

黑客松冠军的配置，我也来用用

你现在的 Claude Code 就是用 `~/.claude/` 作为“全局配置中心”（你截图里那些 `skills / plans / projects / session-env` 等文件夹都在这里），所以把 `everything-claude-code` 用进来，本质就是：把它提供的三类东西（Plugin / Rules / Skills&Hooks）放进你这套 `~/.claude` 体系里，并让 Claude 识别到。

下面给你一个非常宏观的落地路径（不纠结具体命令细节）：

果然，linux系统中~代表的是这个家home目录

显示思路

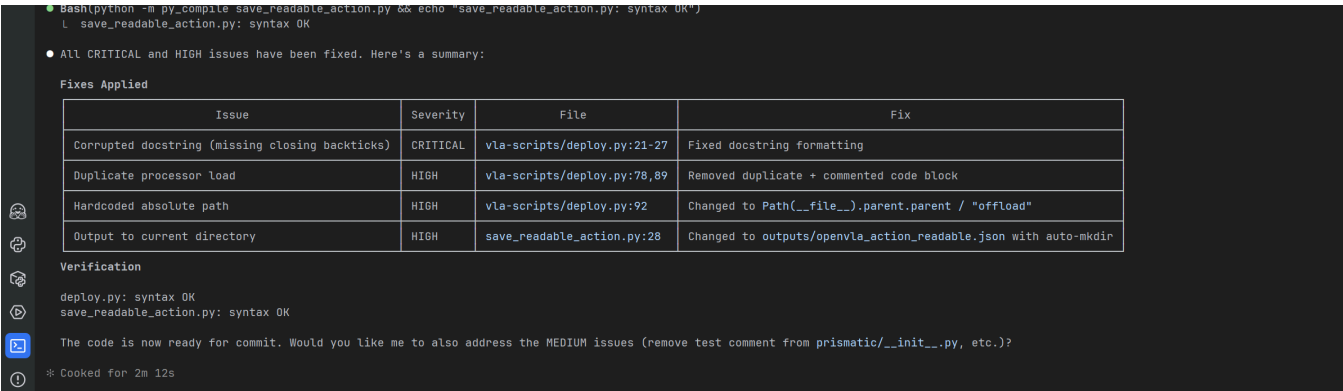
这是一个非常重要的概念澄清。在 Linux 中，符号 `~`（波浪号）代表的是“当前用户的家目录（Home Directory）”，而不是整个系统的“根目录（Root Directory, `/`）”。

根据你提供的截图，你的 Linux 环境中：

1. 你的 `~`（家目录）在哪里？

在 Windows 的文件资源管理器地址栏中，它对应的路径是：

Linux > Ubuntu-22.04 > home > ubuntu



真是夸张，对整个项目的问题进行分层列出，全部解决，牛逼

阶段	目标	你在 Claude Code 里做什么（体现黑客松项目）	产物（必须落盘）
0. 进入项目上下文	让 Claude 读到你的 repo	在项目根目录启动 <code>cclaude</code> ；确保 <code>/</code> 命令可用	<code>PROJECT_CONTEXT_OK</code> （心智确认即可）
1. 任务定义	把“想法”变成可执行实验	<code>/plan</code> ：定义歧义类型、候选生成、澄清策略、评估指标	<code>plan.md</code>
2. 质量门槛	把“能跑”变成“可复现”	<code>/coding-standards</code> ：补齐目录/命名/日志规范	<code>repro_checklist.md</code>
3. 实验骨架生成	把重复步骤模板化	用对应 command/skill：生成 <code>run.sh + config.yaml + log schema</code>	<code>scripts/ configs/ logs_schema.json</code>
4. 实现与修改	最小改动完成闭环	让 Claude 按文件路径改；改完 <code>/code-review</code>	<code>diff.patch</code> （或 <code>commit</code> ）
5. 运行与修复	跑通一次完整闭环	失败就 <code>/build-fix</code> ；成功就保存命令与环境信息	<code>run_log.txt</code> + <code>env.txt</code>
6. 评估与对比	从一次跑通变成可对比实验	固定指标表（成功率/澄清轮数/误澄清率/执行时间）	<code>results.csv</code> + <code>report.md</code>
7. 消融与迭代	把“改一点”变成系统性证据	每次迭代重复： <code>plan → implement → review → run → report</code>	<code>ablation_table.md</code>
8. 技能沉淀	把你常做的事变成“一键”	用 skill/command 固化：生成“消歧实验模板/数据条目模板”	<code>skills/your_lab_skill/</code>
9. 论文资产输出	让实验直接变论文素材	每周生成“方法段落+实验设置+结果表+失败案例” <div>↓</div>	<code>paper_notes/week_x.md</code>

一整套科研开发流程，在日常的实验中要多多有意的使用

给gpt这个记忆功能记住，给这个claude项目功能填上

```
GitHub Project Configurat × + | v

| refactor-cleaner | 重构清理 |
| security-reviewer | 安全审查 |
| tdd-guide         | TDD 指导 |

Commands (21个) - ~/.claude/commands/

/build-fix, /checkpoint, /code-review, /e2e, /eval, /evolve, /go-build, /go-review, /go-test, /instinct-expo
/instinct-import, /instinct-status, /learn, /orchestrate, /plan, /python-review, /refactor-clean, /sessions,
/skill-create, /tdd, /test-coverage, /update-codemaps, /update-docs, /verify

Skills (28个) - ~/.claude/skills/

后端、前端、Django、Spring Boot、Go、Python、PostgreSQL、ClickHouse、TDD、安全审查等模式

Rules (8个) - ~/.claude/rules/

agents.md, coding-style.md, git-workflow.md, hooks.md, patterns.md, performance.md, security.md, testing.md

Hooks - ~/.claude/hooks/hooks.json

- PreToolUse: 阻止 dev server 在 tmux 外运行、tmux 提醒、git push 审查、阻止无用 .md 文件创建
- PostToolUse: PR 创建日志、Prettier 格式化、TypeScript 检查、console.log 警告
- SessionStart/End: 会话状态持久化
- Stop: console.log 审计

配置来源与 everything-claude-code 项目一致，已完整配置。
```

claude强大的skill，一定要多加挖掘。