Unit Testing

The first step of testing will be done iteratively through manually running the programs with pre-determined test cases. No automation will be used here. The units can be broken down into the following two sections:

Client

1. Create multiple instances of client image
2. Concurrently run client instances
3. Properly record status, speed, etc. data in log files
4. Ensure closure of communication for all clients
5. Clear log files for all clients

Server / Load Balancer

1. Set up NGINX servers
2. Spin up multiple instances in docker swarm
3. Load balance incoming client traffic
4. Serve up correct, consistent content in response to clients
5. Ensure accurate server response through load balancer
6. Properly record status, speed, etc. data in log files

Integration/System Testing

Integration testing is intuitive and essential here since it is a must for any success in our project. We must verify proper communication between the clients, the load balancer, and the servers. Since the programs will always run in docker containers, system testing is performed alongside integration testing here.

1. Startup clients and servers on different systems
2. Clients begins sending requests to load balancer
3. Load balancer directs traffic per the given algorithm
4. Servers receive requests and send back response with correct content
5. Load balancer forwards response to the correct client
6. Requests continue from clients for given time frame
7. Servers continue to serve up content correctly
8. Clients cease requests
9. Servers are shut down manually

Customer Testing

We will be meeting with our customer to discuss results and perform some quality assurance. Our customer has open access to our testing process and code. Most of the customer verification will be in analysis of the data we collect.