niqdev / **packtpub-crawler**

Download your daily free Packt Publishing eBook https://www.packtpub.com/packt/offers/free-learning

#packtpub #free-ebook #google-drive #onedrive #ifttt #firebase #heroku #docker

224 commits | **2** branches | **11** releases | **9** contributors | MIT

Branch: **master** | **New pull request** | **Create new file** | **Upload files** | **Find file** | **Clone or download**

niqdev update readme — Latest commit `f8066e2` on Oct 7, 2017

config	Add pushover notification	11 months ago
dev	Fixed tests, resolved conflicts	a year ago
script	Merge pull request #71 from juzim/manual-claim-on-error	9 months ago
.dockerignore	update docker support	a year ago
.gitignore	update readme	5 months ago
Dockerfile	update docker scheduler	a year ago
LICENSE	change license from CC to MIT	2 years ago
Procfile	draft heroku scheduler	2 years ago
README.md	update readme	5 months ago
requirements.txt	Fixed conficts	9 months ago

README.md

packtpub-crawler

Download FREE eBook every day from www.packtpub.com

This crawler automates the following step:

- access to private account
- claim the daily free eBook and weekly Newsletter
- parse title, description and useful information
- download favorite format *.pdf .epub .mobi*
- download source code and book cover
- upload files to Google Drive, OneDrive or via scp
- store data on Firebase
- notify via Gmail, IFTTT, Join or Pushover (on success and errors)
- schedule daily job on Heroku or with Docker

Default command

```
# upload pdf to googledrive, store data and notify via email
python script/spider.py -c config/prod.cfg -u googledrive -s firebase -n gmail
```

Other options

```
# download all format
python script/spider.py --config config/prod.cfg --all

# download only one format: pdf|epub|mobi
python script/spider.py --config config/prod.cfg --type pdf

# download also additional material: source code (if exists) and book cover
python script/spider.py --config config/prod.cfg -t pdf --extras
# equivalent (default is pdf)
python script/spider.py -c config/prod.cfg -e

# download and then upload to Google Drive (given the download url anyone can download it)
python script/spider.py -c config/prod.cfg -t epub --upload googledrive
python script/spider.py --config config/prod.cfg --all --extras --upload googledrive

# download and then upload to OneDrive (given the download url anyone can download it)
python script/spider.py -c config/prod.cfg -t epub --upload onedrive
python script/spider.py --config config/prod.cfg --all --extras --upload onedrive

# download and notify: gmail|ifttt|join|pushover
python script/spider.py -c config/prod.cfg --notify gmail

# only claim book (no downloads):
python script/spider.py -c config/prod.cfg --notify gmail --claimOnly
```

Basic setup

Before you start you should

- Verify that your currently installed version of Python is **2.x** with `python --version`
- Clone the repository `git clone https://github.com/niqdev/packtpub-crawler.git`
- Install all the dependencies `pip install -r requirements.txt` (see also virtualenv)
- Create a config file `cp config/prod_example.cfg config/prod.cfg`
- Change your Packtpub credentials in the config file

```
[credential]
credential.email=PACKTPUB_EMAIL
credential.password=PACKTPUB_PASSWORD
```

Now you should be able to claim and download your first eBook

```
python script/spider.py --config config/prod.cfg
```

Google Drive

From the documentation, Google Drive API requires OAuth2.0 for authentication, so to upload files you should:

- Go to Google APIs Console and create a new Google Drive project named **PacktpubDrive**
- On *API manager* > *Overview* menu
  - Enable Google Drive API
- On *API manager* > *Credentials* menu
  - In *OAuth consent screen* tab set **PacktpubDrive** as the product name shown to users
  - In *Credentials* tab create credentials of type *OAuth client ID* and choose Application type *Other* named **PacktpubDriveCredentials**
- Click *Download JSON* and save the file `config/client_secrets.json`
- Change your Google Drive credentials in the config file

```
[googledrive]
...
googledrive.client_secrets=config/client_secrets.json
googledrive.gmail=GOOGLE_DRIVE@gmail.com
```

Now you should be able to upload your eBook to Google Drive

```
python script/spider.py --config config/prod.cfg --upload googledrive
```

Only the first time you will be prompted to login in a browser which has javascript enabled (no text-based browser) to generate `config/auth_token.json` . You should also copy and paste in the config the *FOLDER_ID*, otherwise every time a new folder with the same name will be created.

```
[googledrive]
...
googledrive.default_folder=packtpub
googledrive.upload_folder=FOLDER_ID
```

Documentation: OAuth, Quickstart, example and permissions

OneDrive

From the documentation, OneDrive API requires OAuth2.0 for authentication, so to upload files you should:

- Go to the Microsoft Application Registration Portal.
- When prompted, sign in with your Microsoft account credentials.
- Find **My applications** and click **Add an app**.
- Enter **PacktpubDrive** as the app's name and click **Create application**.
- Scroll to the bottom of the page and check the **Live SDK support** box.
- Change your OneDrive credentials in the config file
  - Copy your **Application Id** into the config file to **onedrive.client_id**
  - Click **Generate New Password** and copy the password shown into the config file to **onedrive.client_secret**
  - Click **Add Platform** and select **Web**
  - Enter **http://localhost:8080/** as the **Redirect URL**
  - Click **Save** at the bottom of the page

```
[onedrive]
...
onedrive.client_id=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
onedrive.client_secret=XxXxXxXxXxXxXxXxXxXxXxX
```

Now you should be able to upload your eBook to OneDrive

```
python script/spider.py --config config/prod.cfg --upload onedrive
```

Only the first time you will be prompted to login in a browser which has javascript enabled (no text-based browser) to generate `config/session.onedrive.pickle` .

```
[onedrive]
...
onedrive.folder=packtpub
```

Documentation: Registration, Python API

Scp

To upload your eBook via `scp` on a remote server update the configs

```
[scp]
scp.host=SCP_HOST
scp.user=SCP_USER
scp.password=SCP_PASSWORD
scp.path=SCP_UPLOAD_PATH
```

Now you should be able to upload your eBook

```
python script/spider.py --config config/prod.cfg --upload scp
```

Note:

- the destination folder `scp.path` on the remote server must exists in advance
- the option `--upload scp` is incompatible with `--store` and `--notify`

Firebase

Create a new Firebase project, copy the database secret from your settings

```
https://console.firebase.google.com/project/PROJECT_NAME/settings/database
```

and update the configs

```
[firebase]
firebase.database_secret=DATABASE_SECRET
firebase.url=https://PROJECT_NAME.firebaseio.com
```

Now you should be able to store your eBook details on Firebase

```
python script/spider.py --config config/prod.cfg --upload googledrive --store firebase
```

Gmail notification

To *send* a notification via email using Gmail you should:

- Allow "less secure apps" and "DisplayUnlockCaptcha" on your account
- Troubleshoot sign-in problems and examples
- Change your Gmail credentials in the config file

```
[gmail]
...
gmail.username=EMAIL_USERNAME@gmail.com
gmail.password=EMAIL_PASSWORD
gmail.from=FROM_EMAIL@gmail.com
gmail.to=TO_EMAIL_1@gmail.com,TO_EMAIL_2@gmail.com
```

Now you should be able to notify your accounts

```
python script/spider.py --config config/prod.cfg --notify gmail
```

IFTTT notification

- Get an account on IFTTT
- Go to your Maker settings and activate the channel
- Create a new applet using the Maker service with the trigger "Receive a web request" and the event name "packtpub-crawler"
- Change your IFTTT key in the config file

```
[ifttt]
ifttt.event_name=packtpub-crawler
ifttt.key=IFTTT_MAKER_KEY
```

Now you should be able to trigger the applet

```
python script/spider.py --config config/prod.cfg --notify ifttt
```

Value mappings:

- value1: title
- value2: description
- value3: landing page URL

Join notification

- Get the Join Chrome extension and/or App
- You can find your device ids here
- (Optional) You can use multiple devices or groups (group.all, group.android, group.chrome, group.windows10, group.phone, group.tablet, group.pc) separated by comma
- Change your Join credentials in the config file

```
[join]
join.device_ids=DEVICE_IDS_COMMA_SEPARATED_OR_GROUP_NAME
join.api_key=API_KEY
```

Now you should be able to trigger the event

```
python script/spider.py --config config/prod.cfg --notify join
```

Pushover notification

- Get your USER_KEY
- Create a new application
- (Optional) Add an icon
- Change your pushover credentials in the config file

```
[pushover]
pushover.user_key=PUSHOVER_USER_KEY
pushover.api_key=PUSHOVER_API_KEY
```

Heroku

Create a new branch

```
git checkout -b heroku-scheduler
```

Update the `.gitignore` and commit your changes

```
# remove
config/prod.cfg
config/client_secrets.json
config/auth_token.json
# add
dev/
config/dev.cfg
config/prod_example.cfg
```

Create, config and deploy the scheduler

```
heroku login
# create a new app
heroku create APP_NAME --region eu
# or if you already have an existing app
heroku git:remote -a APP_NAME

# deploy your app
git push -u heroku heroku-scheduler:master
heroku ps:scale clock=1

# useful commands
heroku ps
heroku logs --ps clock.1
heroku logs --tail
heroku run bash
```

Update `script/scheduler.py` with your own preferences.

More info about Heroku Scheduler, Clock Processes, Add-on and APScheduler

Docker

Build your image

```
docker build -t niqdev/packtpub-crawler:2.4.0 .
```

Run manually

```
docker run \
  --rm \
  --name my-packtpub-crawler \
  niqdev/packtpub-crawler:2.4.0 \
  python script/spider.py --config config/prod.cfg
```

Run scheduled crawler in background

```
docker run \
  --detach \
  --name my-packtpub-crawler \
  niqdev/packtpub-crawler:2.4.0

# useful commands
docker exec -i -t my-packtpub-crawler bash
docker logs -f my-packtpub-crawler
```

Alternatively you can pull from Docker Hub this fork

```
docker pull kuchy/packtpub-crawler
```

Cron job

Add this to your crontab to run the job daily at 9 AM:

```
crontab -e

00 09 * * * cd PATH_TO_PROJECT/packtpub-crawler && /usr/bin/python script/spider.py --config
config/prod.cfg >> /tmp/packtpub.log 2>&1
```

Systemd service

Create two files in /etc/systemd/system:

1. packtpub-crawler.service

```
[Unit]
Description=run packtpub-crawler

[Service]
User=USER_THAT_SHOULD_RUN_THE_SCRIPT
ExecStart=/usr/bin/python2.7 PATH_TO_PROJECT/packtpub-crawler/script/spider.py -c config/prod.cfg

[Install]
WantedBy=multi-user.target
```

2. packtpub-crawler.timer

```
[Unit]
Description=Runs packtpub-crawler every day at 7

[Timer]
OnBootSec=10min
OnActiveSec=1s
OnCalendar=*-*-* 07:00:00
Unit=packtpub_crawler.service
Persistent=true

[Install]
WantedBy=multi-user.target
```

Enable the script with `sudo systemctl enable packtpub_crawler.timer` . You can test the service with `sudo systemctl start packtpub_crawler.timer` and see the output with `sudo journalctl -u packtpub_crawler.service -f` .

Newsletter

The script downloads also the free ebooks from the weekly packtpub newsletter. The URL is generated by a Google Apps Script which parses all the mails. You can get the code here, if you want to see the actual script, please clone the spreadsheet and go to `Tools > Script editor...` .

To use your own source, modify in the config

```
url.bookFromNewsletter=https://goo.gl/kUciut
```

The URL should point to a file containing only the URL (no semicolons, HTML, JSON, etc).

You can also clone the spreadsheet to use your own Gmail account. Subscribe to the newsletter (on the bottom of the page) and create a filter to tag your mails accordingly.

Troubleshooting

- ImportError: No module named paramiko

Install paramiko with `sudo -H pip install paramiko --ignore-installed`

- Failed building wheel for cryptography

Install missing dependencies as described here

virtualenv

```
# install pip + setuptools
curl https://bootstrap.pypa.io/get-pip.py | python -

# upgrade pip
pip install -U pip

# install virtualenv globally
sudo pip install virtualenv

# create virtualenv
virtualenv env

# activate virtualenv
source env/bin/activate

# verify virtualenv
which python
python --version

# deactivate virtualenv
deactivate
```

Development (only for spidering)

Run a simple static server with

```
node dev/server.js
```

and test the crawler with

```
python script/spider.py --dev --config config/dev.cfg --all
```

Disclaimer

This project is just a Proof of Concept and not intended for any illegal usage. I'm not responsible for any damage or abuse, use it at your own risk.