Azure Databricks

April 2022 Camilla Gaardsted

SuperUsers A/S

Databricks – A modern Lakehouse

aws

Google Cloud

Microsoft Azure

Databricks – Traditional DW vs Lakehouse

	Data Warehouse	Lakehouse
Data formats	Closed	Open
Data types	Structured*	Any type of data
Scalability	Limited**	Highly scalable
Cost	$$$	$
Use cases	BI, SQL	BI, SQL, ML, Real-Time Apps
Data access	SQL only	Open APIs for direct access to files with SQL, R, Python and other languages
Reliability	High-quality, reliable data with ACID transactions	High-quality, reliable data with ACID transactions
Governance	Fine-grained security and governance for row/columnar level for tables	Fine-grained security and governance for row/columnar level for tables
Performance	High	High

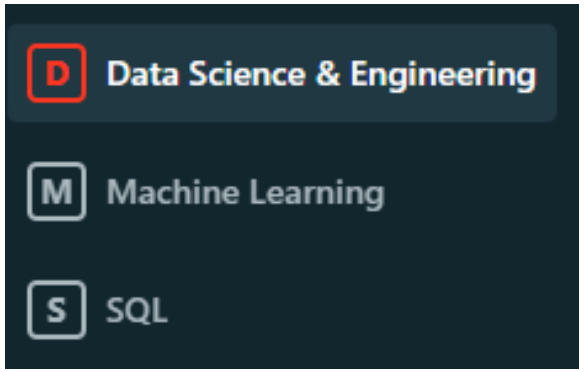
Kilde: https://databricks.com/discoverlakehouse

Azure Databricks

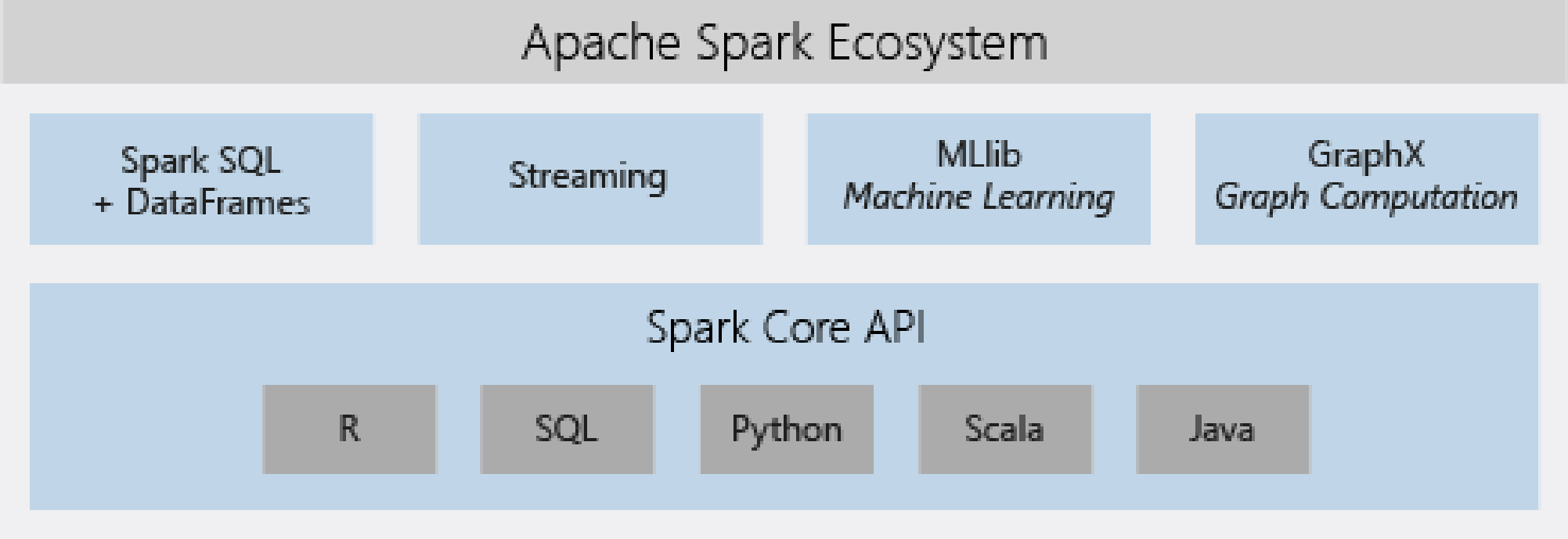
Azure Databricks is a data analytics platform optimized for the Microsoft Azure cloud services platform.

There are 3 environments now with own GUI:

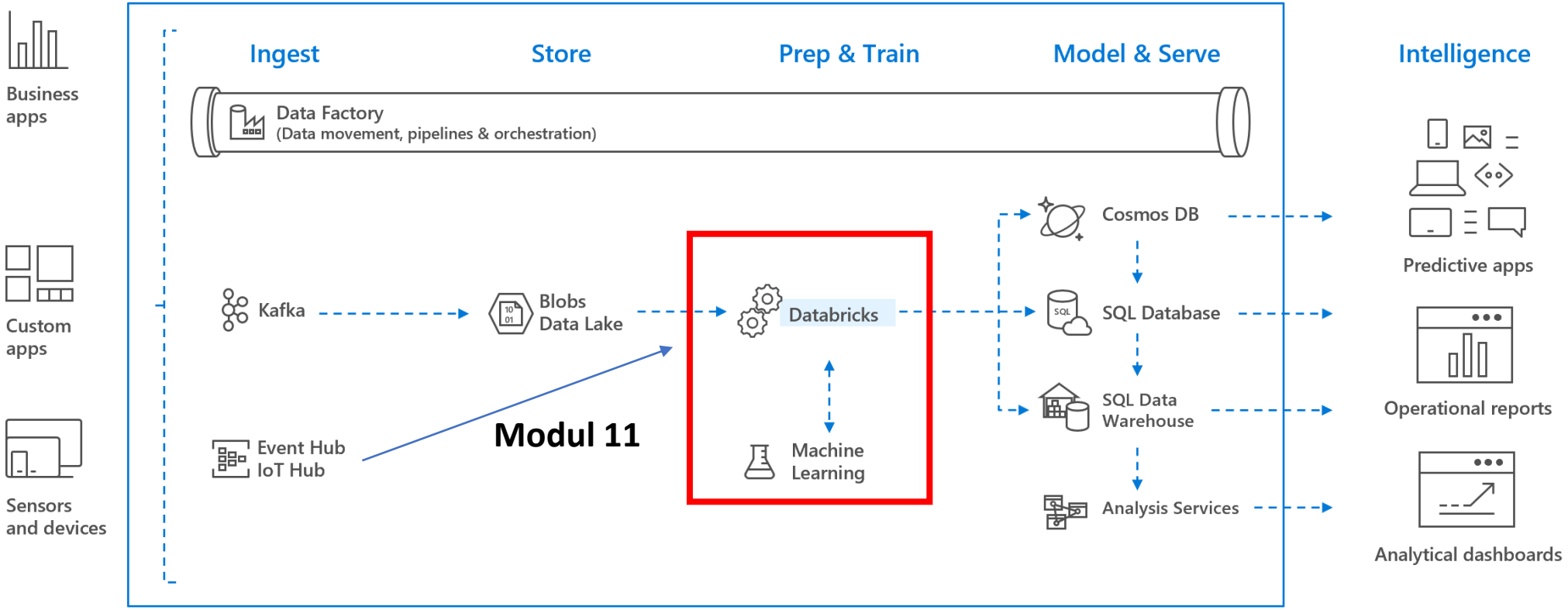
- Databricks SQL
- **Databricks Data Science & Engineering**
- Databricks Machine Learning



Azure Databricks components



Azure Data Bricks



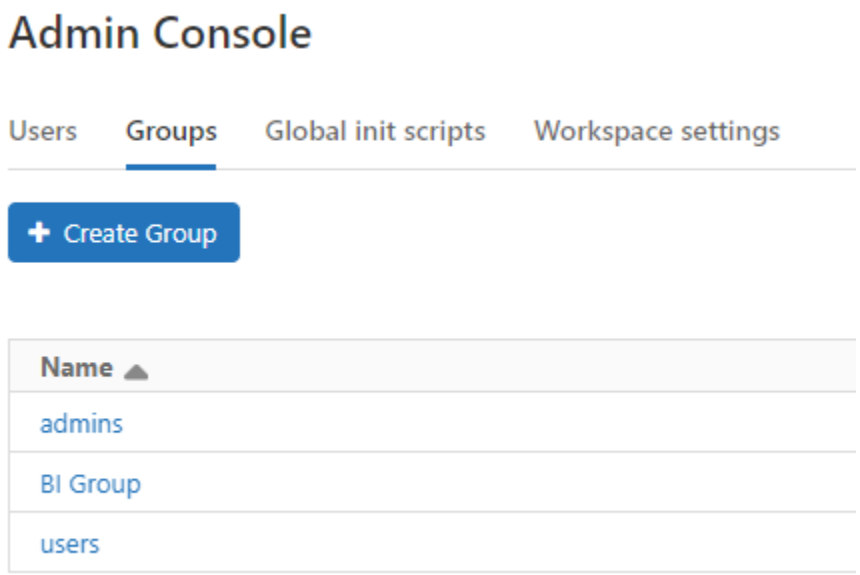
Azure Databricks - Edition

Standard	Premium
Interactive workloads notebooks (collaboratively)	Includes standard features
Automated workloads to run jobs via API or UI	**Role-based access for: notebooks, clusters, jobs, and tables**
Apache Spark on Databricks platform	JDBC/ODBC Endpoint Authentication
Job scheduling with libraries	Audit logs
Job scheduling with Notebooks	Azure AD credential passthrough
Autopilot clusters	Conditional Authentication
Databricks Runtime for ML	Cluster Policies (preview)
MLflow on Databricks (Preview)	IP Access List (preview)
Databricks Delta	Token Management API (preview)
Interactive clusters	Delta Live Tables (DLT)
Notebooks and collaboration	
Ecosystem integrations	

Azure Databricks – Security: admins (built-in)

admins (built-in Azure Databricks group for administrators)

Azure AD users/service principals who are Owner or Contributor on Azure Databricks resource are added to admins



Azure Databricks – Security: Users

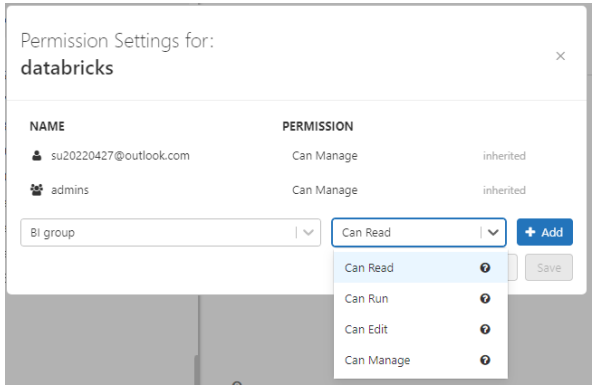
- Users are Azure AD principals

User account provisioning:

- Users/groups access and administration handled manually in Admin Console in GUI in Databricks workspace
- Azure AD SCIM offers synchronization of users and groups (public preview, premium only)

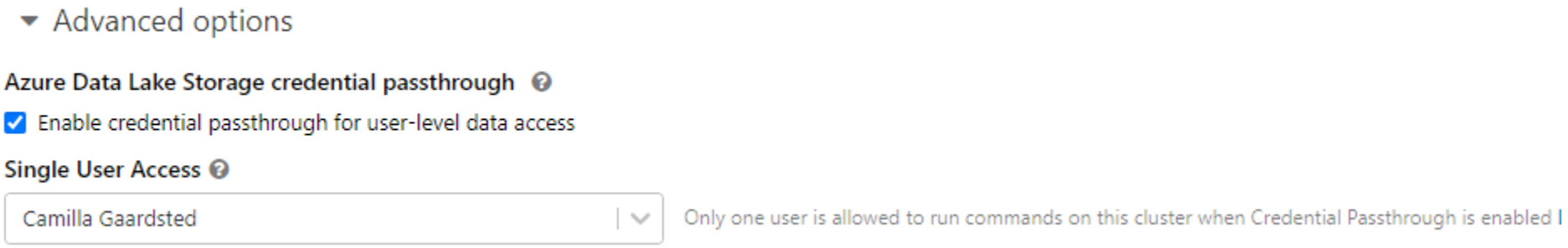
Permissions

Premium offers access control for notebooks, clusters, jobs and tables

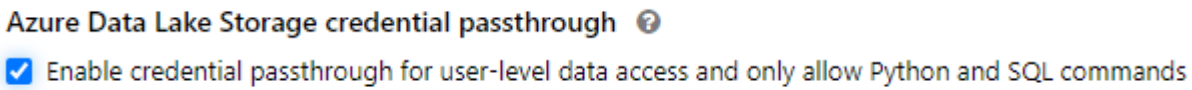


Azure Databricks – data lake credential passthrough

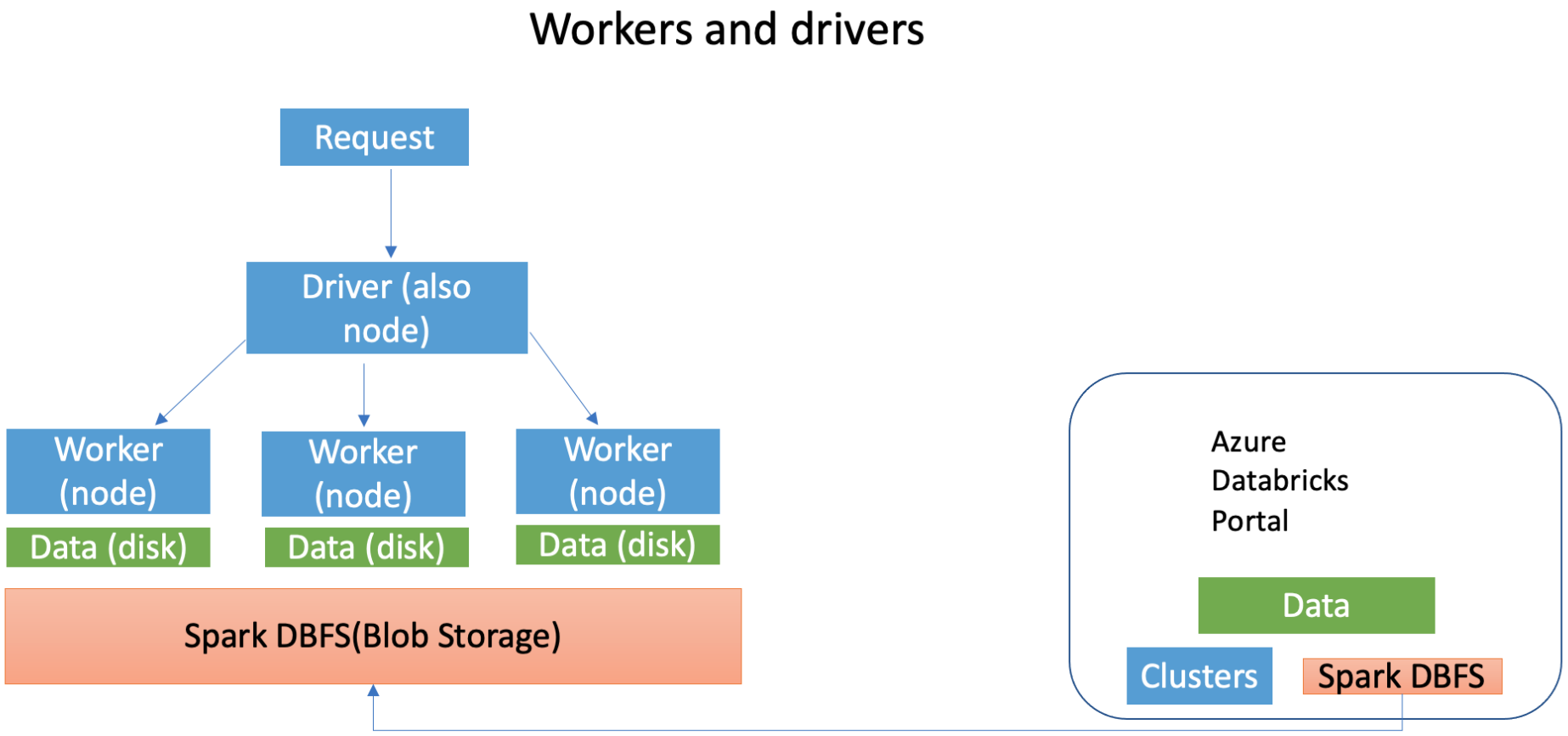
Standard cluster mode allows one single user



High concurrency mode allows multiple users



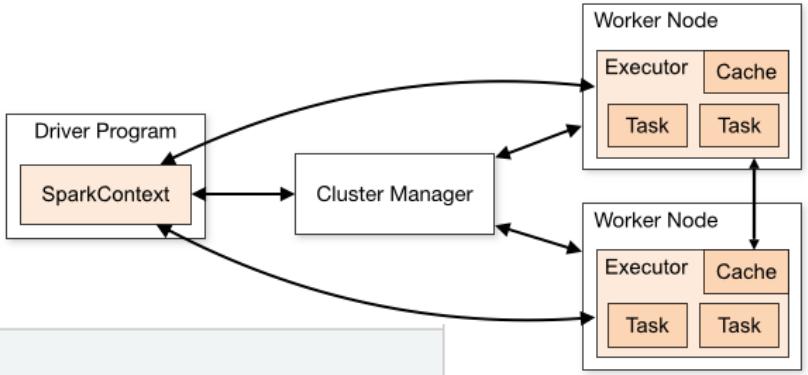
Azure Databriks - Cluster



Kilde: https://tomaztsql.files.wordpress.com/2020/12/screenshot-2020-12-05-at-09.50.41.png

Databricks – Driver, nodes and executors

Glossary



Worker node	Any node that can run application code in the cluster
Executor	A process launched for an application on a worker node, that runs tasks and keeps data in memory or disk storage across them. Each application has its own executors.
Task	A unit of work that will be sent to one executor
Job	A parallel computation consisting of multiple tasks that gets spawned in response to a Spark action (e.g. `save`, `collect`); you'll see this term used in the driver's logs.
Stage	Each job gets divided into smaller sets of tasks called *stages* that depend on each other (similar to the map and reduce stages in MapReduce); you'll see this term used in the driver's logs.

Kilde: https://spark.apache.org/docs/latest/cluster-overview.html

Azure Databricks - Cluster

Cluster mode is defined at creation – cannot be changed later

Mode	Languages	Table access control	Single user
Standard	SQL, Python, R, Scala	No	Yes
High concurrency	SQL, Python, R	Yes	No
Single node	SQL, Python, R, Scala	No	Yes

DBFS – Databricks Filesystem

- Dbutils – databricks utilities for fs etc

Enable DBFS browser in Admin Console (advanced settings)

- Synapse Analytics: MS has made MSSparkUtils (lab 1)

Databricks – Datasources

Directly supported in Databricks runtime or accesible via simple shell commands:

Avro file	Mlflow experiment
Binary file	Parquet file
CSV file	XML file
Image	ZIP file
JSON file	Hive table
LZO compressed file	Delta Lake

Databricks – Datasource connections

Datasources which require a connection to the storage:

Azure Blob storage	MongoDB
Azure Data Lake Storage Gen1	Neo4j
Azure Data Lake Storage Gen2	Redis
Azure Cosmos DB	
Azure Synapse Analytics	Snowflake
Cassandra	SQL Databases using JDBC
Couchbase	SQL Databases using the Apache Spark connector
ElasticSearch	

Databricks – Cluster: Job metrics

```
fileName = userhome + "/cg_pageviews_by_second.csv"
print("Output location: " + fileName)

(csvDF.write                        # Our DataFrameWriter
  .mode("overwrite")                # Replace existing files
  .csv(fileName)                # Write DataFrame to csv files
)
```

(1) Spark Jobs

Job 10 View (Stages: 1/1)

Stage 10: 8/8

```
Output location: dbfs:/user/su20220427@outlook.com/cg_pageviews_by_second.csv
```

Summary Metrics for 8 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	5 s	6 s	6 s	7 s	7 s
GC Time	0.4 s	0.4 s	0.5 s	0.5 s	0.5 s

Showing 1 to 2 of 2 entries

Aggregated Metrics by Executor

Show 20 entries Search:

Executor ID	Logs	Address	Task Time	Total Tasks	Failed Tasks	Killed Tasks	Succeeded Tasks	Excluded
0	stdout stderr	10.139.64.4:45711	27 s	4	0	0	4	false
1	stdout stderr	10.139.64.5:39007	23 s	4	0	0	4	false

Showing 1 to 2 of 2 entries Previous 1 Next

Tasks (8)

Show 20 entries Search:

Index	Task ID	Attempt	Status	Locality level	Executor ID	Host	Logs	Launch Time	Duration	GC Time	Errors
0	75	0	SUCCESS	PROCESS_LOCAL	0	10.139.64.4	stdout stderr	2022-04-27 11:02:38	7 s	0.5 s	
1	76	0	SUCCESS	PROCESS_LOCAL	1	10.139.64.5	stdout stderr	2022-04-27 11:02:38	6 s	0.4 s	
2	77	0	SUCCESS	PROCESS_LOCAL	0	10.139.64.4	stdout stderr	2022-04-27 11:02:38	7 s	0.5 s	
3	78	0	SUCCESS	PROCESS_LOCAL	1	10.139.64.5	stdout stderr	2022-04-27 11:02:38	6 s	0.4 s	
4	79	0	SUCCESS	PROCESS_LOCAL	0	10.139.64.4	stdout	2022-04-27	6 s	0.5 s	

Databricks – RDD and lazy evaluation

pyspark.sql.DataFrame.rdd

Resilient Distributed Datasets (RDD)

RDD.getNumPartitions()

RDD opeations: Transformations vs actions

Some code triggers a job and some does not in a notebook

Data is not fetched before needed (lazy evaluation)

All the transformations are not made before an action triggers it

Databricks – dataframe cache

- Cache

df.cache().count() – will cache the df partitions on the nodes

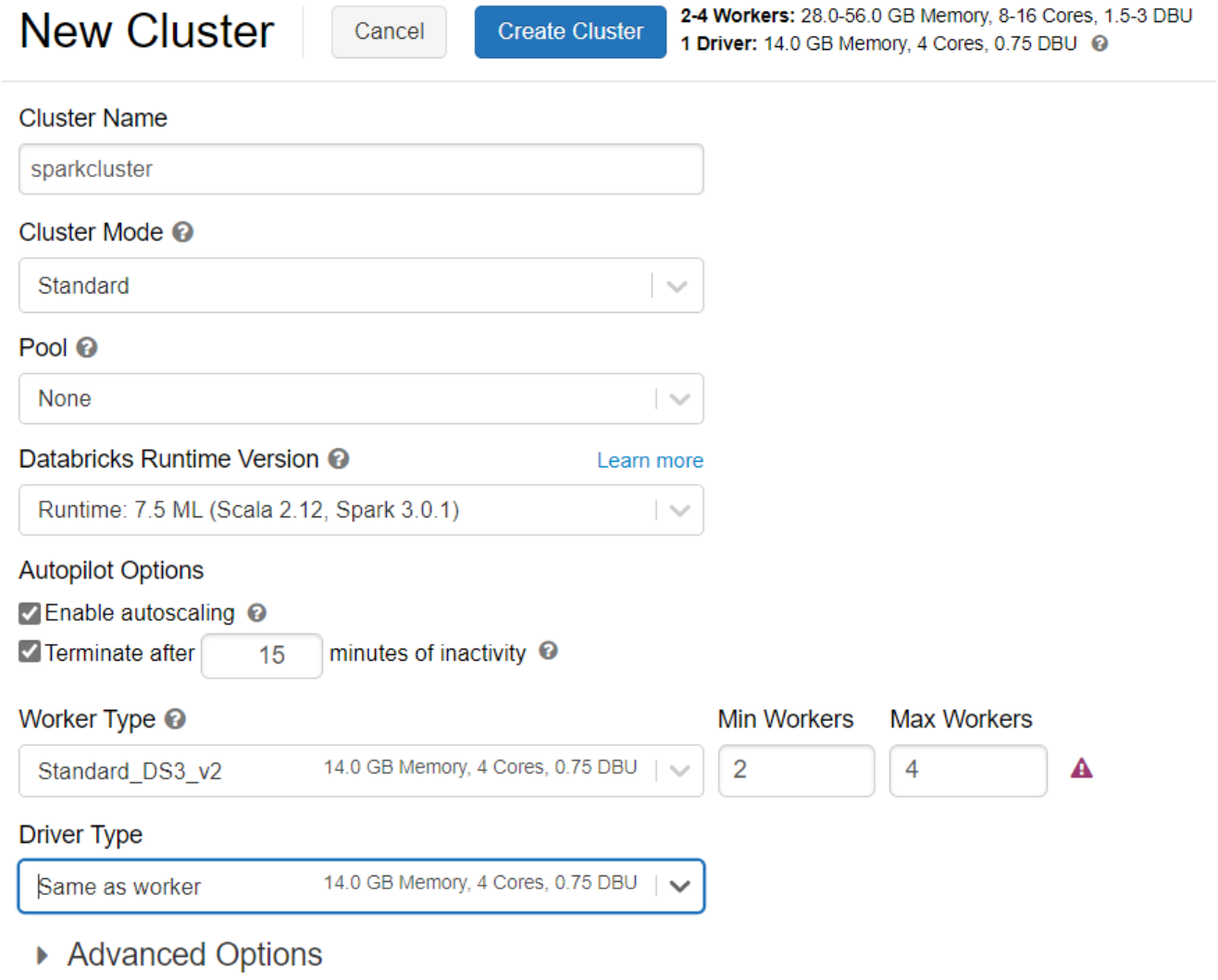
Working with Select in Azure Databricks

SQL	DataFrame (Python)
SELECT col_1 FROM myTable	df.select(col("col_1"))
DESCRIBE myTable	df.printSchema()
SELECT * FROM myTable WHERE col_1 > 0	df.filter(col("col_1") > 0)
..GROUP BY col_2	..groupBy(col("col_2"))
..ORDER BY col_2	..orderBy(col("col_2"))
..WHERE year(col_3) > 1990	..filter(year(col("col_3")) > 1990)
SELECT * FROM myTable LIMIT 10	df.limit(10)
display(myTable) (text format)	df.show()
display(myTable) (html format)	display(df)

Working with transformations in Azure Databricks

Transformations	Description
Select(…)	The select(...) command enables you to specify the columns to include in a query
drop(…)	The drop(...) command enables you to specify the columns you don’t want
distinct(…)	The distinct(…) command returns a distinct set of values in a DataFrame
dropDupicates(…)	The dropDupicates(…) command is an alias of the distinct(…) command.
show(…)	The show(..) command is part of the core Spark API and simply prints the results to the console
display(…)	The display(…) command provides more flexibility than show(…) such as downloading results against csv, rendering charts and showing up to 100 rows
limit(…)	The limit(…) command can be used to control the number of records that are returned to a DataFrame

Azure Databricks



Databricks - Cluster

- You run these workloads as a set of commands in a notebook or as an automated job. Databricks makes a distinction between *all-purpose clusters* (interactive) and *job clusters (automated clusters)*.
- You use all-purpose clusters to analyze data collaboratively using interactive notebooks. You use job clusters to run fast and robust automated jobs.

Notebooks

A notebook is a document and contains

- Computer code (e.g. python)
- Rich text elements (paragraph, equations, links, etc…)
- data resultsets/visualizations

Jupyter Notebooks were originally used for ipython (interactive python)

- Run and edit notebooks in a web browser/text editor
- Document is a json file with ipynb file extension

Now Widely used for python, sql, powershell etc

Databricks - Notebooks

A notebook is a web-based interface to a document that contains runnable code, visualizations, and narrative text

Databricks supports code written in

- Python
- R
- Scala
- SQL

Notebook external formats

A source file with the extension .scala, .py, .sql, or .r.

HTML

- A Databricks notebook with an .html extension.

DBC Archive

- A Databricks archive.

IPython Notebook

- A Jupyter notebook with the extension .ipynb.

R

- An R Markdown document with the extension .Rmd.

Databricks - Notebook

Default sprog her er SQL

Cluster

Textblok

Kodeblok

Kodeoutput

Microsoft Azure | Databricks

Quickstart Notebook (SQL)

mlclustercg | File | Edit | View: Standard | Permissions | Run All | Clear

Home
Workspace
Recents
Data
Clusters
Jobs
Models
Search

The next command creates a table from a Databricks dataset

Cmd 5

```
DROP TABLE IF EXISTS diamonds;

CREATE TABLE diamonds
USING csv
OPTIONS (path "/databricks-datasets/Rdatasets/data-001/csv/ggplot2/diamonds.csv", header "true")
```

OK

Command took 1.14 seconds -- by a user at 19.10.2020 21.34.10 on unknown cluster

Cmd 6

```
SELECT * from diamonds
```

	_c0	carat	cut	color	clarity	depth	table	price	x	y	z
1	1	0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
2	2	0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
3	3	0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
4	4	0.29	Premium	I	VS2	62.4	58	334	4.2	4.23	2.63

Showing the first 1000 rows.

Command took 0.45 seconds -- by a user at 19.10.2020 21.34.10 on unknown cluster

Databricks - Notebooks

Skillpipe and Labs references notebooks here:

https://github.com/solliancenet/microsoft-learning-paths-databricks-notebooks/blob/master/data-engineering/DBC

Databricks File System (DBFS)

Databricks File System (DBFS) is a distributed file system mounted into an Azure Databricks workspace and available on Azure Databricks clusters.

- Allows you to mount storage objects so that you can seamlessly access data without requiring credentials.
- Allows you to interact with object storage using directory and file semantics instead of storage URLs.
- Persists files to object storage, so you won’t lose data after you terminate a cluster.

Databricks – Azure Blob storage

Directly access in a notebook session via storage account acces key or SAS:

```
%python
spark.conf.set(
  "fs.azure.account.key.<storage-account-name>.blob.core.windows.net",
  "<storage-account-access-key>")

dbutils.fs.ls("wasbs://<container-name>@<storage-account-name>.blob.core.windows.net/<directory-name>")
```

Or via Spark Configuration property for all cluster users

For testing purposes only! Credentials are visible for all!

DBFS –Mount Azure Blob storage

- Mount a Blob storage container or a folder inside a container (block blobs only)
- Security
  - Public storage account
  - Shared key
  - Shared Access Signature (SAS)

Databricks - Mount

- <mount-name> becomes the DBFS path
- <conf-key> either
  - fs.azure.account.key.<storage-account-name>.blob.core.windows.net
  - fs.azure.sas.<container-name>.<storage-account-name>.blob.core.windows.net

```
dbutils.fs.mount(
  source = "wasbs://<container-name>@<storage-account-name>.blob.core.windows.net",
  mount_point = "/mnt/<mount-name>",
  extra_configs = {"<conf-key>":dbutils.secrets.get(scope = "<scope-name>", key = "<key-name>")})
```

Databricks – Secret scopes

Two types of secret scops:

- Azure Key Vault-backed scopes
- Databricks-backed scopes

```
camilla@Azure:/usr/bin$ databricks secrets list-scopes
Scope          Backend          KeyVault URL
-----------    -------------    ----------------------------------------
azuresecrets   AZURE_KEYVAULT   https://westuskeyvault2021.vault.azure.net/
bricksscope    DATABRICKS       N/A
```

Databricks - Azure Key Vault backed scopes

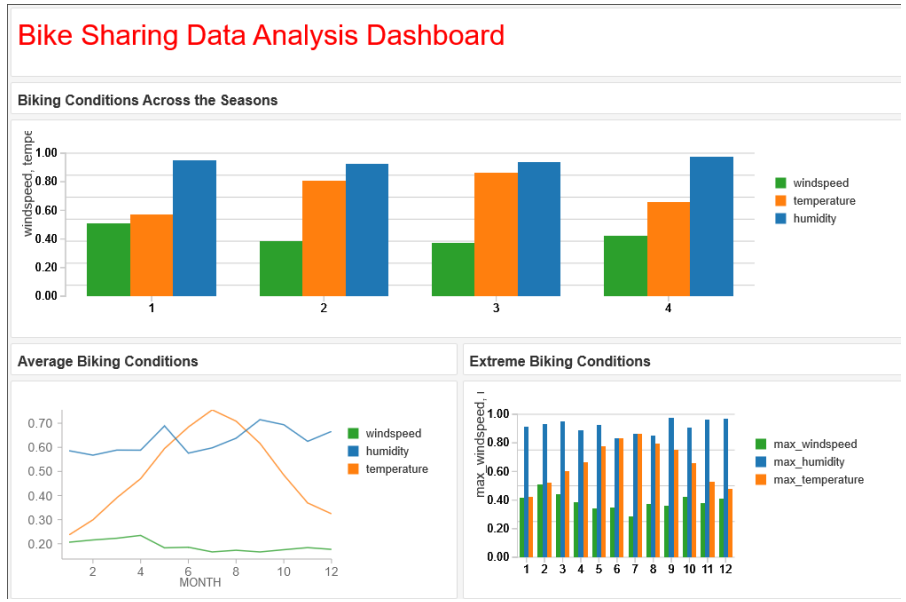
Anvender credentials som hentes fra en Azure Key Vault

Opret secret scope via url/databricks CLI

- Angiv Azure Key vault URI and ResourceID

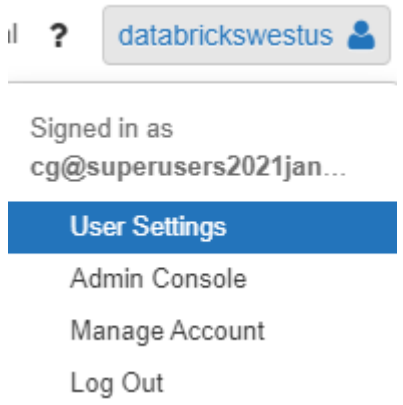
Databricks - Dashboard

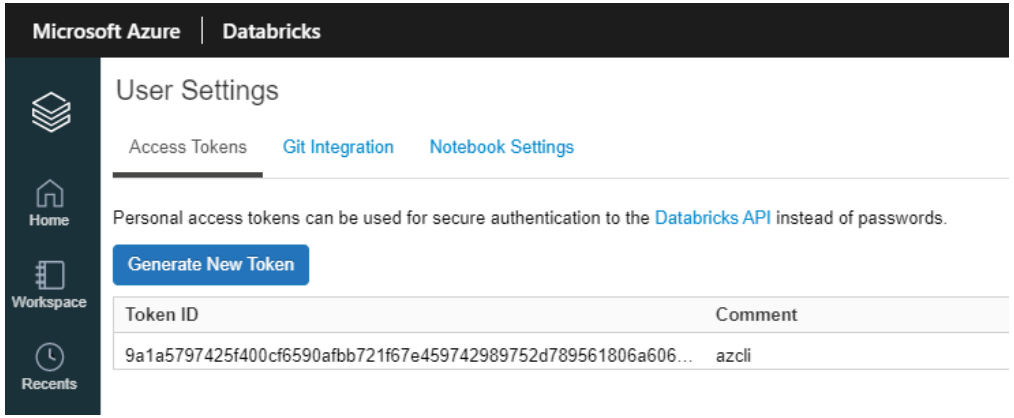
Dashboard allow you to publish graphs and visualizations derived from notebook output and share them in a presentation format with your organization



Databricks - CLI

- Configure with url and token





Databricks - Jobs

A job is a way of running a notebook or JAR either immediately or on a scheduled basis. The other way to run a notebook is interactively in the notebook UI

Create and run jobs using the UI, the CLI, and by invoking the Jobs API. You can monitor job run results in the UI, using the CLI, by querying the API, and through email alerts

Databricks – Delta lake

Delta Lake is an open format storage layer that delivers reliability, security and performance on your data lake — for both streaming and batch operations

ACID

Time travel

Databricks - dokumentation

https://docs.microsoft.com/en-us/azure/databricks

https://docs.databricks.com

https://spark.apache.org/docs/latest

Databricks - DBFS

Browse enable under advanced settigns in admin console

Spark - Dataset/DataFrame

A Dataset is a distributed collection of data (Spark 1.6+ replaced RDD)

The Dataset API is available in Scala and Java

Python does not have the support for the Dataset API

A DataFrame is a *Dataset* organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R/Python, but with richer optimizations under the hood

Spark - Databases and tables

An Azure Databricks database is a collection of tables you query with

SPARK API
SPARK SQL

Global tables are persist and available across all clusters (Hive compatibility)

A local table is just a temporary view in a notebook

SPARK SQL

CREATE TABLE gives a table stored in files

REFRESH <tablename> will update the table data from the files

INSERTs are allowed

UPDATE and DELETE are only supported on tables that support ACID (transaction log) e.g. DELTA tables

DELTA format is default for tables from Databricks Runtime 8.0

Spark SQL - Tables

Global table

Local table

Managed table

Unmanaged table (external table)

Spark – Structured Streaming

Structured Streaming is a scalable and fault-tolerant stream processing engine built on the Spark SQL engine

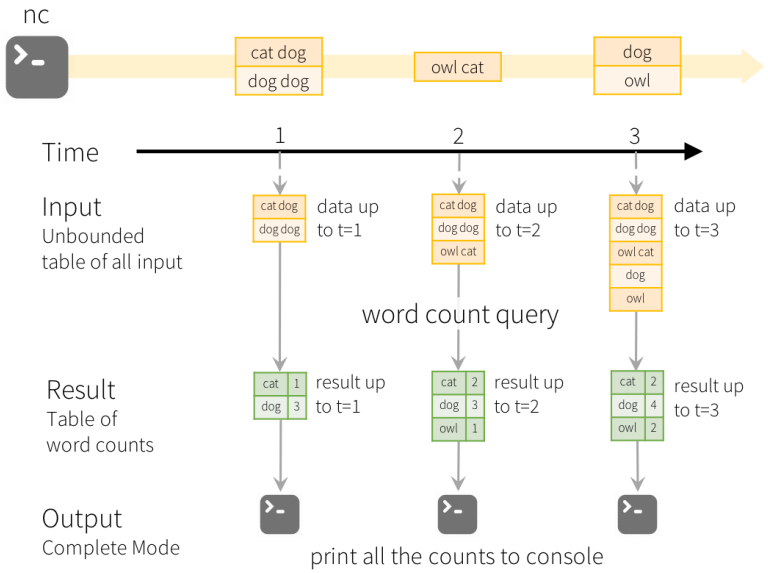
When you load a Delta table as a stream source and use it in a streaming query, the query processes all of the data present in the table as well as any new data that arrives after the stream is started.

Read and write

Databricks – Structured Streaming

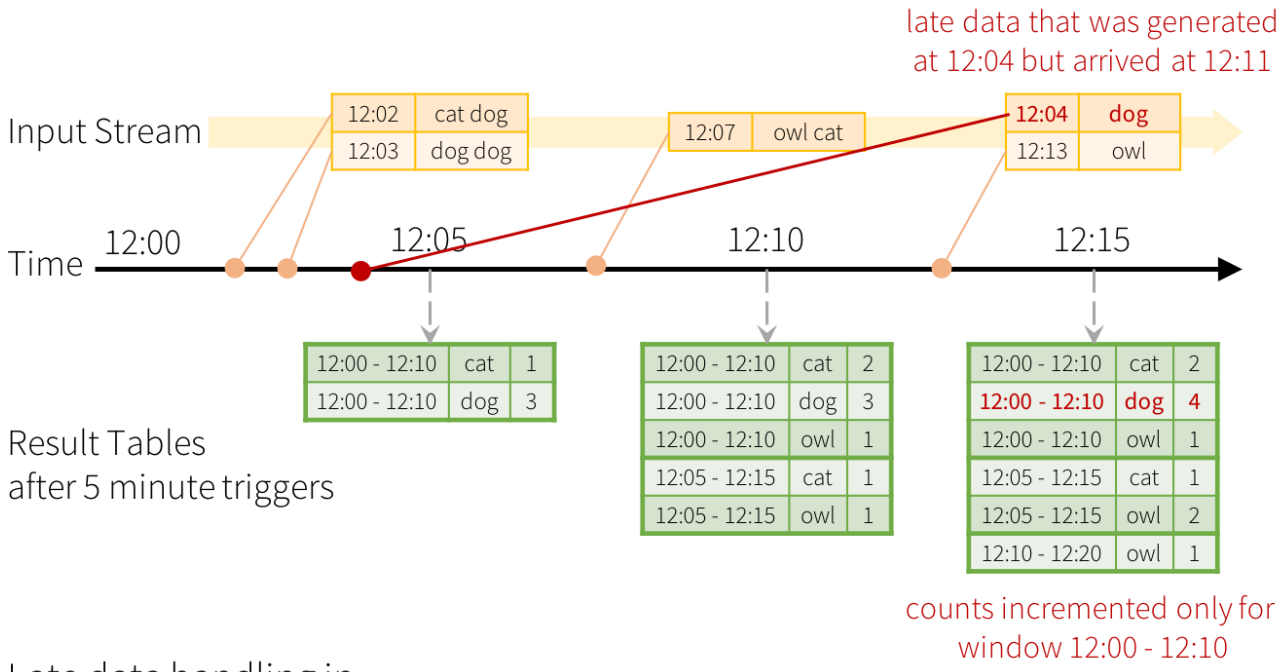
Output is defined and written in

- Complete mode
- Append mode
- Update mode



Model of the Quick Example

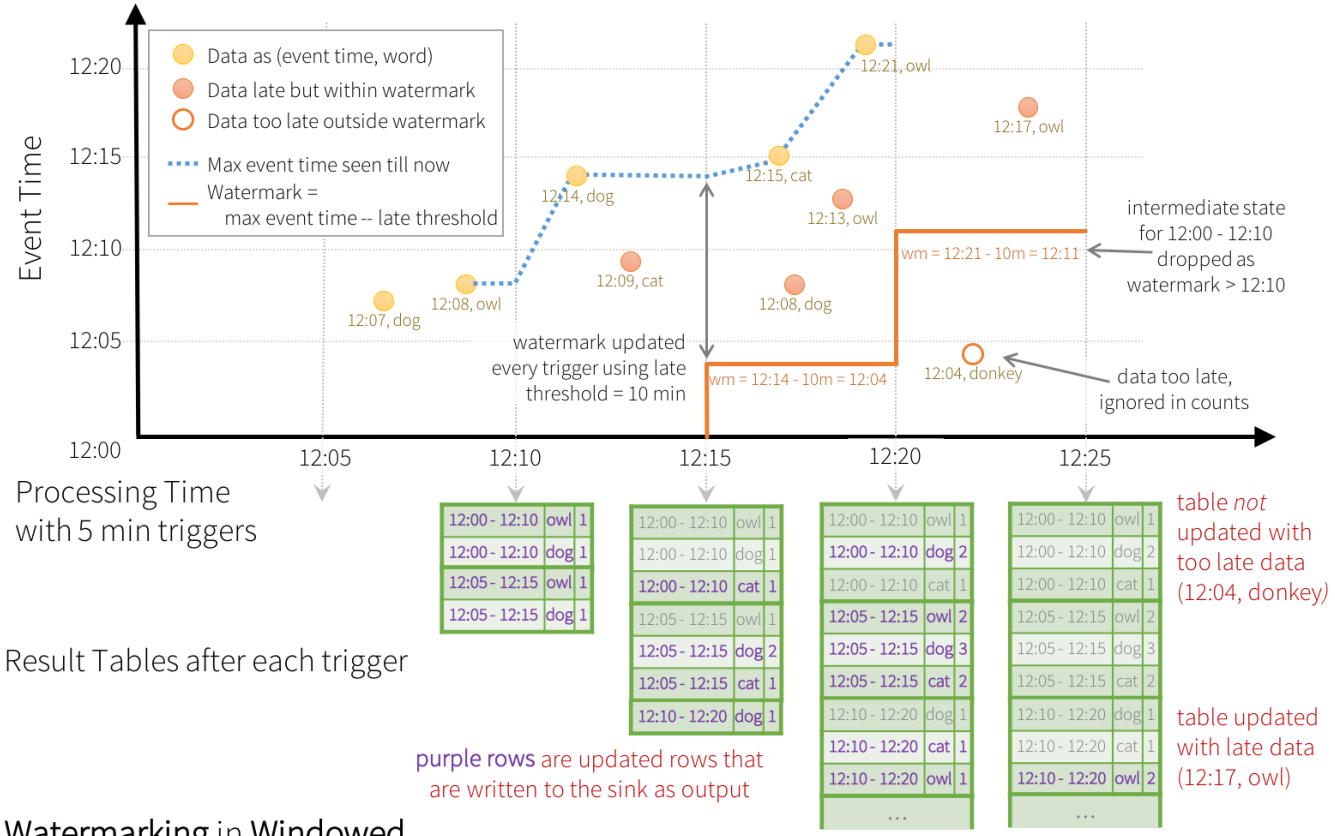
Databricks – Structured Streaming late events



Late data handling in
Windowed Grouped Aggregation

Databricks – Structured Streaming watermark

Late events are ignored



Result Tables after each trigger

purple rows are updated rows that
are written to the sink as output

Watermarking in Windowed
Grouped Aggregation with Update Mode