Exercise 2: Caching in the cloud

In this exercise, you'll need to build a caching system to store users' session data. The session data is addressable by key (the user id) and can range in size between 0.5 KB – 8 KB of data.

The API you'll need to support is composed of the following functions:

- `put (str_key, data, expiration_date)`
- `get(str_key)` → `null or data`

You can expose these functions as rest endpoints, or use any RPC mechanism you desire.

The code behind these endpoints will run on multiple EC2 instances and you need to orchestrate the data between them.

To pass, your solution needs to be able to handle, without data loss, the following scenarios:

- Putting and getting data across multiple keys, that will reside in different nodes.
- Losing a node and not losing any data
- Adding a new node to the system and extending its capacity

The client of the caching service has a single endpoint that it is aware of, you cannot do client-side load balancing. You are free to do load balancing in your code or using the cloud infrastructure to do that.

You do not have to worry about persistence, the data can be kept purely in memory.

You do not have to worry about consistency or concurrency, a read may get an older version of the data, as long as eventually we'll read the latest version.

Deliverables:

Submit your work here: https://forms.gle/osEFP4qN8G3QtSuM8

You need to create:

- The code that would handle the above-mentioned endpoints.
- Include a script that would *deploy* the code to the cloud. Can be bash, cloud formation, custom code, etc.
- Push the results to GitHub or similar service and provide link to the code.
- Inclusion of access keys in the submission will automatically reduce 25% of the grade.