To continue from where we left the experiments, follow all of the instructions from the guide and do the following as well:

On the machine that spark is set up on:

- docker stack deploy -c <docker-compose-file> <stack-name>
- will deploy all of the services listed in the docker compose file

In our experiments we were looking to find a happy medium for performance and security, though it seems that the current architecture contains a bottleneck in pulling from Kafka. It also appears to not be parallelising the data correctly. Try to find a way to get each executor to pull a set of records from Kafka, rather than having the driver do this and then distribute the data.

Start up the Zookeeper, Kafka, and Cassandra servers.
Give everything some time to start up, if the sensors start too early there will be a massive message queue before the spark job is ready to receive data.
Start up the sensors and begin monitoring.

Things to consider adding:

- Dockercloud seems promising in terms of its ability to scale applications. It appears that you can easily interact with the api in python to make changes based on cpu usage, when used together with the virtual lab environment.
- While I believe detecting the injection threats to be relatively trivial, detecting the DOS threats is more complicated. In a distributed system, I am not 100% certain that my implementation will work as expected and you should not take for granted that the broadcast variables will be updated properly within each cycle of the spark job.
    - Also considering adapting to detect for DDOS threats.
- Try to change the spark job to accept JSON data rather than a string.
- Look into potential network issues that cause delays in the spark streaming. Most of executor loss was assumed to have come from the limitations on resources, but it is worth considering network errors.
- Experiment with the number of cores given as well as other resource limitations
- Look into using docker-compose up vs docker stack deploy
    - you will need to make changes to the docker-compose.yml if you do so.
- Use variables in docker compose. https://stackoverflow.com/questions/43544328/pass-argument-to-docker-compose
    - this way you can scale upwards or outwards with variable inputs

Other tips I have been given:

- use dataframes instead of RDDs
- use pyspark instead of scala spark