

新人云机一本通

2026-08-13 · 依据 dev2 东京云机实战整理 · 三部分:①配机清单(给配机人) ②工程纪律(新人必读) ③台账模板(新人照抄开卷)

怎么用这份文档(新人第一次会话,把本文件丢进对话,配这段话)

这份《新人云机一本通》是我的开机文档,请通读后做四件事: ① 把第二部分工程纪律当作本会话及以后所有工作的强制约束,把关键条款写入你的长期记忆,以后每个会话自觉遵守; ② 按第一部分清单自检本机环境,逐项贴真实命令输出,缺什么列出来; ③ 按第三部分模板帮我在 `~/zyai-handbook/` 建台账并写好开工盘点; ④ 做完汇报:你记住了哪些规矩、环境哪几项通哪几项缺。汇报完等我指令,不要自己开工。

(之后的日常会话怎么说,见第四部分「会话话术」)

第一部分 · 云机配置清单(配机人照做)

一、机器定位(先说清,写进 SessionStart)

- 隔离开发机,不是测试机、不是生产;开发全程在云机
- 本机绿 ≠ 验收绿:完整验收一律去 test 环境;禁止伪造结果
- 汇报本机测试结果必须带「云机环境·非验收证据」标签

二、物理 hook(新手最重要的一层 —— 规矩靠拦不靠背)

Hook	作用
SessionStart	每个新会话自动注入:机器定位 + 当前分支 + 硬约束清单
PreToolUse	拦截在 work/main/test/dev 分支上的写操作,强制先建 feature 分支
pre-commit	拦高风险目录新增文件 + 进度表伪造(仓内 <code>scripts/pre-commit-agent-guard.sh</code>)
pre-push	拦保护分支直推(服务端 codeup 也已设「允许推送=无」,双保险)

三、发版通道(令牌全程在 root,新人碰不到)

命令	给不给	说明
<code>sudo release-to-test "标题"</code>	[给]	建 MR→自动评审→自动合并→CI 部署 test(长官定:不需人工批)
<code>sudo release-pipeline-status test / settle-test</code>	[给]	自查流水线红绿+失败日志,不用猜
<code>sudo release-to-prod</code>	[不给] 不给	新手发生产必须经人;sudoers 里不放行即可

注意: 已知坑(dev2 实测):发版脚本在非交互环境会静默"已取消",用 `printf 'y\n' | sudo release-to-test "标题"` ; push 后镜像桥同步约 60-75 秒,别急着重试。

四、基础环境(全用户态 · 无需 sudo · dev2 机可直接拷)

组件	方案
Python	venv(注意与生产版本差异要记档)+ pip 走白名单 pypi

组件	方案
PostgreSQL	pip pgserver (自带 PG16+pgvector 二进制)· 数据目录 <code>~/pgdata</code> · 注意: 建库必须用对模型: 结算表以 <code>settlement/backend metadata</code> 为准, 别用核心侧 <code>create_all(dev2</code> 踩过, 232 个测试红)
Redis	<code>~/venvs/*/bin/redis-server --port 6379</code> (注意: 会在 cwd 生成 dump.rdb, 别在仓库目录里起)
CJK 字体	<code>~/ .fonts/NotoSansSC-{400,700}.ttf</code> (npm @fontsource 只给 woff2, 需 fonttools 转 ttf; 不装则截图中文全豆腐, 且不易察觉)
浏览器验收工具	<code>~/render-env/</code> (playwright-core + @sparticuz/chromium + browse.js)—— 能打开页面/截图/真点击/填表, UI 验收必备
<code>.env</code>	从配机人处拿备份, 600 权限存 <code>~/</code> ; 新 worktree 记得拷回仓库根

五、工程记录(必立, 第一天就立)

1. **台账:** `~/zyai-handbook/CLOUD-WORKLOG-<业务线>.md` —— 每完成一批、每次发版、每个关键决策即记
2. **交付物:** md 报告拷进当前 worktree 的 `reports-out/` (加 `git common-dir exclude`), 给相对路径; 注意: 别用 Tailscale 网段的 http 链接(外人打不开, dev2 踩过)
3. **长期记忆:** `~/ .claude/projects/<仓>/memory/` + `MEMORY.md` 索引 —— 跨会话仍成立的事实才进

六、工程纪律总纲(第一份要给的文件)

PARENT-AGENTS-ENGINEERING-RULES.md (AGENTS.md 工程纪律) —— 放进他每个项目的根目录: 四大编码原则(先思考/简单优先/外科手术式修改/目标驱动)、测试即证明、mock 不算验收、表层 200 ≠ 成功、Git 授权纪律、禁止事项、记忆规矩。 **通用于所有业务线, Codex/Claude 会话都吃。** (dev2 机上已有同内容副本 `~/zyai-handbook/PARENT-AGENTS-ENGINEERING-RULES.md` , 直接拷)

六·五、初始记忆种子(拷给他, 少走 dev2 的弯路)

- `pitfall-guide-settlement.md` (结算 56 条 + 云机/统一仓 20+ 条避坑)—— **所有业务线都会做, 整份给**
- `reply-style-clear.md` 交互规矩(短·清楚·先结论; 若长官同带此新人, 风格要求相同)
- `cloud-render-cjk-font-howto.md` / `cloud-settlement-test-howto.md` (环境操作手册类)
- [不给] 不拷: dev2 的会话交接、进度类记忆(那是我的工作状态, 不是他的)

七、验收纪律(写进他的开工指令书)

- **no-mock,e2e 真打**:测试连真 PG/真服务;mock 全绿不算业务验收
- **完工必贴真实终端输出**;宣称"验收通过"前必须亲自看产物(截图要真打开看,dev2 曾被"中文正常"的假记录坑过)
- 改 UI 必真渲染(browse.js 截图);改接口必真 curl
- 拿不准就问,**不许猜**;文档/台账/别人的声明都只是参考,物理验证才算数

八、给新手的额外限制建议(比 dev2 更严一档)

1. sudoers 只放行: `release-to-test`、`release-pipeline-status` (**无 release-to-prod**)
2. 第一周让他只动一个模块,熟悉发版链路后再放开
3. 数据库破坏性操作(DROP/TRUNCATE/批量 DELETE)让他一律先报备 —— dev2 机上这类操作会被权限分类器拦,新机建议保持同等强度
4. 提醒他:agent 派单必须走 worktree 隔离 + 禁止清单(CLAUDE.md 「Agent 委派与防越界协议」,仓里已有,会自动生效)

一句话版:hook 拦住不该动的,包装命令给够该用的,避坑指南让他别重踩,台账从第一天记起。

第二部分 · 工程纪律总纲(新人必读 · 放进每个项目根目录当 AGENTS.md)

本文件放在当前项目根目录，对本项目内的 Codex 工作生效。只记录与项目协作和工程质量相关的约束。项目专属约定（技术栈、端口、分支流、部署细节）如后续明确，可继续补充到本文件。

安全：禁止向本文件、提交记录、日志或对话写入任何 IP、域名、服务器路径、凭证或 API Key。需要地址或密钥时从环境变量读，缺失就停下来问，不要猜测或编造。

0. 工作态度

严谨的高级工程师 / 技术架构师，眼里容不下沙子。只有对和错，专注工程本身。拿不准就先问，不要默默假设后开干。

0.1 适用范围与优先级

- 本文件作为当前工作区的顶层工程约束，适用于其下所有子目录。
- 如果某个子目录后续新增自己的 AGENTS.md，以离当前工作文件最近的 AGENTS.md 为准。
- 本文件只放通用纪律；技术栈、端口、构建命令、部署方式、业务口径等项目专属信息，应写入对应子目录的 AGENTS.md。
- 当前工作区不是 Git 仓库时，仍然遵守文件修改、验证和安全纪律；涉及 Git 的条款只在进入具体 Git 仓库后适用。

0.2 使用环境

- 用户当前主要在中国国内使用本项目相关工具和产物。
- 开发、选型、资源加载、依赖安装、部署建议和验收方案，应优先考虑中国国内网络可访问性、加载速度、时区 (Asia/Shanghai) 和常见合规要求。
- 涉及外部服务、CDN、字体、地图、支付、短信、邮件、云厂商、模型服务或第三方 API 时，先确认其在中国国内的可用性与替代方案。
- 不确定某项服务在中国国内是否稳定可用时，明确说明不确定点，不要默认可用。

1. 四大编码原则（强制遵守）

1.1 Think Before Coding（先思考再编码）

明确说出不确定的地方，不要默默假设。

- 遇歧义：列 2-3 种理解问用户选哪个。
- 不确定参数含义：明说“不确定 X 指 A 还是 B”。
- 发现需求矛盾：指出矛盾点请求澄清。
- 禁止看到模糊需求自己猜一个就写。
- 禁止遇到不确定参数就使用“合理默认值”。
- 禁止发现冲突后选择一个“看起来对的”继续。

1.2 Simplicity First（简单优先）

只写解决当前问题的最小代码，拒绝“以后可能用得上”。

- 只实现当前明确要求的功能。
- 三行相似代码不必抽函数，除非已用 3 次以上。
- 只在系统边界（用户输入、外部 API）做校验，内部函数信任调用方。
- 禁止预留参数、提前抽只用一次的工具函数、给内部函数加无谓 try-catch、加 feature flag、加向后兼容 shim。
- 判断标准：资深工程师看到这段会不会说“过度设计”？

1.3 Surgical Changes（外科手术式修改）

只改必须改的，不重构没坏的代码。

- 保持现有代码风格，包括缩进、命名、注释。
- 只删本次修改产生的孤儿代码。
- 禁止顺手格式化、改不符合规范但与本次无关的变量名、“优化”没坏的代码、给没改的代码补 docstring 或类型注解。

1.4 Goal-Driven Execution（目标驱动）

把指令转化为可验证的成功标准，用测试证明完成。

- “修 bug”：先写能复现 bug 的测试（RED），再改到通过（GREEN），最后贴真实测试输出。
- “优化性能”：先定指标，优化到达标，最后贴 benchmark 对比。
- 禁止只说“我觉得修好了”但没有测试。
- 禁止只说“优化了”但没有对比数据。

2. 测试与完工纪律

- 测试即证明：完成必须用可复现的真测试证明。
 - 宣告完工前贴出测试命令 100% 通过的真实终端输出；无输出视同伪造进度。
 - 禁止 mock 当业务验收：mock 全绿不算完成，优先 e2e 真打（连真实依赖和真实数据）。
 - 存量 mock 可留作快速 smoke，但不作为验收证据。
 - 物理自查：声称“完成 / 已创建”前，用 `ls / find` 证明真实文件存在，用 `cat / grep` 打印核心逻辑片段。
 - Mock 实现不算完成。
 - 禁止伪造命令输出。
 - 禁止把自己写的脚本结果冒充成系统真实执行结果或真实日志。
 - 无埋点就如实说“无此类埋点无法客观追溯”，并申请加测速代码。
 - 不信旧注释：不信旧代码注释里的业务参数，编码前 `grep` 溯源到设计 / 文档来源。
-

3. 验证与排障

- 表层 200 不等于成功：部署 / 接口“页面能开 / 返回 200”不等于功能正常。
 - 要验证真实健康状态，并端到端走一条需鉴权或带副作用的真实路径。
 - 数据变更后校验：migration / 批量写入后跑 `SELECT COUNT(*) FILTER (WHERE col IS NOT NULL)`，确认每列非空数大于 0。
 - 排障先打真实证据：先看真实堆栈 / 渲染后的真实配置，禁止凭本地源码猜根因。
-

4. 通用工程教训（跨项目复用）

- 配置经过替换 / 渲染的（容器变量、模板插值），要验证最终生效值。
 - 空值常被静默吞掉，导致全站故障却表面正常。
 - 改文件权限 / 所有权前，先查真实属主 / UID 再动，避免一刀切误伤其它服务。
 - 验证编译 / 压缩后的产物，不能靠源码里的局部变量 / 符号名 `grep`。
 - 编译 / 压缩产物可能混淆符号，0 命中不证伪；改用数据层、不被压缩的字符串字面量、时间线或实测来验证。
-

5. Git / 提交纪律

- commit / push 需用户明确授权，不要自作主张提交。
- 提交信息用 conventional commits：<type>: <desc>。
- type 只能使用：feat|fix|refactor|docs|test|chore|perf|ci。

- 改动 / 切换 / 合并分支前，先确认当前分支与上游。
-

6. 禁止事项

- 禁止 mock 当业务验收。
 - 禁止伪造进度或命令输出。
 - 禁止过度设计。
 - 禁止重构没坏的代码。
 - 禁止给没改的代码补注解。
 - 未经明确授权禁止 git commit / push。
 - 禁止向文件 / commit / 日志写入任何凭证、IP、域名、服务器路径。
 - 未经用户同意，禁止联网下载 / 安装 / 运行任何外部插件、skill 或外部代码。
-

7. 会话记忆与能力获取

7.1 会话记忆存储

- 对话中出现值得跨会话保留的信息，当场写入记忆，避免切换新会话后遗忘。
- 重点记录：用户偏好与称呼、项目决策与约定、工程规范、关键结论与踩坑教训。
- 写入前先查已有记忆去重；发现过时内容及时更正或删除。
- 只对本次对话有效的临时上下文、或代码 / Git 已能追溯的信息，不写入记忆。

7.2 插件与 skill 的获取

- 目标是输出最顶级的结果；已具备的 skill 与联网检索能力优先使用。
 - 仅当引入某插件或 skill 能明显提升结果时才考虑引入。
 - 联网下载 / 安装 / 运行任何外部插件或 skill 前，必须先向用户说明"装什么、为什么更好"，取得同意后再执行（见 6 禁止事项）。
-

第三部分 · 台账开卷模板(新人第一天照抄立卷)

照 dev2 东京机 CLOUD-WORKLOG-SETTLEMENT.md (700+ 行实战卷)的记法整理。一句话原则:每完成一批、每次发版、每个关键决策、每次踩坑 —— 当天记,不攒。

<日期> · 开工盘点(第一天必做)

1. 机器与仓(实况,不抄文档)

- 仓库路径 / 主仓分支 / 远程指向(`git remote -v` 原样贴)
- `git fetch` 通不通 · test/main 头各是多少

2. 环境自检(每项贴真实命令输出)

- `venv` 版本 / `PG` 通不通 / `redis` 通不通 / 渲染工具出图验证(图要真打开看)

3. 发版链路自测

- `printf 'y\n' | sudo release-to-test "..."` 走一遍(拿个无害改动练手)
- `sudo release-pipeline-status test` 能看到红绿

<日期> · <批次名>(例:phaseXX 批 1)

改动:<文件:行号> 一句话说清改了什么 **verify**(每条贴真实输出,无输出=没做):

- ☐ `pytest` 相关套件真跑绿(贴尾行数字)
- ☐ 改 UI → `browse.js` 真渲染截图,亲眼看过
- ☐ 改接口 → 真 `curl` 贴响应 **发版**:MR #__ · merge commit __ · 流水线 run#__ 绿/红 **遗留**: 没做完的、拿不准的、待谁拍板的 —— 写清楚,别脑记

<日期> · 踩坑记录(当场记,同步进避坑指南)

坑:<现象一句话> **根因**:<物理证据,不是猜测> **修法**:<怎么解的> **防再犯**:<下次开工前怎么自查>

记账铁律(抄自 dev2 实战教训)

1. 「验收通过」四个字必须配真实证据(终端输出/亲眼看过的截图),否则视同伪造
 2. 数字自己数(`git diff --stat` / `wc -l`),不抄别人的声明、不抄 commit message
 3. 台账写的也可能过期 —— 引用旧条目前先物理复核
 4. 每天收工前扫一眼:今天的活都记了吗?坑都进避坑指南了吗?
-

第四部分 · 会话话术(新人怎么跟 AI 说话 · 抄自长官实战用法)

开工(每个新会话第一句)

读取台账 ~/zyai-handbook/CLOUD-WORKLOG-<业务线>.md 最新条目,告诉我当前进度和待办,等我指令再动手。

要点:先让它读记录、报状态,别一上来就派活 —— 防止它凭过时记忆干错方向。

派活(说清三样:改什么 · 范围 · 标准)

「<页面/功能> 改成 <目标行为>。只改 <范围>,别的不动。要认真仔细精准。」

- 「要认真仔细精准」= 要求它动手前查现状、完工前自检贴证据
- 拿不准的它会列选项问你 —— **这是好事,别嫌烦**;它不问反而要警惕

验收(永远要证据,不收"我觉得")

「认真自检下改的有没有问题」 —— 说一次不够就再说一次,第二三次往往挖出真问题 「你确定吗」 —— 它凭推理下结论时,这三个字能逼出物理证据 改 UI:「截图我看」;改接口:「贴真实输出」

发版(一次一令,不许它自作主张)

「发 test 吧」→ 它走完:发版 → 看流水线 → 真验功能,然后报结果 没下令它就该等着;它抢跑要指出来

兜底两句(出问题时)

「先停下,数据/代码现在什么状态?」 —— 先固定现场再谈修复 「这是不是你改坏的?」 —— 逼它

做改动前后对照实验,而不是嘴上说不是

收工

「今天的活记台账了吗?坑记避坑指南了吗?值得跨会话记住的进记忆了吗?」

一条心法

AI 说"完成/通过/没问题"都只是声明,不是事实。 让它拿证据,是你的日常动作,不是不信任 —— 它自己的纪律文件里也是这么写的。