dclient documentation

Release 0.4-1-g1a0fe65

Simon Willison

Mar 08, 2024

CONTENTS

A client CLI utility for Datasette instances

CHAPTER

ONE

INSTALLATION

Install dclient using pip (or pipx:

```
pip install dclient
```

1.1 As a Datasette plugin

If you also have Datasette installed in the same environment it will register itself as a command plugin:

```
datasette install dclient
```

This means you can run any of these commands using datasette dt instead, like this:

```
datasette dc --help
datasette dc query https://latest.datasette.io/fixtures "select * from facetable limit 1"
```

You can install it into Datasette this way using:

```
datasette install dclient
```

CHAPTER

TWO

CONTENTS

2.1 Running queries

You can run SQL queries against a Datasette instance like this:

```
dclient query https://latest.datasette.io/fixtures "select * from facetable limit 1"
```

Output:

```
[
  {
    "pk": 1,
    "created": "2019-01-14 08:00:00",
    "planet_int": 1,
    "on_earth": 1,
    "state": "CA",
    "_city_id": 1,
    "_neighborhood": "Mission",
    "tags": "[\"tag1\", \"tag2\"]",
    "complex_array": "[{\"foo\": \"bar\"}]",
    "distinct_some_null": "one",
    "n": "n1"
  }
]
```

2.1.1 dclient query –help

```
Usage: dclient query [OPTIONS] URL_OR_ALIAS SQL

  Run a SQL query against a Datasette database URL

  Returns a JSON array of objects

  Example usage:

      dclient query \
        https://datasette.io/content \
        'select * from news limit 10'

```

(continues on next page)

(continued from previous page)

```
Options:
  --token TEXT   API token
  -v, --verbose  Verbose output: show HTTP request
  --help         Show this message and exit.
```

2.2 Aliases

You can assign an alias to a Datasette database using the `dclient alias` command:

```
dclient alias add content https://datasette.io/content
```

You can list aliases with `dclient alias list`:

```
$ dclient alias list
content = https://datasette.io/content
```

Once registered, you can pass an alias to commands such as `dclient query`:

```
dclient query content "select * from news limit 1"
```

2.2.1 dclient alias –help

```
Usage: dclient alias [OPTIONS] COMMAND [ARGS]...

  Manage aliases for different instances

Options:
  --help  Show this message and exit.

Commands:
  add     Add an alias
  list    List aliases
  remove  Remove an alias
```

2.2.2 dclient alias list –help

```
Usage: dclient alias list [OPTIONS]

  List aliases

  Example usage:

      dclient aliases list

Options:
```

(continues on next page)

(continued from previous page)

```
  --json  Output raw JSON
  --help  Show this message and exit.
```

2.2.3 dclient alias add –help

```
Usage: dclient alias add [OPTIONS] NAME URL

  Add an alias

  Example usage:

      dclient alias add content https://datasette.io/content

  Then:

      dclient query content 'select * from news limit 3'

Options:
  --help  Show this message and exit.
```

2.2.4 dclient alias remove –help

```
Usage: dclient alias remove [OPTIONS] NAME

  Remove an alias

  Example usage:

      dclient alias remove content

Options:
  --help  Show this message and exit.
```

2.3 Authentication

`dclient` can handle API tokens for Datasette instances that require authentication.

You can pass an API token to `query` using `-t/--token` like this:

```
dclient query https://latest.datasette.io/fixtures "select * from facetable" -t dstok_
↪mytoken
```

A more convenient way to handle this is to store tokens to be used with different URL prefixes.

2.3.1 Using stored tokens

To always use `dstok_mytoken` for any URL on the `https://latest.datasette.io/` instance you can run this:

```
dclient auth add https://latest.datasette.io/
```

Then paste in the token and hit enter when prompted to do so.

To list which URLs you have set tokens for, run the `auth list` command:

```
dclient auth list
```

To delete the token for a specific URL, run `auth remove`:

```
dclient auth remove https://latest.datasette.io/
```

2.3.2 Testing a token

The `dclient actor` command can be used to test a token, retrieving the actor that the token represents.

```
dclient actor https://latest.datasette.io/content
```

The output looks like this:

```
{
    "actor": {
        "id": "root",
        "token": "dstok"
    }
}
```

2.3.3 dclient auth –help

```
Usage: dclient auth [OPTIONS] COMMAND [ARGS]...

  Manage authentication for different instances

Options:
  --help  Show this message and exit.

Commands:
  add     Add an authentication token for an alias or URL
  list    List stored API tokens
  remove  Remove the API token for an alias or URL
```

2.3.4 dclient auth add –help

```
Usage: dclient auth add [OPTIONS] ALIAS_OR_URL

  Add an authentication token for an alias or URL

  Example usage:

      dclient auth add https://datasette.io/content

  Paste in the token when prompted.

Options:
  --token TEXT
  --help        Show this message and exit.
```

2.3.5 dclient auth list –help

```
Usage: dclient auth list [OPTIONS]

  List stored API tokens

  Example usage:

      dclient auth list

Options:
  --help  Show this message and exit.
```

2.3.6 dclient auth remove –help

```
Usage: dclient auth remove [OPTIONS] ALIAS_OR_URL

  Remove the API token for an alias or URL

  Example usage:

      dclient auth remove https://datasette.io/content

Options:
  --help  Show this message and exit.
```

2.3.7 dclient actor –help

```
Usage: dclient actor [OPTIONS] URL_OR_ALIAS

  Show the actor represented by an API token

  Example usage:

      dclient actor https://latest.datasette.io/fixtures

Options:
  --token TEXT  API token
  --help        Show this message and exit.
```

2.4 Inserting data

The `dclient insert` command can be used to insert data from a local file directly into a Datasette instance, via the Write API introduced in the Datasette 1.0 alphas.

First you'll need to *authenticate* with the instance.

To insert data from a `data.csv` file into a table called `my_table`, creating that table if it does not exist:

```
dclient insert \
  https://my-private-space.datasette.cloud/data \
  my_table data.csv --create
```

You can also pipe data into standard input:

```
curl -s 'https://api.github.com/repos/simonw/dclient/issues' | \
  dclient insert \
    https://my-private-space.datasette.cloud/data \
    issues - --create
```

2.4.1 Streaming data

`dclient insert` works for streaming data as well.

If you have a log file containing newline-delimited JSON you can tail it and send it to a Datasette instance like this:

```
tail -f log.jsonl | \
  dclient insert https://my-private-space.datasette.cloud/data logs - --nl
```

When reading from standard input (filename `-`) you are required to specify the format. In this example that's `--nl` for newline-delimited JSON. `--csv` and `--tsv` are supported for streaming as well, but `--json` is not.

In streaming mode records default to being sent to the server every 100 records or every 10 seconds, whichever comes first. You can adjust these values using the `--batch-size` and `--interval` settings. For example, here's how to send every 10 records or if 5 seconds has passed since the last time data was sent to the server:

```
tail -f log.jsonl | dclient insert \
  https://my-private-space.datasette.cloud/data logs - --nl --create \
  --batch-size 10 \
  --interval 5
```

2.4.2 Supported formats

Data can be inserted from CSV, TSV, JSON or newline-delimited JSON files.

The format of the file will be automatically detected. You can override this by using one of the following options:

- `--csv`
- `--tsv`
- `--json`
- `--nl` for newline-delimited JSON

Use `--encoding <encoding>` to specify the encoding of the file. The default is `utf-8`.

JSON

JSON files should be formatted like this:

```
[
    {
        "id": 1
        "column1": "value1",
        "column2": "value2"
    },
    {
        "id": 2
        "column1": "value1",
        "column2": "value2"
    }
]
```

Newline-delimited files like this:

```
{"id": 1, "column1": "value1", "column2": "value2"}
{"id": 2, "column1": "value1", "column2": "value2"}
```

CSV and TSV

CSV and TSV files should have a header row containing the names of the columns.

By default, `dclient` will attempt to detect the types of the different columns in the CSV and TSV files - so if a column only ever contains numeric integers it will be stored as integers in the SQLite database.

You can disable this and have every value treated as a string using `--no-detect-types`.

Other options

- `--create` - create the table if it doesn't already exist
- `--replace` - replace any rows with a matching primary key
- `--ignore` - ignore any rows with a matching existing primary key
- `--alter` - alter table to add any columns that are missing
- `--pk id` - set a primary key (for if the table is being created)

If you use `--create` a table will be created with rows to match the columns in your uploaded data - using the correctly detected types, unless you use `--no-detect-types` in which case every column will be of type `text`.

2.4.3 dclient insert –help

```
Usage: dclient insert [OPTIONS] URL_OR_ALIAS TABLE FILEPATH

  Insert data into a remote Datasette instance

  Example usage:

      dclient insert \
        https://private.datasette.cloud/data \
        mytable data.csv --pk id --create

Options:
  --csv                   Input is CSV
  --tsv                   Input is TSV
  --json                  Input is JSON
  --nl                    Input is newline-delimited JSON
  --encoding TEXT         Character encoding for CSV/TSV
  --no-detect-types       Don't detect column types for CSV/TSV
  --replace               Replace rows with a matching primary key
  --ignore                Ignore rows with a matching primary key
  --create                Create table if it does not exist
  --alter                 Alter table to add any missing columns
  --pk TEXT               Columns to use as the primary key when creating the
                          table
  --batch-size INTEGER    Send rows in batches of this size
  --interval FLOAT        Send batch at least every X seconds
  -t, --token TEXT        API token
  --silent                Don't output progress
  -v, --verbose           Verbose output: show HTTP request and response
  --help                  Show this message and exit.
```