Coding assignment

You are a freelance developer and a small non-tech company hired you for a quick project.

The company has a cloud-hosted WordPress site, without any self-controlled backend. They want to add new functionality to their website which will allow their users to ask to get a conversion of an amount in 1 currency, to another. Their requirements are:

- High availability
- Smooth handling of burstable loads
- Keep the costs to a minimum - you can only use up to 3 cloud instances with 1 CPU and 2 GB of memory each. The machines will run on AWS.
- Advantage if you use message queues (such as AWS SQS)

Your task is to create a backend server(s) that support that functionality.

Since getting the conversion rate is a long and CPU consuming task (for the sake of this assignment of course) and the company only allows you to buy up to 3 small cloud instances, you let the company know that it will be best to defer the user requests and email the results back to them once it is ready (in a reasonable amount of time).

Your task is to decide on architecture and implement the backend server(s)

Please follow the following instructions:

- The code must run on NodeJS LTS
- Use Typescript and keep type safety as much as possible
- You can use any framework and package manager of your choosing
- The API should be REST API
- Maintain a clean, readable, and structured code

Submission:

- Upload the result to Github so we can review it (make it **private** and invite regevbr@gmail.com as a collaborator)
- Upload a docker-compose that will start all the services.
- If you decide to use a message queue, please also start it in the docker-compose and look for the readme in the starter project (listed below) on how to do that.
- **Send an email with your full name and the repo link to jobs@pruvo.com**

Tips:

- You can use https://github.com/PruvoNet/ts-starter as a starter for the task (do not fork it as you can't make the fork private)
- Use the free plan of https://openexchangerates.org/ to retrieve the currency rates
- If you use SQS, please do it using https://github.com/PruvoNet/squiss-ts.
  You can set up a local instance of SQS with
  https://hub.docker.com/r/softwaremill/elasticmq-native/

using the following config:

```
include classpath("application.conf")
node-address {
  protocol = http
  host = localhost # or the hostname provided by docker compose
  port = 9324
  context-path = ""
}
rest-sqs {
  enabled = true
  bind-port = 9324
  bind-hostname = "0.0.0.0"
  sqs-limits = strict
}
rest-stats {
  enabled = true
  bind-port = 9325
  bind-hostname = "0.0.0.0"
}
generate-node-address = false

queues {
  default {
  }
}

aws {
  region = dummy
  accountId = 000000000000
}
```

and passing

```
{
  accessKeyId: `dummy`,
  secretAccessKey: `dummy`,
  region: 'dummy',
  endpoint: 'http://localhost:9324'
}
```

to the AWS config in squiss-ts.

- Pay attention to the client's requirements
- Make sure as much as you can that your servers will not crash during load peaks

Assumptions:

- Create a dummy async function that will send the email, no need to actually send it

- No need to handle authentication and/or authorization
- There is no need to create an AWS account or use AWS directly in any way