Microsoft

Module: Databricks Development

Microsoft Services

Module Overview

- Lesson 1: Azure Databricks Notebooks and Jobs
- Lesson 2: Working with Storage Options

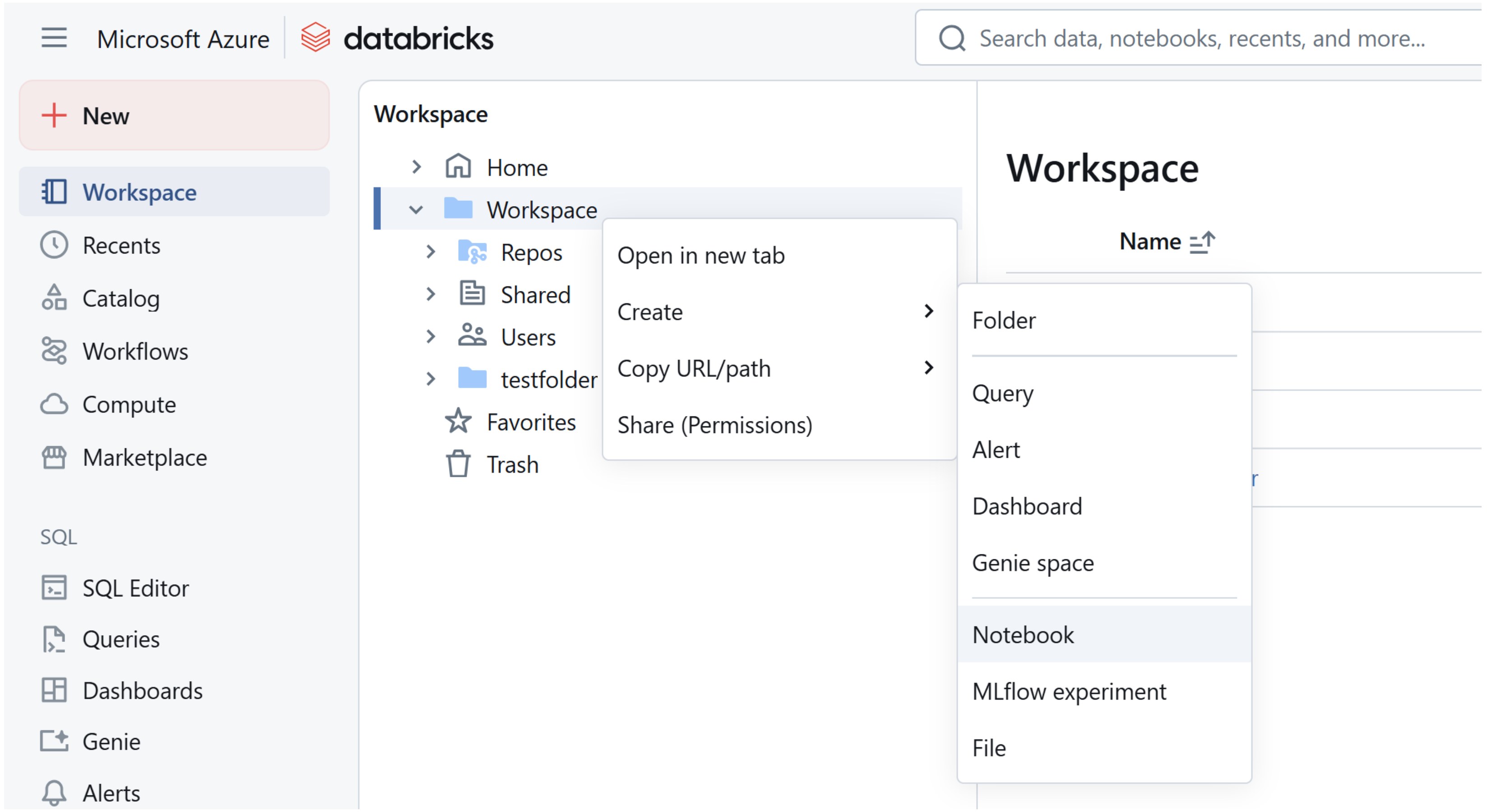
Lesson 1: Azure Databricks Notebooks and Jobs

After completing this lesson, you will be able to:

- Understand usage of notebooks
- Manage notebooks (create, delete, export, import, attach/detach)
- Use notebooks to run commands, create dashboards
- Create and schedule jobs

Databricks Notebooks

- Is an interface for interacting with Azure Databricks
- Is a web-based interface in Azure Databricks which can contain:
    - Code
    - Visualizations
    - Narrative text



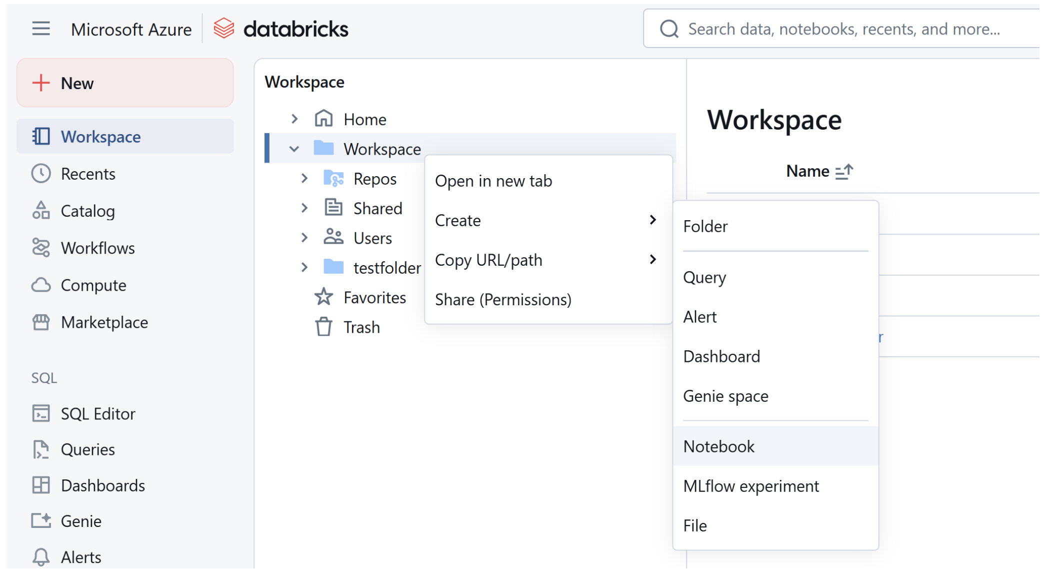
Managing Notebooks

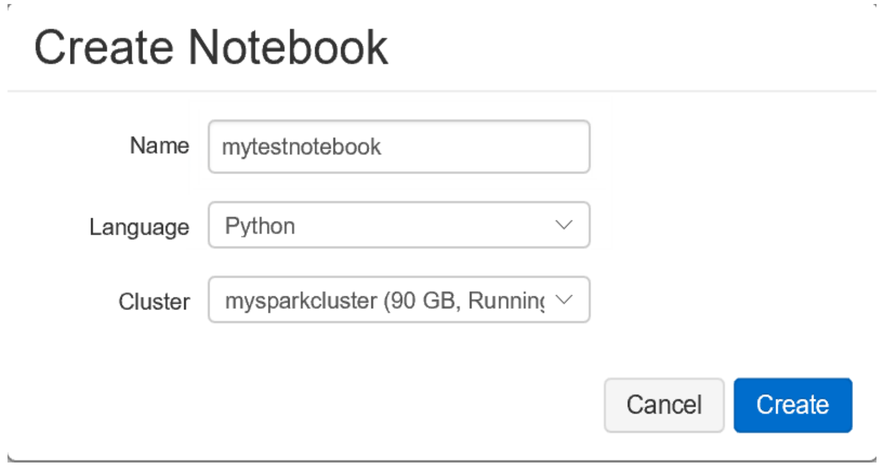
- Notebooks can be managed using:
  - UI
  - CLI
  - Workspace API

- In this lesson, we will focus on UI

Create a notebook

- Click the Workspace button in the sidebar.
- Next to any folder, click the Menu Dropdown on the right side of the text and select Create > Notebook.
- In the Create Notebook dialog, enter a name and select the notebook's primary language.
- If there are running clusters, the Cluster drop-down will display them. Select the cluster to attach the notebook to.
- Click Create




Create Notebook

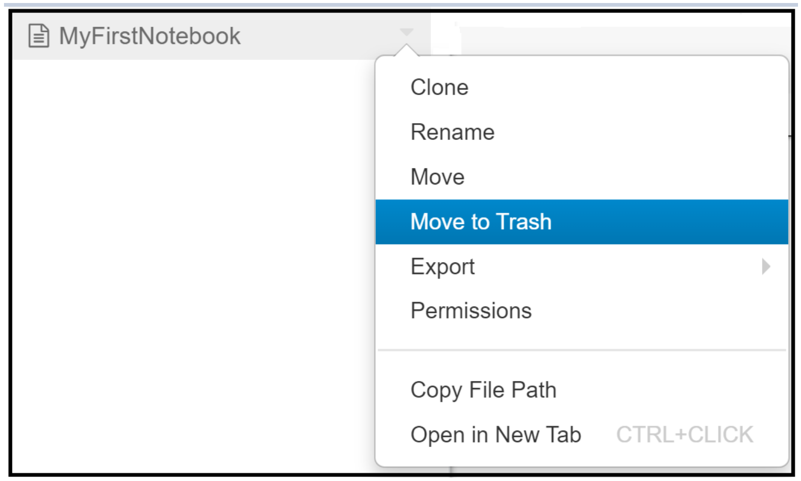
Name: mytestnotebook

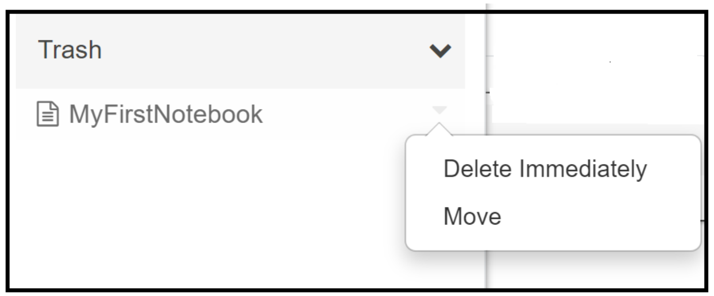
Language: Python

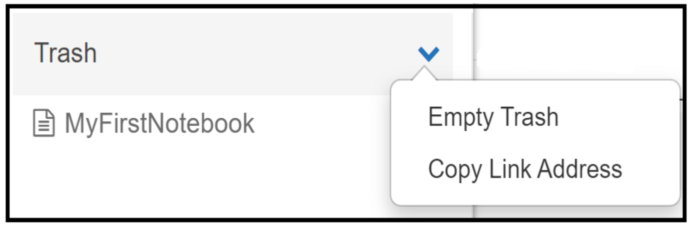
Cluster: mysparkcluster (90 GB, Running

Delete a notebook

- Click the Menu Dropdown at the right side of the notebook and select Move to Trash
- The deleted object will be moved to the user's Trash folder.
- Trash folder is automatically purged after 30 days.

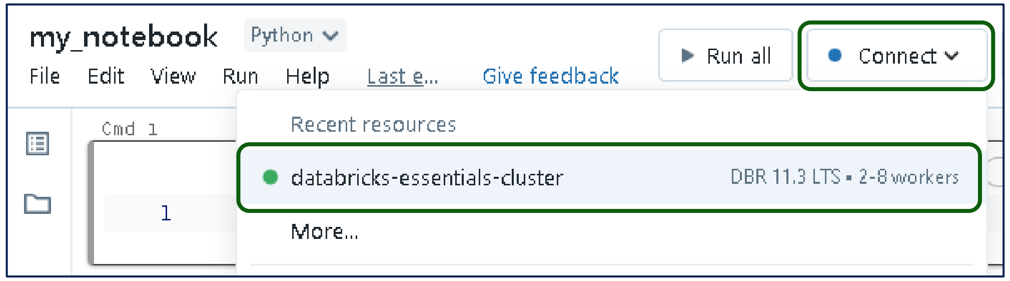


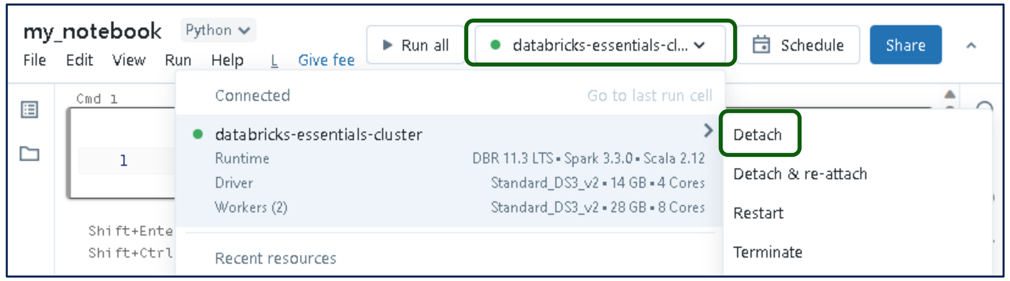




Notebooks and clusters

- Before you can do any work in a notebook, you must first attach the notebook to a cluster.
- Attaching to a cluster, creates an execution context.
- To attach a notebook to a cluster
  - In the notebook toolbar, click the **Connect** drop-down.
  - From it, select a cluster.
- Detach a notebook from a cluster
  - In the notebook toolbar, click on the cluster drop-down.
  - Select Detach.





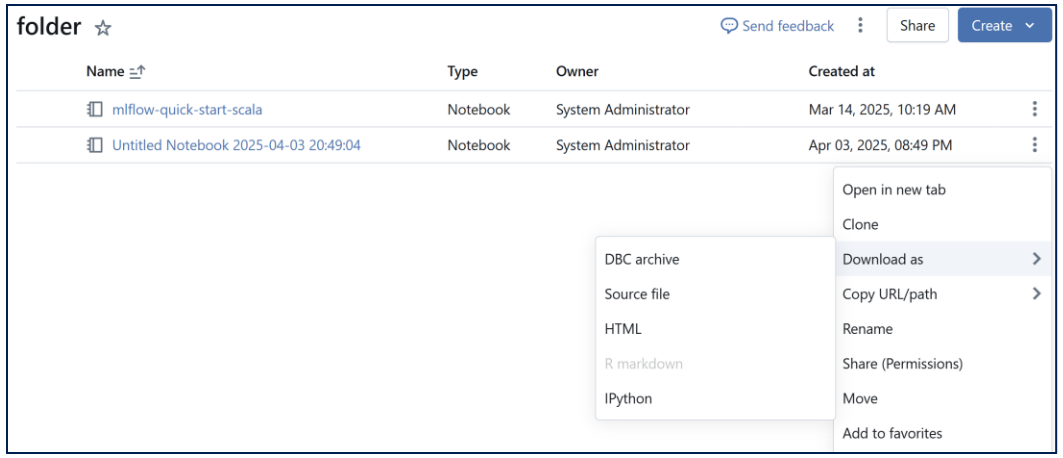
Notebook external formats

Azure Databricks supports several notebook external formats:

- Source File
  - A source file with the extension .scala, .py, .sql, or .r.
- HTML
  - An Azure Databricks notebook with a .html extension.
- DBC Archive
  - A Databricks archive with a .dbc extension.
- IPython Notebook
  - A Jupyter notebook with a .ipynb extension.
- RMarkdown
  - An R Markdown document with a .Rmd extension.

Export/Import Notebook

- Export: In the notebook toolbar, select File -> Export and a format.
- Import: Click the Workspace button or the Home button in the sidebar.
  - In the Workspace or a user folder, click Down Caret and select Import.
  - Specify the URL or browse to a file containing a supported external format.
  - Click Import.

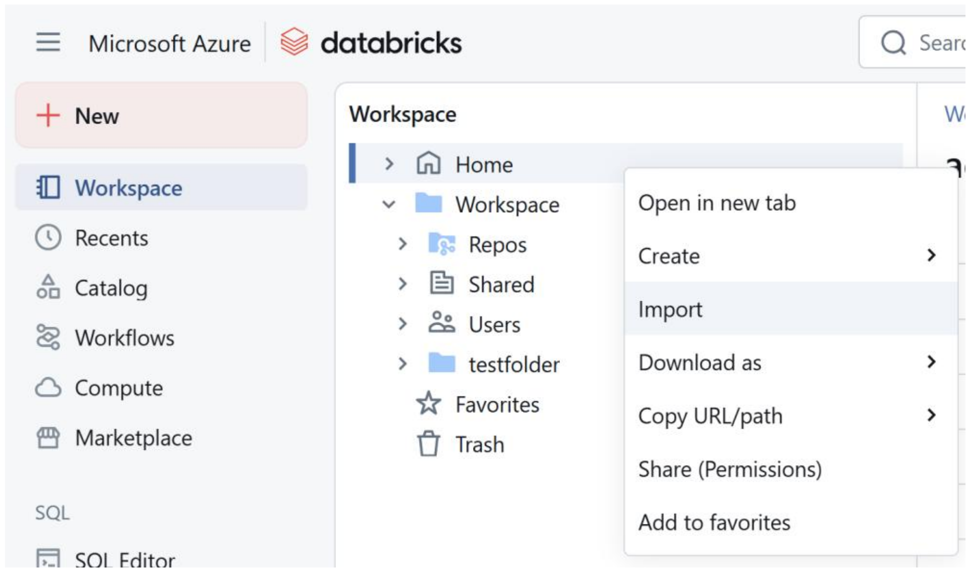


folder ☆ Send feedback Share Create

Name	Type	Owner	Created at
mlflow-quick-start-scala	Notebook	System Administrator	Mar 14, 2025, 10:19 AM
Untitled Notebook 2025-04-03 20:49:04	Notebook	System Administrator	Apr 03, 2025, 08:49 PM

Open in new tab
Clone
Download as >
Copy URL/path >
Rename
Share (Permissions)
Move
Add to favorites

DBC archive
Source file
HTML
R markdown
IPython



Microsoft Azure databricks Searc

New
Workspace
Recents
Catalog
Workflows
Compute
Marketplace
SQL
SQL Editor

Workspace
Home
Workspace
Repos
Shared
Users
testfolder
Favorites
Trash

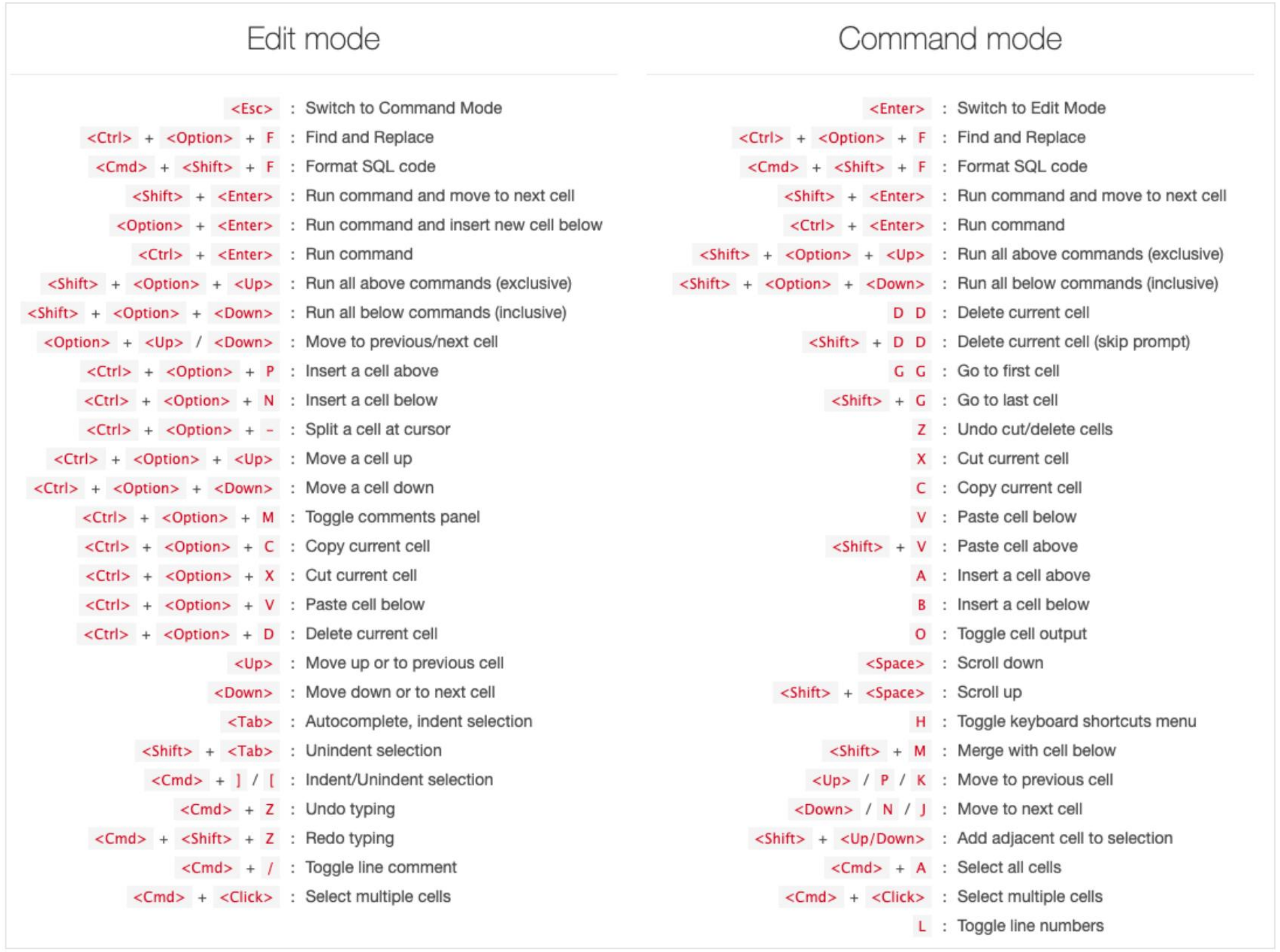
Open in new tab
Create >
Import
Download as >
Copy URL/path >
Share (Permissions)
Add to favorites

Demo: Working with Notebooks

How to create/delete, attach/detach, export/import Notebooks in Azure Databricks

Using notebooks

- A notebook is a collection of runnable cells
- Tasks within notebooks can be performed using UI or keyboard shortcuts



Edit mode

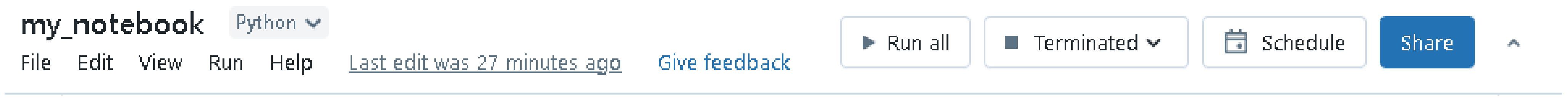
<Esc>	Switch to Command Mode
<Ctrl> + <Option> + F	Find and Replace
<Cmd> + <Shift> + F	Format SQL code
<Shift> + <Enter>	Run command and move to next cell
<Option> + <Enter>	Run command and insert new cell below
<Ctrl> + <Enter>	Run command
<Shift> + <Option> + <Up>	Run all above commands (exclusive)
<Shift> + <Option> + <Down>	Run all below commands (inclusive)
<Option> + <Up> / <Down>	Move to previous/next cell
<Ctrl> + <Option> + P	Insert a cell above
<Ctrl> + <Option> + N	Insert a cell below
<Ctrl> + <Option> + -	Split a cell at cursor
<Ctrl> + <Option> + <Up>	Move a cell up
<Ctrl> + <Option> + <Down>	Move a cell down
<Ctrl> + <Option> + M	Toggle comments panel
<Ctrl> + <Option> + C	Copy current cell
<Ctrl> + <Option> + X	Cut current cell
<Ctrl> + <Option> + V	Paste cell below
<Ctrl> + <Option> + D	Delete current cell
<Up>	Move up or to previous cell
<Down>	Move down or to next cell
<Tab>	Autocomplete, indent selection
<Shift> + <Tab>	Unindent selection
<Cmd> +] / [	Indent/Unindent selection
<Cmd> + Z	Undo typing
<Cmd> + <Shift> + Z	Redo typing
<Cmd> + /	Toggle line comment
<Cmd> + <Click>	Select multiple cells

Command mode

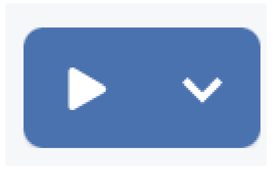
<Enter>	Switch to Edit Mode
<Ctrl> + <Option> + F	Find and Replace
<Cmd> + <Shift> + F	Format SQL code
<Shift> + <Enter>	Run command and move to next cell
<Ctrl> + <Enter>	Run command
<Shift> + <Option> + <Up>	Run all above commands (exclusive)
<Shift> + <Option> + <Down>	Run all below commands (inclusive)
D D	Delete current cell
<Shift> + D D	Delete current cell (skip prompt)
G G	Go to first cell
<Shift> + G	Go to last cell
Z	Undo cut/delete cells
X	Cut current cell
C	Copy current cell
V	Paste cell below
<Shift> + V	Paste cell above
A	Insert a cell above
B	Insert a cell below
O	Toggle cell output
<Space>	Scroll down
<Shift> + <Space>	Scroll up
H	Toggle keyboard shortcuts menu
<Shift> + M	Merge with cell below
<Up> / P / K	Move to previous cell
<Down> / N / J	Move to next cell
<Shift> + <Up/Down>	Add adjacent cell to selection
<Cmd> + A	Select all cells
<Cmd> + <Click>	Select multiple cells
L	Toggle line numbers

Using notebooks

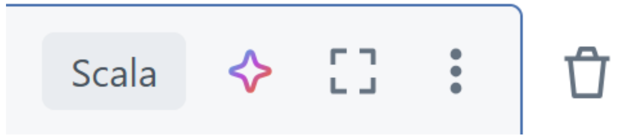
- A notebook has a toolbar that lets you manage the notebook and perform actions within the notebook.



- At upper left corner of each cell, it has
  - Run Cell



- At upper right corner of each cell, it has
  - Cell Language, Minimize/Maximize and Delete



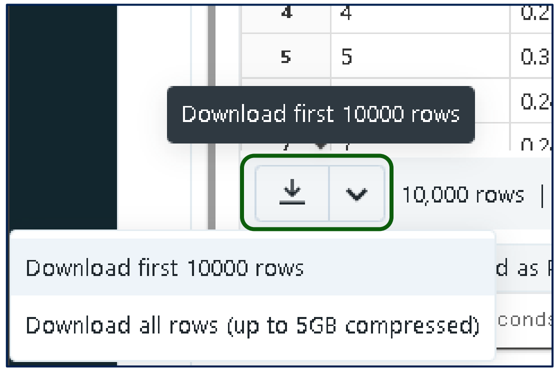
- Actions such as:
  - Adding, deleting cells, mixing languages, documentation, links to notebooks etc etc can be done within cells

Run notebooks

- To run a cell: click the “Run cell” or ctrl + Enter or shift + Enter.
- To run only the selected text in a cell Shift + Ctrl + Enter
- Other options available to run are: “Run all above”, “Run all below”, “Run all”.
- Python and Scala notebooks support error highlighting.
    - The line of code that is throwing the error will be highlighted in the cell
- Notifications alert you to events such as which command is currently running during Run all cells and which commands are in error state.
- You can run a notebook from another notebook by using the %run <notebook> magic command.

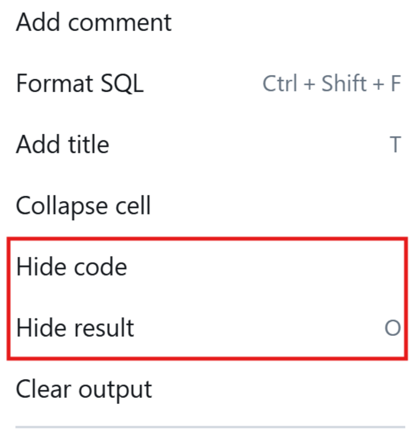
Manage notebook state and results

- To clear the notebook state and results, click **Clear** in the notebook toolbar and select the desired clearing action
- To download a cell result that contains tabular output to your local machine, click the **Download Result** button at the bottom of a cell.



Download first 10000 rows
10,000 rows
Download first 10000 rows
Download all rows (up to 5GB compressed)

- You can **hide** and **show** the cell code and result using the cell actions menu Cell Actions at the top right of the cell.



Dashboards

- Dashboards allow you to publish graphs and visualizations and share them in a presentation format with your organization.
- The elements of a dashboard are output from notebook cells.
- Create a notebook with a combination of visualizations and codes.
- Create a dashboard that displays the notebook output.
- Rearrange and reshape each cell as you see fit.
- Present the dashboard.

Dashboards

- Dashboards can be edited from the dashboard view
- Dashboards do not live refresh when you present them from the dashboard view
- To schedule a dashboard to refresh at a specified interval, schedule the notebook that generates the dashboard graphs

Demo: Working with notebooks

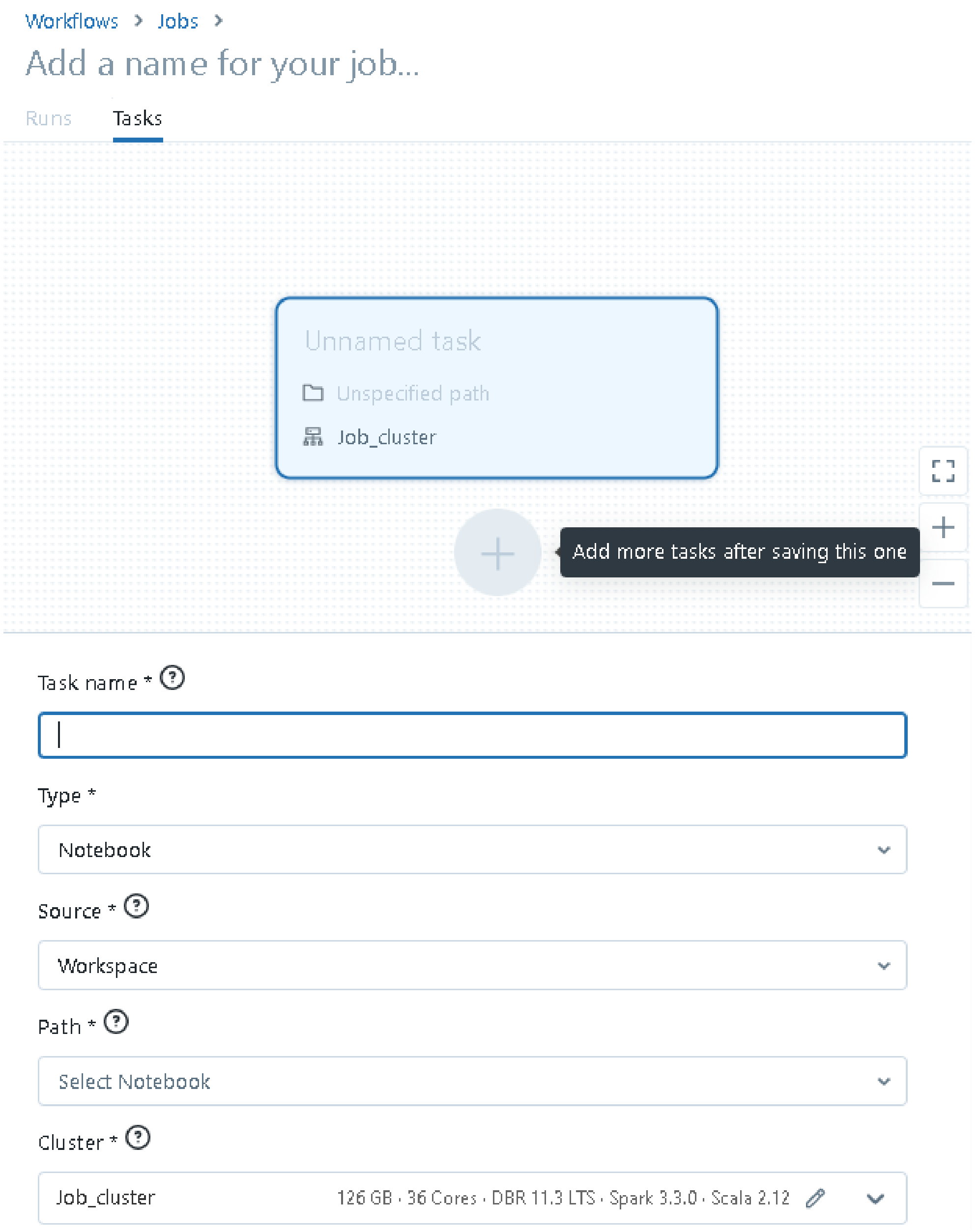
Creating dashboards

Jobs

- A job is a way of running a notebook or JAR either immediately or on a scheduled basis.
- Jobs can be created and run by using the UI, the CLI, and by invoking the Jobs API.
- A job can consist of a single task or can be multi-task workflow with complex dependencies
- Jobs can be monitored by using UI, CLI, API, and through email alerts.

Create a Job

- Click the **Workflows** button in the sidebar
- Click **Create Job**. The task detail page displays.
- Provide a job name
- Provide task name properties by selecting notebook, Set JAR, or Configure spark-submit.
- Select the cluster to run the task against.
- Set any additional task parameters
- Set advanced parameters as required.



Task name *

Type *
Notebook

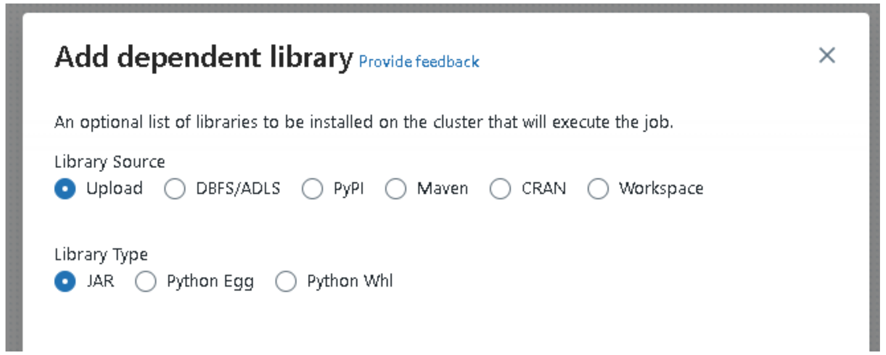
Source *
Workspace

Path *
Select Notebook

Cluster *
Job_cluster 126 GB · 36 Cores · DBR 11.3 LTS · Spark 3.3.0 · Scala 2.12

Task Advanced Options 1/2

- Add dependent libraries
  - A list of libraries to be installed on the cluster that will execute the job.



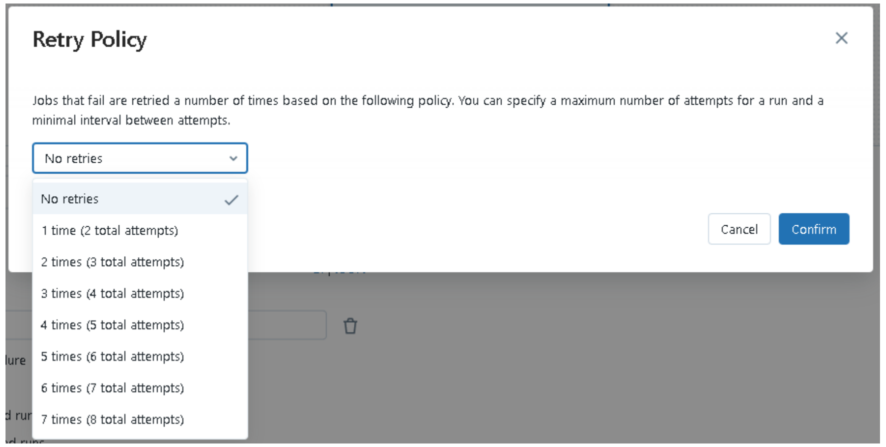
Add dependent library Provide feedback

An optional list of libraries to be installed on the cluster that will execute the job.

Library Source
[✓ Upload] DBFS/ADLS PyPI Maven CRAN Workspace

Library Type
[✓ JAR] Python Egg Python Whl

- Retries:
  - Policy that determines when and how many times failed runs are retried



Retry Policy

Jobs that fail are retried a number of times based on the following policy. You can specify a maximum number of attempts for a run and a minimal interval between attempts.

No retries

- [✓ No retries]
- 1 time (2 total attempts)
- 2 times (3 total attempts)
- 3 times (4 total attempts)
- 4 times (5 total attempts)
- 5 times (6 total attempts)
- 6 times (7 total attempts)
- 7 times (8 total attempts)

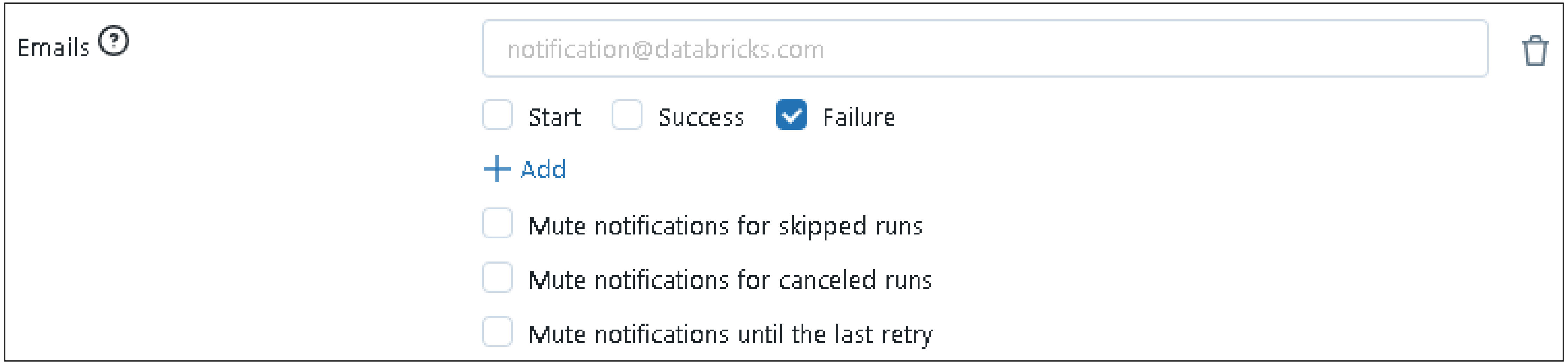
Cancel Confirm

- Timeout:
  - Maximum completion time for a task

Timeout in seconds ?
No timeout

Task Advanced Options 2/2

- Email
  - Email alerts sent in case of job failure or success



Emails

notification@databricks.com

[ ] Start [ ] Success [✓] Failure

+ Add

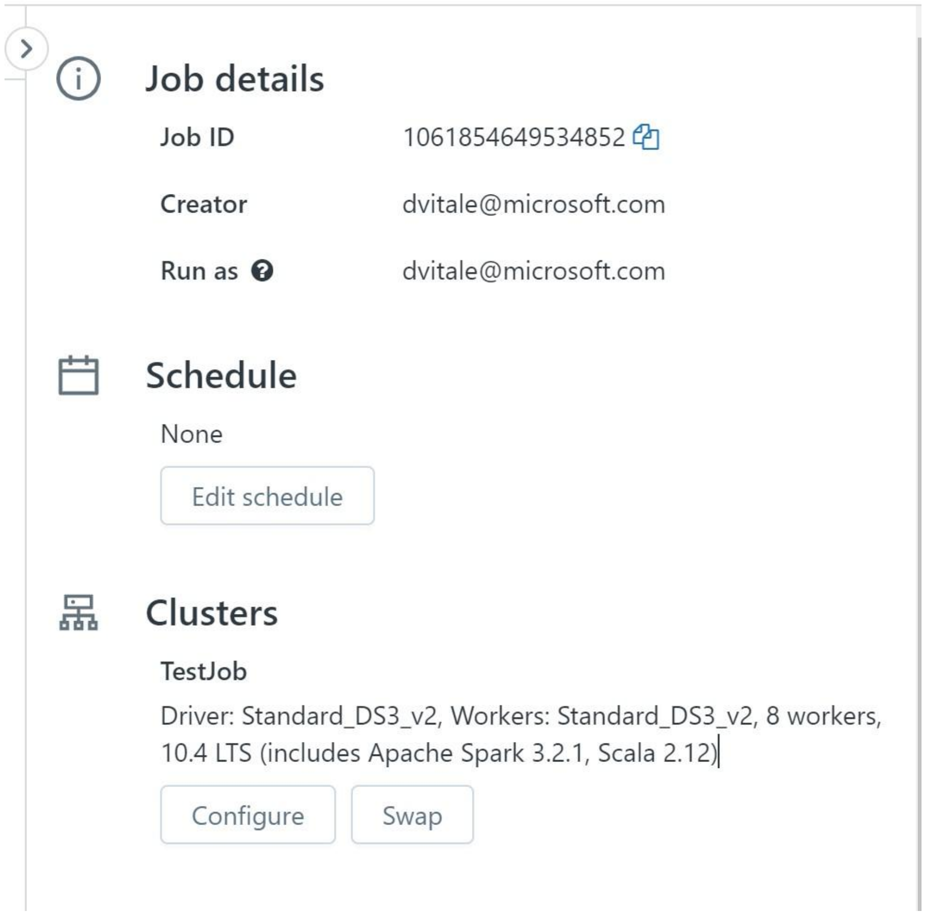
[ ] Mute notifications for skipped runs

[ ] Mute notifications for canceled runs

[ ] Mute notifications until the last retry

Job Schedule

- After the creation of the first task you will be able to set your job schedule by clicking on Edit Schedule



Job details

Job ID	1061854649534852
Creator	dvitale@microsoft.com
Run as	dvitale@microsoft.com

Schedule

None

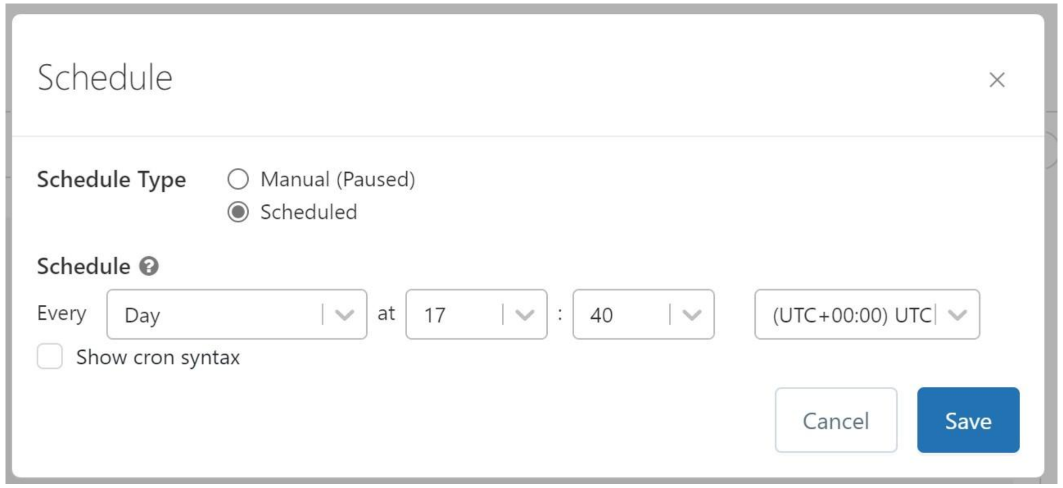
Edit schedule

Clusters

TestJob

Driver: Standard_DS3_v2, Workers: Standard_DS3_v2, 8 workers, 10.4 LTS (includes Apache Spark 3.2.1, Scala 2.12)

Configure Swap



Schedule

Schedule Type: Manual (Paused) [✓ Scheduled]

Schedule

Every Day at 17 : 40 (UTC+00:00) UTC

☐ Show cron syntax

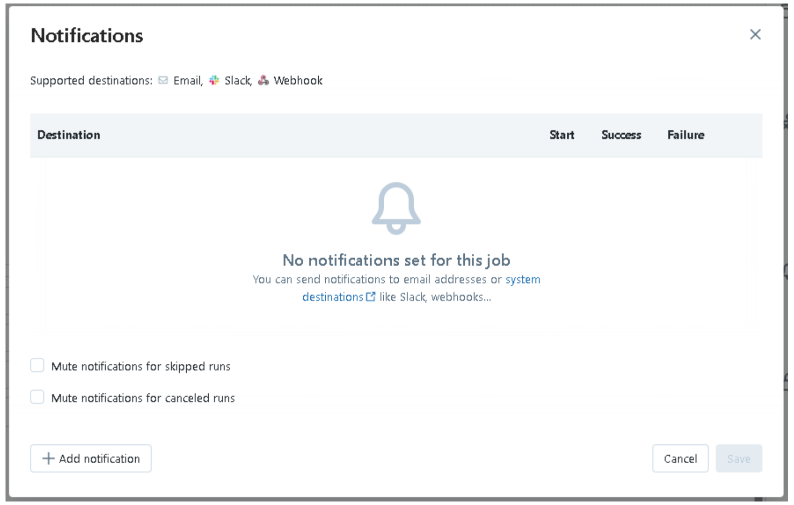
Cancel Save

Job Advanced Options

- Notifications
  - Events sent in case of job failure, success, or timeout. Notifications can be sent through email, slack or a generic webhook

- Maximum concurrent runs
  - The maximum number of runs that can be run in parallel

- Permissions
  - Job access control enable job owners and administrators to grant fine grained permissions on their jobs.



Notifications

Supported destinations: Email, Slack, Webhook

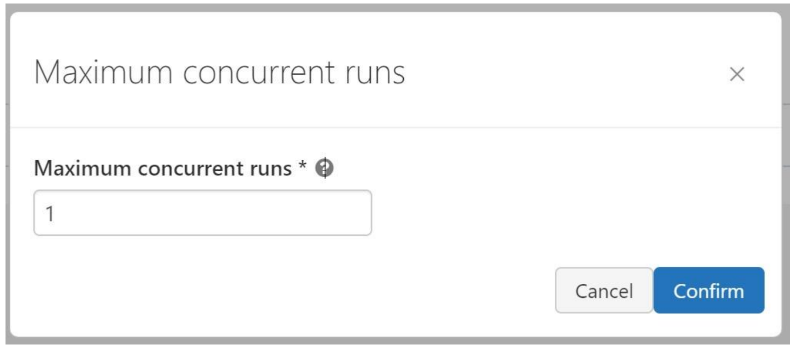
Destination	Start	Success	Failure

No notifications set for this job

You can send notifications to email addresses or system destinations like Slack, webhooks...

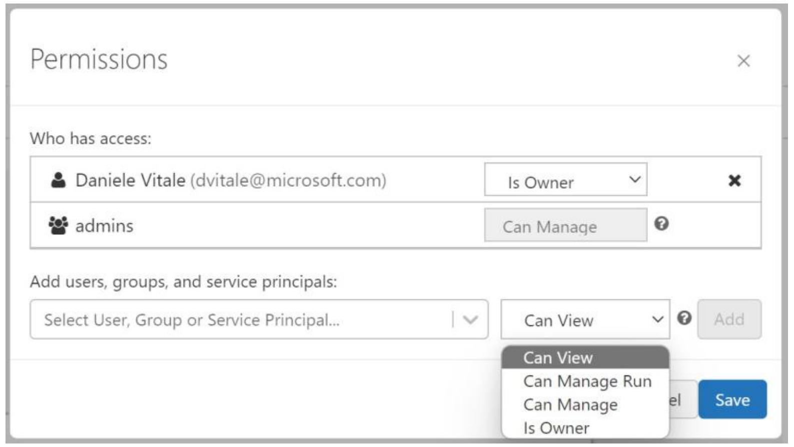
Mute notifications for skipped runs

Mute notifications for canceled runs



Maximum concurrent runs

Maximum concurrent runs *: 1



Permissions

Who has access:

Daniele Vitale (dvitale@microsoft.com)	Is Owner
admins	Can Manage

Add users, groups, and service principals:

Select User, Group or Service Principal... — Can View

Demo: Working with Jobs

Create, schedule, run jobs

Lab: Notebooks and Jobs

Exercise 1: Create/Delete, Export/Import, Attach/Detach notebooks
Exercise 2: Create dashboards using notebooks
Exercise 3: Create jobs to run notebooks on scheduled basis

Knowledge Check

- Databricks Notebooks can be managed using?
- What is the magic command to run a notebook from another notebook?
- How can we run a notebook or JAR either immediately or on a scheduled basis?

Lesson 2: Working with Storage Options

- After completing this lesson, you will be able to:
  - Working with the Databricks File System
  - Mounting Storage Options in your Azure Databricks Workspace

What is DBFS?

- Databricks File System(DBFS) is a distributed file system mounted into an Azure Databricks workspace and available on Azure Databricks clusters. DBFS is an abstraction on top of scalable object storage.
- DBFS uses Azure Blob Storage on the backend to persist the data
- You can store your tables, data files, or logs in DBFS and access it via tools like the Databricks CLI, DBFS API, dbutils, Spark APIs, and even local file APIs

What is DBFS?

- After a cluster is terminated the storage persists in the Azure Blob Storage instance that was created alongside the cluster
  - This allows users to be sure that data that has been modified or created during their sessions will still be there when they restart the cluster
- In addition to just the storage available in your cluster, you can mount other Azure storage options
  - Azure Blob Storage
  - Azure Data Lake Store Gen2
- We will cover mounting in an upcoming presentation.

Databricks Utilities and DBFS

- When you are developing in a Databricks notebook there are various ways of accessing, searching, and performing operations with DBFS
- One common way is to use the Databricks Utilities or **dbutils.fs** functions
- This family of functions allow users to perform shell-like operations with DBFS

Dbutils.fs Functions

Function	Description of Function
cp	Copies a file or directory
head	Returns the first few rows of a dataset
ls	Lists the contents of a directory
mkdir	Creates a given directory, if it does not exist already
mv	Moves a file or directory
put	Writes a given string to a file, encoded in UTF-8
rm	Removes a file or directory
mount	Mounts the given source directory into DBFS at the given mount point
refreshMounts	Forces all machines in the cluster to refresh their mount cache, ensuring they receive the most recent information
unmount	Deletes a DBFS mount point

Demo:

Working with DBFS

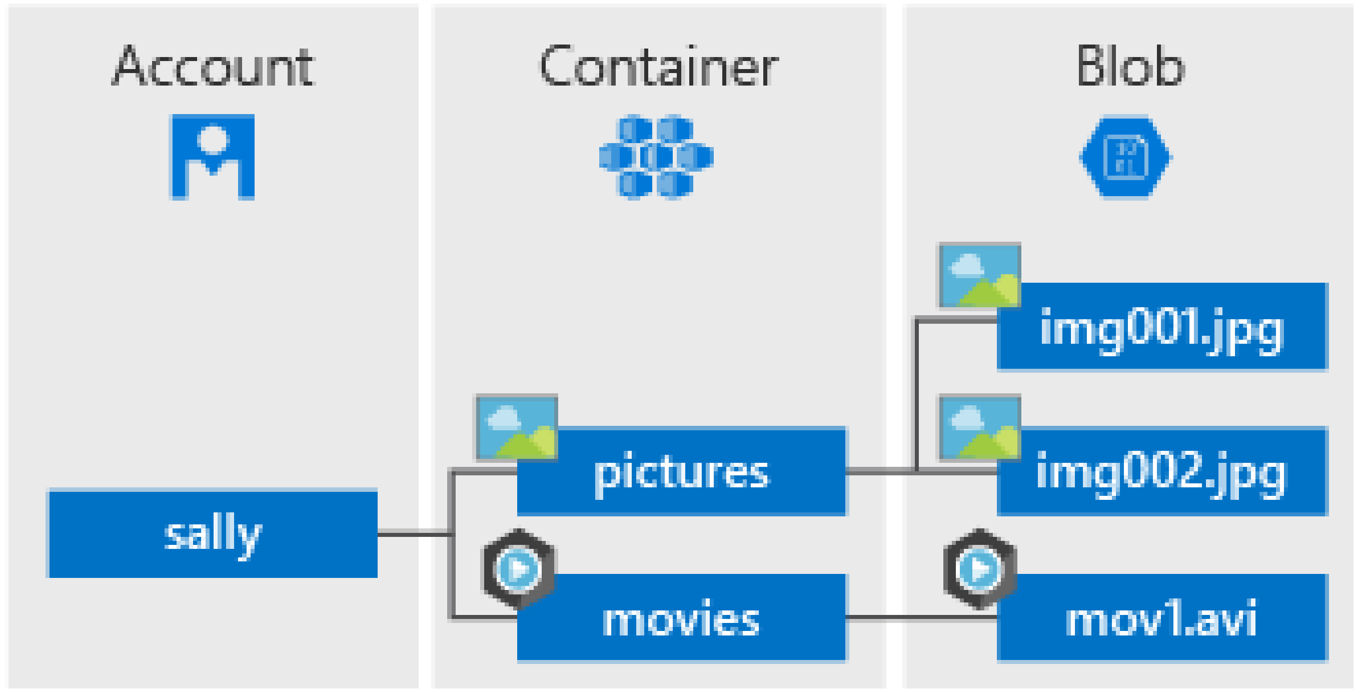
How to work with DBFS

Mounting Storage

- To access Azure data storage services like Azure Data Lake Store or Azure Blob Storage you can mount them in the Databricks File System (DBFS)
- Mounting storage creates a mount point that you, from then on, start at to access the underlying file structure of the data storage service that you are mounting
- Let's look at two popular mountable storage options:
  - Azure Blob Storage,
  - Azure Data Lake Storage (gen2).

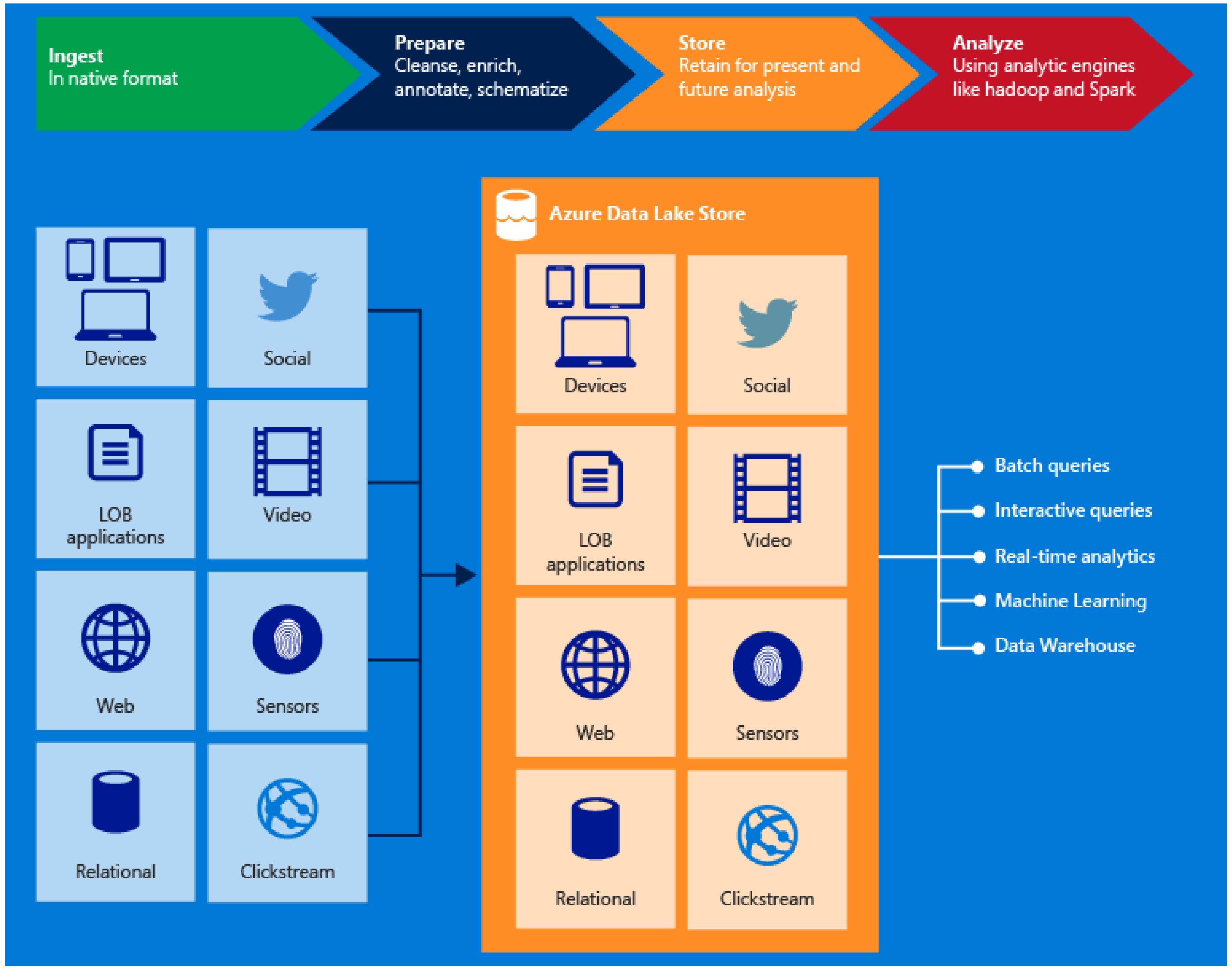
Azure Blob Storage

- Azure Blob Storage
  - Azure’s object storage solution for the cloud
  - Azure Blob Storage can store massive amounts of unstructured data, images, and many more



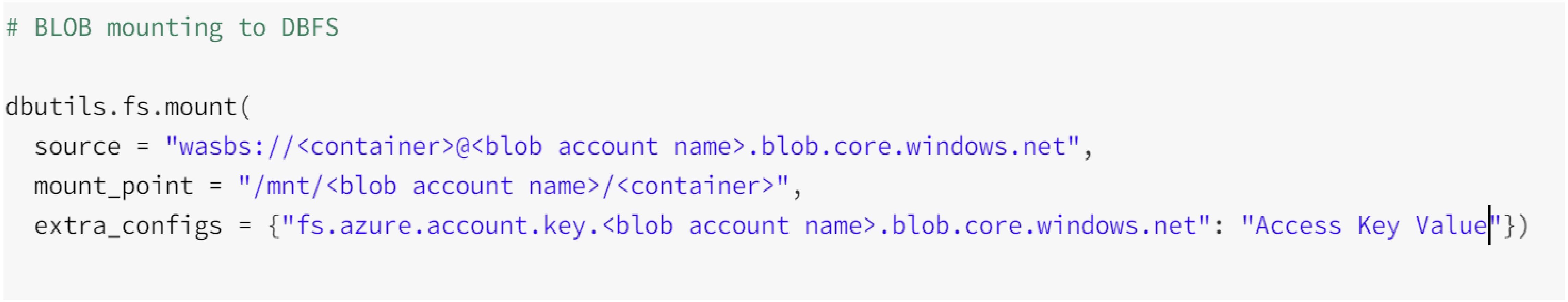
Azure Data Lake Store

- Azure Data Lake Storage (ADLS)
  - ADLS is the storage service for Azure Data Lake
  - Azure Data Lake Storage enables you to capture data of any size, type, and ingestion speed in one single place for operational and exploratory analytics
  - Customers can even use the WebHDFS-compatible REST APIs to access their data



Mounting Blob Storage

- This is a mounting configuration for Azure Blob Storage
  - source: the WASB path (container & blob account name) that can be found in the Azure Portal.
  - mount_point: this is the proposed file path that will be created in DBFS to access the data service.
  - extra_configs: these are additional configurations like access keys or credentials (SAS token) to allow access to the data service.



```
# BLOB mounting to DBFS

dbutils.fs.mount(
  source = "wasbs://<container>@<blob account name>.blob.core.windows.net",
  mount_point = "/mnt/<blob account name>/<container>",
  extra_configs = {"fs.azure.account.key.<blob account name>.blob.core.windows.net": "Access Key Value"})
```

Demo:

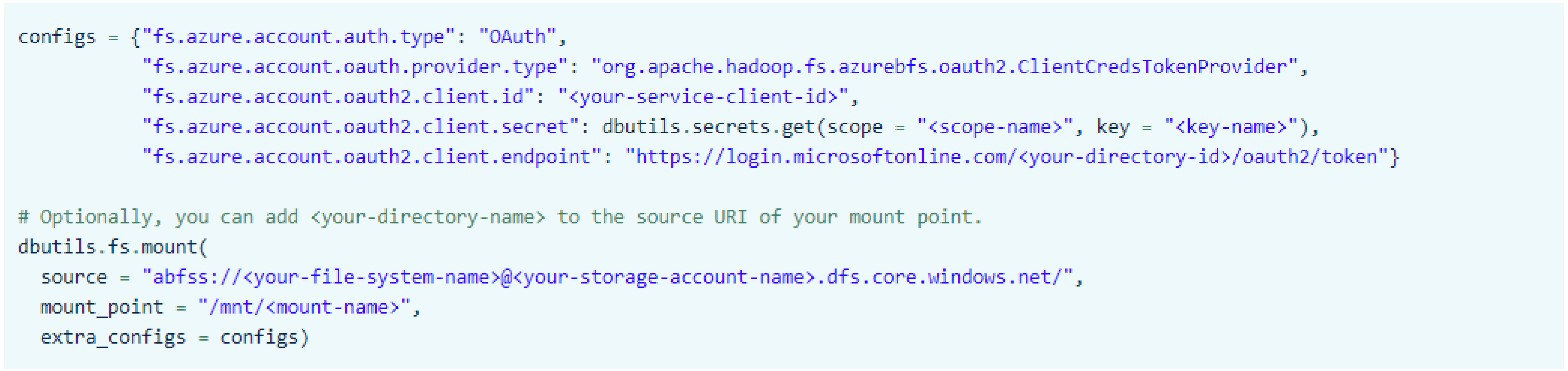
Mount Blob Storage and Azure Data Lake Gen2

How to mount blob storage and azure data lake gen2

Secrets API

- The Secrets API can be used like a normal REST endpoint by using means like the Python Requests library or **curl**
- However, you can also use the Databricks CLI to create and manage secrets
- The Secrets API encrypts your value and allows you to access it securely via the **dbutils.secrets.get** function
- You can use this function in your code to have Databricks go and retrieve the necessary, obfuscated values that are necessary for establishing a mounted storage container in your Databricks workspace

Mounting Data Lake Store gen2 with Secrets



```
configs = {"fs.azure.account.auth.type": "OAuth",
           "fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
           "fs.azure.account.oauth2.client.id": "<your-service-client-id>",
           "fs.azure.account.oauth2.client.secret": dbutils.secrets.get(scope = "<scope-name>", key = "<key-name>"),
           "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/<your-directory-id>/oauth2/token"}

# Optionally, you can add <your-directory-name> to the source URI of your mount point.
dbutils.fs.mount(
  source = "abfss://<your-file-system-name>@<your-storage-account-name>.dfs.core.windows.net/",
  mount_point = "/mnt/<mount-name>",
  extra_configs = configs)
```

- This is a mounting configuration for Azure Data Lake Store
  - service client ID: Application ID found on the information blade of the App within App Registrations in AAD.
  - dbutils.secrets.get(scope = "<scope-name>", key = "<key-name>") retrieves your service credential that has been stored as a secret in a secret scope.
  - directory ID: found under the AAD -> Properties –> Directory ID (bottom part of the blade)

Azure AD Credential Passthrough

- Authenticate automatically to ADLS Gen1 and ADLS Gen2 from Azure Databricks clusters using the same Azure Active Directory (Azure AD) identity used to login in databricks
- You can activate it during cluster creation in the advanced options
- Allow a direct access to Data Lake Storage
- Allow mounting Data Lake Storage to DBFS
- Available only on premium

Demo:

Mount Blob Storage with the Secrets API

How to mount blob storage with secrets API

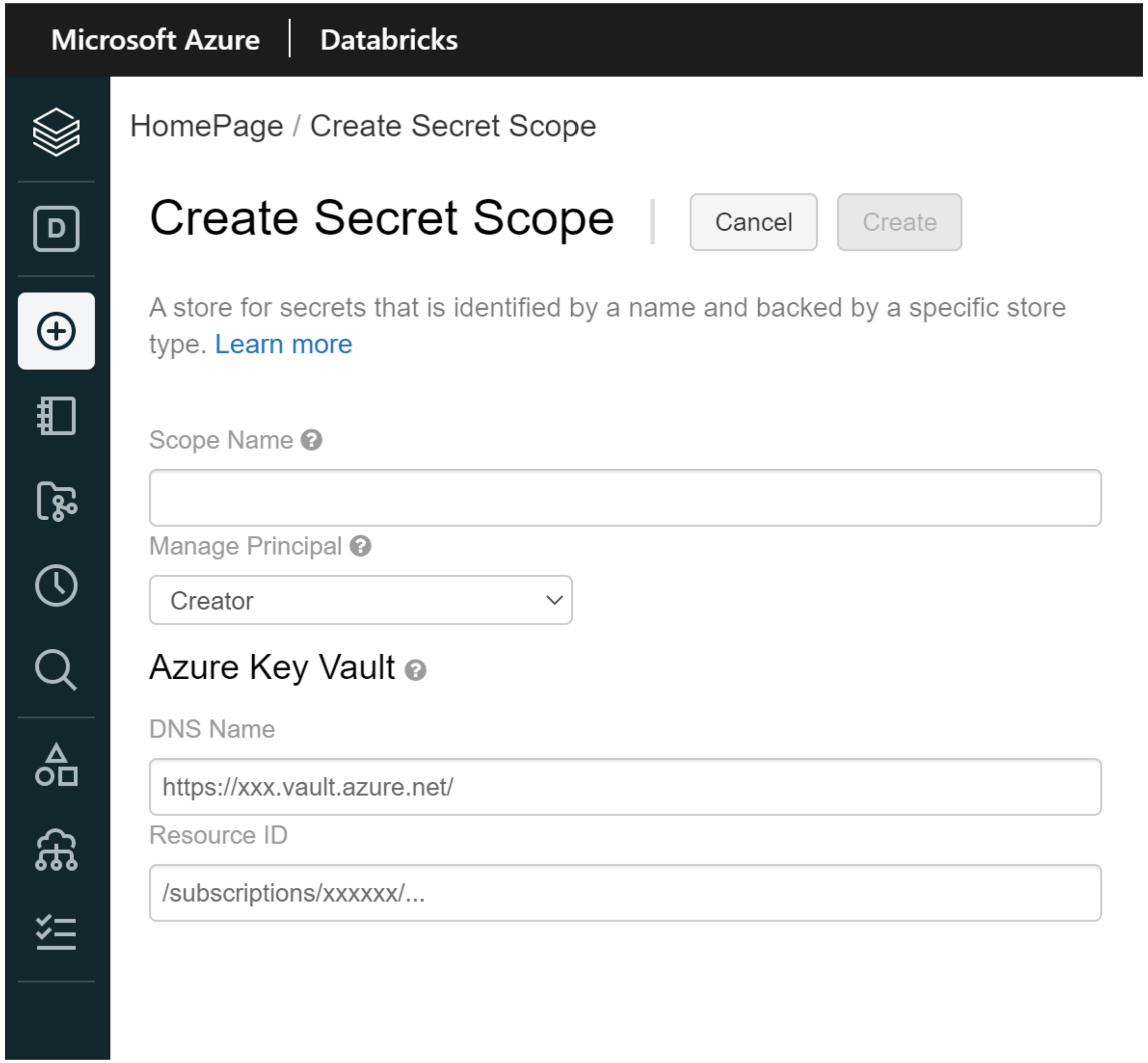
Azure Key Vault

Azure Key Vault is a tool for securely storing and accessing secrets. A secret is anything that you want to tightly control access to, such as API keys, passwords, or certificates.

- Instead of having our secret scopes be Databricks-backed (using secrets API), we can have them be Azure Key Vault-backed.
- To reference secrets stored in an Azure Key Vault, you can create a secret scope backed by Azure Key Vault.
- Creating an Azure Key Vault-backed secret scope is supported ONLY in the Azure Databricks UI. But we can manage the secrets using Azure SetSecret REST API or Azure portal UI.

Azure Key Vault

1. Verify that you have Contributor permission on the Azure Key Vault instance that you want to use to back the secret scope.
2. Go to https://<your_azure_databricks_url>#secrets/createScope (for example, https://westus.azuredatabricks.net#secrets/createScope)
3. Specify whether "All Users" or "Creator" has manage permissions for the scope
4. Set DNS Name and Resource ID. These are available in Properties tab of Azure Key Vault



HomePage / Create Secret Scope

Create Secret Scope

A store for secrets that is identified by a name and backed by a specific store type. Learn more

Scope Name

Manage Principal: Creator

Azure Key Vault

DNS Name: https://xxx.vault.azure.net/

Resource ID: /subscriptions/xxxxxx/...

Microsoft