Twarc-Cloud Documentation

Justin Littman

Apr 03, 2019

Contents:

Twarc-Cloud supports collecting twitter data from Twitter's API using Twarc running in AWS. It is designed to be super scalable and cheap and not require a server or a sysadmin.

Twarc-Cloud supports filter streams, user timelines, and searches.

CHAPTER 1

Requirements

1.1 Python 3 and pip 3

```
$ python3 -V
Python 3.7.0
$ pip3 -V
pip 18.0 from /usr/local/lib/python3.7/site-packages/pip (python 3.7)
```

It is recommended that you use virtualenv or similar to isolate your python environment.

```
$ virtualenv -p python3 ENV
$ source ENV/bin/activate
```

1.2 Terraform

Terraform is used to manager your AWS environment. Download instructions are available here.

On a Mac, you can `brew install terraform`.

```
$ terraform -v
Terraform v0.11.11
```

1.3 Twitter API developer account and app

To access the Twitter API, you need a developer account. Note that once you apply, receiving approval will take at least several days. Filling out the application as completely and accurately as possible will speed up the approval process.

Once you have a developer account, you can create an app for Twarc-Cloud. Please make sure to give your application a unique name, e.g., *twarc-cloud-justinlittman*. Twarc-Cloud will require the consumer API keys for the app.

1.4 AWS account

To run Twarc-Cloud, you need an Amazon Web Services account.

By default, you will have a root user. For security reasons, it is recommended that you create a separate user with the *AdministratorAccess* policy and use that user with Twarc-Cloud.

For either the root user or the administrator user, Twarc-Cloud will require access keys.

For the root user, in the AWS Console, this is under your account > My Security Credentials > Access keys.

For an administrator user, in the AWS Console, this is under Services > IAM > Users then select your user and then Security credentials > Access keys.

It is very important that you keep these keys secure. If they are ever compromised, you can revoke them from the AWS Console.

1.5 Twarc-Cloud

Either clone Twarc-Cloud:

```
$ git clone https://github.com/justinlittman/twarc-cloud.git
```

or download and unzip it.

Then install the requirements:

```
$ cd twarc-cloud
$ pip install -r requirements.txt
```

1.6 Honeybadger (optional)

Honeybadger provides notification of errors that occur during harvesting. It is recommended that you create an account. Note that the Solo plan is sufficient.

Once you have created an account, create a project for Twarc-Cloud. Twarc-Cloud will require the project's API key.

CHAPTER 2

Quick start

2.1 Setup

1. Install the *requirements*.
2. Configure Terraform.

```
$ cd terraform
$ cp example.terraform.tfvars terraform.tfvars
```

   and then update `terraform.tfvars` with your root or administrator AWS access keys and also select a new name for your S3 bucket.

3. Set up your AWS environment using Terraform.

```
$ terraform init
$ terraform apply
```

   Terraform will output some values that are needed in the next step.

   Note that the root or administrator AWS access keys are no longer required by Twarc-Cloud so you can remove them.

4. Configure Twarc-Cloud.

```
$ cd ..
$ cp example.twarc_cloud.ini twarc_cloud.ini
```

   and then update `twarc_cloud.ini` with the values output by Terraform from the previous step. You can also optionally provide a Honeybadger API key.

5. Acquire a Twitter API keys using Twarc.

```
$ twarc configure
```

and then provide your consumer keys. Twarc will then ask you paste a url into a browser, where you will be asked to log into your Twitter account and authorize Twarc-Cloud to access your account.

6. Make sure everything is working:

```
$ python3 twarc_cloud.py
usage: twarc_cloud.py [-h] [-V] [--debug]
                      {collection-config,collection,harvest} ...

Manage AWS resources for Twarc Cloud.

positional arguments:
  {collection-config,collection,harvest}
                        command help
    collection-config   Collection configuration-related commands.
    collection          Collection-related commands.
    harvest             Harvest-related commands.

optional arguments:
  -h, --help            show this help message and exit
  -V, --version         Show version and exit
  --debug

$ python twarc_cloud.py harvest list
No running harvests.
```

2.2 Create a user timeline collection

1. Create a collection configuration file.

```
$ python3 twarc_cloud.py collection-config template user_timeline --id=test_
↪collection
Template written to collection.json.
Add the collection before adding users to collect.
$ cat collection.json
{
  "id": "test_collection",
  "credentials": {
    "consumer_key": "<Your Twitter API consumer key>",
    "consumer_secret": "<Your Twitter API consumer secret>",
    "access_token": "<Your Twitter API access token>",
    "access_token_secret": "<Your Twitter API access token secret>"
  },
  "type": "user_timeline",
  "users": {},
  "delete_users_for": [
    "protected",
    "suspended",
    "not_found"
  ]
}
```

2. Add credentials to the collection configuration.

```
$ python3 twarc_cloud.py collection-config credentials
Added credentials to collection.json.
```

This adds the Twitter API keys that you acquired earlier with Twarc.

3. Add the collection.

```
$ python3 twarc_cloud.py collection add
Collection added.
Don't forget to start or schedule the collection.
```

This copies the collection configuration file to your S3 bucket.

4. Add users to the collection.

```
$ python3 twarc_cloud.py collection-config screennames @justin_littman @not_
↪justin_littman
Getting users ids for screen names. This may take some time ...
Added screen names to collection.json.
Following screen names where not found:
not_justin_littman
```

Twarc-cloud will notify you if any of the users cannot be found. You can also add users by user id and load them from files.

5. Update the collection.

```
$ python3 twarc_cloud.py collection-config update
Collection configuration updated.
```

6. Schedule the collection.

```
$ python3 twarc_cloud.py collection schedule test_collection "rate(7 days)"
Scheduled
```

That's it! A harvest will be performed immediately and then again every 7 days.

2.3 Download the collection

```
    $ python3 twarc_cloud.py collection download test_collection
    Collection downloaded to download/twarc-cloud/collections/test_collection

    $ find download/twarc-cloud2/collections/test_collection -type f
    download/twarc-cloud2/collections/test_collection/harvests/2019/03/09/15/35/07/
↪tweets-20190309153508.jsonl.gz
    download/twarc-cloud2/collections/test_collection/harvests/2019/03/09/15/35/07/
↪users.jsonl
    download/twarc-cloud2/collections/test_collection/harvests/2019/03/09/15/35/07/
↪manifest-sha1.txt
    download/twarc-cloud2/collections/test_collection/harvests/2019/03/09/15/35/07/
↪user_changes.json
    download/twarc-cloud2/collections/test_collection/harvests/2019/03/09/15/35/07/
↪collection.json
    download/twarc-cloud2/collections/test_collection/harvests/2019/03/09/15/35/07/
↪harvest.json
    download/twarc-cloud2/collections/test_collection/changesets/change-
↪20190309153326.json
    download/twarc-cloud2/collections/test_collection/changesets/change-
↪20190309153507.json
```

(continues on next page)

(continued from previous page)

```
    download/twarc-cloud2/collections/test_collection/changesets/change-
↪20190309153304.json
    download/twarc-cloud2/collections/test_collection/collection.json
    download/twarc-cloud2/collections/test_collection/last_harvest.json
```

Some explanation:

- `download/twarc-cloud2/collections/test_collection/harvests/2019/03/09/15/35/07/` contains the files created by the harvest.
  - `tweets-20190309153508.jsonl.gz` contains the tweets as in a newline-delimited, gzip compressed JSON format as retrieved from Twitter's API. In this case there is only one file; depending on the number of tweets and how long a harvest takes, there may be multiple files.
  - `users.jsonl` contains the users in a newline-delimited JSON format as retrieved from Twitter's API.
  - `manifest-sha1.txt` contains a SHA1 checksum for each tweet file in the harvest.
  - `user_changes.json` describes any changes that were found for users, e.g., changed screen names.
  - `collection.json` is the collection configuration file used to perform this harvest.
  - `harvest.json` contains information about the harvest such as the number of tweets collected.
- `download/twarc-cloud2/collections/test_collection/changesets/` contains changeset files that record every change made to the collection configuration.

2.4 Stop the collection

```
    $ python twarc_cloud.py collection stop test_collection
    Stopped
```

CHAPTER 3

Collection types

3.1 User timeline

User timelines are collected using the GET statuses/user_timeline method.

User timelines are always requested using the user id. User ids never change, while a screen name can change. When you add users to a user timeline collection by screen name, Twarc-Cloud will lookup the user id.

The user id and screen name are stored in the collection configuration file. For example:

```
    "users": {
        "481186914": {
          "screen_name": "justin_littman",
        },
        "12": {
          "screen_name": "jack"
        }
    }
```

The user timeline method allows retrieving up to the last 2800 tweets for a user. Twarc-Cloud collects user timelines incrementally, meaning that the first time a harvest collects a user timeline, all available tweets are collected. In subsequent harvests, only new tweets are collected. The state is stored in the collection configuration file as well:

```
    "users": {
        "481186914": {
          "screen_name": "justin_littman",
          "since_id": "1101479829856149504"
        }
    }
```

In addition to retrieving the tweets for a user, Twarc-Cloud will retrieve information about the user using the GET users/show method. These are stored in the `users.jsonl` file.

`user_changes.json` provides any changes that were found for users such as a screen name being changed or an account being deleted.

If a changed screen name is found, `collection.json` will be updated with the new screen name. The `delete_users_for` setting in `collection.json` will determine what happens if a user is deleted, suspended, or protected.

```
    "delete_users_for": [
        "protected",
        "suspended",
        "not_found"
      ]
```

If `protected` is included and a user is found to be protected, the user will be removed from `collection.json`. If `suspended` is included and a user is found to be suspended, the user will be removed. And if `not_found` is included and a user is not found, the user will be removed.

Users can be added to `collection.json` using the following commands:

- `collection-config userids`: Add a list of provided user ids.
- `collection-config userid-files`: Add a list of user ids contained in provided files.
- `collection-config screennames`: Add a list of provided screen names.
- `collection-config screenname-files`: Add a list of screen names contained in provided files.

For screen names, the @ is optional. Also, Twarc-Cloud will retrieve the user id for each screen name. This may take some time.

User timeline collections can be scheduled with the `collection schedule` command and run once with the `collection once` command.

3.2 Search

Searches are collected using the GET search/tweets.json method.

Search queries are stored in the collection configuration file. For example:

```
  "search": {
    "query": "stone OR mueller"
  }
```

To limit the number of records collected per harvest, set `max_records`. For example:

```
  "search": {
    "query": "stone OR mueller",
    "max_records": "1000"
  }
```

Twarc-Cloud collects searches incrementally, meaning that the first time a harvest collects a search, all available tweets are collected. Note that depending on the query, this initial harvest may take up to several days. In subsequent harvests, only new tweets are collected. The state is stored in the collection configuration file.

Search collections can be scheduled with the `collection schedule` command and run once with the `collection once` command.

3.3 Filter stream

Filter streams are collected using the POST statuses/filter method.

The filters for the filter stream are stored in the collection configuration file. For example:

```
{
  "track": "mueller",
}
```

Filter streams run continuously. They are turned on by the `filter start` command and stopped by the `filter stop` command.

Alternatively, if `max_records` is provided, a filter stream will stop after the specified number of tweets are collected. For example:

```
{
  "track": "mueller",
  "max_records": "100000"
}
```

Twitter's API limits keys to being used for only a single filter stream at a time. Twarc-Cloud does not enforce this limitation. If you use them for multiple filter stream collections, they will force each other to stop and mayhem will ensue.

CHAPTER 4

Twarc-Cloud commandline

4.1 General

4.1.1 Help

For any command or subcommand, `-h` will provide additional help.

4.1.2 Bucket

By default the bucket is specified in `twarc_cloud.ini`. For many commands, it can be overridden with `--bucket`.

4.2 Collection configuration commands

Collection configuration commands are for creating and updating collection configuration files.

By default, `collection.json` is the collection configuration file. For many commands, it can be overridden with `--collection-config-filepath`.

4.2.1 Create a template

```
    $ python3 twarc_cloud.py collection-config template filter
    Template written to collection.json.

    $ cat collection.json
    {
      "id": "<Identifier for collection. Should not have spaces. Must be unique for␣
↪bucket.>",
      "keys": {
```

(continues on next page)

(continued from previous page)

```
        "consumer_key": "<Your Twitter API consumer key>",
        "consumer_secret": "<Your Twitter API consumer secret>",
        "access_token": "<Your Twitter API access token>",
        "access_token_secret": "<Your Twitter API access token secret>"
      },
      "type": "filter",
      "filter": {
        "track": "<Comma separated list of terms or hashtags>",
        "follow": "<Comma separated list of user ids>",
        "max_records": "<Optional. Maximum number of records to collect per harvest."
      }
    }
```

You can now fill in the template or use other collection configuration commands to populate it.

4.2.2 Get the latest collection configuration file

To download the latest collection configuration file for an existing collection:

```
    $ python twarc_cloud.py collection-config download test_collection
    Downloaded to collection.json.
```

4.2.3 Add Twitter API keys

```
    $ python twarc_cloud.py collection-config keys
    Added keys to collection.json.
```

4.2.4 Add users

To add users by screen names provided on the commandline:

```
    $ python twarc_cloud.py collection-config screennames @justin_littman @jack @not_
↪justin_littman
    Getting users ids for screen names. This may take some time ...
    Added screen names to collection.json.
    Following screen names where not found:
    not_justin_littman
```

To add users by screen names from files:

```
    $ python twarc_cloud.py collection-config screenname-files screennames.txt
    Getting users ids for screen names. This may take some time ...
    Added screen names to collection.json.
```

To add users by user ids provided on the commandline:

```
    $ python twarc_cloud.py collection-config userids 481186914
    Added user ids to collection.json.
```

To add users by user ids from files:

```
$ python twarc_cloud.py collection-config userid-filenames userids.txt
Added user ids to collection.json.
```

4.2.5 Update

```
$ python twarc_cloud.py collection-config update
Collection configuration updated.
```

Updating the collection configuration file creates a changeset file and copies both to your S3 bucket.

4.2.6 List changes

```
$ python twarc_cloud.py collection-config changes test_collection
credentials -> consumer_key changed from None to mBbq9ruEckIngQztUir8Kn0 on 2019-
↪03-09T15:33:04.577744
credentials -> consumer_secret changed from None to␣
↪Pf28yReBUD9fpLVOsb4r5idZnKQ6xlOomBAjDfs5npFEQ6Rm on 2019-03-09T15:33:04.577744
credentials -> access_token changed from None to 4811346914-
↪5yIyfryJqfscH4dV29YVLOIzjseVsYuRzCLmwO6 on 2019-03-09T15:33:04.577744
credentials -> access_token_secret changed from None to␣
↪S51yYftbEsgdf4WMKMGendxbZO014Zvmv38Tfvc on 2019-03-09T15:33:04.577744
users -> 481186914 -> screen_name changed from None to justin_littman on 2019-03-
↪09T15:33:26.730416
keys -> consumer_key changed from None to mBbq9ruEckIngQztTHUir8Kn0 on 2019-03-
↪10T02:51:34.267589
keys -> consumer_secret changed from None to␣
↪Pf28yReBUD9Xz0pLVOsb4r5idZnKCKQ6xlOomBAjD5npFEQ6Rm on 2019-03-10T02:51:34.267589
keys -> access_token changed from None to 481186914-
↪5yIyfryJqcH4dV29YVL37BOIzjseVsYuRzCLmwO6 on 2019-03-10T02:51:34.267589
keys -> access_token_secret changed from None to␣
↪S51yY5HjfftbEs4WMKMgvGendxbZVsZO014Zvmv38Tfvc on 2019-03-10T02:51:34.267589
users -> 12 -> screen_name changed from None to jack on 2019-03-10T02:51:34.267589
```

The changes are derived from the changeset files that are created whenever a change is made to a collection configuration file.

4.3 Collection commands

Collection commands are for managing collections.

4.3.1 List collections

```
$ python3 twarc_cloud.py collection list
Collections:
candidates_for_congress
mueller
```

4.3.2 Add a collection

```
$ python3 twarc_cloud.py collection add
Collection added.
Don't forget to start or schedule the collection.
```

The default collection configuration file is `collection.json`. When added, it is copied to your S3 bucket.

4.3.3 Schedule, run once, and stop user timeline and search collections

Before running, a collection must be added.

To run once:

```
$ python3 twarc_cloud.py collection once test_collection
Started
```

To schedule:

```
$ python3 twarc_cloud.py collection schedule test_collection "rate(7 days)"
Scheduled
```

The schedule can be specified using a rate or cron expression.

To stop a scheduled collection:

```
$ python3 twarc_cloud.py collection stop test_collection
Stopped
```

And to list scheduled collections:

```
$ python3 twarc_cloud.py collection scheduled
twarc-cloud2_test_collection_schedule => rate(7 days)
```

4.3.4 Start and stop filter collections

Before starting, a collection must be added.

To start:

```
 $ python3 twarc_cloud.py collection timeline-start test_filter
 Started
```

To stop:

```
$ python3 twarc_cloud.py collection timeline-stop test_filter
Stopping ...
Stopped
```

Stopping a filter collection may take a few minutes.

4.3.5 Download a collection

```
    $ python3 twarc_cloud.py collection download test_collection
    Collection downloaded to download/twarc-cloud2/collections/test_collection
```

Files that have already been downloaded will be skpped unless `--clean` is provided.

4.4 Harvest commands

4.4.1 List running harvests

```
    $ python twarc_cloud.py harvest list
    mueller => Bucket: twarc-cloud2. Status: RUNNING
```

4.4.2 Get info on a running harvest

```
    $ python3 twarc_cloud.py harvest running mueller
    mueller => Bucket: twarc-cloud2. Harvest timestamp: 2019-03-10T02:57:27.196194.␣
↪Tweets: 1252. Files: 2 (15MB)
```

4.4.3 Get info on the last harvest

```
    $ python3 twarc_cloud.py harvest last test_collection
    test_collection => Bucket: twarc-cloud2. Harvest timestamp: 2019-03-09T15:35:07.
↪464791. Tweets: 2,140. Files: 1 (855K)
    No user changes.
```

CHAPTER 5

Twitter API keys

Accessing Twitter's API requires application keys and user keys.

5.1 Application keys

Each instance of Twarc-Cloud requires a set of application keys. You can apply for application keys at https://developer.twitter.com/en/apply-for-access. Once you have application keys, they can be provided to Twarc-Cloud as described below.

Application keys are called *consumer key* and *consumer secret*.

5.2 User keys

By authorizing a Twarc-Cloud application, a user is given a set of user keys for a Twitter account. A separate set of user keys can be issued for each Twitter account. User keys are acquired as described below.

User keys are called *access token* and *access token secret*.

5.3 Managing keys

Twarc is used to acquire and manage Twitter API keys. Twarc can manage multiple set of keys. These are stored in `~/.twarc`.

To add keys, execute `twarc configure` and follow the prompts.

To add keys to a `collection.json`, use the `collection-config keys` command. A specific key can be specified by `--profile`. For example:

```
$ python3 twarc_cloud.py collection-config keys --profile justin_littman
Added keys to collection.json.
```

CHAPTER 6

Administration

6.1 Unlocking a collection

To prevent multiple harvests being performed concurrently for a collection, a lock file (`lock.json`) is written to a collection's base directory during a harvest. Harvesters check to see if the lock file is present before beginning.

If a harvest raises a `LockedException` this indicates that a harvest is currently in process or a previous harvest exited uncleanly.

If a collection is locked because multiple harvests are attempting to run concurrently then adjust the schedule.

If a collection is locked because a previous harvest exited uncleanly, then force it be unlocked. To unlock, delete `lock.json` or execute `tweet_harvester`'s `aws unlock` command. For example:

```
$ python3 tweet_harvester.py aws unlock twarc_cloud test_collection
Unlocked
```

6.2 Removing AWS environment

Before removing your AWS environment, all of the files in your S3 bucket must be deleted. This can be done from the AWS console or AWS CLI.

Your AWS environment can then be removed with `terraform destroy`.

6.3 Logs

Logs for harvest ECS tasks are available from AWS Cloudwatch (Services > Cloudwatch > Logs) in the `twarc-cloud-container` log group.

CHAPTER 7

Design and implementation details

7.1 Design principles

- Serverless: No server to maintain or pay for when not in use.
- Use as few AWS services as possible: To reduce complexity and cost.

Thus, there is no web server, database, message queue, etc.

7.2 AWS

Harvests are run as Fargate Elastic Container Service (ECS) tasks with a single container.

- Filter stream are setup as ECS services so that they are restarted if the container fails.
- Scheduled harvests are setup as Cloudwatch Events.

S3 is used to store collections. Twarc-Cloud has its own bucket.

Twarc-Cloud is deployed in its own VPC and has its own ECS cluster.

7.3 Harvester implementation

It is important that a harvest be able to terminate cleanly, where terminate cleanly means writing all of the necessary files and uploading them to S3. In particular, it is necessary to be able to interrupt filter streams which run continuously and are setup as ECS services.

To support interrupting a harvest, the harvester runs a server which supports a `/stop` endpoint, which begins the process of stopping the harvest. It also supports a `/is_stopped` endpoint which returns if the harvest is done stopping. Thus, the process for stopping a filter stream is:

1. twarc_cloud.py invokes `/stop`.

2. The harvester begins stopping the harvest. When the harvest is stopped, the harvester does not exit. (If the harvester exited, ECS would start a new container.)
3. twarc_cloud.py polls `/is_stopped` until the harvester is stopped.
4. twarc_cloud.py stop the ECS service.
5. ECS send a terminate signal to the harvester.
6. The harvester exits.

The harvester's server is also used to provide real-time harvest information (the `/` endpoint) to twarc_cloud.py (the `harvest running` command).

CHAPTER 8

Acknowledgements

Twarc-Cloud is inspired by and borrows heavily from DocNow's Twarc and George Washington University Libraries' Social Feed Manager.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search