**Edifice Data Extraction process:**

**Steps:**

1. **Execute 01\_sftp\_extract.sh bash script using CRON**.
   1. 01\_sftp\_extract.sh:
      1. This cron job extracts sftp files from SPS Commerce SFTP folder and uploads them to “adl://yetiadls.azuredatalakestore.net/clusters/data/raw/edifice/input”
   2. Rename files to remove space as Hadoop FS doesn’t accept space while copying.
   3. Schedule 01\_sftp\_extract.sh using xsbdsa SA using CRON every 6 hours.
2. **Spark program for processing text files**:
   1. HDFSUtil.scala: This scala program does HDFS file manipulations
      1. Checks HDFS file directory
      2. drop and recreate the directory
      3. list child directories with or without absolute path
      4. move files from source to target directory
      5. move child directories into target directory
      6. move input Files into archive directory
   2. edificeReport.scala: This scala program does below tasks in the order.
      1. Splits each file into 3 split files using “####HDR” keyword.
      2. Take retailer, account, year\_day and flag from header and add to each line.
      3. Remove tail from each starts with “/TRL/” from each split file.
      4. Create output schema by adding primary key “ID” to each line in the file.
   3. edificeLoader.scala:
      1. creates spark session and spark context
      2. adds lastUPD column to outputSchema
      3. create a data frame on top of it and write contents to CSV file:
         1. adl://yetiadls.azuredatalakestore.net/clusters/data/raw/edifice/output
3. **Spark-submit:**

spark-submit --class com.yeti.dwh.edifice.edificeLoader \

--master yarn \

/home/avinash\_r/edifice/JAR/edw\_2.11-1.1.6.jar \

adl://yetiadls.azuredatalakestore.net/clusters/data/raw/edifice/input \

adl://yetiadls.azuredatalakestore.net/clusters/data/raw/edifice/output \

adl://yetiadls.azuredatalakestore.net/clusters/data/raw/edifice/target \

adl://yetiadls.azuredatalakestore.net/clusters/data/raw/edifice/archive

spark-submit --class com.yeti.dwh.edifice.HDFSUtil \

--master yarn \

/home/avinash\_r/edifice/JAR/edw\_2.11-1.1.6.jar \

adl://yetiadls.azuredatalakestore.net/clusters/data/raw/edifice/input \

adl://yetiadls.azuredatalakestore.net/clusters/data/raw/edifice/output \

adl://yetiadls.azuredatalakestore.net/clusters/data/raw/edifice/target \

adl://yetiadls.azuredatalakestore.net/clusters/data/raw/edifice/archive

**Data Flow: (in order)**

1. **InputFiles**: adl://yetiadls.azuredatalakestore.net/clusters/data/raw/edifice/input
2. **ProcessedFiles**: adl://yetiadls.azuredatalakestore.net/clusters/data/raw/edifice/output
3. **HiveExternalTableFiles**: adl://yetiadls.azuredatalakestore.net/clusters/data/raw/edifice/target
4. **ArchivePath**: adl://yetiadls.azuredatalakestore.net/clusters/data/raw/edifice/archive

**Process Flow:**

1. 01\_sftp\_extract.sh:
   1. SFTP Folder -🡪 InputFiles
2. edificeLoader.scala:
   1. InputFiles -🡪 processedFiles
3. HDFSUtil.scala:
   1. processedFiles -🡪 HiveExternalTableFiles (compares partition to partition)
   2. InputFiles -🡪 ArchivePath

**Edifice Full Load:**

1. Copy yetidm.edifice table data into staging path (adl://yetiadls.azuredatalakestore.net/clusters/data/raw/edifice/fullload) using ADF.
2. Create hive external table (default.edifice\_stg) on top of
   1. adl://yetiadls.azuredatalakestore.net/clusters/data/raw/edifice/fullload
3. Create edw.edifice table by creating partitioned table.
   1. Partition columns: retailer followed by year\_day

**Performance:**

1. As we are doing file comparisons at HDFS file system level, this method can also be used for timeseries data.
2. HDFSUtil.scala can be re-used for as HDFS JAVA API.