**Spark Submit**

Being able to interact with data using the Spark REPL (or, in our case, Zeppelin notebook) is extremely valuable. But there are times when you want to just run a job as a batch process.

The goal of this lab is to run a batch process that will query our raw Sparkify log data and save the top 20 songs to S3.

In order to do this, we will use the spark-submit utility. This should be run on the master node of your Spark cluster. Take a look at [How To Write Spark Applications in Python](http://blog.appliedinformaticsinc.com/how-to-write-spark-applications-in-python/). Note that the documentation for spark-submit allows for a lot of options you don't need, so try not to get distracted by that.

Note: You may need to unset PYSPARK\_DRIVER\_PYTHON in order to prevent spark-submit from trying to launch Jupyter. You may also need to specify sc = SparkContext() in this case.

Once you've got it working, you may wish to look at the Deploying guide to [Submitting Applications](http://spark.apache.org/docs/latest/submitting-applications.html) and and [Submitting Spark Jobs to an Amazon EMR Cluster](https://aws.amazon.com/premiumsupport/knowledge-center/emr-submit-spark-job-remote-cluster/) but probably not before.

Don't forget to test your code on a subset of your data first! You don't want to have to wait for it to chunk through over a 100 GiB of data only to find it didn't work.

Save your results using one of the saveAs... actions. (Warning: Do **not** try to collect all of your results without limiting it first.)

Your deliverable for this lab is the python file you submit along with instructions on how to run it. Those instructions can either be in a comment block in your python file or in a separate README.

Bonus:

* Save your results as HTML in your static website. To do this, you will probably have to collect the data you want to publish and then render that as a web page and put that in S3 using boto the old fashioned way. Better yet, separate this into two jobs (two scripts), a pyspark job and a traditional python job to be run in series.
* Install spark-submit locally and use it to submit your job to a remote cluster.