# [Proctor Guide]

# Challenge 02 – Ingest and Prepare Data

## Summary

A complete set of Databricks notebooks that covers Challenge02 is available with this documentation in the Solutions folder.

Refer to these as needed to help teams accomplish the tasks in this challenge. The notebooks are broken out per task.

## Task 1 – Copy source data files to your Azure storage location

Tools they could use include Azure Storage Explorer, azcopy, az cli, etc.

## Task 2 – Connect cluster to your Azure storage location

Teams should add the storage account key, in (very specific!) format, to the cluster configuration. See reference below. At the least, teams should add this key-value pair (key and value separated by a space, and with the storage account name and key tokens replaced with the team’s info):

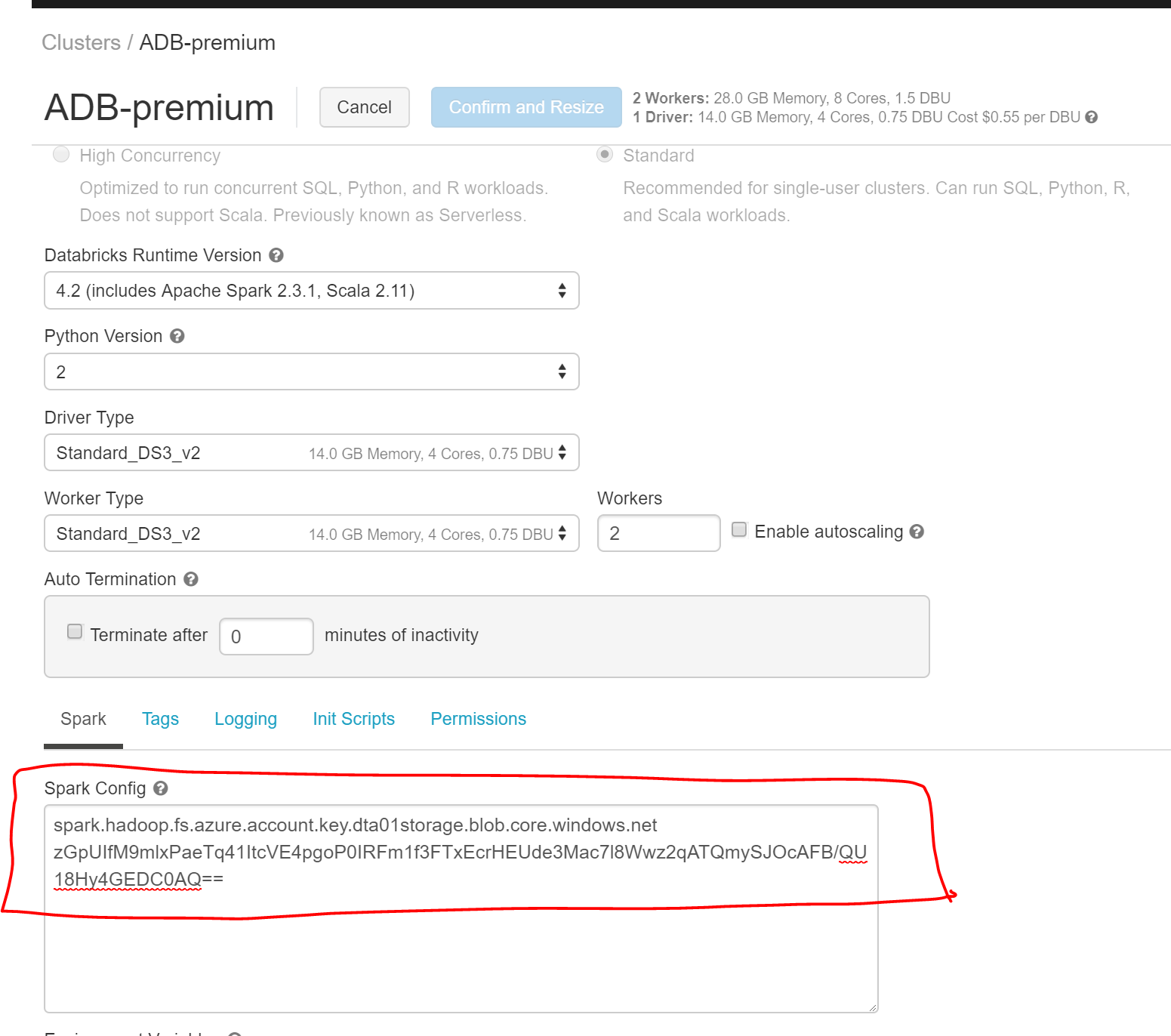
fs.azure.account.key.{storage account name}.blob.core.windows.net {storage account key}

Specifically, the key is the dot-delimited storage account reference, and the value is the storage account key. This will permit Azure storage access from within Databricks notebooks, including creating mount points.

Reference:

<https://docs.azuredatabricks.net/spark/latest/data-sources/azure/azure-storage.html#mount-an-azure-blob-storage-container>

IMPORTANT: as attendees connect their Databricks cluster to Azure storage and create mount point(s), the storage container(s) to which attendees will attach mount point(s) must exist already. If attendees get error messages when trying to create mount points, have them double-check their cluster configuration (storage account name and key) and that the container(s) they are connecting to exist.



## Task 3 – Create a Hive Database

Please refer 03-Prep-Database.scala

## Task 4 – Create Hive Tables for Reference Data

Teams will need to take the following steps:

1. Understand the source file schemas
2. Read each source file into a dataframe
3. Write each dataframe out to Parquet format
   1. This is needed to remove header rows from CSV files, and because Parquet files are faster and more efficient than CSV for Hive operations
4. Create an external Hive table on top of each Parquet output location

Note! Parquet reads and writes can be parallelized and partitioned. This means that a single dataframe can result in multiple Parquet files. Teams should be very careful to write each dataframe out to its own dedicated destination folder!

## Task 5 – Create Hive Tables for Transaction Data

Following a similar process as in task 4, teams should wind up with one Hive table for Yellow, and one for Green.

To succeed, they will need to do the following for each of Yellow and Green:

1. Understand how the source data file schemas change over the years
2. Create an external Hive table with a single schema that covers all the source year schemas
   1. They will need to specify a file system location for this table. The folder they specify should be the root folder to which they will write Parquet outputs. Each of the two Hive tables (yellow, green) should have its own root folder.  
      Examples: /mnt/data/dtafy19/processed/yellow-taxi/ and /mnt/data/dtafy19/processed/green-taxi/
3. Iterate through all the source data folders (for each year, for each month) and:
   1. Using the correct source schema for that year/month, read the source data into a dataframe
   2. Adjust the dataframe to confirm to the single schema of the target Hive table created above
   3. Write output as Parquet, into a new folder underneath the Hive table’s root folder. Example:  
      /mnt/data/dtafy19/processed/yellow-taxi/trip\_year=2016/trip\_month=10/
   4. Consider encouraging teams to partition the data for each year/month to improve query performance.

## Task 6 – Conform and Merge Data Sets

In this task, teams will create one merged master from the yellow and green merged data sets created in task 5.

To do this, they will need to ensure both data sets have the same schema, then union them into one data set and write that out to Parquet, then create a Hive external table on that final Parquet data.

Teams have successfully completed challenge 2 when they can query one Hive table that contains ALL yellow and green data.

## Bonus Task – Connect Databricks Notebook(s) to github

The Databricks documentation for how to do this is here: <https://docs.azuredatabricks.net/user-guide/notebooks/github-version-control.html>

Note that the github UI has changed since the Databricks doc was written. Attendees should now go to github Settings 🡪 Developer Settings 🡪 Personal access tokens and proceed per the Databricks documentation.

If attendees create a new github repo, they should first clone it to their machine and commit a file (e.g. a readme or a .gitignore) and commit/push that. Without an initial commit, git sync in Databricks may fail, but succeed after new github repo is initialized.