



中國人民大學
RENMIN UNIVERSITY OF CHINA



中国人民大学高瓴人工智能学院
Gaoling School of Artificial Intelligence, Renmin University of China

从Autocomplete到CLI Agent: Vibe Coding and Anything

卫雅珂

yakewei@ruc.edu.cn





卫雅珂

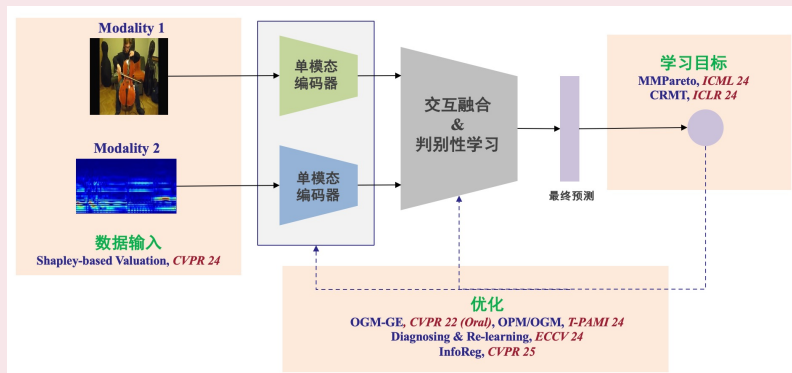
中国人民大学高瓴人工智能学院博士，导师为胡迪副教授

研究方向：多模态学习、多模态大模型

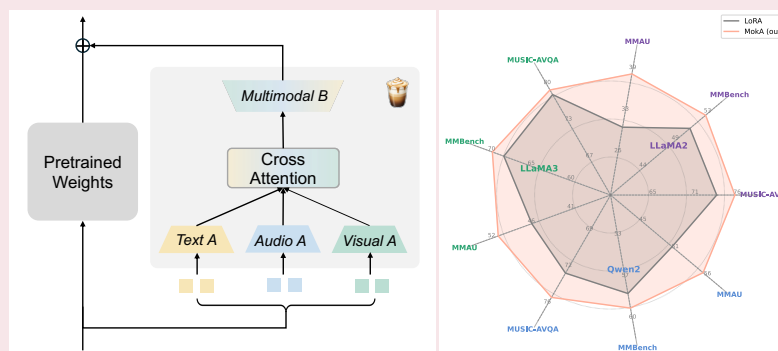
曾获博士生国家奖学金、百度奖学金（全球遴选10人）、入选CVPR博士生论坛等

8月将入职中国人民大学高瓴人工智能学院

代表性成果一 小模型中的模态信息平衡利用



代表性成果二 多模态大模型中的模态高效协同



第一阶段: Autocomplete



GitHub Copilot

2021年开始测试, 正式上线于2022年

- 自动补全工具
- 根据注释提示可以补全函数

Top 5 AI Extensions for VSCode in 2023

By Augustine Ezeh · Published on July 25, 2023

0 comments

TABLE OF CONTENTS

1. GitHub Copilot
 - How does it work?
 - How good is GitHub Copilot?
2. Swimm
 - Create Documentation structure
 - Generate documentation code explanations
 - Pull Request to Documentation
 - Enhance documentation visibility
3. Tabnine
4. Blackbox
 - Blackbox Code Search by comment
5. IntelliCode
 - IntelliCode API Usage Examples
 - IntelliCode Completions
 - Conclusion
 - Comments

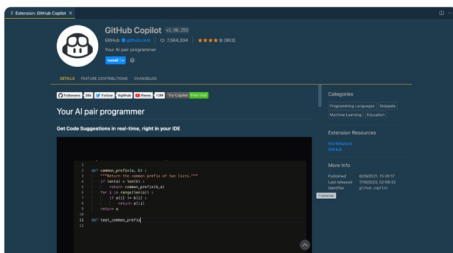


In the fast-paced world of software development, one innovation stands out as a true game-changer: Artificial Intelligence (AI). With its remarkable capabilities, AI has revolutionised the way developers interact with code, reshaping the landscape of modern programming.

There are already more than 400 AI-infused extensions available in the Visual Studio Code Marketplace due to the emergence of new generative AI technologies in the software development industry. From empowering intelligent code suggestions to streamlining repetitive tasks, these AI-driven extensions have elevated developer productivity to unprecedented heights.

Here is a list of the top 5 AI Extensions for VSCode you should be using to improve your developer experience, productivity, and workflow efficiency.

1. GitHub Copilot



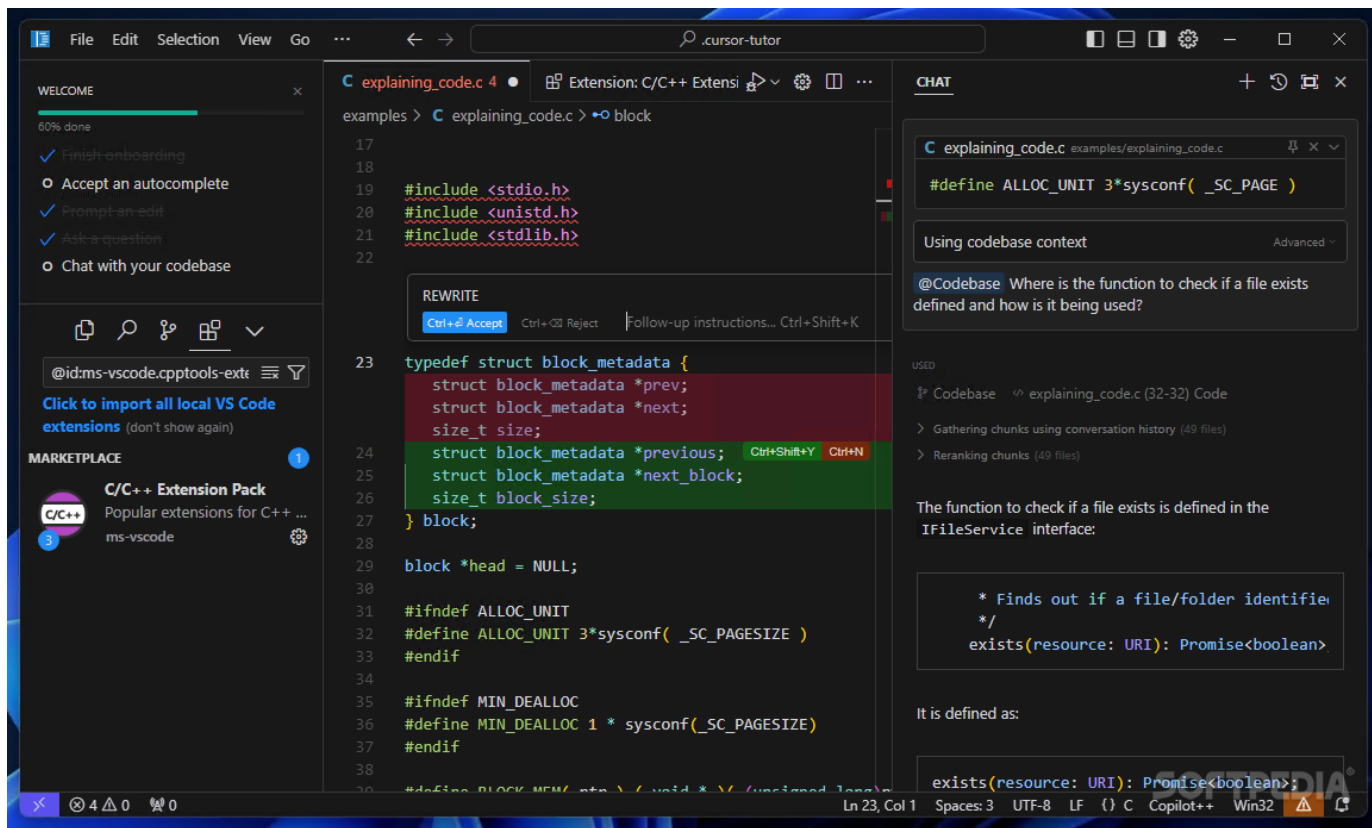
```
JS useRowActions.js
28   },
29   {
30     label: 'Modify rate/allocation',
31     selected: false,
32     onClick: (e, row) => {
33       const id = row[0]?.value;
34       const record = { ...getRecordById(id) };
35       setRowModal(<ModifyModal booking={record} onClose={e => setRowMod
36     },
37   },
38   {
39     label: 'Delete',
40     selected: false,
41     onClick: (e, row) => {
42       const id = row[0]?.value;
43       const booking = { ...getRecordById(id) };
44       setRowModal(<DeleteModal booking={booking} refreshParent={refresh
45     },
46   },
47 ];
48 return rowActions;
49 };
50
51 export default useRowActions;
52
```

第二阶段：IDE Agent



发布于2023年

- 大模型Agent内置在IDE中
- Human-agent协作
- 自然语言编程，prompt-response方式驱动，只需要校验修改
- 可以直接针对整个项目进行更改
- 配置多个主流大模型



第三阶段：CLI Agent



Claude Code发布于2025年7月

- Command Line Agent
- AI从“副驾驶”到“主驾驶”
- 只关心输出，不关心difference
- Human只关注如何定义任务，放手让AI全部执行，主动性更强
- 部署更自由，有terminal的地方就有agent
- **Beyond coding**



从Autocomplete到CLI Agent



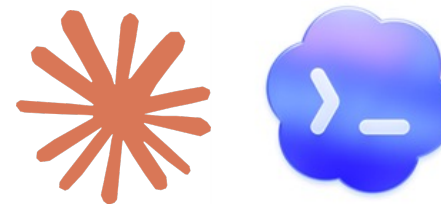
**GitHub
Copilot**

2021年开始测试，正式上线于2022年
AI代码补全工具，可生成函数级代码



发布于2023年
AI原生IDE

以prompt-response的方式完成任务



Claude Code发布于2025年7月
Command Line Agent
AI独立完成任务

QuickStart
快速上手

Rules
永久上下文

Memory
项目内部上下文

Permissions
命令执行权限

Hooks
状态触发事件

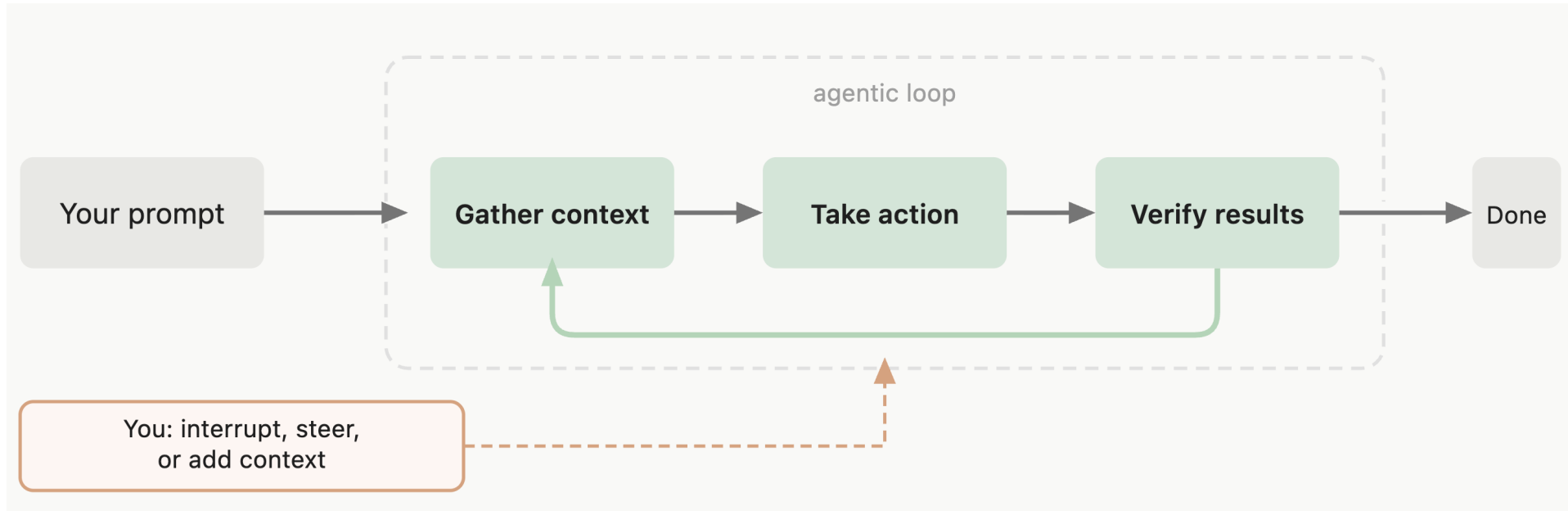
Skills
预置SOP

SubAgent
各自执行

AgentTeam
多Agent合作

Sandbox
隔离的运行环境

Harness
Engineering
更好驾驭Agent



以我们的NeurIPS论文MokA的项目代码为例，如何快速上手并进行修改和拓展

```
Claude Code v2.1.91
Welcome back!

Sonnet 4.6 · API Usage Billing
~/Downloads/demo/MokA

Tips for getting started
✓ Run /init to create a CLAUDE.md file with in...

Recent activity
No recent activity

/init

Explore(Explore codebase structure)
└ Done (31 tool uses · 40.6k tokens · 50s)
(ctrl+o to expand)

Write(CLAUDE.md)
└ Wrote 79 lines to CLAUDE.md
  # CLAUDE.md

  This file provides guidance to Claude Code (claude.ai/code) when working
  with code in this repository.

  ## Project Overview

  MokA is a multimodal LLM framework combining visual, audio, and text modalities, built on a modified LLaVA architecture with QFormer-based projectors and LoRA fine-tuning.

  ## Commands
  ... +69 lines (ctrl+o to expand)

Created CLAUDE.md with:

- Commands for distributed training (shell/train.sh), direct torchrun usage, and all 4 evaluation benchmarks
- Architecture overview with a data-flow diagram showing how frozen encoders → QFormer projectors → LLaVA LLM + LoRA connects
- Component breakdown of models/, modified_models/, modified_peft/, train/, and eval_benchmarks/
- Training config details (batch size, lr, BF16, ZeRO Stage 2) and data format (LLaVA JSON, 32 image tokens)
```

如何快速了解一个新项目？

/init

总结当前codebase的基本结构。

然后继续提问从high-level到specific:

- ❖ *give me an overview of this codebase*
- ❖ *explain the main architecture patterns used here*
- ❖ *what are the model backbone?*

以我们的NeurIPS论文MokA的项目代码为例，如何快速上手并进行修改和拓展

```
MokA claude
Claude Code v2.1.91
Sonnet 4.6 · API Usage Billing
~/Downloads/demo/MokA

) 修改modified_peft, 修改原本设计中的cross-attention模块, 改为使用文本question作为query进行跨模态attention

Explore(Explore modified_peft directory)
└─ Bash(ls -la /Users/ykw/Downloads/demo/MokA/models/ | head -30)
└─ Bash(grep -r "projector\|QFormer" /Users/ykw/Downloads/demo/MokA/models/*.py 2>/dev/null | grep -i "class\|def" | head -20)
└─ Search(pattern: "class.*Projector|class.*QFormer", path: "models")
└─ Interrupted · What should Claude do instead?

) 查看当前LoRA的实现

Searched for 1 pattern, read 1 file (ctrl+o to expand)

现在我理解了当前设计: image token 作为 query, question token 作为 key/value.

要改为 question token 作为 query, image token 作为 key/value, 只需修改 Linear.forward 中的 cross-attention 块 (第 627-653 行):

Update(modified_peft/tuners/lora/layer.py)

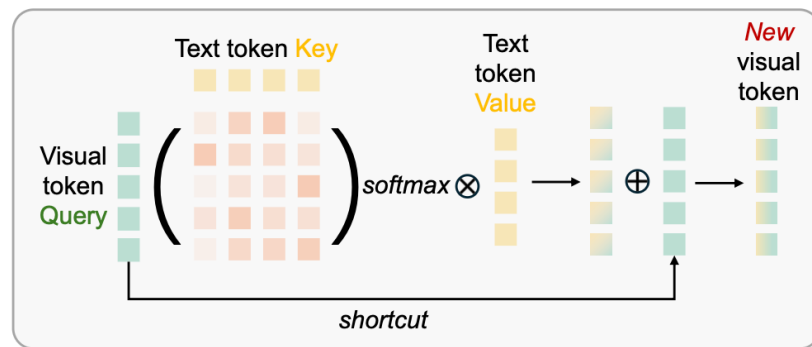
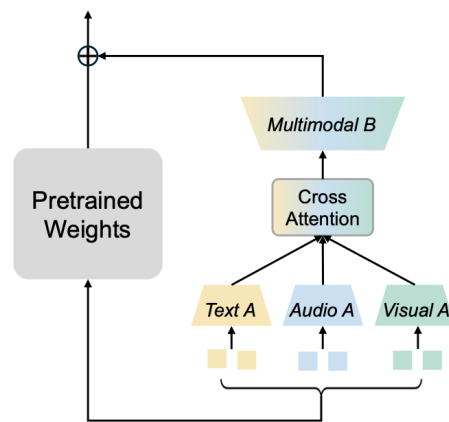
Edit file
modified_peft/tuners/lora/layer.py

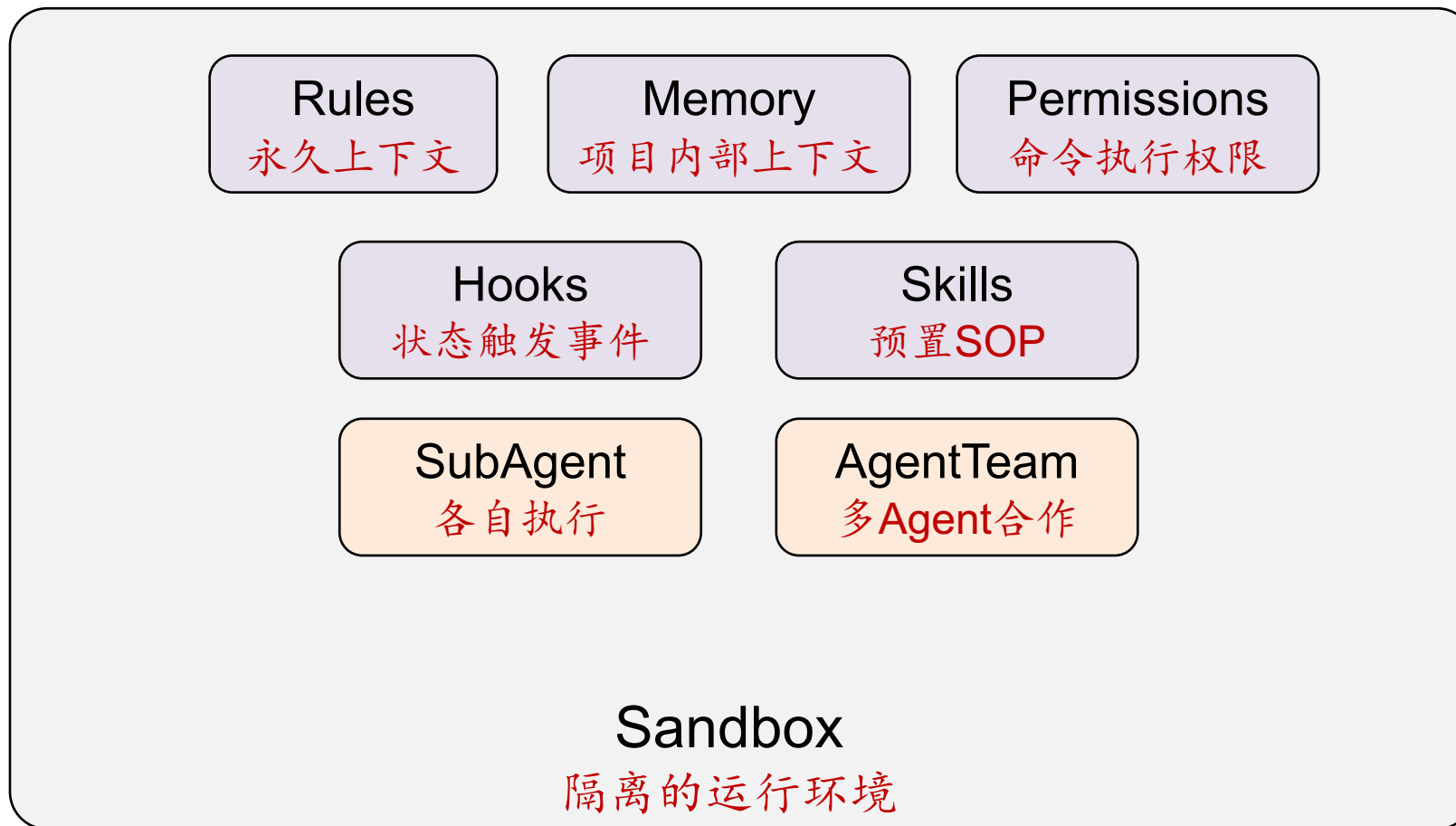
# Step 2: reshape 回 batch 维度, 进行 cross attention
lora_A_output = lora_A_output_flat.reshape(batch, seq_len, 1)

626 - # 如果有 question_mask, 对 image token 和 question token 做 cross
    -attention
626 + # 如果有 question_mask, 对 question token 和 image token 做 cross
    +attention

    if (image_mask_flat == 1).any():
        for i in range(batch):
            # 获取当前样本的 image token 索引
            image_indices = torch.where(my_image_mask[i] == 1)[0]
            if len(image_indices) == 0:
                continue
            629 -
            630 -
            631 -
            632 -
            633 -
```

如何基于该项目进行拓展? 此时以MokA中的交互模块设计为示例。





Rules-全局上下文



Rules: ~/.claude/CLAUDE.md

永久上下文，每次新session都会载入，显式制定规则，例如coding style，错误处理等等。
不建议太长，如有需求可以拆解为多个markdown，按需载入

```
Claude Code v2.1.70
Opus 4.6 with high effort · Claude Max
~/_my_local/code-local/personal-workspace

send me ~/.claude/CLAUDE.md content

Here's your ~/.claude/CLAUDE.md:

---
Global Rules

Code Style

- Comments in English only
- Prefer functional programming over OOP
- Use OOP classes only for connectors and interfaces to external systems
- Write pure functions - only modify return values, never input parameters or global state
- Follow DRY, KISS, and YAGNI principles
- Use strict typing everywhere - function returns, variables, collections
- Check if logic already exists before writing new code
- Avoid untyped variables and generic types
- Never use default parameter values - make all parameters explicit
- Create proper type definitions for complex data structures
- All imports at the top of the file
- Write simple single-purpose functions - no multi-mode behavior, no flag parameters that switch logic. If the user needs multiple modes, they will ask explicitly

Error Handling

- Always raise errors explicitly, never silently ignore them
- Use specific error types that clearly indicate what went wrong
- Avoid catch-all exception handlers that hide the root cause
- Error messages should be clear and actionable
- NO FALLBACKS: Code should either succeed or fail with a clear error. Fallbacks are only allowed if the user explicitly asks for them. Never add automatic fallbacks - they hide real problems
- Transparent debugging: When something fails, show exactly what went wrong and why
- Fix root causes, not symptoms - fallbacks hide real problems that need solving
- External API/service calls: use retries with warnings, raise the last error if all attempts fail
- Error messages must include enough context to debug (request params, response body, status codes) - no generic "something went wrong"
- Logging: no dynamic values interpolated into log message strings - pass them as structured data or extra fields. Exceptions can use f-strings for readability
```

```
~/.claude/rules/
├── code-style.md
├── testing.md
├── api-conventions.md
└── security.md
```

Memory-项目内上下文

Memory: `~/.claude/projects/<project>/memory/`

Claude code会为每个项目维护memory目录，MEMORY.md为入口文件，在整个会话期间，Claude会读取和写入此目录中的文件。

记录当前项目的debug信息，以及当前项目需要遵循的要求等等。

```
~/.claude/projects/<project>/memory/  
├── MEMORY.md          # Concise index, loaded into every session  
├── debugging.md      # Detailed notes on debugging patterns  
├── api-conventions.md # API design decisions  
└── ...                # Any other topic files Claude creates
```

Permissions-可执行边界

Permissions: /permissions

如何防止cli agent删库?

Allow, Ask, Deny

Tool type	Example	Approval required	"Yes, don't ask again" behavior
Read-only	File reads, Grep	No	N/A
Bash commands	Shell execution	Yes	Permanently per project directory and command
File modification	Edit/write files	Yes	Until session end

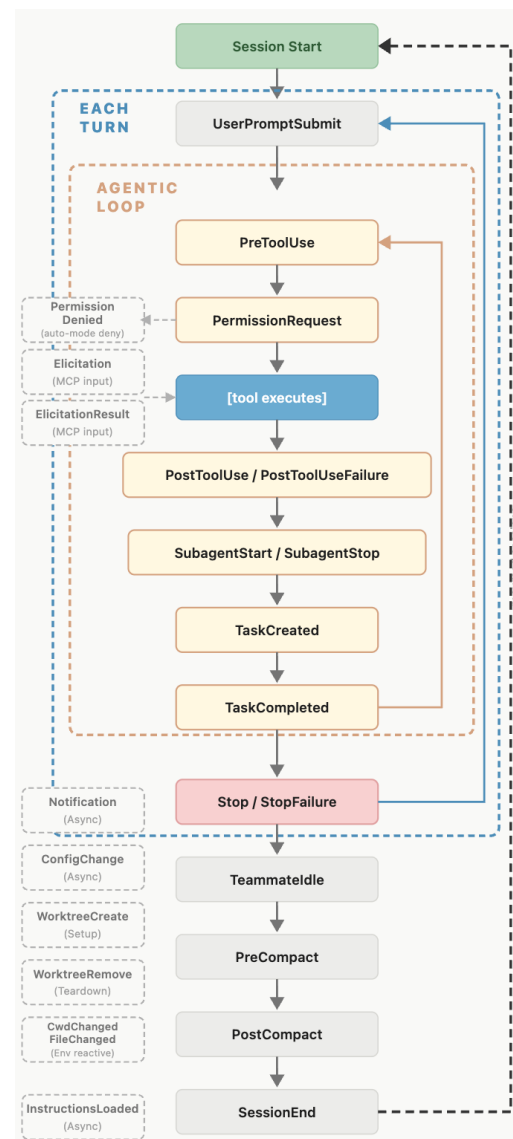
```
> /permissions  
  
Permissions: Recently denied Allow Ask Deny Workspace  
  
Claude Code can read files in the workspace, and make edits when auto-accept edits is on.  
  
- /Users/ykw/Downloads/demo/MokA (Original working directory)  
1. Add directory...
```

Hooks-自动化



Why Hooks?

- 自定义的shell命令，执行确定性脚本，在session的生命周期的特定环节触发。
- Event-driven automations.
- 需要指定：什么时候触发、针对什么操作、运行什么脚本



一个session的生命周期

Hooks-自动化

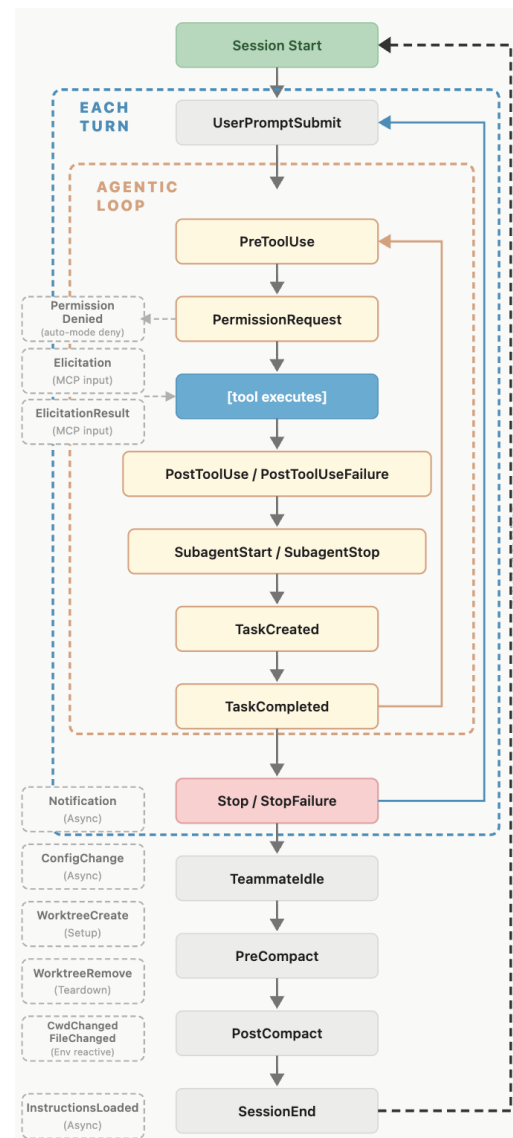


```
{
  "hooks": {
    "CwdChanged": [
      {
        "hooks": [
          {
            "type": "command",
            "command": "direnv export bash >> \"\${CLAUDE_ENV_FILE}\""
          }
        ]
      }
    ]
  }
}
```

在文件切换时切换不同的环境变量

```
{
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "Edit|Write",
        "hooks": [
          {
            "type": "command",
            "command": "jq -r '.tool_input.file_path' | xargs npx prettier --write"
          }
        ]
      }
    ]
  }
}
```

编辑后自动格式化代码



一个session的生命周期

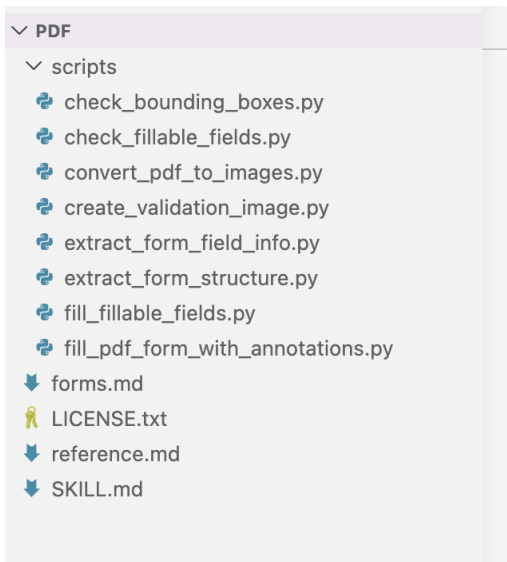
Skills-标准化

Why Skills?

- 大模型的SOP，抽象和复用能力，把流程标准化，避免反复说明，模型反复进行思考消耗。
- 可以是需要注入的domain knowledge，可以是对某类型文件的多种操作，可以是任一固定流程
- 包含背景信息、工具使用说明、参考范式/格式规范等等
 - 处理pdf, xlsx等各种格式文件
 - 抓取并总结当日hugging face daily paper, 保存到Notion页面
 - 在指定的标准benchmark评测模型
 - 等等

Why Skills?

- 大模型的SOP，抽象和复用能力，把流程标准化，避免反复进行思考消耗。
- 包含背景信息、工具使用说明、参考范式/格式规范等等
- 渐进式披露



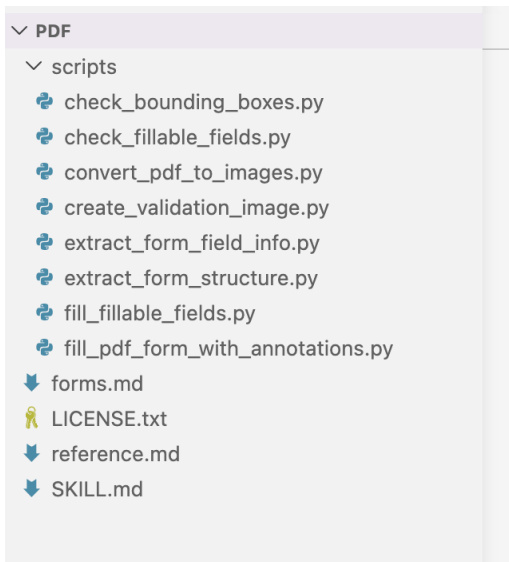
```
1 ---
2 name: pdf
3 description: Use this skill whenever the user wants to do anything with PDF
4 files. This includes reading or extracting text/tables from PDFs, combining
5 or merging multiple PDFs into one, splitting PDFs apart, rotating pages,
6 adding watermarks, creating new PDFs, filling PDF forms, encrypting/
7 decrypting PDFs, extracting images, and OCR on scanned PDFs to make them
8 searchable. If the user mentions a .pdf file or asks to produce one, use
9 this skill.
10 license: Proprietary. LICENSE.txt has complete terms
11 ---
12 # PDF Processing Guide
13 ## Overview
14 This guide covers essential PDF processing operations using Python libraries
15 and command-line tools. For advanced features, JavaScript libraries, and
16 detailed examples, see REFERENCE.md. If you need to fill out a PDF form,
17 read FORMS.md and follow its instructions.
18 ## Quick Start
19 ```python
20 from pypdf import PdfReader, PdfWriter
21
22 # Read a PDF
23 reader = PdfReader("document.pdf")
24 print(f"Pages: {len(reader.pages)}")
25
26 # Extract text
27 text = ""
28 for page in reader.pages:
29     text += page.extract_text()
30 ```
```

```
27 ## Python Libraries
28
29 ### pypdf - Basic Operations
30
31 #### Merge PDFs
32 ```python
33 from pypdf import PdfWriter, PdfReader
34
35 writer = PdfWriter()
36 for pdf_file in ["doc1.pdf", "doc2.pdf", "doc3.pdf"]:
37     reader = PdfReader(pdf_file)
38     for page in reader.pages:
39         writer.add_page(page)
40
41 with open("merged.pdf", "wb") as output:
42     writer.write(output)
43 ```
44
45 #### Split PDF
46 ```python
47 reader = PdfReader("input.pdf")
48 for i, page in enumerate(reader.pages):
49     writer = PdfWriter()
50     writer.add_page(page)
51     with open(f"page_{i+1}.pdf", "wb") as output:
52         writer.write(output)
53 ```
54
```

Why Skills?

- 大模型的SOP，抽象和复用能力，把流程标准化，避免反复进行思考消耗。
- 包含背景信息、工具使用说明、参考范式/格式规范等等
- 渐进式披露

Skill姓名及描述：始终加载（~100 token）



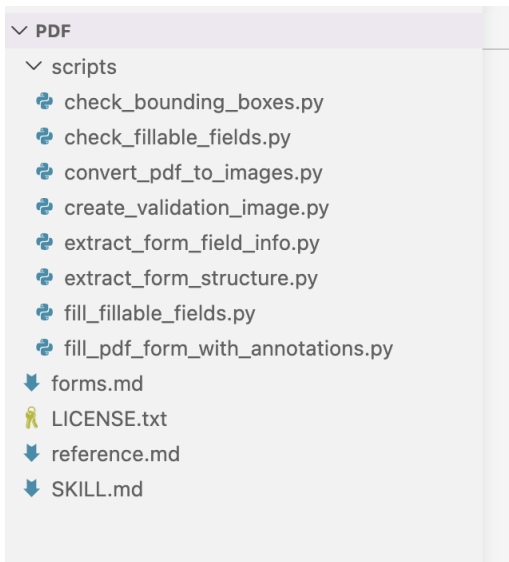
```
1  ---
2  name: pdf
3  description: Use this skill whenever the user wants to do anything with PDF
4  files. This includes reading or extracting text/tables from PDFs, combining
5  or merging multiple PDFs into one, splitting PDFs apart, rotating pages,
6  adding watermarks, creating new PDFs, filling PDF forms, encrypting/
7  decrypting PDFs, extracting images, and OCR on scanned PDFs to make them
8  searchable. If the user mentions a .pdf file or asks to produce one, use
9  this skill.
10 license: Proprietary. LICENSE.txt has complete terms
11
12 # PDF Processing Guide
13
14 ## Overview
15
16 This guide covers essential PDF processing operations using Python libraries
17 and command-line tools. For advanced features, JavaScript libraries, and
18 detailed examples, see REFERENCE.md. If you need to fill out a PDF form,
19 read FORMS.md and follow its instructions.
20
21 ## Quick Start
22
23 ```python
24 from pypdf import PdfReader, PdfWriter
25
26 # Read a PDF
27 reader = PdfReader("document.pdf")
28 print(f"Pages: {len(reader.pages)}")
29
30 # Extract text
31 text = ""
32 for page in reader.pages:
33     text += page.extract_text()
34 ```
```

```
27 ## Python Libraries
28
29 ### pypdf - Basic Operations
30
31 #### Merge PDFs
32 ```python
33 from pypdf import PdfWriter, PdfReader
34
35 writer = PdfWriter()
36 for pdf_file in ["doc1.pdf", "doc2.pdf", "doc3.pdf"]:
37     reader = PdfReader(pdf_file)
38     for page in reader.pages:
39         writer.add_page(page)
40
41 with open("merged.pdf", "wb") as output:
42     writer.write(output)
43 ```
44
45 #### Split PDF
46 ```python
47 reader = PdfReader("input.pdf")
48 for i, page in enumerate(reader.pages):
49     writer = PdfWriter()
50     writer.add_page(page)
51     with open(f"page_{i+1}.pdf", "wb") as output:
52         writer.write(output)
53 ```
54
```

Why Skills?

- 大模型的SOP，抽象和复用能力，把流程标准化，避免反复进行思考消耗。
- 包含背景信息、工具使用说明、参考范式/格式规范等等
- 渐进式披露

Skill姓名及描述：始终加载（~100 token）



```
1  ---
2  name: pdf
3  description: Use this skill whenever the user wants to do anything with PDF
4  files. This includes reading or extracting text/tables from PDFs, combining
5  or merging multiple PDFs into one, splitting PDFs apart, rotating pages,
6  adding watermarks, creating new PDFs, filling PDF forms, encrypting/
7  decrypting PDFs, extracting images, and OCR on scanned PDFs to make them
8  searchable. If the user mentions a .pdf file or asks to produce one, use
9  this skill.
10 license: Proprietary. LICENSE.txt has complete terms
11
12 # PDF Processing Guide
13
14 ## Overview
15
16 This guide covers essential PDF processing operations using Python libraries
17 and command-line tools. For advanced features, JavaScript libraries, and
18 detailed examples, see REFERENCE.md. If you need to fill out a PDF form,
19 read FORMS.md and follow its instructions.
20
21 ## Quick Start
22
23 ```python
24 from pypdf import PdfReader, PdfWriter
25
26 # Read a PDF
27 reader = PdfReader("document.pdf")
28 print(f"Pages: {len(reader.pages)}")
29
30 # Extract text
31 text = ""
32 for page in reader.pages:
33     text += page.extract_text()
34 ```
```

具体的事件描述及处理方式

```
27 ## Python Libraries
28
29 ### pypdf - Basic Operations
30
31 #### Merge PDFs
32 ```python
33 from pypdf import PdfWriter, PdfReader
34
35 writer = PdfWriter()
36 for pdf_file in ["doc1.pdf", "doc2.pdf", "doc3.pdf"]:
37     reader = PdfReader(pdf_file)
38     for page in reader.pages:
39         writer.add_page(page)
40
41 with open("merged.pdf", "wb") as output:
42     writer.write(output)
43 ```
44
45 #### Split PDF
46 ```python
47 reader = PdfReader("input.pdf")
48 for i, page in enumerate(reader.pages):
49     writer = PdfWriter()
50     writer.add_page(page)
51     with open(f"page_{i+1}.pdf", "wb") as output:
52         writer.write(output)
53 ```
54
```

可能涉及对python脚本，
shell命令的执行

Skills-标准化

Why Skills?

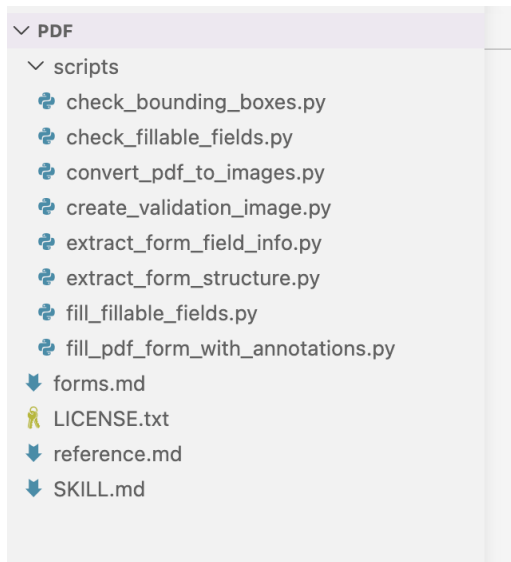
- 大模型的SOP，抽象和复用能力，把流程标准化，避免反复进行思考消耗。
- 包含背景信息、工具使用说明、参考范式/格式规范等等
- 渐进式披露

SKILL.md中的其他内容：触发时加载 (<5k token)

两种触发方式：

1. 模型生成过程中若判定需求与该 skill 匹配，则读取
2. 用户主动在prompt中斜杠调用

Skill姓名及描述：始终加载 (~100 token)



```
1  ---
2  name: pdf
3  description: Use this skill whenever the user wants to do anything with PDF
4  files. This includes reading or extracting text/tables from PDFs, combining
5  or merging multiple PDFs into one, splitting PDFs apart, rotating pages,
6  adding watermarks, creating new PDFs, filling PDF forms, encrypting/
7  decrypting PDFs, extracting images, and OCR on scanned PDFs to make them
8  searchable. If the user mentions a .pdf file or asks to produce one, use
9  this skill.
10 license: Proprietary. LICENSE.txt has complete terms
11
12 # PDF Processing Guide
13
14 ## Overview
15
16 This guide covers essential PDF processing operations using Python libraries
17 and command-line tools. For advanced features, JavaScript libraries, and
18 detailed examples, see REFERENCE.md. If you need to fill out a PDF form,
19 read FORMS.md and follow its instructions.
20
21 ## Quick Start
22
23 ```python
24 from pypdf import PdfReader, PdfWriter
25
26 # Read a PDF
27 reader = PdfReader("document.pdf")
28 print(f"Pages: {len(reader.pages)}")
29
30 # Extract text
31 text = ""
32 for page in reader.pages:
33     text += page.extract_text()
34 ```
```

具体的事件描述及处理方式

SKILL.md

```
27 ## Python Libraries
28
29 ### pypdf - Basic Operations
30
31 #### Merge PDFs
32 ```python
33 from pypdf import PdfWriter, PdfReader
34
35 writer = PdfWriter()
36 for pdf_file in ["doc1.pdf", "doc2.pdf", "doc3.pdf"]:
37     reader = PdfReader(pdf_file)
38     for page in reader.pages:
39         writer.add_page(page)
40
41 with open("merged.pdf", "wb") as output:
42     writer.write(output)
43 ```
44
45 #### Split PDF
46 ```python
47 reader = PdfReader("input.pdf")
48 for i, page in enumerate(reader.pages):
49     writer = PdfWriter()
50     writer.add_page(page)
51     with open(f"page_{i+1}.pdf", "wb") as output:
52         writer.write(output)
53 ```
54
```

可能设计对python脚本, shell命令的执行

文件夹中的其他内容：按需加载 (长度不限)

执行该skill过程中若需要则加载, SKILL.md中可能链接到文件夹中的其他资源

一个抓取每日论文的 skill例子

```
HF-DAILY-NOTION-BRIEF
├── agents
│   ├── openai.yaml
│   └── references
├── output_template.md
├── scripts
│   ├── cleanup_artifacts.py
│   ├── collect_translation_batches.py
│   ├── fetch_hf_daily.py
│   ├── publish_notion_from_file.py
│   ├── run_daily_brief.py
│   ├── run_prepare_daily_brief.py
│   ├── run_publish_daily_brief.py
│   └── summarize_papers.py
└── SKILL.md
```

```
SKILL.md > abc # HF Daily Notion Brief > abc ## Workflow
1 ---
2 name: hf-daily-notion-brief
3 description: 抓取 Hugging Face Daily Papers, 优先读取原始论文摘要, 只对关键词命中的论文生成 3-4 句中文摘要, 输出 Markdown 简报并发布到固定 Notion 父页面下按日期命名的子页面。
4 ---
5
6 # HF Daily Notion Brief
7
8 ## Use This Skill When
9
10 - 需要抓取 Hugging Face Daily Papers 并生成日报
11 - 需要把命中关键词的论文整理成中文 Markdown 简报
12 - 需要把日报发布到固定 Notion 父页面 `32848e40ebe580cf9502f20cbf09d4ad`
13
14 ## Fixed Rules
15
16 1. Run all scripts with the `py310` interpreter, and prefer the interpreter binary directly instead of `conda run`.
17 2. Default `TARGET_DATE` is yesterday in timezone `Asia/Shanghai`, and that default must come from the wrapper script itself.
18 3. Always use absolute `YYYY-MM-DD` for filename and Notion child page title.
19 4. Parent page is fixed: `32848e40ebe580cf9502f20cbf09d4ad` (do not change).
20 5. Publish Markdown as one complete document only (no chunk/block-by-block fallback).
21 6. For least token usage, publish via local file-based script instead of sending full Markdown text through tool arguments.
22 7. Publish scripts use the fixed built-in Notion token and must not read `NOTION_API_KEY` or `NOTION_TOKEN` from the environment.
23
24 ## Path Setup
25
26 Unless the user explicitly overrides it, use these stable direct-entry paths:
27
28 ```bash
29 PYTHON_BIN="/Users/yichenfan/opt/anaconda3/envs/py310/bin/python"
30 PREPARE_CMD="/Users/yichenfan/opt/anaconda3/envs/py310/bin/python /Users/yichenfan/my-skills/hf-daily-notion-brief/scripts/run_prepare_daily_brief.py"
31 PUBLISH_CMD="/Users/yichenfan/opt/anaconda3/envs/py310/bin/python /Users/yichenfan/my-skills/hf-daily-notion-brief/scripts/run_publish_daily_brief.py"
32 ```
33
```

使用时模型自动对skill具体内容进行改进

Skills-标准化

Why Skills?

- 大模型的SOP，抽象和复用能力，把流程标准化，避免反复说明，模型反复进行思考消耗。
- 可以是需要注入的domain knowledge，可以是对某类型文件的多种操作，可以是任一固定流程
- 包含背景信息、工具使用说明、参考范式/格式规范等等
 - 处理pdf, xlsx等各种格式文件
 - 抓取并总结当日hugging face daily paper，保存到Notion页面
 - 在指定的标准benchmark评测模型
 - 等等
- 不建议载入过多skills。占用过多context长度，且尤其在session窗口context已经比较长的情况下，模型无法回溯到注入的skills信息。
- 在输入指令圈定可能使用的skills范围，减少模型的大范围搜索

Why Sandbox?

- 多次审批造成疲劳感
- 影响生产力，减慢开发流程
- 干扰模型自主性

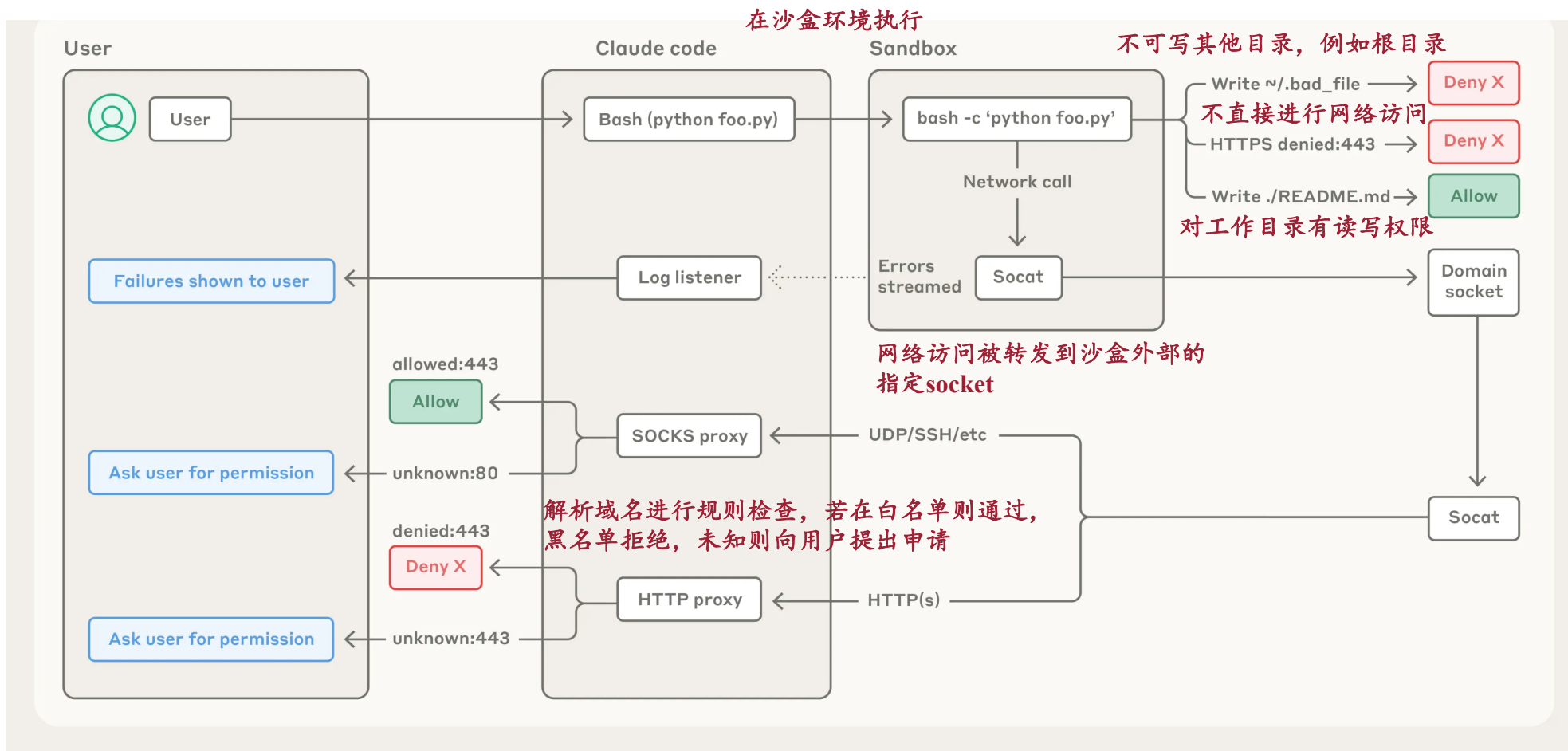
How?

- 明确权限范围
- 减少审批提示
- 增强自主性

Sandbox-受限环境安全运行



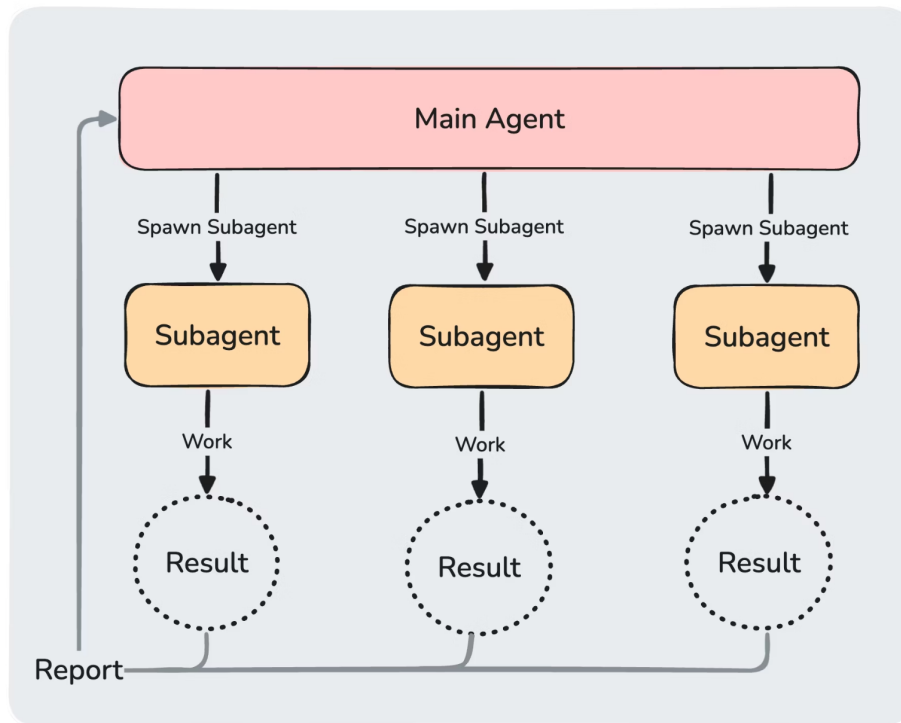
Claude Code沙盒设计：文件系统隔离&网络隔离



应对复杂任务：Subagents与Agent Team

Why Subagent?

- 执行特定任务，在隔离的session中完成任务，节省/避免污染主session的context
- 对于特定任务，可以在隔离的session中给出更精准的instruction，避免主session的context影响特定任务
- 将明确、固定的特定任务交给成本更低的模型，节省成本。例如，主session用opus，subagent用haiku
- 跨项目复用subagent
- Main agent调用或用户触发





应对复杂任务：Subagents与Agent Team



还是抓取每日论文的skill例子，对于中文摘要总结采用单独的子程序处理，隔离context，并行加速

Translation Prompt Template

When running subagents on the exported translation jobs, use a prompt with these constraints:

- Read one JSON item at a time or a small batch of items from the local jobs file.
- For each item, write `summary_zh` as 3-4 Chinese sentences.
- Stay faithful to the full English abstract.
- Preserve model names, dataset names, benchmark names, metrics, percentages, and technical terms.
- Do not add commentary like `这篇文章主要讲了`.
- Output strict JSON only.

Recommended per-item prompt shape:

```
```text
你在处理机器学习论文摘要。请基于给定的完整英文摘要，写 3-4 句忠实、紧凑的中文摘要。
要求：
1. 保留方法、任务、数据集、指标、结果等关键信息。
2. 不要编造原文没有的信息。
3. 不要输出解释、前言或 Markdown。
4. 只输出一个 JSON 对象，格式为 {"index":<int>,"summary_zh": "..."}。
```
```

Expected translation job item shape:

```
```json
{
 "index": 0,
 "title": "Paper title",
 "topics": ["LLM", "Multimodal"],
 "summary": "Full English abstract..."
}
```

共有 22 个待翻译条目，所以这轮按规则起 6 个子代理，批次会是 4/4/4/4/3/3。我先把每批 jobs 写到独立的临时文件，再并行分发。

后台终端已完成以及 /Users/yichenfan/opt/anaconda3/envs/py310/bin/python - <<'PY' import json from pathlib impor...

已生成 6 个智能体

已根据以下指令创建 Mendel (worker): Translate the ML paper abstracts in /tmp/hf\_batch\_jobs\_0.json into Chinese su...

已根据以下指令创建 Huygens (worker): Translate the ML paper abstracts in /tmp/hf\_batch\_jobs\_1.json into Chinese su...

已根据以下指令创建 Popper (worker): Translate the ML paper abstracts in /tmp/hf\_batch\_jobs\_2.json into Chinese su...

已根据以下指令创建 Kierkegaard (worker): Translate the ML paper abstracts in /tmp/hf\_batch\_jobs\_3.json into Chines...

已根据以下指令创建 Kepler (worker): Translate the ML paper abstracts in /tmp/hf\_batch\_jobs\_4.json into Chinese sum...

已根据以下指令创建 Meitner (worker): Translate the ML paper abstracts in /tmp/hf\_batch\_jobs\_5.json into Chinese su...

6 个翻译子代理已经分发出去了，每个只负责自己的 batch 文件，不会互相覆盖。我现在等第一轮全部落盘，然后按技能要求一次性 merge，再走 publish wrapper。

# 应对复杂任务：Subagents与Agent Team



## Why Agent team?

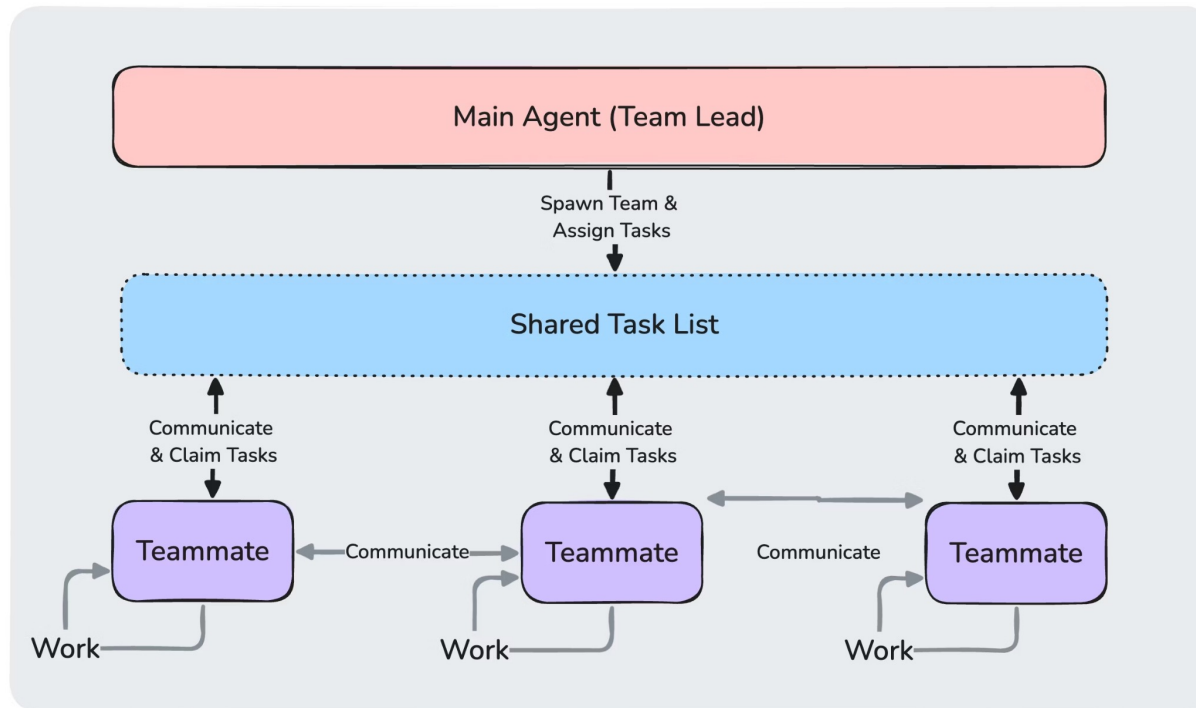
- Agent之间需要交流
- 需要多方面review正确性
- 并行处理讨论，更快收敛到正确答案

### 分别针对不同方面进行code review

```
Create an agent team to review PR #142. Spawn three reviewers:
- One focused on security implications
- One checking performance impact
- One validating test coverage
Have them each review and report findings.
```

### 对于code issue，并行开展检查，并且进行debate

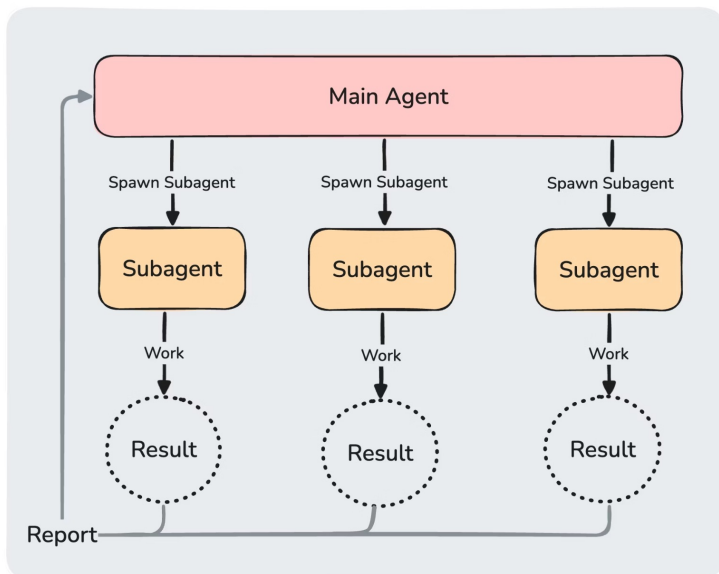
```
Users report the app exits after one message instead of staying connected.
Spawn 5 agent teammates to investigate different hypotheses. Have them talk to
each other to try to disprove each other's theories, like a scientific
debate. Update the findings doc with whatever consensus emerges.
```



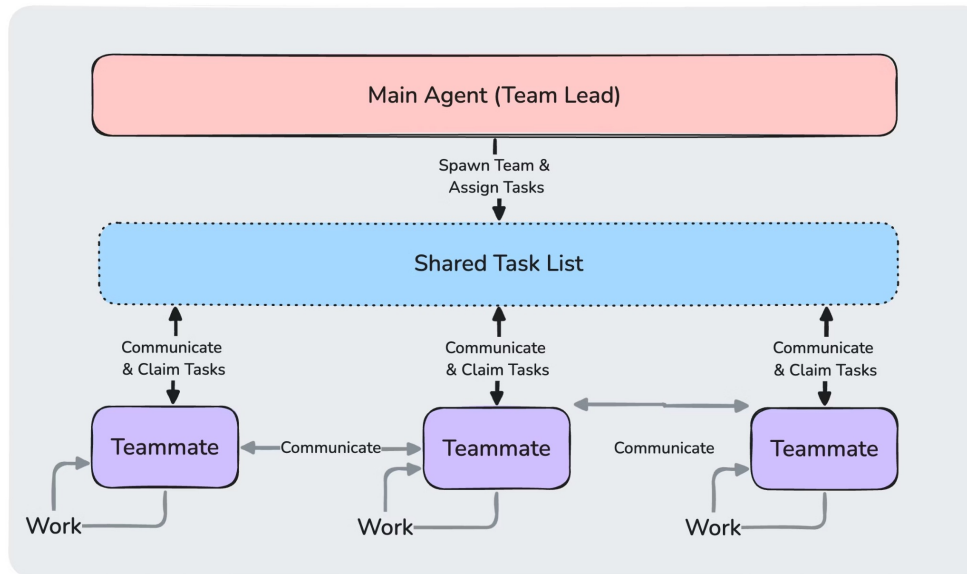
# 应对复杂任务：Subagents与Agent Team



## Subagents



## Agent Teams



	Subagents	Agent teams
<b>Context</b>	Own context window; results return to the caller	Own context window; fully independent
<b>Communication</b>	Report results back to the main agent only	Teammates message each other directly
<b>Coordination</b>	Main agent manages all work	Shared task list with self-coordination
<b>Best for</b>	Focused tasks where only the result matters	Complex work requiring discussion and collaboration
<b>Token cost</b>	Lower: results summarized back to main context	Higher: each teammate is a separate Claude instance

是否需要agent之间交流

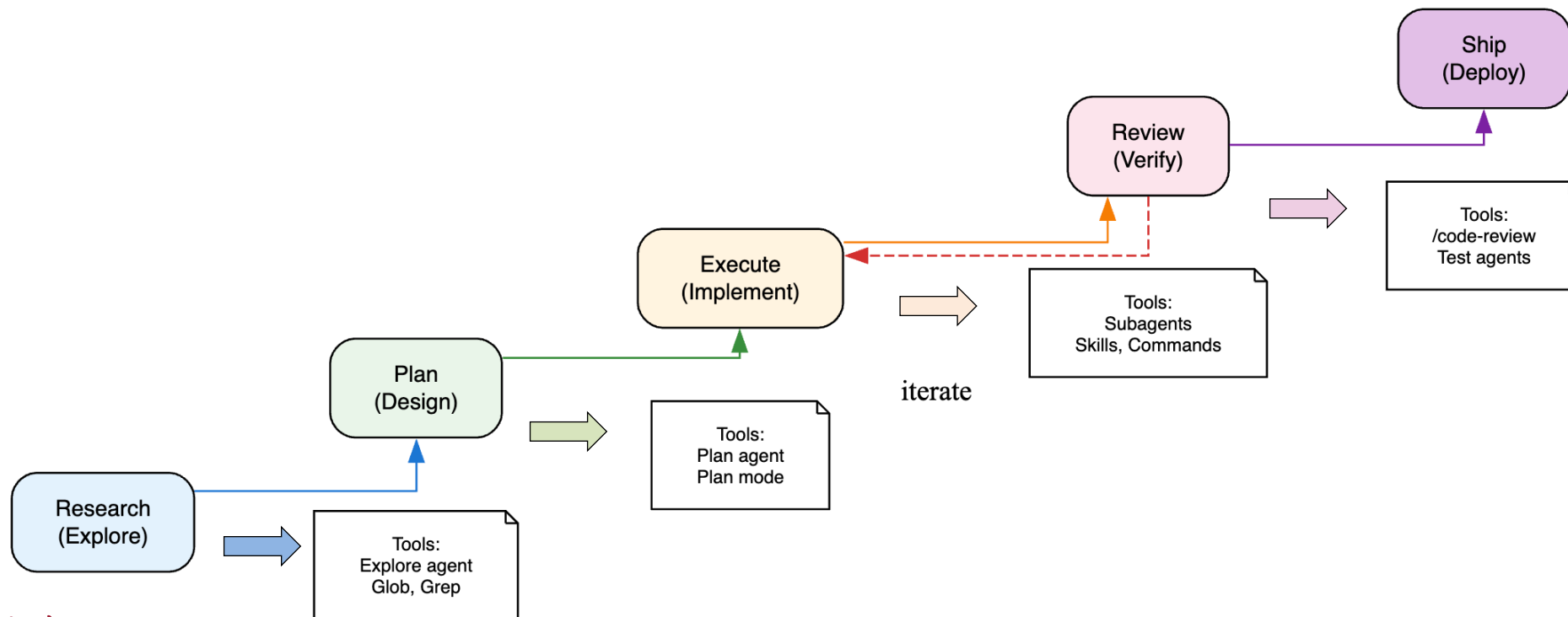
仅执行

需要讨论合作

# 工作实践：Explore、Plan、Execute、Review



- 先探索规划，再执行，同时review迭代
- 其中任意步骤之后都可以插入review，不仅是code review，也可以插入plan review
- 打磨 plan比修改bad code效率更高
- 使用markdown作为媒介，作为human-agent/agent-agent interface

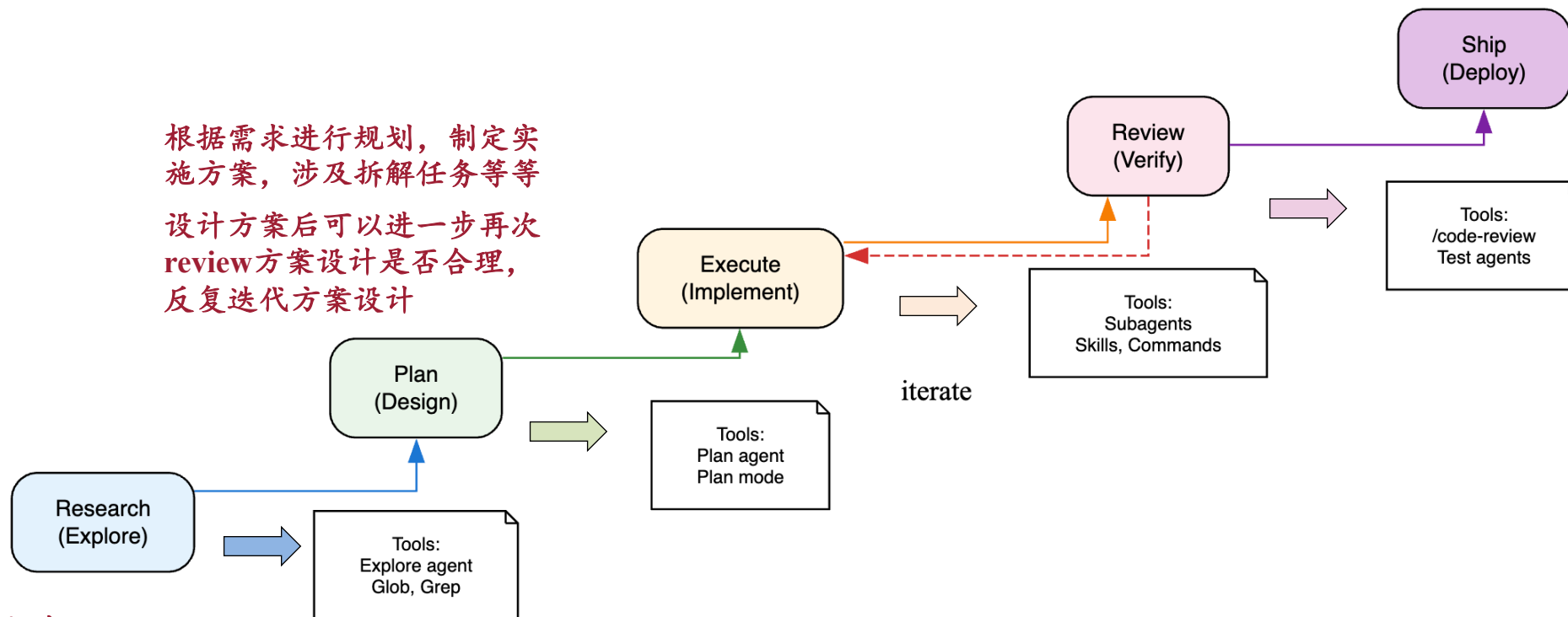


探索当前项目codebase  
充分了解背景信息以及需求

# 工作实践：Explore、Plan、Execute、Review



- 先探索规划，再执行，同时review迭代
- 其中任意步骤之后都可以插入review，不仅是code review，也可以插入plan review
- 打磨 plan比修改bad code效率更高
- 使用markdown作为媒介，作为human-agent/agent-agent interface



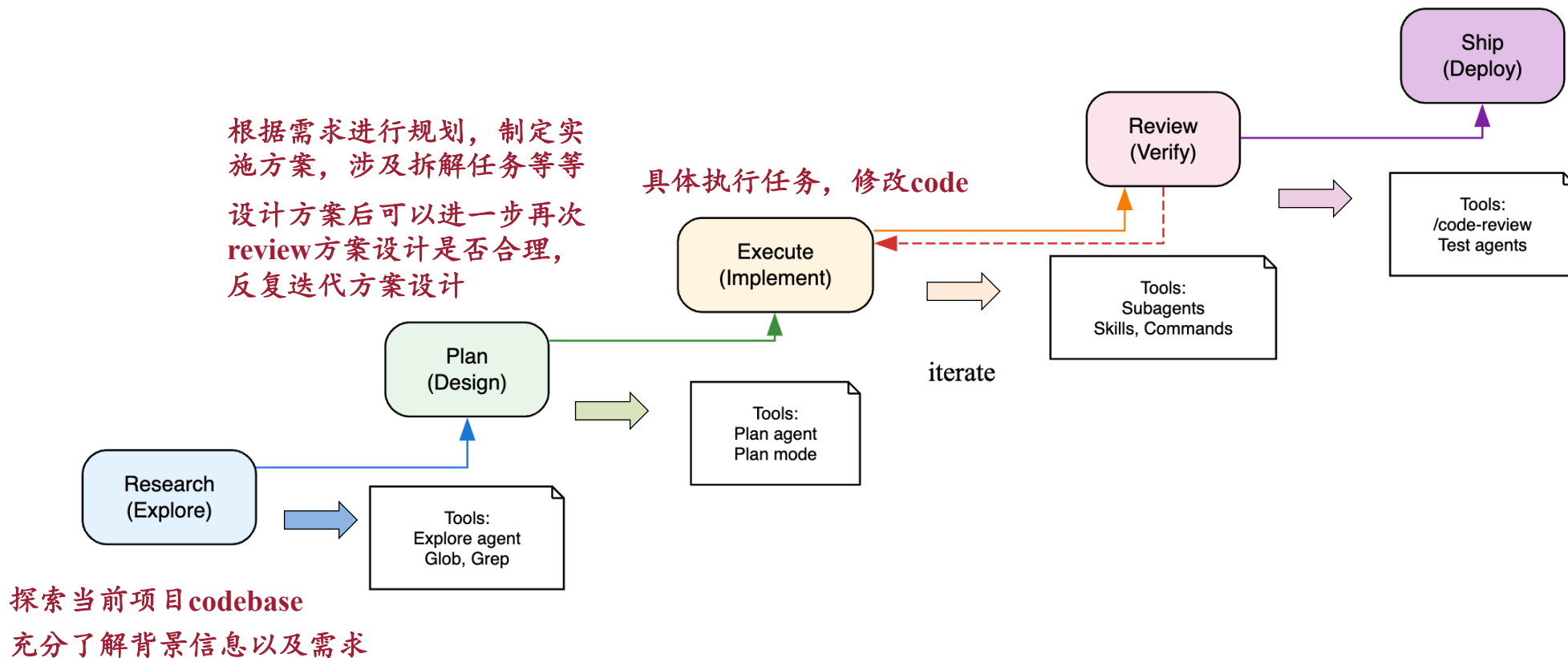
根据需求进行规划，制定实施方案，涉及拆解任务等等  
设计方案后可以进一步再次review方案设计是否合理，反复迭代方案设计

探索当前项目codebase  
充分了解背景信息以及需求

# 工作实践：Explore、Plan、Execute、Review



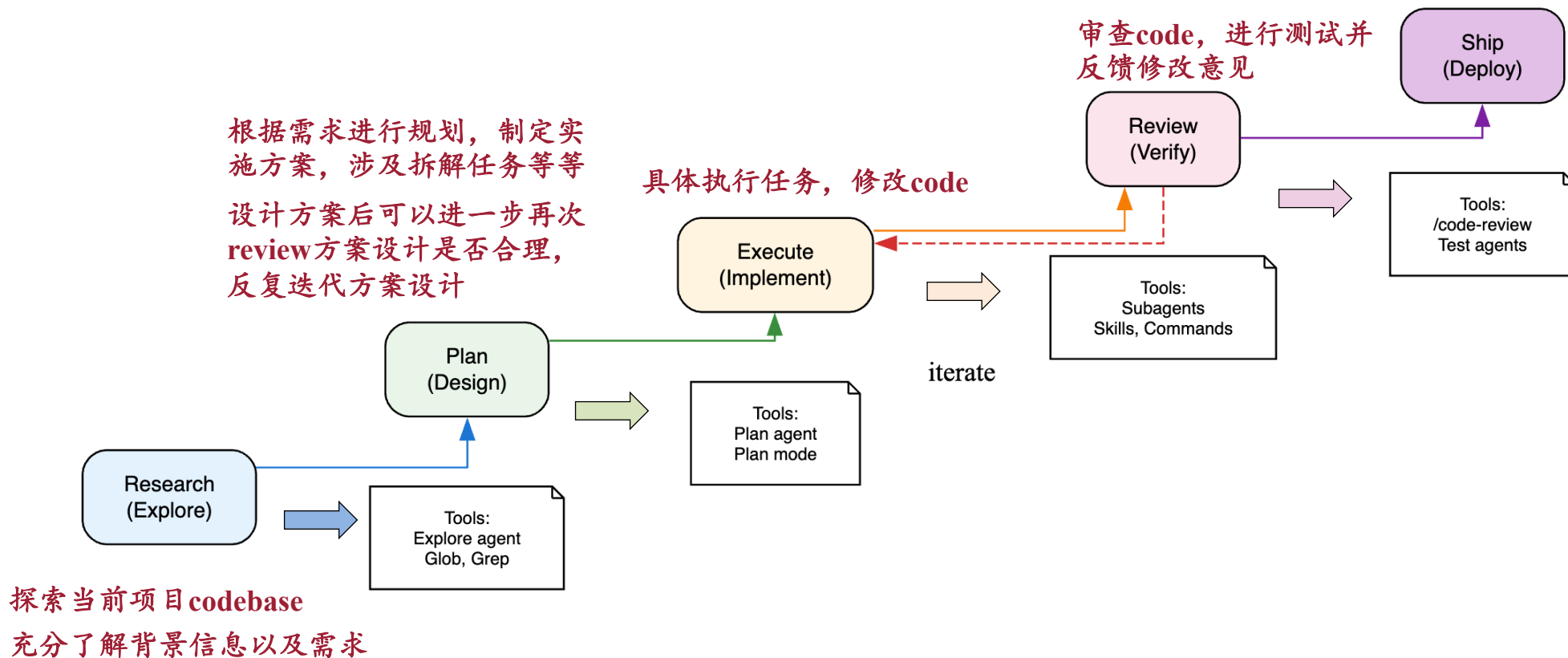
- 先探索规划，再执行，同时review迭代
- 其中任意步骤之后都可以插入review，不仅是code review，也可以插入plan review
- 打磨 plan比修改bad code效率更高
- 使用markdown作为媒介，作为human-agent/agent-agent interface



# 工作实践：Explore、Plan、Execute、Review



- 先探索规划，再执行，同时review迭代
- 其中任意步骤之后都可以插入review，不仅是code review，也可以插入plan review
- 打磨 plan比修改bad code效率更高
- 使用markdown作为媒介，作为human-agent/agent-agent interface




# 常用插件: Ralph Loop

- 自主迭代循环机制。给出一个任务和完成条件后, Agent开始执行该任务;
- 当模型在某次迭代中尝试结束时, 一个 **Stop Hook** 会拦截试图退出的动作, 并**重新注入原始任务提示**, 从而创建一个自我参照的反馈循环。
- 在这个循环中, 模型可以读取上一次迭代改动过的文件、测试结果和 git 历史, 并据此**逐步修正自己的输出直到达到完成条件或达到设定的迭代上限**。

```
Claude Code v2.1.116

Welcome back!

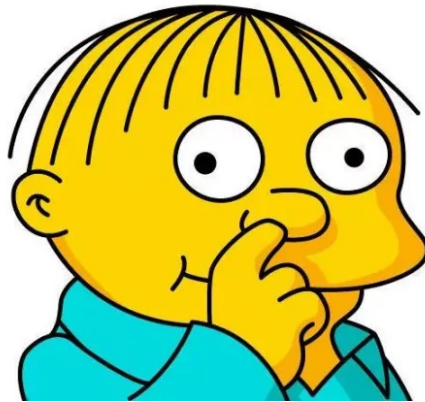


Opus 4.7 (1M context) · API Usage Billing
~/Downloads

Tips for getting started
Run /init to create a CLAUDE.md fil...

Recent activity
No recent activity

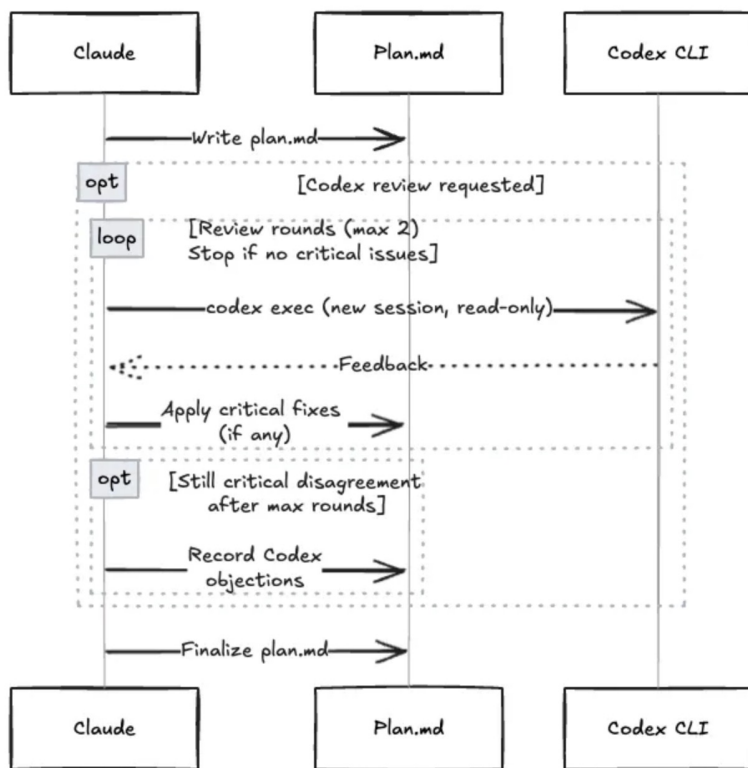
/ralph-loop "your prompt here" --max-iterations 10 --completion-promise "DONE"
```



**Ralph Wiggum**  
《辛普森一家人》

# 工作实践：多家模型协同

- Single-Model Blindness: 使用一家模型, plan、execute、review是同一个, 既当运动员又当裁判, 难以发现自身问题
- Claude Code已支持 /codex-review, 在Claude Code中调用Codex中的gpt-5.5等模型进行review
- 或者可以直接基于markdown作为媒介, 查阅批注plan, 或者review代码



```

 - Record the logprob margin (confidence) for threshold analysis
 Output results.jsonl:
 ("image_path": "...", "label_27b": "content", "pred": "content", "margin": 3.45,
 "lp_yes": -0.12, "lp_no": -3.57, "token": "yes")

 Metrics (printed + saved to metrics.json):
 - Accuracy, Precision/Recall/F1 per class
 - Confusion matrix (TP/TN/FP/FN)
 - Logprob margin distribution (mean, median, p5, p25, p75, p95)
 - Error count (images where neither yes/no found in top_logprobs)

 Config:
 - Endpoint: http://localhost:8090/v1/chat/completions
 - Concurrency: 16 (single GPU, small model)
 - Temperature: 0.0
 - Dataset: filter tags.jsonl to only content/non_content labels (~72,536 images)

 Critical Files Referenced
 File: runs/image_tag_10k/run.py
 Purpose: Async pattern, image loading, base64 encoding, resumability - reuse structure
 File: runs/image_tag_10k/output/tags.jsonl
 Purpose: 73K labeled images (ground truth from 27B)
 File: runs/image_tag_10k/start_servers.sh
 Purpose: vLLM launch flags reference
 File: /mnt/csp_sh/mvvision/share/huggingface/Qwen/Qwen3.5-0.8B/
 Purpose: Model checkpoint
 File: /mnt/csp_sh/mvvision/share/huggingface/Qwen/Qwen3.5-0.8B/chat_template.jinja
 Purpose: Thinking-token injection (lines 147-153)
 Verification
 1. Start server on Gemini pod - confirm "Application startup complete" in vllm log
 2. Run test_logprobs.py - confirm logprobs contain yes/no tokens, no thinking-token issues
 3. Run eval.py - monitor via tail -f runs/zero_shot_0.8B/stdout.log on shared filesystem
 4. Check results - review metrics.json for accuracy/F1, inspect results.jsonl for error patterns

 Claude has written up a plan and is ready to execute. Would you like to proceed?
 > 1. Yes, clear context and auto-accept edits (shift+tab)
 > 2. Yes, auto-accept edits
 > 3. Yes, manually approve edits
 > 4. Type here to tell Claude what to change

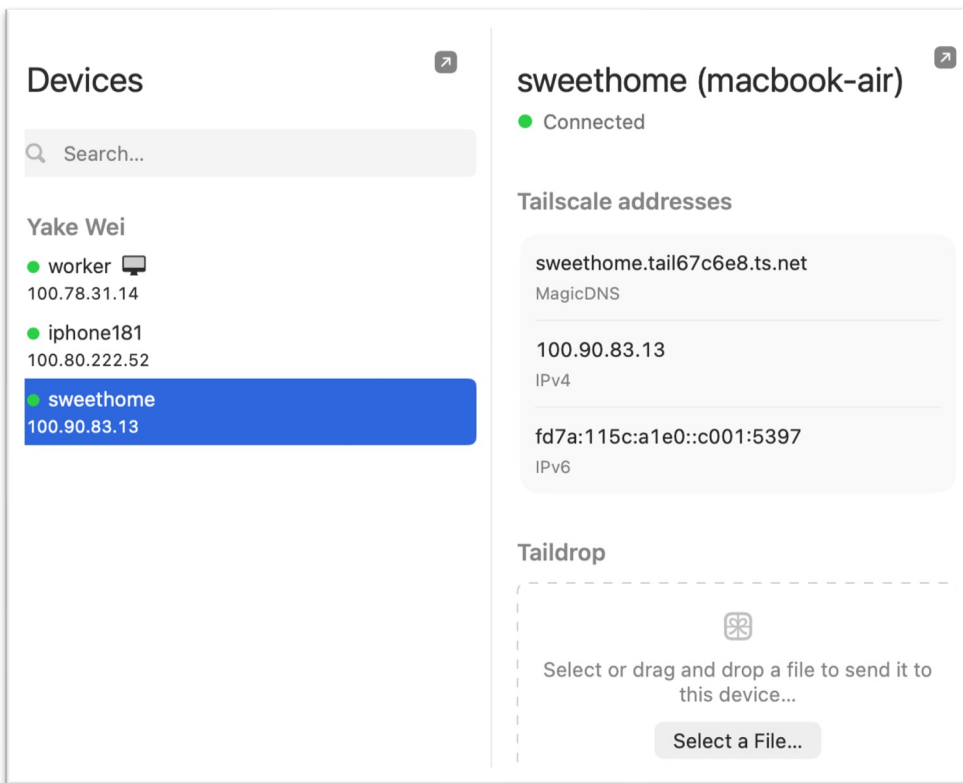
 ctrl-q to edit in VS Code ~/.claude-internal/plans/Lively-fluttering-cocke.md

```

# 多端协作: tailscale+termius

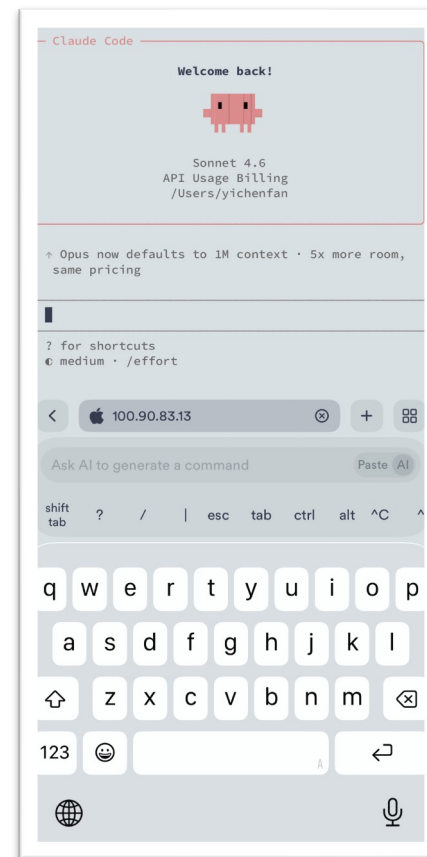


- 哪里有命令行哪里就有agent



Tailscale组网，实现网络内设备互相访问

其中一台设备保持全天候运行，用于处理定时任务或长时任务

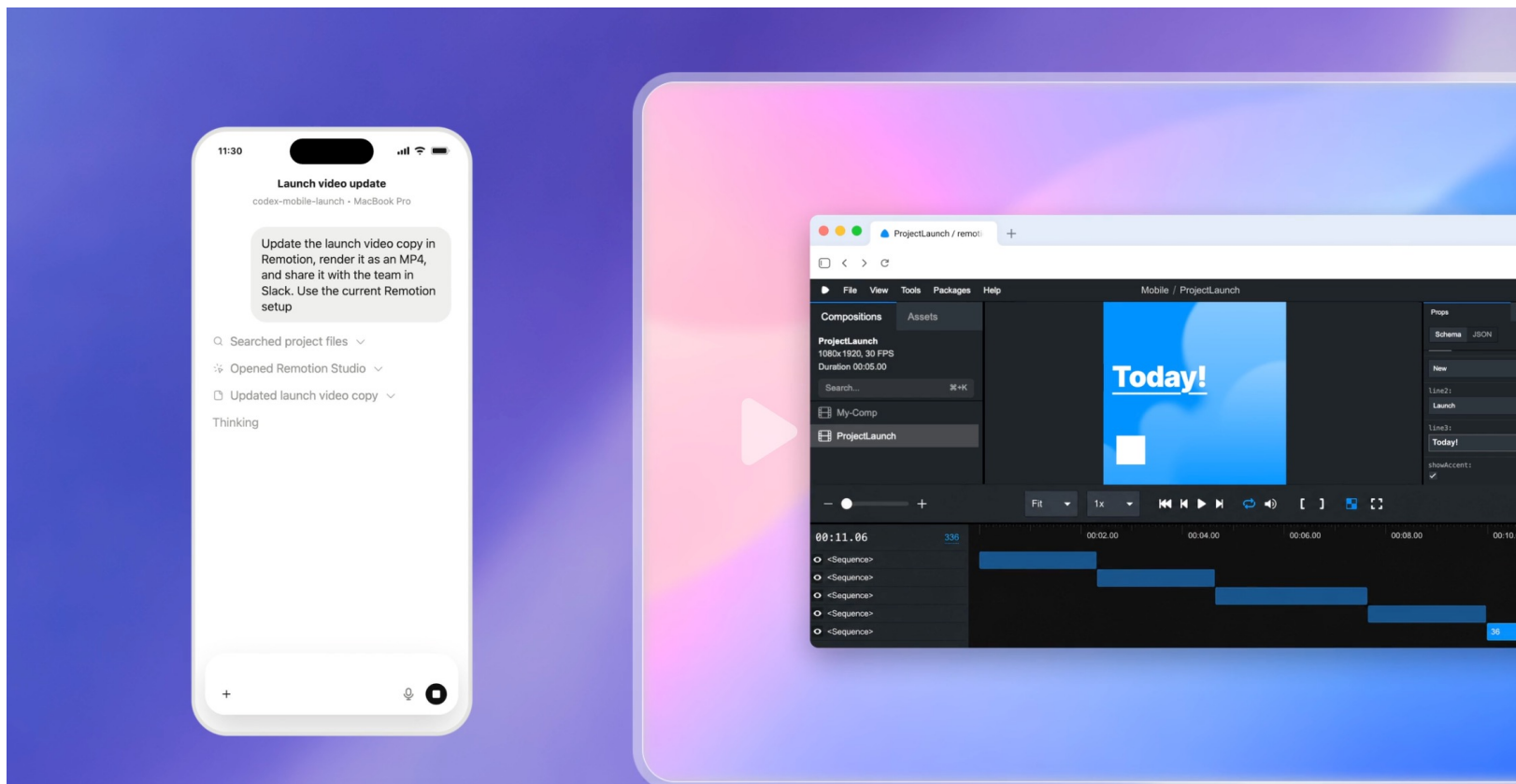


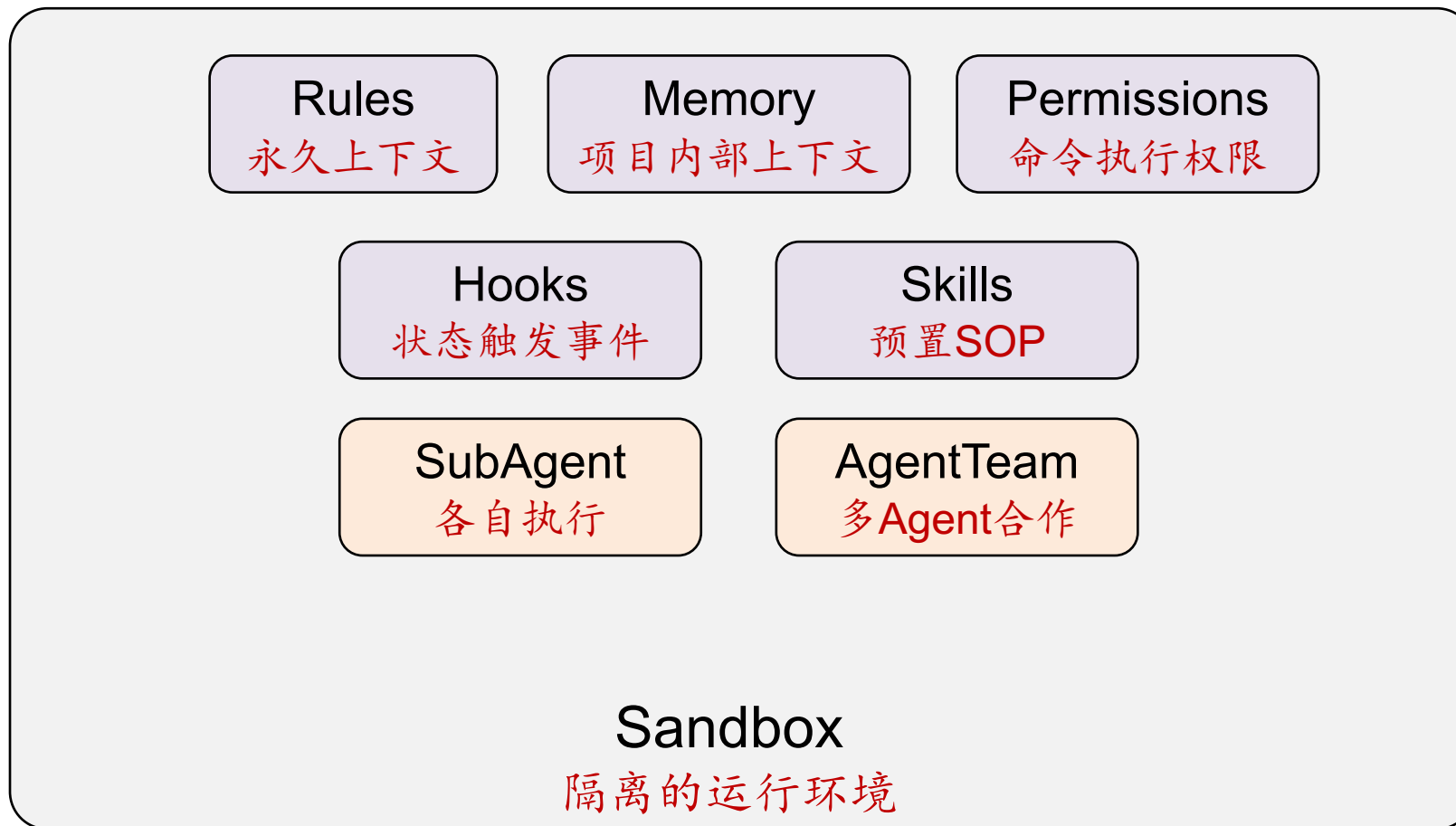
移动端同样可通过termius访问

# 多端协作: tailscale+termius



- Codex已支持多端协同 (iOS+MacOS)
- 支持通过iPhone ChatGPT APP直接访问服务器



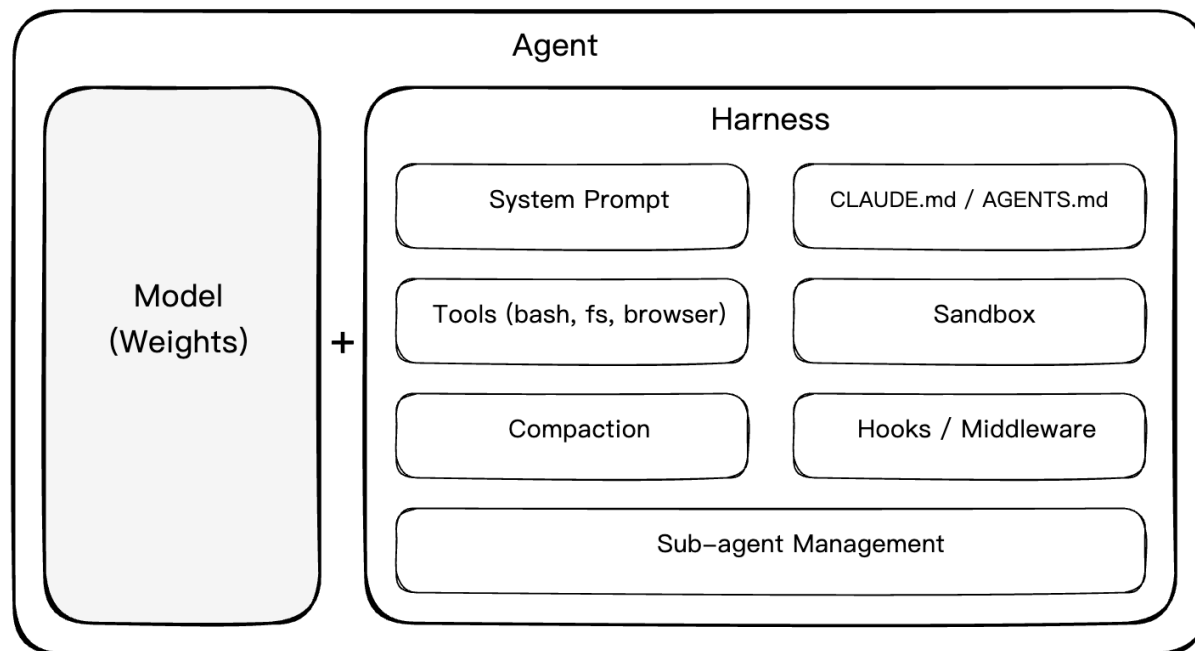


# Harness Engineering



- 源于Mitchell Hashimoto的一篇博文和OpenAI的文章
- 从日常使用技巧，走到更系统化的方法论
- Harness: 除了模型权重以外的一切

Agent = Model + Harness

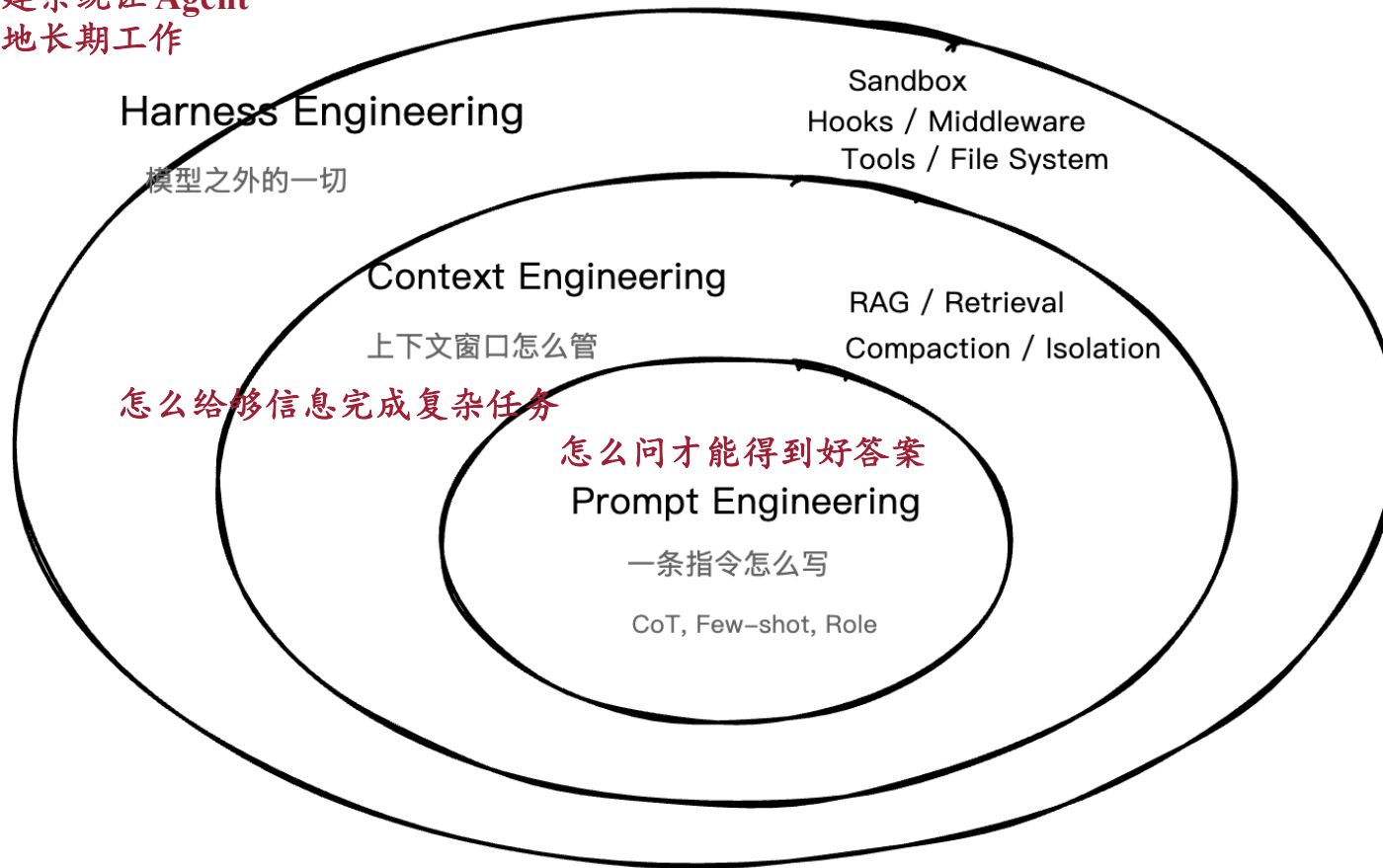


# Harness Engineering



- Prompt Engineering, Context Engineering 与 Harness Engineering

怎么建系统让 Agent  
可靠地长期工作



# 一个Harness Engineering案例



- LongChain团队的实验，仅通过Harness Engineering，在保持模型不变的情况下，在Terminal Bench 2.0上的得分从52.8提高到66.5，提高了13.7分。

terminal-bench Run Terminal-Bench Leaderboard Tasks Registry Contributors News Terminus Discord

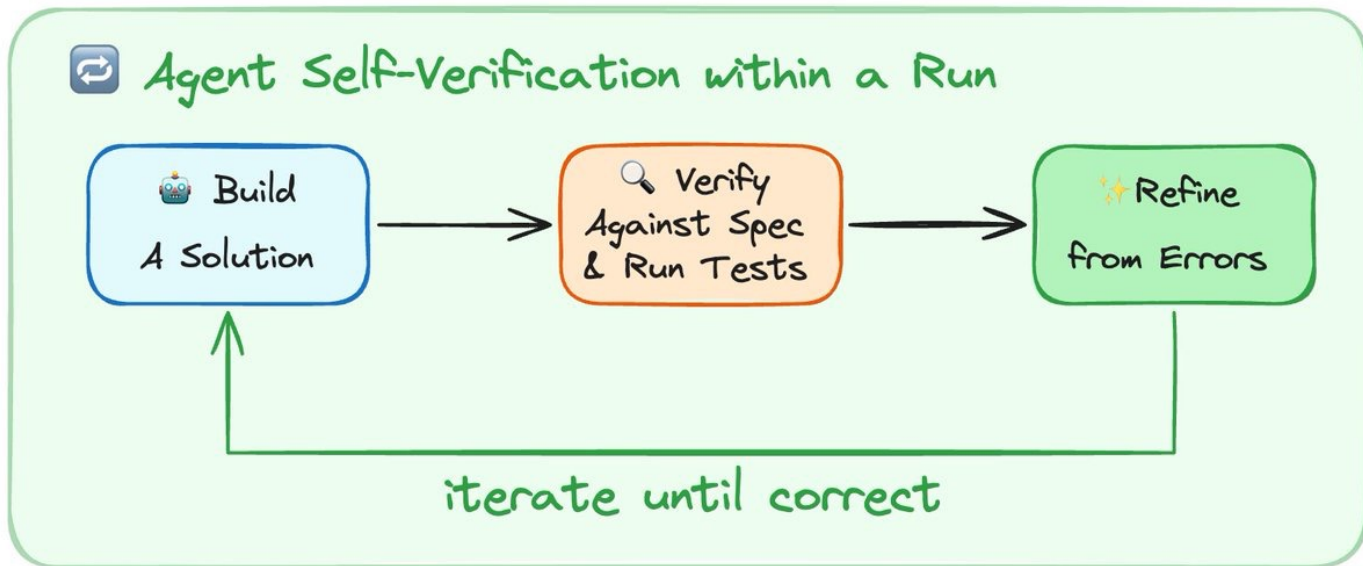
Showing 102 entries Clear filters

Rank	Agent	Model	Date	Agent Org	Model Org	Accuracy
1	Simple Codex	GPT-5.3-Codex	2026-02-06	OpenAI	OpenAI	75.1% ± 2.4
2	CodeBrain-1	GPT-5.3-Codex	2026-02-10	Feeling AI	OpenAI	70.3% ± 2.6
3	Droid	Claude Opus 4.6	2026-02-05	Factory	Anthropic	69.9% ± 2.5
4	Mux	GPT-5.3-Codex	2026-02-09	Coder	OpenAI	68.5% ± 2.4
5	Deep Agents	GPT-5.2-Codex	2026-02-12	LangChain	OpenAI	66.5% ± 3.1
6	Droid	GPT-5.2	2025-12-24	Factory	OpenAI	64.9% ± 2.8
7	Ante	Gemini 3 Pro	2026-01-06	Antigma Labs	Google	64.7% ± 2.7
8	Terminus 2	GPT-5.3-Codex	2026-02-05	Terminal Bench	OpenAI	64.7% ± 2.7
9	Junie CLI	Gemini 3 Flash	2025-12-23	JetBrains	Google	64.3% ± 2.8
10	Droid	Claude Opus 4.5	2025-12-11	Factory	Anthropic	63.1% ± 2.7

# 一个Harness Engineering案例



- 修改 1: 退出前强制自检, 用到了hook
  - 在 Agent 试图退出时拦截, 通过hook强制注入一个 checklist 让它对照任务说明验证。



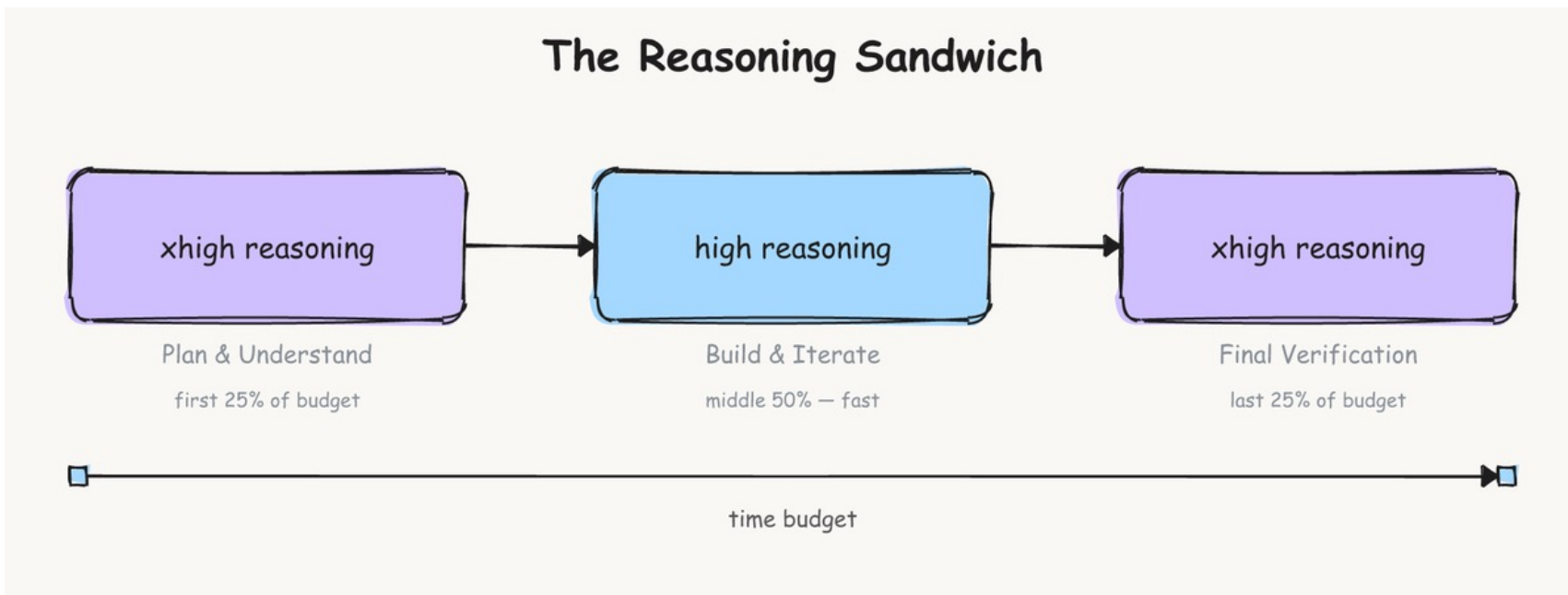
- 修改 2: 启动时注入环境信息
  - Agent 在陌生环境中会浪费大量时间探索目录结构、找Python环境。他们在Agent启动时自动跑一些bash命令扫描环境, 把结果注入上下文。
- 修改 3: 死循环检测
  - Agent有时候会在同一个文件上反复做小修改, 10+次还在原地打转。他们通过hook追踪每个文件的编辑次数, 超过阈值就注入"考虑换个方案"的提示。

# 一个Harness Engineering案例



## • 修改 4: Reasoning Sandwich

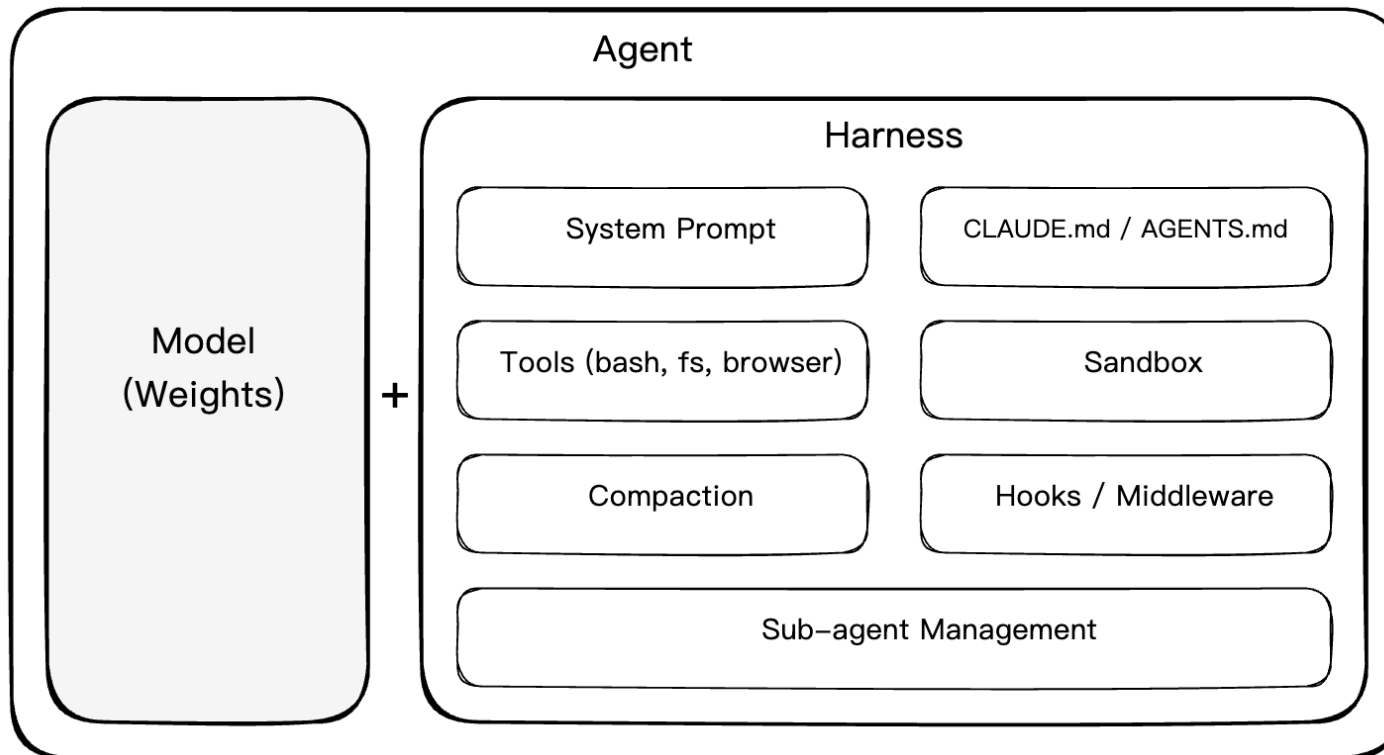
- GPT-5.2-Codex 有 4 档推理强度 (low/medium/high/xhigh)，全程 xhigh 反而得分低 (53.9%)，因为耗时过久。最终他们用 xhigh → high → xhigh 的“三明治”策略——规划和验证用高推理，执行阶段用中等推理。



# 从何处入手Harness Engineering



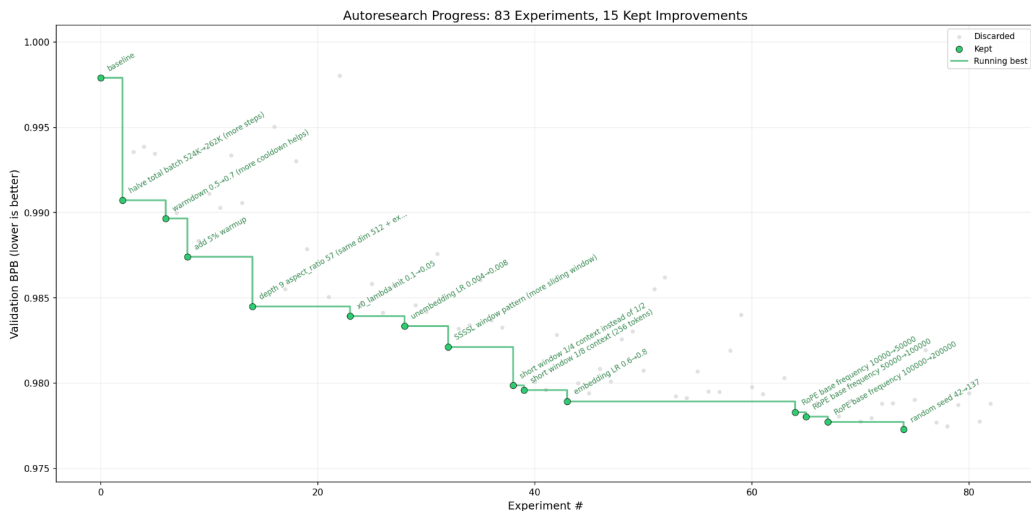
Agent = Model + Harness



# Vibe Research

- 开放心态：AI承担了科研中的dirty work，并加速了idea验证的过程（学习codebase、源码修改、实验、写作润色），每个人都有了自已“实习生”。
- 个人认为比较可行的loop：AI deep research + 人类思想 + AI coding + AI analysis + 人类监督并纠偏

## autoresearch



Andrej Karpathy的Autoresearch工具

FARS: Fully Automated Research System

[X Follow](#)
[YouTube](#)

# FARS Live Deployment Completed

**166** Papers

**417** Hours

**21,600,000,000** Tokens

**\$186,000** Total Cost

Analemma公司的实践：417 hours 产出166 papers



# Thanks for Listening!

卫雅珂

yakewei@ruc.edu.cn

参考来源:

- <https://code.claude.com/docs>
- <https://pyshine.com/Claude-Code-Best-Practices>
- <https://martinfowler.com/articles/harness-engineering.html>
- <https://www.philschmid.de/agent-harness-2026>
- <https://yuanchaofa.com/post/harness-engineering-for-ai-agents>
- <https://x.com/Vtrivedy10/status/2023805578561060992?s=20&ref=blog.langchain.com>