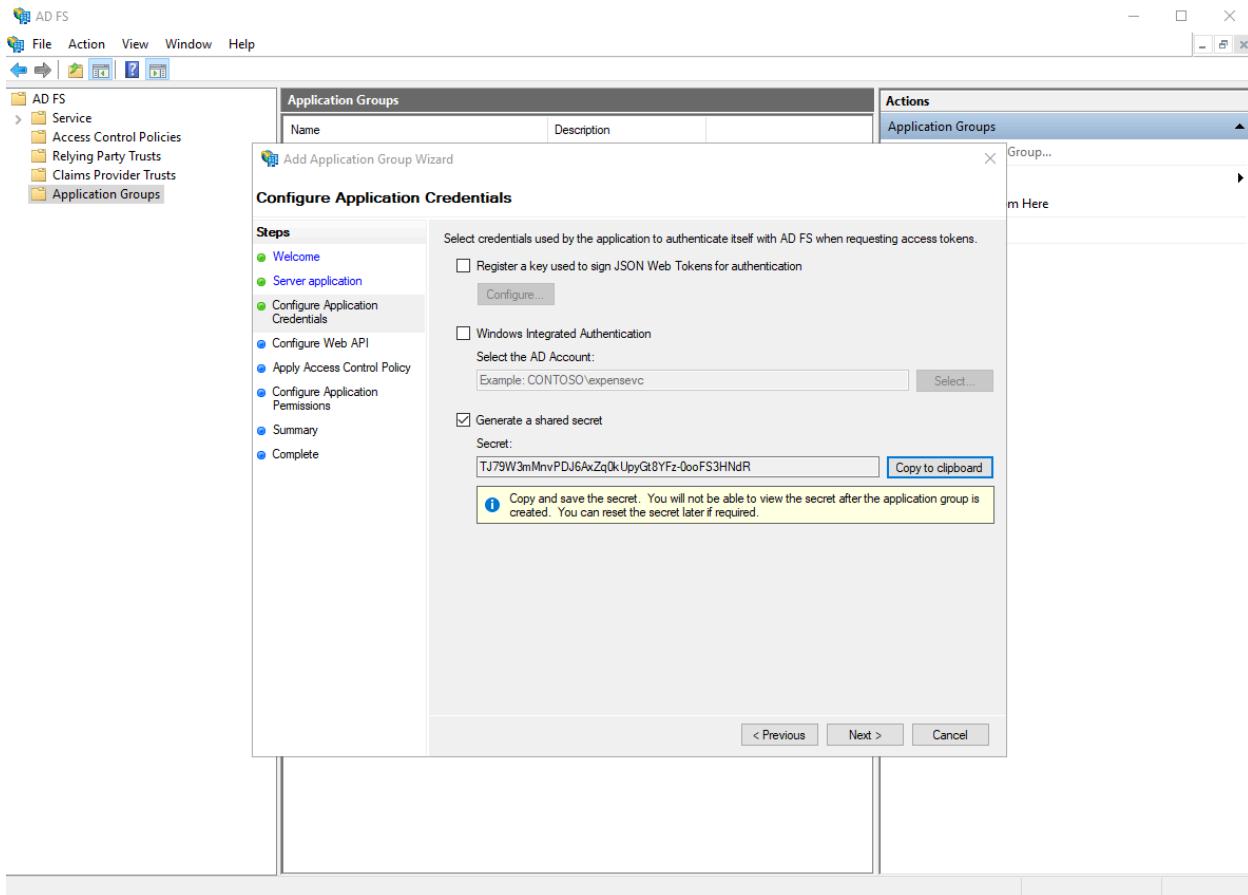
设置 Active Directory Federation Service， 请阅读 [ADFS](#) 获取关于 ADFS 的基本知识 [ADFS 部署指南](#) 以了解如何设置ADFS 服务器。 确保您有一个完全正常运行的 ADFS 服务器， 然后继续下一步。

## 步骤1 通过 ADFS 启用 OAuth

这一步骤详情请参阅 [如何通过 ADFS 启用 OAuth](#)。

当您完成此步骤时， 您应该已经获得了clientId 和clientSecret





其中，在Oauth中，第一张图片中的 Client Identifier 和第二张图片中的 Secret 应该是 clientID 和 clientSecret。

## 激活 Casdoor ADFS Provider

添加一个 Gitee OAuth 提供商并在您的 Casdoor 中填写 `Client ID` 和 `Client Secrets`。

Edit Provider [Save](#) [Save & Exit](#)

Name ②:

Display name ②:

Category ②: OAuth

Type ②: Adfs 

Client ID ②

Client secret ②

Domain ②:

Provider URL ②: <https://openhome.alipay.com/platform/appManage.htm#/app/2021003111697088/overview>

[Save](#) [Save & Exit](#)

# AzureAD

## 介绍

Azure Active Directory (Azure AD) 通过为云端和前提条件下的应用提供单一的身份系统，简化了申请管理。可将软件作为服务(SaaS) 应用软件、在前提下的应用软件和业务线路应用程序添加到Azure AD。然后用户可以一次登录来安全和无缝地访问这些应用程序。以及微软公司提供的办公室365项和其他商业应用程序。

## 如何使用？

注册应用程序的步骤如下所示。

### 步骤1. 注册一个应用程序

首先，[注册一个应用程序](#)。然后根据需要选择一个帐户类型。演示站使用下面显示的类型。

Microsoft Azure Search resources, services, and docs (G+)

[Home](#) >

## Register an application

### \* Name

The user-facing display name for this application (this can be changed later).

casdoor



### Supported account types

Who can use this application or access this API?

- Accounts in this organizational directory only (Default only - Single tenant)
- Accounts in any organizational directory (Any Azure AD directory - Multitenant)
- Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- Personal Microsoft accounts only

[Help me choose...](#)

### Redirect URI (optional)

By proceeding, you agree to the [Microsoft Platform Policies](#)

[Register](#)

## 步骤2. 创建客户端密钥

创建一个 **客户端密钥** 并保存值，这将稍后使用。

casdoor | Certificates & secrets ...

Search (Ctrl+/) Got feedback?

Manage

- [Branding & properties](#)
- [Authentication](#)
- [Certificates & secrets](#) (selected)
- [Token configuration](#)
- [API permissions](#)
- [Expose an API](#)
- [App roles](#)
- [Owners](#)
- [Roles and administrators](#)
- [Manifest](#)

Credentials enable confidential applications to identify themselves to the authentication service when receiving tokens at a web addressable location (using an HTTPS scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.

Application registration certificates, secrets and federated credentials can be found in the tabs below.

Certificates (0) Client secrets (1) Federated credentials (0)

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

[New client secret](#)

Description	Expires	Value	Secret ID
casdoor	1/8/2023	3Xr8Q~dFau2Hwyhg6y8Upb53PCFbuF... <span>Copy</span>	f3c7d37c-1def-4e29-b75f-457fa7c081e8 <span>Delete</span>

## 步骤3. 添加重定向 URI

按照图片中的示例添加casdoor的重定向uri。

The screenshot shows the 'Authentication' configuration page for an Azure AD application. The left sidebar lists several sections: Overview, Quickstart, Integration assistant, Manage (with sub-options: Branding & properties, Authentication, Certificates & secrets, Token configuration, API permissions, Expose an API, App roles, Owners, Roles and administrators, and Manifest). The 'Authentication' option is selected and highlighted with a red box. The main content area is titled 'Platform configurations' and contains a note about redirect URIs. A red box highlights the 'Add a platform' button. Below it, the 'Supported account types' section is shown, with a checked checkbox for 'Accounts in any organizational directory (Any Azure AD directory - Multitenant) accounts (e.g. Skype, Xbox)'. A warning message states: 'To change the supported accounts for an existing registration, use the manifest editor. Take properties may cause errors for personal accounts.' At the bottom are 'Save' and 'Discard' buttons, and a 'Configure' button is visible on the right.

## 步骤4. 授予管理员权限

`user.read` API 默认是打开的。 您可以根据您的需要添加更多的范围。 最后，记得 给予管理员权限。

The screenshot shows the Casdoor API permissions page. The left sidebar has sections for Overview, Quickstart, Integration assistant, Manage (with Branding & properties, Authentication, Certificates & secrets, Token configuration, and API permissions selected), Expose an API, App roles, Owners, Roles and administrators, Manifest, Support + Troubleshooting (Troubleshooting and New support request), and a troubleshooting link.

The main area displays a message: "Successfully granted admin consent for the requested permissions." Below it is a warning: "Starting November 9th, 2020 end users will no longer be able to grant consent to newly registered multitenant apps without verified publishers. [Add MPN ID to verify publisher](#)". A note states: "The 'Admin consent required' column shows the default value for an organization. However, user consent can be customized per permission, user, or app. This column may not reflect the value in your org app will be used. [Learn more](#)".

The "Configured permissions" section lists "Microsoft Graph (5)" with the following table:

API / Permissions name	Type	Description	Admin consent requ...	Status
email	Delegated	View users' email address	No	<span>Granted for Default Dire...</span> ...
offline_access	Delegated	Maintain access to data you have given it access to	No	<span>Granted for Default Dire...</span> ...
openid	Delegated	Sign users in	No	<span>Granted for Default Dire...</span> ...
profile	Delegated	View users' basic profile	No	<span>Granted for Default Dire...</span> ...
User.Read	Delegated	Sign in and read user profile	No	<span>Granted for Default Dire...</span> ...

A note at the bottom says: "To view and manage permissions and user consent, try [Enterprise applications](#)".

## 步骤5. 在casdoor中创建 AzureAD 提供商

最后一步，添加一个AzureAD OAuth提供程序，并在您的Casdoor中填写 **客户端ID** 和 **客户端秘钥**。

Edit Provider

Save

Save & Exit

Name ? : provider\_casdoor\_azuread

Display name ? : Casdoor AzureAD

Category ? : OAuth

Type ? : AzureAD

Client ID ? : 621cc0f0-055f-433f-9894-bfa1bfde169d

Client secret ? : \*\*\*

Provider URL ? : [https://portal.azure.com/#view/Microsoft\\_AAD\\_RegisteredApps/Applications列表](https://portal.azure.com/#view/Microsoft_AAD_RegisteredApps/Applications列表)

Save

Save & Exit

# Infoflow

若要设置 Infoflow OAuth 提供商, 请前往 [Infoflow](#) 并登录使用您的 Infoflow 帐户。

首先, 请访问Infoflow的 [应用中心](#)。



如流 Infoflow 首页 通讯录 应用中心 数据统计 设置

应用(5)

新建应用 应用分组/排序 应用宣传栏

并注册您的 Infoflow 应用。



返回 Casdoor 基本信息 保存 取消

应用logo: 建议使用640\*640, 2M以内的jpg、png图片

应用名称: Casdoor

应用介绍: Casdoor单点登录系统

功能:

应用 (在客户端应用面板中, 为用户提供访问内部系统的入口, [查看客户端示例](#))

机器人 (在企业群聊中, 为用户提供机器人服务, [查看客户端示例](#))

服务号 (以双人会话方式, 为用户提供交流服务, [查看客户端示例](#))

然后你现在可以获取 [AgentID](#)。

< 返回

Casdoor

| 基本信息

应用logo:



应用名称: Casdoor

应用介绍: Casdoor单点登录系统

功能: 应用

AgentID

应用ID: 55

然后导航到 **设置** 标签，并创建一个新的管理组。

如流 Infoway

首页 通讯录 应用中心 数据统计 **设置** ①

基本信息

成员加入

**权限设置** ②

通讯录设置

安全设置

客户端启动页

系统管理组

普通管理组

管理员 ③

**新建下级管理组** ④

暂未设置管理员

通讯录权限

组织架构

将您的结构添加到地址簿权限中，并赋予它以下显示的权限。同时将您刚刚创建的应用程序添加到以下位置。

## 通讯录权限

修改

组织架构

查看

管理 

对部门仅有查看权限时，只可查看被授权的成员资料信息；对部门有管理权限时，可查看成员的所有资料信息

成员ID

姓名

部门

头像

手机号

邮箱

登录帐号

## 应用权限

修改

应用权限

发消息

配置应用

Casdoor



添加敏感接口权限如下所示：

## 敏感接口权限

修改

接口名称	权限开放
获取部门成员	<input checked="" type="checkbox"/>
获取部门列表	<input type="checkbox"/>
获取成员信息	<input checked="" type="checkbox"/>
获取标签成员	<input type="checkbox"/>
维护通信录	<input type="checkbox"/>
获取成员群组列表	<input type="checkbox"/>
获取群组成员列表	<input type="checkbox"/>
维护群组成员	<input type="checkbox"/>
发送群组消息	<input type="checkbox"/>
维护群组话题	<input type="checkbox"/>
维护勋章	<input type="checkbox"/>
通讯录搜索	<input type="checkbox"/>

您将能够在同一页面看到 CorpID and Secret

## 开发者凭据

### Client ID

CorpID

hir...1

Secret

HgH...NB

Client Secret

重置

添加 Infoflow OAuth 提供商并填写 **客户端ID**，**客户端密钥** 和 **代理ID** 在您的 Casdoor。

Edit Provider

Save

Save & Exit

Name ② :

Infoflow

Display name ② :

Infoflow

Category ② :

OAuth

Type ② :

Infoflow

Sub type ② :

Internal

Client ID ②

CorpID

Client secret ②

\*\*\*

Secret

Agent ID ② :

55

AgentID

现在您可以使用 Infoflow 作为第三方服务来完成身份验证！

# Okta

要设置 Okta OIDC 提供商, 请首先访问 [Okta Developer](#) 并注册以获得开发人员帐户。

导航到 Applications > Applications 标签, 点击 Create App Integration, 在 Sign-in method 选择 OIDC - OpenID Connect, 和在 Application type 选择 Web Application, 并点击 Next.

### Create a new app integration

**Sign-in method**

[Learn More](#)

- OIDC - OpenID Connect**  
Token-based OAuth 2.0 authentication for Single Sign-On (SSO) through API endpoints. Recommended if you intend to build a custom app integration with the Okta Sign-In Widget.
- SAML 2.0**  
XML-based open standard for SSO. Use if the Identity Provider for your application only supports SAML.
- SWA - Secure Web Authentication**  
Okta-specific SSO method. Use if your application doesn't support OIDC or SAML.
- API Services**  
Interact with Okta APIs using the scoped OAuth 2.0 access tokens for machine-to-machine authentication.

---

**Application type**

What kind of application are you trying to integrate with Okta?

Specifying an application type customizes your experience and provides the best configuration, SDK, and sample recommendations.

- Web Application**  
Server-side applications where authentication and tokens are handled on the server (for example, Go, Java, ASP.NET, Node.js, PHP)
- Single-Page Application**  
Single-page web applications that run in the browser where the client receives tokens (for example, Javascript, Angular, React, Vue)
- Native Application**  
Desktop or mobile applications that run natively on a device and redirect users to a non-HTTP callback (for example, iOS, Android, React Native)

[Cancel](#) [Next](#)

输入 登录重定向 URI , 例如 `https://door.casdoor.com/callback`。

The screenshot shows the 'Sign-in redirect URIs' section of an Okta application configuration. It includes a checkbox for allowing wildcards in the redirect URI, a text input field containing 'https://door.casdoor.com/callback', and a blue 'Add URI' button.

Sign-in redirect URIs	<input type="checkbox"/> Allow wildcard * in sign-in URI redirect.
Okta sends the authentication response and ID token for the user's sign-in request to these URIs	<input type="text" value="https://door.casdoor.com/callback"/>
<a href="#">Learn More</a>	<a href="#">+ Add URI</a>

在 分配 部分中, 定义您的应用程序的 控制访问 的类型, 然后点击 保存 以创建应用集成。

现在您获得 `Client ID`, `Client secret` 和 `Okta 域名`。

The screenshot shows the 'Client Credentials' settings page. It displays the Client ID ('Ooa4we8u8iivyscpb5d7') and Client secret ('XXXXXXXXXXXXXX...'). Both fields have edit icons and copy/cut/paste buttons.

<b>Client Credentials</b>	
Client ID	Ooa4we8u8iivyscpb5d7
Public identifier for the client that is required for all OAuth flows.	
Client secret	XXXXXXXXXXXXXX...
Secret used by the client to exchange an authorization code for a token. This must be kept confidential! Do not include it in apps which cannot keep it secret, such as those running on a client.	

The screenshot shows the 'General Settings' page. It displays the Okta domain ('dev-53555475.okta.com'). The domain field has an edit icon and copy/cut/paste buttons.

<b>General Settings</b>	
Okta domain	dev-53555475.okta.com

在 Casdoor 控制面板中添加 Okta OAuth 提供商，输入您的 `Client ID`，`Client secret` 和 `Domain`

Edit Provider Save Save & Exit

Name ?: provider\_casdoor\_okta

Display name ?: Casdoor Okta

Category ?: OAuth

Type ?: Okta

Client ID ?: 0oa4we8u8iivyscpb5d7

Client secret ?: \*\*\*

Domain ?: <https://dev-53555475.okta.com/oauth2/default>

Provider URL ?: <https://dev-53555475.okta.com>

Save Save & Exit

### ❗ 正确设置域名

请注意，`域名` 不仅是 `Okta 域名`，`/oauth2/default` 应该附加到它。

在授权服务器上访问 [Okta 文档](#) 获取更多详情。

现在你可以使用 Github 作为第三方服务来完成身份验证！

# 电子邮箱

## 添加电子邮件提供商

1. 点击 **添加** 以添加一个新的提供者。
2. 在 **类别** 中选择 **邮件**

Name <small>?</small> :	email provider
Display name <small>?</small> :	My Email
Category <small>?</small> :	Email
Type <small>?</small> :	Default

3. 在您的smtp服务中填充 **Username**, **Password**, **Host**, **Port**。

Username <small>?</small>	no-reply@casbin.com
Password <small>?</small>	***
Host <small>?</small> :	smtp.qiye.aliyun.com
Port <small>?</small> :	465

4. 填写自定义 **Email Title** and **Email Content** 并保存。



# 短信

我们使用 [casdoor/go-sms-sender](#) 发送短信给Casdoor。现在，[go-sms-sender](#) 支持阿里云、腾讯云和Volc SMS API。如果您想要支持其他短信提供商，您可以提出一个 issue，或提出一个pull request。

## 添加短信提供商

1. 点击 [添加](#) 以添加一个新的提供者。
2. 在 [类别](#) 中选择 [SMS](#)

The screenshot shows the 'Edit Provider' interface. At the top, there are two buttons: 'Edit Provider' and 'Save'. Below these are three input fields: 'Name' (set to 'SMS Provider'), 'Display name' (set to 'My SMS'), and 'Category' (set to 'SMS'). Under the 'Category' dropdown, there are other options: 'OAuth', 'Email', 'SMS' (which is highlighted in blue), and 'Storage'. A note at the bottom left says 'Client secret' with a question mark icon.

3. 选择您的提供商类型 ([阿里云短信](#), [腾讯云短信](#) 或 [Volc Engine SMS](#))

The screenshot shows a dropdown menu for 'Type'. The first item, 'Aliyun SMS', is highlighted in blue, indicating it is selected. Other items in the list are 'Tencent Cloud SMS' and 'Volc Engine SMS'.

4. 从阿里云、腾讯云或 Volc Engine 获取您的信息并填写。

# 示例

## ① 备注

这里我使用阿里云短信服务作为示例

登录我的阿里云工作台后，点击AccessKey 来创建 ID 和密钥。

The screenshot shows the Alibaba Cloud Workbench interface. The top navigation bar includes links for Workbench, Search, Costs, Work Orders, ICP Filing, Enterprise, Support, Apps, and Help. A user profile icon is also present. Below the navigation is a banner for a survey about the SMS service's ease of use. The main content area is titled "SMS Service Overview". It displays a summary of sending volume data and a progress bar indicating 20% completion. A button labeled "Quick Start SMS Service" is visible. On the right side, there are sections for developer guides, monitoring, and various SMS management tasks like adding signers and templates. A specific tab labeled "AccessKey" is highlighted with a red circle.

通过创建AccessKey ID和AccessKey 密钥，我可以获得我的AccessKey ID和访问密钥：

The screenshot shows the "Security Information Management" section of the Alibaba Cloud Workbench. A yellow warning bar at the top提醒用户AccessKey ID和AccessKey Secret是访问阿里云API的密钥，具有该账户完全的权限，请您妥善保管。The main table lists a single AccessKey entry:

AccessKey ID	AccessKey Secret	状态	最后使用时间	创建时间	操作
LTAI4Fy4mVoMjAzC95mt5Wn7	显示	启用	2020年7月19日 20:24:58	2020年7月11日 17:52:50	禁用   删除

填充 AccessKey ID and AccessKey 到 Cassdoor Client ID 和 Client Secret。

# 存储

## 存储

设置上传文件的 Casdoor 存储提供商

## Azure Blob

使用 Azure Blob 作为Casdoor的存储提供商

# 存储

如果您需要使用文件存储服务，例如 `头像上传`，您需要设置存储提供商，并在您的 `应用` 中应用它。

Casdoor 支持两种类型的存储，`本地` 和 `云`。在本章中，您将学习如何添加一个存储提供商来使用这些服务。

## 本地

使用 `本地` 类型，`Client ID`，`Client secret`，`Endpoint` 和 `Bucket` 已不再需要，您可以 `随意填写`，并且它 `不能为空`。

您唯一需要配置的项目是 `Domain` 字段。请遵循格式：

Domain/images

例如，`http://127.0.0.1:7001/images`, `http://door.casbin.org/images` 都是允许的。

但是 `127.0.0.1:7001/images` 是错误的。

## 云端

目前我们支持 `AWS S3` and `阿里云 OSS` 云供应商，并且正在添加更多云存储服务。

用 `Client ID`, `Client secret`, 填写相应字段 `Endpoint` 和 `Bucket` 从您的云端供

应商控制台获得。

:::tip

使用 **非区域** 类型，您可能不需要用于自定义域的 **Domain** 字段。

:::

## 示例：

### ① 备注

这里我使用 **阿里云 OSS** 作为示例

AccessKey 是您访问阿里云 API 的密钥，拥有完全的帐户权限。

所以 在阿里云工作台中创建了 AccessKey

然后创建 OSS 服务：



The screenshot shows the 'Create Bucket' dialog box. At the top, there's a note: '注意: Bucket 创建成功后, 您所选择的 存储类型、区域、存储冗余类型 不支持变更。' (Note: After bucket creation, the selected storage type, region, and redundancy type cannot be changed). Below this, there are input fields: 'Bucket 名称' (Bucket Name) set to 'mycasdoor', which has a character limit of 9/63; '地域' (Region) set to '华北2 (北京)' (North China 2 (Beijing)); and 'Endpoint' set to 'oss-cn-beijing.aliyuncs.com'. A note below the region field says: '相同区域内的产品内网可以互通; 订购后不支持更换区域, 请谨慎选择。' (Products within the same region can communicate via internal network; changing region after purchase is not supported, please choose carefully).

在 Casdoor 填写必要的信息并保存：

Name <a href="#">?</a> :	provider_storage_aliyun_oss
Display name <a href="#">?</a> :	Storage Aliyun OSS
Category <a href="#">?</a> :	Storage
Type <a href="#">?</a> :	Aliyun OSS
Client ID <a href="#">?</a>	LTAIxFoNpNAnPoiT
Client secret <a href="#">?</a>	***
Endpoint <a href="#">?</a> :	oss-cn-beijing.aliyuncs.com
Endpoint (Intranet) <a href="#">?</a> :	oss-cn-beijing-internal.aliyuncs.com
Bucket <a href="#">?</a> :	casbin
Domain <a href="#">?</a> :	<a href="https://cdn.casbin.com/casdoor/">https://cdn.casbin.com/casdoor/</a>
Provider URL <a href="#">?</a> :	<a href="https://oss.console.aliyun.com/bucket/oss-cn-beijing/casbin/object">https://oss.console.aliyun.com/bucket/oss-cn-beijing/casbin/object</a>

然后您可以在应用程序中使用阿里云云存储服务。

# Azure Blob

① 备注

这是 Azure Blob 的一个示例

- 您必须拥有一个 [Azure storage](#) 帐户。

## 步骤1. 选择 Azure Blob

选择 Azure Blob 作为存储类型。

Edit Provider		Save	Save & Exit
Name ② :	provider_ftfzes		
Display name ② :	New Provider - ftfzes		
Category ② :	Storage		
Type ② :	Azure Blob		
Client ID ②	Local File System		
	AWS S3		
Client secret ②	Aliyun OSS		
	Tencent Cloud COS		
Endpoint ② :	Azure Blob		

## 步骤2. 填写 Casdoor 的必要信息

有三个必填字段。 `Client ID`, `Client secret`, `Bucket`. Azure Blob账户的对应关系如下：

名称	Azure 中的名称	是否必填项
Client ID	AccountName	必填
Client secret	AccountKey	必填
Bucket	ContainerName	必填
Domain	DomainName	

- 帐号名称

`AccountName` 是您的帐户名称。

- 账户密钥

`AccountKey` 是您访问 Azure API 的密钥。

**ⓘ 备注**

您可以在 Azure Portal 中，在您的存储帐户左手窗格的“访问密钥”部分下获取您的帐户密钥。

The screenshot shows the Azure Storage account 'casbin' overview page. The left sidebar contains navigation links for Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, and Storage browser (preview). Under Data storage, there are links for Containers, File shares, Queues, Tables, and Shared access signature. Under Security + networking, there are links for Networking, Azure CDN, and Access keys. The right side shows the 'Essentials' and 'Properties' tabs. The 'Essentials' tab displays resource group (move) as default, location as East Asia, subscription as Visual Studio Enterprise, subscription ID (redacted), disk state as Available, and tags as Click here to add tags. The 'Properties' tab is selected and shows Blob service settings: Hierarchical namespace (Disabled), Default access tier (Hot), Blob public access (Enabled), Blob soft delete (Enabled (7 days)), Container soft delete (Enabled (7 days)), Versioning (Disabled), Change feed (Disabled), NFS v3 (Disabled), and Allow cross-tenant replication (Enabled). It also shows File service settings: Large file share (Disabled) and Active Directory (Not configured). The date 2022-04-2 is visible in the bottom right corner.

- 容器名称

您首先需要创建一个容器。也可以点default选择一个默认容器。

Home > casbin

## casbin | Containers

Storage account

Search (Ctrl+)

+ Container Change access level Restore containers Refresh Delete

Overview Activity log Tags Diagnose and solve problems Access Control (IAM) Data migration Events Storage browser (preview)

Containers

Name

- \$logs
- default

2022-04-29 2022-04-29 2022-04-27

- 域名

您Azure CDN中的自定义域名。

fd-profile

Front Door and CDN profiles

Search (Ctrl+)

Purge cache Origin response timeout Delete Refresh

Overview Activity log Access control (IAM) Tags Diagnose and solve problems

Front Door manager Domains Origin groups Rule set Security policies Optimizations Secrets Properties Locks

Analytics Reports Monitoring

Essentials

Resource group (move)	: default
Status	: Active
Location	: Global
Subscription (move)	: Visual Studio Enterprise
Subscription ID	: f2f8e27a-12ff-4412-9d21-bc583a034f8b
Tags (edit)	: Click here to add tags

Properties Monitoring

Endpoints

Endpoint hostname	endpoint-hth4eebgcdz2ex.z01.azurefd.net
	Provision succeeded
	Enabled

Security policy

Custom domains

Domain name	cdn.casploy.com
	Provision succeeded
	Validation approved

Routes

Route name	default-route
(endpoint)	hth4eebgcdz2ex.z01.azurefd.net
	Provision succeeded
	Enabled

2022-04-27

## 步骤3. 保存您的配置

最终结果如下：

Name ② :	Provider_azure
Display name ② :	Provider_azure
Category ② :	Storage
Type ② :	Azure Blob
Client ID ②	casbin
Client secret ②	***
Endpoint ② :	
Endpoint (Intranet) ② :	
Bucket ② :	default
Domain ② :	<a href="https://cdn.casploy.com/">https://cdn.casploy.com/</a>
Provider URL ② :	<a href="https://github.com/organizations/xxx/settings/applications/1234567">https://github.com/organizations/xxx/settings/applications/1234567</a>

然后您可以在应用程序中使用阿里云云存储服务。



&gt; 提供商

&gt; SAML

# SAML

## 概述

使用来自支持SAML 2.0 的外部身份提供商的身份

## Aliyun IDaaS

使用 Aliyun IDaaS 验证用户

## Keycloak

使用 Keycloak 验证用户

# 概述

可以配置Casdoor来支持用户使用外部身份提供者的身份登录到界面，这些身份提供者支持 SAML 2.0。在这种配置中，Casdoor 不能为用户存储任何登录凭证。

现在，Casdoor 支持许多SAML应用程序提供者。供应商的图标将在添加到Casdoor后显示在登录页面中。以下是 Casdoor 支持的提供商：

阿里云 IDaaS	Keycloak
	
	

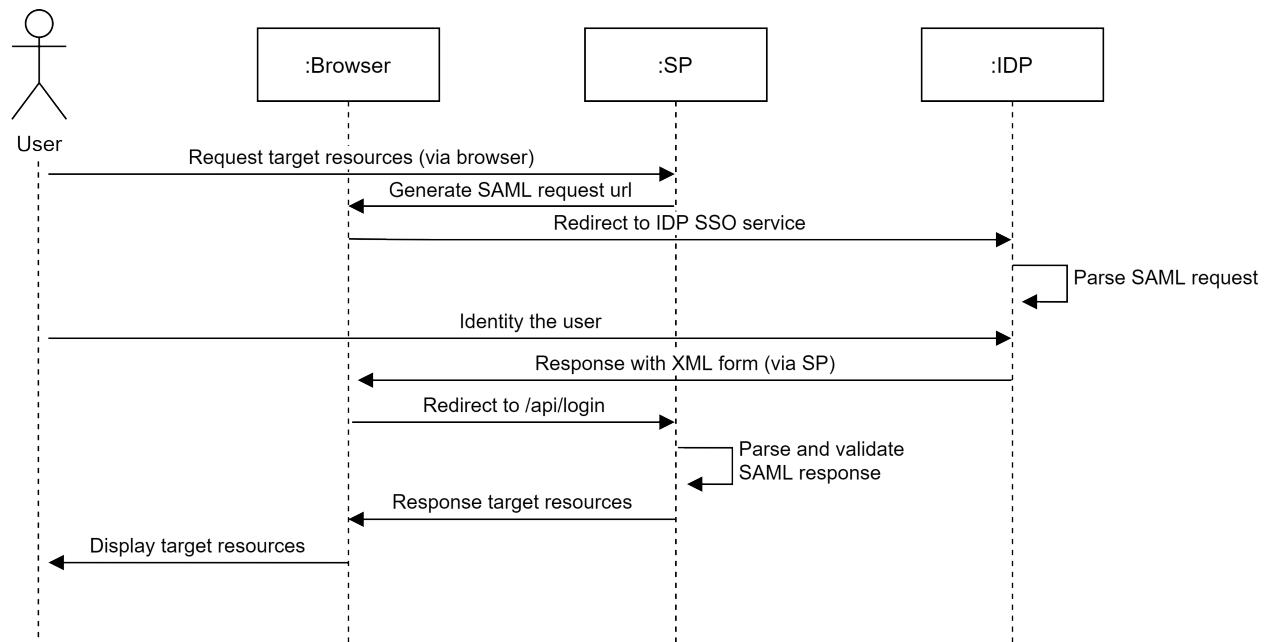
## 条款

- 身份提供商 (IDP) —— 储存身份数据库并向Casdoor提供身份和认证服务的服务。
- 服务提供商(SP) - 为终端用户提供资源的服务，在这种情况下，就是Casdoor 部署。
- 申述消费者服务——身份提供者提出的SAML断言的消费者。

## SAML 集成工作方式

当使用SAML SSO时，用户通过身份提供者登录到Casdoor，而没有向Casdoor传递凭

证。进展情况见下图表。



# Aliyun IDaaS

## 在 Aliyun IDaaS 中创建 SAML 应用程序

登录到 [Aliyun 管理控制台](#), 搜索并前往应用程序标识服务 (IDaaS)。

The screenshot shows the Aliyun Management Console interface. The top navigation bar includes links for Home, Workbench, and various services like SAML, ICP, and App. The main content area is titled "概览页" (Overview Page) under "应用身份管理" (Application Identity Management). It features a section titled "IDaaS (Identity-as-a-Service) 是为企业客户提供的身份访问管理服务, IDaaS支持 EIAM 和 CIAM" (IDaaS is a service provided for enterprise customers for identity access management, supporting EIAM and CIAM). Below this, there's a comparison table for "EIAM 不同版本区别" (Difference between EIAM versions) comparing Standard Edition and Special Edition across various features like single sign-on, user quota, and integration with external systems. To the right, there's a sidebar for "阿里云售后" (Alibaba Cloud After-sales) with a QR code for reporting issues and a "帮助文档" (Help Documentation) section with links to various EIAM-related guides.

点击 EIAM 实例列表 并打开免费版本。

The screenshot shows the "EIAM 实例列表" (EIAM Instance List) page. The left sidebar has a red box around the "EIAM 实例列表" link under "应用身份管理". The main table lists EIAM instances with columns for "实例ID/名称" (Instance ID/Name), "标准版实例ID" (Standard Edition Instance ID), "状态 (全部)" (Status (All)), "规格授权" (Specification Authorization), "最大用户数" (Max Users), "到期时间" (Expiration Time), "产品版本" (Product Version), "用户登录页地址" (User Login Page Address), and "实例开放接口域名" (Instance Open API Domain Name). A blue button labeled "开通免费版" (Enable Free Edition) is located in the top right corner of the table area. The bottom of the page shows pagination controls.

打开后将自动创建并运行一个实例。点击实例名称或 管理 按钮来输入 IDaaS 管理控制台。

The screenshot shows the EIAM Instance List page. On the left, there's a sidebar with '应用身份管理' (Application Identity Management) and 'EIAM 实例列表' (EIAM Instance List). The main area displays a table with one instance: 'idaas-cn-shanghai' (Status: Running, Free Tier, 100 users, V1.9.6-GA). A red box highlights the '管理' (Manage) button next to the instance name. Below the table are navigation buttons: '< 上一页' (Previous Page), '1' (Page 1), and '下一页 >' (Next Page).

输入了 IDaaS 管理控制台后，点击 添加应用程序，搜索 SAML，然后点击 添加应用程序

The screenshot shows the 'Add Application' page under '应用' (Application) in the sidebar. The 'SAML' search term is entered in the search bar. The results table lists several SAML-related applications, each with a '添加应用' (Add Application) button. A red box highlights the '添加应用' button for the 'SAML' entry.

应用图标	应用名称	应用ID	标签	描述	应用类型	操作
S	云安全访问服务SASE	plugin_saml_cas	SASE, SD-WAN安全, ZTNA, SAML	云安全访问服务SASE (Cloud Access Service Edge) 是阿里云首个一站式SaaS办公云安全管控平台，企业无需再投资复杂的堡垒机设备，即可快速构建云上零信任内网访问、上网行为稽查与审计、数据防泄漏、终端全盘加密、SD-WAN安全、办公访问限速等在内的办公云安全体系。兼容标准的SAML协议，可无缝兼容阿里云IDaaS的身份体系，用户可根据在IDaaS中配置的员工身份与组织架构，在云安全访问服务SASE产品中设置安全策略。	Web应用	<a href="#">添加应用</a>
G	阿里云RAM-用户SSO	plugin_aliyun	SSO, SAML, 阿里云	基于 SAML 协议，实现由 IDaaS 单点登录到阿里云控制台；使用该模板，需要在 RAM 中为每个用户单独创建 RAM 子账户，IDaaS 账户和 RAM 子账户通过映射实现单点登录到 RAM。	Web应用	<a href="#">添加应用</a>
G	阿里云RAM-角色SSO	plugin_aliyun_role	SSO, SAML, Aliyun	基于 SAML 协议，实现由 IDaaS 单点登录到阿里云控制台；使用该模板，需要 RAM 中创建 RAM 角色，不需要为每个用户单独创建 RAM 子账户，IDaaS 账户和 RAM 角色通过映射实现单点登录到 RAM。	Web应用	<a href="#">添加应用</a>
M	阿里邮箱	plugin_aimail	SSO, 用户同步, SAML, 阿里云, 邮箱	基于 SAML 协议，实现由 IDaaS 到阿里邮箱的单点登录和用户同步。	Web应用	<a href="#">添加应用</a>
W	WordPressSaml	plugin_wordpress_saml	SSO, SAML, CMS	WordPress 是全世界最流行的泛应用的 CMS (Content Management System, 内容管理系统)，它通过非常强大的插件系统和开放的自然的操作界面，允许千万技术爱好者生产、管理各种类型的网站。从商业网站、政府网站到个人博客、主题论坛，WordPress 所支持的形式非常多样。IDaaS 支持通过 SAML 协议单点登录到 WordPress 网站。	Web应用	<a href="#">添加应用</a>
S	SAML	plugin_saml	SSO, SAML	SAML (Security Assertion Markup Language, 安全断言标记语言，版本 2.0) 基于 XML 协议，使用包含断言 (Assertion) 的安全令牌，在授权 (IDaaS) 和消费者 (应用) 之间传递身份信息，实现基于网络安全的单点登录。SAML 协议是成熟的身份协议，在国内外的公司和机构中有非常广泛的运用。	Web应用	<a href="#">添加应用</a>
G	GitLab	plugin_saml_gitlab	SSO, SAML	GitLab 是一个用于仓库管理系统的开源项目，使用 Git 作为代码管理工具，并在此基础上搭建起来的 web 服务器。	Web应用	<a href="#">添加应用</a>

点击 添加签名密钥。

## 添加应用 (SAML)

×

导入SigningKey	添加SigningKey				
别名	序列号	有效期	秘钥算法	算法长度	操作
暂无数据					

填写所有必需的信息并提交。

## 添加SigningKey

×

* 名称	CASDOOR-TEST
部门名称	请输入部门名称
公司名称	请输入公司名称
* 国家	CN
* 省份	Beijing
城市	请输入城市
* 证书长度	1024
* 有效期	3 年
<button>提交</button>	<button>取消</button>

选择添加的签名密钥。

## 添加应用 (SAML)

×

		导入SigningKey	添加SigningKey			
别名	序列号	有效期	秘钥算法	算法长度	操作	
CN=CASDOOR-TEST, ST=Beijing, C=CN	3322747020095790430	1095	RSA	1024	<button>选择</button> <button>导出</button>	

填写下面所需的所有信息并保存。

- IDP 身份ID：保持与签发者地址在 Casdoor 中相同。
- SP 实体 ID & SP ACS URL (SSO 定位)：现在填写您想要的任何东西。完成 Casdoor 配置后，您需要修改。
- 描述属性：直接填充为用户名。
- 账户关联模式：账户协会

## 添加应用 (SAML)

X

图片大小不超过1MB

应用ID

idaas-cn-shanghai-pvl0hq0ojppugin\_saml

\* 应用名称

CASDOOR-SAML

\* IDP IdentityId

CASDOOR

IDP IdentityId is required

\* SP Entity ID

http://localhost

SP Entity ID is required

\* SP ACS URL(SSO Location)

http://localhost

\* NameIdFormat

urn:oasis:names:tc:SAML:2.0:nameid-format:transient

v

\* Binding

POST

v

SP 登出地址

请输入 SP 登出地址

Assertion Attribute

username

应用子账户

v

-

+

断言属性。设值后，会将值放入SAML断言中。名称为自定义名称，值为账户的属性值。

Sign Assertion



IDaaS发起登录地址

IDaaS发起登录地址

以 http://、https:// 开头，填写后使用 IDaaS 发起登录将会跳转到该地址，而不会使用 SAML 的idp发起登录流程

\* 账户关联方式

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)

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)

提交

取消

## 帐户授权 & 关联

在SAML应用成功添加后，授权提示会高亮。现在不要授权它，添加一个帐户，然后授权它。

转到组织和组，然后点击新帐户。

机构及组

机构及组  
管理员在当前页面对组织架构、部门及其包含的组、账户进行管理，也可以使用AD、LDAP或Excel文件的方式配置导入或同步。  
在左侧的组织架构树中，可以右键点击某个部门对其进行操作，也可以左键选择某个部门，并在右侧为其进行创建账户、创建组、创建部门等操作。

组织架构

查看详情

岗位变动 导入 导出 配置 LDAP 配置钉钉同步

新增账户

账户	组	组织机构
idaas_manager		/

当前账户数 1 / 已购买套餐规格为 100

编号	账户名称	显示名称	类型	目录	操作
1	idaas_manager	默认管理员	自建账户	/	修改 账户同步 同步记录

共 1 条 < 1 > 10 条/页 跳至 1 页

填写所有必需的信息并提交。

## 新建账户

X

### 账户属性

### 扩展属性

### 父级组

父级

阿里云IDaaS

\* 账户名称

casdoor

账户名称不能以特殊字符开始，可包含大写字母、小写字母、数字、中划线(-)、下划线(\_)、点(.)，长度至少 4 位

\* 显示名称

casdoor

\* 密码

\*\*\*\*\*

密码中必须包含大小写字母+数字+特殊字符的组合;长度至少 10 位，密码不能包含"<"和">"。

邮箱

请输入有效的邮箱地址

手机号或邮箱至少填写一个。

手机号

+86 ▾ 151 123456789

手机号或邮箱至少填写一个。

外部ID

外部ID

IDaaS 平台中的唯一身份标识, 若不填将由系统自动生成。

过期时间

过期时间

不填将使用系统默认过期时间 2116-12-31

备注

备注

用户备注信息

提交

取消

转到 **应用程序授权**, 选择您想要授权的帐户, 然后点击 **保存**。

The screenshot shows the 'Application Authorization' section of the Alibaba Cloud IAM console. On the left sidebar, under the '授权' (Authorization) category, '应用授权' (Application Authorization) is selected and highlighted with a red box. In the main content area, the '应用授权主体' (Application Authorization Subject) tab is active. It displays a table of accounts with one entry: 'casdoor'. The 'casdoor' account is selected, indicated by a red box around its checkbox. A large blue '保存' (Save) button is at the bottom of the table. Navigation buttons like '< 1 >' and a page number '1' are visible at the bottom right of the table.

前往 应用程序列表, 点击 查看应用子账户, 然后点击 添加帐户关联。

The screenshot shows the 'Application List' section of the Alibaba Cloud IAM console. On the left sidebar, '应用列表' (Application List) is selected and highlighted with a red box. In the main content area, the '应用列表' (Application List) table has one entry: 'CASDOOR-SAML'. The 'casdoor' account from the previous screenshot is listed here under the '账户信息 - 子账户' (Account Information - Sub-account) column. A red box highlights the '查看应用子账户' (View Application Sub-account) link next to the account name. The table includes columns for Application Name, Application ID, Device Type, Status, Two-factor Authentication, and Operations. The 'Operations' column contains a '授权' (Authorization) button and a '详情' (Details) button, both of which are highlighted with red boxes. Navigation buttons like '< 1 >', a page number '1', and a '10条/页' (10 items per page) dropdown are at the bottom right.

The screenshot shows the Alibaba Cloud IDaaS application management interface. On the left, there's a sidebar with categories like Application Catalog, Add Application, Accounts, Institutions & Groups, Account Management, Classification Management, Authentication, Authentication Sources, RADIUS, Certificate Management, Authorization, and Application Authorization. The main area has tabs for Application Catalog and Accounts. A modal window titled 'Add Account Association' is open, containing a detailed description of what a child account is and how it relates to a primary account. Below this is a table titled 'CASDOOR-SAML' with columns for Primary Account (Account Name), Display Name, Child Account, Child Account Password, Is Associated, Approval Status, Association Time, and Operations. The table is currently empty. At the bottom right of the modal, there are buttons for 'Add Account Association' (highlighted with a red box), 'Batch Import', and 'Batch Export'.

填写需要关联的主账户和子账户，然后点击 **保存**。

主账户存在于IDaaS中，子账户是Cassdoor用户的ID。

The screenshot shows the 'Add Account Association' dialog box. It has two input fields: one for the primary account labeled 'Primary Account (Account Name)' with 'casdoor' typed in, and another for the child account labeled 'Child Account' with '52908237-fa4c-4681-b636-a6afce22fb2e' typed in. At the bottom are two buttons: a blue 'Save' button and a white 'Return' button.

## 导出 IDaaS 元数据

转到 应用程序列表, 点击 查看应用程序详细信息 然后点击 导出 IDaaS SAML Metadada

The screenshot shows the Aliyun Idaas application management interface. On the left, there is a sidebar with various management categories like Application, Account, Authentication, and Audit. The main area is titled "应用列表" (Application List) and shows a table with columns: 应用图标 (Application Icon), 应用名称 (Application Name), and 应用ID (Application ID). One row is selected, showing the icon for "CASDOOR-SAML", the name "CASDOOR-SAML", and the ID "idaas-cn-shanghai-[REDACTED]\_saml". To the right of this table, a detailed view for "应用详情 (CASDOOR-SAML)" is displayed. It includes sections for 图标 (Icon) showing a blue square with a white 'S' and the word "SAML", and for "应用ID" (Application ID) showing "idaas-cn-shanghai-[REDACTED]\_jin\_saml". Other detailed fields include 应用名称 (Application Name: CASDOOR-SAML), 应用Uuid (d9bd59093c5c031b7abbb0efa849f7bRxS506SQ3y7), SigningKey (3322747020095790430(CN=CASDOOR-TEST)), NameIdFormat (urn:oasis:names:tc:SAML:2.0:nameid-format:transient), SP ACS URL (http://localhost), and Binding (POST). A red box highlights the "IDP IdentityId" field which contains "CASDOOR [出 Idaas SAML 元配置文件 | 立即轮转密钥 | 导入]". The "SP Entity ID" field is set to "http://localhost". The "Assertion Attribute" is set to "username:APPLICATIONUSERNAME". The "SP发起地址" (SP Initiation Address) is listed as "https://dvmoykbkx.login.aliyunidaas.com/enduser/api/application/plugin\_saml/idaas-cn-shanghai-[REDACTED]\_login\_saml/sp\_sso?SAMLRequest=jxx&RelayState=yyy".

# 在 Casdoor 配置

在 Casdoor 中创建一个新的提供商。

选择分类为 **SAML**, 输入 **Aliyun Idaas**. 复制元数据内容并粘贴到 **元数据** 输入。 **端点**, 的值, **IdP** 和 **发行商URL** 将在点击 **分析** 按钮后自动生成。

Name ⓘ: casdoor-idaas

Display name ⓘ: casdoor-idaas

Category ⓘ: SAML

Type ⓘ: Aliyun IDaaS

Client ID ⓘ:

Client secret ⓘ:

Metadata ⓘ:

```
<md:IDPSSODescriptor>
<md:EntityDescriptor>
```

**Parse**

Endpoint ⓘ: https://dvmoykbkx.login.aliyunidaas.com/enduser/api/application/plugin\_saml/idaas-cn-shanghai...\_saml/sp\_sso

IdP ⓘ: MIIIBzCCAVCgAwIBAgILhzEz2NMHV4wDQYJKoZIhvCNAAQEFBQAwnjELMAkGA1UEBhMCQ04xEAOBgNVBAgTB0JlaWppbmcsFTATBgNVBAMTDENBU0RPT1tVEVTDAefw0yMTEyMDkwNzEyMTfaFw0yNDEyMDgwNzEyMTfaMDYxCzAJE

Issuer URL ⓘ: CASDOOR

SP ACS URL ⓘ: http://localhost:8000/api/acs

SP Entity ID ⓘ: http://localhost:8000/api/acs

Provider URL ⓘ: https://github.com/organizations/xxx/settings/applications/1234567

**Save**

复制 SP ACS URL 和 SP 实体 ID 并点击 保存 按钮

编辑您想要在 Cassdoor 中配置的应用程序。选择刚刚添加的提供者，然后点击按钮 保存。

Providers ⓘ:

Name	Category	Type	canSignUp	canSignIn	canUnlink	prompted	Action
casdoor-idaas	SAML						

Preview ⓘ: **Test signup page...** **Test signin page...**

## 修改 Aliyun IDaaS 的 SAML 应用程序

禁用应用程序，然后点击 修改应用程序。

The screenshot shows the Alibaba Cloud Application Management interface. On the left, there is a sidebar with various management categories like Application, Account, Authentication, Authorization, Audit, and Settings. The main area is titled '应用列表' (Application List) and displays a table of applications. One row is selected, showing detailed information such as '应用图标' (Application Icon), '应用名称' (Application Name: CASDOOR-SAML), '应用ID' (Application ID: idaas-cn-shanghai-pv0hq0ojppugin\_saml), '设备类型' (Device Type: Web应用), '应用状态' (Status: Enabled, highlighted with a red box), '二次认证状态' (Two-factor authentication status: Enabled), and '操作' (Operations). Below the table, there are sections for '应用信息' (Application Information), '认证信息' (Authentication Information), '账户信息 - 同步' (Account Information - Sync), and '账户信息 - 子账户' (Account Information - Sub-account). There are also tabs for '授权信息' (Authorization Information), '审计信息' (Audit Information), 'API' (API), and '管理应用内权限' (Manage application internal permissions). At the bottom, there are pagination controls showing '共 1 条' (1 page), a current page indicator '1', and other navigation links.

填写 SP 实体 ID and SP ACS URL (SSO location) 将内容复制到Casdoor。 提交并启用应用程序。

## 修改应用 (CASDOOR-SAML)

×

图标



上传文件

图片大小不超过1MB

应用ID

idaas-cn-shanghai-...\\login\_saml

\* 应用名称

CASDOOR-SAML

\* IDP IdentityId

CASDOOR

IDP IdentityId is required

\* SP Entity ID

http://localhost:8000/api/acs

SP Entity ID is required

\* SP ACS URL(SSO Location)

http://localhost:8000/api/acs

\* NameIdFormat

urn:oasis:names:tc:SAML:2.0:nameid-format:transient

\* Binding

POST

SP 登出地址

请输入SP 登出地址

Assertion Attribute

username

应用子账户



断言属性。设值后，会将值放入SAML断言中。名称为自定义名称，值为账户的属性值。

Sign Assertion



IDaaS发起登录地址

IDaaS发起登录地址

以 http://、https://开头，填写后使用 IDaaS 发起登录将会跳转到该地址，而不会使用 SAML 的idp发起登录流程

## 验证效果

转到您刚刚配置的应用程序，您可以在登录页面找到一个图标。

点击图标并跳转到 Aliyun IDaaS 登录页面，然后在验证后成功登录到Casdoor。



---

username, Email or phone

Password

Auto sign in      [Forgot password?](#)

[Sign In](#)

[Sign in with code](#)    No account? [sign up now](#)

A row of social media icons for GitHub, LinkedIn, and others.

---

# Keycloak

JBoss KeyCloak 系统是一个广泛使用的开源身份管理系统，它支持通过 SAML 和 OpenID 连接与应用程序集成。它还可以作为其他提供商，例如LDAP或其他SAML提供商和支持SAML或OpenID连接的应用程序之间的身份经纪人运行。

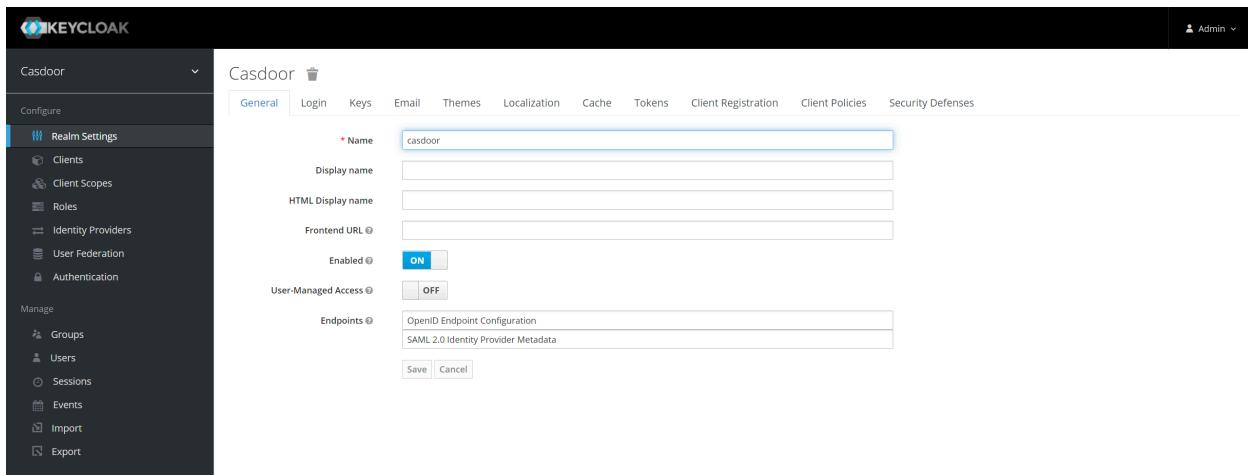
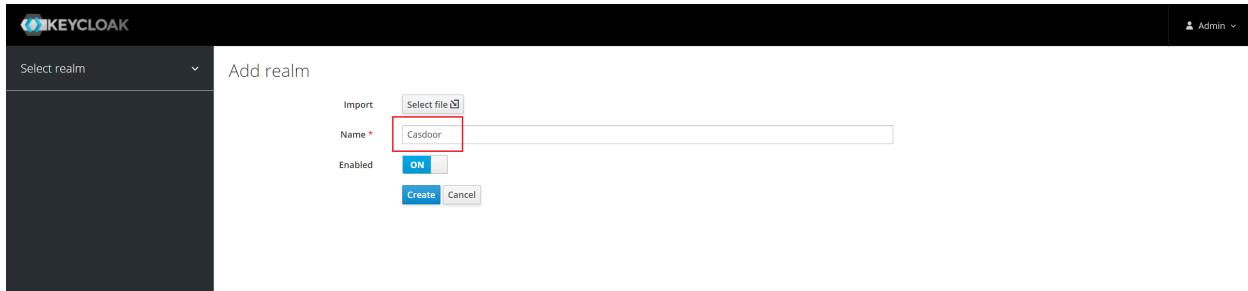
下面是如何在 Keycloak 中配置新客户端条目，并配置Castoor 来允许Keycloak 用户登录UI，这些用户通过Keycloak 配置被授予访问权限。

## Configure Keycloak

一些配置选项和假设，特别是这个示例：

- 我们假设您正在以本地开发模式运行 Casdoor。 Casoor UI可在以下网址查阅：  
`http://localhost:7001` 和服务器可在 `http://localhost:8000` 必要时用适当的URL替换。
- 让我们假设你正在本地运行Keycloak。 Keycloak UI 可在以下网址查阅：  
`http://localhost:8080/auth。`
- 在此基础上，用于此部署的SPACS URL将是： `http://localhost:8000/api/acs`。
- 我们的 SP 实体 ID 将使用相同的URL： `http://localhost:8000/api/acs`。

使用默认领域或创建一个新领域。



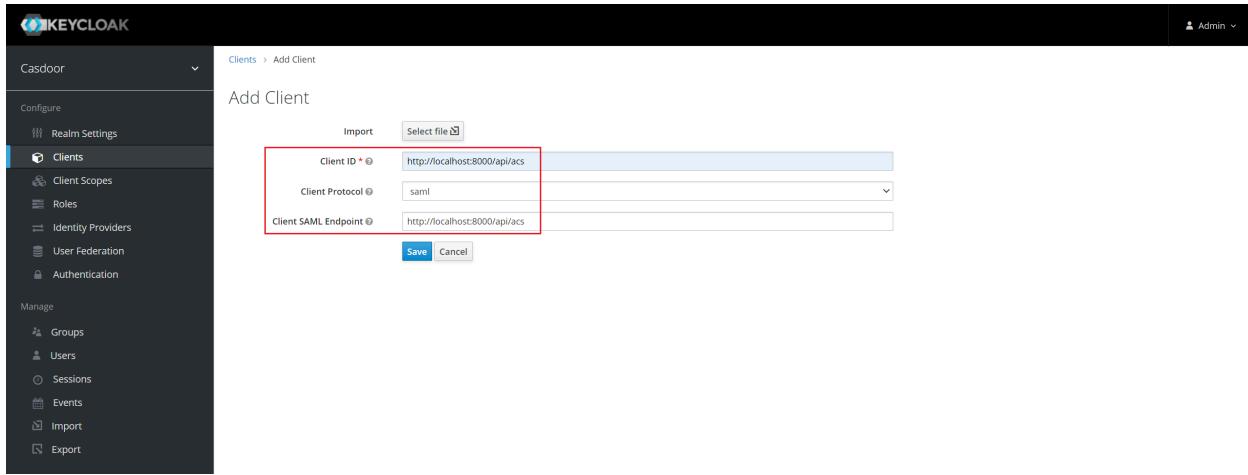
# 在 Keycloak 中添加客户端条目

## ① 信息

在 [Keycloak 文档](#) 中查看更多关于 Keycloak 客户端的详细信息。

在菜单中点击 **客户端** 然后点击 **创建** 去到 **添加客户端** 页面。 填写如下。

- **客户端 ID:** `http://localhost:8000/api/acs` - 这将是以后在 Casdoor 配置中使用的 SP 实体ID。
- **Client Protocol:** `saml`.
- **Client SAML Endpoint:** `http://localhost:8000/api/acs`. - 此 URL 是您想要 Keycloak 服务器发送SAML 请求和响应的地方。一般情况下，应用程序有一个用于处理 SAML 请求的URL。可以在客户端的设置选项卡中设置多个URL。



单击 **Save**（保存）。此动作创建客户端并将您带到 **设置** 选项卡。

以下设置列表部分：

1. **名称** - Casdoor. 这只用于在Keycloak UI中向Keycloak 用户显示友好的名称。它可以供你使用。
2. **启用** - 开启选择。
3. **包括作者声明** - 选择开启
4. **签名文档** - 选择
5. **签名声明** - 关闭。
6. **加密说明** - 关闭
7. **需要客户端签名** - 请关闭
8. **强制命名格式** - 选择开启
9. **名称 ID 格式** - 选择用户名。
10. **有效重定向 URI** - 添加 http://localhost:8000/api/acs.
11. **Master SAML 处理 URL** - http://localhost:8000/api/acs.
12. 精良的谷物SAML端点配置
  - i. **声明消费者服务公开绑定URL** - http://localhost:8000/api/acs。
  - ii. **声明消费者服务重定向绑定URL** - http://localhost:8000/api/acs。

保存该配置。

The screenshot shows the Keycloak administration interface with the following details:

- Left Sidebar:** Shows the navigation menu with "Casdoor" selected under "Clients". Other options include "Configure", "Realm Settings", "Client Scopes", "Roles", "Identity Providers", "User Federation", and "Authentication".
- Header:** Shows the URL "Clients > http://localhost:8000/api/acs" and a user "Admin".
- Main Content:** The "Settings" tab is active, showing the following configuration for the client "Casdoor":
  - Client ID:** http://localhost:8000/api/acs
  - Name:** Casdoor
  - Description:** (empty)
  - Enabled:** ON
  - Always Display in Console:** OFF
  - Consent Required:** OFF
  - Login Theme:** (dropdown menu)
  - Client Protocol:** saml
  - Include AuthnStatement:** ON
  - Include OneTimeUse Condition:** OFF
  - Force Artifact Binding:** OFF
  - Sign Documents:** ON
  - Optimize REDIRECT signing key lookup:** OFF
  - Sign Assertions:** OFF
  - Signature Algorithm:** RSA\_SHA256
  - SAML Signature Key Name:** KEY\_ID
  - Canonicalization Method:** EXCLUSIVE
  - Encrypt Assertions:** OFF
  - Client Signature Required:** OFF
  - Force POST Binding:** OFF
  - Front Channel Logout:** ON
  - Force Name ID Format:** ON
  - Name ID Format:** username
  - Root URL:** (empty)
  - Valid Redirect URLs:** http://localhost:8000/api/acs (with a minus sign and plus sign for modification)
  - Base URL:** (empty)
  - Master SAML Processing URL:** http://localhost:8000/api/acs
  - IDP Initiated SSO URL Name:** (empty)
  - IDP Initiated SSO Relay State:** (empty)
- Advanced Options:** "Fine Grain SAML Endpoint Configuration" and "Advanced Settings" sections are collapsed.
- Bottom:** "Save" and "Cancel" buttons.

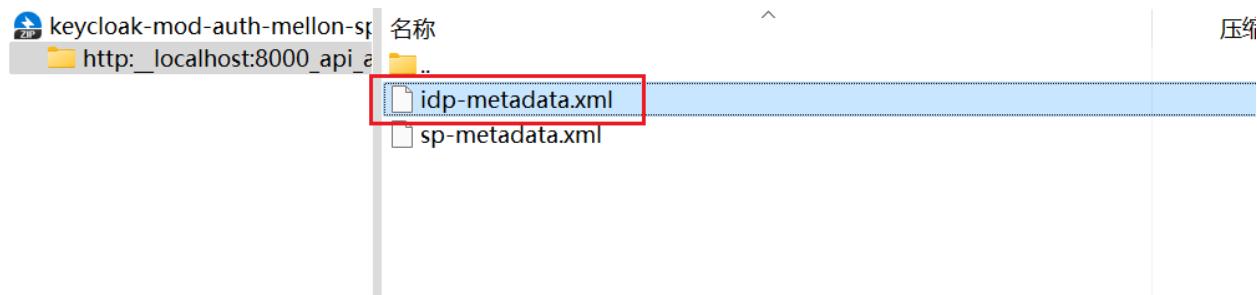
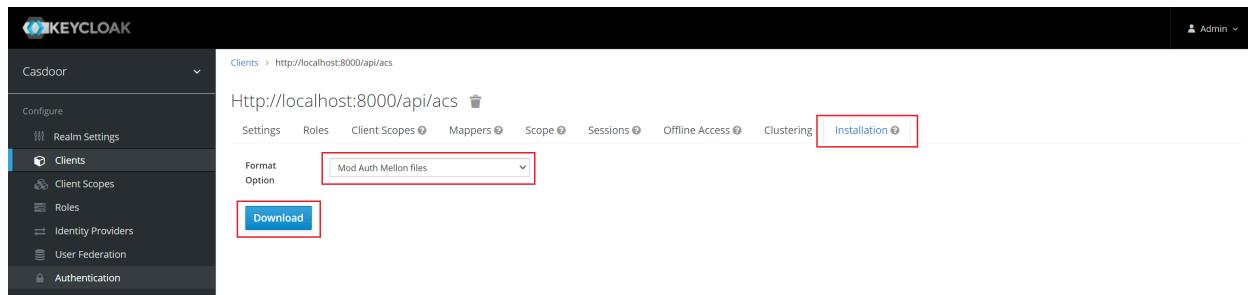
## 💡 提示

如果您想要签名authn请求，您需要启用 **客户端签名需要** 选项并上传您自己生成的证书。在Cassdoor, `token_jwt_key.key` 和 `token_jwt_key.pem` 中使用的私钥和证书位于 **对象** 目录。在Keycloak下，需要点击 **Keys** 页签，点击 **Import** 按钮，选择 **Archive Format** 为 **Certificate PEM** 并上传证书。

点击 **安装** 标签页。

对于Keycloak <= 5.0.0.0, 选择格式选项 - **SAML 元数据 IDPSODescriptor** 并复制元数据。

对于Keycloak 6.0.0+, 选择格式选项 - **Mod Mellon 文件** 并点击 **下载**。解压缩下载的 `.zip`, 定位 `idp-metadata.xml` 并复制元数据。



## 在Casdoor配置

在 Casdoor 中创建一个新的提供商。

选择分类为 SAML, 输入 Keycloak. 复制元数据内容并粘贴到 元数据 输入。 端点 (Endpoint) 的值, IdP 和 Issuer URL 将在点击 分析 按钮后自动生成。 最后点击按钮 保存。

### 💡 提示

如果您在 Keycloak 中启用 客户端签名需要 选项并上传证书, 请在 Casdoor 中启用 签名请求 选项。

Name ⓘ: keycloak-casdoor

Display name ⓘ: keycloak-casdoor

Category ⓘ: SAML

Type ⓘ: Keycloak

Client ID ⓘ:

Client secret ⓘ:

Sign request ⓘ:

Metadata ⓘ: 

```
<md:EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata" 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#" entityID="http://localhost:8080/auth/realm/casdoor"><md:IDPSSODescriptor WantAuthnRequestsSigned="true" protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol"><saml:KeyDescriptor use="signing"><ds:KeyInfo><saml:KeyName>zqpn-3t76G-na5zCz3uPD17bp-4wYtMbWMP2wqUJAHY</saml:KeyName><ds:X509Data><ds:X509Certificate>MIICnTCCAYUCBgF9pAmxSDANBqkqhkIG9w0BAQsFADASMRawDgYDVQQDDAdjYXNkb29yMB4XDThMTIxMDExMDg1OFoXDMxMTIxMDExMTAzOFowEjEQMA4GA1UEAwwHY2FzZG9vcjCCASlwDQYJKoZIhvcNAQEBBQADggEPADCCAQ:
```

[Parse](#)

Endpoint ⓘ: http://localhost:8080/auth/realm/casdoor/protocol/saml

IdP ⓘ: MIICnTCCAYUCBgF9pAmxSDANBqkqhkIG9w0BAQsFADASMRawDgYDVQQDDAdjYXNkb29yMB4XDThMTIxMDExMDg1OFoXDMxMTIxMDExMTAzOFowEjEQMA4GA1UEAwwHY2FzZG9vcjCCASlwDQYJKoZIhvcNAQEBBQADggEPADCCAQ:

Issuer URL ⓘ: http://localhost:8080/auth/realm/casdoor

SP ACS URL ⓘ: http://localhost:8000/api/acs [Copy](#)

SP Entity ID ⓘ: http://localhost:8000/api/acs [Copy](#)

Provider URL ⓘ: <https://github.com/organizations/xxx/settings/applications/1234567>

编辑您想要在 Casdoor 中配置的应用程序。 选择刚刚添加的身份提供商, 然后点击按钮 保存。

Providers ⓘ:	Providers <a href="#">Add</a>	Name	Category	Type	canSignUp	canSignIn	canUnlink	prompted	Action		
		casdoor-idaas	SAML								
		keycloak-casdoor	SAML								

# 验证效果

转到您刚刚配置的应用程序，您可以在登录页面找到Keycloak图标。

点击图标跳转到Keycloak登录页面，验证后成功登录到Casdoor。

A screenshot of a login form. At the top left is the Casdoor logo. Below it are two input fields: one for "username, Email or phone" and one for "Password". Underneath the password field is a checkbox for "Auto sign in" and a link for "Forgot password?". A large blue button in the center contains the text "Sign In". Below this button are links for "Sign in with code" and "No account? sign up now". At the bottom right of the form are two small circular icons.

# 支付

 支付宝

 > 提供商 > 支付 > **支付宝**

# 支付宝

# 验证码

## 概述

添加验证码到您的应用程序

## 默认

在您的应用程序中使用 Casdoor 默认验证码

## reCAPTCHA

添加reCAPTCHA 到您的应用程序

## hCaptcha

将 hCaptcha 添加到您的应用程序

 **Aliyun Captcha**

将Aliyun Captcha添加到您的应用程序

# 概述

Casdoor可以配置支持不同的验证码，以检查是否是人为操作。如果您添加了验证码提供商并在应用程序中应用，用户登录时，注册或忘记密码并需要发送代码，然后验证码检查对话框将会检查是否是人为操作。

现在，Casdoor支持许多验证码提供商。以下是Casdoor支持的提供商：

默认	reCAPTCHA	hCaptcha	Aliyun Captcha
			
			

我们将向您展示如何申请验证码并将其添加到casdoor。

## 添加验证码提供商

1. 导航到您的Casdoor索引页面
2. 点击顶部栏中的 [提供商](#)
3. 点击 [添加](#)，然后您可以在列表顶部看到一个新的提供商
4. 点击新的提供商修改它
5. 选择 [验证码](#) 在 [类别](#) 中
6. 在 [类型](#) 中选择您需要的验证码提供程序
7. 填写最重要的信息，不同的验证码提供者有不同的信息需要填写

# 在应用中使用

1. 单击顶部栏中的 **应用程序** 并选择一个应用程序，编辑.
2. 点击提供商添加按钮，选择您刚刚添加的提供商.
3. 完成！

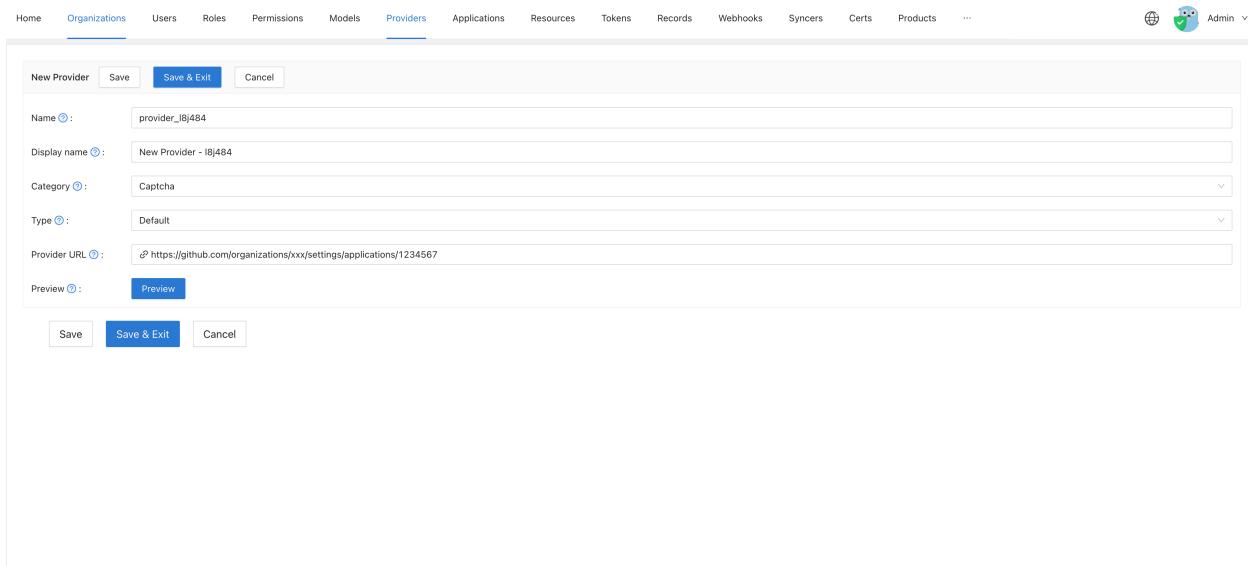
# 默认

默认验证码实现图像生成和验证。 默认的验证码图像是带有定义长度(5) 的位数0-9的顺序。

## 在 Casdoor 配置

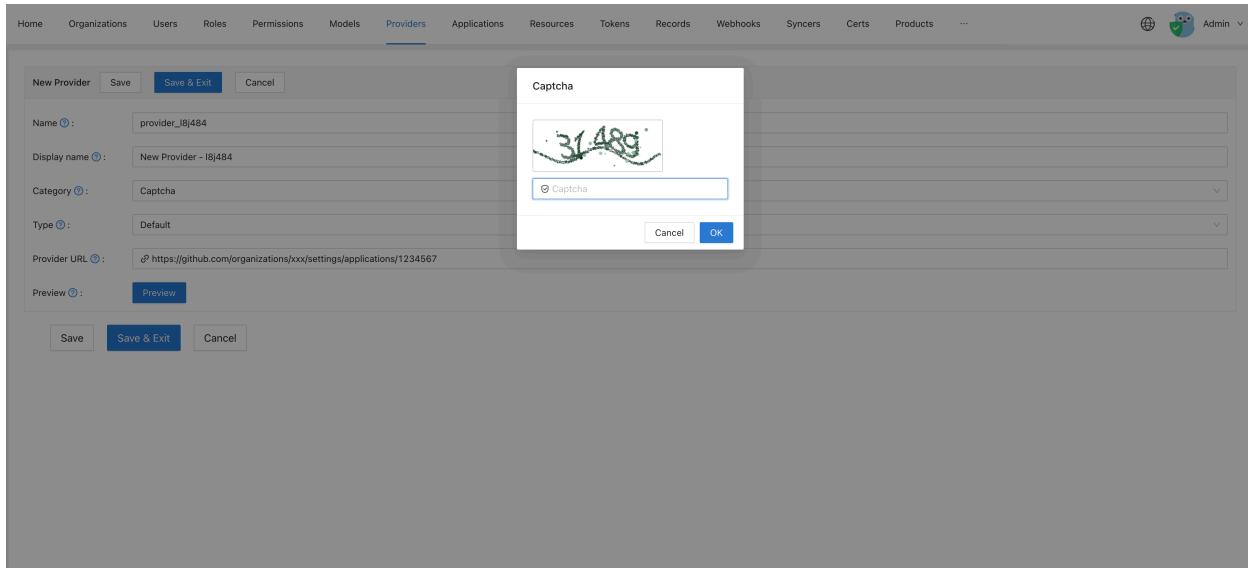
在 Casdoor 中创建一个新的提供商。

选择类别为 验证码 , 类型为 hCaptcha.



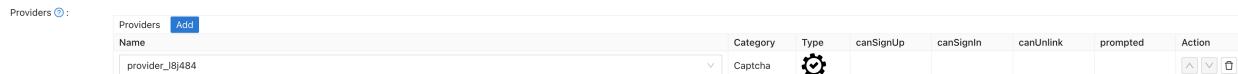
The screenshot shows the Casdoor provider configuration interface. The top navigation bar includes Home, Organizations, Users, Roles, Permissions, Models, Providers (which is selected and highlighted in blue), Applications, Resources, Tokens, Records, Webhooks, Syncers, Certs, Products, and a three-dot menu. On the far right, there is a user icon and the word 'Admin'. Below the navigation, a modal window titled 'New Provider' is open. It contains fields for Name (set to 'provider\_I8j484'), Display name (set to 'New Provider - I8j484'), Category (set to 'Captcha'), Type (set to 'Default'), and Provider URL (set to 'https://github.com/organizations/xxx/settings/applications/1234567'). There are buttons for 'Preview' (highlighted in blue), 'Save', and 'Save & Exit'. At the bottom of the modal, there are 'Save', 'Save & Exit', and 'Cancel' buttons. The background of the main interface shows other provider categories like OAuth, SAML, and OpenID Connect.

你可以点击 预览 button 来预览这个验证码的样式。



# 在应用中使用

编辑您想要在 Cassdoor 中配置的应用程序。选择刚刚添加的身份提供商，然后点击按钮 **保存**。



# reCAPTCHA

reCAPTCHA 由 Google 提供。 我们使用 reCAPTCHA v2 Checkbox。 您可以从此 [链接](#) 中看到更多详细信息。

## 创建一个 API 密钥对

要开始使用 reCAPTCHA，您需要 [注册您的站点的 API 密钥对](#)。 配对的密钥由一个站点密钥组成。 站点密钥用于在您的站点或移动应用程序上调用 reCAPTCHA 服务。 密钥授权您的应用程序后端和 reCAPTCHA 服务器之间的通信来 [验证用户的回应](#)。

首先，选择 [reCAPTCHA 的类型](#) 然后填写授权的域名或 [包名称](#)。 接受服务条款后，点击 [注册](#) 获得新的 API 密钥对。

The screenshot shows the 'Google reCAPTCHA' registration interface. At the top, there's a blue header bar with the title 'Google reCAPTCHA'. Below it, a light blue bar contains a back arrow and the text 'Register a new site'. A yellow bar below that says 'Get unlimited assessments using reCAPTCHA Enterprise'. The main form area has a white background. It starts with a 'Label' field containing 'reCaptcha'. Under 'reCAPTCHA type', 'reCAPTCHA v2' is selected, with the note 'Verify requests with a challenge'. Three options are listed: 'I'm not a robot' Checkbox (selected), 'Invisible reCAPTCHA badge', and 'reCAPTCHA Android'. Below this is a 'Domains' section with '+ casdoor.org'. Under 'Owners', there's an email field 'resultlee@gmail.com (You)' and a placeholder 'Enter email addresses'. At the bottom, a checked checkbox says 'Accept the reCAPTCHA Terms of Service'. A small note at the very bottom states: 'By accessing or using the reCAPTCHA APIs, you agree to the Google APIs [Terms of Use](#), Google [Terms of Use](#), and to the Additional Terms below. Please read and understand all applicable terms and policies before accessing the APIs.'

然后你可以获得一个站点密钥和一个秘密密钥。

The screenshot shows the Google reCAPTCHA registration interface. At the top, it says 'Google reCAPTCHA' and 'Adding reCAPTCHA to your site'. Below that, a message says "'reCaptcha' has been registered.''. It provides instructions to use the site key in HTML code and the secret key for communication between the site and reCAPTCHA. Both keys are displayed in text input fields with copy buttons. At the bottom, there are 'GO TO SETTINGS' and 'GO TO ANALYTICS' buttons.

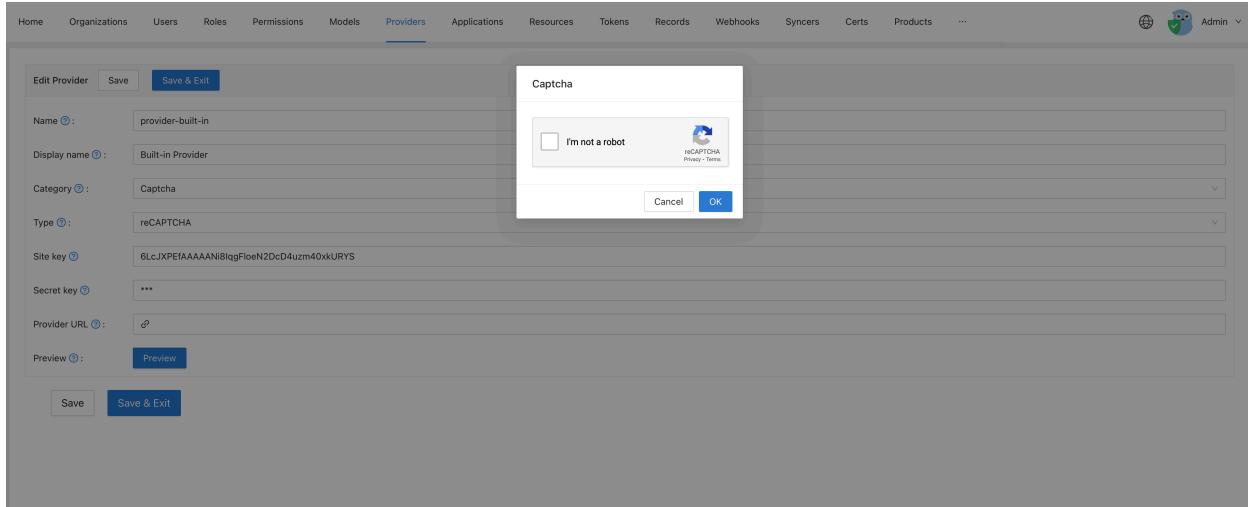
# 在 Casdoor 配置

在 Casdoor 中创建一个新的提供商。

选择类别为 **Captcha**，输入 reCAPTCHA. 你需要完成最后一步创建的站点密钥和秘密密钥。

The screenshot shows the Casdoor provider configuration interface. The 'Providers' tab is selected. A new provider is being created with the name 'reCaptcha'. The 'Category' is set to 'Captcha' and 'Type' to 'reCAPTCHA'. The 'Site key' and 'Secret key' fields are populated with values from the Google reCAPTCHA page. The 'Provider URL' field contains a placeholder URL. A 'Preview' button is available to see the provider's appearance. At the bottom, there are 'Save', 'Save & Exit', and 'Cancel' buttons.

你可以点击 **预览** button 来预览这个验证码的样式。



# 在应用中使用

编辑您想要在 Cassdoor 中配置的应用程序。选择刚刚添加的身份提供商，然后点击按钮 **保存**。

Providers	Add	Category	Type	canSignUp	canSignIn	canUnlink	prompted	Action
reCaptcha	Captcha							

# hCaptcha

hCaptcha 是一个验证码服务提供商，类似于reCAPTCHA。您可以从此 [链接](#) 中看到更多详细信息。

## 创建一个 API 密钥对

若要开始使用 hCaptcha，您需要 [注册您的网站的 API 密钥对](#)。您可以在您的 [个人资料页面](#) 找到您的网站密钥。

然后你可以获得一个站点密钥和一个秘密密钥。

## 在 Casdoor 配置

在 Casdoor 中创建一个新的提供商。

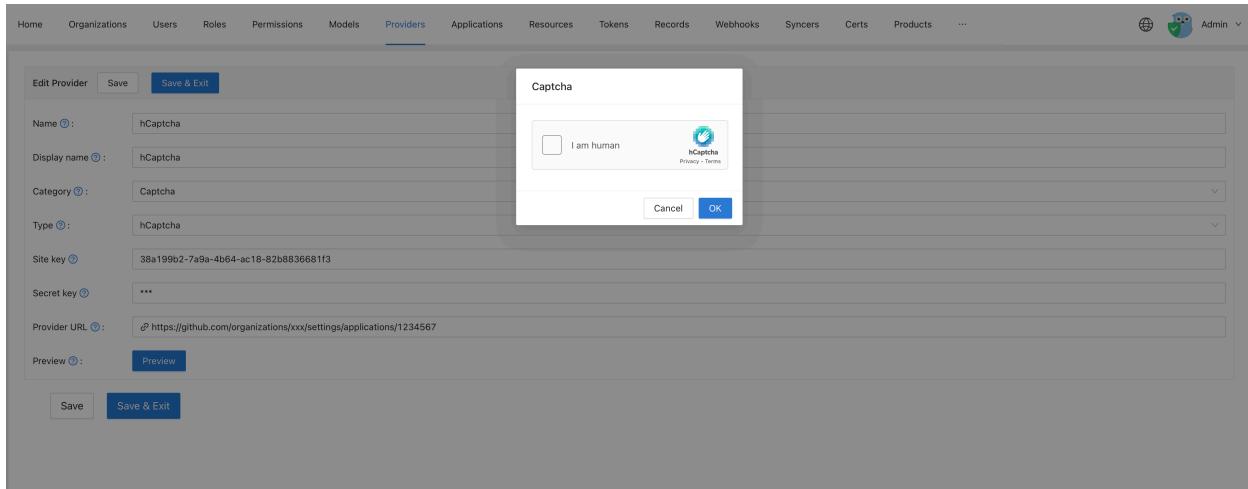
选择类别为 **验证码**，类型为 **hCaptcha**。你需要完成最后一步创建的站点密钥和秘密密钥。

The screenshot shows the Casdoor web interface with the 'Providers' tab selected. A new provider named 'hCaptcha' is being configured. The form fields include:

- Name: hCaptcha
- Display name: hCaptcha
- Category: Captcha
- Type: hCaptcha
- Site key: 38a199b2-7a9a-4b64-ac18-82b8836681f3
- Secret key: 38a199b2-7a9a-4b64-ac18-82b8836681f3
- Provider URL: https://github.com/organizations/xx/settings/applications/1234567
- Preview: Preview button

At the bottom, there are 'Save', 'Save & Exit', and 'Cancel' buttons.

你可以点击 预览 button 来预览这个验证码的样式。



## 在应用中使用

编辑您想要在 Cassdoor 中配置的应用程序。选择刚刚添加的身份提供商，然后点击按钮 保存。



# Aliyun Captcha

Aliyun Captcha是由Aliyun提供的验证码服务。 它包括两种验证验证码方式： [滑动验证](#) and [智能验证](#)。 您可以从此 [链接](#) 中看到更多详细信息。

## 在 Aliyun 中添加验证码配置

登录到 [Aliyun 管理控制台](#), 搜索并前往验证码服务。 然后点击 **确认打开** 以启用验证码服务。



输入验证码关联控制台后，点击 **添加配置**。

公告: 2021年3月18日起, 人机验证产品统一更名为验证码。

配置名称	appkey	scene	验证方式	业务类型	使用场景	最后更新	操作
测试验证码	F [REDACTED] B	nc_other	滑动验证	PC	其它	2022-06-20 23:47:37	自定义样式 系统代码集成
测试智能验证码	FF [REDACTED] B	lc_other	智能验证	PC	其它	2022-06-21 00:44:08	自定义样式 系统代码集成

共有2条 < 1 >

填写所有必需的信息并提交。

① 配置服务内容

② 系统代码集成&测试

③ 完成

④ 下一步

然后您可以在您的控制台中看到您的 场景 and App key

The screenshot shows the Alibaba Cloud Configuration Management interface under the '验证码' (Captcha) category. It lists two configurations:

配置名称	值	场景	验证方式	业务类型	使用场景	最后更新	操作
测试验证码	XXXXXXXXXXXXXXXXXXXX8B	nc_other	滑动验证	PC	其它	2022-06-20 23:47:37	自定义样式   系统代码集成
测试智能验证码	XXXXXXXXXXXXXXXXXXXXbb	lc_other	智能验证	PC	其它	2022-06-21 00:44:08	自定义样式   系统代码集成

Both the '值' (Value) column and the '场景' (Scene) column are highlighted with red boxes.

访问密钥，秘密访问密钥 在您的个人资料中。

## 在 Casdoor 配置

在 Casdoor 中创建一个新的提供商。

选择类别为 验证码，类型为 hCaptcha. 然后选择子类型： 滑动验证 或 智能验证。 您需要完成 访问密钥，秘密访问密钥，场景 和 应用密钥，这都是最后一步创建。

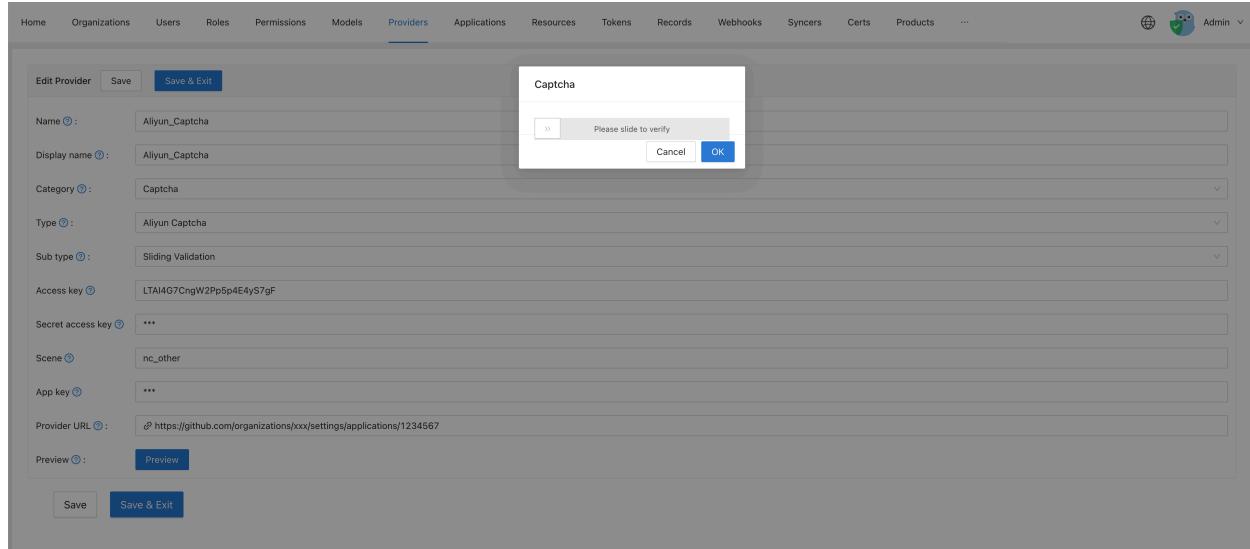
The screenshot shows the Casdoor Provider configuration page. A new provider named 'Aliyun\_Captcha' is being created. The configuration fields are as follows:

Name	Aliyun_Captcha
Display name	Aliyun_Captcha
Category	Captcha
Type	Aliyun Captcha
Sub type	Sliding Validation
Access key	XXXXXXXXXXXXXXXXXXXXgF
Secret access key	jIPXXXXXXXXXXXXXN
Scene	nc_other
App key	FXXXXXXXXXXXXXXXXXXXXBB
Provider URL	https://github.com/organizations/xxx/settings/applications/1234567

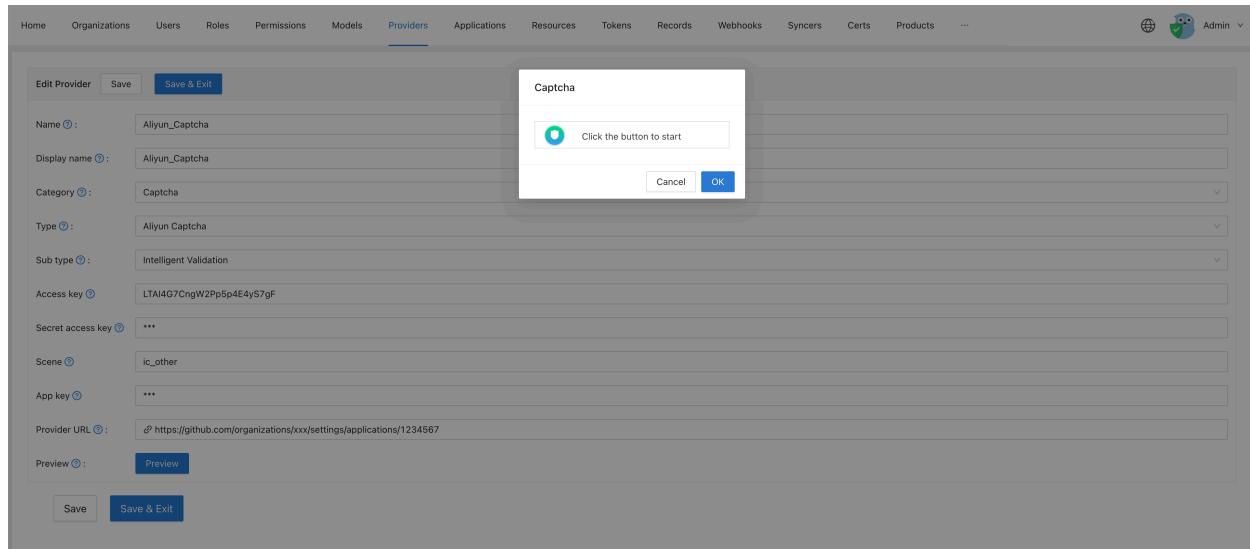
At the bottom, there are 'Save', 'Save & Exit', and 'Cancel' buttons.

你可以点击 预览 button 来预览这个验证码的样式。

下面的图像是 滑动验证 预览：



以下图像是 智能验证 预览：



# 在应用中使用

编辑您想要在 Cassdoor 中配置的应用程序。选择刚刚添加的身份提供商，然后点击按钮 **保存**。

Providers <small>(1)</small>	Add	Category	Type	canSignUp	canSignIn	canUnlink	prompted	Action
Aliyun_Captcha	<input checked="" type="checkbox"/>	Captcha						  



&gt;

资源

# 资源



## 概述

上传资源到Casdoor

# 概述

您可以在 casdoor 上传资源。在上传资源之前，您需要配置一个存储提供商。请查看[存储提供商](#)

您现在至少配置了一个存储提供程序，并将该提供程序添加到您的应用程序中。

Providers [?](#) :

Name	Category	Type
Provider_azure	Storage	A
Github_1	OAuth	G
provider_Alipay	Payment	支

好了！让我们看看如何[上传](#) and [删除](#) 资源的例子。

## 上传资源

用户可以将文件和图像等资源上传到先前配置的[云存储](#)。

Home    Organizations    Users    Roles    Permissions    Providers    Applications    Resources    Tc

Resources [Upload a file...](#)

Provider	Created time	Tag
provider_storage_aliyun_oss	2022-05-18 17:25:21	custom
provider_storage_aliyun_oss	2022-05-18 12:28:01	custom
provider_storage_aliyun_oss	2022-05-17 16:25:39	custom

## 删除资源

如果你不再需要资源，你可以选择点击“删除”按钮来删除它。

	Created time	Tag	Type	Format	File size	Preview	URL	Action
	2022-05-19 23:16:55	custom	image	.jpg	70.3 KB		<button>Copy Link</button>	<button>Delete</button>



&gt;

产品

# 产品



产品

添加您想要销售的产品



支付

查看支付中产品的交易信息

# 产品

您可以添加您想要销售的产品 (或服务)。下面将告诉您如何添加产品。

## 配置产品属性

第一：您需要了解产品的基本属性

[标签](#) [详情](#) [货币](#) [价格](#) [数量](#) [售价](#)

Tag  : Casdoor Summit 2022

Detail  : This is a description

Currency  : USD

Price  : 19

Quantity  : 100

Sold  : 10

## 支付服务提供商

当然，除了设置这些属性外，您还需要为产品添加付款提供者。和多个付款提供者可以添加到产品。

了解如何配置付款提供商, 请参阅 [付款提供商](#)

Payment providers [provider\\_Alipay](#) [X](#)

?: [provider\\_Alipay](#)

Return URL ?: <http://localhost:8000/products/callback>

最后, 填写 **返回 URL**。当付款完成时, 这是从付款提供者页面跳转到的URL。

## 预览产品

搞定! 查看评论并保存:

Preview ?:

[Test buy page..](#)

Buy Product

Name	Product
Detail	This is a subscription.
Image	
Price	\$300 (USD)
Pay	<a href="#">Alipay</a>

# 支付

付款成功后，您可以在 **付款** 中看到产品的交易信息，组织、用户、购买时间、产品名称等

## 发票

您可以输入编辑界面来开具发票

Type	Product	Price	Curre	Action
	A notebook computer	300	USD	<button>Result</button> <button>Edit</button> <button>Delete</button>

填写发票信息。发票类型是 **个人** 和 **机构**。

Message ② :

Person name ② :

Person ID card ② :

Person Email ② :

Person phone ② :

Invoice type ② :

Invoice title ② :

Invoice tax ID ② :

Invoice remark ② :

Invoice URL ② :

Invoice actions ② : Issue Invoice Return to Website

最后，点击“发出发票”按钮。



&gt;

用户

# 用户

## 概述

管理Casdoor的用户

## 角色

用户角色

## 权限

用户权限

# 概述

## User properties

作为一个认证平台，Casdoor 能够管理用户。每个用户都有这些属性：

- `Owner` 用户的所有者组织
- `Name` 用户名，唯一的
- `CreatedTime` 创建时间
- `UpdatedTime`
- `Id` 每个用户的 Id 都是唯一的。
- `Type`
- `Password`
- `PasswordSalt`
- `DisplayName` 显示在界面中的名称
- `FirstName`
- `LastName`
- `Avatar` 用户头像的链接
- `PermanentAvatar`
- `Email`
- `Phone`
- `Location`
- `Address`
- `Affiliation`
- `Title`

- `IdCardType`
- `IdCard`
- `Homepage`
- `Bio`
- `Tag`
- `Region`
- `Language`
- `Gender`
- `Birthday`
- `Education`
- `Score`
- `Karma`
- `Ranking`
- `IsDefaultAvatar`
- `IsOnline`
- `IsAdmin` 用户是否是他的组织的管理员
- `IsGlobalAdmin` 用户是否有权管理 Casdoor
- `IsForbidden`
- `IsDeleted`
- `SignupApplication`
- `Hash`
- `PreHash`
- `CreatedIp`
- `LastSigninTime`
- `LastSigninIp`
- `Roles` 用户角色数组

- `Permissions` 用户权限数组

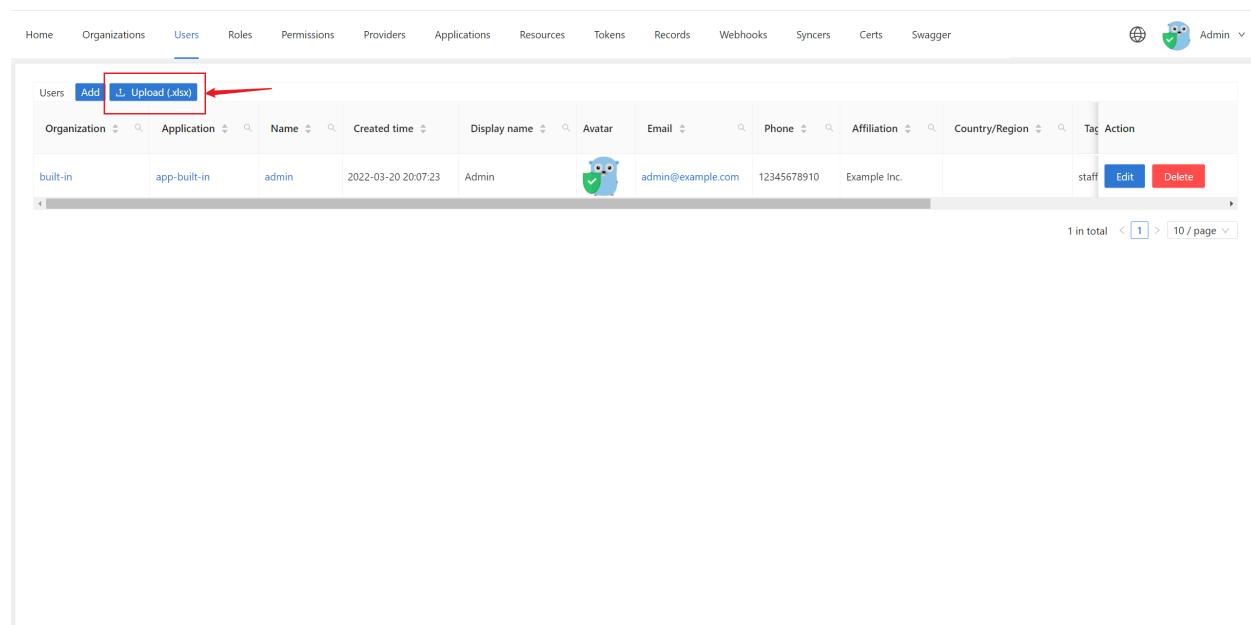
平台唯一ID:

- `Github`
- `Google`
- `QQ`
- `微信`
- `Facebook`
- `钉钉`
- `微博`
- `Gitee`
- `LinkedIn`
- `Wecom`
- `Lark`
- `Gitlab`
- `Adfs`
- `Baidu`
- `Casdoor`
- `Infoflow`
- `Apple`
- `AzureAD`
- `Slack`
- `Steam`
- `Ldap`
- `Properties` 这是一个字符串 → 字符串地图，存储所有其他属性。

# 从CSV文件导入用户

您可以通过上传用户信息的 XLSX 文件来添加新用户或更新现有的 Casdoor 用户。

在管理控制台中，转到用户并点击 **上传(.xlsx)** 按钮。



The screenshot shows the Casdoor user management interface. At the top, there is a navigation bar with links: Home, Organizations, Users, Roles, Permissions, Providers, Applications, Resources, Tokens, Records, Webhooks, Syncers, Certs, and Swagger. On the far right, there is a user profile icon labeled "Admin". Below the navigation bar, there is a search bar and a table listing users. The table has columns for Organization, Application, Name, Created time, Display name, Avatar, Email, Phone, Affiliation, Country/Region, Tag, and Action. A single user row is shown: built-in, app-built-in, admin, 2022-03-20 20:07:23, Admin, an owl icon, admin@example.com, 12345678910, Example Inc., staff, with Edit and Delete buttons. At the bottom of the table, it says "1 in total" and has navigation controls for page 1 of 10. Above the table, there is a button labeled "Upload (.xlsx)" with a red arrow pointing to it. The entire interface is contained within a light gray box.

选择您的 XLSX 文件并点击 Open，用户将被导入。

我们提供了 [模板XLSX文件 user\\_test.xlsx](#) 在 `xlsx` 文件夹中。该模板包括5个用户测试和某些必需的用户属性的信头。

Home   Organizations   **Users**   Roles   Permissions   Providers   Applications   Users uploaded successfully, refreshing the page   Syncers   Certs   Swagger    Admin

**user\_test.xlsx**

Organization	Application	Name	Created time	Display name	Avatar	Email	Phone	Affiliation	Country/Region	Tag	Action
built-in	app-built-in	tesla	2022-03-20 20:49:03	Nikola Tesla		9v73hn@example.com	40738134827	Example Inc.	United States of America	sciencist	<button>Edit</button> <button>Delete</button>
built-in	app-built-in	gauss	2022-03-20 20:48:33	Carl Friedrich Gauss		vqdsan@example.com	98621482844	Example Inc.	Germany	mathematician	<button>Edit</button> <button>Delete</button>
built-in	app-built-in	galileleo	2022-03-20 20:47:58	Galileo Galilei		8p4f38@example.com	22596937332	Example Inc.	Italy	scientist	<button>Edit</button> <button>Delete</button>
built-in	app-built-in	euler	2022-03-20 20:47:08	Leonhard Euler		3dzw4@example.com	74409642681	Example Inc.	Switzerland	mathematician	<button>Edit</button> <button>Delete</button>
built-in	app-built-in	einstein	2022-03-20 20:46:29	Albert Einstein		z6mive@example.com	60062541396	Example Inc.	Germany	scientist	<button>Edit</button> <button>Delete</button>
built-in	app-built-in	admin	2022-03-20 20:07:23	Admin		admin@example.com	12345678910	Example Inc.		staff	<button>Edit</button> <button>Delete</button>

6 in total < 1 > | 10 / page ▾

Made with ❤ by **Casdoor**

# 角色

每个用户可能有多个角色。 您可以在用户的个人资料上看到用户的角色。

Bio [?](#) :

Tag [?](#) : staff

Signup application [?](#) : app-built-in

Roles [?](#) : [role\\_test](#) [role\\_test2](#)

Permissions [?](#) : [permission\\_test](#)

3rd-party logins [?](#) :

Is admin [?](#) :

Is global admin [?](#) :

Is forbidden [?](#) :

Is deleted [?](#) :

## 角色属性

每个角色都具有这些属性：

- 所有者
- 名称
- 创建时间
- 显示名称
- 已启用
- 用户 此角色子用户的数组
- 角色 此角色子角色的数组



# 权限

每个用户可能有多个权限。 您可以在用户个人资料上看到用户的权限。

Bio [?](#) :

Tag [?](#) : staff

Signup application [?](#) : app-built-in

Roles [?](#) : [role\\_test](#) [role\\_test2](#)

Permissions [?](#) : [permission\\_test](#)

3rd-party logins [?](#) :

Is admin [?](#) :

Is global admin [?](#) :

Is forbidden [?](#) :

Is deleted [?](#) :

## 权限属性

权限拥有这些属性

- 所有者
- 名称
- 创建时间
- 显示名称
- 已启用
- 模型
- 用户 此角色子用户的数组

- 角色 此角色子角色的数组
- 资源类型
- 资源 资源数组
- 操作 操作数组
- 效果



&gt;

同步器

# 同步器

## 概述

Casdoor同步用户

## 数据库

使用 Database 同步器同步数据库

## Keycloak

使用 Keycloak Syncer 同步 Keycloak

# 概述

作为一个认证平台，Casdoor 可以轻松地操纵存储在数据库中的用户。

## 同步器

Casdoor 在 `user` 表中存储用户。当您计划使用 Casdoor 作为认证平台时，请不要担心您的应用程序 用户 数据迁移到 Casdoor。Casdoor 提供 **同步器** 来帮助您快速同步用户数据到 Casdoor。

指定您想要同步到 Casdoor 的数据库和用户表。同步器将在指定的间隔后同步数据。欲了解详情，请参阅 [数据库同步器](#)。

## 同步哈希值

Casdoor 使用哈希来确定如何更新用户。Casdoor 将计算表中每个用户的散列值。它是通过用户信息比如用户的密码或手机号码来生成的。

如果特定 `Id` 的用户计算的散列值与原始值相比有所改变，Casdoor 会确认哪个用户表已更新。然后，数据库将更新旧信息，实现 Casdoor 用户表和原始用户表之间的 **双向同步**。

# 数据库

## 数据库同步器

我们作为演示创建的用户表从 [模板 XLSX 文件](#) 导入。

owner	name	created_time	updated_time	id	type	password	password_salt	display_name	first_name	last_name	avatar	permanent_avatar	email
built-in	einstein	2022-03-20T20:46:29+08:00		1c57cc37-37f5-4def-9e9f-082189ef63d2	normal-user	123		Albert Einstein			<a href="https://casbin.org">https://casbin.org</a>		z6mive@
built-in	euler	2022-03-20T20:47:08+08:00		bb7831b4-0d24-4e96-b043-f8fd8d15eb	normal-user	123		Leonhard Euler			<a href="https://casbin.org">https://casbin.org</a>		3dzw4j@
built-in	galileo	2022-03-20T20:47:58+08:00		7920eb6c-f9f5-40ef-8e18-3ac99f49bd15	normal-user	123		Galileo Galilei			<a href="https://casbin.org">https://casbin.org</a>		8p4f38@
built-in	gauss	2022-03-20T20:48:33+08:00		f0c28816-2c0d-479b-b545-cb4cf96fdb36	normal-user	123		Carl Friedrich Gauß			<a href="https://casbin.org">https://casbin.org</a>		vqdsan@
built-in	tesla	2022-03-20T20:49:03+08:00		687c3068-fd21-4d32-b2ba-e13e8b369a	normal-user	123		Nikola Tesla			<a href="https://casbin.org">https://casbin.org</a>		9v73hn@

点击 **Syncers** 标签页并创建一个新的同步器。 填写下面所需的所有信息并保存。

Organization [?](#): built-in

Name [?](#): mysql test

Type [?](#): Database

Host [?](#): localhost

Port [?](#): 3306

User [?](#): root

Password [?](#): root

Database type [?](#): MySQL

Database [?](#): casdoorevdev

Table [?](#): user

Table primary key [?](#):

Table columns [?](#):

Table columns	Add	Column name	Column type	Casdoor column	Is hashed	Action		
		name	string	Name	<input checked="" type="checkbox"/>			
		create_time	string	CreatedTime	<input checked="" type="checkbox"/>			
		id	string	Id	<input checked="" type="checkbox"/>			
		password	string	Password	<input checked="" type="checkbox"/>			

Affiliation table [?](#):

Avatar base URL [?](#): <https://casbin.org>

Sync interval [?](#): 1000



提示

一般而言，您至少需要填写 Casdoor 列中的 `ID` 和 `Name` 以及其他重要信息，如 `createdTime`, `Password`, `DisplayName`.

以下是所需信息。

- `Organization`: 用户将导入的组织
- `Name`: 同步器名称
- `Type`: 选择数据库
- `Host`: 原始数据库主机
- `Port`: 原始数据库端口
- `User`: 原始数据库用户名
- `Password`: 原始数据库密码
- `Database type`: 所有 Xorm 支持的数据库，例如: MySQL, PostgreSQL, SQL Server, Oracle, Sqlite
- `Database`: 原始数据库名称
- `Table`: 原始用户表名称
- 表格列
- `Column name`: 原用户列名称
- `Column type`: 原用户列类型
- `Casdoor column`: casdoor 用户列名称
- `Is hashed`: 是否计算哈希值

然后您可以打开 **启用** 按钮并保存，同步器将开始工作。

Users <a href="#">Add</a> <a href="#"> Upload (xlsx)</a>											
Organization	Application	Name	Created time	Display name	Avatar	Email	Phone	Affiliation	Country/Region	Action	
built-in		euler	2022-04-08 10:10:45							<a href="#">Edit</a> <a href="#">Delete</a>	
built-in		gauss	2022-04-08 10:10:45							<a href="#">Edit</a> <a href="#">Delete</a>	
built-in		tesla	2022-04-08 10:10:45							<a href="#">Edit</a> <a href="#">Delete</a>	
built-in		einstein	2022-04-08 10:10:45							<a href="#">Edit</a> <a href="#">Delete</a>	
built-in		galileo	2022-04-08 10:10:45							<a href="#">Edit</a> <a href="#">Delete</a>	



# Keycloak

## Keycloak 同步器

Keycloak除了能自动配置 Table 和 Table columns 外，其他与 [数据库同步器](#) 基本相同。

此外，Keycloak 同步器将会查询 credential 表，keycloak\_group 表和 user\_group\_membership 表，因为Keycloak 中的用户信息储存在多个表中。

The screenshot shows the configuration page for a Keycloak syncer. The top section contains fields for Organization (built-in), Name (keycloak), Type (Keycloak), Host (localhost), Port (3306), User (root), Password (root), Database type (MySQL), and Database (keycloak). A red box highlights the 'Table' field, which is set to 'user\_entity'. A red arrow points from this field to a note: 'configured automatically after selecting Keycloak as syncer type'. Below this, the 'Table primary key' and 'Table columns' sections are shown. The 'Table columns' section is expanded, displaying a table with columns for Column name (ID, USERNAME, EMAIL, etc.) and Column type (string, boolean). It also includes Casdoor column mappings (Id, Name, DisplayName, Email, FirstName, LastName, CreatedTime, IsForbidden) and a 'Is hashed' column with toggle switches. A red border surrounds this entire table.

Column name	Column type	Casdoor column	Is hashed	Action
ID	string	Id	<input checked="" type="checkbox"/>	<span>Up</span> <span>Down</span> <span>Delete</span>
USERNAME	string	Name	<input checked="" type="checkbox"/>	<span>Up</span> <span>Down</span> <span>Delete</span>
EMAIL	string	DisplayName	<input checked="" type="checkbox"/>	<span>Up</span> <span>Down</span> <span>Delete</span>
EMAIL_VERIFIED	boolean	Email	<input checked="" type="checkbox"/>	<span>Up</span> <span>Down</span> <span>Delete</span>
FIRST_NAME	string	EmailVerified	<input checked="" type="checkbox"/>	<span>Up</span> <span>Down</span> <span>Delete</span>
LAST_NAME	string	FirstName	<input checked="" type="checkbox"/>	<span>Up</span> <span>Down</span> <span>Delete</span>
CREATED_TIMESTAMP	string	LastName	<input checked="" type="checkbox"/>	<span>Up</span> <span>Down</span> <span>Delete</span>
ENABLED	boolean	CreatedTime	<input checked="" type="checkbox"/>	<span>Up</span> <span>Down</span> <span>Delete</span>
		IsForbidden	<input checked="" type="checkbox"/>	<span>Up</span> <span>Down</span> <span>Delete</span>



&gt;

令牌

# 令牌



## 概述

Casdoor令牌简介

# 概述

Casdoor 基于 OAuth。 Tokens 是用户的 OAuth 的 token。

- Owner
- Name
- CreatedTime
- Application
- Organization
- User
- Code
- AccessToken
- ExpireIn 令牌将在数小时后过期
- Scope 授权范围
- TokenType 例如 输入 Bear



&gt;

Webhooks

# Webhooks



## 概述

在Casdoor添加webhooks

# 概述

事件系统允许您构建集成，订阅 Casdoor 上的某些事件。当其中一个事件被触发时，我们将向配置的 URL 发送一个以 json 为主体的 POST 请求。应用程序解析 json 并执行 hook。事件包括注册、登录、注销、更新用户，这些用户存储在记录的操作字段中。事件系统可以用来更新用户的外部问题。



&gt;

部署

# 部署



## Nginx

使用Nginx 来反转代理您的后端Go 程序，快速启动Casdoor 服务



## k8s

在 k8s 中部署 Casdoor

# Nginx

虽然Casdoor 是一个前后端分离的架构，但在生产环境中，后端程序仍然为前端文件提供静态文件服务。因此，您可以使用反向代理软件，如 [Nginx](#) 来代理 Casdoor 域的所有流量，并将其重定向到后端的端口。

在本章中，您将学习如何使用 Nginx 来反向代理您的后端 Go 程序，快速启动 Casdoor 服务。

## 1. 构建前端静态文件

现在假设您已经下载了 Casdoor 并完成了必要的配置。如果没有，请回到 [开始](#) 部分。

您只需要构建静态文件，例如：

[Yarn](#)    [npm](#)

```
yarn install && yarn run build
```

```
npm install && npm run build
```

## 2. 运行后端程序

```
go run main.go
```

或者先构建:

```
go build && ./main
```

## 3. 配置和运行 Nginx

```
vim /path/to/nginx/nginx.conf
```

添加服务器:

```
server {
    listen 80;
    server_name YOUR_DOMAIN_NAME;
    location / {
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
        proxy_redirect off;
        proxy_pass http://127.0.0.1:8000;
    }
}
```

然后重启您的 nginx 进程, 运行:

```
nginx -s reload
```

## 4. 测试

在您最喜欢的浏览器中访问 [http://YOUR\\_DOMAIN\\_NAME](http://YOUR_DOMAIN_NAME)



# k8s

## 在 k8s 中部署 Casdoor

我们已经给出了一个将 Casdoor 部署到 k8s 中的基本示例。在 casdoor 的根目录下，有一个名为“k8s.yaml”的文件，里面包含了在 k8s 中部署 casdoor 时使用的示例最低配置、一个部署和一个服务。

首先，确保你已经修改了conf/app.conf，使 casdoor 能够成功连接数据库，并且数据库正在运行。其次，确保 k8s 能够拉取必要的镜像。

运行

```
kubectl apply -f k8s.yaml
```

很快你就可以通过命令 `kubectl get pods` 看到结果

k8s.yaml 的内容如下

```
# this is only an EXAMPLE of deploying casddor in kubernetes
# please modify this file according to your requirements
apiVersion: v1
kind: Service
metadata:
  #EDIT IT: if you don't want to run casdoor in default namespace,
  #please modify this field
  #namespace: casdoor
  name: casdoor-svc
  labels:
    app: casdoor
```

该文件只是一个示例。比如您可以选择使用 default 以外的命名空间，使用服务类型而不是 nodeport 来暴露 casdoor，或者在 k8s 中使用 use config map 来挂载配置文件，这是 k8s 中比较推荐的方式。





&gt;

LDAP

# LDAP

## 概述

Casdoor 与ldap服务器合作

## 配置

Casdoor LDAP 配置

## LDAP 服务器

在Casdoor中连接LDAP服务器

# 概述

支持当前ldap服务器已被引入到Casdoor。 Casdoor 能够同步用户从ldap服务器到 Casdoor 使用它们作为用户帐户登录。 并使用 ldap 服务器验证它们。 Casdoor 还支持设置 cron 作业，以定期自动同步用户。

## Casdoor-Ldap 同步机制详情

下面将描述 Casdoor 如何与 ldap 服务器合作：

- i. 同步： Casdoor 可以连接到 ldap 服务器获取用户信息，读取所有用户信息（包含 `uidNumber`, `uid`, `cn`, `gidNumber`, `mail`, `email`, `emailAddress`, `telephoneNumber`, `mobile`, `mobileTelephoneNumber`, `registeredAddress`, `postalAddress`），并在 ldap 中为每个用户创建 Casdoor 帐户，并将帐户存储在数据库中。
- ii. 身份验证：正如我们所看到的那样， Casdoor 不会将 ldap 用户密码获取到 casdoor。因此，当从 ldap 服务器同步的帐户尝试通过 Casdoor 登录时， Casdoor 不会检查存储在 casdoor 数据库中的密码，而是连接到 ldap 服务器并验证用户的密码是否正确。
- iii. 用户区分： Casdoor 的同步只支持 `posixAccount` 的子类的 ldap 用户（具体来说， Casdoor 使用 `(objectClass=posixAccount)` 查询 ldap 用户）。 Casdoor 使用 `uid` 来专门识别用户，因此请确保每个 ldap 用户都有不同的 uid。

一旦用户同步到 Casdoor，用户的信息将与 ldap 用户分离，这意味着，如果您修改了在 Casdoor 中的用户信息 用户在 ldap 中的信息不会被修改，反之亦然（但 ldap 用户的密码除外，我们依靠它来验证用户）

# 配置

Ldap 配置属于一个组织，该组织的用户将被同步。

您应该使用全局管理员用户来修改配置。您需要在“组织”页面中输入LDAP用户同步的以下信息。

## 服务器名称

管理员使用易记的名称来区分不同的服务器。

例如: `LDAP 服务器示例`

## 服务器主机

LDAP 服务器主机。

例如: `example.com`

## 服务器端口

LDAP 服务器端口，只允许数字。

例如: `389`

## 基础DN

在LDAP中进行搜索时，Casdoor默认使用Sub搜索模式。基础DN是搜索的基本可识别的名称。Casdoor将返回当前基本DN下的所有用户。

在casdoor中配置的admin帐户至少应该具有base DN的只读权限。

例如: `ou=Example,dc=example,dc=com`

## 管理员

可以登录到指定的LDAP服务器的帐户。

使用 DN 或 ID 登录取决于您想要连接的 LDAP 服务器设置。

例如: `cn=manager,dc=example,dc=com`

## 管理员密码

管理员帐户的密码。

## 自动同步

设置 `0` 禁用自动同步，其他值表示 每隔几分钟。

# LDAP 服务器

许多系统，例如 [Nexus](#) 支持 [ldap](#) 身份验证。在Casdoor中，同样实现了一个简单的LDAP服务器，支持bind与search操作。

下面来介绍如何连接Casdoor中的LDAP服务器并实现简单的登录认证。

## LDAP 服务器端口号

默认情况下，LDAP服务器监听端口 [389](#)，您可以通过更改 [conf/app](#) 中的 [ldapServerPort](#) 来更改默认端口号。

## 工作原理

类似于Casdoor的ldap客户端，ldap服务器中的用户都是 [posixAccount](#) 的子类。

当服务器收到符合LDAP协议传输的一组数据时，它将解析 [cn](#) and [ou](#)，其中 [cn](#) 表示用户名，[ou](#) 表示组织名称。[dc](#) 是什么并不重要。

如果是 [bind](#) 操作，服务器将使用 Casdoor 验证用户名和密码并给予用户权限。

如果它是一个 [search](#) 操作，服务器将根据 [bind](#) 赋予客户端的权限，检查 [search](#) 是否合法，并返回相应响应。

### ① 信息

我们只支持 [Simple Authentication](#)。

## Bind

在Catdoor ldapserver中，我们识别类似于这样的 DN : cn=admin,ou=buildt-in,dc=example,dc=com

所以请将管理员用户的 DN 设置为上面的形式。然后您可以使用此 DN 绑定到 ldap 服务器，使用 Casdoor 用户的密码登录以进行验证。如果服务器验证通过，用户将被授予 Casdoor 中的权限。

## Search

bind 操作成功完成后，您可以执行 search 操作。search 与 bind 在细节上有一些区别。

- 如果您想要搜索特定的用户，比如 built-in 组织下的 Alice，您应该使用这样的 DN : ou=buildt-in,dc=example,c=com，并在过滤器字段中添加 cn=Alice
- 如果您想要在某个组织下搜索所有用户，如 built-in 下的所有用户，您应该使用这样的 DN : ou=built-in,dc=example,dc=com，并在过滤器字段中添加 cn=\*
- 如果您想要搜索所有组织的所有用户(前提是用户有足够的权限)，您应该使用这样的 DN : ou=\*,dc=example,dc=com，并在过滤器字段中添加 cn=\*。



&gt;

集成

# 集成



5 个项目



9 个项目



1 个项目



1 个项目



PHP

2 个项目



Ruby

1 个项目

# Go



## BookStack

在 BookStack 中使用 Casdoor 进行身份验证



## ELK

Casdoor/elk-auth-casdoor概览



## Gitea

在 Gitea 中使用 Casdoor 进行身份验证



## Grafana

在 Grafana 中使用 Casdoor 进行身份验证

 MinIO

配置 Casdoor 作为身份提供者与 MinIO 支持

# BookStack

## 在 BookStack 中使用 Casdoor 进行身份验证

BookStack 是一个开源书和文档共享网站 以及使用 Go 语言开发的开源应用程序，帮助您更好地实现文档阅读管理。

Bookstack-casdoor 已经与 castor集成，您现在可以快速开始一个简单的配置。

### 步骤1. 创建一个Casdoor应用程序

转到您的 Casdoor 并添加您的新应用程序 BookStack。 下面是在Casdoor中创建 BookStack应用程序的示例。

Edit Application Save Save & Exit

Name ? : bookstack

Display name ? : bookstack

Logo ? : URL ? : [https://cdn.casdoor.com/logo/casdoor-logo\\_1185x256.png](https://cdn.casdoor.com/logo/casdoor-logo_1185x256.png)

Preview: 

Home ? :

Description ? :

Organization ? :

Client ID ? :

Client secret ? :

请记住 **名称**, **组织**, **客户端 ID**, 和 **客户密钥**。您将在下一步中使用它们。

## 步骤2. 配置Casdoor 登录

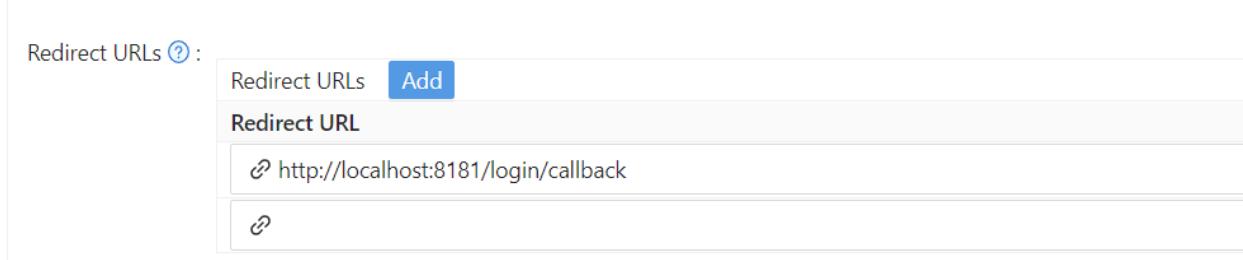
现在, 请移动到BookStack。查找文件: `oauth.conf.example`。

将 `oauth.conf.example` 重命名为 `oauth.conf` 和 **修改配置** 默认内容为:

```
[oauth]
```

## 步骤3. 在Casdoor中填写 重定向URL

最后一步，请回到您添加 BookStack 应用程序的页面，填写 重定向URL 请确保 重定向 URL 和 oauth.conf 文件中的 重定向URL 相同。



The screenshot shows the 'Redirect URLs' section of the Casdoor configuration interface. It includes a header with 'Redirect URLs' and a blue 'Add' button. Below is a table with one row. The first column is 'Redirect URL' and the second column contains the value 'http://localhost:8181/login/callback'. There is also a small 'edit' icon next to the URL.

Redirect URL	
http://localhost:8181/login/callback	edit

现在你已经完成了Casdoor的所有配置！

一旦BookStack成功部署，您可以使用Casdoor返回BookStack和体验以进行登录认证。

# ELK

## Casdoor/elk-auth-casdoor概览

ELK (Elasticsearch、Logstash 和 Kibana) 的缺点是，这些产品原来没有认证机制。只要拥有kibana或ESurl，每个人都可以访问kibana dashboard。后来ELK 集成了一个嵌入式认证系统“Xpack”，其所有高级函数 **都不是免费的** (例如 Oauth, OIDC, LDAP, SAML)，而且只有纯粹的身份验证(设置一套账户和密码) 是免费的，这相当不方便。我们不能为每个人提供一个独特的帐户。

因此，我们已经开发了一个基于 Casdoor 免费的 elk 认证解决方案。**开源和维护中，支持大量高级功能。** Casdoor是一个基于Oauth2的中心化身份验证/单点登录平台。/OIDC, casdoor /elk-auth-casdoor 实际上是一种逆向代理，它旨在拦截所有 http 数据流到elk/kibana， 并指导尚未登录的用户登录。只要用户已登录，此逆向代理将完全透明。

如果此用户未成功通过验证，请求将会临时缓存，用户将被重定向到Casto登录页面。在用户成功登录到casdoor后，缓存请求将会还原并发送到 kibana。因此，如果一个 POST请求（或GET以外的其他请求）被拦截，也是可以的，用户不需要重新填写表格和重新发送请求。逆向代理将为你记住它。

Castor/elk-auth-casdoor 版本库的位置 <https://github.com/casdoor/elk-auth-casdoor>

## 如何使用？

0. 已安装golang环境

1. 转到 [casdoor/elk-auth-casdoor](#) 并获取代码
2. 将您的代理注册为Casdoor应用程序。
3. 修改配置

配置文件位于"conf/app.conf"中。这是一个示例，您应该根据您的实际需求自行更改。

```
appname = .
# port on which the reverse proxy shall be run
httpport = 8080
runmode = dev
#EDIT IT IF NECESSARY. The url of this reverse proxy
pluginEndpoint="http://localhost:8080"
#EDIT IT IF NECESSARY. The url of the kibana
targetEndpoint="http://localhost:5601"
#EDIT IT. The url of casdoor
casdoorEndpoint="http://localhost:8000"
#EDIT IT. The clientID of your reverse proxy in casdoor
clientID=ceb6eb261ab20174548d
#EDIT IT. The clientSecret of your reverse proxy in casdoor
clientSecret=af928f0ef1abc1b1195ca58e0e609e9001e134f4
#EDIT IT. The application name of your reverse proxy in casdoor
appName=ELKProxy
#EDIT IT. The organization to which your reverse proxy belongs in
casdoor
organization=built-in
```

4. 访问 <http://localhost:8080> (在上文示例中) 并按照重定向指南登录，您将会看到 kibana受casdoor的保护和认证。
5. 如果一切运行良好，请不要忘记通过配置您的防火墙，阻止原来的 kibana 端口访问外部，这样其他人只能通过这种逆向代理访问kibana。

# Gitea

## 在 Gitea 中使用 Casdoor 进行身份验证

Gitea 是一个社区管理的轻量代码托管解决方案，写入Go。采用MIT开源协议

Gitea支持第三方身份验证，包括Oauth，这样就可以使用Casdoor进行身份验证。以下是操作教程。

### 准备：

要配置 Gitea 使用 Casdoor 作为身份识别提供者，您需要安装 Gitea 以及访问管理员帐户。

关于如何下载、安装和运行 Gitea 的更多信息，请参阅 <https://docs.gitea.io/en-us/install-from-binary/>

您需要在安装过程中创建管理员帐户。如果您已经注册，管理员将是第一个注册用户。请使用此帐户继续以下操作。

### 1. 创建一个Casdoor应用程序

像这样：

Edit Application

Name ⓘ: application\_9p7eai

Display name ⓘ: New Application - 9p7eai

Logo ⓘ: URL ⓘ: https://cdn.casbin.com/logo/logo\_1024x256.png

Preview: 

Home ⓘ: [View](#)

Description ⓘ:

Organization ⓘ: built-in

Client ID ⓘ: 7ceb9b7fda4a9061ec1c

Client secret ⓘ: 3416238e1edf915eac08b8fe345b2b95cdba7e04

Cert ⓘ: cert-built-in

Redirect URLs

Action	Redirect URLs	Add
<a href="#">Edit</a> <a href="#">Delete</a>	http://localhost:3000/user/oauth2/Casdoor/callback	<a href="#">Add</a>

请记住客户端ID和密码，以便下一步操作。

请不要在此步骤中填写回调url。Url取决于下一步Gitea的配置。稍后我们将返回来设置一个正确的回调url。

## 2. 配置 Gitea 使用 Casdoor

以管理员身份登录。通过右上角的下拉菜单转到“站点管理”页面。然后切换到“认证源”页面

你应该看到类似下面的内容：

The screenshot shows the GitHub interface for managing authentication sources. At the top, there are navigation links: Issues, Pull Requests, Milestones, Explore, a notifications bell icon, a plus sign for creating new items, and a gear icon for settings. Below this is a secondary navigation bar with links: Dashboard, User Accounts, Organizations, Repositories, Webhooks, Authentication Sources (which is underlined, indicating it's the active page), User Emails, Configuration, System Notices, and Monitoring. The main content area is titled "Authentication Source Management (Total: 0)". It features a table with the following columns: ID, Name, Type, Enabled, Updated, Created, and Edit. A blue button labeled "Add Authentication Source" is located at the top right of the table area.

按“添加认证源”按钮并填写类似的表单。

The screenshot shows the "Add Authentication Source" form on GitHub. The form has the following fields:

- Authentication Type: OAuth2
- Authentication Name: Casdoor
- OAuth2 Provider: OpenID Connect
- Client ID (Key): 7ceb9b7fda4a9061ec1c
- Client Secret: 3416238e1edf915eac08b8fe345b2b95cdba7e04
- Icon URL: (empty)
- OpenID Connect Auto Discovery URL: http://localhost:8000/.well-known/openid-configuration
- Skip local 2FA: Leaving unset means local users with 2FA set will still have to pass 2FA to log on
- Additional Scopes: (empty)

请选择认证类型为“oauth2”。

请输入此认证源的名称并 **记住此名称**。此名称将在下一步骤中用于回调url。

请选择 **OpenID Connect** Oauth2 提供商。

填写上一步中记住的**客户端ID**和**客户端密码**。

在 **openid** 连接中填写自动发现URL，它应该是 `<your endpoint of casdoor>/.well-known /openid-configuration`。

按您的意愿填写其他可选配置项。然后提交它。

提交表单

### 3. 配置后台回调url

返回第2步中的应用程序编辑页面并添加以下回调url：

`<endpoint of gitea>/user/oauth2/<authentication source name>/callback`

`<authentication source name>` 是上一步Gitea认证源的名称。

### 4. 在 Gitea 上试试

退出当前管理员帐户。

您应该在登录页面中看到这一点：

The screenshot shows a top navigation bar with 'Sign In' and 'OpenID' buttons. Below is a 'Sign In' form with fields for 'Username or Email Address' and 'Password'. There is a 'Remember this Device' checkbox, a 'Sign In' button, a 'Forgot password?' link, and a 'Need an account? Register now.' link. At the bottom is an 'OpenID Connect' sign-in button.

Sign In

OpenID

Sign In

Username or Email Address \*

Password \*

Remember this Device

**Sign In**   [Forgot password?](#)

[Need an account? Register now.](#)

Sign In With OpenID Connect

按“使用openid登录”按钮，您将被重定向到casdoor登录页面。

登录后您将看到这个：

The screenshot shows a 'Complete New Account' form within a Gitea interface. It includes fields for 'Username' and 'Email Address', both marked with a red asterisk indicating they are required. A 'Complete Account' button is at the bottom.

Explore Help

Register New Account [Link to Existing Account](#)

Complete New Account

Username \*

Email Address \*

admin@example.com

**Complete Account**

按照指示并用一个新的 Gitea 帐户或现有帐户绑定下级帐户。

然后一切都将正常工作。



# Grafana

## 在 Grafana 中使用 Casdoor 进行身份验证

Grafana 支持通过 Oauth 进行认证。因此，用户在Grafana上登录变得非常容易。只有几个步骤和简单的配置就能做到这一点。

这是一个使用 Grafana 的 Casdoor 进行身份验证的教程。在您继续之前，请确保您已安装 grafana 并正在运行。

## 步骤 1: 在Casdoor中创建一个Grafana

这是一个在 Casdoor 中创建应用程序的示例

Edit Application [Save](#) [Save & Exit](#)

Name ⓘ: application\_9p7eai

Display name ⓘ: New Application - 9p7eai

Logo ⓘ: URL ⓘ: [https://cdn.casbin.com/logo/logo\\_1024x256.png](https://cdn.casbin.com/logo/logo_1024x256.png)

Preview:



Home ⓘ: [/](#)

Description ⓘ:

Organization ⓘ: built-in

Client ID ⓘ: 7ceb9b7fda4a9061ec1c

Client secret ⓘ: 3416238e1edf915eac08b8fe345b2b95cdba7e04

Cert ⓘ: cert-built-in

Redirect URLs ⓘ:

Redirect URLs	Add	Action
<a href="http://localhost:3000/login/generic_oauth">http://localhost:3000/login/generic_oauth</a>		

请复制 client secret 和 client ID 以便下一步操作。

请添加 Grafana 回调 url。默认情况下，Gravana 的 oauth 回调是 /login/generic\_oauth。所以请正确地拼接这个 url。

## 第 2 步：修改 Grafana 的配置

默认情况下，在 conf/default.ini 处的 oauth 位置的配置文件。

请找到 auth.generic\_oauth 并修改以下字段：

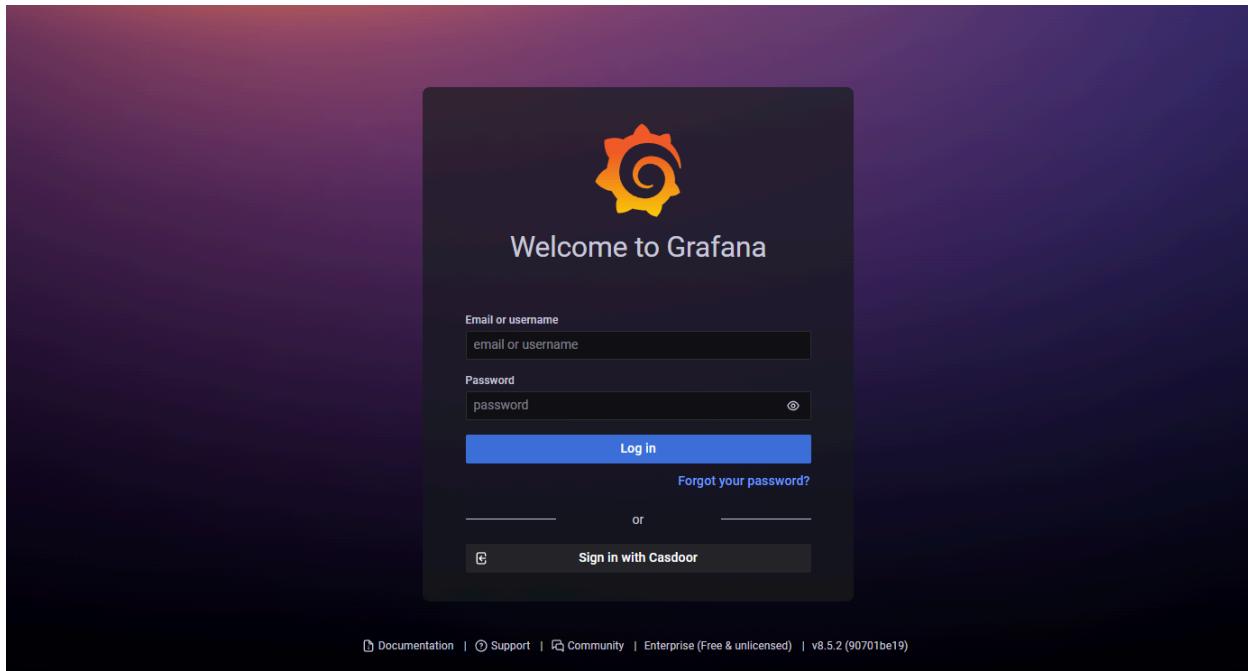
```
[auth.generic_oauth]
name = Casdoor
icon = signin
enabled = true
allow_sign_up = true
client_id = <client id in previous step>
client_secret = <client secret in previous step>
auth_url = <endpoint of casdoor>/login/oauth/authorize
token_url = <endpoint of casdoor>/api/login/oauth/access_token
```

如果您要启用 castoor 的 https 功能，请同时设置 tls\_skip\_verify\_security = true

## 第3步：查看它是否正常运作。

关掉 Grafana 并重启它。

去查看登录页面，你应该看到这个



# MinIO

MinIO 支持使用OpenID Connect (OIDC) 兼容的程序提供服务进行外部身份管理。此文档介绍了通过配置Casdoor为身份提供者来支持MinIO。

## 第 1 步：部署Casdoor & MinIO

第一，应该先部署Casdoor。

您可以参考 [服务器安装](#) 的 Casdoor 官方文档。

在成功部署后，您需要确保：

- Casdoor服务器已成功运行在 <http://localhost:8000> 上了。
- 打开您最喜欢的浏览器并访问 <http://localhost:7001>，您将看到Casdoor的登录页面。
- 输入 `admin` 和 `123` 测试登录功能正常工作。

然后您可以通过以下步骤在自己的应用程序中快速实现基于Casdoor的登录页面。

您可以参考 [这里](#) 来部署您的 MinIO 服务器，以及[这里](#) 对于名为 `mc` 的 MinIO 客户端。

## 第 2 步：配置Casdoor应用

1. 创建或使用现有的 Casdoor 应用程序。
2. 添加您的redirect url

Client ID [?](#) :  Client ID

Client secret [?](#) :  Client Secret

Redirect URLs [?](#) :  
Redirect URLs [Add](#)  
Redirect URL [Add a redirect URL for spring security](#)  
[🔗](#) http://localhost:8082/ui-one/login/oauth2/code/custom

3. 添加您想要的提供商并补充其他设置。

这时，您可以在应用程序设置页面获得两个值：`Client ID` 和 `Client secret`，就像上图一样。我们将在下一步骤中使用它们。

打开您最喜欢的浏览器并访问：`http://CASDOOR_HOSTNAME/.well-known/openid-configuration`，您将看到Casdoor配置的OIDC。

4. 这个步骤对于MinIO是必要的。由于MinIO需要在JWT中使用一个索赔属性来执行其政策，因此您也应该在Casdoor中配置它。目前，Casdoor使用`tag`作为配置MinIO策略的一个环境。

Tag [?](#) :

您可以在这里找到所有支持的策略。

## 第3步：配置MinIO

您可以通过以下命令启动MinIO服务器：

导出`MINIO_ROOT_USER=minio`

您可以使用参数 `--console-address` 来配置地址和端口。

然后您可以在 MinIO 客户端 `mc` 中添加一个服务别名。

```
mc 别名设置了myminio <You console address> minio minio123
```

现在，您可以配置 MinIO 的 OpenID 连接。对于 Casdoor，命令如下：

```
mc admin config set myminio identity_openid
config_url="http://CASDOOR_HOSTNAME/.well-known/openid-
configuration" client_id=<client id> client_secret=<client secret>
claim_name="tag"
```

更详细的参数，您可以参考 [官方文档](#)。

一旦设置成功，请重启 MinIO 实例。

```
mc 管理服务重启myminio
```

## 第 4 步：试用一下 demo！

现在，您可以在浏览器上打开您的 MinIO 控制台，然后点击 `Login with SSO`。

您将重新转至登录页面 您将被重定向到 Casdoor 用户登录页面。登录成功后，您将被重定向到 MinIO 页面并自动登录，您应该现在看到他们可以访问的 buckets 和 objects。

### ⚠ 注意事项

如果您在不同的端口部署 Casdoor 前端和后端，您被重定向的登录页面将是后端端口，它将显示 `404 找不到`。您可以修改端口到前端。然后您可以成功访问 Casdoor 登录页面。



# Java

## Spring Boot

在Spring Boot项目中使用 Casdoor

## Spring Cloud

在Spring Cloud中使用 Casdoor

## Spring Cloud Gateway

在Spring Cloud Gateway中使用 Casdoor

## Spring 安全

2 个项目

 Jenkins 插件

使用 Casdoor 插件来保证您的 Jenkins 安全

 Jenkins OIDC

使用 OIDC 协议作为IDP 连接各种应用程序，如Jenkins

 RuoYi

在 RuoYi-Cloud 上使用 Casdoor

 Pulsar-manager

在 Pulsar-manager 中使用 Casdoor

 ShenYu

在 ShenYu 中使用 Casdoor

# Spring Boot

[casdoor-spring-boot-example](#) 是关于如何在 SpringBoot 项目中使用 [casdoor-spring-boot-starter](#) 的示例。我们将向您展示以下步骤。

## 第1步：部署Casdoor

第一，应部先部署Casdoor。

您可以参考 [服务安装](#) 的 Casdoor 官方文档。请在 [生产模式](#) 中部署您的 castor 实例。

在成功部署后，您需要确保：

- 打开您最喜欢的浏览器并访问 <http://localhost:8000>，您将看到Casdoor的登录页面。
- 输入 `admin` 和 `123` 测试登录功能正常工作。

然后您可以通过以下步骤在自己的应用程序中快速实现基于 Casdoor 的登录页面。

## 第2步：导入 casdoor-spring-boot-starter

您可以使用 maven 或 gradle 导入 casdoor-spring-boot-starter。

[Maven](#) [Gradle](#)

```
<!-- https://mvnrepository.com/artifact/org.casbin/casdoor-spring-boot-starter -->
<dependency>
    <groupId>org.casbin</groupId>
    <artifactId>casdoor-spring-boot-starter</artifactId>
    <version>1.x.y</version>
</dependency>

// https://mvnrepository.com/artifact/org.casbin/casdoor-spring-boot-starter
implementation group: 'org.casbin', name: 'casdoor-spring-boot-starter', version: '1.x.y'
```

## 第 3 步： 初始化配置

初始化需要6个参数，它们都是字符串类型。 | 名称(按顺序排列) | 必选项 | 描述 ||  
----- | --- | ----- | | endpoint | 是 |  
Casdoor 服务URL, 例如 http://localhost:8000 | | clientId | 是 |  
Application.client\_id | | clientSecret | 是 | Application.client\_secret | | certificate |  
Yes | Application.certificate | | organizationName | 是 | Application.organization | |  
applicationName | 否 | Application.name | 您可以使用 Java properties 或 YAML 文件  
来初始化，如下所示。

Properties

YML

---

```
casdoor.endpoint = http://localhost:8000
casdoor.clientId = <client-id>
casdoor.clientSecret = <client-secret>
casdoor.certificate = <certificate>
casdoor.organizationName = built-in
```

```
casdoor:  
  endpoint: http://localhost:8000  
  client-id: <client-id>  
  client-secret: <client-secret>  
  certificate: <certificate>  
  organization-name: built-in  
  application-name: app-built-in
```

### ⚠ 注意事项

您应该用您自己的 Casdoor 实例替换配置，特别是 `clientId`, `clientSecret` 和 `jwtPublicKey`。

## 第 4 步：重定向到登录页面

当您需要访问您的应用程序的身份验证时，您可以发送目标 url 并重定向到 Casdoor 提供的登录页面。请确保您已提前在应用配置中添加回调 url(例如 `http://localhost:8080/login`)。

```
@Resource  
private CasdoorAuthService casdoorAuthService;  
  
@RequestMapping("toLogin")  
public String toLogin() {  
    return "redirect:" +  
    casdoorAuthService.getSigninUrl("http://localhost:8080/login");  
}
```

# 第 5 步： 获取令牌并解析

在 Casdoor 验证通过后，它将被重定向到您的应用程序，并带有 code 和状态。

您可以获取 code 并调用 `getOAuthToken` 方法，然后解析出 jwt 令牌。

`Casdoor User` 包含了由Casdoor提供的有关用户的基本信息。您可以使用它作为关键词在您的应用程序中设置会话。

```
@RequestMapping("login")
public String login(String code, String state, HttpServletRequest
request) {
    String token = "";
    CasdoorUser user = null;
    try {
        token = casdoorAuthService.getOAuthToken(code, state);
        user = casdoorAuthService.parseJwtToken(token);
    } catch (CasdoorAuthException e) {
        e.printStackTrace();
    }
    HttpSession session = request.getSession();
    session.setAttribute("casdoorUser", user);
    return "redirect:/";
}
```

## Service

这两种情况的示例如下所示：

- CasdoorAuthService

- `String token = casdoorAuthService.getOAuthToken(code, "app-`

- ```
built-in");
```
- ```
CasdoorUser casdoorUser =
casdoorAuthService.parseJwtToken(token);
```
- CasdoorUserService
  - ```
CasdoorUser casdoorUser = casdoorUserService.getUser("admin");
```
  - ```
CasdoorUser casdoorUser =
casdoorUserService.getUserByEmail("admin@example.com");
```
  - ```
CasdoorUser[] casdoorUsers = casdoorUserService.getUsers();
```
  - ```
CasdoorUser[] casdoorUsers =
casdoorUserService.getSortedUsers("created_time", 5);
```
  - ```
int count = casdoorUserService.getUserCount("0");
```
  - ```
CasdoorResponse response = casdoorUserService.addUser(user);
```
  - ```
CasdoorResponse response =
casdoorUserService.updateUser(user);
```
  - ```
CasdoorResponse response =
casdoorUserService.deleteUser(user);
```
- CasdoorEmailService
  - ```
CasdoorResponse response = casdoorEmailService.sendEmail(title,
content, sender, receiver);
```
- CasdoorSmsService
  - ```
CasdoorResponse response =
casdoorSmsService.sendSms(randomCode(), receiver);
```
- CasdoorResourceService
  - ```
CasdoorResponse response =
casdoorResourceService.uploadResource(user, tag, parent,
fullFilePath, file);
```
  - ```
CasdoorResponse response =
casdoorResourceService.deleteResource(file.getName());
```

# 更多内容

您可以探索以下项目/文件来了解更多关于Java与Casdoor一体化的信息。

- [casdoor-java-sdk](#)
- [casdoor-spring-boot-starter](#)
- [casdoor-spring-boot-example](#)
- [casdoor-spring-security-example](#)
- [casdoor-spring-security-react-example](#)
- [casdoor-spring-boot-shiro-example](#)

# Spring Cloud

在Spring Cloud微型服务体系下，一般由网关进行认证。请参阅 [casdoor-springcloud-gateway-example](#)。

如果您想要在单个服务中使用Casdoor，您可以参考 [casdoor-spring-boot](#)示例。

无论是网关还是在某个服务负责认证，都需要使用 [casdoor-spring-boot-starter](#)

## 更多内容

您可以查阅以下项目/文件来了解更多关于Java中集成Casdoor信息。

- [casdoor-java-sdk](#)
- [casdoor-spring-boot-starter](#)
- [casdoor-spring-boot-example](#)
- [casdoor-spring-security-example](#)
- [casdoor-spring-security-react-example](#)
- [casdoor-spring-boot-shiro-example](#)
- [casdoor-springcloud-gateway-example](#)

# Spring Cloud Gateway

[casdoor-springcloud-gateway-example](#) 是如何在 Spring Cloud Gateway 中使用 [casdoor-spring-boot-starter](#) 作为一个 OAuth2 插件。我们将向您展示以下步骤。

## 第1步 部署Casdoor

首先，应当部署Casdoor。

您可以参考 [Server Installation](#) 的 Casdoor 官方文档。请在 **生产模式** 中部署您的 Casdoor 实例。

在成功部署后，您需要确认：

- 打开您最喜欢的浏览器并访问 <http://localhost:8000>，您将看到Casdoor的登录页面。
- 输入 `admin` 和 `123` 测试登录功能正常工作。

然后您可以通过以下步骤在自己的应用程序中快速实现基于 Casdoor 的登录页面。

## 第2步：初始化一个Spring Cloud Gateway 项目

您可以直接使用此示例的代码或结合您自己的业务代码。

我们需要一个网关服务和至少一个业务服务。

在这个示例中，`casdoor-gateway` 作为网关服务，`casdoor-api` 作为业务服务。

## 第3步：引入依赖

将 `casdoor-spring-boot-starter` 添加到Spring Cloud Gateway项目。

对于Apache Maven:

```
/casdoor-gateway/pom.xml
```

```
<!-- https://mvnrepository.com/artifact/org.casbin/casdoor-spring-boot-starter -->
<dependency>
    <groupId>org.casbin</groupId>
    <artifactId>casdoor-spring-boot-starter</artifactId>
    <version>1.x.y</version>
</dependency>
```

对于Gradle:

```
// https://mvnrepository.com/artifact/org.casbin/casdoor-spring-boot-starter
implementation group: 'org.casbin', name: 'casdoor-spring-boot-starter', version: '1.x.y'
```

## 第4步：配置属性

初始化需要 6 个参数，它们都是字符串类型：

名称(按顺序排列)	是否必须	描述
endpoint	是	Casdoor 服务URL, 例如 <code>http://localhost:8000</code>
clientId	是	Application.client_id
clientSecret	是	Application.client_secret
certificate	是	Application.certificate
organizationName	是	Application.organization
applicationName	否	Application.name

您可以使用 Java properties 或 YAML 文件来初始化，如下所示。

对于 properties:

```
casdoor.endpoint=http://localhost:8000
casdoor.clientId=<client-id>
casdoor.clientSecret=<client-secret>
casdoor.certificate=<certificate>
casdoor.organizationName=built-in
casdoor.applicationName=app-built-in
```

对于yaml:

```
casdoor:
  endpoint: http://localhost:8000
```

此外，还需要配置网关路由。对于yaml：

```
spring:
  application:
    name: casdoor-gateway
  cloud:
    gateway:
      routes:
        - id: api-route
          uri: http://localhost:9091
          predicates:
            - Path=/api/**
```

## 第5步：添加CasdoorAuthFilter

将 GlobalFilter 的一个实现类添加到网关中做身份验证，例如此示例中的 CasdoorAuthFilter。

如果身份验证失败，则返回到401，前端跳转到统一登录界面。

```
@Component
public class CasdoorAuthFilter implements GlobalFilter, Ordered {

  private static final Logger LOGGER =
LoggerFactory.getLogger(CasdoorAuthFilter.class);

  @Override public int getOrder() {
    return 0;
  }

  @Override public Mono<Void> filter(ServerWebExchange exchange,
GatewayFilterChain chain) {
    return exchange.getSession().flatMap(webSession -> {
      CasdoorUser user =
```

# 第6步：使用Casdoor提供的相关Service

现在提供 5 种Service: `CasdoorAuthService`, `CasdoorUserService`, `Casdoore-mailService`, `CasdoorSmsService` 和 `CasdoorResourceService`

您可以按照下面示例在Gateway项目中创建它们。

```
@Resource  
private CasdoorAuthService casdoorAuthService;
```

当您需要访问您的应用程序的身份验证时，您可以发送目标 url 并重定向到 Casdoor 提供的登录页面。

请确保您已提前在应用配置中添加回调 url(例如 `http://localhost:9090/login`)。

```
@RequestMapping("login")  
public Mono<String> login() {  
    return Mono.just("redirect:" +  
        casdoorAuthService.getSignInUrl("http://localhost:9090/callback"));  
}
```

在 Casdoor 验证通过后，它将被重定向到您的应用程序，并带有 code 和状态。

您可以获取 code 并调用 `getOAuthToken` 方法，然后解析出 jwt 令牌。

`CasdoorUser` 包含了 Casdoor 提供的用户基本信息，您可以将其作为关键字在您的应用程序中设置会话。

```
@RequestMapping("callback")  
public Mono<String> callback(String code, String
```

API示例如下：

- CasdoorAuthService
  - `String token = casdoorAuthService.getOAuthToken(code, "app-built-in");`
  - `CasdoorUser casdoorUser = casdoorAuthService.parseJwtToken(token);`
- CasdoorUserService
  - `CasdoorUser casdoorUser = casdoorUserService.getUser("admin");`
  - `CasdoorUser casdoorUser = casdoorUserService.getUserByEmail("admin@example.com");`
  - `CasdoorUser[] casdoorUsers = casdoorUserService.getUsers();`
  - `CasdoorUser[] casdoorUsers = casdoorUserService.getSortedUsers("created_time", 5);`
  - `int count = casdoorUserService.getUserCount("0");`
  - `CasdoorResponse response = casdoorUserService.addUser(user);`
  - `CasdoorResponse response = casdoorUserService.updateUser(user);`
  - `CasdoorResponse response = casdoorUserService.deleteUser(user);`
- CasdoorEmailService
  - `CasdoorResponse response = casdoorEmailService.sendEmail(title, content, sender, receiver);`
- CasdoorSmsService
  - `CasdoorResponse response = casdoorSmsService.sendSms(randomCode(), receiver);`
- CasdoorResourceService
  - `CasdoorResponse response = casdoorResourceService.uploadResource(user, tag, parent,`

```
fullFilePath, file);  
◦ CasdoorResponse response =  
casdoorResourceService.deleteResource(file.getName());
```

## 第7步：重新启动项目

启动后，打开您最喜欢的浏览器并访问 <http://localhost:9090>，然后点击任何可以向 casdoor-api 请求资源的按钮



# Casdoor

Get Resource

Update Resource

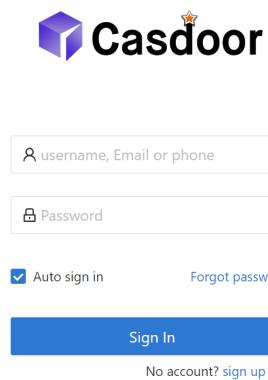
将触发网关认证逻辑。由于您没有登录，您将跳转到登录界面。点击登录按钮。



# Click to login

>Login

随后您可以看到Cassdoor统一的登录平台。



登录成功后，它将跳转到主界面。 然后您可以点击任意资源相关按钮。



# Casdoor

[Get Resource](#)

[Update Resource](#)

"success get resource1"

## 更多内容

您可以探索以下项目/文件来了解更多关于Java与Casdoor一体化的信息。

- [casdoor-java-sdk](#)
- [casdoor-spring-boot-starter](#)
- [casdoor-spring-boot-example](#)
- [casdoor-spring-security-example](#)
- [casdoor-spring-security-react-example](#)
- [casdoor-spring-boot-shiro-example](#)
- [casdoor-springcloud-gateway-example](#)

# Spring 安全

## Spring Security OAuth

这里我们将使用Spring Security 作为示例，向您展示如何将 OIDC 链接到您的应用程序。

## Spring Security Filter

基于Spring Security Filter，如何使用OIDC连接您的应用程序。

# Spring Security OAuth

Casdoor 可以使用 OIDC 协议作为IDP 连接各种应用程序。 这里我们将使用Spring Security作为示例，向您展示如何使用 OIDC 链接到您的应用程序。

## 步骤1. 部署Casdoor

首先，部署Casdoor。

您可以参考Casdoor 官方文档 [Server Installation](#)。

成功部署后，您需要确保：

- Casdoor服务已成功运行，能通过<http://localhost:8000> 访问。
- 打开您最喜欢的浏览器并访问 <http://localhost:7001>，您将看到Casdoor的登录页面。
- 输入 `admin` 和 `123` 测试登录功能正常工作。

然后您可以通过以下步骤在自己的应用程序中快速实现基于 Casdoor 的登录页面。

## 第2步： 配置Casdoor应用程序

1. 创建或使用现有的 Casdoor 应用程序。
2. 添加您的重定向url (您可以在下一节中看到更多关于如何获取重定向url的细节)

Client ID [?](#) : 24a25ea0714d92e78595 **Client ID**

Client secret [?](#) : 155... **Client Secret**

Redirect URLs [?](#) :  
Redirect URLs [Add](#)  
Redirect URL **Add a redirect URL for spring security**  
<http://localhost:8082/ui-one/login/oauth2/code/custom>

3. 添加您想要的提供商并补充其他设置。

不出意外的话，您会在应用程序设置页面看到：**Client ID** 和 **Client secret** 就像上面的图片一样。 我们将在下一步中使用它们。 您将在下一步中使用它们。

打开你喜欢的浏览器，访问：[http://CASDOOR\\_HOSTNAME/.well-known/openid-configuration](http://CASDOOR_HOSTNAME/.well-known/openid-configuration)，你会看到 Casdoor 的 OIDC 配置。

## 步骤3. 配置Spring Security

Spring Security 完全支持 OIDC。

您可以自定义 Spring Security OAuth2 客户端的设置：

### ⚠ 注意事项

您应该用您自己的 Casdoor 实例替换配置，特别是 **<Client ID>** 等。

[application.yml](#)    [application.properties](#)

```
spring:  
  security:  
    oauth2:
```

```
spring.security.oauth2.client.registration.casdoor.client-id=<Client ID>
spring.security.oauth2.client.registration.casdoor.client-secret=<Client Secret>
spring.security.oauth2.client.registration.casdoor.scope=<Scope>
spring.security.oauth2.client.registration.casdoor.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.casdoor.redirect-uri=<Redirect URL>

spring.security.oauth2.client.provider.casdoor.authorization-uri=http://CASDOOR_HOSTNAME:7001/login/oauth/authorize
spring.security.oauth2.client.provider.casdoor.token-uri=http://CASDOOR_HOSTNAME:8000/api/login/oauth/access_token
spring.security.oauth2.client.provider.casdoor.user-info-uri=http://CASDOOR_HOSTNAME:8000/api/get-account
spring.security.oauth2.client.provider.casdoor.user-name-attribute=name
```

#### ⚠ 注意事项

对于Spring Security的默认情况，<Redirect URL> 应该等于 http://<Your Spring Boot Application Endpoint>/<Servlet Prefix if it is configured>/login/oauth2/code/code。例如，对于下面的演示来说，重定向URL应该是 http://localhost:8080/login/oauth2/code/code/custom。您也应该在 casdoor 应用程序中配置它。

您也可以通过 ClientRegistration 在您的代码中自定义设置。您可以在这里找到映射

## 步骤4. 从一个 Demo 开始

1. 我们可以创建 Spring Boot 应用程序。
2. 我们可以添加一个配置，保护所有端点，除了 / 和 /login\*\* 用来给用户登录。

```
@EnableWebSecurity
public class UiSecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests().antMatchers("/", "/login**").permitAll().anyRequest().authenticated().and()
            .oauth2Login();

    }
}
```

3. 我们可以添加一个简单的页面供用户登录。

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Spring OAuth Client Thymeleaf - 1</title>
<link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/
bootstrap.min.css" />
</head>
<body>
    <nav
        class="navbar navbar-expand-lg navbar-light bg-light shadow-
sm p-3 mb-5">
        <a class="navbar-brand" th:href="@{/foos/}">Spring OAuth 客户
端
        Thymeleaf - 1</a>
    </nav>
    <div class="container">
        <label>欢迎！</label> <br /> <a th:href="@{/foos/}"
class="btn btn-primary">登陆</a>
    </div>
</body>
</html>
```

当用户点击 `登录` 按钮时，他将会被重定向到 `Casdoor`。

4. 接下来，我们可以定义我们受保护的资源。我们可以导出一个名为 `/foos` 的端点和一个用于显示的网页。

## Data Model

```
public class FooModel {  
    private Long id;  
    private String name;  
  
    public FooModel(Long id, String name) {  
        super();  
        this.id = id;  
        this.name = name;  
    }  
    public Long getId() {  
        return id;  
    }  
    public void setId(Long id) {  
        this.id = id;  
    }  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

## Controller

```
@Controller  
public class FooClientController {  
    @GetMapping("/foos")  
    public String getFoos(Model model) {
```

## Web page

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Spring OAuth Client Thymeleaf - 1</title>
<link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/
bootstrap.min.css" />
</head>
<body>
    <nav
        class="navbar navbar-expand-lg navbar-light bg-light shadow-
sm p-3 mb-5">
        <a class="navbar-brand" th:href="@{/foos/}">Spring OAuth
Client
            Thymeleaf -1</a>
        <ul class="navbar-nav ml-auto">
            <li class="navbar-text">Hi, <span
sec:authentication="name">preferred_username</span>&ampnbsp&ampnbsp&ampnbsp;
            </li>
        </ul>
    </nav>
    <div class="container">
        <h1>All Foos:</h1>
        <table class="table table-bordered table-striped">
            <thead>
                <tr>
                    <td>ID</td>
                    <td>Name</td>
                </tr>
            </thead>
            <tbody>
                <tr th:if="${foos.empty}">
                    <td colspan="4">No foos</td>
                </tr>
                <tr th:each="foo : ${foos}">
```

### ⚠ 注意事项

所有网页模板都应该放置在 `resources/templates` 下。

## 步骤5. 试用一下demo!

首先，您可以尝试打开您最喜欢的浏览器并直接访问 `/foos`。它将自动重定向到 Casdoor 的登录页面。您可以在这里或从根页面登录。

如果您访问了您的根页面，

Spring OAuth Client Thymeleaf - 1

Welcome !  
[Login](#)

点击 `登录` 按钮，页面将重定向到下级登录页面。



username, Email or phone

Password

Auto sign in      [Forgot password?](#)

[Sign In](#)

[Sign in with code](#)    No account? [sign up now](#)

登录后，页面将重定向到 </foos>。

Spring OAuth Client Thymeleaf -1

Hi, [user](#)

Your Username

### All Foos:

ID	Name
1	a
2	b
3	c

# Spring Security Filter

Casdoor 可以使用 OIDC 协议作为IDP 连接各种应用程序。下面我们将使用spring security中的过滤器来集成casdoor，并向您展示如何使用oidc连接到应用程序。

## 步骤1. 部署Casdoor

首先，部署Casdoor。

您可以参考Casdoor 官方文档 [Server Installation](#)。

成功部署后，您需要确保：

- Casdoor服务已成功运行，能通过<http://localhost:8000> 访问。
- 打开您最喜欢的浏览器并访问 <http://localhost:7001>，您将看到Casdoor的登录页面。
- 输入 `admin` 和 `123` 测试登录功能正常工作。

然后您可以通过以下步骤在自己的应用程序中快速实现基于 Casdoor 的登录页面。

## 第2步： 配置Casdoor应用程序

1. 创建或使用现有的 Casdoor 应用程序。
2. 添加您的重定向url (您可以在下一节中看到更多关于如何获取重定向url的细节)

Name ② : application\_a6ftas → your application name

Display name ② : New Application - a6ftas

Logo ② : URL ② : [https://cdn.casbin.org/img/casdoor-logo\\_1185x256.png](https://cdn.casbin.org/img/casdoor-logo_1185x256.png)

Preview: 

Home ② : [🔗](#)

Description ② :

Organization ② : organization\_carg1b → your organization name

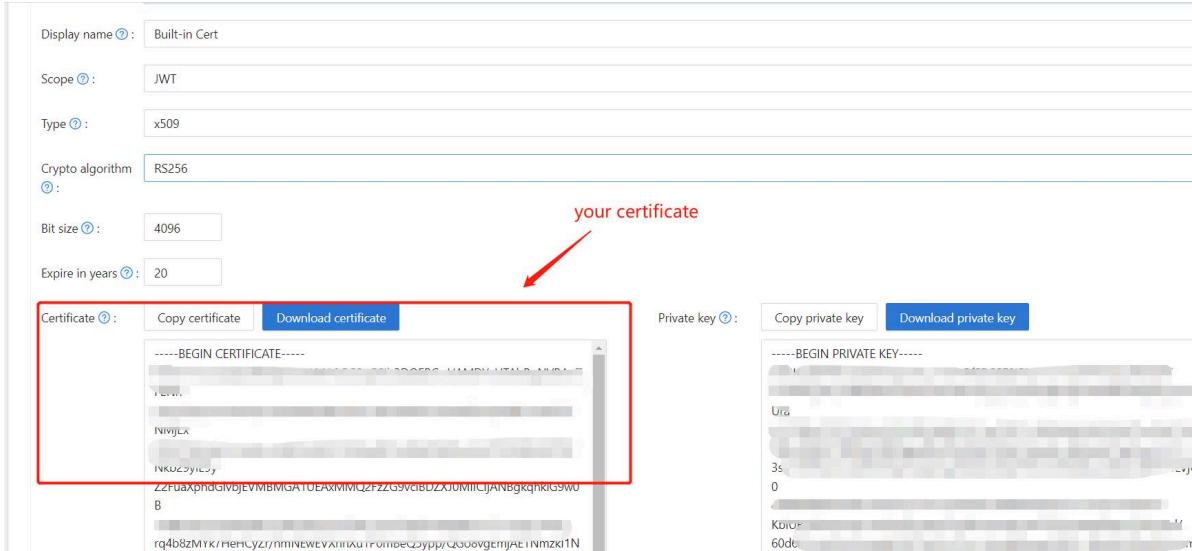
Client ID ② : 3ed7314825ecf955cb19 → your client id

Client secret ② : ee9314ea228... → your client secret

Cert ② : cert-built-in

Redirect URLs ② : Redirect URLs Add  
Redirect URL <http://localhost:3000/callback> → your redirect url

3. 在证书编辑页面上，您可以看到您的 [证书](#)。



#### 4. 添加您想要的提供商并补充其他设置。

不出意外的话，您会在应用程序设置页面看到： 应用程序名称，组织名称，重定向URL，客户端ID，客户密钥，认证。如上所述，我们将在下一步中使用它们。

打开你喜欢的浏览器，访问：[http://CASDOOR\\_HOSTNAME/.well-known/openid-configuration](http://CASDOOR_HOSTNAME/.well-known/openid-configuration)，你会看到 Casdoor 的 OIDC 配置。

## 步骤3. 配置Spring Security

您可以自定义spring security filter 的设置来处理标记：

### ⚠ 注意事项

您应该用您自己的 Casdoor 实例替换配置，特别是 <Client ID> 等。

**服务器：**

```
port: 8080
casdoor:
  endpoint: http://CASDOOR_HOSTNAME: 8000
  client-id: <Client ID>
```

### ⚠ 注意事项

对于前端应用程序来说，`<FRONTEND_HOSTNAME>` 的默认值是 `localhost:3000`。

例如，对于下面的演示来说，重定向URL应该是 `http://localhost:3000/callback`。

您也应该在 `casdoor` 应用程序中配置它。

## 步骤4. 配置前端

您需要安装 `casdoor-js-sdk` 并配置 `SDK`。

1. 安装 `casdoor-js-sdk`。

```
npm i casdoor-js-sdk
# or
yarn add casdoor-js-sdk
```

2. 设置 `SDK`。

```
import Sdk from "casdoor-js-sdk";

// Serverurl is the URL where spring security is deployed
export const ServerUrl = "http://BACKEND_HOSTNAME:8080";

const sdkConfig = {
  serverUrl: "http://CASDOOR_HOSTNAME:8000",
  clientId: "<your client id>",
  appName: "<your application name>",
  organizationName: "<your organization name>",
  redirectPath: "/callback",
```

# 步骤5. 从一个 Demo 开始

1. 我们可以创建 Spring Boot 应用程序。
2. 我们可以添加一些配置来处理 JWT。

```
@EnableWebSecurity
public class SecurityConfig {

    private final JwtTokenFilter jwtTokenFilter;

    public SecurityConfig(JwtTokenFilter jwtTokenFilter) {
        this.jwtTokenFilter = jwtTokenFilter;
    }

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http)
            throws Exception {
        // enable CORS and disable CSRF
        http = http.cors(corsConfig -> corsConfig
                .configurationSource(configurationSource())
                .csrf().disable());

        // set session management to stateless
        http = http
                .sessionManagement()
                .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                .and();

        // set permissions on endpoints
        http.authorizeHttpRequests(authorize -> authorize
                .mvcMatchers("/api/redirect-url", "/api/signin")
                .permitAll()
                .mvcMatchers("/api/**").authenticated()
        );
    }
}
```

3. 我们可以添加简单的 JWT 过滤器来拦截需要验证标记的请求。

```
@Component
public class JwtTokenFilter extends OncePerRequestFilter {

    private final CasdoorAuthService casdoorAuthService;

    public JwtTokenFilter(CasdoorAuthService casdoorAuthService) {
        this.casdoorAuthService = casdoorAuthService;
    }

    @Override
    protected void doFilterInternal(HttpServletRequest request,
                                    HttpServletResponse response,
                                    FilterChain chain)
        throws ServletException, IOException {
        // get authorization header and validate
        final String header =
request.getHeader(HttpHeaders.AUTHORIZATION);
        if (!StringUtils.hasText(header) ||
header.startsWith("Bearer ")) {
            chain.doFilter(request, response);
            return;
        }

        // get jwt token and validate
        final String token = header.split(" ")[1].trim();

        // get user identity and set it on the spring security
        context
        UserDetails userDetails = null;
        try {
            CasdoorUser casdoorUser =
casdoorAuthService.parseJwtToken(token);
            userDetails = new CustomUserDetails(casdoorUser);
        } catch (CasdoorAuthException exception) {
            logger.error("casdoor auth exception", exception);
            chain.doFilter(request, response);
            return;
        }
    }
}
```

当用户访问需要认证的接口时，`JwtTokenFilter` 将从请求头 `授权` 并验证它。

4. 接下来，我们需要定义一个 `控制器` 来处理以下情况：当用户登录到 `casdoor` 时，它将被重定向到服务器并携带 `code` 和 `state`。服务器会需要验证用户的身份，并通过这两个参数获得 `token`。

```
@RestController
public class UserController {

    private static final Logger logger =
    LoggerFactory.getLogger(UserController.class);

    private final CasdoorAuthService casdoorAuthService;

    // ...

    @PostMapping("/api/signin")
    public Result signin(@RequestParam("code") String code,
    @RequestParam("state") String state) {
        try {
            String token = casdoorAuthService.getOAuthToken(code,
state);
            return Result.success(token);
        } catch (CasdoorAuthException exception) {
            logger.error("casdoor auth exception", exception);
            return Result.failure(exception.getMessage());
        }
    }

    // ...
}
```

## 步骤6. 试用一下demo!

首先，您可以尝试通过浏览器访问前端应用程序。如果您尚未登录，它将显示登录按钮。

单击登录按钮，您将被重定向到 [级门](#) 登录页面。

如果您访问了您的根页面，

[Casdoor Login](#)

点击 [登录](#) 按钮，页面将重定向到下级登录页面。



username, Email or phone

Password

Auto sign in      [Forgot password?](#)

[Sign In](#)

No account? [sign up now](#)

Made with ❤ by [Casdoor](#)

登录后，页面将重定向到 [/](#)。



New User - rtsbx4

[Logout](#)

# Jenkins 插件

Casdoor 为用户登录Jenkins提供了一个插件。 这里我们将向您展示如何使用 Casdoor 插件来保证您的 Jenkins 安全。

以下是配置中的一些专有名词：

**CASDOOR\_HOSTNAME**: 私有部署的Casdoor域名或IP。

**JENKINS\_HOSTNAME**: 部署Jenkins的域名或IP。

## 第1步。 部署Casdoor和Jenkins

首先，应该部署 [Casdoor](#) 和 [Jenkins](#)。

在成功部署后，您需要确保：

- 将 Jenkins URL(管理 Jenkins → 配置系统 → Jenkins 位置) 设置为

The screenshot shows the Jenkins configuration interface. In the top navigation bar, 'Dashboard' and 'configuration' are visible. Under 'Jenkins Location', the 'Jenkins URL' field contains 'http://10.144.125.123:6780'. The 'JENKINS\_HOSTNAME' field below it contains 'address not configured yet <nobody@nowhere>'. There is also a note: 'Without a resource root URL, resources will be served from the Jenkins URL with Content-Security-Policy set.' At the bottom of the form are 'Save' and 'Apply' buttons.

2. 可以登录并正常使用Casdoor。
3. 将Casdoor 的 origin value (conf/app.conf) 设置为 CASDOOR\_HOSTNAME。

```
conf > ⚙ app.conf
 8  dbName = casdoor
 9  redisEndpoint =
10 defaultStorageProvider =
11 isCloudIntranet = false
12 authState = "casdoor"
13 httpProxy = "127.0.0.1:10808"
14 verificationCodeTimeout = 10
15 initScore = 2000
16 logPostOnly = true
17 origin = "http://10.144.1.2:8000" | CASDOOR_HOSTNAME
```

## 第2步： 配置Casdoor应用程序

1. 创建或使用现有的 Casdoor 应用程序。
2. 添加重定向URL: http://JENKINS\_HOSTNAME/securityRealm/finish登录

The screenshot shows the Casdoor application configuration interface. It displays the following fields:

- Description: Casdoor for Jenkins
- Organization: built-in
- Client ID: bbd0bd66696e504dec59
- Client secret: d2de01b01...110b47465c
- Redirect URLs:
  - Add button
  - Redirect URL: http://10.144.125.123:6780/securityRealm/finishLogin (highlighted with a red box)
  - Add a redirect url for Jenkins (text)
  - JENKINS\_HOSTNAME (text)

3. 添加您想要的提供商并补充其他设置。

不出意外的话，您可以在应用程序设置页面获得两个值: Client ID 和 Client

`secret` 就像上面的图片一样。 我们将在下一步骤中使用它们。

打开你喜欢的浏览器，访问：`http://CASDOOR_HOSTNAME/.well-known/openid-configuration`，你会看到 Casdoor 的 OIDC 配置。

## 第3步 配置 Jenkins

现在，您可以从市场安装Casdoor 插件，或者上传它的 `jar` 文件。

安装完成后，前往管理Jenkins → 配置全局安全性。

建议: 备份Jenkins `config.xml` 文件，并在设置错误时使用它来恢复。

The screenshot shows the Jenkins 'Configure Global Security' page under the 'Authentication' section. It is configured to use the 'Casdoor Authentication Plugin'. The 'Casdoor Endpoint' field is empty and highlighted in red with the error message 'Casdoor Endpoint is required.'. The 'Client ID' and 'Client Secret' fields are also empty and highlighted in red with the error message 'Client Id is required.' and 'Client Secret is required.' respectively. The 'JWT Public Key' field is empty and highlighted in red with the error message 'Jwt Public Key is required.'. At the bottom, there are 'Save' and 'Apply' buttons, and an 'Advanced...' link.

1. 在 Security 中，选择“Casdoor Authentication Plugin”。
2. 在 Cassdoor Endpoint中，指定上面提到的 `CASDOOR_HOSTNAME`
3. 在客户端ID中，指定上面注明的 `客户端ID`。
4. 在客户端secret中，指定上面注明的 `client secret`

5. 在 JWT 公钥中，指定用于验证 JWT 令牌的公钥。您可以通过点击顶部的 Cert 在 Casdoor 找到公钥。点击 编辑 应用程序后，您可以在下面的页面复制您的公钥。

The screenshot shows the 'Edit Cert' page in Casdoor. At the top, there are tabs for Home, Organizations, Users, Roles, Permissions, Providers, Applications, Resources, Tokens, Records, Webhooks, Syncers, Certs (which is highlighted with a red box and arrow labeled '1'), and Swagger. Below the tabs, there are fields for Name (cert-built-in), Display name (Built-in Cert), Scope (JWT), Type (x509), Crypto algorithm (RSA), Bit size (4096), and Expire in years (20). Under the Public key section, there are two buttons: 'Copy public key' (highlighted with a red box and arrow labeled '2') and 'Download public key'. To the right, there are sections for Private key and Download private key. The public key area contains placeholder text: '-----BEGIN CERTIFICATE-----' followed by a large gray pixelated area.

6. 组织名称和应用程序名称是可选的。您可以指定您的组织和应用程序来验证其他组织和应用程序中的用户。如果他们为空，插件将使用默认的组织和应用程序。
7. 在授权部分，检查“Logged-in users can do anything”。禁用"Allow anonymous read access"。
8. 点击 Ok

现在，Jenkins 会自动将您重定向到 Casdoor 进行认证。

# Jenkins OIDC

Casdoor 可以使用 OIDC 协议作为IDP 连接各种应用程序。这里我们将使用Jenkins作为示例，向您展示如何使用 OIDC 链接到您的应用程序。

以下是配置中的一些专有名词：

`CASDOOR_HOSTNAME`: 私有部署的Casdoor域名或IP。

`JENKINS_HOSTNAME`: 部署Jenkins的域名或IP。

## 第1步。部署Casdoor和Jenkins

首先，应该部署 Casdoor 和 Jenkins。

在成功部署后，您需要确保：

- 将 Jenkins URL(Manage Jenkins → Configure System → Jenkins Location) 设置为 `JENKINS_HOSTNAME`

The screenshot shows the Jenkins configuration interface under 'Dashboard > configuration'. In the 'Jenkins Location' section, the 'Jenkins URL' field contains the value 'http://10.144.125.123:6780', which is highlighted with a red box. Below it, the 'System Admin e-mail address' field contains 'address not configured yet <nobody@nowhere>'. There are also sections for 'Serve resource files from another domain' and 'Global properties', both of which are currently empty. At the bottom, there are 'Save' and 'Apply' buttons.

2. 可以登录并正常使用Casdoor。
3. 将Casdoor 的 origin value (conf/app.conf) 设置为 CASDOOR\_HOSTNAME。

```
conf > ⚙ app.conf
 8  dbName = casdoor
 9  redisEndpoint =
10 defaultStorageProvider =
11 isCloudIntranet = false
12 authState = "casdoor"
13 httpProxy = "127.0.0.1:10808"
14 verificationCodeTimeout = 10
15 initScore = 2000
16 logPostOnly = true
17 origin = "http://10.144.1.2:8000" | CASDOOR_HOSTNAME
```

## 第2步： 配置Casdoor应用程序

1. 创建或使用现有的 Casdoor 应用程序。
2. 添加重定向URL: [http://JENKINS\\_HOSTNAME/securityRealm/finish](http://JENKINS_HOSTNAME/securityRealm/finish) 登录

The screenshot shows the Casdoor application configuration interface. It includes fields for Client ID (b7bd0bd66696e504dec59), Client secret (d2de01b01...110b47465c), and a Redirect URLs section where a URL (http://10.144.125.123:6780/securityRealm/finishLogin) is listed under JENKINS\_HOSTNAME.

Description	Casdoor for Jenkins
Organization	built-in
Client ID	b7bd0bd66696e504dec59
Client secret	d2de01b01...110b47465c
Redirect URLs	Redirect URLs Add Redirect URL http://10.144.125.123:6780/securityRealm/finishLogin Add a redirect url for Jenkins JENKINS_HOSTNAME

3. 添加您想要的provider并补充其他设置。

不出意外的话，您会在应用程序设置页面看到: Client ID 和 Client secret 就像

上面的图片一样。 我们将在下一步中使用它们。

打开你喜欢的浏览器，访问：`http://CASDOOR_HOSTNAME/.well-known/openid-configuration`，你会看到 Casdoor 的 OIDC 配置。

## 第3步 配置 Jenkins

Jenkins本质上不支持OIDC，所以我们需要安装 [OpenId Connect Authentication](#)。

安装完成后，前往Manage Jenkins → 配置全局安全性。

**建议:** 备份Jenkins `config.xml` 文件，并在设置错误时使用它来恢复。

1. 在访问控制中，Security Realm选择“Login with Openid Connect”。
2. 在客户端ID中，指定上面注明的 `客户端ID`。
3. 在Client secret中，指定上面注明的 `Client secret`
4. 在配置模式下，选择 `Automatic configuration` 并填写

`http://CASDOOR_HOSTNAME/.well-known/openid-configuration` 到 Well-known 配置端点。

The screenshot shows the Jenkins Global Security configuration page under the "Security Realms" section. It includes fields for Client ID (with placeholder "Input your Client ID") and Client secret (with placeholder "Input your Client secret" and a "Concealed" button). Below this is the "Configuration mode" section, where "Automatic configuration" is selected and the "Well-known configuration endpoint" field contains the URL `http://10.144.1.2:8000/.well-known/openid-configuration`. A red arrow points to the "Select this" link next to the "Login with Openid Connect" radio button.

如果您的Casdoor是本地部署的，您需要选择 `Manual configuration` 并输入一些信息：

- Token server url: http://**CASDOOR\_HOSTNAME**/api/login/oauth/access\_token
- Authorization server url: http://**CASDOOR\_HOSTNAME**/login/oauth/authorize
- UserInfo server url: http://**CASDOOR\_HOSTNAME**/api/get-account
- Scopes: address phone openid profile offline\_access email  
Configuration mode
  - Automatic configuration
  - Manual configuration

Token server url	http://10.144.1.2:8000/api/login/oauth/access_token	?
<b>CASDOOR_HOSTNAME</b>	Authorization server url	?
UserInfo server url	http://10.144.1.2:8000/login/oauth/authorize	?
Scopes	address phone openid profile offline_access email	?

5. 点击高级设置，填写如下：

- 在用户名字段中，指定 **data.name**
- 在用户名字段中，指定 **data.displayName**
- 在电子邮件字段中，指定 **data.email**

User name field name	data.name
Full name field name	data.displayName
Email field name	data.email
Groups field name	?
Token Field Key To Check	?
Token Field Value To Check	?

6. 在授权部分，检查“Logged-in users can do anything”。禁用“Allow anonymous

read access"。 您可以稍后配置更复杂的授权。现在检查OpenID是否是正常工作状态。

注销Jenkins，现在它应该将您重定向到 Casdoor 进行身份验证。



The screenshot shows the Casdoor login interface. At the top is a large Jenkins logo. Below it are two input fields: one for 'username, Email or phone' with a magnifying glass icon, and another for 'Password' with a lock icon. Underneath these fields are two small links: 'Auto sign in' with a checked checkbox and 'Forgot password?'. A large blue 'Sign In' button is centered below the password field. At the bottom of the form are two smaller links: 'Sign in with code' and 'No account? sign up now'. Below the form is a horizontal line with two social login icons: a black Q and a multi-colored G (representing Google).

# RuoYi

Casdoor可以轻易连接到 RuoYi-cloud。

## 步骤1. 部署Casdoor

首先，部署Casdoor。

您可以参考Casdoor 官方文档 [Server Installation](#)。

在成功部署后，您需要确保：

- Casdoor服务已经成功运行，能通过<http://localhost:8000> 访问。
- 打开您最喜欢的浏览器并访问 <http://localhost:7001>，您将看到Casdoor的登录页面。
- 输入 `admin` 和 `123` 测试登录功能正常工作。

然后您可以通过以下步骤在自己的应用程序中快速实现基于 Casdoor 的登录页面。

## 步骤2. 配置Casdoor

如何配置casdoor可以参考 [casdoor](#)(配置casdoor和开发最好使用不同的浏览器)。

您还应该配置一个组织、应用程序和同步器。您也可以参考 [Casdoor](#)。

需要注意的一些要点：

1. 修改同步器表格中的列

Table columns ⓘ	Add	Column name	Column type	Casdoor column	Is hashed	Action
user_id	integer	Id	✓	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
dept_id	integer	Affiliation	✓	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
user_name	string	Name	✓	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
nick_name	string	DisplayName	✓	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
user_type	string	Type	✓	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
email	string	Email	✓	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
phonenumber	string	Phone	✓	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
sex	string	Gender	✓	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
avatar	string	Avatar	✓	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
password	string	Password	✓	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
del_flag	string	IsDeleted	✓	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
login_ip	string	CreatedIp	✓	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
create_time	string	CreatedTime	✓	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
password	string	Password	✓	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

## 2. 修改组织中的密码类型

Password type ⓘ :

## 3. 您也应该打开软删除。

# 步骤3. 前端改造

## 3.1 跳转到casdoor的登录页

我们可以以前端的sdk，这里我们以vue-sdk为例。在加入vue-sdk后，您可以通过getSignInUrl() 获得casdoor登录页面

您可以以您喜欢的方式链接到它，并且您可以删除一些不再使用的ruoyi-cloud的代码，如账户和密码EL输入框。

## 3.2 接收casdoor返回的code和state

在我们通过casdoor成功登录后，casdoor会把code和state返回给我们创建的页面。我们可以通过create函数获取code和state。

```
created() {
  let url = window.document.location.href // get url
  let u = new URL(url);
```

对于RuoYi-Cloud，我们只是将发送的账户和密码更改为code和state。因此，相对于原始登录，它只是改变发送到后端的内容。

## 步骤4. 后端改造

### 4.1 接收前端返回的code和state

```
@PostMapping("login")
public R<?> callback(@RequestBody CodeBody code) { //we should define
    a CodeBody entity which have code and state
    String token = casdoorAuthService.getAuthToken(code.getCode(),
        code.getState());
    CasdoorUser casdoorUser =
        casdoorAuthService.parseJwtToken(token);
    if(casdoorUser.getName()!=null){
        String casdoorUserName = casdoorUser.getName();

        if(sysLoginService.getUserByCasdoorName(casdoorUserName)==null){ //if
            database haven't this user
            // add this user into database
            sysLoginService.casdoorRegister(casdoorUserName);
        }
    }
    LoginUser userInfo =
        sysLoginService.casdoorLogin(casdoorUser.getName()); //get this
        user's information by database
    return R.ok(tokenService.createToken(userInfo));
}
```

在这种方法中，我们使用 casdoor-SpringBoot-sdk，并对 RuoYi-Cloud 方法稍做修改。

例如，RuoYi-Cloud原本的register使用账户和密码注册，我将register改成了casdoorRegister。

我还添加了一个getUserByCasdoorname这样的方法来检查账户是否存在，并将基于账户

和密码用户切换成了当前用户。

这很简单，因为我们只需要删除检查密码的部分。

## 步骤5. 总结

### 5.1 前端

- 我们需要删除原始的登录和注册功能。
- 我们还需要接受code和state并发送到后端。

### 5.2 后端

RuoYi 后端具有极好的登录和注册功能。 我们只需要稍做改变，因此非常方便。

## 步骤6. 详细步骤

1. 部署和配置Casdoor。 部署并配置casdoor。我们必须注意组织的密码类型必须为bcrypt，因为 RuoYi-Cloud的密码类型是bcrypt。
2. 我们应该使用casdoor的同步器将数据库用户中的用户信息复制到您的casdoor组织。此步骤可以将原始帐户导入到casdoor。
3. 在我们部署了casdoor之后，我们应当改造前端。 我们应该关闭 RuoYi 的验证码功能

```
// checkcode switch  
captchaEnabled: false,  
// register switch  
register: true,
```

请注意，在 nacos 中也要关闭RuoYi-Cloud验证码功能。 请注意，RuoYi-Cloud开放注册功能还需要更改配置 sys.account.registerUser 的值为true。

4. 我们应该添加按钮跳到casdoor的按钮并更改数据的loginForm



```
1 <button>
2   <a href="http://localhost:7001/login/oauth/authorize?client_id=d509b6b3edc8a3d4cce9&response_type=code&redirect_uri=http%3A%2F%2Flocalhost%3D8080%2Fcasdoor">casdoor</a>
3 </button>
```

```
1 loginForm: {
2   code: '',
3   state: ''
4 },
```

你可以通过casdoor-vue-sdk或casdoor-SpringBoot-sdk获得这里的网址。

5. 由于我们不使用原先的登录功能，我们应该删除关于 cookie 和校验码的方法。

所以创建新的函数：

```
created() {
  let url = window.document.location.href//get url
  let u = new URL(url);
  this.loginForm.code = u.searchParams.get('code')//get code and
state
  this.loginForm.state = u.searchParams.get('state')
  if(this.loginForm.code!=null&&this.loginForm.state!=null){//if
code and state is null, execute handleLogin
    this.handleLogin()
  }
}
```

6. 事实上，我们只需要更改发送到后端的参数并删除不需要的函数，此外无需其他改动。

```
handleLogin() {
  console.log("进入handleLogin")
  this.$store.dispatch("Login", this.loginForm).then(() => {
    this.$router.push({ path: this.redirect || "/" }).catch(()=> {});
  }).catch(() => {
    this.loading = false;
    if (this.captchaEnabled) {
      this.getCode();
      console.log(this.getCode)
    }
  });
}
```

```
Login({ commit }, userInfo) {
  const code = userInfo.code
  const state = userInfo.state
  return new Promise((resolve, reject) => {
    login(code, state).then(res => {
      console.log("LOGIN")
      let data = res.data
      setToken(data.access_token)
      commit('SET_TOKEN', data.access_token)
      setExpiresIn(data.expires_in)
      commit('SET_EXPIRES_IN', data.expires_in)
      resolve()
    }).catch(error => {
      reject(error)
    })
  })
},
```

```
export function login(code, state) {
  return request({
    url: '/auth/login',
    headers: {
      isToken: false
    },
    method: 'post',
    data: {code, state}
  })
}
```

## 7. 在后端导入依赖

pom.xml

```
<dependency>
    <groupId>org.casbin</groupId>
    <artifactId>casdoor-spring-boot-starter</artifactId>
    <version>1.2.0</version>
</dependency>
```

您还需要在代码上配置casdoor。 8. 回调函数被定义为重定向函数。 我对 sysLoginService 中的一些方法做了小修改 删除检查密码步骤，因为我们不需要它。

```
@PostMapping("login")
public R<?> callback(@RequestBody CodeBody code) {//we should define
a CodeBody entity which have code and state
    String token = casdoorAuthService.getOAuthToken(code.getCode(),
code.getState());
    CasdoorUser casdoorUser =
casdoorAuthService.parseJwtToken(token);
    if(casdoorUser.getName()!=null){
        String casdoorUserName = casdoorUser.getName();

        if(sysLoginService.getUserByCasdoorName(casdoorUserName)==null){//if
database haven't this user
            // add this user into database
            sysLoginService.casdoorRegister(casdoorUserName);
        }
    }
    LoginUser userInfo =
sysLoginService.casdoorLogin(casdoorUser.getName());//get this
user's information by database
    return R.ok(tokenService.createToken(userInfo));
}
```

9. SysLoginService的新方法

```
public LoginUser casdoorLogin(String username){
    // execute user
    R<LoginUser> userResult =
remoteUserService.getUserInfo(username, SecurityConstants.INNER);
    if (R.FAIL == userResult.getCode())
    {
        throw new ServiceException(userResult.getMsg());
    }

    if (StringUtils.isNull(userResult) ||
StringUtils.isNull(userResult.getData()))
    {
        recordLogService.recordLogininfor(username,
Constants.LOGIN_FAIL, "this user is not exist");
        throw new ServiceException("user: " + username + " is not
exist");
    }
    LoginUser userInfo = userResult.getData();
    SysUser user = userResult.getData().getSysUser();
    if (UserStatus.DELETED.getCode().equals(user.getDelFlag()))
    {
        recordLogService.recordLogininfor(username,
Constants.LOGIN_FAIL, "sorry, your account was deleted");
        throw new ServiceException("sorry, your account: " +
username + " was deleted");
    }
    if (UserStatus.DISABLE.getCode().equals(user.getStatus()))
    {
        recordLogService.recordLogininfor(username,
Constants.LOGIN_FAIL, "your account is disabled, you can contact
admin ");
        throw new ServiceException("sorry, your account: " +
username + " is disabled");
    }
    recordLogService.recordLogininfor(username,
Constants.LOGIN_SUCCESS, "login successfully");
    return userInfo;
}
```

```
public String getUserByCasdoorName(String casdoorUsername){
    R<LoginUser> userResult =
remoteUserService.getUserInfo(casdoorUsername,
SecurityConstants.INNER);
    if (StringUtils.isNull(userResult) ||
StringUtils.isNull(userResult.getData()))
    {
        //if this user is not in Ruoyi-Cloud database and casdoor
have this user, we should create this user in database
        return null;
    }
    String username =
userResult.getData().getSysUser().getUserName();
    return username;
}
```

```
public void casdoorRegister(String username){
    if (StringUtils.isAnyBlank(username))
    {
        throw new ServiceException("User must fill in");
    }
    SysUser sysUser = new SysUser();
    sysUser.setUserName(username);
    sysUser.setNickName(username);
    R<?> registerResult =
remoteUserService.registerUserInfo(sysUser, SecurityConstants.INNER);
    System.out.println(registerResult);
    if (R.FAIL == registerResult.getCode())
    {
        throw new ServiceException(registerResult.getMsg());
    }
    recordLogService.recordLoginInfo(username, Constants.REGISTER,
"register successfully");
}
```

# Pulsar-manager

Casdoor可以轻易连接到Pulsar-manager。

因为 Pulsar-manager中已经添加了连接casdoor 的代码，所以我们只需要在后端的 application.yml中做一些配置，并在前端打开一个开关。

## 步骤1. 部署Casdoor

首先，部署Casdoor。

您可以参考Casdoor 官方文档 [Server Installation](#)。

在成功部署后，您需要确保：

- Casdoor服务已经成功运行，能通过<http://localhost:8000> 访问。
- 打开您最喜欢的浏览器并访问 <http://localhost:7001>，您将看到Casdoor的登录页面。
- 输入 `admin` 和 `123` 测试登录功能正常工作。

然后您可以通过以下步骤在自己的应用程序中快速实现基于 Casdoor 的登录页面。

## 步骤2. 配置Casdoor

如何配置casdoor可以参考 [casdoor](#)(配置casdoor和开发最好使用不同的浏览器)。

您还应该配置一个组织和一个应用程序。您也可以参考 [Casdoor](#)。

## 步骤2.1 创建一个组织

Edit Organization Save Save & Exit

Name ⓘ: pulsar

Display name ⓘ: pulsar

Favicon ⓘ: URL ⓘ: <https://cdn.casbin.org/img/favicon.png>

Preview: 

Website URL ⓘ: <http://localhost:9527/#/login?redirect=%2F>

Password type ⓘ: plain

Password salt ⓘ:

Phone prefix ⓘ: + 86

## 步骤2.2 创建一个应用程序

Name ⓘ: app-pulsar

Display name ⓘ: app-pulsar

Logo ⓘ: URL ⓘ: [https://cdn.casbin.org/img/casdoor-logo\\_1185x256.png](https://cdn.casbin.org/img/casdoor-logo_1185x256.png)

Preview: 

Home ⓘ: [/](#)

Description ⓘ:

Organization ⓘ: pulsar

Client ID ⓘ: 6ba06c1e1a30929fdda

Client secret ⓘ: df926bf913225ebbae9af7ba8d41fe19507eb079

Cert ⓘ: cert-built-in

Redirect URLs ⓘ: Add

Redirect URLs	Action
<a href="http://localhost:9527/callback">http://localhost:9527/callback</a>	  

## 步骤3. 打开 Pulsar-Manager 前端中的开关。

打开此开关，将代码和状态发送到后端。

这个开关在pulsar-manager/front-end/src/router/index.js的第80行

```
- // mode: 'history', // require service support
+ mode: 'history', // require service support
```

## 步骤4. 后端配置

您需要在pulsar-manager/src/main/resources/application.properties的154行加入casdoor的配置

```
casdoor.endpoint = http://localhost:8000
casdoor.clientId = <client id in previous step>
casdoor.clientSecret = <client Secret in previous step>
casdoor.certificate=<client certificate in previous step>
casdoor.organizationName=pulsar
casdoor.applicationName=app-pulsar
```

# ShenYu

ShenYu 可通过插件使用 casdoor。

## 步骤1. 部署Casdoor

首先，部署Casdoor。

您可以参考Casdoor 官方文档 [Server Installation](#)。

成功部署后，您需要确保：

- Casdoor服务已成功运行，能通过<http://localhost:8000> 访问。
- 打开您最喜欢的浏览器并访问 <http://localhost:7001>，您将看到Casdoor的登录页面。
- 输入 `admin` 和 `123` 测试登录功能正常工作。

然后您可以通过以下步骤在自己的应用程序中快速实现基于 Casdoor 的登录页面。

## 步骤2. 配置Casdoor应用程序

1. 创建或使用现有的 Casdoor 应用程序。
2. 添加您的重定向url

Name : app-test → application name

Display name : app-test

Logo : 

Preview:

Home : →

Description :

Organization : built-in → organization name

Client ID : 6e3a84154e73d1fb156a → client id

Client secret : 84209d412a338a42b789c05a3446e623cb7262d → client secret

Cert : cert-built-in

Redirect URLs : Add

Redirect URL	Action
<a href="http://localhost:9195/http/hello">http://localhost:9195/http/hello</a> <span style="color:red">→ redirect url</span>	<span style="color:red">[Delete]</span>

3. 在证书编辑页面上，您可以看到您的 **证书**。

Certificate : Copy certificate Download certificate

```
-----BEGIN CERTIFICATE-----
MIIE+TCCAUgGAwIBAgIDAeJAMA0GCSqGSIb3DQEBCwUAMDYxHTABBgNVBAoTFENh
c2Rvb3IgT3JnYW5pemF0aW9uMRUwEwYDVQQDEwxDYXNkb29yIENlcnQwHhcNMjEx
MDE1MDgxMTUyWhcNNDExMDE1MDgxMTUyWjA2M0R0wGwYDVQQKExRDYXNkb29yIe9y
Z2FuaxphdGlvbjEVMBMGAG1UEAxMMQ2FzZG9vcj8DZXJ0MIIcjANBgkqhkiG9w0B
AQEFAAOCAg8AMIICCgKCAgEAisInpb5E1/ymf0f1RfSDSSE8I7y+lw+RjIj74e5ej
rq4b8zMMyk7HeHCyZr/hmNewEVXnhXu1P0mBeQ5ypp/QGo8vgEmjAETNmzk1NjOQ
CjCYwUrasO/f/Mnl1C0j13vx6mV1kHZjSrKsMhYY1vaxTEP3+VB8Hjg3MHFWrb07
uvFMCJe5W8+0rKErZCKTR8+9VB3janeBz/zQePFvh79bfZate/hLirPK0Go9P1g
OvwloC1A3sarHTP4Qm/LQRt0rHqZFybdySpyWAQvhNaDFE7mTstRS8b/wUjNCUBD
PTSLVjc04WIIIf6Nkfx0Z7KvmbPstSj+btvqcsvRAGtvsB9h62Kptjs1Yn7GAuo
I3qt/4zoKbiURYxkQJXlvwCQsEftUuk5ew5zuPSIDRLoLByQTLbx0jqLAFNfW3g/
pzSDjgd/60d6HTmvbZni4SmjdyFhXCDb1Kn7N+xTojnfaNkwep2REV+RMc0fx4Gu
hRsnlsmkmUDeylZ9aBL9oj11YEQfM2JZEq+RvtUx+wB4y8K/tD1bcY+lfnG5rBpw
IDpS262boq4SRsvb3Z7bB0w4ZxvOfj/1VLoRftPbLifobhfr/AeZMHpiKOXvfz4
yE+hqzi6wdF0VR9xYc/RbSAf7323OsjYnjjEglnUtRohnRgCpjlk/Mt2Kt84Kb0
wn8CAwEAAaMQMA4wDAYDVR0TAQH/BAlwADANBgkqhkiG9w0BAQsFAAOCAgEAn2lf
DKkLX+F1vKRO/5gJ+Plr8P5NKuQkmwH97b8CS2gS1phDyNgic4/LSdzuf4Awe6ve
C06lVdWSlis8UPUPdjmt2uMPSNjwLxG3QsrimMURNlwFLTfRem/heJe0Zgur9J1M
8haawdSdjH2RgmFoDeE2r8NVRfhbR8KnCO1ddTJKuS1N0/irHz21W4jt4rxzCvl
2nR42Fybap3O/g2JXMhNNRowZmNjgpsF7XVENCSuFO1jTywLaqjuXCg54lL7XVLG
omKNNNcc8h1FCelj/nbbGMhodnFWKDTsJcbNmcOPNHo6ixzqMy/Hqc+mWv7maAG
Jtevs3qgMZ8F9Qzr3HpUc6R3ZYWDY/xxPisuKftOPZgtH979XC4mdf0WPnOBLql
2DJ1zaBmjIGolvb7XNVKcUfdXYw85ZTQ5b9cl4e+6bmyWqQltlwt+Ati/uFEV
Xzcj70B4IALX6xau1kLepV9O1GERizYrz5P9NJNA7Ko05AVMp9w0DQTkt+LbXnZE
HHnWKy8xHQKF9sR7YBPGLs/Ac6tviv5ua15Ogj/8dLRZ/veyFfGo2yZsl+hKVU5
nCCJHBcAyFnm1hdvdwEdH33jDBjNB6ciotJZrf/3VYalWSalADosHAgMWfxUWP+h
8XKXmzlxuHbTMQYtZPDgsps5aK+S4Q9wb8RRAYo=
-----END CERTIFICATE-----
```

## 步骤3. 在 shenyu 中使用casdoor插件

### 1. 在 shenyu 中配置casdoor插件

## Plugin

X

\* Plugin: casdoor

### casdoor Configuration

\* application-name: app-test

\* certificate: -----BEGIN CERTIFICATE-----\nMIIE+TCCAuGgAwI

\* client\_id: 6e3a84154e73d1fb156a

\* client\_secrect: a4209d412a33a842b7a9c05a3446e623cbb7262d

\* casdoor endpoint: http://localhost:8000

\* organization-name: test

\* Role: Authentication

\* Sort: 40

Status:

Cancel

Sure

注意：由于shenyu 仅有单行输入框，所以我们需要在每行证书中添加 \n

Certificate  :

[Copy certificate](#)

[Download certificate](#)

```
-----BEGIN CERTIFICATE-----\nMIIE+TCCAuGgAwIBAgIDAeJAMA0GCSqGSIb3DQEBCwUAMDYxHTAbBgNVBAoTFENh\n\nc2Rvb3IgT3JnYW5pemF0aW9uMRUwEwYDVQQDEwxDYXNkb29yIENIcnQwHhcNMjEx\n\nc2Rvb3IgT3JnYW5pemF0aW9uMRUwEwYDVQQDEwxDYXNkb29yIENIcnQwHhcNMjEx\nMDE1MDgxMTUyWhcNNDEzMDE1MDgxMTUyWjA2MR0wGwYDVQQKExRDYXNkb29yIE9y\nZ2FuaXphdGlvbjEVMBMGA1UEAxMMQ2FzZG9vcIBDZXJ0MIICljANBqkqhkiG9w0B\nAQEFAOCAG8AMIICgKCAgEAsInpb5E1/yM0f1RfSDSSE8IR7y+lw+RJjl74e5ej\nrq4b8zMYk7HeHCyZr/hmNEwEVXnhXu1P0mBeQ5yp/PGo8vgEmjaETNmzkl1NjOQ\nCjCYwUrasO/f/Mnl1C0j13vx6mV1kHZjSrKsMhYY1vaxTEP3+VB8Hjg3MHFWrb07\nuvFMCje5W8+0rKErZCKTR8+9VB3janeBz//zQePFVh79bFZate/hLirPK0Go9P1g\nOvwloC1A3sarHTP4Qm/LQRt0rHqZFybdySpyWAQvhNaDFE7mTstRSBb/wUjNCUBD\nPTSLVjC04WIISf6Nkfx0Z7KvmbPstSj+btvcqsvRAGtvdsB9h62Kptjs1Yn7GAuo\nl3qt/4zoKbiURYxkQJXlvwCQsEftUuk5ew5zuPSIDRLoLByQTLbx0jqLAFnfW3g/\npzSDjgd/60d6HTmvbZni4SmjdyFhXCDb1Kn7N+xTojnfaNkwep2REV+RMcdfx4Gu\nhRsnLsmkmUDeylZ9aBL9oj11YEQfM2JZEq+RVtUx+wB4y8K/tD1bcY+lfnG5rBpw\nIDpS262boq4SRVb3Z7b0w4ZxvOfj/1VLoRftjPbLlf0bhfr/AeZMHplKOXvfz4\nyE+hqzi68wdF0VR9xYc/RbSAf7323OsjYnjEgInUtRohnRgCpjlk/Mt2Kt84Kb0\nwn8CAwEAQMA4wDAYDVR0TAQH/BAIwADANBqkqhkiG9w0BAQsFAAOCAgEAn2If\nDKkLX+F1vKRO/5gJ+Plr8P5NKuQkmwH97b8CS2gS1phDyNgIc4/Lsdzuf4Awe6ve\nC06IVdWSlis8UPUPdjmT2uMPSNjwLxG3QsrimMURNwFILTfRem/heJe0Zgur9J1M\n8haawdSdjH2RgmFoDeE2r8NVRfhbR8KnCO1ddTJKuS1N0/irHz21W4jt4rxzCvl\n2nR42Fyap3O/g2JXMhNNROwZmNjgpsF7XVENCSuFO1jTywLaqjuXCg54IL7XVLG\nomKNNNcc8h1FCeKj/nnbGMhodnFWKDTsJcbNmcoPNHo6ixzqMy/Hqc+mWYv7maAG\nJtevs3qgMZ8F9Qzr3HpUc6R3ZYWDY/xxPisuKftOPZgtH979XC4mdf0WPnOBLql\n2DJ1zaBmjIGjolvb7XNVKcUfDXYw85TZQ5b9cl4e+6bmyWqQltlw+Ati/uFEV\nXzCj70B4IALX6xau1kLEpV9O1GERizYRz5P9NJNA7KoO5AVMp9w0DQTkt+LbXnZE\nHHnWKy8xHQKZF9sR7YBPGLs/Ac6tiv5ua15OgJ/8dLRZ/veyFfGo2yZsl+hKVU5\nnCCJHBcAyFnm1hdvdwEdH33jDBjNB6ciotJZrf/3VYalWSaiADosHAgMWfXuWP+h\n8XKXmzlxuHbTMQYtZPDgspS5aK+S4Q9wb8RRAYo=\n-----END CERTIFICATE-----
```

 here not need add \n

你可以复制并粘贴到Shenyu 后台配置的证书。

**您不需要将其保存到casdoor证书编辑页面，因为只需复制即可。**

## 2. 配置shenyu casdoor的插件

The screenshot shows the Apache ShenYu Gateway Management System interface. On the left, there is a sidebar with 'Change Mode' (radio button), 'PluginList' (selected), and a list of plugins: Mock, Cache, Authentication (Sign, Jwt), and Casdoor (selected). The main area has tabs for 'SelectorList' and 'RulesList'. Under 'RulesList', there is a sub-tab 'Synchronous casdoor'. A table titled 'RuleName' lists a single rule: '/http/' with 'Open' status and 'Modify Delete' operations. The table includes columns for RuleName, Open, UpdateTime, and Operation. Navigation buttons at the bottom show page 1 of 12.

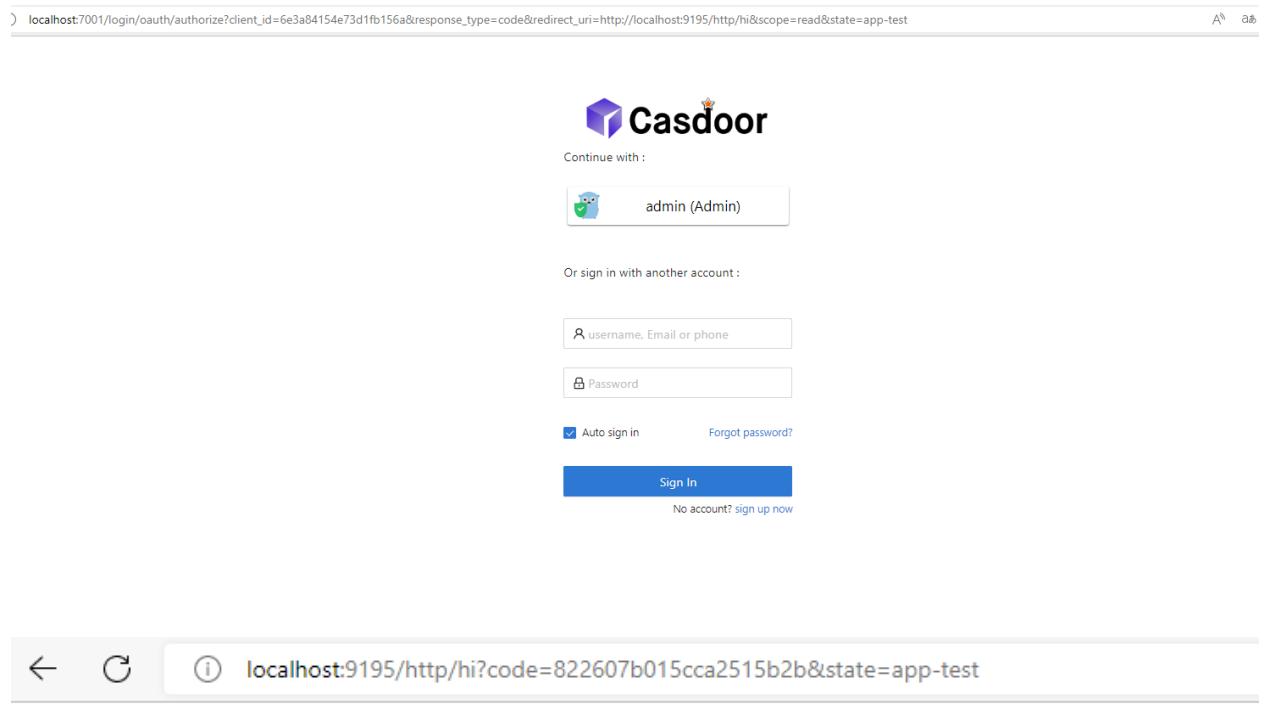
您可以配置您需要使用casdoor配置

## 3. 使用Casdoor提供的相关服务

### 3.1 直接访问Web

The screenshot shows a browser window with the address bar containing 'localhost:9195/http/hi'. Below the address bar, an error message is displayed: {"code":401, "message":"Illegal authorization"}

### 3.2像这样登录casdoor



The screenshot shows the Casdoor login interface. At the top, there is a URL bar with the address `localhost:7001/login/oauth/authorize?client_id=6e3a84154e73d1fb156a&response_type=code&redirect_uri=http://localhost:9195/http/hi&scope=read&state=app-test`. Below the URL bar is the Casdoor logo and the text "Continue with:". A button labeled "admin (Admin)" is shown, which has a small profile picture icon and the text "admin (Admin)". Below this, there is a section for "Or sign in with another account:" containing two input fields: one for "username, Email or phone" and one for "Password". There are also links for "Auto sign in" (with a checked checkbox) and "Forgot password?". At the bottom of the form is a large blue "Sign In" button. Below the "Sign In" button, there is a link "No account? sign up now".

localhost:9195/http/hi?code=822607b015cca2515b2b&state=app-test

hi! null! I'm Shenyu-Gateway System. Welcome!

### 3.3 Headers中的Carry token, 你也可以访问

The screenshot shows the Postman application interface. At the top, it displays a GET request to `http://localhost:9195/http/hi`. Below the URL, there are tabs for Params, Authorization, Headers (8), Body, Pre-request Script, Tests, and Settings. The Headers tab is currently selected, showing the following configuration:

Key	Description
Postman-Token	<calculated when request is sent>
Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.29.2
Accept	*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Authorization	eyJhbGciOiJSUzI1NjIiLmtpZCI6ImNlcnQtYn... <span style="color:red">Your token</span>

Below the headers, there are tabs for Body, Cookies (1), Headers (9), and Test Results. The Body tab is selected, showing the response body:

```
1 hi! null! I'm Shenyu-Gateway System. Welcome!
```

At the bottom right, there are buttons for Save Response, a copy icon, and a search icon.

### 3.4 它也可以保存 Headers 的用户名、id 和组织，以便下次使用。

# JavaScript

## 微信小程序

在 微信小程序中使用 Casdoor

# 微信小程序

## ① 信息

Casdoor支持1.41.0版后的微信小程序

## 介绍

由于微信小程序不支持标准的 OAuth，所以它不能跳转到自主机的 Casdoor 网页进行登录。因此，将 Casdoor 用于微信小程序的过程与普通方案的过程有所不同。

这份文件将讨论如何通过 Casdoor 访问微信小程序。您可以在 GitHub 上找到示例：[casdoor-wechat-miniprogram-example](#)。详细信息可参见微信小程序 [登录文档](#)。

以下是配置中的一些专有名词：

`CASDOOR_HOSTNAME`：私有部署的 Casdoor 域名或 IP。例如：

`https://door.casbin.com`。

## 第一步：部署 Casdoor

首先，您应先部署 [Casdoor](#)。

成功部署后，您需要确认：

1. Casdoor 可以正常登录使用。
2. 将 Casdoor 的 `origin` 值 (`conf/app.conf`) 设置为 `CASDOOR_HOSTNAME`。

```
conf > ⚙ app.conf
 8  dbName = casdoor
 9  redisEndpoint =
10 defaultStorageProvider =
11 isCloudIntranet = false
12 authState = "casdoor"
13 httpProxy = "127.0.0.1:10808"
14 verificationCodeTimeout = 10
15 initScore = 2000
16 logPostOnly = true
17 origin = "http://10.144.1.2:8000"| CASDOOR_HOSTNAME
```

## 第二步： 配置Casdoor应用程序

1. 在 casdoor 创建一个wechat idp, 并填写微信小程序开发平台给您的 APPID 和 APPSECRET :

New Provider [Save](#) [Save & Exit](#) [Cancel](#)

Name [?](#) : provider\_Mini Program

Display name [?](#) : Mini Program

Category [?](#) : OAuth

Type [?](#) : WeChat Mini Program

Client ID [?](#) : \*\*\*

Client secret [?](#) : \*\*\*

Provider URL [?](#) : <https://github.com/organizations/xxx/settings/applications/1234567>

[Save](#) [Save & Exit](#) [Cancel](#)

2. 创建或使用现有的 Casdoor 应用程序。
3. 将上面添加的 idp 添加到您想要使用的应用程序中。

#### ❗ TIPS

为方便起见，casdoor 将在应用程序中读取第一个WeChat类型 idp 默认是微信小程序 idp 。

因此，如果你想要在这个应用中使用微信小程序，请不要在一个应用中添加 WeChat 类型 idp 。

# 第三步： 编写微信小程序

WeChat Mini 方案提供一个内部登录的API并获取代码。 您需要做的只是将此代码发送到Casdoor。 Casdoor 将使用此代码从 WeChat 服务器获取一些信息(例如OpenID、SessionKey 等)。

下面代码展示如何完成上述过程：

```
// login in mini program
wx.login({
  success: res => {
    // this is your login code you need to send to casdoor
    console.log(res.code)

    wx.request({
      url: `${CASDOOR_HOSTNAME}/api/login/oauth/access_token`,
      method: "POST",
      data: {
        "tag": "wechat_miniprogram", // required
        "client_id": "6825f4f0af45554c8952",
        "code": res.code,
        "username": this.data.userInfo.nickName, // update user
        profile, when you login.
        "avatar": this.data.userInfo.avatarUrl,
      },
      header: {
        "content-type": "application/x-www-form-urlencoded",
      },
      success: res => {
        console.log(res)
        this.globalData.accessToken = res.data.access_token // get
        casdoor's accessToken
      }
    })
  }
})
```

值得一提的是，`tag` 参数是强制性的，您需要让用户了解这是来自微信小程序的请求。

上述代码在用户登录时传入微信小程序用户名和头像uri。您也可以通过这两个参数而不首先传递它们。然后在登录成功并获得访问令牌后将它们传递到 casdoor：

```
WX.getUserProfile({
  desc: 'share your info to casdoor',
  success: (res) => {
    this.setData({
      userInfo: res.userInfo,
      hasUserInfo: true
    })
    console.log(app.globalData.accessToken)
    wx.request({
      url: `${CASDOOR_HOSTNAME}/api/update-user`, // casdoor uri
      method: "POST",
      data: {
        "owner": "test",
        "name": "wechat-oGk3T5tIiMFo3SazC075f0HEiE7Q",
        "displayName": this.data.userInfo.nickName,
        "avatar": this.data.userInfo.avatarUrl
      },
      header: {
        "Authorization": "Bearer " + app.globalData.accessToken,
        // Bearer token
        "content-type": "application/json"
      },
      success: (res) => {
        console.log(res)
      }
    })
  }
})
```

此外，您可以使用 `accessToken` 作为任何Casdoor操作的访问令牌

 TIPS

目前 Casdoor 无法将现有账户绑定到微信小程序用户。在 Catdoor 从WeChat 获取 openID , 如果此 id 不存在。将创建一个新用户, 如果它存在, 将使用旧用户。



> 集成 >

Lua

# Lua



APISIX

在 APISIX 中使用 Casdoor

# APISIX

目前有两种方法可以使用 Casdoor 通过 APISIX 插件连接到 APISIX，并保护 APISIX 背后的 API：使用 APISIX 的 Casdoor 插件或使用 APISIX 的 OIDC 插件。

## 通过 APISIX 的 Casdoor 插件连接 Casdoor

这个名为 authz-casdoor 的插件可以保护 APISIX 背后的 api，强制每个请求都经过身份验证，而无需修改 api 代码。

### 如何启用它？

您需要在创建路由时指定此插件，并给出所有必需的字段。参见下面的示例。

```
curl "http://127.0.0.1:9080/apisix/admin/routes/1" -H "X-API-KEY: edd1c9f034335f136f87ad84b625c8f1" -X PUT -d '  
{}  
  "methods": ["GET"],  
  "uri": "/anything/*",  
  "plugins": {  
    "authz-casdoor": {  
      "endpoint_addr": "http://localhost:8000",  
      "callback_url": "http://localhost:9080/anything/callback",  
      "client_id": "7ceb9b7fda4a9061ec1c",  
      "client_secret": "3416238e1edf915eac08b8fe345b2b95cdba7e04"  
    }  
  },  
  "upstream": {
```

在本例中，我们使用apisix的管理API创建了指向“`httpbin.org:80`”的路由“`/anything/*`”，并启用了“`authz casdoor`”。此路由现在受到 Casdoor 的身份验证保护。

## 属性

名称	类型	申请标准	默认	有效期	描述
<code>endpoint_addr</code>	字符串	必填			Casdoor的url。
<code>client_id</code>	字符串	必填			Casdoor 中的 client id。
<code>client_secret</code>	字符串	必填			Casdoor 中的 client secret。
<code>callback_url</code>	字符串	必填			用于接收状态和代码的回调 url。

`endpoint_addr` 和 `callback_url` 不应以“`/`”结尾

在“`authz casdoor`”插件的配置中，我们可以看到四个参数。

第一个是“`callback_url`”。这是OAuth2中的回调url。应该强调的是，此回调url必须属于您为路由指定的“uri”，例如，在本例中，`http://localhost:9080/anything/callback`显然属于“`/anything/*`”。只有通过这种方式，插件才能拦截并利用对 `callback_url` 的访问（这样插件才能在 OAuth2 中获得代码和状态）。`Callback_url` 的逻辑完全由插件实现，因此无需修改服务器来实现此回调。

第二个参数“`endpoint_addr`”显然是Casdoor的url。第三个和第四个参数是“`client_id`”和

“client\_secret”，您可以在注册应用程序时从Casdoor获取这些参数。

## 它是如何工作的？

假设一个以前从未访问过此路由的新用户将访问它(`http://localhost:9080/anything/d?param1=foo¶m2=bar`)，考虑到已启用“authz casdoor”，此访问将首先由“authz casdoor”插件处理。然后检查会话，若此用户尚未通过身份验证后，将拦截访问。用户想要继续访问原始url后，他将被重定向到Casdoor的登录页面。

成功登录用户名和密码(或他使用的任何方法)，Casdoor会将此用户重定向到“callback\_url”，带有GET参数“code”和“state”。因为插件知道“callback\_url”，当这次拦截访问“callback\_url”时，Oauth2中的“授权代码流”逻辑将被触发，这意味着此插件将请求访问令牌以确认此用户是否真的登录。确认后，此插件将会将此用户重定向到原始用户想访问的URL，这是我们先前保存的。登录状态也将保持在会话。

下次此用户想访问此路由后面的网址(例如，`http://localhost:9080/anything/d`)，发现此用户以前已被身份验证。此插件将不再重定向此用户，以使此用户能够不受干扰地访问他在路由下想要的任何路径。

## 通过 APISIX 的 OIDC 插件连接 Casdoor

Casdoor 可以使用 OIDC 协议链接到 APISIX，这份文档将向您展示如何处理相关问题。

以下是配置中的一些专有名词：

`CASDOOR_HOSTNAME`：私有部署的Casdoor域名或IP。

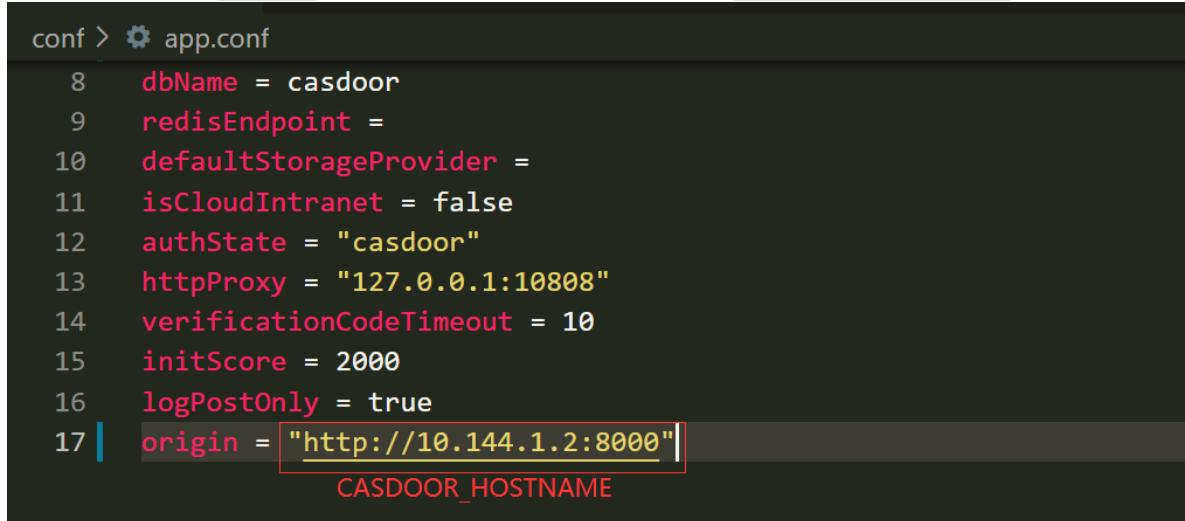
`APISIX_HOSTNAME`：部署 APISIX 的域名或 IP。

### 步骤1. 部署Casdoor和APISIX

首先，应该部署 Casdoor and APISIX。

在成功部署后，您需要确保：

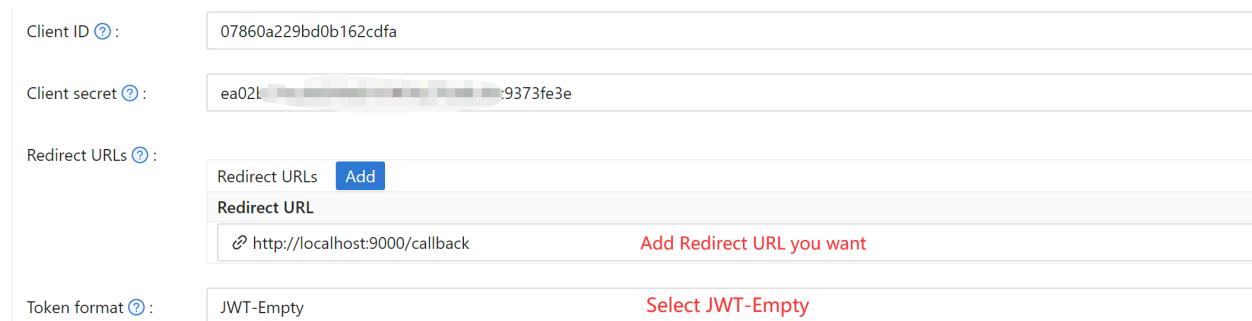
1. 可以登录并正常使用Casdoor。
2. 将Casdoor的 `origin` 值 (conf/app.conf) 设置为 `CASDOOR_HOSTNAME`。



```
conf > ⚙ app.conf
 8  dbName = casdoor
 9  redisEndpoint =
10 defaultStorageProvider =
11 isCloudIntranet = false
12 authState = "casdoor"
13 httpProxy = "127.0.0.1:10808"
14 verificationCodeTimeout = 10
15 initScore = 2000
16 logPostOnly = true
17 | origin = "http://10.144.1.2:8000"
      CASDOOR_HOSTNAME
```

## 步骤2. 配置Casdoor应用程序

1. 创建或使用现有的 Casdoor 应用程序。
2. 添加一个重定向网址： `http://APISIX_HOSTNAME/REDIRECTWHATYOUWANT` 并更改 `REDIRECTWHATYOUWANT` 到你需要的重定向网址上。
3. 选择 "JWT-Empty" 作为令牌格式选项
4. 添加您想要的提供商并补充其他设置。



Client ID ② :	07860a229bd0b162cdfa
Client secret ② :	ea021...9373fe3e
Redirect URLs ② :	<a href="#">Add</a> Redirect URL <input type="text" value="http://localhost:9000/callback"/> <a href="#">Add Redirect URL you want</a>
Token format ② :	<input checked="" type="radio" value="JWT-Empty"/> <a href="#">Select JWT-Empty</a>

不出意外的话，您会在应用程序设置页面看到：`Client ID` 和 `Client secret` 就像上面的图片一样。 我们将在下一步中使用它们。

打开你喜欢的浏览器，访问：[http://CASDOOR\\_HOSTNAME/.well-known/openid-configuration](http://CASDOOR_HOSTNAME/.well-known/openid-configuration)，你会看到 Casdoor 的 OIDC 配置。

## 步骤3. 配置 APISIX

APISIX 拥有官方 OIDC 支持，由 [lua-resetyl-openidc](#) 实现。

您可以根据 APISIX OIDC 文档定制设置，使用以下路由设置：

```
#Use your own X-Api-Key
$ curl -XPOST APISIX_HOSTNAME/apisix/admin/routes -H "X-Api-Key: edd1c9f034335f136f87ad84b625c8f1" -d '{
  "uri": "/get",
  "name": "apisix_casdoor_test",
  "plugins": {
    "openid-connect": {
      "client_id": "Client ID",
      "client_secret": "Client secret",
      "discovery": "http://CASDOOR_HOSTNAME/.well-known/openid-configuration",
      "introspection_endpoint_auth_method": "client_secret_basic",
      "logout_path": "/logout",
      "realm": "master",
      "redirect_uri": "http://APISIX_HOSTNAME/REDIRECTWHATYOUWANT",
      "bearer_only": false,
      "set_id_token_header": false,
      "access_token_in_authorization_header": true,
      "set_access_token_header": true,
      "set_userinfo_header": false,
      "realm": "master"
    }
  },
  "upstream": {
    "type": "roundrobin",
    "nodes": {
      "httpbin.org:80": 1
    }
  }
}'
```

现在, 请访问 `http://APISIX_HOSTNAME/get`, 浏览器会将您重定向到Casdoor登录页面, 并且在登录成功后, 您会看到我们已经向 `httpbin.org` 发出了请求。

# PHP



## 禅道

在禅道中使用 Casdoor 进行身份验证



## ShowDoc

在 ShowDoc 中使用 Casdoor 为 oAuth2 服务器

# 禅道

禅道 是一个 敏捷(scrum) 项目管理系统/工具，但它不支持 OIDC 本身。为了整合禅道和 Casdoor SSO，我们应该通过第三方OIDC 模块 [zentao-oidc](#)，和这个文档将显示你如何去做。

## 步骤1. 部署Casdoor和禅道

首先，应该部署 [Casdoor](#) 和 [禅道](#)。在成功部署后，您需要确保：

1. Casdoor 可以正常登录使用。
2. 您可以成功登录并使用禅道

## 步骤2. 集成禅道 OIDC 第三方模块

安装 [zentao-oidc](#)

```
git clone https://github.com/casdoor/zentao-oidc.git
```

或者您可以下载 ZIP 并解压缩它。

此模块用于在 OpenID 与 SSO 集成时使用 禅道 用法如下：

1. 将整个oidc目录复制到禅道模块中，并使用它作为禅道模块。重命名下载的软件包为“oidc”
2. 配置过滤器

因为禅道框架过滤了URL中的参数，不允许空格。所以你需要在 `/config/my.php` 末尾放置以下代码。

```
$filter->oidc=new stdclass();
$filter->oidc->index=new stdclass();
$filter->oidc->index->paramValue['scope']='reg::any';
```

### 3. 修改 `/module/common/model.php`

将“oidc”放在匿名访问列表中，并添加一行到 `isOpenMethod()` 方法 `model.php` 中。

```
public function isOpenMethod($module, $method){
    if($module == 'oidc' and $method == 'index') return true;
}
```

### 4. 如果您不想显示禅道登录屏幕，直接进入Casto登录屏幕。修改最后一行代码在 `public function checkPriv()` 在 `/module/common/model.php` 中。

```
//return print(js::locate(helper::createLink('user', 'login',
"referer=$referer")));
return print(js::locate(helper::createLink('oidc', 'index',
"referer=$referer")));
```

### 5. 修改 `setSuperVars()` 方法在 `framework/base/router.class.php`, 注释掉以下代码

```
public function setSuperVars()
// unset($_REQUEST);
```

## 步骤3. 配置Casdoor应用程序

1. 创建或使用现有的 Casdoor 应用程序。
2. 添加您的重定向url

The screenshot shows the Casdoor configuration interface. It includes fields for Client ID (d8d7715e24f077066a20), Client secret (redacted), Cert (cert-built-in), and Redirect URLs (http://127.0.0.1/zentao/oidc-index.html). The Redirect URLs section has an 'Add' button.

Client ID	d8d7715e24f077066a20				
Client secret	[REDACTED]				
Cert	cert-built-in				
Redirect URLs	<table border="1"><tr><td>Redirect URLs</td><td>Add</td></tr><tr><td>Redirect URL</td><td>http://127.0.0.1/zentao/oidc-index.html</td></tr></table>	Redirect URLs	Add	Redirect URL	http://127.0.0.1/zentao/oidc-index.html
Redirect URLs	Add				
Redirect URL	http://127.0.0.1/zentao/oidc-index.html				

3. 添加您想要的提供商并补充其他设置。

## 步骤4. 配置 Jenkins

在 oidc 中配置 config.php

```
$config->oidc->clientId=<Your ClientId>;  
$config->oidc->clientSecret=<Your ClientSecret>;  
$config->oidc->issuer="http://localhost:8000";
```

在 module/oidc 公共函数索引() 中设置你的重定向Url

```
$oidc->setRedirectURL($path."/zentao/oidc-index.html");
```

① 备注

这里的URL是指调用 'index' 方法在 'oidc' 模块中。您还需要设置一个变量分隔符。哪个框架默认为破折号： - 请参阅禅道的官方框架以了解详情。 "[禅道框架](#)"

# ShowDoc

## 在 ShowDoc 中使用 Casdoor 进行身份验证

ShowDoc 是一个 IT 团队在线API文档、技术文档工具。 Showdoc 可以轻松使用 Markdown 语法来写美丽的 API 文档、数据字典文档、技术文档、在线Excel文档等等。

Showdoc 支持第三方身份验证，包括Oauth。 以下是操作教程。

### 第1步： 创建一个Casdoor应用程序

转到您的 Casdoor 并添加您的新应用程序 ShowDoc。 下面是在Casdoor中创建 ShowDoc应用程序的示例。

[Edit Application](#)[Save](#)[Save & Exit](#)Name [?](#) :

myApplication

Display name [?](#) :

myApplication

Logo [?](#) :URL [?](#) :[https://cdn.casdoor.com/logo/casdoor-logo\\_1185x256.png](https://cdn.casdoor.com/logo/casdoor-logo_1185x256.png)

Preview:

Home [?](#) :[?](#)Description [?](#) :Organization [?](#) :

built-in

Client ID [?](#) :

208d745196c23df9fd5b

Client secret [?](#) :

4c89f447af77bc276431ab885463ebcb8d6efc3c

Cert [?](#) :

cert-built-in

请记住下一步的 `client ID` 和 `client Secret`。

...信息

请在此步骤中不要填写 `callback url`。Url取决于下一步 `showdoc` 的配置。稍后我们将返回来设置一个正确的回调URL。

...

## 第2步：配置ShowDoc

首先，启动 OAuth2 登录按钮。然后按照示例填写 `callback url` 填写上一步中记住的 `client ID` 和 `client secret`。

The screenshot shows the ShowDoc configuration interface. On the left, there is a sidebar with the following menu items:

- 用户管理
- 项目管理
- 附件管理
- 集成登录** (highlighted with a red box)
- 站点设置
- 关于本站

The main configuration area has the following tabs at the top: `LDAP`, `OAuth2` (highlighted with a red box), and `通用接入`. Below the tabs, there is a toggle switch labeled `启动OAuth2登录` (highlighted with a red box). The configuration fields are as follows:

- `callback url`: `http://127.0.0.1/server/?s=/api/extLogin/oauth2`
- `入口文字提示`: `casdoor sso`
- `Client id`: `208d745196c23df9fd5b`
- `Client secret`: `4c89f447af77bc276431ab885463ebcb8d6efc3c`
- `Oauth host`: `http://` dropdown set to `127.0.0.1:8000`
- `Authorize path`: `/login/oauth/authorize`
- `AccessToken path`: `/api/login/oauth/access_token`

需要 `Authorize path`、`AccessToken path`、`User info path`。您可以填写如下所示。

```
Authorize path: /login/oauth/authorize
AccessToken path: /api/login/oauth/access_token
User info path: /api/get-account
```

## 第3步：在casdoor中配置回调url

返回步骤1中的应用程序编辑页面并添加您填写在ShowDoc中的 `callback url`。

Redirect URLs ② :

Redirect URLs	Add
Redirect URL	<input type="text" value="http://127.0.0.1/server/?s=/api/extLogin/oauth2"/>

## 第4步：在ShowDoc上试试

您应该在登录页面中看到这一点：

# 登录

 用户名/邮箱 密码 验证码[注册新账号](#)[casdoor sso](#)

恭喜您！ 您已完成所有步骤 按 "castor sso" 按钮，您将被重定向到casdoor登录页面。



&gt; 集成

&gt; Ruby

# Ruby

## GitLab

在 GitLab 服务器中使用 Casdoor 进行身份验证

# GitLab

Casdoor 可以使用 OIDC 协议链接到私有部署的 GitLab，该文档将向您展示如何处理相关问题。

以下是配置中的一些专有名词：

`CASDOOR_HOSTNAME`: 私有部署的Casdoor域名或IP。例如:

`https://door.casbin.com`.

`GITLAB_HOSTNAME`: 部署 GitLab 的域名或 IP。例如: `https://gitlab.com`.

## 第 1 步：部署 Casdoor 和 GitLab

首先，[Casdoor](#) 和 [GitLab](#) 应该部署完毕。

成功部署后，您需要确保：

1. Casdoor 可以正常登录使用。
2. 将 Casdoor 的 `origin` 值 (`conf/app.conf`) 设置为 `CASDOOR_HOSTNAME`。

```
conf > ⚙ app.conf
 8  dbName = casdoor
 9  redisEndpoint =
10 defaultStorageProvider =
11 isCloudIntranet = false
12 authState = "casdoor"
13 httpProxy = "127.0.0.1:10808"
14 verificationCodeTimeout = 10
15 initScore = 2000
16 logPostOnly = true
17 origin = "http://10.144.1.2:8000"|
          CASDOOR_HOSTNAME
```

## 第 2 步： 配置 Casdoor 应用程序

1. 创建或使用现有的 Casdoor 应用程序。
2. 添加重定向网址: `http://GITLAB_HOSTNAME/users/auth/openid_connect/callback`。
3. 添加您想要的提供商并补充其他设置。

The screenshot shows the Casdoor application configuration page with the following fields filled in:

- Description: GitLab
- Organization: built-in
- Client ID: eab9...35b6
- Client secret: 95e7...b3a0188a5
- Redirect URLs:
  - Add
  - Redirect URL: `http://GITLAB_HOSTNAME/users/auth/openid_connect/callback` (highlighted with a red box)

不出意外的话，您会在应用程序设置页面看到: `client ID` 和 `client secret` 就像上面的图片一样。 我们将在下一步中使用它们。

打开你喜欢的浏览器，访问：[http://\*\*CASDOOR\\_HOSTNAME\*\*/well-known/openid-configuration](http://<CASDOOR_HOSTNAME>/.well-known/openid-configuration)，你会看到 Casdoor 的 OIDC 配置。

## 第 3 步：配置 GitLab

您可以按照以下步骤进行设置，或根据 [这个文档](#) 进行自定义设置(例如，您正在使用源代码而不是 Omnibus 安装 GitLab)。

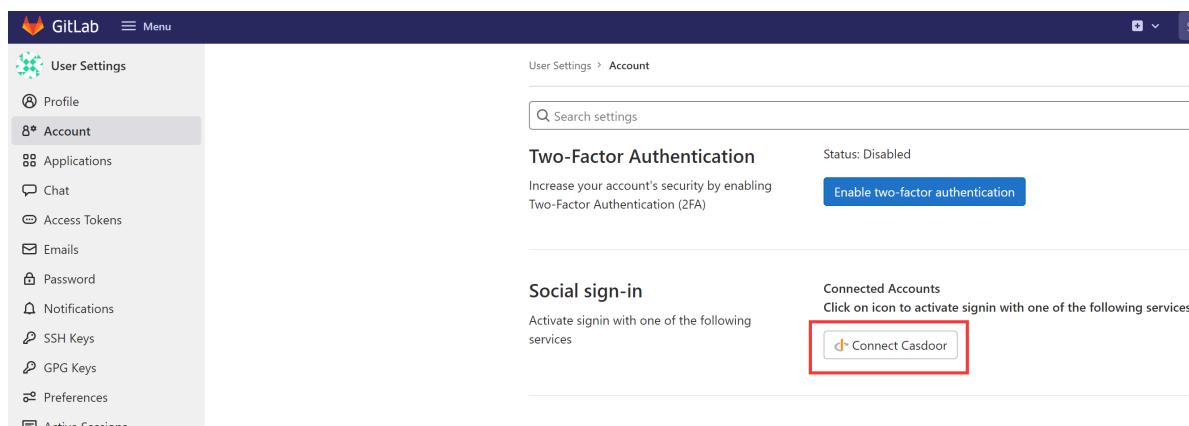
1. 在 GitLab 服务器上，打开配置文件。

```
sudo editor /etc/gitlab/gitlab.rb
```

2. 添加提供商配置。 (HOSTNAME 链接应包括 http 或 https)

```
gitlab_rails['omniauth_providers'] = [
  {
    name: "openid_connect",
    label: "Casdoor", # 登录按钮的可选标签，默认为"Openid Connect"
    args: {
      name: "openid_connect",
      scope: ["openid", "profile", "email"],
      response_type: "code",
      issuer: "<CASDOOR_HOSTNAME>",
      client_auth_method: "query",
      discovery: true,
      uid_field: "preferred_username",
      client_options: {
        identifier: "<你的 CLIENT ID>",
        secret: "<你的 CLIENT SECRET>",
        redirect_uri: "<GITLAB_HOSTNAME>/users/auth/openid_connect/callback"
      }
    }
]
```

3. 重新启动 GitLab 服务器。
4. 每个注册用户都可以打开 `GITLAB_HOSTNAME/-/profile/account`, 连接 Casdoor 账号。



The screenshot shows the 'User Settings > Account' page in GitLab. On the left, there's a sidebar with options like Profile, Account (which is selected), Applications, Chat, Access Tokens, Emails, Password, Notifications, SSH Keys, GPG Keys, and Preferences. The main content area has a 'Two-Factor Authentication' section with a status of 'Disabled' and a 'Enable two-factor authentication' button. Below it is a 'Social sign-in' section with a sub-section for 'Connected Accounts'. A red box highlights the 'Connect Casdoor' button next to the Casdoor icon.

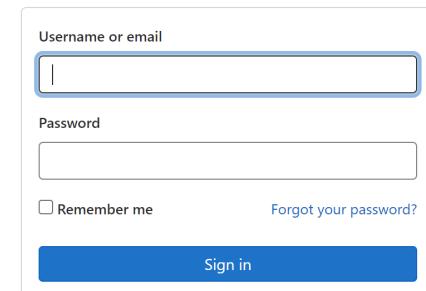
5. 完成！现在，您可以通过 casdoor 登录您自己的 GitLab。

GitLab

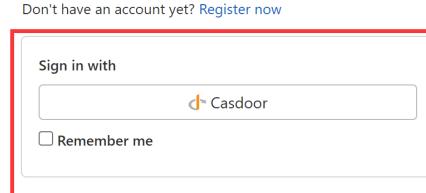
### A complete DevOps platform

GitLab is a single application for the entire software development lifecycle. From project planning and source code management to CI/CD, monitoring, and security.

This is a self-managed instance of GitLab.



The screenshot shows the GitLab login page. It features fields for 'Username or email' and 'Password', and checkboxes for 'Remember me' and 'Forgot your password?'. Below these is a large blue 'Sign in' button. At the bottom, there's a link 'Don't have an account yet? Register now'. A red box highlights the 'Sign in with' section, which includes a 'Casdoor' button and a 'Remember me' checkbox.



The screenshot shows the 'Sign in with' section of the GitLab login page. It includes a 'Casdoor' button and a 'Remember me' checkbox. A red box highlights this entire section.



&gt;

WebAuthn

# WebAuthn

## 概述

在 Casdoor 中使用 webauthn

# 概述

## 概述

我们很高兴地通知您，Casdoor现在支持通过Webauthn登录，这意味着：您也许能够用您的生物识别来登录，例如指纹或人脸识别，甚至是通过U磁盘登录，但前提是您的设备支持这些很酷的授权方法和WebAuthn。

## 什么是 WebAuthn？

Web身份验证API(也称为WebAuthn)是由W3C和FIDO编写的规范，谷歌、Mozilla、微软、Yubico等也参与其中。API允许服务器使用公用钥匙加密而不是密码来注册和认证用户。它允许服务器与强大的身份验证器集成，现已编入设备，例如Windows Hello或Apple's Touch ID。

简而言之，Webauthn要求用户生成公钥——私钥对，并将公钥移交给网站。当用户想登录到网站时，Web生成随机的数字，请用户用私钥加密，并将结果发送回来。收到结果后，网站将尝试使用公钥解密，并且如果解密后的数字与之前生成的随机数字相同。该用户将被视为合法用户，允许登录。我们调用Webauthn credential的公用钥匙与必要的信息(例如用户名或滥用信息的用户授权者)这正是网站存储的内容。

公钥-私钥配对完全是独特独特的三个信息：(用户名、用户授权和网站URL)。这意味着，如果(用户名、用户授权者和网站的URL)是相同的，那么密钥对应该是相同的，反之亦然。

关于WebAuthn技术的更多详细信息，您可以访问<https://webauthn.guide/>。

# 如何在Casdoor中使用 Webauthn ?

在登录页面中，您必须已经看到使用 WebAuthn 登录的选择。但考虑到您尚未获得 Webauthn凭据(也就是webauthn密码，这样说您可能更好地理解) 在这个教程中，我们将向您展示如何创建和管理凭据，如何再创作和管理以及如何使用凭据登录。

## 第 0 步: 修改配置并打开webauthn身份验证

您可以在conf/app.conf 中查看

```
origin = "http://localhost:8000"
```

请确保此配置是您网站的 URL。

## 只有https 支持webauthn，除非您正在使用本地主机

然后作为管理员登录，然后转到您的应用程序的编辑页面。 打开开关"启用WebAuthn signin"。 默认情况下，此功能未启用。

## 第 1 步：转到“我的帐户”页面

第 1 步：转到账户页面。 在这个页面，您可以看到"添加 WebAuthn Credential" 按钮和一个显示您以前注册过的所有Webauthn凭据的列表。

The screenshot shows the Casdoor application configuration interface. At the top, there's a field labeled "Signup application" with a placeholder "(empty)". Below it, a section for "3rd-party logins" shows a GitHub icon with the text "GitHub: (empty)" and a blue "Link" button. Under "WebAuthn credentials", there's a table with one row containing a "Delete" button. The "Roles" and "Permissions" sections are present but empty. At the bottom, there are "Save" and "Save & Exit" buttons.

按下按钮，然后按照您设备的指示注册新凭据进入casdoor。

您可以通过列表中的“删除”按钮删除任何凭据。

## 第 2 步：通过webauthn登录

在这个步骤开始之前，请确保您已经登出了casdoor。

转到页面登录，选择webauthn登录方式，输入您的用户名并按登录按钮，然后按照您设备的说明操作。

(例如，如果您使用指纹和窗口Hello，您就会看到类似的东西)



Windows Security

Making sure it's you

Please sign in as admin to .casdoor.com.

This request comes from Chrome, published by Google LLC.

Auto sign in      [Forgot password?](#)

[Sign in with WebAuthn](#)

[I forgot my PIN](#)

[Cancel](#)



然后您将看到您已经登录。



&gt;

国际化

# 国际化

Casdoor 支持多种语言，将翻译部署到 [Crowdin](#)，我们支持中文、法文、德文、俄文、日文和朝鲜文。

Cassdoor 使用官方的 Crowdin cli 同步来自 Crowdin 的翻译。如果您想要添加更多语言支持，请在 [我们的社区](#)中提出，如果您想帮助我们加快翻译工作，请帮助我们翻译 [Crowdin](#)。



&gt;

贡献者指南

# 为 Casdoor 做贡献

欢迎使用 Casdoor！此文档是介绍如何为 Casdoor 做出贡献的指南

如果您发现有错误或缺失之处，欢迎留下意见或建议

## 开始参与

为 Casdoor 作出贡献的方式多种多样，可以从以下几方面入手：

使用Casdoor并报告问题！ 使用 Casdoor 时，报告问题以促进Casdoor的开发，不管有任何缺陷或 提议。 在 GitHub 上创建issue前，最好先在 [Gitter](#), [Casbin 论坛](#) or QQ 群： [645200447](#) 上讨论。

### ① 信息

创建issue时，请使用英文详细描述您的问题。

帮助编写文档！ 从文档开始贡献是一个很好的选择，可以开始您的贡献。

帮助解决issue！ 我们准备了一个表格，其中包含适合初学者的简单任务，挑战程度不同带有不同标签的标签，请查看此处的表格[Casdoor Easy Tasks](#)。

## 贡献

现在，如果您已经准备好创建 PR，在这里是贡献者的工作流：

1. Fork到您自己的仓库

2. 克隆您的fork到本地仓库
3. 创建一个新分支并在其上工作
4. 保持您的分支同步
5. 提交您的更改(请确保您的提交信息简洁)
6. 将你的提交推送到你的fork仓库中
7. 创建从您的分支到我们的**master**分支的合并请求。

## 合并请求

### 开始之前

Casdoor使用GitHub 作为其开发平台。 因此， 合并请求是贡献的主要来源。

在您打开拉取请求之前， 您需要知道一些基本准则：

- 当你第一次拉取请求时， 你需要签名 CLA。
- 解释您为什么要发送此 PR 以及此 PR 将会在仓库中做些什么。
- **只允许一次提交。** 请确保每个合并请求只做一件事， 否则请分开提交。
- 如果有新添加的文件，请将新文件顶部的 Casdoor 许可证包括在内。

```
// Copyright 2022 The Casdoor Authors. All Rights Reserved.  
//  
// Licensed under the Apache License, Version 2.0 (the "License");  
// you may not use this file except in compliance with the License.
```

## Semantic PRs

您的合并请求应遵循常规承诺的样本。基本要求是有PR 标题或至少一个 提交消息。例如，三个常用的PR标题如下：

### ⚠ 注意事项

PR 标题必须是小写的。

1. fix: a commit of the type `fix` patches a bug in your codebase.

```
fix: prevent racing of requests
```

2. feat: a commit of the type `feat` introduces a new feature to the codebase.

```
feat: allow provided config object to extend other configs
```

3. docs: a commit of the type `docs` add or improve a document.

```
docs: correct spelling of CHANGELOG
```

欲了解更多详情，请参阅 [常规承诺](#)

## 将 PR 链接到问题 (如果存在)

您可以将拉取请求链接到议题，以显示修复正在进行中，并在拉取请求被合并时自动关闭该议题。

## 使用关键词将拉取请求链接到议题

您可以通过在拉取请求说明或提交消息中使用支持的关键词将拉取请求链接到议题。 拉取请求**必须**在默认分支上。

- close
- fix
- resolve

同一仓库中的问题，例如：

```
Fix: #902
```

欲了解更多详情，请参阅 [Link PR to issue](#)。

## 修改PRs

您的 PR 可能不可避免地需要修改。 当代码需要更改时，请 **重新使用同一PR**。 不要关闭 PR 并打开一个新的 PR

下面是一个示例。

- 修改本地代码。
- 修改此提交。

```
git commit --amend
```

- 推送代码到远程仓库

```
git push --force
```

然后，PR 已成功修改！您可以在casdoor库中检查它。

## 相关代码

一些原则：

可读性 - 重要代码应有充分的文件记录。代码风格应与现有的风格一致。

### 命名规范

例如, `signupUrl` 对于变量名字, `Signup URL` 对于前端UI显示

### 如何更新 i18n 数据？

请注意，我们使用 [Crowdin](#) 作为翻译平台和i18next作为翻译工具。当您在 `web/` 目录中使用i18next添加一些单词时，您可以运行 `i18n/generate_test.go` 自动生成 `web/src/locales/**/data.json`。

Run `i18n/generate_test.go`:

```
cd i18n && go test
```

默认情况下，所有语言都以英文填写。鼓励您在您的 PR 已被合并后用 [Crowdin](#)帮助翻译在 `web/src/locales/zh/data.json` 新添加的字符串

#### ⚠ 注意事项

如果您不熟悉其他语言，请不要翻译。保持文件原样。

# 许可证书

为Casdoor作出贡献，即代表您同意您的贡献将遵循 Apache 许可协议。