clickhouse-client ClickHouse SQL shell

START HERE

`clickhouse-client -h <host> -u <user> --password`
Connect and ask for password

`clickhouse-client -q "<sql>"`
Run one query

`SHOW TABLES`
List tables

`DESCRIBE TABLE <table>`
Show columns

`<query> \G`
Vertical output

`clickhouse-client -m`
Multiline mode

`--format CSVWithNames`
Pick output format

`Ctrl+D`
Quit

Connect flags, string, config → `query` → **Query** REPL, -q, stdin, file → `format` → **Result** Pretty or --format → `pipe` → **Output** terminal, pager, file

CONCEPTS

Interactive mode runs a query on Enter without a semicolon; batch mode (-q, stdin, --queries-file) runs and exits. The client speaks the native TCP protocol, not HTTP.

Defaults differ by mode: PrettyCompact in the REPL, TabSeparated in batch mode. Set --format or a FORMAT clause when scripting.

Missing credentials fall back to CLICKHOUSE_USER, CLICKHOUSE_PASSWORD and CLICKHOUSE_HOST; flags and a connection string win over them.

The first config file found wins: --config, ./clickhouse-client.xml|yaml, ~/.config/clickhouse/config.xml|yaml, ~/.clickhouse-client/, /etc/clickhouse-client/. Any option can be a default there.

CONNECT

`clickhouse-client`
Connect to localhost:9000 as default

`clickhouse-client -h <host> --port <port> -u <user> --password <pw>`
Connect with explicit options
Omit the value after --password to be prompted.

`clickhouse-client clickhouse://<user>@<host>:<port>/<db>`
Connect with a connection string
Must be the first argument; combine with other flags except --host and --port.

`clickhouse-client -s --port 9440`
Force a TLS connection
Without --port and --secure both 9440 and 9000 are tried.

`clickhouse-client -d <db>`
Start in a database

`clickhouse-client --connection <name>`
Use a saved connection from config

`clickhouse-client -V`
Print the client version

`clickhouse-client --help --verbose`
List all options including settings

INSPECT

`SHOW DATABASES`
List databases
Alias: \l

`SHOW TABLES`
List tables in the current database
Alias: \d. Also \d FROM <db> or \d LIKE '<pat>'.

`DESCRIBE TABLE <table>`
Show a table's columns
Alias: \d <table>

`USE <db>`
Switch database
Alias: \c <db>

`SHOW CREATE TABLE <table>`
Show the table DDL

`help <name>`
Docs for a function, engine or setting
Also /help <name> and /man <name>.

`SELECT currentDatabase()`
Which database am I in?

`SELECT version()`
Server version

`SELECT * FROM system.tables WHERE database = '<db>'`
Table metadata

`SELECT * FROM system.processes`
Running queries

`SELECT * FROM system.settings WHERE name LIKE '%<x>%'`
Find a setting

RUN QUERIES

`clickhouse-client -q "<sql>"`
Run a query and exit

`clickhouse-client "<sql>"`
Query without --query

`echo "<sql>" | clickhouse-client`
Query from stdin

`clickhouse-client --queries-file <file.sql>`
Run queries from a file
Repeatable; cannot be combined with --query.

`clickhouse-client -q "<sql1>; <sql2>;"`
Several queries at once
INSERT with a non-VALUES format must be followed by an empty line.

`clickhouse-client -q "<sql>" -q "<sql2>"`
Repeat --query

RUN QUERIES (cont.)

`clickhouse-client --ignore-error --queries-file <file>`
Keep going after a failed query
Exit code stays 0 even when some queries failed.

`clickhouse-client -m`
Multiline: send only at ;

`\<Enter>`
Continue a query on the next line
Backslash before the line break in the default mode.

`clickhouse-client --max_threads <n>`
Pass a query setting

`SET <setting> = <value>`
Change a setting in the session

PARAMETERS

`--param_<name>=<value>`
Set a query parameter
Use {name:Type} in the query.

`{<name>:<Type>}`
Typed placeholder in SQL
Example: {id:UInt32}

`{<name>:Identifier}`
Parameterise table or column names

`SET param_<name> = '<value>'`
Set a parameter in the REPL

OUTPUT

`--format <format>`
Choose the output format
Shorthand -f. Any ClickHouse output format works.

`--format CSVWithNames`
CSV with a header row

`--format TabSeparatedWithNames`
TSV with a header row

`--format JSONEachRow`
One JSON object per line

`--format Vertical`
One value per line

`-E`
Same as --format Vertical
Alias --vertical.

`... FORMAT <format>`
Set the format inside the query

`<query> \G`
Vertical output for one query

`--pager 'less -S'`
Page wide results

`--progress off`
Hide the progress display

IMPORT & EXPORT

`clickhouse-client -q "INSERT INTO <table> FORMAT CSV" < <file>`
Load a CSV file

`clickhouse-client -q "INSERT INTO <table> FORMAT CSVWithNames" < <file>`
Load CSV with a header

`clickhouse-client -q "INSERT INTO <table> FORMAT JSONEachRow" < <file>`
Load NDJSON

`<cmd> | clickhouse-client -q "INSERT INTO <table> FORMAT <f>"`
Stream data in from a pipe

`clickhouse-client -q "<select>" --format CSVWithNames > <file>`
Export a query to CSV

`clickhouse-client -q "<select>" --format JSONEachRow | jq .`
Pipe JSON to jq

INTERACTIVE

`Ctrl+D`
Quit the client
Also exit, quit, q or :q.

`Ctrl+C`
Cancel the running query
A second press exits the client.

`Ctrl+R`
Fuzzy history search

`Alt+Shift+E`
Edit the query in $EDITOR
vim by default.

`.`
Repeat the last query
A lone / does the same.

`/clear`
Clear the terminal

RECIPES

Parameterised query to CSV

```
clickhouse-client --
param_day=2026-10-01 \
  -q "SELECT * FROM events
WHERE day = {day:Date}" \
  --format CSVWithNames >
events.csv
```

Load a CSV into ClickHouse Cloud

```
clickhouse-client --host
<id>.clickhouse.cloud --port
9440 \
  --user default --password \
  -q "INSERT INTO <table>
FORMAT CSVWithNames" < data.csv
```