On-Chain Analysis for blockchain data using the Whale Alert API framework in python.

A simple tutorial of how to implement On-Chain Analysis for one specific metric called Inflow and Outflow transactions of BTC to exchanges using the Whale Alert API. In the market is a consensus that this metric shows a correlation with the price of BTC. I am not an expert in On-Chain Analysis, this is one way to work around with topic and to learn and to study more about it.

I strong recomend NOT to use this tutorial to make an investiment decision, and the proposal of this example is to study the topic only.

In this article, I will explain and show how to use the Whale Alert API to track transaction history data and to make your own analysis from differents blockchains, like Bitcoin, Ethereum, Ripple and others, for more information read the documentation Whale Alert API.

In the implementation I will focus **On Chain Analysis** of the BTC blockchain transaction data to look deep dive into **the metric Inflow and Outflow of BTCs**:

- `from exchange to unknown(wallet)`
- `from exchange to exchange`
- `from unknown to exchange`
- `from unknown(wallet) to unknown(wallet)`

This metric shows a correlation with the price of BTC, because more BTCs flowing to the exchanges, there are a high probability that these BTCs can be sold in the market, i.e. a high strong BTC sales pressure making the price go down, and by the contrary, we expect the price go up, because a high flow of BTCs are going out from exchange to unknown(wallet):

In Summary for this article, I will describe how to build the framework(Using one python package avaliable) to get the data and to make the analysis:

NOTE: For this tutorial I will use the Linux Ubuntu 20 to do everything. If you are using Windows the commands will be similars and check on the internet to see how to make tutorial on Windows to install the framework to develop this API.

1 - Setting the environment to build the framework API

First, you need to have the python and pip installed:

```
# if you do not have pip installed.
# you can install it using the CLI - Install python 3
$ sudo apt install -y python3-pip
```

```
# packages and development tools to install
sudo apt install -y build-essential libssl-dev libffi-dev python3-dev

# Ubuntu 20:
# # https://www.digitalocean.com/community/tutorials/how-to-install-python-3-and-set-up-a-programming-environment-on-an-ubuntu-20-04-server

# python version
$ python --version
Python 3.9.7

# pip version
$ pip -V
pip 21.2.4 from ~/anaconda3/lib/python3.9/site-packages/pip (python 3.9)
```

I strong suggest to create python virtual environment to be a self-contained directory tree that includes a Python installation and number of additional packages(This is good practice when you are developing a project, to know more about see the article A Guide to Python Good Practices). Below I will the main steps to make this:

let's begin by installing the `python3-venv` package that provides the `venv` module

```
$ sudo apt install python3-venv
```

If the installation was sucessfully, we can now to create your new virtual environment:

```
# Here you can change the name of the project as you like :)
# I am called the project of 'api-blockchain-data'

$ python3 -m venv api-blockchain-data
```

To start using this virtual environment, you need to activate it by running the activate script:

```
$ source api-blockchain-data/bin/activate
```

After to set the virtual environment, you can install the python packages that will used in this project:

```
## requirements.txt
colored==1.4.3
dash==1.17.0
dash_core_components==1.13.0
dash_html_components==1.1.1
dash_table==4.11.0
icecream==2.1.1
```

```
ipython==8.2.0
matplotlib==3.5.1
notify2==0.3.1
numpy==1.22.2
pandas==1.4.1
requests==2.27.1
seaborn==0.11.2
snoop==0.4.0
whale_alert==0.0.4
```

Now you can to install all the packages with only one command, you need to save the package names above in one file called: `requirements.txt` on the folder of the project

by typing the following command:

```
$ pip install -r requirements.txt
```

Quick note: This is a great feature of the python, you can list all the packages used on the project using the following command:

```
# requirements for this project.
# requirements.txt
$ pip install freeze

# Install pre requirements, make the command below:
$ pip install -r requirements.txt


# Or if you do not know the package names used on the project,
# you can use the library 'pipreq' that is a pip installable
# and automatically generates the file requirements.txt
$ pip install pipreqs
$ pipreqs '/home/project/location/api-blockchain-data'

## It will print:
Successfully saved requirements file in /home/project/location/api-
blockchain-data/requirements.txt

# and typing...
$ pip install -r requirements.txt
```

2 - Build the API using the Whale Alert framework

To use the Whale Alert API you need to get the API key to do the Authentication to have access to the API service. To create an API key please sign up here.

According with the API documentation:

```
API rate limiting is dependent on your plan.
For the free plan the number of requests is limited to 10 per minute.
The personal plan has a rate limit of 60 per minute. If you need more
requests,
please contact us for an Enterprise account.
```

For this tutorial you can use the `free plan`. After you have created the API key is good practice to keep the API key save, DO NOT SHARE WITH ANYONE OR COMMIT TO PUBLIC REPOSITORY.

How can I keep the API save? you can set as environment variable in bash shell. To do this open `~/.bashrc` and then in the final of the file add the following line:

```
export api_key_whale='here go you api kei'
```

Save and exit the file. After that you need to source the file `~/.bashrc` like this:

```
$ source ~/.bashrc
```

To check if everything is right, just type on the terminal:

```
$ echo $api_key_whale
```

This will return the api key that was stored on the bash shell.

Now to get the api key in python, you only need:

```
# python file.
import os

api_key = os.environ['api_key_whale']
```

Another way is to store your 'api key' in one file, for example, api_key_wl.txt in the folder of the project in computer and commit to repository. To avoid commit to repository you can add the file name in to the gitignore file:

```
## gitignore file has the file names that are not going to the repository
# folder the project and create the file gitignore
$ touch .gitignore

# create the file to store the api key.
$ touch api_key_wl.txt
```

```
# add the api key to the file
$ echo "api key" > api_key_wl.txt

# and add the file name to the gitignore
$ echo "api_key_wl.txt" >> .gitignore

#so, Finally:
$ cat .gitignore
api_key_wl.txt
```

After of All the settings. We are already to start. First, I will show how to use the Whale Alert API library.

2.1 - Whale Alert API How does it works?

To begin this example, first install the python library of the Whale Alert API using pip:

```
# install the python package - whale alert
$ pip install whale-alert
```

Now can run the source code below. The source code here.

```python
# libraries
import os

# whale alert api package
from whalealert.whalealert import WhaleAlert
import time

# set object whale
whale = WhaleAlert()

# set the api key
api_key = 'jlo1G5L7K4CBNcw557uOGOzDrq1i7Oca'

# limit of transactions to make the request is 1 for free plan
transaction_count_limit = 1

# For the free plan the number of requests
# is limited to 10 per minute.
start_time = int(time.time() - 600)

# the request to the whale alert
success, transactions, status = whale.get_transactions(start_time,
                                                       api_key=api_key,
                                                       limit=transaction_count_limit
                                                      )

# result - transaction
print('\n')
print(transactions)
```

NOTE: In this example, I put my API key, for this purpose is not a problem, if was for another thing could be a big problem.

When you run the source code, you will see the following result, one BTC transaction, hash, from, to, amount of coins, amount in USD and timestamp:

```
[{'blockchain': 'bitcoin',
  'symbol': 'BTC',
  'id': '1831370432',
  'transaction_type': 'transfer',
  'hash': 'ae9192fc558d334fbfce24c6b517afd5cc5ce371505973b57f7a45aaa2cca101',
  'from': {'address': '36CqxE4K7Mgc6BGK419Ev8oGJJKc1J4WYG', 'owner_type': 'unknown', 'owner': ''},
  'to': {'address': '3Asr9wQMsK7VcNZy5JpNHkBVm6MocUFG4o', 'owner_type': 'unknown', 'owner': ''},
  'timestamp': 1650499814,
  'amount': 118.860756,
  'amount_usd': 4921723.5,
  'transaction_count': 1
}
```

After this point, I will develop the source code step-by-step of the project to scrape the data from the Whale Alert API and store in a CSV file.

3 - Analysis the data by making the plots of the Inflow and Outflow of BTC

References

- Bitcoin: A Peer-to-Peer Electronic Cash System
- The Bitcoin Standard: The Decentralized Alternative to Central Banking
- The Scalability Trilemma in Blockchain
- On Sound Money
- BITCOIN'S ON-CHAIN MARKET CYCLES
- Carbon -> Create and share beautiful images of your source code