

---

# **sc-example**

***Release 0.0.1***

**Sep 27, 2017**



---

## Contents

---

<b>1</b>	<b>Installation</b>	<b>3</b>
1.1	Production . . . . .	3
1.2	Development . . . . .	3
<b>2</b>	<b>Tests</b>	<b>5</b>
<b>3</b>	<b>Documentation</b>	<b>7</b>
<b>4</b>	<b>Continuous Integration</b>	<b>9</b>
<b>5</b>	<b>Contributing</b>	<b>11</b>
<b>6</b>	<b>What's in this module ?</b>	<b>13</b>
<b>7</b>	<b>Contents</b>	<b>15</b>
7.1	Sous Chefs . . . . .	15



An example Sous Chef module generated by `newslynx sc-create`



# CHAPTER 1

---

## Installation

---

### Production

To install `sc-example` for an active installation of `newslynx-core`, clone it and copy into `~/.newslynx/sous-chefs`

```
$ git clone https://github.com/newslynx/sc-example.git
$ mv sc-example/ ~/.newslynx/sous-chefs/
```

Now install it within the same virtual environment as `newslynx`:

```
$ cd ~/.newslynx/sous-chefs/sc-example/
$ pip install .
```

... and if you're running `newslynx` as `sudo`

```
$ cd ~/.newslynx/sous-chefs/sc-example/
$ sudo pip install .
```

Finally, run `newslynx sc-sync` to `sc-example`'s Sous Chefs for all organizations.

```
$ newslynx sc-sync
```

### Development

If you want to modify / add Sous Chefs to `sc-example`, instal it in it's own virtual environment.

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

```
$ mkvirtualenv sc-example
$ git clone https://github.com/newslynx/sc-example.git
```

```
$ cd sc-example  
$ pip install --editable .
```

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

```
% newslinx sc sc_example/say_my_name.yaml --myname='Brian Abelson'
```



## CHAPTER 2

---

### Tests

---

Requires nose

```
$ make all_tests
```



## CHAPTER 3

---

### Documentation

---

Documentation for `sc-example` 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 sc_example -f rst` to generate documentation for all the Sous Chefs in `sc-example` and saves the output to `docs/sous-chefs.rst`
- Builds Sphinx Documentation from these files.



## CHAPTER 4

---

### Continuous Integration

---

Builds for `sc-example` can be found on [Travis](#)



## CHAPTER 5

---

### Contributing

---

See the [contributing guidelines](#).





## CHAPTER 6

---

### What's in this module ?

---

- `README.md`
  - This file
- `VERSION`
  - `sc-example`'s source-of-truth version.
- `requirements.txt`
  - `sc-example`'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 `sc-example`'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 `sc-example`.
  - Includes:
    - \* `make clean`:
      - Cleans out cruft from this directory.
    - \* `make install`:
      - Installs `sc-example`. 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
  - Programmatically generates `Sous Chef` documentation by running `newslynx sc-docs sc_example/ --format=rst > docs/sous-chefs.rst`.
- \* make all\_docs:
  - Builds the sphinx docs for `sc-example` by running the above two commands.
- \* make view\_docs
  - Serves documentation at `localhost:8000`
- \* make register:
  - Registers `sc-example` on `PyPI`.
- \* make distribute:
  - Publishes a new version of `sc-example` to `PyPI`.
- `CONTRIBUTING.md`
- `sc_example`
  - `sc-example`'s source code and `Sous Chef` configuration files.
- `docs`
  - Sphinx documentation for `sc-example`
- `tests`
  - nose tests for `sc-example`

### Sous Chefs

**sc-example** provides access to the following Sous Chefs

#### Say My Name

- Conveniently says your name.
- This Sous Chef runs the python module `sc_example.SayMyName`.
- API Slug: `say-my-name`

#### API Usage

Add this Sous Chef to your authenticated org

```
$ newslynx api sous-chefs create -d=sc_example/say_my_name.yaml
```

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

```
$ newslynx api recipes create sous_chef=say-my-name **options
```

Alternatively pass in a recipe file.

```
$ newslynx api recipes create sous_chef=say-my-name --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 `say-my-name` and stream output. **NOTE** Will not execute the SousChef's `load` method.

```
$ newslynx sc sc_example/say_my_name.yaml option=value1
```

Alternatively pass in a recipe file

```
$ newslynx sc sc_example/say_my_name.yaml --recipe=recipe.yaml
```

## Options

In addition to default recipe options, `say-my-name` also accepts the following

- `my_name`
  - Your name
  - **Required**
  - Should be rendered with a `text` form.
  - Accepts inputs of type:
    - \* `string`