bit.ly/atlanta-bot

Please go to “Downloads” and download the “Initial Files”

THINKFUL

Building A Twitter Bot With Basic Python

February 2017

Bot

- Our bot makes a search on twitter for a given phrase.
- Receives a list of tweets.
- Replies to each of those tweets with a predefined message.

Uses of the bot

- MARKETING
    - Target topics
    - Target competence's followers
- COMMUNITY MANAGEMENT
    - Answer repetitive questions
    - Interact with users
- JUST FOR FUN

Goals

- Build a real project.
- Set up our computer to work with Python.
- Learn some cool Python concepts.
- No previous knowledge required.

Not goals

- This is not a step-by-step intro to Python.
- Also not a review of Twitter's API.

Installing pip

Windows

- In Powershell, go to the "windows" directory of the downloaded files.

- Run: $ python get-pip.py

MAC

```
$ sudo easy_install pip
```

LINUX

```
$ sudo apt-get install python-pip
```

Twitter

Twitter API

- Provides programmatic access to read and write Twitter data
- Docs: https://dev.twitter.com/rest/public
    - Important to check the limits and restrictions
- We will use tweepy to interact with Twitter's API

```
$ pip install tweepy
```

If it fails

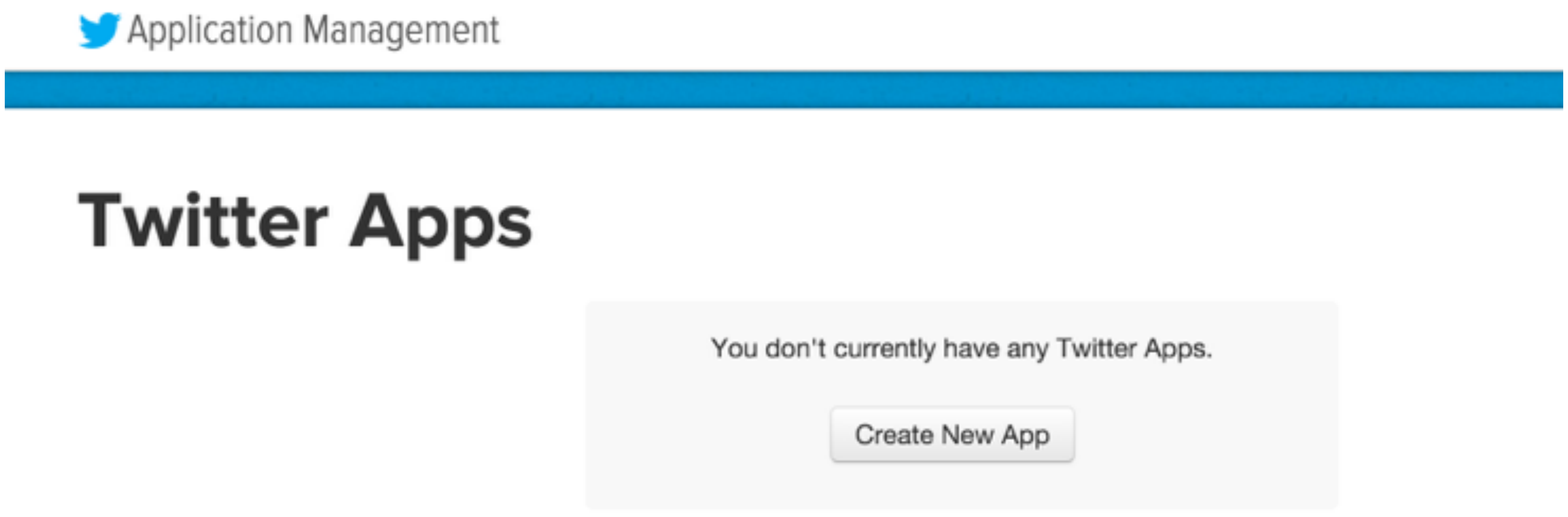
```
$ sudo pip install tweepy --ignore-installed
```

Create a Twitter account

If you don’t have one already

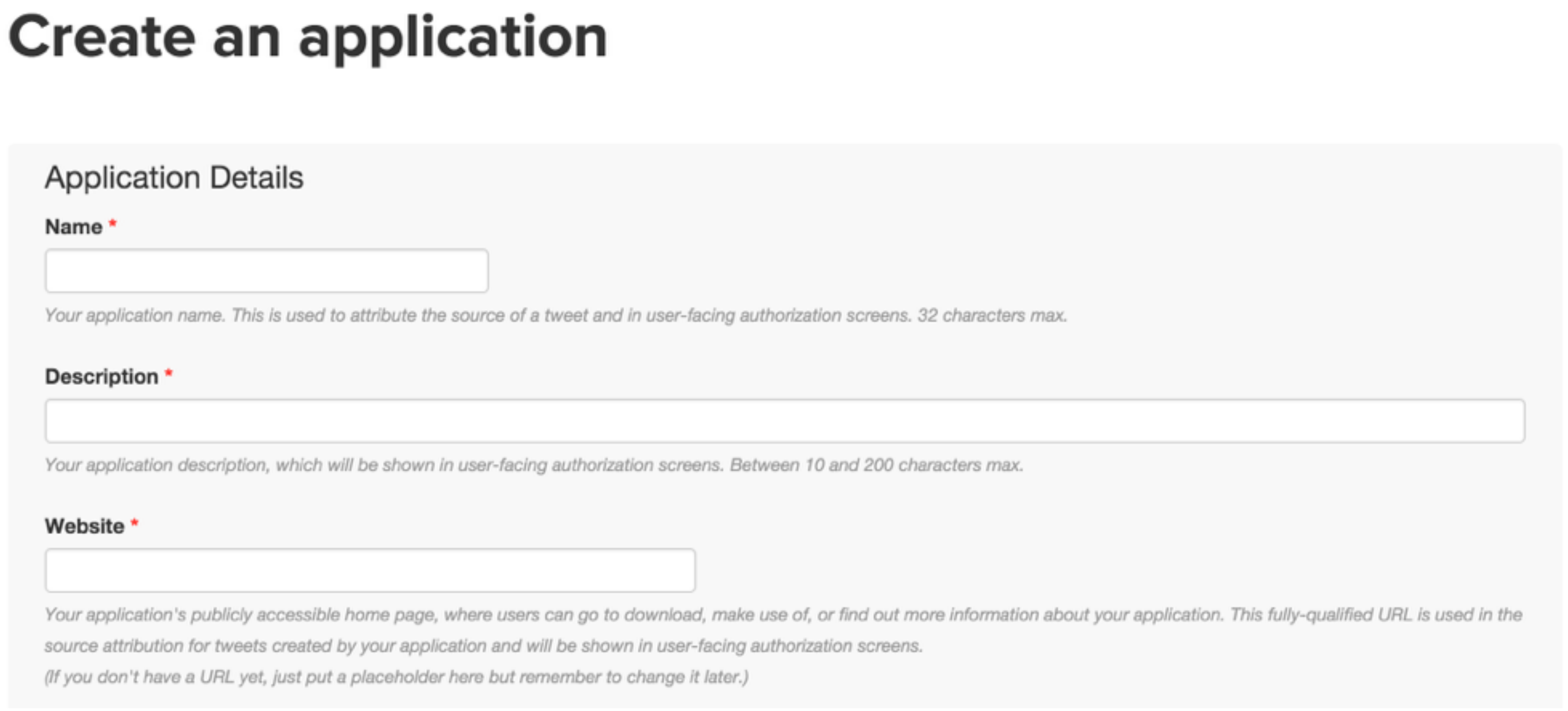
IMPORTANT: You must provide your phone number!
Otherwise you won’t be able to create an app

- When you have your twitter account go to:
    - apps.twitter.com
- Click on "Create new app"


Application Management

Twitter Apps

You don't currently have any Twitter Apps.


Create an application

Application Details

Name *

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Website field must start with: **http://**

- Go to the “**Keys and access tokens**” tab.
- There you will have your “**Consumer key**” and “**Consumer secret**” that we will use in a few moments.
- You need to “**Create your access token**”
    - At the bottom of the screen

BOT

goo.gl/HoOLH4

```
twitter-bot/
    keys.py
    bot.py
```

```
# keys.py

# replace the words in caps with the keys that
# we saw before on apps.twitter.com
keys = {
    ‘consumer_key’: 'CONSUMER_KEY’,
    ‘consumer_secret’: ‘CONSUMER_SECRET’,
    ‘access_token’: ‘ACCESS_TOKEN’,
    ‘access_token_secret’: ‘ACCESS_TOKEN_SECRET’,
}
```

Dictionary

- An unordered set of ‘key: value’ pairs.
- **Curly braces:** {}.
- **Access a value:** keys[‘consumer_key’]
- **Set a single value:** keys[‘extra_value’] = ‘hey’

```
# bot.py

import tweepy

# from our keys module (keys.py), import the keys
from keys import keys


auth = tweepy.OAuthHandler(keys['consumer_key'], keys['consumer_secret'])
auth.set_access_token(keys['access_token'], keys['access_token_secret'])
api = tweepy.API(auth)

query = '"sad alot”'

tweet_list = api.search(
    q=query,     # frase to search
    count=20,    # number of tweets to return
    lang=“en”    # language to search (optional)
)

for tweet in tweet_list:
    screen_name = tweet.user.screen_name

    message = ".@{username} {message}".format(
        username=screen_name,
        message=‘Alot confused, a lot not understand feelings’
    )

    try:
        api.update_status(
            status=message,
            in_reply_to_status_id=tweet.id
        )
        print message

    except tweepy.TweepError as e:
        print e.message
```

```python
# bot.py

import tweepy

# from our keys module (keys.py), import the keys
from keys import keys


# we create the api object
auth = tweepy.OAuthHandler(
   keys[‘consumer_key'],
   keys['consumer_secret'])
auth.set_access_token(
   keys['access_token'],
   keys['access_token_secret'])

api = tweepy.API(auth)
```

Double quotes inside single quotes
to search for that exact phrase

```
query = ‘"sad alot”’

tweet_list = api.search(
  q=query,    # phrase to search
  count=20,   # number of tweets to return
  lang='en'   # language to search
)
```

THINKFUL

Functions

- A function is a block of organized, reusable code.
- Functions have to be defined.
- Functions can return a value.

```python
def search(q, count, lang):
  # do something
  return value
```

```
for tweet in tweet_list:
   # do something

   # don’t copy this just yet
   screen_name = tweet.user.screen_name

   message = '@{username} {message}'.format(
       username=screen_name,
       message='Alot confused, Alot not understand feelings'
   )

   try:
       api.update_status(
           status=message,
           in_reply_to_status_id=tweet.id
       )
       print message

   except tweepy.TweepError as e:
       print e.message[0]['code']
       print e.args[0][0]['code']
```

THINKFUL

For loop

- Used when you have a piece of code which you want to repeat **n** number of times.
- For each **tweet** in **tweet_list**, do something.

Indentation

- To indicate a block of code, you must **indent** each line by the same amount.
- For each **tweet** in **tweet_list**, do something.

```python
for tweet in tweet_list:
  screen_name = tweet.user.screen_name

  message = '@{username} {message}'.format(
    username=screen_name,
    message='Alot confused, Alot not understand feelings'
  )
```

String format

- Replacement fields are delimited by braces {}
- Returns a copy of the string where each replacement field is replaced with the string value of the corresponding argument

```
# same indentation as before

try:
  api.update_status(
    status=message,
    in_reply_to_status_id=tweet.id
  )
  print message

except tweepy.TweepError as error:
  print error.message
```

Try/Except

- When a Python script encounters a situation that it cannot cope with, it raises an exception.
- If you have some suspicious code that may raise an exception, you can defend your program by placing the suspicious code in a **try:** block.
- Also include an **except:** statement, followed by a block of code which handles the problem

```
$ python bot.py
```

Let’s build onto what we have

[•••]

```
api = tweepy.API(auth)

~~query = ‘"sad alot”’~~

ALOT_HERD = [
    #[‘"exact string to search"', 'tweet response')
    [‘"alot of bacon"', 'You just summoned Alot of bacon!’],
    [‘"alot of beer"', 'You just summoned Alot of beer!'],
    [‘"alot of fire"', 'You just summoned Alot of fire!'],
    [‘"alot of mist"', 'You just summoned Alot of mist!'],
    [‘"alot of money"', 'You just summoned Alot of money!'],
]

for alot in ALOT_HERD:
    query = alot[0]

    ~~tweet_list = api.search(q=query, count=20, lang="en")~~
    tweet_list = api.search(q=query, count=5, lang=“en")
```

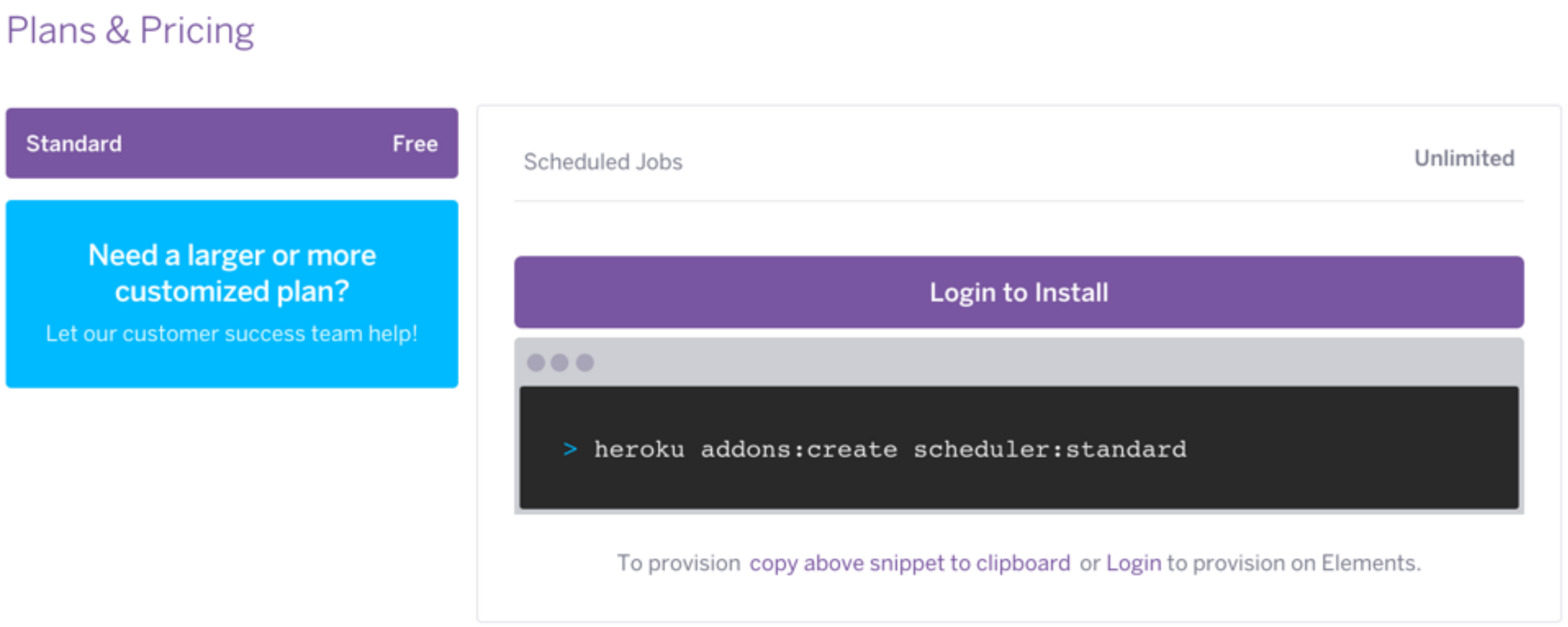
[•••]

Lists

- An ordered set of values.
- list1 = ['physics', 'chemistry', 1997, 2000]
- list1[0] -> ‘physics'
- Indexes start at 0

```python
for alot in ALOT_HERD:
    query = alot[0]

    tweet_list = api.search(q=query, count=5, lang="en")

    for tweet in tweet_list:
        screen_name = tweet.user.screen_name

        message = ".@{username} {message}".format(
           username=screen_name,
           message=alot[1]
        )

       try:
          api.update_status(
             status=message,
             in_reply_to_status_id=tweet.id
          )
          print message

       except tweepy.TweepError as e:
          print e.message[0]['code']
          print e.args[0][0]['code']
```

heroku



Plans & Pricing

Standard	Free

Need a larger or more customized plan?

Let our customer success team help!

Scheduled Jobs	Unlimited

```
> heroku addons:create scheduler:standard
```

To provision copy above snippet to clipboard or Login to provision on Elements.

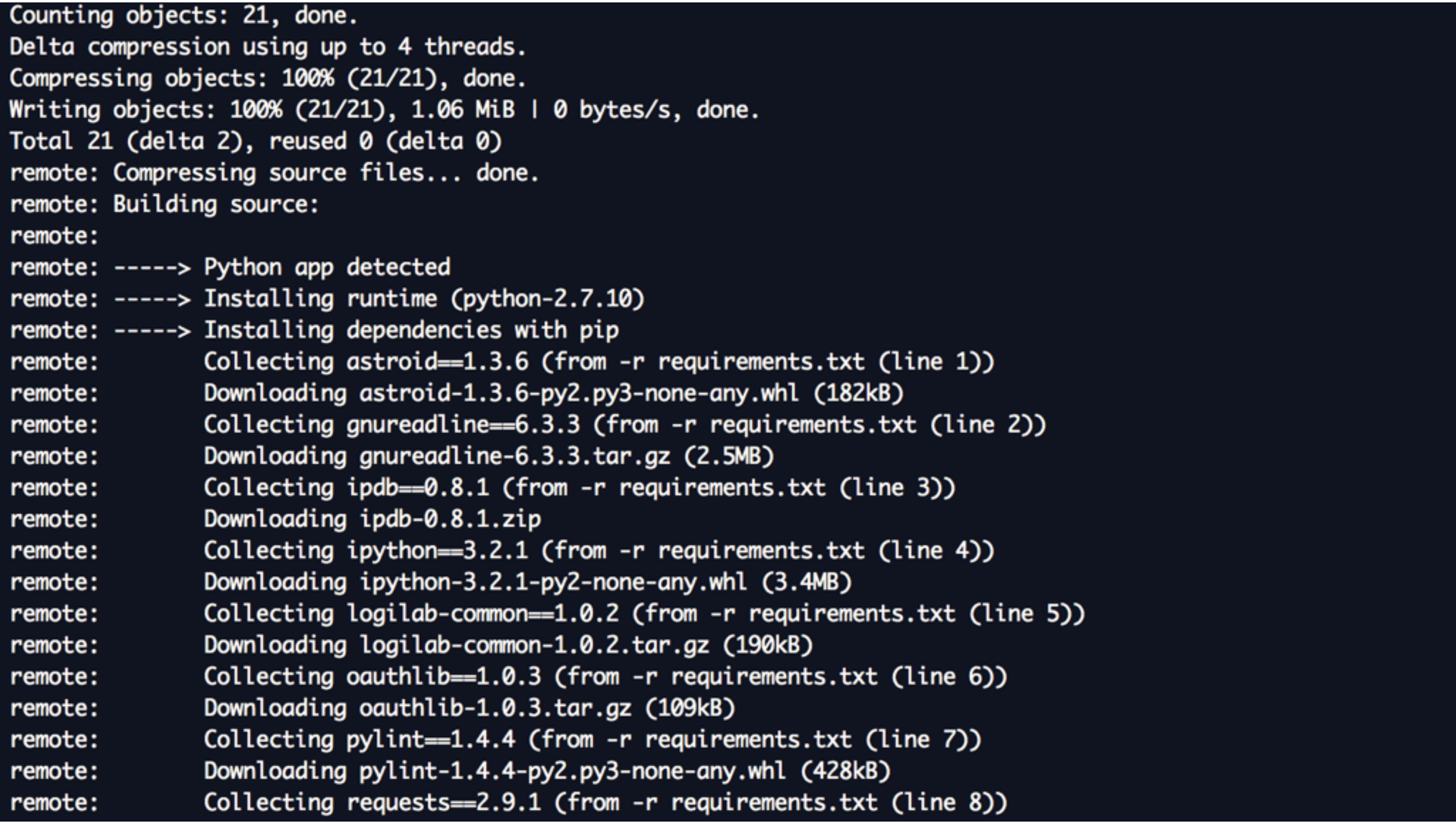
CREATE ACCOUNT

- Create a free account on heroku.com
- Click on your email address (up and to the left of the screen)
- Click on “Manage account”
- Click “Billing”
- Introduce credit card data (won’t be used)

https://toolbelt.heroku.com/

Upload to heroku

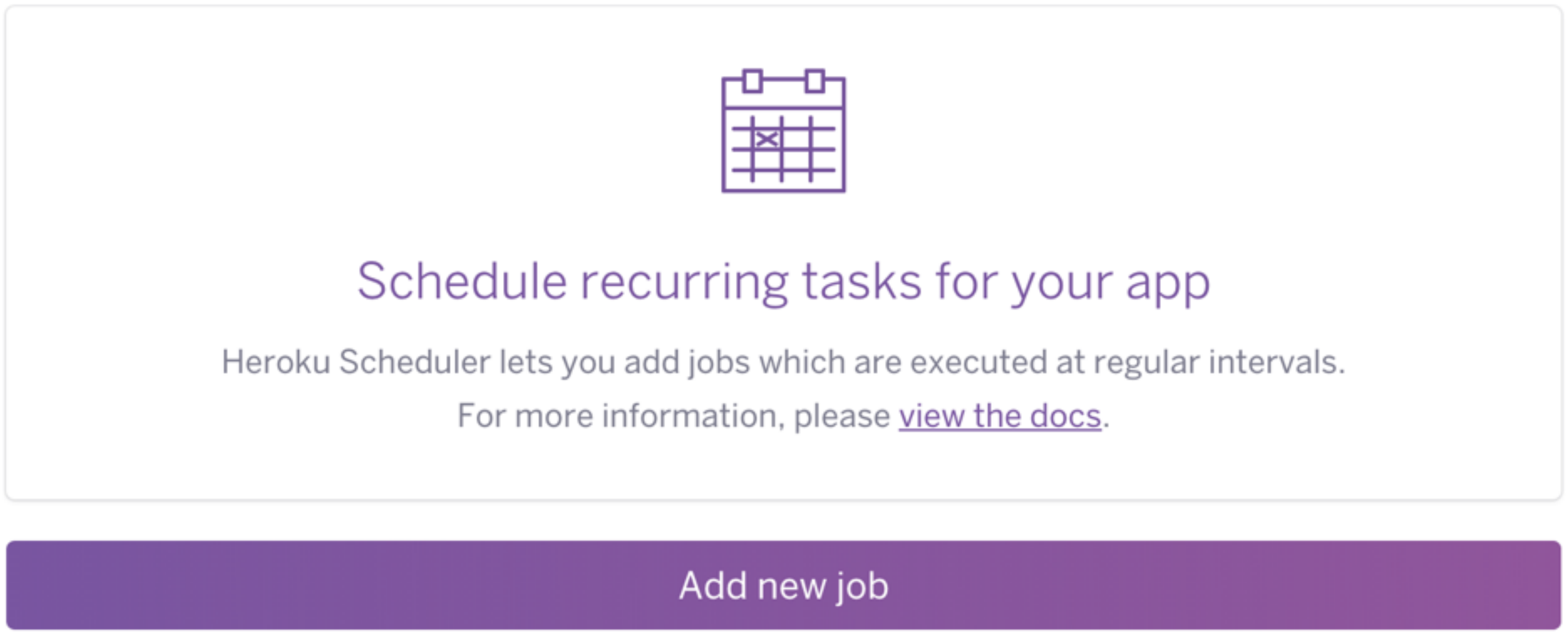
- $ git init
- $ git add .
- $ git commit -m “Add all files”
- $ heroku create —stack cedar
- $ git push heroku master



```
Counting objects: 21, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (21/21), done.
Writing objects: 100% (21/21), 1.06 MiB | 0 bytes/s, done.
Total 21 (delta 2), reused 0 (delta 0)
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Python app detected
remote: -----> Installing runtime (python-2.7.10)
remote: -----> Installing dependencies with pip
remote:        Collecting astroid==1.3.6 (from -r requirements.txt (line 1))
remote:        Downloading astroid-1.3.6-py2.py3-none-any.whl (182kB)
remote:        Collecting gnureadline==6.3.3 (from -r requirements.txt (line 2))
remote:        Downloading gnureadline-6.3.3.tar.gz (2.5MB)
remote:        Collecting ipdb==0.8.1 (from -r requirements.txt (line 3))
remote:        Downloading ipdb-0.8.1.zip
remote:        Collecting ipython==3.2.1 (from -r requirements.txt (line 4))
remote:        Downloading ipython-3.2.1-py2-none-any.whl (3.4MB)
remote:        Collecting logilab-common==1.0.2 (from -r requirements.txt (line 5))
remote:        Downloading logilab-common-1.0.2.tar.gz (190kB)
remote:        Collecting oauthlib==1.0.3 (from -r requirements.txt (line 6))
remote:        Downloading oauthlib-1.0.3.tar.gz (109kB)
remote:        Collecting pylint==1.4.4 (from -r requirements.txt (line 7))
remote:        Downloading pylint-1.4.4-py2.py3-none-any.whl (428kB)
remote:        Collecting requests==2.9.1 (from -r requirements.txt (line 8))
```

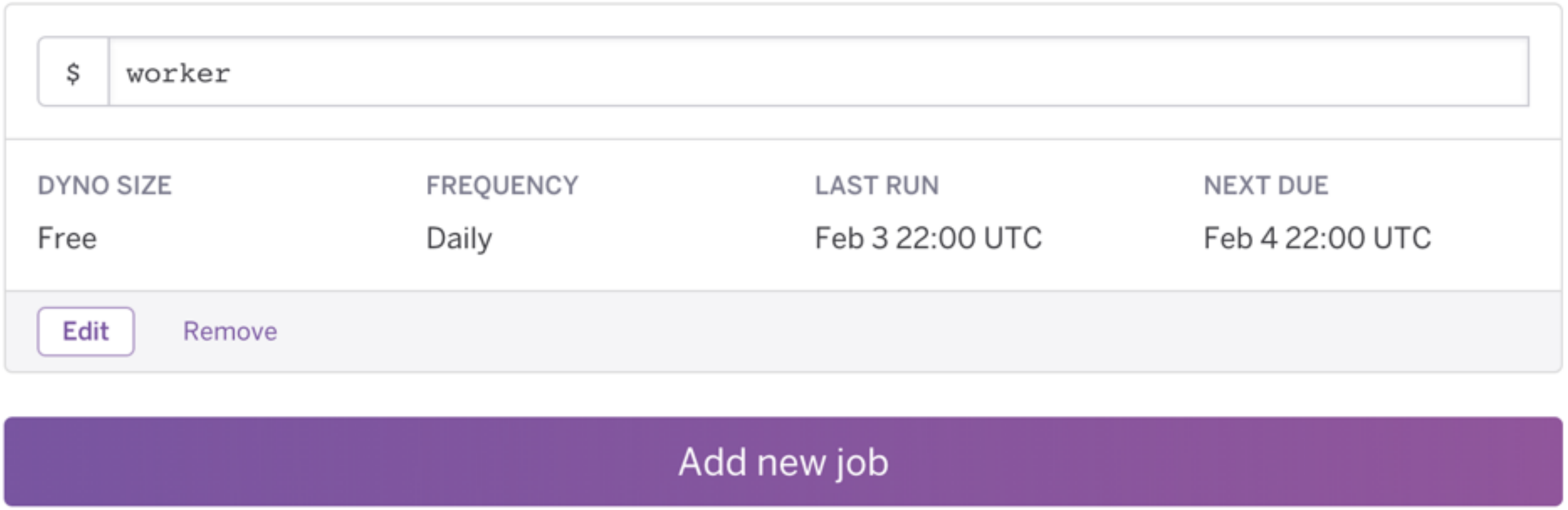
Add scheduler

- $ heroku run worker
- $ heroku addons:add scheduler:standard
    - Will say again that it’s paid, but it’s really free
- $ heroku addons:open scheduler



Schedule recurring tasks for your app

Heroku Scheduler lets you add jobs which are executed at regular intervals.
For more information, please view the docs.



```
$ worker
```

DYNO SIZE	FREQUENCY	LAST RUN	NEXT DUE
Free	Daily	Feb 3 22:00 UTC	Feb 4 22:00 UTC