Blockchain.com SRE Proficiency Test

Times are suggested totals for each stage: they are just indicative of the level of detail we are expecting.

Feel free to use any tools that would be appropriate to delivering a production-ready system.

Software Development

(~90 minutes)

You are tasked with starting a new project at blockchain.com. You have received the following list of requirements from our product team:

We require a non-critical API service (~90% SLO) for some already-existing internal services to use. The service should accept commands on a TCP port composed of ASCII strings. The server outputs an ASCII string as response.

The following commands are accepted:

- `WHO`: outputs the total number of clients connected;
- `WHERE`: outputs the id of the server (a unique identifier);
- `WHY`: output the string `42`;

Please leave enough time to provide a list of items that would need to be added to the project before it would be considered production-ready. (Add this in a `README.md` next to the source code)

Consider

- Maintainability (code hygiene, code robustness, documentation)
- Deployability
- Observability

In this part we are interested in

a) code correctness
b) how you structure a coding project
c) tooling you add to achieve the above goals
d) how straightforward it would be for someone else to continue development where you leave off

e) any decisions you've had to make along the way (please include these in the `README.md`)

In this part, we are not interested in

a) how it fits into the larger ecosystem
b) an exhaustive implementation of all the tooling

System Design

(~30 minutes)

The server you have written in part 1 has now been used in production as part of our non-customer facing micro service architecture for a few months. The engineering team would now like to leverage it as part of our customer-facing infrastructure and as such the target SLO is now 99.95%.

Design a distributed system composed of at least 3 servers. Clients can connect to any server available.

Server response will change to reflect the distributed architecture:

- `WHO`: the outputs should show the total number of clients connected to all available servers
- `WHERE`: the id of the server needs to be a globally unique identifier

Describe the distributed architecture, in particular how you would achieve state replication and fault tolerance

You should justify your ideas in light of the CAP theorem - do you need more information from the engineering / product teams?

Note: code is not required. Design the architecture, describe the architecture, use pseudo code if necessary

Points for discussion afterwards

What considerations would you have if the SLO were 99.9999%?

What changes would you encourage the product team to make in their requirements?

How does this service fit into a larger microservice ecosystem? What tooling will it require to make it useful?