Backend

Reason for Heroku

Tweepy uses two ways of searching for tweets: Searching through the history or listening to real-time tweets. One approach would be to schedule a task, p.e. for every hour, that searches the history of tweets and does the trading afterwards. However, after looking closer into the Documentation, it was discovered that the search service with the Twitter API doesn't show all available tweets. This would lead to incomplete data, which would deflect the strategy and therefore wasn't an option.

When listening to real-time tweets, the python script needs to run the whole time in the background ([FR 80]). It is no option to only run it on the local machine when it is intended to run for days or weeks. When the laptop is closed, the script stops running.

That is where cloud platforms as services come in handy. Today, most of the SaaS-products run in some sort of cloud since it is not very desirable for smaller companies to run and maintain servers for their clients when other bigger corporations such as Google, Amazon or Microsoft offer better, more scalable and cheaper services.

For this project, Heroku was used as the cloud provider, because it works well with Python, has a free tier available and an easy-to-use Postgres Database Add-On ([FR 70]) and GitHub Continuous Integration.

Heroku works with containerization: It automatically detects a change in the GitHub Repository and starts a new Deployment. Because this did not always work, it is still possible to manually deploy in the Heroku Dashboard or with the Heroku CLI.

Heroku will then package the code and dependencies and runs it in lightweight, scalable and isolated containers, which are called *dynos*. In the free tier you have free 550 dynos each month and every time a web service is called, or a script (worker) is run a dyno is consumed.

Using Heroku allows to run the script in the background, store data in a database and schedule trades.

Setting up Heroku

Creating an App in the Heroku Dashboard and initializing it with our codebase in VSCode was the first thing to do. The easiest way to do this is by connecting the Heroku App with the GitHub Repository. But since we were using the Heroku Account from Chainsulting and are not able to access their GitHub account, it was not possible to use GitHub Continuous Integration. Instead, manually deploy every change of code from the terminal with `git push heroku main`.

Secondly, the Heroku Postgres Add-On was added and credentials were stored inside the Config Variables of the Heroku Dashboard.

Our connection with the Database from our python code is established with psycopg2 and SQLAlchemy. Psycopg2 reads the data and SQLAlchemy is responsible for creating a session and then commits the data to the database.

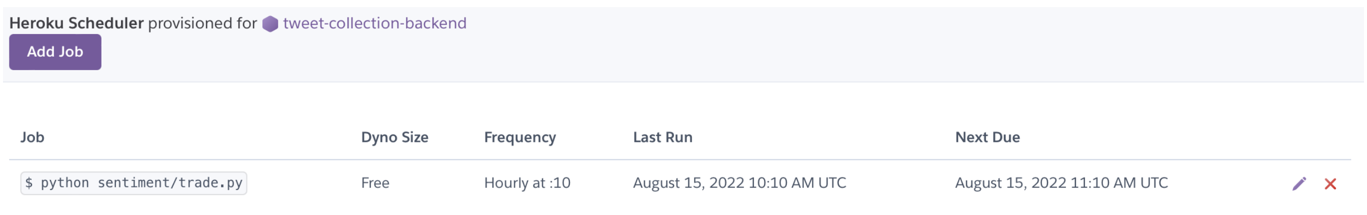
The following example shows how to connect to the database and read the data into a Pandas DataFrame by using a simple SQL Statement.

```
import os
import psycopg2

DATABASE_URL = os.environ['DATABASE_URL']

conn = psycopg2.connect(DATABASE_URL, sslmode='require')

query = f"select * from tweet_data order by id desc limit 10;"

df = pd.read_sql(query, conn)
```

Heroku uses a so called Procfile to start the processes. This meant the worker script for listening to tweets and adding to the database is added to this file:

```
worker: python3 sentiment/runner.py
```

The Heroku Scheduler Add-On is used to execute the trading script every hour. It checks if there are trades for a past timeframe and either buys, sells or does nothing if there is a trade already. More on the trading part here.

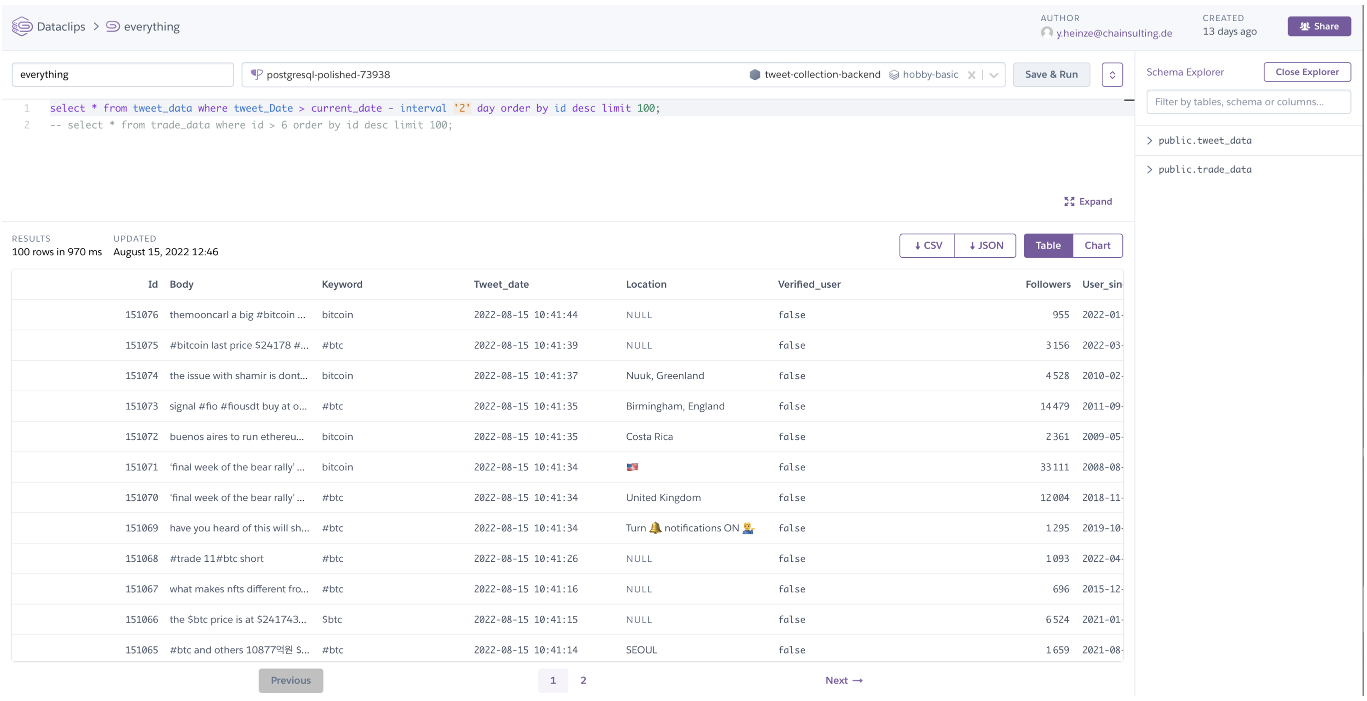


Job	Dyno Size	Frequency	Last Run	Next Due
$ python sentiment/trade.py	Free	Hourly at :10	August 15, 2022 10:10 AM UTC	August 15, 2022 11:10 AM UTC

Figure 11: Scheduler Add-On on Heroku

Checking the Databases

Heroku Dataclips offers to run SQL queries and view the output directly in the Browser. It was a very valuable tool to check quickly if a SQL Statement is executed successfully on the real database and if everything works smoothly in the production phase.



```
select * from tweet_data where tweet_Date > current_date - interval '2' day order by id desc limit 100;
-- select * from trade_data where id > 6 order by id desc limit 100;
```

RESULTS 100 rows in 970 ms UPDATED August 15, 2022 12:46

Id	Body	Keyword	Tweet_date	Location	Verified_user	Followers	User_sin
151076	themooncarl a big #bitcoin ...	bitcoin	2022-08-15 10:41:44	NULL	false	955	2022-01-
151075	#bitcoin last price $24178 #...	#btc	2022-08-15 10:41:39	NULL	false	3 156	2022-03-
151074	the issue with shamir is dont...	bitcoin	2022-08-15 10:41:37	Nuuk, Greenland	false	4 528	2010-02-
151073	signal #fio #fiousdt buy at o...	#btc	2022-08-15 10:41:35	Birmingham, England	false	14 479	2011-09-
151072	buenos aires to run ethereu...	bitcoin	2022-08-15 10:41:35	Costa Rica	false	2 361	2009-05-
151071	'final week of the bear rally' ...	bitcoin	2022-08-15 10:41:34	🇺🇸	false	33 111	2008-08-
151070	'final week of the bear rally' ...	#btc	2022-08-15 10:41:34	United Kingdom	false	12 004	2018-11-
151069	have you heard of this will sh...	#btc	2022-08-15 10:41:34	Turn 🔔 notifications ON 💁‍♂️	false	1 295	2019-10-
151068	#trade 11#btc short	#btc	2022-08-15 10:41:26	NULL	false	1 093	2022-04-
151067	what makes nfts different fro...	#btc	2022-08-15 10:41:16	NULL	false	696	2015-12-
151066	the $btc price is at $241743...	$btc	2022-08-15 10:41:15	NULL	false	6 524	2021-01-
151065	#btc and others 10877억원 $...	#btc	2022-08-15 10:41:14	SEOUL	false	1 659	2021-08-

Figure 12: Dataclip to view the Tweet database with SQL queries

---

Challenges and Solutions

Heroku was an essential tool for this project, but revealed a lot of challenges during configuration. Below are a few challenges and how there were overcome.

No such Procfile

The first problem appeared directly after initializing it with Git and deploying the first time with GitHub Continuous Integration in my own GitHub repository. Although, the Procfile was in the main directory, Heroku could not find it. Weirdly enough, it was fixed after deploying it manually and setting the correct python buildpacks via the Heroku CLI with the following commands:

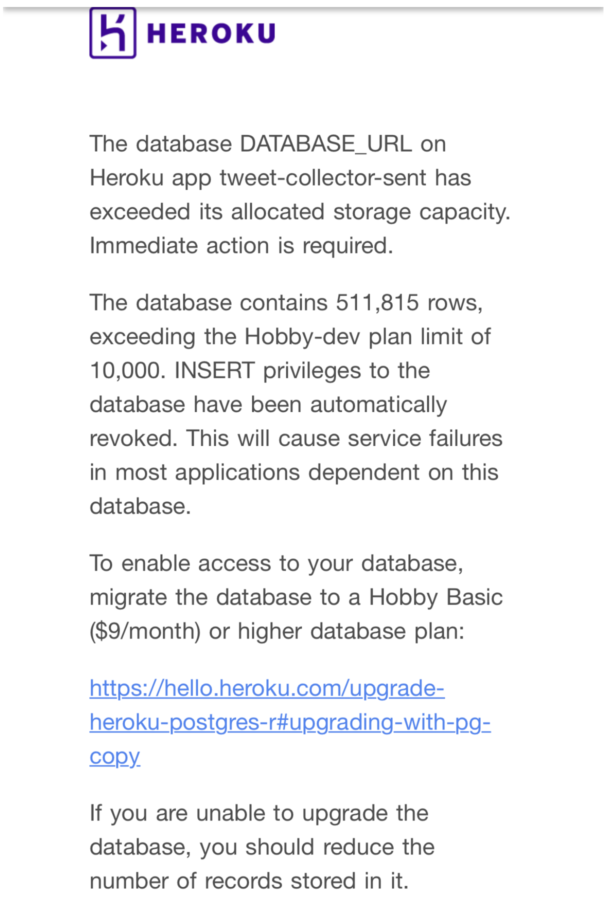
```
heroku buildpacks:set heroku/python`
git push heroku main
```

Connection with Postgres Database

The Postgres Database revealed many problems. You can connect to the Database either via Database URL or via Username, Password and Hostname. Since the latter worked perfectly when connecting to other databases before, this was the first approach, but the password authentication always failed. After a long while of debugging and going through StackOverflow, nothing worked. At random, the connection way was switched to the Database URL and after fixing two further problems the connection was finally established. The URL just needed to be changed from `"postgres://..."`to `"postgresql://.."` and the connection required SSL, which was easily set by `heroku config:set PGSSLMODE=require``

Postgres Row Limit of 10000

The Free Hobby Plan for Heroku Postgres Database has a limit of 10000 rows, and it got filled very quickly. The database was filled in a few hours. But nothing happened. A week passed, and the database was filling up until I got an email from Heroku:



HEROKU

The database DATABASE_URL on Heroku app tweet-collector-sent has exceeded its allocated storage capacity. Immediate action is required.

The database contains 511,815 rows, exceeding the Hobby-dev plan limit of 10,000. INSERT privileges to the database have been automatically revoked. This will cause service failures in most applications dependent on this database.

To enable access to your database, migrate the database to a Hobby Basic ($9/month) or higher database plan:

https://hello.heroku.com/upgrade-heroku-postgres-r#upgrading-with-pg-copy

If you are unable to upgrade the database, you should reduce the number of records stored in it.

Figure 13: Email from Heroku: Database Row Limit reached

The Heroku Database got filled such quickly because the system was collecting Tweets about Bitcoin **and** Cardano. It would be very interesting to compare different coins side by side, but it made sense to only look at Bitcoin after this point to gather more Tweets. In the end this was a waste of time because Chainsulting was so kind and upgraded the Heroku Database Plan to a row limit of 10M, which should last for about a month of Tweets. Of course, this can be improved by better filters and periodically deleting tweets from the database. Unfortunately, when the Database is full, Heroku revokes *Writing-Privileges*, which makes it even impossible to delete single rows via a dataclip. Leaving the only solution to clear the whole database, which has been done a few times before the upgrade.
