Antra

People · Process · Performance

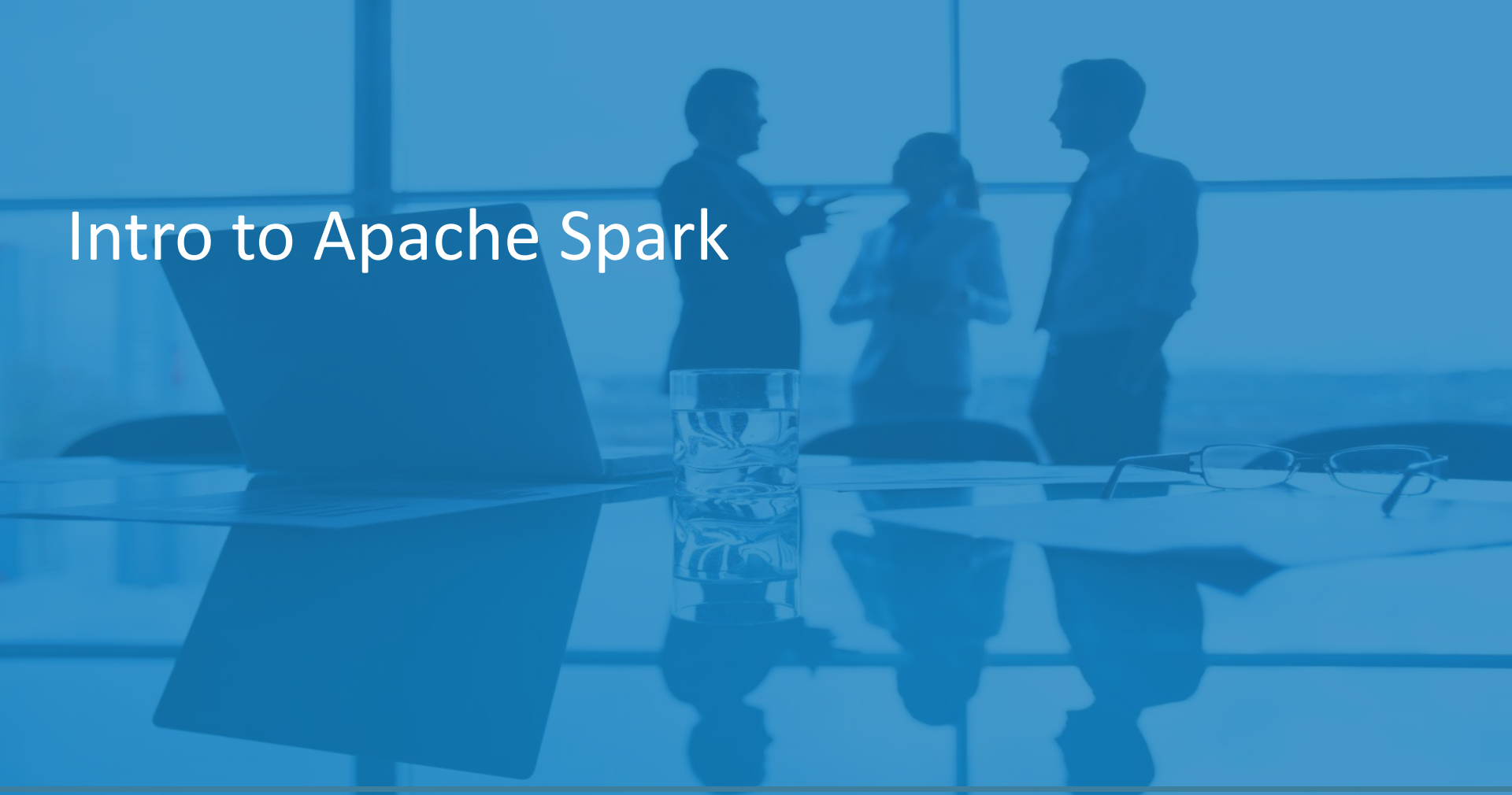
Copyright © 2021 Antra, Inc. All rights reserved.

Agenda

- Intro to Apache Spark
- Intro to Databricks
- Working with Databricks using Python
- Intro to Delta Lake
- Intro to Structured Streaming

Copyright © 2021 Antra, Inc. All rights reserved.

Intro to Apache Spark



Copyright © 2015 Antra, Inc. All rights reserved.

Antra

What Is Apache Spark?

- Unified processing engine that can do
  - Big Data Processing
    - SQL + Streaming + Graph
  - Machine Learning
- Fast, easy to use, sophisticated analytics
- In memory engine that's up to 100 times faster than Hadoop
- Largest open-source data project with 1000+ contributors
- Highly extensible with support for Scala, Java and Python alongside Spark SQL, GraphX, Streaming and MLlib

Copyright © 2021 Antra, Inc. All rights reserved.

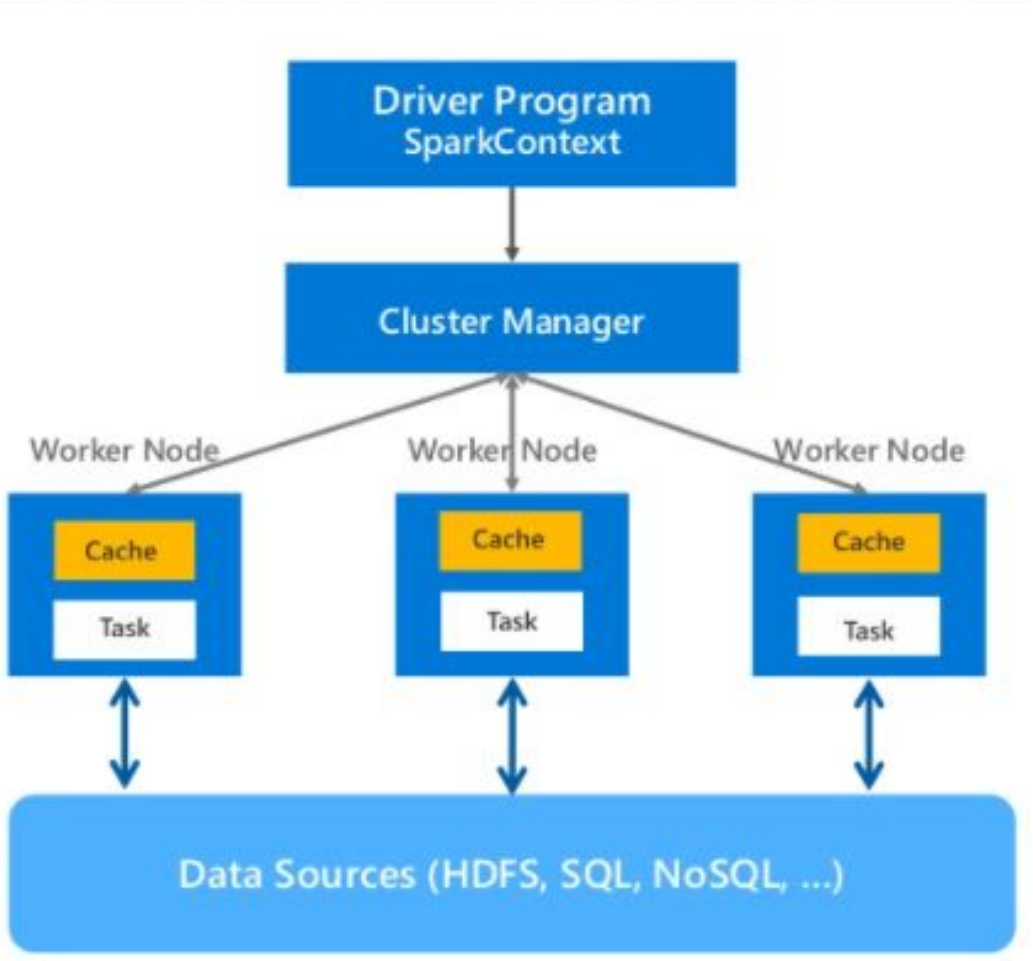
What Is Apache Spark?

- Written in Scala, which is built on top of the Java Virtual Machine(JVM) and Java runtime and thus cross-platform.
- Became ASF(Apache Software Foundation) Top-Level Project in 2014.
- Viewed by some as the successor of Hadoop.

Copyright © 2021 Antra, Inc. All rights reserved.

General Spark Cluster Architecture

- Driver runs the main function and executes parallel operations on the worker nodes
- Results are collected by the driver
- Worker nodes read and write data from/to data sources
- Worker nodes also cache transformed data in memory as RDDs(Resilient Data Sets)



Copyright © 2015 Antra, Inc. All rights reserved.

What is Spark used for?

- ETL operations
- Predictive analysis and machine learning
- Data access operations (such as SQL queries and visualizations)
- Text mining and processing
- Graph applications
- Pattern recognition
- Recommendation Engines
- …

It is now more a matter of what Spark can't do rather what you can do with Spark, and if what you want to do involves big data, chances are you can use a Spark approach.

Copyright © 2021 Antra, Inc. All rights reserved.

Spark SQL, DataFrames and Datasets

Copyright © 2015 Antra, Inc. All rights reserved.

Antra

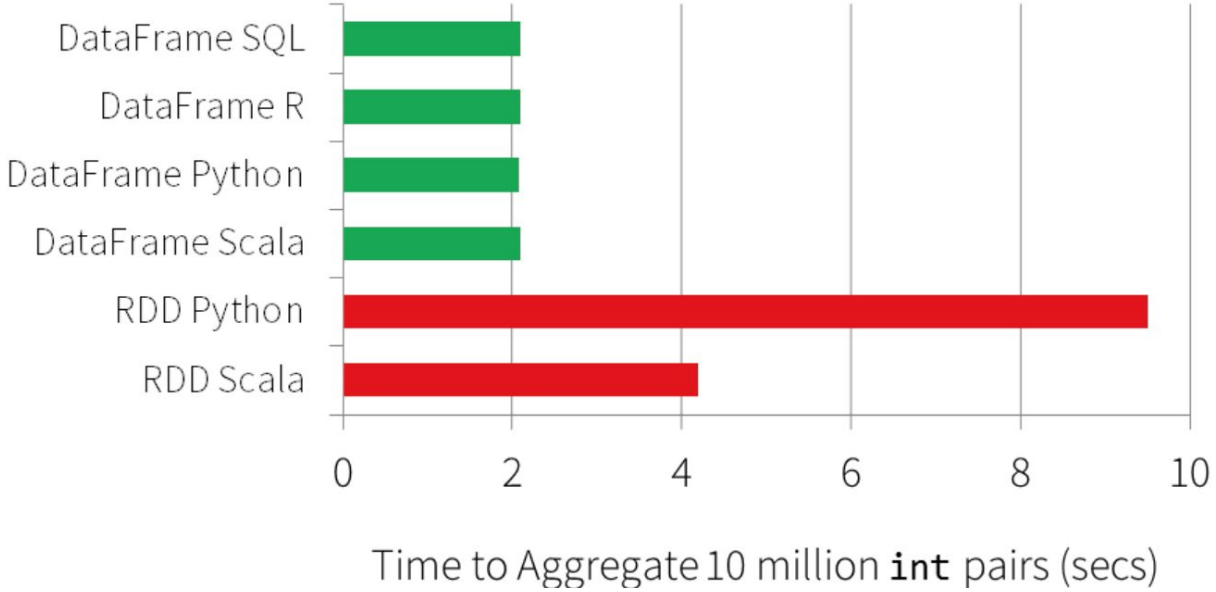
RDDs and DataFrames

- RDD
  - immutable once created and keep track of their lineage to enable failure recovery.
  - Resilient: Fault-tolerant
  - Distributed: Across multiple nodes
  - Dataset: Collection of partitioned data
- DataFrames
  - An abstraction above RDDs
  - 5-20x performance over RDDs due to Optimization

Copyright © 2021 Antra, Inc. All rights reserved.

RDD vs DataFrame performance

- Performance differences between languages are nearly nonexistence for the DataFrames API



Copyright © 2021 Antra, Inc. All rights reserved.

Spark SQL

- A Spark module for structured data processing
- Provides more information about the structure of both the data and the computation being performed.
  - Used internally to optimize performance.
- There are multiple ways to interact with Spark SQL including SQL and the Dataset API
  - The same execution engine is used regardless of API/language used

Copyright © 2021 Antra, Inc. All rights reserved.

SQL

- Spark SQL can be used to execute SQL queries

```
spark.sql("CREATE TABLE IF NOT EXISTS src (key INT, value STRING) USING hive");
spark.sql("LOAD DATA LOCAL INPATH 'examples/src/main/resources/kv1.txt' INTO TABLE src");

// Queries are expressed in HiveQL
spark.sql("SELECT * FROM src").show();
// +---+-------+
// |key|  value|
// +---+-------+
// |238|val_238|
// | 86| val_86|
// |311|val_311|
// ...
```

- “LOAD DATA LOCAL INPATH” is used to load data from a file or directory

Copyright © 2021 Antra, Inc. All rights reserved.

Datasets and DataFrames

- A Dataset is a distributed collection of data. Added in Spark 1.6.
  - Basically a more powerful version of RDD
  - Only available is Scala and Java
  - Many benefits of the Dataset API is already available in Python due to the dynamic nature of the language.
- A DataFrame is a Dataset organized into named columns
  - Conceptually equivalent to a table in a relational DB or a dataframe in R/Python
  - Optimized for performance under the hood
  - Can be constructed from
    - structured data file
    - Tables in hive
    - External DBs
    - RDDs
  - Available in Scala, Java, Python and R.

Copyright © 2021 Antra, Inc. All rights reserved.

Challenges with Spark

- Requires a distributed storage system
- Usually needs other technologies to be present (e.g. Hadoop, Mesos)
- Security model is limited
- Performance tuning is tricky
- Capacity management and cost optimization are hard

Copyright © 2021 Antra, Inc. All rights reserved.

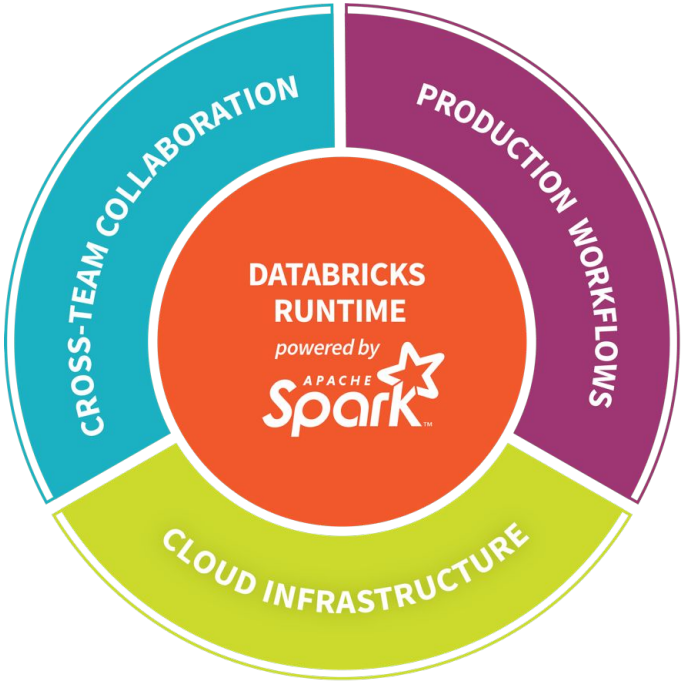
Intro to Azure Databricks

Copyright © 2015 Antra, Inc. All rights reserved.

Antra

Databricks aims to solve the challenges of Apache Spark

- **Fully managed Spark Clusters**
- Cloud based
- **Interactive workspace** for exploration and visualization
- Production pipeline scheduler
- Enterprise-grade security
- Fast



Copyright © 2015 Antra, Inc. All rights reserved.

Antra

What is Azure Databricks?

- Managed service on Azure
- Data analytics platform based on Apache Spark that's optimized for Azure
  - **Azure Databricks SQL Analytics**
    - Easy-to-use platform for
      - Running SQL queries on data lakes
      - Creating multiple visualization types to explore query results
      - Building and sharing dashboards
  - **Azure Databricks Workspace**
    - Interactive workspace for collaboration between data engineers, data scientists, and machine learning engineers.

Copyright © 2015 Antra, Inc. All rights reserved.

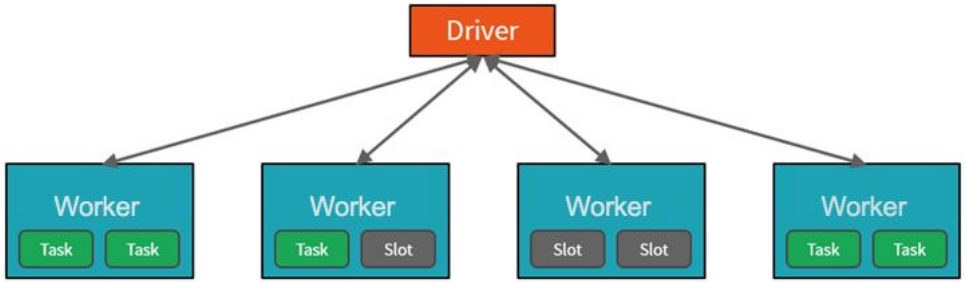
Azure Databricks is highly optimized

- High-speed connectors to Azure storage services like Blob Store and Data Lake
- Auto-scaling and auto-termination
- Caching
- Indexing
- Automatic query optimization

Copyright © 2021 Antra, Inc. All rights reserved.

Databricks High-level Architecture

- Recall Spark utilizes a **master-worker** architecture
- In Databricks, the **notebook** interface is the **driver** program
  - Driver program contains the main loop for the program and creates distributed datasets on the cluster, then applies operations (transformations & actions) to those datasets.
  - Driver programs access Apache Spark through a SparkSession object.
- Azure manages the clusters and auto-scales/auto-terminates based on our usage and settings.



*each thread on the worker is a slot

Copyright © 2021 Antra, Inc. All rights reserved.

Antra

Databricks High-level Architecture

- All Databricks resources are grouped into a managed resource group within your Azure subscription
  - Driver & worker VMs
  - Virtual network
  - Security group
  - Storage account
- Metadata of the clusters, such as scheduled jobs, is stored in an Azure Database with geo-replication for fault tolerance
- Clusters are ran with Azure VMs and Azure Kubernetes Services
- Databricks File System(DBFS) is built on top of Azure Blob storage.

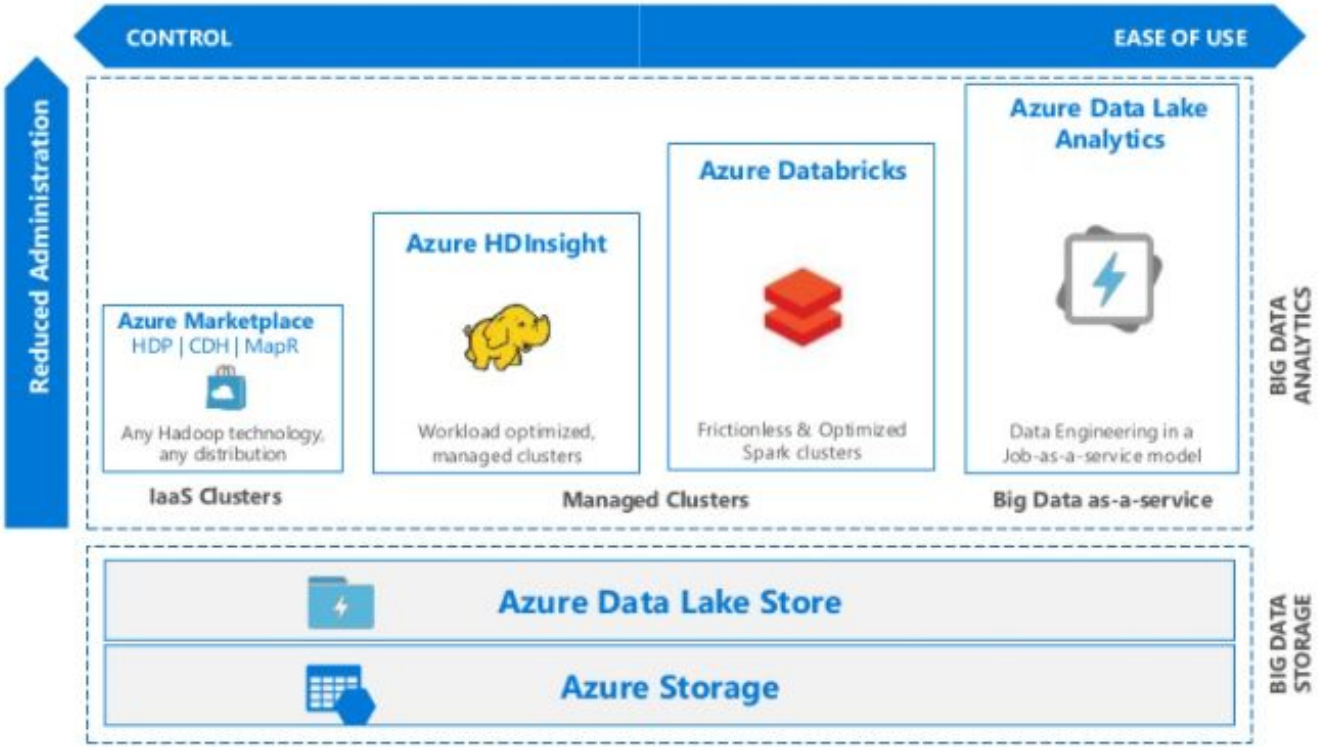
Copyright © 2021 Antra, Inc. All rights reserved.

Driver , worker, jobs, slots and tasks

- The driver is the JVM in which our application runs
- Spark achieves high performance by parallelism on two levels
  - Worker and Slot
    - Jobs are divided into tasks by the driver and sent to slots on workers for parallel processing
- Driver will also decide how to partition the data so it can be distributed across workers
  - Once execution starts, each task will fetch the partition of data assigned to it from the original data source
- Multiple jobs might be needed depending on the work required.
  - data.sort(...).filter(...)
    - Job1 -> filter the data
      - Tasks -> filter certain partition of data
    - Job2 -> sort the data
      - Tasks -> sort certain partition of data

Copyright © 2021 Antra, Inc. All rights reserved.

Other Azure Big Data Solutions



Copyright © 2015 Antra, Inc. All rights reserved.

Antra

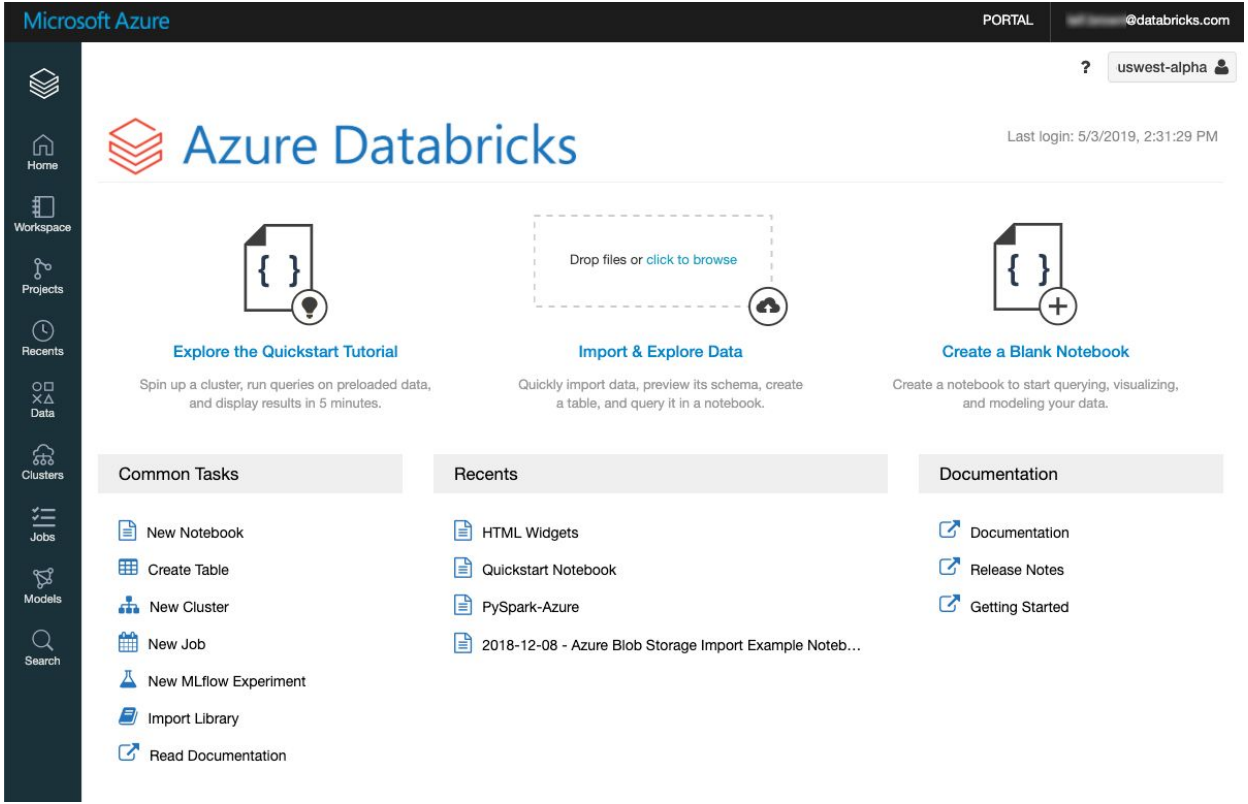
Get started with Azure Databricks

Copyright © 2015 Antra, Inc. All rights reserved.

Antra

Get started with Azure Databricks

- Provision directly from the Azure Portal like any other Azure Services
- Any Azure user with the appropriate subscription and authorization can provision Azure Databricks service

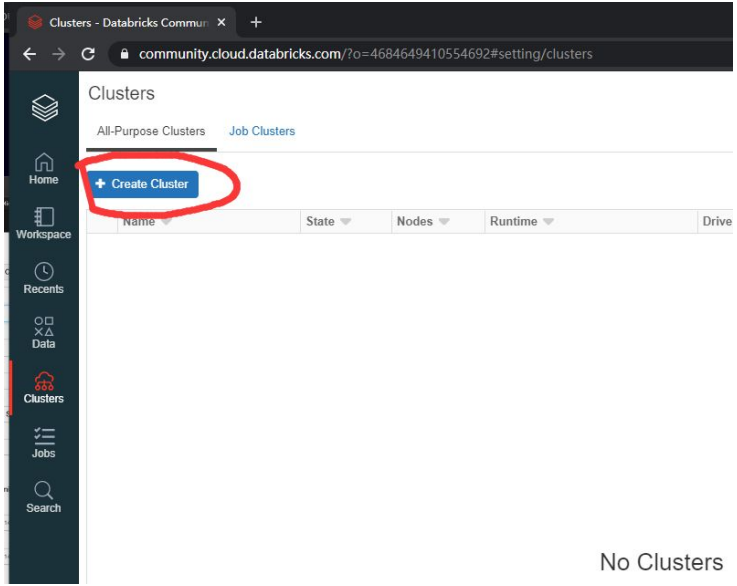


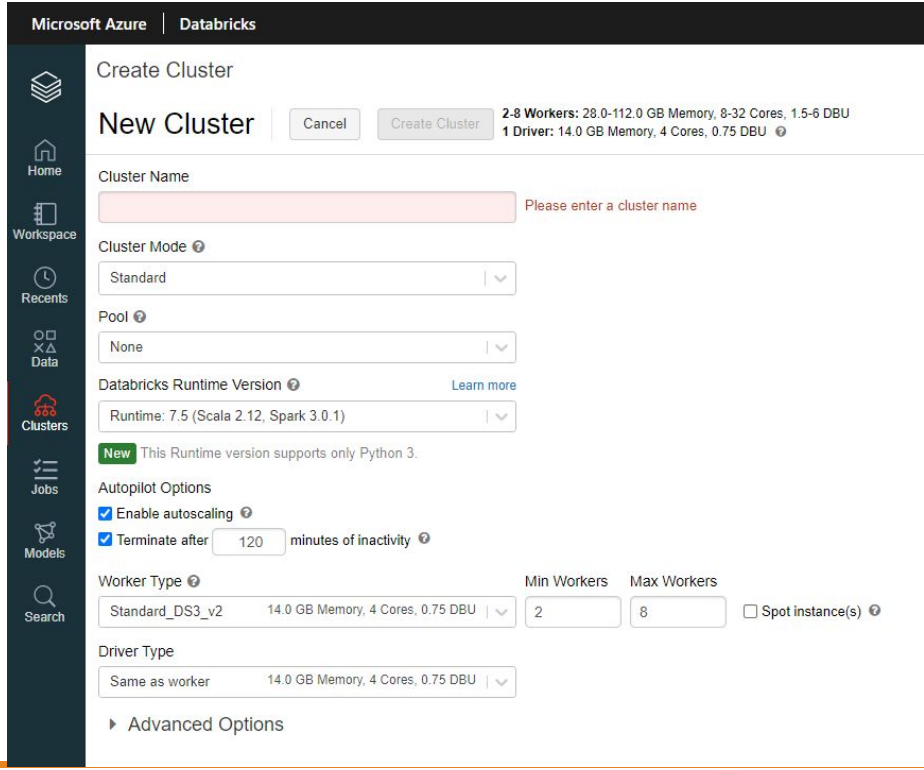
Copyright © 2015 Antra, Inc. All rights reserved.

Antra

Clusters

- Where all the computation happens





Copyright © 2021 Antra, Inc. All rights reserved.

Antra

Cluster modes

- High Concurrency:
  - Optimized to run concurrent SQL, Python, and R workloads. Does not support Scala. Previously known as Serverless.
- Standard:
  - Recommended for single-user clusters. Can run SQL, Python, R, and Scala workloads.
- Single Node:
  - Clusters with no workers. Recommended for single-user clusters computing on small data volumes.

Copyright © 2021 Antra, Inc. All rights reserved.

Pool

- Used to reduce cluster start up time
- When clusters are attached to a pool, the cluster will allocate its driver and worker nodes from the pool.
  - New instances are automatically added by the instance provider if the pool doesn't have enough idle resources
  - When an attached cluster is terminated, the instances it used are returned to the pool and can be used by a different cluster.

Copyright © 2021 Antra, Inc. All rights reserved.

Databricks Runtime Version

- Standard Runtimes also comes with some of the most popular libraries for Java/Scala/R/Python
- ML Runtimes are basically standard runtimes with some additional popular *machine learning and data science* libraries built in.
  - TensorFlow, PyTorch, XGBoost…

Copyright © 2021 Antra, Inc. All rights reserved.

Workspaces

- Workspaces- sort of like Directories- are a convenient way to organize an user's Notebook, Libraries and Dashboards.
- Items in workspaces are organized into hierarchical folders. Folders can hold Libraries, Notebooks, Dashboard or more folders
- Every user has one directory that is private and unshared
- Fine grained access control can be defined on workspaces to enable secure collaborations

Copyright © 2015 Antra, Inc. All rights reserved.

Antra

Libraries

- Containers to hold all the Python, R, Java/Scala libraries
- Resides within workspaces or folders
- Created by importing the source code
- After importing, libraries are immutable
- Customize installation of libraries with Init Scripts by writing custom UNIX scripts
- Can be managed via the Library API

Copyright © 2015 Antra, Inc. All rights reserved.

Antra

Notebooks

- Notebooks are not only for authoring Spark applications but can be run directly on clusters
- Well suited for prototyping, rapid development, exploration, discovery and iterative developments

```
> from multiprocessing.pool import ThreadPool
  pool = ThreadPool(10)
```

Command took 0.07s

```
> pool.map(
    lambda path: dbutils.notebook.run(
      "/Users/eric/etl-test",
      timeout_seconds = 60,
      arguments = {"input-data": path}),
    ["/mnt/data-east", "/mnt/data-west", "/mnt/data-central"])
```

Notebook job #10931
Notebook job #10932
Notebook job #10933

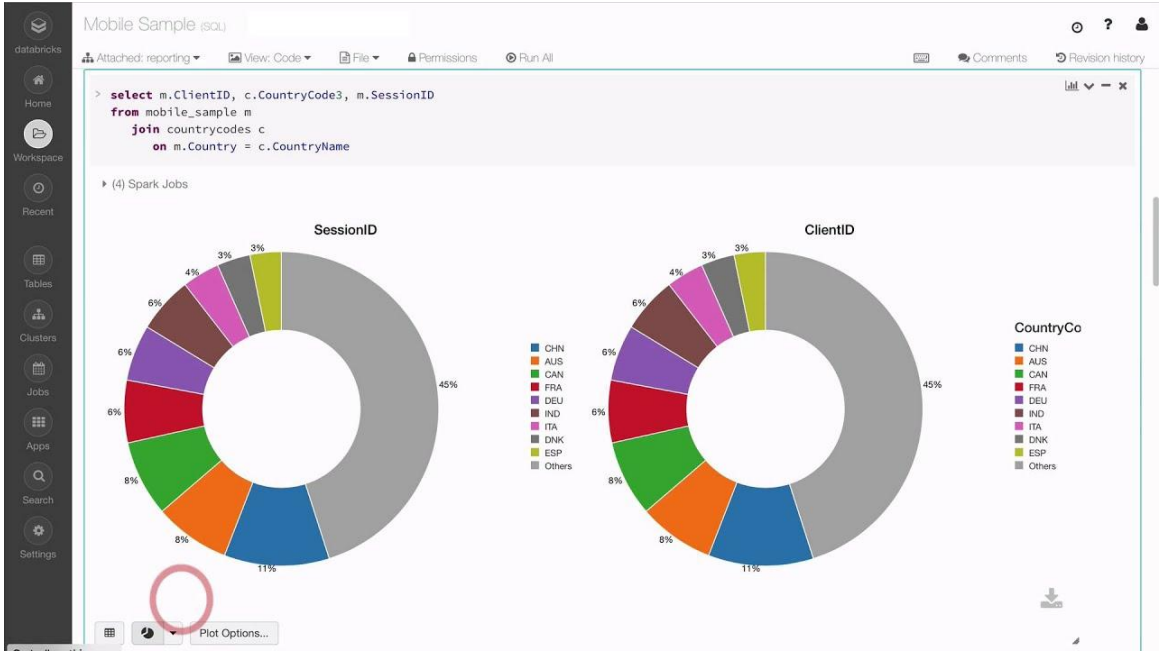
```
Out[30]:
[u'984939 records processed',
 u'788148 records processed',
 u'164705 records processed']
```

Command took 13.81s

Copyright © 2015 Antra, Inc. All rights reserved.

Visualizations

- All notebooks, regardless of their language, support Databricks visualizations
- Visualizations are rendered inside the notebook in-place
- Visualizations are written in HTML
  - HTML of the entire notebook can be saved
  - When using Matplotlib, plots are rendered as images
- Plot type can be easily changed in the selection menu



Copyright © 2015 Antra, Inc. All rights reserved.

Antra

Jobs

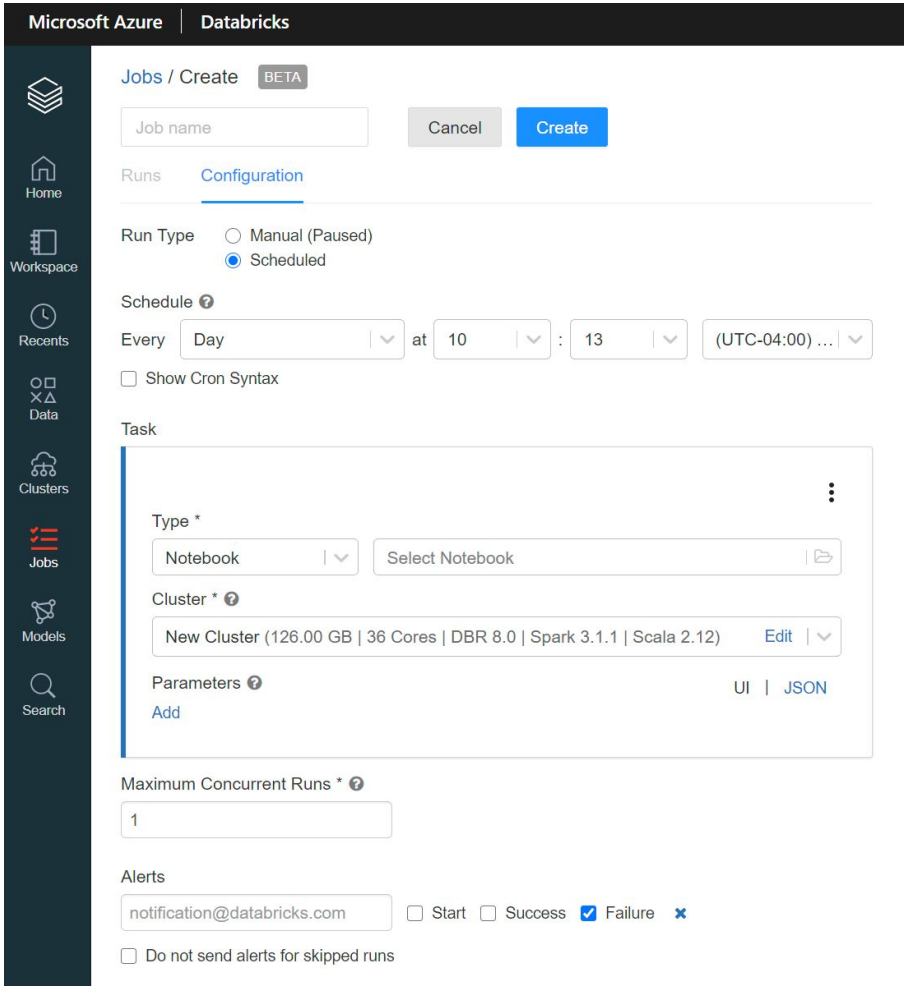
- Spark application code is submitted as a ‘job’ for execution on Azure Databricks clusters
- Job executes either ‘Notebooks’ or ‘Jars’
- Azure Databricks provide a comprehensive set of graphical tools to create, manage and monitor jobs

Copyright © 2015 Antra, Inc. All rights reserved.

Antra

Jobs

- You can schedule the execution of jobs when you create them.



Copyright © 2021 Antra, Inc. All rights reserved.

Antra

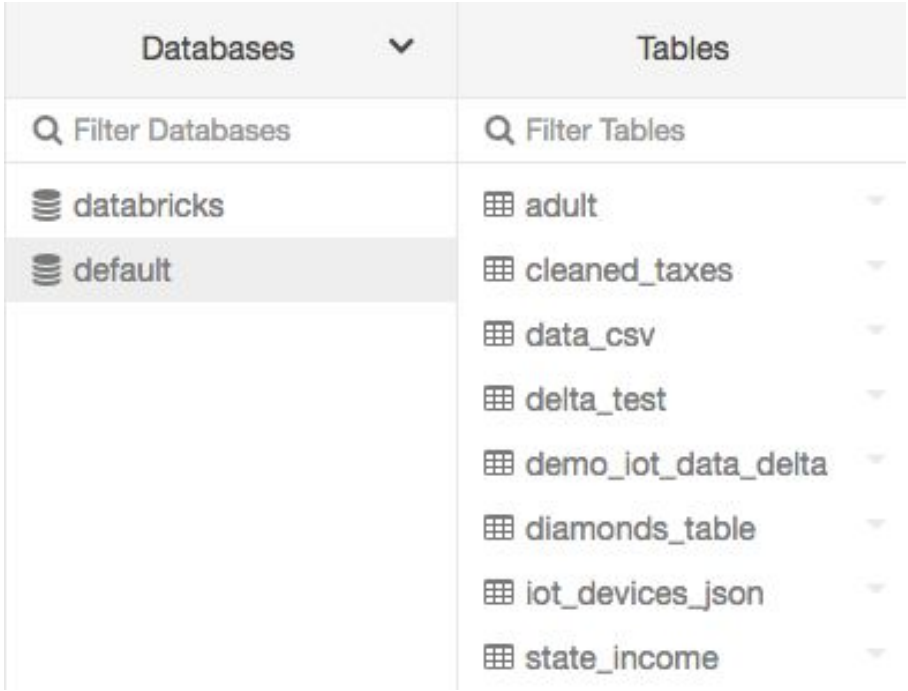
Databricks File System

- Databricks File System (DBFS) is a distributed file system mounted into an Azure Databricks workspace and available on Azure Databricks clusters
- Lifetime of files in the DBFS are NOT tied to the lifetime of clusters
- We can access object stores by mounting them into DBFS

Copyright © 2021 Antra, Inc. All rights reserved.

Databases and tables

- Tables are defined using the GUI in the console or Programmatically using APIs or Notebooks
- Uses the Hive metastore to manage tables, and support all file formats and Hive data sources
- Any Spark operation can be applied to tables



Copyright © 2015 Antra, Inc. All rights reserved.

Antra

dbutils

- This module provides various utilities for users to interact with the rest of Databricks.
- Use **dbutils.help()** to get more details
- Get help for each sub utility
  - dbutils.fs.help()
  - dbutils.meta.help()
  - dbutils.notebook.help()
  - dbutils.widgets.help()
- Magic command **%fs** is equivalent to **dbutils.fs**
  - %fs mount source: String, mountPoint: String, encryptionType: String = "", owner: String = null, extraConfigs: Map = Map.empty[String, String]

Copyright © 2021 Antra, Inc. All rights reserved.

dbutil.fs.mount()

- mount(source: String, mountPoint: String, encryptionType: String = "", owner: String = null, extraConfigs: Map = Map.empty[String, String]): boolean
- We can use **dbutils.fs.mounts()** to check the mounts we have

```
1  mounts = dbutils.fs.mounts()
2
3  for mount in mounts:
4    print(mount.mountPoint + " >> " + mount.source)
5
6  print("-"*80)
```

```
/mnt/training >> s3a://databricks-corp-training/common
/databricks-datasets >> databricks-datasets
/databricks/mlflow-tracking >> databricks/mlflow-tracking
/databricks-results >> databricks-results
/databricks/mlflow-registry >> databricks/mlflow-registry
/ >> DatabricksRoot
--------------------------------------------------------------------------------
```

Copyright © 2021 Antra, Inc. All rights reserved.

dbutils.fs.ls()

- We can use **dbutils.fs.ls** to get the **FileInfo** objects of the directories we have
  - **FileInfo** contains
    - path -> path of the file or directory
    - is_dir -> whether the path points to a directory
    - file_size -> length of the file in bytes or zero if the path is a dir
    - modification_time -> last time, in epoch milliseconds, the file or directory was modified

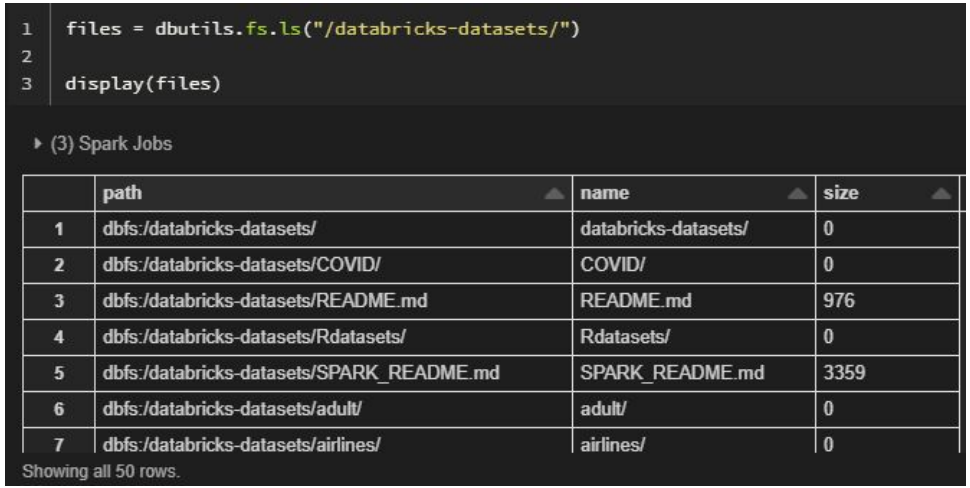
```
1  files = dbutils.fs.ls("/databricks-datasets")
2
3  for fileInfo in files:
4    print(fileInfo.path)
5
6  print("-"*80)
```

```
dbfs:/databricks-datasets/
dbfs:/databricks-datasets/COVID/
dbfs:/databricks-datasets/README.md
dbfs:/databricks-datasets/Rdatasets/
dbfs:/databricks-datasets/SPARK_README.md
```

Copyright © 2021 Antra, Inc. All rights reserved.

display()

- Databricks specific command overloaded with many different capabilities
  - Presents up to 1000 records
  - Exporting data as CSV
  - Rendering a multitude of different graphs
  - Rendering geo-located data on a world map
  - ….
- It’s a great tool for previewing our data in a notebook



```
1  files = dbutils.fs.ls("/databricks-datasets/")
2
3  display(files)
```

▸ (3) Spark Jobs

	path	name	size
1	dbfs:/databricks-datasets/	databricks-datasets/	0
2	dbfs:/databricks-datasets/COVID/	COVID/	0
3	dbfs:/databricks-datasets/README.md	README.md	976
4	dbfs:/databricks-datasets/Rdatasets/	Rdatasets/	0
5	dbfs:/databricks-datasets/SPARK_README.md	SPARK_README.md	3359
6	dbfs:/databricks-datasets/adult/	adult/	0
7	dbfs:/databricks-datasets/airlines/	airlines/	0

Showing all 50 rows.

Copyright © 2021 Antra, Inc. All rights reserved.

Magic Commands

- Identified by a single **%** symbol at the start of a cell
  - Use %sh to execute shell commands on the driver

  - Use **%python/%scala/%sql/%r** to execute code in languages other than the notebook's default

Copyright © 2021 Antra, Inc. All rights reserved.

Antra

Magic Commands

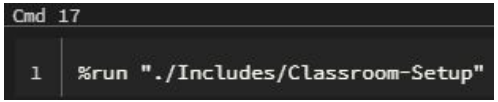
- Use **%md** to render Markdown in a cell
  - # / ## / ###
    - title one /two /three
  - *
    - Unordered list
  - 0.
    - Ordered list
  - **content**
    - Bold content
  - *content*
    - Italicize content
  - ![title](image-url)
    - Render image
  - You can also render tables by constructing a table using | and -

```
| col1  | col2| col3 |
|-------|-----|------|
| 1     | one  | .   |
| 2     | two  | ..  |
| 3     | three| ... |
```

col1	col2	col3
1	one	.
2	two	..
3	three	...

Copyright © 2021 Antra, Inc. All rights reserved.

Magic Commands

- Use **%run** to run another notebook within the current notebook
  - 

Copyright © 2021 Antra, Inc. All rights reserved.

Antra

Spark SQL

- Distributed SQL query engine for processing **structured data**
- Can query data in external databases, structured data files, Hive tables and more
- Both SQL and HiveQL are supported for querying
- Has bindings in Python, Scala and Java
- Has built-in support for structured streaming

Copyright © 2015 Antra, Inc. All rights reserved.

Spark ML

- Enables parallel, distributed ML for large datasets on Spark Clusters
- Offers a set of parallelized machine learning algorithms(MMLSpark, Spark ML, Deep Learning, SparkR)
- Supports Model Selection(hyperparameter tuning) using Cross Validation and Train-Validation Split
- Supports Java, Scala or Python apps using DataFrame-based API
  - Uniform APIs across ML algorithms and languages
  - ML pipelines- combining multiple algorithms into a single pipeline
  - Optimizations through Tungsten and Catalyst
- Spark Mllib comes pre-installed on Azure Databricks
- 3rd party library supports(H20 Sparkling Water, SciKit-learn and XGBoost)

Copyright © 2015 Antra, Inc. All rights reserved.

Spark Structured Streaming

- Unifies streaming, interactive and batch queries—a single API for both static bounded data and streaming unbounded data.
- Runs on Spark SQL. Uses the Spark SQL Dataset/DataFrame API used for batch processing of static data.
- Runs incrementally and continuously and updates the results as data streams in.
- Supports app development in Scala, Java, Python and R.
- Supports streaming aggregations, event-time windows, windowed grouped aggregation, stream-to-batch joins.
- Features streaming deduplication, multiple output modes and APIs for managing/monitoring streaming queries.
- Built-in sources: Kafka, File source (json, csv, text, parquet)

Copyright © 2015 Antra, Inc. All rights reserved.

Working with Databricks using SparkSession

Copyright © 2015 Antra, Inc. All rights reserved.

Antra

SparkSession

- The entry point to programming Spark with the Dataset and DataFrame API.
- Can be used to
  - create DataFrame
  - register DataFrame as tables
  - execute SQL over tables
  - cache tables
  - read parquet files
- We can build a SparkSession using the Builder API

```
>>> spark = SparkSession.builder \
...     .master("local") \
...     .appName("Word Count") \
...     .config("spark.some.config.option", "some-value") \
...     .getOrCreate()
```

Copyright © 2021 Antra, Inc. All rights reserved.

SparkSession Methods

createDataFrame(data[, schema, …])	Creates a **DataFrame** from an **RDD**, a list or a **pandas.DataFrame**.
getActiveSession()	Returns the active SparkSession for the current thread, returned by the builder
newSession()	Returns a new SparkSession as new session, that has separate SQLConf, registered temporary views and UDFs, but shared SparkContext and table cache.
range(start[, end, step, numPartitions])	Create a **DataFrame** with single **pyspark.sql.types.LongType** column named id, containing elements in a range from start to end (exclusive) with step value step.
sql(sqlQuery)	Returns a **DataFrame** representing the result of the given query.
stop()	Stop the underlying **SparkContext**.
table(tableName)	Returns the specified table as a **DataFrame**.

Copyright © 2021 Antra, Inc. All rights reserved.

SparkSession Attributes

builder	A class attribute having a **Builder** to construct **SparkSession** instances.
catalog	Interface through which the user may create, drop, alter or query underlying databases, tables, functions, etc.
conf	Runtime configuration interface for Spark.
read	Returns a **DataFrameReader** that can be used to read data in as a **DataFrame**.
readStream	Returns a **DataStreamReader** that can be used to read data streams as a streaming **DataFrame**.
sparkContext	Returns the underlying **SparkContext**.
streams	Returns a **StreamingQueryManager** that allows managing all the **StreamingQuery** instances active on this context.
udf	Returns a **UDFRegistration** for UDF registration.
version	The version of Spark on which this application is running.

Copyright © 2021 Antra, Inc. All rights reserved.

Read Data in Azure Databricks

csv: dbfs:/databricks-datasets/COVID/covid-19-data/us-counties.csv
json: dbfs:/databricks-datasets/learning-spark-v2/blogs.json
tsv:dbfs:/databricks-datasets/wikipedia-datasets/data-001/pageviews/raw/pageviews_by_second.tsv

Copyright © 2015 Antra, Inc. All rights reserved.

Antra

DataFrameReader

- We can use DataFrameReader to read many different types of data and turn them into DataFrames.

DataFrameReader.csv(path[, schema, sep, …])	Loads a CSV file and returns the result as a **DataFrame**.
DataFrameReader.format(source)	Specifies the input data source format.
DataFrameReader.jdbc(url, table[, column, …])	Construct a **DataFrame** representing the database table named table accessible via JDBC URL url and connection properties.
DataFrameReader.json(path[, schema, …])	Loads JSON files and returns the results as a **DataFrame**.
DataFrameReader.load([path, format, schema])	Loads data from a data source and returns it as a **DataFrame**.
DataFrameReader.option(key, value)	Adds an input option for the underlying data source.
DataFrameReader.options(**options)	Adds input options for the underlying data source.
DataFrameReader.orc(path[, mergeSchema, …])	Loads ORC files, returning the result as a **DataFrame**.
DataFrameReader.parquet(*paths, **options)	Loads Parquet files, returning the result as a **DataFrame**.
DataFrameReader.schema(schema)	Specifies the input schema.
DataFrameReader.table(tableName)	Returns the specified table as a **DataFrame**.

Copyright © 2021 Antra, Inc. All rights reserved.

CSV Example

- Displaying all the files under a path in DBFS

```
1  %fs ls databricks-datasets/COVID/covid-19-data/
```

	path	name	size
1	dbfs:/databricks-datasets/COVID/covid-19-data/.git/	.git/	0
2	dbfs:/databricks-datasets/COVID/covid-19-data/.github/	.github/	0
3	dbfs:/databricks-datasets/COVID/covid-19-data/.gitignore	.gitignore	10
4	dbfs:/databricks-datasets/COVID/covid-19-data/LICENSE	LICENSE	1289
5	dbfs:/databricks-datasets/COVID/covid-19-data/NEW-YORK-DEATHS-METHODOLOGY.md	NEW-YORK-DEATHS-METHODOLOGY.md	2771
6	dbfs:/databricks-datasets/COVID/covid-19-data/NYT-readme.md	NYT-readme.md	1748
7	dbfs:/databricks-datasets/COVID/covid-19-data/PROBABLE-CASES-NOTE.md	PROBABLE-CASES-NOTE.md	3162
8	dbfs:/databricks-datasets/COVID/covid-19-data/README.md	README.md	22959
9	dbfs:/databricks-datasets/COVID/covid-19-data/colleges/	colleges/	0
10	dbfs:/databricks-datasets/COVID/covid-19-data/excess-deaths/	excess-deaths/	0
11	dbfs:/databricks-datasets/COVID/covid-19-data/live/	live/	0
12	dbfs:/databricks-datasets/COVID/covid-19-data/mask-use/	mask-use/	0
13	dbfs:/databricks-datasets/COVID/covid-19-data/us-counties-recent.csv	us-counties-recent.csv	3977154
14	dbfs:/databricks-datasets/COVID/covid-19-data/us-counties.csv	us-counties.csv	45452100
15	dbfs:/databricks-datasets/COVID/covid-19-data/us-states.csv	us-states.csv	703102
16	dbfs:/databricks-datasets/COVID/covid-19-data/us.csv	us.csv	10269

Showing all 16 rows.

Copyright © 2021 Antra, Inc. All rights reserved.

CSV Example

- Use **%fs head** to peek at the beginning of the file

Cmd 21

```
1  %fs head databricks-datasets/COVID/covid-19-data/us-counties.csv
```

```
[Truncated to first 65536 bytes]
date,county,state,fips,cases,deaths
2020-01-21,Snohomish,Washington,53061,1,0
2020-01-22,Snohomish,Washington,53061,1,0
2020-01-23,Snohomish,Washington,53061,1,0
2020-01-24,Cook,Illinois,17031,1,0
2020-01-24,Snohomish,Washington,53061,1,0
2020-01-25,Orange,California,06059,1,0
2020-01-25,Cook,Illinois,17031,1,0
2020-01-25,Snohomish,Washington,53061,1,0
2020-01-26,Maricopa,Arizona,04013,1,0
2020-01-26,Los Angeles,California,06037,1,0
2020-01-26,Orange,California,06059,1,0
2020-01-26,Cook,Illinois,17031,1,0
2020-01-26,Snohomish,Washington,53061,1,0
2020-01-27,Maricopa,Arizona,04013,1,0
2020-01-27,Los Angeles,California,06037,1,0
2020-01-27,Orange,California,06059,1,0
2020-01-27,Cook,Illinois,17031,1,0
2020-01-27,Snohomish,Washington,53061,1,0
2020-01-28,Maricopa,Arizona,04013,1,0
```

Copyright © 2021 Antra, Inc. All rights reserved.

CSV Example

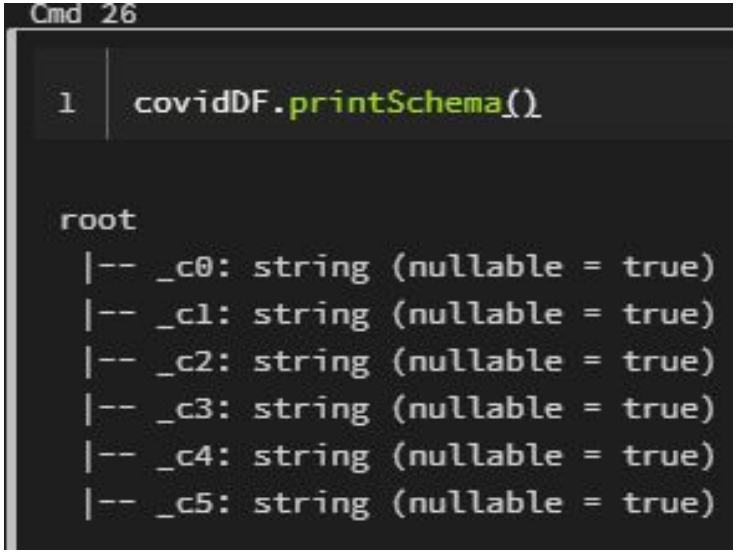
- Using **read.csv()** we get a dataframe with 6 columns

```
Cmd 25

1  covidDF = (spark.read
2              .csv("/databricks-datasets/COVID/covid-19-data/us-counties.csv")
3            )

▸ (1) Spark Jobs
▾ covidDF: pyspark.sql.dataframe.DataFrame
      _c0: string
      _c1: string
      _c2: string
      _c3: string
      _c4: string
      _c5: string
```

Copyright © 2021 Antra, Inc. All rights reserved.

CSV Example

- Dataframe has many methods and attributes
- **printSchema** will print the schema of the DF in tree format



Copyright © 2021 Antra, Inc. All rights reserved.

Antra

CSV Example

- Recall when we peeked the data, the first row is the header
- We can tell spark to treat first row as header by setting “header” to “true” in option

Cmd 34

```
1  covidDF = (spark.read
2             .option("header","true")
3             .csv("/databricks-datasets/COVID/covid-19-data/us-counties.csv")
4             .printSchema()
5            )
```

▸ (1) Spark Jobs

```
root
 |-- date: string (nullable = true)
 |-- county: string (nullable = true)
 |-- state: string (nullable = true)
 |-- fips: string (nullable = true)
 |-- cases: string (nullable = true)
 |-- deaths: string (nullable = true)
```

Copyright © 2021 Antra, Inc. All rights reserved.

CSV Example

- According to the peak, a few columns should be integer values
- We can add the “inferSchema” option to have spark infer the schema automatically

```
1  covidDF = (spark.read
2             .option("header","true")
3             .option("inferSchema","true")
4             .csv("/databricks-datasets/COVID/covid-19-data/us-counties.csv")
5             .printSchema()
6            )
```

▸ (2) Spark Jobs

```
root
 |-- date: string (nullable = true)
 |-- county: string (nullable = true)
 |-- state: string (nullable = true)
 |-- fips: integer (nullable = true)
 |-- cases: integer (nullable = true)
 |-- deaths: integer (nullable = true)
```

Copyright © 2021 Antra, Inc. All rights reserved.

CSV Example

- Inferring the schema will take longer and use more resource
- Spark will have to read the data twice
  - Read data to infer schema
  - Read data again and map to the inferred schema

Copyright © 2021 Antra, Inc. All rights reserved.

CSV Example

- Instead of inferring the schema, we can declare the schema ourselves
  - **Datatypes**: https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql.html#data-types

Cmd 43

```
1  from pyspark.sql.types import *
2
3  covidSchema = StructType([
4    StructField("date", DateType(), True),
5    StructField("county", StringType(), True),
6    StructField("state", StringType(), True),
7    StructField("fips", IntegerType(), True),
8    StructField("cases", IntegerType(), True),
9    StructField("deaths", IntegerType(), True),
10 ])
```

```
1  covidDF = (spark.read
2             .option("header","true")
3             .schema(covidSchema)
4             .csv("/databricks-datasets/COVID/covid-19-data/us-counties.csv")
5             .printSchema()
6            )
```

```
root
 |-- date: date (nullable = true)
 |-- county: string (nullable = true)
 |-- state: string (nullable = true)
 |-- fips: integer (nullable = true)
 |-- cases: integer (nullable = true)
 |-- deaths: integer (nullable = true)
```

Copyright © 2021 Antra, Inc. All rights reserved.

CSV Example

- We only want data from Orange county California.
- Nothing happens here because sort and filter are lazy

Cmd 4

```
1  (covid_df
2   .sort(covid_df["date"].desc())
3   .filter((covid_df["county"] == "Orange") & (covid_df["state"] == "California")))
```

```
Out[27]: DataFrame[date: string, county: string, state: string, fips: int, cases: int, deaths: int]
```

Copyright © 2021 Antra, Inc. All rights reserved.

Transformations vs Actions in Spark

- Transformations are LAZY
  - Transformations are functions that produce new RDD from existing RDDs
  - Transformations are executed only when we call an action
  - There are two types of transformations
    - Narrow
      - all the elements that are required to compute the records in single partition live in the single partition of parent RDD (map,filter,sample….)
    - Wide
      - the elements that are required to compute the records in the single partition may live in many partitions of parent RDD(Distinct,join,GroupByKey...)

Copyright © 2021 Antra, Inc. All rights reserved.

Transformations vs Actions in Spark

- Actions are EAGER
  - Actions are operations that give non-RDD values
  - Values of action are stored to drivers or to the external storage system.
  - Examples
    - Count
    - Collect -> often used to check if the entire RDD can fit in the memory of the driver
    - reduce

Copyright © 2021 Antra, Inc. All rights reserved.

	General	Math / Statistical	Set Theory / Relational	Data Structure / I/O
Transformations	map filter flatMap mapPartitions mapPartitionsWithIndex groupBy sortBy	sample randomSplit	union intersection subtract distinct cartesian zip	keyBy zipWithIndex zipWithUniqueID zipPartitions coalesce repartition repartitionAndSortWithinPartitions pipe
Actions	reduce Collect head show aggregate fold first take foreach top treeAggregate treeReduce foreachPartition collectAsMap toLocalIterator	count takeSample max min sum histogram mean variance stdev sampleVariance countApprox countApproxDistinct	takeOrdered	saveAsTextFile saveAsSequenceFile saveAsObjectFile saveAsHadoopDataset saveAsHadoopFile saveAsNewAPIHadoopDataset saveAsNewAPIHadoopFile

Copyright © 2021 Antra, Inc. All rights reserved.

Cmd 5

```
1  display(covid_df
2   .sort(covid_df["date"].desc())
3   .filter((covid_df["county"] == "Orange") & (covid_df["state"] == "California")))
```

▼ (1) Spark Jobs
▼ Job 24 View (Stages: 1/1)
Stage 24: 4/4

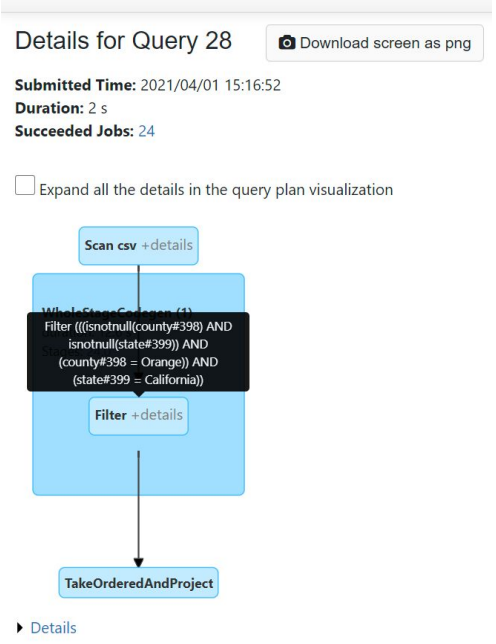
	date	county	state	fips	cases	deaths
1	2021-03-11	Orange	California	6059	263279	4379
2	2021-03-10	Orange	California	6059	263111	4346
3	2021-03-09	Orange	California	6059	262995	4313
4	2021-03-08	Orange	California	6059	262849	4252
5	2021-03-07	Orange	California	6059	262674	4226
6	2021-03-06	Orange	California	6059	262550	4173
7	2021-03-05	Orange	California	6059	262241	4075

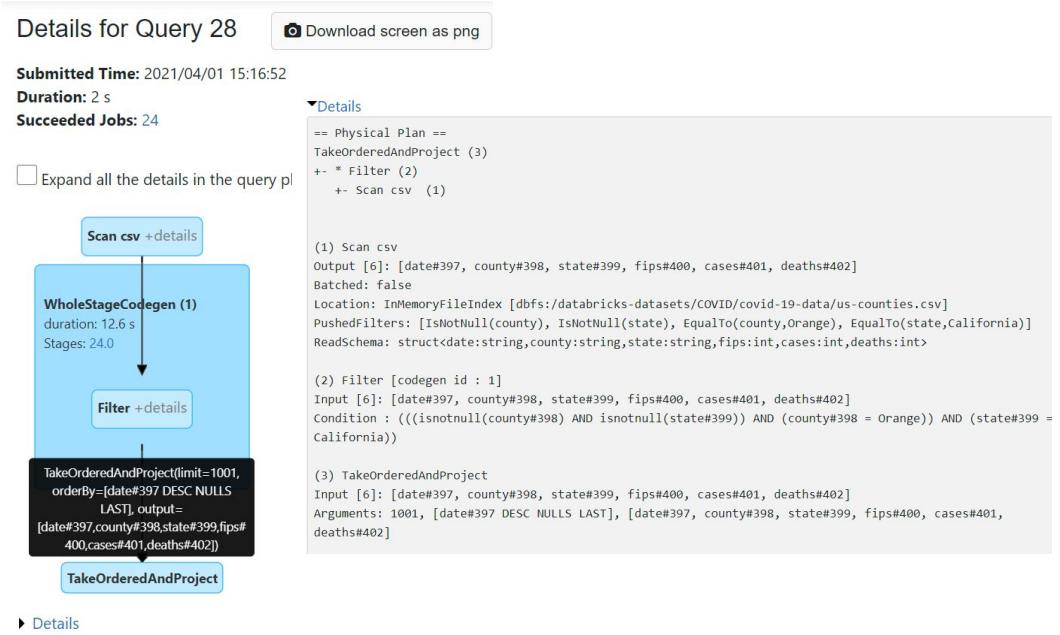
Showing all 412 rows.

Copyright © 2021 Antra, Inc. All rights reserved.

Automatic query optimization

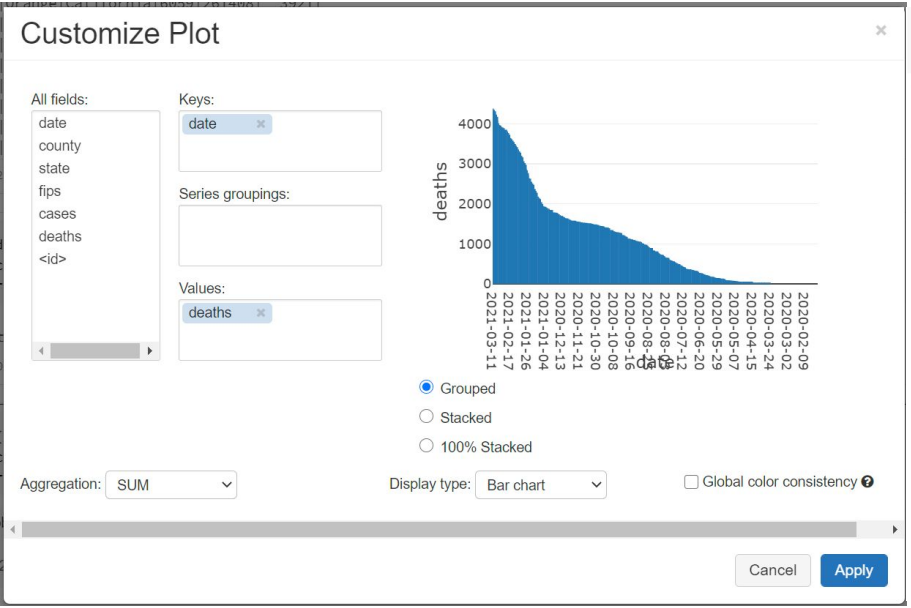
- Sort and filter has been swapped during optimization
-





Copyright © 2021 Antra, Inc. All rights reserved.

Antra



Copyright © 2021 Antra, Inc. All rights reserved.

Antra

Reading JSON Data - inferSchema

- Similar to CSV data but with a few differences
  - No header, so only one job needed even when inferring the schema
  - Column names are extracted from JSON object attributes

Cmd 19

```
1  display(
2    spark.read
3    .option("inferSchema","true")
4    .json("/databricks-datasets/learning-spark-v2/blogs.json")
5    .printSchema()
6  )
```

▸ (1) Spark Jobs

```
root
 |-- Campaigns: array (nullable = true)
 |    |-- element: string (containsNull = true)
 |-- First: string (nullable = true)
 |-- Hits: long (nullable = true)
 |-- Id: long (nullable = true)
 |-- Last: string (nullable = true)
 |-- Published: string (nullable = true)
 |-- Url: string (nullable = true)
```

Copyright © 2021 Antra, Inc. All rights reserved.

Reading JSON Data - Predefined Schema

- Manually defining the schema can be a lot of work, might not be worth it for small files.
- For large files, this might save a lot of time spent on the infer-schema process.

Cmd 20

```
1  from pyspark.sql.types import *
2  jsonSchema= StructType([
3    StructField("Campaigns",ArrayType(StringType()),True),
4    StructField("First",StringType(),True),
5    StructField("Hits",LongType(),True),
6    StructField("Id",LongType(),True),
7    StructField("Last",StringType(),True),
8    StructField("Published",StringType(),True),
9    StructField("Url",StringType(),True)
10 ])
11
12 (spark.read
13   .schema(jsonSchema)
14   .json("/databricks-datasets/learning-spark-v2/blogs.json")
15   .printSchema()
16 )
```

```
root
 |-- Campaigns: array (nullable = true)
 |    |-- element: string (containsNull = true)
 |-- First: string (nullable = true)
 |-- Hits: long (nullable = true)
 |-- Id: long (nullable = true)
 |-- Last: string (nullable = true)
 |-- Published: string (nullable = true)
 |-- Url: string (nullable = true)
```

Copyright © 2021 Antra, Inc. All rights reserved.

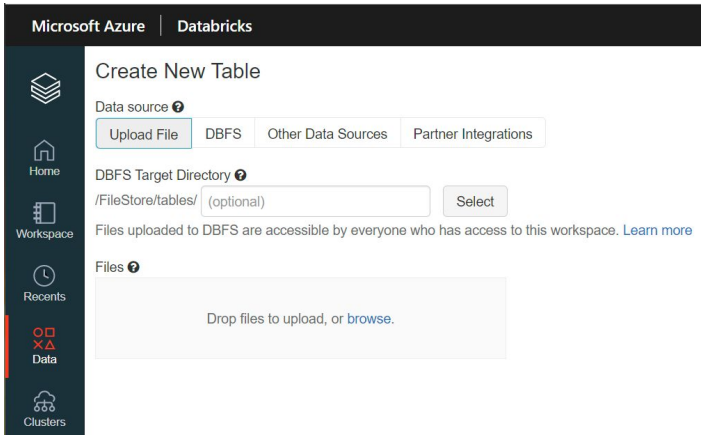
Reading Parquet Data

- Parquet is a column oriented storage format designed for big data
  - Optimized for reading and computing on columns
  - Metadata contains the schema
  - Compress better than row oriented storage
  - Easily splittable into multiple files
- Providing the schema when reading parquet data is not recommended
  - Reading in the schema from parquet metadata files is very cheap
  - Runtime exception if user provided schema is different from the schema stored in metadata.
- When reading, we provide the path to the directory that contains the parquet files instead of the path to individual files.

Copyright © 2021 Antra, Inc. All rights reserved.

Reading Data using the UI

- We can use the “Data” tab in the UI to load “tables”
  - Once we create the table, it will stay in the data tab and we’ll never have to do it again
  - Available for any users with the right permission
  - Users will not see the credentials used to load the table

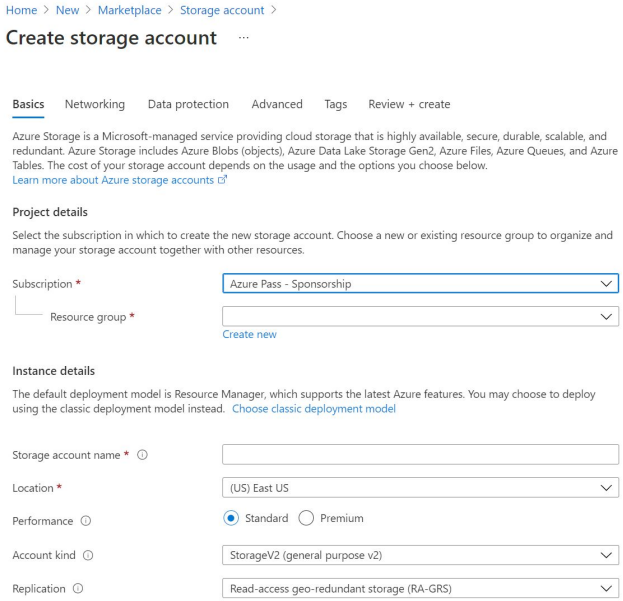


Copyright © 2021 Antra, Inc. All rights reserved.

Antra

Demo-Upload file to azure blob and access from databricks

- Start with creating a storage account
- Select general purpose v2 and enable ADLS Gen2

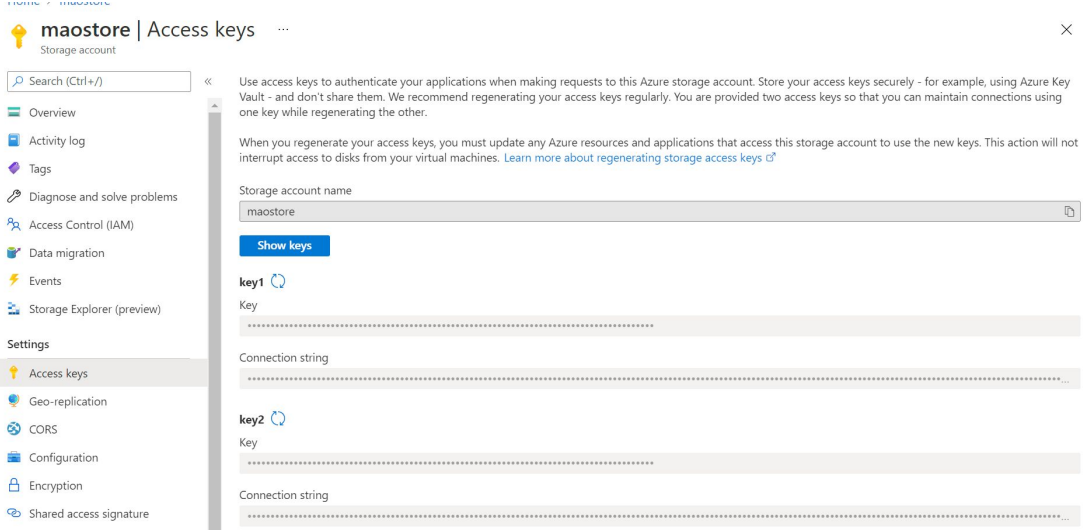


Copyright © 2021 Antra, Inc. All rights reserved.

Antra

Demo

- Go to “Access keys” to grab a connection string and store as a environmental variable.



Copyright © 2021 Antra, Inc. All rights reserved.

Antra

Demo

- Run **pipenv install azure-storage-blob** in terminal
- Create a blob service client
- Either create a container or get a existing container client
- Get a blob client
- ...

Copyright © 2021 Antra, Inc. All rights reserved.

Creating Tables from DataFrames

- Creating tables will allow us to query using Spark SQL
  - **df.createOrReplaceTempView("table_name")** can be used to create a temp table available only in the current notebook.
  - **df.createOrReplaceGlobalTempView("table_name")** can be used to create temp tables available in other notebooks

```
parquetDF.createOrReplaceGlobalTempView("parquet_table")
```

Cmd 22

```
%sql
select * from parquet_table order by requests desc limit(5)
```

▸ (1) Spark Jobs

	project	article	requests	bytes_served
1	en	Main_Page	865692	0
2	en.m	Main_Page	176949	0
3	en	Special:Search	76231	0
4	en.m	Donald_Trump	59847	0
5	en	Midas	55210	0

Copyright © 2021 Antra, Inc. All rights reserved.

Write Data in Azure Databricks

Copyright © 2015 Antra, Inc. All rights reserved.

Antra

- Use **.write** on a DataFrame to write data to the DBFS

```
1  fileName = userhome + "/pageviews_by_second.parquet"
2  print("Output location: " + fileName)
3
4  (csvDF.write                               # Our DataFrameWriter
5    .option("compression", "snappy") # One of none, snappy, gzip, and lzo
6    .mode("overwrite")                       # Replace existing files
7    .parquet(fileName)                       # Write DataFrame to Parquet files
8  )
```

Copyright © 2021 Antra, Inc. All rights reserved.