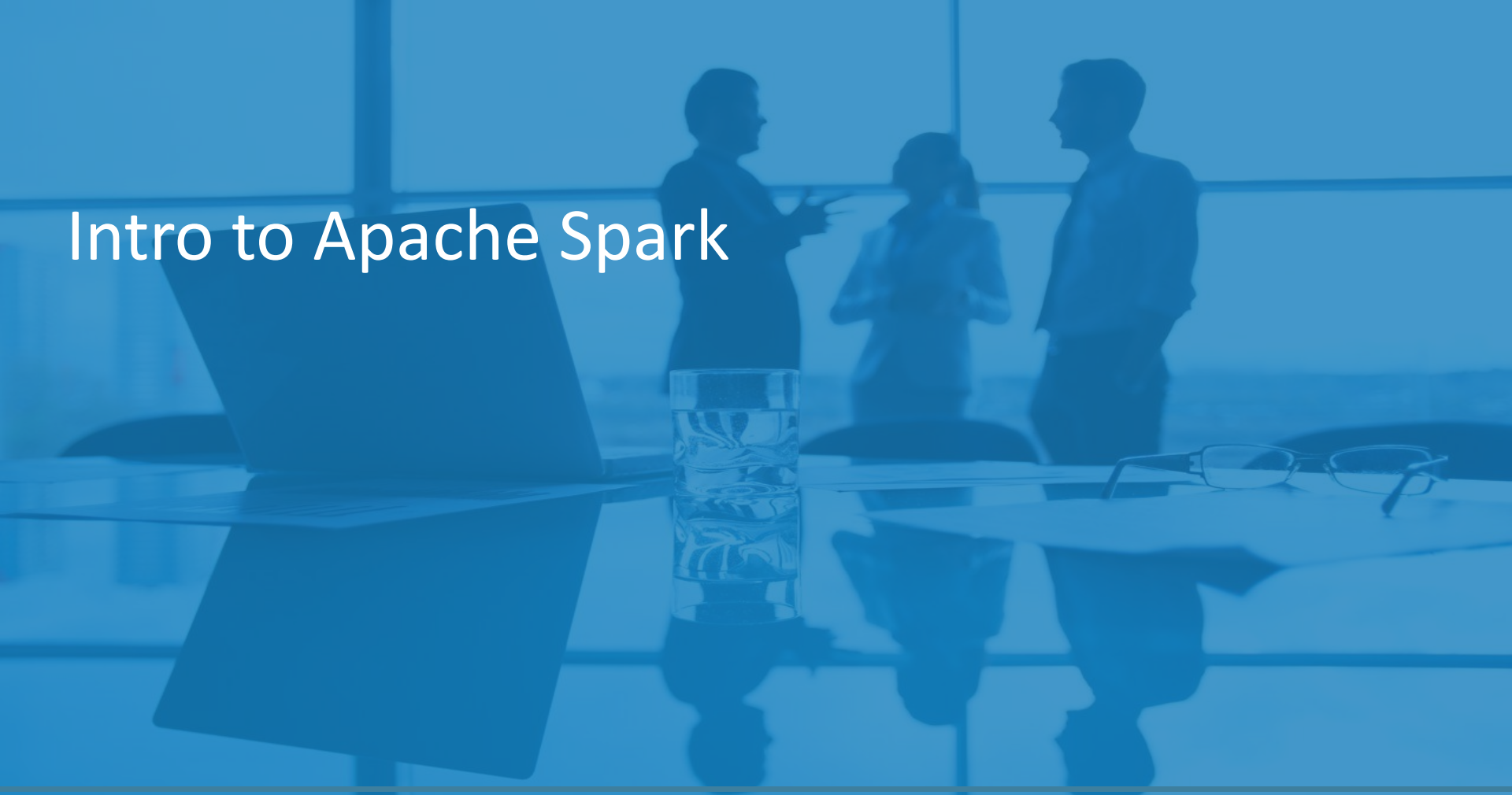
Antra

People · Process · Performance

Agenda

- Intro to Apache Spark
- Intro to Databricks

Intro to Apache Spark



Antra

What Is Apache Spark?

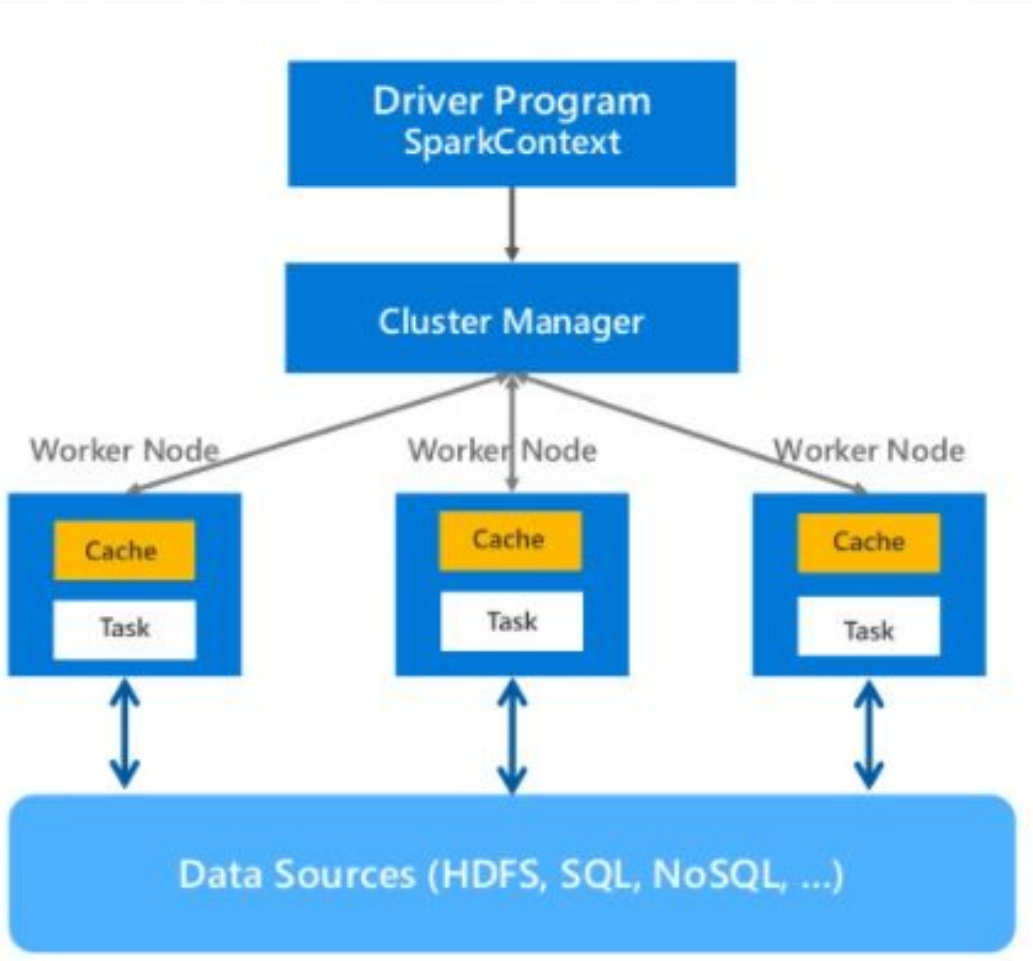
- Unified **processing engine** that can do
  - Big Data Processing
    - SQL + Structured Streaming + Graph
  - Machine Learning
- Fast, easy to use, sophisticated analytics
- In memory engine that's up to 100 times faster than Hadoop
- Largest open-source data project with 1000+ contributors
- Highly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

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.

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)



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.

Spark SQL, DataFrames and Datasets

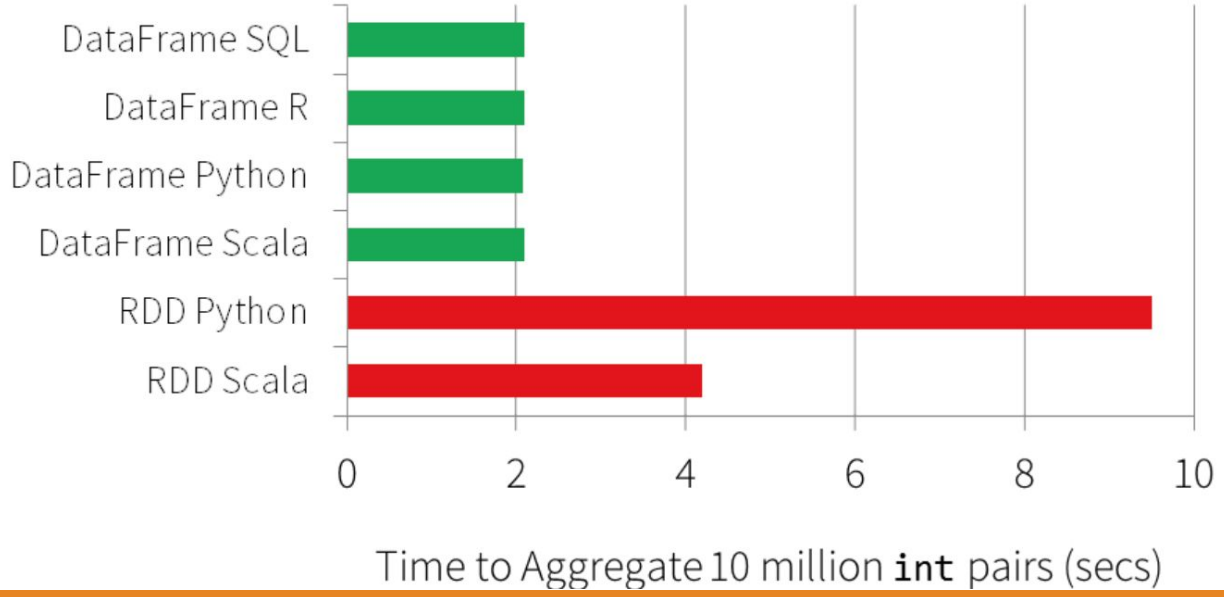
Antra

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

RDD vs DataFrame performance

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



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

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

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/R 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.

Challenges with Spark

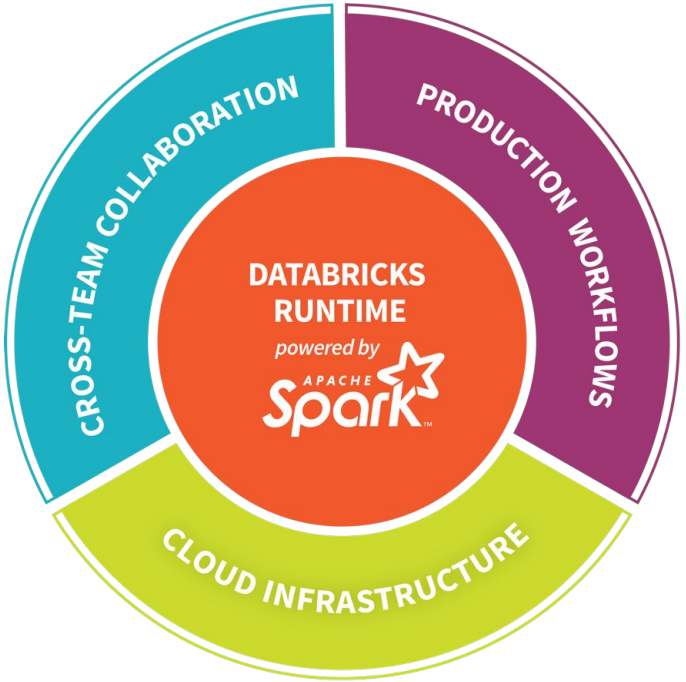
- Requires a distributed storage system
- Usually needs other technologies to be present (e.g. Hadoop(HDFS), Mesos)
- Security model is limited
- Performance tuning is tricky
- Capacity management and cost optimization are hard

Intro to Azure/AWS/GCP Databricks

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



Antra

What is Azure Databricks?

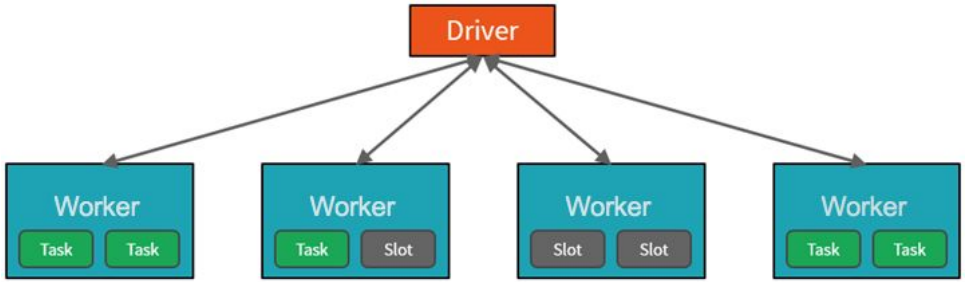
- Managed service on cloud
- Data analytics platform based on Apache Spark that's optimized for Azure
  - **Azure Databricks SQL Analytics**
    - Easy-to-use platform for
      - Running SQL queries on data lakes
      - Creating multiple visualization types to explore query results
      - Building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

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

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

Antra

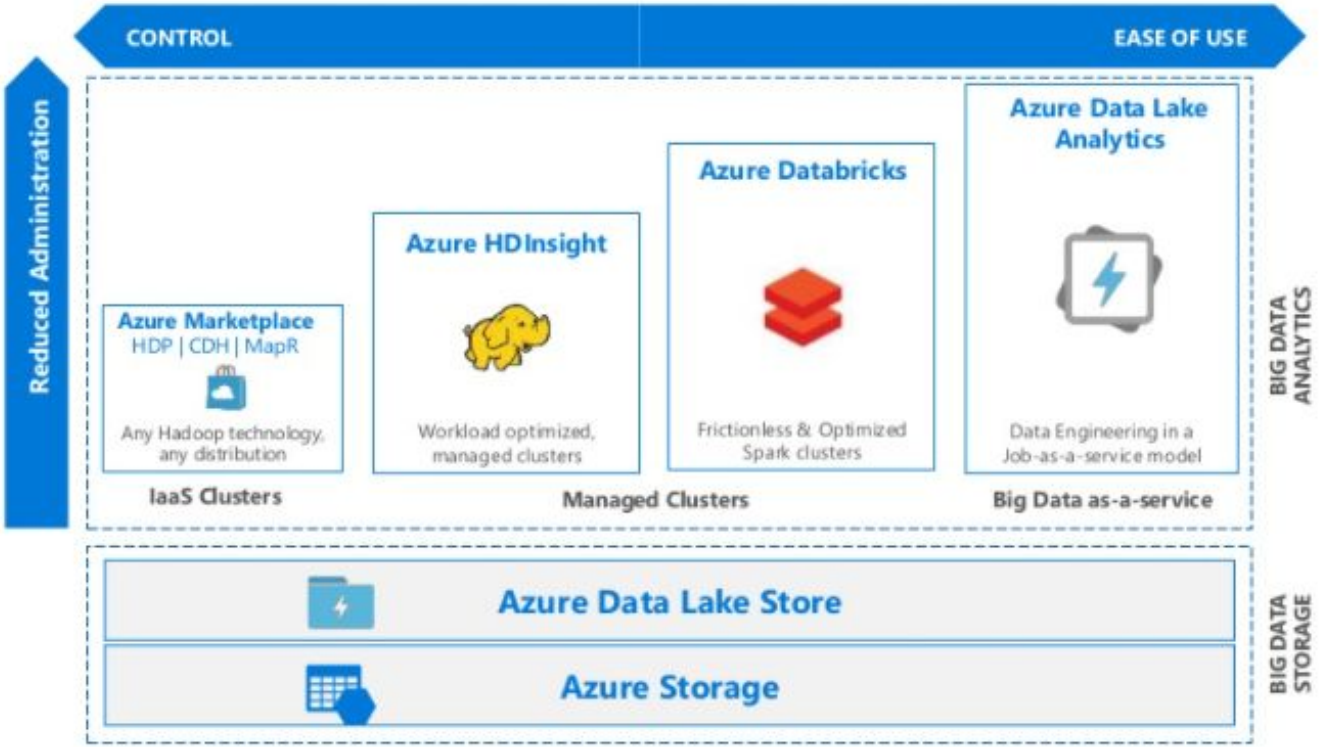
Databricks High-level Architecture

- 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.

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.
  - df.sort(...).filter(...)
    - Job1 -> filter the data
      - Tasks -> filter certain partition of data
    - Job2 -> sort the data
      - Tasks -> sort certain partition of data

Other Azure Big Data Solutions



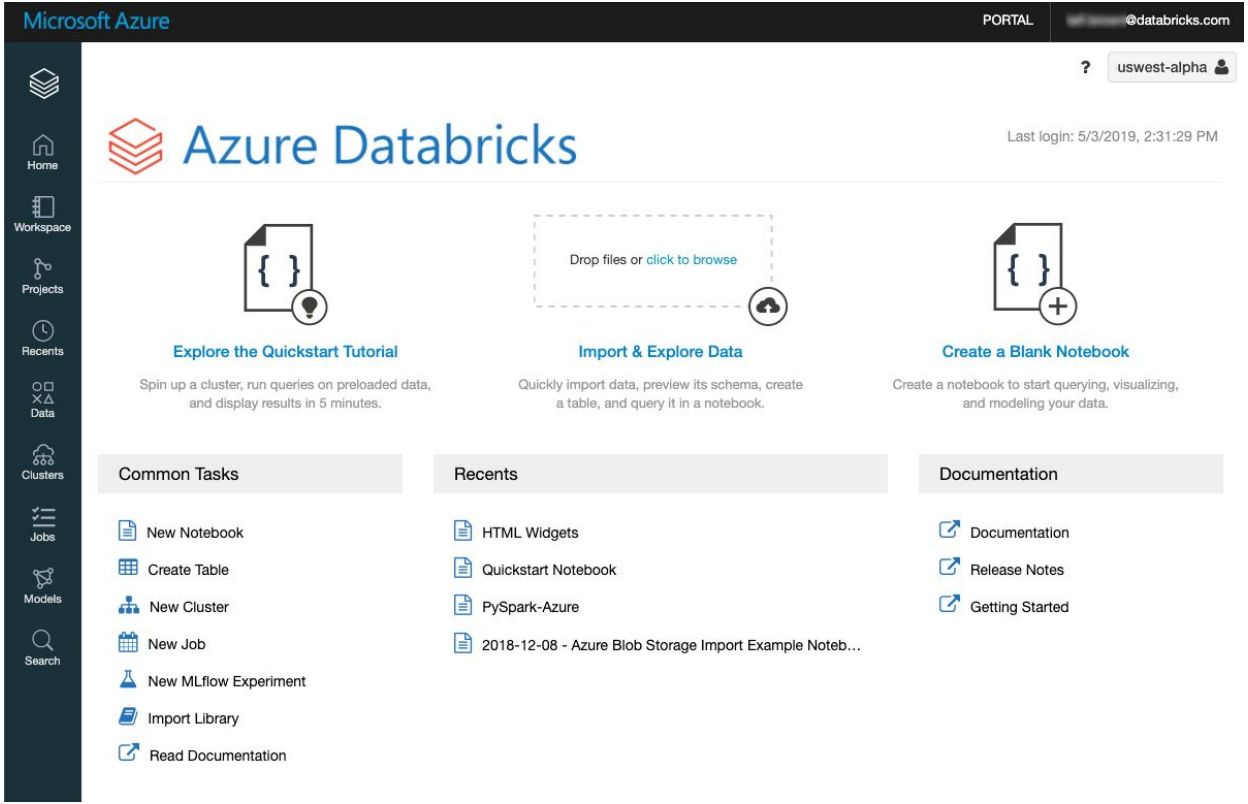
Antra

Get started with Azure Databricks

Antra

Get started with Azure Databricks

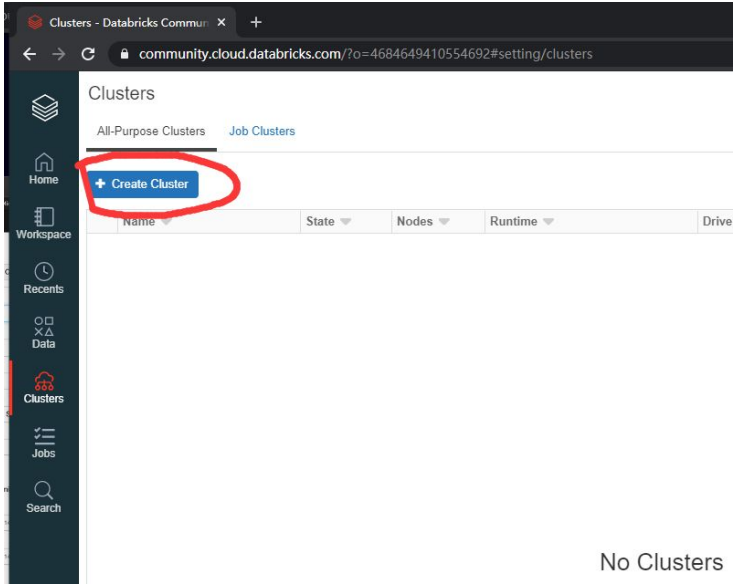
- Provision directly from the Azure Portal like any other Azure Services
- Any Azure user with the appropriate subscription and authorization can provision Azure Databricks service

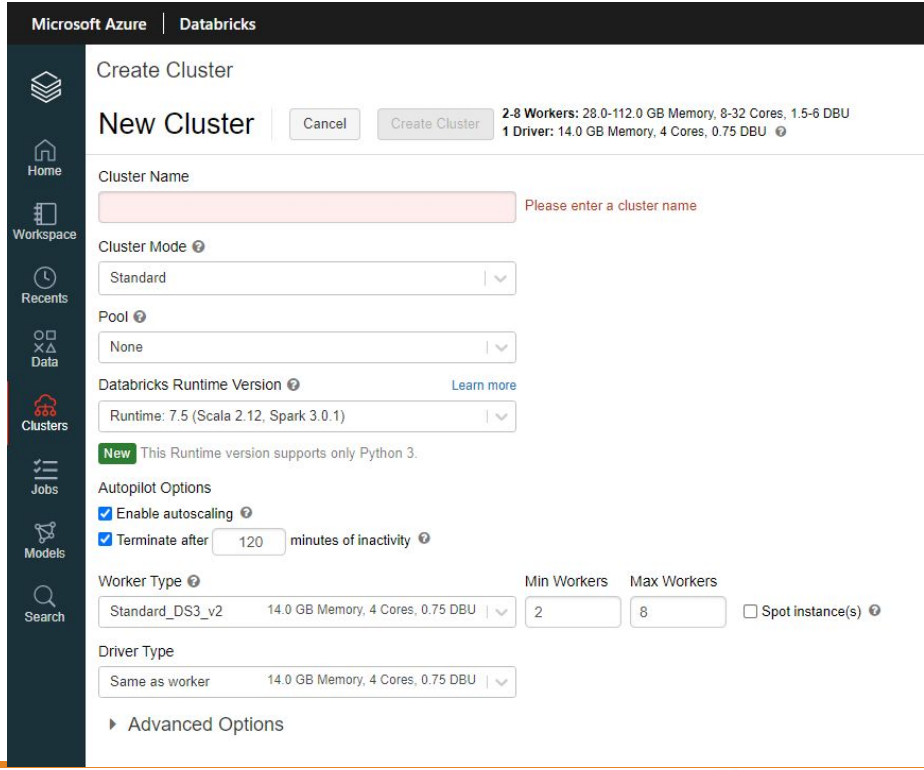


Antra

Clusters

- Where all the computation happens





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.

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.

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…

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

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

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"])
```

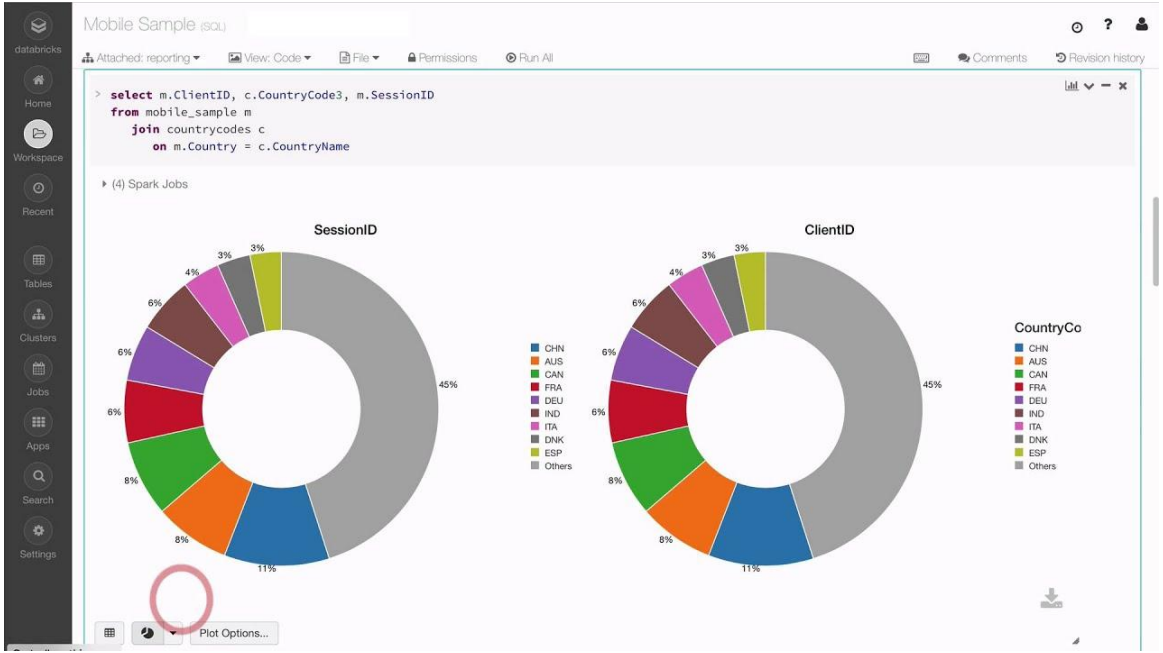
Notebook job #10931
Notebook job #10932
Notebook job #10933

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

Command took 13.81s

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



Antra

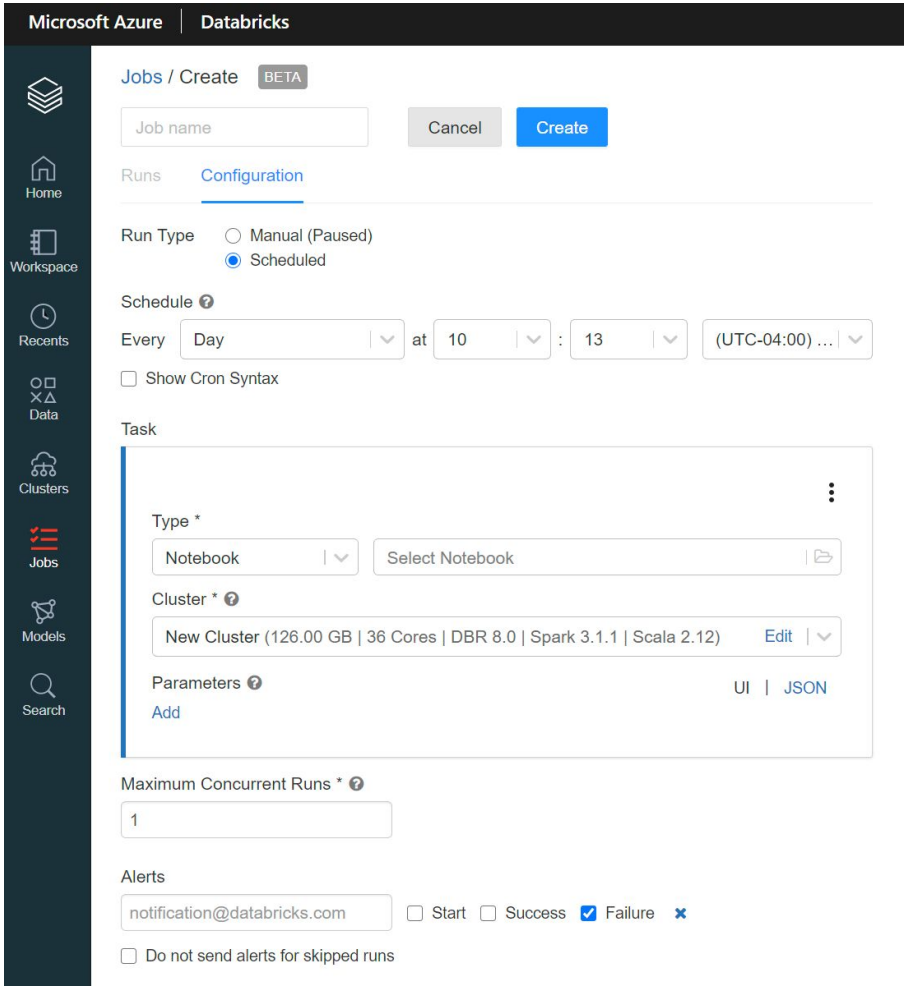
Jobs

- 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

Antra

Jobs

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



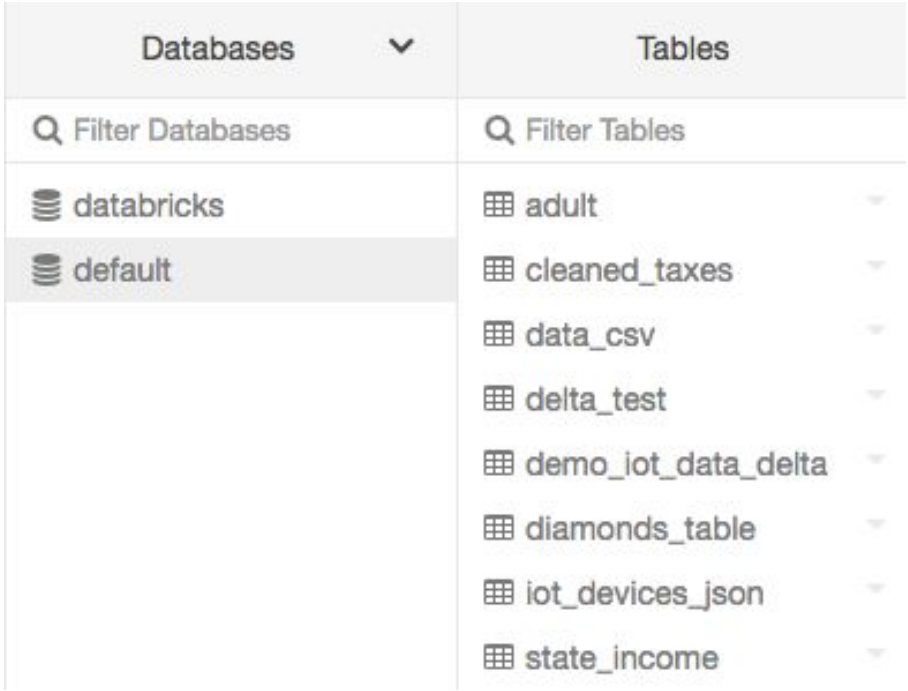
Antra

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

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



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]

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
--------------------------------------------------------------------------------
```

dbutils.fs.ls()

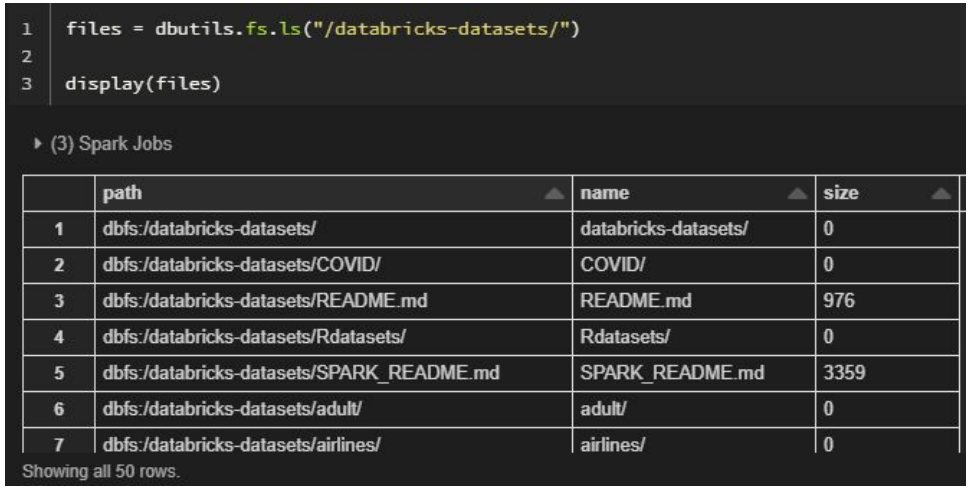
- We can use **dbutils.fs.ls** to get the **FileInfo** objects of the directories we have
  - **FileInfo** contains
    - path -> path of the file or directory
    - is_dir -> whether the path points to a directory
    - file_size -> length of the file in bytes or zero if the path is a dir
    - modification_time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

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.

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

Antra

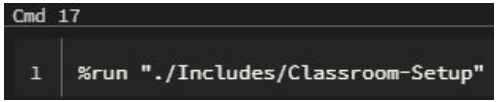
Magic Commands

- 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	...

Magic Commands

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

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

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)

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)