1. What is the "key" in the REDIS database where the web server stores the user's image. What type is the value? Describe the structure of it.
    - The key and value are stored in a json object
    - The key is "id": a randomized uuid
    - The value is "image": the image (base64 encoding)
2. How does the web server communicate with the model (worker) server to hand off work and receive back results? Describe how it works for the web server to respond to web requests.
    - The web server pushes up the data sent by the user to the queue in redis
    - The model server polls for data to come into the redis queue, and when it does, grabs the image and processes the data, putting labels and probability into the database
    - The web server waits for the results to come in to the redis queue and when they do, output the results to the user
3. What is the result of sending "castle_image.jpg" through simple request? What objects with which scores does it identify this as?

    1. church: 0.4136
    2. castle: 0.3930
    3. monastery: 0.1733
    4. palace: 0.0041
    5. vault: 0.0034
4. Does the current system have any issues if you were to have multiple front end servers (run_web_server.py) and multiple workers (run_model_server.py)? What issues are there? Update the code to fix one of these issues, test that it works, and explain with words and diagrams what you did to solve it.

    - The web servers are fine, but the model servers process the request at the same time. This isn't really an issue because Redis uses a "last write wins" approach when dealing with concurrent writes, but it uses twice as many resources for the same output.

5. Polling is the most expensive way to interact with a system. Polling can be reliable, but costly. Rather than having the web server poll REDIS waiting for a key to show up, could you research, think, and implement a way that instead uses notifications and only uses polling as a backup. Test your implementation and describe how you fixed it with words, code, and diagrams.

    - I was thinking through this problem and was brainstorming a way to solve this issue with some sort of notification system. I did some reading on the redis pub/sub method and couldn't exactly figure out how to make it work. I'm not sure if I'll have time to get this done

6. The stress_test.py code doesn't work. Could you fix it?

    - The web server - max number of clients reached

7. Tracing what happens in a multi-agent system can be challenging. Can you write a simple logging function that can be called from any of the files that logs in a consistent format (server name, main running python script, timestamp, action)