# newslynx-sc-shares

***Release 0.0.1***

**Sep 27, 2017**

# Contents

CHAPTER 1

# Installation

## Production

To install `newslynx-sc-shares` for an active installation of `newslynx-core`, run the following command:

```
$ newslynx sc-install https://github.com/newslynx/newslynx-sc-shares.git
```

To add `newslynx-sc-shares` all orgnaizations, run:

```
$ newslynx sc-sync
```

## Development

If you want to modify / add Sous Chefs to `newslynx-sc-shares`, do the following:

**NOTE** Will install a fresh version of `newslynx` via `pip`.

```
$ git clone https://github.com/newslynx/newslynx-sc-shares.git
$ cd newslynx-sc-shares
$ pip install --editable .
```

You should now be able to run `newslynx-sc-shares`'s Sous Chefs in development mode

```
% newslynx sc-run newslynx_sc_shares/say_my_name.yaml --myname='Brian Abelson'
```

CHAPTER 2

# Tests

Requires `nose`

```
$ make all_tests
```

CHAPTER 3

# Documentation

Documentation for `newslynx-sc-shares` is hosted on Read The Docs.

It's generated via the following steps

- converts this file (`README.md`) into a ReStructured Text file, saving it to docs/index.rst
- runs `newslynx sc-docs newslynx_sc_shares -f rst` to generate documentation for all the Sous Chefs in `newslynx-sc-shares` and saves the output to docs/sous-chefs.rst
- Builds Sphinx Documentation from these files.

CHAPTER 4

# Continuous Integration

Builds for `newslynx-sc-shares` can be found on Travis

CHAPTER 5

# Contributing

See the contributing guidelines.

# CHAPTER 6

## What's in this module ?

- README.md
  - This file
- VERSION
  - `newslynx-sc-shares`'s source-of-truth version.
- requirements.txt
  - `newslynx-sc-shares`'s python dependencies.
- MANIFEST.in
  - Specifications for which files to include in the PyPI distribution.
  - See the docs on this here.
- setup.py
  - Specification's for building `newslynx-sc-shares`'s PyPI distribution.
- .travis.yml
  - Configurations for Travis Continuous Integration
  - You must activate this project on travis-ci.org for this to run on subsequent updates.
- Makefile
  - Helpers for managing `newslynx-sc-shares`.
  - Includes:
    * `make clean`:
      · Cleans out cruft from this directory.
    * `make install`:
      · Installs `newslynx-sc-shares`. Assumes that you're in a virtual environment.
    * `make all_tests`:

        - Runs the tests.
    * `make readme`
        - Converts this file to `.rst`, including a table of contents, and saves it to docs/index.rst
    * `make sous_chef_docs`
        - Programmtically generates Sous Chef documentation by running `newslynx sc-docs newslynx_sc_shares/ --format=rst > docs/sous-chefs.rst`.
    * `make all_docs`:
        - Builds the sphinx docs for `newslynx-sc-shares` by running the above two commands.
    * `make view_docs`
        - Serves documentation at localhost:8000
    * `make register`:
        - Registers `newslynx-sc-shares` on PyPI.
    * `make distribute`:
        - Publishes a new version of `newslynx-sc-shares` to PyPI.
- CONTRIBUTING.md
- newslynx_sc_shares
  - `newslynx-sc-shares`'s source code and Sous Chef configuration files.
- docs
  - Sphnix documentation for `newslynx-sc-shares`
- tests
  - `nose` tests for `newslynx-sc-shares`

CHAPTER 7

# Contents

## Sous Chefs

**newslynx-sc-shares** provides access to the following Sous Chefs

### Timeseries Share Counts for Content Items

- Computes a timeseries of share counts for an organization's content items.
- This Sous Chef runs the python module `newslynx_sc_shares.ContentTimeseriesCounts`.
- API Slug: `share-counts-to-content-timeseries`

#### Usage

#### Standalone

Run this Sous Chef via the api, passing in arbitrary runtime options, and stream it's output.

```
$ newslynx api sous-chefs cook -d=newslynx_sc_shares/content_timeseries.yaml --
↪passthrough **options
```

Run this Sous Chef via the api, and if applicable, send it's output to bulkload.

```
$ newslynx api sous-chefs cook -d=newslynx_sc_shares/content_timeseries.yaml **options
```

Do either of the above two, but pass in a recipe file

```
$ newslynx api sous-chefs cook -d=recipe.yaml
```

### Recipes

Add this Sous Chef to your authenticated org

```
$ newslynx api sous-chefs create -d=newslynx_sc_shares/content_timeseries.yaml
```

Create a Recipe with this Sous Chef with command line options.

```
$ newslynx api recipes create sous_chef=share-counts-to-content-timeseries **options
```

Alternatively pass in a recipe file.

```
$ newslynx api recipes create sous_chef=share-counts-to-content-timeseries --
↪data=recipe.yaml
```

Save the outputted `id` of this recipe, and execute it via the API. **NOTE** This will place the recipe in a task queue.

```
$ newslynx api recipes cook id=<id>
```

Alternatively, run the Recipe, passing in arbitrary runtime options, and stream it's output: **NOTE** Will not execute the SousChef's `load` method.

```
$ newslynx api recipes cook id=<id> --passthrough **options
```

### Development

Pass runtime options to `share-counts-to-content-timeseries` and stream output. **NOTE** Will not execute the SousChef's `load` method.

```
$ newslynx sc-run newslynx_sc_shares/content_timeseries.yaml option=value1
```

Alternatively pass in a recipe file

```
$ newslynx sc-run newslynx_sc_shares/content_timeseries.yaml --recipe=recipe.yaml
```

### Options

In addition to default recipe options, `share-counts-to-content-timeseries` also accepts the following

- `days`
  - The number of days past a content item's creation date after which we will stop computing these counts.
  - Should be rendered with a `number` form.
  - Accepts inputs of type:
    * `numeric`
  - Defaults to `30`
- `content_item_types`
  - The content item types to calculate share counts for.
  - Should be rendered with a `checkbox` form.
  - Choose from:

        * video
        * article
        * slideshow
        * interactive
        * podcast
        * all
    - Accepts inputs of type:
        * string
    - Defaults to all

### Metrics

share-counts-to-content-timeseries generates the following Metrics

- facebook_shares
    - Display name: Facebook Shares
    - Type: cumulative
    - Content Levels:
        * timeseries
        * summary
        * comparison
    - Org Levels:
        * timeseries
        * summary
- facebook_likes
    - Display name: Facebook Likes
    - Type: cumulative
    - Content Levels:
        * timeseries
        * summary
        * comparison
    - Org Levels:
        * timeseries
        * summary
- facebook_comments
    - Display name: Facebook Comments
    - Type: cumulative
    - Content Levels:

        * timeseries
        * summary
        * comparison
    - Org Levels:
        * timeseries
        * summary
- linkedin_shares
    - Display name: LinkedIn Shares
    - Type: cumulative
    - Content Levels:
        * timeseries
        * summary
        * comparison
    - Org Levels:
        * timeseries
        * summary
- pinterest_shares
    - Display name: Pinterest Shares
    - Type: cumulative
    - Content Levels:
        * timeseries
        * summary
        * comparison
    - Org Levels:
        * timeseries
        * summary
- reddit_upvotes
    - Display name: Reddit UpVotes
    - Type: cumulative
    - Content Levels:
        * timeseries
        * summary
        * comparison
    - Org Levels:
        * timeseries
        * summary

- `reddit_downvotes`
  - Display name: `Reddit DownVotes`
  - Type: `cumulative`
  - Content Levels:
    - `timeseries`
    - `summary`
    - `comparison`
  - Org Levels:
    - `timeseries`
    - `summary`
- `twitter_shares`
  - Display name: `Twitter Shares`
  - Type: `cumulative`
  - Content Levels:
    - `timeseries`
    - `summary`
    - `comparison`
  - Org Levels:
    - `timeseries`
    - `summary`
- `googleplus_shares`
  - Display name: `Google Plus Shares`
  - Type: `cumulative`
  - Content Levels:
    - `timeseries`
    - `summary`
    - `comparison`
  - Org Levels:
    - `timeseries`
    - `summary`

### Share counts for arbitrary urls.

- Accepts a list of urls and returns their share counts.
- This Sous Chef runs the python module `newslynx_sc_shares.Counts`.
- API Slug: `share-counts-for-urls`

### Usage

### Standalone

Run this Sous Chef via the api, passing in arbitrary runtime options, and stream it's output.

```
$ newslynx api sous-chefs cook -d=newslynx_sc_shares/count.yaml --passthrough␣
↪**options
```

Run this Sous Chef via the api, and if applicable, send it's output to bulkload.

```
$ newslynx api sous-chefs cook -d=newslynx_sc_shares/count.yaml **options
```

Do either of the above two, but pass in a recipe file

```
$ newslynx api sous-chefs cook -d=recipe.yaml
```

### Recipes

Add this Sous Chef to your authenticated org

```
$ newslynx api sous-chefs create -d=newslynx_sc_shares/count.yaml
```

Create a Recipe with this Sous Chef with command line options.

```
$ newslynx api recipes create sous_chef=share-counts-for-urls **options
```

Alternatively pass in a recipe file.

```
$ newslynx api recipes create sous_chef=share-counts-for-urls --data=recipe.yaml
```

Save the outputted `id` of this recipe, and execute it via the API. **NOTE** This will place the recipe in a task queue.

```
$ newslynx api recipes cook id=<id>
```

Alternatively, run the Recipe, passing in arbitrary runtime options, and stream it's output: **NOTE** Will not execute the SousChef's `load` method.

```
$ newslynx api recipes cook id=<id> --passthrough **options
```

### Development

Pass runtime options to `share-counts-for-urls` and stream output. **NOTE** Will not execute the SousChef's `load` method.

```
$ newslynx sc-run newslynx_sc_shares/count.yaml option=value1
```

Alternatively pass in a recipe file

```
$ newslynx sc-run newslynx_sc_shares/count.yaml --recipe=recipe.yaml
```

### Options

In addition to default recipe options, `share-counts-for-urls` also accepts the following

- `urls`
  - The number of days past a content item's creation date after which we will stop computing these counts.
  - **Required**
  - Should be rendered with a `text` form.
  - Accepts inputs of type:
    * `string`
- `sources`
  - The sources to gather share counts from.
  - Should be rendered with a `checkbox` form.
  - Choose from:
    * `twitter`
    * `facebookfql`
    * `reddit`
    * `linkedin`
    * `facebook`
    * `pinterest`
    * `googleplus`
    * `all`
  - Accepts inputs of type:
    * `string`
  - Defaults to `all`