目　录

New API 项目介绍与 Docker 本地部署（含 ChatGPT 订阅接入）

New API 项目介绍与 Docker 本地部署（含 ChatGPT 订阅接入）

New API 项目介绍与 Docker 本地部署

> *核对日期：2026-09-08*
> *适用场景：中文管理后台、个人多模型管理、AI 编程工具接入，以及后续源码二次开发。*
> *本文验证版本：* `v1.0.0-rc.35` *。该版本是 RC 预览版，官方发行说明明确不建议用于生产环境。*

一、项目定位与选型理由

New API 是开源的大模型 API 管理与聚合网关，可以集中配置上游供应商、API 地址、密钥和模型，再向客户端提供统一入口。它本身不运行大模型，也不会自动提供免费的上游调用额度。

基本调用关系：

```
AI 编程工具 / 应用
        ↓
New API：认证、渠道选择、模型映射、用量记录
        ↓
云端模型 API / 第三方兼容接口 / 本地模型服务
```

对于“必须有中文页面、能用 Docker 自部署、将来可以自己修改”的需求，New API 是值得优先评估的选择：

- 官方支持简体和繁体中文，已有渠道、模型、API 密钥、使用日志和系统设置等页面。
- 可以使用 Docker 与 SQLite 在单机上启动，不必先搭建独立数据库。
- 后端采用 Go、Gin，前端采用 React，具备现成的管理界面和供应商适配代码。
- 截至核对日期，仓库约有 4.75 万 Stars、1.14 万 Fork。它们代表关注度和派生仓库数量，不能直接等同于活跃开发者或实际用户数量。

参考：官方仓库、后端依赖、前端依赖。

二、管理界面中的几个核心概念

概念	作用	例子
渠道	配置上游供应商、地址、凭据和可用模型	某个供应商的 API 服务
模型映射	把客户端请求的模型名映射到上游模型名	对外使用统一别名
API 密钥	New API 签发给下游应用使用的访问令牌	给不同编程工具分别创建密钥
分组	组织可访问的渠道与模型范围	个人使用分组、团队使用分组
使用日志	查看请求结果、用量与记账信息	排查请求失败、观察消耗

需要区分两种密钥：上游供应商的 API Key 填在渠道中；编程工具通常填写 New API 创建的下游 API 密钥。

三、Docker 本地预览部署

1. 准备环境

安装并启动 Docker Desktop，或准备 Docker Engine 与 Compose 插件，确认以下命令可正常运行：

```
docker version
docker compose version
```

2. 创建 Compose 配置

新建一个工作目录，将下面内容保存为 `compose.yaml` ：

```
name: new-api-local-preview

services:
  new-api:
    image: calciumion/new-api:v1.0.0-rc.35
    restart: unless-stopped
    ports:
      - "127.0.0.1:3000:3000"
    environment:
      TZ: Asia/Shanghai
    volumes:
      - new-api-data:/data

volumes:
  new-api-data:
```

该配置只开放本机的 3000 端口；配置和 SQLite 数据库存放在独立 Docker 数据卷中，容器重建后仍可保留。

这里固定版本是为了复现预览环境。长期使用时，应重新核对官方发布说明与安全更新情况，选择经过实际业务验证的版本。

3. 启动与初始化

在配置文件所在目录执行：

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

打开本机浏览器访问 `http://localhost:3000` ，按照初始化向导创建管理员，并根据个人用途选择自用模式。登录后即可查看中文控制台。

管理员密码由部署者自行设置，本文不记录真实密码、API Key 或 OAuth 凭据。

4. 日常管理

```
# 查看最近的运行日志
docker compose logs --tail=100 new-api

# 停止并移除容器，保留数据卷
docker compose down

# 再次启动
docker compose up -d
```

`docker compose down -v` 会删除该项目的数据卷，不应用作普通停止命令。升级或修改数据库之前，应先备份数据并准备回退方案。

部署依据：官方 Docker 说明、该版本发行说明。

5. 元数据持久化：支持哪些数据库

New API 的主数据库支持 SQLite、MySQL、PostgreSQL 三种选择。 主数据库负责保存用户、渠道与上游凭据、下游 API 密钥、模型元数据、分组、额度以及系统设置等业务数据。模型推理仍由上游服务执行，主数据库不存储或运行模型权重。

需要区分主数据库、日志数据库与缓存：

存储组件	能否保存主要业务元数据	在 New API 中的用途	配置方式
SQLite	可以	默认主数据库； 日志未分离时也保存在同一数据库	不设置 `SQL_DSN`； 可通过 `SQLITE_PATH` 指定文件位置
MySQL	可以	外部主数据库， 也可作为独立日志数据库	主库使用 `SQL_DSN`， 独立日志库使用 `LOG_SQL_DSN`
PostgreSQL	可以	外部主数据库， 也可作为独立日志数据库	主库使用 `SQL_DSN`， 独立日志库使用 `LOG_SQL_DSN`
Redis	不可以替代主数据库	缓存及相关运行辅助功能，可选	`REDIS_CONN_STRING`
ClickHouse	不可以作为主数据库	当前验证版本支持用作独立日志数据库	`LOG_SQL_DSN`； 不能配置为主库 `SQL_DSN`

上述范围依据 `v1.0.0-rc.35` 的数据库初始化代码与环境变量示例。不同版本可能有差异，尤其不要把“支持日志存储”理解为“支持保存全部元数据”。数据库选择实现、环境变量示例

当前这套 Docker 部署如何保存数据

本次部署使用 SQLite + Docker 命名数据卷：

```
New API 容器
  └── /data
      └── SQLite 数据库文件
          ↕ 持久化挂载
Docker 数据卷：new-api-local-preview_new-api-data
```

- 当前没有设置 `SQL_DSN`，因此选择默认 SQLite。
- 当前没有设置 `LOG_SQL_DSN`，日志与主数据库共用数据库连接。
- 当前没有设置 `REDIS_CONN_STRING`，未启用 Redis。
- 数据卷挂载目标是容器内的 `/data`，数据库不只存在于容器可写层中。
- 普通重启、停止或重建容器时，只要继续使用原数据卷，数据就会保留。
- 在 macOS Docker Desktop 中，命名数据卷由 Docker 的 Linux 虚拟机管理，并不是 Mac 上的 `/data` 文件夹。
- 更改 Compose 项目名或卷名可能使新容器连接到另一个空数据卷；原数据未必丢失，应先检查挂载关系。

- 持久化数据卷不是备份。删除数据卷、重置 Docker Desktop 数据或磁盘故障仍可能导致数据丢失。

日志是否需要单独存储

默认不需要。未设置 `LOG_SQL_DSN` 时，日志复用主数据库，个人使用最容易维护。

当请求日志明显增长、希望单独安排保留策略和备份，或日志写入开始影响业务数据库时，再考虑分离。例如：

```
New API
  ├── SQL_DSN      → PostgreSQL / MySQL：业务元数据
  ├── LOG_SQL_DSN  → 独立数据库 / ClickHouse：日志
  └── Redis        → 可选缓存
```

接入外部数据库之后，业务数据的持久化由数据库服务自身负责。只保留 New API 容器的 `/data` 数据卷，不能代替外部 MySQL、PostgreSQL 或 ClickHouse 的数据卷与备份。

如何选择

- 个人本地预览、少量调用：SQLite 即可。 不必为了“持久化”额外搭建 Redis 或外部数据库。
- 准备长期维护、团队共享或部署多个网关实例：优先评估 PostgreSQL 或 MySQL。 如果已有可靠的 MySQL 环境，可以复用数据库服务，为 New API 建立独立数据库和专用账号。
- 日志规模较大：先考虑独立日志数据库，再按实际需求评估 ClickHouse。
- Redis 按功能和负载需要添加。 即使开启 Redis 自身持久化，它也不会变成 New API 的主数据库。

这里的选型是部署建议，不代表 SQLite 只能用于临时体验，也不代表接入外部数据库就自动具备高可用能力。

从 SQLite 切换到 MySQL / PostgreSQL 的注意事项

修改 `SQL_DSN` 只改变连接目标，不会自动把 SQLite 中的现有用户、渠道、密钥、额度和日志复制到新数据库。 自动建表或表结构升级也不等于跨数据库数据迁移。

建议按以下顺序进行：

1. 记录当前镜像版本、Compose 配置和实际数据卷名称。
2. 暂停写入，制作一致性备份；个人部署可先停止网关再备份 SQLite 数据卷，避免只复制正在写入的数据库主文件。
3. 准备目标数据库和专用账号，在独立测试实例中验证建表及数据迁移方法。
4. 迁移并核对主键、关联关系、用户、渠道、凭据、模型配置、额度与必要日志。
5. 验证登录、渠道测试、真实请求、日志记录和记账结果后，再切换正式实例。
6. 保留原数据和原配置作为回退依据；切换后若已有新写入，回退前还需处理两边数据差异。

数据库连接密码、API Key、OAuth JSON 等凭据应放在受控配置中；文档中的配置只应使用占位符。本次只核对了 SQLite 部署，尚未实际执行到 MySQL / PostgreSQL 的迁移。

四、接入普通模型 API

1. 在“渠道”中创建渠道，选择实际供应商类型。
2. 填写上游凭据，按需要设置 API 地址、模型列表和模型映射。
3. 配置允许访问的分组，并测试渠道。
4. 在“API 密钥”中创建给编程工具使用的令牌。
5. 在编程工具中填写网关地址、下游令牌和模型名称。

对于要求 OpenAI API base URL 的客户端，常见地址形态为 `http://localhost:3000/v1` ；是否带 `/v1` 必须以具体客户端的配置要求为准，不能把同一地址直接套用到所有工具。

“支持 OpenAI 兼容”不等于所有协议、模型和编程功能都完全兼容。 至少需要验证：

- 流式输出是否完整；
- 工具调用参数是否正确，多轮调用能否继续；
- 所需的 Chat Completions、Responses 或 Messages 接口是否受支持；
- 长上下文、缓存用量和错误处理是否符合预期。

项目列出了 Responses、Claude Messages、Gemini 等接口，但当前 README 仍将 Chat Completions 与 Responses 互转标为开发中。接口存在与任意上游可互转是两回事。官方功能说明

五、是否支持 ChatGPT Plus / Pro 账户

1. 当前版本有订阅账户渠道

在本地运行的 `v1.0.0-rc.35` 中文界面中，已确认存在：

渠道 → 创建渠道 → 类型 → ChatGPT Subscription (Codex)

页面要求使用 OAuth JSON 凭据。本次核对的入口是导入凭据，不是直接在 New API 中输入 ChatGPT 账号密码登录。

该版本源码也包含 OAuth 凭据解析、访问令牌刷新以及 Codex 请求适配实现。OAuth 凭据结构、刷新实现

2. 订阅权益与 API 计费不同

接入方式	凭据	使用范围与计费
OpenAI 渠道	OpenAI Platform API Key	按 API 账户及 API 规则计费
ChatGPT Subscription (Codex) 渠道	Codex 登录相关的 OAuth JSON	使用账户可用的 Codex 订阅权益，受实际账户权限和额度限制

OpenAI 官方说明，Codex 支持通过 ChatGPT 登录获得订阅访问，也支持通过 API Key 按量使用。Plus / Pro 包含 Codex 权益，但它们并不自动变成通用 OpenAI API 余额。官方认证说明、官方订阅权益

网关中的第三方订阅适配功能，不应理解为 OpenAI 对该网关全部使用方式的官方支持。

3. 当前订阅渠道的明确限制

本地页面提示该渠道仅限个人使用，不应分发或共享凭据，相关凭据与配置仅限接入 Codex CLI，不适用于其他客户端、平台或渠道。

源码中的适配器主要处理 Responses 请求，并明确拒绝直接调用 `/v1/chat/completions` 与 `/v1/messages` 。因此，不能把接入 Plus / Pro 理解为“可以给所有 AI 编程工具提供全部 ChatGPT 功能”。当前版本适配器

OAuth JSON 内含账户访问凭据，应像密码一样保管，通过可信的本地方式导入；不要提交到 Git、发布到文档或粘贴进聊天。官方对 Codex 认证缓存也给出了同类保管要求。凭据存储说明