Claude

- codebase-memory-mcp — Tutorial de instalación y uso en Linux

codebase-memory-mcp — Tutorial de instalación y uso en Linux

Servidor MCP de inteligencia de código. Indexa codebases en un knowledge graph persistente, permitiendo que Claude consulte la estructura de tu código (funciones, llamadas, clases, rutas HTTP) sin tener que leer archivo por archivo. Esto reduce drásticamente el consumo de tokens en proyectos grandes o multi-repo.

Repositorio: https://github.com/DeusData/codebase-memory-mcp

---

1. Instalación

Instalación en una línea:

```
curl -fsSL https://raw.githubusercontent.com/DeusData/codebase-memory-mcp/main/install.sh | bash
```

Con visualización de grafo (UI web en `localhost:9749`):

```
curl -fsSL https://raw.githubusercontent.com/DeusData/codebase-memory-mcp/main/install.sh | bash -s -- --ui
```

Verificar dónde quedó instalado el binario

```
which codebase-memory-mcp
```

En Linux, el instalador suele colocarlo en `~/.local/bin/codebase-memory-mcp`. Si el comando anterior no devuelve nada, asegúrate de tener `~/.local/bin` en tu `PATH`:

```
export PATH="HOME/.local/bin:PATH"
```

El propio comando `install` detecta automáticamente los agentes instalados (Claude Code, VS Code, etc.) y configura las entradas MCP por ti. Si esto funciona, puedes saltar directamente al paso 3.

---

2. Configuración manual (si no quieres usar el instalador automático)

Añade esta entrada a tu archivo de configuración MCP, por defecto, es muy probable que ya este configurado. Puede ser:

- **Global**: `~/.claude/.mcp.json` (disponible en todos tus proyectos)
- **Por proyecto**: `.mcp.json` en la raíz del repo

```
{
  "mcpServers": {
    "codebase-memory-mcp": {
      "command": "/home/TU_USUARIO/.local/bin/codebase-memory-mcp",
      "args": []
    }
  }
}
```

Sustituye `/home/TU_USUARIO/.local/bin/codebase-memory-mcp` por la ruta real que obtuviste con `which` en el paso anterior.

Reinicia tu agente (Claude Code, etc.) y verifica con:

```
/mcp
```

Deberías ver `codebase-memory-mcp` listado con 14 herramientas disponibles.

---

3. Uso con múltiples proyectos

Un único binario y una única instalación sirven para todos tus proyectos. No necesitas reinstalar nada por repo.

Cada proyecto que indexas queda registrado por separado en una base de datos SQLite local, ubicada en:

```
~/.cache/codebase-memory-mcp/
```

Indexar un proyecto por primera vez

Dentro de la conversación con el agente, simplemente pide:

```
Index this project
```

O manualmente desde la terminal:

```
codebase-memory-mcp cli index_repository '{"repo_path": "/ruta/absoluta/al/repo"}'
```

Ver todos los proyectos indexados

```
codebase-memory-mcp cli list_projects
```

Indexado automático

Para que cada nuevo proyecto se indexe solo al conectar el agente:

```
codebase-memory-mcp config set auto_index true
```

Si tienes repos muy grandes, puedes ajustar el límite de archivos para el auto-index:

```
codebase-memory-mcp config set auto_index_limit 50000
```

Consultar un proyecto específico

Si tienes varios repos indexados y quieres apuntar a uno en concreto, usa el parámetro `project`:

```
codebase-memory-mcp cli search_graph '{"name_pattern": ".*Handler.*", "project": "nombre_del_proyecto"}'
```

Si solo hay un proyecto activo en el contexto del agente, normalmente lo infiere sin que lo especifiques.

---

4. Herramientas MCP más útiles

Herramienta	Para qué sirve
`index_repository`	Indexa un repo en el grafo (sync automático después)
`list_projects`	Lista todos los proyectos indexados
`search_graph`	Búsqueda estructurada por nombre, label, archivo
`trace_path`	Traza quién llama a una función y a qué llama ella
`get_architecture`	Visión general: lenguajes, paquetes, rutas, módulos
`detect_changes`	Mapea un git diff a los símbolos afectados
`get_code_snippet`	Lee el código fuente de una función por su nombre cualificado
`query_graph`	Queries tipo Cypher sobre el grafo
`search_code`	Grep dentro de los archivos ya indexados
`delete_project`	Elimina un proyecto y todos sus datos del grafo

Ejemplos de preguntas en lenguaje natural

What calls ProcessOrder?

Find all HTTP routes

Trace the call path from main

Show me the architecture overview

What changed in my last commit and what does it affect?

---

5. Mantenimiento

Actualizar el binario

```
codebase-memory-mcp update
```

Desinstalar

```
codebase-memory-mcp uninstall
```

Esto elimina configuraciones de agentes, hooks e instrucciones, pero **no** borra el binario ni las bases de datos SQLite.

Resetear todos los datos indexados

```
rm -rf ~/.cache/codebase-memory-mcp/
```

---

6. Troubleshooting rápido

Problema	Solución
`/mcp` no muestra el servidor	Verifica que la ruta en `.mcp.json` sea absoluta. Reinicia el agente. Prueba: `echo '{}' | /ruta/al/binario` debería devolver JSON
`index_repository` falla	Usa siempre ruta absoluta: `repo_path="/ruta/absoluta"`
`trace_path` devuelve 0 resultados	Primero busca el nombre exacto con `search_graph(name_pattern=".*NombreParcial.*")`
Resultados de proyecto equivocado	Añade `project="nombre"` explícitamente. Usa `list_projects` para ver los nombres disponibles
Binario no encontrado tras instalar	Añádelo al PATH: `export PATH="$HOME/.local/bin:$PATH"`

Notas

- Todo el procesamiento es local — tu código nunca sale de tu máquina.
- El proyecto soporta 158 lenguajes vía tree-sitter, con resolución semántica de tipos (Hybrid LSP) para Python, TypeScript/JavaScript, PHP, C#, Go, C, C++, Java, Kotlin y Rust — relevante si trabajas con repos ROS 2 mixtos (Python + C++).
- Licencia MIT.