**Docker Autoscaling Demo Script**

1. If the demo as been run previously, cleanup all the containers:

**./removecontainers.sh**

1. Run: **docker ps**. Show that there are no containers running.
2. Show **http://[docker host]:8080/status.html**. There will be an error in the browser because NGINX PLUS is not yet running. Continue to show status.html after each action that effects the NGINX Plus configuration.
3. Run: **./createenv.sh**. This will setup an environment with one NGINX PLUS load balancer container, one NGINX PLUS web server container and upstream server and one Elasticsearch container and upstream server.
4. Run: **docker ps**.
5. Create some more Elasticsearch containers:

**./addes [number of containers]**

1. Remove some Elasticsearch containers:

**./removes [number of containers]**

1. Create some more NGINX web server containers:

**./addnginxws [number of containers]**

1. Remove some NGINX web server containers:

**./removnginxws [number of containers]**

1. Show script createenv.sh. This script calls the following 3 scripts.
2. Show script addnginxlb.sh. This script creates a container with NGINX PLUS configured as a load balancer with an upstream for NGINX PLUS as a web server and one for Elasticsearch.
3. Show **/var/nginx/config/docker.conf**
4. Show script addnginxws.sh. This script adds a container with NGINX Plus, using the default configuration and adds it as an upstream server. If copies the html content from the Docker host.
5. Show script addes.sh. This script adds a container with Elasticsearch and adds it as an upstream server.
6. Show script addnode.sh. This script is called by the previous two scripts and does the actual container creation and upstream server additions.
7. Show script removenginxws.sh. This script removes one or more NGINX Plus nodes and their containers.
8. Show **[path]/docker\_base/Dockerfile**.
9. Show **[path]/docker\_lb/Dockerfile**.
10. Show **[path]/docker\_ws/Dockerfile**.
11. Generate load: **./runsiege.sh**
12. Run: **./autoscale.py**. Explain that the program will try and keep the requests per second per active node to between 10 and 12. It will add at most 4 nodes on scale up and remove at most 2 nodes on scale down. It is checking every 2 seconds. There can be a maximum of 10 total nodes and a minimum of 2 active nodes.
13. Run: **./seterror.sh [port]** for two NGINX Plus web server instances to cause the health checks to fail. Explain that the autoscaling algorithm will take into account the servers that are actually up.
14. Show autoscale.py
15. Stop the load generator when there are more then 2 active nodes. Show that the program will scale down to two active nodes.
16. Run: **./fixerror [port]** for the two nodes that are failing the health checks and show that two nodes are removed to the minimum of 2 active nodes.