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。

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`。

> **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"`
```

> **提示**
>
> The `Properties` field is a flexible key-value map for storing custom user attributes. See the User Properties documentation for detailed usage examples and best practices.

应用程序

一个**应用程序**代表需要由Casdoor保护的网络服务，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"`
    RedirectUris         []string       `xorm:"varchar(1000)" json:"redirectUris"`
    TokenFormat          string         `xorm:"varchar(100)" json:"tokenFormat"`
    ExpireInHours        int            `json:"expireInHours"`
    RefreshExpireInHours int            `json:"refreshExpireInHours"`
    SignupUrl            string         `xorm:"varchar(200)" json:"signupUrl"`
    SigninUrl            string         `xorm:"varchar(200)" json:"signinUrl"`
    ForgetUrl            string         `xorm:"varchar(200)" json:"forgetUrl"`
    AffiliationUrl       string         `xorm:"varchar(100)" json:"affiliationUrl"`
    TermsOfUse           string         `xorm:"varchar(100)" json:"termsOfUse"`
    SignupHtml           string         `xorm:"mediumtext" json:"signupHtml"`
    SigninHtml           string         `xorm:"mediumtext" json:"signinHtml"`
}
```

每个应用程序都可以有自己定制的注册页面、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"`
```

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 1.21+
- Node.js LTS (20)
- 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。 Based on Xorm Drivers Support, Casdoor currently provides support for the following databases:

- `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 Cassdoor 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 can be configured via a single file: conf/app.conf. As a beginner, you only need to modify `driverName` and `dataSourceName` based on your database (see Configure Database). Below is a complete reference of all configuration options:

Parameter	Default Value	Description
`appname`	`casdoor`	Application name (currently has no practical use)
`httpport`	`8000`	Port that the backend application listens on
`runmode`	`dev`	Running mode: `dev` or `prod`
`copyrequestbody`	`true`	Whether to copy request body for later use
`driverName`	`mysql`	Database driver (e.g., `mysql`, `postgres`, `sqlite`). See Configure Database
`dataSourceName`	`root:123456@tcp(localhost:3306)/`	Database connection string. See Configure Database
`dbName`	`casdoor`	Database name used by Casdoor
`tableNamePrefix`	(empty)	Prefix for table names when using an adapter
`showSql`	`false`	Show SQL statements in logger when log level is greater than INFO
`redisEndpoint`	(empty)	Redis endpoint for session storage (e.g., `localhost:6379`). If empty, sessions are stored locally in `./tmp`. For password: `host:port,db,password`
`defaultStorageProvider`	(empty)	Default storage provider name for file uploads (e.g., avatars). See storage

Parameter	Default Value	Description
`isCloudIntranet`	`false`	Whether provider endpoints use intranet addresses
`authState`	`"casdoor"`	Authorization application name checked during login
`socks5Proxy`	`"127.0.0.1:10808"`	SOCKS5 proxy address for OAuth providers (Google, GitHub, etc.) that may be blocked
`verificationCodeTimeout`	`10`	Verification code expiration time in minutes
`initScore`	`0`	Initial score assigned to new users (used by Casnode, not Casdoor)
`logPostOnly`	`true`	Whether to log only POST requests
`isUsernameLowered`	`false`	Whether to convert usernames to lowercase
`origin`	(empty)	Backend origin URL (e.g., `https://door.casdoor.com`)
`originFrontend`	(empty)	Frontend origin URL if different from backend
`staticBaseUrl`	`"https://cdn.casbin.org"`	CDN URL for static assets used during database initialization
`isDemoMode`	`false`	Enable demo mode restrictions
`batchSize`	`100`	Batch size for bulk

Parameter	Default Value	Description
		operations
`enableErrorMask`	`false`	Whether to mask detailed error messages
`enableGzip`	`true`	Accept and respond with gzip encoding when client supports it
`inactiveTimeoutMinutes`	(empty)	Auto-logout timeout in minutes. Empty or ≤0 means no timeout
`ldapServerPort`	`389`	Port for LDAP server
`ldapsCertId`	`""`	Certificate ID for LDAPS connections
`ldapsServerPort`	`636`	Port for LDAPS (LDAP over SSL) server
`radiusServerPort`	`1812`	Port for RADIUS server
`radiusDefaultOrganization`	`"built-in"`	Default organization for RADIUS authentication
`radiusSecret`	`"secret"`	Shared secret for RADIUS authentication
`quota`	`{"organization": -1, "user": -1, "application": -1, "provider": -1}`	Resource quotas (-1 means unlimited)
`logConfig`	`{"adapter":"file", "filename": "logs/casdoor.log", "maxdays":99999, "perm":"0770"}`	Logging configuration (adapter, file path, rotation, permissions)
`initDataNewOnly`	`false`	Whether to initialize data

Parameter	Default Value	Description
		only for new installations
`initDataFile`	`"./init_data.json"`	Path to data initialization file. See Data Initialization
`frontendBaseDir`	`"../cc_0"`	Base directory for frontend files (only for development)

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",
```

(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
```

> **备注**
>
> The default user of Casdoor has a uid and gid of 1000. When using volume mapping with SQLite (or any storage requiring file permissions), ensure that

> the path (e.g., `/folder/of/app.conf` in the example above) is accessible to uid 1000. This avoids permission errors like `permission denied` when Casdoor writes to mapped volumes.

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

Install the Casdoor chart:

```
helm install casdoor oci://registry-1.docker.io/casbin/casdoor-helm-charts --version v1.702.0
```

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
`replicaCount`	运行 Casdoor 应用的副本数量。	`1`
`image.repository`	Casdoor Docker 图像的仓库。	`casbin`

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

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

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

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

API Response Language

Casdoor APIs support internationalized responses. The default response language is English. To receive error messages and other text content in your preferred language, include the `Accept-Language` header in your API requests:

```
# Example: Get error messages in French
curl -X GET https://door.casdoor.com/api/get-account \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN" \
  -H "Accept-Language: fr"
```

Supported language codes include `en`, `zh`, `es`, `fr`, `de`, `ja`, `ko`, and more. For a complete list and more details, see the Internationalization documentation.

Machine-to-Machine (M2M) Authentication

Machine-to-machine (M2M) authentication is designed for scenarios where services, applications, or backend systems need to authenticate and communicate with APIs **without user interaction**. This is particularly useful for:

- **Backend services** calling Casdoor APIs programmatically
- **CLI tools** that need to interact with your APIs using access tokens
- **B2B enterprise scenarios** where organizations need to generate tokens for API access (e.g., admin tokens for management operations, read tokens for data access)
- **Automated processes** such as scheduled jobs, data synchronization, or system integrations
- **Microservices** that need to authenticate with each other

Casdoor supports M2M authentication through the following methods:

1. **Client Credentials Grant (OAuth 2.0)**: The recommended approach for M2M scenarios. This uses the Client Credentials Grant flow where applications authenticate using their `Client ID` and `Client Secret` to obtain access tokens. See the OAuth Client Credentials Grant documentation for details on obtaining tokens via this flow.

2. **Direct API Authentication with Client ID and Secret**: Use the application's credentials directly in API calls (see method #2 below).

Use Cases for M2M Authentication

- **Organization-level API access**: In B2B scenarios, you can create a Casdoor application for each organization. The application's client credentials provide admin-level permissions for that organization, enabling them to manage their users, generate tokens, and access organizational resources independently.

- **Token generation for downstream services**: Generate access tokens programmatically (using Client Credentials Grant) that can be distributed to CLI tools, read-only services, or other applications that need scoped access to your APIs.

- **Service-to-service authentication**: Backend services can authenticate as an "application" rather than as a user, with permissions equivalent to the organization admin.

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。

```go
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
    }
```

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. By `Client ID` and `Client secret` (Machine-to-Machine)

This method is the primary approach for **machine-to-machine (M2M) authentication**. It allows applications, services, or backend systems to authenticate with Casdoor APIs without any user interaction.

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。 This authentication method is useful when you want to call the API as a "machine", "application", or a "service" instead of a user. The permissions for the API calls will be the same as the application (equivalent to the admin of the organization).

Use cases

- **Service authentication**: Backend services calling Casdoor APIs programmatically
- **Organization management**: In B2B scenarios, create an application per organization to enable them to manage users and generate tokens independently
- **Token generation**: Obtain access tokens via the OAuth Client Credentials Grant flow for distribution to CLI tools or other services

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。

Obtaining access tokens with Client Credentials

For machine-to-machine scenarios where you need to obtain an access token (rather than using client credentials directly), use the **OAuth 2.0 Client Credentials Grant** flow:

1. Make a POST request to `https://<CASDOOR_HOST>/api/login/oauth/access_token` with:

```
{
    "grant_type": "client_credentials",
    "client_id": "YOUR_CLIENT_ID",
    "client_secret": "YOUR_CLIENT_SECRET"
}
```

2. You will receive an access token response:

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

3. Use the `access_token` to call Casdoor APIs (see method #1 above).

For more details, see the Client Credentials Grant documentation.

> **信息**
>
> **For B2B Enterprises**: You can create separate Casdoor applications for each of your customer organizations. Each application has its own `client_id` and `client_secret`, which your customers can use to:
>
> - Authenticate as their organization (with admin privileges)
> - Generate access tokens for their users or services

> - Manage their organization's users and permissions independently
> - Integrate your APIs into their systems without UI-based login flows
>
> This approach allows you to delegate organization management to your customers while maintaining security and isolation between different organizations.

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. 在账户设置页面中创建一对 accessKey 和 accessSecret。
2. 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`

```
curl --location 'http://door.casdoor.com/api/user?accessKey=b86db9dc-6bd7-4997-935c-af480dd2c796&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`

SSO Logout

The SSO logout endpoint `/api/sso-logout` allows you to log out a user from all applications in an organization simultaneously. When called, this endpoint will:

- Delete all active sessions for the user across all applications
- Expire all access tokens issued to the user
- Clear the current session and token

This is particularly useful when you need to ensure a user is completely logged out from all services, such as during security incidents or when implementing organization-wide logout policies.

Endpoint

```
GET or POST /api/sso-logout
```

Authentication

This endpoint requires the user to be authenticated. You can use any of the authentication methods described in the How to authenticate section above.

Example Request

```
# Using access token
curl -X POST https://door.casdoor.com/api/sso-logout \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN"

# Using session cookie
curl -X POST https://door.casdoor.com/api/sso-logout \
  --cookie "casdoor_session_id=abc123def456"
```

Response

```
{
  "status": "ok",
  "msg": "",
  "data": ""
}
```

CORS (Cross-Origin Resource Sharing)

Casdoor implements flexible CORS handling to allow secure cross-origin API requests. The server validates the `Origin` header and returns appropriate `Access-Control-Allow-Origin` headers based on the following rules:

Allowed origins:

- Origins matching your application's **redirect URIs** (configured in the application settings)
- Origins matching the Casdoor server's own hostname
- The configured origin in Casdoor's settings
- Special endpoint exceptions: requests to `/api/login/oauth/access_token` and `/api/userinfo` endpoints, and requests with origin `appleid.apple.com`

How it works:

When you make a cross-origin API request, Casdoor validates the origin through multiple checks: localhost/intranet addresses, matching redirect URIs in any application, matching the server's hostname, or configured origins. If any validation passes, the server includes CORS headers in the response allowing the request. For preflight `OPTIONS` requests, Casdoor returns appropriate headers including allowed methods (`POST`, `GET`, `OPTIONS`, `DELETE`) and credentials support.

Configuration:

To enable CORS for your application, add your frontend's origin to the application's **Redirect URIs** in the Casdoor admin panel. This allows your application to make authenticated API calls from the browser.

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)
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
Cloudreve (Go)	中文	Modify Cloudreve to support Casdoor
KodExplorer (PHP)	中文	Modify KodExplorer to support Casdoor

Deployment

Deploying to Docker

Deploying Casdoor with Docker

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

Deploying to Docker

In this chapter, you will learn how to deploy Casdoor using Docker and Docker Compose with various reverse proxy configurations.

Prerequisites

Before starting, ensure you have:

- Docker installed on your system
- Docker Compose installed (for compose method)
- A domain name pointing to your server (for reverse proxy configurations)

Docker Deployment Methods

Choose your preferred deployment method:

Docker Compose | Docker Run

Using Docker Compose

Create a `docker-compose.yml` file:

```yaml
services:
  casdoor:
    image: casbin/casdoor:latest
    container_name: casdoor
    restart: unless-stopped
```

Start the service:

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

Using Docker Run

Run Casdoor directly with Docker:

```
docker run -d \
  --name casdoor \
  --restart unless-stopped \
  -p 8000:8000 \
  -v $(pwd)/conf:/conf \
  -v $(pwd)/logs:/logs \
  -e GIN_MODE=release \
  casbin/casdoor:latest
```

Reverse Proxy Configuration

For production deployments, it's recommended to use a reverse proxy with TLS certificates. Choose your preferred reverse proxy:

Traefik (Labels) | Traefik (Dynamic) | Nginx | Caddy

Traefik with Docker Labels

Create a `docker-compose.yml` with Traefik labels:

```
services:
  traefik:
    image: traefik:v2.10
    container_name: traefik
```

Create the acme.json file:

```
touch traefik/acme.json
chmod 600 traefik/acme.json
```

Traefik with Dynamic Configuration

Create a `docker-compose.yml` with dynamic configuration:

```yaml
services:
  traefik:
    image: traefik:v2.10
    container_name: traefik
    restart: unless-stopped
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - ./traefik/acme.json:/acme.json
      - ./traefik/traefik.yml:/etc/traefik/traefik.yml
      - ./traefik/dynamic.yml:/etc/traefik/dynamic.yml
    command:
      - --configfile=/etc/traefik/traefik.yml
    networks:
      - casdoor-network

  casdoor:
    image: casbin/casdoor:latest
    container_name: casdoor
    restart: unless-stopped
    environment:
      - GIN_MODE=release
    volumes:
      - ./conf:/conf
      - ./logs:/logs
    networks:
```

Create `traefik/traefik.yml`:

```yaml
api:
  dashboard: true

entryPoints:
  web:
    address: ":80"
    http:
      redirections:
        entrypoint:
          to: websecure
          scheme: https
  websecure:
    address: ":443"

providers:
  docker:
    endpoint: "unix:///var/run/docker.sock"
    exposedByDefault: false
  file:
    directory: /etc/traefik
    watch: true

certificatesResolvers:
  letsencrypt:
    acme:
      email: your-email@example.com
      storage: /acme.json
      httpChallenge:
        entryPoint: web
```

Create `traefik/dynamic.yml`:

```yaml
http:
  routers:
```

Nginx with Let's Encrypt

Create a `docker-compose.yml` with Nginx:

```yaml
services:
  nginx:
    image: nginx:alpine
    container_name: nginx
    restart: unless-stopped
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf
      - ./nginx/conf.d:/etc/nginx/conf.d
      - ./certbot/conf:/etc/letsencrypt
      - ./certbot/www:/var/www/certbot
    depends_on:
      - casdoor
    networks:
      - casdoor-network

  certbot:
    image: certbot/certbot
    container_name: certbot
    volumes:
      - ./certbot/conf:/etc/letsencrypt
      - ./certbot/www:/var/www/certbot
    command: certonly --webroot --webroot-path=/var/www/certbot
--email your-email@example.com --agree-tos --no-eff-email -d
your-domain.com

  casdoor:
    image: casbin/casdoor:latest
    container_name: casdoor
    restart: unless-stopped
    environment:
      - GIN_MODE=release
```

Create `nginx/conf.d/default.conf`:

```
server {
    listen 80;
    server_name your-domain.com;

    location /.well-known/acme-challenge/ {
        root /var/www/certbot;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}

server {
    listen 443 ssl;
    server_name your-domain.com;

    ssl_certificate /etc/letsencrypt/live/your-domain.com/
fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/your-domain.com/
privkey.pem;

    location / {
        proxy_pass http://casdoor:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Initialize SSL certificates:

```
# Create directories
```

Caddy with Automatic HTTPS

Create a `docker-compose.yml` with Caddy:

```yaml
services:
  caddy:
    image: caddy:2-alpine
    container_name: caddy
    restart: unless-stopped
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./caddy/Caddyfile:/etc/caddy/Caddyfile
      - caddy_data:/data
      - caddy_config:/config
    depends_on:
      - casdoor
    networks:
      - casdoor-network

  casdoor:
    image: casbin/casdoor:latest
    container_name: casdoor
    restart: unless-stopped
    environment:
      - GIN_MODE=release
    volumes:
      - ./conf:/conf
      - ./logs:/logs
    networks:
      - casdoor-network

volumes:
  caddy_data:
  caddy_config:

networks:
```

Create `caddy/Caddyfile`:

```
your-domain.com {
    reverse_proxy casdoor:8000 {
        header_up Host {host}
        header_up X-Real-IP {remote}
        header_up X-Forwarded-For {remote}
        header_up X-Forwarded-Proto {scheme}
    }
}
```

Configuration

Create the necessary configuration directories

```
mkdir -p conf logs
```

Download the Casdoor configuration files

```
wget https://raw.githubusercontent.com/casdoor/casdoor/master/conf/
app.conf -O conf/app.conf
wget https://raw.githubusercontent.com/casdoor/casdoor/master/
init_data.json.template -O conf/init_data.json
```

Edit `conf/app.conf` to match your environment settings

> **备注**
>
> The default user of Casdoor has a uid and gid of 1000. When using volume

> mapping with SQLite (or any storage requiring file permissions), ensure that the path (e.g., `/folder/of/app.conf` in the example above) is accessible to uid 1000. This avoids permission errors like `permission denied` when Casdoor writes to mapped volumes.

Testing

After deployment, visit your domain in your browser:

- **Docker Run/Compose only:** `http://your-server-ip:8000`
- **With Reverse Proxy:** `https://your-domain.com`

Troubleshooting

Check container logs

```
# For docker-compose
docker-compose logs casdoor

# For docker run
docker logs casdoor
```

Verify reverse proxy configuration

```
# Check if containers are running
docker ps

# Test connectivity
curl -I http://localhost:8000
```

SSL Certificate Issues

- Ensure your domain points to the correct server IP
- Check that ports 80 and 443 are open in your firewall
- Verify DNS propagation with `nslookup your-domain.com`

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：

```yaml
# this is only an EXAMPLE of deploying casddor in kubernetes
# please modify this file according to your requirements
apiVersion: v1
kind: Service
metadata:
  #EDIT IT: if you don't want to run casdoor in default namespace,
please modify this field
  #namespace: casdoor
  name: casdoor-svc
```

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。

这里提供一个导入或导出配置数据的教程。

导入配置数据

By default, if there is a configuration file named `init_data.json` at the root directory of Casdoor, it will be used to initialize data in Casdoor. You can also specify a custom path for the initialization file by setting the `initDataFile` parameter in `conf/app.conf`:

```
initDataFile = /path/to/your/init_data.json
```

If no custom path is specified, Casdoor will look for `init_data.json` in the root directory where Casdoor runs.

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。

`init_data.json` 可参照模板： init_data.json.template 在使用之前将它重命名为 `init_data.json`

对于 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
```

对于 Kubernetes：

如果你使用Kubernetes部署Casdoor，你可以使用`configmap`来存储`init_data.json`。

```yaml
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`：

```yaml
apiVersion: apps/v1
kind: Deployment
...
spec:
  template:
    ...
    spec:
      containers:
      ...
```

导出配置数据

You can export all Casdoor configuration data to a file for backup or migration purposes. There are two methods available:

Using the Binary (Recommended)

If you're running Casdoor from a binary, use the `-export` flag to dump the database to a JSON file:

```
# Export to default location (init_data_dump.json)
./casdoor -export

# Export to a custom path
./casdoor -export -exportPath /path/to/backup.json
```

The export runs after database initialization but before the server starts, then exits automatically. This method works with any deployment method (binary, Docker, Kubernetes) and doesn't require Go toolchain or source code access.

Using Go Test

If you have access to the source code, you can use the test method:

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

This will generate `init_data_dump.json` in the same directory.

Migrating Data

After exporting, rename `init_data_dump.json` to `init_data.json` and place it in the root directory of your target Casdoor installation. On startup, Casdoor will automatically import the data.

引用

数据初始化支持的所有 Casdoor 对象如下:

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	doc
提供商	struct	doc
ldaps	struct	文档
models	struct	
permissions	struct	doc
payments	struct	文档

对象	Go 结构体	文档
products	struct	文档
resources	struct	文档
roles	struct	文档
syncers	struct	文档
tokens	struct	文档
webhooks	struct	文档
groups	struct	文档
adapters	struct	文档
enforcers	struct	
plans	struct	doc
pricings	struct	文档
invitations	struct	文档
records	struct	
sessions	struct	
subscriptions	struct	doc

对象	Go 结构体	文档
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，你需要修改`GetProvider()`中的`id`参数。 提供者 `id` 的格式是 `<owner>/<name>`。

```go
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。 确保你可以访问这些资源。

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。 To address this, xorm provides a migrate package that can assist with this problem.

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.DropTables(&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

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

Casdoor Authenticator App

Store TOTP code in Casdoor

📄 如何启用单点登录

启用单点登录

📄 Single Sign-Out (SSO Logout)

Implement Single Sign-Out to log users out from all applications simultaneously

📄 Vue SDK

Casdoor Vue SDK

🗃️ 桌面 SDK

4 个项目

🗃️ 移动 SDKs

1 个项目

Casdoor CLI

Using Casdoor's official command-line interface for managing users, groups, and permissions

Casdoor插件

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

Chrome Extension

Using Casdoor in Chrome extension

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

📄 Guest Authentication

Create temporary users without credentials

📄 使用Casdoor作为CAS服务器

如何将Casdoor用作CAS服务器

🗃️ SAML

6 个项目

📄 面部识别

使用 Face ID 登录到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 Cassdoor:

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。

Global OIDC Endpoint

Casdoor's global OIDC discovery URL is:

```
<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",
```

Application-Specific OIDC Endpoints

Besides the global discovery endpoint, you can use application-specific OIDC discovery endpoints. Each application gets its own isolated OIDC configuration with a unique issuer. This comes in handy when running multi-tenant deployments where applications need their own certificates or when you want to gradually migrate applications without affecting others.

The application-specific discovery URL follows this pattern:

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

For example, if you have an application named `app-example`:

```
https://door.casdoor.com/.well-known/app-example/openid-configuration
```

The main difference is that the `issuer` and `jwks_uri` fields in the discovery response contain the application path. The `issuer` becomes `https://door.casdoor.com/.well-known/app-example` instead of just `https://door.casdoor.com`, and the `jwks_uri` points to `/.well-known/app-example/jwks`. Everything else, including the authorization and token endpoints, stays the same.

You can also access the JWKS and WebFinger endpoints for each application:

```
<your-casdoor-backend-host>/.well-known/<application-name>/jwks
<your-casdoor-backend-host>/.well-known/<application-name>/webfinger
```

The JWKS endpoint returns the public keys for verifying tokens. When an application has its own certificate configured, that certificate is used. Otherwise, it falls back to the global certificates.

Here's what the responses look like. The global endpoint returns:

```
{
  "issuer": "https://door.casdoor.com",
  "jwks_uri": "https://door.casdoor.com/.well-known/jwks",
  "authorization_endpoint": "https://door.casdoor.com/login/oauth/
authorize",
  ...
}
```

While the application-specific endpoint for `app-example` returns:

```
{
  "issuer": "https://door.casdoor.com/.well-known/app-example",
  "jwks_uri": "https://door.casdoor.com/.well-known/app-example/
jwks",
  "authorization_endpoint": "https://door.casdoor.com/login/oauth/
authorize",
  ...
}
```

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
originBackend	iss
Aud	aud
Name	preferred_username

Casdoor 用户字段	OIDC用户信息字段
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Cassdoor 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	适用于 Google Firebase 应用		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

```go
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 `https://door.casdoor.com` 或 `http://localhost:8000`
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。

之后，您可以在应用编辑页面选择证书。

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Cassdoor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Cassdoor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Cassdoor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。

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

Casdoor Authenticator App

Overview

Casdoor-Authenticator (https://app.casdoor.org/) is an open-source authenticator App (source code: https://github.com/casdoor/casdoor-authenticator) like Google Authenticator, Microsoft Authenticator or Authy. It provides Multi-Factor Authentication (MFA) with TOTP for both iOS and Android. This app allows users to store Time-based One-Time Password (TOTP) securely in the App, which is cloud-synced with Casdoor, offering a comprehensive solution for managing Two-Factor Authentication (2FA) needs directly from mobile devices.

Key Features

- **Multi-Factor Authentication (MFA)**: Generate secure TOTP-based codes for enhanced account protection.
- **Offline mode**: Generate TOTP codes without an internet connection.
- **Account synchronization**: Securely sync your accounts across multiple devices.
- **Privacy-focused**: All data is encrypted and stored securely.
- **Friendly UI**: Simple and intuitive design for easy navigation.

Android

22:21
84
Casdoor
admin
Search
Google:test/admin
192 592
7s
Github:test/admin
472 889
7s
Amazon:test/admin
124 416
7s
slack:test
835 125
7s

What is TOTP?

TOTP stands for Time-based One-Time Passwords and is a common form of two-factor authentication (2FA). Unique numeric passwords are generated with a standardized algorithm that uses the current time as an input. These time-based passwords:

- Change every 30 seconds for enhanced security
- Are available offline
- Are widely supported by many services and apps
- Provide user-friendly, increased account security as a second factor

How to use the Casdoor Authenticator App?

Step 0: Install the Casdoor Authenticator App

Casdoor-Authenticator is currently available for Android devices. You can download the app from the following sources:

1. Homepage: https://app.casdoor.org
2. Source code: https://github.com/casdoor/casdoor-authenticator
3. Binary release: https://github.com/casdoor/casdoor-authenticator/releases

For developers interested in building the app from source, you can find the source code and build instructions in the Casdoor Authenticator GitHub Repository.

Step 1: Enable Totp Account storage in Casdoor Server

(Optional)

This setup is optional for users who want to store their TOTP codes in the Casdoor server. Before using the Casdoor Authenticator App, you need to make sure that the MFA accounts setting is enabled in the Casdoor server.

Step 2: Log in to the Casdoor Authenticator App

After installing the Casdoor Authenticator App and enabling the MFA accounts setting in the Casdoor server, you have three convenient ways to connect to your Casdoor server:

1. Manual Configuration
    - Tap "Enter Server Manually"
    - Enter your server URL, client ID, and organization name
    - Log in with your Casdoor account credentials
2. QR Code Scan
    - Tap "Scan QR Code"
    - Use your device's camera to scan the QR code from your Casdoor server
    - The QR code can be found in "My Account" → "MFA accounts" section

    - The app will automatically configure the connection

3. Demo Server
    - Tap "Try Demo Server" to connect to a pre-configured demo instance
    - Useful for testing the app's functionality without setting up your own server

Once connected, you can view your TOTP codes and manage your 2FA settings directly from the Casdoor Authenticator App like other authenticator apps.

Migration from Other Authenticator Apps

Migration from Google Authenticator

Select the "Transfer Accounts" option in the menu of Google Authenticator and choose the accounts you want to transfer. Then, click the "Export" button to generate a QR code.

Google Authenticator
Transfer accounts
How it works
Settings
Send feedback
Help

20:23
4G
Back
Transfer accounts
You can transfer accounts to a new device that has Google Authenticator.
Export accounts
Create a QR code to export your accounts
Import accounts
Scan a QR code to bring new accounts in

20:23
4G
Transfer accounts
Cancel
Select accounts
To export, we'll create a QR code that stores the accounts you select below.
Test
Test2

20:23
4G
Select accounts
Cancel
Scan this QR code
Download the Google Authenticator app on your new device. Within the app, scan this QR code.

In the Casdoor Authenticator App, scan the QR code generated by Google Authenticator to import your TOTP data. The app will automatically add the accounts to your Casdoor Authenticator App, allowing you to manage your TOTP codes securely.

22:12
86
Casdoor
Login
Search
Test
145 112
12s

Migration from Microsoft Authenticator (requires root access)

If you are using Microsoft Authenticator, you can migrate your TOTP data to the Casdoor Authenticator App using the Microsoft Authenticator's database backup. Here's how:

1. On your Android device with Microsoft Authenticator installed, locate the app's data directory: `/data/data/com.azure.authenticator/databases/`
2. You will need root access to extract the `PhoneFactor` database file.
3. In the Casdoor Authenticator App, go to the import section
4. Select "Import from Microsoft Authenticator"
5. Choose the extracted `PhoneFactor` database file
6. The app will automatically import all your TOTP accounts from Microsoft Authenticator

20:48
4G
45
Casdoor
admin
Search
example@gmail.com
608 508
26s
@twitter
526 204
26s
amazon-user
526 204
26s
microsoft
145 348
26s
slack
241 907
26s
Home
Settings

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。 你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。

Casdoor
Home
Organizations
Users
Roles
Permissions
Models
Adapters
Providers
Applications
Edit Application
Save
Save & Exit
Name :
app-casbin-oa
Display name :
Casbin OA
Logo :
URL :
https://cdn.casbin.org/img/casbin_logo_1024x256.png
Preview:
casbin
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，我们通过在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。 我们通过在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。

SSO Logout

When using SSO, you might need to log a user out from all applications simultaneously. Casdoor provides an SSO logout endpoint that terminates all active sessions and expires all tokens for a user across all applications in the organization.

To implement SSO logout in your application, make a request to the `/api/sso-logout` endpoint. This endpoint will ensure the user is completely logged out from all integrated applications. For detailed information about the SSO logout API, including authentication methods and request examples, see the SSO Logout section in the Public API documentation.

Single Sign-Out (SSO Logout)

Introduction

Single Sign-Out (SSO Logout) is a feature that allows you to log a user out from all applications in an organization simultaneously. When a user logs out from one application, they are automatically logged out from all other applications that are part of the same SSO ecosystem.

This is particularly useful in scenarios such as:

- **Security incidents**: Immediately terminate all active sessions when a security breach is detected
- **Organization-wide logout policies**: Enforce logout across all services when users leave the organization or change roles
- **Compliance requirements**: Ensure users are completely logged out from all systems when required by regulations
- **User-initiated logout**: Allow users to log out from all applications with a single action

How SSO Logout Works

When the SSO logout endpoint is called, Casdoor performs the following actions:

1. **Delete all active sessions**: All active sessions for the user across all applications in the organization are terminated

2. **Expire all access tokens**: All access tokens that were issued to the user are immediately invalidated
3. **Clear the current session**: The user's current session and authentication state are cleared

This ensures that the user is completely logged out from all integrated applications and cannot access any resources without re-authenticating.

SSO Logout API

Endpoint

```
GET or POST /api/sso-logout
```

The SSO logout endpoint accepts both `GET` and `POST` requests, making it flexible for different integration scenarios.

Authentication

This endpoint requires the user to be authenticated. You can use any of the authentication methods supported by Casdoor:

- **Access token**: Include the access token in the `Authorization` header
- **Session cookie**: Use the session cookie that was set during login
- **Client credentials**: Use the application's client ID and secret for machine-to-machine scenarios

For more details on authentication methods, see the Casdoor Public API documentation.

Request Examples

Using Access Token

```
curl -X POST https://door.casdoor.com/api/sso-logout \
  -H "Authorization: Bearer YOUR_ACCESS_TOKEN"
```

Using Session Cookie

```
curl -X POST https://door.casdoor.com/api/sso-logout \
  --cookie "casdoor_session_id=abc123def456"
```

Using JavaScript Fetch API

```
fetch('https://door.casdoor.com/api/sso-logout', {
  method: 'POST',
  headers: {
    'Authorization': `Bearer ${accessToken}`
  },
  credentials: 'include'
})
  .then(response => response.json())
  .then(data => {
    console.log('Logout successful:', data);
    // Redirect to login page or home page
    window.location.href = '/login';
  })
  .catch(error => {
    console.error('Logout failed:', error);
  });
```

Response

The API returns a standard Casdoor response:

```
{
  "status": "ok",
  "msg": "",
  "data": ""
}
```

Response Fields:

- `status`: Indicates whether the operation was successful (`"ok"`) or failed (`"error"`)
- `msg`: Contains an error message if the operation failed, otherwise empty
- `data`: Additional data returned by the API, typically empty for logout operations

Implementing SSO Logout in Your Application

Using Casdoor SDKs

Most Casdoor SDKs provide built-in support for SSO logout. Here are examples for different platforms:

Go SDK

```go
import "github.com/casdoor/casdoor-go-sdk/casdoorsdk"

func logout(w http.ResponseWriter, r *http.Request) {
    // Get the access token from the session or request
    token := getAccessTokenFromSession(r)

    // Call the SSO logout endpoint
    err := casdoorsdk.Logout(token)
    if err != nil {
        http.Error(w, "Logout failed",
http.StatusInternalServerError)
        return
    }

    // Clear local session
    clearSession(w, r)

    // Redirect to login page
    http.Redirect(w, r, "/login", http.StatusFound)
}
```

JavaScript SDK

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

const CasdoorSDK = new Sdk({
  serverUrl: "https://door.casdoor.com",
  clientId: "YOUR_CLIENT_ID",
  appName: "YOUR_APP_NAME",
  organizationName: "YOUR_ORG_NAME",
});

async function handleLogout() {
```

Python SDK

```python
from casdoor import CasdoorSDK

sdk = CasdoorSDK(
    endpoint="https://door.casdoor.com",
    client_id="YOUR_CLIENT_ID",
    client_secret="YOUR_CLIENT_SECRET",
    certificate="YOUR_CERT",
    org_name="YOUR_ORG_NAME",
    app_name="YOUR_APP_NAME",
)

def logout(access_token):
    try:
        # Call the SSO logout endpoint
        result = sdk.logout(access_token)

        if result['status'] == 'ok':
            # Clear local session
            clear_session()
            return True
        else:
            print(f"Logout failed: {result['msg']}")
            return False
    except Exception as e:
        print(f"Logout error: {e}")
        return False
```

Manual Implementation

If you're not using a Casdoor SDK, you can implement SSO logout manually by making an HTTP request to the logout endpoint:

```
async function logout() {
  const accessToken = localStorage.getItem('access_token');

  try {
    // Use the full Casdoor server URL in your application
    const response = await fetch('https://door.casdoor.com/api/sso-
logout', {
      method: 'POST',
      headers: {
        'Authorization': `Bearer ${accessToken}`,
        'Content-Type': 'application/json'
      },
      credentials: 'include'
    });

    const result = await response.json();

    if (result.status === 'ok') {
      // Clear local storage
      localStorage.removeItem('access_token');
      localStorage.removeItem('user_info');
      sessionStorage.clear();

      // Redirect to login page
      window.location.href = '/login';
    } else {
      console.error('Logout failed:', result.msg);
    }
  } catch (error) {
    console.error('Logout error:', error);
  }
}
```

Best Practices

1. Clear Local State

After calling the SSO logout endpoint, make sure to clear all local authentication state:

- Remove access tokens from local storage or session storage
- Clear any cached user information
- Invalidate any local session cookies
- Reset application state to the logged-out state

2. Handle Logout Errors Gracefully

Even if the SSO logout endpoint fails, you should still clear local authentication state to ensure the user appears logged out from your application:

```
async function logout() {
  try {
    await callSSOLogout();
  } catch (error) {
    console.error('SSO logout failed, clearing local state
anyway:', error);
  } finally {
    // Always clear local state
    clearLocalAuthenticationState();
    redirectToLogin();
  }
}
```

3. Provide User Feedback

Give users clear feedback about the logout process:

```
async function logout() {
  // Show loading indicator
  showLoadingIndicator('Logging out...');

  try {
    await callSSOLogout();
    showSuccessMessage('Logged out successfully');
  } catch (error) {
    showErrorMessage('Logout failed, but you have been logged out
locally');
  } finally {
    clearLocalAuthenticationState();
    // Redirect after a short delay to allow users to see the
message
    setTimeout(() => redirectToLogin(), 1000);
  }
}
```

4. Implement Logout Timeout

Set a reasonable timeout for the logout request to prevent users from waiting indefinitely:

```
async function logout() {
  const timeout = 5000; // 5 seconds

  try {
    await Promise.race([
      callSSOLogout(),
```

Security Considerations

1. Secure Communication

Always use HTTPS when calling the SSO logout endpoint to prevent token interception:

```
// ✅ Good: Uses HTTPS
const logoutUrl = 'https://door.casdoor.com/api/sso-logout';

// ❌ Bad: Uses HTTP (insecure)
const logoutUrl = 'http://door.casdoor.com/api/sso-logout';
```

2. Token Validation

Casdoor validates the access token before processing the logout request. Ensure your token is valid and has not expired before making the logout call.

3. CSRF Protection

When using session cookies for authentication, ensure CSRF protection is enabled to prevent unauthorized logout requests:

```
// Include CSRF token in the request headers
// Replace with your actual Casdoor server URL
fetch('https://door.casdoor.com/api/sso-logout', {
  method: 'POST',
  headers: {
    'X-CSRF-Token': getCsrfToken()
  },
```

4. Audit Logging

Consider logging logout events for security auditing and compliance:

```go
func logout(userID string, token string) error {
    // Call SSO logout
    err := casdoorsdk.Logout(token)

    // Log the logout event
    auditLog.Info(map[string]interface{}{
        "event": "sso_logout",
        "user_id": userID,
        "timestamp": time.Now(),
        "success": err == nil,
    })

    return err
}
```

Troubleshooting

Logout Not Working Across All Applications

If users remain logged in to some applications after SSO logout:

1. **Verify application integration**: Ensure all applications are properly integrated with Casdoor and use the same organization
2. **Check token validation**: Make sure all applications validate tokens on each request
3. **Review session management**: Applications should not rely solely on local sessions; they must validate tokens with Casdoor

Token Still Valid After Logout

If access tokens remain valid after logout:

1. **Verify the logout endpoint**: Ensure you're calling the correct endpoint (`/api/sso-logout`)
2. **Check authentication**: Make sure you're sending valid authentication credentials with the logout request
3. **Review token caching**: Ensure applications don't cache token validation results

Logout Endpoint Returns Error

Common error scenarios:

- **401 Unauthorized**: The access token is invalid or expired. Clear local state and redirect to login.
- **403 Forbidden**: The user doesn't have permission to logout. This is rare and may indicate a configuration issue.
- **500 Internal Server Error**: Server-side error. Log the error and clear local state anyway.

```
async function logout() {
  try {
    // Replace with your actual Casdoor server URL
    const response = await fetch('https://door.casdoor.com/api/sso-logout', {
      method: 'POST',
      headers: { 'Authorization': `Bearer ${token}` }
    });
```

Related Documentation

- Single Sign-On (SSO): Learn how to enable SSO for your applications
- Casdoor Public API: Complete API reference including authentication methods
- Tokens: Understand how Casdoor manages access tokens and sessions
- OAuth 2.0: Learn about OAuth 2.0 flows supported by Casdoor

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`/callback`。

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。

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

解析代码并获取用户信息

```
async function getUserInfo(clientId, clientSecret, code) {
  const { data } = await axios({
    method: "post",
    url: authCodeUrl,
    headers: {
      "content-type": "application/json",
    },
    data: JSON.stringify({
      grant_type: "authorization_code",
      client_id: clientId,
      client_secret: clientSecret,
      code: code,
    }),
  });
  const resp = await axios({
    method: "get",
    url: `${getUserInfoUrl}?accessToken=${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	`CasdoorVariables.cs`
Clientid	您的 Casdoor 应用程序的客户端 ID	`CasdoorVariables.cs`
AppName	您的Casdoor应用程序的名称	`CasdoorVariables.cs`
CallbackURL	您的Casdoor应用程序的回调URL路径。 如果未提供，它将是 `casdoor://callback`	`CasdoorVariables.cs`

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。

Login
casbin
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
Made with by Casdoor

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

Android Emulator - pixel_5_-_api_33:5554
21:06
.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	`mainwindow.h`
client_id	您的 Casdoor 应用程序的客户端 ID	`mainwindow.h`
client_secret	你的Casdoor应用的客户端密钥	`mainwindow.h`
certificate	Casdoor 应用程序证书的公钥	`mainwindow.h`
org_name	您的Casdoor应用程序的名称	`mainwindow.h`

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。

iOS	Android

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。

iOS	Android

你可以在下面的GIF图像中预览整个过程。

iOS	Android

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`/callback`
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 CLI

Casdoor CLI is the official command-line interface for Casdoor, providing a powerful and intuitive way to manage your Casdoor identity and access management system directly from the terminal.

GitHub repository: https://github.com/casdoor/casdoor-cli

Features

OAuth2 Browser-Based Authentication

The CLI uses a secure browser-based OAuth2 flow for authentication, ensuring your credentials are protected through Casdoor's standard authentication mechanism.

Secure Token Storage

Credentials are safely stored using your system's keyring interface (GNOME Keyring on Linux, Keychain on macOS), ensuring tokens never touch disk in plaintext.

User Management

Create, update, and delete users with ease directly from the command line.

Permission Management

Control user permissions through Casdoor's group feature with built-in roles:

- `lector`: Read-only access
- `editor`: Can create users, with limited modification rights
- `administrator`: Full control over user creation, modification, and deletion

Group Management

Create, modify, and delete user groups to organize users and manage permissions efficiently.

Installation

Prerequisites

- Go 1.22.0 or higher
- macOS or Linux operating system
- GNOME Keyring (Linux) or Keychain (macOS) for secure credential storage

> ⚠ **注意事项**
>
> **Platform Support**: Currently supports macOS and Linux (tested on Debian 12 and macOS Sonoma). Windows support via WSL is not available as the CLI requires GNOME's Secret Service DBus interface (GNOME Keyring) for secure credential storage.

macOS

```
make build TARGET_OS=darwin && make install TARGET_OS=darwin
```

Linux

```
make build TARGET_OS=linux && make install TARGET_OS=linux
```

Configure Your Shell

After installation, add `casdoor-cli` to your `PATH`:

For Bash users:

```
echo 'export PATH="/usr/local/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc
```

For Zsh users:

```
echo 'export PATH="/usr/local/bin:$PATH"' >> ~/.zshrc
source ~/.zshrc
```

Verify the installation:

```
casdoor --help
```

Configuration

Casdoor Server Setup

To use the CLI, you need a configured Casdoor application. You have two options:

1. **Using the provided bootstrap data:** An `init_data.json` file is included in the repository to quickly bootstrap Casdoor's configuration. Refer to the Data Initialization documentation for initialization instructions.

2. **Manual configuration:** Create an application directly in your Casdoor admin panel and configure it according to your requirements.

CLI Configuration

On first launch, `casdoor-cli` will prompt you to provide a `config.yaml` file containing your Casdoor connection details. See the included `config.yaml.example` file in the repository for reference.

Required configuration fields:

```yaml
application_name: your-app-name
casdoor_endpoint: https://your-casdoor-instance.com
certificate: |
  -----BEGIN CERTIFICATE-----
  Your certificate content here
  -----END CERTIFICATE-----
client_id: your-client-id
client_secret: your-client-secret
organization_name: your-organization
redirect_uri: http://localhost:9000/callback
```

Your configuration will be securely stored in `~/.casdoor-cli/config.yaml` (base64 encoded) for subsequent use.

Usage

Available Commands

```
Usage:
  casdoor [command]

Available Commands:
  completion  Generate the autocompletion script for the specified
shell
  groups      Manage Casdoor permissions
  help        Help about any command
  login       Login to your Casdoor account
  logout      Logout from your Casdoor account
  users       Manage Casdoor users

Flags:
  -d, --debug   verbose logging
  -h, --help    help for casdoor
```

Login

To authenticate with your Casdoor instance:

```
casdoor login
```

This will open your default browser for OAuth2 authentication.

Managing Users

```
# List users
casdoor users list

# Create a user
casdoor users create

# Update a user
casdoor users update

# Delete a user
casdoor users delete
```

Managing Groups

```
# List groups
casdoor groups list

# Create a group
casdoor groups create

# Update a group
casdoor groups update

# Delete a group
casdoor groups delete
```

Logout

To logout from your Casdoor account:

```
casdoor logout
```

Development

Local Development Environment

A Docker Compose environment is provided in the repository for local testing and development:

```
docker compose up -d
```

> **备注**
>
> Allow a few moments for the Casdoor container to fully initialize. The container will restart multiple times as it sets up the database.

Development Configuration

Create a `config.yaml` file from the provided `config.yaml.example` template at the repository root with your local development settings.

Testing the CLI

Test the login functionality with the default development credentials provided in the repository documentation.

Run directly with Go:

```
go run main.go login
```

Or build and install first:

```
make build TARGET_OS=darwin && make install TARGET_OS=darwin  # For macOS
# OR
make build TARGET_OS=linux && make install TARGET_OS=linux    # For Linux

casdoor login
```

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。

For command-line interface integration, see the Casdoor CLI documentation.

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
Chrome extension	JavaScript	https://github.com/casdoor/casdoor-chrome-extension

要获取官方Casdoor插件的完整列表，请访问Casdoor仓库。

Chrome Extension

casdoor-chrome-extension is an example of how to integrate Casdoor in a Chrome browser extension. We will guide you through the steps below.

Step 1: Deploy Casdoor

Firstly, Casdoor should be deployed.

You can refer to the Casdoor official documentation for the Server Installation. Please deploy your Casdoor instance in **production mode**.

After a successful deployment, make sure the following:

- Open your favorite browser and visit http://localhost:8000. You will see the login page of Casdoor.
- Test the login functionality by entering `admin` as the username and `123` as the password.

After that, you can quickly implement a Casdoor-based login page in your Chrome extension using the following steps.

Step 2: Configure Casdoor Application

Before using Casdoor for authentication in your Chrome extension, you need to configure a Casdoor application:

1. Go to your Casdoor instance and navigate to **Applications**.
2. Create a new application or use an existing one.
3. Configure the application settings:
   - Set the **Redirect URLs** to include your Chrome extension URL. For example: `https://<extension-id>.chromiumapp.org/` or `http://localhost:3000/callback` for development.
   - Note down the **Client ID** and **Client Secret**.
   - Make sure the application has proper OAuth settings enabled.

Step 3: Set Up Chrome Extension

1. Create Manifest File

Create a `manifest.json` file in your Chrome extension project with the necessary permissions:

```
{
  "manifest_version": 3,
  "name": "Casdoor Chrome Extension",
  "version": "1.0.0",
  "description": "Chrome extension integrated with Casdoor",
  "permissions": [
    "identity",
    "storage"
  ],
  "host_permissions": [
    "http://localhost:8000/*",
    "https://door.casdoor.com/*"
  ],
  "action": {
    "default_popup": "popup.html",
    "default_icon": {
      "16": "icons/icon16.png",
```

2. Configure Extension Identity

In the `manifest.json`, you may need to add OAuth2 configuration if using Chrome's identity API:

```
{
  "oauth2": {
    "client_id": "your-client-id.apps.googleusercontent.com",
    "scopes": ["openid", "profile", "email"]
  }
}
```

> ⚠ **注意事项**
>
> Replace the configuration values with your own Casdoor instance, especially the `client_id` and the host permissions URLs.

Step 4: Implement Authentication Flow

1. Create Background Script

Create a `background.js` file to handle the authentication:

```
const CASDOOR_ENDPOINT = "http://localhost:8000";
const CLIENT_ID = "your-client-id";
const CLIENT_SECRET = "your-client-secret";
const ORGANIZATION_NAME = "built-in";
const APPLICATION_NAME = "app-built-in";
const REDIRECT_URI = chrome.identity.getRedirectURL();

// Generate the authorization URL
function getAuthUrl() {
  const state = Math.random().toString(36).substring(7);
  const authUrl = `${CASDOOR_ENDPOINT}/login/oauth/
authorize?client_id=${CLIENT_ID}&response_type=code&redirect_uri=${encodeURIComponent(REDIRECT_URI)}&scope=openid%20profile%20email&state=

  chrome.storage.local.set({ oauthState: state });

  return authUrl;
}

// Handle OAuth callback
async function handleOAuthCallback(redirectUrl) {
  const url = new URL(redirectUrl);
  const code = url.searchParams.get('code');
  const state = url.searchParams.get('state');

  // Verify state
  const { oauthState } = await chrome.storage.local.get('oauthState');
  if (state !== oauthState) {
    throw new Error('Invalid state parameter');
  }

  // Exchange code for token
  const tokenResponse = await fetch(`${CASDOOR_ENDPOINT}/api/login/oauth/access_token`, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      grant_type: 'authorization_code',
      client_id: CLIENT_ID,
```

2. Create Popup HTML

Create a `popup.html` file for the extension popup:

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Casdoor Login</title>
  <style>
    body {
      width: 300px;
      padding: 20px;
      font-family: Arial, sans-serif;
    }
    button {
      width: 100%;
      padding: 10px;
      margin: 5px 0;
      cursor: pointer;
      background-color: #4285f4;
      color: white;
      border: none;
      border-radius: 4px;
    }
    button:hover {
      background-color: #357ae8;
    }
    #user-info {
      margin-top: 20px;
    }
  </style>
</head>
<body>
  <div id="login-section">
    <h2>Casdoor Login</h2>
    <button id="login-btn">Login with Casdoor</button>
  </div>

  <div id="user-section" style="display: none;">
    <h2>Welcome!</h2>
    <div id="user-info"></div>
    <button id="logout-btn">Logout</button>
  </div>

  <script src="popup.js"></script>
</body>
</html>
```

3. Create Popup Script

Create a `popup.js` file to handle user interactions:

```javascript
document.addEventListener('DOMContentLoaded', () => {
  const loginSection = document.getElementById('login-section');
  const userSection = document.getElementById('user-section');
  const loginBtn = document.getElementById('login-btn');
  const logoutBtn = document.getElementById('logout-btn');
  const userInfo = document.getElementById('user-info');

  // Check if user is already logged in
  chrome.runtime.sendMessage({ action: 'getUser' }, (response) => {
    if (response.user) {
```

Step 5: Load and Test the Extension

1. Open Chrome and navigate to `chrome://extensions/`.
2. Enable **Developer mode** in the top right corner.
3. Click **Load unpacked** and select your extension directory.
4. Click on the extension icon in the Chrome toolbar.
5. Click the **Login with Casdoor** button to test the authentication flow.

Step 6: Handle Token Storage and Refresh

To maintain user sessions, you should implement token refresh logic:

```
async function refreshAccessToken(refreshToken) {
  const response = await fetch(`${CASDOOR_ENDPOINT}/api/login/oauth/refresh_token`, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      grant_type: 'refresh_token',
      refresh_token: refreshToken,
      client_id: CLIENT_ID,
      client_secret: CLIENT_SECRET,
    }),
  });

  return await response.json();
}

// Check token validity and refresh if needed
async function getValidAccessToken() {
  const { user } = await chrome.storage.local.get('user');

  if (!user || !user.access_token) {
    return null;
  }

  // Check if token is expired (decode JWT)
  try {
    const payload = JSON.parse(atob(user.access_token.split('.')[1]));
    const expiry = payload.exp * 1000; // Convert to milliseconds

    if (Date.now() >= expiry) {
      // Token expired, refresh it
      if (user.refresh_token) {
        const newTokens = await refreshAccessToken(user.refresh_token);
        await chrome.storage.local.set({ user: newTokens });
        return newTokens.access_token;
      }
      return null;
    }

    return user.access_token;
  } catch (error) {
    console.error('Error checking token validity:', error);
    return null;
  }
}
```

What's More

You can explore the following projects/docs to learn more about integrating Casdoor with Chrome extensions:

- casdoor-chrome-extension - Official example repository
- Casdoor OAuth Documentation
- Chrome Extension Identity API
- Casdoor JavaScript SDK

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 `http://localhost:8000`
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**。 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。

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`http://localhost:8000`
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AcessToken

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

可用的作用域（scope）

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

> **信息**
>
> 您的 OAuth 应用程序可以在首次重定向时带上请求的作用域。 您可以指定多个作用域并使用空格（转义后为%20）分隔：
>
> ```
> https://<CASDOOR_HOST>/login/oauth/authorize?
> client_id=...&
> scope=openid%20email
> ```
>
> 更多详情，请参阅 OIDC 标准

当您的用户通过 Casdoor 身份验证后，他会被 Casdoor 重定到：

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

现在您已经获得授权码，在你的后端应用发送 POST 请求：

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

在你的后端应用

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

您将得到以下响应：

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

> **备注**
>
> Casdoor 也支持 PKCE 功能。 当获取验证码时，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://<CASDOOR_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。

Device Grant

Maybe your devices have limited input capabilities or lack a suitable browser, and you need to use Device Grant. First, you need to make sure you have Device Grant enabled, the request `device_authorization_endpoint` in OIDC discover, then use QR code or text to show `verification_uri` and lead user to enter login page.

Second, you should request `token endpoint` to get Access Token with parameter define in rfc8628.

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

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

您将得到以下响应:

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

使用客户端凭据授权

当应用程序没有前端时，您也可以使用客户端凭据授权。

首先，你需要确保你已启用客户端凭证授权，并发送POST请求到`https://<CASDOOR_HOST>/api/login/oauth/access_token`:

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

您将得到以下响应:

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

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

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://<CASDOOR_HOST>/api/login/oauth/refresh_token`发送POST请求

```
{
    "grant_type": "refresh_token",
    "refresh_token": REFRESH_TOKEN,
    "scope": SCOPE,
    "client_id": ClientId,
    "client_secret": ClientSecret,
}
```

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: CASDOOR_HOST
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Authorization: Basic Y2xpZW50X2lkOmNsaWVudF9zZWNyZXQ=

token=ACCESS_TOKEN&token_type_hint=access_token
```

您将收到以下响应：

```
{
    "active": true,
    "client_id": "c58c...",
    "username": "admin",
    "token_type": "Bearer",
    "exp": 1647138242,
    "iat": 1646533442,
```

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

类型2：HTTP Bearer 令牌

`https://<CASDOOR_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。

Guest Authentication

Guest authentication enables applications to create temporary users without requiring credentials upfront. This is useful for allowing users to access your application immediately while deferring registration until later.

Creating a Guest User

To create a guest user and obtain an access token, send a POST request to the token endpoint:

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

Request Body:

```
{
    "grant_type": "authorization_code",
    "client_id": "your_client_id",
    "client_secret": "your_client_secret",
    "code": "guest-user"
}
```

> **备注**
>
> The special code value `"guest-user"` is a Casdoor-specific extension that triggers guest user creation instead of the standard OAuth authorization code flow.

Response:

```
{
    "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...",
    "id_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...",
    "refresh_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...",
    "token_type": "Bearer",
    "expires_in": 10080,
    "scope": "openid"
}
```

The system automatically creates a guest user with:

- A randomly generated username in the format `guest_<uuid>`
- A random password
- The tag `guest-user` for identification

Upgrading Guest Users

Guest users are automatically upgraded to normal users when they update their credentials through the user update API.

Upgrade triggers:

- Changing the username to a non-guest format (not starting with `guest_`)
- Setting or changing the password

After upgrade, the user's tag changes from `guest-user` to `normal-user`, enabling standard authentication.

Restrictions

Guest users cannot sign in through the standard login flow. They must first upgrade their account by setting proper credentials. This ensures that guest users transition to permanent accounts before using password-based authentication.

Example Integration

```
// Create a guest user
async function createGuestUser() {
  const response = await fetch('https://your-casdoor-host/api/
login/oauth/access_token', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
      grant_type: 'authorization_code',
      client_id: 'your_client_id',
      client_secret: 'your_client_secret',
      code: 'guest-user'
    })
  });

  const data = await response.json();
  return data.access_token;
}

// Later, upgrade the guest user
async function upgradeGuestUser(accessToken, newUsername,
newPassword) {
  const response = await fetch('https://your-casdoor-host/api/
update-user', {
    method: 'POST',
```

Related Documentation

- OAuth 2.0 - Standard OAuth flows
- User Tags - Understanding user tags
- Application Tags - Restricting access by tags

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`

For more information about CAS, its different versions, and parameters for these endpoints, refer to the CAS Protocol Specification.

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。

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。

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。

- **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。

SAML attributes

Some SP will require you to provide external attributes in SAML Response, you can add those in SAML attributes table. And you can insert user's field to it.

For examle

名称	Name format	Value
`https://www.aliyun.com/SAML-Role/Attributes/RoleSessionName`	Unspecified	`$user.name`
`https://www.aliyun.com/SAML-Role/Attributes/Role`	Unspecified	`acs:ram::1879818006829152:role/$user.roles,acs:ram::1879818006829152:saml-provider/testa`

will generate response with external `saml:Attribute`

```
<saml:Attribute Name="https://www.aliyun.com/SAML-Role/Attributes/RoleSessionName"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
    <saml:AttributeValue xsi:type="xs:string">admi122n@outlook.com</saml:AttributeValue>
</saml:Attribute>
<saml:Attribute Name="https://www.aliyun.com/SAML-Role/Attributes/Role"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
    <saml:AttributeValue xsi:type="xs:string"> acs:ram::1879818006829152:role/
role1,acs:ram::1879818006829152:saml-provider/testa</saml:AttributeValue>
```

> **信息**
>
> We only support insert
> `$user.owner`,`$user.name`,`$user.email`,`$user.id`,`$user.phone`,`$user.roles`,`$user.permissions`,`$user.groups`

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
Name	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{}
```

运行上述代码，控制台将显示以下消息。

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

点击URL进行身份验证，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`。

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

- 将`SAML元数据`的内容保存为XML文件。

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**。

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**。

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。

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**。

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。

VPC dashboard

EC2 Global View New

Filter by VPC:

Select a VPC

- 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

Actions | Download client configuration | Create client VPN endpoint

Filter client VPN endpoints

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。

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。

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。

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。

复制登录页面URL。 这将在下一步中使用。

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）。

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。

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。

Gmail
Images
Sign in
Google
Google Search
I'm Feeling Lucky
Google offered in: 日本語
Japan
About
Advertising
Business
How Search works
Privacy
Terms
Settings

Appgate (POST)

在Appgate中的Casdoor作为SAML IdP

Appgate accepte `SAMLResponse` 由 POST 请求发送。 如果您使用另一个也支持POST请求的服务供应商(SP)，您可以引用此文档。

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。

- 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。

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。

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:

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:

Home > Access – Policies > Edit

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
`https://cloud.tencent.com/SAML/Attributes/Role`	未指定	qcs::cam::uin/{AccountID}:roleName/{RoleName1};qcs::cam::uin/{AccountID}:roleName/{RoleName2},qcs::cam::uin/{Accoun provider/{ProviderName}
`https://cloud.tencent.com/SAML/Attributes/RoleSessionName`	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：

```go
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。

面部识别

概述

我们现在通过利用 face-api.js，将 Face ID 登录集成到了 Casdoor 中。

启用方法

在组织的个人页设置项中添加 Face ID 选项

`用户管理` → `组织` → 选择一个组织 → 找到 `个人页设置项` 部分并添加 `Face ID`。

在此之后，你会在用户界面下找到 Face ID 选项，用户可以上传他们的面部数据用于登录。

`用户管理` → `用户` → 选择一个用户 → 找到 `Face IDs` 并添加面部数据。 你最多可以添加 5 条面部数据，并且可以为每条面部数据自定义名称。

第三步：在应用程序的登录方式部分中添加 Face ID 作为登录选项。

`身份认证` → `应用` → 选择一个应用程序 → 转到 `登录方式` 部分 并将 `Face ID` 作为登录选项。

最后，你可以在登录页面使用 `Face ID` 方式进行登录。

1. 在登录页面，选择 `Face ID` 登录方式。
2. 输入用户名。点击 `人脸登录`。
3. 一旦你授予摄像头访问权限，你就可以使用 `Face ID` 进行登录。

Casdoor
Password
Email
Face ID
WebAuthn
admin
Auto sign in
Forgot password?
Sign in with Face ID
No account? sign up now

这是一个演示如何配置 `Face ID` 登录的视频:

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。

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

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`fetch()`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`XXXListPage.js`文件类似
OrganizationEditPage.js	用于编辑一个组织的页面，与所有其他的`XXXEditPage.js`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

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。

Password Obfuscator

Supporting different password obfuscator options.

📄 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Cassdoor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 Cassdoor 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

2. 在组织的列表页中单击 **群组** 按钮

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
Built-in Organization
Built-in Organization
localhost:7001/users/built-in/admin
Casdoor
Home
Organizations
Groups
Users
Roles
Permissions
Models
Adapters
Applications
Providers
Chats
Messages
Resources
Records
Plans
Pricings
Subscriptions
Admin
Edit User
Save
Save & Exit
Organization :
built-in
ID :
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

Password Obfuscator

Here, we will show you how to enable the option to specify the password obfuscator for password parameters in the login and set-password APIs.

Configuration

On the organization edit page, you can find the `Password obfuscator` configuration option. You can select the encryption algorithm from the dropdown list.

- Plain: Password parameters will be transmitted directly in plain text.
- AES: Password parameters will first be encrypted using the AES algorithm and then transmitted in ciphertext form.
- DES: Password parameters will first be encrypted using the DES algorithm and then transmitted in ciphertext form.

Each time you update the encryption algorithm other than Plain, Casdoor will randomly generate an encryption key for you and populate it into the `Password obf key` configuration option. If you want to specify the encryption key for the encryption algorithm, you can modify the key in `Password obf key` configuration option:

> **备注**
>
> If your key does not meet the encryption algorithm requirements, Casdoor will prompt you with the regular expression that the key should meet in the error message.

API Support

Password obfuscation is supported by the following APIs:

- **Login API** (`/api/login`): Encrypts the password field during authentication
- **Set Password API** (`/api/set-password`): Encrypts both `oldPassword` and `newPassword` fields when changing passwords

When password obfuscation is enabled, Casdoor's frontend automatically encrypts password values before sending them to the server. The backend decrypts these values using the configured key and algorithm, then processes them normally.

Backward Compatibility

The set-password API maintains full backward compatibility. If obfuscation is not configured or decryption fails, the API automatically falls back to accepting plaintext passwords. This ensures compatibility with:

- Casdoor SDKs that may not support obfuscation yet
- Direct HTTP API calls using plaintext passwords
- Legacy integrations

Here is a demo video that shows how to use password obfuscator:

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:

Column Name	Selectable Value	Description
Name	-	Account item name.
Visible	`True` / `False`	Select whether this account item is visible on the user home page.
ViewRule	Rule Items	Select a rule to use when viewing the account item. Controls who can **view** this field.
ModifyRule	规则项	选择用于修改帐户项目的规则。 Controls who can **edit** this field.

Understanding View Rule and Modify Rule

View rule and **Modify rule** provide field-level permission control for user account items:

- **View rule**: Determines who can **see** the value of this account field (e.g., email, phone number, address)
- **Modify rule**: Determines who can **change** the value of this account field

This is different from the broader Permission feature, which controls access to applications and resources. View rule and Modify rule specifically control access to individual user profile fields.

Configuration Steps

要自定义账户项目，请遵循以下步骤：

1. Navigate to **Organizations** in the Casdoor sidebar

2. Click on your organization to open the **Edit Organization** page

3. Scroll down to the **Account items** section

Account items :

Account items Add

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

4. Casdoor 提供简单的操作来配置帐户项目：

a. Set item visibility

Control whether this account item is shown on the user home page:

Account items Add

Name	visible	viewRule	modifyRule
Organization	✓	Public	Admin
ID	✗		
Name	✓	Public	Admin
Display name	✓	Public	Self
Avatar	✓	Public	Self
User type	✓	Public	Admin

b. Set viewing and modifying rules

Configure who can view and modify each field:

Available Rules

There are 3 rules available for both View rule and Modify rule:

- **Public**: Everyone has permission. Any user can view/modify this field for any user.
- **Self**: Each user has their own permission. Users can only view/modify their own field values.
- **Admin**: The administrator has permission. Only organization administrators can view/modify this field for users.

Example Use Cases

Here are some common configuration patterns:

Field	View Rule	Modify Rule	Use Case
姓名	Public	Self	Everyone can see names, but users can only change their own

Field	View Rule	Modify Rule	Use Case
邮件	Self	Self	Users can only see and change their own email
手机号	Admin	Admin	Only admins can see and change phone numbers (for privacy)
显示名称	Public	Self	Public profile name visible to all
密码	Self	Self	Users can only change their own password

> **提示**
>
> Use **Admin** rules for sensitive fields like phone numbers, addresses, or internal identifiers that should only be managed by administrators.

> **备注**
>
> These field-level permissions work in conjunction with the broader Permission system in Casdoor. The Permission system controls access to applications and API resources, while View rule and Modify rule control access to specific user profile fields within the **Edit Organization** page configuration.

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`
- `Name`
- `Display name`
- `头像`
- `用户类型`
- `Password`
- `Email`
- `Phone`
- `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：

> **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

Copy signin page URL

Background URL

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。

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。

Remember Multi-Factor Authentication

When logging in to Casdoor, users can choose to have the Multi-Factor Authentication for this account remembered for a specific period. This means they won't be prompted to perform Multi-Factor Authentication for the account again during that time.

You can set the duration for which Multi-Factor Authentication is remembered for an account in the organization settings. As shown, there is an "MFA remember time" option where you can specify the time (e.g., set it to 12 hours as in the example).

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

提供商

配置不同的提供商

登录方式

配置登录方式和登录方式的显示顺序

Exclusive Signin

Configure exclusive signin to allow only one active session per user

注册项目表

配置注册项目表以创建自定义注册页面

Signin Items Table

Configure the signin items table to create a custom signin page

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

Shared Application

Shared application across organizations

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。
- `CreatedTime`：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。
- `RedirectUris`：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。

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 Cassdoor 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 Cassdoor 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.

Verification Code Settings

You can configure the **Code resend timeout** to control how long users must wait before requesting another verification code via email or SMS. Set the value in

seconds (default is 60). This setting determines the countdown timer duration shown to users on the login page. A value of 0 uses the global default.

:::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**

:::

提供商

您还可以通过添加提供商并设置其属性来添加第三方应用程序进行注册。

我们的供应商可以区分不同的情况，您可以通过选择规则为不同的功能选择不同的供应商。 有关各规则项目的详细解释，请参见下表。

规则	描述
注册	对于注册场景，您可以选择“注册”规则，供提供商发送相应的短信或电子邮件模板。
登录	对于登录场景，您可以为提供商选择“登录”规则。
忘记密码	当在应用程序中选择"忘记密码"场景的提供商时，您可以选择"忘记密码"规则。
重置密码	当在应用程序中选择"忘记密码"场景的提供商时，您可以选择"忘记密码"规则。

规则	描述
设置 MFA	对于 MFA 设置验证场景，可以选择 “设置 MFA ”规则。
MFA Auth	对于MFA Auth验证情景，您可以选择"MFA Auth"规则。 欲了解更多关于mfa的信息，请参阅**MFA**
all	如果您想要为所有功能使用单个提供商，您可以选择“all”规则。 这意味着同一个提供商将用于您的应用程序中提到的所有场景。

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。

> **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。 然后我们将`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。

> **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：

Exclusive Signin

Exclusive signin restricts users to maintaining only one active session at a time. When enabled, signing in from a new device or browser automatically terminates all previous sessions for that user.

Configuration

Enable exclusive signin in the application settings:

1. Navigate to **Applications** in Casdoor
2. Select your application
3. Toggle **Enable exclusive signin**

Once enabled, the system enforces single-session access for all users of that application.

Behavior

When a user with exclusive signin enabled signs in:

- Any existing active sessions are immediately terminated
- A new session is created for the current signin
- The user is logged out from all other devices or browsers

For example, if a user signs in on their laptop, then later signs in on their phone, the laptop session is automatically ended. Only the phone session remains active.

Security Implications

Exclusive signin prevents concurrent session abuse where multiple parties could access the same account simultaneously. This is particularly useful when users forget to sign out from shared or public computers, as the next signin automatically invalidates the previous session.

The feature also helps manage server resources by limiting the number of concurrent sessions, though users working across multiple devices will need to re-authenticate when switching between them.

Technical Implementation

When a user signs in with exclusive signin enabled:

1. Casdoor queries all existing sessions for that user and application
2. All found session IDs are destroyed in the backend
3. A new session is created and stored
4. Only the new session ID is maintained in the database

This per-application session management works independently for each application configured in Casdoor.

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。

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` /	选择是否在用户忘记填写此注册项目时提示。

列名	可选值	描述
	`False`	
Label	-	If this signup item start with `Text`, Label should be the html code for this field. If not it will repalce the label of this signup item.
Custom CSS	-	CSS code for this signup item.
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	`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。

项目名称	可选规则	描述
称		
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。
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。

> **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。

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。

Casdoor

* Username:

* Password:

* Email:

Sign Up

Have account? sign in now

Signin Items Table

On the application configuration page, we can configure the signin items table to create a customized registration page. We can add or delete any signin item on this signin items table.

For a detailed explanation of each signin item, please refer to the table below.

Column Name	Selectable Value	Description
Name	-	The name of the signin item.
Visible	`True` / `False`	Select whether this signin item is visible on the registration page.
Label HTML	-	If this signin item is added as a custom item, Label should be the html code for this field.

Column Name	Selectable Value	Description
Custom CSS	-	CSS code for this signin item.
Placeholder	-	The placeholder of the signin item.
Rule	`Rule` `Items`	Select a rule to customize this signin item. Detailed rules are described in the table below.
Action	-	Users can perform actions such as moving this signin item up, moving it down, or deleting it.

The Captcha signin item supports configuration rules to control how verification is presented to users.

Item Name	Selectable Rules	Description
Captcha	`Normal` / `Inline`	Choose how captcha verification is displayed. `Normal` shows a modal dialog when sending verification codes. `Inline` displays the captcha directly on the signin page, streamlining the verification process.

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。

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。

Part 2: 自定义登录面板

现在我们在第一步操作结束的页面：

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>
```

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。 我们还为您提供了三个按钮，可以决定面板的位置。

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。

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 。

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Cassdooro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。

- 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。

Casdoor
Please type an organization to sign in
built-in
Confirm

> **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。

Casdoor uses specific tag values for special user types. The `guest-user` tag identifies temporary users created through guest authentication. These users automatically upgrade to `normal-user` when they set credentials. See Guest Authentication for details.

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。

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。

> **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：

Shared Application

Introduction

If you want to create an application that can be shared with other organizations, you can enable Is Shared field in application(for safety reasons, only built-in organization can create shared application). To specify the organization, you should add `-org-` and organization name after clientId / application name. For example, the clientId of application is `2dc94ccbec09612c04ac`, your organization name is `casbin`, the clientId for your organization is `2dc94ccbec09612c04ac-org-casbin` and the login url for oauth is `https://door.casdoor.com/login/oauth/authorize?client_id=2dc94ccbec09612c04ac-org-casbin&response_type=code&redirect_uri=http://localhost:9000&scope=read&state=casdoor`.

Configuration

1. First create a new application.
2. Enable Is Shared field.
3. add `-org-` to split organization and clientId / application name.

> ⚠ **注意事项**
>
> Once you shared an application, it can be used by all organizations, and cannot be disabled for a particular organization.

Here is a demo video that shows how to use shared application:

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

Introduction

All users associated with a single Casdoor organization share access to the organization's applications. 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。

Understanding Casbin Concepts

Before delving deeper into the topic, it is important to have a basic understanding of how Casbin works and its related concepts:

- **Model**: Defines the structure of your permission policies and the criteria for matching requests against these policies and their outcomes. You can configure models in the **Models** page in Casdoor.
- **Policy**: Describes the specific permission rules (who can access what resources with what actions). You configure policies in the **Permissions** page in Casdoor.
- **Adapter**: An abstraction layer that shields Casbin's executor from the source of the Policy, allowing the storage of Policies in various locations like files or databases. Learn more about Adapters.

> **提示**
>
> Learn More About Casbin

> Visit the Casbin documentation to learn more about access control models and patterns. You can also use the Casbin Online Editor to create and test Model and Policy files for your specific scenarios.

Configuring Permissions in Casdoor

Where to Configure

In the Casdoor Web UI, you'll work with two main pages:

1. **Models Page**: Navigate to **Models** in the sidebar to add or edit Models for your organization.

2. **Permissions Page**: Navigate to **Permissions** in the sidebar to configure permission policies.

How Permissions Work

Returning to the subject of permission configuration in Casdoor:

1. **Add a Model**: First, create a Model for your organization in the **Models** page within the Casdoor Web UI.
2. **Configure a Policy**: Then, add a Policy (permission rules) for your organization in the **Permissions** page.

Casbin在线编辑器可以为您提供针对您特定使用场景定制的模型和策略文件。 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。 For the Policy configuration (i.e., the **Permissions** page in the Casdoor Web UI), refer to the Permission Configuration guide for detailed instructions.

Using Permissions with Your Application

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。 See the Exposed Casbin APIs documentation for definitions of these API interfaces and instructions on how to use them.

Related Features

Account Item Permissions

Casdoor also provides fine-grained permission control at the user account field level through the **Edit Organization** page:

- **View rule**: Control who can view specific user account fields
- **Modify rule**: Control who can modify specific user account fields

These rules can be set to:

- **Public**: Everyone has permission
- **Self**: Each user has their own permission
- **Admin**: Only administrators have permission

Learn more in the Account Customization documentation.

Role-Based Access Control

Casdoor supports role-based permissions where you can assign roles to users and configure permission policies for these roles. This allows you to manage permissions at the role level rather than individual user level.

Next Steps

- Permission Configuration: Learn how to configure each field in the Permission page
- Exposed Casbin APIs: Use Casbin APIs in your external applications
- Adapters: Configure adapters for policy storage
- Account Customization: Configure field-level permissions for user accounts

让我们开始吧！

权限配置

This page explains each field in the **Edit Permission** page where you configure permission policies for your organization.

Accessing the Permission Configuration

To access the permission configuration page:

1. Log in to your Casdoor instance
2. Navigate to **Permissions** in the sidebar
3. Click **Add** to create a new permission or click on an existing permission to edit it

Configuration Fields

Basic Information

Organization

The name of the organization to which the policy belongs. 一个组织可以拥有多个权限策略文件。 You can select the organization from the dropdown menu in the **Edit Permission** page.

Name

The globally unique name of the permission policy in the organization. 它用于识别策略文件。

- Must be unique within the organization
- Used as the identifier when calling Casbin APIs

Display name

A user-friendly name for the permission policy. This is shown in the Casdoor Web UI for better readability.

Model and Storage Configuration

Model

The name of the model file that describes the structure and matching patterns of the permission policy. You can configure models in the **Edit Model** page.

Models define how permission checks are performed. For example:

- Simple ACL (Access Control List)
- RBAC (Role-Based Access Control)
- ABAC (Attribute-Based Access Control)

Learn more about models in the Casbin documentation.

> **提示**
>
> Use the Casbin Online Editor to create and test your model before adding it to Casdoor.

Adapter

Attention! In the current version, this field describes the name of the database table that stores the permission policy, rather than the name of the adapter configured in the **Adapters** page in the Casdoor Web UI.

Casdoor uses its own database to store configured permission policies:

- If this field is **empty**, the permission policy will be stored in the `permission_rule` table
- If **specified**, it will be stored in the specified database table
- If the specified table name does not exist in the database used by Casdoor, it will be created automatically

> **注意事项**
>
> Important
>
> We strongly recommend **specifying different adapters (table names) for different models**, as keeping all policies in the same table may cause conflicts.

Learn more about adapters in the Adapter documentation.

Subject Configuration

These fields define **who** the permission policy applies to.

Sub users

Which users will the permission policy be applied to. You can select specific users from the **Edit Permission** page.

Examples:

- Select specific users like `alice`, `bob`
- Leave empty to not restrict by user

Sub roles

If the RBAC (Role-Based Access Control) model is used, you can select which roles will be applied to the permission policy. You configure roles in the **Edit Role** page.

This will add permission policies such as `g, user, role` for every user in this role.

Examples:

- Select roles like `admin`, `editor`, `viewer`
- All users with these roles will inherit the permissions

> **提示**
>
> Role-Based Permissions

> Using roles is a powerful way to manage permissions at scale. Instead of assigning permissions to individual users, you assign them to roles, and then assign roles to users.

Sub domains

Which domains will the permission policy be applied to. This is useful for multi-tenant scenarios.

Examples:

- `domain1`, `domain2`
- Leave empty if not using domain-based access control

Object and Action Configuration

These fields define **what** resources and **what actions** are controlled by the policy.

Resource type

In the current version, Casdoor does not use this field for external applications that want to authenticate. You can ignore it for now or use it for your own categorization purposes.

Resources

This field describes the resources for which you wish to enforce permission control.

> 备注

> Note that the resources here are **not** those configured in the **Resources** page of the Casdoor Web UI. You can add any string you want here, such as:
>
> - A URL: `/api/users`, `/admin/dashboard`
> - A file name: `document.pdf`, `config.yaml`
> - A resource identifier: `project:123`, `database:users`

You can add multiple resources, and Casdoor will create permission rules for each combination of resource and action.

Actions

This field describes the actions to operate on resources. Similar to resources, it can be any string you want, such as:

- HTTP methods: `GET`, `POST`, `PUT`, `DELETE`
- CRUD operations: `read`, `write`, `update`, `delete`
- Custom actions: `view`, `edit`, `approve`, `publish`

> **注意事项**
>
> Please note that Casdoor will **convert all these strings to lowercase** before storing them. Additionally, Casdoor will **apply all actions to each resource**. You cannot specify that an action only takes effect on certain resources in this configuration page.

If you need fine-grained control over action-resource combinations, you should define this in your Model file.

Effect Configuration

Effect

This option takes effect for Casdoor itself to control application access.

> **信息**
>
> If you want an external application to enforce permission controls using the interface Casdoor exposes, this field won't do anything. You should describe the effect of pattern matching in the Model file using `allow` or `deny` rules.

Example Configuration

As you can see, this configuration page is almost tailor-made for the `(sub, obj, act)` model, which is one of the most common permission models.

Here's an example configuration:

- **Model:** `rbac_model` (Role-Based Access Control)
- **Sub roles:** `admin`, `editor`
- **Resources:** `/api/users`, `/api/posts`
- **Actions:** `read`, `write`

This would allow users with the `admin` or `editor` role to perform `read` and `write` actions on the `/api/users` and `/api/posts` resources.

Related Topics

- Permission Overview: Understand the basics of permissions in Casdoor
- Exposed Casbin APIs: Use permissions in your external applications
- Adapters: Configure policy storage adapters
- Account Customization: Configure View rule and Modify rule for user account fields
- User Roles: Manage user roles in the **Edit Role** page

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 294b09fbc17f95daf2fedd8982f7046ccba1bbd7851d5c1ece4e52bf039d`。

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）。 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

The Enforce API supports multiple query parameters to specify which permission(s) to enforce against. Only one parameter should be provided at a

time:

- `permissionId`: The identity of a specific permission policy (format: `organization name/permission name`)
- `modelId`: The identity of a permission model (format: `organization name/model name`) - enforces against all permissions using this model
- `resourceId`: The identity of a resource - enforces against all permissions for this resource
- `enforcerId`: The identity of a specific enforcer
- `owner`: The organization name - enforces against all permissions in this organization

Request using `permissionId`:

```
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"]'
```

Request using `modelId`:

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

Request using `resourceId`:

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

Response:

```
{
    "status": "ok",
    "msg": "",
    "sub": "",
    "name": "",
    "data": [
        true
    ],
    "data2": [
        "example-org/example-model/example-adapter"
    ]
}
```

Note: When using `modelId`, `resourceId`, `enforcerId`, or `owner` parameters, the response `data` array may contain multiple boolean values (one for each permission that was checked), and `data2` contains the corresponding model and adapter identifiers.

BatchEnforce

The BatchEnforce API supports multiple query parameters to specify which

permission(s) to enforce against. **Only one parameter should be provided at a time:**

- `permissionId`: The identity of a specific permission policy (format: `organization name/permission name`)
- `modelId`: The identity of a permission model (format: `organization name/model name`) - enforces against all permissions using this model
- `enforcerId`: The identity of a specific enforcer
- `owner`: The organization name - enforces against all permissions in this organization

Request using `permissionId`:

```
curl --location --request POST 'http://localhost:8000/api/batch-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"], ["example-org/example-user2", "example-resource", "example-action"], ["example-org/example-user3", "example-resource", "example-action"]]
```

Request using `modelId`:

```
curl --location --request POST 'http://localhost:8000/api/batch-enforce?modelId=example-org/example-model' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic <Your_Application_ClientId> <Your_Application_ClientSecret>' \
--data-raw '[["example-org/example-user", "example-resource", "example-action"], ["example-org/example-user2", "example-resource", "example-action"]]'
```

响应:

```
{
    "status": "ok",
    "msg": "",
    "sub": "",
    "name": "",
    "data": [
        [
            true,
            true,
            false
        ]
    ],
    "data2": [
        "example-org/example-model/example-adapter"
    ]
}
```

Note: When using `modelId`, `enforcerId`, or `owner` parameters, the response `data` array may contain multiple arrays of boolean values (one array for each permission that was checked), and `data2` contains the corresponding model and adapter identifiers.

GetAllObjects

This API retrieves all objects (resources) that a user has access to. It accepts an optional `userId` parameter. If not provided, it uses the logged-in user's session.

Request with `userId` parameter:

```
curl --location --request GET 'http://localhost:8000/api/get-all-objects?userId=example-org/example-user' \
```

Request using session (userId determined from session):

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

响应:

```
{
    "status": "ok",
    "msg": "",
    "data": [
        "app-built-in",
        "example-resource"
    ]
}
```

GetAllActions

This API retrieves all actions that a user can perform. It accepts an optional `userId` parameter. If not provided, it uses the logged-in user's session.

Request with `userId` parameter:

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

Request using session (userId determined from session):

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

响应:

```
{
    "status": "ok",
    "msg": "",
    "data": [
        "read",
        "write",
        "admin"
    ]
}
```

GetAllRoles

This API retrieves all roles assigned to a user. It accepts an optional `userId` parameter. If not provided, it uses the logged-in user's session.

Request with `userId` parameter:

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

Request using session (userId determined from session):

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

响应:

```
{
    "status": "ok",
    "msg": "",
    "data": [
        "role_kcx66l"
    ]
}
```

RunCasbinCommand

This API executes Casbin CLI commands and returns their output. It's designed for running language-specific Casbin command-line tools through Casdoor's backend, supporting languages like Java, Go, Node.js, Python, and others.

The API includes an in-memory cache that stores command results for 5 minutes. When the same command is executed with identical parameters, the cached result is returned immediately without re-executing the command, improving response times and reducing server load.

Request:

```
curl --location --request GET 'http://localhost:8000/api/run-casbin-command?language=go&args=["-v"]' \
--header 'Authorization: Basic <Your_Application_ClientId><Your_Application_ClientSecret>'
```

Parameters:

- `language`: The programming language for the Casbin CLI (e.g., `go`, `java`, `node`, `python`)
- `args`: A JSON-encoded array of command-line arguments (e.g., `["-v"]` for version, `["new"]` for creating new files). Note: URL-encode the JSON array when using it as a query parameter

Response:

```
{
    "status": "ok",
    "msg": "",
    "data": "casbin version 2.x.x"
}
```

The cache key is generated from the language and arguments, so different commands are cached independently. Expired entries are automatically cleaned up to prevent memory growth.

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。

> **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。

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

> **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

- 删除

User : root

Password : 123456

Database type : MySQL

Database : casdoor

Table : casbin_rule

Model : casbin_rule

Policies : Sync

Add

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

概述

添加第三方服务到您的应用程序

OAuth

24 个项目

电子邮箱

6 个项目

短信

5 个项目

🗃️ 通知

8 个项目

🗃️ 存储

9 个项目

🗃️ SAML

4 个项目

🗃️ 支付

6 个项目

🗃️ 验证码

7 个项目

🗃️ Web3

2 个项目

🗃️ Face ID

2 个项目

🗃️ MFA

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

Currently, we have seven types of providers:

- **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，阿里巴巴云验证码，以及Cloudflare Turnstile。

- MFA Providers

  Casdoor supports external authentication servers for multi-factor authentication. Currently supports RADIUS servers for authenticating users as a second factor during login.

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。

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。

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。

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

User Mapping

Map OAuth provider claims to Casdoor user fields

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。

Apple

Add the Apple Sign in OAuth provider to your application.

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 Octa 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`。

User Field Mapping

OAuth providers often return additional user information beyond the standard profile fields. Casdoor's User Mapping feature allows you to automatically populate user profile fields from OAuth claims returned by your identity provider. This is particularly useful when integrating with enterprise identity providers like Okta, Azure AD, or other custom OAuth services that provide rich user metadata.

Automatic Account Linking

When users authenticate via OAuth, Casdoor automatically attempts to link accounts using

multiple strategies: existing OAuth links, email/phone matching (if enabled), and case-insensitive username matching. This is particularly useful for enterprises with existing users who want to enable OAuth authentication without requiring manual account linking.

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！

User Mapping

When users sign in through OAuth providers, Casdoor automatically captures their basic profile information like username, email, and avatar. However, identity providers often return additional claims in their OAuth responses that contain valuable user data. The User Mapping feature allows you to map these extra claims to specific user profile fields in Casdoor.

Supported Fields

You can map OAuth provider claims to the following user fields:

- **phone** - Phone number
- **countryCode** - Country calling code
- **firstName** - First name
- **lastName** - Last name
- **region** - Geographic region
- **location** - Full location or address
- **affiliation** - Organization or company affiliation
- **title** - Job title or position
- **homepage** - Personal website URL
- **bio** - Biography or description
- **tag** - Custom tag or category
- **language** - Preferred language
- **gender** - Gender identity
- **birthday** - Date of birth

- **education** - Educational background
- **idCard** - ID card number
- **idCardType** - Type of ID card

Standard fields (id, username, displayName, email, avatarUrl) are handled automatically and don't need mapping configuration.

Configuration

To configure user mapping for an OAuth provider:

1. Navigate to **Providers** in the top menu
2. Select or create an OAuth provider (e.g., Okta, Azure AD B2C, Google)
3. Scroll to the **User mapping** section
4. Add mappings by specifying:
    - **User field**: The Casdoor user field you want to populate
    - **Claim name**: The exact claim name from your OAuth provider's response

For example, if your identity provider returns a claim named `given_name` and you want to map it to the user's first name in Casdoor:

- User field: `firstName`
- Claim name: `given_name`

Provider-Specific Examples

Okta

Okta returns claims like `given_name`, `family_name`, and `locale`. You might

configure:

- `firstName` → `given_name`
- `lastName` → `family_name`
- `language` → `locale`

Azure AD B2C

Azure AD B2C can return custom claims configured in your user flows. For instance:

- `phone` → `extension_PhoneNumber`
- `title` → `jobTitle`
- `location` → `city`

Generic OAuth Providers

Most OAuth providers following standard protocols return claims in their userinfo endpoint. Check your provider's documentation to find available claim names.

Behavior

The mapping works with these characteristics:

- **Non-destructive**: Existing user field values are preserved. Mapping only updates empty fields.
- **Automatic sync**: When users sign in via OAuth, the mapping is applied automatically.
- **Flexible**: Each provider can have its own unique mapping configuration.
- **Extra claims**: All claims from the provider are stored in the user's extra data,

even if not explicitly mapped.

Common Scenarios

Enterprise SSO

When integrating with enterprise identity providers like Okta or Azure AD, you often want to sync organizational data:

```
title → jobTitle
affiliation → companyName
region → officeLocation
```

Social Login Enhancement

Social providers like Google or Facebook provide basic profile data, but you can capture additional details:

```
location → location
homepage → website
bio → about_me
```

Multi-Provider Setup

Different providers may use different claim names for the same data. Configure each provider independently:

Google OAuth:

- `firstName` → `given_name`

- `lastName` → `family_name`

GitHub OAuth:

- `location` → `location`
- `homepage` → `blog`
- `bio` → `bio`

Technical Details

When a user authenticates through OAuth:

1. Casdoor receives the OAuth token and fetches user info from the provider
2. The provider response includes standard fields plus extra claims
3. Standard fields (username, email, etc.) are processed first
4. User mapping rules are applied to populate additional fields from extra claims
5. All raw claims are stored in the user's OAuth extra data for reference

This ensures that user profiles in Casdoor stay synchronized with your identity provider while maintaining flexibility in how data is structured.

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。

接下来，导航到**OAuth 同意屏幕**选项卡以配置 OAuth 同意屏幕。

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 Sum

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`。

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

If Get password is enabled, you should enable google people api first and add scope `https://www.googleapis.com/auth/user.phonenumbers.read`

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。

然后填写GitHub应用名称、主页URL、描述和回调URL。

> **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！

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

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。

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。

> **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：

为您的应用添加授权重定向URL，作为**您的Casdoor回调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`。

在Casdoor中添加一个LinkedIn OAuth提供商，并填写`Client ID`和`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。

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。

接下来，设置Facebook登录：

为此应用选择Web平台：

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。

> **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。

基本的应用配置几乎完成了！

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`。

您现在可以使用Facebook作为第三方服务进行身份验证！

Apple

To set up the Apple Sign in provider, please go to the Apple Developer website. An active **Apple Developer Program membership** is required.

Step 1: Configure App ID

Create a new App ID or configure an existing one, and ensure **Sign in with Apple** is enabled for it.

Step 2: Create a Services ID

Next, create a new identifier, making sure to select the **Services IDs** type. (The `Identifier` you set here will be your **Client ID** in Casdoor).

Then, configure this Services ID. Enable **Sign in with Apple** and click **Configure**.

Step 3: Configure Redirect URLs

In the configuration screen, set up the **Return URLs** (callback URLs). You need to enter the **Redirect URL shown on the Casdoor provider page** here.

> **SET RETURN URLS CORRECTLY**
>
> The `Return URLs` on Apple **must** exactly match the `Redirect URL` shown on your Casdoor Apple provider configuration page (e.g., `https://your-casdoor-domain.com/callback`).

Step 4: Create a Key

After configuring the Services ID, create a **Key**. When creating the Key, enable **Sign in with Apple** and associate it with your App ID.

After registering the Key, note down the **Key ID** and **download the `.p8` file immediately**. (This file can only be downloaded once, save it securely!)

Important: Find and note down your **Team ID** from the **Membership** page on the Apple Developer Portal.

Step 5: Configure Casdoor Provider

1. **Client ID**: Enter the Apple **Services ID** `Identifier` you created earlier.
2. **Team ID**: Enter your Apple **Team ID** (found on the Membership page).
3. **Key ID**: Enter the Apple **Key ID** you noted down.
4. **Key Text**: Open the downloaded `.p8` file with a text editor. Copy its **entire content** (including the `-----BEGIN...` and `-----END...` lines) and paste it here.
5. **Check Redirect URL**: Verify that the `Redirect URL` shown here in Casdoor has been correctly added to the **Return URLs** in your Apple Services ID configuration.

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"。

Home Organizations Users Roles Permissions Providers Applications Resources Tokens Recor

Edit Provider Save Save & Exit

Name ⓘ :

Display name ⓘ :

Category ⓘ : OAuth

Type ⓘ : Adfs

Client ID ⓘ

Client secret ⓘ

Domain ⓘ :

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

Save Save & Exit

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。

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**。

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。

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`。

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。

Overview

Custom OAuth providers allow you to integrate any OAuth 2.0 compliant authentication service with Casdoor, even if it's not officially supported. This is useful when you want to integrate with:

- Internal enterprise OAuth servers
- Self-hosted authentication systems
- Third-party services not yet officially supported by Casdoor

Multiple Custom Providers Support

Casdoor supports up to **10 different custom OAuth providers** simultaneously. When creating custom providers, you can choose from the following types:

- **Custom** - The first custom provider
- **Custom2** through **Custom10** - Additional custom providers

This allows you to integrate multiple custom OAuth services without conflicts. Each custom provider maintains its own separate configuration and user data fields.

Creating a Custom Provider

To create a new custom provider, navigate to the provider page of Casdoor, and select one of the custom types ("Custom", "Custom2", "Custom3", etc.) in the Type field. 然后，您需要填写`Client ID`、`Client Secret`、`Auth URL`、`Scope`、`Token URL`、`UserInfo URL`和`Favicon`。

- `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=authoritiation_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": "eyJhbGciOiJSUzI1NiIsImtpZCI6Ixxxxxxxxxxxxxx",
    "refresh_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6xxxxxxxxxxxxxx",
    "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商一起在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**。

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

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`。

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`。

> **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

WeChat ✔

Add WeChat OAuth Provider

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。

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时(即WeChat App的内置浏览器)，Cassdoor 将自动跳过这一步。

You can choose whether to enable the WeChat QR code login option on the setting page. To do so, add the WeChat provider in your application configuration and add the WeChat option in your signin methods. Once added, the login page will display a "WeChat" tab as a login option, allowing users to log in by scanning the QR code.

The QR code login process is as follows:

1. On the login page, after selecting the "WeChat" tab, a WeChat QR code will be automatically loaded and displayed.
2. The user scans the QR code using the WeChat app and completes the authorization to log in.
3. If the QR code expires or needs to be refreshed, the user can click the "Refresh" link below the QR code to obtain a new one.

> **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。

Enable WeChat QR Code Login

You can choose whether to enable the WeChat QR code login option on the setting page. To do so, add the WeChat provider in your application configuration and add the WeChat option in your signin methods. Once added, the login page will display a "WeChat" tab as a login option, allowing users to log in by scanning the QR code.

The QR code login process is as follows:

1. On the login page, after selecting the "WeChat" tab, a WeChat QR code will be automatically loaded and displayed.
2. The user scans the QR code using the WeChat app and completes the authorization to log in.
3. If the QR code expires or needs to be refreshed, the user can click the "Refresh" link below the QR code to obtain a new one.

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`。

确保添加`重定向域`，它应该是你的Casdoor域。

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。

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`！

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

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。

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。

在创建应用程序后，应在以下位置设置重定向URL：

在以下位置添加您的 Casdoor 域名：

> **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`。

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

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{:target="_blank"}并使用您的信息流账户登录。
2. 访问信息流应用{:target="_blank"}页面。

3. 注册您的Infoflow应用。

4. 获取`AgentID`。

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。

7. 按照所示添加敏感接口权限。

敏感接口权限 修改

接口名称	权限开放
获取部门成员	☑
获取部门列表	☐
获取成员信息	☑
获取标签成员	☐
维护通信录	☐
获取成员群组列表	☐
获取群组成员列表	☐
维护群组成员	☐
发送群组消息	☐
维护群组话题	☐
维护勋章	☐
通讯录搜索	☐

8. 在同一页面上，你会找到`CorpID`和`Secret`。

9. 在Casdoor中添加一个Infoflow OAuth提供商，并填写`Client ID`、`Client Secret`和`Agent ID`。

您现在可以将Infoflow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 `OAuth`
Type	Choose `Lark`
Client ID	`App ID` obtained from Step 1
Client secret	`App Secret` 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。

Username Handling

Casdoor uses a fallback mechanism to ensure user accounts are created successfully even when Lark's OAuth response has incomplete data. The username field follows this priority:

1. **UserId** - Primary identifier used when available
2. **UnionId** - Links users across multiple Lark organizations
3. **OpenId** - Always present, used as final fallback

This ensures authentication succeeds reliably since OpenId is guaranteed in Lark's OAuth response.

电子邮箱

概述

使用电子邮件进行身份验证

SendGrid

Using SendGrid as an Email Provider

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

Mailpit

Using Mailpit as the SMTP server

概述

Adding an Email provider

1. Click on `Add` to add a new provider.
2. Select `Email` under the `Category` section.

3. Fill in the fields for `Username`, `Password`, `Host`, and `Port` for your SMTP service.

4. Customize the `Email Title` and `Email Content`, then save the changes.

Proxy configuration

If your server cannot directly access the SMTP service (such as Gmail), you can enable the proxy option. When enabled, email traffic will be routed through the SOCKS5 proxy configured in Casdoor's configuration file.

To enable proxy support, toggle the `Enable proxy` switch in the provider settings. This is particularly useful when connecting to external email services from restricted network environments.

Modify email content

You can use the following placeholders to display some variables.

1. `%{user.friendlyName}` is the Display name of user.

2. `%link` is the reset link for forget password function. (You should put `%link` in the `<reset-link></reset-link>` label)

SendGrid

In this guide, we will use SendGrid as an email provider.

Step 1: Create an API Key for Your SendGrid Account

Expand **Settings** from the left navigation bar, then click on **API Keys**. Here, you will see all previously generated API keys. To create a new one, click on **Create API Key** and configure the necessary permissions.

Step 2: Sender Verification

To verify your email sender, choose between **Single Sender Verification** or **Domain Authentication** by referring to the official documentation:
Sender Identity

Step 3: Configure Casdoor as an Email Provider

Create a SendGrid email provider in Casdoor and fill in the following fields:

Required Fields

Field	Description
Secret Key	Your SendGrid API key
发件人地址	Your verified email address (or domain)

Default Fields

Field	Description
Endpoint	Default: `/v3/mail/send`

Field	Description
主机	Default: `https://api.sendgrid.com`

Email Fields

Field	Description
From Name	The display name of the email sender
Email Title	The subject of the email
Email Content	Supports HTML templates
Test Email	The recipient's email address for testing

Finally, click on the **Send Testing Email** button and check your `Test Email` address for the test email.

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：

> **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

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`。

```
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。

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
Test SMTP Connection
Send Testing Email
Provider URL :
https://github.com/organizations/xxx/settings/applications/1234567
provider:SMTP connected successfully
Email sent successfully

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.

Mailpit

Mailpit

In this guide, we will be using Mailpit as the SMTP server. Mailpit is an email-testing tool that operates with a fake SMTP server.

Step 1: Deploy the Mailpit service

The IP address for the Mailpit service is `127.0.0.1`. By default, the Mailpit SMTP server listens on port 1025 and does not use encryption or authentication.

```
❯ { home } mailpit.exe
time="2025/07/19 16:13:51" level=info msg="[smtpd] starting on [::]:1025 (no encryption)"
time="2025/07/19 16:13:51" level=info msg="[http] starting on [::]:8025"
time="2025/07/19 16:13:51" level=info msg="[http] accessible via http://localhost:8025/"
```

Step 2: Create an email provider

Provide the necessary information and save the settings.

Step 3: Send a test email

First, click on the `Test SMTP Connection` button. If you see `provider: SMTP connected successfully`, it means that your Casdoor service can access the Mailpit service.

Next, click on the `Send Testing Email` button. If you see `Email sent successfully`, it means that the test email has been sent successfully from the `From` address to the `Test Email`.

Mailpit

Mark unread | Delete | Download

Return to inbox

attack825 5 minutes ago
To: admin@example.com
Casdoor Verification Code

attack825 13 minutes ago
To: 2707138687@qq.co
Casdoor Verification Code

attack825 15 minutes ago
To: 2707138687@qq.co
Casdoor Verification Code

attack825 17 minutes ago
To: admin@example.com
Casdoor Verification Code

sender@example.com 29 minutes ago
To: recipient@example.com
Email Subject

About

From attack825 <2707138687@qq.com>
To <admin@example.com>
Subject **Casdoor Verification Code**
Date Sat, 19 Jul 2025, 4:37 pm (1.7 kB)
Tags Add tags...

HTML | HTML Source | Text | Headers | Raw | HTML Check 91% | Link Check

Casbin Organization

Casdoor

Admin, here is your verification code

Use this code for your transaction. It's valid for 5 minutes

123456

Thanks

Casbin Team

Casdoor is a brand operated by Casbin organization. For more info please refer to https://casdoor.org

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

We use casdoor/go-sms-sender to send SMS for Casdoor. The `go-sms-sender` library currently supports Twilio, Submail, SmsBao, Alibaba Cloud, Tencent Cloud, Huawei Cloud, and Volc SMS APIs. If you want to add support for other SMS providers, you can either raise an issue or submit a pull request.

Adding an SMS provider

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`。

3. 选择你的提供商的类型。

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。

Proxy configuration

For SMS providers that use HTTP APIs (such as Custom HTTP SMS), you can enable proxy support if your server cannot directly access the SMS service. When enabled, SMS traffic will be routed through the SOCKS5 proxy configured in Casdoor's configuration file.

To enable proxy support, toggle the `Enable proxy` switch in the provider settings. This option is available for Custom HTTP SMS providers and helps when operating in restricted network environments.

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

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。

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`。

- 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。

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。

- 提供商Url

`提供商Url`是通信服务中的端点。

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。

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

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。

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

WeCom

Using WeCom as a notification provider for Casdoor

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	
Microsoft Teams	
WeCom	
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`：

BotFather
bot

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-dFIPO8VXwEto
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。

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`GET`或`POST`方法。
参数名称	URL查询参数名称或主体参数，取决于`方法`。
内容	您想要发送的消息。
Endpoint	您的HTTP地址

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`

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。

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

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	

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 : twitter

Display name : twitter

Organization : admin (Shared)

Category : Notification

Type : Twitter

Client ID : ovciqKv08CQ9pViofp7h3FtTo

Client secret : ***

Client ID 2 : 1465983076912492549-1YxffAckmgEHBkVot74JxUCq2O1RHg

Client secret 2 : ***

Content : test

Chat ID : 1465983076912492549 Send Testing Notification

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

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 :	discord
Display name :	discord
Organization :	admin (Shared)
Category :	Notification
Type :	Discord
Secret key :	***
Content :	test
Chat ID :	1146715329133821972 Send Testing Notification
Provider URL :	https://github.com/organizations/xxx/settings/applications/1234567

WeCom

Step 1: Create a WeCom Group Chat Bot

WeCom (WeChat Work) supports sending notifications through group chat bots. First, create a group chat in WeCom and add a bot to it.

In your WeCom group chat, go to group settings and add a bot. After adding the bot, you will receive a webhook URL.

Step 2: Get the Webhook URL

Copy the webhook URL provided by WeCom. This URL is used to send messages to your group chat.

The webhook URL format is: `https://qyapi.weixin.qq.com/cgi-bin/webhook/send?key=YOUR_KEY`

For more information, refer to the WeCom Webhook Documentation.

Step 3: Configure Casdoor WeCom Provider

In Casdoor, create a new notification provider and select "WeCom" as the type.

Configure the following field:

Name	Name in WeCom
Endpoint	Webhook URL

Paste your webhook URL into the `Endpoint` field. The `Content` field can be used to define a message template for notifications sent to your WeCom group.

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的存储提供商

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

无前缀

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。

编辑对象所有权并检查ACLs已启用。

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 ⓘ :

http://kininaru.s3-website.ap-northeast-1.amazonaws.com

Bucket ⓘ :

kininaru

Path prefix ⓘ :

Domain ⓘ :

https://d20zlk9foisfk0.cloudfront.net

Region ID ⓘ :

ap-northeast-1

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。

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。

> **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。

- ContainerUrl

您可以从容器属性中获取ContainerUrl。

- (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 :

googlegcs

Display name :

googlegcs

Organization :

admin (Shared)

Category :

Storage

Type :

Google Cloud Storage

Service account JSON :

{"type":"service_account","project_id":"casdoor-sample","private_key_id":"sample-key","private_key":"-----BEGIN PRIVATE KEY-----\nsample-private\n-----END PRIVATE KEY-----\n","client

Endpoint :

Bucket :

casdoor

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`。

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`Storage`
Type	选择`MinIO`
Client ID	从步骤1获取的`Access Key`
Client secret	从步骤1获取的`Secret Key`
Endpoint	从步骤1获取的`API地址`
Bucket	从步骤1获取的`Bucket`

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：

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

- 端点，BucketName和区域

- (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

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

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：

阿里巴巴云IDaaS	Keycloak	自定义
✅	✅	✅

条款

- 身份提供商（IDP）——储存身份数据库并向Cassdoor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`SAML`
Type	选择`自定义`
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。

最后，将SAML自定义提供商添加到应用程序的`Providers`。

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`。

您可以使用默认的领域，或者创建一个新的领域。

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。

KEYCLOAK
Admin
Casdoor
Configure
Realm Settings
Clients
Client Scopes
Roles
Identity Providers
User Federation
Authentication
Manage
Groups
Users
Sessions
Events
Import
Export
Clients > http://localhost:8000/api/acs
Http://localhost:8000/api/acs
Settings
Roles
Client Scopes
Mappers
Scope
Sessions
Offline Access
Clustering
Installation
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 AuthnStatement
ON
Include OneTimeUse Condition
OFF
Force Artifact Binding
OFF
Sign Documents
ON
Optimize REDIRECT signing key lookup
OFF
Sign Assertions
OFF
Signature Algorithm
RSA_SHA256
SAML Signature Key Name
KEY_ID
Canonicalization Method
EXCLUSIVE
Encrypt Assertions
OFF
Client Signature Required
OFF
Force POST Binding
OFF
Front Channel Logout
ON
Force Name ID Format
ON
Name ID Format
username
Root URL
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。

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。

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。

阿里巴巴云IDaaS

在阿里巴巴云IDaaS中创建SAML应用

登录到阿里巴巴云管理控制台，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 IDaS 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**。

填写所有必需的信息并提交。

选择添加的签名密钥。

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 Cassdoor 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

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**。

填写所有必需的信息并提交。

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IDaS中，子账户是Cassdoor用户的ID。

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** IDaS SAML Metadadta

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。

复制 SP ACS URL 和 SP **实体** ID 并点击 **保存** 按钮

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

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。

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。

casbin
username, Email or phone
Password
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

AirWallex

Add AirWallex payment provider to your application

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`。

您可以在应用的基本信息中找到`Client ID`和`Secret key`。

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`Payment`
Type	选择`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。

> **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板。 您可以在`API密钥`选项卡下找到`可发布的密钥`和`秘密密钥`。

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`

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
provider_payment_stripe
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
Casdoor
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，然后你可以获取`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

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，以便用户可以使用支付宝购买产品。

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`：APIv3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`

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。

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。

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。

AirWallex

> **备注**
>
> This is an example of how to configure a **AirWallex** payment provider.

Step 1. Get Client ID and API Key

First, you need to have an account at AirWallex. After creating an AirWallex account, log in to the Developer Dashboard using your account credentials. You can find the `CLIENT ID` and `API KEY` under the `API Keys` tab, or add a new custom permission key.

Step 2. Create an AirWallex Payment provider

Next, create an AirWallex Payment provider in Casdoor by filling in the necessary information.

Name	Name in AirWallex
Category	choose `Payment`
Type	choose `AirWallex`

Name	Name in AirWallex
Client ID	`CLIENT ID` obtained from Step 1
Client secret	`API KEY` obtained from Step 1

Step 3. Add the AirWallex Payment provider for your product

Finally, add the AirWallex Payment provider for your product so that users can purchase the product using AirWallex.

Currency : USD

Is recharge :

Price : 80.59

Quantity : 99

Sold : 10

Payment providers : airwallex_test ×

Return URL :

State : Published

Preview : Test buy page..

Buy Product

Name	New Product - drxsq				
Detail		Tag	Casdoor Summit 2022	SKU	product_drxsq
Image	product_drxsq				
Price	**$80.59 (USD)**	Quantity	99	Sold	10
Pay	AirWallex				

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	阿里巴巴云验证码	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 Cassdoor 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。

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。

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。

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。

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。

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。

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。

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`。

阿里云
工作台
搜索...
费用
ICP 备案
企业
支持
工单
简体
daconghd
主账号
为提供更好的安全能力，验证码产品计划2025年8月28日下线空间推理验证形态，下线后存量空间推理配置将自动切换为拼图验证。由于封网延期，预计9月11日生效。
验证码2.0
概览
场景管理
安全管理
自定义策略
白名单策略 NEW
账单管理
版本管理
告警通知
其他
实人认证
验证码2.0 / 场景管理
帮助文档
场景管理
新建场景
验证码形态
接入方式
名称/ID
您现在已经添加了 1 个场景，超过3个将进行计费
场景名称/ID
接入方式
验证码形态
策略状态
策略类型
数据统计
操作
test
1kyr
Web/H5
一点即过
正式
默认
接入引导
编辑
自定义策略
删除
每页显示
10
共 1 条数据
上一页
1
下一页
scene

阿里云
工作台
搜索...
费用
ICP 备案
企业
支持
工单
简体
daconghd
主账号
场景管理中【客户端类型】目的为区分不同接入方式，为了避免歧义，2025年8月20日将更新为【接入方式】，同时Web和H5由于接入方式一致将合并为Web/H5，接入、使用体验及安全能力均不会产生变化。
验证码2.0
概览
场景管理
安全管理
自定义策略
白名单策略 NEW
账单管理
版本管理
告警通知
其他
实人认证
验证码2.0 / 概览
帮助文档
概览
今日汇总指标 (全量)
时间周期：00:00 - 18:00
初始化量
0
验证量
0
验证拦截量
0
验证通过量
0
累计开通时间
按量版
96 天
身份标 gc
AppKey
开通包年包月
购买资源包
展开详情
全量
场景名称
验证码形态
近7天
2025-09-09
2025-09-15
按小时查看
验证趋势
查看详情
2
1.5
1
0.5
0
2025.09.09 01:00
2025.09.10 02:00
2025.09.11 03:00
2025.09.12 04:00
2025.09.13 05:00
2025.09.14 06:00
2025.09.15 07:00
初始化量
验证量
验证拦截量
验证通过量
验签趋势

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`。

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。

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`

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

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。

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。

Face ID

Overview

Using Face ID for authentication

Alibaba Cloud FaceBody

Add libaba Cloud FaceBody as a third-party faceid service to complete authentication

Overview

Casdoor allows for the use of Face ID as a sign-in method.

We only recommend enabling Face ID if you can ensure that the hardware and requests cannot be tampered with.

Adding an Face ID provider

1. Go to your Casdoor index page.
2. Click on `Providers` in the top bar.
3. Click on `Add`, and you will see a new provider added to the list at the top.
4. Click on the new provider to make changes to it.
5. In the `Category` section, select `Face ID`.
6. Choose the specific Face ID provider that you require from the `Type` dropdown.
7. Fill in the necessary information, such as `Client ID` and `Client Secret`.

Alibaba Cloud FaceBody

Introduction

Based on face detection, analysis/comparison technology in images or videos, and human body detection technology, it provides independent modules for face/human body detection and localization, face attribute recognition and face comparison. It can provide developers and enterprises with high-performance online API services for various scenarios such as face AR, biometric identification and authentication, large-scale face retrieval, and photo management.

How to use?

The steps to use alibaba cloud facebody are shown below.

Step 1: Register alibaba cloud facebody

First, visit Alibaba Cloud Facebody website and open a facebody account.

Step 2: Buy CompareFace resource pack

Visit Face body page and buy CompareFace resource pack

Step 3: Create a client secret

Go to console and create AccessKey and AccessSecret

Step 4: Find Endpoint

You can find your endpoint ID in Aliyun doc

for example, if your region id is cn-shanghai, then the endpoint is `facebody.cn-shanghai.aliyuncs.com`

Step 5: Create Alibaba Cloud Facebody provider in Casdoor

The last step is to add an Alibaba Cloud Facebody Face ID provider and fill in the `Client ID`, `Client Secret` and `Endpoint` in your Casdoor.

MFA

Push Notification

Configure push notification provider for MFA

RADIUS

Configure RADIUS provider for MFA

Push Notification

Push notification MFA enables users to receive verification codes through mobile push services during authentication. This method leverages Casdoor's existing notification provider infrastructure, supporting services like Duo, Pushover, Telegram, and 20+ other providers.

Configure Notification Provider

Push notification MFA uses notification providers to send verification codes. To add a notification provider:

1. Navigate to the **Providers** page and click **Add**.
2. Set the **Category** to **Notification** and choose your preferred **Type** (e.g., Telegram, Pushover, Custom HTTP).
3. Configure the provider-specific settings based on your chosen notification service.
4. Click **Save** to create the provider.

Use in Application

After creating the notification provider, add it to your application:

1. Go to your application's edit page.

2. Add the notification provider to the **Providers** list.
3. Users can now select push notification as their MFA method when configuring multi-factor authentication.

User Setup Flow

During MFA setup, users will:

1. Select "Use Push Notification" as their MFA method.
2. Enter their push notification receiver (device token or user ID).
3. Select the notification provider from the configured options.
4. Receive a verification code via push notification.
5. Enter the verification code to complete setup and receive recovery codes.

Authentication Flow

When push notification MFA is enabled, users will receive a verification code via push notification during login. They must enter this code to complete authentication.

RADIUS

RADIUS (Remote Authentication Dial-In User Service) providers allow Casdoor to authenticate users against external RADIUS servers during multi-factor authentication.

Configure RADIUS Provider

To add a RADIUS provider for MFA:

1. Navigate to the **Providers** page and click **Add**.
2. Set the **Category** to **MFA** and **Type** to **RADIUS**.
3. Configure the following fields:
   - **Host**: The IP address or hostname of your RADIUS server (e.g., `10.10.10.10`)
   - **Port**: The RADIUS server port (default is `1812`)
   - **Client Secret**: The shared secret configured on your RADIUS server for authenticating requests
4. Click **Save** to create the provider.

Use in Application

After creating the RADIUS provider, add it to your application:

1. Go to your application's edit page.
2. Add the RADIUS provider to the **Providers** list.
3. Users can now select RADIUS as their MFA method when configuring multi-factor authentication.

During MFA setup and authentication, users will be prompted to enter their RADIUS username and password, which will be verified against the configured RADIUS server.

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。

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。

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。

SaaS Management

概述

Casdoor定价概述

Product

添加您想要销售的产品

支付

查看支付中产品的交易信息

计划

Casdoor计划概述

Pricing

Casdoor定价概览

订阅

Casdoor订阅概览

Transaction

Casdoor Transaction Overview

概述

Casdoor can be used as a subscription management system through its `Product`, `Payment`, `Plan`, `Pricing`, `Subscription`, and `Transaction` features.

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：

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。

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`。

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。

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。

这里有一个演示视频：

Product

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，你需要理解产品的基本属性：

- 标签
- 详情
- 货币
- 价格
- 数量
- 已售出

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

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。

Success URL (Optional)

If you need the provider to redirect users directly to a custom URL instead of the Casdoor callback page, you can fill in the **Success URL** field. When configured, Casdoor will append the payment owner and transaction name as query parameters to your provided URL.

For example, if you set the Success URL to `http://example.com/payment/success`, users will be redirected to:

```
http://example.com/payment/
success?transactionOwner={paymentOwner}&transactionName={paymentName}
```

You can include additional query parameters in your Success URL, such as:

```
http://example.com/payment/
success?customParam=value&transactionOwner={paymentOwner}&transactionName={paymentName}
```

> ⚠ **注意事项**
>
> **Important:** If you configure the Success URL field, you must manually call the `NotifyPayment` API to complete the transaction, otherwise the payment will fail.
>
> Call the API endpoint: `api/notify-payment/{paymentOwner}/{paymentName}` using the parameters provided in the Success URL query string.

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	Casdoor				
Price	**$300 (USD)**	Quantity	99	Sold	10
Pay	Alipay				

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：

.

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。

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`

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`。

Pricing

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：

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
- Via the Pricing flow when purchasing a product with pricing and plan information (available for all user types)
- 通过API

Any user can create a subscription when purchasing a product, enabling flexible conversion from free to paid tiers. Subscription enforcement (requiring an active subscription for access) only applies to users with `type = "paid-user"`.

`定价`、`计划`、`订阅`、`产品`和`支付`之间的关系如下：

pricing
---
plans
plan
---
productName
product
---
payment
---
productName
subscription
---
pricingName
planName
paymentName
userName
user

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 & Ex

Transaction

The `Transaction` feature tracks financial activities for users and organizations in Casdoor, enabling balance management and transaction history.

Transactions are automatically created when users make purchases or recharge their balance. Each transaction updates the corresponding user or organization balance in real-time.

Transaction Categories

Transactions in Casdoor fall into two categories:

User Transactions track individual user balances. When a user transaction is created, it updates both the user's balance and the organization's total user balance sum.

Organization Transactions track the organization's own operational balance, separate from user balances.

Transaction Properties

Every Transaction has these properties:

- `Owner`
- `Name`
- `CreatedTime`

- `Category`: Either "User" or "Organization"
- `Type`: The transaction type (e.g., "Recharge", "Purchase")
- `User`: Required for User category transactions
- `Amount`: Transaction amount (positive for income, negative for expenses)
- `Currency`: The currency code (e.g., "USD", "CNY")
- `State`: Transaction state (e.g., "Pending", "Paid", "Failed")
- `Payment`: Related Payment record (if applicable)

Balance Tracking

Casdoor maintains separate balance fields:

User Balance is stored on individual user records and tracks each user's available funds.

Organization Balances include two fields: `orgBalance` for the organization's own funds, and `userBalance` for the sum of all user balances within that organization.

Balances are automatically updated when transactions are created, modified, or deleted, ensuring consistency across the system.

Viewing Transactions

Transaction history is displayed in two locations:

When editing a user account, all transactions for that user appear in a dedicated table below the user details.

When editing an organization, all organization-level transactions are shown in the organization edit page.

Both views provide a chronological record with transaction details including name, creation time, category, type, amount, and state.

用户

概述

在Casdoor中管理用户

MFA / 2FA

使用MFA / 2FA保护您的账户

User Impersonation

Using master password to impersonate users in Casdoor

用户角色

分配给用户的角色

📄 权限

用户权限

📄 Forms

A tool for customizing list page displays of system entities.

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
- `CreatedTime 创建时间`
- `UpdatedTime`
- `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`: When a user is soft-deleted (`IsDeleted = true`), they cannot sign in through any authentication method, including OAuth providers. This prevents deleted users from re-registering via third-party login.
- `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`: An array of the user's roles (extended field, read-only via User API)
- `Permissions`: An array of the user's permissions (extended field, read-only via User API)

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`

User Tags

The `Tag` field allows you to categorize users for different purposes. Casdoor uses specific tag values for special user types:

- `normal-user`: Standard users with full authentication capabilities
- `guest-user`: Temporary users created through guest authentication without initial credentials
  - Automatically upgrade to `normal-user` when they set a proper username or password
  - Cannot sign in directly until they upgrade their account

You can also define custom tags to restrict application access. See Application Tags for more information.

Email Normalization

Casdoor normalizes all email addresses to lowercase to ensure uniqueness and prevent duplicate accounts. This means that `user@example.com`, `User@Example.com`, and `USER@EXAMPLE.COM` are treated as the same email address, complying with RFC 5321 standards.

This normalization happens automatically during:

- User signup and account creation
- User login and authentication
- Email duplicate checking

Understanding Roles and Permissions Fields

The `Roles` and `Permissions` fields in the User object are **extended fields** that are dynamically populated when retrieving user data. These fields are not stored directly in the User table but are collected from the Roles and Permissions resources through the `ExtendUserWithRolesAndPermissions()` function.

Important: You cannot update roles and permissions through the `/api/update-user` endpoint, even when using the `columns` parameter. To manage user roles and permissions, you must use the dedicated APIs for Roles and Permissions resources.

To assign roles or permissions to users:

- **Roles**: Use the Roles API endpoints to create and assign roles. Visit the Roles management page (e.g., `https://door.casdoor.com/roles`) or use the roles API.
- **Permissions**: Use the Permissions API endpoints to create and assign permissions. Visit the Permissions management page (e.g., `https://door.casdoor.com/permissions`) or use the permissions API.

Using the Properties Field

The `Properties` field is a flexible key-value map (`map[string]string`) that allows you to store custom attributes for users beyond the predefined fields in the User schema. This is particularly useful when you need to:

- Store organization-specific user attributes

- Add custom metadata that doesn't fit into standard fields
- Extend user profiles without modifying the core schema

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。

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。

Casdoor supports multiple second-factor authentication methods including SMS codes, email codes, TOTP authenticator apps, and RADIUS authentication.

Once you enable MFA, Casdoor requires an authentication code every time someone attempts to sign in to your account. 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。

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。

> **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。

Configuring multi-factor authentication using RADIUS

RADIUS MFA allows you to authenticate against an external RADIUS server for the second authentication factor. This is useful when integrating with existing authentication infrastructure.

Before using RADIUS MFA, your administrator must configure a RADIUS provider in the application. Once configured:

1. Enter your RADIUS username when prompted during setup.

2. Enter your RADIUS password to verify the setup. This password will be verified against the configured RADIUS server. During subsequent logins, you'll enter this same RADIUS password as your second factor.

3. Above the "Enable" button, copy your recovery codes and save them to your device. Save them to a secure location because your recovery codes can help

you regain access to your account if you lose access.

> **备注**
>
> The RADIUS authentication communicates with the external RADIUS server configured by your administrator. Make sure you have the correct username and password for your RADIUS account.

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。

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。

User Impersonation

User impersonation allows administrators to temporarily sign in as another user within their organization. This feature is useful for troubleshooting user-specific issues, testing permissions, or providing support without requiring the user's actual password.

Overview

Casdoor supports user impersonation through the **Master Password** feature. When an organization's master password is configured, administrators can use it to sign in as any user within that organization.

How It Works

When you attempt to sign in as a user, Casdoor checks the password in the following order:

1. First, it checks if the entered password matches the organization's master password
2. If not, it checks if the password matches the user's actual password

If either check succeeds, the authentication is successful. This means that when a master password is set, administrators can use it to sign in as any user account.

Setting Up Master Password

To enable user impersonation in your organization:

1. Navigate to your **Organization** settings in the Casdoor admin console
2. Go to the **Organizations** page
3. Click on your organization to edit it
4. Find the **Master Password** field
5. Enter a strong, secure password
6. Save the changes

> ⚠ **注意事项**
>
> Security Important
>
> The master password is extremely sensitive. Anyone with access to the master password can sign in as any user in the organization. It should be:
>
> - Known only to trusted administrators
> - Stored securely (e.g., in a password manager)
> - Changed immediately if compromised
> - Sufficiently complex and unique

Using Master Password for Impersonation

To sign in as a user using the master password:

1. Go to the login page for your organization: `/login/<organization_name>`

> 提示
>
> Example
>
> For an organization named "my-company", use: `https://your-casdoor-domain.com/login/my-company`

2. Enter the username of the user you want to impersonate
3. Enter the organization's master password (not the user's password)
4. Sign in

You will be authenticated as that user and have access to their account with their permissions and settings.

Use Cases

Common scenarios where user impersonation is helpful:

- **Technical Support**: Troubleshoot user-specific issues by viewing exactly what the user sees
- **Testing**: Verify that permissions and roles are working correctly for different user types
- **Emergency Access**: Access critical information when a user is unavailable
- **Demonstration**: Show features or workflows from a specific user's perspective
- **Audit**: Investigate suspicious activity or verify user actions

Disabling User Impersonation

To disable user impersonation:

1. Navigate to your **Organization** settings
2. Find the **Master Password** field
3. Clear the master password field
4. Save the changes

Once removed, administrators will no longer be able to use a master password to sign in as other users.

Related Features

- **Default Password**: Organizations can also set a default password that is assigned to new users when they are created
- **Password Complexity**: Configure password requirements to ensure all passwords (including master password) meet security standards
- **Multi-Factor Authentication (MFA)**: Even when using master password, MFA requirements will still apply if enabled for the user

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。

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

Managing Roles via API

Roles are managed as separate resources in Casdoor. When you retrieve a user object, the `Roles` field is populated dynamically by extending the user data with information from the Roles resource.

To assign or update roles for users, use the Roles API endpoints rather than the User API. You can manage roles through:

1. **Web UI**: Navigate to the Roles management page (e.g., `https://door.casdoor.com/roles`)
2. **API**: Use the role-specific endpoints documented in the Casdoor API reference

The Role resource allows you to define roles with specific users assigned to them. When a user is added to a role's `Users` array, that role will automatically appear in the user's `Roles` field when retrieving user data.

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。

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`: An array of users belonging to this permission

- `Roles`: An array of roles belonging to this permission
- `资源类型`
- `Resources`：资源的数组
- `Actions`：一个动作数组
- `效果`

Managing Permissions via API

Permissions are managed as separate resources in Casdoor. When you retrieve a user object, the `Permissions` field is populated dynamically by extending the user data with information from the Permissions resource.

To assign or update permissions for users, use the Permissions API endpoints rather than the User API. You can manage permissions through:

1. **Web UI**: Navigate to the Permissions management page (e.g., `https://door.casdoor.com/permissions`)
2. **API**: Use the permission-specific endpoints documented in the Casdoor API reference

The Permission resource allows you to define permissions with specific users and roles assigned to them. When a user is added to a permission's `Users` array or belongs to a role in the permission's `Roles` array, that permission will automatically appear in the user's `Permissions` field when retrieving user data.

Forms

The Forms feature is a tool, built to customize the display and configuration of list pages for the system's core entities.Administrators use the Forms feature to configure list page layouts and preview their final appearance.

1. Forms Overview

   The initial landing page of the Forms feature provides a comprehensive overview of all existing Forms in the system.

2. Form Type Selection

   To customize a specific list page, administrators first select the relevant `Form Type`—a classification that maps directly to the entity's list page (e.g., "User List Page", "Provider List Page").

3. Form Items Management

   - **Modify Existing Form Items:** Update properties like `Label` (e.g., rename "User ID" to "Employee ID"), `Width` (adjust pixel column width), and `Visible` (toggle column display on/off).
   - **Delete Unneeded Form Items:** Remove irrelevant columns to declutter the list page.
   - **Reorder Form Items:** Use "Move Up" or "Move Down" controls to adjust the left-to-right sequence of columns on the list page.

4. Form Preview

The preview page mirrors the live system's list page layout, ensuring that column order, labels, widths, and visibility align with requirements before finalizing changes.

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。

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。

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。

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

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。

> **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。

> **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
Save
Save & Exit
Copy signup page URL
Cancel
Organization :
built-in
Name :
invitation_lxxgdh
Display name :
New Invitation - lxxgdh
Code :
[a-z]2333
Default code :
a2333
Quota :
2

localhost:7001/signup/app-built-in?invitationCode=a2333
无痕模式
Casdoor
* Username:
Bob
* Password:
* Email:
Bob@qq.com
* Phone:
+1
15298989898
* Invitation code:
a2333
Sign Up
Have account? sign in now
Powered by Casdoor

演示

IP Whitelist

Overview

Support IP limitation for user entry pages.

Overview

Casdoor supports the ip whitelist function of the entry page. When a user accesses the entry page (login/signup/forget-password), Casdoor will decide whether to allow the user to access the entry page based on whether the client IP is in the whitelist. Here, we will show you how to enable the option to specify the ip whitelist function of the entry page at the user, application and organization levels.

Configuration

User Level

Casdoor will first determine whether the client address meets the user-level ip whitelist requirements.

If you want to specify user-level ip whitelist, you first need to add the "IP whitelist" account item on the edit page of the organization to which the user belongs. Then specify your ip whitelist by filling in the comma separated CIDR list, such as 192.168.1.0/24,25.112.0.0/16. If the ip whitelist is empty, it means there is no restriction on the client IP address.

> **信息**
>
> If you forget how to customize users' **account items**, Please refer to the Account Customization

Application Level

If the client IP address passes the user-level check, Casdoor will proceed to perform application-level check. You can specify the ip whitelist through the `IP whitelist` configuration option on the application edit page.

Organization Level

Organization-level check will be performed last. You can use the `IP whitelist` configuration option on the organization edit page to specify organization-level ip whitelist.

Here is a demo video that shows how to use ip whitelist:

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

Azure AD

Using Azure AD Syncer to synchronize users from Azure Active Directory

Keycloak

使用 Keycloak Syncer 同步 Keycloak

WeCom

Using WeCom Syncer to synchronize users from WeCom

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。

Casdoor supports multiple syncer types to import users from different sources:

- **Database**: Synchronize users from any database supported by Xorm (MySQL, PostgreSQL, SQL Server, Oracle, SQLite). See database syncer.
- **Azure AD**: Synchronize users from Azure Active Directory using Microsoft Graph API. See Azure AD syncer.
- **Keycloak**: Import users directly from Keycloak databases. See Keycloak syncer.
- **WeCom**: Fetch users from WeCom organizations via API. See WeCom syncer.

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		z6mive@
built-in	euler	2022-03-20T20:47:08+08:00		bb7831b4-0d24-4e96-b043-f8fd8d15eb	normal-user	123		Leonhard Euler			https://casbin.org		3dzw4j@
built-in	galieleo	2022-03-20T20:47:58+08:00		7920eb6c-f9f5-40ef-8e18-3ac99f49bdf5	normal-user	123		Galileo Galilei			https://casbin.org		8p4f38@
built-in	gauss	2022-03-20T20:48:33+08:00		f0c288f6-2c0d-479b-b545-cb4cf96db36	normal-user	123		Carl Friedrich Ga			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。

> **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。

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。

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。

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

Azure AD

Azure AD Syncer enables automatic user synchronization from Azure Active Directory (Microsoft Entra ID) to Casdoor. The syncer uses the Microsoft Graph API to fetch user information and keeps your user directory up to date.

Prerequisites

Before configuring the Azure AD syncer, you need to set up an application registration in Azure Portal with the appropriate permissions.

Step 1: Register an Application

Navigate to Azure Portal and register a new application:

1. Go to **Azure Active Directory** → **App registrations** → **New registration**
2. Enter a name for your application
3. Select the appropriate account type (typically "Accounts in this organizational directory only")
4. Click **Register**

Step 2: Create a Client Secret

After registration, create a client secret:

1. In your application, go to **Certificates & secrets**
2. Click **New client secret**
3. Add a description and select an expiration period

4. Click **Add** and copy the secret value immediately (it won't be shown again)

Step 3: Grant API Permissions

Configure the required Microsoft Graph API permissions:

1. Go to **API permissions** → **Add a permission**
2. Select **Microsoft Graph** → **Application permissions**
3. Add the `User.Read.All` permission
4. Click **Grant admin consent** for your organization

> **提示**
>
> The `User.Read.All` permission allows the syncer to read all user profiles in your Azure AD tenant.

Configuration

To create an Azure AD syncer in Casdoor:

1. Navigate to the **Syncers** tab
2. Click **Add** to create a new syncer
3. Fill in the following required fields:

Field	Description
Organization	The Casdoor organization where users will be imported
Name	A unique identifier for this syncer

Field	Description
Type	Select "Azure AD"
Tenant ID	Your Azure AD tenant ID (found in Azure Portal → Azure Active Directory → Overview)
Client ID	The Application (client) ID from your app registration
Client Secret	The client secret value you created earlier

Other database-related fields (Database type, Port, Database, Table) are not used for Azure AD syncer and can be left empty.

Field Mappings

The syncer automatically maps Azure AD user attributes to Casdoor user fields:

Azure AD Field	Casdoor Field	Description
id	Id	User's unique identifier
userPrincipalName	Name	User principal name
displayName	DisplayName	User's display name
givenName	FirstName	First name

Azure AD Field	Casdoor Field	Description
surname	LastName	Last name
mail	Email	Email address
mobilePhone	Phone	Mobile phone number
jobTitle	Title	Job title
officeLocation	Location	Office location
preferredLanguage	Language	Preferred language
accountEnabled	IsForbidden	Account status (inverted)

> **信息**
>
> The `accountEnabled` field is inverted when mapped to `IsForbidden`. When a user is disabled in Azure AD (`accountEnabled: false`), they will be marked as forbidden in Casdoor (`IsForbidden: true`).

Running the Syncer

After configuration:

1. Click **Test Connection** to verify your credentials and permissions
2. Enable the syncer by toggling **Is enabled**
3. Click **Sync** to trigger an immediate synchronization

4. The syncer will automatically fetch all users from your Azure AD tenant

The syncer handles pagination automatically, retrieving all users regardless of the total count.

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。

WeCom

WeCom Syncer allows you to automatically import users from your WeCom (企业微信) organization into Casdoor. The syncer fetches user information from all departments in your WeCom organization through the WeCom API and keeps the user data synchronized.

Configuration

以下字段是必需的:

- **Organization**: The Casdoor organization where users will be imported
- **Name**: A unique name for this syncer
- **Type**: Select "WeCom"
- **Corp ID**: Your WeCom organization's Corp ID
- **Corp Secret**: The secret for your WeCom application

Setup Steps

Step 1: Obtain WeCom Credentials

In your WeCom management platform, navigate to **My Company**, get **Corp ID** in **Company Information**.

In your Self-build App, get **App secret** (Corp Secret).

Optionally, in Sync of Contacts Management Tool, you can get **Sync of Contacts secret** for advanced configurations.

Step 2: Configure the Syncer in Casdoor

Go to Syncers tab, select **WeCom** type and fill in the required information:

- Enter your WeCom **Corp ID** in the Corp ID field
- Enter your application **Secret** (App secret) in the Corp Secret field

Then save the changes.

Click **Test Connection** to verify your credentials before enabling the syncer.

Field Mappings

The syncer automatically maps WeCom user fields to Casdoor user fields:

WeCom Field	Casdoor Field	Description
userid	Id	User's unique identifier
name	DisplayName	User's display name
email	Email	Email address
mobile	Phone	Phone number

WeCom Field	Casdoor Field	Description
avatar	Avatar	Profile picture URL
position	Title	Job title
gender	Gender	Gender (1=Male, 2=Female)
status/enable	IsForbidden	Account status

The syncer automatically handles user account status based on WeCom's status and enable fields:

- **Activated users** (status=1, enable=1): Normal Casdoor users
- **Disabled, not activated, or quit users** (status=2/4/5 or enable=0): Marked as forbidden in Casdoor

Running the Syncer

After configuration, you can:

- Enable **Is enabled** to allow automatic synchronization on schedule
- Use the **Sync** button to manually trigger a synchronization
- The syncer will fetch users from all departments and deduplicate them automatically

Certificates

Overview

Managing Certificates in Casdoor

Overview

Certificates (Certs) in Casdoor are used for signing and verifying tokens, as well as for securing communications in various integrations. They contain cryptographic keys that are essential for authentication and encryption processes.

Certificate Properties

Each certificate in Casdoor has the following properties:

- **Owner**: The organization that owns the certificate
- **Name**: The unique name of the certificate
- **CreatedTime**: When the certificate was created
- **DisplayName**: A human-readable name for the certificate
- **Scope**: The scope of the certificate (e.g., `JWT`, `SAML`, `Payment`)
- **Type**: The type of certificate (e.g., `x509`, `Payment`)
- **CryptoAlgorithm**: The cryptographic algorithm used (e.g., `RS256`, `RSA`)
- **BitSize**: The key size in bits (e.g., 2048, 4096)
- **ExpireInYears**: The expiration period of the certificate in years
- **Certificate**: The public certificate content
- **PrivateKey**: The private key content (stored securely)

Certificate Scopes

Certificates in Casdoor serve different purposes based on their scope:

JWT Certificates

JWT certificates are used to sign and verify JSON Web Tokens (JWT) for authentication. When a user logs in through Casdoor, the access token is signed using the private key of the JWT certificate.

Use cases:

- Signing access tokens for OAuth/OIDC authentication
- Verifying tokens in backend applications
- Securing API communications

The JWT public key is required when configuring Casdoor SDK in your application. You can find and download the public key from the certificate edit page.

Once you have created a JWT certificate, you can select it in your application settings:

SAML Certificates

SAML certificates are used for signing and encrypting SAML assertions in Single Sign-On (SSO) integrations.

Use cases:

- SAML-based SSO integrations
- Signing SAML responses
- Encrypting SAML assertions for security

When configuring SAML, you need to create certificates for both the Identity Provider (IdP) and Service Provider (SP) to ensure secure communication.

For more information about SAML configuration, see the SAML documentation.

Payment Certificates

Payment certificates are used to secure payment transactions with payment providers like Alipay and WeChat Pay.

Use cases:

- Securing API communications with payment gateways
- Signing payment requests
- Verifying payment callbacks

For example, when integrating with Alipay, you need to create certificates for:

- App Cert: Contains the application's public certificate and private key
- Root Cert: Contains the payment provider's certificates

For more details, refer to the Alipay Payment Provider and WeChat Pay Provider documentation.

Creating a Certificate

To create a new certificate in Casdoor:

1. Navigate to the **Certs** page in the Casdoor admin console
2. Click the **Add** button to create a new certificate
3. Fill in the required fields:
   - **Name**: A unique identifier for the certificate
   - **Display Name**: A human-readable name
   - **Scope**: Select the appropriate scope (JWT, SAML, Payment)
   - **Type**: Select the certificate type
   - **Crypto Algorithm**: Choose the cryptographic algorithm (e.g., RS256 for JWT)
   - **Bit Size**: Set the key size (typically 2048 or 4096 bits)
   - **Expire In Years**: Set the expiration period

4. You can either:
    - Generate a new certificate automatically by saving the form
    - Upload an existing certificate and private key

Using Certificates

In Applications

After creating a JWT certificate, you need to associate it with your application:

1. Go to the **Applications** page
2. Edit your application
3. In the **Cert** field, select the certificate you created
4. Save the application

The certificate will now be used to sign all tokens issued by that application.

In SDK Configuration

When configuring Casdoor SDK in your backend application, you need to provide the public key from the JWT certificate:

```
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
```

You can download the public key from the certificate edit page in Casdoor.

For more information about SDK configuration, see the SDK documentation.

In Payment Providers

When setting up payment providers, you need to configure the appropriate certificates:

1. Create the required certificates (e.g., App Cert and Root Cert for Alipay)
2. Go to the **Providers** page
3. Edit or create your payment provider
4. In the **Cert** field, select the certificate you created
5. Save the provider

Best Practices

- **Key Size**: Use at least 2048-bit keys for RSA certificates. For higher security, consider 4096-bit keys.
- **Expiration**: Set appropriate expiration periods. For production environments, 1-5 years is typical.
- **Security**: Keep private keys secure and never expose them in client-side code or public repositories.
- **Rotation**: Regularly rotate certificates before they expire to maintain security.
- **Backup**: Keep secure backups of your certificates and private keys.
- **Separate Certificates**: Use different certificates for different purposes (JWT, SAML, Payment) to limit the impact of potential key compromise.

Certificate Management

You can manage certificates through:

- **Web UI**: The Casdoor admin console provides a user-friendly interface for certificate management
- **API**: Use the Casdoor REST API to programmatically manage certificates
- **Data Initialization**: Certificates can be included in initialization files for automated deployments

For more information about data initialization, see the Data Initialization documentation.

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。

Access Token and ID Token

In Casdoor, the `access_token` and `id_token` are **identical**. Both tokens contain the same JWT payload with user information and claims. This is a design choice in Casdoor that simplifies token management.

This approach means:

- Both tokens contain the same user information and custom claims
- Both tokens can be used interchangeably for authentication and authorization
- The token format and expiration settings apply to both tokens equally
- You cannot configure separate claims for `access_token` and `id_token`

Token Fields

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令牌的选项：

- `JWT`
- `JWT-Empty`
- `JWT-Custom`
- `JWT-Standard`

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）。 JWT-Standard will generate a token with some standard OIDC token fields include email, phone, gender and Address (Address value in other format is not standard).

Webhooks

Webhooks Overview

Configuring Webhooks in Casdoor

Webhooks Overview

概述

Casdoor provides an event-driven system that allows you to integrate with external applications using webhooks. Webhooks enable real-time communication by sending HTTP `POST` requests with a JSON payload to a configured endpoint whenever a specified event occurs. This allows your application to react to Casdoor events such as user sign-ups, logins, logouts, and profile updates.

How Webhooks Work

When an event is triggered in Casdoor:

1. Casdoor sends a `POST` request to the specified webhook URL.
2. The request contains a JSON payload with event details.
3. Your application processes the payload and executes relevant actions based on the event type.

Supported Events

Casdoor webhooks support various user-related events, which are stored in the `action` field of the request payload.

Event	Description
`signup`	Triggered when a user signs up
`login`	Triggered when a user logs in
`logout`	Triggered when a user logs out
`update`	Triggered when user details are updated

Setting Up a Webhook

To configure a webhook in Casdoor:

1. **Navigate to the Casdoor Webhooks Section:**
    - Open your Casdoor instance.
    - Go to **Settings** > **Webhooks**.
2. **Create a New Webhook:**
    - Click on **Add Webhook**.
    - Enter the **Webhook URL** where Casdoor should send event data.
    - Select the events you want to listen to (e.g., `signup`, `login`).
    - (Optional) Add custom headers for authentication.
3. **Save the Webhook Configuration:**
    - Once saved, Casdoor will start sending event notifications to the specified endpoint.

Example Webhook Payload

Here's an example of a JSON payload sent to your webhook when a user logs in:

```
{
  "event": "login",
  "timestamp": 1709452800,
  "user": {
    "id": "12345",
    "username": "johndoe",
    "email": "johndoe@example.com"
  }
}
```

Your application should parse this payload and perform necessary actions, such as logging the event or notifying another service.

Testing Your Webhook

Before deploying your webhook integration, you can test it using tools like:

- Beeceptor – Allows you to create a custom webhook URL and inspect incoming requests.
- Webhook.site – Provides an instant webhook endpoint for testing.

Example Test with Beeceptor

1. Visit Beeceptor and create a new endpoint.
2. Copy the generated webhook URL and configure it in Casdoor.
3. Trigger a test event (e.g., log in to Casdoor).

4. Check Beeceptor's dashboard to inspect the received request.

Handling Webhooks in Your Application

Your server should be able to process incoming webhook requests. Below is a simple example in Node.js:

```
const express = require('express');
const app = express();

app.use(express.json());

app.post('/webhook', (req, res) => {
  console.log('Received webhook:', req.body);
  res.status(200).send('Webhook received');
});

app.listen(3000, () => console.log('Server running on port 3000'));
```

Conclusion

Casdoor webhooks provide a powerful way to integrate with external applications by enabling event-driven interactions. Whether you need to sync user data, trigger notifications, or update external systems, webhooks allow seamless automation.

To ensure a smooth integration, always validate incoming requests and test your webhooks with tools like Beeceptor or Webhook.site before deploying them in production.

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。

LDAPs :

LDAPs Add

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

配置连接到LDAP服务器

Edit LDAP | Save | Save & Exit | Sync LDAP

Organization : built-in

ID : 691edec0-f1ab-4e23-8f9f-a824a383032f

Server name : Example LDAP Server

Server host : example.com

Server port : 389

Enable SSL :

Base DN : ou=built-in,dc=example,dc=com

Search Filter : (objectClass=posixAccount)

Filter fields : uid × Email ×

Admin : cn=admin,dc=example,dc=com

Admin Password : ••••••

Auto Sync : 0 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: `Example LDAP Server`

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`。

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。

Default group

Group to which users belong after synchronization.

> **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。 You can change the default port by modifying the `ldapServerPort` value in conf/app.conf.

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。

- To search for a certain user, such as `Alice` under the `built-in` organization, you should use a `DN` like this: `ou=built-in,dc=example,dc=com`, and add `cn=Alice` in the Filter field.
- To search for all users under a certain organization, such as all users in `built-in`, you should use a `DN` like this: `ou=built-in,dc=example,dc=com`, and add `cn=*` in the Filter field.
- To search for all users in all organizations (assuming the user has sufficient permissions), you should use a `DN` like this: `ou=*,dc=example,dc=com`, and add `cn=*` in the Filter field.
- To search for all users in a specific group, you should use a filter query like this: `(memberOf=organization_name/group_name)` in the Filter field.

Supported RFC-Style Features

Partial Root DSE Query Support

The Root DSE (baseDN="") provides directory capabilities.

- Query namingContexts (organization list): `ldapsearch -x -H ldap://<casdoor-host>:389 -D "cn=admin,ou=built-in" -w <passwd> -b "" -s base "(objectClass=*)" namingContexts`
  Returns visible organization DNs.

- Query subschemaSubentry: `ldapsearch -x -H ldap://<casdoor-host>:389 -D "cn=admin,ou=built-in" -w <passwd> -b "" -s base "(objectClass=*)" subschemaSubentry`
  Returns `subschemaSubentry: cn=Subschema`.

Schema Query Support

Query objectClasses: `ldapsearch -x -H ldap://<casdoor-host>:389 -D "cn=admin,ou=built-in" -w <passwd> -b "cn=Subschema" -s base "(objectClass=*)" objectClasses`
Returns definitions for posixAccount and posixGroup.

POSIX Filters

- `(objectClass=posixAccount)` returns user list.
- `(objectClass=posixGroup)` returns group list under organization (e.g., `ldapsearch -x -H ldap://<casdoor-server>:389 -D "cn=admin,ou=built-in" -w <passwd> -b "ou=<org>" "(objectClass=posixGroup)"`).

Note: `(objectClass=posixGroup)` Does not support combined searches like `(&(objectClass=posixGroup)(cn=<group>))`. Please use `memberOf` for searching members in a group.

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。

Congiure

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/
MASTERSOFT
DIALWAYS
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
/scim/ServiceProviderConfig	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。
/scim/Schemas	GET	提供有关服务提供商模式的详细信息。
/scim/ResourceTypes	GET	指定每个资源的元数据。
/scim/Users/:id	GET	使用资源标识符`id`检索用户。
/scim/Users	GET	使用查询参数查询用户（目前只支持`startIndex`和`count`）。
/scim/Users	POST	创建用户。
/scim/Users/:id	PUT	使用资源标识符`id`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`id`修改用户。
/scim/Users/:id	DEL	删除具有资源标识符`id`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	UpdatedTime
meta.version	UpdatedTime
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

10 个项目

Java

17 个项目

🗂️ JavaScript

2 个项目

🗂️ Lua

1 个项目

🗂️ PHP

6 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/ngx_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。

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）。

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。

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。

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）。

在**注册应用程序**中选择NGINX Plus。

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。

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）。

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。

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）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/`。

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。

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）。 为了便于解释，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`值逐个字符进行比较。

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：

iOS	Android

- 使用Casdoor网页登录:

iOS	Android
admin *** Login Casdoor Login	admin *** Login Casdoor Login

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

Traefik

Using Casdoor for authentication with Traefik reverse proxy

📄 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。

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：

```python
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kubectl访问您的API服务器。 尝试运行一个测试命令。

```
kubectl cluster-info
```

命令行参数方法

或者，您可以通过直接将`id_token`添加到kubectl的命令行参数中进行身份验证。

```
kubectl --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 URLs Add

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。

你会看到身份提供商部分。 在添加部分，从选项中选择OpenID Connect。

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 *

casdoor

Unique name of the new identity provider. This cannot be changed later.

Client ID *

2452f2b5abb6ff131199

Client secret *

••

Issuer URL *

https://demo.casdoor.com

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。

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。

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：

`CASDOOR_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`CASDOOR_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： `<CASDOOR_HOSTNAME>/oauth/callback`。
3. 将重定向URL添加到Casdoor应用程序：

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：

2. 配置此应用程序：

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。

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仪表板。 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 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Cassdoor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：

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：

按“添加认证源”按钮并填写类似的表单。

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：

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：

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：

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。

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。

   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_PASSSWORD=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。

Traefik

Using Casdoor for authentication with Traefik

Traefik is a modern HTTP reverse proxy and load balancer that makes deploying microservices easy. This document shows how to use Casdoor as an authentication provider with Traefik using the traefik-casdoor-auth middleware.

The Traefik Casdoor Auth middleware allows you to protect your services behind Traefik with Casdoor authentication, providing a seamless Single Sign-On (SSO) experience.

Prerequisites

Before you begin, ensure you have:

- Traefik v2.x or v3.x installed and running
- A Casdoor instance deployed and accessible
- Docker and Docker Compose (if using the containerized approach)

Step 1: Deploy Casdoor

First, deploy Casdoor if you haven't already.

You can refer to the Casdoor official documentation for Server Installation.

After a successful deployment, make sure that:

- The Casdoor server is running and accessible
- You can log in to the Casdoor admin interface
- Test the login functionality by entering `admin` and `123`

Step 2: Configure Casdoor Application

1. Create a new Casdoor application or use an existing one.

2. Add your redirect URL. The redirect URL should be in the format:

   ```
   http://<your-domain>/callback
   ```

   For example: `http://localhost:8080/callback`

3. Note down the following values from your application settings:

   - Client ID
   - Client Secret
   - Organization Name
   - Application Name

1. Configure your application to use the appropriate authentication providers as needed.

Step 3: Deploy traefik-casdoor-auth Middleware

There are two ways to deploy the Traefik Casdoor Auth middleware:

Option 1: Using Docker Compose (Recommended)

Create a `docker-compose.yml` file:

```yaml
version: '3'

services:
  traefik:
    image: traefik:v2.10
    container_name: traefik
    restart: unless-stopped
    command:
      - "--api.insecure=true"
      - "--providers.docker=true"
```

Replace the placeholder values:

- `<your-client-id>`: Your Casdoor application client ID
- `<your-client-secret>`: Your Casdoor application client secret
- `<your-organization>`: Your Casdoor organization name
- `<your-application>`: Your Casdoor application name

Start the services:

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

Option 2: Using Binary Deployment

1. Download the latest release from the traefik-casdoor-auth releases page.

2. Create a configuration file `config.yaml`:

```yaml
casdoor:
  endpoint: "http://localhost:8000"
  clientId: "<your-client-id>"
  clientSecret: "<your-client-secret>"
  organizationName: "<your-organization>"
  applicationName: "<your-application>"

server:
  port: 8080
  callbackUrl: "http://localhost/callback"
```

1. Run the middleware:

```
./traefik-casdoor-auth --config config.yaml
```

1. Configure Traefik to use the middleware. Add to your Traefik dynamic configuration:

```yaml
http:
  middlewares:
    casdoor-auth:
      forwardAuth:
        address: "http://localhost:8080/verify"
        authResponseHeaders:
          - "X-Forwarded-User"

  routers:
    my-protected-service:
      rule: "Host(`example.com`)"
      middlewares:
        - casdoor-auth
      service: my-service
```

Step 4: Test the Integration

1. Access your protected service through Traefik (e.g., `http://localhost` if using the Docker Compose example).

2. You should be redirected to the Casdoor login page.

3. Log in with your Casdoor credentials.

4. After successful authentication, you will be redirected back to your protected service.

5. The middleware will set the `X-Forwarded-User` header with the authenticated user's information, which your backend service can use.

Configuration Options

The traefik-casdoor-auth middleware supports the following environment variables:

Variable	Description	Required
`CASDOOR_ENDPOINT`	URL of your Casdoor instance	Yes
`CLIENT_ID`	Casdoor application client ID	Yes
`CLIENT_SECRET`	Casdoor application client secret	Yes
`CASDOOR_ORGANIZATION`	Casdoor organization name	Yes
`CASDOOR_APPLICATION`	Casdoor application name	Yes
`CALLBACK_URL`	OAuth callback URL	Yes
`JWT_SECRET`	Secret for JWT token signing (auto-generated if not set)	No
`SESSION_TIMEOUT`	Session timeout in seconds (default: 3600)	No
`LOG_LEVEL`	Logging level: debug, info, warn, error (default: info)	No

Advanced Configuration

Using HTTPS

For production deployments, it's recommended to use HTTPS. Update your Traefik configuration to include TLS:

```
services:
  traefik:
    command:
      - "--entrypoints.web.address=:80"
      - "--entrypoints.websecure.address=:443"
      - "--certificatesresolvers.myresolver.acme.tlschallenge=true"
      - "--certificatesresolvers.myresolver.acme.email=your-
email@example.com"
      - "--
certificatesresolvers.myresolver.acme.storage=/letsencrypt/
acme.json"
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./letsencrypt:/letsencrypt
```

And update your service router configuration:

```
labels:
  - "traefik.http.routers.whoami.entrypoints=websecure"
  - "traefik.http.routers.whoami.tls.certresolver=myresolver"
```

Customizing Session Behavior

You can customize session timeout and other behaviors:

```
environment:
  - SESSION_TIMEOUT=7200  # 2 hours
  - LOG_LEVEL=debug
```

Troubleshooting

Redirect Loop

If you experience a redirect loop:

1. Verify that the `CALLBACK_URL` matches the redirect URL configured in your Casdoor application.
2. Check that cookies are enabled in your browser.
3. Ensure the middleware can reach the Casdoor endpoint.

Authentication Fails

If authentication fails:

1. Check that the `CLIENT_ID` and `CLIENT_SECRET` are correct.
2. Verify that the Casdoor organization and application names are correct.
3. Check the middleware logs for detailed error messages: `docker logs casdoor-auth`

502 Bad Gateway

If you see a 502 error:

1. Ensure the casdoor-auth service is running: `docker ps`
2. Check that all services are on the same Docker network.
3. Verify the ForwardAuth address is correct in the Traefik configuration.

Resources

- Traefik Casdoor Auth GitHub Repository
- Traefik ForwardAuth Middleware Documentation
- Casdoor Documentation

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：

`CASDOOR_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：

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 castor 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, fullFilePath, 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 `http://localhost:8000`
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：

```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。

```java
@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。

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。

随后您可以看到Cassdoor统一的登录平台。

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。

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）。 ```

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://`CASDOOR_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

```yaml
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://CASDOOR_HOSTNAME:7001/login/oauth/
authorize
            token-uri: http://CASDOOR_HOSTNAME:8000/api/login/oauth/
access_token
            user-info-uri: http://CASDOOR_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>   </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。

登录后，页面将重定向您到`/foos`。

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。

- 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，

点击`Casdoor Login`按钮，页面将重定向到Casdoor的登录页面。

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`。

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`

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。

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`。

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`

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Openld 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。

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`

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`。

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。

username, Email or phone

Password

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：

- `CASDOOR_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`CASDOOR_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：

3. 添加一个配置，并在认证方法中选择OpenID连接单点登录

4. 找到重定向URL：

5. 添加一个重定向URL：

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://`CASDOOR_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 *

casdoor

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 *

https://demo.casdoor.com

your casdoor url

The complete URL of the OpenID Provider. Needs to be unique.

Client ID *

642ec5d6779a2f0e879d

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 *

${preferred_username}

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

https://test.v2tl.com/plugins/servlet/oidc/callback

Copy it to casdoor

Location where the client is sent to after successful account authentication.

Initiate login URL

https://test.v2tl.com/plugins/servlet/oidc/initiate-login

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.

- [x] 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

- [x] 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.

- [x] Show IdP on the login page

Login button text *

casdoor

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。

You have temporary access to administrative functions. Drop access if you no longer require it. For more information, refer to the documentation.

Jira Software | Dashboards | Projects | Issues | Boards | Plans | Create | Search

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	✓	•••
casdoor	OpenID Connect	26 April 2023 7:20 PM	✓	•••

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:

`CASDOOR_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`CASDOOR_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：

6. 按照以下方式配置应用程序：

- `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。

Jira Software
Dashboards
Search
Log In
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?
Atlassian Jira Project Management Software · About Jira · Report a problem
Powered by a free Atlassian Jira evaluation license. Please consider purchasing it today.
ATLASSIAN

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：

- `CASDOOR_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`CASDOOR_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://`CASDOOR_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：

2. 配置应用程序：

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：

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.
Powered by Atlassian Confluence 8.1.1 · Report a bug · Atlassian News
ATLASSIAN
localhost:8090/plugins/servlet/oauth/auth?return_to=http://localhost:8090&idp=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：

   。

2. 在编辑组织时，确保选择正确的密码类型：

   。

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。

3. 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。

4. 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
">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。

5. 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;
```

```java
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;
}
```

```java
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.recordLogininfor(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

步骤 2.2：创建一个应用

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**

3. 在证书编辑页面，您可以查看您的证书

Certificate ⓘ :

Copy certificate　Download certificate


```
-----BEGIN CERTIFICATE-----
MIIE+TCCAuGgAwIBAgIDAeJAMA0GCSqGSIb3DQEBCwUAMDYxHTAbBgNVBAoTFENh
c2Rvb3IgT3JnYW5pemF0aW9uMRUwEwYDVQQDEwxDYXNkb29yIENlcnQwHhcNMjEx
MDE1MDgxMTUyWhcNNDExMDE1MDgxMTUyWjA2MR0wGwYDVQQKExRDYXNkb29yIE9y
Z2FuaXphdGlvbjEVMBMGA1UEAxMMQ2FzZG9vciBDZXJ0MIICIjANBgkqhkiG9w0B
AQEFAAOCAg8AMIICCgKCAgEAsInpb5E1/ym0f1RfSDSSE8IR7y+lw+RJjI74e5ej
rq4b8zMYk7HeHCyZr/hmNEwEVXnhXu1P0mBeQ5ypp/QGo8vgEmjAETNmzkI1NjOQ
CjCYwUrasO/f/MnI1C0j13vx6mV1kHZjSrKsMhYY1vaxTEP3+VB8Hjg3MHFWrb07
uvFMCJe5W8+0rKErZCKTR8+9VB3janeBz//zQePFVh79bFZate/hLirPK0Go9P1g
OvwIoC1A3sarHTP4Qm/LQRt0rHqZFybdySpyWAQvhNaDFE7mTstRSBb/wUjNCUBD
PTSLVjC04WlISf6Nkfx0Z7KvmbPstSj+btvcqsvRAGtvdsB9h62Kptjs1Yn7GAuo
I3qt/4zoKbiURYxkQJXIvwCQsEftUuk5ew5zuPSlDRLoLByQTLbx0JqLAFNfW3g/
pzSDjgd/60d6HTmvbZni4SmjdyFhXCDb1Kn7N+xTojnfaNkwep2REV+RMc0fx4Gu
hRsnLsmkmUDeyIZ9aBL9oj11YEQfM2JZEq+RVtUx+wB4y8K/tD1bcY+IfnG5rBpw
IDpS262boq4SRSvb3Z7bB0w4ZxvOfJ/1VLoRftjPbLIf0bhfr/AeZMHpIKOXvfz4
yE+hqzi68wdF0VR9xYc/RbSAf7323OsjYnjjEgInUtRohnRgCpjIk/Mt2Kt84Kb0
wn8CAwEAAaMQMA4wDAYDVR0TAQH/BAIwADANBgkqhkiG9w0BAQsFAAOCAgEAn2If
DKkLX+F1vKRO/5gJ+Plr8P5NKuQkmwH97b8CS2gS1phDyNgIc4/LSdzuf4Awe6ve
C06lVdWSIis8UPUPdjmT2uMPSNjwLxG3QsrimMURNwFlLTfRem/heJe0Zgur9J1M
8haawdSdJjH2RgmFoDeE2r8NVRfhbR8KnCO1ddTJKuS1N0/irHz21W4jt4rxzCvl
2nR42Fybap3O/g2JXMhNNROwZmNjgpsF7XVENCSuFO1jTywLaqjuXCg54IL7XVLG
omKNNNcc8h1FCeKj/nnbGMhodnFWKDTsJcbNmcOPNHo6ixzqMy/Hqc+mWYv7maAG
Jtevs3qgMZ8F9Qzr3HpUc6R3ZYYWDY/xxPisuKftOPZgtH979XC4mdf0WPnOBLqL
2DJ1zaBmjiGJolvb7XNVKcUfDXYw85ZTZQ5b9cII4e+6bmyWqQltlwt+Ati/uFEV
XzCj70B4lALX6xau1kLEpV9O1GERizYRz5P9NJNA7KoO5AVMp9w0DQTkt+LbXnZE
HHnWKy8xHQKZF9sR7YBPGLs/Ac6tviv5Ua15OgJ/8dLRZ/veyFfGo2yZsI+hKVU5
nCCJHBcAyFnm1hdvdwEdH33jDBjNB6ciotJZrf/3VYaIWSalADosHAgMWfXuWP+h
8XKXmzlxuHbTMQYtZPDgspS5aK+S4Q9wb8RRAYo=
-----END CERTIFICATE-----
```

步骤3：在ShenYu中使用Casdoor插件

1. 在ShenYu中配置Casdoor插件

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-----\n
MIIE+TCCAuGgAwIBAgIDAeJAMA0GCSqGSIb3DQEBCwUAMDYxHTAbBgNVBAoTFENh\n
c2Rvb3IgT3JnYW5pemF0aW9uMRUwEwYDVQQDEwxDYXNkb29yIENlcnQwHhcNMjEx\n
MDE1MDgxMTUyWhcNNDExMDE1MDgxMTUyWjA2MR0wGwYDVQQKExRDYXNkb29yIE9y\n
Z2FuaXphdGlvbjEVMBMGA1UEAxMMQ2FzZG9vciBDZXJ0MIICIjANBgkqhkiG9w0B\n
AQEFAAOCAg8AMIICCgKCAgEAslnpb5E1/ym0f1RfSDSSE8IR7y+Iw+RJjI74e5ej\n
rq4b8zMYk7HeHCyZr/hmNEwEVXnhXu1P0mBeQ5ypp/QGo8vgEmjAETNmzkl1NjOQ\n
CjCYwUrasO/f/Mnl1C0j13vx6mV1kHZjSrKsMhYY1vaxTEP3+VB8Hjg3MHFWrb07\n
uvFMCJe5W8+0rKErZCKTR8+9VB3janeBz//zQePFVh79bFZate/hLirPK0Go9P1g\n
OvwIoC1A3sarHTP4Qm/LQRt0rHqZFybdySpyWAQvhNaDFE7mTstRSBb/wUjNCUBD\n
PTSLVjC04WIISf6Nkfx0Z7KvmbPstSj+btvcqsvRAGtvdsB9h62Kptjs1Yn7GAuo\n
I3qt/4zoKbiURYxkQJXlvwCQsEftUuk5ew5zuPSIDRLoLByQTLbx0JqLAFNfW3g/\n
pzSDjgd/60d6HTmvbZni4SmjdyFhXCDb1Kn7N+xTojnfaNkwep2REV+RMc0fx4Gu\n
hRsnLsmkmUDeylZ9aBL9oj11YEQfM2JZEq+RVtUx+wB4y8K/tD1bcY+IfnG5rBpw\n
IDpS262boq4SRSvb3Z7bB0w4ZxvOfJ/1VLoRftjPbLIf0bhfr/AeZMHplKOXvfz4\n
yE+hqzi68wdF0VR9xYc/RbSAf7323OsjYnjjEgInUtRohnRgCpjlk/Mt2Kt84Kb0\n
wn8CAwEAAaMQMA4wDAYDVR0TAQH/BAIwADANBgkqhkiG9w0BAQsFAAOCAgEAn2If\n
DKkLX+F1vKRO/5gJ+Plr8P5NKuQkmwH97b8CS2gS1phDyNgIc4/LSdzuf4Awe6ve\n
C06IVdWSIis8UPUPdjmT2uMPSNjwLxG3QsrimMURNwFILTfRem/heJe0Zgur9J1M\n
8haawdSdJjH2RgmFoDeE2r8NVRfhbR8KnCO1ddTJKuS1N0/irHz21W4jt4rxzCvl\n
2nR42Fybap3O/g2JXMhNNROwZmNjgpsF7XVENCSuFO1jTywLaqjuXCg54IL7XVLG\n
omKNNNcc8h1FCeKj/nnbGMhodnFWKDTsJcbNmcOPNHo6ixzqMy/Hqc+mWYv7maAG\n
Jtevs3qgMZ8F9Qzr3HpUc6R3ZYYWDY/xxPisuKftOPZgtH979XC4mdf0WPnOBLqL\n
2DJ1zaBmjiGJoIvb7XNVKcUfDXYw85ZTZQ5b9cII4e+6bmyWqQltIwt+Ati/uFEV\n
XzCj70B4IALX6xau1kLEpV9O1GERizYRz5P9NJNA7KoO5AVMp9w0DQTkt+LbXnZE\n
HHnWKy8xHQKZF9sR7YBPGLs/Ac6tviv5Ua15OgJ/8dLRZ/veyFfGo2yZsI+hKVU5\n
nCCJHBcAyFnm1hdvdwEdH33jDBjNB6ciotJZrf/3VYaIWSaIADosHAgMWfXuWP+h\n
8XKXmzIxuHbTMQYtZPDgspS5aK+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

hi! null! I'm Shenyu-Gateway System. Welcome!

3.3 在Headers中携带令牌

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**

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+TCCAuGgAwIBAgIDAeJAMA0GCSqGSIb3DQEBCwUAMDYxHTAbBgNVBAoTFENh
c2Rvb3IgT3JnYW5pemF0aW9uMRUwEwYDVQQDEwxDYXNkb29yIENIcnQwHhcNMjEx
MDE1MDgxMTUyWhcNNDExMDE1MDgxMTUyWjA2MR0wGwYDVQQKExRDYXNkb29yIE9y
Z2FuaXphdGlvbjEVMBMGA1UEAxMMQ2FzZG9vciBDZXJ0MIICIjANBgkqhkiG9w0B
AQEFAAOCAg8AMIICCgKCAgEAsInpb5E1/ym0f1RfSDSSE8IR7y+Iw+RJjI74e5ej
rq4b8zMYk7HeHCyZr/hmNEwEVXnhXu1P0mBeQ5ypp/QGo8vgEmjAETNmzkI1NjOQ
CjCYwUrasO/f/MnI1C0j13vx6mV1kHZjSrKsMhYY1vaxTEP3+VB8Hjg3MHFWrb07
uvFMCJe5W8+0rKErZCKTR8+9VB3janeBz//zQePFVh79bFZate/hLirPK0Go9P1g
OvwIoC1A3sarHTP4Qm/LQRt0rHqZFybdySpyWAQvhNaDFE7mTstRSBb/wUjNCUBD
PTSLVjC04WIISf6Nkfx0Z7KvmbPstSj+btvcqsvRAGtvdsB9h62Kptjs1Yn7GAuo
I3qt/4zoKbiURYxkQJXlvwCQsEftUuk5ew5zuPSIDRLoLByQTLbx0JqLAFNfW3g/
pzSDjgd/60d6HTmvbZni4SmjdyFhXCDb1Kn7N+xTojnfaNkwep2REV+RMc0fx4Gu
hRsnLsmkmUDeyIZ9aBL9oj11YEQfM2JZEq+RVtUx+wB4y8K/tD1bcY+IfnG5rBpw
IDpS262boq4SRSvb3Z7bB0w4ZxvOfJ/1VLoRftjPbLIf0bhfr/AeZMHpIKOXvfz4
yE+hqzi68wdF0VR9xYc/RbSAf7323OsjYnjjEgInUtRohnRgCpjIk/Mt2Kt84Kb0
wn8CAwEAAaMQMA4wDAYDVR0TAQH/BAIwADANBgkqhkiG9w0BAQsFAAOCAgEAn2If
DKkLX+F1vKRO/5gJ+Plr8P5NKuQkmwH97b8CS2gS1phDyNgIc4/LSdzuf4Awe6ve
C06IVdWSIis8UPUPdjmT2uMPSNjwLxG3QsrimMURNwFILTfRem/heJe0Zgur9J1M
8haawdSdJjH2RgmFoDeE2r8NVRfhbR8KnCO1ddTJKuS1N0/irHz21W4jt4rxzCvI
2nR42Fybap3O/g2JXMhNNROwZmNjgpsF7XVENCSuFO1jTywLaqjuXCg54IL7XVLG
omKNNNcc8h1FCeKj/nnbGMhodnFWKDTsJcbNmcOPNHo6ixzqMy/Hqc+mWYv7maAG
Jtevs3qgMZ8F9Qzr3HpUc6R3ZYYWDY/xxPisuKftOPZgtH979XC4mdf0WPnOBLqL
2DJ1zaBmjiGJoIvb7XNVKcUfDXYw85ZTZQ5b9cII4e+6bmyWqQltlwt+Ati/uFEV
XzCj70B4IALX6xau1kLEpV9O1GERizYRz5P9NJNA7KoO5AVMp9w0DQTkt+LbXnZE
HHnWKy8xHQKZF9sR7YBPGLs/Ac6tviv5Ua15OgJ/8dLRZ/veyFfGo2yZsI+hKVU5
nCCJHBcAyFnm1hdvdwEdH33jDBjNB6ciotJZrf/3VYaIWSaIADosHAgMWfXuWP+h
8XKXmzIxuHbTMQYtZPDgspS5aK+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\
c2Rvb3IgT3JnYW5pemF0aW9uMRUwEwYDVQQDEwxDYXNkb29yIENlcnQwHhcNMjEx\n\
MDE1MDgxMTUyWhcNNDExMDE1MDgxMTUyWjA2MR0wGwYDVQQKExRDYXNkb29yIE9y\n\
Z2FuaXphdGlvbjEVMBMGA1UEAxMMQ2FzZG9vciBDZXJ0MIICIjANBgkqhkiG9w0B\n\
AQEFAAOCAg8AMIICCgKCAgEAsInpb5E1/ym0f1RfSDSSE8IR7y+lw+RJjI74e5ej\n\
rq4b8zMYk7HeHCyZr/hmNEwEVXnhXu1P0mBeQ5ypp/QGo8vgEmjAETNmzkI1NjOQ\n\
CjCYwUrasO/f/MnI1C0j13vx6mV1kHZjSrKsMhYY1vaxTEP3+VB8Hjg3MHFWrb07\n\
uvFMCJe5W8+0rKErZCKTR8+9VB3janeBz//zQePFVh79bFZate/hLirPK0Go9P1g\n\
OvwIoC1A3sarHTP4Qm/LQRt0rHqZFybdySpyWAQvhNaDFE7mTstRSBb/wUjNCUBD\n\
PTSLVjC04WllSf6Nkfx0Z7KvmbPstSj+btvcqsvRAGtvdsB9h62Kptjs1Yn7GAuo\n\
I3qt/4zoKbiURYxkQJXIvwCQsEftUuk5ew5zuPSlDRLoLByQTLbx0JqLAFNfW3g/\n\
pzSDjgd/60d6HTmvbZni4SmjdyFhXCDb1Kn7N+xTojnfaNkwep2REV+RMc0fx4Gu\n\
hRsnLsmkmUDeyIZ9aBL9oj11YEQfM2JZEq+RVtUx+wB4y8K/tD1bcY+IfnG5rBpw\n\
IDpS262boq4SRSvb3Z7bB0w4ZxvOfJ/1VLoRftjPbLIf0bhfr/AeZMHpIKOXvfz4\n\
yE+hqzi68wdF0VR9xYc/RbSAf7323OsjYnjjEgInUtRohnRgCpjIk/Mt2Kt84Kb0\n\
wn8CAwEAAaMQMA4wDAYDVR0TAQH/BAIwADANBgkqhkiG9w0BAQsFAAOCAgEAn2lf\n\
DKkLX+F1vKRO/5gJ+Plr8P5NKuQkmwH97b8CS2gS1phDyNgIc4/LSdzuf4Awe6ve\n\
C06lVdWSIis8UPUPdjmT2uMPSNjwLxG3QsrimMURNwFlLTfRem/heJe0Zqur9J1M\n\
8haawdSdJjH2RgmFoDeE2r8NVRfhbR8KnCO1ddTJKuS1N0/irHz21W4jt4rxzCvl\n\
2nR42Fybap3O/g2JXMhNNROwZmNjgpsF7XVENCSuFO1jTywLaqjuXCg54IL7XVLG\n\
omKNNNcc8h1FCeKj/nnbGMhodnFWKDTsJcbNmcOPNHo6ixzgMy/Hqc+mWYv7maAG\n\
Jtevs3qgMZ8F9Qzr3HpUc6R3ZYYWDY/xxPisuKftOPZgtH979XC4mdf0WPnOBLqL\n\
2DJ1zaBmjiGJolvb7XNVKcUfDXYw85ZTZQ5b9clI4e+6bmyWqQItlwt+Ati/uFEV\n\
XzCj70B4lALX6xau1kLEpV9O1GERizYRz5P9NJNA7KoO5AVMp9w0DQTkt+LbXnZE\n\
HHnWKy8xHQKZF9sR7YBPGLs/Ac6tviv5Ua15OgJ/8dLRZ/veyFfGo2yZsI+hKVU5\n\
nCCJHBcAyFnm1hdvdwEdH33jDBjNB6ciotJZrf/3VYaIWSalADosHAgMWfXuWP+h\n\
8XKXmzlxuHbTMQYtZPDqspS5aK+S4Q9wb8RRAYo=\n\
-----END CERTIFICATE-----
casdoor.organization-name=ShardingSphere
casdoor.application-name=ShardingSphere
```

4. **测试一下**

ShardingSphere
username
password
Login
Login with Casdoor
Copyright © The Apache Software Foundation, Licensed under the Apache License 2

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

2.2 创建一个应用

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。

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
```

结果

IoTDB WorkBench
IoTDB WorkBench
Welcome To IoTDB WorkBench
Account
Please Input Account
Password
Forget Password?
Sign In
Sign In With Casdoor

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）。

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

```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。

en_US
DolphinScheduler
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（Sigin → Security → Add OpenID Connect Provider）设置为FIREZONE_HOSTNAME。

2. Casdoor可以正常登录和使用。

3. `CASDOOR_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/`。

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。

3. 配置FireZone：Security → Add OpenID Connect Provider

- `Discovery Document URI`：FireZone Provider Discovery Document URI应该是`https://[CASDOOR_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

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://CASDOOR_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

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-Csrf=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=eyJhbGciOiJIUzI1NiIsImprdSI6Imh0dHBzOi8vbG9jYWxob3N0OjgwODAvdWFhL3Rva
```

Built-in Organization
localhost:8000/login
Casdoor
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://CASDOOR_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。

ThingsBoard
Home
Home
Tenants
Tenant profiles
Resources
Notification center
Settings
Security
General
Two-factor authentication
OAuth2
Tenants
2
Tenant profiles
2
CPU
15% | 8 cores
Devices
9
Assets
2
Users
8
Customers
3
Realtime - last h
100 %
75 %
50 %
25 %
0 %
13:40
Documentation
Getting started
Tenant profiles
API
Widgets Library
Transport messages
History - last 30
Configured features
Email
SMS
Slack
OAuth 2
2FA
0k
May 02
May 05

按照以下方式填写空白:

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：

ThingsBoard | Login
localhost:8080/login
ThingsBoard
Login with Casdoor
OR
Username (email)*
Invalid email format.
Password
Forgot Password?
Login

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

上述示例值可以从Casdoor演示站点获取：app-vue-python-example

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：
>
> `CASDOOR_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 `CASDOOR_HOSTNAME`。

```
conf > app.conf
dbName = casdoor
redisEndpoint =
defaultStorageProvider =
isCloudIntranet = false
authState = "casdoor"
httpProxy = "127.0.0.1:10808"
verificationCodeTimeout = 10
initScore = 2000
logPostOnly = true
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`。

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URl。 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-oGk3T5tIiMFo3SazCO75f0HEiE7Q",
        "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:

`CASDOOR_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。

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://`CASDOOR_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://CASDOOR_HOSTNAME/.well-known/openid-configuration",
      "introspection_endpoint_auth_method": "client_secret_basic",
      "logout_path": "/logout",
      "realm": "master",
      "redirect_uri": "http://APISIX_HOSTNAME/REDIRECTWHATYOUWANT",
      "bearer_only": false,
```

现在，访问`http://APISIX_HOSTNAME/get`，浏览器将会重定向你到Casdoor登录页面。 成功登录后，你会看到已经向httpbin.org发送了一个请求，如下面的截图所示。


```
{
  "args": {},
  "headers": {
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9",
    "Accept-Encoding": "gzip, deflate",
    "Accept-Language": "zh-CN,zh;q=0.9",
    "Cookie": "casdoor_session_id=db6eed5e89bb70c19046856b5b719803; session=MUSCZ-upGgLbbDJbHEL2Iw..|1639973945|kTiWhzu8OmE3QBYbokh3WmPN6uJ1xLeXe6dnl5QbpjWgvMq_veKDd0FY-
EWV0d1FQwDWa6uweDC4bh7M1mrKgABZ1lP6OdZ9HJebMhhmLW3dptnO38F3GhG9hDi7KoSD10CO4a4y7Koq1jNCodLL5zIJZ1K0usRQ3biWo54d8fXbZ_dmF05LrFFsxgYgp2-NegmaAHnE2lHSkrD4wuX0Z6Q-ckfhIvEbU0C
4rxyugiSDthWSP1jfpLuWEbO19gHNbKJPyKJ4YZ5EYvUIbepqrqyyTug5zF6WyIUTqY7TQ7aDQd8iUvetzcrswuj_ESNKTQtDVitaL9GkVqftOiswm2b0ryW9P_lB76ZIOtFtnnUR4_a6TzQjHaKYG9vtZRGSDIYlbTdHuO_F
y0Jlayp7sJltZO9bqurh2YXQGXoXWMxOPdZ44AQ15detkUycFeGAxlIaFgfBQHQiyeCsE4XOJuxHv3gTKu9lYncwuhHJvzohI6KeZkK5P5xJhoZCfd3_yZb8a_ewU0dWXNruPqrsfd0yhYEPVHB-
3XEHdllJbW27tZEDXDzBEVmEAokS5QPDKZkIPwRtqqb6G1OKq4tXQWeNHPOrKaiItnBlEgD1rJbMKpV74nhYeihfsQnsamc_ZS6AaEFOMwAu1fROslAsfCUj6kX47Q1_HV2VoKVfa2WAjPNhzrUAmrrzpwZBmn_lsFSa664f8
9d8U3jmU8wGhL_bRt8-AdRQOGYQht8ghJcat1OBIn6EUPB7G1CVsIaYYIBA2ylRBx6NV-J3NMRUsj5YkMbUqPFOLuR8JkhyYPs2spKMgcJGpAX6oaxVmtwc3U8yx5e4HHM3O2Q8lMP2vW4lxffA81rqXWI4nOqC1B-fQRJSEv
iCi6FOiFKvYMog7cfd4_LK9SIK6RTasyoJpPSEG26onqWSx8-jsRwNRykSb7eoYFThtALOOXmkifgiJT53Id4JHP8O3RXsISguKPWtMjR6T1whIWQOOjr2wn182Xxqh8OprnlL8VLbLv_4eOp6uAbv4RnXfdE2P8Us6UOnNWz
oB6hrD99aM6NK_mxUZdoDGDlaRoeztXgNhvlyMO6SpKaer0xkbw7Gf3hNHq8Oh8xdkhx5QirUK24i2SoD9hqlfl0pTG3ZjJffNY958EJMLxmfbsAcYN3C8or2WrMoVndKWe-LsAlrFydrkHd5cT7AUqc_znij8CUgsYj1C5d-
V1H8OgnbAu1Qvg6gym7qzBe6CcYn7B69CYT8wdrtwyHnGS7rurQy3cu9blq8J1WN9Nwf2QgaovnooLcHEUPVOIMBkqtMlY8OCK7Ht-drf3RB6fNgOWaswbtPA4oUAXOQ9VHALhDzb9I7vBWBO3UTqDBk37OyvJWeOq6YfiY7z
jz4rjx8T8nM1OWEUUzQLNIVCH4bDt1GFX4GalA8hZT5W4WMOdcHL59XE4fXcHhtG18QGxCVrJ71Zz0sxTlugc1UEjzXGGsGX7Dy0DkRMt3n09AJwtKd8rqUATrc21ohgGJjLNWj4K_BOLMfW-ifJQKNxI6NcrbLGU-
P_nIvIiPLRNOdmCjySpShqdNPQCi3B9YiaBLLBO4jBjzLNXkmkswf5LGFQEt16ZsTkcAbWuc1DgqjgFFESRdTS42G1wsUW97xOQ4QATA1WHR4Pyp3oR3tmpOG9GAcjnghc2POatcSOW5Ab70TVTKOVNYbLHnn6NT4tqgyeUDw
Isq4inM4UsmUG4AVzQvaXtPtYrEc03cOsfvFpoTOXH5ird92tLTZ36944oPglOuOr6OWxGbbs3Fi7Pl_BLuo6WkkI5U_FgUrPjEDqrMs_BjhxpDcb4tKavfopUlc4_PhnOyk1pwCNki_UpFX-MyR6Sro-LyAOHN9O7eNXAPig
wxLHbQxklzT9qN_cplDWmRFM9bEGuj4Vyn-1ZJNbSc49tGpLciaY-FfUXPn76At_mAfkJB1DAPT3ruhrs97T-W6k0mMxnWRz7BH4dQeOzYuDPcdqtqPcWZqiOeYjbhF1Jw9sXvj_wOkXBMnI7RNhKfU8a_8ZtBWpqvU9yYqP3
VEzho0kykCHcc_rmcfirndKnTcBDnMUbvcsikMFJpQFccEwZbB3Y73neTEYchs-XpOffSIhAq9BKtK-qWQCou2tYmCIMwvjx3XOXyB-CDyKpEX3oaxYuUzjgK2IR2FXRYhvFA1y4PAii3nvLNqT18f3oLrAZgago6To0i5Vv2
ka1DY5U2MN8NnKRTbaF6yOHgh3HWjdnI1J9GfhhaX9CFF2WOh0KZOaLSHfyPHW5nI3AvcXyPleaITj_clOK4qmo4TDPrdCiG3qjeOLS71gP6itjJBhdhjQe-PaQ855mcU4i3N5EuaQZSorJ3q2lPiXENVqpaKBBlsZj5ansC-
sEhOr9q8s2iZfDk99DJGAzZU6GGCPGfJTOjFbbn2kI1Lox_sDg_1KUXMJv3h2pvQYIzS3nJFVvqj_-SGexIAM9ROzUuQU3F9IF1b_1aenlDOeHEnrQPxxsJMaIYV1_ACPLP_xvK4rMX7pY_pkl-oSQVjYu7v_LATxjQOW31gD
Axgrpiqn4b7AFIvXb4C9n7gQSvJqZwtnaU39VxBkGbWItP8HW4jxhaCS0GuYJTfUSc7AL004i-1NOEATyMv-S9QOeYXwF5aNa1TVRyXtILFk5Gx8TQjwxMXiPeHPaz_NWpxra7GK1mag2GnAhPsPXAiG6GsKa5zZR5tAVc_MF
session_2=ItpAMrr7pWWwIAuQLPdlbyNDNP5NhUVIDVopWYqkFWv3VPMz3VFOKj01M64Vgd08SePmNV184wKtxmpZyrUXy1DwmGO6UPv5Y9WwcKs8JvtQCVoseIdmSdDNMEfb1Jd62Q4vgqtkEJ_xxYW8mySYsMo2Gno_9eC
QptDzhlKzNCIMxHPVxpJFTZqbRyBkned6sKVRhNOdfOO_bnmZla7aQYHhvaLfUXB9biZXbNkLbLr45J51RmprtTSOx8stKs1ZIWpMOkGvARYoiZQPahOs874Qv7V3F7pvxBwAJIr-Ov4LxIdnbc8ISi4qsvP4-dUhp46R76Ev
QNBAyoaRxHivZMASXvbpAjuLx3eUSZcOWBu81E5mq1H_CmlzhgDjBO8KPR6lL7F1fUs5L3j4ZqF2JYjNBf72xrcZoa4eyJWmm2W2pNdJsLHCJYoHGj_7pkWZ19hKzfD-DK6qwLf6uA6VA9vV_KFvPz0s5vYD2UUutwx58uQU4
o4LY7CzU_60MVMOFA8VgSEJgbIhrhAJk7gcQiD5YvdGpjFi9mooYsEqDVe9I4dRyGO1K6cOPlGlv9Nf90bcm5Tsi8mrjr-Bcfb_THBTgapPCHhTtMtlgIbXvLkT-mnRC5chhPblumxzsBLkI8563t2CRCsjVFB_EvAt6n3CNs
mtv0hc2hJ50Mr8azIanuCisjETqSTgKP9BqiQkx1ejKCpPm2Sjgj8MNiMbEGiLwi8NJG81jkPzNuGJONZwFZv3xay9i5ZUkARIS7iaz8seTr3alNWpKaS8ccnHyAGrm-bMdc_tcQO8AgOsA-
ePBfVK4EoBCG4vrvTM_bBgdnoxmV8iBbb5eq4mZ1F8f74usI7XX4elkyto9QAJl8AwOYhnce0DZtQwxDpd7Me2WFZlmdi8ly_FjxXkyTrrSER7vQFBeDFNtbSfuiM6kiRYP53QSblJdiBaecPQJD3cjeCmAHp8cnP-VwZOvIF
ppf31IzSQOfU48scJWongdmaJ_ZjsUb_PUsxFU7TDxbIi2bCd3C0gZR5JiaLESXyIT|ZBOG5HiNCt39_pCbipXoxnlwxy8.",
    "Host": "1[redacted]",
    "Referer": "http://[redacted]0/",
    "Upgrade-Insecure-Requests": 1,
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Safari/537.36",
    "X-Access-Token":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiYWRtaW4iLCJvd251ciI6ImJlaWx0LW1uIiwibm9uY2UiOiJmM2U5MzU2YTFmN2MwMjVmYzNhYmE0OTc4OWI0YjNiOSIsImlzcyI6Imh0dHA6Ly8xNjguMTM
V4cCI6MTY0MDU3NTE0NSwibmJmIjoxNjM5OTcwMzQ1LCJpYXQiOjE2Mzk5NzAzNDV9.BEIAb5s8ywTRYAVDKgLn4thIQpDqlPBD83avwNcE214J-cjirIdxYeagOu6869HK_xUCmdHvfhEeZalhmfWbVwK3YUfMNTIodq7sr8
nvgGcJx393DSPnjJ5YcTvh2fdjB1c7YWGFVWFOypSwd61nOXxRlAfMJqKfKNfyEZMFzni_9VyStxY2C5oJk_0etky9vfuA2rTvNXAfMtAd1bQ_TvRHUD7OITkyBRKCddRzPg_I1K95zumwHqU_okjHvOBAs2SkTOqOwv7S_Xf
MhiHucQFlGR0aE2jIU2GLTJMMnOyHYfNKUshCuWn6BmiD6DeDqPSC-VUPcWRgZsS8pAOacTBbzatJmM7T6gAyA5i5MjhGmslpuRJc7j0539HbMMrYeiTv-RwOAqv_kE1x_RUKmvRoUhKWwQfzzaFnzExMSzlrR91Kn9txteE3
    "X-Amzn-Trace-Id": "Root=1-61bff62a-597370f76f7f7eda061ac1db",
    "X-Forwarded-Host": "[redacted]",
    "X-Userinfo":
"eyJtc2ciOiIiLCJzdWIiOiIOODNiZTBlYiO2NjJiLTQ5YzMtODczZS03MTU0NjY0ZmQzNGEiLCJkYXRhIjp7ImxhcmsiOiIiLCJnaXRsYWIiOiIiLCJhcHBsZSI6IiIsImF6dXJlYWQiOiIiLCJzbGFjayI6IiIsInBhc3N3
sInRhZyI6InNOYWZmIiwicGFzc3dvcmQiOiIxMjMiLCJhZGRyZXNzIjp7fSwicHJvcGVydGllcyI6e3OsImxkYXAiOiIiLCJkaXNwbGF5TmFtZSI6IkFkbWluIiwiYXZhdGFyIjoiaHROcHM6XC9cL2Nhc2Jpbi5vcmdcL2lt
lciI6ImJlaWx0LWluIiwicmVnaW9uIjoiIiwibGFuZ3VhZ2Ui0iIiLCJzY29yZSI6MjAwMCwiY3J1YXR1ZFRpbWUiOiIyMDIxLTEyLTE0VDA1OjQ1OjE3WiIsImlzT25saW51IjpmYWxzZSwidXBkYXR1ZFRpbWUiOiIiLCJp
hbC11c2VyIiwic2lnbnVwQXBwbGljYXRpb24iOiIiLCJoYXNoIjoiMjlmMDA5MzAzODNiOWMyM2U4MzAzNmI4OTg3YTlkNzAiLCJpZENhcmRUeXBlIjoiIiwiaWRDYXJkIjoiIiwiZW1haWwiOiJhZG1pbkBleGFtcGxlLmNv
iOiIiLCJwcmVlYXNoIjoiMjlmMDA5MzAzODNiOWMyM2U4MzAzNmI4OTg3YTlkNzAiLCJjcmVhdGVkSXAiOiIiLCJsYXN0U2lnbmluVGltZSI6IiIsImxhc3RTaWduaW5JcCI6IiIsImdpdGh1YiI6IiIsImdvb2dsZSI6IiIs
iIn0sImRhdGEyIjp7InBob251UHJlZml4IjoiODYiLCJwYXNzd29yZFNhbHQiOiIiLCJkaXNwbGF5TmFtZSI6IkJlaWx0LWluIE9yZ2FuaXphdGlvbiIsImVuYWJsZVNvZnREZWxldGlvbiI6ZmFsc2UsImRlZmF1bHRBdmFC
odHRwczpcL1wvZXhhbXBsZS5jb20iLCJtYXN0ZXJQYXNzd29yZCI6IiIsImZhdml jb24iOiJodHRwczpcL1wvY2RuLmNhc2Jpbi5jb21cL3NOYXRpY1wvZmF2aWNvbi5pY28iLCJjcmVhdGVkVGltZSI6IjIwMjEtMTItMTRU
  },
  "origin": "119.[redacted].8",
  "url": "http:/[redacted]et"
}
```

PHP

Zabbix

Using Casdoor for authentication in Zabbix

WordPress

Integrating CAS (Central Authentication Service) into WordPress using Casdoor (identity provider) and the wp-cassify plugin.

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

Zabbix

Zabbix as a Service Provider (SP), while CASdoor acts as an Identity Provider (IdP). They communicate and collaborate through the SAML2 protocol to achieve single sign - on (SSO) functionality for users in Zabbix.

Step 1: Deploy Casdoor and Zabbix

Firstly, deploy Casdoor and Zabbix. After a successful deployment, make sure:

1. Casdoor can be logged in and used successfully.
2. You can successfully log in and use Zabbix.

Step 2: Adding Certificates

To ensure the security of communication, certificates need to be configured between Zabbix and CASdoor. **Private keys and certificates** should be stored in the `/etc/zabbix/conf/certs/` directory, unless a custom path is provided in `zabbix.conf.php`.

By default, the `zabbix - web - nginx - mysql` Docker container looks for the following locations:

- `/etc/zabbix/conf/certs/sp.key` - SP private key file
- `/etc/zabbix/conf/certs/sp.crt` - SP certificate file
- `/etc/zabbix/conf/certs/idp.crt` - IDP certificate file

Creating Certificates in CASdoor: Log in to the CASdoor management interface and follow the system prompts to create two certificates. These two certificates will be used for communication encryption between Zabbix and CASdoor.

Name	Organization	Created time	Display name	Scope	Type	Crypto algorithm	Bit size	Expire
zabbix_idp	admin (Shared)	2025-06-19 18:50:06	zabbix_Idp	JWT	x509	RS256	4096	20
zabbix_sp	admin (Shared)	2025-06-19 18:48:52	zabbix_sp	JWT	x509	RS256	4096	20
cert-built-in	admin (Shared)	2025-06-14 17:05:33	Built-in Cert	JWT	x509	RS256	4096	20

Copying Certificates and Private Keys: Copy the created certificate and private key files to the Zabbix configuration directory `/etc/zabbix/conf/certs`. If you are using Docker for deployment, you can map local certificate files to the container using volume mounting.

Step 3:Configuring Zabbix

For SAML configuration in Zabbix, three required fields need to be set: `Single Sign - On`, `Issuer`, and `Public Certificate`.

Log in to the Zabbix management interface, click `User` → `authentication` → `SAML settings`.

Configure Zabbix according to the SAML metadata in the CASdoor configuration:

- idP entity ID (Issuer): Corresponds to `entityID="http://localhost:8000"`. This value identifies the entity ID of CASdoor, and Zabbix will communicate with CASdoor based on this ID.
- SSO service URL: Corresponds to `Location="http://localhost:8000/api/saml/redirect/admin/zabbix"`. This is the URL of the single sign - on service provided by CASdoor. When a user initiates a single sign - on request in Zabbix, they will be redirected to this URL for authentication.
- username attribute: The SAML attribute used as the username when logging in to Zabbix. Here, `Name` is used, indicating that Zabbix will use the `Name` attribute in the SAML assertion as the user's login name.
- SP entity ID: A unique SP ID that can be set arbitrarily. This ID is used to identify the Zabbix service provider and needs to be consistent with the

configuration in CASdoor.

Step 4: Configuring CASdoor

Some necessary configurations need to be made in CASdoor to ensure the normal operation of the integration with Zabbix.

Editing Name and Logo: Log in to the CASdoor management interface, find the relevant settings, and edit the application's name and logo for better presentation to users.

Selecting a Certificate: In CASdoor, select `zabbix_idp` as the certificate for signing and encrypting SAML messages to ensure communication security.

Redirect URL: Enter a unique name. In your SP (Zabbix), this may be called `Audience` or `Entity ID`. Ensure that the `Redirect URL` you enter here is consistent with that in your SP; otherwise, single sign - on may fail.

Reply URL: Enter the URL of the ACS (Assertion Consumer Service) for validating SAML responses. This URL is the address where Zabbix receives SAML assertions sent by CASdoor.

Step 5: Creating a Zabbix User

Create a test user in Zabbix to verify the single sign - on functionality.

1. Log in to the Zabbix management interface and find the user management module.
2. Create a user with the username "test" (you can customize the username; this

is just an example).

Username	Name	Last name	User role	Groups	Is online?	Login	Frontend access	API access	Debug mode	Status	P
Admin	Zabbix	Administrator	Super admin role	Internal, Zabbix administrators	Yes (2025-06-20 03:21:35 PM)	Ok	Internal	Enabled	Disabled	Enabled	
guest			Guest role	Disabled, Guests, Internal	No	Ok	Internal	Disabled	Disabled	Disabled	
test	test		Guest role	Guests	No	Ok	System default	Disabled	Disabled	Enabled	

Display

Step 6: Creating a CASdoor User

Add a user in CASdoor with the same username as the one set in Zabbix.

1. Log in to the CASdoor management interface and find the user management module.
2. Add a new user with the same username as the one created in Zabbix.
3. Select Zabbix and enter the user's email address.

Users Add Upload (.xlsx)

Organization	Application	Name	Created time	Display name	Avatar	Email	Phone	Affiliation	Action
built-in	zabbix	test	2025-06-18 17:06:54	test		jlapmu@example.com	17636092052	Example Inc.	Edit Delete
built-in	app-built-in	admin	2025-06-14 17:05:33	Admin		admin@example.com	12345678910	Example Inc.	Edit Delete

2 in total 1 10 / page

Step 7: Zabbix Login Process

After completing the above configurations and user creation, you can test the single sign - on functionality.

Open a browser and visit `localhost/index.php`.

Click `Sign in with Single Sign - On(SAML)`.

You will be redirected to the CASdoor page. On the CASdoor page, enter the corresponding username and password to log in.

If the login is successful, you will be redirected back to `https://localhost:8080/zabbix.index.php`, indicating that the single sign - on functionality is working properly.

Through the above steps, you can successfully complete the integration of Zabbix and CASdoor and achieve single sign - on functionality for users. If you encounter any problems during the configuration process, please refer to relevant documentation or community forums for help.

WordPress

This guide walks you through configuring **CAS-based single sign-on (SSO)** for WordPress, using:

- wp-cassify: A WordPress plugin that enables CAS authentication, allowing users to log into WordPress via a CAS server (e.g., Casdoor).

Step 1: Deploy Casdoor and WordPress

First, deploy Casdoor (as the CAS server) and WordPress.

1.1 Deploy Casdoor

Refer to the official Casdoor installation guide: Casdoor Server Installation.

After deployment:

1. Access the Casdoor web UI via its public/private URL (e.g., `http://<casdoor-server-ip>:8000`).
2. Log in with the default admin credentials (or your custom credentials set during deployment).
3. Confirm you can navigate the dashboard (e.g., access "Applications", "Users")—this ensures Casdoor is ready to act as a CAS IdP.

1.2 Deploy WordPress

Deploy WordPress using your preferred method:

- **Docker**: Use the official WordPress Docker image for quick setup (see WordPress Docker Docs).
- **Traditional Hosting**: Install WordPress on a LAMP/LEMP stack (follow WordPress Official Installation).

1. Access the WordPress site via its URL (e.g., `http://<wordpress-server-ip>`).
2. Complete the initial WordPress setup (create an admin account, site title).
3. Log in to the WordPress Admin Dashboard (`http://<wordpress-server-ip>/wp-admin`) to confirm functionality.

Step 2: Create a Casdoor Application

Casdoor requires an "Application" to act as a bridge between the CAS server and WordPress. Follow these steps to create one:

1. Log in to the Casdoor Admin Dashboard.
2. Navigate to **Applications** > **Add Application**
3. Click **Save** to create the application.

Step 3: Configure the wp-cassify Plugin

The wp-cassify plugin connects WordPress to your Casdoor CAS server. Follow these steps to install and configure it:

3.1 Install wp-cassify

1. Log in to the WordPress Admin Dashboard (`http://<wordpress-server-ip>/wp-admin`).
2. Navigate to **Plugins** > **Add New**.
3. In the search bar, type "wp-cassify" and select the plugin by "wp-cassify".
4. Click **Install Now**, then **Activate** to enable the plugin.

3.2 Configure wp-cassify Settings

1. In the WordPress Admin Dashboard, navigate to **Settings** > **WP-Cassify** (the plugin’s configuration page).

2. Configure User General Settings.

wp-cassify Setting	Value to Enter
`CAS Server Base URL`	Your Casdoor CAS endpoint (e.g., `http://<casdoor-server-ip>:7001/cas/<organization name>/<application name>`).
`Create user if not exist`	Enable

3. Configure User Attribute Synchronization.

WordPress User Field	Casdoor Attribute Name	Example Value
`user_email`	`email`	user@example.org
`user_nickname`	`displayName`	John Doe
`display_name`	`displayName`	John Doe

Step 4: Test CAS Authentication

Verify the integration works by logging into WordPress via Casdoor:

1. Log out of the WordPress Admin Dashboard (if logged in).
2. Access the WordPress login page (`http://<wordpress-server-ip>/wp-admin/index.php`).
3. You'll be redirected to the Casdoor login page. Enter valid Casdoor user credentials.
4. After successful authentication, Casdoor will redirect you back to WordPress—you should now be logged in (as the synced user).

127.0.0.1:8167/wp-admin/profile.php
attack825 home
Howdy, alice
Dashboard
Profile
Collapse menu
Toolbar
Show Toolbar when viewing site
Language
Site Default
Name
Username
alice
Usernames cannot be changed.
First Name
Last Name
Nickname (required)
alice
Display name publicly as
alice
Contact Info
Email (required)
8op3zg@example.com
If you change this, an email will be sent at your new address to confirm it. The new address will not become active until confirmed.
Website
About Yourself
Biographical Info

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。

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");
```

> **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。

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。

`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`。

步骤4：尝试使用ShowDoc

你应该在登录页面上看到以下内容：

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：

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：

3. 在Casdoor应用程序页面中找到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
- `UserInfo server URL`: http://**`CASDOOR_HOSTNAME`**/api/get-account
- `Scopes`: `address phone openid profile offline_access email`

退出Flarum并测试SSO。

Log In
Cardoor SSO
Username or Email
Password
Remember Me
Log In
Forgot password?
Don't have an account? Sign Up

Moodle

Casdoor可以用来使用OAuth连接Moodle。

以下是一些配置设置：

- `CASDOOR_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`CASDOOR_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

   -

2. 配置此应用程序

   -

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"

-

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
H5P
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

-

5. 启用OAuth2插件

-

6. 如果你想阻止编辑Casdoor的电子邮件

-

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
Home
Log in
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
dbName = casdoor
redisEndpoint =
defaultStorageProvider =
isCloudIntranet = false
authState = "casdoor"
httpProxy = "127.0.0.1:10808"
verificationCodeTimeout = 10
initScore = 2000
logPostOnly = true
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。

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。

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。 For more information, read about configuring permission rules in Hasura.

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.yaml，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。

找到你之前输入的用户名，然后点击“编辑”

复制访问令牌

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。

HASURA v2.22.0 | API | DATA | ACTIONS | REMOTE SCHEMAS | EVENTS | SETTINGS | HELP

GraphiQL | REST | Allow List

GraphQL Endpoint

Request Headers

ENABLE	KEY	VALUE
☐	Hasura-Client-Name	casdoor
☑	content-type	application/json
☐	x-hasura-admin-secret	••••••••••••••••
☑	Authorization	Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0eXAiOiJKV1QifQ.eyJvd2SIcil6Imhhc3VyYSIsIm5hbWL
	Enter Key	Enter Value

Explorer

query MyQuery

- users
  - distinct_on:
  - limit:
  - offset:
  - order_by:
  - where:
  - id
  - username
- users_by_pk

GraphiQL | Prettify | History | Explorer | Code Exporter | REST | Derive action | Analyze | Docs

```
query MyQuery {
  users {
    id
    username
  }
}
```

QUERY VARIABLES

```
{
  "data": {
    "users": [
      {
        "id": "4ec7ccee-ec7b-4191-a78d-e11f50686f8b",
        "username": "test"
      }
    ]
  }
}
```

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：

`CASDOOR_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CASDOOR_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：`CASDOOR_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：

○

2. 配置此应用：

○

- `/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

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的运行时信息，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的运行时指标，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。

```yaml
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。