2.1.3 Sprint 2: Flask Application

Now that you have written code that deploys properly we are going to finish off the project by building a web interface that will receive and process JSON objects. At a high-level this application should accept connections, log all requests and then properly process well-formed JSON strings. This document contains details around the specifics.

Requirements

- Your team needs to create a repository (you can use the same one you used before) and write a deploy script which completes the assignment. As before, the deploy script should log into the box via SSH and pulls your code from your repository.
- Your script needs to start a Flask application which accepts HTTP post requests on port 8080.
- To be clear about the type of request that it should accept raw data, within the body, formatted as JSON (application/JSON).
- Each request should be logged as a string, with any hard returns removed, into a log file within `/srv/runme/prefix/Raw.txt` (similar to the raw data in the previous assignment, so that each row represents a new possible data point). You should use a log rotate (either via the operating system or via logging package and TimedRotatingFileHandler) to rotate the file every 2 minutes into another file within that same directory.
- If a request is "valid", where valid is the same definition as in Sprint #1, then the processed output should be put into `/srv/runme/prefix/proc.txt`. The JSON object should be processed in the same way as in the Sprint #1 and the file should be rotated every two minutes.
- As in the previous assignment, a `prefix` will be passed to deploy function which will define the name of the output file.
- In order to test your code, the following line will be appended to the end of the deploy.py code provided.

  ```
  deploy( 'path_to_ssh_key_private_key', 'server-address', 'prefix')
  ```

Hints

- While I recommend using Flask to complete the assignment, other web frameworks may also be used, as long as they can be installed via Pip and are in Python 2.
- *Postman*, a chrome extension and app for sending HTTP requests may be useful.
- To build a testing environment two boxes will be created – one to send and one to receive requests. The two boxes will communicate via HTTP requests, which will require dealing with Amazon's security. This may take a while, so plan for it.

- The code is undertaking multiple operations, feel free to break up your code into multiple pieces and implement them separately.
- You are free to use screen and cron as well as Python libraries available via Pip. If your code requires an uncommon library, please send me an email ahead of time so that I can make sure that it is on the Server.
- The script will run as the "testtest" user and superuser privileges will not be required. All required directories and permissions will be given.