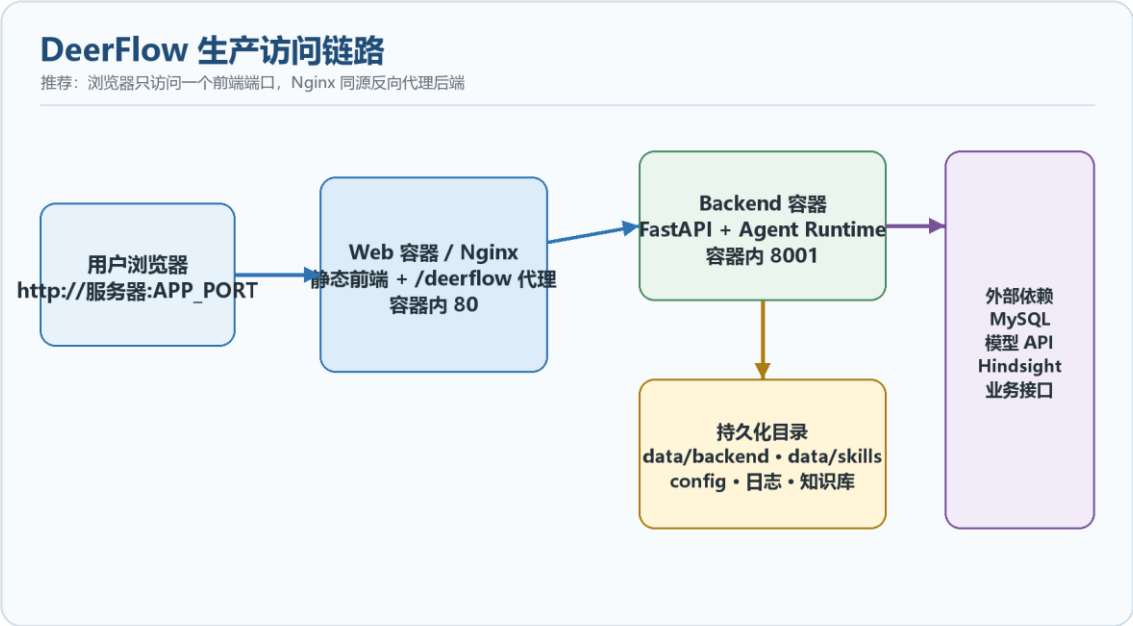
DeerFlow

系统运维手册

通用 Docker 容器版 · 配置、资产与故障处理优先



系统访问与数据流概览

> **本版重点** 以常规 Docker 容器运行环境为基准；将常见报错、config.yaml、.env、技能/智能体和前端管理设置提前，便于业务管理员直接查用。

文档版本	适用范围	更新日期
V1.1	当前 DeerFlow Docker 部署包及本项目定制功能	2026-08-08

密级：内部运维资料。密钥、密码、IP、数据库账号均必须使用占位符；不要写入本手册。

先用目录找到答案

本手册按“先排错、再配置、再维护”的顺序编排。打开 Word 后可使用左侧导航窗格按标题跳转；目录页码没有更新时，按 Ctrl+A 后按 F9。

阅读约定 4

1. 常见报错与第一时间怎么处理 5

1.1 先执行这组无损检查 5

1.2 容器、端口和磁盘 5

1.3 前端、Nginx 和接口 6

1.4 登录、模型、数据库和资产 6

2. config.yaml：模型、max_tokens 与后端配置 7

2.1 新增或修改一个 OpenAI 兼容模型 7

2.2 字段说明与常见误配 8

2.3 如何添加或调整 max_tokens 9

2.4 后端模型变更后的验收 9

3. .env、backend.env、数据库与前端接口地址 10

3.1 .env：项目名、镜像标签、访问端口 11

3.2 backend.env：JWT、模型密钥和数据库连接 11

3.3 config.yaml：登录密码与数据库模式 11

3.4 切换或排查外部 MySQL 12

3.5 前端接口地址与端口 12

4. 技能与智能体：放置、发布、丢失排查 13

4.1 技能放在哪里 14
4.2 智能体放在哪里 14
4.3 推荐的创建、上传和发布流程 15
4.4 手工导入时的安全做法 15
4.5 发布按钮或广场不可用 16
4.6 技能或智能体“丢失”专项排查 16
5. 前端管理：菜单名称、敏感词、主题与账号 18
5.1 配置菜单名称、排序和隐藏 18
5.2 配置敏感词：必须用哪个用户 19
5.3 主题色、布局和嵌入页面 19
5.4 必备账号与角色 20
6. 日常启停、日志和健康检查 21
6.1 常用命令速查 21
6.2 如何打印日志 21
6.3 健康检查和变更后验收 21
7. 数据库、备份、恢复与升级 23
7.1 必须备份的对象 23
7.2 文件和数据库备份示例 23
7.3 恢复与升级原则 23
8. 巡检、安全与交接 25
8.1 每日与每周巡检 25
8.2 最低安全要求 25

8.3 交接清单 25

附录 A：最短命令清单 27

附录 B：问题升级信息模板 27

附录 C：核心事实速记 28

> **最常用的五处** 页面或服务异常先看第 1 章；模型与 max_tokens 看第 2 章；登录、数据库和接口地址看第 3 章；技能/智能体问题看第 4 章；菜单、敏感词、主题看第 5 章。

阅读约定

- “部署目录”指存放 docker-compose.yml、.env、config 和 data 的当前运行目录。所有命令都应在此目录执行。
- 前端和后端不是同一个容器端口。用户通常只访问 APP_PORT；Web/Nginx 会把 /deerflow 请求转发到后端。
- 改配置前先备份原文件；改完只重启受影响服务；保存日志、健康检查和实际功能验证结果。
- 本文中的 <尖括号内容> 都是待替换占位符，不能原样复制到生产环境。

1. 常见报错与第一时间怎么处理

先收集状态再修改配置。大多数问题并非代码故障，而是当前目录错误、环境变量未重新加载、后端未启动、数据库连不上、权限不足或资产被过滤。

1.1 先执行这组无损检查

Shell

```
# 必须在部署目录执行
pwd
sudo docker compose ps
sudo docker compose logs --tail=200 backend
sudo docker compose logs --tail=200 web
curl -fsS http://127.0.0.1:<APP_PORT>/deerflow/health
```

> **先不要做** 不要执行 docker system prune -a；不要删除 data 后重启；不要创建同名技能或智能体覆盖可能的恢复线索；不要把完整 backend.env 发到群里。

1.2 容器、端口和磁盘

现象/报错	优先检查	处理方式
address already in use	APP_PORT 已被其他进程占用	用 ss -lntp 查占用；修改部署目录 .env 的 APP_PORT，然后重建 web。
容器不断 Restarting	backend 的最后 200 行日志	重点查看 YAML、环境变量、数据库 URL、挂载权限或启动命令；修复后再重建 backend。
no space left on device	磁盘、inode、data 和日志目录	先清理可确认的旧日志或无用缓存；不要在未备份时清理 data 或盲目 prune。
permission denied	宿主机挂载目录所有者和容器运行用户	核对 compose volume 与目录权限；只对目标目录做最小权限修正，不使用 chmod -R 777。
Web 访问失败	docker compose ps、APP_PORT、主机防火墙	确认 web 容器正常、端口监听，并放行实际访问端口。

1.3 前端、Nginx 和接口

现象	原因方向	解决
502 Bad Gateway	backend 未启动或 Web 无法访问 backend:8001	先查 backend 日志和 Compose 网络；后端恢复后再刷新前端。
刷新二级页面 404	Nginx 的 SPA 回退缺失	恢复 Nginx 的 try_files 到 /index.html 配置并重启 web。
页面白屏	runtime-config.js 语法或地址错误、缓存未刷新	浏览器 Console/Network 定位；恢复配置，重启 web，使用 Ctrl+F5。
所有 API 都失败	VITE_BACKEND_BASE_URL 或 VITE_LANGGRAPH_BASE_URL 配错	优先恢复 /deerflow 和 /deerflow/api 相对路径；见第 3.5 节。
CORS / Mixed Content	前后端不同源或 HTTPS 页面调用 HTTP 接口	优先同源代理；必须跨域时在后端配置允许 Origin 并统一 HTTPS。

1.4 登录、模型、数据库和资产

现象/状态	最常见原因	首要动作
401 / invalid_api_key	模型 Key 缺失、变量名不一致或已过期	对照 config.yaml 的 $变量名 与 backend.env；更新后强制重建 backend。
404 model not found	model 写成显示名或 base_url 路径不兼容	使用服务商实际 model ID；核对 API 是否要求 /v1。
429 / timeout	并发、配额、模型服务或网络问题	降低并发/重试，检查模型服务和网络连通性。
401 / 403 登录或发布失败	Token 过期、角色/所有者不足、广场未启用	重新登录；核对 system_role、资产所有者和目标广场。
技能/智能体看不到	目录未挂载、数据库切换、过滤、权限或发布状态	按第 4.6 节逐步排查，不要立即新建同名资产。
MySQL Access denied / timeout	账号、密码 URL 编码、host 授权、网络或端口	查看第 3.4 节连接格式和第 7 章数据库恢复原则。

2. config.yaml：模型、max_tokens 与后端配置

模型条目由 config/config.yaml 管理；密钥由 config/backend.env 管理。两处的变量名必须完全一致。新增模型时先写密钥，再写模型条目，再验证后端日志和实际对话。

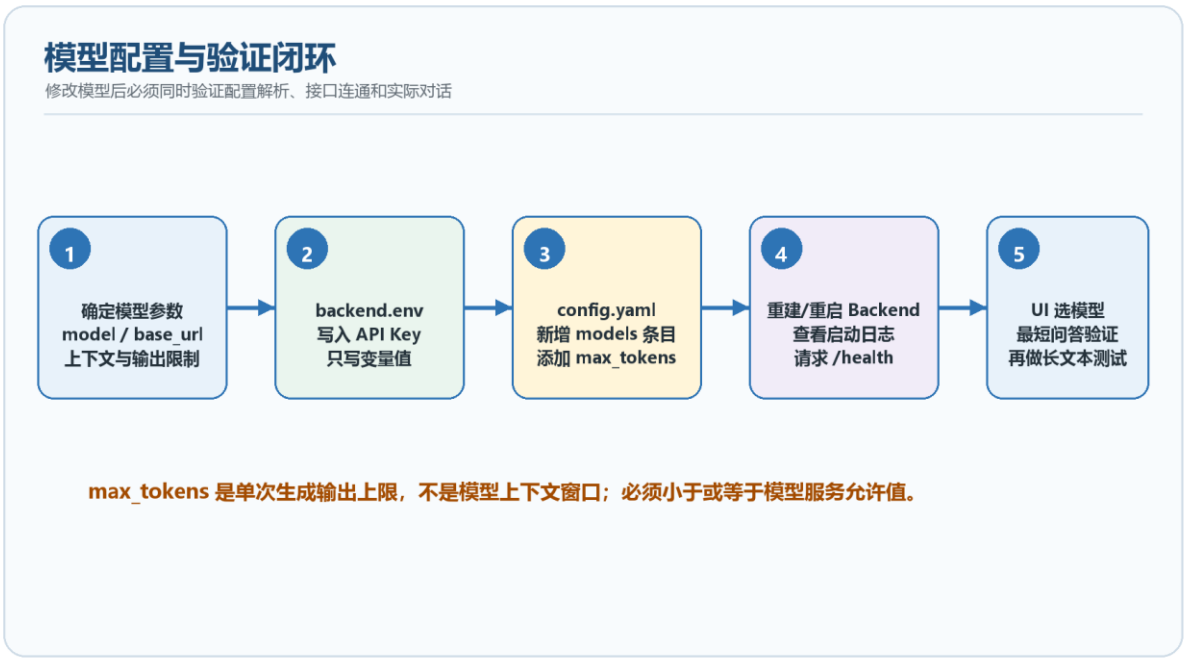


图 2-1 模型配置与验证闭环

2.1 新增或修改一个 OpenAI 兼容模型

1. 备份 config/backend.env 和 config/config.yaml，并记录要替换的模型名称。
2. 在 backend.env 写入密钥，变量名仅供后端读取。

ENV

```
# config/backend.env
INTRANET_MODEL_API_KEY=<服务商分配的密钥>
```

3. 在 config.yaml 的 models: 下增加或修改模型条目。name 是系统引用名；model 是服务商真实模型 ID；display_name 是前端显示名。

YAML

```yaml
models:
  - name: intranet-model
    display_name: 内网模型
    use: langchain_openai:ChatOpenAI
    model: <服务商实际模型 ID>
    api_key: $INTRANET_MODEL_API_KEY
    base_url: https://<模型服务地址>/v1
    request_timeout: 600.0
    max_retries: 2
    max_tokens: 4096
    temperature: 0.7
    supports_vision: false
    supports_thinking: false
```

4. 强制重建 backend 以重新读取 backend.env；随后查看启动日志、健康检查和前端模型列表。

Shell

```shell
sudo docker compose up -d --force-recreate backend
sudo docker compose logs --tail=300 -f backend
curl -fsS http://127.0.0.1:<APP_PORT>/deerflow/health
```

2.2 字段说明与常见误配

字段	用途	常见问题
name	系统内部模型名称，配置引用以它为准	重名会覆盖或引用混乱；不要随意改已被智能体使用的 name。
display_name	前端显示名称	改显示名不等于改 model；可用于给业务人员更易懂的名字。
use	适配器，例如 langchain_openai:ChatOpenAI	适配器与接口协议不匹配会导致初始化失败。
model	模型服务商要求的真实 ID	误填显示名经常造成 404 model not found。
api_key	使用 $变量名 从 backend.env 读取密钥	写明文、变量拼错或漏掉 $ 都会导致鉴权失败。
base_url	接口基地址	有的服务要求 /v1，有的路径已包含 /v1；以服务商接口文档为准。
request_timeout / max_retries	单次等待时间与失败重试次数	超时不等于模型不存在，先检查服务和网络。
max_tokens	单次生成输出上限	不是输入+输出总上下文窗口，不能超过服务商允许的最大输出。

2.3 如何添加或调整 max_tokens

在目标模型条目中添加或修改 max_tokens: <整数>。它控制单次回答最多生成多少 token。回答被截断时可在服务商允许范围内逐步增加；成本或响应时间异常时应适度降低。

现象	判断	处理
回答结尾被截断	max_tokens 太小，或服务端提前结束	先查看 finish_reason；确认服务商上限后逐步提高。
400 invalid max_tokens	超过模型限制或字段不兼容	按服务商模型上限修改；必要时删除该字段验证兼容性。
长文本仍失败	上下文不足，不只是输出上限	缩短输入、启用摘要或选择更大上下文模型。
费用/耗时异常	输出限制过高或提示词过长	降低 max_tokens；优化提示词和业务限流。

2.4 后端模型变更后的验收

- YAML 使用空格缩进，不能使用 Tab；密钥变量名必须同时存在于 backend.env 和 config.yaml。
- 修改 backend.env 后必须使用 docker compose up -d --force-recreate backend；普通 restart 不会重新读取环境文件。
- 只修改 config.yaml 时可重启 backend，但仍应查看日志确认配置解析成功。
- 先进行短问答验证，再验证长文本、工具调用、视觉输入或思考模式（如果该模型声明支持）。
- 记录模型 name、服务地址、变更人、验证时间；不要把 API Key 写入变更单或截图。

3. .env、backend.env、数据库与前端接口地址

部署目录的运行配置分为三类：.env 控制 Compose 与对外端口；backend.env 保存后端密钥和连接串；config.yaml 定义模型、登录、数据库模式等；runtime-config.js 控制前端运行时接口地址。不要进入容器直接改文件。



图 3-1 配置分层与生效方式

3.1 .env：项目名、镜像标签、访问端口

ENV

```
# 部署目录 .env
COMPOSE_PROJECT_NAME=deerflow
DEERFLOW_TAG=<当前镜像包对应标签>
APP_PORT=8090
TZ=Asia/Shanghai
```

变量	作用	修改后的操作
COMPOSE_PROJECT_NAME	Compose 项目名称，用于隔离容器、网络和卷名称	确认未与其他项目冲突；通常不在运行中随意修改。
DEERFLOW_TAG	docker-compose.yml 使用的镜像标签	应与已导入镜像的标签一致；镜像升级时核对交付清单和 SHA256。
APP_PORT	用户访问 Web/Nginx 的宿主机端口	修改后执行 docker compose up -d --force-recreate web；确认新端口开放。
TZ	容器日志和页面显示时区	重建受影响容器后生效。

3.2 backend.env：JWT、模型密钥和数据库连接

ENV

```
# config/backend.env
AUTH_JWT_SECRET=<至少 32 字节的随机值>
INTRANET_MODEL_API_KEY=<模型服务密钥>
MYSQL_DATABASE_URL=mysql+asyncmy://<USER>:<URL_ENCODED_PASSWORD>@<HOST>:3306/<DB>?charset=utf8mb4
```

> **密钥规则** AUTH_JWT_SECRET 是签发登录令牌的服务端密钥，不是用户登录密码。修改它会让所有已登录会话失效。backend.env 的任何改动都要强制重建 backend 才会重新读取。

3.3 config.yaml：登录密码与数据库模式

YAML

```
# config/config.yaml（示意；保留现有其他配置）
auth_login:
  username_login:
    enabled: true

database:
  backend: mysql  # 使用默认文件数据库时为 sqlite
  mysql_url: $MYSQL_DATABASE_URL
```

用户名登录的共享密码或账号管理项应遵循当前 config.yaml 的 auth_login 配置；单个用户的 login_password、system_role 和资产所有权以应用用户管理/数据库记录为准。不要把用户密码与 AUTH_JWT_SECRET 混为一谈。

3.4 切换或排查外部 MySQL

1. 冻结可能影响资产写入的操作，完整备份 data/backend、当前配置和现有数据库。
2. 创建数据库和最小权限账号，确认字符集 utf8mb4，并允许容器所在网络访问。
3. 密码中 @、:、/、#、% 等字符需要 URL 编码；HOST 必须是容器能访问的地址，不能把容器内 127.0.0.1 当成外部数据库。
4. 同时更新 backend.env 的 MYSQL_DATABASE_URL 与 config.yaml 的 database.backend/mysql_url，强制重建 backend 并持续查看迁移日志。
5. 分别用管理员、lqq 和普通用户验证登录、技能、智能体、菜单、敏感词和发布状态。SQLite 与 MySQL 不会自动双向同步。

3.5 前端接口地址与端口

JavaScript

```
// config/runtime-config.js：推荐同源相对路径
window.__DEERFLOW_RUNTIME_CONFIG__ = {
  VITE_BACKEND_BASE_URL: "/deerflow",
  VITE_LANGGRAPH_BASE_URL: "/deerflow/api",
  VITE_AIPI_BASE_URL: "<可选的业务接口地址>",
};
```

推荐保持相对路径。浏览器会自动访问当前域名和 APP_PORT，再由 Nginx 转到 backend:8001，这样最少遇到 CORS、证书与混合内容问题。确需使用独立后端域名时，应同步配置允许 Origin 和 HTTPS。

Shell

```
# 修改前端端口或运行时配置后的推荐操作
sudo docker compose up -d --force-recreate web
sudo docker compose ps
sudo docker compose logs --tail=200 web
# 浏览器使用 Ctrl+F5；在 Network 检查 runtime-config.js 和 /deerflow/health
```

4. 技能与智能体：放置、发布、丢失排查

技能和智能体能否在页面出现，取决于文件、数据库元数据、当前登录身份、发布状态和目录挂载是否一致。优先在前端创建/上传；仅在批量迁移或前端不可用时手工导入。

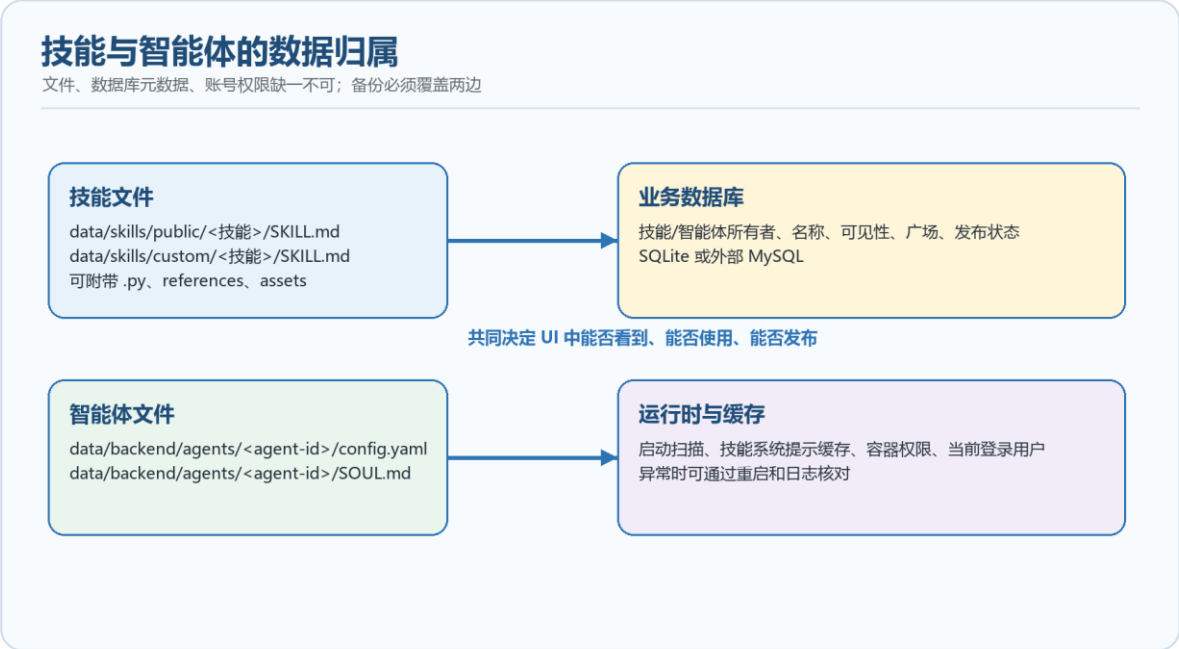


图 4-1 技能和智能体的持久化关系

4.1 技能放在哪里

目录结构

```
data/skills/
├── public/
│   └── <skill-id>/
│       ├── SKILL.md
│       ├── scripts/          # 可选
│       ├── references/       # 可选
│       └── assets/           # 可选
└── custom/
    └── <skill-id>/
        └── SKILL.md
```

目录	含义	注意
data/skills/public	系统共享/公共技能文件分类	放入 public 不等于已发布到广场；仍需有正确元数据、权限和发布状态。
data/skills/custom	用户自定义技能目录	适合从前端创建或上传的技能；应保留完整目录而不是只复制一个文件。
SKILL.md	技能说明、使用条件、操作规范的核心文件	避免写入真实 API Key、数据库密码、内网敏感地址或个人信息。
scripts/references/assets	脚本、参考资料和附件	脚本必须可执行且依赖完整；上传前检查许可证、体积和敏感信息。

4.2 智能体放在哪里

目录结构

```
data/backend/agents/<agent-id>/
├── config.yaml     # 模型、工具、技能、行为参数
└── SOUL.md         # 角色、提示词、表达和规则
```

智能体文件只是一部分。名称、所有者、可见性、广场和发布状态等业务元数据通常在 SQLite 或 MySQL 中。因此恢复或迁移智能体时，必须同时备份/恢复 data/backend 与对应数据库。

4.3 推荐的创建、上传和发布流程

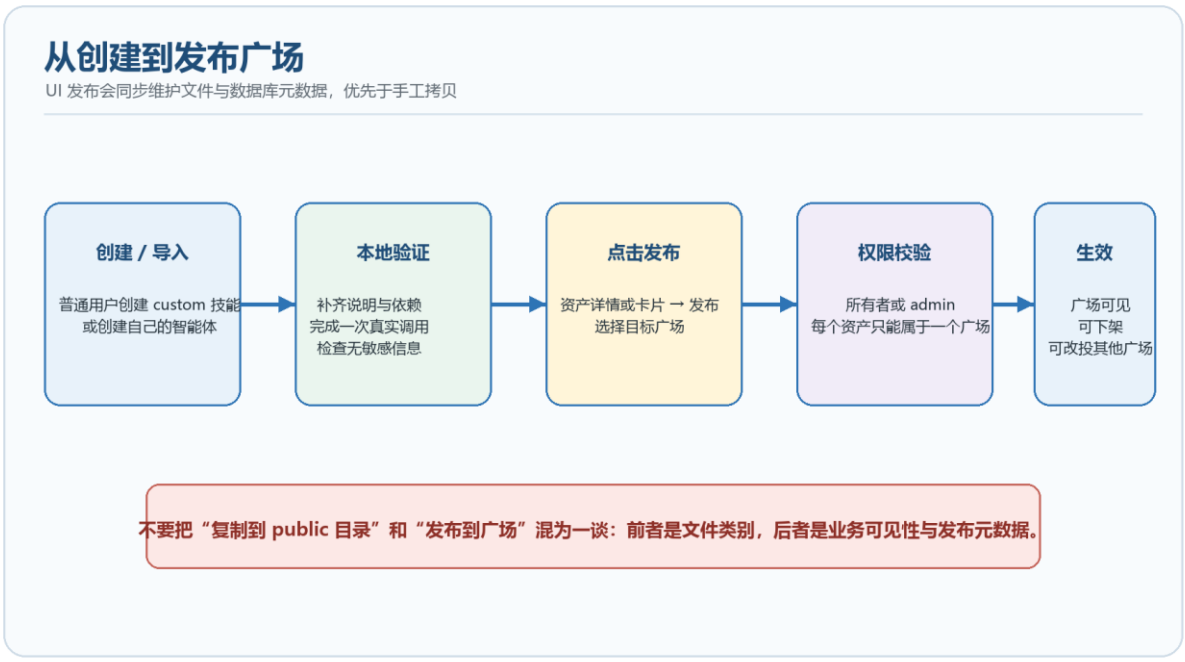


图 4-2 从创建到发布广场

1. 使用普通业务用户或管理员进入技能管理/智能体管理，创建或上传资产并填写名称、说明、依赖和使用范围。
2. 完成本地实际验证：技能要验证工具调用、脚本依赖和文件权限；智能体要验证所选模型、技能/工具和首轮问答。
3. 在资产详情或卡片中点击“发布”，选择一个启用的目标广场。资产所有者或 admin 才能发布/下架。
4. 发布后用普通用户验证广场可见性；重新发布到其他广场通常是迁移，不是多份复制。
5. 若只需要个人使用，保持私有或不发布；不要通过手工复制文件代替发布流程。

4.4 手工导入时的安全做法

Shell

```
# 示例：先将完整技能目录复制到 custom，再确认容器内可见
sudo mkdir -p data/skills/custom/<skill-id>
# 将经过审核的文件复制到上面的目录后执行：
sudo docker compose exec backend sh -lc 'find /app/backend/skills/custom -maxdepth 3 -type f -print'
sudo docker compose restart backend
```

> **手工导入后的边界** 手工复制只解决文件存在的问题，不会自动补齐所有者、可见性或发布元数据。导入后应通过界面核对资产记录；若无记录，按应用支持流程创建或恢复元数据。

4.5 发布按钮或广场不可用

问题	检查项	处理
看不到“发布”按钮	当前用户是否资产所有者或 admin	切换到所有者/admin；不要只凭文件目录判断权限。
选择不到广场	广场是否启用、是否存在默认/私有范围限制	由 admin 在发布广场管理中启用或创建目标广场。
发布后普通用户看不到	广场、私有状态、筛选、缓存和角色	确认实际发布到哪个广场；退出重登并清除筛选；检查发布 API 401/403。
智能体能看见但模型不可用	config.yaml 中模型 name 是否仍存在	恢复同名模型条目或将智能体改到已验证模型后重启 backend。

4.6 技能或智能体“丢失”专项排查

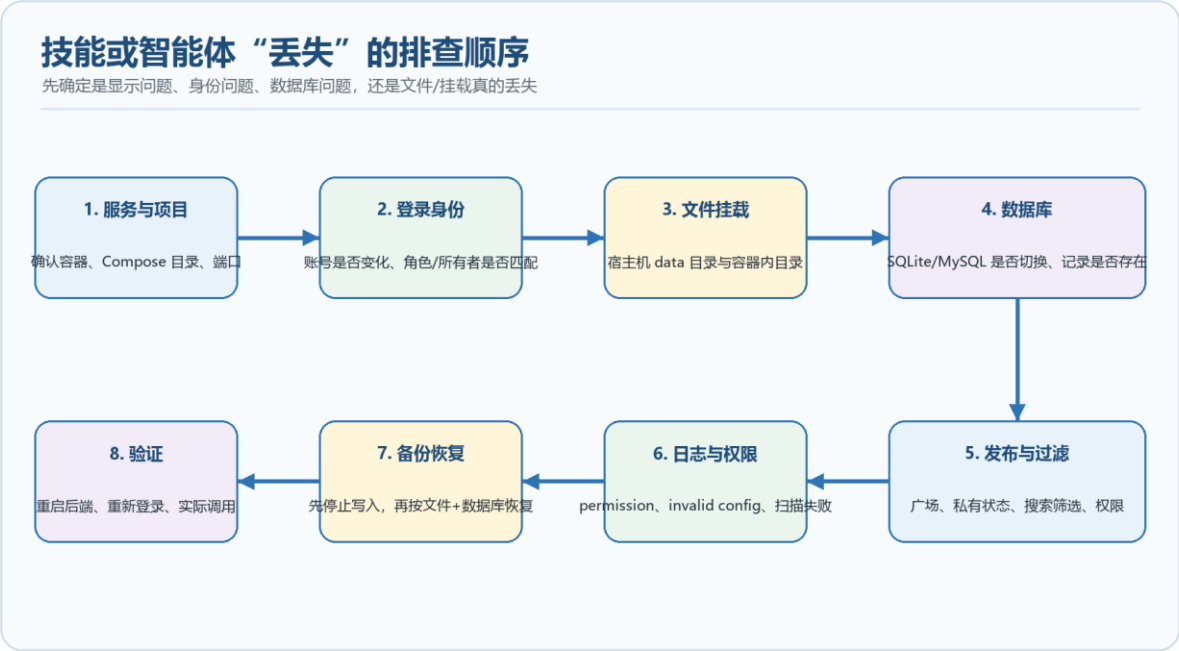


图 4-3 资产丢失排查顺序

1. 确认当前部署目录和服务：执行 pwd、docker compose ps，排除在旧目录启动了另一套服务。
2. 确认登录身份：重新登录，核对用户名、system_role、资产所有者和是否处于独立会话/嵌入会话。
3. 同时检查宿主机和容器内文件。宿主机有而容器没有，重点检查 docker-compose.yml 的 volume 映射与当前工作目录。

Shell

```
find data/skills -maxdepth 3 -type f -print | sort
find data/backend/agents -maxdepth 3 -type f -print | sort
sudo docker compose exec backend sh -lc 'find /app/backend/skills -maxdepth 3 -type f -print | sort'
sudo docker compose exec backend sh -lc 'find /app/backend/.deer-flow/agents -maxdepth 3 -type f -print | sort'
```

4. 确认数据库未切换：查看 config.yaml 的 database.backend 和 backend.env 的 MYSQL_DATABASE_URL。切换数据库不会自动搬迁或同步资产记录。
5. 检查过滤、广场和权限：清除搜索条件，切换“我的/公共/已发布/私有”等视图，并用 admin 核验发布记录。
6. 查看 backend 日志。文件和数据库都缺失时，停止写入，从备份恢复；重启 backend、重新登录并实际调用验证。

5. 前端管理：菜单名称、敏感词、主题与账号

菜单、主题、敏感词和广场主要通过应用管理页面维护，属于数据库中的全局设置。保存后通常立即生效，但应使用管理员和普通用户分别验证。

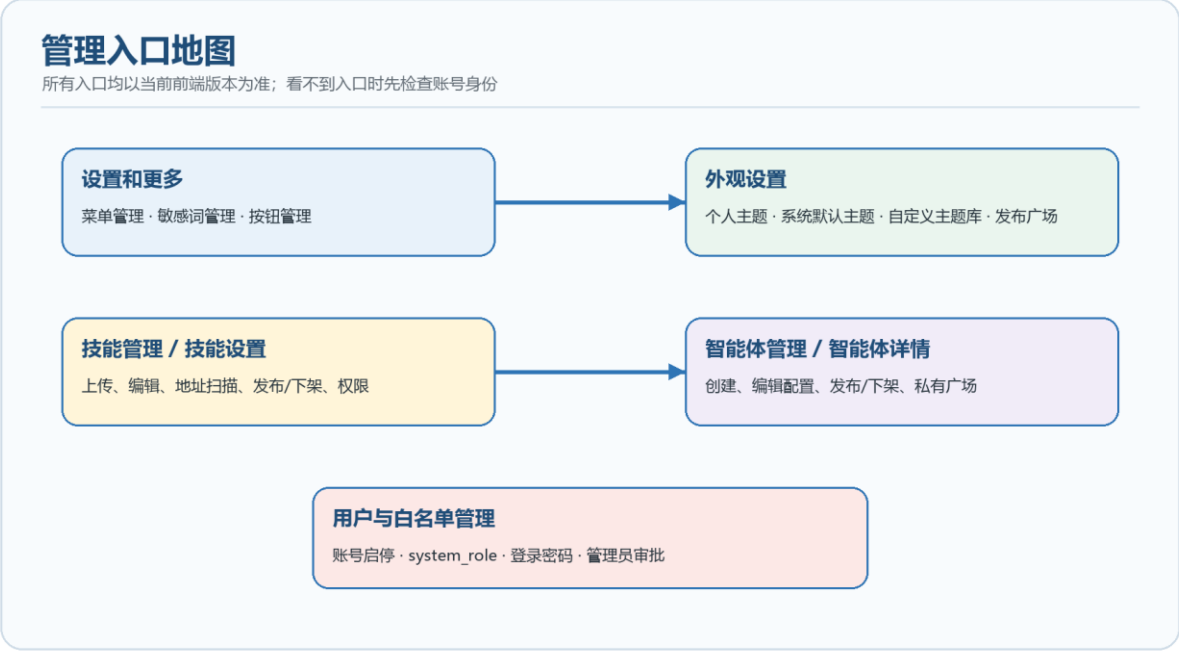


图 5-1 常用管理入口

5.1 配置菜单名称、排序和隐藏

1. 使用 system_role=admin 的 DeerFlow 管理员登录，打开左下角“设置和更多”中的“菜单管理”。
2. 找到目标一级、二级或三级菜单，修改显示名称 label、启用状态、排序或父级；保存后刷新页面验证。
3. 菜单停用会隐藏该节点及其子树；修改名称不要改真实 path、路由 ID 或代码常量。
4. 用普通用户检查左侧菜单、顶部导航和直接访问链接；记录修改前名称、目的和回退方式。

> **安全边界** 隐藏菜单只移除导航入口，不等于阻止用户通过已知 URL 或 API 访问。真正禁用功能还需后端 API 权限、路由守卫和角色权限共同生效。

5.2 配置敏感词：必须用哪个用户

敏感词功能是“展示层脱敏映射”：原始词 -> 替换词。它不会修改数据库里的原文，也不会改变 URL、路由 path 或节点 ID。

> **登录要求** 敏感词管理前端入口、路由守卫和页面级校验都要求当前显示用户名严格等于 lqq。即使 admin 拥有后端写权限，也应使用 lqq 登录来维护页面。邮箱账号时，@ 前缀必须为 lqq。

1. 用 lqq 登录，进入“设置和更多 -> 敏感词管理”。
2. 新增原始词与替换词，设置启用状态；需要初始规则时可使用“导入种子词”合并缺失映射。
3. 保存的是整套规则；修改前先导出/截图并经业务确认，避免替换词冲突或过度脱敏。
4. 刷新页面，检查菜单、导航、标题和普通内容的显示；用普通用户复核最终效果。

5.3 主题色、布局和嵌入页面

配置目标	操作入口/变量	验证
个人主题	设置 -> 外观，用户自己的选择通常优先	新开页面检查浅/深主题、文字对比度和焦点状态。
系统默认与主题库	admin 在外观设置中配置默认主题、主题名称、主色和启用状态	用没有个人覆盖的新测试用户验证。
强制紧凑侧边栏	runtime-config.js 的 VITE_FORCE_COLLAPSED_SIDEBAR	重启 web 并刷新浏览器后检查导航。
iframe 嵌入	URL 参数 embed=1、theme=...、hideJump=1、isolate=1	在嵌入页验证无全局导航、主题和独立会话行为。

> **主题验收** 不仅看按钮颜色。还要检查正文、表格、弹窗、下拉框、禁用、hover、focus、错误和成功状态的对比度；不要用品牌主色铺满大面积背景。

5.4 必备账号与角色

账号类型	至少准备	主要用途	注意
DeerFlow admin	2 个	菜单、主题、广场、用户和全局资产管理	system_role 必须为 admin；保留一个应急管理员。
lqq	1 个	敏感词管理前端入口	显示用户名/邮箱前缀必须严格为 lqq。
普通测试用户	1 个	验收菜单隐藏、资产可见性和广场发布	不能只用管理员验收，管理员会掩盖权限问题。
模型/数据库服务账号	按环境	调用外部服务	遵循最小权限和定期轮换，不与个人账号共用。

6. 日常启停、日志和健康检查

一键脚本和 Docker Compose 命令都应从部署目录运行。日常 restart 适合重启进程；任何环境变量文件改动都必须通过 force-recreate 重建受影响容器。

6.1 常用命令速查

Shell

```
# 整套服务
sudo bash restart.sh
sudo bash stop.sh
sudo bash logs.sh
sudo bash diagnose.sh

# 只处理后端或前端
sudo docker compose restart backend
sudo docker compose up -d --force-recreate backend   # backend.env 改动后
sudo docker compose restart web
sudo docker compose up -d --force-recreate web       # .env 的 APP_PORT / Web 配置改动后
sudo docker compose ps
```

6.2 如何打印日志

Shell

```
# 全部服务日志；Ctrl+C 仅停止跟踪，不停止服务
sudo bash logs.sh

# 只看后端或 Web/Nginx
sudo bash logs.sh backend
sudo bash logs.sh web

# Compose 原生命令：最后 500 行并持续跟踪
sudo docker compose logs --tail=500 -f backend
sudo docker compose logs --tail=500 -f web
```

6.3 健康检查和变更后验收

Shell

```
curl -fsS http://127.0.0.1:<APP_PORT>/deerflow/health
curl -I http://127.0.0.1:<APP_PORT>/
sudo docker compose ps
```

- 容器 Up 不等于业务可用：健康接口成功后仍要登录一次、发起一次对话并验证修改的功能。
- backend.env、.env 改动：force-recreate；config.yaml 改动：至少 restart backend；runtime-config.js 改动：重启 web 并强制刷新浏览器。

- 变更完成后继续观察后端日志至少 5 分钟，确认没有反复重启、401/429/500、数据库错误或模型初始化失败。

7. 数据库、备份、恢复与升级

一次可恢复的备份至少包含 config、data 和当前业务数据库。只保存 Docker 镜像无法恢复用户、技能、智能体、发布广场、菜单、主题与敏感词。外部 MySQL 必须单独导出。

7.1 必须备份的对象

对象	作用	建议频率	恢复价值
config 与 .env	模型、登录、数据库、接口地址和运行配置	每次变更后	恢复服务连接和端口配置。
data/backend	文件数据库、智能体文件及运行数据	每日 + 变更前	恢复智能体、用户和运行相关资产。
data/skills	public/custom 技能文件和附件	每日 + 技能变更后	恢复技能内容和脚本。
data/knowledge	知识库文件	按使用频率	恢复知识资产。
外部 MySQL dump	资产元数据、用户、菜单、主题、广场等	每日 + 变更前	恢复数据库层资产与全局设置。

7.2 文件和数据库备份示例

Shell

```
# 文件备份：确保备份目录位于独立存储，不在当前 data 内
cd <部署目录>
sudo tar -czf <备份目录>/deerflow-files-$(date +%F-%H%M).tar.gz .env config data
sha256sum <备份目录>/deerflow-files-*.tar.gz

# MySQL 备份：密码不要直接写到命令历史，优先使用受控 defaults-extra-file
mysqldump --single-transaction --routines --triggers       -h <MYSQL_HOST> -u <BACKUP_USER>
<DB_NAME> > <备份目录>/deerflow-mysql-$(date +%F-%H%M).sql
```

7.3 恢复与升级原则

- 恢复前先确认备份版本、SHA256、数据模式和目标目录；停止或冻结写入，避免恢复过程中产生新分叉数据。

- 先恢复到隔离或测试环境验证，再进行正式切换；恢复后检查目录权限、数据库连接、健康接口、登录、模型、技能、智能体和广场。
- 升级时保留旧镜像、旧配置和完整备份；新版本使用独立测试端口验证，确认无误再切换。
- SQLite 与 MySQL 的资产数据不会自动双向同步。任何切换、迁移或恢复都要有明确来源和一次性合并计划。

> **高风险操作** 删除部署目录、删除 data、覆盖未知数据库、docker system prune -a 和递归修改所有者都可能造成不可逆损失。执行前必须明确目标、验证备份可用并获得审批。

8. 巡检、安全与交接

8.1 每日与每周巡检

频率	检查内容	合格标准
每日	docker compose ps、/deerflow/health、磁盘/inode、backend/web 日志	服务正常、无重启循环、无连续 5xx/401/429、空间充足。
每周	备份是否完成、最近恢复演练、模型配额与超时、资产发布与权限变更	备份可校验；高频失败有负责人；变更有记录。
每次变更后	端口、日志、健康、登录、实际业务功能、普通用户权限	功能验收通过且可回退。

8.2 最低安全要求

- AUTH_JWT_SECRET、模型 API Key、数据库 URL/密码只保存在受控的 backend.env 或密钥系统中，不进入 Git、截图、聊天记录和前端 runtime-config.js。
- 管理员、lqq、普通测试用户和服务账号分开管理；定期轮换密码/密钥，离职或交接时立即回收不再需要的权限。
- 对外只开放必要的 APP_PORT；后端 8001 默认不直接暴露。数据库地址、模型服务和业务接口应采用最小网络范围。
- 对外发送日志前删除 Authorization、Cookie、Token、API Key、数据库 URL、手机号和身份信息，只保留时间、状态码、trace ID 与关键报错。

8.3 交接清单

项目	交接时填写
生产访问地址 / APP_PORT	
当前部署目录与 Compose 项目名	
镜像版本 / SHA256	

项目	交接时填写
数据库模式、地址与库名（不写密码）	
DeerFlow admin、lqq、普通测试用户责任人	
模型服务与密钥保管责任人	
备份位置、保留期与最近恢复演练日期	
监控、告警和问题升级联系人	

附录 A：最短命令清单

Shell

```
cd <部署目录>
sudo docker compose ps
sudo docker compose logs --tail=500 -f backend
sudo docker compose logs --tail=500 -f web
sudo docker compose restart backend
sudo docker compose up -d --force-recreate backend
sudo docker compose restart web
sudo docker compose up -d --force-recreate web
curl -fsS http://127.0.0.1:<APP_PORT>/deerflow/health
```

附录 B：问题升级信息模板

故障单模板

```
【问题标题】
【发生时间】YYYY-MM-DD HH:mm:ss（时区）
【访问地址/端口】<不含密码>
【当前账号角色】admin / lqq / 普通用户（不要提供密码）
【操作步骤】1... 2... 3...
【实际现象】
【期望结果】
【HTTP 状态码 / trace ID】
【容器状态】docker compose ps
【已脱敏日志】backend/web 最近 200 行
【最近变更】模型、数据库、菜单、主题、镜像或配置
【已采取动作】以及结果
```

附录 C：核心事实速记

主题	结论
前端和后端端口	容器内部不同：Web 80、Backend 8001；用户通常经由 APP_PORT 访问 Web。
配置文件	.env 管端口/Compose；backend.env 管密钥和连接串；config.yaml 管模型/登录/数据库；runtime-config.js 管前端地址。
环境变量生效	backend.env 或 .env 改动后必须 force-recreate 受影响容器；普通 restart 不会重新读取。
模型与 max_tokens	模型条目在 config.yaml；Key 在 backend.env；max_tokens 是单次输出上限，不是上下文窗口。
技能位置	data/skills/public 与 data/skills/custom；目录文件存在不等于已经发布。
智能体位置	data/backend/agents/<agent-id>；还依赖数据库元数据、所有者、可见性和发布状态。
敏感词用户	页面维护需要显示用户名严格为 lqq；admin 不能替代该前端入口条件。
菜单隐藏	只隐藏导航入口，不等于禁止直接 URL 或 API，需要后端权限配合。
资产丢失	先查目录、挂载、数据库、身份、过滤和日志，最后才恢复；不要急着创建同名资产。
完整备份	config + data + 外部 MySQL dump + 镜像版本/SHA256；只备份镜像不足以恢复业务资产。