Zapper Engineering - Take Home Assessment

Context

The goal of this challenge is to assess your aptitudes at problem solving and see how you would work in a day to day Zapper activity. The challenge consists of multiple items to develop, and you will be asked to explain how and why you chose those solutions in a follow up interview. We're interested in the way you think, so focus on what you feel is the best approach and dive in. The challenge itself is scoped to problems we usually face at Zapper.

The assessment is divided into parts that are put together in a sequence that strives to guide you towards a final state of a codebase that accomplishes a goal.

Tasks

1. The first part consists of writing some code that can extract data from Google's public Big Query dataset for the Ethereum network. Here is an article that can help. The dataset id is: *bigquery-public-data.crypto_ethereum*.

   Once connected, write some code that takes a contract address and returns the top 10 addresses that have done the most interactions with the given contract in the last month.

   You can test your implementation using the USDC smart contract : **0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48**

   Note: You may need to create a Google account if you don't already have one and authenticate with Google Cloud. Google offers a good amount of free usage of Big Query that should suffice for this exercise if you only query data for the last month. It is not expected that you pay anything to complete this exercise.

2. Using the same dataset from Big Query and using the same time range (last month), build a system that keeps track of token holders of the given contract. Store the resulting

table in a local database such as SQLite, MySQL or PostgreSQL. The resulting dataset should minimally allow us to know who currently holds the given token (**USDC 0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48**). Feel free to test with a limited number of token holders being imported into your database such as the top 10 addresses from part 1).

3. Write some code that can fetch the current balance of a wallet address in an ERC20 contract from an RPC node. Feel free to use a Web3 library such as Web3.js or Ethers.js to achieve this task. You can find a list of free RPC nodes to use here: https://chainlist.org/chain/1

4. Using a mix of prior code written in steps 1 through 3 to build a system that can generate a historical hourly view of a wallet balance for an ERC20 contract for the past month. Test using the USDC smart contract and any single user wallet address of your choice.

Notes :

- Use the programming language of your choice.
- Links to libraries are just helping tips, you are not required to use them.
- Do as many of the items as possible, not finishing one or more items does not mean failure of the assessment.
- We encourage you to write down as a comment in the code or a note in the README any future steps that you would have done should you have had more time.
- All code should be in a single directory that can be easily navigated. Separation of concern of the code is encouraged.
- Writing tests is not expected but encouraged if it helps you.
- Creating a private Github repository and adding reviewers as collaborators is preferable. Any other way of submitting the assessment is accepted such as sending a compressed version of the code via email, as long as it's clearly communicated.