Arquitectura de Claude Code

Stack Técnico

Componentes Core

1. Context Manager

- Rastrea todos los archivos y operaciones
- Maneja context window de 200K tokens
- Summarization automática
- **Por eso**: Sesiones largas sin perder contexto

2. Permission System

- Sandbox por defecto
- Cada tool requiere permiso
- Configurable por path

- **Modos**: default, acceptEdits, plan, dontAsk, bypassPermissions

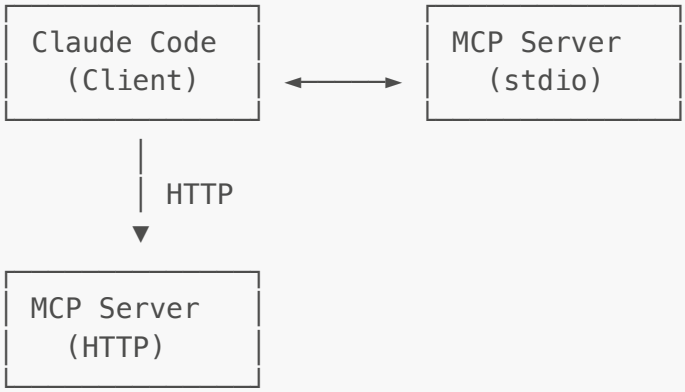
3. Tool Executor

- ~15 tools nativos: Read, Write, Edit, Bash, Grep, Glob
- Function calling de Claude API
- MCP tools se agregan dinámicamente
- **Resultado**: Claude puede "hacer cosas" en tu sistema

4. Git Integration

- Detección automática de repos
- Safety checks en comandos
- Pre-commit hooks integrados
- Co-authored commits

MCP Protocol



Stdio Servers: Procesos locales (stdin/stdout)

- Ejemplo: `@bytebase/dbhub` para PostgreSQL

HTTP Servers: APIs remotas con endpoints MCP

- Ejemplo: `https://api.githubcopilot.com/mcp/`

Cada server expone:

- Tools (funciones que Claude puede llamar)
- Resources (datos que Claude puede leer)
- Prompts (templates reutilizables)

Sistema de Agentes

```
Main Agent (tu sesión)
    │
    ├─→ Spawn Explore Agent (Haiku, foreground)
    │       └─→ Busca archivos, retorna lista
    │
    ├─→ Spawn Plan Agent (background)
    │       └─→ Investiga approach, retorna plan
    │
    └─→ Main agent ejecuta basándose en resultados
```

Built-in agents:

- Explore (Haiku) - búsquedas rápidas
- Plan - research y planning
- General - tareas multi-paso

Custom agents:

- Archivos .md en `.claude/agents/`
- Frontmatter define: name, description, tools, model, permissionMode
- Se auto-invocan cuando task matches description

Estructura de Archivos

```
~/.claude/                    # User-level config
├── settings.json             # Global settings
├── agents/                   # Personal agents
├── skills/                   # Personal skills
└── mcp/                      # MCP server configs

.claude/                      # Project-level config
├── settings.json             # Project permissions
├── agents/                   # Team agents (version control)
├── skills/                   # Team skills (version control)
└── mcp.json                  # Project MCP servers

CLAUDE.md                     # Project context (auto-loaded)
```

Jerarquía: Project > User > Defaults

Version control:

- ✅ Commitear: `.claude/` (team configs)
- ❌ NO commitear: `~/.claude/` (personal configs)

Por Qué Importa Esta Arquitectura

Para Debugging

```
❌ "Claude no funciona"
✅ "MCP server no responde - verificar logs en ~/.claude/mcp/"
✅ "Permission denied - ajustar settings.json path rules"
✅ "Context lleno - usar subagent para aislar operación"
```

Para Extender

```
✅ Entiendes cómo MCP agrega tools dinámicamente
✅ Sabes cómo agents se comunican (filesystem)
✅ Puedes diseñar permission strategies inteligentes
✅ Conoces límites del context manager
```

Para Optimizar

```
✅ Usar Explore agent (Haiku) para búsquedas rápidas = más barato
✅ Background agents para long-running tasks = no bloquea
✅ Path-specific permissions = menos interrupciones
✅ MCP resources vs tools = cuándo cachear datos
```

Preguntas Frecuentes

Q: ¿Es open source? A: No, Claude Code es closed-source. Pero la arquitectura usa standards (TypeScript, Node, MCP es open spec).

Q: ¿Dónde corre Claude (el modelo)? A: En la nube de Anthropic. Claude Code es el client que hace requests a la Claude API.

Q: ¿Cómo aseguro que no envía datos sensibles? A: Permission system + .gitignore configs + deny rules en settings.json. Enterprise: self-host en AWS Bedrock/Google Vertex.

Q: ¿MCP servers corren local o remoto? A: Ambos. Stdio = local process. HTTP = remote API. Tú eliges según tu caso.

Q: ¿Cuántos agents puedo correr en paralelo? A: Técnicamente sin límite, pero cada uno consume API calls y context. Best practice: 2-3 max en paralelo.

Recursos

- **Official docs**: https://code.claude.com/docs/
- **MCP Spec**: https://modelcontextprotocol.io/
- **API Docs**: https://docs.anthropic.com/

- **Examples**: github.com/anthropics/claude-code-examples