Microsoft

Databricks Development: Notebooks & Jobs

Microsoft Services

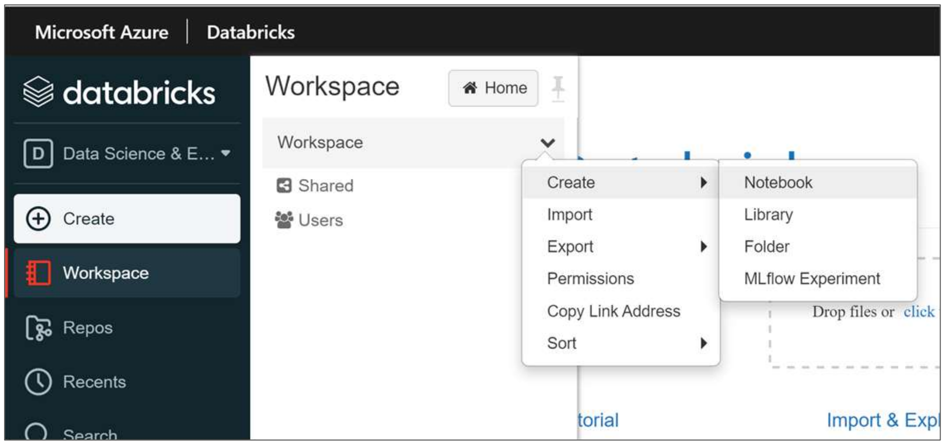
Lesson 1: Azure Databricks Notebooks and Jobs

After completing this lesson, you will be able to:

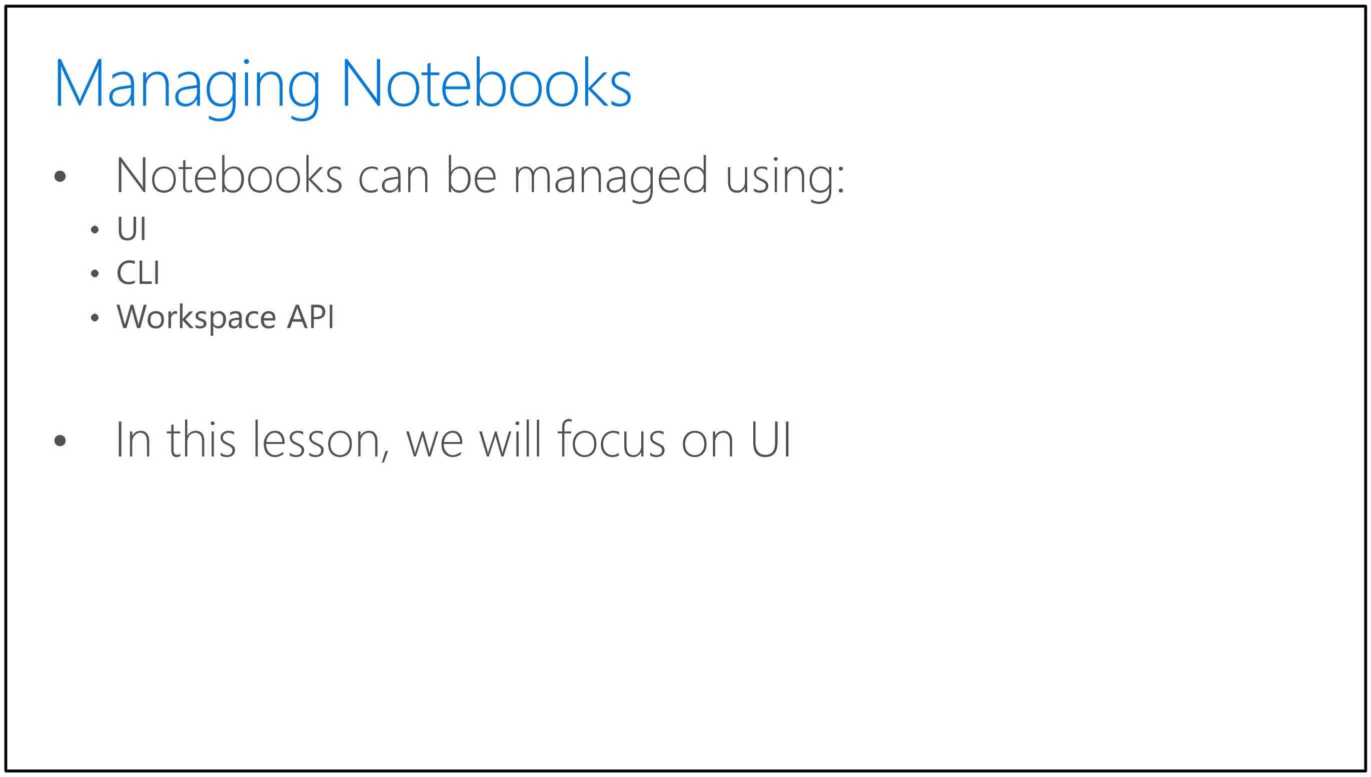
- Understand usage of notebooks
- Manage notebooks (create, delete, export, import, attach/detach)
- Use notebooks to run commands, create dashboards
- Create and schedule jobs

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



To create a notebook click the drop-down arrow next to a workspace or file name. Next expand create and select Notebook. Or navigate to the Azure Databricks tab on the left, and under common tasks select New Notebook.



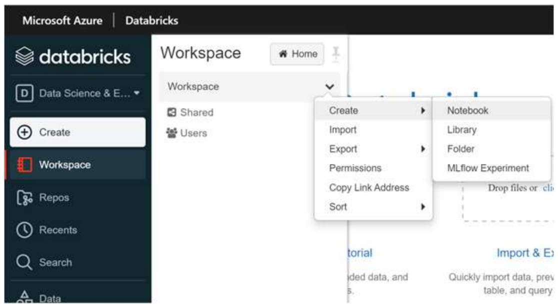
Managing Notebooks

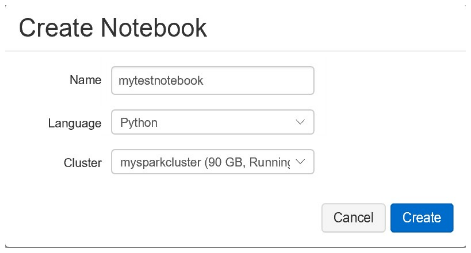
- Notebooks can be managed using:
  - UI
  - CLI
  - Workspace API
- In this lesson, we will focus on UI

[Download Python | Python.org]()

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

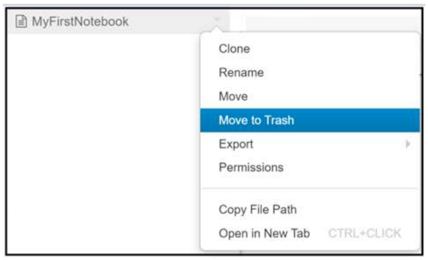
Language: Python

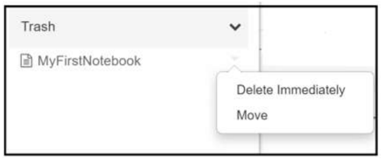
Cluster: mysparkcluster (90 GB, Running

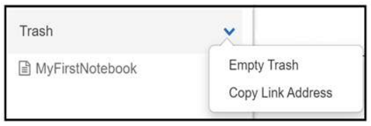
https://docs.microsoft.com/en-us/azure/databricks/notebooks/notebooks-manage#create-a-notebook

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.





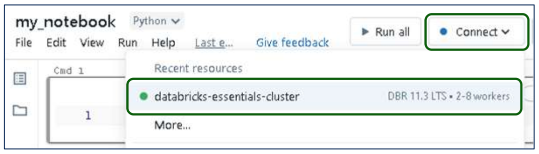


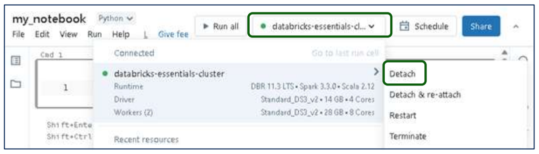
https://docs.microsoft.com/en-us/azure/databricks/workspace/workspace-objects#--delete-an-object

You can permanently delete a notebook in the Trash by selecting the Menu Dropdown to the right of the notebook and selecting Delete Immediately.
You can permanently delete all objects in the Trash by selecting the Down Caret to the right of the Trash folder and selecting Empty Trash.

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.





https://docs.microsoft.com/en-us/azure/databricks/notebooks/notebooks-manage#--attach-a-notebook-to-a-cluster
https://docs.microsoft.com/en-us/azure/databricks/notebooks/notebooks-manage#--detach-a-notebook-from-a-cluster

An execution context contains the state for a REPL environment for each supported programming language: Python, R, Scala, and SQL. When you run a cell in a notebook, the command is dispatched to the appropriate language REPL environment and run

Important

You can also use the REST 1.2 API to create an execution context and send a command to run in the execution context. Similarly, the command is dispatched to the language REPL environment and run. A cluster has a maximum number of execution contexts (145).

Once the number of execution contexts has reached this threshold, you cannot attach a notebook to the cluster or create a new execution context.

An attached notebook has the following Apache Spark variables defined.

Class	Variable Name
SparkContext	sc
SQLContext/HiveContext	sqlContext
SparkSession (Spark 2.x)	spark

Do not create a SparkSession, SparkContext, or SQLContext. Doing so will lead to inconsistent behavior.

To determine the Spark version of the cluster your notebook is attached to, run:

```
spark.version
```

To determine the Databricks Runtime version of the cluster your notebook is attached to, run:

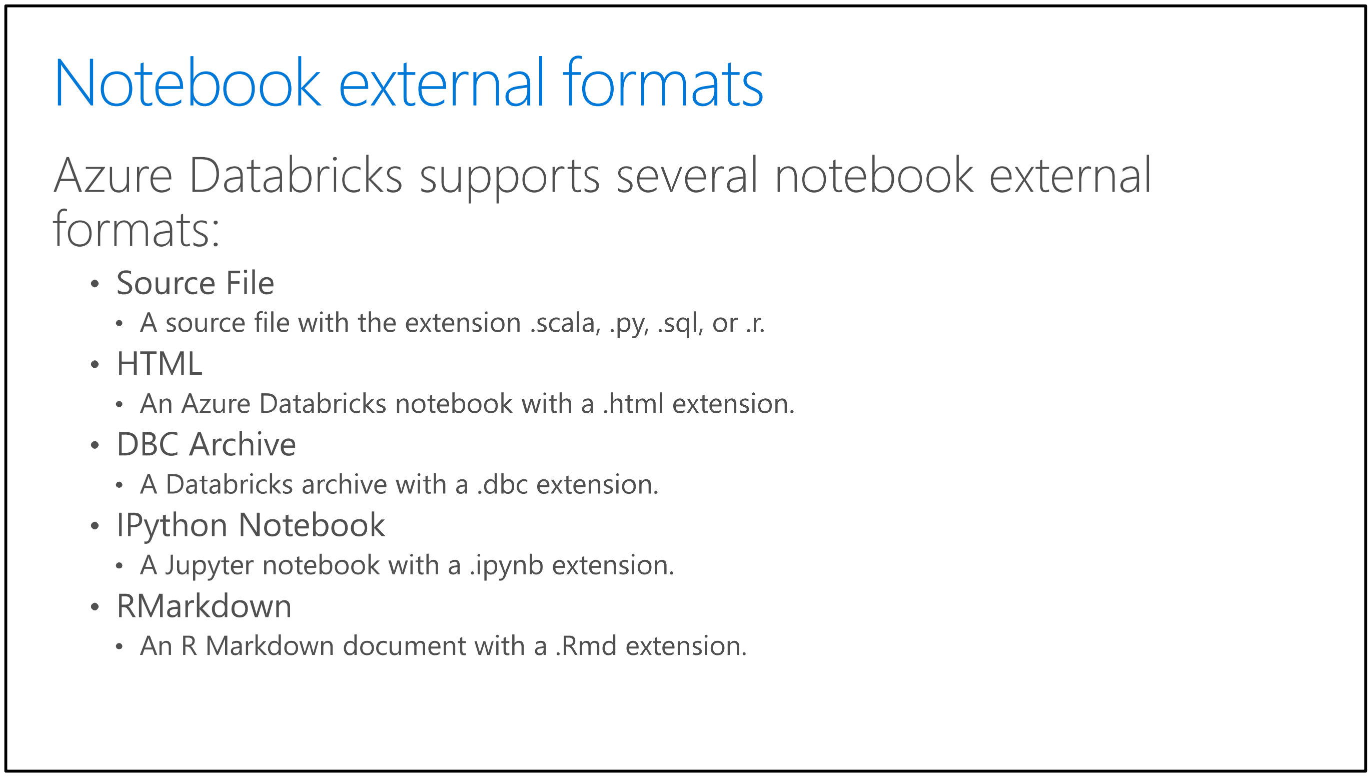
Scala

Copy

```
dbutils.notebook.getContext.tags("sparkVersion")
```

Python

```
spark.conf.get("spark.databricks.clusterUsageTags.sparkVersion")
```



Notebook external formats

Azure Databricks supports several notebook external formats:

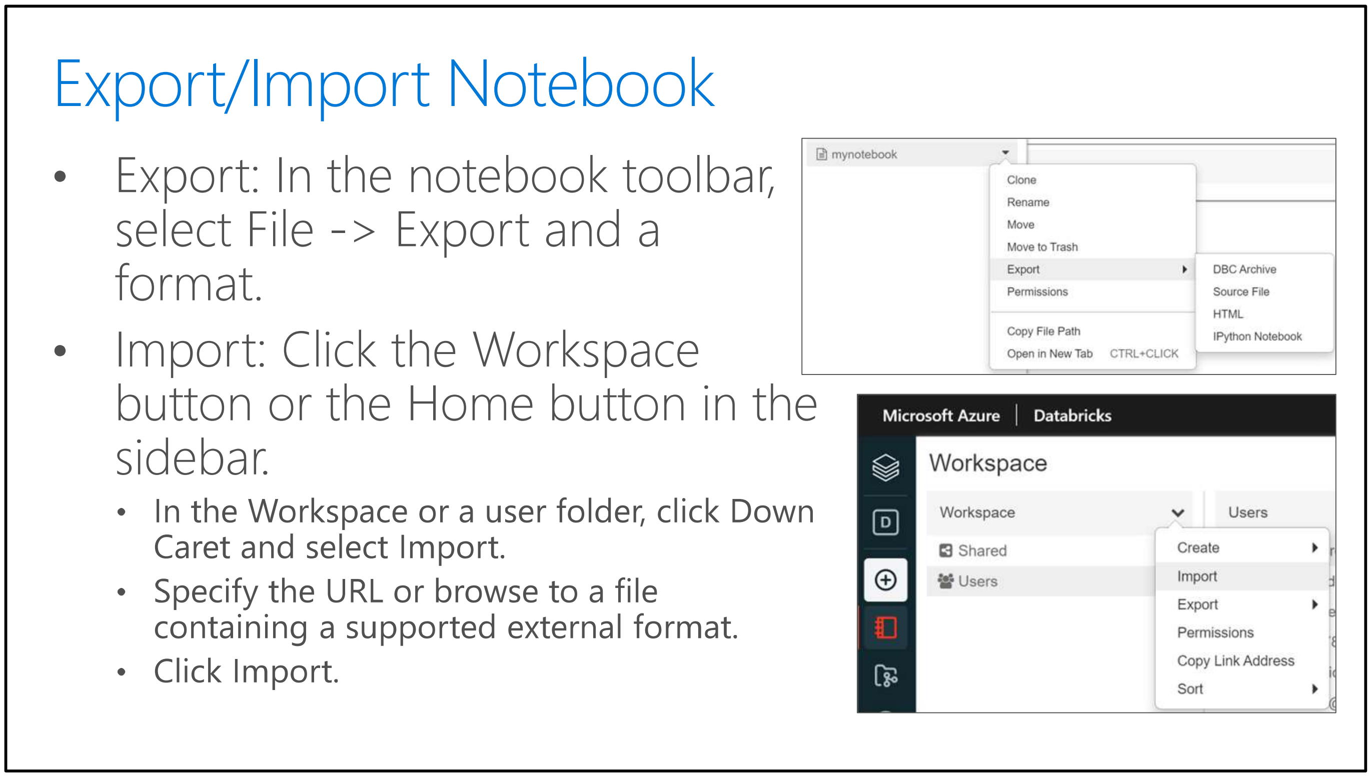
- Source File
  - A source file with the extension .scala, .py, .sql, or .r.
- HTML
  - An Azure Databricks notebook with a .html extension.
- DBC Archive
  - A Databricks archive with a .dbc extension.
- IPython Notebook
  - A Jupyter notebook with a .ipynb extension.
- RMarkdown
  - An R Markdown document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

https://docs.microsoft.com/en-us/azure/databricks/notebooks/notebooks-manage#--notebook-external-formats

DBC Archive:
To allow you to easily distribute Azure Databricks HTML notebooks, Azure Databricks supports the Databricks archive, which is a package that can contain a folder of notebooks or a single notebook. A Databricks archive is a JAR file with extra metadata and has the extension .dbc.

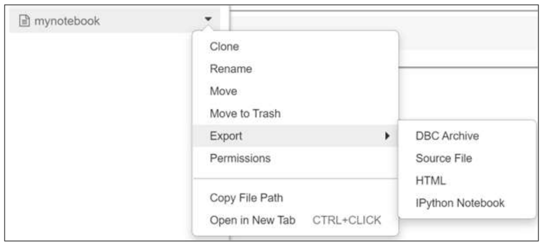
IPython Notebook:
https://jupyter-notebook.readthedocs.io/en/stable/

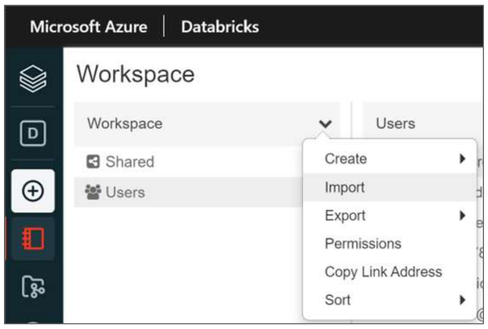
R Markdown document:
https://rmarkdown.rstudio.com/



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.



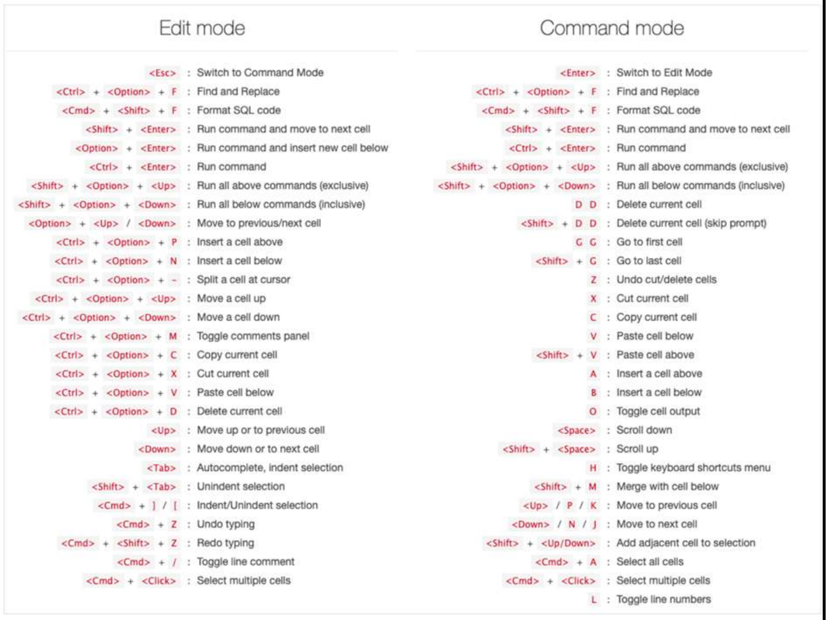


https://docs.microsoft.com/en-us/azure/databricks/notebooks/notebooks-manage#--import-a-notebook
https://docs.microsoft.com/en-us/azure/databricks/notebooks/notebooks-manage#--export-a-notebook

When you export a notebook as an Azure Databricks notebook (HTML), IPython notebook, or archive (DBC), and you have not previously cleared the results, the results of running the notebook are included.

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

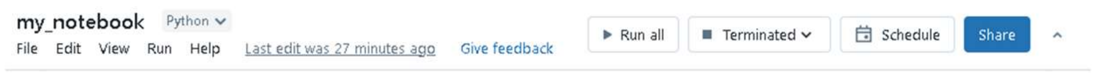
Command mode

<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

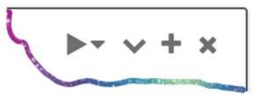
https://docs.azuredatabricks.net/user-guide/notebooks/notebook-use.html#using-notebooks

Using notebooks

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



- At upper right corner of each cell, it has
  - Two drop-down menus: Run Cell and Edit Menu
  - Two actions: Minimize/Maximize and Delete



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

Add a cell
To add a cell, mouse over a cell at the top or bottom and click the Add Cell icon, or access the notebook cell menu at the far right, click Down Caret, and select Add Cell Above or Add Cell Below.

Delete a cell
Go to the cell actions menu Cell Actions at the far right and click Delete Cell (Delete).

Mix languages
The primary language for each cell is shown in ( ) next to the notebook name:
You can override the primary language by specifying the language magic command %<language> at the beginning of a cell. The supported magic commands are: %python, %r, %scala, and %sql. Additionally:

%sh
Allows you to execute shell code in your notebook. Add the -e option in order to fail the cell (and subsequently a job or a run all command) if the shell command has a non-zero exit status.
%fs
Allows you to use Databricks Utilities filesystem commands. For more information, see Databricks File System - DBFS.
%md
Allows you to include various types of documentation, including text, images, and mathematical formulas and equations.

Include Documentation

To include documentation in a notebook you can use the %md magic command to identify Markdown markup. The included Markdown markup is rendered into HTML. For example, this Markdown snippet contains markup for a level-one heading:
%md # Hello This is a Title

Link to other notebooks
You can link to other notebooks or folders in Markdown cells using relative paths. Specify the href attribute of an anchor tag as the relative path, starting with a $ and then follow the same pattern as in Unix file systems:

```
%md
<a href="$./myNotebook">Link to notebook in same folder as current notebook</a>
<a href="$../myFolder">Link to folder in parent folder of current notebook</a>
<a href="$./myFolder2/myNotebook2">Link to nested notebook</a>
```

More info:
https://docs.azuredatabricks.net/user-guide/notebooks/notebook-use.html

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.

To run cells, the notebook must be attached to a cluster. If the cluster is not running, the cluster is started when you run one or more cells.

Run all above or below
Run All Below includes the cell you are in. Run All Above does not.

Python and Scala error highlighting
Python and Scala notebooks support error highlighting. That is, the line of code that is throwing the error will be highlighted in the cell. Additionally, if the error output is a stacktrace, the cell in which the error is thrown is displayed in the stacktrace as a link to the cell. You can click this link to jump to the offending code.

Notifications
Notifications alert you to certain events, such as which command is currently running during Run all cells and which commands are in error state. When your notebook is showing multiple error notifications, the first one will have a link that allows you to clear all notifications.

Run a notebook from another notebook
You can run a notebook from another notebook by using the %run <notebook> magic command. This is roughly equivalent to a :load command in a Scala REPL on your local machine or an import statement in Python. All variables defined in <notebook> become available in your current notebook.
%run must be in a cell by itself, because it runs the entire notebook inline.

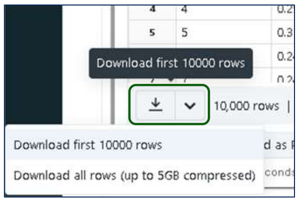
Note: You cannot use %run to run a Python file and import the entities defined in that file into a notebook. To import from a Python file you must package the file into a Python library, create an

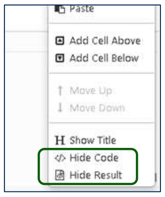
Azure Databricks library from that Python library, and install the library into the cluster you use to run your notebook.

https://docs.microsoft.com/en-us/azure/databricks/notebooks/notebooks-use#--run-a-notebook-from-another-notebook

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


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



Clear notebooks state and results
The clear in the notebook toolbar has the following clearing options:

- Clear Results
- Clear State
- Clear State & Results
- Clear State & Run all

Download a result
You can download a cell result that contains tabular output to your local machine. Click the Download Result button at the bottom of a cell.
A CSV file named export.csv is downloaded to your default download directory.

Hide and show cell content
Cell content consists of cell code and the result of running the cell. You can hide and show the cell code and result using the cell actions menu Cell Actions at the top right of the cell.

To hide cell code:
Click Down Caret and select Hide Code
To hide and show the cell result, do any of the following:

Click Down Caret and select Hide Result
Select Cell Minimize
Type Esc > Shift + o
To show hidden cell code or results, click the Show links:

- Show code
- Show result

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.

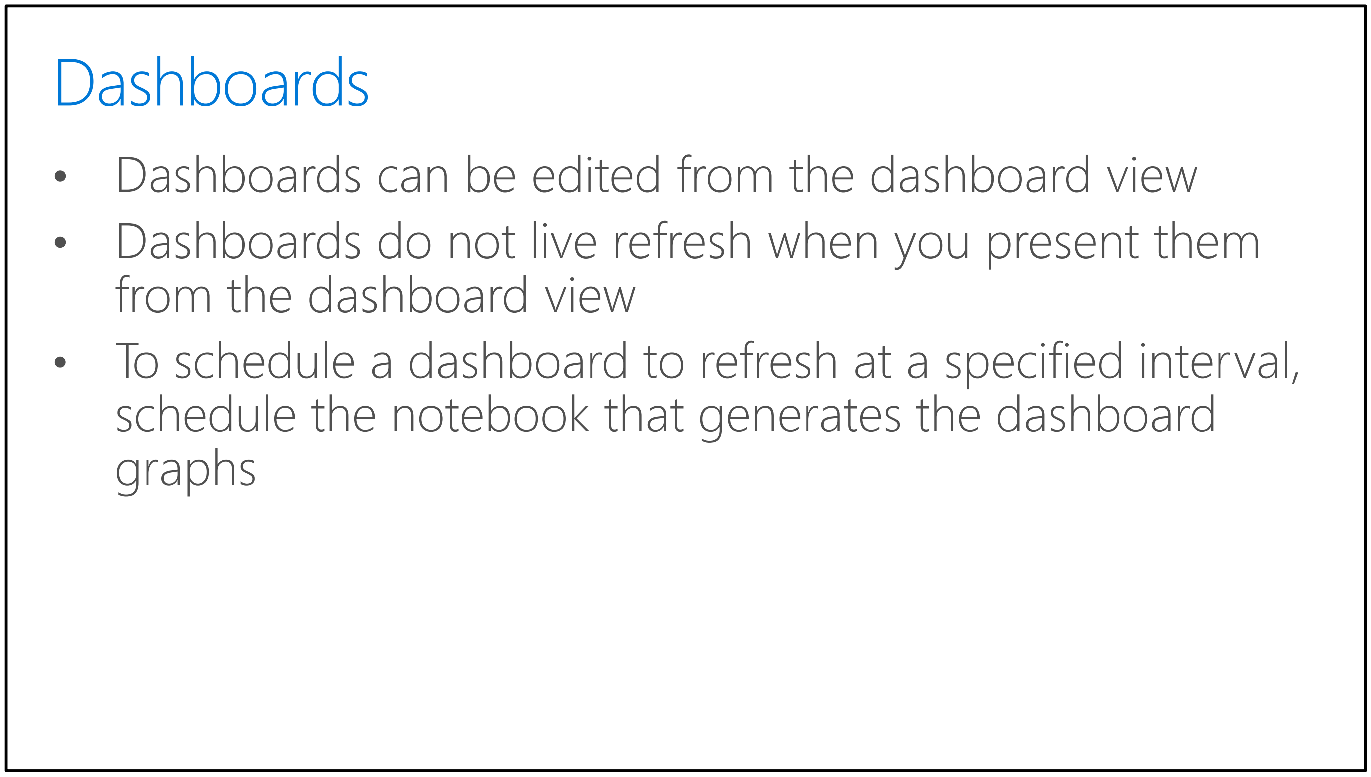
https://docs.microsoft.com/en-us/azure/databricks/notebooks/dashboards

In a nutshell, a Dashboard is a visual report backed by Apache Spark clusters, where users can consume information visually, or even interactively run queries by changing parameters. It is a simple way for users to instantly consume the insights generated by Spark

Publishing these insights as dashboards from Notebooks helps eliminate third-party integrations with the data processing platform, resulting in many handoffs, delays, and tedious work

Dashboards can be created directly from Databricks notebooks with a single click. In fact, a Dashboard is just another view of a notebook. Users can instantly create many different dashboards from one notebook, tailoring the presentation of the same results to different audiences.

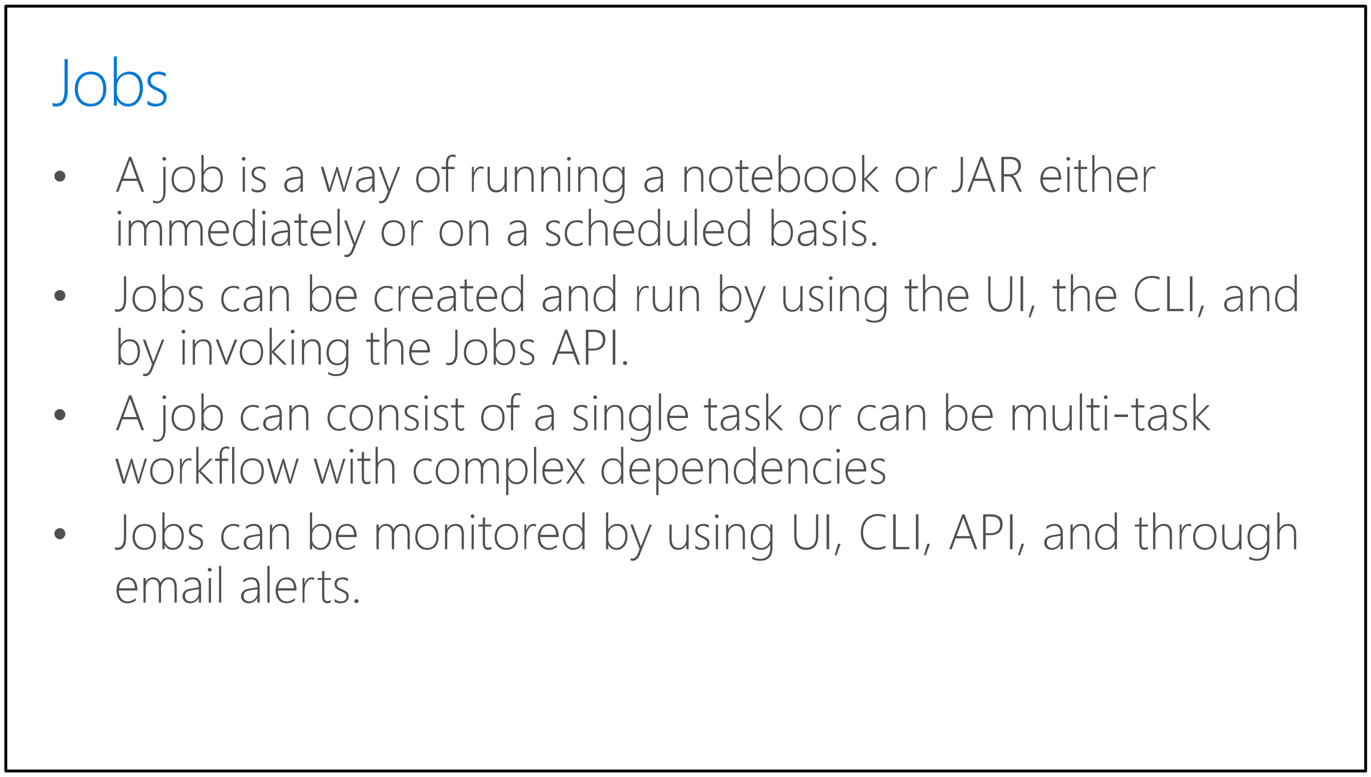
https://databricks.com/blog/2016/02/17/introducing-databricks-dashboards.html



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

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

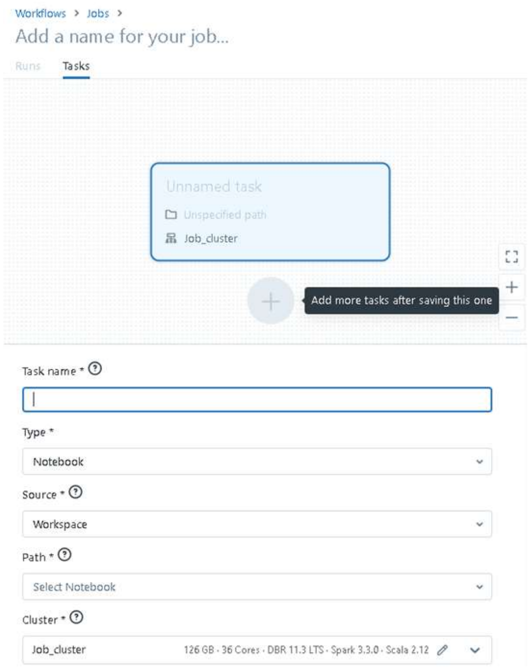
Jobs User Guide:
https://docs.azuredatabricks.net/user-guide/jobs.html

Job commands using CLI:
https://docs.azuredatabricks.net/user-guide/dev-tools/databricks-cli.html#job-cli

Job API:
https://docs.azuredatabricks.net/api/latest/jobs.html#job-api

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

Job parameters:
Notebook - Key-value pairs or a JSON string representing key-value pairs. Such parameters set the value of widgets.
JAR job - Main class and arguments.
spark-submit - Main class, path to the library JAR, and arguments

Dependent libraries:
Click Add next to Dependent Libraries. The dependent libraries are automatically attached to the cluster on launch. Follow the recommendations in Library dependencies for specifying dependencies.
https://docs.azuredatabricks.net/user-guide/jobs.html#library-dependencies

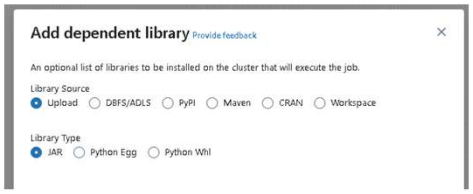
Cluster Type drop-down:
There is a tradeoff between running on a new cluster and on an existing cluster. We recommend that you run on a new cluster for production-level jobs or jobs that are important to complete. Existing clusters work best for tasks such as updating dashboards at regular intervals.
If you select a terminated cluster, and the job owner has Can Restart permission, the cluster is started when the job is scheduled to run.

Run a job:
You can run a job on a schedule or immediately.
The Azure Databricks job scheduler, like the Spark batch interface, is not intended for low latency jobs. Due to network or cloud issues, job runs may occasionally be delayed up to several minutes. In these situations, scheduled jobs will run immediately on service availability.
To run the job immediately, in the Active runs table click Run Now

More Information:
https://docs.azuredatabricks.net/user-guide/jobs.html#library-dependencies

Task Advanced Options 1/2

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



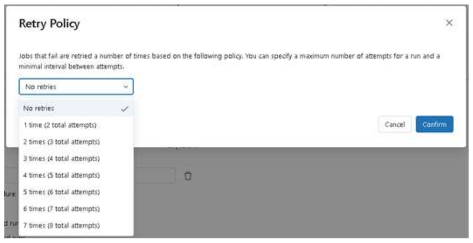
Add dependent library Provide feedback

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)

- Timeout:
  - Maximum completion time for a task

Timeout in seconds: No timeout

Job parameters:
Notebook - Key-value pairs or a JSON string representing key-value pairs. Such parameters set the value of widgets.
JAR job - Main class and arguments.
spark-submit - Main class, path to the library JAR, and arguments

Dependent libraries:
Click Add next to Dependent Libraries. The dependent libraries are automatically attached to the cluster on launch. Follow the recommendations in Library dependencies for specifying dependencies.
https://docs.azuredatabricks.net/user-guide/jobs.html#library-dependencies

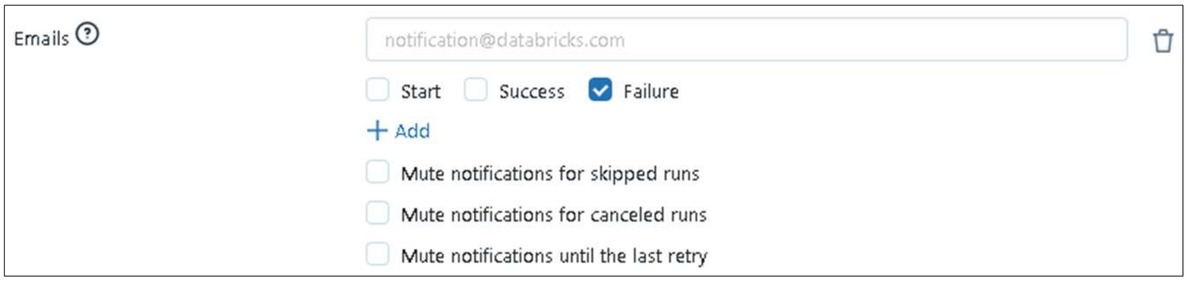
Cluster Type drop-down:
There is a tradeoff between running on a new cluster and on an existing cluster. We recommend that you run on a new cluster for production-level jobs or jobs that are important to complete. Existing clusters work best for tasks such as updating dashboards at regular intervals.
If you select a terminated cluster, and the job owner has Can Restart permission, the cluster is started when the job is scheduled to run.

Run a job:
You can run a job on a schedule or immediately.
The Azure Databricks job scheduler, like the Spark batch interface, is not intended for low latency jobs. Due to network or cloud issues, job runs may occasionally be delayed up to several minutes. In these situations, scheduled jobs will run immediately on service availability.
To run the job immediately, in the Active runs table click Run Now

More Information:
https://docs.azuredatabricks.net/user-guide/jobs.html#library-dependencies

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

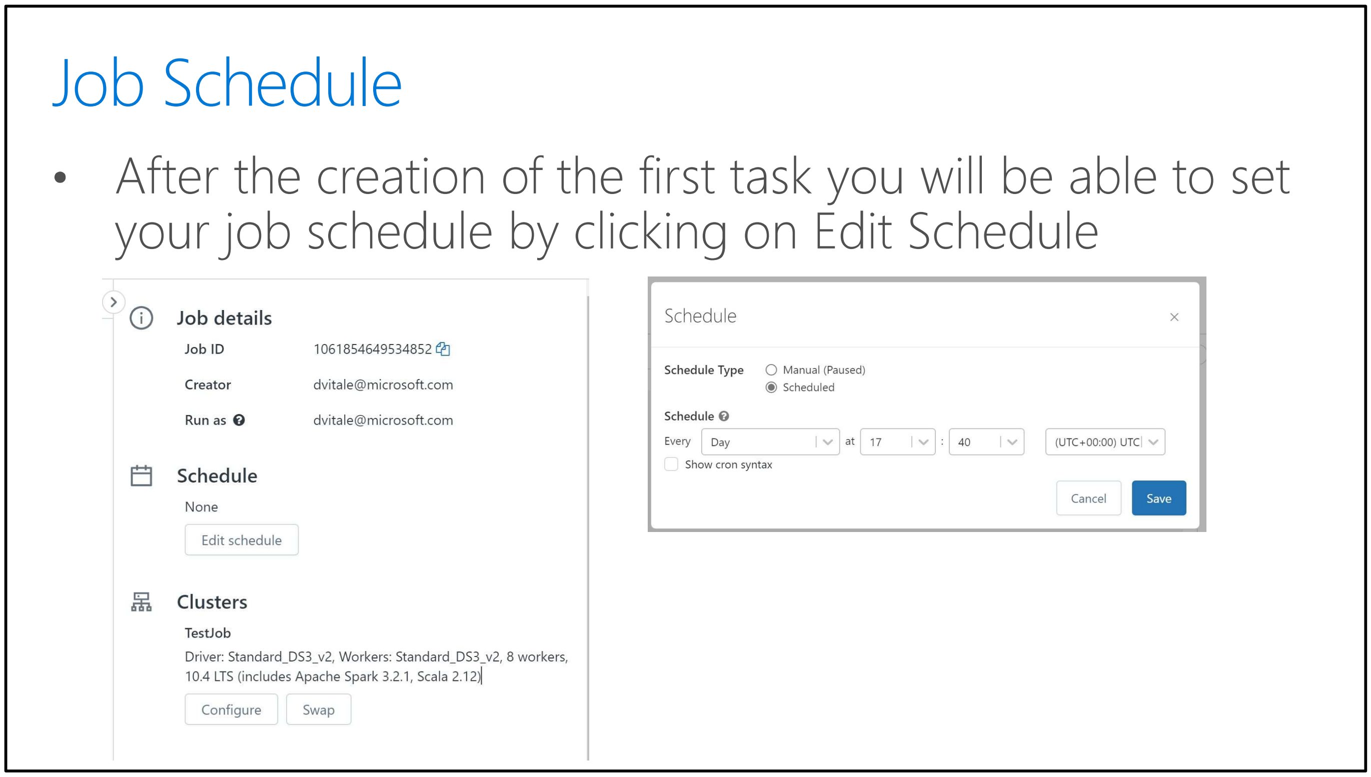
Job parameters:
Notebook - Key-value pairs or a JSON string representing key-value pairs. Such parameters set the value of widgets.
JAR job - Main class and arguments.
spark-submit - Main class, path to the library JAR, and arguments

Dependent libraries:
Click Add next to Dependent Libraries. The dependent libraries are automatically attached to the cluster on launch. Follow the recommendations in Library dependencies for specifying dependencies.
https://docs.azuredatabricks.net/user-guide/jobs.html#library-dependencies

Cluster Type drop-down:
There is a tradeoff between running on a new cluster and on an existing cluster. We recommend that you run on a new cluster for production-level jobs or jobs that are important to complete. Existing clusters work best for tasks such as updating dashboards at regular intervals.
If you select a terminated cluster, and the job owner has Can Restart permission, the cluster is started when the job is scheduled to run.

Run a job:
You can run a job on a schedule or immediately.
The Azure Databricks job scheduler, like the Spark batch interface, is not intended for low latency jobs. Due to network or cloud issues, job runs may occasionally be delayed up to several minutes. In these situations, scheduled jobs will run immediately on service availability.
To run the job immediately, in the Active runs table click Run Now

More Information:
https://docs.azuredatabricks.net/user-guide/jobs.html#library-dependencies



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

Clusters

TestJob

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

Schedule

Schedule Type: Manual (Paused) [✓ Scheduled]

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

Show cron syntax

Set Retry Policy:
Jobs that fail will be 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.

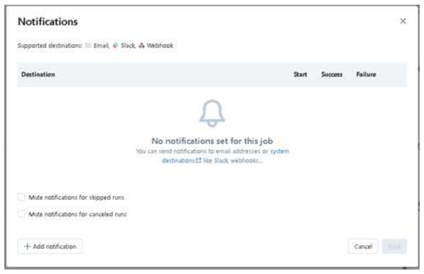
Maximum concurrent runs:
The maximum number of runs that can be run in parallel. On starting a new run, Databricks skips the run if the job has already reached its maximum number of active runs. Set this value higher than the default of 1 if you want to be able to perform multiple runs of the same job concurrently. This is useful for example if you trigger your job on a frequent schedule and want to allow consecutive runs to overlap with each other, or if you want to trigger multiple runs that differ by their input parameters.

Control access to jobs:
Job access control enable job owners and administrators to grant fine grained permissions on their jobs. With job access controls, job owners can choose which other users or groups can view results of the job. Owners can also choose who can manage runs of their job (that is, invoke Run Now and Cancel.)

By default, all users can create and modify jobs unless an administrator enables jobs access control. With jobs access control, individual permissions determine a user's abilities.
More Info: https://docs.azuredatabricks.net/administration-guide/admin-settings/jobs-acl.html

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

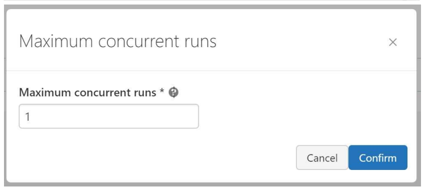
Destination	Start	Success	Failure

No notifications set for this job

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

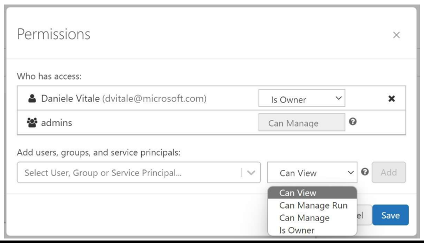
Mute notifications for skipped runs

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

Set Retry Policy:
Jobs that fail will be 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.

Maximum concurrent runs:
The maximum number of runs that can be run in parallel. On starting a new run, Databricks skips the run if the job has already reached its maximum number of active runs. Set this value higher than the default of 1 if you want to be able to perform multiple runs of the same job concurrently. This is useful for example if you trigger your job on a frequent schedule and want to allow consecutive runs to overlap with each other, or if you want to trigger multiple runs that differ by their input parameters.

Control access to jobs:
Job access control enable job owners and administrators to grant fine grained permissions on their jobs. With job access controls, job owners can choose which other users or groups can view results of the job. Owners can also choose who can manage runs of their job (that is, invoke Run Now and Cancel.)

By default, all users can create and modify jobs unless an administrator enables jobs access control. With jobs access control, individual permissions determine a user’s abilities.
More Info: https://docs.azuredatabricks.net/administration-guide/admin-settings/jobs-acl.html

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?

Q: Databricks Notebooks can be managed using?
A: UI, CLI, Workspace API

Q: What is the magic command to run a notebook from another notebook
A: By using the %run <notebook> magic command.
(Azure Databricks workflows has similar function with more capability, Notebook workflows — Databricks Documentation)

How can we run a notebook or JAR either immediately or on a scheduled basis?
A: A job is a way of running a notebook or JAR either immediately or on a scheduled basis

Microsoft