SUPERTOKEN 中文文档

# SuperToken 工具配置操作手册

从环境准备到主流 AI 编程工具接入


**SuperToken**

配置项以 SuperToken 控制台、模型广场和各工具当前版本为准。

**版本日期**

2026-07-16

# 目录

本 PDF 由 SuperToken 文档源自动生成。网页中的命令和配置更新后，运行 `npm run pdf:build` 即可重新导出。

第 1 部分

工具部署概览

# 工具部署

SuperToken 支持多种 AI 编程工具接入。你只需要准备好 Node.js、API Key，然后按对应工具教程完成配置即可。

## 接入信息

| 节点 | 通用 Base URL | OpenAI 兼容 Base URL |
| --- | --- | --- |
| 默认节点 | `https://api.supertoken.cc` | `https://api.supertoken.cc/v1` |
| 香港节点 | `https://hk.supertoken.cc` | `https://hk.supertoken.cc/v1` |

两个节点共用同一套 API Key、模型和接口路径。

默认节点与香港节点功能一致。不同工具使用哪一种路径，会在各自页面中说明。

## 配置完成要过四关


图：配置完成的四阶段验收。 SuperToken 工具配置手册 2026-07；来源：SuperToken 自制流程图，非第三方产品界面；核对日期：2026-07-14；不含账号、Key 或私人数据

| 关卡 | 要证明什么 | 最小证据 |
| --- | --- | --- |
| 1. 配置已保存 | 新进程仍能读取同一 Key 来源、Base URL、模型和启用状态 | 关闭并重新打开工具或终端，字段仍正确 |
| 2. 网关可请求 | Key、协议、地址和模型在真实 API 请求中有效 | 一次最小只读请求成功，状态码和错误已脱敏保存 |
| 3. 客户端确实走 SuperToken | 工具没有回退到 Auto、内置模型或另一份配置 | 明确选择自定义模型；控制台用量或请求记录能对应时间和模型 |
| 4. 真实任务可验收 | 不只是能聊天，而是能在目标工具中完成工作 | 演示仓库中的 diff、测试、输出文件或真实界面状态 |

> **请求成功不等于配置已保存**
>
> 当前进程里的一次测试可能只使用了临时环境变量或尚未保存的表单。保存后关闭工具，重新打开一个新终端或新窗口，再做一次请求；反过来，设置页显示“已保存”也不证明 Key、模型和网关真的可用。

## Base URL 先按协议选择

| 工具或协议 | 填写值 | 常见错误 |
| --- | --- | --- |
| Claude Code、Anthropic 兼容客户端、CC-Switch 对应 Claude 配置 | 所选节点地址，不带 `/v1` | 错加 `/v1`，或把官网 `https://supertoken.cc` 当 API 地址 |
| Codex、Cursor、OpenAI 兼容 SDK | 所选节点地址，带 `/v1` | 漏掉 `/v1`，或填成完整 `/chat/completions` / `/responses` 路径 |
| Gemini CLI | 以本页 Gemini 教程和当前控制台为准 | 把 OpenAI 或 Anthropic 地址直接套用到 Gemini 协议 |

Base URL 只填协议根地址，不要把控制台首页、模型广场页面或具体接口路径粘进去。模型名必须来自当前账号可用的模型或工具专用别名，不能只凭产品宣传名猜测。

## Key 的最小权限与轮换

1. 按人员、设备、工具和环境分别创建 Key；不要让整个团队共用一个长期 Key。
2. 如果控制台支持分组或模型范围，只开放当前工具需要的模型和额度；测试 Key 不要拥有生产系统权限。
3. 记录 Key 的负责人、用途、创建时间和轮换日期，只在密码管理器或批准的 secret store 保存完整值。
4. 日志、截图、Issue 和聊天里不展示完整 Key；排错最多核对变量是否存在和少量尾号，不回显整个值。
5. 轮换时先创建新 Key，在一个客户端完成四关验收，再逐个迁移；确认没有旧请求后撤销旧 Key。
6. 一旦怀疑泄露，先撤销并创建新 Key，再排查传播范围；不要等到“确认被滥用”才处理。

手工写入 shell 启动文件或用户环境变量意味着 Key 会以可恢复形式保存在本机。只在个人受控设备使用这种方式，并限制配置文件权限；共享电脑、CI 和团队环境应改用批准的 secret store、短期凭据或 runner secrets。不要把真实 Key 直接写进仓库里的 `.env`、`config.toml`、`auth.json` 或教程截图。

## 统一错误决策表

| 现象 | 最常见原因 | 按顺序处理 |
| --- | --- | --- |
| `401` / Invalid token | Key 错误、过期、撤销，变量未加载，或客户端读取了另一份配置 | 新终端确认变量“存在但不回显”；检查 Key 来源；重新复制或轮换；再做最小请求 |
| `403` / Forbidden | Key 分组不允许当前模型、账号或组织策略拒绝该动作 | 检查控制台分组、模型范围和账号状态；换到明确允许的模型，不扩大不相关权限 |
| `404` / HTML 页面 | Base URL、协议路径或模型名错误 | 对照上表检查是否缺/多 `/v1`；删除完整接口后缀；从模型广场复制当前模型名 |
| `429` / Rate limit | 请求过快、并发过高，或账号额度/余额策略限制 | 降低并发并等待重试窗口；检查用量、余额、分组额度和模型状态；不要无限快速重试 |
| 明确提示余额或配额不足 | 可用余额、额度或预算上限不足 | 检查控制台用量和分组预算；充值或调整预算后再发一次最小请求 |
| 连接超时 / TLS / DNS | 本机网络、代理、证书、VPN 或目标域名不可达 | 先在同一终端检查 DNS 与 HTTPS；确认代理变量和公司网络策略，不输出代理凭据 |
| 测试成功，但重启后失效 | 只设置了当前进程变量，表单未保存，或新进程加载了另一配置文件 | 重新保存；关闭并重开；确认配置文件路径和 shell 启动文件；再做真实请求 |
| 客户端有回复，但控制台没有对应记录 | 工具回退到 Auto/内置模型，模型别名不对，或 Base URL 开关未启用 | 关闭 Auto 和回退；明确选择自定义模型；检查 Base URL 开关和请求时间 |

排错时一次只改变 Key、Base URL、模型、网络中的一个变量，并保留脱敏错误全文。把所有设置同时重做，通常只会丢失根因。

## AI 编程工具

| 工具 | 说明 | 配置方式 |
| --- | --- | --- |
| Cursor | AI 代码编辑器 | 填写 OpenAI API Key 与 Base URL |
| Claude Code | Anthropic 官方终端编程代理 | 配置环境变量 |
| Gemini CLI | Google 终端 AI 工具 | 写入 `~/.gemini/.env` |

| 工具 | 说明 | 配置方式 |
| --- | --- | --- |
| Codex | OpenAI 终端编程代理 | 编辑 `~/.codex/config.toml` 并设置 `OPENAI_API_KEY` |
| Memory Agent | 项目长期记忆与上下文管理 | 先查看规划说明 |

> **从接入到实战**
>
> 工具能正常对话后，可以进入新的 AI 编程手册，系统学习 Codex 与 Claude Code 的项目规则、权限、MCP、提示词和真实开发工作流。

## 按目标选择

最快接入 **CC-Switch**

图形界面填写 Key、Base URL 和默认模型，新手优先。

Claude 生态 **Claude Code**

适合终端编程代理，使用 Anthropic 兼容接口。

OpenAI 生态 **Codex**

适合在本地仓库里阅读、修改、审查代码。

**编辑器 Cursor**

在 Cursor 模型设置里填写 SuperToken 网关信息。

## 配置工具

| 工具 | 说明 |
| --- | --- |
| CC-Switch ⭐ | 图形化配置管理工具，适合新手快速接入 |

## 其他工具

| 工具 | 说明 |
| --- | --- |
| OpenClaw | 网关面板管理工具 |

## 开始前

1. 根据目标工具确认是否需要 Node.js；Claude Code 原生安装不要求 Node.js。
2. 在控制台创建最小范围、可轮换的 API Key 和对应模型分组。
3. 选择工具并按协议填写 Base URL、Key 来源和模型。

4. 重启客户端，按“四关”完成保存、网关、客户端和真实任务验收。

---

第 2 部分

安装 Node.js

# 安装 Node.js

多数 CLI 工具都依赖 Node.js 运行环境。推荐安装 **Node.js 22+**（LTS）。

## Windows

### Windows 安装

直接从 Node.js 官网 下载 LTS 安装包，双击运行即可。

安装完成后，打开 PowerShell 验证：

```powershell
node -v
npm -v
npx -v
```

> **常见问题**
>
> - 终端重开前环境变量可能未生效
> - 公司机器没有管理员权限时，可使用 nvm-windows 安装到用户目录

## macOS

### macOS 安装

推荐使用 Homebrew 安装：

```bash
brew install node
```

或从 Node.js 官网 下载安装包。

安装完成后验证：

```bash
node -v
npm -v
npx -v
```

> **建议**
>
> 如果你需要频繁切换 Node 版本，优先使用 nvm 管理。

### Linux

### Linux 安装

优先使用 NodeSource 或 nvm 安装 LTS 版本，发行版自带仓库中的版本可能偏旧。

安装完成后验证：

```bash
node -v
npm -v
npx -v
```

确认 `node`、`npm`、`npx` 都在 PATH 中。

看到版本号即安装成功，继续进入你要使用的工具页面。

---

第 3 部分

CC-Switch

# CC-Switch（配置管理工具）

CC-Switch 是一个图形化配置工具，适合不想手动编辑配置文件的用户。你可以在界面里选择工具类型、填入 `API Key`、`Base URL` 和模型，再把配置切换给目标客户端。

## 快速复制

先把常用配置准备好，再进入 CC-Switch 填写。`API Key` 需要从 SuperToken 控制台复制，这里不展示你的私密 Key。

| 节点 | 通用 Base URL | OpenAI 兼容 Base URL |
| --- | --- | --- |
| 默认节点 | `https://api.supertoken.cc` | `https://api.supertoken.cc/v1` |
| 香港节点 | `https://hk.supertoken.cc` | `https://hk.supertoken.cc/v1` |

两个节点共用同一套 API Key、模型和接口路径。

| 节点 | Claude Code、CC-Switch 等 | OpenAI SDK、Codex 等 |
| --- | --- | --- |
| 默认节点 | `https://api.supertoken.cc` | `https://api.supertoken.cc/v1` |
| 香港节点 | `https://hk.supertoken.cc` | `https://hk.supertoken.cc/v1` |

# 使用步骤

## 1. 安装 CC-Switch

先安装并打开 CC-Switch，然后再开始添加和配置供应商。

### Windows

从 CC-Switch GitHub Releases 下载 Windows 安装包：

- 推荐使用 `.msi` 安装包
- 如果你不想安装，也可以下载 `Portable.zip` 便携版

下载后双击运行，完成安装并打开 CC-Switch。

### macOS

推荐使用 Homebrew 安装：

```bash
brew tap farion1231/ccswitch
brew install --cask cc-switch
```

也可以从 CC-Switch GitHub Releases 手动下载 macOS 版本安装包后打开使用。

### Linux

从 CC-Switch GitHub Releases 下载对应你系统的安装包：

- Ubuntu / Debian：下载 `.deb`
- Fedora / RHEL：下载 `.rpm`
- 其他发行版：可使用 `.AppImage`

常见安装方式：

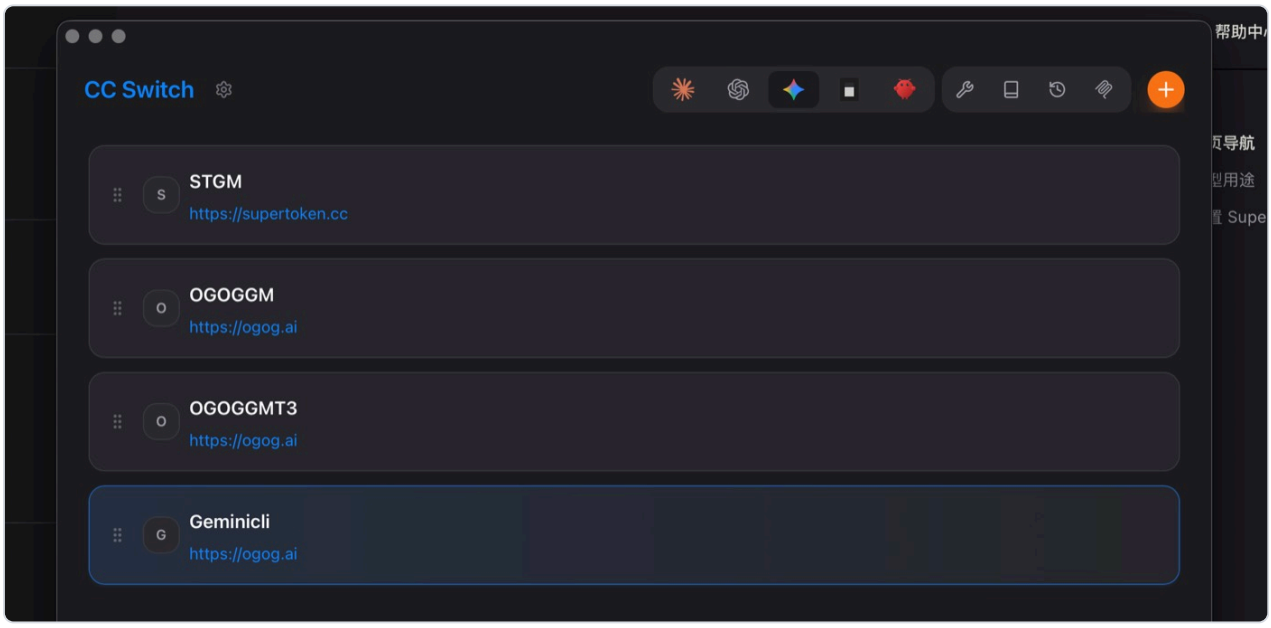
```bash
sudo apt install ./CC-Switch-*.deb
```

或：

```bash
chmod +x CC-Switch-*.AppImage
./CC-Switch-*.AppImage
```

## 2. 选择对应的模型选项

先在 CC-Switch 中选择你要使用的工具或模型类型。




图：在 CC-Switch 中选择目标工具类型。 CC-Switch 历史界面，版本未记录；来源：SuperToken 现有 CC-Switch 实拍，仅用于辨认入口；核对日期：2026-07-14；不含完整 API Key；配置值以当前正文为准

## 3. 配置 API Key、Base URL 和默认模型

进入配置页面后，主要填写这几项：

- `API Key`：你的 SuperToken API Key
- `Base URL`：复制上方所选节点的“通用 Base URL”
- `默认模型`：填写模型广场中你要使用的对应模型

> **旧截图不能当配置值**
>
> 仓库原有配置截图使用了历史示例地址，与当前 API 地址不一致，因此本版不再展示该图。请从上方复制 `https://api.supertoken.cc`（默认节点）或 `https://hk.supertoken.cc`（香港节点）；不要从旧截图抄写 `https://supertoken.cc`。

### 4. 测试是否连通

配置完成后，点击测试，确认当前配置是否可以正常连接。

看到“运行正常”只说明当前测试动作拿到了成功结果。继续完成以下验收：

1. 保存并切换到这条供应商配置。
2. 关闭并重新打开 CC-Switch，确认字段仍存在。
3. 打开目标客户端，明确选择对应模型并发送一次新请求。
4. 在 SuperToken 控制台核对同一时间和模型的请求记录。

如果测试失败，按 统一错误决策表 检查 Key、协议、Base URL、模型和额度；不要直接重建所有配置。

---

第 4 部分

Claude Code 接入

# Claude Code（安装与接入 SuperToken）

Claude Code 是 Anthropic 官方推出的终端编程代理，适合在终端中阅读、修改、重构代码，执行命令和代码审查。

> **这页负责接入**
>
> 完成安装和网关配置后，继续阅读 Claude Code 中文使用手册；想直接看真实项目流程，可以进入 Claude Code 项目实战。

## 你需要准备

- 原生安装不需要 Node.js；只有使用 npm 备用安装方式时才需要 Node.js
- 一个 SuperToken 的 API Key（注册并创建）
- 已创建令牌分组（推荐 Claude Max 或 Claude Kiro）

> **TIP**
>
> Claude Code 使用 Anthropic 兼容接口，地址**不带** `/v1` 。

# 1. 安装 Claude Code

## Windows

打开 PowerShell，使用 Anthropic 官方原生安装程序：

```powershell
irm https://claude.ai/install.ps1 | iex
```

确认安装成功：

```powershell
claude --version
```

## macOS

打开终端（按 `Command + Space`，输入 `Terminal`，回车），使用 Anthropic 官方原生安装程序：

```bash
curl -fsSL https://claude.ai/install.sh | bash
```

验证安装：

```bash
claude --version
```

## Linux

打开终端，使用 Anthropic 官方原生安装程序：

```bash
curl -fsSL https://claude.ai/install.sh | bash
```

验证安装：

```bash
claude --version
```

看到版本号即安装成功。

▶ npm 备用安装方式

后续更新可以运行：

```bash
claude update
```

## 2. 配置环境变量（接入 SuperToken）

环境变量告诉 Claude Code 去哪里调用 AI 服务、使用哪个密钥。需要配置：

- `ANTHROPIC_BASE_URL` — SuperToken 网关地址
- `ANTHROPIC_AUTH_TOKEN` — 你的 SuperToken API Key

**不想手动配置？**

推荐使用 CC-Switch，图形界面填入 Key 即可，无需编辑任何配置文件。以下步骤适合习惯命令行操作的用户。

### Windows

以下命令将配置永久保存到用户级环境变量，重启不丢失。

**设置网关地址：**

```powershell
[Environment]::SetEnvironmentVariable("ANTHROPIC_BASE_URL","https://api.supertoken.cc","User")
```

**设置 API Key（请替换占位符）：**

```powershell
[Environment]::SetEnvironmentVariable("ANTHROPIC_AUTH_TOKEN","你的SuperToken API Key","User")
```

**关闭 PowerShell，重新打开一个新窗口**，然后验证：

```powershell
echo $env:ANTHROPIC_BASE_URL
```

应显示：`https://api.supertoken.cc`

### macOS

macOS 默认使用 Zsh，写入 Zsh 配置文件即可永久生效。

> `~/.zshrc` 是 macOS 的 Shell 配置文件，每次打开终端都会自动加载。下面的命令会自动创建（如果不存在）并追加配置，直接复制粘贴到终端运行即可。

**写入环境变量配置（请替换 API Key）：**

```bash
{
  echo ''
  echo '# SuperToken'
  echo 'export ANTHROPIC_BASE_URL="https://api.supertoken.cc"'
  echo 'export ANTHROPIC_AUTH_TOKEN="你的SuperToken API Key"'
} >> ~/.zshrc
```

**立即生效：**

```bash
source ~/.zshrc
```

**验证：**

```bash
echo "$ANTHROPIC_BASE_URL"
```

应显示：`https://api.supertoken.cc`

## Linux

根据你使用的 Shell 写入对应的配置文件。

> 不确定自己用的是哪个 Shell？运行 `echo $SHELL`，看到 `/bin/bash` 就是 Bash，看到 `/bin/zsh` 就是 Zsh。

> `~/.bashrc`（Bash）和 `~/.zshrc`（Zsh）是 Shell 配置文件，每次打开终端都会自动加载。下面的命令会自动创建（如果不存在）并追加配置，直接复制粘贴运行即可。

**Bash 用户**（大部分 Linux 默认）：

```bash
{
  echo ''
  echo '# SuperToken'
  echo 'export ANTHROPIC_BASE_URL="https://api.supertoken.cc"'
  echo 'export ANTHROPIC_AUTH_TOKEN="你的SuperToken API Key"'
} >> ~/.bashrc
```

**Zsh 用户**：

```bash
{
  echo ''
  echo '# SuperToken'
  echo 'export ANTHROPIC_BASE_URL="https://api.supertoken.cc"'
  echo 'export ANTHROPIC_AUTH_TOKEN="你的SuperToken API Key"'
} >> ~/.zshrc
```

**使配置生效：**

```bash
source ~/.bashrc  # Bash 用户
# 或
source ~/.zshrc   # Zsh 用户
```

**验证：**

```bash
echo "$ANTHROPIC_BASE_URL"
```

应显示：`https://api.supertoken.cc`

你**必须**将 `你的SuperToken API Key` 替换为从 SuperToken 控制台 复制的真实 Key。

## 3. 启动并验证

进入你的项目目录，启动 Claude Code：

```bash
claude
```

可选：先做一次环境检查：

```bash
claude doctor
```

正常对话只能证明当前会话拿到了回复。关闭并重新打开终端，再启动一次 Claude Code；同时在 SuperToken 控制台核对请求时间和模型，才算确认配置已保存且客户端确实走 SuperToken。遇到 `401`、`403`、`404` 或 `429` 时查看 统一错误决策表。

## 4. 下一步

- Claude Code 中文使用手册
- Claude Code 项目实战
- CLAUDE.md、设置与权限

---

第 5 部分

Codex 接入

# Codex（安装与接入 SuperToken）

Codex 是 OpenAI 的终端编程代理，可以在本地仓库中进行代码阅读、修改、执行命令和审查。

> **这页负责接入**
>
> 完成安装和网关配置后，继续阅读 Codex 中文使用手册；想直接看真实项目流程，可以进入 Codex 项目实战。

## 你需要准备

- 已安装 Node.js（推荐 22+ 版本）
- 一个 SuperToken 的 API Key（注册并创建）
- 已创建 GPT 分组（详见 分组与计价）

> **TIP**
>
> Codex 通过 OpenAI 兼容方式接入 SuperToken，地址使用 `https://api.supertoken.cc/v1`。

## 1. 安装 Codex

### Windows

打开 PowerShell，运行：

```powershell
npm install -g @openai/codex
```

验证安装：

```powershell
codex --version
```

### macOS

打开终端，运行：

```bash
npm install -g @openai/codex
```

验证安装：

```bash
codex --version
```

### Linux

打开终端，运行：

```bash
npm install -g @openai/codex
```

验证安装：

```bash
codex --version
```

## 2. 配置 Codex（接入 SuperToken）

Codex 主要需要两部分配置：

- `OPENAI_API_KEY`：你的 SuperToken API Key
- `~/.codex/config.toml`：指定 SuperToken 网关和默认模型

### Windows

### 设置 API Key

```powershell
[Environment]::SetEnvironmentVariable("OPENAI_API_KEY","你的SuperToken API Key","User")
```

### 写入配置文件

```powershell
mkdir -Force "$env:USERPROFILE\.codex"
@"
model = "gpt-5.6"
model_provider = "supertoken"

[model_providers.supertoken]
name = "SuperToken"
base_url = "https://api.supertoken.cc/v1"
wire_api = "responses"
env_key = "OPENAI_API_KEY"
"@ | Out-File -Encoding utf8 "$env:USERPROFILE\.codex\config.toml"
```

## macOS

### 设置 API Key

将下面这段追加到 `~/.zshrc` ：

```bash
{
  echo ''
  echo '# SuperToken for Codex'
  echo 'export OPENAI_API_KEY="你的SuperToken API Key"'
} >> ~/.zshrc
```

然后执行：

```bash
source ~/.zshrc
```

### 写入配置文件

```bash
mkdir -p ~/.codex
cat > ~/.codex/config.toml <<'EOF'
model = "gpt-5.6"
model_provider = "supertoken"

[model_providers.supertoken]
name = "SuperToken"
base_url = "https://api.supertoken.cc/v1"
wire_api = "responses"
env_key = "OPENAI_API_KEY"
EOF
```

### Linux

#### 设置 API Key

根据你的 Shell 写入配置文件：

```bash
{
  echo ''
  echo '# SuperToken for Codex'
  echo 'export OPENAI_API_KEY="你的SuperToken API Key"'
} >> ~/.bashrc
```

然后执行：

```bash
source ~/.bashrc
```

如果你使用的是 Zsh，请写入 `~/.zshrc` 并执行 `source ~/.zshrc`。

#### 写入配置文件

```bash
mkdir -p ~/.codex
cat > ~/.codex/config.toml <<'EOF'
model = "gpt-5.6"
model_provider = "supertoken"

[model_providers.supertoken]
name = "SuperToken"
base_url = "https://api.supertoken.cc/v1"
wire_api = "responses"
env_key = "OPENAI_API_KEY"
EOF
```

你**必须**将 `你的SuperToken API Key` 替换为真实 Key。默认模型写的是 `gpt-5.6`；OpenAI 当前将该别名指向 `gpt-5.6-sol`。接入 SuperToken 时仍要以模型广场实际显示的模型 ID 和分组权限为准。

## 3. 启动并验证

进入你的项目目录后，运行：

```bash
codex
```

能进入并对话只证明当前进程可用。关闭终端，重新打开后再启动 Codex，并在 SuperToken 控制台核对同一时间和模型的请求记录；同时确认 `~/.codex/config.toml` 仍指向 SuperToken。遇到 `401`、`403`、`404` 或 `429` 时查看 统一错误决策表。

## 4. 下一步

- Codex 中文使用手册
- Codex 项目实战
- Codex 常见配置与权限

---

第 6 部分

Cursor 接入

# Cursor（接入 SuperToken）

Cursor 是常用的 AI 代码编辑器。通过 Cursor 的 OpenAI API Key 和自定义 Base URL，可以把部分聊天和编辑类请求接入 SuperToken。

## 你需要准备

- 已安装 Cursor
- 一个 SuperToken 的 API Key（注册并创建）
- 已创建 GPT 分组（详见 分组与计价）

接入信息

Cursor 通过 OpenAI 兼容方式接入 SuperToken。配置时只需要填两项：API Key 和 Base URL。

API Key `YOUR_SUPERTOKEN_API_KEY`
Base URL `https://api.supertoken.cc/v1`

> **前置要求**
>
> Cursor 的自定义 API Base URL 通常需要 Pro、Business 或 Enterprise 订阅。如果 Free 版账号在 Models 页面看不到 `Override OpenAI Base URL` 开关，就无法按此方式接入 SuperToken。可在 `Cursor -> Settings -> Account` 查看订阅状态。

> **TIP**
>
> Cursor 官方说明中，自带 API Key 主要用于聊天模型；Tab 补全等内置能力可能继续使用 Cursor 自带模型。

# 配置步骤

01

### 打开模型设置

进入 **Settings -> Cursor Settings -> Models**。不同版本入口名称可能略有差异，核心是找到 Models 页面。

02

### 填写 API Key

在 **API Keys** 区域打开 **OpenAI API Key** 开关，粘贴你的 SuperToken API Key。

不要带空格或换行。Key 错误时，Cursor 有时会显示成限流或模型不可用。

03

### 填写 Base URL

打开 **Override OpenAI Base URL**，填写 `https://api.supertoken.cc/v1`。

必须带 `/v1`。不要填成根域名，也不要填完整的 `/chat/completions` 接口路径。

04

### 添加自定义模型

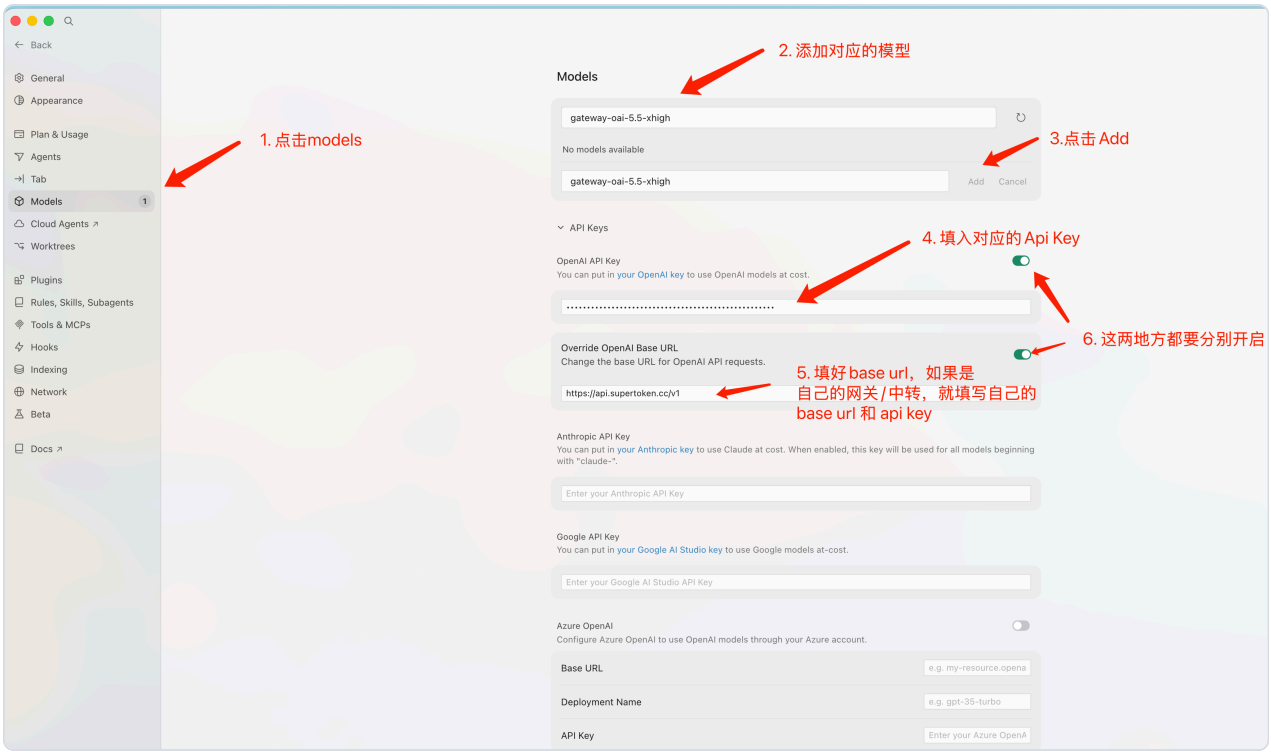
在 Models 页面添加并启用自定义模型。Cursor 使用 `gateway-oai-*` 这类专用模型名，例如 `gateway-oai-5.5` 对应 `gpt-5.5`。如果需要指定思考等级，可以在后面追加等级后缀，例如 `gateway-oai-5.5-xhigh`。

不要直接填写 `gpt-*`、`claude-*`、`o1-*`、`o3-*` 或 `chatgpt-*` 这类原生模型名。Cursor 可能把它们识别成内置模型或应用本地限制，请求不一定发到 SuperToken Base URL。

05

### 选择模型并测试

回到聊天面板，关闭顶部的 **Auto**，在模型下拉中选择刚添加的模型，发送一条简单消息验证。




图：Cursor 自定义模型与 OpenAI Base URL 设置。 Cursor 3.11.13；来源：SuperToken 本机 Cursor 设置实拍；核对日期：2026-07-02；API Key 已遮盖；不含账号信息

## Cursor 模型名怎么填

Cursor 里添加的模型名应使用 SuperToken 为 Cursor 准备的 `gateway-oai-*` 专用名称。具体可用模型以 官网模型广场 和控制台展示为准，常见写法如下：

| Cursor 里填 | 对应模型 | 说明 |
| --- | --- | --- |
| `gateway-oai-5.5` | `gpt-5.5` | 默认思考等级 |
| `gateway-oai-5.5-xhigh` | `gpt-5.5` | 最高思考等级 |
| `gateway-oai-5.4` | `gpt-5.4` | 默认思考等级 |

思考等级通过模型名后缀表达。支持的等级以模型广场为准；如果支持 `-xhigh`，就可以使用类似 `gateway-oai-5.5-xhigh` 的写法。

> **WARNING**
>
> 不要把原生模型名直接填到 Cursor 自定义模型里，例如 `gpt-5.4`、`gpt-5.5`、`claude-sonnet-*`。Cursor 可能把这些名称当作内置模型处理，导致请求没有进入 SuperToken。

# 常见问题

| 现象 | 处理方式 |
| --- | --- |
| 找不到 `Override OpenAI Base URL` | 当前 Cursor 版本或账号可能不支持自定义 OpenAI Base URL。优先确认是否为 Pro、Business 或 Enterprise 订阅，再升级 Cursor 后检查 Models 设置。 |
| 返回 HTML 或 404 | Base URL 填错了。Cursor 里应填写 `https://api.supertoken.cc/v1`，不是完整接口路径。 |
| 401 / Invalid token | API Key 不正确、已失效，或复制时带入了空格。重新到 SuperToken 控制台复制 Key。 |
| 429 / Rate Limit | 可能是请求频率过高、余额不足、模型分组不可用，或 Cursor 因模型名触发本地限制。先确认模型名是否使用了 Cursor 专用别名，再用下方 curl 命令验证 Key。 |
| 请求没有进入 SuperToken | 检查是否直接填了 `gpt-*`、`claude-*`、`o1-*`、`o3-*` 或 `chatgpt-*` 原生名。改用 `gateway-oai-*` 专用模型名。 |
| 聊天面板找不到模型 | 确认 Models 页面中该自定义模型的开关已开启，并关闭聊天面板顶部的 `Auto`。 |
| Tab 补全没有走 SuperToken | Cursor 的 Tab 补全可能继续使用 Cursor 内置模型，这是 Cursor 自带 API Key 模式的限制。 |

# 用 curl 验证 Key

如果 Cursor 报错，先用终端直接验证 SuperToken Key 和模型列表：

```bash
curl https://api.supertoken.cc/v1/models \
  -H "Authorization: Bearer YOUR_SUPERTOKEN_API_KEY"
```

如果这里能返回模型列表，说明 Key 和 SuperToken 网关正常，问题通常在 Cursor 设置项。

这仍不证明 Cursor 已保存并使用该配置。重新打开 Cursor，关闭 `Auto`，明确选择 `gateway-oai-*` 模型，再发送请求并核对 SuperToken 控制台记录。

# 切回 Cursor 自带模型

在 `Settings -> Cursor Settings -> Models` 中：

1. 关闭 `OpenAI API Key`。
2. 清空 `Override OpenAI Base URL`。
3. 删除或关闭自定义模型。

4. 回到聊天面板，重新打开 `Auto` 或选择 Cursor 内置模型。

## 下一步

- API 概览
- OpenAI 格式
- 分组与计价

---

第 7 部分

Gemini CLI 接入

# Gemini CLI（安装与接入 SuperToken）

Gemini CLI 是 Google 推出的终端 AI 工具，可以作为 CLI 助手阅读仓库、进行长上下文分析、通过脚本调用。

## 你需要准备

- 已安装 Node.js（推荐 22+ 版本）
- 一个 SuperToken 的 API Key（注册并创建）
- 已创建 Gemini 分组（详见 分组与计价）

> **TIP**
>
> Gemini CLI 的网关地址通常**不带** `/v1` 后缀。

## 1. 安装 Gemini CLI

### Windows

打开 PowerShell，运行：

```powershell
npm install -g @google/generative-ai-cli
```

验证安装：

```powershell
gemini --version
```

如果提示找不到 `gemini` 命令，尝试替代包名：

```powershell
npm install -g @google/gemini-cli
```

### macOS

打开终端（按 `Command + Space`，输入 `Terminal`，回车），运行：

```bash
npm install -g @google/generative-ai-cli
```

验证安装：

```bash
gemini --version
```

如果提示找不到 `gemini` 命令，尝试替代包名：

```bash
npm install -g @google/gemini-cli
```

### Linux

打开终端，运行：

```bash
npm install -g @google/generative-ai-cli
```

验证安装：

```bash
gemini --version
```

如果提示找不到 `gemini` 命令，尝试替代包名：

```bash
npm install -g @google/gemini-cli
```

## 2. 配置 Gemini CLI（接入 SuperToken）

Gemini CLI 使用自己的配置目录，不需要编辑 Shell 配置文件。

## Windows

### 创建配置目录

```powershell
mkdir -Force "$env:USERPROFILE\.gemini"
```

### 写入环境变量配置文件

创建 `.env` 文件（**请替换 API Key**）：

```powershell
@"
GOOGLE_GEMINI_BASE_URL=https://api.supertoken.cc
GEMINI_API_KEY=你的SuperToken API Key
GEMINI_MODEL=gemini-3.1-flash-image-preview-1
"@ | Out-File -Encoding utf8 "$env:USERPROFILE\.gemini\.env"
```

### 可选：写入认证配置

```powershell
@"
{
  "security": {
    "auth": {
      "selectedType": "gemini-api-key"
    }
  }
}
"@ | Out-File -Encoding utf8 "$env:USERPROFILE\.gemini\settings.json"
```

配置目录位于 `%USERPROFILE%\.gemini\`（通常是 `C:\Users\你的用户名\.gemini\`）。

## macOS

### 创建配置目录

```bash
mkdir -p ~/.gemini
```

### 写入环境变量配置文件

创建 `.env` 文件（**请替换 API Key**）：

```bash
cat > ~/.gemini/.env <<'EOF'
GOOGLE_GEMINI_BASE_URL=https://api.supertoken.cc
GEMINI_API_KEY=你的SuperToken API Key
GEMINI_MODEL=gemini-3.1-flash-image-preview-1
EOF
```

### 可选：写入认证配置

```bash
cat > ~/.gemini/settings.json <<'EOF'
{
  "security": {
    "auth": {
      "selectedType": "gemini-api-key"
    }
  }
}
EOF
```

配置目录位于 `~/.gemini/`（即 `/Users/你的用户名/.gemini/`）。

## Linux

### 创建配置目录

```bash
mkdir -p ~/.gemini
```

### 写入环境变量配置文件

创建 `.env` 文件（**请替换 API Key**）：

```bash
cat > ~/.gemini/.env <<'EOF'
GOOGLE_GEMINI_BASE_URL=https://api.supertoken.cc
GEMINI_API_KEY=你的SuperToken API Key
GEMINI_MODEL=gemini-3.1-flash-image-preview-1
EOF
```

### 可选：写入认证配置

```bash
cat > ~/.gemini/settings.json <<'EOF'
{
  "security": {
    "auth": {
      "selectedType": "gemini-api-key"
    }
  }
}
EOF
```

你**必须**将 `你的SuperToken API Key` 替换为从 SuperToken 控制台 复制的真实 Key。

**配置项说明：**

- `GOOGLE_GEMINI_BASE_URL` — SuperToken 网关地址
- `GEMINI_API_KEY` — 你的 API Key
- `GEMINI_MODEL` — 默认模型（gemini-3.1-flash-image-preview-1 为最新旗舰）

## 3. 启动并验证

```bash
gemini --version
```

发送一条测试消息：

```bash
gemini "你好"
```

收到回复只证明当前进程可用。关闭并重新打开终端，再启动 Gemini CLI；确认配置文件仍生效，并在 SuperToken 控制台核对同一时间和模型的请求记录。遇到认证、模型、地址或限流问题时查看 统一错误决策表。

---

第 8 部分

Memory Agent

# Memory Agent（记忆代理）

Memory Agent 用来沉淀长期项目上下文，例如项目目标、关键约定、常用命令、模型偏好和历史决策。它适合和 Claude Code、Codex、Cursor 等编程工具配合使用，减少每次重新解释项目背景的成本。

> **当前状态**
>
> 本页先作为 Memory Agent 的说明和入口文档。具体接入方式、配置文件路径和可用能力，以后续产品开放为准。

## 适合记录什么

- 项目目标：这个仓库解决什么问题，当前最重要的方向是什么
- 技术约定：框架、目录、命名、代码风格和测试方式
- 常用命令：本地启动、构建、测试、部署和排障命令
- 模型偏好：日常任务、复杂代码、图片/视频任务分别使用什么模型
- 历史决策：为什么选择某个接口、模型、工具或配置方式

## 建议工作流

1. 先用 CC-Switch 或对应工具文档完成 SuperToken 接入。
2. 把项目级约定写入 Memory Agent。
3. 每次开始任务前，让编程代理读取 Memory Agent 中的项目上下文。
4. 任务完成后，把重要决策、命令和踩坑记录回 Memory Agent。

## 推荐结构

```md
# Project Memory

## 项目目标

## 技术栈

## 常用命令

## 模型选择

## 重要决策

## 已知问题
```

## 和 SuperToken 的关系

SuperToken 负责统一模型入口和计费；Memory Agent 负责保存项目上下文。两者配合后，可以让不同 AI 编程工具使用同一套模型网关，并复用同一份项目记忆。

## 下一步

- 用 CC-Switch 快速接入
- 查看模型怎么选
- 浏览工具部署文档

---

第 9 部分

OpenClaw 接入

# OpenClaw（安装与接入 SuperToken）

OpenClaw 是一个网关面板管理工具，适合服务化运行代理。SuperToken 可以作为它的统一上游网关。

## 你需要准备

- 已安装 Node.js（推荐 22+ 版本）
- 一个 SuperToken 的 API Key（注册并创建）

> **TIP**
>
> OpenClaw 地址**带** `/v1`。

## 配置环境变量（接入 SuperToken）

> **不想手动配置？**
>
> 推荐使用 CC-Switch，图形界面填入 Key 即可，无需编辑任何配置文件。以下步骤适合习惯命令行操作的用户。

### Windows

POWERSHELL

```
[Environment]::SetEnvironmentVariable("OPENAI_BASE_URL","https://api.supertoken.cc/v1","User")
[Environment]::SetEnvironmentVariable("OPENAI_API_KEY","你的SuperToken API Key","User")
```

**关闭 PowerShell，重新打开一个新窗口**，然后验证：

```powershell
echo $env:OPENAI_BASE_URL
```

应显示：`https://api.supertoken.cc/v1`

## macOS

> `~/.zshrc` 是 macOS 的 Shell 配置文件，每次打开终端都会自动加载。下面的命令会自动创建（如果不存在）并追加配置，直接复制粘贴到终端运行即可。

```bash
{
  echo ''
  echo '# SuperToken (OpenClaw)'
  echo 'export OPENAI_BASE_URL="https://api.supertoken.cc/v1"'
  echo 'export OPENAI_API_KEY="你的SuperToken API Key"'
} >> ~/.zshrc
```

使配置生效：

```bash
source ~/.zshrc
```

验证：

```bash
echo "$OPENAI_BASE_URL"
```

应显示：`https://api.supertoken.cc/v1`

## Linux

> 不确定自己用的是哪个 Shell？运行 `echo $SHELL`，看到 `/bin/bash` 就是 Bash，看到 `/bin/zsh` 就是 Zsh。

> `~/.bashrc`（Bash）和 `~/.zshrc`（Zsh）是 Shell 配置文件，每次打开终端都会自动加载。下面的命令会自动创建（如果不存在）并追加配置，直接复制粘贴运行即可。

**Bash 用户**（大部分 Linux 默认）：

```bash
{
  echo ''
  echo '# SuperToken (OpenClaw)'
  echo 'export OPENAI_BASE_URL="https://api.supertoken.cc/v1"'
  echo 'export OPENAI_API_KEY="你的SuperToken API Key"'
} >> ~/.bashrc
```

**Zsh 用户**：

```bash
{
  echo ''
  echo '# SuperToken (OpenClaw)'
  echo 'export OPENAI_BASE_URL="https://api.supertoken.cc/v1"'
  echo 'export OPENAI_API_KEY="你的SuperToken API Key"'
} >> ~/.zshrc
```

使配置生效：

```bash
source ~/.bashrc  # Bash 用户
# 或
source ~/.zshrc   # Zsh 用户
```

你**必须**将 `你的SuperToken API Key` 替换为从 SuperToken 控制台 复制的真实 Key。

推荐为 OpenClaw 单独创建一把 Key，避免与个人工具混用。

## 接入重点

- 地址带 `/v1`
- 把供应商配置集中写到环境文件里，方便团队同步

---

## 来源与版本

配置项以 SuperToken 控制台、模型广场和各工具当前版本为准。
本手册核对日期为 2026-07-16。产品界面、模型、命令和账号能力可能更新，快速变化的选项以当前官方页面和本机 `--help` 为准。

1. SuperToken 快速入门
   https://supertoken.cc/guide/
2. SuperToken 工具部署
   https://supertoken.cc/tools/
3. SuperToken API 文档
   https://supertoken.cc/api/