Assignment #6: Secure and Scalable

# Goal

In this assignment, you will improve the security and scalability of your user service by moving it to AWS EC2, securing communications with HTTPS, and isolate your configuration and improve reliability with PM2.

# Prerequisites

**This assignment continues from Assignment #4 (*not #5*);** you will start with your submission for that assignment. If there are major functional issues with your service or game project, you should address them before starting this assignment.

You must complete the two AWS labs **– Tour of AWS and EC2, and PM2 and Nginx –** before you start this assignment. Note that to complete the latter lab, you will have a tested and functional domain, and the class certificates will be installed appropriately.

# Tasks

First, you should update your Node project to launch from **assignment6.js** (as you have updated the filename previously, including in package and package-lock.json, etc.).

You must deploy your service to your EC2 instance, from a directory like '~/assignment6` (for example).

Your **service port** should still be **3100**, though it must be **configurable** with PM2 configuration (see below!). This means the user service will not be externally accessible directly -- only through Nginx.

## Install Redis

You will need to install Redis on your EC2 instance to run your user service. You can review the Redis lab to review the steps:

**sudo apt install redis-server**

You can check the status with **systemctl status redis-server**

You can use **redis-cli** in MobaXterm to interact with your Redis instance (debugging and verifying things are working).

## Install MongoDB Community version 6

You will need to install MongoDB v6 on your EC2 instance as well. Return to the Mongo Lab documentation if you need to, or follow this: <https://www.mongodb.com/docs/v6.0/tutorial/install-mongodb-on-ubuntu/>

**sudo apt-get install gnupg curl**

**curl -fsSL https://www.mongodb.org/static/pgp/server-6.0.asc | \  
sudo gpg -o /usr/share/keyrings/mongodb-server-6.0.gpg --dearmor**

(This gets the public key that you'll need to access this repository)

**cat /etc/lsb-release**

(So you can tell what version of Ubuntu you're running)

If you're running Ubuntu 22.04 ("Jammy") then (all one line):

**echo "deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-server-6.0.gpg ] https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/6.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list**

Reload the packages database:

**sudo apt-get update**

**sudo apt-get install -y mongodb-org**

Our EC2 Ubuntu instance doesn't have the same limitations as the WSL Ubuntu instance does. We can tell the system to start mongodb up automatically - which for Ubuntu means configuring system:

**sudo systemctl start mongod**

And then verify it worked:  
**sudo systemctl status mongod**

And finally, if all is good, tell systemd to restart mongod for us:

**sudo systemctl enable mongod**

Also, you should verify that things are working by starting "mongosh", the "mongo-shell" program - you can list out the databases to verify that things are in a good state:

**mongosh**

**show databases**

A screenshot of a computer program

Description automatically generated

If you see something like the above, things seem good. If you see errors, then you have more debugging to do…

## Configure Nginx

Your service must be publicly accessible via **port 443 over HTTPS**, using Nginx.

You must update/add a service configuration for Nginx for your service, using the appropriate port that your service is running on, and listening for ssl on port 443 (similar to the lab, just on a different port). Don’t forget to reload Nginx after you change the configuration, otherwise it won't "see" the new changes.

Your certificate configuration from the Nginx lab will work for our purposes. You won't need to move the certificates themselves, and you should re-use the certificate configuration lines from the server block for the user service.

While you're at it, you should remove the lab service configuration (i.e., port 4000) from Nginx.

## EC2 Security Groups

Your EC2 instance’s security configuration must accept port 443 (HTTPS) traffic from any IP.

Note that there is a “HTTPS” option in the Security Group “inbound rule” drop-down that will set this port.

You **must ensure that your server is not accessible at Any IP via anything other than HTTPS**!

You should remove the TCP-4000 configuration, and your service must not be accessible via HTTP (port 80) or port 3100!

## Ecosystem and PM2

Your service must be always running on your EC2 instance under PM2, using an ecosystem file.

All configuration elements in your service must be isolated to the PM2 ecosystem file, including **at least these nine elements**, with reasonably descriptive names:

* 1. Listen port
  2. Mongo connection string
     1. Or, *Mongo Address and Mongo Port*, depending on how you want to do it
  3. Mongo database name
  4. Mongo collection name
  5. Redis address
  6. Redis port
  7. Redis expiration duration (currently 10 seconds)
  8. Game port (part of the Connect API response)
  9. Shared secret (i.e., *cs261secret*)

## Game Client Changes

The debug command line for the CS 261 Assignment 4 client should be updated to connect to your own domain via HTTPS (no port specified).

You will include your Assignment 4 game solution, but you will not need to change it – aside from the debugging command line modification and testing to see that it still works!

# Testing Your Work

Your client and server from Assignment 4 must be included with this submission, and it will be tested in a similar way (exemplar client/your server, your client/exemplar server). The primary difference is that the client will be tested connecting to *https://somename.cs261.meancat.com*, where the domain is the one specified in your DNS page in the class notebook.

Your service will also be tested at other ports on your domain (80, 3100, etc.), to ensure that it *fails* – *only* port 443 should be accessible!

Your Node project source code will also be tested locally via PM2, and the environment variables will be modified in the configuration to ensure that the changes are applied. There will always be a *reasonable* value provided for all configuration elements.

The Assignment 6 tests will be run vs. your service running at your domain (https://yourname.cs261.meancat.com), as well as when running locally via PM2 (<http://localhost:3100>).

The Assignment 4 tests will also be run against your service when running locally under PM2, to test the changes to the ecosystem.

To test Assignment 6, you'll need to specify that you want HTTPS as well as the address the test should try:

node putter run path/to/CS261\_Assignment6\_Postman.json --https --address YOURNAME.cs261.meancat.com

# Submission Requirements

Your submission should be named **yourAlias-CS261-6.zip**, where yourAlias is the username you use to log into your email and lab PC.

Your submission will be similar to your Assignment 4 submission, but with your updated Node project. The only change to the game project should be the updated debugging configuration for the client project - you don't need to rename the projects for this assignment.

The submission should be exactly like the Initial project, without any build output, binaries, etc. In particular:

* **node** directory, containing:
  + **assignment6** the primary Javascript file
  + other Javascript files
  + ecosystem.config.js
  + package.json
  + package-lock.json
  + *No node\_modules directory in here!*
* **game** directory, including:
  + Assets *(exactly as provided in Initial)*
  + CProcessing *(exactly as provided in Initial)*
  + Clumsy *(optional)*
  + CS261\_Assignment4
    - *No Debug or Release directories in here!*
  + CS261\_Assignment4\_Client
    - *No Debug or Release directories in here!*
  + CS261\_Assignment4\_Server
    - *No Debug or Release directories in here!*
  + CS261\_Assignment4.sln
  + *No Debug or Release directories out here, either!*

Note that you should not include any built binaries (Debug or Release directories, at any level), nor any .git or .vs directories. **The submission file size should be around 1.9 MB.**

# Rubric

If your game code does not build under Debug and/or Release configurations, crashes, etc., you will likely receive a 0 for the assignment. Similarly, if your Node project does not run or is not active in EC2 after submitting to Moodle, then you will likely receive a 0 for the assignment.

## 30 Points: Unit tests *all* pass via HTTPS and your domain (93 total tests)

The Assignment 6 set of unit tests will be run against your domain and HTTPS. If all 93 tests pass, then you will receive 30 points (otherwise 0 points)**.**

## 30 Points: Unit tests *all* pass locally when run under PM2 with HTTP (93 total tests)

The Assignment 4set of unit tests will be run against localhost and the configured listen port (3100 by default). If all 93 tests pass, then you will receive 30 points (otherwise 0 points).

## 30 Points: All required configuration elements can be modified via the PM2 ecosystem

All **nine** of the configuration elements listed under Task #6, above, are in the ecosystem and used appropriately within your Node code. If all configuration elements have been correctly extracted to the PM2 configuration, then you will receive 30 points. You may receive partial credit if there are issues with some configuration changes.

## 10 Points: Your EC2 instance only responds to HTTPS (443)

Your server should only respond to requests on port 443, and only to traffic for your user service. The default HTTP (80) traffic, port 4000 lab testing, etc. should not be available, nor should the user service be available directly via port 3100, etc. If your server is correctly locked down, you will receive 10 points.

## Assignment 4 Client and Server Operational

Your assignment 4 client/server, which must be included in your submission, will also be tested against both your server (HTTPS) and your service running locally under PM2 (HTTP). If it does not function correctly, then penalties will be applied, depending on the severity of the problem.

## Submission Penalties

In addition to the grading rubric above, you can receive these penalties if your submission is not correct:

* -5 if your submission files are not named as above.
* Up to -10 if your submission includes additional files, depending on the severity.

Other penalties may apply if needed.