cqlsh: the CQL shell

cqlsh: the CQL shell

`cqlsh` is a command-line interface for interacting with Cassandra using CQL (the Cassandra Query Language). It is shipped with every Cassandra package, and can be found in the bin/ directory alongside the cassandra executable. `cqlsh` is implemented with the Python native protocol driver, and connects to the single specified node.

Compatibility

In general, a given version of `cqlsh` is only guaranteed to work with the version of Cassandra that it was released with. In some cases, `cqlsh` may work with older or newer versions of Cassandra, but this is not officially supported.

Optional Dependencies

`cqlsh` ships with all essential dependencies. However, there are some optional dependencies that can be installed to improve the capabilities of `cqlsh`.

pytz

By default, `cqlsh` displays all timestamps with a UTC timezone. For Python 3.9 or higher, timestamps can be displayed in different timezones by modifying the `timezone` option in **cqlshrc** or by setting the environment variable `TZ`. Python 3.8 or lower, however, will also require the installation of pytz library.

cython

The performance of cqlsh's `COPY` operations can be improved by installing cython. This will compile the python modules that are central to the performance of `COPY`.

cqlshrc

The `cqlshrc` file holds configuration options for `cqlsh`. By default, the file is located the user's home directory at `~/.cassandra/cqlshrc`, but a custom location can be specified with the `--cqlshrc` option.

Example config values and documentation can be found in the `conf/cqlshrc.sample` file of a tarball installation. You can also view the latest version of the cqlshrc file online.

credentials

The credentials file contains a user-name and a password for cqlsh. A user-name and a password must be located in a section whose name matches the classname from the `[auth_provider]` section of the cqlshrc file. The credentials file must be owned by the user and no one else has permission to read the file.

Example config values and documentation can be found in the `conf/credentials.sample` file of a tarball installation.

cql history

All CQL commands you execute are written to a history file. By default, CQL history will be written to `~/.cassandra/cql_history`. You can change this default by setting the environment variable `CQL_HISTORY` like `~/some/other/path/to/cqlsh_history` where `cqlsh_history` is a file. All parent directories to history file will be created if they do not exist. If you do not want to persist history, you can do so by setting CQL_HISTORY to /dev/null. This feature is supported from Cassandra 4.1.

Command Line Options

Usage: `cqlsh.py [options] [host [port]]`

CQL Shell for Apache Cassandra

Options:

`--version`

show program's version number and exit

`-h --help`

show this help message and exit

`-C --color`

Always use color output

`--no-color`

Never use color output

`--browser=BROWSER`

The browser to use to display CQL help, where BROWSER can be: one of the supported browsers in https://docs.python.org/3/library/webbrowser.html. browser path followed by %s, example: /usr/bin/google-chrome-stable %s

`--ssl`

Use SSL

`-u USERNAME --username=USERNAME`

Authenticate as user.

`-p PASSWORD --password=PASSWORD`

Authenticate using password.

`-k KEYSPACE --keyspace=KEYSPACE`

Authenticate to the given keyspace.

`-f FILE --file=FILE`

Execute commands from FILE, then exit

`--debug`

Show additional debugging information

`--coverage`

Collect coverage data

`--encoding=ENCODING`

Specify a non-default encoding for output. (Default: utf-8)

`--cqlshrc=CQLSHRC`

Specify an alternative cqlshrc file location.

`--credentials=CREDENTIALS`

Specify an alternative credentials file location.

`--cqlversion=CQLVERSION`

Specify a particular CQL version, by default the highest version supported by the server will be used. Examples: "3.0.3", "3.1.0"

`--protocol-version=PROTOCOL_VERSION`

Specify a specific protcol version otherwise the client will default and downgrade as necessary

`-e EXECUTE --execute=EXECUTE`

Execute the statement and quit.

`--connect-timeout=CONNECT_TIMEOUT`

Specify the connection timeout in seconds (default: 5 seconds).

`--request-timeout=REQUEST_TIMEOUT`

Specify the default request timeout in seconds (default: 10 seconds).

`-t, --tty`

Force tty mode (command prompt).

`-v --v`

Print the current version of cqlsh.

Special Commands

In addition to supporting regular CQL statements, `cqlsh` also supports a number of special commands that are not part of CQL. These are detailed below.

CONSISTENCY

Usage: `CONSISTENCY <consistency level>`

Sets the consistency level for operations to follow. Valid arguments include:

- `ANY`
- `ONE`
- `TWO`
- `THREE`
- `QUORUM`
- `ALL`
- `LOCAL_QUORUM`
- `LOCAL_ONE`
- `SERIAL`
- `LOCAL_SERIAL`

SERIAL CONSISTENCY

Usage: `SERIAL CONSISTENCY <consistency level>`

Sets the serial consistency level for operations to follow. Valid arguments include:

- `SERIAL`
- `LOCAL_SERIAL`

The serial consistency level is only used by conditional updates (`INSERT`, `UPDATE` and `DELETE` with an `IF` condition). For those, the serial consistency level defines the consistency level of the serial phase (or "paxos" phase) while the normal consistency level defines the consistency for the "learn" phase, i.e. what type of reads will be guaranteed to see the update right away. For example, if a conditional write

has a consistency level of QUORUM (and is successful), then a QUORUM read is guaranteed to see that write. But if the regular consistency level of that write is ANY, then only a read with a consistency level of SERIAL is guaranteed to see it (even a read with consistency ALL is not guaranteed to be enough).

SHOW VERSION

Prints the cqlsh, Cassandra, CQL, and native protocol versions in use. Example:

```
cqlsh> SHOW VERSION
[cqlsh 5.0.1 | Cassandra 3.8 | CQL spec 3.4.2 | Native protocol v4]
```

SHOW HOST

Prints the IP address and port of the Cassandra node that cqlsh is connected to in addition to the cluster name. Example:

```
cqlsh> SHOW HOST
Connected to Prod_Cluster at 192.0.0.1:9042.
```

SHOW REPLICAS

Prints the IP addresses of the Cassandra nodes which are replicas for the listed given token and keyspace. This command is available from Cassandra 4.2.

Usage: SHOW REPLICAS <token> (<keyspace>)

Example usage:

```
cqlsh> SHOW REPLICAS 95
['192.0.0.1', '192.0.0.2']
```

SHOW SESSION

Pretty prints a specific tracing session.

Usage: SHOW SESSION <session id>

Example usage:

```
cqlsh> SHOW SESSION 95ac6470-327e-11e6-beca-dfb660d92ad8

Tracing session: 95ac6470-327e-11e6-beca-dfb660d92ad8
```

```
 activity                                                  | timestamp                  |
source    | source_elapsed | client
-----------------------------------------------------------+----------------------------+
-----------+----------------+-----------
                                        Execute CQL3 query | 2016-06-14 17:23:13.979000 |
127.0.0.1 |              0 | 127.0.0.1
 Parsing SELECT * FROM system.local; [SharedPool-Worker-1] | 2016-06-14 17:23:13.982000 |
127.0.0.1 |           3843 | 127.0.0.1
...
```

SOURCE

Reads the contents of a file and executes each line as a CQL statement or special cqlsh command.

`Usage`: `SOURCE <string filename>`

Example usage:

```
cqlsh> SOURCE '/home/calvinhobbs/commands.cql'
```

CAPTURE

Begins capturing command output and appending it to a specified file. Output will not be shown at the console while it is captured.

`Usage`:

```
CAPTURE '<file>';
CAPTURE OFF;
CAPTURE;
```

That is, the path to the file to be appended to must be given inside a string literal. The path is interpreted relative to the current working directory. The tilde shorthand notation (`'~/mydir'`) is supported for referring to `$HOME`.

Only query result output is captured. Errors and output from cqlsh-only commands will still be shown in the cqlsh session.

To stop capturing output and show it in the cqlsh session again, use `CAPTURE OFF`.

To inspect the current capture configuration, use `CAPTURE` with no arguments.

HELP

Gives information about cqlsh commands. To see available topics, enter `HELP` without any arguments. To see help on a topic, use `HELP <topic>`. Also see the `--browser` argument for controlling what browser is used to display help.

HISTORY

Prints to the screen the last `n` cqlsh commands executed on the server. The number of lines defaults to 50 if not specified. `n` is set for the current CQL session so if you set it e.g. to `10`, from that point there will be at most 10 last commands returned to you.

`Usage`:

```
HISTORY <n>
```

TRACING

Enables or disables tracing for queries. When tracing is enabled, once a query completes, a trace of the events during the query will be printed.

`Usage`:

```
TRACING ON
TRACING OFF
```

PAGING

Enables paging, disables paging, or sets the page size for read queries. When paging is enabled, only one page of data will be fetched at a time and a prompt will appear to fetch the next page. Generally, it's a good idea to leave paging enabled in an interactive session to avoid fetching and printing large amounts of data at once.

`Usage`:

```
PAGING ON
PAGING OFF
PAGING <page size in rows>
```

EXPAND

Enables or disables vertical printing of rows. Enabling `EXPAND` is useful when many columns are fetched, or the contents of a single column are large.

Usage:

```
EXPAND ON
EXPAND OFF
```

LOGIN

Authenticate as a specified Cassandra user for the current session.

Usage:

```
LOGIN <username> [<password>]
```

EXIT

Ends the current session and terminates the cqlsh process.

Usage:

```
EXIT
QUIT
```

CLEAR

Clears the console.

Usage:

```
CLEAR
CLS
```

DESCRIBE

Prints a description (typically a series of DDL statements) of a schema element or the cluster. This is useful for dumping all or portions of the schema.

Usage:

```
DESCRIBE CLUSTER
DESCRIBE SCHEMA
DESCRIBE KEYSPACES
```

```
DESCRIBE KEYSPACE <keyspace name>
DESCRIBE TABLES
DESCRIBE TABLE <table name>
DESCRIBE INDEX <index name>
DESCRIBE MATERIALIZED VIEW <view name>
DESCRIBE TYPES
DESCRIBE TYPE <type name>
DESCRIBE FUNCTIONS
DESCRIBE FUNCTION <function name>
DESCRIBE AGGREGATES
DESCRIBE AGGREGATE <aggregate function name>
```

In any of the commands, `DESC` may be used in place of `DESCRIBE`.

The `DESCRIBE CLUSTER` command prints the cluster name and partitioner:

```
cqlsh> DESCRIBE CLUSTER

Cluster: Test Cluster
Partitioner: Murmur3Partitioner
```

The `DESCRIBE SCHEMA` command prints the DDL statements needed to recreate the entire schema. This is especially useful for dumping the schema in order to clone a cluster or restore from a backup.

COPY TO

Copies data from a table to a CSV file.

`Usage`:

```
COPY <table name> [(<column>, ...)] TO <file name> WITH <copy option> [AND <copy option>
...]
```

If no columns are specified, all columns from the table will be copied to the CSV file. A subset of columns to copy may be specified by adding a comma-separated list of column names surrounded by parenthesis after the table name.

The `<file name>` should be a string literal (with single quotes) representing a path to the destination file. This can also the special value `STDOUT` (without single quotes) to print the CSV to stdout.

See `shared-copy-options` for options that apply to both `COPY TO` and `COPY FROM`.

Options for `COPY TO`

`MAXREQUESTS`

The maximum number token ranges to fetch simultaneously. Defaults to 6.

`PAGESIZE`

The number of rows to fetch in a single page. Defaults to 1000.

`PAGETIMEOUT`

By default the page timeout is 10 seconds per 1000 entries in the page size or 10 seconds if pagesize is smaller.

`BEGINTOKEN`, `ENDTOKEN`

Token range to export. Defaults to exporting the full ring.

`MAXOUTPUTSIZE`

The maximum size of the output file measured in number of lines; beyond this maximum the output file will be split into segments. -1 means unlimited, and is the default.

`ENCODING`

The encoding used for characters. Defaults to `utf8`.

`COPY FROM`

Copies data from a CSV file to table.

`Usage`:

```
COPY <table name> [(<column>, ...)] FROM <file name> WITH <copy option> [AND <copy
option> ...]
```

If no columns are specified, all columns from the CSV file will be copied to the table. A subset of columns to copy may be specified by adding a comma-separated list of column names surrounded by parenthesis after the table name.

The `<file name>` should be a string literal (with single quotes) representing a path to the source file. This can also the special value `STDIN` (without single quotes) to read the CSV data from stdin.

See `shared-copy-options` for options that apply to both `COPY TO` and `COPY FROM`.

Options for `COPY FROM`

`INGESTRATE`

The maximum number of rows to process per second. Defaults to 100000.

`MAXROWS`

The maximum number of rows to import. -1 means unlimited, and is the default.

`SKIPROWS`

A number of initial rows to skip. Defaults to 0.

`SKIPCOLS`

A comma-separated list of column names to ignore. By default, no columns are skipped.

`MAXPARSEERRORS`

The maximum global number of parsing errors to ignore. -1 means unlimited, and is the default.

`MAXINSERTERRORS`

The maximum global number of insert errors to ignore. -1 means unlimited. The default is 1000.

`ERRFILE` =

A file to store all rows that could not be imported, by default this is `import_<ks>_<table>.err` where `<ks>` is your keyspace and `<table>` is your table name.

`MAXBATCHSIZE`

The max number of rows inserted in a single batch. Defaults to 20.

`MINBATCHSIZE`

The min number of rows inserted in a single batch. Defaults to 10.

`CHUNKSIZE`

The number of rows that are passed to child worker processes from the main process at a time. Defaults to 5000.

Shared COPY Options

Options that are common to both `COPY TO` and `COPY FROM`.

`NULLVAL`

The string placeholder for null values. Defaults to `null`.

`HEADER`

For `COPY TO`, controls whether the first line in the CSV output file will contain the column names. For COPY FROM, specifies whether the first line in the CSV input file contains column names. Defaults to `false`.

`DECIMALSEP`

The character that is used as the decimal point separator. Defaults to `.`.

`THOUSANDSSEP`

The character that is used to separate thousands. Defaults to the empty string.

`BOOLSTYLE`

The string literal format for boolean values. Defaults to `True,False`.

`NUMPROCESSES`

The number of child worker processes to create for `COPY` tasks. Defaults to 16 for `COPY` tasks. However, at most (num_cores - 1) processes will be created.

`MAXATTEMPTS`

The maximum number of failed attempts to fetch a range of data (when using `COPY TO`) or insert a chunk of data (when using `COPY FROM`) before giving up. Defaults to 5.

`REPORTFREQUENCY`

How often status updates are refreshed, in seconds. Defaults to 0.25.

`RATEFILE`

An optional file to output rate statistics to. By default, statistics are not output to a file.

Escaping Quotes

Dates, IP addresses, and strings need to be enclosed in single quotation marks. To use a single quotation mark itself in a string literal, escape it using a single quotation mark.

When fetching simple text data, `cqlsh` will return an unquoted string. However, when fetching text data from complex types (collections, user-defined types, etc.) `cqlsh` will return a quoted string containing the escaped characters. For example:

Simple data

```
cqlsh> CREATE TABLE test.simple_data (id int, data text, PRIMARY KEY (id));
cqlsh> INSERT INTO test.simple_data (id, data) values(1, 'I''m fine');
cqlsh> SELECT data from test.simple_data; data
----------
 I'm fine
```

Complex data

```
cqlsh> CREATE TABLE test.complex_data (id int, data map<int, text>, PRIMARY KEY (id));
cqlsh> INSERT INTO test.complex_data (id, data) values(1, {1:'I''m fine'});
cqlsh> SELECT data from test.complex_data; data
------------------
```

```
{1: 'I''m fine'}
```