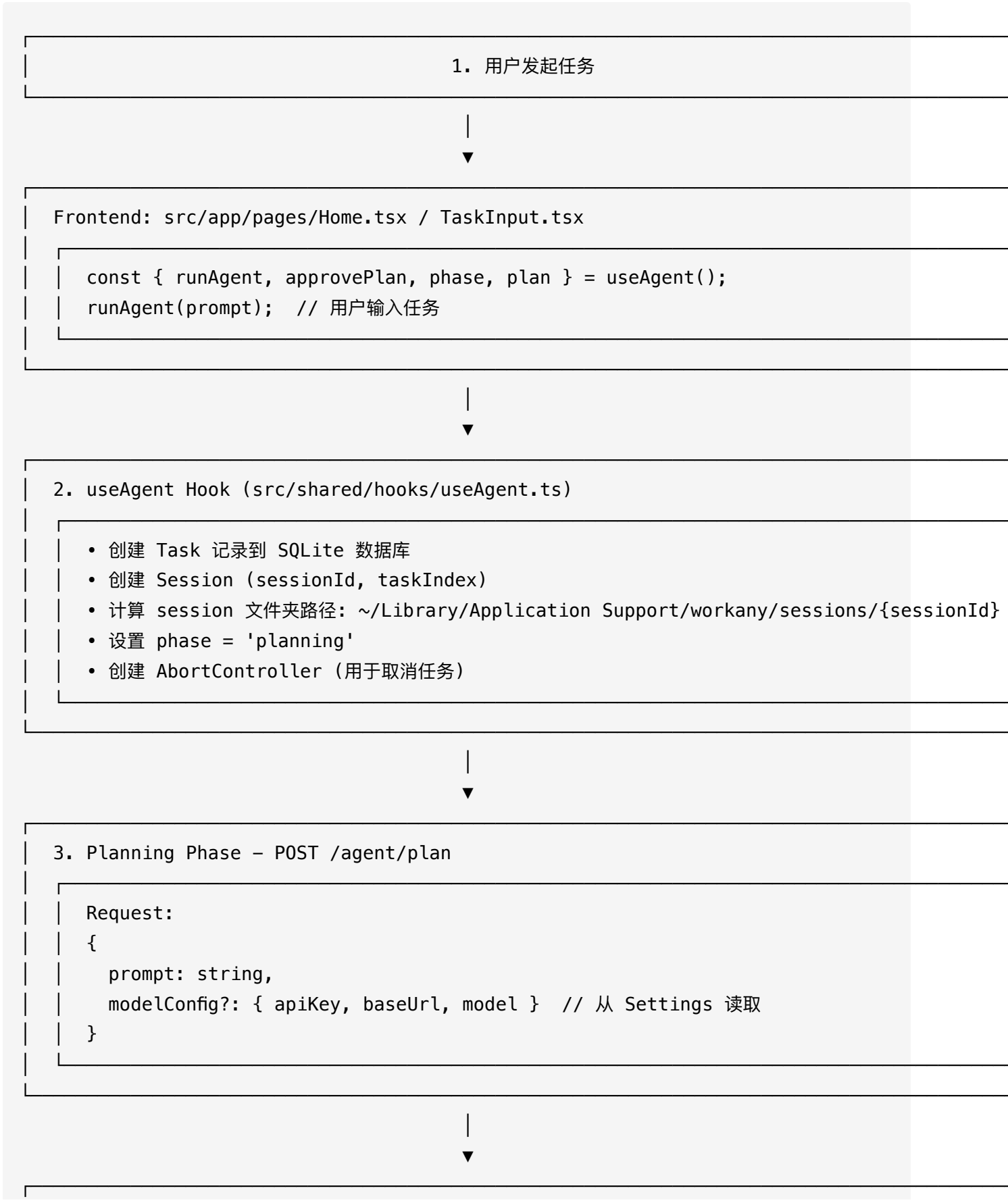
WorkAny 任务执行流程图

完整调用链路 ：从用户输入到获得结果

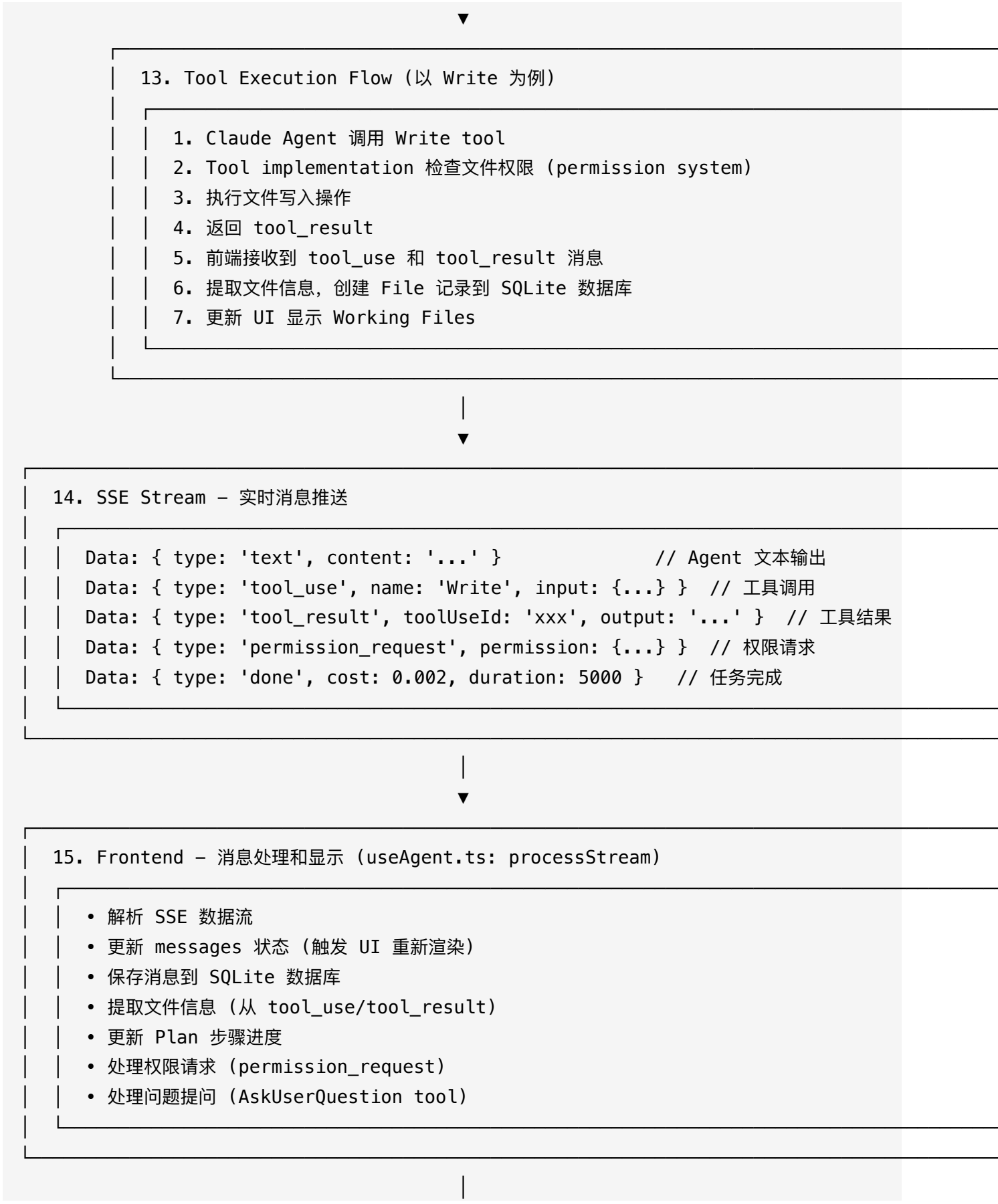

1. 用户发起任务
Frontend: src/app/pages/Home.tsx / TaskInput.tsx
const { runAgent, approvePlan, phase, plan } = useAgent();
runAgent(prompt); // 用户输入任务
2. useAgent Hook (src/shared/hooks/useAgent.ts)
• 创建 Task 记录到 SQLite 数据库
• 创建 Session (sessionId, taskIndex)
• 计算 session 文件夹路径: ~/Library/Application Support/workany/sessions/{sessionId}
• 设置 phase = 'planning'
• 创建 AbortController (用于取消任务)
3. Planning Phase - POST /agent/plan
Request:
{
prompt: string,
modelConfig?: { apiKey, baseUrl, model } // 从 Settings 读取
}

```
│  4. API Agent Routes (src-api/src/app/api/agent.ts)
│
│  ┌──────────────────────────────────────────────────────────────
│  │  agent.post('/plan', async (c) => {
│  │    const session = createSession('plan');     // 创建后端 session
│  │    const readable = createSSEStream(
│  │      runPlanningPhase(body.prompt, session, body.modelConfig)
│  │    );
│  │    return new Response(readable, { headers: SSE_HEADERS });
│  │  });
│  └──────────────────────────────────────────────────────────────
│
└─────────────────────────────────────────────────────────────────
                                 │
                                 ▼
┌─────────────────────────────────────────────────────────────────
│  5. Agent Service (src-api/src/shared/services/agent.ts)
│
│  ┌──────────────────────────────────────────────────────────────
│  │  export async function* runPlanningPhase(
│  │    prompt, session, modelConfig
│  │  ): AsyncGenerator<AgentMessage> {
│  │    const agent = getAgent(modelConfig);  // 获取或创建 Agent 实例
│  │    for await (const message of agent.plan(prompt, { ... })) {
│  │      if (message.type === 'plan' && message.plan) {
│  │        savePlan(message.plan);  // 保存 plan 到全局存储
│  │      }
│  │      yield message;  // SSE 流式输出
│  │    }
│  │  }
│  └──────────────────────────────────────────────────────────────
│
└─────────────────────────────────────────────────────────────────
                                 │
                                 ▼
┌─────────────────────────────────────────────────────────────────
│  6. Claude Agent Implementation (src-api/src/extensions/agent/claude/index.ts)
│
│  ┌──────────────────────────────────────────────────────────────
│  │  async* plan(prompt, options): AsyncGenerator<AgentMessage> {
│  │    // 使用 Claude Agent SDK 创建 planning agent
│  │    const agent = await anthropic.agents.create({
│  │      model: this.model,
│  │      tools: [...],  // 注册工具
│  │      instructions: PLANNING_INSTRUCTIONS  // planning 提示词
```

```
│  │    });
│  │
│  │    const result = await agent.plan({ prompt });
│  │    // 返回 plan 消息
│  │    yield { type: 'plan', plan: result.plan };
│  │  }
│  └──────────────────────────────────────────────────────────────────────
└─────────────────────────────────────────────────────────────────────────
                                    │
                                    ▼
┌─────────────────────────────────────────────────────────────────────────
│  7. SSE Stream Response (前端接收)
│
│  ┌──────────────────────────────────────────────────────────────────────
│  │  Data: { type: 'session', sessionId: 'xxx' }      // Session ID
│  │  Data: { type: 'plan', plan: { id, goal, steps } } // Plan 对象
│  │  Data: { type: 'done' }                            // 完成
│  └──────────────────────────────────────────────────────────────────────
└─────────────────────────────────────────────────────────────────────────
                                    │
                                    ▼
┌─────────────────────────────────────────────────────────────────────────
│  8. 用户审批 Plan (src/components/task/PlanApproval.tsx)
│
│  ┌──────────────────────────────────────────────────────────────────────
│  │  • 显示 Plan 详情: goal, steps
│  │  • 用户选择: approvePlan() 或 rejectPlan()
│  └──────────────────────────────────────────────────────────────────────
└─────────────────────────────────────────────────────────────────────────
                                    │
                    ┌───────────────┴───────────────┐
                    │                               │
                 Reject                          Approve
                    │                               │
                    ▼                               ▼
            ┌───────────┐                  ┌──────────────────────────────
            │  结束任务  │                 │  9. Execution Phase - POST /agent/execute
            │  phase=idle │                │
            └───────────┘                  │  ┌───────────────────────────
                                           │  │  Request:
                                              │  │  {
                                              │  │    planId: string,
                                              │  │    prompt: string,       // 原始 prom
```

```
                                                    │   │    workDir: string,      // session
                                                    │   │    taskId: string,
                                                    │   │    sandboxConfig?: { enabled, provide
                                                    │   │    skillsPath?: string  // 自定义 ski
                                                    │   │  }
                                                    │   └──────────────────────────────────────
                                                    └──────────────────────────────────────────
                                                                          │
                                                                          ▼
┌─────────────────────────────────────────────────────────────────────────────────────────────────
│  10. Agent Service - runExecutionPhase
│   ┌─────────────────────────────────────────────────────────────────────────────────────────────
│   │  export async function* runExecutionPhase(
│   │    planId, session, originalPrompt, workDir, taskId, sandboxConfig, skillsPath
│   │  ): AsyncGenerator<AgentMessage> {
│   │    const agent = getAgent(modelConfig);
│   │    const plan = getPlan(planId);  // 从全局存储获取 plan
│   │
│   │    for await (const message of agent.execute({
│   │      planId, plan, originalPrompt, sessionId, cwd: workDir, taskId,
│   │      abortController: session.abortController,
│   │      sandbox: sandboxConfig,  // 沙箱配置
│   │      skillsPath                  // skills 路径
│   │    })) {
│   │      yield message;  // SSE 流式输出
│   │    }
│   │  }
│   └─────────────────────────────────────────────────────────────────────────────────────────────
└─────────────────────────────────────────────────────────────────────────────────────────────────
                                                                          │
                                                                          ▼
┌─────────────────────────────────────────────────────────────────────────────────────────────────
│  11. Claude Agent - execute()
│   ┌─────────────────────────────────────────────────────────────────────────────────────────────
│   │  async* execute(options): AsyncGenerator<AgentMessage> {
│   │    // 创建 execution agent (带工具和 sandbox)
│   │    const agent = await anthropic.agents.create({
│   │      model: this.model,
│   │      tools: [
│   │        builtinTools,       // Read, Write, Edit, Bash, WebFetch, etc.
```

```
│   │        sandboxTools,       // Sandbox 工具
│   │        mcpTools,           // MCP 工具
│   │        skillTools          // Skills 工具
│   │      ],
│   │      instructions: EXECUTION_INSTRUCTIONS
│   │    });
│   │
│   │    // 执行 plan 的每个步骤
│   │    const result = await agent.run({
│   │      prompt: originalPrompt,
│   │      plan: options.plan
│   │    });
│   │
│   │    // 流式输出消息
│   │    for await (const event of result) {
│   │      if (event.type === 'text') yield { type: 'text', content: event.text };
│   │      if (event.type === 'tool_use') yield { type: 'tool_use', ... };
│   │      if (event.type === 'tool_result') yield { type: 'tool_result', ... };
│   │    }
│   │  }
│   └──────────────────────────────────────────────────────────────────────────────
│
└─────────────────────────────────────────────────────────────────────────────────
                                                                       │
                    ┌────────────────────────────┼────────────────────────────┐
                    │                            │                            │
                    ▼                            ▼                            ▼
            ┌───────────────────┐        ┌───────────────────┐        ┌───────────────────┐
            │  12a. Builtin Tools│        │  12b. Sandbox Tools│        │  12c. MCP/Skills  │
            │  ┌─────────────┐   │        │  ┌─────────────┐   │        │  ┌─────────────┐  │
            │  │ Read        │   │        │  │ Sandbox.exec│   │        │  │ MCP servers │  │
            │  │ Write       │   │        │  │ Sandbox.run │   │        │  │  Skills     │  │
            │  │ Edit        │   │        │  │ Script      │   │        │  └─────────────┘  │
            │  │ Bash        │   │        │  └─────────────┘   │        └───────────────────┘
            │  │ WebFetch    │   │        │                    │
            │  │ WebSearch   │   │        │  Sandbox Provider: │
            │  │ AskUserQuestion│ │        │  - codex (VM)      │
            │  └─────────────┘   │        │  - claude (Container)│
            └───────────────────┘        │  - native (process) │
                                         └────────────────────┘
                                             │
```


13. Tool Execution Flow (以 Write 为例)
1. Claude Agent 调用 Write tool
2. Tool implementation 检查文件权限 (permission system)
3. 执行文件写入操作
4. 返回 tool_result
5. 前端接收到 tool_use 和 tool_result 消息
6. 提取文件信息，创建 File 记录到 SQLite 数据库
7. 更新 UI 显示 Working Files
14. SSE Stream - 实时消息推送
Data: { type: 'text', content: '...' } // Agent 文本输出
Data: { type: 'tool_use', name: 'Write', input: {...} } // 工具调用
Data: { type: 'tool_result', toolUseId: 'xxx', output: '...' } // 工具结果
Data: { type: 'permission_request', permission: {...} } // 权限请求
Data: { type: 'done', cost: 0.002, duration: 5000 } // 任务完成
15. Frontend - 消息处理和显示 (useAgent.ts: processStream)
• 解析 SSE 数据流
• 更新 messages 状态 (触发 UI 重新渲染)
• 保存消息到 SQLite 数据库
• 提取文件信息 (从 tool_use/tool_result)
• 更新 Plan 步骤进度
• 处理权限请求 (permission_request)
• 处理问题提问 (AskUserQuestion tool)

```
                                        ▼
┌──────────────────────────────────────────────────────────────────────
│  16. UI 渲染 (src/components/home/AgentMessages.tsx, ToolExecutionItem.tsx)
│
│  ┌───────────────────────────────────────────────────────────────────
│  │  • 显示对话消息 (text, user)
│  │  • 显示工具调用 (tool_use) with 展开/折叠
│  │  • 显示工具结果 (tool_result)
│  │  • 显示 Plan 进度 (步骤状态)
│  │  • 显示权限请求对话框
│  │  • 显示问题提问对话框
│  └───────────────────────────────────────────────────────────────────
└──────────────────────────────────────────────────────────────────────
                                        │
                                        ▼
┌──────────────────────────────────────────────────────────────────────
│  17. 任务完成
│
│  ┌───────────────────────────────────────────────────────────────────
│  │  • 接收 { type: 'done' } 消息
│  │  • 更新 Task 状态 = 'completed'
│  │  • 设置 phase = 'idle'
│  │  • 标记所有 Plan 步骤 = 'completed'
│  │  • 停止加载动画
│  └───────────────────────────────────────────────────────────────────
└──────────────────────────────────────────────────────────────────────
```

直接执行模式（带图片附件）

当用户上传图片时，跳过 Planning Phase，直接执行：

```
runAgent(prompt, attachments=[{ type: 'image', data: 'base64...' }])
    │
    ▼
POST /agent/ (直接执行，不经过 /plan)
    │
    ▼
runAgent(prompt, images=[...])  // 使用 Claude Agent SDK 的 run() 方法
    │
    ▼
SSE Stream: text → tool_use → tool_result → ... → done
```

会话继续

用户在任务完成后继续对话：

```
continueConversation(reply, attachments?)
    │
    ▼
构建 conversation history (从已有 messages)
    │
    ▼
POST /agent/ (带 conversation 参数)
    │
    ▼
runAgent(prompt, conversation=[...], images=[...])
```

关键数据结构

Agent Message (SSE)

```
{
  type: 'text' | 'tool_use' | 'tool_result' | 'session' | 'done' | 'error' | 'plan' | 'pe
  sessionId?: string,
  content?: string,
  name?: string,           // tool_use: 工具名称
  id?: string,            // tool_use: 工具调用 ID
  input?: object,         // tool_use: 工具输入参数
  toolUseId?: string,     // tool_result: 关联的 tool_use ID
  output?: string,        // tool_result: 工具输出
  plan?: TaskPlan,        // plan: 计划对象
  permission?: {...},     // permission_request: 权限请求
  message?: string,       // error: 错误信息
  cost?: number,          // done: API 调用成本
  duration?: number       // done: 执行时长 (ms)
}
```

Task Plan

```
{
  id: string,
  goal: string,
  steps: [
    { id: string, description: string, status: 'pending' | 'in_progress' | 'completed' |
  ],
  notes?: string,
  createdAt: Date
}
```

工具系统

Builtin Tools (默认可用)

- **Read**: 读取文件内容
- **Write**: 写入文件（创建新文件）
- **Edit**: 编辑文件（字符串替换）
- **Bash**: 执行 shell 命令
- **WebFetch**: 获取网页内容
- **WebSearch**: 网络搜索
- **AskUserQuestion**: 向用户提问
- **TodoWrite**: 任务列表管理

Sandbox Tools (隔离执行)

- **Sandbox.exec**: 在沙箱中执行命令
- **Sandbox.run/file**: 在沙箱中运行脚本文件

MCP Tools (扩展能力)

- 从 `~/.workany/mcp.json` 加载 MCP 服务器
- 动态注册 MCP 提供的工具

Skills (自定义能力)

- 从 `~/.workany/skills/` 和 `~/.claude/skills/` 加载
- 作为自定义工具注册到 Agent

沙箱提供者 (Sandbox Providers)

Provider	Isolation	Network	Use Case
`codex`	VM	❌ Blocked	高隔离需求，不可信代码
`claude`	Container	Varies	Claude Code 集成

Provider	Isolation	Network	Use Case
`native`	Process	✅ Allowed	可信代码 ’本地操作

权限系统

某些工具操作需要用户确认 ：

```
Tool Use (e.g., Bash with risky command)
    │
    ▼
Permission Request → { type: 'permission_request', permission: { id, tool, description, ri
    │
    ▼
Frontend: 显示权限对话框
    │
    ├─ Approve → 继续执行工具
    └─ Deny → 取消操作，返回错误
```

后台任务

用户切换到其他任务时 ’当前任务移至后台继续执行 ：

```
用户点击 Library 中的其他任务
    │
    ▼
loadTask(newTaskId)
    │
    ├─ abortControllerRef.current != null
    ├─ addBackgroundTask({ taskId, sessionId, abortController, isRunning })
    ├─ 清除 UI 状态（不清除 refs）
    └─ 新任务开始，后台任务继续运行并保存到数据库
```

关键文件路径

组件	路径
前端 Hook	`src/shared/hooks/useAgent.ts`
API Routes	`src-api/src/app/api/agent.ts`
Agent Service	`src-api/src/shared/services/agent.ts`
Claude Agent	`src-api/src/extensions/agent/claude/index.ts`
Sandbox API	`src-api/src/app/api/sandbox.ts`
Frontend UI	`src/components/home/AgentMessages.tsx`
Plan Approval	`src/components/task/PlanApproval.tsx`

数据存储

数据	存储位置
Tasks	SQLite: `tasks` 表
Messages	SQLite: `messages` 表
Files	SQLite: `files` 表
Plan	内存: `Map<string, TaskPlan>` (临时)
Session	内存: `Map<string, Session>` (临时)
附件文件	`~/sessions/{sessionId}/attachments/`
工作文件	`~/sessions/{sessionId}/`
日志	`~/.workany/logs/workany.log`