

INIT MANUAL · v4.2

never sur-render

初始化手册 · 写给 AI agent

> 读者：Claude Code、Codex、Cursor、Grok 等能读文件、能跑终端命令的 AI agent
> 任务：问用户一组问题，按回答为用户生成专属的 agent 文档，装好依赖和提供商，定好账号定位与品牌视觉，跑通试片，建好每天的灵感收集和复盘任务

DON BLUFF LLC · 2026

目录 Contents

00 致 AI agent：你要做的七件事 3

01 给 agent 的规矩 4

02 安装终端工具、取内容包、装依赖 5

03 一次性定制问卷 13

04 按回答生成用户的 agent 文档 17

05 试片验收与第一条真片（只用基础版） 25

06 定时任务：晨间灵感 + 每晚复盘 28

07 交付说明、授权（两档）与客服 30

08 进阶：分镜、Remotion 效果、标题封面、进阶剪辑 完整版 33

附 排错与更新 34

标记说明

内容包目录　在内容包目录（`nsrender basic` / `nsrender unpack` 写出来的目录，第 02 节）的根目录里执行。

nsrender　终端工具 `nsrender` 的命令，在哪个目录都能跑。

尚未实现　`nsrender init` 和 `nsrender run` 还没有实现：照实告诉用户，由你按本手册直接做。

花钱 / 对外 · 先问　会扣费或会公开发布的动作，必须先得到用户明确同意。

用户新建　内容包里本来没有、由你按本手册为用户创建的文件或目录。

完整版　只在完整版内容包里有的 skill 或文件。基础版本机没有它们：跳过或用默认值，不要假装有（第 04 节「先查档位」）。

命令、脚本和路径都能在内容包目录里找到；`nsrender` 的命令以 `nsrender --help` 为准。

00

致 AI agent：你要做的七件事

你好，AI agent。never sur-render 是一套“无脸短视频”流水线：选题 → 写稿 → 配音与逐字字幕 → AI 画面 + 代码动效 → 配乐音效 → 封面 → 发布 → 按反馈自我改进。

你的用户刚买下它。先读第 01 节的规矩并全程遵守，再按顺序完成：

#	你做什么	产出
1	检查环境、登录取包（先问哪一档）、装依赖；选配音、画面、发布的提供商，让用户自己把 key 填进 `.env`	`pnpm check-env` 无 FAIL
2	把定制问卷一次发给用户（可分组），收回答；用户拿不准的，你给 2–3 个方案让用户选	问卷回答
3	按回答生成用户自己的 agent 文档，定下账号定位和品牌视觉（图片 / 视频）	`SOUL.md` 、 `MEMORY/` 、 `AGENTS.md` 、 `TOOLS.md` 、 `config/profile.yaml` 、skills 启用表
4	跑不花钱的试片，打开审片页给用户看	`report.json` 里 `check_ok: true`
5	建两个定时任务：晨间灵感 / 选题收集，每晚复盘	用户平台上的两条定时任务
6	用 5–10 行交付：改了什么、用了哪些默认值、还缺什么、下一步	交付说明

两层文档：用户文档优先

第一层 · 用户文档（你按问卷生成，以它为准）

`SOUL.md` （品味、口吻、画面、红线）· `MEMORY/` （定位决定、灵感、复盘记录）· `config/profile.yaml` （所有频道设置）· `AGENTS.md` 里的「This channel」和「Owner rules」两段 · `TOOLS.md` 里的「This channel」一段 · `docs/FEEDBACK-LOG.md` （用户的每条反馈）

第二层 · 内容包（参考库）

`skills/` （选题、写稿、画面、剪辑手法、发布、自我改进的工作流）· `templates/` · `AGENTS.md` / `TOOLS.md` 的其余部分 · `INIT.md` · `docs/PRODUCTION-SOP.md` · `engine/README.md` 。它告诉你“通常怎么做好”，但不知道这位用户是谁。内容包分基础版和完整版两档（第 07 节）。

冲突时：用户文档 > 内容包；用户最新的话 > 用户文档里的旧条目（照新的做，并改掉旧条目，见 `skills/self-improve/SKILL.md` ）。内容包更新时不覆盖第一层。内容包由 `nsrender` 按版本下发，`nsrender basic` / `nsrender unpack` 把它写成内容包目录里的明文文件，你读和改的都是这个目录。

01

给 agent 的规矩

密钥只进 `.env`，你不读、不打印

`cp .env.example .env`（`.env` 已被 .gitignore 忽略），让**用户自己**用编辑器把 key 填进去。不要让用户把 key 贴进对话；不要 `cat` / `echo` .env；不要把 key 写进任何会提交的文件、日志或文档。你只检查“有没有”：`pnpm run profile` 和 `pnpm check-env` 只显示 set / not set。YouTube 的 OAuth token 文件放在内容包目录外。

花钱的动作先问 先问

MiniMax 配音（按字符扣费）、克隆音色、用 Grok 订阅额度生成图片 / 视频、买任何套餐或充值，都要先说清楚“做什么、大概花多少”，等用户明确说可以。配音只在用户批过稿子之后（`approval.script_before_voice`）。

对外的动作先问 先问

发帖、上传、改公开状态、删帖，只在用户看过并批准这条成片之后（`approval.video_before_publish`）。定时任务里永远不放发布。

用户文档优先于内容包

见第 00 节。用户纠正过的事，写进文档或代码，让下次不会再错（`skills/self-improve/SKILL.md`）。

不确定就问，不要猜

用户没回答的项先用默认值，并在交付说明里逐条列出。

产出只进 `out/` 或 `episodes/`

这两个目录不提交。提交稿子、`shots.json`、场景代码和文档，不提交音视频。不要删 `out/<slug>/assets/`：重新生成要花用户的 Grok 额度。

不假装有还没实现的命令

`nsrender init` 和 `nsrender run` 尚未实现 现在跑不了。照实告诉用户，初始化由你按本手册直接做。其余命令以 `nsrender --help` 列出的为准。

不假装有完整版 skill 完整版

文件在本机磁盘上才算有。不在就用基础版做法，告诉用户一次“这部分在完整版里”，不编造它的内容。

每一步都汇报

做完一步就告诉用户改了哪些文件、跑了哪些命令、结果如何。命令失败就贴出原样报错，不要说“应该没问题”。

02

安装终端工具 · 取内容包

1. 装 nsrender nsrender

先要有 Node.js 22 以上（自带 `npm`；没有就按「依赖：要装什么」一页先装）。然后：

```
npm i -g https://acfdywgmrfzptuepufcj.supabase.co/functions/v1/ns-public/cli/nsrender.tgz
nsrender --version        # 打印版本号就是装好了
nsrender doctor           # 检查 node、ffmpeg、Chromium、claude / codex、grok、MiniMax key
```

安装地址是官方下载入口，不在 npm 仓库里，所以要写完整地址。以后升级终端工具，重新运行同一条 `npm i -g` 。

2. 取内容包 nsrender

先问用户要哪一档（下一页）。**两档都要先用 DonBluff 账号登录**；内容包写进一个目录，下面统一叫**内容包目录**：

```
# 两档都先登录。没账号：用户到 https://www.donbluff.com/login?next=/never-surrender 用邮箱注册
nsrender login --email 你的邮箱    # 发邮件登录链接，在这台电脑的浏览器里打开

# 基础版（免费）：登录后直接装
nsrender basic                     # 装到 ./never-sur-render-basic

# 完整版：账号里已有完整版码且有空名额时，上面的 login 已把本机激活，不用再输码
nsrender status                    # 确认本机档位是完整版
nsrender activate                  # 只在账号里还没有这个码时：用户本人在自己的终端输入授权码
nsrender unpack                    # 解到 ./never-sur-render
```

要装到别的目录，两条命令都能加 `--dir <目录>`；目录要是空的，或上次装过同一种内容包的目录。

3. 在内容包目录里装依赖 内容包目录

```
cd never-sur-render-basic    # 完整版：cd never-sur-render；本手册后面的步骤都在这个目录里做
pnpm bootstrap     # 装依赖（--frozen-lockfile）+ Playwright Chromium + 环境自检
pnpm check-env     # 每一项 ok / warn / FAIL，非 ok 的下一行给出修复方法；不改任何东西
pnpm test          # 引擎单元测试，应全部通过
pnpm run profile   # 当前读的是哪个配置文件、引擎会用的值、哪些 key 已设置（不显示值）
```

02

取包：先问基础版还是完整版

问卷之前做。先问用户一句：**“这台电脑要用基础版（免费），还是完整版?”**按回答走下面一条，不要替用户选。

先登录（两档都要）

1. 问用户有没有 DonBluff 账号、注册邮箱是哪个。**没有账号**：请用户自己打开 `https://www.donbluff.com/login?next=/never-surrender` 用邮箱注册，注册好再继续。你不能替用户注册， `nsrender` 也不会创建账号。
2. `nsrender login --email 用户的邮箱` （你这里没有交互终端，**必须带 `--email`** ，不带会报“没法询问邮箱”并退出）：发登录邮件，等浏览器回调（最多 300 秒）。请用户在**跑这条命令的同一台电脑**的浏览器里打开邮件里的链接。
3. 不要让用户把登录链接、token 或密码贴给你。登录成功后 `nsrender` 会自动查账号里的授权（下面 B）；账号里没有码时出现输入授权码的提示：基础版不需要，直接回车跳过。

A. 基础版（免费，登录后装，不用授权码）

- `nsrender basic` ：下载签名的基础版内容包，校验签名后写进 `./never-sur-render-basic` 。基础版不用跑 `nsrender activate` 。
- 没登录就跑，它在下载之前停下并提示先登录：照提示注册 / 登录，再跑一次。

B. 完整版

1. 授权码在**官网购买后**发到用户邮箱，也能在官网「我的授权码」页看到（ `https://www.donbluff.com/never-surrender/dashboard` ）。登录要用**购买时的 DonBluff 账号**邮箱，换账号这个码用不了。
2. **登录就会激活本机**： `nsrender login` 成功后自动检查账号里的授权。账号里已经有兑换过的完整版码、且这个码还有空名额时，本机直接激活为完整版并装好内容包，终端显示「本机已激活，内容包 …」：这时**不用**再跑 `nsrender activate` 、不用再输码。这台电脑占这个码的 1 个名额（下下页「设备名额」）。
3. `nsrender activate` ：只在账号里还没有这个码（新买的码第一次用）时才需要。请**用户自己在自己的终端里**运行，在提示处输入授权码（输入时不显示）；不要让用户把码贴给你。你这里没有交互终端：本机没激活时它报“没有交互终端”并退出，已激活时显示「本机已激活…」并正常退出（附录排错）。
4. `nsrender status` 确认本机档位是完整版，再 `nsrender unpack` 把完整版内容包解到 `./never-sur-render` 。
5. **码不落地**：不把码写进任何文件、配置、 `.env` 、 `MEMORY/` 、 `SOUL.md` 、交付说明、提交或日志；不要把码写在命令参数里；汇报时只说“已用完整版码激活”，不复述码。

02

取包之后：档位、升级、其他命令

C. 以本机为准

装完跑 `nsrender status`。没登录时它显示“未登录”，并提示基础版怎么拿（注册 → 登录 → `nsrender basic`）、完整版要购买的授权码。登录后显示账号；本机已激活为完整版的（登录时自动激活，或用 `nsrender activate` 输过码），还显示本机档位和内容包版本。只装了基础版的设备显示「授权：未激活」和「内容包：基础版 v…，装在 <目录>」，属于正常（`nsrender update` 只用于授权码激活的设备，基础版用不上）：**档位以内容包目录里的文件为准**（第 04 节「先查档位」），不以用户说的为准。激活报错（码无效 / 已停用 / 名额已满）就贴出原样报错，先按 A 装基础版；名额已满见下一页。

以后升级到完整版

- **还没购买**：请用户自己打开 DonBluff 官网的 Never Surrender 页（`https://www.donbluff.com/never-surrender`）。你不报价格、不代付款；用户想先用起来就按 A 装基础版，以后再升级。
- 在要升级的这台电脑上：新买的码第一次用，用户在自己的终端运行 `nsrender activate` 输入完整版码；账号里已经有兑换过的码且有空名额，重新 `nsrender login --email 邮箱` 一次，登录时自动激活，不用输码。然后 `nsrender status` 确认是完整版 → `nsrender unpack --dir <现在的内容包目录>`（例如 `never-sur-render-basic`）。这台设备占 1 个完整版名额（每个码最多 2 台）。
- 第一次把完整版解到 `nsrender basic` 装的目录时，包内文件会换成完整版的版本：先把 `AGENTS.md` / `TOOLS.md` / `docs/FEEDBACK-LOG.md` 里用户写的部分另存，解完再填回去。`SOUL.md`、`MEMORY/`、`config/profile.yaml` 这些用户自己的文件不会被碰。以后再 `nsrender unpack`，你改过的包内文件保留、不覆盖。
- 升级后让 AI **重新读本手册的完整版部分**：第 03 节问卷四、第 04 节「先查档位」和启用表、第 08 节进阶。

其他 nsrender 命令 nsrender

```
nsrender whoami          # 当前登录的邮箱
nsrender status          # 账号、本机档位（基础版 / 完整版）、内容包当前 / 上一版 / 最新版、依赖摘要
nsrender manual          # 手册 PDF 在哪：授权码激活过的设备给本机路径，其余给官网地址
nsrender update          # 用授权码激活的设备：更新到最新版；之后再 nsrender unpack
nsrender rollback        # 回到上一个版本并固定（同 update --rollback）
nsrender logout          # 只清本机登录和解密材料；本机的设备名额保留，换电脑要先解绑（下一页）
```

引导问卷 `nsrender init` 和调度工作流的 `nsrender run <skill>` 尚未实现。初始化由你（AI agent）按本手册直接做：读内容包目录里的 `skills/`，问用户问题，生成文档。

02

设备名额：登录、退出登录、换电脑

每个完整版码最多 2 台设备、终身 2 次解绑。名额跟着“登录并激活过的电脑”走。 nsrender

登录就会占名额

- `nsrender login` 成功后会自动检查账号里的授权：账号里已经有兑换过的完整版码、且这个码还有空名额时，**本机直接激活为完整版并装好内容包**，终端显示「本机已激活，内容包 …」。这时不用再跑 `nsrender activate` 、不用再输码；接着 `nsrender status` 确认档位，再 `nsrender unpack` 。
- **副作用：在别人的电脑上登录，也会占这个码的一个名额。**登录前先告诉用户。只是临时用、或这台电脑只要基础版： `nsrender login --email 邮箱 --no-activate` （登录后不自动激活本机、不下载内容包），再 `nsrender basic` 。
- 只有账号里还没有这个码（新买的码第一次用）时，才需要用户本人在交互终端运行 `nsrender activate` 输入完整版码。**名额满了的码不会让新设备变成完整版**：登录时报名额已满，本机不变；对同一个码再跑 `nsrender activate` 也一样被拒。先解绑一台（下面），或输入另一个还有空名额的完整版码。

退出登录：本机名额保留

`nsrender logout` 只清本机登录和解密材料，**本机的设备名额保留**。同一台电脑重新登录直接认回原来的名额，不占新名额、不扣解绑次数。所以退出登录**不能**把名额让给别的电脑。已发布的 nsrender 1.0.3 退出登录后提示的“请在账号页解绑”，指的就是下面的设备管理页。

换电脑、名额满了：先解绑旧电脑

1. 请用户登录官网，打开**设备管理页**：`https://www.donbluff.com/never-surrender/dashboard/devices` 。在那里看这个码的 2 个名额占了几台，自助解绑一台不用的电脑。名额已满、设备已解绑的提示里写的也是这一页。
2. 解绑**计入终身解绑次数：每个完整版码 2 次**，用一次少一次，先和用户确认要解绑哪一台。
3. 解绑后在新电脑上 `nsrender login --email 邮箱` ：自动激活，占回这个名额；再 `nsrender status` 、`nsrender unpack` 。
4. 解绑次数用完了：请用户写信给客服 **aceking@mail.grokbot.com**（第 07 节「客服」），附上 `nsrender status` 的输出。

> **看名额、解绑都在设备管理页：** `https://www.donbluff.com/never-surrender/dashboard/devices` ，用购买时的 DonBluff 账号登录。「我的授权码」页（ `https://www.donbluff.com/never-surrender/dashboard` ）只用来看授权码，不能解绑。

02

依赖：要装什么

组件	用途	要求 / 怎么装
Node.js	引擎、测试、脚本	≥ 22（`package.json` → `engines`）。macOS：`brew install node@22`；其他系统按 nodejs.org 官方说明装 Node 22+
pnpm	依赖与所有 `pnpm …` 命令	锁定 9.15.4：`corepack enable && corepack prepare pnpm@9.15.4 --activate`
ffmpeg / ffprobe	编码、混音、烧字幕；必须带 libx264	macOS：`brew install ffmpeg`（缺 libx264 时 `brew reinstall ffmpeg`）；Debian / Ubuntu：`sudo apt-get install -y ffmpeg`；不在 PATH 上时在 `.env` 设 `FFMPEG_PATH` / `FFPROBE_PATH`（见表下说明）
Playwright Chromium	代码动效、封面、字幕烧录	`pnpm bootstrap` 会装；单独补装：`pnpm --filter @nsr/engine exec playwright install chromium`；想用别的浏览器时在 `.env` 设 `BROLL_CHROME`
AI agent CLI	执行 skills、跑定时任务	用户已经在用的那个即可（例如 Claude Code `claude`、Codex `codex`），按其官方说明安装并由用户本人登录
Grok Build CLI（`grok`）	AI 静帧、AI 视频片段、封面底图	按 xAI 官方说明安装。登录要**用户本人在自己的交互终端**里运行 `grok` → `/login`（会打开浏览器登录自己的 SuperGrok 账号）；脚本、CI 和你的非交互终端里做不了，你不能替用户登录，只能请用户做完再继续。不在 PATH 上时在 `.env` 设 `GROK_BIN`。没装、没登录也能出片：AI 镜头用 `--skip-gen` 渲染成雾状占位
Python 3（可选）	上传 YouTube：`publish/youtube_upload.py`	依赖：`pip install google-api-python-client google-auth google-auth-oauthlib`（见该文件头部说明）

`GROK_BIN`、`FFMPEG_PATH`、`FFPROBE_PATH`、`BROLL_CHROME` 写进内容包目录的 `.env` 就生效（内容包 1.0.8 起），也可以在 shell 里 `export`；1.0.7 及更早的内容包只认 `export`。详见「填 key、记预算」一页。

> 装任何东西之前先看 `pnpm check-env` 的结果，只装它报 FAIL / warn 而用户确实需要的项。用 Homebrew / apt 装系统软件前，告诉用户你要装什么；需要 `sudo` 的命令请用户自己执行。

Remotion 效果库 `skills/editing-craft/remotion-lab/` 完整版 只是参照样片，主渲染引擎不依赖它，两档初始化时都不用装。

02

依赖：逐项验证

`pnpm check-env`（`scripts/check-env.mts`）就是验收表。它逐行输出 `ok` / `warn` / `FAIL`，并在非 ok 行下面给出修复命令；有 FAIL 时退出码为 1。

行	查什么	不通过时
`node`	版本 ≥ 22	FAIL → 升级 Node
`pnpm`	是否 9.15.4	其他版本 warn，没有 FAIL → corepack 那行命令
`dependencies`	`node_modules` 和引擎的 playwright 是否已装	FAIL → `pnpm install --frozen-lockfile`
`ffmpeg` / `ffprobe`	能否找到（含 `FFMPEG_PATH` / `FFPROBE_PATH`）	FAIL → 按上页安装
`libx264`	ffmpeg 是否带 libx264 编码器	FAIL → 换带 libx264 的 ffmpeg
`grok`	能否找到；Grok 的登录凭据文件在不在（`GROK_AUTH_PATH`，或 `$GROK_HOME/auth.json`，默认 `~/.grok/auth.json`；内容包 1.0.8 起）。`GROK_BIN` 指向别处的程序（包装脚本、另一个账号）时不报 warn，写明没检查登录（`login not checked`）和原因（附录排错）	warn → 用户本人在交互终端 `grok` → `/login`（你做不了），或先用 `--skip-gen`
`chromium`	真的启动一次 Chromium	FAIL → `pnpm --filter @nsr/engine exec playwright install chromium`
`profile`	读的是不是 `config/profile.yaml`	warn → 第 04 节生成它
`minimax`	`MINIMAX_API_KEY` 是否设置（只看有没有）	warn → 只有用 MiniMax 配音时才需要
`agent docs`	`SOUL.md` 是否已生成	warn → 第 04 节生成它

初始化前后的期望

- 装完依赖时：没有 FAIL。`profile`、`agent docs` 两行是 warn 属正常，第 04 节做完就会变 ok。
- 全部做完时：除了用户明确不用的项（例如不用 Grok、自己录音），其余都是 ok；`pnpm test` 全部通过。

终端工具的自检 nsrender

`nsrender doctor` 在任何目录都能跑：检查 node、ffmpeg / ffprobe、Chromium（Playwright）、`claude`、`codex`、grok 和 MiniMax key，同样只看有没有、不打印值。进了内容包目录以后以 `pnpm check-env` 为准，它查得更细。

02

API 提供商：按用户选择配置

先问用户想怎么配音、画面从哪来、发哪些平台，再只配置用到的那几项。引擎目前实际接好的只有下表“已接入”的几项，其他服务不要假装能接进来。

环节	选项	登录 / key（用户自己填）	费用与注意
配音	已接入 **MiniMax TTS**：配音 + 逐字时间戳	`.env` → `MINIMAX_API_KEY`、`MINIMAX_API_HOST`（国际站 `https://api.minimax.io`，国内站 `https://api.minimaxi.com`）	按字符计费 先问；规则见 `docs/MINIMAX_TTS.md`
	自己录音（或用其他配音工具生成音频）	无	从剪辑软件导出 SRT，作为真实时间轴；出片时传 `--voice <mp3> --subs <srt>`
画面	已接入 **Grok Build CLI**：AI 静帧、图生视频、封面底图	用户本人在自己的交互终端 `grok` → `/login`（你不能代登录）；可选 `.env` → `GROK_BIN`	消耗用户的订阅额度 先问。引擎用的是订阅登录，**不要改成 xAI API key**
	代码动效 / 氛围镜头（`motion`、`fog` 等）	无	免费，本地渲染
	自己录的素材（`footage` 镜头：录屏、产品演示）	无	文件放 `videos/<slug>/`，见 `skills/faceless-broll/SKILL.md`
发布	已接入 **YouTube**：`publish/youtube_upload.py`（含自定义封面）	`.env` → `YOUTUBE_TOKEN_FILE`（OAuth token，放内容包目录外）、`YOUTUBE_CATEGORY_ID`	上传配额；未通过 Google 审核的项目只能传成私享 对外
	X	`.env` → `X_API_KEY`、`X_API_SECRET`、`X_ACCESS_TOKEN`、`X_ACCESS_TOKEN_SECRET`	内容包没有 X 发帖脚本：用户自己的发帖工具，或手动发；API 不能选视频封面
	TikTok / Instagram Reels	平台官方 API（如有权限）	没有 API 权限就把文件和文案交给用户手动发

02

填 key、记预算

让用户自己填 key

1. `cp .env.example .env`，告诉用户：用编辑器打开 `.env`，只填要用的那几行，保存。不需要贴给你。
2. 提醒用户在提供商控制台设好用量上限或预算提醒。
3. 验证只看有没有：`pnpm run profile`（打印 `MINIMAX_API_KEY`、`YOUTUBE_TOKEN_FILE`、`X_API_KEY` 的 set / not set）和 `pnpm check-env` 的 `minimax` 行。
4. 把选定的提供商、用途、预算写进 `TOOLS.md` 的「This channel」一段（第 04 节）。

```
cp .env.example .env    # 只复制模板；之后由用户自己编辑
pnpm run profile        # …  MINIMAX_API_KEY  set / not set（只显示有没有）
pnpm check-env          # minimax 行：ok = 已设置；warn = 没有 key（只在用 MiniMax 配音时需要）
```

可选的工具路径覆盖（也在 `.env`）

变量	什么时候设
`GROK_BIN`	`grok` 不在 PATH 上
`FFMPEG_PATH` / `FFPROBE_PATH`	要用指定的 ffmpeg 构建
`BROLL_CHROME`	不用 Playwright 自带的 Chromium，改用本机浏览器

这四个变量写进内容包目录的 `.env` 就生效（**内容包 1.0.8 起**），也可以在 shell 里 `export`；两边都有时以环境变量为准。`pnpm check-env` 和 `nsrender doctor` 读的是同一套。**1.0.7 及更早的内容包只认 `export`**，写在 `.env` 里不生效：先更新内容包，或改用 `export`。

预算怎么管

- 花钱的环节只有三处：MiniMax 配音（按字符）、克隆音色、Grok 订阅额度（AI 图和视频）。渲染、动效、声音都在本机免费完成。
- 每次花钱前先说明这次大概用多少（例如“稿子约 650 字，配一次音”“这条有 9 个 AI 镜头”），用户同意再做。
- 估算长度不花钱：`pnpm broll estimate videos/<slug>/script.txt`；看 Grok 会调用几次不花钱：`pnpm broll make videos/<slug> --dry-run`。
- Grok 生成的素材按提示词缓存在 `out/<slug>/assets/`，重跑只生成新增或改动的镜头——不要删这个目录。

03

定制问卷 · 一

整理成一条消息发给用户（可分组）。用户拿不准时，你按已有回答提 2–3 个具体方案让用户挑。→ 后是写入的文件和字段。第 1–31 题两档都问；第 32–37 题只在完整版问（第四页）。

A　账号定位

1. 频道叫什么？一句话：讲什么、给谁看。
   例：“60 秒讲清一条科技新闻，给北美华人上班族”
   → profile `channel.name` / `channel.niche`；SOUL「Who we make videos for」; MEMORY/channel.md
2. 希望观众怎么形容你？立场是什么？
   选项：做足功课的朋友 / 冷静的分析师 / 犀利但讲理 / 幽默；中立不荐股 / 有观点但给依据
   → profile `channel.positioning`；SOUL「Voice」
3. 对标账号或参考视频（2–3 个链接），各自喜欢它哪一点？
   → SOUL「Approved reference」; MEMORY/channel.md
4. 账号的差异点：别人都在做的，你不做什么、多做什么？
   拿不准时，你根据 1–3 题写 3 个定位方案（一句话定位 + 典型选题 + 封面标题样例），让用户选一个
   → MEMORY/channel.md（定位决定及理由）

B　受众与语言

5. 观众主要在哪些国家 / 地区？
   例：US、CA、EU；或 CN、TW、SG
   → profile `audience.regions`
6. 口播用什么语言？字幕单语还是双语，第二行用什么语言？
   例：中文口播 + 中英双语字幕；英文口播 + 单语字幕
   → profile `audience.languages`、`subtitles.primary_lang` / `secondary_lang` / `segmenter_locale`；`subtitles.max_chars`：中文约 16，拉丁字母约 42；第二行怎么出：第 05 节第 5b 步
7. 典型观众是谁、怎么看视频？
   例：通勤时手机上、半注意力、不开声音也要看懂
   → SOUL「Viewer」
8. 绝对不碰的话题或角度？
   例：具体个股买卖建议、某国内政、医疗诊断
   → profile `audience.avoid_angles`；SOUL「Angles to avoid」

C　内容支柱

9. 3–5 个内容支柱？
   例：AI 与科技 / 市场 / 宏观；或 健身 / 饮食 / 睡眠
   → profile `content.pillars`
10. 需要片尾免责声明吗？写什么？念不念出来？
    → profile `content.disclaimer.lines` / `voiced`
11. 单条多长？
    引擎默认分档：60–180 秒为短片，3–8 分钟为禁区，8 分钟以上为长片（`engine/src/estimate.ts` 的 `BAND_EDGES`，可按频道改）
    → SOUL「Who we make videos for」; 需要时改 `BAND_EDGES`

03

定制问卷 · 二

D　平台

12. 发哪些平台？各平台账号名？
    可选值： `youtube_shorts` / `youtube_long` / `x` / `tiktok` / `instagram_reels` / `other`
    → profile `platforms[]`： `name` 、 `enabled` 、 `handle` 、 `format`
13. YouTube 的视频类别？
    22 人物与博客 / 25 新闻与政治 / 27 教育 / 28 科技
    → profile `platforms[].category_id`

E　节奏与时区

14. 每天几条、几点发、哪几天发？用哪个时区？
    例：每天 08:30 和 12:30，周一到周日，America/New_York（IANA 时区名）
    → profile `schedule.slots` / `days` / `timezone` ， `audience.timezone` ；AGENTS.md「This channel」

F　品牌视觉 · 图片

15. 竖屏 9:16 还是横屏 16:9？
    9:16 → 1080×1920；16:9 → 1920×1080；帧率默认 30
    → profile `visual.width` / `height` / `fps` ， `platforms[].format`
16. 画面风格？
    内置三个预设只是例子： `clean` 彩色、中性（**默认**）； `ink` 黑白水墨插画； `noir` 黑白电影剧照。多数频道应该写自己的：用三个词描述风格（例：“柔和粉彩、扁平插画、留白多”），再给 2–3 张喜欢的参考图
    → profile `visual.default_style` 、 `visual.custom_styles` ；SOUL「Look」；第 04 节「品牌视觉」测试流程
17. 品牌色、固定元素、绝对不要出现的东西？
    例：主色深蓝 + 一点橙；不出现真人脸、品牌 logo、可读文字、血腥画面
    → SOUL「Look」；自定义风格的 `avoid`
18. 要不要独立封面图（推荐：YouTube Shorts 不用第一帧做封面）？封面只放主标题？
    → profile `cover.enabled` / `title_only` / `upload_as_thumbnail` ；SOUL「Cover」

G　品牌视觉 · 视频

19. 镜头怎么动？
    例：缓慢推近、平稳不切镜；或手持感、轻微晃动
    → 自定义风格的 `video_suffix` ；SOUL「Look」
20. 画面组成偏好？
    AI 图 + 慢镜头为主 / 代码动效（图表、数字、对比）为主 / 两者交替；AI 视频片段每段不超过约 13 秒
    → SOUL「Look」；写分镜时按 `skills/faceless-broll/SKILL.md`
21. 声音和节奏（基础）？
    例：配乐极简、转场音只在故事转折处；剪辑偏快 / 偏稳。更细的节奏和组接方法是第 32 题 完整版
    → SOUL「Sound」； `skills/editing-craft/SKILL.md`
22. 交付成什么？
    直接出成片 mp4（默认）/ 出剪映（CapCut）草稿自己再剪
    → MEMORY/decisions.md；每条视频的 `shots.json` → `"output": "capcut"`

03

定制问卷 · 三

H　配音

23. AI 配音（MiniMax）还是自己录?
    → TOOLS.md「This channel」; profile `voice.provider: minimax`
24. MiniMax：系统音色还是已克隆的音色 ID? 模型? 语速（常用 1.0–1.3）? 语气? 账号是国际站还是国内站?
    语气可选: `fluent` / `calm` / `happy` / `sad` / `angry` / `fearful` / `disgusted` / `surprised`；克隆新音色要花钱 先问
    → profile `voice.voice_id` / `model` / `speed` / `emotion` / `language_boost`；`.env` → `MINIMAX_API_HOST`（用户自己填）
25. 有没有常读错的词（多音字、缩写、品牌名）?
    配音时用 `pnpm broll voice … --tone "处理/(chu3)(li3)"` 指定读音
    → MEMORY/decisions.md「读音表」

I　预算与审批

26. 每月预算上限，按项目分：MiniMax、Grok 订阅、其他? 快到上限时怎么提醒?
    → TOOLS.md「This channel」; MEMORY/decisions.md
27. 稿子要先批再配音吗? 成片要先看再发吗?
    默认两个都要，强烈建议保持
    → profile `approval.script_before_voice` / `video_before_publish`

J　灵感来源与定时任务

28. 选题和灵感从哪里找?
    例：X 热搜、YouTube 趋势、Reddit 某版块、微博热搜、Reuters / Bloomberg、行业媒体、指定关注的 5–10 个账号、关键词
    → profile `content.sources`；具体账号和关键词写 MEMORY/inspiration.md 顶部
29. 晨间灵感 / 选题几点给你? 每晚几点复盘? 结果发到哪里（对话、邮件、聊天工具）?
    默认：第一个发布时段前 2–3 小时；复盘 21:00；都按 `schedule.timezone`
    → MEMORY/decisions.md; 第 06 节的定时任务
30. 你的 agent 在哪里跑定时任务?
    本机 cron / Claude Code / Codex / Grok Bot 例行任务 / 其他
    → 第 06 节; MEMORY/decisions.md
31. 复盘要看哪些数字? 你能怎么给我?
    播放、完播率、点赞、新增关注；内容包没有自动拉平台数据的脚本，需用户截图 / 粘贴，或另行授权的数据来源
    → MEMORY/decisions.md; SOUL「What the numbers say」

03

定制问卷 · 四 完整版

这一组依赖完整版 skill。**先按第 04 节「先查档位」确认本机是哪一档：**完整版才问；基础版**整组跳过、不发给用户**，按右列默认值做，并在 `MEMORY/decisions.md` 记一行“进阶题未问（基础版），用默认值”。

K　进阶（只在完整版问）

32. 剪辑节奏进阶：开剪前的剪辑计划要多细？偏好哪些镜头语言和组接方法？
    基础版默认：只按 `skills/editing-craft/SKILL.md` 的节奏、强调、转场规则，和 Q21 的回答
    → SOUL「Sound」「Look」; `MEMORY/decisions.md`; `skills/editing-craft-pro/SKILL.md`
33. 动效 / 手法库：常用哪类库场景？每期新写的场景要不要当天收回库？
    基础版默认：只用入门示例的 4 个 motion 场景（ `library/STARTER.md` ），不合就现写
    → `MEMORY/decisions.md`; `skills/editing-craft/library/INDEX.md`
34. Remotion 效果：强调和转场效果要克制还是明显？哪些效果不要用？
    基础版默认：引擎自带的 `fx_in` / `fx_out` 和基础规则里的强调做法
    → SOUL「Look」; `skills/remotion-effects/SKILL.md`
35. 分镜镜头：每种手法要不要写成配方卡？光效和整画面冲击能接受多少？切点要不要严格落在底乐节拍上？
    基础版默认： `skills/faceless-broll/SKILL.md` 的镜头类型和字段，不用配方卡
    → `MEMORY/decisions.md`; `skills/shot-design/SKILL.md`
36. 标题 / 封面进阶：标题偏“说清楚”还是偏“钩子”？要不要备一套封面，发布后数据差就换？
    基础版默认： `skills/script-writing/SKILL.md` §1c 的封面规则和 Q18
    → SOUL「Cover」; `skills/thumbnail-design/SKILL.md`
37. 脚本设计：开头要几版候选给你挑？结构偏好（冲突先行 / 结论先行）？
    基础版默认： `skills/script-writing/SKILL.md` 的流程和结构
    → SOUL「Voice」; `skills/script-design/SKILL.md`

收齐后，把回答原文存进 `MEMORY/channel.md` 末尾的「初始化问卷原文」，再开始第 04 节。没回答的题用默认值（括号里的或 `config/profile.example.yaml` 里的非示例值），记下来，交付时列出。基础版跳过的第 32–37 题也算“用默认值”，交付时一句话带过即可。

04

生成用户的 agent 文档 · 总览

文件	从哪来	写什么	内容包怎么辅助
`config/profile.yaml`	`cp config/profile.example.yaml config/profile.yaml`	问卷 A–J 里对应字段；只写 `config/profile.schema.json` 里有的字段	引擎和所有 skills 从这里读频道设置（`engine/src/profile.ts`，缺的字段用默认值）
`SOUL.md`	`templates/SOUL.template.md`	口吻、观众、画面、声音、封面、参考、红线；≤ 80 行	模板给出结构和质量线；`pnpm check-env` 检查它是否存在
`MEMORY/` 用户新建	本手册「TOOLS.md 与 MEMORY/」一页	定位决定、各种选择的理由、灵感池、复盘记录、定时任务的任务文本	内容包不含 MEMORY；skills 产出的简报、复盘结论落到这里
`AGENTS.md`	内容包已有	填「This channel」；新增「Owner rules」；阅读顺序加上 `MEMORY/README.md`	其余部分（工作流表、硬规矩）是内容包，原样保留
`TOOLS.md`	内容包已有	新增「This channel」：选了哪些提供商、预算、谁登录了什么	工具表（用途、认证、费用）是内容包，原样保留
skills 启用表	写在 `AGENTS.md` 「This channel」下	只列本机有的 skill（先查档位）、哪段要按频道改写	`skills/*/SKILL.md` 是工作流本身
`docs/FEEDBACK-LOG.md`	内容包已有	删掉注释里的示例条目；以后每条反馈记一条	`skills/self-improve/SKILL.md` 规定格式

顺序

```
cp config/profile.example.yaml config/profile.yaml
cp .env.example .env                    # 如第 02 节还没做；key 由用户自己填
# 写 profile.yaml → SOUL.md → MEMORY/ → AGENTS.md → TOOLS.md → 清 FEEDBACK-LOG 示例
pnpm run profile                        # 必须显示 profile: …/config/profile.yaml，且没有 EXAMPLE 提示
pnpm check-env                          # profile 和 agent docs 两行变成 ok
```

收尾自查：不留占位（应该没有输出；有就补完）

```
grep -rn -e '{{' -e '«' -e 'EXAMPLE' -e 'example:' SOUL.md AGENTS.md TOOLS.md
config/profile.yaml MEMORY/
# 内容包 1.0.8 起只会查出没替换的示例值；1.0.7 及更早：profile.yaml 第一行注释也会被查出来，删掉那一行
```

04

config/profile.yaml

从 example 复制后逐项改。 «Qn» = 填问卷第 n 题的答案；写完后文件里不能再有 « 、 EXAMPLE 、 example: 。字段说明： config/profile.schema.json 。

```yaml
channel:
  name: "«Q1 频道名»"
  handle_placeholder: "«Q12 主账号 @handle»"
  niche: "«Q1 一句话：讲什么、给谁看»"
  positioning: "«Q2 口吻与立场»"
audience:
  languages: [«Q6 口播语言代码，如 zh»]
  regions: [«Q5»]
  timezone: «Q14 IANA 时区»
  avoid_angles: [«Q8»]
platforms:   # 只列 Q12 选了的；
             # name 取 schema 枚举值
  - name: youtube_shorts
    enabled: true
    handle: "«Q12»"
    category_id: "«Q13»"
    format: "9:16"
schedule:
  slots: [«Q14，如 "08:30"»]
  days: [«Q14，mon…sun»]
  timezone: «Q14»
content:
  pillars: [«Q9»]
  sources: [«Q28»]
  disclaimer:      # 不需要就删掉
    lines: [«Q10»]
    voiced: «Q10 true/false»
```

```yaml
visual:
  default_style: «Q16 风格名»
  # ↑ clean（默认）/ ink / noir / 自定义名
  width: «Q15»
  height: «Q15»
  fps: 30
  custom_styles: {}
  # 自定义风格在「品牌视觉」
  # 一页测试选定后再写
subtitles:
  primary_lang: «Q6»
  secondary_lang: «Q6 第二行；单语删掉»
  max_chars: «中文约 16，拉丁约 42»
  segmenter_locale: «Q6»
voice:
  provider: minimax
  model: «Q24，默认 speech-2.6-hd»
  voice_id: "«Q24 系统音色或克隆音色 ID»"
  speed: «Q24»
  emotion: «Q24»
  language_boost: «Q24，如 Chinese»
  sample_rate: 32000
  bitrate: 128000
cover:
  enabled: «Q18»
  title_only: «Q18»
  upload_as_thumbnail: «Q18»
approval:
  script_before_voice: «Q27»
  video_before_publish: «Q27»
```

自己录音的用户： voice_id 不能留示例值，否则 pnpm run profile 会警告；没有音色时可以整段删掉 voice 。 config/profile.yaml 默认被 .gitignore 忽略。 approval 后面还有 reference 和 outro 两段，见下一页。

04

config/profile.yaml · reference 与 outro

接上一页，写在 approval 后面。 reference 是用户自己的范本库和改稿学习（基础 skill voice-reference 、 revision-learning 读它）， outro 是片尾怎么收。本手册的问卷里没有单独的题：**初始化时一般保持默认**；用户有往期定稿、或对片尾收法有要求时，按 INIT.md 第 20b / 20c 题（范本库、改稿学习）和第 14c 题（片尾怎么收）问，再改这里。

```
reference:                # 用户自己的往期稿，写新稿前先读（skills/voice-reference）
  enabled: true
  root: ""                # 空 = studio/<频道 slug>/voice-reference
  registry: registry.json  # root 里的登记文件
  copy_text: true          # 每篇登记的稿子在 <root>/samples/ 留一份原文
  min_read: 2              # 写新稿前至少读几篇已确认 / 已发布的范本
  auto_register:           # 只管自动登记；手动 broll reference add / confirm / publish 总会登记
    on_finalize: true      # pnpm broll finalize … --yes 成功时（用户批了这条成片）
    on_publish: true       # 发布工具拿到回执后；false = 照常上传，只是不登记
  revision:                # 改稿学习：初稿对用户的定稿，记成问题清单（skills/revision-learning）
    enabled: true
    lessons: lessons.json  # 问题清单，root 里的一个 *.json 文件名；写下一条前先读
    comparisons: revisions # root 里的目录，每期存一份初稿对定稿的对照
outro:
  transition: auto         # 最后一镜在 shots.json 没写 fx_out 时怎么收：auto | fade | glow | cut_black
```

- **reference.root**：留空 = studio/<频道 slug>/voice-reference 。要改也**必须在 studio/ 、episodes/ 或 out/ 下面**，这样用户的稿子不会被打进任何内容包。 lessons 和 comparisons 都在 root 里面。写完跑 pnpm broll reference check 确认路径合法。
- **已有往期定稿的用户**： pnpm broll reference add videos/<slug> 登记成候选，再请用户确认（ pnpm broll reference confirm … --by <谁> --why "<用户原话>" ）。用户没有真的确认过，不许登记成已过审。
- **库是空的就是空的**：新频道第一条真片没有范本、也没有问题清单，命令会照实说为空。跳过即可，不要编。
- **outro.transition**： auto 按片尾情绪自动选； fade 淡出、 glow 发光、 cut_black 切黑是固定收法。某条片子要单独处理，就在它的 shots.json 最后一镜写 fx_out 。

04

SOUL.md 与 AGENTS.md

SOUL.md（从 `templates/SOUL.template.md` 生成，放内容包目录的根目录）

模板段落	填什么
中文摘要	一句话：这个频道为谁、讲什么、什么叫好（Q1、Q4）
Who we make videos for	频道与定位（Q1）、市场语言（Q5、Q6）、避开角度（Q8）、观众（Q7）、时长（Q11）、审批（Q27）
Voice	口吻和立场（Q2），免责声明的处理（Q10）
Look	默认风格一句话描述（Q16）、尺寸帧率（Q15）、品牌色与禁用元素（Q17）、镜头运动和画面组成（Q19、Q20）
Sound · Cover · Approved reference · What the numbers say	Q21 · Q18 · 对标账号（Q3），以后加上第一条用户满意的视频文件夹 · 先留空，只由每晚复盘按数据填写

替换所有 `{{…}}`，删掉模板开头的说明注释，≤ 80 行。

AGENTS.md：改四处，其他不动

```
## This channel  <!-- filled by INIT.md step 3 -->

- Channel: «Q1» · Platforms: «Q12 启用的平台» · Slots: «Q14 时段» «Q14 时区»
- Skills（«基础版 / 完整版»，«日期» 核对）：启用表见「skills 启用表」两页，只列本机有的

## Owner rules (override the library)

- 本频道的规矩优先于 skills/ 和 templates/ 的通用做法；冲突时照这里做。
- «问卷里的硬性要求，一条一行»
- 引用的 skill 不在本机就不假装有：用基础版做法，提示一次「这部分在完整版里」

## Read in this order

`SOUL.md` → `MEMORY/README.md` → this file → `TOOLS.md` → top of `docs/FEEDBACK-LOG.md` → the skill for the task. …
```

- 删掉「This channel」原来的第二行（“Not initialized yet? …”）；工作流表只留本机有的 skill，基础版删掉 editing-craft 行里的「Remotion effects library」。
- 「This channel」「Owner rules」两段属于用户文档：内容包更新时保留它们，只合并其余部分。

04

TOOLS.md 与 MEMORY/

TOOLS.md：在表格后追加一段

```
## This channel

| What | Choice | Who signed in / where the key is | Budget / month |
| --- | --- | --- | --- |
| Narration | «Q23: MiniMax 国际站 / 国内站 / 自己录音» | .env → MINIMAX_API_KEY | «Q26» |
| Picture | «Grok Build CLI / 只用代码动效» | owner ran grok → /login | «Q26» |
| Publishing | «Q12 平台: 脚本上传 / 手动» | .env → YOUTUBE_TOKEN_FILE | — |
| Schedules | «Q30 平台»: morning «时间», nightly «时间» | — | — |
```

MEMORY/ 用户新建

内容包原本没有这个目录。它是用户的长期记忆：记“发生过什么、为什么这么定”，不放密钥（是否提交进用户的私有仓库由用户定）；规则和品味仍按 `skills/self-improve/SKILL.md` 的分工写进 SOUL / AGENTS / skill。

文件	初始条目	以后谁写
`MEMORY/README.md`	每个文件放什么；新条目在上、带日期	你
`MEMORY/channel.md`	定位决定（选了哪个方案、为什么、放弃了哪些）、对标账号、问卷原文	定位调整时
`MEMORY/decisions.md`	风格、配音、交付形式、预算、定时任务、数据来源、读音表等决定，带日期和理由	每次用户拍板
`MEMORY/inspiration.md`	顶部：关注的账号、关键词、来源（Q28）；下面：灵感条目（日期、标题、热度数字、链接、对应支柱）	晨间任务
`MEMORY/reviews/`	每晚一份 `<日期>.md`	每晚复盘
`MEMORY/prompts/`	`morning.md` 、 `nightly.md` ：两个定时任务的任务文本（第 06 节）	你

```
# MEMORY/decisions.md 的一条

## «日期»

- 决定: 默认风格用 «风格名»（4 张测试图里选了第 2、3 张的方向）
- 理由: «用户原话»
- 影响: profile → visual.default_style; SOUL.md → Look
```

04

skills 启用表 · 先查档位、完整版 skill

每个 `skills/<名字>/SKILL.md` 是一个工作流（纯 Markdown，开头的 `description` 写了何时使用）。内容包分两档，**本机实际有哪些 skill，以磁盘上的文件为准**，不以本手册、旧文档或你的记忆为准。

1. 查档位

```
nsrender status                       # 账号；本机已激活为完整版的，显示本机档位
ls skills/*/SKILL.md                  # 在内容包目录里跑：本机实际有的 skill
for s in editing-craft-pro script-design thumbnail-design shot-design remotion-effects; do
  test -f "skills/$s/SKILL.md" && echo "$s 有" || echo "$s 没有（完整版）"; done
```

- nsrender `nsrender status` 显示本机占的那个授权的档位（只装了基础版的设备显示「内容包：基础版 v…」和所在目录）；和目录不一致时**以目录为准**，用授权码激活过的设备再运行 `nsrender update` 和 `nsrender unpack --dir <内容包目录>` 。
- 内容包目录 `nsrender basic` 装的只看目录，就是基础版。5 个完整版 skill 都在才算完整版，缺的不写。结果写进 `AGENTS.md` 「This channel」的 `Skills` 一行和 `MEMORY/decisions.md` 。

2. 完整版 skill 完整版（第 1 步查到「有」才写）

Skill	做什么	依附的基础 skill
`editing-craft-pro`	开剪前的计划和结构、镜头语言与组接方法、进阶声音、发布后看数据	`editing-craft`
`shot-design`	分镜镜头：配方卡、光效限额、切点对齐底乐节拍	`faceless-broll`
`remotion-effects`	Remotion 效果组件和它们的时间曲线（只当参照，不参与出片）	`editing-craft`
`script-design`	切入点、开头多版候选、冲突结构、按顺序删到目标长度	`script-writing`
`thumbnail-design`	标题候选、缩略图 / 封面字与画面、小尺寸测试、发布后换图	`script-writing` §1c

完整版还多出整套手艺库 `skills/editing-craft/library/INDEX.md` 、收割流程 `HARVEST.md` 和 `skills/editing-craft/remotion-lab/` 。基础版只有 `library/STARTER.md` 列的 4 个 motion 场景；不要引用列表外的 `lib:<id>` 。

- 本机激活为完整版并 `nsrender unpack` 之后：重跑第 1 步，补上新出现的 skill，补问第 32–37 题。
- 让 agent 找到 skills：读 `AGENTS.md` 的 agent（例如 Codex）不用额外配置；有自己 skills 目录的 agent（例如 Claude Code）在内容包目录 `mkdir -p .claude && ln -s ../skills .claude/skills` 。`nsrender run <skill>` 尚未实现，现在由 agent 直接读 `skills/` 。

04

skills 启用表 · 基础版 skill 与 AGENTS.md 路由

3. 基础版 skill（两档都有）

Skill	做什么	启用条件	初始化时要不要改
`topic-research`	按真实热度选题，每个时段一份简报（`templates/topic-brief.md`）	有晨间任务就启用	不改；读 profile 的 `content.*`
`voice-reference`	写新稿前先读用户自己**已确认 / 已发布**的旧稿，提炼口吻卡和骨架（`pnpm broll reference list`）	有过一条用户确认的稿子以后	不改；第一条真片没有范本，跳过即可
`revision-learning`	把初稿和用户改定的版本逐处对比，沉淀成「问题清单」，写下一条前先读（`pnpm broll reference compare` / `lessons`）	用户改过一次稿以后	不改
`script-writing`	写稿、核实、口语化、配音前复审	必选	§2b 是通用的解说口播示例结构，按用户支柱改写这一节（写进「Owner rules」或改该节并记 FEEDBACK-LOG）
`faceless-broll`	分镜 `shots.json`、画面、声音、封面、渲染	必选	不改；风格从 profile 和 SOUL 读
`editing-craft`	节奏、强调、转场、卡点的基础规则；基础版带 4 个入门 motion 场景	建议启用	不改
`publish`	按时段发到启用的平台	启用了平台就启用	只保留用户启用的平台
`self-improve`	每条反馈沉淀进文档或代码	必选	不改

4. 写进 AGENTS.md：只写本机有的

```
- Skills（基础版，«日期» 核对）：
    topic-research ✓ · voice-reference ✓ · revision-learning ✓ · faceless-broll ✓
    · script-writing ✓（§2b 改写为 «支柱» 结构）· editing-craft ✓ · publish ✓（«平台»）· self-improve ✓
- 完整版另有分镜配方卡、Remotion 效果、标题封面和脚本工具箱、进阶剪辑（本机没有）
```

- 工作流表、启用表里**只出现第 1 步 `ls` 列出的 skill**。完整版的行文件不在就删掉，只留上面最后那一句话。
- 「Owner rules」里要有 AGENTS.md 一页的那条规则：引用的 skill 不在本机就提示可升级，不假装有。

完整的出片顺序在 `docs/PRODUCTION-SOP.md`，每一步都写了读写哪个文件，只用基础版 skill（见第 05 节）。

04

品牌视觉：定下图片和视频风格

内置三个预设只是例子： `clean` （彩色、中性，**默认**）、 `ink` （黑白水墨插画）、 `noir` （黑白电影剧照）。直接用其中一个：设好 `visual.default_style` 、写好 SOUL「Look」即可。多数频道应该写自己的：**先出 4–6 张测试图给用户挑，再写进配置**（ `skills/faceless-broll/SKILL.md` ）。

1. **写候选风格。**在 `config/profile.yaml` → `visual.custom_styles` 下加 1–2 个候选（四个字段都必填）：

```
custom_styles:
  «风格名»:
    image_prefix: "«Q16 的风格描述，英文效果更稳», {aspect}, «构图和质感»"
    image_suffix: "No text, no logos."
    video_suffix: "«Q19 镜头运动，如 Slow calm camera move, no cuts.»"
    avoid: [«Q17 禁用元素»]
```

2. **建测试文件夹** `videos/style-test/` ： `script.txt` 写 4–6 句和频道支柱相关的话；生成草稿字幕：

```
pnpm broll draft-srt --script videos/style-test/script.txt --out videos/style-
test/narration.srt
```

   再仿照 `examples/demo-short/shots.json` 写 `shots.json` ：顶层 `"style": "«风格名»"` 、宽高按 Q15，每句一个 `ai_still` 镜头（不同题材：人物剪影、物件、场景、抽象），可加一个 `ai_video` 镜头测镜头运动。

3. **免费预检：**

```
pnpm broll validate videos/style-test --list
pnpm broll make videos/style-test --dry-run      # 只打印每一条 Grok 调用（含风格前后缀），不
生成
```

4. **征得同意后生成** 先问 （消耗用户的 Grok 订阅额度）：

```
pnpm broll gen videos/style-test
pnpm broll stills videos/style-test              # 每个镜头一帧，输出到 out/style-
test/stills/
```

5. **给用户挑**，记下喜欢 / 不喜欢哪几张和原因；改描述再试，两三轮为限。
6. **定稿：**留下选中的风格（删掉落选的候选）， `visual.default_style` 设为它；SOUL「Look」写一句描述和禁用元素； `MEMORY/decisions.md` 记一条（选了哪几张、用户原话）。

没有 Grok 的用户：画面以代码动效和氛围镜头为主，在 SOUL「Look」写明。测试文件夹可留作以后换风格复用。

05

试片验收（不花钱）

用假的 Grok（ `engine/test/fake-grok.mjs` ）渲染内容包自带的 5 个镜头竖版示例，不消耗任何额度，验证整条渲染流水线。 `--grok-bin` 给的 `.mjs` 由引擎用 node 运行，不需要可执行权限，Windows 上也一样。试片只用引擎和基础版内容，**基础版和完整版跑法、通过标准完全一样**。

```
pnpm test
pnpm broll validate examples/demo-short --list
pnpm broll estimate examples/demo-short
pnpm broll make examples/demo-short --grok-bin "$PWD/engine/test/fake-grok.mjs" --out
/tmp/nsr-first
```

通过标准

- [ ] `pnpm test` 全部通过
- [ ] `make` 的输出里有单独一行 `done` （不是 `done with problems` ），退出码 0
- [ ] `/tmp/nsr-first/render/report.json` 里 `"check_ok": true` ， `"placeholders": []` ， `"grok_failed": []`
- [ ] `/tmp/nsr-first/render/` 下有 `broll_track.mp4` 和 `review.html`
- [ ] 你用 `open` （macOS）或浏览器打开 `review.html` 给用户看过

用户配置的检查

- [ ] `pnpm run profile` 第一行是 `profile: …/config/profile.yaml` ，没有 `EXAMPLE` 字样和 `voice_id` 警告
- [ ] `pnpm check-env` 没有 FAIL； `profile` 、 `agent docs` 两行是 ok
- [ ] 第 04 节的占位自查 `grep` 没有输出

> 试片用的是假 Grok，画面是测试图，不代表真实风格。真实风格以「品牌视觉」一页的测试图为准。第一条真片会花钱（配音、Grok 额度），按下一页走，每个审批点都要用户点头。

失败时：先看终端最后几行和 `report.json` ，再跑 `pnpm check-env` 按提示修；把原样报错告诉用户。

05

第一条真片：只用基础版 skill

第一条真片**不管本机是哪一档，都只用下表的基础版 skill**，先把全程跑通、让用户满意一条；完整版的进阶 skill（第 08 节）从第二条起再加。按 `docs/PRODUCTION-SOP.md` 的顺序走：

步	做什么	用哪个	审批 / 花钱
1	选题：按真实热度排候选，写简报	`skills/topic-research`	用户挑一个
1b	读用户自己的范本：`pnpm broll reference list`（下一页；还没有范本就跳过）	`skills/voice-reference`	—
1c	读问题清单：`pnpm broll reference lessons`（下一页；清单是空的就跳过）	`skills/revision-learning`	—
2	写稿：核实、中立、口语化；先定标题和封面字（该 skill 的 §1c，不是第 1c 步）	`skills/script-writing`	—
3	估长度（不花钱）：`pnpm broll estimate videos/<slug>/script.txt`	引擎	—
4	用户批稿	—	`approval.script_before_voice`
5	配音 + 逐字字幕：`pnpm broll voice`（或用户自己录）	`docs/MINIMAX_TTS.md`	先问
5b	第二语言字幕（设了 `subtitles.secondary_lang` 才做）：`pnpm broll subs-translate`，见下一页	`skills/script-writing` §5	先问 用 Grok 翻译时
6	分镜 `shots.json`：AI 镜头、motion 场景（入门示例或现写）、氛围镜头、封面	`skills/faceless-broll` + `skills/editing-craft`	—
7	预检：`pnpm broll make videos/<slug> --dry-run`，告诉用户有几个 Grok 镜头	引擎	—
8	渲染：`pnpm broll make videos/<slug> --voice <mp3> --subs <srt>`	引擎	先问 Grok 额度
9	自检：`check` 通过，逐镜看开头 / 中间 / 结尾，听音量，封面在安全区内能读	你	—
10	用户批片，再 `pnpm broll finalize …`	引擎	`approval.video_before_publish`
11	按时段发布	`skills/publish`	对外
12	反馈记下来；第二天看数据	`skills/self-improve`	—

05

第一条真片：范本、问题清单、第二语言字幕

第 1b、1c 步：写稿前先读（步号是本手册的，和 SOP 的编号不完全一样）

- **1b 读用户自己的范本**（`skills/voice-reference`）：`pnpm broll reference list`，读排在前面的几篇，在 `videos/<slug>/sources.md` 写口吻卡和骨架（带出处）。第一条真片还没有范本时跳过。
- **1c 读问题清单**（`skills/revision-learning`）：`pnpm broll reference lessons`，每条 `open` 都要对照；自查清单写进 `videos/<slug>/sources.md`。清单是空的就跳过（命令会说明为空）。
- 用户把初稿改成定稿交回来之后：`pnpm broll reference compare --draft <初稿> --final <定稿> --save`，再按 `skills/revision-learning/SKILL.md` 把问题记进清单。

第 5b 步：第二语言字幕

设了 `subtitles.secondary_lang`（问卷第 6 题选了双语字幕）才做：第 5 步配音生成中文 SRT 之后跑。

```
pnpm broll subs-translate --subs videos/<slug>/narration.srt
pnpm broll subs-translate --subs videos/<slug>/narration.srt --from lines.txt
pnpm broll make videos/<slug> --voice <mp3> --subs videos/<slug>/narration.srt
```

- **第一行命令的产出**：按原字幕逐条生成 `narration.en.srt`，条数、时间和原字幕完全一样。语言默认取 `subtitles.secondary_lang`，也可以用 `--lang` 指定。
- **默认用 Grok Build 翻译**（先问）：用的是用户自己的 Grok 登录，耗一点订阅额度，先问用户；Grok 没登录就做不了（登录只能由用户本人在交互终端完成，第 02 节）。
- **不调模型的做法（第二行）**：由你逐条翻译，写进一个文本文件，一行一条，行数和字幕条数一致，再用 `--from lines.txt` 生成。不花钱。
- **渲染前通读一遍**：按 `skills/script-writing/SKILL.md` §5 的规矩检查（自然的口语、不逐字硬译、每条读得完）。
- **渲染（第三行）**：英文轨放在中文 SRT 旁边（`<名字>.en.srt`），`pnpm broll make … --subs <名字>.srt` 会自动把它烧在中文下面，并写出 `subs.zh-en.srt`。别的语言用 `--subs-en <文件>` 指定。

内容包 1.0.8 起才有 `pnpm broll subs-translate`；更早的包先按附录「更新内容包」更新。

基础 skill 里遇到「有就读」的完整版引用（例如 `skills/script-design/SKILL.md`、`library/INDEX.md`），本机没有就跳过，照基础规则做，交付时告诉用户一次“这部分在完整版里”。第一条用户满意的成片，把文件夹写进 SOUL「Approved reference」。

06

定时任务：晨间灵感 + 每晚复盘

时间按 `schedule.timezone` 。两个任务都**只读、只写文档、只发消息给用户**：不配音、不生成画面、不发布。

任务	时间	做什么
晨间灵感 / 选题	第一个发布时段前 2–3 小时（Q29）	按 `skills/topic-research/SKILL.md`：扫 `content.sources` 和 `MEMORY/inspiration.md` 顶部的账号 / 关键词；候选和热度追加进 `MEMORY/inspiration.md`；每个时段写 `briefs/<日期>-<slug>.md`；把排序后的候选发给用户挑
每晚复盘	每天固定时间，默认 21:00（Q29）	按 `skills/self-improve/SKILL.md`：整理当天反馈进 `docs/FEEDBACK-LOG.md`；用用户给的平台数字写 `MEMORY/reviews/<日期>.md`；只把数据支持的结论写进 SOUL「What the numbers say」，样本小要注明

任务文本（存进 `MEMORY/prompts/`，各平台共用）

```
# MEMORY/prompts/morning.md
读 SOUL.md、MEMORY/README.md、AGENTS.md。按 skills/topic-research/SKILL.md 做今天的选题：
候选和热度追加到 MEMORY/inspiration.md；每个发布时段写一份 briefs/<日期>-<slug>.md；
最后把排序后的候选（每条一行：标题、热度数字、链接）发给我挑。不配音、不生成画面、不发布。

# MEMORY/prompts/nightly.md
读 SOUL.md、MEMORY/README.md、AGENTS.md。按 skills/self-improve/SKILL.md 做今晚复盘：
今天的反馈记进 docs/FEEDBACK-LOG.md；用我给的数据写 MEMORY/reviews/<日期>.md；
只把数据支持的结论写进 SOUL.md「What the numbers say」并注明样本量。没有数据就问我要。
```

06

定时任务：按用户的平台建

平台	怎么建

本机 cron	`crontab -e` 加两行；cron 用机器本地时区，与 `schedule.timezone` 不同时自己换算：
	`0 6  * * *  cd /path/to/never-sur-render && claude -p "$(cat MEMORY/prompts/morning.md)"`
	`0 21 * * *  cd /path/to/never-sur-render && codex exec "$(cat MEMORY/prompts/nightly.md)"`
	两行任选 `claude -p`（Claude Code 非交互模式）或 `codex exec`（Codex 非交互模式）。无人值守运行时，请用户在自己的 agent 设置里只放行读写本仓库和联网搜索，不放行任何发布命令。

Claude Code	用它的定时任务功能（如 `/schedule`），把任务文本贴进去，时区选 `schedule.timezone`；云端运行时只能用推送到用户私有远程仓库里的文件。
Codex	用上面的 cron + `codex exec`，或 Codex 自带的定时 / 自动化功能，贴同样的任务文本。
Grok Bot	建两个例行任务，任务内容贴上面两段文本，时区选 `schedule.timezone`，结果发到 Q29 指定的地方。

建好后各手动触发一次，确认能读到内容包目录、能写 `MEMORY/`、消息能送到用户手里；把任务名和时间记进 `TOOLS.md` 「This channel」和 `MEMORY/decisions.md`。

07

交付说明与客服

交付给用户的说明（5–10 行）

1. 新建 / 修改了哪些文件（ `config/profile.yaml` 、 `SOUL.md` 、 `MEMORY/` 、 `AGENTS.md` 、 `TOOLS.md` 、 `docs/FEEDBACK-LOG.md` ，以及 skills 改写的地方）。
2. 本机是哪一档（基础版 / 完整版）、启用了哪些 skill。
3. 哪些题没回答、用了什么默认值（基础版跳过的第 32–37 题一句话带过）。
4. `.env` 里还缺哪些 key，Grok 是否已登录（只说有没有）。
5. 试片结果： `check_ok` 、 `review.html` 的位置；品牌风格定了哪个。
6. 定时任务：在哪个平台、几点、第一次手动触发的结果。
7. 下一步：第一条真片——明早看选题，挑一个，我写稿给你批（配音前不花钱）。

基础版用户只在交付说明里提一次“分镜配方卡、Remotion 效果、标题封面和脚本工具箱、进阶剪辑在完整版里”，之后不主动推销；用户问起再按下一页回答。

客服

邮箱	**aceking@mail.grokbot.com**（安装、配置、登录、激活、档位、设备名额和解绑问题）
写信时	附上 `nsrender status` 或 `pnpm check-env` 的输出和原样报错；**不要**附上任何 key、token 或 `.env` 内容

> 内容包里其他文档对授权的描述（例如付款、设备名额）如果和下一页不一致，以下一页为准，并在交付说明里告诉用户。

07

授权：基础版与完整版

回答用户问题时以本页为准。 nsrender

两档都要登录 DonBluff 账号；档位按设备算。基础版登录后就能装；一台设备只有占上某个完整版码的名额（本机已激活）才是完整版。同一个账号下不同设备可以是不同档位。取包前先问用户要基础版还是完整版，流程见第 02 节「取包」。

基础版 · 免费，要登录，不用授权码

1. **先有账号**：用户自己在官网 `https://www.donbluff.com/login?next=/never-surrender` 用邮箱注册。你不能替用户注册。
2. **登录后装**：`nsrender login --email 邮箱` → `nsrender basic` 。不用授权码，不用付款。

这本手册本身是公开的，没有账号也能读；装内容包才需要登录。

完整版 · 一次买断，永久更新

完整版包含基础版的全部内容，另加第 08 节的进阶 skill。**在 DonBluff 官网的 Never Surrender 页购买**（`https://www.donbluff.com/never-surrender`），付款后授权码发到邮箱，也在官网「我的授权码」页。价格和条款以官网页面为准，你不报价格、不代付款。

占上完整版码名额的设备才拿完整版，每个码最多 2 台。授权码绑定购买时的 DonBluff 账号，要用同一个账号登录。`nsrender login` 成功后会自动检查账号里的授权：账号里已经有兑换过的完整版码、且这个码还有空名额时，**本机直接激活为完整版并装好内容包**，终端显示「本机已激活，内容包 …」，不用再跑 `nsrender activate` 、不用再输码；接着 `nsrender status` 确认档位，再 `nsrender unpack` 。**副作用：在别人的电脑上登录，也会占这个码的一个名额。**只有账号里还没有这个码（新买的码第一次用）时，才需要用户本人在交互终端运行 `nsrender activate` 输入完整版码。名额满了的码不会让新设备变成完整版。 `nsrender activate` 只用于完整版码；完整版码只由用户自己在终端输入，不写进任何文件、配置、记忆或日志。

退出登录、换电脑： `nsrender logout` 只清本机登录和解密材料，本机的设备名额保留；同一台电脑重新登录直接认回原来的名额，不占新名额、不扣解绑次数。**名额满了或要换电脑**：请用户登录官网，打开设备管理页（`https://www.donbluff.com/never-surrender/dashboard/devices`），在那里看这个码的 2 个名额占了几台、自助解绑一台。解绑计入解绑次数，每个完整版码终身 2 次；**解绑次数用完，再写信给客服 aceking@mail.grokbot.com**。步骤见第 02 节「设备名额」。

以后升级：见第 02 节「以后升级到完整版」。升级后 `nsrender status` 确认，再重读手册的完整版部分（第 03 节问卷四、第 04 节、第 08 节）。

用授权码激活的设备离线最多 10 天：设备票据 10 天有效，剩不到 3 天时联网自动复查；过期后要联网复查一次，才能再 `nsrender update` / `nsrender unpack` 。已经解到目录里的文件照常能用。码是否有效由服务器判定，码无效 / 已停用会直接报错。

07

授权：两档内容对照

完整版 ⊇ 基础版：基础版的每个文件完整版都有；完整版只是多出进阶 skill 和整套手艺库。

内容	基础版	完整版
选题、读自己的范本、从改稿里学、写稿（核实、口语化、配音前复审）、发布、自我改进：`topic-research` · `voice-reference` · `revision-learning` · `script-writing` · `publish` · `self-improve`	✓	✓
画面 / 声音 / 封面 / 渲染主流程 `faceless-broll`；引擎、默认镜头类型、字体	✓	✓
剪辑基础规则 `editing-craft`（节奏、强调、转场、卡点、声音）	✓	✓
手艺库 `skills/editing-craft/library/`（motion 场景、效果、字幕样式、镜头语言、组接手法）	4 个入门 motion 场景（`STARTER.md`）	整套（1.0.8 时共 55 条，其中 motion 场景 31 条；每晚收割会增加，以内容包里的 `library/INDEX.md` 为准）+ 收割流程
模板（SOUL、剪辑计划、选题简报）、`AGENTS.md` / `TOOLS.md` / `INIT`	✓	✓（路由表多出完整版的行）
封面规则（`script-writing` §1c）	✓	✓
进阶剪辑 `editing-craft-pro`	—	✓
分镜镜头 `shot-design`（配方卡、光效限额、节拍对齐）	—	✓
Remotion 效果 `remotion-effects` + `remotion-lab`	—	✓
标题 / 缩略图 `thumbnail-design`、脚本设计 `script-design`	—	✓
账号 · 授权码 · 价格	要登录 DonBluff 账号 · 不用授权码 · 免费	要登录 DonBluff 账号 · 完整版码 · 一次买断（官网购买）
设备 · 更新	登录后 `nsrender basic` · 永久更新	占上完整版码名额的设备 · 每个码最多 2 台、终身解绑 2 次 · 永久更新

档位按设备算：只有占上完整版码名额的设备才是完整版（登录时自动激活，或在本机用 `nsrender activate` 输过码）；名额满了的码不会让新设备变成完整版。本机到底是哪一档，以第 04 节「先查档位」的结果为准；表里写“完整版”的内容本机没有时，照基础版做，不要假装有。

08

进阶 完整版

这一章只对完整版有效。开始前按第 04 节「先查档位」确认文件在本机；不在就整章跳过，用基础版做法，不要照本章描述去“模仿”这些 skill。

用法：第一条真片之后，从第二条起按需要加。每个 skill 都只是工具箱，基础 skill 的流程和硬规则照旧（审批、花钱先问、产出目录）。每次用了哪个方法、为什么合适，在 `videos/<slug>/sources.md` 顶部或剪辑计划里写一行。

方向	Skill	什么时候读	产出落在哪
分镜镜头	`skills/shot-design/SKILL.md`	写 `shots.json` 前选每一镜怎么做：读配方卡（意图、参数、已知坑），控制光效和整画面冲击的次数，切点和落定帧对齐底乐节拍，渲完回测误差	`shots.json`；新配方卡
Remotion 效果	`skills/remotion-effects/SKILL.md` `skills/editing-craft/remotion-lab/`	要精确的强调 / 入场 / 转场时间时：先看带标签的样片，抄时间曲线和帧数，用引擎的 kit 在 motion 场景里重画（不要 import 组件）	`scenes/*.js`；新效果进库
标题与缩略图	`skills/thumbnail-design/SKILL.md`	写标题候选、封面字和画面时：一句承诺拆给标题和封面，缩小到手机尺寸还读得清，发布后数据差就换备用那套	`shots.json` 的 `cover`；简报
脚本设计	`skills/script-design/SKILL.md`	找切入点、写开头的几个版本、按冲突排结构、超长时按顺序删；在 `script-writing` 的每一步里挑方法用	`script.txt`、`sources.md`
进阶剪辑	`skills/editing-craft-pro/SKILL.md`	开剪前定计划和结构、选镜头语言和组接方法、Grok 画面的光和看帧检查、进阶声音、发布后看数据复盘	`templates/edit-plan.template.md` 生成的计划；SOUL
整套手艺库	`skills/editing-craft/library/INDEX.md` `HARVEST.md`	写 motion 场景前先查库，合适就 `"scene": "lib:<id>"` 复用；新场景好用就当天收回库	`pnpm broll harvest` → `library-add`

本机激活为完整版后第一次用：重跑第 04 节「先查档位」，把新出现的 skill 补进 `AGENTS.md` 工作流表和启用表，补问第 32–37 题，答案按题目右侧写进 SOUL / `MEMORY/decisions.md`。这些 skill 的规则和频道的「Owner rules」冲突时，照「Owner rules」做。

附

排错：nsrender

现象	处理
`nsrender: command not found`	重新运行 `npm i -g https://acfdywgmrfzptuepufcj.supabase.co/functions/v1/ns-public/cli/nsrender.tgz`，重开终端；报“内置配置里这一项是空的”也是重装这一条
`nsrender basic` 提示需要先登录	照提示做：没有账号先在官网注册，再 `nsrender login --email 邮箱`，登录成功后重跑 `nsrender basic`
`nsrender login` 报“没法询问邮箱”	没有交互终端时必须带邮箱：`nsrender login --email 用户的邮箱`
`nsrender login` 提示邮箱没有对应的账号	请用户先到 `https://www.donbluff.com/login?next=/never-surrender` 用这个邮箱注册，再重跑登录；你不能替用户注册
`nsrender login` 等不到回调	邮件里的链接要在这台电脑的浏览器里打开；超时重跑
`nsrender activate` 报没有交互终端	本机还没激活、又没有交互终端时才报这一条（退出码 2）。请用户在自己的终端里运行并输入完整版码；它只用于完整版码，基础版不用跑；账号里已有码且有空名额的，登录时已自动激活，也不用跑
`nsrender activate` 显示「本机已激活：本机是完整版（edition …），票据到 …。不用再输入授权码。」	正常，不是报错（退出码 0）：本机已有有效授权，没带 `--code` 时不问码、不联系服务器，有没有交互终端都一样。本机是基础版授权时：没有交互终端也只显示这一段（写的是基础版，带升级完整版的做法）；有交互终端则仍提示输入授权码，这是升级完整版的入口。写在 `--code` 后面的码不管本机有没有激活都发给服务器判定
登录 / `nsrender update` 报授权都已绑满（`device_limit`），或 `nsrender activate` 报这个码的名额已满（`code_device_limit`）	本机不变。每个完整版码最多 2 台：请用户登录官网，打开设备管理页 `https://www.donbluff.com/never-surrender/dashboard/devices` 解绑一台（计入解绑次数，每个码终身 2 次），再重新登录；解绑次数用完：写信给客服 aceking@mail.grokbot.com。提示里写的就是设备管理页及其网址
`nsrender logout` 之后名额还占着	正常：退出登录保留本机名额，同一台电脑重新登录认回原名额。要让给别的电脑，按上一行解绑。已发布的 nsrender 1.0.3 退出登录后提示的“请在账号页解绑”，说的就是设备管理页
在别人的电脑上登录，占掉了一个名额	登录会自动激活本机。不想占：登录时带 `--no-activate`；已经占了的要解绑才能拿回（计次数）
`nsrender unpack` / `basic` 报目录已有别的文件	换一个空目录，或用上次装过的那个目录

登录、取包的正常流程见第 02 节「取包」；授权口径见第 07 节。

附

排错：内容包目录

现象	处理
`pnpm: command not found` 或版本不对	`corepack enable && corepack prepare pnpm@9.15.4 --activate`，重开终端
`check-env：dependencies FAIL`	`pnpm install --frozen-lockfile`，或整套 `pnpm bootstrap`
`chromium FAIL`	`pnpm --filter @nsr/engine exec playwright install chromium`；或在 `.env` 用 `BROLL_CHROME` 指向已有的 Chrome
`make` 报 `grok (Grok Build CLI) not found`	用户本人安装，并在自己的交互终端 `grok` → `/login`（你不能代登录）；装在别处就在 `.env` 设 `GROK_BIN`；暂时用 `--skip-gen`（AI 镜头变占位，报告会标 `done with problems`）
`.env` 里的 `GROK_BIN` / `FFMPEG_PATH` / `FFPROBE_PATH` / `BROLL_CHROME` 不生效	内容包是 1.0.7 或更早，只认 shell 里 `export`：按附录「更新内容包」更新到 1.0.8 以上，或先 `export`。两边都有时以环境变量为准
`check-env` 的 `grok` 行写 `login not checked`	不是报错：`GROK_BIN` 指向别处的程序，没法替它检查登录；第一次生成 Grok 镜头时就能看出来。没登录就请用户本人在交互终端 `/login`
Grok 镜头失败	终端的 `✗` 行写了原因（`cannot start … ENOENT / EACCES` 是 `--grok-bin` / `GROK_BIN` 的路径或权限不对），完整输出在 `out/<slug>/assets/logs/`；重跑 `make` 只重试失败的；单个重做 `pnpm broll gen videos/<slug> --only S017 --force` 先问
试片 `✗ … no streaming-json events`，`pnpm test` 有 3 个 broll 测试失败	内容包是 1.0.1：按附录「更新内容包」更新；或先 `chmod +x engine/test/fake-grok.mjs` 再跑
`unknown style "…"`	`shots.json` 的 `style` 或 `visual.default_style` 不是 `clean` / `ink` / `noir`，也不在 `visual.custom_styles` 里
`pnpm run profile` 显示 EXAMPLE	还没生成 `config/profile.yaml`，或 `voice_id` 还是示例值
key 显示 not set	确认写在内容包目录的 `.env`、变量名拼写一致；由用户自己检查内容
skill / `lib:<id>` 找不到	多半是完整版的内容、本机是基础版：改用基础版做法，告诉用户一次（账号里有完整版码但本机没占上名额——名额已满，或登录时带了 `--no-activate`——本机仍是基础版）；本机已用完整版码激活的跑 `nsrender update`，再 `nsrender unpack --dir <内容包目录>`，然后重跑第 04 节「先查档位」
估算时长退出码 3	`pnpm broll estimate` 判定落在禁区档（默认 3–8 分钟）：删稿或扩成长片，或按频道改 `BAND_EDGES`

附

更新内容包

- nsrender **用完整版码激活的**：`nsrender update` 更新到本档位的最新版，再 `nsrender unpack --dir <内容包目录>` 写进目录。你和用户改过的包内文件（例如 `AGENTS.md` / `TOOLS.md` 里的「This channel」「Owner rules」）保留原样并列出来，不覆盖；`SOUL.md`、`MEMORY/`、`config/profile.yaml` 这些用户自己的文件不会被碰。新版有问题时 `nsrender rollback` 回到上一版并固定，再 `nsrender unpack`；再次 `nsrender update` 解除固定。`nsrender status` 看当前 / 上一版 / 最新版。
- nsrender **`nsrender basic` 装的**：对同一个目录再跑一次 `nsrender basic --dir <内容包目录>`（要先登录）。它会**覆盖包内文件**：先把 `AGENTS.md` / `TOOLS.md` / `docs/FEEDBACK-LOG.md` 里用户写的部分另存，更新后再填回去；用户自己的文件不会被碰。
- 内容包目录 更新后跑 `pnpm bootstrap` 和第 05 节的试片。
- 写这一版手册时的内容包是 **1.0.8**：完整版 646 个文件，基础版 137 个文件。本机装的是哪一版、哪一档，以 `nsrender status` 显示的为准；手册里写“内容包 1.0.8 起”的功能，本机版本更早就先更新。

— never sur-render —