



概述

Casdoor is a UI-first [Identity Access Management \(IAM\)](#) / [Single-Sign-On \(SSO\)](#) platform with web UI supporting OAuth 2.0, OIDC, SAML, CAS, LDAP, SCIM, WebAuthn, TOTP, MFA, RADIUS, Google Workspace, Active Directory and Kerberos.

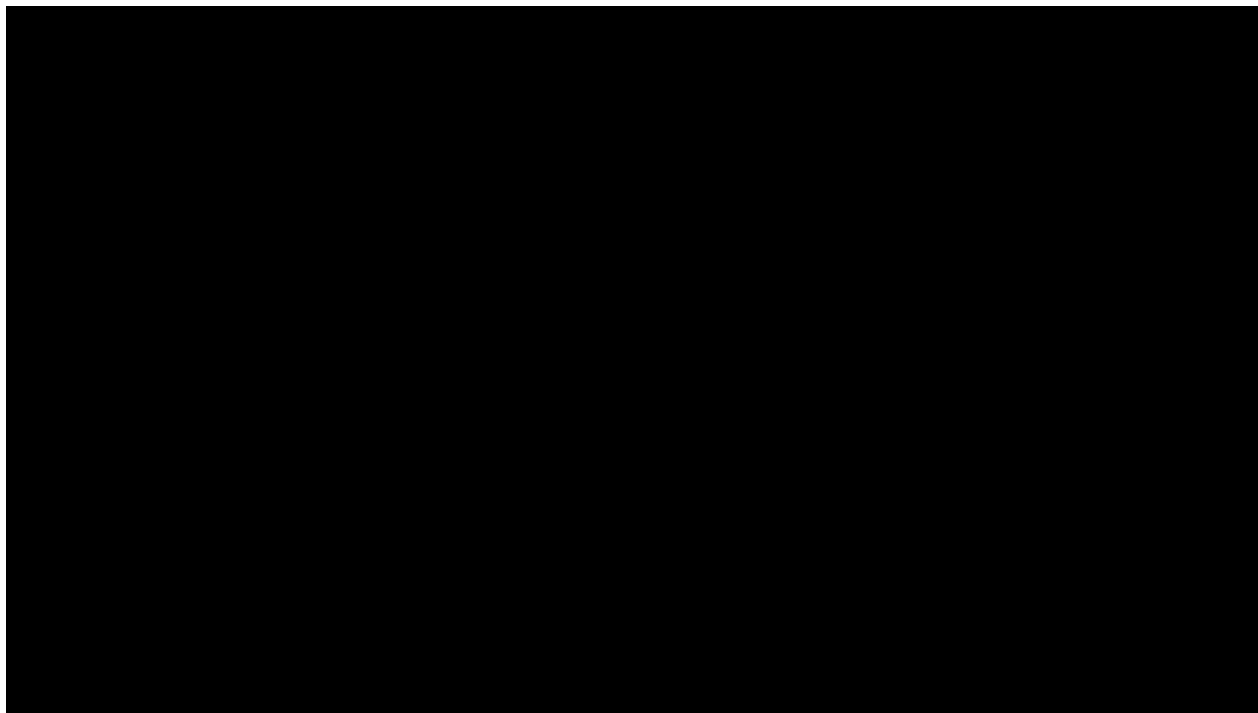
You need to enable JavaScript to run this app.

Casdoor 可为网页和应用程序用户的登录请求提供服务。

Casdoor 的特性:

1. Casdoor 遵循前后端分离架构, 采用 Golang 进行开发。它支持高同步, 提供基于网页的用户界面管理, 并支持10多种语言的本地化。
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. Casdoor 支持基于 [Cassbin](#) 的授权管理。它支持 ACL、RBAC、ABAC 和 RESTful鉴权管理模式。
4. Casdoor 提供了手机验证码、电子邮件验证码以及重置密码的功能。
5. Casdoor 支持日志的审计和记录。
6. Casdoor 可以使用阿里云、腾讯云、七牛云提供的图片CDN云存储功能。
7. Casdoor 允许自定义注册、登录以及找回密码页面。
8. 通过数据库同步支持与现有系统的集成, 从而能够顺利过渡到 Casdoor。
9. Casdoor 支持主流数据库: MySQL、PostgreSQL、SQL Server 等, 并支持扩展插件以支持新的数据库。

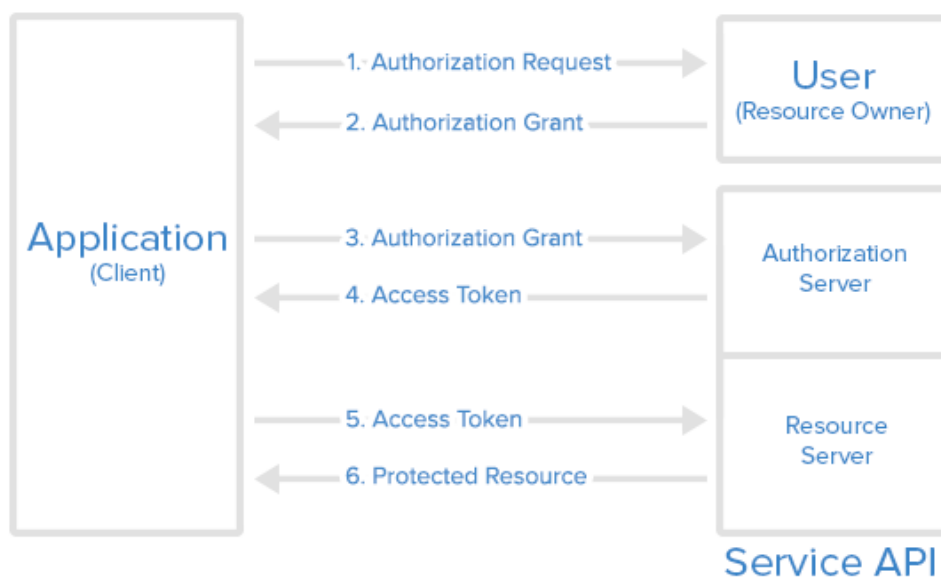
它如何工作？



步骤0（前置知识）

1. Casdoor的授权程序基于OAuth 2.0协议。强烈建议简要了解下 OAuth 2.0 的运行原理。你可以通过此处了解 [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` 这种格式来编写 URL。并使用你的 Casdoor 的网站 URL 替换 `endpoint`，以及使用您的信息替换 `xxx`。

i 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`: 您应该将此设置为您自己的应用程序回调 URL。Casdoor 将使用此信息在授权后发送回响应。
- 对于 `state`: 您应该在此处填写您的应用程序名称。

应用程序将会提示用户：“嘿，我需要一些资源，且该资源需要您的许可我才能访问。您是否可以转到这个网址并为我输入您的用户名和密码？”

使用正确组合的 URL，应用就会向此 URL 发起请求，完成 `授权请求`。

步骤 2（授权认证）

这个步骤非常直接：用户将被重定向到在步骤1中组成的URL，用户将看到来自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 `code` 和 `status` 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 will say: "Looking good ~ this is the user (who is authorizing the Application to get the `code` and `state`) I know in my database, and I will send the `code` and `state` back to the Application using the callback URL (`redirect_uri`)".

将这两部分信息发送回您的应用程序后，应用程序将获得授权，`Authorization Grant` 完成。

提示

Casdoor也提供第三方登录。在这种情况下，你将看到的不是凭证输入页面，而是第三方提供商的列表。您可以使用这些提供商登录您的应用，Casdoor充当中间层（中间件）。

步骤 3（授权认证）

在这一步中，您的应用程序已经拥有了步骤2的代码，并且它将对Casdoor说：“嘿，用户同意给我 `code`。你能验证这个 `code` 并给我 `access_token` 吗？”

步骤 4（访问令牌）

Casdoor对您的应用程序做出回应：“你知道吗，这段 `code` 看起来是合法的。”你必须是正确的应用程序。这是给你的 `access_token`。“有了这个 `code`，Casdoor确认它是一个被授权的应用程序（在步骤2中被正确的用户授权）试图获取 `access_token`（稍后将用于访问更多资源）。

步骤 5（访问令牌）

在这一步中，你的应用程序说：“很好！我刚刚得到了新鲜又美味的 `access_token`。现在我可以使用它从 `Resource Server` 获取更有价值的东西！”

您的应用程序然后转向 `Resource Server`，并说：“嘿，伙计，你能检查一下这个 `access_token`

吗？" 我收到了来自Casdoor的它。您是否想要验证这是否是您发给Casdoor的正确令牌？"

步骤 6 (受保护资源)

Resource Server 对您的应用程序做出响应："还不错。" 这看起来就像我发给Casdoor的那个，而Casdoor说，持有这个access_token的人可以访问这些Protected Resources。那就去吧，拿起来吧！"

这基本上就是Casdoor如何与您的应用程序一起工作的。

提示

Casdoor可以同时充当Authorization Server和Resource Server。换句话说，Casdoor 授权您的应用程序访问资源，通常是当前登录用户的信息，来自Casdoor的数据库。

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 进行身份验证。

- [Casbin-OA](#)
- 源码地址: <https://github.com/casbin/casbin-oa>

Casnode

Casnode 是由 Cassbin 社区开发的官方论坛。

它使用 Casdoor 作为认证平台并管理成员。

- [Casnode](#)
- 源码地址: <https://github.com/casbin/casnode>

结构

Casdoor 包括以下两部分:

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

核心概念

作为Casdoor的管理员，你至少应该熟悉四个核心概念：`Organization`，`User`，`Application`和`Provider`。

💡 提示

在接下来的部分，我们将使用演示站点 <https://door.casdoor.com> 作为示例。

组织

在Casdoor中，组织是用户和应用程序的容器。例如，公司的所有员工或商业的所有客户都可以被抽象为一个组织。以下显示了`Organization`类的定义：

```
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

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

- `built-in` 组织用户，例如 `built-in/admin`：在Casdoor平台上拥有完全管理权限的全局管理员。
- 其他组织的用户，例如 `my-company/alice`：可以注册、登录、登出、更改自己的个人资料等的普通用户。

在Casdoor API中，用户通常被标识为 `<organization_name>/<username>`。例如，Casdoor的默认管理员表示为 `built-in/admin`。此外，`User` 类定义包含一个 `id` 属性，这是一个像 `d835a48f-2e88-4c1f-b907-60ac6b6c1b40` 这样的UUID，可以被应用程序选择作为用户的ID。

💡 提示

对于只针对一个组织的应用程序，为了简化，可以使用 `<username>` 代替 `<organization_name>/<username>` 作为应用程序中的用户ID。

这是`User`类的定义：

```
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"`
}
```

应用程序

一个应用程序代表需要由Casdoor保护的网路服务，例如论坛网站，OA系统或CRM系统。

```
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"`
    RedirectUri    []string      `xorm:"varchar(1000)" json:"redirectUri"`
    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定制的注册页面、登录页面等。根登录页面 `/login`（例如，<https://door.casdoor.com/login>）是仅供Casdoor内置应用程序：`app-built-in`的登录页面。

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

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

登录 URL

通过Casdoor内置的应用程序登录Casdoor非常简单；只需访问Casdoor服务器主页（例如，<https://door.casdoor.com> 用于演示站点）它将自动将您重定向到 `/login`。但是你如何在前端和后端代码中获取其他应用程序的URLs呢？您可以手动连接字符串，或者调用Casdoor SDKs提供的一些实用函数来获取

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
 - 为指定的组织登录: `<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的前端JavaScript代码)

`getSignupUrl()` 和 `getSigninUrl()`: [casdoor-js-sdk](#)

3. 使用后端SDK (用于Go, Java等后端代码)

`GetSignupUrl()` 和 `GetSigninUrl()`: [casdoor-go-sdk](#)

提供商

Casdoor是一个联合的单点登录系统, 支持通过OIDC、OAuth和SAML的多个身份提供商。Casdoor也可以通过电子邮件或短信向用户发送验证码或其他通知。Casdoor 使用 `Provider` 的概念来管理所有这些第三方连接器。

可以在[provider/overview](#)找到Casdoor支持的所有提供商的列表。

```
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"`
  AppId     string `xorm:"varchar(100)" json:"appId"`

  Endpoint string `xorm:"varchar(1000)" json:"endpoint"`
  IntranetEndpoint string `xorm:"varchar(100)" json:"intranetEndpoint"`
  Domain    string `xorm:"varchar(100)" json:"domain"`
  Bucket    string `xorm:"varchar(100)" json:"bucket"`

  Metadata string `xorm:"mediumtext" json:"metadata"`
  IdP      string `xorm:"mediumtext" json:"idP"`
}
```


Casdoor是如何自我管理的?

首次运行Casdoor时，会创建一些内置对象以方便其管理：

- 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。
- 一个名为 `app-built-in` 的内置应用，由 `built-in` 组织管理，代表Casdoor本身。

`built-in` 组织下的所有用户，包括 `admin`，都将在Casdoor平台上拥有完全的管理员权限。因此，如果有多个管理员，建议在 `built-in` 组织下创建新账户。或者，应关闭 `app-built-in` 应用程序的注册通道，以防止不必要的访问。

注意事项

无法通过网页用户界面或RESTful API重命名或删除内置对象。Casdoor在许多地方硬编码了这些保留名称；尝试通过修改数据库来重命名或删除它们可能会导致整个系统崩溃。

服务器安装

安装要求

操作系统

所有主要的操作系统，包括Windows，Linux和macOS，都得到支持。

环境

- [Go 1.17+](#)
- [Node.js LTS \(18\)](#)
- [Yarn 1.x](#)

⚠️ 信息

我们强烈建议使用 [Yarn 1.x](#) 来运行和构建Casdoor前端。使用NPM可能会导致界面样式问题。欲了解更多详情，请见：[casdoor#294](#)。

⚠️ 注意事项

如果您的网络无法直接同步Go依赖包，您需要配置GOPROXY环境变量。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](#)，当前支持的数据库包括：

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

下载

Casdoor 的源代码托管在 GitHub: <https://github.com/casdoor/casdoor>。转到后端代码和 React 前端代码都包含在一个仓库中。

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

Casdoor支持 Go Modules。要下载代码，只需使用git克隆代码：

```
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和PostgreSQL。默认情况下，Casto使用MySQL。

MySQL

Casdoor将会把users, nodes和topics信息存储在一个名为casdoor的MySQL数据库中。如果数据库不存在，则需手动创建。可以在以下位置指定数据库连接字符串

<https://github.com/casdoor/blob/master/conf/app.conf>

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

PostgreSQL

在运行 Casdoor 之前，您需要手动准备PostgreSQL 数据库，因为Castor 需要在使用xorm 打开 Postgres 时指定一个数据库。

假设你已经准备好了一个名为casdoor的数据库，你应该像这样指定app.conf：

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

❗ 对于 PostgreSQL，请确保 `dataSourceName` 有一个非空的 `dbName`，并且也在上面的示例中为 `dbName` 字段复制数据库名称。 :::

CockroachDB

CockroachDB也可以与PostgreSQL驱动程序一起使用，并且配置与PostgreSQL相同。

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

SQLite3

要配置SQLite3，您应该像这样指定 `app.conf`：

```
driverName = sqlite
dataSourceName = "file:casdoor.db?cache=shared"
dbName = casdoor
```

通过 Ini 文件配置

Casdoor可以通过单个文件进行配置：[conf/app.conf](#)，该文件默认包含以下内容：

```
appname = casdoor
httpport = 8000
runmode = dev
SessionOn = true
copyrequestbody = true
driverName = mysql
```

- `appName` 是应用程序名称，目前没有实际用途。
- `httpPort` 是您的后端应用程序正在监听的端口。
- `runmode` 可以设置为 `dev` 或 `prod`。
- `SessionOn` 决定是否启用会话，默认情况下是启用的。
- `driverName`、`dataSourceName` 和 `dbName` 已在前面介绍过。请参阅[配置数据库](#)以获取详细信息。
- `verificationCodeTimeout` 设置了验证码的过期时间。过期后，用户需要再次获取。

作为初学者，你只需要修改两项：`driverName` 和 `dataSourceName`，根据你的数据库进行修改。这个数据库将被Casdoor用来存储所有数据，包括用户、组织和应用程序。

- `tableNamePrefix` 是使用适配器时表的前缀。
- `showSql` 决定了如果日志级别大于INFO，是否在日志器上显示SQL语句。
- `redisEndpoint` 后填写Redis地址，被Beego用于session存储 如果此参数为空，会话数据将作为文件在 `./tmp` 文件夹中本地存储。要使用Redis作为Beego会话存储，值应该是类似于：`redis.example.com:6379`。如果Redis是在本地部署的，你可以使用 `localhost:6379`。如果启用了Redis密码，请使用 `redis.example.com:6379,db,password`。在以下链接查看更多详情：<https://github.com/beego/beedoc/blob/master/en-US/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` 用于识别您的提供者端点是否为内网端点。
- `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Google相关的服务，或者使用 `Google`、`GitHub`、`Facebook`、`LinkedIn` 或 `Steam` 作为OAuth提供商，这在某些地区可能会受到网络限制。

- `initscore` 是每个用户的最初分数。每个用户都有一个得分属性。分数由 `Casnode` 使用，并且在 `Casdoor` 中不控制任何东西。
- `logPostOnly` 用于确定是否仅使用 `post` 方法来添加记录。
- `origin` 是 `origin` 形式的后端域名
- `staticBaseUrl` 是系统初始化数据库时使用的静态图像的地址。
- `enableGzip` 会在请求头包含 `Accept-Encoding=gzip` 时接受并以 `gzip` 编码进行响应。

通过环境变量配置

`Casdoor` 在上述 `ini` 文件中定义的所有配置项都可以通过环境变量进行配置，以及一些 `beego` 配置项（如 `httpport`，`appname`）。

例如，当你尝试启动 `Casdoor` 时，你可以使用类似这样的方式通过环境变量传递配置：

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

此外，`export` 衍生品也是一种可能的方法。环境变量的名称应与您想在 `ini` 文件中使用的名称完全匹配。

注意：环境变量中的配置可以覆盖 `ini` 文件中的配置。

运行

目前有两种启动方法，您可以根据自己的情况选择一种。

开发模式

后端

Casdoor的Go后端默认在8000端口运行。您可以使用以下命令启动Go后端：

```
go run main.go
```

服务器成功运行后，您可以开始进行前端部分。

前端

Casdoor的前端是一个非常经典的>Create-React-App (CRA)项目。它默认在端口7001上运行。使用以下命令来运行前端：

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

在您的浏览器中访问 <http://localhost:7001>。使用默认的全局管理员账户：`built-in/admin` 登录Casdoor控制面板。

```
admin  
123
```

生产模式

后端

将Casdoor Go后端代码编译成可执行文件并启动它。

对于Linux:

```
go build
./casdoor
```

适用于Windows:

```
go build
casdoor.exe
```

前端

将Casdoor前端代码构建成静态资源（.html, .js, .css文件）:

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

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

```
admin
123
```

提示

要使用另一个端口, 请编辑 `conf/app.conf` 并修改 `httpport`, 然后重新启动Go后端。

CASDOOR 端口详情

在dev环境中，前端由 `yarn run` 在7001端口运行，所以如果你想要进入Casdoor登录页面，你需要将Casdoor链接设置为 <http://localhost:7001>。

在prod环境中，前端文件首先由 `yarn build` 构建，并在8000端口上提供服务，因此，如果你想要进入Casdoor登录页面，你需要将Casdoor链接设置为 <https://your-casdoor-url.com:8000>（如果你正在使用反向代理，你需要将链接设置为你的域名）。

以我们的官方论坛Casnode为例

Casnode使用Casdoor处理身份验证。

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，它将会跳转到localhost 7001，这是Casdoor的端口。

当我们把Casnode放入生产环境时，我们将 `serverUrl` 设置为 <https://door.casdoor.com>，所以用户可以使用Casdoor登录或注册。

```
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  ...
11 }
```

(可选) 使用 Docker 运行

安装要求

硬件

如果你想自己构建Docker镜像，请确保你的机器至少有**2GB**的内存。Casdoor的前端是一个React的NPM项目。构建前端至少需要 **2GB** 的内存。内存少于**2GB**可能会导致前端构建失败。

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

操作系统

所有操作系统（Linux，Windows和macOS）都受到支持。

Docker

您可以在 Linux 中使用 Docker (docker-engine 版本 ≥ 17.05)，或在 Windows 和 macOS 中使用 Docker Desktop。

- [Docker](#)

无论操作系统如何，用户必须确保他们拥有docker-engine版本 ≥ 17.05 。这是因为我们在docker-compose.yml中使用了多阶段构建功能，该功能在17.05及以上版本中得到支持。有关更多信息，请参见<https://docs.docker.com/develop/develop-images/multistage-build/>。

如果您也在使用docker-compose，请确保您有docker-compose版本 ≥ 2.2 。对于

Linux用户，您还需要确保已安装docker-compose，因为它与docker-engine是分开的。

获取镜像

我们提供了两个DockerHub 镜像：

名称	描述	建议
casdoor-all-in-one	镜像中包含了Casdoor和MySQL数据库	此图像已包含一个玩具数据库，仅用于测试目的
casdoor	只有Casdoor被包含在图像中	此图像可以连接到您自己的数据库，并在生产中使用

1. [casbin/casdoor-all-in-one](#): 这个镜像包含了casdoor二进制文件，一个MySQL数据库，以及所有必要的配置。这是为了希望快速尝试Casdoor的新用户而设计的。有了这个镜像，您只需一两个命令就可以立即启动Casdoor，无需任何复杂的配置。然而，请注意，我们**不建议**在生产环境中使用此图像。

选项 1: 使用示例数据库

运行容器，并将 `8000` 端口暴露给主机。如果本地主机上不存在该图像，系统将会自动拉取。

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

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

```
admin
123
```

选项-2：尝试直接使用标准图像

提示

如果将配置文件挂载到容器不方便，使用环境变量也是一种可能的解决方案。

example

```
docker run \  
  -e driverName=mysql \  
  -e dataSourceName='user:password@tcp(x.x.x.x:3306)/' \  
  -p 8000:8000 \  
  casbin/casdoor:latest
```

创建 `conf/app.conf`。您可以从Casdoor的`conf/app.conf`中复制它。`有关 `app.conf``的更多详

然后运行

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

无论如何，只需将 `app.conf` 挂载到 `/conf/app.conf`，然后启动容器。

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

```
admin
123
```

选项-3：尝试使用docker-compose

在与 `docker-compose.yml` 文件相同的目录级别中创建一个 `conf/app.conf` 目录。然后，从Casdoor复制 `app.conf`。有关 `app.conf` 的更多详细信息，您可以查看[通过Ini文件](#)。

使用docker-compose创建一个单独的数据库：

```
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员账户：`built-in/admin` 登录Casdoor控制面板

```
admin
123
```

ⓘ 备注

如果你深入研究 `docker-compose.yml` 文件，我们创建的名为“`RUNNING_IN_DOCKER`”的环境变量可能会让你感到困惑。当通过 `docker-compose` 创建数据库 'db' 时，它在你的PC的 `localhost` 上可用，但不在 Casdoor 容器的 `localhost` 上可用。为了防止您因修改 `app.conf` 而遇到麻烦，这对新用户来说可能相当困难，我们提供了这个环境变量，并在 `docker-compose.yml` 中预先分配了它。当此环境变量设置为 `true` 时，`localhost` 将被替换为 `host.docker.internal`，以便 Casdoor 可以访问数据库。

(可选) 尝试使用 K8s Helm

介绍

现在我们展示如何使用 Helm 在 Kubernetes 上部署 Casdoor，以便于管理和扩展。

先决条件

- 一个正在运行的 Kubernetes 集群
- 已安装 Helm v3

安装步骤

步骤 1: 安装 Casdoor 图表

安装 Casdoor 图表:

```
helm install casdoor oci://registry-1.docker.io/casbin/casdoor-helm-charts --version 1.524.0
```

步骤 2: 访问 Casdoor

一旦安装，您可以通过 Kubernetes 集群提供的服务 URL 访问 Casdoor。

定制和配置

通过修改 Helm 图表值来定制您的 Casdoor 安装。有关详细选项，请参考图表中的 [values.yaml](#) 文件。可以配置以下参数。

参数	描述	默认值
<code>replicaCount</code>	运行 Casdoor 应用的副本数量。	1
<code>image.repository</code>	Casdoor Docker 图像的仓库。	<code>casbin</code>

参数	描述	默认值
<code>image.name</code>	Casdoor Docker 图像的名称。	<code>casdoor</code>
<code>image.pullPolicy</code>	Casdoor Docker 图像的拉取策略。	<code>IfNotPresent</code>
<code>image.tag</code>	Casdoor Docker 图像的标签。	<code>""</code>
<code>config</code>	Casdoor 应用的配置设置。	参见 config 字段
<code>database.driver</code>	要使用的数据库驱动程序（支持 mysql, postgres, cockroachdb, sqlite3）。	<code>sqlite3</code>
<code>database.user</code>	数据库用户名。	<code>""</code>
<code>database.password</code>	数据库密码。	<code>""</code>
<code>database.host</code>	数据库主机。	<code>""</code>
<code>database.port</code>	数据库端口。	<code>""</code>
<code>database.databaseName</code>	Casdoor 使用的数据库名称。	<code>casdoor</code>
<code>database.sslMode</code>	数据库连接的 SSL 模式。	<code>disable</code>
<code>service.type</code>	为 Casdoor 创建的 Kubernetes 服务的类型（ClusterIP, NodePort,	<code>ClusterIP</code>

参数	描述	默认值
	LoadBalancer 等)。	
<code>service.port</code>	Casdoor 服务的端口号。	<code>8000</code>
<code>ingress.enabled</code>	是否启用 Casdoor 的 Ingress。	<code>false</code>
<code>ingress.annotations</code>	Ingress 资源的注解。	<code>{}</code>
<code>ingress.hosts</code>	Ingress 资源的主机名。	<code>[]</code>
<code>resources</code>	Casdoor 容器的资源请求和限制。	<code>{}</code>
<code>autoscaling.enabled</code>	是否启用 Casdoor 的水平 Pod 自动扩展。	<code>false</code>
<code>autoscaling.minReplicas</code>	水平 Pod 自动扩展的最小副本数。	<code>1</code>
<code>autoscaling.maxReplicas</code>	水平 Pod 自动扩展的最大副本数。	<code>100</code>
<code>autoscaling.targetCPUUtilizationPercentage</code>	水平 Pod 自动扩展的目标 CPU 利用率百分比。	<code>80</code>
<code>nodeSelector</code>	Pod 分配的节点标签。	<code>{}</code>
<code>tolerations</code>	Pod 分配的容忍	<code>[]</code>

参数	描述	默认值
	标签。	
<code>affinity</code>	Pod 分配的亲和性设置。	<code>{}</code>
<code>extraContainersEnabled</code>	是否启用额外的边车容器。	<code>false</code>
<code>extraContainers</code>	额外的边车容器。	<code>""</code>
<code>extraVolumeMounts</code>	Casdoor 容器的额外卷挂载。	<code>[]</code>
<code>extraVolumes</code>	Casdoor 容器的额外卷。	<code>[]</code>
<code>envFromSecret</code>	从 secret 提供环境变量。	<code>[{name:"",secretName:"",key:""}]</code>
<code>envFromConfigmap</code>	从 configmap 提供环境变量。	<code>[{name:"",configmapName:"",key:""}]</code>
<code>envFrom</code>	从整个 secret 或 configmap 提供环境变量。	<code>`[{name:"",type:"configmap \`</code>

管理部署

升级您的 Casdoor 部署：

```
helm upgrade casdoor casdoor/casdoor-helm-charts
```

卸载 Casdoor：

```
helm delete casdoor
```

有关进一步的管理和定制，请参考 Helm 和 Kubernetes 的文档。

结论

使用 Helm 在 Kubernetes 上部署 Casdoor 简化了在 Kubernetes 环境中管理和扩展您的身份验证服务。

Casdoor 公共 API

Casdoor 前端网页用户界面是一个用 React 开发的SPA（单页应用程序）。React前端使用了Go后端代码暴露的Casdoor RESTful API。这个RESTful API被称为Casdoor Public API。换句话说，通过HTTP调用，你可以像Casdoor网页界面本身一样做任何事情。没有其他限制。以下人员可以利用该API：

- Casdoor的前端
- Casdoor 客户端 SDK（例如，casdoor-go-sdk）
- 来自应用程序一侧的任何其他自定义代码

Casdoor Public API的完整参考可以在Swagger上找到：<https://door.casdoor.com/swagger>。这些Swagger文档是使用Beego的Bee工具自动生成的。如果你想要自己生成Swagger文档，请参见：[如何生成swagger文件](#)

如何使用Casdoor Public API进行身份验证

1. 通过Access token

我们可以使用为经过身份验证的用户授予的访问令牌，以用户自身的身份调用Casdoor Public API。

如何获取访问令牌？

应用程序可以在OAuth登录过程结束时（也就是通过代码和状态获取令牌）获取Casdoor用户的访问令牌。API调用的权限将与用户的权限相同。

以下示例展示了如何通过casdoor-go-sdk在Go中调用GetOAuthToken()函数。

```
func (c *ApiController) Signin() {
    code := c.Input().Get("code")
    state := c.Input().Get("state")

    token, err := casdoorsdk.GetOAuthToken(code, state)
    if err != nil {
        c.ResponseError(err.Error())
        return
    }

    claims, err := casdoorsdk.ParseJwtToken(token.AccessToken)
    if err != nil {
        c.ResponseError(err.Error())
        return
    }
}
```

所有授予的访问令牌也可以通过管理员用户在Tokens页面上的web UI进行访问。例如，访问：<https://door.casdoor.com/tokens> 以查看演示站点。

如何进行身份验证？

1. HTTP `GET` 参数，其URL格式为：

```
/page?access_token=<访问令牌>
```

演示站点示例：https://door.casdoor.com/api/get-global-providers?access_token=eyJhbGciOiJSUzI1NiIs

2. HTTP Bearer令牌，HTTP头格式是：

```
Authorization: Bearer <访问令牌>
```

2. 通过 `Client ID` 和 `Client secret`

如何获取客户端ID和密钥？

应用程序编辑页面（例如，<https://door.casdoor.com/applications/casbin/app-vue-python-example>）将显示应用程序的客户端ID和密钥。当你想要以“机器”、“应用程序”或“服务”的身份而不是用户的身份调用API时，这种认证是有用的。API调用的权限将与应用程序（也就是组织的管理员）的权限相同。

以下示例展示了如何通过casdoor-go-sdk在Go中调用 `GetOAuthToken()` 函数。

如何进行身份验证？

1. HTTP `GET` 参数，其URL格式为：

```
/page?clientId=<客户端ID>&clientSecret=<客户端密钥>
```

演示站点示例：<https://door.casdoor.com/api/get-global-providers?clientId=294b09fbc17f95daf2fe&clientSecret=dd8982f7046ccba1bbd7851d5c1ece4e52bf039d>

2. HTTP 基本认证，其HTTP头格式为：

```
Authorization: Basic <客户端ID和客户端密钥以单个冒号":"连接的Base64编码>
```

如果你不熟悉Base64编码，你可以使用一个库来完成这个任务，因为 `HTTP Basic Authentication` 是一个被许多地方支持的流行标准。

3. 通过 Access key 和 Access secret

我们可以使用Casdoor用户的访问密钥和访问秘密来调用 Casdoor Public API 作为用户本身。访问密钥和访问秘密可以由管理员或用户本人在用户设置页面中配置。 update-user API也可以被调用来更新这些字段。API调用的权限将与用户的权限相同。

如何进行身份验证?

1. HTTP GET 参数, 其URL格式为:

```
/page?accessKey=<The user 's access key>&accessSecret=<the user's access secret>"
```

演示站点示例: <https://door.casdoor.com/api/get-global-providers?accessKey=b86db9dc-6bd7-4997-935c-af480dd2c796/admin&accessSecret=79911517-fc36-4093-b115-65a9741f6b14>

4. 通过 username 和 password

⚠ 注意事项

这种认证方法并不安全, 仅为了兼容性或演示目的而保留在此。我们建议使用前三种身份验证方法。

会发生什么?

用户凭证将会以 GET 参数的形式在请求URL中暴露出来。此外, 如果你使用的是HTTP而不是HTTPS, 用户的凭证将会被网络以明文形式嗅探。

我们可以使用Casdoor用户的用户名和密码来以用户自身的身份调用 Casdoor Public API。用户名采用的格式为 <用户的组织名称>/<用户名>。API调用的权限将与用户相同。

如何进行身份验证?

1. HTTP GET 参数, URL格式为:

```
/page?username=<用户的组织名称>/<用户名>&password=<用户的密码>"
```

演示站点示例: <https://door.casdoor.com/api/get-global-providers?username=built-in/admin&password=123>

教程

产品文档

产品	技术	文档
Dashboard of PingCAP TiDB	React + TypeScript + Go + Gin	Use Casdoor for TiDB Dashboard SSO sign-in (other languages: Chinese , Japanese)
GitLab	Vue + Ruby + Rails	OpenID Connect OmniAuth provider
Apache Shenyu	Java	Casdoor Plugin (other languages: Chinese)
Alist	TypeScript + SolidJS + Go + Gin	Casdoor SSO (other languages: Chinese)
BookStack	jQuery + Bootstrap + Go + Beego	Casdoor integrates registration and login

文章

技术	语言	标题
ASP.NET Core 6	英语	使用本地Casdoor Docker容器在Linux的Windows子系统上进行ASP.NET Core .NET 6演示身份验证项目
OAuth2 Proxy (Go)	中文	Use Casdoor + OAuth-Proxy to protect web applications on public networks
Casnode (JavaScript + React + Go + Beego)	中文	使用Lighthouse设置一个像V2ex一样的论坛
Clou dreve (Go)	中文	Modify Clou dreve to support Casdoor
KodExplorer (PHP)	中文	Modify KodExplorer to support Casdoor



Deployment

部署到NGINX

使用Nginx反向代理您的后端Go程序，并快速启动Casdoor服务。

部署到 Kubernetes

学习如何在 Kubernetes 集群中部署 Casdoor

数据初始化

如何从文件初始化Casdoor 数据

在CDN中托管静态文件

在CDN中托管前端静态文件

在内部网络中托管静态文件

如何部署Casdoor静态资源

数据库迁移

处理Casdoor中的数据库迁移

部署到NGINX

尽管Casdoor遵循前后端分离的架构，但在生产环境中，后端程序仍为前端文件提供静态文件服务。因此，您可以使用像Nginx这样的反向代理软件来代理Casdoor域的所有流量，并将其重定向到后端Go程序监视的端口。

在本章中，您将学习如何使用Nginx反向代理您的后端Go程序，并快速启动Casdoor服务。

1. 构建前端静态文件

假设您已经下载了Casdoor并完成了必要的配置（如果没有，请参考[开始部分](#)），您只需要按照以下方式构建静态文件：

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，先构建它：

```
go build && ./main
```

3. 配置并运行Nginx

```
vim /path/to/nginx/nginx.conf
```

然后，添加一个服务器：

```
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;
    }
}
```

接下来，重启你的Nginx进程。运行：

```
nginx -s reload
```

4. 测试

在你最喜欢的浏览器中访问 `http://YOUR_DOMAIN_NAME`。

部署到 Kubernetes

在 Kubernetes (k8s) 中部署 Casdoor

我们提供了一个在 Kubernetes 集群中部署 Casdoor 的基本示例。在 Casdoor 的根文件夹中，你会找到一个名为 "k8s.yaml" 的文件。此文件包含了在 Kubernetes 中部署 Casdoor 的示例配置，包括一个部署和一个服务。

在开始部署之前，请确保你已经修改了 `conf/app.conf` 文件，以便 Casdoor 可以成功连接到数据库，并且数据库本身正在运行。另外，请确保 Kubernetes 能够拉取必要的镜像。

要部署 Casdoor，请运行以下命令：

```
kubectl apply -f k8s.yaml
```

你可以通过运行 `kubectl get pods` 命令来检查部署状态。

这是 `k8s.yaml` 的内容：

```
# this is only an EXAMPLE of deploying casdoor 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
```

请注意，这个文件只是一个示例。你可以根据你的需求进行各种修改，例如使用不同的命名空间、服务类型，或者使用 `ConfigMap` 来挂载配置文件。在 `Kubernetes` 中，使用 `ConfigMap` 挂载配置文件是生产环境中推荐的方法。

数据初始化

如果你正在将Casdoor与其他服务一起部署为一个完整的应用，你可能希望为用户提供一个开箱即用的功能。这意味着用户可以直接使用应用程序，无需任何配置。

在这种情况下，您可以使用数据初始化通过配置文件在Casdoor中注册您的服务。此文件可以预先定义，或由您自己的服务动态生成。

Here we give a tutorial for importing or exporting config data.

Import Config Data

如果在Casdoor的根目录中有一个名为 `init_data.json` 的配置文件，它将被用来初始化Casdoor中的数据。你只需要将此文件放在Casdoor将要运行的根目录中。

如果您正在使用Casdoor的官方Docker镜像，这里有一些脚本可以帮助您将 `init_data.json` 挂载到容器中。

A template for `init_data.json` is provided at: [init_data.json.template](#). Rename it to `init_data.json` before using it.

For Docker

如果你使用Docker部署Casdoor，你可以使用 `volume` 命令将 `init_data.json` 挂载到容器中。

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

For Kubernetes

如果你使用Kubernetes部署Casdoor，你可以使用 `configmap` 来存储 `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。您可以按照以下方式修改您的 `deployment`：

```
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
```

Export Config Data

You can also export all of Casdoor configuration data into a file for data migration.

A Go test named `TestDumpToFile()` is provided at: [init_data_dump_test.go](#)

```
go test ./object -v -run TestDumpToFile
```

After running this Go test, a file named `init_data_dump.json` will be generated in same directory. This file contains your full Casdoor configuration data. If you want to migrate the data into another Casdoor instance, just rename `init_data_dump.json` to `init_data.json` and move it to root directory of target Casdoor folder.

References

All Casdoor objects supported by the data initialization are as follows:

对象	Go 结构体	文档
组织机构	struct	doc
应用	struct	doc
用户	struct	doc
证书	struct	

对象	Go 结构体	文档
提供商	struct	doc
Idaps	struct	doc
models	struct	
permissions	struct	doc
payments	struct	doc
products	struct	doc
resources	struct	doc
roles	struct	doc
syncers	struct	doc
tokens	struct	doc
webhooks	struct	doc
groups	struct	doc
adapters	struct	doc
enforcers	struct	
plans	struct	doc

对象	Go 结构体	文档
pricings	struct	doc
invitations	struct	doc
records	struct	
sessions	struct	
subscriptions	struct	doc
transactions	struct	

如果你对填写这个模板仍然感到困惑，你可以调用RESTful API，或者使用浏览器的调试模式来查看 `GetXXX` 对这些对象的响应。响应的格式与 `init_data.json` 相同。

在CDN中托管静态文件

前端静态资源，如 .js 和 .css 文件，位于 `web/build/static/`。如果您希望在公共云的CDN服务中部署这些文件，Casdoor提供了一个简化部署过程的脚本。请按照以下步骤操作：

📘 备注

我们假设您已经构建了Casdoor的前端代码。如果您还没有，请参考[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

在填写 `Domain` 字段时，确保以 `/` 结束。

Domain ? :

`https://cdn.casbin.com/casdoor/`

使用说明

该脚本可以在[deployment/deploy_test.go](#)找到。

在[deploy_test.go](#)中，您需要修改 `GetProvider()` 中的 `id` 参数。提供者 `id` 的格式是 `<owner>/<name>`。

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

在进行必要的修改后，使用以下命令来运行脚本：

```
cd deployment
go test
```

如果执行成功，你将会看到：

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

工作原理

脚本的功能：

- 将 `css/` 和 `js/` 文件夹中的所有文件上传到存储提供商指定的CDN服务。
- 将 `web/build/index.html` 中的所有 `.css` 和 `.js` 文件的URL替换为CDN中托管的URL。

你仍然需要保留 `index.html` 文件。在静态文件被上传到CDN后，`index.html` 仍然会通过Casdoor的Go后端被用户请求，而CDN中的静态文件将通过在 `index.html` 中提供的URLs被请求。

在内部网络中托管静态文件

如果您在**内网**上部署Casdoor，您可能无法直接通过互联网访问静态资源。您需要部署可以访问的静态资源，然后在Casdoor中修改三处配置。

部署静态资源

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

克隆该仓库并在网页服务器上**部署**它。确保您可以访问这些资源。

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。前往 [conf/app.conf](#) 并将 `staticBaseUrl` 设置为您的部署地址。

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

数据库迁移

在升级数据库时，存在数据丢失的风险，例如在删除旧字段时。幸运的是，Casdoor利用了xorm，它可以帮助解决许多数据库迁移问题。然而，一些模式和数据迁移仍然需要手动处理，例如在字段名称更改时。

📌 备注

参考[xorm文档](#)以更好地理解xorm的模式操作。

工作原理

如前所述，xorm无法处理字段名称的更改。为了解决这个问题，xorm提供了一个[migrate](#)包，可以帮助解决这个问题。

要处理字段重命名，你可以编写如下代码：

```
migrations := []*migrate.Migration{
    {
        ID: "CasbinRule--fill ptype field with p",
        Migrate: func(tx *xorm.Engine) error {
            _, err :=
tx.Cols("ptype").Update(&xormadapter.CasbinRule{
                Ptype: "p",
            })
            return err
        },
        Rollback: func(tx *xorm.Engine) error {
            return tx.DropTable(&xormadapter.CasbinRule{})
        }
    }
}
```

我们的目标是将 `p_type` 重命名为 `ptype`。然而，由于 `xorm` 不支持字段重命名，我们必须采取更复杂的方法：将 `p_type` 的值赋给 `ptype`，然后删除 `p_type` 字段。

`ID` 字段唯一标识正在执行的迁移。在 `m.Migrate()` 运行后，`ID` 的值将被添加到数据库的迁移表中。

在再次启动项目时，数据库将检查表中是否存在任何现有的 `ID` 字段，并避免执行与相同 `ID` 相关的任何操作。



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 SDKs

使用Casdoor SDKs而不是标准的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

4 个项目

移动 SDKs

1 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。

Next.js

在 Next.js 项目中使用 Casdoor

Nuxt

在Nuxt项目中使用Casdoor

OAuth 2.0

使用AccessToken验证客户端

使用Casdoor作为CAS服务器

如何将Casdoor用作CAS服务器

SAML

6 个项目

Face ID

Use Face ID to log in in Casdoor

WebAuthn

在 Casdoor 中使用webauthn

概览

在本节中，我们将向您展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

作为身份提供商 (Idp)，Casdoor 支持四项认证协议：

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

OAuth 2.0 (OIDC)

什么是 OAuth 2.0?

OAuth 2是一个授权框架，它使得应用程序——如Facebook，GitHub和Casdoor——能够在HTTP服务上获得对用户账户的有限访问权限。它的工作原理是将用户身份验证委托给托管用户帐户的服务，并授权第三方应用程序访问该用户帐户。OAuth 2为网页和桌面应用程序，以及移动设备提供授权流程。

Casdoor的授权程序基于OAuth 2.0协议。我们推荐使用 OAuth 2.0 协议，原因如下：

1. 该协议简单易行，能解决多种问题。
2. 该协议成熟度高且社区支持广泛。

如此，您的应用程序将通过 OAuth 2.0 (OIDC) 与 Casdoor 通讯。这里有三种方式连接到 Casdoor:

标准 OIDC 客户端

标准OIDC 客户端: 使用一个标准的 OIDC 客户端实现，通常在各类编程语言或框架都广泛提供。

什么是OIDC?

OpenID Connect (OIDC) 是一个在OAuth 2.0 框架顶端运行的开放身份验证协议。针对消费者，OIDC允许个人通过单点登录(SSO)访问使用OpenID提供商(OPs)的依赖方站点，如电子邮件提供商或社交网络平台，以验证其身份。它向应用程序或服务提供用户信息、认证背景，并允许访问用户个人资料。

Casdoor 完全支持 OIDC 协议。如果您的应用程序已经通过**标准的OIDC客户端库**使用了另一个OAuth 2.0 (OIDC) 身份提供商，并且您想迁移到Casdoor，使用OIDC发现将使切换到Casdoor变得非常**简单**。

Casdoor SDKs

Casdoor SDKs: 对于大多数编程语言，Casdoor在OIDC之上提供了易于使用的SDK库，其中包含了只在Casdoor中才有的扩展功能。

与标准的OIDC协议相比，Casdoor的SDK提供了更多的功能，如用户管理和资源上传等。通过Casdoor SDK连接到Casdoor需要的时间比使用标准的OIDC客户端库要多，但它提供了最佳的**灵活性**和最**强大的API**。

Casdoor插件

Casdoor 插件：如果您的应用程序是基于流行平台（如 Spring Boot, WordPress 等）构建的，并且 Casdoor（或第三方）已经为其提供了插件或中间件，那么您应该使用它。使用插件比手动调用Casdoor SDK要容易得多，因为前者是专为平台制作的。

插件：

- [Jenkins 插件](#)
- [APISIX 插件](#)

中间件：

- [Spring Boot插件](#)
- [Django 插件](#)

SAML

什么是SAML?

安全鉴别标记语言(SAML)是一种开放标准，允许身份提供者(IDP)将授权证书传给服务提供者。这个术语的意思是，你可以使用一套凭证登录到许多不同的网站。管理每个用户的一个登录要比管理电子邮件、客户关系管理（CRM）软件、活动目录等的单独登录要简单得多。

SAML交易使用可扩展标记语言(XML) 在标识提供者和服务提供者之间进行标准化通信。SAML是用户身份认证和使用服务授权之间的链接。

Casdoor可以用作SAML IdP。目前，Casdoor支持SAML 2.0的主要功能。有关更多详

细信息，请参见[SAML](#)。

示例:

[Casdoor](#)作为Keycloak中的SAML IdP

建议:

1. 该协议是**强大的**，覆盖了许多场景，使其成为最全面的SSO协议之一。
2. 该协议是**大型的**，带有许多可选参数，因此在实际实施中很难100%覆盖所有应用场景。
3. 如果应用程序是**新**开发的，由于其高度的技术复杂性，不建议使用SAML。

CAS

什么是CAS?

中央认证服务（CAS）是网络的单点登录协议。它的目的是允许用户在只提供一次凭证（如用户ID和密码）的情况下访问多个应用程序。它还允许网络应用程序在不获取用户的安全凭证（如密码）的情况下对用户进行身份验证。

Casdoor已实现了CAS 1.0、2.0和3.0的功能。有关更多详细信息，请参见[CAS](#)。

建议:

1. 该协议本身相对轻量级且易于实现，但它只能解决单一场景。
2. CAS 客户端和CAS服务器之间的相互信任是通过在没有任何加密或签名机制的情况下使用接口建立的，目的是确保进一步的安全。
3. CAS协议并无优于其他协议的优点。

集成表

一些应用程序已有连接到Casdoor的示例。您可以按照文档指示操作，快速连接到Casdoor。您可以在[集成表](#)中查看所有应用程序。

标准OIDC 客户端

OIDC 发现

Casdoor已完全实现了OIDC协议。如果您的应用程序已经使用标准的OIDC客户端库连接到另一个OAuth 2.0身份提供商，并且您想要迁移到Casdoor，使用OIDC发现将使您非常容易切换。Casdoor's OIDC discovery URL 是：

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

例如，演示站点的OIDC发现URL是：<https://door.casdoor.com/.well-known/openid-configuration>，它包含以下信息：

```
{
  "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",
    "token id_token",
```

OIDC客户端库列表

以下是一些适用于Go和Java等语言的OIDC客户端库的列表：

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

请注意，上述表格并非详尽无遗。要获取完整的OIDC客户端库列表，您可以在以下位置找到更多详细信息：

1. <https://oauth.net/code/>
2. <https://openid.net/certified-open-id-developer-tools/>

OIDC UserInfo字段

以下表格说明了如何将OIDC UserInfo字段（通过 `/api/userinfo` API）从Casdoor的用户表的属性中映射出来：

Casdoor 用户字段	OIDC用户信息字段
Id	sub

Casdoor 用户字段	OIDC用户信息字段
originBackend	iss
Aud	aud
Name	preferred_username
DisplayName	name
Email	email
Avatar	picture
Location	address
Phone	phone

您可以在[这里](#)查看UserInfo的定义。

Casdoor SDKs

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。

💡 提示

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](#)

移动SDK	描述	SDK 代码库	示例
Android SDK	适用于Android应用	casdoor-android-sdk	casdoor-android-example
iOS SDK	针对iOS应用程序	casdoor-ios-sdk	casdoor-ios-example
React Native SDK	对于React Native应用程序	casdoor-react-native-sdk	casdoor-react-native-example
Flutter SDK	适用于Flutter应用	casdoor-flutter-sdk	casdoor-flutter-example
Firebase SDK	适用于Google Firebase应用		casdoor-firebase-example
Unity游戏SDK	适用于Unity 2D/3D PC/移动游戏	casdoor-dotnet-sdk	casdoor-unity-example
uni-app SDK	对于uni-app应用程序	casdoor-uniapp-sdk	casdoor-uniapp-example

桌面版SDK	描述	SDK代码	示例
Electron SDK	对于Electron应用	casdoor-js-sdk	casdoor-electron-example
.NET 桌面 SDK	适用于 .NET 桌面应用	casdoor-dotnet-sdk	WPF: casdoor-dotnet-desktop-example WinForms: casdoor-dotnet-winform-example Avalonia UI: casdoor-dotnet-avalonia-example
C/C++ SDK	用于C/C++桌面应用程序	casdoor-cpp-sdk	casdoor-cpp-qt-example

Web前端SDK	描述	SDK 代码	示例代码
Javascript SDK	对于传统的非SPA网站	casdoor-js-sdk	Nodejs 后端: casdoor-raw-js-example Go 后端: casdoor-go-react-sdk-example
仅前端的SDK	仅用于前端的SPA网站	casdoor-js-sdk	casdoor-react-only-example

Web前端SDK	描述	SDK 代码	示例代码
React SDK	适用于React网站	casdoor-react-sdk	Nodejs 后端: casdoor-nodejs-react-example Java 后端: casdoor-spring-security-react-example
Next.js SDK	针对Next.js网站		nextjs-auth
Nuxt SDK	对于Nuxt网站		nuxt-auth
Vue SDK	For Vue websites	casdoor-vue-sdk	casdoor-python-vue-sdk-example
Angular SDK	For Angular websites	casdoor-angular-sdk	casdoor-nodejs-angular-example
Flutter SDK	For Flutter Web websites	casdoor-flutter-sdk	casdoor-flutter-example
ASP.NET SDK	For ASP.NET Blazor WASM websites	Blazor.BFF.OpenIDConnect.Template	casdoor-dotnet-blazorwasm-oidc-example
Firebase SDK	For Google Firebase apps		casdoor-firebase-example

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:

Web后端 SDK	描述	Sdk 代码	示例代码
Go SDK	对于Go后端	casdoor-go-sdk	casdoor-go-react-sdk-example
Java SDK	针对Java后端	casdoor-java-sdk	casdoor-spring-boot-starter , casdoor-spring-boot-example , casdoor-spring-security-react-example
Node.js SDK	针对Node.js后端	casdoor-nodejs-sdk	casdoor-nodejs-react-example
Python SDK	针对Python后端	casdoor-python-sdk	Flask: casdoor-python-vue-sdk-example Django: casdoor-django-js-sdk-example FastAPI: casdoor-fastapi-js-sdk-example
PHP SDK	针对PHP后端	casdoor-php-sdk	wordpress-casdoor-plugin
.NET SDK	针对ASP.NET后端	casdoor-dotnet-sdk	casdoor-dotnet-sdk-example
Rust SDK	针对Rust后端	casdoor-rust-sdk	casdoor-rust-example
C/C++ SDK	针对 C/C++ 后端	casdoor-cpp-sdk	casdoor-cpp-qt-example
Dart SDK	针对Dart后端	casdoor-dart-sdk	
Ruby SDK	针对Ruby后端	casdoor-ruby-sdk	

要查看官方Casdoor SDK的完整列表，请参阅：<https://github.com/orgs/casdoor/repositories?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。以casdoor-go-sdk为例：<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

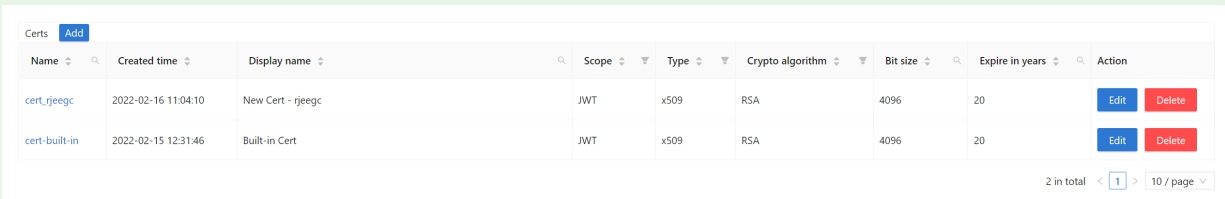
func init() {
    auth.InitConfig(CasdoorEndpoint, ClientId, ClientSecret, JwtPublicKey, CasdoorOrganization, CasdoorApplication)
}
```

`InitConfig()` 的所有参数解释为：

参数	必须	描述
端点	是	Casdoor 服务器 URL，例如 <code>https://door.casdoor.com</code> 或 <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应用程序的证书的公钥
组织名称	是	Casdoor组织的名称
应用程序名称	不	Casdoor应用程序的名称

提示

`jwtPublicKey` 可以在 `Certs` 页面中进行管理。



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。

Public key

Copy public key Download public key

```
-----BEGIN CERTIFICATE-----
MIIE+TCCAUGgAwIBAgIDAAJAAAGCQsG5ib3DQEBCWUAMDYxHTAB8gNBaoTFENh
c2Rvb3I3LnYw5pemF0aW5uMRUwEwYwVWVWVWVWVWVWVWVWVWVWVWVWVWVWVW
MDE1MDgMTUyWhcNNDExMDE1MDgMTUyWjA2MRowGwYwVWVWVWVWVWVWVWVWVW
ZZFuaXphdGlvbGVVbWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVWVW
AQEFAAOCAg8AMICCGKCAgEAsinpb5E1jymOf1RfSDSSSE8IR7Y+lw+RJJ74e5e
rq4b8zMYk7HeHCyZr/hmNEwEVXnhXu1P0mBeQ5ypp/QG08vgEmjAETNmzk1NJOQ
CjCYwUrasO/f/Mn1/CQ13xv6mV1KHZj5rKsMhY1vaxTEP3+VB8Hjg3MHFWrb07
uvFMCle5W8+0rKEZCKTR8+9VB3janeBz//zQePFVh79bZate/hLrPK0Go9P1g
OwvtoC1A3sarHTP4Qm/LQR0tHqZfYbdy5pyWAQhNaDFE7mTstRSBb/wLJNCUBD
PTSLVjC04WISf6NkX0Z7KvmbPst5+btvcqsvRAGvdsB9H62Kpts1Yh7GAuo
13qt4zoKbiURyXkQjXvwcQsEftUkSew5zupSIDRLoByQTLbX0qLAFNFW3g/
pzSDjgg/60d6HTmbZni45mjdyFhXCDb1Kn7N+xtojnfaNkwep2REV+RmC0fx4Gu
hrsnLsmkmUDejZ9aB9L9oj11YEQIMZJZEq+RVUx+wB4y8K/ID1bcY+fnG58Pw
IDp5262boq45RSv3Z7bB0w4ZxvOj/1VLoRtjPbLI0bhfR/AeZMHPkOXvzf4
yE+haqz68wdF0VR9xYc/RBSAF7323OsjYjEgInU/RohrRgCjpk/Mt2Kt84k0
wn8CAwEAAMQMA4wDAVDVROTAQH/BAlwADANBgkqhkiG9w0BAQsFAAOCAgEAnZf
DKLX+F1vKRO/5g+Pir0P5NKuQkmwH97b8CS2g51phDyNgic4/L5dzuf4Awe6ve
C06IvdW5iis8PUPdjmT2uMPSNjwLxG3QsnimMURNwFLTRem/HejeZgur91JM
8haawdSdJH2RgmFoDeE2r8NVRfhr8KncO1ddTJKuS1N0/HZ221W4j4nzCvI
2nR42FybaP3O/g2JXmNNR0wZmNjggsF7XVENCsUFO1jYwLaquXcg54IL7XVLG
omKNNcc8h1FCeKj/nmbGMh0dnFWKDTskbNmcOPN06ixzqMy/Hq+cMwYv7maAG
Jtevs3qgMZ8F9Qzr3HjUcGR3ZYWYDy/xxPisukfOPZgH97Kuf9w0PwPn0BLqL
2D1JzaBmjGJohv7XNNKLUIDYw85ZTQ5b9dI4e+6bmyWqQlwt+atiwFEV
XcJ70B4IALX6xau1kLEpv901GEGerizRz5P9NINA7koOSAVmP9x0DQTKt+LbXnZE
HhNkYk8yHQZf9R7YBGLv/Ac6nir5Ua15OqjJ8dLRz/verfGo2yZst+hkVU5
nCCJHbcyFnm1hdvvdEdH33jDBjNB6cioUzr/3VYaiWsaIAdosHAgMwXUwWP+h
8XKxmzkHbTMYZPDps55ak+S4Q9wb8RRAyo=
-----END CERTIFICATE-----
```

Private key

Copy private key Download private key

```
-----BEGIN PRIVATE KEY-----
MIUKQIBAAKCAgEAsinpb5E1jymOf1RfSDSSSE8IR7Y+lw+RJJ74e5ejr4b8zMY
k7HeHCyZr/hmNEwEVXnhXu1P0mBeQ5ypp/QG08vgEmjAETNmzk1NJOQCYwUra
sO/f/Mn1/CQ13xv6mV1KHZj5rKsMhY1vaxTEP3+VB8Hjg3MHFWrb07uvFMCle5
W8+0rKEZCKTR8+9VB3janeBz//zQePFVh79bZate/hLrPK0Go9P1gOwvtoC1A
3sarHTP4Qm/LQR0tHqZfYbdy5pyWAQhNaDFE7mTstRSBb/wLJNCUBDPTSLVjC0
4WISf6NkX0Z7KvmbPst5+btvcqsvRAGvdsB9H62Kpts1Yh7GAuo13qt4zoKbiUR
yXkQjXvwcQsEftUkSew5zupSIDRLoByQTLbX0qLAFNFW3g/pzSDjgg/60d6HTmb
Zni45mjdyFhXCDb1Kn7N+xtojnfaNkwep2REV+RmC0fx4GuhRsnLsmkmUDejZ9a
B9L9oj11YEQIMZJZEq+RVUx+wB4y8K/ID1bcY+fnG58PwIDp5262boq45RSv3Z
7bB0w4ZxvOj/1VLoRtjPbLI0bhfR/AeZMHPkOXvzf4yE+haqz68wdF0VR9xYc/
RBSAF7323OsjYjEgInU/RohrRgCjpk/Mt2Kt84k0wn8CAwEAAMQMA4wDAVDVRO
TAQH/BAlwADANBgkqhkiG9w0BAQsFAAOCAgEAnZfDKLX+F1vKRO/5g+Pir0P5NK
uQkmwH97b8CS2g51phDyNgic4/L5dzuf4Awe6veC06IvdW5iis8PUPdjmT2u
MPSNjwLxG3QsnimMURNwFLTRem/HejeZgur91JM8haawdSdJH2RgmFoDeE2r8
NVRfhr8KncO1ddTJKuS1N0/HZ221W4j4nzCvI2nR42FybaP3O/g2JXmNNR0w
ZmNjggsF7XVENCsUFO1jYwLaquXcg54IL7XVLGomKNNcc8h1FCeKj/nmbGMh0
dnFWKDTskbNmcOPN06ixzqMy/Hq+cMwYv7maAGJtevs3qgMZ8F9Qzr3HjUcGR
3ZYWYDy/xxPisukfOPZgH97Kuf9w0PwPn0BLqL2D1JzaBmjGJohv7XNNKLUID
Yw85ZTQ5b9dI4e+6bmyWqQlwt+atiwFEVXcJ70B4IALX6xau1kLEpv901GEGer
izRz5P9NINA7koOSAVmP9x0DQTKt+LbXnZEHhNkYk8yHQZf9R7YBGLv/Ac6nir
5Ua15OqjJ8dLRz/verfGo2yZst+hkVU5nCCJHbcyFnm1hdvvdEdH33jDBjNB6
cioUzr/3VYaiWsaIAdosHAgMwXUwWP+h8XKxmzkHbTMYZPDps55ak+S4Q9wb8
RRAyo=
-----END PRIVATE KEY-----
```

Save
Save & Exit

之后，您可以在应用编辑页面选择证书。

Edit Application Save Save & Exit

Name :

Display name :

Logo :

Preview:

Home :

Description :

Organization :

Client ID :

Client secret :

Cert :

Redirect URLs :

Action

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?");
}

export function signin() {
  return CasdoorSdk.signin(ServerUrl);
}

export function showMessage(type, text) {
  if (type === "") {
    return;
  } else if (type === "success") {
    message.success(text);
  } else if (type === "error") {
    message.error(text);
  }
}

export function goToLink(link) {
  window.location.href = link;
}

```

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";

```

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。要查看一个真实的Casnode示例（一个用Go编写的论坛网站），请参见：<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.SetSessionUser(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。

这些函数是通过调用 [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。 您希望用户只需在组织中的任何一个应用中登录一次，然后在转到另一个应用时就能够登录，无需任何额外的点击。

我们提供了这个单点登录功能。 要启用它，你只需要：

- 启用自动登录按钮。
- 填写主页的URL。
- 在应用主页添加一个**静默登录**功能。

📌 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。

启用自动登录后，将不会显示选择框，已登录的用户将直接登录。

配置

1. 填写“主页”字段。 它可以是应用程序的主页或登录页面。

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 ? :

添加静默登录

实际上，我们通过URL中携带参数来实现自动登录。因此，你的应用需要有一种方法在跳转到URL后触发登录。我们提供了[casdoor-react-sdk](#)来帮助你快速实现这个功能。你可以在[use-in-react](#)中看到详细信息。

❗ 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. 在你的主页中，通过检查 `silentSignin` 参数来确定是否需要静默（自动）登录。如果 `silentSignin === 1`，函数应返回 `SilentSignin` 组件，该组件将帮助你发起登录请求。由于你已启用自动登录，用户将在不点击的情况下自动登录。

添加弹出登录

“弹出登录”功能将打开一个小窗口。在子窗口中登录Casdoor后，它将向主窗口发送身份验证信息，然后自动关闭。我们通过URL中携带参数来实现这个功能。

❗ 信息

如何使用

使用[casdoor-js-sdk](#)中的 `popupSignin()` 方法来快速实现这个功能。你可以在[casdoor-nodejs-react-example](#)中看到一个演示。

工作原理

1. 在到应用主页的URL中，我们将携带 `popup` 参数。
2. 当 `popup=1` 在登录参数中时，Casdoor 将把 `code` 和 `state` 作为消息发送到主窗口，并在主窗口中使用 SDK 完成 `token` 的获取。

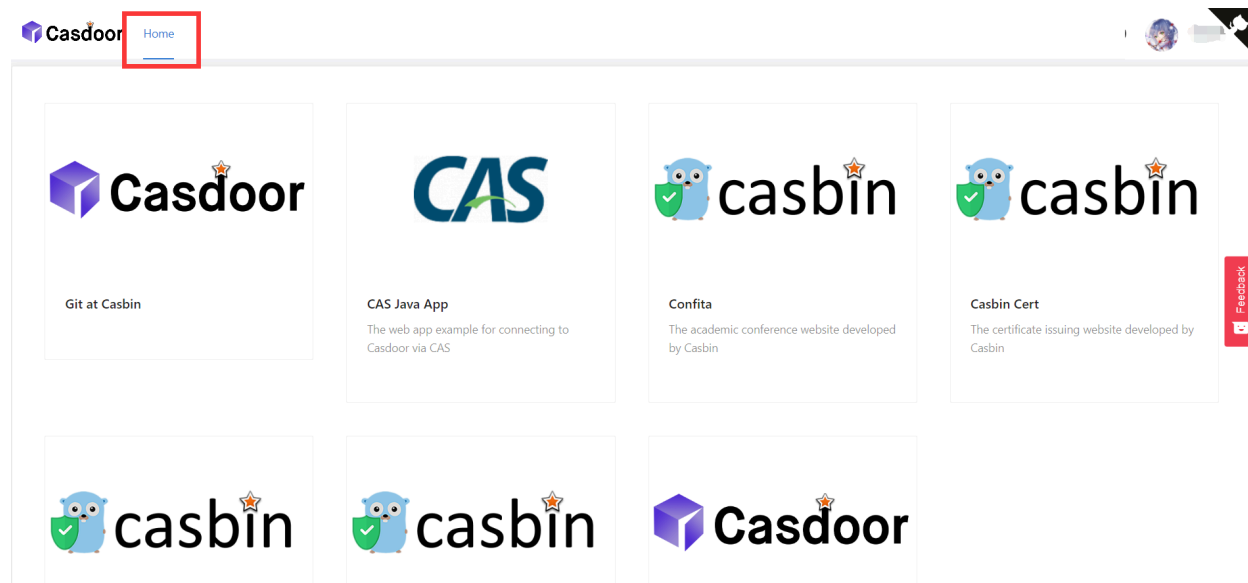
使用 SSO

配置已完成。下面，我们将向你展示如何使用自动登录。

! 信息

确保你的应用可以重定向到用户的个人资料页面。我们在每种语言的 SDK 中都提供了 `getMyProfileUrl(account, returnUrl)` API。

打开个人资料页面，然后转到“主页”页面（/ URL 路径）。你将看到由组织提供的应用列表。值得注意的是，只有“内置”以外的组织中的用户才能在“主页”上看到应用列表。所有全局管理员（在“内置”组织中的人）都看不到它。



点击应用列表中的一个图块，它将跳转到带有GET参数 `?silentSignin=1` 的该应用的主页URL。如果应用已经与Casdoor SSO集成（因此它会识别 `?silentSignin=1` 参数并在后台执行静默登录），它将自动登录到应用。

Vue SDK

Casdoor Vue SDK是为Vue 2和Vue 3设计的，使其使用起来非常方便。

Vue SDK基于casdoor-js-sdk。您也可以直接使用casdoor-js-sdk，这将允许更多的自定义。

请注意，这个插件仍在开发中。如果您有任何问题或建议，请随时通过打开一个[问题](#)与我们联系。

我们现在将向您展示下面的必要步骤。

如果你在阅读README.md后仍然不确定如何使用它，你可以参考这个例子：[casdoor-python-vue-sdk-example](#)以获取更多详情。

该示例的前端使用casdoor-vue-sdk构建，而后端使用casdoor-python-sdk构建。您可以在示例中查看源代码。

安装

```
# NPM
npm install casdoor-vue-sdk

# Yarn
yarn add casdoor-vue-sdk
```

初始化SDK

要初始化SDK，您需要按照以下顺序提供5个字符串参数：

名称	必需的	描述信息
服务器Url	是	您的Casdoor服务器的URL。
客户端Id:	是	您的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应用程序关联的Casdoor组织的名称。
重定向路径	否	您的Casdoor应用程序的重定向URL路径。如果未提供，它将默认为 <code>/callback</code> 。

对于 Vue 3:

```
// 在 main.js 中
import Casdoor from 'casdoor-vue-sdk'

const config = {
  serverUrl: "http://localhost:8000",
```

对于 Vue 2:

```
// 在 main.js 中
import Casdoor from 'casdoor-vue-sdk'
import VueCompositionAPI from '@vue/composition-api'

const config = {
  serverUrl: "http://localhost:8000",
  clientId: "4262bea2b293539fe45e",
  organizationName: "casbin",
  appName: "app-casnode",
  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();
    }
  }
}
```


自动修复

如果 `postinstall` 钩子没有被触发，或者你已经更新了Vue版本，尝试运行以下命令来解决重定向问题：

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

有关切换Vue版本的更多信息，请参考[vue-demi文档](#)。

桌面 SDK

Casdoor的Electron应用示例

这是一个Electron应用程序示例，演示了Casdoor的集成能力。

dotNET 桌面应用

一个用于Casdoor的dotNET桌面应用示例

移动SDKs .NET MAUI应用

Casdoor的.NET MAUI应用示例

Qt 桌面应用程序

一个用于Casdoor的Qt桌面应用示例

Casdoor的Electron应用示例

一个Electron应用示例，演示了Casdoor的集成能力。

如何运行示例

初始化

您需要初始化6个参数，所有这些参数都是字符串类型：

名称	描述	路径
serverUrl	您的Casdoor服务器URL	src/App.js
clientId	您的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	如果未提供，您的Casdoor应用程序的重定向URL路径将为 /callback	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
```

构建电子应用并运行此应用。

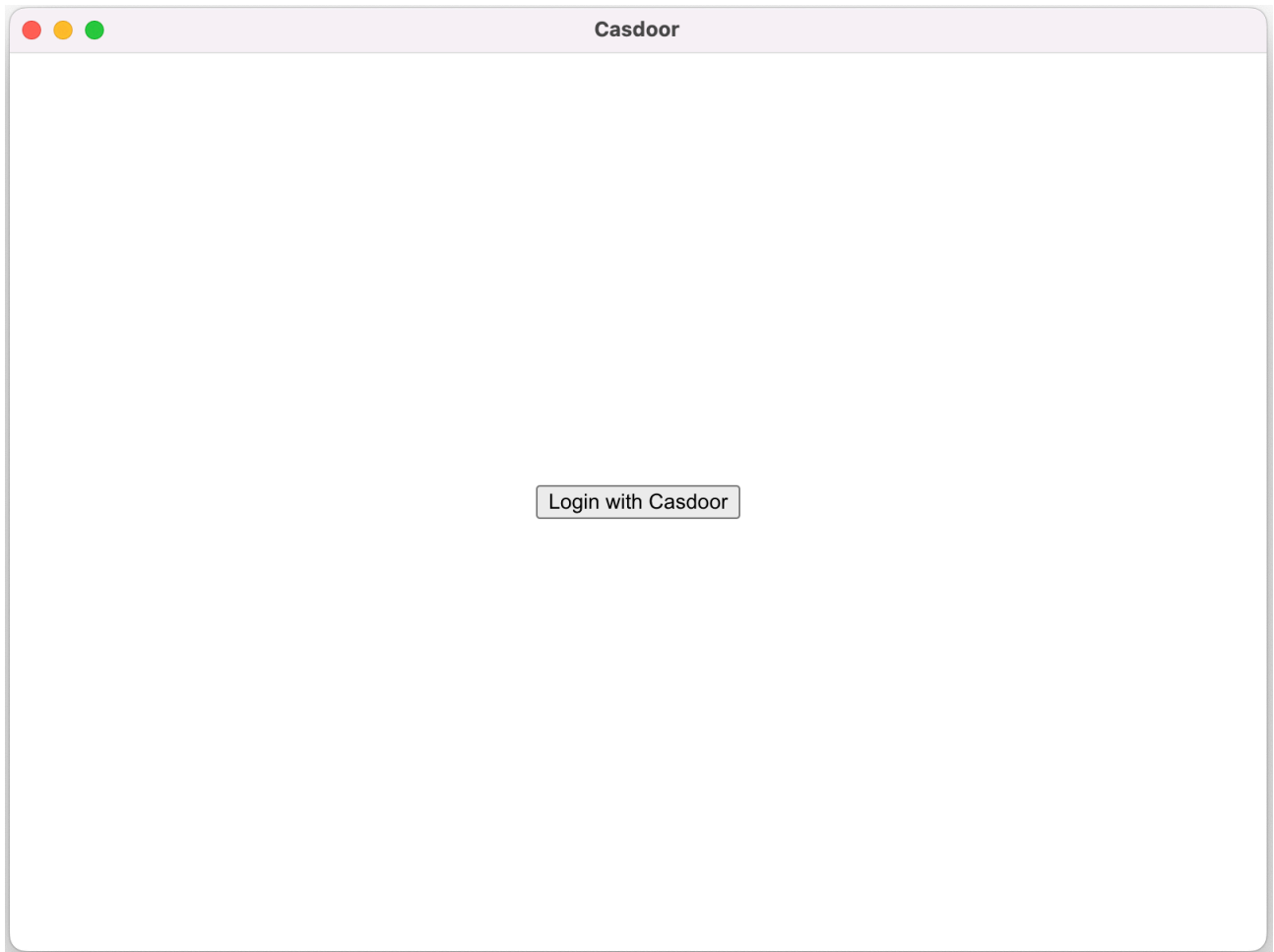
```
npm run make 或者 yarn make
```

打包并分发您的应用程序。它将创建 `out` 文件夹，您的软件包将被定位于：

```
// 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

一旦你运行这个Electron应用程序，一个新的窗口将会出现在你的桌面上。



如果你点击Login with Casdoor按钮，你的默认浏览器将自动打开并显示登录页面。



Auto sign in [Forgot password?](#)

[Sign In](#)

[No account? sign up now](#)

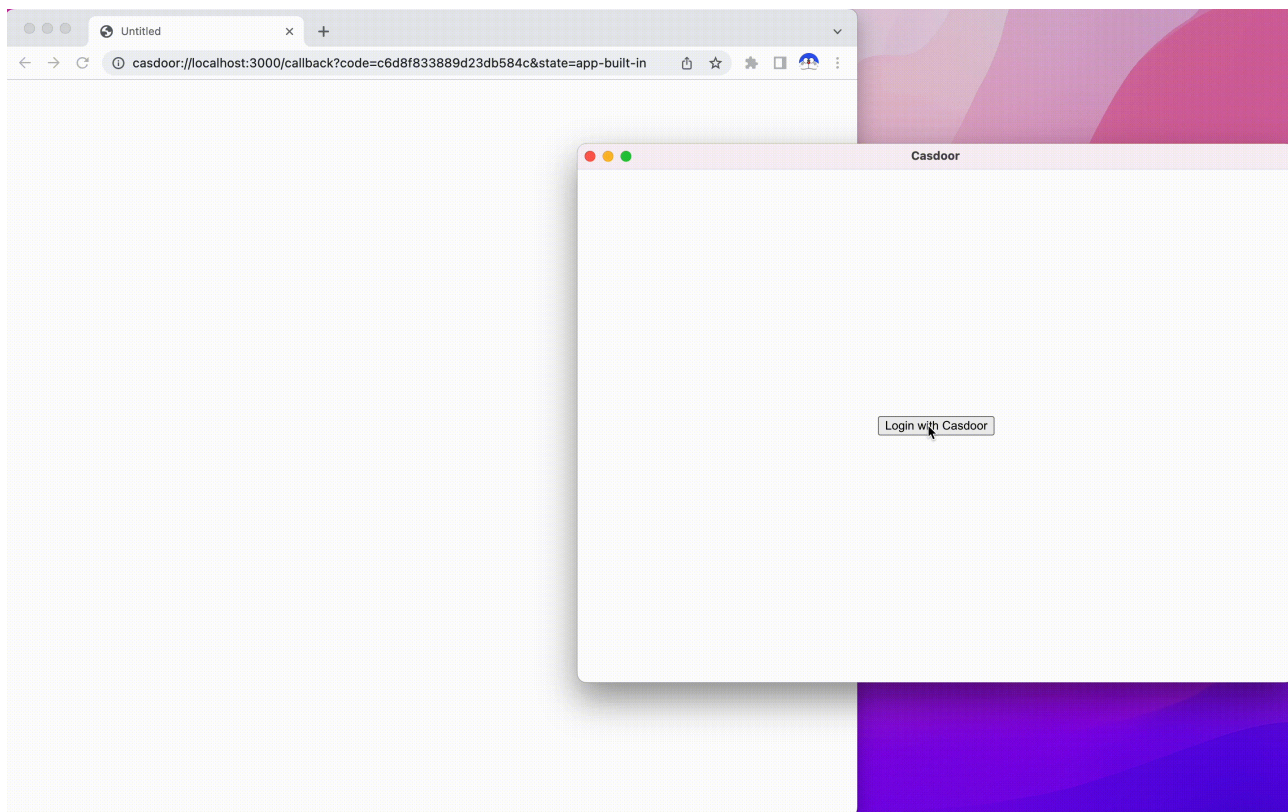


Made with ❤️ by Casdoor

登录成功后，您的Electron应用程序将会打开，并且您的用户名将会在您的应用程序上显示。



您可以在下面的gif图像中预览整个过程。



集成步骤

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);
}
```

这将允许浏览器打开你的电子应用程序并将登录信息发送到电子应用程序。

在浏览器中打开登录URL

```
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;
```

您可以更改前五个参数。

监听开启的应用程序事件

一旦您通过浏览器成功登录，浏览器将打开您的Electron应用程序。因此，你必须监听开放应用程序事件。

```
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browser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: `${getUserInfoUrl}?accessToken=${data.access_token}`,
  });
  return resp.data;
}
```

最后，你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

一个用于Casdoor的Dotnet桌面应用示例。

如何运行示例

先决条件

- [dotNET 6 SDK](#)
- [WebView2 运行时](#) (通常预装在 Windows 上)

初始化

初始化需要5个参数，所有这些参数都是字符串类型：

名称	描述	文件
Domain	您的Casdoor服务器的主机/域	<code>CasdoorVariables.cs</code>
Clientid	您的 Casdoor 应用程序的客户端 ID	<code>CasdoorVariables.cs</code>
AppName	您的Casdoor应用程序的名称	<code>CasdoorVariables.cs</code>
CallbackURL	您的Casdoor应用程序的回调URL路径。如果未提供，它将是 <code>casdoor://callback</code>	<code>CasdoorVariables.cs</code>

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

如果您不设置这些参数，项目将默认使用[Casdoor在线演示](#)作为Casdoor服务器，以及[Casnode](#)作为Casdoor应用程序。

运行中

Visual Studio

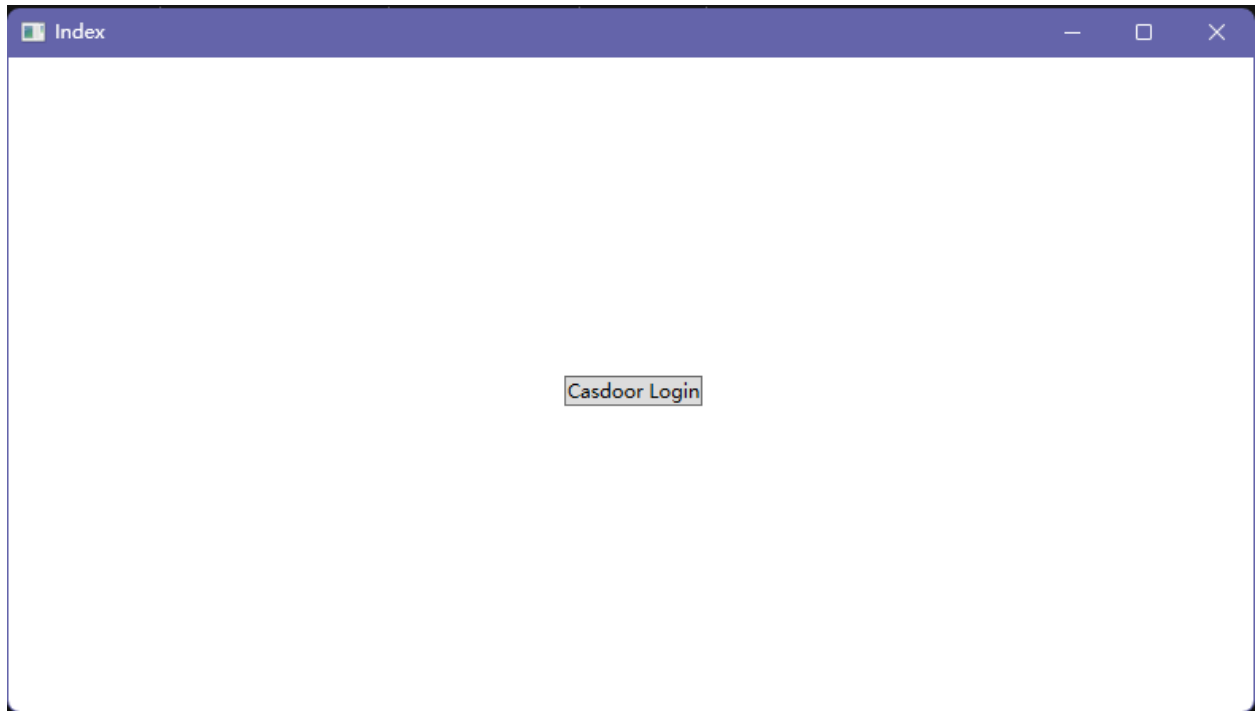
1. 打开 `casdoor-dotnet-desktop-example.sln`
2. 按 `Ctrl + F5` 开始

命令行

1. `cd src/DesktopApp`
2. `dotnet run`

预览

运行dotNET桌面应用程序后，您的桌面上将出现一个新窗口。



如果你点击 `Casdoor Login` 按钮，一个登录窗口将会出现在你的桌面上。



密码 WebAuthn

admin

...

下次自动登录

[忘记密码?](#)

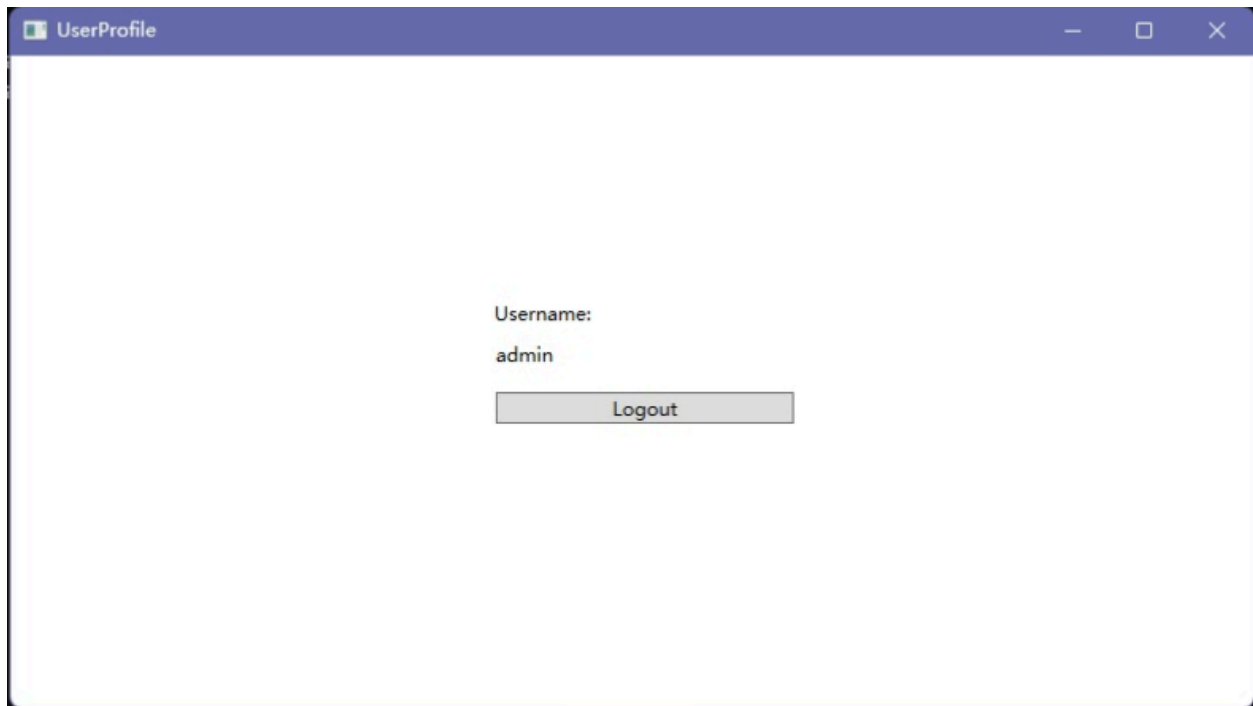
登录

[没有账号? 立即注册](#)

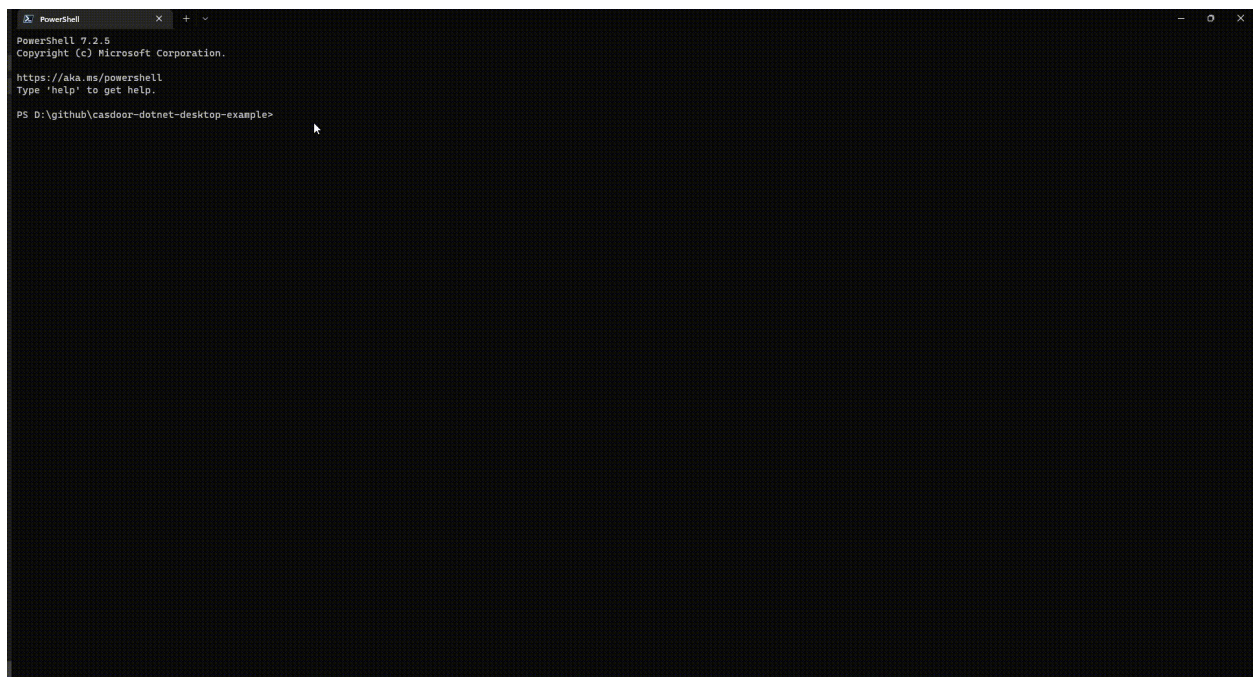


Feedback

成功登录后，用户个人资料窗口将出现在您的桌面上，显示您的用户名。



您可以在下面的GIF图像中预览整个过程。



如何集成

打开登录窗口

```
var login = new Login();  
// 当登录成功时触发, 您将在事件处理器中收到一个授权码  
login.CodeReceived += Login_CodeReceived;  
login.ShowDialog();
```

使用授权码获取用户信息

```
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);
```


移动SDKs .NET MAUI应用

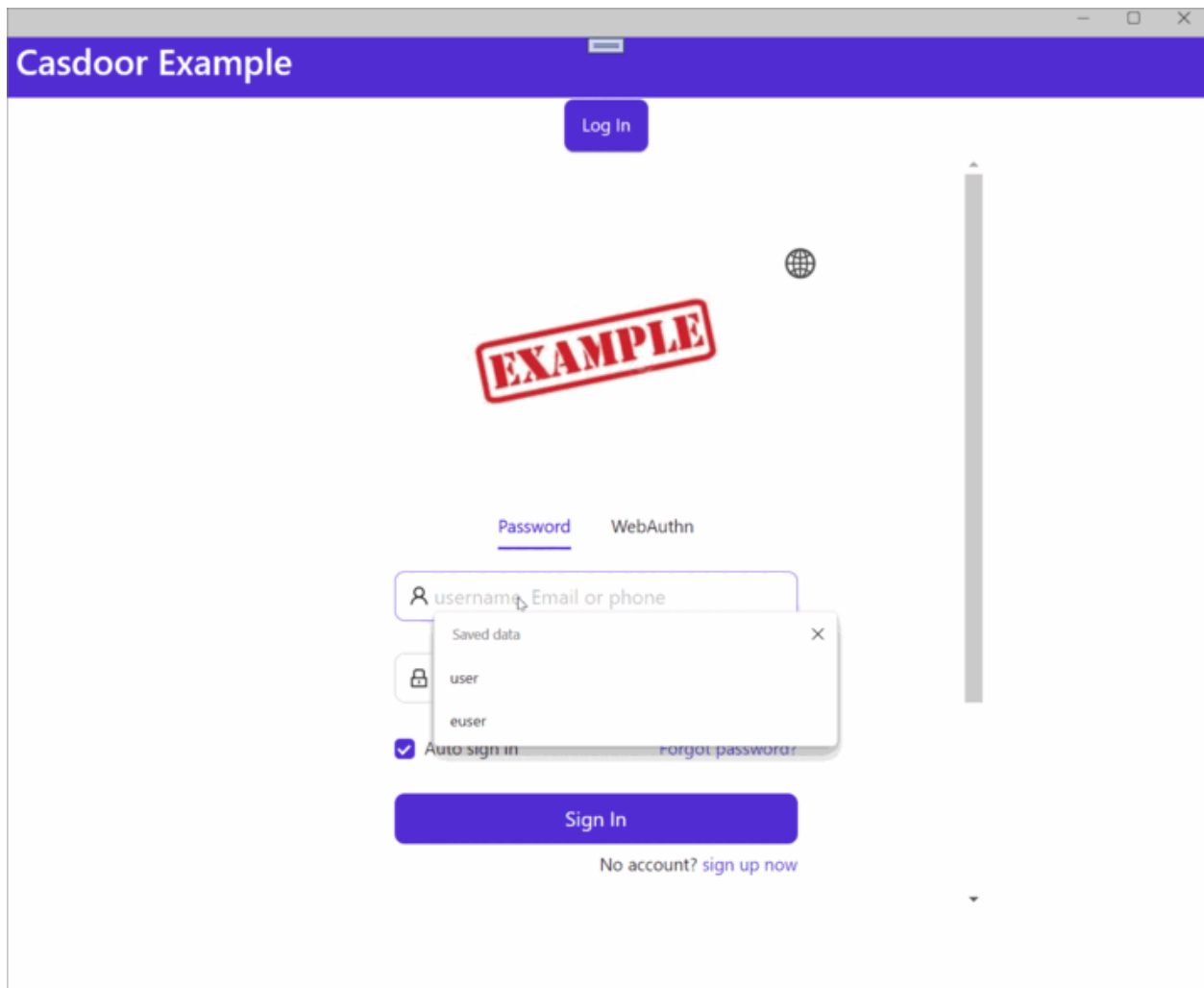
此仓库包含一个.NET MAUI应用和.NET MAUI库，用于演示Casdoor通过OpenID Connect进行身份验证。

演示

安卓

.NET

Windows



要求

- 在您的机器上安装 [.NET 7 SDK](#)
- 需要您的目标平台所需的资产，如 [这里](#) 所述
- Windows 17.3 的 Visual Studio 2022 或 Mac 17.4 的 Visual Studio 2022（可选）

入门

步骤1: 创建一个MAUI应用程序

创建您的[MAUI应用程序](#)。

步骤2: 添加引用

在您的项目中添加对 `Casdoor.MauioIdcClient` 的引用。

步骤3: 添加Casdoor客户端

在服务中将 `CasdoorClient` 添加为单例。

```
builder.Services.AddSingleton(new CasdoorClient(new()  
{  
    Domain = "<your domain>",  
    ClientId = "<your client>",  
    Scope = "openid profile email",  
  
#if WINDOWS  
    RedirectUri = "http://localhost/callback"  
#else  
    RedirectUri = "casdoor://callback"  
#endif  
}));
```

步骤4: 设计用户界面

在 `MainPage` 文件中添加代码。

MainPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="Casdoor.MauiOidcClient.Example.MainPage">

    <ScrollView>
        <VerticalStackLayout>

            <StackLayout
                x:Name="LoginView">
                <Button
                    x:Name="LoginBtn"
                    Text="Log In"
                    SemanticProperties.Hint="Click to log in"
                    Clicked="OnLoginClicked"
                    HorizontalOptions="Center" />

                <WebView x:Name="WebViewInstance" />
            </StackLayout>

            <StackLayout
                x:Name="HomeView"
                IsVisible="false">

                <Label
                    Text="Welcome to .NET Multi-platform App UI"
                    SemanticProperties.HeadingLevel="Level2"
                    SemanticProperties.Description="Welcome to dot net
Multi-platform App UI"
                    FontSize="18"
                    HorizontalOptions="Center" />

                <Button
                    x:Name="CounterBtn"
                    Text="Click me"

```

MainPage.cs

```
namespace Casdoor.MauiOidcClient.Example
{
    public partial class MainPage : ContentPage
    {
        int count = 0;
        private readonly CasdoorClient client;
        private string accessToken;
        public MainPage(CasdoorClient client)
        {
            InitializeComponent();
            this.client = client;

#if WINDOWS
            client.Browser = new
WebViewBrowserAuthenticator(WebViewInstance);
#endif
        }

        private void OnCounterClicked(object sender, EventArgs e)
        {
            count++;

            if (count == 1)
                CounterBtn.Text = $"Clicked {count} time";
            else
                CounterBtn.Text = $"Clicked {count} times";

            SemanticScreenReader.Announce(CounterBtn.Text);
        }

        private async void OnLoginClicked(object sender, EventArgs
e)
        {
            var loginResult = await client.LoginAsync();
            accessToken = loginResult.AccessToken;
        }
    }
}
```

步骤5：支持Android平台

修改 `AndroidManifest.xml` 文件。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/
android">
    <application android:allowBackup="true" android:icon="@mipmap/
appicon" android:roundIcon="@mipmap/appicon_round"
android:supportsRtl="true"></application>
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <queries>
        <intent>
            <action
android:name="android.support.customtabs.action.CustomTabsService"
/>
        </intent>
    </queries>
</manifest>
```

步骤6：启动应用程序

Visual Studio：按 `Ctrl + F5` 开始。

Qt 桌面应用程序

一个为 Casdoor 的 [Qt桌面应用程序示例](#)。

如何运行这个示例

前置要求

- [Qt6 SDK](#)
- [OpenSSL工具包](#)

初始化

你需要初始化7个字符串参数：

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

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

如果你不设置 `endpoint` 参数，这个项目将使用 <http://localhost:8000> 作为默认的Casdoor服务器。

运行应用程序

使用Qt Creator

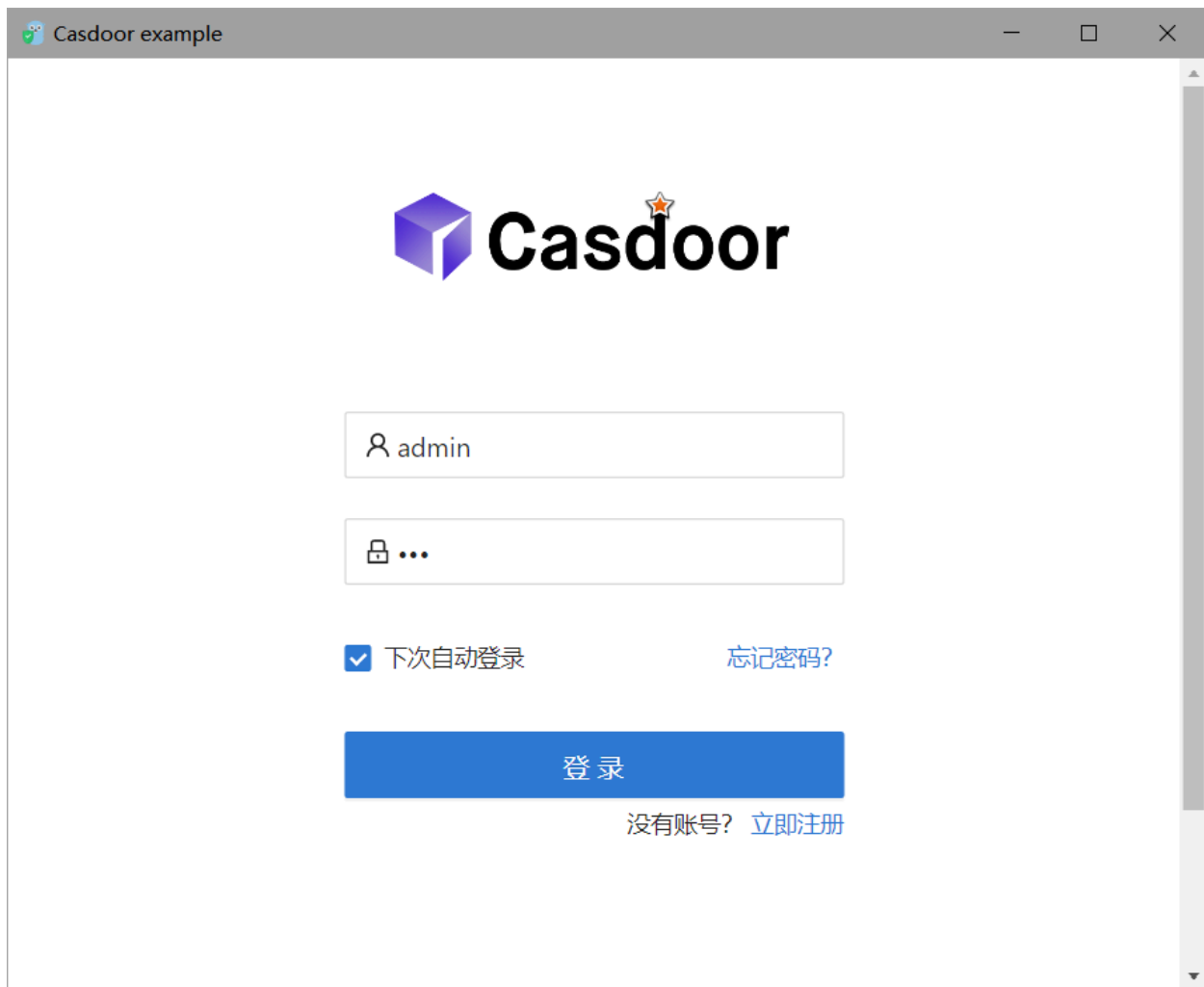
1. 打开 `casdoor-cpp-qt-example.pro`
2. 在 `casdoor-cpp-qt-example.pro` 中设置OpenSSL的 `INCLUDEPATH`
3. 按 `Ctrl + R` 开始

效果预览

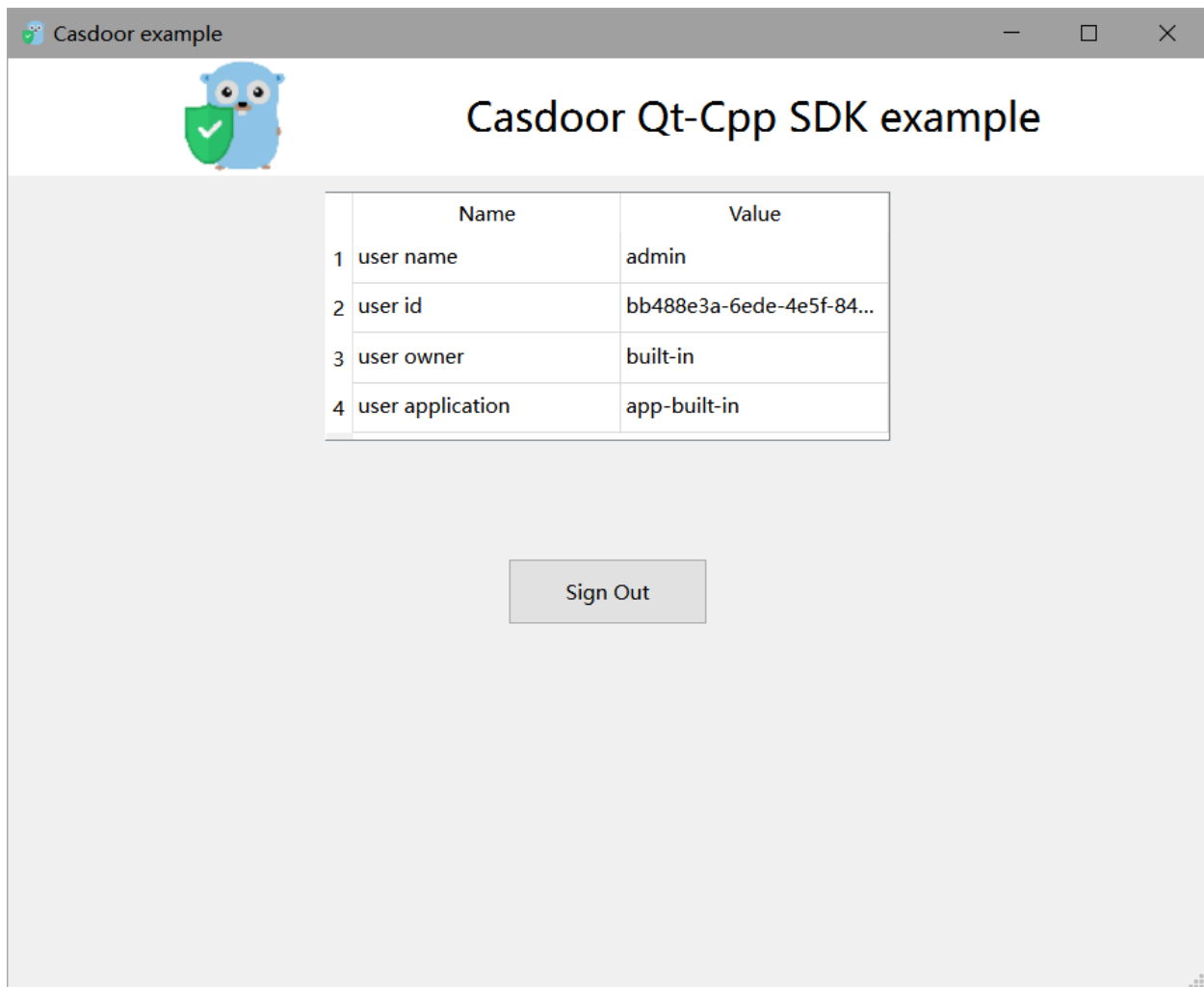
运行这个Qt桌面应用后，你的桌面上会显示一个新窗口。



如果你点击 `Sign In` 按钮，你的桌面上会显示一个登录窗口。



登录成功后，你的桌面上会显示一个用户资料窗口，显示你的用户信息。



你可以在下面的GIF图像中预览整个过程。



如何集成

打开登录窗口

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

监听打开应用事件

```
// 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()));
```

使用授权码获取用户信息

```
// 获取令牌并使用JWT库解析
std::string token = m_casdoor->GetOAuthToken(code.toStdString());
auto decoded = m_casdoor->ParseJwtToken(token);
```

移动 SDKs

React Native应用

Casdoor的React Native移动应用示例

React Native应用

这里有一个[Casdoor React Native移动应用示例](#)，可以帮助你快速了解如何在React Native中使用Casdoor。

如何运行示例

快速开始

- 下载代码

```
git clone git@github.com:casdoor/casdoor-react-native-example.git
```

- 安装依赖

```
cd casdoor-react-native-example  
yarn install  
cd ios/  
pod install
```

- 在ios上运行

```
cd casdoor-react-native-example  
react-native start  
react-native run-ios
```

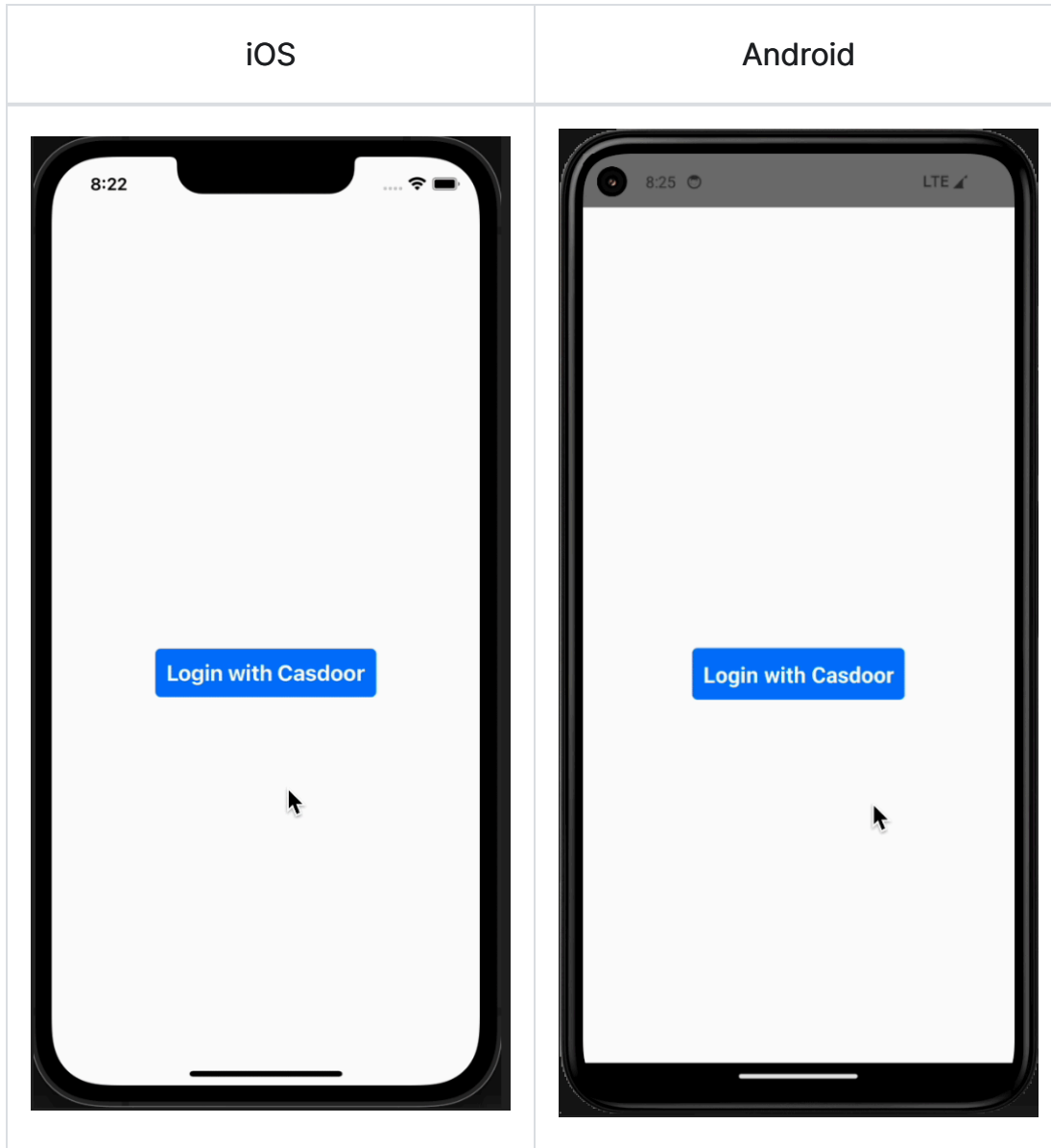
- 在android上运行


```
cd casdoor-react-native-example
react-native start
react-native run-android
```

确保在运行之前打开模拟器或真实设备。

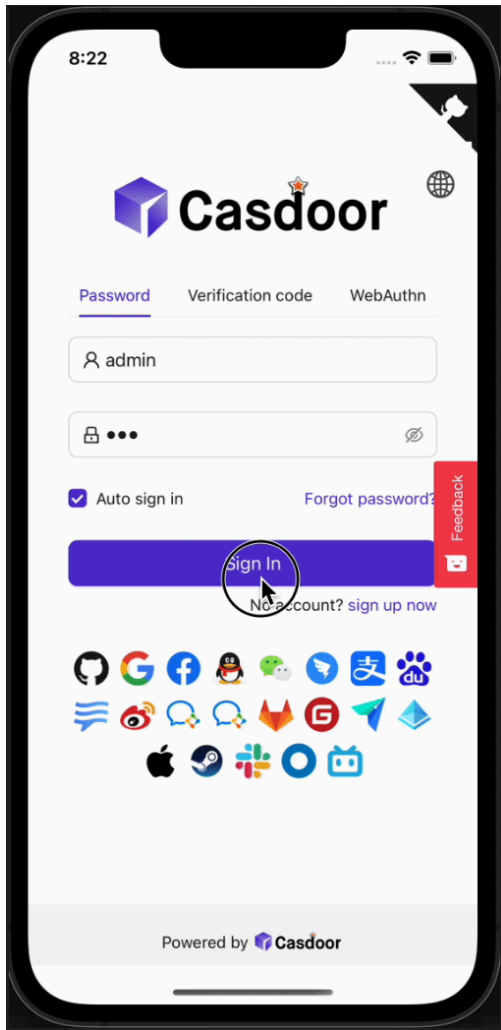
预览

运行这个react-native-example移动应用后，以下窗口将在模拟器或真实设备上显示。

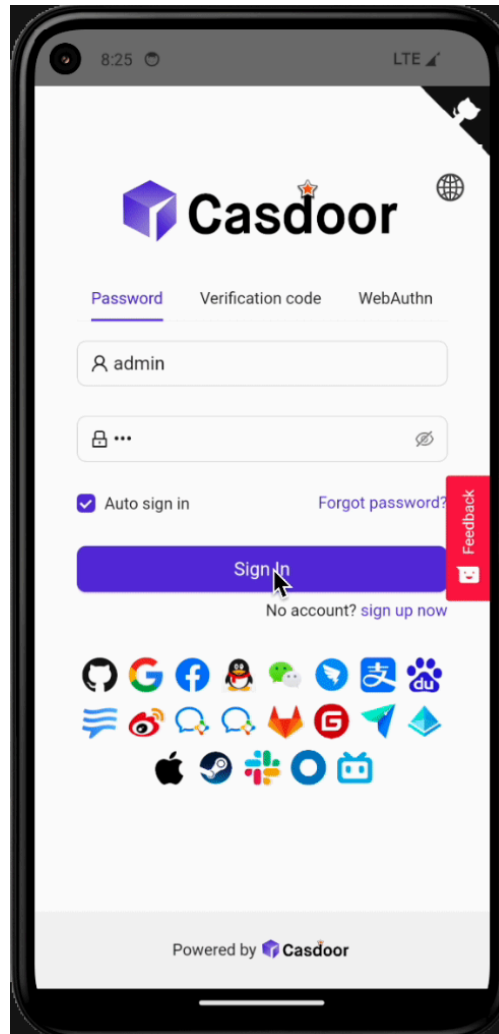


如果你点击 [使用Casdoor登录](#) 按钮，Casdoor登录窗口将出现在屏幕上。

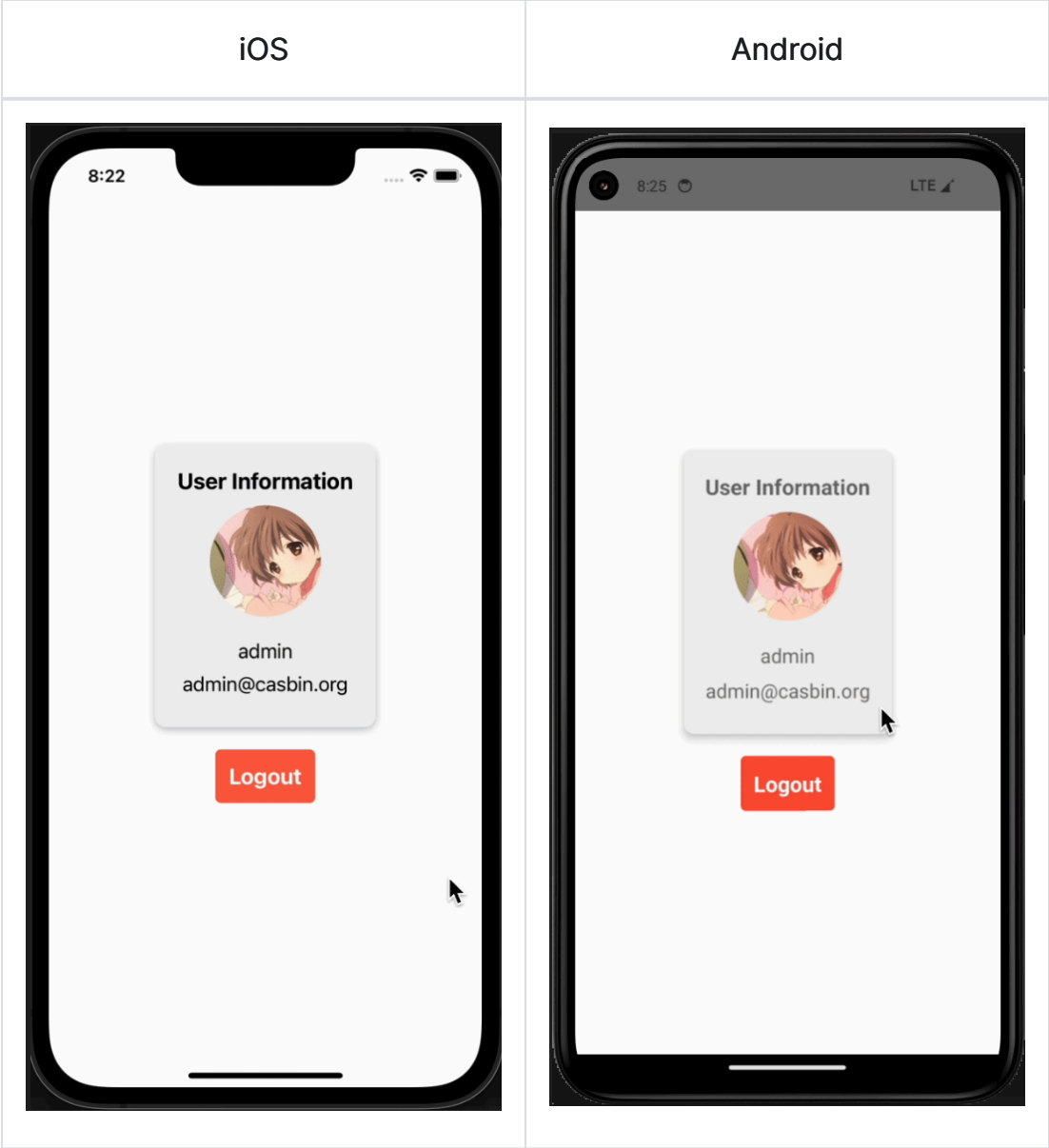
iOS



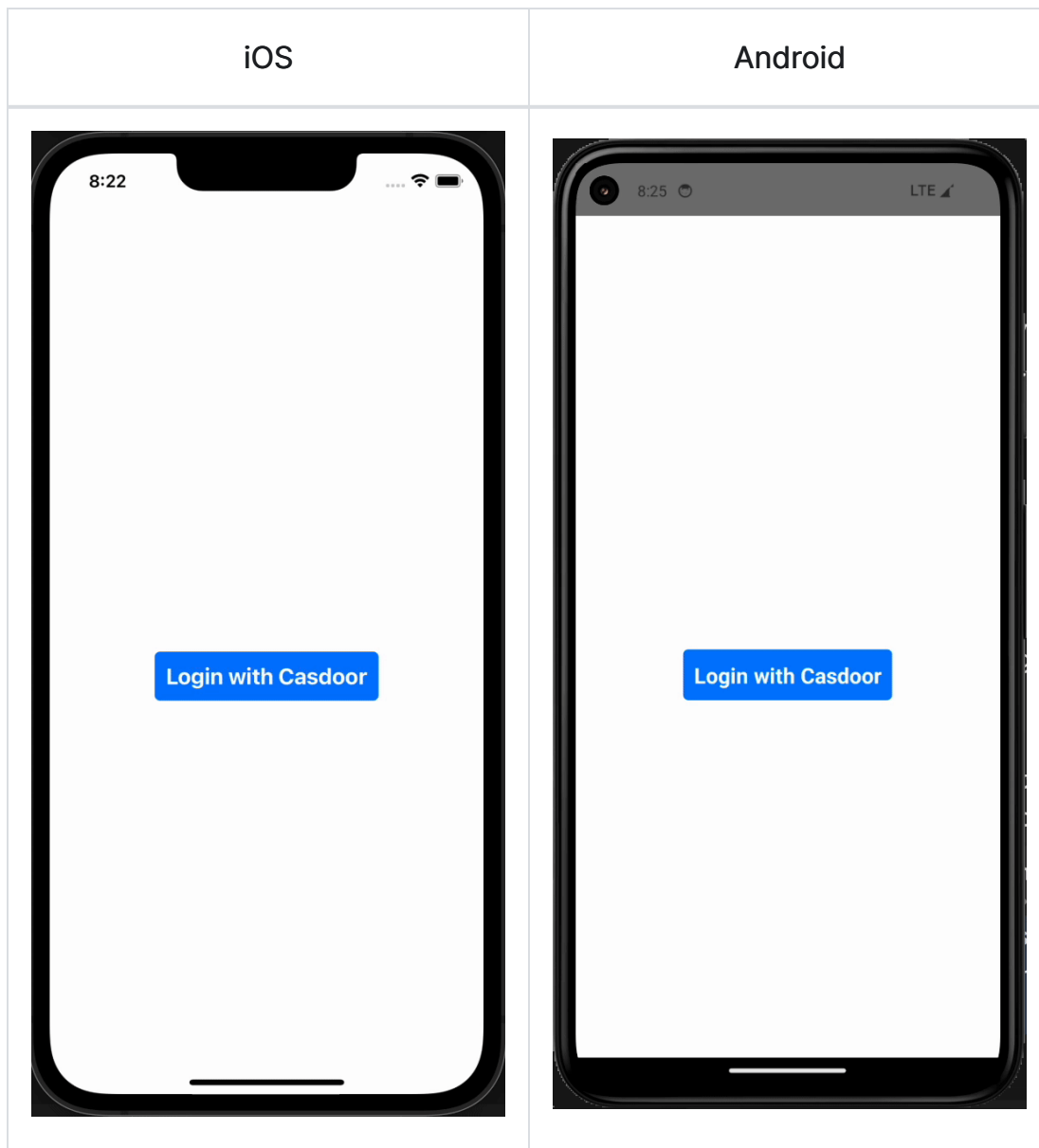
Android



登录成功后，一个用户资料窗口将出现在你的屏幕上，显示你的用户信息。



你可以在下面的GIF图像中预览整个过程。



如何集成

上述示例使用了[casdoor-react-native-sdk](#)，你也可以在你自己的项目中集成这个sdk。

集成和使用sdk非常简单，下面的步骤将向你展示如何集成和使用：

步骤1: 导入SDK

```
# NPM
npm i casdoor-react-native-sdk

# Yarn
yarn add casdoor-react-native-sdk
```

步骤2: 初始化SDK

初始化需要7个参数，都是字符串类型：

名称（按顺序）	必须	描述
serverUrl	是	你的Casdoor服务器URL
clientId	是	你的Casdoor应用的客户端ID
appName	是	你的Casdoor应用的名称
organizationName	是	与你的Casdoor应用关联的Casdoor组织的名称
redirectPath	否	你的Casdoor应用的重定向URL的路径，如果未提供，将是 <code>/callback</code>
signinPath	否	你的Casdoor应用的登录URL的路径

```
import SDK from 'casdoor-react-native-sdk'

const sdkConfig = {
  serverUrl: 'https://door.casdoor.com',
  clientId: 'b800a86702dd4d29ec4d',
  appName: 'app-example',
  organizationName: 'casbin',
  redirectPath: 'http://localhost:5000/callback',
  signinPath: '/api/signin',
};
const sdk = new SDK(sdkConfig)
```

步骤3：使用SDK

在适当的地方使用sdk的相应API接口。

最简单的casdoor授权和认证过程可以通过使用以下三个API实现：

```
// 获取登录url
getSigninUrl()

// 获取访问令牌
getAccessToken(redirectUrl); // http://localhost:5000/
callback?code=b75bc5c5ac65ffa516e5&state=gjmfdgqf498

// 解码jwt令牌以获取用户信息
JwtDecode(jwtToken)
```

如果你想使用其他接口，请查看[casdoor-react-native-sdk](#)以获取更多帮助。

Casdoor插件

Casdoor还为一些非常流行的平台提供插件或中间件，例如Java的Spring Boot，PHP的WordPress，Python的Odoo等等。

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

要获取官方Casdoor插件的完整列表，请访问[Casdoor仓库](#)。

Next.js

`nextjs-auth`是一个如何在next-js项目中集成casdoor的示例。我们将引导您完成以下步骤。

步骤1：部署Casdoor

首先，应部署Casdoor。

您可以参考Casdoor官方文档中的[服务器安装](#)。请在**生产模式**下部署您的Casdoor实例。

成功部署后，请确保以下内容：

- 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。
- 通过输入 `admin` 作为用户名和 `123` 作为密码来测试登录功能。

之后，您可以使用以下步骤在您自己的应用中快速实现基于Casdoor的登录页面。

步骤2：添加中间件

中间件允许您在请求完成之前运行代码。然后，根据传入的请求，您可以通过重写、重定向、修改请求或响应头，或直接响应来修改响应。

使用项目根目录中的 `middleware.ts`（或 `.js`）文件来定义中间件。例如，与 `pages` 或 `app` 同级，或者在 `src` 内（如果适用）。

示例

```
//定义中间件将在哪些路径上运行
const protectedRoutes = ["/profile"];

export default function middleware(req) {
  if (protectedRoutes.includes(req.nextUrl.pathname)) {
    //将传入的请求重定向到不同的URL
    return NextResponse.redirect(new URL("/login", req.url));
  }
}
```

查看next.js官方文档[middleware](#)以获取更多详细信息。

步骤3：使用Casdoor SDK

1.安装SDK

首先，通过NPM或Yarn安装 `casdoor-js-sdk`：

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

或者：

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

2.初始化SDK

然后按照以下顺序初始化6个字符串类型的参数：

名称	必需	描述
serverUrl	是	Casdoor服务器URL, 例如 <code>http://localhost:8000</code>
clientId	是	应用客户端ID
clientSecret	是	应用客户端密钥
organizationName	是	应用组织
appName	是	应用名称
redirectPath	是	重定向URL

示例

```
const sdkConfig = {  
  serverUrl: "https://door.casdoor.com",  
  clientId: "294b09fbc17f95daf2fe",  
  clientSecret: "dd8982f7046ccba1bbd7851d5c1ece4e52bf039d",  
  organizationName: "casbin",  
  appName: "app-vue-python-example",  
  redirectPath: "/callback",  
};
```

注意事项

用您自己的Casdoor实例替换配置值, 特别是 `clientId`, `clientSecret` 和

serverUrl。

3.重定向到登录页面

当您需要对访问您的应用的用户进行身份验证时，您可以发送目标URL并重定向到Casdoor提供的登录页面。

请确保您已经在应用配置中提前添加了回调URL（例如**http://localhost:8080/callback**）。

```
const CasdoorSDK = new Sdk(sdkConfig);
CasdoorSDK.signin_redirect();
```

4.获取令牌和存储

在通过Casdoor验证后，它将带着令牌重定向回您的应用程序。

您可以选择使用cookie来存储令牌。

```
CasdoorSDK.exchangeForAccessToken()
  .then((res) => {
    if (res && res.access_token) {
      //Get Token
      return CasdoorSDK.getUserInfo(res.access_token);
    }
  })
  .then((res) => {
    // Storage Token
    Cookies.set("casdoorUser", JSON.stringify(res));
  });
```

您可以参考Casdoor官方文档了解[如何使用Casdoor SDK](#)。

步骤4：添加中间件认证功能

当用户试图访问受保护的路由时，中间件认证功能会验证他们的身份。如果用户未经认证，他们将被重定向到登录页面或被拒绝访问。

示例

```
//受保护的路由
const protectedRoutes = ["/profile"];
const casdoorUserCookie = req.cookies.get("casdoorUser");
const isAuthenticated = casdoorUserCookie ? true : false;

//认证功能
if (!isAuthenticated &&
protectedRoutes.includes(req.nextUrl.pathname)) {
  return NextResponse.redirect(new URL("/login", req.url));
}
```

Nuxt

`nuxt-auth`是一个如何在nuxt项目中集成casdoor的示例。我们将引导您完成以下步骤。许多步骤与nextjs-auth相似。

步骤1：部署Casdoor

首先，应部署Casdoor。

您可以参考Casdoor官方文档中的[服务器安装](#)。请以**生产模式**部署您的Casdoor实例。

成功部署后，请确保以下内容：

- 打开您喜欢的浏览器并访问**`http://localhost:8000`

页面。

- 通过输入 `admin` 作为用户名和 `123` 作为密码来测试登录功能。

之后，您可以使用以下步骤在您自己的应用中快速实现基于Casdoor的登录页面。

步骤2：添加中间件

中间件允许您在请求完成之前运行代码。然后，根据传入的请求，您可以通过重写、重定向、修改请求或响应头，或直接响应来修改响应。

在项目根目录的 `middleware` 目录中创建 `.js` 或 `.ts` 文件来定义中间件。并且文件名被识别为中间件的名称。例如，在 `nuxt-auth` 中，我们在 `middleware` 目录中创建了一个名为 `myMiddleware.js` 的文件，可以在 `nuxt.config.js` 等其他地方引用为 `myMiddleware`。

示例

```
//定义中间件将在哪些路径上运行
const protectedRoutes = ["/profile"];

export default function ({route, redirect}) {

  if (protectedRoutes.includes(route.path)) {
    //将传入的请求重定向到不同的URL
    redirect('/login');
  }
}
```

要使中间件工作，您应该在 `nuxt.config.js` 中添加路由，如下所示：

```
export default {
  // 其他配置

  // 需要添加的内容
  router: {
    middleware: ['myMiddleware'] // 替换为你的中间件名称
  },
}
```

查看 [nuxt官方文档middleware](#) 以获取更多详细信息。

步骤3：使用Casdoor SDK

1.安装SDK

首先，通过NPM或Yarn安装 `casdoor-js-sdk`：

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

或者：

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

2.初始化SDK

然后按以下顺序初始化6个字符串类型的参数：

名称	必需	描述
serverUrl	是	Casdoor服务器URL，例如 <code>http://localhost:8000</code>
clientId	是	应用客户端ID
clientSecret	是	应用客户端密钥
organizationName	是	应用组织

名称	必需	描述
appName	是	应用名称
redirectPath	是	重定向URL

示例

```
const sdkConfig = {  
  serverUrl: "https://door.casdoor.com",  
  clientId: "294b09fbc17f95daf2fe",  
  clientSecret: "dd8982f7046ccba1bbd7851d5c1ece4e52bf039d",  
  organizationName: "casbin",  
  appName: "app-vue-python-example",  
  redirectPath: "/callback",  
};
```

⚠ 注意事项

将配置值替换为您自己的Casdoor实例，特别是 `clientId`、`clientSecret` 和 `serverUrl`。

3.重定向到登录页面

当您需要验证访问您的应用的用户时，您可以发送目标URL并重定向到Casdoor提供的登录页面。

确保您已经在应用配置中提前添加了回调URL（例如**`http://localhost:8080/callback`**）。

```
const CasdoorSDK = new Sdk(sdkConfig);
CasdoorSDK.signin_redirect();
```

4. 获取令牌和存储

Casdoor验证通过后，它将带着令牌重定向回您的应用。

您可以选择使用cookie来存储令牌。

```
CasdoorSDK.exchangeForAccessToken()
  .then((res) => {
    if (res && res.access_token) {
      // 获取令牌
      return CasdoorSDK.getUserInfo(res.access_token);
    }
  })
  .then((res) => {
    // 存储令牌
    Cookies.set("casdoorUser", JSON.stringify(res));
  });
```

您可以参考Casdoor官方文档中的[如何使用Casdoor SDK](#)。

步骤4：添加中间件认证功能

当用户试图访问受保护的路由时，中间件认证功能会验证他们的身份。如果用户未经认证，他们将被重定向到登录页面或被拒绝访问。

示例

```
import Cookies from "js-cookie";
```


OAuth 2.0

介绍

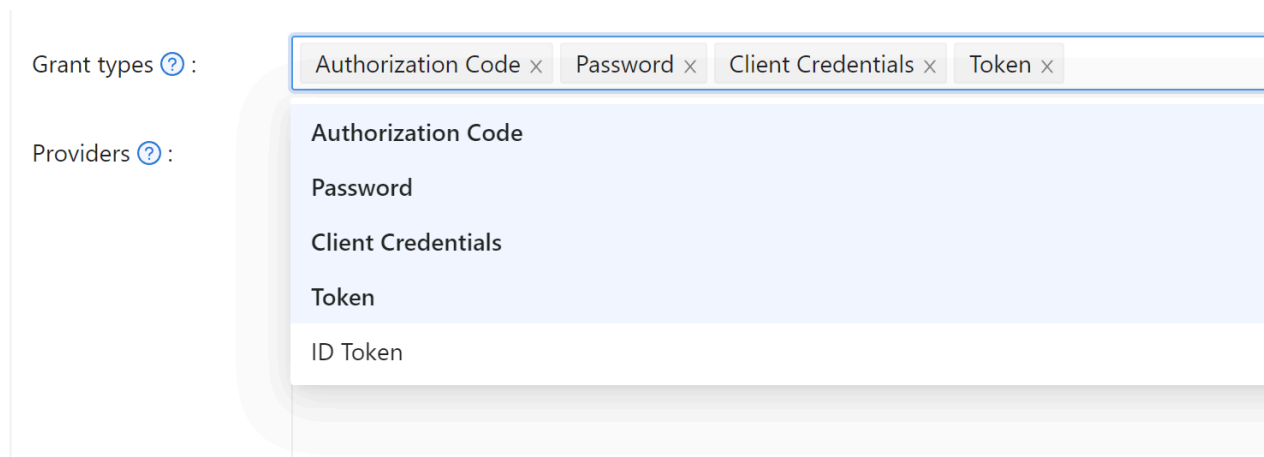
Casdoor 支持使用 AccessToken 验证客户端。在本节中，我们将向您展示如何获取 AccessToken，如何验证 AccessToken，以及如何使用 AccessToken。

如何获取AccessToken

获取访问令牌有两种方式：您可以使用 [Casdoor SDK](#) 详情请参阅SDK文档。这里我们将主要向您展示如何使用 API 来获取访问令牌。

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，重定向您的用户到：

```
https://<CASD00R_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

❗ 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://<CASD00R_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://<CASD00R_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。当获取验证码时，您可以添加两个参数来启用PKCE：

```
&code_challenge_method=S256&code_challenge=YOUR_CHANNELLENGE
```

获取令牌时，您需要传递 `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，但如果您要传递这个参数，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://<CASD00R_HOST>/login/oauth/
authorize?client_id=CLIENT_ID&redirect_uri=REDIRECT_URI&response_type=token&scope=openid&state=STATE
```

在您的用户通过Casdoor进行身份验证后，Casdoor将会将他们重定向到：

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

Casdoor也支持`id_token`作为 `response_type`，这是OpenID的一个特性。

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，您需要确保已启用密码凭据授权，并向以下地址发送POST请求：

```
https://<CASD00R_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,你需要确保你已启用客户端凭证授权,并发送POST请求到 `https://<CASD00R_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Access token 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。

刷新令牌

也许你想更新你的访问令牌,那么你可以使用上面获得的 `refreshToken`。

首先,您需要在应用中设置刷新令牌的过期时间(默认为0小时),并向 `https://<CASD00R_HOST>/api/login/oauth/refresh_token` 发送POST请求

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

你会得到这样的回应：

```
{
  "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

Casdoor 目前支持 [令牌自省](#) 端点。此端点受基本身份验证保护 (ClientId:ClientSecret) 。

```
POST /api/login/oauth/introspect HTTP/1.1
Host: CASD00R_HOST
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Authorization: Basic Y2xpZW50X2lkOmNsawWudF9zZWNyZXQ=

token=ACCESS_TOKEN&token_type_hint=access_token
```

您将收到以下响应：

```
{
  "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 `AccessToken`

您可以使用 `AccessToken` 访问需要认证的 Casdoor API。

例如，有两种不同的方式来请求 `/api/userinfo`。

类型1: 查询参数


```
https://<CASD00R_HOST>/api/userinfo?accessToken=<your_access_token>
```

类型2: HTTP Bearer 令牌

```
https://<CASD00R_HOST>/api/userinfo with the header: "Authorization: Bearer <your_access_token>"
```

Casdoor将解析access_token, 并根据scope返回相应的用户信息。你将收到相同的响应, 看起来像这样:

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

如果您期望获得更多用户信息, 请在获取AccessToken的步骤[授权码授予](#)中添加scope。

userinfo 和 get-account APIs之间的差异

- `/api/userinfo`: 此API作为OIDC协议的一部分返回用户信息。它提供有限的信息, 包括仅在OIDC标准中定义的基本信息。要查看Casdoor支持的可用范围列表, 请参阅[可用范围](#)部分。
- `/api/get-account`: 此API用于检索当前登录账户的用户对象。这是一个特定于Casdoor的API, 允许您获取Casdoor中[用户](#)的所有信息。

使用Casdoor作为CAS服务器

使用Casdoor作为CAS服务器

Casdoor现在可以作为CAS服务器使用。它目前支持CAS 3.0。

简介

Casdoor中的CAS端点前缀是 `<Casdoor endpoint>/cas/<organization name>/<application name>`。这是一个使用端点 `https://door.casdoor.com`，在 `casbin` 组织下名为 `cas-java-app` 的应用程序的示例：

- `/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`

有关CAS，其不同版本以及这些端点的参数的更多信息，请参阅[CAS协议规范](#)。

一个例子

这是一个官方示例 [GitHub 仓库](#)，其中包含一个网页应用，并使用官方的 CAS Java 客户端 [GitHub 仓库](#)。通过这个例子，你可以学习如何通过CAS连接到Casdoor。

📌 备注

注意：目前，Casdoor只支持CAS的所有三个版本：CAS 1.0，2.0和3.0。

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

默认情况下，此应用程序使用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>
```

如果您想要使用CAS 2.0保护此网页应用程序，请将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>
```

对于 CAS 1.0，请使用以下内容：

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

将所有 `casServerUrlPrefix` 参数的实例更改为：

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

将所有的 `casServerLoginUrl` 参数实例更改为：

```
<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配置，请查看[Java CAS客户端GitHub仓库](#)以获取详细信息。



如何连接到Casdoor



SAML

SAML

概述

使用Casdoor作为SAML IdP

AWS Client VPN

使用Casdoor作为SAML IdP

Keycloak

使用Casdoor作为SAML IdP

Google Workspace

使用Casdoor作为SAML IdP

Appgate (POST)

如何使用 Casdoor 作为SAML IdP for Appgate

腾讯云

使用Casdoor作为SAML IdP

概述

现在可以使用Casdoor作为SAML IdP。到目前为止，Casdoor已经支持了SAML 2.0的主要功能。

在SP中的配置

一般来说，SP需要三个必填字段：`Single Sign-On`，`Issuer`和`Public Certificate`。大多数SP可以通过上传XML元数据文件或XML元数据URL来自动完成这些字段。

Casdoor中的SAML端点的元数据是`<Endpoint of casdoor>/api/saml/metadata?application=admin/<application name>`。假设Casdoor的端点是`https://door.casdoor.com`，并且它包含一个名为`app-built-in`的应用程序。XML元数据端点将是：

```
https://door.casdoor.com/api/saml/metadata?application=admin/app-built-in
```

您也可以在应用程序编辑页面中找到元数据。 点击按钮复制URL并粘贴到浏览器中下载XML元数据。

```
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="https://door...
  <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+TCCAuGgAwIBAgIDAeJAMA0GCsgGSIB3DQEBcwUAMDYxHTAbBgNVBaoTFENhc2Rvb3I3JnYw5pemF0aW9uMRUwEwYDVQDEwzDXYXNkbb
        </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>
  </EntityDescriptor>
  <SingleSignOnService Bindings="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location="https://door.casdoor.com/login/saml/authorize/admin/app-built-in"></SingleSignOnServ
```

📄 Copy SAML metadata URL

在Casdoor IdP中的配置

Casdoor支持GET和POST `SAMLResponse`。当Casdoor将 `SAMLResponse` 发送到SP时，Casdoor需要知道SP支持哪些类型的请求。您需要根据您的SP支持的 `SAMLResponse` 类型在Casdoor中配置应用程序。

! 信息

如果您填写了 `Reply URL`，Casdoor将通过POST请求发送 `SAMLResponse`。如果回复URL为空，Casdoor将使用GET请求。您可能会想知道，如果 `Reply URL` 为空，Casdoor如何知道SP的 `Reply URL`。实际上，Casdoor可以通过解析 `SAMLRequest` 获取名为 `AssertionConsumerServiceURL` 的URL，并将带有 `SAMLResponse` 的请求发送到 `AssertionConsumerServiceURL`。 `Reply URL` 将覆盖 `SAMLRequest` 中的 `AssertionConsumerServiceURL`。

- **回复URL**：输入验证SAML响应的ACS的URL。

Grant types [?](#) :

Authorization Code × Password ×

SAML Reply URL

[?](#) :

<https://mycontroller.mycompany.com/admin/saml>

Enable SAML



compress [?](#) :

- **重定向URL**：输入一个唯一的名称。在您的SP中，这可能被称为 `Audience` 或 `Entity ID`。确保您在此处填写的 `Redirect URL` 与您的SP中的一致。

Redirect URLs ? :

Redirect URL
↗ appgate
↗ https://git.casbin.com/user/oauth2/casdoor/callback
↗ http://localhost:3000/callback

用户资料

成功登录后，Casdoor返回的 `SAMLResponse` 中的用户资料有三个字段。XML中的属性和Casdoor中的用户属性的映射如下：

XML属性名称	用户字段
电子邮件	电子邮件
显示名称	显示名称
名称	名称

有关SAML及其不同版本的更多信息，请参见https://en.wikipedia.org/wiki/SAML_2.0。

一个例子

`gosaml2`是基于`etree`和`goxmldsig`的服务提供商的SAML 2.0实现，这是XML数字签名的纯Go实现。我们使用这个库来测试Casdoor中的SAML 2.0，如下所示。

假设您可以通过 `http://localhost:7001/` 访问Casdoor，并且您的Casdoor包含一个

名为 `app-built-in` 的应用程序，该应用程序属于一个名为 `built-in` 的组织。这两个 URL: `http://localhost:6900/acs/example` 和 `http://localhost:6900/saml/acs/example`，应该添加到 `app-built-in` 中的重定向URLs。

```
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)  
    if err != nil {  
        panic(err)  
    }  
}
```

运行上述代码，控制台将显示以下消息。

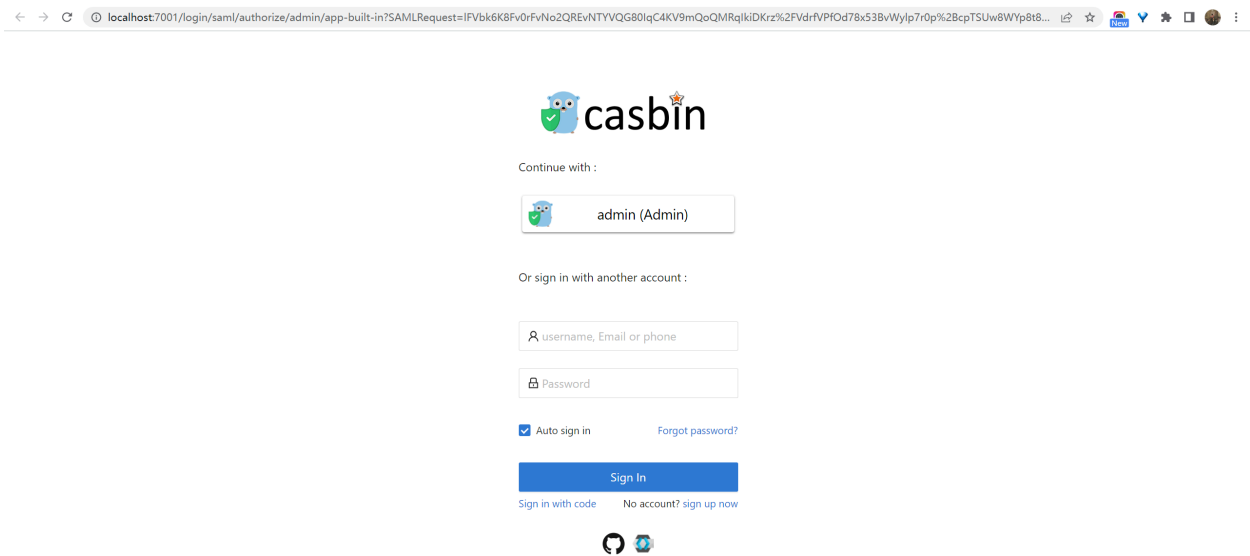
Visit this URL To Authenticate:

`http://localhost:7001/login/saml/authorize/admin/app-built-in?SAMLRequest=1FVbk6K8Fv0rFvNo2QR...`

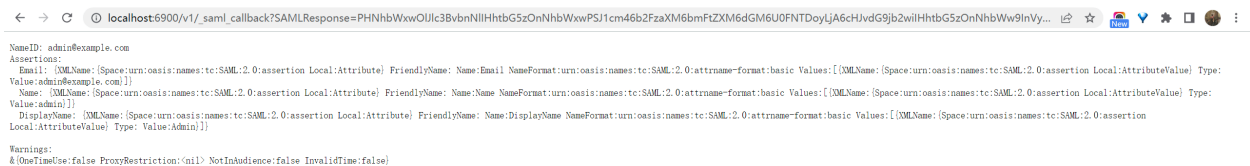
Supply:

SP ACS URL : `http://localhost:6900/v1/_saml_callback`

点击URL进行身份验证，Casdoor的登录页面将被显示。



身份验证后，您将收到如下所示的响应消息。



AWS Client VPN

Casdoor作为AWS Client VPN中的SAML IdP

本指南将向您展示如何配置Casdoor和AWS Client VPN，以在AWS Client VPN中添加Casdoor作为SAML IdP。

先决条件

要完成此设置，您将需要：

- 具有访问服务提供商配置设置的管理权限的AWS账户。
- 带有EC2实例的Amazon VPC
 - [设置VPC](#)
 - [启动EC2实例](#)
 - 在实例安全组中，允许来自VPC CIDR范围的ICMP流量 - 这是测试所需的。
- 将私有证书导入到[AWS证书管理器\(ACM\)](#)
 - [生成并导入证书到ACM](#)
- 运行最新AWS Client VPN软件的Windows或Mac系统。
 - [下载软件](#)

配置SAML应用程序

- 在Casdoor应用程序中，将Redirect URL 设置为 `urn:amazon:webservices:clientvpn`。

Tags [?](#):

Client ID [?](#):

Client secret [?](#):

Cert [?](#):

Redirect URLs [?](#)
:
Redirect URLs [Add](#)
Redirect URL
[🔗](#) `urn:amazon:webservices:clientvpn`

Token format [?](#):

Token expire [?](#): Hours

Refresh token expire [?](#): Hours

Enable password [?](#):

- 将SAML回复URL 设置为 `http://127.0.0.1:35001`。


Signup HTML :

Signin HTML :

Grant types :

SAML reply URL :

Enable SAML compression :

SAML metadata :

```
<EntityDescriptor xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns="urn:
<IDPSSODescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata" protocolSu
<KeyDescriptor use="signing">
<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
<X509Data xmlns="http://www.w3.org/2000/09/xmldsig#">
```

- 将 SAML 元数据 的内容保存为 XML 文件。

```
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:md:
<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+TCCAuGgAwIBAgIDAeJAMA0GCSqGSIb3DQEBCwUAMDYxHTAbBgNVBAoTFENhc2Rvc2JlIHR5cGU9b2t1LmNpdjEzIjEzJnYwI
</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="https://test.v2t1.com/login/saml/authorize/admin/app-
</SingleSignOnService>
</IDPSSODescriptor>
</EntityDescriptor>
```

 Copy SAML metadata URL

配置AWS

将Casdoor配置为AWS身份提供商

1. 打开IAM控制台，从导航栏中选择身份提供商。
2. 点击创建提供商。

3. 为提供商类型指定SAML，为此提供商添加一个唯一的名称，并上传元数据文档 - 与您在上一节中从Casdoor应用程序保存的文件相同。

4. 点击下一步。在下一个屏幕上，点击**创建**。

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

- User groups
- Users
- Roles
- Policies
- Identity providers**
- Account settings

Access reports

- Access analyzer
- Archive rules
- Analyzers
- Settings

IAM > Identity providers

Have you considered using AWS IAM Identity Center? AWS IAM Identity Center makes it easy to centrally manage access to multiple AWS accounts and provide users with single sign-on access to all their assigned accounts from one place. With IAM Identity Center, you can create and manage user identities in IAM Identity Center or easily connect to your existing SAML 2.0 compatible identity provider. Learn more

Identity providers (1) Info

Use an identity provider (IdP) to manage your user identities outside of AWS, but grant the user identities permissions to use AWS resources in your account.

Delete Add provider

Filter by Type

Search All Types

Provider	Type	Creation time
casdoor	SAML	Yesterday

IAM > Identity providers > Create Identity Provider

Add an Identity provider Info

Configure provider

Provider type Info

SAML
Establish trust between your AWS account and a SAML 2.0 compatible Identity Provider such as Shibboleth or Active Directory Federation Services.

OpenID Connect
Establish trust between your AWS account and Identity Provider services, such as Google or Salesforce.

Provider name

Enter a meaningful name to identify this provider

casdoor

Maximum 128 characters. Use alphanumeric or '._-' characters.

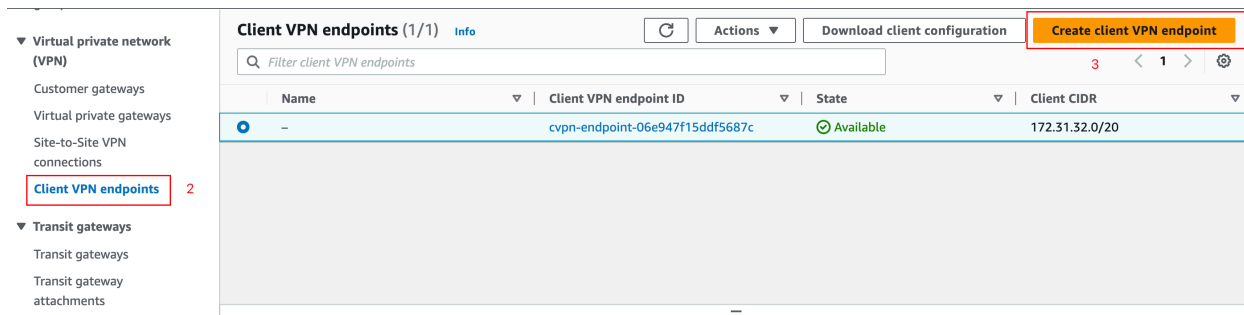
Metadata document Info

Choose file

File needs to be a valid UTF-8 XML document.

创建AWS Client VPN端点

1. 在您选择的AWS区域中打开Amazon VPC控制台。
2. 在左侧导航中，选择**客户端VPN端点**，位于**虚拟专用网络(VPN)**下。
3. 点击**创建客户端VPN端点**。
4. 在**客户端IPv4 CIDR**字段中输入您的远程用户的IP范围，以分配IP范围。
5. 对于**服务器证书ARN**，选择您创建的证书。
6. 对于身份验证选项，选择**使用基于用户的身份验证**，然后选择**联合身份验证**。
7. 对于**SAML提供商ARN**，选择您创建的身份提供商。
8. 点击**创建客户端VPN端点**。



Client IPv4 CIDR [Info](#)

The IP address range, in CIDR notation, from which client IP addresses are allocated.

172.31.32.0/20

CIDR block cannot be larger than /12 or smaller than /22.

Authentication information [Info](#)

Server certificate ARN

The server certificate must be provisioned with or imported into AWS Certificate Manager (ACM).

arn:aws:acm:ap-southeast-1:580652580210:certificate/f028f870-16ee-41b7-8...

Authentication options

Choose one or a combination of authentication methods to use.

- Use mutual authentication
- Use user-based authentication

User-based authentication options

- Active directory authentication
- Federated authentication

SAML provider ARN

The ARN of SAML provider.

arn:aws:iam::580652580210:saml-provider/casdoor

Self-service SAML provider ARN - optional [Info](#)

Select self-service SAML provider ARN

将客户端VPN与目标VPC关联

1. 在客户端VPN选项中选择目标网络关联，然后点击关联目标网络。
2. 从下拉菜单中，选择您想要将您的端点关联的目标VPC和子网。

The screenshot displays the AWS IAM console interface for managing Client VPN endpoints. On the left, a navigation menu includes sections for Virtual private network (VPN), Transit gateways, and Traffic Mirroring. The main content area is titled "Client VPN endpoints (1/1)" and shows a table with one endpoint: "cvpn-endpoint-06e947f15ddf5687c" in an "Available" state with a Client CIDR of "172.31.32.0/20". Below this, a detailed view for the selected endpoint "cvpn-endpoint-06e947f15ddf5687c" is shown, with the "Target network associations" tab selected. This sub-view shows a table with one association: "cvpn-assoc-0bf639762212d5a04" in an "Associated" state, linked to "subnet-0596ebfd975cdd125" and security group "sg-09d2a80e3c2795429".

Name	Client VPN endpoint ID	State	Client CIDR
-	cvpn-endpoint-06e947f15ddf5687c	Available	172.31.32.0/20

Association ID	State	Network ID	Security groups	Endpoint ID
cvpn-assoc-0bf639762212d5a04	Associated	subnet-0596ebfd975cdd125	sg-09d2a80e3c2795429	cvpn-endpoint-06e947f15d

配置SAML组特定授权

1. 在您的客户端VPN选项中，选择**授权规则**选项卡，然后点击**添加授权规则**。
2. 对于要启用的目标网络，指定在先决条件中创建的EC2实例的IP地址。例如，`172.31.16.0/20`。
3. 在授予访问权限下，选择**允许特定访问组中的用户访问**。例如，`casdoor`。
4. 提供一个可选的描述，然后点击**添加授权规则**。

Add authorization rule [Info](#)

Add authorization rules to grant clients access to the networks.

Details

Client VPN endpoint ID
cvpn-endpoint-06e947f15ddf5687c

Destination network to enable access
The IP address, in CIDR notation, of the destination network.

172.31.16.0/20 2

Grant access to:

Allow access to all users

Allow access to users in a specific access group 3

Access group ID
Unique group identifier. It can be active directory SID or group name in IDP.

casdoor

Description - optional
A brief description of the authorization rule.

description

4

Cancel **Add authorization rule**

连接到客户端VPN

1. 选择您刚刚创建的客户端VPN端点。现在应该处于可用状态。
2. 点击下载客户端配置，将配置文件下载到您的桌面。
3. 在您的机器上打开AWS Client VPN桌面应用程序。
4. 在顶部菜单中，选择文件和管理配置文件。
5. 点击添加配置文件，并指向最近下载的文件。
6. 您现在应该在AWS Client VPN软件的列表中看到该配置文件。选择它并点击连接。

- Virtual private cloud
- Your VPCs New
- Subnets
- Route tables
- Internet gateways
- Egress-only internet gateways
- DHCP option sets
- Elastic IPs
- Managed prefix lists
- Endpoints
- Endpoint services
- NAT gateways
- Peering connections
- Security

Client VPN endpoints (1/1) Info

< 1 > ⚙️

Name	Client VPN endpoint ID	State	Client CIDR
-	cvpn-endpoint-06e947f15ddf5687c	Available	172.31.32.0/20

cvpn-endpoint-06e947f15ddf5687c ×

[Details](#) | [Target network associations](#) | [Security groups](#) | [Authorization rules](#) | [Route table](#) | [Connections](#) | [Tags](#)

Details			
Client VPN endpoint ID	Server certificate ARN	Connection log	Transport protocol
cvpn-endpoint-06e947f15ddf5687c	arn:aws:acm:ap-southeast-1:580652580210:certificate/f028f870-16ee-41b7-8b4e-66a2a0ebfe33	false	udp
Description		Cloudwatch log group	Split-tunnel
-		-	Disabled

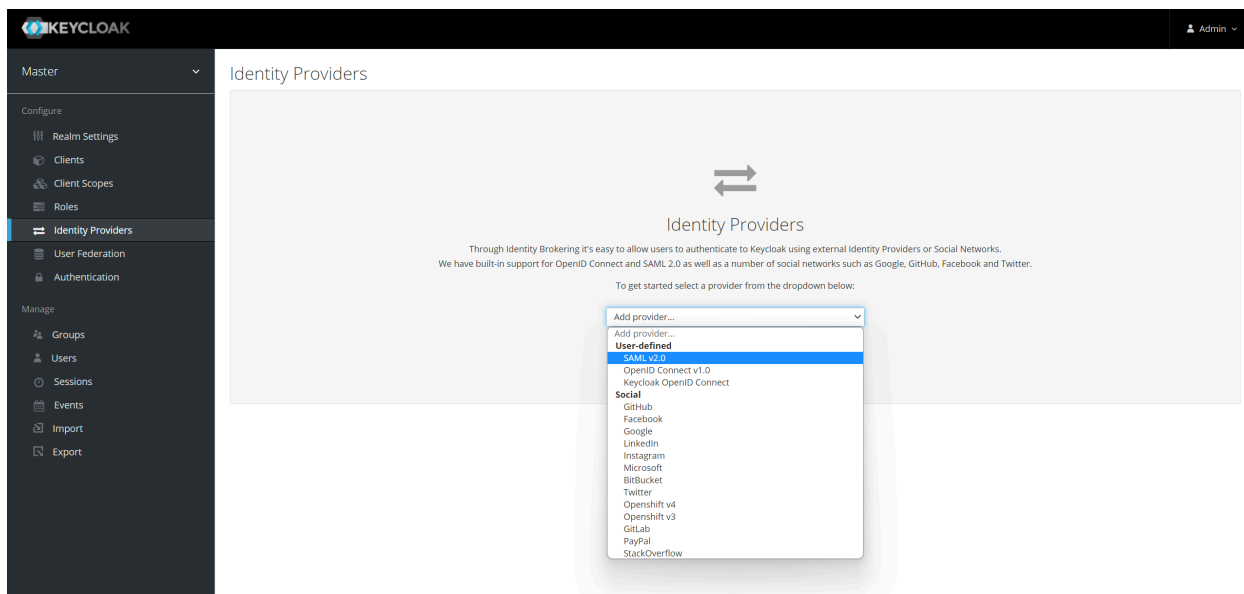
Keycloak

在Keycloak中的Casdoor作为SAML IdP

本指南将向您展示如何配置Casdoor和Keycloak，以在Keycloak中添加Casdoor作为SAML IdP。

在Keycloak中添加SAML IdP

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



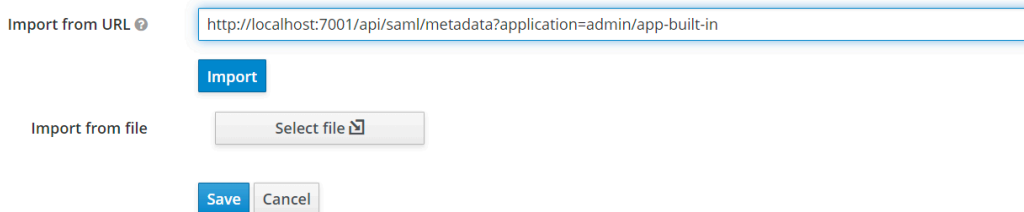
! 信息

您可以访问[Keycloak SAML身份提供商文档](#)以获取更详细的信息。

在Keycloak IdP编辑页面中输入**别名**和**从URL导入**。从URL导入的内容可以在Casdoor

应用程序编辑页面上找到。 点击**导入**，SAML配置将自动填充。

▼ Import External IDP Config ?



记住**服务提供商实体ID**并保存配置。

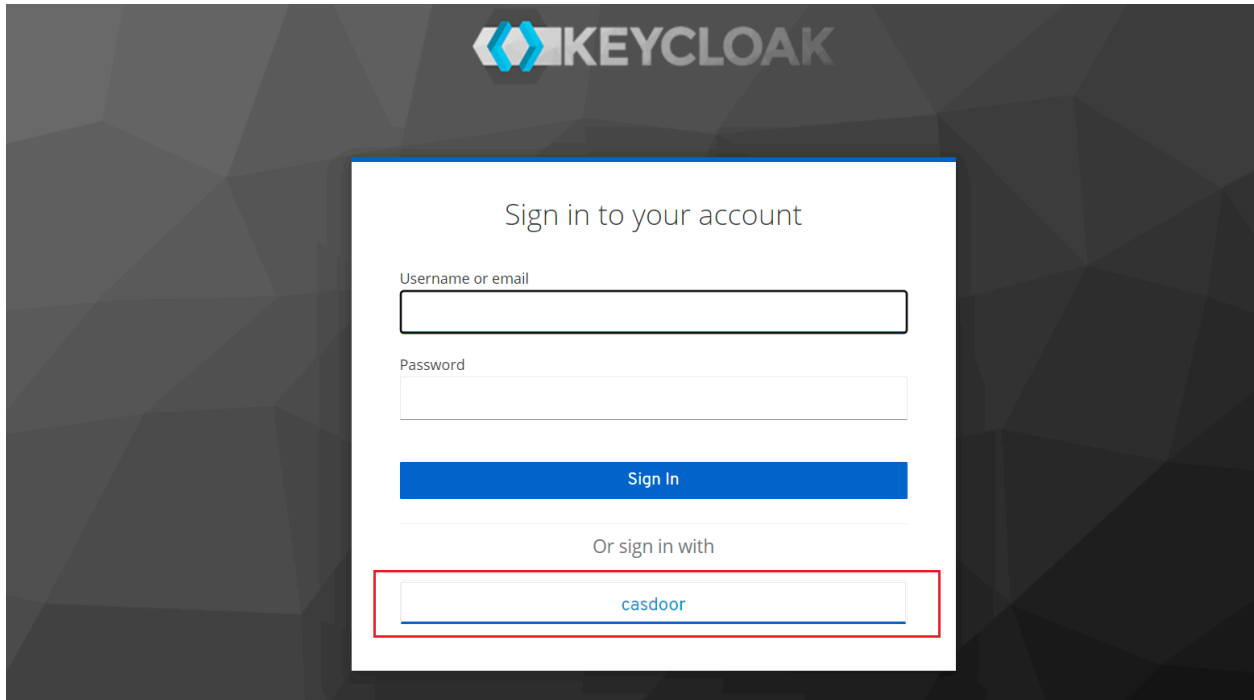
在Casdoor中配置SAML应用程序

在应用程序编辑页面中，添加一个包含来自Keycloak的**服务提供商实体ID**的重定向URL。另外，确保为Keycloak启用SAML压缩。

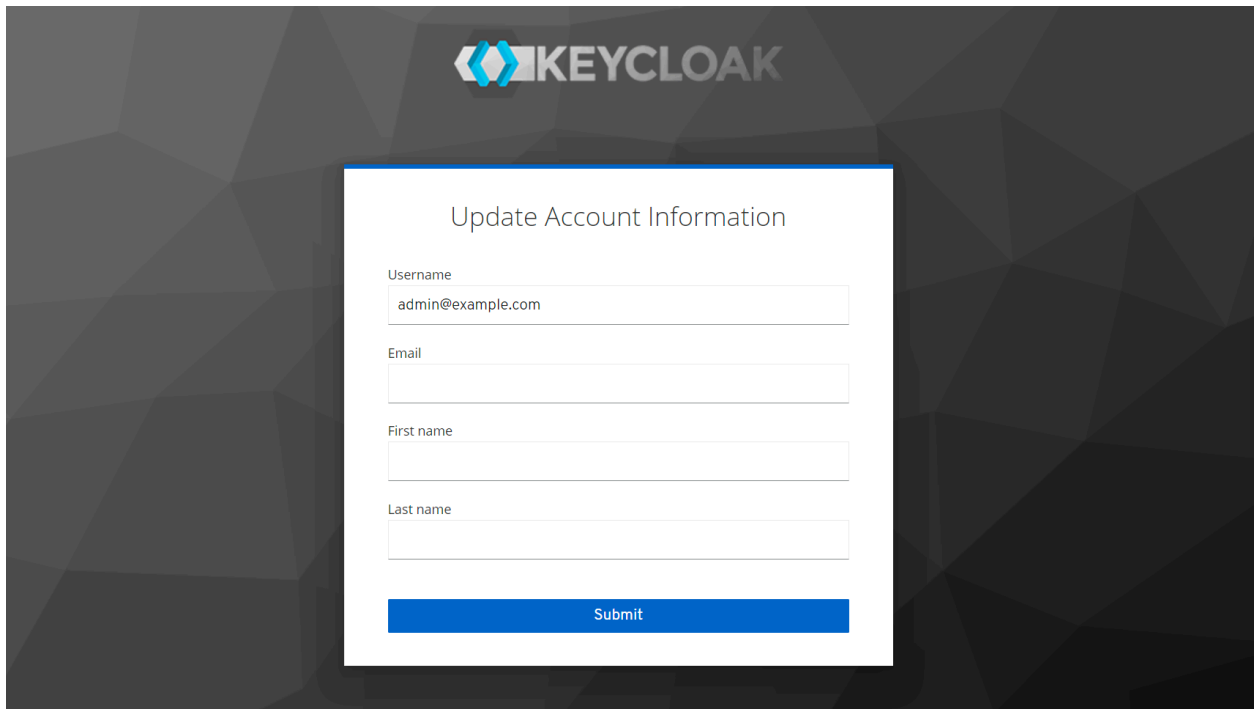


使用Casdoor SAML登录

打开Keycloak登录页面，您会发现一个额外的按钮，允许您使用Casdoor SAML提供商登录Keycloak。



点击该按钮，您将被重定向到Casdoor SAML提供商进行身份验证。身份验证成功后，您将被重定向回Keycloak。然后你需要将用户分配给应用程序。



我们还提供了一个演示视频，展示了整个过程，希望对您有所帮助。

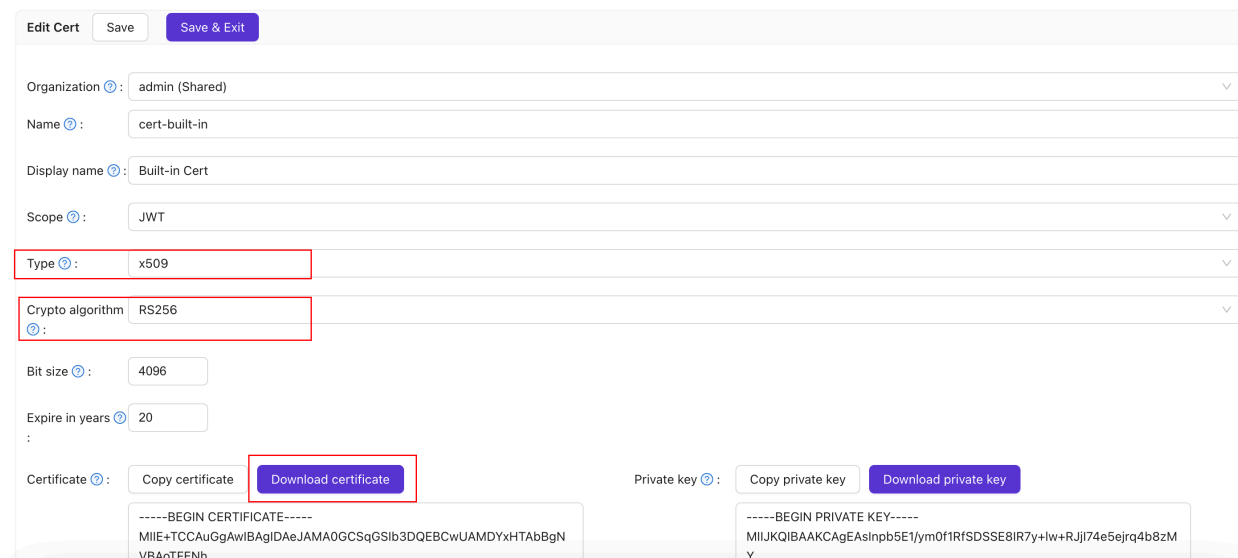
Google Workspace

将Casdoor作为Google Workspace中的SAML IdP

本指南将向您展示如何配置Casdoor和Google Workspace，以在Google Workspace中添加Casdoor作为SAML IdP。

添加证书

在Casdoor中，添加一种类型为X.509的证书，使用RSA加密算法，并下载它。



The screenshot shows the 'Edit Cert' form in Casdoor. The form includes the following fields and buttons:

- Organization: admin (Shared)
- Name: cert-built-in
- Display name: Built-in Cert
- Scope: JWT
- Type: x509 (highlighted with a red box)
- Crypto algorithm: RS256 (highlighted with a red box)
- Bit size: 4096
- Expire in years: 20
- Certificate: Copy certificate, Download certificate (highlighted with a red box)
- Private key: Copy private key, Download private key

The Certificate field contains the following text:

```
-----BEGIN CERTIFICATE-----
MIIE+TCCAuOggAwIBAgIDAeJAMA0GCSqGSIb3DQEBCwUAMDYxHTAbBgN
VBAoTFENh
```

The Private key field contains the following text:

```
-----BEGIN PRIVATE KEY-----
MIlJKQIBAAKAgEAslnpb5E1jym0f1RfSDSSE8IR7y+lw+RJJi74e5ejrq4b8zM
Y
```

配置SAML应用程序

在应用程序编辑页面上，选择您刚刚创建的证书。将您将使用的Google应用程序的域名添加到重定向URL中，例如google.com。

Cert : cert-built-in

Redirect URLs :

Redirect URL	Action
🔗 google.com	⬆️ ⬇️ 🗑️
🔗 gmail.com	⬆️ ⬇️ 🗑️

在SAML回复URL字段中，输入 `https://www.google.com/a/<your domain>/acs`，这是ACS URL。您可以在这里找到有关ACS URL的相关信息：[SSO断言要求](#)。

SAML reply URL : `https://www.google.com/a/casbin.com/acs`


Enable SAML compression 

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" xmlns:protocol="urn:oasis:names:tc:SAML:2.0:protocol">
  <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+TCCAuGgAwIBAgIDAeJAMAOGCSqGS1b3DQEBcwUAMDYxHTAbBgNVBAoTFENhc2Rvd3J3JnYw...
        </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/world...
  </IDPSSODescriptor>
</EntityDescriptor>
```

[Copy SAML metadata URL](#)

复制登录页面URL。这将在下一步中使用。

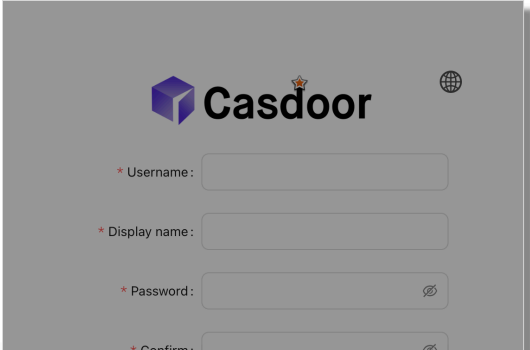
Providers :

Name	Category	Type	Can signup	Can signin	Can unlink	Prompted	Rule	Action
📁 No data								

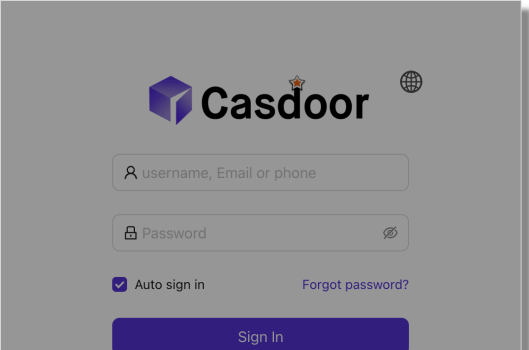
Preview :

[Copy signup page URL](#)

[Copy signin page URL](#)



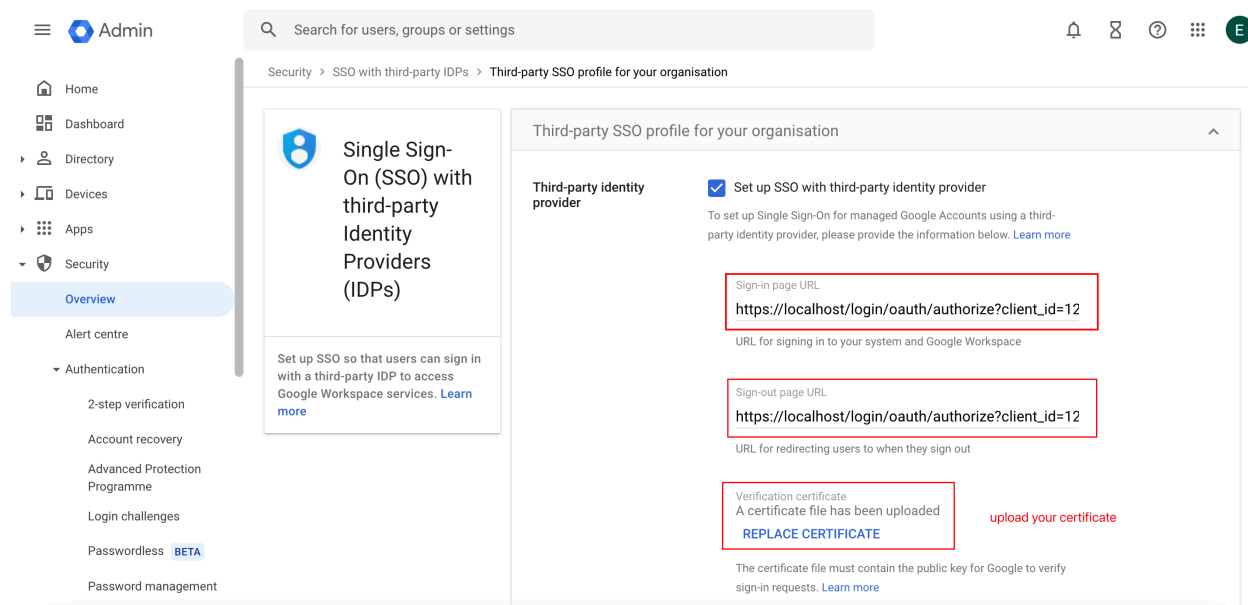
A screenshot of the Casdoor user registration (signup) page. It features the Casdoor logo at the top, followed by input fields for Username, Display name, Password, and Confirm. Each field is marked with an asterisk indicating it is required. The Confirm field includes an eye icon for password visibility.



A screenshot of the Casdoor user login (signin) page. It features the Casdoor logo at the top, followed by input fields for Username, Email or phone, and Password. There is a checkbox for 'Auto sign in' and a 'Forgot password?' link. A prominent blue 'Sign In' button is at the bottom.

为Google Workspace添加第三方SAML IdP

在Google Workspace管理控制台中，导航到**安全性**，然后是**概览**。寻找与**第三方IdP的SSO**部分。点击“添加SSO配置文件”以访问编辑页面。勾选“与第三方身份提供商设置SSO”复选框。将复制的登录页面URL粘贴到**登录页面URL**和**登出页面URL**字段中。上传在上一步中下载的证书。点击“保存”以保存更改。



添加测试用户

在Google Workspace中，创建一个用户名为“test”的用户（您可以自定义用户名，这只是一个例子）。

Your new user can start using Google Workspace within 24 hours. In most cases, it should just take a few minutes.



test test

Username: test@casbin.com

Password: •••••••••••••••• 

[COPY PASSWORD](#) [PRINT](#)

Send sign-in instructions

The email will provide a link to set the password and sign in to Google Workspace

[PREVIEW AND SEND](#)




The user will be assigned licences based on your current subscriptions. [View billing](#)

[ADD ANOTHER USER](#)

DONE

在Casdoor中，添加一个与Google Workspace中设置的用户名相同的用户。确保选择适当的组织并输入用户的电子邮件地址。

Organization ⓘ :	built-in
ID ⓘ :	4899cef3-8eeb-485a-8f6d-12b41df0d8d2
Name ⓘ :	test
Display name ⓘ :	test
Avatar ⓘ :	Preview:  <input type="button" value="Upload a photo..."/>
User type ⓘ :	normal-user
Password ⓘ :	<input type="button" value="Modify password..."/>
Email ⓘ :	test@casbin.com
Phone ⓘ :	+1 <input type="text" value="34086653696"/>

以“google.com”为例，按照以下步骤操作：

1. 点击Google.com页面上的登录按钮。输入用户的电子邮件地址以启动登录过程。
2. 您将被重定向到Casdoor页面。在Casdoor页面上，输入相应的电子邮件地址和密码。
3. 如果登录成功，您将被重定向回google.com。



Google Search

I'm Feeling Lucky

Google offered in: [日本語](#)

Appgate (POST)

在Appgate中的Casdoor作为SAML IdP

Appgate accepts `SAMLResponse` by POST request. If you use another SP that also supports POST requests, you can refer to this document.

Casdoor 配置

转到您的 Casdoor 帐户并添加一个新的应用程序。


在应用程序中输入基本的 SAML 配置：

- 重定向URLs：输入一个唯一的名称。在您的SP中，这可能被称为 `Audience` 或 `Entity ID`。见下表。

Redirect URLs [?](#) :

Redirect URL
🔗 appgate
🔗 https://git.casbin.com/user/oauth2/casdoor/callback
🔗 http://localhost:3000/callback


- Reply URL—键入验证SAML响应的ACS（声明消费者服务）的URL。请参阅下表。


Grant types  :

Authorization Code × Password ×

SAML Reply URL

 :

 <https://mycontroller.mycompany.com/admin/saml>


Enable SAML
compress  :



管理员验证	用户认证
重定向URL = "AppGate"	重定向URL = "AppGate Client"
SAML回复URL = https://mycontroller.your-site-url.com/admin/saml	SAML回复URL = https://redirectserver.your-site-url.com/saml


下载XML元数据文件

您可以复制元数据的URL，并从浏览器下载文件。

Enable SAML
compress  :

SAML metadata  :

```
<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+TCCAuGgAwIBAgIDAeJAMAOGCSqGSIb3DQEBCwUAMDYxHTAbBgNVBAoTFENhc2Rv3I3JnYw5pemF0aF9uMRUwEwYDQDEwYDYNb6
    </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="https://door.casdoor.com/login/saml/authorize/admin/app-gitea"></SingleSignOnService>
<Attribute Name="Email" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic" FriendlyName="E-Mail" xmlns="urn:oasis:names:tc:SAML:2.0:assertion"></Attribute>
<Attribute Name="DisplayName" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic" FriendlyName="displayName" xmlns="urn:oasis:names:tc:SAML:2.0:assertion"></Attribute>
```

 Copy SAML metadata URL

在Appgate中添加SAML IdP

在您的AppGate SDP控制台:

- 选择系统 > 身份提供商。
- 创建一个新的身份提供商。
- 选择SAML类型。
- 按照下表中的详细信息开始配置您的身份提供程序。

	管理员验证
姓名	输入一个唯一的名称，例如“Casdoor SAML Admin”。
单点登录URL	见下文
发行人	见下文
受众	在Casdoor应用中输入 重定向URL
公共证书	见下文

- 上传XML元数据文件以自动完成**单点登录**、**发行人**和**公共证书**字段。
- 点击**选择文件**并选择您之前下载的元数据文件 - 这将自动完成相关字段。

映射属性

将**名称**映射到**用户名**。您完成的表格应该看起来像这样:

Name mapped to claim username

测试集成

在您的AppGate SDP控制器控制台:

- 退出管理员界面。
- 使用以下信息登录:
 - 身份提供商 - 从下拉列表中选择您的Azure IdP。
 - 点击**用浏览器登录**以连接到您的验证器。
- 您可能会看到以下消息: "您没有任何管理权限" - 这确认了测试用户凭据已经被您的身份提供商成功验证。

访问策略

您需要修改访问策略, 以允许管理员使用SAML IdP登录到Appgate。 输入内置管理员策略:

您完成的表格应该看起来像这样:

Editing Policy - Admin

- Enabled
- Disabled

Assignment - Active when custom logic is met ∨

[+ Add New](#)

Custom Logic	
(1 OR 3) AND 2	
1 Identity Provider is local	 
2 user.username is admin	
3 Identity Provider is Casdoor SAML Admin	

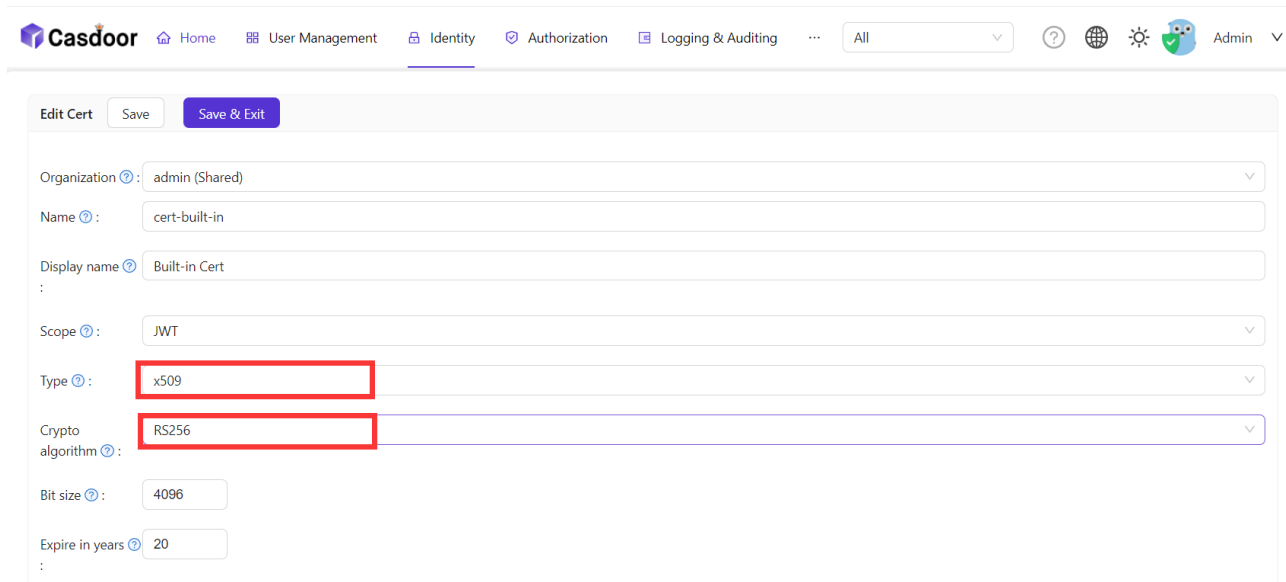
腾讯云

在腾讯云中使用的Casdoor作为SAML IdP

本指南将向您展示如何配置Casdoor和腾讯云，以在腾讯云中添加Casdoor作为SAML IdP。

复制Saml元数据

在Casdoor中，添加一种类型为X.509的证书，使用RSA加密算法。

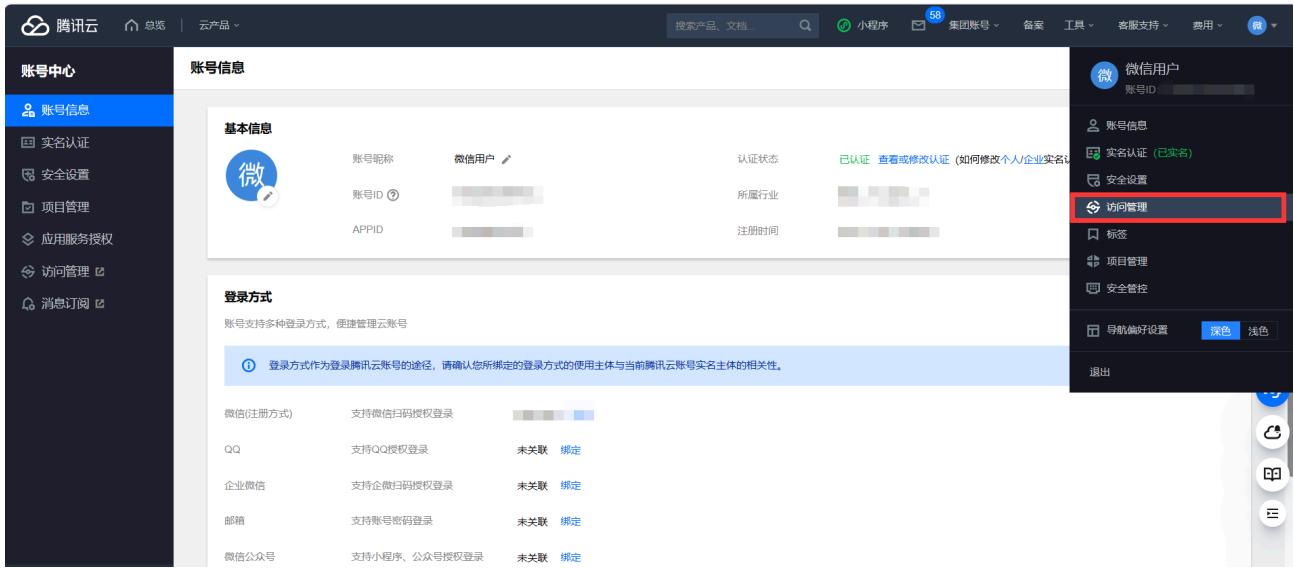


然后复制Casdoor中的SamlMetadata。



在腾讯云中添加SAML IdP

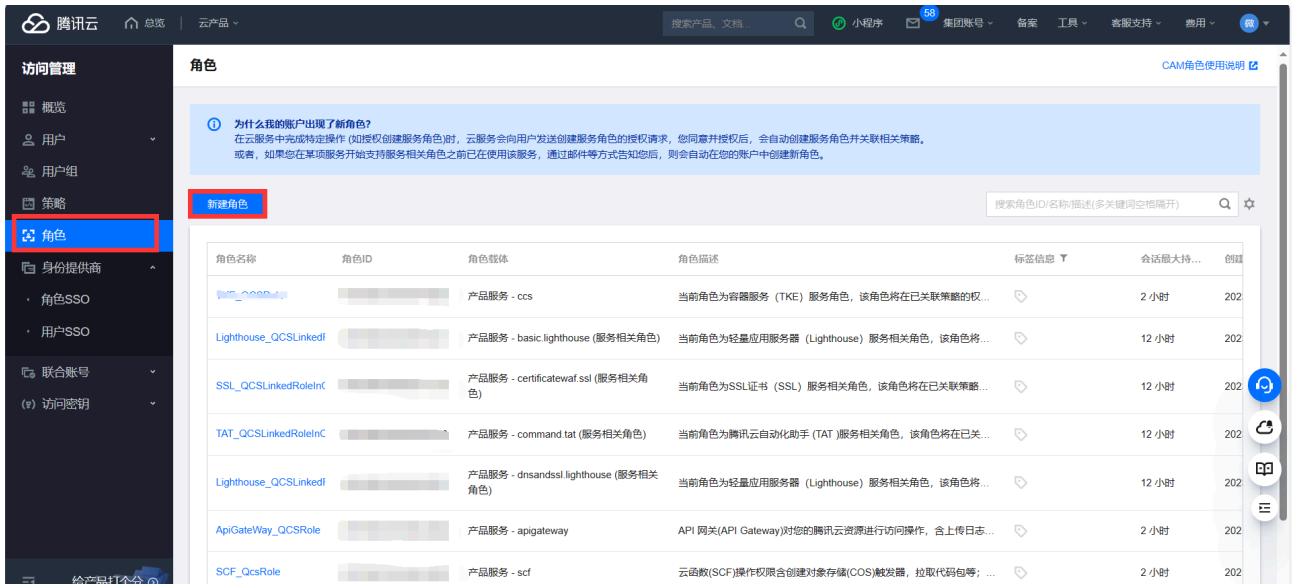
登录腾讯云并进入访问管理界面。



创建一个新的身份提供商并将之前复制的saml元数据上传到腾讯云。

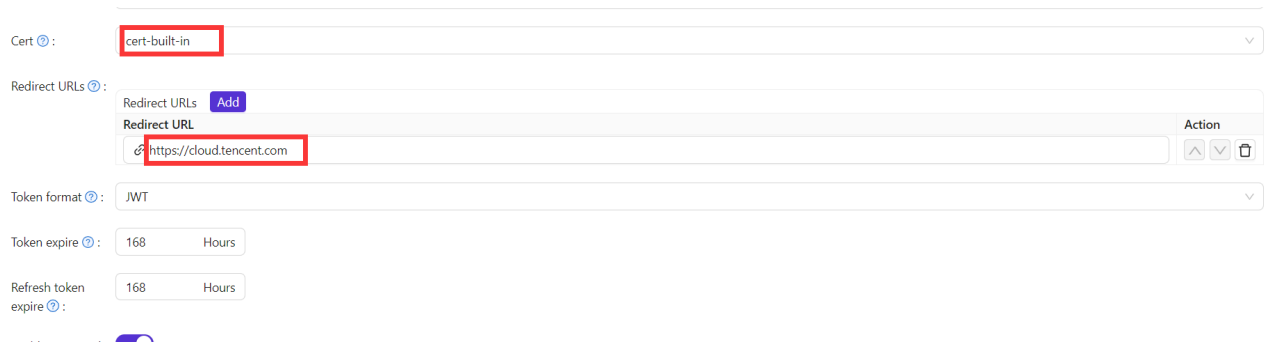


然后创建一个新的角色，并选择之前的身份提供商作为idp提供商。

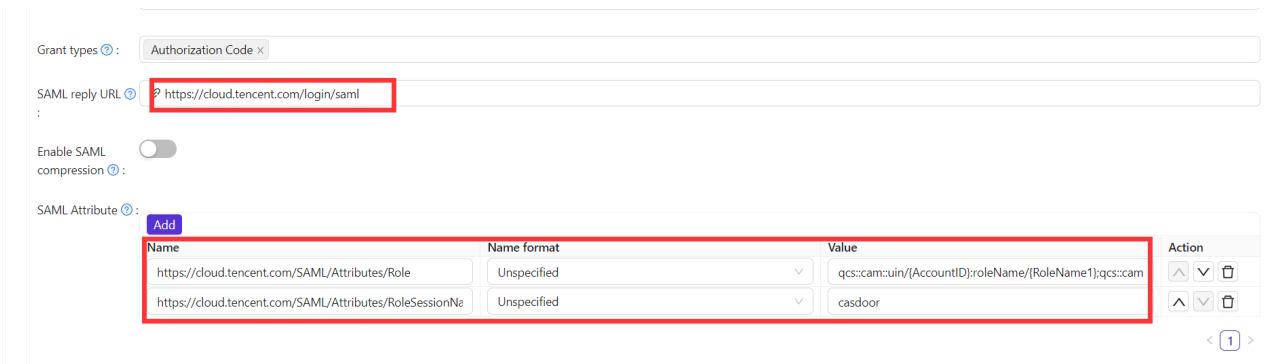


在Casdoor中配置SAML应用

在应用编辑页面, 选择你刚刚创建的证书。在重定向URLs中添加你将使用的腾讯云应用的域名。



在应用编辑页面, 输入ACS URL并配置Saml属性。



Saml属性的配置信息如下:

名称	名称格式	值
<code>https://cloud.tencent.com/SAML/Attributes/Role</code>	未指定	<code>qcs::cam::uin/{AccountID}:roleName/{RoleName1};qcs::cam::uin/{AccountID}:roleName/{RoleName2};qcs::cam::uin/{AccountID}:roleName/{RoleName3};qcs::cam::uin/{AccountID}:roleName/{RoleName4};qcs::cam::uin/{AccountID}:roleName/{RoleName5};qcs::cam::uin/{AccountID}:roleName/{RoleName6};qcs::cam::uin/{AccountID}:roleName/{RoleName7};qcs::cam::uin/{AccountID}:roleName/{RoleName8};qcs::cam::uin/{AccountID}:roleName/{RoleName9};qcs::cam::uin/{AccountID}:roleName/{RoleName10};qcs::cam::uin/{AccountID}:roleName/{RoleName11};qcs::cam::uin/{AccountID}:roleName/{RoleName12};qcs::cam::uin/{AccountID}:roleName/{RoleName13};qcs::cam::uin/{AccountID}:roleName/{RoleName14};qcs::cam::uin/{AccountID}:roleName/{RoleName15};qcs::cam::uin/{AccountID}:roleName/{RoleName16};qcs::cam::uin/{AccountID}:roleName/{RoleName17};qcs::cam::uin/{AccountID}:roleName/{RoleName18};qcs::cam::uin/{AccountID}:roleName/{RoleName19};qcs::cam::uin/{AccountID}:roleName/{RoleName20};qcs::cam::uin/{AccountID}:roleName/{RoleName21};qcs::cam::uin/{AccountID}:roleName/{RoleName22};qcs::cam::uin/{AccountID}:roleName/{RoleName23};qcs::cam::uin/{AccountID}:roleName/{RoleName24};qcs::cam::uin/{AccountID}:roleName/{RoleName25};qcs::cam::uin/{AccountID}:roleName/{RoleName26};qcs::cam::uin/{AccountID}:roleName/{RoleName27};qcs::cam::uin/{AccountID}:roleName/{RoleName28};qcs::cam::uin/{AccountID}:roleName/{RoleName29};qcs::cam::uin/{AccountID}:roleName/{RoleName30};qcs::cam::uin/{AccountID}:roleName/{RoleName31};qcs::cam::uin/{AccountID}:roleName/{RoleName32};qcs::cam::uin/{AccountID}:roleName/{RoleName33};qcs::cam::uin/{AccountID}:roleName/{RoleName34};qcs::cam::uin/{AccountID}:roleName/{RoleName35};qcs::cam::uin/{AccountID}:roleName/{RoleName36};qcs::cam::uin/{AccountID}:roleName/{RoleName37};qcs::cam::uin/{AccountID}:roleName/{RoleName38};qcs::cam::uin/{AccountID}:roleName/{RoleName39};qcs::cam::uin/{AccountID}:roleName/{RoleName40};qcs::cam::uin/{AccountID}:roleName/{RoleName41};qcs::cam::uin/{AccountID}:roleName/{RoleName42};qcs::cam::uin/{AccountID}:roleName/{RoleName43};qcs::cam::uin/{AccountID}:roleName/{RoleName44};qcs::cam::uin/{AccountID}:roleName/{RoleName45};qcs::cam::uin/{AccountID}:roleName/{RoleName46};qcs::cam::uin/{AccountID}:roleName/{RoleName47};qcs::cam::uin/{AccountID}:roleName/{RoleName48};qcs::cam::uin/{AccountID}:roleName/{RoleName49};qcs::cam::uin/{AccountID}:roleName/{RoleName50};qcs::cam::uin/{AccountID}:roleName/{RoleName51};qcs::cam::uin/{AccountID}:roleName/{RoleName52};qcs::cam::uin/{AccountID}:roleName/{RoleName53};qcs::cam::uin/{AccountID}:roleName/{RoleName54};qcs::cam::uin/{AccountID}:roleName/{RoleName55};qcs::cam::uin/{AccountID}:roleName/{RoleName56};qcs::cam::uin/{AccountID}:roleName/{RoleName57};qcs::cam::uin/{AccountID}:roleName/{RoleName58};qcs::cam::uin/{AccountID}:roleName/{RoleName59};qcs::cam::uin/{AccountID}:roleName/{RoleName60};qcs::cam::uin/{AccountID}:roleName/{RoleName61};qcs::cam::uin/{AccountID}:roleName/{RoleName62};qcs::cam::uin/{AccountID}:roleName/{RoleName63};qcs::cam::uin/{AccountID}:roleName/{RoleName64};qcs::cam::uin/{AccountID}:roleName/{RoleName65};qcs::cam::uin/{AccountID}:roleName/{RoleName66};qcs::cam::uin/{AccountID}:roleName/{RoleName67};qcs::cam::uin/{AccountID}:roleName/{RoleName68};qcs::cam::uin/{AccountID}:roleName/{RoleName69};qcs::cam::uin/{AccountID}:roleName/{RoleName70};qcs::cam::uin/{AccountID}:roleName/{RoleName71};qcs::cam::uin/{AccountID}:roleName/{RoleName72};qcs::cam::uin/{AccountID}:roleName/{RoleName73};qcs::cam::uin/{AccountID}:roleName/{RoleName74};qcs::cam::uin/{AccountID}:roleName/{RoleName75};qcs::cam::uin/{AccountID}:roleName/{RoleName76};qcs::cam::uin/{AccountID}:roleName/{RoleName77};qcs::cam::uin/{AccountID}:roleName/{RoleName78};qcs::cam::uin/{AccountID}:roleName/{RoleName79};qcs::cam::uin/{AccountID}:roleName/{RoleName80};qcs::cam::uin/{AccountID}:roleName/{RoleName81};qcs::cam::uin/{AccountID}:roleName/{RoleName82};qcs::cam::uin/{AccountID}:roleName/{RoleName83};qcs::cam::uin/{AccountID}:roleName/{RoleName84};qcs::cam::uin/{AccountID}:roleName/{RoleName85};qcs::cam::uin/{AccountID}:roleName/{RoleName86};qcs::cam::uin/{AccountID}:roleName/{RoleName87};qcs::cam::uin/{AccountID}:roleName/{RoleName88};qcs::cam::uin/{AccountID}:roleName/{RoleName89};qcs::cam::uin/{AccountID}:roleName/{RoleName90};qcs::cam::uin/{AccountID}:roleName/{RoleName91};qcs::cam::uin/{AccountID}:roleName/{RoleName92};qcs::cam::uin/{AccountID}:roleName/{RoleName93};qcs::cam::uin/{AccountID}:roleName/{RoleName94};qcs::cam::uin/{AccountID}:roleName/{RoleName95};qcs::cam::uin/{AccountID}:roleName/{RoleName96};qcs::cam::uin/{AccountID}:roleName/{RoleName97};qcs::cam::uin/{AccountID}:roleName/{RoleName98};qcs::cam::uin/{AccountID}:roleName/{RoleName99};qcs::cam::uin/{AccountID}:roleName/{RoleName100}</code>
<code>https://cloud.tencent.com/SAML/Attributes/RoleSessionName</code>	未指定	casdoor

信息

- 在角色源属性中，将{AccountID}、{RoleName}和{ProviderName}替换为以下内容：
- 将{AccountID}替换为您的腾讯云账户ID，可以在[账户信息 - 控制台](#)中查看。
- 将{RoleName}替换为您在腾讯云中创建的角色名称，可以在[角色 - 控制台](#)中查看。
- 将{ProviderName}替换为您在腾讯云中创建的SAML身份提供商的名称，可以在[身份提供商 - 控制台](#)中查看。

您可以访问[腾讯云SAML身份提供商文档](#)以获取更详细的信息。

使用Casdoor SAML登录

SAML的一般登录步骤如下：用户 → 腾讯云（未登录） → 重定向到Casdoor进行登录 → 腾讯云（已登录）。现在，使用外部代码模拟前两步并生成一个重定向到Casdoor的URL。示例代码：

```
func main() {
    res, err := http.Get("your casdoor application saml metadata url")
    if err != nil {
        panic(err)
    }

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

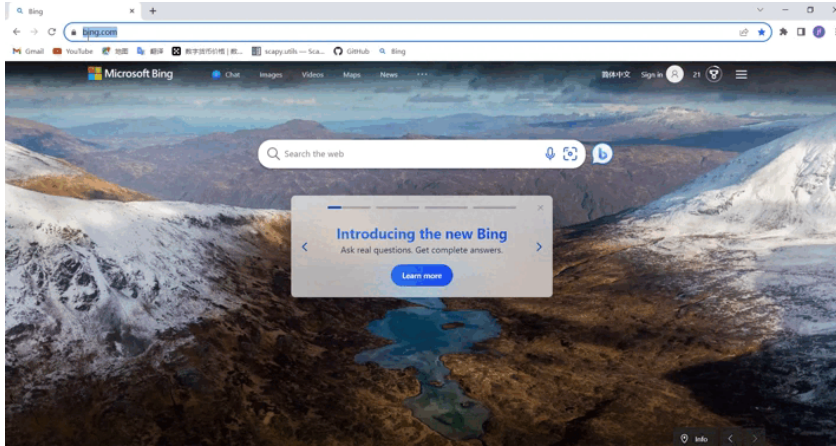
    metadata := &types.EntityDescriptor{}
    err = xml.Unmarshal(rawMetadata, metadata)
    if err != nil {
        panic(err)
    }

    certStore := dsig.MemoryX509CertificateStore{
        Roots: []*x509.Certificate{},
    }

    for _, kd := range metadata.IDPSSODescriptor.KeyDescriptors {
        for idx, xcert := range kd.KeyInfo.X509Data.X509Certificates {
            if xcert.Data == "" {
                panic(fmt.Errorf("metadata certificate(%d) must not be empty", idx))
            }
            certData, err := base64.StdEncoding.DecodeString(xcert.Data)
            if err != nil {
                panic(err)
            }

            idpCert, err := x509.ParseCertificate(certData)
        }
    }
}
```


一旦我们运行代码并获取到auth URL，点击URL就可以测试登录了。我们提供了一个这个过程的演示。



Face ID

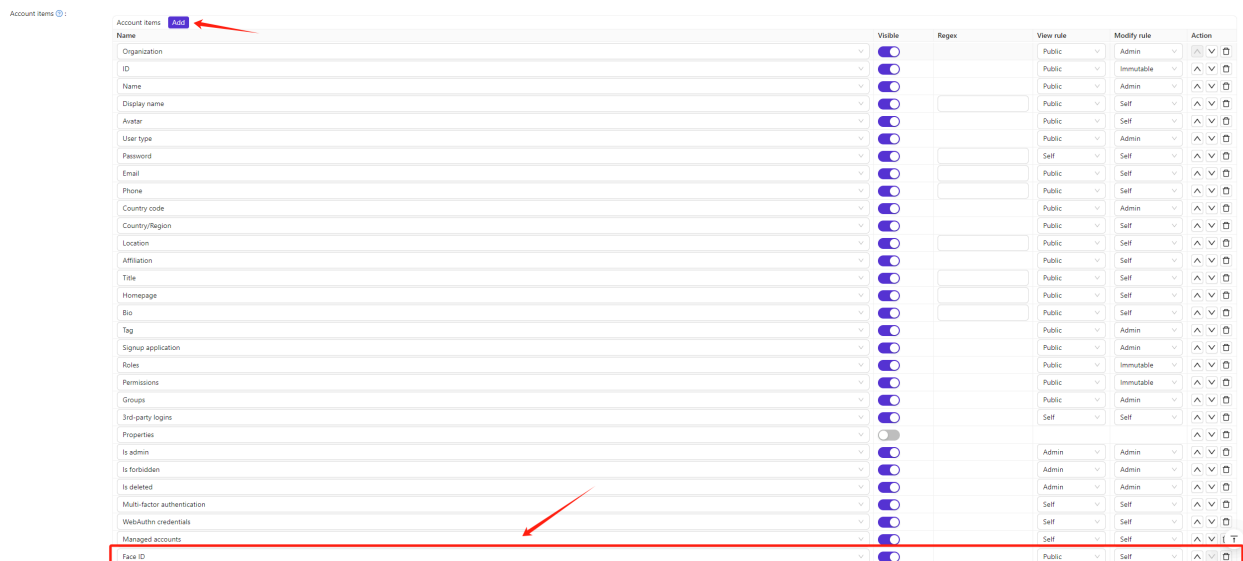
Overview

We've now incorporated `Face ID` login into Casdoor by leveraging `face-api.js`.

Activation method

Add the Face ID option in the organization's Account items

User Management → Organizations → Choose an organization → Locate the Account items section and incorporate Face ID



Afterwards, you'll find the Face ID option under User,

where users can upload their facial data to be used for logging in

User Management → Users → Choose a user → Find Face Ids, and add facial data. You can add up to 5 facial data entries, and you can give each facial data a custom name.



Third step: Incorporate Face ID as a login option under the Signin methods section of the application

Identity → Applications → Choose an application → Go to the Signin methods section and incorporate Face ID as a login option.



Finally, you can log in using the Face ID method on the login page

1. On the login page, select the Face ID login method.
2. Enter the username, click on Sign in with Face ID.
3. Once you grant permission to access your camera, you'll be able to log in using Face ID.



Password Email Face ID WebAuthn

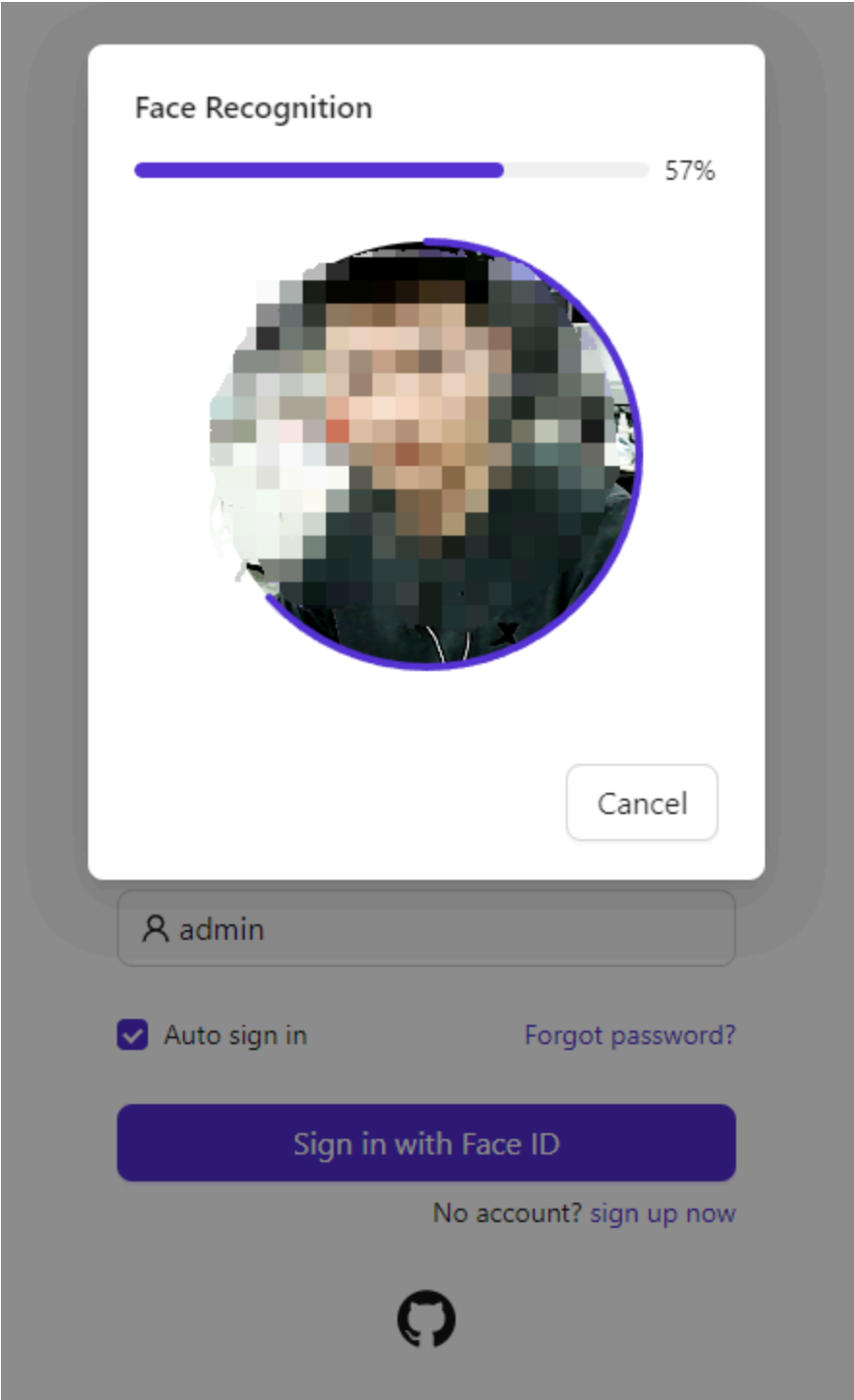
Auto sign in

[Forgot password?](#)

Sign in with Face ID

No account? [sign up now](#)





Here is a video demonstrating how to configure Face ID login:

WebAuthn

概述

我们很高兴地通知Casdoor的客户，Casdoor现在支持使用WebAuthn登录。这意味着您可以使用您的生物识别，如指纹或面部识别，甚至U盘登录，只要您的设备支持这些酷炫的授权方法和WebAuthn。

什么是 WebAuthn?

WebAuthn是Web认证API，由W3C和FIDO与Google、Mozilla、Microsoft、Yubico等公司共同编写的规范。这个API允许服务器使用公钥加密而不是密码来注册和认证用户。它使服务器能够与设备内置的强大认证器集成，如Windows Hello或Apple的Touch ID。

简单来说，WebAuthn要求用户生成一个公钥-私钥对，并将公钥提供给网站。当用户想要登录一个网站时，网站生成一个随机数，并要求用户用他们的私钥加密它并将结果发送回来。收到结果后，网站使用公钥对其进行解密。如果解密后的数字与之前生成的随机数匹配，用户被视为合法用户，并被授予登录权限。公钥和必要信息（如用户名或用户的授权者信息）的组合被称为WebAuthn凭证，由网站存储。

公钥-私钥对独有且唯一地与三个信息关联：用户的用户名、用户的授权者和网站的URL。这意味着，如果（用户的用户名、用户的授权者和网站的URL）的组合相同，那么密钥对应该是相同的，反之亦然。

关于WebAuthn技术的更详细信息，您可以访问<https://webauthn.guide/>。

如何在Casdoor中使用WebAuthn?

在登录页面，您可能已经注意到了使用WebAuthn登录的选项。然而，如果你还没有

WebAuthn凭证（可以类比为WebAuth密码），这个教程将向你展示如何创建和管理一个凭证，然后使用它登录。

步骤0：修改配置并启用WebAuthn认证

在 `conf/app.conf` 文件中，你可以找到以下配置：

```
origin = "http://localhost:8000"
```

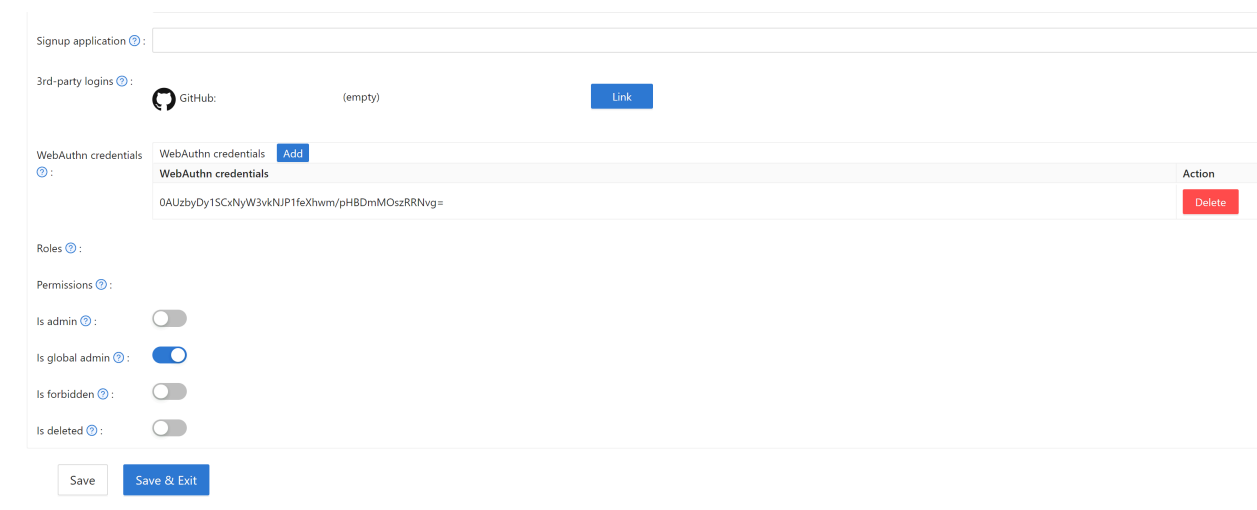
请确保此配置与您网站的URL完全匹配。

注意：只有HTTPS支持WebAuthn，除非你正在使用localhost。

接下来，以管理员身份登录，进入你的应用的编辑页面。打开"启用WebAuthn登录"开关。默认情况下，此功能未启用。

步骤1：进入"我的账户"页面

导航到账户页面。在这个页面，你应该看到"添加WebAuthn凭证"按钮和一个显示你之前注册的所有WebAuthn凭证的列表。



The screenshot shows the Casdoor user management interface. At the top, there is a "Signup application" field. Below it, the "3rd-party logins" section shows "GitHub" with an "(empty)" status and a "Link" button. The "WebAuthn credentials" section is active, displaying a table with one credential. The table has columns for "WebAuthn credentials" and "Action". The credential value is "0AUzbyDy1SCxNyW3wKNJP1feXhwm/pHBDmMOszRRNvg=" and the action is "Delete". Below the table, there are "Roles" and "Permissions" sections. The "Permissions" section has four toggle switches: "Is admin" (off), "Is global admin" (on), "Is forbidden" (off), and "Is deleted" (off). At the bottom, there are "Save" and "Save & Exit" buttons.

WebAuthn credentials	Action
0AUzbyDy1SCxNyW3wKNJP1feXhwm/pHBDmMOszRRNvg=	Delete

点击按钮并按照设备的指示在Casdoor中注册一个新的凭证。你可以使用列表中的"删

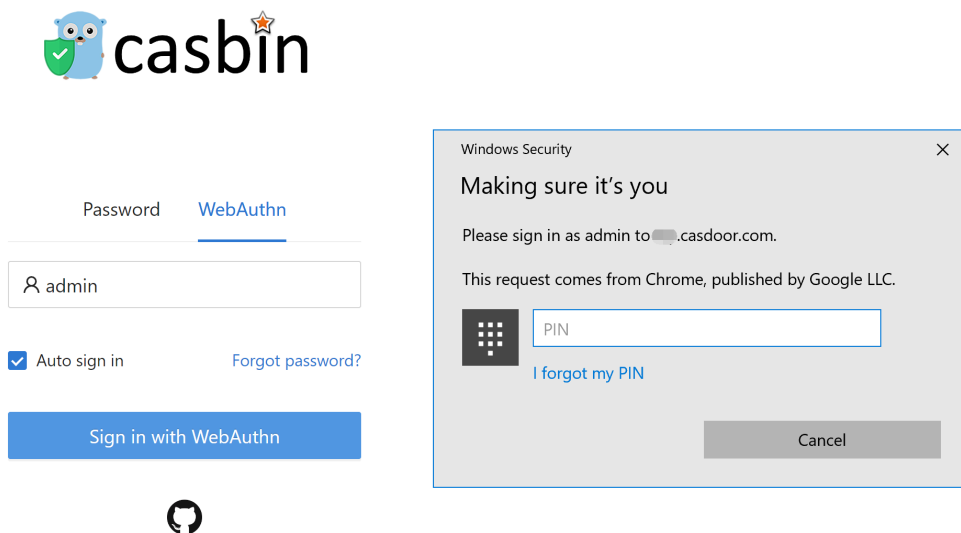
除"按钮删除任何凭证。

步骤2：使用WebAuthn登录

在开始这一步之前，确保你已经退出了Casdoor。

进入登录页面，选择WebAuthn登录方法，输入你的用户名，然后点击登录按钮。按照设备的指示操作。

(例如，如果你正在使用指纹和Windows Hello，你应该看到这样的东西)



然后你就会成功登录。



开发指南

开发指南

前端

Casdoor 前端开发指南

生成 Swagger 文件

生成 Swagger 文件

前端

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

这是一个 **Create-React-App (CRA)** 项目, 遵循下面概述的经典CRA文件夹结构:

文件/目录	描述
public	React的根HTML文件
src	源代码
craco.config.js	Craco配置文件。您可以在此处更改主题颜色 (默认为蓝色)
crowdin.yml	Crowdin i18n 配置文件
package.json	NPM/Yarn 依赖文件
yarn.lock	Yarn lockfile

在 `/src` 目录中, 您将找到几个重要的文件和文件夹:

文件/目录	描述
account	已登录用户的"我的个人资料"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，忘记密码页面等。
backend	调用Go后端API的SDK。它包含所有的 <code>fetch()</code> 调用
basic	Casdoor的主页（仪表板页面），包含了几个卡片小部件
common	共享UI部件
locales	i18n翻译文件以JSON格式，与我们的Crowdin项目同步： https://crowdin.com/project/casdoor-site
App.js	包含所有路由的入口JS文件
Setting.js	其他代码使用的实用函数
OrganizationListPage.js	组织列表的页面，与所有其他的 <code>XXXListPage.js</code> 文件类似
OrganizationEditPage.js	用于编辑一个组织的页面，与所有其他的 <code>XXXEditPage.js</code> 文件类似

生成 Swagger 文件

概述

我们知道，beego框架提供了生成swagger文件的支持，以通过名为"bee"的命令行工具来阐明API。Casdoor也是基于beego构建的。然而，我们发现由bee生成的swagger文件未能使用"@Tag"标签对API进行分类。所以，我们修改了原始的蜜蜂以实现这个功能。

如何写comment

大多数规则与原始蜜蜂注释格式完全相同。唯一的差异在于，API应根据"@Tag"标签划分为不同的组。因此，开发者有责任确保这个标签被正确添加。这是一个例子：

```
// @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() {
```

带有相同"@Tag"标签的API将被放入同一组。

如何生成swagger文件

0. 以正确的格式为API编写注释。
1. 获取此仓库: <https://github.com/casbin/bee>。
2. 构建修改后的蜜蜂。例如, 在casbin/bee的根目录中, 运行以下命令:

```
go build -o mybee .
```

3. 将mybee复制到casdoor的基础目录。
4. 在那个目录中, 运行以下命令:

```
mybee generate docs
```

5. (可选) 如果你想为特定的标签或API生成swagger文档, 以下是一些示例命令:

```
mybee generate docs --tags "Adapter API"  
mybee generate docs --tags "Adapter API,Login API"  
mybee generate docs --apis "add-adapter"  
mybee generate docs --apis "add-adapter,delete-adapter"
```

值得注意的是: 当提供多个标签/接口时, 我们只接受逗号(,)作为分隔符。

然后你会发现新的swagger文件已经生成。



组织

组织

概述

Casdoor的基本单位 — 组织

组织树

组织内的用户群组(group)

密码复杂度

支持不同的密码复杂度选项。

账户自定义

自定义用户帐户项目

自定义主题

了解如何为组织和组织内的应用程序定制主题

管理多因素认证项目

在组织中配置多因素认证项目

概述

组织是Casdoor的基本单位，它管理着用户和应用。如果一个用户登录到一个组织，他可以在无需再次登录的情况下访问属于该组织的所有应用。

在 [应用](#) 和 [提供商](#)配置中，选择一个组织是很重要的一项，它决定了一个用户是否能够使用特定的提供商访问该应用。

我们还可以在Casdoor建立LDAP。更多详细信息，请参阅 [LDAP 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	SHA-256	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	PBKDF2 是一个简单的派生函数，能够	Keycloak

名称	算法	描述	场景
	与 PBKDF2	抵制字典攻击和彩虹表攻击. 它是 Casdoor 为 Keycloak 同步器而采用的原生实现。如果您通过 Keycloak 同步器导入用户，请选择此选项。	

 提示

除了通过应用程序登录到 Casdoor 之外(重定向到Casdoor以便SSO)， Casdoor 用户也可以选择通过该组织的登录页面直接登录到 Casdoor:

`/login/<organization_name>`，例如示例站点中的

<https://door.casdoor.com/login/casbin>

组织树

群组是组织内用户的一个集合 一个用户可以属于多个群组

Group properties

- **Owner**: 群组所属的组织
- **Name**: 群组的唯一名称
- ``
- ``
- ``
- **Type**: 群组可以被区分为 **Physical** 或 **Virtual** 一个用户只能存在在一个 **Physical** 组中, 但可以存在在多个 **Virtual** 组中.
- **ParentGroup**: 群组的父级群组.(最顶级群组的父组是其组织)

管理群组

有两种方式管理群组

1. 在组织列表页, 你可以预览所有组织下的群组

Casdoor Home Organizations **Groups** Users Roles Permissions Models Adapters Applications Providers Chats Messages ...

Groups **Add**

Name	Organization	Created time	Updated time	Display name	Type	Parent group	Action
casdoor_virtual	built-in	2023-06-12 12:37:44	2023-06-12 12:37:51	Casdoor Project Virtual Team	Virtual		Edit Delete
casbin_virtual	built-in	2023-06-12 12:37:18	2023-06-12 12:37:36	Casbin Project Virtual Team	Virtual		Edit Delete
dev_frontend	built-in	2023-06-12 09:43:18	2023-06-12 12:35:51	Dev (Frontend)	Physical		Edit Delete
dev_backend	built-in	2023-06-12 09:20:28	2023-06-12 12:35:58	Dev (Backend)	Physical		Edit Delete
dev	built-in	2023-06-09 18:19:06	2023-06-12 12:36:08	R & D	Physical		Edit Delete
sales	built-in	2023-06-09 01:27:19	2023-06-12 12:36:27	Sales	Physical		Edit Delete
marketing	built-in	2023-06-09 01:26:16	2023-06-12 12:36:32	Marketing	Physical		Edit Delete
hr	built-in	2023-06-09 01:25:46	2023-06-12 12:36:43	HR	Physical		Edit Delete
sales_and_marketing	built-in	2023-06-09 01:23:35	2023-06-12 12:36:57	Sales & Marketing	Physical		Edit Delete

9 in total < 1 > 10 / page

2. 在组织的列表页中单击 **群组** 按钮

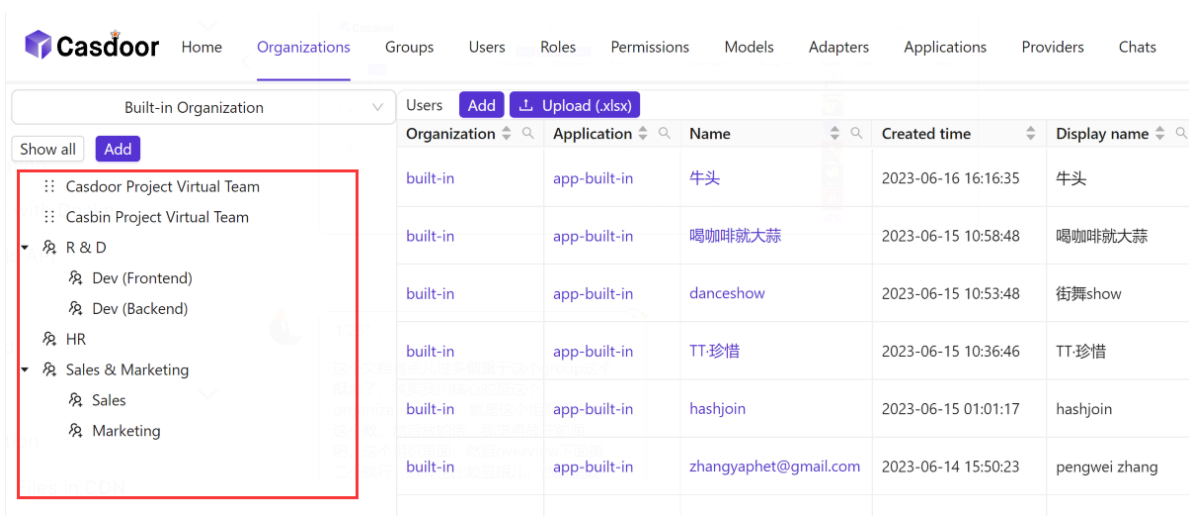
Casdoor Home **Organizations** Groups Users Roles Permissions Models Adapters Applications Providers Chats Messages ...

Organizations **Add**

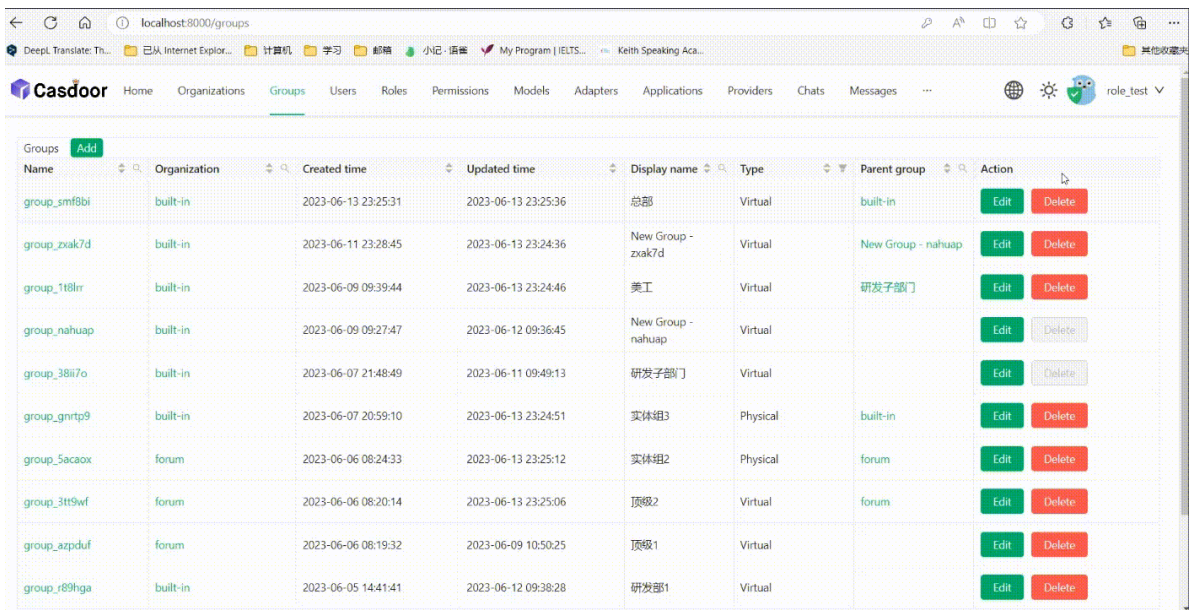
Name	Created time	Display name	Favicon	Website URL	Password type	Password salt	Default	Action
saas	2023-05-31 00:05:42	SaaS Users		https://saas.casbin.com	plain			Groups Users Edit Delete
gsoc	2021-02-11 23:26:20	GSOC Community		https://gsoc.com.cn	plain			Groups Users Edit Delete
casbin	2021-02-11 23:26:20	Casbin Organization		https://forum.casbin.com	plain			Groups Users Edit Delete
built-in	2021-02-10 00:37:06	Built-in Organization		https://door.casdoor.com	plain			Groups Users Edit Delete

4 in total < 1 > 10 / page

这将展示组织下群组的树形结构



这里有一个视频展示了如何管理群组



群组也可以在用户的资料页被编辑。

Title ⓘ : 1122

Homepage ⓘ :

Bio ⓘ :

Tag ⓘ : 222

Karma ⓘ : 333

Signup application ⓘ : app-built-in

Groups ⓘ : 🔗 Dev (Frontend) × :: Casdoor Project Virtual Team ×

Roles ⓘ :

Permissions ⓘ :

密码复杂度

Casdoor支持为每个组织的用户密码自定义密码复杂度选项。

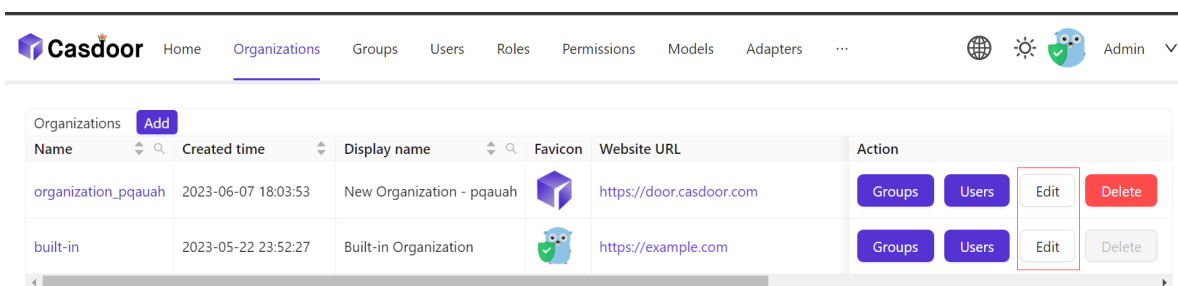
支持的复杂度选项

我们目前支持五个选项：

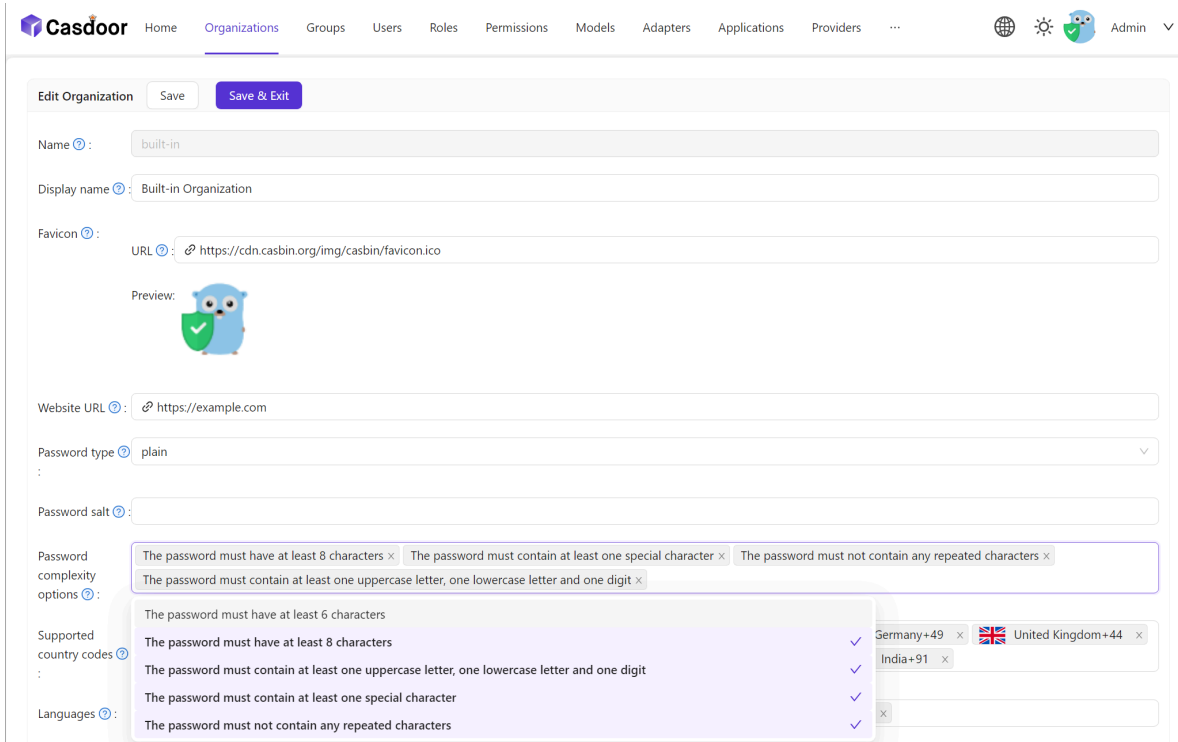
- **AtLeast6**：密码必须至少有六个字符。
- **AtLeast8**：密码必须至少有八个字符。
- **Aa123**：密码必须至少包含一个大写字母，一个小写字母和一个数字。
- **SpecialChar**：密码必须至少包含一个特殊字符。
- **NoRepeat**：密码不能包含任何重复的字符。

如果你想使用多个选项，你可以在组织编辑页面上选择它们：

1. 在组织列表页面上点击**编辑**按钮。



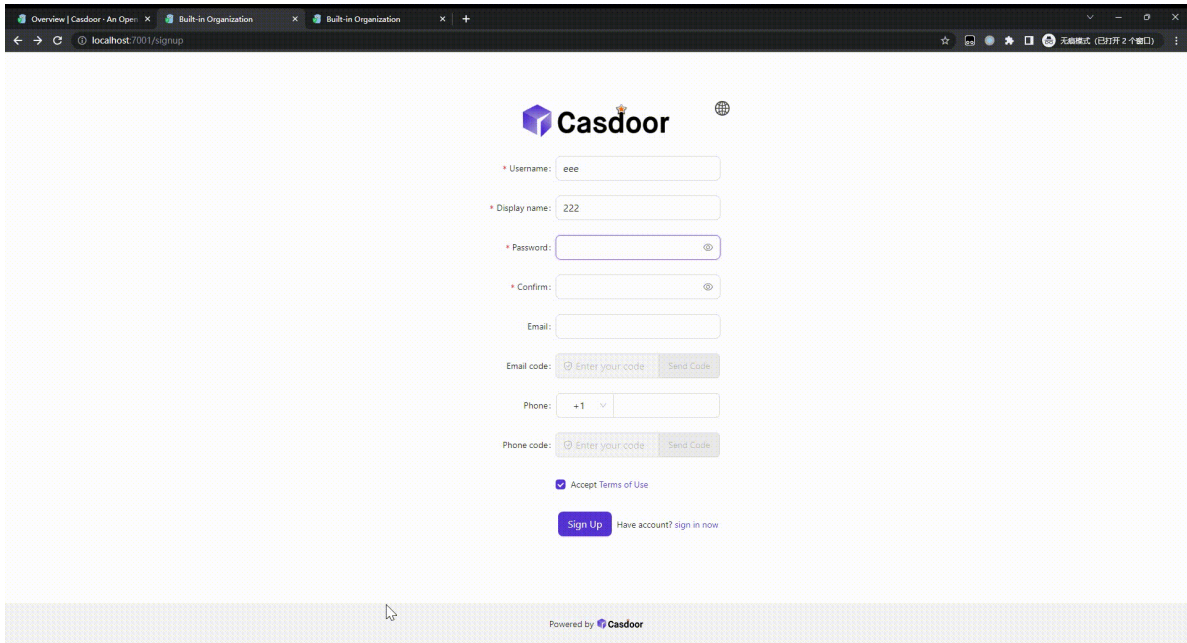
2. 然后在 **密码复杂度选项** 列中选择你需要的选项。



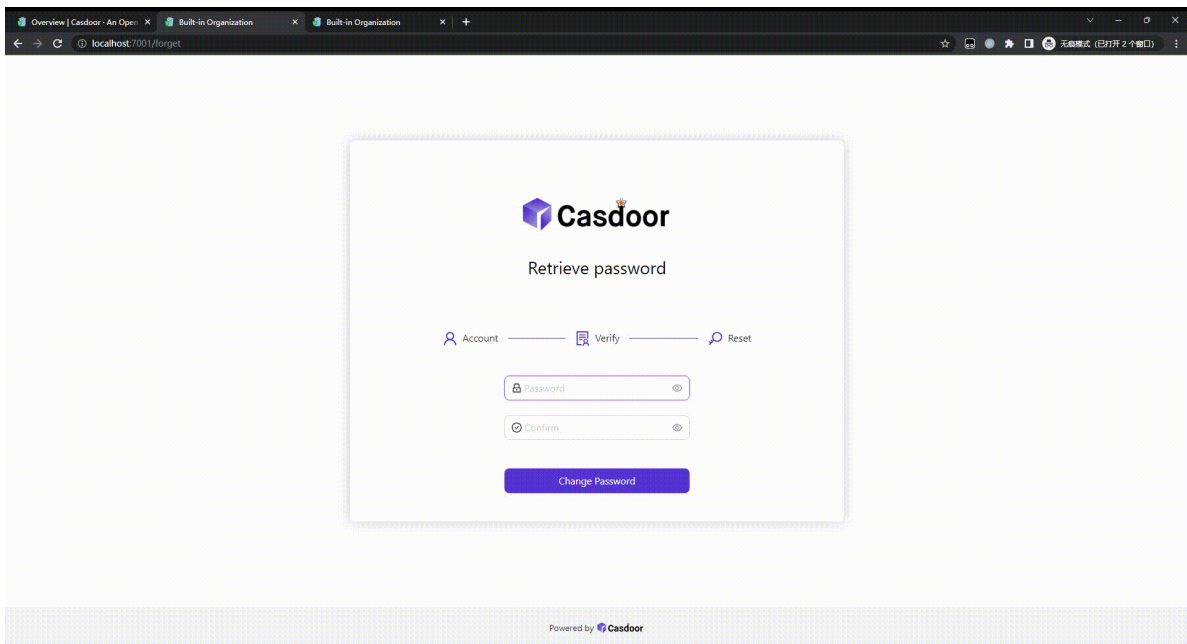
密码复杂度验证

我们在以下页面支持密码复杂度验证：

1. 注册页面。



2. 忘记密码页面。



3. 用户编辑页面。

Overview | Casdoor - An Open Source Identity and Access Management System

localhost:7001/users/built-in/admin

Casdoor Home Organizations Groups Users Roles Permissions Models Adapters Applications Providers Chats Messages Resources Records Plans Prings Subscriptions Admin


Edit User Save Save & Exit

Organization: built-in

ID: f9a29821-1a20-e39c-a7ab-5afa2d10e509

Name: admin

Display name: Admin

Avatar: Preview: 

Upload a photo...

User type: normal-user

Password: Modify password...

Email: admin@example.com [Reset Email...](#)

Phone: +1 12345678910 [Reset Phone...](#)

Country/Region: Please select country/region

Location:

Affiliation: Example Inc.

Title:

Password

New Password input password

Re-enter New input password

Cancel Set Password

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** 及其 **视图规则** 和 **修改规则**。

当您自定义一个组织中的账户项目时，此配置将在该组织所有成员的主页上生效。

如何定制？

帐户项目有四个属性：

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

要自定义账户项目，请遵循以下步骤：

1. 转到组织编辑页面。

2. 您将找到以下选项:

Account items [Add](#)

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	↑ ↓ 🗑️

3. Casdoor 提供简单的操作来配置帐户项目:

- 将项目设置为可见或不可见.

Account items [Add](#)

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	^ v 🗑️
<input type="checkbox"/>	Public		^ v 🗑️
<input checked="" type="checkbox"/>	Self	Admin	^ v 🗑️
<input checked="" type="checkbox"/>	Admin	Self	^ v 🗑️

有 3 条规则可用：

- **Public**：所有人都有权限。
- **Self**：用户有自己的权限。
- **Admin**：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，您可以查看 [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

自定义主题

Casdoor允许您定制主题以满足业务或品牌的UI多样性要求，包括主色和边框半径。

在Casdoor中，可以在全局、组织和应用程序级别定制主题。

1. 全局范围：这是Casdoor的默认主题，适用于选择遵循全局主题的任何组织。修改只能在Casdoor源代码中进行，不能在web UI中修改。
2. 组织范围：可以在组织编辑页面上定制组织的主题。此主题适用于组织内用户的所有Casdoor登录后页面，以及遵循组织主题的应用程序的入口页面（注册，登录，忘记密码等）。
3. 应用程序范围：可以在应用程序编辑页面上定制应用程序的主题。此主题适用于特定应用程序的入口页面（注册，登录，忘记密码等）。

定制组织主题

我们提供一个演示，演示如何为组织配置主题：

The screenshot displays a configuration page with several sections:

- Settings List:** A table of settings with columns for the setting name, a toggle switch, and two dropdown menus. The settings include Roles, Permissions, 3rd-party logins, Properties, Is admin, Is global admin, Is forbidden, Is deleted, WebAuthn credentials, and Managed accounts.
- Theme:** A section with a 'Theme' label and two buttons: 'Follow global theme' and 'Customize theme'.
- LDAPs:** A section with an 'LDAPs' label and an 'Add' button. Below it is a table with columns: Server Name, Server, Base DN, Auto Sync, Last Sync, and Action.

Server Name	Server	Base DN	Auto Sync	Last Sync	Action
Buildin LDAP Server	example.com:389	ou=Buildin,dc=example,dc=com	Disable		Sync Edit Delete

At the bottom of the configuration area are 'Save' and 'Save & Exit' buttons. The footer of the page reads 'Powered by Casdoor'.

! 信息

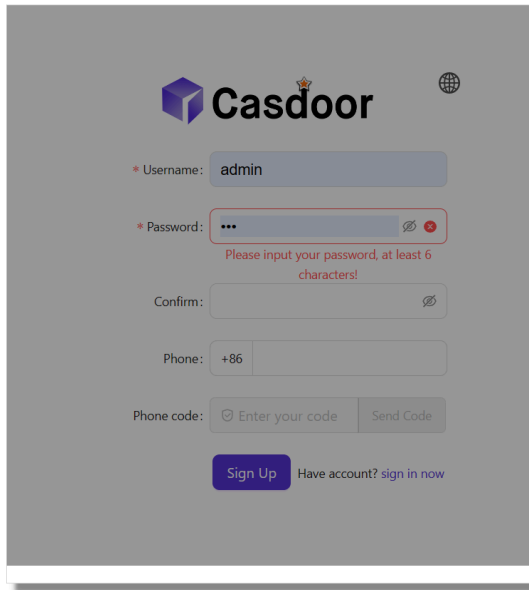
如果您的帐户组织与您正在编辑的组织相同，配置更改将立即生效，如上面的视频所示。但是，如果它们不同，您将需要登录到组织才能看到更改。

定制应用程序主题

应用程序可以使用与组织相同的主题编辑器定制主题。此外，您可以在预览面板中方便地预览主题。

Preview

Copy signup page URL

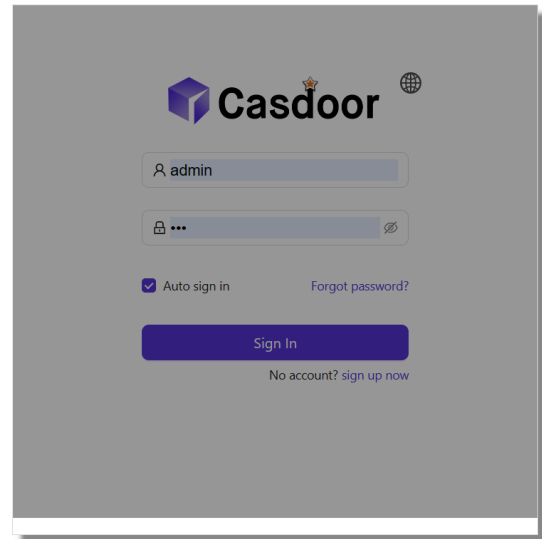


The image shows the Casdoor signup page. At the top left is the Casdoor logo, and at the top right is a globe icon. The form contains the following fields and elements:

- * Username:** A text input field containing the text "admin".
- * Password:** A password input field with three dots. Below it is a red error message: "Please input your password, at least 6 characters!".
- Confirm:** An empty password input field.
- Phone:** A text input field with "+86" in a small box to its left.
- Phone code:** A button labeled "Enter your code" and a button labeled "Send Code".
- Sign Up:** A blue button with the text "Sign Up". To its right is the text "Have account? sign in now".

Background URL

Copy signin page URL



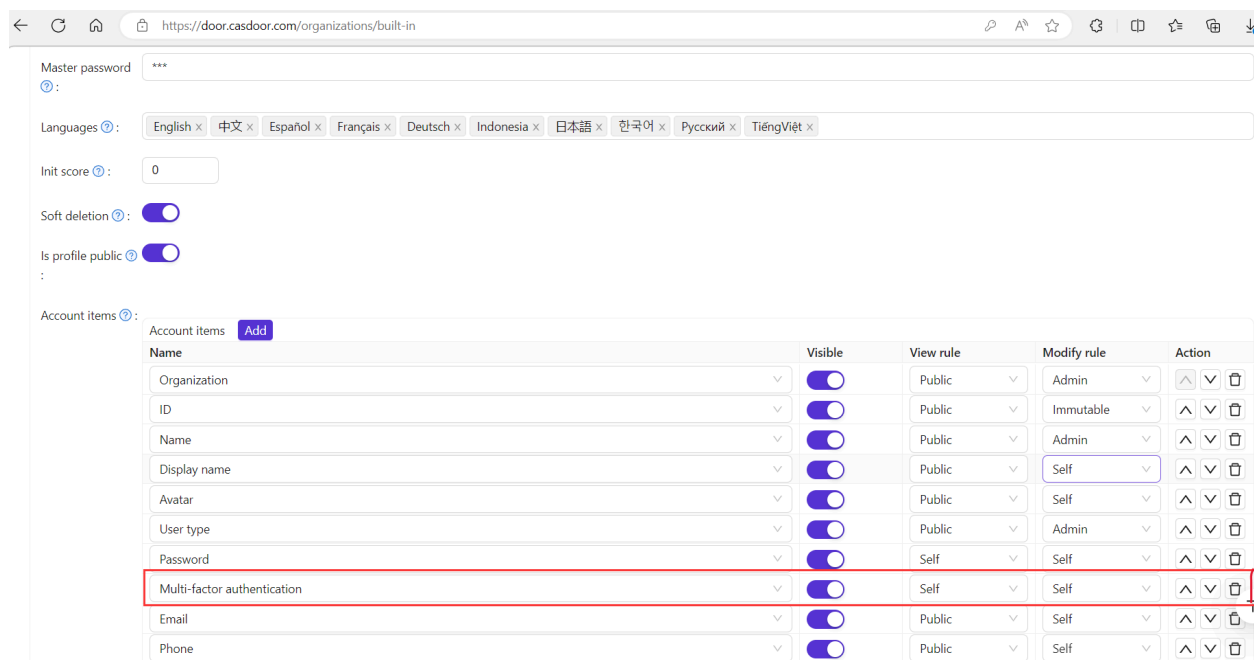
The image shows the Casdoor signin page. At the top left is the Casdoor logo, and at the top right is a globe icon. The form contains the following fields and elements:

- Username:** A text input field with a magnifying glass icon on the left, containing the text "admin".
- Password:** A password input field with three dots and a lock icon on the left.
- Auto sign in:** A checked checkbox with the text "Auto sign in".
- Forgot password?:** A link with the text "Forgot password?".
- Sign In:** A large blue button with the text "Sign In".
- No account? sign up now:** A link with the text "No account? sign up now".

管理多因素认证项目

在组织中添加多因素认证项目

在组织中，管理员可以将多因素认证项目添加到帐户设置中。这允许用户在他们自己的个人资料页面上配置多因素认证。



管理多因素认证项目

您可以管理多因素认证以确定哪些方法对用户可用。

管理多因素认证项目有两个规则：

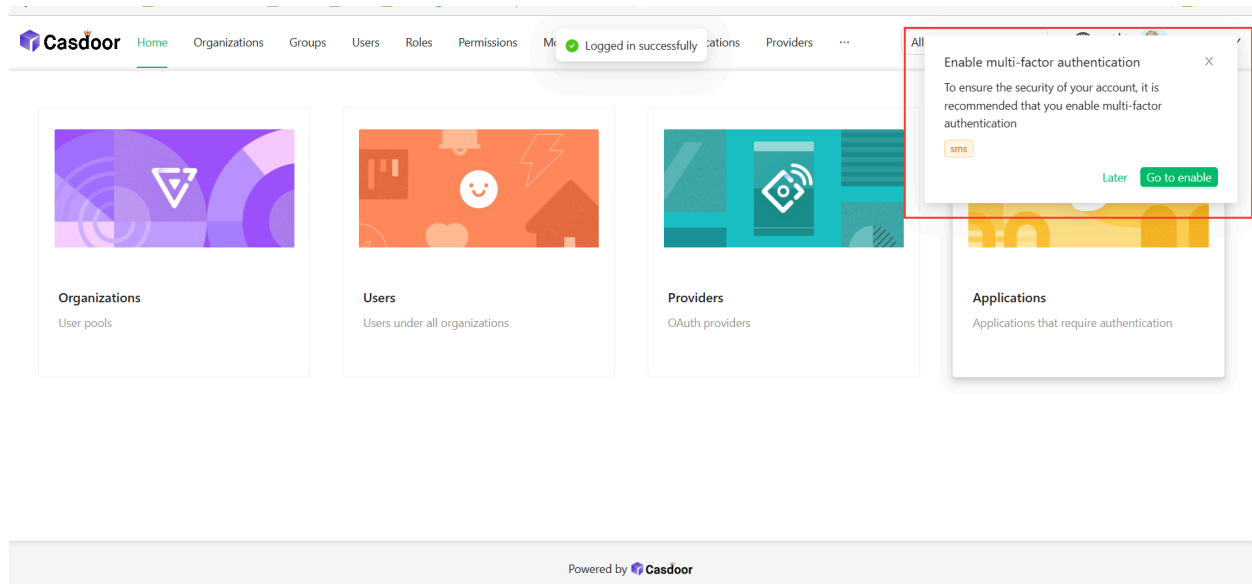
- 可选：用户可以选择是否启用这种类型的多因素认证。

- 提示：如果用户未启用此多因素认证模式，他们在登录Casdoor后将被提示启用它。
- 必需：用户必须启用此多因素认证方法。

MFA items ? : MFA items [Add](#)

Name	Rule	Action
Phone	Prompt	^ v 🗑️
Email	Optional	^ v 🗑️
App	Required	^ v 🗑️

下面的图片显示了提示用户启用多因素认证的通知。



此视频演示了当多因素认证方法设置为必需时，用户需要在完成登录过程之前启用多因素认证。

casbin 首页 API文档 注册 登录

更多新主题

Casnode / 帮助中心 浏览全部节点

Casbin = way to authorization
A place for Casbin-developers and users.
[现在注册](#)
已注册用户 登录

社区活跃度

注册会员	28
主题	0
回复	0

[财富排行榜](#)
[活跃排行榜](#)

我的收藏列表

今日热门主题

帮助中心

RSS



应用

应用

概览

Casdoor应用概述

术语参考

术语参考

应用程序配置

配置应用程序的身份验证

Providers

Configure different providers

登录方式

配置登录方式和登录方式的显示顺序

注册项目表

配置注册项目表以创建自定义注册页面

自定义登录界面

自定义您的应用程序登录页面

指定登录组织

在登录页面上指定登录组织

标签

配置您的应用程序标签

应用邀请码

使用邀请码限制用户通过应用注册

概览

Casdoor 中的每个应用都被称为“application”。它们之间没有关联，不会相互影响，这意味着只要你愿意，你可以单独部署或停止任何应用程序。

如果您想要使用Casdoor为您的网站提供登录服务，您可以将其添加为Casdoor应用。

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项等。

在本章中，您将学习如何从头开始启动您自己的应用程序。

让我们一起探索吧！

术语参考

- **Name**: 创建的应用的名称。
- **CreateTime**: 应用程序创建的时间。
- **DisplayName**: 应用程序向公众显示的名称。
- **Logo**: 应用程序的标志将会在登录和注册页面上显示。
- **HomepageUrl**: 应用程序主页的URL。
- **Description**: 描述应用程序。
- **Tags**: 只有在应用标签中列出标签的用户才能登录。
- **Organization**: 应用所属的组织。
- **EnableSignUp**: 如果用户可以注册。如果不是，应用程序的账户。
- **SignInMethods**: 登录方法的配置
- **SignupItems**: 用户注册时需要填写的字段。
- **Providers**: 为应用程序提供各种服务（如OAuth，电子邮件，短信服务）。
- **ClientId**: OAuth 客户端 ID。
- **ClientSecret**: OAuth客户端密钥。
- **RedirectUri**: 如果用户成功登录，Casdoor将导航到这些URI之一。
- **TokenFormat**: 生成的令牌的格式。它可以是以下格式：**JWT**（包含所有**User**字段），**JWT-Empty**（包含所有非空值）或**JWT-Custom**自定义访问令牌内的**User**字段。
- **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，请在此处填写URL。
- **ForgetUrl**: 与**SignupUrl**相同。
- **AffiliationUrl**:

应用程序配置

当你在你的服务器上部署你的casdoor并设置你的组织后，你可以现在就部署你的应用程序！

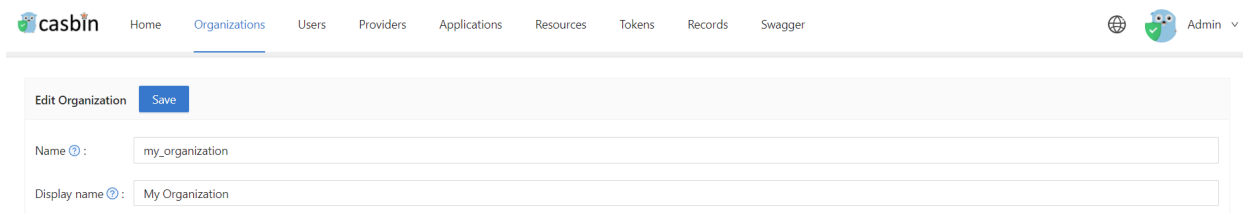
让我们看看如何使用Casdoor配置您的应用程序的身份验证！

📌 备注

例如，我想要使用 Casnode 设置我的论坛

我创建了我的应用程序并填写了必要的配置。

选择我创建的组织，以便组织中的用户也可以使用该应用程序。




这个组织命名为 `my_organization`，我从下拉菜单中选择到它。

Edit Application Save

Name ?:

Display name ?:

Logo ?:
URL:

Preview: 

Home ?:

Description ?:

Organization ?:

Client ID ?:

接下来，我希望我的用户在注册时能够使用 Casdoor 作为认证。因此，我在这里填写重定向URL为 <https://your-site-url.com/callback>。

:::警告

请注意，在提供商应用程序中的 `callback URL` 应该是Casdoor的回调 URL，而 Casdoor 中的 `Redirect URL` 应该是您网站的回调 URL。

进一步了解

为了使认证过程发挥作用，详细步骤如下：

1. 用户将请求发送到Casdoor。

2. 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。
3. 如果认证成功，GitHub 会调用回到 Casdoor，通知 Casdoor 已成功验证。因此，GitHub 认证回调 URL 应该是您的 Casdoor 的回调URL，即 <http://your-casdoor-url.com/callback>
4. 接着，Casdoor 将认证成功通知应用程序。这意味着 Casdoor 回调 URL 应该是您的应用程序的回调 URL，即 <http://your-site-url.com/callback>。

∴

You need to enable JavaScript to run this app.

:::提示

如果您想对应用程序的登录方式进行更个性化的配置，例如禁用某种登录方式或关闭某种登录方式，您可以参考[登录方式](#)

:::

Providers

You can also add third-party apps for sign up by adding providers and setting their properties.

Providers [Add](#)

Name	canSignUp	canSignIn	canUnlink	prompted	Action
provider_casbin_email					⬆️ ⬇️ 🗑️
provider_casbin_sms					⬆️ ⬇️ 🗑️
provider_storage_aliyun_oss					⬆️ ⬇️ 🗑️
provider_casdoor_github_localhost	🔵	🔵	🔵	⚪	⬆️ ⬇️ 🗑️
provider_casdoor_github	🔵	🔵	🔵	⚪	⬆️ ⬇️ 🗑️
provider_casdoor_google	⚪	🔵	🔵	⚪	⬆️ ⬇️ 🗑️
provider_casdoor_qq	⚪	🔵	🔵	⚪	⬆️ ⬇️ 🗑️
provider_casdoor_wechat	⚪	🔵	🔵	⚪	⬆️ ⬇️ 🗑️
provider_casdoor_facebook	⚪	🔵	🔵	⚪	⬆️ ⬇️ 🗑️
provider_casdoor_gitee	⚪	🔵	🔵	⚪	⬆️ ⬇️ 🗑️
provider_casdoor_gitlab	⚪	🔵	🔵	⚪	⬆️ ⬇️ 🗑️

Our provider can differentiate between different scenarios, and you can choose different providers for different functionalities by choosing rules. For a detailed explanation of each rule item, please refer to the table below.

Rule	Description
Signup	For the registration scenario, you can choose the "signup" rule for the provider to send the corresponding SMS or Email template.
Login	For the login scenario, you can choose the "login" rule for the provider.
Forget Password	When selecting a provider for the "Forget Password" scenario in your application, you can choose the "Forget Password" rule.

Rule	Description
Reset Password	When selecting a provider for the "Reset Password" scenario in your application, you can choose the "Reset Password" rule.
Set MFA	For MFA Setup Verification scenario, you can choose the "Set MFA" rule.
MFA Auth	For MFA Auth Verification scenario, you can choose the "MFA Auth" rule. For more information about mfa, you can refer to the MFA
all	If you want to use a single provider for all functionalities, you can choose the "all" rule. This means that the same provider will be used for all scenarios mentioned above in your application.

Providers ⓘ :

Name	Category	Type	Can signup	Can signin	Can unlink	Prompted	Signup group	Rule	Action
provider_captcha_default	Captcha								
provider_xz5v54	SMS							Signup	
provider_cltfc	SMS							All	

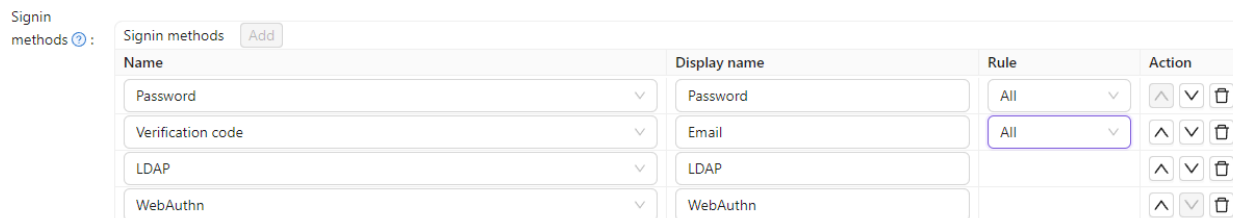
Preview ⓘ :

Copy signup page URL

Copy signin page URL

登录方式

在应用配置页面上，我们可以配置登录项目表。我们可以在表中添加和删除登录项目。



关于每个登录项目的详细解释，请参考下面的表格。目前，只有 **密码**、**验证码**、**WebAuthn** 和 **LDAP** 登录方式可用。

列名	可选值	描述
名称	-	登录方式的名称。
DisplayName	*	登录方式向公众显示的名称。
规则	规则项	选择一个规则来自定义这个登录方式。详细规则在下表中描述。
操作	-	用户可以执行如移动此登录方式向上、向下或删除等操作。

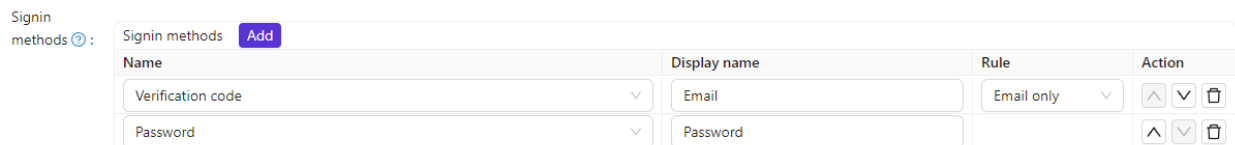
目前，只有 **密码** 和 **验证码** 登录方式支持配置规则。

登录方式名称	可选规则	描述
密码	全部(默认) / 非LDAP	选择用户可用的登录方式。选择全部，那么LDAP用户也可以登录。选择非LDAP，那么LDAP用户被禁止登录。
验证码	全部(默认) / 仅限电子邮件 / 仅限电话	选择用户可用的登录方式。选择全部，那么电子邮件和电话号码都可以用于登录验证。选择仅限电子邮件，那么只允许电子邮件登录。选择仅限电话，那么只允许电话号码进行登录验证。

i 备注

例如，我们希望用户优先使用他们的电子邮件登录，如果他们不能使用电子邮件，那么考虑使用密码登录。

首先，我们配置两个登录选项，验证码和密码，并且验证码是第一个登录选项。然后将验证码规则更改为仅限电子邮件，这样用户只能通过电子邮件接收登录验证码。




为了让用户更容易理解，我们可以更改验证码登录方式的显示名称，这样用户就可以轻松理解这是电子邮件登录。



Email Password

 Email

 Enter your code

Send Code

Auto sign in

[Forgot password?](#)

Sign In

No account? [sign up now](#)



 提示

除LDAP外，所有登录选项默认都是启用的。并且要求至少添加一个登录方式。

这是一个登录方式如何工作的视频：

注册项目表

在应用配置页面上，我们可以配置注册项目表以创建自定义注册页面。我们可以在此注册项目表上添加或删除任何注册项目。

Signup items [?](#) Add

Name	Visible	Required	Prompted	Rule	Action
ID	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Random	↑ ↓ 🗑️
Username	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		↑ ↓ 🗑️
Display name	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	None	↑ ↓ 🗑️
Password	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		↑ ↓ 🗑️
Confirm	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		↑ ↓ 🗑️
Email	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Normal	↑ ↓ 🗑️
Phone	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		↑ ↓ 🗑️
Agreement	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	None	↑ ↓ 🗑️

关于每个注册项目的详细解释，请参考下表。

列名	可选值	描述
名称	-	注册项目的名称。
可见	True / False	选择此注册项目是否在注册页面上可见。
必填	True / False	选择此注册项目是否为必填项。
提示	True / False	选择是否为用户忘记填写此注册项目时提示。

列名	可选值	描述
规则	Rule Items	选择一个规则来自定义此注册项目。详细规则在下表中描述。
操作	-	用户可以执行如移动此注册项目上移、下移或删除等操作。

目前，支持配置规则的注册项目包括 ID、显示名称、电子邮件和协议。

项目名称	可选规则	描述
ID	Random / Incremental	选择用户ID应该是随机生成还是递增。
显示名称	None / Real name / First, last	选择显示名称应该如何呈现。选择无将显示显示名称。选择真实姓名将显示用户的实际姓名。选择名, 姓将分别显示名和姓。
电子邮件	Normal / No verification	选择是否使用验证码验证电子邮件地址。选择正常将需要电子邮件验证。选择无验证将允许无需电子邮件验证即可注册。

项目名称	可选规则	描述
协议	None / Signin / Signin (Default True)	选择用户在登录时是否需要确认使用条款。选择无将不显示任何使用条款，允许用户直接登录。选择登录将要求用户在登录前确认条款。选择登录（默认为真）将默认设置条款为已确认，允许用户直接登录。

i 备注

例如，假设我想设置我的注册页面包括一个电子邮件字段，但不需要电子邮件验证。

首先，我添加了一些注册所需的注册项目，如ID、用户名、密码和电子邮件。

Signup items +

Name	visible	required	prompted	rule	Action
ID	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Incremental	↑ ↓ 🗑
Username	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		↑ ↓ 🗑
Password	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		↑ ↓ 🗑
Email	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	No verification	↑ ↓ 🗑

然后，我选择了电子邮件行的规则项目为无验证。结果，生成的预览注册页面将达到预期效果。



* Username:

* Password:



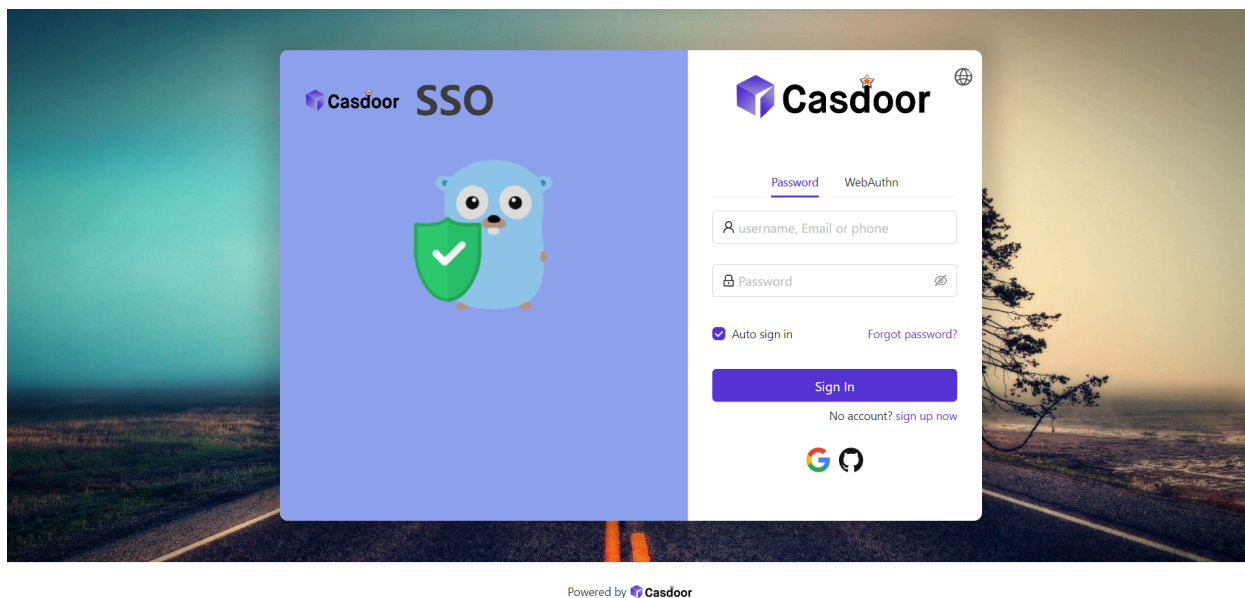
* Email:

Sign Up

Have account? [sign in](#)
now

自定义登录界面

您已创建应用程序。现在, 让我为你展示如何自定义应用的登录页面的UI. 在这个指南中, 我们将创建一个应用的自定义登录页面.



让我们开始吧

Part 1: 增加背景图片

首先, 我们要添加背景图像。默认背景是白色的, 看起来非常简单。



Password WebAuthn

username, Email or phone

Password

Auto sign in

[Forgot password?](#)

Sign In

[No account? sign up now](#)



Powered by Casdoor

若要添加背景图像，请使用您喜欢的图像的 URL，填写 **背景 URL**。如果您填写了有效的URL，预览区域将会显示您所选择的图像。

Background URL

[?](#):

URL [?](#):

[🔗](#) |

Preview:

Form CSS [?](#):

Form position [?](#):

Left

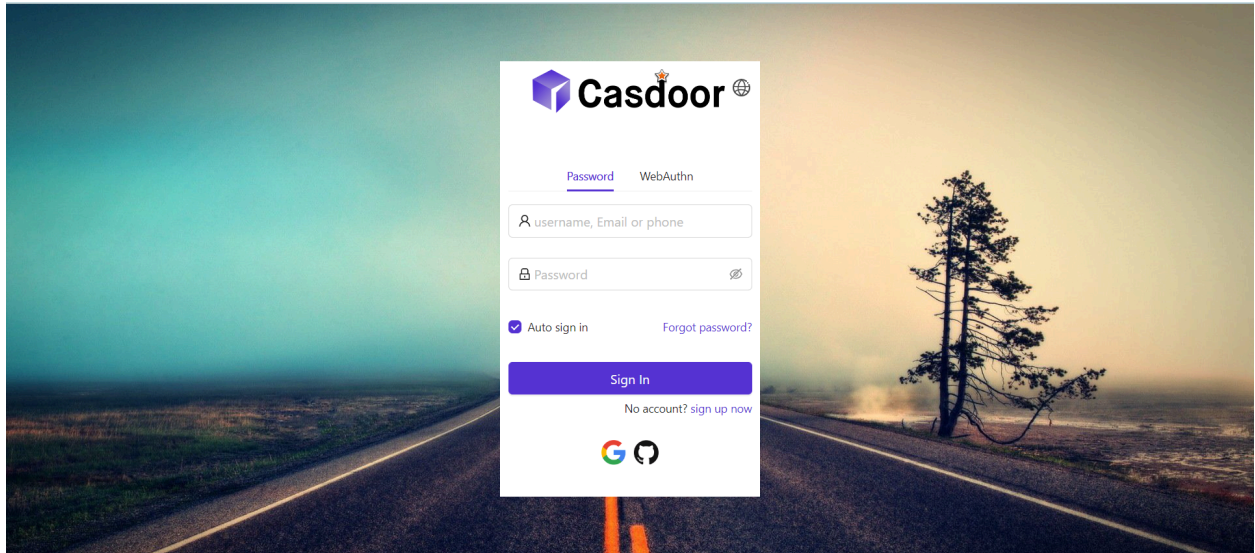
Center

Right

Enable side panel

Part 2: 自定义登录面板

现在我们在第一步操作结束的面板：



Powered by  Casdoor

为了让面板看起来很好，您需要添加一些CSS 代码。复制下面的代码并粘贴到 `表单 CSS` 字段。

```
<style>
.login-panel{
  padding: 40px 30px 0 30px;
  border-radius: 10px;
  background-color: #ffffff;
  box-shadow: 0 0 30px 20px rgba(0, 0, 0, 0.20);
}
</style>
```


Background URL

🔗 :

URL 🔗 :

<https://static.runoob.com/images/demo/demo2.jpg>

Preview:



Form CSS 🔗 :

Form position 🔗 :

Left

Center

Right

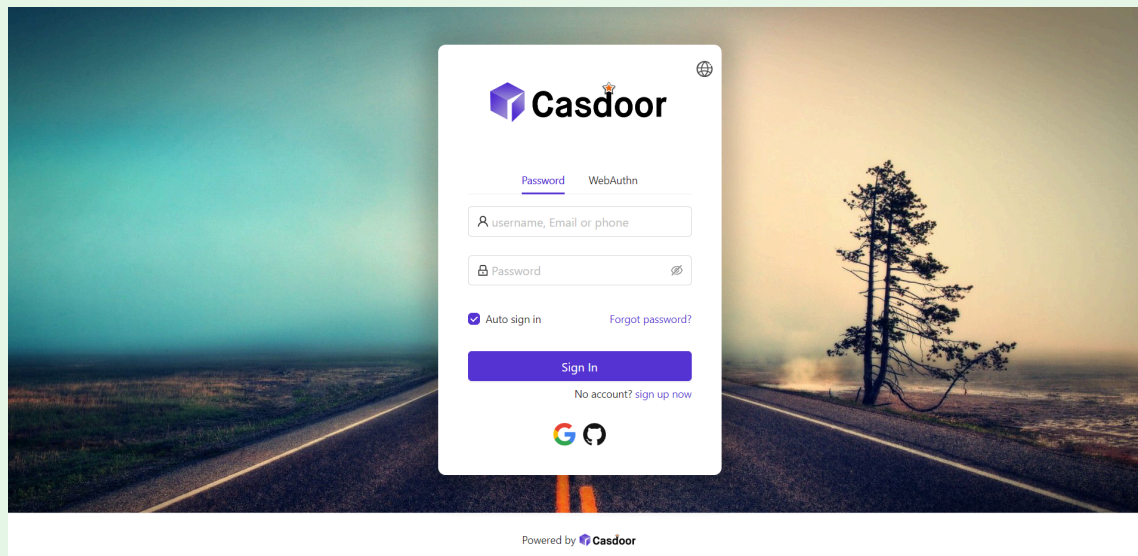
Enable side panel

💡 提示

当您编辑 **form CSS** 时，如果值为空，编辑器将显示默认值。然而，您仍然需要复制内容并将其粘贴到字段。

⋮


填写 **表单 CSS** 后，不要忘记在底部保存配置。好，让我们看看效果。



Part 3: 选择面板位置

现在登录页面就比刚开始的页面更为美观。我们还为您提供三个按钮，可以决定面板的位置。

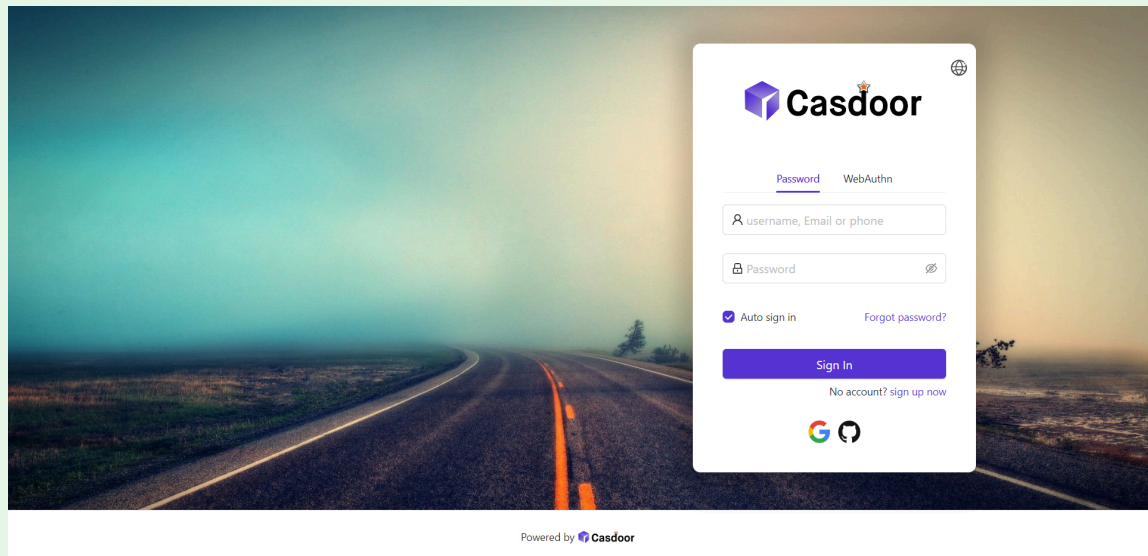
Background URL [?](#) : URL [?](#) :

Preview: 

Form CSS [?](#) :

Form position [?](#) :

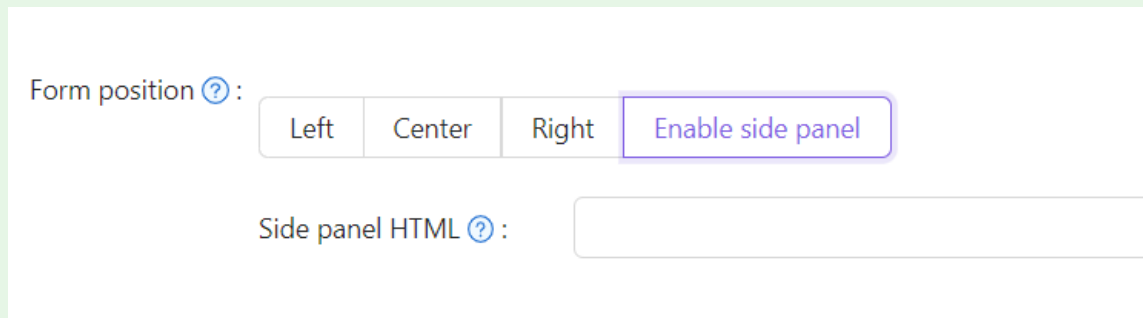
例如，选择 **Right** 按钮：



Part 4: 启用侧边面板

接下来，让我们看看如何启用侧面板并自定义其样式。

首先选择按钮。在 **启用侧面板** 模式时，面板会在中间位置。



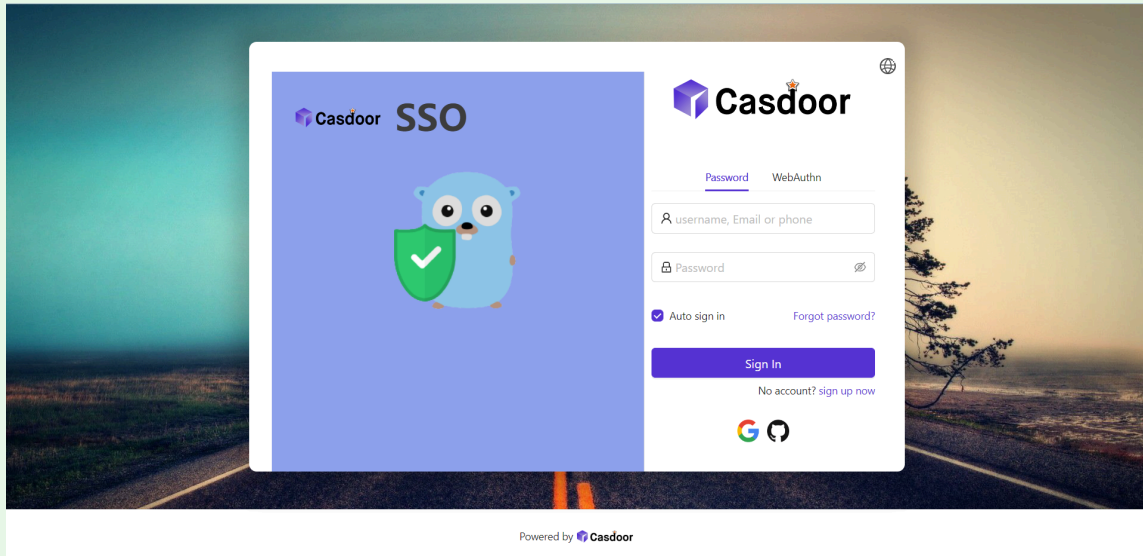
The screenshot shows a configuration interface for a side panel. It features a label "Form position ?:" followed by four buttons: "Left", "Center", "Right", and "Enable side panel". The "Enable side panel" button is highlighted with a purple border. Below this is another label "Side panel HTML ?:" followed by a large, empty text input field.

然后编辑 **侧面板 HTML**，它决定了将显示在侧面板中的内容。我们提供了一个默认模板，所以您可以简单地复制并粘贴它。



```
<style>
  .left-model{
    text-align: center;
    padding: 30px;
    background-color: #8ca0ed;
    position: absolute;
    transform: none;
    width: 100%;
    height: 100%;
  }
  .side-logo{
    display: flex;
    align-items: center;
  }
  .side-logo span {
    font-family: Montserrat, sans-serif;
    font-weight: 900;
  }


```


好，让我们看看效果。显示带有标志和图像的侧面板，但结果不能令人满意。





为了改进外观，您需要在 **表单 CSS** 中修改和添加一些CSS。


Background URL : URL :

Preview: 

Form CSS :

Form position :

Side panel HTML :

Signup items :

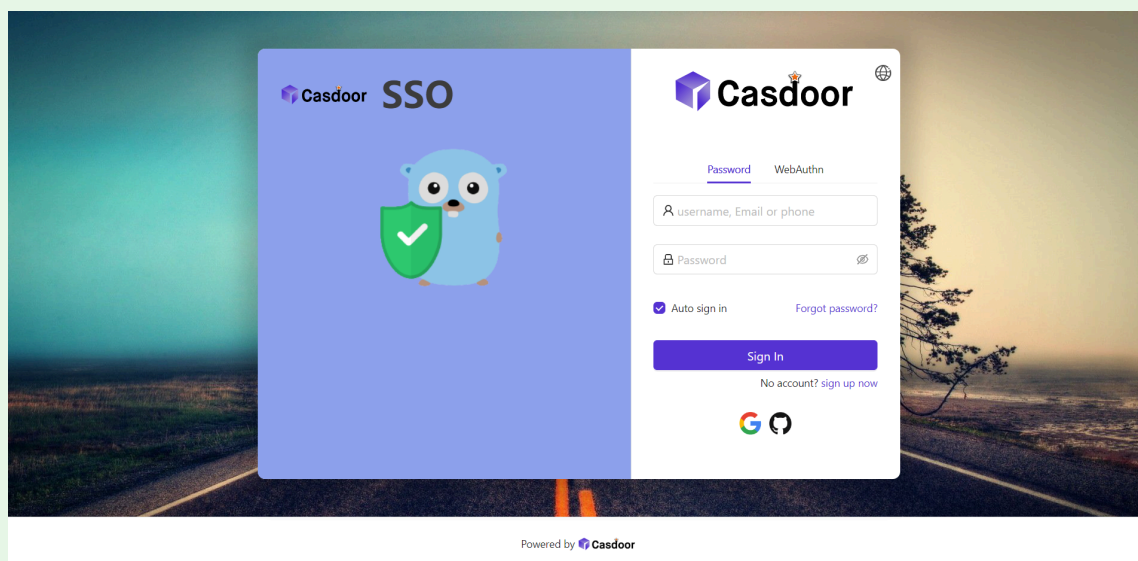
最后代码如下。

```
<style>
  .login-panel{
```

❗ 信息

`.login-panel` 和 `.login-form` 是div的类名。它们对应于页面的不同区域。如果你想要进一步自定义登录页面,你可以在这里写CSS代码,目标是这些类名称。

最后,我们有了一个美丽的登录页面!



总结

我们来总结一下: 我们已添加背景图像, 自定义了登录面板的风格, 并且启用了侧边板。

下面是一些关于定制Casdoor应用程序的额外资源:

- [自定义主题](#): 自定义主题, 包括主颜色和边框半径。
- [注册项目表](#)
- [应用程序配置](#)

感谢你的阅读!

指定登录组织

在这里，我们将向您展示如何启用为应用指定登录组织的选项。

例如，端点 `/login` 是属于**内置**组织的账户的默认登录页面。然而，您可以在属于**内置**组织的**app-built-in**应用上启用指定登录组织的选项。这允许用户在登录时选择一个组织。用户选择组织后，他们将被重定向到 `/login/<organization>`。

配置

在应用编辑页面上，您可以找到 `Org select mode` 配置选项。您可以从下拉列表中选择模式。

Org choice mode

Select

🔍 :

None

Signup URL 🔍 :

Select

Signin URL 🔍 :

Input

- 无：不会显示组织选择页面。
- 输入：用户可以在输入框中输入组织名称。
- 选择：用户可以从下拉列表中选择组织。



Please type an organization to sign in

A white rectangular input field with a thin green border. The text "built-in" is entered into the field, followed by a vertical cursor line.

Confirm



Please select an organization to sign
in

- built-in
- forum
- test
- Star

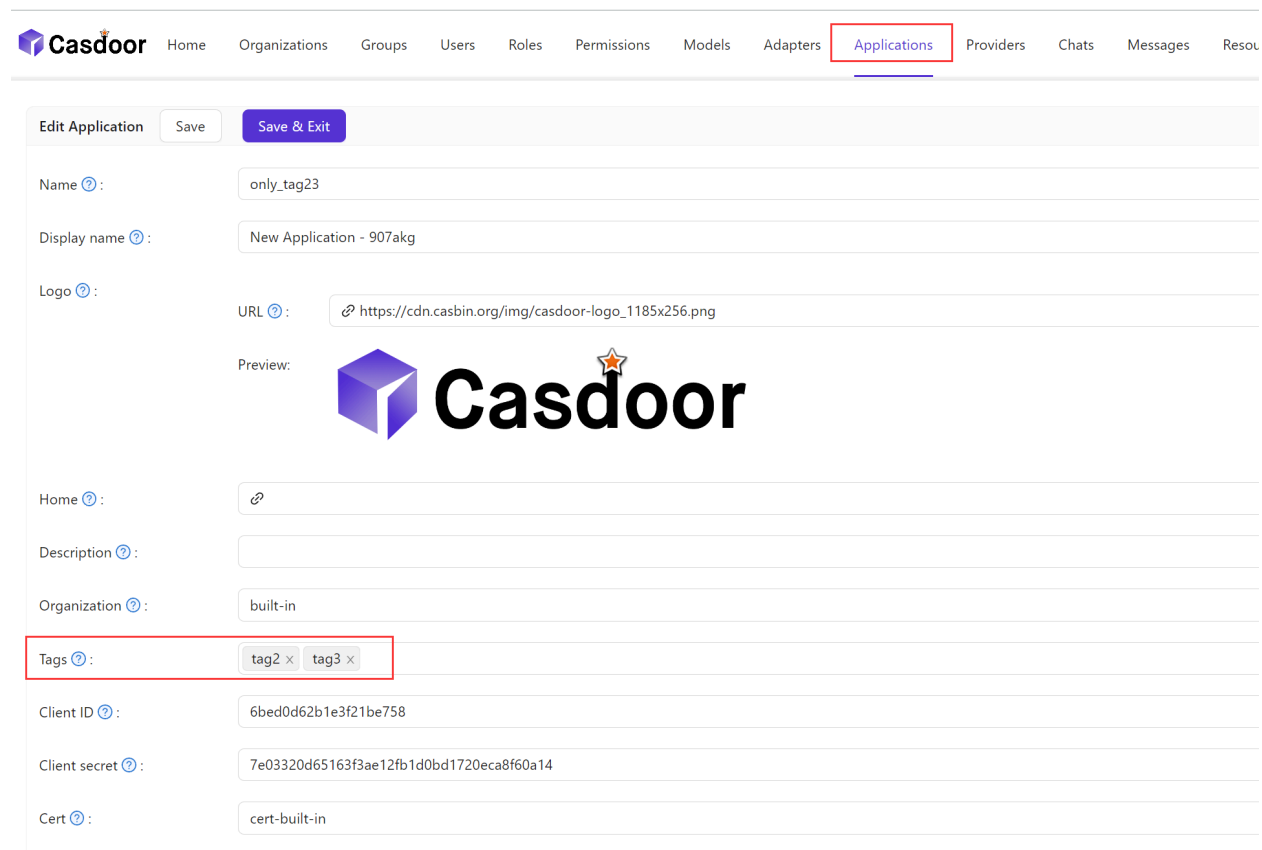
ⓘ 信息

The organization select page will only be shown when the route is `/login` or `<organization>/login`. 这意味着应用应该在组织或app-built-in中设置为默认应用。

标签

应用程序标签用于限制用户访问应用程序。具体来说，只有在应用程序标签中列出的标签的用户才被允许登录。例如，应用程序 `dev_app` 有标签 `dev, prd`。只有带有标签 `dev` 或 `prd` 的用户才能登录 `dev_app`。请注意，管理员和全局管理员用户不受应用程序标签的影响。

在应用程序编辑页面上，您可以找到 `Tags` 配置部分，您可以在那里添加标签。



The screenshot shows the 'Edit Application' interface in Casdoor. The 'Applications' menu item in the top navigation bar is highlighted with a red box. In the form, the 'Tags' field is also highlighted with a red box and contains two tags: 'tag2' and 'tag3', each with a close button (x). Other fields include Name (only_tag23), Display name (New Application - 907akg), Logo (URL: https://cdn.casbin.org/img/casdoor-logo_1185x256.png), Home (empty), Description (empty), Organization (built-in), Client ID (6bed0d62b1e3f21be758), Client secret (7e03320d65163f3ae12fb1d0bd1720eca8f60a14), and Cert (cert-built-in). A preview of the Casdoor logo is shown below the logo field.

这是一个演示应用程序标签如何工作的视频：

应用邀请码

简介

如果你想要限制用户通过应用注册，你可以使用邀请码。邀请码是能够用来允许用户通过应用注册的字符串。它由管理员生成并可重复使用。一个应用可以有多个邀请码。

配置

1. 首先，将“邀请码”添加到应用的注册项中。
2. 然后在应用的设置界面添加邀请码。

Signup items ⓘ: Add

Name	Visible	Required	Prompted	Rule	Action
ID	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Random	⬆️ ⬇️ 🗑️
Username	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	None	⬆️ ⬇️ 🗑️
Display name	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	None	⬆️ ⬇️ 🗑️
Password	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Normal	⬆️ ⬇️ 🗑️
ID card	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	None	⬆️ ⬇️ 🗑️
Email	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	None	⬆️ ⬇️ 🗑️
Phone	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	None	⬆️ ⬇️ 🗑️
Affiliation	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	None	⬆️ ⬇️ 🗑️
Country/Region	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	None	⬆️ ⬇️ 🗑️
Invitation code	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	None	⬆️ ⬇️ 🗑️

Invitation code ⓘ: Add

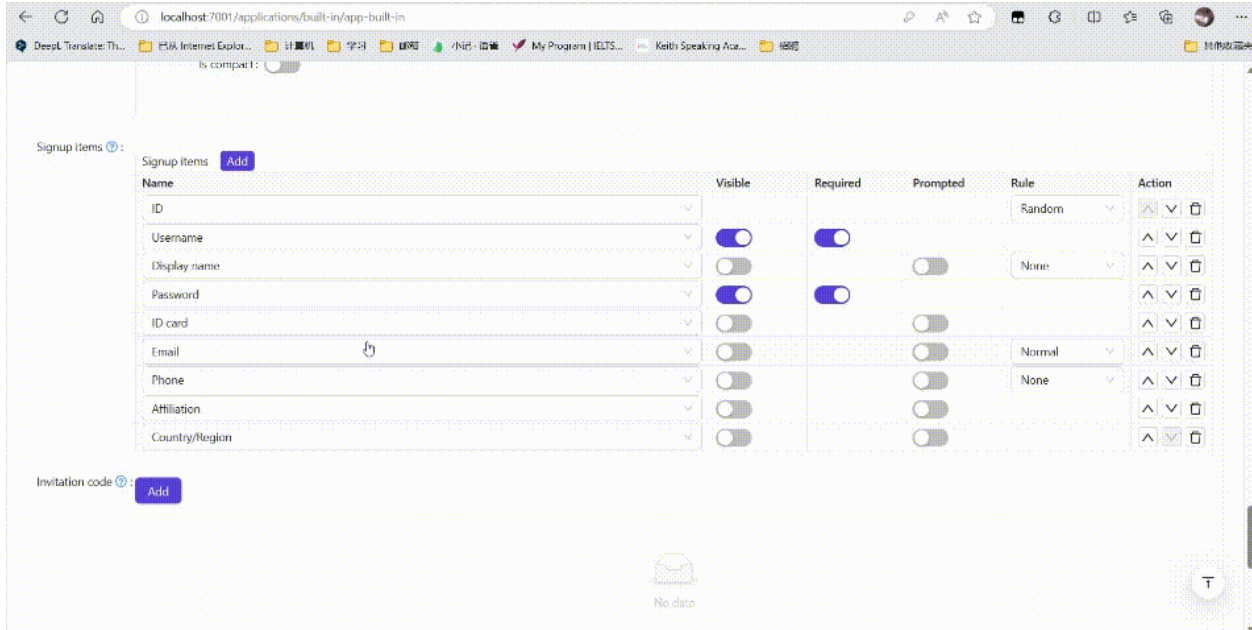
Copy Delete

💡 提示

Once the application has invitation codes, users can only sign up for the application with a valid invitation code. 无论“邀请码”在注册项中是否可见，用户都必须在注册时提供邀请码。因此，如果您想要使用邀请码，您需要将“邀请

码"添加到注册项表中。

以下是一个演示如何配置和使用邀请码的视频：





权限

权限

概述

在组织中使用Casbin管理用户访问权限

权限配置

使用公开的Casbin API来管理组织中用户的访问权限

开放的 Casbin API

使用公开的Casbin API来管理组织中的用户访问权限

适配器

配置适配器并对策略进行基本的增删改查操作

概述

简介

与单个Casdoor组织关联的所有用户共享对该组织的应用程序的访问权限。然而，可能会有一些情况，你希望限制用户访问某些应用程序或应用程序内的特定资源。在这种情况下，您可以利用Casbin提供的Permission功能。

在深入探讨这个话题之前，重要的是要对Casbin的工作方式及其相关概念，如模型，策略和适配器有基本的理解。简而言之，模型定义了您的权限策略的结构以及将请求与这些策略及其结果进行匹配的标准。另一方面，策略描述了具体的权限规则。一旦Casbin拥有了必要的模型和策略信息，它就可以对传入的请求执行权限控制。作为一个抽象层，适配器将Casbin的执行器与策略源隔离开来，允许将策略存储在各种位置，如文件或数据库中。

回到Casdoor的权限配置主题，您可以在Casdoor Web UI中的Model配置项中为您的组织添加一个模型，以及在Permission配置项中为您的组织添加一个策略。Casbin在线编辑器可以为您提供针对您特定使用场景定制模型和策略文件。您可以轻松地通过其Web UI将Model文件导入到Casdoor中，供内置的Casbin使用。然而，对于策略（即Casdoor Web UI中的Permission配置项），需要进一步的说明，这将在后面进行讨论。

就像你的应用程序需要通过Casdoor内置的Casbin来执行权限控制一样，Casdoor本身也利用自己的模型和策略通过Casbin来规范API接口的访问权限。尽管Casdoor可以从内部代码调用Casbin，但外部应用程序无法这样做。作为一种解决方案，Casdoor为外部应用程序提供了一个API，以调用内置的Casbin。我们将很快提供这些API接口的定义以及如何使用它们的说明。

在本章的最后部分，我们将展示一个实际的例子，来演示Casdoor如何与外部应用程序协

同工作进行权限控制。

让我们开始吧！

权限配置

让我们解释一下权限配置页面中的每一项。

- **Organization**: 该策略所属的组织名，一个组织可以拥有多个权限策略文件。
- **名称**: 权限策略在组织中的全球唯一名称。它用于识别策略文件。
- **显示名称**: 不重要。
- **模型**: 描述权限策略的结构和匹配模式的模型文件的名称。
- **适配器**: **注意!** 在当前版本中，此字段描述的是存储权限策略的数据库表的名称，而不是在Casdoor Web UI中配置的适配器菜单项的名称。Casdoor使用自己的数据库来存储配置的权限策略。如果此字段为空，权限策略将存储在 `permission_rule` 表中。否则，它将存储在指定的数据库表中。如果指定的表名在 Casdoor 所使用的数据库中不存在，将会自动创建。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。
- **子用户**: 权限策略将应用于哪些用户。
- **Sub roles**: 如果使用了RBAC模型，哪些角色将被应用该权限策略。这将为该角色中的每一个用户添加诸如 `g user role` 的权限策略。
- **子域**: 权限策略将应用于哪些域。
- **资源类型**: 在当前版本中，Casdoor不使用此字段为想要进行身份验证的外部应用程序。你可以暂且忽略它。
- **Resources**: 这个字段描述了你希望强制实施权限控制的资源。但请注意，这里的资源并非在 Casdoor Web UI 的 Resources 菜单项中所配置的资源。你可以在此处添加任何你所希望的字符串，例如一个 URL 或者一个文件名。
- **操作**: 此字段描述了对资源进行操作的动作。类似于资源，它可以是你想要的任何字符串，比如HTTP方法或其他自然语言。但是请注意，Casdoor 会将这些字符串全部转化成为小写再存储。另外，Casdoor 将会为每一个资源应用所有的动作。你不能指定一个动作只对某些资源生效。

- **Effect**: 这个选项对于 Casdoor 自身控制应用访问权限生效。如果你希望外部应用使用 Casdoor 所暴露的接口来强制实施权限控制，它不会起到任何作用。你应该在 Model 文件中描述模式匹配后的作用。

你可以看到，这个配置页面几乎就是为 **(sub, obj, act)** 模型量身定制的。

开放的 Casbin API

介绍

假设你的应用程序的前端已经获取了已登录用户的 `access_token`，现在想要对用户进行一些访问的认证。你不能简单地将 `access_token` 放入 HTTP 请求头来使用这些 API，因为 Casdoor 使用 `Authorization` 字段来检查访问权限。像 Casdoor 提供的其他 API 一样，`Authorization` 字段由应用客户端 id 和密钥组成，使用 [基本的 HTTP 认证方案](#)。它看起来像 `Authorization: Basic <Your_Application_ClientId><Your_Application_ClientSecret>`。因此，Casbin API 应当被应用的后端服务器调用。以下是如何操作的步骤。

以演示站点中的 [app-vue-python-example](#) 应用为例，授权头应该是：

```
Authorization: Basic 294b09fbc17f95daf2fe  
dd8982f7046ccba1bbd7851d5c1ece4e52bf039d。
```

1. 前端通过 HTTP 请求头将 `access_token` 传递给后端服务器。
2. 后端服务器从 `access_token` 中检索用户 id。

提前说明，这些接口也是为 `(sub, obj, act)` 模型设计的（目前）。URL 参数中的 `permissionId` 是应用权限策略的标识，由组织名称和权限策略名称组成（即 `组织名称/权限名称`）。正文是由 Casbin 模型定义的权限请求格式，通常分别代表 `sub`、`obj` 和 `act`。

除了请求强制执行权限控制的 API 接口以外，Casdoor 也提供了其它一些有助于外部应用获取权限策略信息的接口，也一并列在此处。

Enforce

请求:

```
curl --location --request POST 'http://localhost:8000/api/enforce?permissionId=example-org/example-permission' \  
--header 'Content-Type: application/json' \  
--header 'Authorization: Basic <Your_Application_ClientId>  
<Your_Application_ClientSecret>' \  
--data-raw '["example-org/example-user", "example-resource",  
"example-action"]'
```

响应:

```
{  
  "status": "ok",  
  "msg": "",  
  "sub": "",  
  "name": "",  
  "data": [  
    true  
  ],  
  "data2": null  
}
```

BatchEnforce

请求:

```
curl --location --request POST 'http://localhost:8000/api/batch-enforce?permissionId=example-org/example-permission' \  
--header 'Content-Type: application/json' \  

```

响应:

```
{
  "status": "ok",
  "msg": "",
  "sub": "",
  "name": "",
  "data": [
    [
      true,
      true,
      false
    ]
  ],
  "data2": null
}
```

GetAllObjects

请求:

```
curl --location --request GET 'http://localhost:8000/api/get-all-objects' \
--header 'Authorization: Basic <Your_Application_ClientId> <Your_Application_ClientSecret>'
```

响应:

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

GetAllActions

请求:

```
curl --location --request GET 'http://localhost:8000/api/get-all-actions' \
--header 'Authorization: Basic <Your_Application_ClientId> <Your_Application_ClientSecret>'
```

响应:

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

GetAllRoles

请求:

```
curl --location --request GET 'http://localhost:8000/api/get-all-roles' \
--header 'Authorization: Basic <Your_Application_ClientId> <Your_Application_ClientSecret>'
```

响应:

```
[
  "role_kcx661"
]
```


适配器

Casdoor支持使用UI连接适配器并管理策略规则。在Casbin中，策略规则的存储是作为一个适配器实现的，它充当Casbin的中间件。Casbin用户可以使用适配器从存储中加载策略规则或将策略规则保存到存储中。

适配器

- **类型**：适配器类型。目前支持数据库适配器。
- **主机**
- **端口**
- **用户**
- **密码**
- **数据库类型**：目前支持MySQL, PostgreSQL, SQL Server, Oracle, SQLite 3。
- **数据库**：数据库的名称。
- **表**：表的名称。如果表不存在，将会被创建。

Edit Adapter Save Save & Exit

Organization ⓘ:

Name ⓘ:

Type ⓘ:

Host ⓘ:

Port ⓘ:

User ⓘ:

Password ⓘ:

Database type ⓘ:

Database ⓘ:

Table ⓘ:

! 信息

填写完所有字段后，请记得**保存配置**。然后点击**同步**按钮来加载策略规则。策略规则将显示在下面的表格中。

Policies ⓘ: Sync Add

Rule Type	V0	V1	V2	V3	V4	V5	Option
p	built-in	*	*	*	*	*	
p	app	*	*	*	*	*	
p	*	*	POST	/api/signup	*	*	
p	*	*	POST	/api/get-email-and-phone	*	*	
p	*	*	POST	/api/login	*	*	
p	*	*	GET	/api/get-app-login	*	*	
p	*	*	POST	/api/logout	*	*	
p	*	*	GET	/api/logout	*	*	
p	*	*	GET	/api/get-account	*	*	
p	*	*	GET	/api/userinfo	*	*	



< 1 2 3 4 5 >





















Is enabled ⓘ:

基本的增删改查操作

如果您已成功连接适配器，您可以对策略规则进行基本的增删改查操作。

- 添加

Policies  



Rule Type	V0	V1	V2	V3	V4	V5	Option
p	built-in	*	*	*	*	*	 
p	*	*	POST	/api/signup	*	*	 
p	*	*	POST	/api/get-email-and-phone	*	*	 
p	*	*	POST	/api/login	*	*	 
p	*	*	GET	/api/get-app-login	*	*	 
p	*	*	POST	/api/logout	*	*	 
p	*	*	GET	/api/logout	*	*	 
p	*	*	GET	/api/get-account	*	*	 
p	*	*	GET	/api/userinfo	*	*	 
p	*	*	POST	/api/webhook	*	*	 


< 1 2 3 4 5 >



 提示





















您一次只能添加一条策略。新添加的策略将作为表格中的第一行出现，但实际上将保存在最后一行。所以，当你下次同步策略时，它们将出现在表格的最后一行。

- 编辑

Model  


Model  casbin_rule


Policies  



Rule Type	V0	V1	V2	V3	V4	V5	Option
p	built-in	*	POST	*	*	*	 
p	app	*	*	*	*	*	 
p	*	*	POST	/api/signup	*	*	 
p	*	*	POST	/api/get-email-and-phone	*	*	 
p	*	*	POST	/api/login	*	*	 
p	*	*	GET	/api/get-app-login	*	*	 
p	*	*	POST	/api/logout	*	*	 
p	*	*	GET	/api/logout	*	*	 
p	*	*	GET	/api/get-account	*	*	 
p	*	*	GET	/api/userinfo	*	*	 

< 1 2 3 4 5 >

- 删除

User :

Password :

Database type : 










Database :

Table :

Model : 

Policies :

Rule Type	V0	V1	V2	V3	V4	V5	Option
p	*	*	GET	/api/get-default-application	*	*	 
p	test	*	*	*	*	*	 

< 1 2 3 4 **5** >



提供商

提供商

概述

添加第三方服务到您的应用程序

OAuth

22 个项目

电子邮箱

5 个项目

短信

5 个项目

 **通知**

7 个项目

 **存储**

9 个项目

 **SAML**

4 个项目

 **支付**

5 个项目

 **验证码**

7 个项目

 Web3

2 个项目

概述

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

我们拥有的

目前，我们有六种类型的提供者：

- **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等许多其他OAuth应用程序添加到Casdoor。有关更多详细信息，请参阅[OAuth](#)部分。

- **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OSS服务存储文件。

- **支付提供商**

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，支持的支

付提供商包括支付宝，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，支持的验证码提供商包括默认验证码，reCAPTCHA，hCaptcha，阿里云验证码，以及Cloudflare Turnstile。

如何配置和使用

范围

提供者有不同的范围，这由创建者决定。只有管理员有权限添加和配置提供商。

Casdoor中有两种类型的管理员：






- **全球管理员**：所有在 `built-in` 组织下的用户以及启用 `IsGlobalAdmin` 的用户。由全局管理员创建的提供者可以被所有应用程序使用。
- **组织管理员**：启用 `IsAdmin` 的用户。由组织管理员创建的提供者**只能**被该组织下的应用程序使用（正在开发中...）。

添加到应用程序

按照以下步骤将提供者添加到您的应用程序中。请注意，除非您已添加它，否则您不能在应用程序中使用该提供者。






1. 转到应用程序编辑页面并添加一个新的提供商。

Providers ? :

Name	Category	Type
provider_storage_aliyun_oss	Storage	
provider_casdoor_github	OAuth	
provider_casdoor_google	OAuth	
provider_casdoor_qq	OAuth	
provider_casdoor_wechat	OAuth	
Please select a provider		

2. 选择您想要添加到应用程序的提供商。 您将看到应用程序可以使用的所有提供商。






Providers ? :

Name	Category	Type	canSignUp
provider_storage_aliyun_oss	Storage		<input type="checkbox"/>
provider_casdoor_github	OAuth		<input checked="" type="checkbox"/>
provider_casdoor_google	OAuth		<input checked="" type="checkbox"/>
provider_casdoor_qq	OAuth		<input checked="" type="checkbox"/>
provider_casdoor_wechat	OAuth		<input checked="" type="checkbox"/>
Please select a provider			

Preview ? :

provider_email_submail
provider_4olfdm
provider_casdoor_bilibili
provider_casdoor_okta
provider_casdoor_alipay
provider_casdoor_slack
provider_casdoor_steam
provider_casdoor_infocflow

3. 对于OAuth和Captcha提供商，您可以配置它们的使用方式。 请参阅 [OAuth](#) 和 [验证码](#) 以获取更多信息。

Type	canSignUp	canSignIn	canUnlink	prompted	Rule
					
					Always ▾
	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

最后，**保存配置**。您现在可以尝试在您的应用程序中使用该提供程序。

OAuth

概述

将 OAuth 提供商添加到您的应用程序

谷歌

将Google OAuth提供商添加到您的应用程序

Google One Tap

学习如何在您的应用程序中添加Google One Tap支持

GitHub

将GitHub OAuth提供商添加到您的应用程序

LinkedIn

将LinkedIn OAuth提供商添加到您的应用程序

Facebook

将Facebook OAuth提供商添加到您的应用程序中。

AD FS

将AD FS添加为第三方服务以完成身份验证。

Azure AD

将Azure AD添加为第三方服务以完成身份验证

Azure AD B2C

将 Azure AD B2C 添加为第三方服务以完成身份验证

自定义OAuth

将您的自定义OAuth提供商添加到Casdoor

Okta

将 Okta OAuth 提供商添加到您的应用程序

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

将微信 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提供商添加到您的应用程序中

Gitee

添加Gitee 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

 **Infoflow**

在应用程序中添加 Infoflow OAuth 提供商

































 **Lark**













































将Lark OAuth提供商添加到您的应用程序

概述

Casdoor允许使用其他OAuth应用程序作为登录方法。

目前，Casdoor支持多个OAuth应用程序提供商。一旦这些提供商被添加到Casdoor中，它们的图标将会在登录和注册页面上显示。以下是Casdoor支持的提供商：

提供者	Logo	提供者	Logo	提供者	Logo	提供者	Logo
ADFS		支付宝		亚马逊		苹果	
Auth0		Azure AD		Azure AD B2C		百度	
Bilibili		Bitbucket		盒子		Casdoor	
Cloud Foundry		Dailymotion		Deezer		DigitalOcean	
钉钉		Discord		Tiktok		Dropbox	
Eve Online		Facebook		Fitbit		Gitea	
Gitee		GitHub		GitLab		Google	
Heroku		InfluxCloud		信息流		Instagram	

提供者	Logo	提供者	Logo	提供者	Logo	提供者	Logo
Intercom		Kakao		Lark		Lastfm	
线		LinkedIn		Mailru		Meetup	
微软		Naver		Nextcloud		Okta	
OneDrive		Oura		Patreon		PayPal	
QQ		Salesforce		Shopify		Slack	
SoundCloud		Spotify		Steam		Strava	
Stripe		TikTok		Tumblr		Twitch	
Twitter		Typetalk		Uber		VK	
微信		WeCom		微博		WePay	
Xero		Yahoo		Yammer		Yandex	
Zoom		电子邮件		短信		Battle.net	

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. 前往您的Casdoor首页。
2. 点击顶部栏中的 **Providers**。
3. 点击 **Add**, 您将会看到新的提供商被添加到顶部的列表中。
4. 点击新的提供商以对其进行更改。
5. 在 **Category** 部分中, 选择 **OAuth**。
6. 从 **Type** 下拉菜单中选择您需要的特定OAuth提供商。
7. 填写必要的信息, 例如 **Client ID** 和 **Client Secret**。

应用程序设置


1. 点击顶部栏中的 **Application**, 然后选择要编辑的应用程序。
2. 点击提供商添加按钮, 并选择新添加的提供商。
3. 修改提供商的权限, 例如启用注册、登录和解绑。
4. 你已经准备好了!

谷歌

要设置Google OAuth提供商，请转到[Google API控制台](#)并使用您的Google帐户登录。

Project name *
My Casdoor ?

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

Location *
 No organization [BROWSE](#)

Parent organization or folder

[CREATE](#) [CANCEL](#)

接下来，导航到OAuth 同意屏幕选项卡以配置 OAuth 同意屏幕。

Google Cloud Platform My Casdoor Search products and resources

API APIs & Services OAuth consent screen

- Dashboard
- Library
- Credentials
- OAuth consent screen**
- Domain verification
- Page usage agreements

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

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

按照以下步骤注册您的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

之后，转到凭证选项卡。

Credentials

[+ CREATE CREDENTIALS](#)

 DELETE

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

API Keys



Name

Creation date ↓

No API keys to display

OAuth 2.0 Client IDs



Name

Creation date ↓

No OAuth clients to display

Service Accounts



Email

Name ↑

No service accounts to display

为您的应用创建凭证:

[←](#) 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 *



! 请确保您正确设置了授权重定向URI

在Google OAuth配置中，`Authorized redirect URIs`必须设置为**您的Casdoor的回调URL**，而在Casdoor中的`Redirect URL`应设置为**您的应用程序的回调URL**。

有关更多详细信息，请参阅[应用配置](#)。

创建Client ID后，您将获得`Client ID`和`Client Secret`。

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

487708653175-11oih9gfb2u3tvfp6684qaes5ujjdca.apps.gc

Your Client Secret

HbxoqxxkGSs11CVRuMTVvK57

DOWNLOAD JSON

OK

将Google OAuth提供商添加进来，并在你的Casdoor中输入 `Client ID` 和 `Client Secret`。

Edit Provider Save

Name ? :	<input type="text" value="my_google_privider"/>
Display name ? :	<input type="text" value="Google provider"/>
Category ? :	<input type="text" value="OAuth"/>
Type ? :	<input type="text" value="Google"/>
Client ID ? :	<input type="text" value="487708653175-11oih9gfb2u3tvfp6684qaes5ujjdca.apps.googleusercontent.com"/>
Client secret ? :	<input type="text" value="*****"/>
Provider URL ? :	<input type="text" value="https://console.cloud.google.com/apis/credentials/oauthclient/498643462012-46"/>

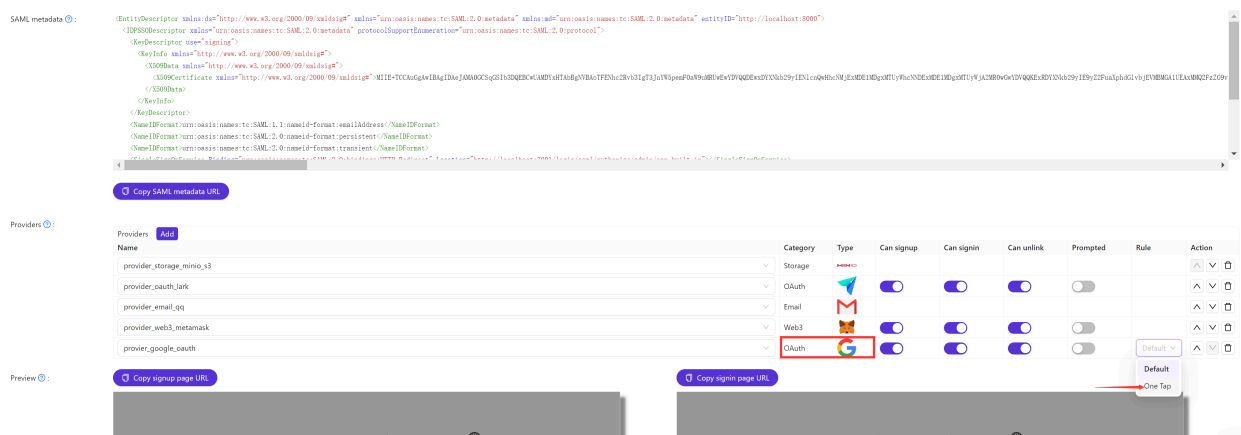
您现在可以使用Google作为第三方服务来完成身份验证。

Google One Tap

步骤1：配置您的应用程序

如果您想通过Google One Tap允许登录，您需要在您的应用程序中添加Google OAuth提供者。有关如何做到这一点的信息，请参考[Google的文档](#)。

一旦您添加了Google OAuth提供者，导航到应用程序编辑页面，添加Google OAuth提供者，并将Rule从Default切换到One Tap。



步骤2：使用Google One Tap登录

设置完成后，用户现在可以使用Google One Tap登录。

GitHub

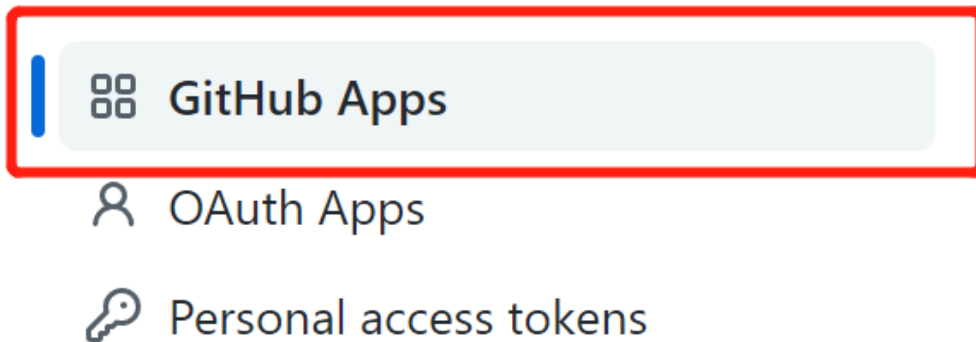
GitHub OAuth支持网页应用流程和设备流程。请继续阅读以获取OAuth凭据。

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

⚠️ 注意事项

小技巧：我们建议您使用GitHub Apps来替换OAuth Apps，因为GitHub Apps可以添加多个重定向URIs，这在部署测试和生产环境时可以带来便利。[GitHub](#)官方也推荐使用GitHub Apps而不是OAuth Apps。

Settings / Developer settings



然后填写GitHub应用名称、主页URL、描述和回调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.

Identifying and authorizing users

Add Callback URL

The full URL to redirect to after a user authorizes an installation.

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 App配置中，**Callback URL** 必须是**你的Casdoor的回调URL**，并且Casdoor中的**Redirect URL** 应该是**你的应用程序的回调URL**。

有关更多详细信息，请阅读[应用配置](#)。

在注册了您的GitHub应用程序后，您现在可以生成您的**Client Secret**了！

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](#)

 Client secret	****dba81954 Added 5 minutes ago by [redacted] Last used within the last week	Delete
 Client secret	****15822f89 Added on 15 Feb by [redacted] Last used within the last week	Delete

添加一个GitHub OAuth提供商，并在您的Casdoor中填写 `Client ID` 和 `Client Secret`。

Edit Provider

Name ⓘ:

Display name ⓘ:

Category ⓘ: ▼

Type ⓘ: ▼

Client ID ⓘ:


Client secret ⓘ:

Provider URL ⓘ:

现在您可以使用GitHub作为第三方服务来完成身份验证。

LinkedIn

要设置LinkedIn OAuth提供商，请转到[LinkedIn Developer](#)页面创建一个新的应用程序。

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

Create an app

* indicates required

App name*

LinkedIn Page*

ⓘ 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



在填写上述表格并创建您的应用程序后，您需要验证与应用程序关联的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](#)

备注

只有公司页面管理员才能验证您的应用并授予其权限。

一旦您的应用程序通过验证，您就可以继续：




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，`authorized redirect URLs`必须是你的Casdoor的回调URL，并且Casdoor中的`Redirect URL`应该是你的应用程序的回调URL。

有关更多详细信息，请阅读[应用配置部分](#)。

然后，您可以获取您的`Client ID`和`Client Secret`。


Application credentials

Authentication keys

Client ID:

860t47n8b4jh7w

Client Secret:

..... 

在Casdoor中添加一个LinkedIn OAuth提供商，并填写`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，请转到[Facebook Developer](#)网站并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 ✕

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







在输入您的姓名和联系电子邮件后，您将被带到Facebook开发者仪表板。

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 Read Docs Set Up	 Webhooks Subscribe to changes and receive updates, in real Read Docs Set Up	 Instant Games Create a cross-platform HTML5 game hosted on Read Docs Set Up

接下来，设置Facebook登录：



Facebook Login

The world's number one social login product.

[Read Docs](#) [Set Up](#)

为此应用选择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

在填写网站URL后，转到**Facebook登录 > 设置**，并输入有效的OAuth重定向URI。

Client OAuth Settings

<input checked="" type="checkbox"/> 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. [?]
<input checked="" type="checkbox"/> Yes	Web OAuth Login Enables web-based Client OAuth Login. [?]
<input type="checkbox"/> No	Force Web OAuth Reauthentication When on, prompts people to enter their Facebook password in order to log in on the web. [?]
<input checked="" type="checkbox"/> Yes	Use Strict Mode for Redirect URIs Only allow redirects that exactly match the Valid OAuth Redirect URIs. Strongly recommended. [?]
<input type="checkbox"/> Yes	Enforce HTTPS Enforce the use of HTTPS for Redirect URIs and the JavaScript SDK. Strongly recommended. [?]
<input type="checkbox"/> No	Embedded Browser OAuth Login Enable webview Redirect URIs for Client OAuth Login. [?]

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. [?]

❗ 正确设置授权的 REDIRECT URL

在Facebook OAuth配置中，**Valid OAuth Redirect URIs** 必须是**你的 Casdoor的回调URL**，并且Casdoor中的**Redirect URL** 应该是**你的应用程序的回调URL**。

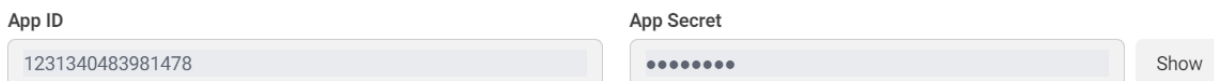
有关更多详细信息，请阅读[应用配置部分](#)。

基本的配置几乎完成了！

将顶部工具栏中的模式从**开发中**切换到**实时**。



现在您可以在Casdoor中使用您的App ID和App Secret。



添加Facebook OAuth提供商，并填写Client ID和Client Secret，用你的Casdoor中的App ID和App Secret。

修改提供商 **保存**

名称 ? : my_facebook_provider

显示名称 ? : Facebook provider

分类 ? : OAuth

类型 ? : Facebook

Client ID ? : 1231340483981478

Client secret ? : *****

您现在可以使用Facebook作为第三方服务进行身份验证！

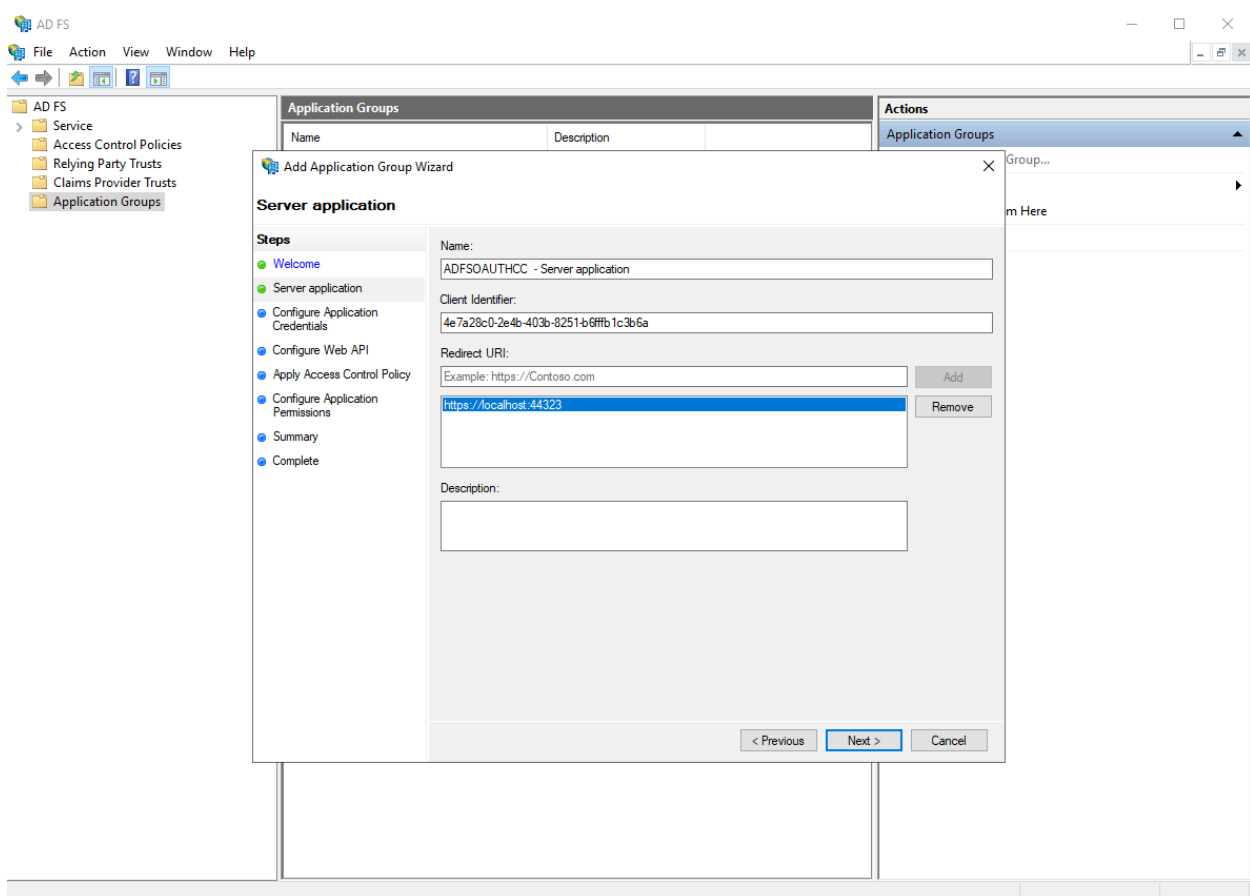
AD FS

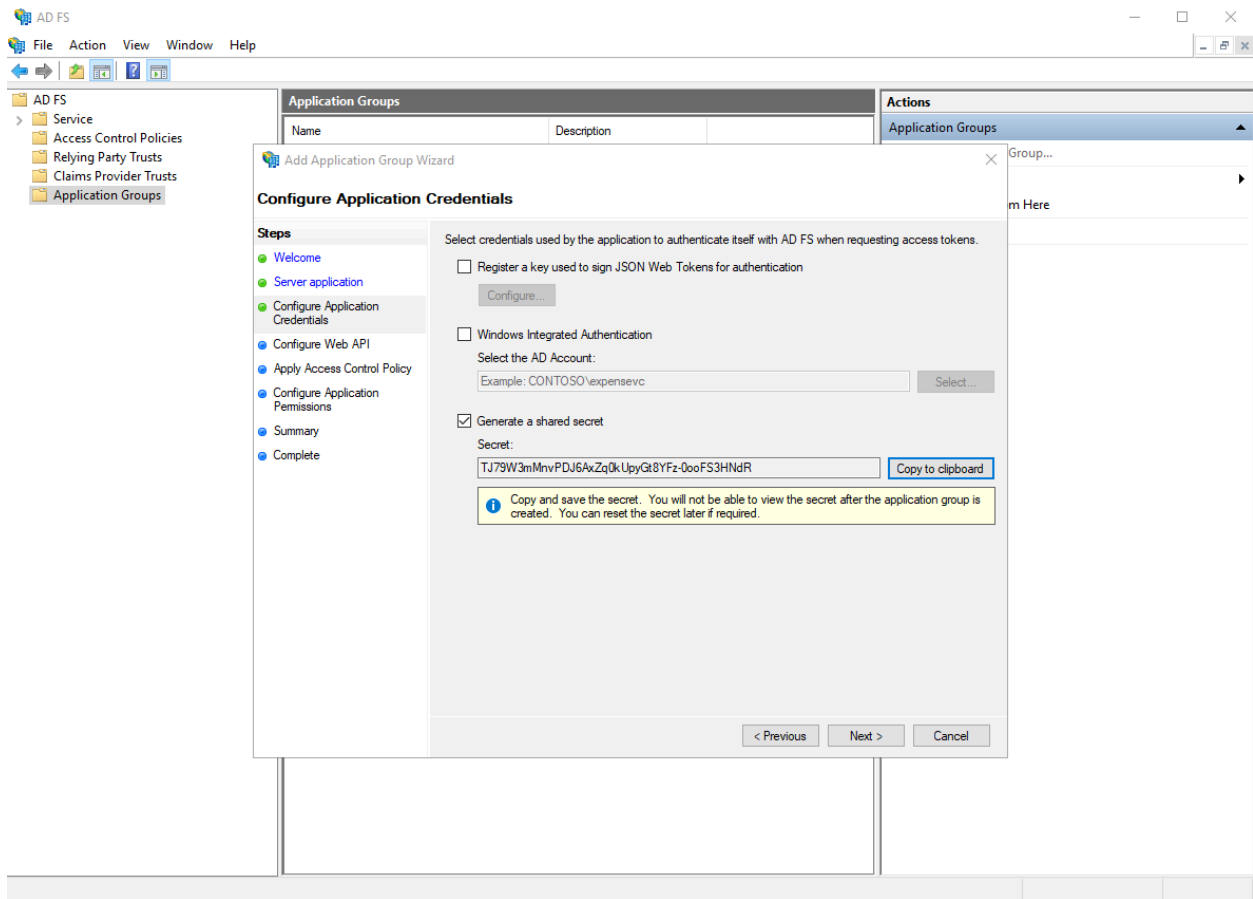
要设置 Active Directory 联合服务，请参考[AD FS 文档](#)以基本了解 ADFS，并查阅[AD FS 部署指南](#)以获取设置 AD FS 服务器的指导。确保在进行下一步操作之前，你已经拥有一个完全可操作的AD FS服务器。

步骤1：通过AD FS启用OAuth

有关逐步创建应用程序的详细说明，请参阅[启用AD FS的OAuth机密客户端指南](#)。

到了这一步的末尾，您应该已经获得了客户端ID和客户端密钥，如下面的截图所示：





第一张图片中的客户端标识符和第二张图片中的秘密应作为OAuth设置中的客户端ID和客户端秘密使用。

启用Casdoor AD FS提供商


在您的Casdoor设置中添加一个AD FS提供商，并输入"Client ID"和"Client Secret"。

Edit Provider

Name [?]:

Display name [?]:

Category [?]:

Type [?]: 

Client ID [?]:

Client secret [?]:

Domain [?]:

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

Azure AD

介绍

Azure Active Directory (Azure AD) 通过为云和本地应用程序提供单一的身份系统，简化了应用程序管理。软件即服务 (SaaS) 应用程序，本地应用程序和业务线 (LOB) 应用程序可以添加到 Azure AD 中。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，并根据需要选择账户类型。演示站使用下面显示的类型。

[Home](#) >

Register an application

* Name

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



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...](#)

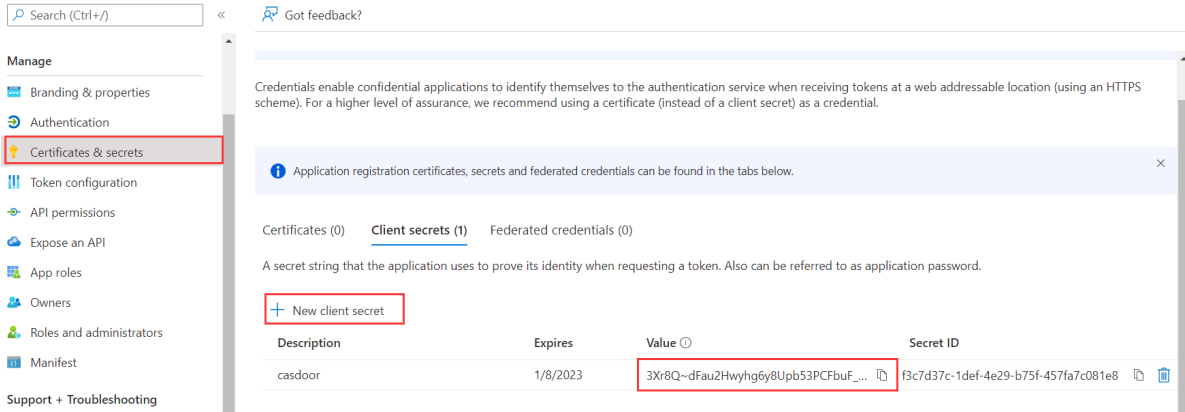
Redirect URI (optional)

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

[Register](#)

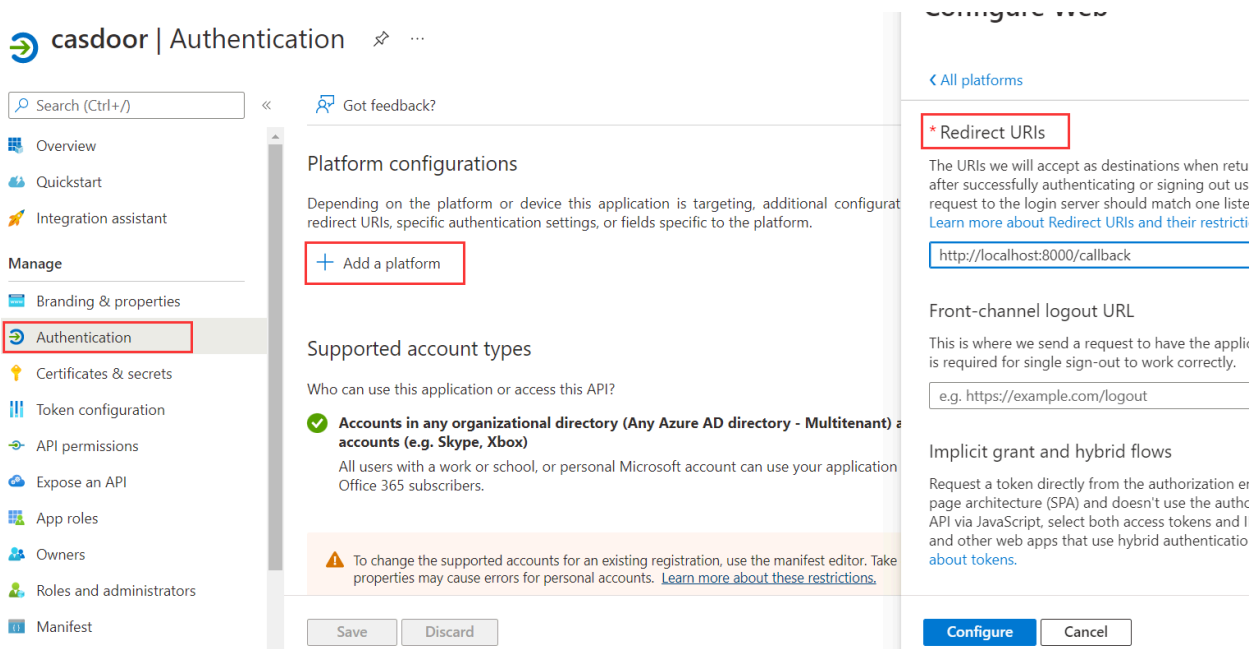
步骤2：创建客户端密钥

创建一个 `client secret` 并保存其值，因为稍后会用到。



步骤3：添加重定向URI

按照图片中的示例为Casdoor添加重定向URI。



步骤4：授予管理员同意

`user.read` API 默认是打开的。您可以根据自己的需要添加更多的作用域。最后，记得给予管理员权限。

Search (Ctrl+/) Refresh Got feedback?

Overview
Quickstart
Integration assistant

Manage

Branding & properties
Authentication
Certificates & secrets
Token configuration
API permissions
Expose an API
App roles
Owners
Roles and administrators
Manifest

Support + Troubleshooting

Troubleshooting
New support request

Successfully granted admin consent for the requested permissions.

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](#)

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 or app will be used. [Learn more](#)

Configured permissions

Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. [Learn more about permissions and consent](#)

+ Add a permission Grant admin consent for Default Directory

API / Permissions name	Type	Description	Admin consent requ...	Status
Microsoft Graph (5)				
email	Delegated	View users' email address	No	Granted for Default Dire...
offline_access	Delegated	Maintain access to data you have given it access to	No	Granted for Default Dire...
openid	Delegated	Sign users in	No	Granted for Default Dire...
profile	Delegated	View users' basic profile	No	Granted for Default Dire...
User.Read	Delegated	Sign in and read user profile	No	Granted for Default Dire...

To view and manage permissions and user consent, try [Enterprise applications](#).

步骤5：在Casdoor中创建AzureAD提供商

最后一步是添加一个AzureAD OAuth提供者，并在您的Casdoor中填写 **Client ID** 和 **Client Secret**。

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/Appli

Save

Save & Exit

Azure AD B2C

介绍

Azure AD B2C 是一种客户身份访问管理解决方案，支持 OpenID Connect、OAuth 2.0 和 SAML 等标准。它允许将面向消费者的应用程序与可扩展和可定制的身份管理解决方案集成。

如何使用？

下面显示了设置 Azure AD B2C 用于身份验证的步骤。

步骤 1：创建 B2C 租户

首先，在您的 Azure 门户中创建一个 B2C 租户。

步骤 2：注册应用程序

在您的 B2C 租户内注册应用程序。

[Home](#) >

Register an application

* Name

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



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...](#)

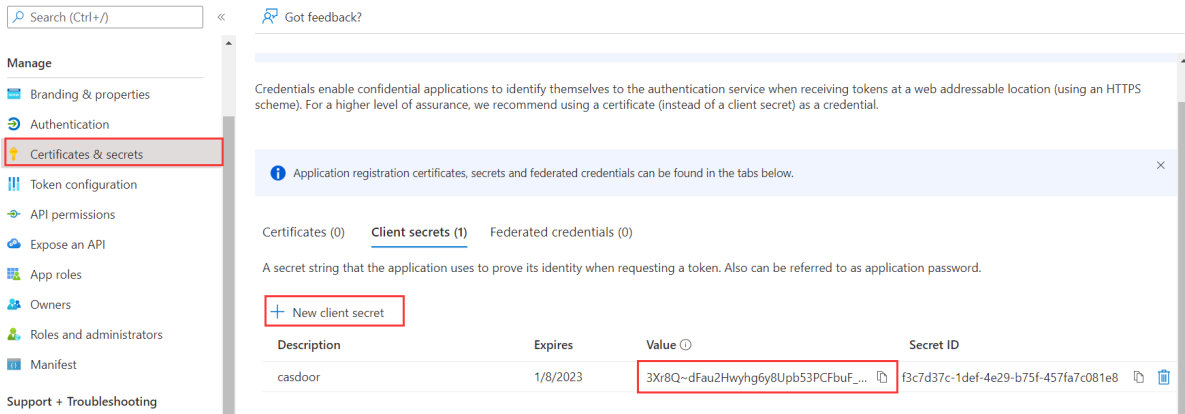
Redirect URI (optional)

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

[Register](#)

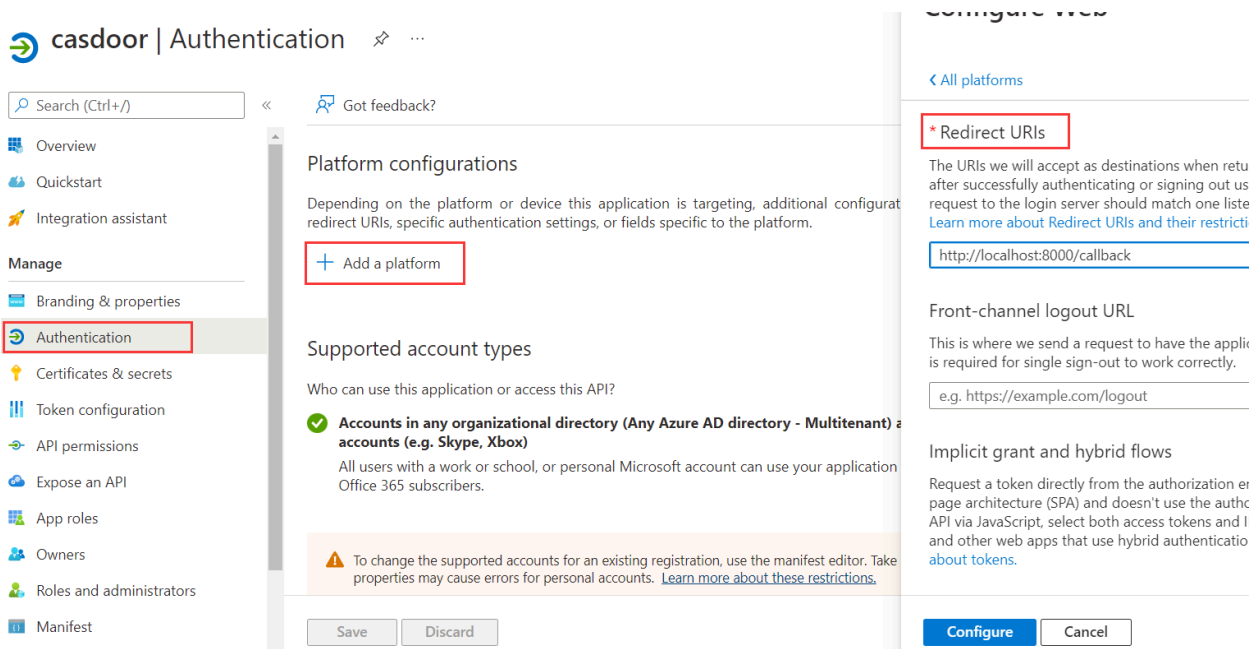
步骤 3: 创建客户端密钥

为您的应用程序创建一个 `client secret` 并保存该值，因为稍后将使用它。



步骤 4: 添加重定向 URIs

在 Azure AD B2C 设置中为您的应用程序添加重定向 URIs。



步骤 5: 定义用户流程

在 Azure AD B2C 中定义用户流程，以管理用户如何注册、登录以及管理他们的个人资料。

步骤 6：在 Casdoor 中创建 Azure AD B2C 提供商

最后，在 Casdoor 中添加一个 Azure AD B2C OAuth 提供商，使用来自您的 B2C 租户的 `Client ID` 和 `Client Secret`。

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/Apli

Save Save & Exit

自定义OAuth

📌 备注

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

要创建一个新的自定义提供商，请导航到Casdoor的提供商页面，并在“类型”字段中选择“自定义”。然后，您需要填写`Client ID`、`Client Secret`、`Auth URL`、`Scope`、`Token URL`、`UserInfo URL`和`Favicon`。

Type ? :	<input type="text" value="Custom"/>
Auth URL ?	<input type="text" value="https://door.casdoor.com/login/oauth/authorize"/>
Scope ?	<input type="text" value="openid profile email"/>
Token URL ?	<input type="text" value="https://door.casdoor.com/api/login/oauth/access_token"/>
UserInfo URL ?	<input type="text" value="https://door.casdoor.com/api/userinfo"/>
Favicon ? :	URL ? : <input type="text" value="🔗"/>
	Preview: <div style="border: 1px solid #ccc; height: 40px; width: 100%;"></div>
Client ID ?	<input type="text"/>
Client secret ?	<input type="text"/>

- `Auth URL` 是自定义提供商的 OAuth 登录页面地址。

如果你在`Auth URL`中填写`https://door.casdoor.com/login/oauth/authorize`，那么，当用户使用这个自

定义提供者登录时，浏览器将首先重定向到

```
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}
```

经过这一步，Casdoor将会识别URL中的code参数。

- **Scope** 是访问 **Auth URL** 时携带的范围参数，您应根据自定义提供商的要求进行填写。
- **Token URL** 是获取accessToken的API端点。

一旦你在前一步获取了代码，Casdoor应该使用它来获取accessToken。

如果你在 **Token URL** 中填入 `https://door.casdoor.com/api/login/oauth/access_token`，那么Casdoor将使用以下命令来访问它

```
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
```

自定义提供者应至少返回以下信息：

```
{
  "access_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjE6IiwiaXN5dWkiOiIyIiwiaWF0Ijoi",
  "refresh_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjE6IiwiaXN5dWkiOiIyIiwiaWF0Ijoi",
  "token_type": "Bearer",
  "expires_in": 10080,
  "scope": "openid profile email"
}
```

- **UserInfo URL** 是通过accessToken获取用户信息的API端点。

如果你在 **UserInfo URL** 中填入 `https://door.casdoor.com/api/userinfo`，那么Casdoor将使用以下命令来访问它

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

自定义提供者应至少返回以下信息：

```
{
  "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。

此标志将与其他第三方登录提供者一起在Casdoor的登录页面上显示。

Okta

要设置Okta OIDC提供商，首先访问[Okta Developer](#)并注册以获取开发者账户。

导航到[应用程序 > 应用程序](#)选项卡，点击[创建应用集成](#)，选择登录方法为OIDC - OpenID Connect，并选择应用类型为Web应用程序，然后点击下一步。

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`。

Sign-in redirect URIs

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

[Learn More](#)

在**任务**部分，为您的应用定义**受控访问**的类型，然后点击**保存**以创建应用集成。

现在你将拥有 **Client ID**，**Client secret** 和 **Okta domain**。

Client Credentials [Edit](#)

Client ID

Public identifier for the client that is required for all OAuth flows.

Client secret

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.

General Settings [Edit](#)

Okta domain

在Casdoor仪表板中添加一个Okta OAuth提供商，输入您的**客户端ID**，**客户端密钥**和**域**。

Edit Provider

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>

⚠ 正确设置域名

请注意，`Domain` 不仅仅是 `Okta domain`；应该在其后追加 `/oauth2/default`。

在授权服务器上访问 [Okta 文档](#) 获取更多详情。

现在您可以使用Okta作为第三方服务来完成身份验证。

Twitter

Twitter（进行中 🚧）

由于官方限制严格，申请Twitter的开发者账号可能会有些繁琐。与其他第三方平台相比，这可能更具挑战性。

首先，访问[开发者门户](#)，如果你还没有账号，就创建一个。Twitter要求你为开发者账号的申请提供详细的应用信息。确保准确填写信息，以避免在审查过程中出现任何问题。

一旦你的申请被批准，你就可以开始创建应用了。你需要在[认证设置](#)部分完成两项重要任务：

1. 手动启用3-legged OAuth。这对于“使用Twitter登录”和代表其他账户发布推文等功能是必要的。
2. 启用[向用户请求电子邮件地址](#)以获取用户的电子邮件地址。

确保仔细填写应用的回调地址和其他必要信息。

微博

Weibo ✓

申请微博开发者账户并不困难，但过程可能会慢，大约需要2-3天。

首先，访问[开发者网站](#)并填写所需的基本信息。然后，你需要等待一个彻底的审查...

一旦你的申请被批准，你将收到客户端ID和客户端密钥。

微信

微信 ✓

若要将微信OAuth提供商添加到你的应用，请遵循以下步骤

1. 访问 [微信开放平台](#) 并注册成为开发者。
2. 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。

Name ?	provider_casdoor_wechat
Display name ?	Casdoor WeChat
Organization ?	admin (Shared)
Category ?	OAuth
Type ?	WeChat
Client ID ?	wx049c70e6c2027b0b
Client secret ?	***
Client ID 2 ?	wxe933a9cd81c396d1
Client secret 2 ?	***
Use WeChat	<input type="checkbox"/>
Media Platform in PC ?	<input type="checkbox"/>
Access token ?	<input type="text"/>
Follow-up action ?	<input type="button" value="Use WeChat Open Platform to login"/> <input type="button" value="Use WeChat Media Platform to login"/>
Provider URL ?	https://open.weixin.qq.com/

服务器配置(已启用)

服务器地址(URL)	https://door.casdoor.com/api/webhook
令牌(Token)	123

WeChat 提供商提供两套不同的密钥:

- 第一组密钥对 (Client ID, Client Secret) 是 WeChat Open Platform (微信开放平台) 的, 并且适用于PC登录场景。它允许你在PC浏览器显示二维码, 让用户能够使用微信APP扫描此二维码进行登录。
- 第二组密钥对 (Client ID 2, Client Secret 2) 以及 Access Token 是 WeChat Media Platform (微信公众平台) 的, 目的是让用户能在微信APP内进行

登录。Access Token 是你在 WeChat Media Platform (微信公众平台) 的服务器配置中填写的 Token。它允许用户使用微信内部的浏览器进行登录，他会将用户重定向到你的 WeChat Official Account (微信公众号) 以实现登录。请注意：微信不支持在除微信APP之外的任何移动浏览器或者APP进行登录。这一限制是由 WeChat 而不是 Casdoor 施加的。

如果您填写了第二对密钥 (Client ID 2, Client Secret 2)，填写了 Access Token 字段并启用了 Enable QR code 开关，那么您可以选择直接使用微信公众平台的信息扫描二维码后登录，或者使用微信开放平台的信息登录，如果您选择使用微信开放平台登录，在用户关注微信公众号后，用户将被要求扫描微信开放平台的二维码进行登录。当用户点击微信按钮进行登录时，Casdoor 会要求用户在继续登录过程之前关注微信公众号。值得注意的是，这只能在PC登录场景中使用，因为手机无法自行扫描二维码。当在移动场景中使用(即WeChat App的内置浏览器)，Casdoor 将自动跳过这一步。

提示

我们建议同时设置两套密钥对用来在 WeChat Open Platform (微信开放平台) 内部连接你的 WeChat Open Platform (微信开放平台) 账号与 WeChat Media Platform (微信公众平台) 账号。从而让 Casdoor 将通过PC和手机登录的用户视为同一用户。

备注

由于 WeChat OAuth 的限制，目前没有任何方法通过微信登录除微信APP以外的任何第三方手机APP或移动浏览器。目前移动端登录只能在微信APP中进行。

更多细节请浏览 [微信开放平台](#)。

企业微信

介绍

WeCom提供了一种使用OAuth的授权登录方法，这使您可以直接从WeCom终端打开的网页中获取成员的身份信息，从而无需登录过程。

有两种类型的应用程序：**内部**应用程序和**第三方**应用程序。

基本配置

要配置WeCom提供商，您需要提供以下参数：

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CorpSecret
Agent ID	应用程序Agentid

❗ 信息

WeCom支持两种授权方法：**静默授权**和**普通授权**。

静默授权：用户点击链接后，页面将重定向到 `redirect_URI?code=CODE&state=STATE`

普通授权：用户点击链接后，会显示一个中间页面供用户选择是否授权。用户确认授权后，他们将被重定向到 `redirect_uri?code=CODE&state=STATE`

有关更多详细信息，请参阅[官方文档](#)。

更多信息

有关内部应用的更多信息，请参阅[内部应用](#)文档。

有关第三方应用的信息，请参阅[第三方应用](#)文档。

腾讯 QQ

腾讯QQ ✓

要将腾讯QQ OAuth提供商添加到您的应用程序中，请访问QQ的认证平台 - [Connect QQ](#)。

首先，您需要申请[成为开发者](#)。在您的申请获得批准后，按照平台的指示获取您的客户端ID和客户端密钥。

钉钉

钉钉 ✓

配置钉钉

要配置钉钉，请访问[钉钉开发者平台](#)并使用您的钉钉账户登录。一旦你在平台上，按照提供的指示获取你的 `客户端 ID` 和 `客户端密钥`。钉钉中的对应术语如下：

术语	钉钉名称
Client ID	AppKey
Client secret	AppSecret

在钉钉中，你可以在应用信息中找到 `Appkey` 和 `AppSecret`。

基础信息

应用信息

开发管理

权限管理

应用功能

机器人与消息推送

事件与回调

登录与分享

酷应用


安全与监控

监控中心

部署与发布

版本管理与发布

应用信息

 casdoor
document

应用凭证

AgentId: 2687194261

AppKey: ding6dposoonm8u4t2g5

AppSecret: hE4cwQ4PjKDSp_ucHTBTqjAAfZfsNGkxwNg1q1FCiiTRW7apxJhzjFOjw46NfFWn

删除应用

删除操作不可逆，该应用所有信息将被删除，请谨慎操作。

删除

确保添加 **重定向域**，它应该是你的Casdoor域。

基础信息

应用信息

开发管理

权限管理

应用功能

机器人与消息推送

事件与回调

登录与分享

酷应用

安全与监控

监控中心

部署与发布

接入登录

添加重定向 URL 作为免登授权码跳转地址。了解更多

* 回调域名

请填写 HTTP/HTTPS 开头的 URL

添加

微应用回调的URL

http://localhost:7001

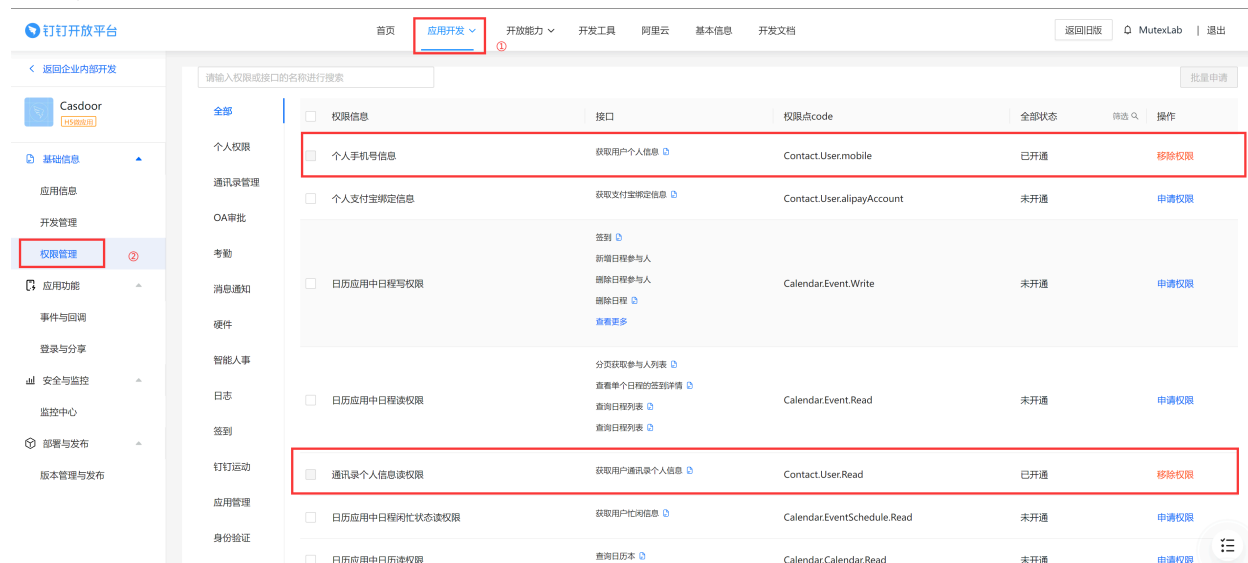
接入分享

嵌入分享SDK，实现一键登录后内容分享。了解更多

iOS 分享

更多详细信息，请参考[钉钉开发者文档](#)。

此外，你需要向钉钉应用添加以下权限：



配置Casdoor

这是钉钉的最终配置：

Name ? :	dingding
Display name ? :	dingding
Organization ? :	admin (Shared)
Category ? :	OAuth
Type ? :	DingTalk
Client ID ? :	ding6dposoonm8u4t2g5
Client secret ? :	***
Provider URL ? :	https://github.com/organizations/xxx/settings/applications/1234567

Steam

Steam ✓

要将Steam OAuth提供商添加到您的应用程序中，请按照以下步骤操作：

1. 访问[Steam WebAPI平台](#)并使用您的Steam账户登录。
2. 为您的Casdoor域名或IP申请一个API密钥。
3. 将您的API密钥作为客户端密钥填入Casdoor。客户端ID不需要填写。
4. 确保您的Steam账户有游戏，以便申请API。

如需获取更详细的信息，请访问[Steam WebAPI文档](#)。

Gitee

要设置 Gitee OAuth 提供商，请转到 [Gitee 开发者](#) 网站。如果你以前没有创建过应用程序，Gitee 工作台将会是这样的：



然后，您可以创建您的Gitee应用。

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>

输入名称、描述、主页和回调URL，并仔细选择权限。

❗ 正确设置授权回调URL

在 Gitee OAuth 配置中，`authorization callback URL` 必须是 **你的 Casdoor 的回调 URL**，并且 Casdoor 中的 `Redirect URL` 应该是 **你的应用程序的回调 URL**。

有关更多详细信息，请阅读[应用配置指南](#)。

创建Gitee应用后，您可以获取 `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

在您的Casdoor中添加一个Gitee OAuth提供商，并输入 `Client ID` 和 `Client Secrets`。

Edit Provider

Save

Name ? :	my_gitee_provider
Display name ? :	Gitee provider
Category ? :	OAuth
Type ? :	Gitee
Client ID ?	300ff94d994a7597850bbafb2d5dc67929676dd8e7176b029e067dc6966ef9c4
Client secret ?	*****

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，因此必须勾选电子邮件选项；否则，将会导致范围授权错误。

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

- All
- user_info Access and update user data, activities, etc
- projects Full control of user projects
- pull_requests Full control of user pull requests
- issues Full control of user issues
- notes Access, create and edit user comments
- keys Full control of user public keys
- hook Full control of user webhook
- groups Full control of user orgs and teams
- gists Access, create and update user gists
- enterprises Full control of user enterprises and teams
- emails Access user emails data

Submit

Delete

百度

要设置Baidu OAuth提供商，请阅读[Baidu文档](#)并按照他们的步骤完成[应用创建](#)。

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.baiu.com>

< 创建工程

* 应用名称: 11/32

传统接入扩展: 合作网站

解决方案: 使用BAE

创建

在创建应用程序后，应在以下位置设置重定向URL：

🔙 Casdoor

- 🏠 基本信息
- 🖥️ 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[redacted]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:

0/500

根域名绑定:

15/255

限制访问OpenAPI的Referer

应用服务器IP地址:

0/500

同时绑定访问OpenAPI服务器IP

确定 取消

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](#)

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


可以同时将应用访问OpenAPI（如Passport、翻译等API）的IP限制在所填的“应用服务器IP地址”的设置项中

⚠ 注意事项

这部分与百度文档中提供的信息有很大的不同：


1. 将URL添加到回调URL设置中很可能会导致URL验证失败，并导致登录失败，因此我们将我们的域名添加到域设置中。
2. 只能添加一个URL或域名，这与文档中的描述非常不同。

然后，您可以获取 `Client ID` 和 `Client Secrets`。


 **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: 


Client ID API Key: HnhK2...QmAp61

Client Secret Secret Key: DTgBZ...1s1bLm1Gha

创建时间: 2022-01-22 16:20:05

更新时间: 2022-01-23 15:45:06

在您的Casdoor中添加一个百度OAuth提供商，并填写 **Client ID** 和 **Client Secrets**。

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

Edit Provider

Name: Baidu

Display name: Baidu

Category: OAuth

Type: Baidu

Client ID: HsM...nWT

Client secret: ***

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

现在您可以使用百度作为第三方服务来完成身份验证!

⚠️ 一般故障排除

如果你遇到百度提示说你的重定向URL是错误的, 以下是一些可能的解决方法:

1. 将您的域名添加到适当的位置, 然后重置Secret (百度重置Secret有一个bug, 它会提示您一个错误, 但刷新页面后Secret已经被刷新)。
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掩码处理, 这与他们的文档显示的用户名和显示名称不同。因此, 我们目前只能使用掩码名称作为用户名。

Infoflow

要设置Infoflow OAuth提供商，请按照以下步骤操作：

1. 请前往[信息流](#)并使用您的信息流账户登录。
2. 访问[信息流应用](#)页面。



3. 注册您的Infoflow应用。



4. 获取AgentID。

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

5. 导航至**设置**选项卡并创建一个新的管理组。



6. 将您的结构添加到地址簿权限，并赋予它必要的权限。另外，将您刚刚创建的应用添加到指定位置。

通讯录权限

[修改](#)

组织架构	查看	管理 [?]
	<input checked="" type="checkbox"/>	<input type="checkbox"/>

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	<input type="radio"/>	<input type="radio"/>

7. 按照所示添加敏感接口权限。

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"/>
维护通讯录	<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"/>

8. 在同一页面上，你会找到 CorpID 和 Secret。

开发者凭据

Client ID

CorpID	hir-216-1
Secret	HgH1-NB

Client Secret [重置](#)

- 在Casdoor中添加一个Infowflow OAuth提供商，并填写 **Client ID**、**Client Secret** 和 **Agent ID**。

Edit Provider

Name ? :	Infowflow
Display name ? :	Infowflow
Category ? :	OAuth
Type ? :	Infowflow
Sub type ? :	Internal
Client ID ? :	<input type="text" value=""/> CorpID
Client secret ? :	<input type="text" value="***"/> Secret
Agent ID ? :	55 AgentID

您现在可以将Infowflow用作第三方服务进行身份验证。

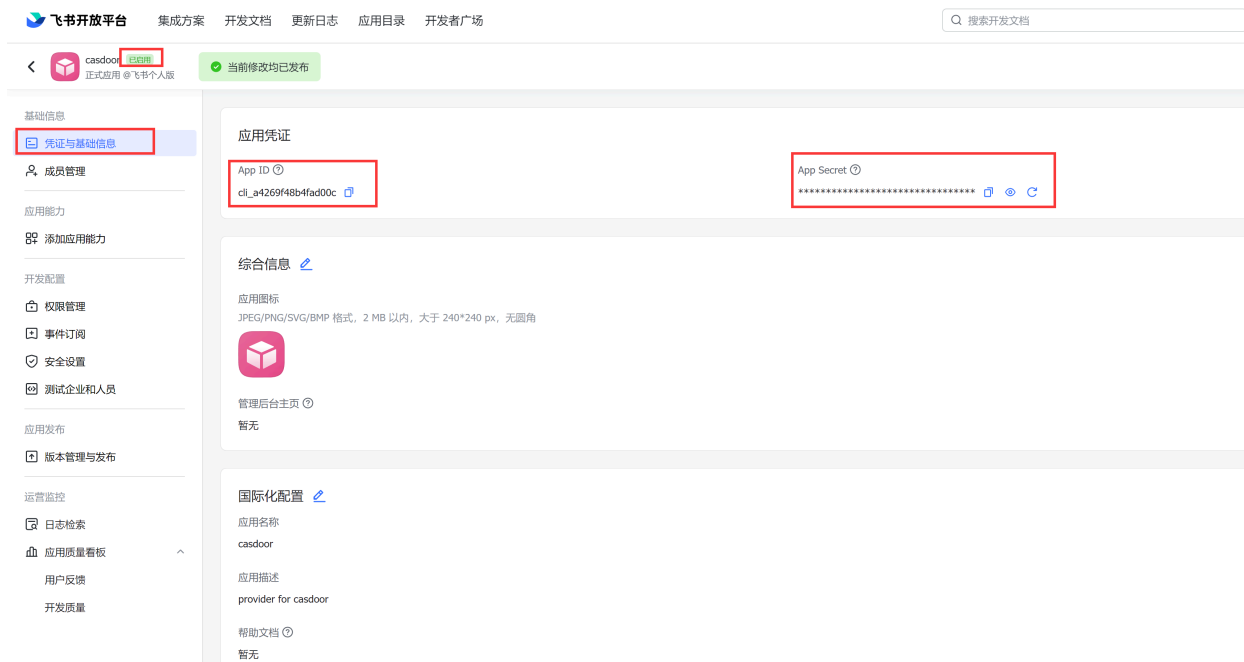
Lark

📘 备注

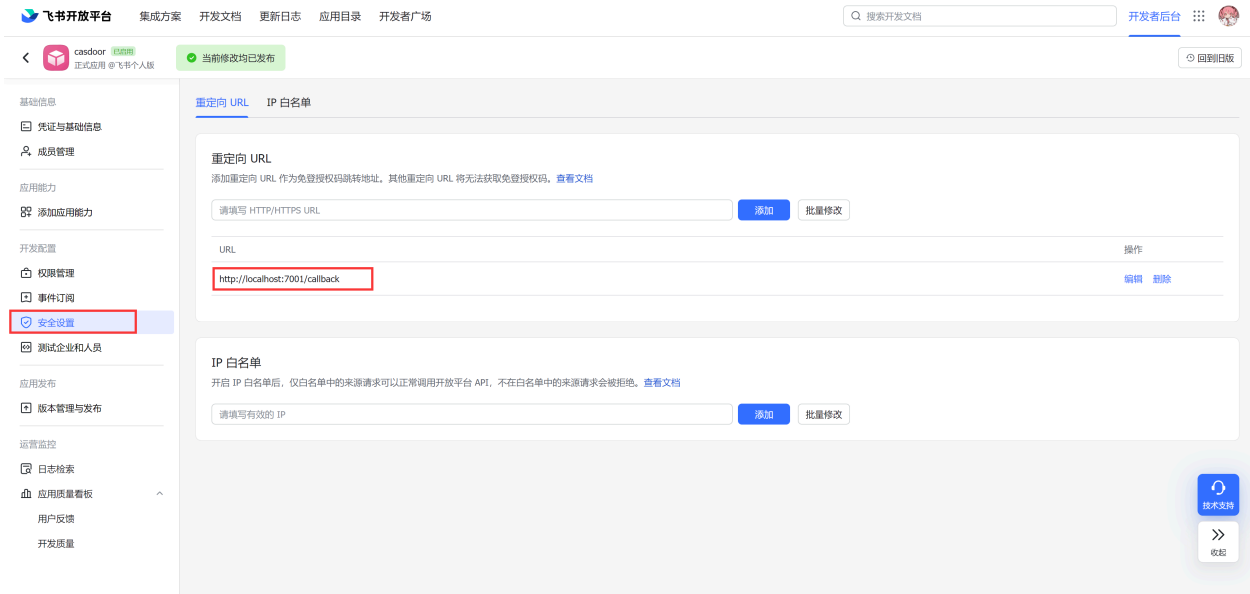
这是一个如何配置Lark OAuth提供商的示例。

步骤1: 创建一个Lark应用程序

首先，您需要在[Lark开放平台](#)上创建一个新的应用程序并启用它。您可以在应用程序的基本信息中找到 `App ID` 和 `App Secret`。



接下来，在您的应用程序的安全设置中添加重定向URL `<your-casdoor-domain>/callback`（例如，`http://localhost:7001/callback`）。



步骤2：创建一个Lark OAuth提供商

现在在Casdoor中创建一个Lark OAuth提供商。填写必要的信息。

名称	Lark中的名称
Category	Choose <code>OAuth</code>
Type	Choose <code>Lark</code>
Client ID	<code>App ID</code> obtained from Step 1
Client secret	<code>App Secret</code> obtained from Step 1



现在您可以使用Lark作为第三方服务来完成身份验证。

电子邮箱

概述

使用电子邮件进行身份验证

SendGrid

使用SendGrid作为SMTP服务器

Azure ACS

使用Azure ACS作为电子邮件提供商

Brevo

使用 Brevo 作为SMTP 服务器


MailHog

使用MailHog作为SMTP服务器


概述

添加电子邮件提供商


1. 点击 **添加** 以添加新的提供商。
2. 在 **类别** 部分下选择 **电子邮件**。

Name  :


email provider

Display name  :

My Email


Category  :

Email


Type  :

Default

3. 为您的SMTP服务填写 **用户名**、**密码**、**主机** 和 **端口** 的字段。

Username  :

no-reply@casbin.com

Password  :

Host  :

 smtp.qiye.aliyun.com

Port  :

465

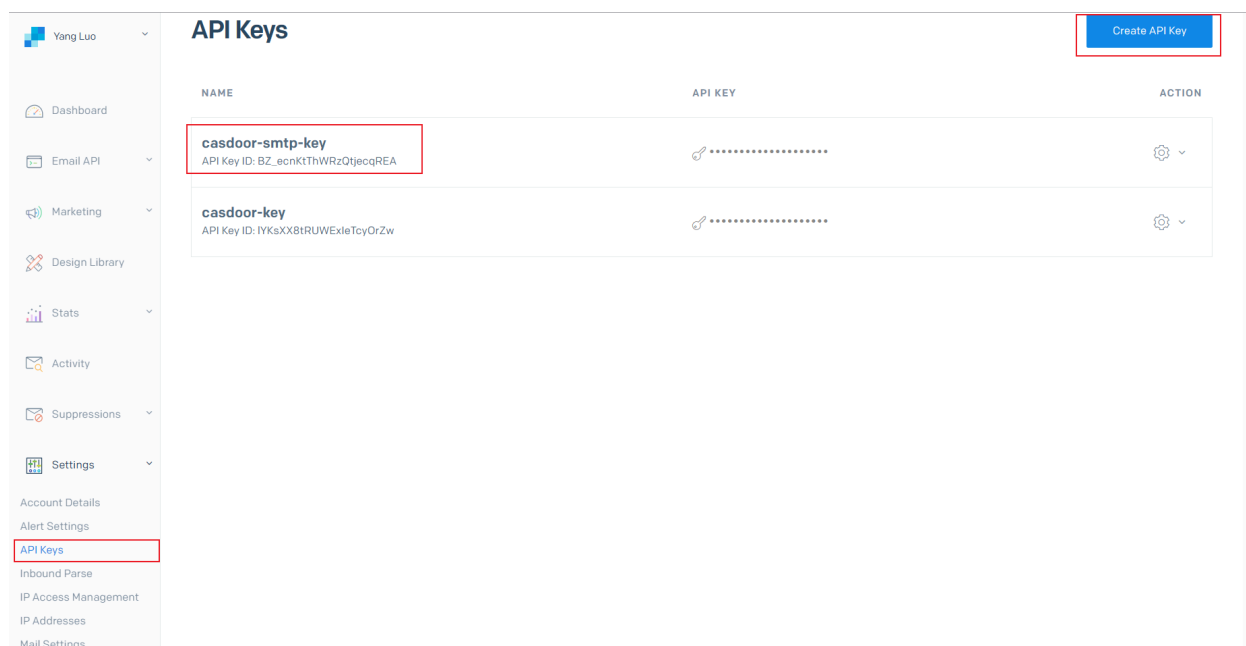
4. 自定义电子邮件标题和电子邮件内容，然后保存更改。

SendGrid

在这个指南中，我们将使用SendGrid作为SMTP服务器。

步骤1：为你的SendGrid账户创建API密钥

从左侧导航栏展开**设置**，然后从此列表中点击**API密钥**选项。在这里，如果你过去生成过任何API密钥，你将会看到所有的API密钥。要生成新的密钥，你需要点击**创建API密钥**并注意权限。





步骤2：发件人验证

参考文档验证你的电子邮件发件人，你可以选择**单一发件人验证**或**域名认证**：[发件人身份](#)

步骤3：配置Casdoor电子邮件提供商

现在在Casdoor中创建一个电子邮件提供商。填写下面的必填字段：

必填字段	备注
用户名	输入 "apikey"
密码	你的SendGrid的API密钥
发件人地址	你的已验证发件人
主机	输入 "smtp.sendgrid.net"
端口	默认为465

Name ? :	sendgrid
Display name ? :	sendgrid
Organization ? :	admin (Shared)
Category ? :	Email
Type ? :	 Default
Username ? :	apikey
Password ? :	***
From address ? :	notifications@casbin.com
From name ? :	casdoor
Host ? :	 smtp.sendgrid.net
Port ? :	465
Disable SSL ? :	<input type="checkbox"/>
Email title ? :	Casdoor Verification Code
Email content ? :	You have requested a verification code at Casdoor. Here is your code: %s, please enter in 5 minutes.
Test Email ? :	<input type="text" value="1270329076@qq.com"/> <input type="button" value="Test SMTP Connection"/> <input type="button" value="Send Testing Email"/>

点击 **测试SMTP连接** 按钮。如果你看到 **provider: SMTP成功连接**，这意味着你的 Casdoor服务可以访问SendGrid服务。

接下来，点击 **发送测试邮件** 按钮。如果你看到 **邮件发送成功**，这意味着测试邮件已经从 **发件人** 地址成功发送到 **测试邮件**。

Azure ACS

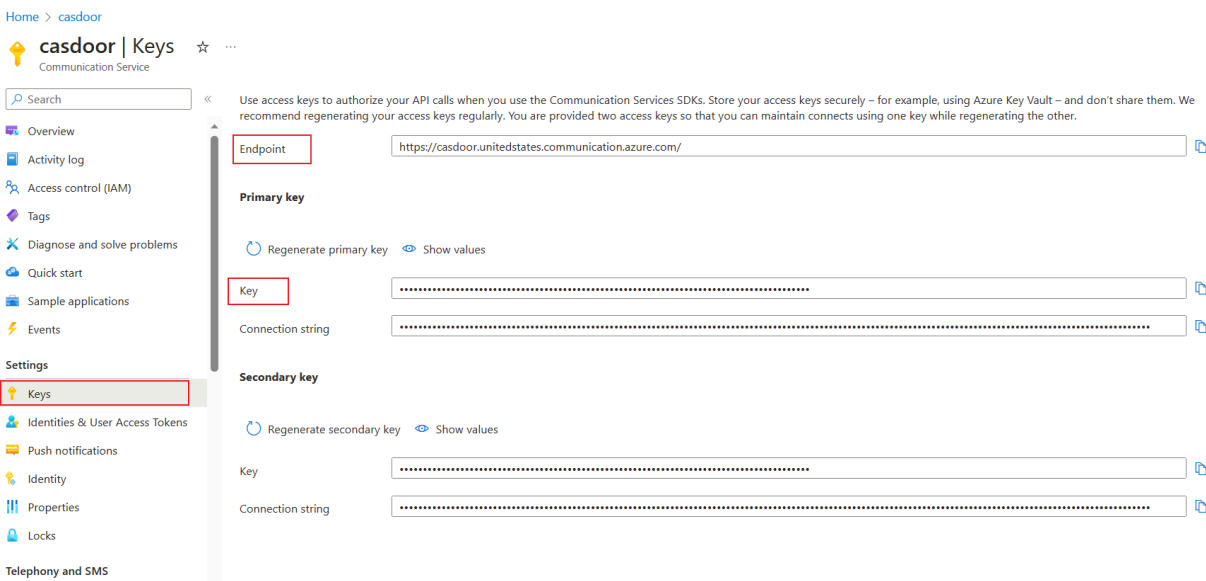
在本指南中，我们将使用ACS作为电子邮件提供商。

步骤1：配置ACS

按照下面的文档完成配置。

- [创建和管理电子邮件通信服务](#)
- [获取一个免费的Azure托管域或添加一个自定义域](#)
- [连接域](#)

复制你的 **Endpoint** 和 **Private Key** 以供使用



步骤2：配置Casdoor电子邮件提供商

现在在Casdoor中创建一个电子邮件提供商，填写必要的信息。 字段与Azure ACS之间

的关系如下:

i 备注

From Address 必须是一个经过验证的电子邮件域。

名称	在Azure ACS中的名称
发件人地址	
密钥	私钥
主机	端点

Name ⓘ:

Display name ⓘ:

Organization ⓘ:

Category ⓘ:

Type ⓘ:

Secret key ⓘ:

From address ⓘ:

Host ⓘ:

Email title ⓘ:

Email content ⓘ:

Test Email ⓘ:

Provider URL ⓘ:

Brevo

在本指南中，我们将使用 Brevo 作为SMTP 服务器。

第 1 步：请求激活您的 Brevo SMTP 帐户

请参阅文档以激活 Brevo SMTP: [发送交易电子邮件使用Brevo SMTP](#)

在我的案例下，当我创建一个工单来激活我的smtp帐户时，我得到了这个答复：



Rafael Guimaraes

Last Response: 2 days ago

Hi there,

Thank you for reaching out!

I've activated your account's SMTP/Transactional capabilities.

You can find your account's SMTP credentials by clicking [here](#).

To get you started, I'll include some useful links about our SMTP/Transactional services:

- [Our complete library of help articles related to SMTP](#)
- [Troubleshooting common issues with SMTP](#)

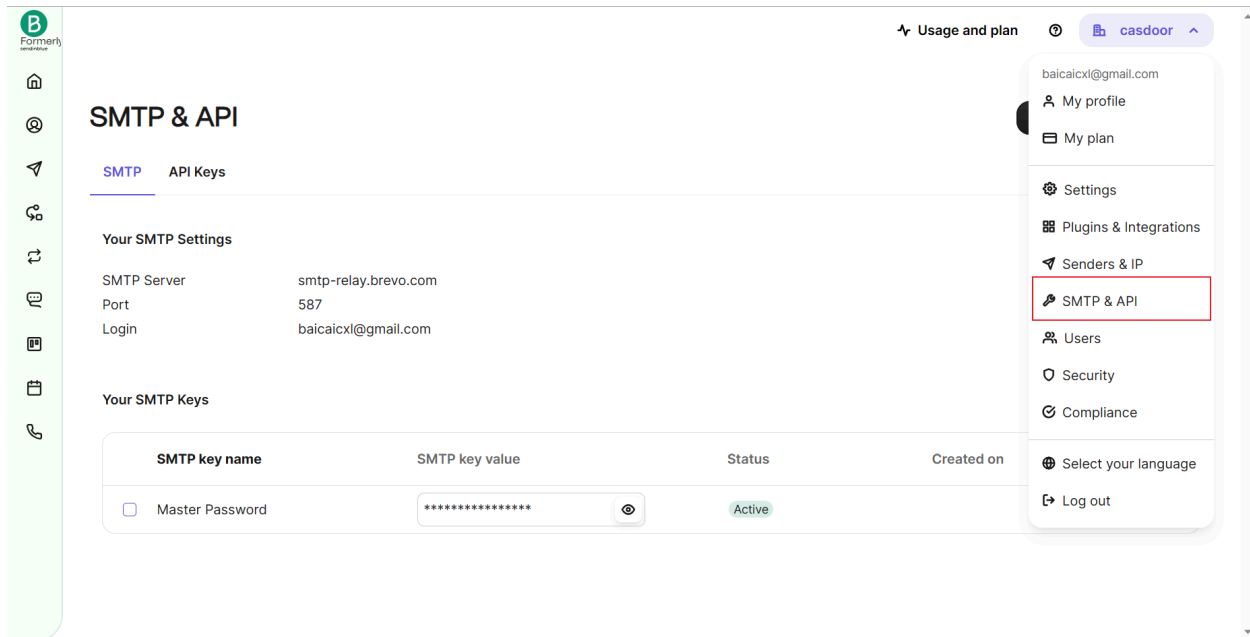
The SMTP port listed by default, Port 587, will be used without a secure connection. If you want to use a secure connection (SSL or TLS), please use Port 465.

Please be sure to let me know if you have more questions or if you need any help.

Best regards,

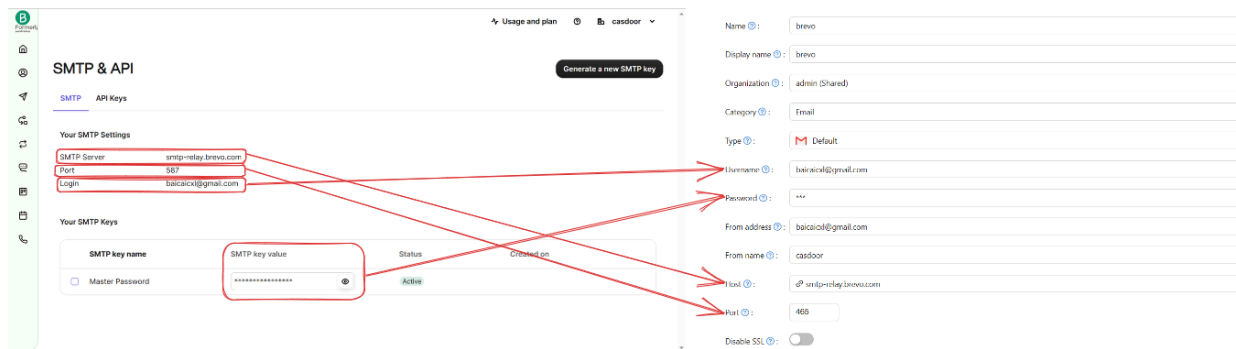
步骤 2: 获取 Brevo SMTP 配置

在你的 Brevo 看板中, 找到 SMTP&API, 查看 [SMTP 服务器](#), [端口](#), [登录](#), [SMTP 验证 Key](#) 信息



第 3 步：配置 Casdoor 邮件提供商

在 Casdoor 中创建一个新的提供商。填写必要的信息。



单击 **测试SMTP连接** 按钮。如果您看到 **提供商：SMTP连接成功**，这意味着您的 Casdoor 服务可以访问 Brevo 服务。

接下来，点击 **发送测试电子邮件** 按钮。如果你看到 **邮件发送成功**，意味着测试邮件已经成功的从 **发件地址** 发送至 **测试接收邮箱**

MailHog

在本指南中，我们将使用MailHog作为SMTP服务器。MailHog是一个使用假SMTP服务器操作的电子邮件测试工具。

步骤1：部署MailHog服务

MailHog服务的IP地址是192.168.24.128，SMTP服务端口是1025。

```
[HTTP] Binding to address: 0.0.0.0:8025
2023/07/13 03:06:43 Serving under http://0.0.0.0:8025/
Creating API v1 with WebPath:
Creating API v2 with WebPath:
[APIv1] KEEPALIVE /api/v1/events
[HTTP] Binding to address: 0.0.0.0:8025
Creating API v1 with WebPath:
Creating API v2 with WebPath:
2023/07/13 03:10:36 Using maildir message storage
2023/07/13 03:10:36 Maildir path is /tmp/mailhog641072855
2023/07/13 03:10:36 [SMTP] Binding to address: 0.0.0.0:1025
2023/07/13 03:10:36 Serving under http://0.0.0.0:8025/
[APIv2] GET /api/v2/jim
[APIv2] GET /api/v2/messages
```

步骤2：创建电子邮件提供商

提供必要的信息并保存设置。

Category ? :	Email	
Type ? :	Default	
Username ? :		
Password ? :		
From address ? :	notification@casdoor.com	
From name ? :	Casdoor Notification	
Host ? :	🔗 192.168.24.128	
Port ? :	1025	
Disable SSL ? :	<input checked="" type="checkbox"/>	
Email title ? :	Casdoor Verification Code (Test)	
Email content ? :	You have requested a verification code at <u>Casdoor</u> (Test). Here is your code: <u>%s</u> , please enter in 5 minutes.	
Test Email ? :	admin@example.com	<input type="button" value="Test SMTP Connection"/> <input type="button" value="Send Testing Email"/>
Provider URL ? :	🔗 https://github.com/organizations/xxx/settings/applications/1234567	

步骤3：发送测试电子邮件

首先，点击 **测试SMTP连接** 按钮。如果你看到 **provider: SMTP connected successfully**，这意味着你的Casdoor服务可以访问MailHog服务。

接下来，点击 **发送测试电子邮件** 按钮。如果你看到 **电子邮件发送成功**，这意味着测试电子邮件已经从 **From** 地址成功发送到 **测试电子邮件**。

Name ⓘ : email_provider

Display name ⓘ : Email Provider

Organization ⓘ : admin (Shared)

Category ⓘ : Email

Type ⓘ : Default

Username ⓘ :

Password ⓘ :

From address ⓘ : notification@casdoor.com

From name ⓘ : Casdoor Notification

Host ⓘ : 192.168.24.128

Port ⓘ : 1025

Disable SSL ⓘ :

Email title ⓘ : Casdoor Verification Code (Test)

Email content ⓘ : You have requested a verification code at Casdoor (Test). Here is your code: %s, please enter in 5 minutes.

Test Email ⓘ : admin@example.com

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

provider:SMTP connected successfully

Email sent successfully

Test SMTP Connection

Send Testing Email

MailHog

Search

Connected

Inbox (4)

Delete all messages

Jim

Jim is a chaos monkey.
Find out more at GitHub.

Enable Jim

From: "Casdoor Notification" <notification@casdoor.com>

Subject: Casdoor Verification Code (Test)

To: admin@example.com

HTML Plain text Source

You have requested a verification code at Casdoor (Test). Here is your code: 123456, please enter in 5 minutes.

短信

概述

使用短信进行身份验证

Twilio

使用Twilio作为Casdoor的SMS提供商

Amazon SNS

使用Amazon SNS作为Casdoor的SMS提供商

Azure ACS

使用ACS作为Casdoor的SMS提供商

阿里云


将阿里云作为Casdoor的短信提供商

概述

我们使用 [casdoor/go-sms-sender](#) 来为Casdoor发送短信。`go-sms-sender` 库目前支持Twilio、Submail、SmsBao、阿里云、腾讯云、华为云和Volc短信API。如果你想增加对其他短信提供商的支持，你可以提出一个问题或提交一个拉取请求。

添加短信提供商

1. 点击 `添加` 来添加一个新的提供商。
2. 在 `类别` 部分选择 `短信`。



Category ⓘ: SMS

Type ⓘ: AI

Client ID ⓘ: Captcha

Client secret ⓘ: Email

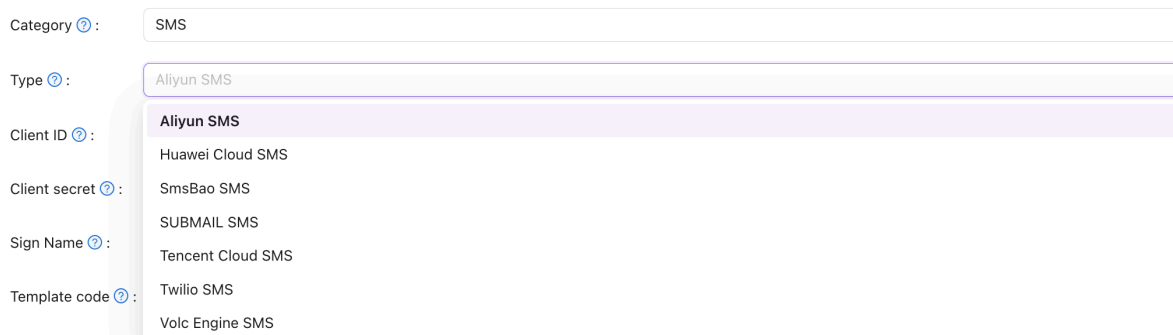
Sign Name ⓘ: OAuth

Template code ⓘ: Payment

SMS

Storage

3. 选择你的提供商的类型。



Category ⓘ: SMS

Type ⓘ: Aliyun SMS

Client ID ⓘ: Aliyun SMS

Client secret ⓘ: Huawei Cloud SMS

Sign Name ⓘ: SmsBao SMS

Template code ⓘ: SUBMAIL SMS

Tencent Cloud SMS

Twilio SMS

Volc Engine SMS

4. 从你的短信提供商处获取必要的信息，并填写相应的字段。

Twilio

在Casdoor中填写必要的信息

有四个必填字段：`客户端ID`，`客户端密钥`，`发送者号码`和`模板代码`。与Twilio帐户的对应关系如下：

名称	Twilio中的名称	必填
Client ID	Account SID	必填
Client secret	Auth Token	必填
Sender number	Twilio电话号码	必填
Template code		必填

Twilio信息

- 帐户SID，授权令牌和Twilio电话号码

Step 4: Invite and upgrade

Invite teammates
Invite developers to your Twilio account to start building! [Learn more about user access management](#)

[Invite teammates](#)

[Upgrade your account](#)
Upgrade your account to send to any number, buy local more. [Learn more about trial account limitations](#)

[Upgrade](#)

[Back](#)

Account Info

Account SID
AC06b73d65c8ee67ce8e448edcc64b6ec6

Auth Token
..... [Show](#)

⚠ Always store your token securely to protect your account. [Learn more](#)

My Twilio phone number
+12186751069

Helpful links

- [How does Twilio work?](#)
- Understand how to use Twilio in a 2
- [SMS Quickstart guides](#)
- Learn the basics of Twilio Messagir
- [Support help center](#)
- Troubleshoot common issues.

配置Casdoor提供商

您可以根据您的需求配置 **模板代码**，然后在 **SMS测试** 中输入您的电话号码进行测试。

Name ⓘ : twilio

Display name ⓘ : twilio

Organization ⓘ : admin (Shared)

Category ⓘ : SMS

Type ⓘ : Twilio SMS

Client ID ⓘ : AC06b73d65c8ee67ce8e448edcc64b6ec6

Client secret ⓘ : ***

Sender number ⓘ : +12186751069

Template code ⓘ : get the message

SMS Test ⓘ : +1 [Send Testing SMS](#)

Provider URL ⓘ : [🔗](#)

Amazon SNS

在Amazon中获取必要的信息

有四个必填字段：`Access Key`，`Secret Access Key`，`Region`和`Template code`。我将向您展示如何从Amazon SNS获取此信息。

- Access Key和Secret Access Key

在身份和访问管理（IAM）中，您可以创建一个`Access Key`和`Secret Access Key`。

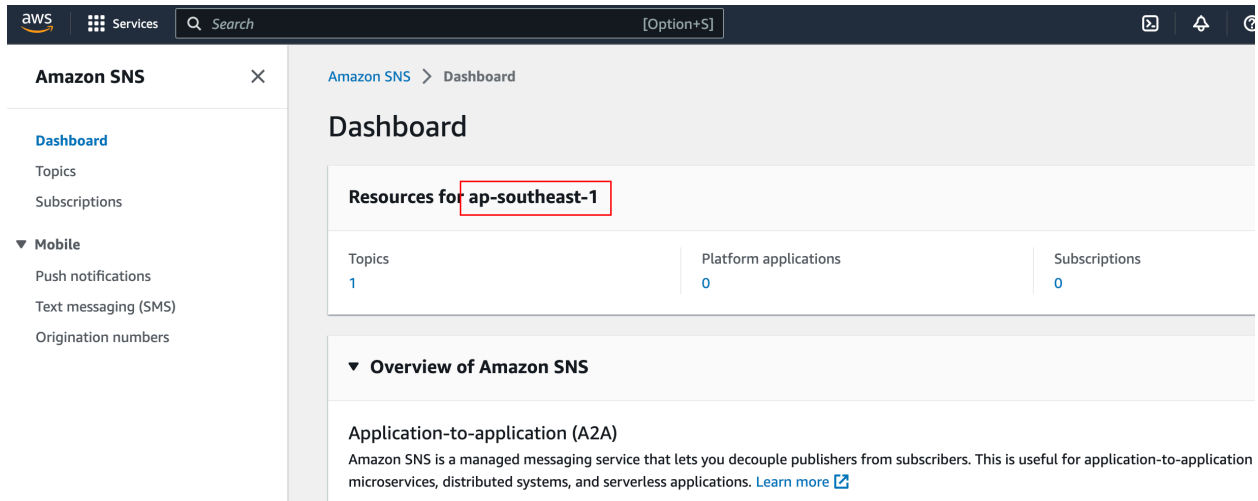
The screenshot shows the AWS IAM console interface. On the left is a navigation sidebar with 'Identity and Access Management (IAM)' selected. The main content area is divided into two sections. The top section, 'Access keys (1)', has a red box around the title. It includes a description, a 'Learn more' link, and a 'Create access key' button. Below is a table with one row of data. The bottom section, 'CloudFront key pairs (0)', includes a description, an 'Upload' button, and a 'Create CloudFront key pair' button. Below this is a table that is currently empty, with a 'No CloudFront key pairs' message and another 'Create CloudFront key pair' button.

Access key ID	Created on	Access key last used	Region last used	Service last used	Status
AKIAIYOMMXHVZLH4LYKGL	16 days ago	None	N/A	N/A	Active

Creation time	CloudFront key ID	Status
No CloudFront key pairs		


- 区域

`Region`与您创建的主题有关。



配置Casdoor提供商

Template code 是您想要发送的消息。在 SMS Test 中输入您的电话号码进行测试。

Name ? :	amazon_sns
Display name ? :	amazon_sns
Organization ? :	admin (Shared)
Category ? :	SMS
Type ? :	 Amazon SNS
Access key ? :	AKIAYOMMXHVZACW5RFMX
Secret access key ? :	***
Region ? :	ap-southeast-1
Template code ? :	enter the message you want to send
SMS Test ? :	+1 <input type="text" value="Input your phone num..."/> <input type="button" value="Send Testing SMS"/>
Provider URL ? :	https://github.com/organizations/xxx/settings/applications/1234567

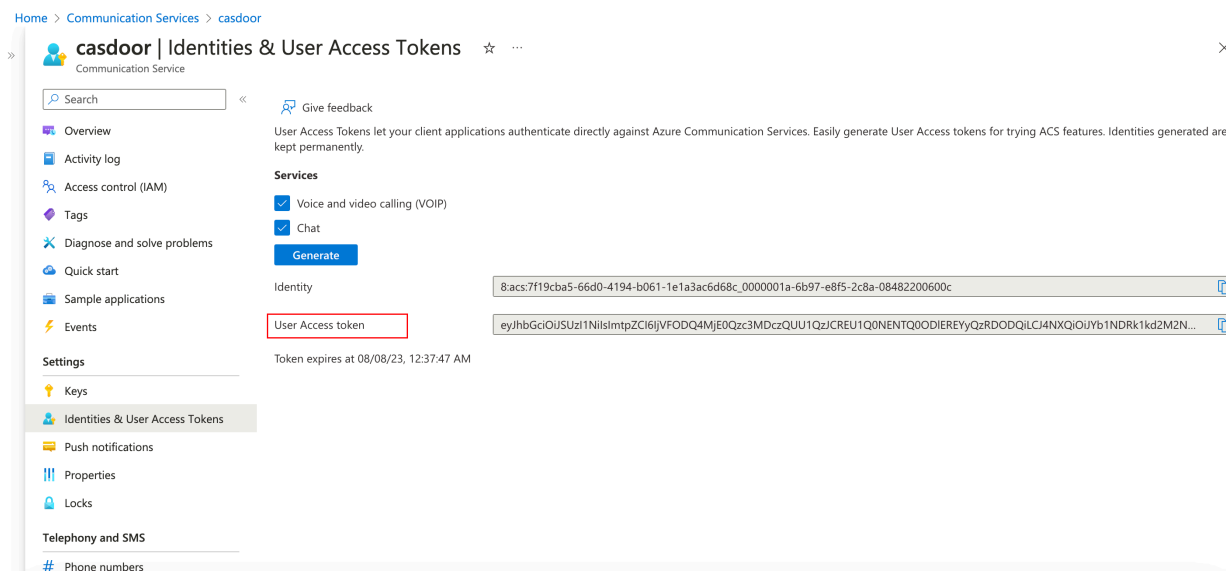
Azure ACS

在Azure中获取必要的信息

有四个必填字段：`客户端密钥`，`发送者号码`，`模板代码`和`提供商url`。我将向你展示如何从Azure ACS获取这些信息。

- `客户端密钥`

在通信服务中，你可以创建一个用户访问令牌，这就是Casdoor中的`客户端密钥`。



- `发送者号码`

`发送者号码`是你在通信服务中创建的电话号码。

Communication Service

Search (Cmd+/) << + Get → Port 🗑 Release 👤 Give feedback

LOCKS

Tools

- Keys
- Identities & User Access Tokens
- Push notifications

Voice Calling - PSTN

- # Phone numbers
- Direct routing (Preview)

SMS

- Short Codes (Preview)

Monitoring

- Insights (preview)
- Metrics
- Diagnostic settings
- Logs

<input checked="" type="checkbox"/>	Number	Status	Cost (monthly)
<input checked="" type="checkbox"/>	1-833-920-3625 📄	Active	\$2

- 提供商Url

提供商Url 是通信服务中的端点。

Communication Service

Search << → Move 🗑 Delete 👤 Give feedback

Overview

- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Quick start
- Sample applications
- Events

Settings

- Keys
- Identities & User Access Tokens

Essentials

Resource group (move) : [casdoor](#)

Status : Active

Location : Global

Subscription (move) : [免费试用](#)

Subscription ID : e054e27a-96e8-4cca-a1cf-32717dcd303c 📄

Tags (edit) : [Add tags](#)

Endpoint : <https://casdoor.unitedstates.communication.azure.com>

Data location : United States

Manage keys : [Click here to manage keys](#)

Build engaging communication experiences at scale

Azure Communication Services brings rich communication APIs to all of your apps across any device, on any platform, using the same reliable and secure infrastructure that powers Microsoft Teams. [Learn more](#)

配置Casdoor提供商

模板代码就是你想发送的消息。在SMS测试中输入你的电话号码进行测试。

Name ? :	<input type="text" value="azure_acs"/>
Display name ? :	<input type="text" value="azure_acs"/>
Organization ? :	<input type="text" value="admin (Shared)"/>
Category ? :	<input type="text" value="SMS"/>
Type ? :	<input type="text" value="Azure ACS"/>
Client ID ? :	<input type="text"/>
Client secret ? :	<input type="text" value="***"/>
Sender number ? :	<input type="text" value="+18339203625"/>
Template code ? :	<input type="text" value="enter the message you want to send"/>
SMS Test ? :	<input type="text" value="+1"/> <input type="text" value="Input your phone num..."/> <input type="button" value="Send Testing SMS"/>
Provider URL ? :	<input type="text" value="https://casdoor.unitedstates.communication.azure.com"/>

阿里云

在Casdoor中填写必要的信息

有四个必填字段：`客户端ID`，`客户端秘密`，`签名名称`，和`模板代码`。与阿里云账户的对应关系如下：

名称	在阿里巴巴中的名称	是必需的
Client ID	AccessKey ID	必需的
Client secret	AccessKey Secret	必需的
Sign Name	Signature	必需的
Template code	Template code	必需的

阿里巴巴信息

- AccessKey ID和AccessKey Secret

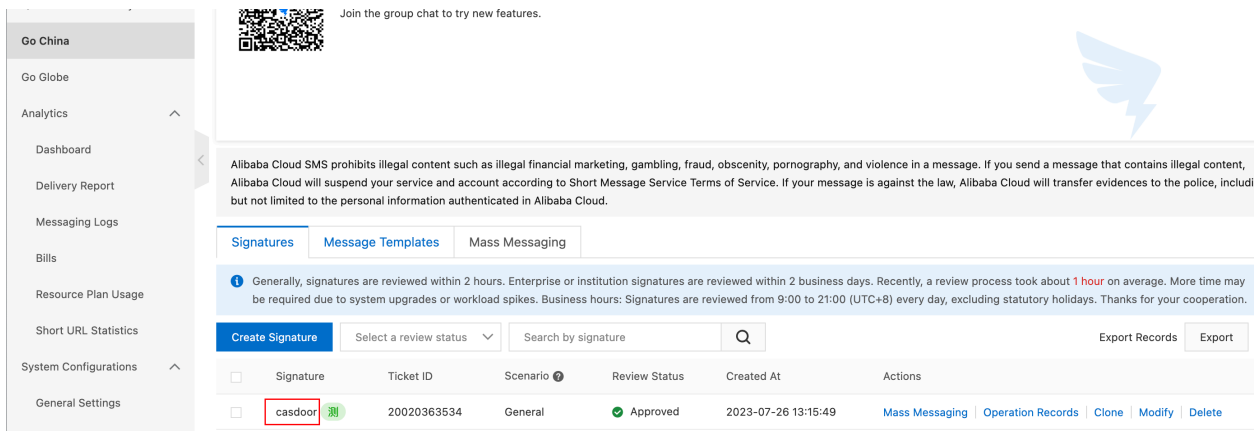
登录我的阿里云工作台后，我点击“AccessKey”创建ID和Secret。



通过创建AccessKey，我获得了我的AccessKey ID和AccessKey Secret：



• 签名



• 模板代码

Short Message Service

- Overview
- Quick Start & Delivery Test
- Go China**
- Go Globe
- Analytics ^
- Dashboard <
- Delivery Report
- Messaging Logs
- Bills
- Resource Plan Usage
- Short URL Statistics
- System Configurations ^
- General Settings
- Domestic SMS Settings

Create Dedicated DingTalk Group Chat

Scan the QR code to create your dedicated DingTalk group chat.
You can use the group chat to submit and modify signatures and message templates, and check whether the signatures and message templates are approved.
Join the group chat to try new features.

Alibaba Cloud SMS prohibits illegal content such as illegal financial marketing, gambling, fraud, obscenity, pornography, and violence in a message. If you send Alibaba Cloud will suspend your service and account according to Short Message Service Terms of Service. If your message is against the law, Alibaba Cloud will not be limited to the personal information authenticated in Alibaba Cloud.

Signatures
Message Templates
Mass Messaging

i 1. Generally, signatures are reviewed within 2 hours. Recently, a review process took about **1 hour** on average. More time may be required due to system up hours: Signatures are reviewed from 9:00 to 21:00 (UTC+8) every day, excluding statutory holidays. Thanks for your cooperation.

2. Message templates provided by Alibaba Cloud SMS do not require submission for approval. However, you are still charged for message delivery.

Create Message Template
Select a template type v
Select a review status v
Search by message template name r

	Template Name	Tag	Ticket ID	Template Code	Template Type	Created At
<input type="checkbox"/>	casdoor 测	v	20020389144	SMS_462155126	Verification Code Message	2023-07-26 17:54:00

配置Casdoor提供商

在 **短信测试** 字段中输入您的电话号码进行测试。

Name ?:

Display name ?:

Organization ?:

Category ?:

Type ?:

Client ID ?:

Client secret ?:

Sign Name ?:

Template code ?:

SMS Test ?:

+86 v

Input your phone num...

Send Testing SMS

Provider URL ?:

通知

概述

在您的应用程序中添加通知提供商

Telegram

使用Telegram作为Casdoor的通知提供者

自定义HTTP

使用自定义HTTP作为Casdoor的通知提供者

Slack

使用Slack作为Casdoor的通知提供者

Google Chat

使用Google Chat作为Casdoor的通知提供者

Twitter

将Twitter用作Casdoor的通知提供商









Discord










将Discord用作Casdoor的通知提供者

概述

Casdoor可以配置为使用各种通知提供商发送通知消息。

目前，Casdoor支持多个通知提供商。 以下是Casdoor支持的提供商：

提供商	Logo
Telegram	
自定义HTTP	
Slack	
Google Chat	
Twitter	
Discord	
Bark	
DingTalk	

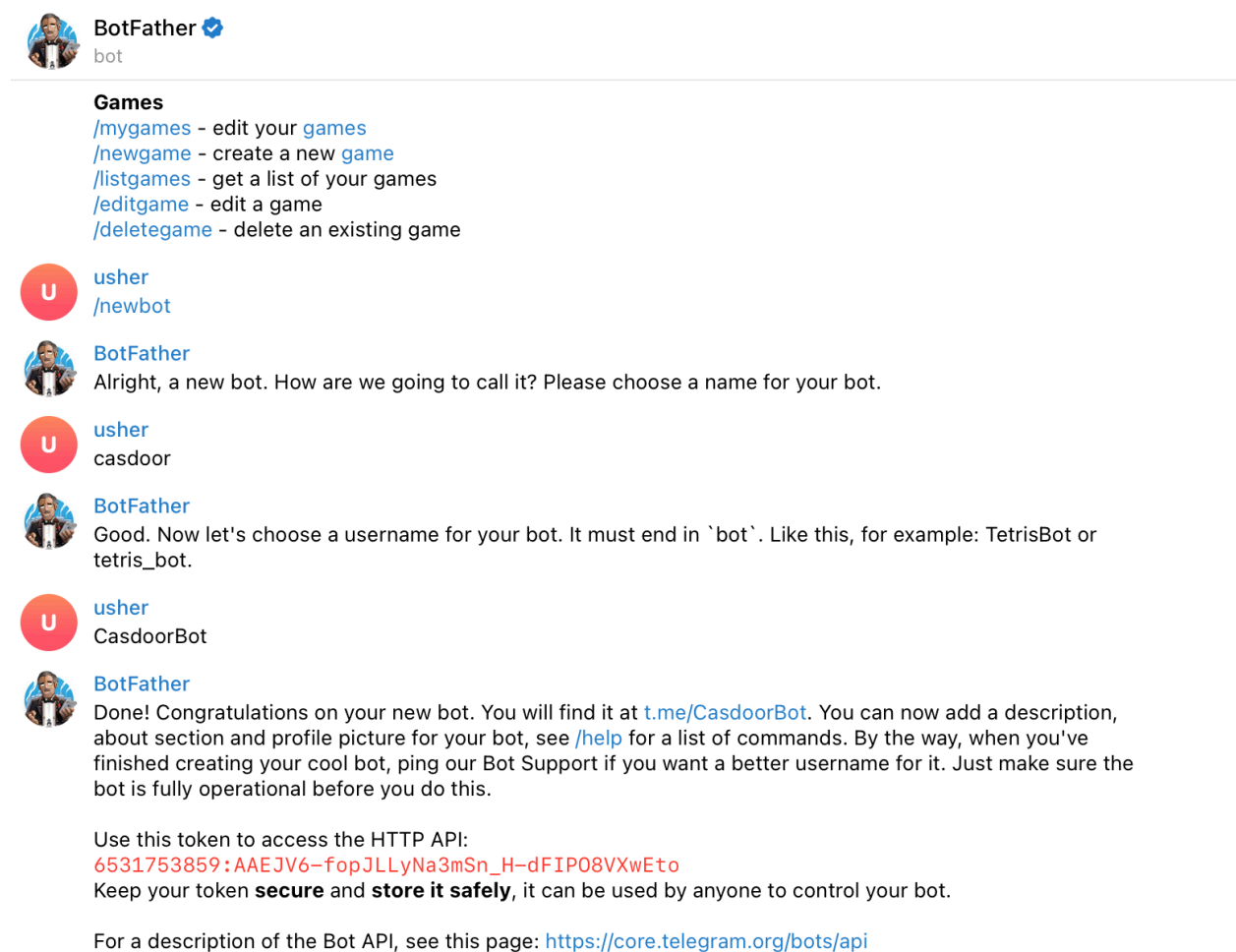
提供商	Logo
Lark	
Line	
Matrix	[matrix]
Microsoft Teams	
Pushbullet	
Pushover	
Reddit	
Rocket Chat	
Viber	
Webpush	

Telegram

步骤1: 获取API令牌

首先, 您需要在Telegram上创建一个账户。创建账户后, 您应该联系BotFather, 这是一个用于创建其他机器人的机器人。

要创建您的机器人, 使用命令 `/newbot` :



The screenshot shows a chat conversation with BotFather. The chat starts with a list of commands for the 'Games' section: `/mygames`, `/newgame`, `/listgames`, `/editgame`, and `/deletegame`. Then, a user named 'usher' sends the command `/newbot`. BotFather responds: 'Alright, a new bot. How are we going to call it? Please choose a name for your bot.' The user 'usher' replies with 'casdoor'. BotFather then asks for a username, and the user replies 'CasdoorBot'. BotFather concludes by congratulating the user and providing the API token: `6531753859:AAEJV6-fopJLLyNa3mSn_H-dFIP08VXwEto`. The chat also includes instructions to keep the token secure and a link to the Bot API documentation.

Games
`/mygames` - edit your [games](#)
`/newgame` - create a new [game](#)
`/listgames` - get a list of your games
`/editgame` - edit a game
`/deletegame` - delete an existing game

usher
`/newbot`

BotFather
Alright, a new bot. How are we going to call it? Please choose a name for your bot.

usher
casdoor

BotFather
Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris_bot.

usher
CasdoorBot

BotFather
Done! Congratulations on your new bot. You will find it at t.me/CasdoorBot. You can now add a description, about section and profile picture for your bot, see [/help](#) for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

Use this token to access the HTTP API:
`6531753859:AAEJV6-fopJLLyNa3mSn_H-dFIP08VXwEto`
Keep your token **secure** and **store it safely**, it can be used by anyone to control your bot.

For a description of the Bot API, see this page: <https://core.telegram.org/bots/api>

您的机器人应该有两个属性：`name`和`username`。创建机器人后，您将收到一个API令牌。

步骤2：获取聊天ID

要找到您的聊天ID，请使用RawDataBot。



usher
/start



RawDataBot
Chat ID : 5850249902
First Name : usher
Last Name : undefined
User Name : undefined



步骤3：配置Casdoor Telegram提供者

有三个必填字段：`App Key`，`Content`和`Chat ID`。字段与Telegram之间的关系如下：

名称	在Telegram中的名称
Secret key	API Token
Chat ID	Chat ID


名称	在Telegram中的名称
Content	

Name ⓘ: telegram

Display name ⓘ: telegram

Organization ⓘ: admin (Shared)

Category ⓘ: Notification

Type ⓘ:  Telegram

Secret key ⓘ: ***

Content ⓘ: test

Chat ID ⓘ: 5850249902 [Send Testing Notification](#)

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

自定义HTTP

📘 备注

Casdoor支持自定义HTTP通知提供者。您可以使用它向特定的HTTP地址发送消息。

配置Casdoor自定义HTTP提供者

有三个必填字段：`方法`，`参数名称`，`内容`和`聊天ID`。

名称	描述
方法	选择 <code>GET</code> 或 <code>POST</code> 方法。
参数名称	URL查询参数名称或主体参数，取决于 <code>方法</code> 。
内容	您想要发送的消息。
聊天ID	您的HTTP地址

Name ⓘ:	<input type="text" value="custom_http"/>
Display name ⓘ:	<input type="text" value="custon_http"/>
Organization ⓘ:	<input type="text" value="admin (Shared)"/>
Category ⓘ:	<input type="text" value="Notification"/>
Type ⓘ:	<input type="text" value="📧 Custom HTTP"/>
Method ⓘ:	<input type="text" value="POST"/>
Parameter ⓘ:	<input type="text" value="test"/>
Content ⓘ:	<input type="text" value="test"/>
Endpoint ⓘ:	<input type="text" value="http://localhost:12345"/> <input type="button" value="Send Testing Notification"/>
Provider URL ⓘ:	<input type="text" value="🔗 https://github.com/organizations/xxx/settings/applications/1234567"/>

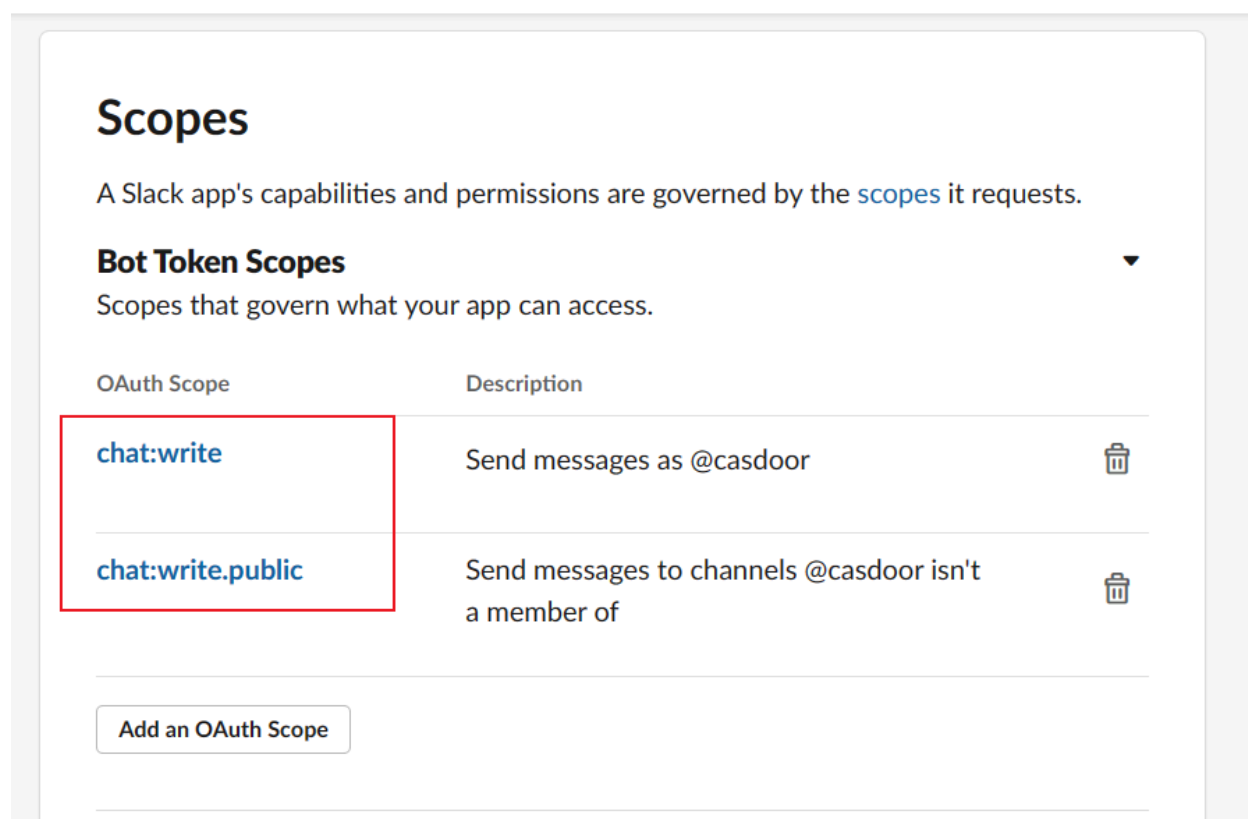
在我的示例中，当我点击 `发送通知消息` 时，我收到了这个请求：

```
Listening on :12345...
Received a request:
Method: POST
URL: /
Body: test
```

Slack

步骤1：配置Slack应用

首先，你需要在[Slack API](#)上创建一个应用。给你的机器人/应用授予以下OAuth权限范围：`chat:write`，`chat:write.public`



Scopes

A Slack app's capabilities and permissions are governed by the [scopes](#) it requests.

Bot Token Scopes ▼

Scopes that govern what your app can access.

OAuth Scope	Description	
<code>chat:write</code>	Send messages as @casdoor	🗑️
<code>chat:write.public</code>	Send messages to channels @casdoor isn't a member of	🗑️

[Add an OAuth Scope](#)

步骤2：获取机器人用户OAuth访问令牌和频道ID

复制你的 `Bot User OAuth Access Token` 以供下面使用。

Features

- App Home
- Org Level Apps
- Incoming Webhooks
- Interactivity & Shortcuts
- Slash Commands
- Workflow Steps
- OAuth & Permissions**
- Event Subscriptions
- User ID Translation
- App Manifest **NEW**
- Beta Features

Submit to App Directory

- Review & Submit

Give feedback

Slack ♥

- Help
- Contact
- Policies
- Our Blog

⚠ At least one redirect URL needs to be set below before this app can be opted into token rotation

Opt In

OAuth Tokens for Your Workspace

These tokens were automatically generated when you installed the app to your team. You can use these to authenticate your app. [Learn more.](#)

User OAuth Token

`xoxp-5865439759200-5827199080935-5865457291440-67810c12dfcae` **Copy**

Access Level: Workspace

Bot User OAuth Token

`xoxb-5865439759200-5865457299776-2tbOAVTNpG7vmLOg7OFJ55t` **Copy**

Access Level: Workspace

Reinstall to Workspace

Redirect URLs

复制你想要发布消息的频道的频道ID。你可以通过右键点击一个频道并选择 **复制名称** 来获取频道ID

The screenshot shows the Slack interface for a workspace named 'casdoor'. On the left sidebar, the 'Channels' list includes '# casdoor', '# general', and '# random'. A context menu is open over the '# casdoor' channel, with the 'Copy' option selected. A sub-menu is displayed over 'Copy', showing 'Copy name', 'Copy link', and 'Copy huddle link'. The main content area shows the 'About' page for the 'casdoor' channel, with the text 'You added casdoor to this workspace.' and a 'Configuration' button.

步骤3：配置Casdoor Slack提供者

有三个必填字段：`App Key`，`Content`，和`Chat ID`。字段与Slack之间的关系如下：


名称	Slack中的名称
Secret key	Access Token
Chat ID	Channel ID
Content	

Name ⓘ: slack

Display name ⓘ: slack

Organization ⓘ: admin (Shared)

Category ⓘ: Notification

Type ⓘ:  Slack

Secret key ⓘ: ***

Content ⓘ: dont reply

Chat ID ⓘ: casdoor [Send Testing Notification](#)

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

Google Chat

步骤1：获取应用默认凭据

为了将notify集成到Google Chat应用中，必须提供应用凭据。有关Google应用凭据JSON的更多信息，请参见：[应用默认凭据是如何工作的](#)

json将如下所示：

```
{
  "type": "service_account",
  "project_id": "",
  "private_key_id": "",
  "private_key": "",
  "client_email": "",
  "client_id": "",
  "auth_uri": "",
  "token_uri": "",
  "auth_provider_x509_cert_url": "",
  "client_x509_cert_url": ""
}
```

步骤3：配置Casdoor Google Chat提供者

在元数据中填写应用凭据。

Name ⓘ : google_chat

Display name ⓘ : google_chat

Organization ⓘ : admin (Shared)

Category ⓘ : Notification

Type ⓘ :  Google Chat

Metadata ⓘ :
{
 "type": "service_account",
 "project_id": "",
 "private_key_id": ""

Content ⓘ : test

Test Notification ⓘ : [Send Testing Notification](#)

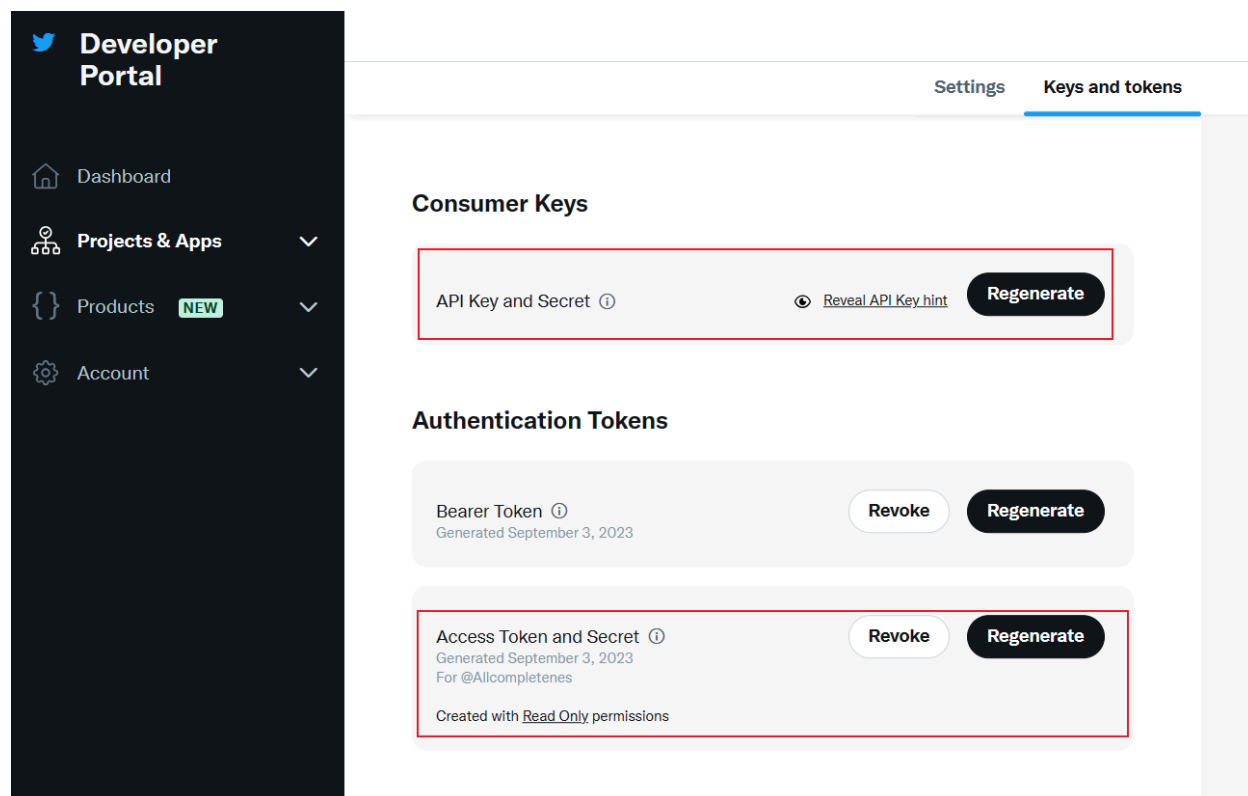
Provider URL ⓘ : <https://github.com/organizations/xxx/settings/applications/1234567>

Twitter

步骤1：从twitter获取配置项

首先，注册一个Twitter开发者账户，在开发者门户内创建一个Twitter应用，参考文档：[认证](#)

复制你的 API Key 和 API Secret， Access Token 和 Access Token Secret



步骤2：获取Twitter ID

Twitter ID 不能直接获取，你可以通过一些第三方工具获取。

- [TweeterID](#)
- [Twiteridfinder](#)

步骤3：配置Casdoor Twitter提供商

有五个必填字段：`Client ID`，`Client secret`，`Client ID 2`，`Client secret 2`和`Chat ID`。字段与Twitter之间的关系如下：

名称	Twitter中的名称
Client ID	API Key
Client secret	API Secret
Client ID 2	Access Token
Client secret 2	Access Token Secret
Chat ID	Twitter ID

Name ⓘ :

Display name ⓘ :

Organization ⓘ :

Category ⓘ :

Type ⓘ :

Client ID ⓘ :

Client secret ⓘ :

Client ID 2 ⓘ :

Client secret 2 ⓘ :

Content ⓘ :

Chat ID ⓘ :

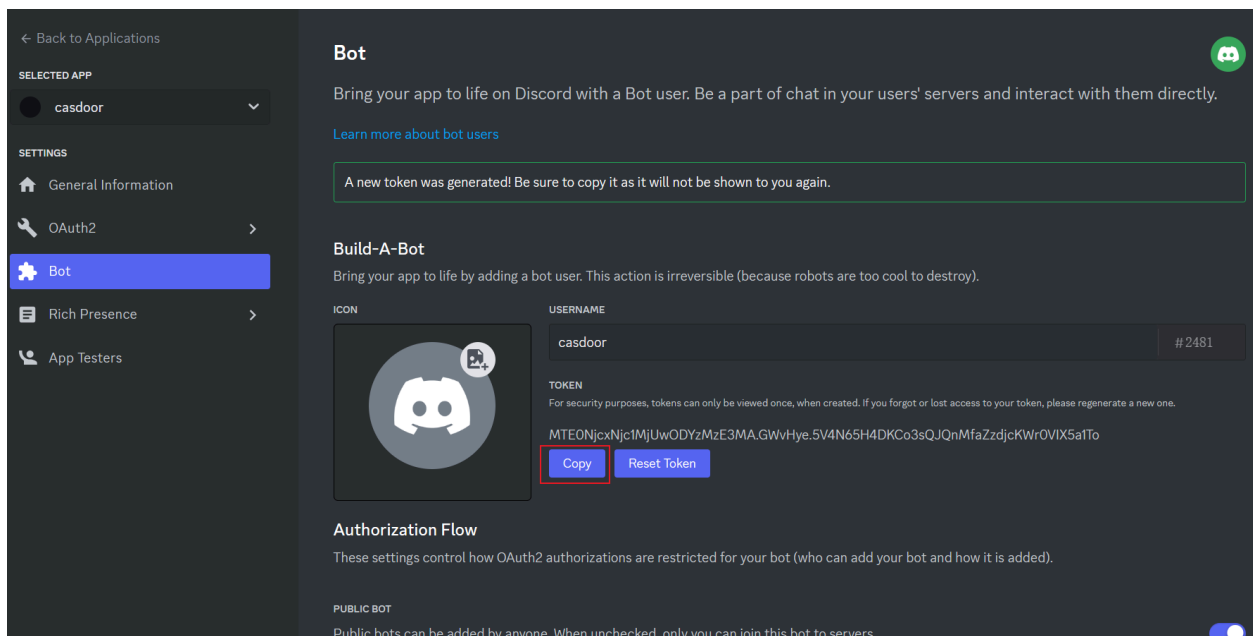
Provider URL ⓘ :

Discord

步骤1：从Discord获取令牌

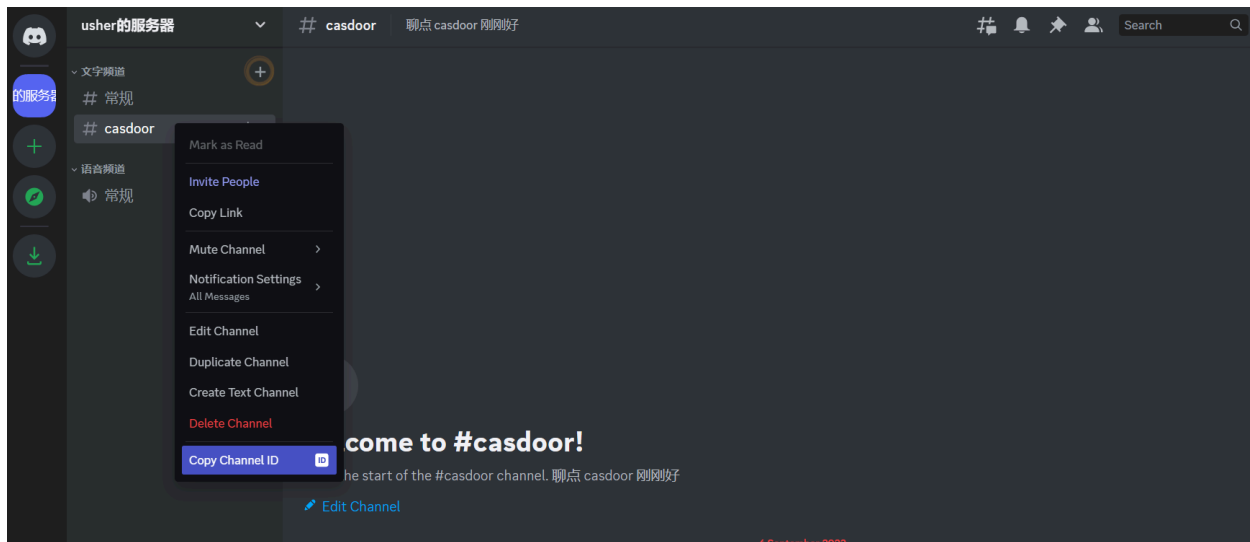
首先，注册Discord开发者门户，创建一个新的应用程序，导航到“Bot”选项卡进行配置。

复制你的Bot `token`



步骤2：获取频道ID

复制你想要发布消息的频道的频道ID。你可以通过右键点击频道并选择 `复制频道ID` 来获取频道ID



步骤3：配置Casdoor Discord提供者

有三个必填字段：`App Key`，`Content`，和`Chat ID`。字段与Discord之间的关系如下：

名称	在Slack中的名称
密钥	令牌
聊天ID	频道ID
内容	

Name ⓘ:	<input type="text" value="discord"/>
Display name ⓘ:	<input type="text" value="discord"/>
Organization ⓘ:	<input type="text" value="admin (Shared)"/>
Category ⓘ:	<input type="text" value="Notification"/>
Type ⓘ:	<input type="text" value="🗨️ Discord"/>
Secret key ⓘ:	<input type="text" value="***"/>
Content ⓘ:	<input type="text" value="test"/>
Chat ID ⓘ:	<input type="text" value="1146715329133821972"/> <input type="button" value="Send Testing Notification"/>
Provider URL ⓘ:	<input type="text" value="🔗 https://github.com/organizations/xxx/settings/applications/1234567"/>

存储

概述

在Casdoor中设置一个存储提供者以上传文件

本地文件系统

使用本地文件系统作为Casdoor的存储提供者

Amazon S3

使用Amazon S3作为Casdoor的存储提供者

Azure Blob

使用 Azure Blob 作为Casdoor的存储提供者

Google Cloud Storage

使用Google Cloud Storage作为Casdoor的存储提供商

MinIO

使用MinIO作为Casdoor的存储提供商

阿里云OSS

使用阿里云OSS作为Casdoor的存储提供商

腾讯云COS

使用腾讯云COS作为Casdoor的存储提供商

Synology NAS

使用Synology NAS作为Casdoor的存储提供商

概述

如果你需要使用文件存储服务，如“头像上传”，你需要在Casdoor中设置一个存储提供者并将其应用到你的应用程序中。

Casdoor支持两种类型的存储：**本地**和**云**。在本章中，你将学习如何添加一个存储提供者以使用这些服务。

项目

- **客户端ID**：由云存储提供者提供的唯一标识符。
- **客户端密钥**：只有Casdoor和云存储服务知道的安全值。
- **端点**：云存储服务的公共URL或域名。
- **端点（内网）**：云存储服务的内部或私有URL或域名。
- **路径前缀**：文件位置的路径前缀。

❗ 信息

默认的**路径前缀**是"/"。例如，当**路径前缀**为空时，一个默认的文件路径将是：

```
https://cdn.casbin.com/casdoor/avatar.png
```

你可以用"abcd/xxxx"填充它，然后你可以在以下位置存储你的头像：

```
https://cdn.casbin.com/abcd/xxxx/casdoor/avatar.png
```

- **存储桶**: 用于存储文件和数据的容器。
- **域名**: 你的云存储的CDN的自定义域名。
- **区域ID**: 用于指定云存储服务所在数据中心区域的标识符。

本地设置

我们支持将文件上传到本地系统。

云

我们支持AWS S3、Azure Blob Storage、MinIO、阿里云OSS、腾讯云COS，并且我们正在不断添加更多的云存储服务。

本地文件系统

❗ 信息

本地文件系统提供者将在Casdoor的 `files` 文件夹中存储您上传的文件。

例如，当您的Casdoor位于 `/home/user/casdoor` 目录时，上传的文件将存储在 `/home/user/casdoor/files` 文件夹中。

配置Casdoor提供者

Name [?](#): localfile

Display name [?](#): localfile

Organization [?](#): admin (Shared)

Category [?](#): Storage

Type [?](#): Local File System

Path prefix [?](#):

Domain [?](#): http://localhost:8000

Provider URL [?](#): [🔗](#)

`Path prefix` 是您文件位置路径的前缀。您可以根据需要填写。在以下示例中，您可

以看到带前缀和不带前缀的区别。

带前缀

Path prefix  :

 casdoor >  files >  images >  resource >  built-in >  admin >  withPrefix.png

无前缀

Path prefix  :

 casdoor >  files >  resource >  built-in >  admin >  withoutPrefix.png

Amazon S3

📘 备注

这是一个Amazon S3的例子。

创建安全凭证

按照文档：[管理访问密钥](#)在Amazon控制台中创建并保存您的 `access key` 和 `secret access key`。

配置您的存储桶

在您的存储桶权限选项中，取消勾选“阻止”选项并保存更改。

Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

Block public access to buckets and objects granted through *new* access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

Block public access to buckets and objects granted through *any* access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

Block public access to buckets and objects granted through *new* public bucket or access point policies

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

Block public and cross-account access to buckets and objects through *any* public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Cancel

Save changes


编辑对象所有权并检查ACLs已启用。

Object Ownership

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.



- ACLs disabled (recommended)**
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

- ACLs enabled**
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

 We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.

Object Ownership

- Bucket owner preferred**
If new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.
- Object writer**
The object writer remains the object owner.

 If you want to enforce object ownership for new objects only, your bucket policy must specify that the bucket-owner-full-control canned ACL is required for object uploads. [Learn more](#) 

Cancel

Save changes

配置Casdoor

名称	在Amazon中的名称	是否必需
Client ID	Access key	必需
Client secret	Secret access key	必需
Endpoint	Endpoint	必需
Endpoint (intranet)	VPC endpoint	

名称	在Amazon中的名称	是否必需
Bucket	Bucket name	必需
Path prefix		
Domain	CloudFront domain	
Region ID	AWS region	必需

填写必要的信息，包括从上一步中 `access key` 和 `secret access key` 获取的 `Client ID` 和 `Client Secret`。您可以参考此文档以获取有关 `endpoint` 格式的信息：[网站端点](#)

Name ? :	awss3
Display name ? :	awss3
Organization ? :	admin (Shared)
Category ? :	Storage
Type ? :	AWS S3
Client ID ? :	AKIAYOMMXHVZC5CHBPNR
Client secret ? :	***
Endpoint ? :	http://kininaru.s3-website.ap-northeast-1.amazonaws.com
Endpoint (Intranet) ? :	
Bucket ? :	kininaru
Path prefix ? :	
Domain ? :	
Region ID ? :	ap-northeast-1
Provider URL ? :	🔗

(可选) 使用VPC

您可以参考官方文档进行配置: [通过AWS PrivateLink访问AWS服务](#)

(可选) 使用CloudFront分发

按照文档配置CloudFront: [配置CloudFront](#)

在域名字段中, 输入您的分发域名。

Endpoint  :

Bucket  :

Path prefix  :

Domain  :

Region ID  :

Provider URL  :

Azure Blob

📘 备注

这是一个Azure Blob的例子。

- 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。

The screenshot shows the configuration form for a new provider in Casdoor. The form includes the following fields and options:

- Edit Provider** | **Save** | **Save & Exit**
- Name**: provider_ftzes
- Display name**: New Provider - ftzes
- Category**: Storage (highlighted with a red box)
- Type**: Azure Blob (selected in a dropdown menu)
- Client ID**: Local File System, AWS S3
- Client secret**: Aliyun OSS, Tencent Cloud COS
- Endpoint**: Azure Blob (highlighted with a red box)

步骤2：在Casdoor中填写必要的信息

需要填写四个必填字段：`Client ID`、`Client secret`、`Endpoint`和`Bucket`。与 Azure Blob账户的对应关系如下：

字段名称	Azure 名称	必需的
Client ID	AccountName	必需
Client secret	AccountKey	必需的
Endpoint	ContainerUrl	必需的
Endpoint (intranet)	PrivateEndpoint	
Bucket	ContainerName	必需的
Path prefix		
Domain	DomainName	

- 账户名称

`AccountName` 就是你的账户名。

- AccountKey

`AccountKey` 是您访问 Azure API 的密钥。

i 备注

您可以从 Azure 门户的 "Access Keys" 部分获取您的账户密钥，该部分位于您的存储账户的左侧面板上。

casbin | Access keys ☆ ...
Storage account

Search << ⌚ Set rotation reminder 🔄 Refresh 🗨 Give feedback

Storage browser
Storage Mover

Data storage

Containers
File shares
Queues
Tables

Security + networking

Networking
Azure CDN
Access keys
Shared access signature
Encryption
Microsoft Defender for Cloud

Data management

Access keys authenticate your applications' requests to this storage account. Keep your keys in a secure location like Azure Key Vault, and replace them often with new keys. The two keys allow you to replace one while still using the other.

Remember to update the keys with any Azure resources and apps that use this storage account.
[Learn more about managing storage account access keys](#)

Storage account name
casbin

key1 🔄 Rotate key
Last rotated: 2023/7/22 (0 days ago)

Key
..... Show

Connection string
..... Show

key2 🔄 Rotate key
Last rotated: 2023/7/22 (0 days ago)

Key
..... Show

Connection string

- ContainerUrl

您可以从容器属性中获取ContainerUrl。

default | Properties

Container



Refresh



Give feedback

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

Shared access tokens

Access policy

Properties

Metadata

NAME

default

URL

https://casbin.blob.core.windows.net/default

LAST MODIFIED

7/22/2023, 5:18:03 PM

ETAG

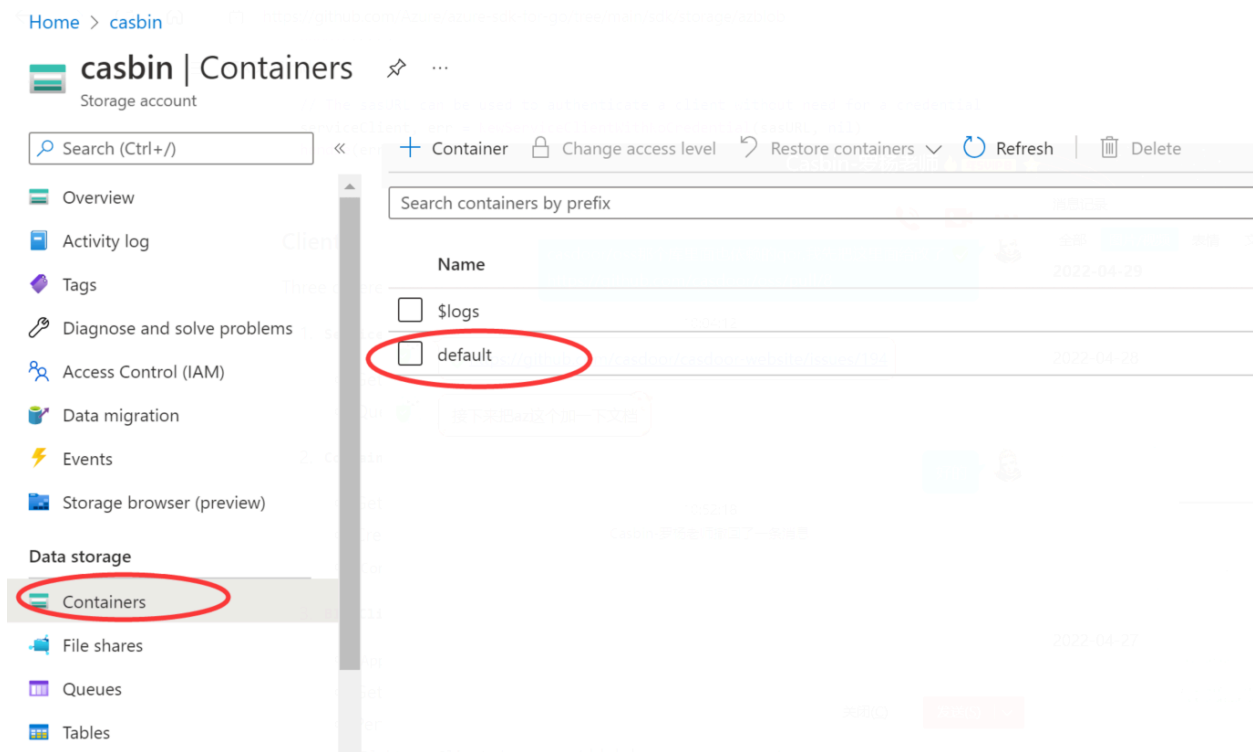
0x8DB8A948D644055

- (可选) PrivateEndpoint

Azure Private Endpoint是一个功能，允许将Azure服务连接到Azure虚拟网络（VNet）的私有子网。您可以参考官方Azure文档进行配置：[私有端点配置](#)

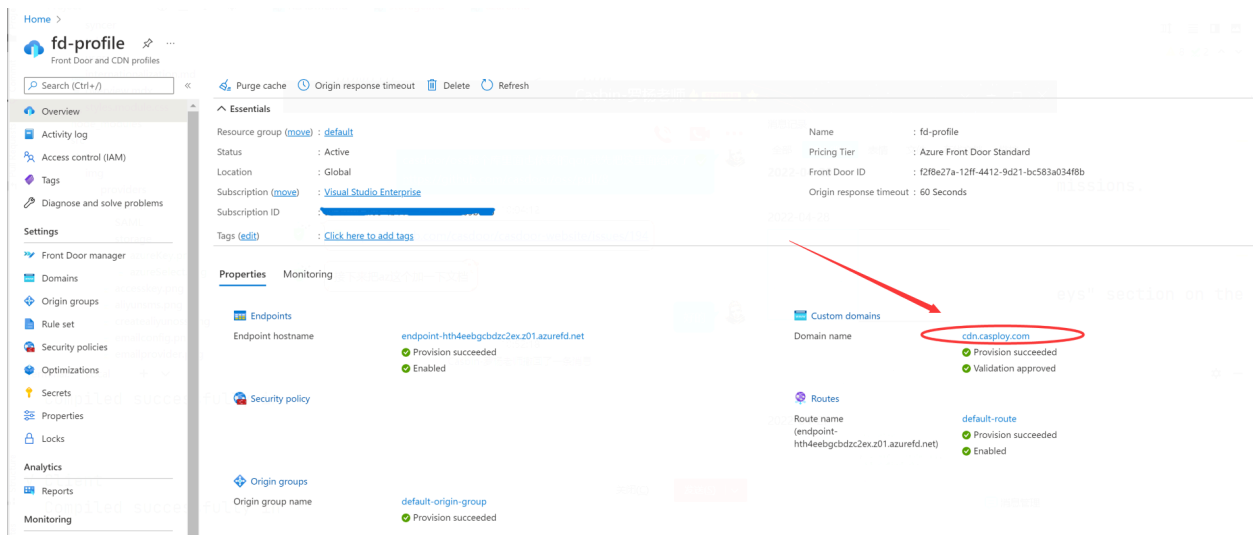
- ContainerName

在这个例子中，创建了一个名为'default'的默认容器。



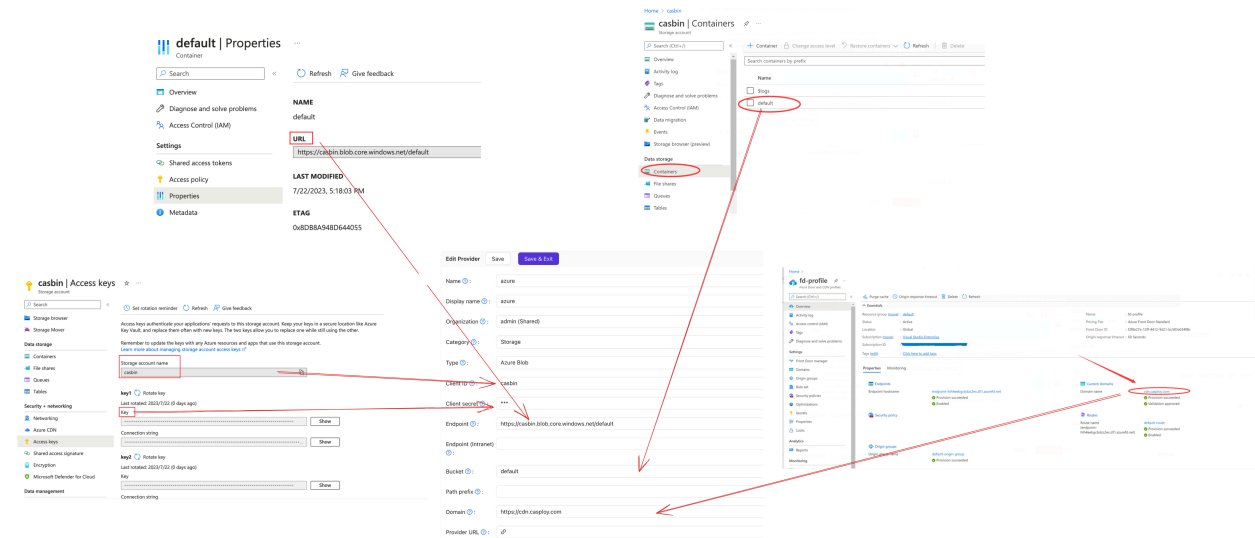
- (可选) 域名

在您的Azure CDN中的自定义域名。



步骤3：保存您的配置

最终结果如下：



现在您可以在您的应用程序中使用Azure Blob存储服务。

Google Cloud Storage

📘 备注

这是一个Google Cloud Storage的例子。

创建安全凭证

按照文档：[Cloud Storage Authentication](#) 在GCP控制台中创建一个具有正确的 **IAM权限** 来访问bucket的 **服务账户**。

配置Casdoor

名称	Google中的名称	是否必需
Service Account JSON	Service Account Key	必需
Endpoint	Endpoint	
Bucket	Bucket name	必需

Name ⓘ:

Display name ⓘ:

Organization ⓘ:

Category ⓘ:

Type ⓘ:

Service account JSON ⓘ:

Endpoint ⓘ:

Bucket ⓘ:

Path prefix ⓘ:

Provider URL ⓘ:

MinIO

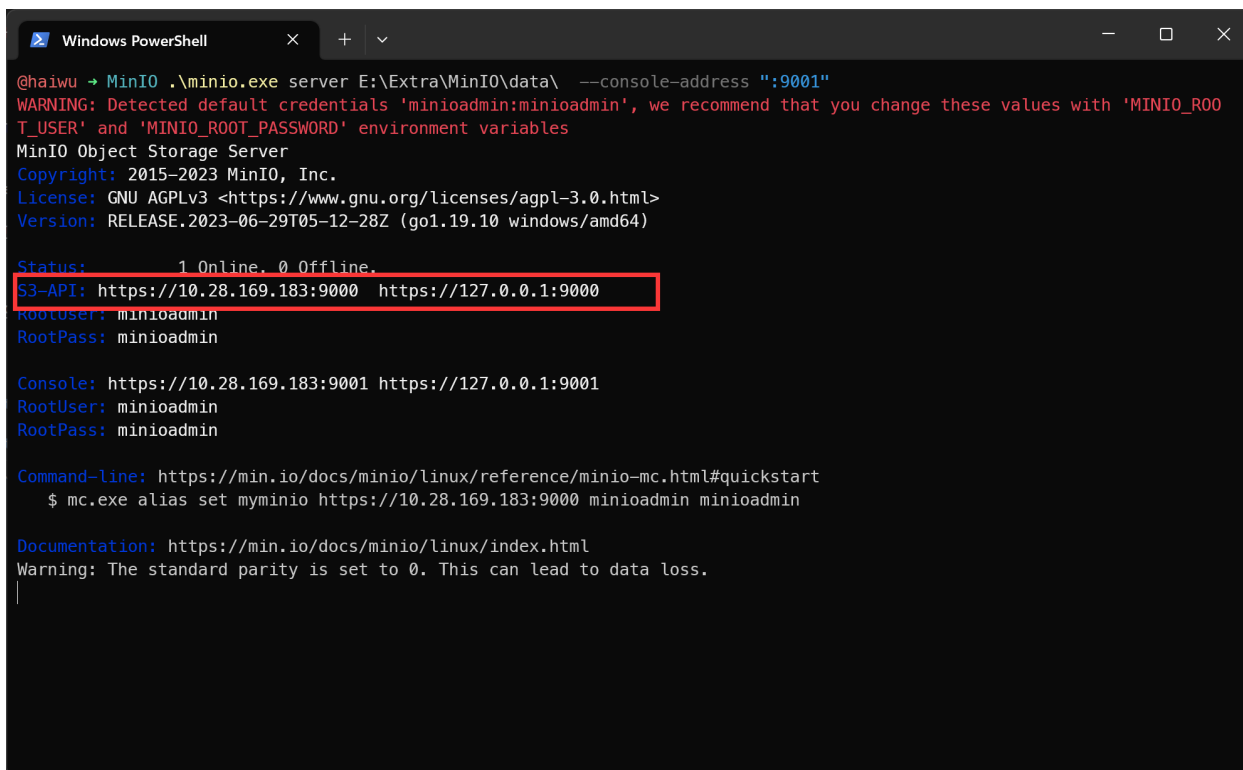
📌 备注

这是一个如何配置MinIO提供商的示例。

MinIO是一个与Amazon S3云存储服务API兼容的高性能对象存储服务。

步骤1：部署MinIO服务

首先，部署启用TLS的MinIO服务。您可以从控制台获取API地址。



```
Windows PowerShell
@haiwu → MinIO .\minio.exe server E:\Extra\MinIO\data\ --console-address ":9001"
WARNING: Detected default credentials 'minioadmin:minioadmin', we recommend that you change these values with 'MINIO_ROOT_USER' and 'MINIO_ROOT_PASSWORD' environment variables
MinIO Object Storage Server
Copyright: 2015-2023 MinIO, Inc.
License: GNU AGPLv3 <https://www.gnu.org/licenses/agpl-3.0.html>
Version: RELEASE.2023-06-29T05-12-28Z (go1.19.10 windows/amd64)

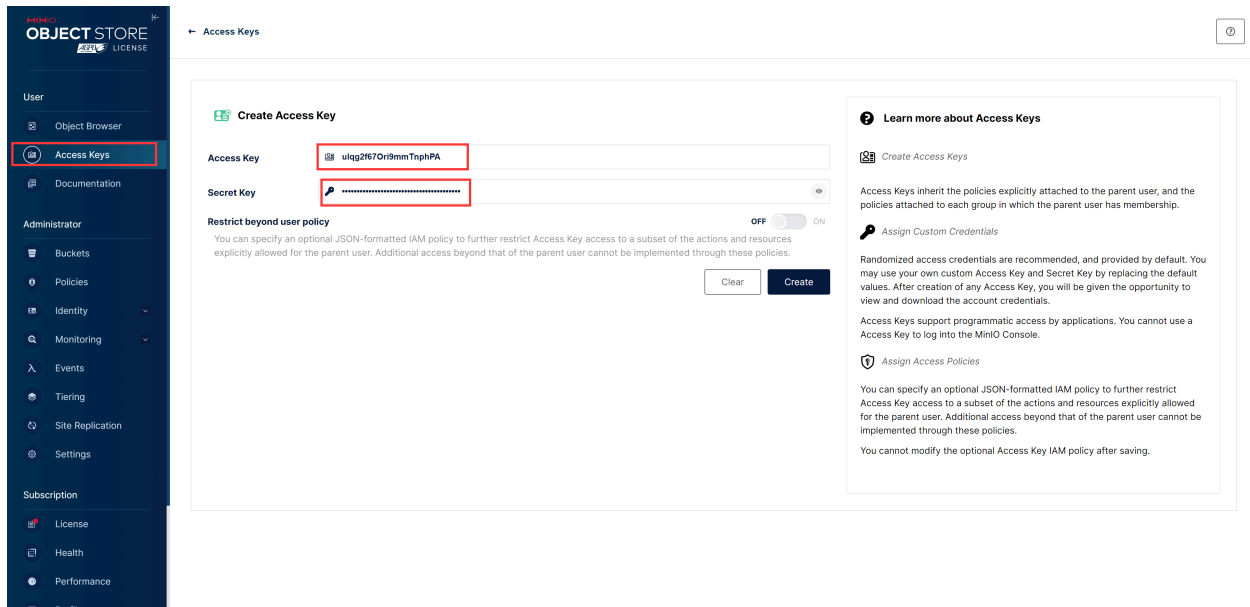
Status:          1 Online, 0 Offline.
S3-API: https://10.28.169.183:9000 https://127.0.0.1:9000
RootUser: minioadmin
RootPass: minioadmin

Console: https://10.28.169.183:9001 https://127.0.0.1:9001
RootUser: minioadmin
RootPass: minioadmin

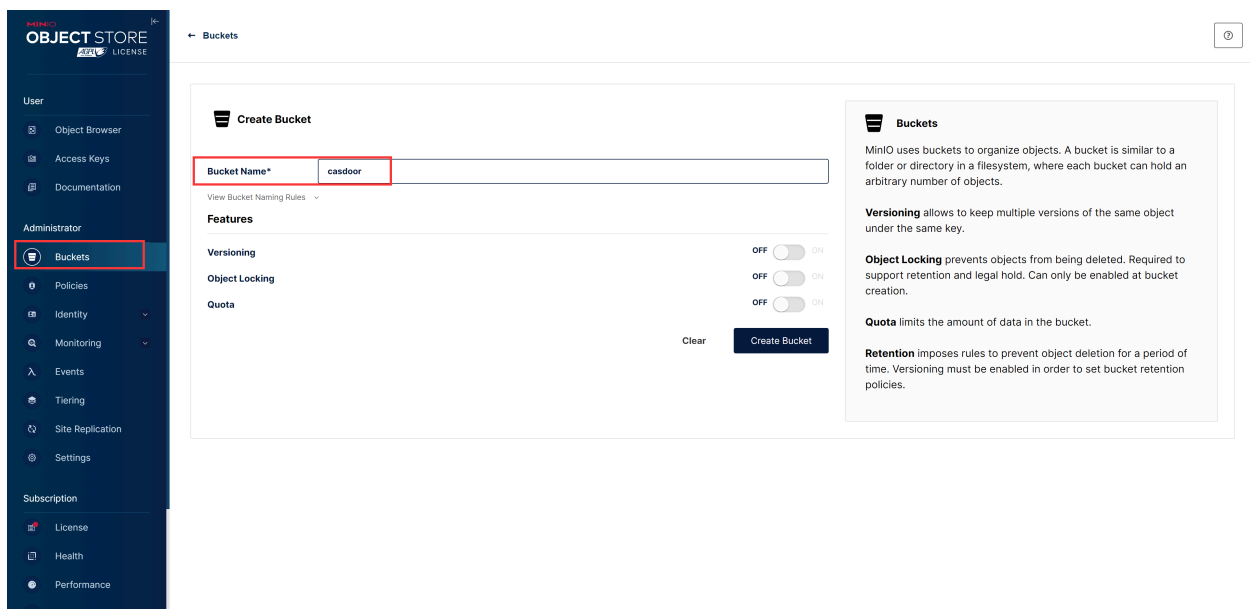
Command-line: https://min.io/docs/minio/linux/reference/minio-mc.html#quickstart
$ mc.exe alias set myminio https://10.28.169.183:9000 minioadmin minioadmin

Documentation: https://min.io/docs/minio/linux/index.html
Warning: The standard parity is set to 0. This can lead to data loss.
```

其次，创建Access Key和Secret key。



第三，创建Bucket。



步骤2：在Casdoor中创建一个MinIO提供商

现在在Casdoor中创建一个MinIO提供商。填写必要的信息。

名称	在MinIO中的名称
Category	选择 <code>Storage</code>
Type	选择 <code>MinIO</code>
Client ID	从步骤1获取的 <code>Access Key</code>
Client secret	从步骤1获取的 <code>Secret Key</code>
Endpoint	从步骤1获取的 <code>API地址</code>
Bucket	从步骤1获取的 <code>Bucket</code>

步骤3：在您的应用程序中使用MinIO存储服务

现在您可以在您的应用程序中使用MinIO存储服务。

阿里云OSS

📘 备注

这是一个阿里云OSS的例子。

AccessKey是您用于以完全账户权限访问阿里云API的密钥。

要创建AccessKey，请按照[阿里云工作台](#)中的说明操作。

接下来，创建OSS服务：

创建 Bucket

🔍 创建存储空间 ✕

⚠️ 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。

Bucket 名称	<input type="text" value="mycasdoor"/>	9/63 ✓
地域	<input type="text" value="华北2 (北京)"/>	▼
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。		
Endpoint	oss-cn-beijing.aliyuncs.com	

在Casdoor中填写必要的信息并保存：

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

您现在可以在您的应用程序中使用阿里云云存储服务。

腾讯云COS

📘 备注

这是一个腾讯云COS的例子。

在Casdoor中填写必要的信息

有五个必填字段：`客户端ID`，`客户端密钥`，`端点`，`存储桶`，和`区域ID`。与腾讯云COS账户的对应关系如下：

名称	在腾讯的名称	必需的
Client ID	SecretId	是
Client secret	SecretKey	是
Endpoint	Endpoint	是
Bucket	BucketName	是
Path prefix		
Domain	CDNDomain	
Region ID	Region	是

腾讯云COS信息

- SecretId和SecretKey

The screenshot shows the 'API密钥管理' (API Key Management) page in the Tencent Cloud console. It features a sidebar with navigation options like '访问管理' (Access Management) and 'API密钥管理'. The main content area includes a '安全提示' (Security Notice) and a '使用提示' (Usage Notice). Below these is a table of API keys:

APPID	密钥	创建时间	最近访问时间	状态
1319606438	SecretId: AKIDdAlMuNrJn8GHI6mLI6NSWbheNr7MVeic SecretKey: *****显示	2023-07-22 19:01:...	2023-07-22 22:09	已启用

- 端点, BucketName和区域

The screenshot displays the 'casdoor-1319606438' bucket details page in the Tencent Cloud console. It includes a '用量概览' (Usage Overview) section with metrics for object count, storage, and traffic. The '基本信息' (Basic Information) section shows the bucket name, region, and creation time. The '域名信息' (Domain Information) section lists the access domain and other settings.

用量概览

对象数量	存储量	本月总流量	本月总请求数
4个	0 B	3.72 KB	126次
较昨天 ↑ 0.00%	较昨天 ↑ 0%	上月总流量 0 B	上月总请求数 0次
较上月同期 ↑ 0%	较上月同期 ↑ 0%		

基本信息

存储桶名称	casdoor-1319606438 (存储桶不支持改名)
所属地域	广州 (中国) ap-guangzhou
创建时间	2023-07-22 18:57:50
访问权限	私有读写

域名信息

访问域名	https://casdoor-1319606438.cos.ap-guangzhou.myqcloud.com
自定义CDN加速域名	0条
自定义源站域名	0条
全球加速域名	未开启
静态网站域名	未开启

- (可选) CDNDomain

您可以参考官方文档进行配置：[配置CDN](#)

配置Casdoor提供商

The image shows two screenshots from the Tencent Cloud console. The left screenshot displays the configuration form for a provider, with red arrows pointing from the 'API Key Management' table in the right screenshot to the corresponding fields in the form. The right screenshot shows the 'API Key Management' page with a table of keys and the 'Bucket List' page for the bucket 'casdoor-1319606438'.

API Key Management Table:

APPID	密钥	创建时间	最近使用时间	状态
casdoor-1319606438	SecretId AKID5A8AAUJh8GH8mLj8NSWbhc7Mw6c SecretKey *****	2023-07-22 19:01:...	2023-07-22 22:09	已启用

Provider Configuration Fields:

- Name: tencentcos
- Display name: tencentcos
- Organization: admin (Shared)
- Category: Storage
- Type: Tencent Cloud COS
- Client ID: AKID5A8AAUJh8GH8mLj8NSWbhc7Mw6c
- Client secret: ***
- Endpoint: casdoor-1319606438.cos.ap-guangzhou.myqcloud.com
- Bucket: casdoor-1319606438
- Path prefix:
- Domain:
- Region ID: ap-guangzhou
- Provider URL: ip

Bucket List Table:

桶名称	创建时间	所有者
casdoor-1319606438	2023-07-22 18:57:50	ap-guangzhou

Synology NAS

📘 备注

这是一个Synology NAS的例子。

在Casdoor中填写必要的信息

有五个必填字段：`Client ID`、`Client secret`和`Endpoint`。与Synology NAS账户的对应关系如下：

名称	在腾讯中的名称	必需
Client ID	SecretId	是
Client secret	SecretKey	是
Endpoint	Endpoint	是
Bucket		
Path prefix		
Domain		
Region ID		

配置Casdoor提供商

The image shows the configuration of a Synology provider in Casdoor. On the left is the 'Edit Provider' form, and on the right are two windows: 'Personal' and 'Synology Assistant'.

Casdoor Edit Provider Form:

- Name: provider_synology
- Display name: New Provider - synology
- Organization: built-in
- Category: Storage
- Type: Synology
- Client ID: password_is_slackonglong
- Client secret: ***
- Endpoint: http://169.254.0.2:5000
- Bucket: (empty)
- Path prefix: (empty)
- Provider URL: https://github.com/organizations/you/settings/applications/1234567

Personal Window:

- Name: password_is_slackonglong (AccessID)
- New Password: (masked) (AccessKey)
- Confirm password: (masked)
- Display language: System default

Synology Assistant Window:

服务器名称	IP 地址	IP 状态	状态	网络物理地址	版本	型号	序列号	WOL
synologyNAS	169.254.0.2	Manual	已就绪	00:11:32:2C:A6:03	6.1.7-15284	DS3617xs	A80DN02468	--

Endpoint: http://169.254.0.2:5000(default http/https port of Synology is 5000/5001)

您可以参考官方文档进行配置：[链接](#)

SAML

概述

使用来自支持SAML 2.0 的外部身份提供商的身份

自定义

配置您的SAML自定义提供商

Keycloak

使用 Keycloak 验证用户




阿里巴巴云IDaaS

使用阿里巴巴云IDaaS进行用户身份验证

概述

Casdoor可以配置为支持用户使用支持SAML 2.0的外部身份提供商的身份登录到UI。在此配置中，Casdoor从不存储任何用户的凭证。

现在，Casdoor支持多个SAML应用程序提供商。在添加到Casdoor后，提供商的图标将在登录页面上显示。以下是Casdoor支持的提供商：

阿里云IDaaS	Keycloak	自定义
		
✓	✓	✓

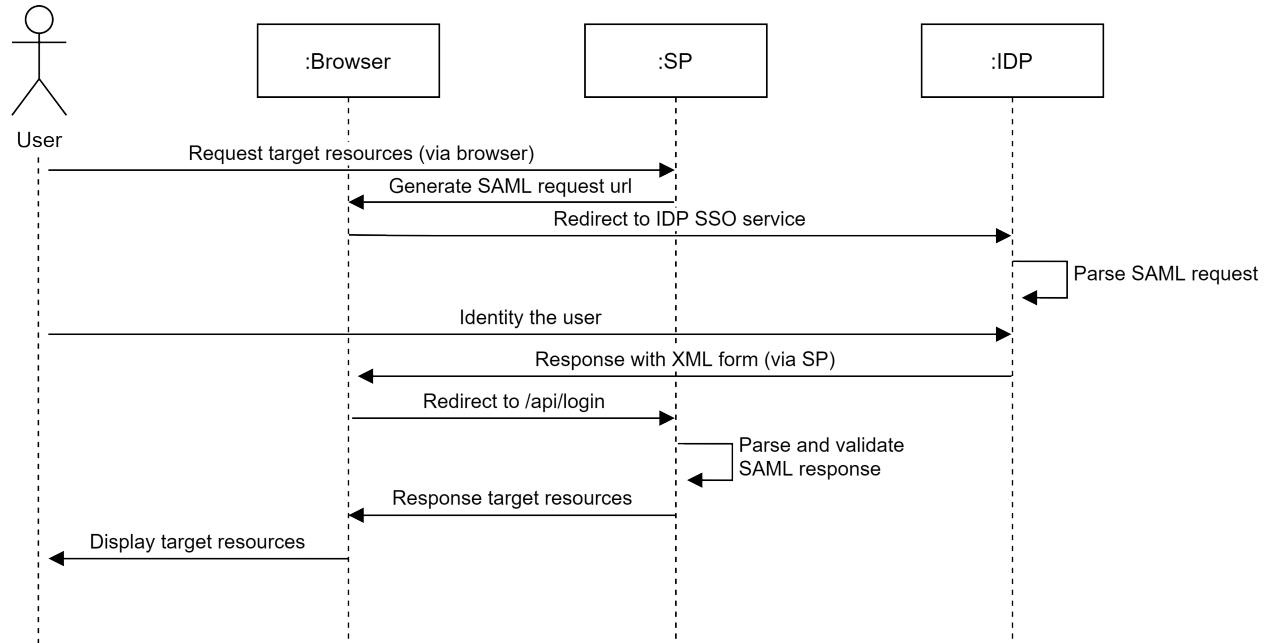
条款

- 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）- 向终端用户提供资源的服务，在这种情况下，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，用户通过身份提供商登录Casdoor，而无需将凭证传递给

Casdoor。进展情况见下图表。



自定义

Casdoor支持配置SAML自定义提供商，您可以使用Casdoor作为服务提供商（SP）连接任何支持SAML 2.0协议的身份提供商（IDP）。

步骤1. 获取IDP的元数据

首先，您需要获取IDP的元数据，这是一个用于描述IDP提供的服务的配置信息的XML文档。它需要包括 `EntityID`、`SSO Endpoint` 等信息。

一些IDP，如Keycloak，需要SP信息来提供元数据。您可以参考文档[Keycloak](#)。

您可以使用oktadev来测试SAML自定义提供商，这是[元数据](#)。


步骤2. 配置SAML自定义提供商


获取IDP的元数据后，创建一个SAML自定义提供商并填写必要的信息。


字段	描述
Category	选择 <code>SAML</code>
Type	选择 <code>自定义</code>
Favicon.URL	IDP logo的URL


字段	描述
Metadata	IDP的元数据


然后点击 **解析** 按钮，**Endpoint**、**IdP**、**Issuer URL**、**SP ACS URL** 和 **SP Entity ID** 将被自动解析。







Name :




Display name :


Organization :


Category :


Type : Custom


User mapping : ID :
 Username :
 Display name :
 Email :
 Avatar :


Favicon : URL :
 Preview: 


Client ID :


Client secret :


Sign request :


Metadata :


Endpoint :

IdP :

Issuer URL :

SP ACS URL :

SP Entity ID :

Provider URL :

最后，将SAML自定义提供商添加到应用程序的 Providers。

Providers AMM

Name	Category	Type	Can signup	Can signin	Can unlink	Prompted	Rule	Action
provider_storage_minio_s3	Storage							^ v □
provider_oauth_lark	OAuth		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		^ v □
provider_email_qq	Email							^ v □
provider_web3_metamask	Web3		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		^ v □
provider_google_oauth	OAuth		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	One Tap	^ v □
provider_web2_onboard	Web2		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		^ v □
saml_custom_provider_okta	SAML							^ v □
saml_custom_provider_keycloak	SAML							^ v □

Keycloak

JBoss [Keycloak](#) 系统是一个广泛使用的开源身份管理系统，支持通过SAML和OpenID Connect与应用程序集成。它还可以作为其他提供商（如LDAP或其他SAML提供商）和支持SAML或OpenID Connect的应用程序之间的身份代理运行。

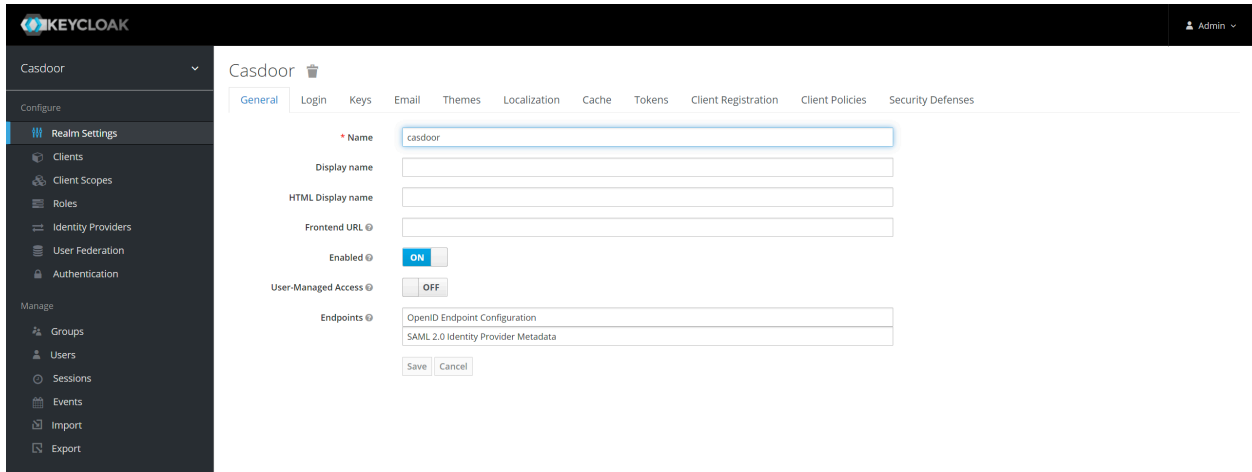
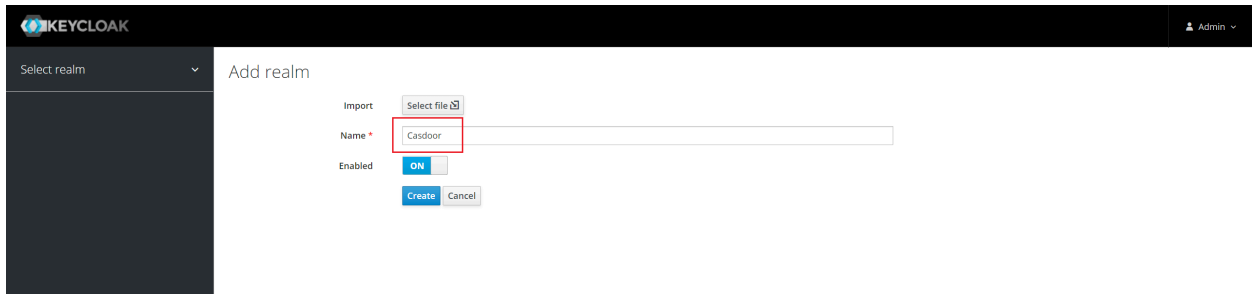
这是一个示例，说明如何在Keycloak中配置新的客户端条目，并配置Casdoor使用它，以允许通过Keycloak配置授予访问权限的Keycloak用户进行UI登录。

Configure Keycloak

对于这个例子，让我们做出以下配置选择和假设：

- 假设你正在本地以开发模式运行Casdoor。Casdoor UI可在 `http://localhost:7001` 处获取，服务器可在 `http://localhost:8000` 处获取。根据需要替换为适当的URL。
- 假设你正在本地运行Keycloak。Keycloak UI可在 `http://localhost:8080/auth` 处获得。
- 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`。

您可以使用默认的区域，或者创建一个新的区域。



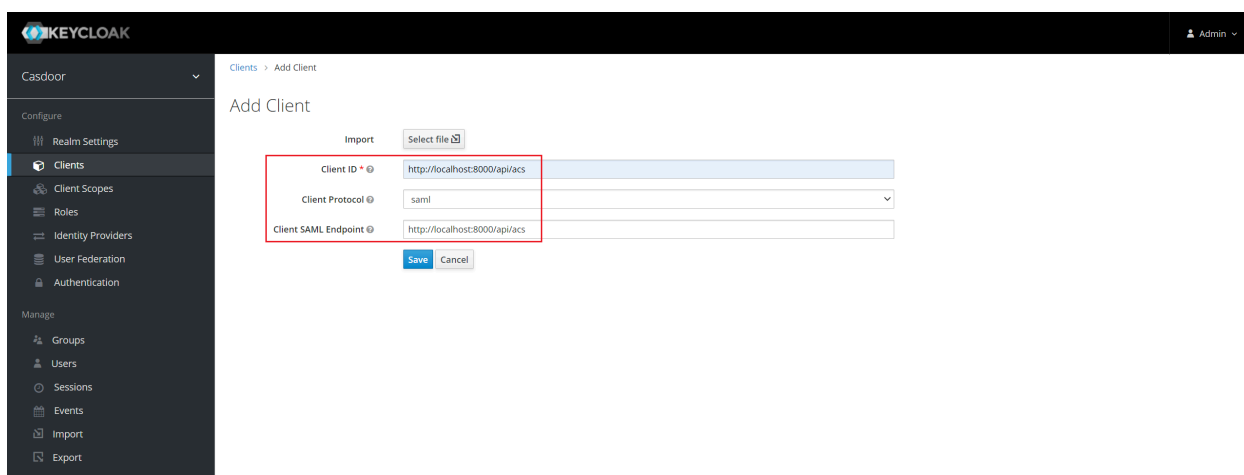
在 Keycloak 中添加客户端条目

❗ 信息

有关Keycloak Clients的更多详细信息，请参阅[Keycloak文档](#)。

在菜单中点击 **客户端** 然后点击 **创建** 去到 **添加客户端** 页面。按照以下方式填写字段：

- **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` .
- **客户端SAML端点:** `http://localhost:8000/api/acs` - 这个URL是你希望 Keycloak服务器发送SAML请求和响应的地方。通常，应用程序有一个用于处理 SAML请求的URL。客户端的设置选项卡中可以设置多个URL。



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

以下是设置的一部分：

1. **名称** - **Casdoor**。这只用于在Keycloak用户界面中向Keycloak用户显示友好的名称。您可以使用任何您喜欢的名字。
2. **已启用** - 选择 **on**。
3. **包含 Authn 语句** - 选择 **on**。
4. **签署文件** - 选择 **on**。
5. **签名断言** - 选择 **off**。
6. **加密断言** - 选择 **off**。
7. **需要客户签名** - 请选择 **off**。
8. **强制名称ID格式** - 选择 **on**。
9. **名称 ID 格式** - 选择 **username**。
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 displays the Keycloak Admin Console interface. The top navigation bar includes the Keycloak logo and the user 'Admin'. The left sidebar shows the navigation menu with 'Clients' selected. The main content area is titled 'Http://localhost:8000/api/acs' and contains various configuration tabs: Settings, Roles, Client Scopes, Mappers, Scope, Sessions, Offline Access, Clustering, and Installation. The 'Settings' tab is active, showing a list of configuration options for the client 'Casdoor'. Each option has a text input field and a toggle switch. The 'Enabled' toggle is turned on. Below the main settings, there are sections for 'Fine Grain SAML Endpoint Configuration' and 'Advanced Settings', both currently collapsed. At the bottom of the configuration area, there are 'Save' and 'Cancel' buttons.

Client ID: http://localhost:8000/api/acs
Name: Casdoor
Description:
Enabled: ON
Always Display in Console: OFF
Consent Required: OFF
Login Theme:
Client Protocol: saml
Include AuthStatement: 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:
Valid Redirect URIs: http://localhost:8000/api/acs
Base URL:
Master SAML Processing URL: http://localhost:8000/api/acs
IDP Initiated SSO URL Name:
IDP Initiated SSO Relay State:
Fine Grain SAML Endpoint Configuration
Assertion Consumer Service POST Binding URL: http://localhost:8000/api/acs
Assertion Consumer Service Redirect Binding URL: http://localhost:8000/api/acs
Logout Service POST Binding URL:
Logout Service Redirect Binding URL:
Logout Service ARTIFACT Binding URL:
Artifact Binding URL:
Artifact Resolution Service:
Advanced Settings
Authentication Flow Overrides
Save Cancel

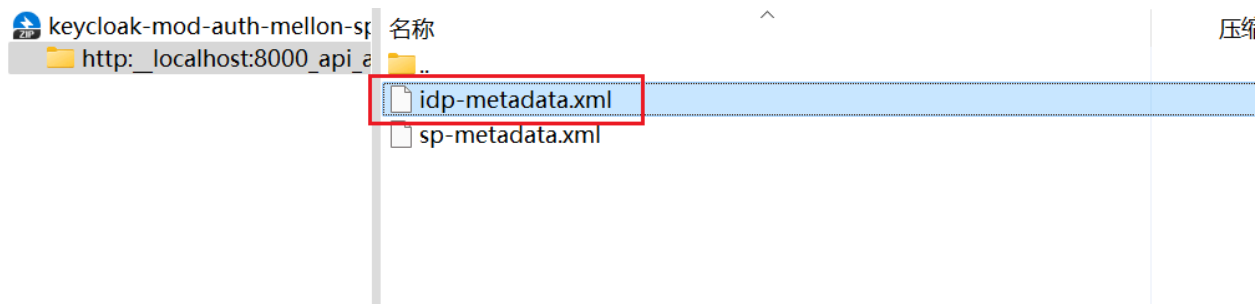
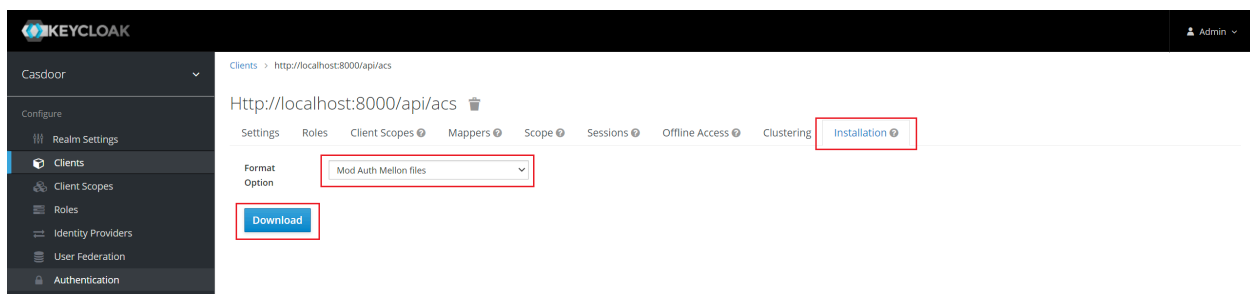
💡 提示

如果您想要签署authn请求，您需要启用**客户端签名要求**选项，并上传自己生成的证书。在Casdoor中使用的私钥和证书，`token_jwt_key.key` 和 `token_jwt_key.pem`，位于`object`目录中。在Keycloak中，您需要点击**密钥**选项卡，点击**导入**按钮，选择**档案格式**为**证书PEM**，并上传证书。

点击 **安装** 标签页。

对于 Keycloak <= 5.0.0，选择格式选项 - **SAML Metadata IDPSSODescriptor** 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**。解压下载的.zip文件，找到 `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**. 将metadata的内容复制并粘贴到**元数据**字段中。点击**解析**按钮后, **端点**、**IdP**和**发行者URL**的值将自动生成。最后, 点击**保存**按钮。

💡 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。

The screenshot shows the configuration form for a new provider in Casdoor. The fields are as follows:

- Name: keycloak-casdoor
- Display name: keycloak-casdoor
- Category: SAML
- Type: Keycloak
- Client ID: (empty)
- Client secret: (empty)
- Sign request: (toggle off)
- 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/realms/casdoor"><md:IDPSSODescriptor WantAuthnRequestsSigned="true" protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol"><md:KeyDescriptor use="signing"><ds:KeyInfo><ds:KeyName>zqm-3k76jG-na5Zc3uIPD17bp-4wYtMbWMPzwqUHAY</ds:KeyName><ds:X509Data><ds:X509Certificate>MIICnTCCAYUCBgF9pAmxSDANBgkqhkiG9w0BAQsFADASMRAdDgYDVQQDDAdjYXNkb29yMB4XDTEyMTIxMDEwMDg1OjEwXDTMxMTIxMDEwMTAzOFowEjEQMA4GA1UEAwwHY2FzZG9vcjCCASlwDQYJKoZIhvcNAQk</ds:X509Certificate>
```
- Parse: (button)
- Endpoint: http://localhost:8080/auth/realms/casdoor/protocol/saml
- IdP: MIIcTCCAYUCBgF9pAmxSDANBgkqhkiG9w0BAQsFADASMRAdDgYDVQQDDAdjYXNkb29yMB4XDTEyMTIxMDEwMDg1OjEwXDTMxMTIxMDEwMTAzOFowEjEQMA4GA1UEAwwHY2FzZG9vcjCCASlwDQYJKoZIhvcNAQk</ds:X509Certificate>
- Issuer URL: http://localhost:8080/auth/realms/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。选择你刚刚添加的提供商, 然后点击**保存**按钮。

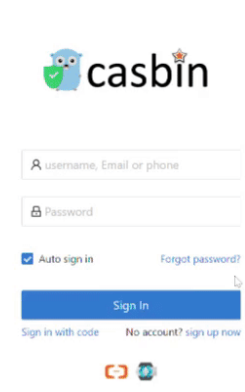
The screenshot shows the 'Providers' section in Casdoor. It contains a table with the following data:

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, 您会在登录页面上找到一个Keycloak图标。

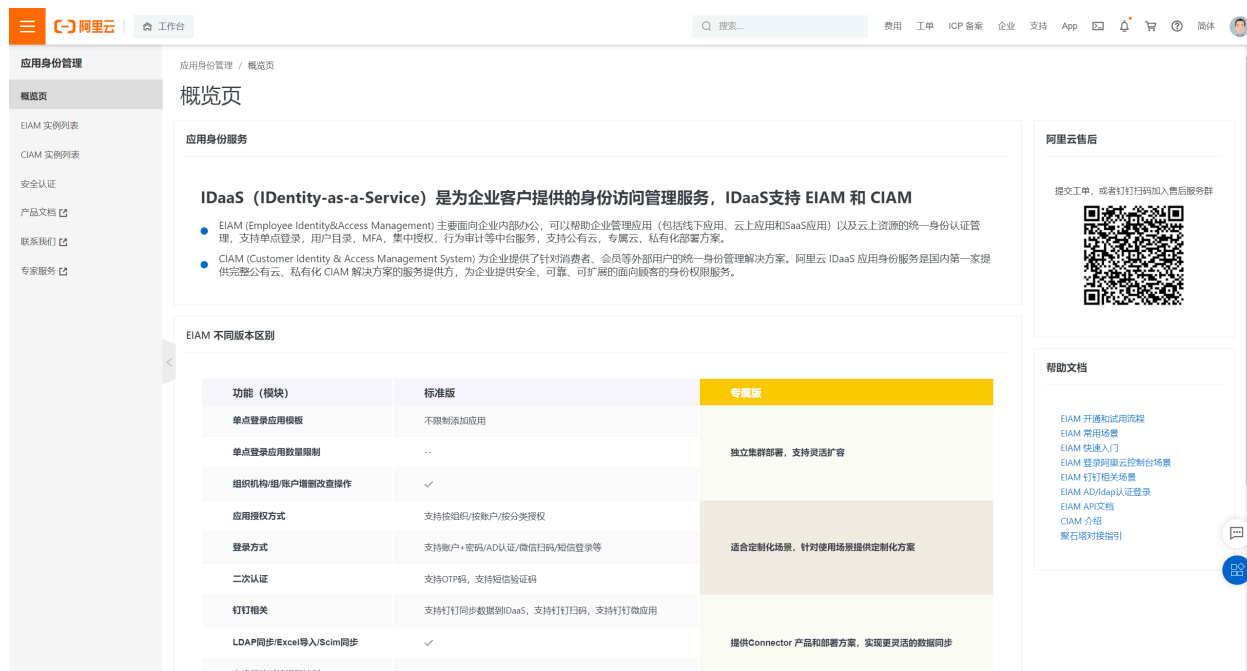
点击图标，您将被重定向到Keycloak登录页面。成功认证后，您将登录到Casdoor。



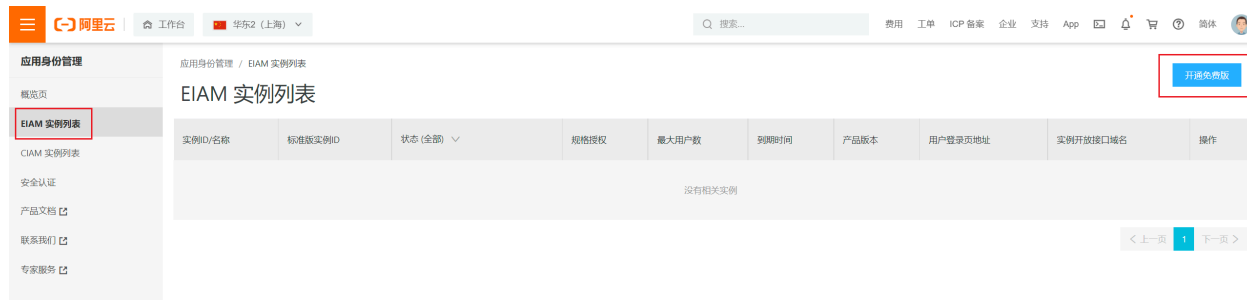
阿里云IDaaS

在阿里云IDaaS中创建SAML应用

登录到[阿里云管理控制台](#)，搜索并进入应用程序身份服务（身份即服务，IDaaS）。



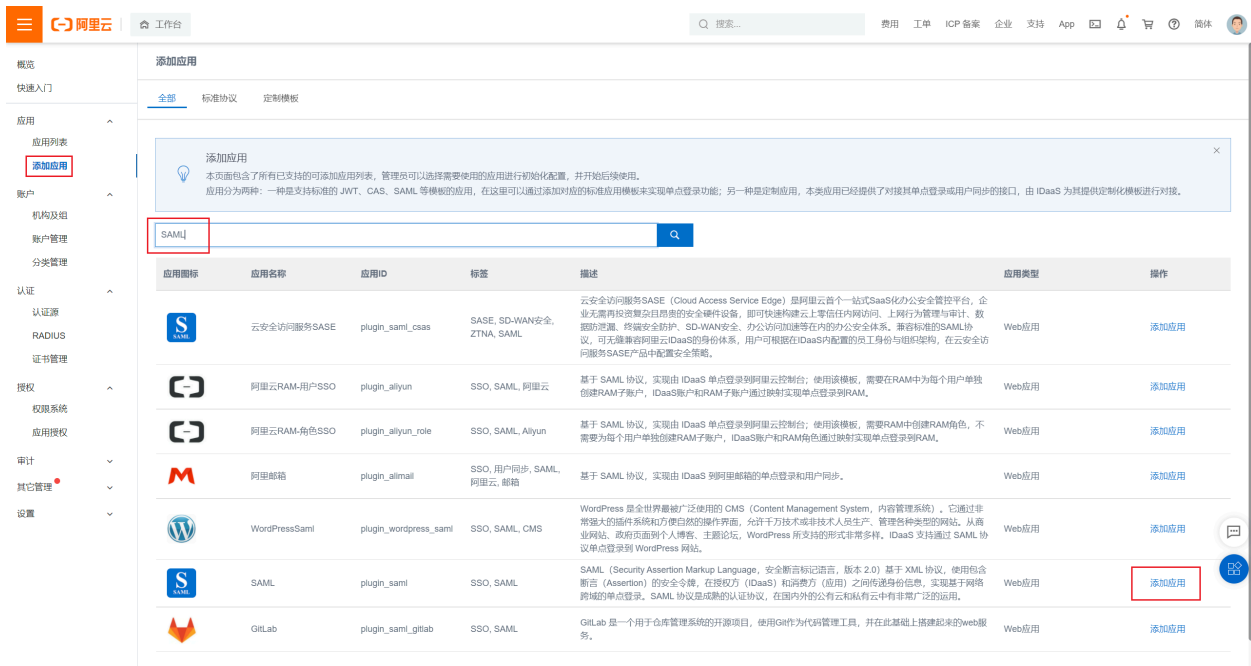
点击 **EIAM 实例列表** 并打开免费版本。



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。



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**



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

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	密钥算法	算法长度	操作
CN=CASDOOR-TEST, ST=Beijing, C=CN	3322747020095790430	1095	RSA	1024	选择 导出

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)



图片大小不超过1MB

应用ID	idaas-cn-shanghai-pv10hq0ojppplugin_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		
* 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	<input checked="" type="checkbox"/>		
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	<input checked="" type="radio"/> 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) <input type="radio"/> 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**。

阿里云 工作台

搜索...

费用 工单 ICP备案 企业 支持 App 购物车 简体

机构及组 数据字典

机构及组

管理员在当前页面可对组织架构、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

在这里对组织架构进行管理。左键可选择组织架构，右键可对组织架构进行操作。

阿里云IDAAS

查看详情 岗位变动 导入 导出 配置LDAP 配置钉钉同步

账户 组 组织机构

新建账户 账户名称 请输入账户名称进行搜索 搜索

当前账户数 1 / 已购套餐规格为 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



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

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

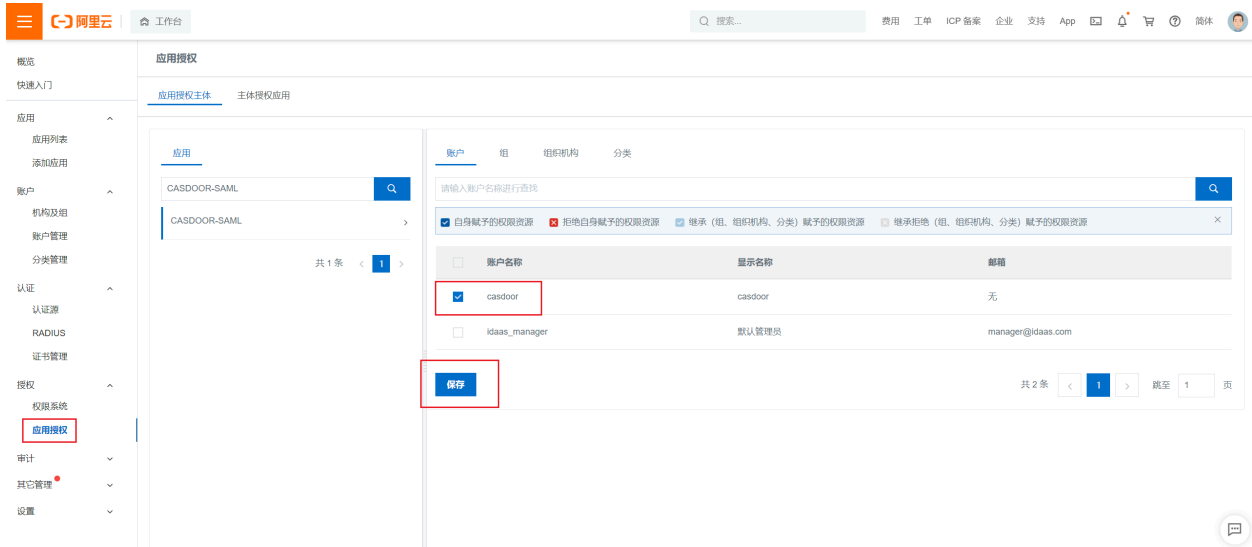
备注

用户备注信息

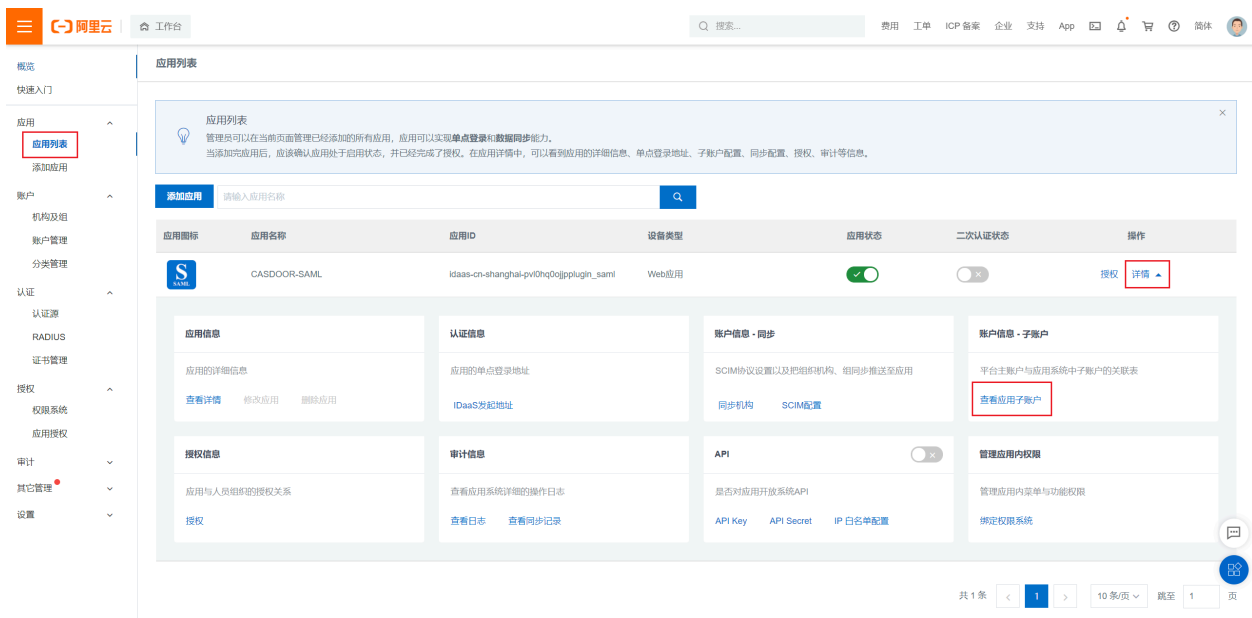
提交

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**。



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。





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。

添加账户关联 ×

* 主账户

* 子账户

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 Metadadta](#)

应用列表

应用详情 (CASDOOR-SAML)

应用ID: idaaS-cn-shanghai-...jin_saml

应用名称: CASDOOR-SAML

应用UId: d9be5906f3c5c031b7abb0efa845f7bRxS506SQ3y7

SigningKey: 3322747020095780430(CN=CASDOOR-TEST)

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

SP ACS URL: http://localhost

IDP IdentityId: CASDOOR [导出 IDaaS SAML 元配置文件](#) [立即轮转密钥 | 导入](#)

SP Entity ID: http://localhost

Binding: POST

Sign Assertion: 是

Assertion Attribute: username=APPLICATIONUSERNAME

IDaaS发起登录地址

SP发起地址: https://dymoykbbkx.login.aliyunidaas.com/enduser/api/application/plugin_saml/idaas-cn-shanghai-.../login_saml/sp_sso?SAMLRequest=xxx&RelayState=yyy

在 Casdoor 配置

在 Casdoor 中创建一个新的提供商。

选择类别为SAML，类型为阿里云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:

```
<?xml-stylesheet type="text/css" href="https://www.w3.org/TR/2001/05/xml.xsl" version="1.0" encoding="UTF-8" standalone="no" ?><md:EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata" name="casdoor-idaas" id="casdoor-idaas" xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" /></md:EntityDescriptor>
```

Parse

Endpoint: https://dvmoykbbkx.login.aliyundaas.com/enduser/api/application/plugin_saml/idaas-cn-shanghai-.../plugin_saml/sp_sso

IdP: MIB5zCCAVCGAwIBAgILHzE2NMHV4wDQYIKoZIhvcNAQEFBQAwNjELMAkGA1UEBhMCQ04xEDAOBgNVBAgTB0JlaWppbmcxFTATBgNVBAMTDENBU0RPTT1HVEVTVDAeFw0yMTYyMDkwNzEyMTFaFw0yNDEyMDgwNzEyMTFaMDYxCzAJE

Issuer URL: 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

Save

复制 SP ACS URL 和 SP 实体 ID 并点击 保存 按钮

编辑您想要在 Casdoor 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: Add


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

Preview: Test signup page... Test signin page...

在阿里云IDaaS中修改SAML应用

禁用应用程序，然后点击 修改应用程序。

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。

图标 

图片大小不超过1MB

应用ID

* 应用名称

* IDP IdentityId
IDP IdentityId is required

* SP Entity ID
SP Entity ID is required

* SP ACS URL(SSO Location)

* NameIdFormat

* Binding

SP 登出地址

Assertion Attribute
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
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，跳转到阿里云IDaaS登录页面，然后在认证后成功登录到Casdoor。



Auto sign in [Forgot password?](#)

[Sign In](#)

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



支付

概述

将支付提供商添加到您的应用程序

PayPal

将PayPal添加为应用的支付提供商

Stripe

将Stripe支付提供商添加到您的应用程序

支付宝

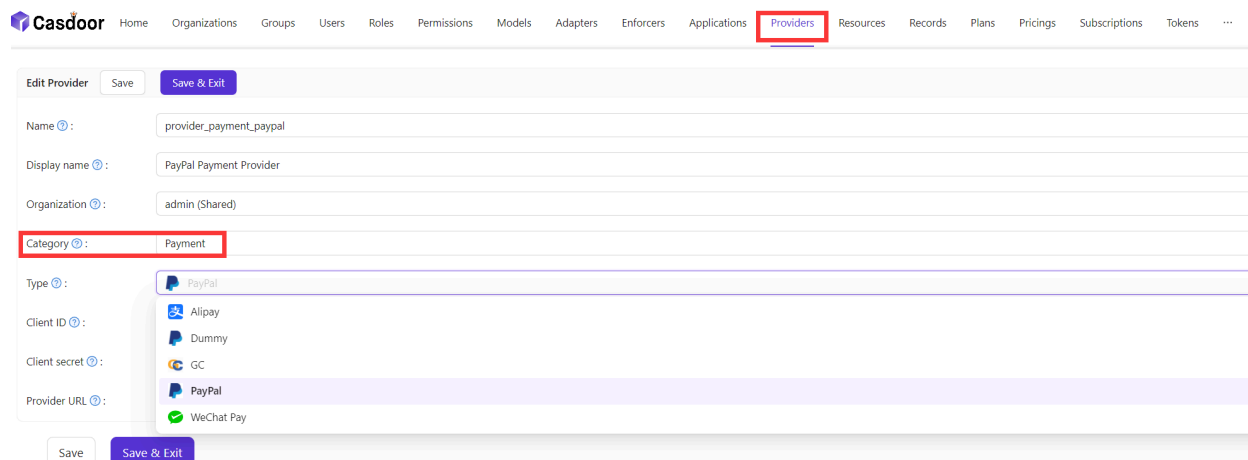
将支付宝支付提供商添加到您的应用程序

 **微信支付**

将微信支付提供商添加到您的应用程序

概述

如果您想在Casdoor中使用支付服务，您需要创建一个支付提供商并将其添加到您的产品中。



要了解如何配置产品，请参考[产品](#)。配置产品后，您可以为产品添加支付提供商，以便用户可以通过支付提供商购买产品。

PayPal

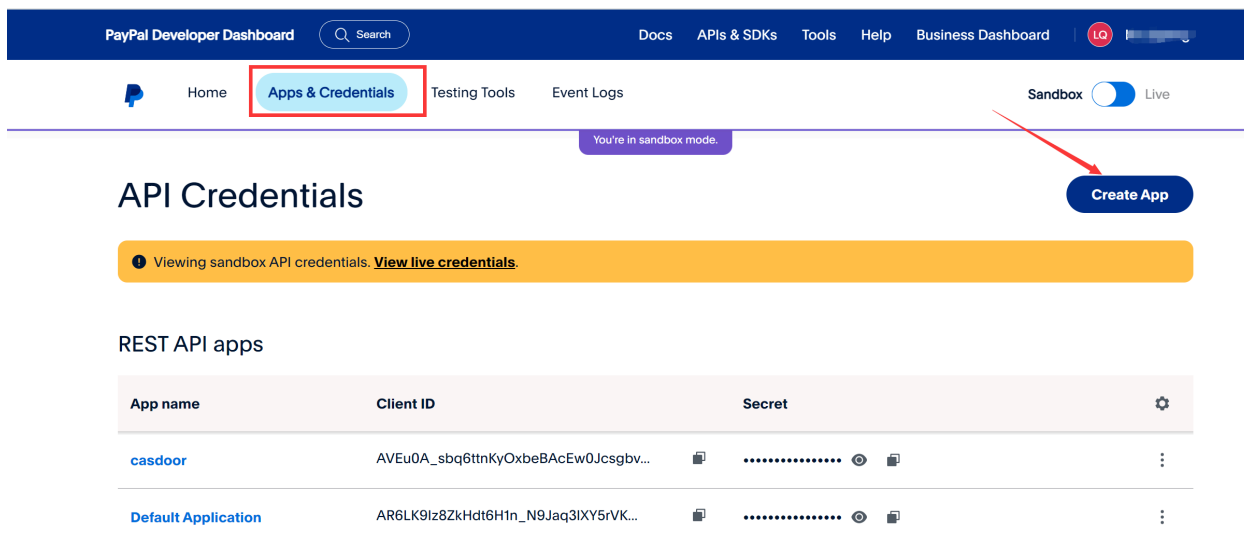
📘 备注

这是一个如何配置PayPal支付提供商的示例。

步骤1: 创建一个PayPal应用

首先, 你需要在PayPal中创建一个应用。要访问PayPal开发者网站, 你应该有一个PayPal商业账户。如果你还没有账户, [先创建一个](#)。

创建PayPal商业账户后, 使用你的账户登录到[开发者控制台](#), 然后点击 **Apps & Credentials** 下的 **Create App**。



The screenshot shows the PayPal Developer Dashboard interface. At the top, there is a navigation bar with 'PayPal Developer Dashboard', a search bar, and links for 'Docs', 'APIs & SDKs', 'Tools', 'Help', and 'Business Dashboard'. Below this, there are tabs for 'Home', 'Apps & Credentials' (highlighted with a red box), 'Testing Tools', and 'Event Logs'. On the right, there is a 'Sandbox' toggle switch set to 'Live' and a 'Create App' button (highlighted with a red arrow). Below the navigation, there is a section titled 'API Credentials' with a yellow banner indicating 'Viewing sandbox API credentials. View live credentials.' Below this, there is a table of REST API apps.

App name	Client ID	Secret	
casdoor	AVEu0A_sbq6ttnKyOxbeBAcEw0Jcsgbv...	⋮
Default Application	AR6LK9Iz8ZkHdt6H1n_N9Jaq3lXY5rVK...	⋮

您可以在应用的基本信息中找到 **Client ID** 和 **Secret key**。

← [Back](#)

casdoor

🔔 Viewing sandbox API credentials. [View live credentials.](#)

API credentials

App name	casdoor
Client ID	AVEu0A_sbq6ttNkyOxbeBAcEw0Jcsgbv2JZvQAAtK JFnaULI-EK-U2XIXcEpEouO9oIknbU7c3m_lfRT5
Secret key 1

[+ Add Second Key](#)

Sandbox account info

[View details](#)

Sandbox URL	https://sandbox.paypal.com
Sandbox Region	C2
Email	sb-qqaiV26894991@business.example.com
Password

Features

步骤2：创建一个PayPal支付提供商

接下来，在Casdoor中创建一个PayPal支付提供商。填写必要的信息：

名称	PayPal中的名称
Category	选择 <input type="text" value="Payment"/>
Type	选择 <input type="text" value="PayPal"/>

名称	PayPal中的名称
Client ID	使用从步骤1获得的 Client ID
Client secret	使用从步骤1获得的 Secret key

步骤3：为你的产品添加PayPal支付提供商

最后，为你的产品添加PayPal支付提供商，这样用户就可以使用PayPal购买产品。

Price: 10.03

Quantity: 99

Sold: 10



Payment providers: provider_payment_paypal x

Return URL:

State: Published

Preview: Test buy page.

Buy Product

Name	Test Product				
Detail	This is the detail of test product	Tag	Casdoor Summit 2022	SKU	test_product
Image					
Price	\$10.03 (USD)	Quantity	99	Sold	10
Pay					

i 备注

以上操作都是在PayPal的 Sandbox 模式下进行的。如果你想在实际生产环境中使用它，你需要在PayPal的 Live 模式下创建一个应用，并在Casdoor的配置文件 `conf/app.conf` 中设置 `runmode=prod`。

Stripe

📘 备注

这是一个如何配置Stripe支付提供商的示例。

步骤1。 获取可发布的密钥和秘密密钥

首先，你需要在Stripe有一个账户。创建Stripe账户后，使用您的账户凭据登录到[开发者仪表盘](#)。您可以在 [API密钥](#) 选项卡下找到 [可发布的密钥](#) 和 [秘密密钥](#)。

API keys [Learn more about API authentication →](#)

🔑 Viewing test API keys. Toggle to view live keys. 🔴 Viewing test data

Standard keys
These keys will allow you to authenticate API requests. [Learn more](#)

NAME	TOKEN	LAST USED	CREATED	
Publishable key	pk_test_5INd4fgHcnRPGqR9hFYxL3BRuQ31828gEgXYbu9N3s5LPvOJg7CJc0rvaFku6o5xawcWTElpVhGGMYoqOy590CHX700eax64aml	Aug 11	📅 Aug 9	...
Secret key	Reveal test key	Aug 11	📅 Aug 9	...

Restricted keys
For greater security, you can create restricted API keys that limit access and permissions for different areas of your account data. [Learn more](#) + Create restricted key

NAME	TOKEN	LAST USED	CREATED
No restricted keys			

步骤2。 创建一个Stripe支付提供商

接下来，在Casdoor中创建一个Stripe支付提供商，填写必要的信息。

名称	Stripe中的名称
Category	选择 支付
Type	选择 Stripe
Client ID	从步骤1获取的 可发布的密钥
Client secret	从步骤1获取的 秘密密钥

Edit Provider Save Save & Exit

Name ⓘ :

Display name ⓘ :

Organization ⓘ :

Category ⓘ :

Type ⓘ : S Stripe

Client ID ⓘ :

Client secret ⓘ :

Provider URL ⓘ :

Save Save & Exit

步骤3。为您的产品添加Stripe支付提供商

最后，为您的产品添加Stripe支付提供商，以便用户可以使用Stripe购买产品。

Currency: USD

Price: 10.04

Quantity: 99

Sold: 10




Payment providers: provider_payment_paypal x provider_payment_stripe x

Return URL: @

State: Published

Preview: test buy page

Buy Product

Name	Test Product				
Detail	This is the detail of test product	Tag	Casdoor Summit 2022	SKU	test_product
Image					
Price	\$10.04 (USD)	Quantity	99	Sold	10
Pay	 PayPal	 Stripe			

支付宝

步骤1。准备

首先，您需要在支付宝开放平台拥有一个商家账户。

在访问支付宝之前，需要做一些准备工作。

您可以参考文档 [接入前的准备](#) 以获取更多信息。

1.1 获取APPID

登录支付宝开放平台控制台并[创建一个应用](#)。

如何获取APPID：[支付宝APPID查询指南](#)

1.2 配置证书

根据[文档](#)生成一个RSA2证书，然后您可以获取 `appPrivateKey.txt` 和 `appPublicKey.txt`。

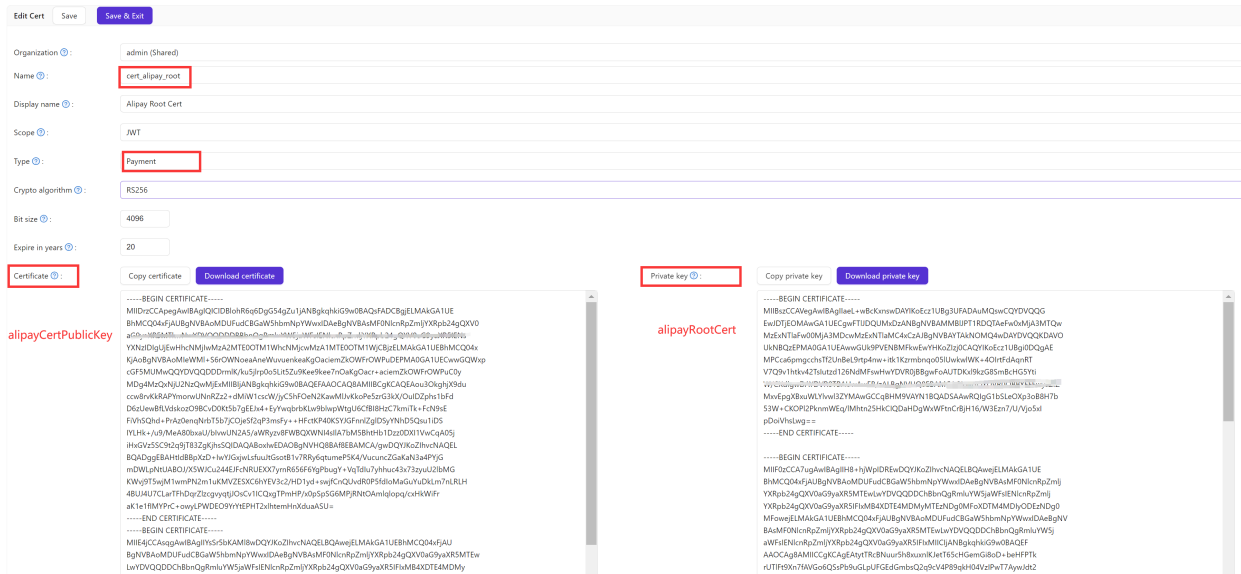
将证书上传到应用程序，然后您可以下载三个文件：`alipayRootCert.crt`，`appCertPublicKey.crt`，`alipayCertPublicKey.crt`。

在Casoor创建一个名为 `App Cert` 的证书：

名称	支付宝中的姓名
Type	选择 Payment
Certificate	appCertPublicKey.crt 的内容
Private key	appPrivateKey.txt 的内容

在Casoor创建一个名为 **Root Cert** 的证书:

名称	支付宝中的名称
Type	选择 Payment
Certificate	alipayCertPublicKey.crt 的内容
Private key	alipayRootCert.crt 的内容



步骤2。创建一个支付宝支付提供商

接下来，在Casdoor中创建一个支付宝支付提供商，填写必要的信息。

名称	支付宝中的姓名
Category	选择 Payment
Type	选择 Alipay
Client ID	从步骤1.1获取的 APPID
Cert	App Cert 在步骤1.2中配置
Root Cert	Root Cert 在步骤1.2中配置

Edit Provider Save Save & Exit

Name: provider_payment_alipay

Display name: Alipay Payment Provider

Organization: admin (Shared)

Category: Payment

Type: Alipay

Client ID: 2021003117621368

Client secret:

Cert: cert_alipay_app

Root Cert: cert_alipay_root

Provider URL: [🔗](#)

Save Save & Exit

步骤3。为您的产品添加支付宝支付提供商

最后，为您的产品添加支付宝支付提供商，以使用户可以使用支付宝购买产品。

Quantity: 99

Sold: 10

Payment providers: provider_payment_paypal x provider_payment_stripe x provider_payment_wechat x provider_payment_alipay x

Return URL: [🔗](#)

State: Published

Preview: [Test buy page](#)

Buy Product

Name	Test Product		
Detail	This is the detail of test product	Tag	Casdoor Summit 2022
	SKU	test_product	
Image			
Price	¥0.01 (CNY)	Quantity	99
		Sold	10
Pay			

Save Save & Exit

微信支付

步骤1. 准备工作

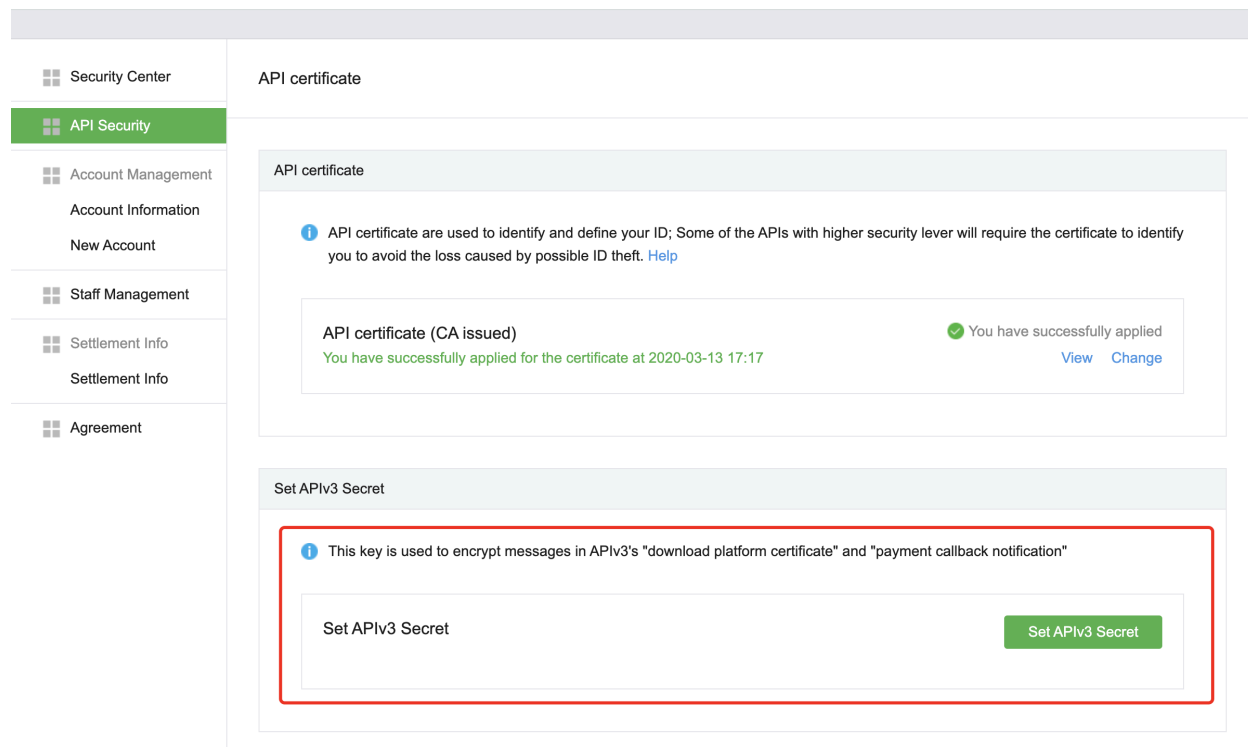
首先，你需要在[微信商户平台](#)拥有一个商户账户。

在接入微信支付之前，需要做一些准备工作。

你可以参考文档[接入前的准备](#)以获取更多信息。

1.1 获取API密钥v3

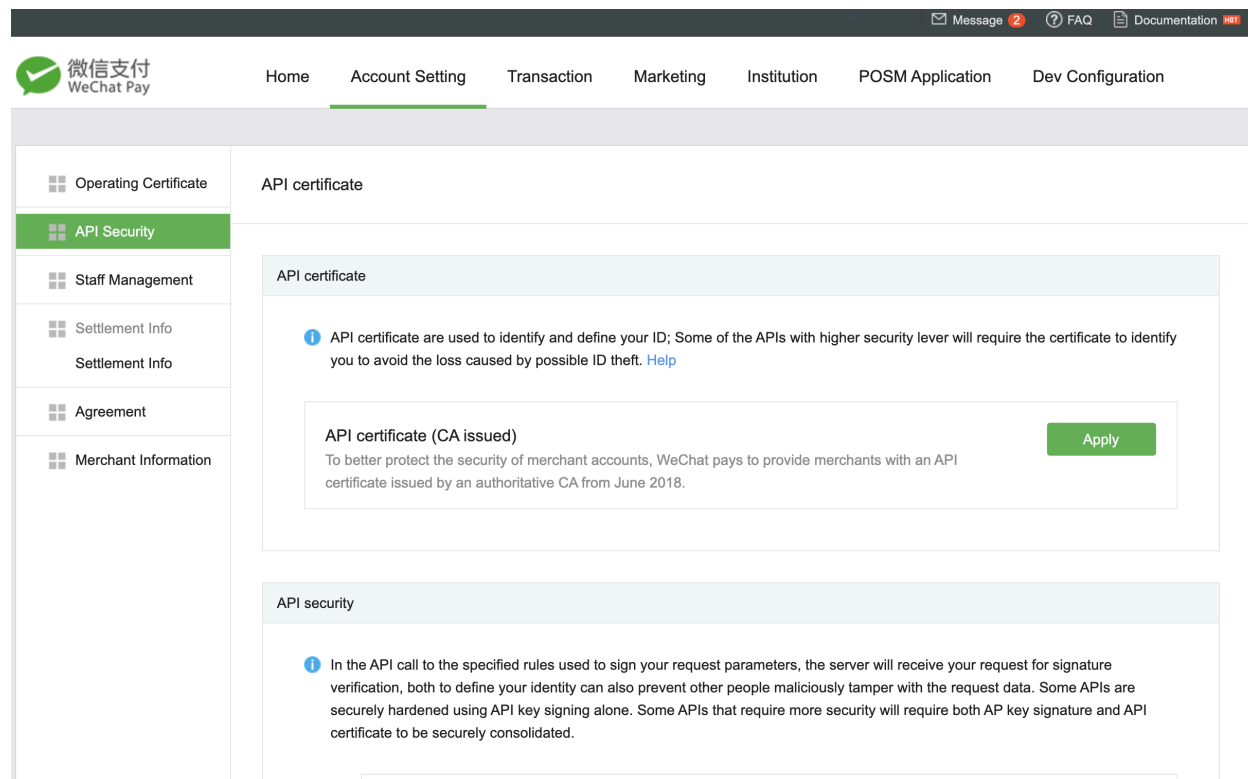
登录微信商户平台，选择 [账户设置](#) > [API安全](#) > [设置APIv3密钥](#)，然后点击 [设置APIv3密钥](#) 获取 [API密钥v3](#)。



如何获取 API 密钥 v3: [API v3 密钥设置](#)

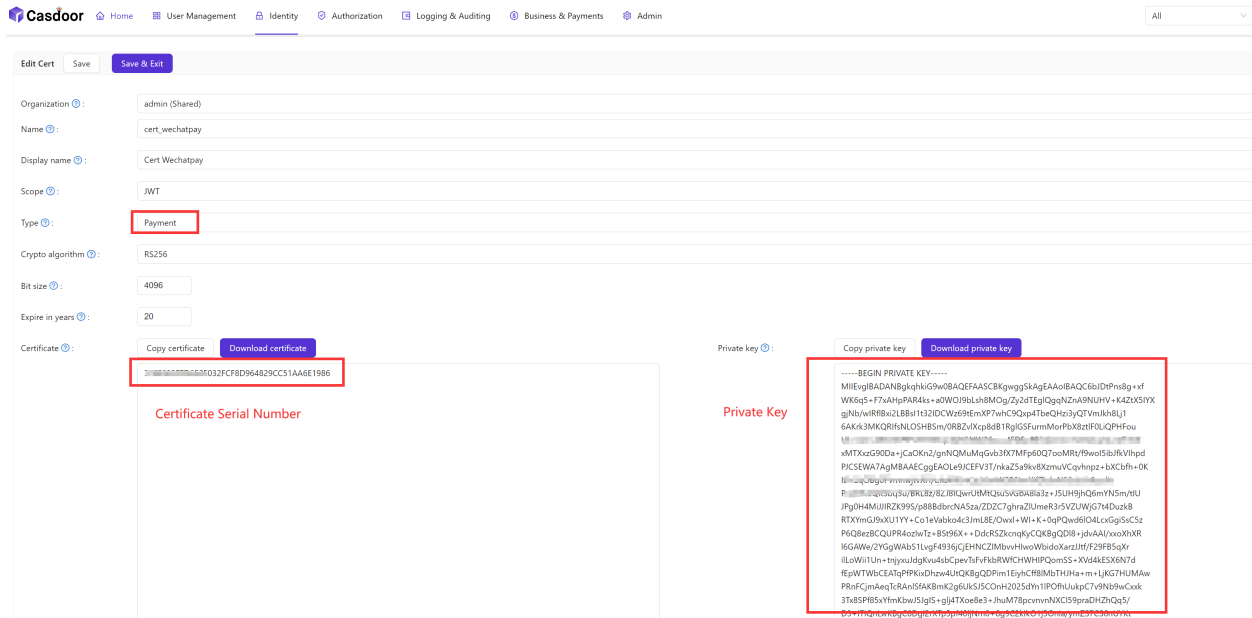
1.2 获取商户证书

你可以登录微信商户平台, 并选择 [账户设置](#) > [API 安全](#) > [API 证书](#) 下载证书。



下载证书后, 根据[如何查看证书序列号](#)获取 [证书序列号](#), 并根据[如何获取证书的私钥](#)获取 [私钥](#)。

然后, 在Casdoor创建一个 [证书](#) 并填写必要的信息。



1.3 获取商户ID和应用ID

如何获取 **商户ID** : [微信支付商户ID查询指南](#)

如何获取 **应用ID** : [微信支付APPID查询指南](#)

步骤2. 创建一个微信支付提供商

接下来, 在Casdoor中填写必要的信息创建一个微信支付提供商。

名称	微信支付中的名称
Category	选择 支付
Type	选择 微信支付
Client ID	从步骤1.3获取的 商户ID

名称	微信支付中的名称
Client secret	从步骤1.1获取的 API密钥v3
App ID	从步骤1.3获取的 应用ID
Cert	在步骤1.2配置的 证书


Edit Provider Save Save & Exit

Name ⓘ : provider_payment_wechat

Display name ⓘ : Wechat Payment Provider

Organization ⓘ : admin (Shared)

Category ⓘ : Payment


Type ⓘ :  WeChat Pay

Client ID ⓘ : 1619999244

Client secret ⓘ : ***

App ID ⓘ : wx933a9cd81c396d1

Cert ⓘ : cert_wechatpay

Provider URL ⓘ : 

Save Save & Exit

步骤3. 为您的产品添加微信支付提供商

最后，为您的产品添加微信支付提供商，以便用户可以使用微信支付购买产品。

Currency: CNY

Price: 0.01

Quantity: 99

Sold: 10

Payment providers: provider_payment_paypal x provider_payment_stripe x provider_payment_wechat x

Return URL: eP

State: Published

Preview: Test buy page.

Buy Product

Name	Test Product		
Detail	This is the detail of test product	Tag	Casdoor Summit 2022
		SKU	test_product
Image			
Price	¥ 0.01 (CNY)	Quantity	99
		Sold	10
Pay			

支持JSAPI支付

目前，Casdoor支持微信支付中的JSAPI支付和原生支付。

要支持JSAPI支付，你应该配置一个支持微信媒体平台的微信OAuth提供商。微信OAuth提供商的客户端ID 2和微信支付提供商的应用ID需要相同。

The image displays two side-by-side screenshots of the 'Edit Provider' configuration interface. The left screenshot shows the configuration for 'provider_casdoor_wechat'. The right screenshot shows the configuration for 'provider_payment_wechatpay'. A red arrow points from the 'Client ID 2' field in the left form to the 'App ID' field in the right form, indicating that the same ID is used for both configurations.

Field	provider_casdoor_wechat	provider_payment_wechatpay
Name	provider_casdoor_wechat	provider_payment_wechatpay
Display name	Casdoor WeChat	Payment - WeChatPay
Organization	admin (Shared)	admin (Shared)
Category	OAuth	Payment
Type	WeChat	WeChat Pay
Client ID	wx049c70e6c2027b0b	1619999244
Client secret	***	***
Client ID 2	wx933a9cd81c396d1	wx933a9cd81c396d1
Client secret 2	***	***
Enable QR code	Off	Off
Provider URL	https://open.weixin.qq.com/	https://pay.weixin.qq.com/index.php/core/cert/api_cert#/

用户通过微信登录后（在移动场景中，例如微信移动应用内的微信内置浏览器），可以基于JSAPI支付使用微信支付购买产品。

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的默认验证码

Cloudflare Turnstile

将Cloudflare Turnstile添加到您的应用程序

reCAPTCHA

添加reCAPTCHA 到您的应用程序

hCaptcha

将 hCaptcha 添加到您的应用程序

阿里云 Captcha

将阿里云 Captcha 添加到您的应用程序













Geetest

将Geetest验证码添加到您的应用程序

概述

Casdoor可以配置为支持不同的验证码，以验证操作是否由人类执行。通过添加验证码提供商并在应用程序中应用它，当用户登录、注册或忘记密码并需要发送代码时，将出现一个验证码检查对话框，以验证操作是否由人类执行。

目前，Casdoor支持多个验证码提供商。以下是Casdoor支持的提供商：

默认	Cloudflare 旋 转门	reCAPTCHA	hCaptcha	阿里云验 证码	Geetest
					
					

我们将向您展示如何应用验证码并将其添加到Casdoor。

添加验证码提供商

1. 导航至您的Casdoor首页。
2. 点击顶部栏中的 **Providers**。
3. 点击 **Add**，然后你会在顶部列表中看到一个新的提供商。
4. 点击新的提供商以修改它。
5. 在 **Category** 中选择 **Captcha**。
6. 在 **Type** 中选择你需要的验证码提供商。

7. 填写最重要的信息。不同的验证码提供商可能需要填写不同的信息。

在应用程序中应用

1. 点击顶部栏中的 `Application`，然后选择一个应用进行编辑。
2. 点击提供商添加按钮，并选择你刚刚添加的提供商。
3. 完成！

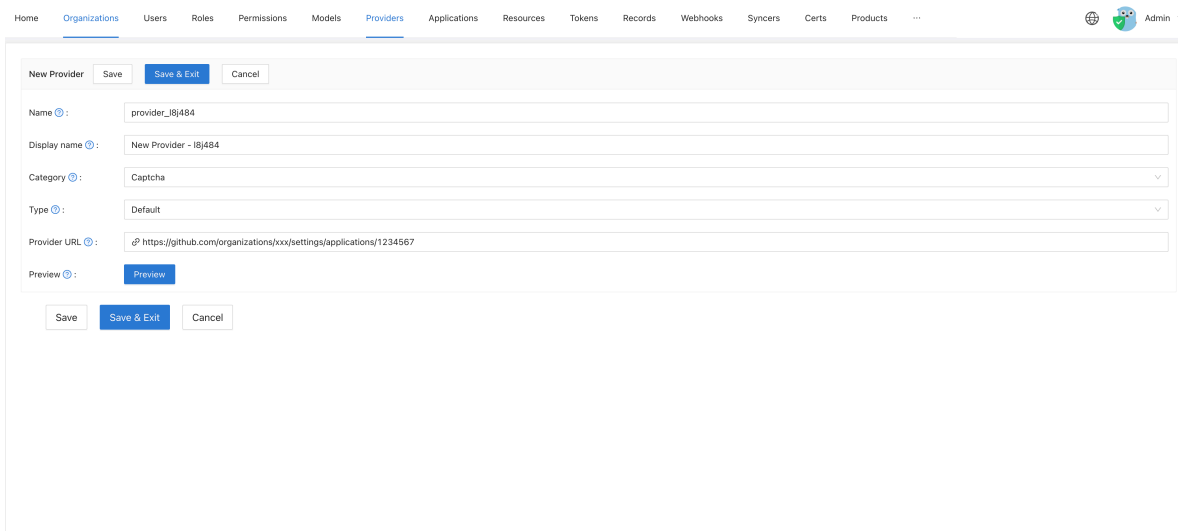
默认

默认的验证码实现会生成并验证一个图像。在默认的验证码图片中，使用了一串长度为5的0-9数字序列。

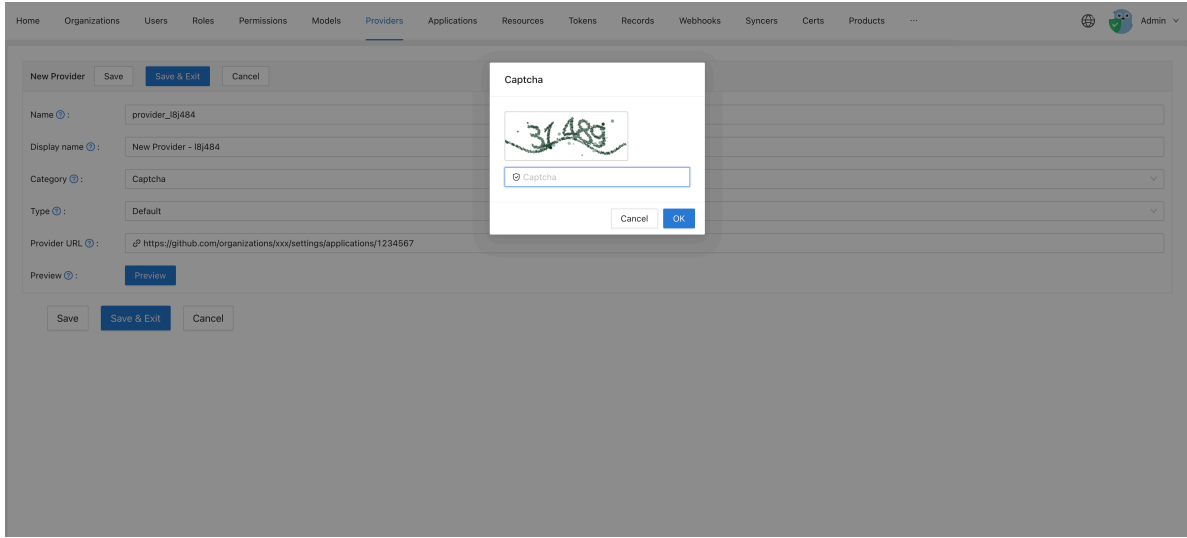
在Casdoor中进行配置

要在Casdoor中配置默认的验证码，请按照以下步骤操作：

1. 在 Casdoor 中创建一个新的提供商。
2. 选择类别为**Captcha**，类型为**Default**。



3. 点击**预览**按钮以预览此验证码的样式。



在您的应用程序中应用

要在您的应用程序中应用默认的验证码，请执行以下操作：

1. 编辑您想要在 Casdoor 中配置的应用程序。
2. 选择你刚刚添加的提供商。有三种类型的规则可供选择：
 - **Always**：登录时始终需要进行人机验证。
 - **None**：永不需要人机验证。如果账户在15分钟内尝试使用错误的密码登录5次，该账户将被封锁。该阻塞将在15分钟后解除。
 - **Dynamic**：在5次登录尝试失败后，将需要进行人机验证，但账户不会被封锁。

Providers ⓘ

Name	Category	Type	canSignUp	canSignIn	canUnlink	prompted	Rule	Action
provider_4olfdm	Captcha						Always	⬆️ ⬇️ 🗑️
provider_casdoor_github	OAuth		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		⬆️ ⬇️ 🗑️
provider_casdoor_google	OAuth		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		⬆️ ⬇️ 🗑️

我们还提供了一个演示视频来展示规则的不同，希望对您有所帮助。

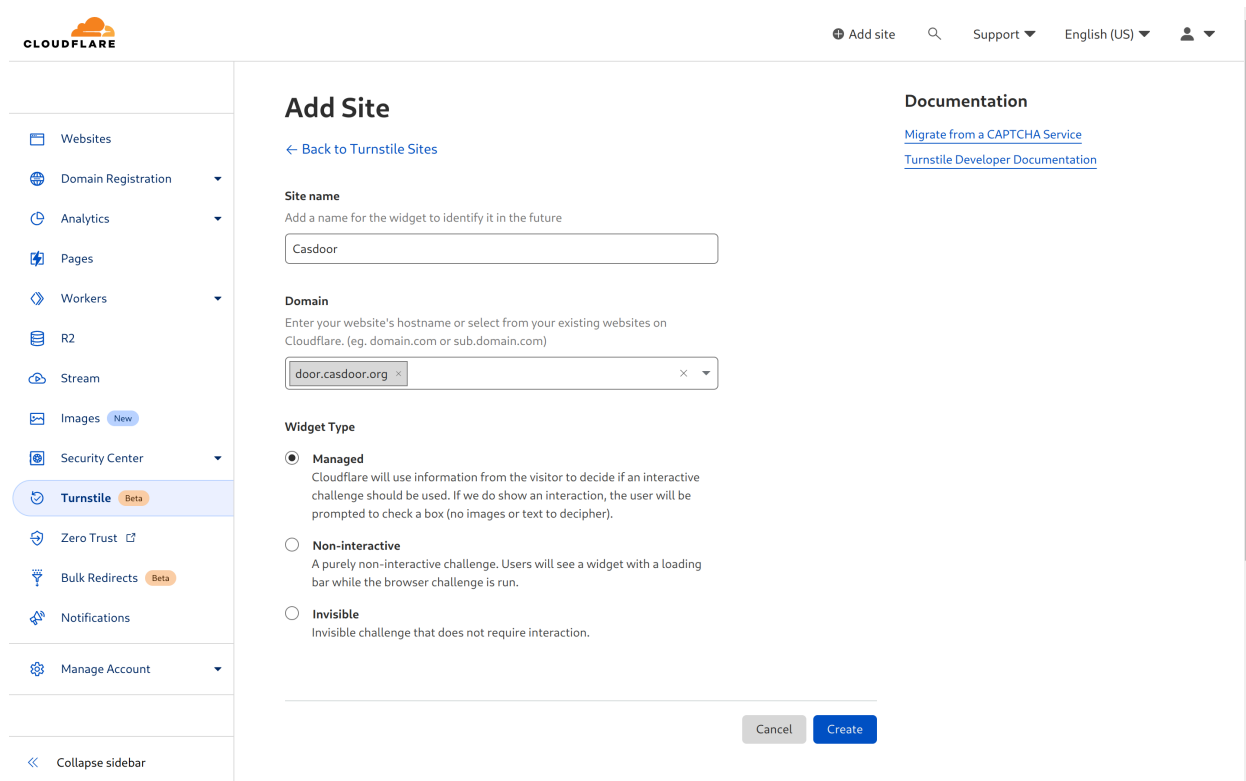
Cloudflare Turnstile

Cloudflare Turnstile是Cloudflare提供的一个CAPTCHA服务，它是对CAPTCHA的一种用户友好、保护隐私的替代方案。您可以在[Turnstile文档](#)中找到更多详细信息。

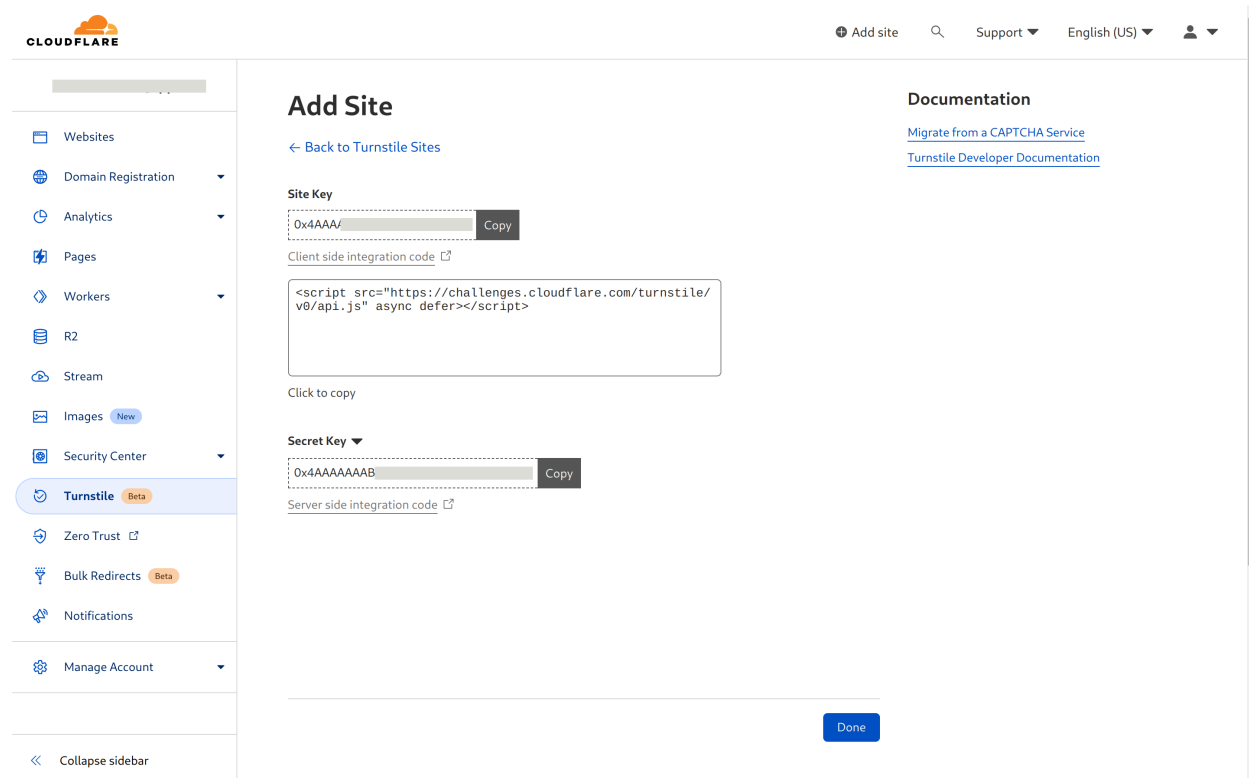
创建一个API密钥对

要开始使用Cloudflare Turnstile，您需要[创建一个Cloudflare账户](#)，导航到导航栏上的Turnstile标签，并获取站点密钥和秘密密钥。

首先，为小部件添加一个名称，以便将来识别它，并输入您的网站的主机名。然后选择小部件类型。建议选择Managed。最后，点击创建。



然后，您将能够获得一个站点密钥和一个秘密密钥。



在Casdoor中配置

在Casdoor中创建一个新的提供者。

将类别选择为**Captcha**，类型选择为**Cloudflare Turnstile**。填写您在上一步中获得的站点密钥和秘密密钥。

Edit Provider

Name:

Display name:

Organization:

Category:

Type:

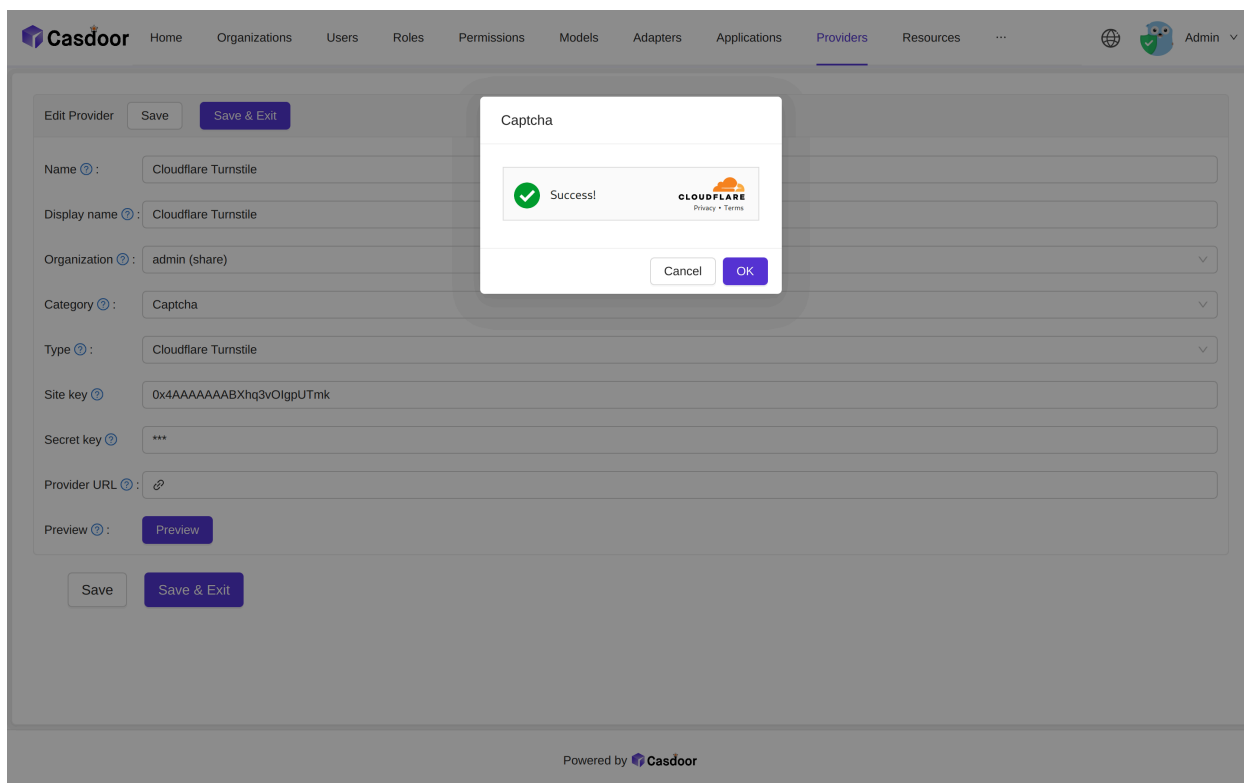
Site key:

Secret key:

Provider URL:

Preview:





您可以点击**预览**按钮，查看此CAPTCHA的样式预览。



应用集成

编辑您想在Casdoor中配置的应用程序。选择您刚刚添加的提供者，然后点击**保存按钮**。

Providers ⓘ :

Name	Category	Type	Can signup	Can signin	Can unlink	Prompted	Rule	Action
Cloudflare Turnstile	Captcha						None	  

reCAPTCHA

reCAPTCHA由Google提供，我们使用的是reCAPTCHA v2复选框。您可以在[此链接](#)中找到更多相关详情。

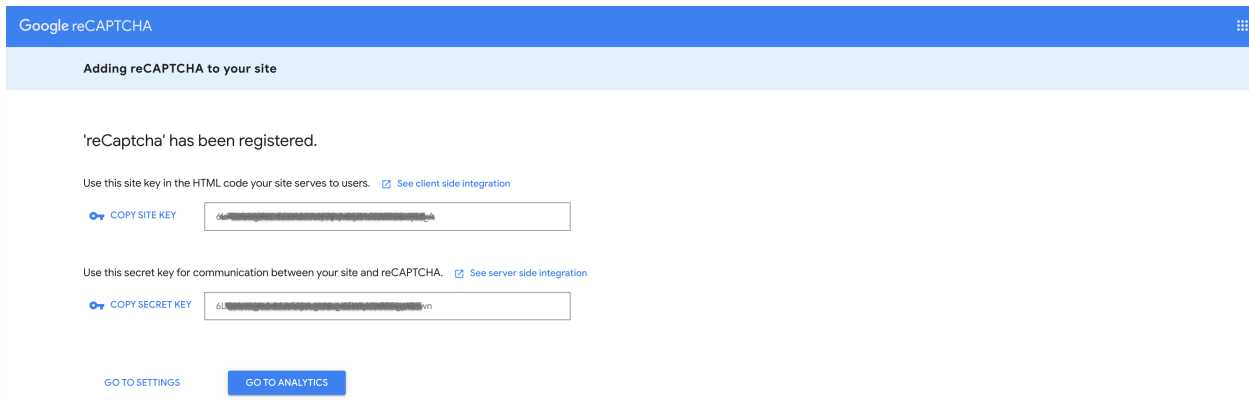
创建一个 API 密钥对

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网站或移动应用上调用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](#)，然后填写授权域名或[包名](#)。在您接受服务条款后，点击[注册](#)以获取新的API密钥对。

The screenshot shows the 'Register a new site' page in the Google reCAPTCHA console. At the top, there's a blue header with 'Google reCAPTCHA' and a hamburger menu icon. Below it is a light blue bar with a back arrow and the text 'Register a new site'. A yellow banner below that says 'Get unlimited assessments using reCAPTCHA Enterprise'. The main form area has a 'Label' field with a help icon and a 'reCaptcha' field with a character count '9 / 50'. Under 'reCAPTCHA type', there are three radio button options: 'reCAPTCHA v3' (Verify requests with a score), 'reCAPTCHA v2' (Verify requests with a challenge), and 'reCAPTCHA v2' with three sub-options: 'I'm not a robot' Checkbox (selected), Invisible reCAPTCHA badge, and reCAPTCHA Android. Below this is a 'Domains' section with a help icon and a list containing '+ casdoor.org'. The 'Owners' section shows 'resultelee@gmail.com (You)' and a '+ Enter email addresses' button. At the bottom, there is a checked checkbox for 'Accept the reCAPTCHA Terms of Service' and a small disclaimer: '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.'

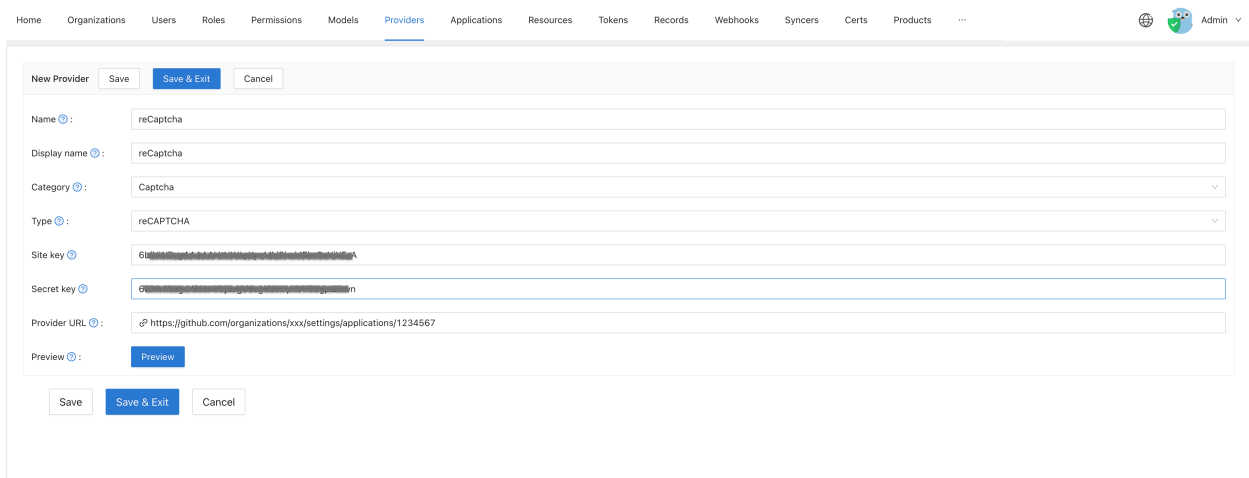
然后，您将收到一个站点密钥和一个秘密密钥。



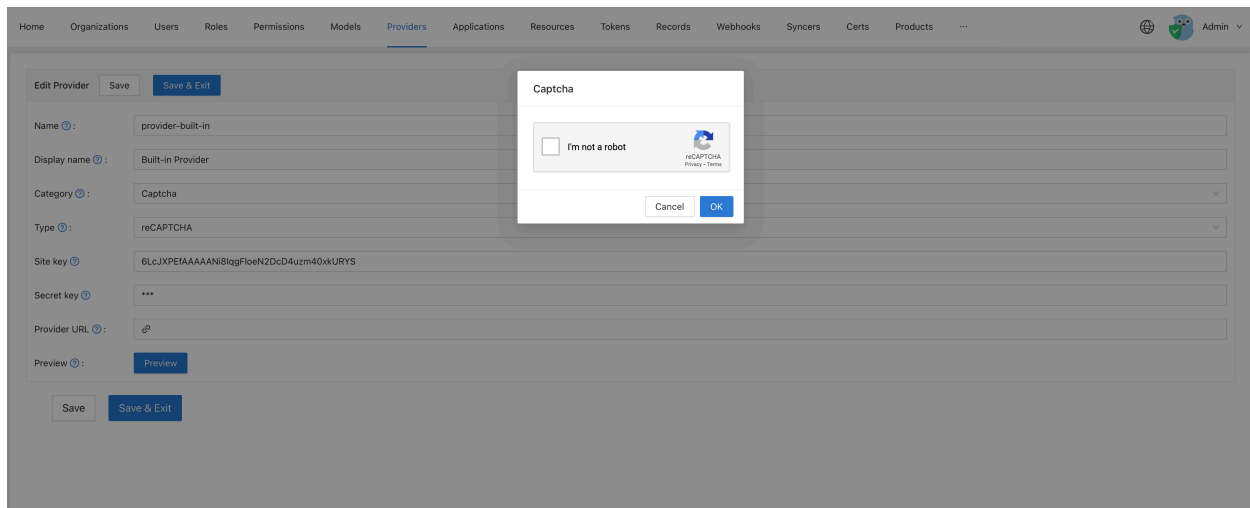
在 Casdoor 配置

在 Casdoor 中创建一个新的提供商。

选择类别为 **Captcha**，类型为 **reCAPTCHA**。您需要提供在上一步中创建的网站密钥和秘密密钥。



您可以点击 **预览** 按钮来查看这个验证码的样式。



在应用程序中应用

编辑您想要在 Cassdoor 中配置的应用程序。选择你刚刚添加的提供商，然后点击**保存按钮**。

Providers ⓘ Add

Name	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。您可以在此处找到有关hCaptcha的更多详细信息[这里](#)。

创建一个 API 密钥对

要开始使用hCaptcha，您需要为您的网站注册一个API密钥对。您可以在您的[个人资料页面](#)上获取您的站点密钥。

一旦您已经注册，您将会收到一个站点密钥和一个秘密密钥。

在 Casdoor 配置

要在Casdoor中配置hCaptcha，请创建一个新的提供商。

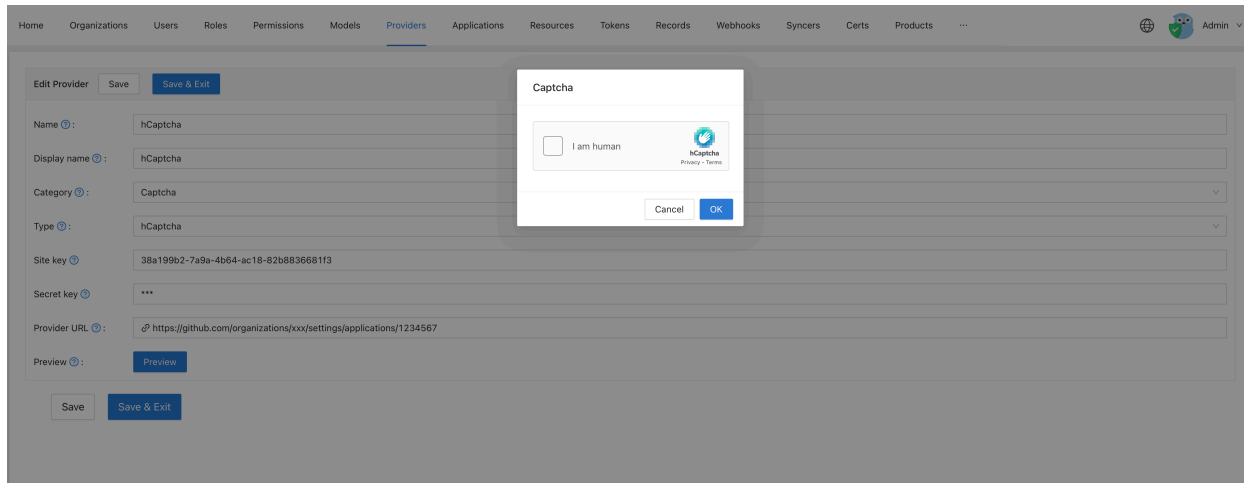
选择类别为**Captcha**，类型为**hCaptcha**。填写在上一步中获得的站点密钥和秘密密钥。

The screenshot shows the 'New Provider' configuration form in Casdoor. The form is titled 'New Provider' and has buttons for 'Save', 'Save & Exit', and 'Cancel'. The fields are as follows:

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

At the bottom of the form, there are buttons for 'Save', 'Save & Exit', and 'Cancel'. The 'Save & Exit' button is highlighted in blue.

您可以点击**预览**按钮，查看验证码样式的效果。



在您的应用程序中应用

转到您想在Casdoor中配置的应用程序。选择你刚刚添加的提供商，然后点击**保存**按钮。



阿里云 Captcha

阿里云 Captcha 是由 阿里云 提供的验证码服务。它提供两种验证方式："滑动验证" 和 "智能验证"。您可以从此 [链接](#) 中查询更多详细信息。

❗ 信息

目前，只支持[阿里云验证码1.0](#)。[阿里云验证码 2.0](#)目前处于公开测试阶段，所以近期没有适应性改变的计划。

在阿里云中添加验证码配置

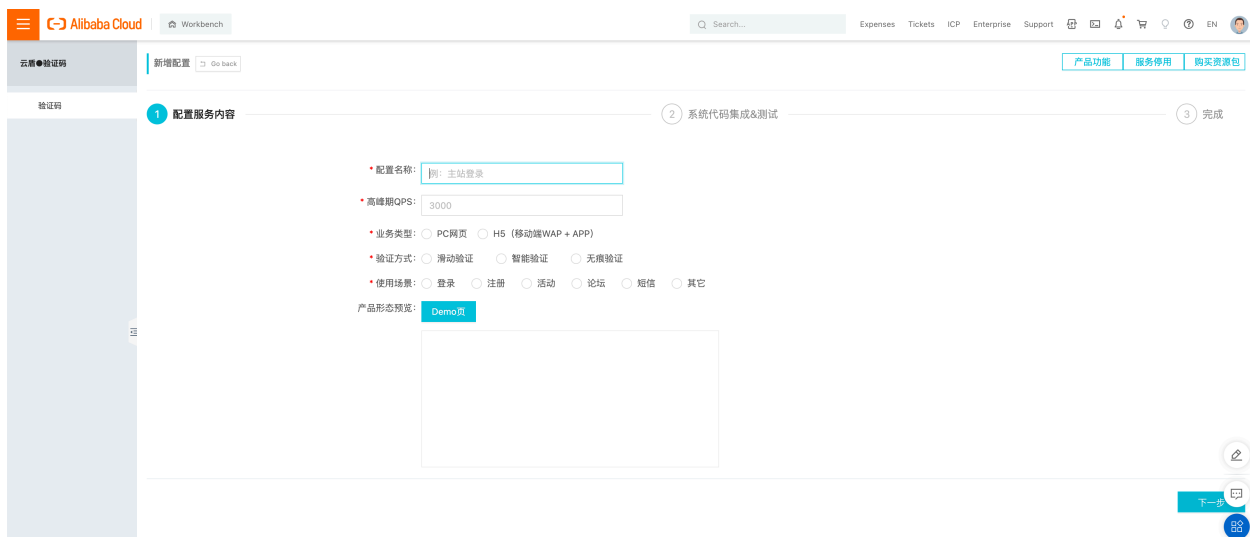
要添加验证码配置，请登录[阿里云管理控制台](#)，搜索并进入验证码服务。然后，点击**确认开启**以启用验证码服务。



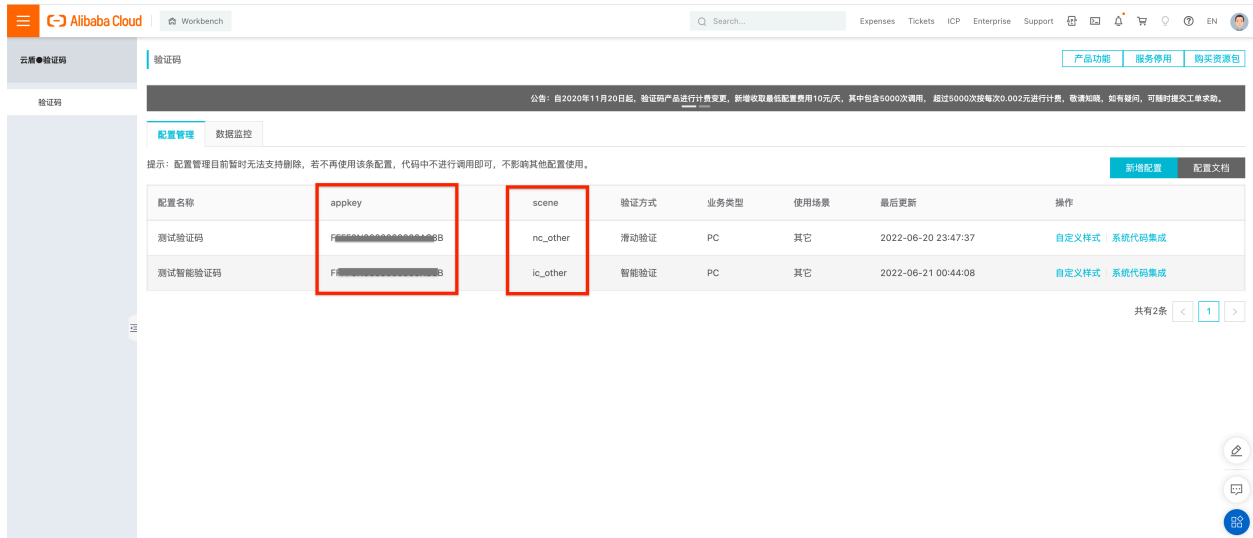
一旦您进入了验证码管理控制台，点击**添加配置**。



填写所有必需的信息并提交表格。



现在，您可以在控制台中查看 `Scene` 和 `App key`。



另外，`Access key` 和 `Secret access key` 可以在您的个人资料中找到。

在 Casdoor 配置

在 Casdoor 中创建一个新的提供商。

选择类别为 `Captcha`，类型为 `hCaptcha`。然后，选择子类型：“滑动验证”或“智能验证”。确保填写在上一步中创建的 `Access key`、`Secret access key`、`Scene` 和 `App key`。

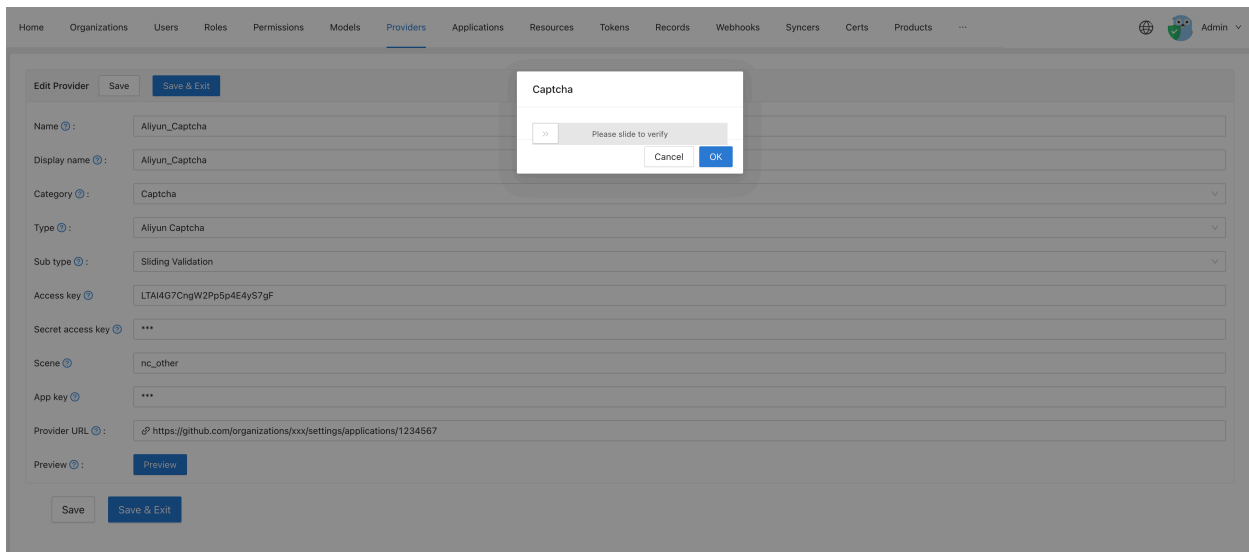
The screenshot shows the 'New Provider' configuration page. The form contains the following fields and values:

- Name: Aliyun_Captcha
- Display name: Aliyun_Captcha
- Category: Captcha
- Type: Aliyun Captcha
- Sub type: Sliding Validation
- Access key: L*****gF
- Secret access key: gP*****N
- Scene: nc_other
- App key: F*****8B
- Provider URL: https://github.com/organizations/xxx/settings/applications/1234567

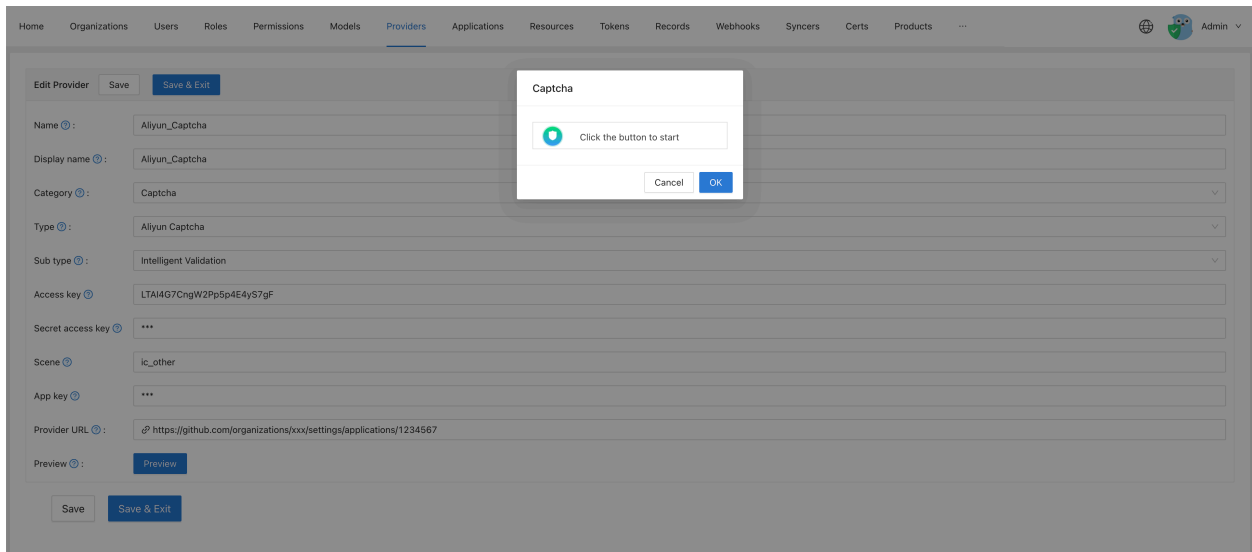
Buttons: Save, Save & Exit, Cancel (top); Save, Save & Exit, Cancel (bottom). A 'Preview' button is located below the Provider URL field.

您可以点击**预览**按钮来查看这个验证码的样式。

以下图片显示了“滑动验证”的预览：



而这张图片展示了“智能验证”的预览：



应用集成

编辑您想要配置Casdoor的应用程序。选择新添加的提供商，然后点击**保存**按钮。

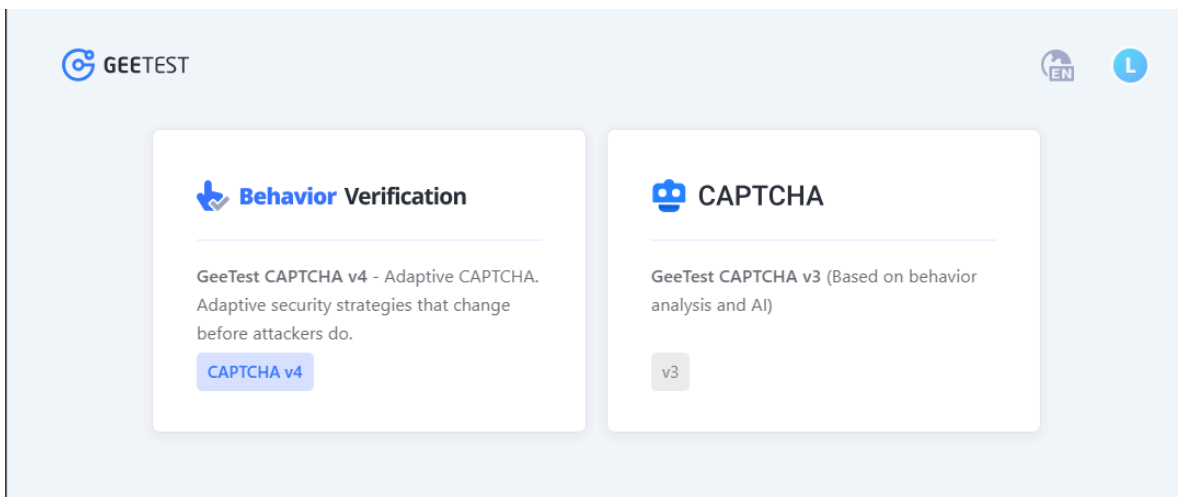


Geetest

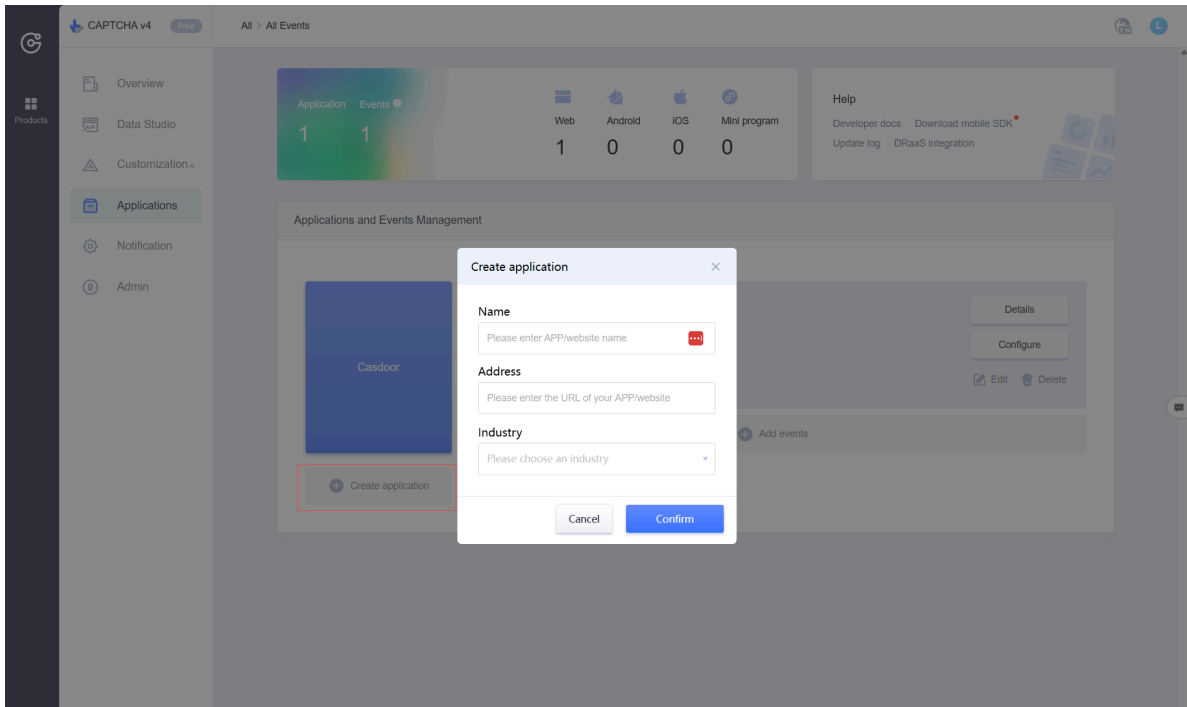
配置Geetest

按照以下步骤配置Geetest并获取公钥和密钥：

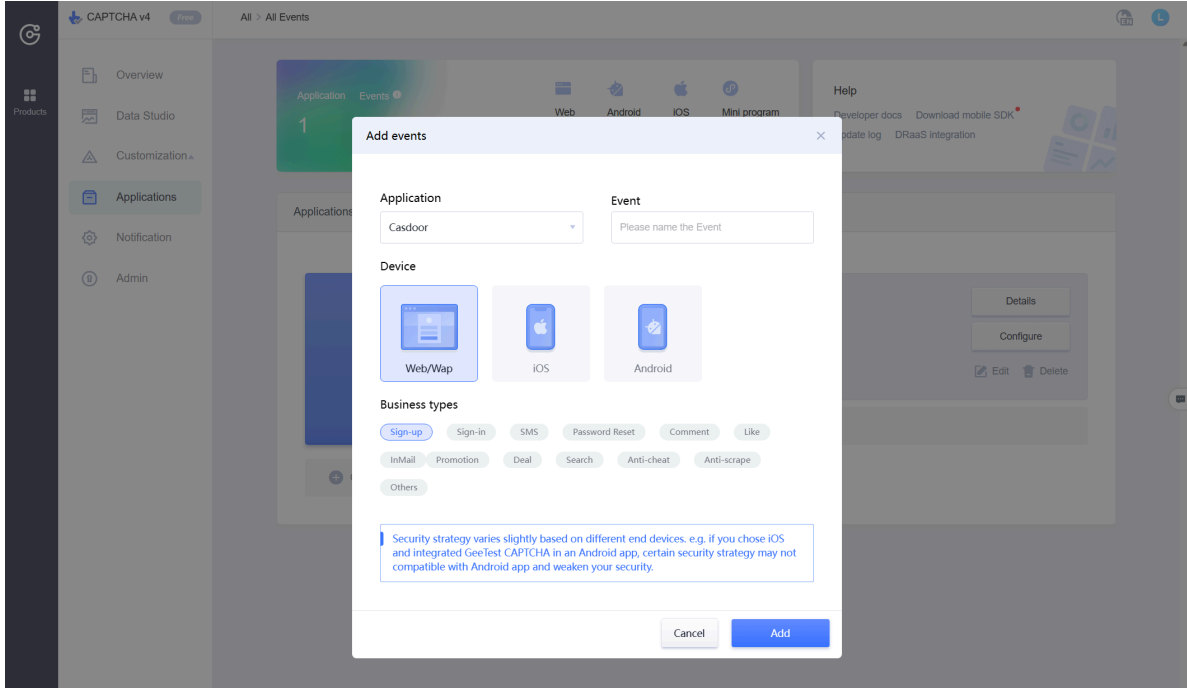
1. 转到Geetest产品页面上的Geetest CAPTCHA V4部分。



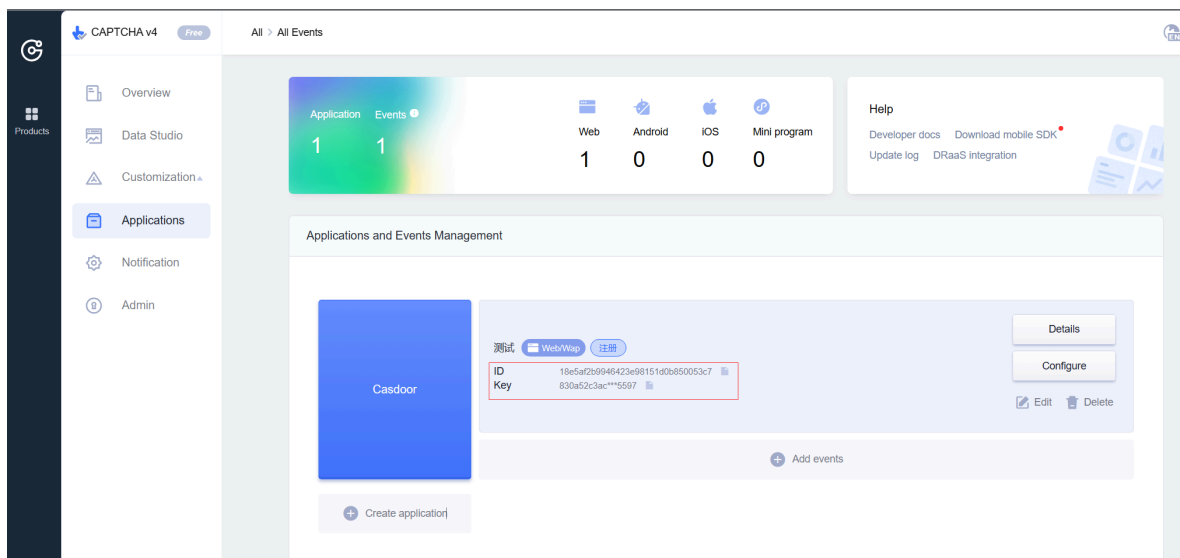
2. 通过输入您的应用程序的名称和地址来创建一个应用程序。



3. 添加事件并为设备选择"web"。



4. 检索ID和Key。



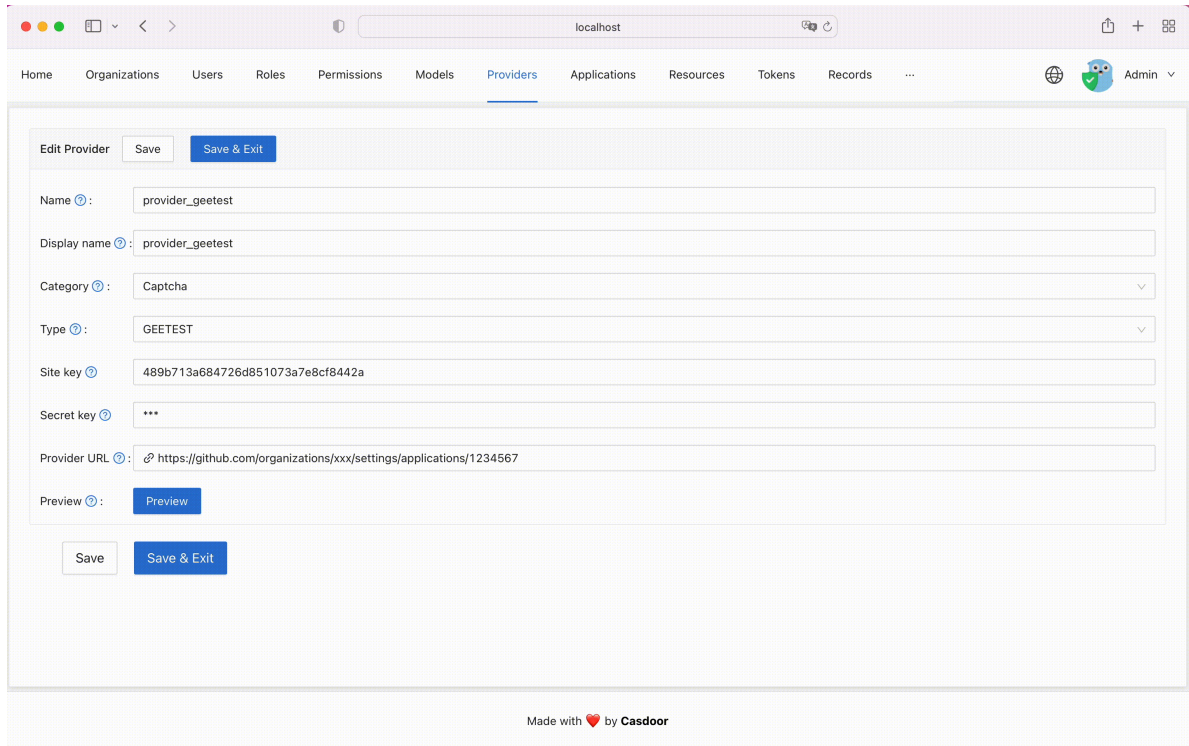
配置Casdoor

按照以下步骤配置Casdoor:

1. 在Casdoor中创建一个新的提供商。

将类别设置为**Captcha**，类型设置为**Geetest**。用从Geetest获取的ID和Key填写 **Site key** 和 **Secret key**。

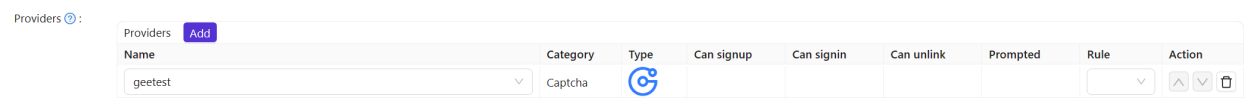
2. 点击预览按钮以预览此验证码的样式。



在您的应用程序中应用

在您的应用程序中应用Geetest配置：

编辑您希望在Casdoor中配置的应用程序。选择您刚刚添加的提供商，然后点击保存按钮。





提供商



Web3

Web3



MetaMask

将MetaMask Web3提供者添加到您的应用程序



Web3-Onboard

将Web3-Onboard Web3提供者添加到你的应用程序

MetaMask

📘 备注

这是一个如何配置MetaMask作为Web3提供者的示例。

MetaMask是一个浏览器扩展和应用程序，既可以作为加密货币钱包，也可以作为区块链应用的入口。Casdoor允许您使用MetaMask作为身份提供者，并启用MetaMask的Web3登录。

步骤1：创建一个MetaMask Web3提供者

首先，您需要在Casdoor中创建一个MetaMask Web3提供者。

名称	描述
Category	选择 Web3
Type	选择 MetaMask

Edit Provider Save Save & Exit

Name ?:

Display name ?:

Organization ?:

Category ?:

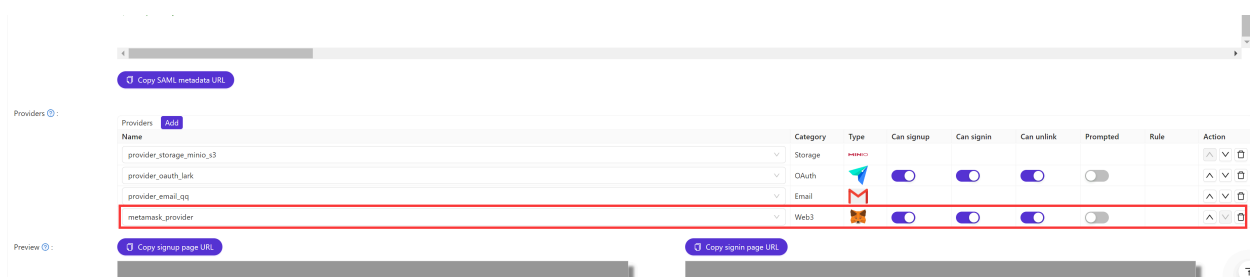
Type ?:

Provider URL ?:

Save Save & Exit

步骤2：将提供者添加到您的应用程序

接下来，将MetaMask Web3提供者添加到您的应用程序。

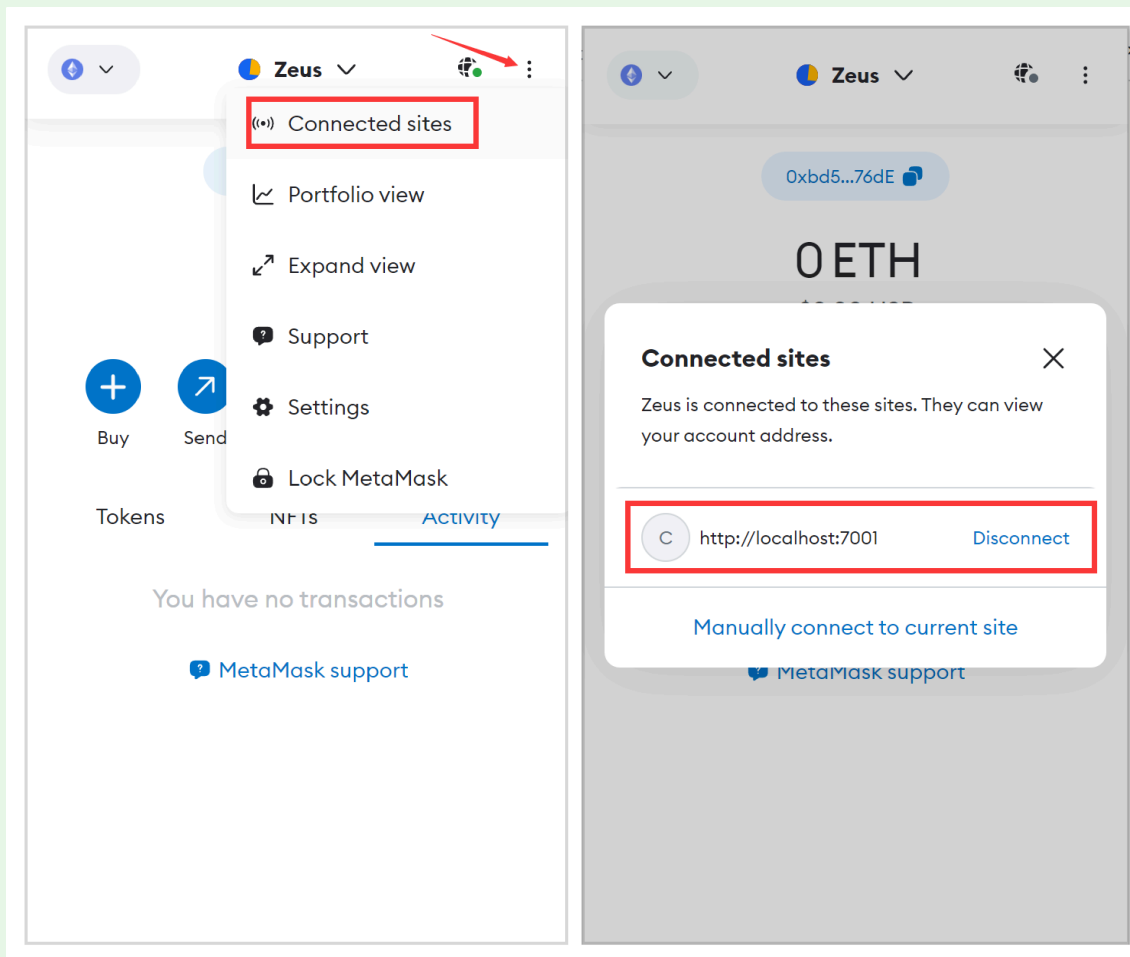


步骤3：使用MetaMask登录

您现在可以使用MetaMask登录了。这里有一个演示视频。

💡 提示

1. 使用MetaMask登录时，请只授权一个以太坊地址。 Casdoor每个用户只绑定一个以太坊地址。
2. 如果您想切换到另一个以太坊地址进行登录，请先断开当前以太坊地址与Casdoor的连接。

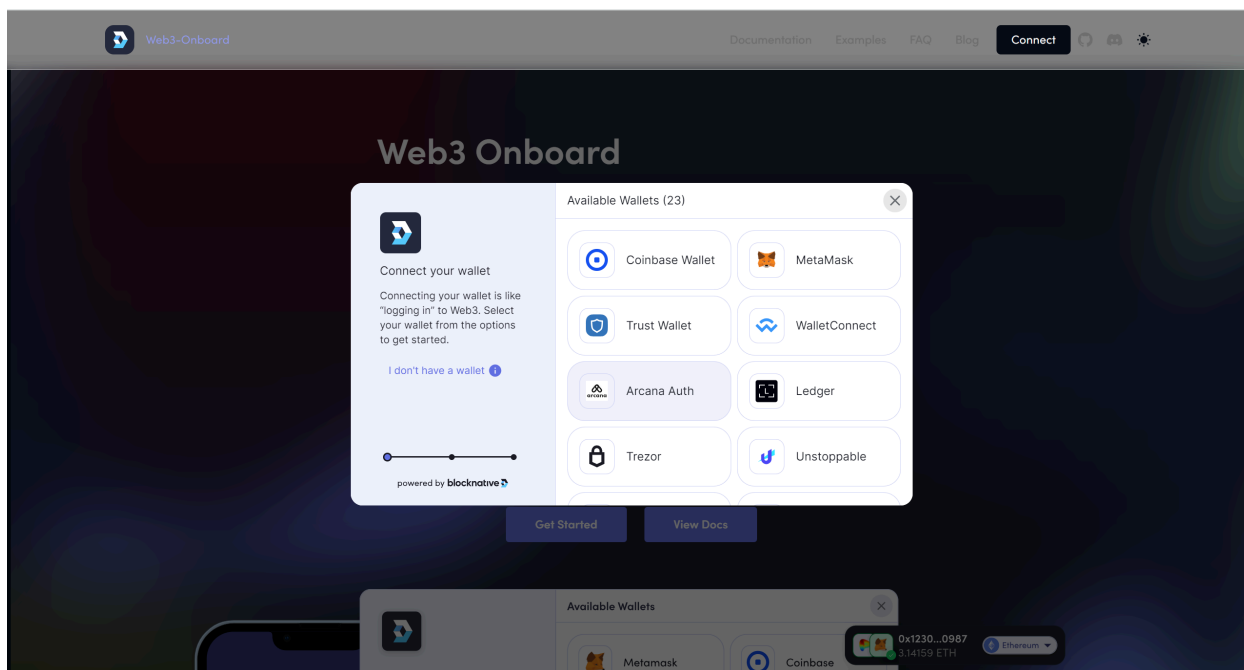


Web3-Onboard

📘 备注

这是一个如何配置Web3-Onboard作为Web3提供者的示例。

Web3-Onboard可以帮助用户使用不同的钱包进行Web3登录。Casdoor允许使用Web3-Onboard作为身份提供者，并启用Web3-Onboard的Web3登录。



步骤1: 创建一个Web3-Onboard Web3提供者

首先，你需要在Casdoor中创建一个Web3-Onboard Web3提供者。

名称	描述
类别	选择 Web3
类型	选择 Web3-Onboard
钱包	选择允许登录的钱包


Edit Provider Save Save & Exit

Name ⓘ: provider_web3_onboard

Display name ⓘ: Web3-Onboard Web3 Provider

Organization ⓘ: admin (Shared)

Category ⓘ: Web3

Type ⓘ:  Web3-Onboard

Wallets ⓘ: Injected Coinbase Trust Gnosis Sequence Tahoe Frontier Infinity Wallet

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


























Save Save & Exit

目前，Casdoor只支持上图中显示的钱包。**Injected** 钱包代表浏览器注入的钱包，如 **MetaMask** 或 **Coinbase**。

步骤2：将提供者添加到你的应用程序

其次，将Web3-Onboard Web3提供者添加到你的应用程序。

Providers 

Name	Category	Type	Can signup	Can signin	Can unlink	Prompted	Rule	Action
provider_storage_minio_s3	Storage							  
provider_oauth_lark	OAuth		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		  
provider_email_qq	Email							  
provider_web3_metamask	Web3		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		  
provider_google_oauth	OAuth		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	One Tap 	  
provider_web3_onboard	Web3		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		  

步骤3：通过Web3-Onboard登录

现在你可以通过Web3-Onboard登录。 这里有一个演示视频。

资源

 概述

上传资源到Casdoor

概述

您可以在Casdoor中上传资源。在上传资源之前，您需要配置存储提供商。请参阅[存储提供商](#)部分以获取更多信息。

一旦您配置了至少一个[存储提供商](#)并将其添加到您的[应用程序](#)中，您就可以继续进行了。

Providers ⓘ :

Name	Category	Type	ca
Provider_azure	Storage		
Github_1	OAuth		
provider_Alipay	Payment		

太棒了! 现在让我们来看一个如何[上传](#)和[删除](#)资源的例子。

上传资源

用户可以上传各种类型的资源，如文件和图片，到您配置的[云存储](#)中。


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

Resources [Upload a file...](#)

Provider	Created time	Tag	
provider_storage_aliyun_oss	source/casbin/leo220yuyaodog/2022_ICM_Problem_D.pdf	2022-05-18 17:25:21	custom
provider_storage_aliyun_oss	source/built-in/admin/美的2021&22Q1交流.pdf	2022-05-18 12:28:01	custom
provider_storage_aliyun_oss	source/casbin/admin/solo.svg	2022-05-17 16:25:39	custom

删除资源

如果您不再需要某个特定资源，您可以选择点击“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		Copy Link	Delete

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)。 以下将指导您如何添加产品的过程。

配置产品属性

首先，您需要理解产品的基本属性：




- 标签
- 详情
- 货币
- 价格
- 数量
- 已售出

Tag  :	Casdoor Summit 2022
Detail  :	This is a description
Currency  :	USD
Price  :	19
Quantity  :	100
Sold  :	10

支付服务提供商

除了设置这些属性外，您还需要为产品添加支付提供商。可以向产品添加多个支付提供商。

要了解如何配置支付提供商，请参考[支付提供商](#)

Payment providers  :	provider_Alipay ×
Return URL  :	 http://localhost:8000/products/callback



最后，填写**返回URL**。这是付款完成后，付款提供商页面将重定向的URL。

预览产品

你已经完成了！ 查看详细信息并保存：

Preview :

[Test buy page..](#)


Buy Product					
Name	Product				
Detail	This is a subscription.	Tag	Casdoor Summit 2022	SKU	product
Image					
Price	\$300 (USD)	Quantity	99	Sold	10
Pay					

支付

付款成功处理后，您将能够在**支付**部分查看产品的交易信息。此信息将包括诸如组织、用户、购买时间和产品名称等详细信息。

发票

要开具发票，请导航至编辑屏幕：

Type ▾	Product ▾ 🔍	Price ▾ 🔍	Currency	Action
	A notebook computer	300	USD	Result Edit Delete

在编辑屏幕上，您需要填写相关的发票信息。有两种可用的发票类型：`individual` 和 `organization`。

要完成此过程，只需点击**"发票"**按钮。

如果您有任何其他问题或疑虑，请告诉我们。



定价

定价

概述

Casdoor定价概述

计划

Casdoor计划概述

定价概览

Casdoor定价概览

订阅

Casdoor订阅概览

概述

Casdoor可以通过其 `Plan`、`Pricing` 和 `Subscription` 功能作为订阅管理系统使用。

您可以选择将哪些计划包含在您的价格列表中，如下图所示：

Casdoor Pricing

Casdoor hosting services provided by Casbin Inc.

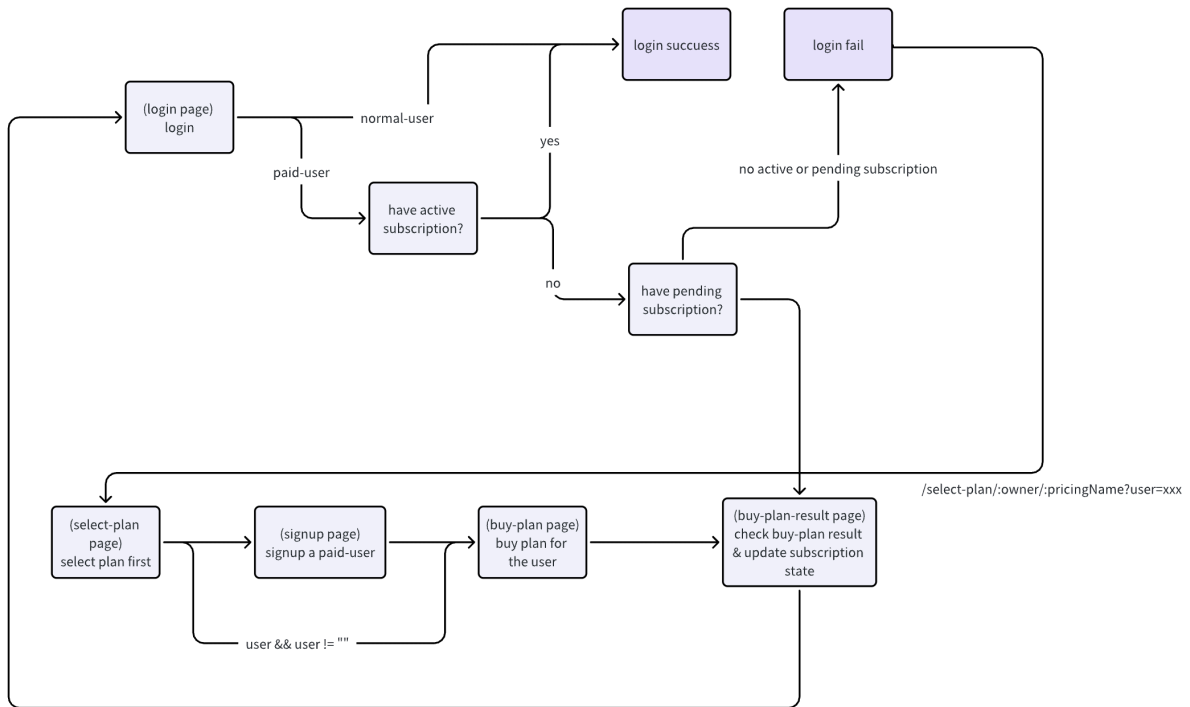
Basic Plan	Premium Plan	Enterprise Plan
\$ 10.01 per month	\$ 20.02 per month	\$ 30.03 per month
For small teams, with limited technical support	For fast growing start-ups, with full technical support	For large & medium-sized enterprise, with full technical support
Getting started	Getting started	Getting started

Free 7-days trial available!

每个 `Pricing` 都属于一个特定的 `Application`。用户可以选择一个计划，并通过 `Pricing` 的相应 `pricing page URL` 注册为 `paid-user`。

一般流程

一般流程看起来像这样：



1. 用户通过访问管理员共享的 `pricing page URL` 进入 `Pricing` 的选择计划页面。

Casdoor Pricing
Casdoor hosting services provided by Casbin Inc.

Basic Plan	Premium Plan	Enterprise Plan
\$ 10.01 per month	\$ 20.02 per month	\$ 30.03 per month
For small teams, with limited technical support	For fast growing start-ups, with full technical support	For large & medium-sized enterprise, with full technical support
Getting started	Getting started	Getting started

2. 用户选择一个 `Plan` 进行订阅并完成注册过程，成为一个 `paid-user`。

Casdoor Pricing

Casdoor hosting services provided by Casbin Inc.



Basic Plan	Premium Plan	Enterprise Plan
\$ 10.01 per month For small teams, with limited technical support	\$ 20.02 per month For fast growing start-ups, with full technical support	\$ 30.03 per month For large & medium-sized enterprise, with full technical support
Getting started	Getting started	Getting started

Free 7-days trial available!



Username:

Display name:



* Password:  

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



3. 注册后，用户将被重定向到所选 **Plan** 的购买计划页面，以继续付款。

Buy Product

Name	Auto Created Product for Plan built-in/plan_premium(Premium Plan)				
Detail	This Product was auto created for Plan built-in/plan_premium(Premium Plan)	Tag	auto_created_product_for_plan	SKU	product_6g2mcm
Image					
Price	\$20.02 (USD)	Quantity	999	Sold	0
Pay					

4. 一旦付款成功完成，用户对 Plan 的 Subscription 就被激活。现在，用户可以作为 paid-user 登录到 Casdoor。



You have successfully completed the payment: Auto Created Product for Plan built-in/plan_premium(Premium Plan)

Please click the below button to return to the original website

[Return to Website](#)

这里有一个演示视频：

计划

`Plan` 描述了一个应用的一系列功能，每个功能都有自己的名称和价格。

`Plan` 的功能取决于Casdoor的 `Role`，它附带一组 `Permissions`。

这允许独立描述 `Plan` 的功能，无论命名和定价如何。

例如，`Plan` 的价格可能会根据国家或日期的不同而有所不同。

下图说明了 `Plan` 和 `Role` 之间的关系。

Plan

- Display Name
- Price per month
- ...

Role

permission 1
permission 2
...
permission N

计划属性

每个 **Plan** 都有以下属性：

- 组织
- 名称
- 创建时间
- 显示名称
- 角色
- 每月价格
- 货币
- **PaymentProviders**：用户可以通过支付提供商购买 **Plan**。有关如何配置支付提供商的信息，请参见[支付提供商](#)。
- 是否启用

The screenshot shows the 'Edit Plan' interface. At the top, there are three buttons: 'Edit Plan', 'Save', and 'Save & Exit'. The form contains the following fields:

- Organization: built-in
- Name: plan_enterprise
- Display name: Enterprise Plan
- Role: (empty)
- Description: For large & medium-sized enterprise, with full technical support
- Price per month: 30.03
- Price per year: 100
- Currency: USD
- Payment providers: provider_payment_stripe x provider_payment_paypal x
- Is enabled:

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

当通过Casdoor创建 **Plan** 时，会自动创建一个相关的 **Product**。


为 **Plan** 配置的信息将自动同步到 **Product**。

当用户购买 **Plan** 时，他们实际上是购买了所选 **Plan** 的相关 **Product**。

Name: product_49fak

Display name: Auto Created Product for Plan built-in/plan_enterprise(Enterprise Plan)

Organization: built-in

Image:
URL: https://cdn.casbin.org/img/casdoor-logo_1185x256.png
Preview: 

Tag: auto_created_product_for_plan

Detail: This Product was auto created for Plan built-in/plan_enterprise(Enterprise Plan)

Description:

Currency: USD

Price: 30.03

Quantity: 999

Sold: 0




Payment providers: provider_payment_stripe x provider_payment_paypal x

Return URL: [https://example.com](#)

State: Published

Preview: [Test buy page](#)

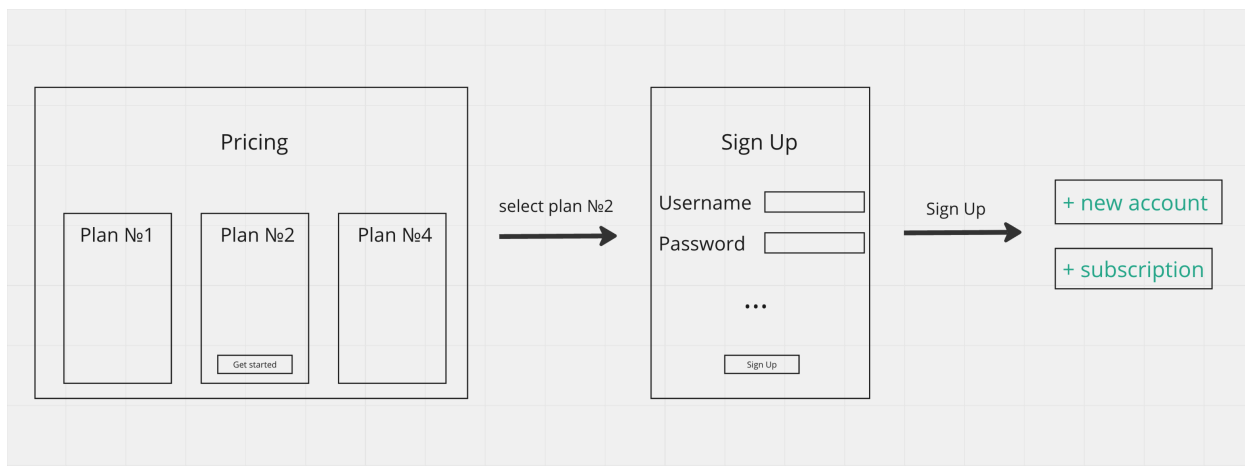
Buy Product

Name	Auto Created Product for Plan built-in/plan_enterprise(Enterprise Plan)				
Detail	This Product was auto created for Plan built-in/plan_enterprise(Enterprise Plan)	Tag	auto_created_product_for_plan	SKU	product_49fak
Image					
Price	\$30.03 (USD)	Quantity	999	Sold	0
Pay	 				

定价概览

定价功能包含一个或多个计划选项，允许用户以不同的价格点注册应用。

下图描述了定价选项的一般流程：



定价属性

每个定价订阅都有以下属性：

- 组织
- 名称
- 创建时间
- 显示名称
- 描述
- 计划：计划数组。
- 是否启用

- 应用

要查看定价界面的示例，请参考下图：

The screenshot displays a pricing configuration interface on the left and its corresponding preview on the right.

Configuration Interface:

- Organization: built-in
- Name: pricing_casdoor
- Display name: Casdoor Pricing
- Description: Casdoor hosting services provided by Casbin Inc.
- Application: app-built-in
- Plans: plan_basic, plan_premium, plan_enterprise
- Trial duration: 7
- Is enabled:
- Preview: [Copy pricing page URL](#)

Pricing Preview:

Casdoor Pricing

Casdoor hosting services provided by Casbin Inc.

Plan	Price	Description	Action
Basic Plan	\$ 10.01 per month	For small teams, with limited technical support	Getting started
Premium Plan	\$ 20.02 per month	For fast growing start-ups, with full technical support	Getting started
Enterprise Plan	\$ 30.03 per month	For large & medium-sized enterprise, with full technical support	Getting started

订阅

订阅功能有助于管理用户选择的计划，使得控制对应用功能的访问变得容易。

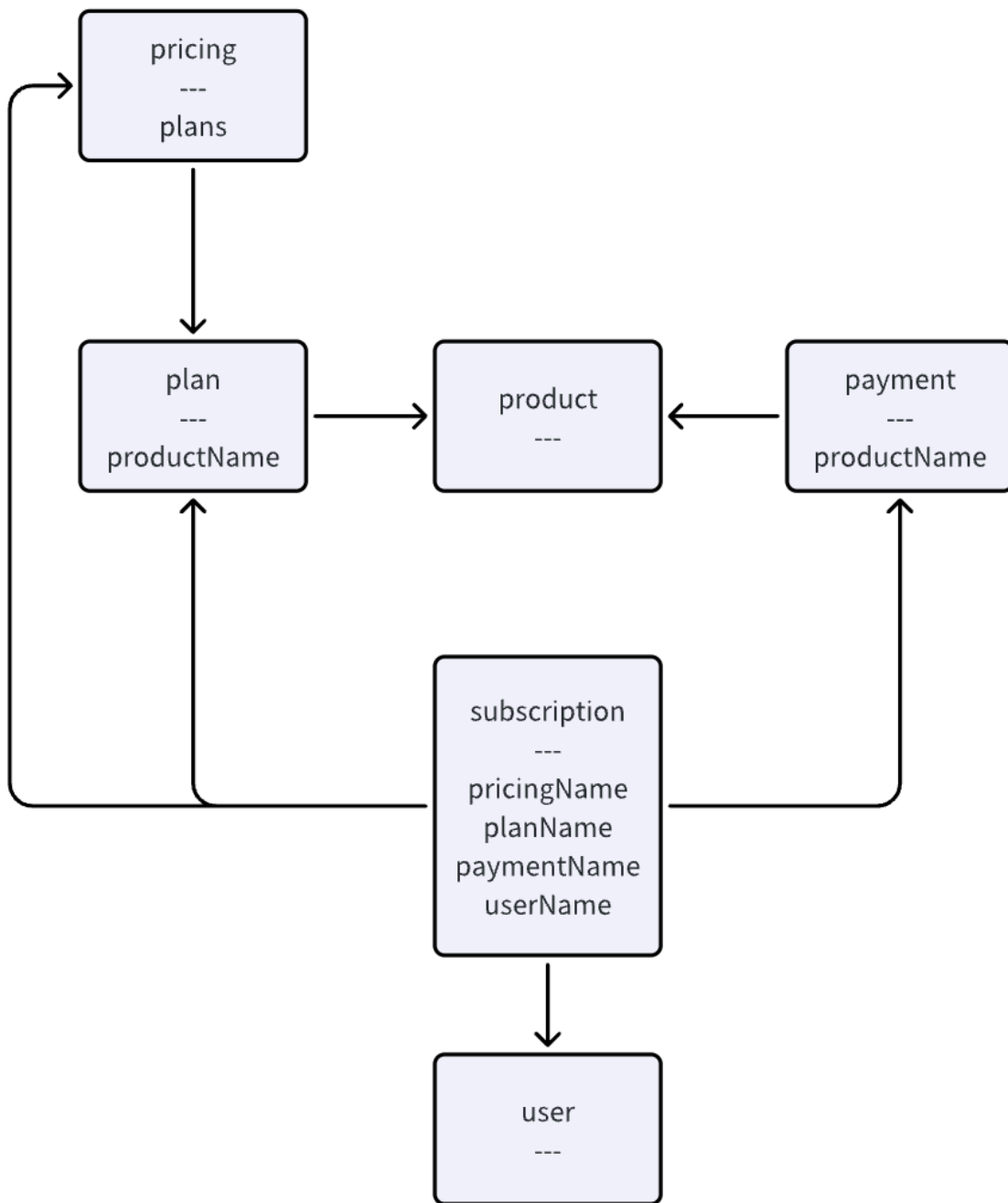
💡 提示

由于每个计划都基于一个角色，您可以将计划的角色分配给用户，并使用强制API进行权限检查。

订阅可以通过三种方式创建：

- 由管理员手动创建
- 通过定价流程（在注册为付费用户并购买所选计划后）
- 通过API

定价、计划、订阅、产品和支付之间的关系如下：



订阅属性

每个订阅都有以下属性：

- 所有者
- 名称
- 创建时间
- 显示名称
- 描述
- 持续时间：订阅的持续时间。
- 开始时间：订阅生效的开始时间。
- 结束时间：订阅生效的结束时间。
- 定价：相关的定价。
- 计划：相关的计划。
- 支付：相关的支付。
- 用户：持有此订阅的用户。
- 状态：目前，订阅有以下状态：待处理、错误、暂停、活动、即将到来、过期。

Edit Subscription Save Save & Exit

Organization: built-in

Name: sub_e719e2

Display name: New Subscription - e719e2

Duration: 30

Start time: 2023-08-25

End time: 2023-09-24

User: paid-user-x

Pricing: pricing_casdoor

Plan: plan_premium

Payment: payment_20230825_160124_2d18867

Description:

State:

- Active
- Pending
- Active
- Upcoming
- Expired
- Error
- Suspended

Save Save & Exit

用户

概述

在Casdoor中管理用户

MFA / 2FA

使用MFA / 2FA保护您的账户

用户角色

分配给用户的角色

权限

用户权限

概述

用户属性

作为一个认证平台，Casdoor 能够管理用户。每个用户都有以下属性：

- `Owner`：拥有用户的组织
- `Name`：唯一的用户名
- `CreateTime` 创建时间
- `UpdateTime`
- `Id`：每个用户的唯一标识符
- `Type`
- `Password`
- `PasswordSalt`
- `PasswordOptions`：密码复杂性选项
- `DisplayName`：在用户界面中显示
- `FirstName`
- 姓氏
- `Avatar`：链接到用户的头像
- 永久头像
- 电子邮件
- 电话
- 位置
- 地址
- 隶属

- 标题
- 身份证类型
- 身份证
- 主页
- 生物
- 标签
- 区域
- 语言
- 性别
- 生日
- 教育
- 得分
- 因果
- 排名
- IsDefaultAvatar
- IsOnline
- IsAdmin: 表示用户是否是其组织的管理员
- IsGlobalAdmin: 表示用户是否有权限管理Casdoor
- IsForbidden
- IsDeleted
- 注册应用程序
- 哈希
- 预哈希
- 创建的IP
- 最后登录时间
- 最后登录IP

- Roles: 用户角色的数组
- Permissions: 用户权限的数组

社交平台登录的唯一ID:

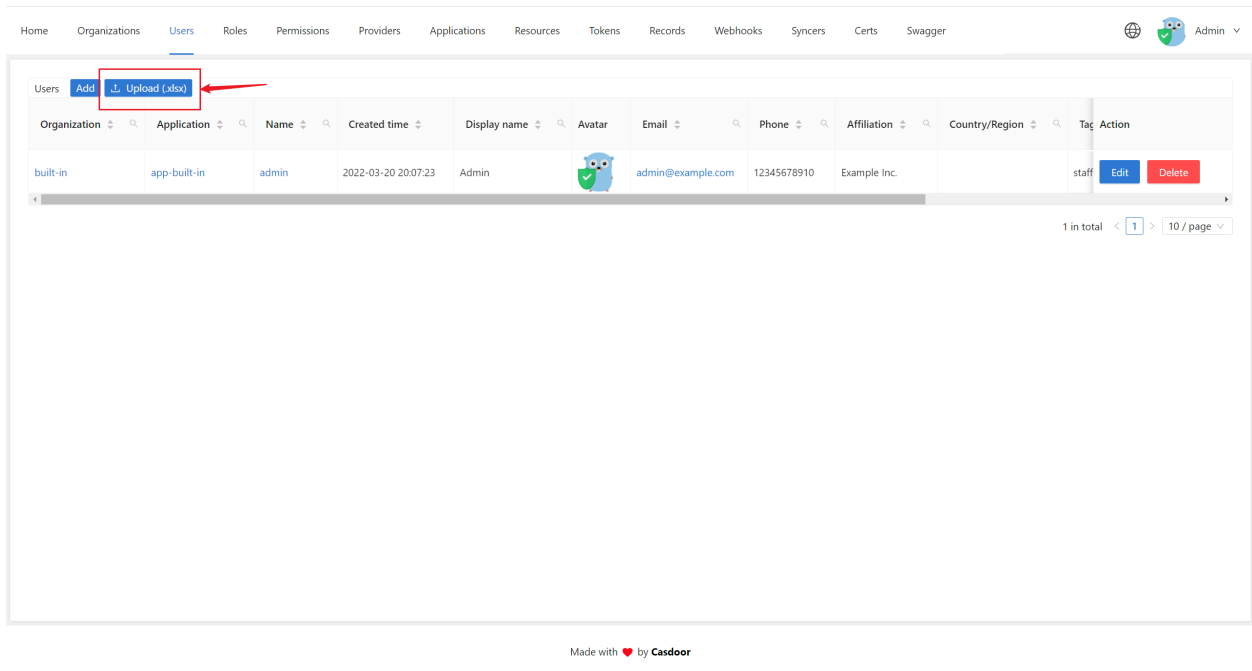
- Github
- Google
- QQ
- 微信
- Facebook
- 钉钉
- 微博
- Gitee
- LinkedIn
- Wecom
- Lark
- Gitlab
- Adfs
- Baidu
- Casdoor
- Infoflow
- Apple
- Azure AD
- Azure AD B2C
- Slack
- Steam
- Ldap

- **Properties**: 一个字符串→字符串映射，用于存储任何额外的属性。

从XLSX文件导入用户

您可以通过上传包含用户信息的XLSX文件来添加新用户或更新现有的Casdoor用户。

在管理员控制台中，转到用户并点击**上传 (.xlsx)** 按钮。



选择你的XLSX文件并点击打开。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。该模板包含5个测试用户和一些必需用户属性的标题。

Home Organizations Users Roles Permissions Providers Applications Users uploaded successfully, refreshing the page Syncers Certs Swagger Admin

Users [Add](#) [Upload \(xlsx\)](#)

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	scier	Edit Delete
built-in	app-built-in	gauss	2022-03-20 20:48:33	Carl Friedrich Gauss		vqdsan@example.com	98621462844	Example Inc.	Germany	math	Edit Delete
built-in	app-built-in	galileo	2022-03-20 20:47:58	Galileo Galilei		8p4f38@example.com	22596937332	Example Inc.	Italy	scier	Edit Delete
built-in	app-built-in	euler	2022-03-20 20:47:08	Leonhard Euler		3dzw4j@example.com	74409642681	Example Inc.	Switzerland	math	Edit Delete
built-in	app-built-in	einstein	2022-03-20 20:46:29	Albert Einstein		z6mive@example.com	60062541396	Example Inc.	Germany	scier	Edit Delete
built-in	app-built-in	admin	2022-03-20 20:07:23	Admin		admin@example.com	12345678910	Example Inc.		staff	Edit Delete

6 in total < 1 > 10 / page

Made with ❤️ by Casdoor

绕过密码加密

当从外部数据库迁移用户到Casdoor时，可能会出现您希望绕过或控制由 `organization` 默认密码类型方法提供的默认加密方法的情况。

这可以通过在用户导入过程中使用 `passwordType` 字段来实现。

备注

使用Bycrypt密码的用户

以下是API路由 `/api/add-user` 的POST体请求示例。

```
{
  "owner": "组织",
  "signupApplication": "first-app",
  "email": "dev@dev.com",
```

在这里，用户的密码已经使用bcrypt算法进行了加密，所以我们将 `passwordType` 指定为"bcrypt"，以通知Casdoor不要再次加密它。

MFA / 2FA

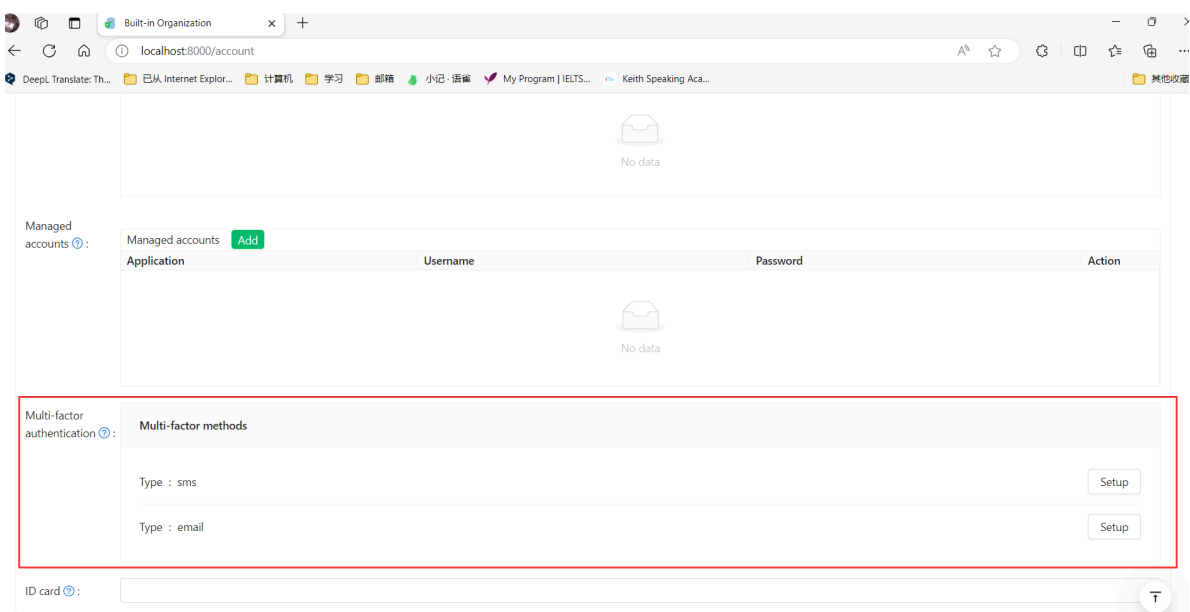
关于多因素身份验证

MFA（多因素身份验证）是一种可以增强用户和系统安全性的安全措施。它要求用户在登录或执行敏感操作时提供两个或更多的身份验证因素。

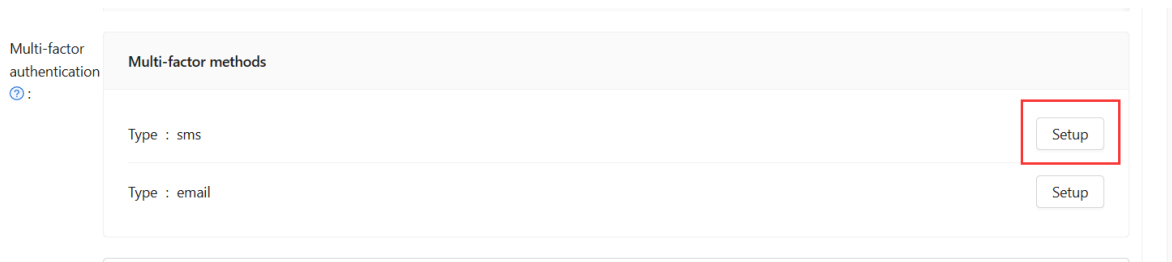
对于Casdoor，第二种身份验证形式是通过短信或电子邮件发送的代码。一旦您启用MFA，Casdoor每次有人试图登录您的账户时都会生成一个身份验证代码。只有知道您的密码并能获取身份验证代码的人才能登录您的账户。

配置MFA

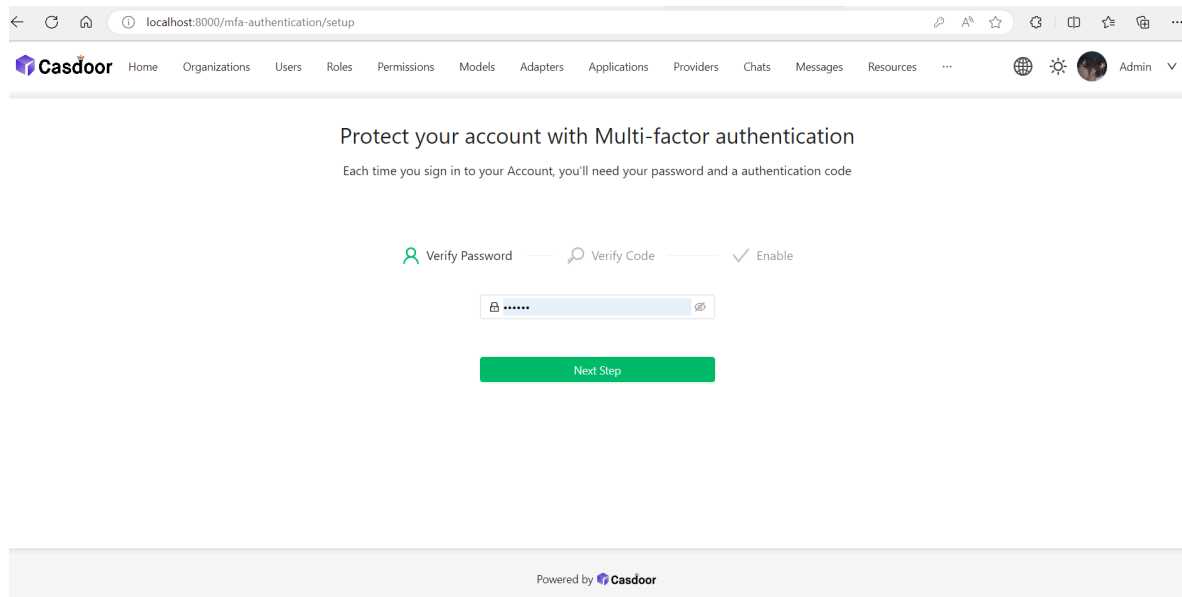
1. 在用户个人资料页面，您可以看到多因素身份验证的配置。如果您看不到它，请确保组织在账户项目表中添加了多因素身份验证项目。



2. 点击“设置”按钮。



3. 输入您的密码并点击“下一步”。



使用TOTP移动应用配置多因素身份验证

基于时间的一次性密码（TOTP）应用自动生成一个在一定时间后更改的身份验证代码。我们建议使用：

- [Google Authenticator](#)
- [Microsoft Authenticator](#).

提示

To configure authentication via TOTP on multiple devices, during setup, scan the QR code using each device at the same time. 如果2FA已经启用，而您想添加另一个设备，您必须从用户个人资料页面重新配置您的TOTP应用。

Protect your account with Multi-factor authentication

Each time you sign in to your Account, you'll need your password and a authentication code

Verify Password — Verify Code ✓ Enable



Scan the QR code with your authenticator app

Or copy the secret to your authenticator app

P757K7XT5MIO5RPZQYSC



Passcode

Next Step

Use email Use SMS

1. 在“验证代码”步骤中，执行以下操作之一：
 - 使用您的移动设备的应用扫描二维码。扫描后，应用会显示一个六位数的代码，您可以在Casdoor上输入。
 - 如果您无法扫描二维码，您可以手动复制并在您的TOTP应用中输入密钥。
2. TOTP移动应用将您的Casdoor账户保存并每隔几秒生成一个新的身份验证代码。在

Casdoor上，将代码输入到“验证码”字段并点击“下一步”。

3. 在“启用”按钮上方，复制您的恢复代码并将它们保存到您的设备上。将它们保存到一个安全的位置，因为您的恢复代码可以帮助您在失去访问权限时重新获得对您账户的访问。

Protect your account with Multi-factor authentication

Each time you sign in to your Account, you'll need your password and a authentication code

 Verify Password —  Verify Code —  Enable

Please save this recovery code. Once your device cannot provide an authentication code, you can reset mfa authentication by this recovery code

ad30de29-3ce0-4e39-a97f-ceff1d503d3c

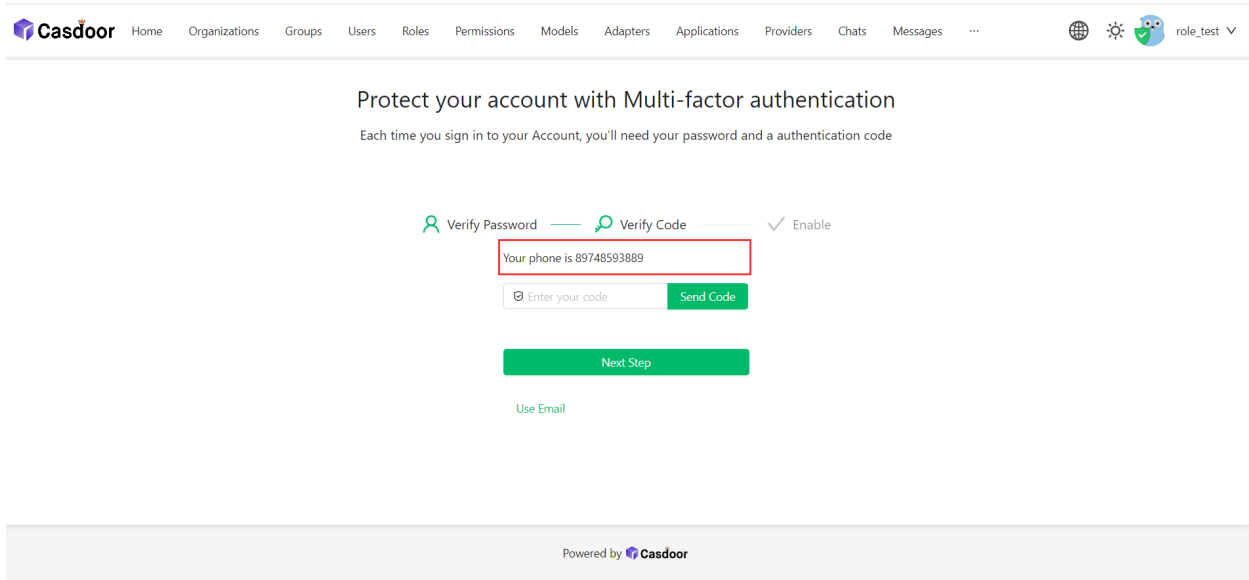
Enable

注意事项

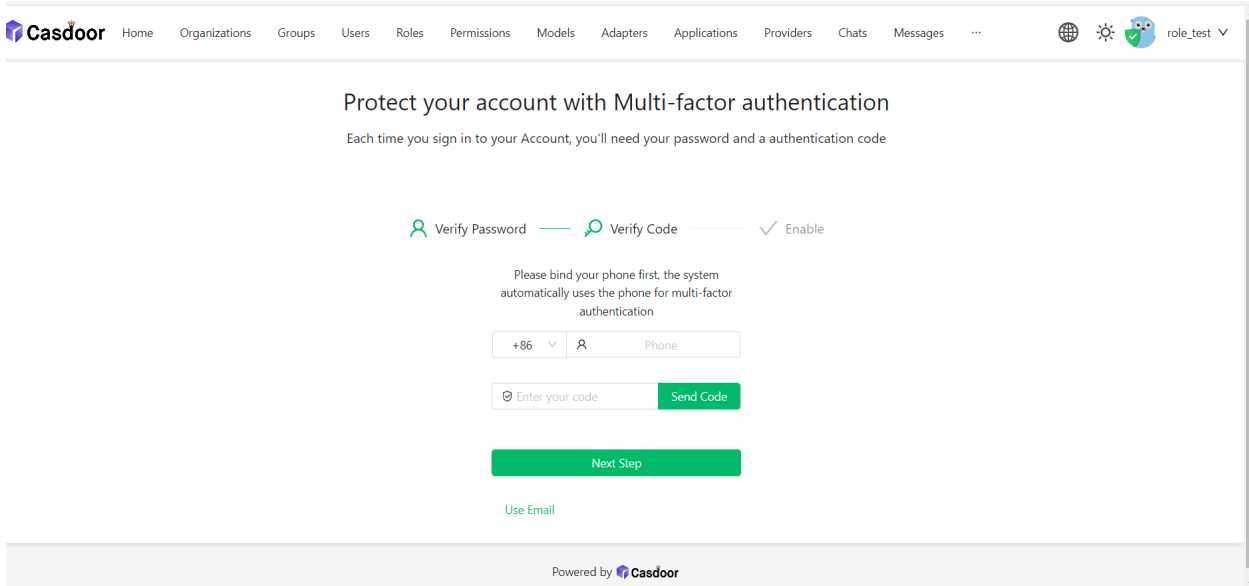
Each recovery code can only be used once. 如果您使用恢复代码登录，它将变为无效。

使用短信配置多因素身份验证

如果您已添加您的手机号码，Casdoor将使用它向您发送短信。



如果您还没有添加您的手机号码，您需要先添加它。



1. 选择您的国家代码并输入您的手机号码。
2. 检查您的信息是否正确并点击“发送代码”。
3. 您将收到一条带有安全代码的短信。然后将代码输入到“输入您的代码”字段并点击“下一步”。

4. 在“启用”按钮上方，复制您的恢复代码并将它们保存到您的设备上。将它们保存到一个安全的位置，因为您的恢复代码可以帮助您在失去访问权限时重新获得对您账户的访问。

使用电子邮件配置多因素身份验证

将电子邮件配置为您的多因素身份验证方法与使用短信类似。

1. 使用您当前的电子邮件或输入您的电子邮件地址并点击“发送代码”。
2. 然后将代码输入到“输入您的代码”字段并点击“下一步”。
3. 在“启用”按钮上方，复制您的恢复代码并将它们保存到您的设备上。将它们保存到一个安全的位置，因为您的恢复代码可以帮助您在失去访问权限时重新获得对您账户的访问。

更改您首选的MFA方法

您可以添加多种MFA方法。只有首选方法才会在您登录时使用。

如果您想设置首选的MFA方法，请点击“设置首选”按钮。

Multi-factor authentication ⓘ: Multi-factor methods Disable

Type : sms 132****0885 Enabled preferred

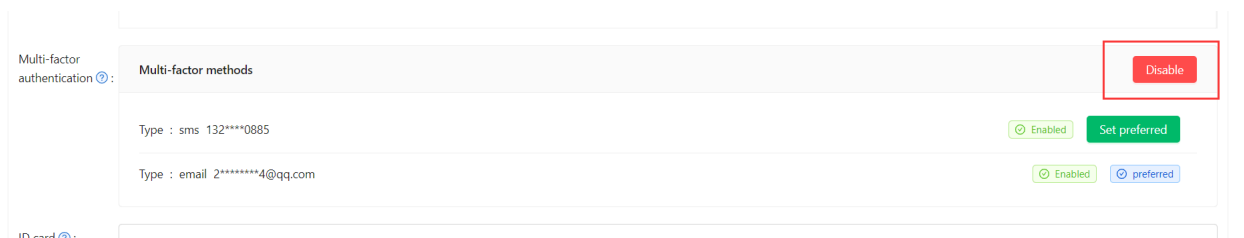
Type : email 2*****4@qq.com Enabled Set preferred

ID card ⓘ:

您首选的方法上将显示“首选”标签。

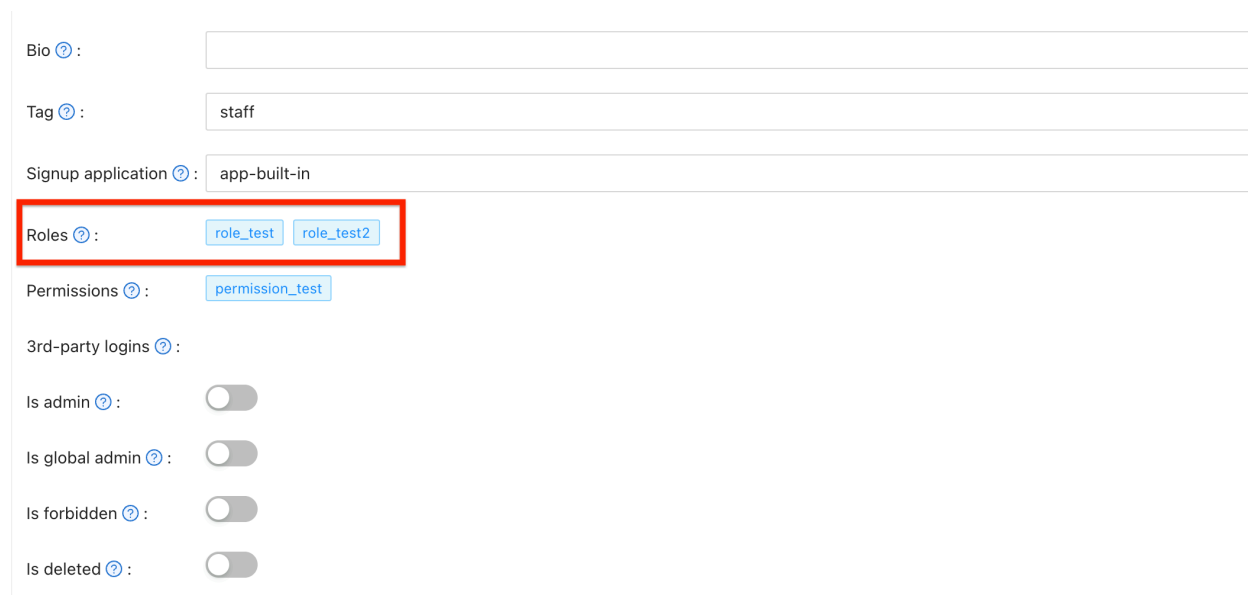
禁用多因素身份验证

如果您想禁用多因素身份验证，请点击“禁用”按钮。您的所有多因素身份验证设置将被删除。



用户角色

每个用户可以拥有多个角色。您可以在用户的个人资料上查看分配给他们的角色。



The screenshot shows a user profile form with the following fields:

- Bio:
- Tag:
- Signup application:
- Roles: (This field is highlighted with a red box)
- Permissions:
- 3rd-party logins:
- Is admin:
- Is global admin:
- Is forbidden:
- Is deleted:

角色属性

每个角色都有以下属性:

- 所有者
- 名称
- 创建时间
- 显示名称
- 已启用
- `Users`: 属于此角色的子用户数组
- `Roles`: 属于此角色的子角色数组

权限

每个用户可能有多个权限。您可以在他们的个人资料上查看用户的权限。

Bio ⓘ:

Tag ⓘ:

Signup application ⓘ:

Roles ⓘ:

Permissions ⓘ:

3rd-party logins ⓘ:

Is admin ⓘ:

Is global admin ⓘ:

Is forbidden ⓘ:

Is deleted ⓘ:

权限属性

权限具有以下属性:

- 所有者
- 名称
- 创建时间
- 显示名称
- 已启用
- 模型
- Users: 此角色的子用户数组

- Roles: 此角色的子角色数组
- 资源类型
- Resources: 资源的数组
- Actions: 一个动作数组
- 效果



邀请码

邀请码



概述

在casdoor中管理邀请

概述

目前，casdoor已经支持更灵活的邀请码方式进行用户注册。一旦管理员以邀请码为必选项打开注册页面，用户只有拥有有效的邀请码才能注册。

Signup items [Add](#)

Name	Visible	Required	Prompted	Label	Placeholder	Regex	Rule
ID							Random
Username	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					
Display name	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					None
Password	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					
Confirm	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					
Email	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					Normal
Phone	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					None
Agreement	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					Only signup
Invitation code	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					

使用邀请码的主要有两种方式，默认添加的是一个随机字符串码，由随机数字和字母组成。为了更灵活，邀请码还支持正则匹配来匹配多个不同的邀请码。

Invitations [Add](#)

Name	Organization	Updated time	Display name	Code	Quota	Used count	Application	Action
invitation_794tdt	built-in	2024-02-04 20:52:15	New Invitation - 794tdt	319jed4tjk	1	0	All	Edit Delete
invitation_147y39	built-in	2024-02-04 20:47:47	New Invitation - 147y39	[a-zA-Z]2333	1	0	All	Edit Delete

2 in total < 1 > 10 / page

邀请属性

Casdoor通过以下属性管理邀请

- **Organization**: 拥有邀请的组织
- **Name**: 唯一的邀请名称
- **Display name**: 显示的邀请名称

- **Code**：邀请码，你可以填写具体的邀请码字符串，也可以填写正则表达式
- **Default code**：用于在邀请链接中填充默认的邀请码。对于随机生成的邀请码，默认码与邀请码相同。对于正则表达式形式的代码，你需要自己填写符合代码中的正则表达式规则的默认代码
- **Quota**：邀请码可以使用的最大次数
- **Used count**：邀请码已被使用的次数
- **Application**：允许使用此邀请码的应用。选择**ALL**使其对组织下的所有应用都可用
- **Username**：使用此邀请注册时需要的特定用户名
- **Email**：使用此邀请注册时需要的特定电子邮件
- **Phone**：使用此邀请注册时需要的特定电话
- **State**：邀请的状态

默认邀请

默认邀请中的邀请码是一个随机生成的数字和字母的字符串，且**Quota**设置为1，只能使用一次。应用默认设置为**ALL**，这意味着对应此邀请的组织下的所有应用都可以使用此邀请码。

New Invitation

Organization

Name

Display name

Code

Default code

Quota

Used count

Application

Username

Email

Phone

State

如果邀请码是为特定用户设置的，你希望用户使用给定的 `username`、`email`、`phone` 和 `invitation code` 注册，你可以通过填写相应的字段来限制用户的注册。如果字段为空或者在注册页面上没有配置，casdoor不会强制验证这些字段

Username

Email

Phone

当需要重复使用邀请码时，你可以将 `Quota` 设置为一个较大的值，例如，如果你希望这个邀请码被使用10次，那么你可以将 `Quota` 设置为10。当你希望停止使用此邀请码注册时，你也可以通过修改邀请的状态为 `Suspended` 来实现。

Quota

Used count

Application

Username

Email

Phone

State

⚠ 注意事项

当在邀请中配置了 `username`、`email` 或 `phone` 时，`quota` 不应大于一。这是因为用户的 `username`、`email` 和 `phone` 应该是唯一的，多个用户不应该能够使用相同的 `username`、`email` 或 `phone` 进行注册。

正则匹配邀请

有时候需要大量的邀请码进行用户注册，一一生成邀请码可能非常低效。Casdoor支持通过正则表达式匹配来验证邀请码。例如，通过将 `Code` 设置为 `"[a-z]2333"`，任何匹配此正则表达式的邀请码都将被成功匹配为有效的邀请码。

Code ⓘ:	<input type="text" value="[a-z]2333"/>
Default code ⓘ:	<input type="text" value="a2333"/>
Quota ⓘ:	<input type="text" value="2"/>
Used count ⓘ:	<input type="text" value="0"/>

ℹ 备注

在使用正则表达式验证邀请码时，每个匹配正则表达式的邀请码只能使用一次，`Quota` 仍然可以限制使用次数。例如，当 `Code` 是 `"[a-z]2333"` 且 `Quota` 是2时，只有最多两个匹配正则表达式的邀请码可以成功使用。

邀请链接

Casdoor支持复制对应邀请的邀请链接。邀请链接中的邀请码对应于Default code字段。因此，对于使用正则表达式的邀请，必须手动填写Default code以生成正确的邀请链接。此外，当使用邀请链接注册时，注册页面将自动填充由对应邀请码的邀请设置的某些字段信息。

New Invitation

Organization

Name

Display name

Code

Default code

Quota

localhost:7001/signup/app-built-in?invitationCode=a2333



* Username:

* Password:

* Email:

* Phone:

* Invitation code:

[Have account? sign in now](#)

Powered by Casdoor

演示



同步器

同步器

概述

在Casdoor中同步用户

数据库

使用数据库同步器同步数据库

Keycloak

使用 Keycloak Syncer 同步 Keycloak

WeCom

使用WeCom Syncer同步数据库

概述

作为一个认证平台，Casdoor可以轻松管理存储在数据库中的用户。

同步器

Casdoor将用户存储在`user`表中。所以，当你计划使用Casdoor作为认证平台时，无需担心将你的应用程序的用户数据迁移到Casdoor。Casdoor提供了一个**同步器**，可以快速帮助您将用户数据同步到Casdoor。

您需要指定要与Casdoor同步的数据库和用户表，同步器将定期负责同步数据。有关更多详细信息，请参阅[数据库同步器](#)。

同步哈希值

Casdoor使用哈希函数来确定如何更新用户。此哈希值是根据表中每个用户的信息（如密码或手机号码）计算得出的。

如果特定**Id**的用户的计算哈希值与原始值相比发生了变化，Casdoor确认用户表已经被更新。随后，数据库更新旧信息，从而实现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

我们创建的用户表作为演示，是从[模板XLSX文件](#)导入的。

owner	name	created_time	updated_time	id	type	password	password_salt	display_name	first_name	last_name	avatar	permanent_avata_email
built-in	einstein	2022-03-20T20:46:29+08:00		1c57cc37-37f5-4def-9e9f-082189ef63d2	normal-user	123		Albert Einstein			https://casbin.org	z6mwe@
built-in	euler	2022-03-20T20:47:08+08:00		bb7831b4-0d24-4e96-b043-fbfd8d15eb	normal-user	123		Leonhard Euler			https://casbin.org	3dzw4j@
built-in	galileo	2022-03-20T20:47:58+08:00		7920eb6c-f9f5-40ef-8e18-3ac9949bdf5	normal-user	123		Galileo Galilei			https://casbin.org	8p4f3@
built-in	gauss	2022-03-20T20:48:33+08:00		f0c288f6-2c0d-479b-b545-cb4c9f6db36	normal-user	123		Carl Friedrich Gau			https://casbin.org	vqdsan@
built-in	tesla	2022-03-20T20:49:03+08:00		687c3068-fd21-4d32-b2ba-e13e8b369a	normal-user	123		Nikola Tesla			https://casbin.org	9v73hn@

要创建新的同步器，请转到[同步器](#)选项卡，并按照下面的示例填写所有必需的信息。然后，保存更改。

Organization:

Name:

Type:

Host:

Port:

User:

Password:

Database type:

Database:

Table:

Table columns:

Column name	Column type	Casdoor column	Is key	Is hashed	Action
<input type="text" value="name"/>	<input type="text" value="string"/>	<input type="text" value="Name"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="text" value="↑"/> <input type="text" value="↓"/> <input type="text" value="🗑"/>
<input type="text" value="id"/>	<input type="text" value="string"/>	<input type="text" value="Id"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="text" value="↑"/> <input type="text" value="↓"/> <input type="text" value="🗑"/>
<input type="text" value="first_name"/>	<input type="text" value="string"/>	<input type="text" value="FirstName"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="text" value="↑"/> <input type="text" value="↓"/> <input type="text" value="🗑"/>

💡 提示

一般来说，你需要至少在Casdoor列中填写 **ID** 和 **Name**。其他重要字段包括

`createdTime`、`Password`和`DisplayName`。

以下字段是必需的：

- `Organization`：用户将被导入的组织
- `Name`：同步器名称
- `Type`：选择"数据库"
- `Host`：原始数据库主机地址
- `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，如MySQL、PostgreSQL、SQL Server、Oracle和SQLite
- `Database`：原始数据库名称
- `Table`：原始用户表名称
- `表格列`
- `Column name`：原始用户列名称
- `Column type`：原始用户列类型
- `Casdoor Column`：Casdoor用户列名称

可选字段：

- `Is hashed`：是否计算哈希值。当此选项启用时，同步器只有在原始表中的用户字段更新时才会同步用户。如果此选项被禁用，即使只更新了字段，同步器仍然会同步用户。简而言之，只有在参与哈希计算的字段（启用"Is hashed"）更新时，用户才会被同步。
- `Is key`：是否是原始表中的用户和Casdoor表中的用户的主键。在同步数据库时，根据选择了"Is key"选项的字段来确定。字段的"Is key"按钮应至少选择一个。如果没有选择，那么默认选择第一个"Is key"选项。

- **Avatar base URL**: 当同步用户时, 如果Avatar base URL不为空且原始user.avatar没有"http"前缀, 新的user.avatar将被Avatar base URL + user.avatar替换。
- **Affiliation table**: 用于从数据库中的此表同步用户的隶属关系。由于隶属关系可能是"Affiliation table"中的int类型代码, 因此需要映射为字符串。参考[getAffiliationMap\(\)](#)。Casdoor有一些冗余字段可以借用, 所以[这里](#)我们使用**score**将int代码映射为字符串名称。

配置好同步器后, 启用**Is enable**选项并保存。同步器将开始工作。

Name	Organization	Created time	Type	Host	Port	User	Password	Database type	Database	Action
syncer_qmpox9	built-in	2023-08-09 18:57:36	Database	localhost	3306	root	password	mysql	auth	<input type="button" value="Sync"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>

你也可以使用"Sync"按钮进行数据库同步。

更新

当**Table columns**设置如下图所示时, 执行更新操作。


Column name	Column type	Casdoor column	Is key	Is hashed	Action
name	string	Name	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	^ v 🗑
id	string	Id	<input type="checkbox"/>	<input checked="" type="checkbox"/>	^ v 🗑
first_name	string	FirstName	<input type="checkbox"/>	<input checked="" type="checkbox"/>	^ v 🗑
password	string	Password	<input type="checkbox"/>	<input checked="" type="checkbox"/>	^ v 🗑













如果两个表中的关键数据不同, 可以根据主键在两个表之间同步数据。

- 在原始表中更新用户
- 在Casdoor表中更新用户

添加

当 `Table columns` 设置如下图所示时，执行添加操作。

Table columns :

Column name	Column type	Casdoor column	Is key	Is hashed	Action
<input type="text" value="name"/>	string	Name	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	  
<input type="text" value="id"/>	string	Id	<input type="checkbox"/>	<input checked="" type="checkbox"/>	  
<input type="text" value="first_name"/>	string	FirstName	<input type="checkbox"/>	<input checked="" type="checkbox"/>	  
<input type="text" value="password"/>	string	Password	<input type="checkbox"/>	<input checked="" type="checkbox"/>	  

如果两个表之间的数据数量不同，可以根据主键将数据添加到数据量较少的表中。

- 在原始表中添加用户
- 在Casdoor表中添加用户

Keycloak

Keycloak 同步器

Keycloak同步器本质上与数据库同步器相同，增加了对Keycloak的Tables和Table columns的自动配置功能。

此外，Keycloak同步器将从credential表，keycloak_group表和user_group_membership表中获取数据，因为Keycloak中的用户信息是存储在多个表中的。

Organization: built-in

Name: keycloak

Type: Keycloak

Host: localhost

Port: 3306

User: root

Password: root

Database type: MySQL

Database: keycloak

Table: user_entity **configured automatically after selecting Keycloak as syncer type**

Table primary key:

Table columns:

Column name	Column type	Casdoor column	Is hashed	Action
ID	string	Id	<input checked="" type="checkbox"/>	⬆️ ⬇️ ⬇️
USERNAME	string	Name	<input checked="" type="checkbox"/>	⬆️ ⬇️ ⬇️
USERNAME	string	DisplayName	<input checked="" type="checkbox"/>	⬆️ ⬇️ ⬇️
EMAIL	string	Email	<input checked="" type="checkbox"/>	⬆️ ⬇️ ⬇️
EMAIL_VERIFIED	boolean	EmailVerified	<input checked="" type="checkbox"/>	⬆️ ⬇️ ⬇️
FIRST_NAME	string	FirstName	<input checked="" type="checkbox"/>	⬆️ ⬇️ ⬇️
LAST_NAME	string	LastName	<input checked="" type="checkbox"/>	⬆️ ⬇️ ⬇️
CREATED_TIMESTAMP	string	CreatedTime	<input checked="" type="checkbox"/>	⬆️ ⬇️ ⬇️
ENABLED	boolean	IsForbidden	<input checked="" type="checkbox"/>	⬆️ ⬇️ ⬇️

WeCom

WeCom Syncer

通过使用WeCom syncer，您可以将WeCom用户和部门数据同步到Casdoor的用户表和组表。

以下字段是必需的：

- **Organization**：将导入用户的组织
- **Name**：syncer的名称
- **Type**：选择"WeCom"
- **User**：您的WeCom公司ID
- **Password**：您的WeCom应用程序密钥
- **ClientSecret**：您的WeCom联系人同步密钥

按照以下步骤进行配置。

步骤1：获取WeCom Syncer配置项

- 在您的WeCom管理平台中，导航到**我的公司**，在**公司信息**中获取**公司ID**。

WeCom Service Provider Console | API Documentation | CSR | Quit

Homepage Contacts Collaboration App Management Customers and Partners Advanced Features Security and Management My Company

Company Information

Company Logo: usher
Recommended size: 702*180 [Go to verify](#)

Company short name: usher 未认证 [Modify](#) Company not verified. After verification, the number of users can be increased.

Company address: [Add](#)

Phone No.: [Add](#)

Company Domain Na...: [Add](#)

Company member: 1 member(s) [Statistics](#)

Company Department: 1 dept(s)

Added/Max.: 1/1000 [Verify now to increase the limit](#)

Invoice Title: [Add](#) Set VAT invoice titles for company members [?](#)

Industry Type: Internet and Related Services [Modify](#)

Company Size: 1-50 [Modify](#)

Creation time: 2023-6-18

Company ID: ww752595f99d89b1ca
Already a WeCom service provider. [Go to Service Provider Platform](#)

- 在您的自建应用中，获取应用密钥。

WeCom Service Provider Console | API Documentation | CSR | Quit

Homepage Contacts Collaboration App Management Customers and Partners Advanced Features Security and Management My Company

Casdoor

Enabled

AgentId: 100003 [Edit](#)

Secret: [View](#)

Allowed users: usher

- 在联系人同步管理工具中，获取 **联系人同步密钥**。

The screenshot shows the 'Sync of Contacts' configuration page in the WeCom Service Provider Console. The page has a navigation bar at the top with 'WeCom' on the left and 'Service Provider Console | API Documentation | CSR | Quit' on the right. Below the navigation bar are several menu items: 'Homepage', 'Contacts', 'Collaboration', 'App Management', 'Customers and Partners', 'Advanced Features', 'Security and Management', and 'My Company'. The main content area is titled 'Sync of Contacts' and includes a 'Back' button. The page contains several sections:

- Sync of Contacts**: A green icon with a person and a plus sign. Below it, the text reads: 'Companies can sync contacts through APIs or authorized third-party service providers [API Documentation](#)'.
- Sync method**: 'API Sync'.
- Permission**: 'Read and edit Contacts' with links for 'Edit' and 'View Permission Details'.
- Secret**: A red box highlights the 'Secret' field and its associated 'View' and 'Send again' buttons.
- Company's Trusted IP**: '3 IP address(es) are configured. [Settings](#)'. Below this, it says: 'Only configured IP addresses can access company data via API.'
- Set event receiving server**: 'The added member or department will be pushed to the following URL in the form of event to ensure the Contacts is synced. [Learn More](#)'.
- Disable API sync**: A link to disable API sync.

步骤2：配置Casdoor WeCom Syncer

转到Syncers选项卡，选择 **WeCom** 类型并填写如下所示的必需信息。然后，保存更改。

Edit Syncer

Save

Save & Exit

Organization ⓘ: built-in

Name ⓘ: wecom_syncer

Type ⓘ: WeCom

Database type ⓘ: MySQL

Host ⓘ: 

Port ⓘ: 0

User ⓘ: ww752595f99d89b1ca Company ID

Password ⓘ:  App secret

Client secret ⓘ:  Sync of contacts secret

Database ⓘ:

Table ⓘ:



令牌

令牌



概述

Casdoor令牌简介

概述

Casdoor是基于OAuth构建的，并使用令牌作为用户的OAuth令牌。


以下是Casdoor中可用的令牌字段：


- Owner
- Name
- CreatedTime
- Application
- Organization
- User
- Code
- AccessToken
- ExpireIn (令牌将在几小时后过期)
- Scope (授权范围)
- TokenType (例如, Bearer 类型)


登录应用程序后，有三种生成JWT令牌选项：


- JWT
- JWT-Empty
- JWT-Custom


选项如下：JWT将生成一个包含所有用户字段的令牌，JWT-Empty将生成一个包含用户所有非空值的令牌，而JWT-Custom将生成一个包含自定义用户令牌字段的令牌（您可以在令牌字段中选择属性）。


Token format : JWT-Custom


Token fields : Owner x CreatedTime x DisplayName x UpdatedTime x


Token expire : Owner ✓


Refresh token expire : CreatedTime ✓

Failed signin limit : UpdatedTime ✓

Failed signin frozen time : Id

Failed signin frozen time : Type

Failed signin frozen time : Password

Failed signin frozen time : PasswordSalt

Owner	✓
Name	
CreatedTime	✓
UpdatedTime	✓
Id	
Type	
Password	
PasswordSalt	



Webhooks

Webhooks

概述

在Casdoor中添加Webhooks

概述

概述

事件系统使您能够创建集成，订阅Casdoor中的特定事件。当触发这些事件之一时，将通过POST请求将JSON负载发送到配置的URL。该应用程序将解析JSON负载并执行指定的函数。事件包括注册、登录、登出和用户更新，所有这些都存储在记录的动作字段中。事件系统可以用来更新用户的外部问题。



LDAP

LDAP

概述

Casdoor与LDAP服务器合作

配置和同步LDAP用户

在Casdoor中配置LDAP以进行用户同步

LDAP 服务器

如何在Casdoor中连接LDAP客户端

概述

Casdoor已经引入了对LDAP服务器的支持。Casdoor能够将用户从LDAP服务器同步到Casdoor，以便将它们用作登录的用户账户，并使用LDAP服务器对它们进行身份验证。Casdoor还支持设置定时任务，以定期自动同步用户。

关于Casdoor-LDAP同步机制的详细信息

Casdoor如何与LDAP服务器合作的方式如下所述：

1. 同步：Casdoor可以连接到LDAP服务器，获取用户信息，并读取所有用户信息（包括uidNumber、uid、cn、gidNumber、mail、email、emailAddress、telephoneNumber、mobile、mobileTelephoneNumber、registeredAddress、postalAddress）。然后，它会为LDAP中的每个用户创建Casdoor账户，并将这些账户存储在数据库中。
2. 身份验证：正如我们所看到的，Casdoor并不获取LDAP用户的密码。当一个从LDAP服务器同步的账户试图通过Casdoor登录时，Casdoor不会检查存储在其数据库中的密码，而是连接到LDAP服务器并验证用户的密码是否正确。
3. 用户识别：Casdoor使用uid来唯一识别用户。因此，请确保每个LDAP用户都有一个唯一的uid。

一旦用户被同步到Casdoor中，他们的信息就会与LDAP用户独立开来。这意味着，如果你在Casdoor中修改了用户的信息，LDAP中的用户信息将不会被修改，反之亦然（除了LDAP用户的密码，因为我们依赖它来验证用户）。

配置和同步LDAP用户

LDAP配置因每个组织而异，因为LDAP用户将被同步到这些组织中。

要修改配置，您需要使用全局管理员用户。在“组织”页面上输入以下信息进行LDAP用户同步。

LDAP ⓘ:

LDAP	添加	LDAP服务器	服务器	基本DN	自动同步	最近同步	操作
Example LDAP Server		example.com:389		ou=People,dc=example,dc=com	Disable		同步 编辑 删除

配置连接到LDAP服务器

Edit LDAP Save Save & Exit Sync LDAP

Organization ⓘ:

ID ⓘ:

Server name ⓘ:

Server host ⓘ:

Server port ⓘ:

Enable SSL ⓘ:

Base DN ⓘ:

Search Filter ⓘ:

Filter fields ⓘ:

Admin ⓘ:

Admin Password ⓘ:

Auto Sync ⓘ: mins

服务器名称

管理者可以用来识别不同服务器的友好名称。

Example:

服务器主机

LDAP服务器的主机名或IP地址。

Example: `example.com`

服务器端口

LDAP服务器的端口号。只允许数字值。

Example: `389`

基础 DN

Casdoor在LDAP中搜索时默认使用Sub搜索模式。Base DN是用于搜索的基本可分辨名称。Casdoor将返回指定Base DN下的所有用户。

在Casdoor中配置的管理员账户应至少具有对基础DN的只读权限。

Example: `ou=Example,dc=example,dc=com`

搜索过滤器

Casdoor使用搜索过滤器来查询LDAP用户。

Example: `(objectClass=posixAccount)`

过滤字段

过滤字段是用户在LDAP服务器中的标识符。当以LDAP用户身份登录Casdoor时，输入的登录用户名被视为LDAP用户的`uid`。您也可以配置其他字段，例如`mail`或`mobile`。

The screenshot shows the Casdoor LDAP configuration page. At the top, there are navigation tabs: Home, Organizations, Users, Roles, Permissions, Models, Adapters, Applications, Providers, Chats, Messages, Resources, and Admin. Below the navigation, there are three buttons: 'Edit LDAP', 'Save', and 'Save & Exit' (highlighted in green), and a 'Sync LDAP' button. The configuration fields are as follows:

- Organization: built-in
- ID: 691edec0-f1ab-4e23-8f9f-a824a383032f
- Server name: Example LDAP Server
- Server host: [Redacted]
- Server port: 389
- Enable SSL: [Toggle Off]
- Base DN: ou=built-in,dc=example,dc=com
- Search Filter: (objectClass=inetOrgPerson)
- Filter fields: [Empty]
- Admin: cn=admin,dc=example,dc=com
- Admin Password: [Redacted]

管理员

可以登录到指定LDAP服务器的账户。

登录方法（DN或ID）取决于您想要连接的LDAP服务器的设置。

Example: `cn=manager,dc=example,dc=com`

管理员密码

LDAP服务器管理员账户的密码。

自动同步

设置为 0 以禁用自动同步。任何其他值表示每隔几分钟同步一次。

同步用户

同步表显示了在特定的 **ou** 中从LDAP服务器获取的所有用户。如果用户已经被同步，复选框将被禁用。您可以通过勾选框来选择用户，然后从LDAP服务器同步所选用户。

<input type="checkbox"/>	CN	UidNumber / Uid	Group Id	Email	Phone	Address
<input type="checkbox"/>	zhan san	1000 / zsan	500			
<input type="checkbox"/>	li si	1001 / lsi	500			
<input type="checkbox"/>	a dmin	1002 / admin	500			
<input type="checkbox"/>	tom brown	1007 / jery	500			
<input type="checkbox"/>	wrie jerry	1003 / wjerry	500			
<input type="checkbox"/>	admin2	1004 / admin2	500			
<input type="checkbox"/>	yyyy	1005 / yyyy	500			

< 1 > 10 / page v

⚠ 注意事项

如果LDAP服务器中的用户的 **uid** 与Casdoor组织中现有用户的 **name** 相同，Casdoor将创建一个新用户，其 **name** 包含 **uid** 和一个随机字符串。然而，由于新同步的用户名称在LDAP服务器中不存在，此用户可能无法登录。因此，建议避免这种情况。

LDAP 服务器

许多系统，如 `Nexus`，支持LDAP认证。Casdoor也实现了一个简单的LDAP服务器，支持绑定和搜索操作。

本文档描述了如何连接到Casdoor中的LDAP服务器并实现简单的登录认证。

LDAP 服务器端口号

LDAP服务器默认监听 `389` 端口。您可以通过修改 `conf/app.conf` 中的 `ldapServerPort` 值来更改默认端口。

工作原理

与Casdoor中的LDAP客户端类似，LDAP服务器中的用户都是 `posixAccount` 的子类。

当服务器接收到LDAP传输的一组数据时，它将解析 `cn` 和 `ou`，其中 `cn` 代表用户名，`ou` 代表组织名称。`dc` 并不重要。

如果是绑定操作，服务器将使用Casdoor来验证用户名和密码，并在Casdoor中授予用户权限。

如果是搜索操作，服务器将根据绑定操作授予客户端的权限，检查搜索操作是否合法，并返回响应。

❗ 信息

我们只支持简单认证。

如何绑定

在Casdoor LDAP服务器中，我们只认可类似于这种格式的DN：`cn=admin,ou=built-in,dc=example,dc=com`。

请将管理员用户的DN设置为上述格式。然后，您可以使用这个DN和用户的密码绑定到LDAP服务器，以登录到Casdoor进行验证。如果服务器验证成功，用户将在Casdoor中被授予权限。

如何搜索

一旦绑定操作成功完成，您就可以进行搜索操作。搜索和绑定操作之间存在一些差异。

- 要搜索某个用户，例如 `built-in` 组织下的 `Alice`，您应该使用类似于这样的DN：`ou=built-in,dc=example,dc=com`，并在过滤器字段中添加 `cn=Alice`。
- 要搜索某个组织下的所有用户，例如 `built-in` 中的所有用户，您应该使用类似于这样的DN：`ou=built-in,dc=example,dc=com`，并在过滤器字段中添加 `cn=*`。
- 要搜索所有组织中的所有用户（假设用户有足够的权限），您应该使用类似于这样的DN：`ou=*,dc=example,dc=com`，并在过滤器字段中添加 `cn=*`。



RADIUS

RADIUS

概述

将Casdoor用作RADIUS服务器

概述

您可以将Casdoor用作RADIUS服务器。RADIUS是一个客户端/服务器协议，客户端可以是NAS或运行RADIUS客户端软件的任何计算机。

Congigure

在部署Casdoor之前，您需要修改 `conf/app.conf` 文件中的RADIUS相关配置，包括服务器端口和密钥：

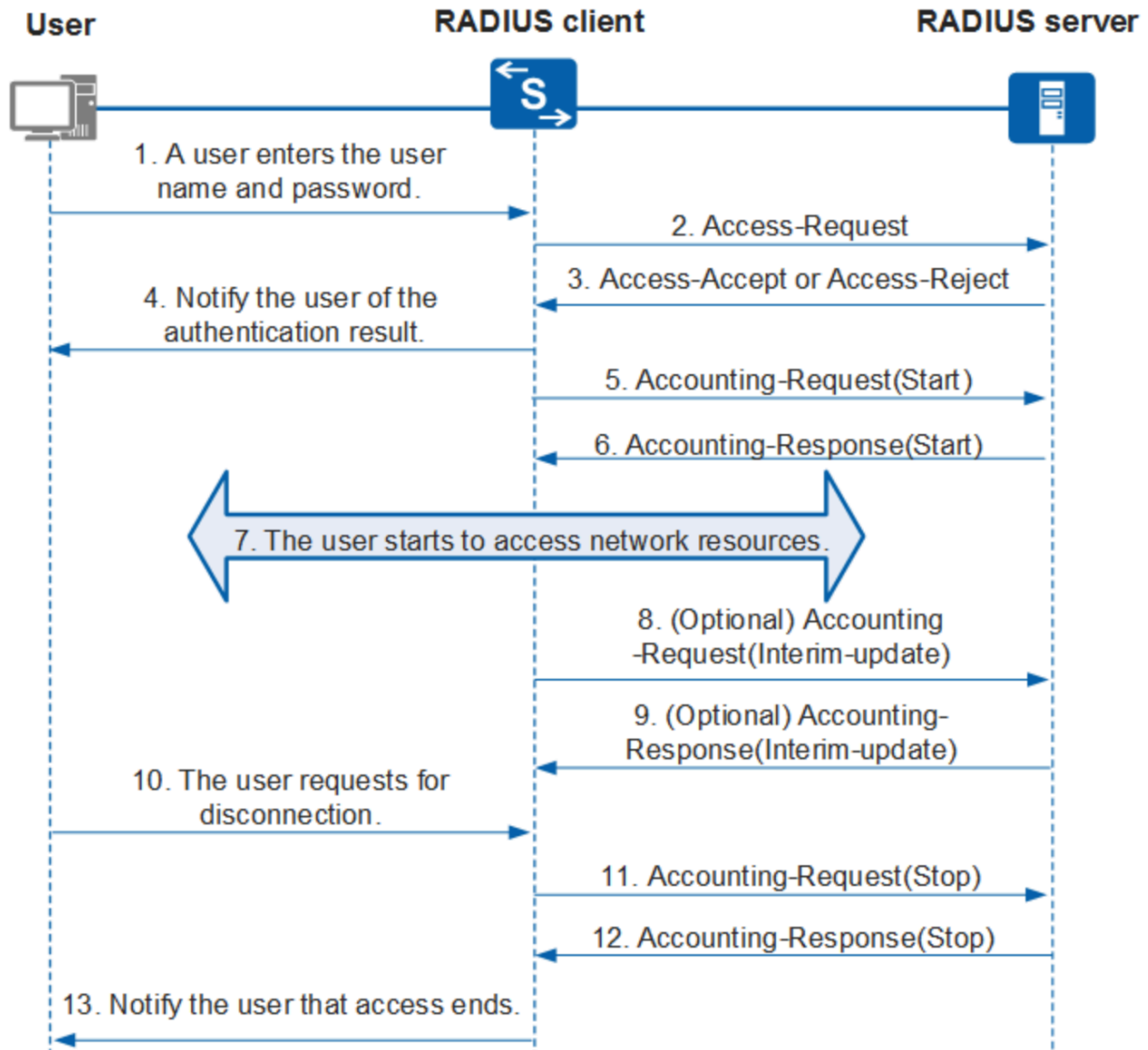
```
radiusServerPort = 1812
radiusSecret = "secret"
```

现在您可以将Casdoor用作RADIUS服务器。

将Casdoor用作RADIUS服务器

Casdoor目前可以支持以下标准RADIUS请求：

- `Access-Request`：认证请求消息由RADIUS客户端发送给Casdoor。Casdoor根据消息中携带的用户信息决定是否允许访问，并回复 `Access-Reject` 或 `Access-Accept`。
- `Accounting-Request`：当用户开始或停止访问网络资源时，RADIUS客户端将发送记账请求（开始/中间更新/停止）消息给Casdoor。Casdoor将记录相关的记账请求消息，并回复 `Accounting-Response`。



由于Casdoor使用组织来管理用户，每个用户都属于特定的组织，因此请求中的Class属性需要设置为用户的组织。

NTRadPing Test Utility

RADIUS Server/port: localhost 1812

Reply timeout (sec.): 3 Retries: 6

RADIUS Secret key: secret

User-Name: admin

Password: **** CHAP



Request type: Authentication Request 0

Additional RADIUS Attributes:
Class=built-in

Acct-Output-Octets 42

Add Remove Clear list Load... Save...

NTRadPing 1.5 - RADIUS Server Testing Tool
? 1999-2003 Master Soft SpA - Italy - All rights reserved
<http://www.dialways.com/>

RADIUS Server reply:

```
Sending authentication request to server localhost:1812
Transmitting packet, code=1 id=0 length=55
received response from the server in 16 milliseconds
reply packet code=2 id=0 length=20
response: Access-Accept
----- attribute dump -----
```

Send Help... Close

SCIM

 概述

将Casdoor用作SCIM服务提供商

概述

SCIM协议是一种基于HTTP的协议，用于通过SCIM模式指定的身份数据的配置和管理。您可以将Casdoor用作SCIM服务提供商。

将Casdoor用作SCIM服务提供商

目前Casdoor只支持 `User Resource Schema`，您可以通过SCIM用户操作来管理用户。您可以通过以下端点与Casdoor进行交互：

端点	方法	描述
<code>/scim/ServiceProviderConfig</code>	GET	提供关于支持的SCIM标准的特性的详细信息，例如，支持的资源。
<code>/scim/Schemas</code>	GET	提供有关服务提供商模式的详细信息。
<code>/scim/ResourceTypes</code>	GET	指定每个资源的元数据。
<code>/scim/Users/:id</code>	GET	使用资源标识符 <code>id</code> 检索用户。
<code>/scim/Users</code>	GET	使用查询参数查询用户（目前只支持 <code>startIndex</code> 和 <code>count</code> ）。
<code>/scim/Users</code>	POST	创建用户。
<code>/scim/Users/:id</code>	PUT	使用资源标识符 <code>id</code> 更新用户。

端点	方法	描述
/scim/Users/:id	PATCH	通过PATCH操作使用资源标识符 <code>id</code> 修改用户。
/scim/Users/:id	DEL	删除具有资源标识符 <code>id</code> 的用户。

有关更多详细信息，请参阅[rfc7644](#)。

用户资源

Casdoor实现了 `User Resource Schema` (SCIM) 和 `User` (Casdoor) 之间的映射。属性之间的映射关系如下：

用户资源模式 (SCIM)	用户 (Casdoor)
id	Id
meta.created	CreatedTime
meta.lastModified	UpdateTime
meta.version	UpdateTime
externalId	ExternalId
userName	Name
password	Password

用户资源模式 (SCIM)	用户 (Casdoor)
displayName	DisplayName
profileUrl	Homepage
userType	Type
name.givenName	FirstName
name.familyName	LastName
emails[0].value	Email
phoneNumbers[0].value	Phone
photos[0].value	Avatar
addresses[0].locality	Location
addresses[0].region	Region
addresses[0].country	CountryCode

由于Casdoor使用组织来管理用户，每个用户都属于特定的组织，所以应该在 `Enterprise User Schema Extension`（由模式URI `urn:ietf:params:scim:schemas:extension:enterprise:2.0:User` 标识）中传递 `organization` 属性。以下是用户资源模式SCIM在JSON格式中的表示：

```
{
  "active": true,
```


集成

 C++

3 个项目

 C#

1 个项目

 Go

9 个项目

 Java

17 个项目

 JavaScript

2 个项目

 Lua

1 个项目

 PHP

4 个项目

 Ruby

1 个项目

 Haskell

1 个项目

 Python

1个项目

C++

Nginx

使用Casdoor与Nginx

NginxCommunityVersion

使用Casdoor、Nginx（非Nginx-Plus）和Oauth2-Proxy

Envoy

在Envoy中使用Casdoor

Nginx

使用Casdoor作为身份提供者（IdP）为NGINX Plus代理的应用程序启用基于OpenID Connect的单点登录。

本指南解释了如何为NGINX Plus代理的应用程序启用单点登录（SSO）。该解决方案使用OpenID Connect作为身份验证机制，以Casdoor作为身份提供者（IdP），以及NGINX Plus作为依赖方。

另见：您可以在项目的GitHub仓库中找到有关NGINX Plus OpenID Connect集成的更多信息。

先决条件

说明假定您具有以下内容：

- 一个正在运行的Casdoor服务器。请参阅Casdoor文档中的[服务器安装和使用Docker尝试](#)。
- 一个NGINX Plus订阅和NGINX Plus R15或更高版本。有关安装说明，请参阅[NGINX Plus管理员指南](#)。
- [NGINX JavaScript模块](#)，用于处理NGINX Plus和IdP之间的交互。安装NGINX Plus后，使用适合您的操作系统的命令安装模块。

对于Debian和Ubuntu：

```
sudo apt install nginx-plus-module-njs
```

对于CentOS, RHEL和Oracle Linux:

```
sudo yum install nginx-plus-module-njs
```

- 以下指令应包含在顶级 (“主”) 配置上下文中的`/etc/nginx/nginx.conf`中, 以便加载NGINX JavaScript模块:

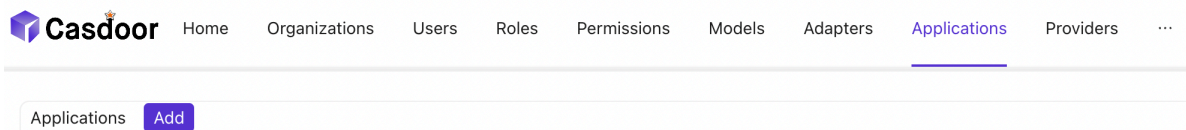
```
load_module modules/nginx_http_js_module.so;
```

配置Casdoor


注意: 以下过程反映了出版时的Casdoor GUI, 但GUI可能会发生变化。使用此指南作为参考, 并根据当前的Casdoor GUI进行调整。


要在Casdoor GUI中为NGINX Plus创建一个Casdoor客户端, 请按照以下步骤操作:

1. 在<http://your-casdoor-url.com/login/>登录您的Casdoor帐户。
2. 在顶部导航栏中, 点击**应用程序**。在打开的**应用程序**页面上, 点击左上角的**添加**按钮。





3. 在打开的**编辑应用程序**页面上, 将**名称**和**显示名称**字段的值更改为您要启用SSO的应用程序的名称。在这里, 我们使用的是NGINX Plus。

Name  :

Display name  :

在**重定向URL**字段中，输入NGINX Plus实例的URL，包括端口号，并以/_codexch结尾（在本指南中，它是https://your-site-url.com:443/_codexch）。

Redirect URLs  :


Redirect URL
 https://my-nginx.example.com:443/_codexch

注意：

- 对于生产环境，我们强烈建议您使用SSL/TLS（端口443）。
 - 即使您使用的是HTTP（80）或HTTPS（443）的默认端口，端口号也是必需的。
4. 记录**客户端ID**和**客户端密钥**字段中的值。您将在[配置NGINX Plus的步骤4](#)中将它们复制到NGINX Plus配置文件中。

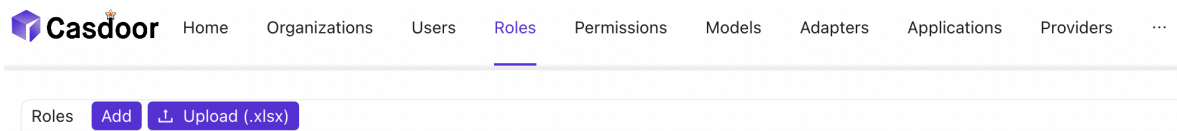
Client ID  :

200c96d5ce52785e74e0e11111111111

Client secret  :

58f13a80b875e775e0e0e0e0e0e0e0e0

5. 点击顶部导航栏中的**角色**，然后点击打开的页面左上角的**添加**按钮。



6. 在打开的**添加**页面上，在**名称**和**显示名称**字段中输入一个值（这里是nginx-casdoor-role），然后点击**保存**按钮。

Name  :


nginx-casdoor-role

Display name

nginx-casdoor-role

 :

7. 在顶部导航栏中，点击**用户**。在打开的**用户**页面上，点击**编辑**编辑现有用户，或者点击左上角的**添加**按钮创建新用户。
8. 在打开的**添加**页面上，根据您的喜好修改**名称**和**显示名称**（这里是user1）。

Name  :

user1

Display name

user1

 :


在注册应用程序中选择NGINX Plus。



Signup

NGINX Plus

application  :


在管理帐户字段中，选择应用程序中的NGINX Plus，并填写用户名和密码。

Managed accounts  :

Managed accounts 		
Application	Username	Password
NGINX Plus 	<input type="text"/>	<input type="password"/>

9. 返回到角色页面，点击nginx-casdoor-role行上的编辑。在打开的页面中，在子用户字段中，选择您刚刚创建的用户名（这里是built-in/user1）。

Sub users  :

built-in/user1 

配置NGINX Plus

要配置NGINX Plus作为OpenID Connect依赖方，请按照以下步骤操作：

1. 首先，创建[nginx-openid-connect](#) GitHub仓库的克隆：

```
git clone https://github.com/nginxinc/nginx-openid-connect
```

2. 将克隆中的以下文件复制到`/etc/nginx/conf.d`目录：

- `frontend.conf`
- `openid_connect.js`
- `openid_connect.server_conf`
- `openid_connect_configuration.conf`

3. 从Casdoor配置中获取授权端点、令牌端点和JSON Web Key (JWK) 文件的URL。打开终端并执行以下`curl`命令，将输出管道到指定的`python`命令以生成可读的配置格式。为了简洁，我们已经截断了输出，只显示相关字段。

```
curl http://<casdoor-server-address>/well-known/openid-configuration | python -m json.tool
{
  "authorization_endpoint": "https://<casdoor-server-address>/login/oauth/authorize",
  "...": "...",
  "token_endpoint": "http://<casdoor-server-address>/api/login/oauth/access_token",
  "...": "...",
  "jwks_uri": "http://<casdoor-server-address>/well-known/jwks",
  "...": "...",

```

4. 使用您喜欢的文本编辑器打开/etc/nginx/conf.d/

openid_connect_configuration.conf。修改以下map指令的"default"参数值为指定的值：

- map \$host \$oidc_authz_endpoint - 使用步骤3中的 authorization_endpoint 的值（在本指南中，https://<casdoor-server-address>/login/oauth/authorize）
- map \$host \$oidc_token_endpoint - 使用步骤3中的 token_endpoint 的值（在本指南中，http://<casdoor-server-address>/api/login/oauth/access_token）
- map \$host \$oidc_client - 使用配置Casdoor的步骤4中客户端ID字段的值
- map \$host \$oidc_client_secret - 使用配置Casdoor的步骤2中客户端密钥字段的值
- map \$host \$oidc_hmac_key - 使用一个独特的，长的，安全的短语

5. 根据使用的NGINX Plus版本配置JWK文件：

- 在NGINX Plus R17及更高版本中，NGINX Plus可以直接从步骤3中指定为 jwks_uri 的URL读取JWK文件。对/etc/nginx/conf.d/frontend.conf进行以下更改：
 - a. 注释掉（或删除）auth_jwt_key_file指令。
 - b. 取消注释auth_jwt_key_request指令。（参数/_jwks_uri引用了 \$oidc_jwt_keyfile 变量的值，该值将在下一步中设置。）
 - c. 将map \$host \$oidc_jwt_keyfile 指令的"default"参数更新为步骤3中 jwks_uri 字段的值（在本指南中，http://<casdoor-server-address>/.well-known/jwks）。
- 在NGINX Plus R16及更早版本中，或者如果您喜欢，JWK文件必须位于本地磁盘上。按照以下步骤操作：
 - a. 将步骤3中 jwks_uri 字段指定的JWK文件的JSON内容复制到本地文件（例如，/etc/nginx/my_casdoor_jwk.json）。

- b. 在`/etc/nginx/conf.d/openid_connect_configuration.conf`中，将`map $host $oidc_jwt_keyfile`指令的"default"参数更改为本地文件的路径。
6. 确保在NGINX Plus配置文件（通常是`/etc/nginx/nginx.conf`）中指定的`user`指令的用户具有读取JWK文件的权限。

测试

打开浏览器并输入您的NGINX Plus实例的地址。然后，尝试使用被分配了NGINX Plus角色的用户的凭据登录。



Casdoor



Auto sign in

[Forgot password?](#)

Sign In

No account? [sign up now](#)

故障排除

请参阅GitHub上nginx-openid-connect仓库中的[故障排除](#)部分。

NginxCommunityVersion

先决条件

本指南假设您具有以下条件：

- 正在运行的Casdoor服务。如果您还没有安装Casdoor服务，请参考[服务器安装](#)或者[使用Docker试用](#)。
- 在编译时启用了`ngx_http_auth_request_module`模块的Nginx开源版。如果你不知道如何启用`ngx_http_auth_request_module`模块，请参考[Nginx模块文档](#)。
- 您希望启用身份验证的网站已成功部署在Nginx上，具有**配置的域名**（而不是使用IP地址），并且可以正常访问。
- OAuth2-Proxy工具（目前，GitHub上有以下两个高星级的热门项目可供选择，您需要选择其中之一）：
 1. [oauth2-proxy/oauth2-proxy](#)（本文中使用的）[GitHub](#) 或 [官方网站](#)
 2. [vouch/vouch-proxy](#) [GitHub](#)

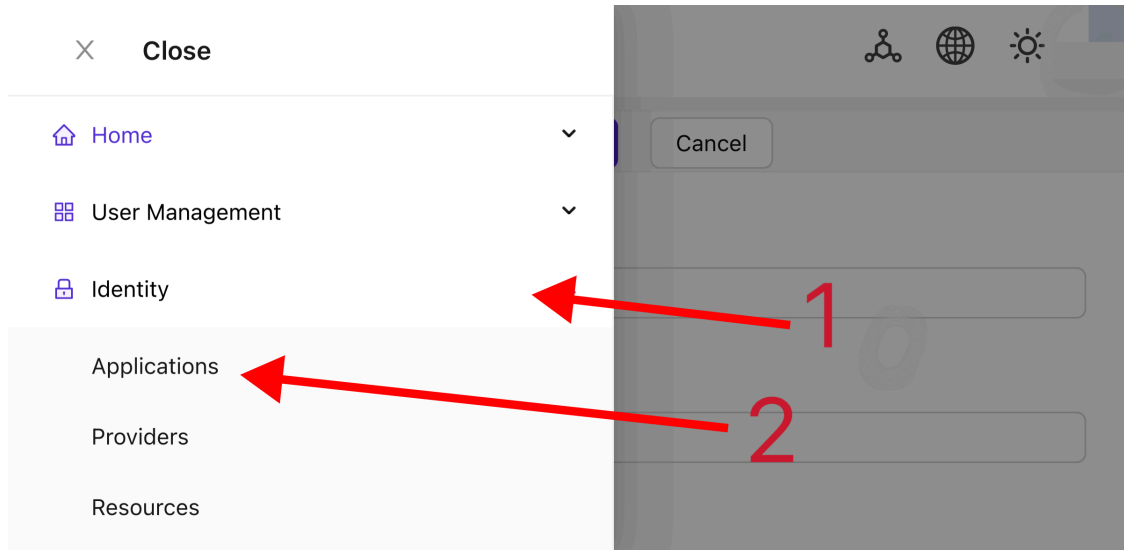
I. 配置CasDoor

注意：本文中的操作基于发布时的Casdoor GUI，但Casdoor GUI可能会根据版本变化。请按照本文中提供的参考配置您部署的Casdoor版本。

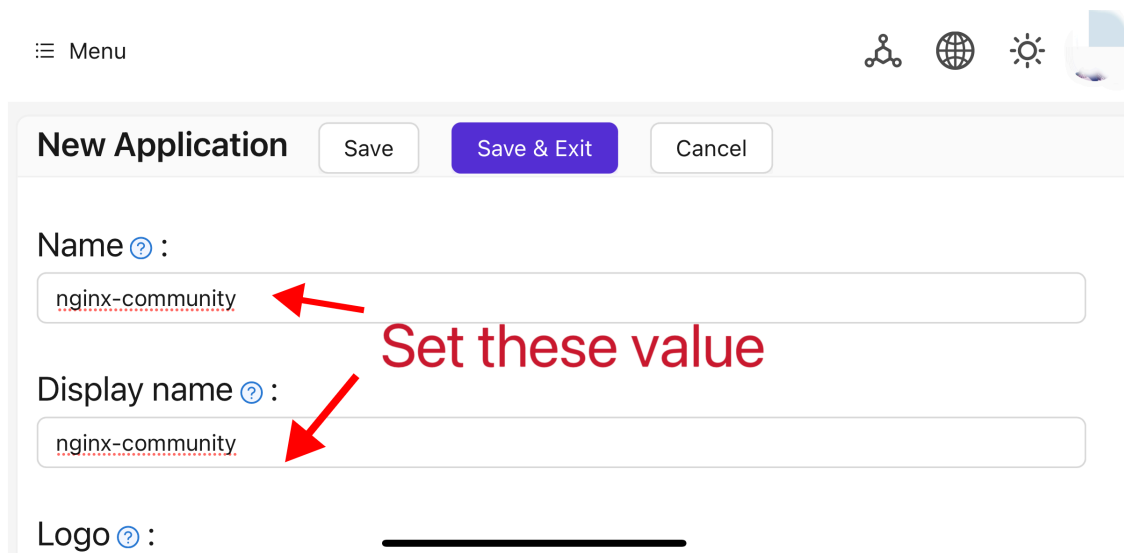
注意：本文中提到的密钥、密码、用户名和其他机密信息都是示例。出于安全原因，您在部署时必须用您自己的相关内容替换它们。

1. 登录到您的Casdoor管理员账户。

2. 在顶部栏中，选择"身份验证" > "应用程序"，然后在"应用程序"页面上点击"添加"。



3. 根据您的项目信息完成应用程序配置。在本文中，我们使用"Nginx-Community"作为示例应用程序名称。



4. 记下"Client ID"和"Client Secret"字段的值。它们将在稍后配置OAuth2-Proxy时使用。然后将"Redirect URL"配置为 `https://project.yourdomain.com/oauth2/callback/`。

Client ID [?](#) :

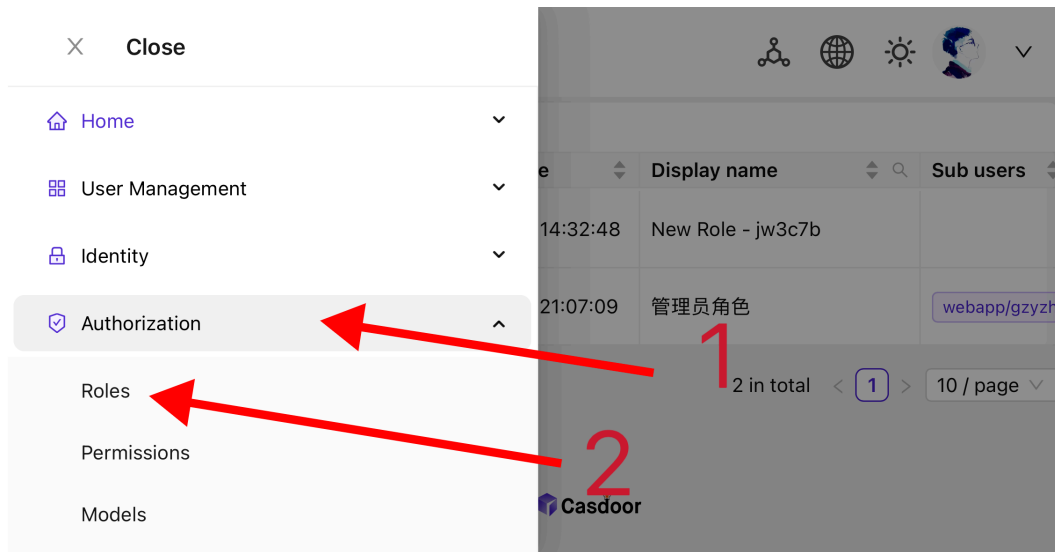
Client secret [?](#) :

Cert [?](#) :

Redirect URLs [?](#) :
 Then, set this value to "https://project.yourdomain.com/oauth2/callback"

Redirect URL	Action

5. 在顶部栏中, 选择"Casbin权限管理" > "角色", 然后在"角色"页面上点击"添加"。



6. 根据您的项目信息完成角色配置。在本文中, 我们使用"nginx_role"作为示例角色名称。

New Role Save Save & Exit Cancel

Organization ⓘ :
webapp

Name ⓘ :
nginx_community

Display name ⓘ :
nginx_community

Set these value
Then, press "save&exit"

7. (可选) 在顶部栏中, 选择"用户管理" > "用户", 然后根据您的需要添加新用户。如果您需要的用户已经存在, 您可以跳过此步骤。在本文中, 我们创建了一个名为"user"的示例用户。
8. 返回到步骤5中提到的"角色"页面, 编辑 `nginx_role` 角色, 并将您需要的用户添加到"包含的用户"选项中。在本文中, 我们在这里添加了之前创建的 `builtin/user`。

II. 配置Oauth2-Proxy

注意: 本文使用Oauth2-Proxy项目作为示例。如果你想使用Vouch代替Oauth2-Proxy, 请参考他们在[GitHub](#)上的官方文档。

注意: 本文假设您的网站配置了受信任的SSL证书并且只允许HTTPS访问, 或者您已经设置了从HTTP访问者自动重定向到HTTPS的设置。这有助于最大限度地保护cookies并防止恶意读取登录令牌。如果您的网站需要通过不安全的HTTP协议访问, 请相应地修改相关命令。关于通过HTTP部署的更多帮助, 请参考Oauth2-Proxy在[GitHub](#)上的官方文档。

提示: OAuth2-Proxy提供了各种部署方法（如源代码编译、Docker安装等）。为了解释，本文使用"预构建二进制文件"进行部署。

1. 转到[GitHub Releases](#)页面，下载与您的操作系统和CPU架构相对应的二进制包。截至2024年1月1日，OAuth-Proxy的最新发布版本为v7.5.1。如果你想下载这个版本的二进制包，你可以执行以下命令，适用于AMD64的Linux：

```
wget -O oauth2-proxy-linux.tar.gz https://github.com/oauth2-proxy/oauth2-proxy/releases/download/v7.5.1/oauth2-proxy-v7.5.1.linux-amd64.tar.gz
```

强烈建议您在下载压缩包后检查官方网站在[GitHub Releases](#)页面提供的SHA256SUM值，并与您下载的包的SHA256SUM值逐字符进行比较。

2. 解压下载的包：

```
tar -zxvf oauth2-proxy-*.tar.gz
```

3. 进入解压后的目录：

```
cd oauth2-proxy-v7.5.1.linux-amd64
```

4. 将获得的二进制文件移动到/usr/local/bin并配置执行权限。根据您的情况，您可能需要使用sudo提升权限。

```
cp ./oauth2-proxy /usr/local/bin
cd /usr/local/bin
chmod +x ./oauth2-proxy
```

5. 测试二进制安装。如果安装成功，执行以下命令后，您应该看到类似于

oauth2-proxy v7.5.1 (built with go1.21.1) 的输出。

```
cd ~
oauth2-proxy --version
```

6. 使用命令行参数运行oauth2-proxy。标记为[required]的参数必须根据您的具体情况进行配置，而标记为[optional]的参数可以优化性能，但也可以省略。为确保oauth2-proxy可以在后台运行，您可以使用Screen或Supervisor等进程监控工具或终端工具。

```
oauth2-proxy \  
--provider=oidc \ #[required] 不要更改  
--client-id=abc123456def \ #[required] 上面步骤I.4中获得的"Client  
ID"  
--client-secret=abc123456def \ #[required] 上面步骤I.4中获得  
的"Client Secret"  
--oidc-issuer-url=https://auth.yourdomain.com \ #[required] 您的  
Casdoor URL (域名或公共IP)  
--redirect-url=https://project.yourdomain.com/oauth2/callback  
\ #[required] https://domain-of-the-project-to-protect/oauth2/  
callback  
--scope=email+profile+groups+openid \ #[required] 从Casdoor获  
得: 用户电子邮件, 用户配置文件, 组和登录认证  
--cookie-domain=project.yourdomain.com \ #[required] 您想要保护的  
项目的域名  
--whitelist-domain=project.yourdomain.com \ #[required] 您想要保  
护的项目的域名  
--cookie-secret=abc123456def \ #[required] 请生成一个随机的数字和字  
母字符串并在这里填写  
--email-domain=* \ #[required] 可接受的用户电子邮件域的列表 (*表示接  
受所有域)。如果用户的电子邮件后缀不在此列表中, 即使登录成功也会返回403错  
误。  
--insecure-oidc-allow-unverified-email=true \ #[required] 是否接  
受未验证电子邮件地址的用户
```

III. 配置Nginx

注意：请再次确认您的Nginx在编译和安装源代码时已启用 `ngx_http_auth_request_module` 模块（编译命令包括 `--with-http_auth_request_module`）。如果你不知道如何启用 `ngx_http_auth_request_module` 模块，请参考[Nginx模块文档](#)。

提示：使用宝塔面板工具安装的Nginx默认不启用此模块。

1. 打开您已经部署并希望保护的网站的配置文件，并进行以下修改：

注意：您需要根据您的具体情况调整此配置文件。由于Nginx版本和其他因素，此配置文件可能无法在所有Nginx实例上顺利工作。请根据您的自己的Nginx信息调整相关内容。

```
server {
    listen 443 ssl http2;

    include /path/to/ssl.conf;

    # 添加以下内容
    location ^~ /oauth2/ {
        proxy_pass http://127.0.0.1:65534; # 将此更改为步骤
        II.6中配置的"--http-address"

        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Scheme $scheme;

        proxy_set_header X-Auth-Request-Redirect $request_uri;
        # 或者，如果你正在处理多个域名：
        # proxy_set_header X-Auth-Request-Redirect
        $scheme://$host$request_uri;
```

2. 保存文件并重新加载您的Nginx。

测试

- 接下来，您可以测试您的实现。
- 在正常情况下，您的用户在登录您的服务时将经历以下过程：
- 在浏览器中打开URL `project.yourdomain.com` → 只看到一个需要登录的页面，包括一个名为"Sign in with OpenID Connect"的按钮 → 点击按钮并被重定向到您的Casdoor地址，那里将要求他们登录 → 用户输入他们的用户名和密码，Casdoor验证他们的凭据 → 自动重定向回您的URL `project.yourdomain.com` → 成功访问您的服务 → 当您设置的 `--cookie-expire` 时间到期时，用户将被要求再次登录。

故障排除

- 如果您的项目没有按预期运行，请检查您的Nginx配置和Oauth2-Proxy配置参数是否正确。
- 您也可以参考Oauth2-Proxy在[GitHub](#)上的官方文档。
- 如果您发现本文档中有任何错误，请随时在[GitHub](#)上请求编辑。

Envoy

先决条件

正在运行的Casdoor服务器。请参考Casdoor文档中的[服务器安装](#)和[使用Docker试用](#)。

配置Casdoor

1. 添加Envoy应用程序。在**重定向URL**字段中，输入Envoy实例的URL，包括端口号，并以/oauth2/callback结束（例如，http://%REQ(:authority)%/oauth2/callback）。记下客户端ID和客户端密钥的值。
2. 添加envoy-casdoor-role角色。
3. 添加user1用户。在注册应用程序中选择**Envoy**。在**管理账户**字段中，从应用程序下拉菜单中选择**Envoy**，并填写用户名和密码。返回到**角色**页面，点击envoy-casdoor-role行上的"编辑"。在打开的页面中，**子用户**字段中，选择你刚刚创建的用户名（在这个例子中，它是built-in/user1）。

配置Envoy

1. 修改envoy.yaml文件中的 `token_endpoint`、`authorization_endpoint` 和 `client_id`。
2. 将token-secret.yaml文件中的 `inline_string` 修改为Casdoor的Envoy客户端密钥。
3. 将hmac-secret.yaml文件中的 `inline_bytes` 修改为一个独特的、长的、安全的短语。
4. 将envoy.yaml、token-secret.yaml和hmac-secret.yaml文件添加到你的Envoy路

径中。

如何运行

1. 使用`envoy.yaml`文件启动Envoy。
2. 访问Envoy正在监听的网站。你应该立即被重定向到Casdoor进行用户身份验证。

C#

Unity

使用Casdoor-dotnet-sdk进行Unity开发。

Unity

步骤1：部署Casdoor

首先，应部署Casdoor。

您可以参考Casdoor官方文档中的[服务器安装](#)。请在**生产模式**下部署您的Casdoor实例。

成功部署后，请确保：

- 打开您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` 来测试登录功能。

或者，您可以使用[Casdoor官方演示站](#)进行快速开始。

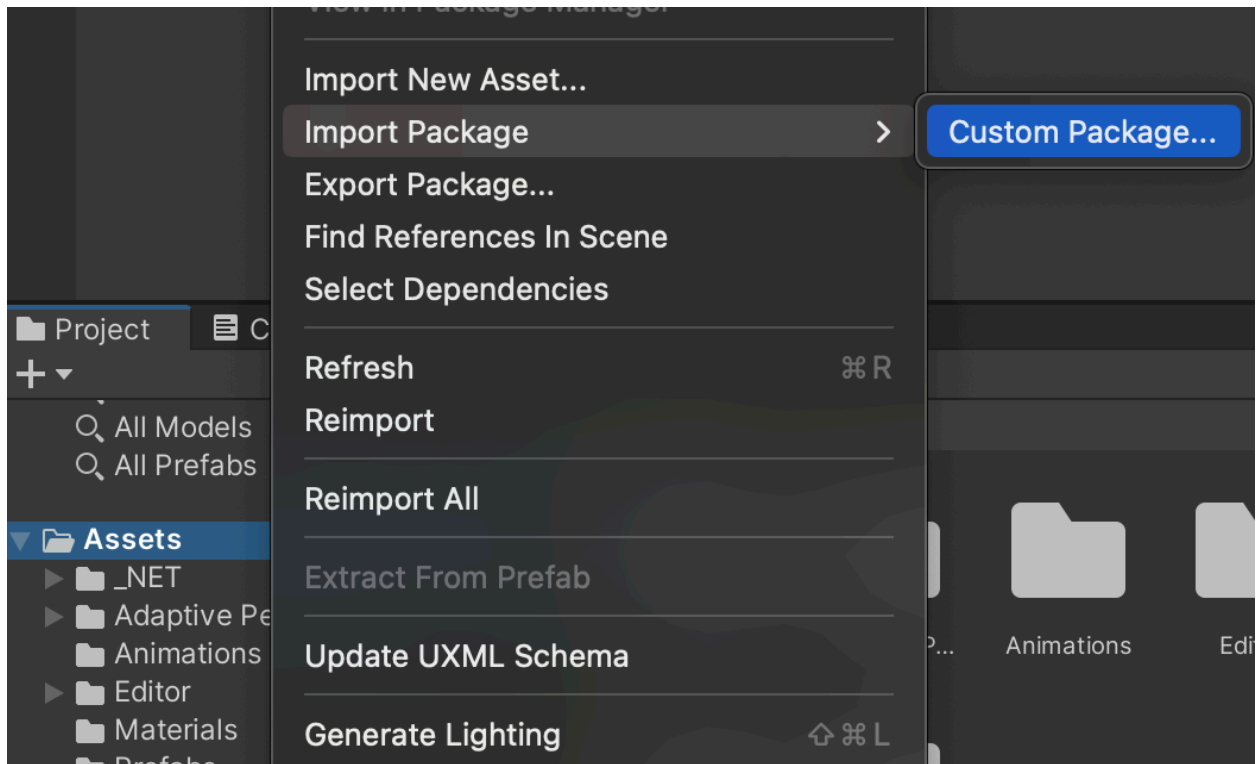
步骤2：导入Casdoor.Client

在[Casdoor-dotnet-sdk](#)中为 `.NET` 导入 `Casdoor.Client`。

一种可选的方法如下：

- `git@github.com:casdoor/casdoor-dotnet-sdk.git`
- 在Sample文件夹中运行ConsoleApp。
- 获取 `/casdoor-dotnet-sdk/src/Casdoor.Client/bin/Debug/net462` 文件夹。

现在，您可以通过下图所示的方法将 `net462` 文件夹导入到您的Unity项目中。当然，您也可以选择其他版本的文件夹。



步骤3：使用

通过查看[casdoor-unity-example](#)来了解如何使用 `Casdoor.Client` SDK进行Unity 3D 移动开发。

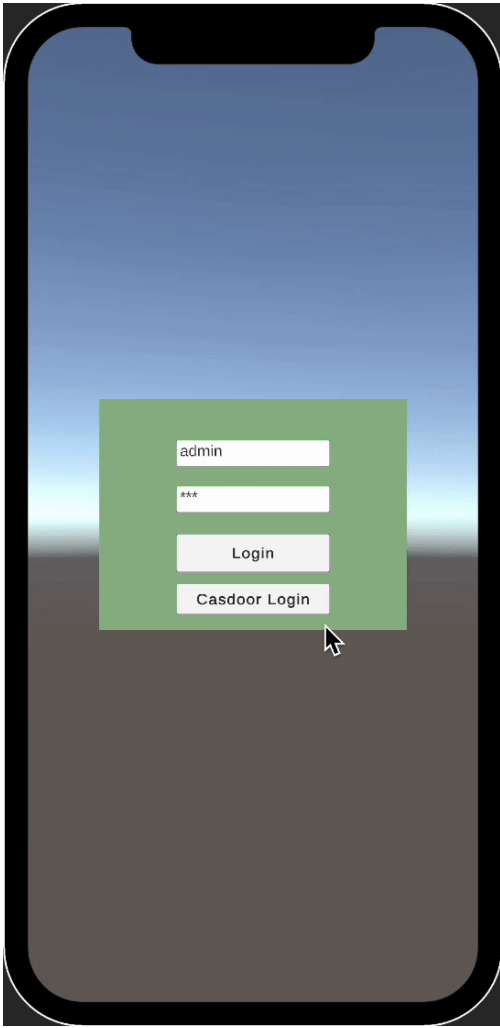
运行casdoor-unity-example后，您将看到以下界面：

- 使用用户名和密码登录：

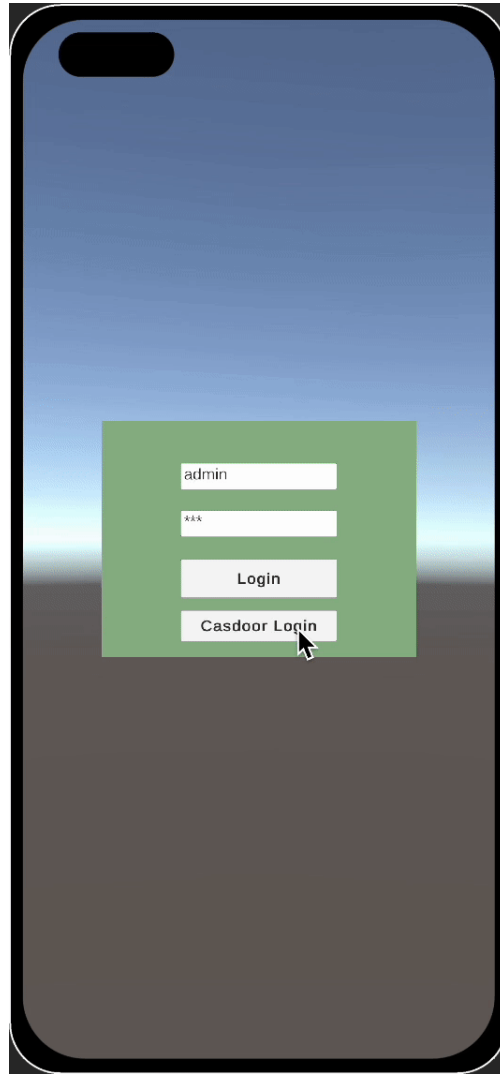


- 使用Casdoor网页登录:

iOS



Android



Go

Kubernetes

在Kubernetes中使用Casdoor进行身份验证

OpenShift

在OpenShift中使用Casdoor进行身份验证

BookStack

在 BookStack 中使用 Casdoor 进行身份验证

Bytebase

使用OAuth2连接各种应用程序，如Bytebase

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

将Casdoor配置为身份提供商以支持MinIO

Portainer

在Portainer中使用Casdoor进行身份验证

Kubernetes

根据[Kubernetes文档](#)，Kubernetes的API服务器可以使用OpenID Connect (OIDC)进行身份验证。本文将指导您如何使用Casdoor在Kubernetes中配置身份验证。

环境要求

在开始之前，请确保您有以下环境：

- 一个Kubernetes集群。
- 像这样的Casdoor应用程序[演示网站](#)。
- kubectl命令工具（可选）。

📌 备注

Kubernetes的`oidc-issuer-url`只接受使用`https://`前缀的URL。因此，您的Casdoor应用程序应部署在HTTPS网站上。


步骤1：创建一个Casdoor应用和用户账户进行身份验证

转到您的Casdoor应用程序并添加一个名为Kubernetes的新应用。请记住 `Name`、`Organization`、`client ID`、`client Secret`，并为此应用添加一些授权类型。

Name [?](#): Kubernetes

Display name [?](#): New Application - Kubernetes

Logo [?](#):
URL [?](#): https://cdn.casbin.org/img/casdoor-logo_1185x256.png

Preview: 

Home [?](#): [🔗](#)

Description [?](#):

Organization [?](#): casbin

Client ID [?](#): Kubernetes

Client secret [?](#): 72c65c3912aec24a9f3ec41b65a7577114ed2bae

Cert [?](#): cert-built-in

Grant types [?](#):
 Authorization Code Password ID Token Refresh Token Client Credentials Token

接下来，向您刚刚创建的应用程序添加一个新用户。请注意，这里使用的 `Organization` 和 `Signup application` 应与之前注册的应用对应。

Organization ? : casbin

ID ? : 202e02e9-9128-496a-a209-fdb336448f56

Name ? : user_pnvm5i

Display name ? : New User - pnvm5i

Avatar ? :

Preview:



Upload a photo...

User type ? : normal-user

Password ? : Modify password...

Email ? : pnvm5i@example.com

Phone ? : +1 78005961394

Country/Region ? : Please select country/region

Location ? :

Affiliation ? : Example Inc.

Title ? :

Homepage ? :

Bio ? :

Tag ? : staff

Signup application ? : Kubernetes

步骤2：使用OIDC身份验证配置Kubernetes API服务器

要启用OIDC插件，您需要在API服务器上配置以下标志：

- `--oidc-issuer-url`：允许API服务器发现公共签名密钥的提供者的URL。
- `--oidc-client-id`：所有令牌必须为其发出的客户端id。

本文使用minikube进行演示。您可以使用以下命令在启动时为minikube的API服务器配置OIDC插件：

```
minikube start --extra-config=apiserver.oidc-issuer-url=https://demo.casdoor.com --extra-config=apiserver.oidc-client-id=294b09fbc17f95daf2fe
```

步骤3：测试OIDC身份验证

获取身份验证信息

由于kubectl缺乏前端，可以通过向Casdoor服务器发送POST请求进行身份验证。这是在Python中向Casdoor服务器发送POST请求并检索 `id_token` 和 `refresh_token` 的代码：

```
import requests
import json

url = "https://demo.casdoor.com/api/login/oauth/access_token"
payload = json.dumps({
```

执行此代码后，您应收到类似于以下的响应：

```
{
  "access_token": "xxx",
  "id_token": "yyy",
  "refresh_token": "zzz",
  "token_type": "Bearer",
  "expires_in": 72000,
  "scope": ""
}
```

现在，您可以使用刚刚获取的 `id_token` 与 Kubernetes API 服务器进行身份验证。

基于HTTP请求的身份验证

将令牌添加到请求头。

```
curl https://www.xxx.com -k -H "Authorization: Bearer $(id_token)"
```

- `https://www.xxx.com` 是 Kubernetes API 服务器的部署地址。

基于Kubectl客户端的身份验证

配置文件方法

将以下配置写入 `~/.kube/config` 文件。您应该用您之前获得的值替换配置文件中的每个配置项。

```
users:
- name: minikube
  user:
    auth-provider:
```

现在，您可以直接使用kubectI访问您的API服务器。 尝试运行一个测试命令。

```
kubectI cluster-info
```

命令行参数方法

或者，您可以通过直接将 `id_token` 添加到kubectI的命令行参数中进行身份验证。

```
kubectI --token=$(id_token) cluster-info
```

OpenShift

OpenShift支持OIDC，所以我们可以将Casdoor与OpenShift集成。以下步骤演示了如何使用[Casdoor在线演示](#)将Casdoor与OpenShift Local集成。

步骤1：创建一个Casdoor应用


在Casdoor中添加一个新的应用，注意以下几点：

- 记住 `Client ID` 和 `Client secret` 以备下一步使用。
- 重定向URL的格式是 `https://oauth-openshift.apps.<cluster_name>.<cluster_domain>/*`。根据你的情况填写。

Name ⓘ: openshift

Display name ⓘ: openshift

Logo ⓘ: URL ⓘ: https://cdn.casbin.org/img/casdoor-logo_1185x256.png

Preview: 

Home ⓘ: [↗](#)

Description ⓘ:

Organization ⓘ: built-in

Client ID ⓘ: 2452f2b5abb6ff131199

Client secret ⓘ: f7b40c97ea35bcd1c17a367c8eec373cc027ee96

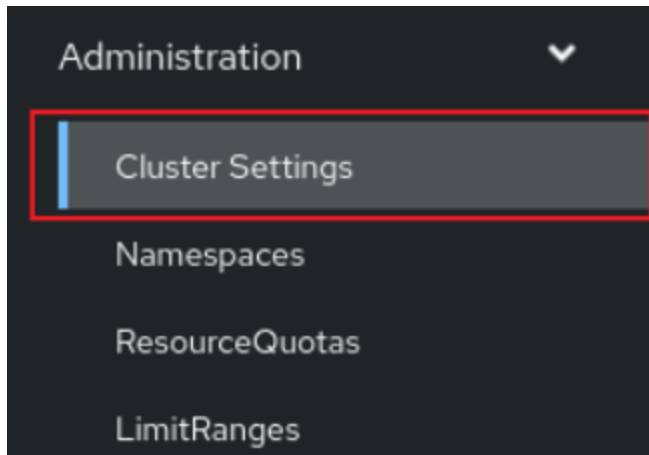
Cert ⓘ: cert-built-in

Redirect URLs ⓘ:

Redirect URL
https://oauth-openshift.apps-crc.testing/*

步骤2: OpenShift OAuth配置

现在以Kubeadmin的身份登录OpenShift控制台。登录后，浏览侧边菜单并找到集群设置。



在全局配置下，你会看到OAuth。

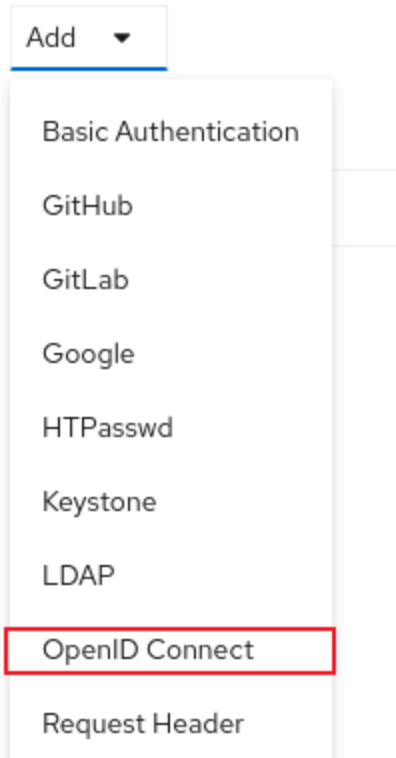
OAuth

OAuth holds cluster-wide information about OAuth. The canonical name is 'cluster'. It is used to configure the integrated OAuth server. This configuration is only honored when the top level Authentication config has type set to IntegratedOAuth. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

你会看到身份提供商部分。在添加部分，从选项中选择OpenID Connect。

Identity providers

Identity providers determine ho



配置OIDC，注意以下几点：

- 填写上一步记住的 `Client ID` 和 `Client Secret`。
- 发行人URL必须使用https，形式为 `https://<casdoor-host>`，再次根据你的情况填写。

Add Identity Provider: OpenID Connect

Integrate with an OpenID Connect identity provider using an Authorization Code Flow.

Name *

Unique name of the new identity provider. This cannot be changed later.

Client ID *

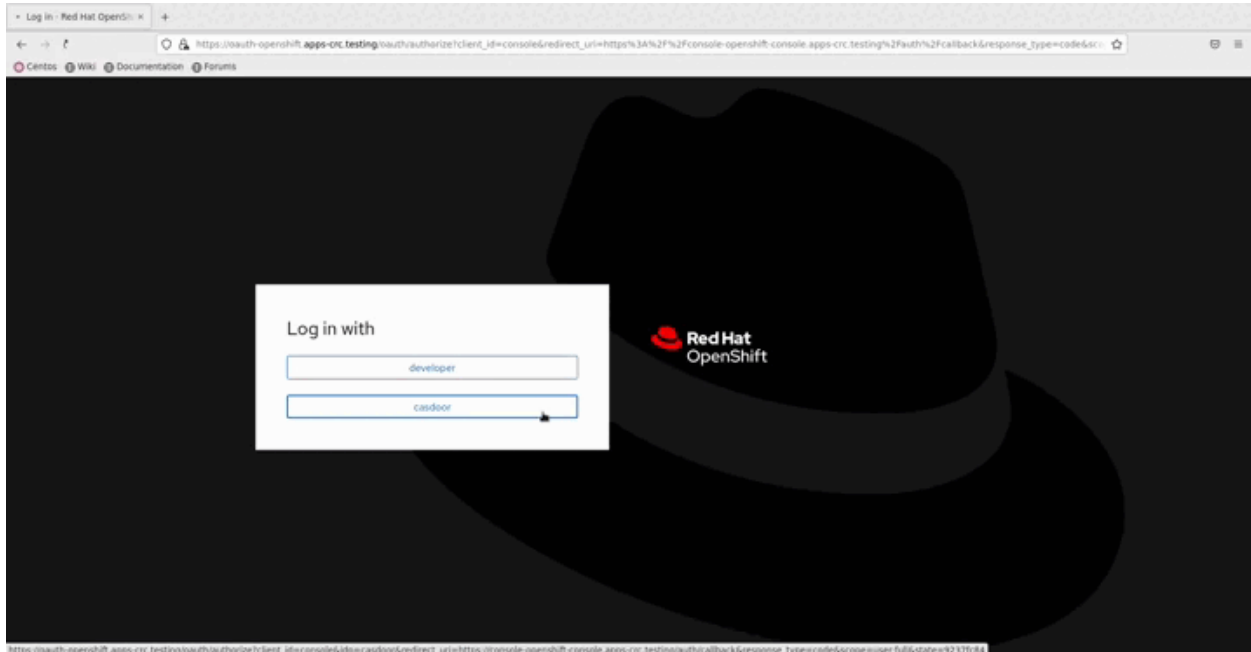
Client secret *

Issuer URL *

The URL that the OpenID provider asserts as its issuer identifier. It must use the https scheme with no URL query parameters or fragment.

步骤3：测试OIDC身份验证

在浏览器中访问OpenShift控制台。你会看到Casdoor（你在上一步中配置的名称）。点击Casdoor登录选项。你将被重定向到Casdoor登录页面。



BookStack

在 BookStack 中使用 Casdoor 进行身份验证

BookStack是一个开源的书籍和文档分享网站，同时也是一个使用Go语言开发的开源应用程序，帮助您更好地管理文档阅读。

BookStack-casdoor 已经与 Casdoor 集成，现在您可以通过简单的配置快速开始使用。

步骤1：创建一个Casdoor应用


前往您的Casdoor并添加一个名为BookStack的新应用程序。这是在Casdoor中创建BookStack应用程序的一个示例。

Edit Application

Name [?](#): bookstack

Display name [?](#): bookstack

Logo [?](#):
URL [?](#): 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`。您将在下一步中需要它们。

步骤2：配置Casdoor登录

接下来，导航到BookStack并找到文件 `oauth.conf.example`。

将 `oauth.conf.example` 重命名为 `oauth.conf` 和 **修改配置** 默认情况下，内容如下：

```
[oauth]
casdoorOrganization = "<组织>"
casdoorApplication = "bookstack"
casdoorEndpoint = http://localhost:8000
clientId = <客户端 ID>
clientSecret = <客户端密钥>
redirectUrl = http://localhost:8181/login/callback
```

步骤3: 在Casdoor中填写 `redirectUrl`

在最后一步中, 返回到您添加BookStack应用程序的页面, 并填写 `Redirect URLs`。确保 `Redirect URL` 与 `oauth.conf` 文件中的 `redirectUrl` 相同。

Redirect URLs [?](#):

Redirect URL
http://localhost:8181/login/callback
http://localhost:8181/login/callback

现在你已经完成了Casdoor的配置!

一旦你成功部署了BookStack, 现在你可以回到你的BookStack并体验使用Casdoor进行登录验证。

Bytebase

Casdoor可以使用OAuth2连接各种应用程序。在这个例子中，我们将使用Bytebase来演示如何使用OAuth2连接到您的应用程序。

以下是配置名称：

`CASD00R_HOSTNAME`：部署Casdoor服务器的域名或IP地址。

`Bytebase_HOSTNAME`：部署Bytebase的域名或IP地址。

步骤1：部署Casdoor和Bytebase

首先，部署Casdoor和Bytebase。

成功部署后，请确保：

1. Casdoor可以正常登录和使用。
2. 在prod模式下部署Casdoor时，您可以将`CASD00R_HOSTNAME`设置为`http://localhost:8000`。查看[生产模式](#)。

步骤2：配置Casdoor应用程序

1. 创建一个新的或使用现有的Casdoor应用程序。
2. 找到重定向URL：`<CASD00R_HOSTNAME>/oauth/callback`。
3. 将重定向URL添加到Casdoor应用程序：

Client ID: e828d692244292b979e

Client secret: bab9f6c2fad67471e11bdf81e074ea192e4f46dd

Cert: cert-built-in

Redirect URLs:

Redirect URL	Action
<CASDOOR_HOSTNAME>/oauth/callback	Add Delete Refresh

在应用程序设置页面上，您将找到两个值：Client ID 和 Client secret。我们将在下一步中使用这些值。

打开您喜欢的浏览器并访问：http://CASDOOR_HOSTNAME/.well-known/openid-configuration。您将看到Casdoor的OIDC配置。

步骤3：配置Bytebase

1. 找到SSO并选择OAuth 2.0:

SQL Review

Risk Center

Custom Approval

Data Anonymization

Data Access Control

Audit Log

Integration

GitOps

SSO

IM

Type

OAuth 2.0 OIDC

Use template

GitHub GitLab Google Custom

Basic information

Name * Custom

Identity Provider ID: idp-custom-f4mw It cannot be changed later. [Edit](#)

Domain

2. 配置此应用程序:

Account > SSO > casdoor

Basic information

Name *
casdoor

Identity Provider ID: ldap-casdoor-mtk

Domain
http://101.43.192.216:8000

Identity provider information

The information is provided by your identity provider.

Client ID *
e828d69226292979e

Client secret *
sensitive - write only

Auth URL *
The link to OAuth login page
http://101.43.192.216:8000/login/oauth/authorize

Scopes *
A space-separated list of scopes to be carried when accessing the Auth URL.
openid profile email

Token URL *
The API address for obtaining access token
http://101.43.192.216:8000/api/login/oauth/access_token

User information URL *
The API address for obtaining user information by access token
http://101.43.192.216:8000/api/get-account

User information mapping

Maps the field names from user info API to the Bytebase user. [Learn more!](#)

name → Bytebase user identifier *

name → Bytebase user display name

email → Bytebase user email

Test Connection | Archive this SSO

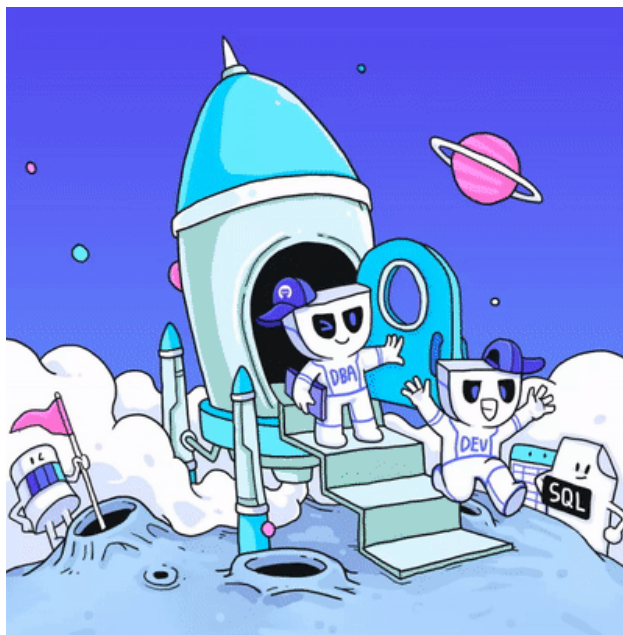
Enterprise Plan v1.15.0

Discard changes | Update

3. 在Casdoor应用程序页面上找到Client ID和Client Secret。

- Token server URL : `http://CASDOOR_HOSTNAME/api/login/oauth/access_token`
- Authorization server URL : `http://CASDOOR_HOSTNAME/login/oauth/authorize`
- User Info server URL : `http://CASDOOR_HOSTNAME/api/get-account`
- Scopes: `address phone openid profile offline_access email`

退出Bytebase并测试SSO。



Bytebase

Email *
jimi@example.com

Password * [Forgot your password?](#)

Sign In

New to Bytebase? [Sign up](#)

or

Sign in with casdoor

English 简体中文
© 2023 Bytebase. All rights reserved.

ELK

Casdoor/elk-auth-casdoor概览

ELK (Elasticsearch, Logstash和Kibana) 最大的缺点之一是, 这些产品最初没有认证机制。因此, 任何知道Kibana或Elasticsearch的URL的人都可以访问Kibana仪表盘。后来, ELK集成了一个名为“Xpack”的嵌入式认证系统。然而, 它的高级功能(如OAuth, OIDC, LDAP, SAML)并不是免费的。只有使用一组账户和密码的普通认证是免费的, 这非常不方便。这种方法不允许我们为公司中的每个人提供唯一的账户。

为了解决这个问题, 我们基于Casdoor开发了一种麋鹿认证解决方案。此解决方案是免费的, 开源的, 正在进行持续的维护, 并支持一系列高级功能。Casdoor是一个基于OAuth 2.0/OIDC的集中认证/单点登录平台。Casdoor/elk-auth-casdoor充当反向代理, 旨在拦截所有朝向ELK/Kibana堆栈的HTTP数据流。它引导尚未登录的用户进行登录。只要用户已登录, 这个反向代理就会透明地运行。

如果用户未被正确验证, 请求将被暂时缓存, 并且用户将被重定向到Casdoor登录页面。一旦用户通过Casdoor登录, 缓存的请求将会被恢复并发送到Kibana。因此, 如果一个POST请求(或者除GET之外的任何其他类型的请求)被拦截, 用户就不需要重新填写表单并重新发送请求。逆向代理将为你记住它。

casdoor/elk-auth-casdoor仓库位于<https://github.com/casdoor/elk-auth-casdoor>。

如何使用?

0. 确保你已经安装了Go编程语言环境。
1. 前往 [casdoor/elk-auth-casdoor](https://github.com/casdoor/elk-auth-casdoor) 并获取代码。

2. 将您的代理注册为Casdoor的应用程序。
3. 修改配置。

配置文件位于 "conf/app.conf"。这是一个例子，你应根据自己的具体需求进行定制。

```
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>（在上述示例中）并按照重定向指导进行登录。然后，你应该会看到Kibana被Casdoor保护并进行了身份验证。
5. 如果一切运行正常，不要忘记通过配置防火墙（或其他方法）来阻止外部访问原始的Kibana端口。这确保了外部人员只能通过这个反向代理来访问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：

Client Application | Save | Save & Exit

Name: application_9p7eai

Display name: New Application - 9p7eai

Logo: URL: 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 URL	Action
http://localhost:3000/user/oauth2/Casdoor/callback	  

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：

Issues Pull Requests Milestones Explore

Dashboard User Accounts Organizations Repositories Webhooks **Authentication Sources** User Emails Configuration System Notices Monitoring

Authentication Source Management (Total: 0) [Add Authentication Source](#)

ID	Name	Type	Enabled	Updated	Created	Edit
----	------	------	---------	---------	---------	------

按“添加认证源”按钮并填写类似的表单。

Issues Pull Requests Milestones Explore

Dashboard User Accounts Organizations Repositories Webhooks **Authentication Sources** User Emails Configuration System Notices Monitoring

Add Authentication Source

Authentication Type * OAuth2

Authentication Name * Casdoor

OAuth2 Provider * OpenID Connect

Client ID (Key) * 7ceb9b7fda4a9061ec1c

Client Secret * 3416238e1edf915eac08b8fe345b2b95cdba7e04

Icon URL

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

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:

[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 

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：

[Register New Account](#) [Link to Existing Account](#)

Complete New Account

Username *

Email Address *

[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进行身份验证。因此，用户使用Casdoor登录Grafana非常容易。只需要几个步骤和简单的配置就可以实现。

这是一个关于如何在Grafana中使用Casdoor进行身份验证的教程。在您继续之前，请确保您已安装并运行了Grafana。

步骤1：在Casdoor中为Grafana创建一个应用


这是在Casdoor中创建应用的一个示例：

Edit Application

Name

Display name

Logo

Preview: 

Home

Description

Organization

Client ID

Client secret

Cert

Redirect URLs
:

Redirect URLs	Action
<input type="text" value="http://localhost:3000/login/generic_oauth"/>	<input type="button" value="Add"/> <input type="button" value="Up"/> <input type="button" value="Down"/> <input type="button" value="Delete"/>

请复制客户端密钥和客户端ID以进行下一步操作。

请添加Grafana的回调URL。默认情况下，Grafana的OAuth回调是 `/login/generic_oauth`。所以请正确地连接这个URL。

步骤2：修改Grafana的配置

默认情况下，OAuth的配置文件位于Grafana的工作目录中的 `conf/defaults.ini`。

请找到部分 `[auth.generic_oauth]` 并修改以下字段：

```
[auth.generic_oauth]
name = Casdoor
icon = signin
enabled = true
allow_sign_up = true
```

关于HTTPS

如果你不希望为Casdoor启用HTTPS，或者你在没有启用HTTPS的情况下部署Grafana，请同时设置 `tls_skip_verify_insecure = true`。

关于使用Casdoor登录后的redirectURI

如果在Grafana中使用Casdoor登录后重定向URI不正确，您可能需要配置 `root_url`。

```
[server]
http_port = 3000
# 用于从浏览器访问Grafana的公开面向的域名
domain = <你的IP地址>
# 完整的公开面向的URL
root_url = %(protocol)s://%(domain)s:%(http_port)s/
```

相关链接：

1. [Grafana 文档](#)
2. [Grafana 默认的ini](#)

关于角色映射

您可能希望配置 `role_attribute_path`，通过 `role_attribute_path` 将您的用户角色映射到Grafana。

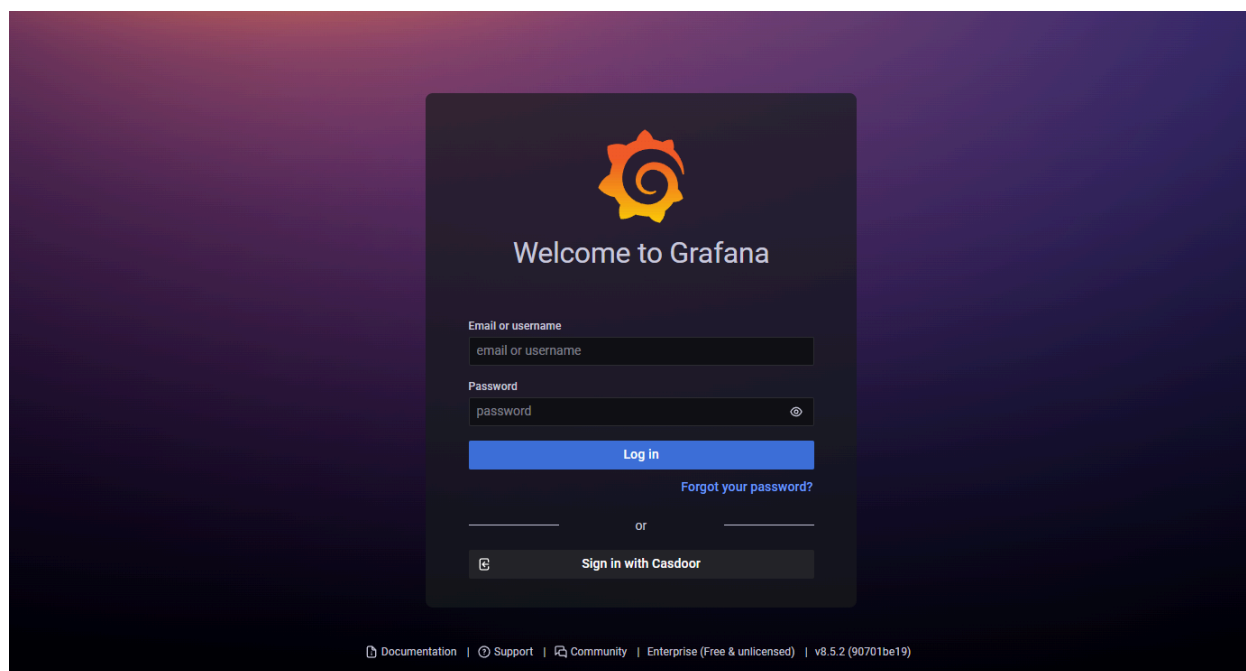
```
[auth.generic_oauth]
role_attribute_path = contains(roles[*].name, 'admin') && 'Admin'
|| contains(roles[*].name, 'editor') && 'Editor' || 'Viewer'
role_attribute_strict = true
```

这里的 `role_attribute_path` 后的 JMESPath 表达式非常重要。请参考 Grafana 文档。

步骤3：查看是否有效

关闭 Grafana 并重新启动它。

前往登录页面。你应该会看到类似这样的东西：



MinIO

MinIO支持使用与OpenID Connect (OIDC)兼容的提供商进行外部身份管理。本文档介绍了如何配置Casdoor作为身份提供商以支持MinIO。

步骤1：部署Casdoor & MinIO

首先，部署Casdoor。

您可以参考Casdoor官方文档中的[服务器安装](#)。

成功部署后，请确保：


- Casdoor 服务器正在 <http://localhost:8000> 上运行。
- 打开你最喜欢的浏览器，访问 <http://localhost:7001>，你将看到Casdoor的登录页面。
- 通过输入 `admin` 和 `123` 来测试登录功能。


接下来，您可以按照以下步骤在您自己的应用中快速实现基于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](#)来获取名为 `mc` 的MinIO客户端。

步骤2：配置Casdoor应用程序

1. 创建一个新的Casdoor应用程序，或使用一个已经存在的。
2. 添加您的重定向URL。

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. 添加您想要的提供者并提供任何必要的设置。

在应用设置页面上，您会找到两个值：`Client ID`和`Client secret`（如上图所示）。我们将在下一步中使用这些值。

打开你最喜欢的浏览器并访问：`http://CASDOOR_HOSTNAME/.well-known/openid-configuration` 来查看Casdoor的OIDC配置。

4. 这个步骤对于MinIO来说是必要的。由于MinIO需要在JWT中使用一个声明属性作为其策略，您也应在Casdoor中进行配置。目前，Casdoor 使用 `tag` 作为配置 MinIO 策略的一种变通方法。

Tag  : `readwrite`

您可以在[这里](#)找到所有支持的策略。

步骤3：配置MinIO

您可以使用以下命令启动 MinIO 服务器：

```
导出MINIO_ROOT_USER=minio
导出 MINIO_ROOT_PASSWORD=minio123
minio server /mnt/export
```

您可以使用 `--console-address` 参数来配置地址和端口。

接下来，使用MinIO客户端 `mc` 添加一个服务别名。

```
mc alias set myminio <您的控制台地址> minio minio123
```

现在，配置MinIO的OpenID Connect。对于Casdoor，命令将是：

```
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"
```

您可以参考[官方文档](#)以获取更详细的参数信息。

一旦成功设置，重新启动MinIO实例。

```
mc 管理服务重启myminio
```

步骤4：尝试演示！

现在，在浏览器中打开您的MinIO控制台，然后点击 `使用SSO登录`。

您将被重定向到Casdoor用户登录页面。成功登录后，您将被重定向到MinIO页面并自动登录。您现在应该能看到您有权限访问的存储桶和对象。

 **注意事项**

如果你在不同的端口部署Casdoor的前端和后端，你被重定向的登录页面将在后端端口上，并且它会显示 `404 not found`。您可以将端口修改为前端端口。然后，您可以成功访问Casdoor登录页面。

Portainer

在Portainer中使用Casdoor进行身份验证

Portainer支持通过OAuth进行身份验证。因此，用户可以轻松地使用Casdoor登录Portainer。只需要几个步骤和简单的配置就可以实现。

这里有一个关于如何在Grafana中使用Casdoor进行身份验证的教程。在你开始之前，请确保你已经安装并运行了Portainer。

以下是配置名称：

`CASD00R_HOST`：部署Casdoor服务器的域名或IP地址。

`PORTAINER_HOST`：部署Portainer的域名或IP地址。


步骤1：在Casdoor中为Portainer创建一个应用

这是在Casdoor中创建应用的一个例子：

Name ⓘ: Portainer_test

Display name ⓘ: Portainer_test

Logo ⓘ: URL ⓘ: https://cdn.casbin.org/img/casdoor-logo_1185x256.png

Preview: 

Home ⓘ: [↗](#)

Description ⓘ:

Organization ⓘ: built-in

Tags ⓘ:

Client ID ⓘ: 2da468d1968c5f85d6b4

Client secret ⓘ: b4db599c84f978425102f161b833625faf9b6b7c

Cert ⓘ: cert-built-in

Redirect URLs ⓘ: [Add](#)

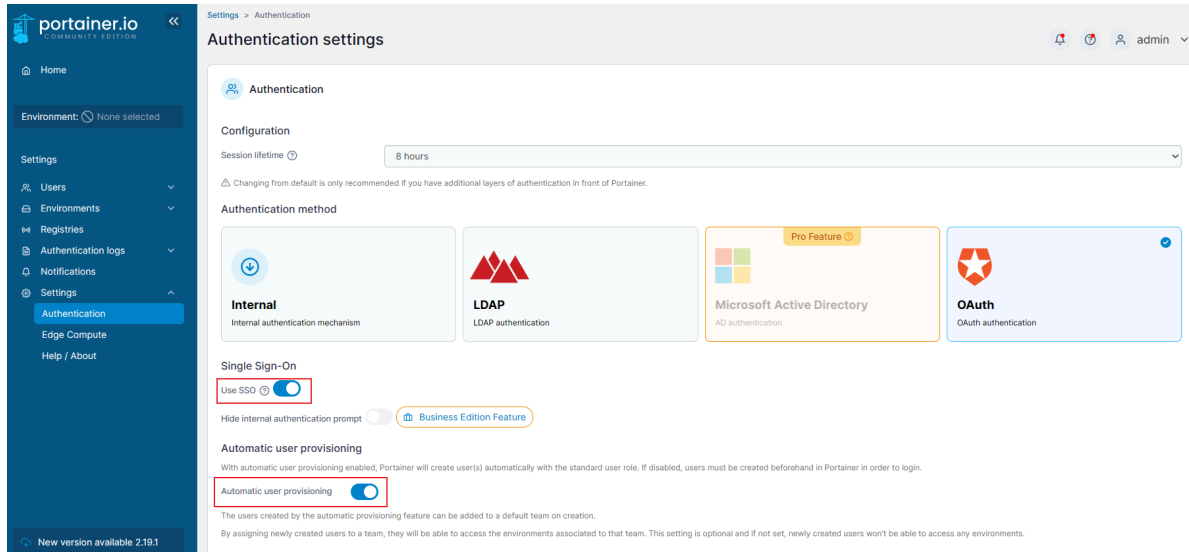
Redirect URL
↗ https://<PORTAINER_HOST>

1. 复制客户端密钥和客户端ID以备下一步使用。
2. 添加一个重定向URL。 这是你的Portainer主机。

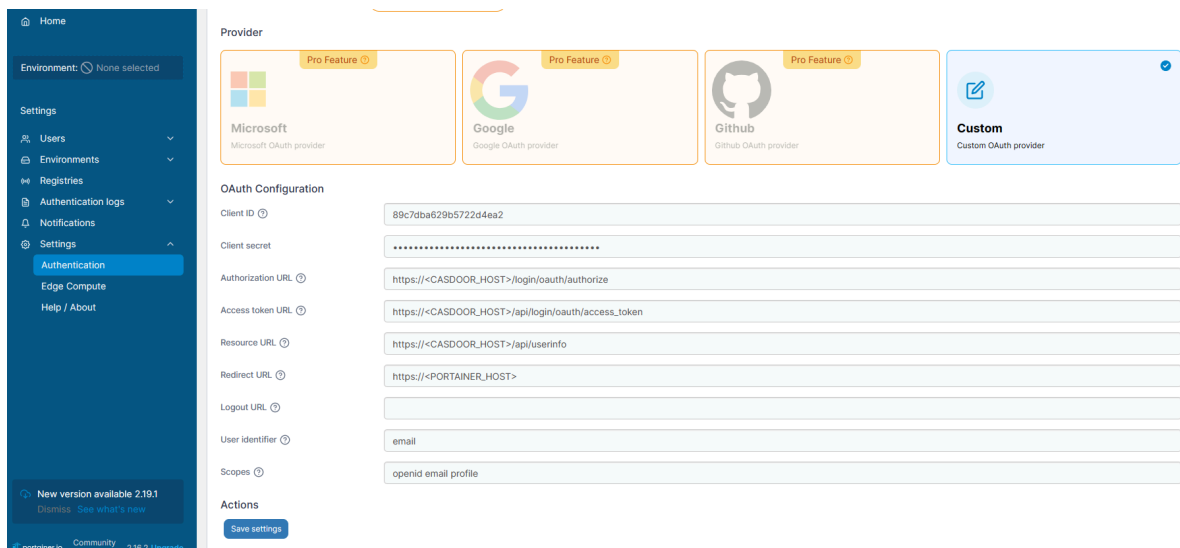
步骤2：配置Portainer

从左侧导航栏展开**设置**，然后从此列表中点击**身份验证**选项。

1. 启用**使用SSO**和**自动用户配置**：

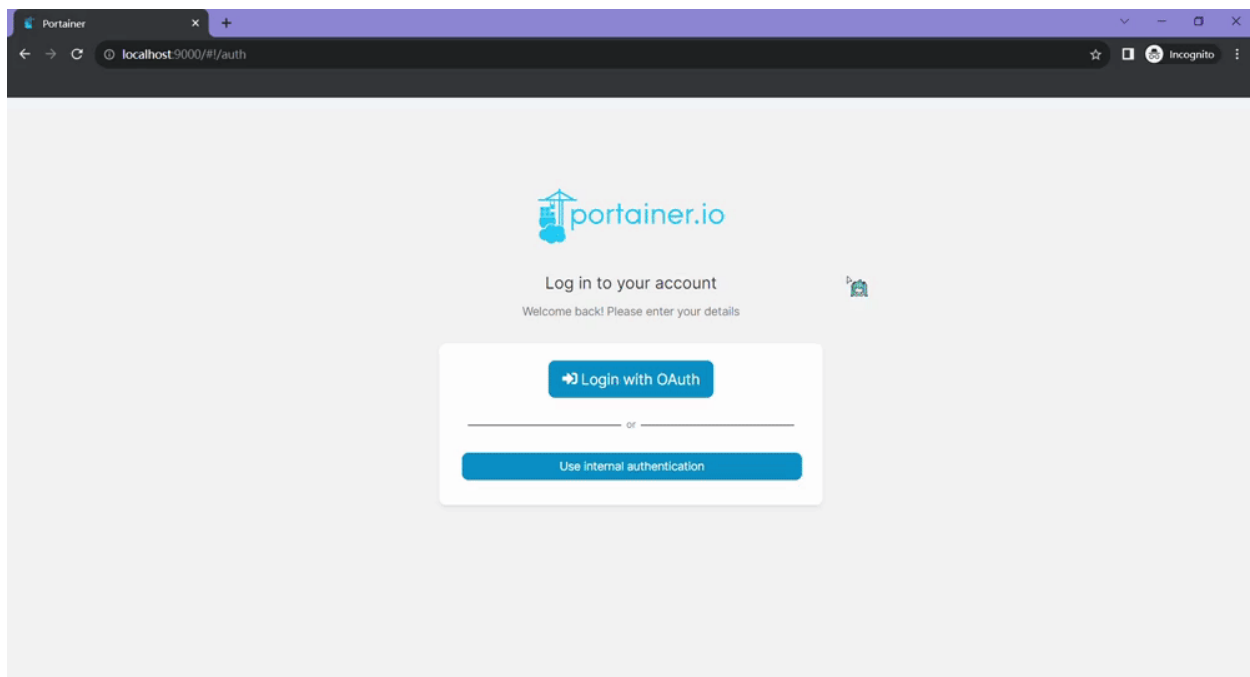


2. 按照以下方式填写必要的信息：



- Authorization URL : `https://<CASDOOR_HOST>/login/oauth/authorize`
- Access token URL : `https://<CASDOOR_HOST>/api/login/oauth/access_token`
- Resource URL : `https://<CASDOOR_HOST>/api/userinfo`
- Redirect URL : `https://<PORTAINER_HOST>`

退出Portainer并进行测试。



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 插件进行 Jenkins 安全

Jenkins OIDC

使用OIDC协议作为IDP连接各种应用，如Jenkins

Jira

2 个项目

使用OIDC协议连接应用程序 - Confluence

学习如何使用OIDC协议作为IDP连接Confluence和其他应用程序。

RuoYi

在 RuoYi-Cloud上使用 Casdoor

Pulsar Manager

在 Pulsar Manager 中使用 Casdoor

在 ShenYu 中使用Cassoor

如何在ShenYu中使用Casdoor

ShardingSphere

在ShardingSphere中使用 Casdoor

Apache IoTDB

使用Casdoor与Apache IoTDB

Apache DolphinScheduler

使用Casdoor进行DolphinScheduler的SSO登录

FireZone

使用OIDC协议作为IDP连接各种应用，如FireZone

Cloud Foundry

了解如何将Casdoor与Cloud Foundry集成，以保护您的应用程序。

Thingsboard

学习如何将Casdoor与Thingsboard集成，以保护您的应用程序

Spring Boot

`casdoor-spring-boot-example`是一个如何在Spring Boot项目中使用`casdoor-spring-boot-starter`的示例。我们将引导您完成以下步骤。

步骤1：部署Casdoor

首先，应部署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 casdoor 实例。

部署成功后，请确保以下内容：

- 打开你最喜欢的浏览器，访问 <http://localhost:8000>。您将看到Casdoor的登录页面。
- 通过输入 `admin` 作为用户名和 `123` 作为密码来测试登录功能。

现在，您可以按照以下步骤在您自己的应用中快速实现基于Casdoor的登录页面。

步骤2：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-
```

```
// https://mvnrepository.com/artifact/org.casbin/casdoor-spring-  
boot-starter  
implementation group: 'org.casbin', name: 'casdoor-spring-boot-  
starter', version: '1.x.y'
```

步骤3：初始化配置

初始化需要按以下顺序的6个字符串类型参数： | 名称 | 必需的 | 描述 | |
----- | --- | ----- | | endpoint | 是 |
Casdoor 服务器 URL，例如 `http://localhost:8000` | | clientId | 是 | 应用程序客户
端ID | | clientSecret | 是 | 应用程序客户端密钥 | | certificate | 是 | 应用程序证书 | |
organizationName | 是 | 应用程序组织 | | applicationName | 否 | 应用程序名称 | 您可
以使用Java属性或YAML文件进行初始化。

[Properties](#) [YML](#)

```
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
```

```
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

将配置值替换为您自己的Casdoor实例，特别是 `clientId`，`clientSecret` 和 `jwtPublicKey`。

步骤4：重定向到登录页面

当你需要验证访问你的应用的用户时，你可以发送目标URL并重定向到Casdoor提供的登录页面。请确保您已经在应用程序配置中添加了回调URL（例如，<http://localhost:8080/login>）。

```
@Resource
private CasdoorAuthService casdoorAuthService;

@RequestMapping("toLogin")
public String toLogin() {
    return "redirect:" +
        casdoorAuthService.getSignInUrl("http://localhost:8080/login");
}
```

步骤5：获取令牌并解析

在通过Casdoor验证后，它将带着代码和状态重定向回您的应用程序。

您可以获取代码并调用 `getOAuthToken` 方法，然后解析JWT令牌。

`Casdoor User` 包含了由Casdoor提供的有关用户的基本信息。您可以使用它来设置应用程序中的会话。

```
@RequestMapping("login")
```

服务

以下是API的示例:

- 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, fullPath, file);`
 - `CasdoorResponse response = casdoorResourceService.deleteResource(file.getName());`

更多资源

您可以探索以下项目/文档，以了解更多关于将Java与Casdoor集成的信息：

- [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微服务系统中，通常在网关处进行身份验证。请参考 [casdoor-springcloud-gateway-example](#) 获取更多信息。

如果您想要在单个服务中使用Casdoor，您可以参考 [casdoor-spring-boot示例](#)。

无论是在网关层还是在单个服务中，都使用了[casdoor-spring-boot-starter](#)。

更多内容

您可以探索以下项目/文档，以了解更多关于将Java与Casdoor集成的信息：

- [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 Cloud Gateway

`casdoor-springcloud-gateway-example` 是一个示例，说明如何在 Spring Cloud Gateway 中使用 `casdoor-spring-boot-starter` 作为 OAuth2 插件。以下是使用它的步骤描述。

步骤1：部署Casdoor

首先，应部署 Casdoor。您可以参考官方 Casdoor 文档中的 [服务器安装](#)。请在 **生产模式** 中部署您的 Casdoor 实例。

成功部署后，您需要确保以下内容：

- 打开你最喜欢的浏览器，访问 <http://localhost:8000>。您将看到 Casdoor 的登录页面。
- 输入 `admin` 和 `123` 来测试登录功能是否正常工作。

之后，您可以按照以下步骤在您自己的应用中快速实现基于 Casdoor 的登录页面。

步骤2：初始化一个Spring Cloud Gateway

您可以直接使用此示例中的代码，或将其与您自己的业务代码结合使用。

您需要一个网关服务和至少一个业务服务。在这个例子中，`casdoor-gateway` 是网关服务，`casdoor-api` 是业务服务。

步骤3：包含依赖项

将 `casdoor-spring-boot-starter` 依赖项添加到您的 Spring Cloud Gateway 项目中。

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'
```

步骤4：配置您的属性

初始化需要6个参数，所有这些参数都是字符串类型。

名称(按顺序排列)	需要	描述
endpoint	是	Casdoor 服务器 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属性或YAML文件来初始化这些参数。

对于属性:

```
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，您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/**
```

步骤5：添加CasdoorAuthFilter

在网关中添加GlobalFilter接口的实现类，用于身份验证，例如本例中使用的CasdoorAuthFilter。

如果身份验证失败，它将返回401状态码给前端，以重定向他们到登录界面。

```
@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 =
```

步骤6：获取服务并使用它

现在提供5项服务：`CasdoorAuthService`，`CasdoorUserService`，`CasdoorEmailService`，`CasdoorSmsService`，以及`CasdoorResourceService`。

您可以在Gateway项目中按照以下方式创建它们。

```
@Resource
private CasdoorAuthService casdoorAuthService;
```

当你需要身份验证来访问你的应用时，你可以发送目标URL并重定向到Casdoor提供的登录页面。

请确保您已经提前在应用程序配置中添加了回调URL（例如，<http://localhost:9090/callback>）。

```
@RequestMapping("login")
public Mono<String> login() {
    return Mono.just("redirect:" +
        casdoorAuthService.getSignInUrl("http://localhost:9090/callback"));
}
```

在Casdoor成功验证后，它将带着代码和状态被重定向回您的应用程序。您可以获取代码并调用`getOAuthToken`方法来解析出JWT令牌。

`CasdoorUser`包含了Casdoor提供的用户基本信息。您可以将其用作关键字，以在您的应用程序中设置会话。

```
@RequestMapping("callback")
public Mono<String> callback(String code, String state,
```

以下是API的示例。

- 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,`

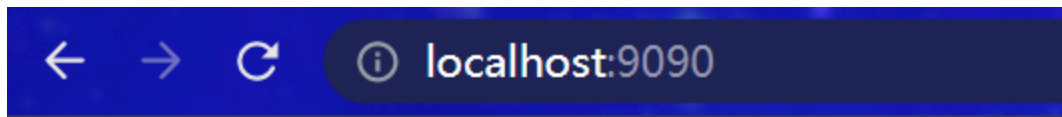
```
fullFilePath, file);
```

- `CasdoorResponse response =`

```
casdoorResourceService.deleteResource(file.getName());
```

步骤7：重启项目

启动项目后，打开你最喜欢的浏览器并访问 <http://localhost:9090>。然后点击任何请求来自 `casdoor-api` 的资源的按钮。



Casdoor

Get Resource

Update Resource

将触发网关认证逻辑。由于您尚未登录，您将被重定向到登录界面。点击登录按钮。



localhost:9090/toLogin

Click to login

Login

随后您可以看到Casdoor统一的登录平台。



Auto sign in

[Forgot password?](#)

Sign In

[No account? sign up now](#)

成功登录后，您将被重定向到主界面。现在你可以点击任何按钮。



localhost:9090

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

以Spring Security为例，演示如何使用OIDC连接到您的应用程序

使用OIDC集成的Spring Security过滤器与Casdoor

本文解释了如何使用Spring Security过滤器通过OIDC将您的应用程序与Casdoor连接。

Spring Security OAuth

Casdoor可以使用OIDC协议作为IDP连接各种应用程序。在本指南中，我们将以Spring Security为例，向您展示如何使用OIDC连接到您的应用程序。

步骤1：部署Casdoor

首先，您需要部署Casdoor。

您可以参考Casdoor 官方文档 [Server Installation](#)。

成功部署Casdoor后，请确保：

- Casdoor服务器正在<http://localhost:8000>上运行。
- 打开您喜欢的浏览器并访问<http://localhost:7001>，您将看到Casdoor的登录页面。
- 通过输入 `admin` 和 `123` 来验证登录功能是否正常工作。

现在，您可以按照以下步骤在自己的应用中快速实现基于Casdoor的登录页面。

步骤2。配置Casdoor应用程序

1. 创建一个新的Casdoor应用程序或使用现有的一个。
2. 添加您的重定向URL（您可以在下一节中找到如何获取重定向URL的更多详细信息）。 ``

Client ID ⓘ :	24a25ea0714d92e78595	Client ID
Client secret ⓘ :	155 [REDACTED]	Client Secret
Redirect URLs ⓘ :	Redirect URLs <input type="button" value="Add"/>	
	Redirect URL	Add a redirect URL for spring security
	http://localhost:8082/ui-one/login/oauth2/code/custom	

3. 添加所需的提供者并填写任何其他设置。

在应用设置页面上，您将找到两个值：`Client ID`和`Client secret`，如上图所示。我们将在下一步中使用这些值。

打开您喜欢的浏览器并访问：`http://CASD00R_HOSTNAME/.well-known/openid-configuration`。在这里，您将找到Casdoor的OIDC配置。

步骤3。配置Spring Security

Spring Security原生支持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:
      client:
        registration:
          casdoor:
            client-id: <Client ID>
            client-secret: <Client Secret>
            scope: <Scope>
            authorization-grant-type: authorization_code
            redirect-uri: <Redirect URL>
        provider:
          casdoor:
            authorization-uri: http://CASD00R_HOSTNAME:7001/login/oauth/
authorize
            token-uri: http://CASD00R_HOSTNAME:8000/api/login/oauth/
access_token
            user-info-uri: http://CASD00R_HOSTNAME:8000/api/get-account
```

```
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>应该像http://<Your Spring Boot Application Endpoint>/<Servlet Prefix if it is configured>/login/oauth2/code/custom这样。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应用程序中配置这个。

您还可以在代码中使用ClientRegistration来自定义设置。您可以在这里找到映射

步骤4：开始使用演示

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** 供用户登录。

```
@EnableWebSecurity
public class UiSecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
```

3. 我们可以为用户添加一个简单的登录页面。

```
<!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>
  </nav>
  <div class="container">
    <label>Welcome!</label> <br /> <a th:href="@{/foos/}"
      class="btn btn-primary">Login</a>
  </div>
</body>
</html>
```

当用户点击 `login` 按钮时，他们将被重定向到 `casdoor`。

4. 接下来，我们可以定义我们的受保护资源。我们可以公开一个名为 `/foos` 的端点和一个用于显示的网页。

数据模型

```
public class FooModel {
  private Long id;
  private String name;

  public FooModel(Long id, String name) {
    super();
    this.id = id;
    this.name = name;
  }
}
```

控制器

```
@Controller
public class FooClientController {
    @GetMapping("/foos")
    public String getFoos(Model model) {
        List<FooModel> foos = new ArrayList<>();
        foos.add(new FooModel(1L, "a"));
        foos.add(new FooModel(2L, "b"));
        foos.add(new FooModel(3L, "c"));
        model.addAttribute("foos", foos);
        return "foos";
    }
}
```

网页

```
<!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>&nbsp;&nbsp;&nbsp;</li>
        </ul>
    </nav>
    <div class="container">
        <h1>All Foos:</h1>
        <table class="table table-bordered table-striped">
            <thead>
```

 注意事项

所有的网页模板都应该放在 `resources/templates` 下。

步骤5：试试演示！

首先，您可以尝试打开您喜欢的浏览器并直接访问 `/foos`。它将自动将您重定向到Casdoor的登录页面。您可以在那里登录或从根页面登录。

如果您访问您的根页面，您将看到Casdoor应用设置。 ``

Spring OAuth Client Thymeleaf - 1

Welcome !

Login

点击 `login` 按钮，页面将重定向您到Casdoor的登录页面。

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

登录后，页面将重定向您到 `/foos`。

Spring OAuth Client Thymeleaf -1

Hi,

Your Username

All Foos:

ID	Name
1	a
2	b
3	c

使用OIDC集成的Spring Security过滤器与Casdoor

Casdoor是一个支持OIDC和各种其他协议的开源IDP。在本文中，我们将看到如何使用Spring Security过滤器和OIDC将Casdoor与您的应用程序集成。

步骤1：部署Casdoor

首先，你需要部署Casdoor服务器。参考[官方文档](#)以获取服务器安装指南。部署成功后，确保：

- Casdoor服务器正在<http://localhost:8000>运行。
- 你可以在<http://localhost:7001>看到Casdoor登录页面。
- 你可以通过使用凭证 `admin` 和 `123` 来测试登录功能。

验证这些步骤后，按照下面的步骤将Casdoor与您的应用程序集成。


步骤2：配置Casdoor应用程序

- 创建一个新的Casdoor应用程序或使用现有的一个。
- 添加你的重定向URL。你可以在下一节中找到更多关于获取重定向URL的信息。

Name ? : application_a6ftas → your application name

Display name ? : New Application - a6ftas

Logo ? :
URL ? : https://cdn.casbin.org/img/casdoor-logo_1185x256.png

Preview: 

Home ? : [↗](#)

Description ? :

Organization ? : organization_carg1b → your organization name

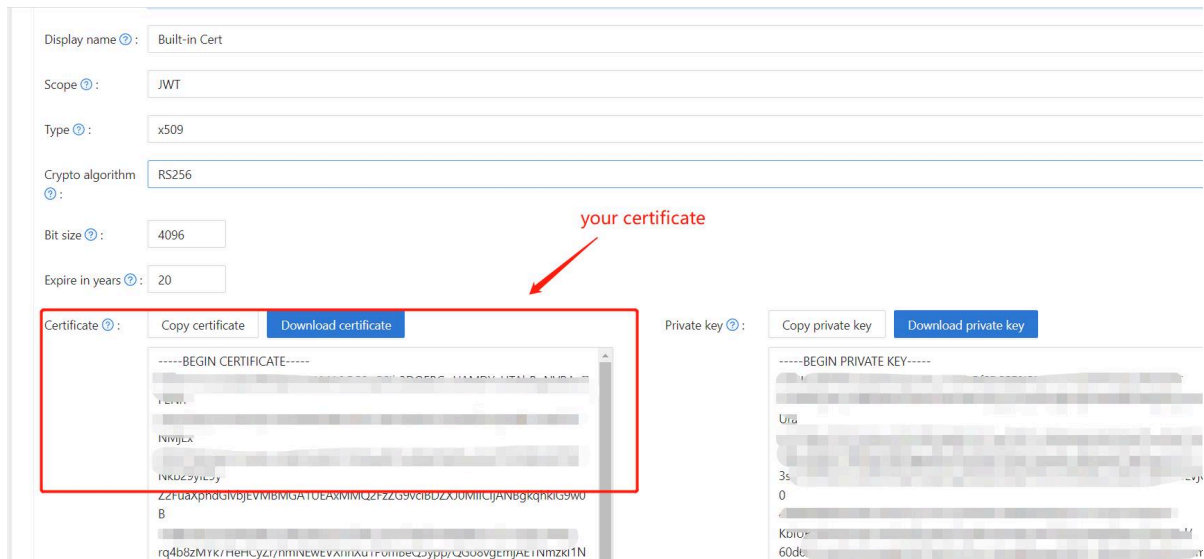
Client ID ? : 3ed7314825ecf955cb19 → your client id

Client secret ? : ee9314ea228 [blurred] → your client secret

Cert ? : cert-built-in

Redirect URLs ? :
Redirect URLs [Add](#)
Redirect URL
<http://localhost:3000/callback> → your redirect url

- 在证书编辑页面上获取你的 `Certificate`。



- 根据需要添加提供商和其他设置。

你可以在应用程序设置页面上获取 `Application Name`、`Organization Name`、`Redirect URL`、`Client ID`、`Client Secret` 和 `Certificate` 的值。我们将在下一步中使用它们。

步骤3：配置Spring Security

你可以自定义Spring Security过滤器的设置来处理令牌：

⚠ 注意事项

确保你用你自己的Casdoor实例替换配置值，特别是 `<Client ID>` 和其他。

```
server:  
  port: 8080  
casdoor:  
  endpoint: http://CASDOOR_HOSTNAME:8000  
  client-id: <Client ID>  
  client-secret: <Client Secret>  
  certificate: <Certificate>  
  organization-name: <Organization Name>  
  application-name: <Application Name>
```

⚠️ 注意事项

对于前端应用程序来说，`<FRONTEND_HOSTNAME>` 的默认值是 `localhost:3000`。在这个演示中，重定向URL是 `http://localhost:3000/callback`。确保在你的 `casdoor` 应用程序中配置这个。

步骤4：配置前端

你需要安装 `casdoor-js-sdk` 并按照以下方式配置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",
};

export const CasdoorSDK = new Sdk(sdkConfig);
```

步骤5：设置演示

1. 创建一个Spring Boot应用程序。
2. 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 {
        // 启用CORS并禁用CSRF
        http = http.cors(corsConfig -> corsConfig
            .configurationSource(configurationSource())
        ).csrf().disable();

        // 将会话管理设置为无状态
        http = http
            .sessionManagement()

        .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
            .and();

        // 在端点上设置权限
        http.authorizeHttpRequests(authorize -> authorize
            .mvcMatchers("/api/redirect-url", "/api/
signin").permitAll()
            .mvcMatchers("/api/**").authenticated()
        );

        // 设置未经授权的请求异常处理程序
```

3. 添加一个简单的JWT过滤器来拦截需要令牌验证的请求。

```
@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 {
        // 获取授权头并验证
        final String header =
request.getHeader(HttpHeaders.AUTHORIZATION);
        if (!StringUtils.hasText(header) ||
!header.startsWith("Bearer ")) {
            chain.doFilter(request, response);
            return;
        }

        // 获取jwt令牌并验证
        final String token = header.split(" ")[1].trim();

        // 获取用户身份并将其设置在spring security上下文中
        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;
        }
    }
}
```

当用户访问需要身份验证的接口时，`JwtTokenFilter` 将从请求头 `Authorization` 获取令牌并验证。

4. 定义一个 `Controller` 来处理用户登录Casdoor时的情况。用户登录后，他们将被重定向到服务器并携带 `code` 和 `state`。然后服务器需要从Casdoor验证用户的身份，并通过这两个参数获取 `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：尝试演示

你可以通过浏览器访问前端应用程序。如果你没有登录，你会看到一个登录按钮。点击它，你将被重定向到Casdoor登录页面。

如果你访问你的根页面,

Casdoor Login

点击 **Casdoor Login** 按钮, 页面将重定向到Casdoor的登录页面。



Auto sign in [Forgot password?](#)

[Sign In](#)

[No account? sign up now](#)

Made with by **Casdoor**

登录后, 你将被重定向到 [/](#)。



New User - rtsbx4

Logout

Jenkins 插件

Casdoor 提供了一个插件，允许用户登录到 Jenkins。在这里，我们将向您展示如何使用 Casdoor 插件进行 Jenkins 安全。

以下是一些配置设置：

`CASDOOR_HOSTNAME`：部署 Casdoor 服务器的域名或 IP。

`JENKINS_HOSTNAME`：部署 Jenkins 的域名或 IP。

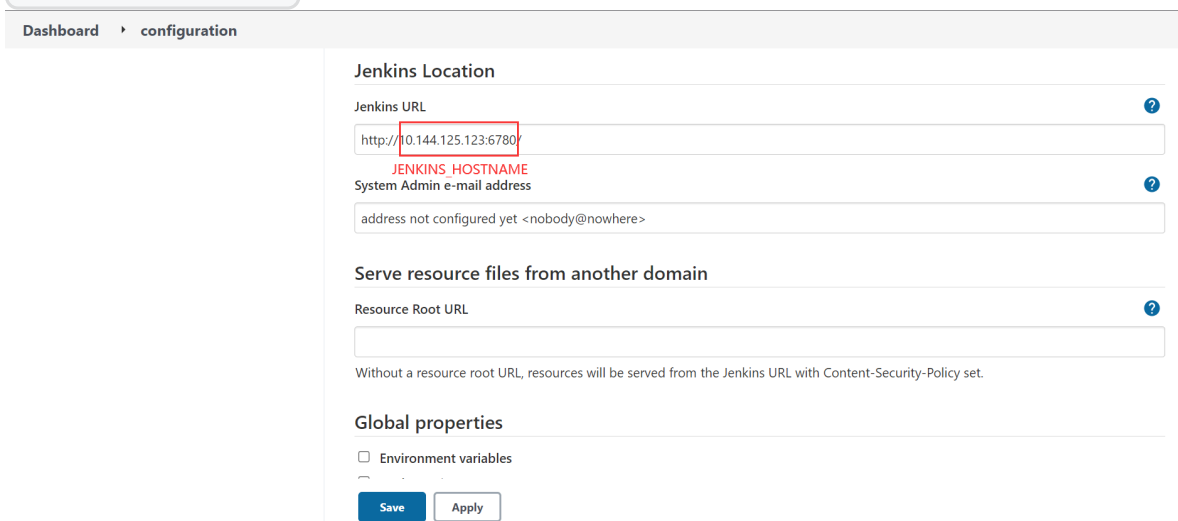
步骤 1：部署 Casdoor 和 Jenkins

首先，部署 [Casdoor](#) 和 [Jenkins](#)。

成功部署后，请确保以下内容：

1. 将 Jenkins URL（管理 Jenkins → 配置系统 → Jenkins 位置）设置为

`JENKINS_HOSTNAME`。



The screenshot shows the Jenkins configuration page with the following details:

- Dashboard > configuration
- Jenkins Location**
 - Jenkins URL: `http://10.144.125.123:6780` (highlighted with a red box)
 - System Admin e-mail address: `address not configured yet <nobody@nowhere>`
- Serve resource files from another domain**
 - Resource Root URL: (empty field)
 - Without a resource root URL, resources will be served from the Jenkins URL with Content-Security-Policy set.
- Global properties**
 - Environment variables
- Buttons: Save, Apply

2. 验证 Casdoor 可以正常登录和使用。
3. 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

1. 创建一个新的 Casdoor 应用或使用现有的一个。
2. 添加一个重定向 URL: `http://JENKINS_HOSTNAME/securityRealm/finishLogin`

`finishLogin`

Description ?	Casdoor for Jenkins				
Organization ?	built-in				
Client ID ?	bbd0bd66696e504dec59 Client ID				
Client secret ?	d2de01b01...110b47465c Client secret				
Redirect URLs ?	<div><p>Redirect URLs Add</p><table><thead><tr><th>Redirect URL</th><th></th></tr></thead><tbody><tr><td>http://10.144.125.123:6780/securityRealm/finishLogin</td><td>Add a redirect url for Jenkins</td></tr></tbody></table></div>	Redirect URL		http://10.144.125.123:6780/securityRealm/finishLogin	Add a redirect url for Jenkins
Redirect URL					
http://10.144.125.123:6780/securityRealm/finishLogin	Add a redirect url for Jenkins				

3. 添加所需的提供商并提供任何额外的设置。

在应用设置页面上，您会找到两个值：`Client ID` 和 `Client secret`，如上图所示。我们将在下一步中使用这些值。

打开您喜欢的浏览器，访问 `http://CASDOOR_HOSTNAME/.well-known/openid-configuration`，查看 Casdoor 的 OIDC 配置。

步骤 3：配置 Jenkins

现在，您可以从市场安装 Casdoor 插件，或者通过上传其 `jar` 文件。

安装完成后，转到管理 Jenkins → 配置全局安全。

建议：备份 Jenkins `config.xml` 文件，并在设置错误的情况下用于恢复。

Configure Global Security

Authentication

Disable remember me

Security Realm

Casdoor Authentication Plugin

Casdoor Endpoint

Client ID

Client Secret

JWT Public Key

Organization Name

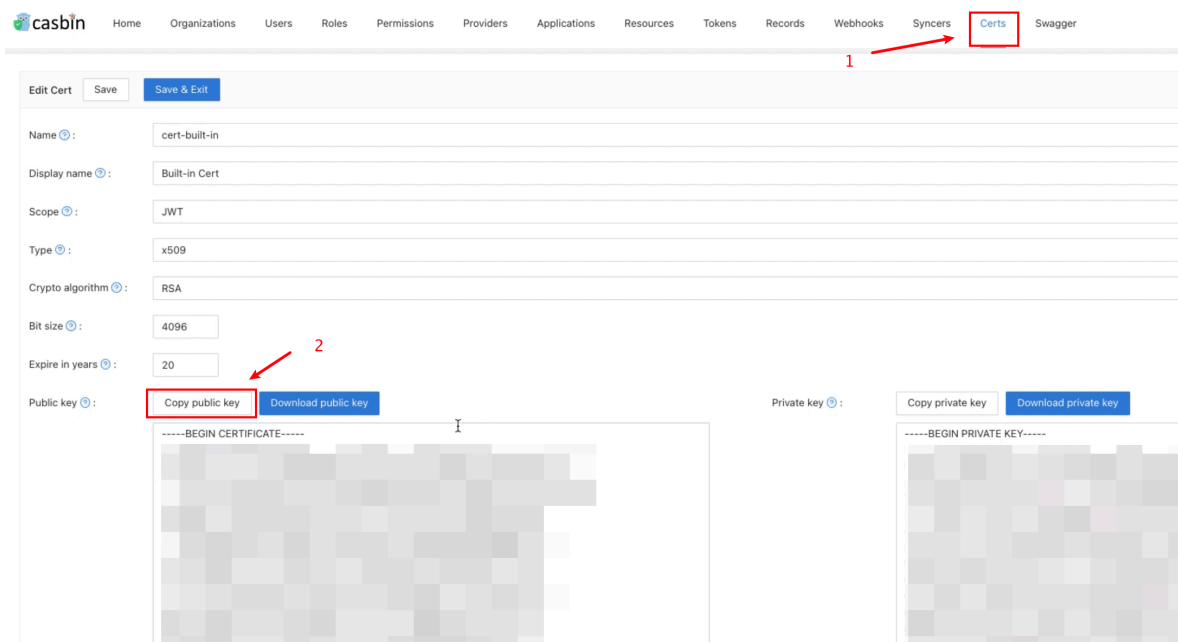
Application Name

Delegate to servlet container

Jenkins' own user database

1. 在安全领域部分，选择 "Casdoor 认证插件"。
2. 在 Casdoor 端点字段中，输入前面提到的 `CASDOOR_HOSTNAME`。
3. 在 Client ID 字段中，输入前面提到的 `Client ID`。

4. 在 Client secret 字段中，输入前面提到的 `Client secret`。
5. 在 JWT 公钥字段中，提供用于验证 JWT 令牌的公钥。您可以通过点击 Casdoor 顶部的 `Cert` 找到公钥。点击您的应用上的 `edit` 后，您可以从以下页面复制公钥。



6. 组织名称和应用名称是可选的。您可以指定您的组织和应用来验证其他组织和应用中的用户。如果这些字段为空，插件将使用默认的组织和应用。
7. 在授权部分，选中 "已登录用户可以做任何事"。禁用 "允许匿名读取访问"。
8. 点击 `Save`。

Jenkins 现在将自动将您重定向到 Casdoor 进行身份验证。

Jenkins OIDC

Casdoor可以使用OIDC协议作为IDP连接各种应用。在这个例子中，我们将使用Jenkins来演示如何使用OIDC连接到你的应用。

以下是配置中使用的一些名称：

- `CASDOOR_HOSTNAME`：部署Casdoor服务器的域名或IP。
- `JENKINS_HOSTNAME`：部署Jenkins的域名或IP。

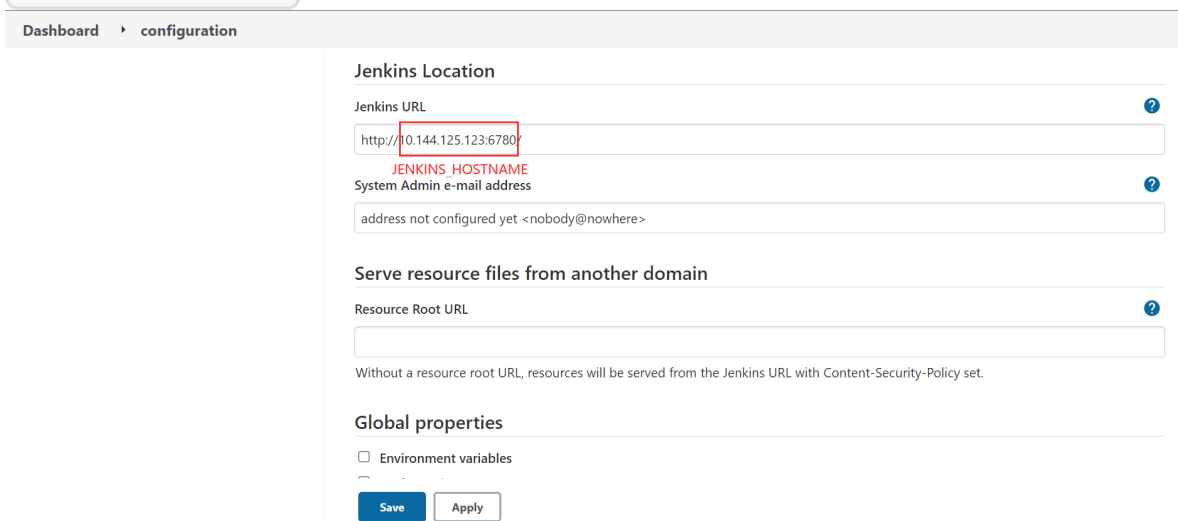
步骤1：部署Casdoor和Jenkins

首先，部署Casdoor和Jenkins。

成功部署后，确保以下内容：

1. 将Jenkins URL（管理Jenkins → 配置系统 → Jenkins位置）设置为

`JENKINS_HOSTNAME`。



The screenshot shows the Jenkins configuration page for the 'Jenkins Location' section. The 'Jenkins URL' field is set to 'http://0.144.125.123:6780', with the IP address part highlighted by a red box. Below it, the 'System Admin e-mail address' field is set to 'address not configured yet <nobody@nowhere>'. The 'Serve resource files from another domain' section is also visible, with the 'Resource Root URL' field empty. At the bottom, there are 'Save' and 'Apply' buttons.

2. 确保Casdoor可以正常登录和使用。
3. 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

1. 创建一个新的Casdoor应用或使用现有的一个。
2. 添加一个重定向URL: `http://JENKINS_HOSTNAME/securityRealm/finishLogin`

finishLogin

Description ⓘ: Casdoor for Jenkins

Organization ⓘ: built-in

Client ID ⓘ: bbd0bd66696e504dec59 Client ID

Client secret ⓘ: d2de01b01...110b47465c Client secret

Redirect URLs ⓘ:

Redirect URL	
http://10.144.125.123:6780/securityRealm/finishLogin	Add a redirect url for Jenkins

JENKINS_HOSTNAME

3. 添加你想要的提供商，并提供任何额外的设置。

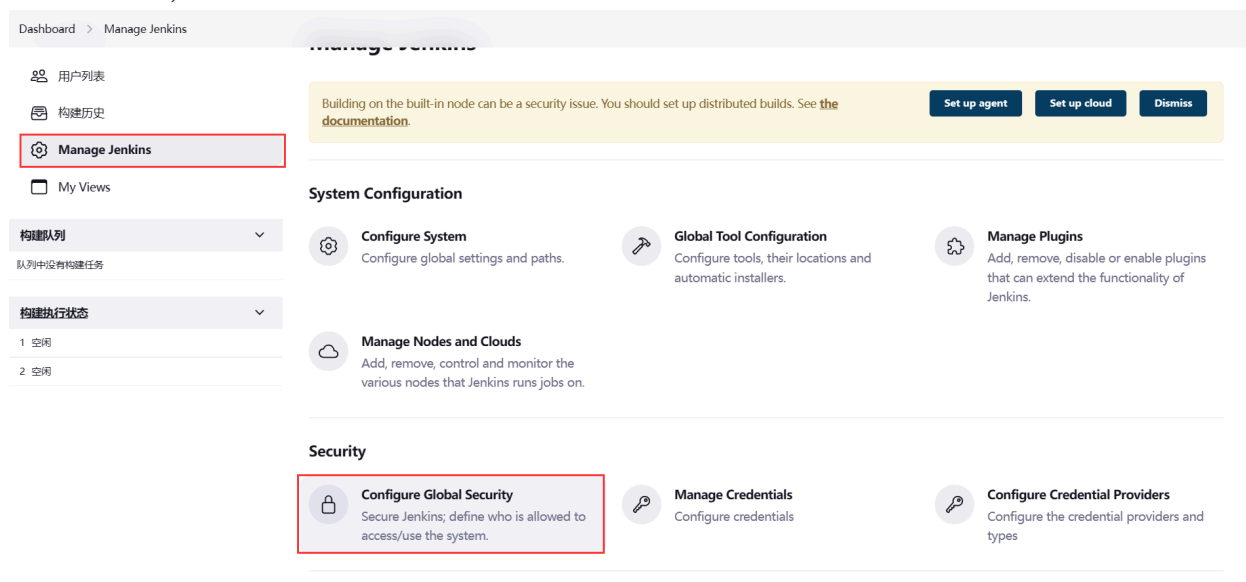
你将从应用设置页面获取两个值：`Client ID`和`Client secret`。我们将在下一步中使用这些值。

打开你最喜欢的浏览器并访问：`http://CASDOOR_HOSTNAME/.well-known/openid-configuration`查看Casdoor的OIDC配置。

步骤3：配置Jenkins

首先，我们需要安装`OpenId Connect Authentication`，因为Jenkins并不原生支持OIDC。

安装完成后，转到**管理Jenkins** → **配置全局安全**。



💡 提示

确保备份Jenkins的`config.xml`文件，以防任何设置错误。

1. 在访问控制中，选择**使用Openid Connect登录**作为**安全领域**。

2. 在Client ID字段中指定上面记下的Client ID。
3. 在Client secret字段中指定上面记下的Client secret。
4. 在配置模式中，选择自动配置，并输入http://CASDOOR_HOSTNAME/.well-known/openid-configuration作为已知配置端点。

Security Realm

Delegate to servlet container ?
 Jenkins' own user database ?
 Login with Openid Connect ? **Select this**

Client id ?

bbd0bd66696e504dec59 **Input your Client ID**

Client secret ?

Concealed **Input your Client secret** Change Password

Configuration mode

Automatic configuration ?
 Manual configuration ?

Well-known configuration endpoint ?

http://10.144.1.2:8000/.well-known/openid-configuration

CASDOOR_HOSTNAME

如果你的Casdoor是本地部署的，你可能需要选择手动配置并提供以下信息：

- 令牌服务器URL：http://CASDOOR_HOSTNAME/api/login/oauth/access_token
- 授权服务器URL：http://CASDOOR_HOSTNAME/login/oauth/authorize
- 用户信息服务器URL：http://CASDOOR_HOSTNAME/api/get-account
- 范围：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

CASDOOR_HOSTNAME

Authorization server url ?

http://10.144.1.2:8000/login/oauth/authorize

UserInfo server url ?

http://10.144.1.2:8000/api/get-account

Scopes

address phone openid profile offline_access email

5. 点击**高级设置**并填写以下内容：

- 在用户名字段中，指定 `name`。
- 在全名字段中，指定 `displayName`。
- 在电子邮件字段中，指定 `email`。



The screenshot shows a form with five input fields, each with a label above it:

- User name field name**: Input field containing the text "name".
- Full name field name**: Input field containing the text "displayName".
- Email field name**: Input field containing the text "email".
- Groups field name**: Input field that is currently empty. A small question mark icon is visible to the right of the label.
- Token Field Key To Check**: Input field that is currently empty. A small question mark icon is visible to the right of the label.

6. 在**授权**部分，启用“已登录用户可以做任何事情”并禁用“允许匿名读取访问”。你可以稍后配置更复杂的授权，但现在，检查OpenID是否正常工作。

退出Jenkins，它应该将你重定向到Casdoor进行身份验证。



Auto sign in

[Forgot password?](#)

[Sign In](#)

[Sign in with code](#)

[No account? sign up now](#)



Jira

通过内置的SSO

使用OIDC协议作为IDP连接各种应用程序，如Jira

使用miniOrange插件

使用OIDC协议作为IDP连接casdoor和Jira

通过内置的SSO

这是连接Casdoor的免费方法，但您的网站必须使用HTTPS。

Casdoor可以使用OIDC协议作为IDP连接各种应用程序。这里有一个[Jira教程](#)。

以下是配置中的一些名称：

- `CASD00R_HOSTNAME`：部署Casdoor服务器的域名或IP。
- `Jira_HOSTNAME`：部署Jira的域名或IP。

步骤1：部署Casdoor和Jira

首先，部署[Casdoor](#)和[Jira](#)。

成功部署后，请确保以下内容：

1. Casdoor可以正常登录和使用。
2. 在`prod`模式下部署Casdoor时，您可以将`CASD00R_HOSTNAME`设置为`http://localhost:8000`。查看[生产模式](#)。

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. 找到认证方法：

The screenshot shows the Jira Administration interface. The top navigation bar includes 'Jira Software', 'Dashboards', 'Projects', 'Issues', 'Boards', 'Plans', and 'Create'. The main content area is titled 'Administration' and contains a search bar and a navigation menu. The 'System' link in the navigation menu is highlighted with a red box and a '2' next to it. The 'Authentication methods' section is active, showing a warning message: 'Make authentication safer. Authenticating with username and password is less secure than through single sign-on. Now that you've configured the latter, consider disabling product login form and basic authentication. Communicate this change to your users. How to disable · Dismiss'. Below this is a table of login options:

Name	Type	Last updated	Show on login page	Actions
Username and password	Product login form	Never	<input checked="" type="checkbox"/>	...
casdoor	OpenID Connect	25 April 2023 4:33 PM	<input checked="" type="checkbox"/>	...

Below the table, there is a section for 'Authentication on API calls' with a checked toggle for 'Allow basic authentication on API calls'. The left sidebar contains various administration categories like 'General configuration', 'SYSTEM SUPPORT', 'SECURITY', and 'Authentication methods' (highlighted with a red box and a '3' next to it). The bottom of the page shows the URL: <https://test.v2tl.com/plugins/servlet/applications/versions-licenses>.

3. 添加一个配置，并在认证方法中选择OpenID连接单点登录

Add new configuration

Name *

Use a unique name for this configuration.

Authentication method

OpenID Connect single sign-on



Users log in using OpenID Connect

4. 找到重定向URL：

Give these URLs to your identity provider

Redirect URL

<https://test.v2tl.com/plugins/servlet/oidc/callback>



Location where the client is sent to after successful account authentication.

5. 添加一个重定向URL：

The screenshot shows a configuration interface with the following fields:

- Client ID: 642ec5d6779a2f0e879d
- Client secret: 26cb47985c47ae3844580536ce2f59872969e109
- Cert: cert-built-in
- Redirect URLs: A table with one entry:

Redirect URL	Action
https://test.v2tl.com/plugins/servlet/oidc/callback	Up, Down, Delete icons

不出所料，您可以在应用设置页面上获取两个值：`Client ID`和`Client secret`，如上图所示。我们将在下一步中使用它们。

打开您最喜欢的浏览器并访问：`http://CASD00R_HOSTNAME/.well-known/openid-configuration`。您将看到Casdoor的OIDC配置。

步骤3：配置Jira

1. 我们需要继续在Jira中配置我们的配置

Edit existing configuration

Name *

Use a unique name for this configuration.

Authentication method

OpenID Connect single sign-on

Users log in using OpenID Connect

OpenID Connect settings

Issuer URL *

your casdoor url

The complete URL of the OpenID Provider. Needs to be unique.

Client ID *

application client ID

The client identifier, as registered with the OpenID Provider.

Client secret *

application client secret [Change](#)

Client secret is used in conjunction with the Client ID to authenticate the client application against the OpenID Provider.

Username mapping *

Used to map IdP claims to the username, e.g. \${sub}

Additional scopes

phone × email × address × profile ×

The default scope is 'openid'. Add more scopes if needed to obtain the username claim.

Redirect URL



Copy it to casdoor

Location where the client is sent to after successful account authentication.

Initiate login URL



URL used for OpenID Provider-initiated login.

Additional settings

The authorization, token, and user info endpoints will be filled automatically if your Identity provider offers this option. If not, you will be asked to provide this information.

- Fill the data automatically from my chosen identity provider.

JIT provisioning

Just-in-time user provisioning allows users to be created and updated automatically when they log in through SSO to Atlassian Data Center applications. [Learn more](#).

- Create users on login to the application

OpenID Connect behaviour

- Remember user logins

If checked, successful login history will be saved and users will be logged in automatically without the need for reauthentication.

Login page settings

Decide if the IdP should be visible on login page and customize what the user will see on the button.

- Show IdP on the login page

Login button text *

The text is shown to the user on the login page. Remaining characters: 33.

Save configuration

Cancel

2. 您可以稍后配置更复杂的授权。现在，检查OpenID是否真的有效。

Administration 🔍 Search Jira admin

Applications Projects Issues Manage apps User management Latest upgrade report **System**

- General configuration
 - Find more admin tools
 - Jira mobile app
- SYSTEM SUPPORT
 - System info
 - Instrumentation
 - Monitoring
 - Database monitoring
 - Integrity checker
 - Logging and profiling
 - Scheduler details
 - Troubleshooting and support tools
 - Clean up
 - Audit log
 - Clustering
- SECURITY
 - Project roles
 - Global permissions

Authentication methods ➕ Add configuration

Manage how users authenticate. Save authentication configurations using SAML, OpenID Connect, or Crowd as the identity provider. [Learn more about using multiple identity providers.](#)

⚠ Make authentication safer
Authenticating with username and password is less secure than through single sign-on. Now that you've configured the latter, consider disabling product login form and basic authentication. Communicate this change to your users.
[How to disable](#) - [Dismiss](#)

Login options

Name	Type	Last updated	Show on login page	Actions
Username and password	Product login form	Never	<input checked="" type="checkbox"/>	⋮
casdoor	OpenID Connect	26 April 2023 7:20 PM	<input checked="" type="checkbox"/>	⋮

Authentication on API calls

Allow basic authentication on API calls.
You can use personal access tokens as a safer alternative method of authentication. See [Using personal access tokens](#).

使用miniOrange插件

本教程解释如何使用miniOrange连接casdoor和Jira。

Casdoor可以使用OIDC协议作为IDP连接各种应用程序。你可以参考这个[Jira](#)教程以获取更多信息。

以下是配置中的一些重要名称：

`CASD00R_HOSTNAME`：部署Casdoor服务器的域名或IP。

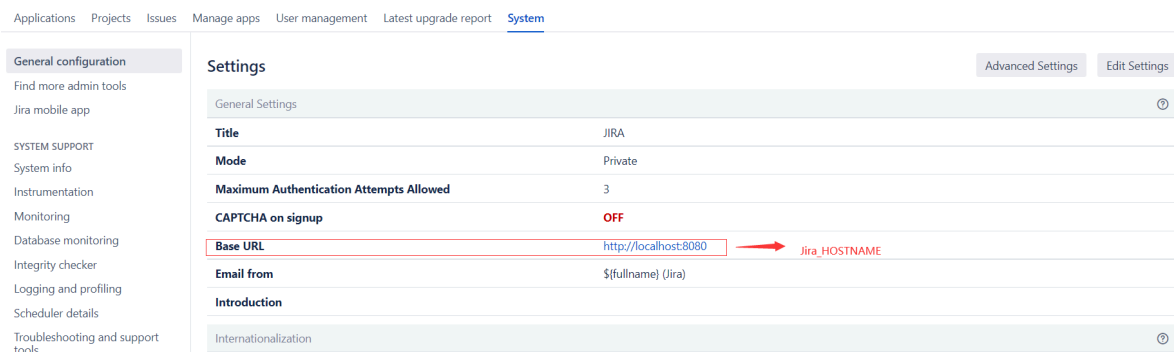
`Jira_HOSTNAME`：部署Jira的域名或IP。

步骤1：部署Casdoor和Jira

首先，部署Casdoor和Jira。

成功部署后，请确保：

1. 将Jira URL（计划 → 管理 → 系统 → 通用配置）设置为 `Jira_HOSTNAME`。

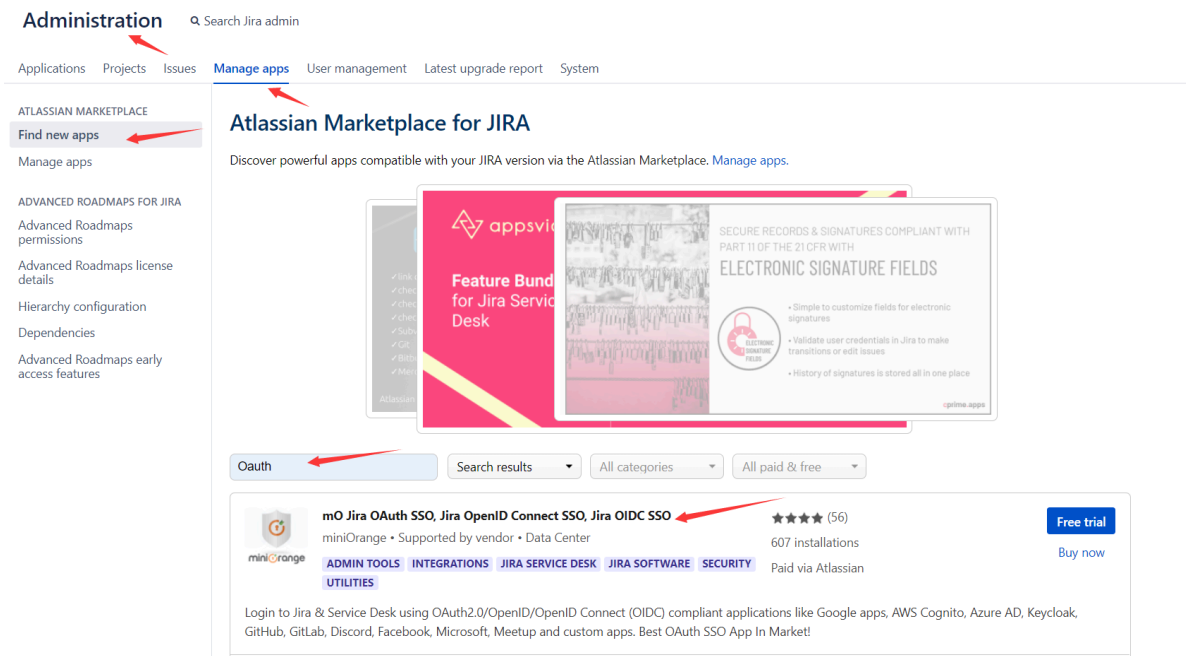


2. Casdoor 可以正常登录使用。
3. 当在 `prod` 模式下部署Casdoor时，你可以将 `CASD00R_HOSTNAME` 设置为

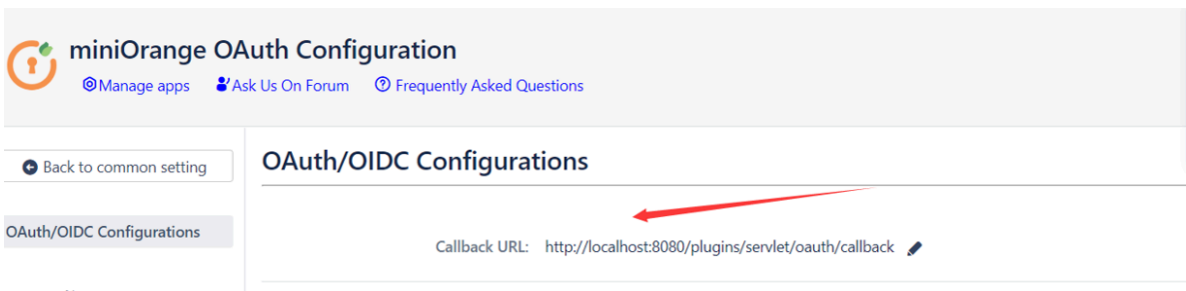
`http://localhost:8000`。详见 [生产模式](#)。

步骤2：配置Casdoor应用程序和Jira

1. 创建一个新的Casdoor应用程序或使用现有的一个。
2. 安装miniOrange应用程序以支持OAuth。你可以在计划→管理→查找新应用→搜索中找到这个应用程序



3. 将 `Selected Application` 设置为 Custom OpenId。
4. 找到重定向URL：



5. 添加重定向URL：

Client ID	514e09591ee5554b16fe				
Client secret	e7f05b14a68fb23e526f08515aefb73bbab7814a				
Cert	cert-built-in				
Redirect URLs	<table border="1"> <thead> <tr> <th>Redirect URL</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td>http://localhost:8080/plugins/servlet/oauth/callback</td> <td> <input type="button" value="Add"/> <input type="button" value="Up"/> <input type="button" value="Down"/> <input type="button" value="Delete"/> </td> </tr> </tbody> </table>	Redirect URL	Action	http://localhost:8080/plugins/servlet/oauth/callback	<input type="button" value="Add"/> <input type="button" value="Up"/> <input type="button" value="Down"/> <input type="button" value="Delete"/>
Redirect URL	Action				
http://localhost:8080/plugins/servlet/oauth/callback	<input type="button" value="Add"/> <input type="button" value="Up"/> <input type="button" value="Down"/> <input type="button" value="Delete"/>				

6. 按照以下方式配置应用程序：

Selected Application: Custom OpenId Import Details

Provider ID: **5c881c25-2e02-42c9-af06-0a71e0beb516**

Custom App Name:

Client Id:

Client Secret:

Scope:

Authorize Endpoint:

Access Token Endpoint:

Logout Endpoint:

Enter the Logout endpoint of your OAuth/OpenID Provider. Leave blank if Logout endpoint not supported by provider.
e.g. If Keycloak Logout endpoint is configured with {hostname}/auth/realms/{realm-name}/protocol/openid-connect/logout Keycloak too.

Save
Test Configuration

- **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`

打开你最喜欢的浏览器并访问：`http://CASDOOR_HOSTNAME/.well-known/openid-configuration`。你将看到Casdoor的OIDC配置。

退出Jira并测试SSO。

Welcome to JIRA

Username

Password

Remember my login on this computer

Not a member? To request an account, please contact your Jira administrators.

[Log In](#) [Use OAuth Login](#)

[Can't access your account?](#)

使用OIDC协议连接应用程序 - Confluence

[Casdoor](#)可以使用OIDC协议作为IDP连接各种应用程序。在本指南中，我们将使用[Confluence](#)作为示例，演示如何使用OIDC连接您的应用程序。

首先，确保您已成功部署Casdoor和Confluence。以下是您需要记住的一些配置名称：

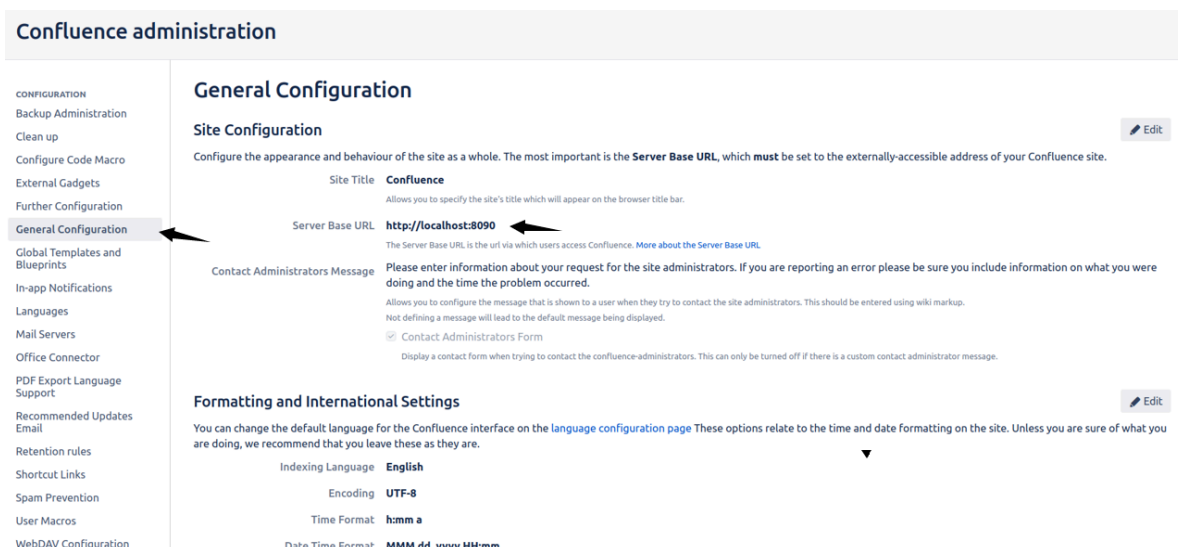
- `CASD00R_HOSTNAME`：部署Casdoor服务器的域名或IP。
- `Confluence_HOSTNAME`：部署Confluence的域名或IP。

步骤1：部署Casdoor和Confluence

首先，部署[Casdoor](#)和[Confluence](#)。

成功部署后，请确保以下内容：

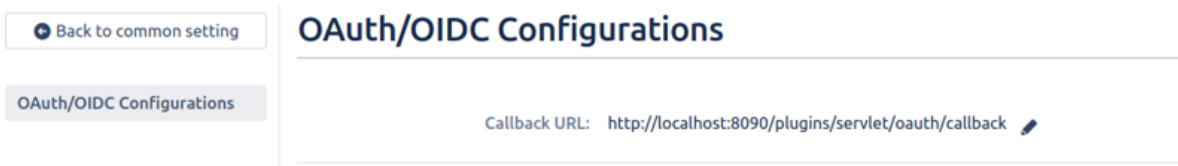
1. 将Confluence URL设置为`Confluence_HOSTNAME`。



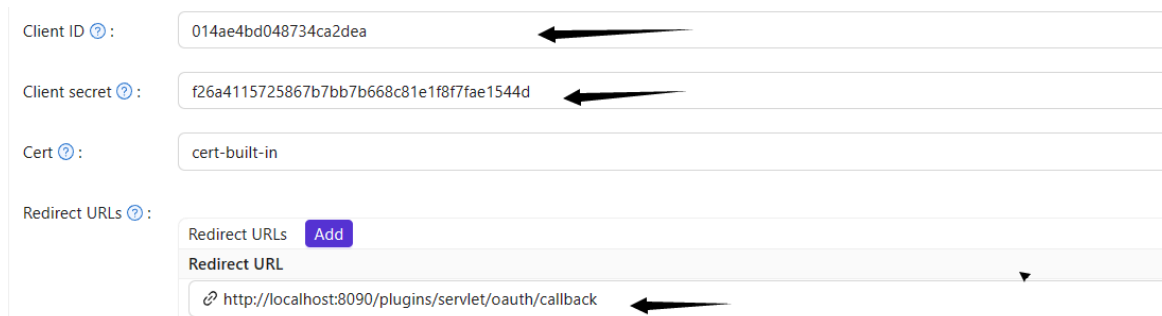
2. Casdoor可以正常登录和使用。
3. 如果您在 `prod` 模式下部署Casdoor，可以将 `CASD00R_HOSTNAME` 设置为 `http://localhost:8000`。有关更多详细信息，请参阅[生产模式](#)。

步骤2：配置Casdoor应用程序

1. 创建一个新的Casdoor应用程序或使用现有的一个。
2. 找到一个重定向URL：



3. 将重定向URL添加到应用程序中：



4. 添加所需的提供者并相应地配置其他设置。

在应用程序设置页面上，您将找到两个值：`Client ID`和`Client Secret`。我们将在下一步中需要这些。

打开您喜欢的浏览器并访问：`http://CASD00R_HOSTNAME/.well-known/openid-configuration`以查看Casdoor的OIDC配置。

步骤3：配置Confluence

1. 安装miniOrange应用程序以支持OAuth。您可以在以下位置找到此应用程序：

The screenshot shows the Atlassian Marketplace interface. On the left, the 'ATLASSIAN MARKETPLACE' section is expanded, with 'Find new apps' selected. The main content area displays search results for 'oauth'. The top result is 'mO Confluence OAuth SSO, Confluence OpenID Connect/OIDC SSO' by miniOrange, which is highlighted with a red arrow. Below it is 'Table Filter and Charts for Confluence' by Stiltsoft. The search bar contains 'oauth' and the results are filtered by 'All categories' and 'All paid & free'.

2. 配置应用程序：

Selected Application: **Custom OpenId** [Import Details](#) [Setup Guide](#)

Provider ID: **4f6b30c1-eba8-4b89-ac02-4a4b7a137b97**

Custom App Name:*

Client Id:*

Client Secret:*

Scope:*

Authorize Endpoint:*

Access Token Endpoint:*

Logout Endpoint:

Enter the Logout endpoint of your OAuth/OpenID Provider. Leave blank if Logout endpoint not supported by provider.
e.g. If Keycloak Logout endpoint is configured with {hostname}/auth/realms/{realm-name}/protocol/openid-connect/logout URL then on

3. 将 Selected Application 设置为 Custom OpenID。
4. 从 Casdoor 应用程序页面获取 Client ID 和 Client Secret。

为 Confluence 配置以下设置：

- 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`

您可以稍后配置更高级的授权设置。现在，检查 OpenID 是否真的有效。

退出 Confluence 并测试 SSO：

Firefox 网络浏览器 4月1日 22:07

Log in - Confluence

localhost:8090/login.action?language=en_GB

Confluence

Log in

Username

Password

Remember me

[Log in](#) [Use OAuth Login](#)

[Forgot your password?](#)

EVALUATION LICENSE Are you enjoying Confluence? Please consider purchasing it today.

Cestina · Dansk · Deutsch · Eesti · English (UK) · English (US) · Español · Français · Íslenska · Italiano · Magyar · Nederlands · Norsk · Polski · Português · Română · Slovenčina · Suomi · Svenska · Pycckий · 中文 · 日本語 · 한국어

Powered by Atlassian Confluence 8.1.1 · [Report a bug](#) · [Atlassian News](#)

ATLASSIAN

localhost:8090/pluginservlet/oauth/authorize?return_to=https://localhost:8090&id=4f6b30c1-eba8-4b89-ac02-4a4b7a137b97&idp=4f6b30c1-eba8-4b89-ac02-4a4b7a137b97

RuoYi

Casdoor可以轻松地与RuoYi-cloud集成。

步骤1：部署Casdoor

首先，部署Casdoor。

您可以参考Casdoor 官方文档 [Server Installation](#) 。

部署成功后，请确保以下内容：

- Casdoor 服务器正在运行于 <http://localhost:8000>。
- 打开你最喜欢的浏览器，访问 <http://localhost:7001> 来访问Casdoor登录页面。
- 通过输入 `admin` 和 `123` 来测试登录功能。

接下来，你可以按照以下步骤在你自己的应用中快速实现一个基于Casdoor的登录页面。

步骤2：配置Casdoor

要配置Casdoor，请按照以下步骤操作：

1. 通过点击[这里](#)在浏览器中打开Casdoor。 建议使用与您的开发浏览器不同的浏览器。
2. 在Casdoor中配置一个组织，一个应用程序和同步器。 您可以在[这里](#)找到详细的操作指南。

以下是一些需要记住的额外要点：

1. 编辑同步器时，请务必检查表格列：

Column name	Column type	Casdoor column	Is hashed	Action
user_id	integer	Id	<input type="checkbox"/>	↑ ↓ □
dept_id	integer	Affiliation	<input type="checkbox"/>	↑ ↓ □
user_name	string	Name	<input type="checkbox"/>	↑ ↓ □
nick_name	string	DisplayName	<input type="checkbox"/>	↑ ↓ □
user_type	string	Type	<input type="checkbox"/>	↑ ↓ □
email	string	Email	<input type="checkbox"/>	↑ ↓ □
phonenumber	string	Phone	<input type="checkbox"/>	↑ ↓ □
sex	string	Gender	<input type="checkbox"/>	↑ ↓ □
avatar	string	Avatar	<input type="checkbox"/>	↑ ↓ □
password	string	Password	<input type="checkbox"/>	↑ ↓ □
del_flag	string	IsDeleted	<input type="checkbox"/>	↑ ↓ □
login_ip	string	Createdip	<input type="checkbox"/>	↑ ↓ □
create_time	string	CreatedTime	<input type="checkbox"/>	↑ ↓ □
password	string	Password	<input type="checkbox"/>	↑ ↓ □

○

2. 在编辑组织时，确保选择正确的密码类型：

Password type :

○

3. 最后，确保你已启用软删除。

请确保仔细遵循这些指示，以正确配置Casdoor。

步骤3. 前端改造

3.1 跳转到Casdoor的登录页面

我们可以使用一个前端SDK，以vue-sdk为例。在你初始化 vue-sdk 之后，你可以通过使用 `getSignInUrl()` 函数来获取 Casdoor 登录页面的 URL。

您可以按照您喜欢的方式进行链接，并随时删除Ruoyi-Cloud中不再需要的任何原始代码，例如原始的账户和密码el-input。

3.2 接受Casdoor返回的代码和状态

通过Casdoor成功登录后，Casdoor会将代码和状态发送到我们设置的页面。我们可以使用create()函数来获取代码和状态。

```
created() {
  let url = window.document.location.href; // 获取URL
  let u = new URL(url);
  this.loginForm.code = u.searchParams.get('code'); // 获取code和
state
  this.loginForm.state = u.searchParams.get('state');
  if (this.loginForm.code != null && this.loginForm.state !=
null) { // 如果code和state不为null, 执行handleLogin
    this.handleLogin();
  }
}
```

对于RuoYi-Cloud，我们只需修改其原来的发送账号和密码的方法，改为发送代码和状态。因此，变化仅在于与原始登录相关的发送到后端的内容。

步骤4：重构你的后端

4.1 接受前端返回的代码和状态

```
@PostMapping("login")
public R<?> callback(@RequestBody CodeBody code) {
  String token =
casdoorAuthService.getOAuthToken(code.getCode(), code.getState());
  CasdoorUser casdoorUser =
casdoorAuthService.parseJwtToken(token);
  如果 (casdoorUser.getName() != null) {
```

在这个方法中，我们使用了casdoor-SpringBoot-sdk方法，并对RuoYi-Cloud方法进行了轻微的修改。

例如，RuoYi-Cloud原始方法使用密码注册账户。我已将其更改为使用 `casdoorRegister` 方法注册账户。

我还添加了一个方法 `getUserByCasdoorName` 来检查账户是否存在，并将方法 `executeUserInfo` 更改为 `executeWithAccount` 以反映这一变化。

这是一个简单的修改，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

- 需要移除现有的登录和注册页面。
- 此外，前端需要接受代码和状态参数，并将它们发送到后端。

5.2 后端

RuoYi后端已经实现了完善的登录和注册功能。我们只需要做一些小修改，这使得整个过程非常方便。

步骤6：详细步骤

1. 部署并配置Casdoor。确保为组织选择bcrypt密码类型，因为RuoYi-Cloud也使用bcrypt来处理密码。
2. 使用Casdoor同步器将数据库用户复制到您的Casdoor组织。这将把原始账户导入到Casdoor。

- 部署Casdoor后，对前端进行更改。禁用 RuoYi 检查代码。

```
// checkcode switch
captchaEnabled: false,
// register switch
register: true,
```

请注意，RuoYi-Cloud的验证码需要再次在Nacos中禁用。另外，RuoYi-Cloud的注册功能需要通过设置 `sys.account.registerUser` 为 `true` 来启用。

- 为用户添加一个用Casdoor登录的按钮，并修改数据的 `loginForm`。

```
<a href="http://localhost:7001/login/oauth/authorize?client_id=d509b6b3edc8a3d4cce9&response_type=code&redirect_uri=http%3A%2F%2Flocalhost%3A7001%2Fcasdoor" >casdoor</a>
loginForm: {
  code: "",
  state: ""
},
```

这里，我已经写下了URL，但你可以使用Casdoor-Vue-SDK或Casdoor-SpringBoot-SDK来获取它。

- 由于我们不再使用原始的登录方法，删除cookie和checkcode方法。

新的 `created` 函数应该如下所示：

```
created() {
  let url = window.document.location.href; // 获取URL
  let u = new URL(url);
  this.loginForm.code = u.searchParams.get('code'); // 获取
code和state
  this.loginForm.state = u.searchParams.get('state');
  if (this.loginForm.code !== null && this.loginForm.state !==
null) { // 如果code和state不为null, 执行handleLogin
```

6. 事实上，我们只需要更改我们发送到后端的参数，并删除不必要的函数。不需要进行其他更改。

```
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. 在后端导入所需的依赖项。

pom.xml

```
<dependency>
  <groupId>org.casbin</groupId>
  <artifactId>casdoor-spring-boot-starter</artifactId>
  <version>1.2.0</version>
</dependency>
```

您还需要在资源文件中配置Casdoor。

8. 将回调函数定义为重定向函数。对 `sysLoginService` 中的一些方法进行更改。删除密码检查步骤，因为它不再需要。

```
@PostMapping("login")
public R<?> callback(@RequestBody CodeBody code) {
  // 定义一个带有code和state的CodeBody实体
  String token =
  casdoorAuthService.getOAuthToken(code.getCode(),
```

9. 向 SysLoginService 添加新方法。

```
public LoginUser casdoorLogin(String username) {
    R<LoginUser> userResult =
remoteUserService.getUserInfo(username,
SecurityConstants.INNER);
    // 执行用户
    if (R.FAIL == userResult.getCode()) {
        throw new ServiceException(userResult.getMsg());
    }

    if (StringUtils.isNull(userResult) ||
StringUtils.isNull(userResult.getData())) {
        recordLogService.recordLogininfor(username,
Constants.LOGIN_FAIL, "该用户不存在");
        throw new ServiceException("用户 " + username + " 不存
在");
    }
    LoginUser userInfo = userResult.getData();
    SysUser user = userResult.getData().getSysUser();
    if
(UserStatus.DELETED.getCode().equals(user.getDelFlag())) {
        recordLogService.recordLogininfor(username,
Constants.LOGIN_FAIL, "对不起, 您的账户已被删除");
        throw new ServiceException("对不起, 您的账户 " + username
+ " 已被删除");
    }
    if (UserStatus.DISABLE.getCode().equals(user.getStatus()))
{
        recordLogService.recordLogininfor(username,
Constants.LOGIN_FAIL, "您的账户已被禁用. 请联系管理员");
        throw new ServiceException("对不起, 您的账户 " + username
+ " 已被禁用");
    }
    recordLogService.recordLogininfor(username,
Constants.LOGIN_SUCCESS, "登录成功");
    return userInfo;
}
```

```

public String getUserByCasdoorName(String casdoorUsername) {
    R<LoginUser> userResult =
remoteUserService.getUserInfo(casdoorUsername,
SecurityConstants.INNER);
    if (StringUtils.isNull(userResult) ||
StringUtils.isNull(userResult.getData())) {
        // 如果用户不在RuoYi-Cloud数据库中, 但在Casdoor中存在, 则在数
        据库中创建用户
        return null;
    }
    String username =
userResult.getData().getSysUser().getUserName();
    return username;
}

```

```

public void casdoorRegister(String username) {
    if (StringUtils.isAnyBlank(username)) {
        throw new ServiceException("用户必须提供用户名");
    }
    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, "注册成功");
}

```

Pulsar Manager

Casdoor 可以轻松连接到 Pulsar Manager。

连接 Casdoor 的代码已经添加到 Pulsar Manager 中，所以我们只需要在后端配置 `application.yml` 文件并启用前端开关。

步骤 1：部署 Casdoor

首先，部署 Casdoor。

您可以参考官方 Casdoor 文档中的 [服务器安装](#)。

部署成功后，请确保以下内容：

- Casdoor 服务器在 <http://localhost:8000> 成功运行。
- 打开您喜欢的浏览器并访问 <http://localhost:7001>。您应该看到 Casdoor 的登录页面。
- 通过输入 `admin` 和 `123` 来测试登录功能。

现在，您可以使用以下步骤在您自己的应用中快速实现基于 Casdoor 的登录页面。

步骤 2：配置 Casdoor

要配置 Casdoor，请参考 [Casdoor](#)（建议使用与开发浏览器不同的浏览器）。


您还应该配置一个组织和一个应用。您可以参考 [Casdoor](#) 获取详细的操作指南。

步骤 2.1: 创建一个组织

Edit Organization

Name

Display name

Favicon
Preview: 

Website URL

Password type


Password salt

Phone prefix

步骤 2.2: 创建一个应用

Name

Display name

Logo
Preview: 

Home

Description

Organization

Client ID

Client secret

Cert

Redirect URLs

Redirect URL	Action
<input type="text" value="http://localhost:9527/callback"/>	<input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="✕"/>

步骤 3: 启用 Pulsar Manager 前端开关

启用此开关以将代码和状态发送到后端。

您可以在 `pulsar-manager/front-end/src/router/index.js` 的第 80 行找到该开关。

```
- // mode: 'history', // require service support  
+ mode: 'history', // require service support
```

步骤 4：配置后端代码

在 `application.properties` 文件中配置 Casdoor 的设置，该文件可以在 `pulsar-manager/src/main/resources/application.properties` 的第 154 行找到。

```
casdoor.endpoint = http://localhost:8000  
casdoor.clientId = <client id from previous step>  
casdoor.clientSecret = <client secret from previous step>  
casdoor.certificate = <client certificate from previous step>  
casdoor.organizationName = pulsar  
casdoor.applicationName = app-pulsar
```


在 ShenYu 中使用Cassoor

ShenYu有一个Casdoor插件，可以启用Casdoor的使用。

步骤1：部署Casdoor

首先，应部署Casdoor。您可以参考官方Casdoor文档的[服务器安装](#)。

成功部署后，请确保：

- Casdoor服务器正在<http://localhost:8000>上运行。
- 打开您喜欢的浏览器并访问<http://localhost:7001>以查看Casdoor登录页面。
- 通过输入 `admin` 和 `123`，登录功能运行正常。

按照上述步骤，您可以通过以下步骤在您的应用中快速实现基于Casdoor的登录页面。


步骤2：配置Casdoor应用

1. 创建一个新的Casdoor应用或使用现有的一个
2. 添加您的重定向URL

Name: → application name

Display name:

Logo:

Preview: 

Home:

Description:

Organization: → organization name

Client ID: → client id

Client secret: → client secret

Cert:

Redirect URLs:

Redirect URL	Action
<input type="text" value="http://localhost:9195/http/hi"/>	<input type="button" value="Add"/>
<input type="text" value="http://localhost:9195/http/hello"/>	<input type="button" value="Remove"/>

→ redirect url

3. 在证书编辑页面，您可以查看您的证书

Certificate:

```

-----BEGIN CERTIFICATE-----
MIIE+TCCAuGgAwIBAgIDAeJAMA0GCSqGSIb3DQEBCwUAMDYxHTAbBgNVBAoTFENh
c2Rvb3IgT3JnYW5pemF0aW9uMRUwEwYDVQQDEwxDYXNkb29yIENlcnQwHhcNMjEx
MDE1MDgxMTUyWhcNNDE1MDgxMTUyWjA2MR0wGwYDVQQKEwRDXNkb29yIyE9y
Z2FuaXphdGlvbEVVBMGA1UEAxMMQ2FzZG9vciBDZXJ0MIIICjANBgkqhkiG9w0B
AQEFAAOCAg8AMIICGKCAgEASlnpb5E1/yM0f1RfSDSSE8IR7y+lw+Rjji74e5ej
rq4b8zMYk7HeHCyZr/hmNEwEVXnhXu1P0mBeQ5ypp/QGo8vgeMjAETNmzkl1NjOQ
CjCYwUrasO/f/Mnl1COj13vx6mV1kHZjSrKsMhYY1vaxTEP3+VB8Hjg3MHFWrb07
uvFMCJe5W8+0rErZCKTR8+9VB3janeBz//zQePFVh79bFZate/hLirPK0Go9P1g
OvwloC1A3sarHTP4Qm/LQRt0rHqZfYbdySpyWAQvhNaDFE7mTstRSBb/wUjNCUBD
PTSLVjC04WllSf6Nkfx0Z7KvmbPstSj+btvcqsvRAGtvsB9h62Kptjs1Yn7GAuo
I3qt/4zoKbiURYxkQXlVwCQsEftUuk5ew5zuPSIDRLoLByQTLbx0JqLAFNfW3g/
pzSDjgd/60d6HTmVbZni4SmjdyFhXCdb1Kn7N+xTojfaNkwep2REV+RMc0fx4Gu
hRsnLsmkmUDeylZ9aBL9oj11YEQfM2JZEq+RVtUx+wB4y8K/tD1bcY+IfnG5rBpw
IDpS262boq4SRsVb3Z7bB0w4ZxvOfj/1VLorftjPbLif0bhfr/AeZMHplK0Xvzf4
yE+hqzi68wdf0VR9xYc/RbSAf7323OsjYnjEginUtRohnRgCpjlk/Mt2Kt84Kb0
wn8CAwEAAAMQMA4wDAYDVR0TAQH/BAIwADANBgkqhkiG9w0BAQsFAAOCAgEAn2lf
DKkLX+F1vKRO/5gJ+Plr8P5NKuQkmwH97b8CS2gS1phDyNglc4/LSdzuf4Awe6ve
C06lVdWSiis8UPUPdjmT2uMPSNjwLxG3QsrMURNwFLtFRem/he0Zgur9J1M
8haawdSdjh2RgmFoDeE2r8NVRfhr8KnCO1ddTJKuS1N0/irHz21W4jt4rxzCvl
2nR42Fybap3O/g2JXMhNNR0wZmNjggsF7XVENCsUfO1jTywLaqjuXCg54IL7XVLG
omKNNNcc8h1FCeKj/nnbGMhodnFWKDTsJcbNmOPNH06ixzqMy/Hqc+mWYv7maAG
Jtevs3qgMZ8F9Qzr3HpUc6R3ZYWWDY/xxPisuKftOPZgtH979XC4mdf0WPnOBLqL
2DJ1za8mjiGJolvb7XNVKcUfDXyW85ZTZQ5b9cl4e+6bmyWqQitlwt+Ati/uFEV
XzCj70B4lALX6xau1kLEpV9O1GERizYrZ5P9NjNA7KoO5AVMp9w0DQTKt+LbXnZE
HHnWky8xHQKZF9sR7YBPGLs/Ac6tviv5Ua15OgJ/8dLRZ/veyFfGo2yZsl+hKVU5
nCCJHBCAyFnm1hdvvdEdH33jDBjNB6ciotZrf/3VYalWSalADosHAgMWfXuWP+h
8XXmzlxuHbTMQYtZPDgspS5aK+S4Q9wb8RRAYo=
-----END CERTIFICATE-----

```

步骤3：在ShenYu中使用Casdoor插件

1. 在ShenYu中配置Casdoor插件

Plugin ✕

* Plugin:

casdoor Configuration

* application-name:

* certificate:

* client_id:

* client_secret:

* casdoor endpoint:

* organization-name:

* Role:

* Sort:

Status:


注意：由于ShenYu只有一个单行输入框，每行证书都必须添加\n。

Certificate ⓘ :

Copy certificate

Download certificate

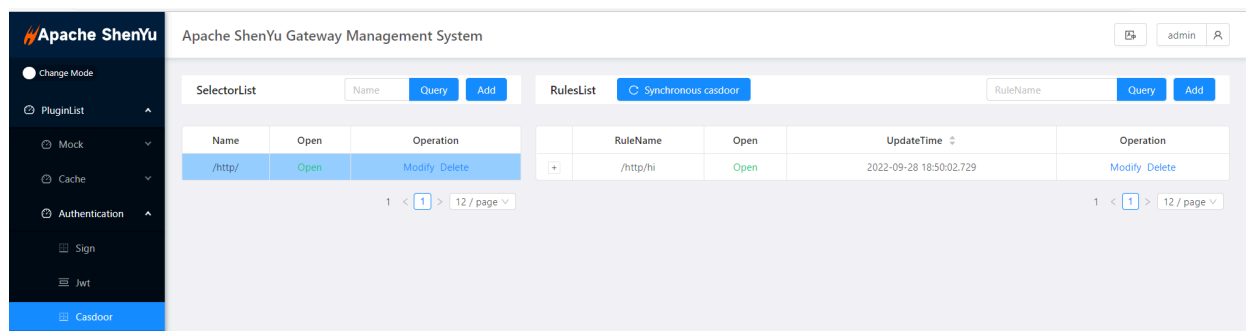
```
-----BEGIN CERTIFICATE-----\nMIIE+TCCAuGgAwIBAgIDAeJAMA0GCSqGSIb3DQEBCwUAMDYxHTAbBgNVBAoTFENh\n c2Rvb3Igt3JnYW5pemF0aW9uMRUwEwYDVQQDEwxDYXNkb29yIENlcnQwHhcNMjEx\n MDE1MDgxMTUyWhcNNDExMDE1MDgxMTUyWjA2MR0wGwYDVQQKExRDYXNkb29yIE9y\n Z2FuaXphdGlvbjEVMBMGA1UEAxMMQ2FzZG9vciBDZXJ0MIIlClJANBgkqhkiG9w0B\n AQEFAAOCAg8AMIICGgKCAgEAsInpb5E1/ym0f1RfSDSSE8IR7y+lw+RjI74e5ej\n rq4b8zMYk7HeHCyZr/hmNEwEVXnhXu1P0mBeQ5ypp/QGo8vgEmjAETNmzkl1NjOQ\n CjCYwUrasO/f/Mnl1C0j13vx6mV1kHZjSrKsMhYY1vaxTEP3+VB8Hjg3MHFWrb07\n uvFMCJe5W8+0rKERZCKTR8+9VB3janeBz//zQePFVh79bFZate/hLirPK0Go9P1g\n OvwoC1A3sarHTP4Qm/LQRt0rHqZFYbdySpyWAQvhNaDFE7mTstRSBb/wUjNCUBD\n PTLVJc04WIISf6Nkfx0Z7KvmbPstSj+btvcqsvRAGtvdsB9h62Kptjs1Yn7GAuo\n l3qt/4zoKbiURYxkQJXlvwCQsEftUuk5ew5zuPSIDRLoLByQTLbx0JqLAFNfW3g\n pzSDjgd/60d6HTmbvZni4SmjdyFhXCdb1Kn7N+xTojnfANkewp2REV+RMc0fx4Gu\n hRsnLsmkmUDeylZ9a8L9oj11YEQfM2JZEq+RVtUx+wB4y8K/tD1bcY+IfnG5rBpw\n lDpS262boq4SRsvb3Z7bB0w4ZxvOfJ/1VLoRftjPbLlF0bhfr/AeZMHplK0Xvfz4\n yE+hqzi68wdF0VR9xYc/RbSAf7323OsjYnjEgInUtRohnRgCpjIk/Mt2Kt84Kb0\n wn8CAwEAAMQMA4wDAYDVR0TAQH/BAlwADANBgkqhkiG9w0BAQsFAAOCAgEAn2If\n DKkLX+F1vKRO/5gJ+Plr8P5NKuQkmwH97b8CS2gS1phDyNglc4/LSdzuf4Awe6ve\n C06lVdWslis8UPUPdjmT2uMPSNjwLxG3QsrimMURNwFILtRem/heJe0Zgur9J1M\n 8haawdSdJjH2RgmFoDeE2r8NVRfRhbR8KnCO1ddTJKuS1N0/irHz21W4jt4rxzCv\n 2nR42Fybp3O/g2JXMhNNROwZmNjgpsF7XVENCuFO1jTywLaqjuXCg54IL7XVLG\n omKNNNcc8h1FCeKj/nnbGMhodnFWKDTsJcbNmcOPNH06ixzqMy/Hqc+mWYv7maAG\n Jtevs3qgMZ8F9Qzr3HpUc6R3ZYWDY/xxPisuKftOPZgtH979XC4mdf0WPnOBLqL\n 2DJ1zaBmjGJolvb7XNVKcUfDXyw85ZTzQ5b9cll4e+6bmyWqQltlwt+Ati/uFEV\n Xzcj70B4lALX6xau1kLEpV9O1GERizYRz5P9NJNA7KoO5AVMp9w0DQTKt+LbXnZE\n HHnWKy8xHQKZF9sR7YBPGls/Ac6tviv5Ua15OgJ/8dLRZ/veyFfGo2yZsl+hKVU5\n nCCJHbcAyFnm1hdvvdEdH33jDBjNB6ciotJzrf/3VYalWSalADosHAgMWFxuWP+h\n 8XKXmzlXuHbTMQYtZPDgspS5aK+S4Q9wb8RRAYo=\n-----END CERTIFICATE-----
```

 here not need add \n

您可以复制它并粘贴到ShenYu Casdoor配置的证书中。

您不需要在Casdoor证书编辑页面保存它，因为它只是用于复制。

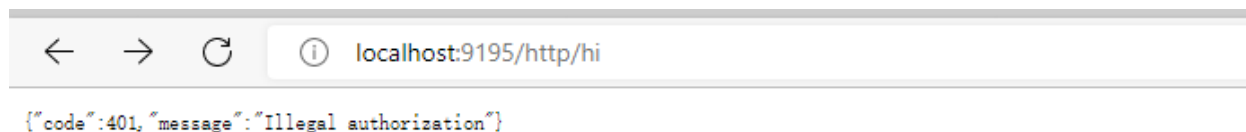
2. 配置ShenYu Casdoor插件



您可以配置Casdoor配置所需的内容。

3. 获取服务并使用它

3.1 直接访问Web



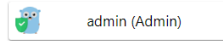
3.2 使用Casdoor登录

localhost:7001/login/oauth/authorize?client_id=6e3a84154e73d1fb156a&response_type=code&redirect_uri=http://localhost:9195/http/hi&scope=read&state=app-test

🔍 🌐



Continue with :



Or sign in with another account :

🔍 username, Email or phone

🔒 Password

Auto sign in [Forgot password?](#)

Sign In

[No account? sign up now](#)



localhost:9195/http/hi?code=822607b015cca2515b2b&state=app-test

hi! null! I'm Shenyu-Gateway System. Welcome!

3.3 在Headers中携带令牌

The screenshot shows the Postman interface for a GET request to `http://localhost:9195/http/hi`. The **Headers** tab is selected, displaying the following headers:

Key	Value	Description
<input checked="" type="checkbox"/> Postman-Token	<calculated when request is sent>	
<input checked="" type="checkbox"/> Host	<calculated when request is sent>	
<input checked="" type="checkbox"/> User-Agent	PostmanRuntime/7.29.2	
<input checked="" type="checkbox"/> Accept	*/*	
<input checked="" type="checkbox"/> Accept-Encoding	gzip, deflate, br	
<input checked="" type="checkbox"/> Connection	keep-alive	
<input checked="" type="checkbox"/> Authorization	eyJhbGciOiJIJSUz1NiIsImtpZCI6ImNlcnQtYn...	Your token

The response body is shown in the **Body** tab, displaying the following JSON:

```
1  { "hi": null, "I'm Shenyu-Gateway System. Welcome!" }
```

3.4 在Headers中保存名称、ID和组织

这使得将来使用它们更加方便。

ShardingSphere

[shardingsphere-elasticjob-ui](#)已经集成了Casdoor。配置后即可使用。

步骤1：部署Casdoor

首先，应部署Casdoor。

您可以参考Casdoor 官方文档 [Server Installation](#) 。

成功部署后，请确保：

- Casdoor服务器已成功运行在<http://localhost:8000>。
- 打开你最喜欢的浏览器并访问<http://localhost:7001>。你将看到Casdoor的登录页面。
- 输入 `admin` 和 `123` 来测试登录功能是否正常。

然后，你可以按照以下步骤在你自己的应用中快速实现基于Casdoor的登录页面。


步骤2：配置Casdoor应用并在ShardingSphere中配置应用

1. 创建或使用现有的Casdoor应用

Name: ShardingSphere

Display name: ShardingSphere

Logo: URL: https://cdn.casbin.org/img/casdoor-logo_1185x256.png

Preview: 

Home: /

Description:

Organization: ShardingSphere

Client ID: 3ed79fa530645fbd3653

Client secret: 54633c82b7796a4332c6976864c6c16bc3b05556

Cert: cert-built-in

Redirect URLs:

Redirect URL	Action
http://localhost:8080	↑ ↓ ✕

RedirectURLs取决于你需要重定向到的URL。选中的数据将在下一步中使用。

2. 在证书编辑页面，你可以看到你的 **Certificate**

Certificate ⓘ :

Copy certificate

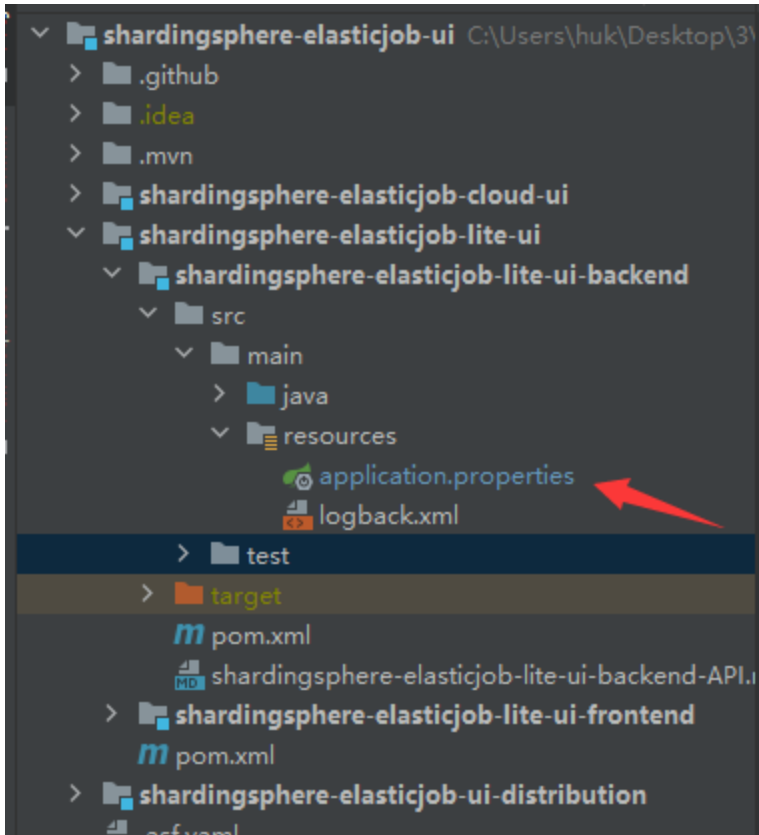
Download certificate

Private

```
-----BEGIN CERTIFICATE-----
MIIE+TCCAuGgAwIBAgIDAeJAMA0GCSqGSIb3DQEBCwUAMDYxHTAb8gNVBAoTFENh
c2Rvb3lgT3JnYW5pemF0aW9uMRUwEwYDVQQDEwxDYXNkb29yIENlcnQwHhcNMjEx
MDE1MDgxMTUyWhcNNDExMDE1MDgxMTUyWjA2MR0wGwYDVQQKEwRDXNkb29yIE9y
Z2FuaXphdGlvbjEVMjE1UEAxMMQ2FzZG9vciBDZXJ0MIIICjANBgkqhkiG9w0B
AQEFAAOCAg8AMIICGKCAgEAsInpb5E1/ym0f1RfSDSSE8IR7y+lw+RjI74e5ej
rq4b8zMYk7HeHCyZr/hmNEwEVXnhXu1P0mBeQ5ypp/QGo8vgEmjAETNmzkI1NjOQ
CjCYwUrasO/f/MnI1C0j13vx6mV1kHZjSrKsMhYY1vaxTEP3+VB8Hjg3MHFWrb07
uvfMCJe5W8+0rKErZCKTR8+9VB3janeBz//zQePFVh79bFZate/hLirPK0Go9P1g
OvwloC1A3sarHTP4Qm/LQRt0rHqZFybdySpyWAQvhNaDFE7mTstRSBb/wUjNCUBD
PTSLVjC04WII5f6Nkfx0Z7KvmbPstSj+btvcqsvRAGtvsB9h62Kptjs1Yn7GAuo
l3qt/4zoKbiURYxkQJXlvwCQsEftUuk5ew5zuPSIDRLoLByQTLbx0JqLAFNfW3g/
pzSDjgd/60d6HTmVbZni4SmjdyFhXCdb1Kn7N+xTojnfaNkwep2REV+RMc0fx4Gu
hRsnLsmkmUDeylZ9aBL9oj11YEQfM2JZEQ+RVtUx+wB4y8K/tD1bcY+IfnG5rBpw
IDpS262boq4SRsVb3Z7b80w4ZxvOfJ/1VLoRftjPbLif0bhfr/AeZMhplKOXvfz4
yE+hqzi68wdf0VR9xYc/RbSAf7323OsjYnjEgInUtRohnRgCpjlk/Mt2Kt84Kb0
wn8CAwEAAAMQMA4wDAYDVR0TAQH/BAIwADANBgkqhkiG9w0BAQsFAAOCAgEAn2If
DKkLX+F1vKRO/5gJ+Plr8P5NKuQkmwH97b8CS2gS1phDyNgIc4/LSdzuf4Awe6ve
C06IVdWslis8UPUPdjmT2uMPSNjwLxG3QsrimMURNwFILtRem/heJe0Zgur9J1M
8haawdSdJjH2RgmFoDeE2r8NVRfhr8R8KnCO1ddTJKuS1N0/irHz21W4jt4rxzCvl
2nR42Fybap3O/g2JXMhNNROwZmNjggsF7XVENCsuFO1jTywLaquXCg54IL7XVLG
omKNNncc8h1FCeKj/nnbGMhodnFWKDTsJcbNmcOPNHo6ixzqMy/Hqc+mWYv7maAG
Jtevs3qgMZ8F9Qzr3HpUc6R3ZYWDY/xxPisuKftOPZgtH979XC4mdf0WPnOBLqL
2DJ1zaBmjiGJolvb7XNVKcUfDXyW85ZTZQ5b9cll4e+6bmyWqQltwt+Ati/uFEV
XzCj70B4IALX6xau1kLEpV9O1GERizYRz5P9NjNA7KoO5AVMp9w0DQTKt+LbXnZE
HHnWky8xHQKZF9sR7YBPGLs/Ac6tviv5Ua15OgJ/8dLRZ/veyFfGo2yZsl+hKVU5
nCCJHBcAyFnm1hdvdwEdH33jdBjNB6ciotJZrf/3VYalWSalADosHAGMWFxUWP+h
8XKXmzlxuHbTMQYtZPDgspS5aK+S4Q9wb8RRAYo=
-----END CERTIFICATE-----
```

3. 在ShardingSphere中配置应用

首先，我们需要找到我们需要配置的application.properties。



接下来，我们需要从Casdoor应用中复制数据并粘贴到应用中。

```
casdoor.endpoint=http://localhost:7001
casdoor.client-id=3ed79fa530645fbd3653
casdoor.client-secret=54633c82b7796a4332c6976864c6c16bc3b05556
casdoor.certificate=\
----BEGIN CERTIFICATE-----\n\
MIIE+TCCAuGgAwIBAgIDAeJAMA0GCSqGSIb3DQEBCwUAMDYxHTAbBgNVBAoTFENh\n\
c2Rvb3Igt3JnYW5pemF0aW9uMRUwEwYDVQQDEwxDYXNkb29yIEJlcnQwHhcNMjEx\n\
MDE1MDgxMTUyWWhcNDE1MDgxMTUyWjA2MR0wGwYDVQQKExRDYXNkb29yIE9y\n\
Z2FuaXphdGlvbjEVMGMGA1UEAxMMQ2FzZG9vcjBDZXJ0MIICIjANBgkqhkiG9w0B\n\
AQEFAAOCBg8AMIICGKCAgEAsInpb5E1/ym0f1RfSDSSE8IR7y+Lw+RJJI74e5ej\n\
rq4b8zMYK7HeHcyZr/hmNEwEVXnhXu1P0mBeQ5ypp/QGo8vgEmjAETNmzkI1Nj0Q\n\
CjCYwUras0/f/MnI1C0j13vx6mV1kHZjSrKsMhY1vaxTEP3+VB8Hjg3MHFWrb07\n\
uvFMCJe5W8+0rKERZCKTR8+9VB3janeBz//zQePFVh79bFZate/hLirPK0Go9P1g\n\
0vwIoC1A3sarHTP4Qm/LQRt0rHqZFybdySpyWAQvhNaDFE7mTstRSBb/wUjNCUBD\n\
PTSLVjC04WllSf6Nkfx0Z7KvmbPstSj+btvcqsvRAGtvsB9h62Kptjs1Yn7GAuo\n\
I3qt/4zoKbiURYxkQJXIvwcQsEftUuk5ew5zuPSLDRLoLByQTLbx0JQLAFNFW3g\n\
pzSDjgd/60d6HTmVbZni4SmjdyFhXCdb1Kn7N+xTojnfANkwp2REV+RMc0fx4Gu\n\
hRsnLsmkmUdeyIZ9aBL9oj11YEQfM2JZEg+RVtUx+wB4y8K/tD1bcY+IfnG5rBpw\n\
IDpS262boq4SRsvb3Z7bB0w4Zxv0fJ/1VLoRftjPbLiF0bhfr/AeZMhpIKOXvfz4\n\
yE+hqzi68wdF0VR9xYc/RbSAf73230sjYnjEgInUtRohnRgCpjIk/Mt2Kt84Kb0\n\
wn8CAwEAAMQMA4wDAYDVR0TAQH/BAIwADANBgkqhkiG9w0BAQsFAAOCBgEAn2lf\n\
DKkLX+F1vKR0/5gJ+Plr8P5NKuQkmwH97b8CS2gS1phDyNgIc4/LSdzuf4Awe6ve\n\
C06LVdWsiS8UPUPdjmT2uMPSNjwLxG3QsrImMURNwFLLTfRem/heJe0Zgur9J1M\n\
aawdSdJjH2RgmFoDeE2r8NVRfhbR8KnC01ddTJKuS1N0/irHz21W4jt4rxzCvL\n\
2nR42FybaP30/g2JXmHnNR0wZmNjgqsF7XVENCsuF01jTywLagjuXCg54IL7XVL6\n\
omKNNcc8h1FCeKj/nnbGMhodnFWKDTsJcbNmc0PNHo6ixzqMy/Hqc+mWYv7maAG\n\
Jtevs3qgMZ8F9Qzr3HpUc6R3ZYWDY/xxPisuKft0PZgtH979XC4mdf0WPn0BLqL\n\
2DJ1zaBmji6Jo1vb7XNVKcUfDXYw85ZTZ05b9clI4e+6bmyWqQItlwt+Ati/uFEV\n\
XzCj70B4LALX6xau1kLEpV9016ERizYRz5P9NJNA7Ko05AVMp9w0DQTKt+LbXnZE\n\
HHnWKy8xHQKZF9sR7YBPGls/Ac6tviy5Ua150gJ/8dLRZ/veyFf6o2yZsI+hKVU5\n\
nCCJHBCAyFnm1hdvvdwEdH33jDBjNB6ciotJZrf/3VYaIWSaLAdosHAgMwfXuWP+h\n\
8XKXmzlxuHbTMQYtZPDgspS5aK+S4Q9wb8RRAYo=\n\
----END CERTIFICATE-----
casdoor.organization-name=ShardingSphere
casdoor.application-name=ShardingSphere
```

4. 测试一下



Login

Login with Casdoor

Apache IoTDB

Casdoor可以轻松连接到[Apache IoTDB](#)。

连接Casdoor的代码已经添加到[Apache IoTDB Web Workbench](#)中，所以我们只需要在后端配置application.yml文件并激活前端开关。

步骤1：部署Casdoor

首先，部署Casdoor。

您可以参考官方Casdoor文档中的[服务器安装](#)。

成功部署后，请确保：

- Casdoor服务器在<http://localhost:8000>成功运行。
- 打开您喜欢的浏览器并访问<http://localhost:7001>，您将看到Casdoor登录页面。
- 通过输入 `admin` 和 `123` 测试登录功能。


完成这些步骤后，您现在可以在您自己的应用中快速实现基于Casdoor的登录页面。

步骤2：配置Casdoor

要配置Casdoor，请参考[casdoor](#)（建议不要在您正在开发的浏览器中使用Casdoor的浏览器）。

您还应该创建一个组织和一个应用。参考[casdoor](#)进行操作。

2.1 创建一个组织

Name	IoTDB
Display name	IoTDB
Favicon	URL: https://cdn.casbin.org/img/favicon.png
Preview:	
Website URL	https://door.casdoor.com
Password type	plain
Password salt	

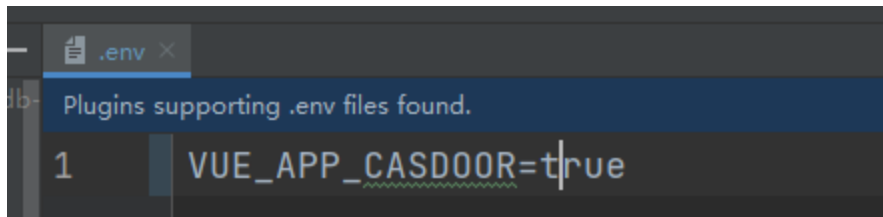
2.2 创建一个应用

Name	app_IoTDB
Display name	app_IoTDB
Logo	URL: https://cdn.casbin.org/img/casdoor-logo_1185x256.png
Preview:	
Home	
Description	
Organization	built-in
Client ID	6f561b83d119be3e1e3c
Client secret	242082e823b31a9b0d3a0a4a5a80cd5e415c75f7
Cert	cert-built-in

步骤3：激活Apache IoTDB Web Workbench前端开关

打开此开关以将代码和状态发送到后端。

此开关可以在`iotdb-web-workbench/fronted/.env`文件中找到。



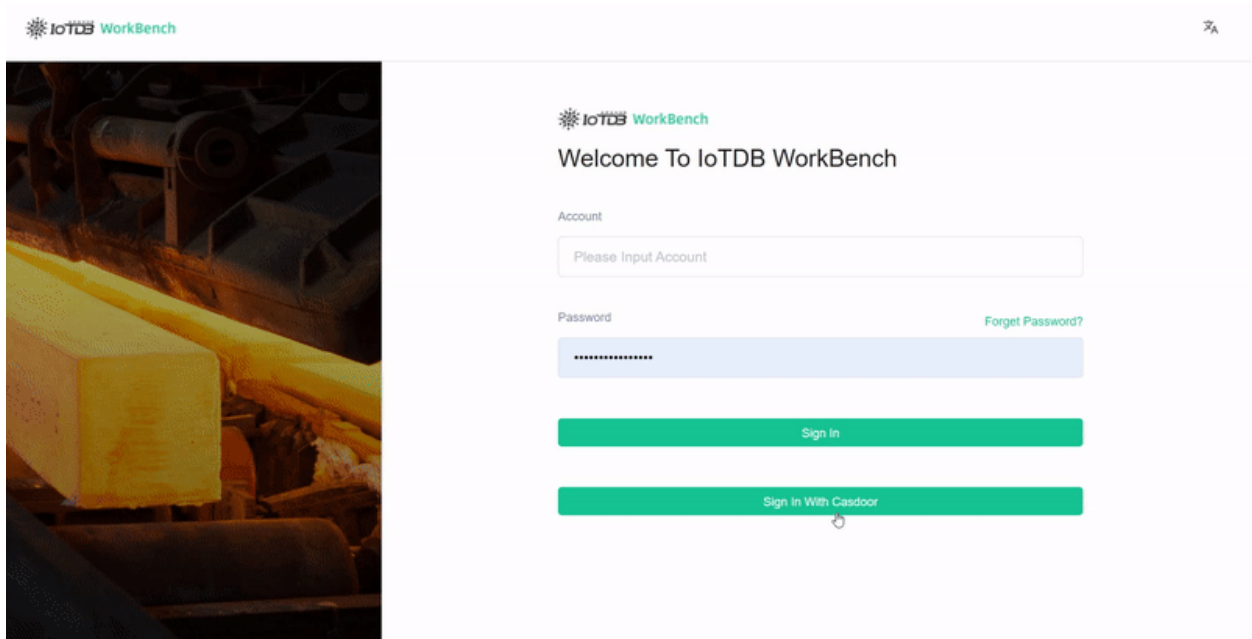
```
.env x
Plugins supporting .env files found.
1 VUE_APP_CASDOOR=true
```

步骤4：配置后端代码

您需要在 `iotdb-web-workbench/backend/src/main/resources/application.properties` 文件中配置 Casdoor 的设置。

```
casdoor.endpoint = http://localhost:8000
casdoor.clientId = <client id from previous step>
casdoor.clientSecret = <client secret from previous step>
casdoor.certificate=<client certificate from previous step>
casdoor.organizationName=IoTDB
casdoor.applicationName=app-IoTDB
```

結果



Apache DolphinScheduler

Casdoor是[Apache DolphinScheduler](#)支持的登录方式之一。

步骤1：部署Casdoor

首先，应部署Casdoor。您可以参考Casdoor官方文档的[服务器安装](#)。

成功部署后，请确保：

- Casdoor服务器在<http://localhost:8000>成功运行。
- 打开您喜欢的浏览器并访问<http://localhost:7001>。您将看到Casdoor的登录页面。
- 通过输入"admin"和"123"来测试登录功能。

部署完成后，您可以按照以下步骤在自己的应用中快速实现基于Casdoor的登录页面。

步骤2：配置Casdoor应用

1. 创建一个新的Casdoor应用或使用现有的一个。
2. 添加您的重定向URL（您可以在下一节中找到更多关于如何获取重定向URL的详细信息）。

The image shows a web form for configuring a Casdoor application. The form includes the following fields and values:

- Name:** application_a6ftas (highlighted with a red box and arrow pointing to the text "your application name")
- Display name:** New Application - a6ftas
- Logo URL:** https://cdn.casbin.org/img/casdoor-logo_1185x256.png
- Preview:** Casdoor logo (a blue cube icon and the word "Casdoor" with a star over the 'o')
- Home:** (empty)
- Description:** (empty)
- Organization:** organization_carg1b (highlighted with a red box and arrow pointing to the text "your organization name")
- Client ID:** 3ed7314825ecf955cb19 (highlighted with a red box and arrow pointing to the text "your client id")
- Client secret:** ee9314ea228 (highlighted with a red box and arrow pointing to the text "your client secret")
- Cert:** cert-built-in
- Redirect URLs:** A table with one entry: http://localhost:3000/callback (highlighted with a red box and arrow pointing to the text "your redirect url")

3. 添加所需的提供商并填写其他必要的设置。

在应用设置页面上，您将找到两个重要的值：`Client ID`和`Client secret`，如上图所示。我们将在下一步中使用这些值。

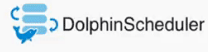
打开您喜欢的浏览器并访问http://CASDOOR_HOSTNAME/.well-known/openid-configuration以查看Casdoor的OIDC配置。

步骤3：配置DolphinScheduler

dolphinscheduler-api/src/main/resources/application.yaml

```
security:
  authentication:
    # Authentication types (supported types: PASSWORD, LDAP,
    CASDOOR_SSO)
    type: CASDOOR_SSO
  casdoor:
    # The URL of your Casdoor server
    endpoint:
    client-id:
    client-secret:
    # The certificate may be multi-line; you can use `|-` for ease
    certificate:
    # The organization name you added in Casdoor
    organization-name:
    # The application name you added in Casdoor
    application-name:
    # The DolphinScheduler login URL
    redirect-url: http://localhost:5173/login
```

现在，DolphinScheduler将自动将您重定向到Casdoor进行身份验证。



SSO Login

FireZone

Casdoor可以使用OIDC协议作为IDP连接各种应用。在这里，我们将使用FireZone作为例子，向您展示如何使用OIDC连接到您的应用。

步骤1：部署Casdoor和FireZone

首先，应部署Casdoor和FireZone。

成功部署后，请确保以下内容：

1. 将FireZone URL（Signin → Security → Add OpenID Connect Provider）设置为 FIREZONE_HOSTNAME。

Firezone

CONFIGURATION

- Users
- Devices
- Rules

SETTINGS

- Defaults**
- Account
- Customization
- Security

DIAGNOSTICS

- WAN Connectivity

Site Settings

Configure default WireGuard settings for this site.

Site Defaults

Allowed IPs

172.21.0.0/16,172.16.0.0/16

Configures the default AllowedIPs setting for devices. AllowedIPs determines which destination IPs get routed through Firezone. Specify a comma-separated list of IPs or CIDRs here to achieve spl

DNS Servers

172.16.250.155

Comma-separated list of DNS servers to use for devices. Leaving this blank will omit the `DNS` section in generated device configs.

Endpoint

localhost:8080

IPv4 or IPv6 address that devices will be configured to connect to. Defaults to this server's public IP if not set.

Persistent Keepalive

0

Interval in seconds to send persistent keepalive packets. Most users won't need to change this. Set to 0 or leave blank to disable. Leave this blank if you're unsure what this means.

MTU

1280


WireGuard interface MTU for devices. Defaults to what's set in the configuration file. Leave this blank if you're unsure what this means.


2. Casdoor可以正常登录和使用。
3. `CASD00R_HOSTNAME`: <http://localhost:8000>, 如果您使用默认的 `app.conf` 部署 Casdoor。


步骤2：配置Casdoor应用


1. 创建一个新的Casdoor应用或使用现有的一个。
2. 添加一个重定向URL：


例如，如果FireZone Provider中的Configid是TEST，那么重定向URL应该是 `http://[FIREZONE_HOST]/auth/oidc/[PROVIDER_CONFIG_ID]/callback/`。


Home :


Description :

Organization :

Client ID :

Client secret :

Cert :

Redirect URLs :

Redirect URLs	Add
Redirect URI	
<input type="text" value="http://localhost:8080/auth/oidc/TEST/callback/"/>	

打开您喜欢的浏览器并访问: `http://[CASD00R_HOSTNAME]/.well-known/openid-configuration`, 您将看到Casdoor的OIDC配置。

3. 配置FireZone: Security → Add OpenID Connect Provider

OIDC Config ✕

Config ID

Label

Scope

Response type

Client ID

Client secret

Discovery Document URI

Auto create users

Save

ConfigID should be the PROVIDER_CONFIG_ID of the redirect URL

- **Discovery Document URI**: FireZone Provider Discovery Document URI应该是 `https://[CASD00R_HOST]/.well-known/openid-configuration`。
- **Scopes**: `openid email profile`
- **ConfigID**: ConfigID应该是重定向URL的PROVIDER_COONFIG_ID, 并且应该对应Casdoor重定向URL。

- `Auto-create users`: 成功登录将自动创建用户。

退出FireZone并测试SSO



Sign In

Please sign in via one of the methods below.

Sign in with TEST

Sign in with email

Cloud Foundry

在集成之前，我们需要在本地部署Casdoor。

然后，我们可以按照以下步骤在我们自己的应用中快速实现基于Casdoor的登录页面。

步骤1：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URL：`http://CASD00R_HOSTNAME/login`



3. 复制客户端ID；我们将在接下来的步骤中需要它。

步骤2：在Casdoor中添加用户

现在你有了应用程序，但没有用户，你需要创建一个用户并分配角色。

转到“用户”页面，然后点击右上角的“添加用户”。这将打开一个新页面，您可以在其中添加新用户。

在添加用户名和组织“Cloud Foundry”后保存用户（其他详细信息是可选的）。

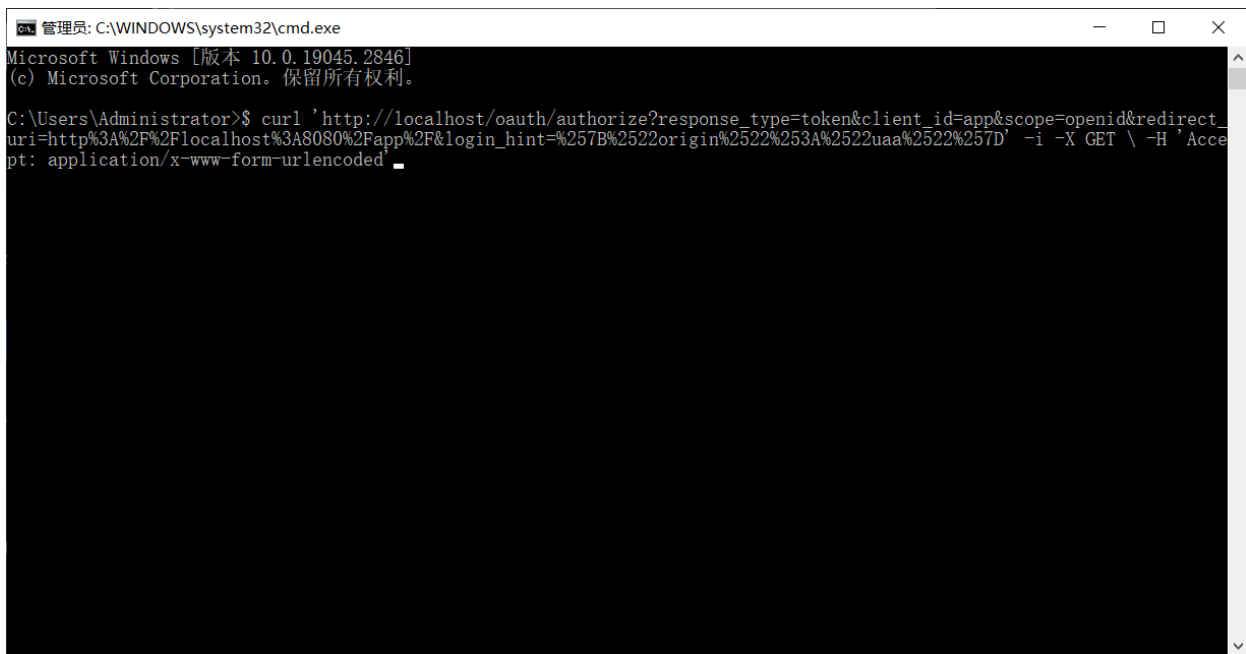
现在，你需要为你的用户设置一个密码，你可以通过点击“管理你的密码”来做到这一点。

为您的用户选择一个密码并确认它。

步骤3：构建Cloud Foundry应用

按照这些步骤启动Cloud Foundry。

- `$ git clone git://github.com/cloudfoundry/uaa.git`
- `$ cd uaa`
- `$./gradlew run`



```
管理员: C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.19045.2846]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Administrator>$ curl 'http://localhost/oauth/authorize?response_type=token&client_id=app&scope=openid&redirect_uri=http%3A%2F%2Flocalhost%3A8080%2Fapp%2F&login_hint=%257B%2522origin%2522%253A%2522uaa%2522%257D' -i -X GET \ -H 'Accept: application/x-www-form-urlencoded'
```

步骤4: 集成Casdoor

现在打开另一个命令行并输入:

```
curl '<http://localhost/oauth/authorize?response_type=token&client_id=app&scope=openid&redirect_uri=http%3A%2F%2Flocalhost%3A8080%2Fapp%2F>' -i -X GET \ -H 'Accept: application/x-www-form-urlencoded'
```

我们已经获得了客户端ID和重定向URI; 我们输入这些参数。

Parameter	Type	Constraints	Description
response_type	String	Required	Space-delimited list of response types. Here, token, i.e. an access token
client_id	String	Required	a unique string representing the registration information provided by the client
scope	String	Optional	requested scopes, space-delimited
redirect_uri	String	Optional	redirection URI to which the authorization server will send the user-agent back once access is granted (or denied), optional if pre-registered by the client

执行命令，我们可以得到下面的结果，这意味着我们已经成功地将Casdoor与Cloud Foundry集成。


```

HTTP/1.1 302 Found
Content-Security-Policy: script-src 'self'
Strict-Transport-Security: max-age=31536000
Set-Cookie: X-Uaa-Csrft=09mMqMDhcwHGLMufnB4YA1; Path=/; Max-Age=86400; Expires=Fri, 5 May 2023 14:53:54 GMT; HttpOnly; SameSite=Lax
Cache-Control: no-store
Content-Language: en
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Location: http://localhost:8080/app/#token_type=bearer&access_token=eyJhbGciOiJIUzI1NiIsImprdiI6Imh0dHBzOjI8vbG9jYXob3N00jgwODAvdWVhL3Rva

```

Built-in Organization x +

localhost:8000/login




username, Email, or phone

Password

Auto sign in [Forgot password?](#)

Sign In

[No account? sign up now](#)

Powered by  Casdoor

+

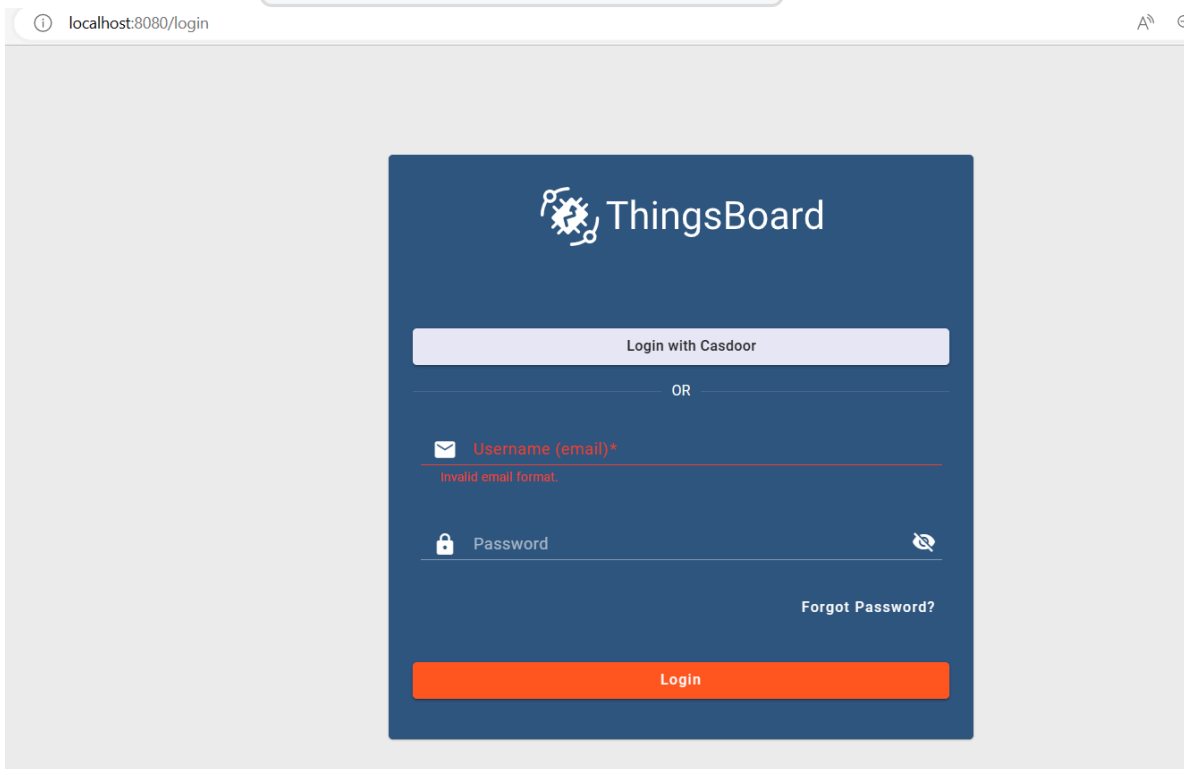
Thingsboard

在集成之前，我们需要在本地部署Casdoor。

然后，我们可以按照以下步骤在我们自己的应用中快速实现基于Casdoor的登录页面。

步骤1：配置Casdoor应用

1. 创建一个新的Casdoor应用或使用现有的应用。
2. 添加重定向URL: `http://CASD00R_HOSTNAME/login`



3. 复制客户端ID和客户端密钥。我们将在接下来的步骤中需要它们。

步骤2：在Casdoor中添加用户

现在你有了应用，你需要创建一个用户并分配一个角色。

转到“用户”页面，然后在右上角点击“添加用户”。这将打开一个新页面，您可以在其中添加新用户。

在添加用户名并选择组织“Thingsboard”后保存用户（其他详细信息可选）。

接下来，您需要为用户设置密码。您可以通过点击“管理您的密码”来做到这一点。

为用户选择一个密码并确认它。

步骤3：准备和构建Thingsboard应用

首先，Thingsboard只支持Java 11 (OpenJDK)。

您可以从以下链接下载：

[JDK下载页面](#)

要启动Thingsboard，请按照以下步骤操作（适用于Windows系统）：

- 下载并解压包。 [下载包](#)
- 根据您的偏好在`\thingsboard\conf\thingsboard.yml`中配置Thingsboard，包括Kafka、PostgreSQL等的配置。
- 在Thingsboard文件夹的命令行中运行 `install.bat -loadDemo` 以安装并添加演示数据。

```
C:\Program Files (x86)\thingsboard>install.bat --loadDemo
Detecting Java version installed.
CurrentVersion 110
Java 11 found!
Installing thingsboard ...
...
ThingsBoard installed successfully!
```

- 在命令行中运行 `net start thingsboard` 以启动 Thingsboard。您应该看到以下输出：

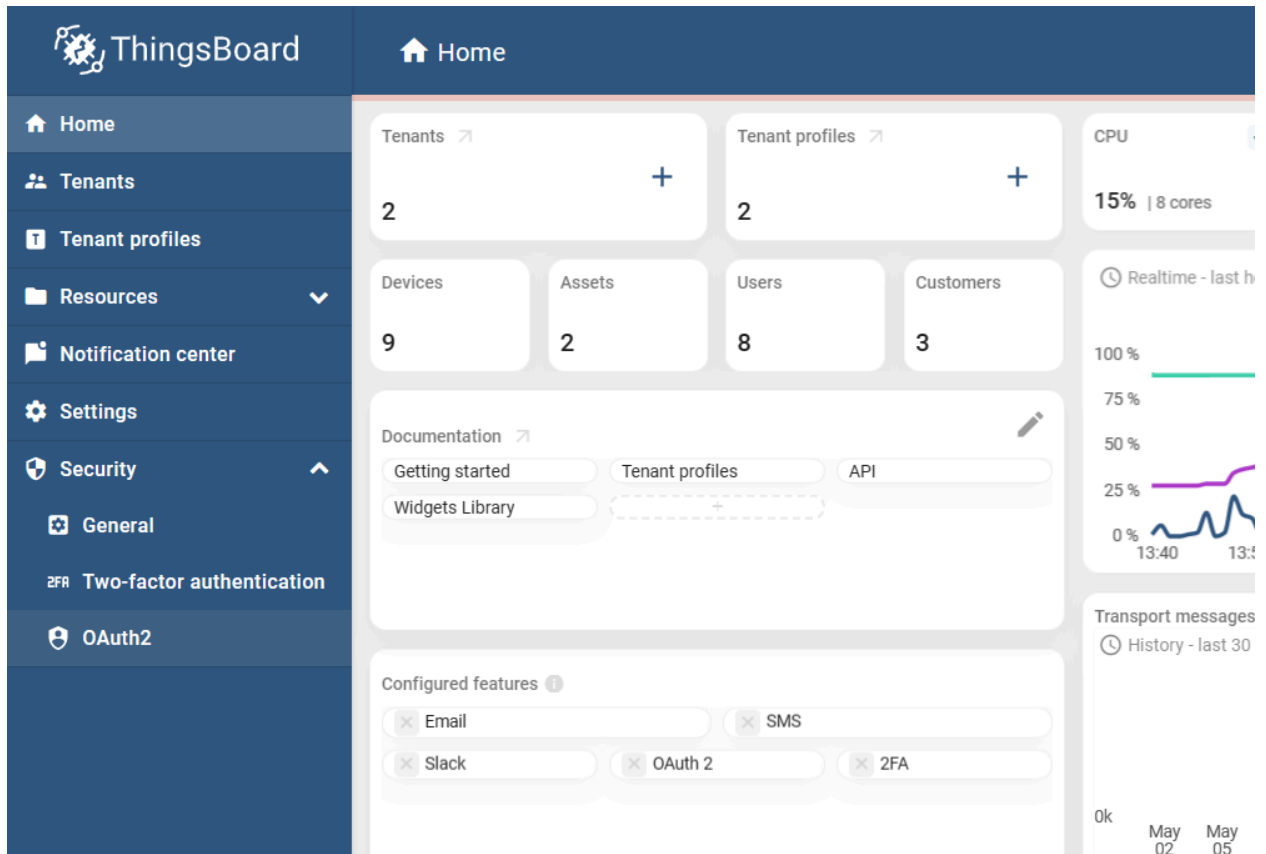
```
The ThingsBoard Server Application service is starting.
The ThingsBoard Server Application service was started successfully.
```

步骤4：集成 Casdoor

现在打开 <http://localhost:8080/> 并登录到管理员账户：



账户：sysadmin@thingsboard.org / 密码：sysadmin

成功登录后，点击页面左下角的 OAuth2 按钮。



按照以下方式填写空白：

Providers

Custom  

Login provider* Custom	Allowed platforms Web, Android, iOS
Client ID* e324f9a3f55e1adac4ef	Client secret* 28b3f98c1f55c1cc57f74b9b1a68b5d2e79

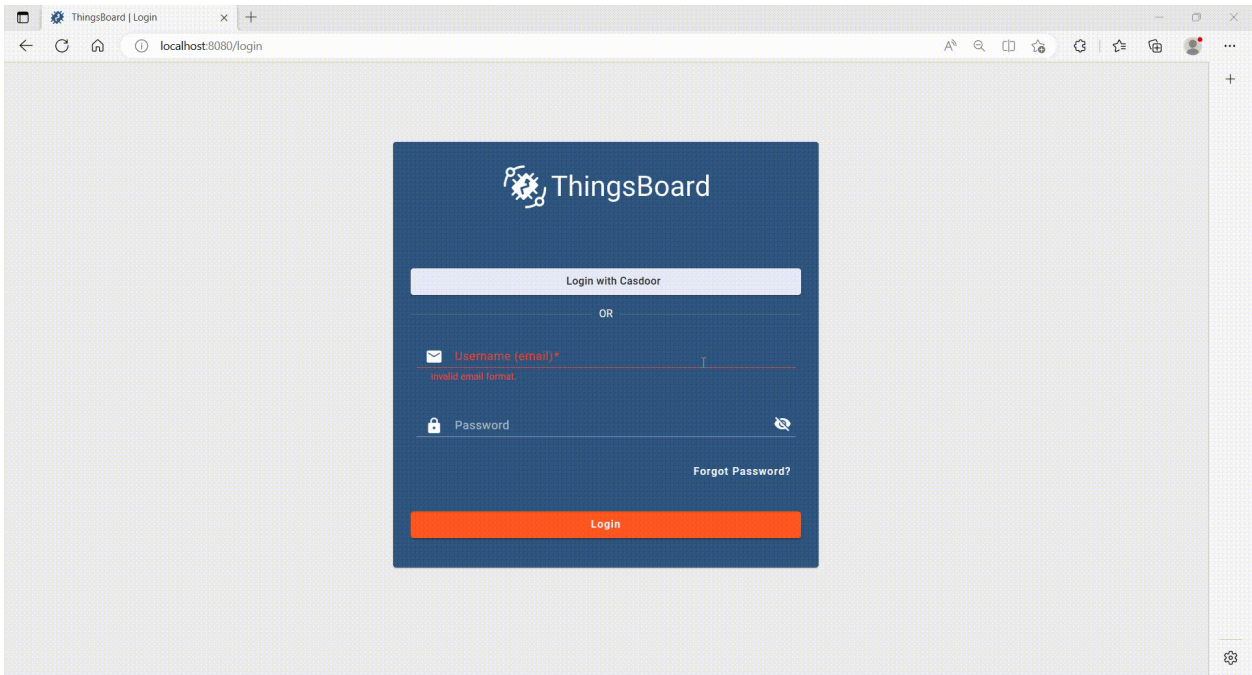
General	Mapper
Access token URI* http://localhost:8000/api/login/oauth/ac	Authorization URI* http://localhost:8000/login/oauth/authori
JSON Web Key URI http://localhost:8000/.well-known/jwks	User info URI http://localhost:8000/api/userinfo
Client authentication method* POST	
Provider label* Casdoor	Login button icon

Allow user creation

您可以从以下链接获取这些值: [OIDC discovery URL](#)

```
{
  "issuer": "https://door.casdoor.com",
  "authorization_endpoint": "https://door.casdoor.com/login/oauth/authori",
  "token_endpoint": "https://door.casdoor.com/api/login/oauth/access_toke",
  "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/int",
  "response_types_supported": [
```

填写这些空白后, 您已经成功将Casdoor与Thingsboard集成。当你登录到 <http://localhost:8080/>时, 你应该看到以下内容:



JavaScript

Firebase

使用Casdoor作为身份提供者的Firebase项目

微信小程序

在微信小程序中使用 Casdoor

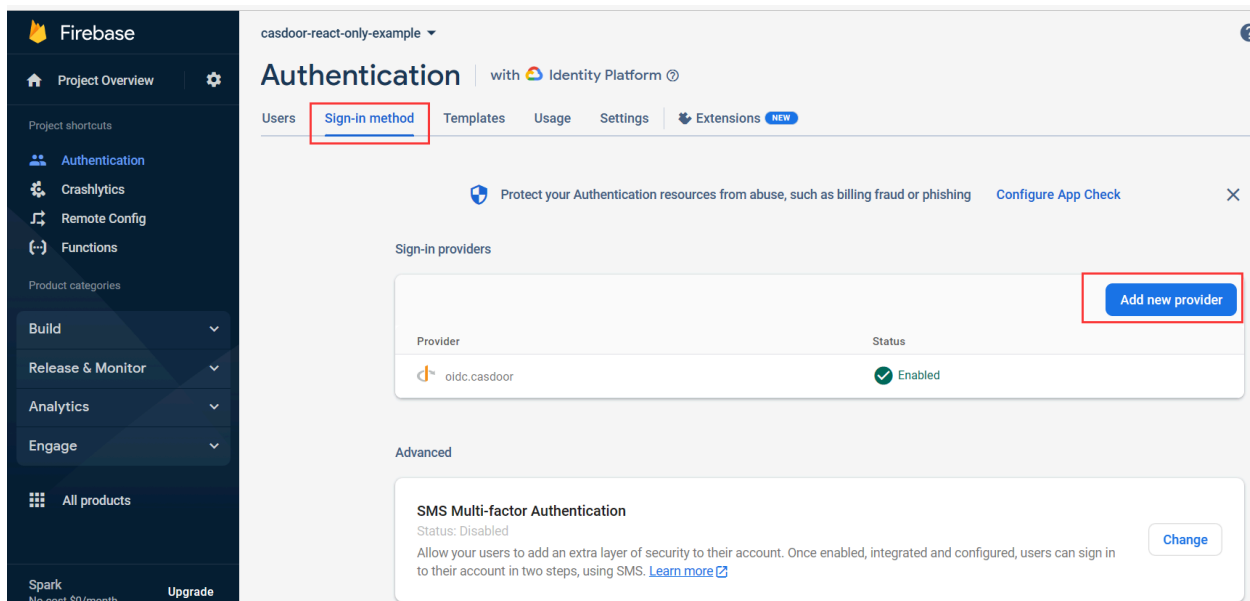
Firebase

Firebase支持OIDC作为身份提供者，您可以使用Casdoor作为Firebase web应用的OIDC提供者。

1. 创建一个Firebase项目

前往[Firebase控制台](#)创建一个项目。

1.1 添加Casdoor作为提供者



您需要首先启用“身份平台”功能，以在Firebase上启用OIDC集成。

在自定义提供者中选择 **OpenID Connect**，填写以下信息：

名称 (按顺序)	描述	示例值
名称	可以是您喜欢的任何字符串	casdoor
客户端ID	Casdoor应用的客户端ID	294b09fbc17f95daf2fe
发行者 (URL)	Casdoor服务器URL	https://door.casdoor.com
客户端 密钥	Casdoor应用的客户端密钥	dd8982f7046ccba1bbd7851d5c1ece4e52bf039d

door.casdoor.com applications/casbin/app-vue-python-example


Casdoor Home User Management Identity Authorization Logging & Auditing Business & Payments Admin

Edit Application Save Save & Exit

Name: app-vue-python-example

Display name: Casdoor Vue Python Example

Logo: URL: https://cdn.casbin.org/img/casdoor-logo_1185x256.png

Preview: 

Home: https://demo.gsoc.com.cn

Description:

Organization: casbin

Tags:

Client ID: 294b09fbc17f95daf2fe

Client secret: dd8982f7046ccba1bbd7851d5c1ece4e52bf039d

Cert: cert-built-in

上述示例值可以从Casdoor演示站点获取: [app-vue-python-example](#)


1 Define new OIDC provider

Grant type

 Code flow Implicit flow (id_token)

Name

casdoor

Provider ID: oidc.casdoor 

Client ID

294b09fbc17f95daf2fe


Issuer (URL)

https://door.casdoor.com

Client secret

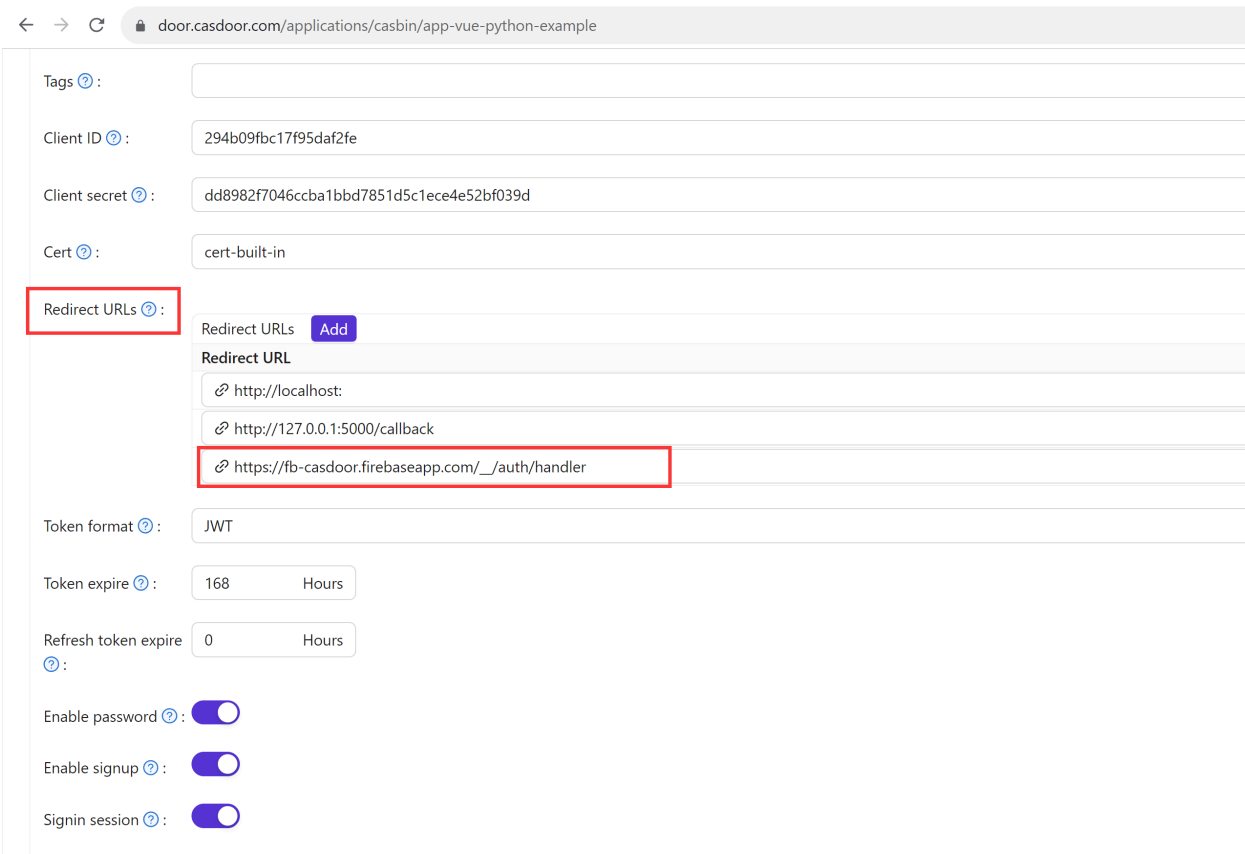
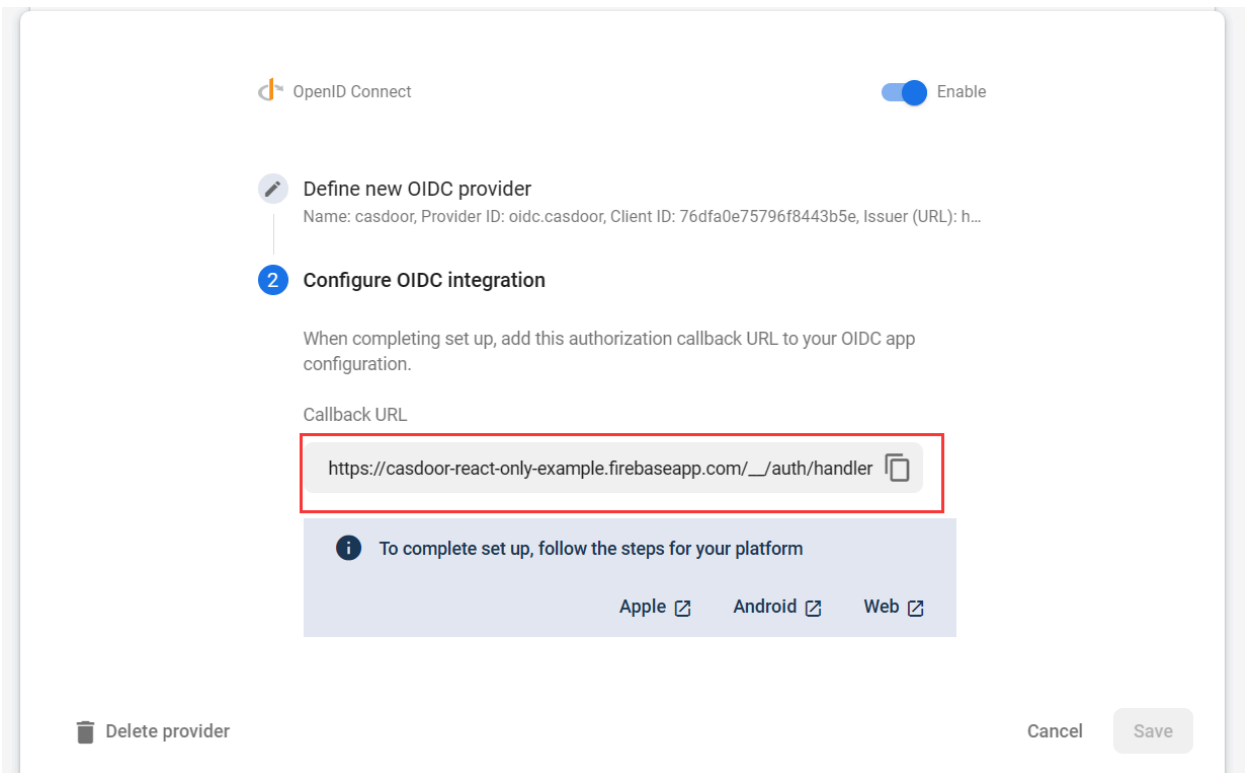
dd8982f7046ccba1bbd7851d5c1ece4e52bf039d

Next

 Configure OIDC integration

1.2 添加回调url

将回调URL添加到Casdoor应用的重定向URL:



在这里，我们提供一个示例[casdoor-firebase-example](#)，供您在应用中使用Casdoor身

份验证。要查看如何在应用中使用Casdoor身份验证的更多详细信息，请参阅[Firebase 文档](#)。

微信小程序

ⓘ 信息

从1.41.0版本开始，Casdoor现在支持微信小程序。

介绍

由于微信小程序不支持标准的OAuth，因此无法重定向到自托管的Casdoor登录页面。因此，使用Casdoor进行微信小程序的过程与常规程序的过程不同。

本文档将解释如何将Casdoor集成到微信小程序中。您可以在GitHub上找到这个集成的例子：[casdoor-wechat-miniprogram-example](#)。有关更详细的信息，请参阅[微信小程序登录文档](#)。

配置包括以下名称：

`CASD00R_HOSTNAME`：部署Casdoor服务器的域名或IP地址，例如，`https://door.casbin.com`。

步骤1：部署Casdoor

首先，应部署[Casdoor服务器](#)。

成功部署Casdoor后，您需要确保：

1. Casdoor可以正常访问和使用。
2. 将Casdoor的 `origin` 值 (conf/app.conf) 设置为 `CASD00R_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. 在Casdoor中创建一个微信IDP，并提供微信小程序开发平台给你的APPID和APPSECRET。

New Provider

Name [?]:

Display name [?]:

Category [?]:

Type [?]:

Client ID [?]:

Client secret [?]:

Provider URL [?]:

2. 创建一个新的Casdoor应用程序，或使用一个已经存在的。
3. 将在上一步中创建的IDP添加到您想要使用的应用程序中。

! TIPS

为了方便，Casdoor默认将应用中的第一个微信类型的IDP视为微信小程序的IDP。

因此，如果你想在这个应用中使用微信小程序，不要在一个应用中添加多个微信类型的IDP。

步骤3：编写微信小程序代码

微信小程序提供了一个API，用于内部登录并获取代码。代码应该然后被发送到Casdoor。Casdoor将使用此代码从微信服务器检索信息（如OpenID和SessionKey）。

以下代码演示了如何完成上述过程：

```
// 在小程序中登录
wx.login({
  success: res => {
    // 这是需要发送给Casdoor的登录代码
    console.log(res.code)

    wx.request({
      url: `${CASDOOR_HOSTNAME}/api/login/oauth/access_token`,
      method: "POST",
      data: {
        "tag": "wechat_miniprogram", // 必填
        "client_id": "6825f4f0af45554c8952",
        "code": res.code,
        "username": this.data.userInfo.nickName, // 登录时更新用户资料。
        "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 // 获取Casdoor的访问令牌
      }
    })
  }
})
```


需要注意的是，`tag` 参数对于告知Casdoor这是来自微信小程序的请求是必须的。

上述代码在登录时包含了微信小程序用户的用户名和头像URI。您可以选择分别传递这两个参数，然后在成功登录并获取访问令牌后将它们传递给Casdoor：

```
wx.getUserProfile({
  desc: '将您的信息分享给Casdoor',
  success: (res) => {
    this.setData({
      userInfo: res.userInfo,
      hasUserInfo: true
    })
    console.log(app.globalData.accessToken)
    wx.request({
      url: `${CASDOOR_HOSTNAME}/api/update-user`, // Casdoor URL
      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，您可以将访问令牌用作您需要的任何Casdoor操作的持有人令牌。

ⓘ 提示

目前，Casdoor无法将现有账户绑定到微信小程序用户。在Casdoor从微信获取OpenID后，如果ID不存在，它将创建一个新用户，如果存在，它将使用现有用户。

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后面的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

这个插件，`authz-casdoor`，可以保护APISIX后面的API，强制每一个请求都要经过身份验证，而无需修改API的代码。

如何启用它？

在创建路由时，您需要指定此插件并提供所有必需的字段。参见下面的示例。

```
curl "http://127.0.0.1:9180/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": {
```

在这个例子中，我们使用APISIX的管理员API，创建了一个指向"httpbin.org:80"的路由"/anything/*"，并启用了"authz-casdoor"插件。此路由现在已在Casdoor的身份验证保护之下。

属性

名称	类型	申请标准	默认	有效期	描述
endpoint_addr	字符串	必填			Casdoor的URL。
client_id	字符串	必填			Casdoor中的客户端ID。
client_secret	字符串	必填			Casdoor中的客户端密钥。
callback_url	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。应强调的是，这个回调URL **必须**属于您为路由指定的"uri"。例如，在这个例子中，<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（这样插件就可以获取OAuth2中的代码和状态）。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？

假设一个从未访问过此路由的新用户即将访问它 (<http://localhost:9080/anything/d?param1=foo¶m2=bar>)。考虑到 "authz-casdoor" 已启用，此访问将首先由 "authz-casdoor" 插件处理。在检查会话并确认该用户尚未通过身份验证后，将会拦截此次访问。保留用户想要访问的原始 URL，他们将被重定向到 Casdoor 的登录页面。

在成功使用用户名和密码（或他们使用的任何其他方法）登录后，Casdoor 会将此用户重定向到带有 GET 参数 "code" 和 "state" 的 "callback_url"。因为 "callback_url" 是由插件知道的，所以当这次对 "callback_url" 的访问被拦截时，将会触发 OAuth2 中的 "Authorization code Grant Flow" 逻辑。这意味着插件将请求访问令牌以确认此用户是否真的已登录。经过这个确认后，插件将会把用户重定向到他们想要访问的原始 URL，这个 URL 是我们之前保存的。已登录的状态也将保留在会话中。

下次此用户想要访问此路由背后的 URL（例如，<http://localhost:9080/anything/d>），在发现此用户之前已经被认证后，这个插件不再重定向此用户。这样，用户可以在不受干扰的情况下访问此路由下的任何内容。

通过 APISIX 的 OIDC 插件连接 Casdoor

Casdoor 可以使用 OIDC 协议连接到 APISIX，本文档将向您展示如何操作。

以下是配置中使用的一些名称：

CASD00R_HOSTNAME：部署 Casdoor 服务器的域名或 IP。

APISIX_HOSTNAME：部署 APISIX 的域名或 IP。

步骤1: 部署Casdoor和APISIX

首先, 部署Casdoor和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. 创建一个新的Casdoor应用程序, 或使用一个已经存在的。
2. 添加一个重定向URL: `http://APISIX_HOSTNAME/REDIRECTWHATYOUWANT`, 并将 `REDIRECTWHATYOUWANT` 替换为所需的重定向URL。
3. 选择 "JWT-Empty" 作为令牌格式选项。
4. 添加所需的提供商并配置其他设置。

Client ID ⓘ:	07860a229bd0b162cdfa
Client secret ⓘ:	ea02l[REDACTED]:9373fe3e
Redirect URLs ⓘ:	<div>Redirect URLs Add</div> <div>Redirect URL</div> <div>http://localhost:9000/callback Add Redirect URL you want</div>
Token format ⓘ:	JWT-Empty Select JWT-Empty

在应用设置页面上，您将找到如上图所示的 `Client ID` 和 `Client Secret` 值。我们将在下一步中使用它们。

打开你最喜欢的浏览器并访问：`http://CASD00R_HOSTNAME/.well-known/openid-configuration`，在那里你会找到Casdoor的OIDC配置。

步骤3：配置APISIX

APISIX 拥有官方 [OIDC](#) 支持，由 [lua-resety-openidc](#) 实现。

您可以根据APISIX OIDC文档自定义设置。以下路由设置将被使用：

```
# 使用您自己的X-API-Key
$ curl -X POST APISIX_HOSTNAME/apisix/admin/routes -H "X-API-Key:
edd1c9f034335f136f87ad84b625c8f1" -d '{
  "uri": "/get",
  "name": "apisix_casdoor_test",
  "plugins": {
    "openid-connect": {
      "client_id": "客户端ID",
      "client_secret": "客户端密钥",
      "discovery": "http://CASD00R_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,
```


PHP

禅道

在禅道中使用 Casdoor 进行身份验证

在ShowDoc中使用Casdoor作为OAuth2服务器

在ShowDoc中使用Casdoor作为OAuth2服务器

Flarum

使用OAuth2连接各种应用程序，如Flarum

Moodle

使用OAuth连接Moodle

禅道

Zentao是一个敏捷（scrum）项目管理系统/工具，但它本身不支持OIDC。要将Zentao与Casdoor SSO集成，我们需要使用一个名为`zentao-oidc`的第三方OIDC模块，本文档将向您展示如何操作。

步骤1：部署Casdoor和Zentao

首先，部署Casdoor和Zentao。成功部署后，请确保：

1. Casdoor 可以正常登录使用。
2. 您可以成功登录并使用Zentao。

步骤2：集成Zentao OIDC第三方模块

通过运行以下命令安装 `zentao-oidc`：

```
git clone https://github.com/casdoor/zentao-oidc.git
```

或者，您可以下载ZIP文件并解压它。

此模块用于将Zentao与OpenId的SSO进行集成。以下是如何使用它：

1. 将整个`oidc`目录复制到Zentao的模块中，并将其作为Zentao的一个模块使用。将下载的包重命名为“oidc”。
2. 配置过滤器。

由于Zentao框架会过滤URL中的参数并且不允许有空格，您需要将以下代码放在 `/config/my.php` 的末尾。

```
$filter->oidc = new stdClass();
$filter->oidc->index = new stdClass();
$filter->oidc->index->paramValue['scope'] = 'reg::any';
```

3. 修改 `/module/commom/model.php`。

将 'oidc' 添加到匿名访问列表中，并在 `model.php` 的 `isOpenMethod` 方法中添加一行。

```
public function isOpenMethod($module, $method)
{
    if ($module == 'oidc' and $method == 'index') {
        return true;
    }
}
```

4. 如果你不希望看到Zentao的登录界面，可以直接前往Casdoor的登录界面。

修改 `/module/common/model.php` 中的 `public function checkPriv()` 的最后一行代码。

```
//返回 print(js::locate(helper::createLink('user', 'login',
"referer=$referer")));
返回 print(js::locate(helper::createLink('oidc', 'index',
"referer=$referer")));
```

5. 修改 `framework/base/router.class.php` 中的 `setSuperVars()` 方法，并注释掉以下语句。

```
public function setSuperVars()  
// unset($_REQUEST);
```

步骤3：配置Casdoor应用程序

1. 创建一个新的Casdoor应用程序，或使用一个已经存在的。
2. 添加您的重定向URL。

Client ID ? :	<input type="text" value="d8d7715e24f077066a20"/>				
Client secret ? :	<input type="password" value="REDACTED"/>				
Cert ? :	<input type="text" value="cert-built-in"/>				
Redirect URLs ? :	<table><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填写其他所需设置。

步骤4：配置Zentao

在 `oidc` 目录中配置 `config.php` 文件。

```
$config->oidc->clientId = "<您的ClientId>";  
$config->oidc->clientSecret = "<您的ClientSecret>";  
$config->oidc->issuer = "http://localhost:8000";
```

在 `module/oidc` 中设置你的重定向URL，在 `public function index()` 方法中。

```
$oidc->setRedirectURL($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。默认情况下，框架使用破折号("-")。请参考官方的Zentao框架以获取更多详情。 ["zentaoPHP框架"](#)

在ShowDoc中使用Casdoor作为OAuth2服务器

在ShowDoc中使用Casdoor进行身份验证

ShowDoc是一个在线API文档和技术文档工具，非常适合IT团队使用。ShowDoc使得使用Markdown语法编写美观的API文档、数据字典文档、技术文档、在线Excel文档等变得简单易行。

ShowDoc支持第三方认证，包括OAuth2。这是一个实现此目标的教程。

步骤1：创建一个Casdoor应用


前往你的Casdoor并添加一个名为ShowDoc的新应用。这是在Casdoor中创建ShowDoc应用程序的一个示例。

Edit Application Save **Save & Exit**

Name [?]: myApplication

Display name [?]: myApplication

Logo [?]:
URL [?]: 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

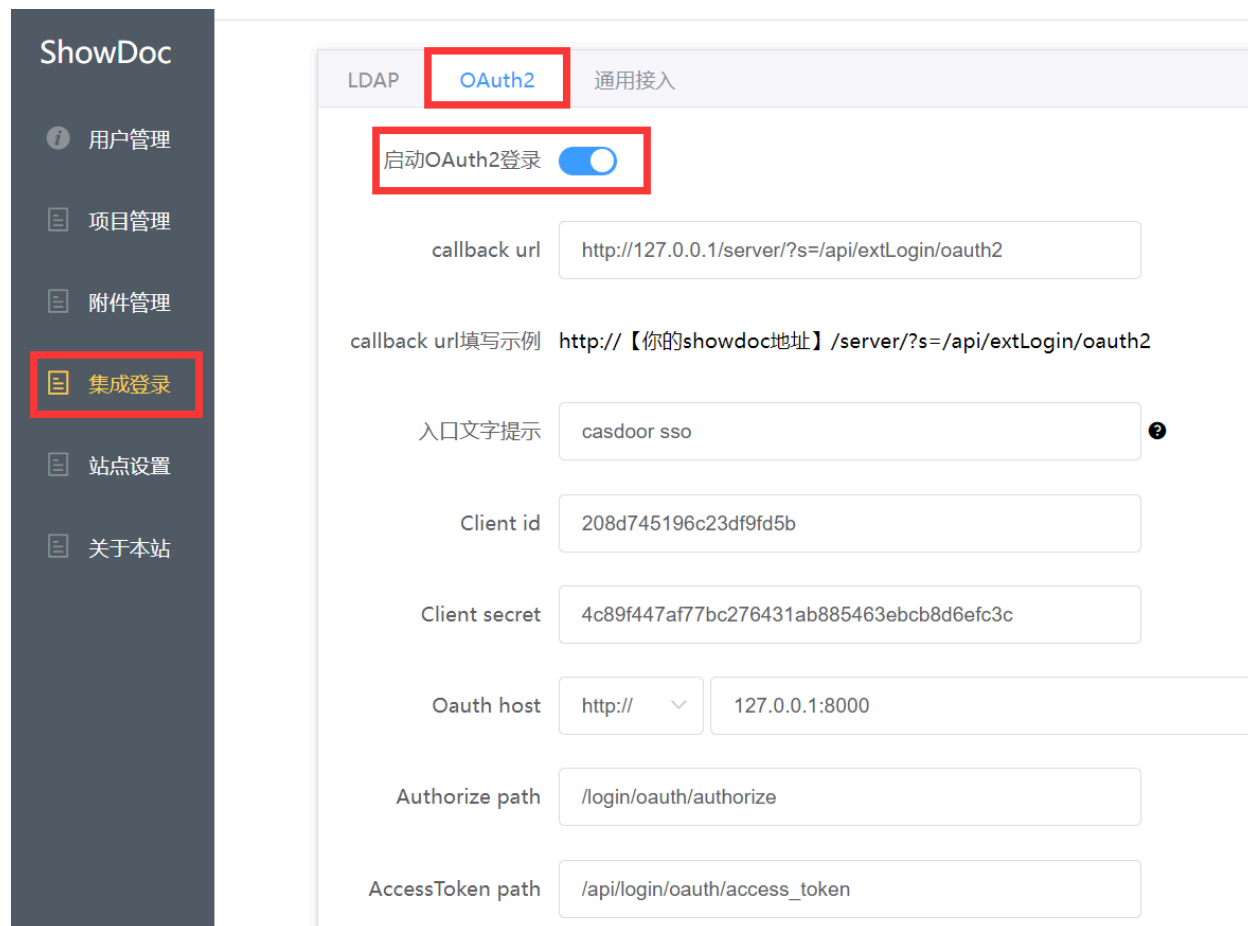
请记住下一步所需的 **客户端 ID** 和 **客户端密钥**。

! 信息

请不要在此步骤中填写**回调URL**。URL取决于下一步中ShowDoc的配置。我们稍后会回来设置一个正确的回调URL。

步骤2：配置ShowDoc

首先，启用OAuth2登录按钮。然后，按照示例填写回调URL。在上一步中记住的client ID和client secret填入。



The screenshot shows the ShowDoc configuration interface. On the left is a dark sidebar with the '集成登录' (Integrated Login) option highlighted in a red box. The main content area has three tabs: 'LDAP', 'OAuth2', and '通用接入', with 'OAuth2' selected and highlighted in a red box. Below the tabs, there is a toggle switch for '启动OAuth2登录' (Enable OAuth2 Login), which is turned on and highlighted in a red box. The configuration fields are as follows:

- callback url: `http://127.0.0.1/server/?s=/api/extLogin/oauth2`
- callback url填写示例: `http://【你的showdoc地址】/server/?s=/api/extLogin/oauth2`
- 入口文字提示: `casdoor sso`
- Client id: `208d745196c23df9fd5b`
- Client secret: `4c89f447af77bc276431ab885463ebcb8d6efc3c`
- Oauth host: `http://` (dropdown) `127.0.0.1:8000`
- Authorize path: `/login/oauth/authorize`
- AccessToken path: `/api/login/oauth/access_token`

Authorize path、AccessToken path和User info path是必需的。您可以按照下面的示例来填写。

```
Authorize path: /login/oauth/authorize
AccessToken path: /api/login/oauth/access_token
User info path: /api/get-account
```

步骤3：在Casdoor中配置回调URL

返回到步骤1中的应用编辑页面，并添加你在ShowDoc中填写的回调URL。

Redirect URLs ⓘ :

Redirect URL
http://127.0.0.1/server/?s=/api/extLogin/oauth2

步骤4：尝试使用ShowDoc

你应该在登录页面上看到以下内容：

登录



注册新账号

[casdoor sso](#)

恭喜您！ 您已完成所有步骤 按下'Casdoor SSO'按钮，您将被重定向到Casdoor登录页面。

Flarum

Casdoor可以使用OAuth2连接各种应用程序。在这个例子中，我们将向您展示如何使用OAuth2将Flarum连接到您的应用程序。

以下是您需要的一些配置名称：

`CASDOOR_HOSTNAME`：部署Casdoor服务器的域名或IP。

`Flarum_HOSTNAME`：部署Flarum的域名或IP。

步骤1：部署Casdoor和Flarum

首先，部署Casdoor和Flarum。

成功部署后，请确保：

1. 您已经下载了Flarum插件FoF Passport。
2. Casdoor可以正常登录和使用。
3. 在prod模式下部署Casdoor时，您可以设置CASDOOR_HOSTNAME = `http://localhost:8000`。查看[生产模式](#)。

步骤2：配置Casdoor应用程序

1. 创建一个新的Casdoor应用程序或使用现有的一个。
2. 找到重定向URL： `<CASDOOR_HOSTNAME>/auth/passport`。
3. 将重定向URL添加到Casdoor应用程序：

The screenshot shows a configuration interface with the following fields:


- Client ID:** 014ae4bd048734ca2dea
- Client secret:** f26a4115725867b7bb7b668c81e18f77ae1544d
- Cert:** cert-built-in
- Redirect URLs:** A table with one entry: <your flarum install>/auth/passport.

在应用程序设置页面上，您会找到两个值：Client ID 和 Client secret。我们将在下一步中使用这些值。

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的OIDC配置。

步骤3：配置Flarum

1. 安装插件FoF Passport。
2. 配置应用程序：

 **FoF Passport**
The OAuth2 (and Laravel passport) compatible oauth extension

Enabled

OAuth authorization URL
`https://door.casdoor.com/login/oauth/authorize`

OAuth token URL
`https://door.casdoor.com/api/login/oauth/access_token`

Api URL providing user details when authenticated
`https://door.casdoor.com/api/user`

OAuth application ID
`014ae4bd048734ca2dea`

OAuth application secret
`f26a4115725867b7bb7b668c81e1f8f7fae1544d`

OAuth scopes to request
`openid profile email`

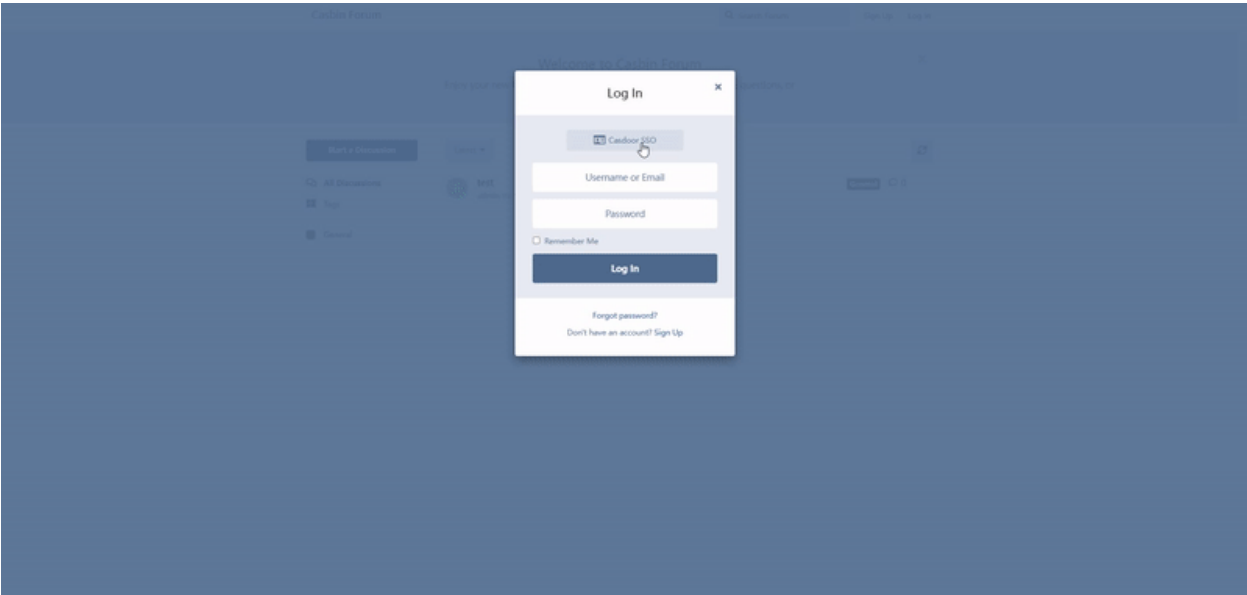
Label for login button
`Casdoor SSO`

Icon for login button
`far fa-id-card`

3. 在Casdoor应用程序页面中找到Client ID和Client Secret。

- Token server URL: `http://CASD00R_HOSTNAME/api/login/oauth/access_token`
- Authorization server URL: `http://CASD00R_HOSTNAME/login/oauth/authorize`
- UserInfo server URL: `http://CASD00R_HOSTNAME/api/get-account`
- Scopes: `address phone openid profile offline_access email`

退出Flarum并测试SSO。



Moodle

Casdoor可以用来使用OAuth连接Moodle。

以下是一些配置设置：

- `CASD00R_HOSTNAME`：部署Casdoor服务器的域名或IP。
- `Moodle_HOSTNAME`：部署Moodle的域名或IP。

步骤1：部署Casdoor和Moodle

首先，部署Casdoor和Moodle。

成功部署后，确保以下内容：

1. Casdoor可以登录并无问题地使用。
2. 在`prod`模式下部署Casdoor时，您可以将`CASD00R_HOSTNAME`设置为`http://localhost:8000`。查看[生产模式](#)。

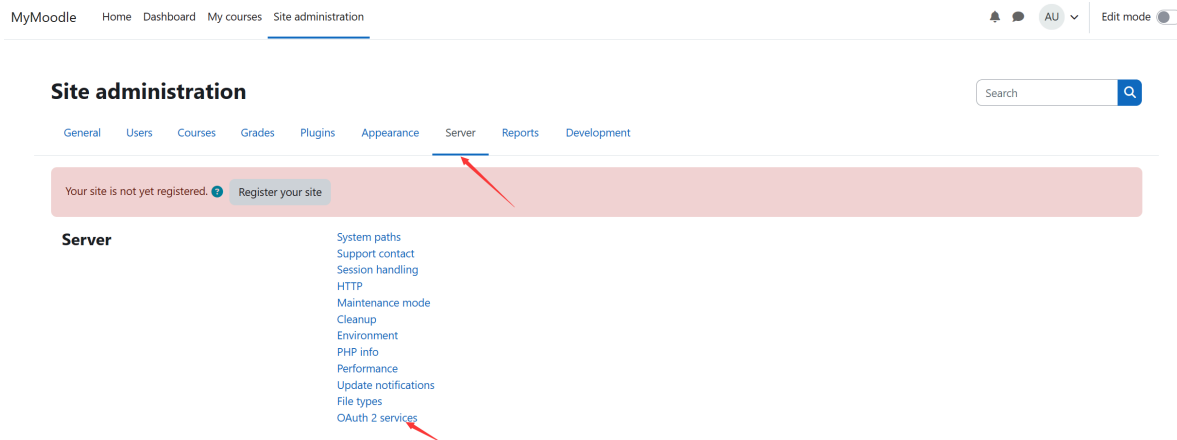
步骤2：配置Casdoor应用程序

1. 创建一个新的Casdoor应用程序或使用现有的一个。
2. 找到重定向URL：`Moddle_HOSTNAME/admin/oauth2callback.php`。
3. 将重定向URL添加到Casdoor应用程序。

有关OAuth的更多信息，请参阅[OAuth](#)。

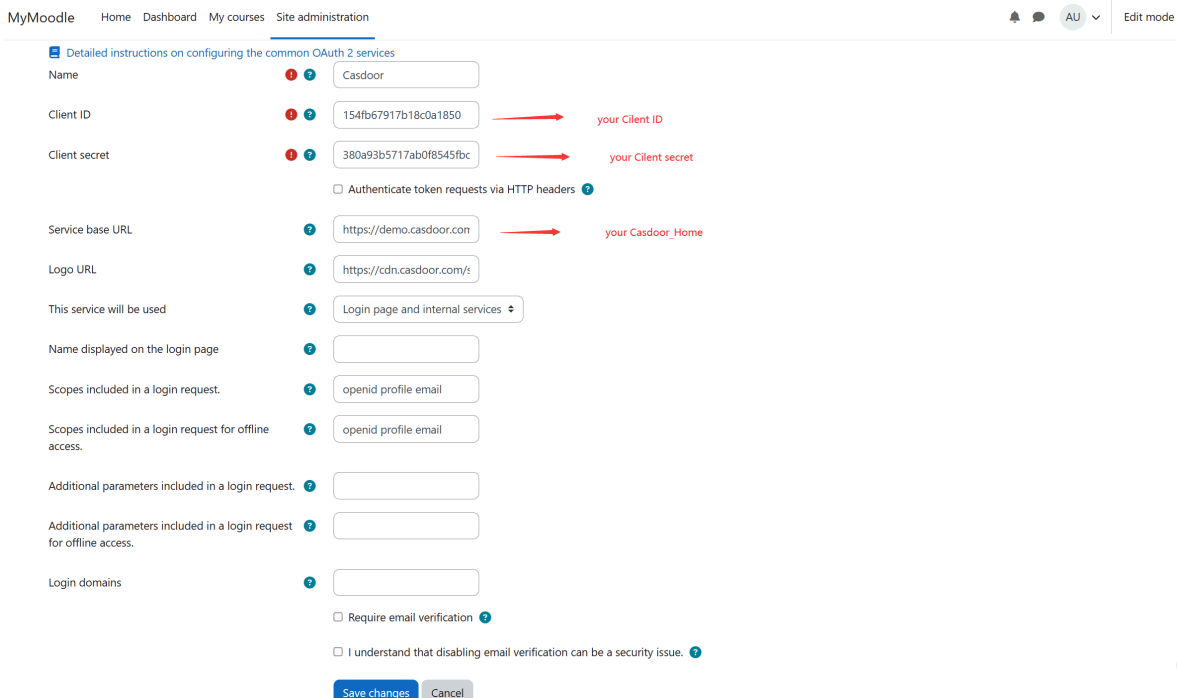
步骤3：配置Moodle

1. 定位OAuth



o













2. 配置此应用程序



o

3. 配置此映射

User field mappings for issuer: Casdoor



External field name	Internal field name	Edit
address	address	 
email	email	 
name	firstname	 
phone	phone1	 
picture	picture	 
perferred_username	username	 

Create new user field mapping for issuer "Casdoor"

o

4. 定位OAuth2插件

General Users Courses Grades **Plugins** Appearance Server Reports Development

Your site is not yet registered.  Register your site 

Plugins

Install plugins
Plugins overview

Activity modules

- Manage activities
- Common activity settings
- Assignment
 - Assignment settings
 - Submission plugins
 - Manage assignment submission plugins
 - File submissions
 - Online text submissions
- Feedback plugins
 - Manage assignment feedback plugins
 - Feedback comments
 - Annotate PDF
 - File feedback
 - Offline grading worksheet
- Book
- Chat
- Database
- External tool
 - Manage tools
- Feedback
- File
- Folder
- Forum
- Glossary
- HSP
- IMS content package
- Lesson
- Page
- Quiz
 - General settings
 - Safe Exam Browser templates
 - Safe Exam Browser access rules
- SCORM package
- Text and media area
- URL
- Workshop


Admin tools

- Manage admin tools
- Accessibility
 - Brickfield registration
 - Accessibility toolkit settings
- Reports
- Recycle bin

Antivirus plugins

Manage antivirus plugins

Authentication

- Manage authentication 
- Email-based self-registration
- Manual accounts
- OAuth 2

o

5. 启用OAuth2插件

Manage authentication

Available authentication plugins

Name	Users	Enable	Up/Down	Settings	Test settings	Uninstall
Manual accounts	2			Settings		
No login	0					
Email-based self-registration	0		↓	Settings		Uninstall
OAuth 2	8		↑	Settings	Test settings	

o

6. 如果你想阻止编辑Casdoor的电子邮件

Lock user fields

You can lock user data fields. This is useful for sites where the user data is maintained by the administrators manually by editing user records or uploading using the 'Upload users' facility. If you are locking fields that are required by Moodle, make sure that you provide that data when creating user accounts or the accounts will be unusable.

Consider setting the lock mode to 'Unlocked if empty' to avoid this problem.

Lock value (First name) <small>auth_oauth2 field_lock_firstname</small>	Unlocked	Default: Unlocked
Lock value (Last name) <small>auth_oauth2 field_lock_lastname</small>	Unlocked	Default: Unlocked
Lock value (Email address) <small>auth_oauth2 field_lock_email</small>	Locked	Default: Unlocked
Lock value (City/town) <small>auth_oauth2 field_lock_city</small>	Unlocked	Default: Unlocked
Lock value (Country) <small>auth_oauth2 field_lock_country</small>	Unlocked	Default: Unlocked
Lock value (Language) <small>auth_oauth2 field_lock_lang</small>	Unlocked	Default: Unlocked

here is switch to lock email

o

有关Moodle的更多信息，请参阅[Moodle](#)和[字段映射](#)。

退出Moodle并测试SSO。

MyMoodle Website

Ruby

GitLab

在自行开发的GitLab服务器中使用Casdoor进行身份验证

GitLab

Casdoor可以使用OIDC协议链接到自部署的GitLab服务器，本文档将向您展示如何操作。

⚠️ 注意事项

正如[GitLab文档](#)所述，GitLab只能与使用HTTPS的OpenID提供商一起工作，因此你需要使用HTTPS部署Casdoor，例如将Casdoor放在配置了SSL证书的NGINX反向代理后面。Casdoor本身默认只通过HTTP在8000端口监听，没有HTTPS相关功能。

以下是在配置中提到的一些名称：

`CASDOOR_HOSTNAME`：部署Casdoor服务器的域名或IP，例如，

`https://door.casbin.com`。

`GITLAB_HOSTNAME`：部署GitLab的域名或IP，例如，`https://gitlab.com`。

步骤1：部署Casdoor和GitLab

首先，应部署Casdoor和GitLab。

成功部署后，您需要确保：

1. Casdoor可以正常登录和使用。
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
```

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URL: `http://GITLAB_HOSTNAME/users/auth/openid_connect/callback`。
3. 添加您想要的提供者并补充其他设置。

Description ⓘ:

Organization ⓘ:

Client ID ⓘ: Client ID

Client secret ⓘ: Client secret

Redirect URLs ⓘ:

Redirect URLs

Redirect URL
<input type="text" value="http://GITLAB_HOSTNAME/users/auth/openid_connect/callback"/> GitLab redirect url

值得注意的是，您可以在应用设置页面上获取两个值：`Client ID`和`Client secret`（参见上图），我们将在下一步中使用它们。

打开你最喜欢的浏览器并访问：`http://CASDOOR_HOSTNAME/.well-known/openid-configuration`，在那里你将看到Casdoor的OIDC配置。

步骤3：配置GitLab

You can follow the steps below to set this up, or make custom changes according to [this document](#) (e.g., if you are installing GitLab using source code rather than the Omnibus).

1. 在 GitLab 服务器上，打开配置文件。

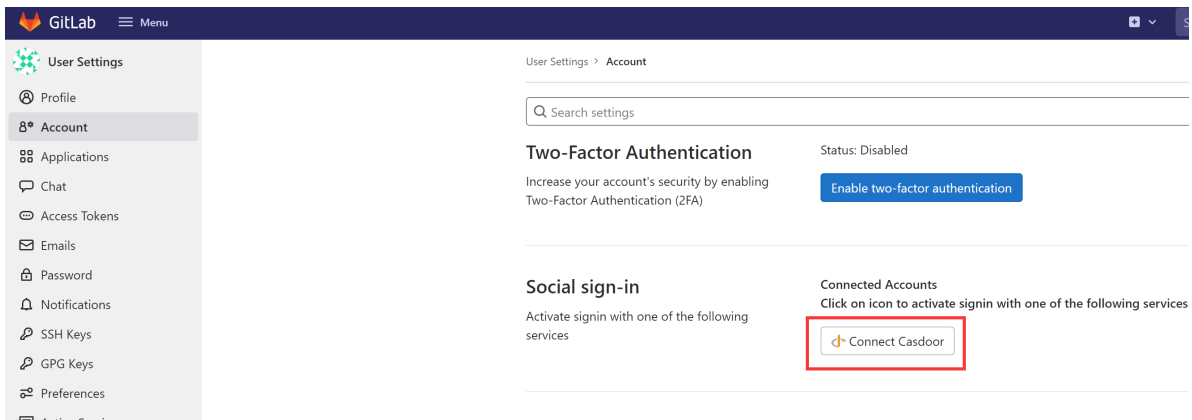
```
sudo editor /etc/gitlab/gitlab.rb
```

2. 添加提供商配置。(HOSTNAME URL应包含http或https)

```
gitlab_rails['omniauth_providers'] = [  
  {  
    name: "openid_connect",  
    label: "Casdoor", # 可选的登录按钮标签, 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: "<YOUR CLIENT ID>",  
        secret: "<YOUR CLIENT SECRET>",  
        redirect_uri: "<GITLAB_HOSTNAME>/users/auth/  
openid_connect/callback"
```


3. 重新启动 GitLab 服务器。

4. 每个已注册用户都可以打开 `GITLAB_HOSTNAME /-/profile/account` 并连接 Casdoor 账户。



5. 完成！现在，您可以使用 Casdoor 登录到您自己的 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.

Username or email

Password

Remember me [Forgot your password?](#)

Don't have an account yet? [Register now](#)

Sign in with

Remember me

Haskell

Hasura

在集成之前，我们需要在本地部署Casdoor。

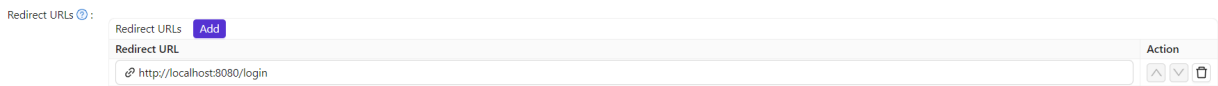
Hasura

在集成之前，我们需要在本地部署Casdoor。

然后我们可以按照以下步骤在我们自己的应用中快速实现基于Casdoor的登录页面。

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URL: `http://CASDOOR_HOSTNAME/login`

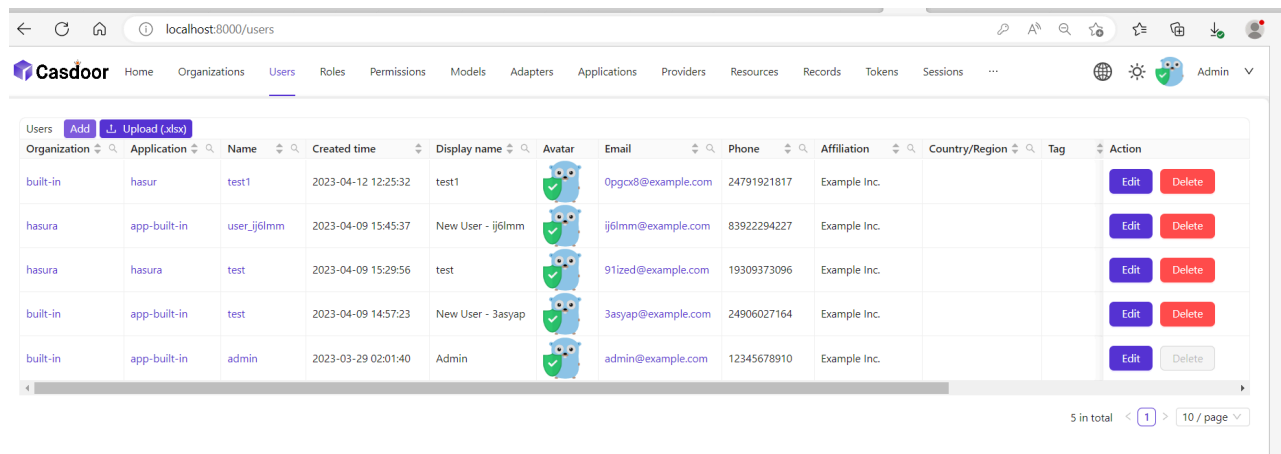


3. 复制客户端ID；我们将在接下来的步骤中需要它。

在Casdoor中添加一个用户

现在你有了应用程序，但没有用户。这意味着你需要创建一个用户并分配角色。

转到“用户”页面，然后点击右上角的“添加用户”。这将打开一个新页面，您可以在其中添加新用户。



在添加用户名并添加Hasura组织（其他详细信息可选）后保存用户。

现在你需要为你的用户设置一个密码，你可以通过点击“管理你的密码”来做到这一点。

为您的用户选择一个密码并确认。

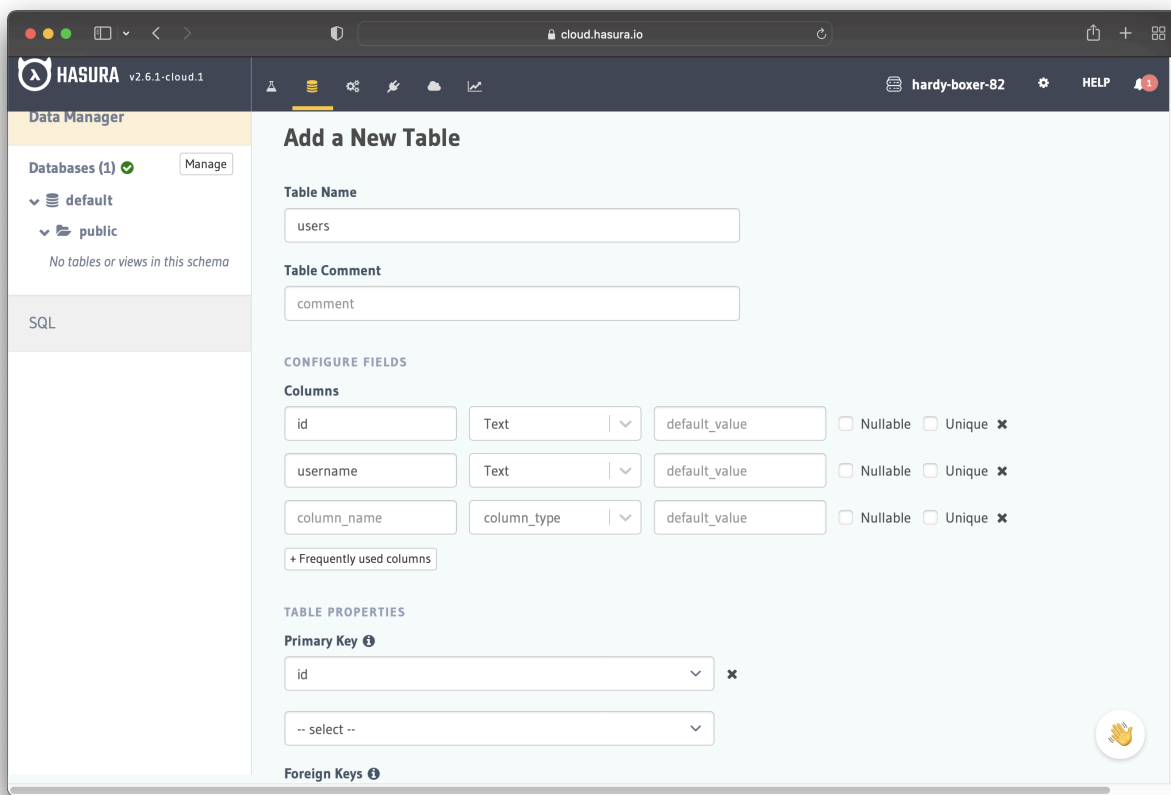
构建Hasura应用

通过Docker或Hasura Cloud启动Hasura。

现在创建一个带有以下列的 `users` 表：

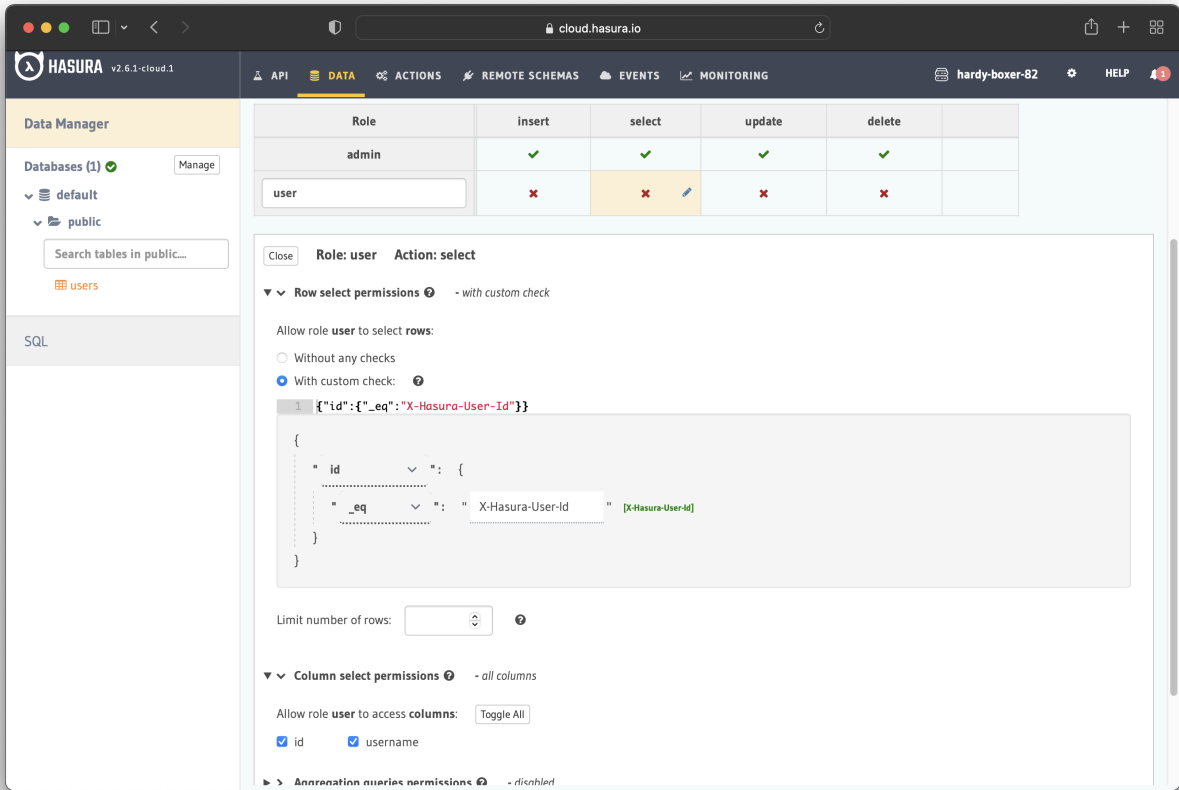
- `id` 类型为文本（主键）
- `username` 类型为文本

参考下面的图片作为参考。



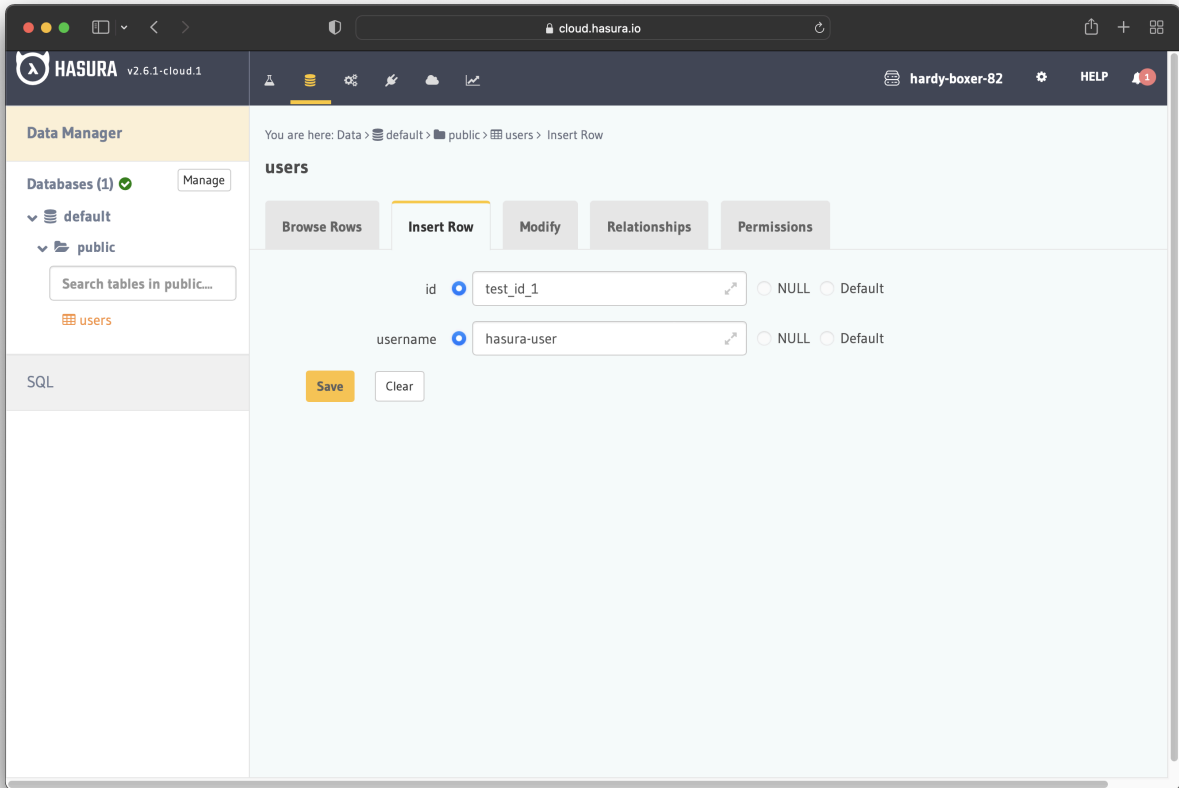
下一步是为应用创建一个 `user` 角色。用户应该只能看到他们自己的记录，而不能看到其他人的记录。

按照下图所示配置 `user` 角色。有关更多信息，请阅读[在Hasura中配置权限规则](#)。



这样，用户就不能读取其他人的记录。他们只能访问他们自己的。

出于测试目的，添加一个虚拟用户。这是为了确保当你使用JWT令牌时，你只能看到你自己的用户详情，而不能看到其他用户的详情。



现在你需要在Hasura中设置 `JWT_SECRET`。

用Casdoor配置Hasura

在这一步，你需要将 `HASURA_GRAPHQL_JWT_SECRET` 添加到Hasura。

为此，转到Hasura `docker-compose.yml`，然后按照下面的方式添加新的 `HASURA_GRAPHQL_JWT_SECRET`。

`HASURA_GRAPHQL_JWT_SECRET` 应该是以下格式。记住将 `<Casdoor endpoint>` 更改为你自己的Casdoor的URL（如 `https://door.casdoor.com`）

```
HASURA_GRAPHQL_JWT_SECRET: '{"claims_map": {
  "x-hasura-allowed-roles": {"path": "$.roles"},
  "x-hasura-default-role": {"path": "$.roles[0]"},
  "x-hasura-user-id": {"path": "$.id"}
}, "jwk_url": "<Casdoor endpoint>/well-known/jwks"}
```

保存更改并重新加载docker。

```
## enable debugging mode. It is recommended to disable this in production
HASURA_GRAPHQL_DEV_MODE: "true"
HASURA_GRAPHQL_ENABLED_LOG_TYPES: startup, http-log, webhook-log, websocket-log, query-log
HASURA_GRAPHQL_ADMIN_SECRET: myadminsecretkey
HASURA_GRAPHQL_JWT_SECRET: '{"claims_map": {
  "x-hasura-allowed-roles": ["user", "editor"],
  "x-hasura-default-role": "user",
  "x-hasura-user-id": "4ec7ccee-ec7b-4191-a78d-e11f50686f8b"
}, "jwk_url": "https://door.casdoor.com/.well-known/jwks"}'
```

获取JWT令牌

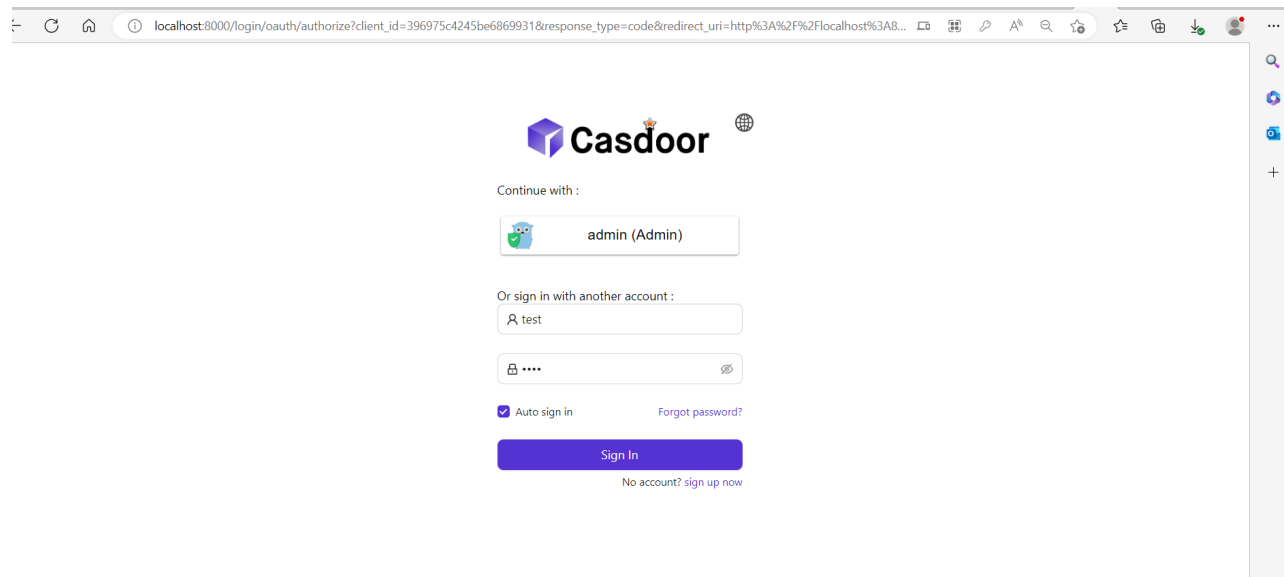
由于没有客户端实现，你可以通过以下URL获取你的访问令牌：

```
http://localhost:8000/login/oauth/authorize?client_id=<client ID>&response_type=code&redirect_uri=http%3A%2F%2Flocalhost%3A8080%2Flogin&scope=read&state=app-built-in<public certificate>>
```

将 `client ID` 更改为你之前复制的ID，并输入Casdoor的公共证书，你可以在Casdoor的Certs页面中找到。

然后输入你之前为Hasura创建的用户名和密码。

点击“登录”



返回到Casdoor/Token页面。

localhost:8000/tokens

Casdoor Home Organizations Users Roles Permissions Models Adapters Applications Providers Resources Records Sessions ... Admin

Tokens Add

Name	Created time	Application	Organization	User	Authorization code	Access token	Action
b6ea3e35-abc4-41d8-a1a2-01f00fd8264b	2023-04-12 13:06:53	hasura	hasura	test	433b504b4f6a593e4a11	eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0e...	Edit Delete
16024557-df21-4779-bfb9-959e5dae078c	2023-04-12 12:51:47	hasur	built-in	test1	2879fbc282019cf7f23c	eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0e...	Edit Delete
f3cb1070-c2d4-40f0-8bc0-59919d26d162	2023-04-11 15:04:00	hasura	hasura	test	2a370971798d403c6ef	eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0e...	Edit Delete
64993582-2322-4df7-ab20-cb23201bc77b	2023-04-11 00:37:22	springboot	built-in	admin	a2396037c3ba4fd9221e	eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0e...	Edit Delete
f65a3813-a655-47f0-9c9a-f08ce4607815	2023-04-11 00:31:37	springboot	built-in	admin	d048c7f9cd1469fd829d	eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0e...	Edit Delete
5828069e-15eb-4c92-933c-fecda8ed621c	2023-04-11 00:06:54	springboot	built-in	admin	7cc27dc752cc4188ac8d	eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0e...	Edit Delete
2277e0f2-7e78-462f-a654-3c53759784af	2023-04-11 00:05:17	springboot	built-in	admin	56141e709a06931b7faa	eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0e...	Edit Delete
55bd324a-6039-40f6-b707-2a55d78ae911	2023-04-11 00:05:07	springboot	built-in	admin	9a1413bc172591a64353	eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0e...	Edit Delete
4b30acbe-fa22-4387-8098-9a4e670f6972	2023-04-10 23:59:19	springboot	built-in	admin	88b0997b675917f20fdc	eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0e...	Edit Delete

找到你之前输入的用户名，然后点击“编辑”

复制访问令牌

Edit Token Save Save & Exit

Name: b6ea3e35-abc4-41d8-a1a2-01f00fd8264b

Application: hasura

Organization: hasura

User: test

Authorization code: 433b504b4f6a593e4a11

Access token: eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0eXAI0iUkV1QjFQeyJvd25ici6mhhc3VyYSIsIm5hbWUiOiJ0ZXN0iWY3JlYXRIZFRpbWUjOiYiYMDlzlTA0LTASVDE1OjI5OjU2KzA4OjAwIiwidXNlbnRlYXRIZFRpbWUjOiIiLCJpZCI6IjRlY

Expires in: 604800

Scope: read

Token type: Bearer

Save Save & Exit

现在你可以使用访问令牌来发出经过身份验证的请求。Hasura返回了适当的用户，而不是从数据库返回所有用户。

Python

 JumpServer

使用CAS连接JumpServer

JumpServer

Casdoor可以用来连接JumpServer。

以下是配置中的一些名称：

`CASD00R_HOSTNAME`：部署Casdoor服务器的域名或IP。

`JumpServer_HOSTNAME`：部署JumpServer的域名或IP。

步骤1：部署Casdoor和JumpServer

首先，部署Casdoor和JumpServer。

部署成功后，请确保以下内容：

1. Casdoor可以正常登录和使用。
2. 在`prod`模式下部署Casdoor时，可以将`CASD00R_HOSTNAME`设置为`http://localhost:8000`。查看[生产模式](#)。

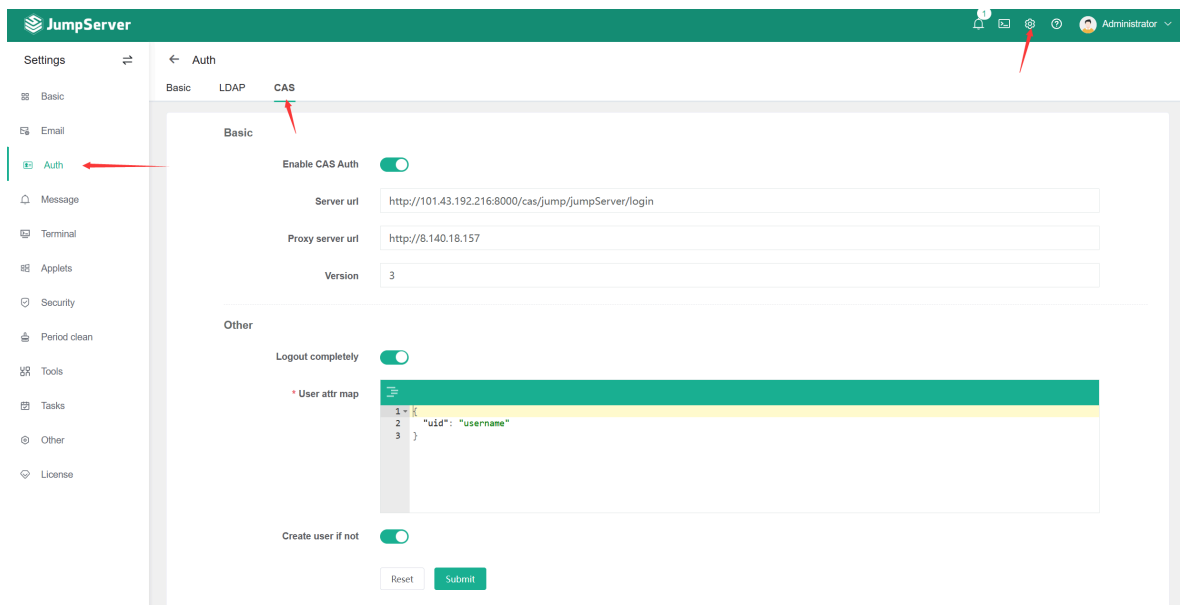
步骤2：配置Casdoor应用

1. 创建一个新的Casdoor应用或使用现有的应用。
2. 找到一个重定向URL：`CASD00R_HOSTNAME /cas/ 你的组织 / 你的应用 /login`。
3. 将你的重定向URL添加到Casdoor应用：`JumpServer_HOSTNAME`。

有关CAS的更多信息，请参考文档。

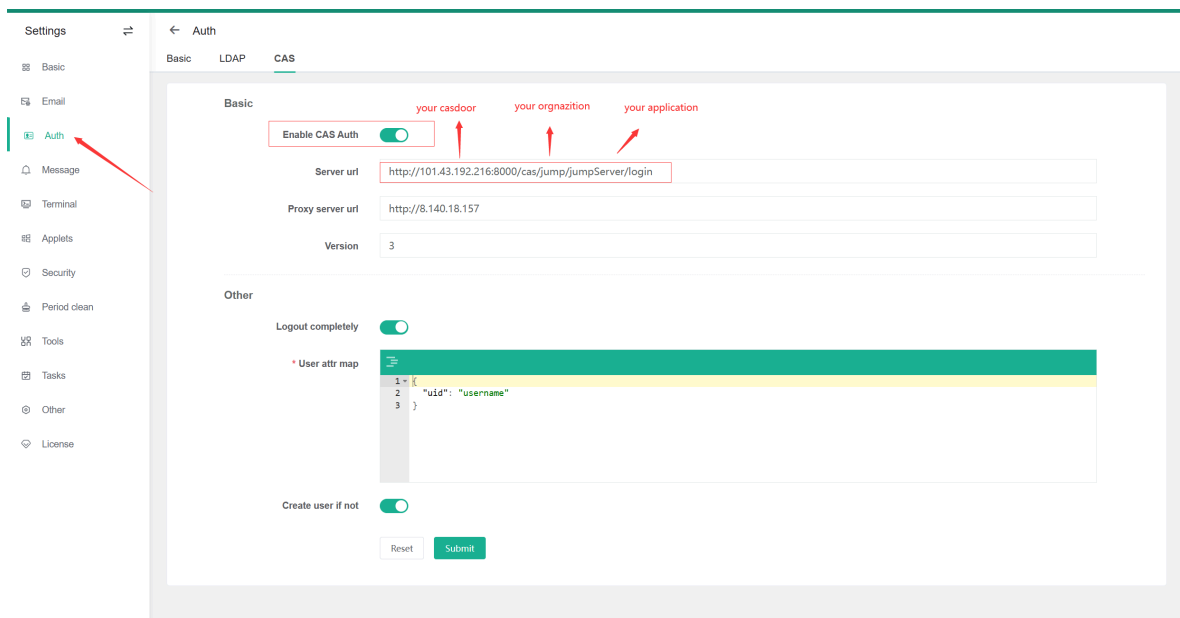
步骤3：配置JumpServer

1. 找到Auth:



o

2. 配置此应用:

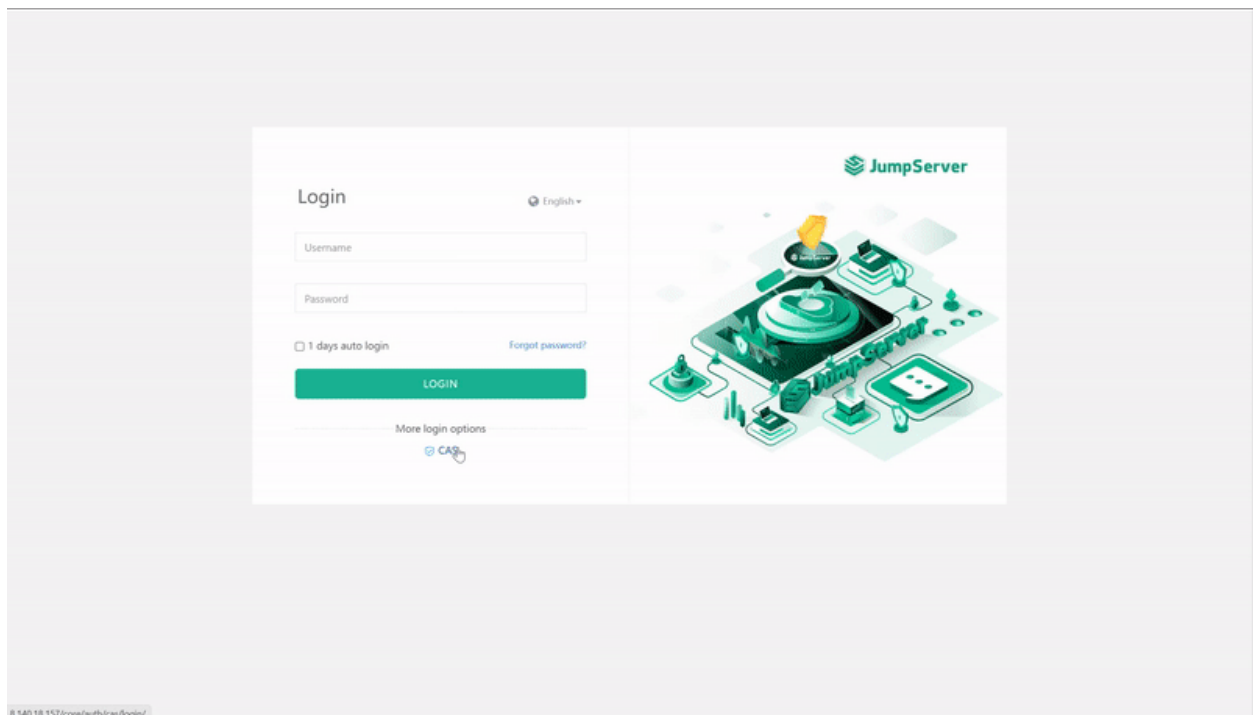


o

- `/login` 端点: `https://door.casdoor.com/cas/casbin/cas-java-app/login`。
- `/logout` 端点: `https://door.casdoor.com/cas/casbin/cas-java-app/logout`。
- `/serviceValidate` 端点: `https://door.casdoor.com/cas/casbin/cas-java-app/serviceValidate`。
- `/proxyValidate` 端点: `https://door.casdoor.com/cas/casbin/cas-java-app/proxyValidate`。

有关CAS和JumpServer的更多信息, 请参考文档。

退出JumpServer并测试SSO:





监控

监控

Web UI

在Casdoor网页上监控运行时信息

Prometheus

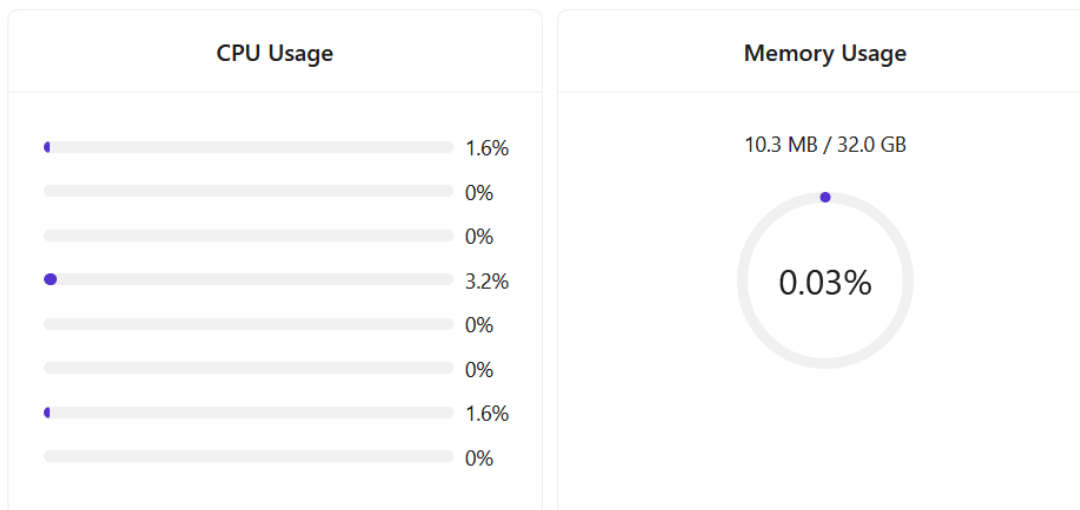
使用Prometheus收集有关运行Casdoor的信息。

Web UI

您可以在[Casdoor网页](#)上监控Casdoor的运行信息，包括CPU使用率，内存使用率，API延迟和API吞吐量。

在UI上，您可以查看以下信息：

- CPU使用率和内存使用率



- API延迟，包括计数次数和平均延迟

API Latency			
GET	/api/get-cert	3	0.667
GET	/api/get-certs	27	1.333
GET	/api/get-chats	27	1.519
GET	/api/get-default-application	3	5.333
GET	/api/get-email-and-phone	1	1.000
GET	/api/get-global-providers	58	1.293

- API吞吐量，包括总吞吐量和每个API的吞吐量

API Throughput		
Total Throughput: 2		
Name	Method	Throughput
/api/get-prometheus-info	GET	1
/api/get-system-info	GET	1

Prometheus

要收集Casdoor的运行指标，如API吞吐量，API延迟，CPU使用率，内存使用率等，您需要配置您的Prometheus配置文件。

```
global:
  scrape_interval: 10s # 获取指标的时间间隔

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
  - job_name: 'casdoor' # 要监控的应用程序的名称
    static_configs:
      - targets: ['localhost:8000'] # Casdoor 部署的后端地址
    metrics_path: '/api/metrics' # 收集指标的路径
```

配置成功后，您将在Prometheus中找到以下信息：



国际化

Casdoor支持多种语言. 通过部署[Crowdin](#) 翻译, 我们可以提供 西班牙语、法语、德语、中文、印尼语、日语、韩语等等的支持.

Casdoor使用官方的 [Crowdin cli](#) 来同步Crowdin 的翻译。如果您想要添加对其他语言的支持, 请提交您在 [社区](#) 中的提案。此外, 如果您想要帮助我们加快翻译工作, 请考虑帮助我们翻译 [Crowdin](#)。



贡献者指南

欢迎使用 Casdoor! 此文档作为如何为Casdoor做出贡献的指南。

如果您发现任何不正确或缺失的信息, 请留下您的评论或建议。

参与其中

有许多方式可以为Casdoor做贡献。以下列出了一些贡献方式:

- 使用Casdoor并报告问题。 在使用Casdoor时, 如果遇到任何问题 - 无论是错误还是建议 - 请在[GitHub Discussions](#)或[Discord](#)上报告, 然后再在GitHub上提交问题。

! 信息

在报告问题时, 请使用英语详细描述您的问题。

- 帮助编写文档。 开始从文档工作做起是一个好的选择。
- 帮助解决问题。 我们有一个表格, 其中包含适合初学者的简单任务, 位于[Casdoor Easy Tasks](#), 并用不同的标签标记不同级别的挑战。

贡献

如果你准备创建一个PR, 这里是贡献者的工作流程:

1. 将其分叉到你自己的仓库。

2. 将你的分支克隆到本地仓库。
3. 创建一个新的分支并在其上工作。
4. 保持你的分支同步。
5. 提交你的更改。确保你的提交信息简洁明了。
6. 将你的提交推送到你的分叉仓库。
7. 创建从您的分支到我们的**master**分支的合并请求。

合并请求

在你开始之前

Casdoor以GitHub作为其开发平台，拉取请求是主要的贡献来源。

在开启拉取请求之前，你需要了解一些事情：

- 您首次创建拉取请求时，需要签署**CLA**。
- 解释为什么你要提交这个拉取请求，以及它将对仓库产生什么影响。
- 只允许一个提交。如果PR做的事情超过一件，请将其拆分。
- 如果有任何新添加的文件，请在新文件的顶部包含Casdoor许可证。

```
// 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标题必须为小写。

1. **修复**：一个类型的提交 `fix` 修复了你的代码库中的一个错误。

```
fix: prevent racing of requests
```

2. **特性**：一个类型的提交 `feat` 向代码库引入了一个新的功能。

```
feat: allow provided config object to extend other configs
```

3. **文档**：一个类型的提交 `docs` 添加或改进文档。

```
docs: correct spelling of CHANGELOG
```

有关更多详细信息，请参阅[Conventional Commits](#)页面。

将PRs与Issues关联

您可以将拉取请求链接到问题，以显示正在进行的修复，并在合并拉取请求时自动关闭问题。

使用关键词将拉取请求链接到问题

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

例如，在同一个仓库中的一个问题：

```
Fix: #902
```

有关更多详细信息，请参阅[将 Pull Request 链接到问题](#)。

修改PRs

您的PR可能需要修改。当代码需要更改时，请修改同一PR；不要关闭PR并开启一个新的。这是一个例子：

- 在你的本地修改代码。
- 修改你的提交。

```
git commit --amend
```

- 推送代码到远程仓库

```
git push --force
```

然后，你就成功地修改了PR！

相关代码

一些原则：

- 可读性：重要的代码应该有良好的文档记录。代码风格应与现有的一致。

命名规则

例如，变量名使用 `signupUrl`，UI使用 `Signup URL`。

如何更新i18n数据？

请注意，我们使用 [Crowdin](#) 作为翻译平台，使用 `i18next` 作为翻译工具。当你在 `web/` 目录中使用 `i18next` 添加字符串时，你可以运行 `i18n/generate_test.go` 来自动生成 `web/src/locales/**/data.json`。

运行 `i18n/generate_test.go`：

```
cd i18n && go test
```

默认情况下，所有语言都以英文填写。在你的PR被合并后，我们鼓励你帮助翻译新添加的字符串在 `web/src/locales/zh/data.json` 通过 [Crowdin](#)。

注意事项

如果你对某种语言不熟悉，请不要翻译它；保持文件原样。

许可证书

通过向Casdoor做出贡献，您同意您的贡献将根据Apache许可证进行许可。