Databricks Overview

Elements

- Workspaces
- Clusters
- Notebooks

Getting started

- Start a cluster.
- Upload `meat_consumption_kg_meat_per_capita_per_country.csv` as a data table.
- Fire up a new notebook.

Querying the table

```
%sql
select * from `meat_consumption_kg_meat_per_capita_per_country`
```

```
%sql
select count(*) from `meat_consumption_kg_meat_per_capita_per_country`
```

```
%python
df = spark.sql('select * from meat_consumption_kg_meat_per_capita_per_country')
display(df)
```

Magic commands

- `%scala`
- `%r`
- `%sql`
- `%sh`
- `%fs`
- `%md`

Databricks File System (DBFS)

- Storage is not persistent on an Apache Spark cluster.
- Instead, information is saved onto a **storage cluster**.
- This could be either DBFS or HDFS.

Navigating the file system

- `%fs ls`
- `%fs ls /databricks-datasets`
- `%fs head /databricks-datasets/airlines/README.md`
- `%fs`

The `dbutils` package

- The magic behind `%ls` is the `dbutils` package which is being executed behind the scenes.

```
dbutils.fs.ls("/databricks-datasets")
dbutils.fs.head("/databricks-datasets/airlines/README.md")
```

The `dbutils` package (cont.)

- The advantage of using `dbutils` directly is that you can invoke it in code.

```
files = dbutils.fs.ls("/")
for f in files:
    print(f.name)
```

```
fnames = [print(f.name) for f in dbutils.fs.ls("/")]
print(fnames)
```

Retrieving remote files with wget

```
from requests import get
with open('/tmp/f1.zip', "wb") as f:
  response = get('http://ergast.com/downloads/f1db_csv.zip')
  f.write(response.content)
```

```
from zipfile import ZipFile
with ZipFile('/tmp/f1.zip', 'r') as zip:
  files = zip.namelist()
  for file in files:
    print(file)
```

Retrieving remote files with wget (cont.)

```
with ZipFile('/tmp/f1.zip', 'r') as zip:
  zip.extract('seasons.csv','/tmp')
```

```
dbutils.fs.mv("file:/tmp/seasons.csv", "dbfs:/tmp/seasons.csv")
```

Retrieving remote files with wget (cont.)

- Writing the data into a table

```
df = spark \
.read \
.format("csv") \
.option("inferSchema","true") \
.option("header","false") \
.load("dbfs:/tmp/seasons.csv") \
.selectExpr("_c0 as year", "_c1 as url")
df.write.saveAsTable('seasons')
```

```
%sql
drop table test;
create temporary table test (year INT, url STRING) using csv options (path
"dbfs:/tmp/seasons.csv", header "false");
select * from test;
```

FileStore

- Everything you put in this folder is accessible through a web browser.
- `%fs cp /databricks-datasets/airlines/README.md /FileStore`
- This would be visible in
  - `https://<databricks-instance>/files/README.md`
  - `https://community.cloud.databricks.com/files/README.md?o=####`
- You can use FileStore to save a logo.
  - `displayHTML("<img src = '/files/images/logo.png'>")`
- To upload directly files to FileStore, the easiest way is to enable the option in Settings > Admin Console.

Schemas, databases and tables

- Tables in Databricks correspond to Apache Spark DataFrames.
- These are two-dimensional structures like spreadsheets or tables in a database.
- They can be **local** (stored in the local cluster) or **global** (available across all clusters and stored in Hive Metastore).
- All information about tables, columns and data types is stored in Hive Metastore.