AgentSQL

Zhidun

Getting Started

v4.6.0

agentsql.cn

v4.6.0 Documentation

Contents

what you will get

Be clear about what not to do

Preconditions and compatibility matrix

Path A: 5 minutes of zero configuration to see the effect (one command)

Path B: Connect to your own library (real production closed loop)

- 1. Install and start
- 2. Log in to the console
- 3. Add data source
- 4. Create Agent
- 5. Establish a minimum allow permission policy
- 6. Connect to MCP client
- Task-level temporary read-only key: application, alien approval and result budget

5-minute acceptance checklist

v4.4.0: Read-only check before upgrade

Next step

Get started quickly with AgentSQL

Language: English | Simplified Chinese

> Current guidance: Applies to v4.4.0 and compatible 4.4.x maintenance releases. The historical introduction version and acceptance record retain the original facts; for daily entry, see document index .

> This article explains the installation and use of AgentSQL v4.4.0. Before executing the following installation, demo and mirroring commands to fix the v4.4.0 tag, first check the official assets and signatures in GitHub Releases, and confirm that the corresponding GHCR mirror can be pulled anonymously; the source code repository is `github.com/cuipengdba/agentsql`.

AgentSQL is a database security gateway and production-level MCP Server placed between the AI Agent and the business database, allowing each SQL to pass identity, permissions, rules and decision-making checks before execution, and execute within auditable boundaries.

```
People → AI Agent / LLM → MCP → AgentSQL → PostgreSQL / MySQL business database
```

AgentSQL provides both a database security gateway and its own data Q&A interface. The Q&A service can turn natural language into proposed SQL; that SQL must still pass the gateway's identity, authorization, policy, and audit checks. AgentSQL does not replace database accounts and permissions. A least-privileged database account remains the final boundary.

what you will get

- Assign an independent API Key and `readonly`, `dml`, and `ddl` capability levels to each Agent.
- Create a default denial authorization policy by Agent × Data Source × Object.
- Perform analysis, rule judgment, controlled execution, result row restriction and basic masking on a single SQL statement.
- Check the evidence chain and application layer audit of `allow`, `deny`, `warn`, `approve`, and `error` on the console.
- 9 fixed base tools are provided to MCP clients via Streamable HTTP or stdio; an additional 8 tools are provided for PostgreSQL B5 sessions/scheduled transactions enabled by default.

Be clear about what not to do

- Does not replace database accounts, network isolation, TLS termination, backup, monitoring and change management.
- Requests that bypass AgentSQL and connect directly to the database are not protected.
- PostgreSQL's B2 column-level authorization, B5 cross-request logical sessions and planned transactions are all enabled by factory default; MySQL does not enter the B2 PostgreSQL path and does not support B5 cross-request transactions. Only one top-level SQL is still allowed per operation, subject to preflight plans and session security boundaries.
- Approval does not automatically execute SQL; auditing is not regulatory-level WORM.
- There is currently no support for full DLP, enterprise identity, high availability, or Kubernetes orchestration.

Preconditions and compatibility matrix

range	Currently supported	Description
Install the host with one click	`linux/amd64`、`linux/arm64`；glibc 2.28+、systemd	The linux/arm64 native glibc package is provided starting from v0.4.0; `x86_64` / `amd64` corresponds to amd64, `aarch64` / `arm64` corresponds to arm64
other hosts	Docker / Docker Compose	v4.4.0 GHCR tag is subject to `linux/amd64` + `linux/arm64` multi-arch manifest and anonymous pull verification; musl / Alpine, CentOS 7 and platforms other than the above two architectures do not provide native packages
business database	MySQL 8；PostgreSQL 14–18	Create an independent, least privileged running account for AgentSQL
Control plane storage	SQLite；PostgreSQL 15+	SQLite is the default; PostgreSQL can separate metadata and audit
Browser console	`http://127.0.0.1:7780`	Default bound loopback address
Local Live Demo	Docker Engine / Docker Desktop、Docker Compose v2	Compose and verified GHCR images using the v4.4.0 exact tag; no source code cloning or building required

Path A: 5 minutes of zero configuration to see the effect (one command)

This is the dedicated Demo mode for v4.4.0: using synthetic data and fixed demo identity, there is no need to clone the source code, edit any configuration, or build an image. Before execution, you must check that the Compose file in the v4.4.0 Git tag and the three GHCR images it references are public and can be pulled anonymously; the first startup will pull the image and seed fixed demo data.

Just one `docker-compose.yml` (execute in any empty directory):

```
# 1) Create a new empty directory and enter
mkdir agentsql-demo && cd agentsql-demo

# 2) Download the self-contained compose (only this file, about 2KB)
curl -fsSL -o docker-compose.yml
https://raw.githubusercontent.com/cuipengdba/agentsql/v4.4.0/deploy/quickstart/docker-compose.yml

# 3) One command to pull up all containers (postgres + mysql + seed + gateway)
docker compose up -d
```

Equivalent command for Windows (PowerShell):

```
New-Item -ItemType Directory -Force agentsql-demo | Out-Null; Set-Location agentsql-demo
Invoke-WebRequest -OutFile docker-compose.yml
https://raw.githubusercontent.com/cuipengdba/agentsql/v4.4.0/deploy/quickstart/docker-compose.yml
docker compose up -d
```

Wait for about 1–2 minutes. After all four containers are ready, open http://127.0.0.1:17880: the login page has automatically filled in the demo account, just click "Login". You can also check the progress in the terminal:

```
docker compose ps            # The gateway status changes to healthy and is ready.
docker compose logs -f seed  # Seeing DEMO_SEED_OK means seeding is complete
```

After entering the "Demonstration Station", press the five script cards to experience them one by one (each card has built-in accurate fixed SQL, click to see the real results):

1. **column authorization + masking**: Sealed JOIN query is released, and the two columns `phone` and `email` in the returned results are masked.
2. **column authorization is missing**: Querying the unauthorized column is rejected and does not reach the business database.

3. **object has exceeded the authority of**: access to `internal_notes` is explicitly denied by the policy, and the table content is not returned.
4. **DDL exceeds authority**: `DROP TABLE` is intercepted.
5. The **write operation is converted to manual**: `UPDATE` is not executed directly, but is converted to `approve` and awaits manual approval.

Complete cleanup after the experience (delete containers and demo data volumes):

```
docker compose down -v
```

> Demo description: All credentials are fixed demo values, written in compose, and are only used for local experience. Do not connect to real data or databases. The Demo site is only bound to the loopback address `127.0.0.1` by default and is not open to the LAN.

Path B: Connect to your own library (real production closed loop)

The following steps must be completed in order. If the authorization policy is missing, no table can be accessed even if the data source and Agent have been created.

1. Install and start

The linux/arm64 native glibc package is provided starting from **v0.4.0:** one-click installation supports `linux/amd64` (`uname -m` is `x86_64` / `amd64`) and `linux/arm64` (`aarch64` / `arm64`), both require glibc 2.28+, systemd, and the command requires root permissions:

The following example fixesv4.4.0. Before executing, confirm that the official Release provides this version.`install.sh`、`SHA256SUMS`and signature; pressPublish asset descriptionUse an independently verified public key to verify the signature manifest, and then use the hash in the manifest to verify the installer.

```
# Applicable prerequisites: linux/amd64 or linux/arm64 + glibc 2.28+ + systemd
curl -fsSLo install.sh https://github.com/cuipengdba/agentsql/releases/download/v4.4.0/install.sh
# First use the verified SHA256SUMS to verify install.sh, and then run:
sudo sh ./install.sh install --version v4.4.0
```

In addition to one-click installation, you can also build using source code or Docker Compose according to the deployment guide. After confirming that the v4.4.0 image has been made public and that the dual-architecture manifest and anonymous pull have passed, run `docker pull ghcr.io/cuipengdba/agentsql:v4.4.0`; the manifest should match `linux/amd64` / `linux/arm64`. Release status and assets are available on the GitHub Releases page.

2. Log in to the console

Open http://127.0.0.1:7780, the default administrator username is `admin`. Non-interactive installation will not print the random password to the terminal; root can read `/etc/agentsql/agentsql.env`, or add `--show-password` explicitly when reinstalling.

3. Add data source

First create a separate minimum-privileged running account for AgentSQL in the business database. Never use `superuser`, database owner or MySQL `root`. Fill in `host`, `port`, `database`, `username`, `password` in "Data Source", grant the minimum table permissions according to the actual query needs, and then click "Test Connection".

For designated instances of PostgreSQL and Vastbase G100 Build 9 PG mode, you can select `verify-full` in the data source form and fill in the CA absolute path and optional certificate host name that are readable by the gateway. The Vastbase server must refuse plaintext connections for this read-only account; other database TLS capabilities are configured according to their respective acceptance ranges. For details, see v1.8.0 TLS acceptance.

4. Create Agent

Create an identity in "Agent" and select the ability level:

- `readonly`: read-only query.
- `dml`: Query and `INSERT`, `UPDATE`, `DELETE`.
- `ddl`: Contains structure change capabilities.

Save the API Key that will be displayed only once after creation. Please put it into the client's secret storage immediately and do not write it to the repository or chat history.

5. Establish a minimum allow permission policy

> **Key steps: AgentSQL rejects without authorization by default. Just by creating the data source and Agent, you still cannot access any tables.**

Create a new `allow` policy that binds the Agent and the data source in "Permissions", starting from the single table you really need, such as `public.customers` or `customers`. Do not use `*` or `schema.*` as ordinary examples: although both are legal, they represent full column authorization for the entire table, which will invalidate the column-level whitelist.

After installing v4.4.0, the console "Access Wizard" can reuse the same Agent and data source selection to do connection and permission pre-checking, target SQL offline interpretation, and generate keyless access instructions for Dify 1.17.1 or MCP Inspector 0.16.8. Preflight does not execute Agent SQL; the number of policies greater than zero or the offline conclusion being "unknown" does not prove that the target SQL is authorized. Please verify the allow and deny requests respectively on the real client, and then use the audit ID in the response to jump to the audit details for verification. For configuration steps and restrictions on client reachable addresses, see user manual .

6. Connect to MCP client

Streamable HTTP is preferred when the service is already running by the installer or systemd. According to MCP access guide , merge the following minimum skeleton into the client configuration; the specific file path and HTTP transport key name of the client are subject to the official MCP document of the client:

```
{
  "mcpServers": {
    "agentsql": {
      "url": "http://127.0.0.1:7780/mcp",
      "headers": {
        "Authorization": "Bearer <Agent API Key>"
      }
    }
  }
}
```

After restarting or reloading the client, first call `list_datasources` and `list_schema`. Then send an authorized single `SELECT` with `WHERE` through `query`:

```
-- Applicable premise: Replace with authorized real tables and columns; submit through MCP query tool
SELECT id, name FROM public.customers WHERE id = 1 LIMIT 10;
```

Confirm that `allow` appears on the "Audit" page. Then use `readonly` Agent to send an update without `WHERE` through `execute_write` and fill in `reason`:

```
-- Applicable premise: only used for verification interception; submit and fill in reason through MCP
execute_write tool
UPDATE public.customers SET name = 'blocked-test';
```

The request should be rejected by R002 (unconditional batch write) and/or R003 (read-only Agent write operation) and does not reach the business database; confirm `deny` on the audit page. Tool layer pre-rejection may occur before the pipeline, so errors such as missing `reason` do not guarantee an audit; `reason` must be filled in during acceptance.

MCP entry provided from **v0.7.0:** Clients that support `2026-07-28` can use `http://127.0.0.1:7780/mcp/2026-07-28` instead. Each POST must carry the Agent Bearer Key, `MCP-Protocol-Version: 2026-07-28`, and `Mcp-Method`. When calling the tool, it must also carry `Mcp-Name` that is consistent with the tool name; the request body `params._meta` must declare the same version of the protocol and client capabilities. This portal does not require `initialize` and does not have the transport session, GET event stream, and B5 session/scheduled transaction facilities of the old portal. Existing clients continue to use `/mcp`; for request examples and version boundaries, see MCP access guide .

Task-level temporary read-only key: application, alien approval and result budget

Users with `query.execute` submit precise SQL, Agent, data source, expiration time up to 24 hours, and result budget of 1-10 times (default 1 time) in the console "Temporary read-only authorization". Another user with the same tenant who has both `strategy.manage` and `query.execute` views the SQL and approves or rejects it; after approval, only the original applicant can receive the task key at one time. The list only displays the SQL hash and status, and the key only appears in the response of successful collection. This Key can only call MCP `query`, and the original SQL bytes must be exactly the same as when applied; the original Agent's object permissions, rules, masking, and database read-only account constraints continue to take effect. It can be revoked when used up, and the management and execution records can be checked by the authorization ID on the "Audit" page. The direct issuance API of v0.7.0 returns HTTP 410 in this version, and the existing v0.7.0 Key will continue to take effect according to the expiration and revocation rules. Operation and recovery boundaries see task-level temporary read-only authorization .

5-minute acceptance checklist

- `GET /healthz` returns HTTP 200, and `status` in JSON is `ok` and contains `version`.
- `GET /readyz` returns HTTP 200.
- The "Audit" page of the console has at least 1 `allow` and 1 `deny`.
- `deny` details show that the rule is hit, and the target business table is not updated.

- The API Key, database password, and `AGENTSQL_SECRET` are not written to the command history, repository, or public logs.

```
# Applicable premise: AgentSQL is running on this machine 127.0.0.1:7780
curl -fsS http://127.0.0.1:7780/healthz
curl -fsS -o /dev/null -w '%{http_code}\n' http://127.0.0.1:7780/readyz
```

v4.4.0: Read-only check before upgrade

Use v4.4.0 `agentsqlctl` with the target configuration to check control-plane schema versions first. Migrations for managed file operations and SQLBot child-binding parent records were introduced in v4.2.0; v4.4.0 retains combined metadata schema 29, separate metadata schema 27, and audit schema 7. Upgrades from v4.1.0 or earlier require paired metadata and audit backups, stopping the old gateways and Q&A service, waiting for the B2 lease to expire, and a paired restore exercise in isolation. Follow the maintenance-window guide for the three-gateway topology; do not run old and new binaries together:

```
agentsqlctl doctor --config /etc/agentsql/config.yaml --format json
```

The exit code `0` indicates that all configured migration flows are the current version, `2` indicates that there are pending migrations, and `1` indicates errors in configuration, connection, or migration history. This check is read-only, does not perform DDL, and does not replace backups or database structure and data consistency checks. Before upgrading, you must stop writing and back up the control plane database and `AGENTSQL_SECRET` in pairs; when deploying PostgreSQL sub-databases, you must back up the metadata and audit libraries at the same time. Then the migration owner explicitly migrates. For the steps, see deployment guide and current version operation and maintenance manual . The released v0.6.0 installation package does not have the new command options in this section.

Next step

- User Manual: Press the console menu to learn each ability and boundary.
- Guide to sensitive column discovery: Explicit table ranges, sampling safety, confidence, and masked draft boundaries.
- MCP Access Guide : Two transports, 11 basic tools, B5 session/transaction tools, call timing and client configuration.
- Task-level temporary read-only authorization : issuance, revocation, precise SQL and audit boundaries.

- Deployment Guide : source code, Compose, systemd, upgrade, backup and control plane migration.
- Current version operation and maintenance manual : Migration, audit faults and troubleshooting of old and new MCP entrances.
- Local Live Demo: Full demo credentials, reset scripts and security checklist.