

Seçkin Tozlu
(<http://www.seckintozlu.com/en/>)

How to Run Scrapy Spiders on Cloud Using Heroku and Redis

📅 January 3, 2015 (<http://www.seckintozlu.com/1148-how-to-run-scrapy-spiders-on-the-cloud-using-heroku-and-redis.html>) / 👤 seckintozlu (<http://www.seckintozlu.com/en/author/dustyboy>) / 📁 Heroku (<http://www.seckintozlu.com/konular/heroku>), Python (<http://www.seckintozlu.com/konular/python>), Redis (<http://www.seckintozlu.com/konular/redis>), Scrapy (<http://www.seckintozlu.com/konular/scrapy>)

Overview

This tutorial aims to guide its readers install required scrapy plugins for heroku and redis support, deploy a sample spider to heroku and run it periodically (daily, hourly etc.) and store scraped items into a redis instance. We will use free heroku machine and redis add-on so you can have a running spider on the cloud for free.

I assume you already have Scrapy installed and working on your computer. If you don't have it, refer to Scrapy installation guide (<http://doc.scrapy.org/en/latest/intro/install.html>) for environment-specific installation instructions. To check whether you have scrapy or not, you can execute `scrapy version` in a terminal.

You also need Git to download the source code and to deploy to heroku.

Let's Get Ready

You will need some additional scrapy modules to complete this tutorial. Install them as below:

```
$ pip install scrapy-redis
```

```
$ pip install scrapy-heroku
```

```
$ pip install scrapyd
```

The Source Code

In this tutorial we'll use a simple scrapy spider that fetches today's featured picture and its description from Wikipedia. Go to your favorite source code directory and download the spider code from github:

```
$ git clone https://github.com/stozlu/scrapy-heroku-redis-tutorial.git
```

If you look at items.py you'll see we have a simple item class consisting of two fields:

```
1 # items.py
2 import scrapy
3
4 class FeaturedPictureItem(scrapy.Item):
5
6     url = scrapy.Field()
7     description = scrapy.Field()
```

In the spider code below, we are instantiating `FeaturedPictureItem` and assigning url and description using XPath selector.

```
1 # wikipedia_spider.py
2 import scrapy
3 from scrapy_heroku_redis_tutorial import items
4
5 class WikipediaSpider(scrapy.Spider):
6     name = "wikipedia_featured_picture"
7     allowed_domains = ["wikipedia.org"]
8     start_urls = [
9         "https://en.wikipedia.org/wiki/Main_Page (https://en.wikipedia.org/wiki/Main_Page)"
10    ]
11
12    def parse(self, response):
13
14        item = items.FeaturedPictureItem()
15        item['url'] = "https:" + response.xpath('//div[@id="mp-tfp"]//a[@class="image"]//img/
16        item['description'] = response.xpath('normalize-space(//div[@id="mp-tfp"]//p)').extra
17        yield item
```

If you are done inspecting the spider code, let's check out settings.py

```
1 # settings.py
2 BOT_NAME = 'scrapy_heroku_redis_tutorial'
3
4 SPIDER_MODULES = ['scrapy_heroku_redis_tutorial.spiders']
5 NEWSPIDER_MODULE = 'scrapy_heroku_redis_tutorial.spiders'
6
7 # Store scraped item in redis for post-processing.
8 ITEM_PIPELINES = {
9     'scrapy_redis.pipelines.RedisPipeline': 999,
10 }
11
12 # Specify the host and port to use when connecting to Redis (optional).
13 REDIS_HOST = 'localhost'
14 REDIS_PORT = 6379
15
16 # Specify the full Redis URL for connecting (optional).
17 # If set, this takes precedence over the REDIS_HOST and REDIS_PORT settings.
18 # REDIS_URL = 'redis://user:pass@hostname:9001 (redis://user:pass@hostname:9001)'
```

We should focus on a few things here. We are adding `scrapy_redis.pipelines.RedisPipeline` – which is a part of scrapy-redis package – as an item pipeline in `ITEM_PIPELINES` dictionary. This pipeline item will push scraped items to a redis list named `spider_name:items`

As you can imagine, we'll need a redis instance to store data. `REDIS_HOST`, `REDIS_PORT` and `REDIS_URL` can be used to point to your redis instance. You either need `REDIS_HOST` and `REDIS_PORT` combination or `REDIS_URL` on its own. I prefer using `REDIS_HOST` and `REDIS_PORT` for my local redis instance, and `REDIS_URL` for the cloud one.

Lastly, let's take a look at `scrapy.cfg` file:

```
1 # scrapy.cfg
2 [settings]
3 default = scrapy_heroku_redis_tutorial.settings
4
5 # Uncomment following lines to enable heroku deployment
6 # [scrapyd]
7 # application = scrapy_heroku.app.application
8
9 # [deploy]
10 # url = http://<YOUR_HEROKU_APP_NAME>(<http://<YOUR_HEROKU_APP_NAME>).herokuapp.com:80/
11 # project = scrapy_heroku_redis_tutorial
12
13 [deploy:local]
14 url = http://localhost:6800/
15 project = scrapy_heroku_redis_tutorial
```

In `scrapy.cfg` we have heroku related lines commented out, we'll get to that when we deploy this spider into heroku. In the last block we have a target definition for local deployment, which we'll use in a second.

Running the Spider Locally on Scrapyd

Before moving on to Heroku deployment, let's get familiar with the project, deploy it to a local scrapyd daemon and see it running. This is a great practice as we'll be simulating what will actually happen in heroku host, so you will have a better understanding of what's going on. However, if you believe you are already comfortable with scrapyd settings and know how to deploy/schedule a spider, or if you are feeling impatient and prefer deploying to Heroku right away, feel free to move to the next section.

Make sure you have installed scrapy-redis and scrapyd packages as mentioned in the beginning of the tutorial.

Configuring Redis

First of all, you need to have access to a redis instance to run the spider project. I'm going to use a local installation of redis just because it's super easy to install redis on Ubuntu.

```
$ sudo apt-get install redis-server
```

```
$ sudo service redis-server status|start|stop # use this to control redis server
```

Refer to <http://redis.io/download> (<http://redis.io/download>) for environment specific installation options.

Note: If you don't want to install a local redis server or if you can't install it because you are on Windows, you can use a free redis cloud solution like <https://redislabs.com/> (<https://redislabs.com/>)

If you are using a local redis server you don't need to change anything in `settings.py` file as it's already pointing to `localhost:6379`. But if you have a redis instance somewhere in the cloud, you need to uncomment `REDIS_URL` line in `settings.py` and provide your redis endpoint.

Deploying the Project

Now open two terminal windows, we'll use the first one to start scrapyd daemon:

```
$ scrapyd # command to start scrapyd
2015-01-03 04:37:39+0000 [-] Log opened.
2015-01-03 04:37:39+0000 [-] twistd 14.0.2 (/usr/bin/python 2.7.6) starting up.
2015-01-03 04:37:39+0000 [-] reactor class: twisted.internet.epollreactor.EPollReactor.
2015-01-03 04:37:39+0000 [-] Site starting on 6800
2015-01-03 04:37:39+0000 [-] Starting factory ;
2015-01-03 04:37:39+0000 [Launcher] Scrapyd 1.0.1 started: max_proc=4, runner='scrapyd.runner'
```

Once scrapyd is started, you can visit <http://localhost:6800/> (<http://localhost:6800/>) to see a simple web interface where you can monitor running spiders, scraped items, logs etc.

Leave scrapyd running on the first terminal. Switch to second terminal window run the following commands:

```
$ cd scrapy-heroku-redis-tutorial/ # where you cloned the project initially
$ scrapyd-deploy -l
local http://localhost:6800/
```

You are seeing only one target named **"local"** because heroku target is commented out in **scrapy.cfg**

Now execute the following command which should deploy the project to local target in the same window:

```
$ scrapy deploy local -p scrapy_heroku_redis_tutorial
Packing version 1420252899
Deploying to project "scrapy_heroku_redis_tutorial" in http://localhost:6800/addversion.json
Server response (200):
{"status": "ok", "project": "scrapy_heroku_redis_tutorial", "version": "1420252899", "spiders": 1}
```

Note: In the output above, if you see a warning about not having service_identity module installed, you could get rid of it by running 'pip install service_identity'

Running the Spider

Deploying a project as we did just now doesn't mean you are running a spider. That is a separate task we could accomplish using the scrapyd JSON API. To schedule a spider run the following command:

```
$ curl http://localhost:6800/schedule.json -d project=scrapy_heroku_redis_tutorial -d  
spider=wikipedia_featured_picture  
{“status”: “ok”, “jobid”: “250a113e92f911e489e508002751e440”}
```

This means you have successfully scheduled the spider. Now you can go into web interface <http://localhost:6800/> (<http://localhost:6800/>) and check out the scraped item.

If you want to verify that you have stored this item in Redis, you can use `redis-cli` which was installed as part of `redis-server`:

```
$ redis-cli  
> lrange wikipedia_featured_picture:items 0 -1 # spider name: wikipedia_featured_picture  
1) {“url”:  
  \”https://upload.wikimedia.org/wikipedia/commons/thumb/0/04/Anthocharis_cardamines_female_MichaD.jpg/360px-  
  Anthocharis_cardamines_female_MichaD.jpg\”, \”description”:\”A female Anthocharis cardamines, a species of  
  butterfly in the family Pieridae. Found through Europe and into Asia, this butterfly prefers damp grassy areas.\”}
```

Running the Spider on Heroku

First, we'll need to create a heroku account and download the client tool.

Create a Heroku Account and Download Toolbelt

If you don't have a Heroku account, go to <https://www.heroku.com/> (<https://www.heroku.com/>) and sign up for a free account. Then, download the heroku toolbelt from <https://toolbelt.heroku.com/> (<https://toolbelt.heroku.com/>), this is a client to communicate with your heroku host.

After you are done, simply type `"heroku login"` in a terminal window, enter email and password you used when you created your heroku account. You should be all set if you see "Authentication successful" message.

```
$ heroku login  
Enter your Heroku credentials.  
Email: youremail@yourhost.com  
Password (typing will be hidden):  
Authentication successful.
```

Create a Heroku App

Creating a heroku application is very simple when you use the heroku client. Just do the following:

```
$ cd scrapy-heroku-redis-tutorial
$ heroku create
Creating rocky-island-3510... done, stack is cedar-14
https://rocky-island-3510.herokuapp.com/ | https://git.heroku.com/rocky-island-3510.git
Git remote heroku added
```

Important: The output of above command will include a randomly-created application name. In this case it's "rocky-island-3510", make sure to save your application name as we'll need it very soon.

Note that creating a heroku app doesn't mean you have deployed your project. We'll do that in a few minutes.

We must update scrapy.cfg file so it has the correct application name. Also, uncomment heroku related configuration lines to enable heroku deployment. Your scrapy.cfg should look very similar to what you see below:

```
1  # updated scrapy.cfg
2  [settings]
3  default = scrapy_heroku_redis_tutorial.settings
4
5  # Uncomment following lines to enable heroku deployment
6  [scrapy]
7  application = scrapy_heroku.app.application
8
9  [deploy]
10 url = http://rocky-island-3510.herokuapp.com:80/ # replace rocky-island-3510 with your app n
11 project = scrapy_heroku_redis_tutorial
12
13 [deploy:local]
14 url = http://localhost:6800/
15 project = scrapy_heroku_redis_tutorial
```

Configure Postgres

scrapy-heroku module requires a Postgres database available on your heroku host. You can easily install a postgres database add-on like this:

```
$ heroku addons:add heroku-postgresql
Adding heroku-postgresql on rocky-island-3510... done, v7 (free)
Attached as HEROKU_POSTGRESQL_COPPER_URL
Database has been created and is available
! This database is empty. If upgrading, you can transfer
! data from another database with pgbackups:restore.
Use heroku addons:docs heroku-postgresql to view documentation.
```

This will create a DATABASE_URL environment variable pointing to your new postgres database. You can see that value if you run the following command:

```
$ heroku config:get DATABASE_URL
```

Configure Redis

The next step is to configure a redis instance so we can store scraped items on the cloud. You can do that by adding a free redis add-on to your heroku account. Execute the following command:

```
$ heroku addons:add rediscloud -app rocky-island-3510 # replace 'rocky-island-3510' with your app name
Adding rediscloud on rocky-island-3510... done, v3 (free)
Use heroku addons:docs rediscloud to view documentation.
```

Heroku will require you to enter billing information even if you use free tier of this add-on. So the above command will fail if you haven't provided a billing method for your heroku account. If you don't want to enter credit card information on heroku, you can create a free account on <https://redislabs.com/> (https://redislabs.com/) and use this instance instead of creating a heroku add-on.

The next step is to retrieve redis url and enter it in settings.py. If you installed heroku add-on, you can get it easily as below. Otherwise you need to go into your dashboard in redis provider's website to retrieve it.

```
heroku config:get REDIS_CLOUD_URL
```

Take the output of the above command – which is your redis url – and assign it to REDIS_URL field in your settings.py file. Don't forget to uncomment that line too. At this point, your settings.py should look very similar to this:

```
1  # updated settings.py
2  BOT_NAME = 'scrapy_heroku_redis_tutorial'
3
4  SPIDER_MODULES = ['scrapy_heroku_redis_tutorial.spiders']
5  NEWSPIDER_MODULE = 'scrapy_heroku_redis_tutorial.spiders'
6
7  # Store scraped item in redis for post-processing.
8  ITEM_PIPELINES = {
9      'scrapy_redis.pipelines.RedisPipeline': 999,
10 }
11
12 # Specify the host and port to use when connecting to Redis (optional).
13 # REDIS_HOST = 'localhost' # you don't have to comment this line but it helps avoid confusion
14 # REDIS_PORT = 6379        # you don't have to comment this line but it helps avoid confusion
15
16 # Specify the full Redis URL for connecting (optional).
17 # If set, this takes precedence over the REDIS_HOST and REDIS_PORT settings.
18 REDIS_URL = 'redis://rediscloud:<your_password_goes_here>;@<your_hostname_goes_here>:<redis_p
```

Deploying to Heroku

Deploying a project to heroku is as simple as pushing a piece of code to remote repository.

```
$ cd scrapy-heroku-redis-tutorial
$ git add .
$ git commit -m "update" # you must commit your changes, otherwise heroku won't pick them up
$ git push heroku master
```

This will push our sample project to heroku remote, then heroku will detect it and start building it. It will read **requirements.txt** which is provided in the root directory of the sample project and install those modules. Then it will start your project by executing what you provided in **Procfile**, which is **scrapyd** for our case. Once the process is complete, you can check heroku logs to make sure things started up fine.

```
$ heroku logs
```

```
heroku[web.1]: Starting process with command `scrapyd`
```

```
app[web.1]: 2015-01-03 07:43:09+0000 [-] reactor class: twisted.internet.epollreactor.EPollReactor.
```

```
app[web.1]: 2015-01-03 07:43:09+0000 [-] twistd 14.0.2 (/app/.heroku/python/bin/python 2.7.9) starting up.
```

```
app[web.1]: 2015-01-03 07:43:09+0000 [-] Site starting on 28332
```

```
app[web.1]: 2015-01-03 07:43:09+0000 [Launcher] Scrapy 1.0.1 started: max_proc=16, runner='scrapyd.runner'
```

```
app[web.1]: 2015-01-03 07:43:09+0000 [-] Starting factory
```

```
app[web.1]: 2015-01-03 07:43:09+0000 [-] Log opened.
```

```
heroku[web.1]: State changed from starting to up
```

Now run the following command to ensure that at least one instance of the app is running:

```
$ heroku ps:scale web=1
```

```
Scaling dynos... done, now running web at 1:1X.
```

If everything went fine, now you should be able to see scrapyd web interface by running **'heroku open'** or navigating to <http://your-app-name.herokuapp.com> (<http://your-app-name.herokuapp.com>)

Running the Spider

Again, we deployed the spider project but haven't scheduled a spider yet. Now let's do that by running the curl command against heroku. Don't forget to replace application name:

```
$ heroku run curl http://rocky-island-3510.herokuapp.com:80/schedule.json -d project=default -d
```

```
spider=wikipedia_featured_picture
```

```
Running `curl http://rocky-island-3510.herokuapp.com:80/schedule.json -d project=default -d
```

```
spider=wikipedia_featured_picture` attached to terminal... up, run.1813
```

```
{"status": "ok", "jobid": "d1936de4932411e49435ae39031036c8"}
```

You have successfully scheduled your spider! Go to scrapyd web interface and make sure you were able to scrape the item correctly. For redis verification, we'll use redis-cli again, but this time we'll be connecting to a remote host:

```
$ redis-cli -h redis_host_name -p redis_port -a redis_password
```

```
> lrange wikipedia_featured_picture:items 0 -1
```

```
1) {"url\
```



```
\\"https://upload.wikimedia.org/wikipedia/commons/thumb/0/04/Anthocharis_cardamines_female_MichaD.jpg/360px-Anthocharis_cardamines_female_MichaD.jpg\\",\\"description\\":\\"A female Anthocharis cardamines, a species of butterfly in the family Pieridae. Found through Europe and into Asia, this butterfly prefers damp grassy areas.\\"}"
```

Running Spiders Periodically on Heroku

If you'd like to run a spider periodically on heroku, you can use heroku scheduler add-on. It gives you 750 free dyno-hour (<https://devcenter.heroku.com/articles/usage-and-billing#750-free-dyno-hours-per-app>) per app each month, which will cover you all the time when you're using a free 1X dyno. But in case heroku changes this allotment, you might want to double check before using this add-on. Let's install:

```
$ heroku addons:add scheduler
```

Adding scheduler on rocky-island-3510... done, v9 (free)

This add-on consumes dyno hours, which could impact your monthly bill.

To learn more: http://devcenter.heroku.com/addons_with_dyno_hour_usage to manage scheduled jobs run:

```
heroku addons:open scheduler
```

Use `heroku addons:docs scheduler` to view documentation.


In order to schedule a task, go to <https://scheduler.heroku.com/dashboard>

(<https://scheduler.heroku.com/dashboard>) or run `'heroku addons:open scheduler'` which opens the same web interface. On the dashboard you can set scheduler settings, for the task field enter the same curl command you used in the previous section. It probably makes sense to set frequency to daily, since we're fetching today's featured picture from wikipedia which changes daily.

Like this:

Loading...

([/#facebook](#)) ([/#twitter](#)) ([/#linkedin](#)) ([/#pocket](#)) ([/#email](#))

 (<https://www.addtoany.com/share?url=http%3A%2F%2Fwww.seckintozlu.com%2F1148-how-to-run-scrapy-spiders-on-the-cloud-using-heroku-and-redis.html&title=How%20to%20Run%20Scrapy%20Spiders%20on%20Cloud%20Using%20Heroku%20and%20Redis>)

 heroku (<http://www.seckintozlu.com/etiketler/heroku-2>), heroku addon

(<http://www.seckintozlu.com/etiketler/heroku-addon>), python (<http://www.seckintozlu.com/etiketler/python-2>), redis

(<http://www.seckintozlu.com/etiketler/redis-2>), rediscloud (<http://www.seckintozlu.com/etiketler/rediscloud>), scrapy

(<http://www.seckintozlu.com/etiketler/scrapy-2>), scrapy-heroku (<http://www.seckintozlu.com/etiketler/scrapy-heroku>),

scrapy-redis (<http://www.seckintozlu.com/etiketler/scrapy-redis>), scrapyd

(<http://www.seckintozlu.com/etiketler/scrapyd>), spider (<http://www.seckintozlu.com/etiketler/spider>)

« Three Levels of API Testing

(<http://www.seckintozlu.com/1020-three-levels-of-api-testing.html>)

Leave a Reply

Enter your comment here...



(l



(l

h

c

I

F

U

(l

h

c

T

(l

t

S

(l

s

jk

I

F

U

(l

h

c



(l



(l

h

c

I

F

U

(l

h

c

T

(l

t

S

(l

s

jk

I

F

U

(l

h

c



(l



(l

h

c

I

F

U

(l

h

c

T

(l

t

S

(l

s

jk

I

F

U

(l

h

c



(l



(l

h

c

I

F

U

(l

h

c

T

(l

t

S

(l

s

jk

I

F

U

(l

h

c



(l



(l

h

c

I

F

U

(l

h

c

T

(l

t

S

(l

s

jk

I

F

U

(l

h

c



(l



(l

h

c

I

F

U

(l

h

c

T

(l

t

S

(l

s

j

I

F

U

(l

h

c

Proudly powered by WordPress (<https://wordpress.org/>) | Theme: Blogi by CantoThemes (<https://www.cantothemes.com/>).

u