概述

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

You need to enable JavaScript to run this app.

Casdoor 可为网页UI和应用程序用户的登录请求提供服务。

Casdoor 的特性：

1. 前后端分离的架构，使用 Go 语言开发，Casdoor 支持高并发，提供基于Web的用户管理 UI，并支持中、英等多种语言。

2. Casdoor支持第三方应用登录，如GitHub、谷歌、QQ、微信等，并支持通过插件扩展第三方登录。

3. 使用 Casbin 基于授权管理，Casdoor 支持 ACL，RBAC，ABAC，RESTful 等访问控制模型。

4. Casdoor可通过手机验证码、邮箱验证码登录，还有找回密码功能。

5. 审查和记录访问日志。

6. 使用阿里云、腾讯云、七牛云提供的图片CDN云存储。

7. 个性化的注册、登录和找回密码页面。

8. Casdoor 支持通过数据库同步的方法来与现有系统集成，用户可以顺利过渡到 Casdoor。

9. Casdoor 支持主流数据库: MySQL、PostgreSQL、SQL Server 等, 并支持扩展插件的新数据库。

如何操作?

步骤0 （前置知识）

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

步骤 1 (授权请求)

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

> **提示**
>
> 对于 `xxx` 的部分需要写上什么?
>
> - 对于 `client_id`: 您可以在每个应用程序里找到它
> - 对于 `redirect_uri`: 您应该将此设置为你自己的应用程序回调URL, 通过这个信息, Casdoor 可以知道在授权后向哪里发送信息
> - 对于 `state`: 您应该用您的应用程序名称填写这个内容

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

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

步骤 2（授权认证）

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

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

这两个关键词发到您的应用后，应用便得到授权，至此就完成了 `授权` 环节。

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

步骤 3（授权认证）

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

步骤 4（访问令牌）

这一步骤中，Casdoor 会通知应用程序： *"这个 `code` 应该是合法的，你一定就是那个正确的应用程序。 这是 `access_token`。*
有了 `code`，Casdoor 知道它是一个已授权的应用（第二步中用户给予的授权），试图获取 `access_token`（稍后将会用来获取更有用的信息）。

步骤 5（访问令牌）

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

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

步骤 6 (受保护资源)

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

这就是Casdoor 如何与您的应用程序一起工作。

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

在线演示

Casdoor

这里是一个由Casbin部署的在线演示。

- Casdoor官方演示

全局管理员登录:

- 用户名： `admin`
- 密码： `123`

Casbin-OA

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

- Casbin-OA
- 源代码： https://github.com/casbin/casbinoa

Casnode

Casnode 是Casbin社区开发的官方论坛。

它使用 Casdoor 作为认证平台并管理成员。

- Casnode
- 源代码： https://github.com/casbin/casnode

结构

Casdoor包含两个部分：

名称	描述	语言	源代码
前端	Casdoor的前端Web界面	JavaScript + React	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，您至少应该熟悉4个核心概念：`组织(Organization)`，`用户(User)`，`应用(Application)` 和 `提供商(Provider)`。

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

组织

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

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

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

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

用户

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

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

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

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

用户的类别定义如下：

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

应用程序

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

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

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

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

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

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

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

1. 手动连接字符串：

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

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

`getSignupUrl()` and `getSigninUrl()`: casdoor-js-sdk

3. 使用后端 SDK (使用Go Java 等后端代码)：

`GetSignupUrl()` and `GetSigninUrl()`: casdoor-go-sdk

提供商

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

Currently, All providers supported by Casdoor can be found here: provider/overview

```go
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?

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

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

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

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

服务器安装

安装要求

操作系统

支持所有主流的操作系统，包括Windows、Linux和macOS。

环境

- Go 1.17+
- Node.js LTS (16或14)
- Yarn 1.x

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

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

数据库

Casdoor使用 XORM 与数据库进行交互。 基于 Xorm Drivers Support, 当前支持的数据库包括:

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

下载

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

名称	描述	语言	源代码
前端	Casdoor的网页前端界面	JavaScript + React	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，您直接通过git克隆仓库就可以了：

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

配置

配置数据库

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

MySQL:

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

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

PostgreSQL:

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

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

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

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

CockroachDB:

You can also use cockroachdb with postgres driver. It has same configuration as postgreSQL.

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

> **信息**
>
> For CockroachDB, don't forget to add `serial_normalization=virtual_sequence` to the `dataSourceName` like the above example. otherwise you will get error regarding existed database, whenever the service started or restarted. Notice, this must be added before the database created.

Sqlite3

You should specify `app.conf` like this:

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

通过 Ini 文件配置

Casdoor can be configured via a single file: conf/app.conf, the content of which by default is:

```
appname = casdoor
httpport = 8000
runmode = dev
SessionOn = true
copyrequestbody = true
driverName = mysql
dataSourceName = root:123456@tcp(localhost:3306)/
dbName = casdoor
tableNamePrefix =
showSql = false
redisEndpoint =
defaultStorageProvider =
isCloudIntranet = false
authState = "casdoor"
socks5Proxy = "127.0.0.1:10808"
verificationCodeTimeout = 10
initScore = 2000
logPostOnly = true
origin = "https://door.casdoor.com"
staticBaseUrl = "https://cdn.casbin.org"
```

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

证码。

Despite all the configurable fields, as a beginner, you only need to modify two items: `driverName` and `dataSourceName` based on your database. This database will be used by Casdoor to store all data, including users, organizations, applications and so on.

- `tableNamePrefix` 是使用适配器时表格的前缀。
- `showSql` : 如果日志级别大于INFO 则显示 SQL 语句或不在 Logger 上。
- `redisEndpoint` 后填写Redis地址，被Beego用于session存储 如果此参数为空，session数据将被储存在 `./tmp` 文件夹中 使用Redis作为Beego的session存储，此参数的示例为： `redis.example.com:6379`。 如果Redis启用了密码，请填写 `redis.example.com:6379,db,password`。 详情请参阅 https://beego.vip/docs/module/session.md#saving-provider-config
- `defaultStorageProvider` 是默认的文件存储服务名称。 如果您需要使用文件存储服务，例如 `头像上传`， 您需要设置存储提供商，并在您的 `应用` 中应用它。 详情请参阅 **存储**
- `isCloudIntranet` 用于确定您的提供者端是否是内联网端点。
- `authstate` 是授权应用程序名称。 登录时将检查该参数。
- `socks5Proxy` 是 SOCKS 代理服务器 IP 地址。 **设置代理端口**， 因为我们有与谷歌相关的服务或使用 `Google` `GitHub` `Facebook` `LinkedIn` `Steam` 作为OAuth Provider ， 在某些领域将受到网络限制。
- `initscore` 是每个用户的最初分数。 每个用户都有一个得分属性。 得分被 Casnode 所使用。 分数无法控制 Casdoor 中的任何东西。
- `logPostOnly` 用于识别是否只使用帖子方法添加记录。
- `origin` 是origin形式的后端域名
- `staticBaseUrl` 是系统初始化数据库时静态图像的地址。

通过环境变量配置

All configuration items defined by Casdoor in the ini file mentioned above can be chosen to configuration via environmental variables, so can **some of the beego configurations** items(httpport,appname).

For example, when you try to start casdoor, you can use something like this to pass the configuration via environmental variables.

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

Besides, `export` derivatives are also a possible method. The names of environmental variables should be exactly the same with the names you want to use in the ini file.

*Note: configurations in environmental variables can **override** the configurations in ini file.*

运行

There are currently two methods to start, you can choose one according to your own situation.

开发模式

Backend

Casdoor's Go backend runs at port 8000 by default. You can start the Go backend with the following command:

```
go run main.go
```

After the server is successfully running, we can start the frontend part.

Frontend

Casdoor's frontend is a very classic Create-React-App (CRA) project. It runs at port `7001` by default. Use the following commands to run the frontend:

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

Visit: http://localhost:7001 in your browser. Log into Casdoor dashboard with the default global admin account: `built-in/admin`

```
admin
123
```

生产模式

Backend

Build Casdoor Go backend code into executable and start it.

For Linux:

```
go build
./casdoor
```

For Windows:

```
go build
casdoor.exe
```

Frontend

Build Casdoor frontend code into static resources (.html, .js, .css files):

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

Visit: http://localhost:8000 in your browser. Log into Casdoor dashboard with the default global admin account: `built-in/admin`

```
admin
123
```

> **提示**
>
> To use another port, please edit `conf/app.conf` and modify `httpport`, then restart the Go backend.

> **CASDOOR PORT DETAILS**
>
> In **dev** environment, the frontend is run by `yarn run` in port 7001, so if you want to go to Casdoor login page, you need set Casdoor link as **http://localhost:7001**.
>
> In **prod** environment, the frontend files is first built by `yarn build` and served in port 8000, so if you want to go to Casdoor login page, you need

to set Casdoor link as **http://SERVER_IP:8000** (If you are using reverse proxy, you need to set the link as **your domain**).

Take our official forum Casnode as an example:

Casnode uses Casdoor to handle authentication.

When we are testing Casnode in **dev** environment, we set the `serverUrl` as http://localhost:7001, so when we test signin and signup functionality using Casdoor, it will go to localhost 7001 which is the Casdoor port.

And when we put Casnode to **prod** environment, we set the `serverUrl` as https://door.casdoor.com, so users can signin or signup using Casdoor.

```
import * as ConfBackend from "./backend/ConfBackend.js"

export const OutbConfig = {
  // serverUrl: "https://door.casbin.com",
  serverUrl: "http://localhost:7001",
  clientId: "014ae4bd048734ca2dea",
```

(可选) 使用 Docker 运行

安装要求

硬件

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

如果您只需要运行预编译的镜像，请确保您的机器至少有100MB 的内存。

操作系统

支持所有操作系统 (Linux, Windows 和 macOS)

Docker

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

- Docker

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

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

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

获取镜像

我们提供了两个DockerHub 镜像：

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

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

选项 1: 使用示例数据库

使用端口 `8000` 运行容器。 如果本地host中不存在，它将自动下载镜像。

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

> ⚠ **注意事项**

> 中国等地区的一些用户通常使用 Docker 镜像服务，例如 Alibaba Cloud Image Booster (English) 实现比DockerHub 更高的下载速度。 然而，有一个问题，那些服务提供的 `latest` 标签不是最新的。 它可能会通过获取 `latest` 标签来产生一个非常旧的镜像。 为了缓解这个问题，您可以使用以下命令明确指定镜像版本号：
>
> ```
> docker pull casbin/casdoor-all-in-one:$(curl -sS "https://hub.docker.com/v2/repositories/casbin/casdoor-all-in-one/tags/?page_size=1&page=2" | sed 's/,/,\n/g' | grep '"name"' |awk -F '"' '{print $4}')
> ```
>
> 注意：上面的命令使用的 Linux 工具，例如 `curl`, `ed`, `grep`, `awk`. 如果您正在使用 Windows，请确保您在 Linux 样式外壳中运行，如 `Git Shell` 或 `Cygwin`。 `CMD` 或 `PowerShell` 将无法工作。

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

```
admin
123
```

选项 2: 使用 docker-compose

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

> 版本号:
>
> ```
> docker pull casbin/casdoor:$(curl -sS "https://hub.docker.com/v2/repositories/casbin/casdoor/tags/?page_size=1&page=2" | sed 's/,/,\n/g' | grep '"name"' |awk -F '"' '{print $4}')
> ```
>
> 注意：上面的命令使用的 Linux 工具，例如 `curl`, `ed`, `grep`, `awk`. 如果您正在使用 Windows，请确保您在 Linux 样式外壳中运行，如 `Git Shell` 或 `Cygwin`。 `CMD` 或 `PowerShell` 将无法工作。

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

通过docker-compose 创建一个独立的数据库:

```
docker-compose up
```

这样就完成了！ 🛫

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

```
admin
123
```

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

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

选项3 直接尝试使用标准镜像

> **注意事项**
>
> 中国等地区的一些用户通常使用 Docker 镜像服务，例如 Alibaba Cloud Image Booster (English) 实现比DockerHub 更高的下载速度。 然而，有一个问题，那些服务提供的 `latest` 标签不是最新的。 它可能会通过获取 `latest` 标签来产生一个非常旧的镜像。 为了缓解这个问题，您可以使用以下命令明确指定镜像版本号：
>
> ```
> docker pull casbin/casdoor:$(curl -sS "https://hub.docker.com/v2/repositories/casbin/casdoor/tags/?page_size=1&page=2" | sed 's/,/,\n/g' | grep '"name"' |awk -F '"' '{print $4}')
> ```
>
> 注意：上面的命令使用的 Linux 工具，例如 `curl`, `sed`, `grep`, `awk`. 如果您正在使用 Windows，请确保您在 Linux 样式外壳中运行，如 `Git Shell` 或 `Cygwin`。 `CMD` 或 `PowerShell` 将无法工作。

> **提示**
>
> 如果不方便挂载配置文件，使用环境变量也是一种可能的解决方法。
>
> example
>
> ```
> docker run \
>  -e driverName=mysql \
> ```

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

然后运行

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

将app.conf 挂载到 /conf/app.conf 并启动它。

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

```
admin
123
```

Casdoor公共API

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

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

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

教程

产品文档

产品	技术	文档
Dashboard of PingCAP TiDB	React + Typescript + Go + Gin	Use Casdoor for TiDB Dashboard SSO sign-in (other languages: Chinese, Japanese)
GitLab	Vue + Ruby + Rails	OpenID Connect OmniAuth provider
Apache Shenyu	Java	Casdoor Plugin (other languages: Chinese)
Alist	Typescript + SoildJS + 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	ASP.Net Core .net 6 Demo Authentication Project using local Casdoor Docker Container on Windows Subsystem for Linux
OAuth2 Proxy (Go)	中文	Use Casdoor + OAuth-Proxy to protect web applications on public networks
Casnode (Javascript + React + Go + Beego)	中文	Use Lighthouse to set up a forum like v2ex
Cloudreve (Go)	中文	Modify Cloudreve to support Casdoor
KodExplorer (PHP)	中文	Modify KodExplorer to support Casdoor

Deployment

数据初始化

如何从文件初始化Casdoor 数据

将静态文件托管到CDN

在CDN中托管前端静态文件

将静态文件托管在局域网内

如何部署Casdoor静态资源

数据库迁移

处理Casdoor中的 数据库迁移

数据初始化

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

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

使用方式

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

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

Docker

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

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

Kubernetes

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

```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。 您可以按照如下例子修改 `deployment` 的定义文件:

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

文件内容

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

以下是每个对象对应的Go结构体和文档链接:

对象	Go 结构体	文档
组织机构	stuct	doc
应用	stuct	doc
用户	stuct	doc
提供商	stuct	doc
证书	stuct	
ldaps	stuct	doc

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

将静态文件托管到CDN

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

> **备注**
>
> 我们假定您已经构建过了Cassdoor的前端代码。 如果还没有，请参照： 文档。

准备工作

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

> **注意事项**
>
> 当您填写 `域` 字段时，请以 '/' 结束
>
> Domain ⓘ : https://cdn.casbin.com/casdoor/

使用说明

脚本放在文件deployment/deploy_test.go中。

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

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

然后使用以下命令来运行脚本:

```
cd deployment
go test
```

如果执行成功，您将看到:

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

工作原理

脚本的功能:

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

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

将静态文件托管在局域网内

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

部署静态资源

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

克隆 仓库，在网页服务器上部署静态资源。 请确保您可以访问该资源。

配置Casdoor

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

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

数据库迁移

当数据库升级时，数据很可能崩溃，我们需要删除旧字段。 幸运的是，Casdoor 使用的 xorm 能帮助我们处理许多有关数据库迁移的问题。 但我们仍需自行处理一些模式和数据迁移，例如 **字段名称发生更改时**。

> **备注**
>
> 有关Schema 操作的信息，您可以参阅xorm docs

工作原理

如上所述，当字段名称发生更改时，xorm将无法进行任何操作， 但是它提供了一个 迁移 包来帮助我们解决这个问题。

你可以通过以下代码来处理字段重命名相关问题:

```
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{})
```

我们想要实现的目标是： **将 `p_type` 重命名为 `ptype`** 但既然xorm **不支持字段重命名**我们只能使用更复杂的方式：将 `p_type` 的值分配给 `ptype`然后删除 `p_type` 字段。

`ID` 字段特指我们进行的迁移。 `m.Migrate()` 运行后， `ID` 的值将被添加到数据库的迁移表。

当项目再次启动时 数据库将检查表中现有的 `ID` 字段，不会执行相同 `ID` 的操作。

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

Using Casdoor SDKs instead of standard OIDC protocol

如何启用单点登录

启用单点登录

📄 Vue SDK

Casdoor Vue SDK

🗃️ 桌面 SDK

3 个项目

📄 Casdoor插件

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

📄 OAuth 2.0

使用AccessToken验证客户端

📄 CAS

使用 Casdoor 作为 CAS 服务器

🗂️ SAML

3 个项目

📄 WebAuthn

在 Casdoor 中使用webauthn

概览

在本节中，我们将显示如何将您的应用程序连接到Casdoor。

作为服务提供商(SP)，Casdoor 支持两项认证协议：

- `OAuth 2.0 (OIDC)`
- `SAML`

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，允许应用程序例如Facebook，GitHub, 和Casdoor 访问HTTP服务上的用户帐户。 它的工作原理是将用户身份验证委托给主机服务，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了 授权流程。

Casdoor的授权程序基于OAuth 2.0协议 。 我们建议使用 OAuth 2.0 协议的原因在于：

1. 该协议简单易行，能解决多种问题。
2. 成熟度高且社区支持广泛

因此，您的应用程序将通过 OAuth 2.0 (OIDC) 与 Casdoor 连接。 具体而言，有三种方式连接Casdoor：

标准 OIDC 客户端

标准OIDC 客户端: 标准OIDC 客户端在各类编程语言或框架都广泛应用。

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 完全执行了OIDC协议。 如果您的应用程序已经运行了另一个 OAuth 2，那么 (OIDC) 身份提供商一般会通过标准的 OIDC 客户端库提供服务，如果您想迁移到 Casdoor，可使用OIDC discovery帮您轻松切换到Casdoor。

Casdoor SDKs

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

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 标准客户端库要高，但前者能提供最佳灵活性和最强API。

Casdoor插件

Casdoor 插件: 如果您的应用建立在一个流行的平台上(如Spring Boot, WordPress等), 并且Casdoor(或第三方) 已经为它提供了一个插件或中间件，那么就可以直接使用。 使用插件比手动使用 Casdoor SDK 更容易，因为前者是专门为平台制作的。

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。 该术语意味着您可以使用一组凭据登录多个不同的网站。 管理一个用户的单次登录要比管理分别登录到电子邮件、客户关系管理(CRM) 软件、Active Directory等简单得多。

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 是用户身份验证和使用服务授权之间的链接。

Casdoor可以使用 **SAML IdP**。 目前Casdoor支持SAML2.0 的主要功能。 More details

see SAML.

示例:

Casdoor 作为一个 SAML IdP in Keycloak

建议:

1. 该协议十分 **强大** 且适用于多种情景，可以说它是最全面的 SSO 协议之一。
2. 该协议 **涵盖范围过大**, 有很多可选参数, 因此，在实操中很难100%涵盖所有应用场景。
3. 如果应用程序是 **新开发的**，那么**不是很推荐** SAML，因为其技术过于复杂。

CAS

什么是CAS?

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

Casdoor已实现 CAS 1.0, 2.0, 3.0 功能。 详情见 CAS

建议:

1. 该协议本身较为轻量，易于执行，但它适用的场景较为单一。
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协议与其他协议相比没有优势。

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 中看到所有应用程序。

标准OIDC 客户端

OIDC discovery

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

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

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

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

OIDC 客户端库列表

这里我们列出了一些OIDC 客户端库，如Go 和 Java 等语言：

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

上表远远没有完成。 OIDC 客户端库的完整列表请查看更多详情：

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

Casdoor 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

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

后端 SDK	描述	SDK 码	示例
Go SDK	适用于Go后端	casdoor-go-sdk	Casnode, Casbin-OA, Confita

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

For a full list of the official Casdoor SDKs, please see: 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。 例举Casdoor-go-sdk 为示例：https://github.com/casbin/casnode/blob/6d4c55f5c9a3c4bd8c85f2493abad3553b9c7ac0/controllers/account.go#L51-L64

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

//go:embed token_jwt_key.pem
var JwtPublicKey string

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

`InitConfig()` 的所有参数解释为：

参数	是否必须	描述
endpoint	是	Casdoor 的服务URL，例如： `https://door.casdoor.com` or `http://localhost:8000`

参数	是否必须	描述
clientId	是	Casdoor 应用程序的客户端 ID
clientSecret	是	Casdoor 应用程序的客户端密钥
jwtPublicKey	是	Casdoor 应用程序证书的公钥
organizationName	是	Casdoor 组织的名称
applicationName	否	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?");
}
```

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";
import * as Setting from "./Setting";

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

  componentWillMount() {
    this.login();
  }

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

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

3. 获取登录 URL

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

4. 获取并验证token

步骤如下：

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

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

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

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

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

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

4. 用token识别用户

> **信息**
>
> 这一部分实际上是您的应用程序本身的业务逻辑，而不是OIDC、OAuth 或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。

如何启用单点登录

简介

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

我们提供这个单点登录，您只需要：

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

> **备注**
>
> Casdoor 提供的基本登录过程允许用户通过 选择当前登录的用户或使用另一个账户登录到组织中的其他应用程序。
>
> 启用自动登录后，选择框将不显示，登录用户将直接登录。

配置

1. 填充字段 **首页**。 它可以是应用程序的主页或登录页面。

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

添加静音签名

事实上，我们通过在URL上传参数来实现自动登录。 所以您的应用程序需要有一种方法来在跳转到URL后触发 登录。 我们提供了 casdoor-react-sdk 以便您快速实现此功能。详细信息请参阅use-in-react.

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

Add Popup Signin

`popup signin` will pop up a small window. After logging in to Casdoor in the child window, it will send authentication information to the main window and then close automatically. We implement it by carrying parameters on the URL.

> **信息**
>
> How to use
>
> Use the method `popupSignin()` in sdk casdoor-js-sdk to quickly implement the feature. You can see a demo in casdoor-nodejs-react-example.

> How it works
>
> 1. In the URL to the application home page, we will carry the `popup` parameter.
> 2. When `popup=1` in login params, Casdoor will send `code` and `state` as a message to main window and finish get `token` in main window by SDK.

使用 SSO

配置已完成，下面将展示如何使用自动登录。

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

Open the profile page and go to the "Home" page (`/` URL path). You will see the application list provided by the organization. It's notable that only users in organizations other that `built-in` can see the application list in the "Home" page. All the global administrators (aka in the `built-in` organization) cannot see it.

Click on a tile in the application list, it will jump to the homepage URL of that application with GET parameter: `?silentSignin=1` and automatically log into the application if the application has integrated with Casdoor SSO (so it will recognize the `?silentSignin=1` parameter and perform silent login in the background).

Vue SDK

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

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

这个插件还在开发中！ 如果您有任何问题和建议，请联系我们 issue

我们将向您展示以下步骤。

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

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

安装

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

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

Init SDK

初始化需要 5 个参数，它们都是字符串类型:

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

install:

对于Vue3:

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

对于Vue2:

```
// in main.js
import Casdoor from 'casdoor-vue-sdk'
import VueCompositionAPI from '@vue/composition-api'
const config = {
  serverUrl: "http://localhost:8000",
  clientId: "4262bea2b293539fe45e",
  organizationName: "casbin",
  appName: "app-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();
    }
  }
}
</script>
```

自动修复

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

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

关于Vue 版本的更多信息：请参见 vue-demi 文档

桌面 SDK

Electron 应用

一个 Electron 桌面应用程序用于Casdoor的示例

Dotnet桌面应用程序

一个 Dotnet 桌面应用程序用于Casdoor的示例

Qt 桌面应用程序

一个 Qt 桌面应用程序用于Casdoor的示例

Electron 应用

一个为 Casdoor的 Electron 桌面应用程序示例。

如何运行示例

初始化

您需要初始化6个参数，它们都是字符串类型：

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

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

可用命令

在项目目录中，您可以运行：

`npm run dev` 或者 `yarn dev`

构建electron应用并运行此应用。

`npm run make` 或者 `yarn make`

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

```
// 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

在您运行此electron应用程序后，将在您的桌面上显示一个新的窗口。

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

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

您可以通过下面的Gif图像预览整个过程。

如何整合?

设置自定义协议

首先，您需要设置自定义协议名为 `casdoor`。

```
const protocol = "casdoor";

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

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

通过浏览器打开登录网址

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

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

您可以更改前5个参数。

监听开启的应用程序事件

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

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

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

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

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

解析代码并获取用户信息

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

最后，您可以解析代码，并按照OAuth文档页面获取用户信息。

Dotnet桌面应用程序

一个为 Casdoor的 Dotnet 桌面应用程序示例。

如何运行示例

前置要求

dotnet6 sdk

webview2 runtime (已经在您的窗口中预装了)

初始化

初始化需要 5 个参数，它们都是字符串类型:

名称	描述	文件
Domain	您的 Casdoor 服务器主机/域	`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路径将是 `casdoor://callback` 如果没有	`CasdoorVariables.cs`

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

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

运行

Visual Studio

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

命令行

1. cd src/DesktopApp
2. dotnet run

效果预览

在运行此dotnet桌面应用程序后，您的桌面将显示一个新窗口。

如果您点击 `Casdoor Login` 按钮，您的桌面将显示登录窗口。

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，桌面上将显示一个用户配置文件窗口。 它显示您的用户名。

您可以通过下面的 gif 图像预览整个过程。

如何整合?

打开登录窗口

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

使用认证码获取用户信息

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

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

    return token?.AccessToken;
}

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

Qt 桌面应用程序

一个为 Casdoor的 Qt桌面应用程序示例。

如何运行示例

前置要求

Qt6 sdk

OpenSSL 工具包

初始化

您需要初始化7个参数，它们都是字符串类型：

名称	描述	文件
endpoint	您的 Casdoor 服务器主机/域	`mainwindow.h`
client_id	您的 Casdoor 应用程序的客户端 ID	`mainwindow.h`
client_secret	您的Casdoor应用程序的客户端密钥	`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`

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

运行

Qt Creator

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

效果预览

在运行此Qt桌面应用程序后，您的桌面将显示一个新窗口。

如果您点击 `Sign In` 按钮，您的桌面将显示登录窗口。

成功登录后，桌面上将显示一个用户配置文件窗口，它会显示您的信息。

您可以通过下面的 gif 图像预览整个过程。

如何整合?

打开登录窗口

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

监听开启的应用程序事件

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

使用认证码获取用户信息

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

Casdoor插件

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

Casdoor 插件	语言	源代码
Spring Boot插件	Java	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

完整的Casdoor官方插件列表请查看： Casdoor repositories。

OAuth 2.0

介绍

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

如何获取AcessToken

You have two ways to get the AccessToken: you can use the Casdoor SDKs, for details please refer to the SDK documentation, here we will mainly show you how to use the API to get the AccessToken.

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

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

获取授权码

首先重定向您的用户请求到:

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

可用的作用域（scope）

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

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

当您的用户通过casdoor身份验证后，他的请求会被casdoor重定到：

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

现在您已经获得授权码，在你的后端应用发送 POST 请求：

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

在你的后端应用

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

您将得到以下响应：

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

> **备注**
>
> Casdoor也支持 PKCE 功能。 当发送获取授权码请求时，您可以通过添加两个参数来启用 PKCE。
>
> ```
> &code_challenge_method=S256&code_challenge=YOUR_CHANLLENGE
> ```
>
> 获取令牌时，您需要通过 `code_verifier` 参数来验证 PKCE 。 值得一提的是，启用PKCE 后，Client_secret并不是必需的，但如果您要发送这个参数，它的值就必须是正确的。

隐式授权

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

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

当您的用户通过casdoor身份验证后，他的请求会被casdoor重定到：

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

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

使用资源拥有者的密码凭据授权

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

首先，您需要确保您已启用密码凭证授权，并发送一个 POST 请求：

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

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

您将得到以下响应：

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

使用客户端凭据授权

当应用程序没有前端时，您也可以使用客户端凭据授权。

首先，您需要确保您已启用客户端凭据授权，并发送一个 POST 请求到 `https://<CASDOOR_HOST>/api/login/oauth/access_token`：

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

您将得到以下响应：

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

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

更新访问令牌

如果您想要更新访问令牌，您可以使用上面的 `refreshToken`。

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

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

您将得到响应:

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

如何验证访问令牌

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

```
POST /api/login/oauth/introspect HTTP/1.1
Host: CASDOOR_HOST
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Authorization: Basic Y2xpZW50X2lkOmNsaWVudF9zZWNyZXQ=

token=ACCESS_TOKEN&token_type_hint=access_token
```

您将得到以下响应:

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

如何使用访问令牌

您可以使用AccessToken访问需要认证的 Casdoor API。

例如，请求 `/api/userinfo` 的两种不同方法。

方法 1 查询参数:

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

方法 2 HTTP Bearer token

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

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

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

如果您需要更多用户信息，在申请访问令牌获取授权码这一步添加更多 `scope` 。

`userinfo` 和 `get-account` API 之间的差异

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

CAS

使用 Casdoor 作为 CAS 服务器

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

简介

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

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

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

See https://apereo.github.io/cas/6.6.x/protocol/CAS-Protocol-Specification.html for more information about CAS and its different versions, as well as parameters for these endpoints.

一个示例

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

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

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

默认情况下，此应用使用 CAS 3.0，由以下配置指定。

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

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

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

如果您想要使用 CAS 1.0，请使用

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

对于参数“casServerUrlPrefix”，将其更改为

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

对于参数“casServerLoginUrl”，将其更改为

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

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

> **备注**

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

SAML

Overview

Using Casdoor as SAML IdP

Keycloak

Using Casdoor as SAML IdP

Appgate (POST)

How to use Casdoor as SAML IdP for Appgate

Overview

Casdoor now can be used as SAML IdP. Up to now the Casdoor has supported the main feature of SAML 2.0 .

Configuration in SP

Generally, SP needs the `Single Sign-On`, `Issuer` and `Public Certificate` three required fields. Most of the SP can get these fields by uploading the XML Metadata file or the XML Metadata URL to autocomplete.

The metadata of SAML endpoint in Casdoor is `<Endpoint of casdoor>/api/saml/metadata?application=admin/<application name>`. Suppose the endpoint of Casdoor is `https://door.casdoor.com`, which contains an application called `app-built-in`. The XML Metadata endpoint will be:

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

And you can also find the metadata in the application edit page. Click the button to copy the URL and paste it in browser to download the XML Metadata.

Configuration in Casdoor IdP

Casdoor supports both GET and POST `SAMLResponse`. Casdoor need to know what types of request the SP supports, when Casdoor sends the `SAMLResponse` to SP. You need to configure the application in casdoor based on the `SAMLResponse` type supported by your SP.

> **信息**
>
> If you fill the `Reply URL`, Casdoor will send the `SAMLResponse` by **POST** Request. If the Reply URL is empty, Casdoor will use **GET** request. You might wonder how casdoor knows the `Reply URL` of the SP, if the `Reply URL` is empty. Actually, Casdoor can get the URL called `AssertionConsumerServiceURL` by parsing the `SAMLRequest`. And send the request with `SAMLResponse` to `AssertionConsumerServiceURL`.
>
> The `Reply URL` will overwrites the `AssertionConsumerServiceURL` in `SAMLRequest`.

- **Reply URL** Type in the URL of the ACS verifying the SAML response.

- **Redirect URL** Type in a unique name. This may be called `Audience` or `Entity`

`ID` in your SP. Make sure you fill the same `Redirect URL` here as in your SP.

User profile

After logged in successfully, the user profile in the `SAMLResponse` Casdoor returned has three fields. The attributes in the xml and the attributes of the user in casdoor are mapped as follows:

XML Attribute Name	User field
Email	email
DisplayName	displayName
Name	name

See https://en.wikipedia.org/wiki/SAML_2.0 for more information about SAML and its different versions.

An example

The gosaml2 is a SAML 2.0 implementation for Service Providers based on etree

and goxmldsig, a pure Go implementation of XML digital signatures. And we use this library to test the SAML 2.0 in Casdoor as below.

Suppose you can access Casdoor through `http://localhost:7001/`, and your Casdoor contains an application called `app-built-in` which belongs to an organization called `built-in`. The URLs, `http://localhost:6900/acs/example` and `http://localhost:6900/saml/acs/example`, should be added to the Redirect URLs in `app-built-in`.

```
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)
    }
```

Run the above codes and the console will display the following message.

```
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
```

Click the URL to authenticate, the login page of Casdoor will display.

You will get the response messages as below after authenticating.

Keycloak

Casdoor as a SAML IdP in keycloak

This guide will show you how to configure Casdoor and Keycloak to add Casdoor as a SAML IdP in Keycloak.

Add SAML IdP in Keycloak

Open Keycloak admin page, click **Identity Providers** and select **SAML v2.0** from the list of providers.

> **信息**
>
> You can visit Keycloak SAML Identity Providers documentation to get more detailed information.

Enter the **Alias** and the **Import from URL** in Keycloak IdP edit page. The content of **Import from URL** can be found in the Casdoor application edit page. Click **Import** and the SAML config will be filled automatically.

You should remember the **Service Provider Entity ID** and then save the configuration.

Configure SAML application in Casdoor

In the application edit page, add a redirect URL which the content of it is **Service Provider Entity ID** in Keycloak. And you should enable SAML compress for Keycloak.

Login using Casdoor SAML

Open the Keycloak login page and you can find the additional button that allows you to login to Keycloak using the Casdoor SAML provider.

Click on the button and you will be redirected to the Casdoor SAML provider for the authentication. After the successful authentication, you will be redirected back to Keycloak. Then you need to assign users to the application.

We also provide a demo video to demonstrate the entire process, which we hope will be helpful to you.

Appgate (POST)

Casdoor as a SAML IdP in Appgate

Appgate accepts the `SAMLResponse` sent by POST Request. If you use other SP that also supports POST request, you can refer to this document.

Casdoor configuration

Go to your Casdoor and add a new application.

Enter basic SAML configuration in the application:

- Redirect URLs – Type in a unique name. This may be called `Audience` or `Entity ID` in your SP. See the table below.

- Reply URL – type in the URL of the ACS verifying the SAML response, refer to the table below

Administrator Authentication	User Authentication
Redirect URL = “AppGate”	Redirect URL = “AppGate Client”
SAML Reply URL = https://mycontroller.mycompany.com/admin/saml	SAML Reply URL = https://redirectserver.mycompany.com/saml

Download the XML metadata file

You can copy the URL of metadata and download the file from your browser.

Add SAML IdP in Appgate

In your AppGate SDP console:

- Select System > Identity Providers
- Create a new Identity Provider
- Choose the type SAML
- Start configuring your identity provider following the details in the tables below

	Administrator Authentication
Name	Enter a unique name eg: "Casdoor SAML Admin"
Single Sign-on URL	See below
Issuer	See below
Audience	Type in the **Redirect URL** from the Casdoor application
Public Certificate	See below

- Upload the XML Metadata file to autocomplete **Single Sign-On**, **Issuer** and **Public Certificate fields**
- Click **Choose a file** and select the metadata file that you created previously - this will autocomplete the relevant fields

Map Attributes

Mapping the **Name** to **username**, your completed form should look something like this:

Test Integration

On your AppGate SDP Controller console:

- Log out of the admin UI
- Log in using the following information:
- Identity Provider – choose your Azure IdP from the drop down list
- Click **Sign in with browser** to connect to your authenticator
- You may see the following message: “You don’t have any administration rights” – this confirms that the test user credentials have been successfully authenticated by your Identity Provider.

Access Policy

Your need to modify the access policy that the administrator can log in the Appgate by the SAML idp. Enter Builtin Administrator Policy:

Your completed form should look something like this:

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**

WebAuthn

概述

我们非常高兴地通知您，Casdoor现在支持通过 Webauthn 登录，这意味着： 您能通过生物识别登录，例如指纹或面部识别登录，甚至是通过U盘登录，但前提是您的设备支持这些授权方法和 WebAuthn。

什么是 WebAuthn?

Web身份验证API(也称为WebAuthn：https://webauthn.io/) 是由W3C和FIDO编写的规范，谷歌、Mozilla、微软、Yubico等也参与其中。 API允许服务器使用公用钥匙加密而非密码来注册和认证用户。 **它允许服务器与强大的身份验证器集成，现已编入设备，例如Windows Hello 或 Apple's Touch ID**。

简而言之，Webauthn要求用户生成公钥——私钥对，并将公钥移交给网站。 当用户想登录到网站时， Web 生成随机的数字，请用户用私钥加密，并将结果发送回来。 收到结果后，网站将尝试使用公钥解密， 并且如果解密后的数字与之前生成的随机数字相同。 该用户将会被视为合法用户，允许登录。 我们调用 Webauthn credential的公钥与必要的信息(例如用户名或滥用信息的用户授权者) 这正是网站存储的内容。

公钥-私钥对与这三种信息（用户名、用户授权以及网站URL）截然不同。 这意味着，如果(用户名、用户授权者和网站的URL) 是相同的，那么密钥对应该是相同的，反之亦然。

关于WebAuthn 技术的更多详细信息，可参阅 https://webauthn.guide/。

如何在Casdoor中使用 Webauthn？

在登录页面中，您肯定已看到使用 WebAuthn 登录的选择。 但考虑到您尚未获得

Webauthn凭据(也就是webauth密码，这样说您可能更好地理解)， 在这个教程中，我们将向您展示如何创建和管理凭据，如何再创作和管理以及如何使用凭据登录。

第 0 步: 修改配置并打开webauthn身份验证

您可以在conf/app.conf 中查看

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

请确保此配置是您网站的 URL。

只有https 支持webauthn，除非您正在使用本地主机

然后作为管理员登录，然后转到您的应用程序的编辑页面。 打开开关"启用WebAuthn signin"。 默认情况下，此功能未启用。

第 1 步：转到“我的帐户”页面

第 1 步：转到账户页面。 在这个页面，您可以看到"添加 WebAuthn Credential" 按钮和一个显示您以前注册过的所有Webauthn凭据的列表。

按下按钮，然后按照您设备的指示注册新凭据进入casdoor。

您可以通过列表中的“删除”按钮删除凭据。

第 2 步：通过webauthn登录

在这个步骤开始之前，请确保您已经退出casdoor的登录。

转到页面登录，选择webauthn登录方式， 输入您的用户名并按登录按钮，然后按照您设备的说明操作。

(例如，如果您使用指纹和 Windows Hello，您就会看到类似的东西)

然后您将看到您已经登录。

开发指南

前端

Casdoor 前端发展指南

生成 Swagger 文件

生成 Swagger 文件

前端

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

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

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

在 `/src` 中，有以下几个重要文件或文件夹：

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

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

生成 Swagger 文件

概述

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

如何写comment

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

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

具有相同"@Tag"标签的 api 将会被放入同一个组。

如何生成swagger文件

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

```
buid -o mybee
```

3. 复制mybe到casdoor的基础目录
4. 在该目录中运行

```
mybee generate docs
```

之后你会发现生成新的swagger文件。

组织

概述

Casdoor的基本单位 —组织

账户自定义

自定义用户帐户项目

Customize theme

Customize themes for organizations and applications under the organization

概述

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

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

我们还可以在Cassdoor建立LDAP。 欲了解更多详情，请参阅 文档。

Casdoor提供了多种密码存储算法，可在组织编辑页面中选择。

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

名称	算法	描述	场景
	和 PBKDF2	够抗拒 dictionary attacks 和 rainbow table attacks。 它最初在 Casdoor 里用于Keycloak 语法。 如果您通过Keycloak 同步方式导入用户，请选择此选项。	

提示

Besides logging into Casdoor via an application (which redirects to Casdoor for SSO), a Casdoor user can also choose to directly log into Casdoor via the organization's login page: `/login/<organization_name>`, e.g., https://door.casdoor.com/login/casbin in the demo site.

账户自定义

介绍

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

当您自定义一个组织中的账户项目时，此配置 将对该组织所有成员的主页生效。

如何定制?

帐户项目有四个属性:

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

输入组织编辑页面，您可以找到以下内容:

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	

Casdoor 提供非常简单的操作来配置：

- 将项目设置为可见或不可见

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

- 设置查看和修改规则

有 3 条规则:

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

账户列表

以下是帐户项目中的所有字段。 关于描述详情，您可以看 用户。

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

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

Customize theme

Casdoor allows you to customize theme to satisfy UI diversity from business or brand requirements, including primary color, border radius.

In Casdoor, the scope of theme includes global, organization, and application.

1. Global scope: this is the default theme of Casdoor and is applied to any organization that chooses to follow the global theme. It can only be modified in Casdoor source code, there is no way to modify it in web UI.

2. Organization scope: the theme for an organization can be customized in the organization edit page. The theme takes effect in all the Casdoor after-login pages for the users in the organization and the entry pages (signup, signin, forget password, etc.) of the applications that follow the organization theme.

3. Application scope: the theme for an application can be customized in the application edit page. The theme takes effect in the the entry pages (signup, signin, forget password, etc.) of the application.

Customize organization theme

We provide a demo to demonstrate how to config theme in organization:

> **信息**
>
> If your account organization is same as the organization you are editing, after you save the configuration, it will take effect immediately as the video above show. But if they are different, you need to log in the organization to see the effect.

Customize application theme

Applications customize theme use the same theme editor as the organization. But even more conveniently, you can preview the theme in the preview panel.

Preview :

Copy signup page URL

Copy signin page URL

Background URL

应用

概览

Casdoor应用概述

术语参考

术语参考

应用程序配置

配置您的应用程序身份验证

注册项目表

配置注册项表以创建自定义注册页面

自定义登录界面

自定义您的应用程序登录页面

概览

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

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

这样用户在访问组织中的所有应用程序时就无需重复登录。

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

在本章中，您将学到如何从零开始将Casdoor配置进您自己的程序。

让我们一起探索吧！

术语参考

- `Name` 被创建的应用程序的名字
- `CreatedTime` 应用程序被创建的时间
- `DisplayName` 应用程序向公众显示的名称
- `Logo` 应用程序Logo将显示在登录和注册页面
- `HomepageUrl` 应用程序主页的 url
- `Description` 描述应用程序
- `Organization` 应用程序所属的组织
- `EnablePassword` 用户是否可以通过密码登录
- `EnableSignUp` 用户是否可以注册。 如果没开启，应用程序不能注册新用户
- `SignupItems` 用户注册时需要填写的条目
- `Providers` 为应用程序提供的服务(例如OAuth、电子邮件、短信服务)
- `ClientId` OAuth客户端ID
- `ClientSecret` OAuth客户端密码
- `RedirectUries` 如果用户成功登录，Casdoor将导航到uries中的一个uri
- `TokenFormat`: The format of the generated token. It can be either `JWT` (containing all `User` fields) or `JWT-Empty` containing all non-empty values
- `ExpireInHours` Login will expire after hours
- `SigninUrl`
- `SignupUrl` If you provide a sign up service independently out of Casdoor, please fill the url here
- `ForgetUrl` Same as `SignupUrl`
- `AffiliationUrl`

应用程序配置

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

让我们看看如何使用Casdoor配置您的应用程序的身份认证系统吧！

> **备注**
>
> 例如，在这里，我想要使用 Casnode 设置我的论坛

创建我的应用程序并填写一些必要的配置。

选择我创建的组织，来使这个组织中的用户可以使用此应用程序。

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

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

> ⚠ **注意事项**
>
> 所以，我们需要记住提供商应用程序中的 `callback URL` 是 **Casdoor的回调url**，而Casdoor中的 `Redirect URL` 是 **您网站的callback url**。
>
> **更深层次的理解**
>
> 如果我想要认证进度正常运行，详细步骤如下：

> 用户发送请求到 Casdoor，Casdoor 使用 `Client ID`（客户端ID） 和 `Client Secret` （客户端密钥）获取GitHub、谷歌或其他供应商的身份验证。
>
> 如果身份验证成功，GitHub 回调到 Casdoor 告诉 Casdoor 验证成功， 因此 GitHub 授权callback URL应该是我的Casdoor callback URL，它是 http://your-casdoor-url.com/callback, 然后Casdoor 告诉应用程序身份验证成功，意味着Casdoor callback URL应该变成了我的应用程序的callback URL, 也就是 http://your-site-url.com/callback。

然后你可以通过添加提供商和设置它的属性来添加第三方应用程序。

You need to enable JavaScript to run this app.

提示

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

Token expire :
168
Hours
Password ON :
Enable signup :

注册项目表

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

关于每个注册项的详细说明，请见下表。

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

参数	可选值	描述
	`Items`	细规则见下表。
Action	-	用户可以将这个注册项向上移动、向下移动，或删除这个注册项。

So far, the signup items that support configuration rules include `ID`, `Display name`, `Email` and `Agreement`.

规则	可选规则	描述
ID	`Random` / `Incremental`	选择用户ID是随机生成还是递增生成。
Display name	`None` / `Real name` / `First, last`	选择名称的展示方式。 选择 `None` 将展示 `Display name`， 选择 `Real name` 将展示 `Real name`， 选择 `First, last` 将展示 `First name` 和 `last name`。
Email	`Normal` / `No verification`	选择是否验证邮箱验证码。 选择 `普通` 将验证邮箱验证码。 选择 `No verification` 则不验证邮箱验证码。
Agreement	`None` / `Signin` / `Signin (Default True)`	Select whether the user needs to confirm terms of use when logging in. Choose `None` to not display terms of use, and users can log in directly. Choose `Signin` to require users to confirm the terms before logging in. Choose

规则	可选规则	描述
		`Signin (Default True)` to set the terms confirmed by default, and users can log in directly.

> **备注**
>
> 例如，我想设置需要注册邮箱的注册页面，但不需要邮件验证码来验证此邮箱。

首先，我添加了一些注册所需的注册项目，例如ID、用户名、密码、电子邮件。

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

Casdoor

* Username:

* Password:

* Email:

Sign Up

Have account? sign in now

自定义登录界面

您已创建应用程序。 下面将向您展示如何自定义应用程序的登录界面。 在本指南中，我们将创建以下应用程序登录页面：

让我们开始吧！

1、添加背景图像

首先，我们要添加背景图像。 默认背景是白色的，看起来很简约。

- `背景URL` 背景图像URL。

选择您喜欢的背景图像并填写 `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。

2、自定义登录面板

现在我们在第一步操作结束的页面：

现在您需要添加一些css使面板看起来更美观。 您可以复制下面的代码并粘贴到字段 `Form CSS` 中。

```
<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。 但并不是填在字段中。 您需要复制内容并粘贴。

:::

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。

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：

4、启用侧面板

您现在将看到如何启用侧面板并自定义样式。

首先选择按钮。 在 **启用侧面板** 模式时，面板在中间位置。

然后编辑 `侧面板HTML`，它决定了侧面板中显示的内容。 与 `Form CSS`相同，我们提供了一个默认模板。 复制粘贴即可。

```
<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。 带有徽标和图像的侧面板已显示出来，但整体效果看起来差强人意。

您需要修改 `Form CSS` 中的一些css。

最后代码如下。

```
<style>
```

> **信息**
>
> `.login-panel`, `.login-form` 是div的类名。 它们对应于页面的不同区域。 欲了解更多详情，您可以通过开发者工具搜索。 在确认类名称后，您可以在这里填写 CSS 以更灵活的方式自定义登录页面。

最后，我们便设置好了一个美观的登录页面。

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。

更多关于Casdoor应用程序的介绍：

- Customize theme Customize the theme, including primary color, border radius.
- 注册项目表
- 应用程序配置

感谢您的阅读！

权限

概述

使用 Casbin 管理用户在组织中的访问权限

权限配置

使用 Casbin 管理用户在组织中的访问权限

开放的 Casbin API

使用 Casbin 管理用户在组织中的访问权限

Adapter

Config adapter and basic CRUD to policy

概述

Introduction

Casdoor 中每个组织的应用程序都共享该组织中的所有用户，因此这些用户都可以访问这些应用程序。 有时您可能想限制用户访问某些应用程序或某个应用程序中的某些资源。 In this case, you can use `Permission` implemented by Casbin.

在继续深入前，你应该对 Casbin 的工作原理及其相关概念有所了解，例如 Model、Policy、Adapter。 简而言之，Model 定义了你的权限策略结构，且请求应当如何匹配这些权限策略以及其所产生的作用。 Policy 则是你具体权限规则描述。 Casbin 在获得了 Model 和 Policy 信息后便可对到来的请求强制执行权限控制。 Adapter 作为抽象层为 Casbin 的执行器屏蔽了 Policy 的来源，使得 Policy 可以存放于各处，例如文件或数据库。

回到 Casdoor 的权限配置话题。 在 Casdoor Web UI 中，你可以在 `Model` 配置项中为你的组织添加 Model，并在 `Permission` 配置项中为你的组织添加 Policy。 借助 Casbin Online Editor，你可以得到适用于你的使用场景的 Model 和 Policy 文件。 你可以很容易地将 Model 文件通过Casdoor Web UI 导入至 Casdoor 中供内置 Casbin 使用。 但对于 Policy 而言（即 Casdoor Web UI 中的 `Permission` 配置项），则需要在此进行一些额外的说明。 让我们在后文继续提及。

正如你的应用需要通过 Casdoor 内置的 Casbin 来强制实施权限控制一样，作为一个内置应用，Casdoor 内部也通过 Casbin 使用自己的 Model 和 Policy 来控制 API 接口的调用权限。 然而，Casdoor 可以通过内部代码来调用 Casbin，而外部应用却无法做到。 因此，Casdoor 为向外部应用暴露了调用内置 Casbin 的 API 接口。 我们将在后文向你展示这些 API 接口的定义及其使用方式。

在本章的末尾，我们将使用一个实际例子来向您展示外部应用程序如何与 Casdoor 协作

进行权限控制。

让我们开始吧！

权限配置

让我们依次解释 Permission 配置页面中的每一项。

- `Organization`：该策略所属的组织名， 一个组织可以拥有多个权限策略文件。
- `Name`：权限策略名称，在组织中全局唯一。 用于标识该权限策略文件。
- `Display name`：没什么重要的。
- `Model`：描述权限策略结构及其匹配模式的模型文件名称。
- `Adapter`：注意！ 在当前版本中该字段描述的是存放权限策略的数据库表名，而非在 Casdoor Web UI 中的 Adapter 菜单项所配置的适配器名称。 Casdoor 会使用其自身的数据库来存放正在配置的权限策略。 如果该字段为空，权限策略将被存储至 `permission_rule` 表中，否则将存储在指定的数据库表中。 如果指定的表名在 Casdoor 所使用的数据库中不存在，将会自动创建。 我们强烈建议**为不同模型指定不同的适配器**， 因为将所有的策略保存在同一个表格中可能导致冲突。
- `Sub users`：哪些用户将被应用该权限策略。
- `Sub roles`：如果使用了RBAC模型，哪些角色将被应用该权限策略。 这将为该角色中的每一个用户添加诸如 `g user role` 的权限策略。
- `Sub domains`：哪些域将被应用该权限策略。
- `Resource type`：事实上，在当前版本中，对于外部希望鉴权的应用来说，Casdoor 并没有使用该字段。 你可以暂且忽略它。
- `Resources`：这个字段描述了你希望强制实施权限控制的资源。 但请注意，这里的资源并非在 Casdoor Web UI 的 Resources 菜单项中所配置的资源。 你可以在此处添加任何你所希望的字符串，例如一个 URL 或者一个文件名。
- `Actions`：这个字段描述了操作资源的动作。 与资源相同，它可以是任何你所希望的字符串，例如 Http Method 或其他自然语言。 但是请注意，Casdoor 会将这些字符串全部转化成为小写再存储。 另外，Casdoor 将会为每一个资源应用所有的动作。 你无法指定某个动作只对某些资源生效。

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

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

开放的 Casbin API

介绍

让我们假设你的应用前端已经获取到了已登录用户的 `access_token`，并且现在希望为用户的某些访问进行权限鉴定。 你不能简单地将 `access_token` 放置到 HTTP 请求头中来使用这些 API，因为 Casdoor 使用 `Authorization` 字段来检查访问权限。 像其他由 Casdoor 提供的 API 一样，`Authorization` 字段由应用的 client id 和 secret 组成，使用基本 HTTP 认证方案。 它看起来像 `Basic XXX`。 因此，Casbin API 应当被应用的后端服务器调用。 以下是关于如何使用的步骤。

1. 前端通过 HTTP 请求头将 `access_token` 传递到后端服务器。
2. 后端服务器从 `access_token` 中获取用户 ID，这也是下文所描述的 API 中的参数 `v0`。

提前说明，这些接口也几乎是为 `(sub, obj, act)` 模型所设计的（就现阶段而言）。 接口中的 `id` 为所应用的权限策略标识，由组织名和权限策略名组成（即 `organization name/permission name`）。 `v0`、`v1`、`v2`依次对应权限模型所描述的策略结构，通常分别代表`sub`、`obj`、`act`。

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

Enforce

请求:

```
curl --location --request POST 'http://localhost:8000/api/enforce'
```

响应:

```
true
```

BatchEnforce

请求:

```
curl --location --request POST 'http://localhost:8000/api/batch-enforce' \
--header 'Content-Type: application/json' \
--header 'Authorization: Basic client_id_and_secret' \
--data-raw '[{"id":"example-org/example-permission", "v0":"example-org/example-user1", "v1":"example-resource", "v2":"example-action"}, {"id":"example-org/example-permission", "v0":"example-org/example-user2", "v1":"example-resource", "v2":"example-action"}, {"id":"example-org/example-permission", "v0":"example-org/example-user3", "v1":"example-resource", "v2":"example-action"}]'
```

响应:

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

GetAllObjects

请求:

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

响应:

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

GetAllActions

请求:

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

响应:

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

GetAllRoles

请求:

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

响应:

```
[
    "role_kcx66l"
]
```

Adapter

Casdoor supports using the UI to connect the adapter and manage the policy rules. In Casbin, the policy storage is implemented as an adapter (aka middleware for Casbin). A Casbin user can use an adapter to load policy rules from a storage, or save policy rules to it.

Adapter

- `type` : Adapter type. Now support database adapter.
- `Host`
- `Port`
- `User`
- `Password`
- `Database type` : Now support MySQL, PostgreSQL, SQL server, Oracle, SQLite 3.
- `Database` : The database name.
- `Table` : The table name. If the table does not exist, it will be created.
- `model` : You can select one model belonging to the organization of the adapter.

Edit Adapter Save **Save & Exit**

Organization :	built-in
Name :	casdoor_adapter
Type :	Database
Host :	localhost
Port :	3306
User :	root
Password :	123456
Database type :	MySQL
Database :	casdoor
Table :	casbin_rule
Model :	casbin_rule
Policies :	Sync

> **信息**
>
> After fill all the fields, please don't forget to **save** the config. Then click the **sync** button to load the policy rules. The policy rules will be shown in the below table.

Policies : Sync

Add

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

< 1 2 3 4 5 >

Is enabled :

Basic CURD

If you connect the adapter successfully, you can make basic CURD to the policy rules.

- Add

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

1 2 3 4 5

> **提示**
>
> You can only add one policy at one time. The newly added policy is in the first row in the table, but actually, it will be saved in the last row. So next time you sync the policies, they will appear in the last row of the table.

- Edit

Model :
casbin_rule
Policies :
Sync
Add
Rule Type V0 V1 V2 V3 V4 V5 Option
p built-in * POST * * *
p app * * * * *
p * * POST /api/signup * *
p * * POST /api/get-email-and-phone * *
p * * POST /api/login * *
p * * GET /api/get-app-login * *
p * * POST /api/logout * *
p * * GET /api/logout * *
p * * GET /api/get-account * *
p * * GET /api/userinfo * *
1 2 3 4 5

- Delete

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

19 个项目

电子邮箱

使用电子邮件来完成身份验证

短信

使用短信来完成身份验证

🗃️ 存储

3 个项目

🗃️ SAML

3 个项目

🗃️ 支付

2 个项目

🗃️ 验证码

6 个项目

概述

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

我们所拥有的

现在，我们有六种类型的提供商：

- **OAuth 提供商**

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

- **短信提供商**

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

- **电子邮件提供商**

  电子邮件提供者与短信提供者类似。

- **存储提供商**

  Casdoor允许用户使用本地文件系统或云端服务存储文件。

- **支付服务提供商**

  Casdoor 可以添加付款提供者，用于在产品页面上添加付款方法。 目前，受支持的

付款提供者包括支付宝、微信支付、PayPal和GC。

- **验证码提供商**

  Casdoor支持在用户流程中配置验证码。 Currently, the supported captcha providers include Default Captcha, reCAPTCHA, hCaptcha, Aliyun Captcha and Cloudflare Turnstile.

如何配置和使用

范围

提供商有不同的范围。 提供者的范围取决于创建者。 只有管理员有权添加和配置提供者。 Cassdoor有两种管理员。

- **全局管理员**: 内置 `下的所有用户` 组织和用户启用 `IsGlobalAdmin` 由全局管理员创建的供应商可以被所有应用程序使用。

- **组织管理员**: 用户启用 `Ismin`. 组织管理员创建的提供者只能使用 **组织下的应用程序只能使用**。 (正在开发...)

添加应用程序

以下是为您的应用程序添加提供商的步骤。 在没有添加提供商之前，你还不能在应用中使用。

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。 这里将显示应用程序可以使用的所有提供商。

3. 对于 OAuth and Captcha 提供商，您可以配置该用法。 详见 OAuth and Captcha

最后, **保存** 配置。 然后您可以尝试在应用程序中使用提供商。

OAuth

概述

将 OAuth 提供商添加到您的应用程序

自定义提供商

添加您自己的自定义OAuth 提供商

Twitter

添加Twitter OAuth 提供商到您的应用程序

微博

将 Weibo OAuth 提供商添加到您的应用程序

微信

将Wechat OAuth 提供商添加到您的应用程序

企业微信

将 WeCom OAuth 提供商添加到您的应用程序

腾讯 QQ

添加腾讯QQ OAuth提供商到您的应用程序。

钉钉

添加钉钉 OAuth 到您的应用程序

Steam

将 Steam OAuth 添加到您的应用程序

GitHub

添加Github OAuth 提供商到您的应用程序

Gitee

添加Gitee OAuth 提供商到您的应用程序

领英（LinkedIn）

添加Linkedin OAuth 提供商到您的应用程序

Facebook

将 Facebook OAuth 提供商添加到您的应用程序

谷歌

添加Google OAuth 提供商到您的应用程序

百度

向您的应用程序添加 Baidu OAuth 提供商

AD FS

添加AD FS 作为第三方服务来完成身份验证

AzureAD

添加 AzureAD 作为第三方服务来完成身份验证

Infoflow

在应用程序中添加 Infoflow OAuth 提供商

Okta

将 Octa OAuth 提供商添加到您的应用程序

概述

Cadoor 可以使用其他 OAuth 应用程序登录。

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

Provider	Logo	Provider	Logo	Provider	Logo	Provider	Logo
Adfs		Alipay		Amazon		Apple	
Auth0		AzureAD		Baidu		Battle.net	
Bilibili		Bitbucket		Box		Casdoor	
Cloud Foundry		Dailymotion		Deezer		DigitalOcean	
DingTalk		Discord		Douyin		Dropbox	
Eve Online		Facebook		Fitbit		Gitea	
Gitee		GitHub		GitLab		Google	
Heroku		InfluxCloud		Infoflow		Instagram	
Intercom		Kakao		Lark		Lastfm	

Provider	Logo	Provider	Logo	Provider	Logo	Provider	Logo
Line		LinkedIn		Mailru		Meetup	
MicrosoftOnline		Naver		Nextcloud		Okta	
OneDrive		Oura		Patreon		Paypal	
QQ		SalesForce		Shopify		Slack	
SoundCloud		Spotify		Steam		Strava	
Stripe		TikTok		Tumblr		Twitch	
Twitter		Typetalk		Uber		VK	
WeChat		WeCom		Weibo		Wepay	
Xero		Yahoo		Yammer		Yandex	
Zoom		Email		SMS			

我们将向您展示如何申请第三方服务并将其添加到Casdoor。

申请成为开发者

在此之前，你需要理解一些概念。

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

添加 OAuth 提供商

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

应用中

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

自定义提供商

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

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

- `Auth URL` 是自定义提供商的 OAuth 登录页面地址。

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

录到此自定义提供商时，浏览器将先跳转到

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

授权完成后，自定义提供商应该重定向到

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

然后，此 URL 中的代码参数将会被 Casdoor 识别。

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

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

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

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

  自定义提供商至少应该返回以下内容

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

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

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

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

自定义提供商至少应该返回以下内容

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

- `Favicon` 是自定义提供商的标识URL。

  这个标识将与其他第三方登录供应商一起显示在Casdoor的登录页面上。

Twitter

Twitter(仍在开发中🚧)

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

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

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

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

微博

微博✔

申请微博的开发者账户并不困难， 但速度相对较慢。 大约需要2-3天。

访问 开发者网站，填写基本信息，等待审核通过。

审核通过后，您将得到 Client Id 和 Client Secret。

微信

微信:注意事项:

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

WeChat 提供商提供两套不同的密钥:

- 第一个密钥对 (`客户端 ID`, , `客户端密钥`) 是 `WeChat 打开平台(从二维码到二维码)`, 它仅适用于PC登录场景。 它可以在 PC 中显示二维码，用户可以使用手机扫描代

码。 PC 浏览器允许使用 WeChat 登录。

- 第二个密钥对 (`客户端 ID 2`, , `客户端 secretreet 2`是 `WeChat 媒体平台 (conctionscreen condition)`, 仅适用于WeChat-app 登录场景。 它允许用户使用 WeChat 在 WeChat 移动APP 内登录 它会跳转到您的 `WeChat 官方帐户 (微信公众号)` 以登录。 值得注意的是，在移动场景中，WeChat 本身不支持在 WeChat APP 以外的登录，比如在其他移动浏览器 (H5) 或 APP 中。 这是对WeChat而不是Casdoor的限制。

如果您填写了第二个密钥对(`客户端 ID 2`, `客户端密钥 2`并启用 `启用 QR 码` 开关， 当用户点击WeChat 按钮登录时， **casdoor将首先要求用户关注WeChat官方帐户 (从二维码中提取)**, 然后继续登录。 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。 在移动场景中使用castor 将自动跳过此步(在WeChat mobile APP中使用WeChat 内置浏览器)。

> **提示**
>
> 我们建议同时设置两套钥匙。 并链接您的 `WeChat 开放平台(即时扫描)` 账户和 `WeChat 媒体平台(即时连接)` 账户一起在 `WeChat 开放平台(即时连接)` 中 所以通过 PC 登录的 WeChat 用户和手机可以被识别为 Cassdoor 中的同一用户。
>
> :::
>
> > **备注**
> >
> > 由于 WeChat OAuth 的限制，目前没有方法通过 WeChat 登录到第三方移动APP 或 WeChat APP以外的移动浏览器。 必须在 WeChat 中移动登录。
>
> 欲了解更多详情，请访问 微信开放平台。

企业微信

介绍

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

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

基本设置

要配置 企业微信 提供商，下面的表格描述了所需参数。

参数描述:

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

> **信息**
>
> 企业微信有两种授权方法。 **静默** 授权和 **正常** 授权。
>
> **静默授权**: 在用户点击链接后，页面是 `重定向_URI? code=CODE&state=STATE`
>
> **正常授权**: 在用户点击链接之后, 一个中间页面将显示给用户以选择是否为 授权。用户确认授权后，转到 `重定向uri?code=CODE&state=STATE`
>
> 欲了解更多详情，请参阅 文档。

更多

欲了解更多关于内部应用程序的信息，请参阅 内部应用程序。

关于第三方应用程序，请参阅 第三方应用。

腾讯 QQ

腾讯QQ✔

访问QQ认证平台 - 连接 QQ。

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

钉钉

钉钉✔

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

欲了解更多信息，请访问 钉钉开放平台.

此外，您需要将以下权限添加到钉钉中：

Steam

Steam✔

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

欲了解更多详情，请访问 Steam WebAPI doc。

GitHub

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

首先，请访问 GitHub 开发者设置 注册一个新的 GitHub 应用。

注意事项

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

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

> **正确设置授权回调URL**
>
> 在 Github OAuth 配置中，`应用回调地址` 必须是 **您的Casdoor的回调URL**并且 Casdoor 中的 `Redirect URL` 应该为 **您的应用回调地址**
>
> 更多详情请阅读 应用配置

注册您的 GitHub 应用程序后，您现在可以生成您的 `客户端密钥`！

添加一个 GitHub OAuth 提供商并填写 `客户端ID` and `客户端密钥`

现在您可以使用 GitHub 作为第三方服务来完成认证。

Gitee

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

然后您可以创建您的Gitee应用程序。

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

> **正确设置授权应用回调地址**
>
> 在 Gitee OAuth 配置中， `应用回调地址` 必须是 **您的Casdoor的回调URL**并且 Casdoor 中的 `Redirect URL` 应该为 **您的应用回调地址**
>
> 更多详情请阅读 应用配置

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

添加一个 Gitee OAuth 提供商并在您的 Casdoor 中填写 `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，必须选中电子邮件选项，否则会导致授权错误。

领英（LinkedIn）

要设置 Linkedin OAuth 提供商，请前往 Linkedin 开发者 创建一个新的应用。

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

> **备注**
>
> 只有公司页面管理员可以验证您的应用，并允许您的应用

在您的应用验证后，您可以继续：

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， `应用回调URL` 必须是 **您的Casdoor的回调URL**并且 Casdoor 中的 `Redirect URL` 应该为 **您的应用回调地址**
>
> 更多详情请阅读 应用配置

然后你就可以获得你的`Client ID`和`Client Secret`

添加一个 Linkedin OAuth 提供商并在您的 Casdoor 中填写 `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，请前往 Facebook开发人员 创建一个新的应用程序。

选择您要创建的应用类型。

填写姓名并联系电子邮件后，您可以进入 facebook 开发者面板。

然后设置 Facebook 登录：

选择此应用的 Web 平台：

填写网站URL后，您可以访问 Facebook Login > **设置**并填写有效的 OAuth Redirect URI

> **正确设置授权的 REDIRECT URL**
>
> 在 Facebook OAuth 配置中， `有效的 OAuth Redirect URIs` 必须是 **您的 Casdoor 回调 url**, and the `Redirect URL` in Casdoor should be **your application callback url**
>
> 更多详情请阅读 应用配置

基本应用程序配置即将完成！

在仪表板顶部栏中切换模式从 In development 到 Live

然后您的 `App ID` and `App secrets` 可以在 Casdoor 中使用。

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

现在你可以使用 Facebook 作为第三方服务来完成身份验证！

谷歌

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

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

并注册您的 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

然后导航到 Credential 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

并为您的应用创建凭据:

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

> **正确设置授权重定向 URI**
>
> 在 Google OAuth 配置中， `应用回调URL` 必须是 **您的Casdoor的回调URL**并且 Casdoor 中的 `Redirect URL` 应该为 **您的应用回调地址**
>
> 更多详情请阅读 应用配置

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

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

`Secret`

现在你可以使用 Google 作为第三方服务来完成身份验证！

百度

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

在创建您的应用后，重定向URL设置在以下位置：

在以下位置添加您的 Casdoor 域名：

> **注意事项**
>
> 这部分与百度给出的文档中的实际情况有很大的不同:
>
> 1. 将URL添加到回调URL设置中的话很可能验证URL失败，导致登录失败，所以我们要将我们的域名添加到域名设置中。
> 2. 只能添加一个URL或域名，这与文档有很大不同。

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

添加一个百度 OAuth 提供商并在您的 Casdoor 中填写 `Client ID` 和 `Client Secrets`。

现在你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

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

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

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

AD FS

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

步骤1 通过 ADFS 启用 OAuth

这一步骤详情请参阅 如何通过 ADFS 启用 OAuth 。

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

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

激活 Casdoor ADFS Provider

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

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

AzureAD

介绍

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

如何使用?

注册应用程序的步骤如下所示。

步骤1. 注册一个应用程序

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

步骤2. 创建客户端密钥

创建一个 客户端密钥 并保存值，这将稍后使用。

步骤3. 添加重定向 URI

按照图片中的示例添加casdoor的重定向uri。

步骤4. 授予管理员权限

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

步骤5. 在casdoor中创建 AzureAD 提供商

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

Edit Provider Save Save & Exit

Name :	provider_casdoor_azuread
Display name :	Casdoor AzureAD
Category :	OAuth
Type :	AzureAD
Client ID	621cc0f0-055f-433f-9894-bfa1bfde169d
Client secret	***
Provider URL :	https://portal.azure.com/#view/Microsoft_AAD_RegisteredApps/Appli

Save Save & Exit

Infoflow

若要设置 Infoflow OAuth 提供商，请前往 Infoflow 并登录使用您的 Infoflow 帐户。

首先，请访问Infoflow的 应用中心。

并注册您的 Infoflow 应用。

然后你现在可以获取 `AgentID`。

然后导航到 **设置** 标签，并创建一个新的管理组。

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

添加敏感接口权限如下所示：

敏感接口权限 修改

接口名称	权限开放
获取部门成员	☑
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	☐

您将能够在同一页面看到 `CorpID` and `Secret`

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

现在您可以使用 Infoflow 作为第三方服务来完成身份验证！

Okta

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

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

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

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

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

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

> **正确设置域名**
>
> 请注意， `域名` 不仅是 `Okta 域名`, `/oauth2/default` 应该附加到它。
>
> 在授权服务器上访问 Okta 文档 获取更多详情。

现在你可以使用 Github 作为第三方服务来完成身份验证！

电子邮箱

添加电子邮件提供商

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

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

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

短信

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

添加短信提供商

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

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

4. 从阿里云、腾讯云或 Volc Engine 获取您的信息并填写。

示例

> **备注**
>
> 这里我使用阿里云短信服务作为示例

登录我的阿里云工作台后，点击AccessKey 来创建 ID 和密钥。

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

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

存储

概述

设置上传文件的 Casdoor 存储提供商

Azure Blob

使用 Azure Blob 作为Casdoor的存储提供商

Aliyun OSS

使用 Azure OSS作为Casdoor的存储提供商

概述

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

Casdoor 支持两种类型的批量操作 - Local 或 Cloud。 在本章中，您将学习如何添加一个存储提供商来使用这些服务。

项目

- `客户端ID`
- `客户端密钥`
- `地域节点 (外网)`
- `路径perfix`: 文件位置的路径前缀。

> **信息**
>
> 默认 `路径erfix` 是 `/`. 例如，当 `路径erfix` 为空时，默认文件路径:
>
> ```
> https://cdn.casbin.com/casdoor/avatar.png
> ```
>
> 您可以用 `abcd/xxxx`填充它，然后您可以将您的头像存储在:
>
> ```
> https://cdn.casbin.com/abcd/xxxx/casdoor/avatar.png
> ```

- `存储桶`

- `域`: 您的云存储的 CDN 自定义域名。

本地设置

使用 **本地** 类型，您唯一需要配置的项目是 `域` 字段。 请跟随引导操作

```
域/图像
```

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

但是 `127.0.0.1:7001/images` 是错误的。

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

云端

目前，我们支持 AWS S3, Aliyun OSS, Tencent Cloud COS, MinIO 和 Azure Blob 云服务商，正在添加更多云存储服务。

用 `Client ID`, `Client secret`, 填写相应字段 `Endpoint` 和 `Bucket` 从您的云端供应商控制台获得。

> **提示**
>
> 使用 `非区域` 类型，您可能不需要用于自定义域的 `域` 字段。
>
> :::

Azure Blob

> **备注**
>
> 这是 Azure Blob 的一个示例

- 您必须拥有一个 Azure storage 帐户。

步骤1. 选择 Azure Blob

选择 Azure Blob 作为存储类型。

步骤2. 填写 Casdoor 的必要信息

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

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

- 帐号名称

  `AccountName` 是您的帐户名称。

- 账户密钥

  `AccountKey` 是您访问 Azure API的密钥。

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

- 容器名称

  您首先需要创建一个容器。 也可以点default选择一个默认容器。

- 域名

您Azure CDN中的自定义域名。

步骤3. 保存您的配置

最终结果如下：

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

然后您可以在应用程序中使用阿里云云存储服务。

Aliyun OSS

> **备注**
>
> 这是 Azure OSS 的一个示例.

AccessKey 是您访问阿里云 API 的密钥，拥有完全的帐户权限。

所以 在阿里云工作台中创建了 AccessKey

然后创建 OSS 服务：

在 Casdoor 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

然后您可以在应用程序中使用阿里云云存储服务。

SAML

概述

使用来自支持SAML 2.0 的外部身份提供商的身份

Aliyun IDaaS

使用 Aliyun IDaS 验证用户

Keycloak

使用 Keycloak 验证用户

概述

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

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

阿里云 IDaaS	Keycloak
✅	✅

条款

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

SAML 集成工作方式

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

证。 进展情况见下图表。

User
:Browser
:SP
:IDP
Request target resources (via browser)
Generate SAML request url
Redirect to IDP SSO service
Parse SAML request
Identity the user
Response with XML form (via SP)
Redirect to /api/login
Parse and validate
SAML response
Response target resources
Display target resources

Aliyun IDaaS

在 Aliyun IDaaS 中创建 SAML 应用程序

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

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。

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

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**。

修改 Aliyun IDaS 的 SAML 应用程序

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。

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

casbin
username, Email or phone
Password
Auto sign in
Forgot password?
Sign In
Sign in with code
No account? sign up now

Keycloak

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

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

Configure Keycloak

一些配置选项和假设，特别是这个示例：

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

使用默认领域或创建一个新领域。

在 Keycloak 中添加客户端条目

> **信息**
>
> 在 Keycloak 文档 中查看更多关于Keycloak 客户端的详细信息。

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

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

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

以下设置列表部分:

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

保存该配置。

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

点击 **安装** 标签页。

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

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

在Casdoor配置

在 Casdoor 中创建一个新的提供商。

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

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

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

验证效果

转到您刚刚配置的应用程序，您可以在登录页面找到Keycloak图标。

点击图标跳转到Keycloak登录页面，验证后成功登录到Casdoor。

支付

支付宝

WeChatPay

Add Wechat OAuth provider to your application

支付宝

WeChatPay

Step1. Deploy Casdoor

Firstly, the 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, you need to ensure:

- Open your favorite browser and visit **http://localhost:8000**, you will see the login page of Casdoor.
- Input `admin` and `123` to test login functionality is working fine.

Then you can quickly implement a casdoor based login page in your own app with the following steps.

Step2. Configure payment callback notification address

Before that, please log in to the WeChat Pay Merchant Platform. In order for Casdoor to receive payment result notifications, you need to set a callback notification address in the WeChat Pay Merchant Platform.

- In Merchant Dashboard, go to `Development Configuration` → `Payment Configuration`.
- Find the `Callback Notification Address` setting, and click the `Modify`

button.

- Fill in the payment callback notification address of the Casdoor instance. For example:

```
https://your-casdoor-instance.com/api/wechat-payment-callback
```

Step3. Configure API Security

Get APIv3 key

To ensure the security of API calls, you need to configure API security settings in the WeChat Pay Merchant Platform.

Log in the WeChat merchant platform → `Account center` → `Account settings` → `API security` → `APIv3 key` → `set up`

Get the serial number of the merchant certificate

It is also necessary to obtain the serial number of the merchant certificate. The following are the steps to obtain it.

Log in the WeChat merchant platform → `Account center` → `Account settings` → `API security` → `API certificate management` → copy the serial number.

You can refer to the WeChat payment merchant ID query guide , APIv3 key settings and How to view the certificate serial number for help.

Step4. Add payment provider

Select the WeChat Pay as the Payment type.

Get Payment URL

Log in the WeChat merchant platform → `Commodity centered` → `Development arrangement` → `payment arrangements` → `Payment URL` → `copy`

Fill the necessary information in Casdoor

There are four required fields, `Client ID`, `Client secret`, `appId`, `Provider URL`. The relationship corresponding to the Azure Blob account is as follows:

Name	Name in WeChat Pay	is required
Client ID	merchant ID	required
Client secret	APIv3 key	required
appId	appId	required
Provider URL	Payment URL	required

The acquisition of `merchant ID` and `APIv3 key` is as mentioned before. For appId, see here for more help.

Step5. Add WeChat Cert

In this step, you need two required fields, `serial number`, `private key`.

How to obtain the serial number has been stated in the third step.

To get `private key`, click here for help.

Step6. Add WeChat Payment Provider in Products

Final effect:

What's more

You can explore the following projects/docs to learn more about WeChat Pay.

- wechatpay-apiv3-go-sdk
- Wechat payment development document

验证码

概述

添加验证码到您的应用程序

默认

在您的应用程序中使用 Casdoor 默认验证码

reCAPTCHA

添加reCAPTCHA 到您的应用程序

hCaptcha

将 hCaptcha 添加到您的应用程序

Aliyun Captcha

将Aliyun Captcha添加到您的应用程序

Cloudflare Turnstile

Add Cloudflare Turnstile to your application

概述

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

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

默认	reCAPTCHA	hCaptcha	Aliyun Captcha	Cloudflare Turnstile
✅	✅	✅	✅	✅

我们将向您展示如何申请验证码并将其添加到casdoor。

添加验证码提供商

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

在应用中使用

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

默认

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

在 Casdoor 配置

在 Casdoor 中创建一个新的提供商。

选择类别为 **验证码** , 类型为 hCaptcha.

你可以点击 **预览** botton来预览这个验证码的样式。

在应用中使用

编辑您想要在 Cassdoor 中配置的应用程序。 选择刚添加的提供商。 有两类规则：

- 登录时总是开启 ， 发送验证码。
- 关闭

reCAPTCHA

reCAPTCHA 由 Google提供。 我们使用 reCAPTCHA v2 Checkbox。 您可以从此 链接中看到更多详细信息。

创建一个 API 密钥对

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

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

然后你可以获得一个站点密钥和一个秘密密钥。

在 Casdoor 配置

在 Casdoor 中创建一个新的提供商。

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

你可以点击 **预览** botton来预览这个验证码的样式。

在应用中使用

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

hCaptcha

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

创建一个 API 密钥对

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

然后你可以获得一个站点密钥和一个秘密密钥。

在 Casdoor 配置

在 Casdoor 中创建一个新的提供商。

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

你可以点击 **预览** botton来预览这个验证码的样式。

在应用中使用

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

Aliyun Captcha

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

在 Aliyun 中添加验证码配置

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

输入验证码关联控制台后，点击 **添加配置**。

填写所有必需的信息并提交。

然后您可以在您的控制台中看到您的 `场景` and `App key`

访问密钥, 秘密访问密钥 在您的个人资料中。

在 Casdoor 配置

在 Casdoor 中创建一个新的提供商。

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

你可以点击 **预览** botton来预览这个验证码的样式。

下面的图像是 `滑动验证` 预览：

以下图像是 `智能验证` 预览：

在应用中使用

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

Cloudflare Turnstile

Cloudflare Turnstile is a captcha service provided by Cloudflare, which is a user-friendly, privacy preserving alternative to captcha. You can see more details from Turnstile Docs.

Create an API key pair

To start using Cloudflare Turnstile, you need to Create a Cloudflare account, navigate to the `Turnstile` tab on the navigation bar, and get the Site Key and Secret Key.

First, add a name for the widget to identify it in the future and enter your website's hostname. Then choose the widget type. `Managed` is recommended. Click **Create**.

Then you can get a site key and a secret key.

Configure in Casdoor

Create a new provider in Casdoor.

Select category as **Captcha** , type as **Cloudflare Turnstile** . And you need to fulfill the site key and the secret key which is created by last step.

And you can click the **Preview** button to preview the style of this captcha.

Applied in application

Edit the application you want to configure in Casdoor. Select the provider just added and click the button **Save**.

资源

概述

上传资源到Casdoor

概述

您可以在 casdoor 上传资源。 在上传资源之前，您需要配置一个存储提供商。 请查看 存储提供商

您现在至少配置了一个**存储提供程序**，并将该提供程序添加到你的**应用程序**中。

好了！ 让我们看看如何 **上传** and **删除** 资源的例子。

上传资源

用户可以将文件和图像等资源上传到先前配置的 云存储。

删除资源

如果你不再需要资源，你可以选择点击“删除”按钮来删除它。

产品

产品

添加您想要销售的产品

支付

查看支付中产品的交易信息

产品

您可以添加您想要销售的产品 (或服务)。 下面将告诉您如何添加产品.

配置产品属性

第一： 您需要了解产品的基本属性 标签 详情 货币 价格 数量 售价

支付服务提供商

当然，除了设置这些属性外，您还需要为产品添加付款提供者。 和多个付款提供者可以添加到产品。

了解如何配置付款提供商，请参阅 付款提供商

最后，填写 **返回** URL。 当付款完成时，这是从付款提供者页面跳转到的URL。

预览产品

搞定！ 查看评论并保存：

支付

付款成功后，您可以在 **付款** 中看到产品的交易信息， 组织、 用户、 购买时间、 产品名称等

发票

您可以输入编辑界面来开具发票

填写发票信息。发票类型是 `个人` 和 `机构`。

Message :
Person name :
Person ID card :
Person Email :
Person phone :
Invoice type :
Invoice title :
Invoice tax ID :
Invoice remark :
Invoice URL :
Invoice actions :
Issue Invoice
Return to Website

最后，点击“**发出发票**”按钮。

用户

概述

管理Casdoor的用户

角色

用户角色

权限

用户权限

概述

User properties

作为一个认证平台，Casdoor 能够管理用户。 每个用户都有这些属性：

- `Owner` 用户的所有者组织
- `Name` 用户名，唯一的
- `CreatedTime 创建时间`
- `UpdatedTime`
- `Id` 每个用户的 Id 都是唯一的。
- `Type`
- `Password`
- `PasswordSalt`
- `DisplayName` 显示在界面中的名称
- `FirstName`
- `LastName`
- `Avatar` 用户头像的链接
- `PermanentAvatar`
- `Email`
- `Phone`
- `Location`
- `Address`
- `Affiliation`
- `Title`

- `IdCardType`
- `IdCard`
- `Homepage`
- `Bio`
- `Tag`
- `Region`
- `Language`
- `Gender`
- `Birthday`
- `Education`
- `Score`
- `Karma`
- `Ranking`
- `IsDefaultAvatar`
- `IsOnline`
- `IsAdmin` 用户是否是他的组织的管理员
- `IsGlobalAdmin` 用户是否有权管理 Casdoor
- `IsForbidden`
- `IsDeleted`
- `SignupApplication`
- `Hash`
- `PreHash`
- `CreatedIp`
- `LastSigninTime`
- `LastSigninIp`
- `Roles` 用户角色数组

- `Permissions` 用户权限数组

平台唯一ID:

- `Github`
- `Google`
- `QQ`
- `微信`
- `Facebook`
- `钉钉`
- `微博`
- `Gitee`
- `LinkedIn`
- `Wecom`
- `Lark`
- `Gitlab`
- `Adfs`
- `Baidu`
- `Casdoor`
- `Infoflow`
- `Apple`
- `AzureAD`
- `Slack`
- `Steam`
- `Ldap`
- `Properties` 这是一个字符串 → 字符串地图，存储所有其他属性。

从CSV文件导入用户

您可以通过上传用户信息的 XLSX 文件来添加新用户或更新现有的 Casdoor 用户。

在管理控制台中，转到用户并点击 **上传(.xlsx)** 按钮。

选择您的 XLSX 文件并点击 Open，用户将被导入。

我们提供了 模板XLSX 文件 `user_test.xlsx` 在 `xlsx` 文件夹中。 该模板包括5个用户测试和某些必需的用户属性的信头。

Home Organizations Users Roles Permissions Providers Applications Users uploaded successfully, refreshing the page Syncers Certs Swagger Admin

Users Add Upload (.xlsx)

user_test.xlsx

Organization	Application	Name	Created time	Display name	Avatar	Email	Phone	Affiliation	Country/Region	Tag	Action
built-in	app-built-in	tesla	2022-03-20 20:49:03	Nikola Tesla		9v73hn@example.com	40738134827	Example Inc.	United States of America	scier	Edit Delete
built-in	app-built-in	gauss	2022-03-20 20:48:33	Carl Friedrich Gauss		vqdsan@example.com	98621482844	Example Inc.	Germany	matl	Edit Delete
built-in	app-built-in	galieleo	2022-03-20 20:47:58	Galileo Galilei		8p4f38@example.com	22596937332	Example Inc.	Italy	scier	Edit Delete
built-in	app-built-in	euler	2022-03-20 20:47:08	Leonhard Euler		3dzw4j@example.com	74409642681	Example Inc.	Switzerland	matl	Edit Delete
built-in	app-built-in	einstein	2022-03-20 20:46:29	Albert Einstein		z6mive@example.com	60062541396	Example Inc.	Germany	scier	Edit Delete
built-in	app-built-in	admin	2022-03-20 20:07:23	Admin		admin@example.com	12345678910	Example Inc.		staff	Edit Delete

6 in total 1 10 / page

Made with ♥ by **Casdoor**

角色

每个用户可能有多个角色。 您可以在用户的个人资料上看到用户的角色。

角色属性

每个角色都具有这些属性:

- 所有者
- 名称
- 创建时间
- 显示名称
- 已启用
- 用户 此角色子用户的数组
- 角色 此角色子角色的数组

权限

每个用户可能有多个权限。 您可以在用户个人资料上看到用户的权限。

权限属性

权限拥有这些属性

- 所有者
- 名称
- 创建时间
- 显示名称
- 已启用
- 模型
- 用户 此角色子用户的数组

- `角色` 此角色子角色的数组
- `资源类型`
- `资源` 资源数组
- `操作` 操作数组
- `效果`

同步器

概述

Casdoor同步用户

数据库

使用 Databse 同步器同步数据库

Keycloak

使用 Keycloak Syncer 同步 Keycloak

概述

作为一个认证平台，Casdoor 可以轻松地操纵存储在数据库中的用户。

同步器

Casdoor 在 **user** 表中存储用户。 当您计划使用 Casdoor作为认证平台时，请不要担心您的应用程序 **用户** 数据迁移到Casdoor。 Casdoor提供 **同步器** 来帮助您快速同步用户数据到 Casdoor。

指定您想要同步到Casdoor的数据库和用户表。 同步器将在指定的间隔后同步数据。 欲了解详情，请参阅 数据库同步器。

同步哈希值

Casdoor使用哈希来确定如何更新用户。 Casdoor 将计算表中每个用户的散列值。 它是通过用户信息比如用户的密码或手机号码来生成的。

如果特定 `Id` 的用户计算的散列值与原始值相比有所改变，Casdoor会确认哪个用户表已更新。 然后，数据库将更新旧信息，实现Casdoor用户表和原始 用户表之间的 **双向同步**。

数据库

数据库同步器

我们作为演示创建的用户表从 模板 XLSX 文件 导入。

owner	name	created_time	updated_time	id	type	password	password_salt	display_name	first_name	last_name	avatar	permanent_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@

点击 Syncers 标签页并创建一个新的同步器。 填写下面所需的所有信息并保存。

提示

> 一般而言，您至少需要填写Casdoor列中的 `ID` 和 `Name` 以及其他重要信息，如 `createdTime`, `Password`, `DisplayName`.

The following are the required fields.

- `Organization`: The organization that the user will import
- `Name`: The syncer name
- `Type`: Select database
- `Host`: The original database host
- `Port`: The original database port
- `User`: The original database username
- `Password`: The original database password
- `Database type`: All Xorm supported databases, like: MySQL, PostgreSQL, SQL Server, Oracle, Sqlite
- `Database`: The original database name
- `Table`: The original user table name
- `表格列`
- `Column name`: The original user column name
- `Column type`: The original user column type
- `Casdoor Column`: The casdoor user column name

Optional fields

- `Is hashed`: Whether to calculate hash value. When enable "**Is hashed**", if the field of user in origin table updated, the syncer will sync this user. Disable "Is hashed", meaning if only the field update, the syncer need not sync the user. In short, the user does not synchronize until the fields involved in the hash calculation(enable "Is hashed") are updated.
- `Avatar base URL`: When sync users, if **Avatar base URL** is not empty and

origin **user.avatar** not hasPrefix "http", new **user.avatar** will be replaced by **Avatar base URL + user.avatar**.

- `Affiliation table`: It is used to sync the affiliation of user from this table in database. Because the affiliation may be code of int type in "Affiliation table", so we need to map the int to a string. See getAffiliationMap() . Because Casdoor has some redundant fields to borrow, here we use `score` to map the int code to a string name.

然后您可以打开 **启用** 按钮并保存，同步器将开始工作。

Keycloak

Keycloak 同步器

Keycloak除了能自动配置`Table` 和 `Table columns`外，其他与 数据库同步器 基本相同。

此外，Keycloak 同步器将会查询 `credential` 表, `keycloak_group` 表 和 `user_group_membership` 表，因为Keycloak 中的用户信息储存在多个表中。

令牌

概述

Casdoor令牌简介

概述

Casdoor 基于 OAuth。 Tokens 是用户的 OAuth 的 token。

- `Owner`
- `Name`
- `CreatedTime`
- `Application`
- `Organization`
- `User`
- `Code`
- `AccessToken`
- `ExpireIn` 令牌将在数小时后过期
- `Scope` 授权范围
- `TokenType` 例如 输入 `Bear`

There are two options to generate a JWT Token after logging into the application:

- `JWT`
- `JWT-Empty`

The JWT option will create a token with all `User` fields. The `JWT-Empty` will create a token with all non-empty values for the user.

Webhooks

概述

在Casdoor添加webhooks

概述

概述

事件系统允许您构建集成，订阅 Casdoor 上的某些事件。 当其中一个事件被触发时，我们将向配置的 URL 发送一个以 json 为主体的 POST 请求。 应用程序解析 json 并执行 hook。 事件包括注册、登录、注销、更新用户，这些用户存储在记录的操作字段中。 事件系统可以用来更新用户的外部问题。

部署

Nginx

使用Nginx 来反转代理您的后端Go 程序，快速启动Casdoor 服务

k8s

在 k8s 中部署 Casdoor

Nginx

虽然Casdoor 是一个前后端分离的架构，但在生产环境中，后端程序仍然为前端文件提供静态文件服务。 因此， 您可以使用反向代理软件，如 Nginx 来代理 Casdoor 域的所有流量，并将其重定向到后端的端口。

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

1. 构建前端静态文件

现在假设您已经下载了 Casdoor 并完成了必要的配置。 如果没有，请回到 **开始** 部分。

您只需要构建静态文件，例如：

Yarn npm

```
yarn install && yarn run build
```

```
npm install && npm run build
```

2. 运行后端程序

```
go run main.go
```

或者先构建:

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

3. 配置和运行 Nginx

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

添加服务器:

```
server {
    listen 80;
    server_name YOUR_DOMAIN_NAME;
    location / {
        proxy_set_header    Host            $http_host;
        proxy_set_header    X-Real-IP       $remote_addr;
        proxy_set_header    X-Forwarded-For
$proxy_add_x_forwarded_for;
        proxy_redirect      off;
        proxy_pass http://127.0.0.1:8000;
    }
}
```

然后重启您的 nginx 进程，运行:

```
nginx -s reload
```

4. 测试

在您最喜欢的浏览器中访问 `http://YOUR_DOMAIN_NAME`

k8s

在 k8s 中部署 Casdoor

我们已经给出了一个将 Casdoor 部署到 k8s 中的基本示例。 在 casdoor 的根目录下，有一个名为"k8s.yaml"的文件，里面包含了在 k8s 中部署 casdoor 时使用的示例最低配置、一个部署和一个服务。

首先，确保你已经修改了conf/app.conf，使 casdoor 能够成功连接数据库，并且数据库正在运行。 其次，确保 k8s 能够拉取必要的镜像。

运行

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

很快你就可以通过命令 `kubectl get pods` 看到结果

k8s.yaml 的内容如下

```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
  labels:
    app: casdoor
```

该文件只是一个示例。 比如您可以选择使用 default 以外的命名空间，使用服务类型而不是 nodeport 来暴露 casdoor，或者在 k8s 中使用 use config map 来挂载配置文件，这是 k8s 中比较推荐的方式。

LDAP

概述

Casdoor 与ldap服务器合作

Config and Sync LDAP Users

Casdoor LDAP 配置

LDAP 服务器

在Casdoor中连接LDAP服务器

概述

支持当前ldap服务器已被引入到Casdoor。 Casdoor 能够同步用户从ldap服务器到Casdoor使用它们作为用户帐户登录。 并使用 ldap服务器验证它们。 Casdoor还支持设置cron作业，以定期自动同步用户。

Casdoor-Ldap 同步机制详情

How Casdoor cooperates with a ldap server is described as follows:

1. Synchronization: Casdoor can connect to ldap server fetch users' information, reads all users' information (include `uidNumber`,`uid`, `cn`, `gidNumber`, `mail`, `email`, `emailAddress`, `telephoneNumber`, `mobile`, `mobileTelephoneNumber`, `registeredAddress`, `postalAddress`), and creates Casdoor accounts for each user in ldap, and stores the accounts in database.

2. Authentication: As we have seen, Casdoor doesn't fetch ldap users' passwords to Casdoor. Thus, when the account which is synchronized from ldap server tries to log in through Casdoor, instead of checking password stored in casdoor's database, Casdoor connects to ldap server and verifies whether the user's password is correct.

3. User distinguished: Casdoor uses `uid` to exclusively identify a user, thus please make sure every ldap user has a different uid.

Once a user is synchronized into Casdoor,that user's information is detached with the ldap user, which means, if you modify the user's information in Casdoor, the user's information in ldap won't be modified and vice versa (except ldap user's password, we rely on it to authenticate a user)

Config and Sync LDAP Users

Ldap 配置属于一个组织，该组织的用户将被同步。

您应该使用全局管理员用户来修改配置。 您需要在“组织”页面中输入LDAP用户同步的以下信息。

Config to connect LDAP server

Server Name

A friendly name is used by managers to identify different servers.

> 例如： `LDAP 服务器示例`

Server Host

LDAP server's host or IP address.

> 例如： `example.com`

Server Port

LDAP server's ports, only allow numbers.

> 例如: `389`

Base DN

Casdoor uses Sub search mode by default when searching in LDAP. Base DN is the basic distinguished name of the search. Casdoor will return all users under the current base DN.

The admin account configured in casdoor should have at least read-only permissions at base DN.

> 例如: `ou=Example,dc=example,dc=com`

Search filter

Casdoor uses search filter to query ldap users.

> e.g: `(objectClass=posixAccount)`

Filter fields

Filter fields are the identifier of the user in LDAP server, When you log in to Casdoor as an LDAP user. Casdoor regards the entered login username as the `uid` of LDAP user. You can also config other filed, such as `mail`, `mobile`.

Admin

An account that can log in to the specified LDAP server.

Login with DN or ID depends on the LDAP server settings you want to connect.

> e.g: `cn=manager,dc=example,dc=com`

Admin Password

Password of LDAP server Admin account.

Auto Sync

Set `0` to disable auto sync, other value means **Sync every few minutes**.

Sync users

The sync table displays all users get from the LDAP server in the specific `ou`. If

the users have been synced, the checkbox will be disabled. You can check the box to select users, then sync the selected users from the LDAP server.

> **注意事项**
>
> If the `uid` of the user in LDAP server is same as the `name` of a user existed in the organization of Casdoor, Casdoor will create a new user that the `name` include the `uid` and a random string. But the user may can not to login, because the name of the new synced user not exist in LDAP server. So try to avoid that.

LDAP 服务器

许多系统，例如 `Nexus` 支持 `ldap` 身份验证。 在Casdoor中，同样实现了一个简单的LDAP服务器，支持bind与search操作。

下面来介绍如何连接Casdoor中的LDAP服务器并实现简单的登录认证。

LDAP 服务器端口号

默认情况下，LDAP服务器监听端口 `389`， 您可以通过更改 conf/app中的 `ldapServerPort` 来更改默认端口号。

工作原理

类似于Casdoor的ldap客户端，ldap服务器中的用户都是 `poxisAccount` 的子类。

当服务器收到符合LDAP协议传输的一组数据时， 它将解析 `cn` and `ou`, 其中 `cn` 表示用户名, `ou` 表示组织名称。 `dc` 是什么并不重要。

如果是`bind`操作，服务器将使用 Casdoor 验证用户名和密码并给予用户权限。

如果它是一个search操作， 服务器将根据bind赋予客户端的权限，检查search是否合法并返回相应响应

> **信息**
>
> 我们只支持 **`Simple Authentication`**。

Bind

在Catdoor ldapserver中，我们识别类似于这样的 `DN`： `cn=admin,ou=buildt-in,dc=example,dc=com`

所以请将管理员用户的 `DN` 设置为上面的形式。 然后您可以使用此`DN` 绑定到 ldap服务器，使用Casdoor用户的密码登录以进行验证。 如果服务器验证通过，用户将被授予Casdoor中的权限。

Search

bind操作成功完成后，您可以执行search操作。 search与bind在细节上有一些区别。

- 如果您想要搜索特定的用户，比如 `built-in` 组织下的 `Alice`， 您应该使用这样的`DN`：`ou=build-in,dc=example c=com`, 并在过滤器字段中添加 `cn=Alice`
- 如果您想要在某个组织下搜索所有用户，如 `built-in`下的所有用户， 您应该使用这样的`DN`：`ou=built-in,dc=example,dc=com`, 并在过滤器字段中添加 `cn=*`
- 如果您想要搜索所有组织的所有用户(前提是用户有足够的权限)， 您应该使用这样的`DN`：`ou=*,dc=example ,dc=com`, 并在过滤器字段中添加 `cn=*`。

集成

Go

6 个项目

Java

14 个项目

JavaScript

1 个项目

Lua

1 个项目

🗃️ PHP

3 个项目

🗃️ Ruby

1 个项目

Go

BookStack

在 BookStack 中使用 Casdoor 进行身份验证

Bytebase

Using OAuth2 to connect various applications, like 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

配置 Casdoor 作为身份提供者与 MinIO 支持

BookStack

在 BookStack 中使用 Casdoor 进行身份验证

BookStack 是一个开源书和文档共享网站 以及使用 Go 语言开发的开源应用程序，帮助您更好地实现文档阅读管理。

Bookstack-casdoor 已经与 castor集成，您现在可以快速开始一个简单的配置。

步骤1. 创建一个Casdoor应用程序

转到您的 Casdoor 并添加您的新应用程序 BookStack。 下面是在Casdoor中创建BookStack应用程序的**示例**。

请记住 `名称`, `组织`, `客户端 ID`, 和 `客户密钥`。 您将在下一步中使用它们。

步骤2. 配置Cassdoor 登录

现在，请移动到BookStack。 查找文件： `oauth.conf.example`。

将 `oauth.conf.example` 重命名为 `oauth.conf` 和 **修改配置** 默认内容为：

```
[oauth]
```

步骤3. 在Casdoor中填写 `重定向Url`

最后一步，请回到您添加 BookStack **应用程序**的页面， 填写 `重定向URL` 请确保 `重定向URL` 和 `oauth.conf` 文件中的 `重定向URL` 相同。

现在你已经完成了Casdoor的所有配置！

一旦BookStack成功部署，您可以使用Cassdoor 返回BookStack和体验以进行登录认证。

Bytebase

Casdoor can use OAuth2 to connect various applications. Here we will use Bytebase as an example to show you how to use OAuth2 to connect to your applications.

The following are some of the names in the configuration:

`CASDOOR_HOSTNAME`: Domain name or IP where Casdoor server is deployed.

`Bytebase_HOSTNAME`: Domain name or IP where Bytebase is deployed.

Step1. Deploy Casdoor and Bytebase

Firstly, the Casdoor and Bytebase should be deployed.

After a successful deployment, you need to ensure:

1. Casdoor can be logged in and used normally.
2. You can set CASDOOR_HOSTNAME = `http://localhost:8000`. When deploy Casdoor in `prod` mode. See production mode.

Step2. Configure Casdoor application

1. Create or use an existing Casdoor application.
2. Find a redirect url: `<CASDOOR_HOSTNAME>/oauth/callback`
3. Add your redirect url to casdoor application:

Not surprisingly, you can get two values on the application settings page: `Client ID` and `Client secret` like the picture above, we will use them in the next step.

Open your favorite browser and visit: **http://`CASDOOR_HOSTNAME`/.well-known/openid-configuration**, you will see the OIDC configure of Casdoor.

Step3. Configure Bytebase

1. You should find sso and use OAuth 2.0

2. You should config this app

3. You can find Client Id and Client Secret in Casdoor application page.

- `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`

Log out of Bytebase, and test SSO.

ELK

Casdoor/elk-auth-casdoor概览

ELK (Elasticsearch、Logstash 和 Kibana) 的缺点是，这些产品原来没有认证机制。 只要拥有kibana或ESurl，每个人都可以访问kibana dashboard。 后来ELK 集成了一个嵌入式认证系统“Xpack”，其所有高级函数 **都不是免费的** (例如 Oauth, OIDC, LDAP, SAML), 而且只有纯粹的身份验证(设置一套账户和密码) 是免费的，这相当不方便。 我们不能为每个人提供一个独特的帐户。

因此，我们已经开发了一个基于 Cassdoor **免费的 elk 认证解决方案**。 **开源和维护中，支持大量高级功能**。 Casdoor是一个基于Oauth2的中心化身份验证/单点登录平台。 /OIDC, casdoor /elk-auth-casdoor 实际上是一种逆向代理， 它旨在拦截所有 http 数据流到elk/kibana ，并指导尚未登录的用户登录。 只要用户已登录，此逆向代理将完全透明。

如果此用户未成功通过验证，请求将会临时缓存，用户将被重定向到Casto登录页面。 在用户成功登录到casdoor后，缓存请求将会还原并发送到 kibana。 因此，如果一个POST请求（或GET以外的其他请求）被拦截，也是可以的，用户不需要重新填写表格和重新发送请求。 逆向代理将为你记住它。

Castor/elk-auth-casdoor 版本库的位置 https://github.com/casdoor/elk-auth-casdoor

如何使用?

0. 已安装golang环境

1. 转到 casdoor/elk-auth-casdoor 并获取代码

2. 将您的代理注册为Casdoor应用程序。

3. 修改配置

配置文件位于"conf/app.conf"中。 这是一个示例，您应该根据您的实际需求自行更改。

```
appname = .
# port on which the reverse proxy shall be run
httpport = 8080
runmode = dev
#EDIT IT IF NECESSARY. The url of this reverse proxy
pluginEndpoint="http://localhost:8080"
#EDIT IT IF NECESSARY. The url of the kibana
targetEndpoint="http://localhost:5601"
#EDIT IT. The url of casdoor
casdoorEndpoint="http://localhost:8000"
#EDIT IT. The clientID of your reverse proxy in casdoor
clientID=ceb6eb261ab20174548d
#EDIT IT. The clientSecret of your reverse proxy in casdoor
clientSecret=af928f0ef1abc1b1195ca58e0e609e9001e134f4
#EDIT IT. The application name of your reverse proxy in casdoor
appName=ELKProxy
#EDIT IT. The organization to which your reverse proxy belongs in
casdoor
organization=built-in
```

4. 访问 http://localhost:8080 (在上文示例中) 并按照重定向指南登录，您将会看到kibana受casdoor的保护和认证。

5. 如果一切运行良好，请不要忘记通过配置您的防火墙，阻止原来的 kibana 端口访问外部， 这样其他人只能通过这种逆向代理访问kibana。

Gitea

在 Gitea 中使用 Casdoor 进行身份验证

Gitea 是一个社区管理的轻量代码托管解决方案，写入Go。 采用MIT开源协议

Gitea支持第三方身份验证，包括Oauth，这样就可以使用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 进行认证。 因此，用户在Grafana上登录变得非常容易。 只有几个步骤和简单的配置就能做到这一点。

这是一个使用 Grafana 的 Cassdoor 进行身份验证的教程。 在您继续之前，请确保您已安装 grafana 并正在运行。

Step 1 Create an app for Grafana in Casdoor

这是一个在 Casdoor 中创建应用程序的示例

请复制 client secret 和 client ID以便下一步操作 。

Please add the callback url of Grafana. By default, Grafana's oauth callback is `/login/generic_oauth`. 所以请正确地拼接这个 url 。

Step 2: Modify the configuration of Grafana

By default the configuration file for oauth locates at `conf/defaults.ini` in the workdir of Grafana.

请找到 `auth.generic_oauth` 并修改以下字段:

```
[auth.generic_oauth]
name = Casdoor
```

About HTTPS

If you don't want HTTPS enabled for casdoor or if you deploy grafana without HTTPS enabled, please also set `tls_skip_verify_insecure = true`

About redirectURI after Sign In With Casdoor

If the redirect uri is not right after Sign In with Casdoor in Grafana, you may want to configure root_url

```
[server]
http_port = 3000
# The public facing domain name used to access grafana from a
browser
domain = <your ip here>
# The full public facing url
root_url = %(protocol)s://%(domain)s:%(http_port)s/
```

related links:

1. Grafana doc
2. Grafana defaults.ini

About Role Mappping:

You may want to configure role_attribute_path to map your user's role to Grafana via role_attribute_path

```
[auth.generic_oauth]
role_attribute_path = contains(roles[*].name, 'admin') && 'Admin'
```

the JMESPath expression after role_attribute_path is very important here. read grafana doc please

第3步：查看它是否正常运作。

Shutdown grafana and restart it.

Go to see the login page, you are supposed to see something like this

MinIO

MinIO 支持使用OpenID Connect (OIDC) 兼容的程序提供服务进行外部身份管理。 此文档介绍了通过配置Casdoor为身份提供者来支持MinIO。

第 1 步： 部署Casdoor & MinIO

第一，应该先部署Casdoor。

您可以参考 服务器安装 的 Casdoor 官方文档。

在成功部署后，您需要确保：

- Casdoor服务器已成功运行在 http://localhost:8000 上了。
- 打开您最喜欢的浏览器并访问 http://localhost:7001，您将看到Casdoor的登录页面。
- 输入 `admin` 和 `123` 测试登录功能正常工作。

然后您可以通过以下步骤在自己的应用程序中快速实现基于Casdoor的登录页面。

您可以参考 这里 来部署您的 MinIO 服务器， 以及这里 对于名为 `mc 的 MinIO 客户端`。

第 2 步： 配置Casdoor应用

1. 创建或使用现有的 Casdoor 应用程序。
2. 添加您的redirect url

3. 添加您想要的提供商并补充其他设置。

这时， 您可以在应用程序设置页面获得两个值： `Client ID` 和 `Client secret`， 就像上图一样。 我们将在下一步骤中使用它们。

打开您最喜欢的浏览器并访问： http://`CASDOOR_HOSTNAME`/.well known / openid-configur, 您将看到Casdoor配置的OIDC。

4. 这个步骤对于MinIO是必要的。 由于MinIO 需要在 JWT 中使用一个索赔属性来执行其政策，因此您也应该在Casdoor中配置它。 目前，Casdoor使用 `tag` 作为配置 MinIO 策略的一个环境。

您可以在这里找到所有支持的策略 。

第 3 步： 配置 MinIO

您可以通过以下命令启动 MinIO 服务器：

```
导出MINIO_ROOT_USER=minio
```

您可以使用参数 `--console-address` 来配置地址和端口。

然后您可以在 MinIO 客户端 `mc` 中添加一个服务别名。

```
mc 别名设置了myminio <You console address> minio minio123
```

现在，您可以配置 MinIO 的 OpenID 连接。 对于Casdoor，命令如下：

```
mc admin config set myminio identity_openid
config_url="http://CASDOOR_HOSTNAME/.well-known/openid-
configuration" client_id=<client id> client_secret=<client secret>
claim_name="tag"
```

更详细的参数，您可以参考 官方文档。

一旦设置成功，请重启 MinIO 实例。

```
mc 管理服务重启myminio
```

第 4 步： 试用一下demo！

现在，您可以在浏览器上打开您的 MinIO 控制台，然后点击 `Login with SSO`。

您将重新转至登录页面 您将被重定向到Casdoor用户登录页面。登录成功后，您将被重定向到 MinIO 页面并自动登录， 您应该现在看到他们可以访问的buckets和objects。

> **注意事项**
>
> 如果您在不同的端口部署Casdoor前端和后端， 您被重定向的登录页面将是后端端口，它将显示 `404 找不到`。 您可以修改端口到前端。 然后您可以成功访问Casdooor登录页面。

Java

Spring Boot

在Spring Boot项目中使用 Casdoor

Spring Cloud

在Spring Cloud中使用 Casdoor

Spring Cloud Gateway

在Spring Cloud Gateway中使用 Casdoor

Spring 安全

2 个项目

Jenkins 插件

使用 Casdoor 插件来保证您的 Jenkins 安全

Jenkins OIDC

使用 OIDC 协议作为IDP 连接各种应用程序，如Jenkins

Jira

使用 OIDC 协议作为IDP 连接各种应用程序，如Jira

Confluence

Using OIDC protocol as IDP to connect various applications, like Confluence

RuoYi

在 RuoYi-Cloud上使用 Casdoor

Pulsar-manager

在 Pulsar-manager 中使用 Cassdoor

ShenYu

在 ShenYu 中使用Cassoor

ShardingSphere

在ShardingSphere中使用 Casdoor

Apache IoTDB

Using Casdoor Apache IoTDB

FireZone

Using OIDC protocol as IDP to connect various applications, like FireZone

Spring Boot

casdoor-spring-boot-example 是关于如何在 SpringBoot 项目中使用 casdoor-spring-boot-starter 的示例。 我们将向您展示以下步骤。

第 1 步： 部署Casdoor

第一，应部先部署Casdoor。

您可以参考 服务安装 的 Casdoor 官方文档。 请在 **生产模式** 中部署您的 castor 实例。

在成功部署后，您需要确保：

- 打开您最喜欢的浏览器并访问 http://localhost:8000，您将看到Casdoor的登录页面。
- 输入 `admin` 和 `123` 测试登录功能正常工作。

然后您可以通过以下步骤在自己的应用程序中快速实现基于 Casdoor 的登录页面。

第2步： 导入 casdoor-spring-boot-starter

您可以使用 maven 或 gradle 导入 casdoor-spring-boot-starter。

Maven Gradle

```
<!-- https://mvnrepository.com/artifact/org.casbin/casdoor-spring-boot-starter -->
<dependency>
    <groupId>org.casbin</groupId>
    <artifactId>casdoor-spring-boot-starter</artifactId>
    <version>1.x.y</version>
</dependency>
```

```
// https://mvnrepository.com/artifact/org.casbin/casdoor-spring-boot-starter
implementation group: 'org.casbin', name: 'casdoor-spring-boot-starter', version: '1.x.y'
```

第 3 步： 初始化配置

初始化需要6个参数，它们都是字符串类型。 | 名称(按顺序排列) | 必选项 | 描述 | | ---------- | --- | ---------- | | endpoint | 是 | Casdoor 服务URL, 例如 `http://localhost:8000` | | clientId | 是 | Application.client_id | | clientSecret | 是 | Application.client_secret | | certificate | Yes | Application.certificate | | organizationName | 是 | Application.organization | | applicationName | 否 | Application.name | 您可以使用 Java properties 或 YAML 文件来初始化，如下所示。

Properties YML

```
casdoor.endpoint = http://localhost:8000
casdoor.clientId = <client-id>
casdoor.clientSecret = <client-secret>
casdoor.certificate = <certificate>
casdoor.organizationName = built-in
```

```yaml
casdoor:
  endpoint: http://localhost:8000
  client-id: <client-id>
  client-secret: <client-secret>
  certificate: <certificate>
  organization-name: built-in
  application-name: app-built-in
```

> **注意事项**
>
> 您应该用您自己的 Casdoor 实例替换配置，特别是 `clientId`, `clientSecret` 和 `jwtPublicKey`。

第 4 步： 重定向到登录页面

当您需要访问您的应用程序的身份验证时，您可以发送目标 url 并重定向到 Casdoor 提供的登录页面。 请确保您已提前在应用配置中添加回调 url(例如 http://localhost:8080/login)。

```java
@Resource
private CasdoorAuthService casdoorAuthService;

@RequestMapping("toLogin")
public String toLogin() {
    return "redirect:" +
casdoorAuthService.getSigninUrl("http://localhost:8080/login");
}
```

第 5 步： 获取令牌并解析

在 Casdoor 验证通过后，它将被重定向到您的应用程序，并带有 code 和状态。

您可以获取 code 并调用 `getOAuthToken` 方法，然后解析出 jwt 令牌。

`Casdoor User` 包含了由Casdoor提供的有关用户的基本信息。 您可以使用它作为关键词在您的应用程序中设置会话。

```
@RequestMapping("login")
public String login(String code, String state, HttpServletRequest
request) {
    String token = "";
    CasdoorUser user = null;
    try {
        token = casdoorAuthService.getOAuthToken(code, state);
        user = casdoorAuthService.parseJwtToken(token);
    } catch (CasdoorAuthException e) {
        e.printStackTrace();
    }
    HttpSession session = request.getSession();
    session.setAttribute("casdoorUser", user);
    return "redirect:/";
}
```

Service

这两种情况的示例如下所示:

- CasdoorAuthService
  - `String token = casdoorAuthService.getOAuthToken(code, "app-`

`built-in");`

  - `CasdoorUser casdoorUser = casdoorAuthService.parseJwtToken(token);`
- CasdoorUserService
  - `CasdoorUser casdoorUser = casdoorUserService.getUser("admin");`
  - `CasdoorUser casdoorUser = casdoorUserService.getUserByEmail("admin@example.com");`
  - `CasdoorUser[] casdoorUsers = casdoorUserService.getUsers();`
  - `CasdoorUser[] casdoorUsers = casdoorUserService.getSortedUsers("created_time", 5);`
  - `int count = casdoorUserService.getUserCount("0");`
  - `CasdoorResponse response = casdoorUserService.addUser(user);`
  - `CasdoorResponse response = casdoorUserService.updateUser(user);`
  - `CasdoorResponse response = casdoorUserService.deleteUser(user);`
- CasdoorEmailService
  - `CasdoorResponse response = casdoorEmailService.sendEmail(title, content, sender, receiver);`
- CasdoorSmsService
  - `CasdoorResponse response = casdoorSmsService.sendSms(randomCode(), receiver);`
- CasdoorResourceService
  - `CasdoorResponse response = casdoorResourceService.uploadResource(user, tag, parent, fullFilePath, file);`
  - `CasdoorResponse response = casdoorResourceService.deleteResource(file.getName());`

更多内容

您可以探索以下项目/文件来了解更多关于Java 与Casdoor一体化的信息。

- casdoor-java-sdk
- casdoor-spring-boot-starter
- casdoor-spring-boot-example
- casdoor-spring-security-example
- casdoor-spring-security-react-example
- casdoor-spring-boot-shiro-example

Spring Cloud

在Spring Cloud微型服务体系下，一般由网关进行认证。 请参阅 casdoor-springcloud-gateway-example。

如果您想要在单个服务中使用Casdoor，您可以参考 casdoor-spring-boot示例。

无论是网关还是在某个服务负责认证，都需要使用 casdoor-spring-boot-starter

更多内容

您可以查阅以下项目/文件来了解更多关于Java中集成Casdoor信息。

- casdoor-java-sdk
- casdoor-spring-boot-starter
- casdoor-spring-boot-example
- casdoor-spring-security-example
- casdoor-spring-security-react-example
- casdoor-spring-boot-shiro-example
- casdoor-springcloud-gateway-example

Spring Cloud Gateway

casdoor-springcloud-gateway-example 是如何在 Spring Cloud Gateway 中使用 casdoor-spring-boot-starter 作为一个 OAuth2 插件。 我们将向您展示以下步骤。

第1步 部署Casdoor

首先，应当部署Casdoor。

您可以参考 Server Installation 的 Casdoor 官方文档。 请在 **生产模式** 中部署您的 Casdoor 实例。

在成功部署后，您需要确认：

- 打开您最喜欢的浏览器并访问 **http://localhost:8000**，您将看到Casdoor的登录页面。
- 输入 `admin` 和 `123` 测试登录功能正常工作。

然后您可以通过以下步骤在自己的应用程序中快速实现基于 Casdoor 的登录页面。

第2步: 初始化一个Spring Cloud Gateway 项目

您可以直接使用此示例的代码或结合您自己的业务代码。

我们需要一个网关服务和至少一个业务服务。

在这个示例中， `casdoor-gateway` 作为网关服务， `casdoor-api` 作为业务服务。

第3步：引入依赖

将 `casdoor-spring-boot-starter` 添加到Spring Cloud Gateway项目。

对于Apache Maven：

/casdoor-gateway/pom.xml

```
<!-- https://mvnrepository.com/artifact/org.casbin/casdoor-spring-boot-starter -->
<dependency>
    <groupId>org.casbin</groupId>
    <artifactId>casdoor-spring-boot-starter</artifactId>
    <version>1.x.y</version>
</dependency>
```

对于Gradle：

```
// https://mvnrepository.com/artifact/org.casbin/casdoor-spring-boot-starter
implementation group: 'org.casbin', name: 'casdoor-spring-boot-starter', version: '1.x.y'
```

第4步：配置属性

初始化需要 6 个参数，它们都是字符串类型：

名称(按顺序排列)	是否必须	描述
endpoint	是	Casdoor 服务URL, 例如 `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 properties 或 YAML 文件来初始化，如下所示。

对于 properties：

```
casdoor.endpoint=http://localhost:8000
casdoor.clientId=<client-id>
casdoor.clientSecret=<client-secret>
casdoor.certificate=<certificate>
casdoor.organizationName=built-in
casdoor.applicationName=app-built-in
```

对于yaml：

```
casdoor:
  endpoint: http://localhost:8000
```

此外，还需要配置网关路由。 对于yaml：

```yaml
spring:
  application:
    name: casdoor-gateway
  cloud:
    gateway:
      routes:
        - id: api-route
          uri: http://localhost:9091
          predicates:
            - Path=/api/**
```

第5步：添加CasdoorAuthFilter

将 GlobalFilter 的一个实现类添加到网关中做身份验证，例如此 示例中的 CasdoorAuthFilter。

如果身份验证失败，则返回到401，前端跳转到统一登录界面。

```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 =
```

第6步：使用Casdoor提供的相关Service

现在提供 5 种Service： `CasdoorAuthService`, `CasdoorUserService`, `Casdoore-mailService`, `CasdoorSmsService` 和 `CasdoorResourceService`

您可以按照下面示例在Gateway项目中创建它们。

```
@Resource
private CasdoorAuthService casdoorAuthService;
```

当您需要访问您的应用程序的身份验证时，您可以发送目标 url 并重定向到 Casdoor 提供的登录页面。

请确保您已提前在应用配置中添加回调 url(例如 http://localhost:9090/login)。

```
@RequestMapping("login")
public Mono<String> login() {
    return Mono.just("redirect:" +
casdoorAuthService.getSigninUrl("http://localhost:9090/callback"));
}
```

在 Casdoor 验证通过后，它将被重定向到您的应用程序，并带有 code 和状态。

您可以获取 code 并调用 `getOAuthToken` 方法，然后解析出 jwt 令牌。

`CasdoorUser` 包含了 Casdoor 提供的用户基本信息，您可以将其作为关键字在您的应用程序中设置会话。

```
@RequestMapping("callback")
public Mono<String> callback(String code,String
```

API示例如下：

- CasdoorAuthService
  - `String token = casdoorAuthService.getOAuthToken(code, "app-built-in");`
  - `CasdoorUser casdoorUser = casdoorAuthService.parseJwtToken(token);`
- CasdoorUserService
  - `CasdoorUser casdoorUser = casdoorUserService.getUser("admin");`
  - `CasdoorUser casdoorUser = casdoorUserService.getUserByEmail("admin@example.com");`
  - `CasdoorUser[] casdoorUsers = casdoorUserService.getUsers();`
  - `CasdoorUser[] casdoorUsers = casdoorUserService.getSortedUsers("created_time", 5);`
  - `int count = casdoorUserService.getUserCount("0");`
  - `CasdoorResponse response = casdoorUserService.addUser(user);`
  - `CasdoorResponse response = casdoorUserService.updateUser(user);`
  - `CasdoorResponse response = casdoorUserService.deleteUser(user);`
- CasdoorEmailService
  - `CasdoorResponse response = casdoorEmailService.sendEmail(title, content, sender, receiver);`
- CasdoorSmsService
  - `CasdoorResponse response = casdoorSmsService.sendSms(randomCode(), receiver);`
- CasdoorResourceService
  - `CasdoorResponse response = casdoorResourceService.uploadResource(user, tag, parent,`

`fullFilePath, file);`

- `CasdoorResponse response = casdoorResourceService.deleteResource(file.getName());`

第7步：重新启动项目

启动后，打开您最喜欢的浏览器并访问 http://localhost:9090，然后点击任何可以向 `casdoor-api请求资源的按钮`

将触发网关认证逻辑。 由于您没有登录，您将跳转到登录界面。 点击登录按钮。

随后您可以看到Cassdoor统一的登录平台。

登录成功后，它将跳转到主界面。 然后您可以点击任意资源相关按钮。

更多内容

您可以探索以下项目/文件来了解更多关于Java 与Casdoor一体化的信息。

- casdoor-java-sdk
- casdoor-spring-boot-starter
- casdoor-spring-boot-example
- casdoor-spring-security-example
- casdoor-spring-security-react-example
- casdoor-spring-boot-shiro-example
- casdoor-springcloud-gateway-example

Spring 安全

Spring Security OAuth

这里我们将使用Spring Security 作为示例，向您展示如何将 OIDC 链接到您的应用程序。

Spring Security Filter

基于Spring Security Filter，如何使用OIDC连接您的应用程序。

Spring Security OAuth

Casdoor 可以使用 OIDC 协议作为IDP 连接各种应用程序。 这里我们将使用Spring Security作为示例，向您展示如何使用 OIDC 链接到您的应用程序。

步骤1. 部署Casdoor

首先，部署Casdoor。

您可以参考Casdoor 官方文档 Server Installation 。

成功部署后，您需要确保：

- Casdoor服务已成功运行，能通过http://localhost:8000 访问。
- 打开您最喜欢的浏览器并访问 http://localhost:7001，您将看到Casdoor的登录页面。
- 输入 `admin` 和 `123` 测试登录功能正常工作。

然后您可以通过以下步骤在自己的应用程序中快速实现基于 Casdoor 的登录页面。

第2步： 配置Casdoor应用程序

1. 创建或使用现有的 Casdoor 应用程序。
2. 添加您的重定向url (您可以在下一节中看到更多关于如何获取重定向url的细节)

3. 添加您想要的提供商并补充其他设置。

不出意外的话，您会在应用程序设置页面看到：`Client ID` 和 `Client secret` 就像上面的图片一样。 我们将在下一步中使用它们。 您将在下一步中使用它们。

打开你喜欢的浏览器，访问：http://`CASDOOR_HOSTNAME`/.well-known/openid-configuration，你会看到 Casdoor 的 OIDC 配置。

步骤3. 配置Spring Security

Spring Security 完全支持 OIDC。

您可以自定义Spring Security OAuth2 客户端的设置：

> **注意事项**
>
> 您应该用您自己的 Casdoor 实例替换配置，特别是 `<Client ID>` 等。

application.yml　application.properties

```yaml
spring:
  security:
    oauth2:
```

```
spring.security.oauth2.client.registration.casdoor.client-id=<Client ID>
spring.security.oauth2.client.registration.casdoor.client-secret=<Client Secret>
spring.security.oauth2.client.registration.casdoor.scope=<Scope>
spring.security.oauth2.client.registration.casdoor.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.casdoor.redirect-uri=<Redirect URL>

spring.security.oauth2.client.provider.casdoor.authorization-uri=http://CASDOOR_HOSTNAME:7001/login/oauth/authorize
spring.security.oauth2.client.provider.casdoor.token-uri=http://CASDOOR_HOSTNAME:8000/api/login/oauth/access_token
spring.security.oauth2.client.provider.casdoor.user-info-uri=http://CASDOOR_HOSTNAME:8000/api/get-account
spring.security.oauth2.client.provider.casdoor.user-name-attribute=name
```

> ⚠ **注意事项**
>
> 对于Spring Security的默认情况， `<Redirect URL>` 应该等于 `http://<You Spirng Boot Application Endpoint>/<Servlet Prefix if it is configured>/login/oauth2/code/code`。 例如，对于下面的演示来说，重定向URL应该是 `http://localhost:8080/login/oauth2/code/code/custom`。 您也应该在 `casdoor` 应用程序中配置它。

您也可以通过 `ClientRegistration` 在您的代码中自定义设置。 您可以在这里找到映射

步骤4. 从一个 Demo 开始

1. 我们可以创建 Spring Boot 应用程序。
2. 我们可以添加一个配置，保护所有端点，除了 `/` 和 `/login**` 用来给用户登录。

```
@EnableWebSecurity
public class UiSecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests().antMatchers("/",
"/login**").permitAll().anyRequest().authenticated().and()
                .oauth2Login();

    }
}
```

3. 我们可以添加一个简单的页面供用户登录。

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Spring OAuth Client Thymeleaf - 1</title>
<link rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/
bootstrap.min.css" />
</head>
<body>
    <nav
        class="navbar navbar-expand-lg navbar-light bg-light shadow-
sm p-3 mb-5">
        <a class="navbar-brand" th:href="@{/foos/}">Spring OAuth 客户
端
            Thymeleaf - 1</a>
    </nav>
    <div class="container">
        <label>欢迎 ! </label> <br /> <a th:href="@{/foos/}"
            class="btn btn-primary">登陆</a>
    </div>
</body>
</html>
```

当用户点击 `登录` 按钮时，他将会被重定向到 `Casdoor`。

4. 接下来，我们可以定义我们受保护的资源。 我们可以导出一个名为`/foos`的端点和一个用于显示的网页。

Data Model

```
public class FooModel {
    private Long id;
    private String name;

    public FooModel(Long id, String name) {
        super();
        this.id = id;
        this.name = name;
    }
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

Controller

```
@Controller
public class FooClientController {
    @GetMapping("/foos")
    public String getFoos(Model model) {
```

Web page

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Spring OAuth Client Thymeleaf - 1</title>
<link rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/
bootstrap.min.css" />
</head>
<body>
    <nav
        class="navbar navbar-expand-lg navbar-light bg-light shadow-
sm p-3 mb-5">
        <a class="navbar-brand" th:href="@{/foos/}">Spring OAuth
Client
            Thymeleaf -1</a>
        <ul class="navbar-nav ml-auto">
            <li class="navbar-text">Hi, <span
sec:authentication="name">preferred_username</span>   
            </li>
        </ul>
    </nav>
    <div class="container">
        <h1>All Foos:</h1>
        <table class="table table-bordered table-striped">
            <thead>
                <tr>
                    <td>ID</td>
                    <td>Name</td>
                </tr>
            </thead>
            <tbody>
                <tr th:if="${foos.empty}">
                    <td colspan="4">No foos</td>
                </tr>
                <tr th:each="foo : ${foos}">
```

> **注意事项**
>
> 所有网页模板都应该放置在 `resources/templates` 下。

步骤5. 试用一下demo！

首先，您可以尝试打开您最喜欢的浏览器并直接访问 `/foos`。 它将自动重定向到 Casdoor 的登录页面。 您可以在这里或从根页面登录。

如果您访问了您的根页面，

点击 `登录` 按钮，页面将重定向到下级登录页面。

登录后，页面将重定向到 `/foos`。

Spring Security Filter

Casdoor 可以使用 OIDC 协议作为IDP 连接各种应用程序。 下面我们将使用spring security中的过滤器来集成casdoor，并向您展示如何使用oidc连接到应用程序。

步骤1. 部署Casdoor

首先，部署Casdoor。

您可以参考Casdoor 官方文档 Server Installation 。

成功部署后，您需要确保：

- Casdoor服务已成功运行，能通过http://localhost:8000 访问。
- 打开您最喜欢的浏览器并访问 http://localhost:7001，您将看到Casdoor的登录页面。
- 输入 `admin` 和 `123` 测试登录功能正常工作。

然后您可以通过以下步骤在自己的应用程序中快速实现基于 Casdoor 的登录页面。

第2步： 配置Casdoor应用程序

1. 创建或使用现有的 Casdoor 应用程序。
2. 添加您的重定向url (您可以在下一节中看到更多关于如何获取重定向url的细节)

3. 在证书编辑页面上，您可以看到您的 `证书`。

4. 添加您想要的提供商并补充其他设置。

不出意外的话，您会在应用程序设置页面看到：`应用程序名称`，`组织名称`，`重定向URL`，`客户端 ID`，`客户密钥`，`认证`. 如上所述，我们将在下一步中使用它们。

打开你喜欢的浏览器，访问：http://`CASDOOR_HOSTNAME`/.well-known/openid-configuration，你会看到 Casdoor 的 OIDC 配置。

步骤3. 配置Spring Security

您可以自定义spring security filter 的设置来处理标记：

> ⚠ **注意事项**
>
> 您应该用您自己的 Casdoor 实例替换配置，特别是 `<Client ID>` 等。

```
服务器:
  port: 8080
casdoor:
  endpoint: http:/CASDOOR_HOSTNAME: 8000
  client-id: <Client ID>
```

> **注意事项**
>
> 对于前端应用程序来说，`<FRONTEND_HOSTNAME>` 的默认值是 `localhost:3000`。
>
> 例如，对于下面的演示来说，重定向URL应该是 `http://localhost:3000/callback`。
>
> 您也应该在 `casdoor` 应用程序中配置它。

步骤4. 配置前端

您需要安装 `casdoor-js-sdk` 并配置 `SDK`。

1. 安装 `casdoor-js-sdk`。

```
npm i casdoor-js-sdk
# or
yarn add casdoor-js-sdk
```

2. 设置 `SDK`。

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

// Serverurl is the URL where spring security is deployed
export const ServerUrl = "http://BACKEND_HOSTNAME:8080";

const sdkConfig = {
  serverUrl: "http://CASDOOR_HOSTNAME:8000",
  clientId: "<your client id>",
  appName: "<your application name>",
  organizationName: "<your organization name>",
  redirectPath: "/callback",
```

步骤5. 从一个 Demo 开始

1. 我们可以创建 Spring Boot 应用程序。

2. 我们可以添加一些配置来处理 JWT。

```
@EnableWebSecurity
public class SecurityConfig {

    private final JwtTokenFilter jwtTokenFilter;

    public SecurityConfig(JwtTokenFilter jwtTokenFilter) {
        this.jwtTokenFilter = jwtTokenFilter;
    }

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity
http) throws Exception {
        // enable CORS and disable CSRF
        http = http.cors(corsConfig -> corsConfig
                .configurationSource(configurationSource())
        ).csrf().disable();

        // set session management to stateless
        http = http
                .sessionManagement()

.sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                .and();

        // set permissions on endpoints
        http.authorizeHttpRequests(authorize -> authorize
                .mvcMatchers("/api/redirect-url", "/api/
signin").permitAll()
                .mvcMatchers("/api/**").authenticated()
        );
```

3. 我们可以添加简单的 JWT 过滤器来拦截需要验证标记的请求。

```
@Component
public class JwtTokenFilter extends OncePerRequestFilter {

    private final CasdoorAuthService casdoorAuthService;

    public JwtTokenFilter(CasdoorAuthService casdoorAuthService) {
        this.casdoorAuthService = casdoorAuthService;
    }

    @Override
    protected void doFilterInternal(HttpServletRequest request,
                                    HttpServletResponse response,
                                    FilterChain chain)
            throws ServletException, IOException {
        // get authorization header and validate
        final String header =
request.getHeader(HttpHeaders.AUTHORIZATION);
        if (!StringUtils.hasText(header) ||
!header.startsWith("Bearer ")) {
            chain.doFilter(request, response);
            return;
        }

        // get jwt token and validate
        final String token = header.split(" ")[1].trim();

        // get user identity and set it on the spring security
context
        UserDetails userDetails = null;
        try {
            CasdoorUser casdoorUser =
casdoorAuthService.parseJwtToken(token);
            userDetails = new CustomUserDetails(casdoorUser);
        } catch (CasdoorAuthException exception) {
            logger.error("casdoor auth exception", exception);
            chain.doFilter(request, response);
            return;
```

当用户访问需要认证的接口时， `JwtTokenFilter` 将从请求头 `授权` 并验证它。

4. 接下来，我们需要定义一个`控制器`来处理以下情况：当用户登录到`casdoor`时，它将被重定向到服务器并携带`code`和`state`。 服务器会需要验证用户的身份，并通过这两个参数获得 `token`。

```
@RestController
public class UserController {

    private static final Logger logger =
LoggerFactory.getLogger(UserController.class);

    private final CasdoorAuthService casdoorAuthService;

    // ...

    @PostMapping("/api/signin")
    public Result signin(@RequestParam("code") String code,
@RequestParam("state") String state) {
        try {
            String token = casdoorAuthService.getOAuthToken(code,
state);
            return Result.success(token);
        } catch (CasdoorAuthException exception) {
            logger.error("casdoor auth exception", exception);
            return Result.failure(exception.getMessage());
        }
    }

    // ...

}
```

步骤6. 试用一下demo！

首先，您可以尝试通过浏览器访问前端应用程序。 如果您尚未登录，它将显示登录按钮。

单击登录按钮，您将被重定向到 `级门` 登录页面。

如果您访问了您的根页面，

点击 `登录` 按钮，页面将重定向到下级登录页面。

登录后，页面将重定向到 `/`。

New User - rtsbx4

Logout

Jenkins 插件

Casdoor 为用户登录Jenkins提供了一个插件。 这里我们将向您展示如何使用 Casdoor 插件来保证您的 Jenkins 安全。

以下是配置中的一些专有名词:

`CASDOOR_HOSTNAME`: 私有部署的Casdoor域名或IP。

`JENKINS_HOSTNAME`: 部署Jenkins的域名或IP。

第1步。 部署Casdoor和Jenkins

首先，应该部署 Casdoor 和 Jenkins。

在成功部署后，您需要确保:

1. 将 Jenkins URL(管理 Jenkins → 配置系统 → Jenkins 位置) 设置为 `JENKINS_HOSTNAME`

2. 可以登录并正常使用Casdoor 。
3. 将Casdoor 的 `origin` value (conf/app.conf) 设置为 `CASDOOR_HOSTNAME`。

```
conf > app.conf
8   dbName = casdoor
9   redisEndpoint =
10  defaultStorageProvider =
11  isCloudIntranet = false
12  authState = "casdoor"
13  httpProxy = "127.0.0.1:10808"
14  verificationCodeTimeout = 10
15  initScore = 2000
16  logPostOnly = true
17  origin = "http://10.144.1.2:8000"
                CASDOOR_HOSTNAME
```

第2步： 配置Casdoor应用程序

1. 创建或使用现有的 Casdoor 应用程序。
2. 添加重定向URL： `http://JENKINS_HOSTNAME/securityRealm/finish登录`

3. 添加您想要的提供商并补充其他设置。

不出意外的话，您可以在应用程序设置页面获得两个值： `Client ID` 和 `Client`

`secret` 就像上面的图片一样。 我们将在下一步骤中使用它们。

打开你喜欢的浏览器，访问：http://`CASDOOR_HOSTNAME`/.well-known/openid-configuration，你会看到 Casdoor 的 OIDC 配置。

第3步 配置 Jenkins

现在，您可以从市场安装Casdoor 插件，或者上传它的 `jar` 文件。

安装完成后，前往管理Jenkins → 配置全局安全性。

建议: 备份Jenkins `config.xml` 文件，并在设置错误时使用它来恢复。

1. 在 Security 中，选择“Casdoor Authentication Plugin”。
2. 在Cassdoor Endpoint中，指定上面提到的 `CASDOOR_HOSTNAME`
3. 在客户端ID中，指定上面注明的 `客户端ID`。
4. 在客户端secret中，指定上面注明的 `Client secret`

5. 在 JWT 公钥中，指定用于验证 JWT 令牌的公钥。 您可以通过点击顶部的 `Cert` 在Casdoor找到公钥。 点击 `编辑` 应用程序后，您可以在下面的页面复制您的公钥。

6. 组织名称和应用程序名称是可选的。 您可以指定您的组织和应用程序来验证其他组织和应用程序中的用户。 如果他们为空，插件将使用默认的组织和应用程序。
7. 在授权部分，检查“Logged-in users can do anything”。 禁用"Allow anonymous read access"。
8. 点击 `Ok`

现在，Jenkins会自动将您重定向到Cassdoor 进行认证。

Jenkins OIDC

Casdoor 可以使用 OIDC 协议作为IDP 连接各种应用程序。 这里我们将使用Jenkins作为示例，向您展示如何使用 OIDC 链接到您的应用程序。

以下是配置中的一些专有名词:

`CASDOOR_HOSTNAME`: 私有部署的Casdoor域名或IP。

`JENKINS_HOSTNAME`: 部署Jenkins的域名或IP。

第1步。 部署Casdoor和Jenkins

首先，应该部署 Casdoor 和 Jenkins。

在成功部署后，您需要确保:

1. 将 Jenkins URL(Manage Jenkins → Configure System → Jenkins Location) 设置为 `JENKINS_HOSTNAME`

2. 可以登录并正常使用Casdoor 。
3. 将Casdoor 的 `origin` value (conf/app.conf) 设置为 `CASDOOR_HOSTNAME`。

```
conf > app.conf
8   dbName = casdoor
9   redisEndpoint =
10  defaultStorageProvider =
11  isCloudIntranet = false
12  authState = "casdoor"
13  httpProxy = "127.0.0.1:10808"
14  verificationCodeTimeout = 10
15  initScore = 2000
16  logPostOnly = true
17  origin = "http://10.144.1.2:8000"
                 CASDOOR_HOSTNAME
```

第2步： 配置Casdoor应用程序

1. 创建或使用现有的 Casdoor 应用程序。
2. 添加重定向URL： `http://JENKINS_HOSTNAME/securityRealm/finishLogin`

3. 添加您想要的provider并补充其他设置。

不出意外的话，您会在应用程序设置页面看到： `Client ID` 和 `Client secret` 就像

上面的图片一样。 我们将在下一步中使用它们。

打开你喜欢的浏览器，访问：http://`CASDOOR_HOSTNAME`/.well-known/openid-configuration，你会看到 Casdoor 的 OIDC 配置。

第3步 配置 Jenkins

First, we need to install OpenId Connect Authentication, Jenkins does not natively support OIDC.

After completing the installation, go to **Manage Jenkins** → **Configure Global Security**.

> **提示**
>
> Back up the Jenkins `config.xml` file, and use it to recover in case of setup errors.

1. In Access Control, **Security Realm** select `Login with Openid Connect`.
2. 在客户端ID中，指定上面注明的 `Client ID`。

3. 在Client serect中，指定上面注明的 `Client secret`
4. In Configuration mode, select `Automatic configuration` and fill in http://`CASDOOR_HOSTNAME`/.well-known/openid-configuration into Well-known configuration endpoint.

If your casdoor is deployed locally, you may need to select `Manual configuration` and input some information:

- 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`

5. Click **Advanced setting**, fill in the following:

- In User name field, specify `name`
- In Full name field, specify `displayName`
- In Email field, specify `email`

6. In the **Authorization** section, check “Logged-in users can do anything”. Disable “Allow anonymous read access”. You can configure more complex authorization later, for now check if OpenID actually works.

Log out of Jenkins, it should now redirect you to Casdoor for authentication.

username, Email or phone

Password

Auto sign in

Forgot password?

Sign In

Sign in with code

No account? sign up now

Jira

Casdoor 可以使用 OIDC 协议作为IDP 连接各种应用程序。 这里我们将以 Jira 作为示例来告诉您如何使用 OIDC 连接到您的应用程序。

以下是配置中的一些专有名词:

`CASDOOR_HOSTNAME`: 私有部署的Casdoor域名或IP。

`Jira_HOSTNAME`: 部署 Jira 的域名或 IP。

步骤1. 部署 Casdoor 和 Jira

首先，应该部署Casdoor 和 Jira 。

成功部署后，您需要确保:

1. 设置 Jira URL(Plans → Administration → System → 常规配置) 为 `Jira_HOSTNAME`

2. Casdoor 可以正常登录使用。
3. 您可以设置 CASDOOR_HOSTNAME = `http://localhost:8000`。 在 `prod` 模式下部署Casdoor。 详见 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. 查找重定向url：

3. 添加您的重定向url:

4. 添加您想要的提供商并补充其他设置。

不出意外的话，您会在应用程序设置页面看到： `Client ID` 和 `Client secret` 就像上面的图片一样。 我们将在下一步中使用它们。

打开你喜欢的浏览器，访问：http://`CASDOOR_HOSTNAME`/.well-known/openid-configuration，你会看到 Casdoor 的 OIDC 配置。

步骤3. 配置 Jira

1. 你应该安装 miniOrange 应用程序来支持 OAuth.你可以在 Plans->Administration->查找新的应用程序→serch

2. 您应该配置此应用

3. 将 `选定的应用程序` 设置为自定义 OpenId
4. 您可以在Cassdoor 应用程序页面找到客户端ID和密码。

- `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`

1. 您应该配置用户配置文件映射就像这样

2. 您可以稍后配置更复杂的授权。现在检查OpenID是否是正常工作状态。

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

Confluence

Casdoor can use OIDC protocol as IDP to connect various applications. Here we will use Confluence as an example to show you how to use OIDC to connect to your applications.

The following are some of the names in the configuration:

`CASDOOR_HOSTNAME`: Domain name or IP where Casdoor server is deployed.

`Confluence_HOSTNAME`: Domain name or IP where Confluence is deployed.

Step1. Deploy Casdoor and Confluence

Firstly, the Casdoor and Confluence should be deployed.

After a successful deployment, you need to ensure:

1. Set Confluence URL to `Confluence_HOSTNAME`.

2. Casdoor can be logged in and used normally.
3. You can set CASDOOR_HOSTNAME = `http://localhost:8000`. When deploy Casdoor in `prod` mode. See production mode.

Step2. Configure Casdoor application

1. Create or use an existing Casdoor application.
2. Find a redirect url:

3. Add a redirect url:

4. Add provider you want and supplement other settings.

Not surprisingly, you can get two values on the application settings page: `Client ID` and `Client secret` like the picture above, we will use them in the next step.

Open your favorite browser and visit: **http://`CASDOOR_HOSTNAME`/.well-known/openid-configuration**, you will see the OIDC configure of Casdoor.

Step3. Configure Confluence

1. You should install a miniOrange app to support OAuth.You can find this app in

2. You should config this app

3. Set `Selected Application` to Custom OpenId
4. You can find Client Id and Client Secret in Casdoor application page.

- `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`

1. You can configure more complex authorization later, for now check if OpenID actually works.

Log out of Confluence, and test 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可以轻易连接到 RuoYi-cloud。

步骤1. 部署Casdoor

首先，部署Casdoor。

您可以参考Casdoor 官方文档 Server Installation 。

在成功部署后，您需要确保：

- Casdoor服务已经成功运行，能通过http://localhost:8000 访问。
- 打开您最喜欢的浏览器并访问 http://localhost:7001，您将看到Casdoor的登录页面。
- 输入 `admin` 和 `123` 测试登录功能正常工作。

然后您可以通过以下步骤在自己的应用程序中快速实现基于 Casdoor 的登录页面。

步骤2. 配置Casdoor

如何配置casdoor可以参考 casdoor(配置casdoor和开发最好使用不同的浏览器)。

您还应该配置一个组织、应用程序和同步器。 您也可以参考 Casdoor。

需要注意的一些要点：

1. 修改同步器表格中的列

Table columns ⓘ :

Table columns Add

Column name	Column type	Casdoor column	Is hashed	Action
user_id	integer	Id		
dept_id	integer	Affiliation		
user_name	string	Name		
nick_name	string	DisplayName		
user_type	string	Type		
email	string	Email		
phonenumber	string	Phone		
sex	string	Gender		
avatar	string	Avatar		
password	string	Password		
del_flag	string	IsDeleted		
login_ip	string	CreatedIp		
create_time	string	CreatedTime		
password	string	Password		

2. 修改组织中的密码类型

Password type ⓘ : bcrypt

3. 您也应该打开软删除。

步骤3. 前端改造

3.1 跳转到casdoor的登录页

我们可以前端的sdk，这里我们以vue-sdk为例。 在 加入vue-sdk 后，您可以通过 getSigninUrl() 获得casdoor登录页面

您可以以您喜欢的方式链接到它，并且您可以删除一些不再使用的ruoyi-cloud的代码，如账户和密码EL输入框。

3.2 接收casdoor返回的code和state

在我们通过casdoor成功登录后，casdoor会把code和state返回给我们创建的页面。 我们可以通过create函数获取code和state。

```
created() {
    let url = window.document.location.href//get url
    let u = new URL(url);
```

对于RuoYi-Cloud，我们只是将发送的账户和密码更改为code和state。 因此，相对于原始登录，它只是改变发送到后端的内容。

步骤4. 后端改造

4.1 接收前端返回的code和state

```
@PostMapping("login")
public R<?> callback(@RequestBody CodeBody code) {//we should define
a CodeBody entity which have code and state
    String token = casdoorAuthService.getOAuthToken(code.getCode(),
code.getState());
    CasdoorUser casdoorUser =
casdoorAuthService.parseJwtToken(token);
    if(casdoorUser.getName()!=null){
        String casdoorUserName = casdoorUser.getName();

if(sysLoginService.getUserByCasdoorName(casdoorUserName)==null){//if
database haven't this user
            // add this user into database
            sysLoginService.casdoorRegister(casdoorUserName);
        }
    }
    LoginUser userInfo =
sysLoginService.casdoorLogin(casdoorUser.getName());//get this
user's information by database
    return R.ok(tokenService.createToken(userInfo));
}
```

在这种方法中，我们使用 casdoor-SpringBoot-sdk ，并对 RuoYi-Cloud 方法稍做修改。

例如，RuoYi-Cloud原本的register使用账户和密码注册，我将register改成了casdoorRegister。

我还添加了一个getUserByCasdoorname这样的方法来检查账户是否存在，并将基于账户

和密码用户切换成了当前用户。

这很简单，因为我们只需要删除检查密码的部分。

步骤5. 总结

5.1 前端

- 我们需要删除原始的登录和注册功能。
- 我们还需要接受code和state并发送到后端。

5.2 后端

RuoYi 后端具有极好的登录和注册功能。 我们只需要稍做改变，因此非常方便。

步骤6. 详细步骤

1. 部署和配置Casdoor。 部署并配置casdoor。我们必须注意组织的密码类型必须为 bcrypt ，因为 RuoYi-Cloud的密码类型是bcrypt 。

2. 我们应该使用casdoor的同步器将数据库用户中的用户信息复制到您的casdoor组织。此步骤可以将原始帐户导入到casdoor。

3. 在我们部署了casdoor之后，我们应当改造前端。 我们应该关闭 RuoYi 的校验码功能

```
// checkcode switch
captchaEnabled: false,
// register switch
register: true,
```

   请注意，在 nacos 中也要关闭RuoYi-Cloud校验码功能。 请注意，RuoYi-Cloud开放注册功能还需要更改配置 sys.account.registerUser 的值为true。

4. 我们应该添加按钮跳到casdoor的按钮并更改数据的loginForm

```
<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: ""
},
```

你可以通过casdoor-vue-sdk或casdoor-SpringBoot-sdk获得这里的网址。

5. 由于我们不使用原先的登录功能，我们应该删除关于 cookie 和校验码的方法。

所以创建新的函数：

```
created() {
    let url = window.document.location.href//get url
    let u = new URL(url);
    this.loginForm.code = u.searchParams.get('code')//get code and state
    this.loginForm.state = u.searchParams.get('state')
    if(this.loginForm.code!=null&&this.loginForm.state!=null){//if code and state is null, execute handleLogin
      this.handleLogin()
    }
}
```

6. 事实上，我们只需要更改发送到后端的参数并删除不需要的函数，此外无需其他改动。

```
handleLogin() {
  console.log("进入handleLogin")
      this.$store.dispatch("Login", this.loginForm).then(() => {
        this.$router.push({ path: this.redirect || "/" }).catch(()=>{});
      }).catch(() => {
        this.loading = false;
        if (this.captchaEnabled) {
          this.getCode();
          console.log(this.getCode)
        }
      });
  }
}
```

```
Login({ commit }, userInfo) {
  const code = userInfo.code
  const state = userInfo.state
  return new Promise((resolve, reject) => {
    login(code, state).then(res => {
      console.log("LOGIN")
      let data = res.data
      setToken(data.access_token)
      commit('SET_TOKEN', data.access_token)
      setExpiresIn(data.expires_in)
      commit('SET_EXPIRES_IN', data.expires_in)
      resolve()
    }).catch(error => {
      reject(error)
    })
  })
},
```

```
export function login(code, state) {
  return request({
    url: '/auth/login',
    headers: {
      isToken: false
    },
    method: 'post',
    data: {code, state}
  })
}
```

7. 在后端导入依赖

pom.xml

```
<dependency>
    <groupId>org.casbin</groupId>
    <artifactId>casdoor-spring-boot-starter</artifactId>
    <version>1.2.0</version>
</dependency>
```

您还需要在代码上配置casdoor。 8. 回调函数被定义为重定向函数。 我对 sysLoginService 中的一些方法做了小修改 删除检查密码步骤，因为我们不需要它。

```
@PostMapping("login")
public R<?> callback(@RequestBody CodeBody code) {//we should define
a CodeBody entity which have code and state
    String token = casdoorAuthService.getOAuthToken(code.getCode(),
code.getState());
    CasdoorUser casdoorUser =
casdoorAuthService.parseJwtToken(token);
    if(casdoorUser.getName()!=null){
        String casdoorUserName = casdoorUser.getName();

if(sysLoginService.getUserByCasdoorName(casdoorUserName)==null){//if
database haven't this user
            // add this user into database
            sysLoginService.casdoorRegister(casdoorUserName);
        }
    }
    LoginUser userInfo =
sysLoginService.casdoorLogin(casdoorUser.getName());//get this
user's information by database
    return R.ok(tokenService.createToken(userInfo));
}
```

9. SysLoginService的新方法

```java
public LoginUser casdoorLogin(String username){
    // execute user
    R<LoginUser> userResult =
remoteUserService.getUserInfo(username, SecurityConstants.INNER);
    if (R.FAIL == userResult.getCode())
    {
        throw new ServiceException(userResult.getMsg());
    }

    if (StringUtils.isNull(userResult) ||
StringUtils.isNull(userResult.getData()))
    {
        recordLogService.recordLogininfor(username,
Constants.LOGIN_FAIL, "this user is not exist");
        throw new ServiceException("user: " + username + " is not
exist");
    }
    LoginUser userInfo = userResult.getData();
    SysUser user = userResult.getData().getSysUser();
    if (UserStatus.DELETED.getCode().equals(user.getDelFlag()))
    {
        recordLogService.recordLogininfor(username,
Constants.LOGIN_FAIL, "sorry, your account was deleted");
        throw new ServiceException("sorry, your account: " +
username + " was deleted");
    }
    if (UserStatus.DISABLE.getCode().equals(user.getStatus()))
    {
        recordLogService.recordLogininfor(username,
Constants.LOGIN_FAIL, "your account is disabled, you can contact
admin ");
         throw new ServiceException(sorry, your account: " +
username + " is disabled");
    }
    recordLogService.recordLogininfor(username,
Constants.LOGIN_SUCCESS, "login successfully");
    return userInfo;
}
```

```java
public String getUserByCasdoorName(String casdoorUsername){
    R<LoginUser> userResult =
remoteUserService.getUserInfo(casdoorUsername,
SecurityConstants.INNER);
    if (StringUtils.isNull(userResult) ||
StringUtils.isNull(userResult.getData()))
    {
        //if this user is not in RuoYi-Cloud database and casdoor
have this user, we should create this user in database
        return null;
    }
    String username =
userResult.getData().getSysUser().getUserName();
    return username;
}
```

```java
public void casdoorRegister(String username){
    if (StringUtils.isAnyBlank(username))
    {
        throw new ServiceException("User must fill in");
    }
    SysUser sysUser = new SysUser();
    sysUser.setUserName(username);
    sysUser.setNickName(username);
    R<?> registerResult =
remoteUserService.registerUserInfo(sysUser, SecurityConstants.INNER);
    System.out.println(registerResult);
    if (R.FAIL == registerResult.getCode())
    {
        throw new ServiceException(registerResult.getMsg());
    }
    recordLogService.recordLogininfor(username, Constants.REGISTER,
"register successfully");
}
```

Pulsar-manager

Casdoor可以轻易连接到Pulsar-manager。

因为 Pulsar-manager中已经添加了连接casdoor 的代码，所以我们只需要在后端的application.yml中做一些配置，并在前端打开一个开关 。

步骤1. 部署Casdoor

首先，部署Casdoor。

您可以参考Casdoor 官方文档 Server Installation 。

在成功部署后，您需要确保:

- Casdoor服务已经成功运行，能通过http://localhost:8000 访问。
- 打开您最喜欢的浏览器并访问 http://localhost:7001，您将看到Casdoor的登录页面。
- 输入 `admin` 和 `123` 测试登录功能正常工作。

然后您可以通过以下步骤在自己的应用程序中快速实现基于 Casdoor 的登录页面。

步骤2. 配置Casdoor

如何配置casdoor可以参考 casdoor(配置casdoor和开发最好使用不同的浏览器)。

您还应该配置一个组织和一个应用程序。 您也可以参考 Casdoor。

步骤2.1创建一个组织

步骤2.2 创建一个应用程序

步骤3. 打开 Pulsar-Manager 前端中的开关。

打开此开关，将代码和状态发送到后端。

这个开关在pulsar-manager/front-end/src/router/index.js的第80行

```
-   // mode: 'history', // require service support
+   mode: 'history', // require service support
```

步骤4. 后端配置

您需要在pulsar-manager/src/main/resources/application.properties的154行加入casdoor的配置

```
casdoor.endpoint = http://localhost:8000
casdoor.clientId = <client id in previous step>
casdoor.clientSecret = <client Secret in previous step>
casdoor.certificate=<client certificate in previous step>
casdoor.organizationName=pulsar
casdoor.applicationName=app-pulsar
```

ShenYu

ShenYu 可通过插件使用 casdoor。

步骤1. 部署Casdoor

首先，部署Casdoor。

您可以参考Casdoor 官方文档 Server Installation 。

成功部署后，您需要确保：

- Casdoor服务已成功运行，能通过http://localhost:8000 访问。
- 打开您最喜欢的浏览器并访问 http://localhost:7001，您将看到Casdoor的登录页面。
- 输入 `admin` 和 `123` 测试登录功能正常工作。

然后您可以通过以下步骤在自己的应用程序中快速实现基于 Casdoor 的登录页面。

步骤2. 配置Casdoor应用程序

1. **创建或使用现有的 Casdoor 应用程序**。
2. **添加您的重定向url**

3. 在证书编辑页面上，您可以看到您的 证书。

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 仅有单行输入框，所以我们需要在每行证书中添加 \n

Certificate ⓘ :

Copy certificate | Download certificate


```
-----BEGIN CERTIFICATE-----\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/RbSAf7323OsjYnjjEgInUtRohnRgCpjIk/Mt2Kt84Kb0\n
wn8CAwEAAaMQMA4wDAYDVR0TAQH/BAIwADANBgkqhkiG9w0BAQsFAAOCAgEAn2If\n
DKkLX+F1vKRO/5gJ+PIr8P5NKuQkmwH97b8CS2gS1phDyNgIc4/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

你可以复制并粘贴到Shenyu 后台配置的证书。

您不需要将其保存到casdoor证书编辑页面，因为只需复制即可。

2. 配置shenyu casdoor的插件

您可以配置您需要使用casdoor配置

3. 使用Casdoor提供的相关服务

3.1 直接访问Web

3.2像这样登录casdoor

hi! null! I'm Shenyu-Gateway System. Welcome!

3.3 Headers中的Carry token，你也可以访问

3.4 它也可以保存 Headers 的用户名、id 和组织，以便下次使用。

ShardingSphere

shardingsphere-elasticjob-ui已经集成了Casdoor。 配置后我们可以使用它。

步骤1. 部署 Casdoor

首先，部署Casdoor。

您可以参考Casdoor 官方文档 Server Installation 。

成功部署后，您需要确保：

- Casdoor服务已成功运行，能通过http://localhost:8000 访问。
- 打开您最喜欢的浏览器并访问 http://localhost:7001，您将看到Casdoor的登录页面。
- 输入 `admin` 和 `123` 测试登录功能正常工作。

然后您可以通过以下步骤在自己的应用程序中快速实现基于 Casdoor 的登录页面。

步骤2. 配置ShardingSphere 中的 cassdoor 应用程序并配置

1.创建或使用现有的 Casdoor 应用程序

重定向URL 取决于您需要重定向的网址。 他选择的数据将在下次使用。

2. 在证书编辑页面上，您可以看到您的 `证书`。

Certificate ⓘ :

Copy certificate | Download certificate

Private


```
-----BEGIN CERTIFICATE-----
MIIE+TCCAuGgAwIBAgIDAeJAMA0GCSqGSIb3DQEBCwUAMDYxHTAbBgNVBAoTFENh
c2Rvb3IgT3JnYW5pemF0aW9uMRUwEwYDVQQDEwxDYXNkb29yIENlcnQwHhcNMjEx
MDE1MDgxMTUyWhcNNDExMDE1MDgxMTUyWjA2MR0wGwYDVQQKExRDYXNkb29yIE9y
Z2FuaXphdGlvbjEVMBMGA1UEAxMMQ2FzZG9vciBDZXJ0MIICIjANBgkqhkiG9w0B
AQEFAAOCAg8AMIICCgKCAgEAslnpb5E1/ym0f1RfSDSSE8IR7y+Iw+RJjl74e5ej
rq4b8zMYk7HeHCyZr/hmNEwEVXnhXu1P0mBeQ5ypp/QGo8vgEmjAETNmzkI1NjOQ
CjCYwUrasO/f/MnI1C0j13vx6mV1kHZjSrKsMhYY1vaxTEP3+VB8Hjg3MHFWrb07
uvFMCJe5W8+0rKErZCKTR8+9VB3janeBz//zQePFVh79bFZate/hLirPK0Go9P1g
OvwIoC1A3sarHTP4Qm/LQRt0rHqZFybdySpyWAQvhNaDFE7mTstRSBb/wUjNCUBD
PTSLVjC04WIISf6Nkfx0Z7KvmbPstSj+btvcqsvRAGtvdsB9h62Kptjs1Yn7GAuo
I3qt/4zoKbiURYxkQJXIvwCQsEftUuk5ew5zuPSIDRLoLByQTLbx0JqLAFNfW3g/
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/irHz21W4jt4rxzCvl
2nR42Fybap3O/g2JXMhNNROwZmNjgpsF7XVENCSuFO1jTywLaqjuXCg54IL7XVLG
omKNNNcc8h1FCeKj/nnbGMhodnFWKDTsJcbNmcOPNHo6ixzqMy/Hqc+mWYv7maAG
Jtevs3qgMZ8F9Qzr3HpUc6R3ZYYWDY/xxPisuKftOPZgtH979XC4mdf0WPnOBLqL
2DJ1zaBmjiGJoIvb7XNVKcUfDXYw85ZTZQ5b9cIl4e+6bmyWqQltlwt+Ati/uFEV
XzCj70B4lALX6xau1kLEpV9O1GERizYRz5P9NJNA7KoO5AVMp9w0DQTkt+LbXnZE
HHnWKy8xHQKZF9sR7YBPGLs/Ac6tviv5Ua15OgJ/8dLRZ/veyFfGo2yZsI+hKVU5
nCCJHBcAyFnm1hdvdwEdH33jDBjNB6ciotJZrf/3VYalWSaIADosHAgMWfXuWP+h
8XKXmzlxuHbTMQYtZPDgspS5aK+S4Q9wb8RRAYo=
-----END CERTIFICATE-----
```


3.配置在 ShardingSphere 中的应用程序

首先，我们需要找到应用程序以及我们需要配置的属性。

其次，我们需要复制Casdoor应用程序中的数据并将其粘贴到应用中。

```
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\
8baawdSdJjH2RgmFoDeE2r8NVRfhbR8KnCO1ddTJKuS1N0/irHz21W4jt4rxzCvl\n\
2nR42Fybap3O/g2JXMhNNROwZmNjgpsF7XVENCSuFO1jTywLaqjuXCg54IL7XVLG\n\
omKNNNcc8h1FCeKj/nnbGMhodnFWKDTsJcbNmcOPNHo6ixzqMy/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

ShardingSphere
username
password
Login
Login with Casdoor
Copyright © The Apache Software Foundation, Licensed under the Apache License 2

Apache IoTDB

Casdoor can simply connect to Apache IoTDB.

Because the code for connecting casdoor has been added in Apache IoTDB Web Workbench, we just need to configure application.yml in back-end and open front switch.

Step1. Deploy Casdoor

Firstly, the Casdoor should be deployed.

You can refer to the Casdoor official documentation for the Server Installation.

After a successful deployment, you need to ensure:

- The Casdoor server is successfully running on **http://localhost:8000**.
- Open your favorite browser and visit **http://localhost:7001**, you will see the login page of Casdoor.
- Input `admin` and `123` to test login functionality is working fine.

Then you can quickly implement a Casdoor-based login page in your own app with the following steps.

Step2. Configure Casdoor

Configure casdoor can refer to casdoor(Configure casdoor's browser better not use one browser as your develop browser).

You also should configure an organization and an application. You also can refer to casdoor.

2.1 you should create an organization

2.2 you should create an application

Step3. Open Apache IoTDB Web Workbench front-end switch

Open this switch to make code and state send to back-end.

This switch in the iotdb-web-workbench/fronted/.env

Step4. Configure back-end code

You should configure casdoor's Configuration in the iotdb-web-workbench/backend/src/main/resources/application.properties

```
casdoor.endpoint = http://localhost:8000
casdoor.clientId = <client id in previous step>
casdoor.clientSecret = <client Secret in previous step>
casdoor.certificate=<client certificate in previous step>
casdoor.organizationName=IoTDB
casdoor.applicationName=app-IoTDB
```

Result

IoTDB WorkBench
IoTDB WorkBench
Welcome To IoTDB WorkBench
Account
Please Input Account
Password
Forget Password?
Sign In
Sign In With Casdoor

FireZone

Casdoor can use OIDC protocol as IDP to connect various applications. Here we will use FireZone as an example to show you how to use OIDC to connect to your applications.

Step 1. Deploy Casdoor and FireZone

Firstly, the Casdoor and FireZone should be deployed.

After a successful deployment, you need to ensure:

1. Set FireZone URL(Sigin → Security → Add OpenID Connect Provider) to FIREZONE_HOSTNAME.

2. Casdoor can be logged in and used normally.

3. `CASDOOR_HOSTNAME`: http://localhost:8000. If you deploy Casdoor using default `app.conf`.

Step 2. Configure Casdoor application

1. Create or use an existing Casdoor application.

2. Add a redirect url:

For example, the Configid in the FireZone Provider is TEST, so the redirect URL should be `http://[FIREZONE_HOST]/auth/oidc/[PROVIDER_CONFIG_ID]/callback/`

Open your favorite browser and visit: `http://[CASDOOR_HOSTNAME]/.well-known/openid-configuration`, you will see the OIDC configure of Casdoor.

3. Configure FireZone, Security → Add OpenID Connect Provider

- `Discovery Document URI`: FireZone Provider Discovery Document URI should be `https://[CASDOOR_HOST]/.well-known/openid-configuration`
- `Scopes`: `openid email profile`

- `ConfigID`: ConfigID should be the PROVIDER_COONFIG_ID of the redirect URL and should correspond to casdoor redirect URL
- `Auto create users`: Successful login will automatically create a user

Log out of FireZone, and test SSO.

JavaScript

微信小程序

在 微信小程序中使用 Casdoor

微信小程序

> **信息**
>
> Casdoor支持1.41.0版后的微信小程序

介绍

由于微信小程序不支持标准的 OAutth，所以它不能跳转到自主机的 Casdoor 网页进行登录。 因此，将Casdoor 用于微信小程序的过程与普通方案的过程有所不同。

这份文件将讨论如何通过Casdoor访问微信小程序。 您可以在 GitHub 上找到示例：casdoor-wechat-miniprogram-example。 详细信息可参见微信小程序 登录文档 。

> 以下是配置中的一些专有名词：
>
> `CASDOOR_HOSTNAME`：私有部署的Casdoor域名或IP。 例如：`https://door.casbin.com`.

第一步： 部署Casdoor

首先，您应先部署 Casdoor。

成功部署后，您需要确认：

1. Casdoor 可以正常登录使用。
2. 将 Casdoor 的 `origin` 值 (conf/app.conf) 设置为 `CASDOOR_HOSTNAME`。

```
conf > app.conf
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
```

第二步： 配置Casdoor应用程序

1. 在 casdoor 创建一个wechat idp, 并填写微信小程序开发平台给您的 `APPID` 和 `APPSECRET` ：

2. 创建或使用现有的 Casdoor 应用程序。
3. 将上面添加的 idp 添加到您想要使用的应用程序中。

> **TIPS**
>
> 为方便起见，casdoor 将在应用程序中读取第一个WeChat类型 idp 默认是微信小程序 idp 。
>
> 因此，如果你想要在这个应用中使用微信小程序，请不要在一个应用中添加 WeChat 类型 idp 。

第三步： 编写微信小程序

WeChat Mini 方案提供一个内部登录的API并获取代码。 您需要做的只是将此代码发送到Casdoor。 Casdoor 将使用此代码从 WeChat 服务器获取一些信息(例如OpenID、SessionKey 等)。

下面代码展示如何完成上述过程:

```
// login in mini program
wx.login({
  success: res => {
    // this is your login code you need to send to casdoor
    console.log(res.code)

    wx.request({
      url: `${CASDOOR_HOSTNAME}/api/login/oauth/access_token`,
      method: "POST",
      data: {
        "tag": "wechat_miniprogram", // required
        "client_id": "6825f4f0af45554c8952",
        "code": res.code,
        "username": this.data.userInfo.nickName, // update user
profile, when you login.
        "avatar": this.data.userInfo.avatarUrl,
      },
      header:{
        "content-type": "application/x-www-form-urlencoded",
      },
      success: res => {
        console.log(res)
        this.globalData.accessToken = res.data.access_token // get
casdoor's accessToken
      }
    })
```

值得一提的是，`tag` 参数是强制性的，您需要让户了解这是来自微信小程序的请求。

上述代码在用户登录时传入微信小程序用户名和头像uri。 您也可以通过这两个参数而不首先传递它们。 然后在登录成功并获得访问令牌后将它们传递到 casdoor：

```
wx.getUserProfile({
  desc: 'share your info to casdoor',
  success: (res) => {
    this.setData({
      userInfo: res.userInfo,
      hasUserInfo: true
    })
    console.log(app.globalData.accessToken)
    wx.request({
      url: `${CASDOOR_HOSTNAME}/api/update-user`, // casdoor uri
      method: "POST",
      data: {
        "owner": "test",
        "name": "wechat-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，您可以使用 accessToken 作为任何Casdoor操作的访问令牌

> TIPS
>
> 目前 Casdoor 无法将现有账户绑定到微信小程序用户。 在 Catdoor 从WeChat 获取 openID ，如果此 id 不存在。 将创建一个新用户，如果它存在，将使用旧用户。

Lua

APISIX

在 APISIX 中使用 Casdoor

APISIX

目前有两种方法可以使用 Casdoor 通过 APISIX 插件连接到 APISIX， 并保护 APISIX 背后的 API： 使用 APISIX 的 Casdoor 插件或使用 APISIX 的 OIDC 插件。

通过 APISIX 的 Casdoor 插件连接 Casdoor

这个名为 authz-casdoor 的插件可以保护 APISIX 背后的 api， 强制每个请求都经过身份验证，而无需修改 api 代码。

如何启用它?

您需要在创建路由时指定此插件，并给出所有必需的字段。 参见下面的示例。

```
curl "http://127.0.0.1:9080/apisix/admin/routes/1" -H "X-API-KEY:
edd1c9f034335f136f87ad84b625c8f1" -X PUT -d '
{
  "methods": ["GET"],
  "uri": "/anything/*",
  "plugins": {
    "authz-casdoor": {
        "endpoint_addr":"http://localhost:8000",
        "callback_url":"http://localhost:9080/anything/callback",
        "client_id":"7ceb9b7fda4a9061ec1c",
        "client_secret":"3416238e1edf915eac08b8fe345b2b95cdba7e04"
    }
  },
  "upstream": {
```

在本例中，我们使用apisix的管理API创建了指向“httpbin.org:80”的路由"/anything/*"，并启用了“authz casdoor”。 此路由现在受到 Casdoor 的身份验证保护。

属性

名称	类型	申请标准	默认	有效期	描述
endpoint_addr	字符串	必填			Casdoor的url。
client_id	字符串	必填			Casdoor 中的 client id。
client_secret	字符串	必填			Casdoor 中的 client secret。
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。 应该强调的是，此回调url**必须属于您为路由指定的“uri”**，例如，在本例中，http://localhost:9080/anything/callback显然属于"/anything/*"。 只有通过这种方式，插件才能拦截并利用对 callback_url 的访问（这样插件才能在 Oauth2 中获得代码和状态）。 Callback_url 的逻辑完全由插件实现，因此无需修改服务器来实现此回调。

第二个参数“endpoint_addr”显然是Casdoor的url。 第三个和第四个参数是“client_id”和

“client_secret”，您可以在注册应用程序时从Casdoor获取这些参数。

它是如何工作的?

假设一个以前从未访问过此路由的新用户将访问它(http://localhost:9080/anything/d?param1=foo¶m2=bar)，考虑到已启用“authz casdoor”，此访问将首先由“authz casdoor”插件处理。 然后检查会话，若此用户尚未通过身份验证后，将拦截访问。 用户想要继续访问原始url后，他将被重定向到Casdoor的登录页面。

成功登录用户名和密码(或他使用的任何方法)， Casdoor 会将此用户重定向到“callback_url”，带有GET 参数“code”和“state”。 因为插件知道"callback_url"，当这次拦截访问"callback_url"时， Oauth2中的“授权代码流”逻辑将被触发， 这意味着此插件将请求访问令牌以确认此用户是否真的登录。 确认后，此插件将会将此用户重定向到原始用户想访问的URL，这是我们先前保存的。 登录状态也将保持在会话。

下次此用户想访问此路由后面的网址 (例如，http://localhost:9080/anything/d)，发现此用户以前已被身份验证。 此插件将不再重定向此用户，以使此用户能够不受干扰地访问他在路由下想要的任何路径。

通过 APISIX 的 OIDC 插件连接 Casdoor

Casdoor 可以使用 OIDC 协议链接到 APISIX，这份文档将向您展示如何处理相关问题。

以下是配置中的一些专有名词:

`CASDOOR_HOSTNAME`： 私有部署的Casdoor域名或IP。

`APISIX_HOSTNAME`: 部署 APISIX 的域名或 IP。

步骤1. 部署Casdoor和APISIX

首先，应该部署 Casdoor and APISIX。

在成功部署后，您需要确保：

1. 可以登录并正常使用Casdoor。
2. 将Casdoor的 `origin` 值 (conf/app.conf) 设置为 `CASDOOR_HOSTNAME`。

```
conf > app.conf
8   dbName = casdoor
9   redisEndpoint =
10  defaultStorageProvider =
11  isCloudIntranet = false
12  authState = "casdoor"
13  httpProxy = "127.0.0.1:10808"
14  verificationCodeTimeout = 10
15  initScore = 2000
16  logPostOnly = true
17  origin = "http://10.144.1.2:8000"
                CASDOOR_HOSTNAME
```

步骤2. 配置Casdoor应用程序

1. 创建或使用现有的 Casdoor 应用程序。
2. 添加一个重定向网址： `http://APISIX_HOSTNAME/REDIRECTWHATYOWANT` 并更改 `REDIRECTWHATYOWANT` 到你需要的重定向网址上。
3. 选择 "JWT-Empty" 作为令牌格式选项
4. 添加您想要的提供商并补充其他设置。

不出意外的话，您会在应用程序设置页面看到： `Client ID` 和 `Client secret` 就像上面的图片一样。 我们将在下一步中使用它们。

打开你喜欢的浏览器，访问：http://`CASDOOR_HOSTNAME`/.well-known/openid-configuration，你会看到 Casdoor 的 OIDC 配置。

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 APISX OIDC 文档定制设置，使用以下路由设置：

```
#Use your own X-Api-Key
$ curl  -XPOST APISIX_HOSTNAME/apisix/admin/routes -H "X-Api-Key: edd1c9f034335f136f87ad84b625c8f1" -d '{
  "uri": "/get",
  "name": "apisix_casdoor_test",
  "plugins": {
    "openid-connect": {
      "client_id": "Client ID",
      "client_secret": "Client secret",
      "discovery": "http://CASDOOR_HOSTNAME/.well-known/openid-configuration",
      "introspection_endpoint_auth_method": "client_secret_basic",
      "logout_path": "/logout",
      "realm": "master",
      "redirect_uri": "http://APISIX_HOSTNAME/REDIRECTWHATYOUWANT",
      "bearer_only": false,
      "set_id_token_header": false,
      "access_token_in_authorization_header": true,
      "set_access_token_header": true,
      "set_userinfo_header": false,
      "realm": "master"
    }
  },
  "upstream": {
    "type": "roundrobin",
    "nodes": {
      "httpbin.org:80": 1
```

现在，请访问 `http://APISIX_HOSTNAME/get`，浏览器会将您重定向到Casdoor登录页面， 并且在登录成功后，您会看到我们已经向 httpbin.org 发出了请求。


```
{
  "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-
EWV0d1FQwDWa6uweDC4bh7M1mrKgABZ11P60dZ9HJebMhhmLW3dptnO38F3GhG9hDi7KoSD10CO0a4y7Koq1jNCodLL5zIJZ1K0usRQ3biWo54d8fXbZ_dmF05LrFFsxgYgp2-NegmaAHnE21HSkrD4wuX0Z6Q-ckfhIvEbU0C
4rxyugiSDthWSP1jfpLuWEbO19gHNbKJPyKJ4YZ5EYvUIbepqrqyyTug5zF6WyIUTqY7TQ7aDQd8iUvetzcrswuj_ESNKTQtDVitaL9GkVqftOiswm2b0ryW9P_lB76ZIOtFtnnUR4_a6TzQjHaKYG9vtZRGSDIYlbTdHu0_F
y0Jlayp7sJltZO9bqurh2YXQGXoXWMxOPdZ44AQ15detkUycFeGAxlIaFgfBQHQiyeCsE4XOJuxHv3gTKu9lYncwuhHJvzohI6KeZkK5P5xJhoZCfd3_yZb8a_ewU0dWXNruPqrsfd0yhYEPVHB-
3XEHdllJbW27tZEDXDzBEVmEAokS5QPDKZkIPwRtqqb6G1OKq4tXQWeNHPOrKaiItnBlEgD1rJbMKpV74nhYeihfsQnsamc_ZS6AaEFOMwAu1fROslAsfCUj6kX47Q1_HV2VoKVfa2WAjPNhzrUAmrrzpwZBmn_lsFSa664f8
9d8U3jmU8wGhL_bRt8-AdRQOGYQht8ghJcat1OBIn6EUPB7G1CVsIaYYIBA2ylRBx6NV-J3NMRUsj5YkMbUqPFOLuR8JkhyYPs2spKMgcJGpAX6oaxVmtwc3U8yx5e4HHM3O2Q8lMP2vW41xffA81rqXWI4nOqC1B-fQRJSEv
iCi6FOiFKvYMog7cfd4_LK9SIK6RTasyoJpPSEG26onqWSx8-jsRwNRykSb7eoYFThtALOOXmkifgiJT53Id4JHP8O3RXsISguKPWtMjR6T1whIWQOOjr2wn182Xxqh8OprnlL8VLbLv_4e0p6uAbv4RnXfdE2P8Us6UOnNWz
oB6hrD99aM6NK_mxUZdoDGDlaRoeztXgNhvlyMO6SpKaer0xkbw7Gf3hNHq8Oh8xdkhx5QirUK24i2SoD9hqlfl0pTG3ZjJffNY958EJMLxmfbsAcYN3C8or2WrMoVndKWe-LsAlrFydrkHd5cT7AUqc_znij8CUgsYj1C5d-
V1H8OgnbAu1Qvg6gym7qzBe6CcYn7B69CYT8wdrtwyHnGS7rurQy3cu9blq8J1WN9Nwf2QgaovnooLcHEUPVOIMBkqtMlY8OCK7Ht-drf3RB6fNgOWaswbtPA4oUAXOQ9VHALhDzb9I7vBWBO3UTqDBk37OyvJWeOq6YfiY7z
jz4rjx8T8nMlOWEUUzQLNIVCH4bDt1GFX4GalA8hZT5W4WMOdcHL59XE4fXcHhtG18QGxCVrJ71Zz0sxTlugclUEjzXGGsGX7Dy0DkRMt3n09AJwtKd8rqUATrc21ohgGJjLNWj4K_BOLMfW-ifJQKNxI6NcrbLGU-
P_nIvIiPLRNOdmCjySpShqdNPQCi3B9YiaBLLBO4jBjzLNXkmkswf5LGFQEt16ZsTkcAbWuc1DgqjgFFESRdTS42G1wsUW97xOQ4QATAlWHR4Pyp3oR3tmpOG9GAcjnghc2POatcSOW5Ab70TVTKOVNYbLHnn6NT4tqgyeUDw
Isq4inM4UsmUG4AVzQvaXtPtYrEc03cOsfvFpoTOXH5ird92tLTZ36944oPglOuOr6OWxGbbs3Fi7Pl_BLuo6WkkI5U_FgUrPjEDqrMs_BjhxpDcb4tKavfopUlc4_PhnOyk1pwCNki_UpFX-MyR6Sro-LyAOHN9O7eNXAPig
wxLHbQxklzT9qN_cplDWmRFM9bEGuj4Vyn-1ZJNbSc49tGpLciaY-FfUXPn76At_mAfkJB1DAPT3ruhrs97T-W6k0mMxnWRz7BH4dQeOzYuDPcdqtqPcWZqiOeYjbhF1Jw9sXvj_wOkXBMnI7RNhKfU8a_8ZtBWpqvU9yYqP3
VEzho0kykCHcc_rmcfirndKnTcBDnMUbvcsikMFJpQFccEwZbB3Y73neTEYchs-XpOffSIhAq9BKtK-qWQCou2tYmCIMwvjx3XOXyB-CDyKpEX3oaxYuUzjgK2IR2FXRYhvFA1y4PAii3nvLNqT18f3oLrAZgago6To0i5VvZ
ka1DY5U2MN8NnKRTbaF6yOHgh3HWjdnI1J9GfhhaX9CFF2WOh0KZOaLSHfyPHW5nI3AvcXyPleaITj_clOK4qmo4TDPrdCiG3qjeOLS71gP6itjJBhdhjQe-PaQ855mcU4i3N5EuaQZSorJ3q21PiXENVqpaKBB1sZj5ansC-
sEhOr9q8s2iZfDk99DJGAzZU6GGCPGfJTOjFbbn2kI1Lox_sDg_1KUXMJv3h2pvQYIzS3nJFVvqj_-SGexIAM9ROzUuQU3F9IF1b_1aenlDOeHEnrQPxxsJMaIYV1_ACPLP_xvK4rMX7pY_pkl-oSQVjYu7v_LATxjQOW31gI
Axgrpiqn4b7AFIvXb4C9n7gQSvJqZwtnaU39VxBkGbWItP8HW4jxhaCS0GuYJTfUSc7ALO04i-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":
"eyJtc2ciOiIiLCJzdWIiOiI0ODNiZTBlYiO2NjJiLTQ5YzMtODczZS03MTU0NjY0ZmQzNGEiLCJkYXRhIjp7Imxhcmsi0iIiLCJnaXRsYWIiOiIiLCJhcHBsZSI6IiIsImF6dXJlYWQiOiIiLCJzbGFjayI6IiIsInBhc3N3
sInRhZyI6InN0YWZmIiwicGFzc3dvcmQiOiIxMjMiLCJhZGRyZXNzIjp7fSwicHJvcGVydGl1cyI6e3OsImxkYXAiOiIiLCJkaXNwbGF5TmFtZSI6IkFkbWluIiwiYXZhdGFyIjoiaHROcHM6XC9cL2Nhc2Jpbi5vcmdcL2lt
lciI6ImJlaWx0LWluIiwicmVnaW9uIjoiIiwibGFuZ3VhZ2Ui0iIiLCJzY29yZSI6MjAwMCwiY3J1YXR1ZFRpbWUiOiIyMDIxLTEyLTE0VDA1OjQ1OjE3WiIsImlzT25saW51IjpmYWxzZSwidXBkYXR1ZFRpbWUiOiIiLCJp
hbC11c2VyIiwic2lnbnVwQXBwbGljYXRpb24iOiIiLCJoYXNoIjoiMjlmMDA5MzAzODNiOWMyM2U4MzAzNmI4OTg3YTlkNzAiLCJpZENhcmRUeXBlIjoiIiwiaWRDYXJkIjoiIiwiZW1haWwiOiJhZG1pbkBleGFtcGxlLmNv
iOiIiLCJwcmVlYXNoIjoiMjlmMDA5MzAzODNiOWMyM2U4MzAzNmI4OTg3YTlkNzAiLCJjcmVhdGVkSXAiOiIiLCJsYXN0U2lnbmluVG1tZSI6IiIsImxhc3RTaWduaW5JcCI6IiIsImdpdGh1YiI6IiIsImdvb2dsZSI6IiIs
iIn0sImRhdGEyIjp7InBob251UHJlZml4IjoiODYiLCJwYXNzd29yZFNhbHQiOiIiLCJkaXNwbGF5TmFtZSI6IkJ1aWx0LWluIE9yZ2FuaXphdGlvbiIsImVuYWJsZVNvZnREZWxldGlvbiI6ZmFsc2UsImRlZmF1bHRBdmFC
odHRwczpcL1wvZXhhbXBsZS5jb20iLCJtYXN0ZXJQYXNzd29yZCI6IiIsImZhdml jb24iOiJodHRwczpcL1wvY2RuLmNhc2Jpbi5jb21cL3NOYXRpY1wvZmF2aWNvbi5pY28iLCJjcmVhdGVkVGltZSI6IjIwMjEtMTItMTRU
  },
  "origin": "119.[redacted]8",
  "url": "http:/[redacted]et"
}
```

PHP

禅道

在禅道中使用 Casdoor 进行身份验证

ShowDoc

在 ShowDoc 中使用 Casdoor 为 oAuth2 服务器

Flarum

Using OAuth2 to connect various applications, like Flarum

禅道

禅道 是一个 敏捷(scrum) 项目管理系统/工具，但它不支持 OIDC 本身。 为了整合禅道和 Casdoor SSO，我们应该通过第三方OIDC 模块 zentao-oidc, 和这个文档将显示你如何去做。

步骤1. 部署Casdoor和禅道

首先，应该部署 Casdoor 和 禅道。 在成功部署后，您需要确保:

1. Casdoor 可以正常登录使用。
2. 您可以成功登录并使用禅道

步骤2. 集成禅道 OIDC 第三方模块

安装 zentao-oidc

```
git clone https://github.com/casdoor/zentao-oidc.git
```

或者您可以下载 ZIP 并解压缩它。

此模块用于在 OpenID 与 SSO 集成时使用 禅道 用法如下:

1. 将整个oidc目录复制到禅道模块中，并使用它作为禅道模块。 重命名下载的软件包为“oidc”

2. 配置过滤器

因为禅道框架过滤了URL中的参数，不允许空格。 所以你需要在 `/config/my.php` 末尾放置以下代码。

```
$filter->oidc=new stdclass();
$filter->oidc->index=new stdclass();
$filter->oidc->index->paramValue['scope']='reg::any';
```

3. 修改 `/module/commom/model.php`

   将“oidc”放在匿名访问列表中，并添加一行到 `isOpenMethody` 方法 `model.php` 中。

```
public function isOpenMethod($module, $method){
   if($module == 'oidc' and $method == 'index')  return true;
}
```

4. 如果您不想显示禅道登录屏幕，直接进入Casto登录屏幕。 修改最后一行代码在 `public function checkPriv()` 在`/module/common/model.php` 中。

```
//return print(js::locate(helper::createLink('user', 'login',
"referer=$referer")));
return print(js::locate(helper::createLink('oidc', 'index',
"referer=$referer")));
```

5. 修改 `setSuperVars()` 方法在 `framework/base/router.class.php`, 注释掉以下代码

```
    public function setSuperVars()
//  unset($_REQUEST);
```

步骤3. 配置Casdoor应用程序

1. 创建或使用现有的 Casdoor 应用程序。
2. 添加您的重定向url

3. 添加您想要的提供商并补充其他设置。

步骤4. 配置 Jenkins

在 oidc 中配置 `config.php`

```
$config->oidc->clientId="<Your ClientId>";
$config->oidc->clientSecret="<Your ClientSecrect>";
$config->oidc->issuer="http://localhost:8000";
```

在 module/oidc `公共函数索引()` 中设置你的重定向Url

```
$oidc->setRedirectURL($path."/zentao/oidc-index.html");
```

备注

这里的URL是指调用 'index' 方法在 'oidc ' 模块中。 您还需要设置一个变量分隔符。 哪个框架默认为破折号：- 请参阅禅道的官方框架以了解详情。 "禅道框架"

ShowDoc

在 ShowDoc 中使用 Casdoor 进行身份验证

ShowDoc 是一个 IT 团队在线API文档、技术文档工具。 Showdoc 可以轻松使用 Markdown 语法来写美丽的 API 文档、数据字典文档、技术文档、在线Excel文档等等。

Showdoc 支持第三方身份验证，包括Oauth。 以下是操作教程。

第1步： 创建一个Casdoor应用程序

转到您的 Casdoor 并添加您的新应用程序 **ShowDoc**。 下面是在Casdoor中创建ShowDoc应用程序的**示例**。

请记住下一步的`client ID`和`client Secret`。

:::信息

请在此步骤中不要填写 **callback url** Url取决于下一步showdoc 的配置。 稍后我们将返回来设置一个正确的回调URL。

:::

第2步： 配置ShowDoc

首先，启动 oAuth2 登录按钮。 然后按照示例填写 `callback url` 填写上一步中记住的 `client ID`和`client secret`。

需要`Authorize path`、`AccessToken path`、`User info path`。 您可以填写如下所示。

```
Authorize path:   /login/oauth/authorize
AccessToken path:   /api/login/oauth/access_token
User info path:   /api/get-account
```

第3步： 在casdoor中配置回调url

返回步骤1中的应用程序编辑页面并添加您填写在ShowDoc中的 `callback url`。

第4步： 在ShowDoc上试试

您应该在登录页面中看到这一点：

恭喜您！ 您已完成所有步骤 按 "castor sso" 按钮，您将被重定向到casdoor登录页面。

Flarum

Casdoor can use OAuth2 to connect various applications. Here we will use Flarum as an example to show you how to use OAuth2 to connect to your applications.

The following are some of the names in the configuration:

`CASDOOR_HOSTNAME`: Domain name or IP where Casdoor server is deployed.

`Flarum_HOSTNAME`: Domain name or IP where Flarum is deployed.

Step1. Deploy Casdoor and Flarum

Firstly, the Casdoor and Flarum should be deployed.

After a successful deployment, you need to ensure:

1. Download the Flarum plugin FoF Passport
2. Casdoor can be logged in and used normally.
3. You can set CASDOOR_HOSTNAME = `http://localhost:8000`. When deploy Casdoor in `prod` mode. See production mode.

Step2. Configure Casdoor application

1. Create or use an existing Casdoor application.
2. Find a redirect url: `<CASDOOR_HOSTNAME>/auth/passport`
3. Add your redirect url to casdoor application:

Not surprisingly, you can get two values on the application settings page: `Client ID` and `Client secret` like the picture above, we will use them in the next step.

Open your favorite browser and visit: **http://`CASDOOR_HOSTNAME`/.well-known/openid-configuration**, you will see the OIDC configure of Casdoor.

Step3. Configure Flarum

1. You should install a plugin FoF Passport
2. You should config this app

3. You can find Client Id and Client Secret in Casdoor application page.

- `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`

Log out of Flarum, and test SSO.

Log In
Cardoor SSO
Username or Email
Password
Remember Me
Log In
Forgot password?
Don't have an account? Sign Up

Ruby

GitLab

在 GitLab 服务器中使用 Casdoor 进行身份验证

GitLab

Casdoor 可以使用 OIDC 协议链接到私有部署的 GitLab，该文档将向您展示如何处理相关问题。

> **注意事项**
>
> As GitLab docs said, GitLab only works with OpenID providers that use HTTPS, so you need to deploy Casdoor with HTTPS, like putting Casdoor behind a NGINX reverse proxy with SSL certificate setup. Casdoor itself only listens to 8000 port by default via HTTP and has no HTTPS related functionality.

The following are some of the names in the configuration:

`CASDOOR_HOSTNAME`: Domain name or IP where Casdoor server is deployed. e.g., `https://door.casbin.com`.

`GITLAB_HOSTNAME`: Domain name or IP where GitLab is deployed. e.g., `https://gitlab.com`.

第 1 步： 部署 Casdoor 和 GitLab

Firstly, the Casdoor and GitLab should be deployed.

After a successful deployment, you need to ensure:

1. Casdoor 可以正常登录使用。
2. 将 Casdoor 的 `origin` 值 (conf/app.conf) 设置为 `CASDOOR_HOSTNAME`。

```
conf > app.conf
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

第 2 步： 配置 Casdoor 应用程序

1. 创建或使用现有的 Casdoor 应用程序。
2. 添加重定向网址：`http://GITLAB_HOSTNAME/users/auth/openid_connect/callback`。
3. 添加您想要的提供商并补充其他设置。

Not surprisingly, you can get two values on the application settings page: `Client ID` and `Client secret` like the picture above, and we will use them in the next

step.

Open your favorite browser and visit: **http://`CASDOOR_HOSTNAME`/.well-known/openid-configuration**, you will see the OIDC configure of Casdoor.

第 3 步： 配置 GitLab

You can follow the steps below to set this up, or make custom changes according to this document(e.g., you are installing GitLab using source code rather than 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 链接应包括 http 或 https)

```
gitlab_rails['omniauth_providers'] = [
    {
        name: "openid_connect",
        label: "Casdoor", # 登录按钮的可选标签，默认为"Openid
Connect"
        args: {
            name: "openid_connect",
            scope: ["openid", "profile", "email"],
            response_type: "code",
            issuer:  "<CASDOOR_HOSTNAME>",
            client_auth_method: "query",
            discovery: true,
            uid_field: "preferred_username",
            client_options: {
                identifier: "<你的 CLIENT ID>",
                secret: "<你的 CLIENT SECRET>",
```

3. 重新启动 GitLab 服务器。

4. 每个注册用户都可以打开`GITLAB_HOSTNAME`/-/profile/account，连接 Casdoor 账号。

5. 完成！ 现在，您可以通过 casdoor 登录您自己的 GitLab。

国际化

Casdoor支持多种语言。 通过 Crowdin，我们支持中文、法文、德文、俄文、日文和朝鲜文。

Casdoor使用官方的 Crowdin cli 来同步Crowdin 的翻译。 如果您想要添加更多语言支持，请在 我们的社区中提出， 如果您想帮助我们加快翻译工作，可通过 Crowdin帮助我们翻译。

为 Casdoor 做贡献

欢迎使用 Casdoor！ 此文档是介绍如何为 Casdoor 做出贡献的指南

如果您发现有错误或缺失之处，欢迎留下意见或建议

开始参与

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！ 使用 Casdoor 时，可以提出遇到的问题以促进Casdoor的开发，不管是漏洞还是提议都很欢迎。 Before filing an issue on GitHub, it would be better to discuss first on Discord or GitHub Discussions

> **信息**
>
> 创建issue时，请使用英文详细描述您的问题。

帮助编写文档！ 从文档开始贡献是一个很好的选择。

帮助解决issue！ 我们准备了一个表格，其中包含适合初学者的简单任务，挑战程度不同带有不同标签的标签，请查看此处的表格Casdoor Easy Tasks。

贡献

现在，如果您已经准备好创建 PR，在这里是贡献者的工作流：

1. Fork到您自己的仓库

2. 克隆您的fork到本地仓库
3. 创建一个新分支并在其上工作
4. 保持您的分支同步
5. 提交您的更改(请确保您的提交信息简洁)
6. 将你的提交推送到你的fork仓库中
7. 创建从您的分支到我们的master分支的合并请求。

合并请求

开始之前

Casdoor使用GitHub 作为其开发平台。 因此，合并请求是贡献的主要来源。

在您打开拉取请求之前，您需要知道一些基本准则：

- 当你第一次拉取请求时，你需要签名 CLA。
- 解释您为什么要发送此 PR 以及此 PR 将会在仓库中做些什么。
- **只允许一次提交**。 请确保每个合并请求只做一件事，否则请分开提交。
- 如果有新添加的文件，请将新文件顶部的 Casdoor 许可证包括在内。

```
// Copyright 2022 The Casdoor Authors. All Rights Reserved.
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
```

Semantic PRs

您的合并请求应遵循常规承诺的样本。 基本要求是有PR 标题或至少一个 提交消息。 例如，三个常用的PR标题如下：

> ⚠ **注意事项**
>
> PR 标题必须是小写的。

1. **fix**: a commit of the *type* `fix` patches a bug in your codebase.

```
fix: prevent racing of requests
```

2. **feat**: a commit of the *type* `feat` introduces a new feature to the codebase.

```
feat: allow provided config object to extend other configs
```

3. **docs**: a commit of the *type* `docs` add or improve a document.

```
docs: correct spelling of CHANGELOG
```

欲了解更多详情，请参阅 常规承诺

将 PR 链接到问题 (如果存在)

您可以将拉取请求链接到议题，以显示修复正在进行中，并在拉取请求被合并时自动关闭该议题。

使用关键词将拉取请求链接到议题

您可以通过在拉取请求说明或提交消息中使用支持的关键词将拉取请求链接到议题。 拉取请求**必须**在默认分支上。

- close
- fix
- resolve

同一仓库中的问题，例如：

```
Fix: #902
```

欲了解更多详情，请参阅 Link PR to issue。

修改PRs

您的 PR 可能不可避免地需要修改。 当代码需要更改时，请 **重新使用同一PR**。 不要关闭 PR 并创建新的 PR

下面是一个示例。

- 修改本地代码。
- 修改此提交。

```
git commit --amend
```

- 推送代码到远程仓库

```
git push --force
```

然后，PR 已成功修改！ 您可以在casdoor仓库中查看。

相关代码

一些原则:

可读性 - 重要代码应有充分的文件记录。 代码风格应与现有的风格一致。

命名规范

例如,`signupUrl`对于变量名字,`Signup URL`对于前端UI显示

如何更新 i18n 数据?

请注意，我们使用 Crowdin 作为翻译平台和i18next作为翻译工具。 当您在`web/`目录中使用i18next添加一些单词时，您可以运行`i18n/generate_test.go` 自动生成`web/src/locales/**/data.json`。

Run `i18n/generate_test.go`:

```
cd i18n && go test
```

默认情况下，所有语言都以英文填写。 鼓励您在您的 PR 已被合并后用 Crowdin帮助翻译在`web/src/locales/zh/data.json`新添加的字符串

> ⚠ **注意事项**
>
> 如果您不熟悉其他语言，请不要翻译。 保持文件原样。

许可证书

为Casdoor作出贡献，即代表您同意您的贡献将遵循 Apache 许可协议。