Overview

Casibase is an open-source Domain Knowledge Database & IM & Forum Software powered by ChatGPT.

You need to enable JavaScript to run this app.

Casibase features

1. With a separate front-end and back-end architecture developed in Golang, Casibase supports high concurrency, provides web-based management UI and supports multiple languages (Chinese, English).

2. Casibase supports third-party application login, such as GitHub, Google, QQ, WeChat, etc., and supports the extension of third-party login with plugins.

3. Based on embedding and prompt engineering for knowledge management, Casibase supports customized embedding methods and language models.

4. Casibase supports integration with existing systems by db sync, so users can transition to Casibase smoothly.

5. Casibase supports mainstream databases: MySQL, PostgreSQL, SQL Server, etc., and supports the extension of new databases with plugins.

6. Casibase is a powerful tool for asset management, enabling easy connection to assets via RDP, VNC, and SSH protocols and efficient handling of remote connections to machines.

7. Casibase's Security Log Auditing feature allows you to effortlessly track and monitor remote connections with detailed audit logging, recording connection start time, duration, and other relevant details, and also enables you to capture and analyze API logs for Casdoor operations, enhancing security and operational transparency.

8. Casibase supports database management. Casibase's Database Management feature allows you to easily connect, manage, and organize databases while controlling access and simplifying user management and authorization for

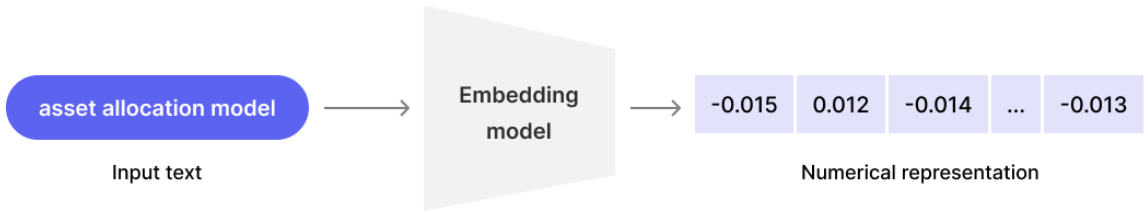
database resources.

9. Casibase is an open-source container cloud platform based on Docker and Kubernetes. It is suitable for individuals or organizations to build their own dedicated container cloud environment. Based on the Casbin permission management engine, Casibase implements fine-grained access control policies. Users can easily create, orchestrate, and manage container applications on Casibase. The project focuses on optimizing Casibase's application orchestration, service governance, and platform visualization core functions, improving platform usability and manageability, making it a leading lightweight container cloud platform.

How it works

Step 0 (Pre-knowledge)

Casibase's knowledge retrieval process is based on embedding and prompt engineering, so it is highly recommended that you take a brief look at how embedding works. An introduction to Embedding.



Step 1 (Importing Knowledge)

To get started with Casibase, users need to follow these steps to import knowledge and create a domain-specific knowledge database:

1. Configure Storage: In the Casibase dashboard, users should first configure the storage settings. This involves specifying the storage system to be used for storing knowledge-related files, such as documents, images, or any other relevant data. Users can choose from a variety of storage options based on their preferences and requirements.

2. Upload Files to Storage: Once the storage is set up, users can proceed to upload files containing domain-specific knowledge into the configured storage system. These files can be in various formats, such as text documents, images, or structured data files like CSV or JSON.

3. Select Embedding Method for Knowledge Generation: After the files are uploaded, users have the option to choose the embedding method for generating knowledge and corresponding vectors. Embeddings are numerical representations of textual or visual content, which facilitate efficient similarity search and data analysis.

> TIP
>
> How knowledge is embedded?
>
> - For textual data: Users can choose from various embedding methods, such as Word2Vec, GloVe, or BERT, to convert the textual knowledge into meaningful vectors.
>
> - For visual data: If the uploaded files contain images or visual content,

users can select image embedding techniques like CNN-based feature extraction to create representative vectors.

- More methods coming soon...

By following these steps, users can populate their domain knowledge database with relevant information and corresponding embeddings, which will be used for effective searching, clustering, and retrieval of knowledge within Casibase. The embedding process allows the system to understand the context and relationships between different pieces of knowledge, enabling more efficient and insightful knowledge management and exploration.

Step 2 (Retrieving Knowledge)

After importing your `domain knowledge`, Casibase transforms it into `vectors` and stores these vectors in a `vector database`. This vector representation enables powerful functions like `similarity search` and `efficient retrieval of related information`. You can quickly find relevant data based on context or content, enabling advanced querying and uncovering valuable insights within your domain knowledge.

Step 3 (Building the Prompt)

Casibase performs a similarity search on the stored knowledge vectors to find the closest match to the user's query. Using the search results, it creates a `prompt template` to frame a specific question for the `language model`. This ensures accurate and contextually relevant responses, delivering comprehensive answers based on the domain knowledge in Casibase.

Step 4 (Achieving the Goal)

At this stage, using Casibase, you have successfully acquired the knowledge you sought. Through the innovative combination of domain knowledge transformed into vectors and powerful language models like ChatGPT, Casibase provides you with accurate and relevant responses to your inquiries. This enables you to efficiently access and utilize the domain-specific information stored in Casibase, meeting your knowledge requirements with ease.

Step 5 (Optional Fine-tuning)

If you find that the results are not entirely satisfactory, you can try to get better results by doing the following:

- Tweaking Language Model Parameters
- Asking multiple questions
- Optimizing the original files

By utilizing these fine-tuning options, you can improve the efficiency of your knowledge management in Casibase, ensure that the system is better aligned with your goals, and provide more accurate and insightful information.

> ⓘ HINTS
>
> Other ways to optimize results (may require source code changes):
>
> - Updating `Embedding` Results: Refine the knowledge representation by adjusting the embedding results of your domain knowledge.

> - Modifying `Prompt` Templates: By customizing the prompts, you can elicit more precise responses from the language model.
> - Exploring Different `Language Models`: Experiment with different models to find the one that best suits your requirements for generating responses.

Online demo

Read-only site (any modification operation will fail)

- Chat bot (https://ai.casibase.com)
- Admin UI (https://ai-admin.casibase.com)

Writable site (original data will be restored for every 5 minutes)

- Chat bot (https://demo.casibase.com)
- Admin UI (https://demo-admin.casibase.com)

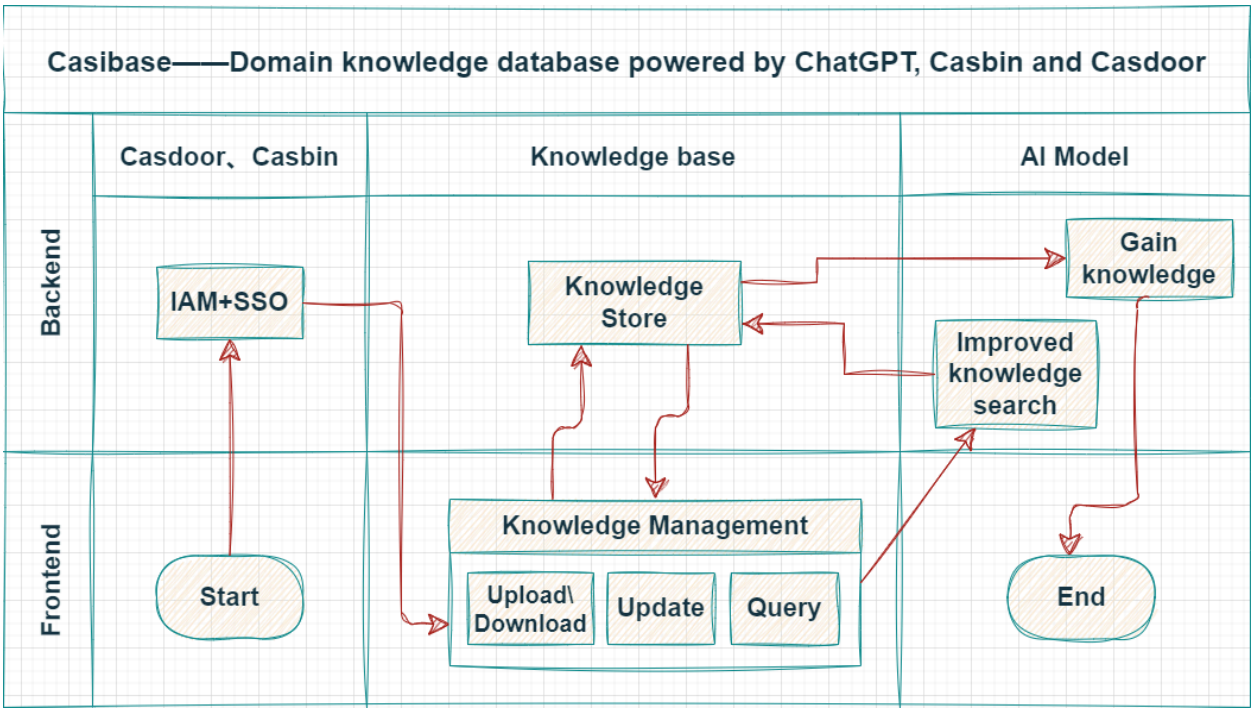
Global admin login:

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

Architecture

Casibase contains 2 parts:

Name	Description	Language	Source code
Frontend	User interface for the casibase application	JavaScript + React	https://github.com/casibase/casibase/tree/master/web
Backend	Server-side logic and API for casibase	Golang + Beego + MySQL	https://github.com/casibase/casibase



Supported Models

Language Model

Model	Sub Type	Link
OpenAI	gpt-4-32k-0613`gpt-4-32k-0314`gpt-4-32k`gpt-4-0613`gpt-4-0314`gpt-4`gpt-3.5-turbo-0613`gpt-3.5-turbo-0301`gpt-3.5-turbo-16k`gpt-3.5-turbo-16k-0613`gpt-3.5-turbo`text-davinci-003`text-davinci-002`text-curie-001`text-babbage-001`text-ada-001`text-davinci-001`davinci-instruct-beta`davinci`curie-instruct-beta`curie`ada`babbage	OpenAI
Hugging Face	meta-llama/Llama-2-7b, tiiuae/falcon-180B, bigscience/bloom, gpt2, baichuan-inc/Baichuan2-13B-Chat, THUDM/chatglm2-6b	Hugging Face
Claude	claude-2, claude-v1, claude-v1-100k, claude-instant-v1, claude-instant-v1-100k, claude-v1.3, claude-v1.3-100k, claude-v1.2, claude-v1.0, claude-instant-v1.1, claude-instant-v1.1-100k, claude-instant-v1.0	Claude
OpenRouter	google/palm-2-codechat-bison, google/palm-2-chat-bison, openai/gpt-3.5-turbo, openai/gpt-3.5-turbo-16k, openai/gpt-4,	OpenRouter

Model	Sub Type	Link
	openai/gpt-4-32k, anthropic/claude-2, anthropic/claude-instant-v1, meta-llama/llama-2-13b-chat, meta-llama/llama-2-70b-chat, palm-2-codechat-bison, palm-2-chat-bison, gpt-3.5-turbo, gpt-3.5-turbo-16k, gpt-4, gpt-4-32k, claude-2, claude-instant-v1, llama-2-13b-chat, llama-2-70b-chat	
Ernie	ERNIE-Bot, ERNIE-Bot-turbo, BLOOMZ-7B, Llama-2	Ernie
iFlytek	spark-v1.5, spark-v2.0	iFlytek
ChatGLM	chatglm2-6b	ChatGLM
MiniMax	abab5-chat	MiniMax
Local	custom-model	Local Computer

Embedding Model

Model	Sub Type	Link
OpenAI	AdaSimilarity, BabbageSimilarity, CurieSimilarity, DavinciSimilarity, AdaSearchDocument, AdaSearchQuery, BabbageSearchDocument, BabbageSearchQuery, CurieSearchDocument, CurieSearchQuery, DavinciSearchDocument,	OpenAI

Model	Sub Type	Link
	DavinciSearchQuery, AdaCodeSearchCode, AdaCodeSearchText, BabbageCodeSearchCode, BabbageCodeSearchText, AdaEmbeddingV2	
Hugging Face	sentence-transformers/all-MiniLM-L6-v2	Hugging Face
Cohere	embed-english-v2.0, embed-english-light-v2.0, embed-multilingual-v2.0	Cohere
Ernie	default	Ernie
Local	custom-embedding	Local Computer

Core Concepts

As Casibase's user, you should get familiar with at least 4 core concepts: `Provider`, `Storage`, `Chat` and `Vector`.

Providers

Providers are the backbone of Casibase, offering essential services and integration with external systems. The Provider class definition is shown as follows:

```
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"`
    ClientId     string `xorm:"varchar(100)" json:"clientId"`
    ClientSecret string `xorm:"varchar(2000)" json:"clientSecret"`
    ProviderUrl  string `xorm:"varchar(200)" json:"providerUrl"`
}
```

> TIP
>
> There are two primary types of providers in Casibase:
>
> - Storage Providers. The Storage Providers facilitates the storage and

retrieval of data within Casibase. It supports various storage options, including:

  - AWS
  - Azure
  - Local File System

- **AI Providers.** The AI Providers are responsible for handling AI-related tasks and services in Casibase. It supports multiple AI models and technologies, including:

  - OpenAI
  - ChatGLM
  - InternLM

Vectors

Vectors in Casibase represent numerical representations of different types of data. These vectors enable efficient processing and analysis of information. Some of the vector types available are:

- Text Vector
- Image Vector
- ... (other vector types)

The Vector class definition is shown as follows:

```
type Vector struct {
    Owner       string    `xorm:"varchar(100) notnull pk"
```

Chats

Chats are at the core of interactive communication between users and the AI models in Casibase. They consist of three essential components:

- Question: The user's input or query, seeking information or assistance.
- Query Prompt: A formatted version of the user's question, prepared for processing by the AI models.
- Answer: The AI-generated response to the user's question, providing relevant information or solutions.

The Chat class definition is shown as follows:

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

    DisplayName  string   `xorm:"varchar(100)" json:"displayName"`
    Category     string   `xorm:"varchar(100)" json:"category"`
    Type         string   `xorm:"varchar(100)" json:"type"`
    User1        string   `xorm:"varchar(100)" json:"user1"`
    User2        string   `xorm:"varchar(100)" json:"user2"`
    Users        []string `xorm:"varchar(100)" json:"users"`
    MessageCount int      `json:"messageCount"`
}
```

Embedding

Embedding is the process of transforming various types of data, such as text and images, into dense vector representations. This step is crucial for facilitating efficient data processing and analysis within Casibase.

TIP

- By embedding, the questions in chat and the knowledge files in storage will be turned into vectors and used in the next step of knowledge search.
- Casibase's default embedding method is provided by OpenAI at a rate of up to three calls per minute. We recommend minimizing coupling between knowledge files to facilitate embedding and further processing.

Server Installation

Requirements

OS

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

Environment

- Go 1.20+
- Node.js LTS (18)
- Yarn 1.x

> INFO
>
> The use of Casibase is divided into two steps:
>
> - step1: Deploy and run Casdoor
> - step2: Deploy and run Casibase (this docs)
>
> We strongly suggest you use Yarn 1.x to run & build Casdoor&Casibase frontend, using NPM might cause UI styling issues, see more details at: casdoor#294

> CAUTION
>
> For **Chinese** users, in order to download the Go dependency packages successfully, you need to use a Go proxy by Configuring the GOPROXY environment variable. We strongly recommend: https://goproxy.cn/

Database

Casibase uses XORM to talk to the database. Based on Xorm Drivers Support, Casibase currently provides support for the following databases:

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

guacd

Casibase uses guacamole-server to provide remote desktop access. If you want to use this feature, you need to install guacamole-server first. If you haven't installed guacamole-server, please refer to guacamole-server Installation.

You can also run guacd in docker with the following command:

```
docker run -d --name guacd -p 4822:4822 guacamole/guacd
```

Download

The source code of Casibase is hosted at GitHub: https://github.com/casibase/casibase. Both the Go backend code and React frontend code are inside the

single repository.

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

Casibase supports `Go Modules`. To download the code, you can just simply clone the code via git:

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

Configuration

Configure Casdoor

Please refer to Casdoor-SSO section to configure Casdoor.

Remember your `clientId`, `clientSecret`, `organization`, `application` and so on in Casdoor configuration, we will use them later.

Configure Database

Casibase supports mysql, mssql, sqlite3, postgres. Casibase uses mysql by

default.

MySQL

Casibase will store its users, nodes and topics information in a MySQL database named: `casibase`. If the database does not exist, it needs to be created manually. The DB connection string can be specified at: https://github.com/casibase/casibase/blob/master/conf/app.conf

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

PostgreSQL

Since we must choose a database when opening Postgres with xorm, you should prepare a database manually before running Casibase.

Let's assume that you have already prepared a database called `casibase`, then you should specify `app.conf` like this:

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

> INFO
>
> For PostgreSQL, make sure `dataSourceName` has non-empty `dbName` and leave the standalone `dbName` field empty like the above example.

CockroachDB

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

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

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

Sqlite3

You should specify `app.conf` like this:

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

Custom configuration

Casibase supports custom configuration, you can modify the configuration file `conf/app.conf` to change the configuration.

```
casdoorEndpoint = <Your Casdoor endpoint>
clientId = <Your Casdoor application's client ID>
clientSecret = <Your Casdoor application's client secret>
casdoorOrganization = <Your Casdoor organization name>
casdoorApplication = <Your Casdoor application name>
```

Run

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

> ⚠ CAUTION
>
> **Casibase** requires **Casdoor** to provide access control and some back-end services, so you must make sure **Casdoor** is running properly before running **Casibase**.
>
> How to install and run Casdoor:
>
> - Casdoor Installation

Development mode

Backend

Casibase's Go backend runs at port 14000 by default. You can start the Go backend with the following command:

```
go run main.go
```

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

Frontend

Casibase's frontend is a very classic Create-React-App (CRA) project. It runs at port `13001` by default. Use the following commands to run the frontend:

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

Production mode

Backend

Build Casibase Go backend code into executable and start it.

For Linux:

```
go build
./casibase
```

For Windows:

```
go build
casibase.exe
```

Frontend

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

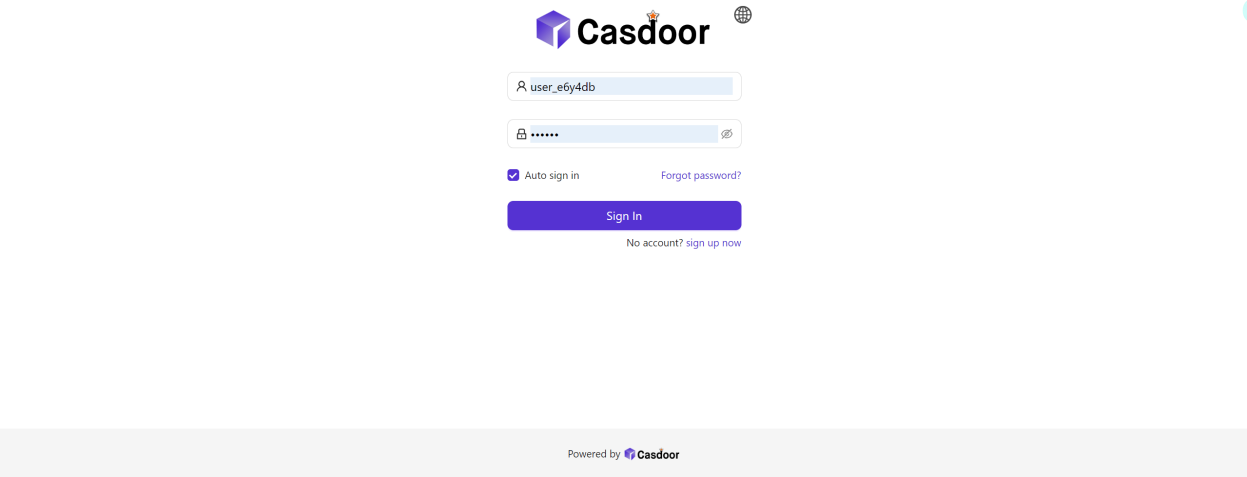
```
cd web
```

Nginx

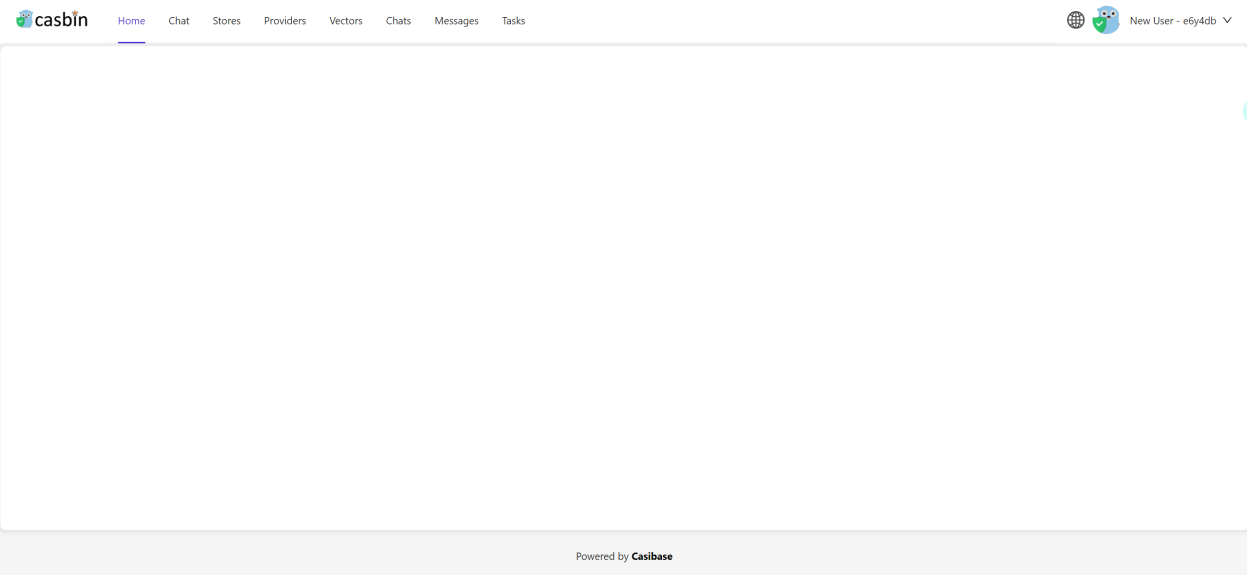
> TIP
>
> If you use nginx as a reverse proxy, you need to add the following configuration to the nginx configuration file:
>
> ```
> location / {
>     *** your configuration ***
>     proxy_set_header Upgrade $http_upgrade;
>     proxy_set_header Connection "upgrade";
> }
> ```
>
> Because Casibase uses websocket to communicate with guacd.

Preview

Visit: `http://localhost:13001` in your browser. Login into Casibase dashboard with the user account you have just registered in Casdoor:



Then you will go to the home page of Casibase:



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

(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. Casibase'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 use docker-compose, please ensure you have **docker-compose version >= 2.2**. For Linux users, note that docker-compose needs to be installed separately from docker-engine.

Get the image

We have provided two DockerHub images:

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

1. casbin/casibase-all-in-one: This image includes the casibase binary, a MySQL database, and all the necessary configurations. It is designed for new users who want to try Casibase quickly. With this image, you can start Casibase 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 `14000` exposed to the host. The image will be automatically pulled if it doesn't exist on the local host.

```
docker run -p 14000:14000 casbin/casibase-all-in-one
```

Visit http://localhost:14000 in your browser. Log into the Casibase dashboard with the default global admin account: `built-in/admin`

```
admin
123
```

Option-2: 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 Casibase. For more details about `app.conf`, you can see Via Ini file.

Below is a minimal but complete `docker-compose.yml` example that starts a MySQL database and the Casibase service. It configures Casibase to connect to the database using MySQL. Save this file as `docker-compose.yml` (next to a `conf` folder if you want to mount a custom `app.conf`).

```
services:
  db:
    image: mysql:8.0
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: 123456
      MYSQL_DATABASE: casibase
    volumes:
      - db_data:/var/lib/mysql
    ports:
      - "3306:3306" # optional: expose DB to host

  casibase:
```

What does the above compose file do:

- The Casibase container connects to the database using the Compose service name `db` (i.e. `db:3306`). When both services run in the same Docker network (default for compose), using the service name as host is the simplest and most reliable approach.
- The `dataSourceName` above uses the MySQL root account for simplicity. For production use please create a dedicated DB user and a strong password.
- Mounting `./conf/app.conf` into `/conf/app.conf` is optional. If you prefer environment variables, you can remove the mount and rely on the `driverName` and `dataSourceName` variables.
- If both `app.conf` and environment variables are provided, the environment variables take precedence and will override the corresponding settings in app.conf.

> NOTE
>
> Casdoor: By default Casibase uses the hosted Casdoor instance at `https://door.casdoor.com` for user authentication. If you need to manage users, applications, or customize the authentication flow, you must deploy your own Casdoor instance and update Casibase's `app.conf` (or the equivalent environment variables) to point to your Casdoor server. You can look at Casdoor configuration for more details.
>
> **`RUNNING_IN_DOCKER`**: By default `RUNNING_IN_DOCKER` is enabled in docker image. When enabled, Casibase replaces `localhost` with the Docker bridge address (for example, `host.docker.internal` or the equivalent bridge hostname) so that the container can reach services running on the host.

Bring up the services:

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

Check logs (follow):

```
docker-compose logs -f casibase
```

Visit http://localhost:14000 in your browser. Log into the Casibase dashboard with the default global admin account: `built-in/admin`

```
admin
123
```

Stop and remove containers and volumes (data removed):

```
docker-compose down -v
```

Option-3: 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)/' \
> ```

Create `conf/app.conf`. You can copy it from conf/app.conf in Casibase. For more details about `app.conf`, you can see Via Ini file.

Then run

```
docker run  -p 14000:14000 -v /folder/of/app.conf:/conf casbin/
casibase:latest
```

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

Visit http://localhost:14000 in your browser. Log into the Casibase dashboard with the default global admin account: `built-in/admin`

```
admin
123
```

(Optional) Try with K8s Helm

Introduction

This guide shows how to deploy Casibase on Kubernetes using Helm for easy and scalable management. Helm simplifies the deployment process and allows for easy configuration management.

Prerequisites

- A running Kubernetes cluster
- Helm v3 installed
- kubectl configured to connect to your cluster
- A MySQL/PostgreSQL database (recommended for production)
- A Casdoor instance for authentication

Configuration

Before installation, you must create an application configuration file. The Helm chart will not work with default values.

Pre step: Create app.conf File

Create an `app.conf` file with your specific settings:

You can view more details about the configuration options in the Casibase Configuration Documentation.

Or check the latest configuration options in app.conf example.

```
appname = casibase
httpport = 14000
runmode = prod
SessionOn = true
copyrequestbody = true

# Database Configuration - REQUIRED
driverName = mysql
dataSourceName = your-username:your-password@tcp(your-db-host:3306)/
dbName = your-database

# Casdoor Authentication - REQUIRED
casdoorEndpoint = https://door.casdoor.com
clientId = your-client-id
clientSecret = your-client-secret
casdoorOrganization = "your-organization"
casdoorApplication = "your-application"
redirectPath = /callback

# Optional Settings
redisEndpoint =
guacamoleEndpoint = 127.0.0.1:4822
isDemoMode = false
disablePreviewMode = false
logPostOnly = true
landingFolder =
cacheDir = "C:/casibase_cache"
appDir = ""
isLocalIpDb = false
audioStorageProvider = ""
providerDbName = ""
socks5Proxy = "127.0.0.1:10808"
publicDomain = ""
adminDomain = ""
enableExtraPages = false
shortcutPageItems = []
usageEndpoints = []
iframeUrl = ""
forceLanguage = ""
defaultLanguage = "en"
```

Optional: Using Secrets for Sensitive Data

For production environments, create a Kubernetes secret with your configuration:

```
# Create secret from app.conf file
kubectl create secret generic casibase-config --from-
file=app.conf=./app.conf
```

Installation Steps

Step 1: Prepare Configuration Files

Ensure you have created both:

- `app.conf` - Application configuration

Step 2: Install with Configuration File

> INFO
>
> Visit the Casbin Helm Chart to find the latest version.

Install Casibase by passing the app.conf file directly:

```
# Method 1: Pass app.conf content as appConfig parameter
helm install casibase oci://registry-1.docker.io/casbin/casibase-helm-
chart \
  --version v1.549.0 \
  --set-file appConfig=./app.conf
```

Alternative Installation with Secret

If using secrets for sensitive data:

```
# Create secret first
kubectl create secret generic casibase-config --from-
file=app.conf=./app.conf

# Install with secret reference (no additional values file needed)
helm install casibase oci://registry-1.docker.io/casbin/casibase-helm-
chart \
  --version v1.549.0 \
  --set appConfig="" \
  --set appConfigFromSecret=casibase-config
```

Step 3: Verify Installation

Check the deployment status:

```
kubectl get pods
kubectl get services
kubectl logs -l app.kubernetes.io/name=casibase
```

Step 4: Access Casibase

Once installed, Casibase will be accessible through the Kubernetes service on port 14000. If you enabled ingress, it will be available at your configured domain.

Configuration Options Reference

The following table shows the main configuration parameters available in the Helm chart:

Parameter	Description	Default Value
`replicaCount`	Number of Casibase replicas to run	`1`
`image.repository`	Docker image repository	`casbin`
`image.name`	Docker image name	`casibase`
`image.pullPolicy`	Image pull policy	`IfNotPresent`
`image.tag`	Image tag (defaults to chart appVersion)	`""`
`appConfig`	Application configuration (app.conf content)	See values.yaml
`appConfigFromSecret`	Mount app.conf from secret instead	`""`

Parameter	Description	Default Value
`service.type`	Kubernetes service type	`ClusterIP`
`service.port`	Service port	`14000`
`ingress.enabled`	Enable ingress	`false`
`ingress.hosts`	Ingress hosts configuration	`[]`
`resources`	CPU/ Memory resource requests and limits	`{}`
`autoscaling.enabled`	Enable horizontal pod autoscaler	`false`
`autoscaling.minReplicas`	Minimum number of replicas	`1`
`autoscaling.maxReplicas`	Maximum number of replicas	`100`

Parameter	Description	Default Value
`autoscaling.targetCPUUtilizationPercentage`	CPU utilization threshold	`80`
`nodeSelector`	Node labels for pod assignment	`{}`
`tolerations`	Toleration labels for pod assignment	`[]`
`affinity`	Affinity settings for pod assignment	`{}`

Advanced Configuration Options

For production deployments, consider these additional options:

```yaml
# Autoscaling
autoscaling:
  enabled: true
  minReplicas: 2
  maxReplicas: 10
  targetCPUUtilizationPercentage: 70
```

Managing the Deployment

Upgrading Casibase

To upgrade your Casibase deployment to a new version:

```
helm upgrade casibase oci://registry-1.docker.io/casbin/casibase-helm-chart --version <new-version>
```

To upgrade with custom values:

```
helm upgrade casibase oci://registry-1.docker.io/casbin/casibase-helm-chart --version <new-version> \
  -f custom-values.yaml
```

Checking Deployment Status

Monitor your deployment:

```
# Check pod status
kubectl get pods -l app.kubernetes.io/name=casibase

# Check service status
kubectl get svc -l app.kubernetes.io/name=casibase

# View logs
kubectl logs -l app.kubernetes.io/name=casibase

# Describe deployment
helm status casibase
```

Uninstalling Casibase

To completely remove Casibase from your cluster:

```
helm uninstall casibase
```

Troubleshooting

Common Issues

1. **Pod not starting**: Check logs with `kubectl logs <pod-name>`
2. **Service not accessible**: Verify service configuration and ingress setup
3. **Database connection issues**: Ensure database credentials and connectivity are correct
4. **Configuration errors**: Validate your `appConfig` syntax
5. **Casdoor authentication failures**: Verify Casdoor endpoint and credentials
6. **Domain/URL issues**: Check domain configuration and DNS resolution

Configuration-Related Issues

Problem: Casdoor authentication not working

- Verify `casdoorEndpoint` is accessible from the cluster
- Check `clientId` and `clientSecret` are correct
- Ensure `redirectPath` matches your Casdoor application configuration

Problem: Configuration syntax errors

```
# Validate YAML syntax before deployment
```

Getting Help

- Check pod events: `kubectl describe pod <pod-name>`
- View Helm release info: `helm status casibase`
- Review configuration: `helm get values casibase`

Conclusion

Using Helm to deploy Casibase on Kubernetes provides a robust, scalable solution for managing your knowledge base platform. The chart offers flexible configuration options to suit various deployment scenarios, from development environments to production clusters.

Key benefits of this approach:

- **Easy deployment and updates** through Helm commands
- **Flexible configuration** through values files
- **Kubernetes-native** scaling and management
- **Production-ready** with proper resource management and health checks

For more advanced configurations and troubleshooting, refer to the Kubernetes documentation and Helm documentation.

Casibase Public API

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

- Casibase's frontend
- Casibase client SDKs (e.g., casibase-java-sdk)
- Any other customized code from the application side

The full reference for the `Casibase Public API` can be found on Swagger: https://ai-admin.casibase.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 `Casibase Public API`

Casibase Public API supports two application-level authentication methods: `Bearer Token` and `Basic Auth`. The `Bearer Token` method is recommended as it is more secure.

SDK Authentication Example (Java)

To illustrate how authentication is handled in practice, here is an example from the Casibase Java SDK. The following code shows how the SDK constructs the

credential for API requests. This process authenticates the SDK with application-level permissions, effectively acting as an admin.

```
// ...

    protected void Service(Config config, AuthTypeEnum authType)
throws Exception {
        this.config = config;
        switch (authType){
            case BASIC:
                this.credential =
Credentials.basic(config.clientId, config.clientSecret);
                break;
            case BEARER:
                String token = config.clientId + ":" +
config.clientSecret;
                this.credential = "Bearer " +
DigestUtils.md5Hex(token);
                break;
            default:
                throw new Exception("Invalid auth type");
        }
    }
```

The example above demonstrates how to prepare the credential for both authentication types:

- BASIC: It uses a helper (`okhttp3.Credentials`) to perform the standard Base64 encoding for Basic Authentication.
- BEARER: It constructs the token by taking the MD5 hash of `clientId:clientSecret` (using `org.apache.commons.codec.digest.DigestUtils`) and prepending the result with "Bearer ".

1. By `Bearer Token` (Recommended)

This method is more secure because it uses a static access token, which is a hashed value of your `clientId` and `clientSecret`.

How to get the access token?

The access token is calculated using the following formula: `md5(clientId + ":" + clientSecret)`

How to authenticate?

The access token must be provided in the `Authorization` header as a Bearer Token.

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

2. By `Basic Auth`

This method uses the `clientId` and `clientSecret` directly for authentication. It is considered less secure because the `clientSecret` might be exposed. It is supported for convenience and compatibility purposes.

How to authenticate?

HTTP Basic Authentication: This is the standard way.

```
Authorization: Basic <The Base64 encoding of
"clientId:clientSecret">
```

If you are not familiar with Base64 encoding, you can use a library for this, as

`HTTP Basic Authentication` is a widely supported standard.

Where to find the Client ID and Secret?

Both authentication methods require a `clientId` and `clientSecret`. You can find these values for your application in the Casibase configuration file: `conf/app.conf`.

Container Cloud

Casibase is an open-source **Container Cloud Platform** built on the foundations of **Docker** and **Kubernetes**. It is designed for individuals and organizations to easily build, manage, and operate their own private cloud environments with a focus on simplicity and application-centric management.

The Challenge: Complexity in the Cloud-Native Era

In the world of modern software, Kubernetes has become the standard for running applications. However, its power comes with significant complexity. Deploying even a seemingly simple application, like a WordPress blog, requires orchestrating a multitude of distinct Kubernetes resources:

- **Deployments** to manage the application pods (the WordPress server itself).
- **Services** to expose the application to the network.
- **PersistentVolumeClaims** to request storage for the database and uploads.
- **StatefulSets** to manage the database pods (like MySQL).
- **ConfigMaps** and **Secrets** to handle configuration and sensitive data.

Managing these individual components manually is often called a "resource-centric" approach. This approach presents several challenges:

1. **High Learning Curve**: Users must have a deep understanding of various Kubernetes resources and how they interact.
2. **Operational Burden**: Manually creating, updating, and deleting these resources is tedious and prone to human error.
3. **Lack of Atomicity**: There is no way to treat the entire "WordPress application"

as a single, atomic unit. You cannot simply "install" or "uninstall" it with one action.

4. **Inconsistency**: Ensuring that the application is deployed identically across development, testing, and production environments is difficult.

The Casibase Approach: From Managing Resources to Managing Applications

Casibase fundamentally simplifies this process by shifting the focus from managing individual resources to managing the **application as a whole**. We believe you should be able to manage your applications without getting lost in the weeds of Kubernetes YAML configurations.

To achieve this, Casibase introduces a powerful, application-centric model built on two core concepts:

1. Templates: The Reusable Blueprint

A **Template** in Casibase is a complete, reusable blueprint for an application. It encapsulates all the necessary Kubernetes resource manifests required to deploy a service. Think of it as a "package" for a cloud application.

- **What it contains**: A template holds the base YAML configurations for all the components of an application (Deployments, Services, etc.), structured for use with Kustomize.
- **The Goal**: To make the underlying complexity transparent. Once a template for WordPress is created, anyone can use it to deploy WordPress without needing to know the details of its Kubernetes architecture.

2. Applications: The Live Instance

An **Application** is a live, running instance created from a Template. It represents a specific deployment of that template in your cluster.

- **Customization**: When you create an Application, you select a Template and then provide your own specific configurations, such as the number of replicas, a custom domain name, or a specific database password.
- **How it works**: These custom configurations are treated as "patches" or "overlays." Casibase uses Kustomize to intelligently merge your custom parameters with the base manifests from the template, generating the final, complete configuration.
- **Lifecycle Management**: The Application becomes the single unit you interact with. You can deploy, monitor, update, and delete the entire application with single clicks in the UI.

By adopting this model, Casibase transforms the complex task of cloud-native deployment into a streamlined, intuitive workflow. Instead of wrestling with `kubectl` and YAML files, you can manage the entire lifecycle of your applications through a clean web interface: **select a template, fill in a few parameters, and click deploy**.

Beginner Guide

📄 Add a Storage Provider

Discover how to integrate a storage provider with Casibase.

📄 Add an AI Model Provider

Learn how to add a model provider to enhance Casibase functionality.

📄 Add an Embedding Provider

Explore how to integrate an embedding provider with Casibase.

📄 Add a Text-to-Speech Model Provider

Learn how to add a text-to-speech provider to enhance Casibase functionality.

Add a Speech-to-Text Provider

Learn how to add a speech-to-text provider to enhance Casibase functionality.

Add a Store

Learn how to add a store to your Casibase knowledge base system.

Chats with AI

Implement AI chat functionality in your Casibase knowledge base system.

Add a Storage Provider

This document is a step-by-step tutorial designed for beginners. It will guide you through the process of integrating a storage provider with Casibase, our powerful knowledge base system.

Introduction

Adding a storage provider to Casibase enables you to efficiently manage and store data, making it an essential component for your knowledge base system.

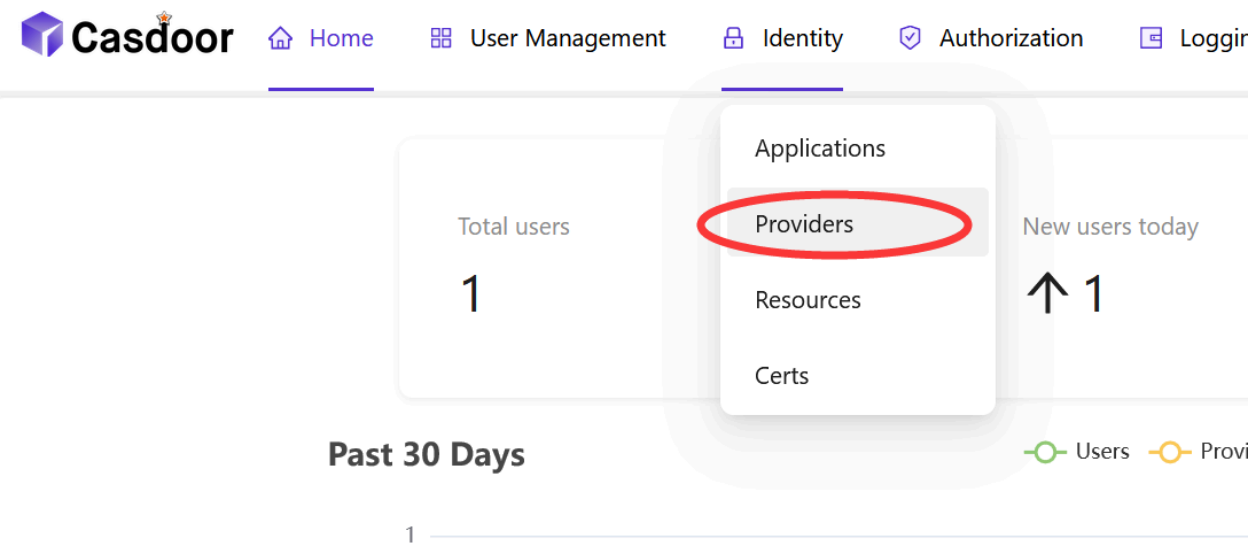
If you're new to integrating storage providers, don't worry. We've broken down the process into simple steps that anyone can follow.

Step 1: Deploy Casdoor and Casibase

If you haven't done, please refer to the Deploy Casdoor and Casibase tutorial.

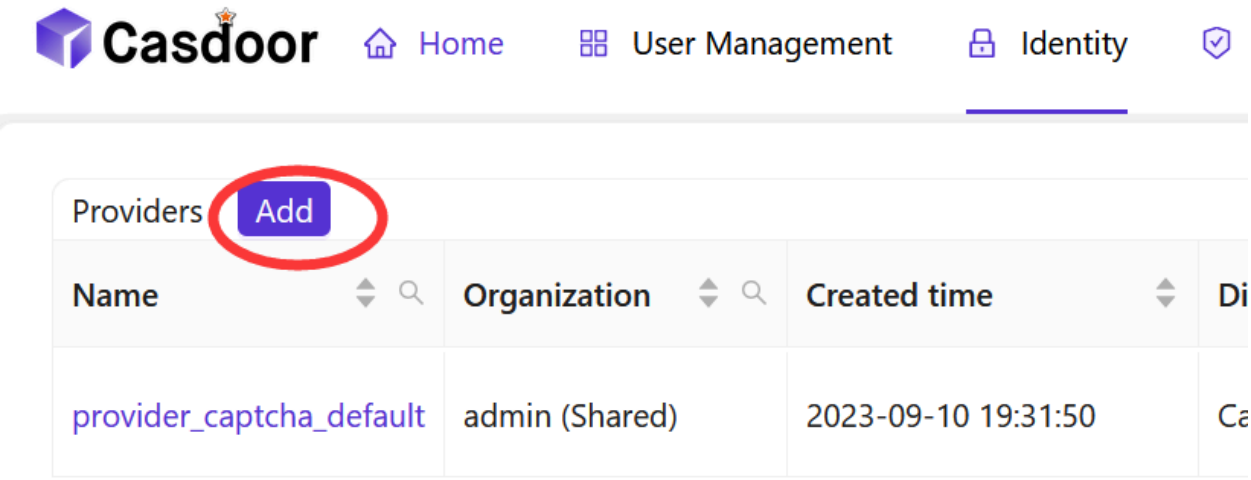
Step 2: Add a New Storage Provider

Storage providers are used to store data. They can be added in Casdoor by clicking the `Identity - Providers` button on the home page.



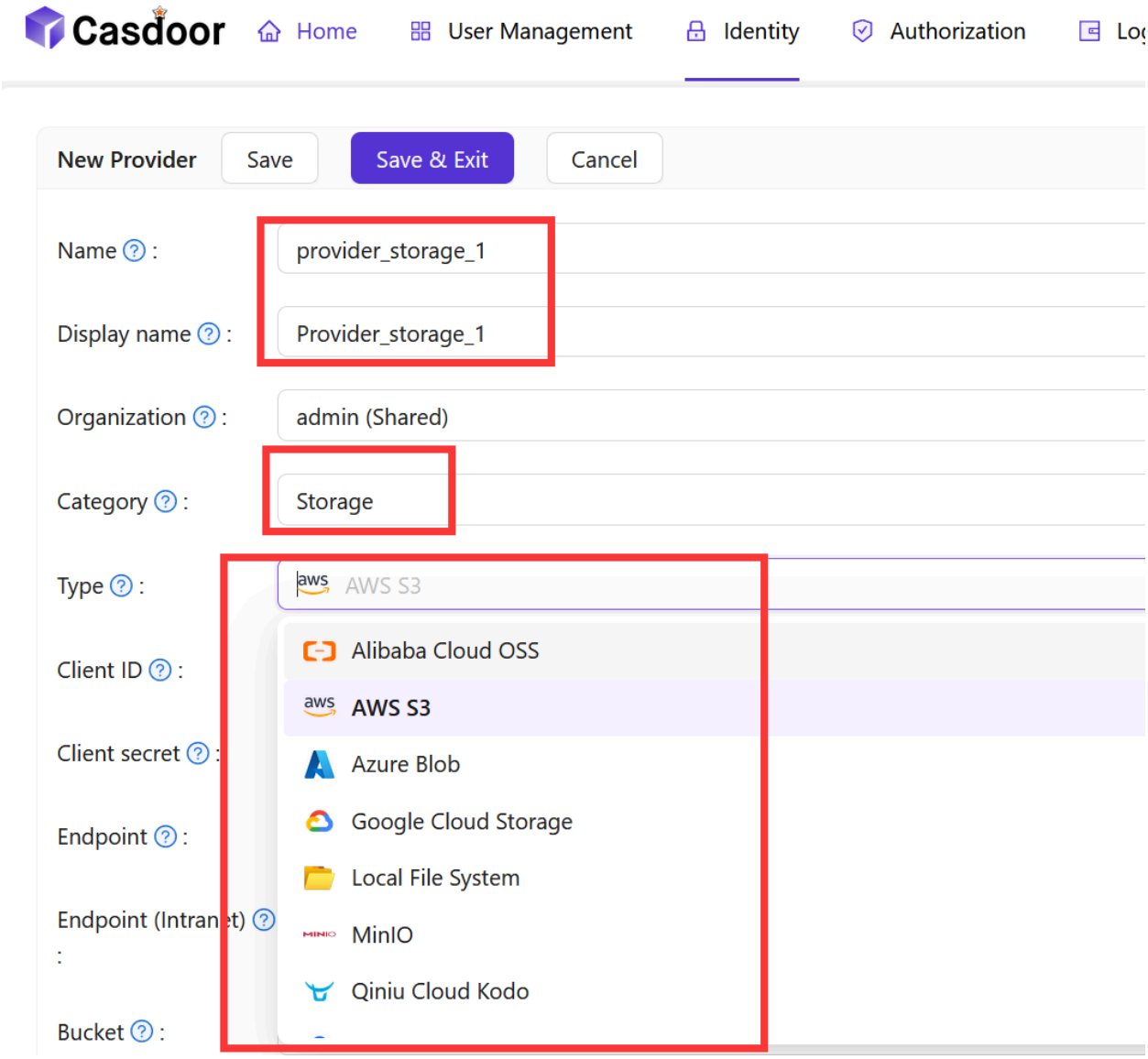
Step 2.1: Add a storage provider

Click the `Add` button to add a storage provider.



Step 2.2: Fill in the storage provider information

Fill in the storage provider information and click the `Save & Exit` button.



TIP

Casdoor supports many storage providers, including:

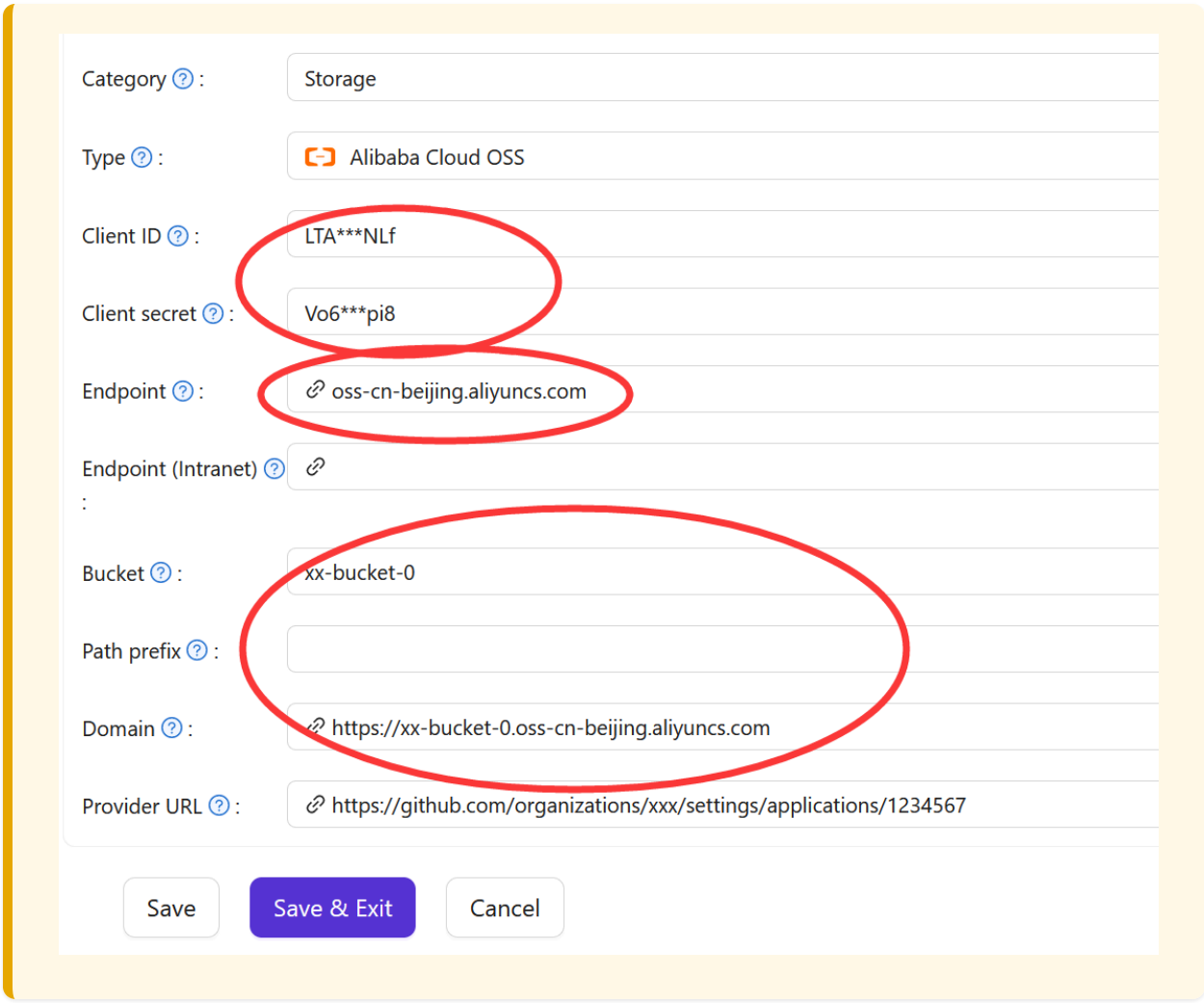
- AWS S3
- Azure Blob
- Google Cloud Storage
- MinIO

- Qiniu Cloud Kodo
- Alibaba Cloud OSS ...

Example

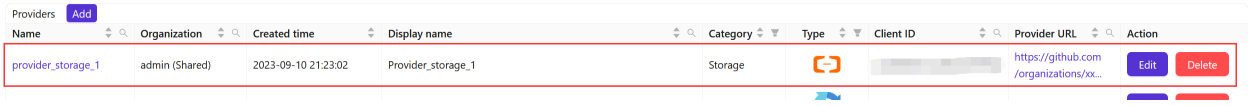
Add an Aliyun OSS storage provider

> ⚠ CAUTION
>
> - Client ID: The AccessKey ID of your Aliyun OSS account.
> - Client Secret: The AccessKey Secret of your Aliyun OSS account.
>
> `****` is the placeholder for your Aliyun OSS account information.



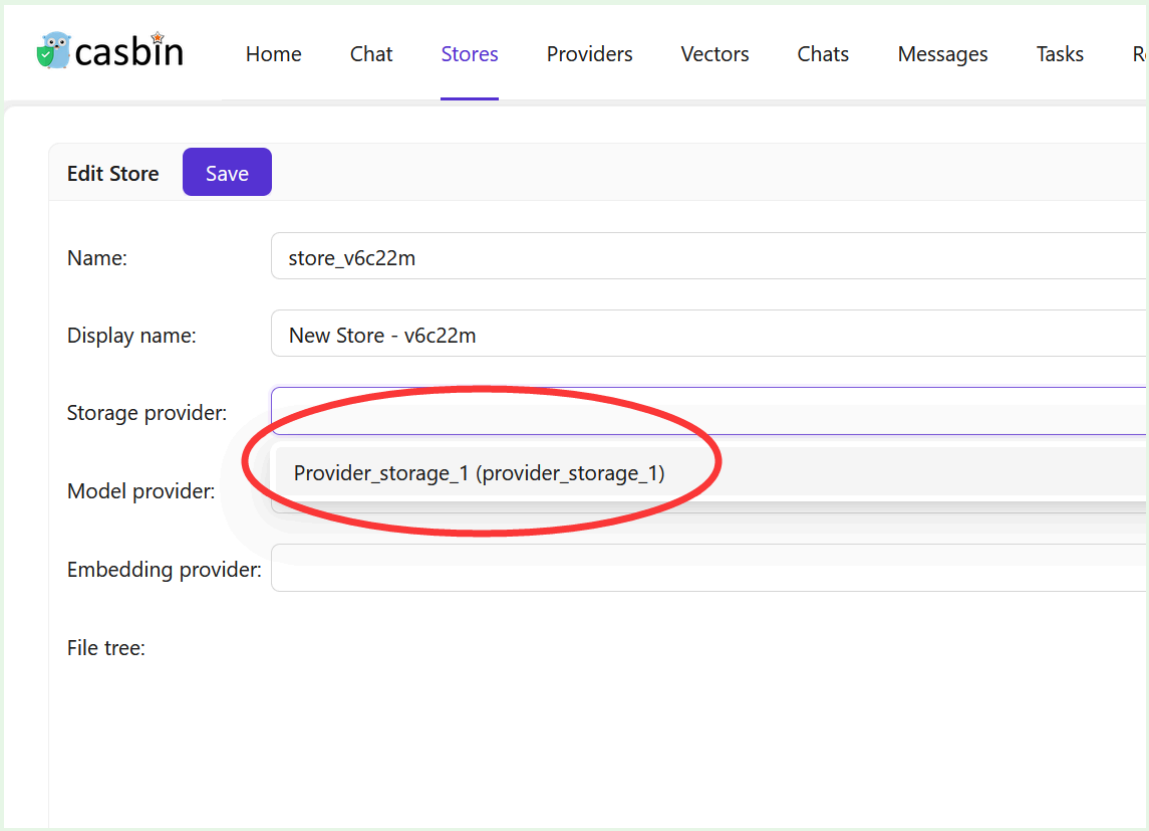
Step 2.3: View the storage provider

After adding the storage provider, you can view the storage provider information.

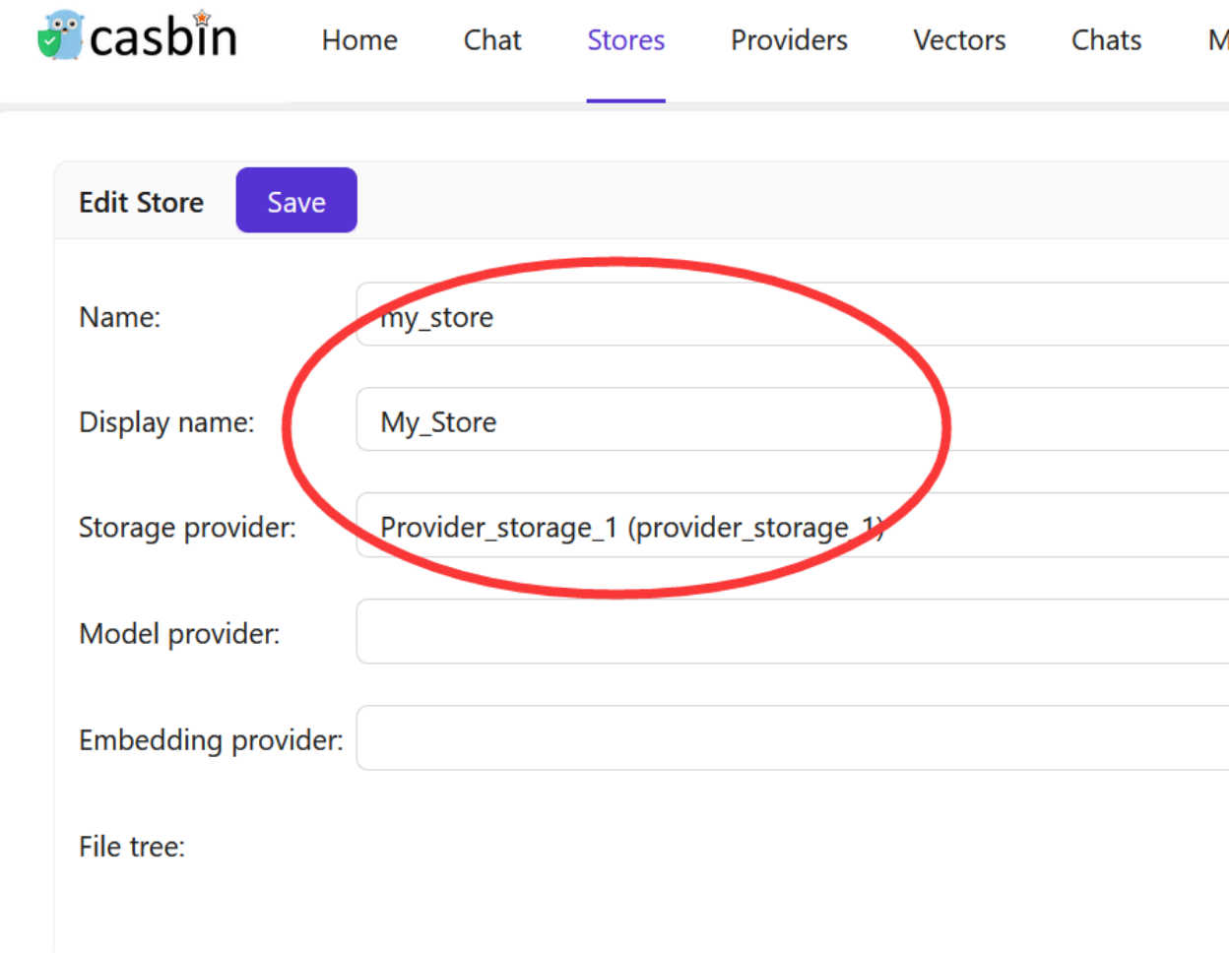


> **TIP**
>
> Storage providers come from Casdoor. You can add a storage provider in Casdoor and then add it to Casibase.

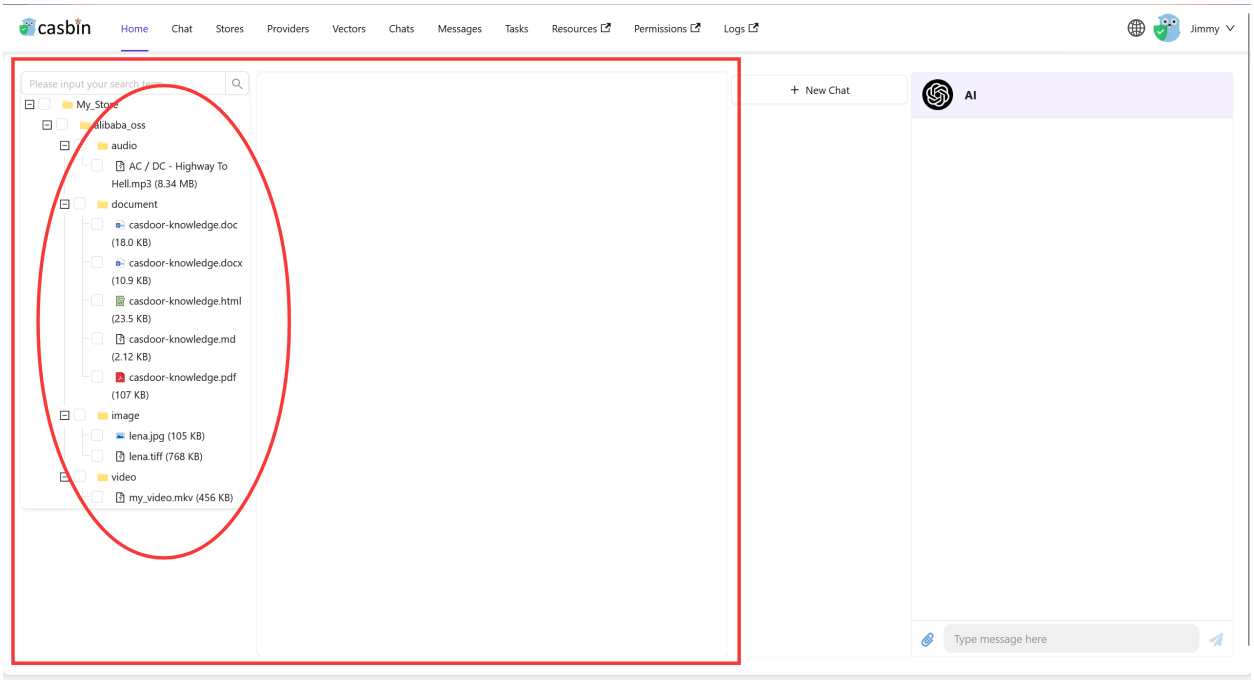
Refer to Step 2: Add a New Storage Provider for more information.



Store Example



Save the configuration, return to the home page, and you'll see the file-tree of the storage provider.



Now you can manage your data in Casibase.

In the next chapter, we'll learn how to add an `AI model provider` to Casibase.

Add an AI Model Provider

This document is a step-by-step tutorial designed for beginners. It will guide you through the process of integrating a model provider with Casibase, our powerful knowledge base system.

Introduction

Adding a model provider to Casibase enables you to enhance its functionality by incorporating machine learning models and AI capabilities. Model providers allow you to analyze and process data within your knowledge base system, making it more intelligent and efficient.

If you're new to integrating model providers, don't worry. We've broken down the process into simple steps that anyone can follow.

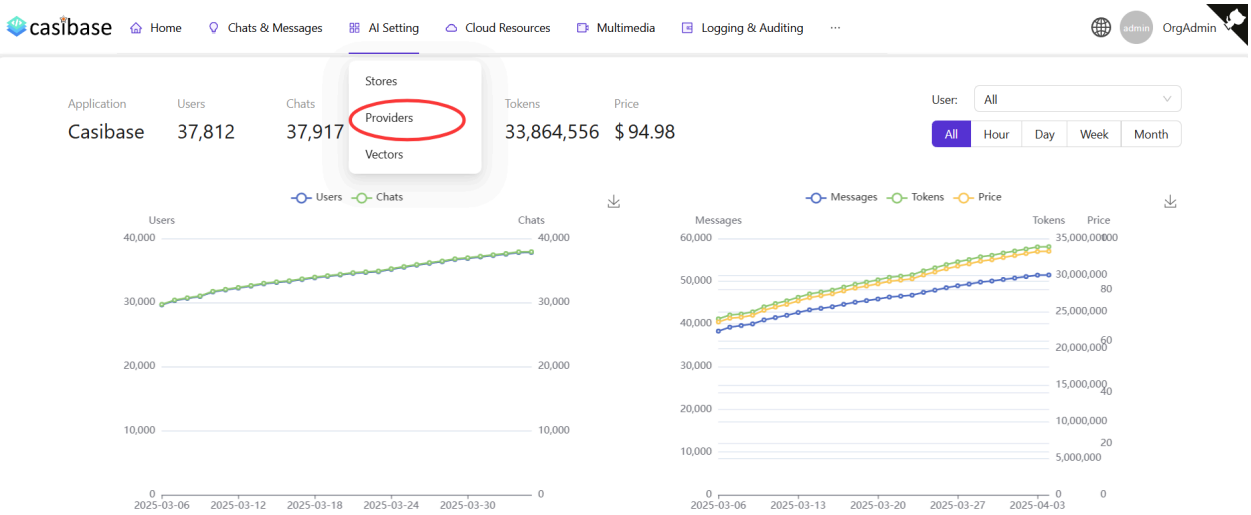
Step 1: Deploy Casdoor and Casibase

Before you can add an AI model provider, make sure you have Casdoor and Casibase deployed. If you haven't done, please refer to the Deploy Casdoor and Casibase tutorial.

Step 2: Add a New Model Provider

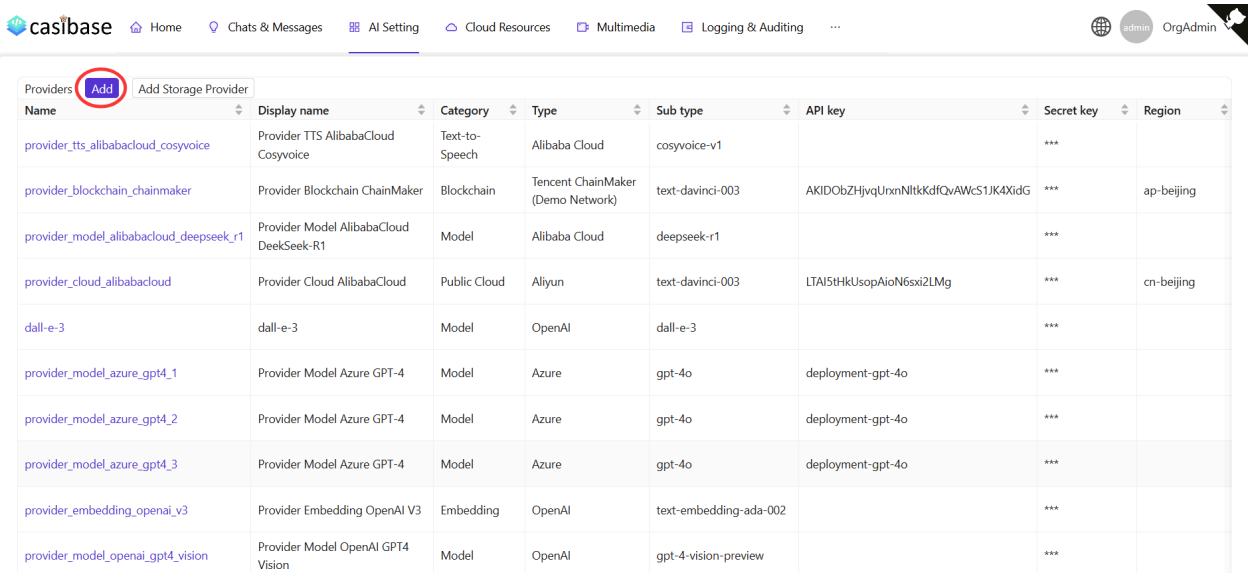
Model providers are used to integrate LLM into Casibase. You can add them by following these steps:

Click the `Providers` button on the home page.



Step 2.1: Add a Model Provider

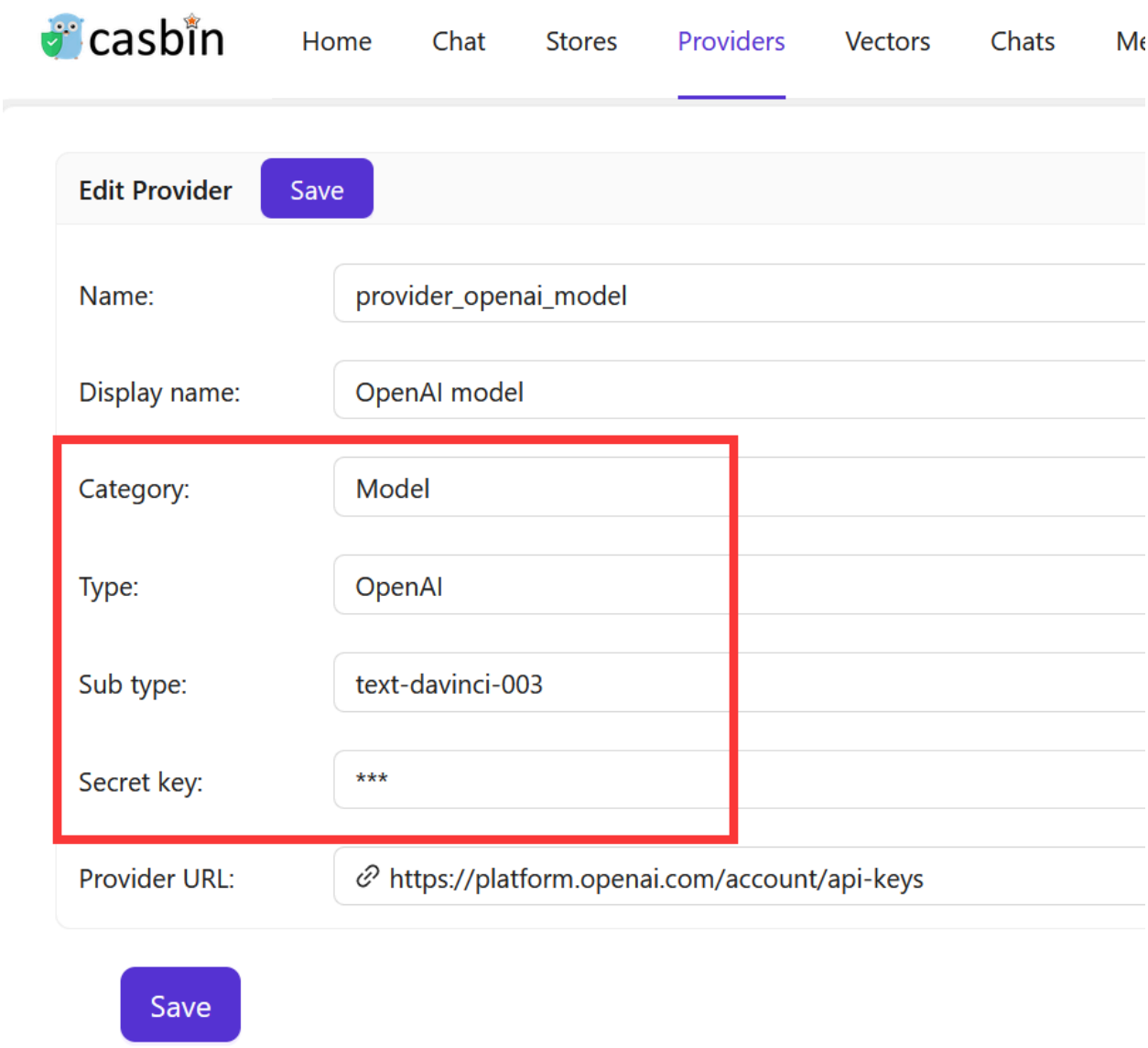
Click the `Add` button to add a model provider.



Name	Display name	Category	Type	Sub type	API key	Secret key	Region
provider_tts_alibabacloud_cosyvoice	Provider TTS AlibabaCloud Cosyvoice	Text-to-Speech	Alibaba Cloud	cosyvoice-v1		***	
provider_blockchain_chainmaker	Provider Blockchain ChainMaker	Blockchain	Tencent ChainMaker (Demo Network)	text-davinci-003	AKIDObZHjvqUrxnNltkKdfQvAWcS1JK4XidG	***	ap-beijing
provider_model_alibabacloud_deepseek_r1	Provider Model AlibabaCloud DeekSeek-R1	Model	Alibaba Cloud	deepseek-r1		***	
provider_cloud_alibabacloud	Provider Cloud AlibabaCloud	Public Cloud	Aliyun	text-davinci-003	LTAI5tHkUsopAioN6sxi2LMg	***	cn-beijing
dall-e-3	dall-e-3	Model	OpenAI	dall-e-3		***	
provider_model_azure_gpt4_1	Provider Model Azure GPT-4	Model	Azure	gpt-4o	deployment-gpt-4o	***	
provider_model_azure_gpt4_2	Provider Model Azure GPT-4	Model	Azure	gpt-4o	deployment-gpt-4o	***	
provider_model_azure_gpt4_3	Provider Model Azure GPT-4	Model	Azure	gpt-4o	deployment-gpt-4o	***	
provider_embedding_openai_v3	Provider Embedding OpenAI V3	Embedding	OpenAI	text-embedding-ada-002		***	
provider_model_openai_gpt4_vision	Provider Model OpenAI GPT4 Vision	Model	OpenAI	gpt-4-vision-preview		***	

Step 2.2: Fill in Model Provider Details

Fill in the model provider details and click the `Save & Exit` button.



TIP

Casibase supports many model providers, including:

- Hugging Face
  - meta-llama/Llama-2-7b

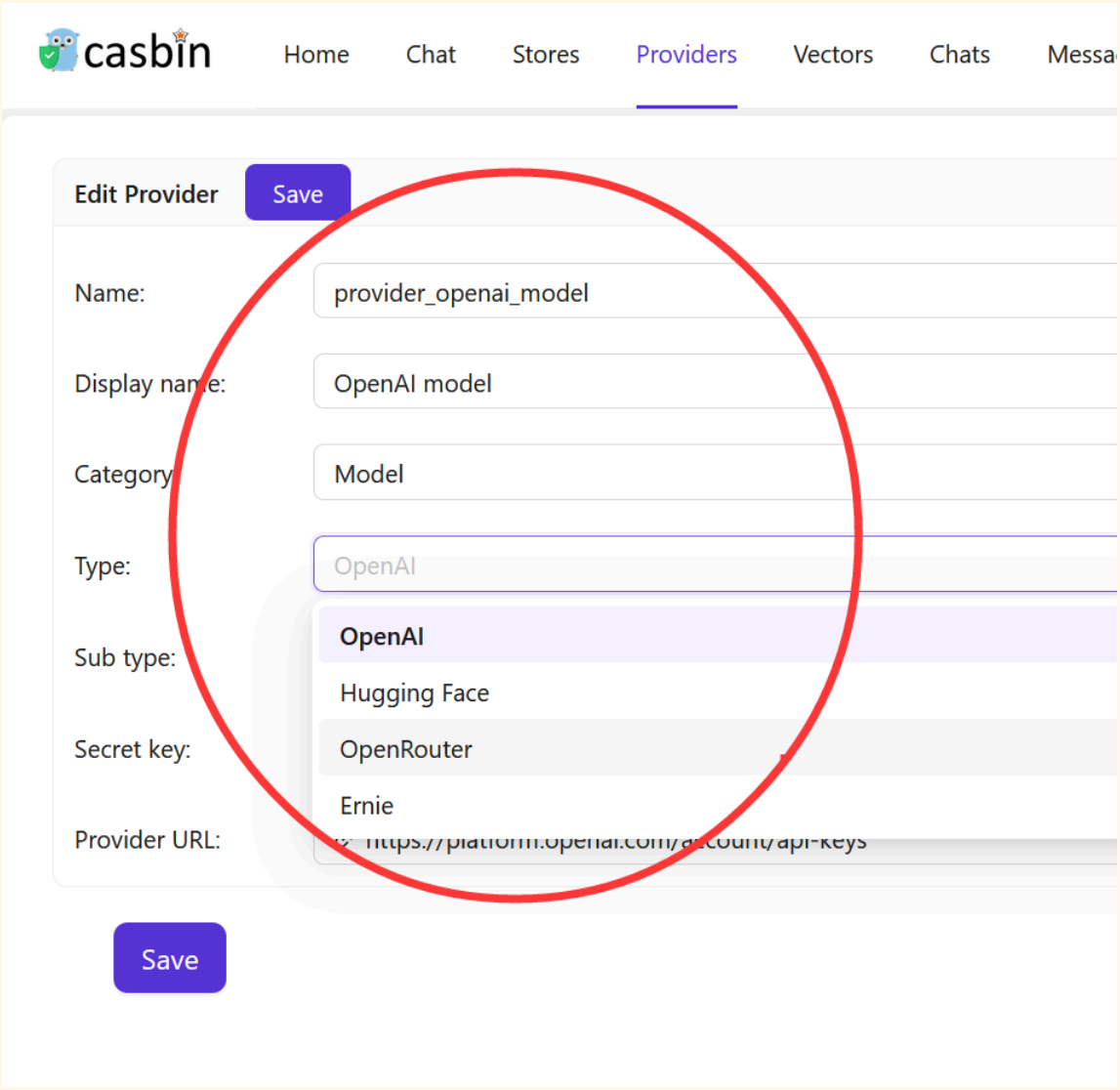
    - THUDM/chatglm2-6b
    - baichuan-inc/Baichuan2-13B-chat
    - gpt2
    - ……
- OpenRouter
    - anthropic/claude-2
    - palm-2-chat-bison
    - palm-2-codechat-bison
    - openai/gpt-4
    - ……
- OpenAI
    - text-davinci-003
    - gpt-3.5-turbo
    - gpt-4
    - ……

⚠ CAUTION

- Category: The first-level category of the model provider. For example, `Model` and `Embedding`.
- Type: The second-level category of the model provider. For example, `OpenAI` and `Hugging Face`.
- SecretKey: The secret key of your OpenAI account.

Example

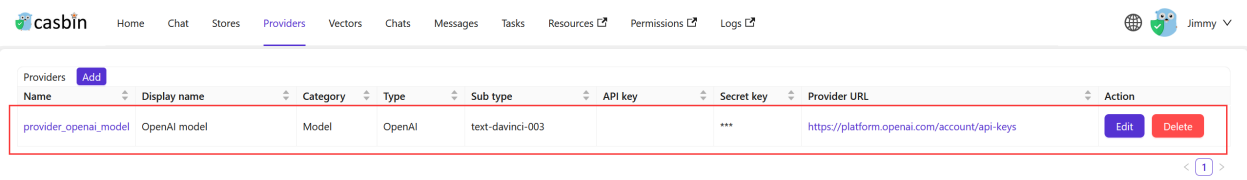
Add an OpenAI model provider



> ⚠ CAUTION
>
> Some models don't support **streaming-output**. Known models that support streaming-output include:
>
> - `gpt-3.5-turbo-0613`

After adding a model provider, you can use it to analyze and process data in Casibase using chatbots, question answering, and other AI capabilities.

Return to the model provider list page:



Now that you've added a model provider, you can use it to analyze and process data in Casibase using chatbots, question answering, and other AI capabilities.

In the next chapter, we'll learn how to add an embedding provider to Casibase.

Add an Embedding Provider

This document is a step-by-step tutorial designed for beginners. It will guide you through the process of integrating an embedding provider with Casibase, our powerful knowledge base system.

Introduction

Embedding is a technique used to represent words and documents as vectors. Embedding providers allow you to analyze and process data within your knowledge base system, making it more intelligent and efficient.

Refer to the Core Concepts section of our previous documentation for more information about embedding.

In Casibase, you can add an embedding provider by following these steps:

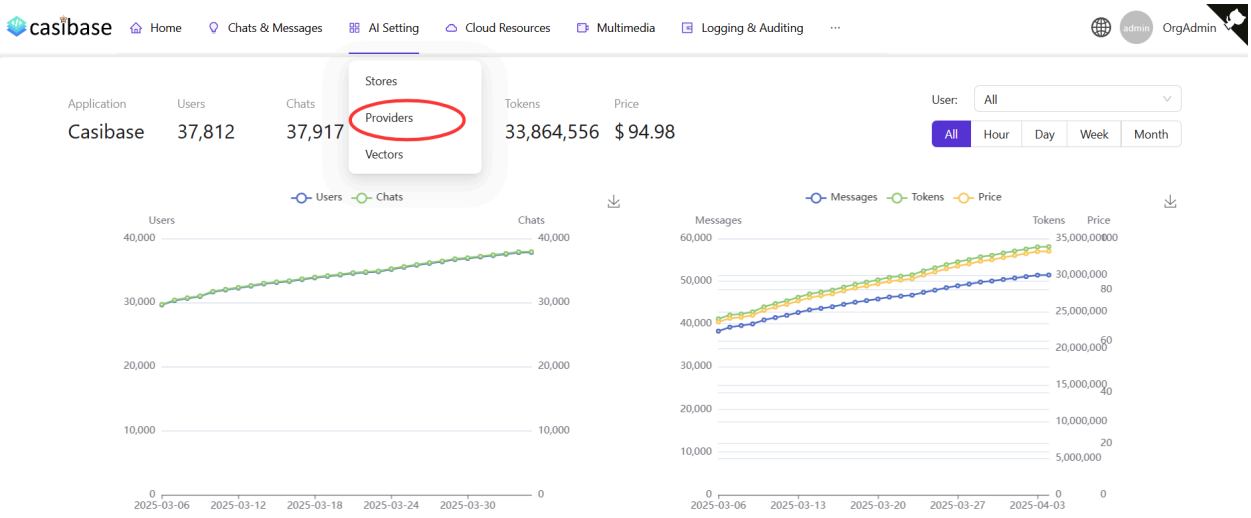
Step 1: Deploy Casdoor and Casibase

Before you can add an embedding model provider, make sure you have Casdoor and Casibase deployed. If you haven't done, please refer to the Deploy Casdoor and Casibase tutorial.

Step 2: Add a New Embedding Provider

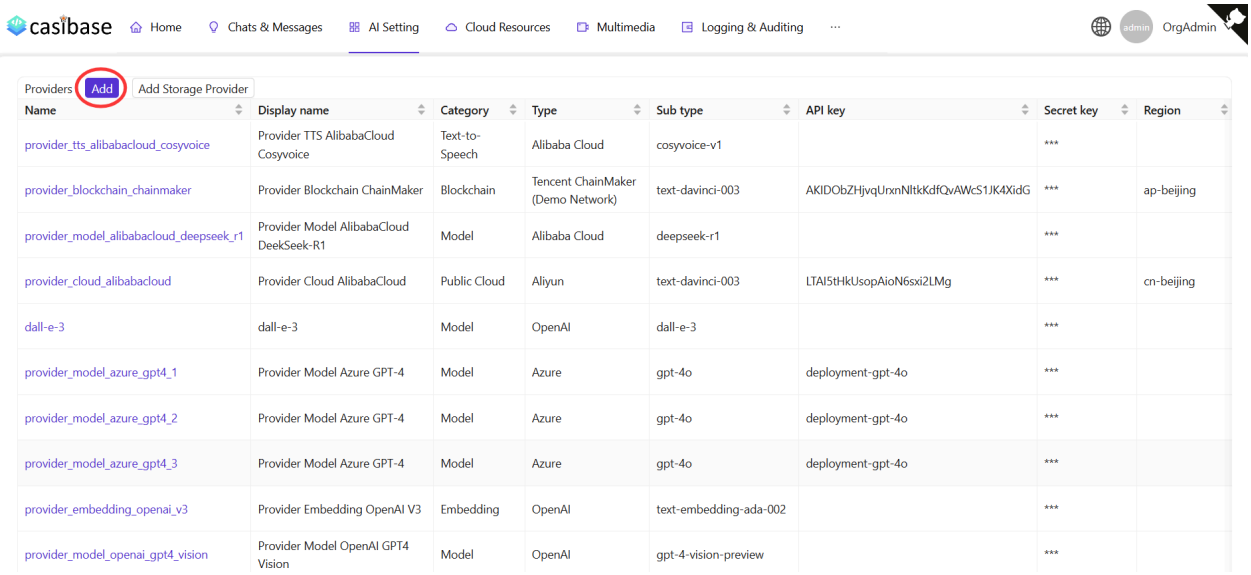
Embedding providers are used to integrate embedding into Casibase. You can add them by following these steps:

Click the `Providers` button on the home page.



Step 2.1: Add an Embedding Provider

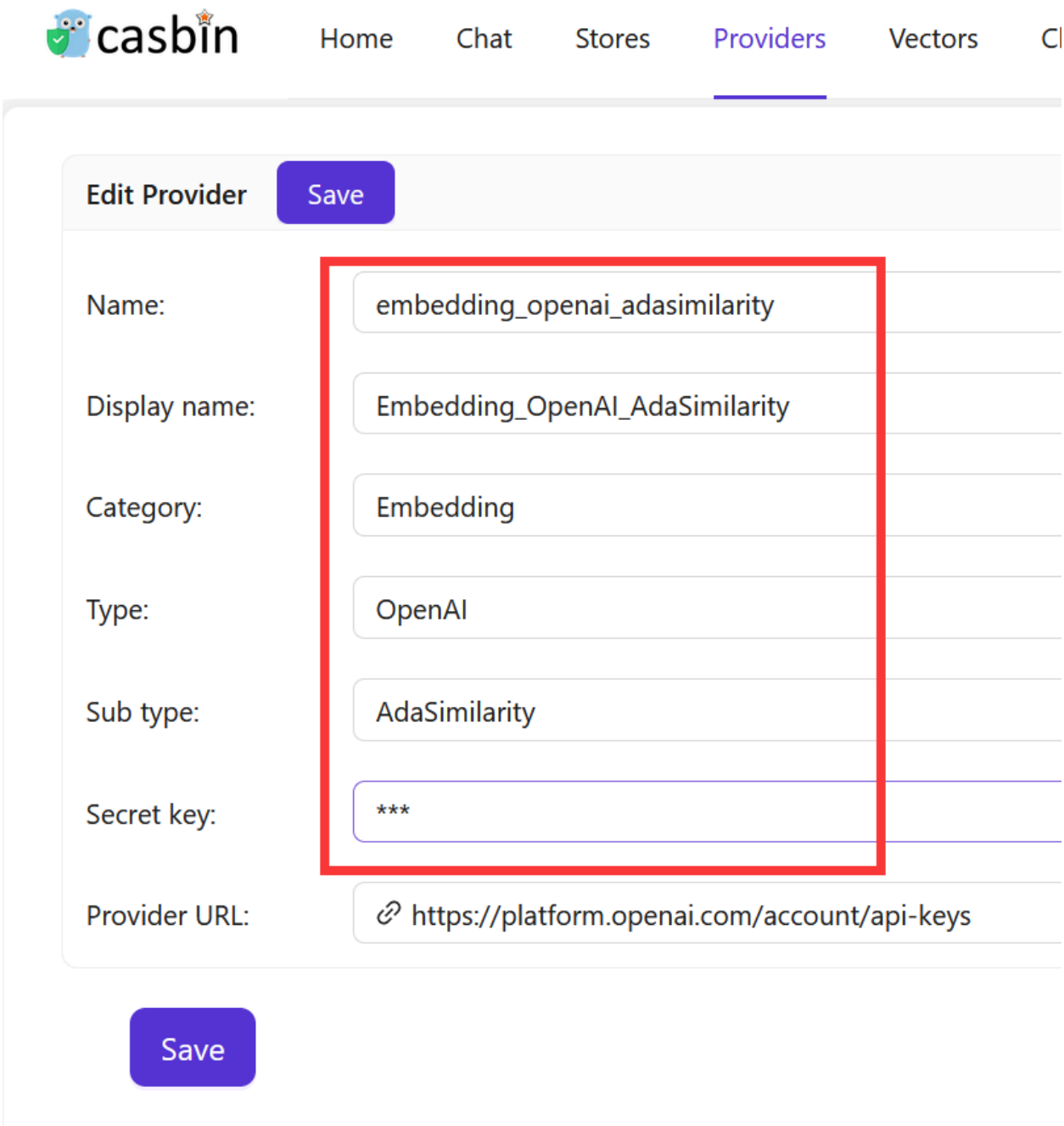
Click the `Add` button to add an embedding provider.



Name	Display name	Category	Type	Sub type	API key	Secret key	Region
provider_tts_alibabacloud_cosyvoice	Provider TTS AlibabaCloud Cosyvoice	Text-to-Speech	Alibaba Cloud	cosyvoice-v1		***	
provider_blockchain_chainmaker	Provider Blockchain ChainMaker	Blockchain	Tencent ChainMaker (Demo Network)	text-davinci-003	AKIDObZHjvqUrxnNltkKdfQvAWcS1JK4XidG	***	ap-beijing
provider_model_alibabacloud_deepseek_r1	Provider Model AlibabaCloud DeekSeek-R1	Model	Alibaba Cloud	deepseek-r1		***	
provider_cloud_alibabacloud	Provider Cloud AlibabaCloud	Public Cloud	Aliyun	text-davinci-003	LTAI5tHkUsopAioN6sxi2LMg	***	cn-beijing
dall-e-3	dall-e-3	Model	OpenAI	dall-e-3		***	
provider_model_azure_gpt4_1	Provider Model Azure GPT-4	Model	Azure	gpt-4o	deployment-gpt-4o	***	
provider_model_azure_gpt4_2	Provider Model Azure GPT-4	Model	Azure	gpt-4o	deployment-gpt-4o	***	
provider_model_azure_gpt4_3	Provider Model Azure GPT-4	Model	Azure	gpt-4o	deployment-gpt-4o	***	
provider_embedding_openai_v3	Provider Embedding OpenAI V3	Embedding	OpenAI	text-embedding-ada-002		***	
provider_model_openai_gpt4_vision	Provider Model OpenAI GPT4 Vision	Model	OpenAI	gpt-4-vision-preview		***	

Step 2.2: Fill in Embedding Provider Details

Fill in the embedding provider details and click the `Save & Exit` button.

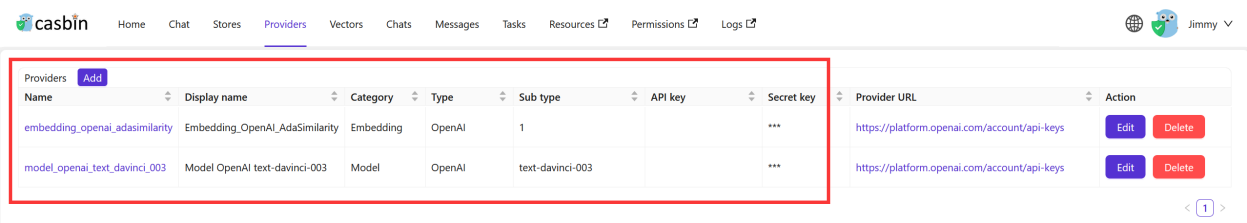


TIP

Same as the Model Provider section, Casibase supports many embedding providers, including:

- OpenAI
  - AdaSimilarity
  - DavinciSimilarity
  - AdaEmbedding2
  - ......
- Hugging Face
  - sentence-transformers/paraphrase-MiniLM-L6-v2
  - ......

Return providers list page:



Now, you can use the embedding provider to convert text to vectors.

After adding an embedding provider, you can use it to retrieve similar documents in Casibase. For more information, please refer to the Core Concepts section of our previous documentation.

In the next chapter, we will learn how to integrate storage providers, model providers, and embedding providers with Casibase.

Add a Text-to-Speech Model Provider

This document is a step-by-step tutorial designed for beginners. It will guide you through the process of integrating a text-to-speech provider with Casibase, our powerful knowledge base system.

Introduction

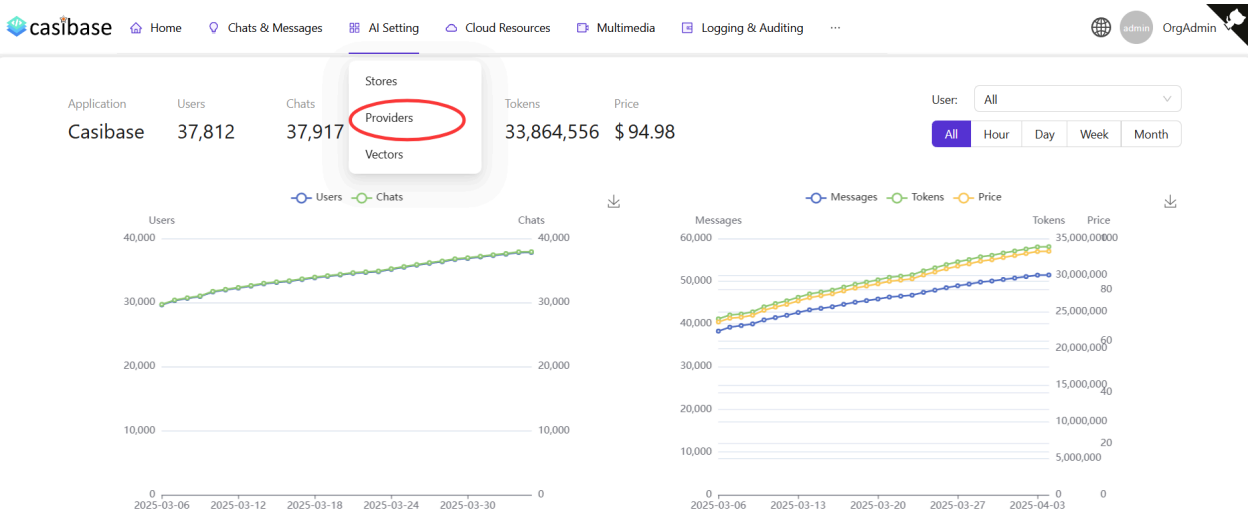
Text-to-Speech (TTS) is a technology that converts text into spoken voice output. TTS providers allow your Casibase applications to communicate with users through synthesized speech, enhancing the user experience and accessibility of your knowledge base system.

In Casibase, integrating a TTS provider enables your AI applications to verbally respond to queries, creating more interactive and engaging user experiences.

Add a New Text-to-Speech Provider

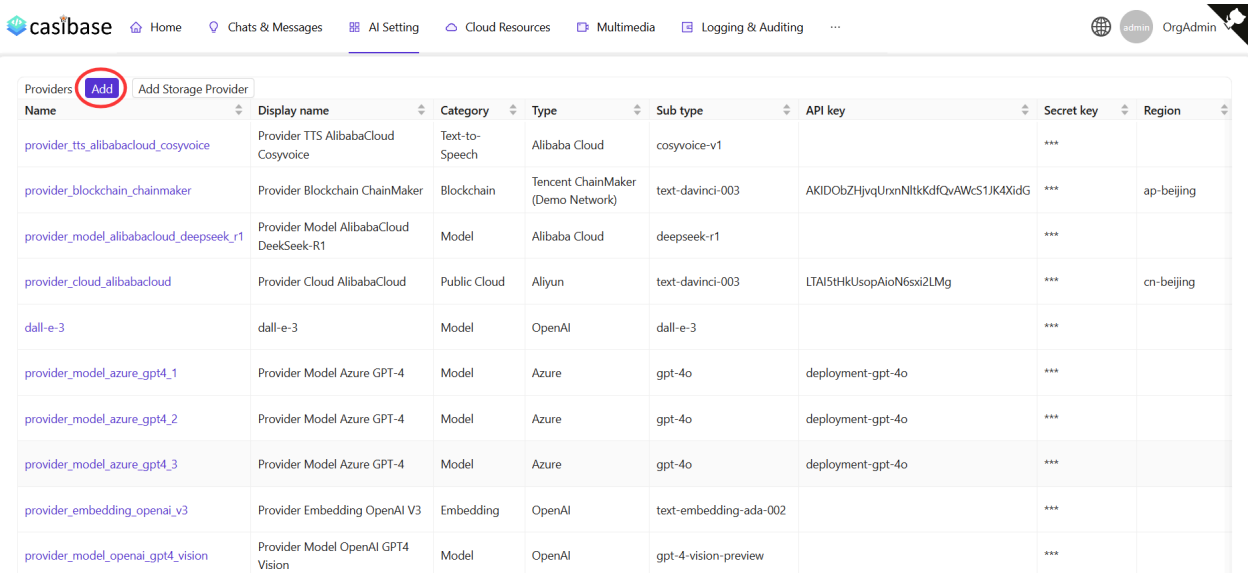
Text-to-Speech providers are used to integrate voice synthesis capabilities into Casibase. You can add them by following these steps:

Click the `Providers` button on the page.



Add a Text-to-Speech Provider

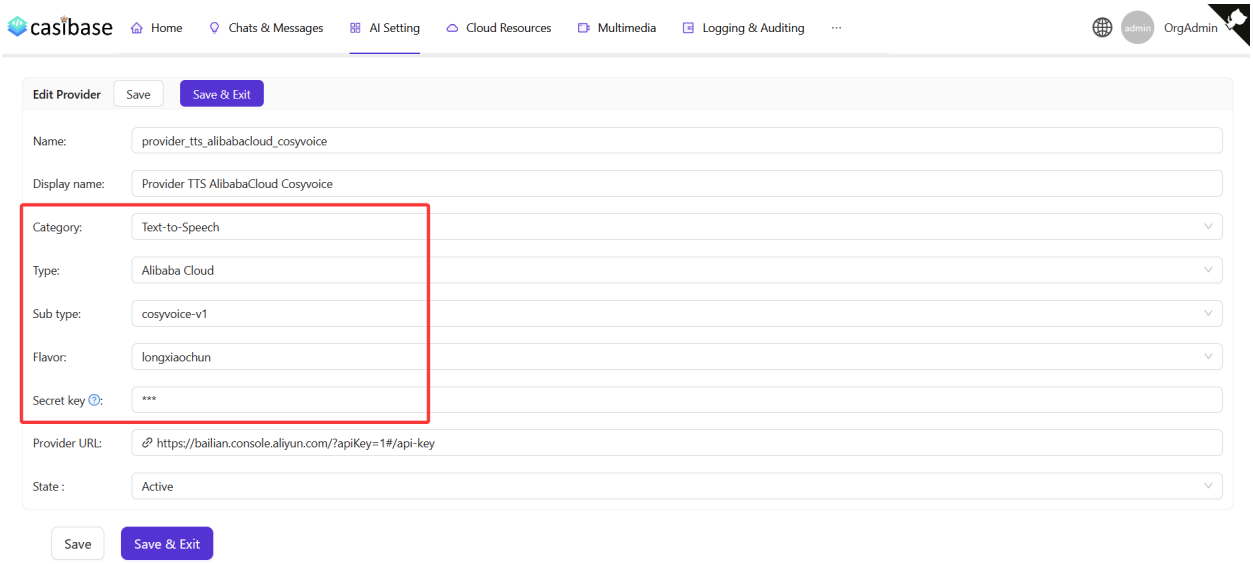
Click the `Add` button to add a Text-to-Speech provider.



Name	Display name	Category	Type	Sub type	API key	Secret key	Region
provider_tts_alibabacloud_cosyvoice	Provider TTS AlibabaCloud Cosyvoice	Text-to-Speech	Alibaba Cloud	cosyvoice-v1		***	
provider_blockchain_chainmaker	Provider Blockchain ChainMaker	Blockchain	Tencent ChainMaker (Demo Network)	text-davinci-003	AKIDObZHjvqUrxnNltkKdfQvAWcS1JK4XidG	***	ap-beijing
provider_model_alibabacloud_deepseek_r1	Provider Model AlibabaCloud DeekSeek-R1	Model	Alibaba Cloud	deepseek-r1		***	
provider_cloud_alibabacloud	Provider Cloud AlibabaCloud	Public Cloud	Aliyun	text-davinci-003	LTAI5tHkUsopAioN6sxi2LMg	***	cn-beijing
dall-e-3	dall-e-3	Model	OpenAI	dall-e-3		***	
provider_model_azure_gpt4_1	Provider Model Azure GPT-4	Model	Azure	gpt-4o	deployment-gpt-4o	***	
provider_model_azure_gpt4_2	Provider Model Azure GPT-4	Model	Azure	gpt-4o	deployment-gpt-4o	***	
provider_model_azure_gpt4_3	Provider Model Azure GPT-4	Model	Azure	gpt-4o	deployment-gpt-4o	***	
provider_embedding_openai_v3	Provider Embedding OpenAI V3	Embedding	OpenAI	text-embedding-ada-002		***	
provider_model_openai_gpt4_vision	Provider Model OpenAI GPT4 Vision	Model	OpenAI	gpt-4-vision-preview		***	

Fill in Text-to-Speech Provider Details

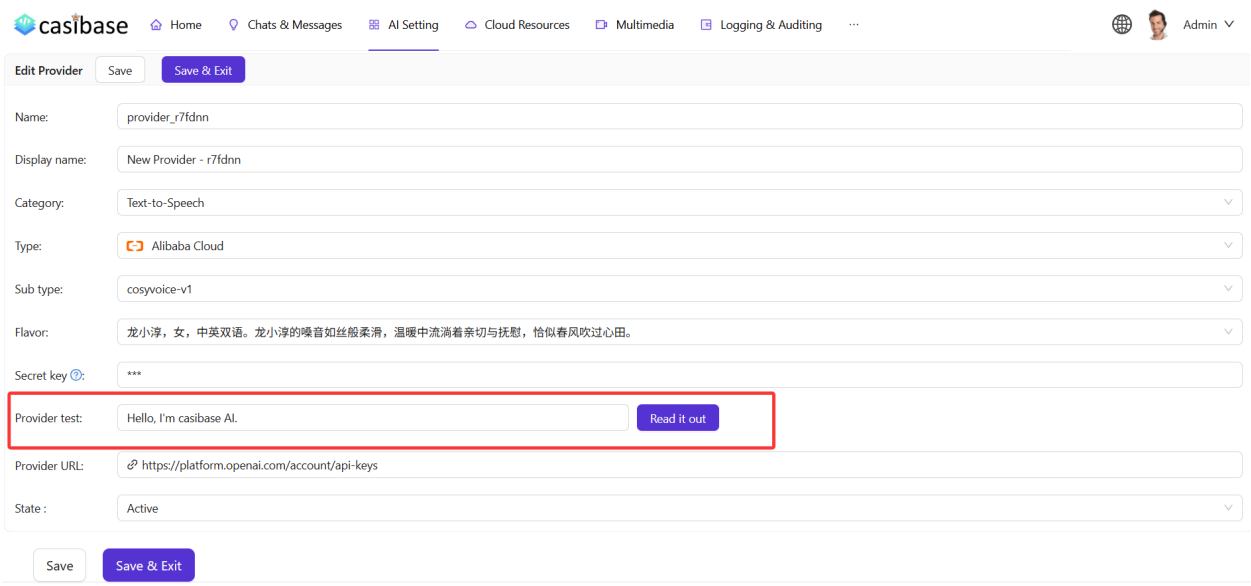
Fill in the embedding provider details and click the `Save & Exit` button.



> **TIP**
>
> Casibase currently supports the following Text-to-Speech provider:
>
> - Alibaba Cloud
>   - cosyvoice-v1 (with multiple voice options)

Testing Your Text-to-Speech Provider

You can test your TTS provider by clicking the `Read it out` button. This will allow you to enter text and hear the synthesized speech output.



This testing feature allows you to verify your TTS configuration before implementing it in your applications, ensuring the voice quality and settings meet your requirements.

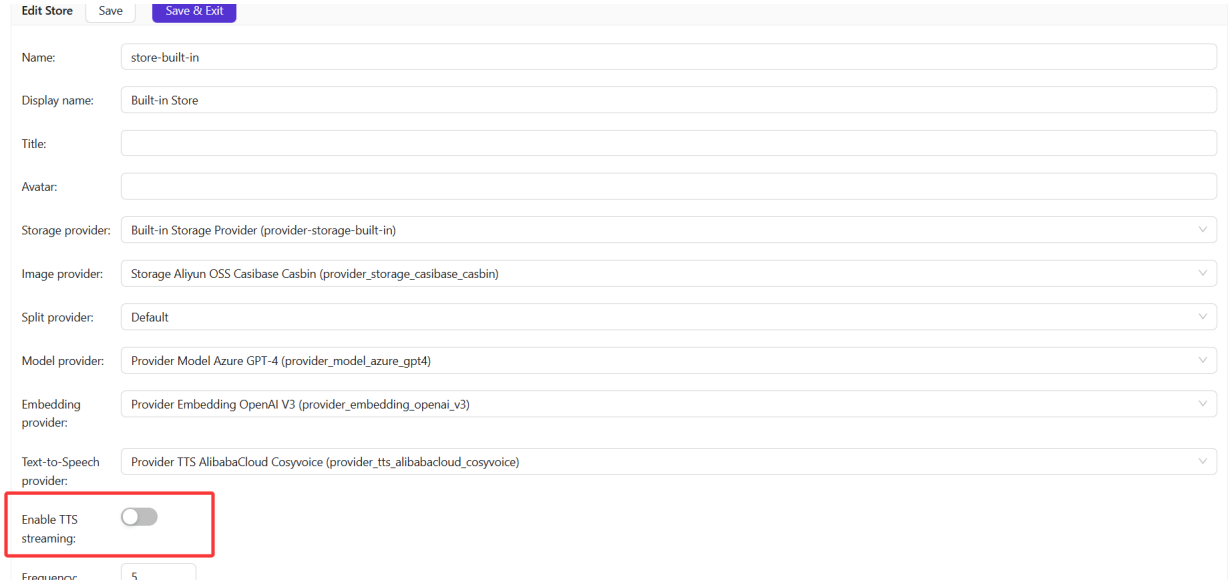
Voice Options for Alibaba Cloud

When using Alibaba Cloud's `cosyvoice-v1`, you can choose from various voice options:

- longwan
- longcheng
- ……

Using Text-to-Speech in Stores

After adding a Text-to-Speech provider, you can select this provider in your store settings and choose whether to enable TTS streaming.



Now, your store can convert text responses to speech, providing a more interactive experience for users.

Add a Speech-to-Text Provider

This document is a step-by-step tutorial designed for beginners. It will guide you through the process of integrating a speech-to-text provider with Casibase, our powerful knowledge base system.

Introduction

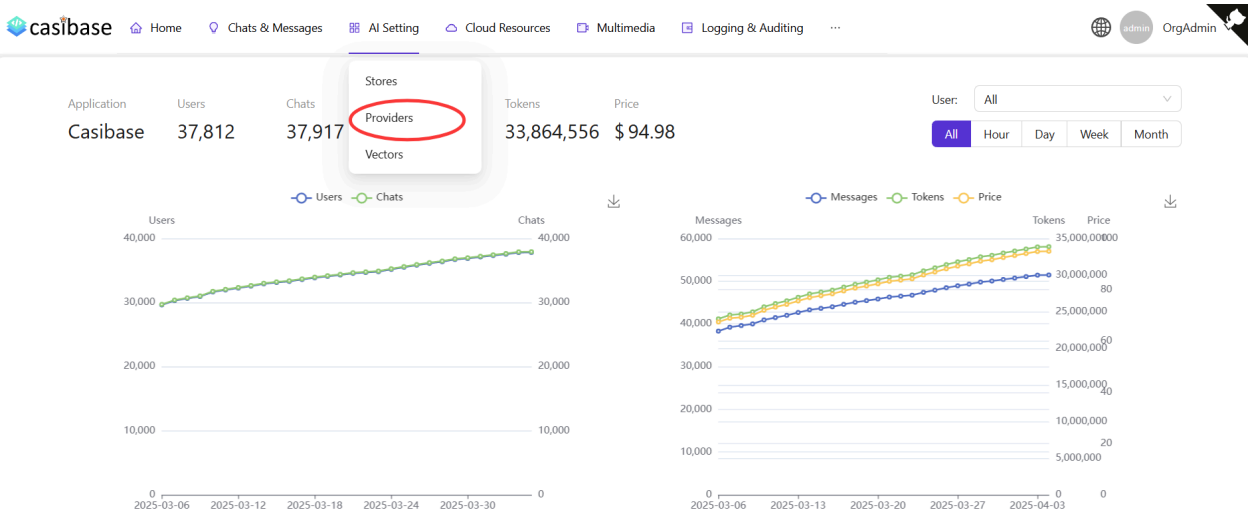
Speech-to-Text (STT) is a technology that converts spoken language into written text. STT providers allow your Casibase applications to understand and process spoken user input, enhancing the user experience and accessibility of your knowledge base system.

In Casibase, integrating an STT provider enables your AI applications to receive and process voice queries, creating more interactive and natural user interactions.

Add a New Speech-to-Text Provider

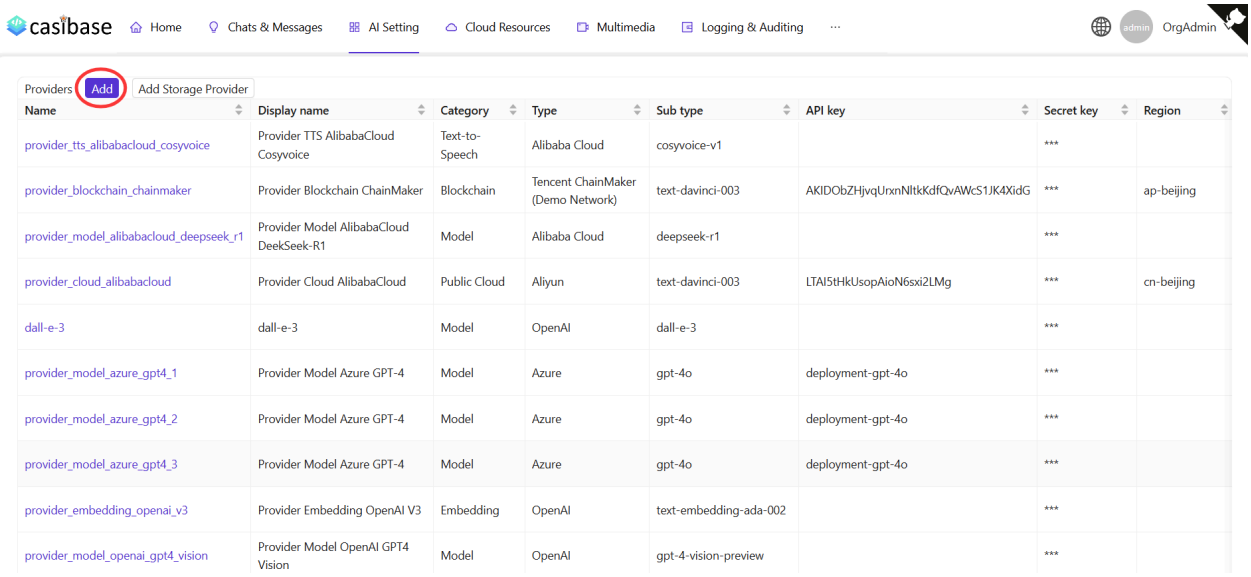
Speech-to-Text providers are used to integrate voice recognition capabilities into Casibase. You can add them by following these steps:

Click the `Providers` button on the page.



Add a Speech-to-Text Provider

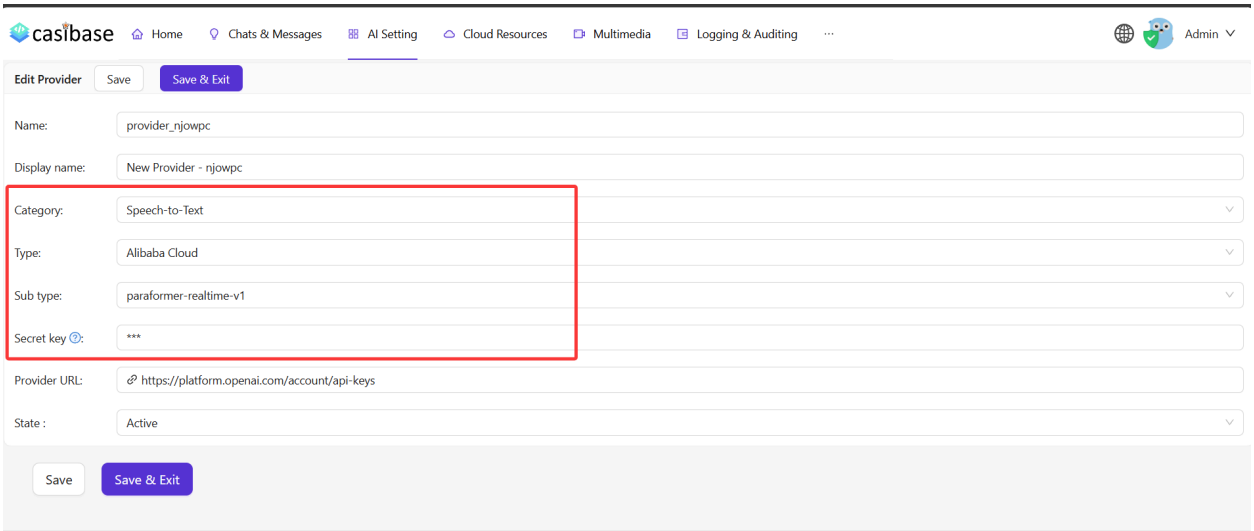
Click the `Add` button to add a Speech-to-Text provider.



Name	Display name	Category	Type	Sub type	API key	Secret key	Region
provider_tts_alibabacloud_cosyvoice	Provider TTS AlibabaCloud Cosyvoice	Text-to-Speech	Alibaba Cloud	cosyvoice-v1		***	
provider_blockchain_chainmaker	Provider Blockchain ChainMaker	Blockchain	Tencent ChainMaker (Demo Network)	text-davinci-003	AKIDObZHjvqUrxnNltkKdfQvAWcS1JK4XidG	***	ap-beijing
provider_model_alibabacloud_deepseek_r1	Provider Model AlibabaCloud DeekSeek-R1	Model	Alibaba Cloud	deepseek-r1		***	
provider_cloud_alibabacloud	Provider Cloud AlibabaCloud	Public Cloud	Aliyun	text-davinci-003	LTAI5tHkUsopAioN6sxi2LMg	***	cn-beijing
dall-e-3	dall-e-3	Model	OpenAI	dall-e-3		***	
provider_model_azure_gpt4_1	Provider Model Azure GPT-4	Model	Azure	gpt-4o	deployment-gpt-4o	***	
provider_model_azure_gpt4_2	Provider Model Azure GPT-4	Model	Azure	gpt-4o	deployment-gpt-4o	***	
provider_model_azure_gpt4_3	Provider Model Azure GPT-4	Model	Azure	gpt-4o	deployment-gpt-4o	***	
provider_embedding_openai_v3	Provider Embedding OpenAI V3	Embedding	OpenAI	text-embedding-ada-002		***	
provider_model_openai_gpt4_vision	Provider Model OpenAI GPT4 Vision	Model	OpenAI	gpt-4-vision-preview		***	

Fill in Speech-to-Text Provider Details

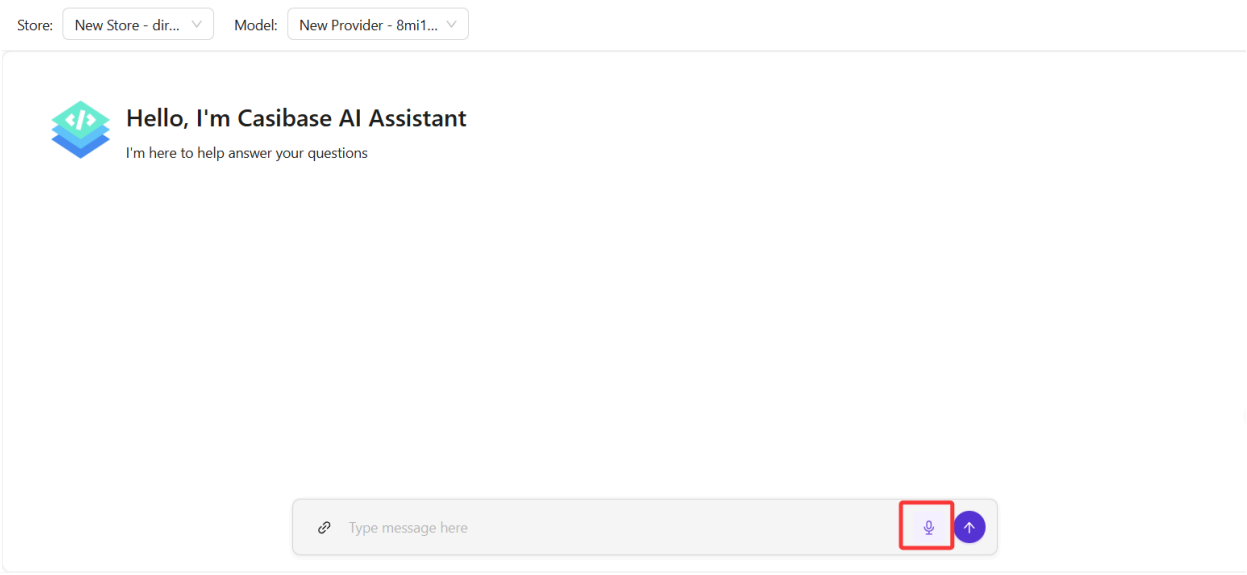
Fill in the speech-to-text provider details and click the `Save & Exit` button.



Using Voice Recognition

When you click the voice recognition button in your Casibase application, the following process occurs:

1. The browser will request permission to access your microphone
2. Once granted, the system will begin listening and automatically convert your speech to text
3. After you finish speaking, the recognized text will be automatically sent as a message



This feature enables hands-free interaction with your Casibase applications, making them more accessible and convenient to use.

> TIP
>
> Casibase currently supports the following Speech-to-Text provider:
>
> - Alibaba Cloud
>   - paraformer-realtime-v1

Add a Store

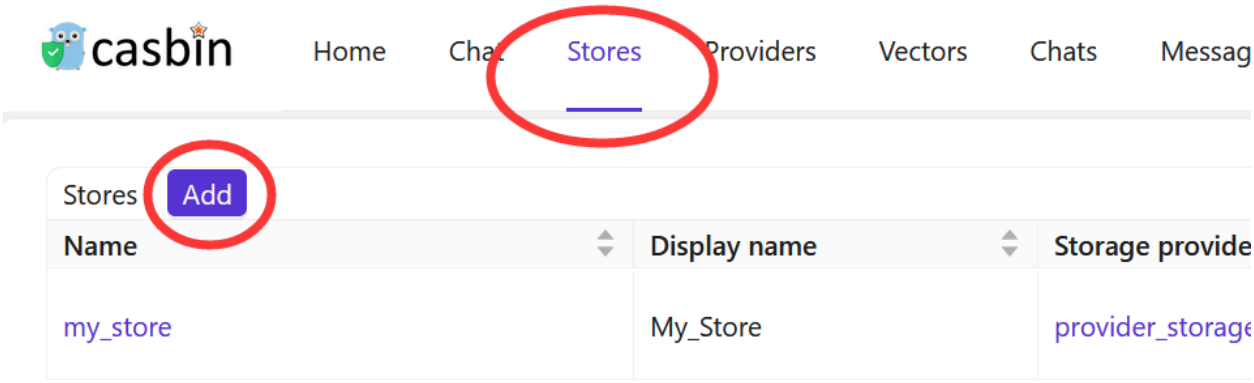
We have added a storage provider, a model provider, and an embedding provider. Now we need to configure a store to use these providers.

> ⚠ CAUTION
>
> This guide assumes that you have already deployed a Casibase knowledge base system. If you haven't done, please refer to the Deploy Casdoor and Casibase tutorial.
>
> Besides, this guide assumes that you have already added a storage provider, a model provider, and an embedding provider. If you have not, please follow the Add a Storage Provider, Add a AI Model Provider, and Add an Embedding Provider guides.

Step 1: Add a New Store

Stores are used to integrate storage, model, and embedding providers into Casibase. You can add them by following these steps:

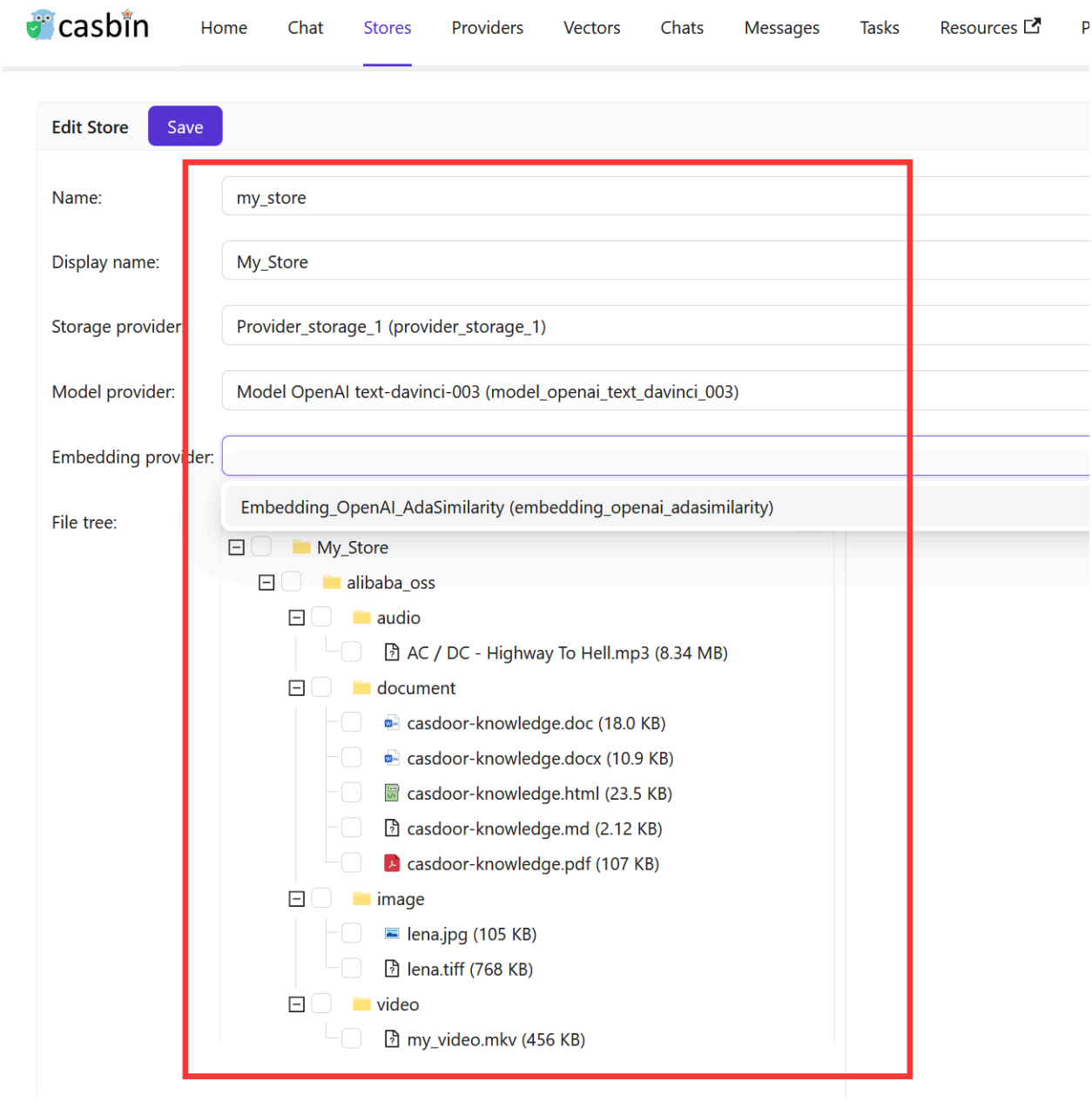
Click the `Stores` button on the home page and then click the `Add` button to add a store.



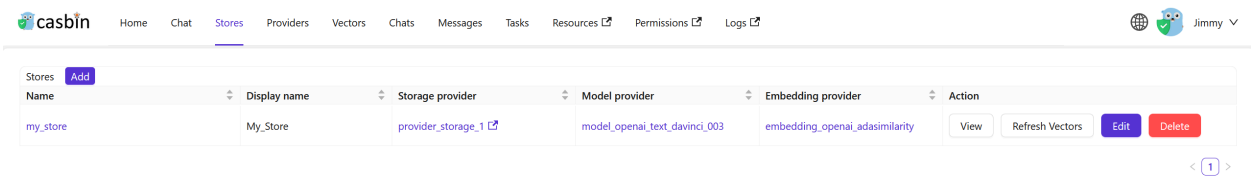
Step 2: Fill in Store Details

Select the storage provider, model provider, and embedding provider you added before.

Fill in the store details and click the `Save & Exit` button.



Click the `Save & Exit` button and return to the stores list page:



Now, you can use the store to store knowledge base data, convert text to vectors, and chat with the chatbot.

In the next section, we will learn how to chat with the chatbot in Casibase.

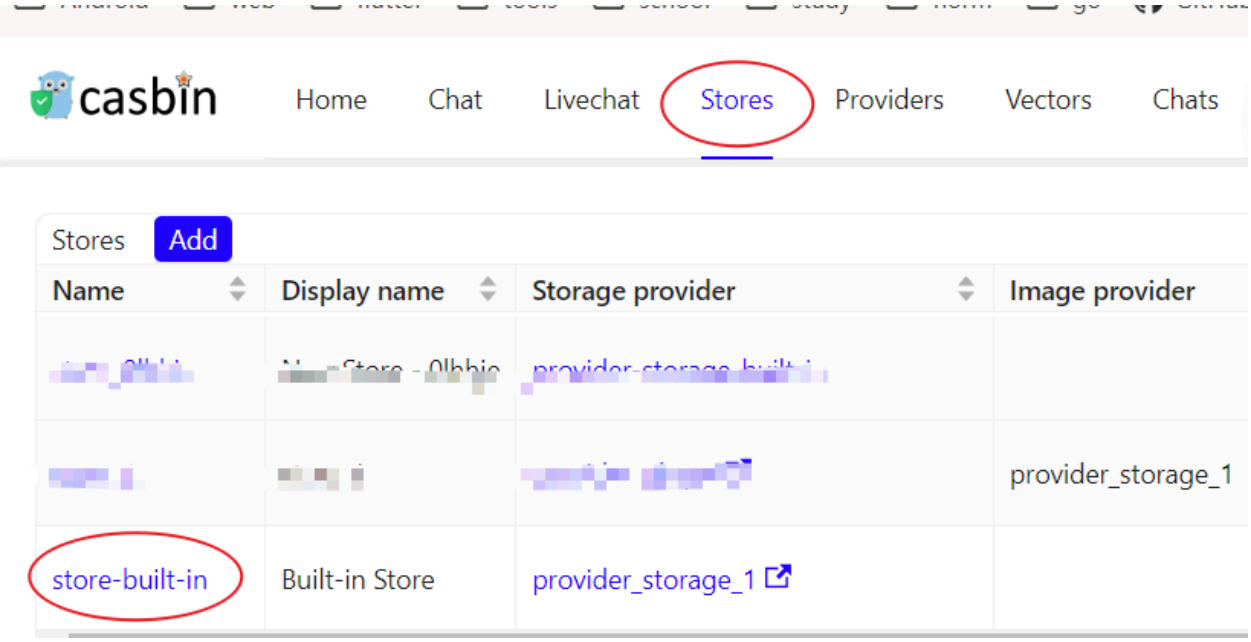
Support Multi-store

The multi-store mode provides users with different models, suggestions, and more within each distinct store.

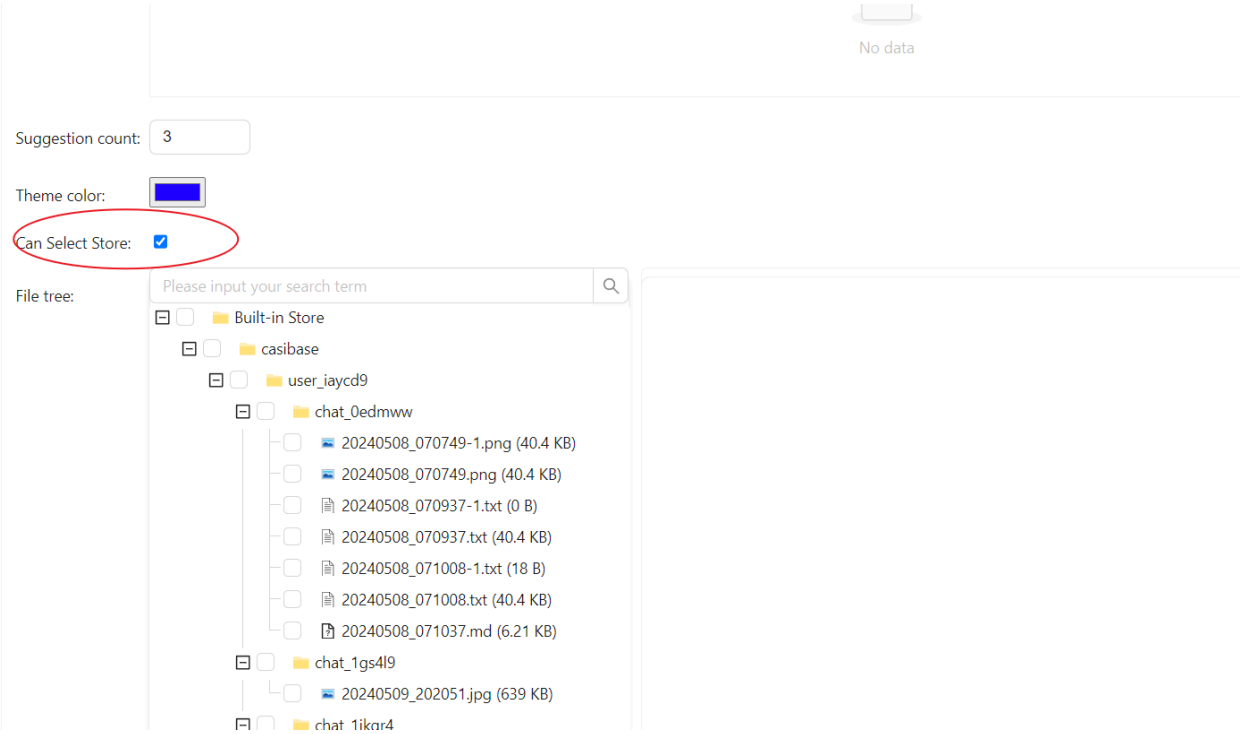
Step 1: Enable Multi-store

First, you should enable multi-store mode in the built-in store.

Click the `Stores` button on the home page and then click the `store-built-in` button to enter the store-built-in store.



Scroll down and find the `Can Select Store` field, tick it.

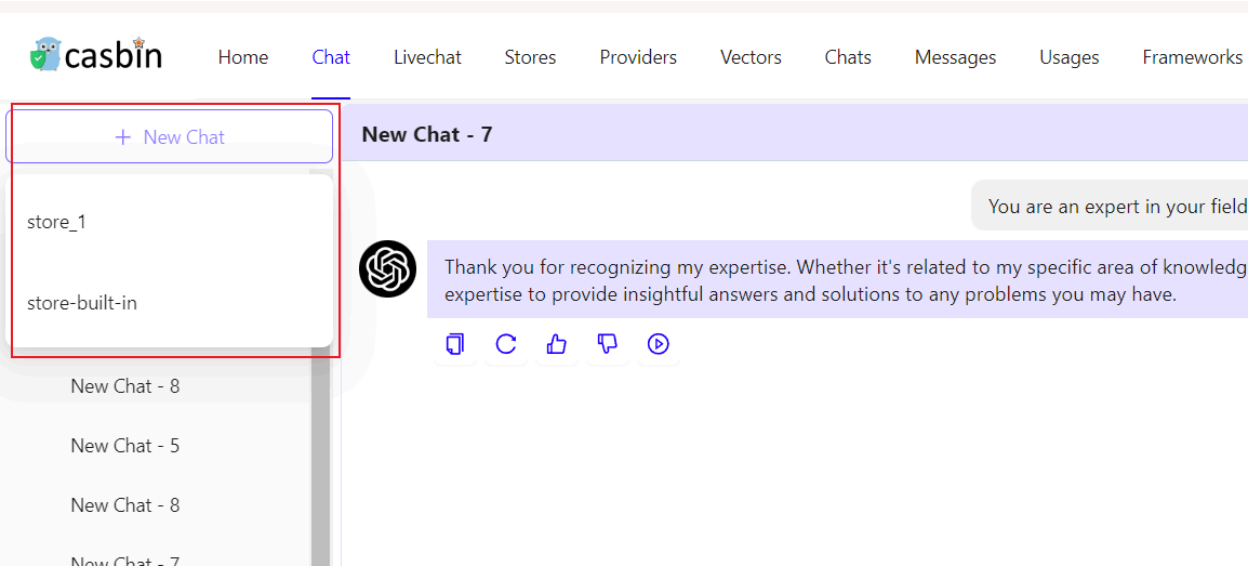


Step 2: Add Usable Stores

The multi-store mode only provides usable stores. To make a store usable, you need to configure its storage provider, model provider, and embedding provider.

Step 3: Select For Conversation

Casibase provides a very convenient method for selecting a store.



Just hover your mouse over "New Chat" and then you can select the Store you wish to use from the list that appears below.

If you click the "New Chat" button, the system will assign you a default Store.

Chats with AI

This document is a step-by-step tutorial designed for beginners. It will guide you through the process of implementing AI chat functionality in your Casibase knowledge base system.

Introduction

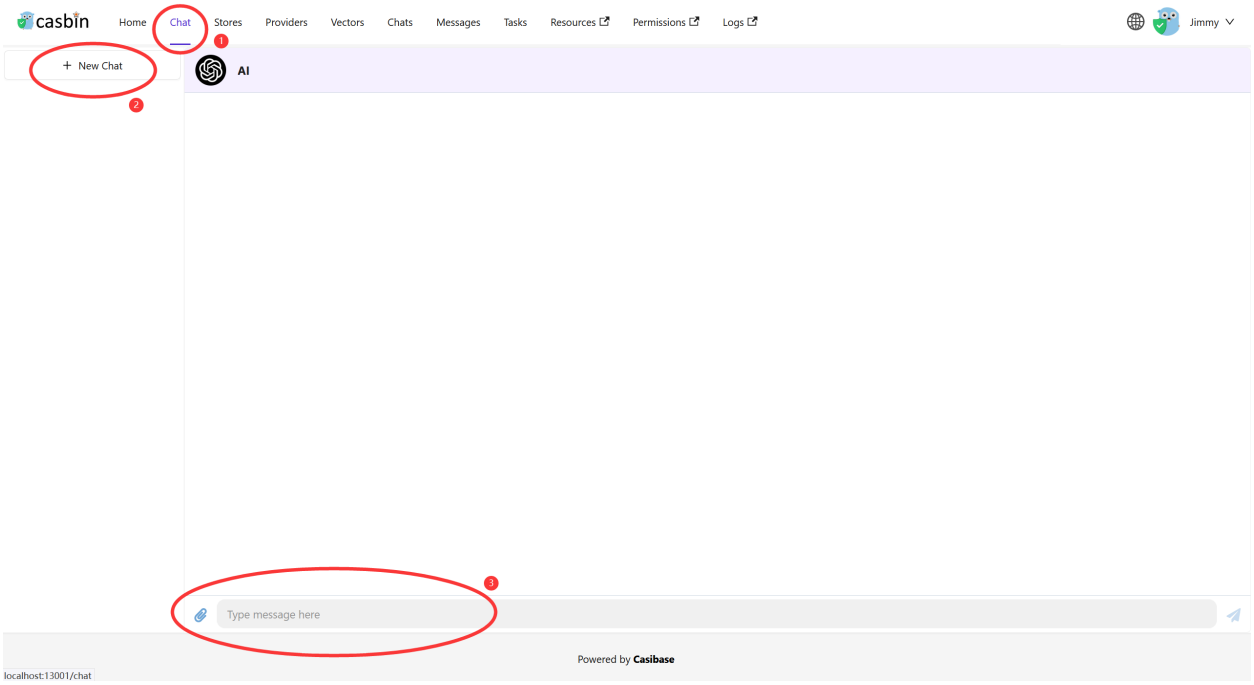
In previous sections, we have deployed Casdoor and Casibase, integrated a storage provider, a model provider, and a embedding provider with Casibase, and added a store to use these providers.

Refer to the Add a Store section of our previous documentation for more information about stores.

Now, let's implement AI chat functionality in Casibase.

Step 1: Add a New Chat

Click the `Chats` button on the home page and then click the `New Chat` button to add a chat.



Step 2: Send a Message

Write a message and click the `Send` button to send it.

Step 3: Knowledge Base Chat

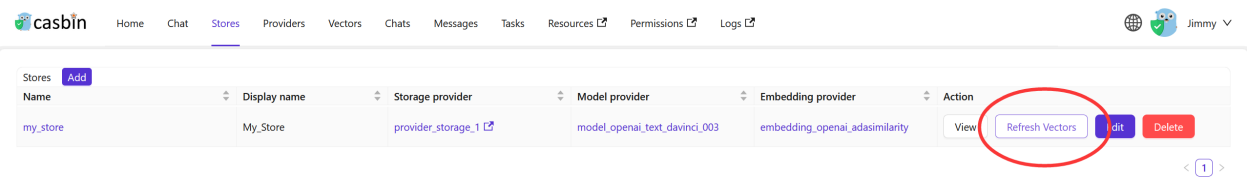
Additionally, you can chat with the chatbot in the knowledge base.

There are some requirements for the knowledge base chat:

- The knowledge base must have a store.
- The store must have a model provider.
- The store must have an embedding provider.

- The store must have a storage provider.
- The storage provider must have a readable document (e.g. a markdown file, docx file and pdf file).

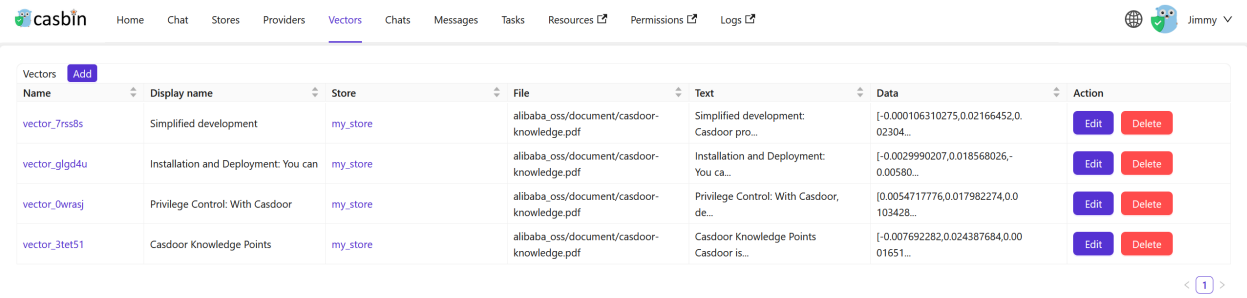
Once you have met these requirements, you can return to the `Stores` page and click the `Refresh Vectors` button to embedding the knowledge base data.



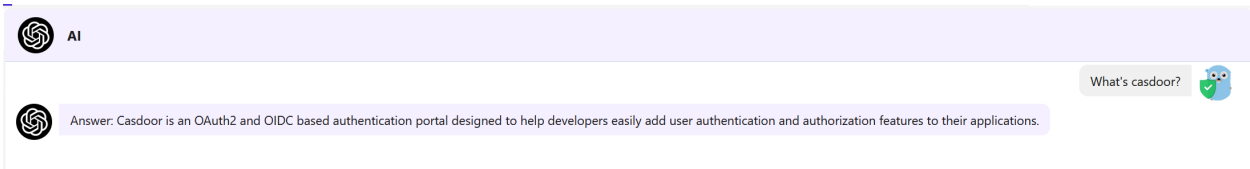
The button will be disabled when the embedding is in progress.

After the embedding is complete, you can click the `Vectors` button in the navigation bar to view the vectors.

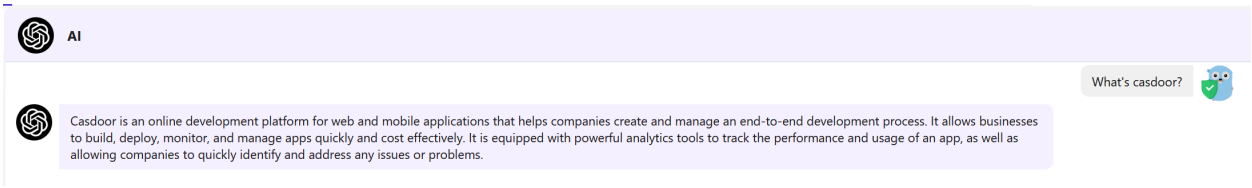
Result:



Let's **chat with the chatbot in the knowledge base.**



Compare the results with **non-knowledge base chat**:



> ⚠ CAUTION
>
> The embedding rate is related to two factors:
>
> - The documents in the knowledge base:
>   - Number of documents: The more documents, the longer the embedding time.
>   - Size of documents: The larger the document size, the longer the embedding time.
> - The embedding provider:
>   - API rate limit: The more API rate limit, the faster the embedding speed.
>   - API concurrency: The more API concurrency, the faster the embedding speed.
>
> For example, if you use the OpenAI API as the embedding provider, the embedding rate is related to the OpenAI API rate limit and concurrency.

Conclusion

In this guide, we have learned how to implement AI chat functionality in Casibase.

Now, you can chat with the chatbot in Casibase. Enjoy it!

More information about Casibase can be found in the Core Concepts section of our documentation.

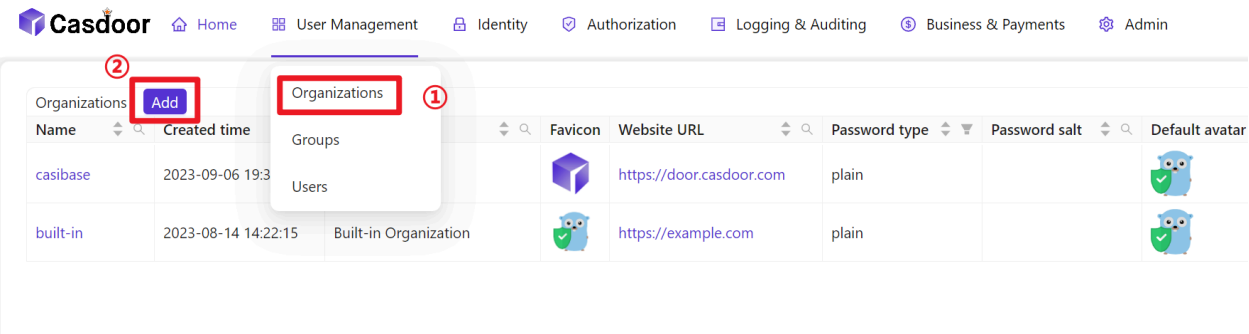
Casdoor-SSO

Casibase uses Casdoor as its identity and single-sign-on (SSO) provider. Make sure to deploy it in advance.

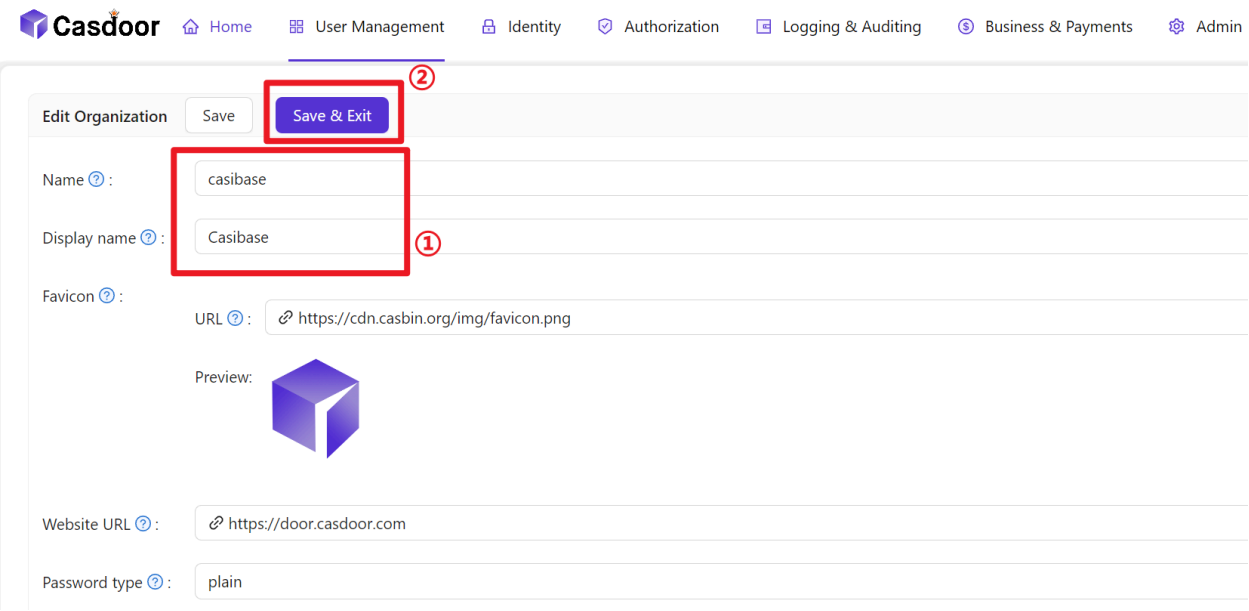
Please refer to Casdoor Server Installation to install and configure Casdoor.

Follow these steps to setup Casdoor for casibase:

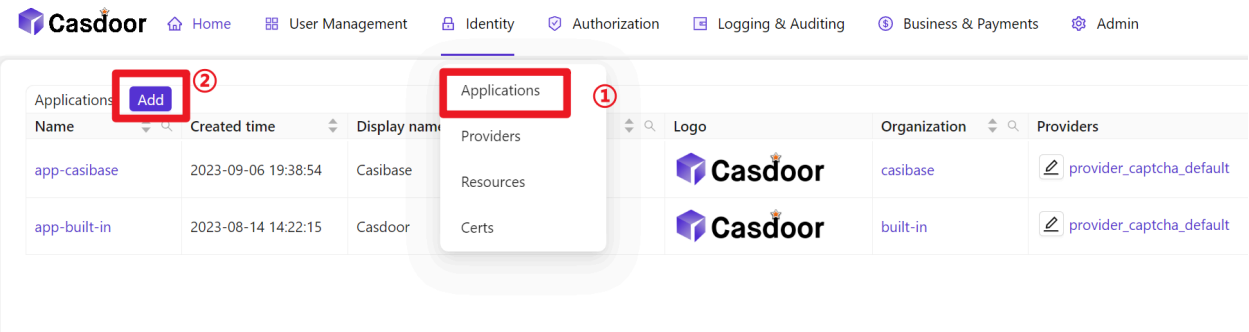
- Create an Organization



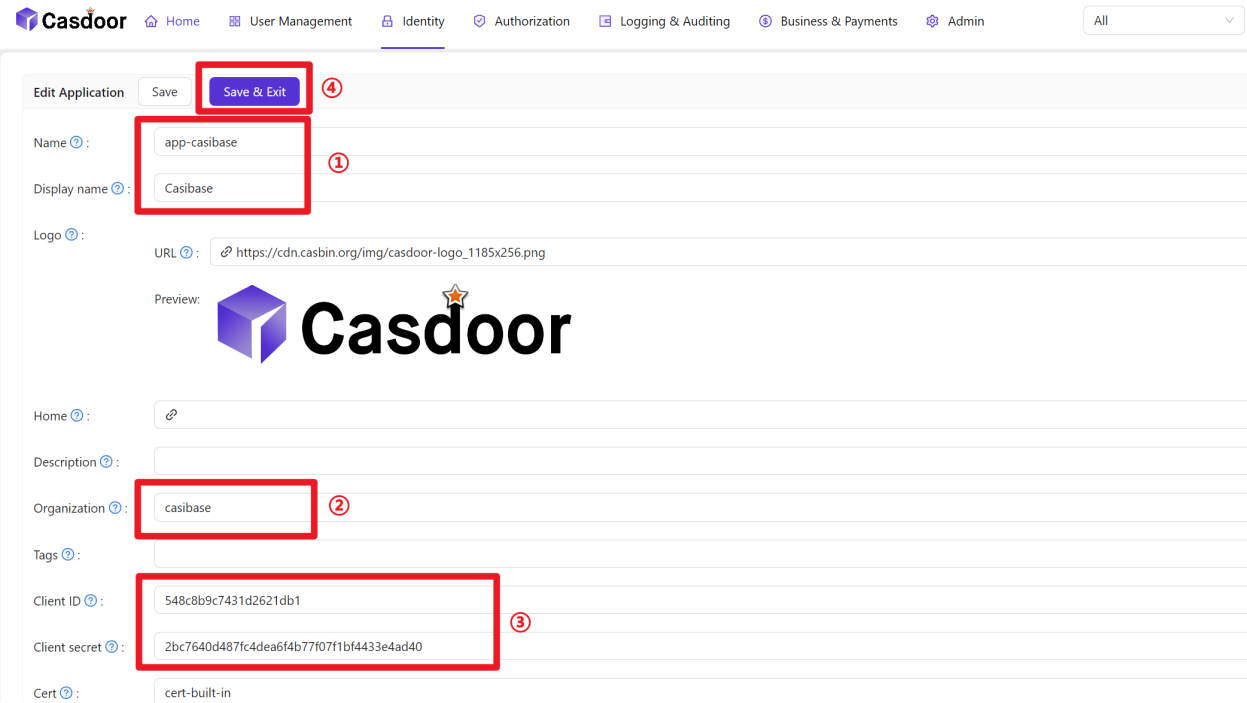
- Configure information about the Organization



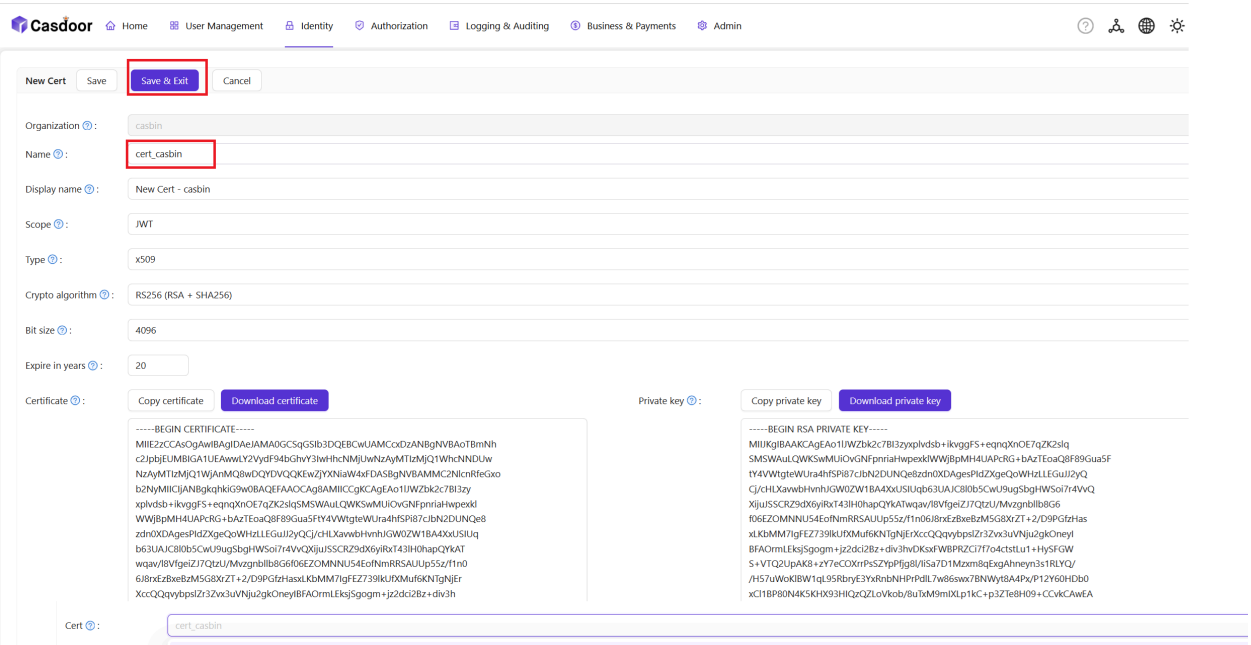
- Create a new Application



- Configuring Application Information (Remember Name, ClientID and ClientSecret)

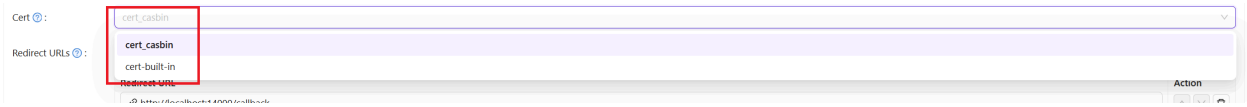


- Create a Certificate: In the Casdoor dashboard, choose Cert → Add, keep `Algorithm` as `RS256` (default), enter a name, and click Save.

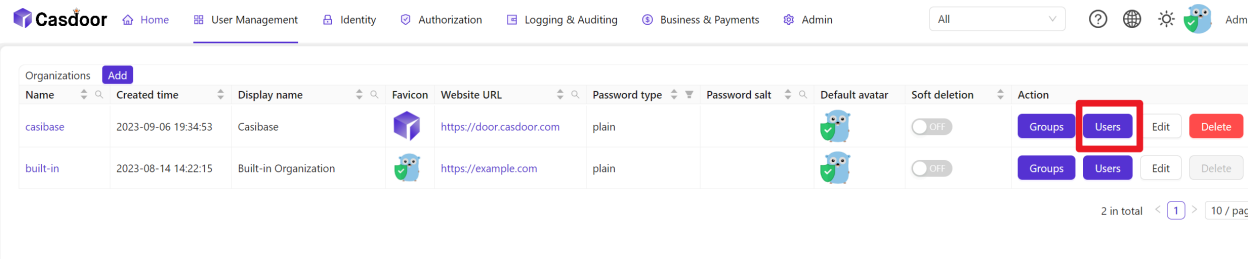


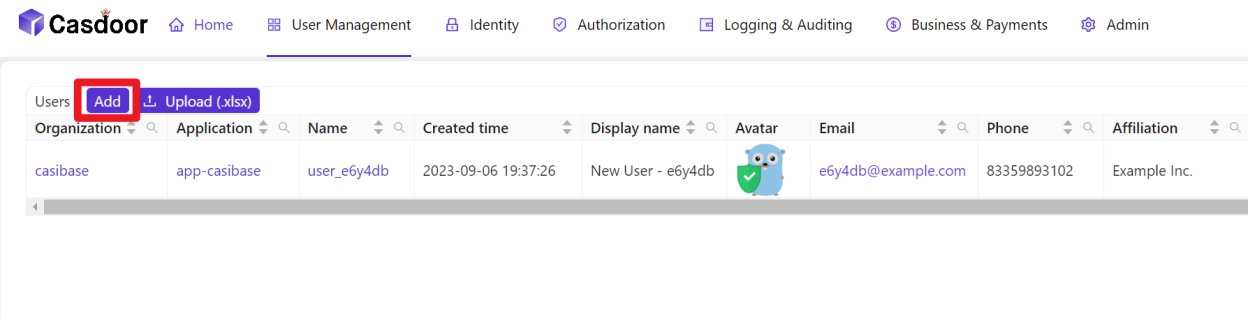
- Bind the Certificate to the Application: Open the *Config* tab of your newly

created Application, select the certificate you just created from the Cert dropdown, and click Save.

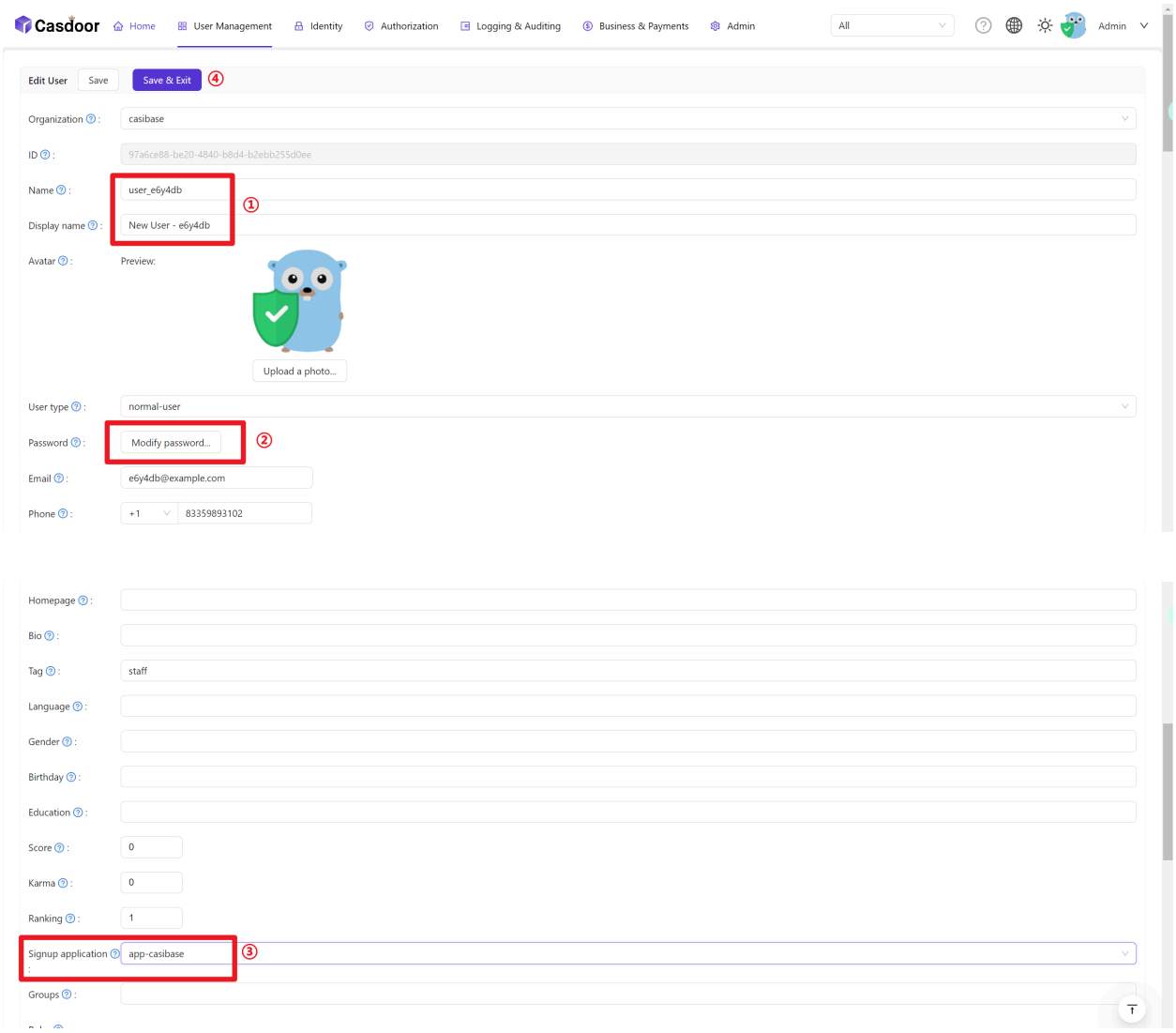


- Add a member to the newly created organization





- Configure member information (remember its Name as well as Password)


Casdoor
Home
User Management
Identity
Authorization
Logging & Auditing
Business & Payments
Admin
All
Admin
Edit User
Save
Save & Exit
④
Organization :
casibase
ID :
97a6ce88-be20-4840-b8d4-b2ebb255d0ee
Name :
user_e6y4db
①
Display name :
New User - e6y4db
Avatar :
Preview:
Upload a photo...
User type :
normal-user
Password :
Modify password...
②
Email :
e6y4db@example.com
Phone :
+1
83359893102
Homepage :
Bio :
Tag :
staff
Language :
Gender :
Birthday :
Education :
Score :
0
Karma :
0
Ranking :
1
Signup application :
app-casibase
③
Groups :

Developer Guide

📄 Generating Swagger Files

Generating Swagger 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". Casibase 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 Casibase.

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.

Deployment

📄 Deploy Casdoor and Casibase

Discover how to deploy Casdoor and Casibase.

Deploy Casdoor and Casibase

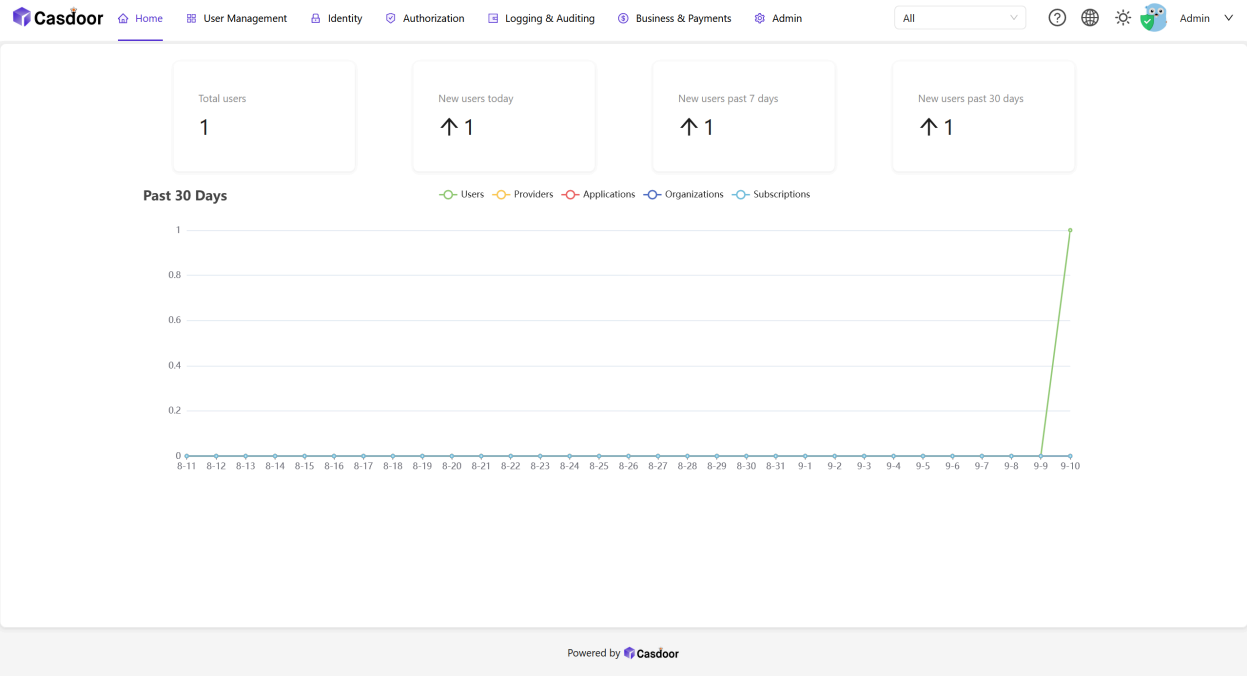
Introduction

> TIP
>
> What is Casdoor?
>
> Casdoor is a powerful authentication system that provides a secure and reliable login experience. It's a prerequisite for Casibase, so be sure to deploy it first.
>
> Refer to the Casdoor website for more information.

Step 1: Deploy Casdoor

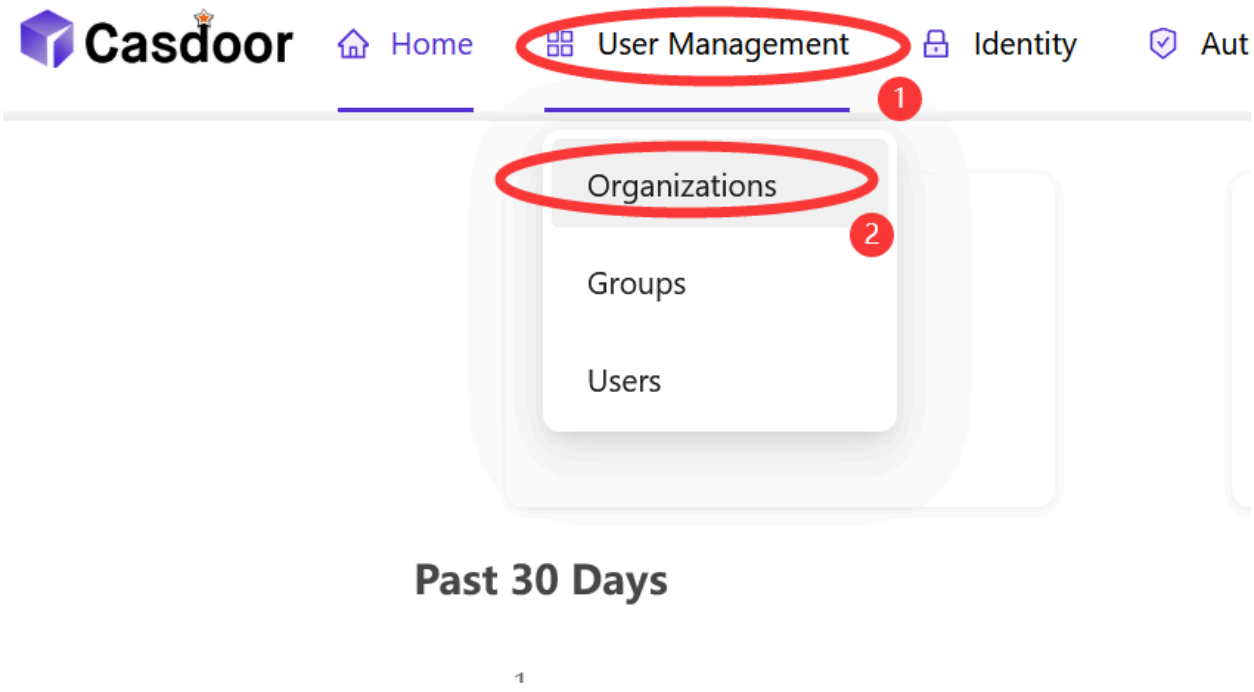
In Casdoor Deployment Guide, you can find the detailed steps to deploy Casdoor.

Once you've deployed Casdoor, you'll look like this:



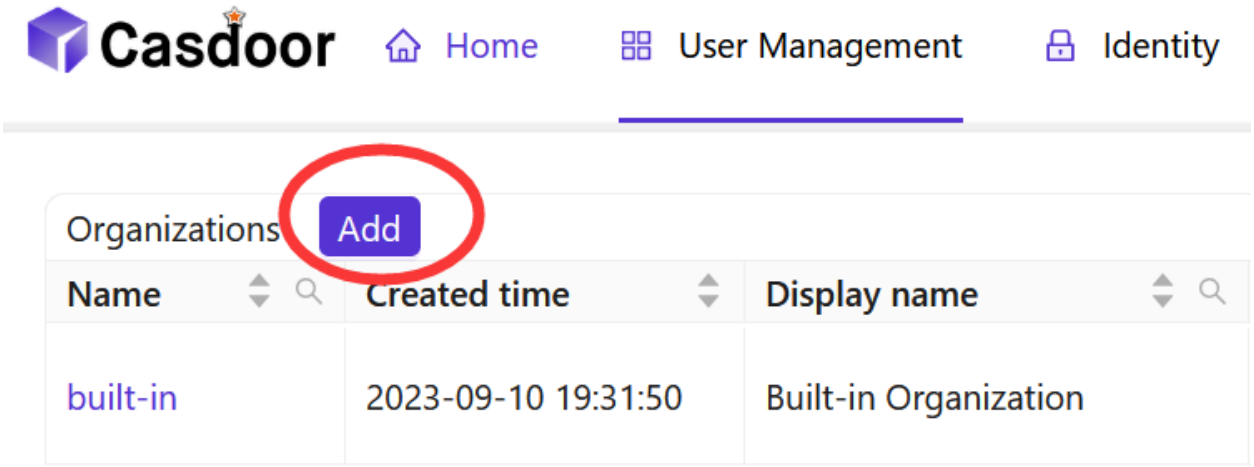
Step 2: Create an organization in Casdoor

In Casdoor, you can create an organization to manage your users and applications. You can create an organization by clicking the `User Management - Organizations` button on the home page.



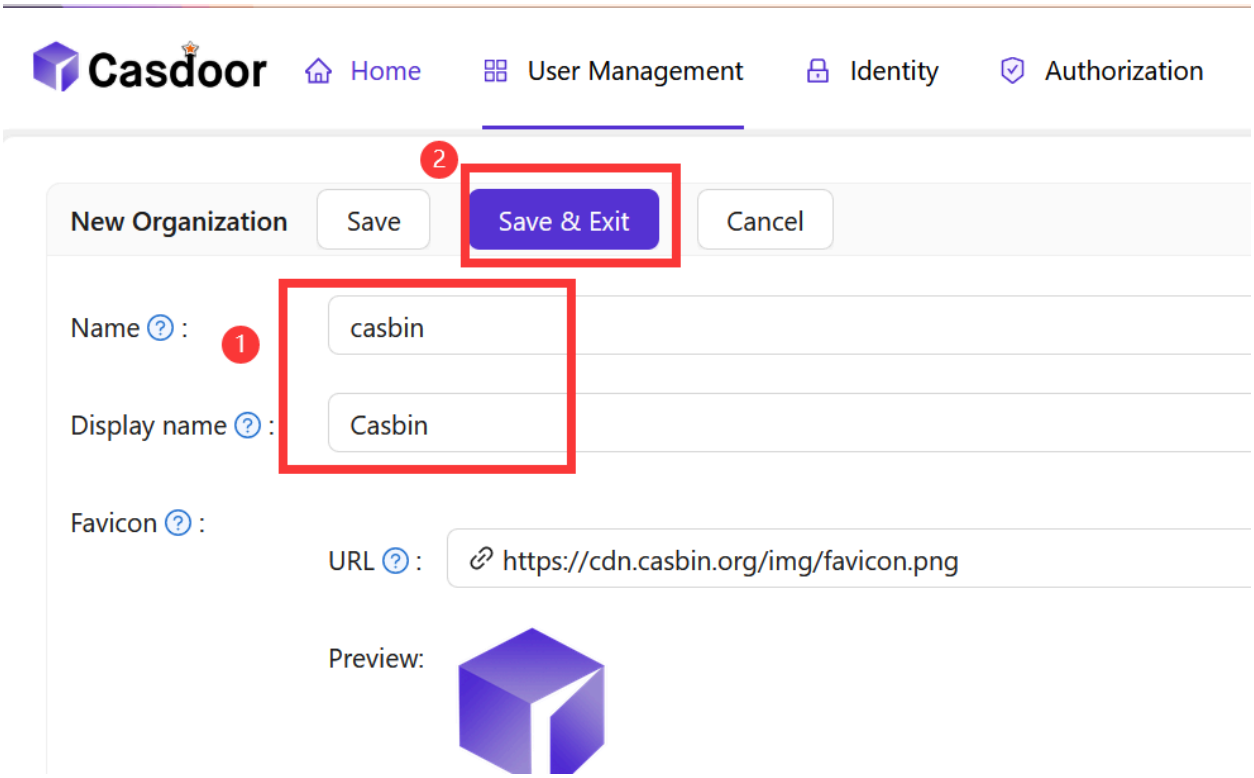
Step 2.1: Add an organization

Click the `Add` button to add an organization.



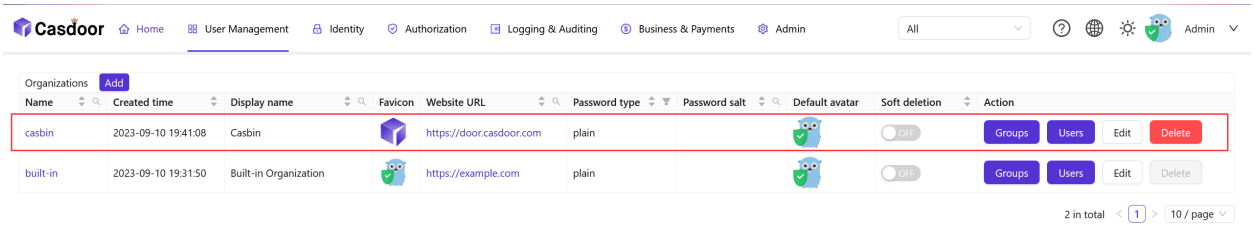
Step 2.2: Fill in the organization information

Fill in the organization information and click the `Save & Exit` button.



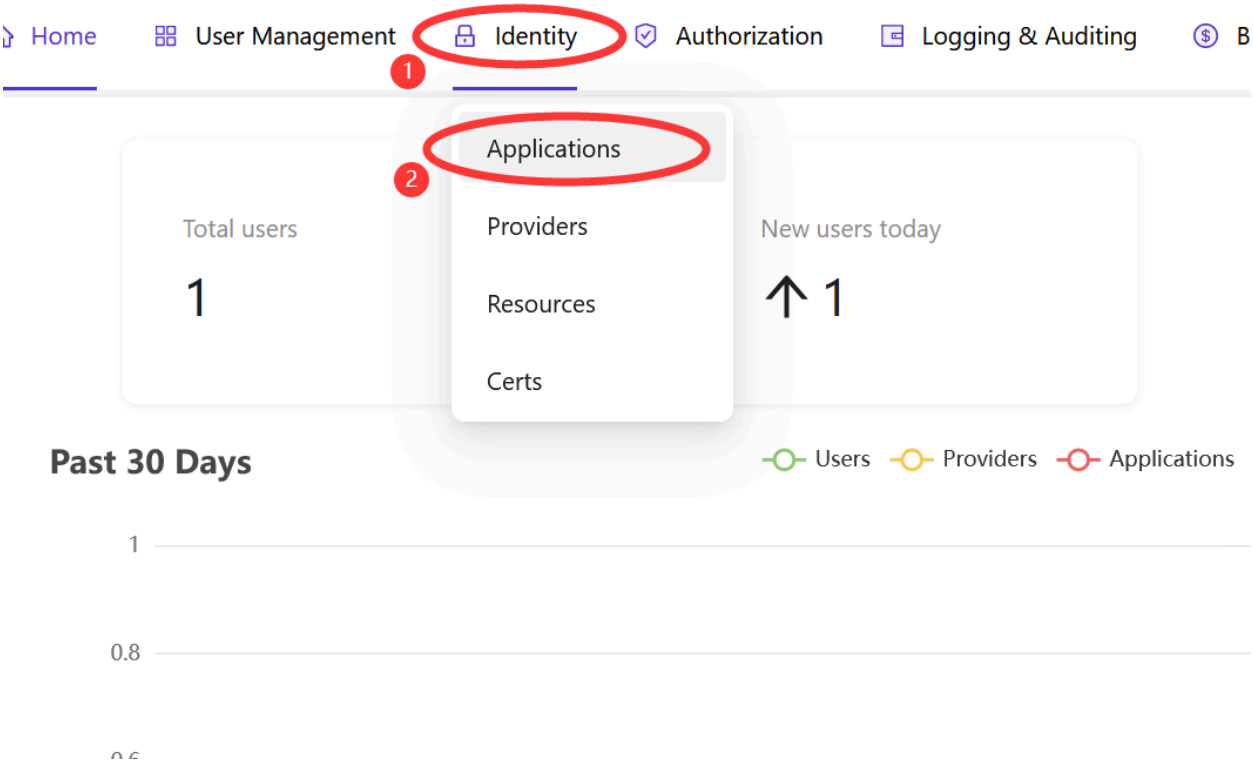
Step 2.3: View the organization

After adding the organization, you can view the organization information.



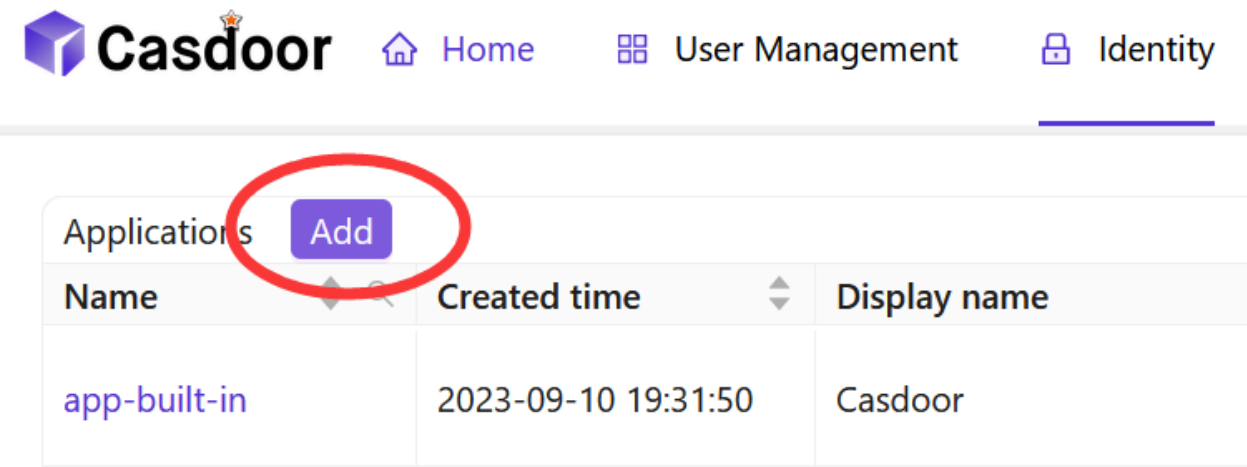
Step 3: Create an application in Casdoor

In Casdoor, you can create an application to manage your users and organizations. You can create an application by clicking the `Identity - Applications` button on the home page.



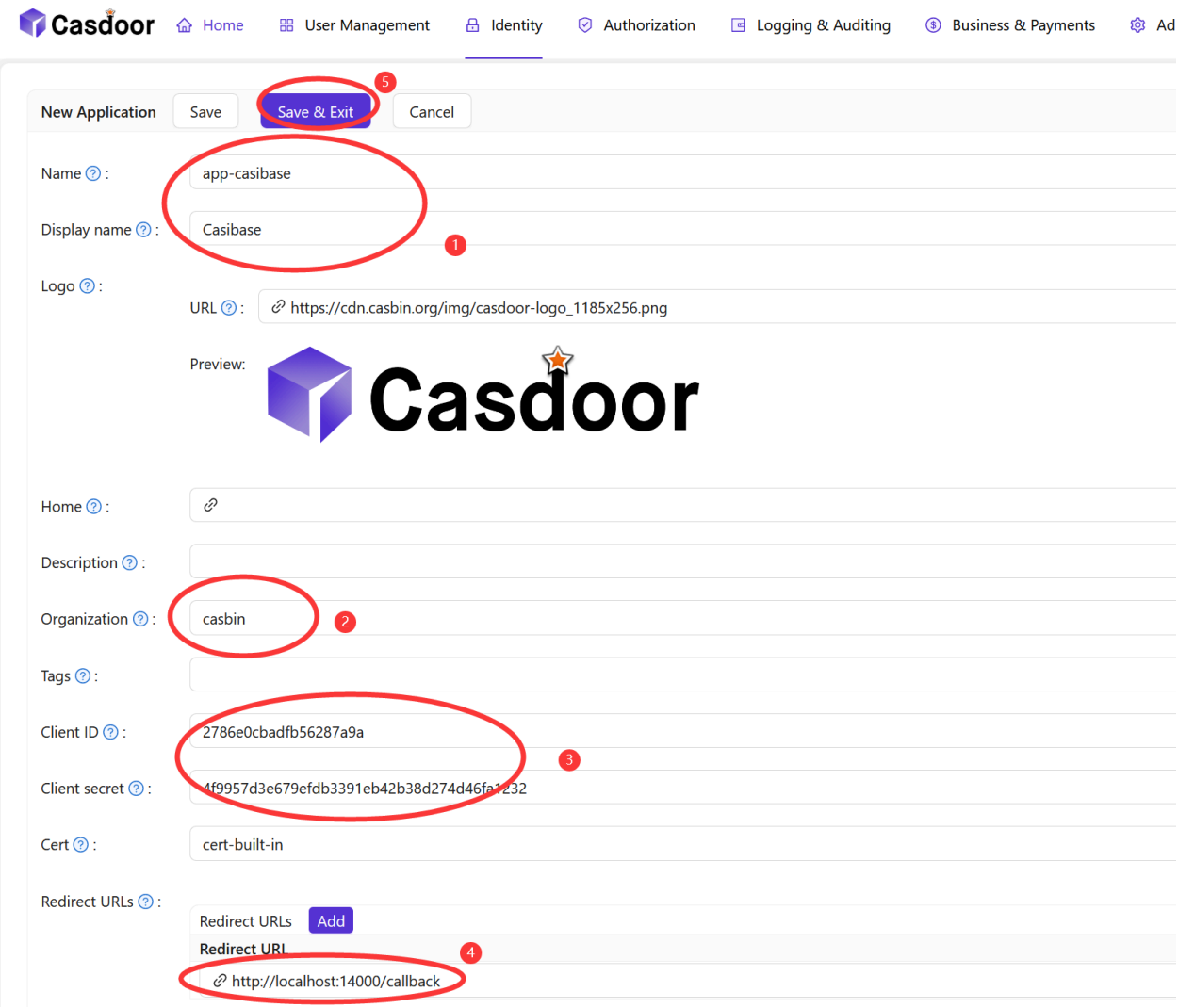
Step 3.1: Add an application

Click the `Add` button to add an application.



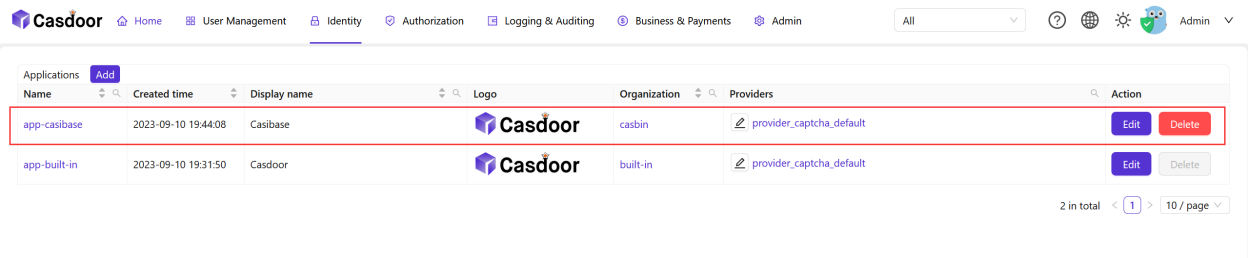
Step 3.2: Fill in the application information

Fill in the application information and click the `Save & Exit` button.



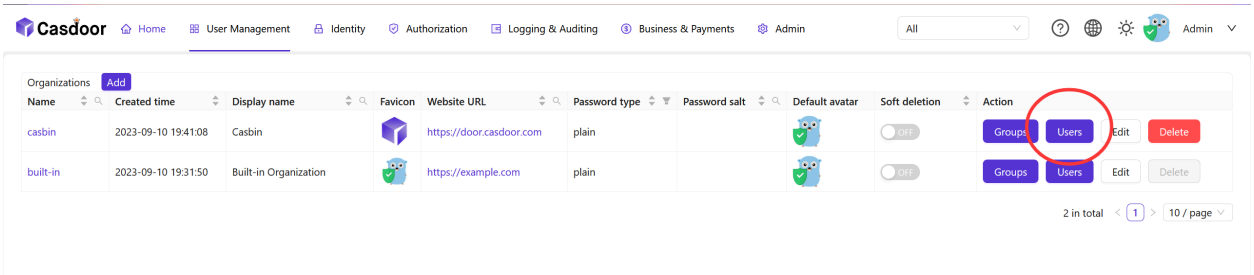
Step 3.3: View the application

After adding the application, you can view the application information.



Step 4: Create a user in Casdoor for Casibase

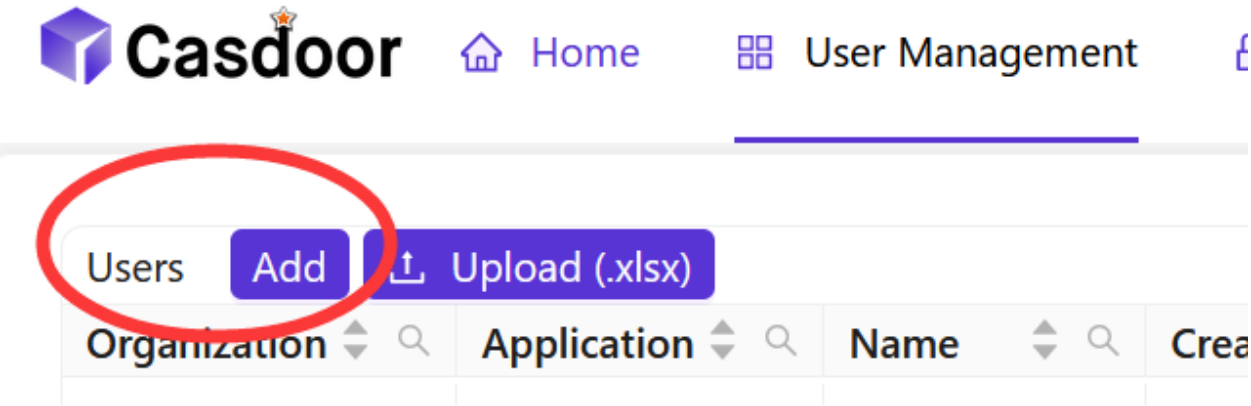
In Casdoor, you can create a user to login Casibase. You can create a user by clicking the `User Management - Organizations - Users` button from the home page.



> **TIP**
>
> A user is a member of an organization who can login to applications in the organization.
>
> Refer to the Casdoor website for more information.

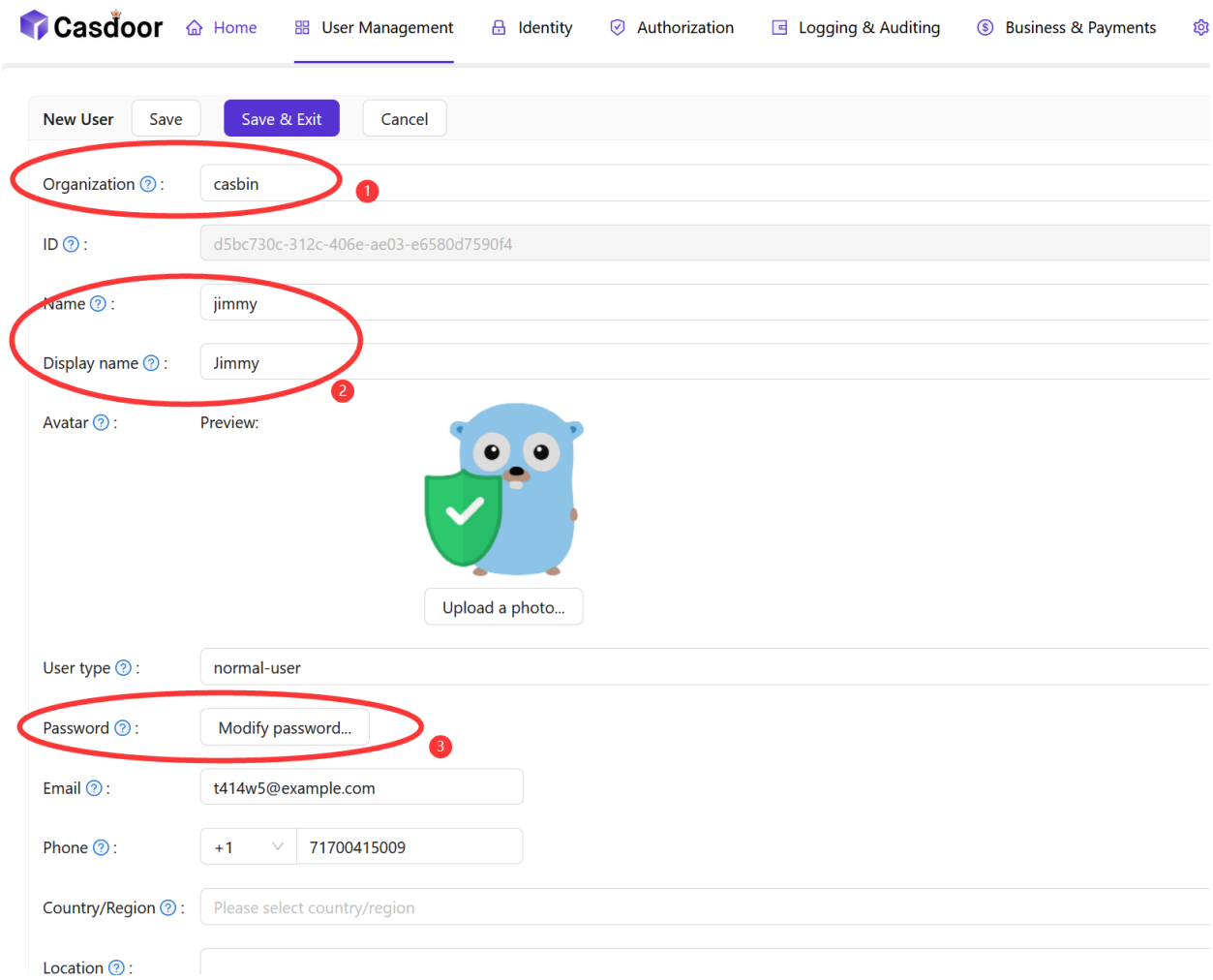
Step 4.1: Add a user

Click the `Add` button to add a user.



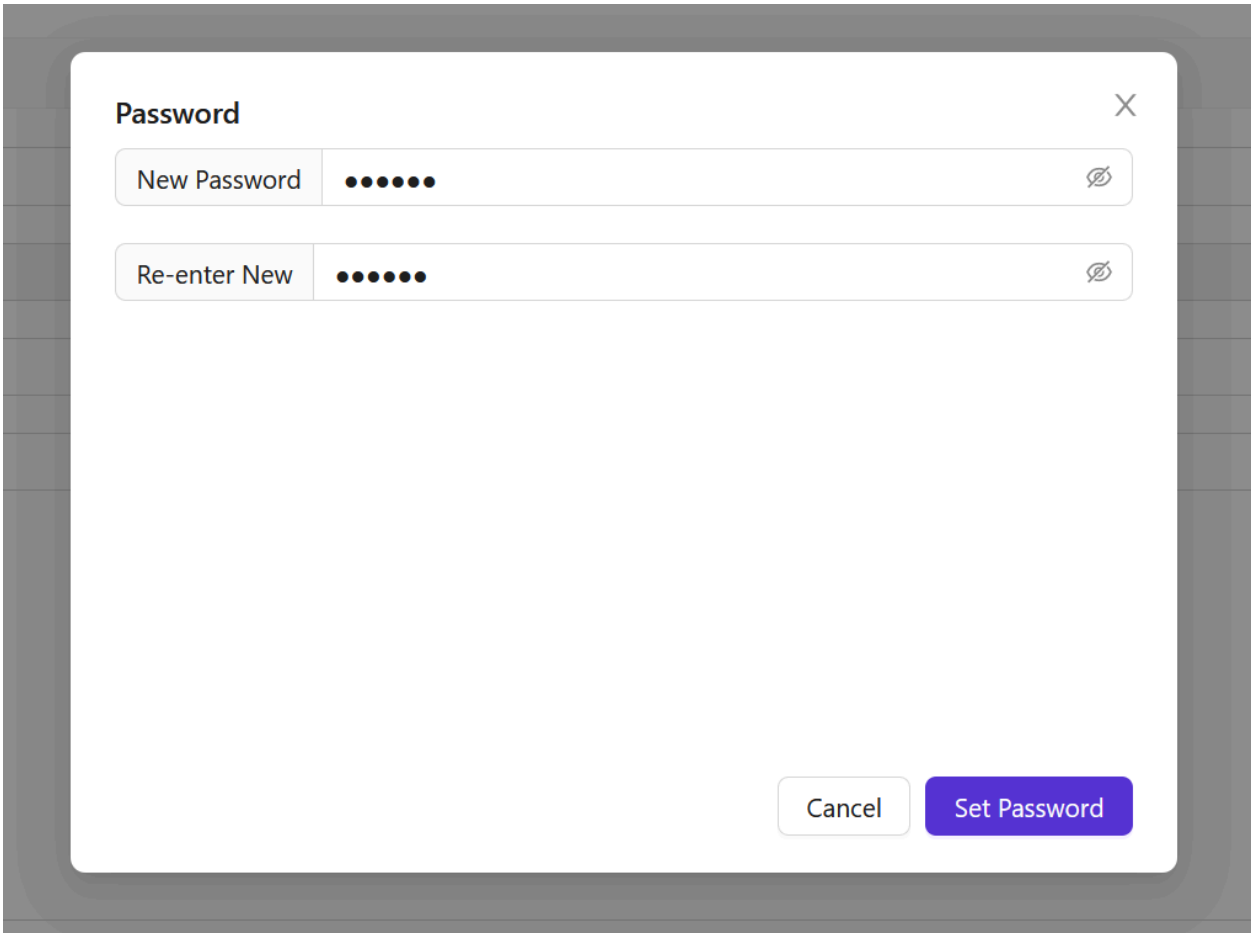
Step 4.2: Fill in the user information

Fill in the user information and click the `Save & Exit` button.



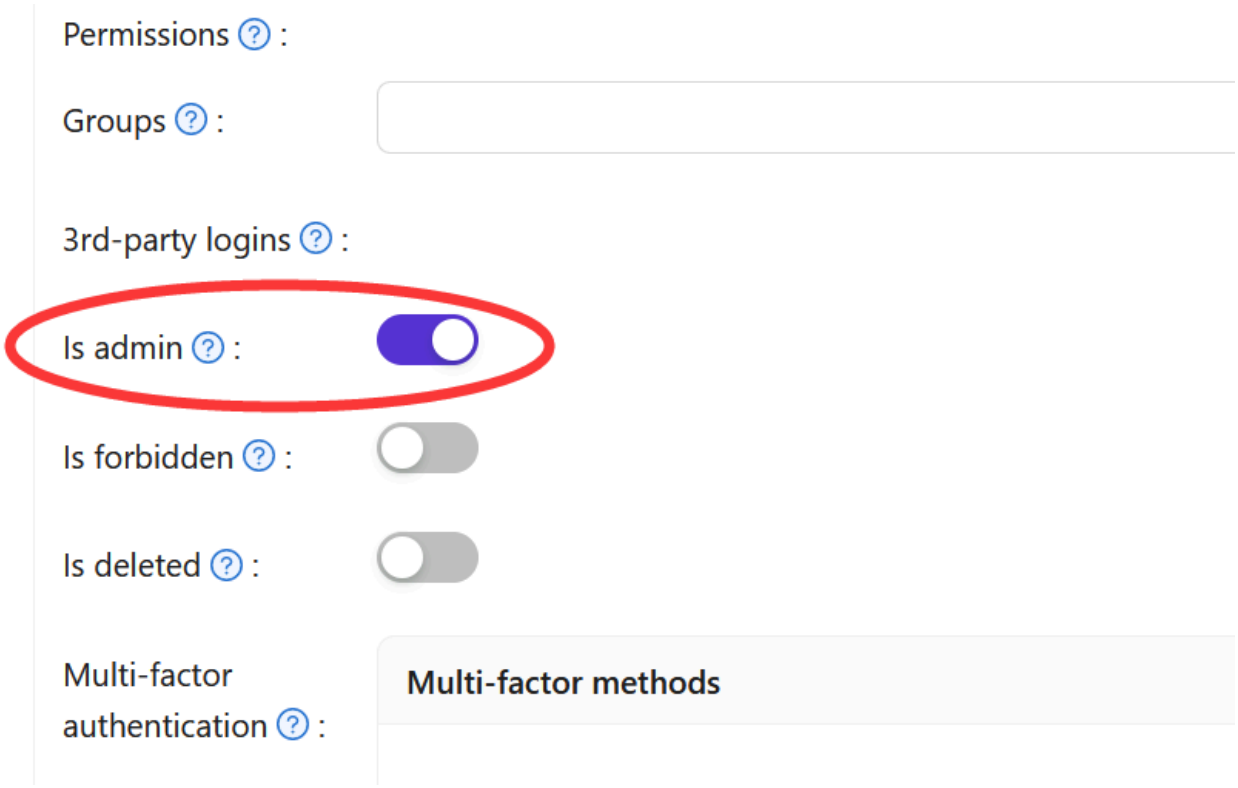
- Password

You can set the user's password by clicking the `Modify password` button.



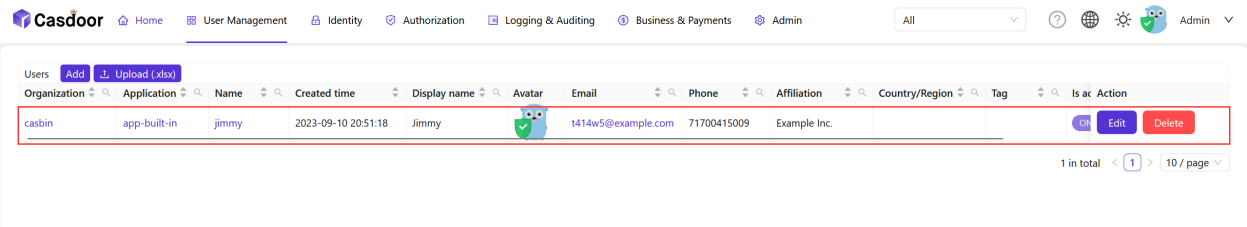
- Admin

You can set the user's admin permission by clicking the `Is admin` button.



Step 4.3: View the user

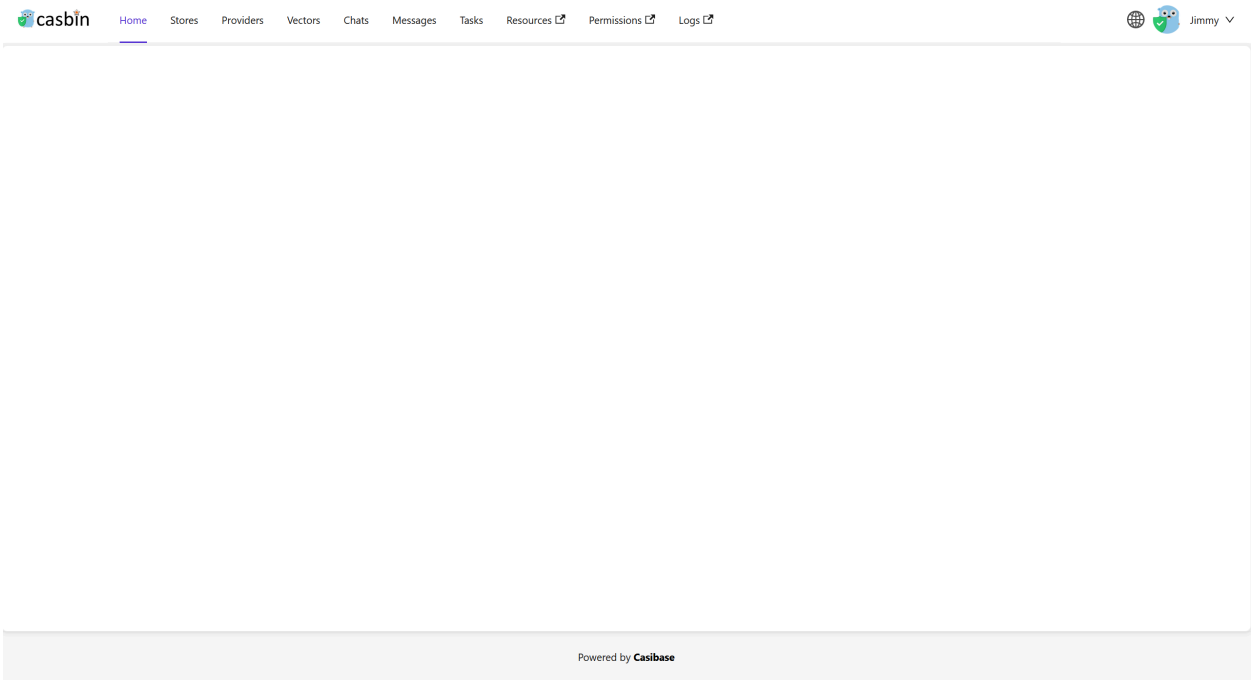
After adding the user, you can view the user information.



Step 5: Deploy Casibase

Like Casdoor, you can deploy Casibase by following the Casibase Deployment Guide.

Once you've deployed Casibase, you'll look like this:


casbin
Home
Stores
Providers
Vectors
Chats
Messages
Tasks
Resources
Permissions
Logs
Jimmy
Powered by Casibase

How to Connect to Casibase

📄 Overview

Learn about different ways to connect to and integrate with Casibase.

📄 Casibase SDKs

Learn how to integrate and use Casibase SDKs with your applications.

📄 Using Casibase OpenAI API Compatible Interface

Learn how to connect external chat UIs to Casibase using OpenAI API compatibility.

Overview

Overview

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

Casibase provides two main methods for integrating with your applications:

- `Casibase SDK` - For direct integration with Casibase's API
- `OpenAI API Compatibility` - For connecting existing OpenAI-compatible UIs and clients

Casibase SDK

What is Casibase SDK?

Casibase SDK provides a programmatic way to interact with Casibase services. It offers a convenient set of APIs that allow developers to manage tasks, knowledge bases, and other features of Casibase directly from their applications.

We recommend using the Casibase SDK for the following reasons:

1. It provides direct access to Casibase-specific functionality
2. It simplifies authentication and configuration
3. It handles error cases and provides a more developer-friendly experience

Currently, Casibase offers a Java SDK, with more language support planned for the future.

OpenAI API Compatibility

What is OpenAI API Compatibility?

Casibase supports the OpenAI API format, allowing you to connect any OpenAI-compatible chat UI or client application to Casibase. This makes it easy to use popular open-source chat interfaces with Casibase's backend.

We recommend using the OpenAI API compatibility for the following reasons:

1. It allows you to use your preferred chat UI with Casibase
2. It simplifies integration if you're already using OpenAI-compatible tools
3. It provides a standardized way to interact with Casibase's AI capabilities

This approach is particularly useful if you want to quickly integrate Casibase with existing applications that already support the OpenAI API format.

Casibase SDKs

Introduction

Casibase provides SDKs to help developers integrate with Casibase's APIs more easily. The SDKs offer a convenient way to interact with Casibase's services for tasks like managing AI conversations, knowledge bases, and more.

Currently, Casibase offers a Java SDK, with more language support planned for the future.

Backend SDK	Description	SDK code	Example code
Java SDK	For Java backends	casibase-java-sdk	-

How to use Casibase SDK?

1. Backend SDK configuration

When your application starts up, you need to initialize the Casibase SDK config by providing the required parameters.

Take casibase-java-sdk as an example:

```
CasibaseConfig config = new CasibaseConfig(
    "https://demo-admin.casibase.com",  // endpoint
    "your-client-id",                   // clientId
    "your-client-secret",               // clientSecret
```

All the parameters for initialization are explained as follows:

Parameter	Required	Description
endpoint	Yes	Casibase Server URL, like https://demo-admin.casibase.com or http://localhost:14000
clientId	Yes	Client ID for the Casibase application
clientSecret	Yes	Client secret for the Casibase application
organizationName	Yes	The name for the Casibase organization, e.g., casbin
applicationName	No	The name for the Casibase application, e.g., app-casibase

2. Available Services

Once you have initialized the configuration, you can create and use the available services. Currently, the only available service is `TaskService`.

```
TaskService taskService = new TaskService(config);
```

TaskService

`TaskService` supports basic task operations, such as:

- `getTask(String name)`: Get a single task by task name.
- `getTasks()`: Get all tasks under the organizationName.

- `addTask(Task task)`: Add a new task to the database.
- `updateTask(Task task)`: Update an existing task in the database.
- `deleteTask(Task task)`: Delete a task from the database.

Using Casibase OpenAI API Compatible Interface

This document is a step-by-step tutorial designed for beginners. It will guide you through the process of connecting external chat UIs to Casibase using its OpenAI API compatibility feature.

Introduction

Casibase now supports integration with external chat UIs through OpenAI API compatibility. This feature allows you to use popular open-source chat interfaces with Casibase's backend, giving you more flexibility in how you interact with your knowledge base system.

If you're looking to use your preferred chat UI with Casibase, this guide will walk you through the simple setup process.

Step 1: Set Up Casibase with a Model Provider

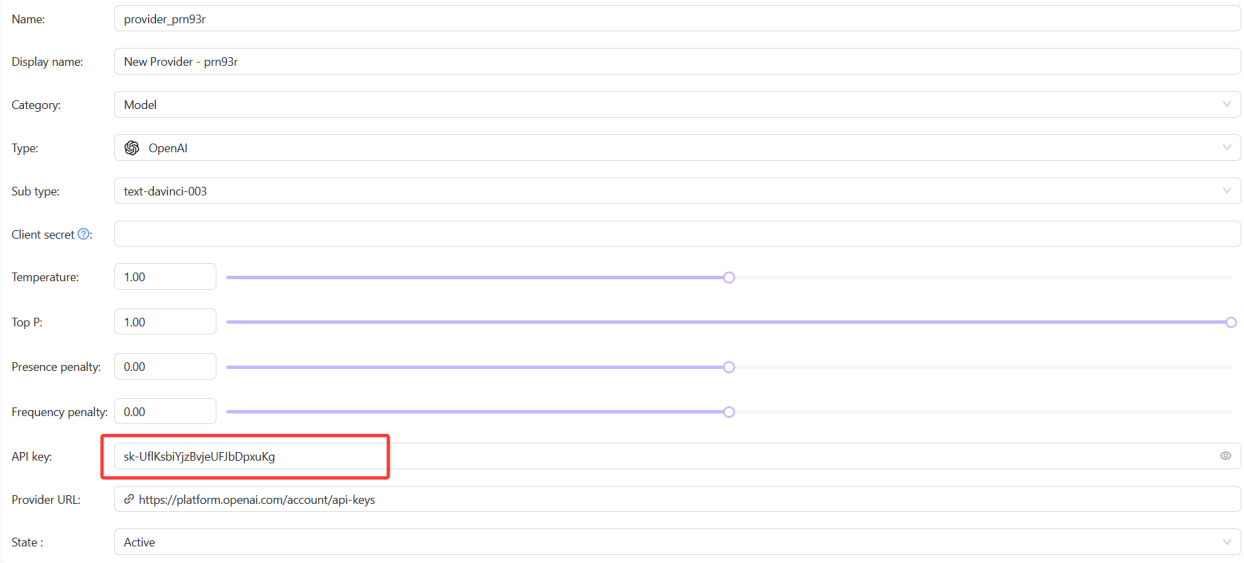
Before connecting an external UI, ensure you have Casibase properly set up with a model provider. If you haven't done this yet, please refer to the Add an AI Model Provider tutorial.

Step 2: Get Your OpenAI-compatible API Key

When you create a model provider in Casibase, an API key is automatically generated. This key allows external applications to communicate with Casibase using the OpenAI API format.

Step 2.1: Access Your API Key

Navigate to the `Providers` section and select your model provider. Only administrators can view and modify API keys.



> **TIP**
>
> If the API key field is empty, Casibase will automatically generate a new key when you save the provider.

Step 3: Configure Your External Chat UI

Once you have your API key, you can configure your external chat UI to connect to Casibase.

Step 3.1: Configure with chatgpt-web

For this example, we'll use chatgpt-web, a popular open-source ChatGPT interface.

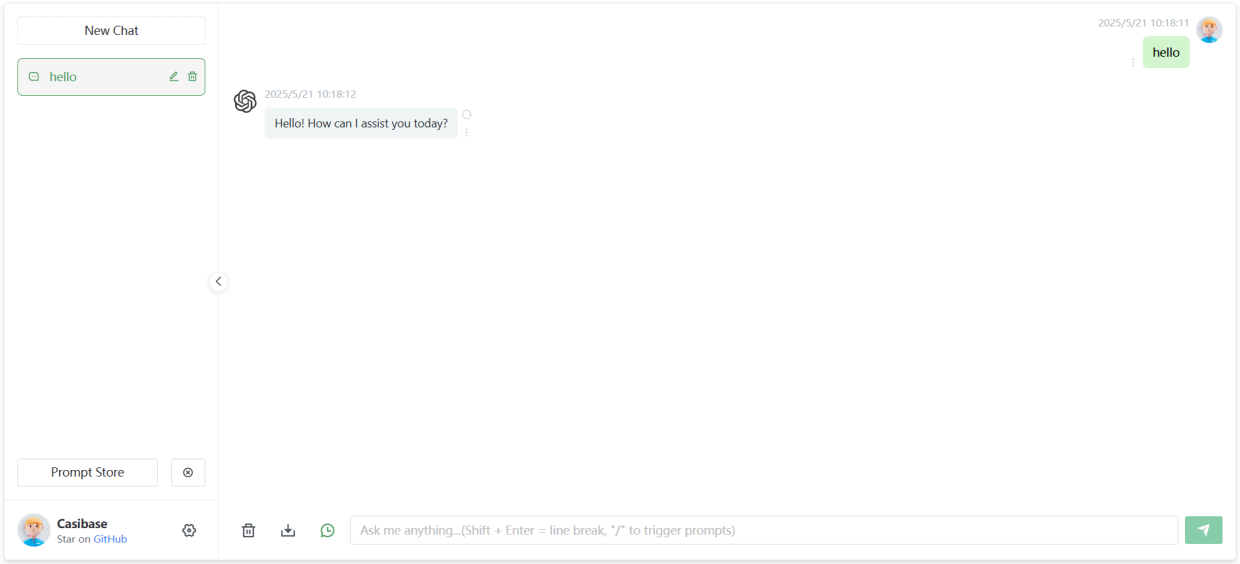
1. Locate the `service/.env` file in your chatgpt-web installation
2. Modify the following variables:
    - `OPENAI_API_KEY`: Set this to your Casibase-generated API key
    - `OPENAI_API_BASE_URL`: Set this to `http://your-casibase-backend:port/api`

```
# Example configuration
OPENAI_API_KEY=sk-UflKsbiYjzBvjeUFJbDpxuKg
OPENAI_API_BASE_URL=http://localhost:14000/api
```

Make sure your Casibase backend is accessible from the machine running your chat UI. Check firewall settings if you encounter connection issues.

Step 4: Test Your Integration

Start your chat UI application and test the connection. You should now be able to interact with Casibase through your preferred interface. If everything is set up correctly, you should see responses from Casibase in your chat UI.



Compatible Chat UIs

Casibase's OpenAI API compatibility has been tested with these popular chat interfaces:

- chatgpt-web

Other chat UIs that use the standard OpenAI API format should also work with Casibase.

Providers

Overview

Providers Overview

Model Providers

Introduction

Embedding Providers

Introduction

Storage Providers

Introduction

📄 Text-to-Speech Providers

Introduction

📄 Speech-to-Text Providers

Introduction

🗃️ Blockchain Providers

2 items

🗃️ Private Cloud Providers

1 items

Overview

Casibase is an open source AI knowledge base system designed to provide efficient and flexible knowledge management and dialogue solutions for enterprises. One of its core features is Providers, which allow users to integrate multiple AI models and storage services to enhance the functionality and performance of the system: Providers are classified into three main categories: Model Providers, Embedding Providers, and Storage Providers, where Model Providers and Embedding Providers are collectively referred to as AI Providers, which, together with Storage Providers, are responsible for handling the AI models and data storage, respectively.

1. Model Providers

Model Providers is a component in Casibase for integrating and managing AI models. It allows users to integrate various pre-trained AI models into the system for smarter knowledge processing and dialogue generation. With Model Provider, users can easily switch between different AI models, choosing the most appropriate model according to specific needs.

Casibase supports a variety of popular AI models, including but not limited to:

Model Provider Types

- Hugging Face: e.g. meta-llama/Llama-2-7b, THUDM/chatglm2-6b
- OpenAI: e.g. gpt-3.5-turbo, gpt-4
- Claude: e.g. claude-2, claude-instant-v1
- Ernie: e.g. ERNIE-Bot, ERNIE-Bot-turbo

2. Embedding Providers

Data vectorisation

The main role of Embedding Providers is to transform various types of data (e.g., text, images, etc.) into dense vector representations. This transformation is a key step in data processing and analysis in Casibase, enabling data to be stored, retrieved and analysed in a more efficient manner.

Knowledge Retrieval

By converting both the data in the knowledge base and the user's query into vectors, Embedding Providers enables the system to perform fast knowledge retrieval based on vector similarity. This greatly improves the efficiency and accuracy of knowledge base retrieval.

Flexible model support

Embedding Providers support a variety of embedding models, users can choose the most suitable model according to their needs.

3. Storage Providers

We can configure the storage providers in Casdoor. and use it in Casibase, which is the component used to manage Casibase data storage and retrieval. It allows users to store data in different storage services and access the data through a unified interface. With Storage Providers, users can flexibly choose storage services to ensure data security and efficient access. supports two types of storage: Local and Cloud.

4. Text-to-Speech Providers

Text-to-Speech (TTS) Providers is a component in Casibase that enables the conversion of text responses into natural-sounding speech. It allows the system to communicate with users through voice synthesis, enhancing the interactive experience of the knowledge base system.

Provider Support

Currently, Casibase supports Alibaba Cloud's Text-to-Speech service, with various voice options available through the cosyvoice-v1 interface. The system is designed to be extensible, allowing for the integration of additional TTS providers in the future.

5. Speech-to-Text Providers

Speech-to-Text (STT) Providers is a component in Casibase that enables the conversion of spoken language into written text. It allows the system to understand and process voice queries, enhancing the interactive experience of the knowledge base system.

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.

Model Providers

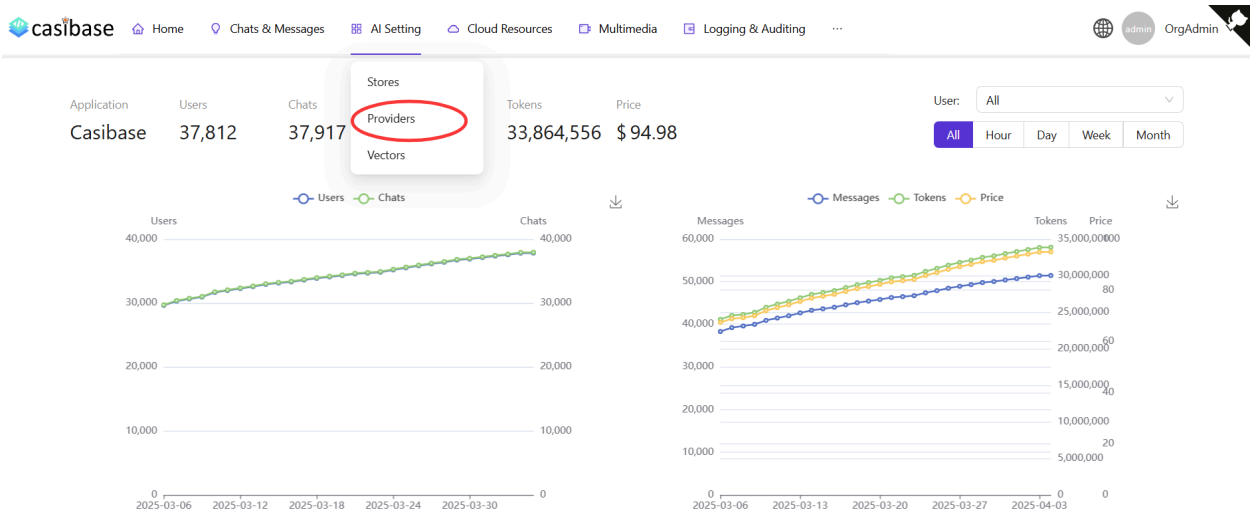
Introduction

Adding a model provider to Casibase enables you to enhance its functionality by incorporating machine learning models and AI capabilities. Model providers allow you to analyze and process data within your knowledge base system, making it more intelligent and efficient.

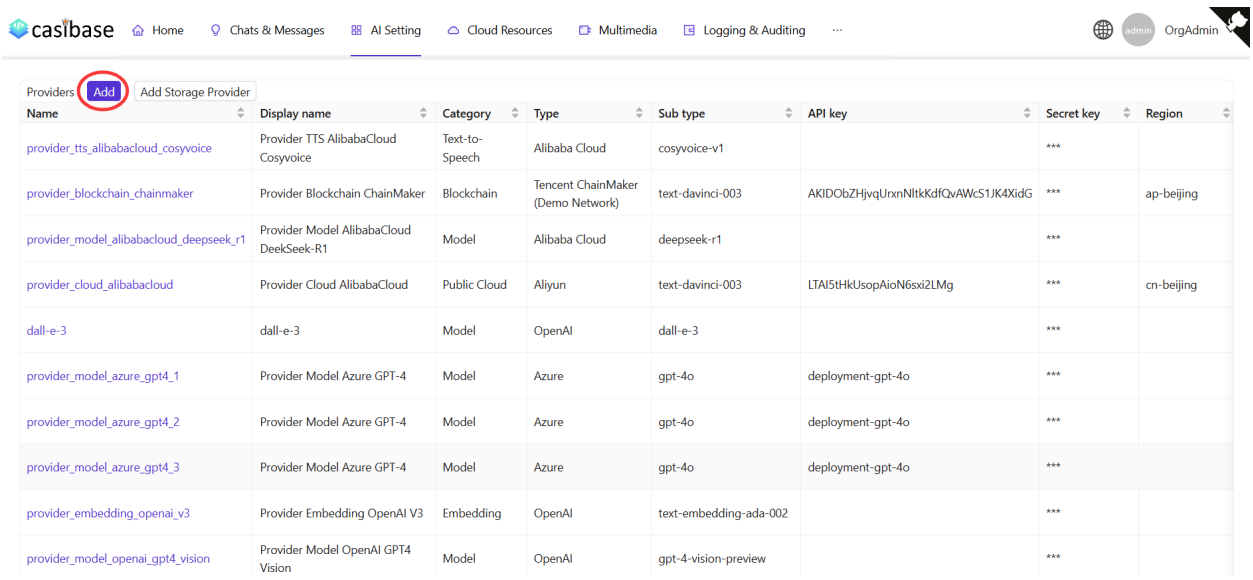
Add a Model Provider

Model providers are used to integrate LLM into Casibase. You can add them by following these steps:

Click the `Providers` button on the home page.

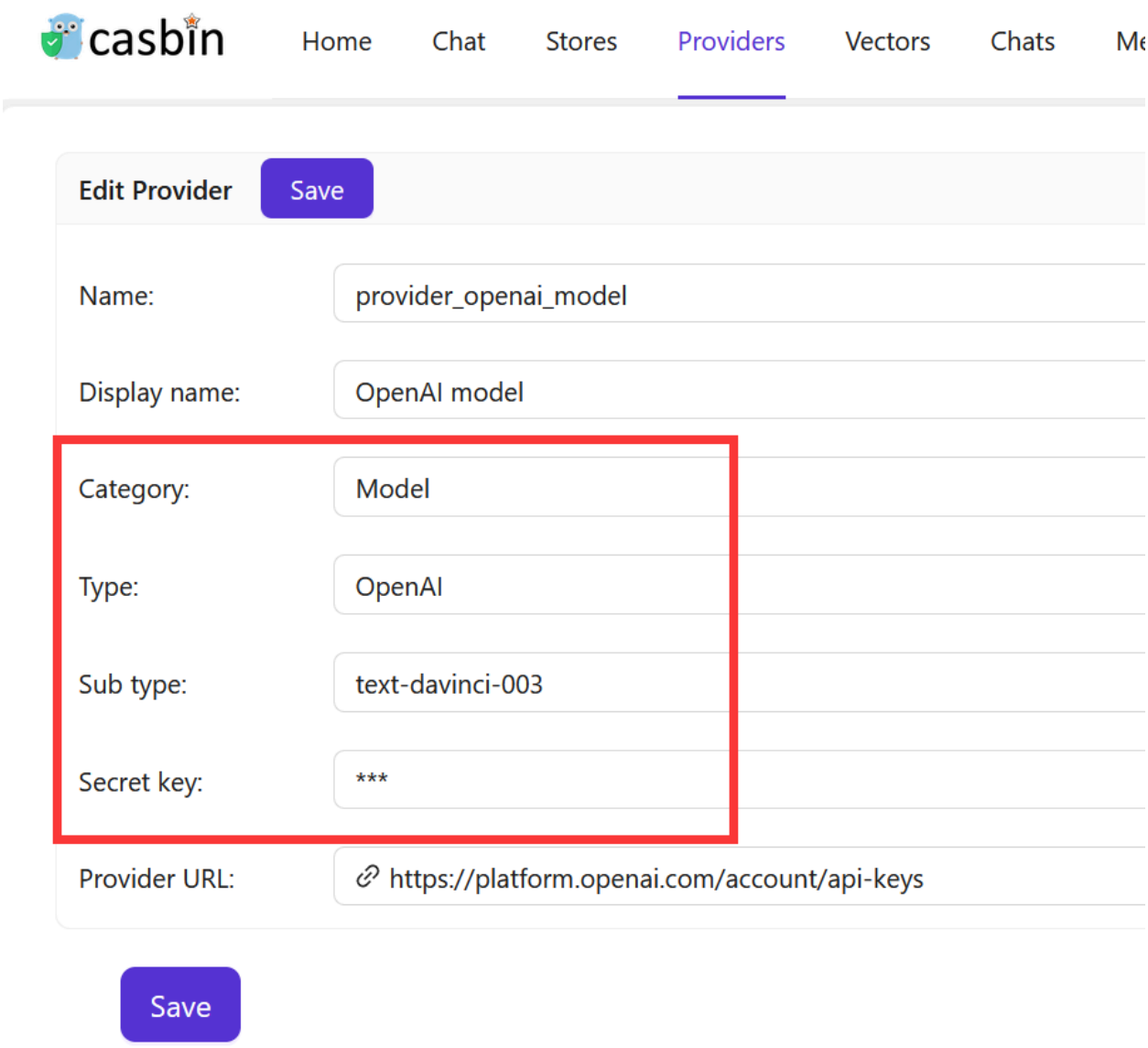


Click the `Add` button to add a model provider.



Fill in Model Provider Details

Fill in the model provider details and click the `Save & Exit` button.



TIP

Casibase supports many model providers, including:

- Hugging Face
  - meta-llama/Llama-2-7b

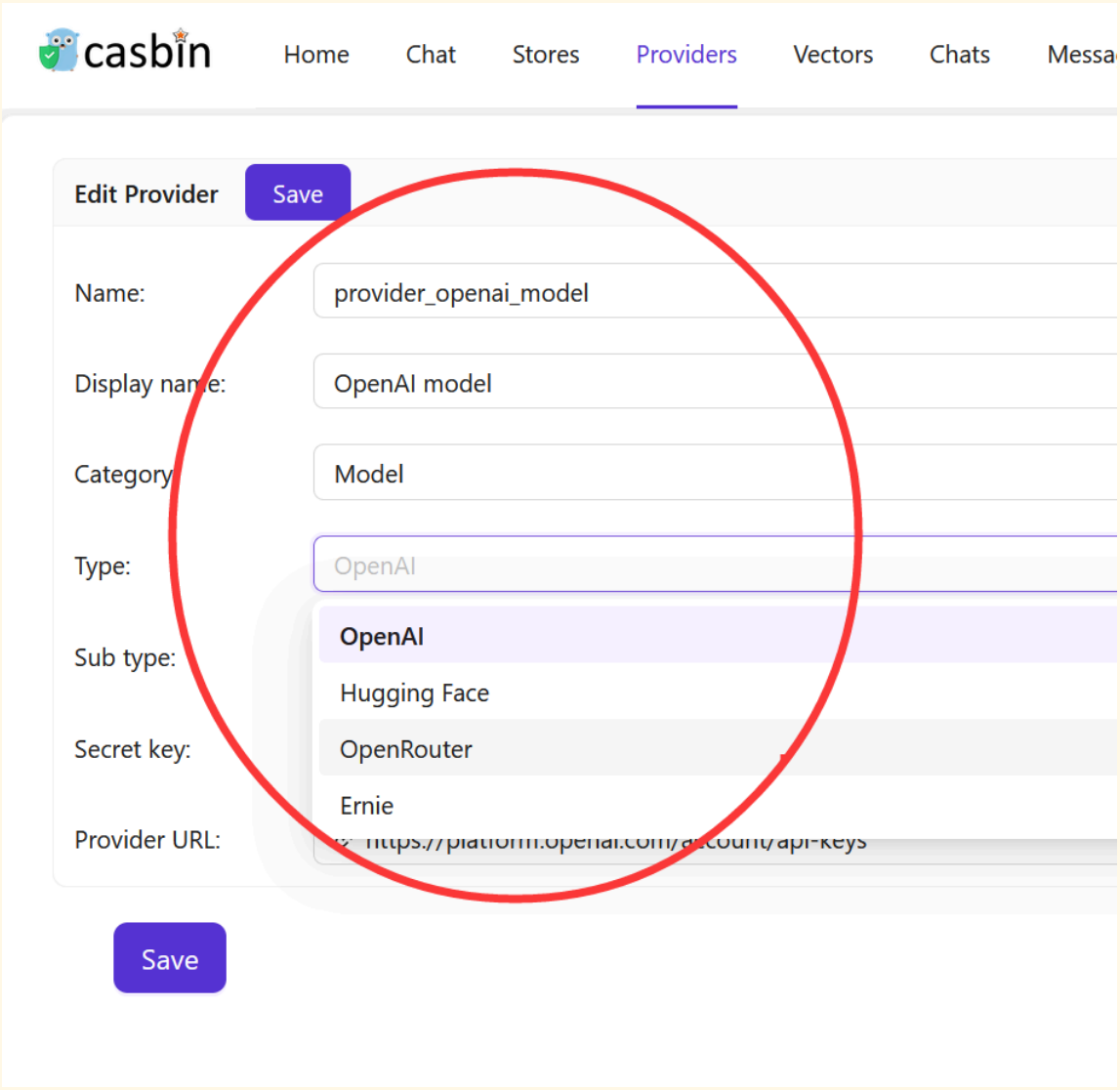
    - THUDM/chatglm2-6b
    - baichuan-inc/Baichuan2-13B-chat
    - gpt2
    - ……
- OpenRouter
    - anthropic/claude-2
    - palm-2-chat-bison
    - palm-2-codechat-bison
    - openai/gpt-4
    - ……
- OpenAI
    - text-davinci-003
    - gpt-3.5-turbo
    - gpt-4
    - ……

⚠ CAUTION

- Category: The first-level category of the model provider. For example, `Model` and `Embedding`.
- Type: The second-level category of the model provider. For example, `OpenAI` and `Hugging Face`.
- SecretKey: The secret key of your OpenAI account.

Example

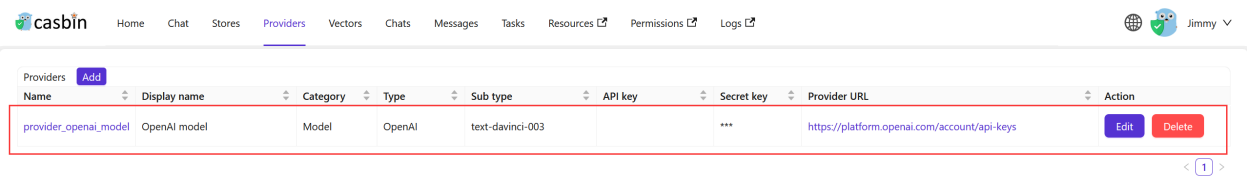
Add an OpenAI model provider



> ⚠ CAUTION
>
> Some models don't support **streaming-output**. Known models that support streaming-output include:
>
> - `gpt-3.5-turbo-0613`

After adding a model provider, you can use it to analyze and process data in Casibase using chatbots, question answering, and other AI capabilities.

Return to the model provider list page:



Now that you've added a model provider, you can use it to analyze and process data in Casibase using chatbots, question answering, and other AI capabilities.

Embedding Providers

Introduction

Embedding is a technique used to represent words and documents as vectors. Embedding providers allow you to analyze and process data within your knowledge base system, making it more intelligent and efficient.

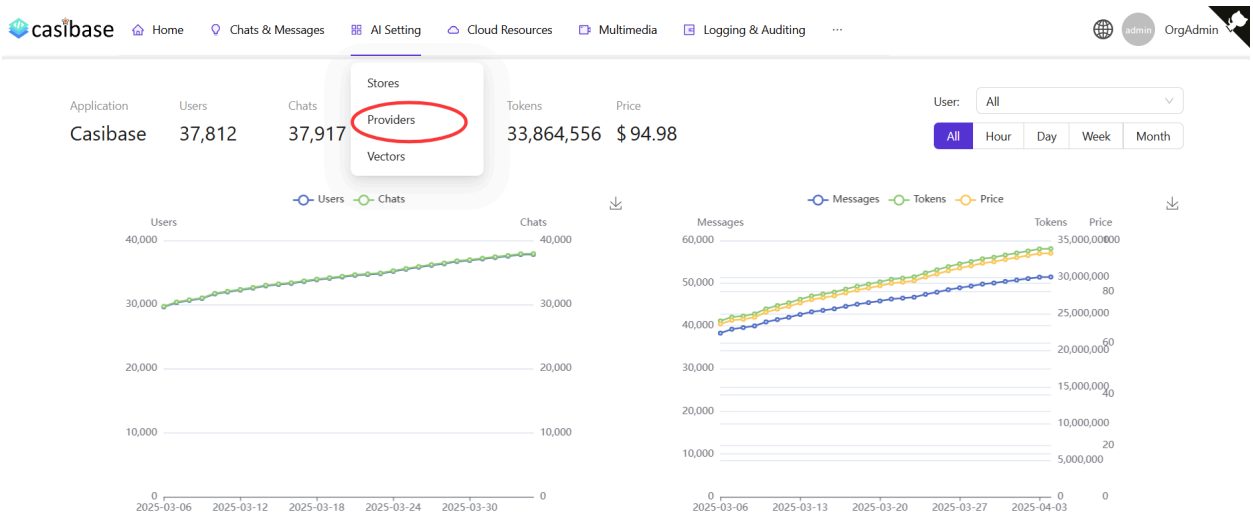
Refer to the Core Concepts section of our previous documentation for more information about embedding.

In Casibase, you can add an embedding provider by following these steps:

Add a New Embedding Provider

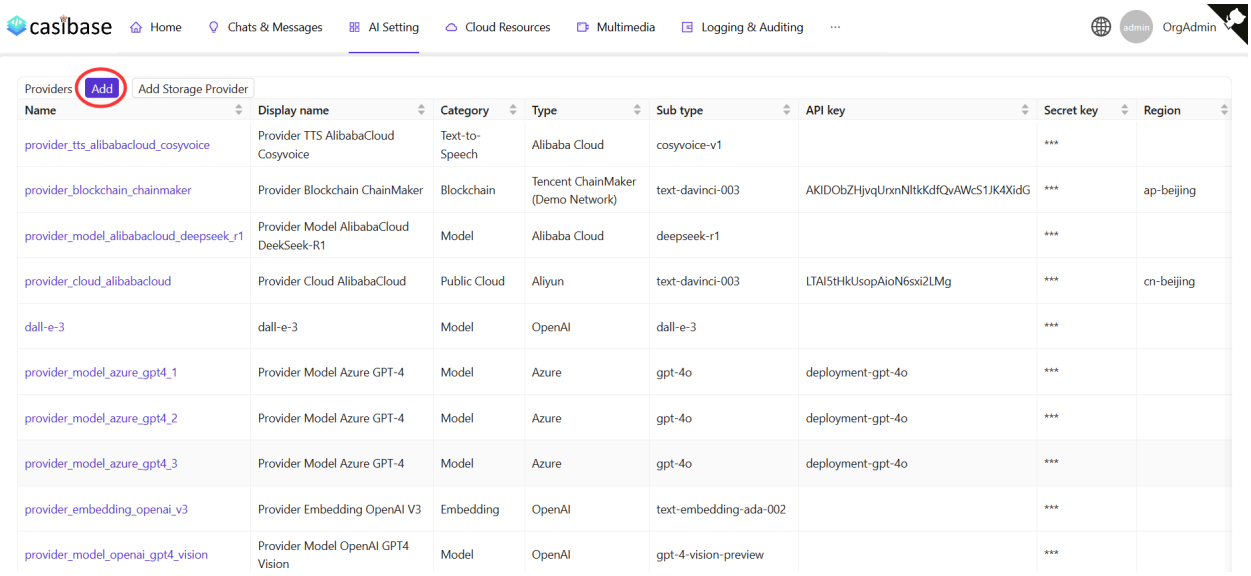
Embedding providers are used to integrate embedding into Casibase. You can add them by following these steps:

Click the `Providers` button on the page.



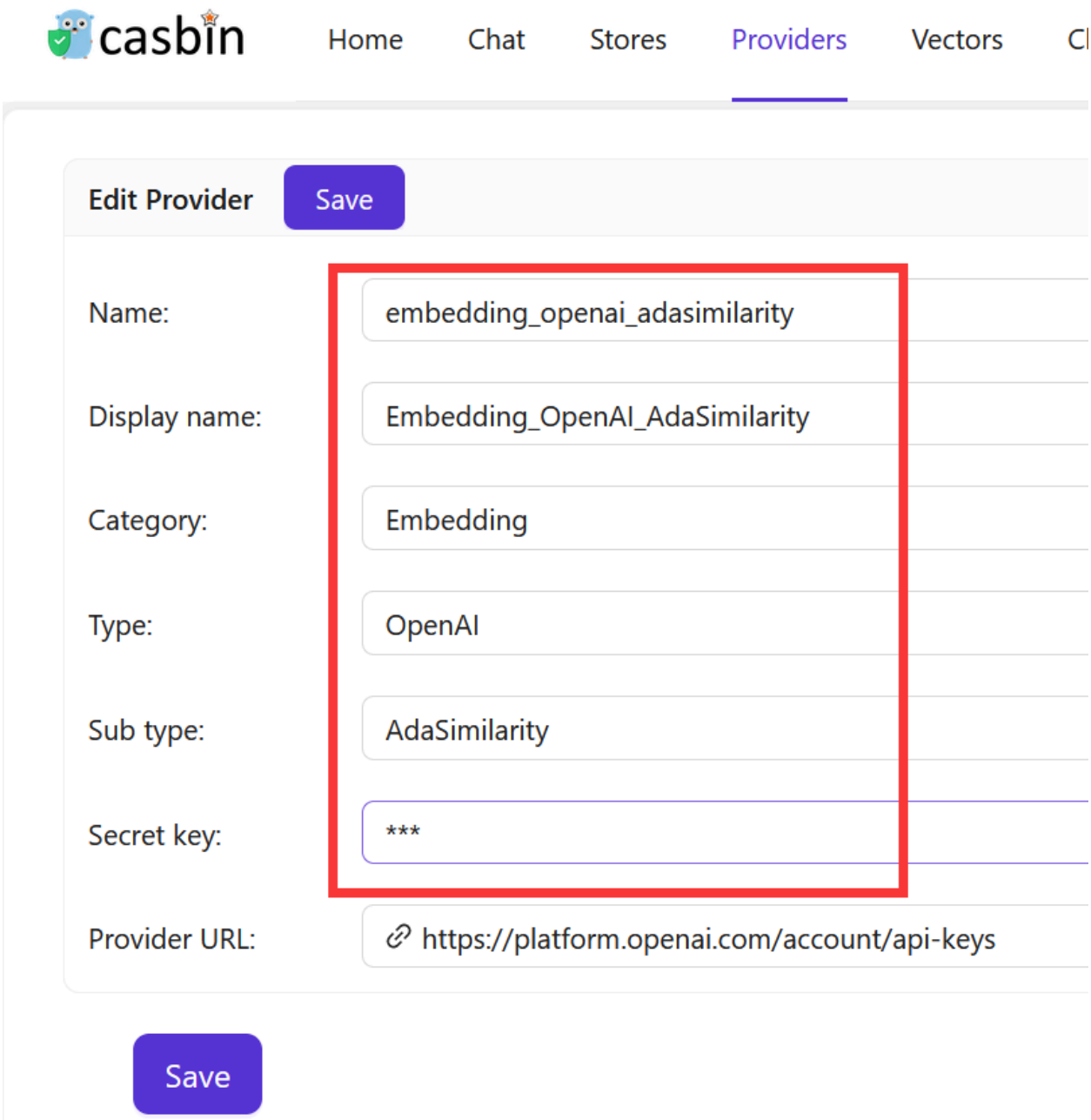
Add an Embedding Provider

Click the `Add` button to add an embedding provider.



Fill in Embedding Provider Details

Fill in the embedding provider details and click the `Save & Exit` button.



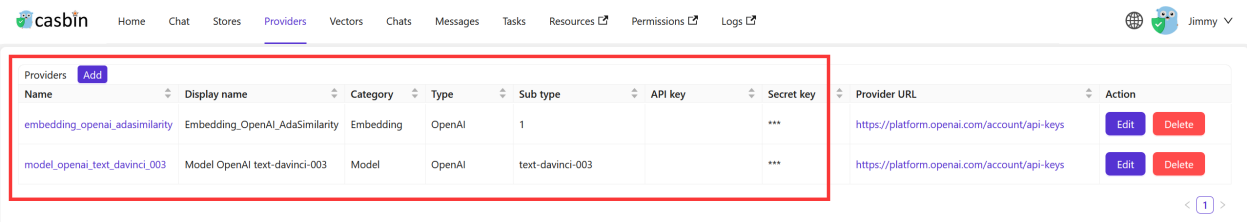
TIP

Casibase supports many embedding providers, including:

- OpenAI

    - AdaSimilarity
    - DavinciSimilarity
    - AdaEmbedding2
    - .....
- Hugging Face
    - sentence-transformers/paraphrase-MiniLM-L6-v2
    - .....

Return providers list page:



Now, you can use the embedding provider to convert text to vectors.

After adding an embedding provider, you can use it to retrieve similar documents in Casibase. For more information, please refer to the Core Concepts section of our previous documentation.

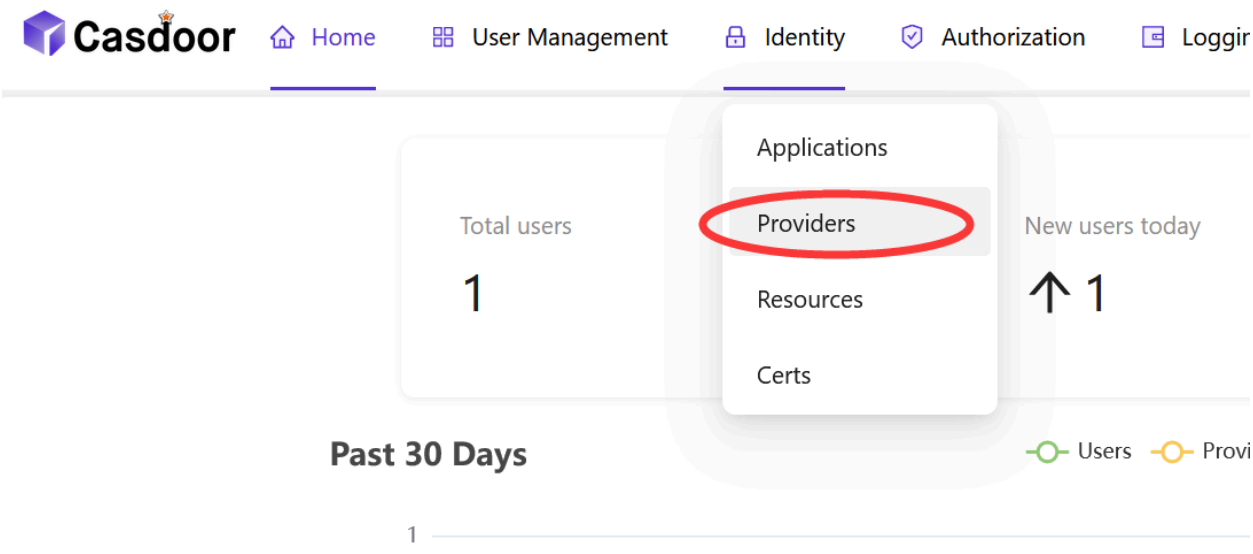
Storage Providers

Introduction

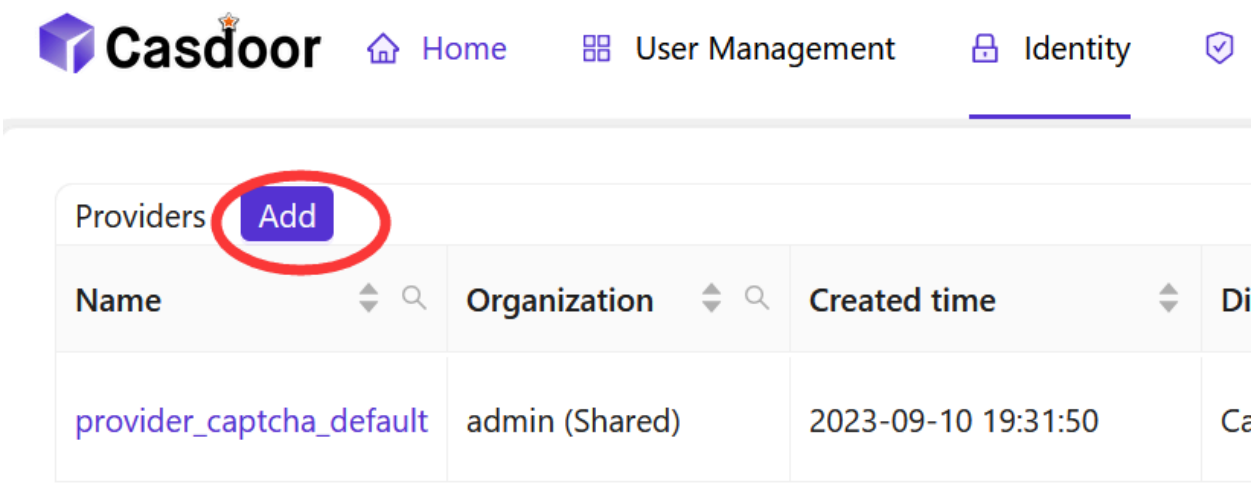
Adding a storage provider to Casibase enables you to efficiently manage and store data, making it an essential component for your knowledge base system.

Add a New Storage Provider

Storage providers are used to store data. They can be added in Casdoor by clicking the `Identity - Providers` button on the home page.

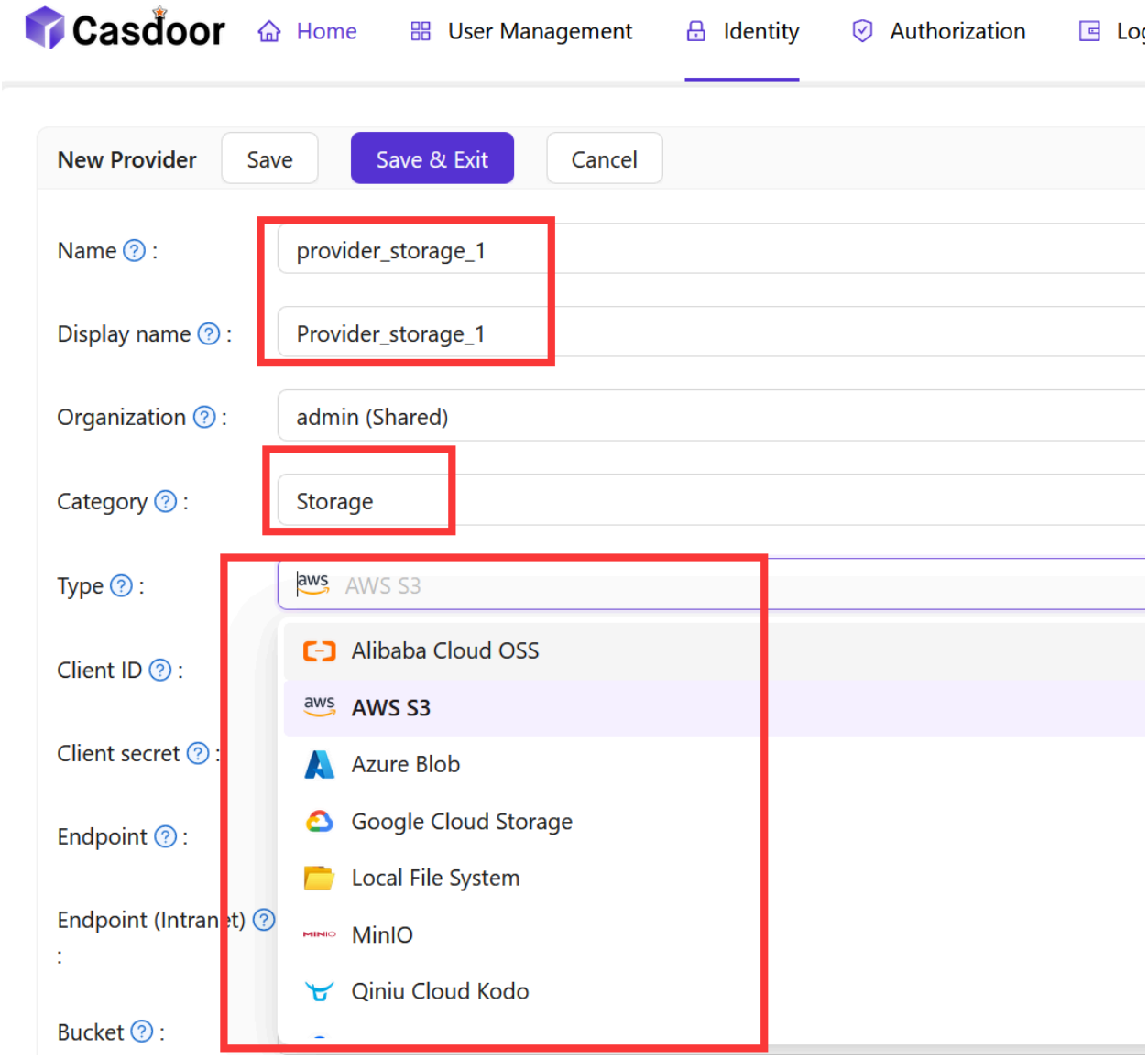


Click the `Add` button to add a storage provider.



Fill in the storage provider information

Fill in the storage provider information and click the `Save & Exit` button.



TIP

Casdoor supports many storage providers, including:

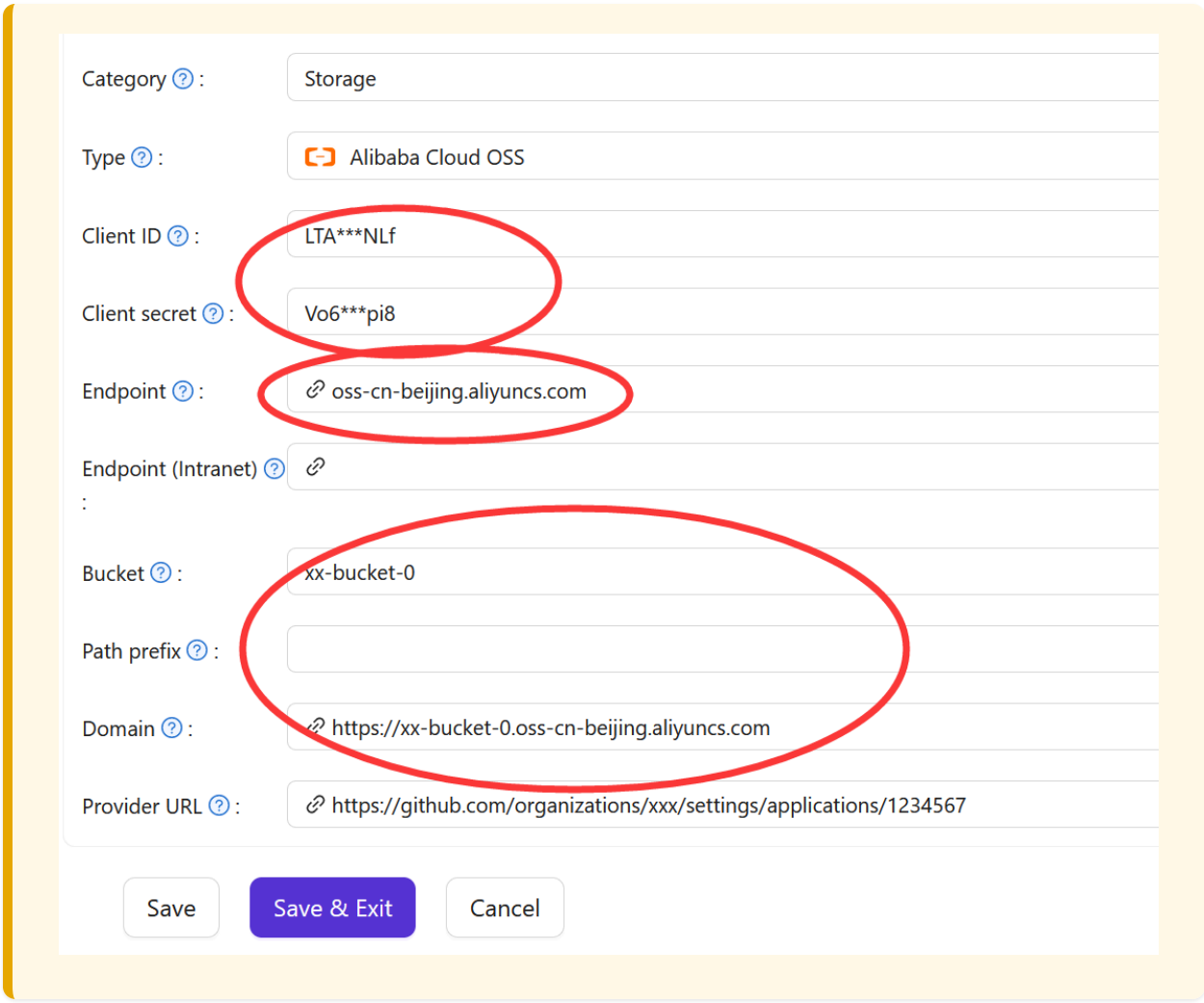
- AWS S3
- Azure Blob
- Google Cloud Storage
- MinIO

- Qiniu Cloud Kodo
- Alibaba Cloud OSS ...

Example

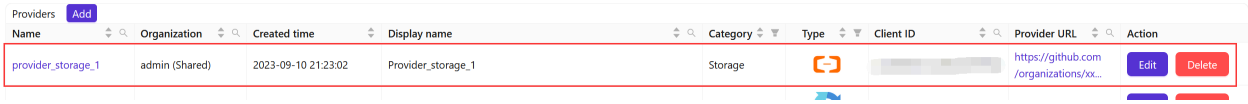
Add an Aliyun OSS storage provider

> ⚠ CAUTION
>
> - Client ID: The AccessKey ID of your Aliyun OSS account.
> - Client Secret: The AccessKey Secret of your Aliyun OSS account.
>
> `****` is the placeholder for your Aliyun OSS account information.



View the storage provider

After adding the storage provider, you can view the storage provider information.



Text-to-Speech Providers

Introduction

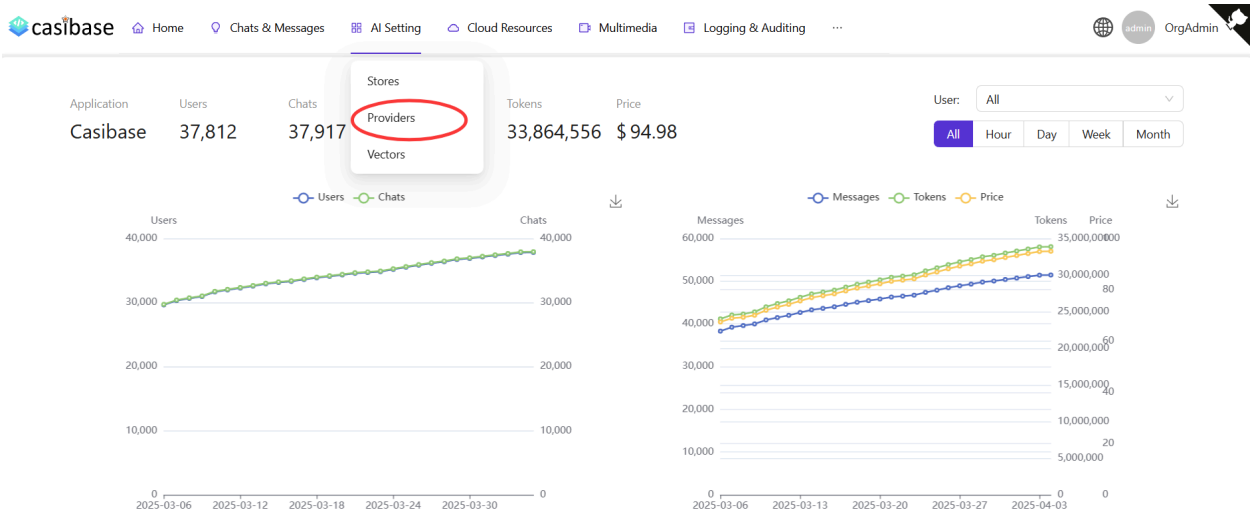
Text-to-Speech (TTS) is a technology that converts text into spoken voice output. TTS providers allow your Casibase applications to communicate with users through synthesized speech, enhancing the user experience and accessibility of your knowledge base system.

In Casibase, integrating a TTS provider enables your AI applications to verbally respond to queries, creating more interactive and engaging user experiences.

Add a New Text-to-Speech Provider

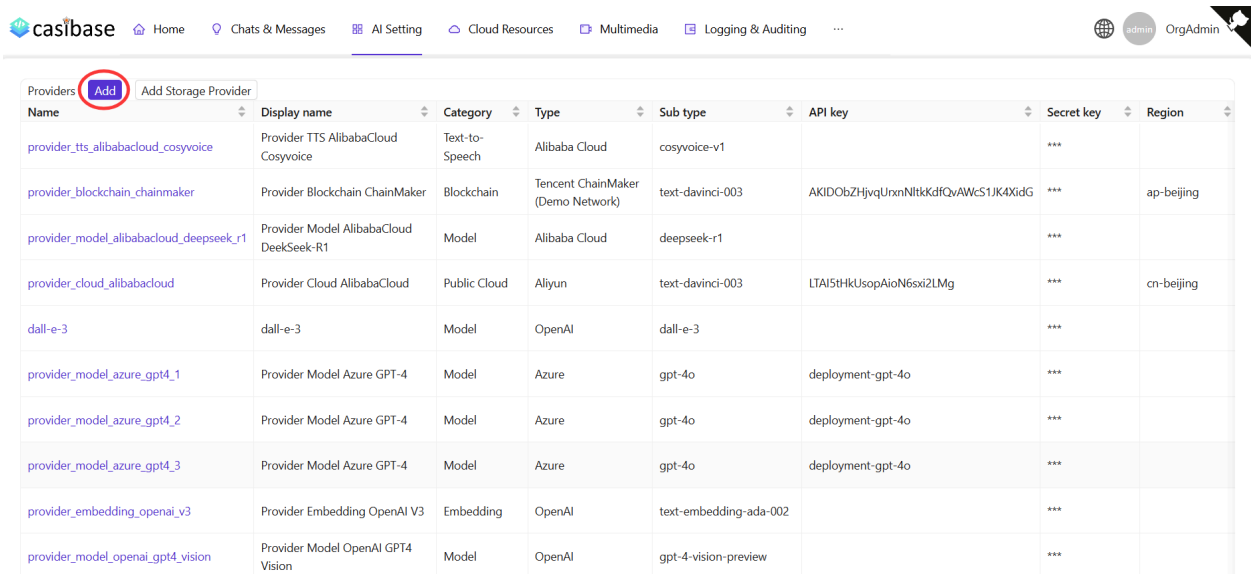
Text-to-Speech providers are used to integrate voice synthesis capabilities into Casibase. You can add them by following these steps:

Click the `Providers` button on the page.



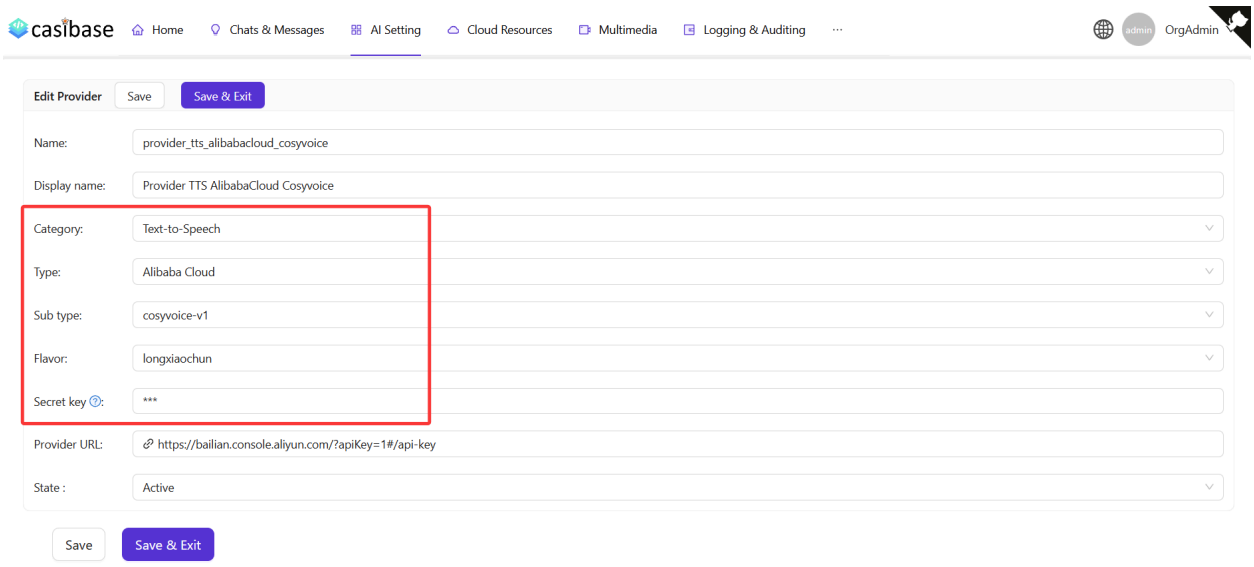
Add a Text-to-Speech Provider

Click the `Add` button to add a Text-to-Speech provider.



Fill in Text-to-Speech Provider Details

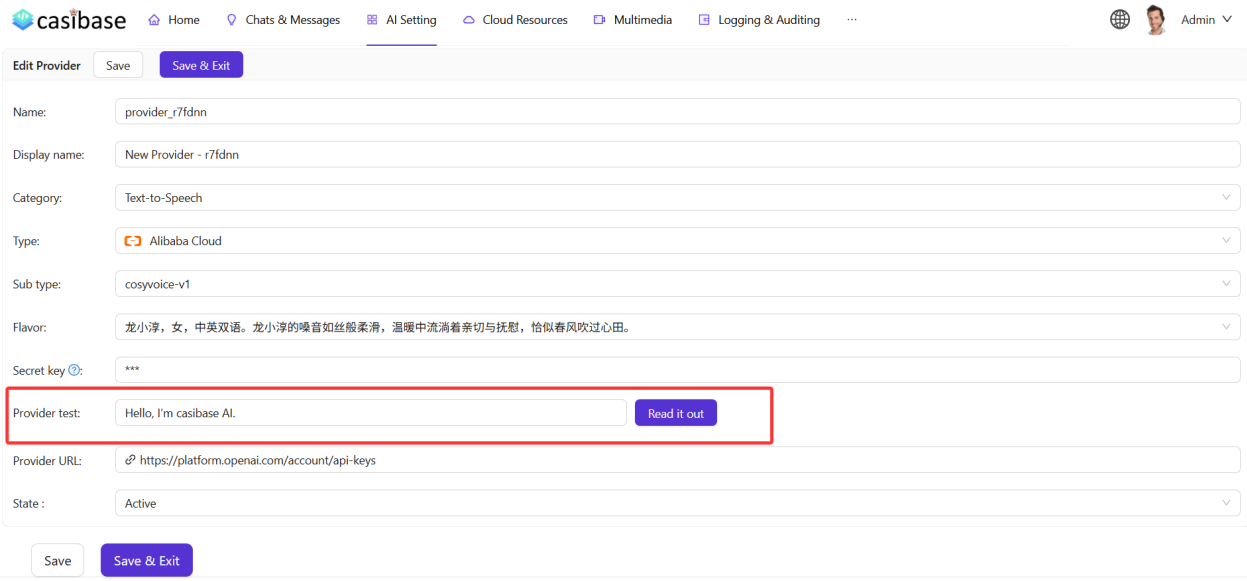
Fill in the embedding provider details and click the `Save & Exit` button.



> TIP
>
> Casibase currently supports the following Text-to-Speech provider:
>
> - Alibaba Cloud
>   - cosyvoice-v1 (with multiple voice options)

Testing Your Text-to-Speech Provider

You can test your TTS provider by clicking the `Read it out` button. This will allow you to enter text and hear the synthesized speech output.



This testing feature allows you to verify your TTS configuration before implementing it in your applications, ensuring the voice quality and settings meet your requirements.

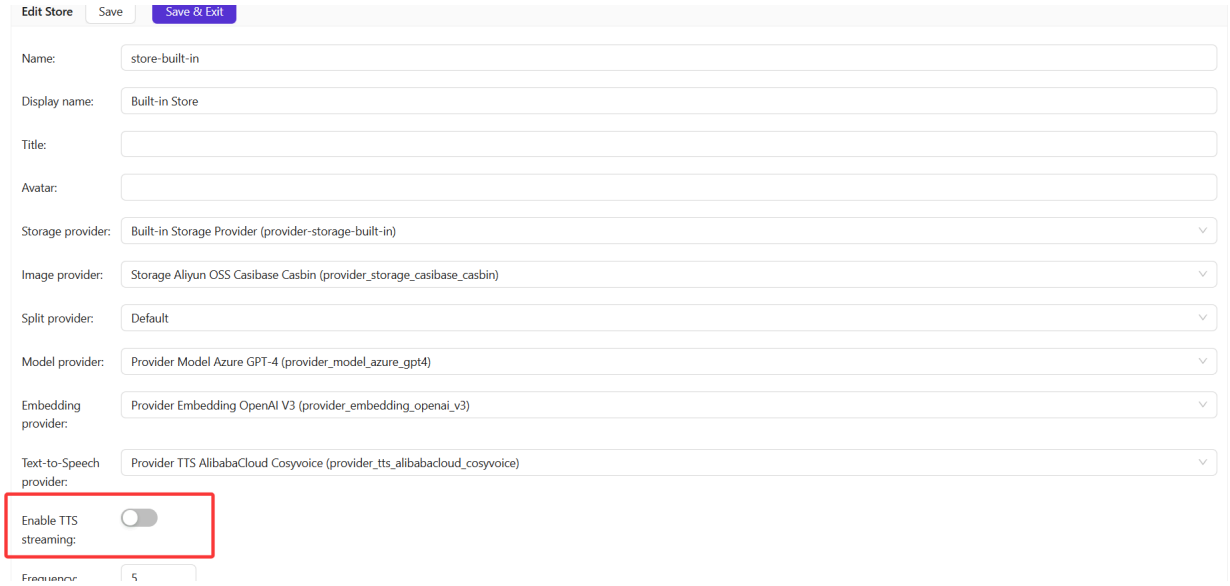
Voice Options for Alibaba Cloud

When using Alibaba Cloud's `cosyvoice-v1`, you can choose from various voice options:

- longwan
- longcheng
- ……

Using Text-to-Speech in Stores

After adding a Text-to-Speech provider, you can select this provider in your store settings and choose whether to enable TTS streaming.



Now, your store can convert text responses to speech, providing a more interactive experience for users.

Speech-to-Text Providers

Introduction

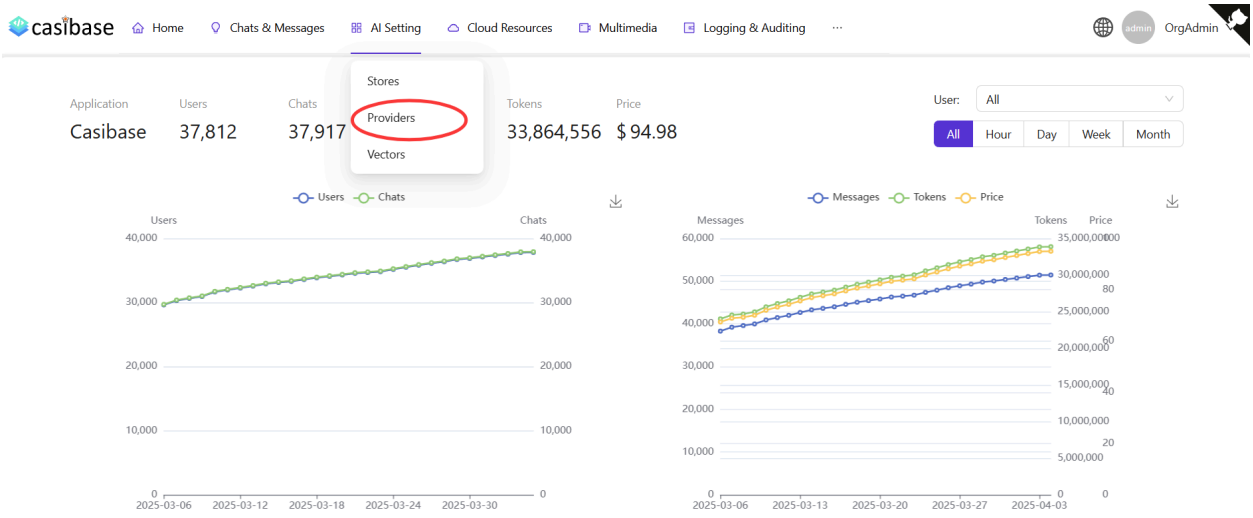
Speech-to-Text (STT) is a technology that converts spoken language into written text. STT providers allow your Casibase applications to understand and process spoken user input, enhancing the user experience and accessibility of your knowledge base system.

In Casibase, integrating an STT provider enables your AI applications to receive and process voice queries, creating more interactive and natural user interactions.

Add a New Speech-to-Text Provider

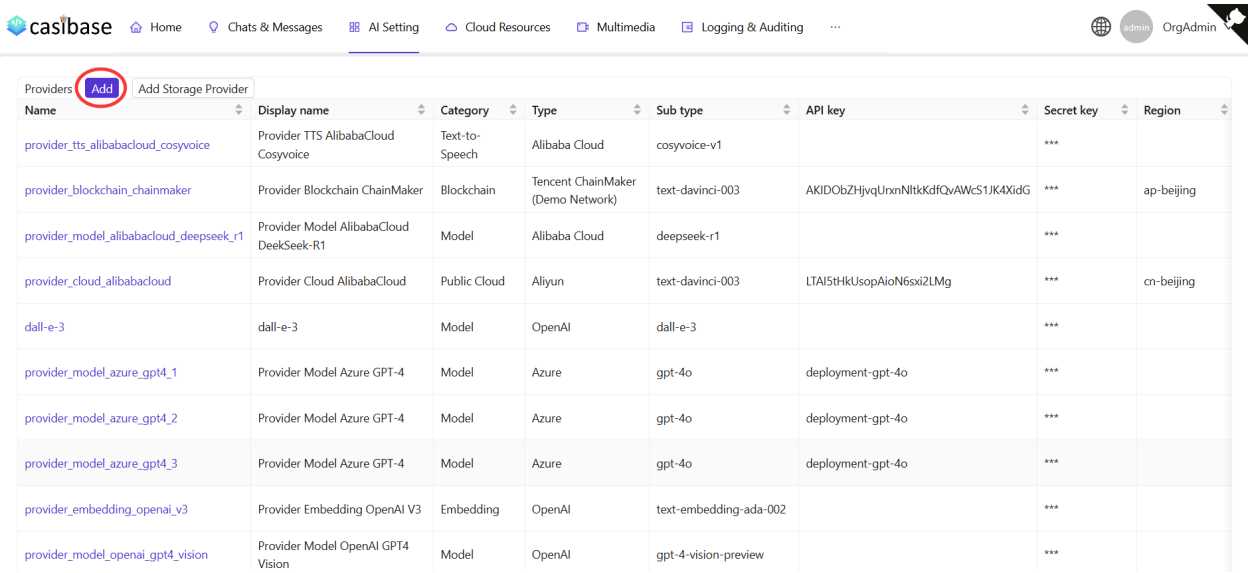
Speech-to-Text providers are used to integrate voice recognition capabilities into Casibase. You can add them by following these steps:

Click the `Providers` button on the page.



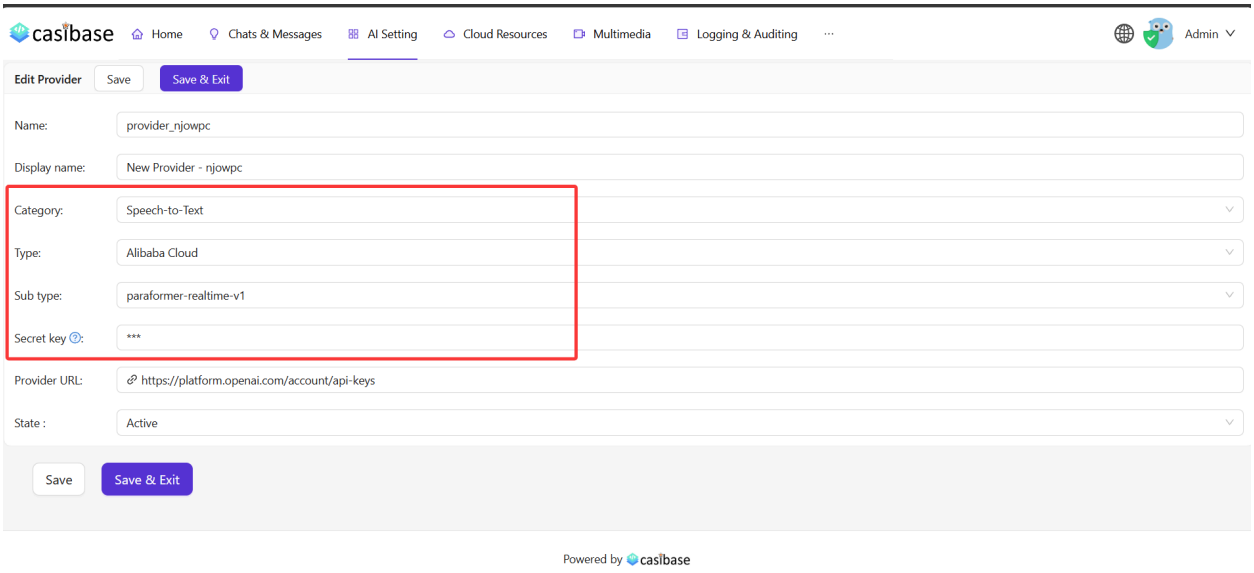
Add a Speech-to-Text Provider

Click the `Add` button to add a Speech-to-Text provider.



Fill in Speech-to-Text Provider Details

Fill in the speech-to-text provider details and click the `Save & Exit` button.

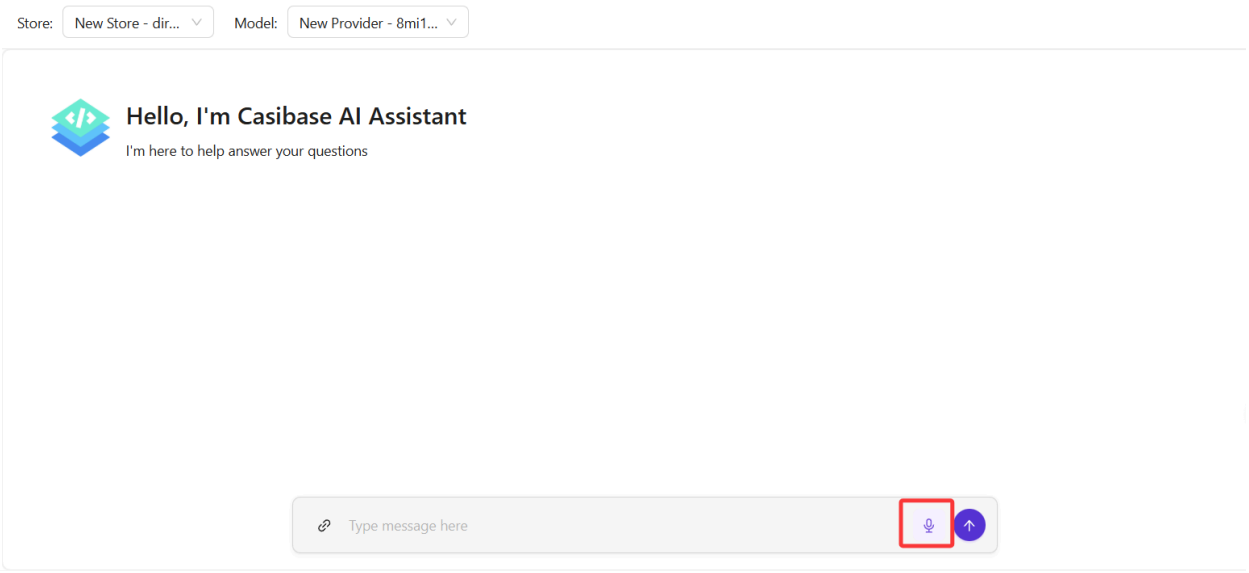


Using Voice Recognition

When you click the voice recognition button in your Casibase application, the following process occurs:

1. The browser will request permission to access your microphone
2. Once granted, the system will begin listening and automatically convert your speech to text
3. After you finish speaking, the recognized text will be automatically sent as a message

This feature enables hands-free interaction with your Casibase applications, making them more accessible and convenient to use.



> TIP
>
> Casibase currently supports the following Speech-to-Text provider:
>
> - Alibaba Cloud
>   - paraformer-realtime-v1

BlockChain Providers

Introduction

Blockchain technology provides an immutable and transparent ledger for data integrity verification. In Casibase, blockchain providers serve as a crucial security layer by uploading data to blockchain networks, ensuring that your knowledge base data cannot be tampered with or altered maliciously.

By leveraging blockchain's decentralized and cryptographic properties, Casibase can guarantee data authenticity and provide audit trails for all data modifications. This is particularly important for organizations that need to maintain data integrity compliance or require verifiable proof of data authenticity.

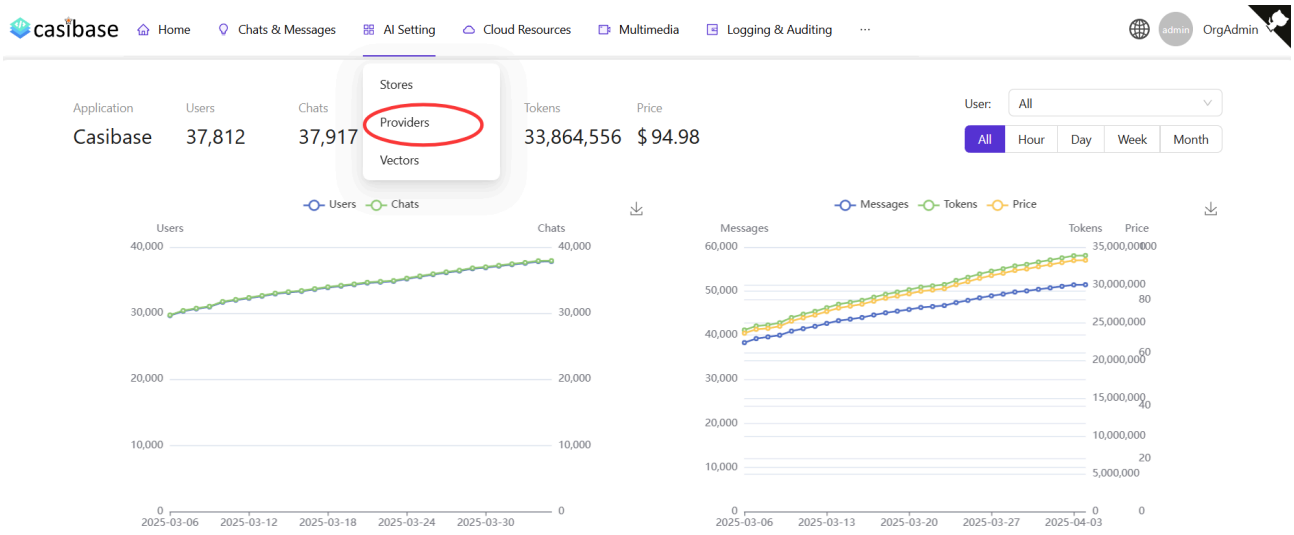
Refer to the Core Concepts section of our previous documentation for more information about providers.

In Casibase, you can add a blockchain provider by following these steps:

Add a New Blockchain Provider

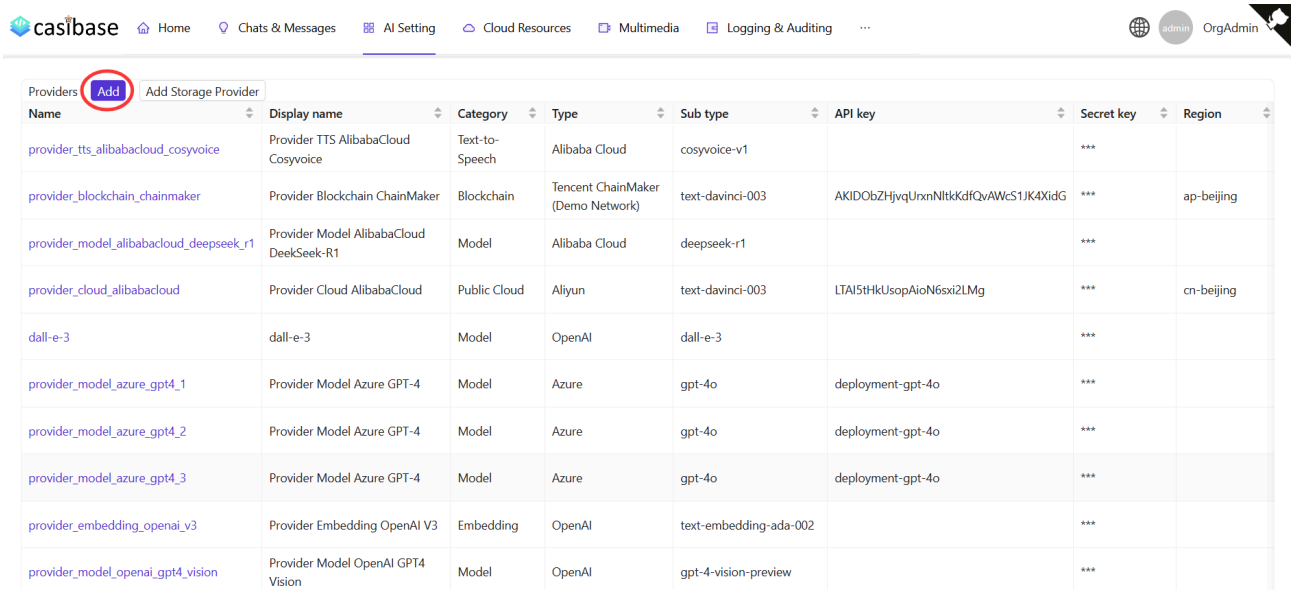
Blockchain providers are used to integrate blockchain data integrity features into Casibase. You can add them by following these steps:

Click the `Providers` button on the page.



Add a Blockchain Provider

Click the `Add` button to add a blockchain provider.



Fill in Blockchain Provider Details

Fill in the required configuration according to the blockchain you use, then click the `Save & Exit` button to save.

More information about the configuration can be found below:

Chainmaker Configuration

ChainMaker is a high-performance, enterprise-grade blockchain platform developed under the leadership of China. It is designed to provide secure, controllable, and sc...

Ethereum Configuration

Ethereum is a decentralized blockchain platform that enables smart contracts and decentralized applications (dApps). It is one of the most popular blockchain platforms...

TIP

Casibase supports various blockchain networks for data integrity verification, including:

- ChainMaker
  - ChainMaker
  - Tencent Chainmaker
- Ethereum
  - Private networks (compatible with Ethereum JSON-RPC)
    - Geth
    - Ganache
    - Other Ethereum JSON-RPC compatible chains

Now, you can use the blockchain provider to ensure data integrity and prevent tampering.

After adding a blockchain provider, you can use it in Casibase to create immutable data records. The data records will be committed to the blockchain, providing proof of authenticity and preventing any unauthorized tampering.

Chainmaker Configuration

ChainMaker is a high-performance, enterprise-grade blockchain platform developed under the leadership of China. It is designed to provide secure, controllable, and scalable blockchain infrastructure for industries such as finance, government, and supply chain. ChainMaker supports multiple consensus mechanisms, smart contracts, privacy protection, and other features, meeting the needs of large-scale commercial applications. Its open-source, modular architecture allows developers to flexibly customize and extend functionalities according to actual business scenarios.

In this chapter, you will learn how to configure and use ChainMaker, including setting up storage providers and other operations, to help you quickly get started and apply the features of the ChainMaker platform.

1. Configuration field description

When configuring a ChainMaker provider in Casibase, you need to fill in several key fields. Each field has a specific meaning and is required for the correct integration with the ChainMaker blockchain. The following list explains the purpose of each field:

- `Name`: The unique identifier for this blockchain provider.
- `Display name`: The display name shown in the UI for this provider.
- `Category`: The type of service, here it should be `Blockchain`.
- `Type`: The blockchain type, here it should be `ChainMaker`.
- `orgId`: The organization ID in the ChainMaker network.
- `ChainId`: The chain ID of the ChainMaker blockchain.
- `AuthType`: The account mode. Currently, only `permissionedwithcert` is

supported.

- `User key`: The user's private key for authentication in the ChainMaker server.
- `User cert`: The user's certificate for authentication in the ChainMaker server.
- `Sign key`: The user's private key for signing transactions in the ChainMaker server.
- `Sign cert`: The user's certificate for signing transactions in the ChainMaker server.
- `Node address`: The address of the ChainMaker node to connect to.
- `Contract name`: The name of the smart contract to interact with.
- `Invoke method`: The method name to invoke on the contract.
- `Browser URL`: The URL for viewing the blockchain in a browser.
- `Chainmaker endpoint`: The API endpoint for the ChainMaker service. See: https://github.com/casibase/chainmaker-server

Please make sure to fill in each field accurately according to your ChainMaker deployment information. This will ensure that Casibase can successfully connect and interact with your ChainMaker blockchain.

Next, we will use "Deploying ChainMaker via the Management Console" as an example for our introduction. If you have not yet deployed ChainMaker, please follow the ChainMaker documentation for deployment.

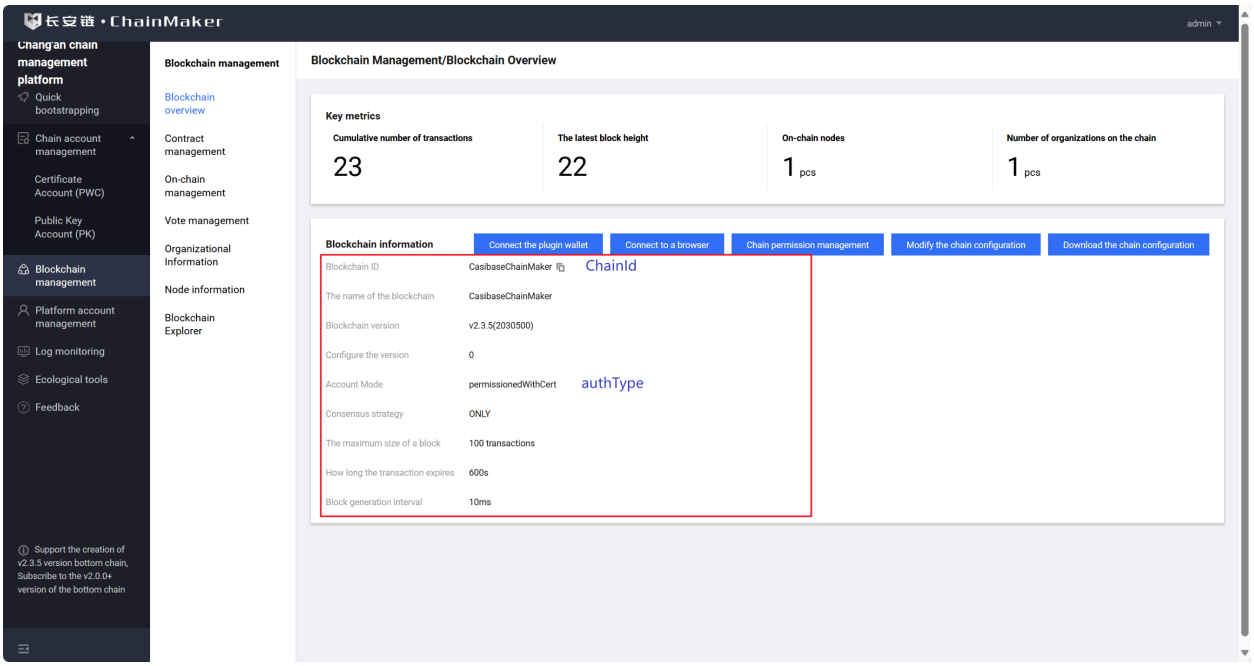
2. Configure ChainMaker

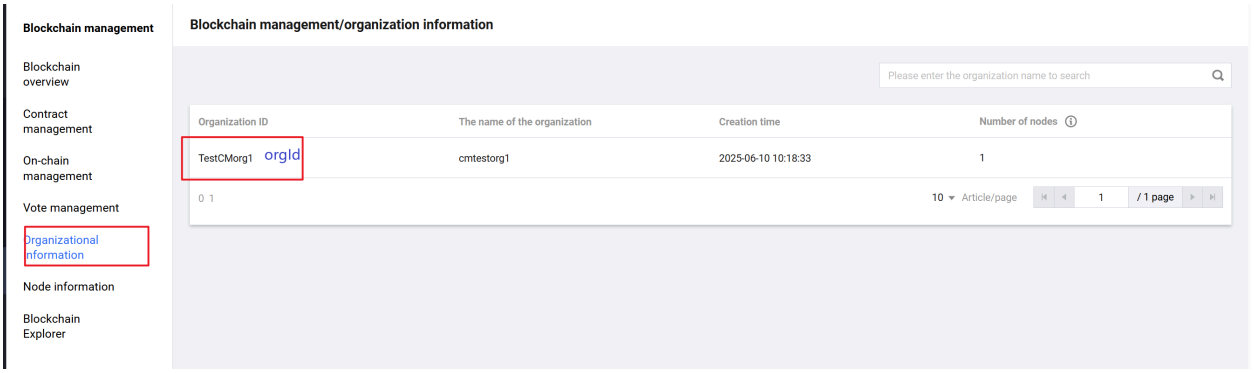
2.1 chainId, orgId and authType configuration

Obtaining Blockchain Information from the Web Panel

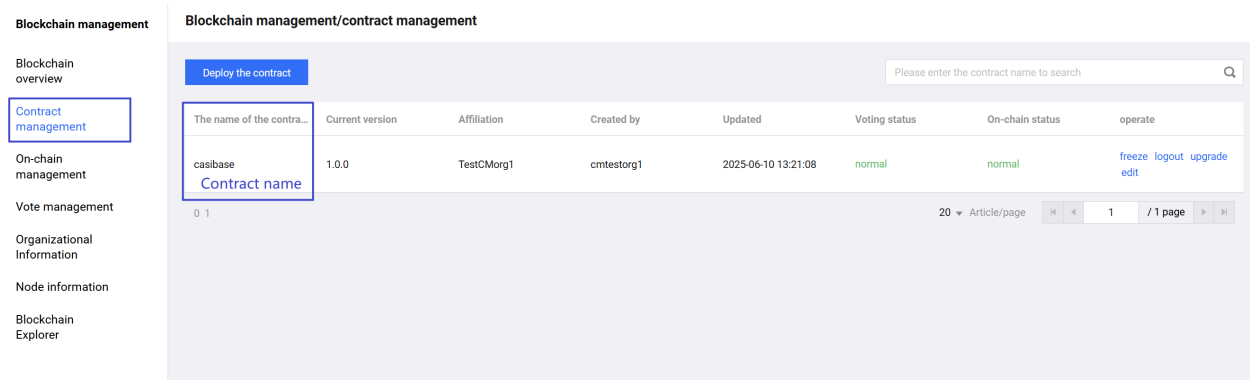
To retrieve blockchain information such as `chainId`, `orgId`, and `authType`, log in

to the **ChainMaker Management Console** (Web panel). Navigate to the relevant **blockchain management** section, where these configuration details are displayed. Copy the required values and use them when configuring the ChainMaker provider in Casibase.

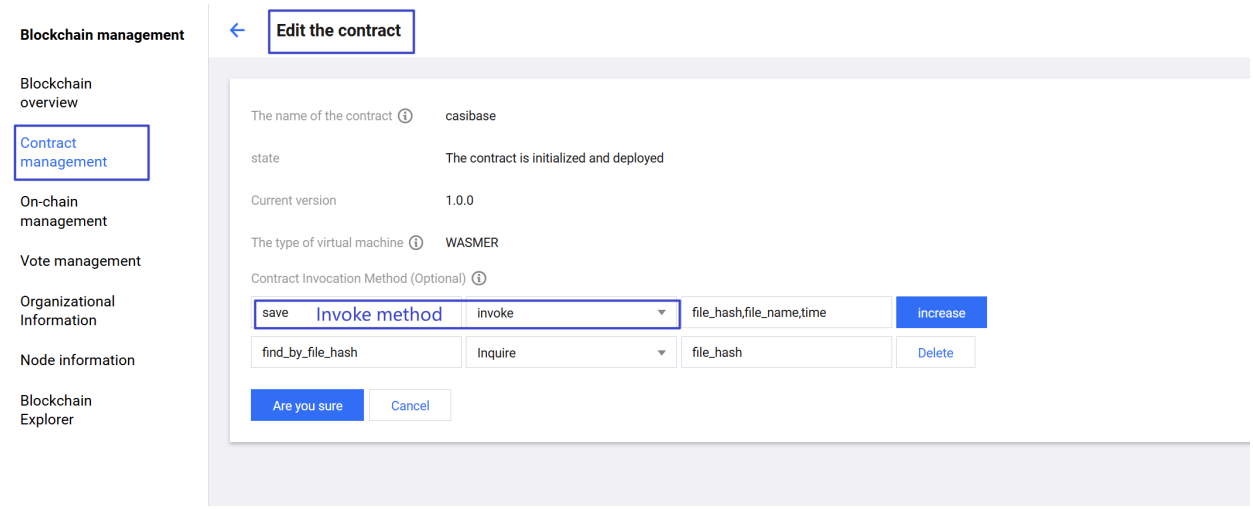




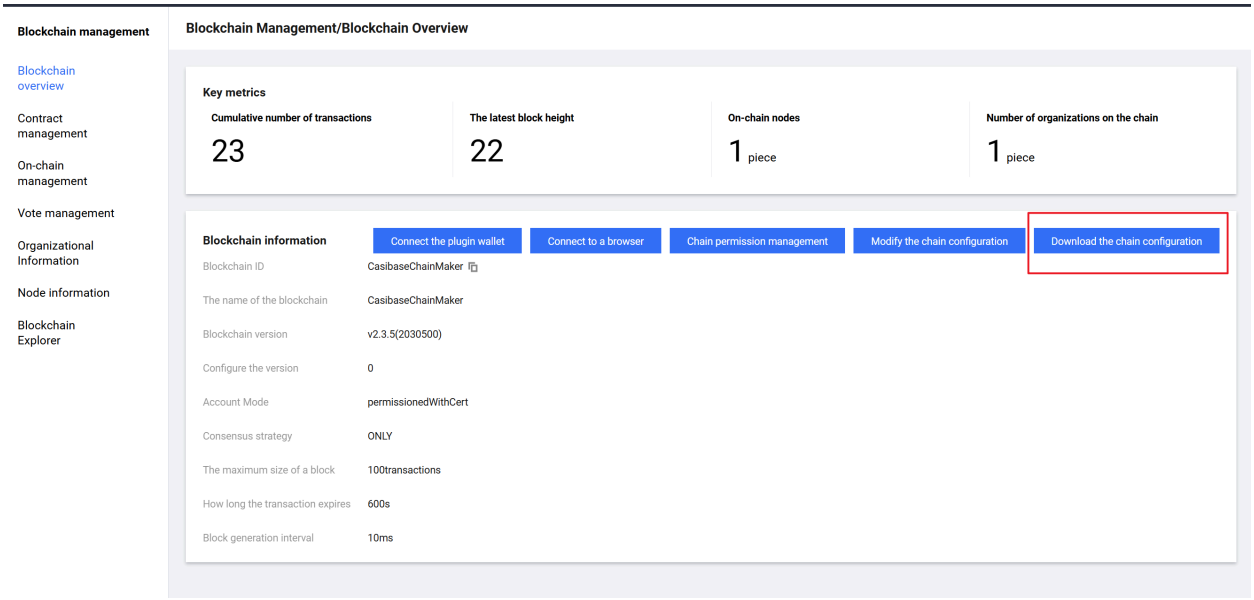
2.2 Contract name and Invoke method configuration



Click the edit button in the "operate" column:



2.3 key, certificate, and nodeAddr configuration

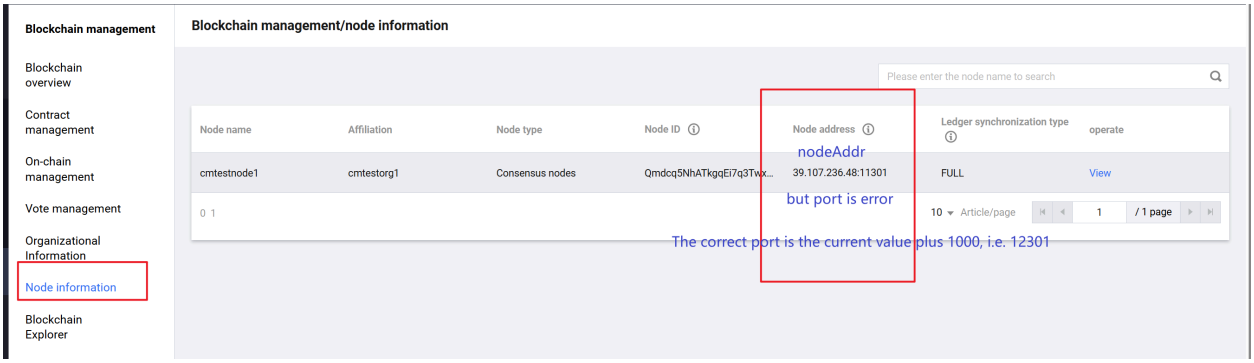


Click the "download the chain configuration" button and extract the archive:

2.3.1 node addr

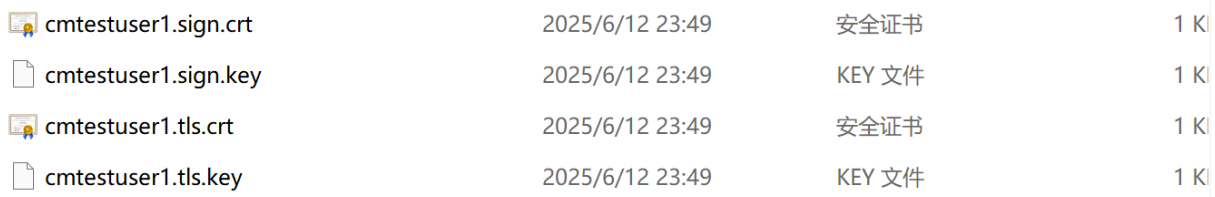
You can find the node_addr configuration in ~\CasibaseChainMaker\sdk_configs\sdk_config.yml.

Alternatively, you can check nodeAddr elsewhere, but note that the displayed port is the p2p port. The one we use should be the rpc port, which is the p2p port + 1000 by default.



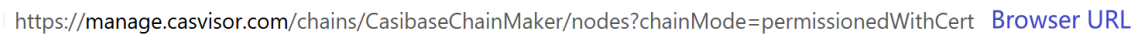
2.3.2 user certificate and key

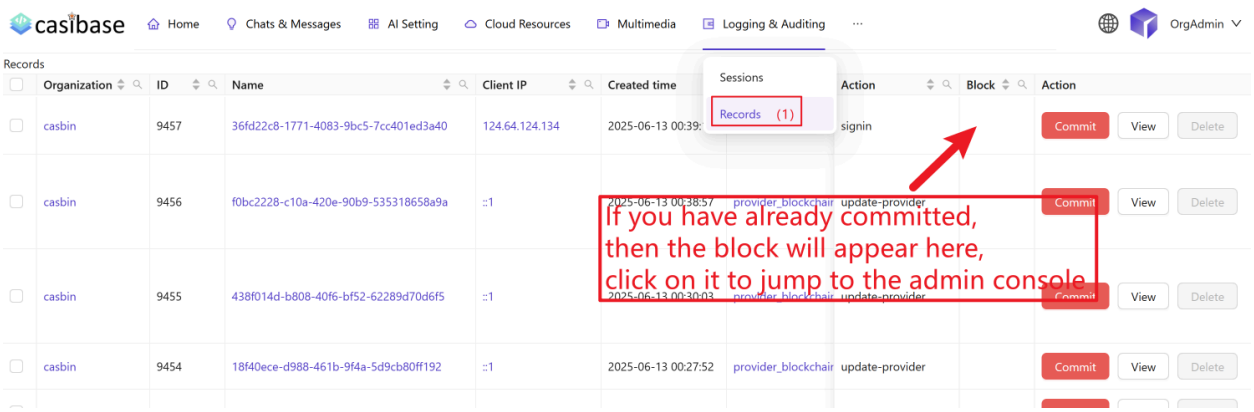
You can find the corresponding user certificate and key in ~\CasibaseChainMaker\sdk_configs\crypto-config\TestCMorg1\user\cmtestuser1. Fill in the provider fields accordingly.



2.4 Browser URL and ChainMaker endpoint

The Browser URL refers to the URL of the ChainMaker management console, which allows for quick access from Casibase in the future:





The ChainMaker endpoint refers to the IP and port of the ChainMaker server that Casibase needs to connect to, for example: 127.0.0.1:13900

Ethereum Configuration

Ethereum is a decentralized blockchain platform that enables smart contracts and decentralized applications (dApps). It is one of the most popular blockchain platforms, supporting a wide range of applications from DeFi to NFTs. Ethereum uses a proof-of-stake consensus mechanism and provides a robust ecosystem for developers to build and deploy smart contracts.

> INFO
>
> In this chapter, you will learn how to configure and use Ethereum, including setting up blockchain providers and other operations, to help you quickly get started and apply the features of the Ethereum platform.

1. Configuration field description

When configuring an Ethereum provider in Casibase, you need to fill in several key fields. Each field has a specific meaning and is required for the correct integration with the Ethereum blockchain.

Field Descriptions:

- **`Name`**: The unique identifier for this blockchain provider.
- **`Display name`**: The display name shown in the UI for this provider.
- **`Category`**: The type of service, here it should be `Blockchain`.
- **`Type`**: The blockchain type, here it should be `Ethereum`.
- **`Private key`**: The private key of the Ethereum account used for signing transactions.
- **`Contract Address`**: The address of the smart contract to interact with on the

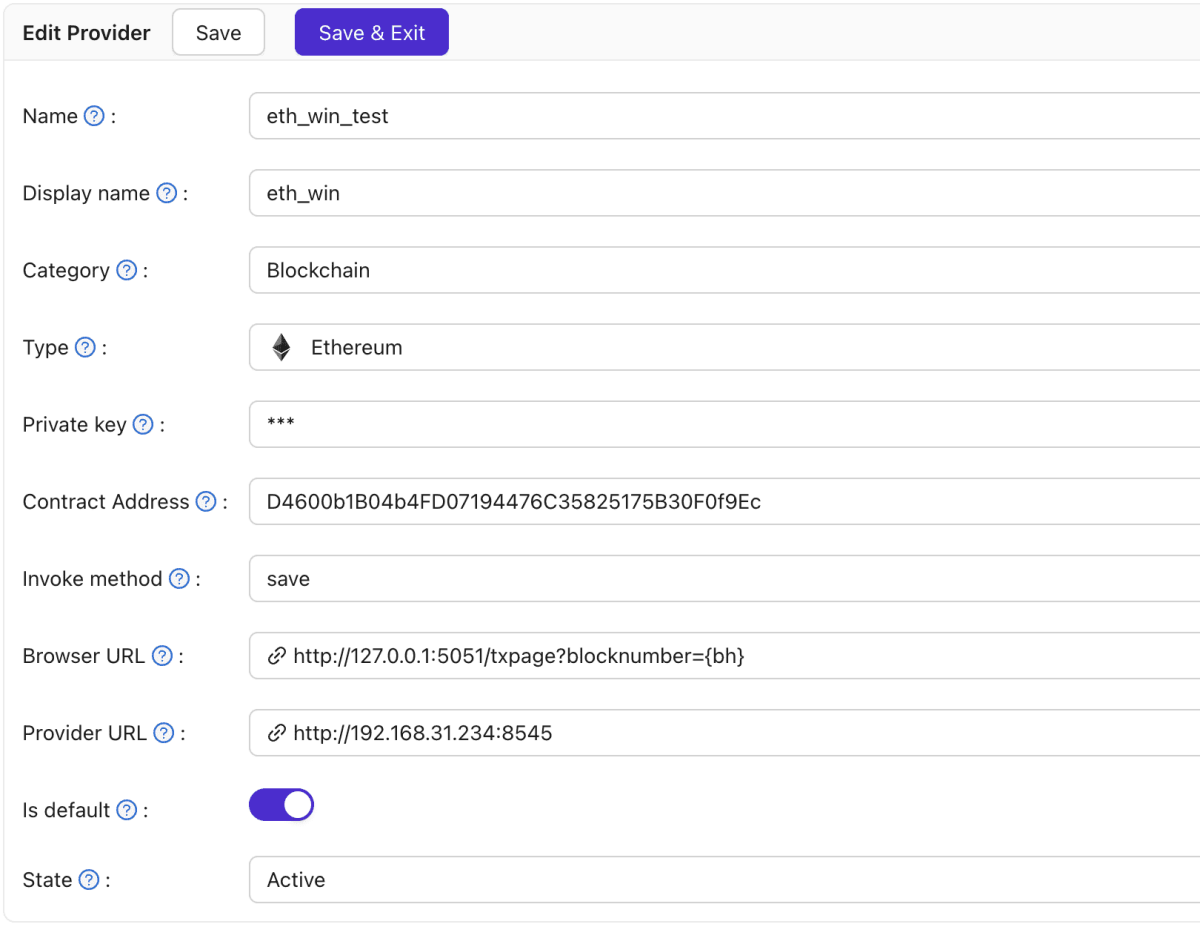
Ethereum blockchain.

- **`Invoke method`**: The method name to invoke on the smart contract.
- **`Browser URL`**: The URL for viewing the blockchain in a browser, with block number template support.
  - Format: `http://127.0.0.1:5051/txpage?blocknumber={bh}` where `{bh}` will be replaced with the actual block number when visiting the block.
- **`Provider URL`**: The JSON-RPC endpoint URL for connecting to the Ethereum network (e.g., Geth, Ganache, or other node).

Please make sure to fill in each field accurately according to your Ethereum deployment information. This will ensure that Casibase can successfully connect and interact with your Ethereum blockchain.

2. Configure Ethereum

Example



2.1 Provider URL Configuration

The Provider URL is the JSON-RPC endpoint that Casibase will use to communicate with the Ethereum network. **This is the first and most important configuration** as it establishes the connection to your Ethereum network.

Example Provider URL

```
http://127.0.0.1:8545
```

You can use:

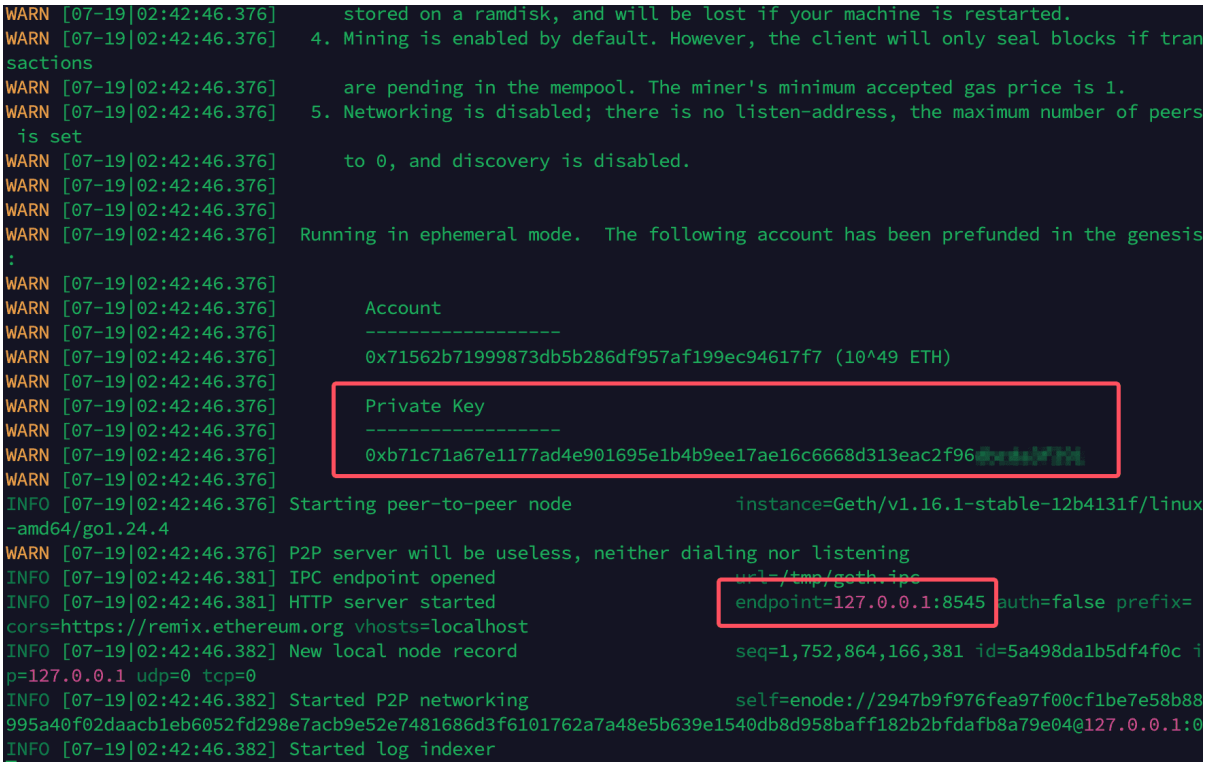
- geth: A popular Ethereum client that provides a JSON-RPC interface.
- ganache: A personal blockchain for Ethereum development that can be used for testing and development purposes.
- other: Ethereum JSON-RPC compatible chains. Any other Ethereum-compatible chain that supports the JSON-RPC interface.

Example: Geth Dev Mode

To quickly experiment with Ethereum using Geth, you can start Geth in developer mode. This mode launches a local Ethereum node with instant mining and pre-funded accounts, making it ideal for testing and development.

```
geth --dev --http --http.api eth,web3,net --http.corsdomain
"https://remix.ethereum.org"
```

This command starts a local Ethereum node with HTTP JSON-RPC enabled and sets the CORS domain to allow cross-origin requests from `https://remix.ethereum.org`. This configuration is suitable for online contract deployment and interaction using Remix web-based tools. You can use the default account (private key can be found in the `geth` console at first launch) and the endpoint (`http://127.0.0.1:8545`) as your Provider URL in Casibase for immediate testing and development.



Understanding the Console Output:

When you run Geth in dev mode, the console will display important information as shown in the image above:

- **Private Key**: The console shows the private key of the pre-funded account that you can use for testing. This key is automatically generated and displayed in the console output.
- **HTTP Endpoint**: The console confirms that the HTTP JSON-RPC server is running on `http://127.0.0.1:8545`. This is the endpoint address you should use as your Provider URL in Casibase.
- **Account Address**: The corresponding Ethereum address for the generated private key is also displayed.

Important: Copy and save these values immediately as they are essential for configuring your Casibase provider. The private key will be needed for the `Private key` field, and the HTTP endpoint will be your `Provider URL`.

For more details, see the Geth Dev Mode documentation.

2.2 Private key Configuration

The private key is essential for signing transactions on the Ethereum blockchain.

> NOTE
>
> The private key should be provided without the hexadecimal prefix `0x`.

Example Private Key

```
# Example private key in geth dev mode (without 0x prefix)
b71c71a67e1177ad4e901695e1b4b9ee17ae16c6668d313eac2f96dbcda3f291
```

> DANGER
>
> This is just an example private key for demonstration purposes. **Never use this key in production or for real funds!**

You can obtain your private key from various sources:

- **Initially generated by Geth**: When you start Geth in `dev` mode, it generates a pre-funded account with a private key displayed in the console.
- **Ethereum Clients**: Generate a new account using Ethereum clients like Geth.

> INFO
>
> Casibase will use *** to replace the private key on the frontend after the submission.

2.3 Invoke Method and Contract Address Configuration

Invoke Method Configuration

The invoke method is the specific function name in the smart contract that you want to call.

In Casibase, **your smart contract should implement specific methods to ensure compatibility**:

- **`save`**: This method is used to store data in the contract. It should accept parameters as a tuple (struct).

  ```
  struct DataItem {
      string key;
      string field;
      string value;
  }

  // Define event, returns key, field, value in order
  event DataSaved(string key, string field, string value);

  // Save struct data and emit an event for tracking
  function save(DataItem memory _data) public {
      emit DataSaved(_data.key, _data.field, _data.value);
  }
  ```

  Method name `save` can be customized, but it should accept a struct as an argument.

You can refer to the Example to see how to implement the `save` method in your smart contract.

Make sure your contract includes these methods to enable seamless integration with Casibase.

Contract Address Configuration

```
# Example contract address (without 0x prefix)
c36fED2CE2E1Bb14b330465f4498D4892C8ee194
```

The contract address is the deployed smart contract's address on the Ethereum blockchain. You can obtain the contract address after deploying a smart contract.

Example for Contract Deployment Reference

To deploy a smart contract on Ethereum, you can refer to the Casibase/contract-storage-eth. This repository provides sample Solidity contracts and deployment scripts using Go and Remix.

Getting Started with the Repository:

1. **Get Example Code**: Clone or download the repository to access sample Solidity contracts and deployment scripts.
2. **Contract Compilation**: Pre-compiled contract artifacts (ABI and bytecode) are available in the releases section of the repository.
3. **Setup for Go Script Deployment**: If using the Go deployment script, download the contract artifacts from releases and place them in the `build/` folder within the Go script's working directory.
4. **Deployment Options**: You can deploy the contract using either:
    - **Go Script**: Use the provided Go deployment script in the repository for programmatic deployment (requires contract artifacts in `build/` folder)
    - **Remix IDE**: Deploy contracts online using Remix with the contract source code

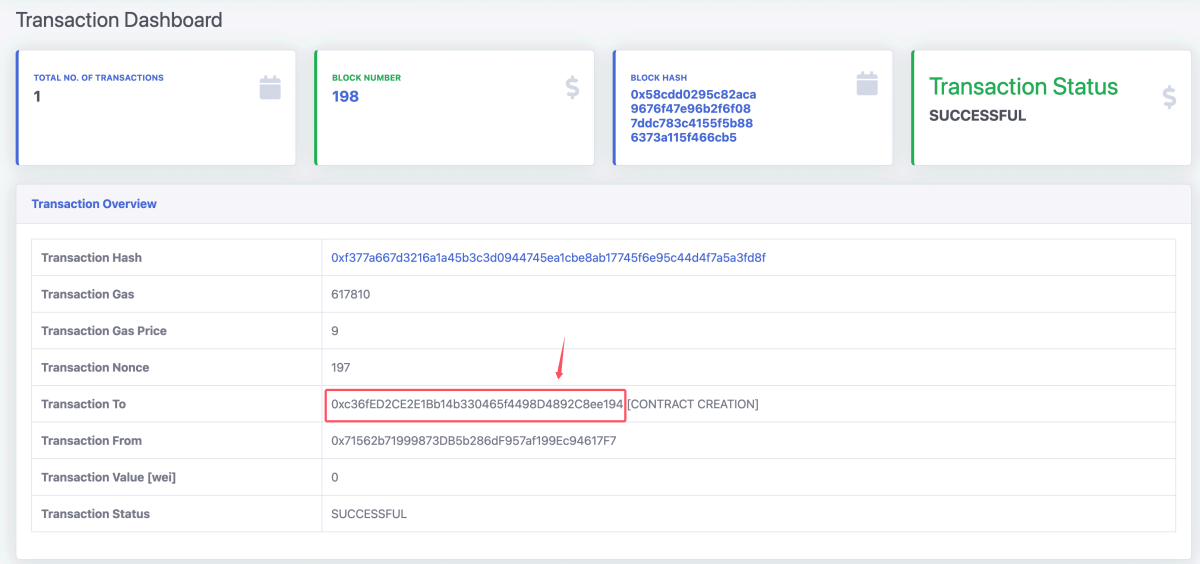
After deployment using either method, you can obtain the contract address from the deployment output.

Use the go script in the reference to deploy the contract

```
$ go run deploy.go
Starting contract deployment...
Connected to Ethereum node: http://192.168.31.234:8545
Deploying from address: 0x71562b71999873DB5b286dF957af199Ec94617F7
Loaded bytecode from: build/SaveContract.bin
Loaded ABI from: build/SaveContract.abi
Gas price: 9 wei
Gas limit: 0
Deploying contract...
Transaction sent: 0xf377a667d3216a1a45b3c3d0944745ea1cbe8ab17745f6e95c44d4f7a5a3fd8f
Contract address: 0xc36fED2CE2E1Bb14b330465f4498D4892C8ee194
Waiting for transaction confirmation...
Contract deployed successfully!
Gas used: 611787
Block number: 198

Running contract test...
Calling save function with: key=test_key_123, field=test_field, value=test_value_456
Save transaction: 0xb060530b6de01bd0537595b86dd0ebcac9007ebab3f24c096c42857ac6fdb3f2
Save function called successfully!
Retrieved data - Key: test_key_123, Field: test_field, Value: test_value_456
Log data - Key: test_key_123, Field: test_field, Value: test_value_456
```

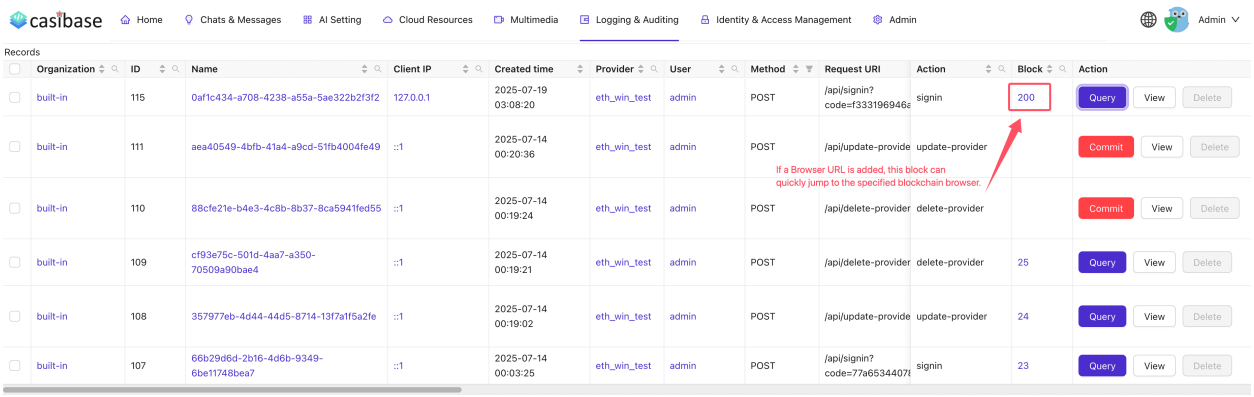
Or you can use the block explorer to find the contract address.



2.4 Browser URL Configuration

The Browser URL lets you view specific blockchain blocks and transactions in a web browser. By using a template with the `{bh}` placeholder, Casibase can automatically redirect you to the corresponding block details in your chosen blockchain explorer.

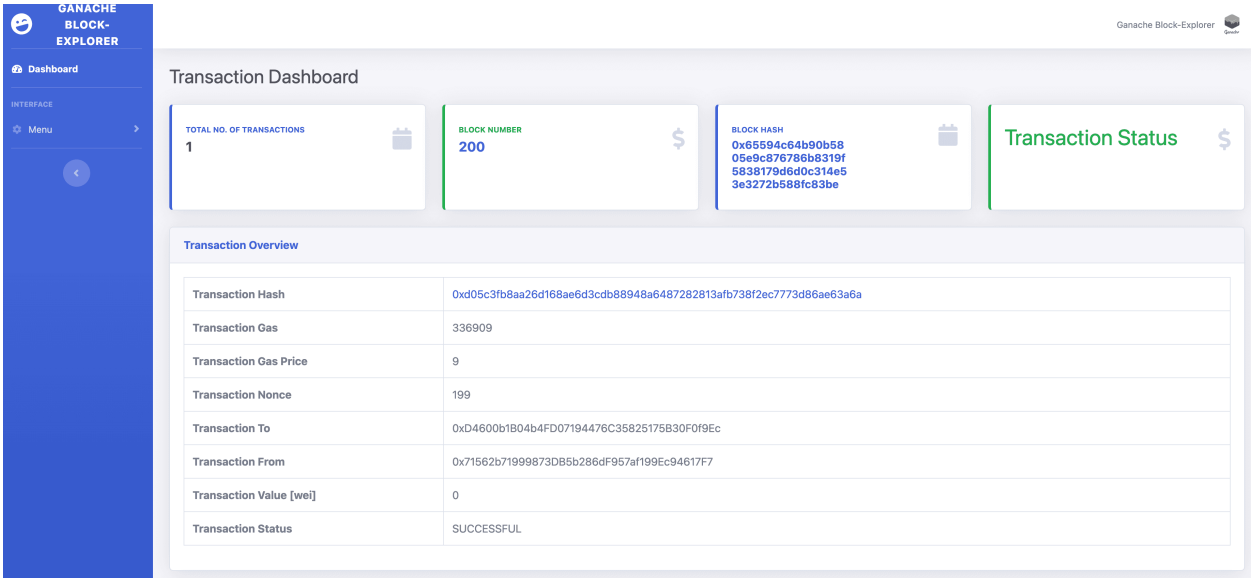
```
http://127.0.0.1:5051/txpage?blocknumber={bh}
```



> **TEMPLATE FOR BROWSER URL**
>
> When you use the `{bh}` placeholder in the Browser URL template, Casibase will replace it with the actual block number and allow you to jump directly to the relevant block information in your blockchain explorer.

Example: Ganache CLI Block Explorer

To quickly view Ethereum blocks and transactions, you can use the open-source blockchain explorer casibase/ganache-cli-block-explorer. This tool provides a simple web interface for browsing blocks, transactions, and contract events on your local Ethereum node.



> **Note:** This explorer is based on vivekganesan01/ganache-cli-block-explorer and includes additional features contributed by Casibase.

After installation and startup, you can access block details directly from the above address as the Browser URL in Casibase.

Private Cloud Providers

Introduction

In Casibase, **Private Cloud Providers** act as a bridge, allowing you to **connect to and manage** various cloud-native resources, such as **Docker** and **Kubernetes (K8s)**, directly from the Casibase interface. Their core objective is to provide a centralized dashboard for monitoring and operating your containerized services, integrating their management seamlessly into your Casibase workflow.

By configuring a provider, you enable Casibase to communicate with your private cloud or on-premises data center. This provides an ideal solution for organizations that want a unified interface to manage both their knowledge base and the infrastructure it runs on, enhancing operational efficiency and control.

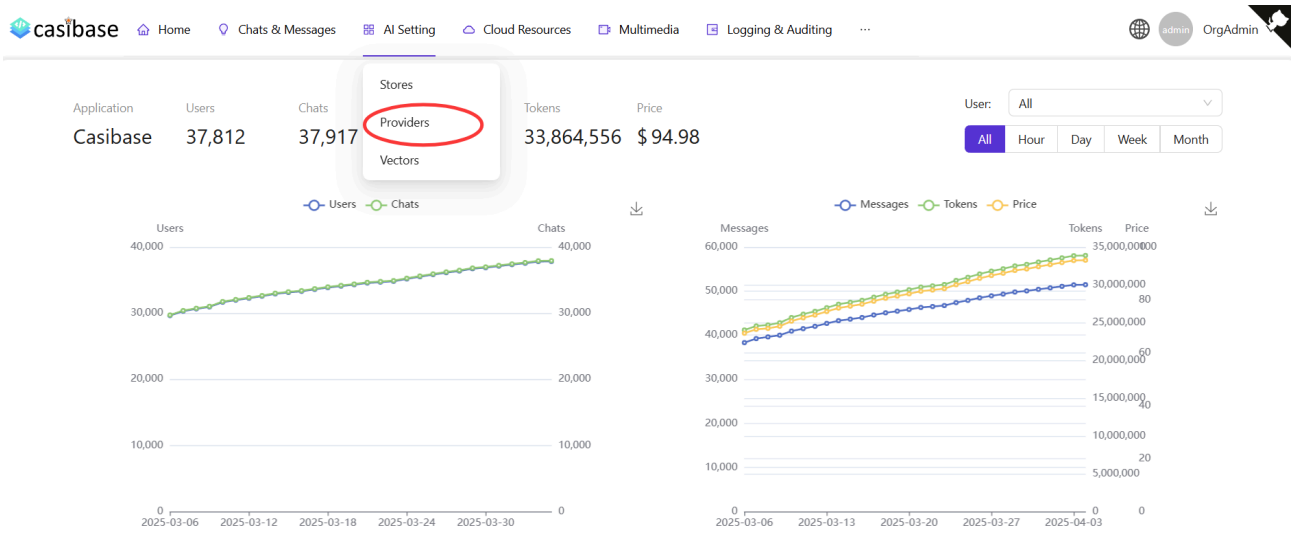
Refer to the Core Concepts section of our previous documentation for more information about providers.

In Casibase, you can add a private cloud provider by following these steps:

Add a New Private Cloud Provider

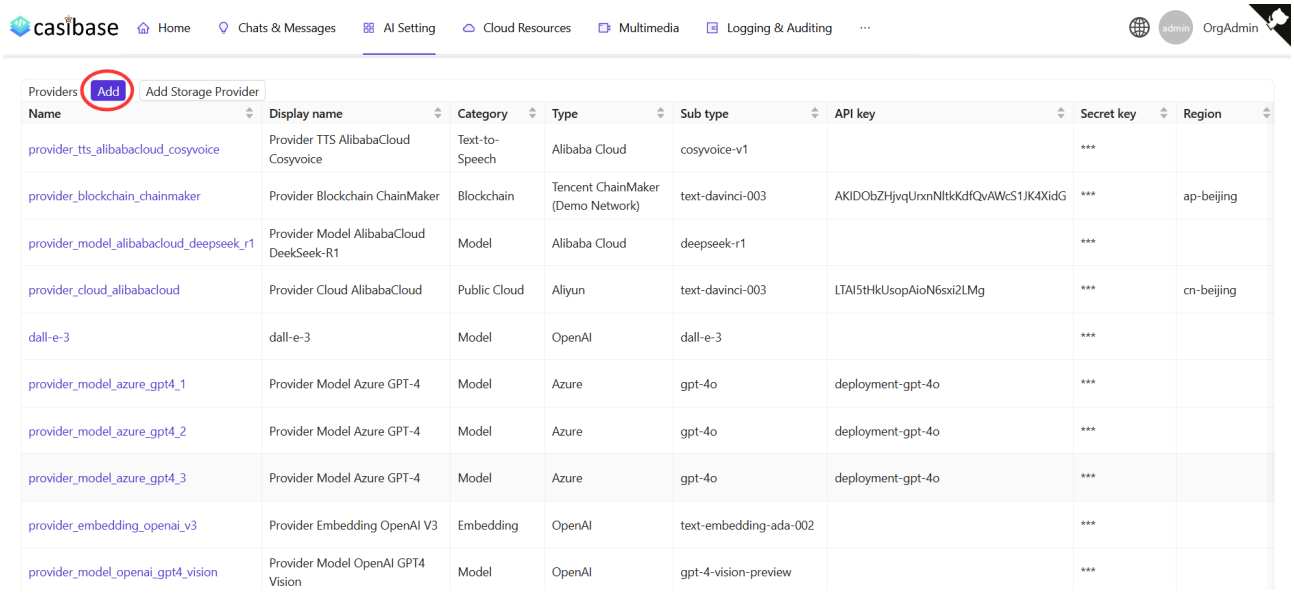
Private cloud providers are used to integrate cloud-native management features into Casibase. You can add them by following these steps:

Click the `Providers` button on the page.



Add a Private Cloud Provider

Click the `Add` button to add a new private cloud provider.



Fill in Private Cloud Provider Details

Fill in the required configuration according to the cloud-native platform you use, then click the `Save & Exit` button to save.

More information about the configuration can be found below:

> 📄 **Kubernetes Configuration**
>
> Kubernetes (K8s) is an open-source container orchestration platform for automating the deployment, scaling, and management of containerized applications. It has bec...

> **TIP**
>
> Casibase supports several mainstream cloud-native technologies and platforms, including:
>
> - **Docker**: To connect to a Docker host and manage the lifecycle of its containers (e.g., start, stop, view status) directly within Casibase.
> - **Kubernetes (K8s)**: To connect to a Kubernetes cluster and manage its resources, such as Pods and Deployments, providing a high-level orchestration view within Casibase.

Now, you can use this private cloud provider to monitor and manage services in your cloud-native environment.

After adding a private cloud provider, you can use Casibase as a control panel to oversee your containerized applications, simplifying management and providing a unified view of your services alongside your knowledge base.

Kubernetes Configuration

Kubernetes (K8s) is an open-source container orchestration platform for automating the deployment, scaling, and management of containerized applications. It has become the de facto standard for managing applications in modern, cloud-native environments. By providing a robust framework for running distributed systems resiliently, Kubernetes simplifies complex operational tasks.

In this chapter, you will learn how to configure and use a Kubernetes provider in Casibase. This will allow you to connect Casibase to your Kubernetes cluster, enabling you to monitor and manage your cloud resources directly from the Casibase interface.

1. Configuration Field Description

When configuring a Kubernetes provider in Casibase, you need to fill in several key fields. Each field has a specific meaning and is required for the correct integration with your Kubernetes cluster. The following list explains the purpose of each field:

- `Name`: The unique identifier for this private cloud provider.
- `Display name`: The display name shown in the UI for this provider.
- `Category`: The type of service; here it should be `Private Cloud`.
- `Type`: The cloud-native platform type; here it should be `Kubernetes`.
- `Config text`: The raw text content of your `kubeconfig` file, which contains the credentials and endpoint information needed to connect to your Kubernetes cluster.

Please make sure to fill in each field accurately. The `Config text` is crucial for establishing a successful connection.

2. Configure Kubernetes

The primary method for connecting Casibase to your Kubernetes cluster is by using your `kubeconfig` file.

2.1 Obtain Your Kubeconfig File

Before proceeding, you must ensure that the `kubeconfig` file you intend to use can successfully connect to your Kubernetes cluster. A reliable way to get the raw configuration is to run the following command in your terminal:

```
kubectl config view --raw > kubeconfig.yaml
```

This command will save the complete, flattened configuration into a file named `kubeconfig.yaml` in your current directory. You can then open this file to copy its contents.

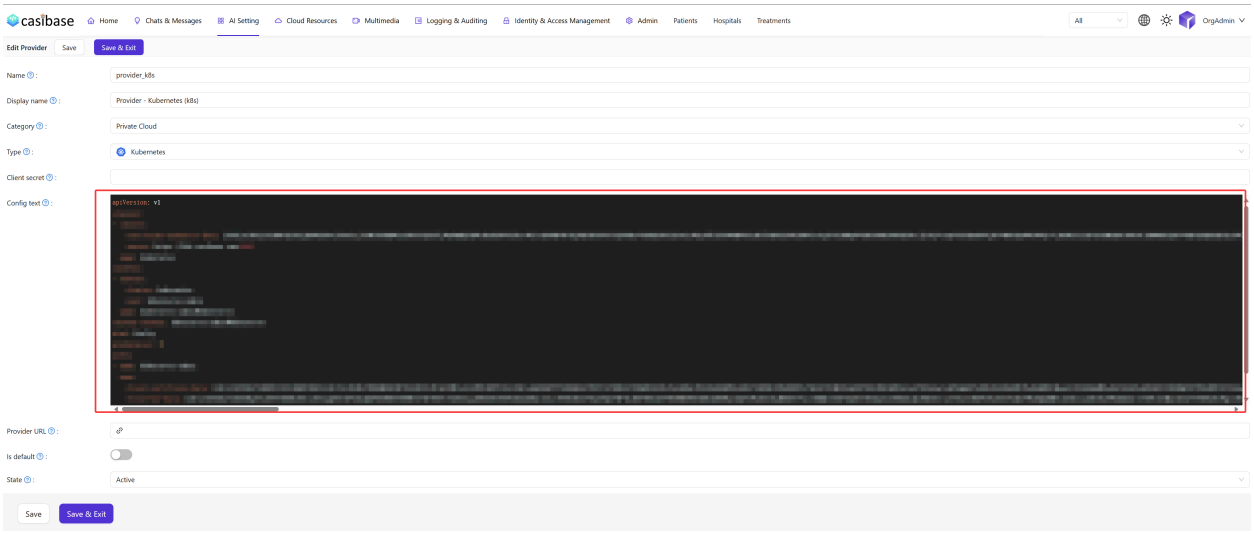
You can test your configuration file with a command that checks for pods across all namespaces. This is a more reliable test to confirm connectivity.

```
kubectl --kubeconfig=./kubeconfig.yaml get pods -A
```

If this command runs successfully (even if it just lists pods from system namespaces), you are ready to proceed.

2.2 Fill in the Provider Details

Copy the entire content of your valid `kubeconfig.yaml` file and paste it into the `Config text` field in the provider configuration form.

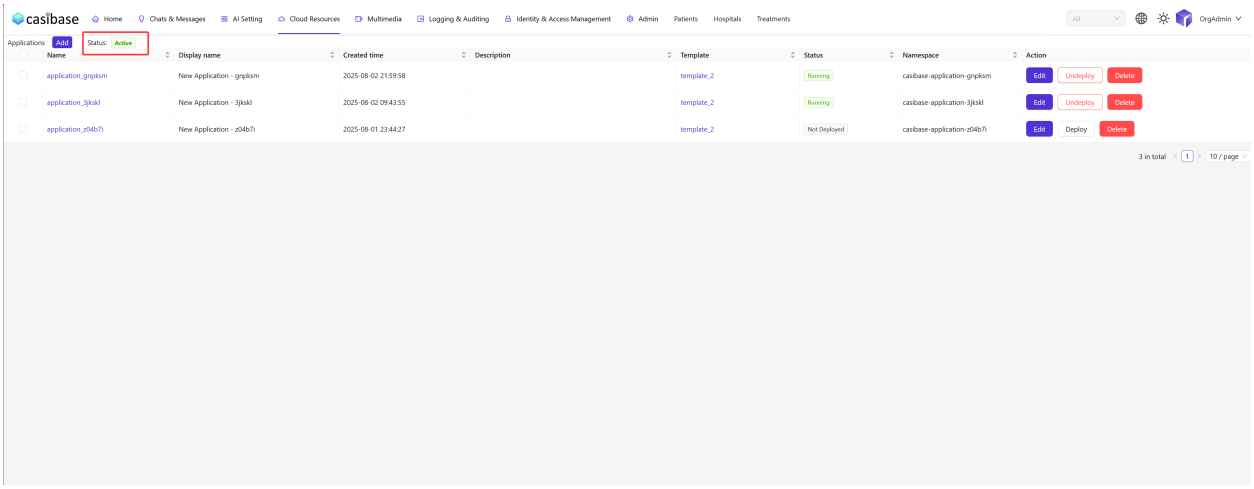


3. Verify the Connection

After you have filled in the details and saved the provider, you can verify if the connection was successful.

Navigate to the **Cloud Resources** > **Applications** section within Casibase. Here, you will see a list of your configured providers. Check the status of the Kubernetes provider you just added.

- **Active**: If the status is `Active`, Casibase has successfully connected to your Kubernetes cluster.
- **Inactive**: If the status is `Inactive`, there was an issue with the connection. Please double-check the content of your `Config text` and ensure that there is network connectivity between Casibase and your Kubernetes cluster's API server.



Once the connection is active, you can begin to monitor and manage your Kubernetes resources through Casibase.

Stores

📄 Overview

Stores Overview

📄 Store Configuration

After adding storage providers, model providers and embedding providers, we can configure the stores

Overview

1. Overview of the Stores Function

In Casibase, the Stores function is one of its core modules, which allows users to integrate storage, modelling, and embedding service providers for knowledge base data storage, text vector conversion, and interaction with chatbots. With the Stores feature, users can build an efficient, flexible and powerful AI knowledge management system.

2.Advantages of Stores

2.1 Multi-model integration

Casibase's Stores feature supports multiple mainstream AI language models, including OpenAI (e.g., GPT-3.5, GPT-4), Azure OpenAI, HuggingFace, Google Gemini, and so on. This multi-model support allows users to choose the most suitable AI model for their specific needs and find a balance between performance, cost and features.

2.2 Multiple storage and embedding options

Users are free to choose storage and embedding service providers to meet different data storage and processing needs. This flexibility enables users to configure the most appropriate storage and embedding solution based on their technology stack and business requirements.

2.3 Multi-Store Mode

Casibase supports a multi-Store model that allows users to use different models, storage and embedding services in different Stores to provide customised services for different scenarios and users. This feature enables users to flexibly configure and switch Stores according to different business requirements.

2.4 Cross-Store Vector Sharing

Stores in Casibase can be configured to use vectors from other stores through the Vector stores field. This allows you to create a main store that searches across multiple specialized knowledge bases, or let different stores share their knowledge with each other. Instead of duplicating content, stores can dynamically access relevant information from other stores while maintaining their own separate vector databases.

3.Summary

Casibase's Stores feature provides users with a powerful knowledge management tool that enables them to flexibly build and manage knowledge bases by integrating multiple AI models, stores and embedded services. Its multi-Store model and enterprise-level features further enhance the flexibility and security of the system, which is suitable for a variety of application scenarios.

Casibase is an open source AI knowledge base system designed to provide efficient and flexible knowledge management and dialogue solutions for enterprises. One of its core features is Providers, which allows users to integrate multiple AI models and storage services to enhance the functionality and performance of the system.Providers are divided into three main categories: Model Providers, Embedding Provides and Storage Providers, which are responsible for handling AI models and data storage, respectively.

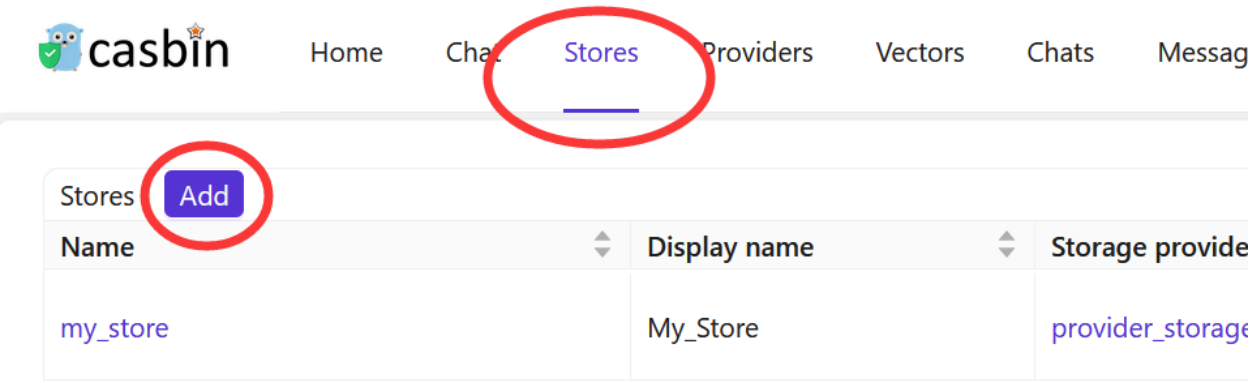
Store Configuration

After adding storage providers, model providers and embedding providers, we can configure the stores

1.Add a New Store

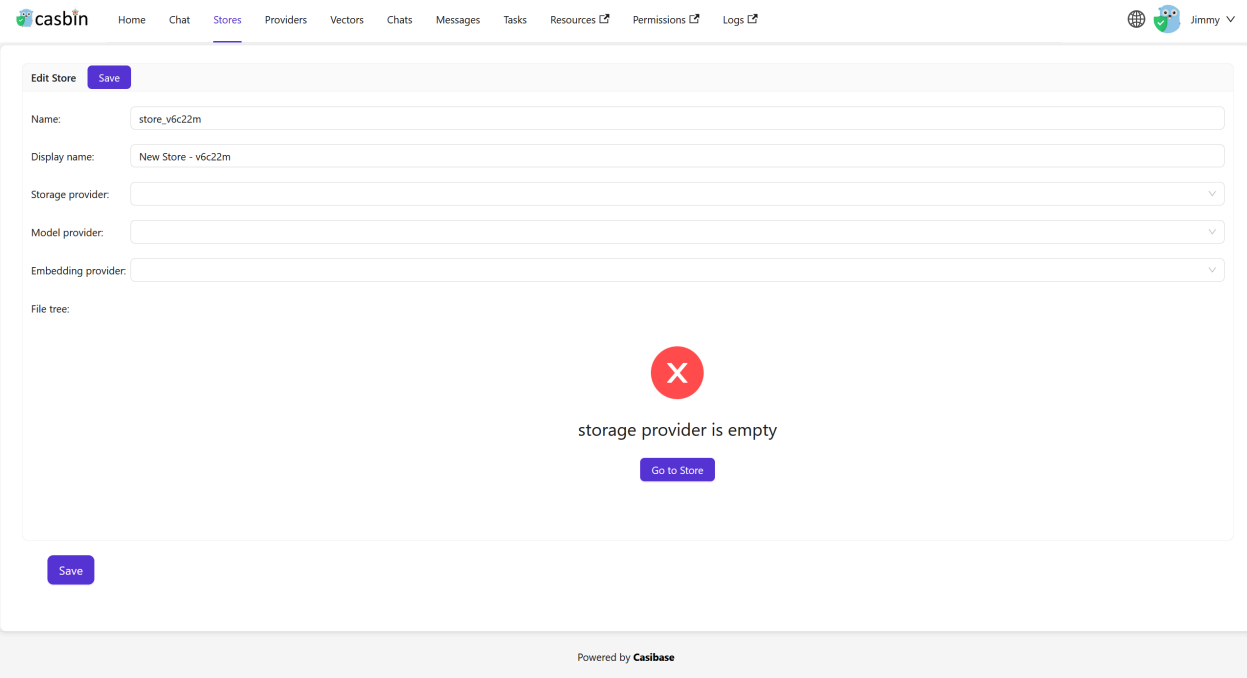
Stores are used to integrate storage, model, and embedding providers into Casibase. You can add them by following these steps:

Click the `Stores` button on the home page and then click the `Add` button to add a store.

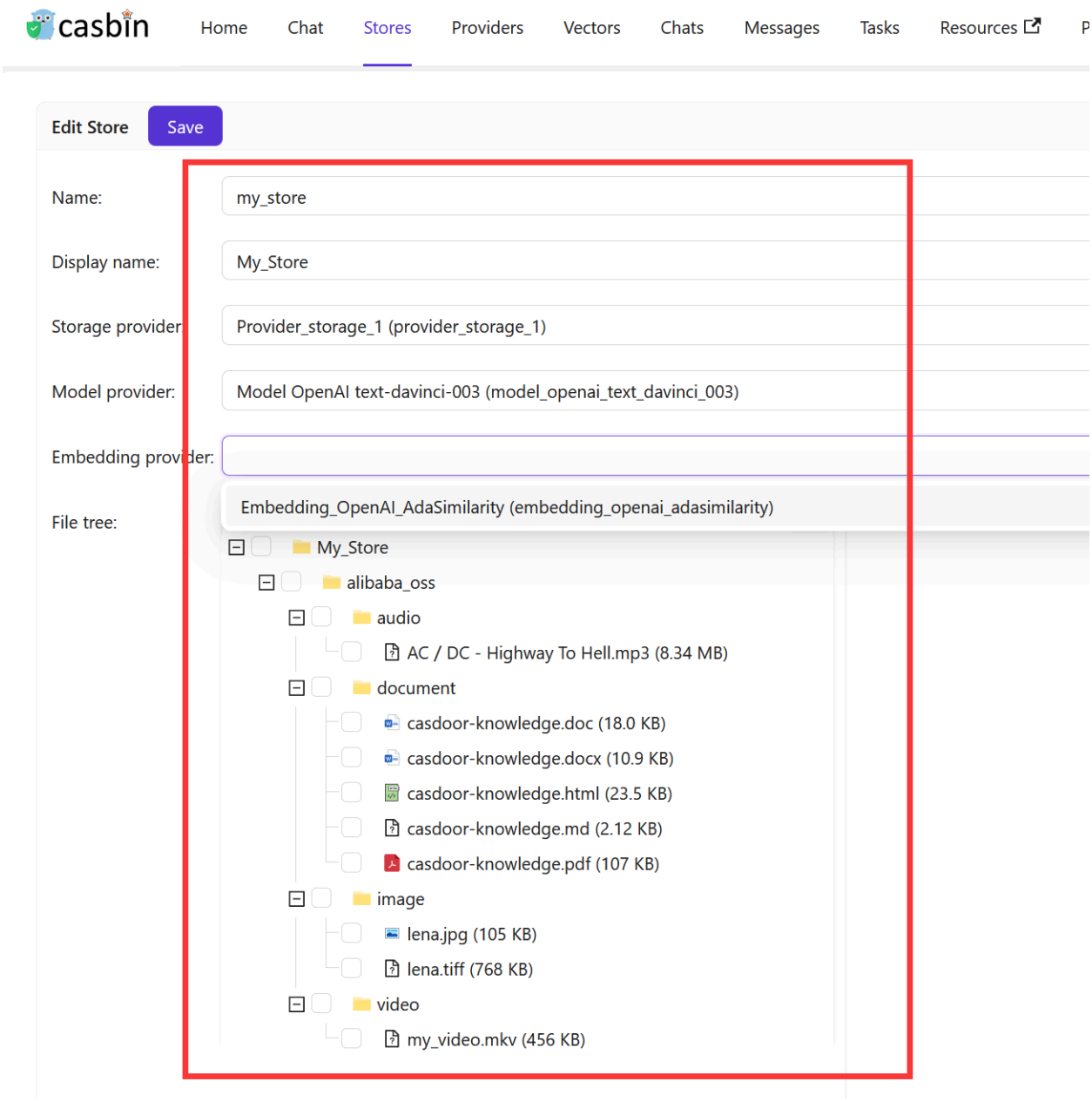


Fill in Store Details

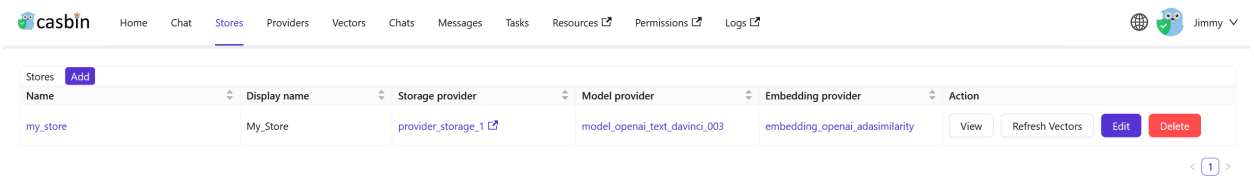
Fill in the store details and click the `Save & Exit` button.



Select the storage provider, model provider, embedding provider, text-to-speech provider and speech-to-text provider you added before.



Click the `Save & Exit` button and return to the stores list page:



Now, you can use the store to store knowledge base data, convert text to vectors, and chat with the chatbot.

Vector Stores

Sometimes you need one store to search through knowledge from multiple other stores. For example, you might have separate stores for different topics or departments, but want a main store that can answer questions by searching across all of them. The **Vector stores** field makes this possible by letting a store use vectors from other stores in addition to its own.

To configure this, navigate to the store edit page and find the **Vector stores** field. You can select one or more stores from the dropdown list. When you chat with this store, it will automatically search through both its own vectors and the vectors from all the stores you selected. This way, you can create a centralized knowledge hub without duplicating content across multiple stores.

> TIP
>
> The store always uses its own vectors automatically. You only need to specify additional stores in the **Vector stores** field when you want to search across multiple knowledge bases.

In the next section, we will learn how to chat with the chatbot in Casibase.

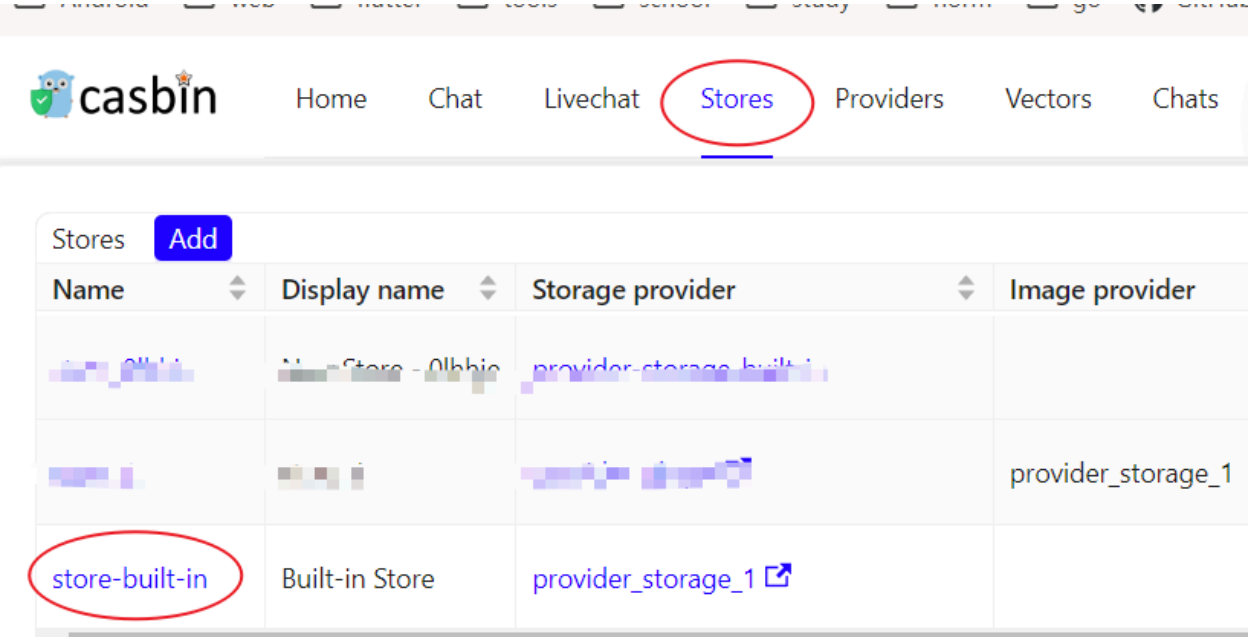
2.Support Multi-store

The multi-store mode provides users with different models, suggestions, and more within each distinct store.

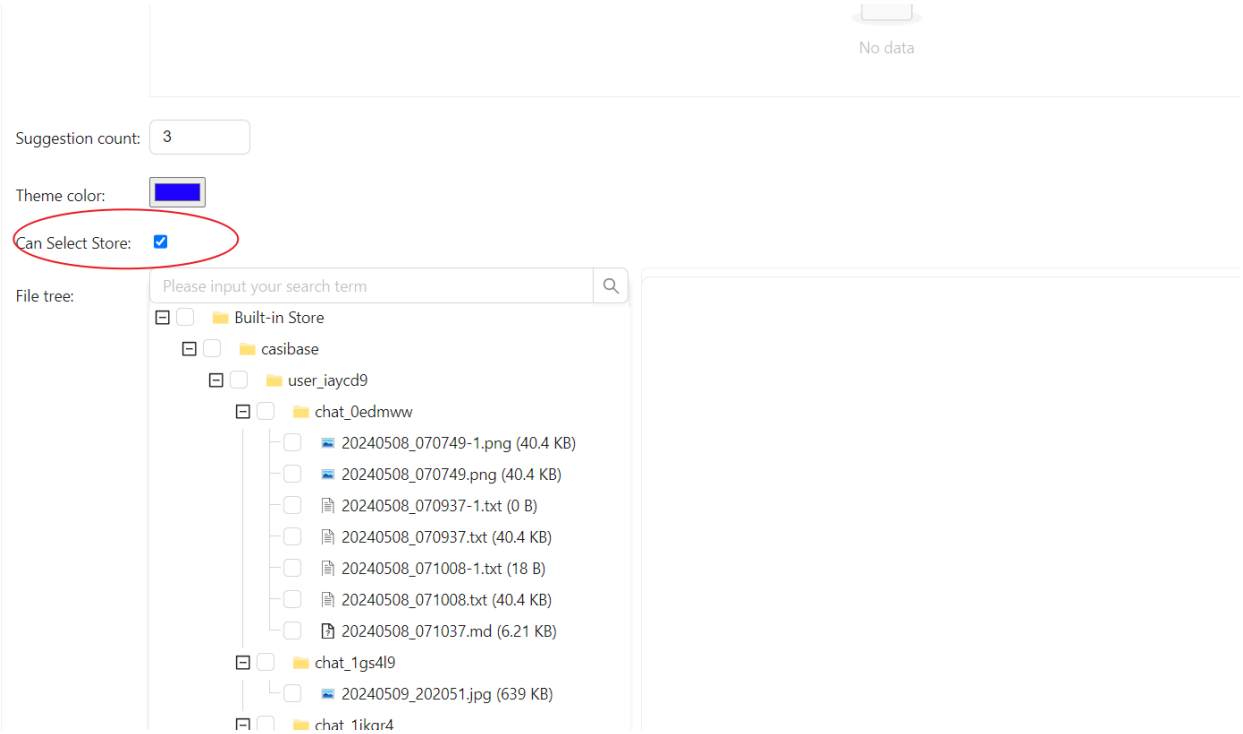
Enable Multi-store

First, you should enable multi-store mode in the built-in store.

Click the `Stores` button on the home page and then click the `store-built-in` button to enter the store-built-in store.



Scroll down and find the `Can Select Store` field, tick it.

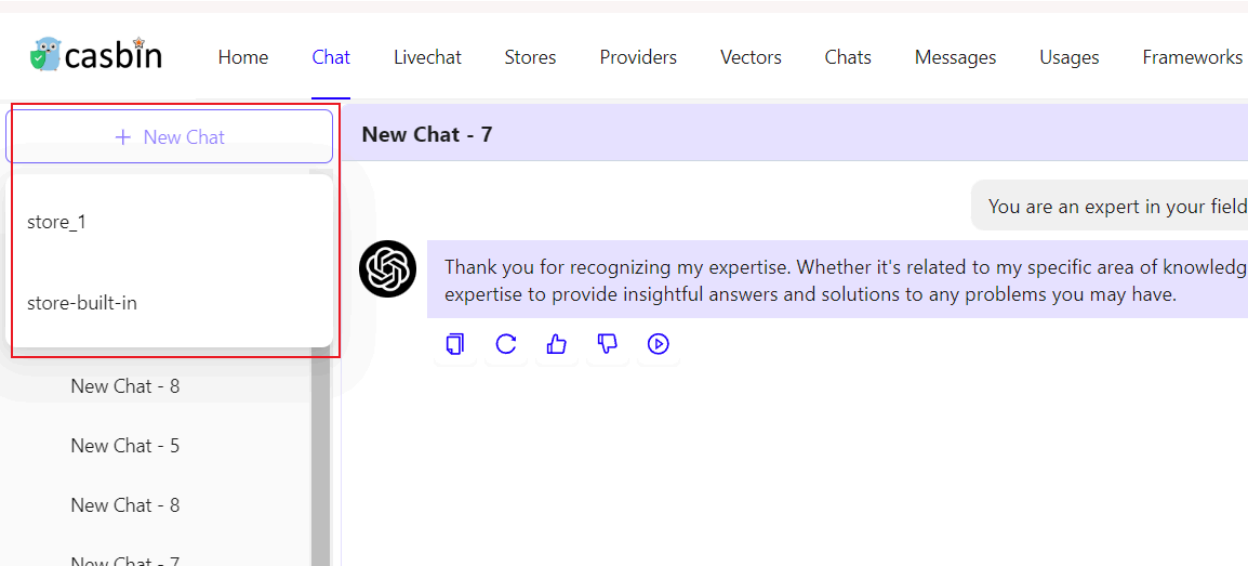


Add Usable Stores

The multi-store mode only provides usable stores. To make a store usable, you need to configure its storage provider, model provider, and embedding provider.

Select For Conversation

Casibase provides a very convenient method for selecting a store.



Just hover your mouse over "New Chat" and then you can select the Store you wish to use from the list that appears below.

If you click the "New Chat" button, the system will assign you a default Store.

Forms

📄 Forms Overview

Introduction to Forms in Casibase

📄 Form Configuration

How to configure and customize forms in Casibase

Forms Overview

Introduction

Forms in Casibase provide a powerful way to customize the display of list pages throughout the application. By configuring forms, you can control which columns are visible, their order, and their width in various list views such as Records, Providers, Stores, and more.

The Forms feature was ported from Casdoor and allows administrators to tailor the user interface to their specific needs, improving usability and focusing on the most relevant information.

Key Features

Customizable Column Visibility

Control which columns appear in list pages by toggling the visibility setting for each form item. This helps reduce clutter and focus on the most important data.

Adjustable Column Width

Set custom widths for each column to optimize the display based on the content type and your screen size preferences.

Column Ordering

Arrange columns in the order that makes the most sense for your workflow by reordering form items.

Multiple Form Categories

Forms support different categories to serve various purposes:

- **Table**: Traditional table-based forms for displaying structured data
- **iFrame**: Forms that can embed external content
- **List Page**: Forms specifically designed to customize list page columns

Form Structure

Each form in Casibase consists of:

- **Organization**: The organization that owns the form
- **Name**: Unique identifier for the form
- **Display Name**: Human-readable name shown in the interface
- **Position**: Placement or order of the form
- **Category**: Type of form (Table, iFrame, or List Page)
- **Type**: Specific form type (e.g., "records" for the Records list page)
- **URL**: Associated URL or endpoint (for Table and iFrame categories)
- **Form Items**: Collection of columns or fields to display

Form Items

Form items define the individual columns in a list page. Each form item includes:

- **Name**: Internal column identifier (e.g., "organization", "name", "createdTime")
- **Label**: Display label shown in the column header
- **Type**: Data type (currently "Text" for list pages)

- **Visible**: Whether the column is displayed
- **Width**: Column width in pixels

Use Cases

Forms are particularly useful for:

1. **Customizing Record Views**: Tailor the Records list page to show only relevant columns for your use case
2. **Simplifying Complex Tables**: Hide technical columns that aren't needed by all users
3. **Optimizing Screen Space**: Adjust column widths to fit more information on screen
4. **Role-Based Views**: Create different forms for different user roles or workflows

Getting Started

To start using Forms in Casibase:

1. Navigate to the Forms section in the Casibase admin interface
2. Create a new form or edit an existing one
3. Configure the form category, type, and items
4. Preview your changes in real-time
5. Save and apply the form to the corresponding list page

For detailed instructions on configuring forms, see the Form Configuration guide.

Form Configuration

Accessing Forms

Forms can be accessed through the Casibase admin interface:

1. Log in to your Casibase admin dashboard
2. Navigate to the Forms section from the main menu
3. You'll see a list of existing forms organized by category and type

Creating a New Form

Step 1: Basic Information

To create a new form:

1. Click the **Add** button on the Forms list page
2. Fill in the basic form information:
   - **Organization**: Select the organization (typically your organization name)
   - **Name**: Enter a unique identifier for the form
   - **Display Name**: Provide a human-readable name
   - **Position**: Set the position or order (optional)

Step 2: Select Form Category

Choose the appropriate category for your form:

- **Table**: For traditional table-based forms
- **iFrame**: For embedding external content

- **List Page**: For customizing list page columns (recommended for most use cases)

Step 3: Configure Form Type

If you selected **List Page** as the category:

1. Choose the **Type** from the dropdown menu
    - Currently supported: **Records**
    - More types will be added in future versions
2. The form will automatically populate with default columns for the selected type

Step 4: Customize Form Items

For each form item (column), you can configure:

Name

The internal identifier for the column. This corresponds to the data field being displayed.

Available columns for Records include:

- `organization`: Organization name
- `id`: Record ID
- `name`: Record name
- `clientIp`: Client IP address
- `createdTime`: Creation timestamp
- `provider`: AI provider name

- `provider2`: Secondary provider
- `user`: Associated user
- `method`: HTTP method
- `requestUri`: Request URI
- `language`: Language
- `query`: Query parameters
- `region`: Geographic region
- `city`: City
- `unit`: Unit information
- `section`: Section
- `response`: Response data
- `object`: Related object
- `errorText`: Error messages
- `isTriggered`: Trigger status
- `action`: Action column
- `block`: Block information
- `block2`: Secondary block information

Visible

Toggle to show or hide the column in the list page. Hidden columns are still available in the form configuration but won't appear in the UI.

Width

Set the column width in pixels. This helps optimize the display based on the content length and your screen size.

> TIP
>
> - Use narrower widths (90-120px) for short fields like IDs, dates, and status indicators
> - Use wider widths (200-300px) for longer text fields like names, descriptions, and URIs
> - Adjust widths based on your typical content length to avoid truncation

Step 5: Reorder Columns

Arrange columns in your preferred order:

1. Use the **Up** arrow button to move a column up
2. Use the **Down** arrow button to move a column down
3. The leftmost columns will appear first in the list page

Step 6: Add or Remove Columns

- **Add Column**: Click the **Add** button to create a new custom column
- **Remove Column**: Click the **Delete** button next to a column to remove it
- **Reset to Default**: Click **Reset to Default** to restore the original column configuration

Step 7: Preview and Save

1. View the **Preview** section at the bottom of the form editor
2. The preview shows how your form will appear in the actual list page
3. Click on the preview to open the full list page in a new window
4. Once satisfied, click **Save & Exit** to apply your changes

Editing Existing Forms

To modify an existing form:

1. Navigate to the Forms list page
2. Click on the form name or the edit button
3. Make your desired changes
4. Click **Save & Exit** to apply the updates

Changes take effect immediately for all users viewing the corresponding list page.

Form Categories in Detail

List Page Forms

List Page forms are the most commonly used type in Casibase. They allow you to:

- Customize which columns appear in list views
- Control column order and width
- Show/hide columns based on user needs
- Create optimized views for different workflows

> ⚠ CAUTION
>
> When configuring List Page forms:
>
> - The **Action** column is always displayed at the end, regardless of form item configuration
> - At least one column should be visible for the list page to be functional

- Column names must match the actual data fields available in the backend

Table Forms

Table forms are used for structured data display in table format. Configuration is similar to List Page forms but may have different available fields based on the data source.

iFrame Forms

iFrame forms allow you to embed external content or applications within the Casibase interface:

1. Set the **URL** field to the external content address
2. Configure display settings as needed
3. The content will be displayed in an embedded frame

Vectors

📄 Overview

Vectors Overview

📄 Vectors Generation

The generation of vectors needs to be used in conjunction with stores, which means that you need to configure stores before you can understand vectors.

Overview

In Casibase, vectors are one of its core strengths. Vector technology plays a key role in knowledge representation and retrieval, and by pairing it with the stores feature, which converts data such as text and images into dense vectors, Casibase enables efficient similarity search and data analysis.

For information on the definition of vectors, see the core-concepts section in our previous documentation.

Application of vector technology in Casibase

Knowledge Embedding

Users can upload files in various formats (e.g. TXT, Markdown, Docx, PDF, etc.) and select embedding methods (e.g. Word2Vec, GloVe, BERT, etc.) to generate knowledge and corresponding vectors. These vectors are stored in a vector database for quick retrieval and query.

Similarity Search

Casibase converts the knowledge into vectors and stores them in a vector database. This vector representation supports a powerful similarity search function, which allows users to quickly find relevant information based on context or content.

Vectors Generation

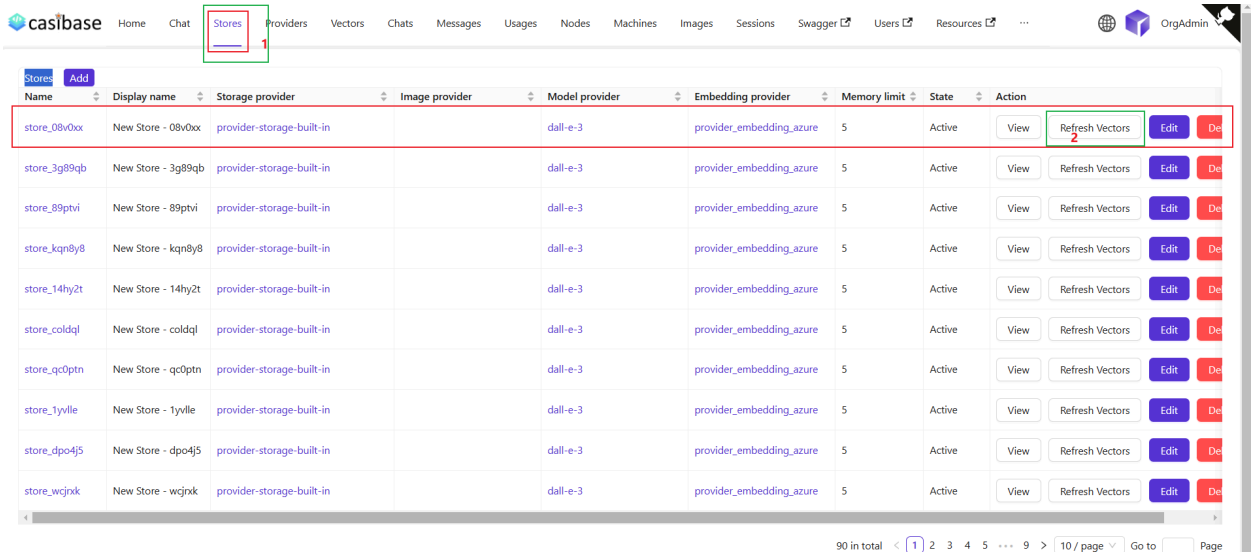
The generation of vectors needs to be used in conjunction with stores, which means that you need to configure stores before you can understand vectors.

Vectors are actually the result of embedding, which is the process of converting various types of data, such as text and images, into dense vector representations. This step is essential to facilitate efficient data processing and analysis within Casibase. With embedding, questions in chat and knowledge files in storage will be converted into vectors that will be used in the next step of knowledge search.

1. Refresh Vectors

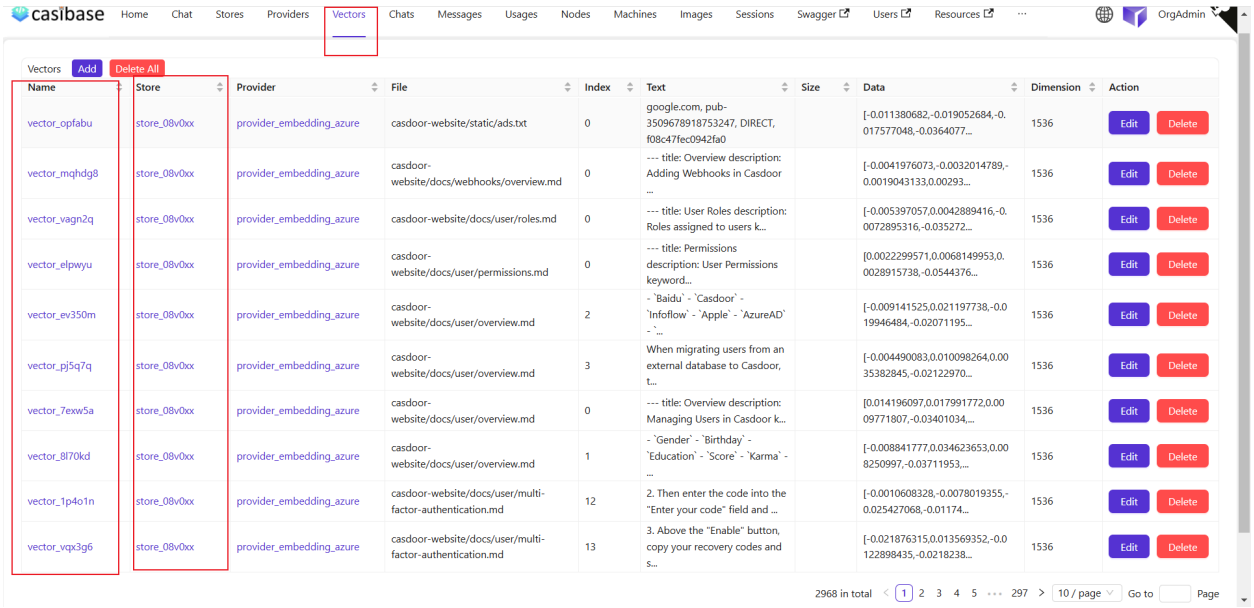
The Refresh Vectors action is set as a button on each store data under the stores menu. In stores, since we will be setting up storage providers, it will provide us with a file tree for storing user files, so after configuring stores, save the configuration and return to the home page and you will see the file tree for the storage providers.

By clicking on the Refresh Vectors button for a particular stores, it will generate the corresponding vectors for all the files in the file tree for that stores by embedding them. The following figure shows the page and the operation.

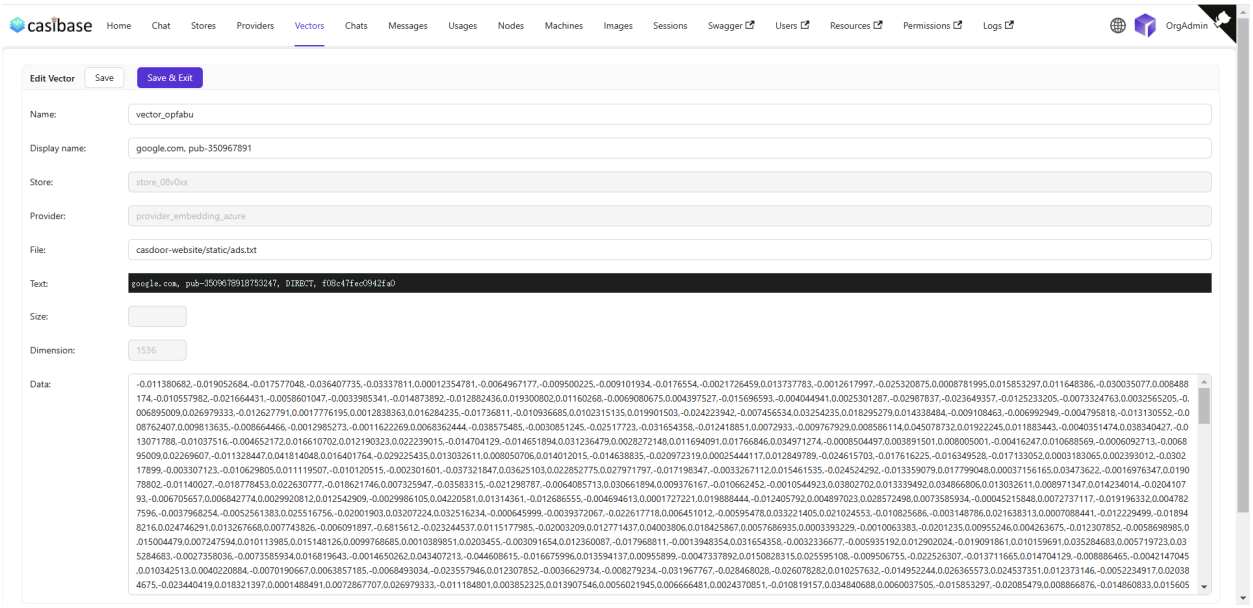


2. View vectors

After that, we can view the specific vectors generated by that storages in the vector menu.



We can see that the files in the stores from the previous step of refreshing vectors have been converted into vectors to display here.



The edit page of my vectors shows specific information such as the name of the store, the name of the embedding model, the name of the file in which the embedding was performed, the file size, the dimension, the vectors data, and so on.

TextSplitters

📄 Overview

Text Splitters Overview

Overview

Text Splitters are a crucial component in building large language model (LLM) applications. Their primary role is to break long texts into multiple shorter segments, which facilitates subsequent tasks such as text embeddings, retrieval-augmented generation (RAG), and question-answering systems.

In LLMs, text splitting is performed for several main reasons:

- Optimizing Efficiency and Accuracy: By decomposing large blocks of text into smaller segments, the relevance and accuracy of the embeddings produced by the LLM can be optimized. Chunking helps ensure that the embedded content contains minimal noise while retaining semantic relevance. For instance, in semantic search, when indexing a document corpus, each document contains valuable information on specific topics. Applying an effective chunking strategy ensures that search results accurately capture the essence of a user's query.
- Limiting the Context Window Size: When using models like GPT-4, there is a limit to the number of tokens that can be processed. For example, GPT-4 has a context window size limit of 32K tokens. While this limit is generally not an issue, it is important to consider chunk size from the beginning. If the text chunks are too large, information might be lost or not all content may be embedded in the context, which can affect the model’s performance and output.
- Handling Long Documents: While embedding vectors for long documents can capture the overall context, they might overlook important details pertaining to specific topics, leading to outputs that are either imprecise or incomplete. Chunking enables better control over the extraction and embedding of information, thereby reducing the risk of information loss.

Casibase currently offers multiple splitting methods, allowing users to apply

different processing strategies for various text scenarios.

Default Text Splitter

The default text splitter is designed to efficiently segment text based on token count and textual structure. Its splitting strategy includes:

- Line Reading and Paragraph Recognition: The text is read line by line, with consecutive blank lines used to accurately determine paragraph breaks. It also sensitively identifies natural breakpoints through markers, ensuring logical and precise text segmentation.
- Special Handling for Code Blocks: Code blocks enclosed by ``` symbols are treated separately. The number of lines within a code block determines whether it can stand alone as a segment. This mechanism preserves the integrity of code blocks while effectively preventing any single text segment from exceeding the token limit.
- Maintaining Sentence Integrity: Throughout the splitting process, strict adherence to sentence integrity is maintained, ensuring that sentences are never divided. This feature guarantees that each text segment contains a complete unit of information. Regardless of the complexity of the text, segmentation is accurately performed at sentence boundaries, effectively avoiding ambiguity and information loss due to broken sentences.

Q&A Splitter

The Q&A splitter focuses on the precise segmentation of question-and-answer formatted texts and offers the following core advantages:

- Accurate Splitting of Q&A Units: It uses a line-by-line scanning mechanism to intelligently identify the structure of Q&A texts. By determining whether each line begins with “Q:” or “A:”, it precisely locates the boundaries between

questions and answers, ensuring that each Q&A pair is completely segmented. This guarantees the independence and completeness of each Q&A unit, providing clean data for subsequent Q&A processing and analysis.

- Clear and Logical Implementation: The code is simple and intuitive, making it easy to understand and maintain. By managing the state of the current Q&A pair and a flag indicating whether an answer is being collected, the process of text segmentation is clearly controlled, ensuring the correct pairing of each Q&A unit.

Chats

📄 Overview

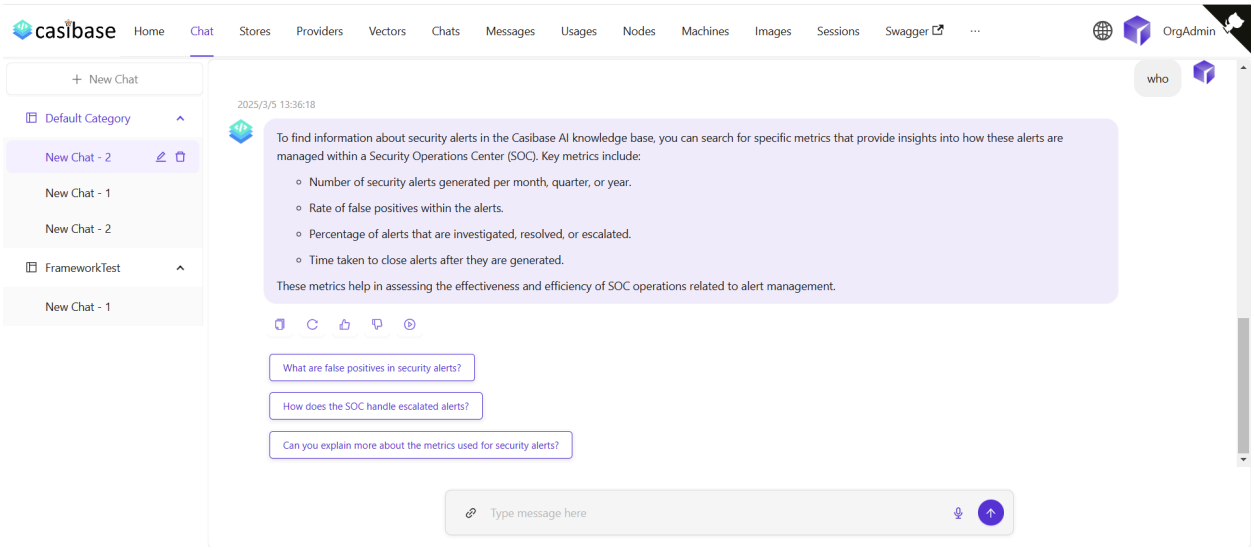
Chats Overview

Overview

In this section, we introduce the most central part of Casibase: chat and its management.

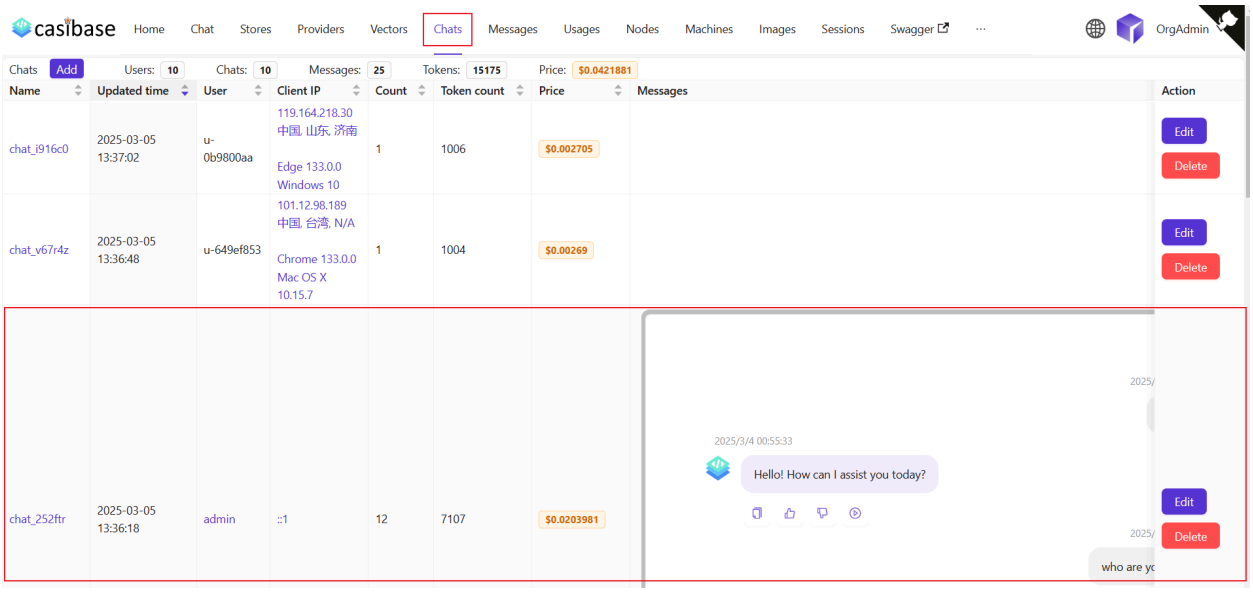
1. Chat

Once we have configured the store, we can have a dialogue with the AI. This is shown in the image below:



2. Chats (Chat management)

We can manage our chat sessions from the Chats menu.



This page allows the user to view the information of the created chats, and the user can also click on Edit to view or edit them. They display the following information:

- `Name`: The name of the created chat.
- `Updated time`: The time when the chat is Updated.
- `User`: The user to whom the chat belongs.
- `Client IP`: Client IP of the chat.
- `Count`: Number of inputs and outputs for this chat.
- `Token count`: The total number of tokens used for this chat.
- `Price`: Total price spent on this chat.
- `Messages`: Showing the content of the chat's message.
- `Store`: Display the Store to which the chat belongs.
- `Category`: Display the Category to which the chat belongs.

Messages

📄 Overview

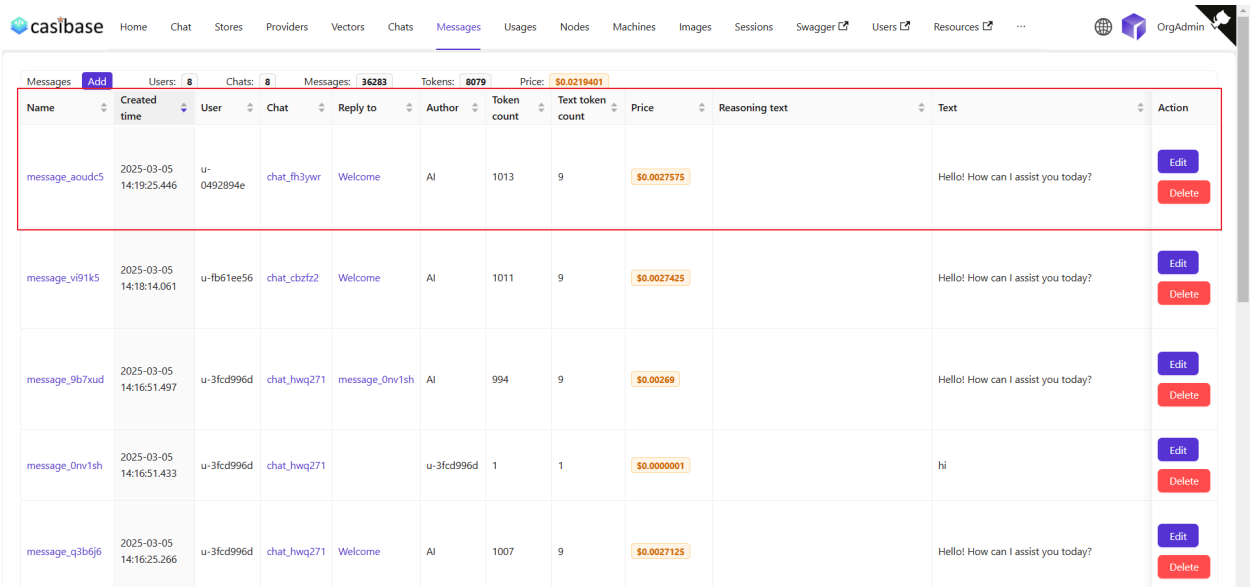
Messages Overview

Overview

In this section, we introduce the functionality of message in Casibase.

Messages

The messages module manages all the messages in our sessions, it shows the creation time of each message, the chat it belongs to, the parent message, the number of tokens, the price, the text message of the reply, the vectors, the suggestions and so on.



Container Cloud

📄 Overview

Container Cloud Overview

📄 Template

In Casibase, a Template is a reusable base configuration for an application. It contains the core Kubernetes manifest files, typically structured for use with Kustomize. Y...

🗃️ Applications

1 items

Overview

Once you have successfully connected Casibase to your private cloud providers (like Kubernetes), this section will guide you on how to manage cloud-native resources directly through the Casibase interface.

Casibase provides a powerful system based on Docker and Kubernetes, designed for individuals and organizations to build their own dedicated container cloud environment. Built on the Casbin permission management engine, it implements fine-grained access control policies for secure and controllable private cloud operations.

Core Concepts

Casibase utilizes a streamlined two-part system for managing container deployments:

1. Templates: Reusable Application Blueprints

Templates are pre-configured Kubernetes manifests that serve as blueprints for your applications. Each template contains:

- Base Configuration: Complete Kubernetes resources (Deployments, Services, ConfigMaps, etc.) required to run an application
- Customizable Parameters: Configurable fields that can be modified during deployment
- Version Management: Template versioning for consistent deployments across environments

2. Applications: Live Application Instances

Applications are running instances created from templates. They represent actual workloads deployed to your Kubernetes cluster:

- Parameter Customization: Override template defaults with specific configurations (replicas, image versions, resource limits)
- Namespace Isolation: Each application runs in its own dedicated namespace for security and organization
- Lifecycle Management: Complete application lifecycle control from deployment to termination

Key Features

Declarative Application Orchestration: Transform from resource-level management to application-level management, simplifying complex multi-resource deployments into simple "select template → configure → deploy" workflows.

Service Governance Integration: Built-in support for service mesh and gateway templates (Istio, Linkerd, Nginx Ingress) enabling one-click deployment of microservice governance capabilities including service discovery, circuit breaking, and rate limiting.

Enhanced Platform Visualization:

- Application Dashboard: Monitor and manage all your deployed applications with real-time status updates
- Resource Insights: Deep visibility into underlying Kubernetes resources, logs, and events for each application

Kustomize-Powered Flexibility: Leverages Kubernetes-native Kustomize for configuration management, ensuring consistency across development, testing, and production environments while maintaining the ability to customize deployments per environment.

This approach helps you standardize your infrastructure, ensure deployment consistency, and streamline the process of launching and managing containerized services. It eliminates the complexity of manual Kubernetes resource orchestration while maintaining full control over your applications.

Please proceed to the following sections to learn more about managing templates and applications:

- Kubernetes Templates
- Kubernetes Applications

Template

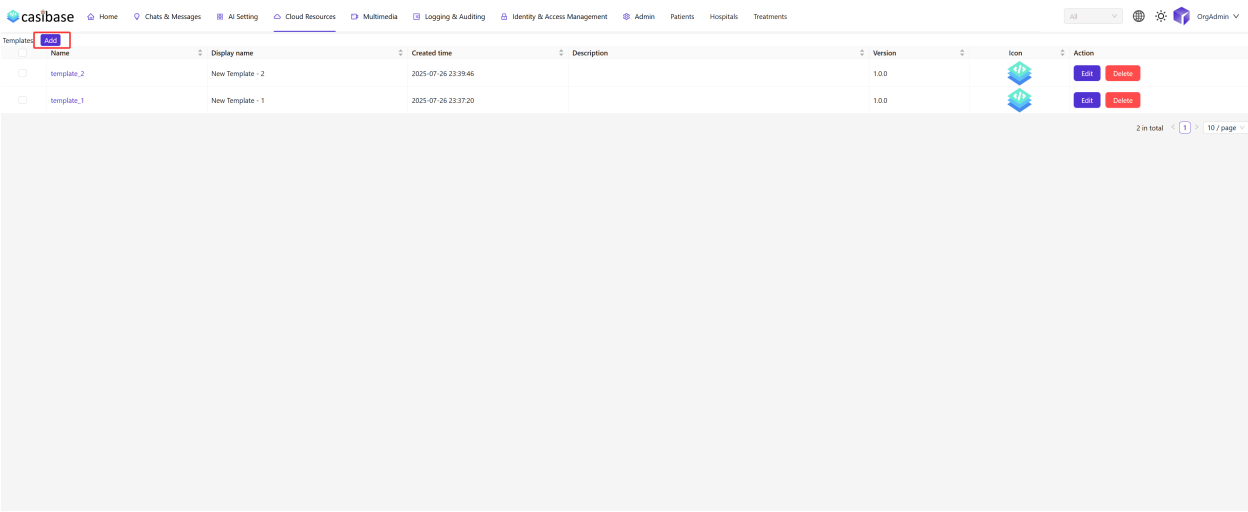
In Casibase, a **Template** is a reusable base configuration for an application. It contains the core Kubernetes manifest files, typically structured for use with Kustomize. You define a template once, and it can then be used as a blueprint to create multiple, customized application instances.

This model allows you to standardize your deployment patterns, ensuring consistency and simplifying the process of launching new services.

This chapter will guide you through creating and managing templates in Casibase.

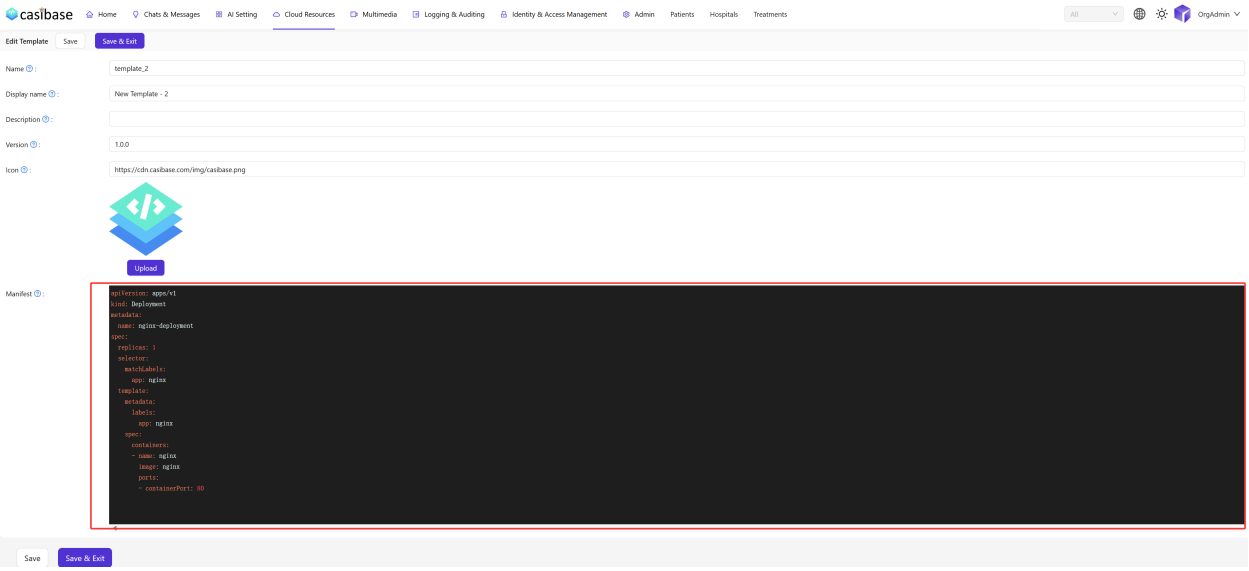
Create a New Template

First, navigate to the **Cloud Resources** > **Templates** section and click the `Add` button to open the creation page.



You will need to fill in the following fields, which correspond to the template's properties:

- `Name`: A unique identifier for the template (e.g., `my-app-template`). This is a required field.
- `Display name`: A user-friendly name that will be shown in the UI (e.g., `My App Template`).
- `Description`: A brief description of what this template is for.
- `Version`: The version of the template (e.g., `1.0.0`).
- `Icon`: A URL to an icon image that represents the template in the UI.
- `Manifest`: The raw YAML text of your Kubernetes manifests. This content serves as the base for Kustomize deployments.



After saving, your template will be available in the selection list when you create a new application.

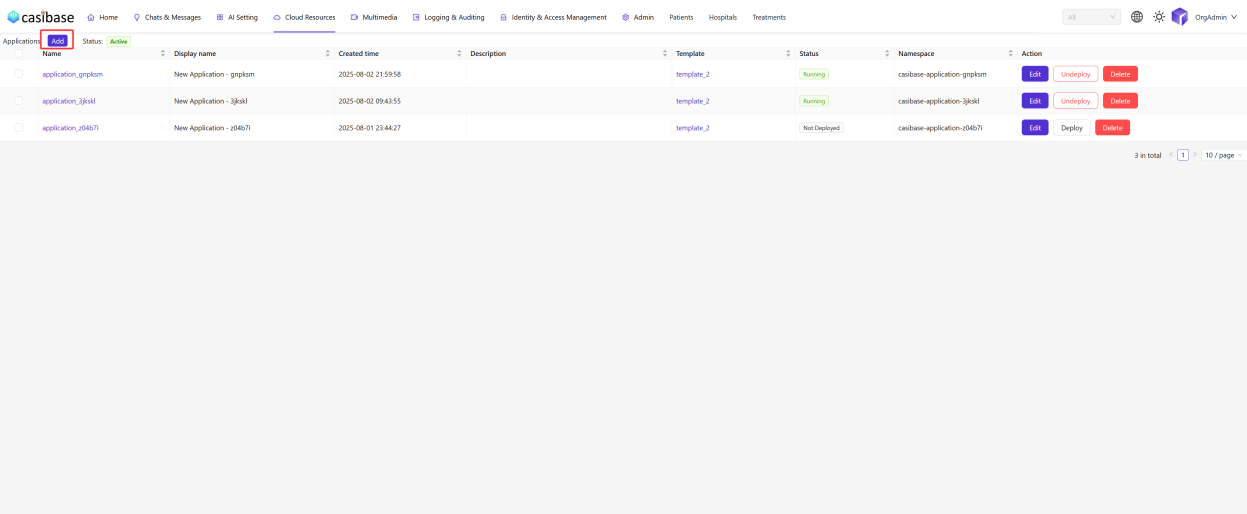
Application

An **Application** in Casibase is a specific, deployable instance created from a **Template**. Before you can create an application, you must first have at least one template defined. When you create an application, you select a base template and can then apply specific customizations before deploying it to your Kubernetes cluster.

This chapter will guide you through creating, deploying, and managing applications in Casibase.

Create a New Application

Navigate to the **Cloud Resources** > **Applications** section and click the `Add` button to open the creation page.



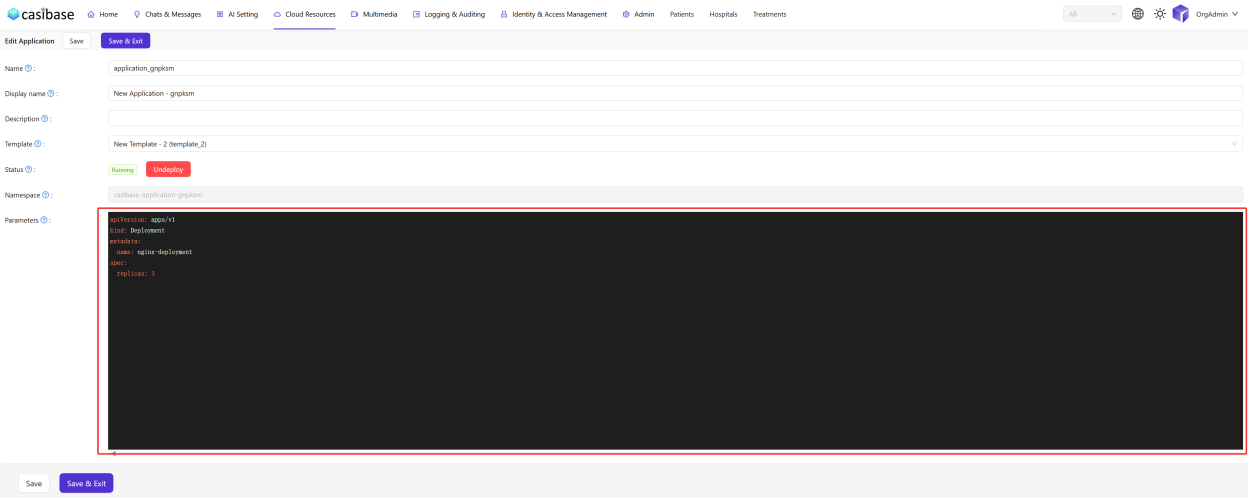
The key fields for an application are:

- `Name`: A unique name for your application instance (e.g., `my-app-prod`). This

is a required field.

- `Display name`: A user-friendly name that will be shown in the UI (e.g., `My App (Production)`).
- `Description`: A brief description of this specific application instance.
- `Template`: Select a pre-existing template from the dropdown list. This will be the base for your application.
- `Parameters`: This field is used for customization. Here you can provide specific Kustomize patches or other variable substitutions in YAML format to override or extend the base `Manifest` from the selected template.

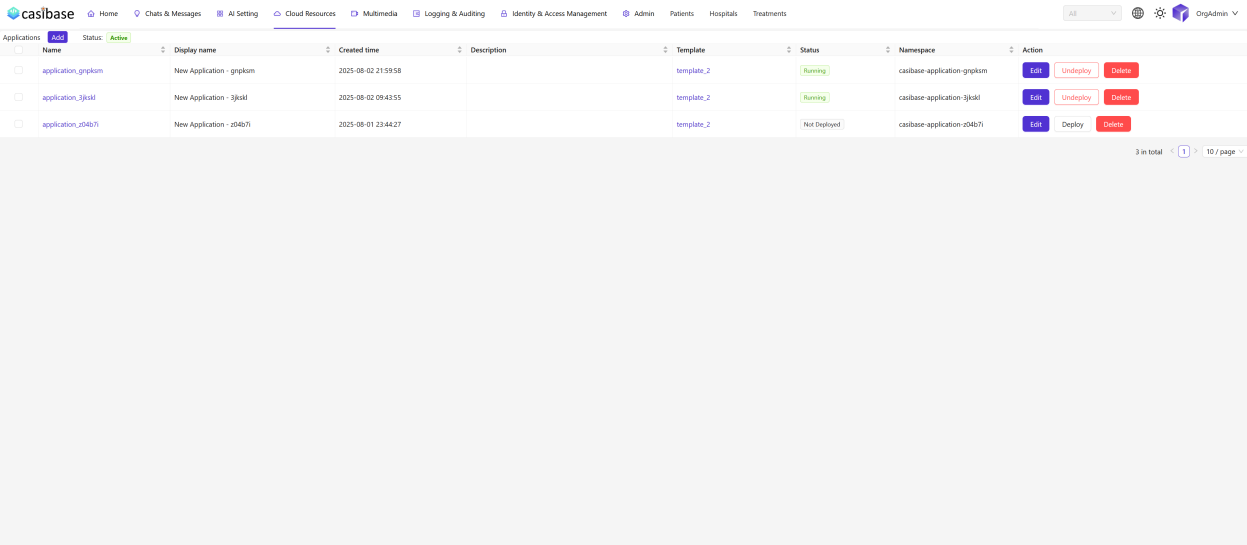
Note: Fields like `Status` and `Namespace` are managed by the system. The **`Namespace` is automatically generated based on the application name upon creation and cannot be modified by the user.** The `Status` is updated based on its deployment state (e.g., `Not Deployed`, `Running`, `Pending`).



Deploying and Monitoring an Application

After creating an application, it will appear in the applications list. From here, you can manage its lifecycle.

- **Deploy**: Click the `Deploy` button to apply the application's configuration to your Kubernetes cluster. Casibase will use Kustomize to merge the base template's `Manifest` with your application's `Parameters` and run `kubectl apply`.
- **Undeploy**: The `Undeploy` button will remove the application's resources from your Kubernetes cluster.



By using this template-and-application model, you can effectively standardize and scale your Kubernetes deployments through the Casibase interface.

Creating Databases with KubeBlocks

Overview

KubeBlocks is an open-source Kubernetes operator and toolset designed to simplify the complexity of running and managing data infrastructure, such as databases, message queues, and streaming systems, on Kubernetes. It provides a declarative approach to deploying and managing stateful applications, allowing you to manage them as easily as stateless ones.

This guide will walk you through the process of creating a database cluster using KubeBlocks.

Why KubeBlocks?

- **Simplified Management**: Automates the database lifecycle, including deployment, upgrades, scaling, and monitoring.
- **Production-Ready**: Supports high availability, backup and restore, and robust monitoring.
- **Versatile**: Supports a wide range of databases, including MySQL, PostgreSQL, MongoDB, Redis, and more.

Installing KubeBlocks

You can install KubeBlocks using `Helm`. For more installation options, refer to the KubeBlocks Installation Guide.

This guide will use Helm for the installation.

Step 1: Deploy Snapshot Controller

KubeBlocks requires the Snapshot Controller to manage volume snapshots. First, check if it is already installed in your cluster.

```
kubectl get crd volumesnapshotclasses.snapshot.storage.k8s.io
kubectl get crd volumesnapshots.snapshot.storage.k8s.io
kubectl get crd volumesnapshotcontents.snapshot.storage.k8s.io
```

If it is not installed, you can deploy it using the following commands:

```
helm repo add piraeus-charts https://piraeus.io/helm-charts/
helm repo update
# Update the namespace to an appropriate value for your
environment (e.g. kb-system)
helm install snapshot-controller piraeus-charts/snapshot-
controller -n kb-system --create-namespace
```

Then, verify the installation:

```
kubectl get pods -n kb-system | grep snapshot-controller
```

The Snapshot Controller should be in the `Running` state.

Step 2: Get the Latest KubeBlocks Version

Get the latest stable version tag (e.g. v1.0.1):

```
curl -s "https://api.github.com/repos/apecloud/kubeblocks/releases?per_page=100&page=1" | jq -r '.[] | select(.prerelease == false) | .tag_name' | sort -V -r | head -n 1

# Example output:
# v1.0.1
```

Step 3: Create KubeBlocks CRDs

Create the Custom Resource Definitions (CRDs) required by KubeBlocks.

```
# Replace <VERSION> with the version you selected
kubectl create -f https://github.com/apecloud/kubeblocks/releases/download/<VERSION>/kubeblocks_crds.yaml

# Example: If the version is v1.0.1
kubectl create -f https://github.com/apecloud/kubeblocks/releases/download/v1.0.1/kubeblocks_crds.yaml
```

Step 4: Install KubeBlocks with Helm

1. Add the KubeBlocks Helm repository:

   ```
   helm repo add kubeblocks https://apecloud.github.io/helm-charts
   helm repo update
   ```

2. Install KubeBlocks: This command installs the KubeBlocks chart into the `kb-system` namespace.

   ```
   helm install kubeblocks kubeblocks/kubeblocks --namespace kb-system --create-namespace
   ```

Creating a Database

Once KubeBlocks is installed, you can create a database cluster using `kubectl`.

Currently supported databases include:

- MySQL
- PostgreSQL
- MongoDB
- Redis
- Kafka
- Milvus
- Qdrant
- RabbitMQ
- Elasticsearch

Example: Creating a Demo MySQL Cluster

1. Create a file named **`my-mysql-cluster.yaml`**:

```yaml
apiVersion: apps.kubeblocks.io/v1
kind: Cluster
metadata:
  name: mycluster
  namespace: demo
spec:
  # Deletes all resources when the cluster is deleted
  terminationPolicy: Delete
  componentSpecs:
    - name: mysql
```

2. Apply the manifest to create the cluster:

```
kubectl apply -f my-mysql-cluster.yaml
```

More Information

For more detailed information, advanced configurations, and troubleshooting with KubeBlocks, please refer to the official KubeBlocks documentation.

Nodes

Overview

Casibase nodes Overview

RDP

Casibase nodes RDP

VNC

Casibase nodes VNC

Overview

Casibase helps you to manage nodes, and connect to your nodes remotely, including remote desktop via RDP, VNC, SSH, and Telnet.

Protocol:

- SSH
- RDP
- VNC
- Telnet

Every node has the following basic properties:

- `Organization`: The organization that the node belongs to.
- `Name`: The unique node name.
- `Description`: The Description of the node.
- `IP`: Domain name or IP address.
- `Protocol`: The port number of the Protocol.
- `Port`: The port number of the node.
- `Username`: The username to connect to the node, such as `root`, `administrator`, `sa`, etc.
- `Password`: The password to connect to the node.
- `OS`: The operating system of the node, including `Windows` and `Linux`, used to classify the node.
- `Tag`: The tag of the node, used to classify the node.

In this chapter, you will learn how to start connecting to your nodes.

Let's explore together!

RDP

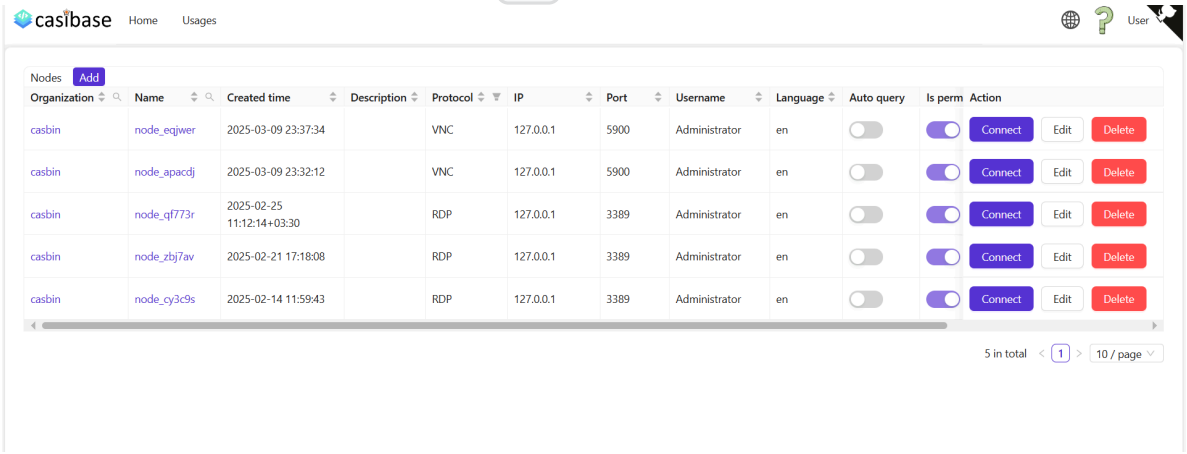
Casibase Support Connect to your nodes via RDP protocol:

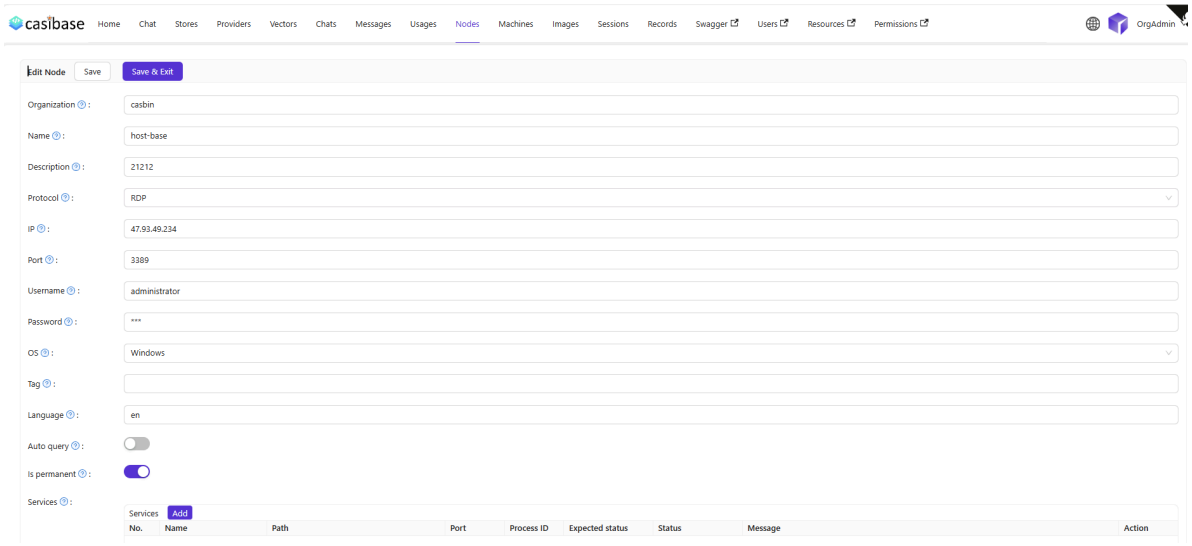
Rdp connection

1. Start Guacamole Server

```
docker run --name guacd -d -p 4822:4822 guacamole/guacd
```

2. Add a new node, set protocol to `rdp`



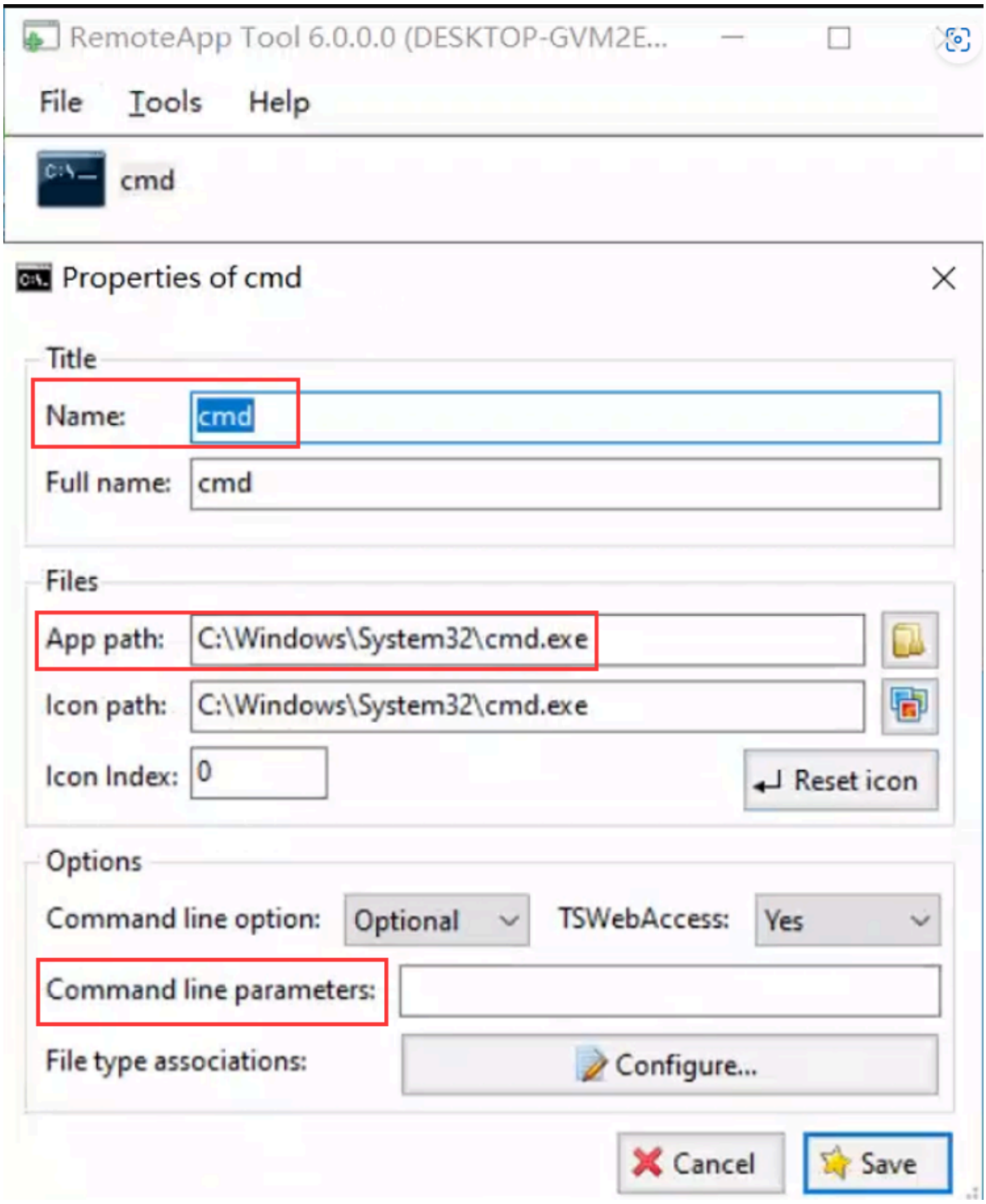


3. Connect to your node by clicking the `connect` button

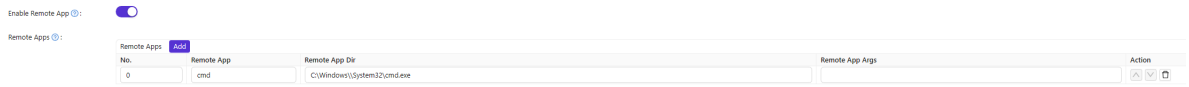
Remote App

We support remote app on Windows nodes, you can add remote apps on `node Edit` page, and then you can connect to your remote app by clicking the `connect` button.

1. Configure your remote app on the server end.
   You can use RemoteApp Tool to register apps.



2. Configure the remote app information in the node edit page according to the server-end configuration. 'remoteAppName', 'remoteAppDir', and 'remoteAppArgs' are required.



refer to Configuring Guacamole — Apache Guacamole Manual v1.5.3

3. Connect to your remote app.

VNC

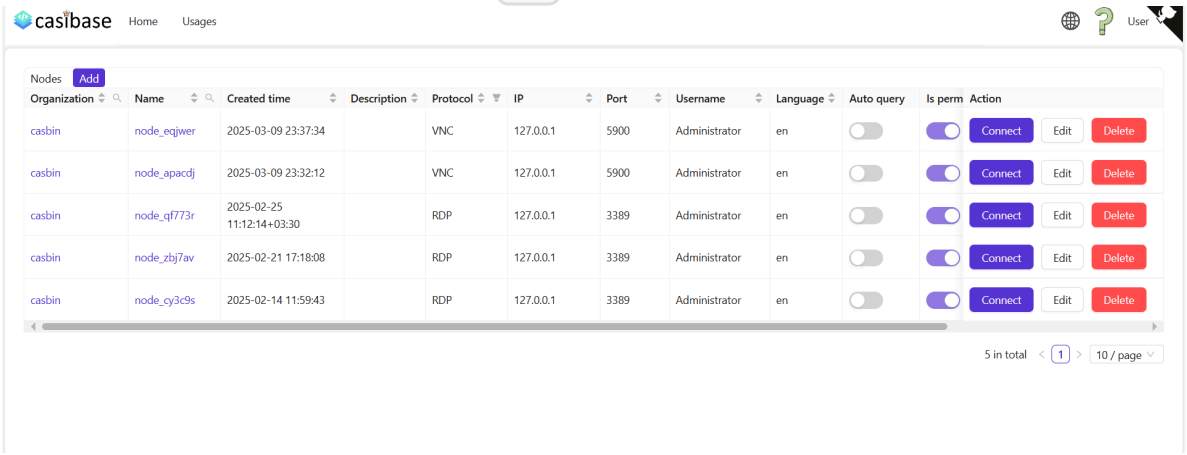
VCN Connect

VCN connection is similar to RDP connections.

1. Start Guacamole Server

```
docker run --name guacd -d -p 4822:4822 guacamole/guacd
```

2. Add a new node, set protocol to `vnc`



3. Connect to your node by clicking the `connect` button.