Nexthink Technical Assessment

Role: Software Engineer (Demo)

We recommend that you not spend more than **4 hours** on this assessment. The project should be in modern Rust and created with consideration of **best practices and industry standards**. And if something is not clear, please make reasonable assumptions, document them and carry on.

We are aware that many trade-offs will be necessary in designing and creating a solution with these constraints, and we will keep that in mind. Your submission will be evaluated on the correctness of the solution and the overall quality of the code.

Please submit your code by Email with a zip/tarball attachment of the project files.

In-memory Message Storage Service

Create a service that accepts requests containing a key (string) and a tenant name (string) from **multiple** clients. The service should respond to each request with a timestamp (when the request was processed), a message id (long) and status of the message (boolean saying if it was a new message for that tenant).

Additional details:

- You may use any well-known communication channel/protocol, eg., gRPC, REST, SOAP
    - We recommend using Tokio and\or Tonic to help you get started.
- The message format is for you to define, according to the protocol
- Key strings have a fixed structure, a valid example is: `K-4bbf1-P`

Sample usage:

Client1 sends a message ("K-h53dk-A") with tenant name "3bd1c697", and gets response:

> { "timestamp": 123456, "id": 1, "new": true }

Client2 sends a message ("K-h53dk-A") with tenant name "75682017", and gets response:

> { "timestamp": 123457, "id": 2, "new": true }

Client1 sends a new message ("K-867vc-C") with tenant name "3bd1c697", and gets response:

> { "timestamp": 123458, "id": 3, "new": true }

Client2 sends the same message ("K-h53dk-A") with the tenant name "75682017", and gets a response with the original message id:

> { "timestamp": 123459, "id": 2, "new": false }

Open questions:

1. How would you approach monitoring of this service in production?
2. How would you prevent losing the in-memory cache between restarts?