Django Serverless Cron Documentation

Release 0.1.3

Paul Onteri

Sep 04, 2022

CONTENTS

Contents:

CHAPTER

ONE

DJANGO-SERVERLESS-CRON

django-serverless-cron is a Django app with a simpler approach running cron jobs. This is done through exposing a HTTP endpoint to invoke the jobs that allows you to run any task without having to manage always-on infrastructure.

There is also an option to run jobs via management commands and the Django admin.

1.1 Why?

This is essentially a replacement/supplement for a traditional OS 'cron' or 'job scheduler' system:

- Serverless cron jobs no-longer a pain. Note that if you have alternatives like django-crontab or celery working well for you, good for you! You probably don't need this. However, it's okay to be curious.
- Schedule jobs to run at a frequency that is less than 1 min. (crontab is limited to 1 min)
- The machine running crontab is no longer a single point of failure.
- The problem with the above systems is that they are often configured at the operating system level, which means their configuration is probably not easily 'portable' and 'debug-able' (if you are developing on Windows, the scheduler works differently from Linux or Unix). Also can not easily be integrated into a development environment.
- Manually triggered cron jobs. Eg: via the Django Admin.
- Alternative to cron services that aren't always available on free (and sometimes paid) web hosting services.
- Easier access to cron job execution logs and monitoring execution failures.
- No need to learn crontab. Think of it as a friendlier alternative to traditional cron jobs. Simple cron job creation. No need for cron syntax, no guessing on job frequency. Easy controls.
- Designed around services like *Google Cloud Scheduler* and *Amazon EventBridge*.

1.2 Documentation

Documentation is graciously hosted at https://django-serverless-cron.readthedocs.io.

1.3 Contributions

Feel free to make pull requests and submit issues/requests. Find more detailed instructions under the *contributing* section.

Alternatively, you can leave a star on the repo to show your support.

1.4 Quickstart

1.4.1 Installation

Install Django Serverless Cron:

```
pip install django-serverless-cron
```

1.4.2 Settings

Add it to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    # ...
    'django_serverless_cron'
    # ...
)
```

Add jobs to your settings file:

```
SERVERLESS_CRONJOBS = [
    # (
    #   '1_hours',                      # frequency (seconds, minutes, hours, days,␣
↪weeks) -> in this case, every one hour
    #   'mail.jobs.send_mail_function',  # path to task/function functions -> in this␣
↪case, send_mail_function()
    #   {'kwarg1': 'foo'}               # kwargs passed to the function
    # ),
    (
        '1_days',
        'your_app.services.your_job_function',
        {'kwarg1': 'foo', 'kwarg2': 'bar', "is_bulk": True}
    ),
    (
        '1_hours',
        'mail.jobs.send_mail_function',
        {}                              # job without kwargs
```

(continues on next page)

(continued from previous page)

```
    ),
]
```

1.4.3 URL patterns

Add the jobs to your URL patterns:

```
from django.urls import path, re_path, include #add re_path and include
from django_serverless_cron import urls as django_serverless_cron_urls


urlpatterns = [
    # ...
    re_path(r'^', include(django_serverless_cron_urls)),
    #...
]
```

1.4.4 Migrate

```
python manage.py migrate
```

1.5 Running Jobs

1.5.1 In Development

Running Jobs through HTTP requests

Call the */run* path to run all jobs:

Example:

```
curl http://localhost:8000/run
```

or

```
import requests

x = requests.get('http://localhost:8000/run')
```

Running Jobs through the management command

This will run the jobs every 30 seconds:

```
python manage.py serverless_cron_run
```

You can alternatively add the *–single_run='True'* option to run the jobs just once.

1.5.2 In Production

Similar to in development, we can call the */run* path via fully managed services which are usually ridiculously cheap. Examples:

- https://cloud.google.com/scheduler -> Great feature set, easy to use, reasonable free tier & very cheap.
- https://aws.amazon.com/eventbridge
- https://azure.microsoft.com/en-gb/services/logic-apps formerly https://docs.microsoft.com/en-us/azure/scheduler/scheduler-intro
- https://cron-job.org/en/ -> Absolutely free and open-source: https://github.com/pschlan/cron-job.org
- https://www.easycron.com
- https://cronhub.io
- https://cronless.com -> Has 30 Second Cron Jobs
- https://github.com/features/actions; https://docs.github.com/en/actions/using-workflows/workflow-syntax-for-github-actions#onschedule -> eg making a HTTP request using *curl* in a step
- https://www.cronjob.de
- https://zeplo.io
- https://catalyst.zoho.com/help/cron.html
- https://www.cronjobservices.com

1.6 Related media

For more learning check out:

- https://dev.to/googlecloud/when-you-re-not-around-trigger-cloud-run-on-a-schedule-53p4 | https://youtu.be/XIwbIimM49Y
- https://aws.amazon.com/blogs/compute/using-api-destinations-with-amazon-eventbridge/
- https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/RunLambdaSchedule.html
- https://www.ibm.com/cloud/blog/how-to-schedule-rest-api-calls-on-ibm-cloud
- https://vercel.com/docs/concepts/solutions/cron-jobs
- cron-like recurring task scheduler design https://stackoverflow.com/a/3980935/10904662
- https://www.dailyhostnews.com/google-cloud-launches-fully-managed-cron-job-scheduler-for-enterprises
- Cloud Scheduler from Fireship https://www.youtube.com/watch?v=WUPEUjvSBW8

1.7 Credits

Tools used in rendering this package:

- Cookiecutter
- cookiecutter-djangopackage

CHAPTER

TWO

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

2.1 Types of Contributions

2.1.1 Report Bugs

Report bugs at https://github.com/paulonteri/django-serverless-cron/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

2.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

2.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

2.1.4 Write Documentation

Django Serverless Cron could always use more documentation, whether as part of the official Django Serverless Cron docs, in docstrings, or even on the web in blog posts, articles, and such.

2.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/paulonteri/django-serverless-cron/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

2.2 Get Started!

Ready to contribute? Here's how to set up *django-serverless-cron* for local development.

1. Fork the *django-serverless-cron* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-serverless-cron.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv django-serverless-cron
$ cd django-serverless-cron/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 django_serverless_cron tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

2.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/paulonteri/django-serverless-cron/pull_requests and make sure that the tests pass for all supported Python versions.

2.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_django_serverless_cron
```

2.5 Tests

Does the code actually work?

```
source <YOURVIRTUALENV>/bin/activate
(myenv) $ pip install tox
(myenv) $ tox
```

2.6 Development commands

```
pip install -r requirements_dev.txt
invoke -l
```

CHAPTER

THREE

CREDITS

3.1 Development Lead

- Paul Onteri <https://paulonteri.com>

3.2 Contributors

None yet. Why not be the first?

CHAPTER

FOUR

HISTORY

4.1 0.1.3 (2022-01-02)

- fix bug in running the management command
- wrap *run_all_jobs()* in a class

4.2 0.1.2 (2022-01-01)

- Fix typo in the docs

4.3 0.1.1 (2022-01-01)

Birth: First release on PyPI.

Has the ability to:

- Run jobs via the */run* path
- Run jobs via the management command *serverless_cron_run.py*