EcomLabs

STEP-BY-STEP GUIDE

Claude Code and Postgres over MCP, with no delete rights

The read-only role's SQL, the MCP server config for Claude Code with no password in the repo, and the security test that confirms DELETE, UPDATE and INSERT are rejected. Tested on PGlite and with DBHub 1.6.1.

FOR	Teams connecting Claude Code to a PostgreSQL database
TESTED WITH	Node 22.22.3 · PGlite 0.5.10 (PostgreSQL 18.6) · DBHub 1.6.1 · Windows 11
UPDATED	November 11, 2026
DOCS	code.claude.com/docs/en/mcp
ARTICLE	ecomlabs.dev/g/mcp-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before you start	What you will set up
Step 01	The read-only role
Step 02	The DBHub config
Step 03	Register the server in Claude Code
Step 04	Point it at a replica
Step 05	The security test
When you finish	Checklist

BEFORE YOU START

What you will set up

A Postgres role that can only read, the DBHub MCP server configured with that role in Claude Code, and a test that confirms writes are rejected. We tested the SQL on November 11, 2026 on PGlite 0.5.10 (PostgreSQL 18.6) with `node --test`: 18 of 18 tests passed. We tested the server with DBHub 1.6.1 and a stub MCP client, with no Claude Code and no model involved.

The core idea: the server's read-only mode helps, but Postgres is what rejects a `DELETE`, because of the role's privileges.

STEP 01

The read-only role

Save it as `role.sql` and run it as a user that can create roles. Replace `app_owner` with the role that owns your tables (the one that runs your migrations):

```
CREATE ROLE mcp_readonly WITH LOGIN CONNECTION LIMIT 3;

GRANT USAGE ON SCHEMA public TO mcp_readonly;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO mcp_readonly;

ALTER DEFAULT PRIVILEGES FOR ROLE app_owner IN SCHEMA public
  GRANT SELECT ON TABLES TO mcp_readonly;

ALTER ROLE mcp_readonly SET default_transaction_read_only = on;
ALTER ROLE mcp_readonly SET statement_timeout = '10s';
```

- **Password:** keep it out of the SQL. In psql, as an admin: `\password mcp_readonly`.
- `FOR ROLE`**:** only the creating role's default privileges apply. If another role creates tables, repeat the line for it.
- **Database older than PostgreSQL 15:** also run `REVOKE CREATE ON SCHEMA public FROM PUBLIC;`
- **Personal data:** if Claude does not need it, grant `SELECT` per table or per column instead of `ALL TABLES`.

STEP 02

The DBHub config

DBHub only accepts `readonly` in its TOML file (the `--readonly` flag is gone). Save `dbhub.toml`:

```
[[sources]]
id = "replica"
dsn = "${MCP_PG_DSN}"

[[tools]]
name = "execute_sql"
source = "replica"
readonly = true
max_rows = 200
```

The TOML holds no password: it reads the `MCP_PG_DSN` variable. Set it in your shell or secrets manager, shaped like this:

```
postgres://mcp_readonly:<password>@<replica-host>:5432/<database>?sslmode=require
```

`sslmode=require` encrypts without verifying the certificate; `verify-full` also checks the certificate and hostname.

STEP 03

Register the server in Claude Code

Just for you (local scope, stored in `~/.claude.json`):

```
claude mcp add --env MCP_PG_DSN="$MCP_PG_DSN" --transport stdio postgres-ro \
  -- npx -y @bytebase/dbhub@1.6.1 --config /absolute/path/dbhub.toml
```

- `--transport stdio` sits between `--env` and the name: if the name comes right after `--env`, the CLI rejects it.
- Use an absolute path for `--config`: the docs do not say which directory a stdio server starts in.
- Run `/mcp` in Claude Code and check that the server shows a checkmark.

For the team (project scope, a `.mcp.json` file at the repo root, which does go in git):

```
{
  "mcpServers": {
    "postgres-ro": {
      "command": "npx",
      "args": ["-y", "@bytebase/dbhub@1.6.1", "--config", "${DBHUB_CONFIG}"],
      "env": {
        "MCP_PG_DSN": "${MCP_PG_DSN}"
      }
    }
  }
}
```

> **CAUTION**
>
> Never write the password into `.mcp.json`: only `${VAR}` references. With `claude mcp add`, `~/.claude.json` stores the expanded value; it stays out of git, but treat it as a secret. If a password reached a commit, change it with `\password`. We did not run this command or this file: we checked them against `claude mcp add --help` (Claude Code 2.1.238) and the documentation.

STEP 04

Point it at a replica

- On a standby, `transaction_read_only` is always true, and writes, DDL and `GRANT` produce errors.
- Create the role on the primary: with physical replication it reaches the replica.
- The replica lags: data from a second ago may not be there yet.
- Long queries are canceled once `max_standby_streaming_delay` is exceeded.
- Without a replica, use the same role on the primary with a low `statement_timeout` and `max_rows`.

We did not test a replica (PGlite has no replication); this step rests on the PostgreSQL documentation.

STEP 05

The security test

Save `probe.sql`, replace `orders` with one of your tables, and run it once against the primary with `psql "$MCP_PG_DSN" -f probe.sql`:

```
SHOW default_transaction_read_only;
SET default_transaction_read_only = off;
SELECT current_user, session_user;
SELECT rolsuper, rolcreaterole, rolcreatedb, rolbypassrls FROM pg_roles WHERE rolname = current_user;
SELECT count(*) AS owned_tables FROM pg_tables WHERE tableowner = current_user;
DELETE FROM orders WHERE false;
UPDATE orders SET total = total WHERE false;
INSERT INTO orders SELECT * FROM orders WHERE false;
```

LINE	EXPECTED RESULT
`SHOW`	`on` (the `ALTER ROLE` applied at login)
`current_user`	`mcp_readonly`
`pg_roles`	four `false` values
`owned_tables`	`0`
The three writes	`permission denied for table orders`

- The `SET` turns read-only off for this session; without it, everything fails with `cannot execute DELETE in a read-only transaction` and you never see the privilege check.

- `WHERE false` touches no rows even if the role were wrong, and Postgres still checks the privilege.
- Then ask Claude Code to run `DELETE FROM orders WHERE false` with the Postgres tool: with `readonly = true`, DBHub answers `READONLY_VIOLATION`.

In our test with DBHub 1.6.1, `DELETE` and `COMMIT; DELETE` got `READONLY_VIOLATION` with `readonly = true`, and `permission denied for table orders` without it. The table kept its rows.

> **CAUTION**
> Audit `SECURITY DEFINER` functions that write: PostgreSQL grants `EXECUTE` to `PUBLIC` by default, and in our test one of them deleted a row when called from `mcp_readonly`. Remove it with `REVOKE EXECUTE ON FUNCTION ... FROM PUBLIC`. Never run write tests without `WHERE false` against your production database.

WHEN YOU FINISH

Checklist

- [ ] The MCP server uses its own role, not the app's or the migrations user.
- [ ] The role has `USAGE` on the schema and `SELECT` only where needed.
- [ ] `ALTER DEFAULT PRIVILEGES FOR ROLE` names the role that creates your tables.
- [ ] The password was set with `\password` and lives only in `MCP_PG_DSN`.
- [ ] `dbhub.toml` has `readonly = true` and `max_rows`; the version is pinned.
- [ ] `.mcp.json` only holds `${VAR}` references.
- [ ] The DSN points at a replica, or at the primary with a low `statement_timeout`.
- [ ] `probe.sql` gets `permission denied` on all three writes and `0` owned tables.
- [ ] No `SECURITY DEFINER` function that writes grants `EXECUTE` to `PUBLIC`.