Algorithmic Trading System Using Alpaca Markets API

By
Raymond Fries

Purpose

No one is going to care more about your finances than you!

Thanks to technological advances and advances in the financial industry it is now easier than ever to take control of your financial destiny.

Alpaca.Markets provides an easy to use api that allows you to create and test algorithmic trading systems with real time stock market data and their innovative paper trading api.

Technology Used

Operating System: Ubuntu >= 18.0

Programing Language: Python 3.10

Backend: Postgresql 14.7, Celery 5.2.7,
Redis 6.0.16, Channels 4.0,
Alpaca-py .0.7.1

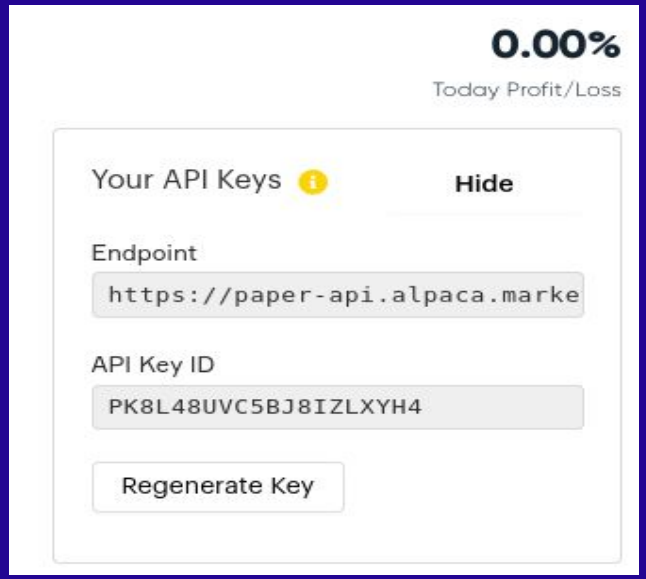
Frontend: Django 4.1.7, Javascript

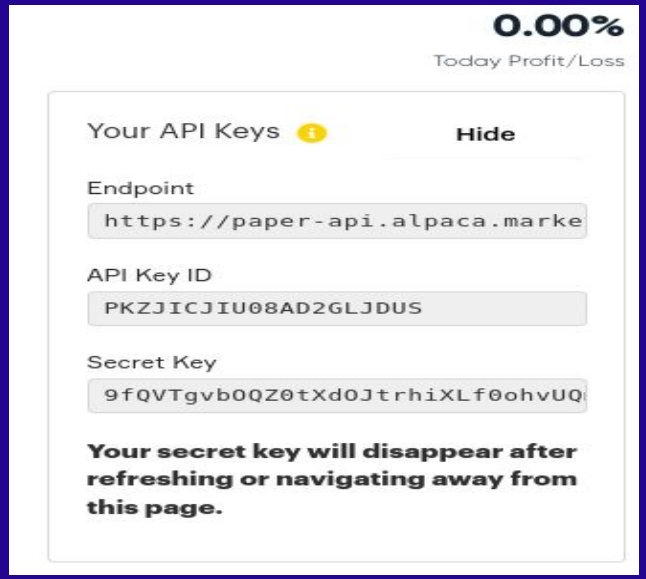
How to Use

First you need to create an alpaca.markets account.
The account is free and only requires a valid email address to set up.

Once email is validated, sign in to the account and generate the api key and secret key. Save them to your systems environmental variables.

https://app.alpaca.markets/paper/dashboard/overview





Set Environmental Variables

AL_AK=Alpaca api key
AL_SK=Alpaca secret key
RT_DATABASE_NAME=postgres
RT_DATABASE_USER=postgres
RT_DATABASE_HOST=localhost
RT_DATABASE_PASS=postgres User’s Password
RT_DATABASE_PORT=5432

Database Set Up

Database Population

After the database has been created and the system environmental variables have been set, there is a command that will need to be ran.

The command will create database tables and populate those tables with historical minute data from January 2022 until the most current trading day.

Navigate to the project's alpaca folder and type in the command window: ./database_population.sh

```
/final_608/alpaca$ ./database_population.sh
```

Django Database Set Up

Django is both a frontend and backend framework. That does a lot of the MVC type of work for you.

In order to integrate Django with the database system a couple of Django commands will have to be ran that will create django controlled tables and the custom tables created for the project.

First Command: makemigrations.py

First navigate to the Django project folder called fp. Then type in: python3 manage.py makemigrations

```
/final_608/fp$ python3 manage.py makemigrations
```

Second Command: migrate.py

Next type in: python3 manage.py migrate

```
/final_608/fp$ python3 manage.py migrate
```

These two commands will build the django models(tables) in the postgres database.

Getting It All Running

The final steps to getting the application running will require a command from the alpaca file, that will start the market data and trade data streams. It will also require a couple commands to get the django celery tasks and beats running.

Starting the Data Streams From Alpaca

Navigate to the alpaca folder and type in:
./alpaca.sh

```
/final_608/alpaca$ ./alpaca.sh
```

You will see this once the streams are started

```
'event'
's'
's'
's'
```

Starting the Django Celery beats and workers

Starting Celery Beats

Open a new terminal and navigate to the fp directory: /final_608/fp

Type in: celery -A fp beat -l INFO

```
/final_608/fp$ celery -A fp beat -l INFO
```

The -l INFO will print the logs of the celery beat to the terminal

```
Configuration ->
    . broker -> redis://localhost:6379//
    . loader -> celery.loaders.app.AppLoader
    . scheduler -> celery.beat.PersistentScheduler
    . db -> celerybeat-schedule
    . logfile -> [stderr]@%INFO
    . maxinterval -> 5.00 minutes (300s)
```

Starting Celery Workers

Open a new terminal again and navigate to the fp directory: /final_608/fp
Type in: celery -A fp worker -l INFO

```
/final_608/fp$ celery -A fp worker -l INFO
```

You should see this on the terminal after starting.

```
-------------- celery@ray-desktop v5.2.7 (dawn-chorus)
--- ***** -----
-- ******* ---- Linux-5.19.0-41-generic-x86_64-with-glibc2.35 2023-05-14 16:48:3
2
- *** --- * ---
- ** ---------- [config]
- ** ---------- .> app:         fp:0x7f5c7b3e28f0
- ** ---------- .> transport:   redis://localhost:6379//
- ** ---------- .> results:     disabled://
- *** --- * --- .> concurrency: 12 (prefork)
-- ******* ---- .> task events: OFF (enable -E to monitor tasks in this worker)
--- ***** -----
-------------- [queues]
               .> celery           exchange=celery(direct) key=celery
```

The Django Master Command

While still in the fp directory type in:

python3 manage.py master_command

```
/final_608/fp$ python3 manage.py master_command
```

If all runs correctly you will see a screen that looks similar to:

```
| Initializing redis listener...[subscribing channel: "intraday_data"]
2023-05-14 21:33:42,171|INFO|alpaca_data| Connected to redis.
2023-05-14 21:33:42,174|INFO|positions_data| Initializing redis listener...[subs
cribing channel: "positions_data"]
2023-05-14 21:33:42,176|INFO|positions_data| Connected to redis.
2023-05-14 21:33:42,180|INFO|trade_data| Initializing redis listener...[subscrib
ing channel: "transactions_data"]
2023-05-14 21:33:42,182|INFO|trade_data| Connected to redis.
Watching for file changes with StatReloader
Performing system checks...
```

Last Step!

Open a browser and type in localhost:8000

The homepage consists of the writing aspects of the project.
There are two links that will take you to the real time application. One link is in the upper left hand corner. While the other link is at the the end of the writing.
The link in the upper right hand corner, my name, is linked back to the homepage.

Disclaimer

This trading strategy is not meant to be used with real money. It’s not a winning strategy!
It is for demonstration purposes ONLY using Alpaca.Markets ***PAPER TRADING API ONLY***.