- spWe expect, and suggest, that you spend no more than 4-6 hours getting this done;
- We don't want to derail your life, and we know you have other things to do -- feel free to wait for a weekend to get this working;
- We'd like you to let us know by end of tomorrow, $DATEOFNEXTBUSINESSDAY, when we can expect the working product;
- Do, please, let me know if you have any questions about this assignment, or if there's anything unclear.
- In production, we prefer simple, maintainable, code. We'd like to see the same here – we're not looking for cleverness.

Infura Read Cache

Specification

At ConsenSys, we provide services for dApp developers to interact with the blockchain through the infura.io API gateway. Sometimes, the scale of their operations puts an unnecessary burden upon the gateway and in order to alleviate this problem, we implement read caching for the APIs. Caching reads also provides an opportunity to define custom views of authoritative data, which saves some processing time for downstream services.

In your programming language of choice, using whatever libraries or frameworks make sense, develop an Infura API read caching service for Ethereum blockchain data. The following endpoints should be cached periodically. API endpoints outside of this set should be proxied through the service to Infura.

- /v1/jsonrpc/:network/eth_gasPrice
- /v1/jsonrpc/:network/eth_getBlockByHash
- /v1/jsonrpc/:network/eth_getBlockTransactionCountByHash
- /v1/jsonrpc/:network/eth_getTransactionByBlockHashAndIndex

With the cached data, implement the following custom views, with pagination:

- List of all transactions in a block.
- List of transactions in a block to a specific address.
- Average value of gas price over last 24h.
- Average value of gas price over last 168h.

The following implementation guidelines should be observed:

- The service should be delivered as a git repository with a README.md that provides instructions for building, running and testing.
- The service should implement a health check mechanism that will work with a load balancer such as the Amazon ELB.
- The service should accept a port parameter on startup so that we can customize how it runs.
- The service should connect to the Ropsten testnet.

Deployment considerations (interview preparation):

- How would you deploy this service into production and maintain it?
- How would you scale this service to serve a significantly (1000X) larger client load while maintaining the same API call load against the Infura API?
- How would you support multiple different versions of this service in a production deployment?