🪝 Gientech Hooks 配置-培训指南

> 手把手教你使用 `/hooks-creator-gientech` 技能，为项目配置智能 Hook 通知系统

📖 背景介绍

GienCoder Hooks 是GienCoder IDE 插件提供的一套事件驱动机制，允许你在 Agent 工作的各个生命周期节点（如工具执行前、会话启动、任务完成时）注册自定义脚本，实现**权限检查、日志记录、上下文注入、通知提醒**等功能。

本指南将以一个实际案例——**"任务完成时发送系统通知"**——带你完整走一遍从零配置 Hooks 的流程。

🎯 实战目标

需求	说明
触发时机	每轮任务完成时
提醒方式	桌面通知弹窗 + 系统提示音
支持系统	macOS（也兼容 Linux）
实现效果	任务完成自动弹出通知，并播放提示音

🔲 第 1 步：确认 Skill 已上传并下发到工作空间

在使用 `/hooks-creator-gientech` 技能之前，需要确保该 Skill 已经在 Giencoder 管理后台配置好，并已下发到你的工作空间。

操作指引

1. **打开 Giencoder 管理后台 — 浏览器访问 Giencoder 管理页面**
2. **进入 Skill 管理** — 找到 Skill 管理模块，查看已上传的 Skill 列表
3. **确认技能存在** — 检查列表中是否有 `hooks-creator-gientech` 这个 Skill
    - 如果不存在，需要先上传该 Skill 的配置文件
    - 如果已存在，进入下一步
4. **下发到工作空间** — 将该 Skill 关联到你的目标工作空间，点击下发/同步按钮
5. **验证下发成功** — 回到 IDE，确认工作空间已接收到最新的 Skill 配置

> ⚠️ **注意**：如果 Skill 未下发到工作空间，在 IDE 中输入 `/hooks-creator-gientech` 将无法识别，技能不会生效。

验证方法

在 IDE 的 Agent 对话窗口中输入 `/`，如果智能提示中出现了 `hooks-creator-gientech` 选项，说明技能已下发成功 ✅

📋 第 2 步：调用技能

确认技能已就绪后，在 IDE 的 Agent 对话窗口中，输入以下指令唤起技能：

```
/hooks-creator-gientech 创建一个hook，当任务完成的时候提醒我任务完成，并给我发一个系统通知消息。
```

> 💡 **提示**：`/hooks-creator-gientech` 是技能名称，后面跟的是你的具体需求描述。描述越清晰，技能生成的配置越精准。

💬 第 3 步：回答引导问题

技能会通过问答方式引导你确定需求，你需要回答 1-2 个简单问题。

问题 1：选择触发时机

```
你希望通知在哪个时机触发?
□ 每轮任务完成时（TurnStop）— 推荐
□ 会话结束时（SessionEnd）
```

✅ **推荐选择：「每轮任务完成时（TurnStop）」**

> **为什么选 TurnStop?**
>
> - `TurnStop` 在每一轮 Agent 执行完成后都会触发，适合做**进度提醒**
> - `SessionEnd` 只在会话销毁时触发一次，适合做**清理和汇总**
> - 如果你希望每完成一个步骤都得到反馈，选 TurnStop 更合适

问题 2：选择通知方式

```
通知方式偏好是?
□ 仅系统通知（桌面弹窗）
□ 系统通知 + 提示音（声音+弹窗）
```

✅ **推荐选择：「系统通知 + 提示音」**

> 组合提醒效果最好：提示音引起注意，弹窗显示具体内容，双重保障不会错过。

⚙️ 第 4 步：技能自动生成文件

回答完问题后，技能会自动生成以下文件：

生成的文件结构

```
你的项目根目录/
├── .gientech/
│   └── hooks/
│       ├── config.json          ← Hook 配置文件（核心）
│       └── notify-turn.sh       ← 通知脚本（可自定义）
```

📄 config.json 详解

```json
{
  "hooks": {
    "TurnStop": [
      {
        "command": "bash .gientech/hooks/notify-turn.sh",
        "timeout": 5000
      }
    ]
  }
}
```

字段	说明	注意事项
`hooks`	所有 Hook 配置的根对象	固定写法
`TurnStop`	事件名，表示"每轮任务完成时"	区分大小写，不能写错
`command`	要执行的脚本路径（相对于项目根目录）	必须以 `.gientech/hooks/` 开头
`timeout`	脚本超时时间（毫秒）	注意是 `timeout`，不是 `timeout_ms`

📄 notify-turn.sh 脚本逻辑

脚本的核心逻辑是：

1. **读取环境变量** — 通过 `HOOK_EVENT` 获取事件名，`HOOK_INPUT` 获取事件数据
2. **判断任务状态** — 从输入中解析 `stop_reason`，区分正常完成/出错/中止
3. **执行通知** — 播放系统提示音（`afplay`）+ 发送桌面通知（`osascript`）
4. **输出结果** — 非阻塞事件只需输出 `{}`

> ⚠️ **重要规则**：脚本通过 `HOOK_INPUT` 环境变量读取数据，**不能从 stdin 读取**，否则会导致脚本永久阻塞！

🔍 第 5 步：验证配置

技能会自动完成以下验证，你也可以手动检查：

验证 JSON 格式

```
python3 -c "import json; json.load(open('.gientech/hooks/config.json')); print('✅ JSON 格式正确')"
```

验证脚本可执行

```
chmod +x .gientech/hooks/notify-turn.sh
ls -la .gientech/hooks/
```

验证清单

检查项	正确写法	错误写法
事件名拼写	`TurnStop`	`Turnstop`、`turn_stop`
字段名	`timeout`	`timeout_ms`、`time_out`
字段名	`command`	`cmd`、`script`、`path`
脚本路径	`.gientech/hooks/notify-turn.sh`	`hooks/notify.sh`（缺少相对路径前缀）

🚀 第 6 步：重启会话生效

配置完成后，**重启当前 Gincoder IDE 或者 新建 会话**，新的 Hook 配置就会自动加载。（默认加载当前工程）

> 不需要重启 IDE，只需要在对话窗口中开始一个新会话即可。

验证是否生效

当任务完成时，你会看到/听到：

1. 🔊 **一声提示音**（Hero 音效）
2. 💬 **桌面右上角弹窗**，显示"任务完成通知 — 任务完成！"

如果任务出错或中止，通知内容也会相应变化：

- 正常完成 → "任务完成！" ✅
- 执行出错 → "本轮执行出错" ❌
- 手动中止 → "本轮被中止" ⏹️

🧩 扩展：注册更多 Hook

掌握了基础流程后，你可以举一反三，注册更多场景的 Hook。

支持的 Hook 事件一览

事件	触发时机	可阻塞	典型用途
`PreToolUse`	工具执行前	✅	权限检查、参数修改
`PostToolUse`	工具执行成功后	❌	日志记录、结果处理
`PostToolUseFailure`	工具执行失败后	❌	错误处理、重试逻辑
`SessionStart`	会话启动时	❌	注入上下文、加载配置
`SessionEnd`	会话销毁时	❌	清理资源、生成报告
`TurnStop`	Turn 结束时	❌	日志记录、状态更新、通知
`UserPromptSubmit`	用户提交消息时	✅	消息过滤、添加上下文
`SubagentStart`	子 Agent 启动时	❌	日志记录、统计
`SubagentStop`	子 Agent 停止时	✅	停止父 Agent
`StreamEnd`	流结束时	✅	停止会话

💡 思考环节：Hook 还能用在哪些研发场景？

现在你已经掌握了配置 Hook 的完整流程。停下来想一想：**除了任务完成通知，Hook 还能在你的日常研发中解决哪些痛点？**

下面是一些研发场景的参考思路，看看哪些戳中了你的痛点 👇

🔐 场景一：安全门禁 — 防止误操作

痛点：手滑执行了 `rm -rf /`、`DROP TABLE`、`git push --force` 等毁灭性操作

Hook 方案：`PreToolUse` 拦截危险命令

```
# 检测到 rm -rf 项目根目录 → 直接拒绝
# 检测到 git push --force → 弹出确认询问
```

效果：Agent 执行危险操作前会先问你是否确认，避免悲剧 🛡️

> 调用方式：`/hooks-creator-gientech 我想在工具执行前检查危险命令，比如 rm -rf，禁止执行`

🧪 场景二：自动运行测试

痛点：改完代码忘记跑测试，提交后 CI 挂了

Hook 方案：`PostToolUse` 监听代码文件变更后自动跑测试

```
// 检测到 PostToolUse 事件
// 如果修改的是 src/**/*.java 或 src/**/*.ts
// 自动执行：npm test 或 mvn test -pl 对应模块
```

效果：改完代码，Agent 自动帮你跑一遍相关测试，红灯马上知道 🔴🟢

> 调用方式：`/hooks-creator-gientech 我想在修改代码文件后自动运行相关测试`

📚 场景三：项目上下文自动注入

痛点：每次新开会话都要跟 Agent 说一遍"我们的项目是 Java 21 + Spring Boot 3..."

Hook 方案：`SessionStart` + `UserPromptSubmit` 自动注入上下文

事件	做什么
`SessionStart`	检测 `pom.xml` / `package.json`，自动注入项目技术栈信息
`UserPromptSubmit`	在你每次提问前，自动追加当前项目的架构说明

效果：Agent 从第一句话就了解你的项目，不用反复交代 📖

> 调用方式：`/hooks-creator-gientech 我需要在会话启动时自动检测项目类型，注入上下文信息`

🔄 场景四：错误自动重试

痛点：Agent 执行命令失败，有时候是网络抖动，重试一次就好了

Hook 方案：`PostToolUseFailure` 自动重试

```
// 检测到工具执行失败
// 如果是网络超时类错误 → 自动重试 1 次
// 如果还是失败 → 记录错误详情到日志文件
```

效果：偶发性的网络问题自动解决，不打断你的工作流 🔄

> 调用方式：`/hooks-creator-gientech 我想在工具执行失败时自动重试`

👀 场景五：关键文件变更监控

痛点：不知道谁改了 `application.yml` 或 `Dockerfile`，导致线上出问题

Hook 方案：`PostToolUse` 监听关键文件变更 + 通知

```
# 检测到 .env、application*.yml、Dockerfile 被修改
# 发送桌面通知："⚠️ 注意：application-prod.yml 已被修改"
# 同时记录变更前后的内容对比
```

效果：关键配置被改，你第一时间知道 🚨

> 调用方式：`/hooks-creator-gientech 我想监控关键配置文件（如 application.yml）的修改并通知我`

📊 场景六：工具调用统计

痛点：想知道 Agent 每天在做什么，花了多少时间

Hook 方案：所有事件全量记录 → 生成统计报告

```
# 记录每次工具调用及耗时
# 统计：今天 Agent 读了多少文件、写了多少代码、
#       执行了多少命令、耗时多少
# SessionEnd 时导出统计报告
```

效果：数据驱动，知道 Agent 的工作效率 📈

> 调用方式：`/hooks-creator-gientech 我想记录每次工具调用的信息，方便统计和调试`

📋 场景决策速查表

场景	推荐事件	复杂度	研发价值
🔐 安全门禁	`PreToolUse`	⭐	💎💎💎💎💎
🧪 自动跑测试	`PostToolUse`	⭐⭐⭐	💎💎💎💎💎
📚 上下文注入	`SessionStart`	⭐	💎💎💎💎
🔄 错误重试	`PostToolUseFailure`	⭐⭐	💎💎💎
👀 文件监控	`PostToolUse`	⭐	💎💎💎
📊 调用统计	全事件	⭐⭐⭐	💎💎💎

> ✍️ **你的思考**：以上场景中，哪个最符合你当前项目的痛点？或者你还能想到其他可以用 Hook 解决的问题？欢迎记录下来，后续一一实践。

❓ 常见问题

Q1: Hook 没有生效怎么办？

1. 检查 `config.json` 是否在 `.gientech/hooks/` 目录下
2. 检查 JSON 格式是否正确（可用 `python3 -m json.tool` 验证）
3. 确认事件名拼写正确（区分大小写）
4. 确认脚本路径正确（相对于项目根目录）
5. 重启会话后再试

Q2: 脚本执行报错怎么排查？

- 非阻塞事件（如 `TurnStop`）出错不会影响主流程，错误被静默忽略

- 可以手动执行脚本测试：`bash .gientech/hooks/notify-turn.sh`
- 检查脚本是否有执行权限：`chmod +x .gientech/hooks/notify-turn.sh`

Q3: 一个事件可以注册多个 Hook 吗?

可以。按数组顺序依次执行，如果某个 Hook 返回 `{ continue: false }`，后续 Hook 不再执行。

```
{
  "hooks": {
    "PreToolUse": [
      { "command": "node .gientech/hooks/security-check.js", "timeout": 5000 },
      { "command": "node .gientech/hooks/log-tool-call.js" }
    ]
  }
}
```

Q4: 如何修改已有配置?

直接编辑 `.gientech/hooks/config.json` 和对应的脚本文件即可，修改后重启会话生效。

📚 总结

通过本教程，你完成了：

1. ✅ 确认 Skill 已上传并下发到工作空间
2. ✅ 调用 `/hooks-creator-gientech` 技能
3. ✅ 回答引导问题确定需求
4. ✅ 自动生成 `config.json` 和脚本文件
5. ✅ 验证配置正确性
6. ✅ 重启会话使 Hook 生效

核心要点记住三件事：

- 🎯 **明确需求** — 想清楚触发时机和要做什么
- 📝 **字段名不能错** — `command`、`timeout`、事件名都要拼写正确
- 🔄 **重启会话** — 修改配置后必须重启才能生效

培训材料版本：v1.0 | 基于 /hooks-creator-gientech 技能实践编写