Exercise #1: Demonstrate your linux skills and ability to work with new open source technologies

- Please go to the DataStax website, download and install DataStax Enterprise and Opscenter.
    - Execute our Portfolio Demo Hive Example
    - We recommend you do this on Linux and if you don't have access to a server to use EC2 , VMWare or virtual box.
- Assignment: Provide screen shots and also any feedback or gotchas.

Exercise #2: Show us how you think about testing

- Background: Cassandra is architected for failure so when nodes go down, if the cluster is configured correctly, you can continue to serve reads or writes, and then incrementally repair the node once it comes back online.
    - When you configure a cluster, you startup with a set of nodes, and then you define a keyspace which is a collection of column families / tables. When you define a keyspace, you specify the number of replicas to store, and then all column families in that keyspace will inherit that value.
    - When you read or write, you specify a consistency level, which is the number of replicas which must acknowledge the read/write action.
    - If a node goes down, and the consistency level is less than the number of total nodes, then reads and writes can complete successfully as long as the requested number of replicas is still online.
    - After the node is brought up, you can run the repair command to repair any new writes to that node.
- Assignment: For a 3-node cluster, can you think of some interesting test scenarios to validate that you were able to correctly repair data that was inserted while the one node was down?
- Assignment: What strategy would you use for automating this suite of scenarios?

Exercise #3: Demonstrate coding skills for a distributed system (Python / Java / Perl acceptable)

- Goal: Implement a ***mock*** cluster and expose operations on that cluster.
- Example architecture:
    a. A class to represent a node: should have an ip address, name, and load (doesn't matter whether it's disk or cpu, etc).
    b. A class to represent a cluster should have a name and a collection of nodes.
    c. A class to represent a thread for the monitoring in step 2 below implement and expose whatever operations on each class above as needed to accomplish the following:
- Example Scenario:

    a. Initialize a cluster of 5 nodes
    b. Start a new thread that checks the load of each node in the cluster every second and stores this information for later use
    c. Load the nodes of the cluster so that the load is distributed unevenly. pause for 5 seconds after loading each node
    d. Print the status of the cluster providing each nodes' name, ip address, and current load
    e. Cause the load to become evenly distributed
    f. Repeat step c.
    g. Print out the information obtained from the thread in step b..

- Guidelines:
    - The above references to "load" are to be taken as a simple number and doesn't need to be real data. If it is though that's ok too.
    - Use whichever language you feel most comfortable with.
    - Use whatever architecture you want to accomplish the task, but the final product should be runnable from a standard linux command line