

Crawler Instance Base

Instance details

Type	t3.nano
AWS Account ID	820871936241
VPC ID	vpc-0ddb8146e071c3c33 (carro-data-vpc)
Generated AMI ID	ami-041e912b310020b7e

Setup

Install system dependencies

- 1 `sudo apt update`
- 2 `sudo apt-get install make`

Clone the repo

- 1 `git clone git@github.com:TrustyCars/carro-crawler-service-py.git`
- 2 `git checkout develop`
- 3 `git pull`

Install prerequisites


- 1 `cd ~/carro-crawler-service-py`
- 2 `make install-conda`
- 3 `make install-sys`
- 4 `make pip-install`

Activate anaconda base environment


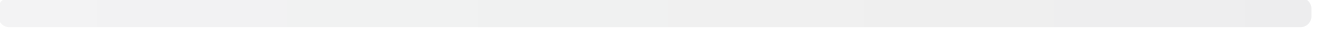

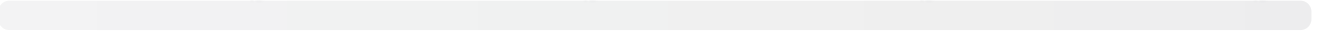

```
1 ~/miniconda3/bin/conda init
```

To take effect, logout and login to the server back

Crawler instances

 Subpage table of contents

Unable to preview

-  
-  
-  

Appendix

Alembic and DB migration for new columns

Path: `/home/ubuntu/carro-crawler-service-py/migrations/alembic.ini`

```
1 [DEFAULT]
2
3 sqlalchemy.url = postgresql+psycopg2://postgres:EZf%%jvT4HYtB*J6bhGSF@data-
  carro-crawler-01.cyij8crywzth.ap-southeast-1.rds.amazonaws.com/carro_wholesale
```

Generate Migration File

```
1 conda activate scrapyd
2 cd /path/to/carro-crawler-service-py/migrations
3 alembic revision -m 'changes message'
```

Open and update the migration file accordingly.

Don't forget to put the migration file to git repo.

Migrate database

```
1 conda activate scrapyd
```

```
2 cd /path/to/carro-crawler-service-py/migrations
3 alembic upgrade head
```

Common config values

Following values should be set in the `.env` file. `.env` is located under `/path/to/carro-crawler-service-py/ccspy`.

```
1 DB_CONNECTION=postgresql
2 DB_HOST=
3 DB_PORT=5432
4 DB_DATABASE=
5 DB_SCHEMA=
6 DB_USERNAME=postgres
7 DB_PASSWORD=
8 DB_INTEGRATION=true
9
10 AWS_KEY=
11 AWS_SECRET=
12 AWS_BUCKET=carro-crawler-service-testing
13 AWS_REGION=ap-southeast-1
14 S3_ENABLED=true
15
16 REDIS_HOST=
17 REDIS_PORT=6379
18 REDIS_DB=0
19 REDIS_USERNAME
20 REDIS_PASSWORD=GBi+IEaRKTuBPbLzRAJ5F/E0Qo2GnW1LEd28kmZaMW4=
21 REDIS_SOCKET_TIMEOUT
22 REDIS_SOCKET_CONNECT_TIMEOUT
23 REDIS_CLIENT_NAME=crawler-name
24 REDIS_INTEGRATION=True
25
26 # For Scrape Lead
27 OLX_SCRAPES_LEAD_SELLER_LOCATION=Jakarta D.K.I,Jawa Barat,Banten,Jawa Tengah,Jawa
28 CARMUDI_SCRAPES_LEAD_SELLER_LOCATION=DKI Jakarta,Jawa Barat,Banten,Jawa Timur
29 MOBIL123_SCRAPES_LEAD_SELLER_LOCATION=DKI Jakarta,Jawa Barat,Banten,Jawa Timur
30 CINTAMOBIL_SCRAPES_LEAD_SELLER_LOCATION=DKI Jarkata,Jawa Barat,Banten,Jawa Timur
31
32 ROBOTSTXT_OBEY=false
33 FAKE_USERAGENT_ENABLED=true
34 PROXY_ENABLED=true
35 PROXY_LIST=/path/to/carro-crawler-service-py/scrapy/proxies/proxy_list.txt
36 DONT_REMOVE_PROXY=true
```

Testing Connection

Run the following command to test db, aws s3, and redis connections.

```
1 cd /path/to/carro-crawler-service-py
2 python -m ccspy.cmd testconn
```

List crawler names

Crawler names can be listed using the following command:

```
1 cd /path/to/carro-crawler-service-py
2 conda activate scrapyd
3 scrapy list
```

Troubleshooting a crawler

Running a crawler manually

Login to a crawler instance. Enter the `scripts` directory of the project.

```
1 cd /path/to/carro-crawler-service-py/scripts
```

Then, run a crawler with its name using bash.

```
1 bash run_spider.sh <crawler-name> >> /tmp/output.log
2
3 # for example, the following command runs a olx crawler
4 # bash run_spider.sh id.co.olx >> /tmp/output.log
```

Please see [List crawler names](#) section to view the crawler names. *Downloaded pages are stored in the s3 and their appropriate URLs are stored in redis.*

Output can be viewed via `/tmp/output.log`.

```
1 tail -f /tmp/output.log
```

To exit, press `Ctrl+C`.

Running a parser manually

Run a parser with crawler name using bash.

```
1 cd /path/to/carro-crawler-service-py/scripts
2
3 bash run_parser.sh <crawler-name> >> /tmp/parsed.log
4
5 # for example, the following command runs a parser for downloaded olx pages
6 # bash run_parser.sh id.co.olx >> /tmp/parsed.log
```

Useful commands

Python environment is created during the setup. The environment name is `scrapyd`. To activate the environment, please run as follows:

```
1 conda activate scrapyd
```

Download a page and save to s3

Use `scrapy fetch` command to download and save a page to s3.

```
1 # scrapy fetch --spider=<crawler-name> "<url>"
2 scrapy fetch --spider=com.carousell.sg "https://www.carousell.sg/p/hyundai-avant
```

Parse live page which is not downloaded yet

Use `scrapy parse` command to parse a live page.

```
1 # scrapy parse --spider=<crawler-name> --callback=parse_debug "<url>"
2 scrapy parse --spider=com.carousell.sg --callback=parse_debug "https://www.carou
```

Parse downloaded page (which is stored in the s3 bucket)

Copy the s3 object URL. A page from Carousell will be used as an example.

```
1 # format: crawler-name/webpage.html
```

```
2 com.carousell.sg/0000e1360031df8973ce9600f1f845c62546b76789e4fb1abdd768b7534ca10
```

We have s3 object URL, but original URL is required to parse the page. To get the original URL, use `url-origin` command.

```
1 # python -m ccspy.cmd url-origin <crawler-name> <s3object-id>
2 python -m ccspy.cmd url-origin com.carousell.sg 0000e1360031df8973ce9600f1f845c6
3
4 # Output: https://www.carousell.sg/p/honda-freed-1-5-g-7-seater-honda-sensing-a-
```

Then, we can parse the downloaded page using `s3parse` command.

```
1 # python -m ccspy.cmd s3parse <crawler-name> "<url>"
2 python -m ccspy.cmd s3parse com.carousell.sg "https://www.carousell.sg/p/honda-f
```