



Overview

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

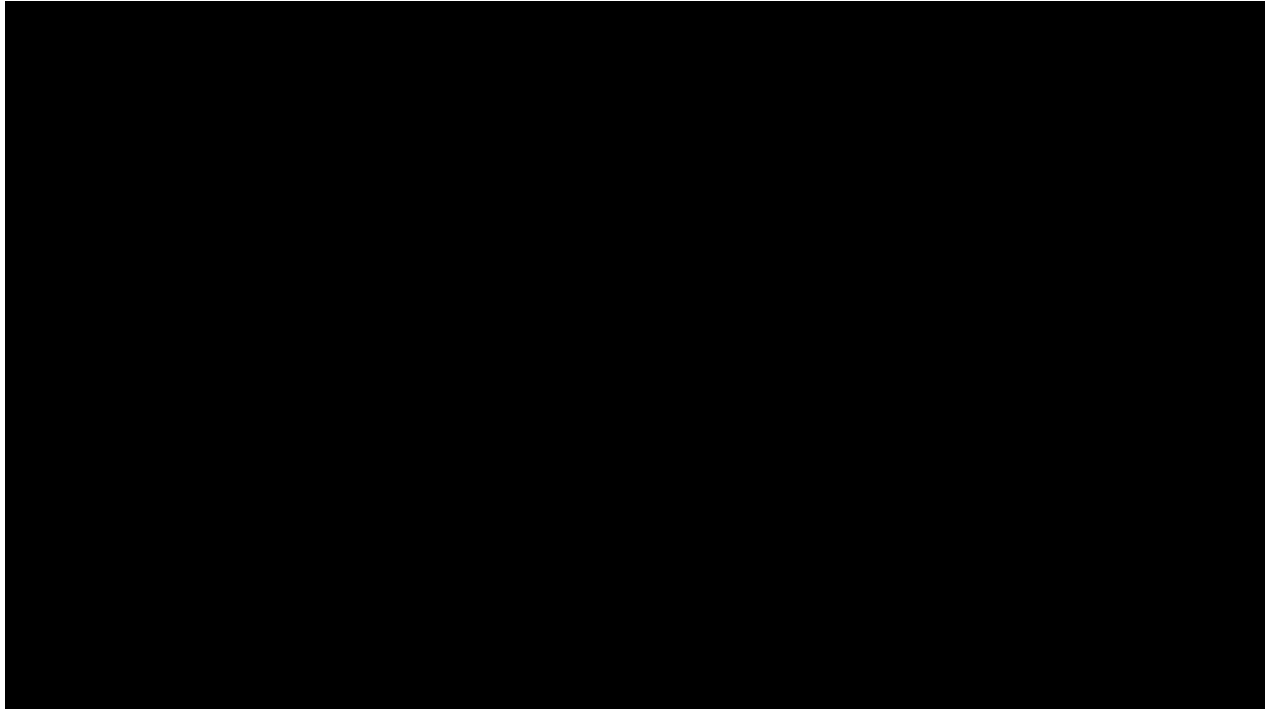
You need to enable JavaScript to run this app.

Casdoor serves both the web UI and the login requests from application users.

Casdoor features

1. Casdoor follows a front-end and back-end separate architecture, developed by Golang. It supports high concurrency, provides a web-based UI for management, and supports localization in 10+ languages.
2. Casdoor supports third-party application login, such as GitHub, Google, QQ, and WeChat, and it supports extending third-party login with plugins.
3. Casdoor supports authorization management based on [Casbin](#). It supports ACL, RBAC, ABAC, and RESTful access control models.
4. Casdoor provides phone verification code, email verification code, and password retrieval functions.
5. Casdoor supports auditing and recording of accessing logs.
6. Casdoor integrates with Alibaba Cloud, Tencent Cloud, and Qiniu Cloud image CDN cloud storage.
7. Casdoor allows customization of registration, login, and password retrieval pages.
8. Casdoor supports integration with existing systems by database synchronization, enabling smooth transition to Casdoor.
9. Casdoor supports mainstream databases such as MySQL, PostgreSQL, and SQL Server, and it supports the extension of new databases with plugins.

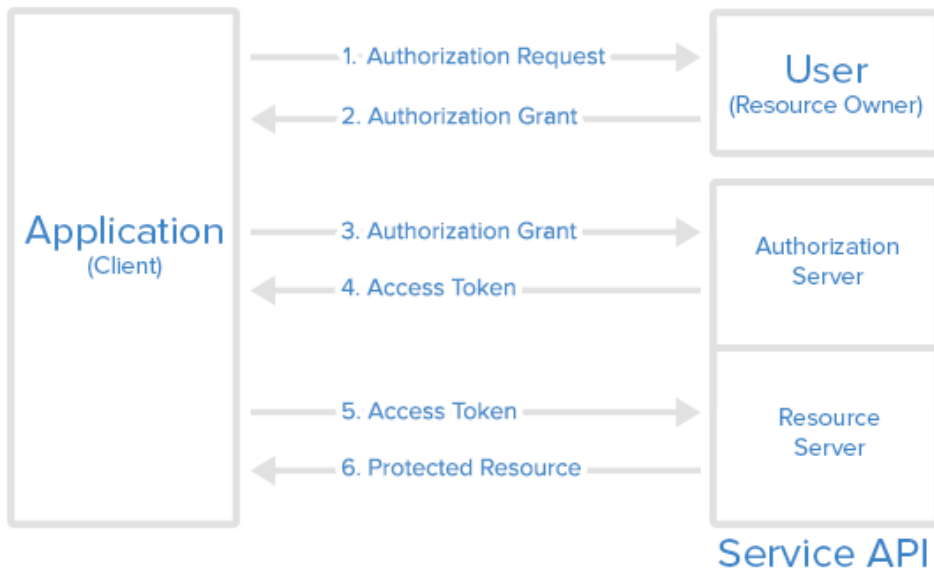
How it works



Step 0 (Pre-knowledge)

1. Casdoor follows the authorization process built upon the OAuth 2.0 protocol. It is highly recommended to have a brief understanding of how OAuth 2.0 works. You can refer to this [introduction](#) to OAuth 2.0.

Abstract Protocol Flow



Step 1 (Authorization Request)

Your Application (which could be a website or any other application) should compose a URL in the following format: `endpoint/login/oauth/authorize?client_id=xxx&response_type=code&redirect_uri=xxx&scope=read&state=xxx`. Replace `endpoint` with your Casdoor's host URL and `xxx` with your own information.

i HINTS

How to fill out the `xxx` parts?

- For `client_id`: you can find this under each individual Application
- For `redirect_uri`: you should set this to your own Application's callback URL. Casdoor will use this information to send the response back after authorization.
- For `state`: you should fill this out with your Application name.

The Application will prompt the user: *"Hey, I need some resources and I need your permission to access these resources. Can you go to this URL and enter your username and password for me?"*

With the correctly composed URL, your Application will make the user launch a request to this URL, and the `Authorization Request` is completed.

Step 2 (Authorization Grant)

This step is straightforward: the user is redirected to the URL composed in Step 1, and the user will see the login page from Casdoor. By entering the correct username and credentials into the login page, Casdoor now knows the identity of the user and is about to send two pieces of information back to the callback URL set in Step 1: `code` and `state`.

The user opens the URL and provides the credentials to Casdoor. Casdoor will say: "Looking good ~ this is the user (who is authorizing the Application to get the `code` and `state`) I know in my database, and I will send the `code` and `state` back to the Application using the callback URL (`redirect_uri`)".

With these two pieces of information sent back to your Application, the authorization is granted to the app, and the `Authorization Grant` is completed.



TIP

Casdoor also provides third-party logins. In this case, instead of seeing the credential entry page, you will see a list of third-party providers. You can log in to your app using these providers, with Casdoor acting as a middle layer (middleware).

Step 3 (Authorization Grant)

In this step, your Application already has the code from Step 2, and it will speak to Casdoor: "Hey, the user agreed to give me the `code`. Can you verify this `code` and give me the `access_token`?"

Step 4 (Access Token)

Casdoor responds to your Application: "You know what, this `code` seems legit. You must be the right Application. Here's the `access_token` for you." With this `code`, Casdoor confirms that it is an authorized Application (authorized by the correct user in Step 2) trying to obtain the `access_token` (which will be used later to access more resources).

Step 5 (Access Token)

In this step, your Application says: "Nice! I just got the fresh-and-tasty `access_token`. Now I can use it to access something more valuable from the `Resource Server`!"

Your Application then turns to the `Resource Server` and says: "Hey buddy, can you check out this `access_token`? I received it from Casdoor. Do you want to verify if this is the correct token you issued to Casdoor?"

Step 6 (Protected Resource)

The `Resource Server` responds to your Application: "Not bad. It seems just like the one I issued to Casdoor, and Casdoor says whoever holds this `access_token` can access these `Protected Resources`. So go ahead and take it!"

And that's basically how Casdoor works with your Application.

HINT

Casdoor can act as both an `Authorization Server` and a `Resource Server`. In other words, Casdoor authorizes your Application to access resources, usually the currently logged-in user's information, from Casdoor's database.

Online demo

Casdoor

Here is an online demo deployed by Casbin.

- [Casdoor official demo](#)

Global admin login:

- Username: `admin`
- Password: `123`

Casbin-OA

Casbin-OA is one of the Casbin web apps. It uses Casdoor for authentication.

- [Casbin-OA](#)
- Source code: <https://github.com/casbin/casbin-oa>

Casnode

Casnode is the official forum developed by the Casbin community.

It uses Casdoor as the authentication platform and manages members.

- [Casnode](#)
- Source code: <https://github.com/casbin/casnode>

Architecture

Casdoor consists of two parts:

Name	Description	Language	Source code
Frontend	Web frontend UI for Casdoor	JavaScript + React	https://github.com/casdoor/casdoor/tree/master/web
Backend	RESTful API backend for Casdoor	Golang + Beego + SQL	https://github.com/casdoor/casdoor

Core Concepts

As a Casdoor administrator, you should be familiar with at least four core concepts: `Organization`, `User`, `Application`, and `Provider`.

TIP

In the following parts, we will use the demo site <https://door.casdoor.com> as an example.

Organization

In Casdoor, an organization is a container for users and applications. For example, all the employees of a company or all the customers of a business can be abstracted as one organization. The `Organization` class definition is shown below:

```
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:"createTime"`

    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"`
}
```

User

In Casdoor, a user can log into an application. Each user can belong to only one organization but can log into multiple applications owned by the organization. Currently, there are two types of users in Casdoor:

- `built-in` organization users, such as `built-in/admin`: global administrators who have full administrative power on the Casdoor platform.
- Other organizations' users, such as `my-company/alice`: normal users who can sign up, sign in, sign out, change their own profile, etc.

In the Casdoor API, a user is typically identified as `<organization_name>/<username>`. For example, the default administrator of Casdoor is denoted as `built-in/admin`. Additionally, the `User` class definition includes an `id` property, which is a UUID like `d835a48f-2e88-4c1f-b907-60ac6b6c1b40` and can be chosen as a user's ID by an application.

TIP

For applications that are only for one organization, it's possible to use `<username>` instead of `<organization_name>/<username>` as the user ID across the application for simplicity.

Here's the `User` class definition:

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


Application

An **application** represents a web service that needs to be protected by Casdoor, such as a forum site, an OA system, or a CRM system.

```
type Application struct {
  Owner          string          `xorm:"varchar(100) notnull pk" json:"owner"`
  Name           string          `xorm:"varchar(100) notnull pk" json:"name"`
  CreatedTime    string          `xorm:"varchar(100)" json:"createdTime"`
  DisplayName    string          `xorm:"varchar(100)" json:"displayName"`
  Logo           string          `xorm:"varchar(100)" json:"logo"`
  HomepageUrl    string          `xorm:"varchar(100)" json:"homepageUrl"`
  Description    string          `xorm:"varchar(100)" json:"description"`
  Organization    string          `xorm:"varchar(100)" json:"organization"`
  Cert           string          `xorm:"varchar(100)" json:"cert"`
  EnablePassword bool            `json:"enablePassword"`
  EnableSignUp   bool            `json:"enableSignUp"`
  EnableSignInSession bool          `json:"enableSignInSession"`
  EnableCodeSignIn bool          `json:"enableCodeSignIn"`
  Providers      []*ProviderItem `xorm:"mediumtext" json:"providers"`
  SignupItems    []*SignupItem  `xorm:"varchar(1000)" json:"signupItems"`
  OrganizationObj *Organization  `xorm:"- " json:"organizationObj"`
  ClientId       string         `xorm:"varchar(100)" json:"clientId"`
  ClientSecret   string         `xorm:"varchar(100)" json:"clientSecret"`
  RedirectUri    []string       `xorm:"varchar(1000)" json:"redirectUri"`
  TokenFormat    string         `xorm:"varchar(100)" json:"tokenFormat"`
  ExpireInHours  int            `json:"expireInHours"`
  RefreshExpireInHours int          `json:"refreshExpireInHours"`
  SignupUrl      string         `xorm:"varchar(200)" json:"signupUrl"`
  SignInUrl     string         `xorm:"varchar(200)" json:"signInUrl"`
  ForgetUrl     string         `xorm:"varchar(200)" json:"forgetUrl"`
  AffiliationUrl string         `xorm:"varchar(100)" json:"affiliationUrl"`
  TermsOfUse    string         `xorm:"varchar(100)" json:"termsOfUse"`
  SignupHtml    string         `xorm:"mediumtext" json:"signupHtml"`
  SignInHtml    string         `xorm:"mediumtext" json:"signInHtml"`
}
```

Each application can have its own customized sign-up page, sign-in page, and more. The root login page `/login` (e.g., <https://door.casdoor.com/login>) is the sign-in page only for Casdoor's built-in application: `app-built-in`.

An application is a "portal" or "interface" for a user to log into Casdoor. A user must go through one application's sign-in page to log into Casdoor.

Application	Sign-up page URL	Sign-in page URL
app-built-in	https://door.casdoor.com/signup	https://door.casdoor.com/login
app-casnode	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
app-casbin-oa	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

Login URLs

It's very easy to log into Casdoor via Casdoor's built-in application; simply visit Casdoor server homepage (e.g., <https://door.casdoor.com>) for

demo site) and it will automatically redirect you to `/login`. But how do you get the URLs for other applications in frontend and backend code? You can either concatenate strings manually or call some utility functions provided by Casdoor SDKs to get the URLs:

1. Manually concatenating strings

- Sign-up page URL
 - Signup for the specified application: `<your-casdoor-hostname>/signup/<your-application-name>`
 - Signup by OAuth: `<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`
 - Signup automatically: `<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`
- Sign-in page URL
 - Sign-in for the specified organization: `<your-casdoor-hostname>/login/<your-organization-name>`
 - Sign-in by OAuth: `<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. Using frontend SDK (for frontend JavaScript code using React, Vue, or Angular)

`getSignupUrl()` and `getSignInUrl()`: [casdoor-js-sdk](#)

3. Using backend SDK (for backend code using Go, Java, etc.)

`GetSignupUrl()` and `GetSignInUrl()`: [casdoor-go-sdk](#)

Provider

Casdoor is a federated single sign-on system that supports multiple identity providers via OIDC, OAuth, and SAML. Casdoor can also send verification codes or other notifications to users via email or SMS. Casdoor uses the concept of `Provider` to manage all these third-party connectors.

A list of all providers supported by Casdoor can be found at [provider/overview](#).

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

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

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

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

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

How does Casdoor manage itself?

Upon running Casdoor for the first time, some built-in objects are created to facilitate its management:

- A built-in organization named `built-in`.
- A user named `admin` in the `built-in` organization.
- A built-in application named `app-built-in`, administered by the `built-in` organization, representing Casdoor itself.

All users under the `built-in` organization, including `admin`, will have full administrator privileges on the Casdoor platform. Therefore, if there are multiple administrators, it is advisable to create new accounts under the `built-in` organization. Alternatively, the sign-up channel for the `app-built-in` application should be closed to prevent unwanted access.

CAUTION

It is not possible to rename or delete the built-in objects via both the web UI or the RESTful API. Casdoor has hardcoded these reserved names in many places; attempting to rename or delete them by modifying the DB may cause the entire system to crash.

Server Installation

Requirements

Operating System

All major operating systems, including Windows, Linux, and macOS, are supported.

Environment

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

INFO

We strongly suggest using [Yarn 1.x](#) to run and build Casdoor frontend. Using NPM might cause UI styling issues. For more details, see: [casdoor#294](#).

CAUTION

If your network fails to directly sync the Go dependency packages successfully, you need to use a Go proxy by configuring the GOPROXY environment variable. We strongly recommend using: <https://goproxy.cn/>

Database

Casdoor uses [XORM](#) to communicate with the database. Based on [Xorm Drivers Support](#), Casdoor currently provides support for the following databases:

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

Download

The source code of Casdoor is hosted on GitHub: <https://github.com/casdoor/casdoor>. Both the Go backend code and React frontend code are contained in a single repository.

Name	Description	Language	Source code
Frontend	Web frontend UI for Casdoor	JavaScript + React	https://github.com/casdoor/casdoor/tree/master/web
Backend	RESTful API backend for	Golang + Beego +	https://github.com/casdoor/casdoor

Name	Description	Language	Source code
	Casdoor	XORM	

Casdoor supports `Go Modules`. To download the code, simply clone the code using git:

```
cd path/to/folder
git clone https://github.com/casdoor/casdoor
```

Configuration

Configure Database

Casdoor supports MySQL, MSSQL, SQLite3, and PostgreSQL. By default, Casdoor uses MySQL.

MySQL

Casdoor stores user, node, and topic information in a MySQL database named `casdoor`. If the database does not exist, it must be created manually. The DB connection string can be specified at: <https://github.com/casdoor/casdoor/blob/master/conf/app.conf>

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

PostgreSQL

Before running Casdoor, you need to manually prepare a database for PostgreSQL, as Casdoor requires selecting a database when opening Postgres with xorm.

Assuming you have already prepared a database called `casdoor`, you should specify `app.conf` like this:

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

! INFO

For PostgreSQL, ensure that `dataSourceName` has a non-empty `dbName` and also duplicate the database name for the `dbname` field as shown in the example above.

CockroachDB

CockroachDB can also be used with the PostgreSQL driver and has the same configuration as PostgreSQL.

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


! INFO

For CockroachDB, remember to add

`serial_normalization=virtual_sequence` to the `dataSourceName` as shown in the example above. Otherwise, you will get an error regarding an existing database whenever the service starts or restarts. Note that this must be added before the database is created.

SQLite3

To configure SQLite3, you should specify `app.conf` like this:

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

Via Ini file

Casdoor can be configured via a single file: [conf/app.conf](#), which by default contains the following content:

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

- `appName` is the application name, which currently has no practical use.
- `httpPort` is the port that your backend application is listening on.
- `runmode` can be set to `dev` or `prod`.
- `SessionOn` determines whether to enable session and is enabled by default.
- `driverName`, `dataSourceName`, and `dbName` were introduced earlier. Please see [Configure Database](#) for details.
- `verificationCodeTimeout` sets the expiration time of the verification code. After expiration, the user needs to obtain it again.

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, and applications.

- `tableNamePrefix` is the prefix of the table when using an adapter.
- `showSql` determines whether to show SQL statements on the logger if the log level is greater than INFO.
- `redisEndpoint` is the Redis endpoint used by Beego session storage. If this parameter is empty, the session data will be stored locally as files in the `./tmp` folder. To use Redis as Beego session storage, the value would be something like: `redis.example.com:6379`. If Redis is deployed locally, you can use `localhost:6379`. If Redis password is enabled, use `redis.example.com:6379,db,password`. See more details at: <https://github.com/beego/beedoc/blob/master/en-US/module/session.md#saving-provider-config>.
- `defaultStorageProvider` is the default file storage service name. If you need to use file storage services such as avatar upload, you need to set up a storage provider and apply it in your application. See [storage](#) for details.
- `isCloudIntranet` is used to identify whether your provider endpoint is an intranet endpoint.
- `authState` is the authorization application name. This parameter will be

checked when logging in.

- `socks5Proxy` is the SOCKS proxy server IP address. Set the proxy port because we have Google-related services or use `Google`, `GitHub`, `Facebook`, `LinkedIn`, or `Steam` as OAuth Providers, which may be restricted by the network in some areas.
- `initScore` is the initial score of each user. Each user has a score attribute. The score is used by `Casnode` and does not control anything in Casdoor.
- `logPostOnly` is used to determine whether only the post method is used to add a record.
- `origin` is the origin backend domain name.
- `staticBaseUrl` is the address of the static image used when the system initializes the database.
- `enableGzip` will accept and respond with gzip encoding if the request header includes `Accept-Encoding=gzip`.

Via Environment Variables

All configuration items defined by Casdoor in the ini file mentioned above can be configured via environment variables, as well as 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 environment variables:

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

In addition, `export` derivatives are also a possible method. The names of environmental variables should exactly match the names you want to use in the ini file.

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

Run

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

Development Mode

Backend

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

```
go run main.go
```

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

Frontend

Casdoor's frontend is a very classic [Create-React-App \(CRA\)](#) project. It runs on 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 the Casdoor dashboard with the default global admin account: `built-in/admin`.

```
admin  
123
```

Production Mode

Backend

Build the Casdoor Go backend code into an executable and start it.

For Linux:

```
go build  
./casdoor
```

For Windows:

```
go build  
casdoor.exe
```

Frontend

Build the 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 the Casdoor dashboard with the default global admin account: `built-in/admin`.

admin
123



TIP

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

ⓘ CASDOOR PORT DETAILS

In the `dev` environment, the frontend is run by `yarn run` on port 7001, so if you want to go to the Casdoor login page, you need to set the Casdoor link as <http://localhost:7001>.

In the `prod` environment, the frontend files are first built by `yarn build` and served on port 8000, so if you want to go to the Casdoor login page, you need to set the Casdoor link as <https://your-casdoor-url.com:8000> (if you are using a 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 the `dev` environment, we set the `serverUrl` as <http://localhost:7001>, so when we test the signin and signup functionality using Casdoor, it will go to localhost 7001, which is the Casdoor port.

And when we put Casnode into the `prod` environment, we set the `serverUrl` as <https://door.casdoor.com>, so users can sign in or sign up using Casdoor.

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

(Optional) Try with Docker

Requirements

Hardware

If you want to build the Docker image yourself, please ensure that your machine has at least 2GB of memory. Casdoor's frontend is an NPM project of React. Building the frontend requires at least 2GB of memory. Having less than 2GB of memory may result in a frontend build failure.

If you only need to run the pre-built image, please ensure that your machine has at least 100MB of memory.

OS

All operating systems (Linux, Windows, and macOS) are supported.

Docker

You can use Docker (docker-engine version ≥ 17.05) in Linux or Docker Desktop in Windows and macOS.

- [Docker](#)

Regardless of the operating system, users must ensure that they have docker-engine version ≥ 17.05 . This is because we utilize the multi-stage build feature in the docker-compose.yml, which is supported in versions 17.05 and above. For more information, see <https://docs.docker.com/develop/develop-images/multistage-build/>.

If you are also using docker-compose, please ensure that you have **docker-compose version >= 2.2**. For Linux users, you also need to make sure that docker-compose is installed, as it is separate from docker-engine.

Get the image

We have provided two DockerHub images:

Name	Description	Suggestion
casdoor-all-in-one	Both Casdoor and a MySQL database are included in the image	This image already includes a toy database and is only for testing purposes
casdoor	Only Casdoor is included in the image	This image can be connected to your own database and used in production

1. [casbin/casdoor-all-in-one](#): This image includes the casdoor binary, a MySQL database, and all the necessary configurations. It is designed for new users who want to try Casdoor quickly. With this image, you can start Casdoor immediately with just one or two commands, without any complex configuration. However, please note that we **do not recommend** using this image in a production environment.

Option-1: Use the toy database

Run the container with port `8000` exposed to the host. The image will be automatically pulled if it doesn't exist on the local host.

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

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

```
admin  
123
```

Option-2: Try directly with the standard image



TIP

If it is not convenient to mount the configuration file to a container, using environment variables is also a possible solution.

example

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

Create `conf/app.conf`. You can copy it from [conf/app.conf](#) in Casdoor. For more details about `app.conf`, you can see [Via Ini file](#).

Then run

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

Anyway, just mount the `app.conf` to `/conf/app.conf` and start the container.

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

```
admin  
123
```

Option-3: Try with docker-compose

Create a `conf/app.conf` directory in the same directory level as the `docker-compose.yml` file. Then, copy `app.conf` from Casdoor. For more details about `app.conf`, you can see [Via Ini file](#).

Create a separate database using docker-compose:

```
docker-compose up
```

That's it! ✈️

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

```
admin  
123
```

i NOTE

If you dig deeper into the docker-compose.yml file, you may be puzzled by the environment variable we created called "RUNNING_IN_DOCKER". When the database 'db' is created via docker-compose, it is available on your PC's localhost but not the localhost of the Casdoor container. To prevent you from running into troubles caused by modifying app.conf, which can be quite difficult for a new user, we provided this environment variable and pre-assigned it in the docker-compose.yml. When this environment variable is set to true, localhost will be replaced with host.docker.internal so that Casdoor can access the database.

(Optional) Try with K8s Helm

Introduction

Now we show how to deploy Casdoor on Kubernetes using Helm for easy and scalable management.

Prerequisites

- A running Kubernetes cluster
- Helm v3 installed

Installation Steps

Step 1: Install the Casdoor Chart

Install the Casdoor chart:

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

Step 2: Accessing Casdoor

Once installed, Casdoor can be accessed at the provided service URL by your Kubernetes cluster.

Customization and Configuration

Customize your Casdoor installation by modifying the Helm chart values. For detailed options, refer to the [values.yaml](#) file in the chart. The following parameters can be configured.

Parameter	Description	Default Value
<code>replicaCount</code>	Number of replicas of the Casdoor application to run.	1
<code>image.repository</code>	Repository for the Casdoor Docker	<code>casbin</code>

Parameter	Description	Default Value
	image.	
<code>image.name</code>	Name of the Casdoor Docker image.	<code>casdoor</code>
<code>image.pullPolicy</code>	Pull policy for the Casdoor Docker image.	<code>IfNotPresent</code>
<code>image.tag</code>	Tag for the Casdoor Docker image.	<code>""</code>
<code>config</code>	Configuration settings for the Casdoor application.	See config field
<code>database.driver</code>	Database driver to use (supports mysql, postgres, cockroachdb, sqlite3).	<code>sqlite3</code>
<code>database.user</code>	Database username.	<code>""</code>
<code>database.password</code>	Database password.	<code>""</code>
<code>database.host</code>	Database host.	<code>""</code>
<code>database.port</code>	Database port.	<code>""</code>

Parameter	Description	Default Value
<code>database.databaseName</code>	Name of the database used by Casdoor.	<code>casdoor</code>
<code>database.sslMode</code>	SSL mode for the database connection.	<code>disable</code>
<code>service.type</code>	Type of Kubernetes service to create for Casdoor (ClusterIP, NodePort, LoadBalancer, etc.).	<code>ClusterIP</code>
<code>service.port</code>	Port number for the Casdoor service.	<code>8000</code>
<code>ingress.enabled</code>	Whether to enable Ingress for Casdoor.	<code>false</code>
<code>ingress.annotations</code>	Annotations for the Ingress resource.	<code>{}</code>
<code>ingress.hosts</code>	Hostnames for the Ingress resource.	<code>[]</code>
<code>resources</code>	Resource requests and	<code>{}</code>

Parameter	Description	Default Value
	limits for the Casdoor container.	
<code>autoscaling.enabled</code>	Whether to enable Horizontal Pod Autoscaler for Casdoor.	<code>false</code>
<code>autoscaling.minReplicas</code>	Minimum number of replicas for Horizontal Pod Autoscaler.	<code>1</code>
<code>autoscaling.maxReplicas</code>	Maximum number of replicas for Horizontal Pod Autoscaler.	<code>100</code>
<code>autoscaling.targetCPUUtilizationPercentage</code>	Target CPU utilization percentage for Horizontal Pod Autoscaler.	<code>80</code>
<code>nodeSelector</code>	Node labels for pod assignment.	<code>{}</code>
<code>tolerations</code>	Toleration labels for pod assignment.	<code>[]</code>
<code>affinity</code>	Affinity	<code>{}</code>

Parameter	Description	Default Value
	settings for pod assignment.	
<code>extraContainersEnabled</code>	Whether to enable additional sidecar containers.	<code>false</code>
<code>extraContainers</code>	Additional sidecar containers.	<code>""</code>
<code>extraVolumeMounts</code>	Additional volume mounts for the Casdoor container.	<code>[]</code>
<code>extraVolumes</code>	Additional volumes for the Casdoor container.	<code>[]</code>
<code>envFromSecret</code>	Provide Environment variable from secret.	<code>[{name:"", secretName:"", key:""}]</code>
<code>envFromConfigmap</code>	Provide Environment variable from configmap.	<code>[{name:"", configmapName:"", key:""}]</code>
<code>envFrom</code>	Provide Environment variable from entire secret or configmap.	<code>[{name:"", type:"configmap \\ secret"}]</code>

Managing the Deployment

To upgrade your Casdoor deployment:

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

To uninstall Casdoor:

```
helm delete casdoor
```

For further management and customization, refer to the Helm and Kubernetes documentation.

Conclusion

Using Helm to deploy Casdoor on Kubernetes simplifies the management and scalability of your authentication services within your Kubernetes environment.

Casdoor Public API

Casdoor frontend web UI is a [SPA \(Single-Page Application\)](#) developed in React. The React frontend consumes the Casdoor RESTful API exposed by the Go backend code. This RESTful API is referred to as the `Casdoor Public API`. In Another word, with HTTP calls, you can do everything just like how Casdoor web UI itself does. There's no other limitations. The API can be utilized by the following:

- Casdoor's frontend
- Casdoor client SDKs (e.g., `casdoor-go-sdk`)
- Any other customized code from the application side

The full reference for the `Casdoor Public API` can be found on Swagger: <https://door.casdoor.com/swagger>. These Swagger docs are automatically generated using Beego's Bee tool. If you want to generate the Swagger docs by yourself, see: [How to generate the swagger file](#)

How to authenticate with `Casdoor Public API`

1. By `Access token`

We can use the access token granted for an authenticated user to call `Casdoor Public API` as the user itself.

How to get the access token?

The application can get the access token for the Casdoor user at the end of OAuth login process (aka get the token by code and state). The permissions for the API calls will be the same as the user.

The below examples shows how to call `GetOAuthToken()` function in Go via `casdoor-go-sdk`.

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

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

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

All granted access tokens can also be accessed via the web UI by an admin user in the Tokens page. For example, visit: <https://door.casdoor.com/tokens> for the demo site.

How to authenticate?

1. HTTP `GET` parameter, the URL format is:

```
/page?access_token=<The access token>
```

Demo site example: https://door.casdoor.com/api/get-global-providers?access_token=eyJhbGciOiJIUzI1NiIs

2. HTTP Bearer token, the HTTP header format is:

```
Authorization: Bearer <The access token>
```

2. By `Client ID` and `Client secret`

How to get the client ID and secret?

The application edit page (e.g., <https://door.casdoor.com/applications/casbin/app-vue-python-example>) will show the client ID and secret for an application. This authentication is useful when you want to call the API as a "machine", "application" or a "service" instead of a user. The permissions for the API calls will be the same as the application (aka the admin of the organization).

The below examples shows how to call `GetOAuthToken()` function in Go via `casdoor-go-sdk`.

How to authenticate?

1. HTTP `GET` parameter, the URL format is:

```
/page?clientId=<The client ID>&clientSecret=<the client secret>
```

Demo site example: <https://door.casdoor.com/api/get-global-providers?clientId=294b09fbc17f95daf2fe&clientSecret=dd8982f7046ccba1bbd7851d5c1ece4e52bf039d>

2. [HTTP Basic Authentication](#), the HTTP header format is:

```
Authorization: Basic <The Base64 encoding of client ID and client secret joined by a single colon ":">
```

If you are not familiar with the Base64 encoding, you can use a library to do that because `HTTP Basic Authentication` is a popular standard supported by many places.

3. By Access key and Access secret

We can use the access key and access secret for a Casdoor user to call `Casdoor Public API` as the user itself. The access key and access secret can be configured in the user setting page by an admin or the user himself. The `update-user` API can also be called to update these fields. The permissions for the API calls will be the same as the user.

How to authenticate?

1. HTTP `GET` parameter, the URL format is:

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

Demo site example: `https://door.casdoor.com/api/get-global-providers?accessKey=b86db9dc-6bd7-4997-935c-af480dd2c796/admin&accessSecret=79911517-fc36-4093-b115-65a9741f6b14`

4. By username and password

CAUTION

This authentication method is not safe and kept here only for compatibility or demo purposes. We recommend using the previous three authentication methods instead.

What will happen?

The user credential will be exposed as `GET` parameters in the request URL. Moreover, the user credential will be sniffed in plain text by the network if you are using HTTP instead of HTTPS.

We can use the username and password for a Casdoor user to call `Casdoor Public API` as the user itself. The username takes the format of `<The user's organization name>/<The user name>`. The permissions for the API calls will be the same as the user.

How to authenticate?

1. HTTP `GET` parameter, the URL format is:

```
/page?username=<The user's organization name>/<The user name>&password=<the user's password>"
```

Demo site example: `https://door.casdoor.com/api/get-global-providers?username=built-in/admin&password=123`

Tutorials

Product Documentation

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

Articles

Technologies	Language	Title
ASP.NET Core 6	English	ASP.NET Core .NET 6 Demo Authentication Project using local Casdoor Docker Container on Windows Subsystem for Linux
OAuth2 Proxy (Go)	Chinese	Use Casdoor + OAuth-Proxy to protect web applications on public networks
Casnode (JavaScript + React + Go + Beego)	Chinese	Use Lighthouse to set up a forum like V2ex
Cloudfre (Go)	Chinese	Modify Cloudfre to support Casdoor
KodExplorer (PHP)	Chinese	Modify KodExplorer to support Casdoor

Deployment

Deploying to NGINX

Use Nginx to reverse proxy your backend Go program and quickly start the Casdoor service.

Deploying to Kubernetes

Learn how to deploy Casdoor in a Kubernetes cluster

Data Initialization

How to initialize Casdoor data from files

Hosting Static Files in a CDN

Hosting frontend static files in a CDN

Hosting Static Files in an Intranet

How to deploy Casdoor static resources

DB Migration

Handling DB Migration in Casdoor

Deploying to NGINX

Though Casdoor follows a front-end back-end separation architecture, in a production environment, the back-end program still provides static file services for front-end files. Hence, you can employ reverse proxy software like [Nginx](#) to proxy all traffic for the Casdoor domain and redirect it to the port monitored by the backend Go program.

In this chapter, you will learn how to use Nginx to reverse proxy your backend Go program and quickly start the Casdoor service.

1. Build front end static files

Assuming you have downloaded Casdoor and completed the necessary configuration (if not, refer to the [Get started](#) section), you only need to build the static files as follows:

[Yarn](#) `npm`

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

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

2. Run the back-end program

```
go run main.go
```

Or, build it first:

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

3. Configure and run Nginx

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

Then, add a server:

```
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;
    }
}
```

Next, restart your Nginx process. Run:

```
nginx -s reload
```

4. Test

Visit `http://YOUR_DOMAIN_NAME` in your favorite browser.

Deploying to Kubernetes

Deploy Casdoor in Kubernetes (k8s)

We provide a basic example of deploying Casdoor in a Kubernetes cluster. In the root folder of Casdoor, you will find a file named "k8s.yaml". This file contains an example configuration for deploying Casdoor in Kubernetes, including a deployment and a service.

Before starting the deployment, ensure that you have modified the `conf/app.conf` file so that Casdoor can connect to the database successfully and that the database itself is running. Also, make sure that Kubernetes is able to pull the necessary images.

To deploy Casdoor, run the following command:

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

You can check the deployment status by running the command `kubectl get pods`.

Here is the content of `k8s.yaml`:

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

Please note that this file is only an example. You can make various modifications as per your requirements, such as using a different namespace, service type, or a ConfigMap to mount the configuration file. Using a ConfigMap is a recommended approach in Kubernetes for mounting configuration files in a production environment.

Data Initialization

If you are deploying Casdoor with other services as a complete application, you may want to provide an **out-of-the-box** feature for users. This means that users can directly use the application without any configuration.

In such a situation, you can use data initialization to register your service in Casdoor through a configuration file. This file can be pre-defined or dynamically generated by your own service.

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

Import Config Data

If there is a configuration file named `init_data.json` at the root directory of Casdoor, it will be used to initialize data in Casdoor. All you have to do is place this file in the root directory where Casdoor will run.

If you are using the official Docker image of Casdoor, here are some scripts that can help you to mount `init_data.json` into the container.

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

For Docker

If you deploy Casdoor with Docker, you can use the `volume` command to mount `init_data.json` into the container.

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

For Kubernetes

If you deploy Casdoor with Kubernetes, you can use the `configmap` to store `init_data.json`.

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

You can mount the data into Casdoor `Pods` by mounting the `configmap`. You can modify your `deployment` as follows:

```
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
```

Export Config Data

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

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

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

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

References

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

Object	Go Struct	Documentation
organizations	struct	doc
applications	struct	doc
users	struct	doc
certs	struct	

Object	Go Struct	Documentation
providers	struct	doc
ldaps	struct	doc
models	struct	
permissions	struct	doc
payments	struct	doc
products	struct	doc
resources	struct	doc
roles	struct	doc
syncers	struct	doc
tokens	struct	doc
webhooks	struct	doc
groups	struct	doc
adapters	struct	doc
enforcers	struct	
plans	struct	doc

Object	Go Struct	Documentation
pricings	struct	doc
invitations	struct	doc
records	struct	
sessions	struct	
subscriptions	struct	doc
transactions	struct	

If you still feel confused about filling out this template, you can call the RESTful API or use the debug mode of your browser to see the response of `GetXXX` to these objects. The responses are in the same format as `init_data.json`.

Hosting Static Files in a CDN

Frontend static resources, such as .js and .css files, are located in `web/build/static/`. If you wish to deploy these files in a public cloud's CDN service, Casdoor provides a script that simplifies the deployment process. Please follow the steps below.

NOTE


We assume that you have already built the frontend code of Casdoor. If you have not, please refer to the [documentation](#).

Preparation

First, you need to create a valid [Storage Provider](#) in the Casdoor UI. You can refer to the [example](#).

CAUTION

When filling in the `Domain` field, be sure to end it with a '/

Domain  :

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

Usage

The script can be found at [deployment/deploy_test.go](#).

In [deploy_test.go](#), you need to modify the `id` parameter in `GetProvider()`. The format of the provider `id` is `<owner>/<name>`.

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

After making the necessary modification, use the following commands to run the script:

```
cd deployment
go test
```

If the execution is successful, you will see:

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

How it works

The script will:

- Upload all the files in the `css/` and `js/` folders to the CDN service specified

by the storage provider.

- Replace all the URLs of the `.css` and `.js` files in `web/build/index.html` with the URLs hosted in the CDN.

You still need to keep the `index.html` file. After the static files are uploaded to the CDN, `index.html` will still be requested by users through Casdoor's Go backend, and the static files in the CDN will be requested through the URLs provided in `index.html`.

Hosting Static Files in an Intranet

If you are deploying Casdoor on an intranet, you may not be able to access the static resources directly over the internet. You need to deploy the static resources where you can access them, and then modify the configuration in Casdoor in three places.

Deploy static resources

All static resources in Casdoor, including images, logos, CSS, etc., are stored in the [casbin/static repository](#).

Clone the repository and deploy it on a web server. Make sure you can access the resources.

Modify in Casdoor

You can simply modify the configuration file to set the static resource address to where you deployed it. Go to [conf/app.conf](#) and set `staticBaseUrl` to your deployed address.

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

DB Migration

When upgrading the database, there is a risk of data loss, such as when deleting an old field. Luckily, Casdoor utilizes [xorm](#), which assists with many database migration problems. However, some schema and data migrations must still be handled manually, such as when a field name is changed.

NOTE

Refer to the [xorm docs](#) for a better understanding of xorm's schema operations.

How it Works

As mentioned earlier, xorm is unable to handle field name changes. To address this, xorm provides a [migrate](#) package that can assist with this problem.

To handle field renaming, you can write code like this:

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

Our objective is to rename `p_type` to `ptype`. However, since xorm does not support field renaming, we must resort to a more intricate approach: assigning the value of `p_type` to `ptype`, and subsequently deleting the `p_type` field.

The `ID` field uniquely identifies the migration being performed. After `m.Migrate()` runs, the value of `ID` will be added to the migrations table of the database.

Upon starting the project again, the database will check for any existing `ID` field in the table and refrain from performing any operations associated with the same `ID`.

How to Connect to Casdoor

Overview

Connect your app to Casdoor

Standard OIDC Client

Using OIDC discovery to migrate to Casdoor

Casdoor SDKs

Using Casdoor SDKs instead of standard OIDC protocol

How to Enable Single Sign-On

Enable Single Sign-On

Vue SDK

Casdoor Vue SDK

Desktop SDKs

4 items

Mobile SDKs

1 items

Casdoor Plugin

Using Casdoor plugins or middlewares in other frameworks like Spring Boot, WordPress, Odoo, etc.

Next.js

Using Casdoor in a Next.js project

Nuxt

Using Casdoor in a Nuxt project

OAuth 2.0

Using Access Token to authenticate clients

Using Casdoor as a CAS Server

How to use Casdoor as a CAS server

SAML

6 items

Face ID

Use Face ID to log in in Casdoor

WebAuthn

Use WebAuthn in Casdoor

Overview

In this section, we will show you how to connect your application to Casdoor.

As a Service Provider (SP), Casdoor supports two authentication protocols:

- OAuth 2.0 (OIDC)
- SAML

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

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

OAuth 2.0 (OIDC)

What is OAuth 2.0?

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

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

recommend using the OAuth 2.0 protocol for the following reasons:

1. The protocol is simple and easy to implement, and can solve many scenarios.
2. It has a high maturity level and extensive community support.

Therefore, your application will communicate with Casdoor via OAuth 2.0 (OIDC). There are three ways to connect to Casdoor:

Standard OIDC client

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

What is OIDC?

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

Casdoor fully supports the OIDC protocol. If your application is already using another OAuth 2.0 (OIDC) identity provider via a **standard OIDC client library**, and you want to migrate to Casdoor, using OIDC discovery will make it very **easy** to switch to Casdoor.

Casdoor SDKs

Casdoor SDKs: For most programming languages, Casdoor provides easy-to-use

▶ SDK libraries on top of OIDC, with extended functionality that is only available in Casdoor.

Compared to the standard OIDC protocol, Casdoor's SDK provides more functionalities, like user management and resource uploading, among others. Connecting to Casdoor via the Casdoor SDK requires more time than using a standard OIDC client library, but it offers the best **flexibility** and the most **powerful** API.

Casdoor plugin

Casdoor plugin: If your application is built on top of a popular platform (like Spring Boot, WordPress, etc.) and Casdoor (or a third party) has already provided a plugin or middleware for it, you should use it. Using a plugin is much easier than manually invoking the Casdoor SDK because the former is specially made for the platform.

Plugins:

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

Middleware:

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

SAML

What is SAML?

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

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

Casdoor can be used as an **SAML IdP**. Currently, Casdoor supports the main features of **SAML 2.0**. For more details, see [SAML](#).

Example:

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

Suggestions:

1. The protocol is **powerful** and covers many scenarios, making it one of the most comprehensive SSO protocols.
2. The protocol is **large**, with many optional parameters, so it is difficult to cover all application scenarios 100% in the actual implementation.
3. If the application is **newly** developed, SAML is **not** recommended due to its high technical complexity.

CAS

What is CAS?

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

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

Suggestions:

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

Integrations table

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

Standard OIDC Client

OIDC Discovery

Casdoor has fully implemented the OIDC protocol. If your application is already using a standard OIDC client library to connect to another OAuth 2.0 identity provider, and you want to migrate to Casdoor, using OIDC discovery will make it very easy for you to switch. Casdoor's OIDC discovery URL is:

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

For example, the OIDC discovery URL for the demo site is:

<https://door.casdoor.com/well-known/openid-configuration>, and it contains the following information:

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

List of OIDC Client Libraries

Here is a list of some OIDC client libraries for languages like Go and Java:

OIDC client library	Language	Link
go-oidc	Go	https://github.com/coreos/go-oidc
pac4j-oidc	Java	https://www.pac4j.org/docs/clients/openid-connect.html

Please note that the above table is not exhaustive. For a full list of OIDC client libraries, you can find more details at:

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

OIDC UserInfo Fields

The following table illustrates how OIDC UserInfo fields (via the `/api/userinfo` API) are mapped from properties of Casdoor's User table:

Casdoor User Field	OIDC UserInfo Field
Id	sub

Casdoor User Field	OIDC UserInfo Field
originBackend	iss
Aud	aud
Name	preferred_username
DisplayName	name
Email	email
Avatar	picture
Location	address
Phone	phone

You can see the definition of UserInfo [here](#).

Casdoor SDKs

Introduction

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

Casdoor SDKs can be divided into two categories:

1. **Frontend SDK:** Like Javascript SDK, Vue SDK for websites, Android or iOS SDKs for Apps, etc. Casdoor supports providing authentication for both websites and mobile Apps.
2. **Backend SDK:** SDKs for backend languages like Go, Java, Node.js, Python, PHP, etc.



TIP

If your website is developed in a frontend and backend separated manner, then you can use the Javascript SDK: [casdoor-js-sdk](#) or React SDK: [casdoor-react-sdk](#) or Vue SDK: [casdoor-vue-sdk](#) to integrate Casdoor in frontend. If your web application is a traditional website developed by JSP or PHP, you can just use the backend SDKs only. See an example: [casdoor-python-vue-sdk-example](#)

Mobile SDK	Description	SDK code	Example code
Android SDK	For Android apps	casdoor-android-sdk	casdoor-android-example
iOS SDK	For iOS apps	casdoor-ios-sdk	casdoor-ios-example
React Native SDK	For React Native apps	casdoor-react-native-sdk	casdoor-react-native-example
Flutter SDK	For Flutter apps	casdoor-flutter-sdk	casdoor-flutter-example
Firebase SDK	For Google Firebase apps		casdoor-firebase-example
Unity Games SDK	For Unity 2D/3D PC/Mobile games	casdoor-dotnet-sdk	casdoor-unity-example
uni-app SDK	For uni-app apps	casdoor-uniapp-sdk	casdoor-uniapp-example

Desktop SDK	Description	SDK code	Example code
Electron SDK	For Electron apps	casdoor-js-sdk	casdoor-electron-example
.NET Desktop SDK	For .NET desktop apps	casdoor-dotnet-sdk	WPF: casdoor-dotnet-desktop-example WinForms: casdoor-dotnet-winform-example Avalonia UI: casdoor-dotnet-avalonia-example
C/C++ SDK	For C/C++ desktop apps	casdoor-cpp-sdk	casdoor-cpp-qt-example

Web frontend SDK	Description	SDK code	Example code
Javascript SDK	For traditional non-SPA websites	casdoor-js-sdk	Nodejs backend: casdoor-raw-js-example Go backend: casdoor-go-react-sdk-example

Web frontend SDK	Description	SDK code	Example code
Frontend-only SDK	For frontend-only SPA websites	casdoor-js-sdk	casdoor-react-only-example
React SDK	For React websites	casdoor-react-sdk	Nodejs backend: casdoor-nodejs-react-example Java backend: casdoor-spring-security-react-example
Next.js SDK	For Next.js websites		nextjs-auth
Nuxt SDK	For Nuxt websites		nuxt-auth
Vue SDK	For Vue websites	casdoor-vue-sdk	casdoor-python-vue-sdk-example
Angular SDK	For Angular websites	casdoor-angular-sdk	casdoor-nodejs-angular-example
Flutter SDK	For Flutter Web websites	casdoor-flutter-sdk	casdoor-flutter-example
ASP.NET SDK	For ASP.NET Blazor WASM websites	Blazor.BFF.OpenIDConnect.Template	casdoor-dotnet-blazorwasm-oidc-example
Firebase SDK	For Google Firebase apps		casdoor-firebase-example

Next, use one of the following backend SDKs based on the language of your backend:

Web backend SDK	Description	Sdk code	Example code
Go SDK	For Go backends	casdoor-go-sdk	casdoor-go-react-sdk-example
Java SDK	For Java backends	casdoor-java-sdk	casdoor-spring-boot-starter , casdoor-spring-boot-example , casdoor-spring-security-react-example
Node.js SDK	For Node.js backends	casdoor-nodejs-sdk	casdoor-nodejs-react-example
Python SDK	For Python backends	casdoor-python-sdk	Flask: casdoor-python-vue-sdk-example Django: casdoor-django-js-sdk-example FastAPI: casdoor-fastapi-js-sdk-example
PHP SDK	For PHP backends	casdoor-php-sdk	wordpress-casdoor-plugin
.NET SDK	For ASP.NET backends	casdoor-dotnet-sdk	casdoor-dotnet-sdk-example
Rust SDK	For Rust backends	casdoor-rust-sdk	casdoor-rust-example
C/C++ SDK	For C/C++ backends	casdoor-cpp-sdk	casdoor-cpp-qt-example

Web backend SDK	Description	Sdk code	Example code
Dart SDK	For Dart backends	casdoor-dart-sdk	
Ruby SDK	For Ruby backends	casdoor-ruby-sdk	

For a full list of the official Casdoor SDKs, please see: <https://github.com/orgs/casdoor/repositories?q=sdk&type=all&language=&sort=>

How to use Casdoor SDK?

1. Backend SDK configuration

When your application starts up, you need to initialize the Casdoor SDK config by calling the `InitConfig()` function with required parameters. Take casdoor-go-sdk as example: <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)
}
```

All the parameters for `InitConfig()` are explained as follows:

Parameter	Must	Description
endpoint	Yes	Casdoor Server URL, like <code>https://door.casdoor.com</code> or <code>http://localhost:8000</code>
clientId	Yes	Client ID for the Casdoor application
clientSecret	Yes	Client secret for the Casdoor application
jwtPublicKey	Yes	The public key for the Casdoor application's cert
organizationName	Yes	The name for the Casdoor organization
applicationName	No	The name for the Casdoor application



TIP

The `jwtPublicKey` can be managed in the `Certs` page as below.

Name	Created time	Display name	Scope	Type	Crypto algorithm	Bit size	Expire in years	Action
cert_rjeegc	2022-02-16 11:04:10	New Cert - rjeegc	JWT	x509	RSA	4096	20	Edit Delete
cert-built-in	2022-02-15 12:31:46	Built-in Cert	JWT	x509	RSA	4096	20	Edit Delete

2 in total < 1 > 10 / page

You can find the public key in the cert edit page, copy it or download it for the sdk.

Public key

Copy public key Download public key

```
-----BEGIN CERTIFICATE-----
MIIE+TCCAUGgAwIBAgIDAEJAMAOGC3qG5ib3DEQCBCwUAMDYXhTAb8gNWBAoTFENh
c2Rvb3g3LnYyW5pemF0wUeW5uMRUwEwYDQDEwXMDgYDQDEwXMDgYDQDEwXMDgYDQ
MDE1MDgYDQDEwXMDgYDQDEwXMDgYDQDEwXMDgYDQDEwXMDgYDQDEwXMDgYDQDE
ZzFuaXphdGlvbG9kLW9kb29yY29yY29yY29yY29yY29yY29yY29yY29yY29yY29yY2
AQEFAAOCAG8AMICGKCAgEAsinpb5E1Ym0f1RfSDSS5E8IR7y+lw+Rjj74e5ej
rq4b8zMYk7HeHCyZr/hmNEwEvXnhXu1P0mbQ5ypp/QG08vgEmjAETNmzk1NjOQ
CJCYUrasO/Mn1CQj13xv6m1kHjZ5rKsMhYY1vaxTEP3+V8Hhg3MHFwrb07
uvFMCJE5W8+0rErZCKTR8+9VB3janeBz/zQePFVh79bFzate/hLirPKoG99P1g
OwwoC1A3sarHTP4Qm/LQR0HqZfYb5dyPwAQvhNaDE7mTsrRSBb/wujNCUBD
PTSLVjC04WISf6Nkx0Z7kVmbPstJ5+btvcqsRAGvdsB9h62Kptjs1Yn7GAuo
l3qt4zokbiURyKQjXlvwCQsEftUkSew5zuPSIDRL0LByQTLbx0LqLAFNfW3g/
pzSDjgd/60d6HtmvZni4SmjdyFhXCdb1Kkn7N+xtojfnaNkwep2REV+Rm0fx4Gu
hRsnlsmkUmDeyI29aBt90q11YEQfM2JZEeq+RVtUx+wb4y8K/ID1bcY+fnG5fBpw
IDpS262boog4SRsV3Z7b0w4ZxvOj/1VLoRftjPbLlfbhfr/AeZMHpKoxXvft4
yE+hzqz6WdF0VR9yYc/R5SA73230sjmEjInURhnmRgCpjik/M2K184Kb0
wn8CAwEAAAMQMA4wD4YDVR0TAQHhBaiwADANBgkqhkiG9w0BAQsFAAOCAgEAnZif
DKLX+FlvKRO/SgJ+Pr8P5NkuQmhw97b8CS2g51phDyNglc4/L5dz04Awe6ve
C06lVdWSl8uPUPqjm72uMPSNWjwLxG3QsmMURNwFLTRem/hel0Zgur91M
8haawdSdJh2RgmFoDeEzr8NVRh8KNC01ddTjKus1N0/rH221W4j4tZsvCvl
2nR42Fyap3O/g2JXmHmNROwZmNjggsF7XVENC5uFO1JYwLauXcg54L7XVLG
omnKNNcc8h1FceKj/nbMhmodnfWkDTsIcbNmcOPNH06ixqMy/Hgc+mWYV7maAG
Jtevs3qgMz8F9Qzr3HpcU6R3ZYyWDY/xxPisukTOPZgtH979XC4mdf0WPnOBLqL
2Dj1zaBmjiGjovb7XNvKcUDYXw85ZTzQ5b9cl4e+6bmyWqQlTwt+Ati/ufEV
XzCj70B4lX6xau1kLepV901GERizRz9P9NNINA7koO5AVmP90DQTKt+LbXnzE
HHwWky8xHqZQF9sR7YBPLGs/Ac6niv5UA150qj/8dlRZ/veyfGo2yZsi+hKVU5
nCJHBCAyFnm1hdvdwEdH33jDBjNB6ciotZfz3VyalWSalAdosHAGMWXkuWP+h
8XkXmzkuHbTMQYzPDgSp5sak+S4Q9wb8RRAYo=
-----END CERTIFICATE-----
```

Private key

Copy private key Download private key

```
-----BEGIN PRIVATE KEY-----
MIIEKQIBAAKCAgEAsinpb5E1Ym0f1RfSDSS5E8IR7y+lw+Rjj74e5ejrq4b8zMY
k7HeHCyZr/hmNEwEvXnhXu1P0mbQ5ypp/QG08vgEmjAETNmzk1NjOQJCYUras
o/Mn1CQj13xv6m1kHjZ5rKsMhYY1vaxTEP3+V8Hhg3MHFwrb07uvFMCJE5W
8+0rErZCKTR8+9VB3janeBz/zQePFVh79bFzate/hLirPKoG99P1gOwwoC1A
3sarHTP4Qm/LQR0HqZfYb5dyPwAQvhNaDE7mTsrRSBb/wujNCUBDPTSLVjC0
4WISf6Nkx0Z7kVmbPstJ5+btvcqsRAGvdsB9h62Kptjs1Yn7GAuoL3qt4zo
kbiURyKQjXlvwCQsEftUkSew5zuPSIDRL0LByQTLbx0LqLAFNfW3g/pzSDjgd
/60d6HtmvZni4SmjdyFhXCdb1Kkn7N+xtojfnaNkwep2REV+Rm0fx4GuhRsnlsm
kUmDeyI29aBt90q11YEQfM2JZEeq+RVtUx+wb4y8K/ID1bcY+fnG5fBpwIDp
S262boog4SRsV3Z7b0w4ZxvOj/1VLoRftjPbLlfbhfr/AeZMHpKoxXvft4yE+
hzqz6WdF0VR9yYc/R5SA73230sjmEjInURhnmRgCpjik/M2K184Kb0wn8CAwEA
AAAMQMA4wD4YDVR0TAQHhBaiwADANBgkqhkiG9w0BAQsFAAOCAgEAnZifDKLX
+FlvKRO/SgJ+Pr8P5NkuQmhw97b8CS2g51phDyNglc4/L5dz04Awe6veC06lV
dWSl8uPUPqjm72uMPSNWjwLxG3QsmMURNwFLTRem/hel0Zgur91M8haawdSdJh
2RgmFoDeEzr8NVRh8KNC01ddTjKus1N0/rH221W4j4tZsvCvl2nR42Fyap3O
/g2JXmHmNROwZmNjggsF7XVENC5uFO1JYwLauXcg54L7XVLGomnKNNcc8h1FceK
j/nbMhmodnfWkDTsIcbNmcOPNH06ixqMy/Hgc+mWYV7maAGJtevs3qgMz8F9Qz
r3HpcU6R3ZYyWDY/xxPisukTOPZgtH979XC4mdf0WPnOBLqL2Dj1zaBmjiGjov
b7XNvKcUDYXw85ZTzQ5b9cl4e+6bmyWqQlTwt+Ati/ufEVXzCj70B4lX6xau1kL
epV901GERizRz9P9NNINA7koO5AVmP90DQTKt+LbXnzEHHwWky8xHqZQF9sR7
YBPLGs/Ac6niv5UA150qj/8dlRZ/veyfGo2yZsi+hKVU5nCJHBCAyFnm1hdvdwE
dH33jDBjNB6ciotZfz3VyalWSalAdosHAGMWXkuWP+h8XkXmzkuHbTMQYzPDg
Sp5sak+S4Q9wb8RRAYo=
-----END PRIVATE KEY-----
```

Save Save & Exit


Then you can select the cert in the application edit page.

Edit Application Save Save & Exit

Name:

Display name:

Logo: URL:

Preview: 

Home:

Description:

Organization:

Client ID:

Client secret:

Cert: ←

Redirect URLs:

Action

2. Frontend configuration

First, install `casdoor-js-sdk` via NPM or Yarn:

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

Or:

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

Then define the following utility functions (better in a global JS file like `Setting.js`):

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

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

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

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

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

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

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

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

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

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

In the entrance file of your frontend code (like `index.js` or `app.js` in React), you need to initialize the `casdoor-js-sdk` by calling the `InitConfig()` function with required parameters. The first 4 parameters should use the same value as the Casdoor backend SDK. The last parameter `redirectPath` is relative path for the redirected URL, returned from Casdoor's login page.

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

(Optional) Because we are using React as example, our `/callback` path is hitting the React route. We use the following React component to receive the `/callback` call and send to the backend. You can ignore this step if you are redirecting to backend directly (like in JSP or PHP).

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

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

  componentWillMount() {
    this.login();
  }

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

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

export default withRouter(AuthCallback);
```

3. Get login URLs

Next you can show the "Sign up" and "Sign in" buttons or links to your users. The URLs can either be retrieved in the frontend or backend. See more details at: </docs/basic/core-concepts#login-urls>

4. Get and verify access token

Here are the steps:

1. The user clicks the login URL and is redirected to Casdoor's login page, like: `https://door.casdoor.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. The user enters username & password and clicks `Sign In` (or just click the third-party login button like `Sign in with GitHub`).
3. The user is redirected back to your application with the authorization code issued by Casdoor (like: `https://forum.casbin.com?code=xxx&state=yyy`), your application's backend needs to exchange the authorization code with the access token and verify that the access token is valid and issued by Casdoor. The functions `GetOAuthToken()` and `ParseJwtToken()` are provided by Casdoor backend SDK.

The following code shows how to get and verify the access token. For a real example of Casnode (a forum website written in Go), see: <https://github.com/casbin/casnode/blob/6d4c55f5c9a3c4bd8c85f2493abad3553b9c7ac0/controllers/account.go#L51-L64>

```
// get code and state from the GET parameters of the redirected URL
code := c.Input().Get("code")
state := c.Input().Get("state")

// exchange the access token with code and state
token, err := auth.GetOAuthToken(code, state)
if err != nil {
    panic(err)
}

// verify the access token
claims, err := auth.ParseJwtToken(token.AccessToken)
if err != nil {
    panic(err)
}
```

If `ParseJwtToken()` finishes with no error, then the user has successfully logged into the application. The returned `claims` can be used to identify the user later.

4. Identify user with access token

! INFO

This part is actually your application's own business logic and not part of OIDC, OAuth or Casdoor. We just provide good practices as a lot of people don't know what to do for the next step.

In Casdoor, access token is usually identical as ID token. They are the same thing. So the access token contains all information for the logged-in user.

The variable `claims` returned by `ParseJwtToken()` is defined as:

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

1. `User`: the User object, containing all information for the logged-in user, see definition at: </docs/basic/core-concepts#user>
2. `AccessToken`: the access token string.
3. `jwt.RegisteredClaims`: some other values required by JWT.

At this moment, the application usually has two ways to remember the user session: `session` and `JWT`.

Session

The Method to set session varies greatly depending on the language and web framework. E.g., Casnode uses [Beego web framework](#) and set session by calling: `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) // set session
```

JWT

The `accessToken` returned by Casdoor is actually a JWT. So if your application uses JWT to keep user session, just use the access token directly for it:

1. Send the access token to frontend, save it in places like `localStorage` of the browser.
2. Let the browser send the access token to backend for every request.
3. Call `ParseJwtToken()` or your own function to verify the access token and get logged-in user information in your backend.

5. (Optional) Interact with the User table

! INFO

This part is provided by `Casdoor Public API` and not part of the OIDC or OAuth.

Casdoor Backend SDK provides a lot of helper functions, not limited to:

- `GetUser(name string)`: get a user by username.
- `GetUsers()`: get all users.
- `AddUser()`: add a user.
- `UpdateUser()`: update a user.
- `DeleteUser()`: delete a user.
- `CheckUserPassword(auth.User)`: check user's password.

These functions are implemented by making RESTful calls against `Casdoor Public API`. If a function is not provided in Casdoor Backend SDK, you can make RESTful calls by yourself.

How to Enable Single Sign-On

Introduction

You have connected Casdoor and configured more than one application in an organization. You want users to sign in once to any app in the organization and then be able to sign in when they go to another app without any extra clicks.

We offer this single sign-on feature. To enable it, you just need to:

- Enable the Auto Sign-In button.
- Fill in the URL for the home page.
- Add a **Silent Sign-In** function to the application home page.

NOTE

The basic sign-in process provided by Casdoor allows users to log in to other applications in the organization by selecting the user who is currently logged in or using another account.

After enabling auto sign-in, the selection box will not be displayed, and the logged-in user will log in directly.

Configuration


1. Fill in the "home" field. It can be the application's home page or the login page.

Edit Application

Name ? :

Display name ? :

Logo ? :
URL ? :

Preview: 

Home ? :

Description ? :

2. Enable the Auto Sign-In button.

Password ON ? :

Enable signup ? :

Signin session ? :

Auto signin ? :

Enable code
signin ? :

Enable WebAuthn
signin ? :

Add Silent Sign-In

In fact, we implement auto login by carrying parameters in the URL. Therefore, your applications need to have a method to trigger the login after jumping to the URL. We provide the [casdoor-react-sdk](#) to help you quickly implement this feature. You can see the details in [use-in-react](#).

! INFO

How it works

1. In the URL to the application home page, we will carry the `silentSignin` parameter.
2. In your home page, determine whether you need to log in silently (automatically) by checking the `silentSignin` parameter. If `silentSignin === 1`, the function should return the `SilentSignin` component, which will help you initiate a login request. Since you have auto-login enabled, users will log in automatically without clicking.

Add Popup Sign-In

The "popup sign-in" feature will open 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 this feature by carrying parameters in the URL.

! INFO

How to use

Use the `popupSignIn()` method in the `casdoor-js-sdk` to quickly implement this 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` is in the login parameters, Casdoor will send `code` and `state` as a message to the main window and finish getting the `token` in the main window using the SDK.

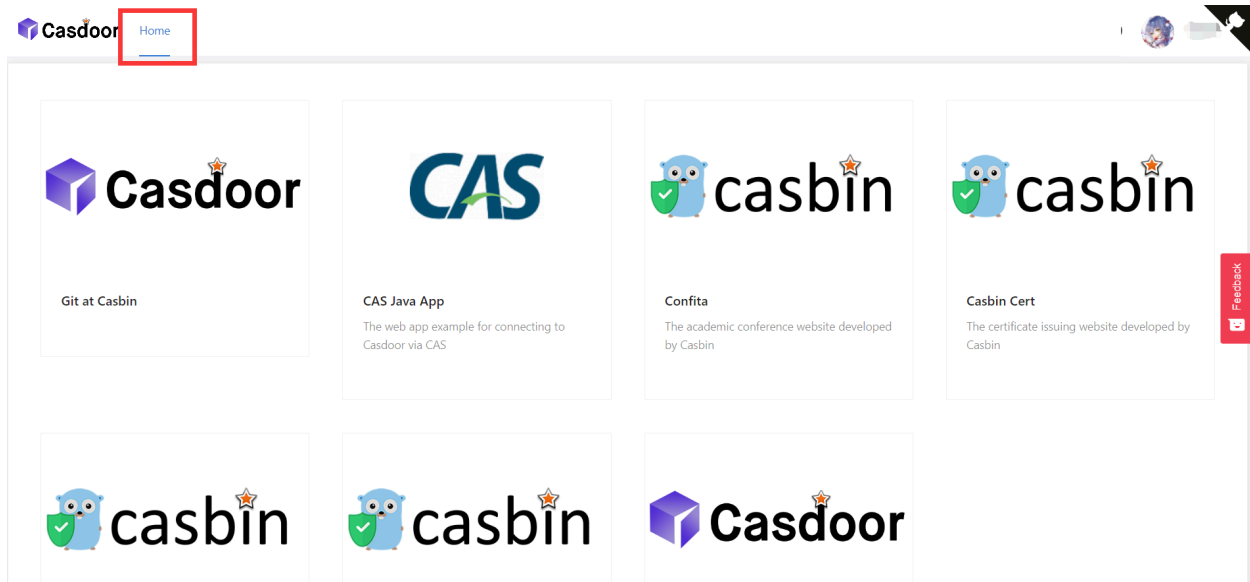
Using SSO

The configuration is complete. Below, we will show you how to use auto login.

! INFO

Make sure your application can redirect to the user's profile page. The `getMyProfileUrl(account, returnUrl)` API is provided in our SDK for each language.

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



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

Vue SDK

The Casdoor Vue SDK is designed for Vue 2 and Vue 3, making it very convenient to use.

The Vue SDK is based on `casdoor-js-sdk`. You can also use the `casdoor-js-sdk` directly, which will allow for more customization.

Please note that this plugin is still in development. If you have any questions or suggestions, please feel free to contact us by opening an [issue](#).

We will now show you the necessary steps below.

If you are still unsure how to use it after reading the README.md, you can refer to the example: [casdoor-python-vue-sdk-example](#) for more details.

The example's front-end is built with `casdoor-vue-sdk`, while the back-end is built with `casdoor-python-sdk`. You can view the source code in the [example](#).

Installation

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

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

Initializing the SDK

To initialize the SDK, you will need to provide 5 string parameters in the following order:

Name	Required	Description
serverUrl	Yes	The URL of your Casdoor server.
clientId	Yes	The Client ID of your Casdoor application.
appName	Yes	The name of your Casdoor application.
organizationName	Yes	The name of the Casdoor organization linked to your Casdoor application.
redirectPath	No	The path of the redirect URL for your Casdoor application. If not provided, it will default to <code>/callback</code> .

For Vue 3:

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

const config = {
  serverUrl: "http://localhost:8000",
  clientId: "4262bea2b293539fe45e",
  organizationName: "casbin",
```

For Vue 2:

```
// 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')
```

Example

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

Auto Fix

If the `postinstall` hook does not get triggered or if you have updated the Vue version, try running the following command to resolve the redirecting issue:

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

For more information about switching Vue versions, please refer to the [vue-demi docs](#).

Desktop SDKs

Electron App Example for Casdoor

This is an Electron app example that demonstrates Casdoor's integration capabilities.

dotNET Desktop App

A dotNET desktop app example for Casdoor

Mobile SDKs .NET MAUI App

A .NET MAUI App example for Casdoor

Qt Desktop App

A Qt desktop app example for Casdoor

Electron App Example for Casdoor

An [Electron app example](#) that demonstrates Casdoor's integration capabilities.

How to Run the Example

Initialization

You need to initialize 6 parameters, all of which are string type:

Name	Description	Path
serverUrl	Your Casdoor server URL	src/App.js
clientId	The Client ID of your Casdoor application	src/App.js
appName	The name of your Casdoor application	src/App.js
redirectPath	The path of the redirect URL for your Casdoor application, will be <code>/callback</code> if not provided	src/App.js
clientSecret	The Client Secret of your Casdoor application	src/App.js
casdoorServiceDomain	Your Casdoor server URL	public/ electron.js

If you don't set these parameters, this project will use the [Casdoor online demo](#) as the default Casdoor server and use the [Casnode](#) as the default Casdoor application.

Available Commands

In the project directory, you can run:

`npm run dev` or `yarn dev`

Builds the electron app and runs this app.

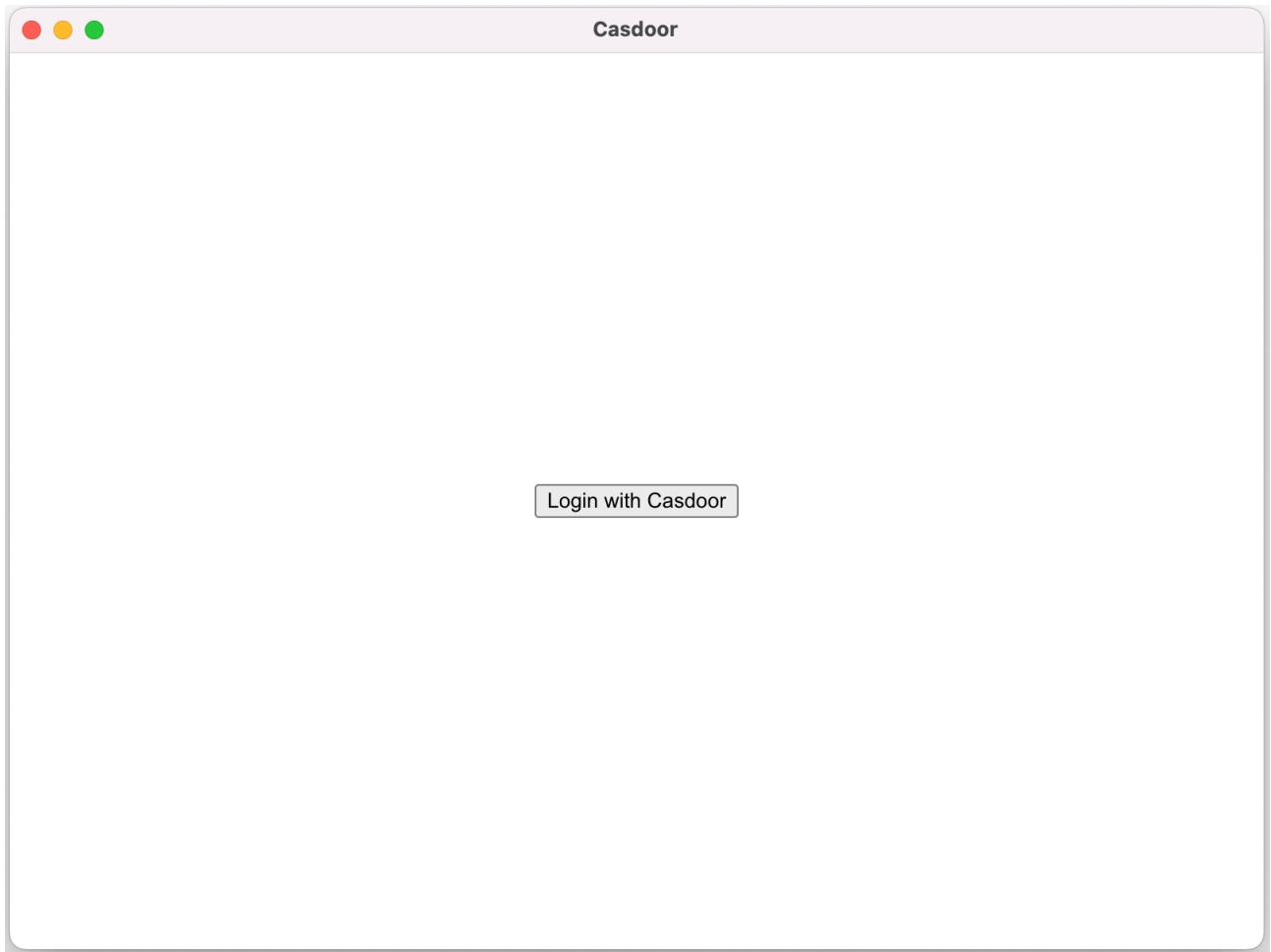
`npm run make` or `yarn make`

Packages and distributes your application. It will create the `out` folder where your package will be located:

```
// Example for macOS out/  
├─ 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
```

Preview

Once you run this Electron application, a new window will appear on your desktop.



If you click the `Login with Casdoor` button, your default browser will automatically open and display the login page.



Auto sign in [Forgot password?](#)

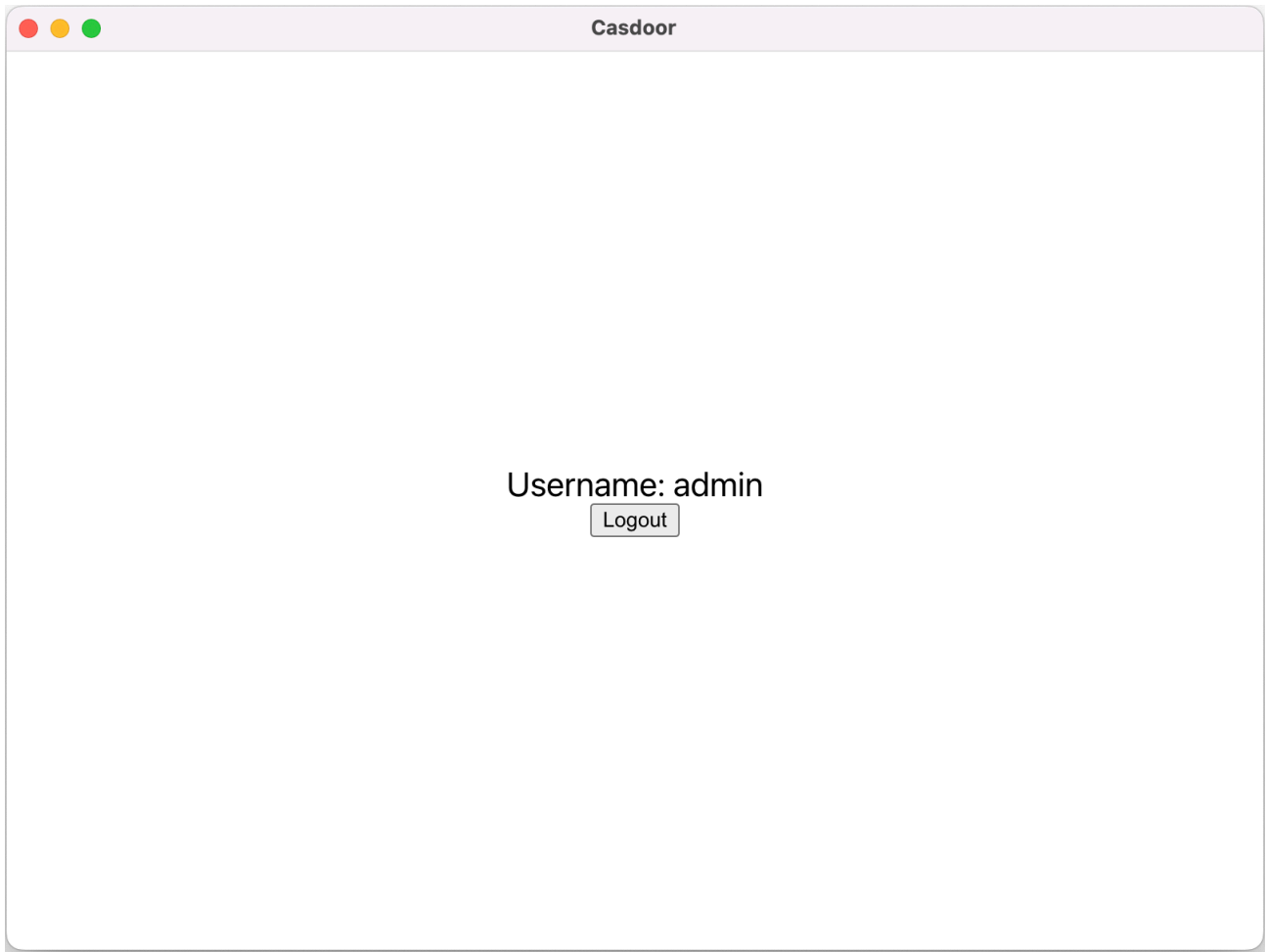
[Sign In](#)

[No account? sign up now](#)

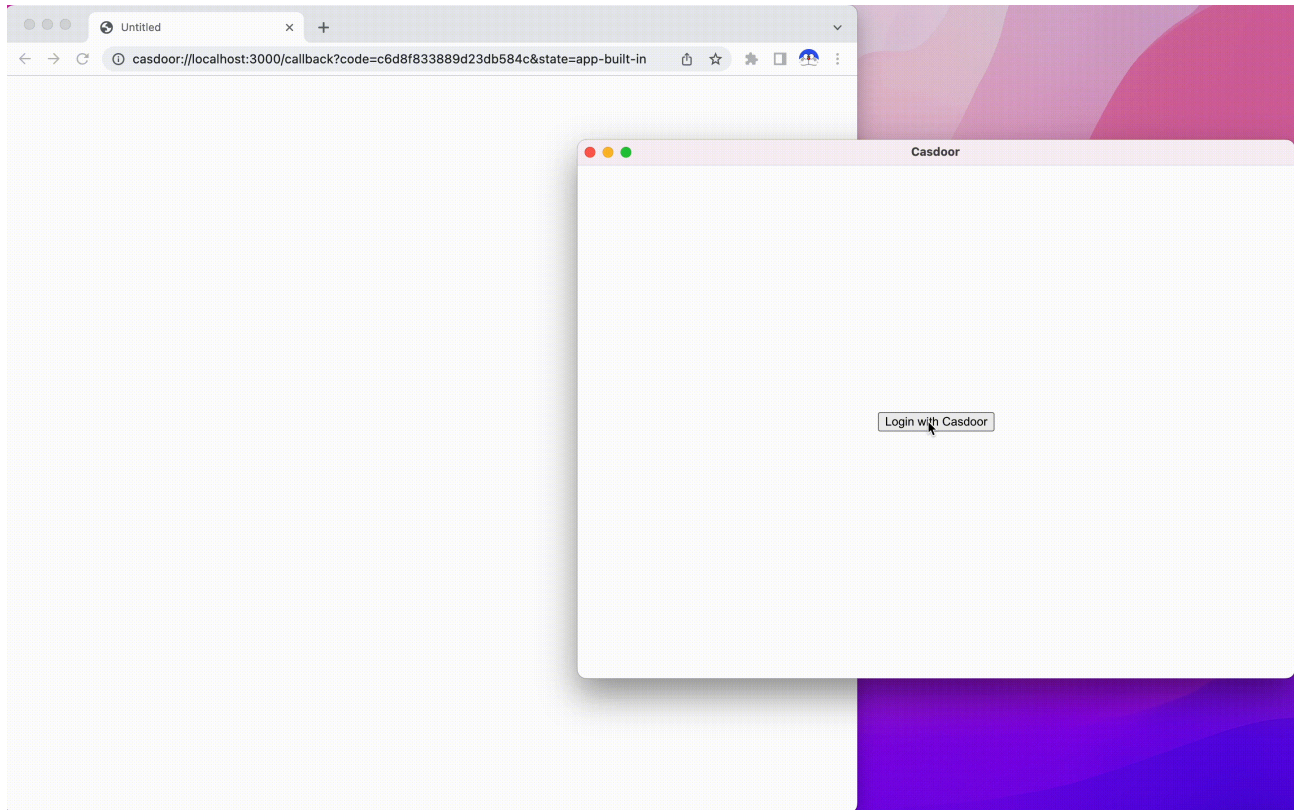


Made with ❤️ by [Casdoor](#)

Following a successful login, your Electron application will open, and your user name will be displayed on your application.



You can preview the entire process in the gif image below.



Integration Steps

Set the custom protocol

Firstly, you need to set the custom protocol called `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);
}
```

This will allow the browser to open your electron application and send the login info to the electron application.

Open the login URL in the browser

```
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;
```

You can change the first five parameters.

Listen to the open application event

Once you successfully log in through the browser, the browser will open your Electron application. Therefore, you must listen to the open application event.

```
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);
      }
    }
  });
}
```

You can get the code from the browser, which is `casdoor_code` or `params.code`.

Parse the code and get the user info

```
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}`,
  });
}
```

Finally, you can parse the code and get the user info following the [OAuth docs page](#).

dotNET Desktop App

A [Dotnet desktop app example](#) for Casdoor.

How to Run the Example

Prerequisites

- [dotNET 6 SDK](#)
- [WebView2 Runtime](#) (It is usually preinstalled on Windows)

Initialization

The initialization requires 5 parameters, all of which are of type string:

Name	Description	File
Domain	The host/domain of your Casdoor server	<code>CasdoorVariables.cs</code>
ClientId	The Client ID of your Casdoor application	<code>CasdoorVariables.cs</code>
AppName	The name of your Casdoor application	<code>CasdoorVariables.cs</code>
CallbackUrl	The path of the callback URL for your Casdoor application. If not	<code>CasdoorVariables.cs</code>

Name	Description	File
	provided, it will be <code>casdoor://callback</code>	
ClientSecret	The Client Secret of your Casdoor application	<code>CasdoorVariables.cs</code>

If you do not set these parameters, the project will default to using the [Casdoor online demo](#) as the Casdoor server and the [Casnode](#) as the Casdoor application.

Running

Visual Studio

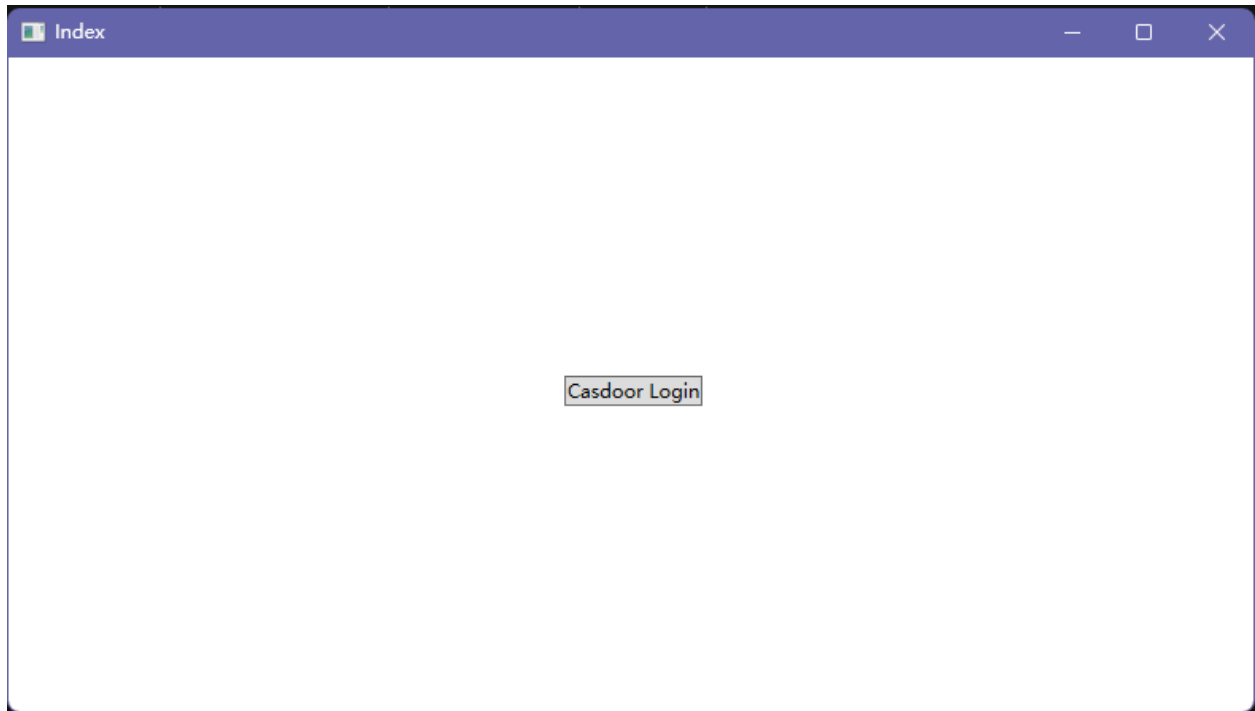
1. Open `casdoor-dotnet-desktop-example.sln`
2. Press `Ctrl + F5` to start

Command Line

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

Preview

After running the dotNET desktop application, a new window will appear on your desktop.

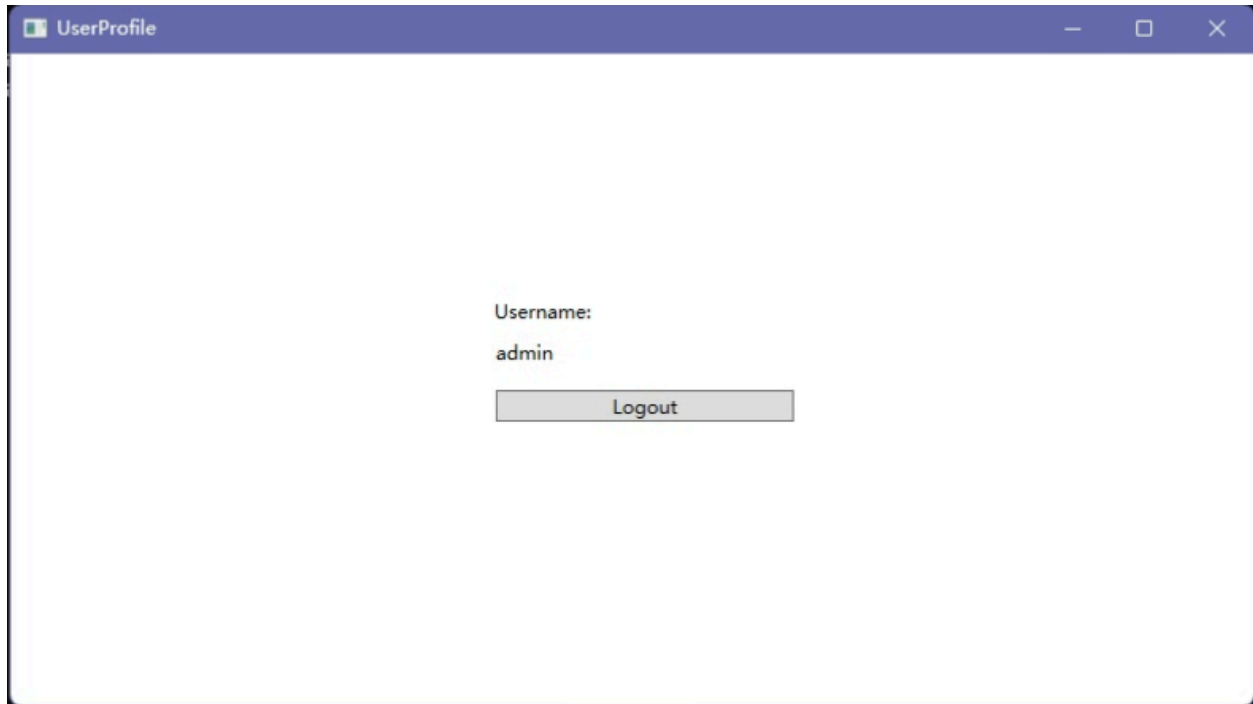


If you click the `Casdoor Login` button, a login window will appear on your

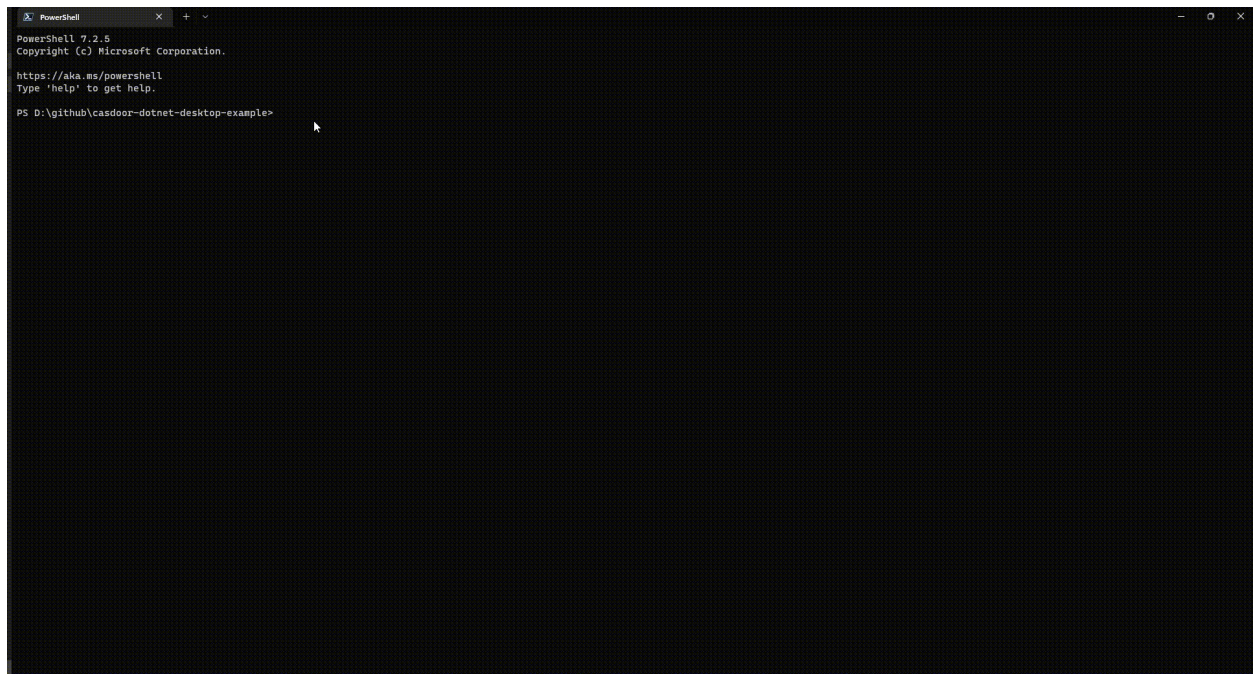


desktop.

After successfully logging in, a user profile window will appear on your desktop, displaying your username.



You can preview the entire process in the GIF image below.



How to Integrate

Opening the Login Window

```
var login = new Login();  
// Triggered when login succeeds, you will receive an auth code in  
the event handler  
login.CodeReceived += Login_CodeReceived;  
login.ShowDialog();
```

Using the Auth Code to Get User Info

```
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",
```


Mobile SDKs .NET MAUI App

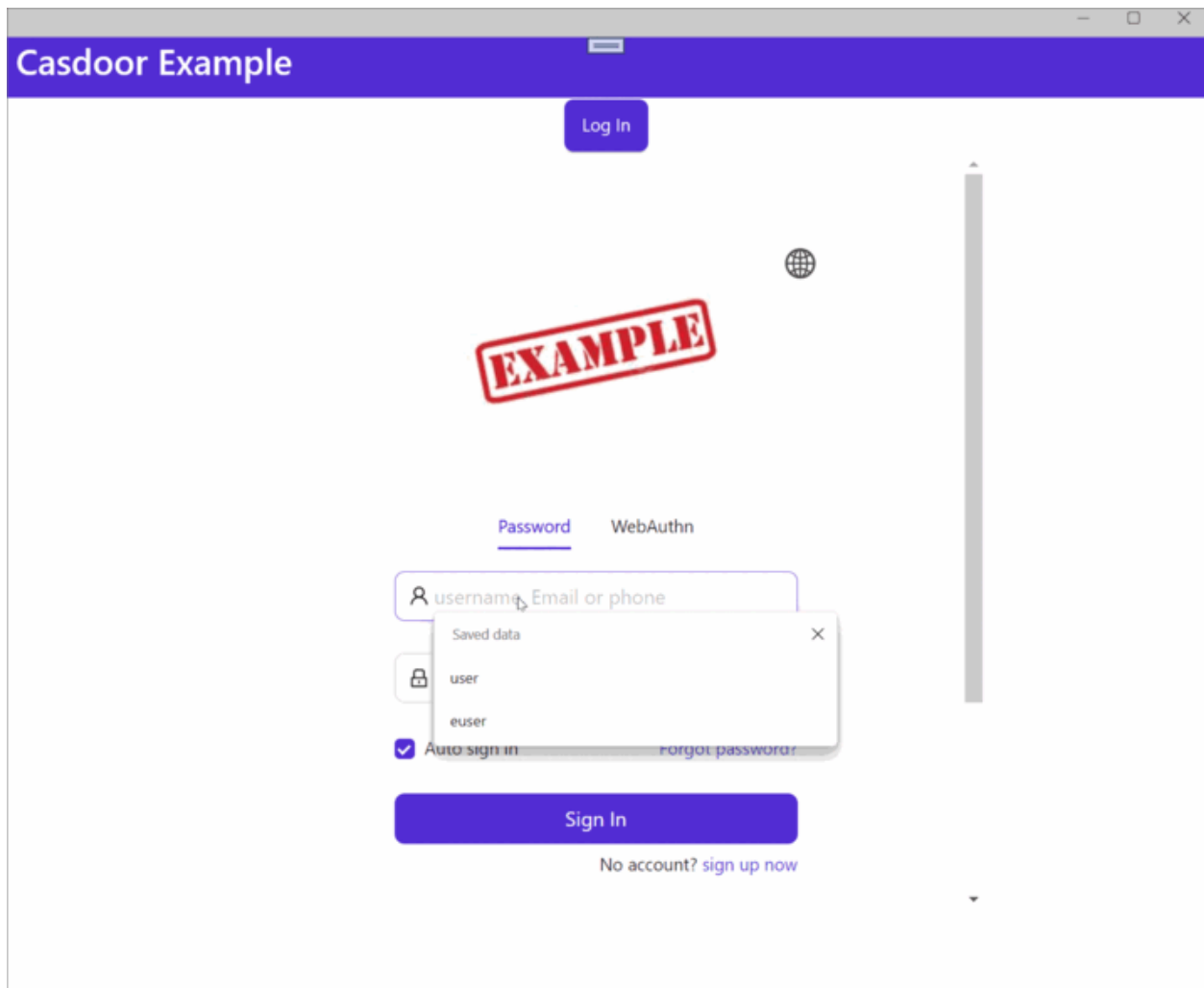
This repository contains a [.NET MAUI app](#) and [.NET MAUI library](#) for demonstrating Casdoor authentication by OpenID Connect.

Demonstration

Android

.NET

Windows



Requirements

- [.NET 7 SDK](#) installed on your machine
- The required assets needed for your target platform(s), as described [here](#)
- Visual Studio 2022 for Windows 17.3 or Visual Studio 2022 for Mac 17.4 (optional)

Getting Started

Step 1: Create a MAUI Application

Create your [MAUI Application](#).

Step 2: Add a Reference

Add a reference to the `Casdoor.MauioIdcClient` in your project.

Step 3: Add the Casdoor Client

Add `CasdoorClient` as a singleton in the services.

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

Step 4: Design the UI

Add code to the `MainPage` file.

MainPage.xaml

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

    <ScrollView>
        <VerticalStackLayout>

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

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

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

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

                <Button
                    x:Name="CounterBtn"
                    Text="Click me"
                />
            </StackLayout>
        </VerticalStackLayout>
    </ScrollView>
</ContentPage>
```

MainPage.cs

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

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

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

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

            SemanticScreenReader.Announce(CounterBtn.Text);
        }

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

Step 5: Support the Android Platform

Modify the `AndroidManifest.xml` file.

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

Step 6: Launch the Application

Visual Studio: Press Ctrl + F5 to start.

Qt Desktop App

A [Qt desktop app example](#) for Casdoor.

How to Run the Example

Prerequisites

- [Qt6 SDK](#)
- [OpenSSL toolkit](#)

Initialization

You need to initialize 7 string parameters:

Name	Description	File
endpoint	Your Casdoor server host/domain	<code>mainwindow.h</code>
client_id	The Client ID of your Casdoor application	<code>mainwindow.h</code>
client_secret	The Client Secret of your Casdoor application	<code>mainwindow.h</code>
certificate	The public key for the Casdoor application's cert	<code>mainwindow.h</code>
org_name	The name of your Casdoor organization	<code>mainwindow.h</code>

Name	Description	File
app_name	The name of your Casdoor application	mainwindow.h
redirect_url	The path of the callback URL for your Casdoor application, will be <code>http://localhost:8080/callback</code> if not provided	mainwindow.h

If you don't set the `endpoint` parameter, this project will use <http://localhost:8000> as the default Casdoor server.

Running the Application

Using Qt Creator

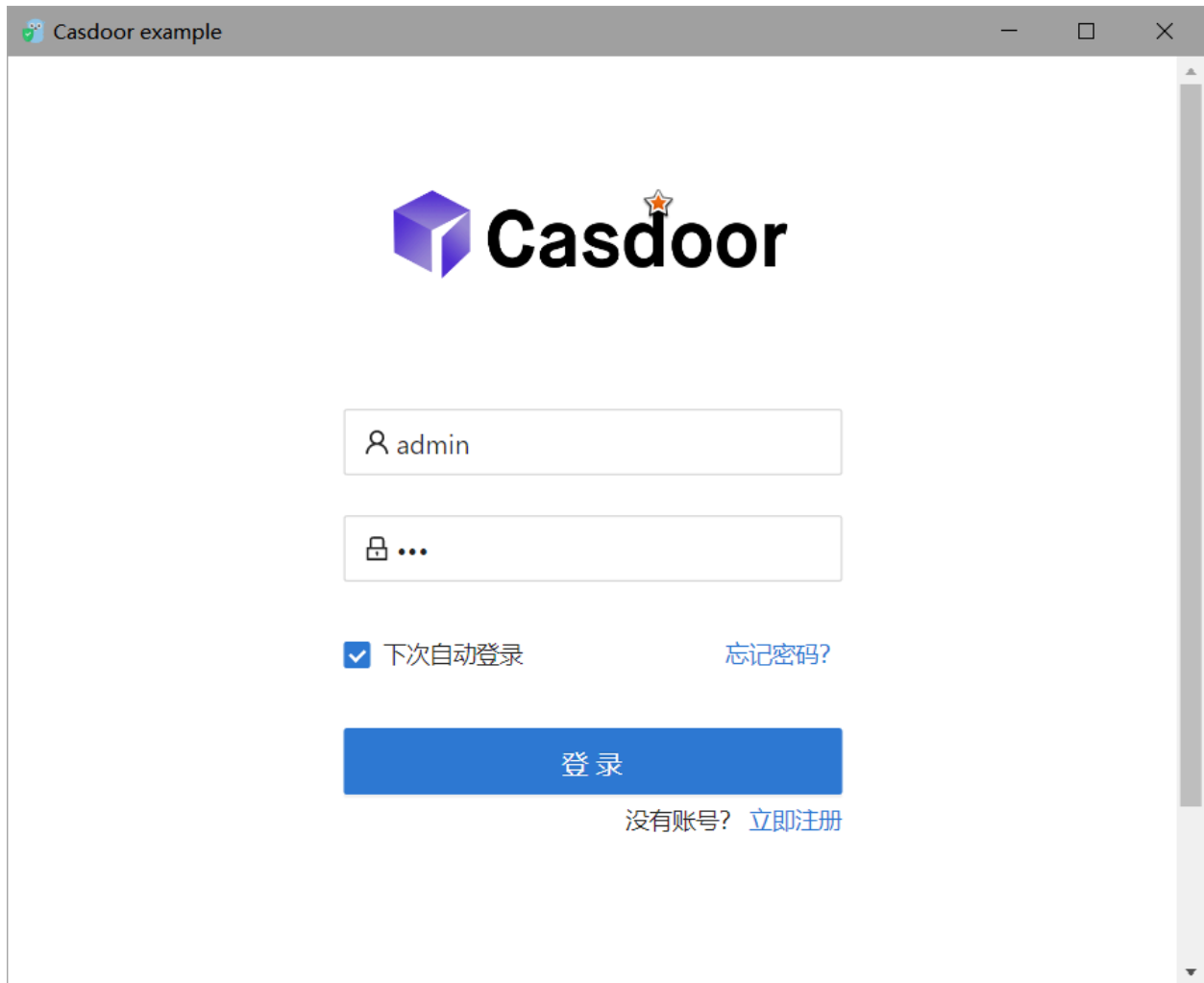
1. Open `casdoor-cpp-qt-example.pro`
2. Set the `INCLUDEPATH` of OpenSSL in `casdoor-cpp-qt-example.pro`
3. Press `Ctrl + R` to start

Preview

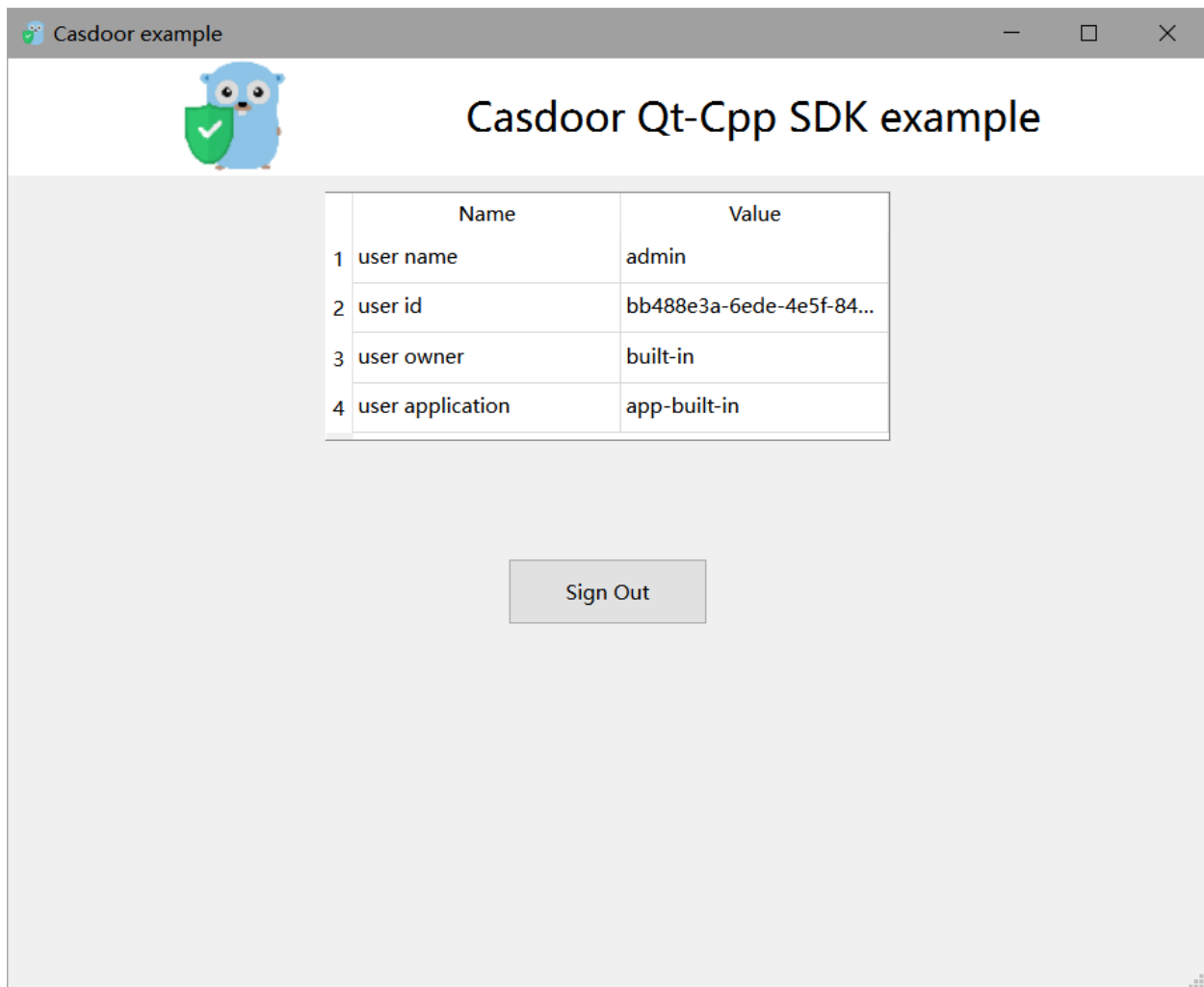
After running this Qt desktop application, a new window will be shown on your desktop.



If you click the `Sign In` button, a login window will be shown on your desktop.



After a successful login, a user profile window will be shown on your desktop, displaying your user information.



You can preview the entire process in the following GIF image.



How to Integrate

Opening the Login Window

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

Listening to the Open Application Event

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

Using Auth Code to Get the User Info

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

Mobile SDKs

React Native App

A React Native mobile app example for Casdoor

React Native App

There is a [Casdoor React Native mobile app example](#) to get you up to speed on how to use Casdoor in React Native.

How to Run the Example

Quick Start

- download the code

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

- install dependencies

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

- run on ios

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

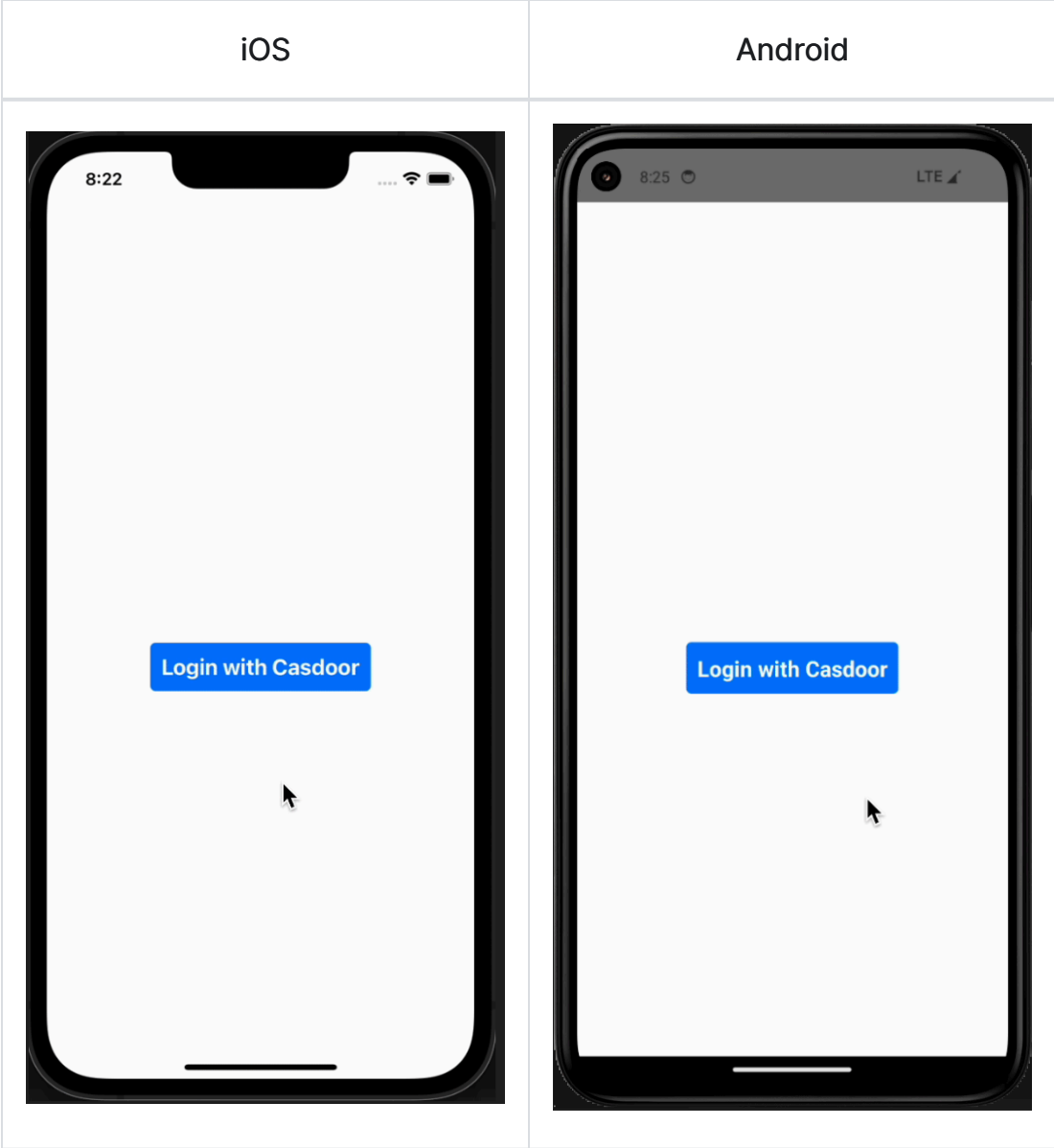
- run on android

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

Make sure to turn on the emulator or real device before running.

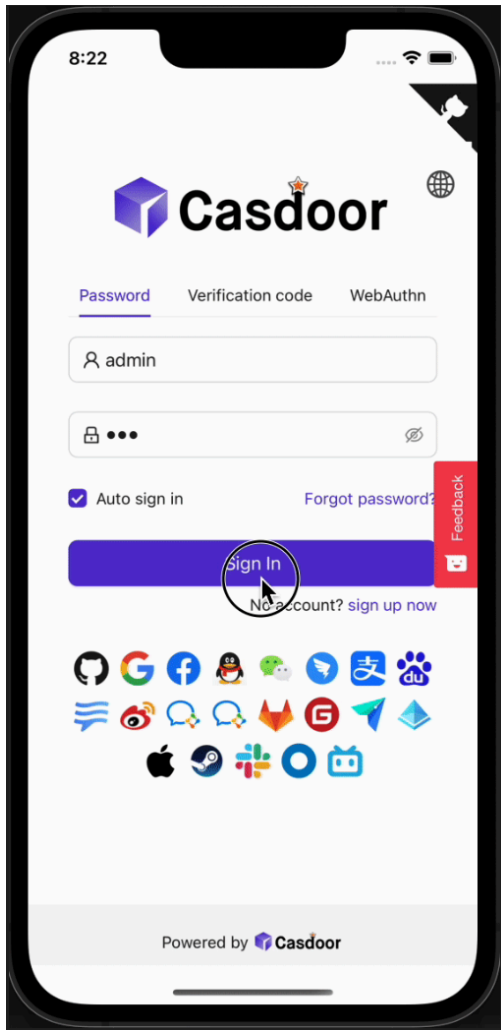
Preview

After running this react-native-example mobile application, the following window will be displayed on the emulator or real device.

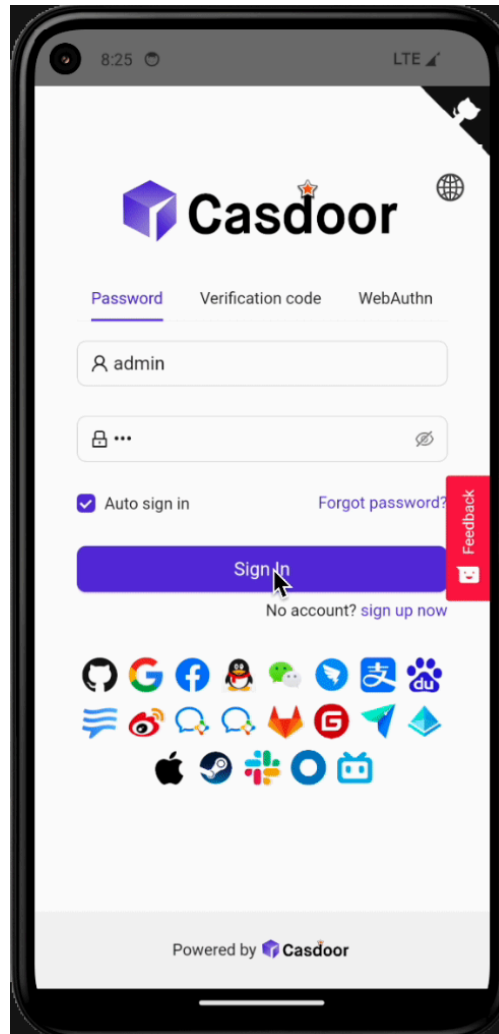


If you click the `Login with Casdoor` button, the Casdoor login window will appear on the screen.

iOS

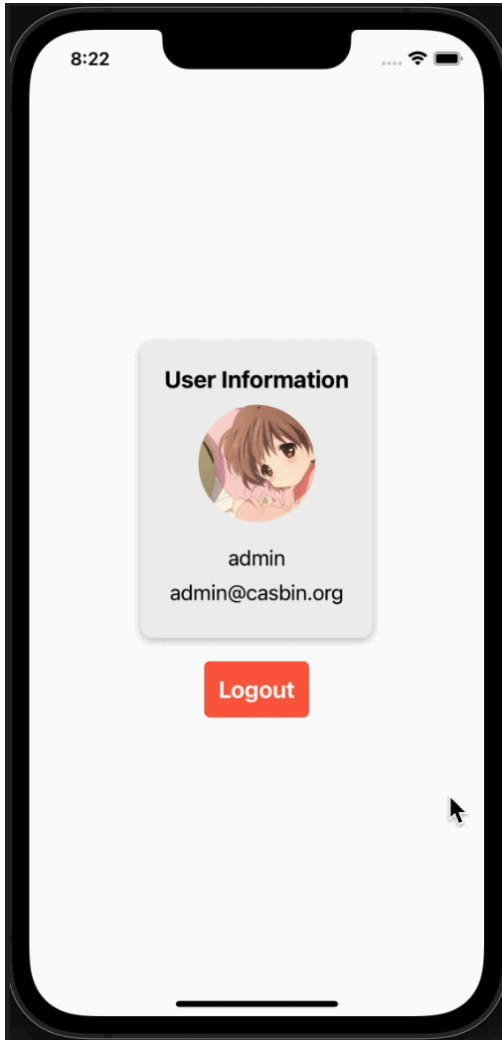


Android

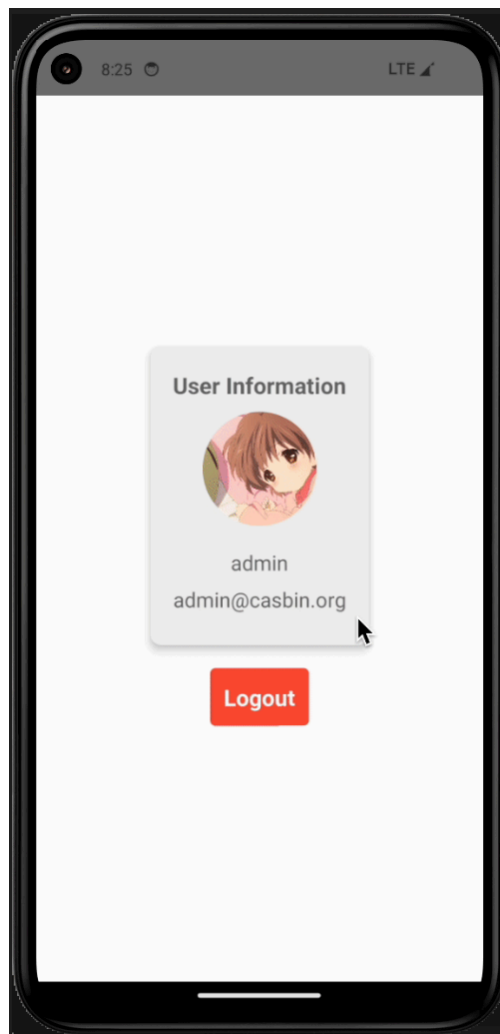


After a successful login, a user profile window will appear on your screen displaying your user information.

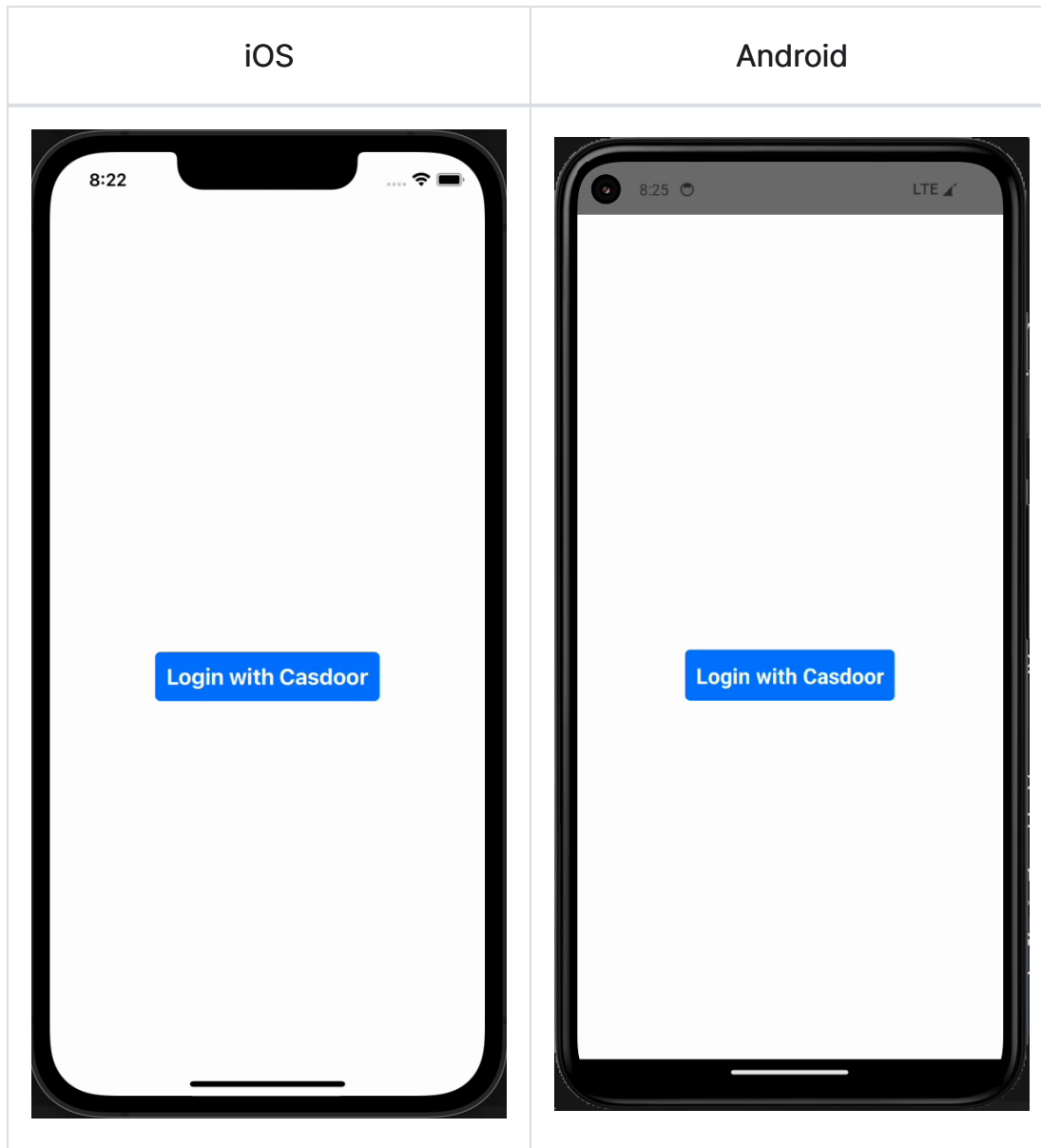
iOS



Android



You can preview the entire process in the following GIF image.



How to Integrate

The above example uses [casdoor-react-native-sdk](#), you can also integrate this sdk in your own project.

The integration and use of the sdk is very simple, the following steps will show

you how to integrate and use:

Step 1: Import SDK

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

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

Step 2: Initialize SDK

Initialization requires 7 parameters, which are all string type:

Name (in order)	Must	Description
serverUrl	Yes	your Casdoor server URL
clientId	Yes	the Client ID of your Casdoor application
appName	Yes	the name of your Casdoor application
organizationName	Yes	the name of the Casdoor organization connected with your Casdoor application
redirectPath	No	the path of the redirect URL for your Casdoor application, will be <code>/callback</code> if not provided
signinPath	No	the path of the signin URL for your Casdoor application

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

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

Step 3: Use SDK

Use the corresponding API interface of the sdk at the appropriate place.

The simplest casdoor authorization and authentication process can be realized by using the following three APIs:

```
// get the signin url
getSigninUrl()

// get Access Token
getAccessToken(redirectUrl); // http://localhost:5000/
callback?code=b75bc5c5ac65ffa516e5&state=gjmf dgqf498

// decode jwt token to get user info
JwtDecode(jwtToken)
```

If you want to use other interfaces, please check [casdoor-react-native-sdk](#) for more help.

Casdoor Plugin

Casdoor also provides plugins or middlewares for some very popular platforms, such as Java's Spring Boot, PHP's WordPress, and Python's Odoo, among others.

Casdoor plugin	Language	Source code
Spring Boot plugin	Java	https://github.com/casdoor/casdoor-spring-boot-starter
Spring Boot example	Java	https://github.com/casdoor/casdoor-spring-boot-example
WordPress plugin	PHP	https://github.com/casdoor/wordpress-casdoor-plugin
Odoo plugin	Python	https://github.com/casdoor/odoo-casdoor-oauth
Django plugin	Python	https://github.com/casdoor/django-casdoor-auth

For a complete list of the official Casdoor plugins, please visit the [Casdoor repositories](#).

Next.js

[nextjs-auth](#) is an example of how to integrate casdoor in a next-js project. We will guide you through the steps below.

Step 1: Deploy Casdoor

Firstly, Casdoor should be deployed.

You can refer to the Casdoor official documentation for the [Server Installation](#). Please deploy your Casdoor instance in production mode.

After a successful deployment, make sure the following:

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

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

Step 2: Add Middleware

Middleware allows you to run code before a request is completed. Then, based on the incoming request, you can modify the response by rewriting, redirecting, modifying the request or response headers, or responding directly.

Use the file `middleware.ts` (or `.js`) in the root of your project to define Middleware. For example, at the same level as `pages` or `app`, or inside `src` if

applicable.

Example

```
//define which paths Middleware will run on
const protectedRoutes = ["/profile"];

export default function middleware(req) {
  if (protectedRoutes.includes(req.nextUrl.pathname)) {
    //redirect the incoming request to a different URL
    return NextResponse.redirect(new URL("/login", req.url));
  }
}
```

See next.js official documentation [middleware](#) for more details.

Step 3: Use Casdoor SDK

1. Install the SDK

First, install `casdoor-js-sdk` via NPM or Yarn:

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

Or:

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

2. Initializing the SDK

Then initialization 6 string-type parameters in the following order:

Name	Required	Description
serverUrl	Yes	Casdoor Server URL, such as <code>http://localhost:8000</code>
clientId	Yes	Application client ID
clientSecret	Yes	Application client secret
organizationName	Yes	Application organization
appName	Yes	Application name
redirectPath	Yes	redirected URL

Example

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

⚠ CAUTION

Replace the configuration values with your own Casdoor instance, especially the `clientId`, `clientSecret`, and `serverUrl`.

3. Redirect to the Login Page

When you need to authenticate users who access your app, you can send the target URL and redirect to the login page provided by Casdoor.

Make sure you have added the callback URL (e.g. <http://localhost:8080/callback>) in the application configuration beforehand.

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

4. Get Token and Storage

After the Casdoor verification is passed, it will redirect back to your application with token.

You can opt in to use cookie to storage the token.

```
CasdoorSDK.exchangeForAccessToken()
  .then((res) => {
    if (res && res.access_token) {
      //Get Token
      return CasdoorSDK.getUserInfo(res.access_token);
    }
  })
  .then((res) => {
    // Storage Token
```

You can refer to the Casdoor official documentation for the [How to use Casdoor SDK](#).

Step 4: Add Middleware Authentication Function

when users attempt to access a protected route, Middleware Authentication function verifies their identity. If the user is not authenticated, they are redirected to a login page or denied access.

Example

```
//protected route
const protectedRoutes = ["/profile"];
const casdoorUserCookie = req.cookies.get("casdoorUser");
const isAuthenticated = casdoorUserCookie ? true : false;

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

Nuxt

[nuxt-auth](#) is an example of how to integrate casdoor in a nuxt project. We will guide you through the steps below. Many steps are similar to nextjs-auth.

Step 1: Deploy Casdoor

Firstly, Casdoor should be deployed.

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

After a successful deployment, make sure the following:

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

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

Step 2: Add Middleware

Middleware allows you to run code before a request is completed. Then, based on the incoming request, you can modify the response by rewriting, redirecting, modifying the request or response headers, or responding directly.

Create `.js` or `.ts` files in `middleware` directory in the root of your project to define Middleware. And the filenames are identified as the names of middleware.

For example, in [nuxt-auth](#), we create a file named `myMiddleware.js` in `middleware` directory, which can be referenced as `myMiddleware` in other places like `nuxt.config.js`.

Example

```
//define which paths Middleware will run on
const protectedRoutes = ["/profile"];

export default function ({route, redirect}) {

  if (protectedRoutes.includes(route.path)) {
    //redirect the incoming request to a different URL
    redirect('/login');
  }
}
```

To make middleware work, you should add router in `nuxt.config.js`, like that:

```
export default {
  // other configurations

  // what to add
  router: {
    middleware: ['myMiddleware'] // replace to your middleware name
  },
}
```

See [nuxt official documentation middleware](#) for more details.

Step 3: Use Casdoor SDK

1. Install the SDK

First, install `casdoor-js-sdk` via NPM or Yarn:

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

Or:

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

2. Initializing the SDK

Then initialization 6 string-type parameters in the following order:

Name	Required	Description
serverUrl	Yes	Casdoor Server URL, such as <code>http://localhost:8000</code>
clientId	Yes	Application client ID
clientSecret	Yes	Application client secret
organizationName	Yes	Application organization

Name	Required	Description
appName	Yes	Application name
redirectPath	Yes	redirected URL

Example

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

CAUTION

Replace the configuration values with your own Casdoor instance, especially the `clientId`, `clientSecret`, and `serverUrl`.

3.Redirect to the Login Page

When you need to authenticate users who access your app, you can send the target URL and redirect to the login page provided by Casdoor.

Make sure you have added the callback URL (e.g. <http://localhost:8080/callback>) in the application configuration beforehand.


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

4. Get Token and Storage

After the Casdoor verification is passed, it will redirect back to your application with token.

You can opt in to use cookie to storage the token.

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

You can refer to the Casdoor official documentation for the [How to use Casdoor SDK](#).

Step 4: Add Middleware Authentication Function

when users attempt to access a protected route, Middleware Authentication function verifies their identity. If the user is not authenticated, they are redirected to a login page or denied access.

Example

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

const protectedRoutes = ["/profile"];

export default function ({route, redirect}) {
  const casdoorUserCookie = Cookies.get('casdoorUser');
  const isAuthenticated = !!casdoorUserCookie;

  if (!isAuthenticated && protectedRoutes.includes(route.path)) {
    redirect('/login');
  }
}
```

OAuth 2.0

Introduction

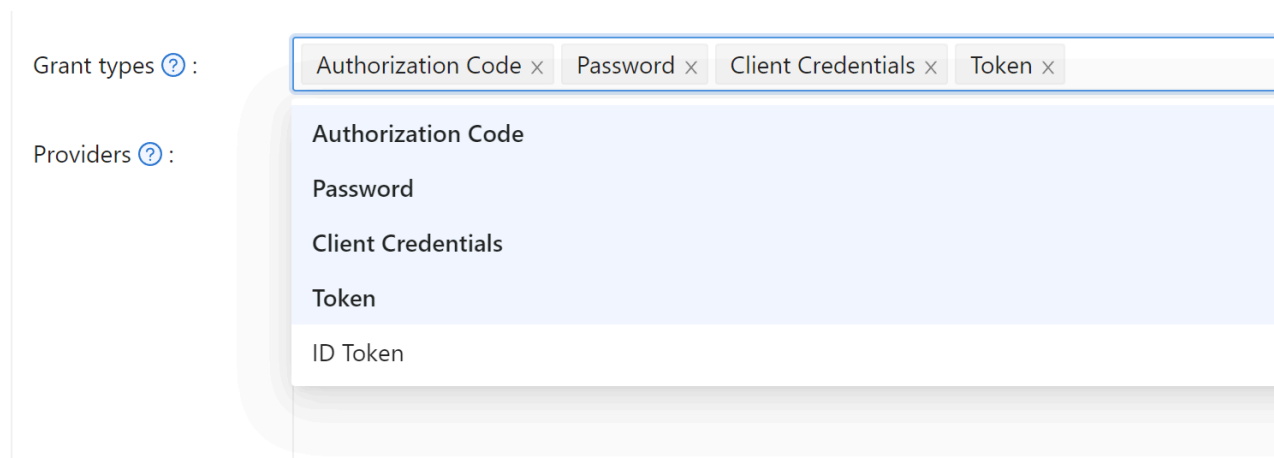
Casdoor supports using Access Token to authenticate clients. In this section, we will show you how to obtain an Access Token, how to verify an Access Token, and how to use an Access Token.

How to Get an Access Token

There are two ways to obtain an Access Token: you can use the [Casdoor SDKs](#). For detailed information, please refer to the SDK documentation. Here, we will mainly show you how to use the API to get the Access Token.

Casdoor supports four OAuth grant types: [Authorization Code Grant](#), [Implicit Grant](#), [Resource Owner Password Credentials Grant](#), and [Client Credentials Grant](#).

For security reasons, the Casdoor app has the authorization code mode turned on by default. If you need to use other modes, please go to the appropriate app to set it.



Authorization Code Grant

First, redirect your users to:

```
https://<CASD00R_HOST>/login/oauth/authorize?  
client_id=CLIENT_ID&  
redirect_uri=REDIRECT_URI&  
response_type=code&  
scope=openid&  
state=STATE
```

Available scopes

Name	Description
openid (no scope)	sub (user's id), iss (issuer), and aud (audience)
profile	user profile info, including name, displayName, and avatar
email	user's email address
address	user's address
phone	user's phone number

ⓘ INFO

Your OAuth Application can request the scopes in the initial redirection. You can specify multiple scopes by separating them with a space using %20:

```
https://<CASD00R_HOST>/login/oauth/authorize?  
client_id=...&  
scope=openid%20email
```

For more details, please see the [OIDC standard](#)

After your user has authenticated with Casdoor, Casdoor will redirect them to:

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

Now that you have obtained the authorization code, make a POST request to:

```
https://<CASD00R_HOST>/api/login/oauth/access_token
```

in your backend application:

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

You will get the following response:

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

NOTE

Casdoor also supports the PKCE feature. When getting the authorization code, you can add two parameters to enable PKCE:

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

When getting the token, you need to pass the `code_verifier` parameter to verify PKCE. It is worth mentioning that with PKCE enabled, `Client_Secret` is not required, but if you pass it, it must be the correct value.

Implicit Grant

Maybe your application doesn't have a backend, and you need to use Implicit Grant. First, you need to make sure you have Implicit Grant enabled, then redirect your users to:

```
https://<CASD00R_HOST>/login/oauth/
authorize?client_id=CLIENT_ID&redirect_uri=REDIRECT_URI&response_type=token&scope=openid&state=STATE
```

After your user has authenticated with Casdoor, Casdoor will redirect them to:

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

Casdoor also supports the `id_token` as `response_type`, which is a feature of OpenID.

Resource Owner Password Credentials Grant

If your application doesn't have a frontend that redirects users to Casdoor, then you may need this.

First, you need to make sure you have Password Credentials Grant enabled and send a POST request to:

```
https://<CASD00R_HOST>/api/login/oauth/access_token
```

```
{
  "grant_type": "password",
  "client_id": ClientId,
```

You will get the following response:

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

Client Credentials Grant

You can also use Client Credentials Grant when your application does not have a frontend.

First, you need to make sure you have Client Credentials Grant enabled and send a POST request to

`https://<CASD00R_HOST>/api/login/oauth/access_token`:

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

You will get the following response:

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

It is important to note that the AccessToken obtained in this way differs from the first three in that it corresponds to the application rather than to the user.

Refresh Token

Maybe you want to update your Access Token, then you can use the `refreshToken` you obtained above.

First, you need to set the expiration time of the Refresh Token in the application (default is 0 hours), and send a POST request to `https://<CASD00R_HOST>/api/login/oauth/refresh_token`

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

You will get a response like this:

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

How to Verify Access Token

Casdoor currently supports the [token introspection](#) endpoint. This endpoint is protected by Basic Authentication (ClientId:ClientSecret).

```
POST /api/login/oauth/introspect HTTP/1.1
Host: CASD00R_HOST
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Authorization: Basic Y2xpZW50X2lk0mNsawVudF9zZWNYZXQ=

token=ACCESS_TOKEN&token_type_hint=access_token
```

You will receive the following response:

```
{
  "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"
}
```

How to Use `AccessToken`

You can use `AccessToken` to access Casdoor APIs that require authentication.

For example, there are two different ways to request `/api/userinfo`.

Type 1: Query parameter

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

Type 2: HTTP Bearer token

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

Casdoor will parse the access_token and return corresponding user information according to the `scope`. You will receive the same response, which looks like this:

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

If you expect more user information, add `scope` when obtaining the AccessToken in step [Authorization Code Grant](#).

Differences between the `userinfo` and `get-account` APIs

- `/api/userinfo`: This API returns user information as part of the OIDC protocol. It provides limited information, including only the basic information defined in OIDC standards. For a list of available scopes supported by Casdoor, please refer to the [available scopes](#) section.
- `/api/get-account`: This API retrieves the user object for the currently logged-in account. It is a Casdoor-specific API that allows you to obtain all the information of the `user` in Casdoor.

Using Casdoor as a CAS Server

Using Casdoor as a CAS Server

Casdoor can now be used as a CAS server. It currently supports CAS 3.0.

Overview

The CAS endpoint prefix in Casdoor is `<Casdoor endpoint>/cas/<organization name>/<application name>`. Here is an example using the endpoint `https://door.casdoor.com` with an application named `cas-java-app` under the organization `casbin`:

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

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

An Example

Here is an official example [GitHub Repository](#) that contains a web app and utilizes the official CAS Java client [GitHub Repository](#). By going through this example, you can learn how to connect to Casdoor via CAS.

NOTE

Note: Currently, Casdoor only supports all three versions of CAS: CAS 1.0, 2.0, and 3.0.

The CAS configuration is located in `src/main/webapp/WEB-INF/web.yml`.

By default, this app uses CAS 3.0, which is specified by the following configurations:

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

If you want to protect this web app using CAS 2.0, change the CAS Validation Filter to the following:

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

For CAS 1.0, use the following:

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

For all instances of the `casServerUrlPrefix` parameter, change them to:

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

For all instances of the `casServerLoginUrl` parameter, change them to:

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

If you need to customize more configurations, see the [Java CAS client GitHub Repository](#) for detailed information.

SAML

Overview

Using Casdoor as SAML IdP

AWS Client VPN

Using Casdoor as a SAML IdP

Keycloak

Using Casdoor as a SAML IdP

Google Workspace

Using Casdoor as a SAML IdP

Appgate (POST)

How to Use Casdoor as SAML IdP for Appgate

Tencent Cloud

Using Casdoor as a SAML IdP

Overview

Casdoor can now be used as a SAML IdP. Up to this point, Casdoor has supported the main features of SAML 2.0.

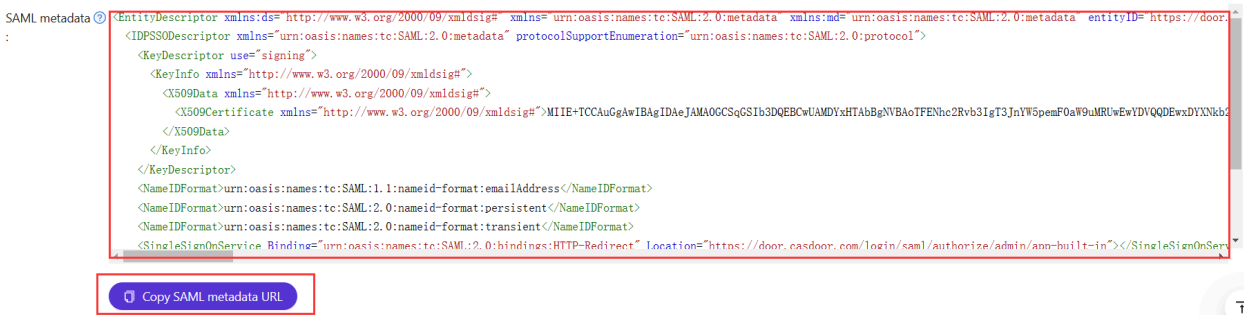
Configuration in SP

In general, the SP requires three required fields: `Single Sign-On`, `Issuer`, and `Public Certificate`. Most SPs can obtain these fields by uploading the XML Metadata file or the XML Metadata URL for autocompletion.

The metadata of the 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`, and it 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
```

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



Configuration in Casdoor IdP

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

! INFO

If you fill in 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 overwrite the `AssertionConsumerServiceURL` in `SAMLRequest`.

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

Grant types [?](#) :

Authorization Code × Password ×

SAML Reply URL

[?](#) :

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

Enable SAML
compress [?](#) :



- **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.

Redirect URLs [?](#) :

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

User profile

After successfully logging in, the user profile in the returned `SAMLResponse` from Casdoor 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

`gosaml2` is a SAML 2.0 implementation for Service Providers based on `etree` and `goxmldsig`, a pure Go implementation of XML digital signatures. We use this library to test the SAML 2.0 in Casdoor as shown 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)  
    }  
  
    metadata := &types.EntityDescriptor{}  
    err = xml.Unmarshal(rawMetadata, metadata)  
    if err != nil {
```


Run the above code, and the console will display the following message.

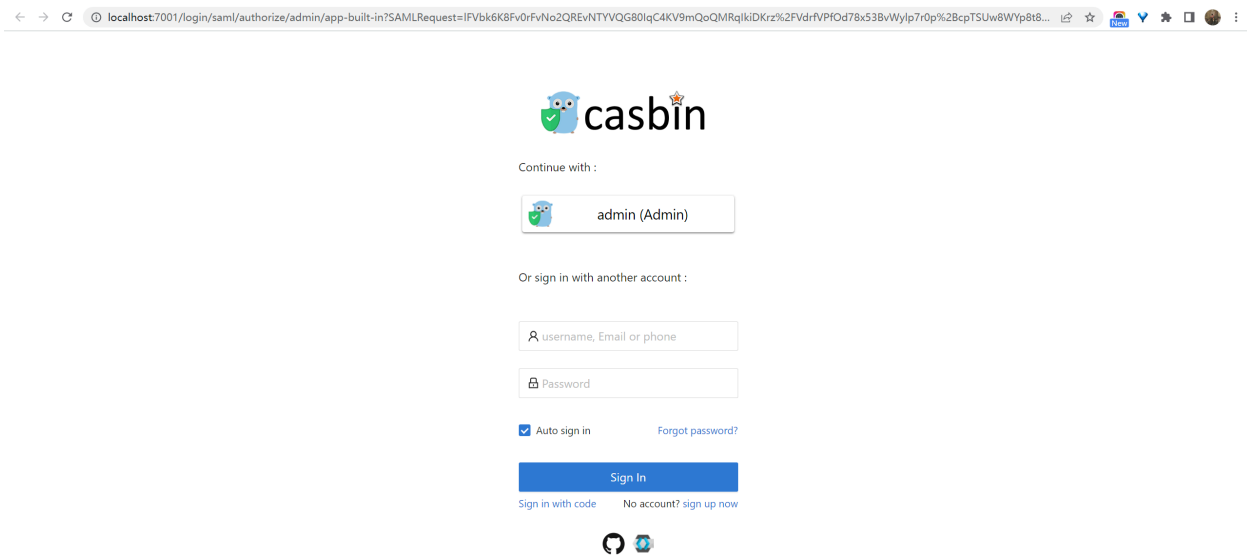
Visit this URL To Authenticate:

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

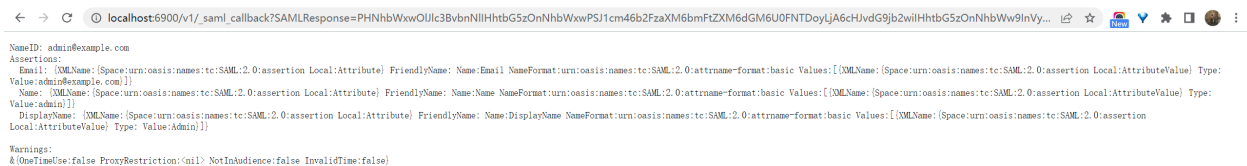
Supply:

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

Click the URL to authenticate, and the login page of Casdoor will be displayed.



After authenticating, you will receive the response messages as shown below.



AWS Client VPN

Casdoor as a SAML IdP in AWS Client VPN

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

Prerequisites

To complete this setup, you will need:

- An AWS Account with administrative rights to access configuration settings of the service provider.
- An Amazon VPC with an EC2 instance
 - [Setting up the VPC](#)
 - [Launching an EC2 instance](#)
 - In the instance Security Group, allow ICMP traffic from the VPC CIDR range - this is needed for testing.
- A private certificate imported into [AWS Certificate Manager \(ACM\)](#)
 - [Generating and importing a certificate to ACM](#)
- A Windows or Mac system running the latest AWS Client VPN software.
 - [Download the software](#)

Configure SAML Application

- In the Casdoor Application, set the **Redirect URL** to `urn:amazon:webservicess:clientvpn`.

Tags [?](#) :

Client ID [?](#) :

Client secret [?](#) :

Cert [?](#) :

Redirect URLs [?](#) :

Redirect URLs [Add](#)

Redirect URL
🔗 urn:amazon:webservicess:clientvpn


Token format [?](#) :

Token expire [?](#) :

Refresh token expire [?](#) :

Enable password [?](#) :

- Set the **SAML reply URL** to `http://127.0.0.1:35001`.


Signup HTML  :

Signin HTML  :

Grant types  :

SAML reply URL  :

Enable SAML compression  :

SAML metadata  :

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

- Save the content in the **SAML metadata** as an XML file.

```
SAML metadata <EntityDescriptor xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns="urn:oasis:names:tc:SAML:2.0:metadata" xmlns:md="urn:oasis:names:tc:SAML:2.0:md"
<IDPSSODescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata" protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
<KeyDescriptor use="signing">
<KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
<X509Data xmlns="http://www.w3.org/2000/09/xmldsig#">
<X509Certificate xmlns="http://www.w3.org/2000/09/xmldsig#">MIIE+TCCAuGgAwIBAgIDAeJAMA0GCSqGSIb3DQEBCwUAMDYxHTAbBgNVBAoTFENhc2Rvb3JlIGt3JnYwI
</X509Data>
</KeyInfo>
</KeyDescriptor>
<NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</NameIDFormat>
<NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:persistent</NameIDFormat>
<NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient</NameIDFormat>
<SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location="https://test.v2t1.com/login/saml/authorize/admin/app-
</SingleSignOnService>
</IDPSSODescriptor>
</EntityDescriptor>
```

 Copy SAML metadata URL

Configure AWS

Configure Casdoor as an AWS Identity Provider

1. Open the IAM console and select Identity providers from the navigation bar.
2. Click Create a Provider.

3. Specify SAML for the Provider Type, add a unique name for this provider, and upload the metadata document - the same file you saved from the Casdoor Application in the previous section.

4. Click Next Step. On the next screen, click Create.

Identity and Access Management (IAM)

Search IAM

Dashboard

▼ Access management

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

▼ Access reports

- Access analyzer
 - Archive rules
 - Analyzers
 - Settings

IAM > Identity providers

Have you considered using AWS IAM Identity Center?

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

Identity providers (1)

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

Delete Add provider

Filter by Type

Search All Types

Provider	Type	Creation time
casdoor	SAML	Yesterday

Add an Identity provider [Info](#)

Configure provider

Provider type [Info](#)

SAML

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

OpenID Connect

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

Provider name

Enter a meaningful name to identify this provider

casdoor

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

Metadata document [Info](#)

 Choose file

3

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

Create an AWS Client VPN Endpoint

1. Open the Amazon VPC console in an AWS Region of your choice.
2. On the left-hand side navigation, select **Client VPN Endpoints** under **Virtual Private Network (VPN)**.
3. Click **Create Client VPN Endpoint**.
4. Enter the IP range for your remote users in the **Client IPv4 CIDR** field to allocate an IP range.
5. For **Server Certificate ARN**, select the certificate you created.
6. For **Authentication Options**, select **Use user-based authentication**, then **Federated authentication**.

7. For SAML provider ARN, select the identity provider you created.

8. Click Create Client VPN Endpoint.

Client VPN endpoints (1/1) [Info](#) Actions Download client configuration Create client VPN endpoint

Filter client VPN endpoints

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

Client IPv4 CIDR [Info](#)

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

172.31.32.0/20

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

Authentication information [Info](#)

Server certificate ARN

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

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

Authentication options

Choose one or a combination of authentication methods to use.

Use mutual authentication

Use user-based authentication

User-based authentication options

Active directory authentication

Federated authentication

SAML provider ARN

The ARN of SAML provider.

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

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

Select self-service SAML provider ARN

Associate a Client VPN with a Target VPC

1. Select Target network associations in the Client VPN options, then click Associate target network.
2. From the drop-down menu, select the target VPC and subnet you want to associate your endpoint with.

The screenshot displays the AWS IAM console interface for managing Client VPN endpoints. On the left, a navigation pane lists various VPN-related options, with 'Client VPN endpoints' selected. The main content area shows a table of Client VPN endpoints. One endpoint, 'cvpn-endpoint-06e947f15ddf5687c', is shown in an 'Available' state with a Client CIDR of '172.31.32.0/20'. Below this, a modal window titled 'cvpn-endpoint-06e947f15ddf5687c' is open, showing the 'Target network associations' tab. This tab contains a table of associations. One association is listed with ID 'cvpn-assoc-0bf639762212d5a04', state 'Associated', network ID 'subnet-0596ebfd975cdd125', security group 'sg-09d2a80e3c2795429', and endpoint ID 'cvpn-endpoint-06e947f15d'. The 'Associate target network' button is highlighted with a red box.

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

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

Configure SAML Group-Specific Authorization

1. Choose the Authorization rules tab in your Client VPN options and click Add Authorize rule.
2. For Destination network to enable, specify the IP address of your EC2 instance created in the prerequisites. For example, `172.31.16.0/20`.
3. Under Grant access to, select Allow access to users in a specific access group. For example, `casdoor`.
4. Provide an optional description and click Add authorization rule.

Add authorization rule [Info](#)

Add authorization rules to grant clients access to the networks.

Details

Client VPN endpoint ID
cvpn-endpoint-06e947f15ddf5687c

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

172.31.16.0/20 2

Grant access to:

Allow access to all users

Allow access to users in a specific access group 3

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

casdoor

Description - *optional*
A brief description of the authorization rule.

description

4

Cancel **Add authorization rule**

Connect to Client VPN

1. Select the Client VPN endpoint you just created. It should now be in the Available state.
2. Click **Download Client Configuration** to download the configuration profile to your desktop.
3. Open the AWS Client VPN desktop app on your machine.
4. In the top menu, select File and Manage Profiles.
5. Click Add Profile and point to the recently downloaded file.

6. You should now see the profile in the list on the AWS Client VPN software. Select it and click Connect.

The screenshot displays the AWS Management Console interface for Client VPN endpoints. On the left, a navigation sidebar shows 'Virtual private cloud' and 'Security' sections. The main content area is titled 'Client VPN endpoints (1/1) Info'. At the top right, there are buttons for 'Download client configuration' (highlighted with a red box) and 'Create client VPN endpoint'. Below the title bar is a search filter and a table with one entry:

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

Below the table, the details for the selected endpoint 'cvpn-endpoint-06e947f15ddf5687c' are shown. The 'Details' tab is active, displaying the following information:

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

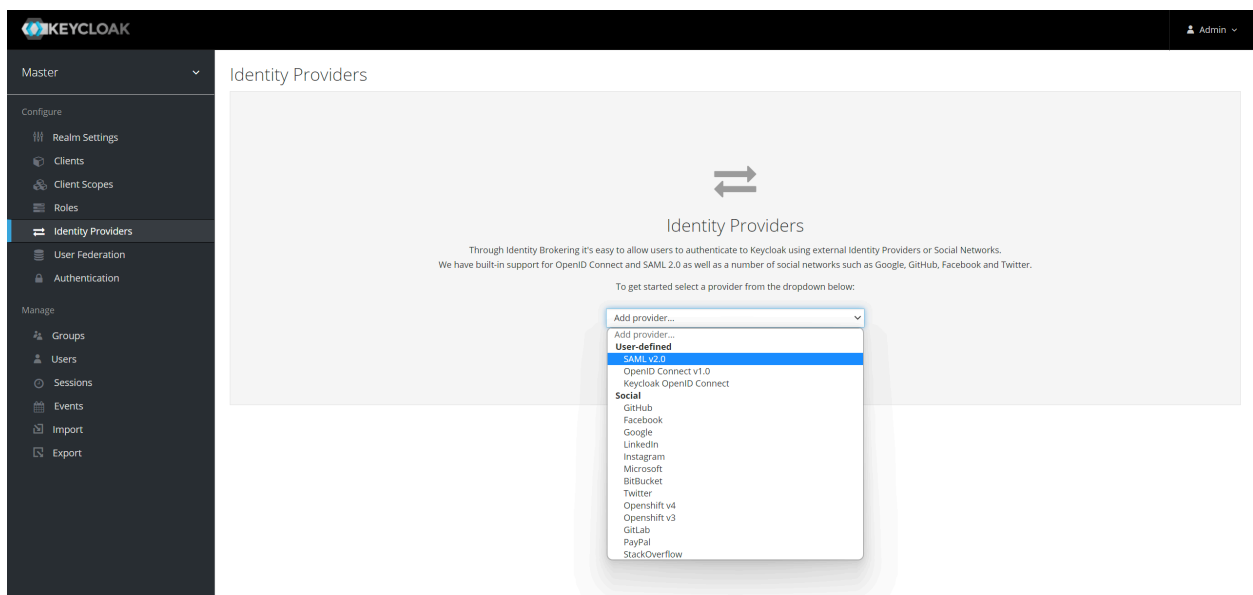
Keycloak

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.

Adding SAML IdP in Keycloak

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



! INFO

You can visit the Keycloak SAML Identity Providers [documentation](#) to get more detailed information.

Enter the Alias and the Import from URL in the Keycloak IdP edit page. The

content of the **Import from URL** can be found on the Casdoor application edit page. Click **Import** and the SAML config will be filled automatically.

▼ Import External IDP Config ?

Import from URL

Import

Import from file

Save

Remember the Service Provider Entity ID and save the configuration.

Configuring the SAML application in Casdoor

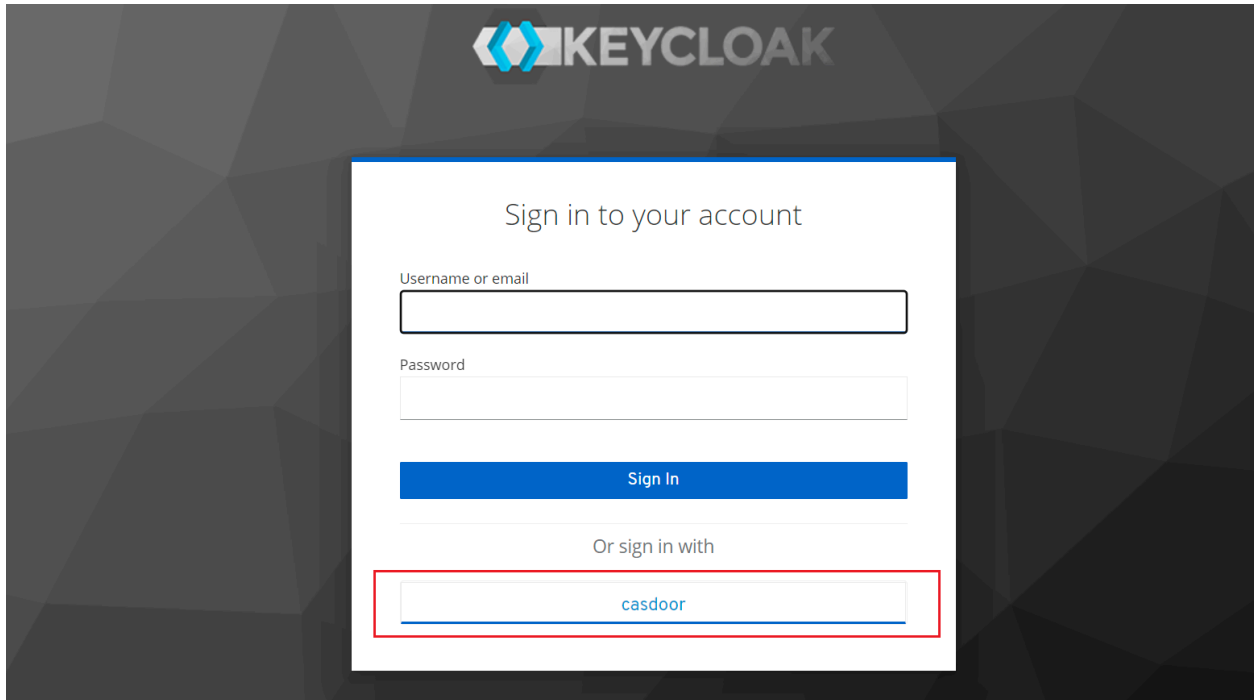
In the application edit page, add a redirect URL which contains the Service Provider Entity ID from Keycloak. Also, make sure to enable SAML compress for Keycloak.

Enable SAML compress

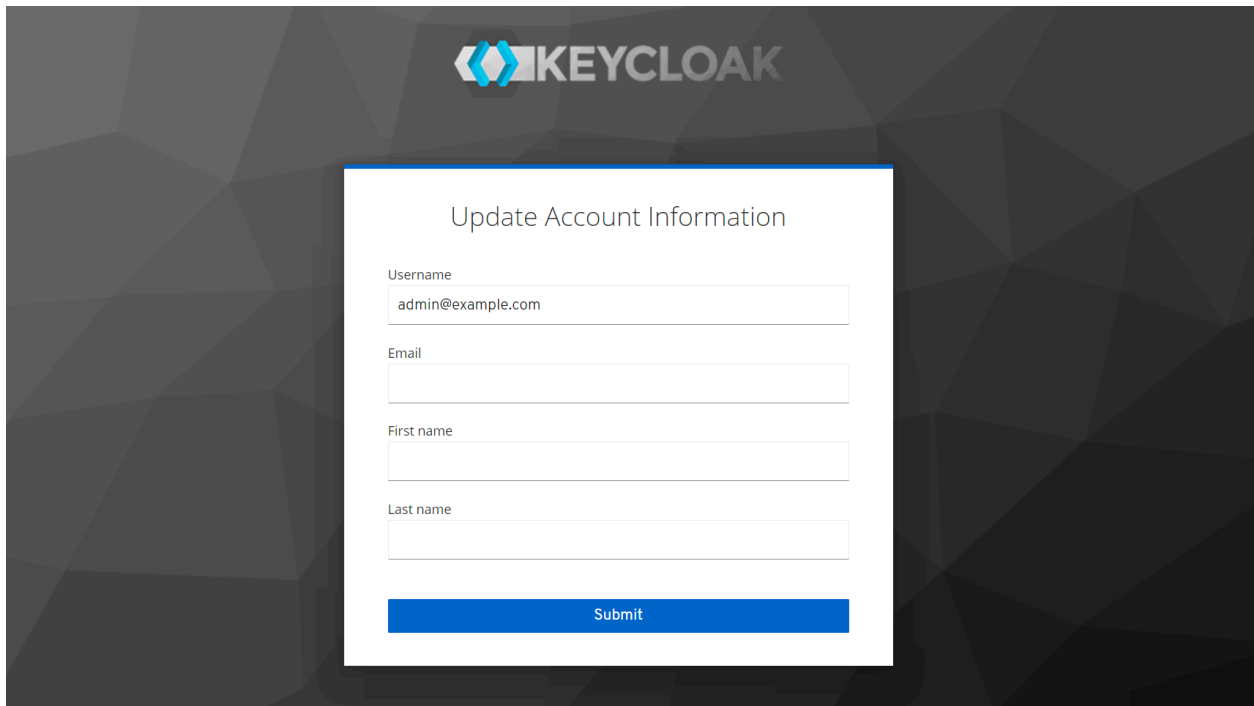
```
SAML metadata: <EntityDescriptor xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns="urn:oasis:names:tc:SAML:2.0:metadata" xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" entityID="http://localhost:8000">
  <IDPSSODescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata" protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <KeyDescriptor use="signing">
      <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        <X509Data xmlns="http://www.w3.org/2000/09/xmldsig#">
          <X509Certificate xmlns="http://www.w3.org/2000/09/xmldsig#"MIIE+TCCAuGgAwIBAgIDAeJAMAOGCSqGS1b3DQEBCqUAMDFxHTAbBgNVBaoTFENhc2Rvb3I3JnIWF5pemF0a9uMRUvEwYDVQQDEwYXNkb29yIENlcnQwHhcNMjExMDE1MDgxM
        </X509Data>
      </KeyInfo>
    </KeyDescriptor>
    <NameIDFormat urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</NameIDFormat>
    <NameIDFormat urn:oasis:names:tc:SAML:2.0:nameid-format:persistent</NameIDFormat>
    <NameIDFormat urn:oasis:names:tc:SAML:2.0:nameid-format:transient</NameIDFormat>
  </IDPSSODescriptor>
  <SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location="http://localhost:7001/loia/saml/authorize/admin/app-built-in"></SingleSignOnService>
</EntityDescriptor>
```

Logging in using Casdoor SAML

Open the Keycloak login page and you will find an additional button that allows you to log in to Keycloak using the Casdoor SAML provider.



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



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

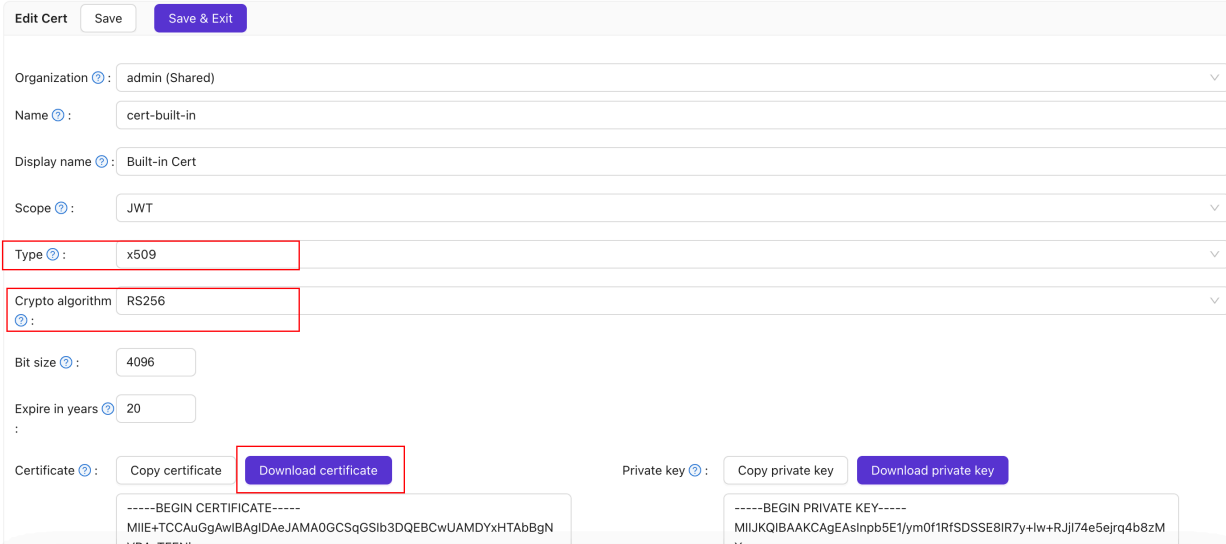
Google Workspace

Casdoor as a SAML IdP in Google Workspace

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

Add Certificate

In Casdoor, add a certificate of type X.509 with RSA crypto algorithm and download it.



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


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

The certificate content is displayed as follows:


```
-----BEGIN CERTIFICATE-----
MIIE+TCCAuGgAwIBAgIDAeJAMA0GCSqGSIb3DQEBCwUAMDYxHTAbBgN
VBAoTFENh
-----BEGIN PRIVATE KEY-----
MIJKQIBAAKAgEAslnpb5E1jym0f1RfSDSSE8IR7y+lw+RJJj74e5ejrq4b8zM
Y
```


Configure SAML Application

On the application edit page, select the certificate you just created. Add the domain name of the Google application you will use in the **Redirect URLs**, such as google.com.

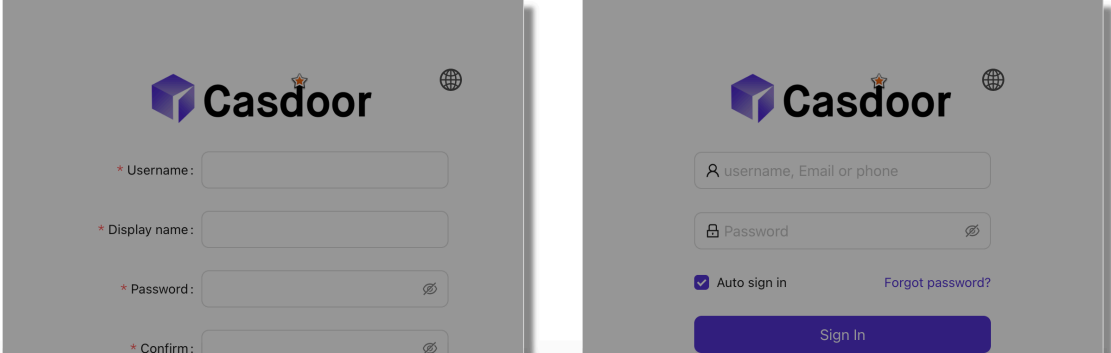
Providers  :

Providers [Add](#)

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

Preview  :

[Copy signup page URL](#) [Copy signin page URL](#)




Add Third-Party SAML IdP for Google Workspace

In the Google Workspace Admin console, navigate to **Security** and then **Overview**. Look for the **SSO with third-party IdP** section. Click on "Add SSO profile" to access the editing page. Check the "Set up SSO with third-party identity provider" checkbox. Paste the copied sign-in page URL into the **Sign-in page URL** and **Sign-out page URL** fields. Upload the certificate downloaded in the previous step. Click "Save" to save the changes.

The screenshot displays the Google Admin console interface. On the left is a navigation sidebar with categories like Home, Directory, Devices, Apps, and Security. The main content area is titled "Third-party SSO profile for your organisation". It features a summary card for "Single Sign-On (SSO) with third-party Identity Providers (IDPs)" and a configuration panel. The configuration panel includes a checked checkbox for "Set up SSO with third-party identity provider" and instructions to provide a sign-in page URL, a sign-out page URL, and a verification certificate. The sign-in and sign-out URLs are highlighted with red boxes and are both "https://localhost/login/oauth/authorize?client_id=12". The verification certificate section shows a message "A certificate file has been uploaded" with a "REPLACE CERTIFICATE" button and a red "upload your certificate" link.

Add Users for Testing

In Google Workspace, create a user with the username "test" (you can customize the username, this is just an example).

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

As an example using "google.com," follow these steps:

1. Click on the login button on the Google.com page. Enter the user's email address to initiate the login process.
2. You will be redirected to the Casdoor page. On the Casdoor page, enter the corresponding email address and password.
3. If the login is successful, you will be redirected back to google.com.



Google Search

I'm Feeling Lucky

Google offered in: [日本語](#)

Appgate (POST)

Casdoor as a SAML IdP in Appgate

Appgate accepts the `SAMLResponse` sent by a POST request. If you use another Service Provider (SP) that also supports a POST request, you can refer to this document.

Casdoor Configuration

Go to your Casdoor account 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.

Redirect URLs [?](#):

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


- Reply URL – Type in the URL of the ACS (Assertion Consumer Service) that verifies the SAML response. Refer to the table below.


Grant types  :

Authorization Code × Password ×

SAML Reply URL

 :

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



Enable SAML
compress  :




Administrator Authentication	User Authentication
Redirect URL = "AppGate"	Redirect URL = "AppGate Client"
SAML Reply URL = https://mycontroller.your-site-url.com/admin/saml	SAML Reply URL = https://redirectserver.your-site-url.com/saml


Download the XML metadata file

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

Enable SAML 
compress  :

SAML metadata 

```
<KeyDescriptor use="signing">
  <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
    <X509Data xmlns="http://www.w3.org/2000/09/xmldsig#">
      <X509Certificate xmlns="http://www.w3.org/2000/09/xmldsig#">MIIE+TCCAuGgAwIBAgIDAeJAMAOGCSqGSIb3DQEBCwUAMDYxHTAbBgNVBAoTFENhc2Rv3Igt3JnYw5pemF0aF9uMRUwEwYDQDEwYDXXNhbz
    </X509Data>
  </KeyInfo>
</KeyDescriptor>
<NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</NameIDFormat>
<NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:persistent</NameIDFormat>
<NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient</NameIDFormat>
<SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location="https://door.casdoor.com/login/saml/authorize/admin/app-gitea"></SingleSignOnService>
<Attribute Name="Email" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic" FriendlyName="E-Mail" xmlns="urn:oasis:names:tc:SAML:2.0:assertion"></Attribute>
<Attribute Name="DisplayName" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic" FriendlyName="displayName" xmlns="urn:oasis:names:tc:SAML:2.0:assertion"></Attribute>
```

 Copy SAML metadata URL

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.

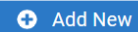
	Administrator Authentication
Name	Enter a unique name, e.g. "Casdoor SAML Admin".
Single Sign-on URL	See below
Issuer	See below
Audience	Type in the Redirect URL from the Casdoor application
Public Certificate	See below

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

Map Attributes

Map the Name to username. Your completed form should look something like this:

Map Attributes to User Claims

 Add New

Name mapped to claim username

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

You need to modify the access policy to allow administrators to log in to Appgate using the SAML IdP. Enter the Builtin Administrator Policy:

Your completed form should look something like this:

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	

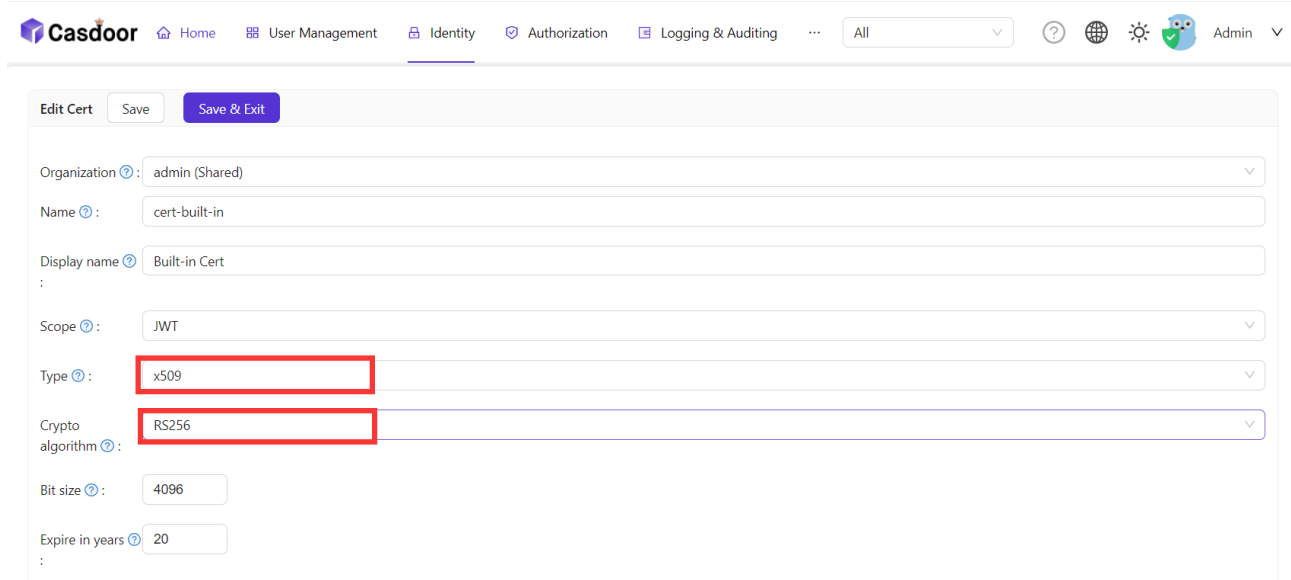
Tencent Cloud

Casdoor as a SAML IdP in Tencent Cloud

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

Copy Saml MetaData

In Casdoor, add a certificate of type X.509 with RSA crypto algorithm.



Organization: admin (Shared)

Name: cert-built-in

Display name: Built-in Cert

Scope: JWT

Type: x509

Crypto algorithm: RS256

Bit size: 4096

Expire in years: 20

Then copy the SamlMetadata in Casdoor.

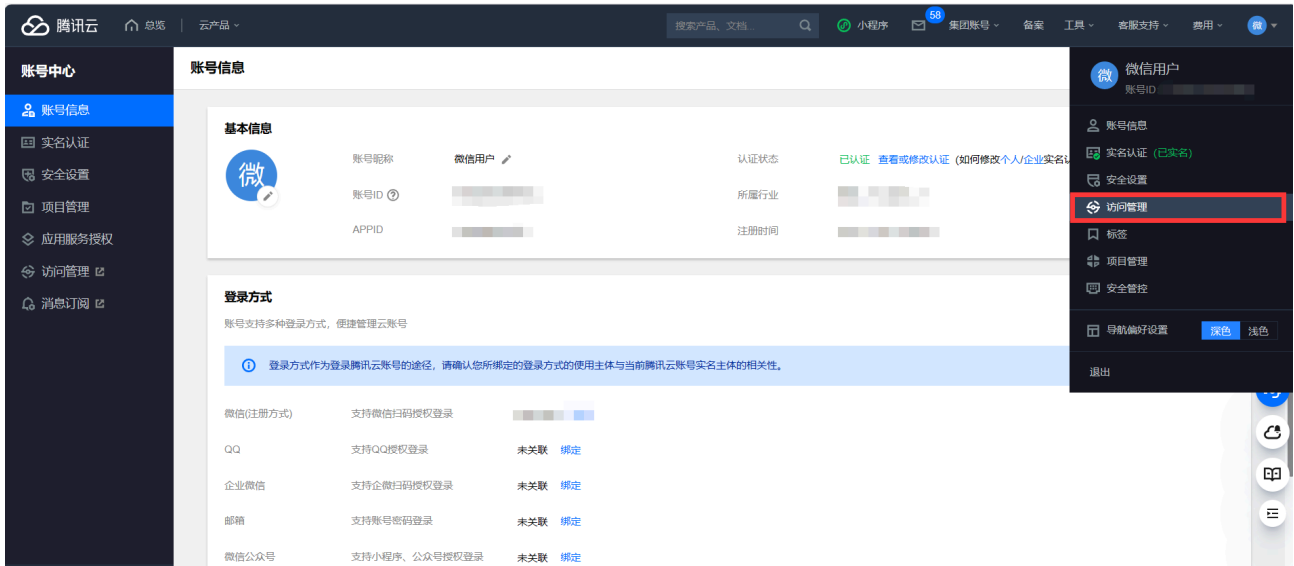


```
<EntityDescriptor xmlns:ds= "http://www.w3.org/2000/09/xmldsig#" xmlns= "urn:oasis:names:tc:SAML:2.0:metadata" xmlns:md= "urn:oasis:names:tc:SAML:2.0:metadata" entityID= "http://localhost:7001/login/saml/authorize/admin/application_tencent_cloud">  
<IDPSSODescriptor xmlns= "urn:oasis:names:tc:SAML:2.0:metadata" protocolSupportEnumeration= "urn:oasis:names:tc:SAML:2.0:protocol">  
<KeyDescriptor use= "signing">  
<KeyInfo xmlns= "http://www.w3.org/2000/09/xmldsig#">  
<X509Data xmlns= "http://www.w3.org/2000/09/xmldsig#">  
<X509Certificate xmlns= "http://www.w3.org/2000/09/xmldsig#">MIIE+TCCAuGgAwIBAgIDAeJAMA0GCSqSgSlb3DQEBCwUAMDYxHTAbBgNVBaoTFENhc2RvbnUzInlW5pemF0aW9uMRUwEwYDVQDEwYXNkb29  
</X509Certificate>  
</X509Data>  
</KeyInfo>  
</KeyDescriptor>  
</IDPSSODescriptor>  
<NameIDFormat urn: "urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress" />  
<NameIDFormat urn: "urn:oasis:names:tc:SAML:2.0:nameid-format:persistent" />  
<NameIDFormat urn: "urn:oasis:names:tc:SAML:2.0:nameid-format:transient" />  
<SingleSignOnService Binding= "urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect" Location= "http://localhost:7001/login/saml/authorize/admin/application_tencent_cloud" /></SingleS
```

Providers: Providers Add

Adding SAML IdP in Tencent Cloud

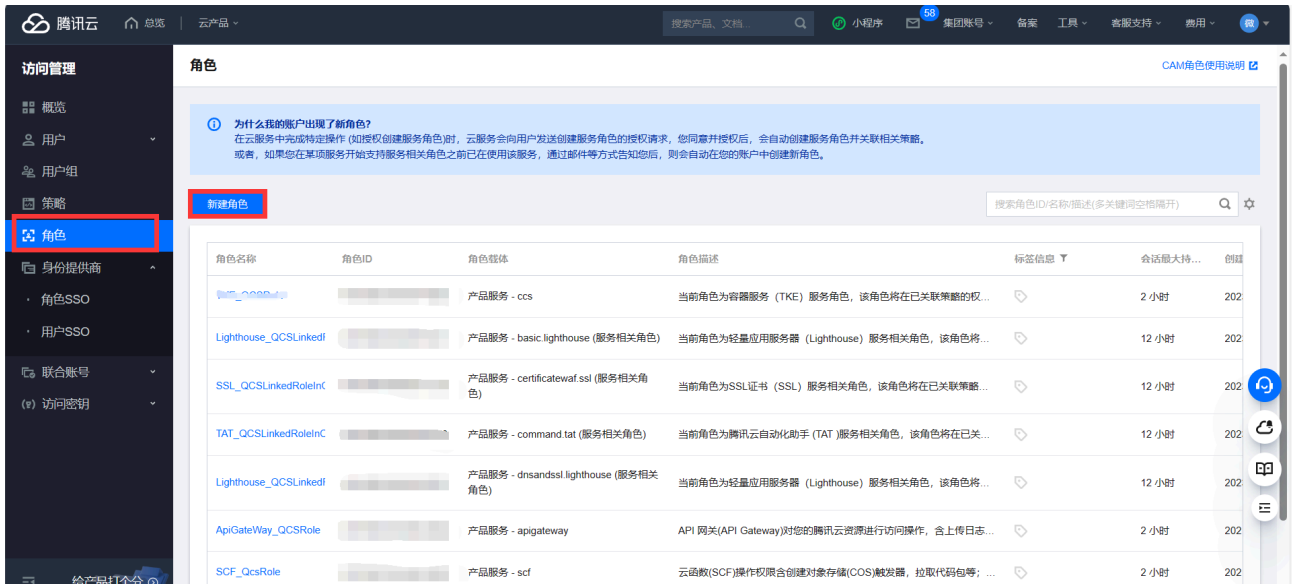
Log in to Tencent Cloud and enter the access management interface.



Create a new Identity Providers and upload the previously copied saml metadata to Tencent Cloud.

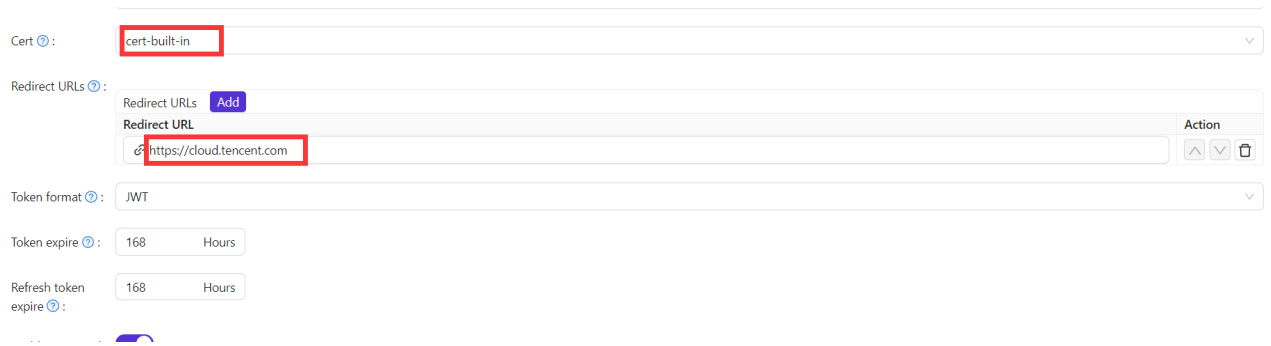


Then Create a new ROLE and select the previously Identity Providers as idp provider.

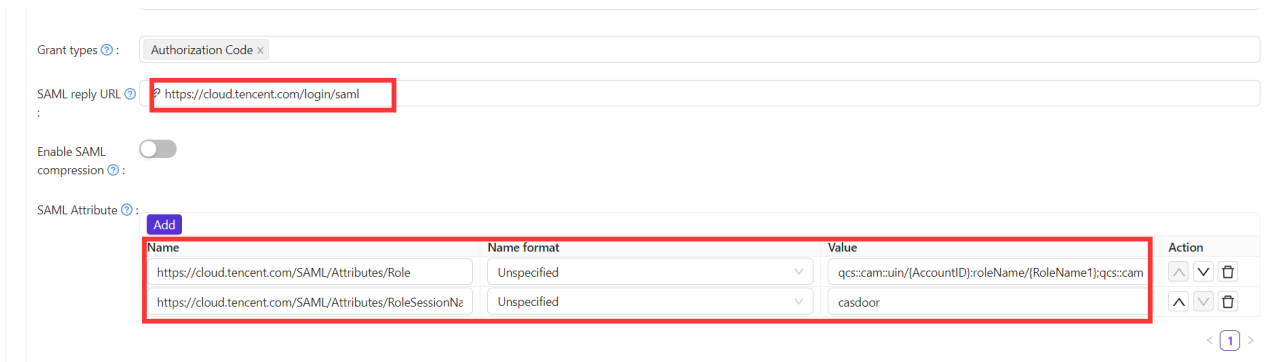


Configuring the SAML application in Casdoor

On the application edit page, select the certificate you just created. Add the domain name of the Tencent Cloud application you will use in the Redirect URLs.



In the application edit page, enter the ACS URL and configure the Saml Attribute.



The configuration information for Saml Attribute is as follows:

Name	Name Format	Value
https://cloud.tencent.com/SAML/Attributes/Role	Unspecified	qcs::cam::uin/{AccountID}:roleName/{RoleName1};qcs::cam::uin/{AccountID}:roleName/{RoleName2};qcs::cam::uin/provider/{ProviderName}
https://cloud.tencent.com/SAML/Attributes/RoleSessionName	Unspecified	casdoor

INFO

- In the Role source attribute, replace {AccountID}, {RoleName}, and {ProviderName} with the following content:
- Replace {AccountID} with your Tencent Cloud account ID, which can be viewed in the [Account Information - Console](#).
- Replace {RoleName} with the role name you created in Tencent Cloud, which can be viewed in the [Roles - Console](#).
- Replace {ProviderName} with the name of the SAML identity provider you created in Tencent Cloud, which can be viewed in the [Identity Providers - Console](#).

You can visit the Tencent Cloud SAML Identity Providers [documentation](#) to get more detailed information.

Logging in using Casdoor SAML

The general login steps for SAML are as follows: User → Tencent Cloud (not logged in) → Redirect to Casdoor for login → Tencent Cloud (logged in). Now, use code externally to simulate the first two steps and generate a URL that redirects to Casdoor. Sample code:

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

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

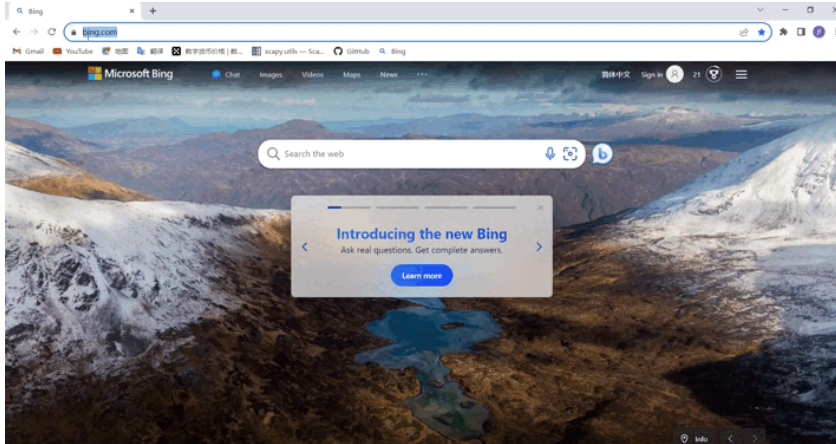
    metadata := &types.EntityDescriptor{}
    err = xml.Unmarshal(rawMetadata, metadata)
    if err != nil {
        panic(err)
    }

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

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

        idpCert, err := x509.ParseCertificate(certData)
        if err != nil {
            panic(err)
        }
    }
}
```

Once we run the code and obtain the **auth URL**, clicking on the URL will allow us to test the login. we provide a demo this process.



Face ID


Overview

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

Activation method

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

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

Account items 

Name	Visible	Regex	View rule	Modify rule	Action
Organization	<input checked="" type="checkbox"/>		Public	Admin	
ID	<input checked="" type="checkbox"/>		Public	Immutable	
Name	<input checked="" type="checkbox"/>		Public	Admin	
Display name	<input checked="" type="checkbox"/>		Public	Self	
Avatar	<input checked="" type="checkbox"/>		Public	Self	
User type	<input checked="" type="checkbox"/>		Public	Admin	
Password	<input checked="" type="checkbox"/>		Self	Self	
Email	<input checked="" type="checkbox"/>		Public	Self	
Phone	<input checked="" type="checkbox"/>		Public	Self	
Country code	<input checked="" type="checkbox"/>		Public	Admin	
Country/Region	<input checked="" type="checkbox"/>		Public	Self	
Location	<input checked="" type="checkbox"/>		Public	Self	
Affiliation	<input checked="" type="checkbox"/>		Public	Self	
Title	<input checked="" type="checkbox"/>		Public	Self	
Homepage	<input checked="" type="checkbox"/>		Public	Self	
Bio	<input checked="" type="checkbox"/>		Public	Self	
Tag	<input checked="" type="checkbox"/>		Public	Admin	
Signup application	<input checked="" type="checkbox"/>		Public	Admin	
Roles	<input checked="" type="checkbox"/>		Public	Immutable	
Permissions	<input checked="" type="checkbox"/>		Public	Immutable	
Groups	<input checked="" type="checkbox"/>		Public	Admin	
3rd-party logins	<input checked="" type="checkbox"/>		Self	Self	
Properties	<input checked="" type="checkbox"/>				
Is admin	<input checked="" type="checkbox"/>		Admin	Admin	
Is forbidden	<input checked="" type="checkbox"/>		Admin	Admin	
Is deleted	<input checked="" type="checkbox"/>		Admin	Admin	
Multi-factor authentication	<input checked="" type="checkbox"/>		Self	Self	
WebAuthn credentials	<input checked="" type="checkbox"/>		Self	Self	
Managed accounts	<input checked="" type="checkbox"/>		Self	Self	
Face ID	<input checked="" type="checkbox"/>		Public	Self	

Afterwards, you'll find the Face ID option under User, where users can upload their facial data to be used for

logging in

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

Face ID	Name	FaceIDData	Action
	y80ar	[0-1230016932027435; 0.07634161412715912; -0.017656213255429269; ...; 0.000207712109704739; 0.15079118768986154; 0.00523115823102713]	Delete

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

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

Name	Display name	Rule	Action
Password	Password	Non-LDAP	⬆️ ⬇️ ⬇️
Verification code	Email	Email only	⬆️ ⬇️ ⬇️
WebAuthn	WebAuthn		⬆️ ⬇️ ⬇️
Face ID	Face ID		⬆️ ⬇️ ⬇️

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

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



Password Email Face ID WebAuthn

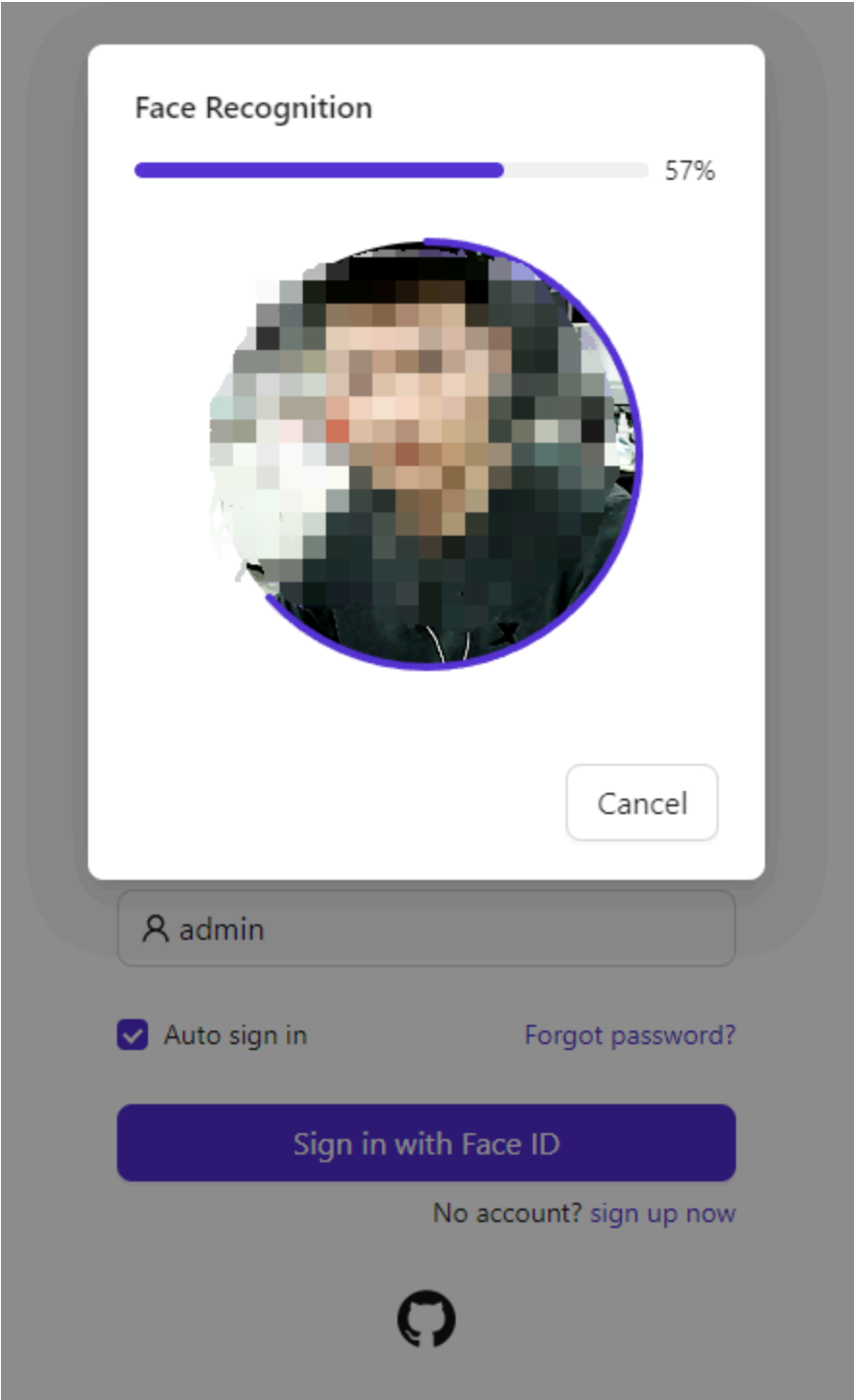
Auto sign in

[Forgot password?](#)

Sign in with Face ID

No account? [sign up now](#)





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

WebAuthn

Overview

We are delighted to inform Casdoor's customers that Casdoor now supports logging in with WebAuthn. This means that you can log in using your biological identifications such as fingerprints or facial recognition, or even U-disks, provided that your device supports these cool authorization methods and WebAuthn.

What is WebAuthn?

WebAuthn is the Web Authentication API, a specification written by the W3C and FIDO in collaboration with Google, Mozilla, Microsoft, Yubico, and others. This API allows servers to register and authenticate users using public key cryptography instead of a password. It enables servers to integrate with strong authenticators built into devices, such as Windows Hello or Apple's Touch ID.

To put it simply, WebAuthn requires users to generate a public key-private key pair and provide the public key to the website. When a user wants to log in to a website, the web generates a random number and asks the user to encrypt it with their private key and send the result back. Upon receiving the result, the website uses the public key to decrypt it. If the decrypted number matches the random number generated earlier, the user is considered a legitimate user and is granted access to log in. The combination of the public key and necessary information, like the username or information about the user's authorizer, is called the WebAuthn Credential, which is stored by the website.

The public key-private key pair is exclusively and uniquely associated with three pieces of information: the user's username, the user's authorizer, and the website's URL. This means that if the combination of (user's username, user's

authorizer, and the website's URL) is the same, the key pair should be identical, and vice versa.

For more detailed information about WebAuthn technology, you can visit <https://webauthn.guide/>.

How to use WebAuthn in Casdoor?

On the login page, you may have already noticed the option to log in using WebAuthn. However, if you don't have a WebAuthn credential yet (which can be likened to a WebAuth password), this tutorial will show you how to create and manage a credential and then log in using it.

Step 0: Modify the configurations and enable WebAuthn authentication

In the `conf/app.conf` file, you can find the following configuration:

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


Please ensure that this configuration exactly matches the URL of your website.



Note: Only HTTPS is supported for WebAuthn, unless you are using localhost.


Next, log in as the administrator and go to the edit page of your application. Turn on the "Enable WebAuthn signin" switch. By default, this feature is not enabled.

Step 1: Go to "My Account" page


Navigate to the account page. On this page, you should see the "Add WebAuthn Credential" button and a list displaying all the WebAuthn credentials you have previously registered.


Signup application :


3rd-party logins :  GitHub: (empty) [Link](#)


WebAuthn credentials : [WebAuthn credentials](#) [Add](#)


WebAuthn credentials	Action
0AUzbyDy1SCxNyW3vkNJP1feXhwm/pHBDmMOszRRNvg=	Delete


Roles :

Permissions :

Is admin :

Is global admin :

Is forbidden :

Is deleted :

Click the button and follow the instructions of your device to register a new credential in Casdoor. You can remove any credentials using the "delete" button in the list.

Step 2: Log in using WebAuthn

Before starting this step, make sure you have logged out of Casdoor.

Go to the login page, select the WebAuthn login method, enter your username, and click the login button. Follow the instructions of your device.

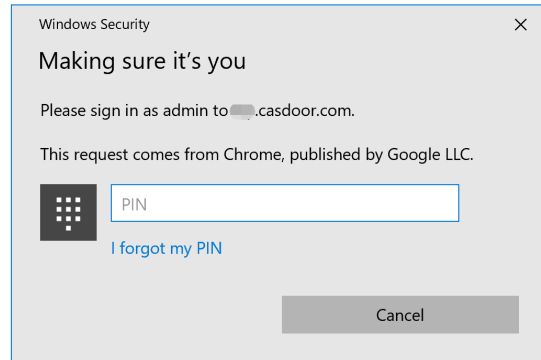
(For example, if you are using fingerprint and Windows Hello, you should see something like this)



Password [WebAuthn](#)

Auto sign in [Forgot password?](#)

Sign in with WebAuthn



You will then be logged in successfully.

Developer Guide

 **Frontend**

Casdoor Frontend Development Guide

 **Generating Swagger Files**

Generating Swagger Files

Frontend

The source code for Casdoor's frontend is located inside the `/web` folder:
<https://github.com/casdoor/casdoor/tree/master/web>

It is a **Create-React-App (CRA)** project, which follows the classic CRA folder structure as outlined below:

File/Directory	Description
public	The root HTML file for React
src	Source code
craco.config.js	The Craco configuration file. You can change the theme color (blue by default) here
crowdin.yml	Crowdin i18n configuration file
package.json	NPM/Yarn dependency file
yarn.lock	Yarn lock file

Inside the `/src` directory, you will find several important files and folders:

File/Directory	Description
account	The "My profile" page for logged-in users
auth	All code related to authentication, such as OAuth,

File/Directory	Description
	SAML, sign up page, sign in page, forget password page, etc.
backend	The SDK for calling the Go backend API. It contains all the <code>fetch()</code> calls
basic	The homepage (dashboard page) for Casdoor, which contains several card widgets
common	Shared UI widgets
locales	i18n translation files in JSON, synced with our Crowdin project: https://crowdin.com/project/casdoor-site
App.js	The entry JS file containing all the routes
Setting.js	Utility functions used by other code
OrganizationListPage.js	The page for the organization list, similar to all other <code>XXXListPage.js</code> files
OrganizationEditPage.js	The page for editing one organization, similar to all other <code>XXXEditPage.js</code> files

Generating Swagger Files

Overview

As we know, the beego framework provides support for generating swagger files to clarify the API via the command line tool called "bee". Casdoor is also built based on beego. However, we found that the swagger files generated by bee failed to categorize the APIs with the "@Tag" label. So, we modified the original bee to implement this function.

How to write the comment

Most rules are exactly identical to the original bee comment formats. The only discrepancy is that the API shall be divided into different groups according to the "@Tag" label. Therefore, developers are obliged to ensure that this tag is correctly added. Here is an example:

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

APIs with the same "@Tag" labels will be put into the same group.

How to generate the swagger file

0. Write comments for the API in the correct format.
1. Fetch this repository: <https://github.com/casbin/bee>.
2. Build the modified bee. For example, in the root directory of casbin/bee, run the following command:

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

3. Copy mybee to the base directory of casdoor.
4. In that directory, run the following command:

```
mybee generate docs
```

5. (Optional) If you want to generate swagger document for specific tags or apis, here are some example commands:

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

Notably: We only accept a comma `,` as the separator when multiple tags/apis provided.

Then you will find that the new swagger files are generated.

Organizations

Overview

Casdoor basic unit — organization

Organization Tree

User groups within an organization

Password Complexity

Supporting different password complexity options.

Account Customization

Customizing users' account items

 **Customizing Themes**

Learn how to customize themes for organizations and applications within an organization

 **Manage Multi-Factor Authentication Items**

Configure Multi-Factor Authentication Items in Organization

Overview

An organization is the basic unit of Casdoor, which manages users and applications. If a user signs in to an organization, then they can access all applications belonging to the organization without signing in again.

In the configuration of [applications](#) and [providers](#), choosing an organization is important, as it determines whether a user can access the application using specific providers.

We can also set up LDAP in Casdoor. For more details, please see the [LDAP](#) documentation.

Casdoor provides multiple password storage algorithms that can be selected on the organization edit page.

Name	Algorithm	Description	Scenario
plain	-	The password will be stored in cleartext. (default)	-
salt	SHA-256	SHA-256 is a patented cryptographic hash function that outputs a value that is 256 bits long.	-
md5-salt	MD5	The MD5 message-digest algorithm is a cryptographically broken but still widely used hash function producing a 128-bit hash value.	Discuz!

Name	Algorithm	Description	Scenario
bcrypt	bcrypt	bcrypt is a password-hashing function and is used to hash and salt passwords securely.	Spring Boot, WordPress
pbkdf2-salt	SHA-256 and PBKDF2	PBKDF2 is a simple cryptographic key derivation function that is resistant to dictionary attacks and rainbow table attacks. It was originally implemented in Casdoor for the Keycloak syncer. Select this option if you are importing users using the Keycloak syncer.	Keycloak

 TIP

In addition to 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.

Organization Tree

Groups are a collection of users within an organization. A user can belong to multiple groups.

Group properties

- `Owner`: The organization that owns the group
- `Name`: Unique group name
- `displayName`
- `CreatedTime`
- `UpdatedTime`
- `Type`: Groups can be classified as either `Physical` or `Virtual`. A user can only belong to one `Physical` group but can be in multiple `Virtual` groups.
- `ParentGroup`: The parent group of a group (The parent group of the top-level groups in the organization is the organization itself)

Managing groups

There are two ways to manage groups:

1. On the groups list page, you can view all the groups within the organization.

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

Groups **Add**

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

9 in total < 1 > 10 / page

2. Click the Groups button on the organization list page.

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

Organizations **Add**

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

4 in total < 1 > 10 / page

This will display the tree structure of the groups within the organization.

The screenshot shows the 'Users' page in Casdoor. The top navigation bar includes Home, Organizations, Groups, Users, Roles, Permissions, Models, Adapters, Applications, Providers, and Chats. The 'Users' page has a search bar for 'Built-in Organization' and buttons for 'Add' and 'Upload (.xlsx)'. A sidebar on the left shows a tree view of the organization structure, including 'Casdoor Project Virtual Team', 'Casbin Project Virtual Team', 'R & D' (with sub-items 'Dev (Frontend)' and 'Dev (Backend)'), 'HR', 'Sales & Marketing' (with sub-items 'Sales' and 'Marketing'), and 'Built-in Team'. The main table lists users with columns for Organization, Application, Name, Created time, and Display name.

Organization	Application	Name	Created time	Display name
built-in	app-built-in	牛头	2023-06-16 16:16:35	牛头
built-in	app-built-in	喝咖啡就大蒜	2023-06-15 10:58:48	喝咖啡就大蒜
built-in	app-built-in	danceshow	2023-06-15 10:53:48	街舞show
built-in	app-built-in	TT珍惜	2023-06-15 10:36:46	TT珍惜
built-in	app-built-in	hashjoin	2023-06-15 01:01:17	hashjoin
built-in	app-built-in	zhangyaphet@gmail.com	2023-06-14 15:50:23	pengwei zhang

Here is a video that shows how to manage groups:

The screenshot shows the 'Groups' page in Casdoor. The top navigation bar includes Home, Organizations, Groups, Users, Roles, Permissions, Models, Adapters, Applications, Providers, Chats, Messages, and a user profile dropdown. The 'Groups' page has an 'Add' button and a table listing groups with columns for Name, Organization, Created time, Updated time, Display name, Type, Parent group, and Action.

Name	Organization	Created time	Updated time	Display name	Type	Parent group	Action
group_smf8bi	built-in	2023-06-13 23:25:31	2023-06-13 23:25:36	总部	Virtual	built-in	Edit Delete
group_zxak7d	built-in	2023-06-11 23:28:45	2023-06-13 23:24:36	New Group - zxak7d	Virtual	New Group - nahuap	Edit Delete
group_1t8lr	built-in	2023-06-09 09:39:44	2023-06-13 23:24:46	美工	Virtual	研发子部门	Edit Delete
group_nahuap	built-in	2023-06-09 09:27:47	2023-06-12 09:36:45	New Group - nahuap	Virtual		Edit Delete
group_38ii7o	built-in	2023-06-07 21:48:49	2023-06-11 09:49:13	研发子部门	Virtual		Edit Delete
group_gnrtp9	built-in	2023-06-07 20:59:10	2023-06-13 23:24:51	实体组3	Physical	built-in	Edit Delete
group_5aca0x	forum	2023-06-06 06:24:33	2023-06-13 23:25:12	实体组2	Physical	forum	Edit Delete
group_3tt9wf	forum	2023-06-06 08:20:14	2023-06-13 23:25:06	顶级2	Virtual	forum	Edit Delete
group_azpdul	forum	2023-06-06 08:19:32	2023-06-09 10:50:25	顶级1	Virtual		Edit Delete
group_r89fha	built-in	2023-06-05 14:41:41	2023-06-12 09:38:28	研发部1	Virtual		Edit Delete

Groups can also be edited in a user's profile.

Title ⓘ : 1122

Homepage ⓘ :

Bio ⓘ :

Tag ⓘ : 222

Karma ⓘ : 333

Signup application ⓘ : app-built-in

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

Roles ⓘ :

Permissions ⓘ :

Password Complexity

Casdoor supports customizing password complexity options for user passwords in each organization.

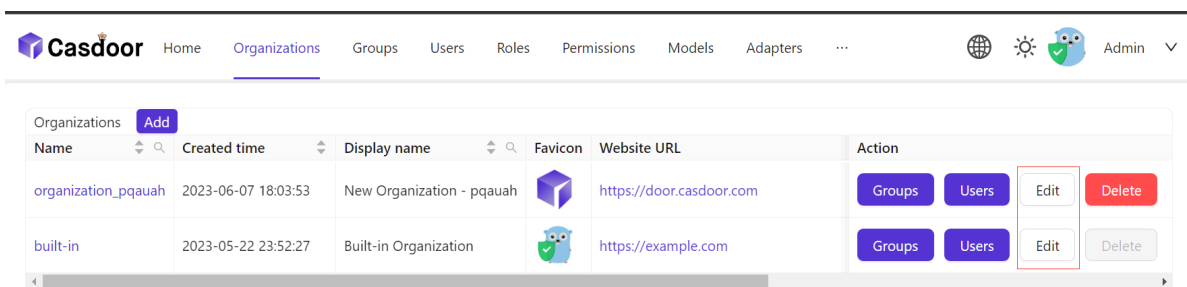
Supported Complexity Options

We currently support five options:




- `AtLeast6`: The password must have at least six characters.
- `AtLeast8`: The password must have at least eight characters.
- `Aa123`: The password must contain at least one uppercase letter, one lowercase letter, and one digit.
- `SpecialChar`: The password must contain at least one special character.
- `NoRepeat`: The password must not contain any repeated characters.

If you want to use multiple options, you can select them on the organization edit page:

1. Click the **Edit** button on the organization list page.




2. Then select the option you need in the `Password complexity options` column.


Casdoor Home Organizations Groups Users Roles Permissions Models Adapters Applications Providers ...    Admin ▾


Edit Organization Save Save & Exit

Name: built-in

Display name: Built-in Organization

Favicon: URL:  https://cdn.casbin.org/img/casbin/favicon.ico

Preview: 

Website URL:  https://example.com

Password type: plain

Password salt:

Password complexity options:

- The password must have at least 8 characters ×
- The password must contain at least one special character ×
- The password must not contain any repeated characters ×
- The password must contain at least one uppercase letter, one lowercase letter and one digit ×

Supported country codes:

- The password must have at least 6 characters ✓
- The password must have at least 8 characters ✓
- The password must contain at least one uppercase letter, one lowercase letter and one digit ✓
- The password must contain at least one special character ✓
- The password must not contain any repeated characters ✓

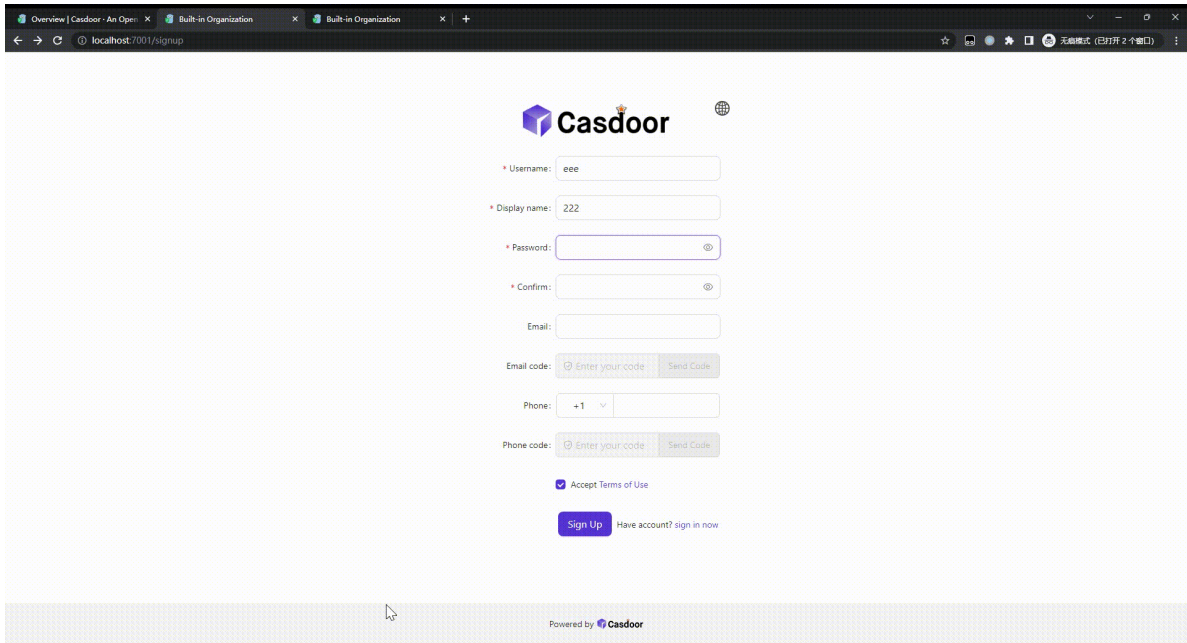
Languages:

- Germany+49 ×
- United Kingdom+44 ×
- India+91 ×

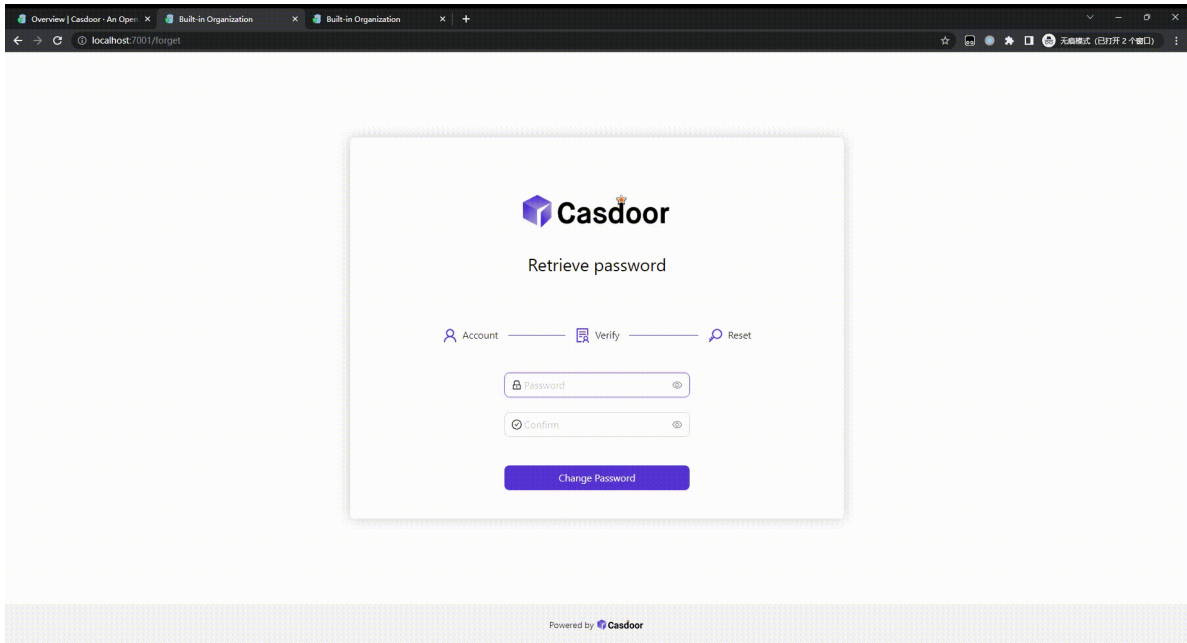
Password Complexity Validation

We support password complexity validation on the following pages:

1. Sign up page.



2. Forget password page.



3. User edit page.

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

localhost:7001/users/built-in/admin

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


Edit User Save Save & Exit

Organization: built-in

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

Name: admin

Display name: Admin

Avatar: Preview: 

Upload a photo...

User type: normal-user

Password: Modify password...

Email: admin@example.com Reset Email...

Phone: +1 12345678910 Reset Phone...

Country/Region: Please select country/region

Location:

Affiliation: Example Inc.

Title:

Cancel Set Password

Password

New Password input password

Re-enter New input password

Account Customization

Introduction

In an organization, you can customize users' account items. This includes whether each item is **visible** and its **view rule** and **modify rule**.

When you customize account items in an organization, this configuration takes effect on the home page of all members of that organization.


How to Customize?

Account items have four attributes:

Column Name	Selectable Value	Description
Name	-	Account item name.
Visible	<input type="checkbox"/> True / <input type="checkbox"/> False	Select whether this account item is visible on the user home page.
ViewRule	Rule Items	Select a rule to use when viewing the account item.
ModifyRule	Rule Items	Select a rule to use when modifying the account item.

To customize account items, follow these steps:

1. Go to the Organization Edit page.
2. You will find the following options:

Account items  [Add](#)

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

3. Casdoor provides simple operations to configure account items:

- Set the item to be visible or invisible.

Account items [Add](#)

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

- Set viewing and modifying rules.

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

There are 3 rules available:

- **Public**: Everyone has permission.
- **Self**: Each user has their own permission.
- **Admin**: The administrator has permission.

Account Table

Below are all the fields in the account item. For descriptions, you can refer to [user](#).

- Organization
- ID
- Name
- Display name
- Avatar
- User type
- Password
- Email
- Phone
- Country/Region
- Location

- Affiliation
- Title
- Homepage
- Bio
- Tag
- Signup application
- 3rd-party logins
- Properties
- Is admin
- Is global admin
- Is forbidden
- Is deleted

Customizing Themes

Casdoor allows you to customize themes to meet the UI diversity requirements of businesses or brands, including primary color and border radius.

Within Casdoor, themes can be customized at the global, organization, and application levels.

1. Global scope: This is the default theme of Casdoor and is applied to any organization that chooses to follow the global theme. Modifications can only be made in the Casdoor source code and cannot be modified in the web UI.
2. Organization scope: The theme for an organization can be customized on the organization edit page. This theme applies to all Casdoor after-login pages for users within the organization, as well as the entry pages (signup, signin, forget password, etc.) of applications that follow the organization theme.
3. Application scope: The theme for an application can be customized on the application edit page. This theme applies to the entry pages (signup, signin, forget password, etc.) of the specific application.

Customizing the Organization Theme

We provide a demo to demonstrate how to configure the theme for an organization:

Roles	<input checked="" type="checkbox"/>	Public	Immutable	^	v	🗑️
Permissions	<input checked="" type="checkbox"/>	Public	Immutable	^	v	🗑️
3rd-party logins	<input checked="" type="checkbox"/>	Self	Admin	^	v	🗑️
Properties	<input checked="" type="checkbox"/>	Admin	Admin	^	v	🗑️
Is admin	<input checked="" type="checkbox"/>	Admin	Admin	^	v	🗑️
Is global admin	<input checked="" type="checkbox"/>	Admin	Admin	^	v	🗑️
Is forbidden	<input checked="" type="checkbox"/>	Admin	Admin	^	v	🗑️
Is deleted	<input checked="" type="checkbox"/>	Admin	Admin	^	v	🗑️
WebAuthn credentials	<input checked="" type="checkbox"/>	Self	Self	^	v	🗑️
Managed accounts	<input checked="" type="checkbox"/>	Self	Self	^	v	🗑️

Theme ⓘ:

LDAPs ⓘ:

Server Name	Server	Base DN	Auto Sync	Last Sync	Action
BuildIn LDAP Server	example.com:389	ou=BuildIn,dc=example,dc=com	Disable		<input type="button" value="Sync"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>

Powered by Casdoor

! INFO


If your account organization is the same as the organization you are editing, the configuration changes will take effect immediately as shown in the video above. However, if they are different, you will need to log in to the organization to see the changes.

Customizing the Application Theme

Applications can customize themes using the same theme editor as the organization. Additionally, you can preview the theme conveniently in the preview panel.

Preview

Copy signup page URL



* Username:

* Password:
Please input your password, at least 6 characters!


Confirm:

Phone: +86

Phone code:

[Have account? sign in now](#)

Copy signin page URL



Auto sign in [Forgot password?](#)

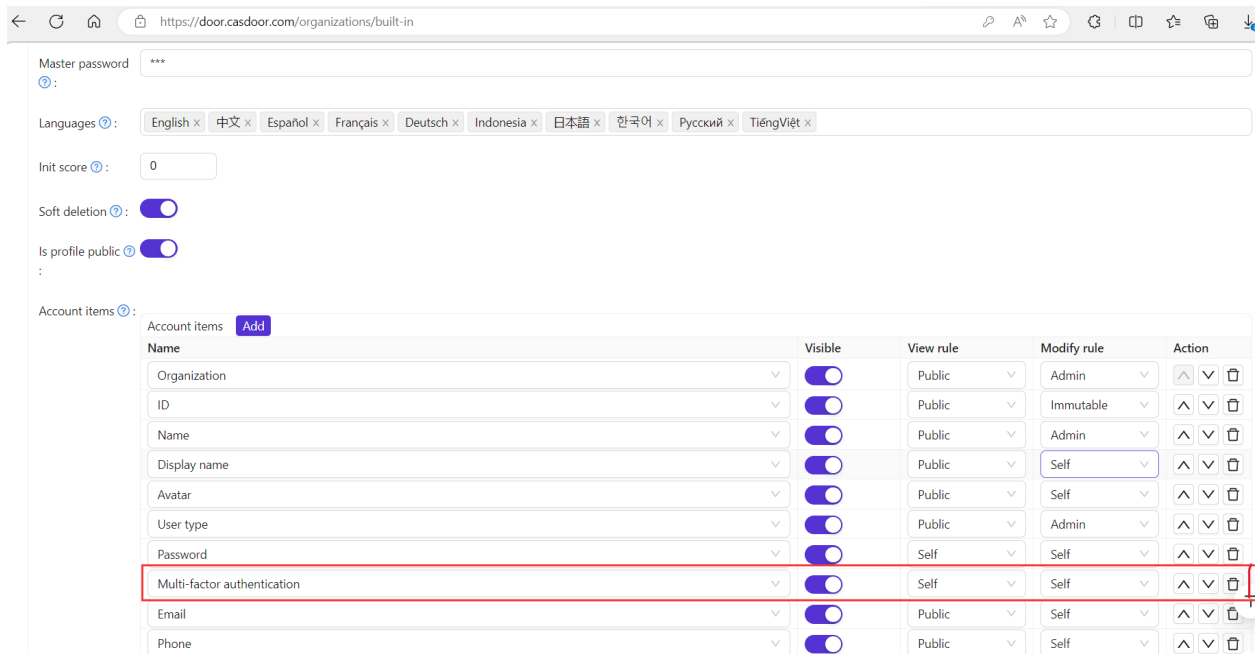
[No account? sign up now](#)

Background URL

Manage Multi-Factor Authentication Items

Add Multi-Factor Authentication Item in Organization

In the organization, admins can add Multi-Factor Authentication items to the account settings. This allows users to configure Multi-Factor Authentication on their own profile pages.



The screenshot shows the Casdoor organization settings page. The URL is <https://door.casdoor.com/organizations/built-in>. The page includes a master password field, language selection, and several toggle switches for 'Init score', 'Soft deletion', and 'Is profile public'. The 'Account items' section is expanded, showing a table of items with columns for Name, Visible, View rule, Modify rule, and Action. The 'Multi-factor authentication' item is highlighted with a red box.

Name	Visible	View rule	Modify rule	Action
Organization	<input checked="" type="checkbox"/>	Public	Admin	↑ ↓ 🗑️
ID	<input checked="" type="checkbox"/>	Public	Immutable	↑ ↓ 🗑️
Name	<input checked="" type="checkbox"/>	Public	Admin	↑ ↓ 🗑️
Display name	<input checked="" type="checkbox"/>	Public	Self	↑ ↓ 🗑️
Avatar	<input checked="" type="checkbox"/>	Public	Self	↑ ↓ 🗑️
User type	<input checked="" type="checkbox"/>	Public	Admin	↑ ↓ 🗑️
Password	<input checked="" type="checkbox"/>	Self	Self	↑ ↓ 🗑️
Multi-factor authentication	<input checked="" type="checkbox"/>	Self	Self	↑ ↓ 🗑️
Email	<input checked="" type="checkbox"/>	Public	Self	↑ ↓ 🗑️
Phone	<input checked="" type="checkbox"/>	Public	Self	↑ ↓ 🗑️

Manage Multi-Factor Authentication Items

You can manage Multi-Factor Authentication to determine which methods are available to users.

There are two rules for managing Multi-Factor Authentication items:

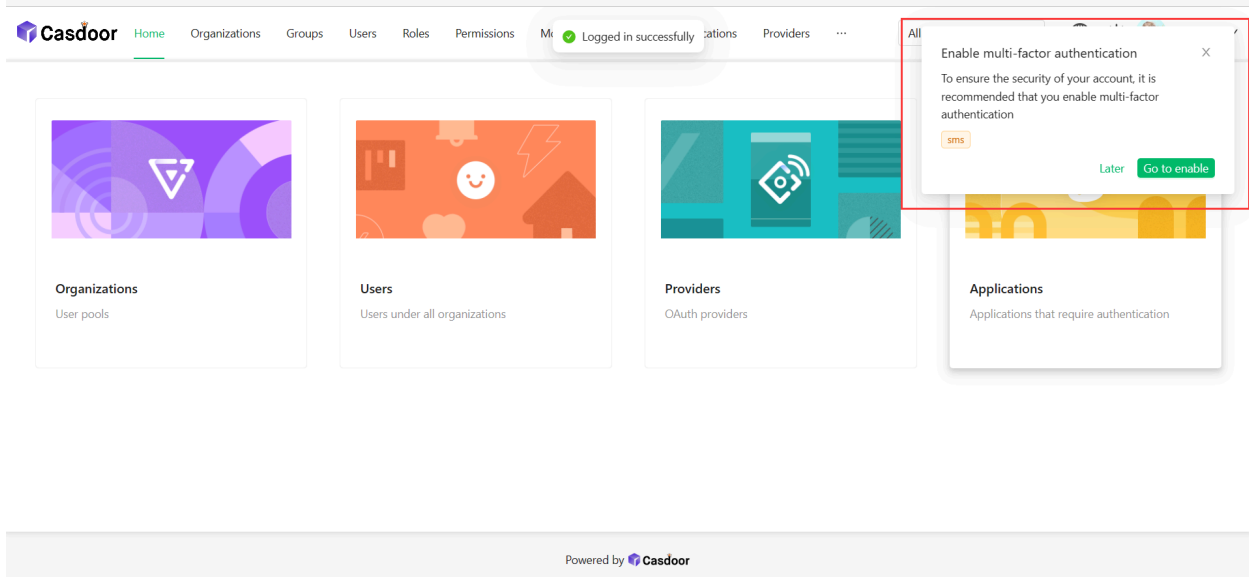
- Optional: Users can choose whether to enable this type of Multi-Factor Authentication.
- Prompt: If the user does not enable this Multi-Factor Authentication mode, they will be prompted to enable it after logging in to Casdoor.
- Required: Users must enable this Multi-Factor Authentication method.



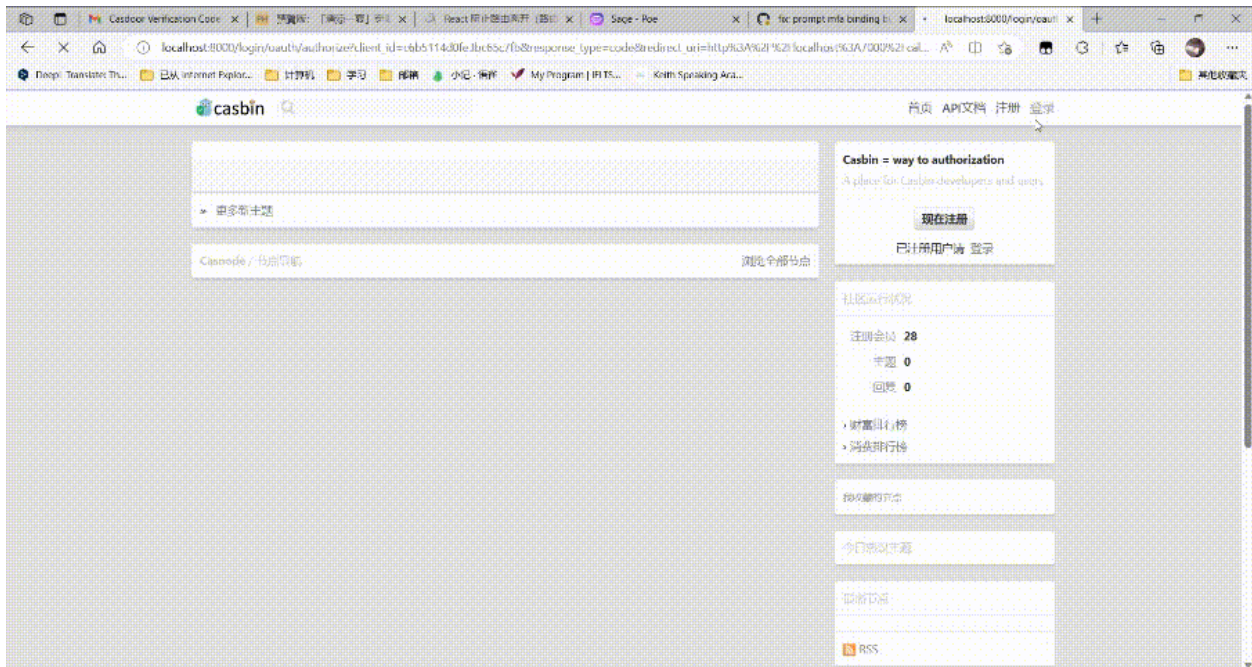
MFA items [?](#) : MFA items

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

The image below shows the notification that prompts users to enable Multi-Factor Authentication.



This video demonstrates that when the Multi-Factor Authentication method is set to required, users need to enable Multi-Factor Authentication before they can complete the login process.



Applications

 **Overview**

Casdoor Application Overview

 **Terminology Reference**

Terminology reference

 **Application Config**

Configure your application's authentication

 **Providers**

Configure different providers

Signin Methods

Configure the login method and the display order of the login methods

Signup Items Table

Configure the signup items table to create a custom registration page

Login UI Customization

Customize the login page UI for your application

Specify Login Organization

Specify the login organization on the login page

Tags

Configure your application tags

 **Application Invitation Code**

Restrict application sign up with invitation codes

Overview

Every application in Casdoor is called an "application". They are not related and do not affect each other, which means you can deploy or stop any application separately, as long as you like.

If you want to use Casdoor to provide login service for your web apps, you can add them as Casdoor applications.

Users can access all applications in their organizations without logging in twice.

The application configuration is very flexible and simple. You can set whether to allow password login or third-party login, configure the third-party applications you want users to log in to, and you can even customize the signup items of the application, etc.

In this chapter, you will learn how to start your own application from scratch.

Let's explore together!

Terminology Reference

- **Name**: The name of the created app.
- **CreatedTime**: The time when the application is created.
- **DisplayName**: The name which the application displays to the public.
- **Logo**: Application logos will be displayed on the login and sign up pages.
- **HomepageUrl**: The URL of the application's homepage.
- **Description**: Describes the application.
- **Tags**: Only users with tags listed in the application tags can login.
- **Organization**: The organization that the app belongs to.
- **EnableSignUp**: If users can sign up. If not, accounts of the application.
- **SignInMethods**: Configuration of Sign-in Methods
- **SignupItems**: Fields that need to be filled in when users register.
- **Providers**: Provide all kinds of services for the applications (such as OAuth, Email, SMS service).
- **ClientId**: OAuth client ID.
- **ClientSecret**: OAuth client secret.
- **RedirectUris**: Casdoor will navigate to one of the URIs if the user logged in successfully.
- **TokenFormat**: The format of the generated token. It can be in the following formats: **JWT** (containing all **User** fields), **JWT-Empty** (containing all non-empty values) or **JWT-Custom** customizing **User** fields inside access token.
- **ExpireInHours**: Login will expire after hours.
- **SignInUrl**:
- **SignupUrl**: If you provide a sign-up service independently outside of Casdoor, please fill in the URL here.
- **ForgetUrl**: Same as **SignupUrl**.

- `AffiliationUrl`:

Application Config

After you deploy Casdoor on your server and set up your organization, you can now deploy your applications!

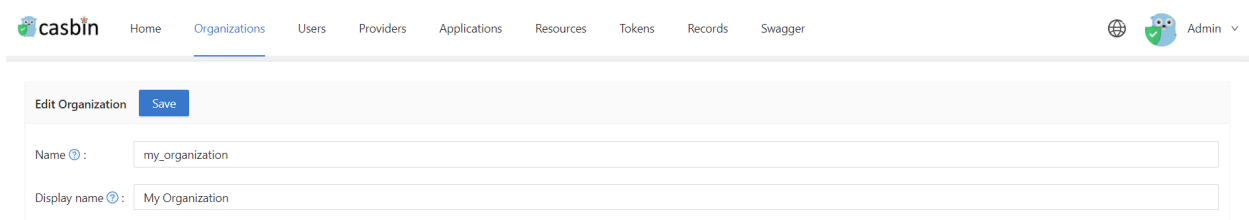
Let's see how to configure your application's authentication using Casdoor!

NOTE

For example, I want to set up my Forum using Casnode.

I create my application and fill in some necessary configurations.

Select the organization I created so that users in this organization can use this application.



The screenshot shows the Casbin web interface. The top navigation bar includes 'casbin', 'Home', 'Organizations', 'Users', 'Providers', 'Applications', 'Resources', 'Tokens', 'Records', and 'Swagger'. On the right, there is a globe icon and an 'Admin' dropdown menu. The main content area is titled 'Edit Organization' and features a 'Save' button. Below the title, there are two input fields: 'Name' with the value 'my_organization' and 'Display name' with the value 'My Organization'.


Since this organization is named `my_organization`, I choose it from the drop-down menu.

Edit Application Save

Name [?]:

Display name [?]:

Logo [?]:
URL:

Preview: 

Home [?]:

Description [?]:

Organization [?]:

Client ID [?]:

Next, I want my users to be able to use Casdoor for authentication when they sign up. So, I fill in the redirect URL here as <https://your-site-url.com/callback>.

⚠ CAUTION

Please note that the **callback URL** in the provider application should be Casdoor's callback URL, and the **Redirect URL** in Casdoor should be your website's callback URL.

Further Understanding

To make the authentication process work, the detailed steps are as follows:

1. Users send a request to Casdoor.
2. Casdoor uses the `Client ID` and `Client Secret` to authenticate with GitHub, Google, or other providers.
3. If the authentication is successful, GitHub calls back to Casdoor to notify Casdoor about the successful authentication. Therefore, the GitHub authorization callback URL should be your Casdoor's callback URL, which is <http://your-casdoor-url.com/callback>.
4. Casdoor then informs the application about the authentication success. This means that the Casdoor callback URL should be your application's callback URL, which is <http://your-site-url.com/callback>.

You need to enable JavaScript to run this app.

 TIP

If you want to do more personalized configuration of the application's sign-in methods, such as disabling a certain sign-in method or turning off a certain sign-in method, you can refer to the [Signin Methods](#)

Providers

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

Name	canSignUp	canSignIn	canUnlink	prompted	Action
provider_casbin_email					⬆️ ⬇️ 🗑️
provider_casbin_sms					⬆️ ⬇️ 🗑️
provider_storage_aliyun_oss					⬆️ ⬇️ 🗑️
provider_casdoor_github_localhost	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	⬆️ ⬇️ 🗑️
provider_casdoor_github	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	⬆️ ⬇️ 🗑️
provider_casdoor_google	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	⬆️ ⬇️ 🗑️
provider_casdoor_qq	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	⬆️ ⬇️ 🗑️
provider_casdoor_wechat	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	⬆️ ⬇️ 🗑️
provider_casdoor_facebook	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	⬆️ ⬇️ 🗑️
provider_casdoor_gitee	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	⬆️ ⬇️ 🗑️
provider_casdoor_gitlab	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	⬆️ ⬇️ 🗑️

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

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

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

Providers ⓘ :

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

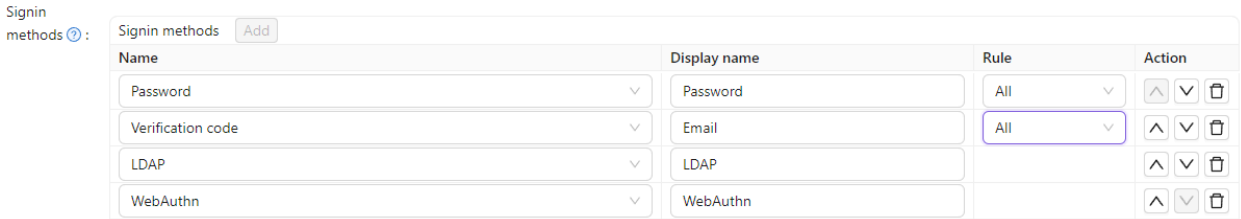
Preview ⓘ :

Copy signup page URL

Copy signin page URL

Signin Methods

On the Application Configuration page, we can configure the sign-in item table. We can add and remove sign-in items from the table.



For a detailed explanation of each sign-in item, please refer to the table below. Currently, only Password, verification code, WebAuthn and LDAP login methods are available.

Column Name	Selectable Value	Description
Name	-	The name of the sign-in method.
DisplayName	-	The name which the sign-in method displays to the public.
Rule	Rule Items	Select a rule to customize this sign-in method. Detailed rules are described in the table below.
Action	-	Users can perform actions such as moving this sign-in method up, moving it down, or deleting it.


At present, configuration rules are only supported for the `Password` and `Verification code` sign-in methods.







Sign-in Method Name	Selectable Rules	Description
Password	<code>All(default)</code> / <code>Non-LDAP</code>	Select the sign-in methods available to the user. Choosing <code>All</code> , then LDAP users can also sign-in. Choosing <code>Non-LDAP</code> , then LDAP users are prohibited from sign-in.
Verification code	<code>All(default)</code> / <code>Email only</code> / <code>Phone only</code>	Select the sign-in methods available to the user. Choosing <code>All</code> , then both email and phone numbers can be verified for sign-in. Choosing <code>Email only</code> , then only email login is allowed. Choosing <code>Phone only</code> , then only the phone number is allowed to authenticate the login.

i NOTE

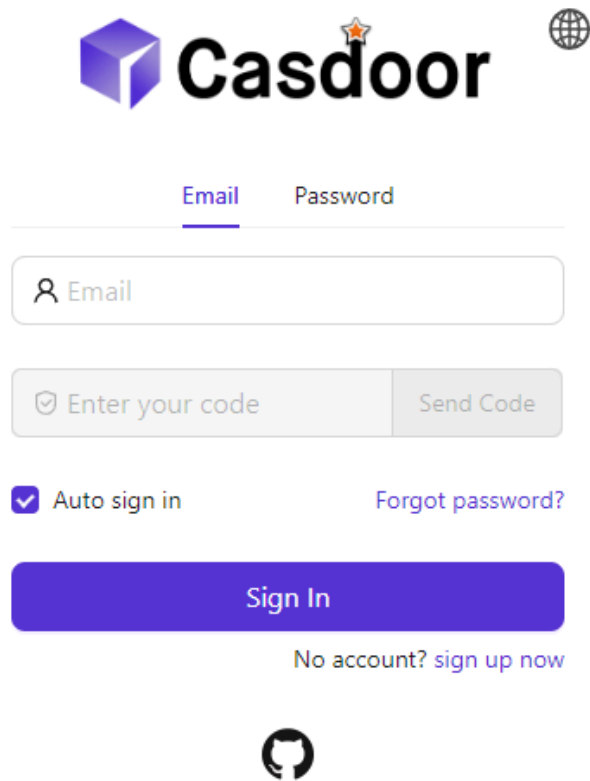
For example, we want users to prioritize logging in with their email, and then consider logging in with a password if they can't use their email.


First, we configure two login options, `Verification Code` and `Password`, and `Verification Code` is the first login option. Then we change the `verification code` rule to `Email only`, so that the user can only receive the login verification code by email.

Signin methods :

Name	Display name	Rule	Action
Verification code	Email	Email only	  
Password	Password		  

To make it easier for users to understand, we can change the display name of the `Verification code` login method so that users can easily understand that it is an email login.



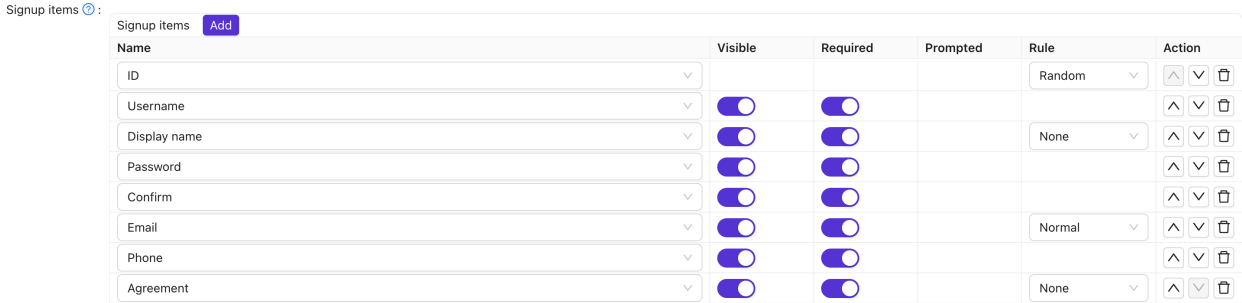
 TIP

All login options, except for LDAP, are enabled by default. And it is required that at least one sign-in method be added.

Here is a video of how the sign-in method works:

Signup Items Table

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



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

Column Name	Selectable Value	Description
Name	-	The name of the signup item.
Visible	<input type="checkbox"/> True / <input type="checkbox"/> False	Select whether this signup item is visible on the registration page.
Required	<input type="checkbox"/> True / <input type="checkbox"/> False	Select whether this signup item is mandatory.
Prompted	<input type="checkbox"/> True / <input type="checkbox"/> False	Select whether to prompt the user when they forget to fill in this signup item.

Column Name	Selectable Value	Description
Rule	Rule Items	Select a rule to customize this signup item. Detailed rules are described in the table below.
Action	-	Users can perform actions such as moving this signup item up, moving it down, or deleting it.

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

Item Name	Selectable Rules	Description
ID	Random / Incremental	Select whether the user ID should be randomly generated or incremented.
Display name	None / Real name / First, last	Choose how the display name should be presented. Choosing None will display Display name. Choosing Real name will display the user's actual name. Choosing First, last will display the first and last name separately.
Email	Normal / No verification	Select whether to verify the email address with a verification code. Choosing Normal will require email verification. Choosing No verification will allow signup without email

Item Name	Selectable Rules	Description
		verification.
Agreement	<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px; display: inline-block;">None /</div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px; display: inline-block;">Signin /</div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px; display: inline-block;">Signin</div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px; display: inline-block;">(Default</div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px; display: inline-block;">True)</div>	<p>Select whether the user needs to confirm the terms of use when logging in. Choosing None will not display any terms of use, allowing users to log in directly. Choosing Signin will require users to confirm the terms before logging in. Choosing Signin (Default True) will set the terms as confirmed by default, allowing users to log in directly.</p>

i NOTE

For example, let's say I want to set up my registration page to include an email field, but without requiring email verification.

Firstly, I added some signup items necessary for registration, such as ID, Username, Password, and Email.

Signup items Add

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

Then, I selected the email row's rule item as **No verification**. As a result, the generated preview registration page will have the desired effect.



* Username:

* Password:



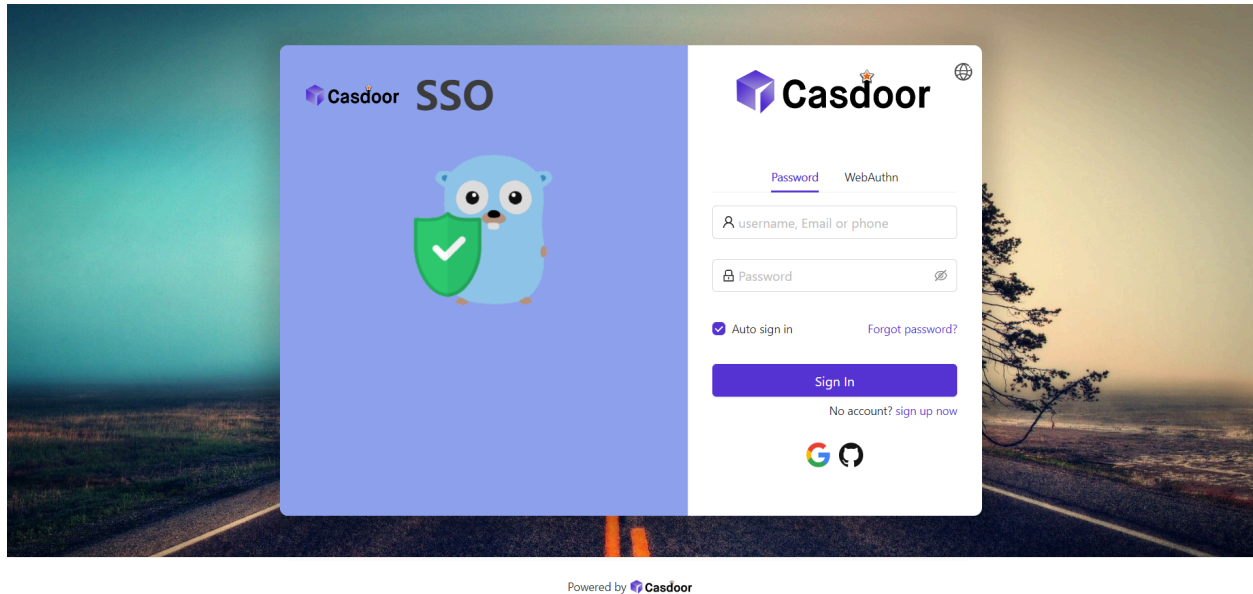
* Email:

Sign Up

Have account? [sign in](#)
now

Login UI Customization

You have created the application. Now, let me show you how to customize the login page UI of your application. In this guide, we will create a customized login page for your application.



Let's get started!

Part 1: Add a background image

First, let's add a background image. The default background is white, which looks very simple.



Password WebAuthn

username, Email or phone

Password

Auto sign in

[Forgot password?](#)

Sign In

[No account? sign up now](#)



Powered by Casdoor

To add a background image, fill in the **Background URL** with the URL of the image you like. The preview area will display the image if the URL is valid.

Background URL

[?](#) :

URL [?](#) :

[🔗](#) |

Preview:

Form CSS [?](#) :

Form position [?](#) :

Left

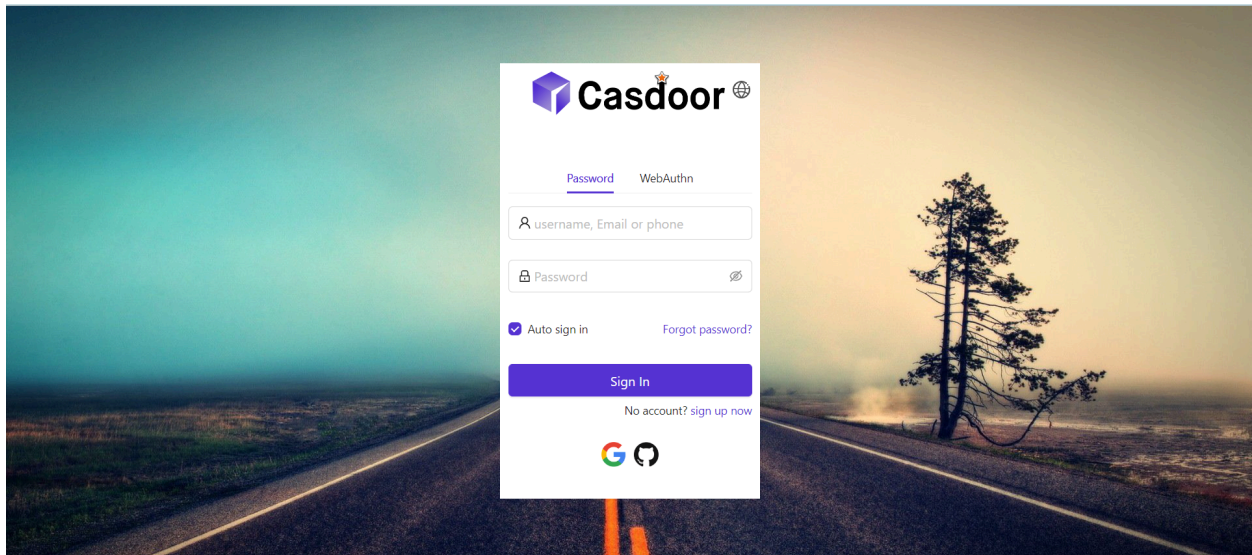
Center

Right

Enable side panel

Part 2: Customize the login panel

Here's where you were at the end of the first part:




Powered by  Casdoor

To make the panel look nice, you need to add some CSS code to it. Copy the code below and paste it into the `Form CSS` field.

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

Background URL [?](#) :

URL [?](#) : <https://static.runoob.com/images/demo/demo2.jpg>

Preview: 

Form CSS [?](#) :

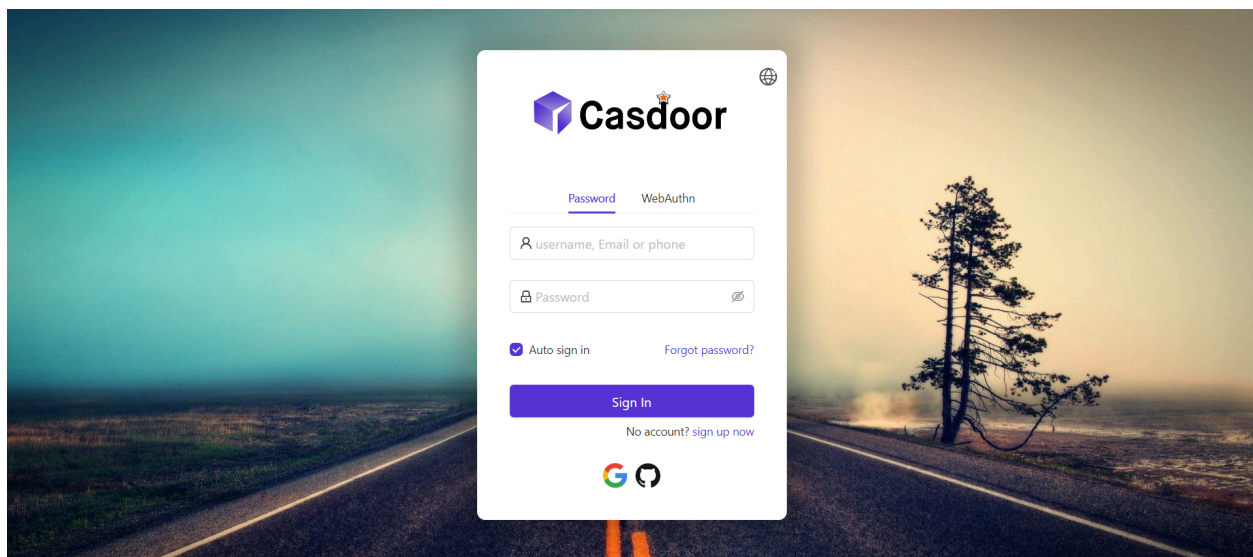
Form position [?](#) :



TIP

When editing the **Form CSS**, if the value is empty, the editor will show the default value. However, you still need to copy the content and paste it into the field.

After filling the **Form CSS**, don't forget to save the configuration at the bottom. Now, let's see the effect.




Part 3: Select the panel position

Now, the login page looks much prettier than before. We also provide three buttons for you to decide the position of the panel.

Background URL




URL  :


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

Preview:



Form CSS  :

```
<style>.login-panel{ padding: 40px 30px 0 30px; border-radius: 10px; b
```

Form position  :

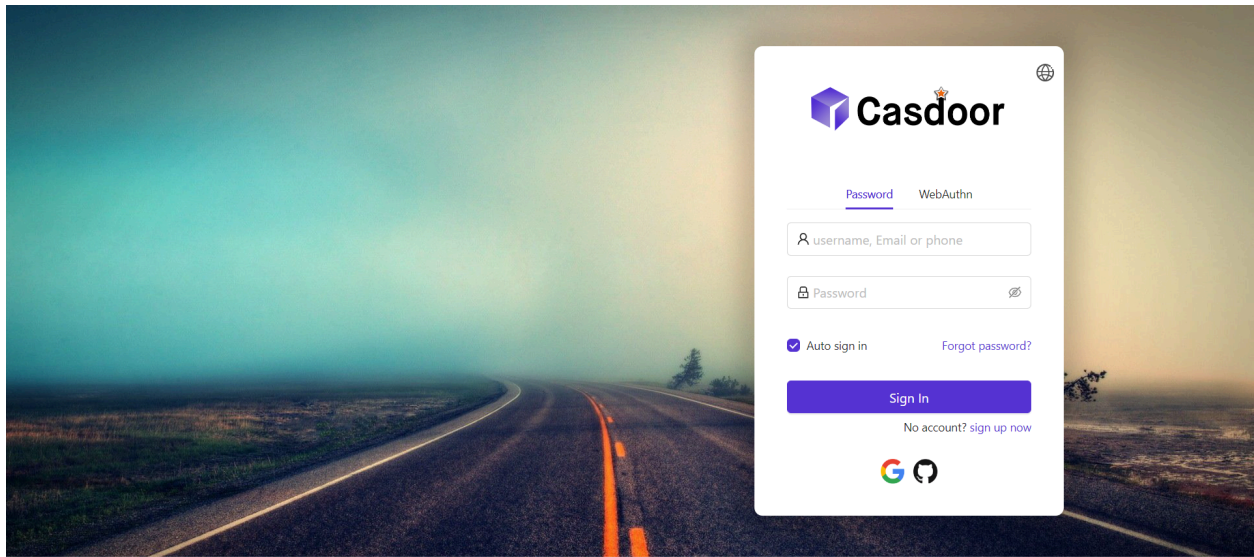
Left

Center

Right

Enable side panel

For example, let's select the Right button:




Powered by  Casdoor


Part 4: Enable the side panel

Next, let's see how to enable a side panel and customize its style.

First, select the button. In the Enable Side Panel mode, the panel will be centered.

Form position  :

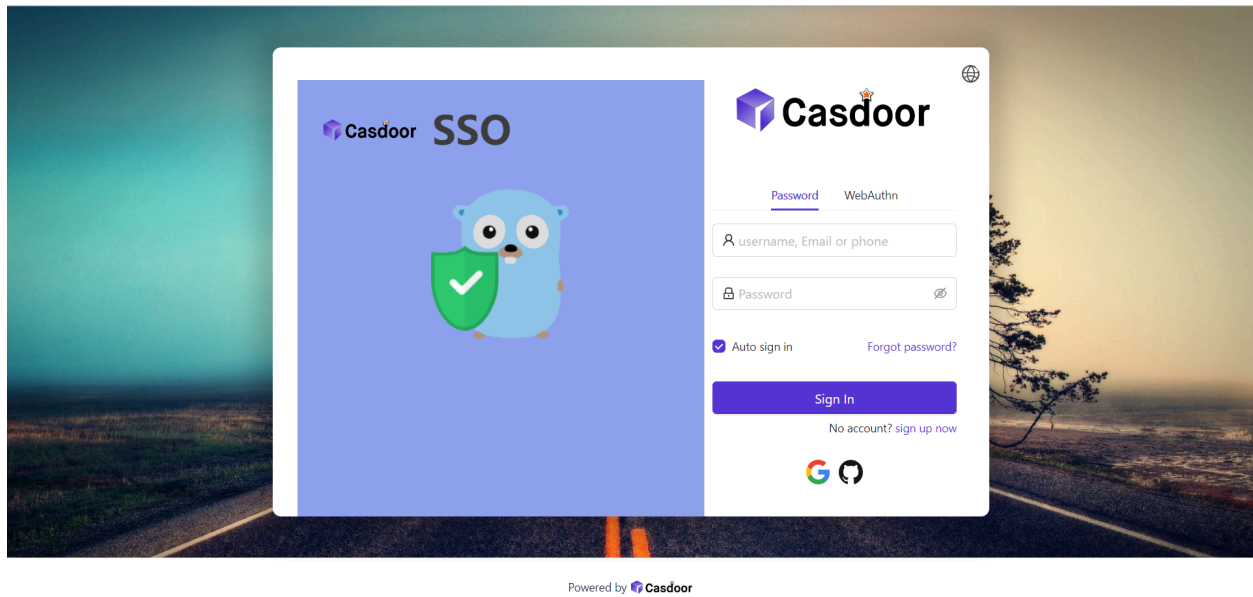
Left	Center	Right	Enable side panel
------	--------	-------	-------------------

Side panel HTML  :

Then, edit the `Side panel HTML`, which determines the content that will be shown in the side panel. We provide a default template, so you can simply copy and paste it.

```
<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;
    font-size: 2.4rem;
    line-height: 1.3;
    margin-left: 16px;
    color: #404040;
  }
  .img{
    max-width: none;
    margin: 41px 0 13px;
  }
</style>
<div class="left-model">
  <span class="side-logo"> 
    <span>SS0</span>
  </span>
  <div class="img">
    
  </div>
</div>
```


Let's see the effect. The side panel with a logo and image is shown, but the result is not satisfactory.



To improve the look, you need to modify and add some CSS in the **Form CSS**.

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

Preview:

Form CSS: `<style> .login-panel{ padding: 40px 30px 0 30px; border-radius: 10px; background-color: #ffffff; box-shadow: 0 0 30px 20px rg`

Form position: Left Center Right Enable side panel

Side panel HTML: `<style> .left-model{ text-align: center; padding: 30px; background-color: #8ca0ed; position: absolut`

Signup items: Signup items Add

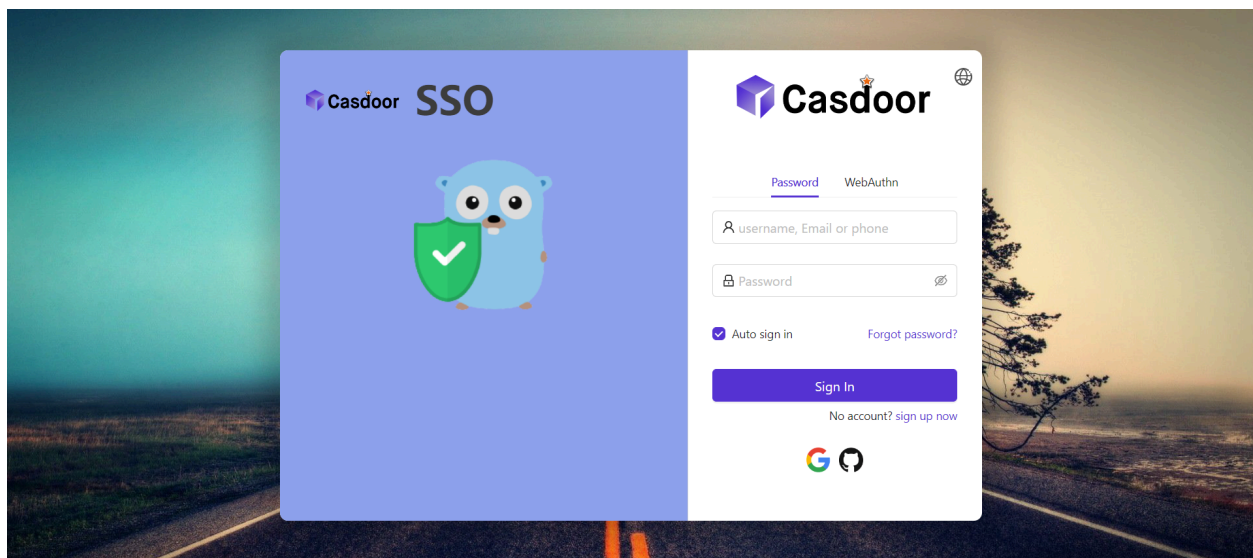
The final code is as follows.

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

! INFO

`.login-panel` and `.login-form` are the class names of div elements. They correspond to different areas of the page. If you want to customize the login page further, you can write CSS code here, targeting these class names.

Finally, we have a beautiful login page!



Review

To summarize, we have added a background image, customized the login panel style, and enabled the side panel.

Here are some additional resources about application customization in Casdoor:

- [Customize Theme](#): Customize the theme, including the primary color and border radius.
- [Signup Items Table](#)
- [Application Config](#)

Thank you for reading!

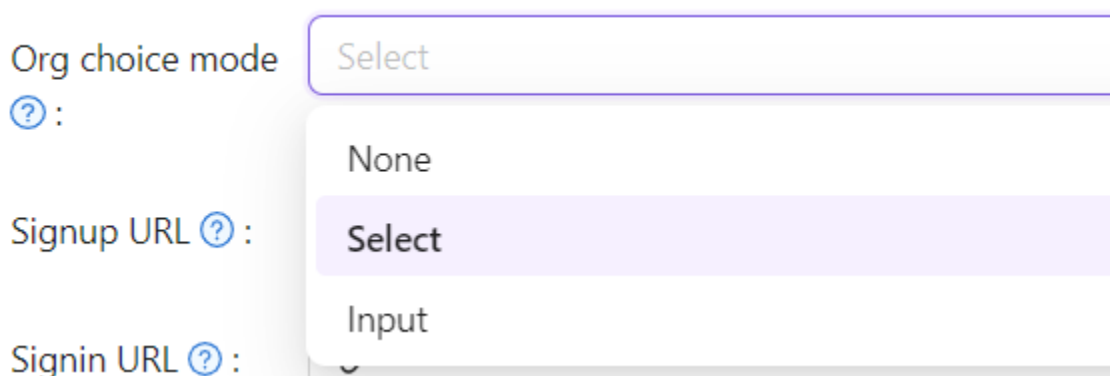
Specify Login Organization

Here, we will show you how to enable the option to specify the login organization for the application.

For example, the endpoint `/login` is the default sign-in page for accounts belonging to the **built-in** organization. However, you can enable the option to specify the login organization on the **app-built-in** application that belongs to the **built-in** organization. This allows the user to select an organization when logging in. After the user selects the organization, they will be redirected to `/login/<organization>`.

Configuration

On the application edit page, you can find the `Org select mode` configuration option. You can select the mode from the dropdown list.



The image shows a configuration interface with three labels on the left: 'Org choice mode', 'Signup URL', and 'Signin URL'. Each label has a blue question mark icon to its right. A dropdown menu is open, showing three options: 'None', 'Select', and 'Input'. The 'Select' option is highlighted with a light purple background. The 'Org choice mode' label is partially obscured by the dropdown menu.

- None: The organization select page will not be shown.
- Input: The user can input the organization name in the input box.
- Select: The user can select the organization from the dropdown list.



Please type an organization to sign in

Confirm



Please select an organization to sign
in

- built-in
- forum
- test
- Star

 INFO

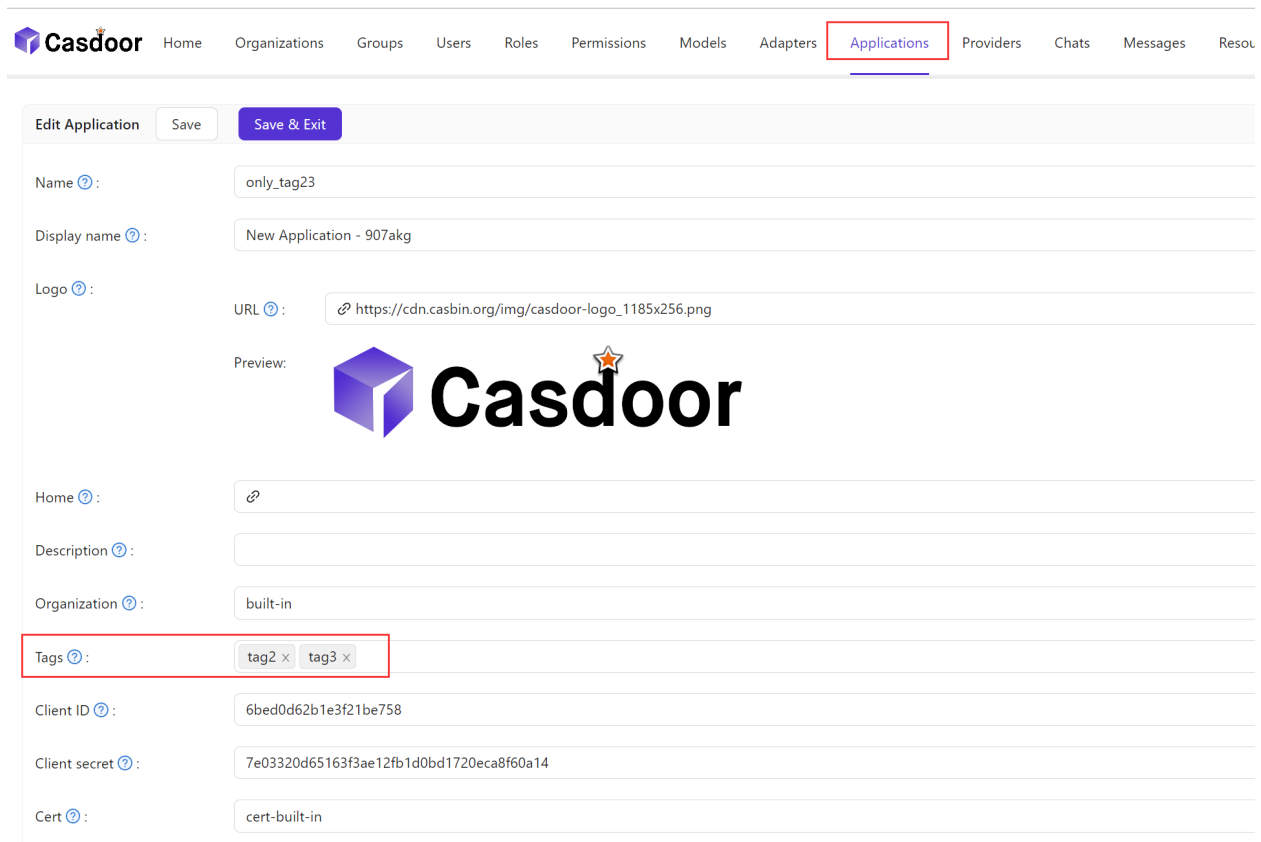
The organization select page will only be shown when the route is `/login` or `<organization>/login`. This means that the application should be set as the **default application** in the organization or the app-built-in.

Tags

The application tags are used to restrict user access to the application.

Specifically, only users with tags listed in the application tags are allowed to log in. For example, the application `dev_app` has tags `dev`, `prd`. Only users with the tag `dev` or `prd` can log in to `dev_app`. Please note that admin and global admin users are not affected by application tags.

On the application edit page, you can find the `Tags` configuration section where you can add tags.



The screenshot shows the Casdoor application edit page. The navigation bar at the top includes Home, Organizations, Groups, Users, Roles, Permissions, Models, Adapters, Applications (highlighted), Providers, Chats, Messages, and Resources. The main content area is titled "Edit Application" and contains several form fields:

- Name: only_tag23
- Display name: New Application - 907akg
- Logo: URL: https://cdn.casbin.org/img/casdoor-logo_1185x256.png
- Preview: Casdoor logo with a star icon
- Home: /
- Description: (empty)
- Organization: built-in
- Tags: tag2 x tag3 x (highlighted with a red box)
- Client ID: 6bed0d62b1e3f21be758
- Client secret: 7e03320d65163f3ae12fb1d0bd1720eca8f60a14
- Cert: cert-built-in

Here is a video demonstrating how application tags work:

Application Invitation Code

Introduction

If you want to restrict application sign up, you can use invitation codes. An invitation code is a string that can be used to sign up for the application. It is generated by the administrator and can be used multiple times. An application can have multiple invitation codes.

Configuration

1. First, add the "Invitation code" signup item to the signup item table.
2. Then, add the invitation code on the application configuration page.

Signup items ⓘ

Signup items [Add](#)

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

Invitation code ⓘ

[Add](#)

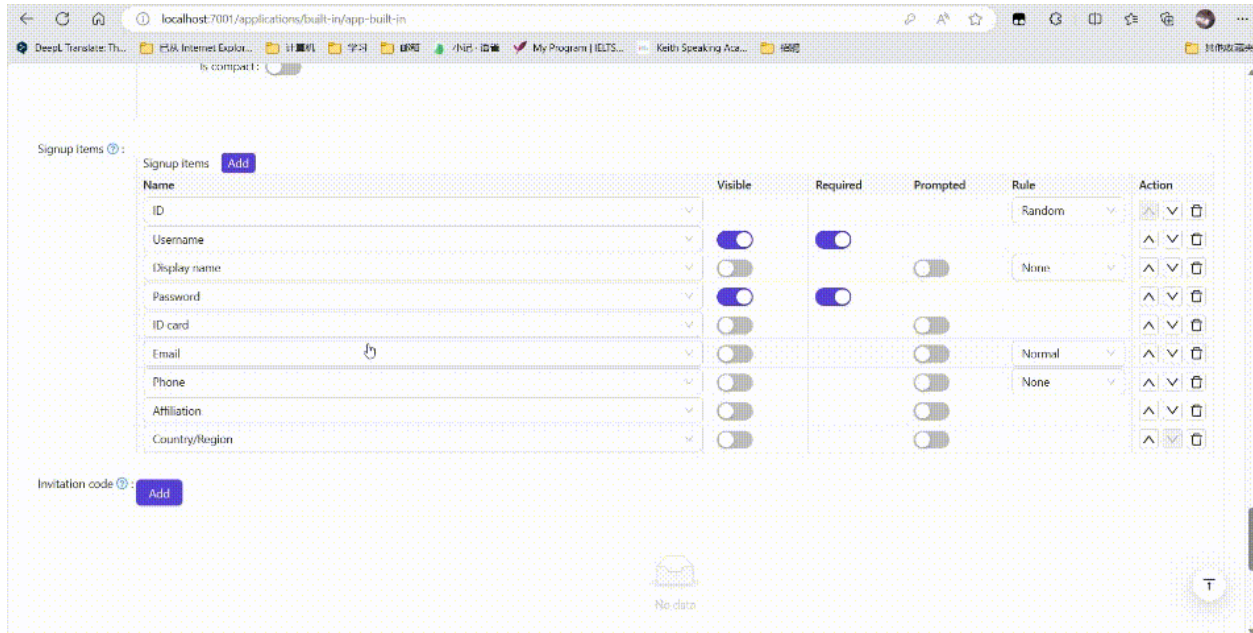
[Copy](#) [Delete](#)

💡 TIP

Once the application has invitation codes, users can only sign up for the application with a valid invitation code. Regardless of whether the "Invitation code" signup item is visible or not, users must provide the

invitation code during sign up. So, if you want to use invitation codes, you need to add the "Invitation code" signup item to the signup item table.

Here is a demo video that shows how to configure and use the invitation code:



Permissions

Overview

Using Casbin to manage user access rights in organizations

Permission Configuration

Using exposed Casbin APIs to manage users' access rights in an organization

Exposed Casbin APIs

Using exposed Casbin APIs to manage user access rights in organizations

Adapter

Configure adapter and perform basic CRUD operations on policy

Overview

Introduction

All users associated with a single Casdoor organization share access to the organization's applications. However, there may be instances where you want to restrict user access to certain applications or specific resources within an application. In such cases, you can utilize the `Permission` feature provided by [Casbin](#).

Before delving deeper into the topic, it is important to have a basic understanding of how Casbin works and its related concepts, such as Models, Policies, and Adapters. In a nutshell, a Model defines the structure of your permission policies and the criteria for matching requests against these policies and their outcomes. A Policy, on the other hand, describes the specific permission rules. Once Casbin has the necessary Model and Policy information, it can enforce permission control on incoming requests. Acting as an abstraction layer, an Adapter shields Casbin's executor from the source of the Policy, allowing the storage of Policies in various locations like files or databases.

Returning to the subject of permission configuration in Casdoor, you can add a Model for your organization in the `Model` configuration item within the Casdoor Web UI, and a Policy for your organization in the `Permission` configuration item. The [Casbin Online Editor](#) can provide you with Model and Policy files tailored to your specific usage scenarios. You can effortlessly import the Model file into Casdoor through its Web UI for use by the built-in Casbin. However, for the Policy (i.e., the `Permission` configuration item in the Casdoor Web UI), further instructions are necessary, which will be discussed later.

Just as your application needs to enforce permission control through Casdoor's built-in Casbin, Casdoor itself utilizes its own Model and Policy to regulate access

permissions for the API interfaces through Casbin. Though Casdoor can call Casbin from internal code, external applications cannot. As a solution, Casdoor exposes an API for external applications to call the built-in Casbin. We will provide definitions of these API interfaces and instructions on how to use them shortly.

Towards the end of this chapter, we will showcase a practical example to demonstrate how Casdoor works in collaboration with external applications for permission control.

Let's get started!

Permission Configuration

Let's explain each item in the Permission Configuration page.

- **Organization**: The name of the organization to which the policy belongs. An organization can have multiple permission policy files.
- **Name**: The globally unique name of the permission policy in the organization. It is used to identify the policy file.
- **Display name**: Not important.
- **Model**: The name of the model file that describes the structure and matching patterns of the permission policy.
- **Adapter**: **Attention!** In the current version, this field describes the name of the database table that stores the permission policy, rather than the name of the adapter configured in the Adapter menu item in the Casdoor Web UI. Casdoor uses its own database to store configured permission policies. If this field is empty, the permission policy will be stored in the `permission_rule` table. Otherwise, it will be stored in the specified database table. If the specified table name does not exist in the database used by Casdoor, it will be created automatically. We strongly recommend **specifying different adapters for different models**, as keeping all policies in the same table may cause conflicts.
- **Sub users**: Which users will the permission policy be applied to.
- **Sub roles**: If the RBAC model is used, which roles will be applied to the permission policy. This will add permission policies such as `g user role` for every user in this role.
- **Sub domains**: Which domains will the permission policy be applied to.
- **Resource type**: In the current version, Casdoor does not use this field for external applications that want to authenticate. You can ignore it for now.
- **Resources**: This field describes the resources for which you wish to enforce

permission control. Note, however, that the resources here are not those configured in the Resources menu item of the Casdoor Web UI. You can add any string you want here, such as a URL or a filename.

- **Actions**: This field describes the actions to operate on resources. Similar to resources, it can be any string you want, such as an HTTP method or other natural language. But please note that Casdoor will convert all these strings to lowercase before storing. Additionally, Casdoor will apply all actions to each resource. You cannot specify that an action only takes effect on certain resources.
- **Effect**: This option takes effect for Casdoor itself to control application access. If you want an external application to enforce permission controls using the interface Casdoor exposes, it won't do anything. You should describe the effect of pattern matching in the Model file.

As you can see, this configuration page is almost tailor-made for the (sub, obj, act) model.

Exposed Casbin APIs

Introduction

Let's assume that your application's front-end has obtained the `access_token` of the logged-in user and now wants to authenticate the user for some access. You cannot simply place the `access_token` into the HTTP request header to use these APIs because Casdoor uses the `Authorization` field to check the access permission. Like any other APIs provided by Casdoor, the `Authorization` field consists of the application client id and secret, using the [Basic HTTP Authentication Scheme](#). It looks like `Authorization: Basic <Your_Application_ClientId> <Your_Application_ClientSecret>`. For this reason, Casbin APIs should be called by the application backend server. Here are the steps on how to do it.

Take the [app-vue-python-example](#) application in the demo site for example, the authorization header should be: `Authorization: Basic 294b09fbc17f95daf2fed8982f7046ccba1bbd7851d5c1ece4e52bf039d`.

1. The front-end passes the `access_token` to the backend server through the HTTP request header.
2. The backend server retrieves the user id from the `access_token`.

As a note in advance, these interfaces are also designed (for now) for the `(sub, obj, act)` model. The `permissionId` in the URL parameters is the identity of the applied permission policy, which consists of the organization name and the permission policy name (i.e., `organization name/permission name`). The body is the request format defined by the Casbin model of the permission, usually representing `sub`, `obj` and `act` respectively.

In addition to the API interface for requesting enforcement of permission control, Casdoor also provides other interfaces that help external applications obtain permission policy information, which is also listed here.

Enforce

Request:

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

Response:

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

BatchEnforce

Request:

```
curl --location --request POST 'http://localhost:8000/api/batch-enforce?permissionId=example-org/example-permission' \  
--header 'Content-Type: application/json' \  
--header 'Authorization: Basic <Your_Application_ClientId>  
<Your_Application_ClientSecret>' \  
--data-raw '[["example-org/example-user", "example-resource",  
"example-action"], ["example-org/example-user2", "example-  
resource", "example-action"], ["example-org/example-user3",  
"example-resource", "example-action"]]'
```

Response:

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

GetAllObjects

Request:

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

Response:

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

GetAllActions

Request:

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

Response:

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

GetAllRoles

Request:

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

Response:

```
[  
  "role_kcx661"  
]
```

Adapter

Casdoor supports using the UI to connect the adapter and manage policy rules. In Casbin, the storage of policy rules is implemented as an adapter, which acts as 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. Currently supports database adapter.
- `Host`
- `Port`
- `User`
- `Password`
- `Database type`: Currently supports MySQL, PostgreSQL, SQL Server, Oracle, SQLite 3.
- `Database`: The name of the database.
- `Table`: The name of the table. If the table does not exist, it will be created.

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

! INFO

After filling in all the fields, please remember to **save** the configuration. Then click the **sync** button to load the policy rules. The policy rules will be displayed in the table below.

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	*	*	 

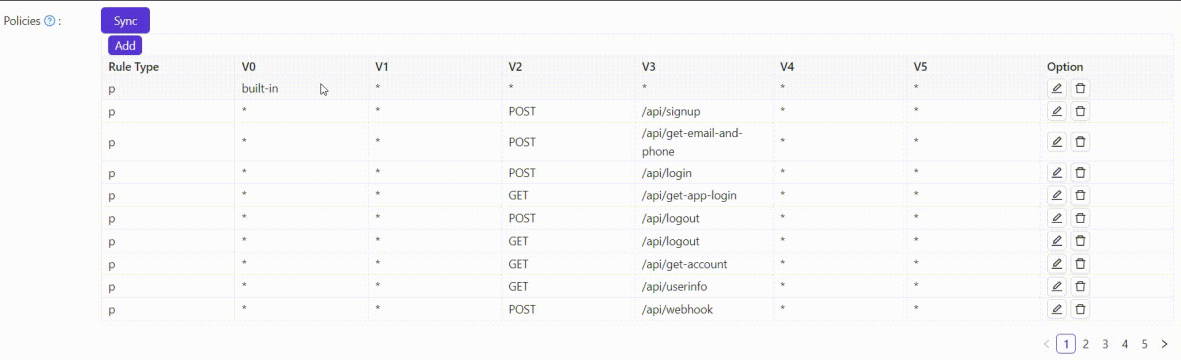
< 1 2 3 4 5 >





















Is enabled ⓘ : T

Basic CRUD Operations

If you have successfully connected the adapter, you can perform basic CRUD operations on 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	*	*	 

TIP

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

- Edit

Table: casbin_rule

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 >

Providers

Overview

Add third-party services to your application

OAuth

22 items

Email

5 items

SMS

5 items

 **Notification**

7 items

 **Storage**

9 items

 **SAML**

4 items

 **Payment**

5 items

 **Captcha**

7 items

 Web3

2 items

Overview

Casdoor utilizes providers to offer third-party services for the platform. In this chapter, you will learn how to add providers to Casdoor.

What We Have

Currently, we have six types of providers:

- **OAuth Providers**

Casdoor allows users to sign in through other OAuth applications. You can add GitHub, Google, QQ, and many other OAuth applications to Casdoor. For more details, please refer to the [OAuth](#) section.

- **SMS Providers**

Casdoor sends SMS to users when they want to verify their phone numbers. SMS providers are used to send SMS in Casdoor.

- **Email Providers**

Email providers are similar to SMS providers.

- **Storage Providers**

Casdoor allows users to store files using the local file system or cloud OSS services.

- **Payment Providers**

Casdoor can add payment providers, which will be used to add payment

methods to products on the product page. Currently, the supported payment providers include Alipay, WeChat Pay, PayPal, and GC.

- **Captcha Providers**

Casdoor supports configurable captcha in user flows. Currently, the supported captcha providers include Default Captcha, reCAPTCHA, hCaptcha, Alibaba Cloud Captcha, and Cloudflare Turnstile.

How to Configure and Use

Scope

Providers have different scopes, which are determined by the creator. Only Administrators have the permission to add and configure providers. There are two types of Administrators in Casdoor:






- **Global Administrator:** All users under the `built-in` organization and the users who enable `IsGlobalAdmin`. The providers created by the Global Administrator can be used by all applications.
- **Organization Administrator:** Users who enable `IsAdmin`. The providers created by the Organization Administrator can **only** be used by the applications under the organization (under development...).

Add to Application

Follow the steps below to add providers to your application. Note that you cannot use the provider in your application until you have added it.






1. Go to the application edit page and add a new provider row.

Providers ? :

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

2. Select a provider that you want to add to the application. You will see all the providers that the application can use.

Providers ? :






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

Preview ? :

provider_email_submail
provider_4olfdm
provider_casdoor_bilibili
provider_casdoor_okta
provider_casdoor_alipay
provider_casdoor_slack
provider_casdoor_steam
provider_casdoor_infocflow

3. For OAuth and Captcha providers, you can configure their usage. See [OAuth](#)

and [Captcha](#) for more information.

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

Finally, save the configuration. You can now try using the provider in your application.

OAuth

Overview

Add OAuth providers to your application

Google

Add the Google OAuth provider to your application

Google One Tap

Learn how to add Google One Tap support to your application

GitHub

Add GitHub OAuth provider to your application

 **LinkedIn**

Add LinkedIn OAuth provider to your application

 **Facebook**

Add the Facebook OAuth provider to your application.

 **AD FS**

Add AD FS as a third-party service to complete authentication.

 **Azure AD**

Add Azure AD as a third-party service to complete authentication

 **Azure AD B2C**

Add Azure AD B2C as a third-party service to complete authentication

 **Custom OAuth**

Add your custom OAuth provider to Casdoor

 **Okta**

Add Okta OAuth provider to your application

 **Twitter**

Add Twitter OAuth provider to your application

 **Weibo**

Add Weibo OAuth provider to your application

 **WeChat**

Add WeChat OAuth provider to your application

 **WeCom**

Add WeCom OAuth provider to your application

 **Tencent QQ**

Add Tencent QQ OAuth provider to your application

 **DingTalk**

Add DingTalk OAuth provider to your application

 **Steam**

Add the Steam OAuth provider to your application

 **Gitee**

Add Gitee OAuth provider to your application

 **Baidu**

Add Baidu OAuth provider to your application

 **Infoflow**

Add Infoflow OAuth provider to your application

































 **Lark**













































Add Lark OAuth provider to your application

Overview

Casdoor allows for the use of other OAuth applications as a sign-in method.

Currently, Casdoor supports multiple OAuth application providers. The icons of these providers will be displayed on the login and signup pages once they have been added to Casdoor. The following are the providers that Casdoor supports:

Provider	Logo	Provider	Logo	Provider	Logo	Provider	Logo
ADFS		Alipay		Amazon		Apple	
Auth0		Azure AD		Azure AD B2C		Baidu	
Bilibili		Bitbucket		Box		Casdoor	
Cloud Foundry		Dailymotion		Deezer		DigitalOcean	
DingTalk		Discord		Tiktok		Dropbox	
Eve Online		Facebook		Fitbit		Gitea	
Gitee		GitHub		GitLab		Google	
Heroku		InfluxCloud		Infocflow		Instagram	

Provider	Logo	Provider	Logo	Provider	Logo	Provider	Logo
Intercom		Kakao		Lark		Lastfm	
Line		LinkedIn		Mailru		Meetup	
Microsoft		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		Battle.net	

We will show you how to apply for a third-party service and add it to Casdoor.

Apply to become a developer

Before this, there are some general concepts you need to understand.

- **RedirectUrl**, Redirect address after authentication, fill in your application address, such as `https://forum.casbin.com/`
- **Scope**, Permission granted to you by the user, such as basic profile, Email address and posts and others.
- **ClientId/AppId, ClientKey/AppSecret**, This is the most important information, and it is what you need to get after you apply for a developer account. You can not share the key/secret with anyone.

Add an OAuth provider

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

Application Setup


1. Click on **Application** in the top bar and select the desired application to edit.
2. Click on the provider add button and choose the newly added provider.
3. Modify the provider's permissions, such as enabling registration, login, and unbinding.
4. You're all set!

Google

To set up the Google OAuth provider, please go to the [Google API console](#) and log in using your Google account.

Project name *
My Casdoor ?

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

Location *
 No organization [BROWSE](#)

Parent organization or folder

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

Next, navigate to the OAuth consent screen tab to configure the OAuth consent screen.

Google Cloud Platform My Casdoor Search products and resources

API APIs & Services OAuth consent screen

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

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

User Type

Internal ?

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

External ?

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

CREATE

Register your Google app by following these steps:

Edit app registration

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

App information

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

App name *

The name of the app asking for consent

User support email *

For users to contact you with questions about their consent

App logo

[BROWSE](#)

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

App domain

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

Application home page

Afterward, go to the Credential tab.

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

Create a credential for your app:

[←](#) 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 *



ⓘ ENSURE THAT YOU SET THE AUTHORIZED REDIRECT URIS CORRECTLY

In the Google OAuth configuration, the **Authorized redirect URIs** must be set to your Casdoor's callback URL, while the **Redirect URL** in Casdoor should be set to your application's callback URL.

For more details, please refer to the [App configuration](#).

After creating the Client ID, you will obtain the **Client ID** and **Client Secret**.

OAuth client created

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



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

Your Client ID

487708653175-11oih9gfb2u3tvfp6684qaes5ujjdca.apps.gc

Your Client Secret

HbxoqxxkGSs11CVRuMTVvK57

DOWNLOAD JSON

OK

Add the Google OAuth provider and enter the `Client ID` and `Client Secret` in your Casdoor.

Edit Provider

Name [?](#):

Display name [?](#):

Category [?](#):

Type [?](#):

Client ID [?](#):

Client secret [?](#):

Provider URL [?](#):

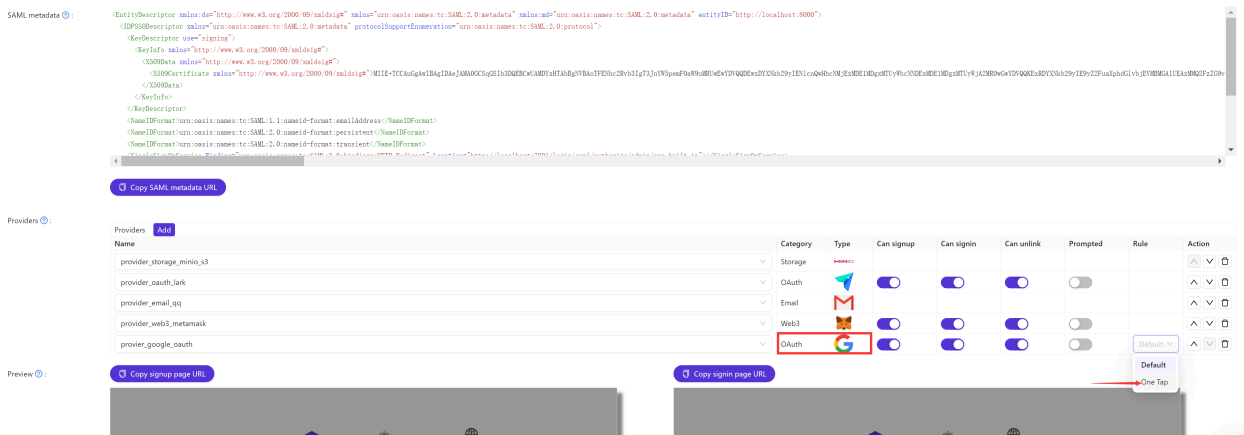
You can now use Google as a third-party service to complete authentication.

Google One Tap

Step 1: Configure Your Application

If you want to allow login through Google One Tap, you'll need to add Google OAuth Provider to your application. For information on how to do this, please refer to [Google's documentation](#).

Once you've added the Google OAuth Provider, navigate to the application edit page, add the Google OAuth Provider, and switch the **Rule** from **Default** to **One Tap**.



Step 2: Logging In with Google One Tap

With the setup completed, users can now log in with Google One Tap.

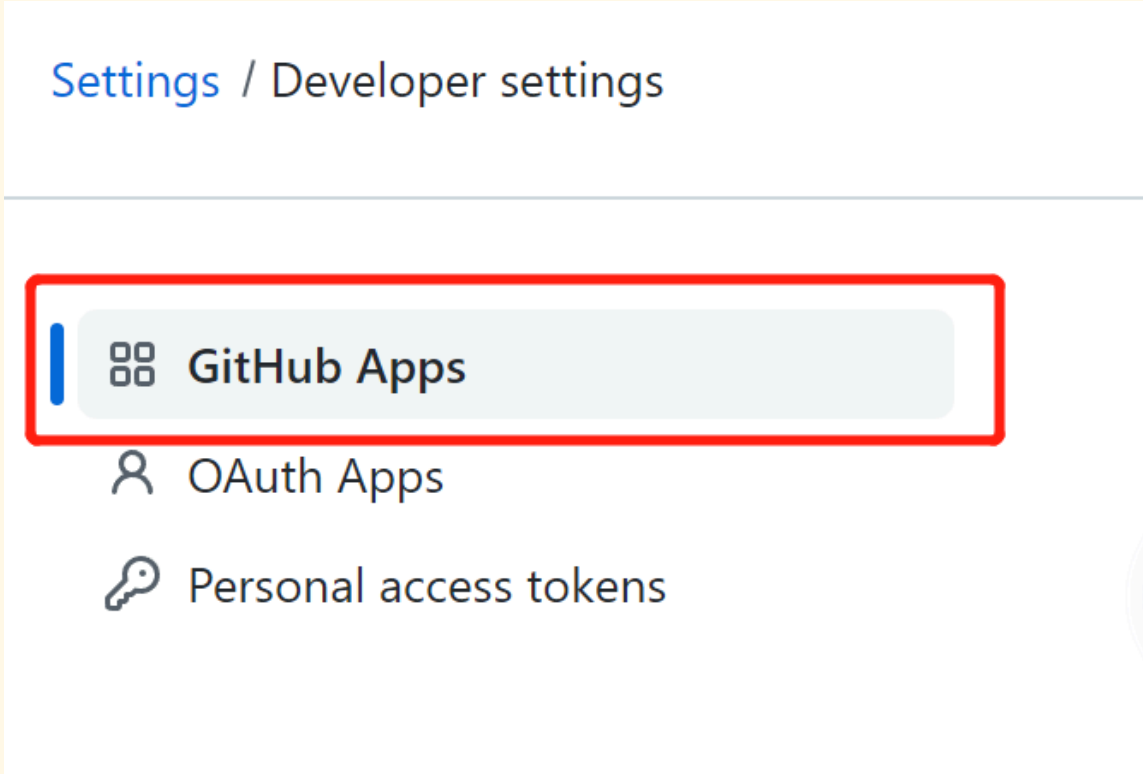
GitHub

GitHub OAuth supports both the web application flow and device flow. Please continue reading to obtain OAuth credentials.

First, please visit the [GitHub developer settings](#) to register a new GitHub App.

 CAUTION

Tricks: We recommend that you use GitHub Apps to replace OAuth Apps because GitHub Apps can add multiple redirect URIs, which can bring convenience when deploying test and production environments. The [GitHub](#) official also recommends using GitHub Apps instead of OAuth Apps.



Then fill in the GitHub App name, Homepage URL, description, and Callback URL.

GitHub App name *

The name of your GitHub App.

Write

Preview

Markdown supported

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

Homepage URL *

The full URL to your GitHub App's website.

Identifying and authorizing users

Add Callback URL

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

Callback URL

 Delete

Callback URL

 Delete

⚠️ SET THE AUTHORIZATION CALLBACK URL CORRECTLY

In the GitHub App config, the **Callback URL** must be your Casdoor's **callback URL**, and the **Redirect URL** in Casdoor should be your application's callback URL.

For more details, please read [App config](#).

After registering your GitHub App, you can now generate your `Client Secret`!

About

Owned by: [redacted]

App ID: [redacted]



Client ID: lv1 [redacted] d2e

[Revoke all user tokens](#)

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

Client secrets

[Generate a new client secret](#)

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

Add a GitHub OAuth provider and fill in the `Client ID` and `Client Secret` in your Casdoor.

Edit Provider

Name ⓘ:

Display name ⓘ:

Category ⓘ: ▼

Type ⓘ: ▼

Client ID ⓘ:


Client secret ⓘ:

Provider URL ⓘ:

Now you can use GitHub as a third-party service to complete authentication.

LinkedIn

To set up the LinkedIn OAuth provider, please go to the [LinkedIn Developer](#) page to create a new app.


 DEVELOPERS Products Docs and tools ▾ Resources ▾ My apps ▾

Create an app


* indicates required

App name*

LinkedIn Page*

 This action can't be undone once the app is saved.


The LinkedIn Company Page you select will be associated with your app. Verification can be done by a Page Admin. Please note this cannot be a member profile page. [Learn more](#)

[+ Create a new LinkedIn Page](#) 

Privacy policy URL

App logo*

This is the logo displayed to users when they authorize with your app



After filling in the form above and creating your app, you'll need to verify the LinkedIn page associated with the app.



Identity Cloud Login

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

[Settings](#)

[Auth](#)

[Products](#)

[Analytics](#)

[Team members](#)

App settings

Delete app

Company:



Identity Cloud Documentation
Computer Software; 1-10 employees

Verify




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

[Learn more](#)

NOTE

Only the company page administrator can verify your app and grant permission to it.

Once your app is verified, you can continue:




Identity Cloud Login
Client ID: 860t47n8b4jh7w | Created: Sep 4, 2020

[Settings](#) [Auth](#) [Products](#) [Analytics](#) [Team members](#)


Products

Additional available products




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

[Select](#)



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

[Select](#)



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


[Select](#)

Add authorized redirect URLs for your app as your Casdoor callback URL.

Authorized redirect URLs for your app

No redirect URLs added

[+ Add redirect URL](#)

 SET AUTHORIZED REDIRECT URLS CORRECTLY

In the LinkedIn OAuth configuration, the `authorized redirect URLs` must be your Casdoor's callback URL, and the `Redirect URL` in Casdoor should be your application's callback URL.

For more details, please read the [App Config](#) section.

You can then obtain your `Client ID` and `Client Secret`.


Application credentials

Authentication keys

Client ID:

860t47n8b4jh7w

Client Secret:

..... 

Add a LinkedIn OAuth provider and fill in the `Client ID` and `Client Secret` in your Casdoor.

Edit Provider

Save

Name [?](#) :

my_linkedin_provider

Display name [?](#) :

Linkedin provider

Category [?](#) :

OAuth

Type [?](#) :

LinkedIn

Client ID [?](#)

860t47n8b4jh7w

Client secret [?](#)

Now you can use LinkedIn as a third-party service to complete authentication!







Facebook

To set up the Facebook OAuth provider, please go to the [Facebook Developer](#) website and create a new app.

Select the type of app you are going to create.

Select an app type ✕

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

-  Create or manage business assets like Pages, Events, Groups, Ads, Messenger and Instagram Graph API using the available business permissions, features and products.
-  **Consumer**
Connect consumer products, and permissions, like Facebook Login and Instagram Basic Display to your app.
-  **Instant Games**
Create an HTML5 game hosted on Facebook.
-  **Gaming**
Connect an off-platform game to Facebook Login.
-  **Workplace**
Create enterprise tools for Workplace from Facebook.
-  **None**
Create an app with combinations of consumer and business permissions and products.

[Learn More About App Types](#) Cancel Continue







After entering your name and contact email, you will be taken to the Facebook Developer dashboard.

FACEBOOK for Developers Docs Tools Support My Apps Search developer documentation


Casdoor App ID: 1231340483981478 In development

- Dashboard
- Settings
- Roles
- Alerts
- App Review
- Products [Add Product](#)

Add a Product

 Facebook Login The world's number one social login product. Read Docs Set Up	 Audience Network Monetize your app and grow revenue with ads from Facebook advertisers. Read Docs Set Up	 App Events Understand how people engage with your business across apps, devices, platforms and websites. Read Docs Set Up
 Messenger Customize the way you interact with people on Read Docs Set Up	 Webhooks Subscribe to changes and receive updates, in real Read Docs Set Up	 Instant Games Create a cross-platform HTML5 game hosted on Read Docs Set Up

Next, set up Facebook login:



Facebook Login

The world's number one social login product.

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

Choose the Web platform for this app:

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



iOS



Android



Web



Other

After filling in the website URL, go to Facebook Login > Settings and enter valid OAuth Redirect URIs.

Client OAuth Settings

- Client OAuth Login**
Enables the standard OAuth client token flow. Secure your application and prevent abuse by locking down which token redirect URIs are allowed with the options below. Disable globally if not used. [?]
- Web OAuth Login**
Enables web-based Client OAuth Login. [?]
- Force Web OAuth Reauthentication**
When on, prompts people to enter their Facebook password in order to log in on the web. [?]
- Use Strict Mode for Redirect URIs**
Only allow redirects that exactly match the Valid OAuth Redirect URIs. Strongly recommended. [?]
- Enforce HTTPS**
Enforce the use of HTTPS for Redirect URIs and the JavaScript SDK. Strongly recommended. [?]
- Embedded Browser OAuth Login**
Enable webview Redirect URIs for Client OAuth Login. [?]

Valid OAuth Redirect URIs

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

Valid OAuth redirect URIs.

⚠️ SET AUTHORIZED REDIRECT URLS CORRECTLY

In the Facebook OAuth configuration, the **Valid OAuth Redirect URIs** must be your Casdoor's callback URL, and the **Redirect URL** in Casdoor should be your application's callback URL.

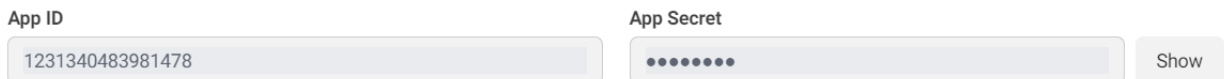
For more details, please read the [App Configuration](#) section.

The basic app configuration is almost complete!

Switch the mode from **In development** to **Live** in the top bar of the dashboard.



Now you can use your **App ID** and **App Secret** in Casdoor.



Add the Facebook OAuth provider and fill in the **Client ID** and **Client Secret** with the **App ID** and **App Secret** from your Casdoor.

修改提供商 **保存**

名称 [?]: my_facebook_provider

显示名称 [?]: Facebook provider

分类 [?]: OAuth

类型 [?]: Facebook

Client ID [?]: 1231340483981478

Client secret [?]: *****

You can now use Facebook as a third-party service for authentication!

AD FS

To set up Active Directory Federation Service, please refer to the [AD FS documentation](#) for a basic understanding of ADFS, and consult the [AD FS Deployment Guide](#) for instructions on setting up an AD FS server. Ensure that you have a fully operational AD FS server before proceeding to the next steps.

Step 1: Enabling OAuth via AD FS

For detailed instructions on creating an app step by step, refer to the [Enabling OAuth Confidential Clients with AD FS](#) guide.

By the end of this step, you should have obtained a client ID and client secret as shown in the following screenshots:

AD FS

File Action View Window Help

Application Groups

Name	Description
------	-------------

Actions

Application Groups

Add Application Group Wizard

Server application

Steps

- Welcome
- Server application
- Configure Application Credentials
- Configure Web API
- Apply Access Control Policy
- Configure Application Permissions
- Summary
- Complete

Name: ADFS0AUTHCC - Server application

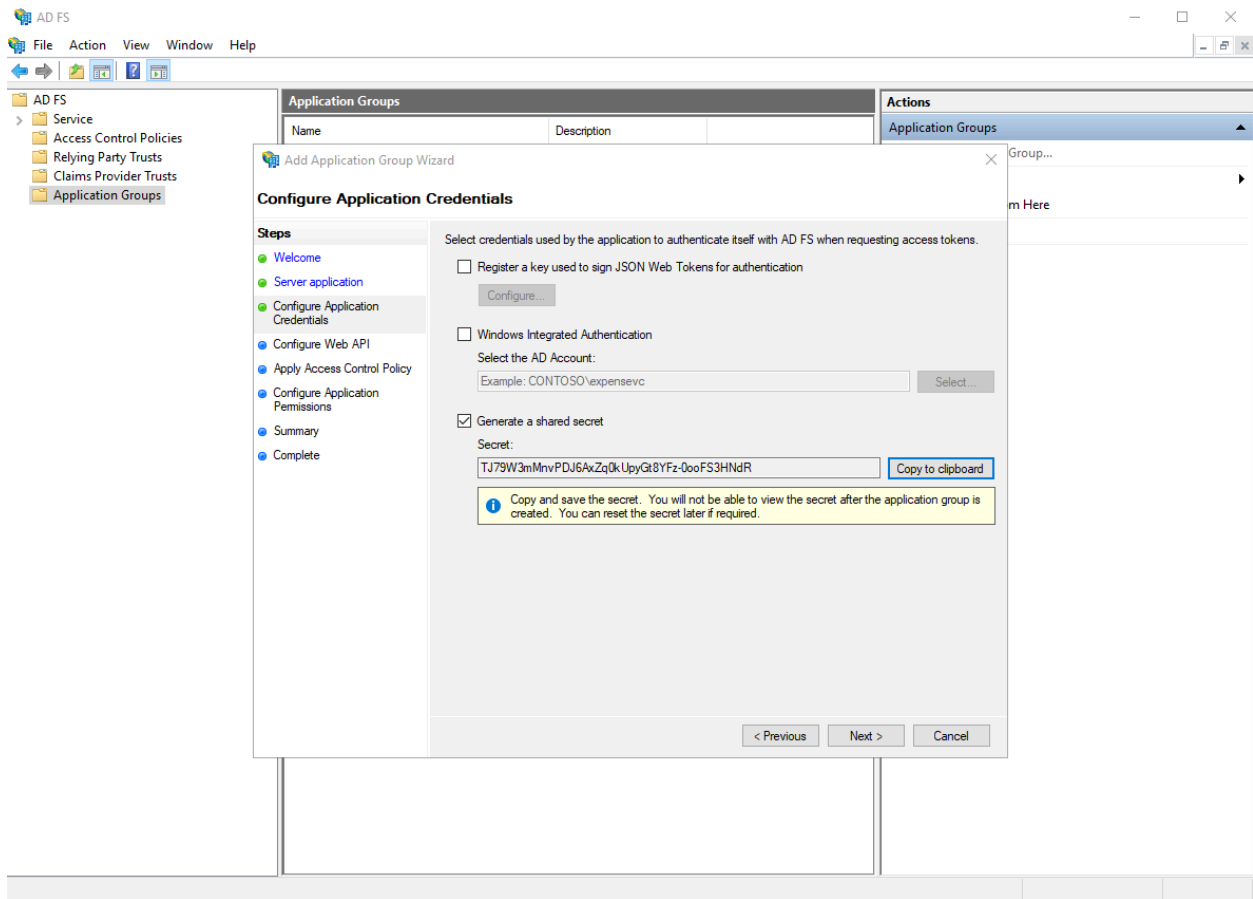
Client Identifier: 4e7a28c0-2e4b-403b-8251-b6ffb1c3b6a

Redirect URI: Example: <https://Contoso.com> Add Remove

<https://localhost:44323>

Description:

< Previous Next > Cancel



The client identifier in the first picture and the secret in the second picture should be used as the client ID and client secret in the OAuth setup.

Enabling Casdoor AD FS Provider


Add an AD FS provider and enter the "Client ID" and "Client Secret" in your Casdoor settings.

Edit Provider

Name [?]:

Display name [?]:

Category [?]:

Type [?]: 

Client ID [?]:

Client secret [?]:

Domain [?]:

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

Azure AD

Introduction

Azure Active Directory (Azure AD) simplifies application management by providing a single identity system for cloud and on-premises applications. Software as a Service (SaaS) applications, on-premises applications, and Line of Business (LOB) applications can be added to Azure AD. Users can then log in once for secure and seamless access to these applications, as well as Office 365 and other business applications provided by Microsoft.

How to use?

The steps to register an app are shown below.

Step 1: Register an application

First, [register](#) an application and choose the account type as needed. The demo station uses the type shown below.



[Home](#) >

Register an application ⋮

* Name

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

Supported account types

Who can use this application or access this API?

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

[Help me choose...](#)

Redirect URI (optional)

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

[Register](#)

Step 2: Create a client secret

Create a `client secret` and save the value because it will be used later.

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

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

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

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

+ New client secret

Description	Expires	Value	Secret ID
casdoor	1/8/2023	3Xr8Q~dFau2Hwyhg6Y8Upb53PCFbuF...	f3c7d37c-1def-4e29-b75f-457fa7c081e8

Step 3: Add redirect URIs

Follow the example in the picture to add the redirect URIs for Casdoor.

Platform configurations

Depending on the platform or device this application is targeting, additional configuration options, such as redirect URIs, specific authentication settings, or fields specific to the platform.

+ Add a platform

Supported account types

Who can use this application or access this API?

Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
All users with a work or school, or personal Microsoft account can use your application. Office 365 subscribers.

To change the supported accounts for an existing registration, use the manifest editor. Take care as changing properties may cause errors for personal accounts. [Learn more about these restrictions.](#)

Save Discard

Configure Web

< All platforms

* Redirect URIs

The URIs we will accept as destinations when returning after successfully authenticating or signing out. The request to the login server should match one listed here. [Learn more about Redirect URIs and their restrictions.](#)

http://localhost:8000/callback

Front-channel logout URL

This is where we send a request to have the application sign out. This is required for single sign-out to work correctly.

e.g. https://example.com/logout

Implicit grant and hybrid flows

Request a token directly from the authorization endpoint using the SPA architecture and doesn't use the authentication API via JavaScript, select both access tokens and ID tokens. [Learn more about tokens.](#)

Configure Cancel

Step 4: Grant admin consent

The `user.read` API is open by default. You can add more scopes according to your needs. Finally, remember to grant admin consent.

Search (Ctrl+/)

Refresh | Got feedback?

- Overview
- Quickstart
- Integration assistant
- Manage
 - Branding & properties
 - Authentication
 - Certificates & secrets
 - Token configuration
 - API permissions**
 - Expose an API
 - App roles
 - Owners
 - Roles and administrators
 - Manifest
- Support + Troubleshooting
 - Troubleshooting
 - New support request

Successfully granted admin consent for the requested permissions.

Starting November 9th, 2020 end users will no longer be able to grant consent to newly registered multitenant apps without verified publishers. [Add MPN ID to verify publisher](#)

The "Admin consent required" column shows the default value for an organization. However, user consent can be customized per permission, user, or app. This column may not reflect the value in your or app will be used. [Learn more](#)

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

+ Add a permission Grant admin consent for Default Directory

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

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

Step 5: Create AzureAD provider in Casdoor

The last step is to add an AzureAD OAuth provider and fill in the `Client ID` and `Client Secret` in your Casdoor.

Edit Provider

Save

Save & Exit

Name ? :

provider_casdoor_azuread

Display name ? :

Casdoor AzureAD

Category ? :

OAuth

Type ? :

AzureAD

Client ID ?

621cc0f0-055f-433f-9894-bfa1bfde169d

Client secret ?

Provider URL ? :

https://portal.azure.com/#view/Microsoft_AAD_RegisteredApps/Appli

Save

Save & Exit

Azure AD B2C

Introduction

Azure AD B2C is a customer identity access management solution, supporting standards like OpenID Connect, OAuth 2.0, and SAML. It allows the integration of consumer-facing applications with a scalable and customizable identity management solution.

How to use?

The steps to set up Azure AD B2C for authentication are shown below.

Step 1: Create a B2C Tenant

First, create a B2C Tenant in your Azure portal.

Step 2: Register an application

Register an application within your B2C tenant.



[Home](#) >

Register an application ...

* Name

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

Supported account types

Who can use this application or access this API?

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

[Help me choose...](#)

Redirect URI (optional)

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

[Register](#)

Step 3: Create a client secret

Create a `client secret` for your application and save the value as it will be used later.

Search (Ctrl+/) << Got feedback?

Manage

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

Support + Troubleshooting

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

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

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

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

+ New client secret

Description	Expires	Value	Secret ID
casdoor	1/8/2023	3Xr8Q~dFau2Hwyhg6Y8Upb53PCFbuF...	f3c7d37c-1def-4e29-b75f-457fa7c081e8

Step 4: Add redirect URIs

Add the redirect URIs for your application in the Azure AD B2C settings.

casdoor | Authentication

Search (Ctrl+/) << Got feedback?

Overview

- Quickstart
- Integration assistant

Manage

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

Platform configurations

Depending on the platform or device this application is targeting, additional configuration options, such as redirect URIs, specific authentication settings, or fields specific to the platform.

+ Add a platform

Supported account types

Who can use this application or access this API?

- Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)**
All users with a work or school, or personal Microsoft account can use your application. Office 365 subscribers.

*** Redirect URIs**

The URIs we will accept as destinations when returning after successfully authenticating or signing out using this application. The request to the login server should match one listed here. [Learn more about Redirect URIs and their restrictions.](#)

Front-channel logout URL

This is where we send a request to have the application sign out. The request to the login server should match one listed here.

Implicit grant and hybrid flows

Request a token directly from the authorization endpoint using the SPA architecture (SPA) and doesn't use the authentication API via JavaScript, select both access tokens and ID tokens. [Learn more about tokens.](#)

Step 5: Define User Flows

Define user flows in Azure AD B2C to manage how users sign up, sign in, and manage their profiles.

Step 6: Create Azure AD B2C provider in Casdoor

Finally, add an Azure AD B2C OAuth provider in Casdoor, using the `Client ID` and `Client Secret` from your B2C tenant.

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/Apply










Save Save & Exit

Custom OAuth

NOTE

Casdoor supports custom providers. However, the custom providers must follow the standard process of 3-legged OAuth, and the return values of `Token URL` and `UserInfo URL` must conform to the format specified by Casdoor.

To create a new custom provider, navigate to the provider page of Casdoor, and select "Custom" in the Type field. You will then need to fill in `Client ID`, `Client Secret`, `Auth URL`, `Scope`, `Token URL`, `UserInfo URL`, and `Favicon`.

Type 	<input type="text" value="Custom"/>
Auth URL 	<input type="text" value="https://door.casdoor.com/login/oauth/authorize"/>
Scope 	<input type="text" value="openid profile email"/>
Token URL 	<input type="text" value="https://door.casdoor.com/api/login/oauth/access_token"/>
UserInfo URL 	<input type="text" value="https://door.casdoor.com/api/userinfo"/>
Favicon 	URL  : <input type="text" value="🔗"/>
	Preview:
Client ID 	<input type="text"/>
Client secret 	<input type="text"/>

- `Auth URL` is the custom provider's OAuth login page address.

If you fill in `https://door.casdoor.com/login/oauth/authorize` as the `Auth URL`, then, when a user

logs in with this custom provider, the browser will first redirect to

```
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}`
```

After authorization is completed, the custom provider should redirect to

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

After this step, Casdoor will recognize the code parameter in the URL.

- `Scope` is the scope parameter carried when accessing the `Auth URL`, and you should fill it in as per the custom provider's requirements.
- `Token URL` is the API endpoint for obtaining the accessToken.

Once you obtain the code in the previous step, Casdoor should use it to get the accessToken.

If you fill in `https://door.casdoor.com/api/login/oauth/access_token` as the `Token URL`, then Casdoor will access it using the following command

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

The custom provider should return at least the following information:

```
{
  "access_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjE2IiwiaXNjaWkiOiJ1bnR5cCI6IjE2IiwiaWF0Ijoi",
  "refresh_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjE2IiwiaXNjaWkiOiJ1bnR5cCI6IjE2IiwiaWF0Ijoi",
  "token_type": "Bearer",
  "expires_in": 10080,
  "scope": "openid profile email"
}
```

- `User Info URL` is the API endpoint for obtaining user information via the accessToken.

If you fill in `https://door.casdoor.com/api/userinfo` as the `User Info URL`, then Casdoor will access it using the following command

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

The custom provider should return at least the following information:

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

- `Favicon` is the logo URL of a custom provider.

This logo will be displayed on Casdoor's login page together with other third-party login providers.

Okta

To set up the Okta OIDC provider, first visit [Okta Developer](#) and sign up to get a developer account.

Navigate to the Applications > Applications tab, click **Create App Integration**, select a Sign-in method of **OIDC - OpenID Connect**, and choose an Application type of **Web Application**, then click **Next**.

Create a new app integration ✕

Sign-in method
[Learn More](#)

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

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

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

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

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

Enter the Sign-in redirect URIs, such as `https://door.casdoor.com/callback`.

Sign-in redirect URIs Allow wildcard * in sign-in URI redirect.

Okta sends the authentication response and ID token for the user's sign-in request to these URIs

[Learn More](#)

×

[+ Add URI](#)

In the **Assignments** section, define the type of **Controlled access** for your app and then click **Save** to create the app integration.

Now you will have the `Client ID`, `Client secret`, and `Okta domain`.

Client Credentials [Edit](#)

Client ID

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

Client secret

Secret used by the client to exchange an authorization code for a token. This must be kept confidential! Do not include it in apps which cannot keep it secret, such as those running on a client.

General Settings [Edit](#)

Okta domain

Add an Okta OAuth provider in the Casdoor dashboard by entering your `Client ID`, `Client secret`, and `Domain`.

Edit Provider Save Save & Exit

Name [?]:

Display name [?]:

Category [?]:

Type [?]:

Client ID [?]:

Client secret [?]:

Domain [?]:

Provider URL [?]:

Save Save & Exit

⚠️ SET DOMAIN CORRECTLY

Note that the `Domain` is not just the `Okta domain`; `/oauth2/default` should be appended to it.

Visit [Okta docs on authorization servers](#) to get more details.

Now you can use Okta as a third-party service to complete authentication.

Twitter

Twitter (Work in Progress)

Applying for a developer account on Twitter can be a bit cumbersome due to the strict official restrictions. It may be more challenging compared to other third-party platforms.

To get started, visit the [Developer Portal](#) and create an account if you don't have one. Twitter requires you to provide detailed information about your application for a developer account. Make sure to fill in the information accurately to avoid any issues during the review process.

Once your application is approved, you can proceed to create an application. You need to complete two important tasks in the **Authentication settings** section:

1. Manually enable **3-legged OAuth**. This is necessary for features such as "Sign in with Twitter" and posting Tweets on behalf of other accounts.
2. Enable **Request email address from users** to obtain the user's email address.

Make sure to carefully fill in the callback address and other required information for your application.

Weibo

Weibo ✓

Applying for a developer account with Weibo is not difficult, but the process can be slow, taking about 2-3 days.

To get started, visit the [Developer Website](#) and fill in the required basic information. Then, you will need to wait for a thorough review...

Once your application is approved, you will receive the Client Id and Client Secret.

WeChat

WeChat ✓

To add WeChat OAuth provider to your application, follow these steps:

1. Visit the [WeChat developer platform](#) and register as a developer.
2. After your web application or mobile application is approved, you will receive your App ID and App Secret.

Name ? : provider_casdoor_wechat

Display name ? : Casdoor WeChat

Organization ? : admin (Shared)

Category ? : OAuth

Type ? : WeChat

Client ID ? : wx049c70e6c2027b0b

Client secret ? : ***

Client ID 2 ? : wxe933a9cd81c396d1

Client secret 2 ? : ***

Use WeChat

Media Platform in PC ? :

Access token ? :

Follow-up action ? : Use WeChat Open Platform to login Use WeChat Media Platform to login

Provider URL ? : <https://open.weixin.qq.com/>

服务器配置(已启用)

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

令牌(Token) 123

The WeChat provider offers two different sets of keypairs:

- The first keypair (Client ID, Client Secret) is for the WeChat Open Platform (📺📺📺📺📺) and is designed for the PC login scenario. It allows you to display a QR code in the PC browser, which users can scan using the WeChat app on their mobile phone to sign in.
- The second keypair (Client ID 2, Client Secret 2) and Access Token

field is for the `WeChat Media Platform (*****)` and is intended for the inside-WeChat-app login scenario. `Access Token` field is the `Token` you fill in the `server configuration` of the `WeChat Media Platform (*****)`. It enables users to log in with the WeChat built-in browser inside the WeChat mobile app, which will redirect them to your `WeChat Official Account (****)` to log in. Please note that WeChat does not support logging in outside of the WeChat app in other mobile browsers or apps. This limitation is imposed by WeChat and not by Casdoor.

If you fill in the second keypair (`Client ID 2`, `Client Secret 2`), fill the `Access Token` field and enable the `Enable QR code` switch, then you can choose to login directly using the information from the `WeChat Media Platform (*****)` after scanning the QR code, or use the information from the `WeChat Open Platform (*****)` to login, if you choose `use Wechat Open Platform to login`, after user follow the the WeChat official account (`****`), users will be required to scan the QR code of `WeChat Open Platform (*****)` to login. Casdoor will ask the user to follow the WeChat official account (`****`) before proceeding with the login process when the user clicks on the WeChat button to login. It's important to note that this functionality is only available in the PC login scenario because a mobile phone cannot scan the QR code by itself. When used in the mobile scenario (i.e., the WeChat built-in browser inside the WeChat mobile app), Casdoor will automatically skip this step.

TIP

We recommend setting both key sets at the same time and linking your `WeChat Open Platform (*****)` account and `WeChat Media Platform (*****)` account together inside the `WeChat Open Platform (*****)`. This will allow Casdoor to recognize a WeChat user logged in through both PC and mobile as the same user.

 NOTE

Due to the limitations of WeChat OAuth, there is currently no way to log in via WeChat in a third-party mobile app or in a mobile browser other than the WeChat app. The mobile login must happen inside the WeChat app for now.

For more detailed information, please visit the [WeChat Open Platform](#).

WeCom

Introduction

WeCom provides an authorized login method using OAuth, which allows you to obtain members' identity information directly from the webpage opened by the WeCom terminal, eliminating the need for a login process.

There are two types of applications: **internal** applications and **third-party** applications.

Basic Configuration

To configure a WeCom provider, you need to provide the following parameters:

Parameter Description:

Parameter	Description
Sub type	Internal or Third-party
Method	Silent or Normal
Client ID	The enterprise CorpID
Client secret	The enterprise CorpSecret
Agent ID	Application agentid

! INFO

WeCom supports two authorization methods: **Silent authorization** and **normal authorization**.

Silent authorization: After the user clicks the link, the page is redirected to `redirect_URI? code=CODE&state=STATE`

Normal authorization: After the user clicks the link, a middle page is displayed for the user to choose whether to authorize or not. After the user confirms the authorization, they are redirected to

`redirect_uri?code=CODE&state=STATE`

For more details, please refer to the [official documentation](#).

More Information

For more information about internal applications, please refer to the [Internal Application](#) documentation.

For information about third-party applications, please refer to the [Third-Party Application](#) documentation.

Tencent QQ

Tencent QQ ✓

To add Tencent QQ OAuth provider to your application, visit the authentication platform of QQ - [Connect QQ](#).

First, you need to apply to [become a developer](#). After your application is approved, follow the instructions of the platform to obtain your Client Id and Client Secret.

DingTalk

DingTalk ✓

Configuring DingTalk

To configure DingTalk, visit the [DingTalk developer platform](#) and log in using your DingTalk account. Once you're on the platform, follow the instructions provided to obtain your `Client Id` and `Client Secret`. The corresponding terms in DingTalk are as follows:

Term	DingTalk Name
Client ID	AppKey
Client secret	AppSecret

In DingTalk, you can find the `Appkey` and `AppSecret` in the App Info.

基础信息

应用信息

开发管理

权限管理

应用功能

机器人与消息推送

事件与回调

登录与分享

酷应用


安全与监控

监控中心

部署与发布

版本管理与发布

应用信息

 casdoor
document

应用凭证

AgentId: 2687194261

AppKey: ding6dposoonm8u4t2g5

AppSecret: hE4cwQ4PjKDSp_ucHTBTqjAAfZfsNGkxwNg1q1FCiiTRW7apxJhzjFOjw46NfFWn

删除应用

删除操作不可逆，该应用所有信息将被删除，请谨慎操作。

删除

Make sure to add the **Redirect Domain**, which should be your Casdoor domain.

基础信息

应用信息

开发管理

权限管理

应用功能

机器人与消息推送

事件与回调

登录与分享

酷应用

安全与监控

监控中心

部署与发布

接入登录

添加重定向 URL 作为免登授权码跳转地址。了解更多

* 回调域名

请填写 HTTP/HTTPS 开头的 URL

添加

微应用回调的URL

http://localhost:7001

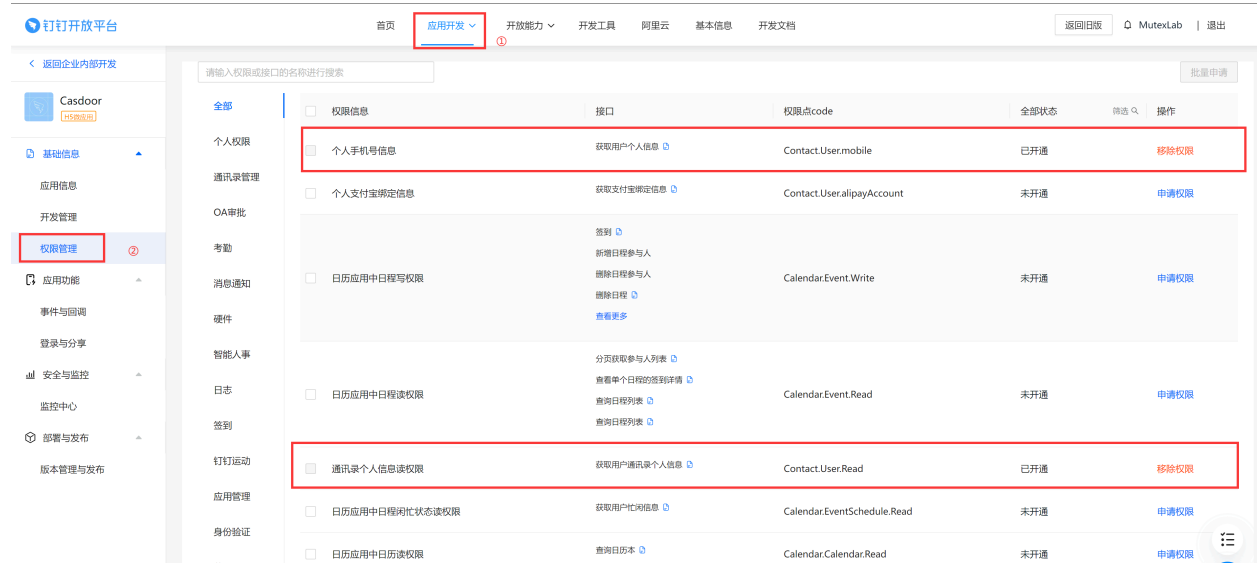
接入分享

嵌入分享SDK，实现一键登录后内容分享。了解更多

iOS 分享

For more detailed information, please refer to the [DingTalk developer docs](#).

Additionally, you need to add the following permissions to the DingTalk application:



Configuring Casdoor

Here's the final configuration for DingTalk:

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

Steam

Steam ✓

To add the Steam OAuth provider to your application, follow these steps:

1. Visit the [Steam WebAPI platform](#) and log in using your Steam account.
2. Apply for an API Key for your Casdoor domain or IP.
3. Fill in your API Key as the Client Secret into Casdoor. The ClientID does not need to be filled.
4. Make sure that your Steam account has games in order to apply for the API.

For more detailed information, please visit the [Steam WebAPI documentation](#).

Gitee

To set up the Gitee OAuth provider, please go to the [Gitee developer](#) website. If you haven't created applications before, the Gitee workbench will look like this:



You can then create your Gitee app.

创建第三方应用

应用名称 *

应用描述

应用主页 *

应用回调地址 * 

Enter the name, description, homepage, and callback URL, and carefully choose the permissions.

❗ SET THE AUTHORIZATION CALLBACK URL CORRECTLY

In the Gitee OAuth config, the `authorization callback URL` must be your Casdoor's callback URL, and the `Redirect URL` in Casdoor should be your application's callback URL.

For more details, please read the [App config guide](#).

After creating the Gitee app, you can obtain the `Client ID` and `Client Secrets`!

Casdoor (今日请求次数: 0 次)

应用名称 *

Casdoor

Client ID

300ff94d994a7597850bbafb2d5dc67929676dd8e7176b029e067dc6966ef9c4

Client Secret

60be2e4e0f3fb8286cfe9f129ab0c3d6b40718a964dade150a8095eb2748730c

重置 Client Secret

移除已授权用户的有效 Token

Add a Gitee OAuth provider and enter the `Client ID` and `Client Secrets` in your Casdoor.

Edit Provider

Save

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

Now you can use Gitee as a third-party service to complete authentication!

 CAUTION

Since Casdoor needs to obtain the user's email, the email option must be checked; otherwise, it will cause scope authorization errors.

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

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

Submit

Delete

Baidu

To set up the Baidu OAuth provider, please read the [Baidu documentation](#) and follow their steps to complete the [application creation](#).

开发者服务管理

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

← 创建工程

* 应用名称: 11/32

传统接入扩展: 合作网站

解决方案: 使用BAE

[创建](#)

After creating your app, the redirect URL should be set in the following position:

↶ Casdoor

- 基本信息
- 接入类型
- 其他应用
- 开发者服务
- Oauth2.0
- 安全设置**

基本信息

 名称:

Icon: 

ID: 25547043

API Key: Hn[redacted]yQmAp61

Add your Casdoor domain in the following position:

Casdoor

ID API Key Secret Key

基本信息

接入类型

其他应用

开发者服务

Oauth2.0

安全设置

安全设置

Implicit Grant授权方式 启用 禁用

授权回调页:

0/500

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

根域名绑定:

15/255

限制访问OpenAPI的Referer

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

应用服务器IP地址:

0/500

同时绑定访问OpenAPI服务器IP

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

确定 取消

⚠ CAUTION

This part is very different from the information provided in the Baidu documentation:

1. Adding the URL to the callback URL setting will most likely fail to validate the URL and cause the login to fail, so we add our domain name to the domain setting.
2. Only one URL or domain name can be added, which is very different from the documentation.

Then you can obtain the `Client ID` and `Client Secrets`.

 **Casdoor**


基本信息

接入类型

 其他应用

开发者服务


 OAuth2.0

 安全设置

基本信息

 名称: Casdoor

Icon: 

ID: 


Client ID API Key: HnhK2...QmAp61

Client Secret Secret Key: DTgBZ...1s1bLm1Gha

创建时间: 2022-01-22 16:20:05

更新时间: 2022-01-23 15:45:06

Add a Baidu OAuth provider and fill in the **Client ID** and **Client Secrets** in your Casdoor.

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

Edit Provider

Name: Baidu

Display name: Baidu

Category: OAuth

Type: Baidu

Client ID: HsN...nWT

Client secret: ***

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

Now you can use Baidu as a third-party service to complete authentication!

ⓘ GENERAL TROUBLESHOOTING

If you encounter a Baidu prompt that states your redirect URL is incorrect, here are some ways you might be able to fix it:

1. Add your domain name to the appropriate location and then reset the Secret (Baidu reset Secret has a bug, it will prompt you an error, but after refreshing the page the Secret has been refreshed).
2. If the above methods do not solve the problem, we suggest you delete the application and create a new one, and set your domain name first.

Another problem is that the user name returned by Baidu is masked, unlike their documentation which shows the user name and displayed name.

Therefore, we can currently only use the masked name as the user name.

Infoflow

To set up the Infoflow OAuth provider, please follow these steps:

1. Go to [Infoflow](#) and log in using your Infoflow account.
2. Visit the [Infoflow Application](#) page.



3. Register your Infoflow app.



4. Obtain the `AgentID`.

基本信息

应用logo: 

应用名称: Casdoor

应用介绍: Casdoor单点登录系统

功能: 应用

AgentID 应用ID: 55

5. Navigate to the Setting tab and create a new management group.



6. Add your structure to the address book permissions and give it the necessary permissions. Also, add the application you just created to the specified location.

通讯录权限

[修改](#)

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

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

- 成员ID
- 姓名
- 部门
- 头像
- 手机号
- 邮箱
- 登录帐号

应用权限

[修改](#)

应用权限	发消息	配置应用
Casdoor	<input checked="" type="radio"/>	<input type="radio"/>

7. Add the sensitive interface permissions as shown.

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

8. On the same page, you will find the `CorpID` and `Secret`.

开发者凭据

Client ID

CorpID	hir-216-1	
Secret	HgH1-NB	重置

Client Secret

9. Add an Infowflow OAuth provider to Casdoor and fill in the **Client ID**, **Client Secret**, and **Agent ID**.

Edit Provider Save Save & Exit

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

You can now use Infowflow as a third-party service for authentication.

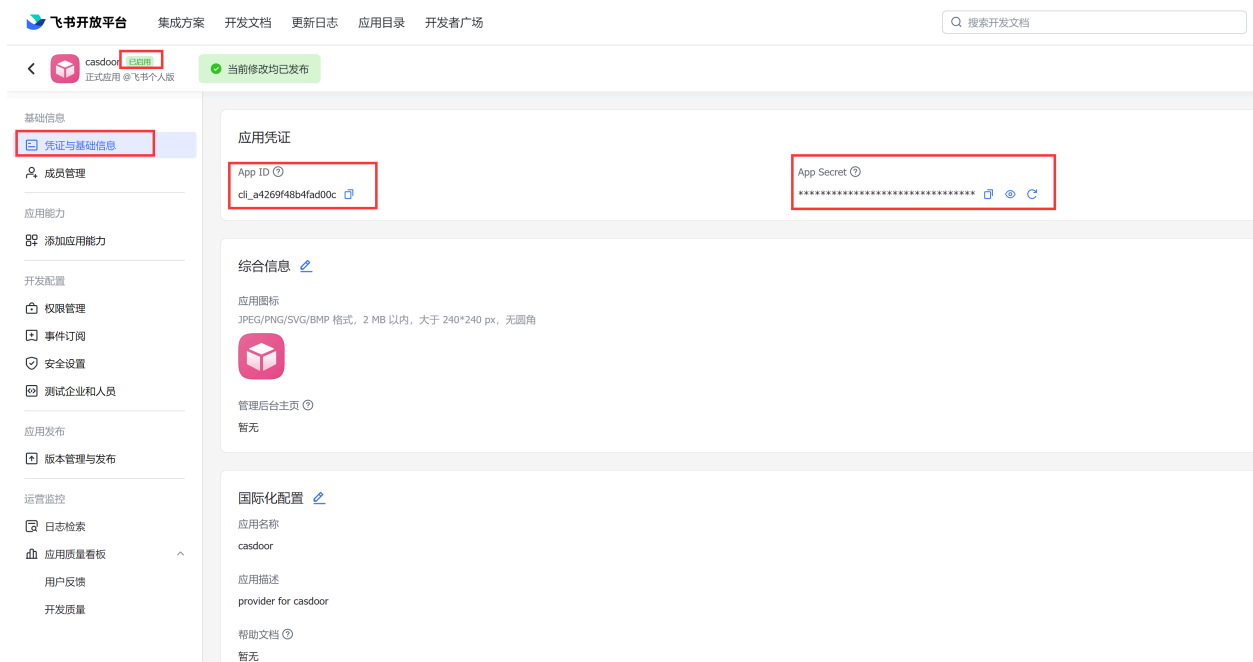
Lark

NOTE

This is an example of how to configure a Lark OAuth provider.

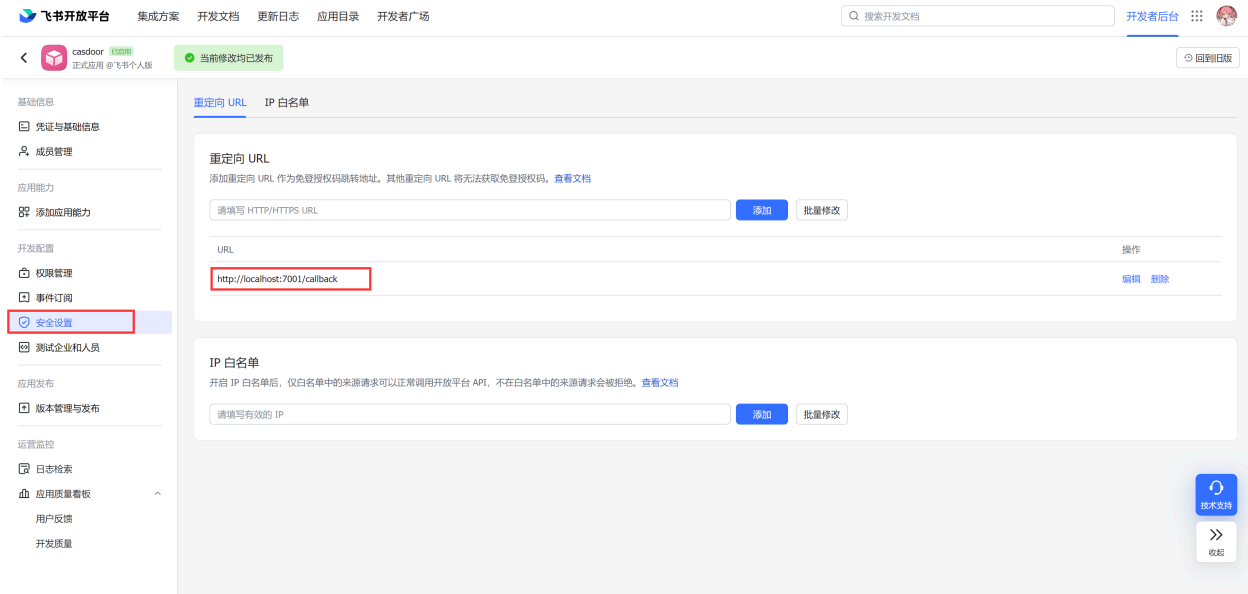
Step 1: Create a Lark application

First, you need to create a new application on the [Lark Open Platform](#) and enable it. You can find the `App ID` and `App Secret` in the basic information of your application.



The screenshot shows the Lark Open Platform console for a user named 'casdoor'. The left sidebar contains navigation options such as '基础信息', '成员管理', '应用能力', '开发配置', '应用发布', '运营监控', and '开发质量'. The main content area is divided into three sections: '应用凭证' (Application Credentials), '综合信息' (General Information), and '国际化配置' (Internationalization Configuration). In the '应用凭证' section, the 'App ID' is 'cli_a4269f48b4fad00c' and the 'App Secret' is a long string of asterisks. In the '综合信息' section, the application icon is a red cube, and the application name is 'casdoor'. In the '国际化配置' section, the application name is 'casdoor' and the application description is 'provider for casdoor'.

Next, add the redirect URL `<your-casdoor-domain>/callback` (e.g., `http://localhost:7001/callback`) in the security settings of your application.



Step 2: Create a Lark OAuth provider

Now create a Lark OAuth provider in Casdoor. Fill in the necessary information.

Name	Name in Lark
Category	Choose <code>OAuth</code>
Type	Choose <code>Lark</code>
Client ID	<code>App ID</code> obtained from Step 1
Client secret	<code>App Secret</code> obtained from Step 1



Now you can use Lark as the third-party service to complete authentication.

Email

Overview

Using Email for authentication

SendGrid

Using SendGrid as the SMTP server

Azure ACS

Using Azure ACS as the email provider

Brevo

Using Brevo as the SMTP server


MailHog

Using MailHog as the SMTP server


Overview

Adding an Email provider


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

Name  :

email provider

Display name  :

My Email


Category  :

Email


Type  :

Default


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

Username  :

no-reply@casbin.com

Password  :

Host  :

 smtp.qiye.aliyun.com

Port  :

465

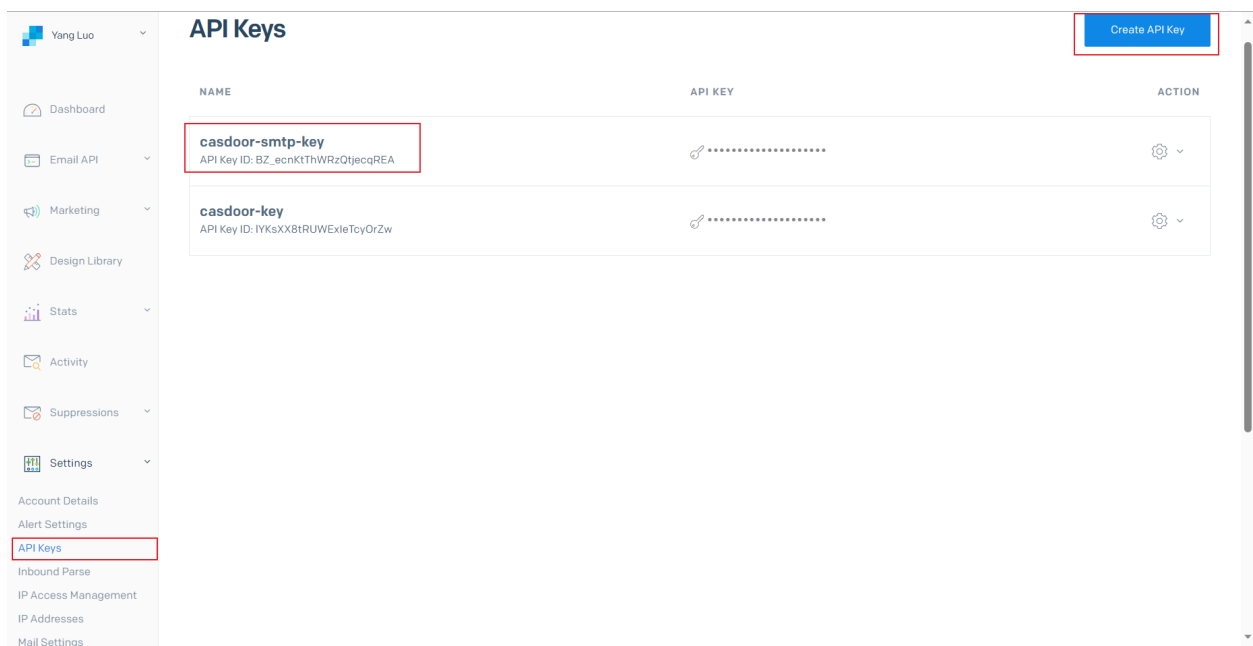
4. Customize the `Email Title` and `Email Content`, then save the changes.

SendGrid

In this guide, we will be using SendGrid as the SMTP server.

Step 1: Create API key for your SendGrid account

Expand the **Settings** from the left navigation bar, click on the **API Keys** option from this list. Here, you will see all of your API keys if you have generated any in the past. To generate a new one, you need to click on **Create API Key** and pay attention to the permissions.



The screenshot shows the SendGrid 'API Keys' management page. On the left is a navigation sidebar with 'API Keys' highlighted. The main content area has a 'Create API Key' button in the top right. Below it is a table with two API keys:

NAME	API KEY	ACTION
casdoor-smtp-key API Key ID: BZ_ecnKtThWRzQJecqREA	*****	Settings
casdoor-key API Key ID: IYksXX8TRUWExieTcyOrZw	*****	Settings


Step 2: Sender Verification

Refer to the document to verify your email sender, you can choose **Single Sender Verification** or **Domain Authentication**: [Sender Identity](#)

Step 3: Configure Casdoor email Provider

Now create an email provider in Casdoor. Fill in the required fields below:

Required fields	Remark
Username	Enter "apikey"
Password	Your SendGrid's API key
From Address	Your verified sender
Host	Enter "smtp.sendgrid.net"
Port	Default is 465

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

Click on the **Test SMTP Connection** button. If you see **provider: SMTP connected successfully**, it means that your Casdoor service can access the SendGrid service.

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

Azure ACS

In this guide, we will be using ACS as the Email Provider.

Step 1: Config ACS

Follow the documentation below, complete configuration.

- [Create and manage Email Communication Service](#)
- [Get a free Azure managed domain](#) or [Add a custom domain](#)
- [Connect domain](#)

Copy your **Endpoint** and **Private Key** for usage

Home > casdoor

casdoor | Keys ☆ ...
Communication Service

Search

Use access keys to authorize your API calls when you use the Communication Services SDKs. Store your access keys securely – for example, using Azure Key Vault – and don't share them. We recommend regenerating your access keys regularly. You are provided two access keys so that you can maintain connects using one key while regenerating the other.

Endpoint

Primary key

Regenerate primary key Show values

Key

Connection string

Secondary key

Regenerate secondary key Show values

Key

Connection string

Settings

- Keys
- Identities & User Access Tokens
- Push notifications
- Identity
- Properties
- Locks
- Telephony and SMS

Step 2: Configure Casdoor email Provider

Now create an email provider in Casdoor, fill in the necessary information. The relationship between the fields and Azure ACS is as follows:

i NOTE

From Address must be a verified email domain.

Name	Name in Azure ACS
From Address	
Secret key	Private Key
Host	Endpoint

Name ⓘ :

Display name ⓘ :

Organization ⓘ :

Category ⓘ :

Type ⓘ :

Secret key ⓘ :

From address ⓘ :

Host ⓘ :

Email title ⓘ :

Email content ⓘ :

Test Email ⓘ :

Provider URL ⓘ :

Brevo

In this guide, we will be using Brevo as the SMTP server.

Step 1: Request the activation of your Brevo SMTP account

Refer to the documentation to activate Brevo SMTP: [Send transactional emails using Brevo SMTP](#)

In my case, when I created a ticket to activate my smtp account, I got this reply:



Rafael Guimaraes

Last Response: 2 days ago

Hi there,

Thank you for reaching out!

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

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

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

- [Our complete library of help articles related to SMTP](#)
- [Troubleshooting common issues with SMTP](#)

The SMTP port listed by default, Port 587, will be used without a secure connection. If you want to use a secure connection (SSL or TLS), please use Port 465.

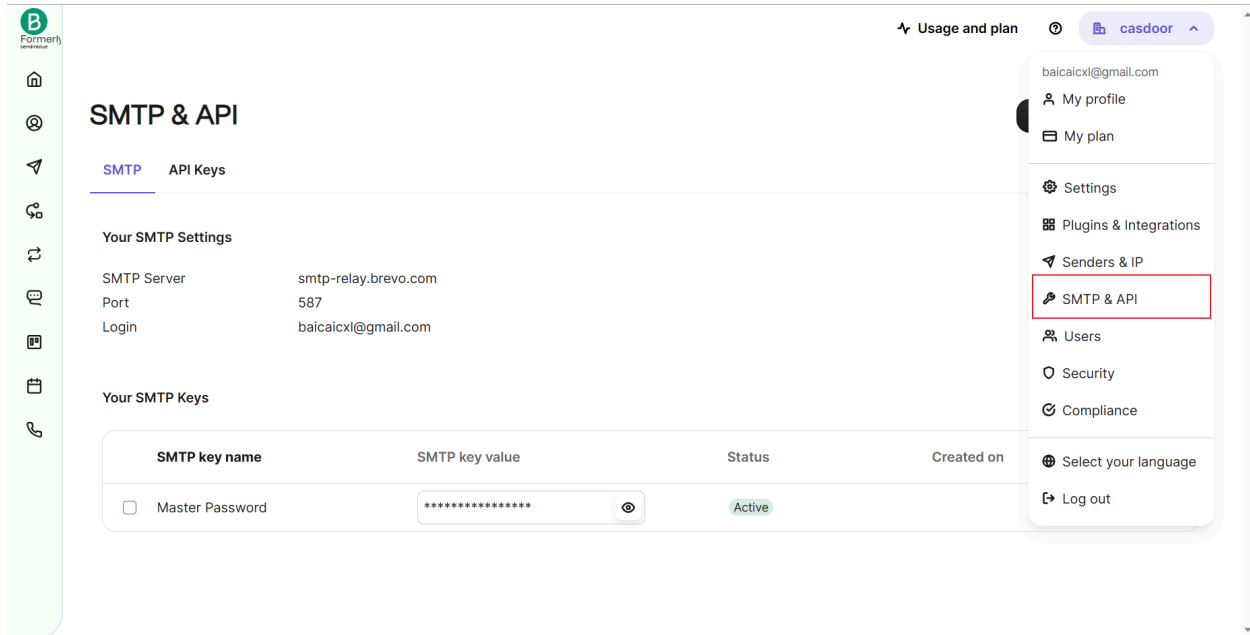
|

Please be sure to let me know if you have more questions or if you need any help.

Best regards,

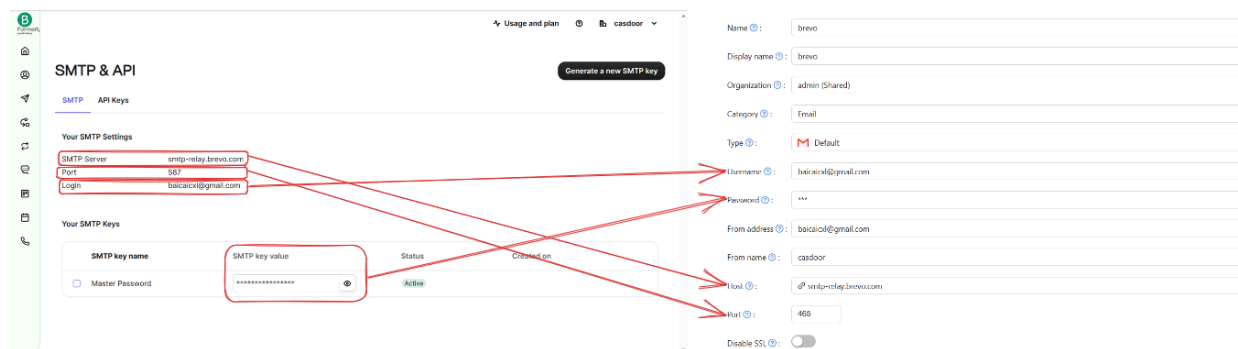
Step 2: Get Brevo SMTP configuration

In your Brevo dashboard, find SMTP&API, get `SMTP Server`, `Port`, `Login`, `SMTP key value`



Step 3: Configure Casdoor email Provider

Now create an email provider in Casdoor. Fill in the necessary information.



Click on the **Test SMTP Connection** button. If you see **provider: SMTP connected successfully**, it means that your Casdoor service can access the Brevo service.

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

MailHog

In this guide, we will be using MailHog as the SMTP server. [MailHog](#) is an email-testing tool that operates with a fake SMTP server.

Step 1: Deploy the MailHog service

The IP address for the MailHog service is `192.168.24.128`, and the SMTP service port is `1025`.

```
[HTTP] Binding to address: 0.0.0.0:8025
2023/07/13 03:06:43 Serving under http://0.0.0.0:8025/
Creating API v1 with WebPath:
Creating API v2 with WebPath:
[APIv1] KEEPALIVE /api/v1/events
[HTTP] Binding to address: 0.0.0.0:8025
Creating API v1 with WebPath:
Creating API v2 with WebPath:
2023/07/13 03:10:36 Using maildir message storage
2023/07/13 03:10:36 Maildir path is /tmp/mailhog641072855
2023/07/13 03:10:36 [SMTP] Binding to address: 0.0.0.0:1025
2023/07/13 03:10:36 Serving under http://0.0.0.0:8025/
[APIv2] GET /api/v2/jim
[APIv2] GET /api/v2/messages
```

Step 2: Create an email provider

Provide the necessary information and save the settings.

Category ? :	<input type="text" value="Email"/>
Type ? :	<input type="text" value="Default"/>
Username ? :	<input type="text"/>
Password ? :	<input type="password"/>
From address ? :	<input type="text" value="notification@casdoor.com"/>
From name ? :	<input type="text" value="Casdoor Notification"/>
Host ? :	<input type="text" value="192.168.24.128"/>
Port ? :	<input type="text" value="1025"/>
Disable SSL ? :	<input checked="" type="checkbox"/>
Email title ? :	<input type="text" value="Casdoor Verification Code (Test)"/>
Email content ? :	<input type="text" value="You have requested a verification code at <u>Casdoor</u> (Test). Here is your code: <u>%s</u>, please enter in 5 minutes."/>
Test Email ? :	<input type="text" value="admin@example.com"/> <input type="button" value="Test SMTP Connection"/> <input type="button" value="Send Testing Email"/>
Provider URL ? :	<input type="text" value="https://github.com/organizations/xxx/settings/applications/1234567"/>

Step 3: Send a test email

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

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

Name ⓘ : email_provider

Display name ⓘ : Email Provider

Organization ⓘ : admin (Shared)

Category ⓘ : Email

Type ⓘ : Default

Username ⓘ :

Password ⓘ :

From address ⓘ : notification@casdoor.com

From name ⓘ : Casdoor Notification

Host ⓘ : 192.168.24.128

Port ⓘ : 1025

Disable SSL ⓘ :

Email title ⓘ : Casdoor Verification Code (Test)

Email content ⓘ : You have requested a verification code at Casdoor (Test). Here is your code: %s, please enter in 5 minutes.

Test Email ⓘ : admin@example.com

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

provider:SMTP connected successfully

Email sent successfully

Test SMTP Connection

Send Testing Email

MailHog

Search

Connected

Inbox (4)

Delete all messages

Jim

Jim is a chaos monkey.
Find out more at GitHub.

Enable Jim

From: "Casdoor Notification" <notification@casdoor.com>

Subject: Casdoor Verification Code (Test)

To: admin@example.com

HTML Plain text Source

You have requested a verification code at Casdoor (Test). Here is your code: 123456, please enter in 5 minutes.

SMS

Overview

Using SMS for authentication

Twilio

Using Twilio as an SMS provider for Casdoor

Amazon SNS

Using Amazon SNS as an SMS provider for Casdoor

Azure ACS

Using ACS as an SMS provider for Casdoor

Alibaba Cloud

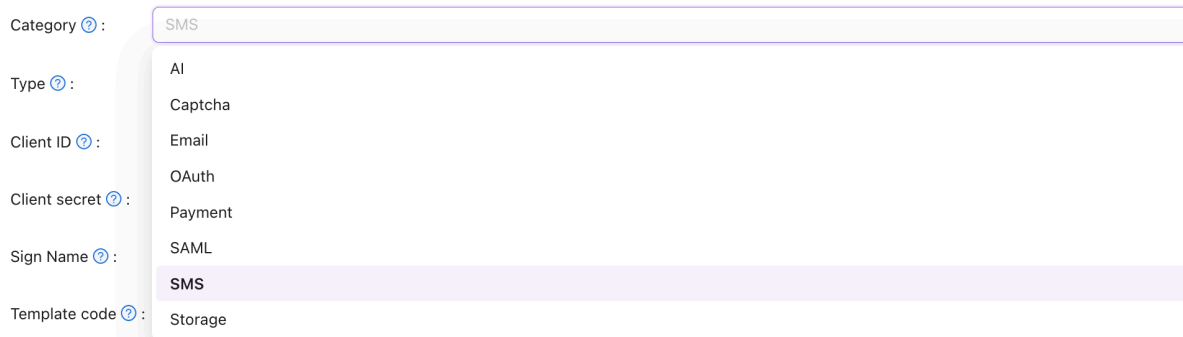
Using Alibaba Cloud as an SMS provider for Casdoor

Overview

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

Adding an SMS provider

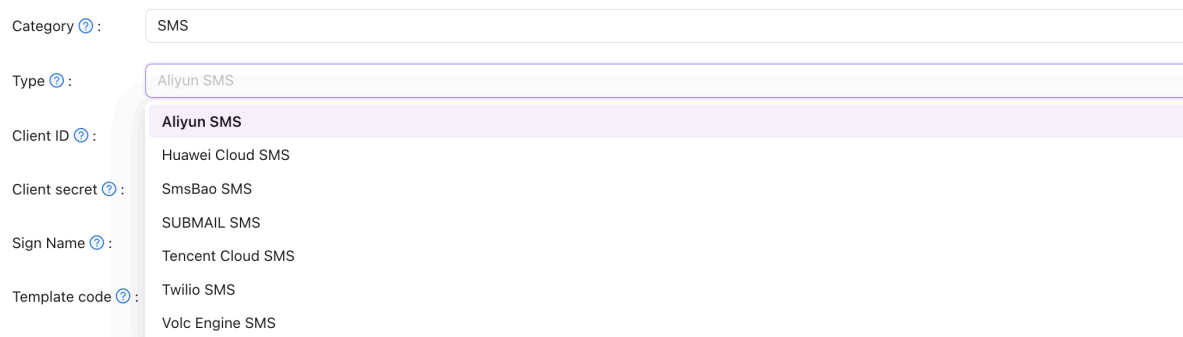
1. Click on `Add` to add a new provider.
2. Select `SMS` in the `Category` section.



The screenshot shows a dropdown menu for the 'Category' field. The menu is open, displaying a list of categories: AI, Captcha, Email, OAuth, Payment, SAML, SMS, and Storage. The 'SMS' option is highlighted with a light purple background. To the left of the dropdown, the labels for the fields are visible: Category, Type, Client ID, Client secret, Sign Name, and Template code.

Field	Value
Category	SMS
Type	AI
Type	Captcha
Client ID	Email
Client ID	OAuth
Client secret	Payment
Sign Name	SAML
Sign Name	SMS
Template code	Storage

3. Choose the type of your provider.



The screenshot shows a dropdown menu for the 'Type' field. The menu is open, displaying a list of provider types: Aliyun SMS, Huawei Cloud SMS, SmsBao SMS, SUBMAIL SMS, Tencent Cloud SMS, Twilio SMS, and Volc Engine SMS. The 'Aliyun SMS' option is highlighted with a light purple background. To the left of the dropdown, the labels for the fields are visible: Category, Type, Client ID, Client secret, Sign Name, and Template code.

Field	Value
Category	SMS
Type	Aliyun SMS
Type	Aliyun SMS
Type	Huawei Cloud SMS
Client ID	SmsBao SMS
Client secret	SUBMAIL SMS
Sign Name	Tencent Cloud SMS
Sign Name	Twilio SMS
Template code	Volc Engine SMS

4. Retrieve the necessary information from your SMS provider and fill out the corresponding fields.

Twilio

Fill in the necessary information in Casdoor

There are four required fields: `Client ID`, `Client secret`, `Sender number`, and `Template code`. The corresponding relationship to the Twilio account is as follows:

Name	Name in Twilio	Required
Client ID	Account SID	Required
Client secret	Auth Token	Required
Sender number	Twilio phone number	Required
Template code		Required

Twilio information

- Account SID, Auth Token, and Twilio phone number

Step 4: Invite and upgrade

Invite teammates
Invite developers to your Twilio account to start building! [Learn more about user access management](#)

Upgrade your account
Upgrade your account to send to any number, buy local more. [Learn more about trial account limitations](#)

Buttons: Invite teammates, Upgrade, < Back

Account Info

- Account SID**: AC06b73d65c8ee67ce8e448edcc64b6ec6
- Auth Token**: [Show](#)
⚠ Always store your token securely to protect your account. [Learn more](#)
- My Twilio phone number**: +12186751069

Helpful links

- [How does Twilio work?](#)
- Understand how to use Twilio in a 2
- [SMS Quickstart guides](#)
- Learn the basics of Twilio Messagir
- [Support help center](#)
- Troubleshoot common issues.

Configure Casdoor provider

You can configure the `template code` to suit your requirements, and then enter your phone number in `SMS Test` to test.

Name ⓘ: twilio

Display name ⓘ: twilio

Organization ⓘ: admin (Shared)

Category ⓘ: SMS

Type ⓘ: Twilio SMS

Client ID ⓘ: AC06b73d65c8ee67ce8e448edcc64b6ec6

Client secret ⓘ: ***

Sender number ⓘ: +12186751069

Template code ⓘ: get the message

SMS Test ⓘ: +1

Provider URL ⓘ:

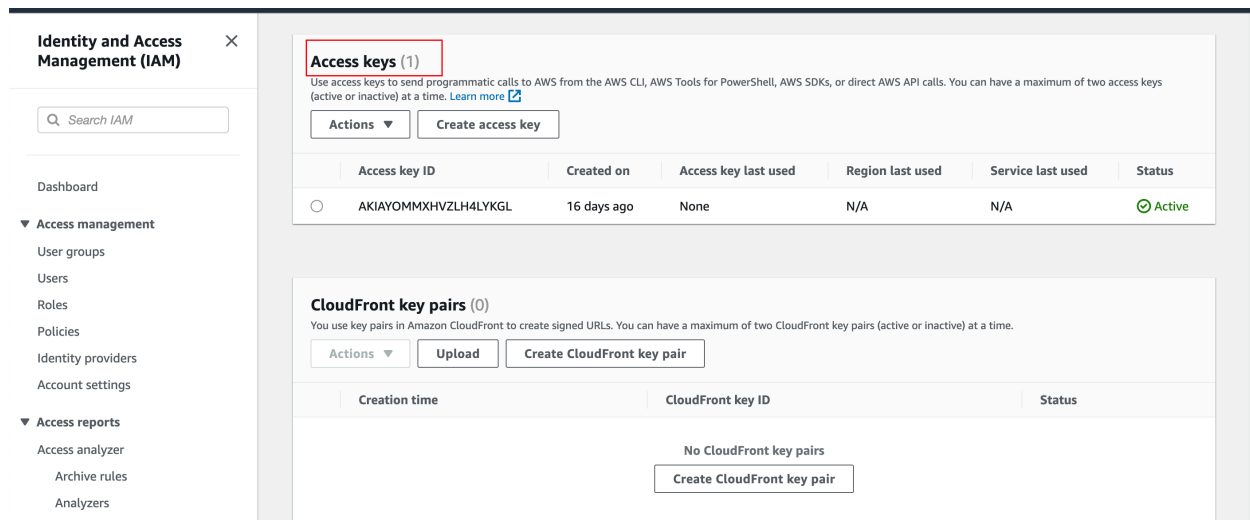
Amazon SNS

Obtaining the necessary information in Amazon

There are four required fields: `Access Key`, `Secret Access Key`, `Region`, and `Template code`. I will show you how to obtain this information from Amazon SNS.

- Access Key and Secret Access Key

In Identity and Access Management (IAM), you can create an `Access Key` and `Secret Access Key`.

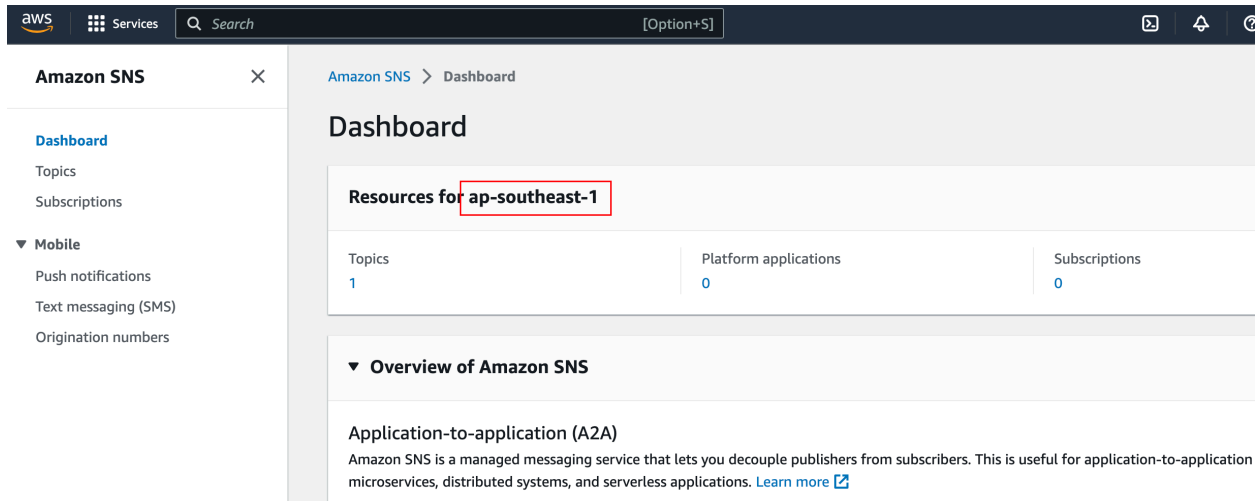


The screenshot shows the AWS IAM console interface. On the left is a navigation sidebar with 'Identity and Access Management (IAM)' selected. The main content area is divided into two sections. The top section, 'Access keys (1)', has a red box around the title. It includes a description, a 'Create access key' button, and a table with one entry. The bottom section, 'CloudFront key pairs (0)', includes a 'Create CloudFront key pair' button and a message indicating no key pairs are present.

Access key ID	Created on	Access key last used	Region last used	Service last used	Status
<input type="radio"/> AKIAYOMMXHVZLH4LYKGL	16 days ago	None	N/A	N/A	Active


- Region

The `Region` is related to the topic you created.



Configuring the Casdoor provider

The `Template code` is the message you want to send. Enter your phone number in the `SMS Test` to test.

Name ? :	amazon_sns
Display name ? :	amazon_sns
Organization ? :	admin (Shared)
Category ? :	SMS
Type ? :	 Amazon SNS
Access key ? :	AKIAYOMMXHVZACW5RFMX
Secret access key ? :	***
Region ? :	ap-southeast-1
Template code ? :	enter the message you want to send
SMS Test ? :	+1 <input type="text" value="Input your phone num..."/> <input type="button" value="Send Testing SMS"/>
Provider URL ? :	https://github.com/organizations/xxx/settings/applications/1234567

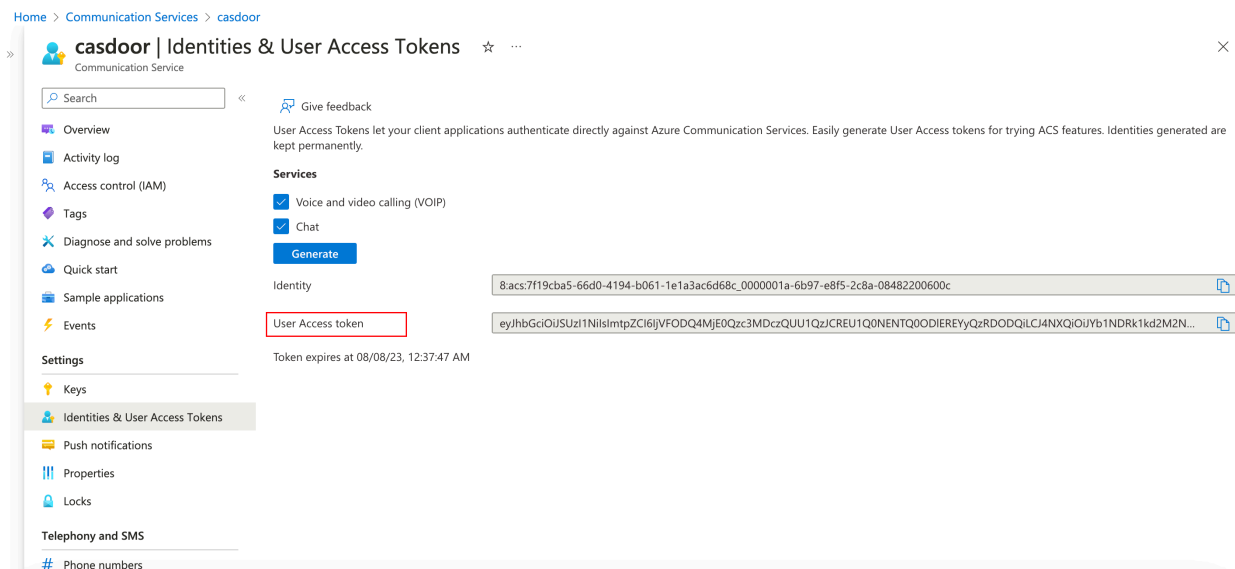
Azure ACS

Obtaining the necessary information in Azure

There are four required fields: `Client secret`, `Sender number`, `Template code`, and `Provider Url`. I will show you how to obtain this information from Azure ACS.

- `Client secret`

In Communication Service, you can create a User Access Token, which is the `Client secret` in Casdoor.



- `Sender number`

The `Sender number` is the phone number you create in Communication Service.

Communication Service

Search (Cmd+/) << + Get → Port 🗑 Release 👤 Give feedback

LOCKS

Tools

- Keys
- Identities & User Access Tokens
- Push notifications

Voice Calling - PSTN

- # Phone numbers
- Direct routing (Preview)

SMS

- Short Codes (Preview)

Monitoring

- Insights (preview)
- Metrics
- Diagnostic settings
- Logs

Number	Status	Cost (monthly)
✓ 1-833-920-3625 📄	✓ Active	\$2

- Provider Url

The `Provider Url` is the endpoint in Communication Service.

Communication Service

Search << → Move 🗑 Delete 👤 Give feedback

Overview

- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Quick start
- Sample applications
- Events

Settings

- Keys
- Identities & User Access Tokens

Essentials

Resource group (move) : [casdoor](#)

Status : Active

Location : Global

Subscription (move) : [免费试用](#)

Subscription ID : e054e27a-96e8-4cca-a1cf-32717dcd303c 📄

Tags (edit) : [Add tags](#)

Endpoint : <https://casdoor.unitedstates.communication.azure.com>

Data location : United States

Manage keys : [Click here to manage keys](#)

Build engaging communication experiences at scale

Azure Communication Services brings rich communication APIs to all of your apps across any device, on any platform, using the same reliable and secure infrastructure that powers Microsoft Teams. [Learn more](#)

Configure Casdoor provider

The `template code` is the message you want to send. Enter your phone number in `SMS Test` to test.

Name ? :	<input type="text" value="azure_acs"/>
Display name ? :	<input type="text" value="azure_acs"/>
Organization ? :	<input type="text" value="admin (Shared)"/>
Category ? :	<input type="text" value="SMS"/>
Type ? :	<input type="text" value="Azure ACS"/>
Client ID ? :	<input type="text"/>
Client secret ? :	<input type="text" value="***"/>
Sender number ? :	<input type="text" value="+18339203625"/>
Template code ? :	<input type="text" value="enter the message you want to send"/>
SMS Test ? :	<input type="text" value="+1"/> <input type="text" value="Input your phone num..."/> <input type="button" value="Send Testing SMS"/>
Provider URL ? :	<input type="text" value="https://casdoor.unitedstates.communication.azure.com"/>

Alibaba Cloud

Fill in the necessary information in Casdoor

There are four required fields: `Client ID`, `Client secret`, `Sign Name`, and `Template code`. The corresponding relationship with the Alibaba Cloud account is as follows:

Name	Name in Alibaba	is required
Client ID	AccessKey ID	required
Client secret	AccessKey Secret	required
Sign Name	Signature	required
Template code	Template code	required

Alibaba information

- AccessKey ID and AccessKey Secret

After logging into my Alibaba Cloud workbench, I click on "AccessKey" to create an ID and Secret.

工作台 | 搜索... | 费用 工单 ICP 备案 企业 支持 App 消息 购物车 帮助 简体 用户头像

短信服务概览 | 短信服务操作指南 产品动态

【有奖调研】阿里云短信服务易用性有奖调研 [点击进入](#)

用户状态/类型: 正常 / 个人用户

新手引导 | OpenAPI 开发者门户 | 开发者指南 | **AccessKey**

发送量数据 | 数据获取时间 22:33:52

用户监控信息

暂无预警信息, [前往设置](#)

快速操作入口

国内消息

已有短信签名0个 [添加签名](#)

已有短信模板0个 [添加模板](#)

已有群发助手任务0个 [提交发送任务](#)

国际/港澳台消息

快速上手短信服务, 从这里开始!
您已完成20%的学习进度, 继续努力吧。
[快速学习短信服务](#)

By creating an AccessKey, I obtain my AccessKey ID and AccessKey Secret:

工作台 | 搜索... | 费用 工单 ICP 备案 企业 支持 App 消息 购物车 帮助 简体 用户头像

安全信息管理

① AccessKey ID和AccessKey Secret是您访问阿里云API的密码, 具有该账户完全的权利, 请您妥善保管。

用户AccessKey [创建AccessKey](#)

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

- Signature

Go China | Go Globe | Analytics | Dashboard | Delivery Report | Messaging Logs | Bills | Resource Plan Usage | Short URL Statistics | System Configurations | General Settings

Join the group chat to try new features.

Alibaba Cloud SMS prohibits illegal content such as illegal financial marketing, gambling, fraud, obscenity, pornography, and violence in a message. If you send a message that contains illegal content, Alibaba Cloud will suspend your service and account according to Short Message Service Terms of Service. If your message is against the law, Alibaba Cloud will transfer evidences to the police, includi but not limited to the personal information authenticated in Alibaba Cloud.

Signatures | Message Templates | Mass Messaging

Generally, signatures are reviewed within 2 hours. Enterprise or institution signatures are reviewed within 2 business days. Recently, a review process took about 1 hour on average. More time may be required due to system upgrades or workload spikes. Business hours: Signatures are reviewed from 9:00 to 21:00 (UTC+8) every day, excluding statutory holidays. Thanks for your cooperation.

Create Signature | Select a review status | Search by signature | Export Records | Export

Signature	Ticket ID	Scenario	Review Status	Created At	Actions
<input type="checkbox"/> casdoor 测	20020363534	General	Approved	2023-07-26 13:15:49	Mass Messaging Operation Records Clone Modify Delete

- Template code

Short Message Service

- Overview
- Quick Start & Delivery Test
- Go China**
- Go Globe
- Analytics
- Dashboard
- Delivery Report
- Messaging Logs
- Bills
- Resource Plan Usage
- Short URL Statistics
- System Configurations
- General Settings
- Domestic SMS Settings

Create Dedicated DingTalk Group Chat

Scan the QR code to create your dedicated DingTalk group chat.
 You can use the group chat to submit and modify signatures and message templates, and check whether the signatures and message ten
 Join the group chat to try new features.

Alibaba Cloud SMS prohibits illegal content such as illegal financial marketing, gambling, fraud, obscenity, pornography, and violence in a message. If you send
 Alibaba Cloud will suspend your service and account according to Short Message Service Terms of Service. If your message is against the law, Alibaba Cloud w
 but not limited to the personal information authenticated in Alibaba Cloud.

Signatures **Message Templates** Mass Messaging

1. Generally, signatures are reviewed within 2 hours. Recently, a review process took about 1 hour on average. More time may be required due to system up
 hours: Signatures are reviewed from 9:00 to 21:00 (UTC+8) every day, excluding statutory holidays. Thanks for your cooperation.
 2. Message templates provided by Alibaba Cloud SMS do not require submission for approval. However, you are still charged for message delivery.

Create Message Template Select a template type Select a review status Search by message template name Tag Fil

Template Name	Tag	Ticket ID	Template Code	Template Type	Created At
casdoor 测		20020389144	SMS_462155126	Verification Code Message	2023-07-26 17:54:00

Configure Casdoor provider

Enter your phone number in the **SMS Test** field to test.

Name: alibaba

Display name: alibaba

Organization: admin (Shared)

Category: SMS

Type: Aliyun SMS

Client ID: LTAI5tFwxoA51CnSiQFyyPU5

Client secret: ***

Sign Name: casdoor

Template code: SMS_462155126

SMS Test: +86 Input your phone num... Send Testing SMS

Provider URL:

Notification

Overview

Add Notification providers to your application

Telegram

Using Telegram as a notification provider for Casdoor

Custom HTTP

Using Custom HTTP as a notification provider for Casdoor

Slack

Using Slack as a notification provider for Casdoor

Google Chat

Using Google Chat as a notification provider for Casdoor

Twitter

Using Twitter as a notification provider for Casdoor









Discord










Using Discord as a notification provider for Casdoor

Overview

Casdoor can be configured to send notification messages using various Notification providers.

Currently, Casdoor supports multiple Notification providers. Here are the providers that Casdoor supports:

Provider	Logo
Telegram	
Custom HTTP	
Slack	
Google Chat	
Twitter	
Discord	
Bark	
DingTalk	

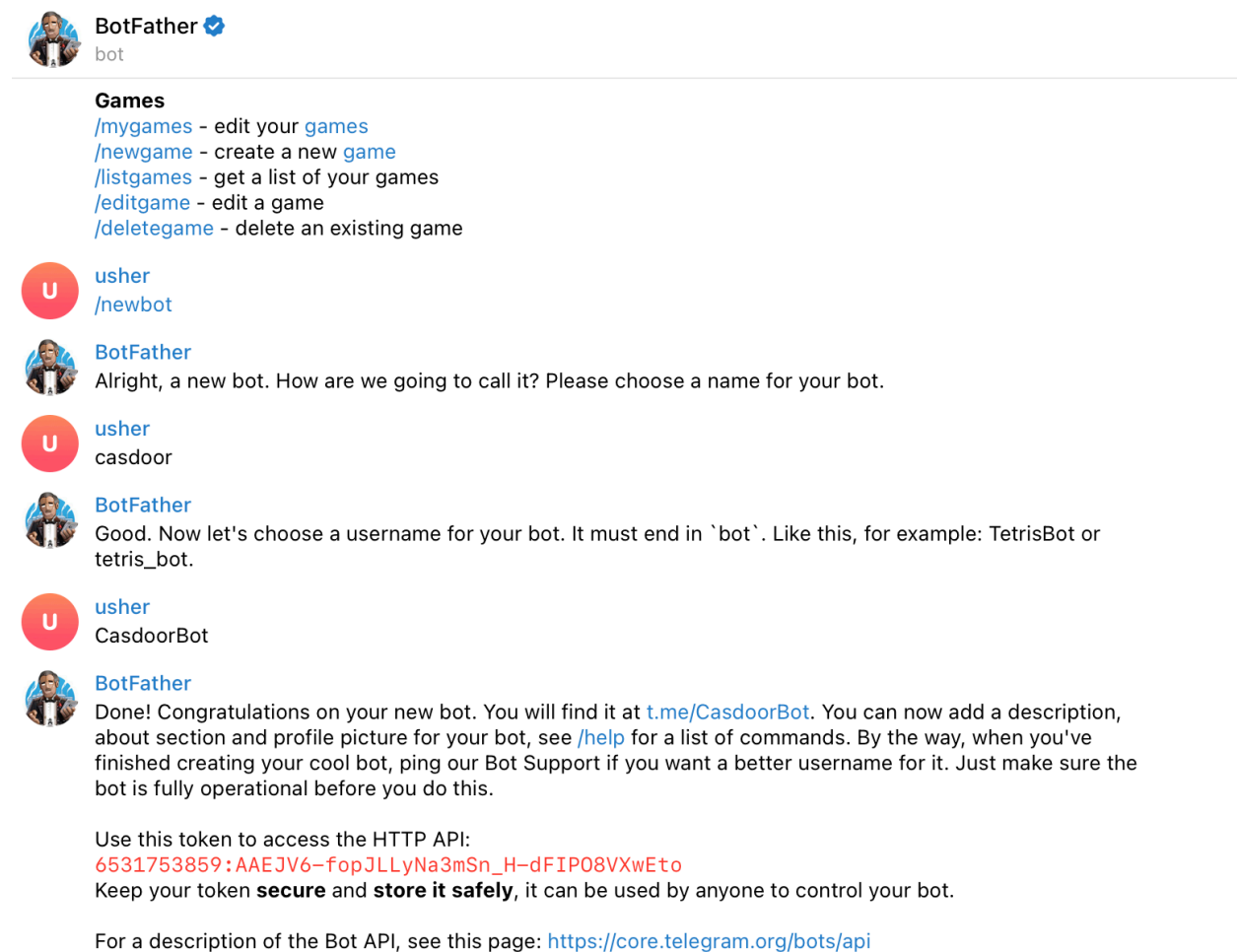
Provider	Logo
Lark	
Line	
Matrix	[matrix]
Microsoft Teams	
Pushbullet	
Pushover	
Reddit	
Rocket Chat	
Viber	
Webpush	

Telegram

Step 1: Get API Token

First, you need to create an account on [Telegram](#). After creating an account, you should contact the [BotFather](#), which is a bot used to create other bots.

To create your bot, use the command `/newbot`:



The screenshot shows a chat interface with BotFather. The chat history includes:

- BotFather** (bot):
 - Games**
 - `/mygames` - edit your [games](#)
 - `/newgame` - create a new [game](#)
 - `/listgames` - get a list of your games
 - `/editgame` - edit a game
 - `/deletegame` - delete an existing game
- usher**:
 - `/newbot`
- BotFather**:
 - Alright, a new bot. How are we going to call it? Please choose a name for your bot.
- usher**:
 - casdoor
- BotFather**:
 - Good. Now let's choose a username for your bot. It must end in ``bot``. Like this, for example: TetrisBot or tetris_bot.
- usher**:
 - CasdoorBot
- BotFather**:
 - Done! Congratulations on your new bot. You will find it at t.me/CasdoorBot. You can now add a description, about section and profile picture for your bot, see [/help](#) for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

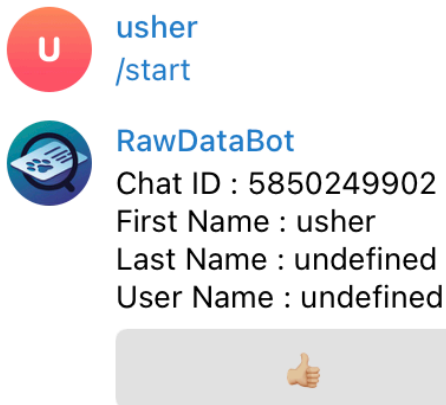
Use this token to access the HTTP API:
`6531753859:AAEJV6-fopJLLyNa3mSn_H-dFIP08VXwEto`
Keep your token **secure** and **store it safely**, it can be used by anyone to control your bot.

For a description of the Bot API, see this page: <https://core.telegram.org/bots/api>

Your bot should have two attributes: a `name` and a `username`. After creating the bot, you will receive an `API Token`.

Step 2: Get Chat ID

To find your chat ID, use [RawDataBot](#).



Step 3: Configure Casdoor Telegram Provider

There are three required fields: `App Key`, `Content`, and `Chat ID`. The relationship between the fields and Telegram is as follows:

Name	Name in Telegram
Secret key	API Token
Chat ID	Chat ID
Content	

Name ⓘ: telegram

Display name ⓘ: telegram

Organization ⓘ: admin (Shared)

Category ⓘ: Notification

Type ⓘ:  Telegram

Secret key ⓘ: ***

Content ⓘ: test

Chat ID ⓘ: 5850249902

[Send Testing Notification](#)

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

Custom HTTP

i NOTE

Casdoor supports Custom HTTP Notification Provider. You can use it to send messages to specific HTTP addresses.

Configure Casdoor Custom HTTP Provider

There are three required fields: `Method`, `Parameter name`, `Content`, and `Chat ID`.

Name	Description
Method	Select <code>GET</code> or <code>POST</code> method.
Parameter name	URL query parameter name or body parameter, depending on the <code>method</code> .
Content	The message you want to send.
Chat ID	Your HTTP address

Name ⓘ :	<input type="text" value="custom_http"/>
Display name ⓘ :	<input type="text" value="custon_http"/>
Organization ⓘ :	<input type="text" value="admin (Shared)"/>
Category ⓘ :	<input type="text" value="Notification"/>
Type ⓘ :	<input type="text" value="M Custom HTTP"/>
Method ⓘ :	<input type="text" value="POST"/>
Parameter ⓘ :	<input type="text" value="test"/>
Content ⓘ :	<input type="text" value="test"/>
Endpoint ⓘ :	<input type="text" value="http://localhost:12345"/> <input type="button" value="Send Testing Notification"/>
Provider URL ⓘ :	<input type="text" value="https://github.com/organizations/xxx/settings/applications/1234567"/>

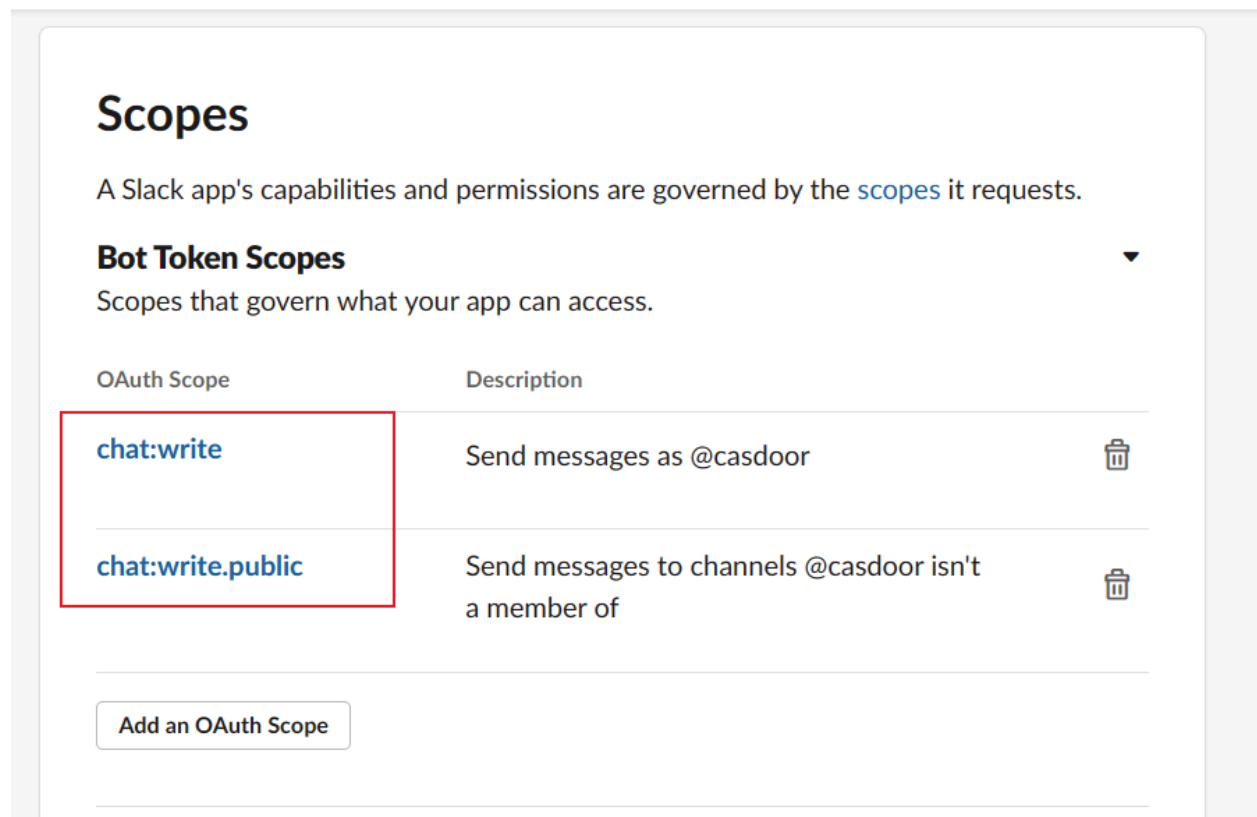
In my example, when I click `Send Notification Message`, I receive this request:

```
Listening on :12345...
Received a request:
Method: POST
URL: /
Body: test
```

Slack

Step 1: Config Slack App

First, you need to create an app on [Slack API](#). Give your bot/app the following OAuth permission scopes: `chat:write`, `chat:write.public`



The screenshot shows the 'Scopes' configuration page for a Slack app. The page title is 'Scopes' and it explains that a Slack app's capabilities and permissions are governed by the scopes it requests. Under the 'Bot Token Scopes' section, there is a table listing the configured OAuth scopes. Two scopes are listed: 'chat:write' and 'chat:write.public'. Both are highlighted with a red box. Below the table is a button labeled 'Add an OAuth Scope'.

OAuth Scope	Description
<code>chat:write</code>	Send messages as @casdoor
<code>chat:write.public</code>	Send messages to channels @casdoor isn't a member of

Step 2: Get Bot User OAuth Access Token and Channel ID

Copy your `Bot User OAuth Access Token` for usage below.

Features

- App Home
- Org Level Apps
- Incoming Webhooks
- Interactivity & Shortcuts
- Slash Commands
- Workflow Steps
- OAuth & Permissions**
- Event Subscriptions
- User ID Translation
- App Manifest NEW
- Beta Features

Submit to App Directory

- Review & Submit

Give feedback

Slack ♥

- Help
- Contact
- Policies
- Our Blog

⚠ At least one redirect URL needs to be set below before this app can be opted into token rotation

Opt In

OAuth Tokens for Your Workspace

These tokens were automatically generated when you installed the app to your team. You can use these to authenticate your app. [Learn more.](#)

User OAuth Token

xoxp-5865439759200-5827199080935-5865457291440-67810c12dfcae
Copy

Access Level: Workspace

Bot User OAuth Token

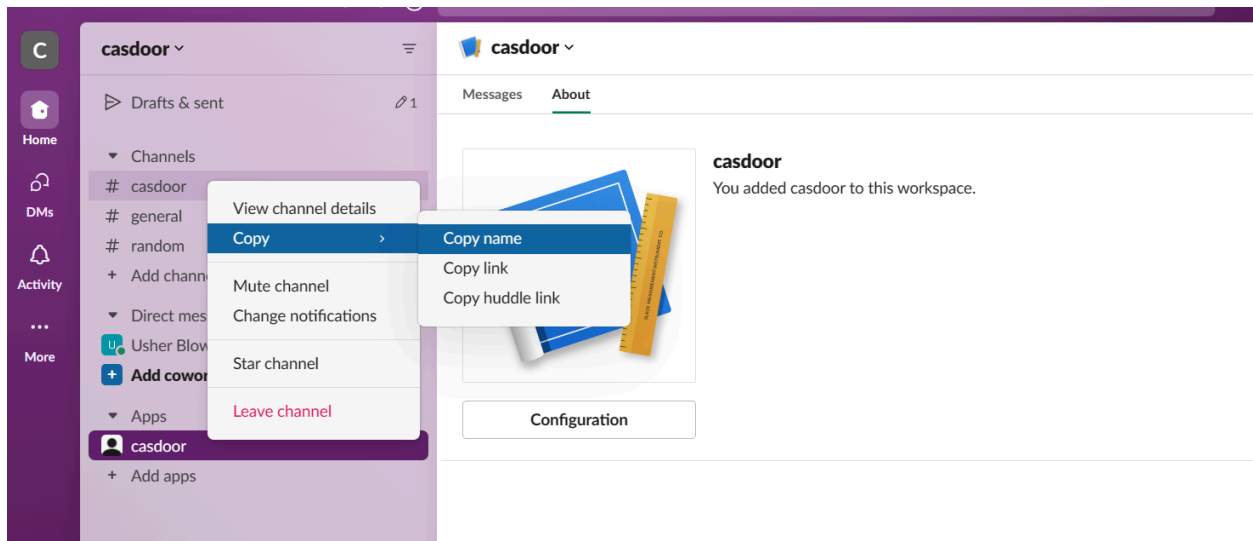
xoxb-5865439759200-5865457299776-2tbOAVTNpG7vmLOg7OFJS5t
Copy

Access Level: Workspace

Reinstall to Workspace

Redirect URLs

Copy the Channel ID of the channel you want to post a message to. You can grab the Channel ID by right clicking a channel and selecting copy name



Step 3: Configure Casdoor Slack Provider

There are three required fields: `App Key`, `Content`, and `Chat ID`. The relationship between the fields and Slack is as follows:


Name	Name in Slack
Secret key	Access Token
Chat ID	Channel ID
Content	

Name ⓘ : slack

Display name ⓘ : slack

Organization ⓘ : admin (Shared)

Category ⓘ : Notification

Type ⓘ :  Slack

Secret key ⓘ : ***

Content ⓘ : dont reply

Chat ID ⓘ : casdoor

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

Google Chat

Step 1: Get Application Default Credentials

In order to integrate notify with a Google Chat Application, `Application Credentials` must be supplied. For more information on Google Application credential JSON see: [How Application Default Credentials works](#)

The json will look like this:

```
{
  "type": "service_account",
  "project_id": "",
  "private_key_id": "",
  "private_key": "",
  "client_email": "",
  "client_id": "",
  "auth_uri": "",
  "token_uri": "",
  "auth_provider_x509_cert_url": "",
  "client_x509_cert_url": ""
}
```

Step 3: Configure Casdoor Google Chat Provider

Fill in the `Application credential` in the metadata.

Name ⓘ : google_chat

Display name ⓘ : google_chat

Organization ⓘ : admin (Shared)

Category ⓘ : Notification

Type ⓘ :  Google Chat

Metadata ⓘ :
{
 "type": "service_account",
 "project_id": "",
 "private_key_id": ""

Content ⓘ : test

Test Notification ⓘ : [Send Testing Notification](#)

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

Twitter

Step 1: Get the configuration items from twitter

First, sign up for a Twitter developer account, create a Twitter App within the developer portal refer to the documentation: [Authentication](#)

Copy your `API Key` and `API Secret`, `Access Token` and `Access Token Secret`

The screenshot shows the Twitter Developer Portal interface. On the left is a dark sidebar with the 'Developer Portal' logo and navigation links: Dashboard, Projects & Apps, Products (marked 'NEW'), and Account. The main content area is titled 'Keys and tokens' and has two tabs: 'Settings' and 'Keys and tokens'. Under 'Consumer Keys', there is a card for 'API Key and Secret' with a 'Reveal API Key hint' link and a 'Regenerate' button. Under 'Authentication Tokens', there are two cards: 'Bearer Token' (Generated September 3, 2023) with 'Revoke' and 'Regenerate' buttons, and 'Access Token and Secret' (Generated September 3, 2023, For @Allcompletenes, Created with Read Only permissions) with 'Revoke' and 'Regenerate' buttons. Red boxes highlight the 'API Key and Secret' field and the 'Access Token and Secret' card.

Step 2: Get Twitter ID

`Twitter ID` can't be obtained directly, you can get it from some third-party tools.

- [TweeterID](#)
- [Twiteridfinder](#)

Step 3: Configure Casdoor Twitter Provider

There are five required fields: `Client ID`, `Client secret`, `Client ID 2`, `Client secret 2` and `Chat ID`. The relationship between the fields and Twitter is as follows:

Name	Name in Twitter
Client ID	API Key
Client secret	API Secret
Client ID 2	Access Token
Client secret 2	Access Token Secret
Chat ID	Twitter ID

Name ⓘ :

Display name ⓘ :

Organization ⓘ :

Category ⓘ :

Type ⓘ :

Client ID ⓘ :

Client secret ⓘ :

Client ID 2 ⓘ :

Client secret 2 ⓘ :

Content ⓘ :

Chat ID ⓘ :

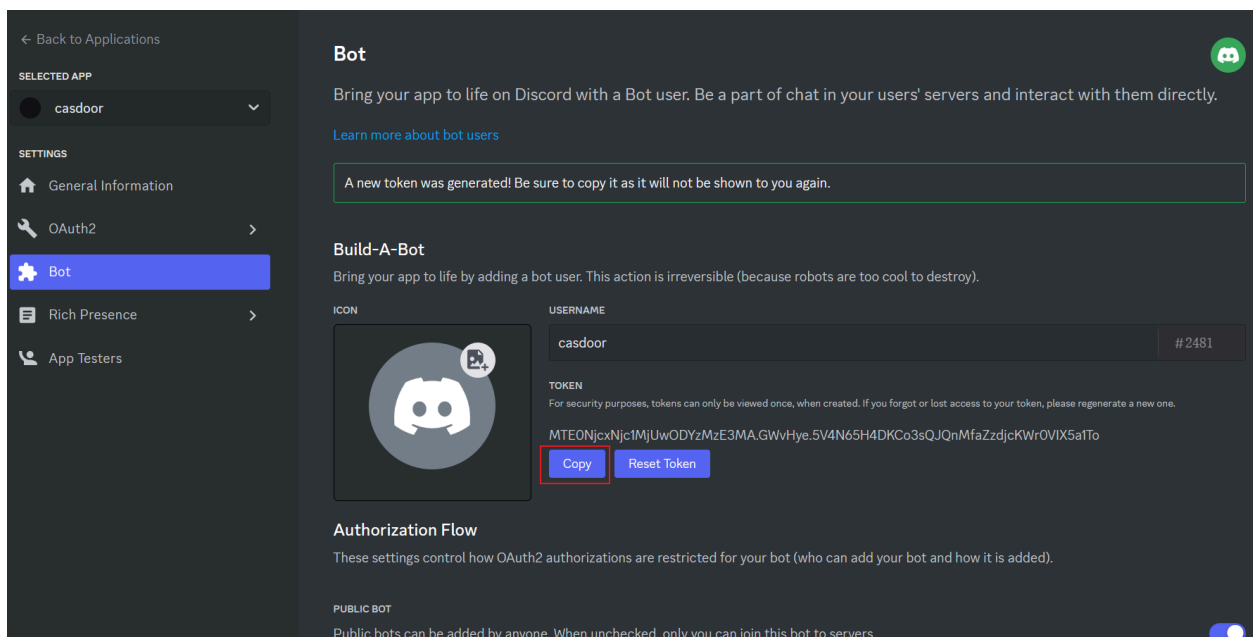
Provider URL ⓘ :

Discord

Step 1: Get Token from Discord

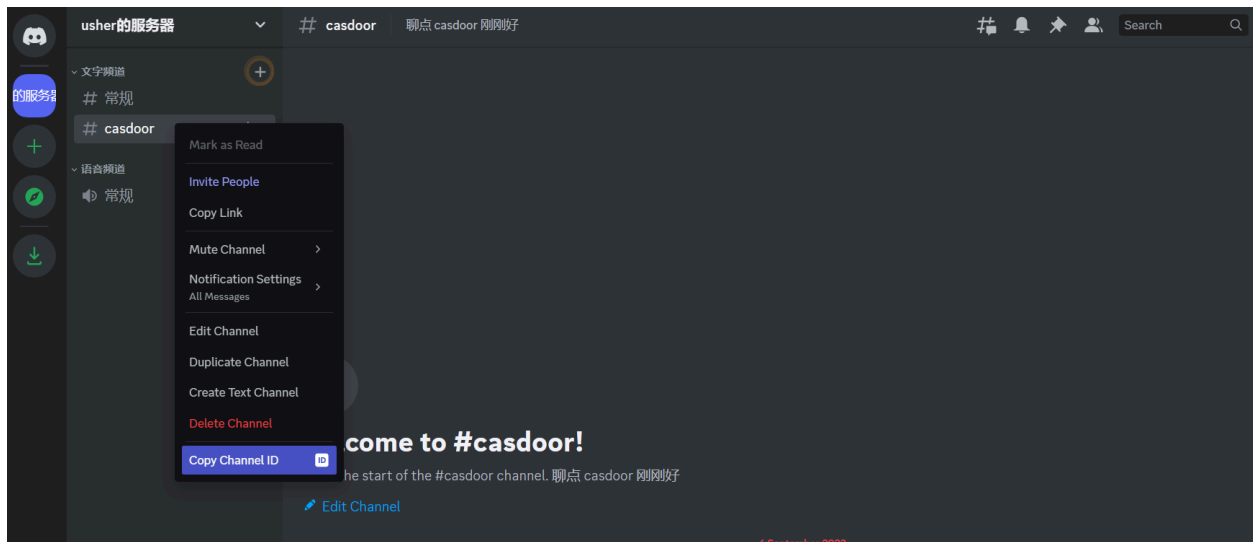
First, sign up for the Discord Developer portal, create a new application, navigate to the “Bot” tab to configure it.

Copy your Bot `token`



Step 2: Get Channel ID

Copy the Channel ID of the channel you want to post a message to. You can grab the Channel ID by right clicking a channel and selecting `Copy Channel ID`



Step 3: Configure Casdoor Discord Provider

There are three required fields: `App Key`, `Content`, and `Chat ID`. The relationship between the fields and Discord is as follows:

Name	Name in Slack
Secret key	Token
Chat ID	Channel ID
Content	

Name ⓘ:	<input type="text" value="discord"/>
Display name ⓘ:	<input type="text" value="discord"/>
Organization ⓘ:	<input type="text" value="admin (Shared)"/>
Category ⓘ:	<input type="text" value="Notification"/>
Type ⓘ:	<input type="text" value="Discord"/>
Secret key ⓘ:	<input type="text" value="***"/>
Content ⓘ:	<input type="text" value="test"/>
Chat ID ⓘ:	<input type="text" value="1146715329133821972"/> <input type="button" value="Send Testing Notification"/>
Provider URL ⓘ:	<input type="text" value="https://github.com/organizations/xxx/settings/applications/1234567"/>

Storage

Overview

Setting up a storage provider for uploading files in Casdoor

Local File System

Using the Local File System as a storage provider for Casdoor

Amazon S3

Using Amazon S3 as a storage provider for Casdoor

Azure Blob

Using Azure Blob as a storage provider for Casdoor

Google Cloud Storage

Using Google Cloud Storage as a storage provider for Casdoor

MinIO

Using MinIO as a storage provider for Casdoor

Alibaba Cloud OSS

Using Alibaba Cloud OSS as a storage provider for Casdoor

Tencent Cloud COS

Using Tencent Cloud COS as a storage provider for Casdoor

Synology NAS

Using Synology NAS as a storage provider for Casdoor

Overview

If you need to use file storage services, such as "avatar upload", you will need to set up a storage provider and apply it to your application in Casdoor.

Casdoor supports two types of storage: Local and Cloud. In this chapter, you will learn how to add a storage provider to use these services.

Item

- **Client ID:** A unique identifier provided by the cloud storage provider.
- **Client secret:** A secure value known only to Casdoor and the cloud storage service.
- **Endpoint:** The public URL or domain of the cloud storage service.
- **Endpoint (Intranet):** The internal or private URL or domain of the cloud storage service.
- **Path prefix:** Path prefix for the file location.

! INFO

The default `Path prefix` is `"/"`. For example, when the `Path prefix` is empty, a default file path would be:

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

You can fill it with `"abcd/xxxx"`, and then you can store your avatar in:

```
https://cdn.casbin.com/abcd/xxxx/casdoor/avatar.png
```

- **Bucket:** A container used for storing files and data.
- **Domain:** The custom domain name of the CDN for your cloud storage.
- **Region ID:** An identifier used to specify the data center region where the cloud storage service is located.

Local

We support uploading files to the local system.

Cloud

We support **AWS S3**, **Azure Blob Storage**, **MinIO**, **Alibaba Cloud OSS**, **Tencent Cloud COS**, and we are constantly adding more Cloud storage services.

Local File System

! INFO

The Local File System provider will store your uploaded files in the Casdoor `files` folder.

For example, when your Casdoor is located in the `/home/user/casdoor` directory, the uploaded files will be stored in the `/home/user/casdoor/files` folder.

Configure the Casdoor provider

Name ? :	<input type="text" value="localfile"/>
Display name ? :	<input type="text" value="localfile"/>
Organization ? :	<input type="text" value="admin (Shared)"/>
Category ? :	<input type="text" value="Storage"/>
Type ? :	<input type="text" value="Local File System"/>
Path prefix ? :	<input type="text"/>
Domain ? :	<input type="text" value="http://localhost:8000"/>
Provider URL ? :	<input type="text" value="🔗"/>

The `Path prefix` is the prefix of the location path for your files. You can fill it in

as needed. In the following example, you can see the difference with or without the prefix.

With prefix

Path prefix [?](#) :

 casdoor >  files >  images >  resource >  built-in >  admin >  withPrefix.png

Without prefix

Path prefix [?](#) :

 casdoor >  files >  resource >  built-in >  admin >  withoutPrefix.png

Amazon S3

 NOTE

This is an example of Amazon S3.

Create security credentials

Follow the document: [Managing access keys](#) to create and save your `access key` and `secret access key` in the Amazon console.

Configure your bucket

In your bucket permissions options, uncheck the "block" option and save the changes.

Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

Block public access to buckets and objects granted through *new* access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

Block public access to buckets and objects granted through *any* access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

Block public access to buckets and objects granted through *new* public bucket or access point policies

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

Block public and cross-account access to buckets and objects through *any* public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Cancel

Save changes


Edit the object ownership and check ACLs enabled.

Object Ownership

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

ACLs disabled (recommended)
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.



ACLs enabled
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

 We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.

Object Ownership

Bucket owner preferred
If new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.

Object writer
The object writer remains the object owner.

 If you want to enforce object ownership for new objects only, your bucket policy must specify that the bucket-owner-full-control canned ACL is required for object uploads. [Learn more](#) 

Cancel

Save changes

Configure Casdoor

Name	Name in Amazon	Is Required
Client ID	Access key	Required
Client secret	Secret access key	Required
Endpoint	Endpoint	Required
Endpoint (intranet)	VPC endpoint	

Name	Name in Amazon	Is Required
Bucket	Bucket name	Required
Path prefix		
Domain	CloudFront domain	
Region ID	AWS region	Required

Fill in the necessary information, including the `Client ID` and `Client Secret` obtained from the `access key` and `secret access key` in the previous step. You can refer to this documentation for information on the formatting of the `endpoint`:

[Website endpoints](#)

Name ? :	awss3
Display name ? :	awss3
Organization ? :	admin (Shared)
Category ? :	Storage
Type ? :	AWS S3
Client ID ? :	AKIAYOMMXHVZC5CHBPNR
Client secret ? :	***
Endpoint ? :	http://kininaru.s3-website.ap-northeast-1.amazonaws.com
Endpoint (Intranet) ? :	
Bucket ? :	kininaru
Path prefix ? :	
Domain ? :	
Region ID ? :	ap-northeast-1
Provider URL ? :	🔗

(Optional) Use VPC

You can refer to the official documentation for configuration: [Access AWS services through AWS PrivateLink](#)

(Optional) Use CloudFront distribution

Follow the document to configure CloudFront: [Configure CloudFront](#)

In the domain field, enter your distribution domain name.

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

Bucket  : kininaru

Path prefix  :

Domain  : https://d20zlk9foisfk0.cloudfront.net

Region ID  : ap-northeast-1

Provider URL  : 

Azure Blob

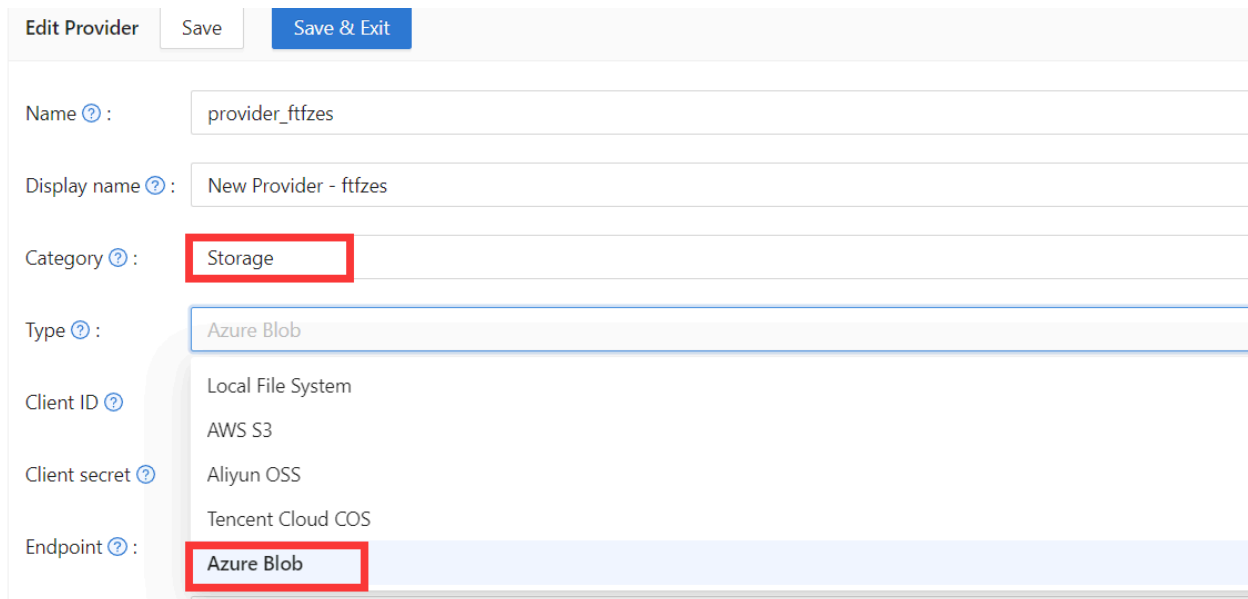
NOTE

This is an example of Azure Blob.

- You must have an [Azure storage](#) account.

Step 1: Select Azure Blob

Select Azure Blob as the storage type.



The screenshot shows the 'Edit Provider' configuration page in Casdoor. At the top, there are three buttons: 'Edit Provider', 'Save', and 'Save & Exit'. Below the buttons, the configuration fields are as follows:

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

The 'Storage' category and 'Azure Blob' type are highlighted with red boxes. The 'Endpoint' field also has a red box around the 'Azure Blob' option.

Step 2: Fill in the necessary information in Casdoor

There are four required fields: `Client ID`, `Client secret`, `Endpoint`, and `Bucket`. The corresponding relationship to the Azure Blob account is as follows:

Field Name	Azure Name	Required
Client ID	AccountName	Required
Client secret	AccountKey	Required
Endpoint	ContainerUrl	Required
Endpoint (intranet)	PrivateEndpoint	
Bucket	ContainerName	Required
Path prefix		
Domain	DomainName	

- AccountName

The `AccountName` is your AccountName.

- AccountKey

The `AccountKey` is your key to access the Azure API.

(i) NOTE

You can obtain your account key from the Azure Portal under the "Access Keys" section on the left-hand pane of your storage account.

casbin | Access keys ☆ ...
Storage account

Search << Set rotation reminder Refresh Give feedback

Storage browser
Storage Mover

Data storage

Containers
File shares
Queues
Tables

Security + networking

Networking
Azure CDN
Access keys
Shared access signature
Encryption
Microsoft Defender for Cloud

Data management

Access keys authenticate your applications' requests to this storage account. Keep your keys in a secure location like Azure Key Vault, and replace them often with new keys. The two keys allow you to replace one while still using the other.

Remember to update the keys with any Azure resources and apps that use this storage account.
[Learn more about managing storage account access keys](#)

Storage account name
casbin

key1 Rotate key
Last rotated: 2023/7/22 (0 days ago)
Key
..... Show

Connection string
..... Show

key2 Rotate key
Last rotated: 2023/7/22 (0 days ago)
Key
..... Show

Connection string

- ContainerUrl

You can obtain the ContainerUrl from your container properties.

default | Properties

Container



Refresh Give feedback

- Overview
- Diagnose and solve problems
- Access Control (IAM)

Settings

- Shared access tokens
- Access policy
- Properties
- Metadata

NAME

default

URL

https://casbin.blob.core.windows.net/default

LAST MODIFIED

7/22/2023, 5:18:03 PM

ETAG

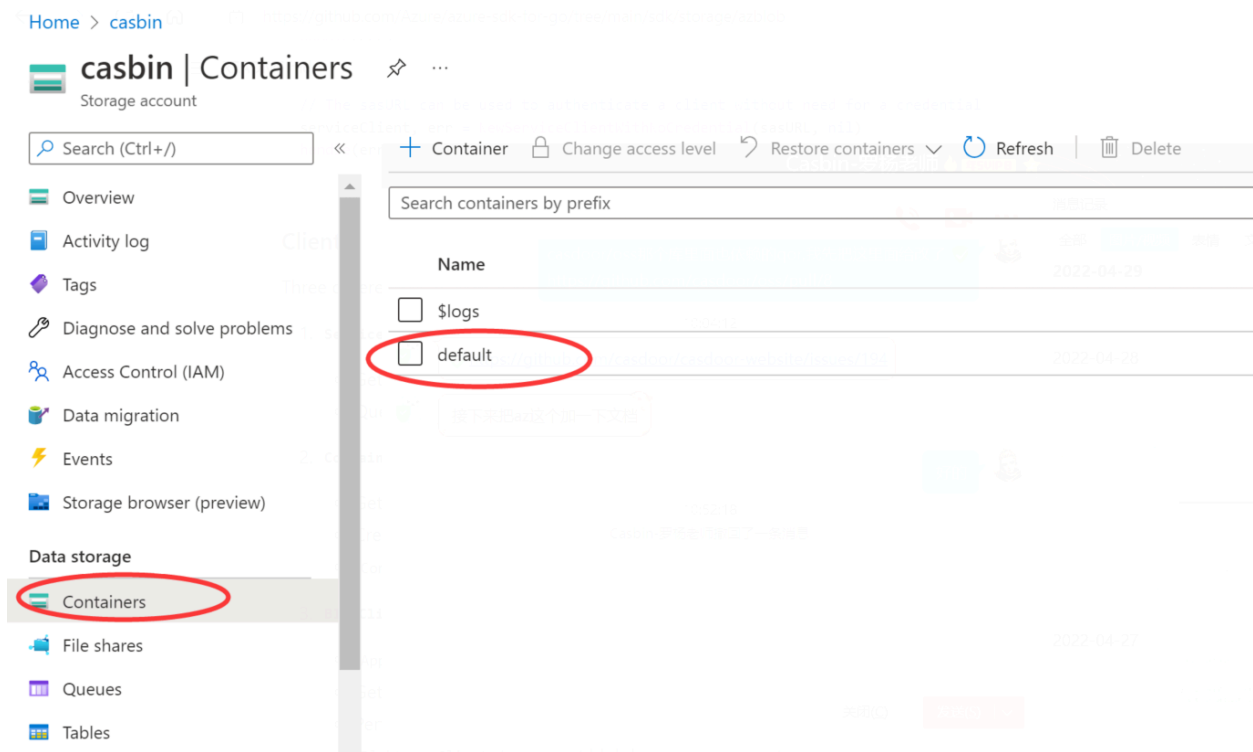
0x8DB8A948D644055

- (Optional) PrivateEndpoint

Azure Private Endpoint is a feature that allows connecting Azure services to Azure Virtual Network (VNet) private subnets. You can refer to the official Azure documentation for configuration: [private endpoint config](#)

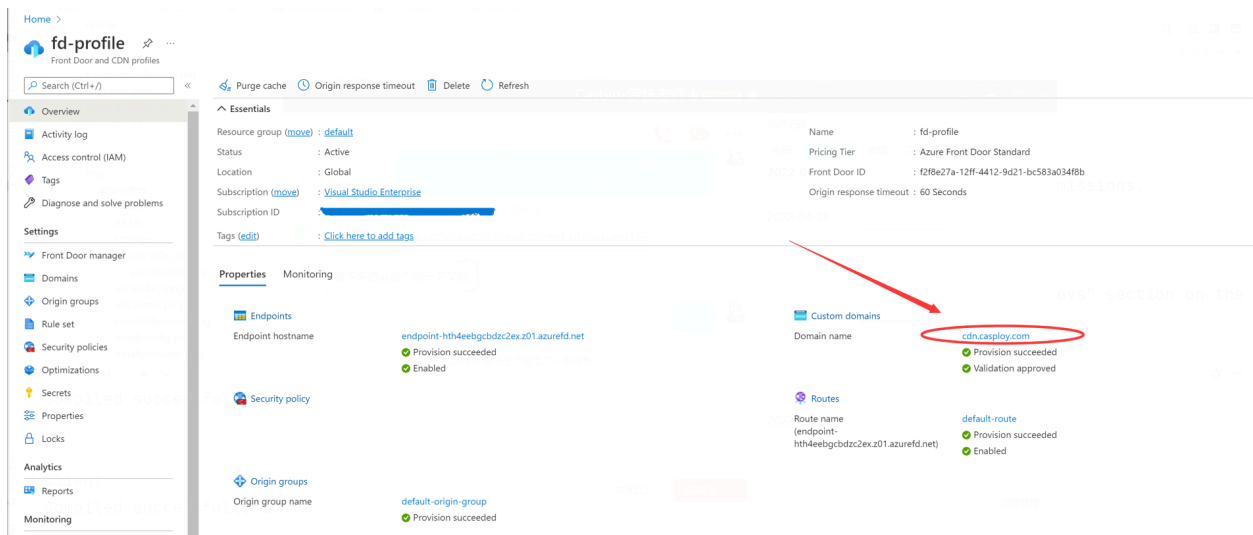
- ContainerName

In this example, a default container called 'default' is created.



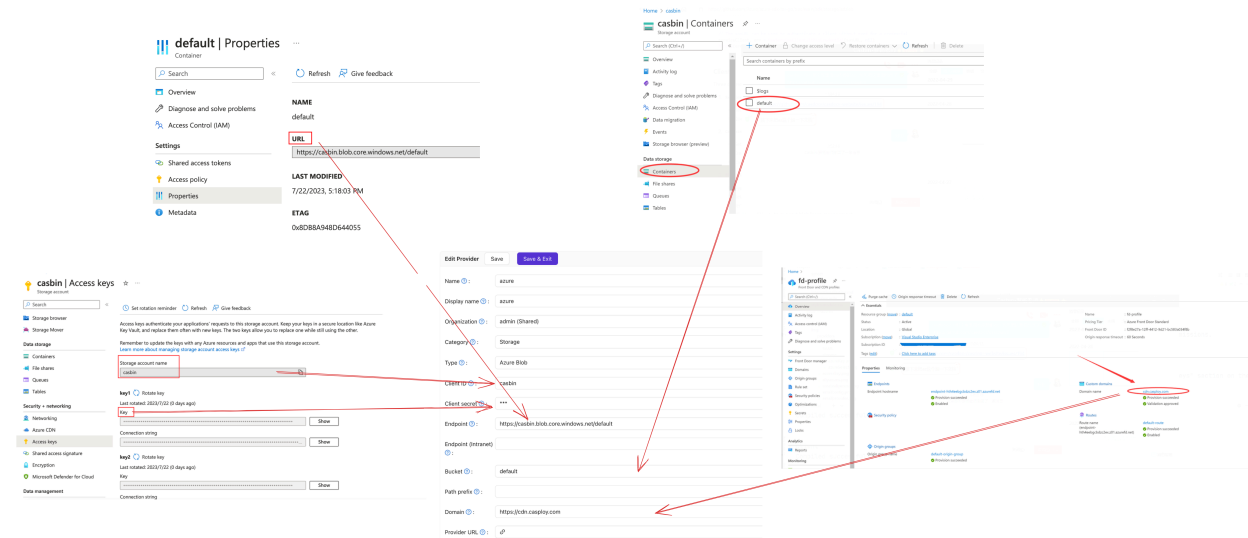
- (Optional) DomainName

The custom domain name in your Azure CDN.



Step 3: Save your configuration

The final result is as follows:



Now you can use Azure Blob Storage services in your application.

Google Cloud Storage

i NOTE

This is an example of Google Cloud Storage.

Create security credentials

Follow the document: [Cloud Storage Authentication](#) to create a [service account](#) with the correct [IAM permissions](#) to access the bucket in the GCP console.

Configure Casdoor

Name	Name in Google	Is Required
Service Account JSON	Service Account Key	Required
Endpoint	Endpoint	
Bucket	Bucket name	Required

Name ⓘ:

Display name ⓘ:

Organization ⓘ:

Category ⓘ:

Type ⓘ:

Service account JSON ⓘ:

Endpoint ⓘ:

Bucket ⓘ:

Path prefix ⓘ:

Provider URL ⓘ:

MinIO

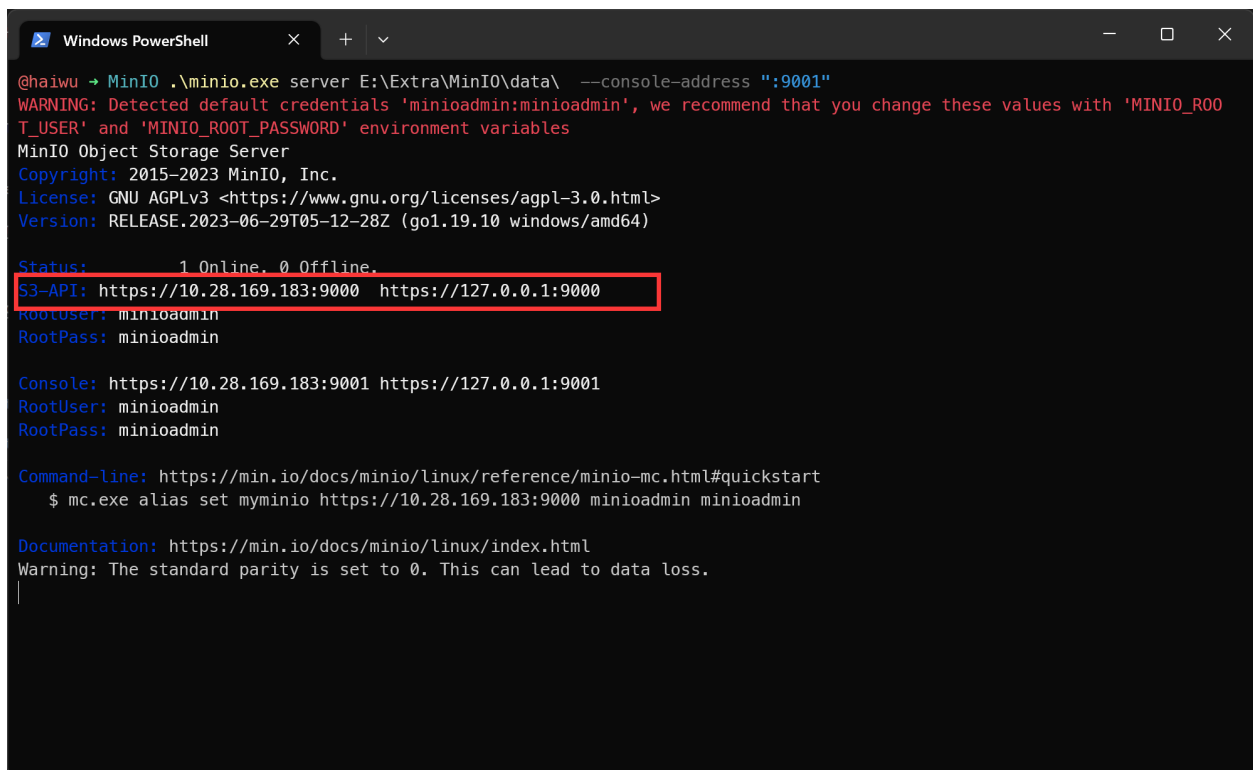
NOTE

This is an example of how to configure a MinIO provider.

MinIO is a high-performance object storage service that is API compatible with Amazon S3 cloud storage service.

Step 1: Deploy the MinIO service

First, deploy the MinIO service with TLS enabled. You can obtain the `API address` from the console.



```
Windows PowerShell
@haiwu → MinIO .\minio.exe server E:\Extra\MinIO\data\ --console-address ":9001"
WARNING: Detected default credentials 'minioadmin:minioadmin', we recommend that you change these values with 'MINIO_ROOT_USER' and 'MINIO_ROOT_PASSWORD' environment variables
MinIO Object Storage Server
Copyright: 2015-2023 MinIO, Inc.
License: GNU AGPLV3 <https://www.gnu.org/licenses/agpl-3.0.html>
Version: RELEASE.2023-06-29T05-12-28Z (go1.19.10 windows/amd64)

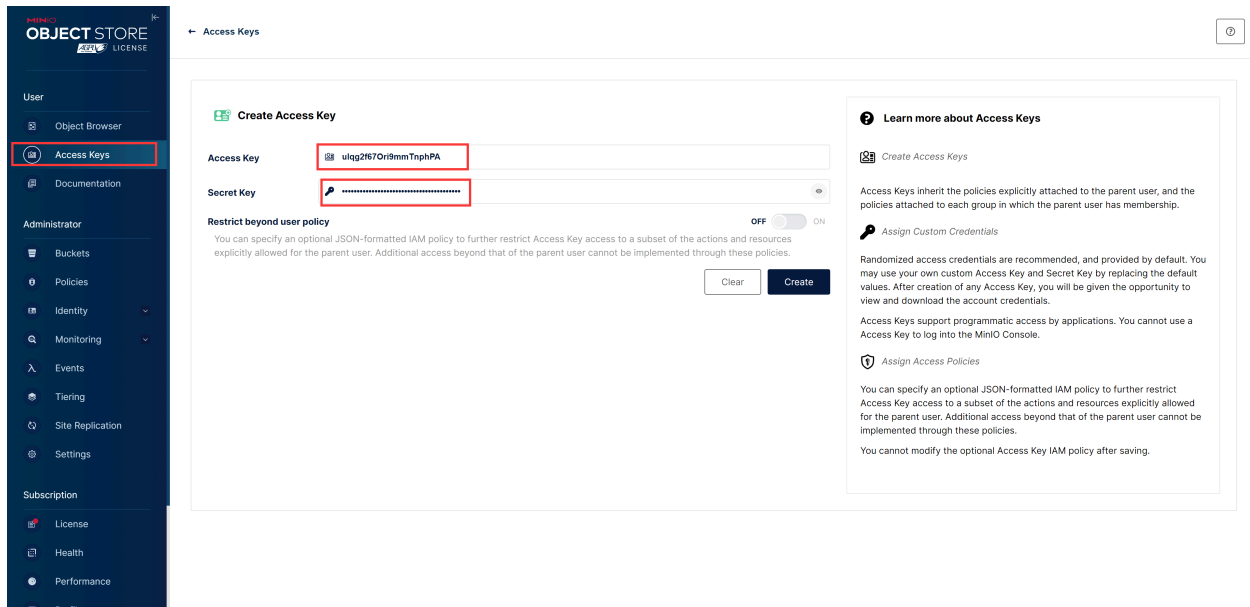
Status:          1 Online, 0 Offline.
S3-API: https://10.28.169.183:9000 https://127.0.0.1:9000
RootUser: minioadmin
RootPass: minioadmin

Console: https://10.28.169.183:9001 https://127.0.0.1:9001
RootUser: minioadmin
RootPass: minioadmin

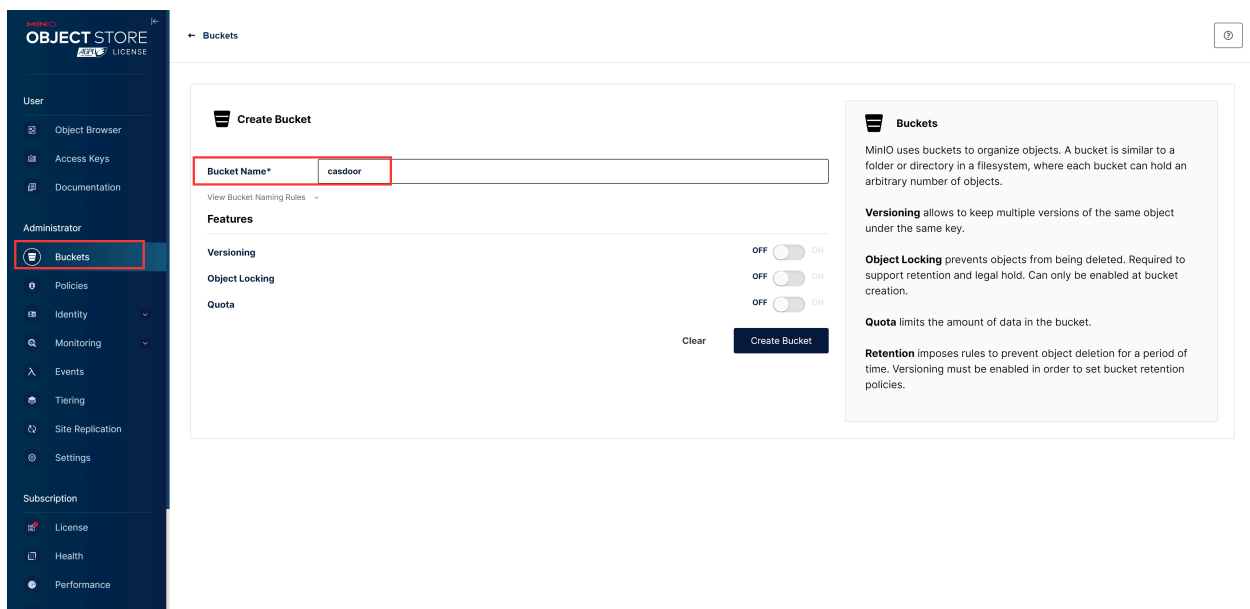
Command-line: https://min.io/docs/minio/linux/reference/minio-mc.html#quickstart
$ mc.exe alias set myminio https://10.28.169.183:9000 minioadmin minioadmin

Documentation: https://min.io/docs/minio/linux/index.html
Warning: The standard parity is set to 0. This can lead to data loss.
```

Second, create the `Access Key` and `Secret key`.



Third, create the **Bucket**.



Step 2: Create a MinIO provider in Casdoor

Now create a MinIO provider in Casdoor. Fill in the necessary information.

Name	Name in MinIO
Category	choose Storage
Type	choose MinIO
Client ID	Access Key obtained from Step 1
Client secret	Secret Key obtained from Step 1
Endpoint	API address obtained from Step 1
Bucket	Bucket obtained from Step 1

Edit Provider Save Save & Exit

Name: minio

Display name: minio

Organization: admin (Shared)

Category: Storage

Type: MinIO

Client ID: ulqg2t67Orl9mmTnphPA

Client secret: ***

Endpoint: http://10.28.169.183:9000

Bucket: casdoor

Path prefix:

Provider URL:

```

minio - MinIO: Linux server [X]@MinIOData console address: 0.0.0.0:9000
WARNING: Detecting insecure credentials 'minioadmin:minioadmin' we recommend that you change these values with 'MINIO_ROOT_USER' and 'MINIO_ROOT_PASSWORD' environment variables
MinIO Object Storage Server
Copyright 2015-2023 MinIO, Inc.
License: GPL v3.0 http://www.gnu.org/licenses/gpl-3.0.html
Server: RELEASE.2023-06-29T04-12-38Z (pid.16.18 w/indon/amd64)
Build: 0.0.0-0-gdf11ee
API: https://10.28.169.183:9000 https://127.0.0.1:9000
MinIO root path: /minio/data
MinIO root url: minioadmin
MinIO console: https://10.28.169.183:9001 https://127.0.0.1:9001
MinIO user: minioadmin
MinIO pass: minioadmin
MinIO logs: https://minio/docs/minio/linux/index.html#quickstart
& mc.exe & mc s3c myminio https://10.28.169.183:9000 minioadmin minioadmin
Documentation: https://minio.io/docs/minio/linux/index.html
Warning: The standard parity is set to 0. This can lead to data loss.

```

Step 3: Use MinIO storage service in your application

Now you can use the MinIO storage service in your application.

Alibaba Cloud OSS

NOTE

This is an example of Alibaba Cloud OSS.

The AccessKey is your key to access Alibaba Cloud API with full account permissions.

To create an AccessKey, follow the instructions in the [Alibaba Cloud workbench](#).

Next, create the OSS service:

创建 Bucket ? 创建存储空间 ×

! 注意: Bucket 创建成功后, 您所选择的 **存储类型、区域、存储冗余类型** 不支持变更。

Bucket 名称	<input type="text" value="mycasdoor"/> 9/63 
地域	<input type="text" value="华北2 (北京)"/> 
相同区域内的产品内网可以互通; 订购后不支持更换区域, 请谨慎选择。	
Endpoint	oss-cn-beijing.aliyuncs.com

Fill in the necessary information in Casdoor and save:

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

You can now use Alibaba Cloud cloud storage services in your application.

Tencent Cloud COS

i NOTE

This is an example of Tencent Cloud COS.

Fill in the necessary information in Casdoor

There are five required fields: Client ID, Client secret, Endpoint, Bucket, and Region ID. The corresponding relationship to the Tencent Cloud COS account is as follows:

Name	Name in Tencent	Required
Client ID	SecretId	Yes
Client secret	SecretKey	Yes
Endpoint	Endpoint	Yes
Bucket	BucketName	Yes
Path prefix		
Domain	CDNDomain	
Region ID	Region	Yes

Tencent Cloud COS information

- SecretId and SecretKey

The screenshot shows the Tencent Cloud API Key Management console. The left sidebar contains navigation options like '访问管理' (Access Management) and 'API密钥管理' (API Key Management). The main content area displays a table of API keys. One key is highlighted with its details:

APPID	密钥	创建时间	最近访问时间	状态
1319606438	SecretId: AKIDdAlMuNrJn8GHl6mLl6NSWbheNr7MVeic SecretKey: *****显示	2023-07-22 19:01:...	2023-07-22 22:09	已启用

- Endpoint, BucketName, and Region

The screenshot shows the Tencent Cloud COS console for bucket 'casdoor-1319606438'. The left sidebar contains navigation options like '返回桶列表' (Return to Bucket List) and '用量概览' (Usage Overview). The main content area displays usage statistics and bucket details:

用量概览 (Usage Overview):

- 对象数量: 4个 (Object Count: 4)
- 存储量: 0 B (Storage: 0 B)
- 本月总流量: 3.72 KB (Monthly Total Traffic: 3.72 KB)
- 本月总请求数: 126次 (Monthly Total Requests: 126)

基本信息 (Basic Information):

- 存储桶名称: casdoor-1319606438 (存储桶不支持改名)
- 所属地域: 广州 (中国) (ap-guangzhou)
- 创建时间: 2023-07-22 18:57:50
- 访问权限: 私有读写

域名信息 (Domain Information):

- 访问域名: https://casdoor-1319606438.cos.ap-guangzhou.myqcloud.com
- 自定义CDN加速域名: 0条
- 自定义源站域名: 0条
- 全球加速域名: 未开启
- 静态网站域名: 未开启

- (Optional) CDNDomain

You can refer to the official documentation for configuration: [Config CDN](#)

Configure Casdoor provider

The image shows two screenshots from the Tencent Cloud console. The top screenshot displays the 'API Key Management' page, where a new API key has been created. The key ID is AKID5A8AUAJn8GH6mL8NSWbhcH7MwUc and the secret key is SecretKey*****. The bottom screenshot shows the 'Bucket List' page for the bucket 'casdoor-1319606438'. The bucket is located in the 'ap-guangzhou' region and is currently empty. Red arrows connect the configuration details in the screenshots to the corresponding fields in the configuration form on the left.

Configuration Form Fields:

- Name: tencentcos
- Display name: tencentcos
- Organization: admin (Shared)
- Category: Storage
- Type: Tencent Cloud COS
- Client ID: AKID5A8AUAJn8GH6mL8NSWbhcH7MwUc
- Client secret: ****
- Endpoint: casdoor-1319606438.cos.ap-guangzhou.myqcloud.com
- Bucket: casdoor-1319606438
- Path prefix:
- Domain:
- Region ID: ap-guangzhou
- Provider URL: ip

API Key Management Table:

APPID	密钥	创建时间	最近使用时间	状态
casdoor-1319606438	SecretKey AKID5A8AUAJn8GH6mL8NSWbhcH7MwUc SecretKey*****	2023-07-22 19:01:...	2023-07-22 22:09	已启用

Bucket List Table:

对象数量	存储空间	本月总流量	本月总请求数
4	0 B	3.72 GB	126 次

Synology NAS

i NOTE

This is an example of Synology NAS.

Fill in the necessary information in Casdoor

There are five required fields: `Client ID`, `Client secret` and `Endpoint`. The corresponding relationship to the Synology NAS account is as follows:

Name	Name in Tencent	Required
Client ID	SecretId	Yes
Client secret	SecretKey	Yes
Endpoint	Endpoint	Yes
Bucket		
Path prefix		
Domain		
Region ID		

Configure Casdoor provider

The image shows a composite screenshot illustrating the configuration of a Synology provider in Casdoor. On the left, the Casdoor 'Edit Provider' form is visible, with red arrows pointing to the following fields:

- Name: provider_synology
- Display name: New Provider - synology
- Organization: built-in
- Category: Storage
- Type: Synology
- Client ID: password_is_slackonglong
- Client secret: ***
- Endpoint: http://169.254.0.2:5000
- Bucket: (empty)
- Path prefix: (empty)
- Provider URL: https://github.com/organizations/you/settings/applications/1234567

On the right, a 'Personal' account management window shows the 'AccessID' field containing 'password_is_slackonglong' and the 'AccessKey' field containing '*****'. A red arrow points from the 'AccessID' field to the 'Client ID' field in the Casdoor form.

Below the Casdoor form, the Synology Assistant interface is shown, displaying a table of discovered devices:

服务器名称	IP 地址	IP 状态	状态	网络物理地址	版本	型号	序列号	WOL
synologyNAS	169.254.0.2	Manual	已就绪	00:11:32:2C:A6:03	6.1.7-15284	DS3617xs	A80DN02468	--

Below the table, the endpoint is specified: **Endpoint: http://169.254.0.2:5000 (default http/https port of Synology is 5000/5001)**. A red arrow points from this endpoint to the 'Endpoint' field in the Casdoor form.

You can refer to the official documentation for configuration: [link](#)

SAML

Overview

Using identities from external identity providers that support SAML 2.0

Custom

Configure your SAML Custom Provider

Keycloak

Using Keycloak to authenticate users







Alibaba Cloud IDaaS

Using Alibaba Cloud IDaaS to authenticate users

Overview

Casdoor can be configured to support user login to the UI using identities from external identity providers that support SAML 2.0. In this configuration, Casdoor never stores any credentials for the users.

Now, Casdoor supports multiple SAML application providers. Icons of the providers will be displayed on the login page after being added to Casdoor. Here are the providers that Casdoor supports:

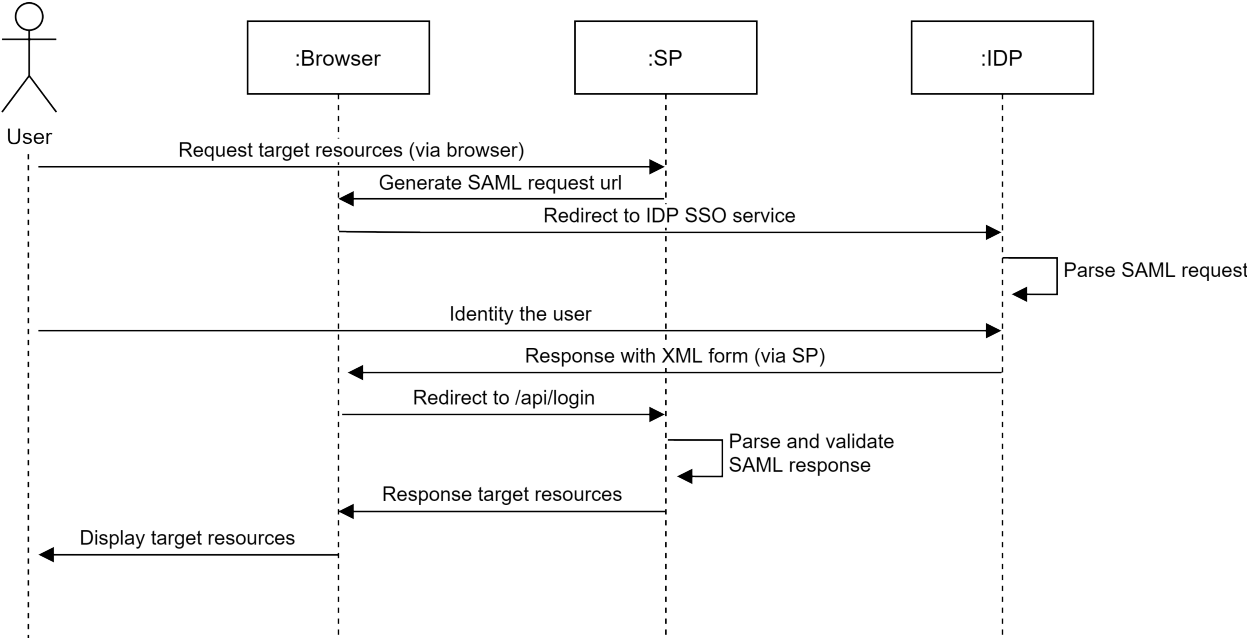
Alibaba Cloud IDaaS	Keycloak	Custom
		
		

Terms

- Identity Provider (IDP) - The service that stores the identity database and provides identity and authentication services to Casdoor.
- Service Provider (SP) - The service that provides resources to the end user, in this case, the Casdoor deployment.
- Assertion Consumer Service (ACS) - The consumer of SAML assertions generated by the Identity Provider.

How SAML integration works

When using SAML SSO, users log into Casdoor via the identity provider without ever passing credentials to Casdoor. The progress is shown in the following diagram.



Custom

Casdoor supports configuring SAML Custom Provider, and you can use Casdoor as a Service Provider (SP) to connect any Identity Provider (IDP) that support SAML 2.0 protocol.

Step1. Get the metadata of IDP

First, you need to obtain the metadata of IDP, which is a XML document used to describe the configuration information of the services provided by IDP. It needs to include information such as `EntityID`, `SSO Endpoint`, etc.

Some IDPs, such as Keycloak, require SP information to provide metadata. You can refer to the document [Keycloak](#).

You can use oktadev to test the SAML Custom Provider, here is the [metadata](#).


Step2. Configure SAML Custom Provider


After obtain the metadata of IDP, create a SAML Custom Provider and fill the necessary information.


Field	Description
Category	Choose <code>SAML</code>
Type	Choose <code>Custom</code>


Field	Description
Favicon.URL	The URL of the IDP logo
Metadata	The metadata of IDP



Then click **Parse** button, and fields **Endpoint**, **IdP**, **Issuer URL**, **SP ACS URL** and **SP Entity ID** will be automatically parsed.


Name  : saml_custom_provider_oktadev


Display name  : saml_custom_provider_oktadev





Organization  : admin (Shared)


Category  : SAML



Type  :  Custom


User mapping  :


ID  :


Username  :
 First Name
 Display name  :
 Last Name
 Email  :
 Email
 Avatar  :


Favicon  :


URL  :  https://cdn.casbin.org/img/social_okta.png

Preview: 


Client ID  :


Client secret  :


Sign request  :


Metadata  :


```
<EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata" entityID="urn:example:idp">
  <IDPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <KeyDescriptor use="signing">
      <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
```



Endpoint  : http://idp.oktadev.com

IdP  : MIIDPCCAiQCCQDydJgOlzsqbzANBgkqhkiG9w0BAQUFADBGMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcmspYTEWMBQGAG1UEBxMNU2FuiEZyYW5jaXNjbzEQMA4GA1UEChMHStr


Issuer URL  : urn:example:idp

































SP ACS URL  : /api/acs

SP Entity ID  : /api/acs

Provider URL  : 

Finally, add the SAML Custom Provider to **Providers** of the appliciton.

Providers 

Name	Category	Type	Can sign up	Can sign in	Can unlink	Prompted	Rule	Action
provider_storage_minio_s3	Storage		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		  
provider_oauth_bark	OAuth		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		  
provider_email_0q	Email		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		  
provider_web3_metamask	Web3		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		  
provider_google_oauth	OAuth		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	One Tap	  
provider_web3_onboard	Web3		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		  
saml_custom_provider_oktadev	SAML		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		  
saml_custom_provider_keycloak	SAML		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		  

Keycloak

The JBoss [Keycloak](#) system is a widely used and open-source identity management system that supports integration with applications via SAML and OpenID Connect. It can also operate as an identity broker between other providers such as LDAP or other SAML providers and applications that support SAML or OpenID Connect.

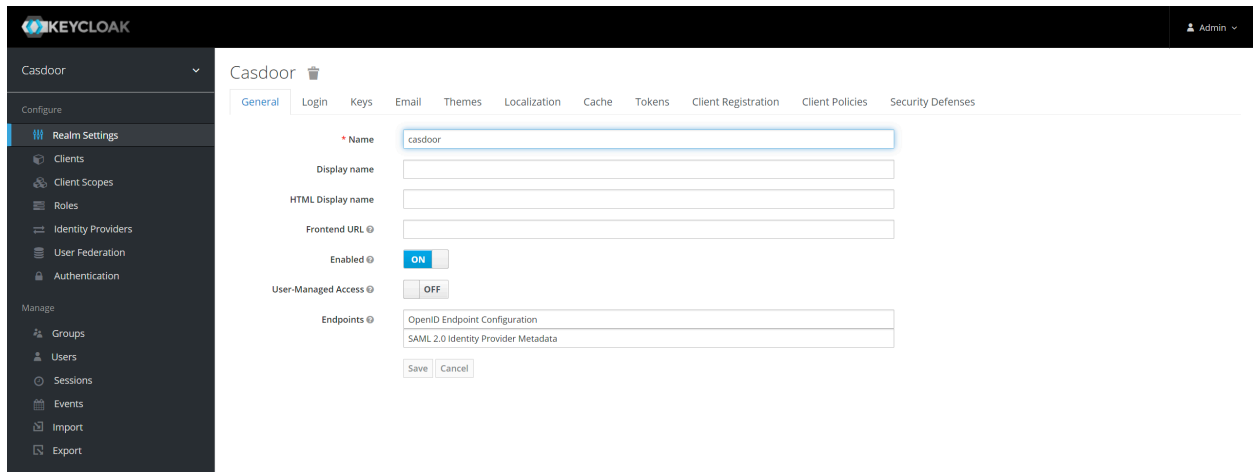
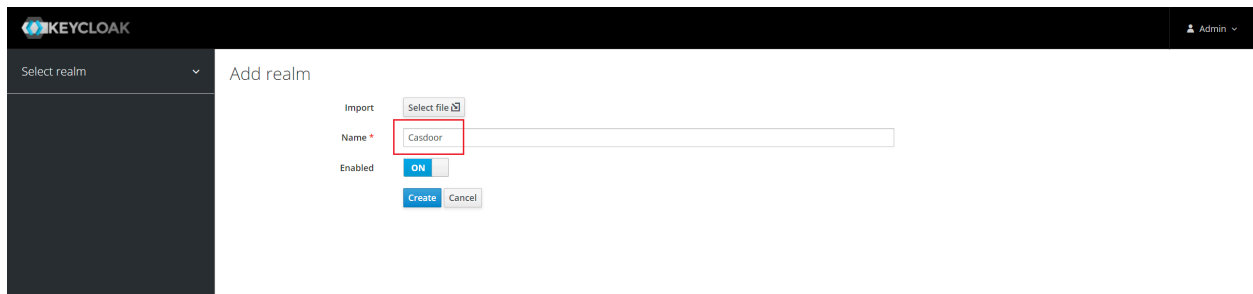
Here is an example of how to configure a new client entry in Keycloak and configure Casdoor to use it to allow UI login by Keycloak users who are granted access via Keycloak configuration.

Configure Keycloak

For this example, let's make the following configuration choices and assumptions:

- Assume that you are running Casdoor in dev mode locally. The Casdoor UI is available at `http://localhost:7001` and the server is available at `http://localhost:8000`. Replace with the appropriate URL as needed.
- Assume that you are running Keycloak locally. The Keycloak UI is available at `http://localhost:8080/auth`.
- Based on that, the SP ACS URL for this deployment will be: `http://localhost:8000/api/acs`.
- Our SP Entity ID will use the same URL: `http://localhost:8000/api/acs`.

You can use the default realm or create a new realm.



Add a client entry in Keycloak

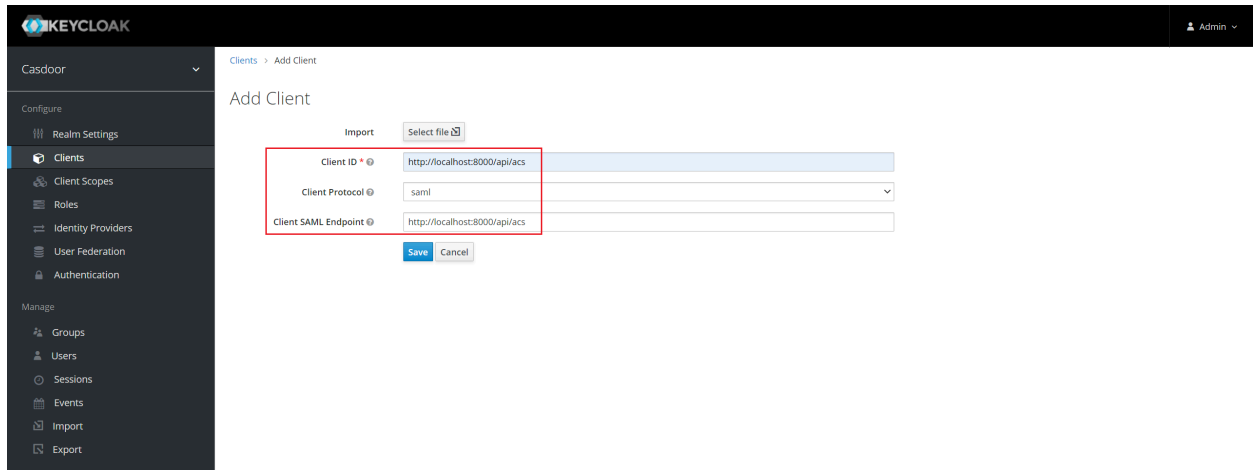
! INFO

For more details about Keycloak Clients, refer to the [Keycloak documentation](#).

Click **Clients** in the menu and then click **Create** to go to the **Add Client** page. Fill in the fields as follows:

- **Client ID:** `http://localhost:8000/api/acs` - This will be the SP Entity ID used in the Casdoor configuration later.
- **Client Protocol:** `saml`.
- **Client SAML Endpoint:** `http://localhost:8000/api/acs` - This URL is where

you want the Keycloak server to send SAML requests and responses. Generally, applications have one URL for processing SAML requests. Multiple URLs can be set in the Settings tab of the client.



Click Save. This action creates the client and brings you to the Settings tab.

The following are part of the settings:

1. Name - Casdoor. This is only used to display a friendly name to Keycloak users in the Keycloak UI. You can use any name you prefer.
2. Enabled - Select on.
3. Include Authn Statement - Select on.
4. Sign Documents - Select on.
5. Sign Assertions - Select off.
6. Encrypt Assertions - Select off.
7. Client Signature Required - Select off.
8. Force Name ID Format - Select on.
9. Name ID Format - Select username.
10. Valid Redirect URIs - Add http://localhost:8000/api/acs.
11. Master SAML Processing URL - http://localhost:8000/api/acs.

12. Fine Grain SAML Endpoint Configuration

i. Assertion Consumer Service POST Binding URL -

`http://localhost:8000/api/acs.`

ii. Assertion Consumer Service Redirect Binding URL -

`http://localhost:8000/api/acs.`

Save the configuration.

Client ID

Name

Description

Enabled

Always Display in Console

Consent Required

Login Theme

Client Protocol

Include AuthnStatement

Include OneTimeUse Condition

Force Artifact Binding

Sign Documents

Optimize REDIRECT signing key lookup

Sign Assertions

Signature Algorithm

SAML Signature Key Name

Canonicalization Method

Encrypt Assertions

Client Signature Required

Force POST Binding

Front Channel Logout

Force Name ID Format

Name ID Format

Root URL

Valid Redirect URIs

Base URL

Master SAML Processing URL

IDP Initiated SSO URL Name

IDP Initiated SSO Relay State

> Fine Grain SAML Endpoint Configuration

Assertion Consumer Service POST Binding URL

Assertion Consumer Service Redirect Binding URL

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

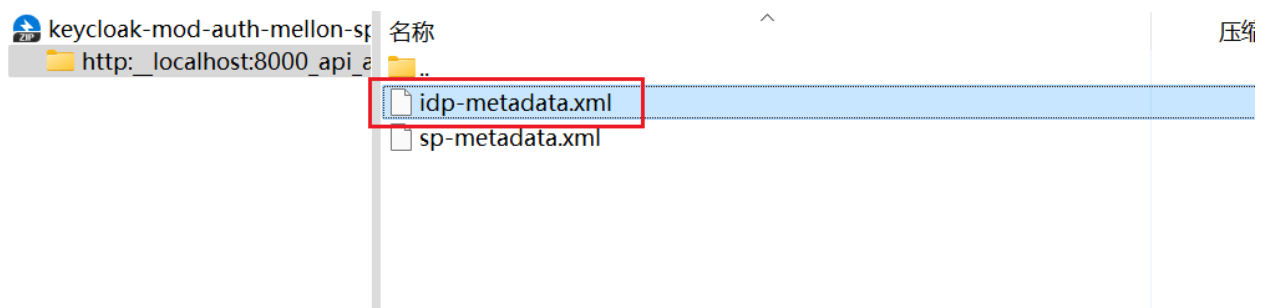
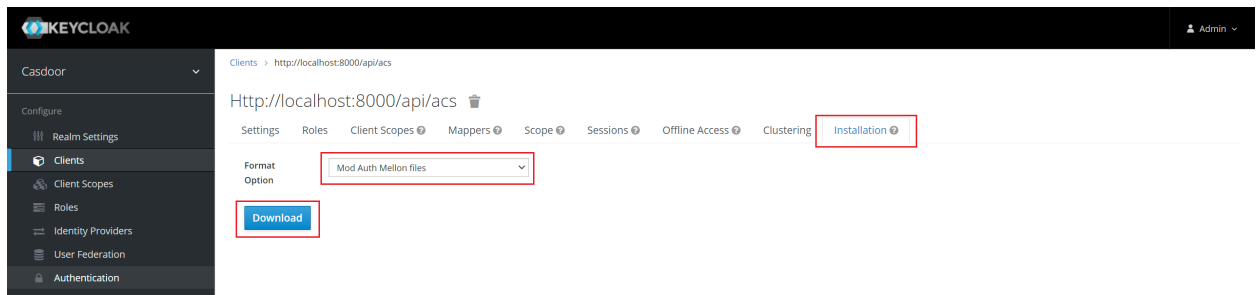


If you want to sign the authn request, you need to enable the **Client Signature Required** option and upload the certificate generated by yourself. The private key and certificate used in Casdoor, `token_jwt_key.key` and `token_jwt_key.pem`, are located in the **object** directory. In Keycloak, you need to click the **Keys** tab, click the **Import** button, select **Archive Format** as **Certificate PEM**, and upload the certificate.

Click **Installation** tab.

For Keycloak $\leq 5.0.0$, select **Format Option - SAML Metadata IDPSSODescriptor** and copy the metadata.

For Keycloak $6.0.0+$, select **Format Option - Mod Auth Mellon files** and click **Download**. Unzip the downloaded.zip, locate `idp-metadata.xml`, and copy the metadata.



Configure in Casdoor

Create a new provider in Casdoor.

Select category as **SAML**, type as **Keycloak**. Copy the content of metadata and paste it into the **Metadata** field. The values of **Endpoint**, **IdP**, and **Issuer URL** will be generated automatically after clicking the **Parse** button. Finally, click the **Save** button.





TIP

If you enable the **Client Signature Required** option in Keycloak and upload the certificate, please enable the **Sign request** option in Casdoor.

Name	<input type="text" value="keycloak-casdoor"/>
Display name	<input type="text" value="keycloak-casdoor"/>
Category	<input type="text" value="SAML"/>
Type	<input type="text" value="Keycloak"/>
Client ID	<input type="text"/>
Client secret	<input type="text"/>
Sign request	<input type="checkbox"/>
Metadata	<pre><md:EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata" xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" xmlns:ds="http://www.w3.org/2000/09/xmldsig#" entityID="http://localhost:8080/auth/realms/casdoor"> <md:IDPSSODescriptor WantAuthnRequestsSigned="true" protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol"> <md:KeyDescriptor use="signing"> <ds:KeyInfo> <ds:KeyName> zqm-3k76G-na5Zc3uIPDI7bp-4wYtMbWMPzwwUHAY </ds:KeyName> <ds:X509Data> <ds:X509Certificate> MIICnTCCAYUCBgF9pAmxSDANBgkqhkiG9w0BAQsFADASMRAwDgYDVQDDADjYXNkb29yMB4XDTEwMTIxMDExMDg1OFoXDTEwMTIxMDExMTAzOFowEjEQMA4GA1UEAwwHY2FzZG9vcjCCASlwDQYJKoZIhvcNAQEBBQADggEPADCCAQg...</pre>
	<input type="button" value="Parse"/>
Endpoint	<input type="text" value="http://localhost:8080/auth/realms/casdoor/protocol/saml"/>
IdP	<input type="text" value="MIICnTCCAYUCBgF9pAmxSDANBgkqhkiG9w0BAQsFADASMRAwDgYDVQDDADjYXNkb29yMB4XDTEwMTIxMDExMDg1OFoXDTEwMTIxMDExMTAzOFowEjEQMA4GA1UEAwwHY2FzZG9vcjCCASlwDQYJKoZIhvcNAQEBBQADggEPADCCAQg..."/>
Issuer URL	<input type="text" value="http://localhost:8080/auth/realms/casdoor"/>
SP ACS URL	<input type="text" value="http://localhost:8000/api/acs"/> <input type="button" value="Copy"/>
SP Entity ID	<input type="text" value="http://localhost:8000/api/acs"/> <input type="button" value="Copy"/>
Provider URL	<input type="text" value="https://github.com/organizations/xxx/settings/applications/1234567"/>

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

Providers [Add](#)

Name	Category	Type	canSignUp	canSignIn	canUnlink	prompted	Action
casdoor-idaas	SAML						^ v x
keycloak-casdoor	SAML						^ v x

Validate the effect

Go to the application you just configured and you will find a Keycloak icon on the login page.

Click the icon and you will be redirected to the Keycloak login page. After successful authentication, you will be logged into Casdoor.




Username, Email or phone

Password

Auto sign in [Forgot password?](#)

[Sign In](#)

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





Alibaba Cloud IDaaS

Create SAML application in Alibaba Cloud IDaaS

Login to the [Alibaba Cloud management console](#), search and go to the Application Identity Service (IDentity-as-a-Service, IDaaS).

The screenshot shows the 'Overview' page of the Alibaba Cloud IDaaS service. The page title is '应用身份管理 / 概览页'. The main content area is titled '应用身份服务' and contains a section 'IDaaS (IDentity-as-a-Service) 是企业客户提供的身份访问管理服务, IDaaS支持 EIAM 和 CIAM'. Below this, there are two bullet points describing EIAM and CIAM. To the right, there is a QR code for customer support and a '帮助文档' (Help Docs) section with links to various guides. At the bottom, there is a table comparing '标准版' (Standard Edition) and '专业版' (Professional Edition) features.

功能 (模块)	标准版	专业版
单点登录应用模板	不限制添加应用	
单点登录应用数量限制	--	
组织机构/租户/租户增量操作	✓	
应用授权方式	支持按组织/按用户/按分类授权	
登录方式	支持账户+密码/AD认证/微信扫码/短信登录等	
二次认证	支持OTP码, 支持短信验证码	
钉钉相关	支持钉钉同步数据到IDaaS, 支持钉钉扫码, 支持钉钉微应用	
LDAP同步/Excel导入/Scim同步	✓	
自建系统对接授权控制	✓	
		独立集群部署, 支持灵活扩容
		适合定制化场景, 针对使用场景提供定制化方案
		提供Connector 产品和部署方案, 实现更灵活的数据同步

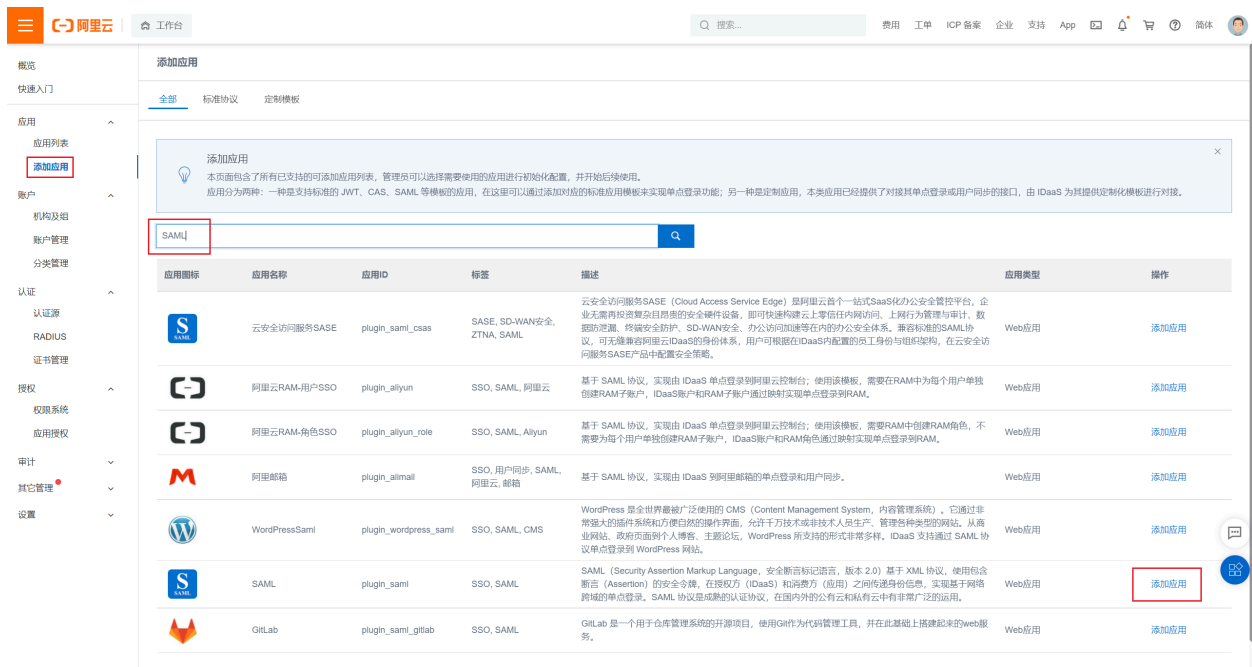
Click [EIAM Instance List](#) and open the free version.

The screenshot shows the 'EIAM Instance List' page in the Alibaba Cloud management console. The page title is '应用身份管理 / EIAM 实例列表'. The main content area is a table with columns: '实例ID/名称', '标准版实例ID', '状态 (全部)', '规格授权', '最大用户数', '到期时间', '产品版本', '用户登录页地址', '实例开放接口域名', and '操作'. A red box highlights the '开通免费版' (Open Free Version) button in the '操作' column. Below the table, there is a message '没有相关实例' (No related instances) and a pagination bar showing '1' of 1 pages.

An instance will be created and run automatically after opening. Click on the instance name or the **Manage** button to enter the IDaaS management console.



After entering the IDaaS management console, click **Add Application**, search for **SAML**, and click **Add Application**.



Click **Add SigningKey**.

添加应用 (SAML)



导入SigningKey

添加SigningKey

别名	序列号	有效期	秘钥算法	算法长度	操作
----	-----	-----	------	------	----

暂无数据

Fill in all required information and submit.

添加SigningKey



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

Select the added SigningKey.

添加应用 (SAML)



导入SigningKey

添加SigningKey

别名	序列号	有效期	秘钥算法	算法长度	操作
CN=CASDOOR-TEST, ST=Beijing, C=CN	3322747020095790430	1095	RSA	1024	选择 导出

Fill in all the required information below and submit.

- IDP IdentityId: Keep the same as Issuer URL in Casdoor.
- SP Entity ID & SP ACS URL(SSO Location): Now fill in whatever you want. After completing the configuration of Casdoor, you need to come to modify.
- Assertion Attribute: Directly fill in as username.
- Account Association Mode: Account Association

添加应用 (SAML)



图片大小不超过1MB

应用ID	idaas-cn-shanghai-pvl0hq0ojppplugin_saml		
* 应用名称	CASDOOR-SAML		
* IDP IdentityId	CASDOOR IDP IdentityId is required		
* SP Entity ID	http://localhost SP Entity ID is required		
* SP ACS URL(SSO Location)	http://localhost		
* NameIdFormat	urn:oasis:names:tc:SAML:2.0:nameid-format:transient		
* Binding	POST		
SP 登出地址	请输入SP 登出地址		
Assertion Attribute	username	应用子账户	- +
	断言属性。设值后，会将值放入SAML断言中。名称为自定义名称，值为账户的属性值。		
Sign Assertion	<input checked="" type="checkbox"/>		
IDaaS发起登录地址	IDaaS发起登录地址 以 http://、https:// 开头，填写后使用 IDaaS 发起登录将会跳转到该地址，而不会使用 SAML 的idp发起登录流程		
* 账户关联方式	<input checked="" type="radio"/> 账户关联 (系统按主子账户对应关系进行手动关联，用户添加后需要管理员审批) <input type="radio"/> 账户映射 (系统自动将主账户名称或指定的字段映射为应用的子账户)		
	提交	取消	

Account authorization & association

After the application is successfully added, an authorization prompt will pop up. Do not authorize it now, add an account and then authorize it.

Go to Organizations and Groups and click on New Account.

阿里云 工作台

搜索...

费用 工单 ICP备案 企业 支持 App 购物车 简体

机构及组 数据字典

机构及组

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

组织架构

在这里对组织架构进行管理。左键可选择组织架构，右键可对组织架构进行操作。

阿里云IDAAS

查看详情 岗位变动 导入 导出 配置LDAP 配置钉钉同步

账户 组 组织机构

新建账户 账户名称 请输入账户名称进行搜索 搜索

当前账户数 1 / 已购套餐规格为 100

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

共 1 条 < 1 > 10 条/页 跳至 1 页

Fill in all required information and submit.

新建账户



账户属性

扩展属性

父级组

父级

阿里云IDAAS

* 账户名称

casdoor

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

* 显示名称

casdoor

* 密码

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

邮箱

请输入有效的邮箱地址

手机号或邮箱至少填写一个。

手机号

+86 151

手机号或邮箱至少填写一个。

外部ID

外部ID

IDaaS 平台中的唯一身份标识, 若不填将由系统自动生成。

过期时间

过期时间



不填将使用系统默认过期时间 2116-12-31

备注

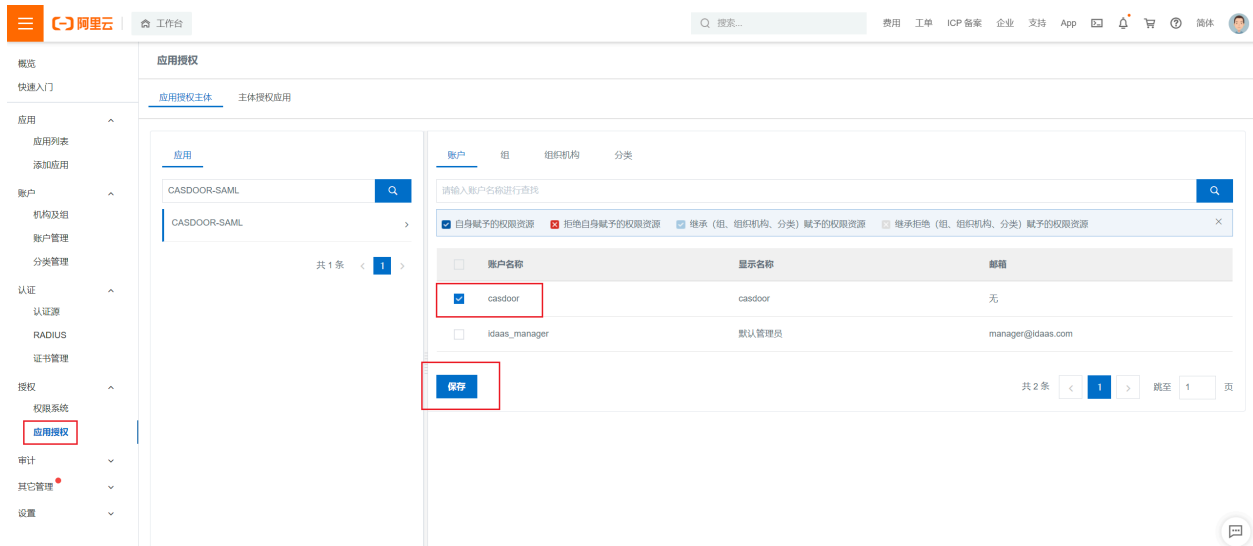
备注

用户备注信息

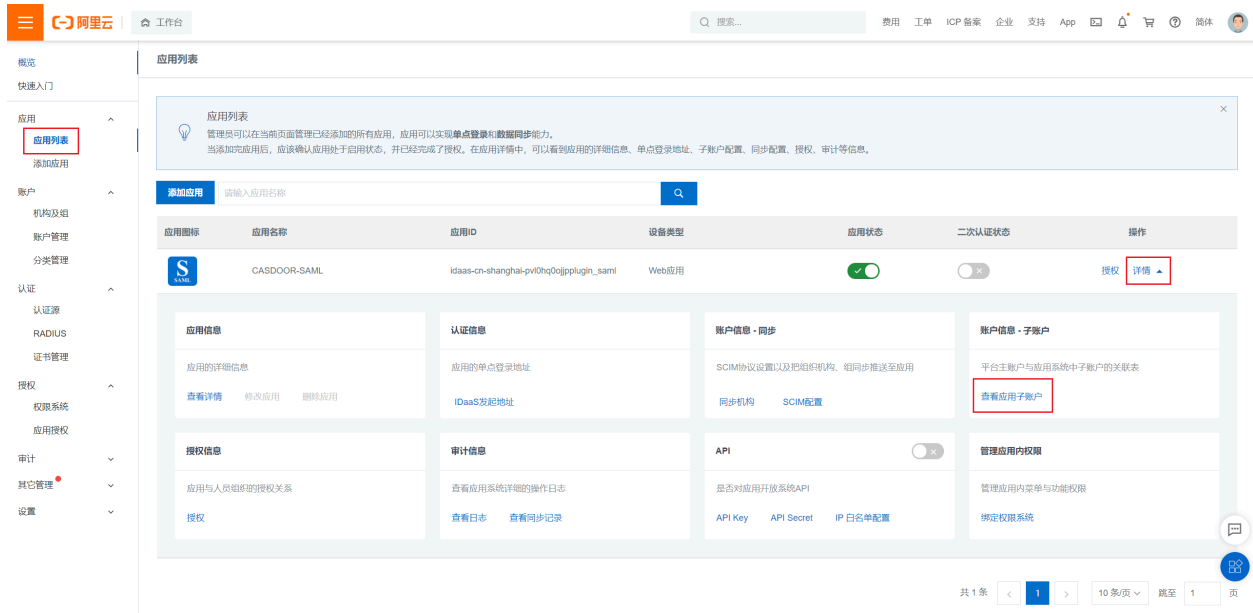
提交

取消

Go to Application Authorization, select the accounts you want to authorize and click Save.



Go to the Application List, click View application sub-accounts, and then click Add account association.



应用列表 / 子账户

子账户

添加账户关联 批量导入 批量导出

子账户

子账户指的是在指定应用系统中，用户会以什么身份进行访问。主账户指的是 IDaaS 中的账户。在进行单点登录时，IDaaS 会向应用系统传递对应的子账户，该子账户需要在应用系统中存在且可识别。

举例：IDaaS 中有主账户张三（用户名 zhangsan），在企业的 BPM 应用系统中，这个用户的用户名是 agoodman，即子账户应为 agoodman，与主账户 zhangsan 进行关联。

账户关联方式：在应用创建时，如果选择了账户映射，即默认主账户和子账户完全一致，无需配置。如果选择了账户关联，则需要在这里进行手动子账户创建和主子账户关联。

CASDOOR-SAML

主账户（账户名称）

账户名称	显示名称	子账户	子账户密码	是否关联	审批状态	关联时间	操作
暂无数据							

共 0 条 < 1 > 跳至 1 页

Fill in the primary and sub accounts that need to be associated and click **Save**.

The primary account exists in IDaaS, and the sub account is the ID of the user in Casdoor.

添加账户关联

* 主账户

* 子账户

保存 返回

Export IDaaS Metadata

Go to the Application List, click View Application Details and click Export IDaaS SAML Metadata.

The screenshot displays the Alibaba Cloud IAM console interface. On the left, a navigation sidebar includes sections like '应用列表' (Application List), '添加应用' (Add Application), and '应用详情' (Application Details). The main content area is titled '应用详情 (CASDOOR-SAML)'. It features a table with application metadata and a detailed configuration section. The configuration section includes fields for SigningKey, NameIDFormat, SP ACS URL, IDP IdentityId, SP Entity ID, Binding, Sign Assertion, Assertion Attribute, IDaaS发起登录地址, and SP发起地址. A red box highlights the 'IDP IdentityId' field, which contains the value 'CASDOOR' and a link to '导出 IDaaS SAML 元配置文件' (Export IDaaS SAML Metadata File).

应用图标	应用名称	应用ID
	CASDOOR-SAML	idaas-cn-shanghai-..._jin_saml

应用信息	认证信息
应用的详细信息 查看详情 修改应用 删除应用	应用的单点登录地址 IDaaS发起地址

授权信息	审计信息
应用与人员组织的授权关系 授权	查看应用系统的操作日志 查看日志 查看同步记录

图标	
应用ID	idaas-cn-shanghai-..._jin_saml
应用名称	CASDOOR-SAML
应用Uuid	d9e05906f3c5c031b7abb0efa845f7bRxS506SQ3y7
SigningKey	3322747020095780430(CN=CASDOOR-TEST)
NameIDFormat	urn:oasis:names:tc:SAML:2.0:nameid-format:transient
SP ACS URL	http://localhost
IDP IdentityId	CASDOOR 导出 IDaaS SAML 元配置文件 立即轮转密钥 导入
SP Entity ID	http://localhost
Binding	POST
Sign Assertion	是
Assertion Attribute	username=APPLICATIONUSERNAME
IDaaS发起登录地址	
SP发起地址	https://dymoykbbkx.login.aliyunidaas.com/enduser/api/application/plugin_saml/idaas-cn-shanghai-..._jin_saml/sp_sso?SAMLRequest=xxx&RelayState=yyy

Configure in Casdoor

Create a new provider in Casdoor.

Select category as **SAML**, type as **Alibaba Cloud IdaaS**. Copy the content of metadata and paste it to the **Metadata** input. The values of **Endpoint**, **IdP** and **Issuer URL** will be generated automatically after clicking the **Parse** button.

Name: casdoor-idaas

Display name: casdoor-idaas

Category: SAML

Type: Aliyun IDaaS

Client ID:

Client secret:

Metadata:

```
<?xml version='1.0' encoding='utf-8'><md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:base="http://www.w3.org/2001/XMLSchema-instance"><md:IDPSSODescriptor></md:IDPSSODescriptor></md:EntityDescriptor>
```

Endpoint: https://dvmoykbbkx.login.aliyundaas.com/endpoint/api/application/plugin_saml/idaas-cn-shanghai-..._saml/sp_sso

IdP: MIB5zCCAVcGAWiBAGllHzEz2NMHV4wDQYIKoZlhcNAQEFBQAwNjELMAkGA1UEBHMCMQ04xEDACBgNVBAGTB0JlaWppbmcxFTATBgNVBAMTDENBUORPT1I1VEVTVDAeFw0yMTYyMDkwNzEyMTFaFw0yNDEyMDgwNzEyMTFaMDYxCzAJE

Issuer URL: CASDOOR

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

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

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

Save

Copy the SP ACS URL and the SP Entity ID and click the Save button.

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

Providers:

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

Preview: Test signup page... Test signin page...

Modify SAML application in Alibaba Cloud IDaaS

Disable the application and then click **Modify Application**.

阿里云 工作台

应用列表

应用列表
管理页可以在当前页面管理已经添加的所有应用，应用可以实现单点登录和数据同步能力。
当添加完应用后，应该确认应用处于启用状态，并已经完成了授权。在应用详情页中，可以看到应用的详细信息、单点登录地址、子账户配置、同步配置、授权、审计等信息。

添加应用 请输入应用名称

应用图标	应用名称	应用ID	设备类型	应用状态	二次认证状态	操作
	CASDOOR-SAML	idaas-cn-shanghai-pv0hq0ojjplugin_saml	Web应用	<input checked="" type="checkbox"/>	<input type="checkbox"/>	授权 详情

应用信息
应用的详细信息
查看详情 修改应用 删除应用

认证信息
应用的单点登录地址
IDaaS发起地址

账户信息 - 同步
SCIM协议设置以及把组织机构、组同步推送至应用
同步机构 SCIM配置

账户信息 - 子账户
平台主账户与应用系统中子账户的关联表
查看应用子账户

授权信息
应用与人员组织的授权关系
授权


审计信息
查看应用系统详细的操作日志
查看日志 查看同步记录

API
是否对应用开放系统API
API Key API Secret IP白名单配置

管理应用内权限
管理应用内菜单与功能权限
绑定权限系统

共 1 条 < 1 > 10 条/页 跳至 1 页

Fill in SP Entity ID and SP ACS URL(SSO Location) with the content copied in Casdoor. Submit and enable application.

图标	 <input type="button" value="上传文件"/> 图片大小不超过1MB
应用ID	idaas-cn-shanghai-...login_saml
* 应用名称	CASDOOR-SAML
* IDP IdentityId	CASDOOR IDP IdentityId is required
* SP Entity ID	http://localhost:8000/api/acs SP Entity ID is required
* SP ACS URL(SSO Location)	http://localhost:8000/api/acs
* NameIdFormat	urn:oasis:names:tc:SAML:2.0:nameid-format:transient
* Binding	POST
SP 登出地址	请输入SP 登出地址
Assertion Attribute	username 应用子账户 - + 断言属性。设值后，会将值放入SAML断言中。名称为自定义名称，值为账户的属性值。
Sign Assertion	<input checked="" type="checkbox"/>
IDaaS发起登录地址	IDaaS发起登录地址 以 http://、https:// 开头，填写后使用 IDaaS 发起登录将会跳转到该地址，而不会使用 SAML 的idp发起登录流程

Validate the effect

Go to the application you just configured and you can find that there is an icon in the login page.

Click the icon and jump to the Alibaba Cloud IDaaS login page, and then successfully login to the Casdoor after authentication.



Auto sign in

[Forgot password?](#)

[Sign In](#)

[Sign in with code](#)

[No account? sign up now](#)



Payment

Overview

Add Payment providers to your application

PayPal

Add PayPal as a payment provider to your application

Stripe

Add Stripe payment provider to your application

Alipay

Add Alipay payment provider to your application

WeChat Pay

Add WeChat Pay payment provider to your application

Overview

If you want to use payment services in Casdoor, you need to create a Payment provider and add it to your products.

The screenshot shows the Casdoor interface with the 'Providers' menu item highlighted in red. The 'Edit Provider' form is displayed with the following fields:

- Name: provider_payment_paypal
- Display name: PayPal Payment Provider
- Organization: admin (Shared)
- Category: Payment (highlighted with a red box)
- Type: PayPal (selected, with a dropdown menu open showing options: PayPal, Alipay, Dummy, GC, PayPal, WeChat Pay)
- Client ID: (empty)
- Client secret: (empty)
- Provider URL: (empty)

Buttons for 'Save' and 'Save & Exit' are visible at the top and bottom of the form.

To learn how to configure a product, refer to [Product](#). After configuring a product, you can add Payment providers for the product so that users can purchase the product through the Payment providers.

PayPal

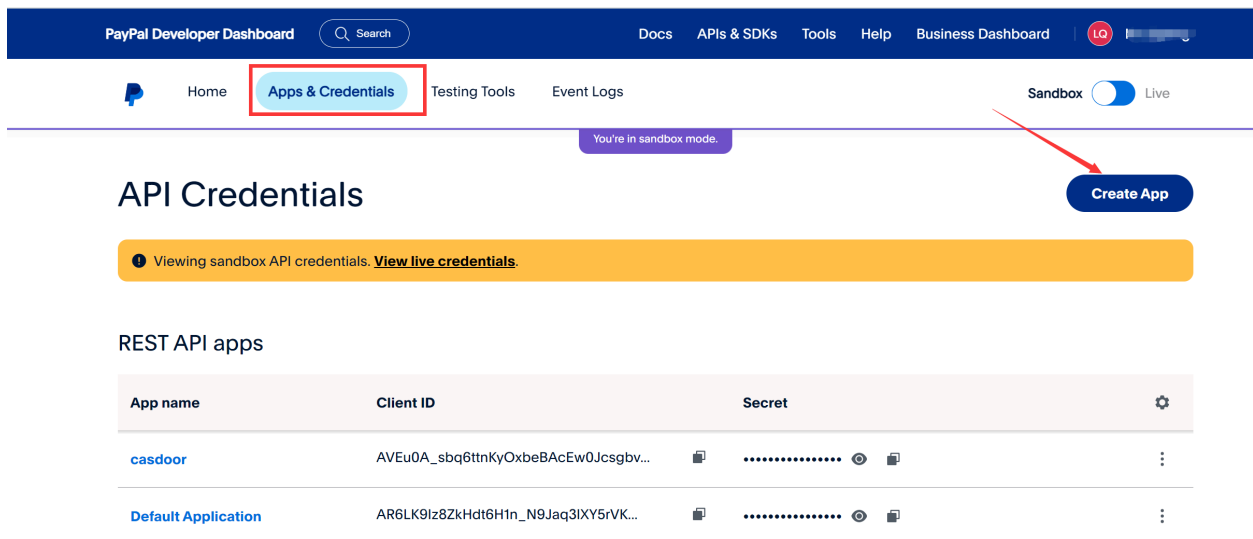
NOTE

This is an example of how to configure the PayPal payment provider.

Step 1: Create a PayPal application

First, you need to create an application in PayPal. To access the PayPal Developer site, you should have a PayPal business account. If you don't have an account, [create one](#) first.

After you create a PayPal business account, log in to the [Developer Dashboard](#) using your account and then click on `Create App` under `Apps & Credentials`.



The screenshot shows the PayPal Developer Dashboard interface. The top navigation bar includes 'PayPal Developer Dashboard', a search bar, and links for 'Docs', 'APIs & SDKs', 'Tools', 'Help', and 'Business Dashboard'. The main navigation area has 'Home', 'Apps & Credentials' (highlighted with a red box), 'Testing Tools', and 'Event Logs'. A 'Sandbox' toggle is set to 'Live'. A red arrow points to the 'Create App' button in the top right corner. Below the navigation, a yellow banner indicates 'Viewing sandbox API credentials. [View live credentials](#)'. The 'REST API apps' section contains a table with the following data:

App name	Client ID	Secret	
casdoor	AVEu0A_sbq6ttnKyOxbeBAcEw0Jcsgbv...	ⓘ 📄 ⋮
Default Application	AR6LK9Iz8ZkHdt6H1n_N9JaQ3IXY5rVK...	ⓘ 📄 ⋮

You can find the `Client ID` and `Secret key` in the basic information of your application.

← [Back](#)

casdoor

🔔 Viewing sandbox API credentials. [View live credentials.](#)





API credentials

App name	casdoor 
Client ID	AVEu0A_sbq6ttnKyOxbeBAcEw0Jcsgbv2JZvQAAtK  JFnaULI-EK-U2XIXcEpEouO9oIknbU7c3m_lfRT5
Secret key 1   

[+ Add Second Key](#)

Sandbox account info

[View details](#)

Sandbox URL	https://sandbox.paypal.com 
Sandbox Region	C2
Email	sb-qqaiiv26894991@business.example.com 
Password  

Features

Step 2: Create a PayPal payment provider

Next, create a PayPal payment provider in Casdoor. Fill in the necessary information:

Name	Name in PayPal
Category	Choose <input type="text" value="Payment"/>

Name	Name in PayPal
Type	Choose <code>PayPal</code>
Client ID	Use the <code>Client ID</code> obtained from Step 1
Client secret	Use the <code>Secret key</code> obtained from Step 1

Step 3: Add the PayPal payment provider for your product

Finally, add the PayPal payment provider for your product so that users can purchase the product using PayPal.

Price:

Quantity:

Sold:



Payment providers:

Return URL:

State:

Preview: [Test buy page](#)

Buy Product

Name	Test Product				
Detail	This is the detail of test product	Tag	Casdoor Summit 2022	SKU	test_product
Image					
Price	\$10.03 (USD)	Quantity	99	Sold	10
Pay					

i NOTE

The above operations are all performed in PayPal's `Sandbox` mode. If you want to use it in a live production environment, you need to create an application in PayPal's `Live` mode and set `runmode=prod` in Casdoor's configuration file `conf/app.conf`.

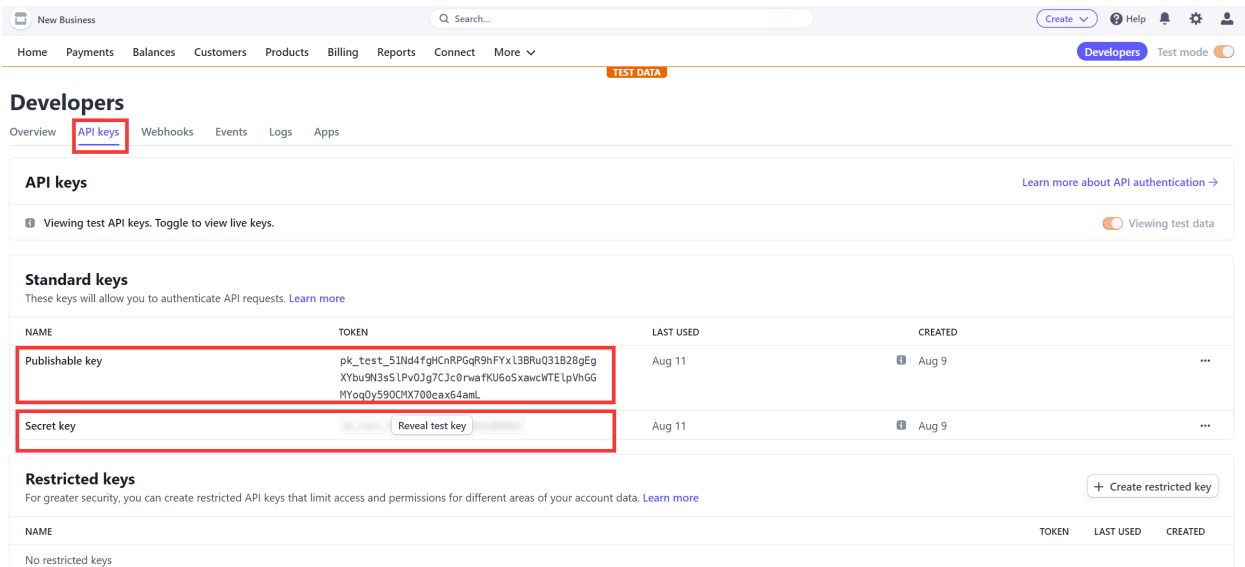
Stripe

NOTE

This is an example of how to configure a Stripe payment provider.

Step 1. Get Publishable Key and Secret Key

First, you need to have an account at [Stripe](#). After creating a Stripe account, log in to the [Developer Dashboard](#) using your account credentials. You can find the `Publishable key` and `Secret key` under the `API keys` tab.



The screenshot shows the Stripe Developer Dashboard. The 'API keys' tab is selected and highlighted with a red box. Below the navigation, there is a 'TEST DATA' indicator. The main content area is titled 'API keys' and includes a toggle for 'Viewing test API keys'. Under the 'Standard keys' section, a table lists the 'Publishable key' and 'Secret key', both of which are highlighted with red boxes. The 'Publishable key' row shows a long alphanumeric token and a creation date of 'Aug 9'. The 'Secret key' row shows a shorter alphanumeric token and a creation date of 'Aug 9'. There is also a 'Restricted keys' section at the bottom, which is currently empty.

NAME	TOKEN	LAST USED	CREATED
Publishable key	pk_test_51Nd4fghCnRPGq9hFYxL3BRuQ31B28gEgXYbu9l3eS1PvOjg7CjC0rvaafKU6o5xawcWTElpVhGGMYoqDy590CMX700caX64amL	Aug 11	Aug 9
Secret key	Reveal test key	Aug 11	Aug 9

Step 2. Create a Stripe Payment provider

Next, create a Stripe Payment provider in Casdoor by filling in the necessary

information.

Name	Name in Stripe
Category	choose <code>Payment</code>
Type	choose <code>Stripe</code>
Client ID	<code>Publishable key</code> obtained from Step 1
Client secret	<code>Secret key</code> obtained from Step 1

Edit Provider Save Save & Exit

Name ⓘ :

Display name ⓘ :

Organization ⓘ :

Category ⓘ :

Type ⓘ :

Client ID ⓘ :

Client secret ⓘ :


Provider URL ⓘ :


Save Save & Exit


Step 3. Add the Stripe Payment provider for your product


Finally, add the Stripe Payment provider for your product so that users can


purchase the product using Stripe.


Currency : USD


Price : 10.04


Quantity : 99

Sold : 10




Payment providers : provider_payment_paypal x provider_payment_stripe x

Return URL : eP

State : Published

Preview : [Test buy page.](#)

Buy Product

Name	Test Product				
Detail	This is the detail of test product	Tag	Casdoor Summit 2022	SKU	test_product
Image					
Price	\$10.04 (USD)	Quantity	99	Sold	10
Pay	 PayPal  Stripe				

Alipay

Step 1. Preparation

First, you need to have a merchant account at Alipay Open Platform.

Before accessing the Alipay, there are some preparations that need to be done.

You can refer to the documentation [preparation before access](#) for more information.

1.1 Get APPID

Login the Alipay Open Platform Console and [create an application](#).

How to get the `APPID` : [Alipay APPID Query Guide](#)

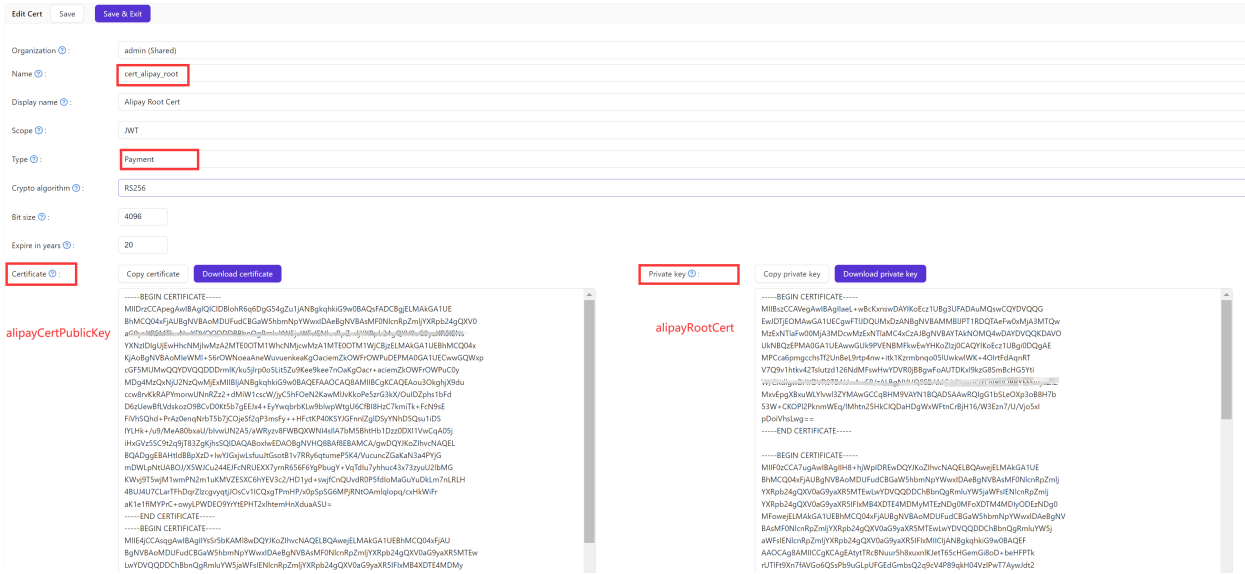
1.2 Configure Cert

Generate an RSA2 certificate based on the [document](#) and then you can obtain the `appPrivateKey.txt` and `appPublicKey.txt`.

Upload the certificate to the applicaiton and then you can download three files: `alipayRootCert.crt`, `appCertPublicKey.crt`, `alipayCertPublicKey.crt`.

Create a Cert called `App Cert` at Casoor:

Name	Name in Alipay
Type	choose <code>Payment</code>



Step 2. Create an Alipay Payment provider

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

Name	Name in Alipay
Category	choose Payment
Type	choose Alipay
Client ID	APPID obtained from Step 1.1
Cert	App Cert configured at Step 1.2
Root Cert	Root Cert configured at Step 1.2


Edit Provider Save Save & Exit

Name ⓘ: provider_payment_alipay

Display name ⓘ: Alipay Payment Provider

Organization ⓘ: admin (Shared)

Category ⓘ: Payment


Type ⓘ:  Alipay

Client ID ⓘ: 2021003117621368

Client secret ⓘ:

Cert ⓘ: cert_alipay_app

Root Cert ⓘ: cert_alipay_root

Provider URL ⓘ: 

Save Save & Exit

Step 3. Add the Alipay Pay Payment provider for your product

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

Quantity ⓘ: 99

Sold ⓘ: 10






Payment providers ⓘ: provider_payment_paypal x provider_payment_stripe x provider_payment_wechat x provider_payment_alipay x

Return URL ⓘ: eP

State ⓘ: Published

Preview ⓘ: Test buy page...

Buy Product

Name	Test Product		
Detail	This is the detail of test product	Tag	Casdoor Summit 2022
		SKU	test_product
Image			
Price	¥0.01 (CNY)	Quantity	99
		Sold	10
Pay	 PayPal  Stripe  WeChat Pay  Alipay		

Save Save & Exit

WeChat Pay

Step 1. Preparation

First, you need to have a merchant account at [WeChat Merchant Platform](#).

Before accessing the WeChat Pay, there are some preparations that need to be done.

You can refer to the documentation [preparation before access](#) for more information.

1.1 Get API Key v3

Log in to WeChat Merchant Platform, select `Account Settings > API Security > Set APIv3 Secret`, and click `Set APIv3 secret` to get the `API Key v3`.

Security Center

API Security

Account Management

Account Information

New Account

Staff Management

Settlement Info

Settlement Info

Agreement

API certificate

API certificate

API certificate are used to identify and define your ID; Some of the APIs with higher security level will require the certificate to identify you to avoid the loss caused by possible ID theft. [Help](#)

API certificate (CA issued) ✔ You have successfully applied

You have successfully applied for the certificate at 2020-03-13 17:17 [View](#) [Change](#)

Set APIv3 Secret

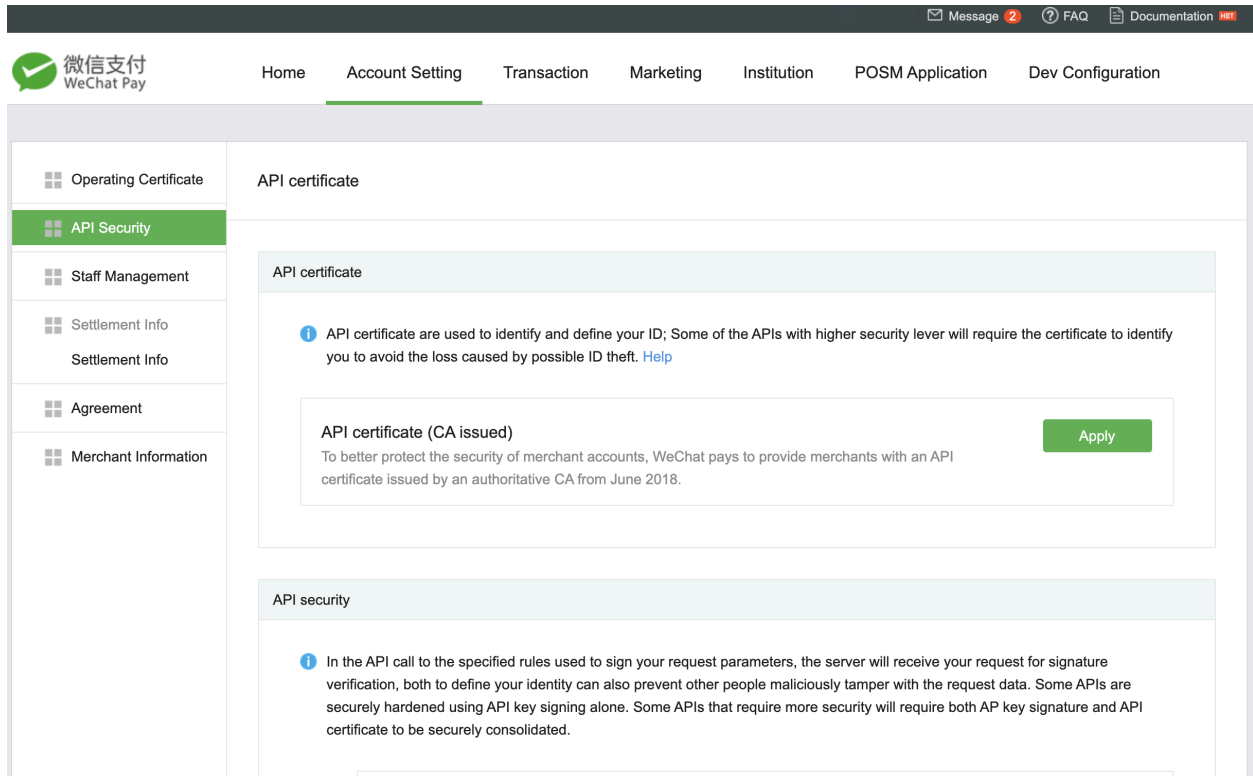
This key is used to encrypt messages in APIv3's "download platform certificate" and "payment callback notification"

Set APIv3 Secret Set APIv3 Secret

How to get API Key v3 : [APIv3 Key Settings](#)

1.2 Get Merchant Certificate

You can log in to WeChat Merchant Platform, and select [Account Settings > API Security > API Certificate](#) to download the certificate.



After download the certificate, get the **Certificate Serial Number** according to [How to view the Certificate Serial Number](#) and **Private Key** according to [How to get Private Key of Certificate](#).

Then, create a **cert** at Casdoor and fill the necessary information.

Name	Name in WeChat Pay
Type	choose WeChat Pay
Client ID	Merchant ID obtained from Step 1.3
Client secret	API Key v3 obtained from Step 1.1
App ID	App ID obtained from Step 1.3
Cert	Cert configured at Step 1.2

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

Name ⓘ : provider_payment_wechat

Display name ⓘ : Wechat Payment Provider

Organization ⓘ : admin (Shared)

Category ⓘ : Payment

Type ⓘ : WeChat Pay

Client ID ⓘ : 1619999244

Client secret ⓘ : ***

App ID ⓘ : wxe933a9cd81c396d1

Cert ⓘ : cert_wechatpay

Provider URL ⓘ : [🔗](#)

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

Step 3. Add the WeChat Pay Payment provider for your product

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

The screenshot shows the configuration interface for a product in Casdoor. On the left, there are fields for Currency (CNY), Price (0.01), Quantity (99), and Sold (10). Below these are tabs for Payment providers: provider_payment_paypal, provider_payment_stripe, and provider_payment_wechat (highlighted with a red box). There are also fields for Return URL, State (Published), and a Preview button.

The main preview area, titled "Buy Product", displays the following information:

Name	Test Product		
Detail	This is the detail of test product	Tag	Casdoor Summit 2022
		SKU	test_product
Image			
Price	¥ 0.01 (CNY)	Quantity	99
		Sold	10
Pay			

Support for JSAPI payment

Currently, Casdoor supports [JSAPI payment](#) and [Native payment](#) in WeChat Pay.

To support JSAPI payment, you should configure a [WeChat OAuth Provider](#) which support [WeChat Media Platform](#). The [Client ID 2](#) of WeChat OAuth Provider and the [App ID](#) of WeChat Pay Payment Provider need to be same.

The image displays two side-by-side screenshots of the 'Edit Provider' configuration interface. Both forms have a header with 'Edit Provider', 'Save', and 'Save & Exit' buttons.

Left Screenshot (Casdoor WeChat Provider):

- Name: provider_casdoor_wechat
- Display name: Casdoor WeChat
- Organization: admin (Shared)
- Category: OAuth
- Type: WeChat
- Client ID: wx049c70e6c2027b0b
- Client secret: ***
- Client ID 2: wx933a9cd81c396d1
- Client secret 2: ***
- Enable QR code:
- Provider URL: https://open.weixin.qq.com/

Right Screenshot (Payment - WeChatPay Provider):

- Name: provider_payment_wechatpay
- Display name: Payment - WeChatPay
- Organization: admin (Shared)
- Category: Payment
- Type: WeChat Pay
- Client ID: 1619999244
- Client secret: ***
- App ID: wx933a9cd81c396d1
- Cert: cert_wechatpay
- Provider URL: https://pay.weixin.qq.com/index.php/core/cert/api_cert#/

A red arrow points from the 'Client ID 2' field in the left form to the 'App ID' field in the right form, indicating that the same value is used for both.

After log in via WeChat(in the mobile scenario: e.g. the WeChat built-in browser inside the WeChat mobile app), users can purchase product using WeChat Pay based on JSAPI payment.

Captcha

Overview

Add a captcha to your application

Default

Using Casdoor's default captcha in your application

Cloudflare Turnstile

Add Cloudflare Turnstile to your application

reCAPTCHA

Add reCAPTCHA to your application

 **hCaptcha**

Add hCaptcha to your application

 **Alibaba Cloud Captcha**

Add Alibaba Cloud Captcha to your application













 **Geetest**

Add Geetest Captcha to your application

Overview

Casdoor can be configured to support different captchas to verify if the operation is performed by a human. By adding a captcha provider and applying it in the application, when users login, register, or forget their password and need to send a code, a captcha check dialog will appear to verify if the operation is performed by a human.

Currently, Casdoor supports multiple captcha providers. The following are the providers supported by Casdoor:

Default	Cloudflare Turnstile	reCAPTCHA	hCaptcha	Alibaba Cloud Captcha	Geetest
					
					

We will show you how to apply a captcha and add it to Casdoor.

Add a captcha provider

1. Navigate to your Casdoor index page.
2. Click on `Providers` in the top bar.
3. Click on `Add`, then you will see a new provider in the top list.
4. Click on the new provider to modify it.
5. Select `Captcha` in the `Category`.

6. Choose the captcha provider you need in the `Type`.
7. Fill in the most important information. Different captcha providers may require different information to be filled in.

Applying in the application

1. Click on `Application` in the top bar and choose one application to edit.
2. Click on the provider add button and select the provider you just added.
3. Done!

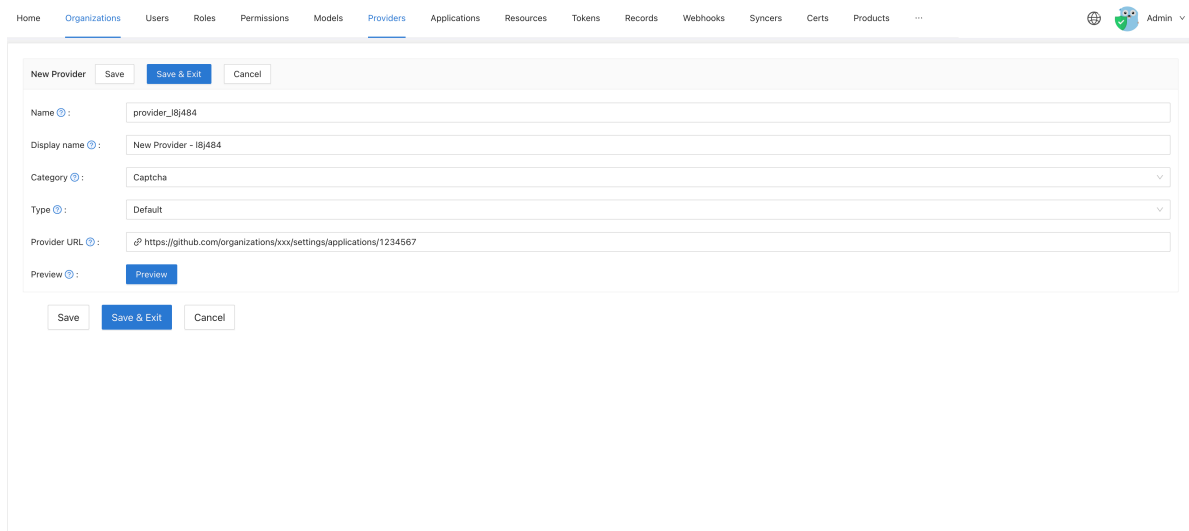
Default

The default captcha implementation generates and verifies an image. In the default captcha image, a sequence of digits 0-9 is used with a defined length of 5.

Configuring in Casdoor

To configure the default captcha in Casdoor, follow these steps:

1. Create a new provider in Casdoor.
2. Select the category as **Captcha**, and the type as **Default**.

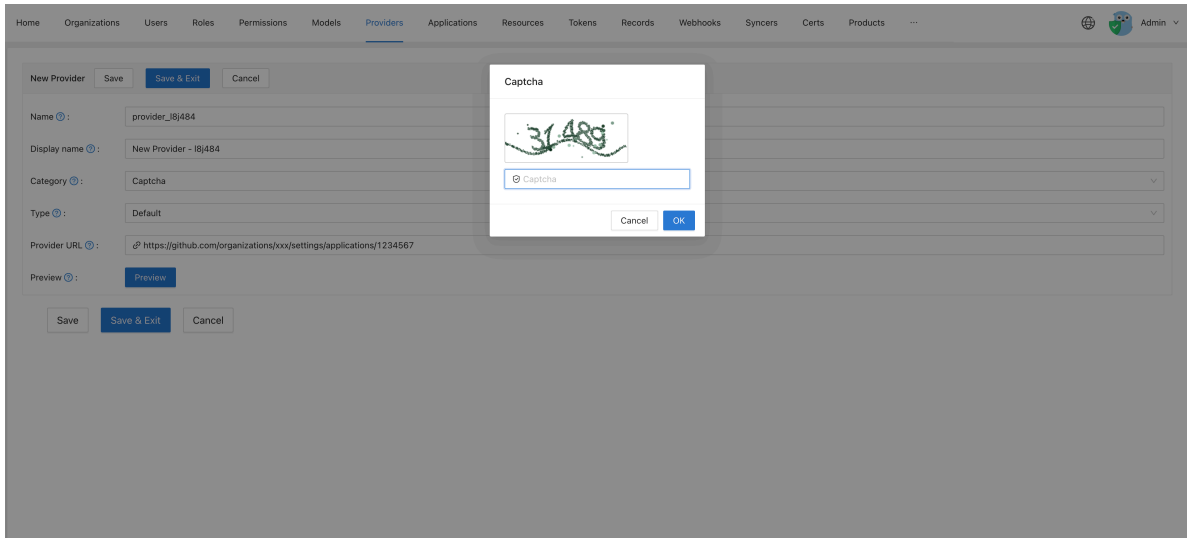


The screenshot shows the 'New Provider' configuration form in the Casdoor interface. The form is titled 'New Provider' and has three buttons at the top: 'Save', 'Save & Exit', and 'Cancel'. The form contains the following fields:

- Name**: provider_18j484
- Display name**: New Provider - 18j484
- Category**: Captcha (selected from a dropdown menu)
- Type**: Default (selected from a dropdown menu)
- Provider URL**: <https://github.com/organizations/xxx/settings/applications/1234567>
- Preview**: A blue button labeled 'Preview'.

At the bottom of the form, there are three buttons: 'Save', 'Save & Exit', and 'Cancel'.

3. Click on the **Preview** button to preview the style of this captcha.



Applying in your application

To apply the default captcha in your application, do the following:

1. Edit the application you want to configure in Casdoor.
2. Select the provider that you just added. There are three types of rules available:
 - **Always**: Always requires human-machine verification during login.
 - **None**: Never requires human-machine verification. The account will be blocked if it attempts to login with the wrong password for the 5th time within 15 minutes. The block will be lifted after 15 minutes.
 - **Dynamic**: After 5 failed login attempts, human-machine verification will be required but the account will not be blocked.

Providers ⓘ:

Name	Category	Type	canSignUp	canSignIn	canUnlink	prompted	Rule	Action
provider_4olfdm	Captcha						Always	⬆️ ⬇️ ⬇️
provider_casdoor_github	OAuth		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		⬆️ ⬇️ ⬇️
provider_casdoor_google	OAuth		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		⬆️ ⬇️ ⬇️

We also provide a demo video to demonstrate the differences in rules, which we hope will be helpful to you.

Cloudflare Turnstile

Cloudflare Turnstile is a CAPTCHA service provided by Cloudflare, which is a user-friendly, privacy-preserving alternative to CAPTCHA. You can find more details in the [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 obtain 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. It is recommended to choose `Managed`. Finally, click `Create`.

CLOUDFLARE

Add site 🔍 Support English (US) 👤

Navigation: Websites, Domain Registration, Analytics, Pages, Workers, R2, Stream, Images, Security Center, **Turnstile** (Beta), Zero Trust, Bulk Redirects (Beta), Notifications, Manage Account, Collapse sidebar

Add Site

[← Back to Turnstile Sites](#)

Site name
Add a name for the widget to identify it in the future


Domain
Enter your website's hostname or select from your existing websites on Cloudflare. (eg. domain.com or sub.domain.com)

Widget Type

- Managed**
Cloudflare will use information from the visitor to decide if an interactive challenge should be used. If we do show an interaction, the user will be prompted to check a box (no images or text to decipher).
- Non-interactive**
A purely non-interactive challenge. Users will see a widget with a loading bar while the browser challenge is run.
- Invisible**
Invisible challenge that does not require interaction.

Documentation
[Migrate from a CAPTCHA Service](#)
[Turnstile Developer Documentation](#)

You will then be able to obtain a site key and a secret key.

Casdoor Home Organizations Users Roles Permissions Models Adapters Applications **Providers** Resources ...  Admin ▾

Edit Provider

Name ⓘ:

Display name ⓘ:

Organization ⓘ:

Category ⓘ:

Type ⓘ:

Site key ⓘ:


Secret key ⓘ:

Provider URL ⓘ:

Preview ⓘ:

Powered by  Casdoor

You can click the Preview button to see a preview of the style of this CAPTCHA.

Casdoor Home Organizations Users Roles Permissions Models Adapters Applications **Providers** Resources ...  Admin ▾

Edit Provider

Name ⓘ:

Display name ⓘ:

Organization ⓘ:

Category ⓘ:

Type ⓘ:



Site key ⓘ:


Secret key ⓘ:

Provider URL ⓘ:

Preview ⓘ:

Captcha





 Success! 

Powered by  Casdoor

Application Integration

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

Providers ⓘ

Name	Category	Type	Can signup	Can signin	Can unlink	Prompted	Rule	Action
Cloudflare Turnstile	Captcha						None	  

reCAPTCHA

reCAPTCHA is provided by Google, and we use reCAPTCHA v2 Checkbox. You can find more details about it at this [link](#).

Create an API key pair

To start using reCAPTCHA, you need to [sign up for an API key pair](#) for your site. The key pair consists of a site key and secret key. The site key is used to invoke the reCAPTCHA service on your site or mobile application. The secret key authorizes communication between your application backend and the reCAPTCHA server to [verify the user's response](#).

First, choose the [type of reCAPTCHA](#) and then fill in the authorized domains or [package names](#). After you have accepted the terms of service, click **Register** to obtain a new API key pair.

The screenshot shows the 'Register a new site' page in the Google reCAPTCHA console. The page has a blue header with the 'Google reCAPTCHA' logo and a light blue navigation bar with a back arrow and the text 'Register a new site'. Below this is a yellow banner that reads 'Get unlimited assessments using reCAPTCHA Enterprise'. The main content area is white and contains several sections: 'Label' with a text input field containing 'reCaptcha' and a character count '9 / 50'; 'reCAPTCHA type' with three radio button options: 'reCAPTCHA v3' (Verify requests with a score), 'reCAPTCHA v2' (Verify requests with a challenge), and 'reCAPTCHA v2' with three sub-options: 'I'm not a robot' Checkbox (selected), 'Invisible reCAPTCHA badge', and 'reCAPTCHA Android'; 'Domains' with a plus sign and the text 'casdoor.org'; 'Owners' with the email 'resultelee@gmail.com' and '(You)'; and a checkbox for 'Accept the reCAPTCHA Terms of Service' which is checked. At the bottom, there is a small disclaimer: 'By accessing or using the reCAPTCHA APIs, you agree to the Google APIs Terms of Use, Google Terms of Use, and to the Additional Terms below. Please read and understand all applicable terms and policies before accessing the APIs.'

You will then receive a site key and a secret key.

The screenshot shows the Google reCAPTCHA console interface. At the top, it says "Google reCAPTCHA" and "Adding reCAPTCHA to your site". Below this, a message states "'reCaptcha' has been registered." There are two sections for key generation. The first section, "Use this site key in the HTML code your site serves to users.", includes a link to "See client side integration" and a "COPY SITE KEY" button next to a text box containing a long alphanumeric string. The second section, "Use this secret key for communication between your site and reCAPTCHA.", includes a link to "See server side integration" and a "COPY SECRET KEY" button next to a text box containing another long alphanumeric string. At the bottom, there are two buttons: "GO TO SETTINGS" and "GO TO ANALYTICS".

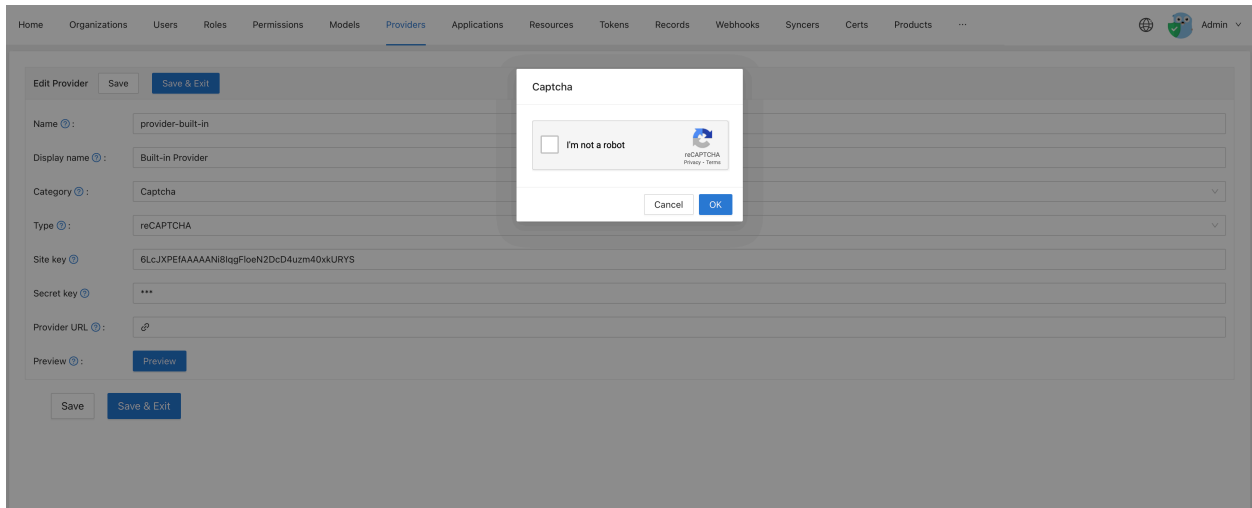
Configure in Casdoor

Create a new provider in Casdoor.

Select the category as **Captcha** and the type as **reCAPTCHA**. You need to provide the site key and secret key created in the previous step.

The screenshot shows the "New Provider" configuration form in Casdoor. The form has a header with "New Provider" and buttons for "Save", "Save & Exit", and "Cancel". The form fields are: "Name" (reCaptcha), "Display name" (reCaptcha), "Category" (Captcha), "Type" (reCAPTCHA), "Site key" (6Ld...), "Secret key" (6Ld...), and "Provider URL" (https://github.com/organizations/xxx/settings/applications/1234567). There is a "Preview" button at the bottom of the form. The bottom of the form also has "Save", "Save & Exit", and "Cancel" buttons.

You can click the **Preview** button to see the style of this captcha.



Apply in the application

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

Providers ⓘ Add

Name	Category	Type	canSignUp	canSignIn	canUnlink	prompted	Action
reCaptcha	Captcha						⬆ ⬇ 🗑

hCaptcha

hCaptcha is a captcha service provider, similar to reCAPTCHA. You can find more details about hCaptcha [here](#).

Create an API key pair

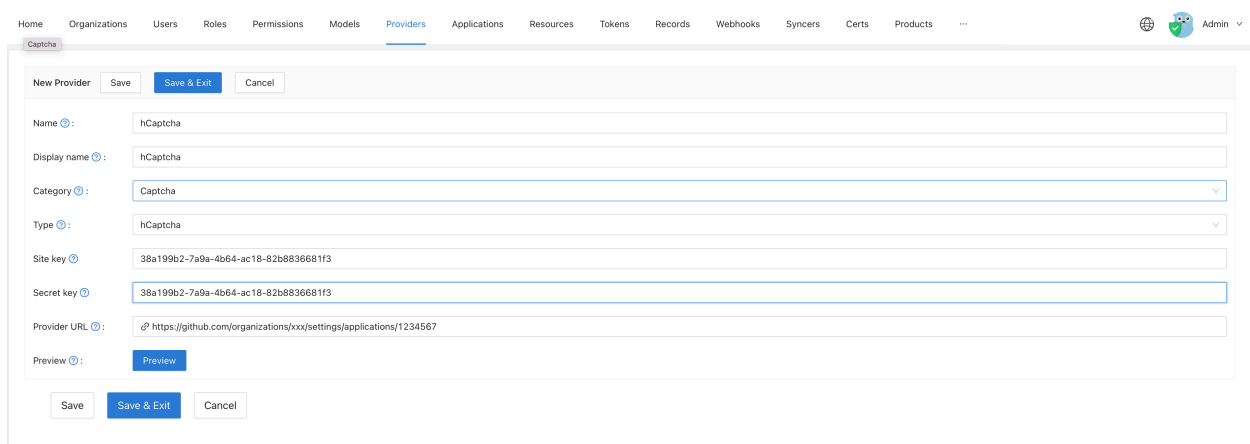
To start using hCaptcha, you need to sign up for an API key pair for your site. You can obtain your site key on your [profile page](#).

Once you have signed up, you will receive a site key and a secret key.

Configure in Casdoor

To configure hCaptcha in Casdoor, create a new provider.

Select the category as **Captcha** and the type as **hCaptcha**. Fill in the site key and secret key obtained in the previous step.

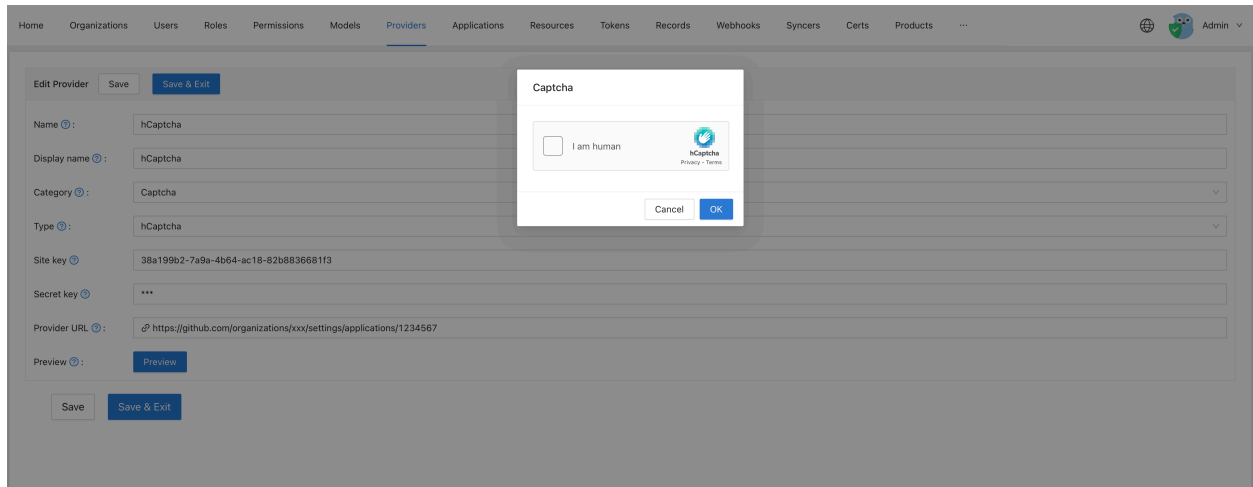


The screenshot shows the 'New Provider' configuration form in Casdoor. The form is titled 'New Provider' and has buttons for 'Save', 'Save & Exit', and 'Cancel'. The fields are as follows:

- Name: hCaptcha
- Display name: hCaptcha
- Category: Captcha (selected from a dropdown menu)
- Type: hCaptcha (selected from a dropdown menu)
- Site key: 38a199b2-7a9a-4b64-ac18-82b8836681f3
- Secret key: 38a199b2-7a9a-4b64-ac18-82b8836681f3
- Provider URL: <https://github.com/organizations/xxx/settings/applications/1234567>

At the bottom of the form, there is a 'Preview' button and another set of 'Save', 'Save & Exit', and 'Cancel' buttons.

You can click the **Preview** button to see how the captcha style will look.



Apply in your application

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



Alibaba Cloud Captcha

Alibaba Cloud Captcha is a captcha service provided by Alibaba Cloud. It offers two ways to verify captcha: "Sliding Validation" and "Intelligent Validation". You can find more details about it in this [link](#).

! INFO

Currently, only [Alibaba Cloud Captcha 1.0](#) is supported. [Alibaba Cloud Captcha 2.0](#) is currently in the public testing phase, so there are no plans for adaptation in the near term.

Add Captcha Configuration in Alibaba Cloud

To add the Captcha configuration, log in to the [Alibaba Cloud management console](#), search for and go to the Captcha Service. Then, click on **Confirm Open** to enable the Captcha Service.



Once you have entered the Captcha management console, click on **Add configuration**.

Alibaba Cloud Workbench

Search...

Expenses Tickets ICP Enterprise Support

云盾●验证码 验证码

公告：2021年3月18日起，人脸识别产品统一更名为验证码。

配置管理 数据监控

提示：配置管理目前暂时无法支持删除。若不再使用该条配置，代码中不进行调用即可，不影响其他配置使用。

新增配置 配置文档

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

共有2条 < 1 >

Fill in all the required information and submit the form.

Alibaba Cloud Workbench

Search...

Expenses Tickets ICP Enterprise Support

云盾●验证码 新增配置

验证码

1 配置服务内容 2 系统代码集成&测试 3 完成

配置名称: 同: 网站登录

高峰期QPS: 3000

业务类型: PC网页 HS (移动端WAP + APP)

验证方式: 滑动验证 智能验证 无痕验证

使用场景: 登录 注册 活动 论坛 短信 其它

产品形态预览: Demo

下一步

Now, you can view the **Scene** and **App key** in your console.

Also, the `Access key` and `Secret access key` can be found in your profile.

Configure in Casdoor

Create a new provider in Casdoor.

Select the category as `Captcha`, and the type as `hCaptcha`. Then, choose the sub-type: "Sliding Validation" or "Intelligent Validation". Make sure to fill in the `Access key`, `Secret access key`, `Scene`, and `App key` that you created in the previous step.

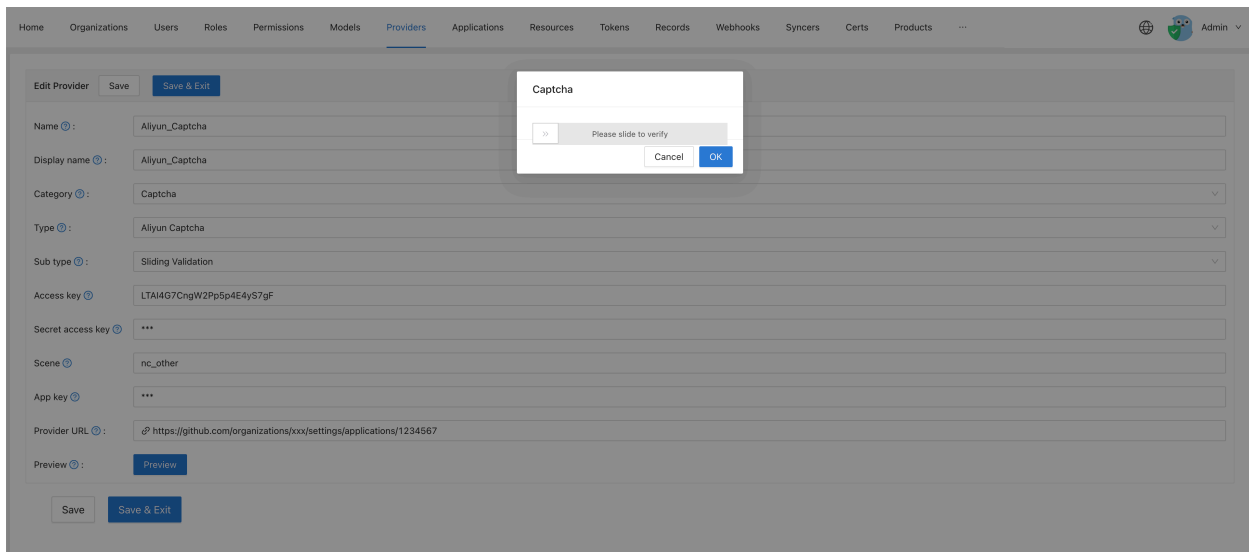
The screenshot shows the 'New Provider' configuration page. The form contains the following fields and values:

- Name: Aliyun_Captcha
- Display name: Aliyun_Captcha
- Category: Captcha
- Type: Aliyun Captcha
- Sub type: Sliding Validation
- Access key: LTAI4G7CngW2pp5p4E4yS7gF
- Secret access key: jP5*****N
- Scene: nc_other
- App key: F*****88
- Provider URL: https://github.com/organizations/xxx/settings/applications/1234567

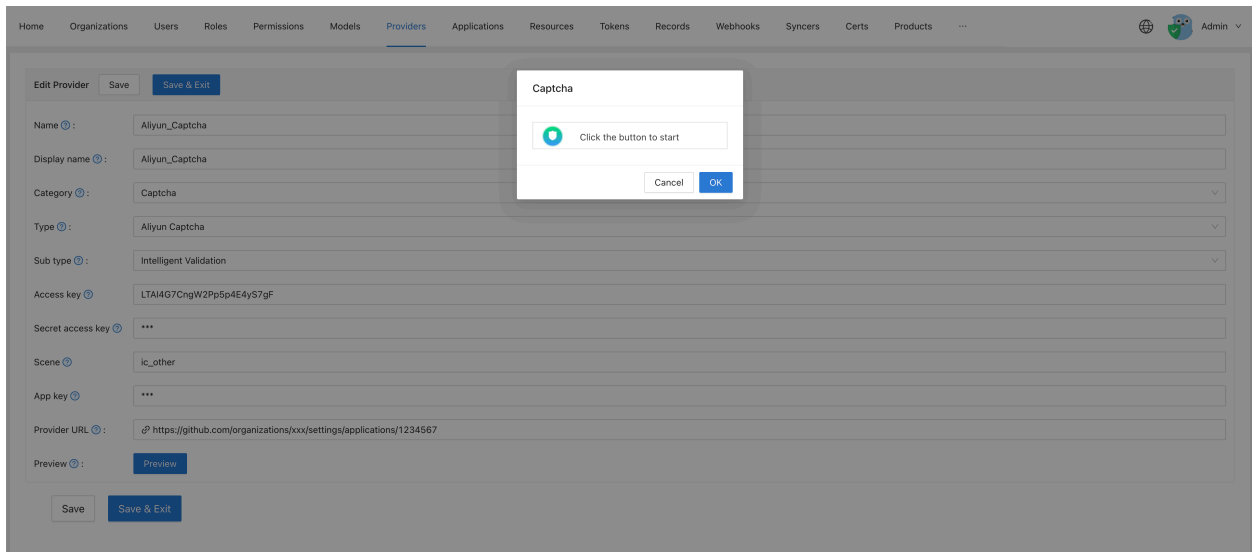
Buttons: Save, Save & Exit, Cancel, Preview.

You can click on the **Preview** button to see the style of this captcha.

The following image shows the preview of "Sliding Validation":



And this image shows the preview of "Intelligent Validation":



Application Integration

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

Providers 

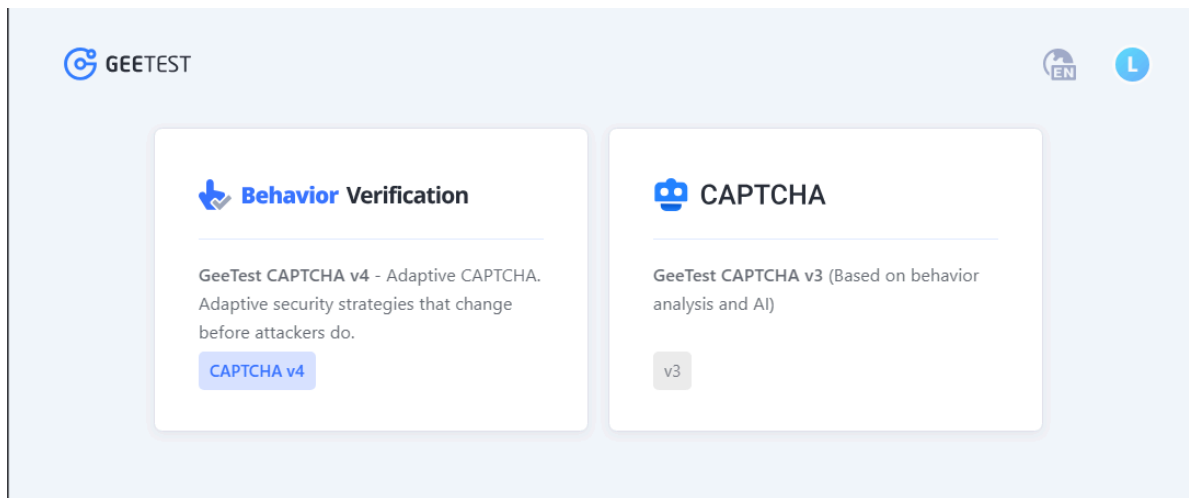
Name	Category	Type	canSignUp	canSignIn	canUnlink	prompted	Action
Aliyun_Captcha	Captcha						 

Geetest

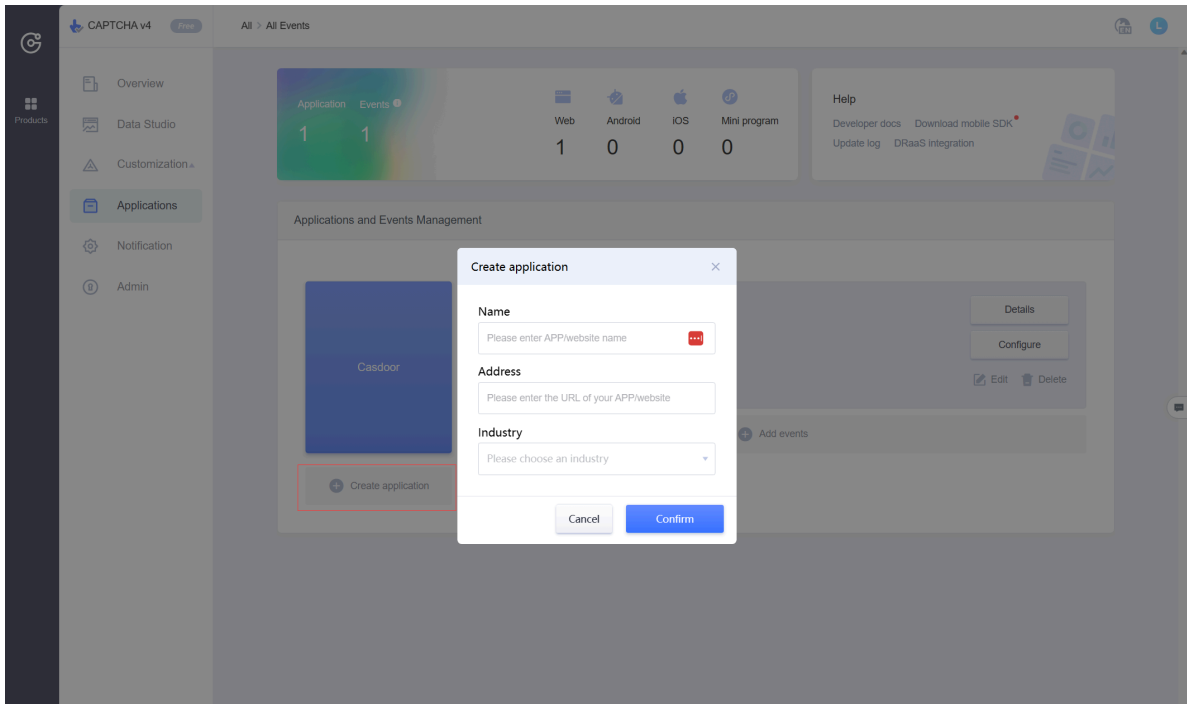
Configure Geetest

To configure Geetest and obtain the public and secret key, follow these steps:

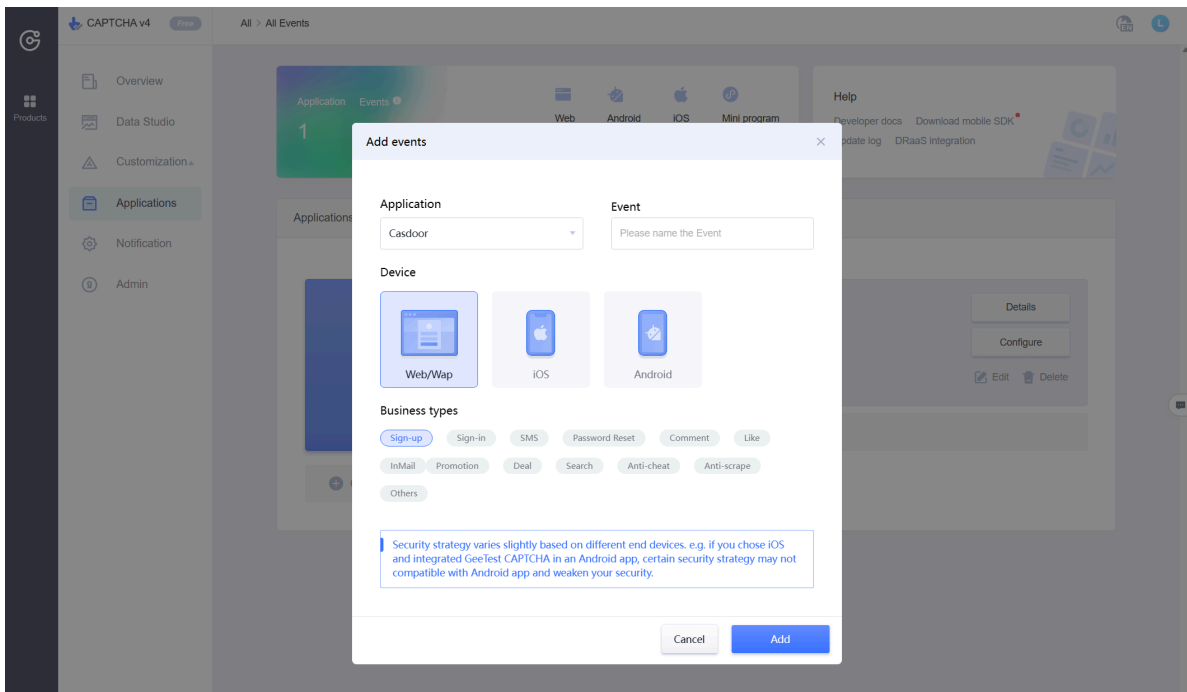
1. Go to the Geetest CAPTCHA V4 section on the [Geetest product page](#).



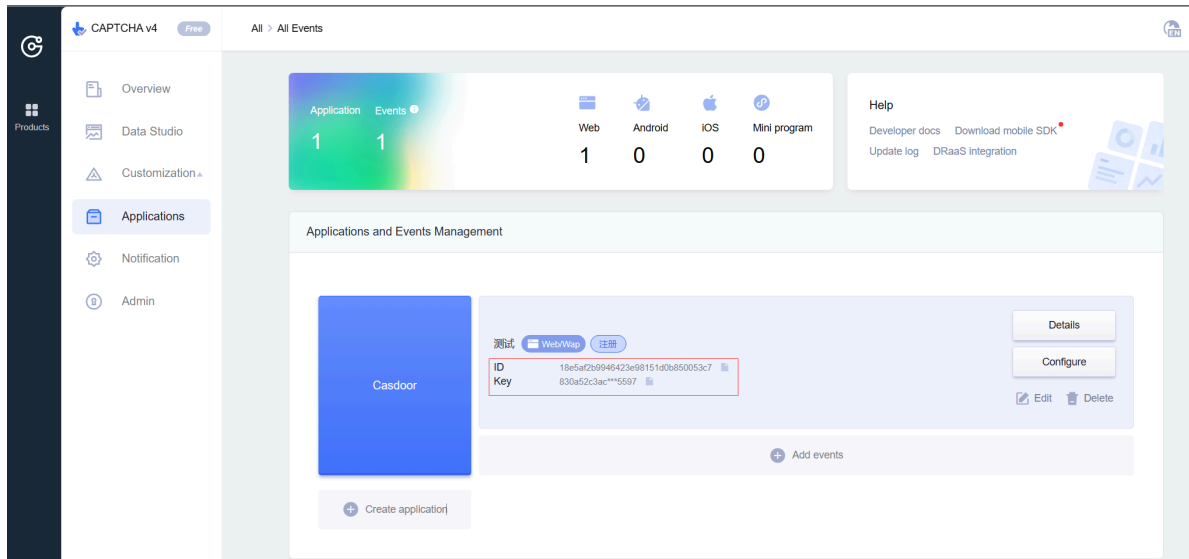
2. Create an application by entering the name and address for your application.



3. Add events and choose "web" for the device.



4. Retrieve the ID and Key.



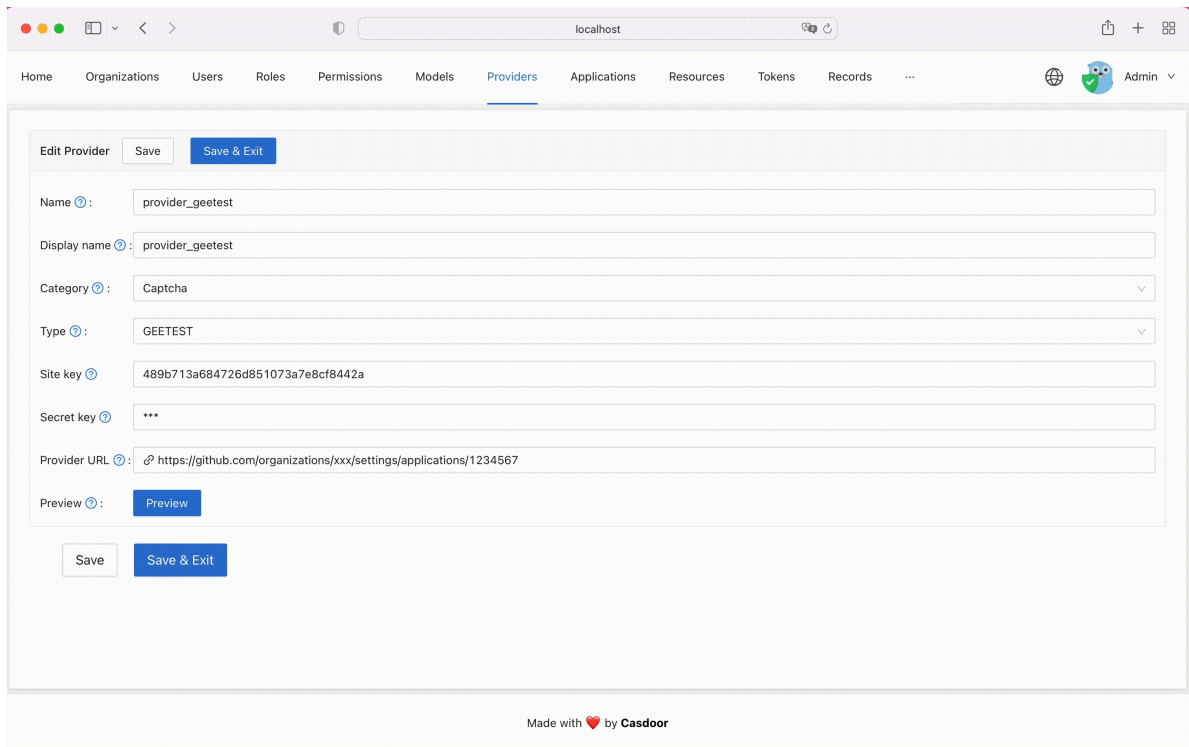
Configure Casdoor

Follow these steps to configure Casdoor:

1. Create a new provider in Casdoor.

Set the category as **Captcha** and the type as **Geetest**. Fill in the **Site key** and **Secret key** with the ID and Key obtained from Geetest.

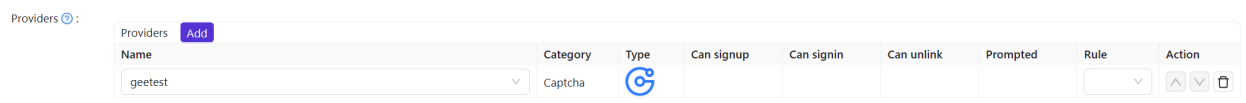
2. Click the Preview button to preview the style of this captcha.



Apply in your application

To apply the Geetest configuration in your application:

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



Web3

MetaMask

Adding the MetaMask Web3 provider to your application

Web3-Onboard

Add the Web3-Onboard Web3 provider to your application

MetaMask

i NOTE

This is an example of how to configure **MetaMask** as a Web3 provider.

MetaMask is a browser extension and app that functions as both a cryptocurrency wallet and a gateway to blockchain apps. Casdoor allows you to use MetaMask as an identity provider and enables Web3 login with MetaMask.

Step 1: Create a MetaMask Web3 provider

To start, you need to create a MetaMask Web3 provider in Casdoor.

Name	Description
Category	Choose <input type="text" value="Web3"/>
Type	Choose <input type="text" value="MetaMask"/>

Edit Provider Save **Save & Exit**

Name [?]:

Display name [?]:

Organization [?]:

Category [?]:

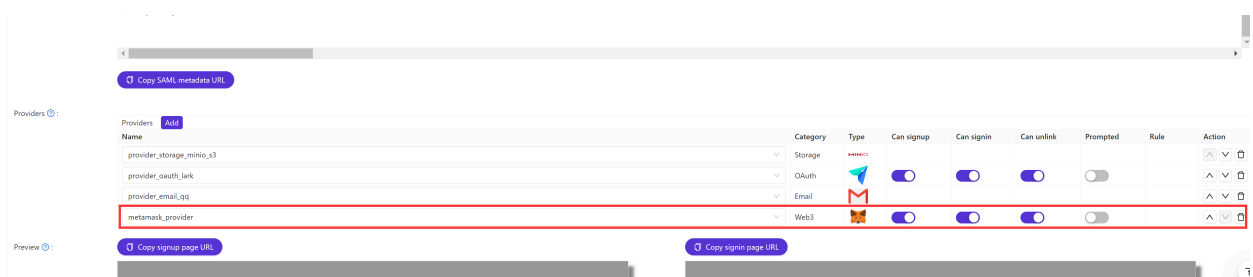
Type [?]:

Provider URL [?]:

Save **Save & Exit**

Step 2: Add the provider to your application

Next, add the MetaMask Web3 provider to your application.

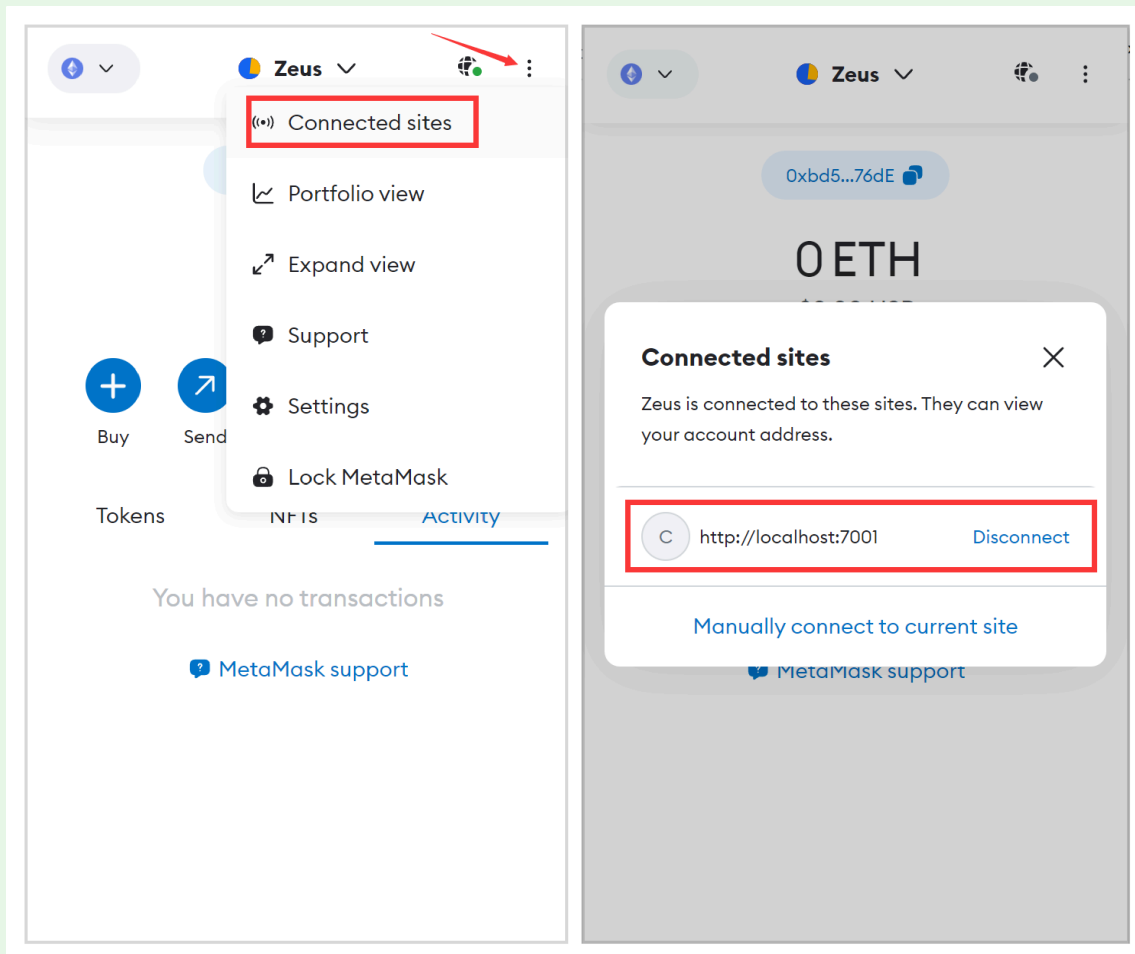


Step 3: Login with MetaMask

You can now log in with MetaMask. Here is a demo video.

💡 TIP

1. When logging in with MetaMask, please authorize only one Ethereum address. Casdoor will only bind one Ethereum address per user.
2. If you want to switch to another Ethereum address for login, please disconnect the connection between the current Ethereum address and Casdoor first.

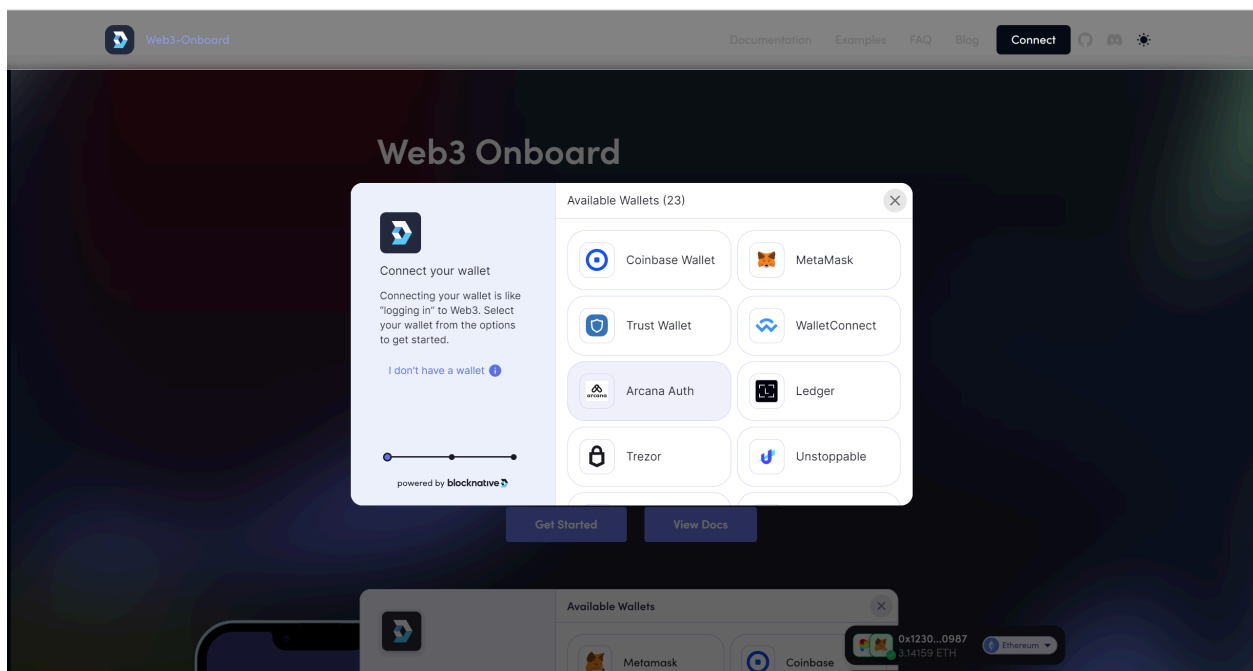


Web3-Onboard

NOTE

This is an example of how to configure **Web3-Onboard** as a Web3 provider.

Web3-Onboard can help users use different wallets for Web3 login. Casdoor allows using Web3-Onboard as an identity provider and enables Web3 login with Web3-Onboard.



Step 1: Create a Web3-Onboard Web3 provider

First, you need to create a Web3-Onboard Web3 provider in Casdoor.

Name	Description
Category	Choose Web3
Type	Choose Web3-Onboard
Wallets	Choose the wallets that are allowed to log in

Edit Provider Save Save & Exit

Name ⓘ :

Display name ⓘ :

Organization ⓘ :

Category ⓘ :

Type ⓘ :

Wallets ⓘ : Injected Coinbase Trust Gnosis Sequence Taho Frontier Infinity Wallet

Provider URL ⓘ :




















Save Save & Exit

Currently, Casdoor only supports the wallets shown in the image above. The **Injected** wallets represent browser-injected wallets such as **MetaMask** or **Coinbase**.

Step 2: Add the provider to your application

Second, add the Web3-Onboard Web3 provider to your application.

Providers :

Name	Category	Type	Can signup	Can signin	Can unlink	Prompted	Rule	Action
provider_storage_minio_s3	Storage							 
provider_oauth_lark	OAuth		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		 
provider_email qq	Email							 
provider_web3_metamask	Web3		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		 
provider_google_oauth	OAuth		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	One Tap 	 
provider_web3_onboard	Web3		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		 

Step 3: Login with Web3-Onboard

Now you can log in through Web3-Onboard. Here is a demo video.

Resources

 **Overview**




Upload resources in Casdoor

Overview

You can upload resources in Casdoor. Before uploading resources, you need to configure a storage provider. Please refer to the [Storage Provider](#) section for more information.

Once you have configured at least one storage provider and added it to your application, you can proceed.

Providers ⓘ :

Name	Category	Type	ca
Provider_azure	Storage		
Github_1	OAuth		
provider_Alipay	Payment		

Great! Now let's take a look at an example of how to **upload** and **delete** resources.

Uploading Resources


Users can upload various types of resources, such as files and images, to the [cloud storage](#) that you have configured.

Resources [Upload a file...](#)

Provider	Source	Created time	Tag
provider_storage_aliyun_oss	source/casbin/leo220yuyaodog/2022_ICM_Problem_D.pdf	2022-05-18 17:25:21	custom
provider_storage_aliyun_oss	source/built-in/admin/美的2021&22Q1交流.pdf	2022-05-18 12:28:01	custom
provider_storage_aliyun_oss	source/casbin/admin/solo.svg	2022-05-17 16:25:39	custom

Deleting Resources

If you no longer need a particular resource, you can choose to delete it by clicking the "Delete" button.

Created time	Tag	Type	Format	File size	Preview	URL	Action
2022-05-19 23:16:55	custom	image	.jpg	70.3 KB		Copy Link	Delete

Products

 **Products**

Add products that you want to sell

 **Payment**

View the transaction information of the products in Payment

Products

You can add the product (or service) you want to sell. The following will guide you through the process of adding a product.

Configuring Product Attributes

First, you need to understand the basic properties of the product:

- Tag
- Detail
- Currency
- Price
- Quantity
- Sold

Tag [?] :	<input type="text" value="Casdoor Summit 2022"/>
Detail [?] :	<input type="text" value="This is a description"/>
Currency [?] :	<input type="text" value="USD"/>
Price [?] :	<input type="text" value="19"/>
Quantity [?] :	<input type="text" value="100"/>
Sold [?] :	<input type="text" value="10"/>

Payment Provider

In addition to setting these properties, you also need to add payment providers to the product. Multiple payment providers can be added to a product.

To learn how to configure a payment provider, refer to [Payment Provider](#)

Payment providers [?] :	<input type="text" value="provider_Alipay x"/>
	<input type="text" value="provider_Alipay"/>
Return URL [?] :	<input type="text" value="http://localhost:8000/products/callback"/>



Finally, fill in the **Return URL**. This is the URL to which the payment provider page will redirect after the payment is completed.

Preview the Product

You're done! Review the details and save:

Preview :

[Test buy page..](#)


Buy Product					
Name	Product				
Detail	This is a subscription.	Tag	Casdoor Summit 2022	SKU	product
Image					
Price	\$300 (USD)	Quantity	99	Sold	10
Pay					

Payment

After the payment is successfully processed, you will be able to view the transaction information of the products in the **Payment** section. This information will include details such as the organization, user, purchase time, and product name.

Invoice

To issue an invoice, navigate to the edit screen:

Type ▾	Product ▾	Price ▾	Currency	Action
	A notebook computer	300	USD	<input type="button" value="Result"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>

On the edit screen, you will need to fill in the relevant invoice information. There are two invoice types available: `individual` and `organization`.

To complete the process, simply click on the "issue invoice" button.

Please let us know if you have any further questions or concerns.

Pricing

 **Overview**

Casdoor Pricing Overview

 **Plan**

Casdoor Plan Overview

 **Pricing Overview**

An Overview of Casdoor Pricing

 **Subscription**

Casdoor Subscription Overview

Overview

Casdoor can be used as a subscription management system through its [Plan](#), [Pricing](#), and [Subscription](#) features.

You can choose which plans to include in your price list, as shown in the pictures below:

Casdoor Pricing

Casdoor hosting services provided by Casbin Inc.

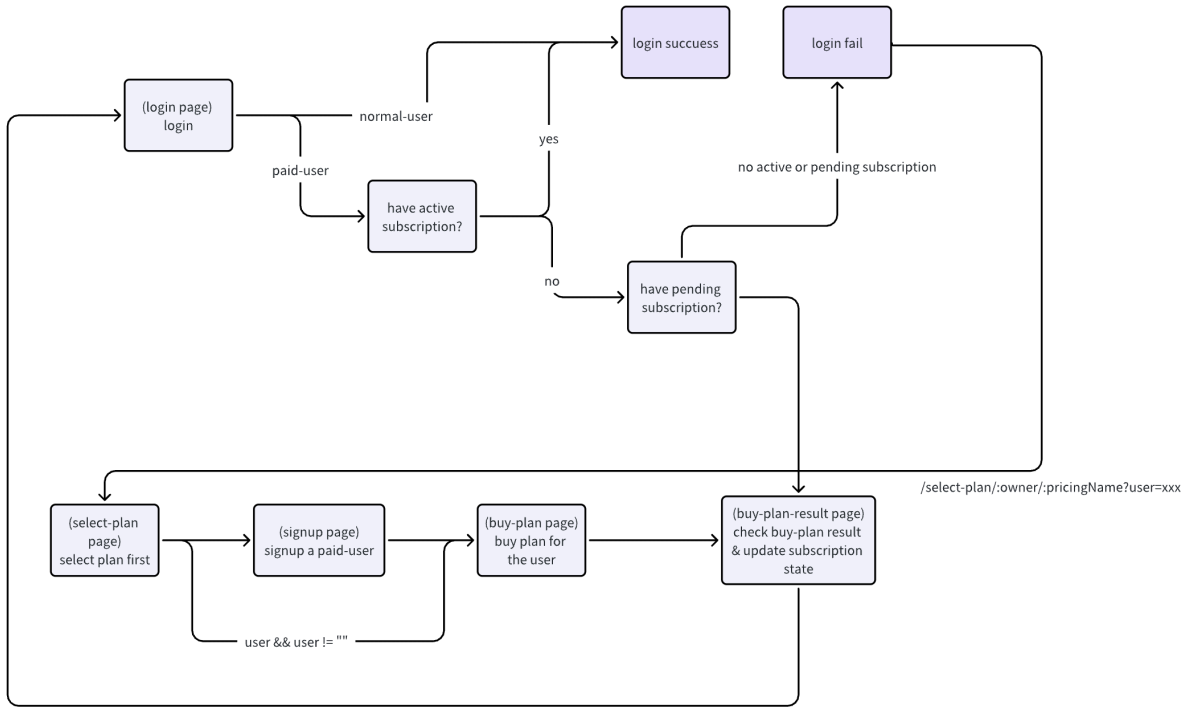
Basic Plan	Premium Plan	Enterprise Plan
\$ 10.01 per month For small teams, with limited technical support	\$ 20.02 per month For fast growing start-ups, with full technical support	\$ 30.03 per month For large & medium-sized enterprise, with full technical support
Getting started	Getting started	Getting started

Free 7-days trial available!

Each [Pricing](#) belongs to a specific [Application](#). Users can select a plan and sign up as a [paid-user](#) through the corresponding [pricing page URL](#) of the [Pricing](#).

General flow

The general flow looks like this:



1. Users enter the select-plan page of the **Pricing** by accessing the **pricing page URL** shared by the admin.

Edit Pricing
Save
Save & Exit

Organization 🔗:

Name 🔗:

Display name 🔗:

Description 🔗:

Application 🔗:

Plans 🔗: plan_basic plan_premium plan_enterprise

Trial duration 🔗:

Is enabled 🔗:

Preview 🔗: 🔗 Copy pricing page URL

Casdoor Pricing

Casdoor hosting services provided by Casbin Inc.

Basic Plan	Premium Plan	Enterprise Plan
<p>\$ 10.01 per month</p> <p style="font-size: 0.8em;">For small teams, with limited technical support</p> <p style="background-color: #4a7ebb; color: white; padding: 5px; border-radius: 3px; font-size: 0.8em;">Getting started</p>	<p>\$ 20.02 per month</p> <p style="font-size: 0.8em;">For fast growing start-ups, with full technical support</p> <p style="background-color: #4a7ebb; color: white; padding: 5px; border-radius: 3px; font-size: 0.8em;">Getting started</p>	<p>\$ 30.03 per month</p> <p style="font-size: 0.8em;">For large & medium-sized enterprise, with full technical support</p> <p style="background-color: #4a7ebb; color: white; padding: 5px; border-radius: 3px; font-size: 0.8em;">Getting started</p>


- Users select a **Plan** to subscribe and complete the signup process, becoming a **paid-user**.

Casdoor Pricing

Casdoor hosting services provided by Casbin Inc.

Basic Plan	Premium Plan	Enterprise Plan
\$ 10.01 per month <small>For small teams, with limited technical support</small>	\$ 20.02 per month <small>For fast growing start-ups, with full technical support</small>	\$ 30.03 per month <small>For large & medium-sized enterprise, with full technical support</small>
Getting started	Getting started	Getting started

Free 7-days trial available!


🌐

Username:

Display name:



* Password: 🔗 🟢

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



- After signing up, users will be redirected to the buy-plan page for the selected **Plan** to proceed with the payment.

Buy Product

Name	Auto Created Product for Plan built-in/plan_premium(Premium Plan)				
Detail	This Product was auto created for Plan built-in/plan_premium(Premium Plan)	Tag	auto_created_product_for_plan	SKU	product_6g2mcm
Image					
Price	\$20.02 (USD)	Quantity	999	Sold	0
Pay					

- Once the payment is successfully completed, the user's **Subscription** for the **Plan** is activated. Now, users can log in to Casdoor as a **paid-user**.



You have successfully completed the payment: Auto Created Product for Plan built-in/plan_premium(Premium Plan)

Please click the below button to return to the original website

[Return to Website](#)

Here is a demo video:

Plan

The `Plan` describes a list of features for an application, each with its own name and price.

The features of a `Plan` depend on the Casdoor `Role`, which comes with a set of `Permissions`.

This allows for the independent description of a `Plan`'s features, regardless of naming and pricing.

For example, a `Plan` may have different prices depending on the country or date.

The following picture illustrates the relationship between a `Plan` and a `Role`.

Plan

- Display Name
- Price per month
- ...

Role

permission 1
permission 2
...
permission N

Plan Properties

Every `Plan` has the following properties:

- `Organization`
- `Name`
- `CreatedTime`
- `DisplayName`
- `Role`
- `PricePerMonth`
- `Currency`
- `PaymentProviders`: Users can purchase the `Plan` through the Payment providers. For information on how to configure a Payment provider, see [Payment provider](#).
- `IsEnabled`

Edit Plan Save Save & Exit

Organization ⓘ :	built-in
Name ⓘ :	plan_enterprise
Display name ⓘ :	Enterprise Plan
Role ⓘ :	
Description ⓘ :	For large & medium-sized enterprise, with full technical support
Price per month ⓘ :	30.03
Price per year ⓘ :	100
Currency ⓘ :	USD
Payment providers ⓘ :	provider_payment_stripe x provider_payment_paypal x
Is enabled ⓘ :	<input checked="" type="checkbox"/>

Save Save & Exit

When a **Plan** is created through Casdoor, a related **Product** is automatically created.

The information configured for the **Plan** will be automatically synchronized to the **Product**.


When users buy a **Plan**, they are essentially purchasing the related **Product** of the selected **Plan**.

product_49faf

Auto Created Product for Plan built-in/plan_enterprise(Enterprise Plan)

built-in

URL: https://cdn.casbin.org/img/casdoor-logo_1185x256.png

Preview: 

Tag: auto_created_product_for_plan

Detail: This Product was auto created for Plan built-in/plan_enterprise(Enterprise Plan)

Description:

Currency: USD

Price: 30.03

Quantity: 999

Sold: 0




Payment providers: provider_payment_stripe x provider_payment_paypal x

Return URL: <#>

State: Published

Preview: [Test buy page](#)

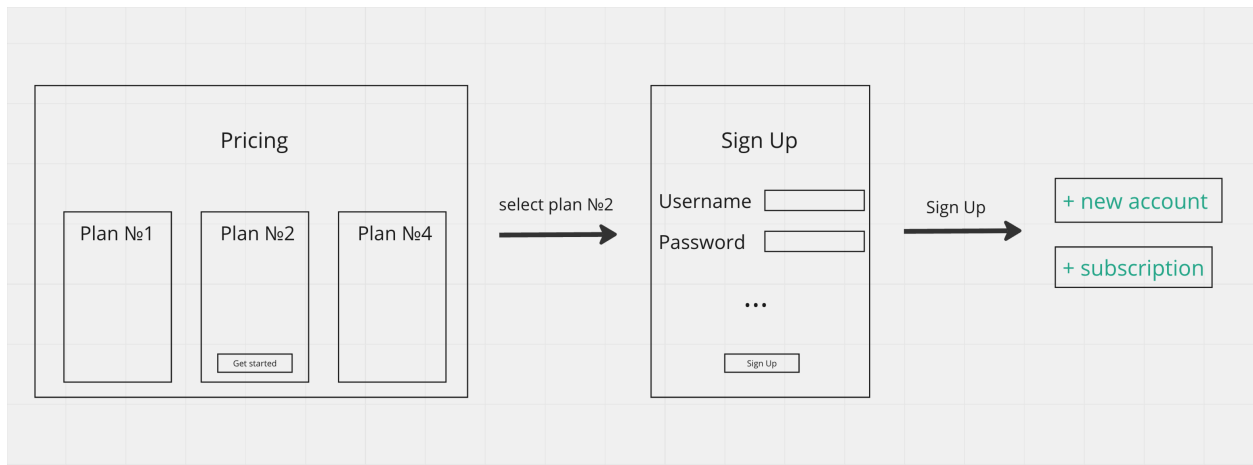
Buy Product

Name	Auto Created Product for Plan built-in/plan_enterprise(Enterprise Plan)				
Detail	This Product was auto created for Plan built-in/plan_enterprise(Enterprise Plan)	Tag	auto_created_product_for_plan	SKU	product_49faf
Image					
Price	\$30.03 (USD)	Quantity	999	Sold	0
Pay	 				

Pricing Overview

The `Pricing` feature contains one or more `Plan` options, allowing users to sign up for `Applications` at different price-points.

The general flow of pricing options is depicted in the image below:



Pricing Properties

Every `Pricing` subscription has the following properties:

- `Organization`
- `Name`
- `CreatedTime`
- `DisplayName`
- `Description`
- `Plans`: An array of Plans.
- `IsEnabled`
- `Application`

To see an example of the pricing interface, refer to the image below:

The image shows a configuration interface for pricing on the left and a preview of the pricing page on the right.

Configuration Interface:

- Organization: built-in
- Name: pricing_casdoor
- Display name: Casdoor Pricing
- Description: Casdoor hosting services provided by Casbin Inc.
- Application: app-built-in
- Plans: plan_basic x | plan_premium x | plan_enterprise x
- Trial duration: 7
- Is enabled:
- Preview: [Copy pricing page URL](#)

Pricing Page Preview:

Casdoor Pricing

Casdoor hosting services provided by Casbin Inc.

Basic Plan	Premium Plan	Enterprise Plan
\$ 10.01 per month	\$ 20.02 per month	\$ 30.03 per month
For small teams, with limited technical support	For fast growing start-ups, with full technical support	For large & medium-sized enterprise, with full technical support
Getting started	Getting started	Getting started

Subscription

The `Subscription` feature helps in managing a user's selected `Plan`, making it easy to control the access to `Application` features.

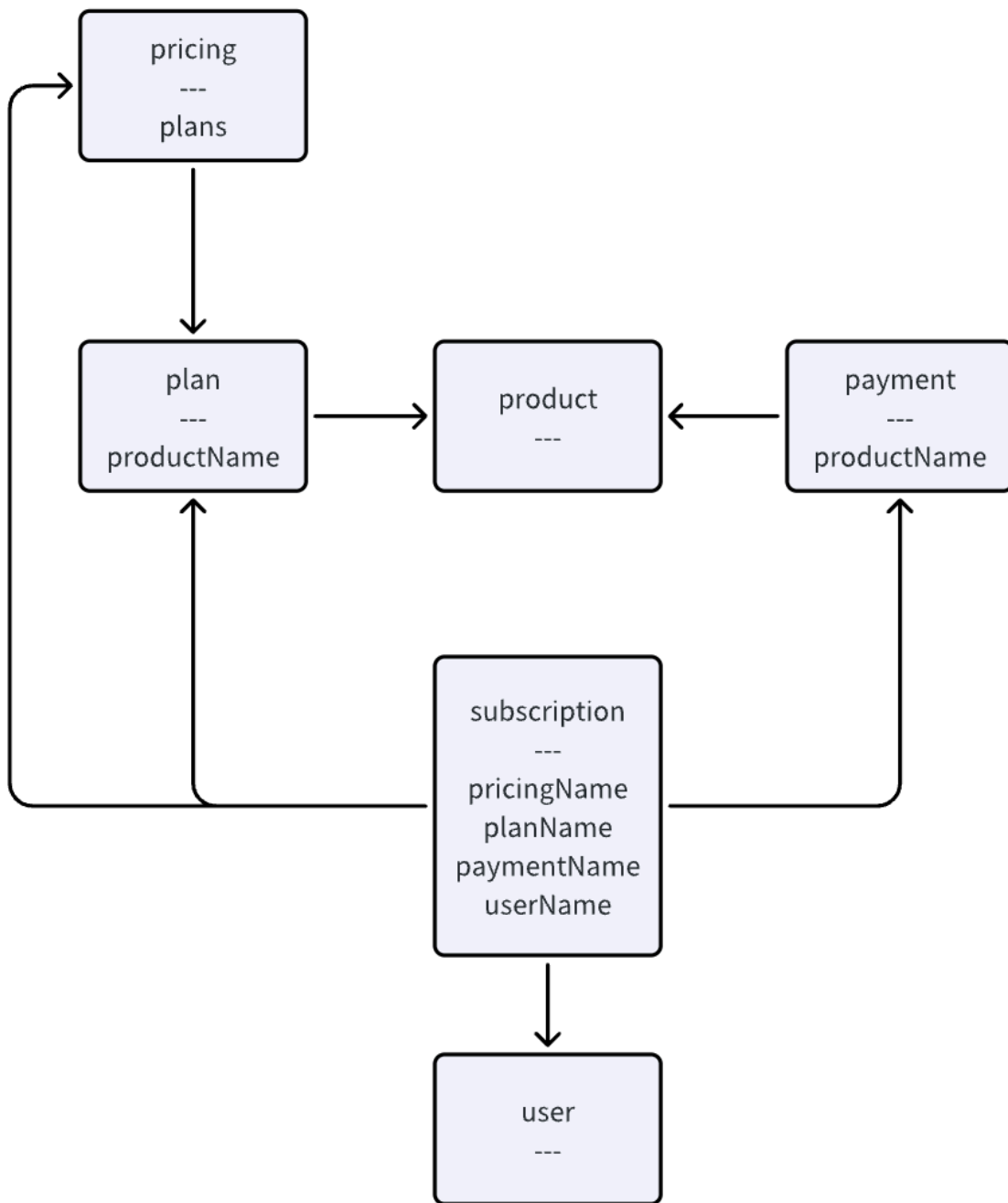


Since each `Plan` is based on a `Role`, you can assign the Plan's Role to a user and use the enforce API for permission checking.

A `Subscription` can be created in three ways:

- Manually by an admin
- Via the Pricing flow (after signing up as a `paid-user` and purchasing the selected `Plan`)
- Via API

The relationship between `Pricing`, `Plan`, `Subscription`, `Product`, and `Payment` is as follows:



Subscription properties

Every Subscription has these properties:

- `Owner`
- `Name`
- `CreatedTime`
- `DisplayName`
- `Description`
- `Duration`: The duration of the Subscription.
- `StartTime`: The starting time for the Subscription to take effect.
- `EndTime`: The end time for the Subscription to take effect.
- `Pricing`: The related Pricing.
- `Plan`: The related Plan.
- `Payment`: The related Payment.
- `User`: The user who holds this Subscription.
- `State`: Currently, the Subscription has the following states: `Pending`, `Error`, `Suspended`, `Active`, `Upcoming`, `Expired`.

Edit Subscription Save Save & Exit

Organization: built-in

Name: sub_e719e2

Display name: New Subscription - e719e2

Duration: 30

Start time: 2023-08-25

End time: 2023-09-24

User: paid-user-x

Pricing: pricing_casdoor

Plan: plan_premium

Payment: payment_20230825_160124_2d18867

Description:

State:

- Active
- Pending
- Active
- Upcoming
- Expired
- Error
- Suspended

Save Save & Exit

Users

 **Overview**

Managing Users in Casdoor

 **MFA / 2FA**

Secure your account with MFA / 2FA

 **User Roles**

Roles assigned to users

 **Permissions**

User Permissions

Overview

User Properties

As an authentication platform, Casdoor is able to manage users. Every user has the following properties:

- `Owner`: The organization that owns the user
- `Name`: The unique username
- `CreatedTime`
- `UpdatedTime`
- `Id`: Unique identifier for each user
- `Type`
- `Password`
- `PasswordSalt`
- `PasswordOptions`: Password complexity options
- `DisplayName`: Displayed in the user interface
- `FirstName`
- `LastName`
- `Avatar`: A link to the user's avatar
- `PermanentAvatar`
- `Email`
- `Phone`
- `Location`
- `Address`
- `Affiliation`
- `Title`

- IdCardType
- IdCard
- Homepage
- Bio
- Tag
- Region
- Language
- Gender
- Birthday
- Education
- Score
- Karma
- Ranking
- IsDefaultAvatar
- IsOnline
- IsAdmin: Indicates whether the user is an admin of their organization
- IsGlobalAdmin: Indicates whether the user has permission to manage the Casdoor
- IsForbidden
- IsDeleted
- SignupApplication
- Hash
- PreHash
- CreatedIp
- LastSigninTime
- LastSigninIp
- Roles: An array of the user's roles

- `Permissions`: An array of the user's permissions

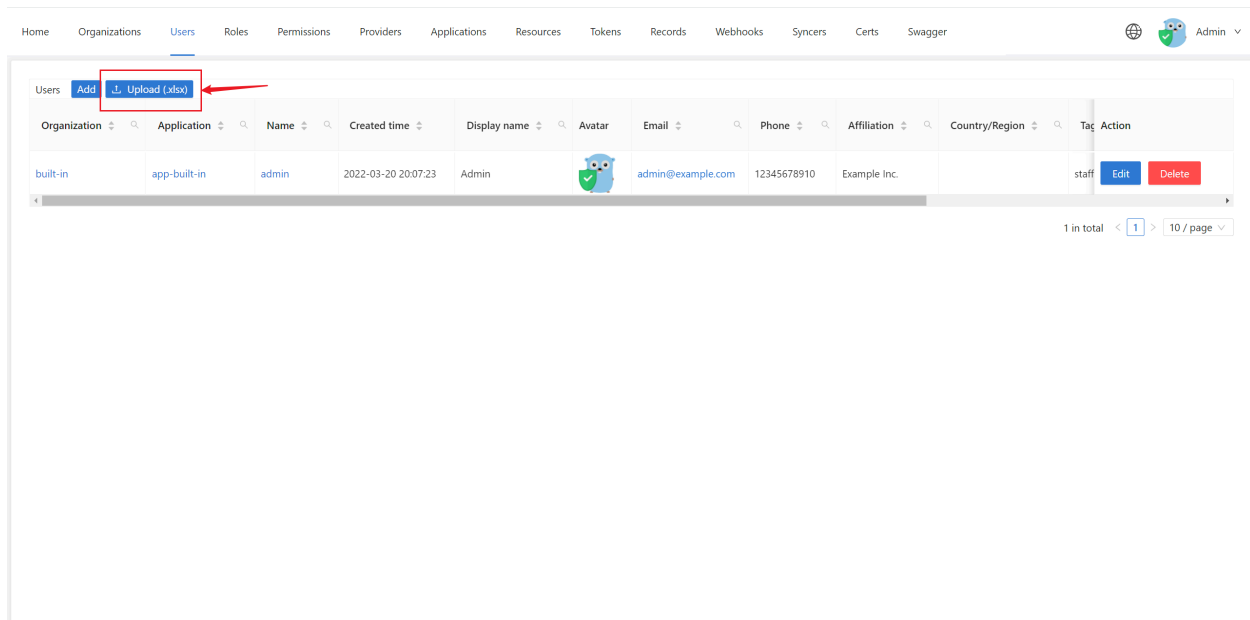
Unique IDs for social platform logins:

- `Github`
- `Google`
- `QQ`
- `WeChat`
- `Facebook`
- `DingTalk`
- `Weibo`
- `Gitee`
- `LinkedIn`
- `Wecom`
- `Lark`
- `Gitlab`
- `Adfs`
- `Baidu`
- `Casdoor`
- `Infoflow`
- `Apple`
- `Azure AD`
- `Azure AD B2C`
- `Slack`
- `Steam`
- `Ldap`
- `Properties`: A string->string map that stores any additional properties.

Importing Users from XLSX File

You can add new users or update existing Casdoor users by uploading an XLSX file containing user information.

In the Admin Console, go to Users and click the Upload (.xlsx) button.



Made with ❤️ by Casdoor

Select your XLSX file and click Open. The users will be imported.

We provide a [template XLSX file](#) named `user_test.xlsx` in the `xlsx` folder. The template includes 5 test users and headers for some required user properties.

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

Users [Add](#) [Upload \(xlsx\)](#)

user_test.xlsx

Organization	Application	Name	Created time	Display name	Avatar	Email	Phone	Affiliation	Country/Region	Tag	Action
built-in	app-built-in	tesla	2022-03-20 20:49:03	Nikola Tesla		9v73hn@example.com	40738134827	Example Inc.	United States of America	scier	Edit Delete
built-in	app-built-in	gauss	2022-03-20 20:48:33	Carl Friedrich Gauss		vqdsan@example.com	98621462844	Example Inc.	Germany	math	Edit Delete
built-in	app-built-in	galileo	2022-03-20 20:47:58	Galileo Galilei		8p4f38@example.com	22596937332	Example Inc.	Italy	scier	Edit Delete
built-in	app-built-in	euler	2022-03-20 20:47:08	Leonhard Euler		3dzwj@example.com	74409642681	Example Inc.	Switzerland	math	Edit Delete
built-in	app-built-in	einstein	2022-03-20 20:46:29	Albert Einstein		z6mive@example.com	60062541396	Example Inc.	Germany	scier	Edit Delete
built-in	app-built-in	admin	2022-03-20 20:07:23	Admin		admin@example.com	12345678910	Example Inc.		staff	Edit Delete

6 in total < 1 > 10 / page

Made with ❤️ by Casdoor

Bypass password encryption

When migrating users from an external database to Casdoor, there might be situations where you want to bypass or control the default encryption method provided by `organization` default Password type method.

This can be achieved by using the `passwordType` field during user import.

i USER WITH BYCRYPT PASSWORD

Below is an example of a POST body request for the API route `/api/add-user`.

```
{
  "owner": "organization",
  "signupApplication": "first-app",
```

Here, the user's password is already encrypted using the bcrypt algorithm, so we specify the `passwordType` as "bcrypt" to inform Casdoor not to encrypt it again.

MFA / 2FA

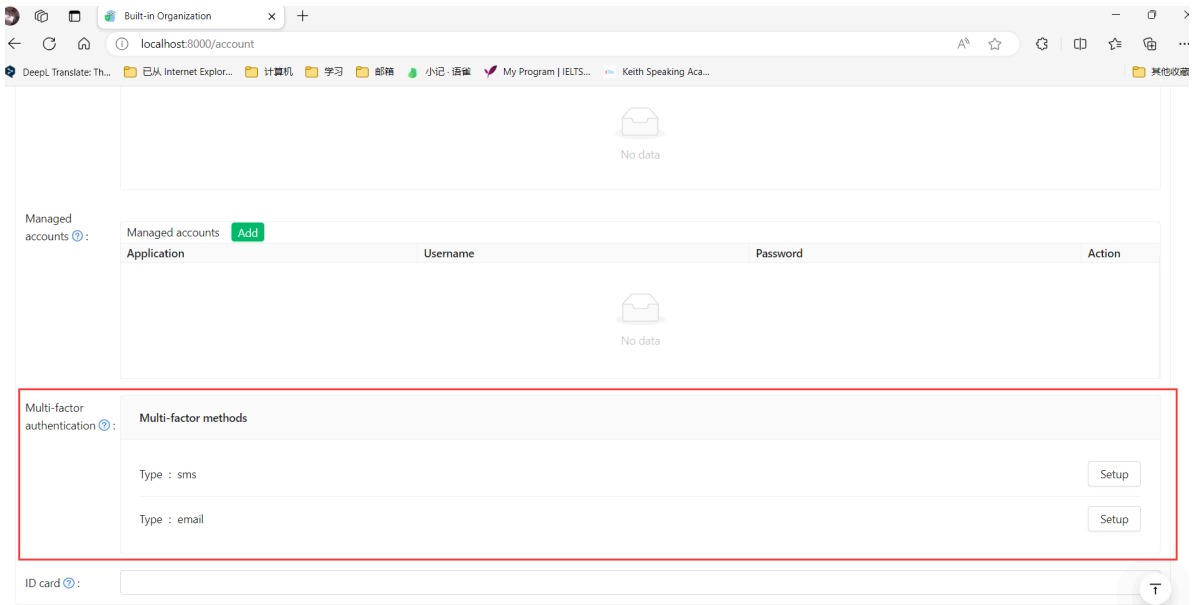
About multi-factor authentication

MFA (Multi-Factor Authentication) is a security measure that can enhance the security of users and systems. It requires users to provide two or more factors of authentication to verify their identity when logging in or performing sensitive operations.

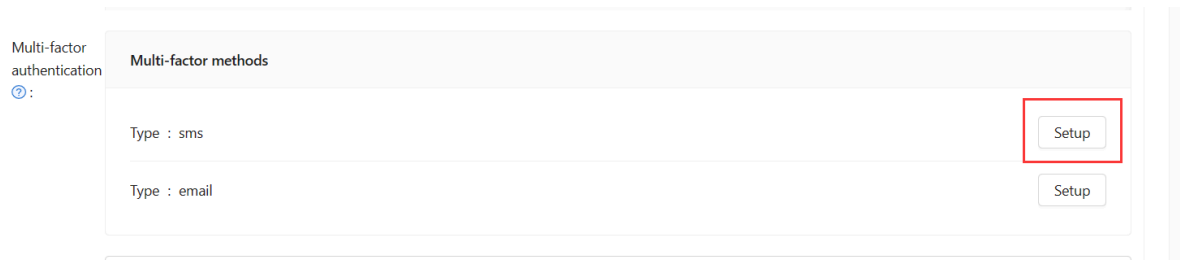
For Casdoor, the second form of authentication is a code that is sent as a text message or email. Once you enable MFA, Casdoor generates an authentication code every time someone attempts to sign in to your account. The only way someone can sign in to your account is if they know both your password and have access to the authentication code.

Configuring MFA

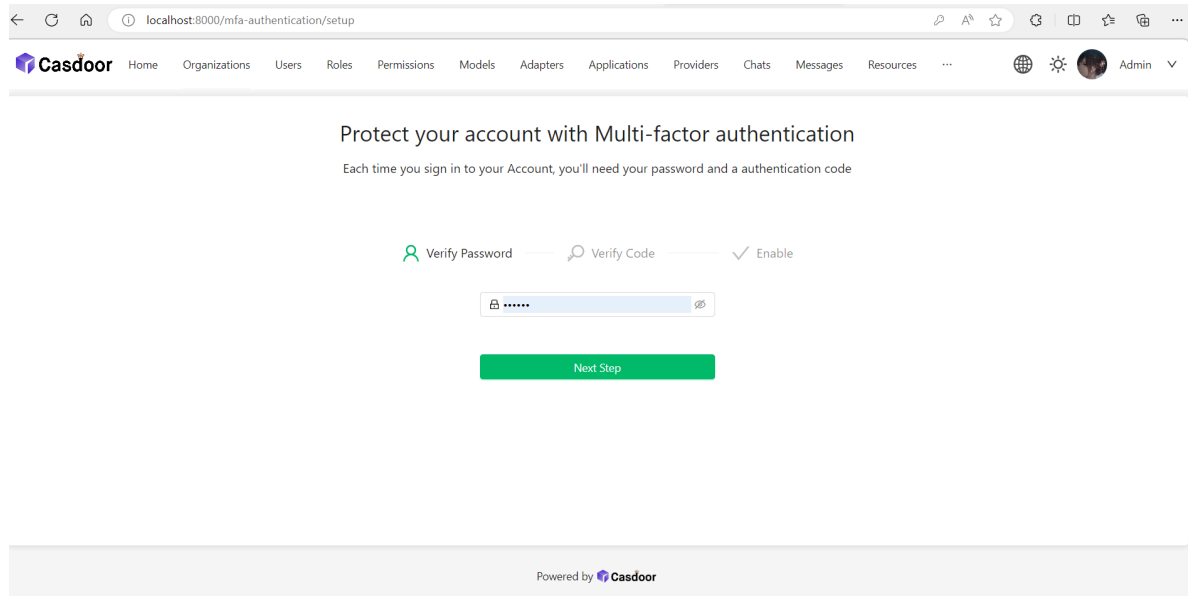
1. On the user profile page, you can see the configuration of multi-factor authentication. If you cannot see it, make sure the organization has added the multi-factor authentication item in the account items table.



2. Click the "setup" button.



3. Type your password and click "Next Step".



Configuring multi-factor authentication using a TOTP mobile app

A time-based one-time password (TOTP) application automatically generates an authentication code that changes after a certain period of time. We recommend using:

- [Google Authenticator](#)
- [Microsoft Authenticator](#).

TIP

To configure authentication via TOTP on multiple devices, during setup, scan the QR code using each device at the same time. If 2FA is already enabled, and you want to add another device, you must reconfigure your TOTP app from the user profile page.

Protect your account with Multi-factor authentication

Each time you sign in to your Account, you'll need your password and a authentication code

Verify Password — Verify Code — ✓ Enable



Scan the QR code with your authenticator app

Or copy the secret to your authenticator app

P757K7XT5MIO5RPZQYSC



Passcode

Next Step

Use email Use SMS

1. In the "Verify Code" step, do one of the following:
 - Scan the QR code with your mobile device's app. After scanning, the app displays a six-digit code that you can enter on Casdoor.
 - If you cannot scan the QR code, you can manually copy and enter the secret in your TOTP app instead.
2. The TOTP mobile application saves your account on Casdoor and generates a new authentication code every few seconds. On Casdoor, type the code into the "Passcode" field and click "Next Step".
3. Above the "Enable" button, copy your recovery codes and save them to your device. Save them to a secure location because your recovery codes can help

you regain access to your account if you lose access.

Protect your account with Multi-factor authentication

Each time you sign in to your Account, you'll need your password and a authentication code

 Verify Password —  Verify Code —  Enable

Please save this recovery code. Once your device cannot provide an authentication code, you can reset mfa authentication by this recovery code

ad30de29-3ce0-4e39-a97f-ceff1d503d3c

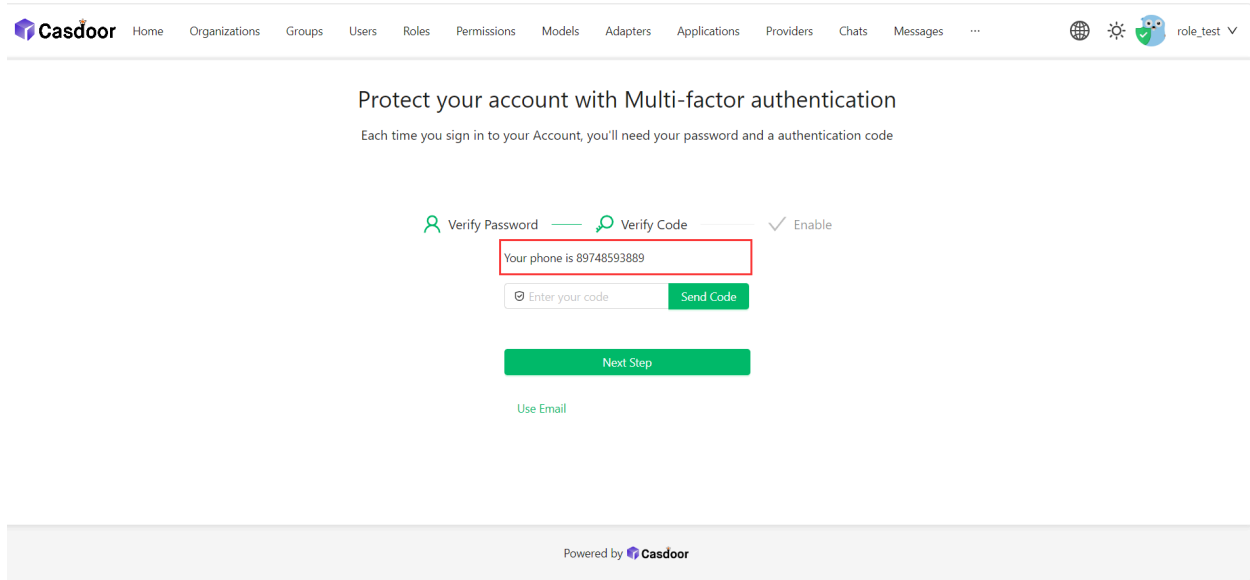
Enable

CAUTION

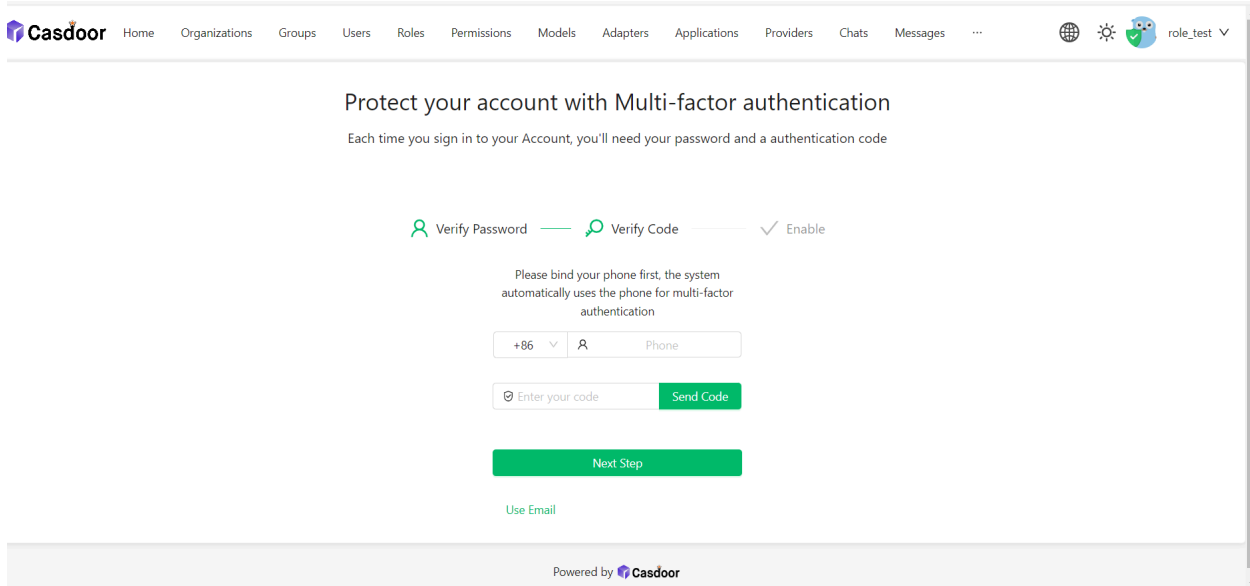
Each recovery code can only be used once. If you use a recovery code to sign in, it will become invalid.

Configuring multi-factor authentication using text messages

If you have added your mobile phone number, Casdoor will use it to send you a text message.



If you have not added your mobile phone number, you need to add it first.



1. Select your country code and enter your mobile phone number.
2. Check if your information is correct and click "Send Code".
3. You will receive a text message with a security code. Then enter the code into the "Enter your code" field and click "Next Step".

4. Above the "Enable" button, copy your recovery codes and save them to your device. Save them to a secure location because your recovery codes can help you regain access to your account if you lose access.

Configuring multi-factor authentication using email

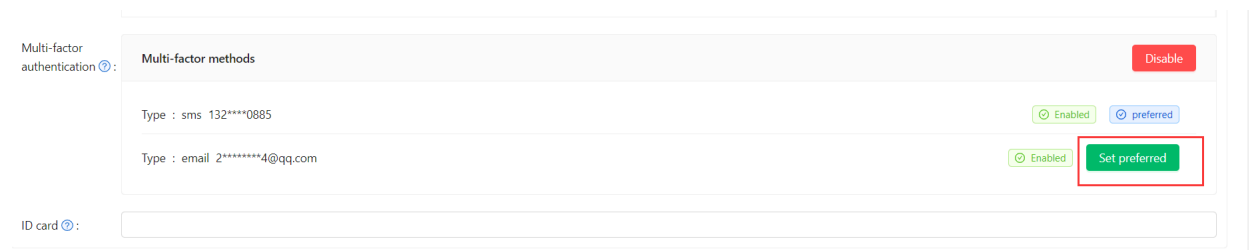
Configuring email as your multi-factor authentication method is similar to using text messages.

1. Use your current email or enter your email address and click "Send Code".
2. Then enter the code into the "Enter your code" field and click "Next Step".
3. Above the "Enable" button, copy your recovery codes and save them to your device. Save them to a secure location because your recovery codes can help you regain access to your account if you lose access.

Changing your preferred MFA method

You can add multiple MFA methods. Only the preferred method will be used when you sign in.

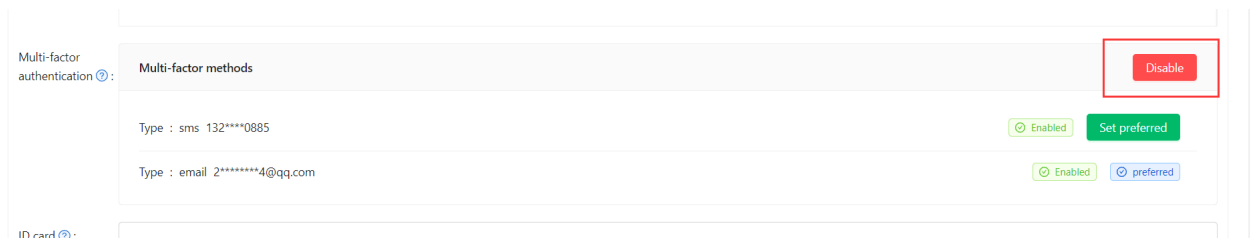
If you want to set a preferred MFA method, click the "Set preferred" button.



A "Preferred" label will be displayed on your preferred method.

Disabling multi-factor authentication

If you want to disable multi-factor authentication, click the "Disable" button. All your multi-factor authentication settings will be deleted.



User Roles

Each user can have multiple roles. You can view the roles assigned to a user on their profile.

Bio ⓘ :

Tag ⓘ :

Signup application ⓘ :

Roles ⓘ : role_test role_test2

Permissions ⓘ : permission_test

3rd-party logins ⓘ :

Is admin ⓘ :

Is global admin ⓘ :

Is forbidden ⓘ :

Is deleted ⓘ :

Role Properties

Every role has the following properties:

- `Owner`
- `Name`
- `CreatedTime`
- `DisplayName`
- `IsEnabled`
- `Users`: An array of sub users belonging to this role
- `Roles`: An array of sub roles belonging to this role

Permissions

Each user may have multiple permissions. You can view the user's permissions on their profile.

Bio ⓘ :

Tag ⓘ :

Signup application ⓘ :

Roles ⓘ : role_test role_test2

Permissions ⓘ : permission_test

3rd-party logins ⓘ :

Is admin ⓘ :

Is global admin ⓘ :

Is forbidden ⓘ :

Is deleted ⓘ :

Permission Properties

Permissions have the following properties:

- `Owner`
- `Name`
- `CreatedTime`
- `DisplayName`
- `IsEnabled`
- `Model`
- `Users`: An array of this role's sub-users
- `Roles`: An array of this role's sub-roles

- ResourceType
- Resources: An array of the resources
- Actions: An array of actions
- Effect

Invitations

 **Overview**

Managing invitations in casdoor

Overview

Currently casdoor already supports a more flexible invitation code method for user registration. Once the administrator opens the registration page with the invitation code as a mandatory option, users can only register if they have a valid invitation code.

Signup items [Add](#)

Name	Visible	Required	Prompted	Label	Placeholder	Regex	Rule
ID	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				Random
Username	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
Display name	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				None
Password	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
Confirm	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
Email	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				Normal
Phone	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				None
Agreement	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				Only signup
Invitation code	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				

There are two main ways to use invitation codes, the default added is a random string code, composed of random numbers and letters. In order to be more flexible, the invitation code also supports regular matching to match multiple different invitation codes.

Invitations [Add](#)

Name	Organization	Updated time	Display name	Code	Quota	Used count	Application	Action
invitation_794tdt	built-in	2024-02-04 20:52:15	New Invitation - 794tdt	319jed4tjk	1	0	All	Edit Delete
invitation_147y39	built-in	2024-02-04 20:47:47	New Invitation - 147y39	[a-zA-Z]2333	1	0	All	Edit Delete

2 in total < 1 > 10 / page

Invitation Properties

Casdoor manages invitations through the following properties


- Organization**: The organization that owns the invitation


- **Name**: The unique invitation name
- **Display name**: Displayed Invitation Name
- **Code**: Invitation code, you can fill in the specific invitation code string, you can also fill in the regular expression
- **Default code**: Used to populate the default invitation code in the invitation link. For randomly generated invitation codes, the default code is the same as the invitation code. For code in regular expression form, you need to fill in the default code by yourself that matches the regular expression rule in the code
- **Quota**: Maximum number of times an invitation code can be used
- **Used count**: Number of times the invitation code has been used
- **Application**: Allow applications that use this invitation code. Selecting **ALL** makes it available to all apps under the organization
- **Username**: Specific username required when registering with this invitation
- **Email**: Specific email required when registering with this invitation
- **Phone**: Specific phone required when registering with this invitation
- **State**: Status of invitation


Default Invitation


The invitation code in the default invitation is a randomly generated string of numbers and letters, and with **Quota** set to 1, it can only be used once. Application are set to **ALL** by default, which means that all apps under the organization corresponding to this invitation can use this invitation code.


New Invitation


Organization :


Name :


Display name :


Code :


Default code :


Quota :


Used count :

Application :

Username :

Email :

Phone :

State :

If the invitation code is set for a specific user and you want the user to register with the given `username`, `email`, `phone` and `invitation code`, you can restrict the user's registration by filling in the corresponding fields. If the fields are empty or if they are not configured on the registration page, casdoor does not force validation of these fields

Username :

Email :

Phone :

When it is necessary to reuse an invitation code, you can set `Quota` to a larger value, for example, if you want this invitation code to be used 10 times, then you can set `Quota` to 10. When you wish to stop registering with this invitation code, you can also do this by modifying the status of the invitation to `Suspended`.

Quota ⓘ:

Used count ⓘ:

Application ⓘ:

Username ⓘ:

Email ⓘ:

Phone ⓘ:

State ⓘ:

 CAUTION

When `username`, `email`, or `phone` is configured in the invitation, the `quota` should not be greater than one. This is because the user's `username`, `email`, and `phone` should be unique, and multiple users should not be able to register using the same `username`, `email`, or `phone`.

Regular Match Invitation

Sometimes there is a need for a large number of invitation codes for user registration, and generating invitation codes one by one can be very inefficient. Casdoor supports validating invitation codes through regular expression matching. For example, by setting the `Code` as `"[a-z]2333"`, any invitation code that matches this regular expression will be successfully matched as a valid invitation code.

Code ⓘ:

Default code ⓘ:

Quota ⓘ:

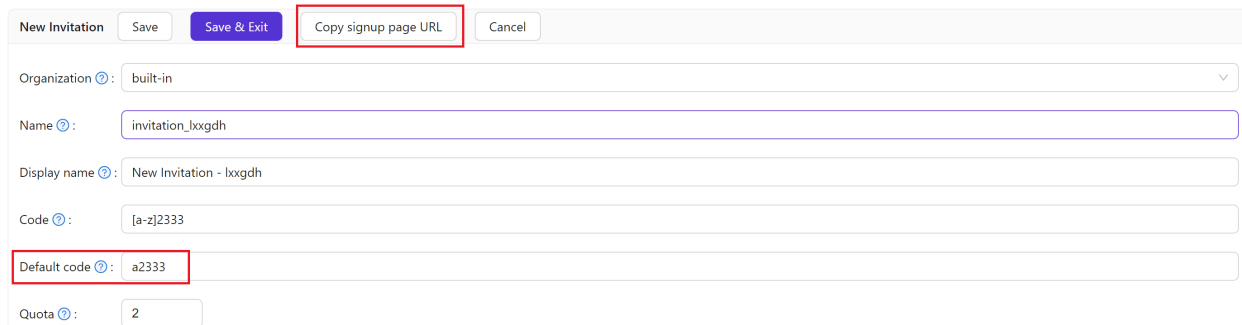
Used count ⓘ:

 NOTE

When using regular expressions to validate invitation codes, each invitation code that matches the regular expression can only be used once, and the **Quota** can still limit the number of usages. For example, when the **Code** is "[a-z]2333" and the **Quota** is 2, only a maximum of two invitation codes that match the regular expression can be successfully used.

Invitation Link

Casdoor supports copying the invitation link corresponding to an invitation. The invitation code in the invitation link corresponds to the Default code field. Therefore, for invitations that use regular expressions, the Default code must be manually filled in to generate the correct invitation link. Additionally, when registering using an invitation link, the registration page will automatically populate certain field information set by the invitation corresponding to the invitation code.



The screenshot shows the 'New Invitation' form in Casdoor. The form includes the following fields and controls:

- Buttons: 'New Invitation', 'Save', 'Save & Exit', 'Copy signup page URL' (highlighted with a red box), and 'Cancel'.
- Organization: 'built-in' (dropdown menu)
- Name: 'invitation_lxxgdh'
- Display name: 'New Invitation - lxxgdh'
- Code: '[a-z]2333'
- Default code: 'a2333' (highlighted with a red box)
- Quota: '2'



* Username:

* Password:

* Email:

* Phone:

* Invitation code:

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

Demo

Syncer

Overview

Synchronizing users in Casdoor

Database

Using Database Syncer to synchronize databases

Keycloak

Using Keycloak Syncer to synchronize Keycloak

WeCom

Using WeCom Syncer to synchronize databases

Overview

As an authentication platform, Casdoor can easily manage users stored in databases.

Syncer

Casdoor stores users in the `user` table. So, when you plan to use Casdoor as an authentication platform, there is no need to worry about migrating your application's user data into Casdoor. Casdoor provides a **syncer** to quickly help you synchronize user data to Casdoor.

You need to specify the database and user table that you want to synchronize with Casdoor, and the syncer will take care of syncing the data at regular intervals. For more details, refer to the [database syncer](#).

Synchronization hash

Casdoor uses a hash function to determine how to update a user. This hash value is calculated for each user in the table, using information such as the password or mobile phone number.

If the calculated hash value of a user with a specific `Id` changes compared to the original value, Casdoor confirms that the user table has been updated.

Subsequently, the database updates the old information, thereby achieving **bilateral synchronization** between the Casdoor user table and the original user table.

Database

Database Syncer

The users table we created as a demo is imported from the [template XLSX file](#).

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	3dzwj@
built-in	galileo	2022-03-20T20:47:58+08:00		7920eb6c-f9f5-40ef-8e18-3ac99f49bdff5	normal-user	123		Galileo Galilei			https://casbin.org	8p4f38@
built-in	gauss	2022-03-20T20:48:33+08:00		f0c288f6-2c0d-479b-b545-cb4c9f6db36	normal-user	123		Carl Friedrich Gau			https://casbin.org	vqdsan@
built-in	tesla	2022-03-20T20:49:03+08:00		687c3068-fd21-4d32-b2ba-e13e8b369a	normal-user	123		Nikola Tesla			https://casbin.org	9v73hn@

To create a new syncer, go to the Syncers tab and fill in all the required information as shown below. Then, save the changes.

Organization: built-in

Name: syncer_qmpox9

Type: Database

Host: localhost

Port: 3306

User: root

Password: password

Database type: MySQL

Database: auth

Table: user

Table columns:

Column name	Column type	Casdoor column	Is key	Is hashed	Action
name	string	Name	<input type="checkbox"/>	<input checked="" type="checkbox"/>	↑ ↓ 🗑
id	string	Id	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	↑ ↓ 🗑
first_name	string	FirstName	<input type="checkbox"/>	<input checked="" type="checkbox"/>	↑ ↓ 🗑



TIP

In general, you need to fill in at least the **ID** and **Name** in the Casdoor columns. Other important fields include **createdTime**, **Password**, and **DisplayName**.

The following fields are required:

- **Organization**: The organization that the user will be imported to
- **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 such as MySQL, PostgreSQL, SQL Server, Oracle, and SQLite
- **Database**: The original database name
- **Table**: The original user table name
- **Table columns**
- **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 this option is enabled, the syncer will only synchronize the user if the field of the user in the origin table is updated. If this option is disabled, the syncer will still synchronize the user even if only the field is updated. In short, the user will not be synchronized until the fields involved in the hash calculation (enabled "Is hashed") are updated.
- **Is key**: Whether it is the primary key of the user in the origin table and the user in the Casdoor table. When synchronizing the database, it is determined based on the field whose "Is key" option is selected. At least one of the "Is key" buttons for fields should be selected. If none are selected, the first "Is

key" option is selected by default.

- **Avatar base URL**: When syncing users, if the **Avatar base URL** is not empty and the origin `user.avatar` does not have the prefix "http", the new `user.avatar` will be replaced by **Avatar base URL + user.avatar**.
- **Affiliation table**: It is used to sync the affiliation of the user from this table in the database. Since the affiliation may be a code of type int in the "Affiliation table", it needs to be mapped to a string. Refer to [getAffiliationMap\(\)](#). Casdoor has some redundant fields to borrow, so [here](#) we use `score` to map the int code to a string name.

Once you have configured the syncer, enable the **Is enable** option and save. The syncer will start working.

Syncers [Add](#)

Name	Organization	Created time	Type	Host	Port	User	Password	Database type	Database	Action
syncer_qmpox9	built-in	2023-08-09 18:57:36	Database	localhost	3306	root	password	mysql	auth	Sync Edit Delete

You can also use the "Sync" button for database synchronization.

Update

When the **Table columns** are set as shown in the following figure, the update operation is performed.

Table columns [Add](#)

Column name	Column type	Casdoor column	Is key	Is hashed	Action
name	string	Name	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	^ v 🗑
id	string	Id	<input type="checkbox"/>	<input checked="" type="checkbox"/>	^ v 🗑
first_name	string	FirstName	<input type="checkbox"/>	<input checked="" type="checkbox"/>	^ v 🗑
password	string	Password	<input type="checkbox"/>	<input checked="" type="checkbox"/>	^ v 🗑


If the data in the two tables is different for the key, you can synchronize the data between the two tables based on the primary key.

- Update user in the original table

- Update user in the Casdoor table

Add

When the **Table columns** are set as shown in the following figure, the add operation is performed.

Table columns  [Add](#)

Column name	Column type	Casdoor column	Is key	Is hashed	Action
<input type="text" value="name"/>	string	Name	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	↑ ↓ 🗑️
<input type="text" value="id"/>	string	Id	<input type="checkbox"/>	<input checked="" type="checkbox"/>	↑ ↓ 🗑️
<input type="text" value="first_name"/>	string	FirstName	<input type="checkbox"/>	<input checked="" type="checkbox"/>	↑ ↓ 🗑️
<input type="text" value="password"/>	string	Password	<input type="checkbox"/>	<input checked="" type="checkbox"/>	↑ ↓ 🗑️

If the number of data between the two tables is different, add the data to the table with the lower number of data based on the primary key.

- Add user in the original table
- Add user in the Casdoor table

Keycloak

Keycloak Syncer

The Keycloak syncer is essentially the same as the [database syncer](#), with the added functionality of automatic configuration for Keycloak `Tables` and `Table columns`.

Furthermore, the Keycloak syncer will fetch data from the `credential` table, `keycloak_group` table, and `user_group_membership` table, as user information in Keycloak is stored across multiple tables.

Organization: built-in

Name: keycloak

Type: Keycloak

Host: localhost

Port: 3306

User: root

Password: root

Database type: MySQL

Database: keycloak

Table: user_entity **configured automatically after selecting Keycloak as syncer type**

Table primary key:

Table columns:

Column name	Column type	Casdoor column	Is hashed	Action
ID	string	Id	<input checked="" type="checkbox"/>	⬆ ⬇ ⬇
USERNAME	string	Name	<input checked="" type="checkbox"/>	⬆ ⬇ ⬇
USER_NAME	string	DisplayName	<input checked="" type="checkbox"/>	⬆ ⬇ ⬇
EMAIL	string	Email	<input checked="" type="checkbox"/>	⬆ ⬇ ⬇
EMAIL_VERIFIED	boolean	EmailVerified	<input checked="" type="checkbox"/>	⬆ ⬇ ⬇
FIRST_NAME	string	FirstName	<input checked="" type="checkbox"/>	⬆ ⬇ ⬇
LAST_NAME	string	LastName	<input checked="" type="checkbox"/>	⬆ ⬇ ⬇
CREATED_TIMESTAMP	string	CreatedTime	<input checked="" type="checkbox"/>	⬆ ⬇ ⬇
ENABLED	boolean	IsForbidden	<input checked="" type="checkbox"/>	⬆ ⬇ ⬇

WeCom

WeCom Syncer

By using WeCom syncer, you can sync WeCom user and department data to Casdoor's user table and group table.

The following fields are required:

- **Organization**: The organization that the user will be imported to
- **Name**: The syncer's name
- **Type**: Select "WeCom"
- **User**: Your WeCom Company ID
- **Password**: Your WeCom App secret
- **ClientSecret**: Your WeCom Sync of Contacts secret

Follow the steps below to configure.


Step 1: Get WeCom Syncer configuration items

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

WeCom Service Provider Console | API Documentation | CSR | Quit

Homepage Contacts Collaboration App Management Customers and Partners Advanced Features Security and Management My Company

Company Information

Company Logo  Recommended size: 702*180 [Go to verify](#)

Company short name **usher** 未认证 [Modify](#) Company not verified. After verification, the number of users can be increased.

Company address [Add](#)

Phone No. [Add](#)

Company Domain Na... [Add](#)

Company member **1 member(s)** [Statistics](#)

Company Department **1 dept(s)**

Added/Max. **1/1000** [Verify now to increase the limit](#)

Invoice Title [Add](#) Set VAT invoice titles for company members [?](#)

Industry Type **Internet and Related Services** [Modify](#)

Company Size **1-50** [Modify](#)

Creation time **2023-6-18**

Company ID **ww752595f99d89b1ca**


Already a WeCom service provider. [Go to Service Provider Platform](#)

- In your Self-build App, get **App secret**.

WeCom Service Provider Console | API Documentation | CSR | Quit


Homepage Contacts Collaboration **App Management** Customers and Partners Advanced Features Security and Management My Company

« Back **Casdoor**

 **Casdoor** [←](#) No app info Enabled

AgentId **1000003** [Edit](#)

Secret [View](#)

Allowed users 

- In Sync of Contacts Management Tool, get `Sync of Contacts secret`.

The screenshot shows the 'Sync of Contacts' configuration page in the WeCom Service Provider Console. The page has a dark blue header with the WeCom logo and navigation links: Service Provider Console, API Documentation, CSR, and Quit. Below the header is a light blue navigation bar with links: Homepage, Contacts, Collaboration, App Management, Customers and Partners, Advanced Features, Security and Management (highlighted), and My Company. The main content area is titled 'Sync of Contacts' and includes a 'Back' button. A green icon with a person symbol is next to the title. Below the title is a description: 'Companies can sync contacts through APIs or authorized third-party service providers API Documentation'. The page is divided into several sections by horizontal lines:

- Sync method:** API Sync
- Permission:** Read and edit Contacts, Edit, View Permission Details
- Secret:** View, Send again (highlighted with a red box)
- Company's Trusted IP:** 3 IP address(es) are configured. Settings. Only configured IP addresses can access company data via API.
- Set event receiving server:** The added member or department will be pushed to the following URL in the form of event to ensure the Contacts is synced. Learn More
- Disable API sync:** (link)

Step2: Config Casdoor WeCom Syncer

Go to Syncers tab, select `WeCom` type and fill in the required information as shown below. Then, save the changes.

Edit Syncer

Save

Save & Exit

Organization ⓘ: built-in

Name ⓘ: wecom_syncer

Type ⓘ: WeCom

Database type ⓘ: MySQL

Host ⓘ: 

Port ⓘ: 0

User ⓘ: ww752595f99d89b1ca Company ID

Password ⓘ:  App secret

Client secret ⓘ:  Sync of contacts secret

Database ⓘ:

Table ⓘ:

Tokens

 **Overview**

Introduction to tokens in Casdoor

Overview

Casdoor is built on OAuth and utilizes tokens as users' OAuth tokens.


The following are the available token fields in Casdoor:


- Owner
- Name
- CreatedTime
- Application
- Organization
- User
- Code
- AccessToken
- ExpireIn (Tokens will expire in hours)
- Scope (Scope of authorization)
- TokenType (e.g., Bearer type)


After logging into the application, there are three options to generate a JWT Token:


- JWT
- JWT-Empty
- JWT-Custom


The options are as follows: JWT will generate a token containing all User fields, JWT-Empty will generate a token with all non-empty values for the user, and JWT-Custom will generate a token containing custom User Token fields (you can choose attributes in the Token fields).


Token format : JWT-Custom


Token fields : Owner x CreatedTime x DisplayName x UpdatedTime x


Token expire : Owner ✓


Refresh token expire : CreatedTime ✓

Failed signin limit : UpdatedTime ✓

Failed signin frozen time : Id

Failed signin frozen time : Type

Failed signin frozen time : Password

Failed signin frozen time : PasswordSalt

Owner	✓
Name	
CreatedTime	✓
UpdatedTime	✓
Id	
Type	
Password	
PasswordSalt	

Webhooks

 **Overview**

Adding Webhooks in Casdoor

Overview

Overview

Event systems enable you to create integrations that subscribe to specific events in Casdoor. When one of these events is triggered, a JSON payload will be sent to the configured URL via a POST request. The application will parse the JSON payload and execute the specified function. Events include signup, login, logout, and user updates, all of which are stored in the action field of the record. Event systems can be used to update an external issue from users.

LDAP

Overview

Casdoor cooperates with an LDAP server

Configuring and Syncing LDAP Users

Configuring LDAP in Casdoor for user synchronization

LDAP Server

How to connect LDAP client in Casdoor

Overview

Support for an LDAP server has been introduced into Casdoor. Casdoor is able to synchronize users from LDAP servers to Casdoor in order to use them as user accounts for logging in, and authenticate them using the LDAP servers. Casdoor also supports setting up cron jobs to synchronize users automatically on a regular basis.

Details about Casdoor-LDAP synchronization mechanism

The way Casdoor cooperates with an LDAP server is described as follows:

1. Synchronization: Casdoor can connect to an LDAP server, fetch users' information, and read all users' information (including `uidNumber`, `uid`, `cn`, `gidNumber`, `mail`, `email`, `emailAddress`, `telephoneNumber`, `mobile`, `mobileTelephoneNumber`, `registeredAddress`, `postalAddress`). It then creates Casdoor accounts for each user in the LDAP, and stores these accounts in the database.
2. Authentication: As we have seen, Casdoor does not fetch LDAP users' passwords. When an account that is synchronized from the LDAP server tries to log in through Casdoor, instead of checking the password stored in Casdoor's database, Casdoor connects to the LDAP server and verifies whether the user's password is correct.
3. User identification: Casdoor uses `uid` to exclusively identify a user. Therefore, please ensure that every LDAP user has a unique `uid`.

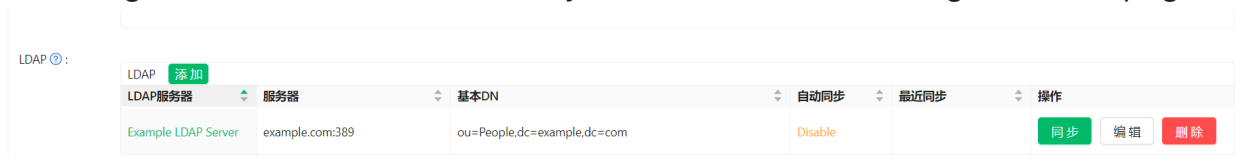
Once a user is synchronized into Casdoor, their information is independent from the LDAP user. This means that if you modify the user's information in Casdoor,

the user's information in the LDAP will not be modified, and vice versa (except for the LDAP user's password, as we rely on it to authenticate the user).

Configuring and Syncing LDAP Users

LDAP configurations are specific to each organization, as LDAP users will be synchronized into them.

To modify the configuration, you need to use a global admin user. Enter the following information for LDAP user synchronization on the "organization" page.



LDAP ⓘ:

LDAP	添加	LDAP服务器	服务器	基本DN	自动同步	最近同步	操作
Example LDAP Server		example.com:389		ou=People,dc=example,dc=com	Disable		同步 编辑 删除

Configuring Connection to LDAP Server

Edit LDAP Save Save & Exit Sync LDAP

Organization ⓘ:

ID ⓘ:

Server name ⓘ:

Server host ⓘ:

Server port ⓘ:

Enable SSL ⓘ:

Base DN ⓘ:

Search Filter ⓘ:

Filter fields ⓘ:

Admin ⓘ:

Admin Password ⓘ:

Auto Sync ⓘ: mins

Server Name

A friendly name that managers can use to identify different servers.

Example:

Server Host

The host or IP address of the LDAP server.

Example: `example.com`

Server Port

The port number of the LDAP server. Only numeric values are allowed.

Example: `389`

Base DN

Casdoor uses Sub search mode by default when searching in LDAP. The Base DN is the basic distinguished name used for the search. Casdoor will return all users under the specified Base DN.

The admin account configured in Casdoor should have at least read-only permissions at the Base DN.

Example: `ou=Example,dc=example,dc=com`

Search Filter

Casdoor uses a search filter to query LDAP users.

Example: `(objectClass=posixAccount)`

Filter Fields

Filter fields are the identifiers of the user in the LDAP server. When logging in to Casdoor as an LDAP user, the entered login username is regarded as the `uid` of the LDAP user. You can also configure other fields, such as `mail` or `mobile`.

The screenshot shows the Casdoor LDAP configuration interface. At the top, there is a navigation bar with the Casdoor logo and various menu items like Home, Organizations, Users, Roles, Permissions, Models, Adapters, Applications, Providers, Chats, Messages, Resources, and Admin. Below the navigation bar, there are three buttons: 'Edit LDAP', 'Save', and 'Save & Exit' (highlighted in green), and a 'Sync LDAP' button. The main form contains the following fields:

- Organization: built-in
- ID: 691edec0-f1ab-4e23-8f9f-a824a383032f
- Server name: Example LDAP Server
- Server host: [Redacted]
- Server port: 389
- Enable SSL: [Toggle off]
- Base DN: ou=built-in,dc=example,dc=com
- Search Filter: (objectClass=inetOrgPerson)
- Filter fields: [Empty]
- Admin: cn=admin,dc=example,dc=com
- Admin Password: [Redacted]

Admin

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

The login method (DN or ID) depends on the LDAP server settings you want to connect to.

Example: `cn=manager,dc=example,dc=com`

Admin Password

The password for the LDAP server Admin account.

Auto Sync

Set to 0 to disable auto sync. Any other value means Sync every few minutes.

Synchronizing Users

The sync table displays all the users obtained from the LDAP server within the specific **ou**. If the users have already been synced, the checkbox will be disabled. You can select the users by checking the box, and then sync the selected users from the LDAP server.

<input type="checkbox"/>	CN	UidNumber / Uid	Group Id	Email	Phone	Address
<input type="checkbox"/>	zhan san	1000 / zsan	500			
<input type="checkbox"/>	li si	1001 / lsi	500			
<input type="checkbox"/>	a dmin	1002 / admin	500			
<input type="checkbox"/>	tom brown	1007 / jery	500			
<input type="checkbox"/>	wrie jerry	1003 / wjery	500			
<input type="checkbox"/>	admin2	1004 / admin2	500			
<input type="checkbox"/>	yyyy	1005 / yyyy	500			

< 1 > 10 / page

CAUTION

If the **uid** of a user in the LDAP server is the same as the **name** of an existing user in the Casdoor organization, Casdoor will create a new user with a **name** that includes the **uid** and a random string. However, this user may not be able to log in because the name of the newly synced user does not exist in the LDAP server. Therefore, it is recommended to avoid this situation.

LDAP Server

Many systems, like `Nexus`, support LDAP authentication. Casdoor also implements a simple LDAP server, which supports bind and search operations.

This document describes how to connect to the LDAP server in Casdoor and implement simple login authentication.

LDAP Server Port

The LDAP server listens on port `389` by default. You can change the default port by modifying the `ldapServerPort` value in [conf/app.conf](#).

How it Works

Similar to the LDAP client in Casdoor, the users in the LDAP server are all subclasses of `posixAccount`.

When the server receives a set of data transmitted by the LDAP, it will parse the `cn` and `ou`, where `cn` represents the username and `ou` represents the organization name. The `dc` does not matter.

If it is a bind operation, the server will use Casdoor to verify the username and password and grant the user permission in Casdoor.

If it is a search operation, the server will check whether the search operation is legal, according to the permissions granted to the client by the bind operation, and return a response.



We only support Simple Authentication.

How to Bind

In Casdoor LDAP server, we only recognize `DN` similar to this format:

```
cn=admin,ou=built-in,dc=example,dc=com.
```

Please set the `DN` of the admin user to the above format. Then, you can use this `DN` to bind to the LDAP server with the user's password to log in to Casdoor for verification. If the server verification is successful, the user will be granted authority in Casdoor.

How to Search

Once the bind operation completes successfully, you can perform the search operation. There are some differences between search and bind operations.

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

RADIUS

 **Overview**

Use Casdoor as RADIUS server

Overview

You can use Casdoor as a RADIUS server. RADIUS is a client/server protocol, the client can be a NAS or any computer running RADIUS client software.

Congigure

Before deploying Casdoor, you need to modify the RADIUS-related configurations in the `conf/app.conf` file, including the server port and secret:

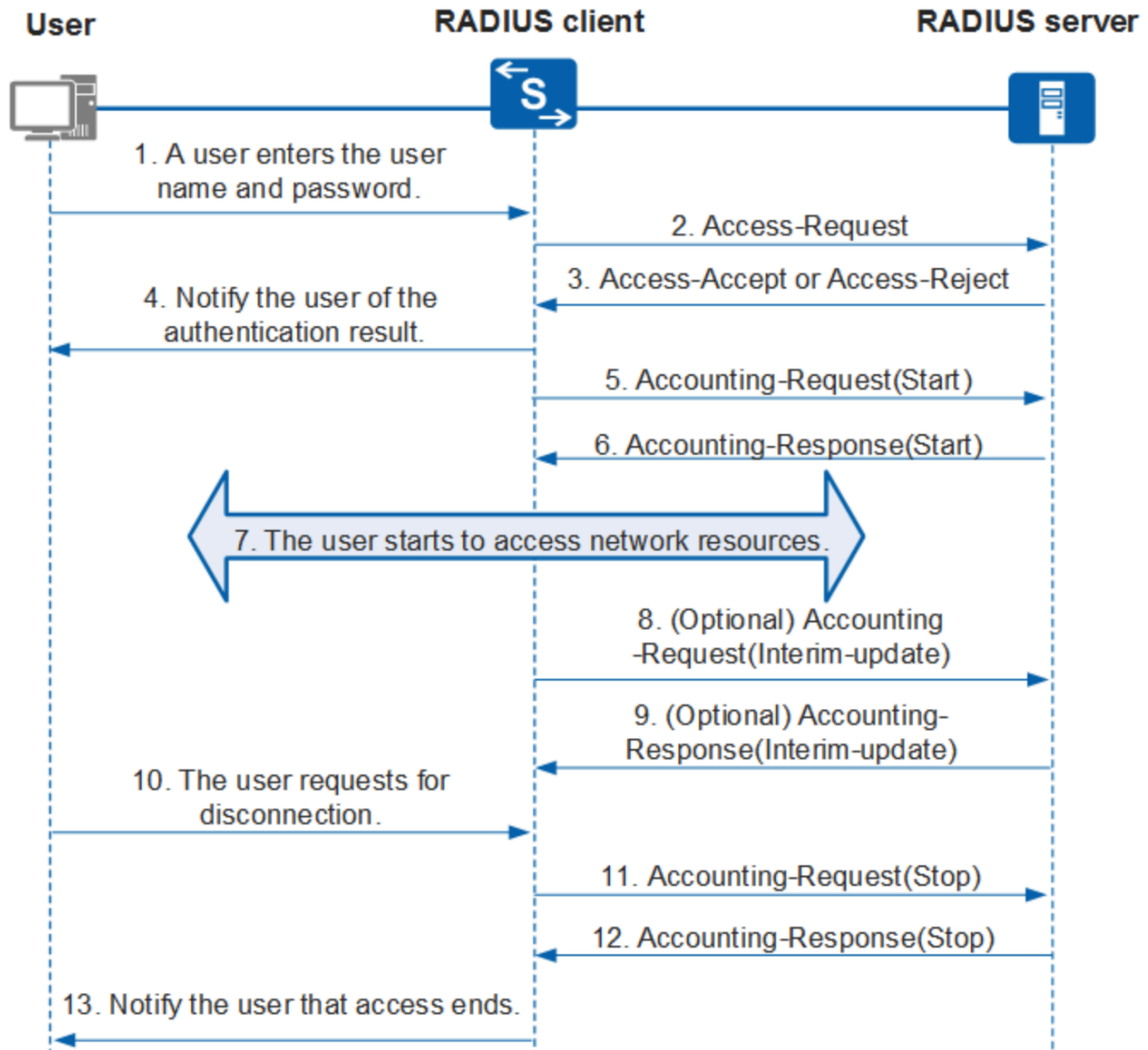
```
radiusServerPort = 1812
radiusSecret = "secret"
```

Now you can use Casdoor as RADIUS server.

Use Casdoor as RADIUS server

Casdoor currently can support follow standard RADIUS request:

- `Access-Request` : The authentication request message is sent by the RADIUS client to the Casdoor. Casdoor determines whether to allow access based on the user information carried in the message and reply with `Access-Reject` or `Access-Accept` .
- `Accounting-Request` : When a user starts or stops accessing network resources, the RADIUS client will send accounting request (Start/Interim-update/Stop) message to Casdoor. Casdoor will record relevant accounting request message and reply with `Accounting-Response` .



Since Casdoor use Organization to manage User, where each User belongs to a specific Organization, the `Class` attribute in the request needs to be set as the User's Organization.

NTRadPing Test Utility

RADIUS Server/port: localhost 1812

Reply timeout (sec.): 3 Retries: 6

RADIUS Secret key: secret

User-Name: admin

Password: **** CHAP



Request type: Authentication Request 0

Additional RADIUS Attributes:
Class=built-in

Acct-Output-Octets 42

Add Remove Clear list Load... Save...

NTRadPing 1.5 - RADIUS Server Testing Tool
? 1999-2003 Master Soft SpA - Italy - All rights reserved
<http://www.dialways.com/>

RADIUS Server reply:

```
Sending authentication request to server localhost:1812
Transmitting packet, code=1 id=0 length=55
received response from the server in 16 milliseconds
reply packet code=2 id=0 length=20
response: Access-Accept
----- attribute dump -----
```

Send Help... Close

SCIM

Overview

Use Casdoor as SCIM service provider

Overview

The SCIM protocol is a HTTP-based protocol for provisioning and managing identity data specified through SCIM schemas. You can use Casdoor as a SCIM service provider.

Use Casdoor as SCIM service provider

Currently Casdoor only support `User Resource Schema`, you can manage users through SCIM User operations. You can interact with the Casdoor through the following endpoints:

Endpoint	Method	Description
<code>/scim/ServiceProviderConfig</code>	GET	Provide details about the features of the SCIM standard that are supported, for example, the resources that are supported.
<code>/scim/Schemas</code>	GET	Provide details about the service provider schemas.
<code>/scim/ResourceTypes</code>	GET	Specify metadata about each resource.
<code>/scim/Users/:id</code>	GET	Retrieve a user with resource identifier <code>id</code> .
<code>/scim/Users</code>	GET	Query users with query parameters

Endpoint	Method	Description
		(currently only support <code>startIndex</code> and <code>count</code>).
<code>/scim/Users</code>	POST	Create a user.
<code>/scim/Users/:id</code>	PUT	Update a user with resource identifier <code>id</code> .
<code>/scim/Users/:id</code>	PATCH	Modify a user with resource identifier <code>id</code> by PATCH operation.
<code>/scim/Users/:id</code>	DEL	Delete a user with resource identifier <code>id</code> .

For more details, please refer to [rfc7644](#).

User Resource

Casdoor implements the mapping between `User Resource Schema` (SCIM) and `User` (Casdoor). The mapping relationship between attributes is as follows:

User Resource Schema (SCIM)	User (Casdoor)
<code>id</code>	<code>Id</code>
<code>meta.created</code>	<code>CreatedTime</code>
<code>meta.lastModified</code>	<code>UpdatedTime</code>

User Resource Schema (SCIM)	User (Casdoor)
meta.version	UpdateTime
externalId	ExternalId
userName	Name
password	Password
displayName	DisplayName
profileUrl	Homepage
userType	Type
name.givenName	FirstName
name.familyName	LastName
emails[0].value	Email
phoneNumbers[0].value	Phone
photos[0].value	Avatar
addresses[0].locality	Location
addresses[0].region	Region
addresses[0].country	CountryCode

Since Casdoor use Organization to manage User, where each User belongs to a specific Organization, the `organization` attribute should be passed in `Enterprise User Schema Extension` (identified by the schema URI `urn:ietf:params:scim:schemas:extension:enterprise:2.0:User`). Here is a User Resource Schema SCIM representation in JSON format:

```
{
  "active": true,
  "addresses": [
    {
      "country": "CN",
      "locality": "Shanghai",
      "region": "CN"
    }
  ],
  "displayName": "Bob~",
  "emails": [
    {
      "value": "test1@casdoor.com"
    }
  ],
  "externalId": "1234123543234234",
  "id": "ceacbc6-40d0-48f1-af23-0990232d570a",
  "meta": {
    "resourceType": "User",
    "created": "2023-10-08T23:51:55+08:00",
    "lastModified": "2023-10-12T20:38:49+08:00",
    "location": "Users/ceacbc6-40d0-48f1-af23-0990232d570a",
    "version": "2023-10-12T20:38:49+08:00"
  },
  "name": {
    "familyName": "bob",
    "formatted": "alice bob",
    "givenName": "alice"
  },
  "nickName": "Bob~",
  "phoneNumbers": [
```


Integrations

 C++

3 items

 C#

1 items

 Go

9 items

 Java

17 items

 JavaScript

2 items

 Lua

1 items

 PHP

4 items

 Ruby

1 items

 Haskell

1 items

 Python

1 items

C++

Nginx

Using Casdoor with Nginx

NginxCommunityVersion

Using Casdoor with Nginx (Not Nginx-Plus) and Oauth2-Proxy

Envoy

Using Casdoor in Envoy

Nginx

Enable OpenID Connect-based single sign-on for applications proxied by NGINX Plus using Casdoor as the identity provider (IdP).

This guide explains how to enable single sign-on (SSO) for applications that are being proxied by NGINX Plus. The solution uses OpenID Connect as the authentication mechanism, with [Casdoor](#) as the identity provider (IdP), and NGINX Plus as the relying party.

See Also: You can find more information about the NGINX Plus OpenID Connect integration in the project's GitHub repository.

Prerequisites

The instructions assume that you have the following:

- A running Casdoor server. Refer to the Casdoor documentation for [Server Installation](#) and [Try with Docker](#).
- An NGINX Plus subscription and NGINX Plus R15 or later. For installation instructions, see the [NGINX Plus Admin Guide](#).
- The [NGINX JavaScript module](#), which is required for handling the interaction between NGINX Plus and the IdP. After installing NGINX Plus, install the module using the appropriate command for your operating system.

For Debian and Ubuntu:

```
sudo apt install nginx-plus-module-njs
```

For CentOS, RHEL, and Oracle Linux:

```
sudo yum install nginx-plus-module-njs
```

- The following directive should be included in the top-level (“main”) configuration context in `/etc/nginx/nginx.conf` in order to load the NGINX JavaScript module:

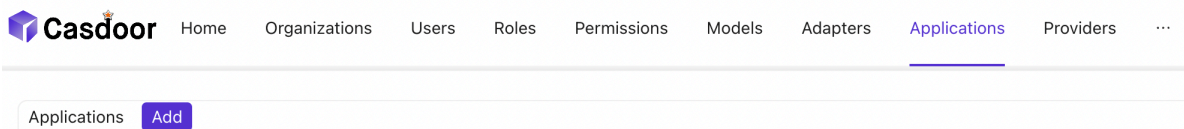
```
load_module modules/ngx_http_js_module.so;
```

Configuring Casdoor

Note: The following procedure reflects the Casdoor GUI at the time of publication, but the GUI is subject to change. Use this guide as a reference and adapt to the current Casdoor GUI as necessary.


To create a Casdoor client for NGINX Plus in the Casdoor GUI, follow these steps:

1. Log in to your Casdoor account at <http://your-casdoor-url.com/login/>.
2. In the top navigation column, click **Application**. On the **Application** page that opens, click the **Add** button in the upper left corner.




3. On the **Edit Application** page that opens, change the value in the **Name** and **Display name** fields to the name of the application for which you’re enabling SSO. Here, we’re using NGINX Plus.

Name  :

Display name  :

In the **Redirect URLs** field, type the URL of the NGINX Plus instance including the port number, and ending in `/_codexch` (in this guide it is https://your-site-url.com:443/_codexch).

Redirect URLs  :


Redirect URL
https://my-nginx.example.com:443/_codexch

Notes:

- For production, we strongly recommend that you use SSL/TLS (port 443).
 - The port number is mandatory even when you're using the default port for HTTP (80) or HTTPS (443).
4. Record the values in the **Client ID** and **Client Secret** fields. You will copy them into the NGINX Plus configuration file in [Step 4 of Configuring NGINX Plus](#).

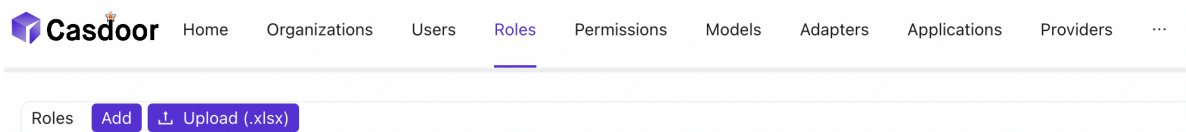
Client ID  :

200c96d5ce52789e74e0e0e0e0e0e0e0e


Client secret  :

58f13a80b875e77e0e0e0e0e0e0e0e0e

5. Click **Roles** in the top navigation column, then click the **Add** button in the upper left corner of the page that opens.



6. On the **Add** page that opens, type a value in the **Name** and **Display Name** fields (here it is nginx-casdoor-role) and click the **Save** button.

Name  :

nginx-casdoor-role

Display name

nginx-casdoor-role

 :

7. In the top navigation column, click **Users**. On the **Users** page that opens, either click **Edit** to edit one of the existing users or click the **Add** button in the upper left corner to create a new user.
8. On the **Add** page that opens, modify the **Name** and **Display Name** as you like (here it is user1).

Name  :

user1

Display name

user1

 :

Select NGINX Plus in the Signup application.

Signup

NGINX Plus

application  :


In the **Managed accounts** field, select **NGINX Plus** in **Application** and fill in the username and password.

Managed accounts  :

Application	Username	Password
NGINX Plus 	<input type="text"/>	<input type="password"/>

9. Go back to the **Roles** page and click **Edit** on the `nginx-casdoor-role` row. In the opened page, in the **Sub users** field, select the username you just created (here it is `built-in/user1`).

Sub users  :

built-in/user1 

Configuring NGINX Plus

To configure NGINX Plus as the OpenID Connect relying party, follow these steps:

1. Start by creating a clone of the [nginx-openid-connect](https://github.com/nginxinc/nginx-openid-connect) GitHub repository:

```
git clone https://github.com/nginxinc/nginx-openid-connect
```

2. Copy the following files from the clone to the `/etc/nginx/conf.d` directory:

- `frontend.conf`
- `openid_connect.js`
- `openid_connect.server_conf`
- `openid_connect_configuration.conf`

3. Retrieve the URLs for the authorization endpoint, token endpoint, and JSON Web Key (JWK) file from the Casdoor configuration. Open a terminal and execute the following `curl` command, piping the output to the indicated `python` command to generate a readable configuration format. For brevity, we have truncated the output to display only the relevant fields.

```
curl http://<casdoor-server-address>/.well-known/openid-configuration | python -m json.tool  
{
```

4. Open `/etc/nginx/conf.d/openid_connect_configuration.conf` using your preferred text editor. Modify the "default" parameter value for each of the following `map` directives with the specified values:
 - `map $host $oidc_authz_endpoint` – Use the value of the `authorization_endpoint` from [Step 3](#) (in this guide, `https://<casdoor-server-address>/login/oauth/authorize`)
 - `map $host $oidc_token_endpoint` – Use the value of the `token_endpoint` from [Step 3](#) (in this guide, `http://<casdoor-server-address>/api/login/oauth/access_token`)
 - `map $host $oidc_client` – Use the value in the **Client ID** field from [Step 4 of Configuring Casdoor](#)
 - `map $host $oidc_client_secret` – Use the value in the **Client Secret** field from [Step 2 of Configuring Casdoor](#)
 - `map $host $oidc_hmac_key` – Use a unique, long, and secure phrase

5. Configure the JWK file based on the version of NGINX Plus being used:
 - In NGINX Plus R17 and later, NGINX Plus can directly read the JWK file from the URL specified as `jwt_key_uri` in [Step 3](#). Make the following changes to `/etc/nginx/conf.d/frontend.conf`:
 - a. Comment out (or remove) the `auth_jwt_key_file` directive.
 - b. Uncomment the `auth_jwt_key_request` directive. (The parameter `jwt_key_uri` refers to the value of the `$oidc_jwt_keyfile` variable, which will be set in the next step.)
 - c. Update the "default" parameter of the `map $host $oidc_jwt_keyfile` directive to the value obtained from the `jwt_key_uri` field in [Step 3](#) (in this guide, `http://<casdoor-server-address>/.well-known/jwks`).
 - In NGINX Plus R16 and earlier, or if preferred, the JWK file must be located

on the local disk. Follow these steps:

- a. Copy the JSON contents from the JWK file specified in the `jwtks_uri` field in [Step 3](#) (in this guide, `http://<casdoor-server-address>/.well-known/jwks`) to a local file (e.g., `/etc/nginx/my_casdoor_jwk.json`).
 - b. In `/etc/nginx/conf.d/openid_connect_configuration.conf`, change the "default" parameter of the `map $host $oidc_jwt_keyfile` directive to the path of the local file.
6. Ensure that the user specified in the `user` directive within the NGINX Plus configuration file (usually `/etc/nginx/nginx.conf`) has read permissions for the JWK file.

Testing

Open a browser and enter the address of your NGINX Plus instance. Then, attempt to log in using the credentials of a user who has been assigned the NGINX Plus role.



Casdoor



 username, Email or phone

 Password



Auto sign in

[Forgot password?](#)

Sign In

No account? [sign up now](#)

Troubleshooting

Please refer to the [Troubleshooting](#) section in the [nginx-openid-connect](#) repository on GitHub.

NginxCommunityVersion

Prerequisites

This guide assumes that you have the following conditions:

- Running Casdoor service. If you haven't installed Casdoor service yet, please refer to [Server Installation](#) or [Try with Docker](#).
- Nginx open-source edition with `ngx_http_auth_request_module` module enabled at compile time. If you don't know how to enable the `ngx_http_auth_request_module` module, please refer to the [Nginx Module Document](#).
- The website on which you want to enable authentication is successfully deployed on Nginx, with a **configured domain name** (instead of using an IP address), and can be accessed normally.
- OAuth2-Proxy tool (currently, the following two popular projects with high stars are available on GitHub, and you need to choose one of them):
 1. [oauth2-proxy/oauth2-proxy](#) (used in this article) [GitHub](#) OR [Official-Website](#)
 2. [vouch/vouch-proxy](#) [GitHub](#)

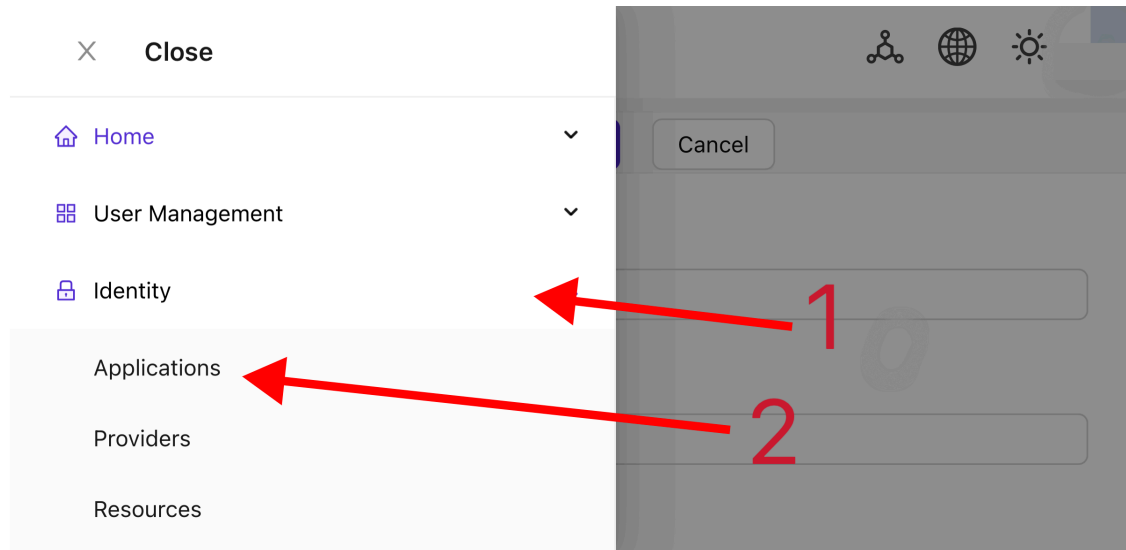
I. Configure CasDoor

Note: The operations in this article are based on the Casdoor GUI at the time of publication, but the Casdoor GUI may change depending on the version. Please follow the references provided in this article to configure your deployed Casdoor version.

Note: The keys, passwords, usernames, and other confidential information

mentioned in this article are all examples. For security reasons, you must replace them with your own relevant content when deploying.

1. Log in to your Casdoor admin account.
2. In the top bar, select "Identity Authentication" > "Applications", and then click "Add" on the "Applications" page.



3. Complete the application configuration based on your project information. In this article, we use "Nginx-Community" as the example application name.

☰ Menu 🌐 ⚙️ 📄

New Application Save Save & Exit Cancel

Name ? :

nginx-community

Display name ? :

nginx-community

Logo ? :

Set these value

- Take note of the values of the "Client ID" and "Client Secret" fields. They will be used when configuring OAuth2-Proxy later. Then configure the "Redirect URL" as `https://project.yourdomain.com/oauth2/callback/`.

Client ID ? :

811a0b0

Client secret ? :

677ea728670 [REDACTED] [REDACTED] ecae4eb09aa

Cert ? :

cert-built-in

Redirect URLs ? :

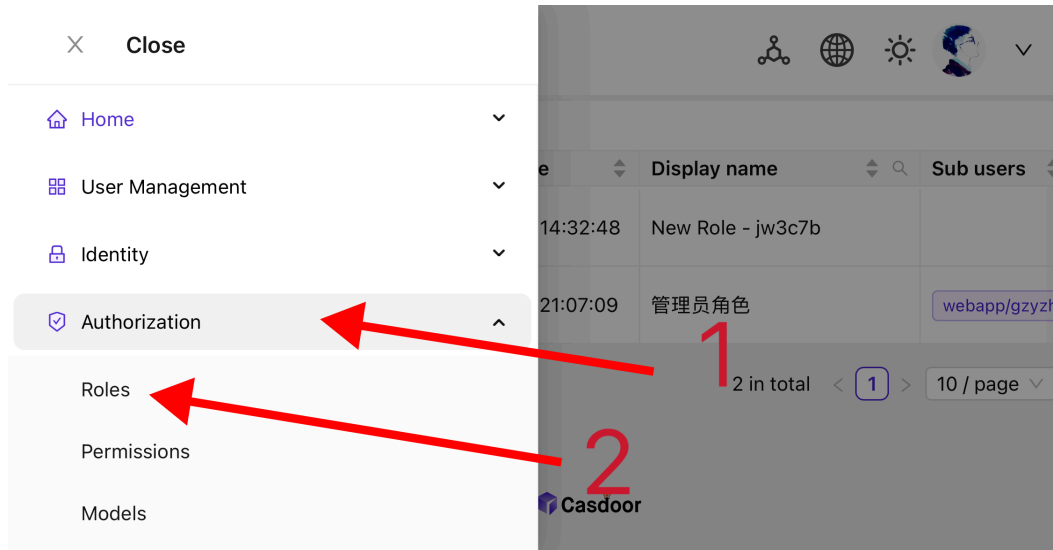
Record these information

Then, set this value to
"https://project.yourdomain.com/oauth2/callback"

Redirect URLs Add

Redirect URL	Action

- In the top bar, select "Casbin Permission Management" > "Roles", and then click "Add" on the "Roles" page.



- Complete the role configuration based on your project information. In this article, we use "nginx_role" as the example role name.

New Role

Organization ⓘ :
webapp

Name ⓘ :
nginx_community

Display name ⓘ :
nginx_community

**Set these value
Then, press "save&exit"**

- (Optional) In the top bar, select "User Management" > "Users", and then add new users based on your needs. If the users you need already exist, you can skip this step. In this article, we create an example user named "user".
- Go back to the "Roles" page mentioned in step 5, edit the `nginx_role` role, and add the users you need to the "Included Users" option. In this article, we

add the previously created `builtin/user` here.

II. Configure Oauth2-Proxy

Note: This article uses the Oauth2-Proxy project as an example. If you want to use Vouch instead of Oauth2-Proxy, please refer to their official documentation on [GitHub](#).

Note: This article assumes that your site is configured with a trusted SSL certificate and only allows HTTPS access, or that you have set up automatic redirection from HTTP visitors to HTTPS. This helps maximize the protection of cookies and prevents malicious reading of login tokens. If your site needs to be accessed via the insecure HTTP protocol, please modify the relevant commands accordingly. For more help with deploying via HTTP, please refer to the official documentation of Oauth2-Proxy on [GitHub](#).

Tips: [OAuth2-Proxy](#) provides various deployment methods (such as source code compilation, Docker installation, etc.). For ease of explanation, this article uses the "pre-built binary" for deployment.

1. Go to the [GitHub Releases](#) page and download the binary package corresponding to your operating system and CPU architecture. As of January 1, 2024, the latest release version of OAuth-Proxy is `v7.5.1`. If you want to download the binary package for this version, you can execute the following command for Linux with AMD64:

```
wget -O oauth2-proxy-linux.tar.gz https://github.com/
oauth2-proxy/oauth2-proxy/releases/download/v7.5.1/
oauth2-proxy-v7.5.1.linux-amd64.tar.gz
```

It is strongly recommended that you check the `SHA256SUM` value provided by

the official website on the [GitHub Releases](#) page after downloading the compressed package and compare it with the `SHA256SUM` value of the package you downloaded, character by character.

2. Extract the downloaded package:

```
tar -zxvf oauth2-proxy-*.tar.gz
```

3. Enter the extracted directory:

```
cd oauth2-proxy-v7.5.1.linux-amd64
```

4. Move the obtained binary file to `/usr/local/bin` and configure it with executable permissions. You may need to elevate permissions using `sudo` depending on your situation.

```
cp ./oauth2-proxy /usr/local/bin  
cd /usr/local/bin  
chmod +x ./oauth2-proxy
```

5. Test the binary installation. If the installation is successful, after executing the following command, you should see output similar to `oauth2-proxy v7.5.1 (built with go1.21.1)`.

```
cd ~  
oauth2-proxy --version
```

6. Run `oauth2-proxy` with command-line parameters. Parameters marked with `[required]` must be configured according to your specific situation, while

parameters marked with [optional] can optimize performance but can also be omitted. To ensure that oauth2-proxy can run in the background, you can use process monitoring tools like `Screen` or `Supervisor` or terminal tools.

```
oauth2-proxy \  
--provider=oidc \  
--client-id=abc123456def \  
--client-secret=abc123456def \  
--oidc-issuer-url=https://auth.yourdomain.com \  
--redirect-url=https://project.yourdomain.com/oauth2/callback \  
--scope=email+profile+groups+openid \  
--cookie-domain=project.yourdomain.com \  
--whitelist-domain=project.yourdomain.com \  
--cookie-secret=abc123456def \  
--email-domain=* \  
--insecure-oidc-allow-unverified-email=true \  
--http-address=http://127.0.0.1:65534 \  
--cookie-expire=24h0m0s
```


III. Configure Nginx

Note: Please confirm again that your Nginx has enabled the `ngx_http_auth_request_module` module when compiling and installing from source code (the compilation command includes `--with-http_auth_request_module`). If you don't know how to enable the `ngx_http_auth_request_module` module, please refer to the [Nginx Module Document](#).

Tips: Nginx installed using the Baota panel tool does not enable this module by default.

1. Open the configuration file of the website you have already deployed and want to protect, and make the following modifications:

Note: You need to adjust this configuration file according to your specific situation. Due to Nginx versions and other factors, this configuration file may not work smoothly on all Nginx instances. Please adjust the relevant content based on your own Nginx information.

```
server {
    listen 443 ssl http2;

    include /path/to/ssl.conf;

    # Add the following content
    location ^~ /oauth2/ {
        proxy_pass      http://127.0.0.1:65534; # Change this
        # to the "--http-address" configured in step II.6

        proxy_set_header Host                $host;
        proxy_set_header X-Real-IP          $remote_addr;
```

2. Save the file and reload your Nginx.

Testing

- Next, you can test your implementation.
- In normal circumstances, your users will go through the following process when logging in to your service:
- Open the URL `project.yourdomain.com` in a browser → Only see a page requiring login, including a button named "Sign in with OpenID Connect" → Click the button and be redirected to your Casdoor address, where they will be asked to log in → Users enter their username and password, and Casdoor verifies their credentials → Automatically redirect back to your URL `project.yourdomain.com` → Successfully access your service → Users will be asked to log in again when the `--cookie-expire` time you set expires.

Troubleshooting

- If your project is not running as expected, please check your Nginx configuration and Oauth2-Proxy configuration parameters for correctness.
- You can also refer to the official documentation of Oauth2-Proxy on [GitHub](#).
- If you find any errors in this document, please feel free to request edits on GitHub.

Envoy

Prerequisites

A running Casdoor server. Please refer to the Casdoor documentation for [Server Installation](#) and [Try with Docker](#).

Configuring Casdoor

1. Add the Envoy application. In the **Redirect URLs** field, enter the URL of the Envoy instance including the port number, and ending with `/oauth2/callback` (e.g., `http://%REQ(:authority)%/oauth2/callback`). Make a note of the values in the Client ID and Client Secret.
2. Add the `envoy-casdoor-role` role.
3. Add the `user1` user. Select **Envoy** in the Signup application. In the **Managed accounts** field, select **Envoy** in the Application dropdown and fill in the username and password. Go back to the **Roles** page and click "Edit" on the `envoy-casdoor-role` row. In the opened page, in the **Sub users** field, select the username you just created (in this case, it is `built-in/user1`).

Configure Envoy

1. Modify the `token_endpoint`, `authorization_endpoint`, and `client_id` in the `envoy.yaml` file.
2. Modify the `inline_string` in the `token-secret.yaml` file to the Client Secret of Envoy from Casdoor.
3. Modify the `inline_bytes` in the `hmac-secret.yaml` file with a unique, long, and secure phrase.

4. Add the `envoy.yaml`, `token-secret.yaml`, and `hmac-secret.yaml` files to your Envoy path.

How to Run

1. Start Envoy using the `envoy.yaml` file.
2. Go to the website where Envoy is listening. You should immediately be redirected to Casdoor for user authentication.

C#

Unity

Use the Casdoor-dotnet-sdk for Unity development.

Unity

Step 1: Deploy Casdoor

Firstly, Casdoor should be deployed.

You can refer to the Casdoor official documentation for the [Server Installation](#). Please deploy your Casdoor instance in production mode.

After a successful deployment, ensure that:

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

Alternatively, you can use the [official Casdoor demo station](#) for a quick start.

Step 2: Import Casdoor.Client

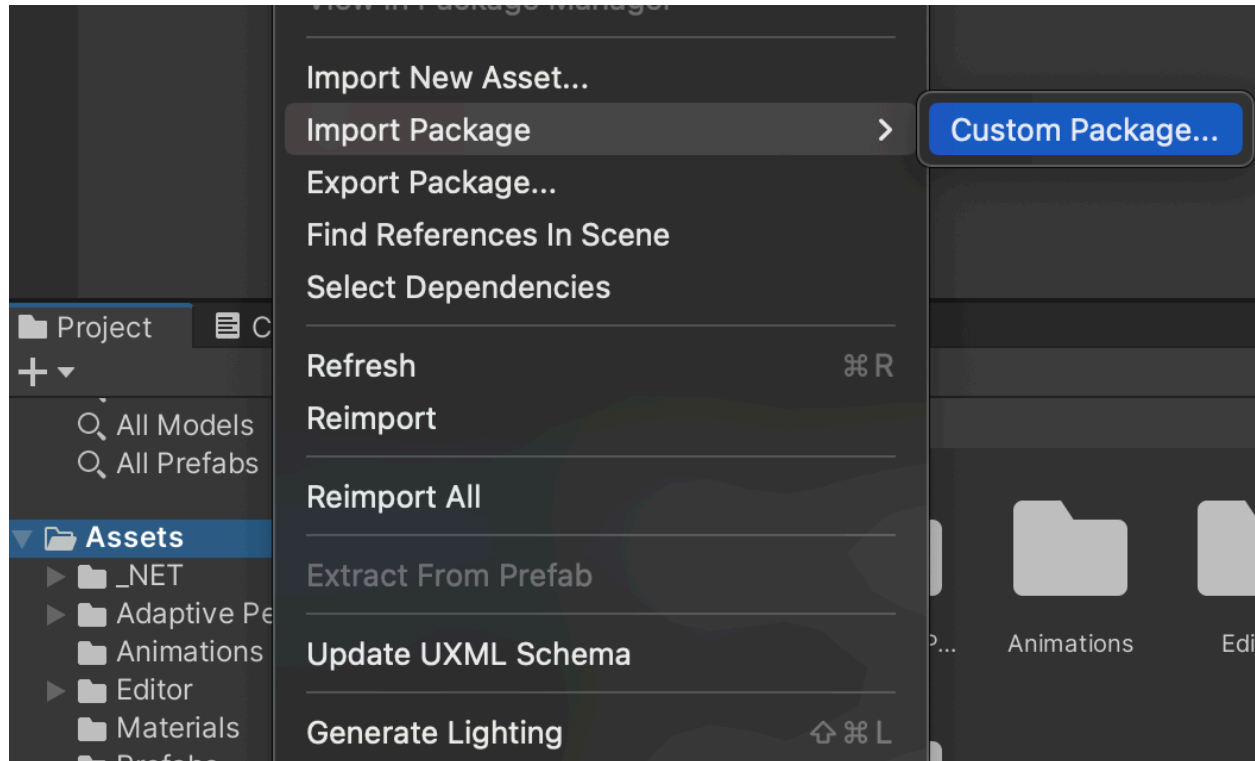
Import `Casdoor.Client` for `.NET` in the `Casdoor-dotnet-sdk`.

One optional method is as follows:

- `git@github.com:casdoor/casdoor-dotnet-sdk.git`
- Run `ConsoleApp` in the `Sample` folder.
- Get the `/casdoor-dotnet-sdk/src/Casdoor.Client/bin/Debug/net462` folder.

Now, you can import the `net462` folder into your Unity project through the method shown in the figure below. Of course, you can also choose folders of other

versions.



Step 3: Usage

Learn how to use the `Casdoor.Client` SDK for Unity 3D mobile development by looking at [casdoor-unity-example](#).

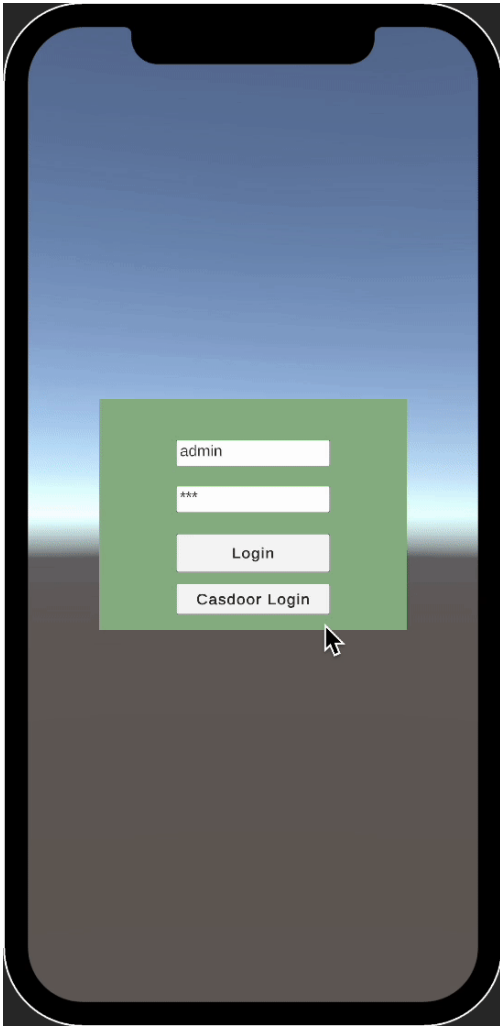
After running the `casdoor-unity-example`, you will see the following interfaces:

- Login with username and password:

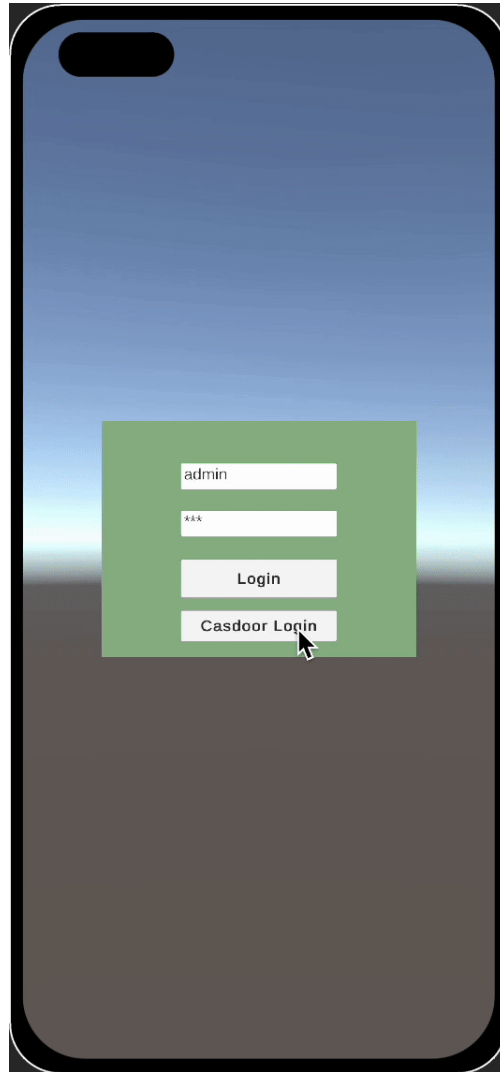


- Login with the Casdoor web page:

iOS



Android



Go

Kubernetes

Using Casdoor for Authentication in Kubernetes

OpenShift

Using Casdoor for authentication in OpenShift

BookStack

Using Casdoor for authentication in BookStack

Bytebase

Using OAuth2 to connect various applications, like Bytebase

 **ELK**

Overview of casdoor/elk-auth-casdoor

 **Gitea**

Using Casdoor for authentication in Gitea

 **Grafana**

Using Casdoor for authentication in Grafana

 **MinIO**

Configuring Casdoor as an identity provider to support MinIO

 **Portainer**

Using Casdoor for authentication in Portainer

Kubernetes

According to the [Kubernetes documentation](#), the API Server of Kubernetes can be authenticated using OpenID Connect (OIDC). This article will guide you on how to configure authentication in Kubernetes using Casdoor.

Environment Requirements

Before starting, please make sure that you have the following environment:

- A Kubernetes cluster.
- A Casdoor application like this [demo website](#).
- kubectl command tool (optional).

NOTE

Kubernetes `oidc-issuer-url` only accepts URLs which use the `https://` prefix. So your Casdoor application should be deployed on an HTTPS website.


Step 1: Creating a Casdoor App and User Account for Authentication

Go to your Casdoor application and add a new application called Kubernetes. Please remember the `Name`, `Organization`, `client ID`, `client Secret`, and add some grant types to this app.

Name ⓘ : Kubernetes

Display name ⓘ : New Application - Kubernetes

Logo ⓘ :
URL ⓘ : https://cdn.casbin.org/img/casdoor-logo_1185x256.png

Preview: 

Home ⓘ : [🔗](#)

Description ⓘ :

Organization ⓘ : casbin

Client ID ⓘ : Kubernetes

Client secret ⓘ : 72c65c3912aec24a9f3ec41b65a7577114ed2bae

Cert ⓘ : cert-built-in

Grant types ⓘ : Authorization Code × Password × ID Token × Refresh Token × Client Credentials × Token ×

Next, add a new user to the application that you just created. Please note that the `Organization` and `Signup application` used here should correspond to the app registered earlier.

Organization ? : casbin

ID ? : 202e02e9-9128-496a-a209-fdb336448f56

Name ? : user_pnvm5i

Display name ? : New User - pnvm5i

Avatar ? :

Preview:



Upload a photo...

User type ? : normal-user

Password ? : Modify password...

Email ? : pnvm5i@example.com

Phone ? : +1 78005961394

Country/Region ? : Please select country/region

Location ? :

Affiliation ? : Example Inc.

Title ? :

Homepage ? :

Bio ? :

Tag ? : staff

Signup application ? : Kubernetes

Step 2: Configure Kubernetes API Server with OIDC Authentication

To enable the OIDC plugin, you need to configure the following flags on the API server:

- `--oidc-issuer-url`: URL of the provider that allows the API server to discover public signing keys.
- `--oidc-client-id`: A client id that all tokens must be issued for.

This article uses minikube for demonstration. You can configure the OIDC plugin for the minikube's API server using the following command at startup:

```
minikube start --extra-config=apiserver.oidc-issuer-url=https://demo.casdoor.com --extra-config=apiserver.oidc-client-id=294b09fbc17f95daf2fe
```

Step 3: Test OIDC Authentication

Obtain Authentication Information

Due to the lack of a frontend in kubectl, authentication can be performed by sending a POST request to the Casdoor server. Here is the code in Python which sends a POST request to the Casdoor server and retrieves the `id_token` and `refresh_token`:

```
import requests
```


After executing this code, you should receive a response similar to the following:

```
{
  "access_token": "xxx",
  "id_token": "yyy",
  "refresh_token": "zzz",
  "token_type": "Bearer",
  "expires_in": 72000,
  "scope": ""
}
```

Now, you can use the `id_token` that you just obtained to authenticate with the Kubernetes API server.

HTTP Request-Based Authentication

Add the token to the request header.

```
curl https://www.xxx.com -k -H "Authorization: Bearer $(id_token)"
```

- `https://www.xxx.com` is the Kubernetes API server deployment address.

Kubectl Client-Based Authentication

Configuration File Method

Write the following configuration to the `~/.kube/config` file. You should replace each configuration item in the configuration file above with the values you obtained earlier.

```
users:
```

Now, you can directly access your API server using kubectl. Try running a test command.

```
kubectl cluster-info
```

Command Line Argument Method

Alternatively, you can authenticate by directly adding the `id_token` to the command line parameters of kubectl.

```
kubectl --token=$(id_token) cluster-info
```

OpenShift

OpenShift supports OIDC, so we can integrate Casdoor with OpenShift. The following steps demonstrate how to integrate Casdoor with OpenShift Local using the [online demo of Casdoor](#).

Step 1: Create an Casdoor application


Add a new application in Casdoor, noting the following points:

- Remember the `Client ID` and `Client secret` for the next step.
- The format of the Redirect URL is `https://oauth-openshift.apps.<cluster_name>.<cluster_domain>/*`. Fill it in depending on your situation.

Name ⓘ : openshift

Display name ⓘ : openshift

Logo ⓘ :
URL ⓘ : https://cdn.casbin.org/img/casdoor-logo_1185x256.png

Preview: The logo for Casdoor, featuring a blue 3D cube icon to the left of the word "Casdoor" in a bold, black, sans-serif font. A small orange star is positioned above the letter 'o' in "door".

Home ⓘ : [↗](#)

Description ⓘ :

Organization ⓘ : built-in

Client ID ⓘ : 2452f2b5abb6ff131199

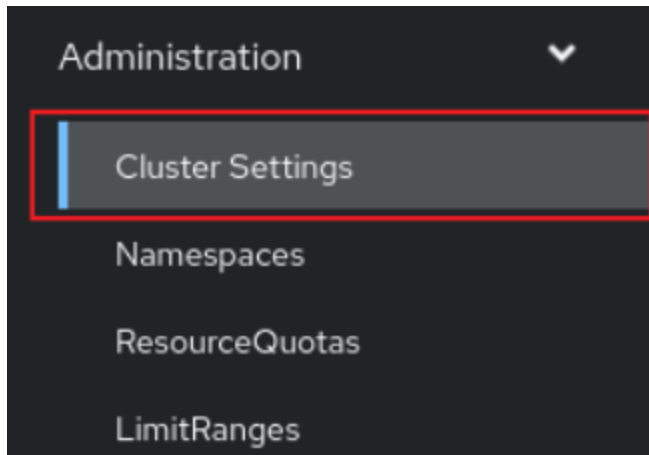
Client secret ⓘ : f7b40c97ea35bcd1c17a367c8eec373cc027ee96

Cert ⓘ : cert-built-in

Redirect URLs ⓘ :
Redirect URLs [Add](#)
Redirect URL
https://oauth-openshift.apps-crc.testing/*

Step 2: OpenShift OAuth Configuration

Now log into the OpenShift Console as Kubeadmin. Once you are logged in, browse to the side menu and locate the Cluster settings.



Under Global Configuration, you will see OAuth.

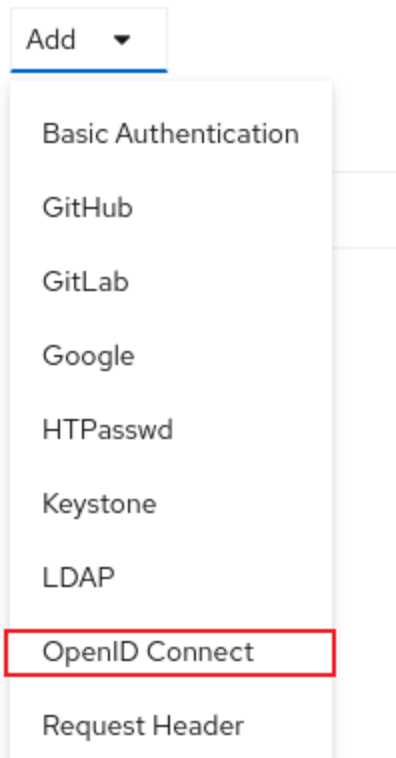
[OAuth](#)

OAuth holds cluster-wide information about OAuth. The canonical name is 'cluster'. It is used to configure the integrated OAuth server. This configuration is only honored when the top level Authentication config has type set to IntegratedOAuth. Compatibility level 1: Stable within a major release for a minimum of 12 months or 3 minor releases (whichever is longer).

You will see the Identity Provider section. In the ADD section, select OpenID Connect from the options.

Identity providers

Identity providers determine ho



Configure OIDC, noting the following points:

- Fill in the `Client ID` and `Client Secret` remembered from the previous step.
- The Issuer URL must use https, in the form `https://<casdoor-host>`, again depending on your situation.

Add Identity Provider: OpenID Connect

Integrate with an OpenID Connect identity provider using an Authorization Code Flow.

Name *

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

Client ID *

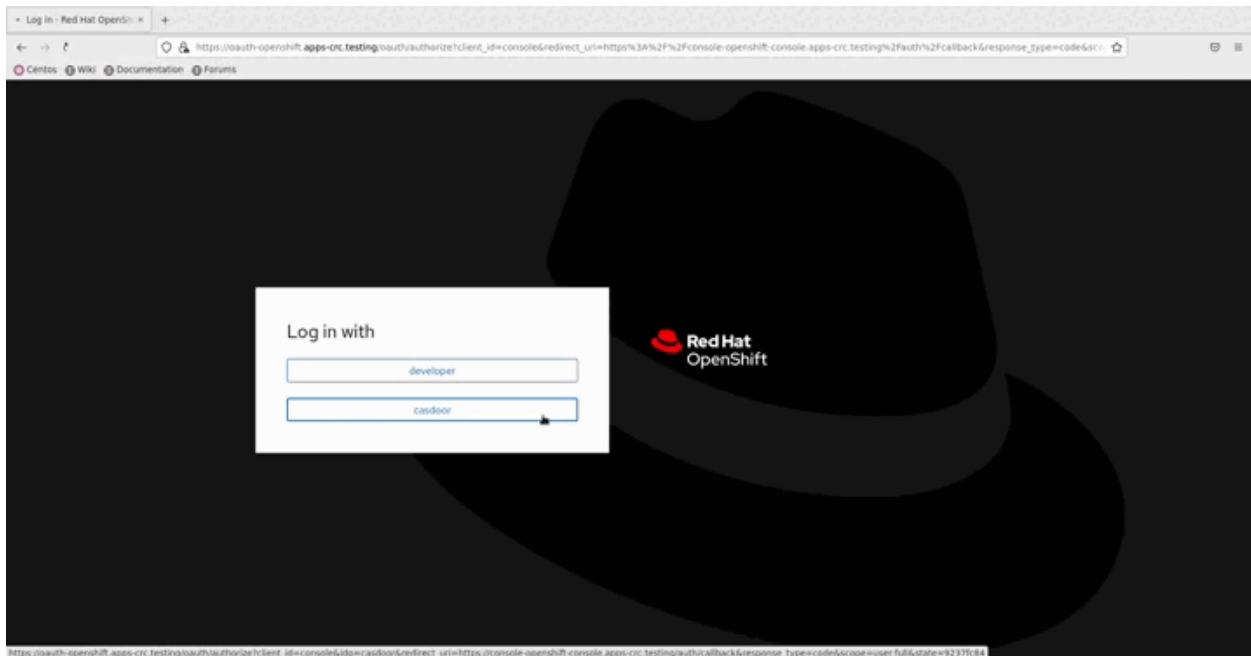
Client secret *

Issuer URL *

The URL that the OpenID provider asserts as its issuer identifier. It must use the https scheme with no URL query parameters or fragment.

Step 3: Test OIDC Authentication

Access the OpenShift console in the browser. You will see Casdoor (the name you configured in the previous step). Click on the Casdoor login option. You will be redirected to the Casdoor login page.



BookStack

Using Casdoor for authentication in BookStack

[BookStack](#) is an open-source book and document sharing site, as well as an open-source application developed using the Go language to help you better manage document reading.

BookStack-casdoor has been integrated with Casdoor, and you can now quickly get started with a simple configuration.

Step 1: Create a Casdoor application


Go to your Casdoor and add a new application called **BookStack**. Here is an example of creating the BookStack application in Casdoor.

Edit Application

Name [?](#): bookstack

Display name [?](#): bookstack

Logo [?](#):
URL [?](#): https://cdn.casdoor.com/logo/casdoor-logo_1185x256.png

Preview: 

Home [?](#): [↗](#)

Description [?](#):

Organization [?](#):

Client ID [?](#):

Client secret [?](#):

Please remember the `Name`, `Organization`, `client ID`, and `client Secret`. You will need them in the next step.

Step 2: Configure Casdoor Login

Next, navigate to BookStack and find the file `oauth.conf.example`.

Rename `oauth.conf.example` to `oauth.conf` and modify the configuration. By default, the content is as follows:

```
[oauth]
casdoorOrganization = "<Organization>"
casdoorApplication = "bookstack"
casdoorEndpoint = http://localhost:8000
clientId = <client ID>
clientSecret = <client Secret>
redirectUrl = http://localhost:8181/login/callback
```

Step 3: Fill in the `redirectUrl` in Casdoor

In the final step, go back to the page where you added the BookStack application and fill in the `Redirect URLs`. Make sure the `Redirect URL` is the same as the `redirectUrl` in the `oauth.conf` file.

Redirect URLs ⓘ :

Redirect URL
http://localhost:8181/login/callback
⌘

Now that you have completed the Casdoor configuration!

You can now go back to your BookStack and experience using Casdoor for login authentication once you have successfully deployed BookStack.

Bytebase

[Casdoor](#) can use OAuth2 to connect various applications. In this example, we will use [Bytebase](#) to demonstrate how to use OAuth2 to connect to your applications.

The following are the configuration names:

`CASD00R_HOSTNAME`: The domain name or IP address where the Casdoor server is deployed.

`Bytebase_HOSTNAME`: The domain name or IP address where Bytebase is deployed.

Step 1: Deploy Casdoor and Bytebase

Firstly, deploy [Casdoor](#) and [Bytebase](#).

After successful deployment, make sure that:

1. Casdoor can be logged in and used normally.
2. You can set `CASD00R_HOSTNAME` to `http://localhost:8000` when deploying Casdoor in `prod` mode. See [production mode](#).

Step 2: Configure Casdoor application

1. Create a new or use an existing Casdoor application.
2. Find the redirect URL: `<CASD00R_HOSTNAME>/oauth/callback`.
3. Add the redirect URL to the Casdoor application:

Client ID: e828d692244292b979e

Client secret: bab9f6c2fad67471e11bd81e074ea19d2e44f6dd

Cert: cert-built-in

Redirect URLs: <CASDOOR_HOSTNAME>/oauth/callback

On the application settings page, you will find two values: **Client ID** and **Client secret**. We will use these values in the next step.

Open your favorite browser and visit: http://<CASDOOR_HOSTNAME>/.well-known/openid-configuration. You will see the OIDC configuration of Casdoor.

Step 3: Configure Bytebase

1. Find SSO and select OAuth 2.0:

SQL Review

Risk Center

Custom Approval

Data Anonymization

Data Access Control

Audit Log

Integration

GitOps

SSO

IM

Type

OAuth 2.0 OIDC

Use template

GitHub

GitLab

Google

Custom

Basic information

Name *

Custom

Identity Provider ID: idp-custom-f4mw It cannot be changed later. [Edit](#)

Domain

2. Configure this app:

Account > SSO > casdoor

Basic information

Name *
casdoor

Identify Provider ID: ldap-casdoor-mtk

Domain
http://101.43.192.216:8000

Identity provider information

The information is provided by your identity provider.

Client ID *
e828d69224292979e

Client secret *
sensitive - write only

Auth URL *
The link to OAuth login page
http://101.43.192.216:8000/login/oauth/authorize

Scopes *
A space-separated list of scopes to be carried when accessing the Auth URL.
openid profile email

Token URL *
The API address for obtaining access token
http://101.43.192.216:8000/api/login/oauth/access_token

User information URL *
The API address for obtaining user information by access token
http://101.43.192.216:8000/api/get-account

User information mapping

Maps the field names from user info API to the Bytebase user. [Learn more](#)

name → Bytebase user identifier *

name → Bytebase user display name

email → Bytebase user email

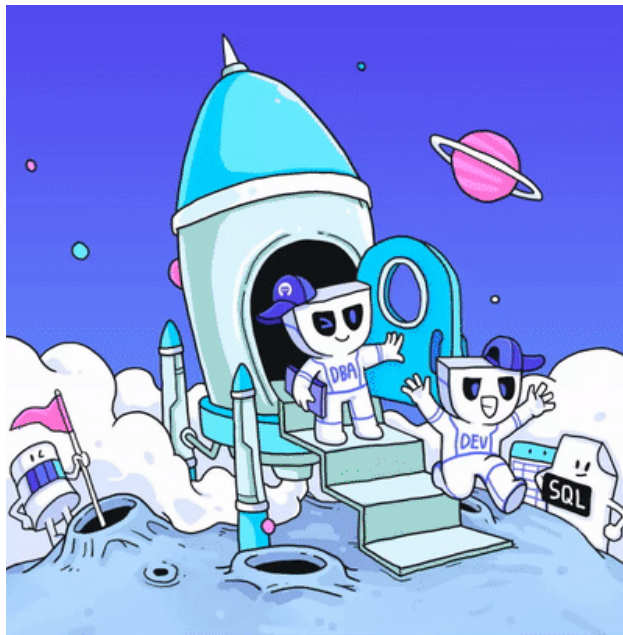
Test Connection | Archive this SSO

Discard changes | Update

3. Find the Client ID and Client Secret on the Casdoor application page.

- Token server URL: `http://CASDOOR_HOSTNAME/api/login/oauth/access_token`
- Authorization server URL: `http://CASDOOR_HOSTNAME/login/oauth/authorize`
- User Info server URL: `http://CASDOOR_HOSTNAME/api/get-account`
- Scopes: `address phone openid profile offline_access email`

Log out of Bytebase and test SSO.



Bytebase

Email *
jmi@example.com

Password * [Forgot your password?](#)

Sign in

New to Bytebase? [Sign up](#)

or

Sign in with casdoor

English 简体中文
© 2023 Bytebase. All rights reserved.

ELK

Overview of casdoor/elk-auth-casdoor

One of the biggest drawbacks of ELK (Elasticsearch, Logstash, and Kibana) is that originally these products had no authentication mechanism. As a result, anyone with the URL of Kibana or Elasticsearch could access the Kibana dashboard. Later on, ELK integrated an embedded authentication system called "Xpack." However, its advanced functions (such as OAuth, OIDC, LDAP, SAML) are not free. Only plain authentication, using a set of accounts and passwords, is available free of charge, which is quite inconvenient. This approach does not allow us to provide a unique account for everyone in a corporation.

To address this issue, we have developed an elk authentication solution based on Casdoor. This solution is free, open-source, under ongoing maintenance, and supports a wide range of advanced features. Casdoor is a centralized authentication/Single-Sign-On platform based on OAuth 2.0/OIDC. Casdoor/elk-auth-casdoor serves as a reverse proxy designed to intercept all HTTP data flow towards the ELK/Kibana stack. It guides users who haven't logged in to log in. This reverse proxy operates transparently as long as the user has logged in.

If a user hasn't been correctly authenticated, the request will be temporarily cached, and the user will be redirected to the Casdoor login page. Once the user logs in through Casdoor, the cached request will be restored and sent to Kibana. Therefore, if a POST request (or any other request type besides GET) is intercepted, the user won't need to refill the form and resend the request. The reverse proxy will remember it for you.

The casdoor/elk-auth-casdoor repository is located at <https://github.com/casdoor/elk-auth-casdoor>.

How to run it?

0. Ensure that you have the Go programming language environment installed.
1. Go to [casdoor/elk-auth-casdoor](#) and fetch the code.
2. Register your proxy as an app with Casdoor.
3. Modify the configuration.

The configuration file is located at "conf/app.conf". Here is an example, which you should customize based on your specific needs.

```
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. Visit <http://localhost:8080> (in the above example) and log in following the redirection guidance. You should then see Kibana protected and authenticated by Casdoor.

5. If everything works well, don't forget to block external access to the original Kibana port by configuring your firewall (or another method). This ensures that outsiders can only access Kibana via this reverse proxy.

Gitea

Using Casdoor for authentication in Gitea

[Gitea](#) is a community managed lightweight code hosting solution written in Go. It is published under the MIT license.

Gitea supports 3rd-party authentication including Oauth, which makes it possible to use Casdoor to authenticate it. Here is the tutorial for achieving this.

Preparations

To configure Gitea to use Casdoor as identification provider, you need to have Gitea installed as well as access to administrator account.


For more information about how to download, install and run Gitea see <https://docs.gitea.io/en-us/install-from-binary/>

You are supposed to create an administrator account during installation. If you didn't, the administrator will be the first registered user. Please use this account proceed the following procedures.

1. Create an Casdoor application

Like this

The screenshot shows the 'Edit Application' form in Casdoor. The fields are as follows:

- Name: application_9p7eai
- Display name: New Application - 9p7eai
- Logo URL: https://cdn.casbin.com/logo/logo_1024x256.png
- Preview: 
- Home: [↗](#)
- Description: (empty)
- Organization: built-in
- Client ID: 7ceb9b7fda4a9061ec1c
- Client secret: 3416238e1edf915eac08b8fe345b2b95cdba7e04
- Cert: cert-built-in
- Redirect URLs:

Redirect URL	Action
http://localhost:3000/user/oauth2/Casdoor/callback	↕ ↴ ↵

Please remember the client ID and client Secret for the next step.

Please don't fill in the callback url in this step. The url depends on the configurations on gitea in the next step. Later we will come back to set a correct callback url.

2. Configure Gitea to use Casdoor

Log in as administrator. Go to 'Site Administration' page via drop-down menu in the upper right corner. Then Switch to "Authentication Source" Page.

You are supposed to see something like this.

Issues Pull Requests Milestones Explore

Dashboard User Accounts Organizations Repositories Webhooks **Authentication Sources** User Emails Configuration System Notices Monitoring

Authentication Source Management (Total: 0) [Add Authentication Source](#)

ID	Name	Type	Enabled	Updated	Created	Edit
----	------	------	---------	---------	---------	------

Press the "Add Authentication Source" Button, and fill in the form like this.

Issues Pull Requests Milestones Explore

Dashboard User Accounts Organizations Repositories Webhooks **Authentication Sources** User Emails Configuration System Notices Monitoring

Add Authentication Source

Authentication Type * OAuth2

Authentication Name * Casdoor

OAuth2 Provider * OpenID Connect

Client ID (Key) * 7ceb9b7fda4a9061ec1c

Client Secret * 3416238e1edf915eac08b8fe345b2b95cdba7e04

Icon URL

OpenID Connect Auto Discovery URL * http://localhost:8000/well-known/openid-configuration

Skip local 2FA
Leaving unset means local users with 2FA set will still have to pass 2FA to log on

Additional Scopes

Please choose the authentication type as "oauth2".

Please input a name for this authentication source and **remember this name**. This name will be used for the `callback_url` in the next step.

Please choose the `OpenID Connect` OAuth2 Provider.

Fill in the **Client ID** and **Client Secret** remembered in the previous step.

Fill in the openid connect auto discovery url, which is supposed to be `<your endpoint of casdoor>/.well-known/openid-configuration`.

Fill in the other optional configuration items as you wish. And then submit it.

Submit the form.

3. Configure the callback url in casdoor

Go back to the application edit page in step 2, and add the following callback url:


```
<endpoint of gitea>/user/oauth2/<authentication source name>/callback
```

The `<authentication source name>` is the name for authentication source in Gitea in the previous step.

4. Have a try on Gitea

Logout the current administrator account.

You are supposed to see this in login page:

[Sign In](#)  [OpenID](#)

Sign In


Username or Email Address *

Password *

Remember this Device


[Sign In](#) [Forgot password?](#)

[Need an account? Register now.](#)

Sign In With  **OpenID**

Press the 'sign in with openid' button and you will be redirected to casdoor login page.

After login you will see this:

 [Explore](#) [Help](#) [← Sign In](#)

[Register New Account](#) [Link to Existing Account](#)

Complete New Account

Username *

Email Address *

[Complete Account](#)

Follow the instructions and bind the casdoor account with a new gitea account or

existing account.

Then everything will be working correctly.

Grafana

Using Casdoor for authentication in Grafana

[Grafana](#) supports authentication via OAuth. Therefore, it is extremely easy for users to use Casdoor to log in to Grafana. Only several steps and simple configurations are needed to achieve that.

Here is a tutorial on how to use Casdoor for authentication in Grafana. Before you proceed, please ensure that you have Grafana installed and running.

Step 1: Create an app for Grafana in Casdoor


Here is an example of creating an app in Casdoor:

Edit Application

Name

Display name

Logo

Preview: 

Home

Description

Organization

Client ID

Client secret

Cert

Redirect URLs

Redirect URL	Action
<input type="text" value="http://localhost:3000/login/generic_oauth"/>	<input type="button" value="Up"/> <input type="button" value="Down"/> <input type="button" value="Delete"/>

Please copy the client secret and client ID for the next step.

Please add the callback URL of Grafana. By default, Grafana's OAuth callback is `/login/generic_oauth`. So please concatenate this URL correctly.

Step 2: Modify the configuration of Grafana

By default, the configuration file for OAuth is located at `conf/defaults.ini` in the workdir of Grafana.

Please find the section `[auth.generic_oauth]` and modify the following fields:

```
[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 correct after signing 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 documentation](#)
2. [Grafana defaults.ini](#)

About Role Mapping

You may want to configure `role_attribute_path` to map your user's role to Grafana via [role_attribute_path](#).

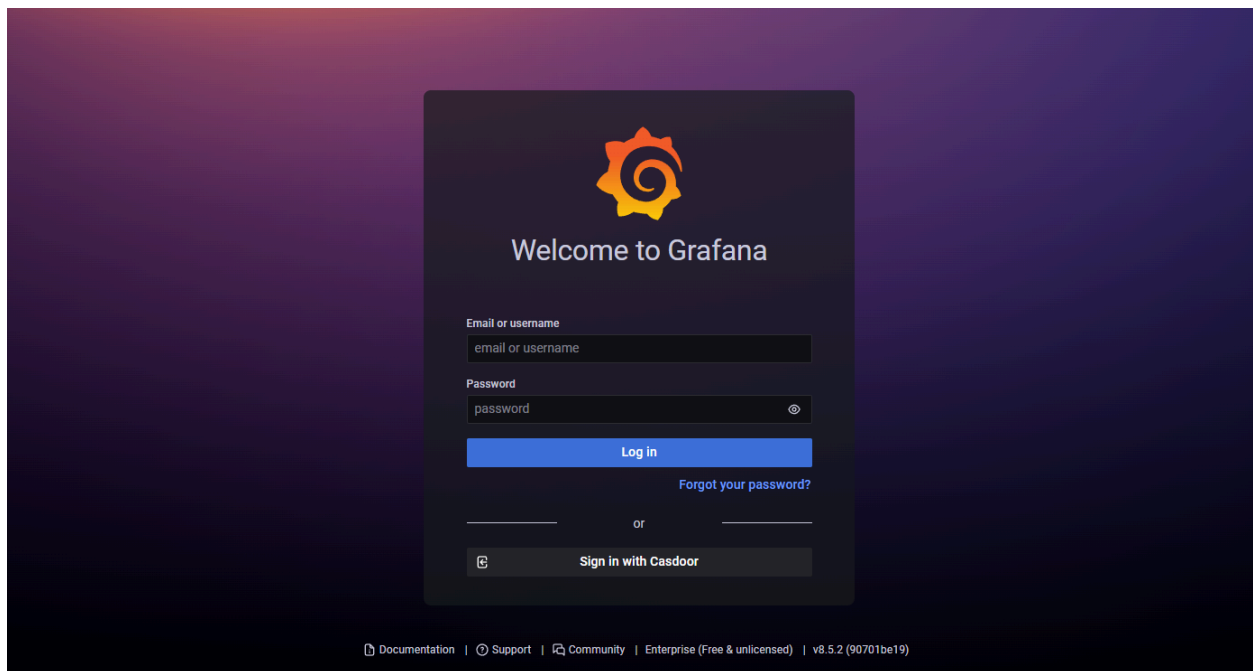
```
[auth.generic_oauth]
role_attribute_path = contains(roles[*].name, 'admin') && 'Admin'
```

The JMESPath expression after `role_attribute_path` is very important here. Please refer to the Grafana documentation.

Step 3: See if it works

Shutdown Grafana and restart it.

Go to the login page. You should see something like this:



MinIO

MinIO supports external identity management using an OpenID Connect (OIDC)-compatible provider. This document covers the configuration of Casdoor as an identity provider to support MinIO.

Step 1: Deploy Casdoor & MinIO

First, deploy Casdoor.

You can refer to the Casdoor official documentation for [Server Installation](#).

After a successful deployment, make sure that:


- The Casdoor server is running on <http://localhost:8000>.
- Open your favorite browser and visit <http://localhost:7001> to see the login page of Casdoor.
- Test the login functionality by entering `admin` and `123`.


Next, you can quickly implement a Casdoor-based login page in your own app by following these steps.


You can refer to [here](#) to deploy your MinIO server and [here](#) for the MinIO client called `mc`.

Step 2: Configure Casdoor Application

1. Create a new Casdoor application or use an existing one.
2. Add your redirect URL.

Client ID  : 24a25ea0714d92e78595 **Client ID**

Client secret  : 155... **Client Secret**

Redirect URLs  :

Redirect URLs	Add
Redirect URL	Add a redirect URL for spring security
http://localhost:8082/ui-one/login/oauth2/code/custom	

3. Add the provider you want and provide any necessary settings.

On the application settings page, you will find two values: **Client ID** and **Client secret** (as shown in the picture above). We will use these values in the next step.

Open your favorite browser and visit: http://CASDOOR_HOSTNAME/.well-known/openid-configuration to see the OIDC configuration of Casdoor.

4. This step is necessary for MinIO. As MinIO needs to use a claim attribute in JWT for its policy, you should configure it in Casdoor as well. Currently, Casdoor uses **tag** as a workaround for configuring MinIO's policy.

Tag  :

readwrite

You can find all the supported policies [here](#).

Step 3: Configure MinIO

You can start a MinIO server using the following commands:

```
export MINIO_ROOT_USER=minio
export MINIO_ROOT_PASSWORD=minio123
minio server /mnt/export
```

You can use the `--console-address` parameter to configure the address and port.

Next, add a service alias using the MinIO client `mc`.

```
mc alias set myminio <Your console address> minio minio123
```

Now, configure the OpenID Connect of MinIO. For Casdoor, the command will be:

```
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"
```

You can refer to the [official document](#) for more detailed parameters.

Once successfully set, restart the MinIO instance.


```
mc admin service restart myminio
```

Step 4: Try the demo!

Now, open your MinIO console in the browser and click on `Login with SSO`.

You will be redirected to the Casdoor user login page. Upon successful login, you will be redirected to the MinIO page and logged in automatically. You should now

see the buckets and objects that you have access to.

 CAUTION

If you deploy the frontend and backend of Casdoor on different ports, the login page you are redirected to will be on the backend port and it will display `404 not found`. You can modify the port to the frontend one. Then you can access the Casdoor login page successfully.

Portainer

Using Casdoor for authentication in Portainer

[Portainer](#) supports authentication via OAuth. Therefore, it is easy for users to use Casdoor to log in to Portainer. Only several steps and simple configurations are needed to achieve that.

Here is a tutorial on how to use Casdoor for authentication in Grafana. Before you proceed, please ensure that you have Portainer installed and running.

The following are the configuration names:

`CASD00R_HOST`: The domain name or IP address where the Casdoor server is deployed.

`PORTAINER_HOST`: The domain name or IP address where Portainer is deployed.


Step 1: Create an app for Portainer in Casdoor

Here is an example of creating an app in Casdoor:

Name ⓘ: Portainer_test

Display name ⓘ: Portainer_test

Logo ⓘ: URL ⓘ: https://cdn.casbin.org/img/casdoor-logo_1185x256.png

Preview: 

Home ⓘ: [🔗](#)

Description ⓘ:

Organization ⓘ: built-in

Tags ⓘ:

Client ID ⓘ: 2da468d1968c5f85d6b4

Client secret ⓘ: b4db599c84f978425102f161b833625faf9b6b7c

Cert ⓘ: cert-built-in

Redirect URLs ⓘ: [Add](#)

Redirect URL
🔗 https://<PORTAINER_HOST>

1. Copy the client secret and client ID for the next step.
2. Add a Redirect URL. It's your Portainer host.

Step 2: Configure Portainer

Expand the **Settings** from the left navigation bar, click on the **Authentication** option from this list.

1. Enable Use SSO and Automatic user provisioning:

portainer.io COMMUNITY EDITION

Settings > Authentication

Authentication settings

Authentication

Configuration

Session lifetime 8 hours

Changing from default is only recommended if you have additional layers of authentication in front of Portainer.

Authentication method

- Internal: Internal authentication mechanism
- LDAP: LDAP authentication
- Microsoft Active Directory: AD authentication (Pro Feature)
- OAuth: OAuth authentication

Single Sign-On

Use SSO

Hide internal authentication prompt Business Edition Feature

Automatic user provisioning

Automatic user provisioning

With automatic user provisioning enabled, Portainer will create user(s) automatically with the standard user role. If disabled, users must be created beforehand in Portainer in order to login.

The users created by the automatic provisioning feature can be added to a default team on creation. By assigning newly created users to a team, they will be able to access the environments associated to that team. This setting is optional and if not set, newly created users won't be able to access any environments.

New version available 2.19.1

2. Fill in the necessary information as follows:

Home

Environment: None selected

Settings

- Users
- Environments
- Registries
- Authentication logs
- Notifications
- Settings
 - Authentication
 - Edge Compute
 - Help / About

New version available 2.19.1

provider

- Microsoft: Microsoft OAuth provider (Pro Feature)
- Google: Google OAuth provider (Pro Feature)
- Github: Github OAuth provider (Pro Feature)
- Custom: Custom OAuth provider

OAuth Configuration

Client ID 89c7dba629b5722d4ea2

Client secret

Authorization URL https://<CASDOOR_HOST>/login/oauth/authorize

Access token URL https://<CASDOOR_HOST>/api/login/oauth/access_token

Resource URL https://<CASDOOR_HOST>/api/userinfo

Redirect URL https://<PORTAINER_HOST>

Logout URL

User identifier email

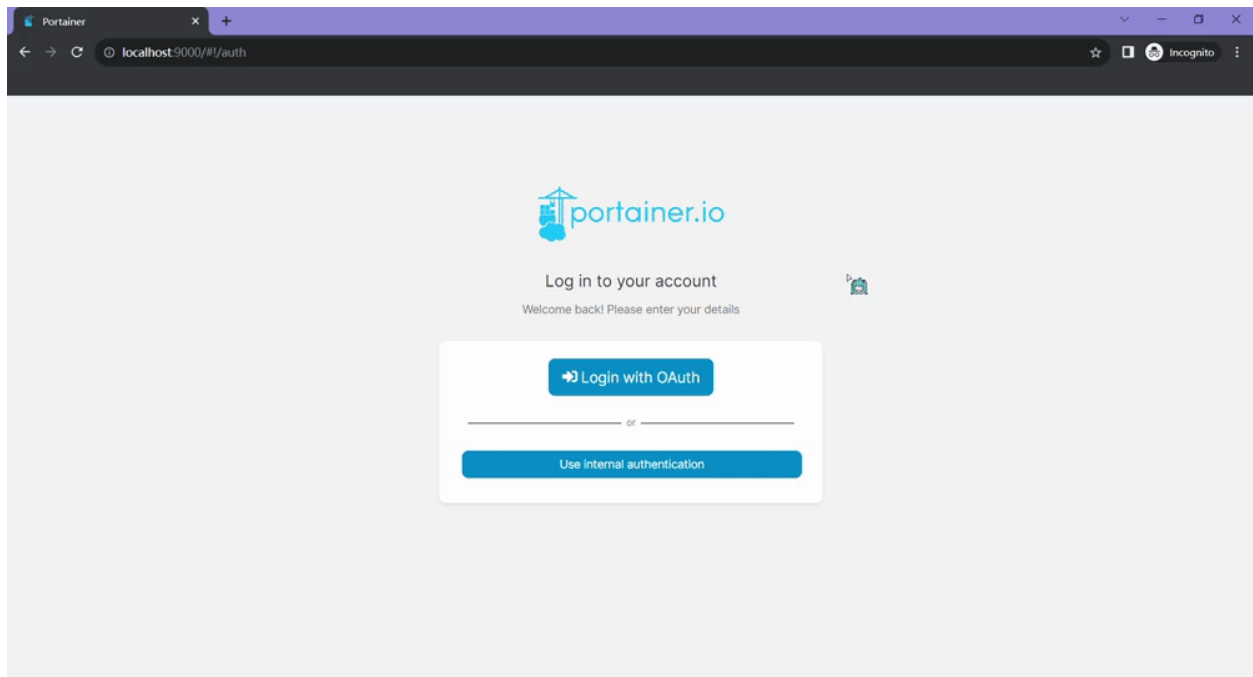
Scopes openid email profile

Actions

Save settings

- Authorization URL: https://<CASDOOR_HOST>/login/oauth/authorize
- Access token URL: https://<CASDOOR_HOST>/api/login/oauth/access_token
- Resource URL: https://<CASDOOR_HOST>/api/userinfo
- Redirect URL: https://<PORTAINER_HOST>

Log out of Portainer and test.



Java

Spring Boot

Using Casdoor in a Spring Boot project

Spring Cloud

Using Casdoor in Spring Cloud

Spring Cloud Gateway

Using Casdoor in Spring Cloud Gateway

Spring Security

2 items

Jenkins Plugin

Using the Casdoor plugin for Jenkins security

Jenkins OIDC

Using the OIDC protocol as an IDP to connect various applications, like Jenkins

Jira

2 items

Connecting Applications with OIDC Protocol - Confluence

Learn how to use OIDC protocol as IDP to connect Confluence and other applications.

RuoYi

Using Casdoor in RuoYi-Cloud

Pulsar Manager

Using Casdoor in Pulsar Manager

Using Casdoor in ShenYu

How to use Casdoor with ShenYu

ShardingSphere

Using Casdoor in ShardingSphere

Apache IoTDB

Using Casdoor with Apache IoTDB

Apache DolphinScheduler

Using Casdoor for DolphinScheduler SSO login

 **FireZone**

Using the OIDC protocol as the IDP to connect various applications, such as FireZone

 **Cloud Foundry**

Learn how to integrate Casdoor with Cloud Foundry to secure your applications.

 **Thingsboard**

Learn how to integrate Casdoor with Thingsboard to secure your applications

Spring Boot

[casdoor-spring-boot-example](#) is an example of how to use [casdoor-spring-boot-starter](#) in a Spring Boot project. We will guide you through the steps below.

Step 1: Deploy Casdoor

Firstly, Casdoor should be deployed.

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

After a successful deployment, make sure the following:

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

Now, you can quickly implement a Casdoor-based login page in your own app using the following steps.

Step 2: Import casdoor-spring-boot-starter

You can import the `casdoor-spring-boot-starter` using Maven or Gradle.

[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'
```

Step 3: Initialize Config

Initialization requires 6 string-type parameters in the following order:

Name	Required	Description
endpoint	Yes	Casdoor Server URL, such as <code>http://localhost:8000</code>
clientId	Yes	Application client ID
clientSecret	Yes	Application client secret
certificate	Yes	Application certificate
organizationName	Yes	Application organization

Name	Required	Description
applicationName	No	Application name

You can use Java properties or YAML files for initialization.

Properties [YML](#)

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

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

CAUTION

Replace the configuration values with your own Casdoor instance, especially the `clientId`, `clientSecret`, and `jwtPublicKey`.

Step 4: Redirect to the Login Page

When you need to authenticate users who access your app, you can send the target URL and redirect to the login page provided by Casdoor.

Make sure you have added the callback URL (e.g. <http://localhost:8080/login>) in the application configuration beforehand.

```
@Resource
private CasdoorAuthService casdoorAuthService;

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

Step 5: Get Token and Parse

After the Casdoor verification is passed, it will redirect back to your application with the code and state.

You can get the code and call the `getOAuthToken` method, then parse the JWT token.

`CasdoorUser` contains the basic information about the user provided by Casdoor. You can use it to set the session in your application.

```
@RequestMapping("login")
public String login(String code, String state, HttpServletRequest
```

Services

Examples of APIs are shown below:

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

- CasdoorResourceService
 - `CasdoorResponse response = casdoorResourceService.uploadResource(user, tag, parent, fullPath, file);`
 - `CasdoorResponse response = casdoorResourceService.deleteResource(file.getName());`

More Resources

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

- [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

In the Spring Cloud microservice system, general authentication occurs at the gateway. Please refer to [casdoor-springcloud-gateway-example](#) for more information.

If you want to use Casdoor in a single service, you can refer to [casdoor-spring-boot-example](#).

Whether it's in the gateway layer or in a single service, both use the [casdoor-spring-boot-starter](#).

What's more

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

- [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

The [casdoor-springcloud-gateway-example](#) is an example of how to use the [casdoor-spring-boot-starter](#) as an OAuth2 plugin in Spring Cloud Gateway. The steps to use it are described below.

Step 1: Deploy Casdoor

Firstly, Casdoor should be deployed. You can refer to the official Casdoor documentation for the [Server Installation](#). Please deploy your Casdoor instance in production mode.

After a successful deployment, you need to ensure the following:

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

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

Step 2: Initialize a Spring Cloud Gateway

You can use the code from this example directly or combine it with your own business code.

You need a gateway service and at least one business service. In this example, `casdoor-gateway` is the gateway service and `casdoor-api` is the business service.

Step 3: Include the dependency

Add the `casdoor-spring-boot-starter` dependency to your Spring Cloud Gateway project.

For 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>
```

For Gradle:

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

Step 4: Configure your properties

Initialization requires 6 parameters, all of which are of type string.

Name (in order)	Required	Description
endpoint	Yes	Casdoor Server URL, such as <code>http://localhost:8000</code>
clientId	Yes	Application.client_id
clientSecret	Yes	Application.client_secret
certificate	Yes	Application.certificate
organizationName	Yes	Application.organization
applicationName	No	Application.name

You can use Java properties or YAML files to initialize these parameters.

For 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
```

For YAML:

```
casdoor:
  endpoint: http://localhost:8000
  client-id: <client-id>
```


In addition, you need to configure Gateway Routing. For YAML:

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

Step 5: Add the CasdoorAuthFilter

Add an implementation class of the GlobalFilter interface to the gateway for identity verification, such as the CasdoorAuthFilter used in this example.

If the authentication fails, it returns a 401 status code to the frontend to redirect them to the login interface.

```
@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) {
```

Step 6: Get the Service and use it

Now provide 5 services: `CasdoorAuthService`, `CasdoorUserService`, `CasdoorEmailService`, `CasdoorSmsService`, and `CasdoorResourceService`.

You can create them as follows in the Gateway project.

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

When you require authentication for accessing your app, you can send the target URL and redirect to the login page provided by Casdoor.

Please make sure that you have added the callback URL (e.g., <http://localhost:9090/callback>) in the application configuration in advance.

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

After successful verification by Casdoor, it will be redirected back to your application with a code and state. You can get the code and call the `getOAuthToken` method to parse out the JWT token.

`CasdoorUser` contains the basic information about the user provided by Casdoor. You can use it as a keyword to set the session in your application.

```

@RequestMapping("callback")
public Mono<String> callback(String code, String state,
    ServerWebExchange exchange) {
    String token = "";
    CasdoorUser user = null;
    try {
        token = casdoorAuthService.getOAuthToken(code, state);
        user = casdoorAuthService.parseJwtToken(token);
    } catch(CasdoorAuthException e) {
        e.printStackTrace();
    }
    CasdoorUser finalUser = user;
    return exchange.getSession().flatMap(session -> {
        session.getAttributes().put("casdoorUser", finalUser);
        return Mono.just("redirect:/");
    });
}

```

Examples of the APIs are shown below.

- CasdoorAuthService
 - `String token = casdoorAuthService.getOAuthToken(code, "app-built-in");`
 - `CasdoorUser casdoorUser = casdoorAuthService.parseJwtToken(token);`
- CasdoorUserService
 - `CasdoorUser casdoorUser = casdoorUserService.getUser("admin");`
 - `CasdoorUser casdoorUser = casdoorUserService.getUserByEmail("admin@example.com");`
 - `CasdoorUser[] casdoorUsers = casdoorUserService.getUsers();`
 - `CasdoorUser[] casdoorUsers = casdoorUserService.getSortedUsers("created_time", 5);`
 - `int count = casdoorUserService.getUserCount("0");`

- `CasdoorResponse response = casdoorUserService.addUser(user);`
- `CasdoorResponse response = casdoorUserService.updateUser(user);`
- `CasdoorResponse response = casdoorUserService.deleteUser(user);`
- CasdoorEmailService
 - `CasdoorResponse response = casdoorEmailService.sendEmail(title, content, sender, receiver);`
- CasdoorSmsService
 - `CasdoorResponse response = casdoorSmsService.sendSms(randomCode(), receiver);`
- CasdoorResourceService
 - `CasdoorResponse response = casdoorResourceService.uploadResource(user, tag, parent, fullPath, file);`
 - `CasdoorResponse response = casdoorResourceService.deleteResource(file.getName());`

Step 7: Restart the project

After starting the project, open your favorite browser and visit

<http://localhost:9090>. Then click any button that requests resources from

`casdoor-api`.



localhost:9090

Casdoor

Get Resource

Update Resource

The gateway authentication logic will be triggered. Since you are not logged in, you will be redirected to the login interface. Click the Login button.



localhost:9090/toLogin

Click to login

Login

You can see the unified login platform of Casdoor.



Auto sign in

[Forgot password?](#)

Sign In

[No account? sign up now](#)

After a successful login, you will be redirected to the main interface. Now you can click any button.



localhost:9090

Casdoor

Get Resource

Update Resource

"success get resource1 "

What's more

You can explore the following projects/docs to learn more about the integration of Java with Casdoor.

- [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 Security

Spring Security OAuth

Using Spring Security as an example to demonstrate how to use OIDC to connect to your applications

Spring Security Filter with OIDC integration for Casdoor

This article explains how to use Spring Security Filter to connect your application with Casdoor using OIDC.

Spring Security OAuth

Casdoor can use the OIDC protocol as an IDP to connect various applications. In this guide, we will use Spring Security as an example to show you how to use OIDC to connect to your applications.

Step 1: Deploy Casdoor

First, you need to deploy Casdoor.

You can refer to the Casdoor official documentation for the [Server Installation](#).

After successfully deploying Casdoor, make sure:

- The Casdoor server is running on <http://localhost:8000>.
- Open your favorite browser and visit <http://localhost:7001>, where you will see the login page of Casdoor.
- Verify that the login functionality is working fine by entering `admin` and `123`.

Now, you can quickly implement a Casdoor-based login page in your own app by following the steps below.

Step 2. Configure Casdoor application

1. Create a new Casdoor application or use an existing one.
2. Add your redirect URL (You can find more details on how to obtain the redirect URL in the next section).

Client ID ⓘ : 24a25ea0714d92e78595 **Client ID**

Client secret ⓘ : 155 [REDACTED] **Client Secret**

Redirect URLs ⓘ :

Redirect URL
http://localhost:8082/ui-one/login/oauth2/code/custom

3. Add the desired provider and fill in any additional settings.

On the application settings page, you will find two values: `Client ID` and `Client secret`, as shown in the image above. We will use these values in the next step.

Open your preferred browser and visit: `http://CASDOOR_HOSTNAME/.well-known/openid-configuration`. Here, you will find the OIDC configuration of Casdoor.

Step 3. Configure Spring Security

Spring Security natively supports OIDC.

You can customize the settings of Spring Security OAuth2 Client:

⚠ CAUTION

You should replace the configuration with your own Casdoor instance, especially the `<Client ID>` and others.

`application.yml` `application.properties`

```
spring:
  security:
    oauth2:
      client:
        registration:
```

```
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
```

CAUTION

For the default situation of Spring Security, the `<Redirect URL>` should be like `http://<Your Spring Boot Application Endpoint>/<Servlet Prefix if it is configured>/login/oauth2/code/custom`. For example, in the following demo, the redirect URL should be `http://localhost:8080/login/oauth2/code/custom`.

You should also configure this in the `casdoor` application.

You can also customize the settings using `ClientRegistration` in your code. You can find the mapping [here](#)

Step 4: Get Started with a Demo

1. We can create a Spring Boot application.
2. We can add a configuration that protects all endpoints except `/` and `/login**` for users to log in.

```

@EnableWebSecurity
public class UiSecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
            .antMatchers("/", "/login**")
            .permitAll()
            .anyRequest()
            .authenticated()
            .and()
            .oauth2Login();
    }
}

```

3. We can add a naive page for the user to log in.

```

<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Spring OAuth Client Thymeleaf - 1</title>
<link rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/
bootstrap.min.css" />
</head>
<body>
    <nav
        class="navbar navbar-expand-lg navbar-light bg-light shadow-sm
p-3 mb-5">
        <a class="navbar-brand" th:href="@{/foos/}">Spring OAuth Client
            Thymeleaf - 1</a>
    </nav>
    <div class="container">
        <label>Welcome!</label> <br /> <a th:href="@{/foos/}"
            class="btn btn-primary">Login</a>
    </div>
</body>
</html>

```

When the user clicks the `login` button, they will be redirected to `casdoor`.

4. Next, we can define our protected resources. We can expose an endpoint called `/foos` and a web page for display.

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) {
        List<FooModel> foos = new ArrayList<>();
        foos.add(new FooModel(1L, "a"));
        foos.add(new FooModel(2L, "b"));
        foos.add(new FooModel(3L, "c"));
    }
}
```

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>&nbsp;&nbsp;&nbsp;&nbsp;</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}">
          <td>
            <span th:text="{foo.id}">ID</span>
          </td>
          <td>
            <span th:text="{foo.name}">Name</span>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
</body>
</html>
```

 CAUTION

All the web page templates should be placed under `resources/templates`.

Step 5: Try the demo!

Firstly, you can try opening your favorite browser and directly visiting `/foos`. It will automatically redirect you to Casdoor's login page. You can log in there or from the root page.

If you visit your root page, you will see the Casdoor Application Setting.

Spring OAuth Client Thymeleaf - 1

Welcome !

Login

Click the `login` button and the page will redirect you to Casdoor's login page.

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

After logging in, the page will redirect you to `/foos`.

Spring OAuth Client Thymeleaf -1

Hi,

Your Username

All Foos:

ID	Name
1	a
2	b
3	c

Spring Security Filter with OIDC integration for Casdoor

Casdoor is an open-source IDP that supports OIDC and various other protocols. In this article, we will see how to integrate Casdoor with your application using Spring Security Filter and OIDC.

Step 1: Deploy Casdoor

First, you need to deploy the Casdoor server. Refer to the [official documentation](#) for server installation instructions. After successful deployment, ensure that:

- The Casdoor server is running at <http://localhost:8000>.
- You can see the Casdoor login page at <http://localhost:7001>.
- You can test the login functionality by logging in with the credentials `admin` and `123`.

After verifying these steps, follow the steps below to integrate Casdoor with your application.


Step 2: Configure Casdoor Application

- Create a new Casdoor application or use an existing one.
- Add your redirect URL. You can find more information about obtaining the redirect URL in the next section.

Name ? : application_a6ftas → your application name

Display name ? : New Application - a6ftas

Logo ? :
URL ? : https://cdn.casbin.org/img/casdoor-logo_1185x256.png

Preview: 

Home ? : [↗](#)

Description ? :

Organization ? : organization_carg1b → your organization name

Client ID ? : 3ed7314825ecf955cb19 → your client id

Client secret ? : ee9314ea228 [blurred] → your client secret

Cert ? : cert-built-in

Redirect URLs ? :
Redirect URLs [Add](#)
Redirect URL
<http://localhost:3000/callback> → your redirect url

- Obtain your **Certificate** on the certificate editing page.

Display name: Built-in Cert

Scope: JWT

Type: x509

Crypto algorithm: RS256

Bit size: 4096

Expire in years: 20

Certificate: Copy certificate Download certificate

-----BEGIN CERTIFICATE-----
 -----BEGIN PRIVATE KEY-----

Private key: Copy private key Download private key

- Add the provider and other settings as needed.

You can obtain the values for `Application Name`, `Organization Name`, `Redirect URL`, `Client ID`, `Client Secret`, and `Certificate` on the application settings page. We will use them in the next step.

Step 3: Configure Spring Security

You can customize the settings of the Spring Security filters to process tokens:

⚠ CAUTION

Make sure you replace the configuration values with your own Casdoor instance, especially `<Client ID>` and the others.

```
server:
  port: 8080
casdoor:
  endpoint: http://CASDOOR_HOSTNAME:8000
  client-id: <Client ID>
  client-secret: <Client Secret>
  certificate: <Certificate>
```

⚠ CAUTION

For frontend applications, the default value of `<FRONTEND_HOSTNAME>` is `localhost:3000`. In this demo, the redirect URL is `http://localhost:3000/callback`. Make sure to configure this in your `casdoor` application.

Step 4: Configure Frontend

You need to install `casdoor-js-sdk` and configure the SDK as follows:

1. Install `casdoor-js-sdk`.

```
npm i casdoor-js-sdk
# or
yarn add casdoor-js-sdk
```

2. Set up `SDK`.

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

// Serverurl is the URL where spring security is deployed
export const ServerUrl = "http://BACKEND_HOSTNAME:8080";

const sdkConfig = {
  serverUrl: "http://CASDOOR_HOSTNAME:8000",
  clientId: "<your client id>",
  appName: "<your application name>",
  organizationName: "<your organization name>",
  redirectPath: "/callback",
};

export const CasdoorSDK = new Sdk(sdkConfig);
```

Step 5: Set Up a Demo

1. Create a Spring Boot application.
2. Add some configurations to handle 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()
        );

        // set unauthorized requests exception handler
```

3. Add a simple JWT filter to intercept requests that require token verification.

```
@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;
        }
    }
}
```

When the user accesses the interface requiring authentication, `JwtTokenFilter` will obtain the token from the request header `Authorization` and verify it.

4. Define a `Controller` to handle when the user logs in to Casdoor. After the user logs in, they will be redirected to the server and carry the `code` and `state`. The server then needs to verify the user's identity from Casdoor and obtain the `token` through these two parameters.

```
@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());
        }
    }

    // ...
}
```

Step 6: Try the Demo

You can access the frontend application through your browser. If you are not logged in,

you will see a login button. Click on it, and you will be redirected to the Casdoor login page.

If you visit your root page,

Casdoor Login

Click the **Casdoor Login** button, and the page will redirect to Casdoor's login page.




Auto sign in

[Forgot password?](#)

Sign In

[No account? sign up now](#)

Made with  by **Casdoor**

After logging in, you will be redirected to .



New User - rtsbx4

Logout

Jenkins Plugin

Casdoor provides a plugin that allows users to log in to Jenkins. Here, we will show you how to use the Casdoor plugin for Jenkins security.

The following are some of the configuration settings:

`CASD00R_HOSTNAME`: The domain name or IP where the Casdoor server is deployed.

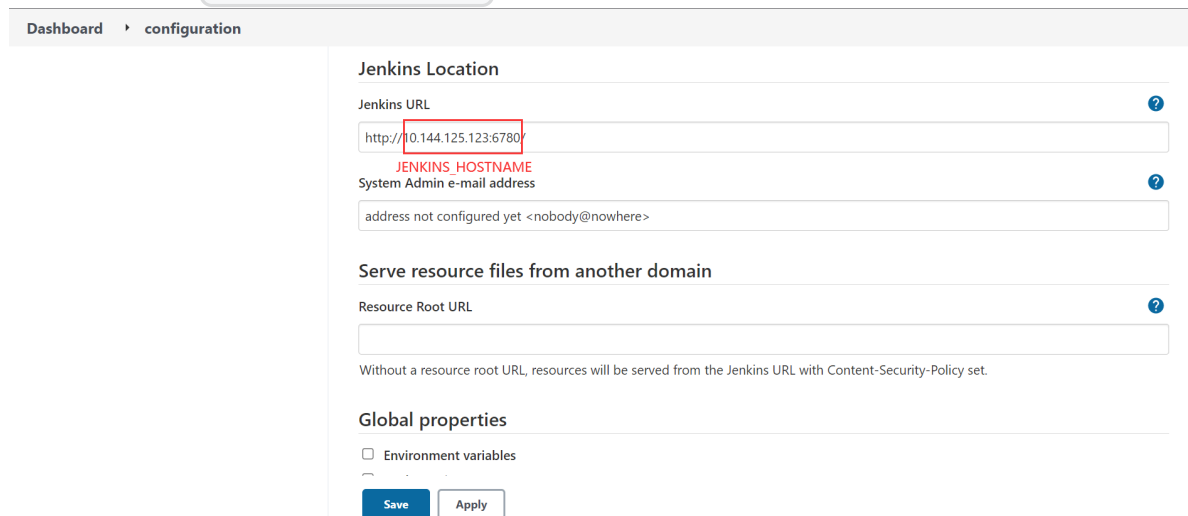
`JENKINS_HOSTNAME`: The domain name or IP where Jenkins is deployed.

Step 1: Deploy Casdoor and Jenkins

Firstly, deploy [Casdoor](#) and [Jenkins](#).

After a successful deployment, ensure the following:

1. Set the Jenkins URL (Manage Jenkins → Configure System → Jenkins Location) to `JENKINS_HOSTNAME`.



The screenshot shows the Jenkins configuration page for the 'Jenkins Location' section. The 'Jenkins URL' field is highlighted with a red box and contains the value 'http://0.144.125.123:6780'. Below it, the 'System Admin e-mail address' field is set to 'address not configured yet <nobody@nowhere>'. The 'Serve resource files from another domain' section has an empty 'Resource Root URL' field. The 'Global properties' section has an unchecked checkbox for 'Environment variables'. At the bottom, there are 'Save' and 'Apply' buttons.

Dashboard > configuration

Jenkins Location

Jenkins URL ?
http://0.144.125.123:6780

System Admin e-mail address ?
address not configured yet <nobody@nowhere>

Serve resource files from another domain

Resource Root URL ?

Without a resource root URL, resources will be served from the Jenkins URL with Content-Security-Policy set.

Global properties

Environment variables

2. Verify that Casdoor can be logged in and used normally.
3. Set the `origin` value of Casdoor (conf/app.conf) to `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
```

Step 2: Configure the Casdoor Application

1. Create a new Casdoor application or use an existing one.
2. Add a redirect URL: `http://JENKINS_HOSTNAME/securityRealm/finishLogin`

Description ⓘ: Casdoor for Jenkins

Organization ⓘ built-in

Client ID ⓘ: bbd0bd66696e504dec59 Client ID

Client secret ⓘ: d2de01b01...110b47465c Client secret

Redirect URLs ⓘ:

Redirect URL	
http://10.144.125.123:6780/securityRealm/finishLogin	Add a redirect url for Jenkins

JENKINS_HOSTNAME

3. Add the desired provider and provide any additional settings.

On the application settings page, you will find two values: `Client ID` and `Client secret`, as shown in the picture above. We will use these values in the next step.


Open your favorite browser and visit `http://CASDOOR_HOSTNAME/.well-known/openid-configuration` to view the OIDC configuration of Casdoor.

Step 3: Configure Jenkins

Now, you can install the Casdoor plugin from the marketplace or by uploading its `jar` file.

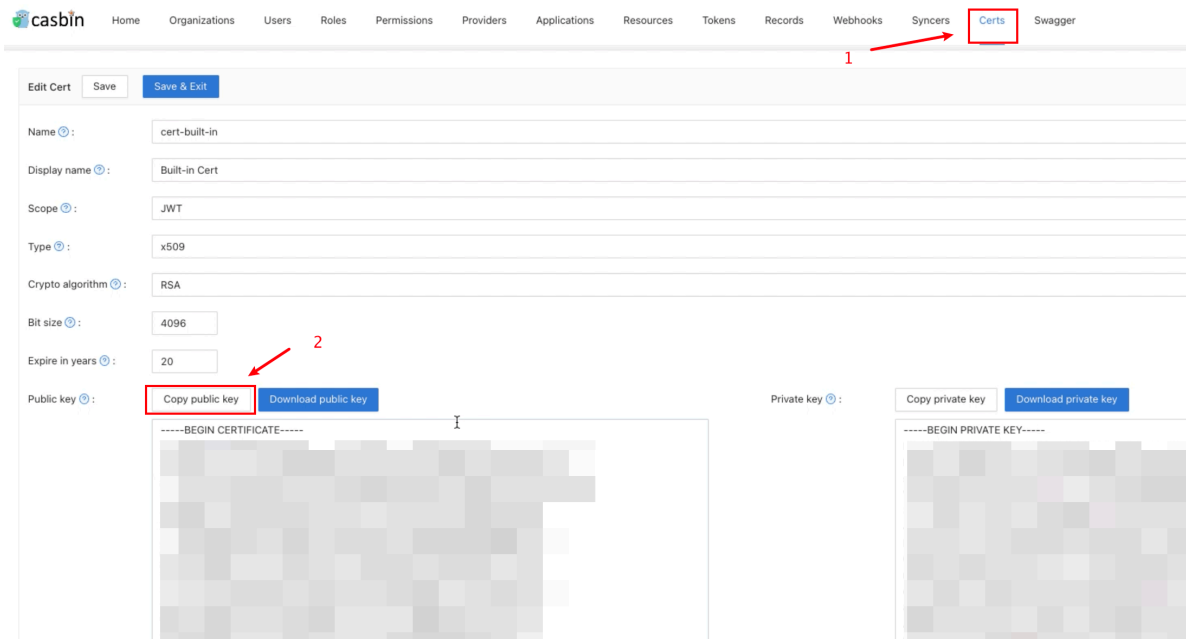
After the installation is complete, go to Manage Jenkins → Configure Global Security.

Suggestion: Back up the Jenkins `config.xml` file and use it for recovery in case of setup errors.



The screenshot shows the 'Configure Global Security' page in Jenkins. The 'Authentication' section has a checkbox for 'Disable remember me'. The 'Security Realm' section has a radio button selected for 'Casdoor Authentication Plugin'. Below this, there are several input fields with error messages: 'Casdoor Endpoint' (empty, error: 'Casdoor Endpoint is required.'), 'Client ID' (empty, error: 'Client Id is required.'), 'Client Secret' (empty, error: 'Client Secret is required.'), and 'JWT Public Key' (empty, error: 'Jwt Public Key is required.'). There are also fields for 'Organization Name' and 'Application Name'. At the bottom, there are radio buttons for 'Delegate to servlet container' and 'Jenkins' own user database', a 'Save' button, an 'Apply' button, and an 'Advanced...' link.

1. In the Security Realm section, select "Casdoor Authentication Plugin".
2. In the Casdoor Endpoint field, enter the `CASDOOR_HOSTNAME` mentioned earlier.
3. In the Client ID field, enter the `Client ID` mentioned earlier.
4. In the Client secret field, enter the `Client secret` mentioned earlier.
5. In the JWT Public Key field, provide the public key used to validate the JWT token. You can find the public key in Casdoor by clicking on `Cert` at the top. After clicking `edit` on your application, you can copy the public key from the following page.



6. Organization Name and Application Name are optional. You can specify your organization and application to verify users in other organizations and applications. If these fields are left empty, the plugin will use the default organization and application.
7. In the Authorization section, check "Logged-in users can do anything". Disable "Allow anonymous read access".
8. Click `Save`.

Jenkins will now automatically redirect you to Casdoor for authentication.

Jenkins OIDC

Casdoor can use the OIDC protocol as an IDP to connect various applications. In this example, we will use Jenkins to demonstrate how to use OIDC to connect to your applications.

The following are some of the names used in the configuration:

- `CASD00R_HOSTNAME`: The domain name or IP where the Casdoor server is deployed.
- `JENKINS_HOSTNAME`: The domain name or IP where Jenkins is deployed.

Step 1: Deploy Casdoor and Jenkins

Firstly, deploy [Casdoor](#) and [Jenkins](#).

After a successful deployment, ensure the following:

1. Set the Jenkins URL (Manage Jenkins → Configure System → Jenkins Location) to `JENKINS_HOSTNAME`.

Dashboard > configuration

Jenkins Location

Jenkins URL ?
http://10.144.125.123:6780
JENKINS_HOSTNAME

System Admin e-mail address ?
address not configured yet <nobody@nowhere>

Serve resource files from another domain

Resource Root URL ?

Without a resource root URL, resources will be served from the Jenkins URL with Content-Security-Policy set.

Global properties

Environment variables

2. Ensure that Casdoor can be logged in and used normally.
3. Set Casdoor's `origin` value (conf/app.conf) to `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
```

Step 2: Configure the Casdoor application

1. Create a new Casdoor application or use an existing one.

2. Add a redirect URL: `http://JENKINS_HOSTNAME/securityRealm/finishLogin`

Description ⓘ: Casdoor for Jenkins

Organization ⓘ: built-in

Client ID ⓘ: bbd0bd66696e504dec59 Client ID

Client secret ⓘ: d2de01b01[REDACTED]110b47465c Client secret

Redirect URLs ⓘ:

Redirect URL	
http://10.144.125.123:6780/securityRealm/finishLogin	Add a redirect url for Jenkins

JENKINS_HOSTNAME

3. Add the provider you want and provide any additional settings.

You will obtain two values from the application settings page: `Client ID` and `Client secret`. We will use these values in the next step.

Open your favorite browser and visit: `http://CASDOOR_HOSTNAME/.well-known/openid-configuration` to view the OIDC configuration of Casdoor.

Step 3: Configure Jenkins

First, we need to install [OpenId Connect Authentication](#) as Jenkins does not natively support OIDC.

After the installation is complete, go to `Manage Jenkins` → `Configure Global Security`.

Dashboard > Manage Jenkins

Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#). [Set up agent](#) [Set up cloud](#) [Dismiss](#)

用户列表
构建历史
Manage Jenkins
My Views

构建队列
队列中没有构建任务

构建执行状态
1 空闲
2 空闲

System Configuration

- Configure System**
Configure global settings and paths.
- Global Tool Configuration**
Configure tools, their locations and automatic installers.
- Manage Plugins**
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Manage Nodes and Clouds**
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Security

- Configure Global Security**
Secure Jenkins; define who is allowed to access/use the system.
- Manage Credentials**
Configure credentials
- Configure Credential Providers**
Configure the credential providers and types



TIP


Make sure to back up the Jenkins `config.xml` file to recover in case of any setup errors.

1. In Access Control, select `Login with Openid Connect` as the Security Realm.
2. Specify the `Client ID` noted above in the Client ID field.
3. Specify the `Client secret` noted above in the Client secret field.
4. In the Configuration mode, select `Automatic configuration` and enter `http://CASDOOR_HOSTNAME/.well-known/openid-configuration` as the Well-known configuration endpoint.

Security Realm

Delegate to servlet container ?


Jenkins' own user database ?

Login with Openid Connect  Select this ?

Client id ?

bbd0bd66696e504dec59 Input your Client ID

Client secret ?

 Concealed Input your Client secret Change Password

Configuration mode

Automatic configuration ?

Well-known configuration endpoint ?

http://10.144.1.2:8000/well-known/openid-configuration

CASDOOR_HOSTNAME

Manual configuration ?

If your Casdoor is deployed locally, you may need to select **Manual configuration** and provide the following 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**

Configuration mode

Automatic configuration ?

Manual configuration ?

Token server url ?

http://10.144.1.2:8000/api/login/oauth/access_token

CASDOOR_HOSTNAME

Authorization server url ?

http://10.144.1.2:8000/login/oauth/authorize

UserInfo server url ?

http://10.144.1.2:8000/api/get-account

Scopes

address phone openid profile offline_access email

5. Click on **Advanced settings** and fill in the following:

- In the User name field, specify **name**.

- In the Full name field, specify `displayName`.
- In the Email field, specify `email`.

User name field name

Full name field name

Email field name

Groups field name ?

Token Field Key To Check ?


6. In the **Authorization** section, enable “Logged-in users can do anything” and disable “Allow anonymous read access”. You can configure more complex authorization later, but for now, check if OpenID works correctly.

Log out of Jenkins, and it should redirect you to Casdoor for authentication.



Auto sign in [Forgot password?](#)

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



Jira

Via Built-in SSO

Using the OIDC protocol as an IDP to connect various applications, such as Jira

Using the miniOrange Plugin

Connect casdoor and Jira using the OIDC protocol as the IDP

Via Built-in SSO

This is a free method to connect Casdoor, but your website must use HTTPS.

[Casdoor](#) can use the OIDC protocol as an IDP to connect various applications. Here is a [Jira](#) tutorial.

The following are some of the names in the configuration:

- `CASDOOR_HOSTNAME`: Domain name or IP where the Casdoor server is deployed.
- `Jira_HOSTNAME`: Domain name or IP where Jira is deployed.

Step 1: Deploy Casdoor and Jira

Firstly, deploy [Casdoor](#) and [Jira](#).

After a successful deployment, ensure the following:

1. Casdoor can be logged in and used normally.
2. You can set `CASDOOR_HOSTNAME` to `http://localhost:8000` when deploying Casdoor in `prod` mode. See [production mode](#).

Step 2: Configure Casdoor application

1. Create or use an existing Casdoor application.
2. Find Authentication methods:

Administration Search Jira admin

Applications Projects Issues Manage apps User management Latest upgrade report **System** 2

General configuration
Find more admin tools
Jira mobile app

SYSTEM SUPPORT
System info
Instrumentation
Monitoring
Database monitoring
Integrity checker
Logging and profiling
Scheduler details
Troubleshooting and support tools
Clean up
Audit log
Clustering

SECURITY
Project roles
Global permissions
Password Policy
User sessions
Authentication methods 3
Administering personal access
Address
<https://test.v2tl.com/plugins/servlet/applications/versions-licenses>

Authentication methods
Manage how users authenticate. Save authentication configurations using SAML, OpenID Connect, or Crowd as the identity provider. [Learn more about using multiple identity providers.](#)

Make authentication safer
Authenticating with username and password is less secure than through single sign-on. Now that you've configured the latter, consider disabling product login form and basic authentication.
Communicate this change to your users.
[How to disable](#) · [Dismiss](#)

Login options

Name	Type	Last updated	Show on login page	Actions
Username and password	Product login form	Never	<input checked="" type="checkbox"/>	...
casdoor	OpenID Connect	25 April 2023 4:33 PM	<input checked="" type="checkbox"/>	...

Authentication on API calls
 Allow basic authentication on API calls.
You can use personal access tokens as a safer alternative method of authentication. See [Using personal access tokens](#).

3. Add a Configuration and choose OpenID Connection single sign-on in the Authentication method

Add new configuration

Name *

Use a unique name for this configuration.

Authentication method

OpenID Connect single sign-on



Users log in using OpenID Connect

4. Find the redirect URL:

Give these URLs to your identity provider

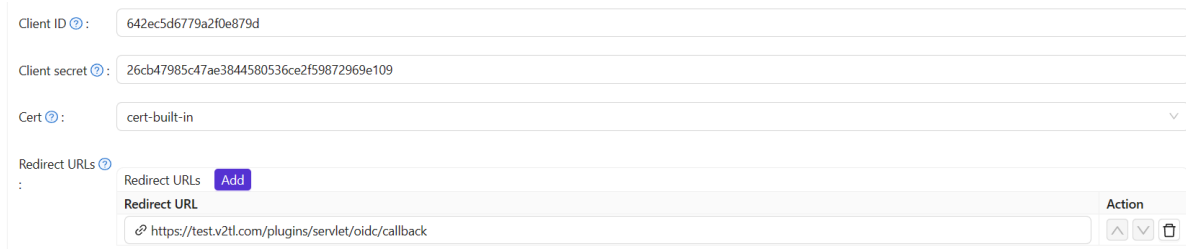
Redirect URL

<https://test.v2tl.com/plugins/servlet/oidc/callback>



Location where the client is sent to after successful account authentication.

5. Add a redirect URL:



The screenshot shows a configuration interface with the following fields:

- Client ID: 642ec5d6779a2f0e879d
- Client secret: 26cb47985c47ae3844580536ce2f59872969e109
- Cert: cert-built-in
- Redirect URLs: A table with one entry:

Redirect URL	Action
https://test.v2tl.com/plugins/servlet/oidc/callback	Up, Down, Delete icons

Not surprisingly, you can obtain 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 configuration of Casdoor.

Step 3: Configure Jira

1. We need to continue configuring our Configuration in Jira

Edit existing configuration

Name *

Use a unique name for this configuration.

Authentication method

OpenID Connect single sign-on

Users log in using OpenID Connect

OpenID Connect settings

Issuer URL *

your casdoor url

The complete URL of the OpenID Provider. Needs to be unique.

Client ID *

application client ID

The client identifier, as registered with the OpenID Provider.

Client secret *

application client secret [Change](#)

Client secret is used in conjunction with the Client ID to authenticate the client application against the OpenID Provider.

Username mapping *

Used to map IdP claims to the username, e.g. \${sub}

Additional scopes

phone × email × address × profile ×

The default scope is 'openid'. Add more scopes if needed to obtain the username claim.

Redirect URL

`https://test.v2tl.com/plugins/servlet/oidc/callback`



Copy it to casdoor

Location where the client is sent to after successful account authentication.

Initiate login URL

`https://test.v2tl.com/plugins/servlet/oidc/initiate-login`



URL used for OpenID Provider-initiated login.

Additional settings

The authorization, token, and user info endpoints will be filled automatically if your Identity provider offers this option. If not, you will be asked to provide this information.

- Fill the data automatically from my chosen identity provider.

JIT provisioning

Just-in-time user provisioning allows users to be created and updated automatically when they log in through SSO to Atlassian Data Center applications. [Learn more](#).

- Create users on login to the application

OpenID Connect behaviour

- Remember user logins

If checked, successful login history will be saved and users will be logged in automatically without the need for reauthentication.

Login page settings

Decide if the IdP should be visible on login page and customize what the user will see on the button.

- Show IdP on the login page

Login button text *

casdoor

The text is shown to the user on the login page. Remaining characters: 33.

Save configuration

Cancel

2. You can configure more complex authorization later. For now, check if OpenID actually works.

Administration Search Jira admin

Applications Projects Issues Manage apps User management Latest upgrade report **System**

- General configuration
 - Find more admin tools
 - Jira mobile app
- SYSTEM SUPPORT
 - System info
 - Instrumentation
 - Monitoring
 - Database monitoring
 - Integrity checker
 - Logging and profiling
 - Scheduler details
 - Troubleshooting and support tools
 - Clean up
 - Audit log
 - Clustering
- SECURITY
 - Project roles
 - Global permissions

Authentication methods Add configuration

Manage how users authenticate. Save authentication configurations using SAML, OpenID Connect, or Crowd as the identity provider. [Learn more about using multiple identity providers.](#)

⚠️ Make authentication safer
Authenticating with username and password is less secure than through single sign-on. Now that you've configured the latter, consider disabling product login form and basic authentication. Communicate this change to your users.
[How to disable](#) - [Dismiss](#)

Login options

Name	Type	Last updated	Show on login page	Actions
Username and password	Product login form	Never	<input checked="" type="checkbox"/>	⋮
casdoor	OpenID Connect	26 April 2023 7:20 PM	<input checked="" type="checkbox"/>	⋮

Authentication on API calls

Allow basic authentication on API calls.
You can use personal access tokens as a safer alternative method of authentication. See [Using personal access tokens](#).

Using the miniOrange Plugin

This tutorial explains how to use [miniOrange](#) to connect casdoor and Jira.

[Casdoor](#) can use the OIDC protocol as the IDP to connect various applications. You can refer to this [Jira](#) tutorial for more information.

The following are some important names in the configuration:

`CASD00R_HOSTNAME`: The domain name or IP where the Casdoor server is deployed.

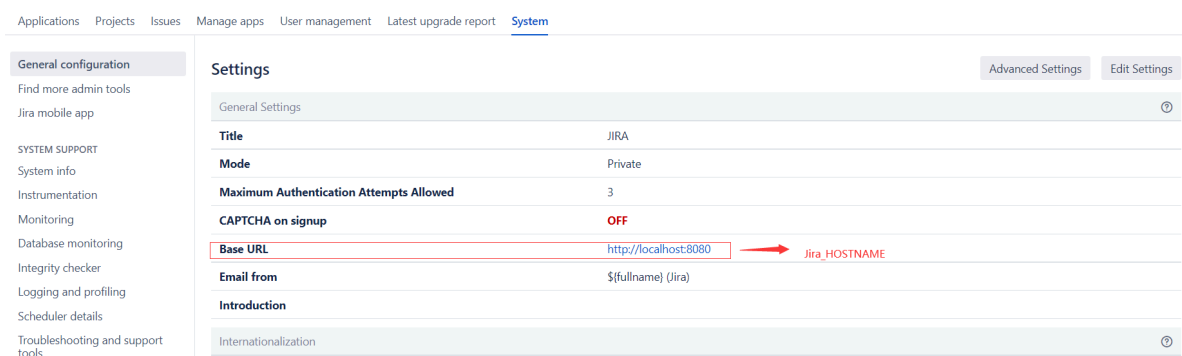
`Jira_HOSTNAME`: The domain name or IP where Jira is deployed.

Step 1: Deploy Casdoor and Jira

Firstly, deploy [Casdoor](#) and [Jira](#).

After successful deployment, make sure:

1. Set Jira URL (Plans → Administration → System → General Configuration) to `Jira_HOSTNAME`.



The screenshot shows the Jira Administration interface. The 'System' tab is selected, and the 'General configuration' section is active. The 'Settings' page displays various configuration options. The 'Base URL' field is highlighted with a red box and a red arrow pointing to the value 'http://localhost:8080'. A red arrow also points from the text 'jira_HOSTNAME' to the right of the 'Base URL' field, indicating that the value should be updated to match the Jira hostname.

Setting	Value
Title	JIRA
Mode	Private
Maximum Authentication Attempts Allowed	3
CAPTCHA on signup	OFF
Base URL	http://localhost:8080
Email from	\${fullname} (jira)
Introduction	
Internationalization	

2. Casdoor can be logged in and used normally.
3. You can set `CASD00R_HOSTNAME` to `http://localhost:8000` when deploying Casdoor in `prod` mode. See [production mode](#).

Step 2: Configure Casdoor Application and Jira

1. Create a new Casdoor application or use an existing one.
2. Install the [miniOrange](#) app to support OAuth. You can find this app in Plans->Administration->Find new apps->search

The screenshot shows the Atlassian Marketplace for Jira interface. The top navigation bar includes 'Administration' and 'Manage apps'. The left sidebar has 'Find new apps' highlighted. The main content area displays 'Atlassian Marketplace for JIRA' with a search filter set to 'Oauth'. The search results show the 'mO Jira OAuth SSO, Jira OpenID Connect SSO, Jira OIDC SSO' app by miniOrange, which has 607 installations and a 5-star rating. The app description mentions it is the best OAuth SSO app in the market and supports various authentication providers like Google, AWS, Azure, and GitHub.

3. Set `Selected Application` to Custom OpenId.
4. Find the redirect URL:

miniOrange OAuth Configuration
 Manage apps Ask Us On Forum Frequently Asked Questions

Back to common setting

OAuth/OIDC Configurations

Callback URL: <http://localhost:8080/plugins/servlet/oauth/callback>

5. Add the redirect URL:

Client ID: 514e09591ee5554b16fe

Client secret: e7f05b14a68fb23e526f08515aefb73bbab7814a

Cert: cert-built-in

Redirect URLs

Redirect URL	Action
http://localhost:8080/plugins/servlet/oauth/callback	⬆ ⬇ ⬇

6. Configure the app as follows:

Selected Application: **Custom OpenId** Import Details

Provider ID: 5c881c25-2e02-42c9-af06-0a71e0beb516

Custom App Name: casdoor

Client Id: 514e09591ee5554b16fe

Client Secret: e7f05b14a68fb23e526f08515aefb73bbab7814a

Scope: openid email profile address phone offline_access

Authorize Endpoint: http://localhost:8000/login/oauth/authorize

Access Token Endpoint: http://localhost:8000/api/login/oauth/access_token

Logout Endpoint: Enter the Logout Endpoint URL

Enter the Logout endpoint of your OAuth/OpenID Provider. Leave blank if Logout endpoint not supported by provider. e.g. If Keycloak Logout endpoint is configured with {hostname}/auth/realms/{realm-name}/protocol/openid-connect/logout, Keycloak too.

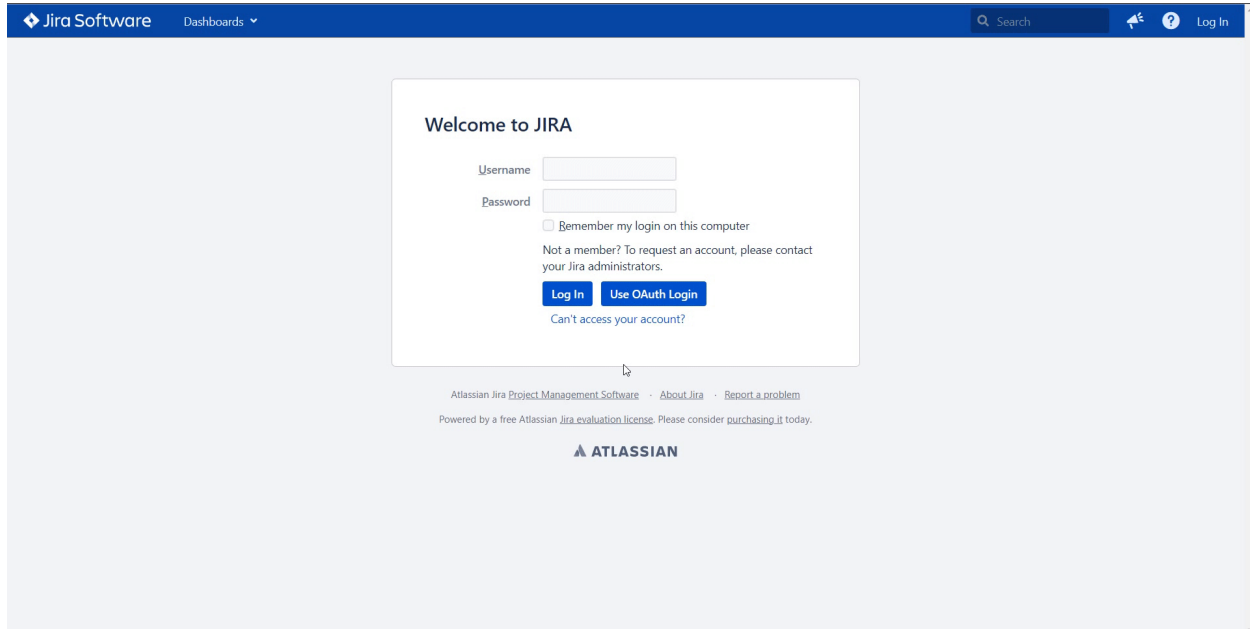
Save Test Configuration

- Token server URL: `http://CASD00R_HOSTNAME/api/login/oauth/access_token`
- Authorization server URL: `http://CASD00R_HOSTNAME/login/oauth/authorize`

- UserInfo server URL: `http://CASDOOR_HOSTNAME/api/get-account`
- Scopes: `address phone openid profile offline_access email`

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

Log out of Jira and test SSO.



Connecting Applications with OIDC Protocol - Confluence

[Casdoor](#) can use OIDC protocol as an IDP to connect various applications. In this guide, we will use [Confluence](#) as an example to demonstrate how to use OIDC to connect your applications.

To start, make sure you have deployed Casdoor and Confluence successfully. Here are a few configuration names you need to remember:

- `CASD00R_HOSTNAME`: Domain name or IP where Casdoor server is deployed.
- `Confluence_HOSTNAME`: Domain name or IP where Confluence is deployed.

Step 1: Deploy Casdoor and Confluence

First, deploy [Casdoor](#) and [Confluence](#).

After successful deployment, ensure the following:

1. Set Confluence URL to `Confluence_HOSTNAME`.

Confluence administration

General Configuration

Site Configuration Edit

Configure the appearance and behaviour of the site as a whole. The most important is the **Server Base URL**, which **must** be set to the externally-accessible address of your Confluence site.

Site Title **Confluence**
Allows you to specify the site's title which will appear on the browser title bar.

Server Base URL **http://localhost:8090** Edit
The Server Base URL is the url via which users access Confluence. [More about the Server Base URL](#).

Contact Administrators Message
Please enter information about your request for the site administrators. If you are reporting an error please be sure you include information on what you were doing and the time the problem occurred.
Allows you to configure the message that is shown to a user when they try to contact the site administrators. This should be entered using wiki markup. Not defining a message will lead to the default message being displayed.
 Contact Administrators Form
Display a contact form when trying to contact the confluence-administrators. This can only be turned off if there is a custom contact administrator message.

Formatting and International Settings Edit

You can change the default language for the Confluence interface on the [language configuration page](#) These options relate to the time and date formatting on the site. Unless you are sure of what you are doing, we recommend that you leave these as they are.

Indexing Language **English**

Encoding **UTF-8**

Time Format **h:mm a**

Date Time Format **MMM dd, yyyy HH:mm**

2. Casdoor can be logged in and used normally.
3. You can set `CASD00R_HOSTNAME` to `http://localhost:8000` if you deploy Casdoor in `prod` mode. Refer to the [production mode](#) for more details.

Step 2: Configure Casdoor application

1. Create a new Casdoor application or use an existing one.
2. Find a redirect URL:

[Back to common setting](#)

OAuth/OIDC Configurations

Callback URL: `http://localhost:8090/plugins/servlet/oauth/callback` Edit

3. Add the redirect URL to the application:

Client ID ?: `014ae4bd048734ca2dea` Edit

Client secret ?: `f26a4115725867b7bb7b668c81e1f87fae1544d` Edit

Cert ?: `cert-built-in` Edit

Redirect URLs ?:

Redirect URLs Add

Redirect URL `http://localhost:8090/plugins/servlet/oauth/callback` Edit

4. Add the desired provider and configure other settings accordingly.

On the application settings page, you will find two values: `Client ID` and `Client Secret`. We will need these in the next step.

Open your favorite browser and visit: `http://CASDOOR_HOSTNAME/.well-known/openid-configuration` to see the OIDC configuration of Casdoor.

Step 3: Configure Confluence

1. Install the [miniOrange](#) app to support OAuth. You can find this app in:

The screenshot shows the Atlassian Marketplace interface. On the left, a navigation sidebar lists various configuration options, with 'ATLASSIAN MARKETPLACE' and 'Find new apps' highlighted. The main content area is titled 'Find new apps' and displays search results for 'oauth'. The top result is 'mO Confluence OAuth SSO, Confluence OpenID Connect/OIDC SSO' by miniOrange, which has 409 installations and a 4-star rating. An arrow points to this app. Below it is 'Table Filter and Charts for Confluence' by Stiltsoft Europe OÜ, which has 15,179 installations and a 4-star rating. A magnifying glass is overlaid on the miniOrange app card.

2. Configure the app:

Selected Application: **Custom OpenId** [Import Details](#) [Setup Guide](#)

Provider ID: **4f6b30c1-eba8-4b89-ac02-4a4b7a137b97**

Custom App Name:*

Client Id:*

Client Secret:*

Scope:*

Authorize Endpoint:*

Access Token Endpoint:*

Logout Endpoint:

Enter the Logout endpoint of your OAuth/OpenID Provider. Leave blank if Logout endpoint not supported by provider.
e.g. If Keycloak Logout endpoint is configured with {hostname}/auth/realms/{realm-name}/protocol/openid-connect/logout URL then o

3. Set **Selected Application** to Custom OpenID.
4. Retrieve the Client ID and Client Secret from the Casdoor application page.

Configure the following settings for Confluence:

- **Token server URL**: `http://CASD00R_HOSTNAME/api/login/oauth/access_token`
- **Authorization server URL**: `http://CASD00R_HOSTNAME/login/oauth/authorize`
- **UserInfo server URL**: `http://CASD00R_HOSTNAME/api/get-account`
- **Scopes**: `address phone openid profile offline_access email`

You can configure more advanced authorization settings later. For now, check if OpenID actually works.

Log out of Confluence and test SSO:

Firefox 网络浏览器 4月1日 22:07

Log in - Confluence

localhost:8090/login.action?language=en_GB

Confluence

Log in

Username

Password

Remember me

[Log in](#) [Use OAuth Login](#)

[Forgot your password?](#)

EVALUATION LICENSE Are you enjoying Confluence? Please consider purchasing it today.

Cestina · Dansk · Deutsch · Eesti · English (UK) · English (US) · Español · Français · Íslenska · Italiano · Magyar · Nederlands · Norsk · Polski · Português · Română · Slovenčina · Suomi · Svenska · Pycckий · 中文 · 日本語 · 한국어

Powered by Atlassian Confluence 8.1.1 · [Report a bug](#) · [Atlassian News](#)

ATLASSIAN

localhost:8090/pluginservlet/oauth/authorize?return_to=https://localhost:8090&id=4f6b30c1-eba8-4b89-ac02-4a4b7a137b97&idp=4f6b30c1-eba8-4b89-ac02-4a4b7a137b97

RuoYi

Casdoor can be easily integrated with RuoYi-cloud.

Step 1: Deploy Casdoor

First, deploy Casdoor.

You can refer to the Casdoor official documentation for the [Server Installation](#).

After successful deployment, ensure the following:

- The Casdoor server is running at <http://localhost:8000>.
- Open your favorite browser and visit <http://localhost:7001> to access the Casdoor login page.
- Test the login functionality by entering `admin` and `123`.

Next, you can quickly implement a Casdoor-based login page in your own app following these steps.

Step 2: Configure Casdoor

To configure Casdoor, please follow these steps:

1. Open Casdoor in a browser by clicking [here](#). It is recommended to use a different browser than your development browser.
2. Configure an organization, an application, and the Synchronizer in Casdoor. You can find detailed instructions on how to do this [here](#).

Here are some additional points to keep in mind:

1. When editing the syncer, make sure to check the table columns:

Table columns ⓘ Add

Column name	Column type	Casdoor column	Is hashed	Action
user_id	integer	Id	<input type="checkbox"/>	↑ ↓ 🗑
dept_id	integer	Affiliation	<input type="checkbox"/>	↑ ↓ 🗑
user_name	string	Name	<input type="checkbox"/>	↑ ↓ 🗑
nick_name	string	DisplayName	<input type="checkbox"/>	↑ ↓ 🗑
user_type	string	Type	<input type="checkbox"/>	↑ ↓ 🗑
email	string	Email	<input type="checkbox"/>	↑ ↓ 🗑
phonenumber	string	Phone	<input type="checkbox"/>	↑ ↓ 🗑
sex	string	Gender	<input type="checkbox"/>	↑ ↓ 🗑
avatar	string	Avatar	<input type="checkbox"/>	↑ ↓ 🗑
password	string	Password	<input type="checkbox"/>	↑ ↓ 🗑
del_flag	string	IsDeleted	<input type="checkbox"/>	↑ ↓ 🗑
login_ip	string	Createdip	<input type="checkbox"/>	↑ ↓ 🗑
create_time	string	CreatedTime	<input type="checkbox"/>	↑ ↓ 🗑
password	string	Password	<input type="checkbox"/>	↑ ↓ 🗑

2. When editing the organization, make sure to select the correct password type:

Password type ⓘ:

3. Lastly, ensure that you have enabled soft deletion.

Please make sure to follow these instructions carefully to properly configure Casdoor.

Step 3. Reform your front-end

3.1 Jump to Casdoor's login page

We can use a front-end SDK, taking vue-sdk as an example here. After you initialize vue-sdk, you can obtain the Casdoor login page URL by using the `getSignInUrl()` function.

You can link it in the way you prefer, and feel free to delete any original code from

Ruoyi-Cloud that is no longer necessary, such as the original account and password el-input.

3.2 Accept the code and state returned by Casdoor

After successfully logging in through Casdoor, Casdoor sends the code and state to the page we set up. We can retrieve the code and state using the create() function.

```
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 are not null, execute handleLogin
    this.handleLogin();
  }
}
```

For RuoYi-Cloud, we simply modify its original method of sending the account and password to send the code and state instead. Therefore, the change is only in what is sent to the backend, in relation to the original login.

Step 4: Refactor your back-end

4.1 Accept the code and state returned by the front-end

```
@PostMapping("login")
```

In this method, we are using the casdoor-SpringBoot-sdk method and making slight modifications to the RuoYi-Cloud method.

For example, the RuoYi-Cloud original method registers an account with a password. I have changed it to register an account using the `casdoorRegister` method.

I have also added a method `getUserByCasdoorName` to check if the account exists, and changed the method `executeUserInfo` to `executeWithAccount` to reflect this change.

This is an easy modification, as we only need to remove the part that checks the password.

Step 5: Summary

5.1 Front-end

- The existing login and register pages need to be removed.
- Additionally, the front-end needs to accept code and state parameters and send them to the back-end.

5.2 Back-end

The RuoYi back-end already has a well-implemented login and registration function. We just need to make some minor modifications, which makes the process highly convenient.

Step 6: Detailed Steps

1. Deploy and configure Casdoor. Be sure to select the bcrypt password type for the organization, as RuoYi-Cloud also uses bcrypt for passwords.
2. Use Casdoor syncers to copy database users to your Casdoor organization. This will import the original accounts into Casdoor.
3. After deploying Casdoor, make changes to the front-end. Disable the RuoYi check code.

```
// checkcode switch
captchaEnabled: false,
// register switch
register: true,
```

Note that the RuoYi-Cloud captcha needs to be disabled in Nacos again. Also, the RuoYi-Cloud registration function needs to be enabled by setting `sys.account.registerUser` to `true`.

4. Add a button for users to log in with Casdoor, and modify the data's `loginForm`.

```
<a href="http://localhost:7001/login/oauth/authorize?client_id=d509b6b3edc8a3d4cce9&response_type=code&redirect_uri=http%3A%2Flocalhost%3A7001%2Flogin/oauth/callback" data-bbox="154 682 880 705">casdoor</a>

loginForm: {
  code: "",
  state: ""
},
```

Here, I have written the URL, but you can obtain it using the Casdoor-Vue-SDK or Casdoor-SpringBoot-SDK.

5. Since we are no longer using the original login method, delete the cookie and checkcode methods.

The new `created` function should look like this:

```
created() {
  let url = window.document.location.href; // Get the URL
  let u = new URL(url);
  this.loginForm.code = u.searchParams.get('code'); // Get
  the code and state
  this.loginForm.state = u.searchParams.get('state');
  if (this.loginForm.code != null && this.loginForm.state !=
  null) { // If code and state are not null, execute handleLogin
    this.handleLogin();
  }
}
```

6. In fact, we only need to change the parameter we send to the back-end and delete the unnecessary functions. No other changes are necessary.

```
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. Import the required dependency in the back-end.

```
pom.xml

<dependency>
  <groupId>org.casbin</groupId>
  <artifactId>casdoor-spring-boot-starter</artifactId>
  <version>1.2.0</version>
</dependency>
```

You also need to configure Casdoor in the resource file.

8. Define the callback function as the redirect function. Make changes to some methods in `sysLoginService`. Delete the password check step because it is no longer needed.

```
@PostMapping("login")
public R<?> callback(@RequestBody CodeBody code) {
    // Define a CodeBody entity with 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 the user is not in the RuoYi-Cloud database
            but exists in Casdoor, create the user in the database
            sysLoginService.casdoorRegister(casdoorUserName);
        }
    }
    LoginUser userInfo =
sysLoginService.casdoorLogin(casdoorUser.getName());
    // Get the user's information from the database
    return R.ok(tokenService.createToken(userInfo));
}
```

9. Add new methods to `SysLoginService`.

```
public LoginUser casdoorLogin(String username) {
    R<LoginUser> userResult =
```

```

public String getUserByCasdoorName(String casdoorUsername) {
    R<LoginUser> userResult =
remoteUserService.getUserInfo(casdoorUsername,
SecurityConstants.INNER);
    if (StringUtils.isNull(userResult) ||
StringUtils.isNull(userResult.getData())) {
        // If the user is not in the RuoYi-Cloud database but
exists in Casdoor, create the user in the database
        return null;
    }
    String username =
userResult.getData().getSysUser().getUserName();
    return username;
}

```

```

public void casdoorRegister(String username) {
    if (StringUtils.isAnyBlank(username)) {
        throw new ServiceException("User must provide a
username");
    }
    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, "Registration successful");
}

```


Pulsar Manager

Casdoor can easily connect to Pulsar Manager.

The code for connecting Casdoor has already been added to Pulsar Manager, so we just need to configure the `application.yml` file in the back-end and enable the front-end switch.

Step 1: Deploy Casdoor

First, deploy Casdoor.

You can refer to the official Casdoor documentation for the [Server Installation](#).

After a successful deployment, ensure the following:

- The Casdoor server is running successfully at <http://localhost:8000>.
- Open your favorite browser and visit <http://localhost:7001>. You should see the login page of Casdoor.
- Test the login functionality by entering `admin` and `123`.

Now, you can quickly implement a Casdoor-based login page in your own app using the following steps.

Step 2: Configure Casdoor

To configure Casdoor, refer to [Casdoor](#) (it is recommended to use a different browser than your development browser).

You should also configure an organization and an application. You can refer to


[Casdoor](#) for detailed instructions.

Step 2.1: Create an organization

Edit Organization

Name

Display name

Favicon
Preview: 

Website URL

Password type


Password salt

Phone prefix

Step 2.2: Create an application

Name

Display name

Logo
Preview: 

Home

Description

Organization

Client ID

Client secret

Cert

Redirect URLs
Redirect URL

Step 3: Enable the Pulsar Manager front-end switch

Enable this switch to send code and state to the back-end.

You can find the switch on line 80 of `pulsar-manager/front-end/src/router/index.js`.

```
- // mode: 'history', // require service support  
+ mode: 'history', // require service support
```

Step 4: Configure the back-end code

Configure Casdoor's settings in the `application.properties` file, which can be found on line 154 of `pulsar-manager/src/main/resources/application.properties`.

```
casdoor.endpoint = http://localhost:8000  
casdoor.clientId = <client id from previous step>  
casdoor.clientSecret = <client secret from previous step>  
casdoor.certificate = <client certificate from previous step>  
casdoor.organizationName = pulsar  
casdoor.applicationName = app-pulsar
```


Using Casdoor in ShenYu

ShenYu has a Casdoor plugin to enable the use of Casdoor.

Step 1: Deploy Casdoor

Firstly, Casdoor should be deployed. You can refer to the official Casdoor documentation for [Server Installation](#).

After a successful deployment, please ensure that:

- The Casdoor server is running on <http://localhost:8000>.
- Open your preferred browser and visit <http://localhost:7001> to see the Casdoor login page.
- Login functionality is working fine by inputting `admin` and `123`.


After following the above steps, you can quickly implement a Casdoor-based login page in your app with the following steps.

Step 2: Configure the Casdoor application

1. Create a new Casdoor application or use an existing one
2. Add your redirect URL

Name: app-test → application name

Display name: app-test

Logo: 

URL: https://cdn.casbin.org/img/casdoor-logo_1185x256.png

Home: /

Description:

Organization: built-in → organization name

Client ID: 6e3a84154e73d1fb156a → client id

Client secret: a42095412a33a842b7a9c05a3446e623cb07262d → client secret

Cert: cert-built-in

Redirect URLs:

Redirect URL	Action
/ http://localhost:9195/http/hi	↕
/ http://localhost:9195/http/hello	↕

→ redirect url

3. On the certificate editing page, you can view your **Certificate**

Certificate: Copy certificate Download certificate

```

-----BEGIN CERTIFICATE-----
MIIE+TCCAuGgAwIBAgIDAeJAMA0GCSqGSIb3DQEBCwUAMDYxHTAbBgNVBAoTFENh
c2Rvb3IgT3JnYW5pemF0aW9uMRUwEwYDVQQDEwxDYXNkb29yIENlcnQwHhcNMjEx
MDE1MDgxMTUyWhcNNDExMDE1MDgxMTUyWjA2MR0wGwYDVQQKEwRDXNkb29yE9y
Z2FuaXphdGlvbEVVBMGA1UEAxMMQ2FzZG9vciBDZXJ0MIIICjANBgkqhkiG9w0B
AQEFAAOCAg8AMIICGKCAgEASlnpb5E1/yM0f1RfSDSSE8IR7y+lw+Rjji74e5ej
rq4b8zMYk7HeHCyZr/hmNEwEVXnhXu1P0mBeQ5ypp/QGo8vvgEmjAETNmzkl1NjOQ
CjCYwUrasO/f/Mnl1COj13vx6mV1kHZjSrKsMhYY1vaxTEP3+VB8Hjg3MHFWrb07
uvFMCJe5W8+0rErZCKTR8+9VB3janeBz//zQePFVh79bFzate/hLirPK0Go9P1g
OvwloC1A3sarHTP4Qm/LQRt0rHqZFybdySpyWAQvhNaDFE7mTstRSBb/wUjNCUBD
PTSLVjC04WllSf6Nkfx0Z7KvmbPstSj+btvcqsvRAGtvdsB9h62Kptjs1Yn7GAuo
I3qt/4zoKbiURYxkQjXlVwCQsEftUuk5ew5zuPSIDRL0LByQTLbx0JqLAFNfW3g/
pzSDjgd/60d6HTmVbZni4SmjdyFhXCdb1Kn7N+xTojfaNkwep2REV+RMc0fx4Gu
hRsnLsmkmUDeylZ9aBL9oj11YEQfM2JZEj+RvtUx+wB4y8K/tD1bcY+IfnG5rBpw
IDpS262boq4SR5vb3Z7bB0w4ZxvOfj/1VLoRftjPbLif0bhfr/AeZMHplK0Xvzf4
yE+hqzi68wdf0VR9xYc/RbSAf7323OsjYnjEginUtRohnRgCpjik/Mt2Kt84Kb0
wn8CAwEAAAMQMA4wDAYDVR0TAQH/BAIwADANBgkqhkiG9w0BAQsFAAOCAgEAn2lf
DKkLX+F1vKRO/5gl+Plr8P5NKuQkmwH97b8CS2gS1phDyNgic4/LSdzuf4Awe6ve
C06lVdWSiis8UPUPdjmT2uMPSNjwLxG3QsrMURNwFLtFRem/heJe0Zgur9J1M
8haawdSdjh2RgmFoDeE2r8NVRfhr8KnCO1ddTJKuS1N0/irHz21W4jt4rxzCvl
2nR42Fybap3O/g2JXMhNNR0wZmNjggsF7XVENCSuFO1jTywLaquXCg54IL7XVLG
omKNNNcc8h1FCeKj/nnbGMhodnFWKDTsJcbNmOPNH06ixzqMy/Hqc+mWYv7maAG
Jtevs3qgMZ8F9Qzr3HpUc6R3ZYWWDY/xxPisuKftOPZgtH979XC4mdf0WPnOBLqL
2DJ1za8mjiGJolvb7XNVKcUfDXyw85ZTZQ5b9cl4e+6bmyWqQitlwt+Ati/uFEV
XzCj70B4lALX6xau1kLEpV9O1GERizYrZ5P9NjNA7KoO5AVMp9w0DQTKt+LbXnZE
HHnWKy8xHQKZF9sR7YBPGLs/Ac6tviv5Ua15Ogj/8dLRZ/veyFfGo2yZsi+hKVU5
nCCJHBCAyFnm1hdvvdEdH33jDBjNB6ciotJzrf/3VYalWSalADosHAgMwfxuWP+h
8XXmzlxuHbTMQytZPDgspS5aK+S4Q9wb8RRAYo=
-----END CERTIFICATE-----

```

Step 3: Use the Casdoor plugin in

ShenYu

1. Configure the Casdoor plugin in ShenYu

Plugin ✕

* Plugin:

casdoor Configuration

* application-name:

* certificate:

* client_id:

* client_secret:

* casdoor endpoint:

* organization-name:

* Role:

* Sort:

Status:

Note: As ShenYu only has a single line input box, `\n` must be added in every line


of certificate.

Certificate ⓘ :

Copy certificate

Download certificate

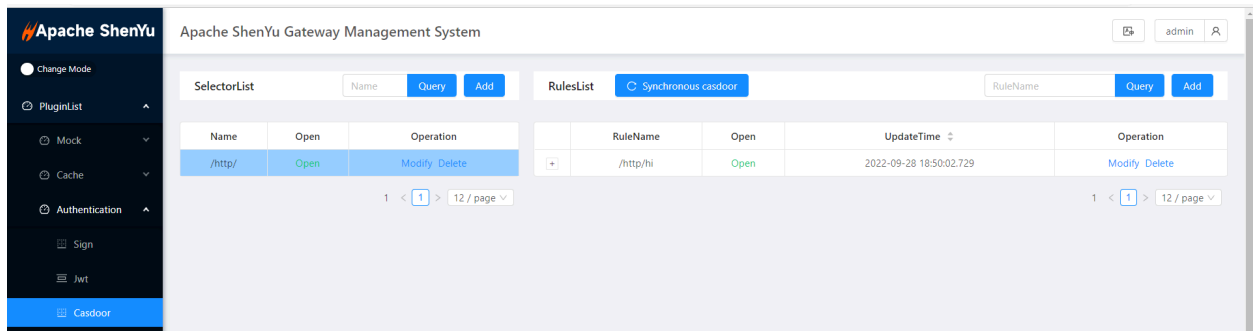
```
-----BEGIN CERTIFICATE-----\n
MIIE+TCCAuGgAwIBAgIDAeJAMA0GCSqGSIb3DQEBCwUAMDYxHTAbBgNVBAoTFENh\n
c2Rvb3IgT3JnYW5pemF0aW9uMRUwEwYDVQQDEwxDYXNkb29yLENicnQWHhcNMjEx\n
MDE1MDgxMTUyWhcNNDE1MDgxMTUyWjA2MR0wGwYDVQQKExRDYXNkb29yLE9y\n
Z2FuaXphdGlvbjEVMBMGA1UEAxMMQ2FzZG9vciBDZXJ0MIIIClJANBgkqhkiG9w0B\n
AQEFAAOCAg8AMIICGgKCAgEAsInpb5E1/ym0f1RfSDSSE8IR7y+lw+RjJl74e5ej\n
rq4b8zMYk7HeHCyZr/hmNEwEVXnhXu1P0mBeQ5ypp/QGo8vgEmjAETNmzkl1NjOQ\n
CjCYwUrasO/f/Mnl1C0j13vx6mV1kHZjSrKsMhYY1vaxTEP3+VB8Hjg3MHFWrb07\n
uvFMCJe5W8+0rKERZCKTR8+9VB3janeBz//zQePFVh79bFZate/hLirPK0Go9P1g\n
OvwloC1A3sarHTP4Qm/LQRt0rHqZFybdySpyWAQvhNaDFE7mTstRSBb/wUjNCUBD\n
PTSLVjC04WIIISf6Nkfx0Z7KvmbPstSj+btvcqsvRAGtvdsB9h62Kpts1Yn7GAuo\n
l3qt/4zoKbiURYxkQJXlvwCQsEftUuk5ew5zuPSIDRLoLByQTLbx0JqLAFNfW3g\n
pzSDjgd/60d6HTmVbZni4SmjdyFhXCdb1Kn7N+xTojnfaNkwep2REV+RMc0fx4Gu\n
hRsnLsmkmUDeylZ9aBL9oj11YEQfM2JZEq+RVtUx+wB4y8K/tD1bcY+IfnG5rBpw\n
IDpS262boq4SR5vb3Z7bB0w4ZxvOfJ/1VLoRftjPbLif0bhfr/AeZMHplK0Xvzf4\n
yE+hqzi68wdF0VR9xYc/RbSAf7323OsjYnjEgInUtRohnRgCpjlk/Mt2Kt84Kb0\n
wn8CAwEAAAMQMA4wDAYDVR0TAQH/BAIwADANBgkqhkiG9w0BAQsFAAOCAgEAn2If\n
DKkLX+F1vKRO/5gJ+Plr8P5NKuQkmwH97b8CS2gS1phDyNglc4/LSdzuf4Awe6ve\n
C06iVdWSlis8UPUPdjmT2uMPSNjwLxG3QsrMURNwFILTfRem/heJe0Zgur9J1M\n
8haawdSdJjH2RgmFoDeE2r8NVRfhbR8KnCO1ddTJKuS1N0/irHz21W4jt4rxzCv\n
2nR42Fyap3O/g2JXMhNNROwZmNjgpsF7XVENCsuFO1jTywLaquXCg54IL7XVLG\n
omKNNNcc8h1FCeKj/nnbGMhodnFWKDTsJcbNmcOPNH06ixzqMy/Hqc+mWYv7maAG\n
Jtevs3qgMZ8F9Qzr3HpUc6R3ZYWDY/xxPisuKftOPZgtH979XC4mdf0WPnOBLqL\n
2DJ1zaBmjGJolvb7XNVKcUfDXyW85ZTzQ5b9c1l4e+6bmyWqQtltw+Ati/uFEV\n
XzCj70B4IALX6xau1kLEpV9O1GERizYRz5P9NjNA7KoO5AVMp9w0DQTKt+LbXnZE\n
HHnWKy8xHQKZF9sR7YBGLs/Ac6tviv5Ua15OgJ/8dLRZ/veyFfGo2yZsl+hKVU5\n
nCCJHBcAyFnm1hdvvdEdH33jDBjNB6ciotJzrf/3VYalWSalADosHAgMWfXuWP+h\n
8XKXmzlXuHbTMQYtZPDgspS5aK+S4Q9wb8RRAYo=\n
-----END CERTIFICATE-----
```

 here not need add \n

You can copy it and paste it into the certificate of the ShenYu Casdoor config.

You don't need to save it in the Casdoor certificate editing page, as it is only for copying.

2. Configure the ShenYu Casdoor plugin

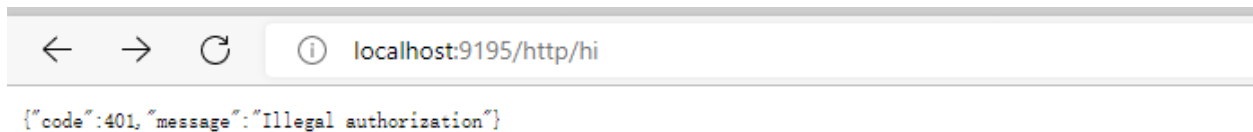


The screenshot shows the Apache ShenYu Gateway Management System interface. On the left is a dark sidebar with the 'Apache ShenYu' logo and a menu including 'Change Mode', 'PluginList', 'Mock', 'Cache', 'Authentication', 'Sign', 'Jwt', and 'Casdoor'. The main area is titled 'Apache ShenYu Gateway Management System' and contains two tables. The 'SelectorList' table has columns for Name, Open, and Operation, with a row for '/http/'. The 'RulesList' table has columns for RuleName, Open, UpdateTime, and Operation, with a row for '/http/hi'. Both tables have pagination controls at the bottom.

You can configure what you need for the Casdoor config.

3. Getting the service and using it

3.1 Directly visit the Web



The screenshot shows a web browser address bar with the URL 'localhost:9195/http/hi'. Below the address bar, the browser displays a JSON error response: `{"code":401,"message":"Illegal authorization"}`.

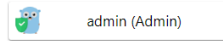
3.2 Use Casdoor Login

localhost:7001/login/oauth/authorize?client_id=6e3a84154e73d1fb156a&response_type=code&redirect_uri=http://localhost:9195/http/hi&scope=read&state=app-test

🔍 🌐



Continue with :



Or sign in with another account :

🔍 username, Email or phone

🔒 Password

Auto sign in

[Forgot password?](#)

Sign In

[No account? sign up now](#)



localhost:9195/http/hi?code=822607b015cca2515b2b&state=app-test

hi! null! I'm Shenyu-Gateway System. Welcome!

3.3 Carry the token in Headers

The screenshot shows the Postman interface for a GET request to `http://localhost:9195/http/hi`. The **Headers** tab is selected, displaying a table of headers:

Key	Value	Description
<input checked="" type="checkbox"/> Postman-Token	<calculated when request is sent>	
<input checked="" type="checkbox"/> Host	<calculated when request is sent>	
<input checked="" type="checkbox"/> User-Agent	PostmanRuntime/7.29.2	
<input checked="" type="checkbox"/> Accept	*/*	
<input checked="" type="checkbox"/> Accept-Encoding	gzip, deflate, br	
<input checked="" type="checkbox"/> Connection	keep-alive	
<input checked="" type="checkbox"/> Authorization	eyJhbGciOiJIJSuz1NiIsImtpZCI6ImNlcnQtYn...	Your token

Below the headers, the **Body** tab is active, showing the response in **Text** format:

```
1 hi! null! I'm Shenyu-Gateway System. Welcome!
```

3.4 Save name, ID and organization in Headers

This makes it easier to use them in the future.

ShardingSphere

[shardingsphere-elasticjob-ui](#) has integrated Casdoor. You can use it after configuring it.

Step 1: Deploy Casdoor

Firstly, Casdoor should be deployed.

You can refer to the Casdoor official documentation for the [Server Installation](#).

After a successful deployment, make sure:

- 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 if the login functionality is working fine.

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


Step 2: Configure Casdoor application and configure application in ShardingSphere

1. Create or use an existing Casdoor application

Name: ShardingSphere

Display name: ShardingSphere

Logo: URL: https://cdn.casbin.org/img/casdoor-logo_1185x256.png

Preview: 

Home: /

Description:

Organization: ShardingSphere

Client ID: 3ed79fa530645fbd3653

Client secret: 54633c82b7796a4332c6976864c6c16bc3b05556

Cert: cert-built-in

Redirect URLs:

Redirect URL	Action
http://localhost:8080	<input type="button" value="Add"/> <input type="button" value="Up"/> <input type="button" value="Down"/> <input type="button" value="Delete"/>

The RedirectURLs depend on the URL you need to redirect to. The selected data will be used in the next step.

2. On the certificate editing page, you can see your **Certificate**

Certificate ⓘ :

Copy certificate

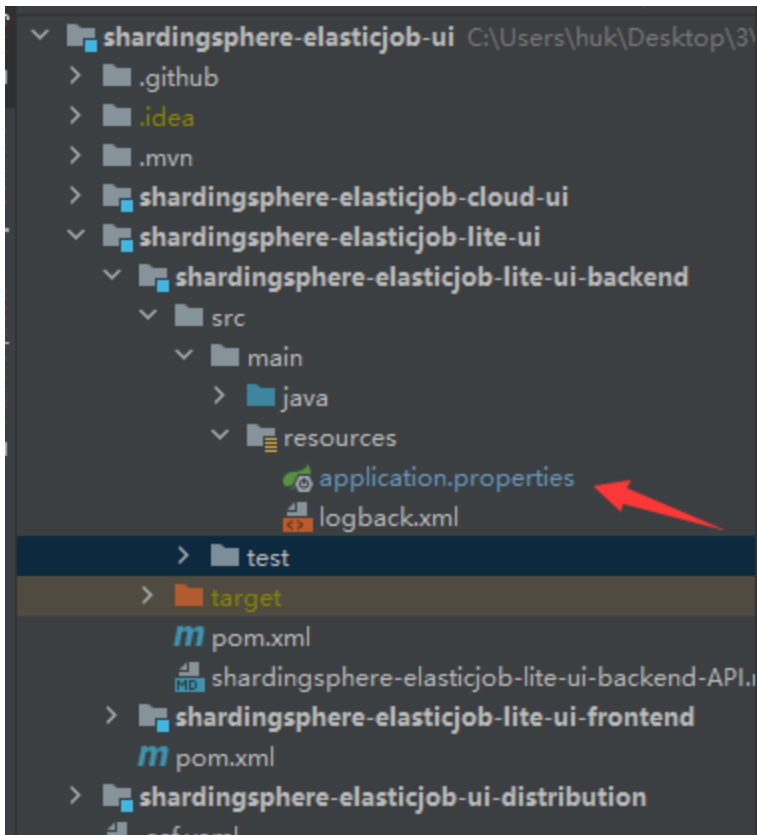
Download certificate

Private

```
-----BEGIN CERTIFICATE-----
MIIE+TCCAuGgAwIBAgIDAeJAMA0GCSqGSIb3DQEBCwUAMDYxHTAb8gNVBAoTFENh
c2Rvb3IgT3JnYW5pemF0aW9uMRUwEwYDVQQDEwxDYXNkb29yIENlcnQwHhcNMjEx
MDE1MDgxMTUyWhcNNDExMDE1MDgxMTUyWjA2MR0wGwYDVQQKEwRDXNkb29yIE9y
Z2FuaXphdGlvbjEVMjBGA1UEAxMMQ2FzZG9vciBDZXJ0MIIICjANBgkqhkiG9w0B
AQEFAAOCAg8AMIICGKCAgEAsInpb5E1/ym0f1RfSDSSE8IR7y+lw+RjI74e5ej
rq4b8zMYk7HeHCyZr/hmNEwEVXnhXu1P0mBeQ5ypp/QGo8vgEmjAETNmzkI1NjOQ
CjCYwUrasO/f/MnI1C0j13vx6mV1kHZjSrKsMhYY1vaxTEP3+VB8Hjg3MHFWrb07
uvfMCJe5W8+0rKErZCKTR8+9VB3janeBz//zQePFVh79bFZate/hLirPK0Go9P1g
OvwloC1A3sarHTP4Qm/LQRt0rHqZFybdySpyWAQvhNaDFE7mTstRSBb/wUjNCUBD
PTSLVjC04WII5f6Nkfx0Z7KvmbPstSj+btvcqsvRAGtvsB9h62Kptjs1Yn7GAuo
l3qt/4zoKbiURYxkQJXlvwCQsEftUuk5ew5zuPSIDRLoLByQTLbx0JqLAFNfW3g/
pzSDjgd/60d6HTmVbZni4SmjdyFhXCdb1Kn7N+xTojnfaNkwep2REV+RmC0fx4Gu
hRsnLsmkmUDeylZ9aBL9oj11YEQfM2JZEQ+RVtUx+wB4y8K/tD1bcY+IfnG5rBpw
IDpS262boq4SRsVb3Z7b80w4ZxvOfJ/1VLoRftjPbLif0bhfr/AeZMhplKOXvfz4
yE+hqzi68wdf0VR9xYc/RbSAf7323OsjYnjEgInUtRohnRgCpjlk/Mt2Kt84Kb0
wn8CAwEAAAMQAA4wDAYDVR0TAQH/BAIwADANBgkqhkiG9w0BAQsFAAOCAgEAn2If
DKkLX+F1vKRO/5gJ+Plr8P5NKuQkmwH97b8CS2gS1phDyNgIc4/LSdzuf4Awe6ve
C06IVdWslis8UPUPdjmT2uMPSNjwLxG3QsrimMURNwFILTfRem/heJe0Zgur9J1M
8haawdSdJjH2RgmFoDeE2r8NVRfhhR8KnCO1ddTJKuS1N0/irHz21W4jt4rxzCvl
2nR42Fybap3O/g2JXMhNNROwZmNjgppsF7XVENCsuFO1jTywLaqjuXCg54IL7XVLG
omKNNNcc8h1FCeKj/nnbGMhodnFWKDTsJcbNmcOPNHo6ixzqMy/Hqc+mWYv7maAG
Jtevs3qgMZ8F9Qzr3HpUc6R3ZYWDY/xxPisuKftOPZgtH979XC4mdf0WPnOBLqL
2DJ1zaBmjiGJolvb7XNVKcUfDXyW85ZTZQ5b9cll4e+6bmyWqQltwt+Ati/uFEV
XzCj70B4IALX6xau1kLEpV9O1GERizYRz5P9NjNA7KoO5AVMp9w0DQTKt+LbXnZE
HHnWky8xHQKZF9sR7YBPGLs/Ac6tviv5Ua15OgJ/8dLRZ/veyFfGo2yZsl+hKVU5
nCCJHBcAyFnm1hdvdwEdH33jdBjNB6ciotJZrf/3VYalWSalADosHAgMWFxUWP+h
8XKXmzlxuHbTMQYtZPDgspS5aK+S4Q9wb8RRAYo=
-----END CERTIFICATE-----
```

3. Configure the application in ShardingSphere

First, we need to find the application.properties that we need to configure.



Next, we need to copy the data from the Casdoor application and paste it into the application.



Login

Login with Casdoor

Apache IoTDB

Casdoor can easily connect to [Apache IoTDB](#).

The code for connecting Casdoor has already been added in [Apache IoTDB Web Workbench](#), so all we need to do is configure the application.yml file in the back-end and activate the front-end switch.

Step 1: Deploy Casdoor

First, deploy Casdoor.

You can refer to the official Casdoor documentation for the [Server Installation](#).

After deploying successfully, ensure that:

- The Casdoor server is running successfully at <http://localhost:8000>.
- Open your preferred browser and visit <http://localhost:7001>, where you will see the Casdoor login page.
- Test the login functionality by entering `admin` and `123`.

With these steps completed, you can now quickly implement a Casdoor-based login page in your own application.


Step 2: Configure Casdoor


To configure Casdoor, refer to [casdoor](#) (It is recommended not to use the same browser you are developing in to configure Casdoor's browser).



You should also create an organization and an application. Refer to [casdoor](#) for


instructions.


2.1 Create an organization


Name :


Display name :

Favicon : URL :

Preview: 

Website URL :

Password type :


Password salt :

2.2 Create an application

Name :

Display name :

Logo : URL :

Preview: 

Home :

Description :

Organization :

Client ID :

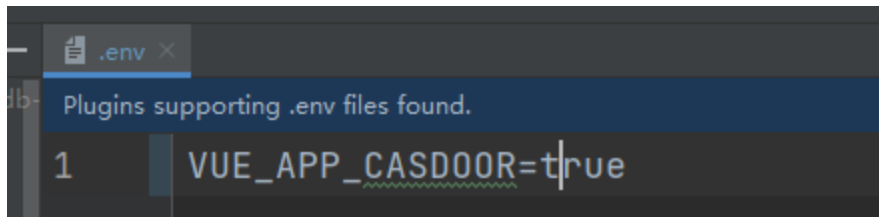
Client secret :

Cert :

Step 3: Activate Apache IoTDB Web Workbench front-end switch

Open this switch to send the code and state to the back-end.

This switch can be found in `iotdb-web-workbench/fronted/.env` file.



```
.env x
Plugins supporting .env files found.
1 VUE_APP_CASDOOR=true
```

Step 4: Configure the back-end code

You need to configure Casdoor's settings in the `iotdb-web-workbench/backend/src/main/resources/application.properties` file.

```
casdoor.endpoint = http://localhost:8000
casdoor.clientId = <client id from previous step>
casdoor.clientSecret = <client secret from previous step>
casdoor.certificate=<client certificate from previous step>
casdoor.organizationName=IoTDB
casdoor.applicationName=app-IoTDB
```

Result

The screenshot shows the login interface for IoTDB WorkBench. The page layout includes a top navigation bar with the logo and a user profile icon. A vertical image on the left side depicts industrial equipment. The central content area contains the following elements:

- Header:** IoTDB WorkBench logo and "Welcome To IoTDB WorkBench" text.
- Account Field:** A text input field with the placeholder "Please Input Account".
- Password Field:** A password input field with a light blue background and masked characters (dots).
- Buttons:** A green "Sign In" button and a green "Sign In With Casdoor" button.
- Links:** A "Forget Password?" link located to the right of the password field.

Apache DolphinScheduler

Casdoor is one of the supported login methods for [Apache DolphinScheduler](#).

Step 1: Deploy Casdoor

Firstly, Casdoor should be deployed. You can refer to the Casdoor official documentation for [Server Installation](#).

After a successful deployment, please ensure that:

- The Casdoor server is running successfully at <http://localhost:8000>.
- Open your favorite browser and visit <http://localhost:7001>. You will see the login page of Casdoor.
- Test the login functionality by inputting "admin" and "123".

Once the deployment is completed, you can quickly implement a Casdoor-based login page in your own app by following the steps below.

Step 2: Configure Casdoor Application

1. Create a new Casdoor application or use an existing one.
2. Add your redirect URL (You can find more details about how to obtain the redirect URL in the next section).

The screenshot shows the Casdoor application settings page. The following fields are highlighted with red boxes and labeled with red arrows:

- Name:** application_a6ftas → your application name
- Display name:** New Application - a6ftas
- Logo URL:** https://cdn.casbin.org/img/casdoor-logo_1185x256.png
- Preview:** Casdoor logo (a blue cube icon and the text "Casdoor" with a star above the 'o')
- Home:** [empty]
- Description:** [empty]
- Organization:** organization_carg1b → your organization name
- Client ID:** 3ed7314825ecf955cb19 → your client id
- Client secret:** ee9314ea228 [blurred] → your client secret
- Cert:** cert-built-in
- Redirect URLs:** [empty]
- Redirect URL:** http://localhost:3000/callback → your redirect url

3. Add the desired provider and fill in other necessary settings.

On the application settings page, you will find two important values: `Client ID` and `Client secret`, as shown in the picture above. We will use these values in the next step.

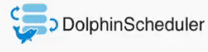
Open your favorite browser and visit `http://CASDOOR_HOSTNAME/.well-known/openid-configuration` to view the OIDC configuration of Casdoor.

Step 3: Configure DolphinScheduler

dolphinscheduler-api/src/main/resources/application.yaml

```
security:
  authentication:
    # Authentication types (supported types: PASSWORD, LDAP,
    CASDOOR_SSO)
    type: CASDOOR_SSO
  casdoor:
    # The URL of your Casdoor server
    endpoint:
    client-id:
    client-secret:
    # The certificate may be multi-line; you can use `|-` for ease
    certificate:
    # The organization name you added in Casdoor
    organization-name:
    # The application name you added in Casdoor
    application-name:
    # The DolphinScheduler login URL
    redirect-url: http://localhost:5173/login
```

Now, DolphinScheduler will automatically redirect you to Casdoor for authentication.



SSO Login

FireZone

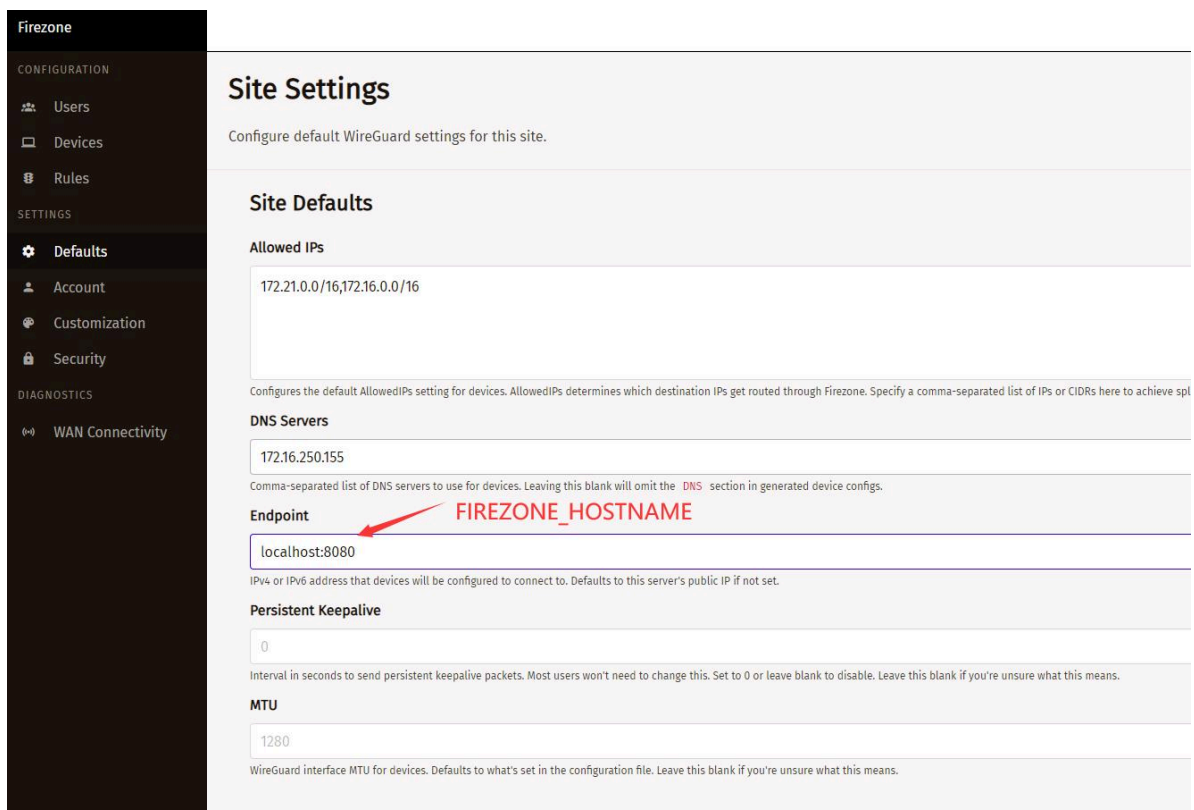
Casdoor can use the OIDC protocol as the IDP to connect various applications. Here, we will use FireZone as an example to show you how to use OIDC to connect to your applications.

Step 1: Deploy Casdoor and FireZone

Firstly, Casdoor and FireZone should be deployed.

After a successful deployment, ensure the following:

1. Set the FireZone URL (Signin → Security → Add OpenID Connect Provider) to FIREZONE_HOSTNAME.



The screenshot shows the Firezone web interface. On the left is a dark sidebar with a navigation menu. The main content area is titled "Site Settings" and contains several configuration sections:

- CONFIGURATION**
 - Users
 - Devices
 - Rules
- SETTINGS**
 - Defaults** (highlighted)
 - Account
 - Customization
 - Security
- DIAGNOSTICS**
 - WAN Connectivity

The "Site Defaults" section includes the following fields:

- Allowed IPs:** 172.21.0.0/16,172.16.0.0/16
- DNS Servers:** 172.16.250.155
- Endpoint:** localhost:8080. A red arrow points to this field with the label "FIREZONE_HOSTNAME".
- Persistent Keepalive:** 0
- MTU:** 1280

Below each field is a brief description of its function.

2. Casdoor can be logged in and used normally.
3. `CASD00R_HOSTNAME`: <http://localhost:8000>, if you deploy Casdoor using the default `app.conf`.

Step 2: Configure Casdoor application

1. Create a new Casdoor application or use an existing one.
2. Add a redirect URL:

For example, if the Configid in the FireZone Provider is TEST, the redirect URL should be `http://[FIREZONE_HOST]/auth/oidc/[PROVIDER_CONFIG_ID]/callback/`.

The screenshot shows the configuration page for a Casdoor application. The fields are as follows:

- Home: `http://localhost:8080`
- Description: (empty)
- Organization: `built-in`
- Client ID: `0159c45127541d48e433`
- Client secret: `add1be9982640e048fcf46770d75674b918484af`
- Cert: `cert-built-in`
- Redirect URLs: A table with one entry:

Redirect URL
<code>http://localhost:8080/auth/oidc/TEST/callback/</code>

Open your favorite browser and visit: `http://[CASD00R_HOSTNAME]/.well-known/openid-configuration`, and you will see the OIDC configuration of Casdoor.

3. Configure FireZone: Security → Add OpenID Connect Provider

OIDC Config ✕

Config ID

Label

Scope

Response type

Client ID

Client secret

Discovery Document URI

Auto create users

Save

ConfigID should be the PROVIDER_CONFIG_ID of the redirect URL

- **Discovery Document URI**: The FireZone Provider Discovery Document URI should be `https://[CASD00R_HOST]/.well-known/openid-configuration`.
- **Scopes**: `openid email profile`
- **ConfigID**: The ConfigID should be the PROVIDER_COONFIG_ID of the

redirect URL and should correspond to the Casdoor redirect URL.

- `Auto-create users`: Successful login will automatically create a user.

Log out of FireZone and test SSO



Sign In

Please sign in via one of the methods below.

Sign in with TEST

Sign in with email

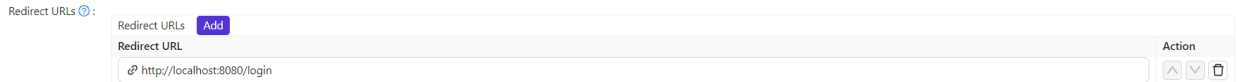
Cloud Foundry

Before the integration, we need to deploy Casdoor locally.

Then, we can quickly implement a Casdoor-based login page in our own app with the following steps.

Step 1: Configure Casdoor application

1. Create or use an existing Casdoor application.
2. Add a redirect URL: `http://CASD00R_HOSTNAME/login`



3. Copy the client ID; we will need it in the following steps.

Step 2: Add a user in Casdoor

Now that you have the application, but not a user, you need to create a user and assign the role.

Go to the "Users" page and click on "Add user" in the top-right corner. This opens a new page where you can add the new user.

Save the user after adding a username and the organization "Cloud Foundry" (other details are optional).

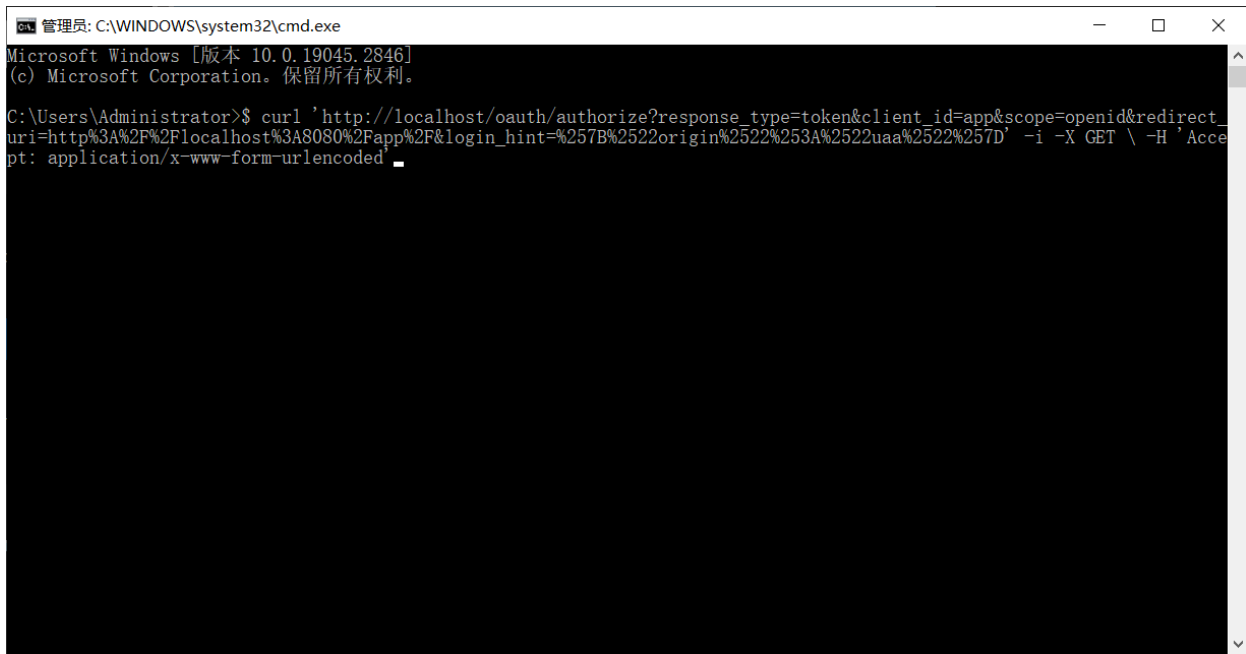
Now, you need to set up a password for your user, which you can do by clicking on "Manage your password".

Choose a password for your user and confirm it.

Step 3: Build the Cloud Foundry App

Start the Cloud Foundry by following these steps.

- `$ git clone git://github.com/cloudfoundry/uaa.git`
- `$ cd uaa`
- `$./gradlew run`



```
管理员: C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.19045.2846]
(c) Microsoft Corporation。保留所有权利。

C:\Users\Administrator>$ curl 'http://localhost/oauth/authorize?response_type=token&client_id=app&scope=openid&redirect_uri=http%3A%2F%2Flocalhost%3A8080%2Fapp%2F&login_hint=%257B%2522origin%2522%253A%2522uaa%2522%257D' -i -X GET \ -H 'Accept: application/x-www-form-urlencoded'
```

Step 4: Integrate Casdoor

Now open another command line and input:

```
curl '<http://localhost/oauth/authorize?response_type=token&client_id=app&scope=openid&redirect_uri=http%3A%2F%2Flocalhost%3A8080%2Fapp%2F>' -i -X GET \ -H 'Accept: application/x-www-form-urlencoded'
```

We have already obtained the client ID and redirect URI before; we input these parameters.

Parameter	Type	Constraints	Description
response_type	String	Required	Space-delimited list of response types. Here, token, i.e. an access token
client_id	String	Required	a unique string representing the registration information provided by the client
scope	String	Optional	requested scopes, space-delimited
redirect_uri	String	Optional	redirection URI to which the authorization server will send the user-agent back once access is granted (or denied), optional if pre-registered by the client

Execute the command, and we can get the result below, which means that we have successfully integrated Casdoor with Cloud Foundry.


```

HTTP/1.1 302 Found
Content-Security-Policy: script-src 'self'
Strict-Transport-Security: max-age=31536000
Set-Cookie: X-Uaa-Csrft=09mMqMDhcwHGLMufnB4YA1; Path=/; Max-Age=86400; Expires=Fri, 5 May 2023 14:53:54 GMT; HttpOnly; SameSite=Lax
Cache-Control: no-store
Content-Language: en
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Location: http://localhost:8080/app/#token_type=bearer&access_token=eyJhbGciOiJIUzI1NiIsImprdiI6Imh0dHBzOi8vbG9jYXRob3N0OjgWODAvdWVhL3Rva

```

Built-in Organization x +

localhost:8000/login




username, Email, or phone

Password

Auto sign in [Forgot password?](#)

[Sign In](#)

[No account? sign up now](#)

Powered by  Casdoor

+

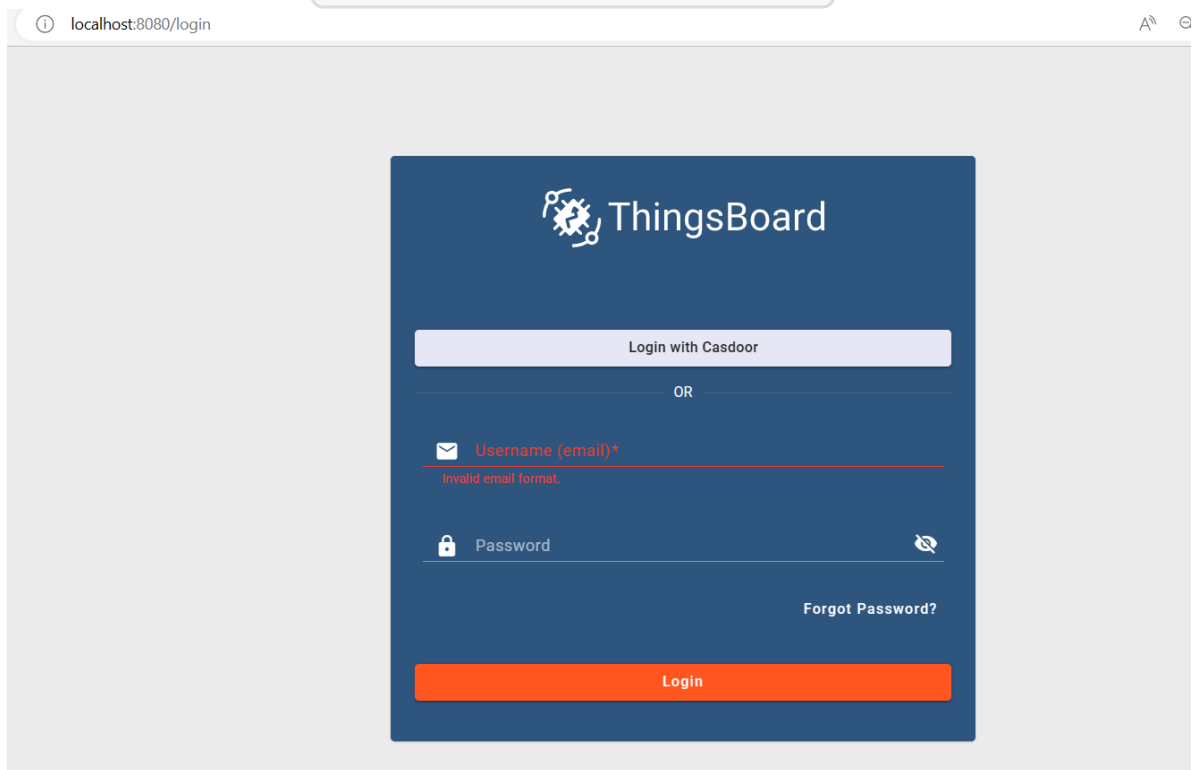
Thingsboard

Before the integration, we need to deploy Casdoor locally.

Then, we can quickly implement a Casdoor-based login page in our own app by following these steps.

Step 1: Configure Casdoor application

1. Create a new Casdoor application or use an existing one.
2. Add a redirect URL: `http://CASD00R_HOSTNAME/login`



3. Copy the client ID and client secret. We will need them in the following steps.

Step 2: Add a user in Casdoor

Now that you have the application, you need to create a user and assign a role.

Go to the "Users" page and click on "Add user" in the top right corner. This will open a new page where you can add the new user.

Save the user after adding a username and selecting the organization "Thingsboard" (other details are optional).

Next, you need to set up a password for the user. You can do this by clicking on "Manage your password".

Choose a password for the user and confirm it.

Step 3: Prerequisites and Build Thingsboard App

First of all, Thingsboard only supports Java 11 (OpenJDK).

You can download it from the following link:

[JDK Download Page](#)

To start Thingsboard, follow these steps (for Windows system):

- Download and extract the package. [Download the package](#)
- Configure Thingsboard in `\thingsboard\conf\thingsboard.yml` according to your preferences, including the configuration of Kafka, PostgreSQL, and others.

- Run `install.bat -loadDemo` in the command line in the Thingsboard folder to install and add demo data.

```
C:\Program Files (x86)\thingsboard>install.bat --loadDemo
Detecting Java version installed.
CurrentVersion 110
Java 11 found!
Installing thingsboard ...
...
ThingsBoard installed successfully!
```

- Run `net start thingsboard` in the command line to start Thingsboard. You should see the following output:

```
The ThingsBoard Server Application service is starting.
The ThingsBoard Server Application service was started successfully.
```

Step 4: Integrate Casdoor

Now open <http://localhost:8080/> and log in to the admin account:



Account: sysadmin@thingsboard.org / Password: sysadmin

After successfully logging in, click the OAuth2 button at the bottom left of the page.

The screenshot shows the ThingsBoard Home dashboard. On the left is a dark blue sidebar with navigation items: Home, Tenants, Tenant profiles, Resources, Notification center, Settings, Security, General, Two-factor authentication, and OAuth2. The main content area is light gray and contains several widgets. At the top, there are two large widgets for 'Tenants' (count 2) and 'Tenant profiles' (count 2), each with a plus sign. Below these are four smaller widgets for 'Devices' (9), 'Assets' (2), 'Users' (8), and 'Customers' (3). A 'Documentation' widget is present with tabs for 'Getting started', 'Tenant profiles', and 'API', and a 'Widgets Library' section. At the bottom, a 'Configured features' widget shows 'Email', 'SMS', 'Slack', 'OAuth 2', and '2FA' as active features. On the right side, there is a 'CPU' widget showing '15% | 8 cores' and a 'Realtime - last h' line chart with a y-axis from 0% to 100% and x-axis labels '13:40' and '13:'. Below the chart is a 'Transport messages' widget with a 'History - last 30' label. At the bottom right, there are date labels 'May 02' and 'May 05'.

Fill in the blanks as follows:

Providers

Custom  

Login provider* Custom	Allowed platforms Web, Android, iOS
Client ID* e324f9a3f55e1adac4ef	Client secret* 28b3f98c1f55c1cc57f74b9b1a68b5d2e79

General	Mapper
Access token URI* http://localhost:8000/api/login/oauth/ac	Authorization URI* http://localhost:8000/login/oauth/authori
JSON Web Key URI http://localhost:8000/.well-known/jwks	User info URI http://localhost:8000/api/userinfo
Client authentication method* POST	
Provider label* Casdoor	Login button icon

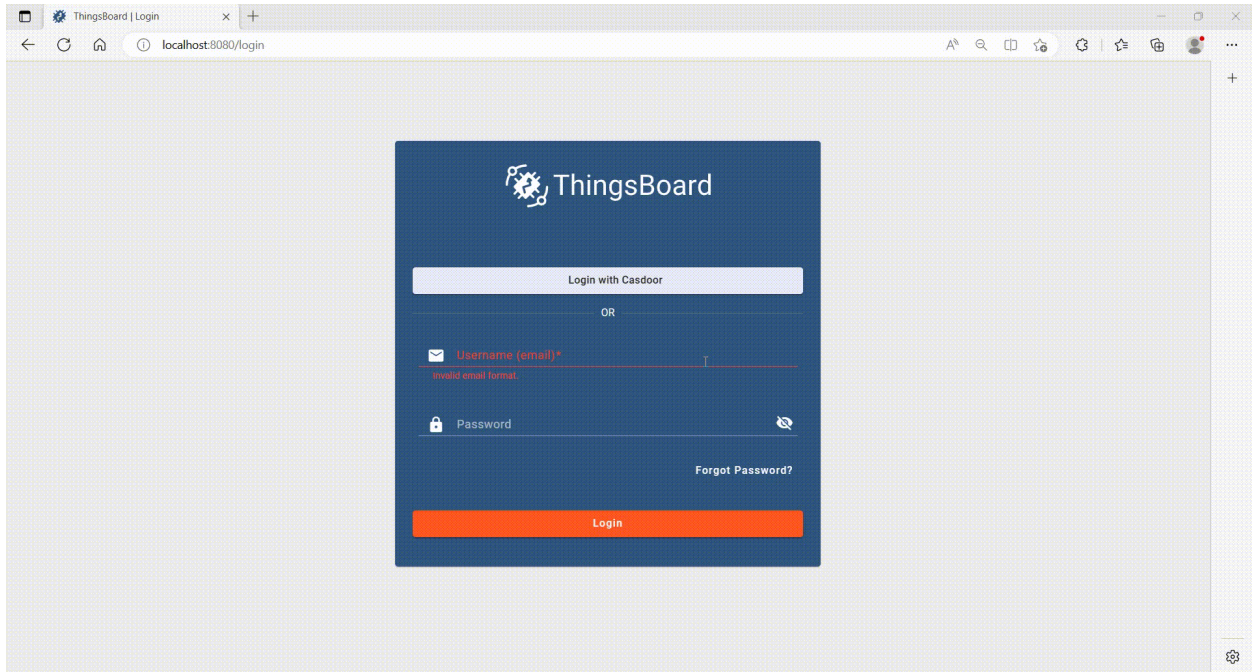
Allow user creation

You can get these values from the following link: [OIDC discovery URL](#)

```
{
  "issuer": "https://door.casdoor.com",
  "authorization_endpoint": "https://door.casdoor.com/login/oauth/authori",
  "token_endpoint": "https://door.casdoor.com/api/login/oauth/access_toke",
  "userinfo_endpoint": "https://door.casdoor.com/api/userinfo",
  "jwks_uri": "https://door.casdoor.com/.well-known/jwks",
  "introspection_endpoint": "https://door.casdoor.com/api/login/oauth/int",
  "response_types_supported": [
```

After filling in these blanks, you have successfully integrated Casdoor with Thingsboard. When you log in to <http://localhost:8080/>, you should see the

following:



JavaScript

Firestore

Firestore project using Casdoor as Identity Provider

WeChat MiniProgram

Using Casdoor in WeChat MiniProgram

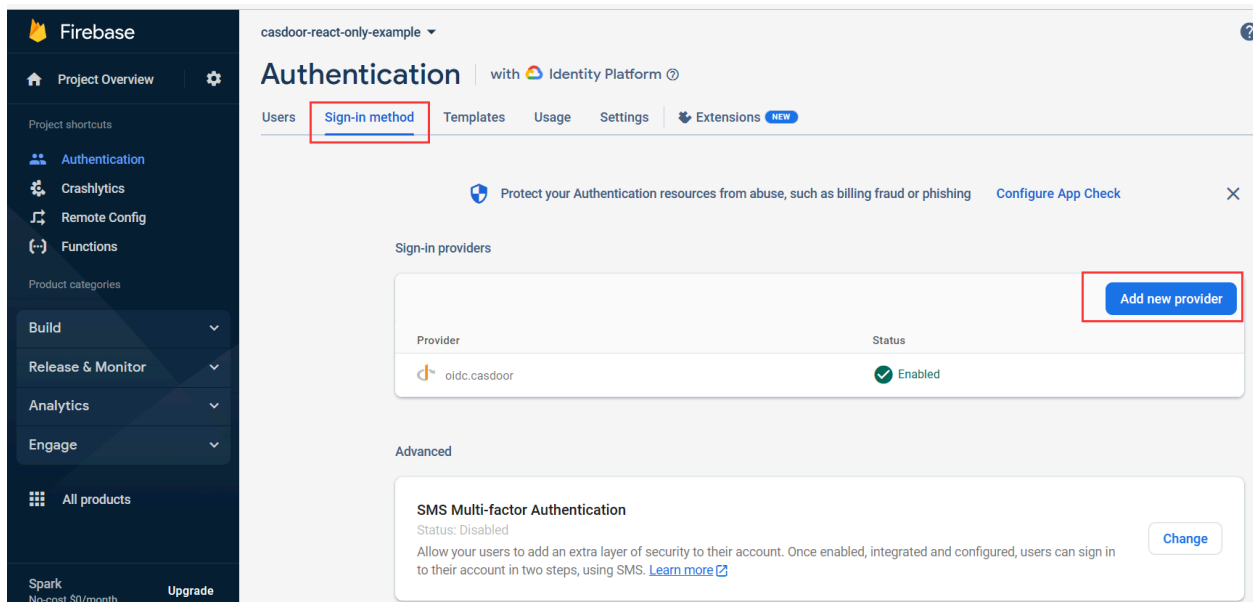
Firebase

Firebase supports OIDC as an Identity Provider, you can use Casdoor as an OIDC provider for Firebase web app.

1. Create a Firebase project

Go to [Firebase Console](#) to create a project.

1.1 Add Casdoor as provider



You need to enable "Identity Platform" feature first to enable OIDC integration on Firebase.

Select `OpenID Connect` in Custom providers, fill in the following information:

Name (in order)	Description	Example value
Name	Any be any string you would like	casdoor
Client ID	Client ID for the Casdoor application	294b09fbc17f95daf2fe
Issuer (URL)	Casdoor server URL	https://door.casdoor.com
Client Secret	Client secret for Casdoor application	dd8982f7046ccba1bbd7851d5c1ece4e52bf039d

door.casdoor.com applications/casbin/app-vue-python-example


Casdoor Home User Management Identity Authorization Logging & Auditing Business & Payments Admin

Edit Application Save Save & Exit

Name: app-vue-python-example

Display name: Casdoor Vue Python Example

Logo: URL: https://cdn.casbin.org/img/casdoor-logo_1185x256.png

Preview: 

Home: https://demo.gsoc.com.cn

Description:

Organization: casbin

Tags:

Client ID: 294b09fbc17f95daf2fe

Client secret: dd8982f7046ccba1bbd7851d5c1ece4e52bf039d

Cert: cert-built-in

The above examples values can be retrieved from Casdoor demo site: [app-vue-python-example](#)


1 Define new OIDC provider

Grant type

Code flow Implicit flow (id_token)

Name

casdoor

Provider ID: oidc.casdoor 

Client ID

294b09fbc17f95daf2fe


Issuer (URL)

https://door.casdoor.com

Client secret

dd8982f7046ccba1bbd7851d5c1ece4e52bf039d

Next

 Configure OIDC integration

1.2 Add callback url

Add Callback URL to Casdoor application Redirect URLs:

OpenID Connect Enable

1 Define new OIDC provider
Name: casdoor, Provider ID: oidc.casdoor, Client ID: 76dfa0e75796f8443b5e, Issuer (URL): h...

2 Configure OIDC integration

When completing set up, add this authorization callback URL to your OIDC app configuration.

Callback URL

https://casdoor-react-only-example.firebaseio.com/_/auth/handler

i To complete set up, follow the steps for your platform

[Apple](#) [Android](#) [Web](#)

Delete provider Cancel

door.casdoor.com/applications/casbin/app-vue-python-example

Tags :

Client ID : 294b09fbc17f95daf2fe

Client secret : dd8982f7046ccb1bbd7851d5c1ece4e52bf039d

Cert : cert-built-in

Redirect URLs :

Redirect URLs

Redirect URL

<http://localhost:>

<http://127.0.0.1:5000/callback>

https://fb-casdoor.firebaseio.com/_/auth/handler

Token format : JWT

Token expire : 168 Hours

Refresh token expire : 0 Hours

Enable password :

Enable signup :

Signin session :

Here we provide an example [casdoor-firebase-example](#) for you to use Casdoor

authentication in your app. To see more details for how to use Casdoor authentication in your app, please refer to [Firebase document](#).

WeChat MiniProgram

! INFO

Casdoor now supports WeChat Mini Program starting from version 1.41.0.

Introduction

Since WeChat Mini Program does not support standardized OAuth, it cannot redirect to the self-hosted Casdoor webpage for login. Therefore, the process of using Casdoor for WeChat Mini Program is different from that of regular programs.

This document will explain how to integrate Casdoor into WeChat Mini Program. You can find an example for this integration on GitHub here: [casdoor-wechat-miniprogram-example](#). For more detailed information, please refer to the WeChat Mini Program [login document](#).

The configuration includes the following names:

`CASD00R_HOSTNAME`: The domain name or IP address where the Casdoor server is deployed, e.g., `https://door.casbin.com`.

Step 1: Deploy Casdoor

Firstly, the [Casdoor server](#) should be deployed.

After successfully deploying Casdoor, you need to ensure:

1. Casdoor can be accessed and used normally.

2. Set Casdoor's `origin` value (conf/app.conf) to `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
```

Step 2: Configure Casdoor Application

1. Create a WeChat IDP in Casdoor and provide the `APPID` and `APPSECRET` given to you by the WeChat Mini Program development platform.

New Provider

Name [?]:

Display name [?]:

Category [?]:

Type [?]:

Client ID [?]:

Client secret [?]:

Provider URL [?]:

2. Create a new Casdoor application or use an existing one.
3. Add the IDP created in the previous step to the application you want to use.

! TIPS

For convenience, Casdoor will treat the first WeChat type IDP in the application as the WeChat Mini Program IDP by default.

Therefore, if you want to use WeChat Mini Program in this app, do not add multiple WeChat type IDPs in one app.

Step 3: Write WeChat MiniProgram Code

WeChat Mini Program provides an API to internally log in and obtain the code. The code should then be sent to Casdoor. Casdoor will use this code to retrieve information (such as OpenID and SessionKey) from the WeChat server.

The following code demonstrates how to accomplish the above process:

```
// Login in mini program
wx.login({
  success: res => {
    // This is the login code that needs to be sent 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
        "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 access token
      }
    })
  }
})
```

It is important to note that the `tag` parameter is mandatory to inform Casdoor that this is a request from the WeChat Mini Program.

The above code includes the username and avatar URI of the WeChat Mini Program user during login. You can choose to pass these two parameters separately and then pass them to Casdoor after a successful login and obtaining the access token:

```
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 URL
      method: "POST",
      data: {
        "owner": "test",
        "name": "wechat-oGk3T5tIiMFo3SazC075f0HEiE7Q",
        "displayName": this.data.userInfo.nickName,
        "avatar": this.data.userInfo.avatarUrl
      },
      header: {
        "Authorization": "Bearer " + app.globalData.accessToken,
        // Bearer token
        "content-type": "application/json"
      },
      success: (res) => {
        console.log(res)
      }
    })
  }
})
```


Additionally, you can use the access token as a bearer token for any Casdoor operation you require.

 TIPS

Currently, Casdoor is unable to bind existing accounts to WeChat Mini Program users. After Casdoor retrieves the OpenID from WeChat, it will either create a new user if the ID does not exist, or use the existing user if it does.

Lua

 APISIX

Using Casdoor in APISIX

APISIX

Currently, there are 2 methods to use Casdoor to connect to APISIX via APISIX plugins and protect the APIs behind APISIX: using APISIX's Casdoor plugin or using APISIX's OIDC plugin.

Connect Casdoor via APISIX's Casdoor plugin

This plugin, `authz-casdoor`, can protect APIs behind APISIX, forcing every single request to get authenticated without modifying the code of the API.

How to enable it

You need to specify this plugin when creating the route and provide all the required fields. Here is an example.

```
curl "http://127.0.0.1:9180/apisix/admin/routes/1" -H "X-API-KEY:
edd1c9f034335f136f87ad84b625c8f1" -X PUT -d '
{
  "methods": ["GET"],
  "uri": "/anything/*",
  "plugins": {
    "authz-casdoor": {
      "endpoint_addr": "http://localhost:8000",
      "callback_url": "http://localhost:9080/anything/callback",
      "client_id": "7ceb9b7fda4a9061ec1c",
      "client_secret": "3416238e1edf915eac08b8fe345b2b95cdba7e04"
    }
  },
  "upstream": {
```

In this example, we created a route `"/anything/*"` pointed to `"httpbin.org:80"` using APISIX's admin API, with the `"authz-casdoor"` plugin enabled. This route is now under the authentication protection of Casdoor.

Attributes

Name	Type	Requirement	Default	Valid	Description
<code>endpoint_addr</code>	string	required			The URL of Casdoor.
<code>client_id</code>	string	required			The client ID in Casdoor.
<code>client_secret</code>	string	required			The client secret in Casdoor.
<code>callback_url</code>	string	required			The callback URL which is used to receive state and code.

endpoint_addr and callback_url should not end with '/'

In the configuration of the `"authz-casdoor"` plugin, we can see four parameters.

The first one is `"callback_url"`. This is the callback URL in OAuth2. It should be emphasized that this callback URL **must belong to the "uri" you specified for the route**. For example, in this example, <http://localhost:9080/anything/callback> obviously belongs to `"/anything/*"`. Only by this way, the visit toward the `callback_url` can be intercepted and utilized by the plugin (so that the plugin can

get the code and state in OAuth2). The logic of the `callback_url` is implemented completely by the plugin, so there is no need to modify the server to implement this callback.

The second parameter `"endpoint_addr"` is obviously the URL of Casdoor. The third and fourth parameters are `"client_id"` and `"client_secret"`, which you can acquire from Casdoor when you register an app.

How it works?

Suppose a new user who has never visited this route before is going to visit it (<http://localhost:9080/anything/d?param1=foo¶m2=bar>). Considering that `"authz-casdoor"` is enabled, this visit would be processed by the `"authz-casdoor"` plugin first. After checking the session and confirming that this user hasn't been authenticated, the visit will be intercepted. With the original URL the user wants to visit kept, they will be redirected to the login page of Casdoor.

After successfully logging in with a username and password (or whatever method they use), Casdoor will redirect this user to the `"callback_url"` with GET parameters `"code"` and `"state"` specified. Because the `"callback_url"` is known by the plugin, when the visit toward the `"callback_url"` is intercepted this time, the logic of the `"Authorization code Grant Flow"` in OAuth2 will be triggered. This means that the plugin will request the access token to confirm whether this user is really logged in. After this confirmation, the plugin will redirect this user to the original URL they want to visit, which was kept by us previously. The logged-in status will also be kept in the session.

Next time this user wants to visit the URL behind this route (for example, <http://localhost:9080/anything/d>), after discovering that this user has been authenticated previously, this plugin won't redirect this user anymore. This way, the user can visit whatever they want under this route without being interfered.

Connect Casdoor via APISIX's OIDC plugin

Casdoor can use the OIDC protocol to connect to APISIX, and this document will show you how to do it.

The following are some of the names used in the configuration:

`CASD00R_HOSTNAME`: Domain name or IP where the Casdoor server is deployed.

`APISIX_HOSTNAME`: Domain name or IP where APISIX is deployed.

Step 1: Deploy Casdoor and APISIX

Firstly, deploy [Casdoor](#) and [APISIX](#).

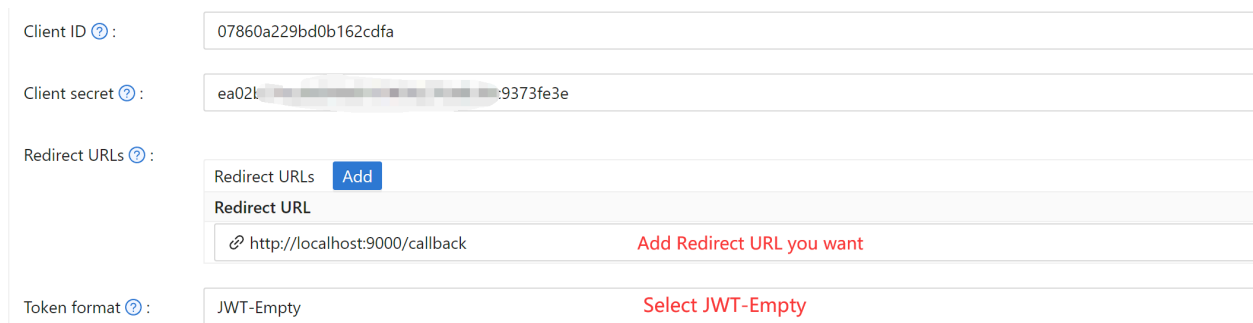
After a successful deployment, you need to ensure:

1. Casdoor can be logged in and used normally.
2. Set Casdoor's `origin` value (conf/app.conf) to `CASD00R_HOSTNAME`.

```
conf > ⚙ app.conf
8   dbName = casdoor
9   redisEndpoint =
10  defaultStorageProvider =
11  isCloudIntranet = false
12  authState = "casdoor"
13  httpProxy = "127.0.0.1:10808"
14  verificationCodeTimeout = 10
15  initScore = 2000
16  logPostOnly = true
17  origin = "http://10.144.1.2:8000"
      CASD00R_HOSTNAME
```

Step 2: Configure Casdoor application

1. Create a new Casdoor application or use an existing one.
2. Add a redirect URL: `http://APISIX_HOSTNAME/REDIRECTWHATYOUWANT`, and replace `REDIRECTWHATYOUWANT` with the desired redirect URL.
3. Select "JWT-Empty" for the Token format option.
4. Add the desired provider and configure other settings.



The screenshot shows the Casdoor application settings form. It includes fields for Client ID (07860a229bd0b162cdfa), Client secret (ea02l...9373fe3e), Redirect URLs (with an 'Add' button and a list containing 'http://localhost:9000/callback'), and Token format (JWT-Empty). Red text prompts are visible: 'Add Redirect URL you want' and 'Select JWT-Empty'.

On the application settings page, you will find the `Client ID` and `Client Secret` values as shown in the picture above. We will use them in the next step.

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

Step 3: Configure APISIX

APISIX has official [OIDC](#) support, which is implemented using [lua-resty-openidc](#).

You can customize the settings according to the APISIX OIDC documentation. The following routing settings will be used:

```
# Use your own X-API-Key
$ curl -X POST APISIX_HOSTNAME/apisix/admin/routes -H "X-API-Key:
edd1c9f034335f136f87ad84b625c8f1" -d '{
```


PHP

Zentao

Using Casdoor for authentication in Zentao

Using Casdoor as an OAuth2 Server in ShowDoc

Using Casdoor as an OAuth2 server in ShowDoc

Flarum

Using OAuth2 to connect various applications, like Flarum

Moodle

Using OAuth to connect Moodle

Zentao

[Zentao](#) is an agile (scrum) project management system/tool, but it does not support OIDC itself. To integrate Zentao with Casdoor SSO, we need to use a 3rd-party OIDC module called [zentao-oidc](#), and this document will show you how to do it.

Step 1: Deploy Casdoor and Zentao

Firstly, deploy [Casdoor](#) and [Zentao](#). After a successful deployment, make sure:

1. Casdoor can be logged in and used successfully.
2. You can successfully log in and use Zentao.

Step 2: Integrate Zentao OIDC third-party module

Install [zentao-oidc](#) by running the following command:

```
git clone https://github.com/casdoor/zentao-oidc.git
```

Alternatively, you can download the ZIP and unzip it.

This module is used to integrate Zentao with SSO for OpenId. Here's how to use it:

1. Copy the entire `oidc` directory to the module of Zentao and use it as a module of Zentao. Rename the downloaded package to "oidc".

2. Configure the filter.

Since the Zentao framework filters the parameters in the URL and does not allow spaces, you need to put the following code at the end of `/config/my.php`.

```
$filter->oidc = new stdClass();
$filter->oidc->index = new stdClass();
$filter->oidc->index->paramValue['scope'] = 'reg::any';
```

3. Modify `/module/commom/model.php`.

Add 'oidc' to the anonymous access list and add a line to the `isOpenMethod` method of `model.php`.

```
public function isOpenMethod($module, $method)
{
    if ($module == 'oidc' and $method == 'index') {
        return true;
    }
}
```

4. If you don't want the Zentao login screen to appear, go directly to the Casdoor login screen.

Modify the last line of code in `public function checkPriv()` in `/module/common/model.php`.

```
//return print(js::locate(helper::createLink('user', 'login',
"referer=$referer")));
return print(js::locate(helper::createLink('oidc', 'index',
```

5. Modify the `setSuperVars()` method inside `framework/base/router.class.php` and comment out the following statements.

```
public function setSuperVars()  
// unset($_REQUEST);
```

Step 3: Configure Casdoor Application

1. Create a new Casdoor application or use an existing one.
2. Add your redirect URL.

Client ID ? :	<input type="text" value="d8d7715e24f077066a20"/>				
Client secret ? :	<input type="password" value="REDACTED"/>				
Cert ? :	<input type="text" value="cert-built-in"/>				
Redirect URLs ? :	<table><tr><td>Redirect URLs</td><td>Add</td></tr><tr><td>Redirect URL</td><td><input type="text" value="http://127.0.0.1/zentao/oidc-index.html"/></td></tr></table>	Redirect URLs	Add	Redirect URL	<input type="text" value="http://127.0.0.1/zentao/oidc-index.html"/>
Redirect URLs	Add				
Redirect URL	<input type="text" value="http://127.0.0.1/zentao/oidc-index.html"/>				

3. Add the provider you want and fill in other required settings.

Step 4: Configure Zentao

Configure the `config.php` file in the `oidc` directory.

```
$config->oidc->clientId = "<Your ClientId>";
```

Set your redirect URL in `module/oidc` in the `public function index()` method.

```
$oidc->setRedirectURL($path."/zentao/oidc-index.html");
```

i NOTE

The URL here refers to calling the 'index' method in the 'oidc' module. You also need to set a variable separator. By default, the framework uses a dash ("-"). Please refer to the official Zentao framework for more details.

["zentaoPHP???"](#)

Using Casdoor as an OAuth2 Server in ShowDoc

Using Casdoor for Authentication in ShowDoc

[ShowDoc](#) is an online API documentation and technical documentation tool that is perfect for IT teams. ShowDoc makes it easy to use Markdown syntax to write beautiful API documents, data dictionary documents, technical documents, online Excel documents, and more.

ShowDoc supports 3rd-party authentication, including OAuth2. Here is a tutorial for achieving this.

Step 1: Create a Casdoor Application

Go to your Casdoor and add a new application called ShowDoc. Here is an example of creating the ShowDoc application in Casdoor.

Edit Application

Save

Save & Exit

Name ? : myApplication

Display name ? : myApplication

Logo ? :

URL ? : https://cdn.casdoor.com/logo/casdoor-logo_1185x256.png

Preview:



Home ? : [↗](#)

Description ? :

Organization ? : built-in

Client ID ? : 208d745196c23df9fd5b

Client secret ? : 4c89f447af77bc276431ab885463ebcb8d6efc3c

Cert ? : cert-built-in

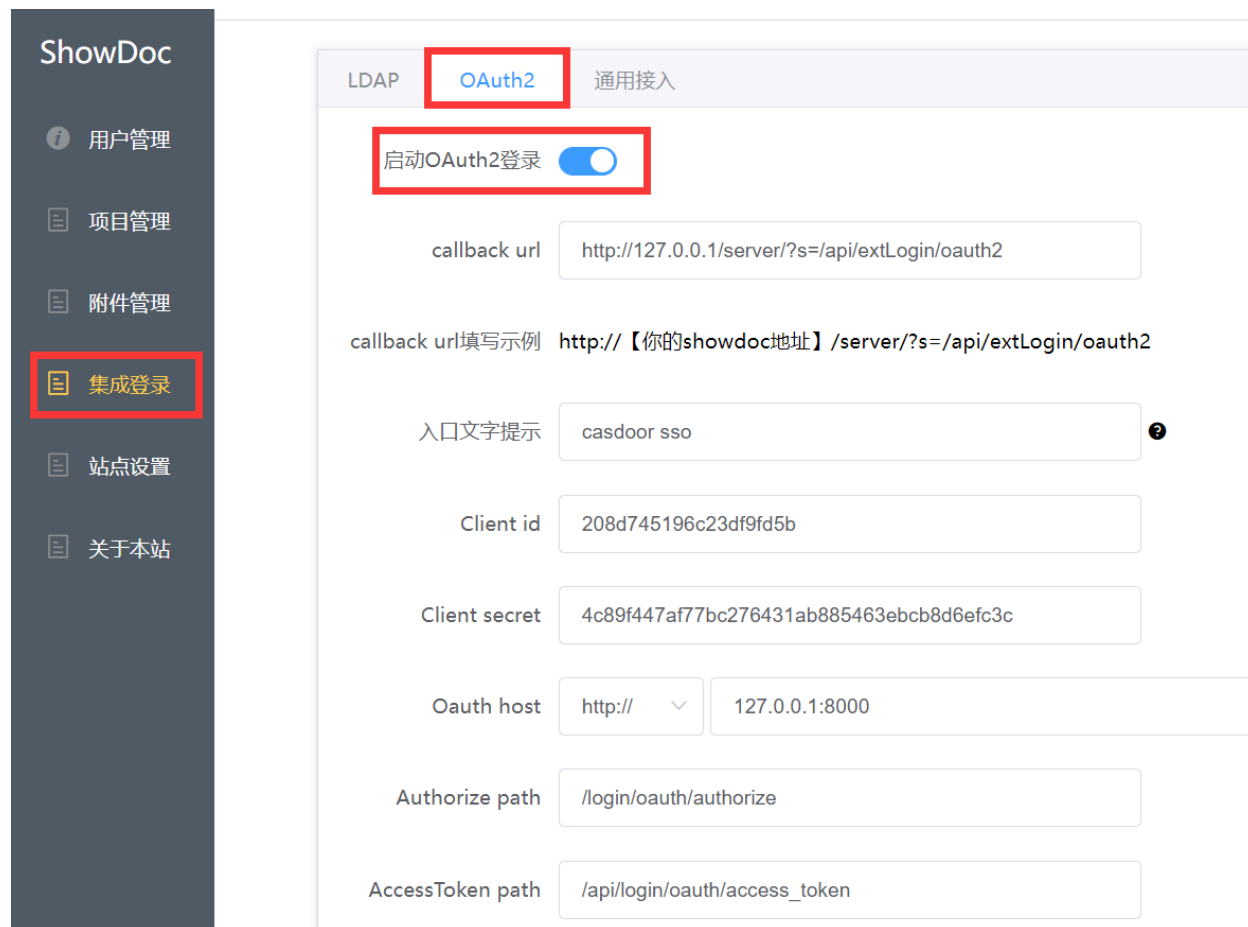
Please remember the `client ID` and `client Secret` for the next step.

! INFO

Please don't fill in the **callback URL** in this step. The URL depends on the configurations on ShowDoc in the next step. We will come back to set a correct callback URL later.

Step 2: Configure ShowDoc

First, enable the OAuth2 login button. Then, fill in the `callback URL` as shown in the example. Fill in the `client ID` and `client secret` that were remembered in the previous step.



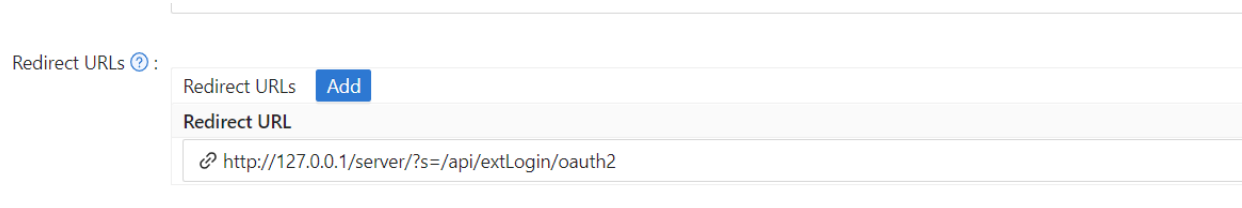
The screenshot shows the ShowDoc configuration interface. On the left is a dark sidebar with the following menu items: 用户管理, 项目管理, 附件管理, 集成登录 (highlighted with a red box), 站点设置, and 关于本站. The main content area has three tabs: LDAP, OAuth2 (selected and highlighted with a red box), and 通用接入. Under the OAuth2 tab, there is a toggle switch for '启动OAuth2登录' which is turned on and highlighted with a red box. Below this are several input fields: 'callback url' with the value 'http://127.0.0.1/server/?s=/api/extLogin/oauth2'; a text example 'callback url填写示例 http://【你的showdoc地址】/server/?s=/api/extLogin/oauth2'; '入口文字提示' with the value 'casdoor sso'; 'Client id' with the value '208d745196c23df9fd5b'; 'Client secret' with the value '4c89f447af77bc276431ab885463ebcb8d6efc3c'; 'Oauth host' with a dropdown set to 'http://' and the text '127.0.0.1:8000'; 'Authorize path' with the value '/login/oauth/authorize'; and 'AccessToken path' with the value '/api/login/oauth/access_token'.

`Authorize path`, `AccessToken path`, and `User info path` are required. You can fill them in as shown below.

```
Authorize path:    /login/oauth/authorize
AccessToken path: /api/login/oauth/access_token
User info path:   /api/get-account
```

Step 3: Configure the Callback URL in Casdoor

Go back to the application edit page in step 1 and add the `callback URL` that you filled in ShowDoc.



The screenshot shows a section titled "Redirect URLs" with a help icon. Below the title is a table with one row. The table has two columns: "Redirect URL" and "Add". The "Redirect URL" column contains the text "http://127.0.0.1/server/?s=/api/extLogin/oauth2".

Redirect URL	Add
http://127.0.0.1/server/?s=/api/extLogin/oauth2	

Step 4: Have a Try on ShowDoc

You should see the following on the login page:

登录



登录

注册新账号

[casdoor sso](#)

Congratulations! You have completed all the steps. Press the 'Casdoor SSO' button, and you will be redirected to the Casdoor login page.

Flarum

[Casdoor](#) can use OAuth2 to connect various applications. In this example, we will show you how to use OAuth2 to connect Flarum to your applications.

Here are some configuration names you will need:

`CASDOOR_HOSTNAME`: The domain name or IP where the Casdoor server is deployed.

`Flarum_HOSTNAME`: The domain name or IP where Flarum is deployed.

Step 1: Deploy Casdoor and Flarum

First, deploy [Casdoor](#) and [Flarum](#).

After a successful deployment, make sure:

1. You have downloaded the Flarum plugin [FoF Passport](#).
2. Casdoor can be logged in and used normally.
3. You can set `CASDOOR_HOSTNAME = http://localhost:8000` when deploying Casdoor in `prod` mode. See [production mode](#).

Step 2: Configure Casdoor application

1. Create a new Casdoor application or use an existing one.
2. Find the redirect URL: `<CASDOOR_HOSTNAME>/auth/passport`.
3. Add the redirect URL to the Casdoor application:

The screenshot shows a configuration form with the following fields:


- Client ID:** 014ae4bd048734ca2dea
- Client secret:** f26a4115725867b7bb7b668c81e18f77ae1544d
- Cert:** cert-built-in
- Redirect URLs:** A table with one entry: `<your flarum install>/auth/passport`. The table has columns for "Redirect URL" and "Action".

On the application settings page, you will find two values: **Client ID** and **Client secret**. We will use these values in the next step.

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

Step 3: Configure Flarum

1. Install the plugin [FoF Passport](#).
2. Configure the app:

**FoF Passport**

The OAuth2 (and Laravel passport) compatible oauth extension

Enabled

OAuth authorization URL

`https://door.casdoor.com/login/oauth/authorize`

OAuth token URL

`https://door.casdoor.com/api/login/oauth/access_token`

Api URL providing user details when authenticated

`https://door.casdoor.com/api/user`

OAuth application ID

`014ae4bd048734ca2dea`

OAuth application secret

`f26a4115725867b7bb7b668c81e1f8f7fae1544d`

OAuth scopes to request

`openid profile email`

Label for login button

`Casdoor SSO`

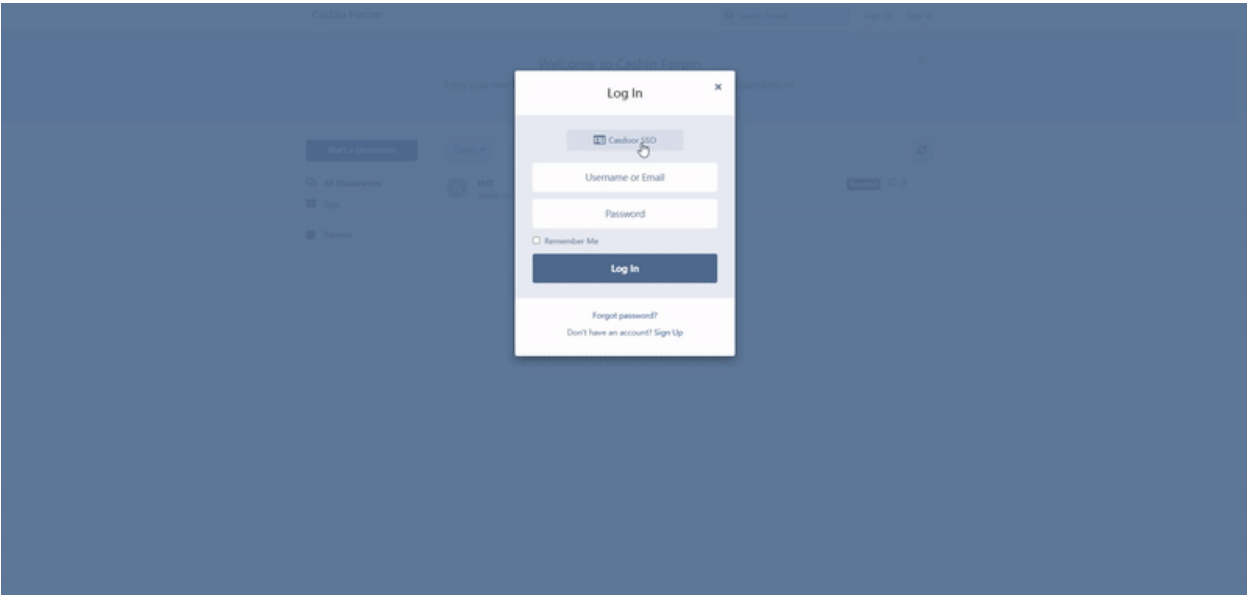
Icon for login button

`far fa-id-card`

3. Find the Client ID and Client Secret in the 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.



Moodle

[Casdoor](#) can be used to connect [Moodle](#) using OAuth.

The following are some configuration settings:

- `CASD00R_HOSTNAME`: The domain name or IP where the Casdoor server is deployed.
- `Mood1e_H0STNAME`: The domain name or IP where Moodle is deployed.

Step 1: Deploy Casdoor and Moodle

First, deploy [Casdoor](#) and [Moodle](#).

After successful deployment, ensure the following:

1. Casdoor can be logged in and used without issues.
2. You can set `CASD00R_HOSTNAME` as `http://localhost:8000` when deploying Casdoor in `prod` mode. See [production mode](#).

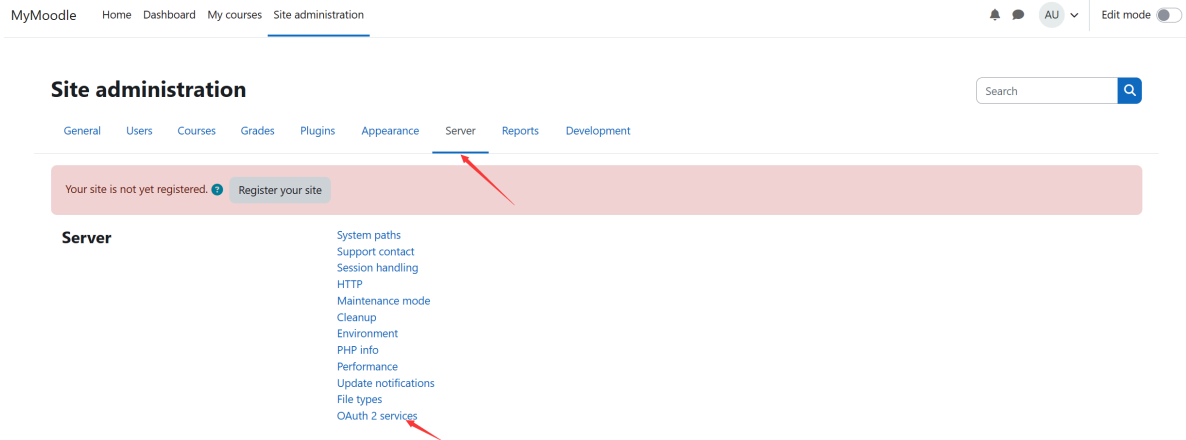
Step 2: Configure Casdoor Application

1. Create a new Casdoor application or use an existing one.
2. Find the redirect URL: `Modd1e_H0STNAME/admin/oauth2callback.php`.
3. Add the redirect URL to the Casdoor application.

For more information on OAuth, refer to [OAuth](#).

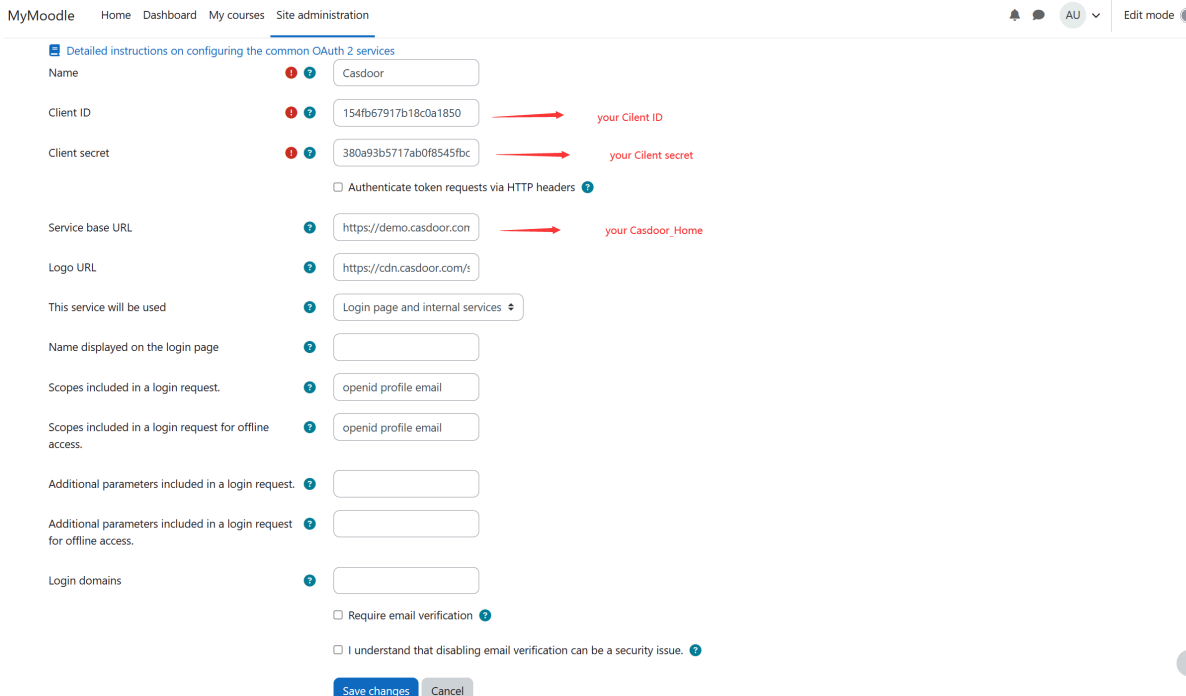
Step 3: Configure Moodle

1. Locate OAuth



The screenshot shows the Moodle Site Administration interface. The top navigation bar includes 'MyMoodle', 'Home', 'Dashboard', 'My courses', and 'Site administration'. The 'Server' menu item is highlighted with a red arrow. Below the navigation bar, a message states 'Your site is not yet registered. Register your site'. The 'Server' section is expanded, showing a list of sub-items: System paths, Support contact, Session handling, HTTP, Maintenance mode, Cleanup, Environment, PHP info, Performance, Update notifications, File types, and OAuth 2 services. A red arrow points to 'OAuth 2 services'.

2. Configure this application



The screenshot shows the Moodle OAuth 2 services configuration page. The top navigation bar is the same as in the previous screenshot. The page title is 'Detailed instructions on configuring the common OAuth 2 services'. The configuration form includes the following fields and options:

- Name: Casdoor
- Client ID: 154fb67917b18c0a1850 (with a red arrow pointing to the text 'your Client ID')
- Client secret: 380a93b5717ab0f8545fbc (with a red arrow pointing to the text 'your Client secret')
- Authenticate token requests via HTTP headers
- Service base URL: https://demo.casdoor.com (with a red arrow pointing to the text 'your Casdoor_Home')
- Logo URL: https://cdn.casdoor.com/!
- This service will be used: Login page and internal services
- Name displayed on the login page: (empty field)
- Scopes included in a login request: openid profile email
- Scopes included in a login request for offline access: openid profile email
- Additional parameters included in a login request: (empty field)
- Additional parameters included in a login request for offline access: (empty field)
- Login domains: (empty field)
- Require email verification
- I understand that disabling email verification can be a security issue.

At the bottom of the form are 'Save changes' and 'Cancel' buttons.

3. Configure this mapping

User field mappings for issuer: Casdoor

External field name	Internal field name	Edit
address	address	⚙️ 🗑️
email	email	⚙️ 🗑️
name	firstname	⚙️ 🗑️
phone	phone1	⚙️ 🗑️
picture	picture	⚙️ 🗑️
perferred_username	username	⚙️ 🗑️

Create new user field mapping for issuer "Casdoor"

4. Locate the OAuth2 plugin

General Users Courses Grades **Plugins** Appearance Server Reports Development

Your site is not yet registered. [Register your site](#)

Plugins

Install plugins
Plugins overview

Activity modules

- Manage activities
- Common activity settings
- Assignment
 - Assignment settings
 - Submission plugins
 - Manage assignment submission plugins
 - File submissions
 - Online text submissions
- Feedback plugins
 - Manage assignment feedback plugins
 - Feedback comments
 - Annotate PDF
 - File feedback
 - Offline grading worksheet
- Book
- Chat
- Database
- External tool
 - Manage tools
- Feedback
- File
- Folder
- Forum
- Glossary
- HSP
- IMS content package
- Lesson
- Page
- Quiz
 - General settings
 - Safe Exam Browser templates
 - Safe Exam Browser access rules
- SCORM package
- Text and media area
- URL
- Workshop

Admin tools

- Manage admin tools
- Accessibility
 - Brickfield registration
 - Accessibility toolkit settings
- Reports
- Recycle bin

Antivirus plugins

Manage antivirus plugins



Authentication

- Manage authentication
- Email-based self-registration
- Manual accounts
- OAuth 2

5. Enable the OAuth2 plugin

Manage authentication

Available authentication plugins

Name	Users	Enable	Up/Down	Settings	Test settings	Uninstall
Manual accounts	2			Settings		
No login	0					
Email-based self-registration	0		↓	Settings		Uninstall
OAuth 2	8		↑	Settings	Test settings	

6. If you want to prevent the editing of Casdoor's email

Lock user fields

You can lock user data fields. This is useful for sites where the user data is maintained by the administrators manually by editing user records or uploading using the 'Upload users' facility. If you are locking fields that are required by Moodle, make sure that you provide that data when creating user accounts or the accounts will be unusable.

Consider setting the lock mode to 'Unlocked if empty' to avoid this problem.

Lock value (First name) <small>auth_oauth2 field_lock_firstname</small>	Unlocked	Default: Unlocked
Lock value (Last name) <small>auth_oauth2 field_lock_lastname</small>	Unlocked	Default: Unlocked
Lock value (Email address) <small>auth_oauth2 field_lock_email</small>	Locked	Default: Unlocked
Lock value (City/town) <small>auth_oauth2 field_lock_city</small>	Unlocked	Default: Unlocked
Lock value (Country) <small>auth_oauth2 field_lock_country</small>	Unlocked	Default: Unlocked
Lock value (Language) <small>auth_oauth2 field_lock_lang</small>	Unlocked	Default: Unlocked

here is switch to lock email

For more information on Moodle, refer to [Moodle](#) and [Fields mapping](#).

Log out of Moodle and test SSO.

MyMoodle Website

Ruby

GitLab

Using Casdoor for authentication in a self-developed GitLab server

GitLab

Casdoor can use the OIDC protocol to link to a self-deployed GitLab server, and this document will show you how to do it.

CAUTION

As the [GitLab docs](#) state, GitLab only works with OpenID providers that use HTTPS, so you need to deploy Casdoor with HTTPS, such as putting Casdoor behind an NGINX reverse proxy with an SSL certificate setup. Casdoor itself only listens on port 8000 by default via HTTP and has no HTTPS-related functionality.

The following are some of the names mentioned in the configuration:

`CASDOOR_HOSTNAME`: The domain name or IP where the Casdoor server is deployed, e.g., `https://door.casbin.com`.

`GITLAB_HOSTNAME`: The domain name or IP where GitLab is deployed, e.g., `https://gitlab.com`.

Step 1: Deploy Casdoor and GitLab

Firstly, Casdoor and GitLab should be deployed.

After a successful deployment, you need to ensure:

1. Casdoor can be logged into and used normally.
2. Set Casdoor's `origin` value (conf/app.conf) to `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
```

Step 2: Configure Casdoor application

1. Create or use an existing Casdoor application.
2. Add a redirect URL: `http://GITLAB_HOSTNAME/users/auth/openid_connect/callback`.
3. Add the provider you want and supplement other settings.

Description ⓘ :

Organization ⓘ :

Client ID ⓘ : Client ID

Client secret ⓘ : Client secret

Redirect URLs ⓘ :

Redirect URLs

Redirect URL	
http://GITLAB_HOSTNAME/users/auth/openid_connect/callback	GitLab redirect url

Notably, you can get two values on the application settings page: `Client ID` and `Client secret` (see 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`, where you will see the OIDC configuration of Casdoor.

Step 3: Configure GitLab

You can follow the steps below to set this up, or make custom changes according to [this document](#) (e.g., if you are installing GitLab using source code rather than the Omnibus).

1. On your GitLab server, open the configuration file.

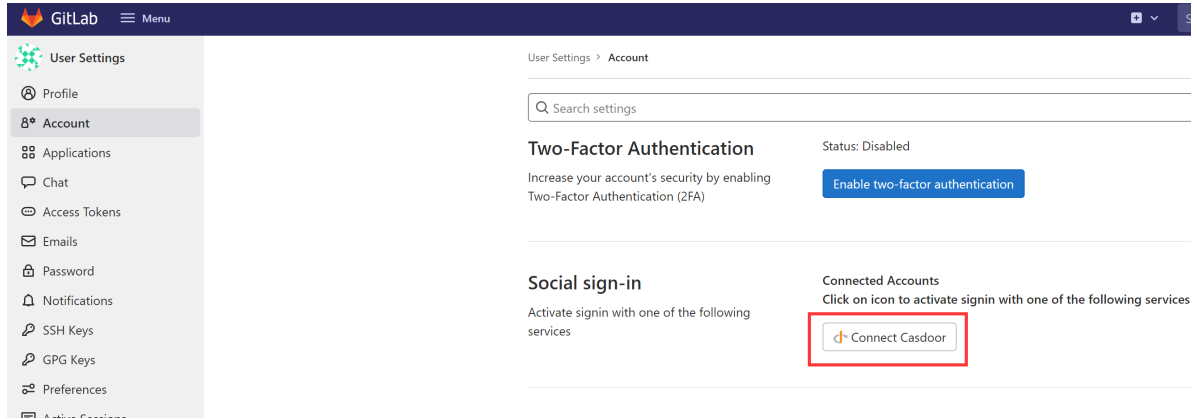
```
sudo editor /etc/gitlab/gitlab.rb
```

2. Add the provider configuration. (The HOSTNAME URL should include http or https)

```
gitlab_rails['omniauth_providers'] = [
  {
    name: "openid_connect",
    label: "Casdoor", # optional label for the login
    button, defaults to "Openid Connect"
    args: {
      name: "openid_connect",
      scope: ["openid", "profile", "email"],
      response_type: "code",
      issuer: "<CASDOOR_HOSTNAME>",
      client_auth_method: "query",
      discovery: true,
      uid_field: "preferred_username",
      client_options: {
        identifier: "<YOUR CLIENT ID>",
        secret: "<YOUR CLIENT SECRET>",
        redirect_uri: "<GITLAB_HOSTNAME>/users/auth/"
```


3. Reboot your GitLab server.

4. Each registered user can open `GITLAB_HOSTNAME` `/-/profile/account` and connect the Casdoor account.



5. Finish. Now, you can log in to your own GitLab using Casdoor.



GitLab

A complete DevOps platform

GitLab is a single application for the entire software development lifecycle. From project planning and source code management to CI/CD, monitoring, and security.

This is a self-managed instance of GitLab.

Username or email

Password

Remember me [Forgot your password?](#)

Sign in

Don't have an account yet? [Register now](#)

Sign in with

Remember me

Haskell

Hasura

Before the integration, we need to deploy Casdoor locally.

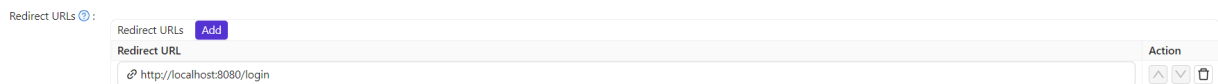
Hasura

Before the integration, we need to deploy Casdoor locally.

Then we can quickly implement a Casdoor-based login page in our own app with the following steps.

Configure Casdoor application

1. Create or use an existing Casdoor application.
2. Add a redirect URL: `http://CASD00R_H0STNAME/login`

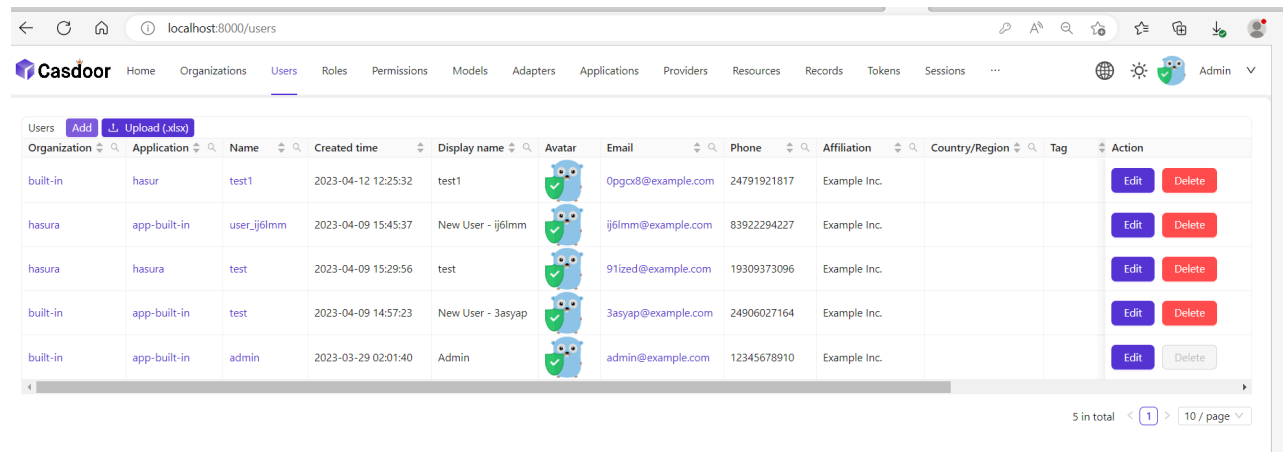


3. Copy the client ID; we will need it in the following steps.

Add a user in Casdoor

Now that you have the application, but not a user. That means you need to create a user and assign the role.

Go to the "Users" page and click on "Add user" in the top right corner. That opens a new page where you can add the new user.



Save the user after adding a username and adding the organization Hasura (other details are optional).

Now you need to set up a password for your user, which you can do by clicking "manage your password."

Choose a password for your user and confirm it.

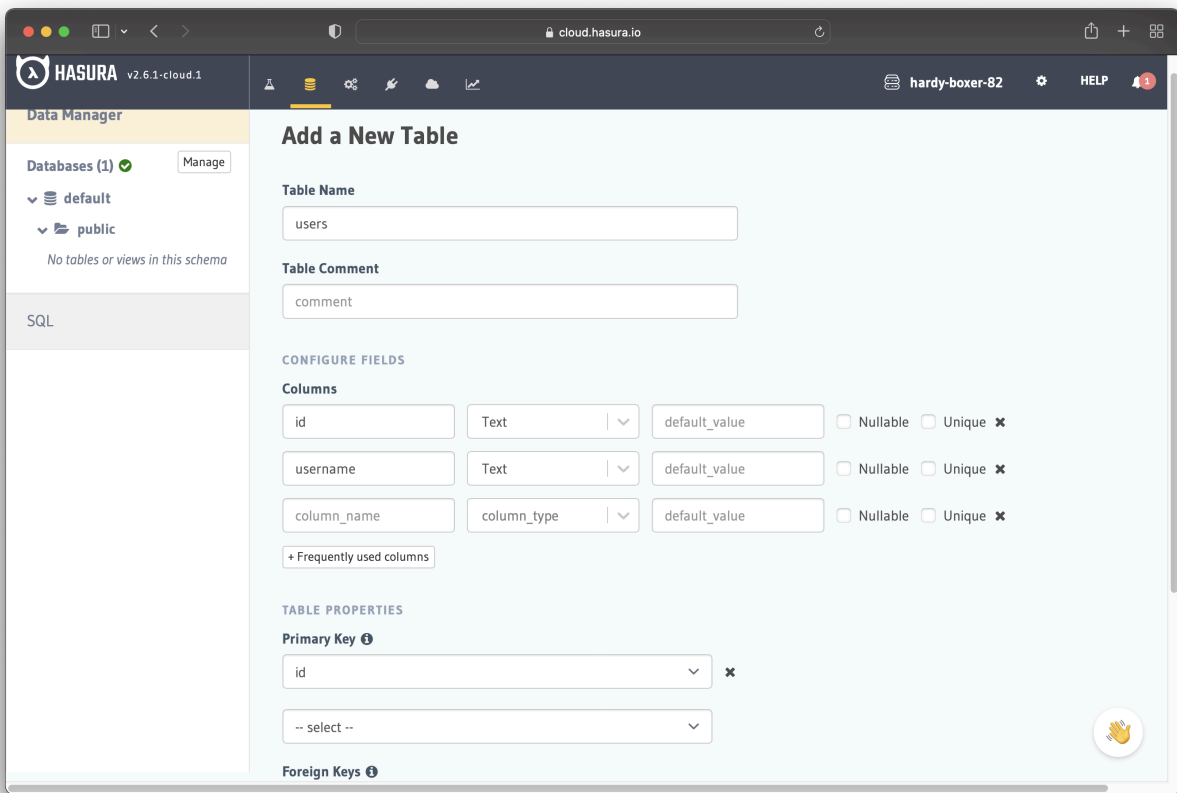
Build the Hasura App

Start the Hasura by Docker or Hasura Cloud.

Now create a `users` table with the following columns:

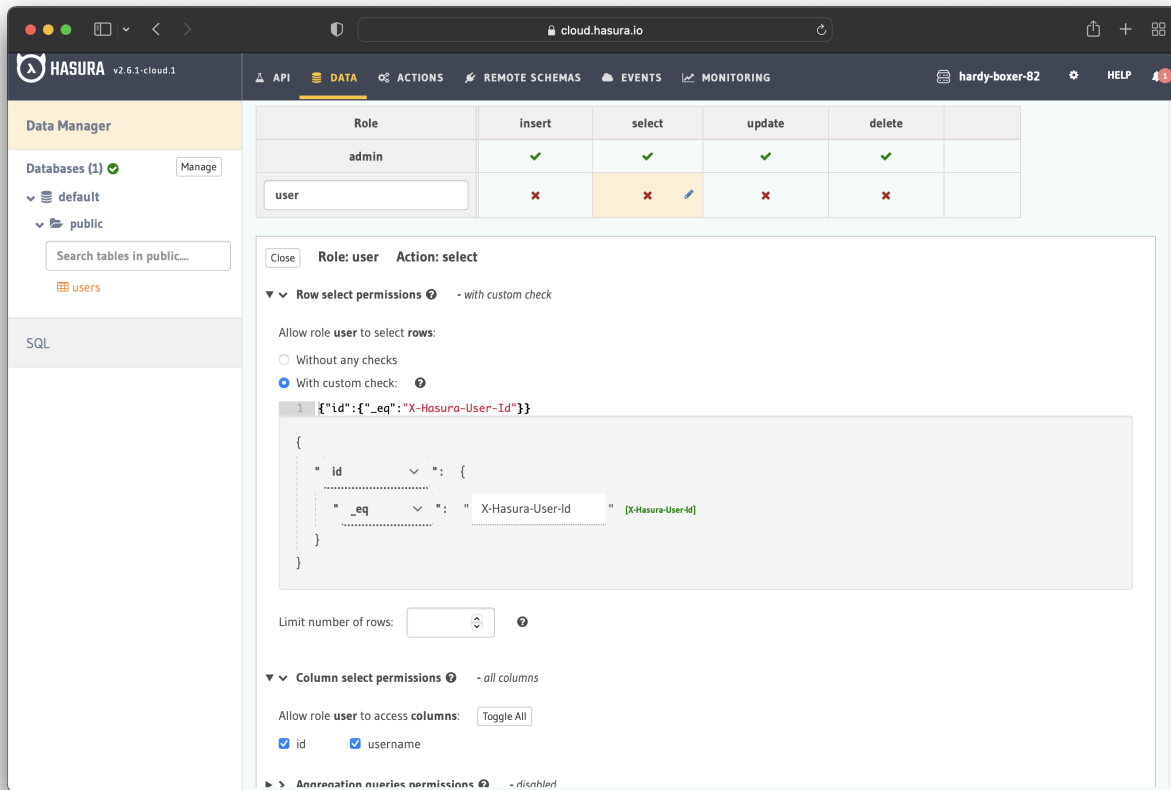
- `id` of type Text (Primary Key)
- `username` of type Text

Refer to the image below for reference.



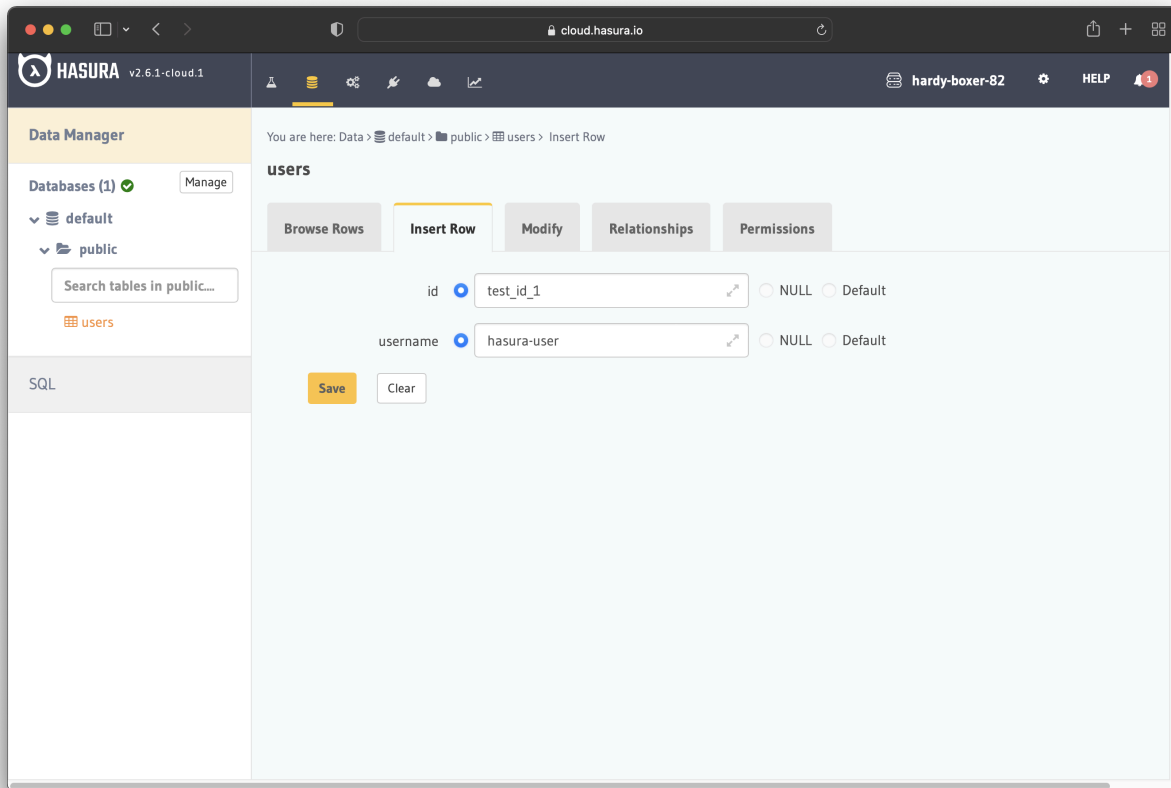
The next step is to create a `user` role for the app. Users should be able to see only their records but not other people's records.

Configure the `user` role as shown in the image below. For more information, read about [configuring permission rules in Hasura](#).



This way, users cannot read other people's records. They can only access theirs.

For testing purposes, add a dummy user. This is to ensure that when you use the JWT token, you only see your user's details and not other users' details.



Now you need to set the `JWT_SECRET` in Hasura.

Configure Hasura with Casdoor

In this step, you need to add the `HASURA_GRAPHQL_JWT_SECRET` to Hasura.

To do so, go to the Hasura docker-compose.yml and then add the new `HASURA_GRAPHQL_JWT_SECRET` as below.

The `HASURA_GRAPHQL_JWT_SECRET` should be in the following format. Remember to change `<Casdoor endpoint>` to your own Casdoor's URL (like `https://door.casdoor.com`)

```
HASURA_GRAPHQL_JWT_SECRET: '{"claims_map": {
  "x-hasura-allowed-roles": {"path": "$.roles"},
  "x-hasura-default-role": {"path": "$.roles[0]"},
  "x-hasura-user-id": {"path": "$.id"}
}, "jwk_url": "<Casdoor endpoint>/well-known/jwks"}
```

Save the change and reload the docker.

```
## enable debugging mode. It is recommended to disable this in production
HASURA_GRAPHQL_DEV_MODE: "true"
HASURA_GRAPHQL_ENABLED_LOG_TYPES: startup, http-log, webhook-log, websocket-log, query-log
HASURA_GRAPHQL_ADMIN_SECRET: myadminsecretkey
HASURA_GRAPHQL_JWT_SECRET: '{"claims_map": {
  "x-hasura-allowed-roles": ["user", "editor"],
  "x-hasura-default-role": "user",
  "x-hasura-user-id": "4ec7ccee-ec7b-4191-a78d-e11f50686f8b"
}, "jwk_url": "https://door.casdoor.com/.well-known/jwks"}'
```

Retrieve the JWT Token

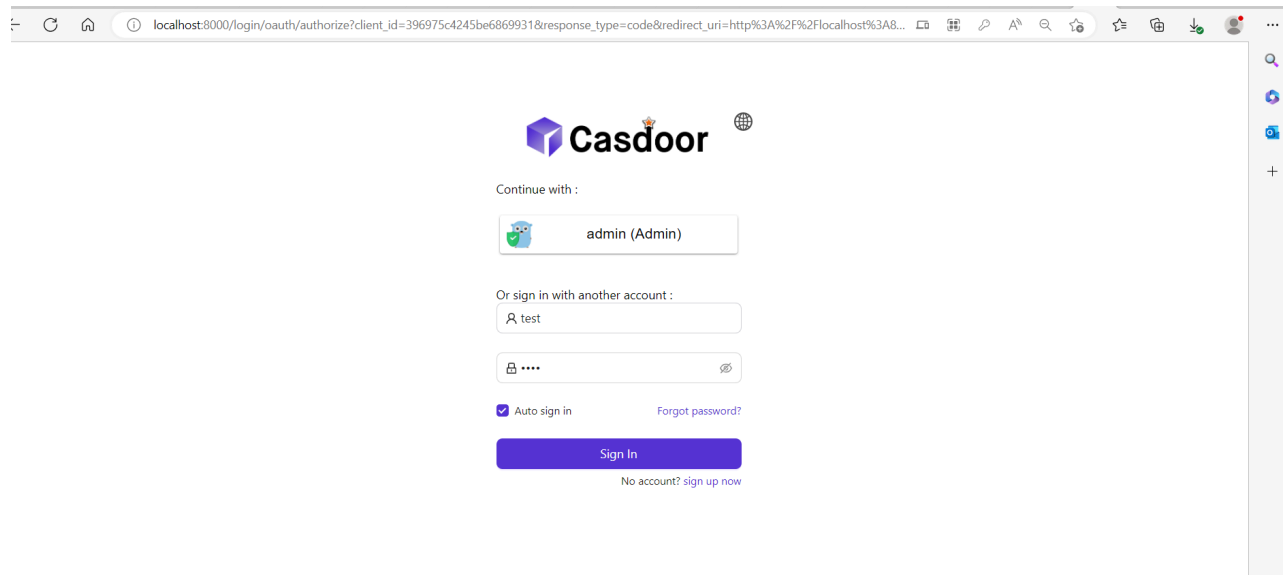
Since there is no client implementation, you can get your access token by making a request by the below URL:

```
http://localhost:8000/login/oauth/authorize?client_id=<client ID>&response_type=code&redirect_uri=http%3A%2F%2Flocalhost%3A8080%2Flogin&scope=read&state=app-built-in<public certificate>>
```

Change `client ID` to the ID you copied before and input the public certificate of Casdoor, which you can find in Casdoor's Certs page.

Then input the username and password you created for Hasura before.

Click "Sign in"



Go back to the Casdoor/Token page.

localhost:8000/tokens

Casdoor Home Organizations Users Roles Permissions Models Adapters Applications Providers Resources Records Tokens Sessions ... Admin

Tokens Add

Name	Created time	Application	Organization	User	Authorization code	Access token	Action
b6ea3e35-abc4-41d8-a1a2-01f00fd8264b	2023-04-12 13:06:53	hasura	hasura	test	433b504b4f6a593e4a11	eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0e...	Edit Delete
16024557-df21-4779-bfb9-959e5dae078c	2023-04-12 12:51:47	hasur	built-in	test1	2879bc282019cf7f23c	eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0e...	Edit Delete
f3cb1070-c2d4-40f0-8bc0-59919d26d162	2023-04-11 15:04:00	hasura	hasura	test	2a370971798d403c6ef	eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0e...	Edit Delete
64993582-2322-4df7-ab20-cb23201bc77b	2023-04-11 00:37:22	springboot	built-in	admin	a2396037c3ba4fd9221e	eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0e...	Edit Delete
f65a3813-a655-47f0-9c9a-f08ce4607815	2023-04-11 00:31:37	springboot	built-in	admin	d048c7f9cd1469fd829d	eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0e...	Edit Delete
5828069e-15eb-4c92-933c-fecda8ed621c	2023-04-11 00:06:54	springboot	built-in	admin	7cc27dc752cc4188ac8d	eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0e...	Edit Delete
2277e0f2-7e78-462f-a654-3c53759784af	2023-04-11 00:05:17	springboot	built-in	admin	56141e709a06931b7faa	eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0e...	Edit Delete
55bd324a-6039-40f6-b707-2a55d78ae911	2023-04-11 00:05:07	springboot	built-in	admin	9a1413bc172591a64353	eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0e...	Edit Delete
4b30acbe-fa22-4387-8098-9a4e670f6972	2023-04-10 23:59:19	springboot	built-in	admin	88b0997b675917f20fdc	eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0e...	Edit Delete

Find the Username you input before, then click "edit"

Copy the Access Token

Edit Token Save Save & Exit

Name: b6ea3e35-abc4-41d8-a1a2-01f00fd8264b

Application: hasura

Organization: hasura

User: test

Authorization code: 433b504b4f6a593e4a11

Access token: eyJhbGciOiJSUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtaW4iLCJ0eXAiOiJKV1QiLCJ0eXNpd25iOiI6Imhhc3VyYSIsIm5hbWUiOiJ0eXN0iWiwY3JlYXRIZFRpbWUjOiJyMDIzTA0LTASVDE1OjI5OjU2KzA4OjAwIiwidXBkYXRIZFRpbWUjOiJpZCI6IjRlYy...

Expires in: 604800

Scope: read

Token type: Bearer

Save Save & Exit

Now you can use the access token to make the authenticated request. Hasura returned the appropriate user rather than returning all the users from the database.

GraphQL Endpoint
Request Headers

ENABLE	KEY	VALUE	
<input type="checkbox"/>	Hasura-Client-Name	casdoor	✕
<input checked="" type="checkbox"/>	content-type	application/json	✕
<input type="checkbox"/>	x-hasura-admin-secret	🔒 ✕
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiIsImtpZCI6ImNlcnQtYnVpbHQtYW4iLCJ0eXAiOiJKV1QiLCJ0eXciOiJkaXIiLCJvc2ciOiJ1bmh3IiwiaWF0IjoiYyYySjlmShbDwL	🔒 ✕
	Enter Key	Enter Value	

Explorer GraphQL Prettify History Explorer Code Exporter REST Derive action Analyze < Docs

query MyQuery

- users
 - distinct_on:
 - limit:
 - offsets:
 - order_by:
 - where:
 - id
 - username
 - users_by_pk

```
1 query MyQuery {
2   users {
3     id
4     username
5   }
6 }
7
```

```
+ {
+   "data": {
+     "users": [
+       {
+         "id": "4ac7ccce-ec7b-4191-a78d-e11f50686f8b",
+         "username": "test"
+       }
+     ]
+   }
+ }
```

QUERY VARIABLES

Python

JumpServer

Using CAS to connect JumpServer

JumpServer

[Casdoor](#) can be used to connect [JumpServer](#).

The following are some of the names in the configuration:

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

`JumpServer_HOSTNAME`: The domain name or IP where JumpServer is deployed.

Step 1: Deploy Casdoor and JumpServer

Firstly, deploy [Casdoor](#) and [JumpServer](#).

After successful deployment, ensure the following:

1. Casdoor can be logged in and used normally.
2. You can set `CASDOOR_HOSTNAME` to `http://localhost:8000` when deploying Casdoor in `prod` mode. See [production mode](#).

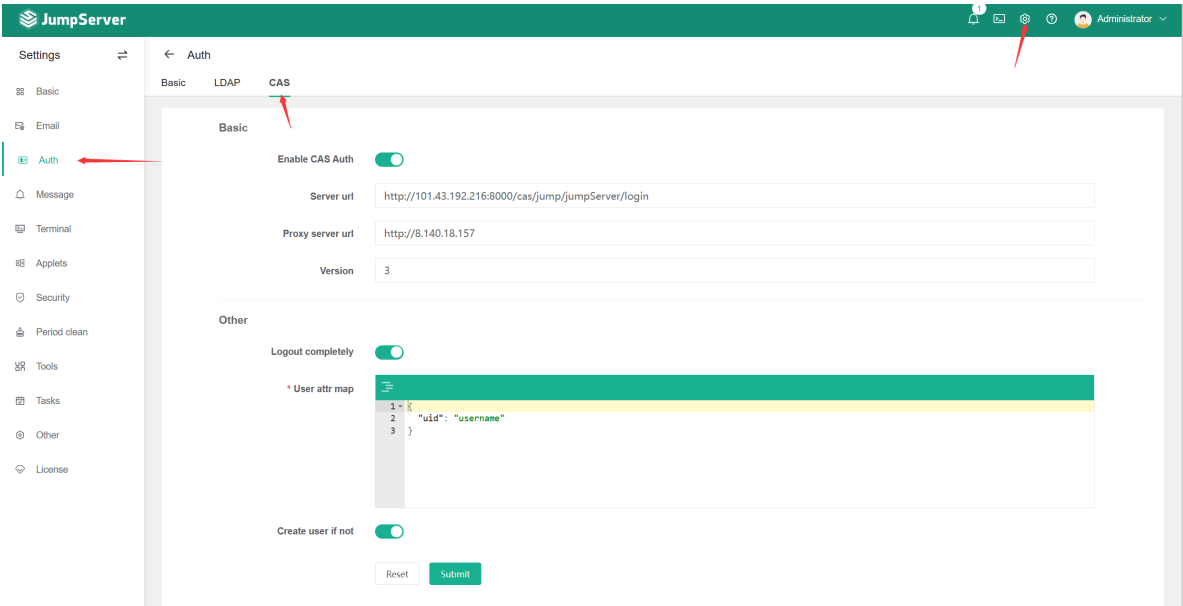
Step 2: Configure Casdoor application

1. Create a new Casdoor application or use an existing one.
2. Find a redirect URL: `CASDOOR_HOSTNAME/cas/your organization/your application/login`.
3. Add your redirect URL to the Casdoor application: `JumpServer_HOSTNAME`.

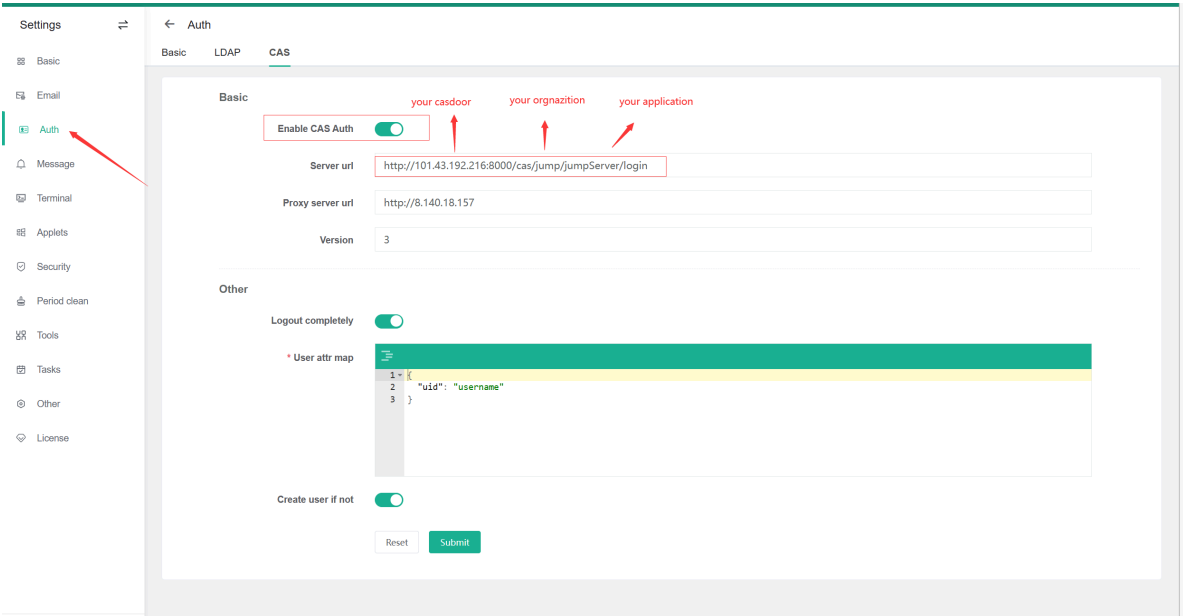
For more information about [CAS](#), refer to the documentation.

Step 3: Configure JumpServer

1. Find Auth:



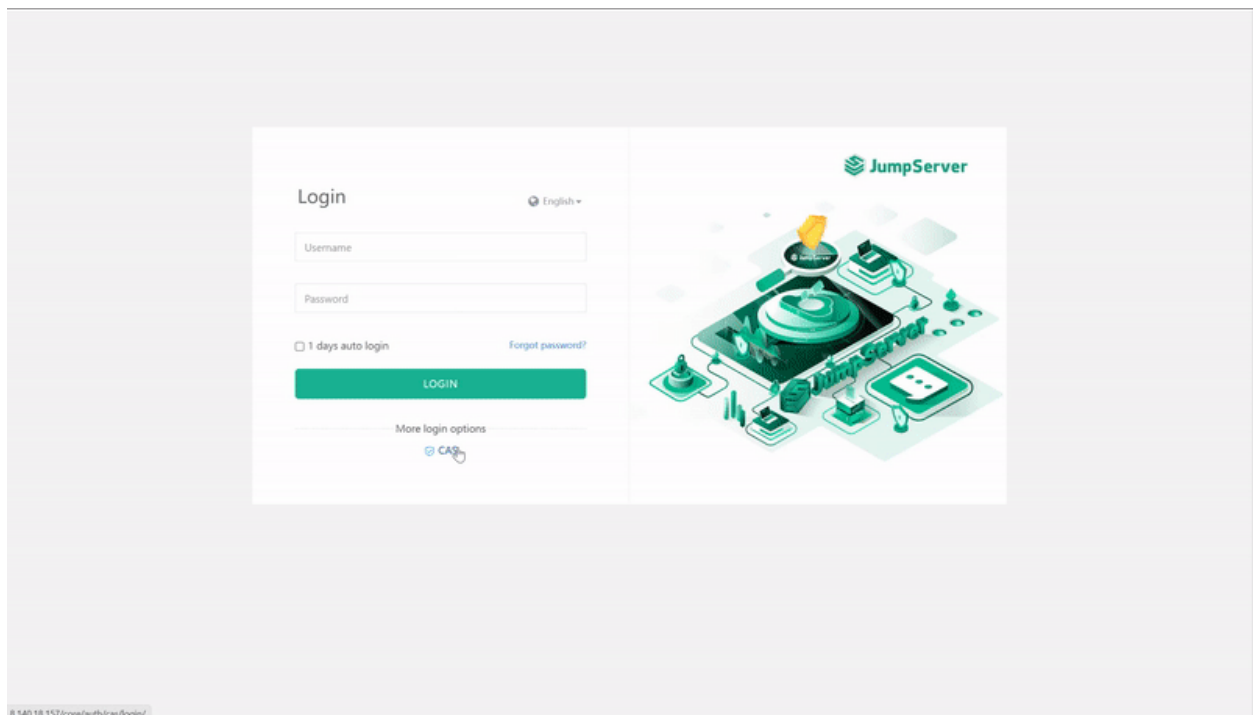
2. Configure this app:



- `/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`.

For more information about [CAS](#) and [JumpServer](#), refer to the documentation.

Log out of JumpServer and test SSO:



Monitoring

Web UI

Monitor runtime information on the Casdoor web page

Prometheus

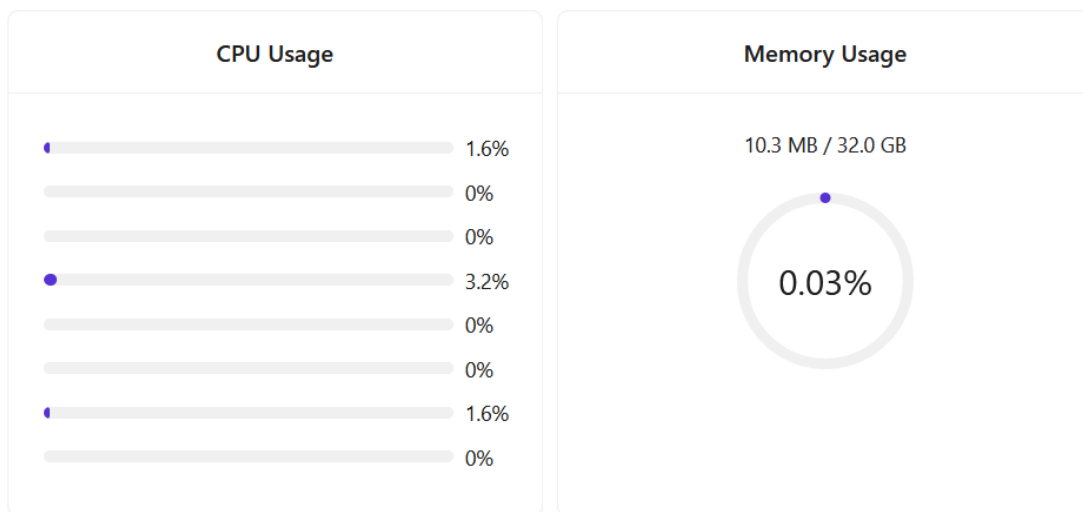
Use Prometheus to collect information about running Casdoor.

Web UI

You can monitor the runtime information of Casdoor on the [Casdoor web page](#), including CPU Usage, Memory Usage, API Latency, and API Throughput.

On the UI, you can view the following information:

- CPU Usage and Memory Usage



- API Latency, including count times and average latency

API Latency			
GET	/api/get-cert	3	0.667
GET	/api/get-certs	27	1.333
GET	/api/get-chats	27	1.519
GET	/api/get-default-application	3	5.333
GET	/api/get-email-and-phone	1	1.000
GET	/api/get-global-providers	58	1.292

- API Throughput, including total throughput and throughput per API

API Throughput		
Total Throughput: 2		
Name	Method	Throughput
/api/get-prometheus-info	GET	1
/api/get-system-info	GET	1

Prometheus

To collect Casdoor's runtime metrics, such as API Throughput, API Latency, CPU Usage, Memory Usage, and more, you need to configure your Prometheus profile.

```
global:
  scrape_interval: 10s # The time interval for fetching metrics

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
  - job_name: 'casdoor' # Name of the application to be monitored
    static_configs:
      - targets: ['localhost:8000'] # Back-end address of Casdoor
      deployment
    metrics_path: '/api/metrics' # Path for collecting indicators
```

After successful configuration, you will find the following information in Prometheus:



Internationalization

Casdoor supports multiple languages. By deploying the translations to [Crowdin](#), we can provide support for Spanish, French, German, Chinese, Indonesian, Japanese, Korean, and more.

Casdoor utilizes the official Crowdin CLI to synchronize translations from Crowdin. If you wish to add support for additional languages, please submit your proposal in [our community](#). Moreover, if you would like to contribute to expediting the translation work, kindly consider assisting us in translating on [Crowdin](#).

Contributor Guide

Welcome to Casdoor! This document serves as a guideline on how to contribute to Casdoor.

If you find any incorrect or missing information, please leave your comments or suggestions.

Get Involved

There are many ways to contribute to Casdoor. Here are some ideas to get started:

- Use Casdoor and report issues. When using Casdoor, report any issues - whether they're bugs or proposals - on [GitHub Discussions](#) or on [Discord](#) before filing an issue on GitHub.

INFO

Please use English to describe the details of your problem when reporting an issue.

- Help with documentation. Starting your contribution work with docs is a good choice.
- Help solve issues. We have a table that contains easy tasks suitable for beginners under [Casdoor Easy Tasks](#), with different levels of challenges labeled with different tags.

Contributing

If you are ready to create a PR, here is the workflow for contributors:

1. Fork to your own repository.
2. Clone your fork to a local repository.
3. Create a new branch and work on it.
4. Keep your branch in sync.
5. Commit your changes. Make sure your commit message is concise.
6. Push your commits to your forked repository.
7. Create a pull request from your branch to our **master** branch.

Pull Requests

Before You Get Started

Casdoor uses GitHub as its development platform, and pull requests are the primary source of contributions.

Here are some things you need to know before opening a pull request:

- You need to sign the CLA when you first create a pull request.
- Explain why you are submitting the pull request and what it will do to the repo.

- Only one commit is allowed. If the PR does more than one thing, please split it.
- If there are any newly added files, please include the Casdoor license at the top of the new file(s).

```
// 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.  
// You may obtain a copy of the License at  
//  
//     http://www.apache.org/licenses/LICENSE-2.0  
//  
// Unless required by applicable law or agreed to in writing,  
// software  
// distributed under the License is distributed on an "AS IS"  
// BASIS,  
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or  
// implied.  
// See the License for the specific language governing permissions  
// and  
// limitations under the License.
```

Semantic PRs

Your pull requests should follow the Conventional Commits spec. The basic requirement is that only the PR title or at least one commit message. For example, three commonly used PR titles are given below:

CAUTION

PR titles must be in lowercase.

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` adds or improves documentation.

```
docs: correct spelling of CHANGELOG
```

For more details, please refer to the [Conventional Commits](#) page.

Linking PRs with Issues

You can link a pull request to an issue to show a fix is in progress and to automatically close the issue when the pull request is merged.

Linking a Pull Request to an Issue Using a Keyword

You can link a pull request to an issue by using a supported keyword in the pull request's description or in a commit message. The pull request **must be** on the default branch.

- close
- fix
- resolve

An issue in the same repository, for instance:

Fix: #902

For more details, please refer to [Linking a Pull Request to an Issue](#).

Modifying PRs

Your PR may need revision. Please modify the same PR when the code needs changes; don't close the PR and open a new one. Here is an example:

- Modify the code on your local.
- Modify your commit.

```
git commit --amend
```

- Push to your remote repository.

```
git push --force
```

Then, you will have successfully modified the PR!

Code Related

Some Principles:

- Readability: important code should be well-documented. Code style should comply with the existing one.

Naming Convention

For instance, `signupUrl` for variable names, `Signup URL` for UI.

How to Update i18n Data?

Please note that we use [Crowdin](#) as a translating platform and `i18next` as a translating tool. When you add strings using `i18next` in the `web/` directory, you can run the `i18n/generate_test.go` to auto-generate `web/src/locales/**/data.json`.

Run `i18n/generate_test.go`:

```
cd i18n && go test
```

All languages are filled in English by default. After your PR has been merged, you are encouraged to help translate the newly added strings in `web/src/locales/zh/data.json` by [Crowdin](#).

CAUTION

If you are not familiar with a language, please do not translate it; keep the file as it is.

License

By contributing to Casdoor, you agree that your contributions will be licensed under the Apache License.