growth-too-marshal Documentation

Release 0

Leo Singer

Apr 22, 2021

CONTENTS:

This is the manual for the GROWTH Target of Opportunity Marshal, or ToO Marshal for short. It is a platform that has been developed by the Global Relay of Observatories Watching Transients Happen (GROWTH) collaboration in order to coordinate follow-up observations of multimessenger transients. The ToO Marshal's responsibilities include:

1. Ingest alerts for astrophysical multimessenger transients from LIGO/Virgo, IceCube, Fermi, Swift, and other experiments.
2. Notify on-duty GROWTH astronomers when multimessenger transients occur that meet triggering criteria for science programs.
3. Plan optimal observations for a heterogeneous network of ground-based telescopes including ZTF, DECam, KPED, Gattini, and GROWTH-India.
4. Submit observations to robotic telescope queues and monitor the progress of observations.
5. Provide a central interface for vetting candidates from these facilities in combination with external data sources including the Census of the Local Universe (CLU) galaxy catalog.
6. Automatically compose GCN Circular astronomical bulletins.

Note: *Caveat emptor*: This repository has been made publicly available in the spirit of open-source software. However, much of the code is very specific to the particular facilities, instruments, and data sources that we are using, and may not be immediately generalizable.

CHAPTER

ONE

INSTALLATION

1.1 Supported Python versions

The growth-too-marshal project requires Python 3.6.

1.2 Install using Conda for development and testing

These instructions use the Miniconda Python distribution and are suitable for installing growth-too-marshal for development and testing on any Linux or macOS machine. If you already have Miniconda or Anaconda installed, then skip the first two steps.

1. Download the 64-bit Python 3 installer for Miniconda for your operating system.
   - If you are on Linux, run this command:

     ```
     $ curl https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh >
     ↪ miniconda.sh
     ```

   - If you are on macOS, run this command:

     ```
     $ curl https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-x86_64.sh␣
     ↪> miniconda.sh
     ```

2. Run the Miniconda installer:

   ```
   $ sh miniconda.sh
   ```

   Agree to the terms and conditions and install to the directory of your choice. No need to run *sudo* here if you are installing for local development. By default, it will install into `~/miniconda3`, which is just fine.

   When the installer asks, `Do you wish the installer to initialize Miniconda3 in your ~/.bash_profile ? [yes|no]`, I suggest answering *no*.

   Note: For unattended, non-interactive installation, you can add the *-b* option to automatically agree to the license terms:

   ```
   $ sh miniconda.sh -bf
   ```

3. Create a new Conda environment with this command:

```
$ ~/miniconda3/bin/conda create -ym --prefix=~/growth-too-marshal python=3.6
```

4. "Activate" the environment to add it to your current shell session:

```
$ source ~/miniconda3/bin/activate ~/growth-too-marshal
```

5. Next, we will install several pre-built Python packages using `conda` itself:

```
$ conda config --add channels anaconda
$ conda config --add channels conda-forge
$ conda install -y astropy astropy-healpix celery ephem flask flask-login flask-
↪mail flask-sqlalchemy flask-wtf flower healpy humanize h5py ipython ligo-
↪gracedb ligo-segments ligo.skymap lxml networkx pandas passlib postgresql␣
↪psycopg2 pygcn pytest pytz pyvo redis redis-py sphinx sqlalchemy sqlalchemy-
↪utils
```

6. Next, we'll check out the source code with `git`:

```
$ git clone https://github.com/growth-astro/growth-too-marshal.git ~/growth-too-
↪marshal/src
```

7. Install the marshal itself, and its remaining dependencies, using `pip`:

```
$ pip install -e ~/growth-too-marshal/src
```

The ToO Marshal is now installed. Optionally, you can run the unit tests at this point to check that everything was installed correctly:

```
$ cd ~/growth-too-marshal/src
$ python setup.py test
```

Now, proceed to the next section to configure the PostgreSQL database.

1.2.1 Configure PostgreSQL

The ToO Marshal uses a PostgreSQL database to store all of its data. Follow these instructions to initialize, start, and populate the PostgreSQL database.

Note: These instructions are suitable for using the Conda installation of PostgreSQL. Advanced users might want to adapt these instructions to their own needs by using a PostgreSQL database that is installed and managed by their package manager such as `apt-get` or `port`.

---

1. Initialize PostgreSQL by running this command:

```
$ initdb -D ~/growth-too-marshal/var/lib/postgresql
```

2. Start the PostgreSQL server:

```
$ pg_ctl -D ~/growth-too-marshal/var/lib/postgresql start
```

3. Create an empty database for the ToO Marshal:

```
$ createdb growth-too-marshal
```

4. The ToO Marshal provides a tool to create and populate its tables.

- (Recommended for development) To create the tables and populate them with some sample events and a sample user account:

```
$ growth-too db create --sample
```

- Or, to create the tables without any sample events or user accounts:

```
$ growth-too db create
```

The PostgreSQL database is now initialized, running, and populated. Proceed to the next section to start Redis.

1.2.2 Configure Redis

The ToO Marshal uses Redis as a backend for its Celery asynchronous task queue for managing background jobs. To start Redis, run this command:

```
$ redis-server --daemonize yes
```

The Redis server is now running. Proceed to the next section for application configuration.

1.2.3 Application configuration for development

There are a few last steps to complete the configuration of the ToO Marshal for development and testing.

1. The GROWTH ToO Marshal fetches user passwords from an htpasswd file. Create an htpasswd file with a password for the sample user `fritz` (as in Fritz Zwicky, of course) by running this command and entering a password:

```
$ growth-too passwd fritz
```

CHAPTER

TWO

CONFIGURATION

Listing 1: application.cfg

```
# Sample application.cfg file

# Server name for the Marshal
# Note: Without this, server name will default to localhost.
SERVER_NAME = 'skipper.caltech.edu:8081'

# Default email that receives notifications
# Note: Without this, email alerts will be broken.
EMAIL_TOO = "XXXXXXXXXXXXXXX@gmail.com"
# GMail based username and password
MAIL_USERNAME = "XXXXXXXXXXXXXX@gmail.com"
MAIL_PASSWORD = "XXXXXXXXXXXXXXXX"

# Twilio account parameters
# Note: Without this, phone and text alerts will be broken.
TWILIO_ACCOUNT_SID = 'XXXXXXXXXXXXXXXXXXXXXXXXXX'
TWILIO_AUTH_TOKEN = 'XXXXXXXXXXXXXXXXXXXXXXXXXX'
TWILIO_FROM = 'XXXXXXXXXXXXXXXXXXXX'
```

Listing 2: .netrc

```
# Sample .netrc file
# Without this, access to TAP interface will be broken (i.e.
# reference image coverage, observations, etc.).
machine irsa.ipac.caltech.edu login XXXXXXXXX password XXXXXXXXXXXX
```

CHAPTER

THREE

RUNNING THE TOO MARSHAL

Use the `growth-too` command line tool for starting and managing the ToO Marshal. The `growth-too` tool has a number of subcommands. The table below is a quick guide to the most useful commands for development and testing.

Task	Command line
Web application	
Run web app	`growth-too run --with-threads`
Run web app (debugger enabled)	`FLASK_ENV=development growth-too run --with-threads`
Database	
Initialize database	`growth-too db create`
Initialize database, populate with example events	`growth-too db create --sample`
Wipe database	`growth-too db drop`
Wipe database, then initialize again	`growth-too db recreate`
Background processing	
Run Celery worker	`growth-too celery worker --loglevel info`
Run GCN listener	`growth-too gcn`
Run periodic task scheduler	`growth-too celery beat`
Run Flower console	`growth-too celery flower`
Admin	
Enter Python console	`growth-too shell`
Add user/password	`growth-too passwd`

CHAPTER

FOUR

CONTRIBUTING

Contributors may familiarize themselves with Celery itself by going through the Flask Quickstart and First Steps with Celery tutorials.

4.1 Code style

Code should be written in the **PEP 8** style and must pass linting by Flake8. To check code style, run the following commands in the top of your source directory:

```
$ pip install flake8 pep8-naming
$ flake8 --show-source .
```

4.2 Documentation

Documentation strings should be written in the Numpydoc style.

CHAPTER

FIVE

DEPLOYMENT

For production, the GROWTH ToO Marshal is deployed using Docker.

5.1 Getting the Docker image

Pull latest Docker image Docker Hub:

```
docker pull growthastro/growth-too-marshal
```

Or build the Docker image locally:

```
docker-compose build
```

In case you need to manually push a locally built image to Docker Hub:

```
docker build -t growthastro/growth-too-marshal .
docker push growthastro/growth-too-marshal
```

5.2 Running the Marshal

Initialize the database and populate it with some sample alerts:

```
docker-compose run celery db create --sample
```

Start the ToO Marshal (navigate to `http://localhost:8081/` in your browser):

```
docker-compose up -d
```

Stop the ToO Marshal:

```
docker-compose down
```

5.3 Troubleshooting

Run an interactive PostgreSQL shell:

```
docker-compose run --rm postgres psql -h postgres -U postgres
```

Run an interactive Python shell:

```
docker-compose run --rm redis redis-cli -h redis
```

Run an interactive Flask (Python) shell:

```
docker-compose run --rm --entrypoint growth-too flask shell
```

Run an interactive Celery (Python) shell:

```
docker-compose run --rm celery celery shell
```

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

P