Technical Document: Azure Databricks Administrator

Contents

[Chapter 1: Introduction to Azure Databricks 1](#_Toc189558098)

[What is Azure Databricks: 2](#_Toc189558099)

[Key Features and Benefits: 2](#_Toc189558100)

[Databricks in Azure: 2](#_Toc189558101)

[Use Case and Industry Applications: 2](#_Toc189558102)

[Chapter 2: Create/Practice - Azure Databricks Workspace using Azure Portal 4](#_Toc189558103)

[Workspace: 4](#_Toc189558104)

[Workspace Assets: 5](#_Toc189558105)

[Chapter 3: Create and run simple spark job. 7](#_Toc189558106)

[Chapter 4: Databricks Utilities 9](#_Toc189558107)

[List available utilities: 9](#_Toc189558108)

[3. Notebook utility (dbutils.notebook): 12](#_Toc189558109)

[Chapter :5 Databricks administration 13](#_Toc189558110)

[Databricks admin types: 13](#_Toc189558111)

[What are Account admins: 14](#_Toc189558112)

[Responsibilities of Account admins: 14](#_Toc189558113)

[Assign a catalog to specific workspaces: 16](#_Toc189558114)

[What are Workspace admins: 18](#_Toc189558115)

[Workspace admin responsibilities: 19](#_Toc189558116)

[Chapter :6 Access Control Lists 23](#_Toc189558117)

[Chapter: 7 Clusters 37](#_Toc189558118)

# Chapter 1: Introduction to Azure Databricks

## What is Azure Databricks:

Azure Databricks is an analytics platform that uses Apache spark. Data scientists, engineers, and business analysts can work together using azure data bricks one-click setup, interactive workspace, and optimized workflows to analyse massive amounts of data and extract valuable insights.

Due to the enormously scalable processing power of Azure, Azure Databricks enables data engineers to execute large-scale Spark workloads, achieving unmatched speed and cost-efficiency in the cloud with auto-scaling, caching, indexing, and query optimization.

Databricks is a unified processing engine capable of analysing massive volumes of data using SQL, graph processing, machine learning, and real-time stream analysis.

## Key Features and Benefits:

The best advantage is the deep integration into the Azure subscription.

Your cloud-based extract, transform, and load procedure may use Azure data factory as one of its components.

Databricks supports several languages. Scala is the primary language, although it can also communicate effectively with  SQL and R and works well with Python.

You can save, retrieve, and update data by integrating with the azure data lake store and blob storage.

There is the possibility of working together in a notebook set. People can leave comments in the margins of the notebook, much like they can do on google docs, and may post those comments in real-time. In addition to it, there is revision control, which stores revisions.

## Databricks in Azure:

The Microsoft Azure cloud services platform is geared for use with the Azure Databricks data analytics platform. Azure data bricks provides users with the following three environments:

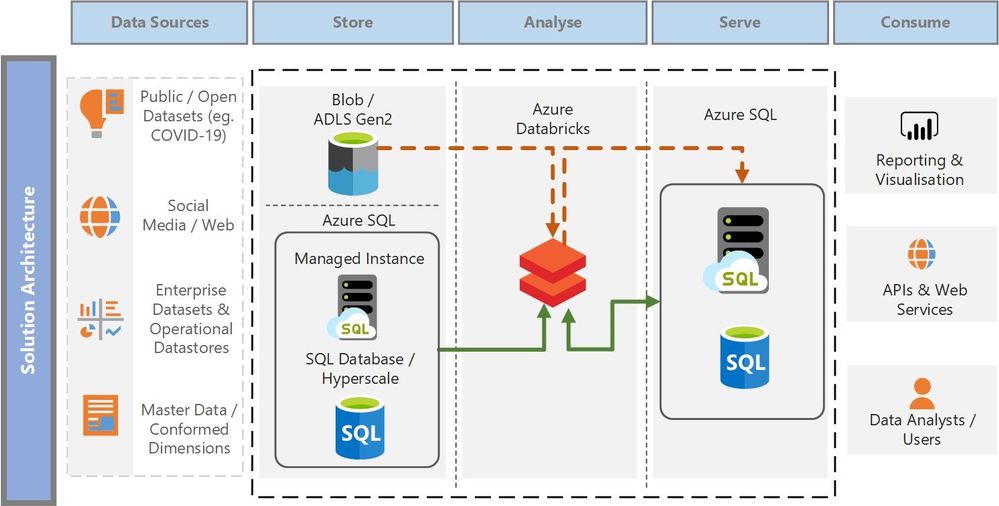
Databricks SQL

Databricks data science and engineering

Databricks machine learning

## Use Case and Industry Applications:

The architecture diagram below shows the various layers in a typical data analytics scenario.



To demonstrate the architectural pattern mentioned above, Let’s use publicly available COVID-19 dataset and runs a machine learning model in Azure Databricks to predict the fatalities of COVID-19 per day in each country.

thumbnail image 2 of blog post titled 
 
 
  
 
 
 
    
  
   
    
      
       COVID-19 data analytics and reporting with Azure Databricks and Azure SQL
       
      
     
   
  
 
   
 
 
 
 
 


The solution extracts the COVID-19 public dataset from the pandemic data lake into Azure Databricks as a Spark Data Frame.

The extracted COVID-19 dataset is cleaned, pre-processed, trained and scored using Machine Learning model.

The resulting dataset with the predicted scores is stored into a staging table in Azure SQL Managed Instance for further downstream transformation.

Common data dimension tables and the staging tables from Azure SQL Managed Instance are read into Data Frames in Azure Databricks.

The Data Frames containing the necessary dimension and staging data are further refined, joined and transformed to produce a denormalized fact table for reporting. Denormalization of data is highly recommended for reporting and data visualization as the data is structured in a way that optimizes performance of reporting queries and enables slicing-and-dicing of data as desired by business decision makers.

The resulting denormalized data is written to Azure SQL Managed Instance which is ready to serve the data to its consumers.

# Chapter 2: Create/Practice - Azure Databricks Workspace using Azure Portal

## Workspace:

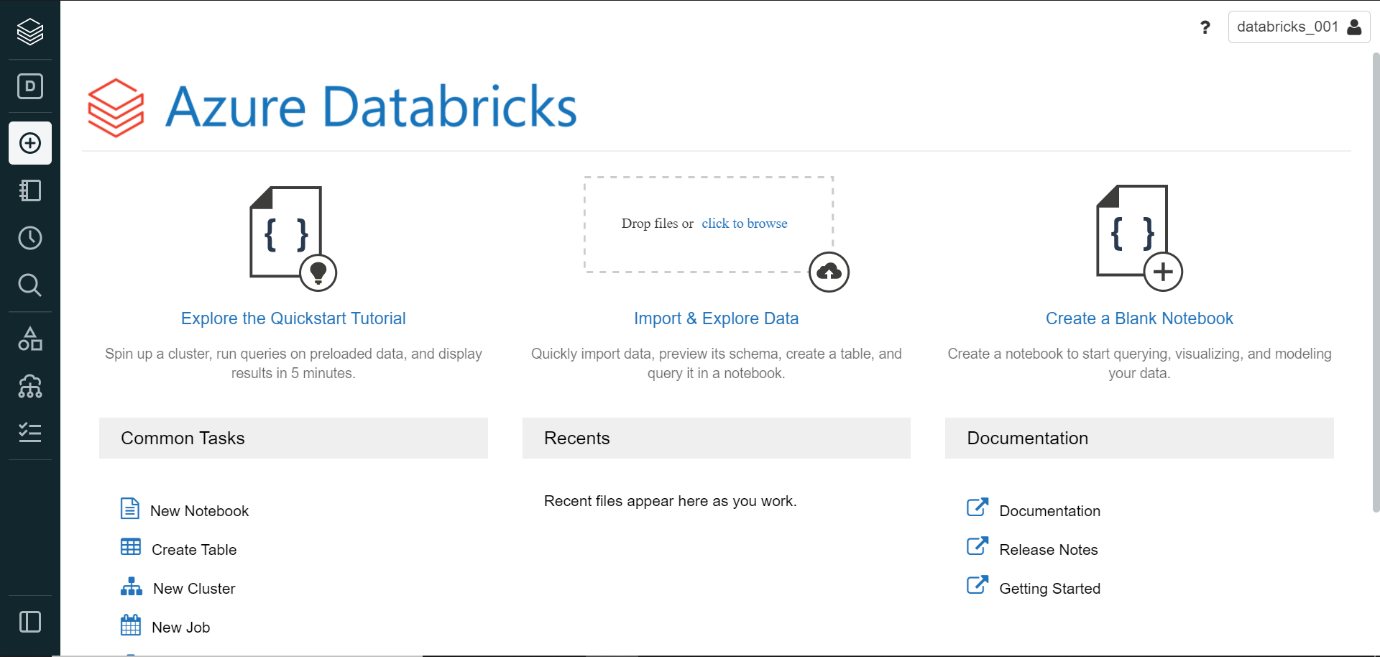
An Azure Databricks workspace is an environment for accessing all Azure Databricks Assets.

Workspace contains below objects into folders.

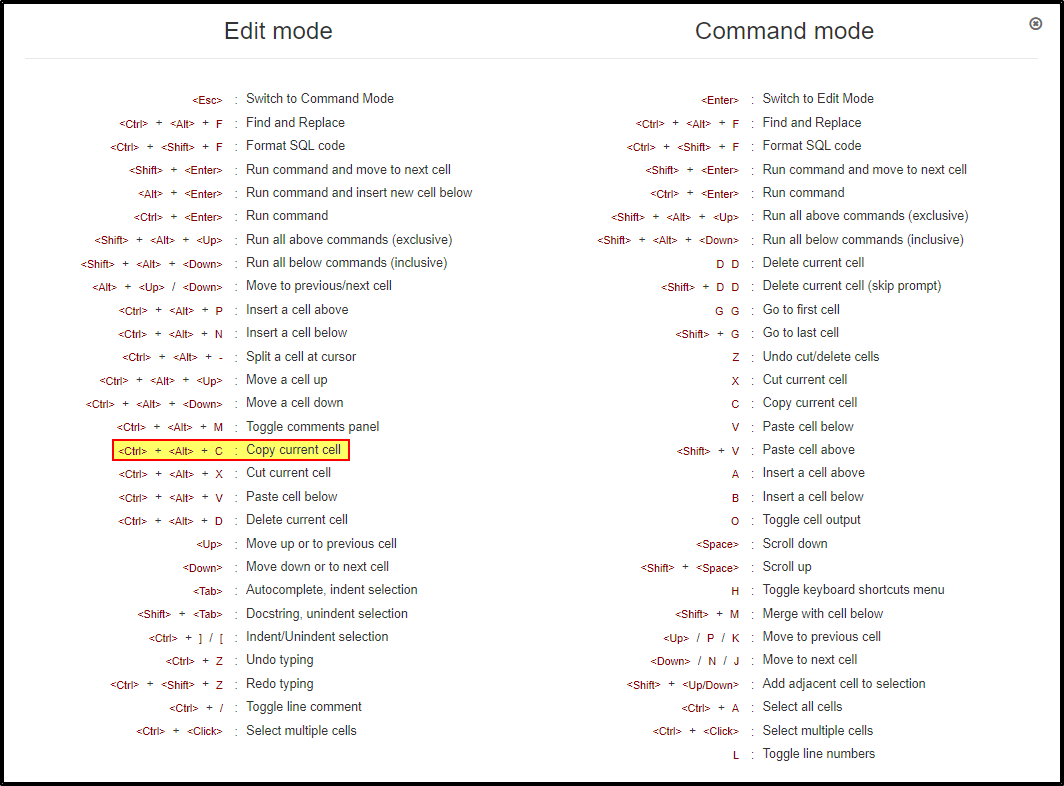
* Notebooks
* Libraries
* Experiments

We can manage the workspace using Workspace UI, Databricks CLI and Databricks REST API.

In workspace UI, we can get help by clicking “? “Icon at top right-hand side.



Some of the keyboard shortcuts for working with notebooks are display below.



## Workspace Assets:

Clusters

Notebooks

Jobs

Libraries

Data

Experiments

Cluster:

Databricks Cluster is a set of computation resources and configurations on which we can run data engineering, data science and data analytics workloads, such as production ETRL pipelines, streaming analytics, ad-hoc analytics and machine learning.

Notebook:

Notebook is a web-based interface to a document that contains runnable code, visualization and narrative text.

Data:

We can import data into distrusted file system mounted into an azure data bricks workspace and work with it in Azure Databricks notebooks and cluster.

Experiments:

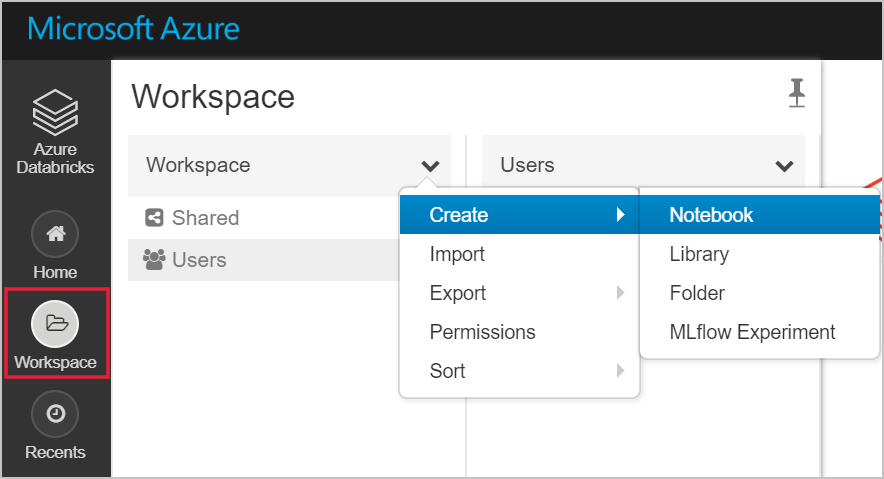
ML-Flow experiments let us run ML-flow machine learning models

Folders:

Folders contain all static assets within workspace like notebook, experiments, and other sub-folders.

Just click on folder name to open or close the folder and to view its contents.

In order to perform any action on folder just click on dropdown arrow.



Special Folders:

An azure data bricks workspace has three special folders: workspace, shared and users. We can’t rename or move special folders.

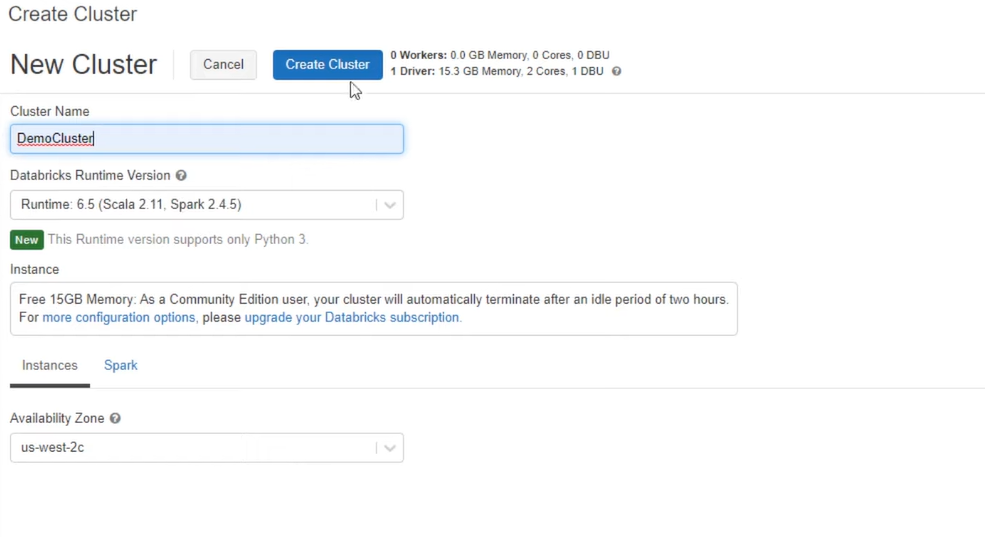
The Workspace root folder is container for all organizational Azure Databricks Static assets.

Shared is for sharing objects across the organization. All users will have full permissions for all objects in shared.

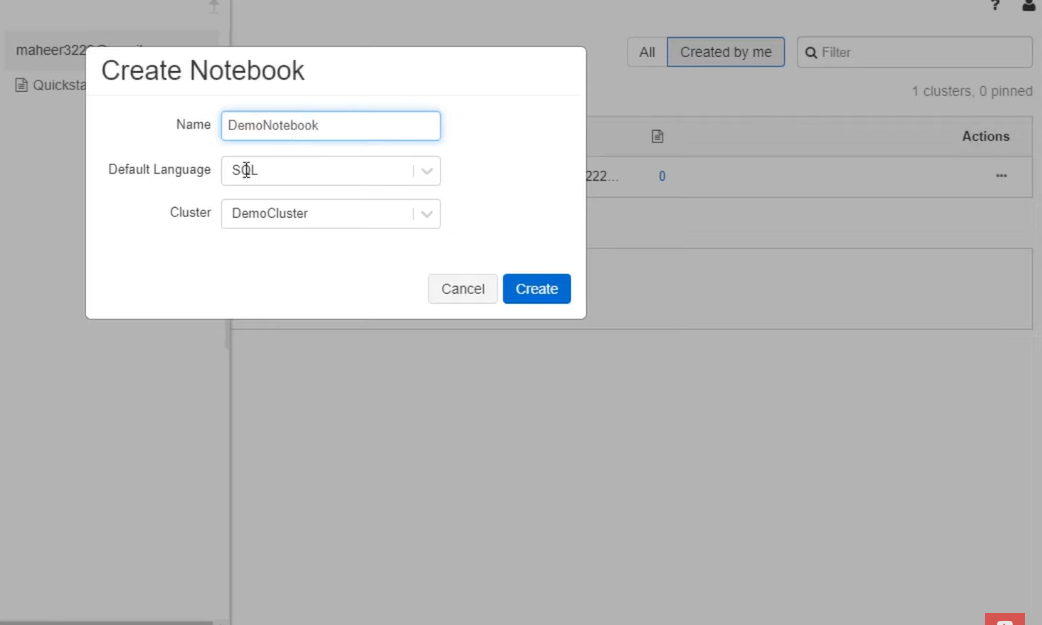
User contains a folder foe each user. We will call it as a User Home folder. Objects in the folder are by default private to that user.

# Chapter 3: Create and run simple spark job.

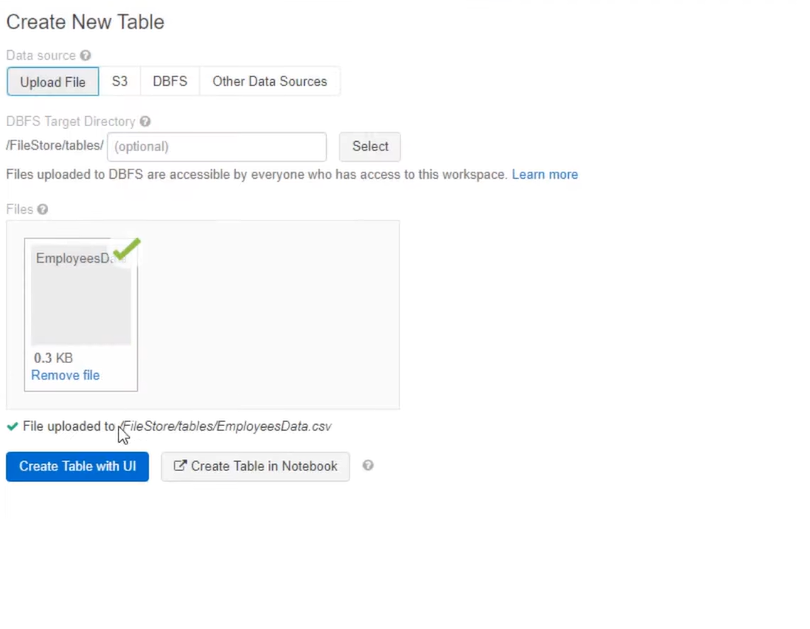
Steps 1: Create Cluster



Step 2: Create Notebook



Step 3: upload/import file in DBFS



Step 4: Create table from csv file.



Step 5: Edit Notebook and run the queries to get desired output.



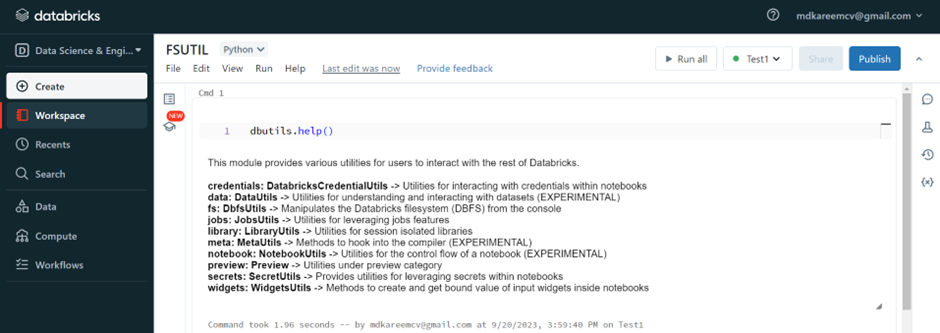
# Chapter 4: Databricks Utilities

Databricks Utilities (dbutils) make it easy to perform powerful combinations of tasks. You can use the utilities to work with object storage efficiently, to chain and parameterize notebooks, and to work with secrets.

dbutils utilities are available in Python, R, and Scala notebooks.

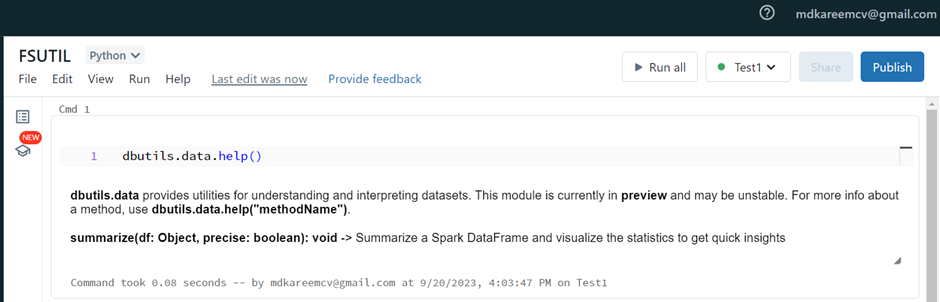
## List available utilities:

To list available utilities along with a short description for each utility, run dbutils.help() for Python.



1. Data utility (dbutils.data):

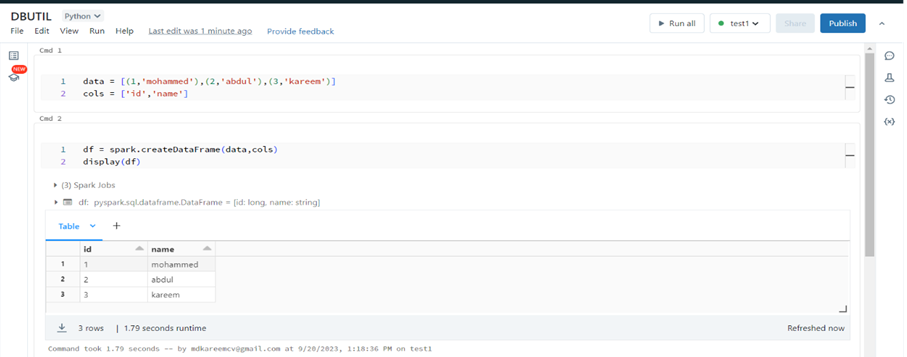
The data utility allows you to understand and interpret datasets. To list the available commands, run dbutils.data.help().



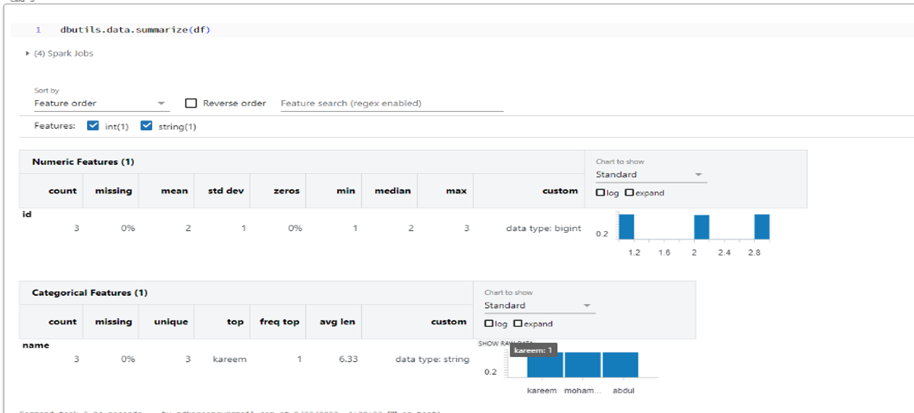
Data utility has command called summarize (tbuild.data.summarize).

Summarize command will calculate and displays summary statistics of an apache spark Data frame.

Let’s create one simple data frame.



2. Execute tbuild.data.summarize command on above data frame.



2. File system utility (dbutils.fs):

The file system utility allows us to access Databrick’s File System (DBFS), making it easier to use Databricks as a file system.

To list the available commands, run dbutils.fs.help()

Command: [cp](https://docs.databricks.com/en/dev-tools/databricks-utils.html#dbutils-fs-cp), [head](https://docs.databricks.com/en/dev-tools/databricks-utils.html#dbutils-fs-head), [ls](https://docs.databricks.com/en/dev-tools/databricks-utils.html#dbutils-fs-ls), [mkdirs](https://docs.databricks.com/en/dev-tools/databricks-utils.html#dbutils-fs-mkdirs), [mount](https://docs.databricks.com/en/dev-tools/databricks-utils.html#dbutils-fs-mount), [mounts](https://docs.databricks.com/en/dev-tools/databricks-utils.html#dbutils-fs-mounts), [mv](https://docs.databricks.com/en/dev-tools/databricks-utils.html#dbutils-fs-mv), [put](https://docs.databricks.com/en/dev-tools/databricks-utils.html#dbutils-fs-put), [refreshMounts](https://docs.databricks.com/en/dev-tools/databricks-utils.html#dbutils-fs-refreshmounts), [rm](https://docs.databricks.com/en/dev-tools/databricks-utils.html#dbutils-fs-rm), [unmount](https://docs.databricks.com/en/dev-tools/databricks-utils.html#dbutils-fs-unmount), [updateMount](https://docs.databricks.com/en/dev-tools/databricks-utils.html#dbutils-fs-updatemount)

2.1. ls command (dbutils.fs.ls):

Use to lists the contents of a directory.

2.2. cp command (dbutils.fs.cp):

Copies a file or directory, possibly across filesystems.

2.3. head command (dbutils.fs.head):

Returns up to the specified maximum number bytes of the given file.

2.4. mkdirs command (dbutils.fs.mkdirs):

Use to creates the given directory if it does not exist.

2.5. mounts command (dbutils.fs.mounts):

It is used to mount the specified source directory into DBFS.

We can mount Azure Blob storage either by using Account key or by using SAS key.

Syntax:

dbutils.fs.mount(

[Source=”wasbs://<container-name>@<storage-acount-name>.blob.core.windows.net](mailto:Source=)”,

mount\_point=”/mnt/<mount-name>”,

extra\_configs={“<conf-key>”:”AccountKey”)}

)

In Case of Account key:

<conf-key> is fs.azure.account.key<storage-account-name>.blob.core.windows.net

In Case of SAS key:

<conf-key>isfs.azure.sas.<container-name>.<storage-account-name>.blob.core.windows.net

2.6. mv command (dbutils.fs.mv):

Moves a file or directory, possibly across filesystems. A move is a copy followed by a delete, even for moves within filesystems.

2.7. put command (dbutils.fs.put):

Writes the specified string to a file.

Following example will writes the string “Hello, Databricks!” to a file named hello\_db.txt under /temp directory, If the file exits, it will be overwritten.

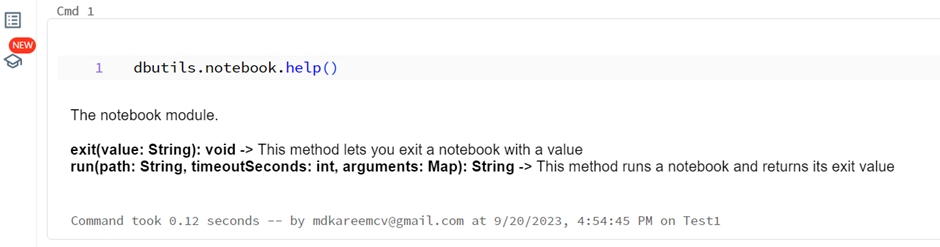
dbutils.fs.put("/tmp/hello\_db.txt", "Hello, Databricks!", True)

## 3. Notebook utility (dbutils.notebook):

The notebook utility allows you to chain together notebooks and act on their results.

Command: exit, run

To list the available commands, run dbutils.notebook.help



3.1. run command (dbutils.notebook.run):

Runs a notebook and returns its exit value. The notebook will run in the current cluster by default.

3.2. exit command (dbutils.notebook.exit):

Exits a notebook with a value.

cmd2 will skipped and will not be executed, as we have assigned exit value in cmd1.

# Chapter :5 Databricks administration

## Databricks admin types:

1. Account Admins:

Manage the Databricks account, including workspace creation, user management, cloud resources, and account usage monitoring.

2.Workspace Admins:

Manage workspace identities, access control, settings, and features for individual workspaces in the account.

Additionally, users can be assigned these feature-specific admin roles, which have narrower sets of privileges.

Marketplace amins:

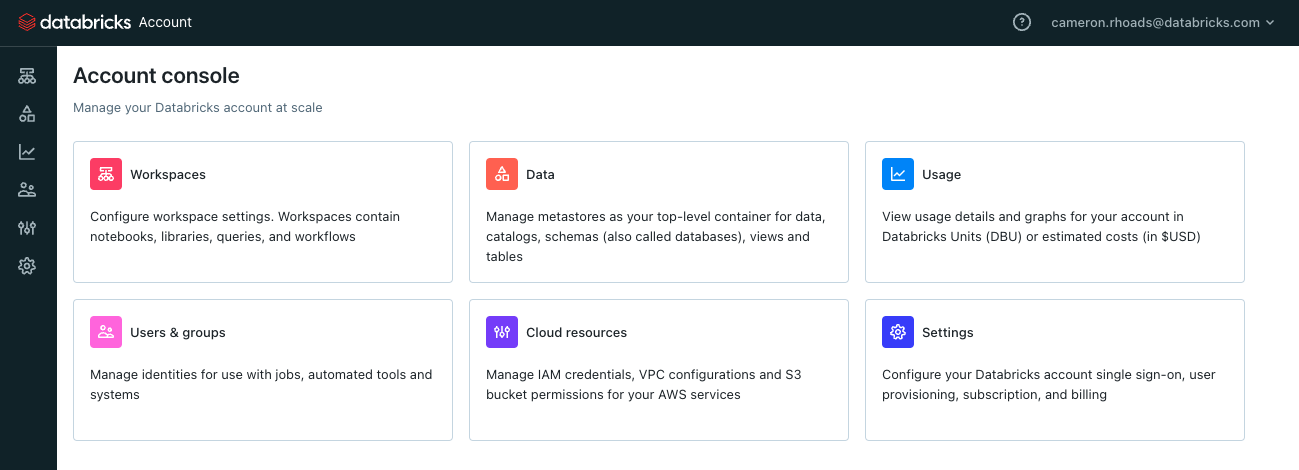
Manage their account’s Databricks Marketplace provider profile, including creating and managing Marketplace listings.

Metastore admins:

Manage privileges and ownership for all securable objects within a Unity catalog meta store, such as who can create catalogs or query a table.

Account console:

The account console is where account admins manage their Databricks account.



## What are Account admins:

Account admins have privileges over the entire Databricks account. As an account admin, we can create workspaces, configure cloud resources, view usage data, and manage account identities, settings, and subscriptions.

Account admins can also delegate the account admin and workspace admin roles to any other user.

## Responsibilities of Account admins:

As an account admin, your responsibilities include:

Creating and managing workspaces

Enabling Unity Catalog.

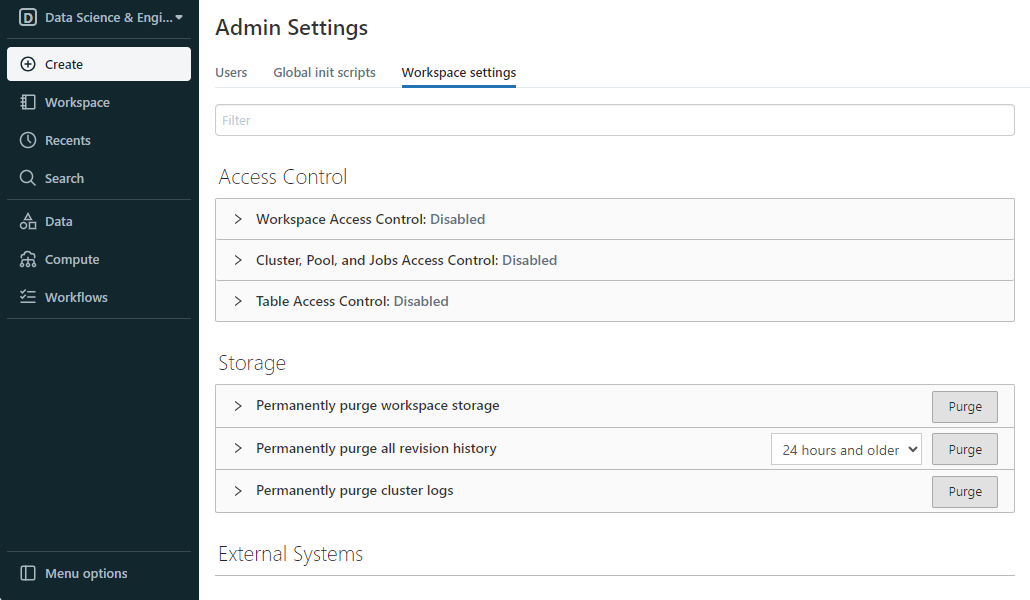
Managing Identities

Monitoring account usage Logs

Managing the account subscription

1. Create and manage workspaces:

Only account admins can create new workspaces, the admin settings page is where we can manage features and settings for your Databricks workspace. To access the admin settings, click your username in the top bar of the Databricks workspace and select Admin Settings.

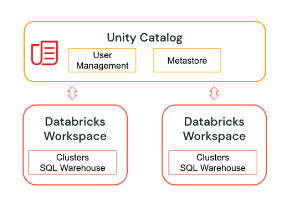


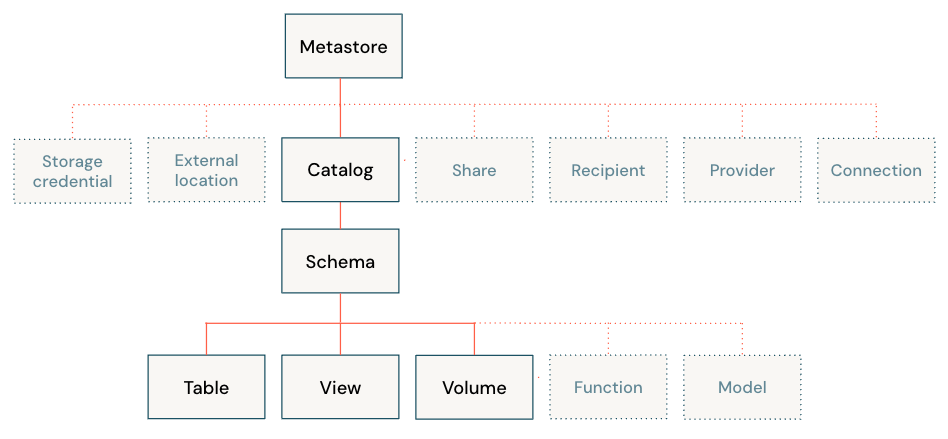
2.Enabling Unity Catalog:

An account admin is needed to enable Unity Catalog in your account. The process involves creating a Unity Catalog metastore, which can only be done by an account admin.

What is Unity Catalog:

Unity Catalog provides centralized access control, auditing, lineage, and data discovery capabilities across Databricks workspaces.





2.1. Metastore: A meta store is the top-level container of objects in Unity Catalog. It stores metadata about data assets (tables and views) and the permissions that govern access to them. Databricks account admins should create one meta store for each region in which they operate and assign them to Databricks workspaces in the same region.

2.2. Catalog: A catalog is the first layer of Unity Catalog’s three-level namespace. It’s used to organize your data assets.

Steps to create catalog:

Log in to a workspace that is linked to the metastore.

Click Catalog.

Click the Create Catalog button.

Specify the location where data for managed tables in the catalog will be stored.

Click Create.

Specify the workspace that the catalog is bound to. by default, the catalog is shared with all workspaces attached to the current metastore. If the catalog will contain data that should be restricted to specific workspaces, go to the Workspaces tab and add those workspaces.

Assign permissions for your catalog.

## Assign a catalog to specific workspaces:

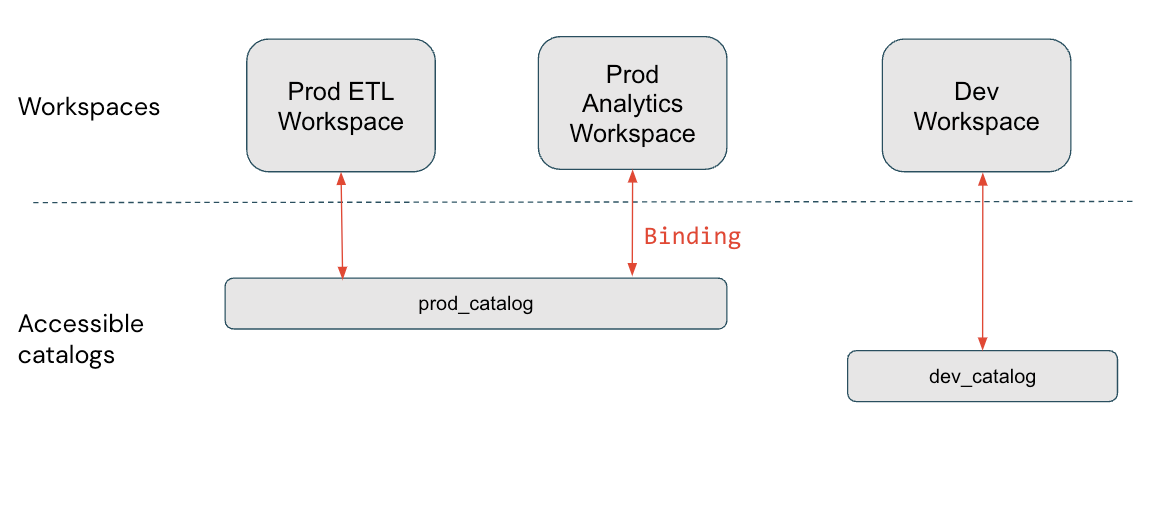
We can limit catalog access to specific workspaces in the account, also known as workspace-catalog binding. The default is to share the catalog with all workspaces attached to the current metastore.

Typical use cases for binding a catalog to specific workspaces include:

Ensuring that users can only access production data from a production workspace environment.

Ensuring that users can only process sensitive data from a dedicated workspace.

Let's take an example of production and development isolation. Here production data catalogs can only be accessed from production workspaces, this supersedes any individual grants that are issued to users.



In this diagram, prod\_catalog is bound to two production workspaces. Suppose a user has been granted access to a table in prod\_catalog called my\_table (using GRANT SELECT ON my\_table TO <user>). If the users try to access my\_table in the Dev workspace, they receive an error message. The user can access my\_table only from the Prod ETL and Prod Analytics workspaces.

2.3. Schema: A schema (also called a database) is the second layer of Unity Catalog’s three-level namespace. A schema organizes tables and views. Users can see all schemas on which they have been assigned the USE SCHEMA permission, along with the USE CATALOG permission on the schema’s parent catalog. To access or list a table or view in a schema, users must also have SELECT permission on the table or view.

2.4. Volume:

A volume resides in the third layer of Unity Catalog’s three-level namespace. Volumes contain directories and files for data stored in any format. Volumes provide non-tabular access to data, meaning that files in volumes cannot be registered as tables.

To create the volume user must have CREATE VOLUME and USE SCHEMA permissions on the schema, and they must have the USE CATALOG permission on its parent catalog.

To read files and directories stored inside a volume, users must have the READ VOLUME permission, the USE SCHEMA permission on its parent schema, and the USE CATALOG permission on its parent catalog.

To add, remove, or modify files and directories stored inside a volume, users must have WRITE VOLUME permission, the USE SCHEMA permission on its parent schema, and the USE CATALOG permission on its parent catalog.

A volume can be managed or external.

3.Manage identities:

If Unity Catalog is enabled for at least one workspace in account, identities (users, groups, and service principals) should be managed in the account console. Account admins can grant permissions and assign workspaces to these identities.

3.1. Databricks identities:

There are three types of Databricks identity:

Users: User identities recognized by Databricks and represented by email addresses.

Service principals: Identities for use with jobs, automated tools, and systems such as scripts, apps, and CI/CD platforms.

Groups: A collection of identities used by admins to manage group access to workspaces, data, and other securable objects. All Databricks identities can be assigned as members of groups. There are two types of groups in Databricks: account groups and workspace-local groups.

3.2. Who can manage identities in Databricks?

To manage identities in Databricks, you must have one of the following roles on a service principal or group.

Account admins can add users, service principals, and groups to the account and assign them admin roles. Account admins can update and delete users, service principals, and groups in the account. They can give users access to workspaces, as long as those workspaces use [identity federation](https://docs.databricks.com/en/administration-guide/users-groups/index.html#assign-users-to-workspaces).

Workspace admins can add users, service principals to the Databricks account. They can also add groups to the Databricks account if their workspaces are enabled for identity federation. Workspace admins can grant users, service principals, and groups access to their workspaces. They cannot delete users and service principals from the account. Workspace admins can also manage workspace-local groups.

Group managers can manage group membership and delete the group. They can also assign other users the group manager role. Account admins have the group manager role on all groups in the account. Workspace admins have the group manager role on account groups that they create.

Service principal managers can manage roles on a service principal. Account admins have the service principal manager role on all service principals in the account. Workspace admins have the service principal manager role on service principals that they create.

3.3. Assigning admin roles

Account admins can assign other users as account admins. They can also become Unity Catalog metastore admins by creating a metastore, and they can transfer the metastore admin role to another user or group.

Both account admins and workspace admins can assign other users as workspace admins. The workspace admin role is determined by membership in the workspace admins group, which is a default group in Databricks and cannot be deleted.

Account admins can also assign other users as Marketplace admins.

3.4. How to Assign account admin roles to a user:

Step 1: As an account admin, log in to the [account console](https://accounts.cloud.databricks.com/).

Step 2: In the sidebar, click User management.

Step 3: Find and click the username.

Step 4: On the Roles tab, turn on Account admin or Marketplace admin.

4 Monitor account usage logs:

Only account admins can configure audit logs for their account, for information on audit logs.

Account admins can also view and download billable usage logs from the account console.

5 Manage account subscription:

Account admins can manage aspects of their Databricks subscription from the account console.

## What are Workspace admins:

Workspace admins have admin privileges within a single workspace. They can manage workspace-level identities, regulate compute use, and enable and delegate role-based access control.

Workspace admins are the only users who have access to the workspace’s admin settings page. As a workspace admin, you can access admin settings by clicking your username in the top bar of the Databricks workspace and selecting Admin Settings.

## Workspace admin responsibilities:

As a workspace admin, major responsibilities include:

Managing identities in the workspace

Creating and managing compute resources

Managing workspace features and settings

1. Manging identities in workspaces:

If workspace is enabled for unity catalog, identities should be added at the account level. Workspace admins can then assign users, groups, and service principals to their workspace.

1.1. Assign the workspace admin role to a user using the workspace admin settings page:

To assign the workspace admin role using the workspace admin settings page, do the following:

Step 1: As a workspace admin, log in to the Databricks workspace.

Step 2: Click your username in the top bar of the Databricks workspace and select Admin Settings.

Step 3: On the Users tab, find the user and select the admin checkbox.

To remove the admin role from a workspace user, perform the same steps, but clear the admin checkbox.

1.2. Assign a user to a workspace using the workspace admin settings page:

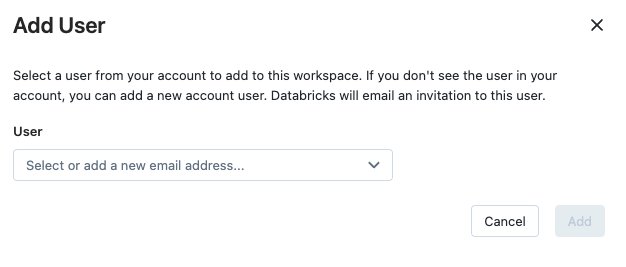
To add a user to a workspace using the workspace admin settings page, do the following:

Step 1: As a workspace admin, log in to the Databricks workspace.

Step 2: Click your username in the top bar of the Databricks workspace and select Admin Settings.

Step 3: On the Users tab, click Add User.

Step 4: Select an existing user to assign to the workspace or create a new one.



Note: To create a new user, click the drop-down arrow in the search box and then click + Add new user.

Step 5: Click Add.

1.3. Remove a user from a workspace using the workspace admin settings page:

Step 1: As a workspace admin, log in to the Databricks workspace.

Step 2: Click your username in the top bar of the Databricks workspace and select Admin Settings.

Step 3: On the Users tab, find the user and click the Remove User Icon at the far right of the user row.

Step 4: Click Delete to confirm.

1.4. Deactivate a user in your Databricks workspace:

Workspace admins can deactivate users in a Databricks workspace. A deactivated user cannot login to the workspace or access it from Databricks APIs, however all of the user’s permissions and workspace objects remain unchanged. When a user is deactivated:

The user cannot login to the workspaces from any method.

Applications or scripts that use the tokens generated by the user can no longer access the Databricks API. The tokens remain but cannot be used to authenticate while a user is deactivated.

Notebooks owned by the user remain.

Clusters owned by the user remain running.

Scheduled jobs created by the user have to be assigned to a new owner to prevent them from failing.

When a user is reactivated, they can login to the workspace with the same permissions. Databricks recommends deactivating users instead of removing them because removing a user is a destructive action. We cannot deactivate a user using the workspace admin settings page, instead use the Workspace Users API.

2. Create and manage compute resources

Workspace admins can create SQL warehouses (a compute resource that lets run SQL commands on data objects within Databricks SQL) and clusters for their workspace users.

To create a SQL warehouse user must be a workspace admin or a user with unrestricted cluster creation permissions.

To manage a SQL warehouse, user must be a workspace admin or have the Can Manage permission on the SQL warehouse.

2.1. To create a SQL warehouse using the web UI:

Step 1: Click SQL Warehouses in the sidebar.

Step 2: Click Create SQL Warehouse.

Step 3: Enter a Name for the warehouse.

Step 4: Accept the default warehouse settings or edit them.

Step 5: Click Create.

2.2. Different type of Warehouse:

Databricks SQL supports three warehouse types, each with different levels of performance and feature support.

Classic: Supports entry level performance features and a limited set of Databricks SQL functionality.

Pro: Supports additional Databricks SQL performance features (compared to classic) and supports all Databricks SQL functionality.

Serverless: Supports all features in the pro SQL warehouse type, as well as advanced Databricks SQL performance features. SQL warehouses run in the customer’s Databricks account using serverless compute.

2.3. Warehouse settings:

Creating a SQL warehouse in the UI allows to update the following settings:

Cluster Size: represents the size of the driver node and number of worker nodes associated with the cluster. The default is X-Large. To reduce query latency, increase the size.

Auto Stop: determines whether the warehouse stops if it’s idle for the specified number of minutes. Idle SQL warehouses continue to accumulate DBU and cloud instance charges until they are stopped.

For Pro and classic SQL warehouses the default is 45 minutes, which is recommended for typical use. The minimum is 10 minutes.

For Serverless SQL warehouses the default is 10 minutes, which is recommended for typical use. The minimum is 5 minutes when you use the UI.

Scaling: sets the minimum and maximum number of clusters that will be used for a query. The default is a minimum and a maximum of one cluster. We can increase the maximum clusters if want to handle more concurrent users for a given query. Databricks recommends a cluster for every 10 concurrent queries.

Type: determines the type of warehouse. If serverless is enabled account, serverless will be the default

2.4. Monitor a SQL warehouse:

To monitor a SQL warehouse, click the name of a SQL warehouse and then the Monitoring tab. On the Monitoring tab, following monitoring elements will show:

Live statistics: Live statistics show the currently running and queued queries, active SQL sessions, the warehouse status, and the current cluster count.

Query count chart: The query count chart shows the number of queries running or queued on the warehouse during the selected time frame. ￼

Query history table: The query history table shows all of the queries active during the selected time frame, their start time and duration, and the user that executed the query. We can filter the queries by user, query duration, query status, and query type.

Time scale filter: The monitoring time scale filter sets the time range for the query count chart, running cluster chart, and the query history and event log table. The default time range is 8 hours, but we can specify 24 hours, 7 days, or 14 days. we can also click and drag on the bar chart to change the time range.

Running clusters chart: The running clusters chart shows the number of clusters allocated to the warehouse during the selected time frame. During a cluster recycle, this count may temporarily exceed configured maximum.

3. Manage workspaces features and settings:

Workspace admins are responsible for managing select workspace behaviour and settings.

3.1 Databricks SQL admin settings:

The admin settings page includes a tab for SQL Settings. These settings control the Databricks SQL presentation and behaviour for all Databricks SQL users in organization. These settings include:

Date & Time Format: The default date and time formats in query visualizations.

Visualization: Whether to display the Plotly toolbar in chart visualizations.

Download: Whether to allow users to download query results.

Dashboard subscriptions: Enable users to subscribe to dashboards in the workspace.

Failure emails: Decide which owners are emailed when refresh schedules fail.

SAML Token Expiry Warning: Whether to prompt the user to log in again if SAML authentication is used and their SAML token will expire within a few minutes. This helps to prevent disruption when running queries or refreshing dashboards. We can disable this option if you do not require SAML tokens to run queries.

# Chapter :6 Access Control Lists

In Databricks, we can use access control lists (ACLs) to configure permission to access clusters, pools, jobs, and workspace objects like notebooks, experiments, and folders. All users can create and modify objects unless access control is enabled on that object. workspace admins have privilege to enable and disable access control.

1. Enable access control for workspace objects:

Step 1: Go to the admin setting Page.

Step 2: Click the Workspace Settings tab.

Step 3: Click the Workspace Access Control toggle.

Step 4: Click Confirm.

2. Enable access control for clusters, jobs, and pools:

Step 1: Go to the admin setting Page.

Step 2: Click the Workspace Settings tab.

Step 3: Click the Cluster, Pool and Jobs Access Control toggle.

Step 4: Click Confirm.

3. Prevent users from seeing objects they do not have access to

Step 1: Go to the admin setting Page.

Step 2: Click the Workspace Settings tab.

Step 4: Click the Workspace Visibility Control toggle to prevent users from seeing objects in the workspace file browser that they do not have access to.

Step 5: Click the Cluster Visibility Control toggle to prevent users from seeing clusters that they do not have access to.

Step 6: Click the Job Visibility Control toggle to prevent users from seeing jobs that they do not have access to.

1. Workspace object access control:

With workspace object access control, individual permissions determine a user’s abilities to modify workspace objects. An Azure Databricks workspace admin must enable it for the workspace.

Folder permissions: We can assign five permission levels to Folders: No Permissions, Can Read, Can Run, Can Edit, and Can Manage. The table lists the abilities for each permission.

| Ability | No Permissions | Can Read | Can Run | Can Edit | Can Manage |
| --- | --- | --- | --- | --- | --- |
| List items in folder | X | x | x | x | x |
| View items in folder |  | x | x | x | x |
| Clone and export items |  | x | x | x | x |
| Create, import, and delete items |  |  |  |  | x |
| Move and rename items |  |  |  |  | x |
| Change permissions |  |  |  |  | x |

Notebook permissions: We can assign five permission levels to Notebooks: No Permissions, Can Read, Can Run, Can Edit, and Can Manage. The table lists the abilities for each permission.

| Ability | No Permissions | Can Read | Can Run | Can Edit | Can Manage |
| --- | --- | --- | --- | --- | --- |
| View cells |  | x | x | x | x |
| Comment |  | x | x | x | x |
| Run via %run or notebook workflows |  | x | x | x | x |
| Attach and detach notebooks |  |  | x | x | x |
| Run commands |  |  | x | x | x |
| Edit cells |  |  |  | x | x |
| Change permissions |  |  |  |  | x |

File permissions: We can assign five permission levels to files: No Permissions, Can Read, Can Run, Can Edit, and Can Manage. The table lists the abilities for each permission.

| Ability | No Permissions | Can Read | Can Run | Can Edit | Can Manage |
| --- | --- | --- | --- | --- | --- |
| Read file |  | x | x | x | x |
| Comment |  | x | x | x | x |
| Attach and detach file (.py, .R, .sql, .scala) |  |  | x | x | x |
| Run file interactively (.py, .R, .sql, .scala) |  |  | x | x | x |
| Edit file |  |  |  | x | x |
| Change permissions |  |  |  |  | x |

Repos permissions: We can assign five permission levels to repos: No Permissions, Can Read, Can Run, Can Edit, and Can Manage. The table lists the abilities for each permission.

| Ability | No Permissions | Can Read | Can Run | Can Edit | Can Manage |
| --- | --- | --- | --- | --- | --- |
| List items in repo | X | x | x | x | x |
| View items in repo |  | x | x | x | x |
| Clone and export items |  | x | x | x | x |
| Run notebooks in repo |  |  | x | x | x |
| Edit notebooks in repo |  |  |  | x | x |
| Create, import, and delete items |  |  |  |  | x |
| Move and rename items |  |  |  |  | x |
| Change permissions |  |  |  |  | x |

Configure notebook, folder, and repos permissions:

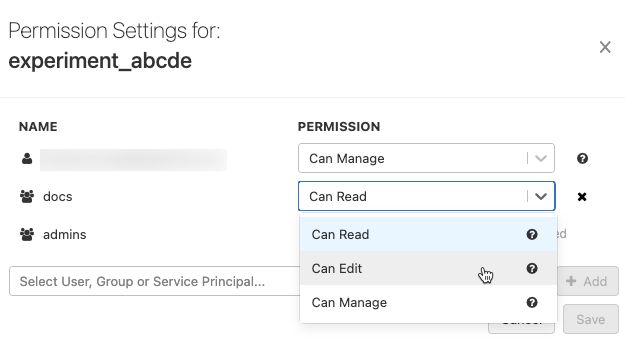
Step 1: Select **Permissions** from the drop-down menu for the notebook, folder, or repo.

A screenshot of a computer

Description automatically generated

Step 2: To grant permissions to a user or group, select from the **Add Users, Groups, and Service Principals** drop-down, select the permission, and click **Add**:A screenshot of a computer

Description automatically generated

Step 3: To change the permissions of a user or group, select the new permission from the permission drop-down:

Step 4: After changes are done in the dialog, **Done** changes to **Save Changes** and a **Cancel** button appears. Click **Save Changes** or **Cancel**.

2. Cluster access control:

2.1. Types of permissions

We can configure two types of cluster permissions:

The Allow unrestricted cluster creation entitlement controls the ability to create clusters.

Cluster-level permissions control ability to use and modify a specific cluster.

When cluster access control is enabled:

An administrator can configure whether a user can create clusters.

Any user with Can Manage permission for a cluster can configure whether a user can attach to, restart, resize, and manage that cluster.

2.2. Cluster-level permissions:

There are four permission levels for a cluster: No Permissions, Can Attach To, Can Restart, and Can Manage. The table lists the abilities for each permission.

| Ability | No Permissions | Can Attach To | Can Restart | Can Manage |
| --- | --- | --- | --- | --- |
| Attach notebook to cluster |  | x | x | x |
| View Spark UI |  | x | x | x |
| View cluster metrics |  | x | x | x |
| View driver logs |  | x | x | x |
| Terminate cluster |  |  | x | x |
| Start cluster |  |  | x | x |
| Restart cluster |  |  | x | x |
| Edit cluster |  |  |  | x |
| Attach library to cluster |  |  |  | x |
| Resize cluster |  |  |  | x |
| Modify permissions |  |  |  | x |

2.3. Configure cluster-level permissions.

Cluster access control must be [enabled](https://learn.microsoft.com/en-us/azure/databricks/security/auth-authz/access-control/enable-access-control#cluster-acl) and you must have Can Manage permission for the cluster.

Click Compute in the sidebar.

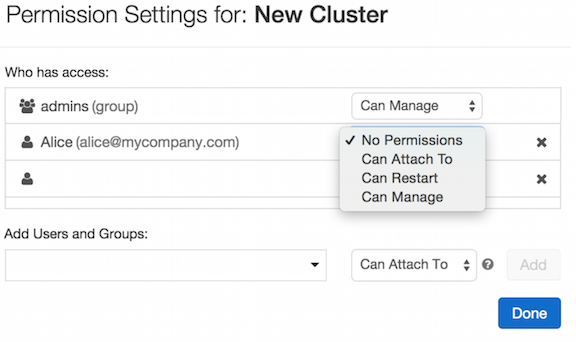
Click the name of the cluster you want to modify.

Click Permissions at the top of the page.

In the Permission settings for dialog, you can:

Select users and groups from the Add Users and Groups drop-down and assign permission levels for them.

Update cluster permissions for users and groups that have already been added, using the drop-down menu beside a user or group name.



A screenshot of a computer

Description automatically generated

3. Jobs access control:

Enabling access control for jobs, allows job owners to control who can view job results or manage runs of a job. an Azure Databricks workspace admin must enable it for the workspace.

3.1. Job permissions:

There are five permission levels for jobs: No Permissions, Can View, Can Manage Run, Is Owner, and Can Manage. Workspace admins are granted the Can Manage permission by default, and they can assign that permission to non-admin users.

The table lists the abilities for each permission.

| Ability | No Permissions | Can View | Can Manage Run | Is Owner | Can Manage |
| --- | --- | --- | --- | --- | --- |
| View job details and settings | X | x | x | x | x |
| View results, Spark UI, logs of a job run |  | x | x | x | x |
| Run now |  |  | x | x | x |
| Cancel run |  |  | x | x | x |
| Edit job settings |  |  |  | x | x |
| Modify permissions |  |  |  | x | x |
| Delete job |  |  |  | x | x |

3.2. Configure job permissions:

User must have Can Manage or Is Owner permission.

Go to the details page for a job.

Click the Edit permissions button in the Job details panel.

In the pop-up dialog box, assign job permissions via the drop-down menu beside a user’s name.

A screenshot of a computer

Description automatically generated

4 Hive meta store table access control:

Each Azure Databricks workspace deploys with a built-in Hive metastore as a managed service. An instance of the metastore deploys to each cluster and securely accesses metadata from a central per-workspace repository.

By default, a cluster allows all users to access all data managed by the workspace’s built-in Hive metastore unless table access control is enabled for that cluster. Table access control lets you programmatically grant and revoke access to objects in your workspace’s Hive metastore from Python and SQL. When table access control is enabled, users can set permissions for data objects that are accessed using that cluster.

4.1 Enable table access control for a cluster:

Table access control is available in two versions:

[SQL-only table access control](https://learn.microsoft.com/en-us/azure/databricks/data-governance/table-acls/table-acl#sql-only-table-access-control), which restricts users to SQL commands.

[Python and SQL table access control](https://learn.microsoft.com/en-us/azure/databricks/data-governance/table-acls/table-acl#python-and-sql-table-access-control), which allows users to run SQL, Python, and PySpark commands.

To enable SQL-only table access control on a cluster and restrict that cluster to use only SQL commands, set the following flag in the cluster’s [Spark conf](https://learn.microsoft.com/en-us/azure/databricks/clusters/configure#spark-configuration):

spark.databricks.acl.sqlOnly true

4.2 Enable table access control for your workspace:

Before users can configure Python and SQL table access control, an Azure Databricks workspace must enable table access control for the Azure Databricks workspace and deny users access to clusters that are not enabled for table access control.

Step 1: Go to the admin setting page.

Step 2: Click the Workspace Settings tab.

Step 3: Click the Cluster, Pool and Jobs Access Control toggle.

Step 4: Click Confirm.

Step 5: Click the Table Access Control toggle.

Step 6: Click Confirm.

Note: To ensure that users access only the data that we want them to, we must restrict users to clusters with table access control enabled. We should ensure that:

Users do not have permission to create clusters. If they create a cluster without table access control, they can access any data from that cluster.

Users do not have Can Attach To permission for any cluster that is not enabled for table access control.

A screenshot of a password

Description automatically generated

5 Manage privileges on objects in the Hive metastore:

Privileges on data objects managed by the Hive metastore can be granted by either a workspace admin or the owner of an object. We can manage privileges for Hive metastore objects by using SQL commands.

Syntax:

GRANT privilege\_type ON securable\_object TO principal

**privilege\_type** is a Hive metastore privilege type.

**securable\_object** is a securable object in hive metastore.

principal is a user, service principal or group.

Example:

GRANT CREATE ON SCHEMA my\_schema TO `alf@melmak.et`;

GRANT ALL PRIVILEGES ON TABLE forecasts TO finance;

GRANT SELECT ON TABLE sample\_data TO `alf@melmak.et`;

Note: When table access control is enabled on a cluster or SQL warehouse, a user who creates a schema, table, view, or function becomes its owner. The owner is granted all privileges and can grant privileges to other users.

5.1 Privileges we can grant on Hive metastore objects:

SELECT: gives read access to an object.

CREATE: gives ability to create an object (for example, a table in a schema).

MODIFY: gives ability to add, delete, and modify data to or from an object.

USAGE: does not give any abilities but is an additional requirement to perform any action on a schema object.

READ\_METADATA: gives ability to view an object and its metadata.

CREATE\_NAMED\_FUNCTION: gives ability to create a named UDF in an existing catalog or schema.

MODIFY\_CLASSPATH: gives ability to add files to the Spark class path.

ALL PRIVILEGES: gives all privileges (is translated into all the above privileges).

6. Manage service principals:

Account admins can add service principals to your Azure Databricks account using the account console.

To use service principals on Azure Databricks, an admin user must create a new Azure Active Directory (Azure AD) application and then add it to the Azure Databricks workspace to use as a service principal. To create an Azure AD service principal below are the steps,

Step 1: Sign into the Azure portal.

Step 2: Search for and select **Azure Active Directory**.

Step 3: Click **+ Add** and select **App registration**.

Step 4: For **Name**, enter a name for the application.

Step 5: In the **Supported account types** section, select **Accounts in this organizational directory only (Single tenant)**.

Step 6: Click **Register**.

Step 7: Within **Manage**, click **Certificates & secrets.**

Step 8:On the **Client secrets** tab, click **New client secret**.

A screenshot of a computer

Description automatically generated

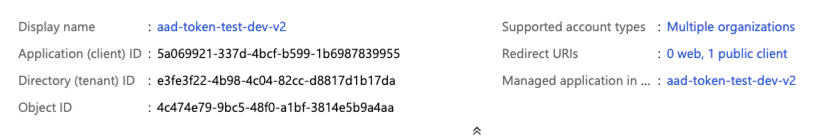
Step 9: For **Expires**, select an expiry time for the client secret, and then click **Add**.

Step 10: Copy and store the client secret’s **Value** in a secure place, as this client secret is the password for your application.

Step 11: On the application page’s Overview page, in the Essentials section, copy the following values:

Application (client) ID

Directory (tenant) ID



6.1. Add service principals to your account using the account console:

Step 1: As an account admin, log in to the account console.

Step 2: In the sidebar, click **User management**.

Step 3: On the **Service principals** tab, click **Add service principal**.

Step 4: Enter a name for the service principal.

Step 5: Under **UUID**, paste the Application (client) ID for the service principal.

Step 6: Click **Add**.

6.2. Assign account admin roles to a service principal:

Step 1: As an account admin, log in to the account console.

Step 2: In the sidebar, click **User management**.

Step 3: On the **Service principals** tab, find and click the username.

Step 4: On the **Roles** tab, turn on **Account admin** or **Marketplace admin**.

6.3. Assign a service principal to a workspace using the account console:

Step 1: As an account admin, log in to the account console.

Step 2: In the sidebar, click **Workspaces**.

Step 3: Click your workspace name.

Step 4: On the **Permissions** tab, click **Add permissions**.

Step 5: Search for and select the service principal, assign the permission level (workspace **User** or **Admin**), and click **Save**.

6.4. Deactivate a service principal in your Azure Databricks account:

Account admins can deactivate service principals across an Azure Databricks account. A deactivated service principal cannot authenticate to the Azure Databricks account or workspaces. However, all the service principal’s permissions and workspace objects remain unchanged. When a service principal is deactivated, the following is true:

The service principal cannot authenticate the account or any of their workspaces from any method.

Applications or scripts that use the tokens generated by the service principal are no longer able to access the Databricks API. The tokens remain but cannot be used to authenticate while a service principal is deactivated.

Clusters owned by the service principal remain running.

Scheduled jobs created by the service principal fail unless they are assigned to a new owner.

When a service principal is reactivated, they can login to Azure Databricks with the same permissions.

Manage service principals in your workspace

Workspace admins can manage service principals in their workspaces using the workspace admin settings page.

6.5. Add a service principal to a workspace using the workspace admin settings:

Step 1: As a workspace admin, log in to the Azure Databricks workspace.

Step 2: Click your username in the top bar of the Azure Databricks workspace and select Admin Settings.

Step 3: On the Service principals tab, click Add service principal.

Step 4: Select an existing service principal to assign to the workspace or add a new one.

To add a new service principal, click the drop-down arrow in the search box and then click + Add new service principal. Paste the Application (client) ID for the service principal and enter a display name.

6.6. Assign the workspace admin role to a service principal using the workspace admin settings page:

Step 1: As a workspace admin, log in to the Azure Databricks workspace.

Step 2: Click your username in the top bar of the Azure Databricks workspace and select Admin Settings.

Step 3: On the Groups tab, select the Admins group.

Step 4: Click Add users or service principals.

Step 5: Select the service principal and click Confirm.

To remove the workspace admin role from a service principal, remove the service principal from the admin group.

6.7. Manage workspace entitlements for a service principal:

An entitlement is a property that allows a user, service principal, or group to interact with Azure Databricks in a specified way. Entitlements are assigned to users at the workspace level. The following table lists entitlements and the workspace UI and API property name that you use to manage each one.

| Entitlement name (UI) | Entitlement name (API) | Default | Description |
| --- | --- | --- | --- |
| Workspace access | workspace-access | Granted by default. | When granted to a user or service principal, they can access the Data Science & Engineering and Databricks Machine Learning persona-based environments.  Can’t be removed from workspace admins. |
| Databricks SQL access | databricks-sql-access | Granted by default. | When granted to a user or service principal, they can access Databricks SQL. |
| Allow unrestricted cluster creation | allow-cluster-create | Not granted to users or service principals by default. | When granted to a user or service principal, they can create clusters. You can restrict access to existing clusters using  cluster level permissions.   Can’t be removed from workspace admins. |
| Allow pool creation (not available via UI) | allow-instance-pool-create | Can’t be granted to individual users or service principals. | When granted to a group, its members can create instance pools.  Can’t be removed from workspace admins. |

# Chapter: 7 Clusters

7.1. Introduction:

An Azure Databricks cluster is a set of computation resources and configurations on which you run data engineering, data science, and data analytics workloads, such as production ETL pipelines, streaming analytics, ad-hoc analytics, and machine learning.

Azure Databricks makes a distinction between **all-purpose clusters** and **job clusters**. We can use all-purpose clusters to analyse data collaboratively using interactive notebooks, while job clusters is use to run fast and robust automated jobs.

You can create an all-purpose cluster using the UI, CLI, or REST API. You can manually terminate and restart an all-purpose cluster. Multiple users can share such clusters to do collaborative interactive analysis.

The Azure Databricks job scheduler creates a job cluster when you run a job on a new job cluster and terminates the cluster when the job is complete. You cannot restart a job cluster.

7.2. Cluster Configuration:

The cluster creation UI lets us select the cluster configuration specifics, including:

The Policy.

The access mode, which controls the security features used when interacting with data.

The runtime version.

The cluster worker and driver node type.

7.2.1. Policies:

A cluster policy is a tool used to limit a user or group’s cluster permissions based on a set of policy rules.

Cluster policies will:

Limit users to creating clusters with prescribed settings.

Limit users to creating a certain number of clusters.

Simplify the user interface and enable more users to create their own clusters (by fixing and hiding some values).

Control cost by limiting per cluster maximum cost (by setting limits on attributes whose values contribute to hourly price).

Personal Compute policy:

Personal Compute is an Azure Databricks-managed cluster policy available, by default, on all Azure Databricks workspaces. Granting users access to this policy enables them to create single-machine compute resources in Azure Databricks for their individual use.

Admins can manage access and customize the policy rules to fit their workspace’s needs.

7.2.2. Databricks Runtime versions

Databricks Runtime is the set of core components that run on your clusters. Select the runtime using the **Databricks Runtime Version** dropdown when you create or edit a cluster.

Databricks Runtime versions are released on a regular basis:

**Major** versions are represented by an increment to the version number that precedes the decimal point (the jump from 3.5 to 4.0, for example). They are released when there are major changes, some of which may not be backwards-compatible.

**Feature** versions are represented by an increment to the version number that follows the decimal point (the jump from 3.4 to 3.5, for example). Each major release includes multiple feature releases. Feature releases are always backwards compatible with previous releases within their major release.

**Long Term Support** versions are represented by an **LTS** qualifier (for example, **3.5 LTS**). For each major release, we declare a “canonical” feature version, for which we provide three full years of support.

Which Databricks Runtime version should use:

For all-purpose computing, Databricks recommends using the latest Databricks Runtime version. Using the most current version will ensure you have the latest optimizations and most up-to-date compatibility between your code and preloaded packages.

For job clusters running operational workloads, consider using the Long-Term Support (LTS) Databricks Runtime version. Using the LTS version will ensure you don’t run into compatibility issues and can thoroughly test your workload before upgrading.

For advanced machine learning use cases, consider the specialized Databricks Runtime version.

All Databricks Runtime versions include Apache Spark. New versions add components and updates that improve usability, performance, and security.

7.2.3. Worker and driver node types:

A cluster consists of one driver node and zero or more worker nodes. You can pick separate cloud provider instance types for the driver and worker nodes.

Worker type:

Azure Databricks worker nodes run the Spark executors and other services required for proper functioning clusters. When you distribute your workload with Spark, all the distributed processing happens on worker nodes. Azure Databricks runs one executor per worker node. Therefore, the terms executor and worker are used interchangeably in the context of the Databricks architecture.

Worker node IP addresses:

Azure Databricks launches worker nodes with two private IP addresses each. The node’s primary private IP address hosts Azure Databricks internal traffic. The secondary private IP address is used by the Spark container for intra-cluster communication.

Driver type:

The driver node maintains state information of all notebooks attached to the cluster. The driver node also maintains the SparkContext, interprets all the commands you run from a notebook or a library on the cluster, and runs the Apache Spark master that coordinates with the Spark executors.

Enable Autoscaling:

When **enable autoscaling** is checked, you can provide a minimum and maximum number of workers for the cluster. Databricks then chooses the appropriate number of workers required to run your job.

To set the minimum and maximum number of workers your cluster will auto scale between, use the **Min workers** and **Max workers** fields next to the **Worker type** dropdown.

Benefits of autoscaling:

With autoscaling, Azure Databricks dynamically reallocates workers to account for the characteristics of your job. Certain parts of your pipeline may be more computationally demanding than others, and Databricks automatically adds additional workers during these phases of your job (and removes them when they’re no longer needed).

 Autoscaling thus offers two advantages:

Workloads can run faster compared to a constant-sized under-provisioned cluster.

Autoscaling clusters can reduce overall costs compared to a statically-sized cluster.

How autoscaling behaves:

Workspace in the Premium and Enterprise pricing plans use optimized autoscaling. Workspaces on the standard pricing plan use standard autoscaling.

Optimized autoscaling has the following characteristics:

Scales up from min to max in 2 steps.

Can scale down, even if the cluster is not idle, by looking at the shuffle file state.

Scales down based on a percentage of current nodes.

On job clusters, scales down if the cluster is underutilized over the last 40 seconds.

On all-purpose clusters, scales down if the cluster is underutilized over the last 150 seconds.

The spark.databricks.aggressiveWindowDownS Spark configuration property specifies in seconds how often a cluster makes down-scaling decisions. Increasing the value causes a cluster to scale down more slowly. The maximum value is 600.

Standard autoscaling is used in standard plan workspaces. Standard autoscaling has the following characteristics:

Starts with adding 8 nodes. Then scales up exponentially, taking as many steps required to reach the max.

Scales down when 90% of the nodes are not busy for 10 minutes and the cluster has been idle for at least 30 seconds.

Scales down exponentially, starting with 1 node.

7.2.4. Pools

To reduce cluster, start time, you can attach a cluster to a predefined pool of idle instances, for the driver and worker nodes. The cluster is created using instances in the pools. If a pool does not have sufficient idle resources to create the requested driver or worker nodes, the pool expands by allocating new instances from the instance provider. When an attached cluster is terminated, the instances it used are returned to the pools and can be reused by a different cluster.

Pool instance types:

A pool consists of both idle instances kept ready for new clusters and instances in use by running clusters. All of these instances are of the same instance provider type, selected when creating a pool.

Create a pool using the UI:

To create a pool using the UI:

Click **Compute** in the sidebar.

Click the **Pools** tab.

Click the **Create Pool** button.

Specify the pool configuration.

Click the **Create** button.

Attach a cluster to a pool:

To attach a cluster to a pool using the cluster creation UI, select the pool from the **Driver Type** or **Worker Type** dropdown while configuring the cluster. Available pools are listed at the top of each dropdown list. We can use the same pool or different pools for the driver node and worker nodes.

Pool size and auto termination:

When we create a pool, to control its size, we can set three parameters: minimum idle instances, maximum capacity, and idle instance auto termination.

1. Minimum Idle Instances

The minimum number of instances the pool keeps idle. These instances do not terminate, regardless of the auto termination settings. If a cluster consumes idle instances from the pool, Azure Databricks provisions additional instances to maintain the minimum.

2. Maximum Capacity

The maximum number of instances the pool can provision. If set, this value constrains all instances (idle + used). If a cluster using the pool requests more instances than this number during autoscaling, the request fails with an INSTANCE\_POOL\_MAX\_CAPACITY\_FAILURE error.

This configuration is optional. Azure Databricks recommend setting a value only in the following circumstances:

You have an instance quota you must stay under.

You want to protect one set of work from impacting another set of work. For example, suppose an instance quota is 100 and we have teams A and B that need to run jobs. We can create pool A with a max 50 and pool B with max 50 so that the two teams share the 100 quotas fairly.

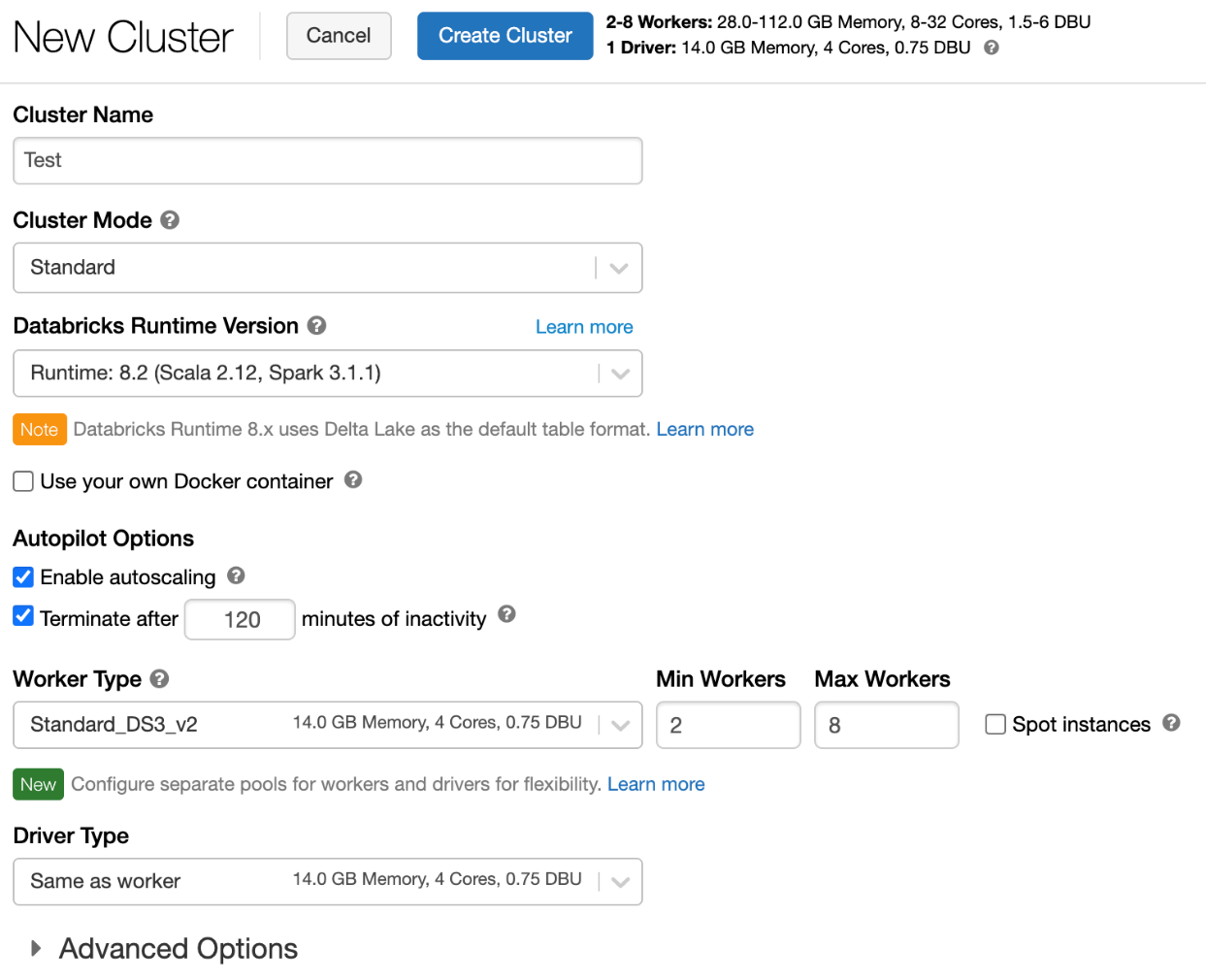
You need to cap cost.

Idle Instance Auto Termination:

The time in minutes above the value set in minimum idle instances that instances can be idle before being terminated by the pool.

7.3. Let’s create a Cluster:

Following image will shows the configuration options available when we create and edit Azure Databricks clusters:



1. Cluster policy:

To configure a cluster policy, select the cluster policy in the Policy drop-down.

A screenshot of a computer

Description automatically generated

2. Cluster mode:

Azure Databricks supports three cluster modes: Standard, High Concurrency, and Single node. The default cluster mode is Standard.



Note: We cannot change the cluster mode after a cluster is created. If you want a different cluster mode, you must create a new cluster.

a. High Concurrency clusters

A High Concurrency cluster is a managed cloud resource. The key benefits of High Concurrency clusters are that they provide fine-grained sharing for maximum resource utilization and minimum query latencies.

High Concurrency clusters can run workloads developed in SQL, Python, and R. The performance and security of High Concurrency clusters is provided by running user code in separate processes, which is not possible in Scala.

In addition, only High Concurrency clusters support table access control.

To create a High Concurrency cluster, set Cluster Mode to High Concurrency.

b. Single Node clusters

A Single Node cluster has no workers and runs Spark jobs on the driver node.

In contrast, a Standard cluster requires *at least one* Spark worker node in addition to the driver node to execute Spark jobs.

To create a Single Node cluster, set Cluster Mode to Single Node.

A blue line on a white background

Description automatically generated

3. Auto terminate:

The cluster configuration includes an auto terminate setting whose default value depends on cluster mode:

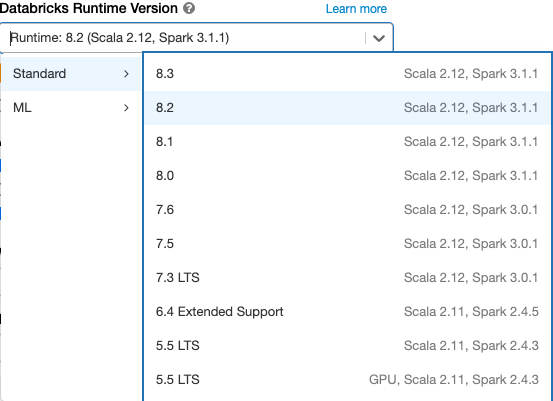
Standard and Single Node clusters terminate automatically after 120 minutes by default.

High Concurrency clusters do not terminate automatically by default.

A Standard cluster is recommended for single users only. Standard clusters can run workloads developed in Python, SQL, R, and Scala.

4. Databricks Runtime

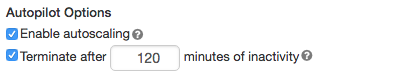
Databricks runtimes are the set of core components that run on your cluster. Azure Databricks offers several types of runtimes and several versions of those runtime types in the **Databricks Runtime Version** drop-down when you create or edit a cluster.



5. Enable and configure autoscaling:

To allow Azure Databricks to resize your cluster automatically, you enable autoscaling for the cluster and provide the min and max range of workers.

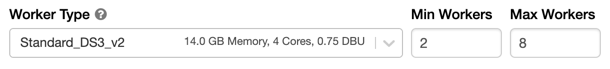
For All-Purpose cluster - On the Create Cluster page, select the **Enable autoscaling** checkbox in the **Autopilot Options** box.



For Job cluster - On the Configure Cluster page, select the **Enable autoscaling** checkbox in the **Autopilot Options** box:



Configure the min and max workers:



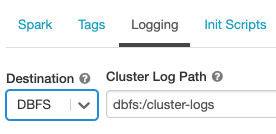
Cluster log delivery:

When you create a cluster, you can specify a location to deliver the logs for the Spark driver node, worker nodes, and events. Logs are delivered every five minutes to your chosen destination. When a cluster is terminated, Azure Databricks guarantees to deliver all logs generated up until the cluster was terminated.

To configure the log delivery location:

On the cluster configuration page, click the **Advanced Options** toggle.

Click the **Logging** tab.



Select a destination type.

Enter the cluster log path.