

REFERENCIA RÁPIDA DE CLAUDE CODE

Comandos CLI, flujos de trabajo, MCP, hooks, configuración

Primeros Pasos

Instalar y Lanzar

```
npm install -g @anthropic-ai/claude-code
claude # iniciar sesión interactiva
claude --version # verificar versión
claude update # actualizar a la última versión
```

Autenticación

```
claude login # OAuth por navegador
export ANTHROPIC_API_KEY="sk-ant-..."
claude logout # cerrar sesión
```

Comandos Slash

Comandos de Sesión

```
/help Mostrar comandos disponibles
/clear Limpiar historial de conversación
/compact Condensar contexto para ahorrar tokens
/cost Mostrar uso de tokens y costo
/status Mostrar información de sesión y modelo
/new Iniciar una nueva conversación
/config Abrir o ver configuración
/doctor Diagnosticar problemas de configuración
/init Crear un CLAUDE.md para el proyecto
/login Autenticarse con Anthropic
/logout Cerrar autenticación
```

Comandos de Flujo de Trabajo

```
/bug Reportar un error
/review Solicitar una revisión de código
/pr Crear o actualizar un pull request
/commit Hacer commit de los cambios staged con un mensaje
```

Atajos de Teclado

```
Ctrl+C Cancelar generación actual
Ctrl+D Salir de Claude Code (EOF)
Escape Cancelar entrada / edición actual
Tab Autocompletar rutas de archivos y comandos
Up/Down Navegar historial de entrada
```

Flags CLI

Flags Comunes

```
-p, --print Modo no interactivo (headless)
--model Establecer modelo: `opus`, `sonnet`, `haiku`
--output-format Salida como `text`, `json`, `stream-json`
--allowedTools Restringir herramientas: `Edit,Read,Bash`
--max-turns N Limitar turnos de conversación
--debug Habilitar logging de depuración
-n, --name Nombrar la sesión
```

Headless / Scripting

```
claude -p "explain this error" < log.txt
claude -p "list todos" --output-format json
echo "fix typos" | claude -p
```

Archivos de Configuración

Jerarquía de CLAUDE.md

```
./CLAUDE.md Instrucciones a nivel de proyecto (en el repo)
./.claude/settings.json Configuración del proyecto (permisos, hooks)
~/./claude/CLAUDE.md Instrucciones globales del usuario (todos los proyectos)
~/./claude/settings.json Configuración global del usuario
~/./claude/projects/*/CLAUDE.md Instrucciones de usuario por proyecto (fuera del repo)
```

Variables de Entorno

```
ANTHROPIC_API_KEY Clave API para autenticación directa
CLAUDE_MODEL Override del modelo por defecto
CLAUDE_CONFIG_DIR Ruta personalizada del directorio de configuración
```

Permisos

settings.json

```
{
  "permissions": {
    "allow": ["Read", "Glob", "Grep"],
    "deny": ["Bash(rm *)"]
  }
}
```

Modos de Permiso

```
default Preguntar antes de herramientas de riesgo
--dangerously-skip-permissions Permitir todas las herramientas (solo CI/scripts)
--allowedTools Flag CLI para restringir conjunto de herramientas
```

Servidores MCP

¿Qué Son los Servidores MCP?

Los servidores Model Context Protocol extienden Claude Code con herramientas personalizadas — bases de datos, APIs, servicios. Se gestionan vía CLI o `./mcp.json`.

Gestión por CLI

```
claude mcp add server-name -- cmd arg1 arg2
claude mcp list
claude mcp remove server-name
```

Config ./mcp.json

```
{
  "mcpServers": {
    "my-db": {
      "command": "python",
      "args": ["-m", "mcp_server db"],
      "env": { "DB_URL": "${DATABASE_URL}" }
    }
  }
}
```

Alcance

```
./mcp.json Servidores MCP a nivel de proyecto
~/./claude/mcp.json Servidores MCP globales del usuario
claude mcp add -s user Agregar al alcance de usuario
claude mcp add -s project Agregar al alcance de proyecto
```

Hooks

Eventos de Hook

```
/PreToolUse Se ejecuta antes de usar una herramienta
/PostToolUse Se ejecuta después de completar una herramienta
/Notification Se ejecuta cuando Claude envía una notificación
/Stop Se ejecuta cuando Claude termina una respuesta
```

Ejemplo de settings.json

```
{
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "Bash",
        "hooks": [
          {
            "type": "command",
            "command": "check-command.sh $INPUT"
          }
        ]
      }
    ]
  }
}
```

Comportamiento de Hooks

```
exit 0 Permitir que la herramienta continúe
exit 2 Bloquear la ejecución de la herramienta
stdout Feedback JSON a Claude
```

SDK y Automatización

Scripting Headless

```
# Single prompt, get JSON output
claude -p "summarize main.py" \
  --output-format json
```

```
# Pipe input
git diff | claude -p "review this diff"
```

CI / GitHub Actions

```
- uses: anthropics/claude-code-action@v1
  with:
    claude_args: >
      --max-turns 5
      --model claude-sonnet-4-6
```

Consejos

Usa `--max-turns` para limitar el costo en automatización. Usa `--output-format json` para parsear resultados programáticamente. Combina con `--allowedTools` para mayor seguridad.

Modelos

Selección de Modelo

```
claude --model opus # mayor capacidad
claude --model sonnet # equilibrado (por defecto)
claude --model haiku # más rápido, más económico
```

Atajos de Modelo

```
opus Claude Opus — máxima capacidad
sonnet Claude Sonnet — por defecto, equilibrado
haiku Claude Haiku — rápido y económico
--model full-name ej. `claude-sonnet-4-6`
```

Buenas Prácticas

Escribir CLAUDE.md

Incluye convenciones del proyecto, stack tecnológico, comandos de build e instrucciones de prueba en CLAUDE.md. Mantenlo conciso — Claude lo lee en cada sesión. Usa `/init` para generarlo.

Gestión de Costos

```
/cost Verificar uso de tokens en curso
/compact Comprimir contexto cuando crece mucho
--max-turns Limitar turnos en automatización
sonnet / haiku Usar modelos más económicos para tareas simples
```

Prompts Efectivos

```
Sé específico "Fix the null check in auth.ts:42" > "fix bug"
Da contexto Referencia archivos, funciones, mensajes de error
Usa @file Referencia archivos directamente: `@src/main.ts`
Usa imágenes Arrastra capturas de pantalla para contexto visual
Itera Sigue refinando; usa /clear para reiniciar
```