**S3 Buckets**

With the convenient AWS UI, we can easily mistake AWS S3 (Simple Storage Service) equivalent as Dropbox or even Google Drive. This is not the case for S3. S3 stores an object, and when you identify an object, you need to specify a bucket, and key to identify the object. For example,

df = spark.read.load(“s3://my\_bucket/path/to/file/file.csv”)

From this code, s3://my\_bucketis the bucket, and path/to/file/file.csv is the key for the object. Thankfully, if we’re using spark, and all the objects underneath the bucket have the same schema, you can do something like below.

df = spark.**read**.load(“s3:*//my\_bucket/”)*

This will generate a dataframe of all the objects underneath the my\_bucket with the same schema. Pretend some structure in s3 like below:

my\_bucket

|*---test.csv*

path/**to**/

|*--test2.csv*

**file**/

|*--test3.csv*

|*--file.csv*

If all the csv files underneath my\_bucket, which are test.csv, test2.csv, test3.csv, and file.csv have the same schema, the dataframe will be generated without error, but if there are conflicts in schema between files, then the dataframe will not be generated. As an engineer, you need to be careful on how you organize your data lake.

### Differences between HDFS and AWS S3

Since Spark does not have its own distributed storage system, it leverages using HDFS or AWS S3, or any other distributed storage. Primarily in this course, we will be using AWS S3, but let’s review the advantages of using HDFS over AWS S3.

Although it would make the most sense to use AWS S3 while using other AWS services, it’s important to note the differences between AWS S3 and HDFS.

* **AWS S3** is an **object storage system** that stores the data using key value pairs, namely bucket and key, and **HDFS** is an **actual distributed file system** which guarantees fault tolerance. HDFS achieves fault tolerance by having duplicate factors, which means it will duplicate the same files at 3 different nodes across the cluster by default (it can be configured to different numbers of duplication).
* HDFS has usually been **installed in on-premise systems**, and traditionally have had engineers on-site to maintain and troubleshoot Hadoop Ecosystem, which **cost more than having data on cloud**. Due to the **flexibility of location** and **reduced cost of maintenance**, cloud solutions have been more popular. With extensive services you can use within AWS, S3 has been a more popular choice than HDFS.
* Since **AWS S3 is a binary object store**, it can **store all kinds of format**, even images and videos. HDFS will strictly require a certain file format - the popular choices are **avro** and **parquet**, which have relatively high compression rate and which makes it useful to store large dataset.