Exercise 2: Dynamic workload

In this exercise, you'll need to build a queue & work management system for parallel processing.

Assume that you need to run a computation over data submitted by users. For the purpose of this exercise, we'll assume that they upload a small binary data (16 – 256 KB) and we need to compute some number of SHA512 iterations on the data.

In other words, assume that the "work" that you do is:

```python
def work(buffer, iterations):
    import hashlib
    output = hashlib.sha512(buffer).digest()
    for i in range(iterations - 1):
        output = hashlib.sha512(output).digest()
    return output
```

Note: you don't have to use Python, this is an example only.

You'll need to create a system, which will offer the following endpoints:

- **PUT** /enqueue?iterations=**num**– with the body containing the actual data.
  The response for this endpoint would be the id of the submitted work (to be used later)
- **POST** /pullCompleted?top=**num** – return the latest completed work items (the final value for the work and the work id).

The system should be able to dynamically able to handle load and deliver the results as soon as possible.

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

To pass, your solution needs to be able to handle, the following scenarios:

- Submitting work to the system and able to get the results.
- Be able to adjust dynamically to the actual workload on the system. Adding more worker nodes or reducing their numbers based on current / projected needs.
- The number of nodes handle the *enqueue* and *pullCompleted* endpoints must be exactly 2, you cannot add additional instance to handle those.

You do not have to worry about persistence, the data can be kept purely in memory. You cannot use any cloud service outside of EC2 instances. The purpose is for you to implement the backend of this directly.

Your code is not required to handle failure modes (a machine failing, network split, etc), however, you do need to create a guide that explain expected failures and how you would handle them if this was a real-world project.

Deliverables:

Submit your work by email to oren.eini@post.idc.ac.il. The email should **not** contain the project itself, but a link to it.

You need to create:

- The code that would handle the above-mentioned endpoints.
- A document detailing failure modes and how to deal with them if this was a real-world project.
- Include a script that would *deploy* the code to the cloud. Can be bash, cloud formation, custom code, etc.
- Upload the results (OneDrive, S3, GitHub, Google Drive, etc) and provide link to the code.
- Inclusion of access keys in the submission will automatically reduce 25% of the grade.