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

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

Clone the repo

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

Install prerequisites

```
cd ~/carro-crawler-service-py
make install-conda
make install-sys
make pip-install
```

Activate anaconda base environment

```
~/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`

```
[DEFAULT]

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

Generate Migration File

```
conda activate scrapyd
cd /path/to/carro-crawler-service-py/migrations
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

```
conda activate scrapyd
```

```
cd /path/to/carro-crawler-service-py/migrations
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`.

```
DB_CONNECTION=postgresql
DB_HOST=
DB_PORT=5432
DB_DATABASE=
DB_SCHEMA=
DB_USERNAME=postgres
DB_PASSWORD=
DB_INTEGRATION=true

AWS_KEY=
AWS_SECRET=
AWS_BUCKET=carro-crawler-service-testing
AWS_REGION=ap-southeast-1
S3_ENABLED=true

REDIS_HOST=
REDIS_PORT=6379
REDIS_DB=0
REDIS_USERNAME
REDIS_PASSWORD=GBi+IEaRKTuBPbLzRAJ5F/E0Qo2GnW1LEd28kmZaMW4=
REDIS_SOCKET_TIMEOUT
REDIS_SOCKET_CONNECT_TIMEOUT
REDIS_CLIENT_NAME=crawler-name
REDIS_INTEGRATION=True

# For Scrape Lead
OLX_SCRAPE_LEAD_SELLER_LOCATION=Jakarta D.K.I,Jawa Barat,Banten,Jawa Tengah,Jawa
CARMUDI_SCRAPE_LEAD_SELLER_LOCATION=DKI Jakarta,Jawa Barat,Banten,Jawa Timur
MOBIL123_SCRAPE_LEAD_LOCATION=DKI Jakarta,Jawa Barat,Banten,Jawa Timur
CINTAMOBIL_SCRAPE_LEAD_LOCATION=DKI Jarkata,Jawa Barat,Banten,Jawa Timur

ROBOTSTXT_OBEY=false
FAKE_USERAGENT_ENABLED=true
PROXY_ENABLED=true
PROXY_LIST=/path/to/carro-crawler-service-py/scrapyd/proxies/proxy_list.txt
DONT_REMOVE_PROXY=true
```

Testing Connection

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

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

List crawler names

Crawler names can be listed using the following command:

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

Troubleshooting a crawler

Running a crawler manually

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

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

Then, run a crawler with its name using bash.

```
bash run_spider.sh <crawler-name> >> /tmp/output.log

# for example, the following command runs a olx crawler
# 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`.

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

> To exit, press `Ctrl+C`.

Running a parser manually

Run a parser with crawler name using bash.

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

bash run_parser.sh <crawler-name> >> /tmp/parsed.log

# for example, the following command runs a parser for downloaded olx pages
# 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:

```
conda activate scrapyd
```

Download a page and save to s3

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

```
# scrapy fetch --spider=<crawler-name> "<url>"
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.

```
# scrapy parse --spider=<crawler-name> --callback=parse_debug "<url>"
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.

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

```
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.

```
# python -m ccspy.cmd url-origin <crawler-name> <s3object-id>
python -m ccspy.cmd url-origin com.carousell.sg 0000e1360031df8973ce9600f1f845c6

# 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.

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