ADVANCING ANALYTICS

Operational & Secret Management

databricks

Operational and Secret Management

Managing our Databricks environments and workflows effectively has many benefits: improved collaboration and developer workflow, better security, and greater control over governance.

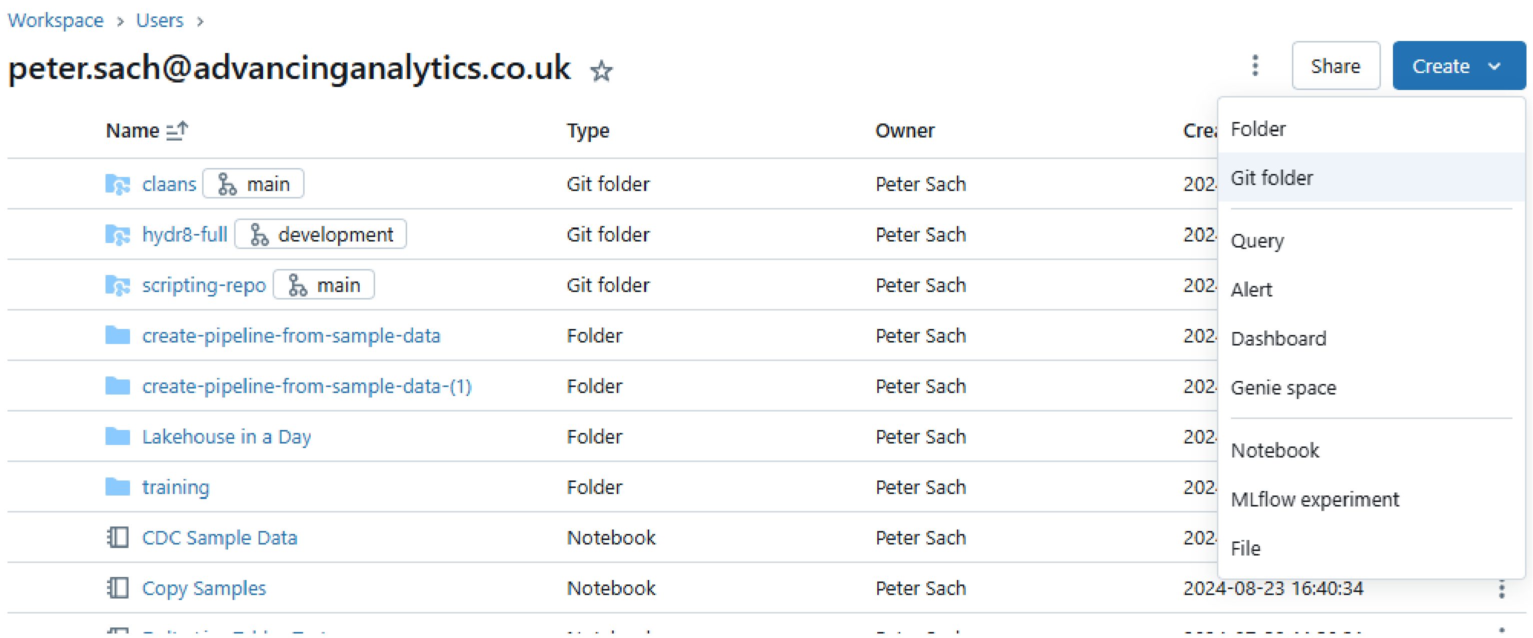
Databricks comes with several tools to help with this:

- **Repositories** (or Repos) to source control our notebooks.
- **DBUtils** helps us interact with the management layer.
- **Secret management** to keep sensitive values secure.
- **Databricks CLI** to manage the environment through the terminal.
- **DBConnect** allows us to work in Databricks using common IDEs.

Repositories

Databricks Git Folders

- In any modern data project, version control is a must
- Easy to clone code directly into a workspace using Git Folders
- Setup Git Folders using Workspace tab -> Create -> Git Folder
- Better collaboration, easier rollback, and more control over changes



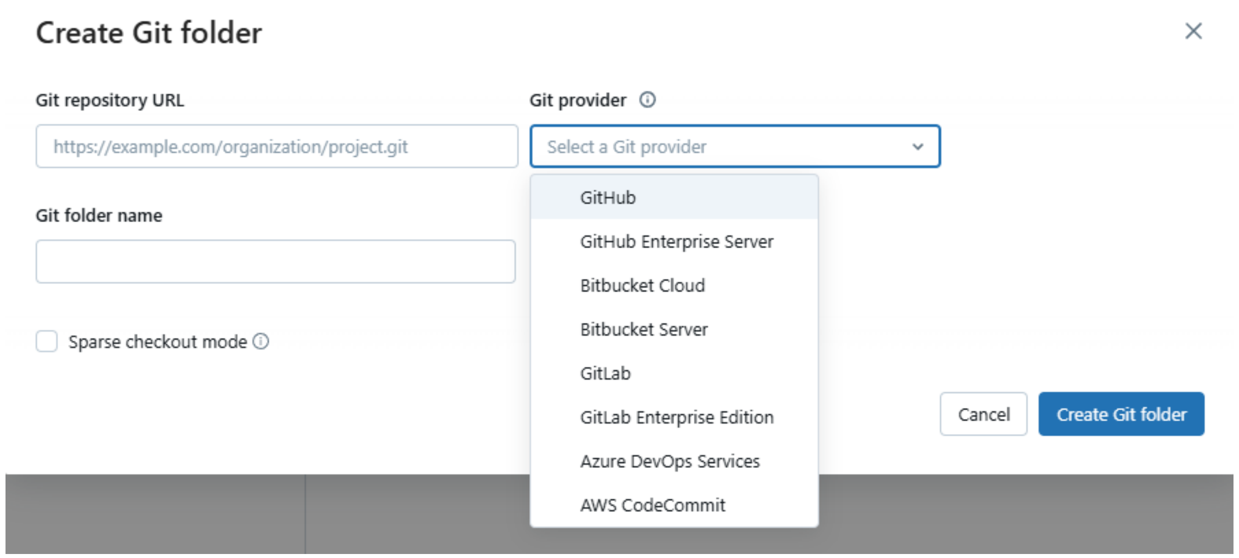
Workspace > Users >

peter.sach@advancinganalytics.co.uk

Name	Type	Owner	Crea
claans main	Git folder	Peter Sach	202
hydr8-full development	Git folder	Peter Sach	202
scripting-repo main	Git folder	Peter Sach	202
create-pipeline-from-sample-data	Folder	Peter Sach	202
create-pipeline-from-sample-data-(1)	Folder	Peter Sach	202
Lakehouse in a Day	Folder	Peter Sach	202
training	Folder	Peter Sach	202
CDC Sample Data	Notebook	Peter Sach	202
Copy Samples	Notebook	Peter Sach	2024-08-23 16:40:34

ADVANCING ANALYTICS

Source Control Providers



Create Git folder

Git repository URL: https://example.com/organization/project.git

Git provider: Select a Git provider

- GitHub
- GitHub Enterprise Server
- Bitbucket Cloud
- Bitbucket Server
- GitLab
- GitLab Enterprise Edition
- Azure DevOps Services
- AWS CodeCommit

Git folder name

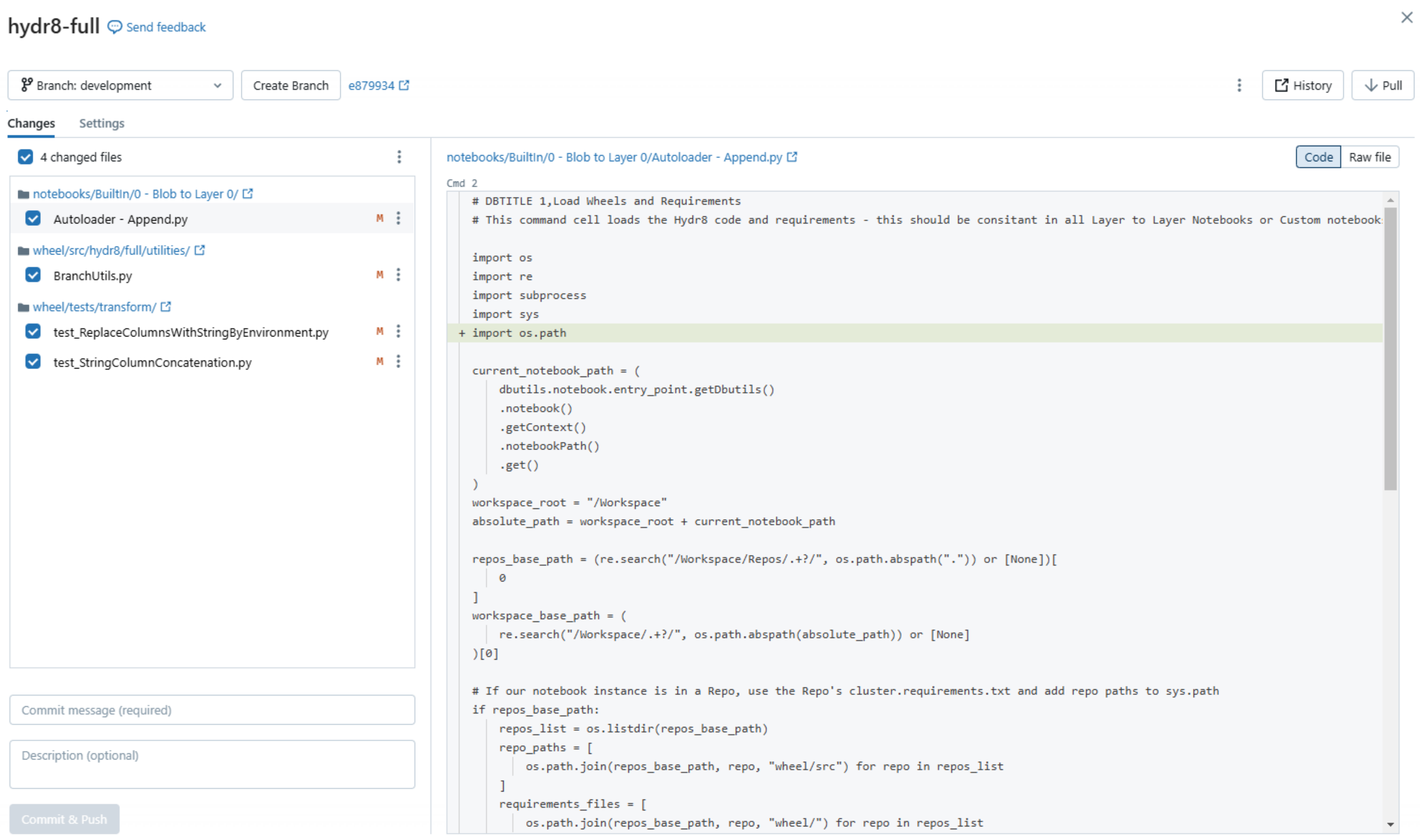
Sparse checkout mode

Bitbucket

GitLab

ADVANCING ANALYTICS

Repository UI



4 changed files

notebooks/BuiltIn/0 - Blob to Layer 0/
- Autoloader - Append.py M

wheel/src/hydr8/full/utilities/
- BranchUtils.py M

wheel/tests/transform/
- test_ReplaceColumnsWithStringByEnvironment.py M
- test_StringColumnConcatenation.py M

Commit message (required)

Description (optional)

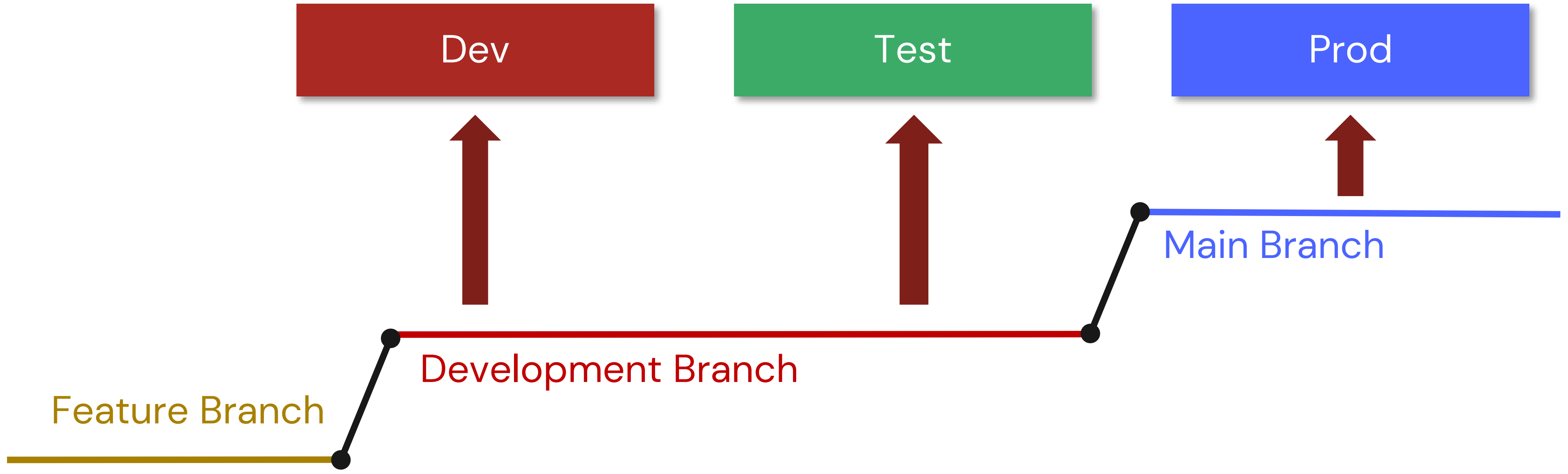
Commit & Push

notebooks/BuiltIn/0 - Blob to Layer 0/Autoloader - Append.py

Cmd 2

```
# DBTITLE 1,Load Wheels and Requirements
# This command cell loads the Hydr8 code and requirements - this should be consitant in all Layer to Layer Notebooks or Custom notebook

import os
import re
import subprocess
import sys
+ import os.path

current_notebook_path = (
    dbutils.notebook.entry_point.getDbutils()
    .notebook()
    .getContext()
    .notebookPath()
    .get()
)
workspace_root = "/Workspace"
absolute_path = workspace_root + current_notebook_path

repos_base_path = (re.search("/Workspace/Repos/.+?/", os.path.abspath(".")) or [None])[
    0
]
workspace_base_path = (
    re.search("/Workspace/.+?/", os.path.abspath(absolute_path)) or [None]
)[0]

# If our notebook instance is in a Repo, use the Repo's cluster.requirements.txt and add repo paths to sys.path
if repos_base_path:
    repos_list = os.listdir(repos_base_path)
    repo_paths = [
        os.path.join(repos_base_path, repo, "wheel/src") for repo in repos_list
    ]
    requirements_files = [
        os.path.join(repos_base_path, repo, "wheel/") for repo in repos_list
```

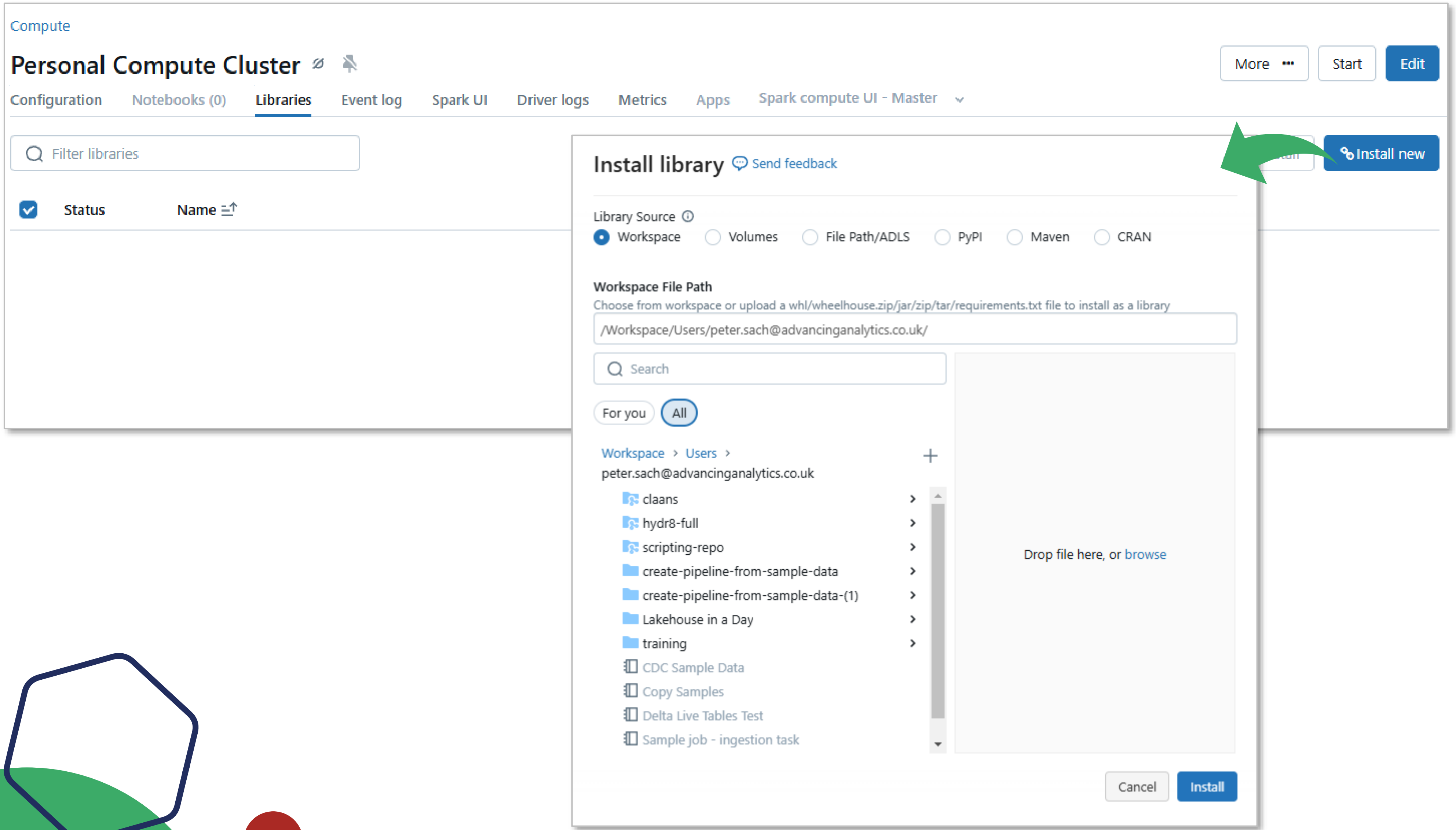
Branching & Deployment



ADVANCING ANALYTICS

Package Management

Installing Libraries



Compute

Personal Compute Cluster

Install library Send feedback

Library Source: [✓ Workspace] Volumes, File Path/ADLS, PyPI, Maven, CRAN

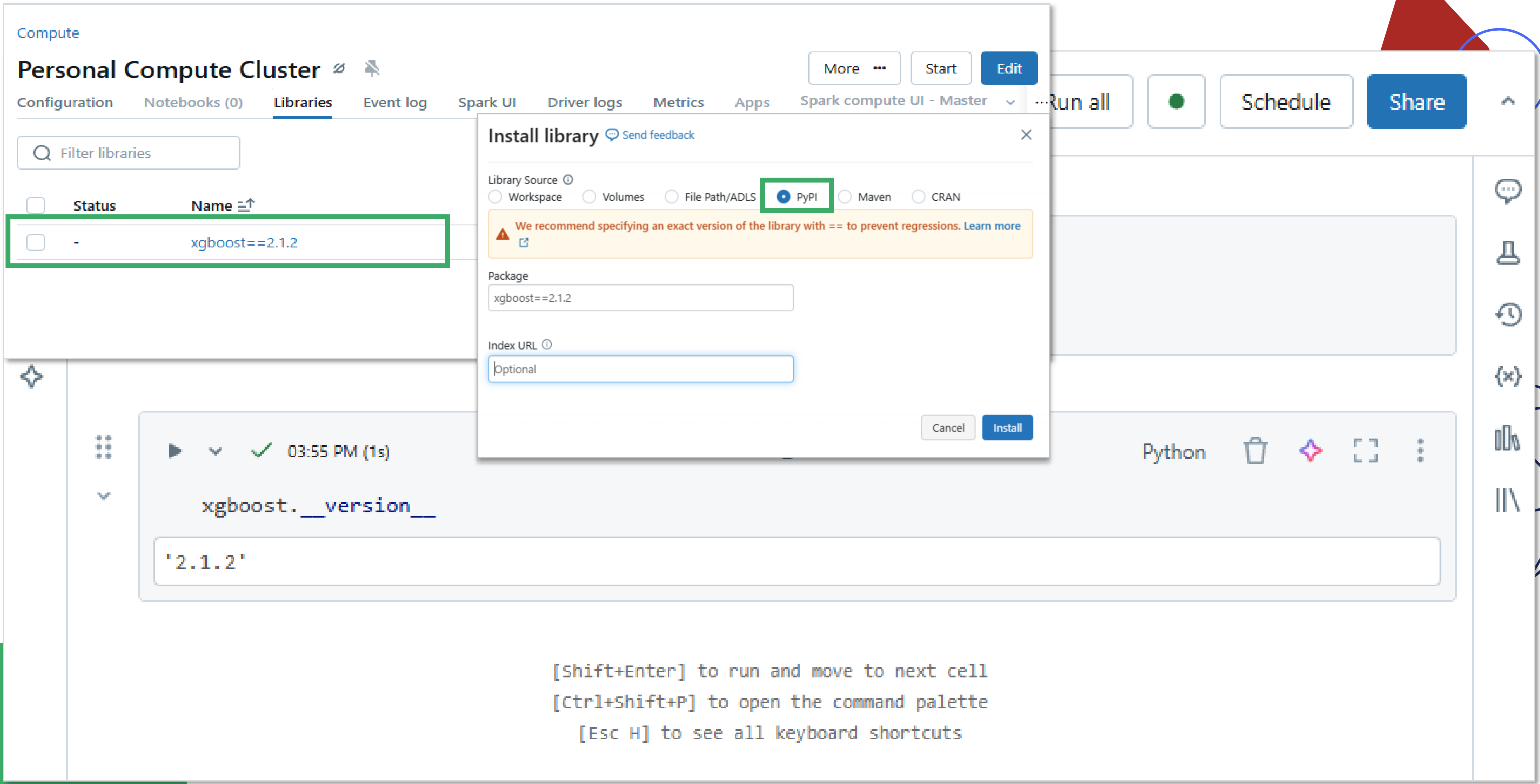
Workspace File Path

Choose from workspace or upload a whl/wheelhouse.zip/jar/zip/tar/requirements.txt file to install as a library

/Workspace/Users/peter.sach@advancinganalytics.co.uk/

Search
For you
All
Workspace › Users ›
peter.sach@advancinganalytics.co.uk
claans
hydr8-full
scripting-repo
create-pipeline-from-sample-data
create-pipeline-from-sample-data-(1)
Lakehouse in a Day
training
CDC Sample Data
Copy Samples
Delta Live Tables Test
Sample job - ingestion task
Drop file here, or browse
Cancel
Install

ADVANCING ANALYTICS



Status	Name
-	xgboost==2.1.2

Install library

Library Source: Workspace, Volumes, File Path/ADLS, [✓ PyPI], Maven, CRAN

We recommend specifying an exact version of the library with == to prevent regressions. Learn more

Package: xgboost==2.1.2

Index URL: Optional

03:55 PM (1s)

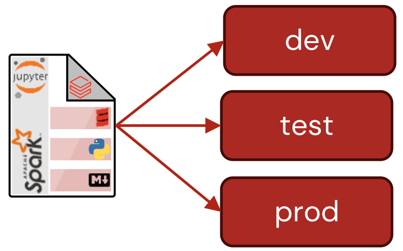
```
xgboost.__version__
```

```
'2.1.2'
```

```
[Shift+Enter] to run and move to next cell
[Ctrl+Shift+P] to open the command palette
[Esc H] to see all keyboard shortcuts
```

Code Distribution & Deployment

- Code Distribution



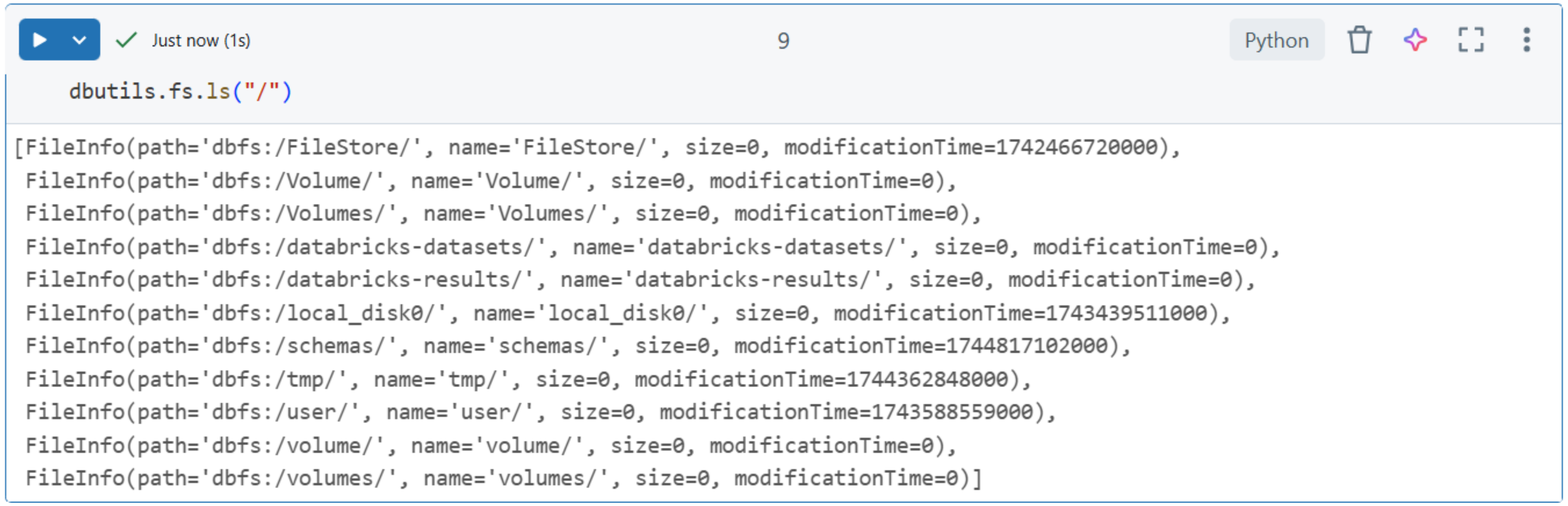
- Python Wheel
  - Designed to **organize and distribute** code quickly
  - Pre-compiled package for **quick and fast installation**
  - Binary format, contains all the **modules and files**
  - **Single object** that's easy to **deploy**
  - Keeps code **well-organized** and **reusable**
  - Easily **integrates** into CI/CD pipelines

- Databricks Asset Bundles (DAB)
  - Package entire Databricks project – including resources and workflows etc
  - Databricks **recommended** approach for code distribution
  - **Databricks Infrastructure-as-code (IaC)** approach
  - Configure and deploy as **single package**
  - Automates deployment **across environments**
  - Easily **integrates** into CI/CD pipelines

Databricks Utilities (dbutils)

Introduction to dbutils

- Built-in Databricks Utilities library
- Helps interact with the environment
- Run commands from notebooks
- Useful for files, secrets, jobs, and more

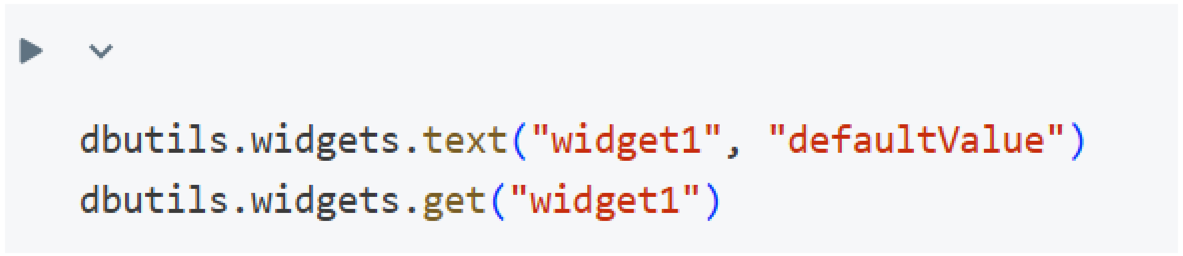


```
dbutils.fs.ls("/")
```

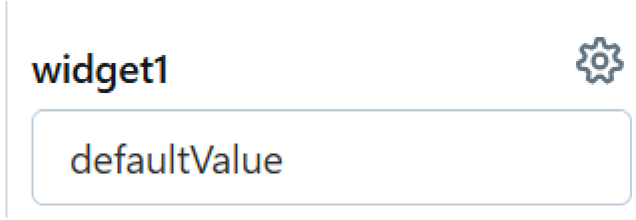
```
[FileInfo(path='dbfs:/FileStore/', name='FileStore/', size=0, modificationTime=1742466720000),
 FileInfo(path='dbfs:/Volume/', name='Volume/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/Volumes/', name='Volumes/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/databricks-datasets/', name='databricks-datasets/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/databricks-results/', name='databricks-results/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/local_disk0/', name='local_disk0/', size=0, modificationTime=1743439511000),
 FileInfo(path='dbfs:/schemas/', name='schemas/', size=0, modificationTime=1744817102000),
 FileInfo(path='dbfs:/tmp/', name='tmp/', size=0, modificationTime=1744362848000),
 FileInfo(path='dbfs:/user/', name='user/', size=0, modificationTime=1743588559000),
 FileInfo(path='dbfs:/volume/', name='volume/', size=0, modificationTime=0),
 FileInfo(path='dbfs:/volumes/', name='volumes/', size=0, modificationTime=0)]
```

Uses for dbutils

- **dbutils.help()** to explore available commands
- **dbutils.fs** lets you manage files in DBFS
- **dbutils.widgets** enables notebook parameters



```
dbutils.widgets.text("widget1", "defaultValue")
dbutils.widgets.get("widget1")
```



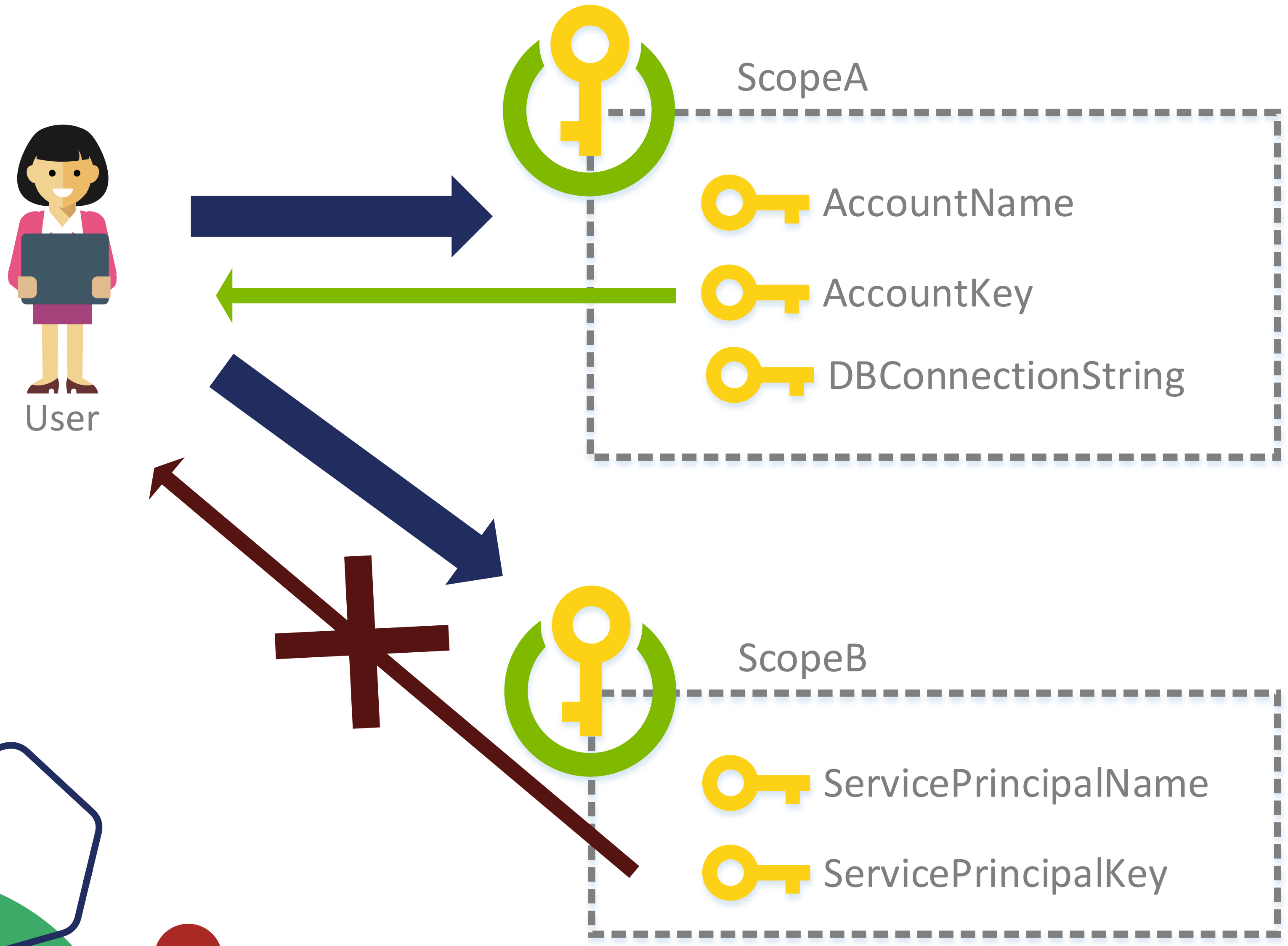
widget1

defaultValue

Demo: Using dbutils

Secret Management

Secrets Scope

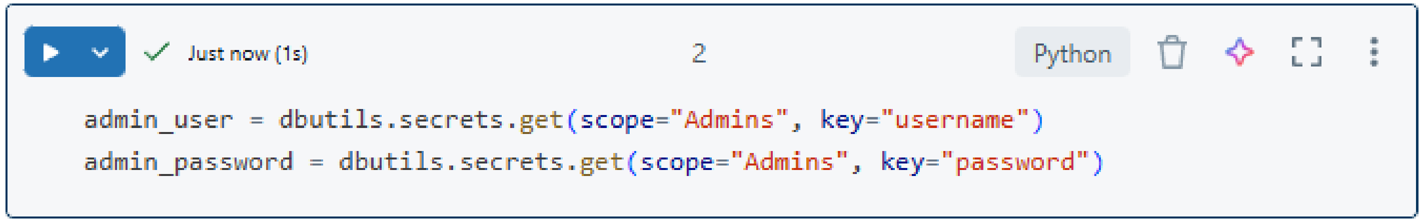


Secrets are never displayed in Databricks notebooks, even if you have access!

Any attempt to display the value will return **[REDACTED]**!

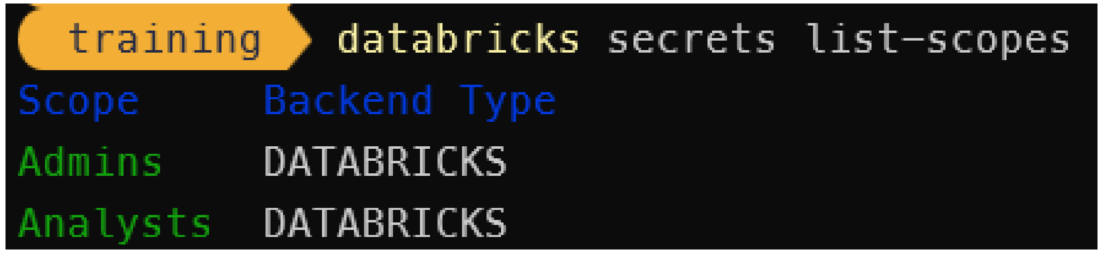
Secrets Retrieval

Secrets are retrieved through the "dbutils" function library within scala/python scripts



```python
admin_user = dbutils.secrets.get(scope="Admins", key="username")
admin_password = dbutils.secrets.get(scope="Admins", key="password")
```

But administrated through the Databricks Command Line Interface (CLI)



```
 training  databricks secrets list-scopes
Scope     Backend Type
Admins    DATABRICKS
Analysts  DATABRICKS
```

Secrets - Databricks-Backed

- **Managed and stored** by Databricks
- Scopes created and managed via **CLI only**
- Access is **controlled per scope**

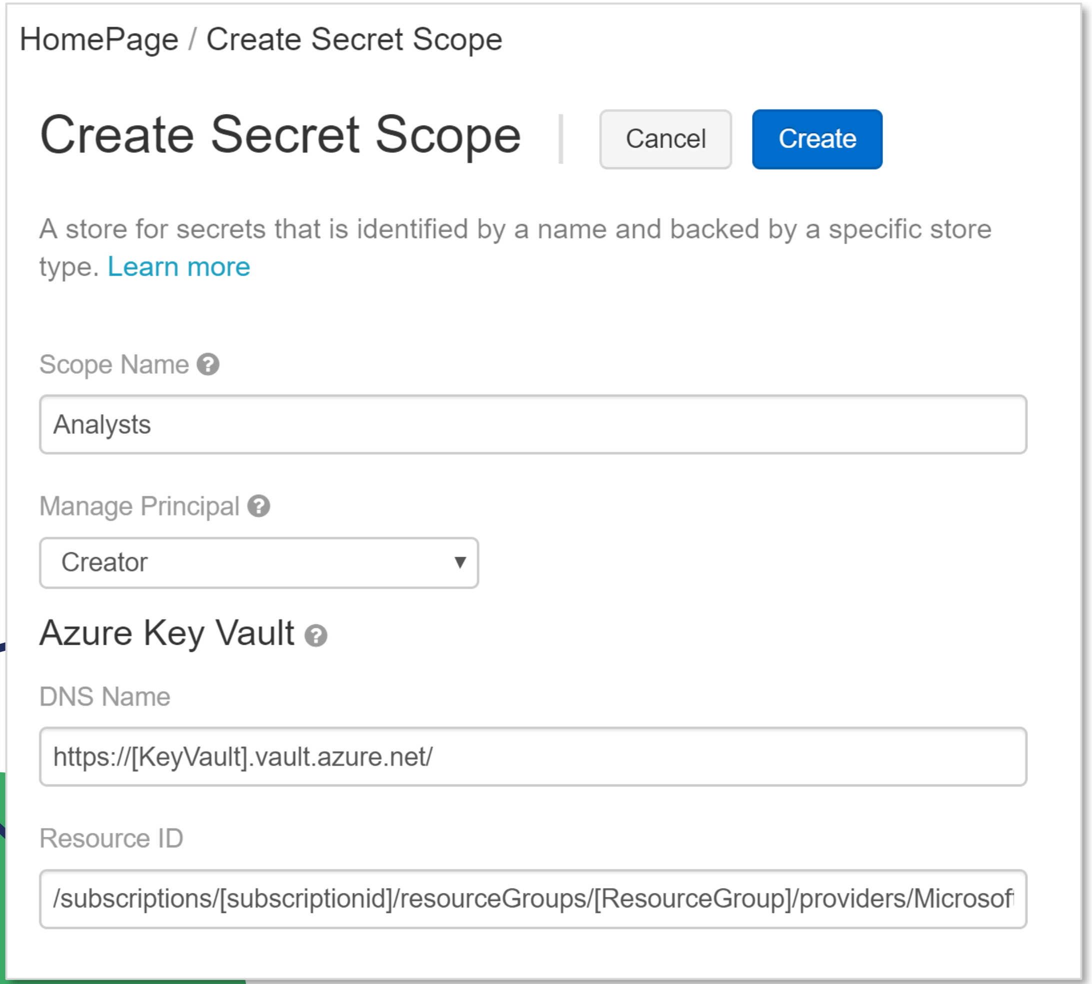
```
C:\> databricks secrets create-scope --scope Admins

C:\> databricks secrets put-secret  Admins  MySecret --string-value Pa$$word

C:\> databricks secrets list-secrets --scope Admins
Key            Last Updated Timestamp
----------    --------------
MySecret       1564142221488

C:\> databricks secrets put-acl Admins  terry@advancinganalytics.co.uk READ
```

- **Less flexible** than Key Vault scopes
- No need to **manage external resources**
- Good for **managing multiple secrets** directly

ADVANCING ANALYTICS

Secrets - Key Vault-Backed

*https://<workspace-url>**#secrets/createScope***



HomePage / Create Secret Scope

Create Secret Scope | Cancel | Create

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

Scope Name	Analysts
Manage Principal	Creator

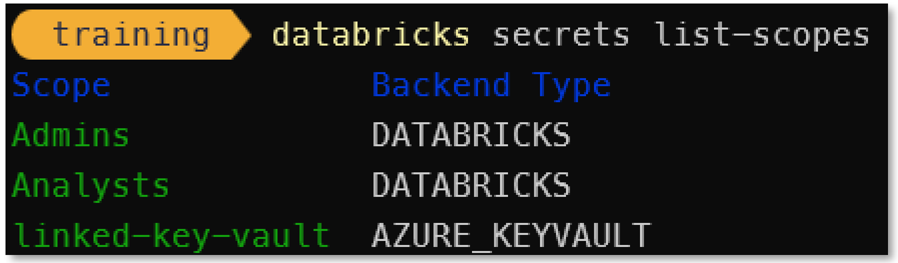
Azure Key Vault

DNS Name	https://[KeyVault].vault.azure.net/
Resource ID	/subscriptions/[subscriptionid]/resourceGroups/[ResourceGroup]/providers/Microsof

An alternative is to use Key Vault. This is a better approach as the vault can be shared by other resources such as Data Factory, meaning you only store your password in one place.

This makes managing expiring certificates, cycling keys etc much easier!

User access to this scope is still managed via the CLI



```
 training  databricks secrets list-scopes
Scope             Backend Type
Admins            DATABRICKS
Analysts          DATABRICKS
linked-key-vault  AZURE_KEYVAULT
```

ADVANCING ANALYTICS

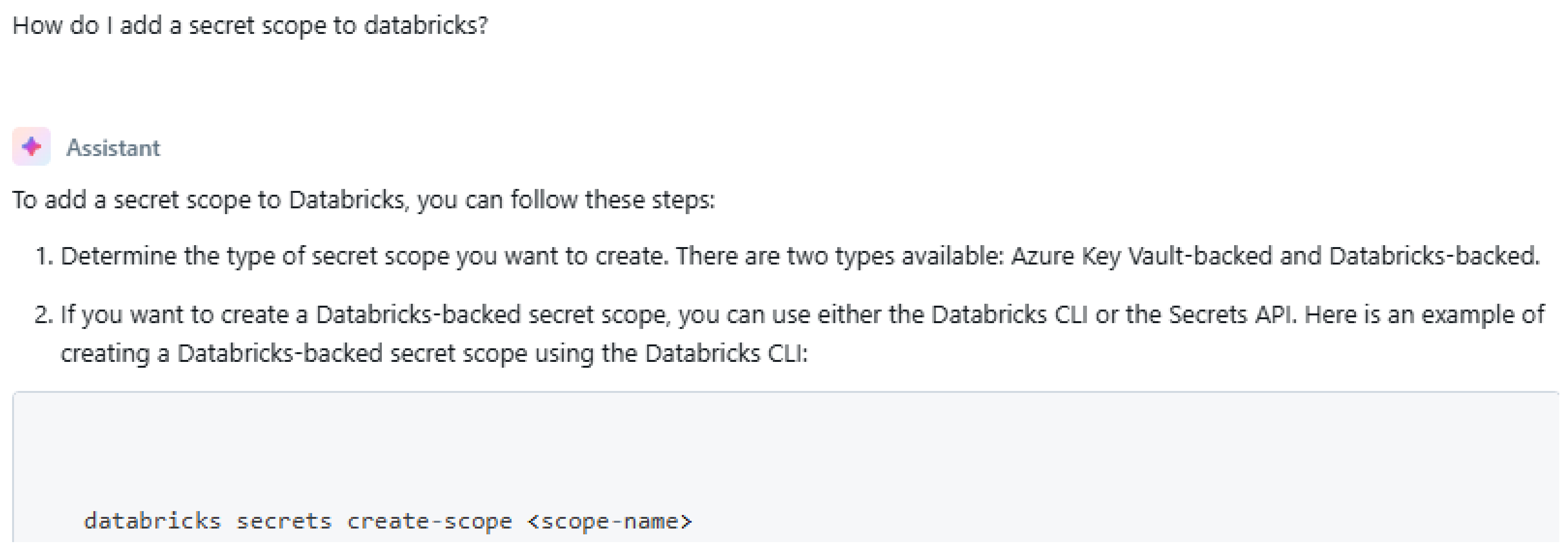
Some Useful Links...

Many administration tasks can be achieved via the CLI, but it's not easy to automate tasks using it, without having a dedicated machine set up. In these cases, we have a couple of other options:

Databricks REST API
https://docs.databricks.com/api/azure/workspace/introduction

Databricks SDK (Python, Java, Go)
https://docs.databricks.com/en/dev-tools/sdk-python.html

Asking the Databricks assistant



How do I add a secret scope to databricks?

Assistant

To add a secret scope to Databricks, you can follow these steps:

1. Determine the type of secret scope you want to create. There are two types available: Azure Key Vault-backed and Databricks-backed.
2. If you want to create a Databricks-backed secret scope, you can use either the Databricks CLI or the Secrets API. Here is an example of creating a Databricks-backed secret scope using the Databricks CLI:

```
databricks secrets create-scope <scope-name>
```

Demo: Secret Management

Command Line Interface (CLI)

Installing the CLI

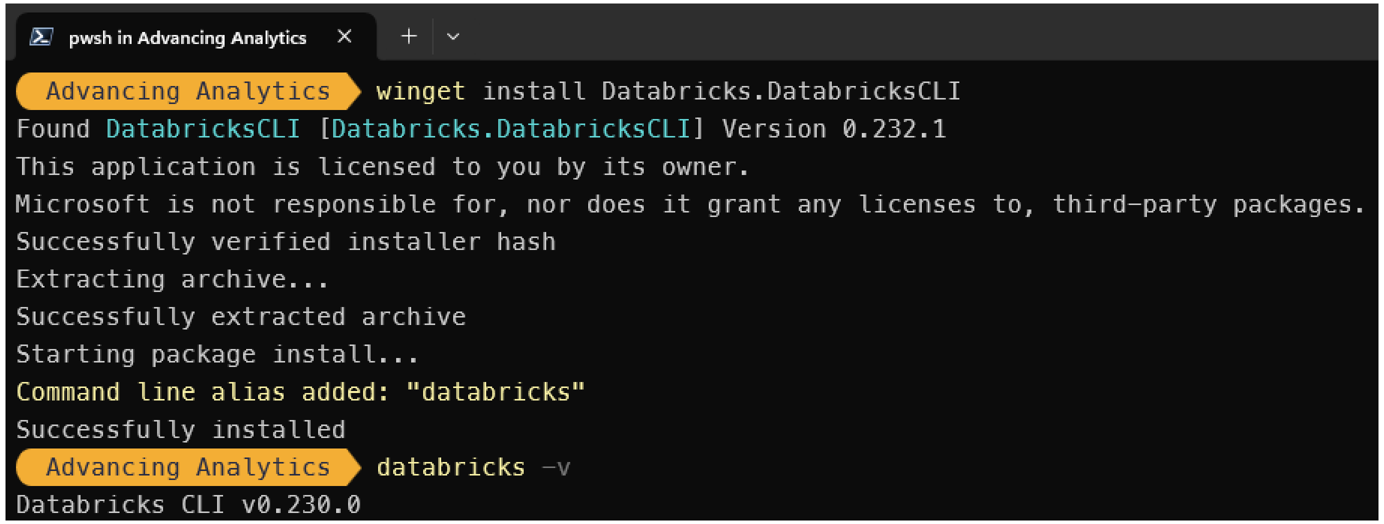
1. You can install in a variety of ways (see https://docs.databricks.com/en/dev-tools/cli/install.html)

2. On Windows, using Winget

3. From the terminal, run the following command:

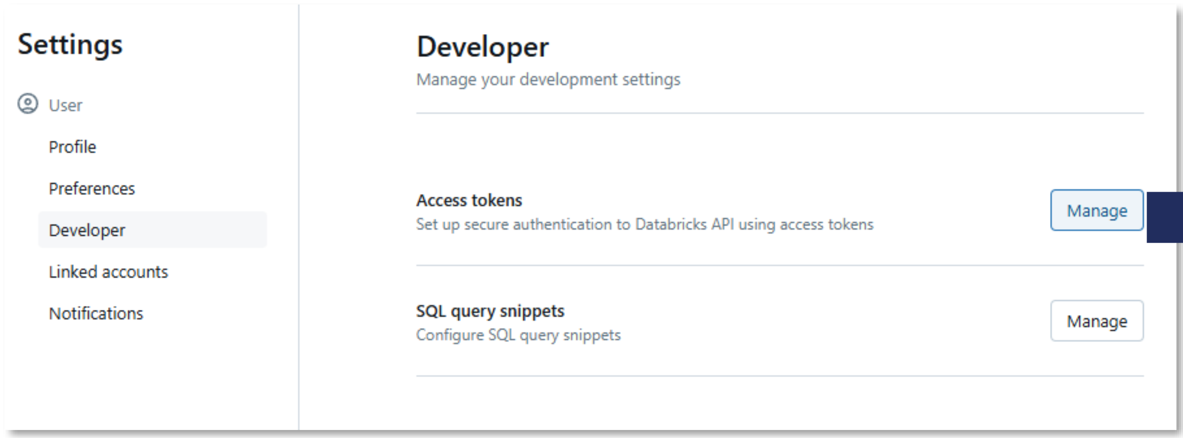
```
winget install Databricks.DatabricksCLI
```

4. From the terminal, type "databricks" to run the CLI



```
 Advancing Analytics  winget install Databricks.DatabricksCLI
Found DatabricksCLI [Databricks.DatabricksCLI] Version 0.232.1
This application is licensed to you by its owner.
Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.
Successfully verified installer hash
Extracting archive...
Successfully extracted archive
Starting package install...
Command line alias added: "databricks"
Successfully installed
 Advancing Analytics  databricks -v
Databricks CLI v0.230.0
```

Generate Workspace Token



Settings

User

Profile

Preferences

Developer

Linked accounts

Notifications

Developer

Manage your development settings

Access tokens

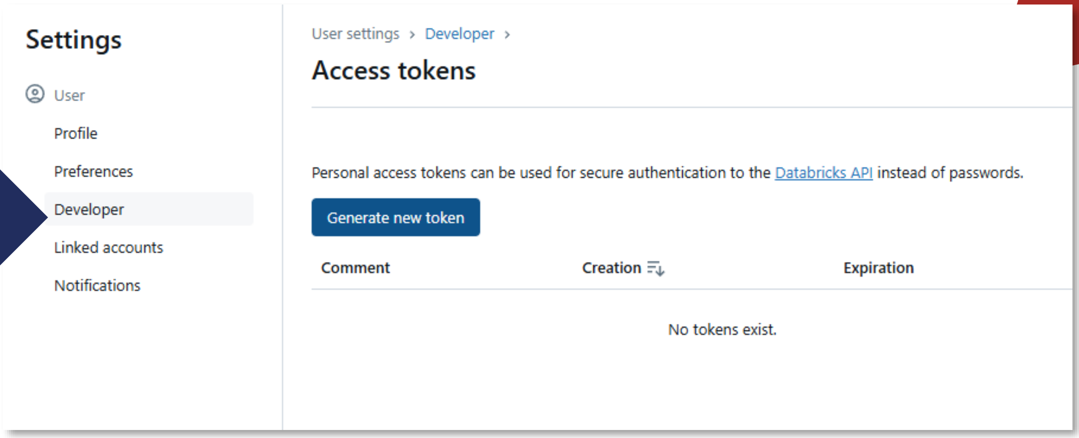
Set up secure authentication to Databricks API using access tokens

Manage

SQL query snippets

Configure SQL query snippets

Manage



Settings

User

Profile

Preferences

Developer

Linked accounts

Notifications

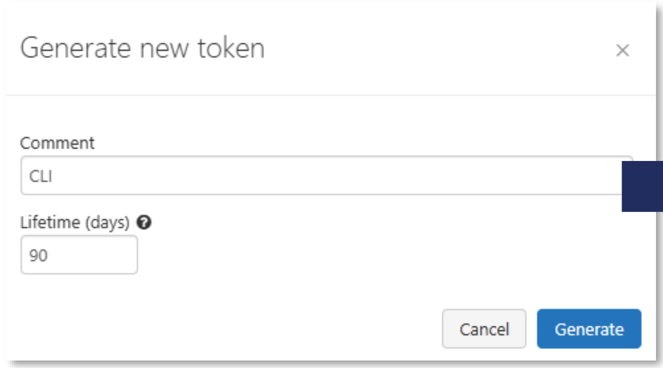
User settings › Developer ›

Access tokens

Personal access tokens can be used for secure authentication to the Databricks API instead of passwords.

Generate new token

Comment	Creation	Expiration
No tokens exist.		

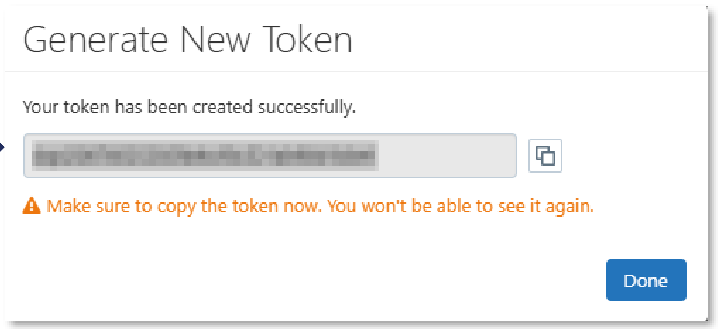


Generate new token

Comment: CLI

Lifetime (days): 90

Cancel | Generate



Generate New Token

Your token has been created successfully.

[redacted]

⚠ Make sure to copy the token now. You won't be able to see it again.

Done

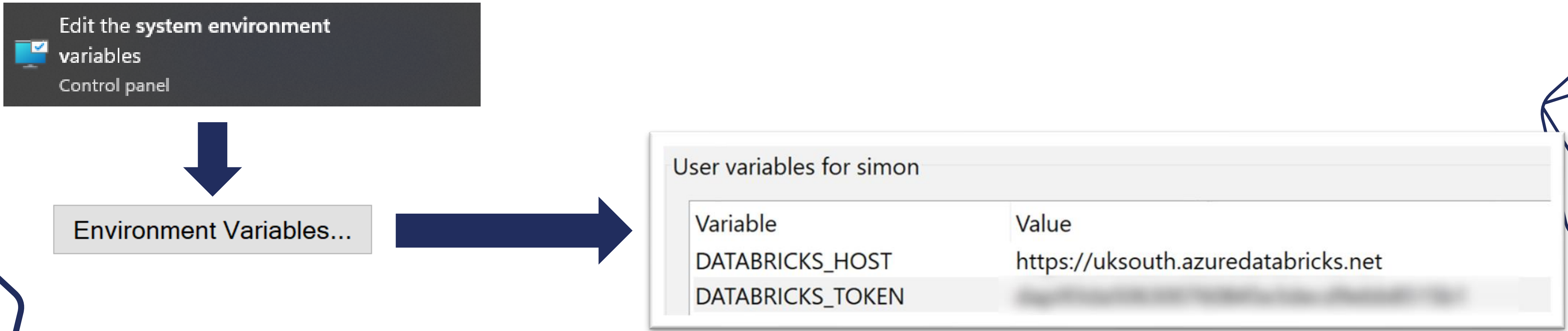
ADVANCING ANALYTICS

Configuring the CLI

To configure the CLI manually, run the databricks configure command from the terminal:

```
C:\> databricks configure --token
Databricks host:
https://uksouth.azuredatabricks.net
Personal access token: [Token]
```

Alternatively, you can use System Environment Variables!



User variables for simon

Variable	Value
DATABRICKS_HOST	https://uksouth.azuredatabricks.net
DATABRICKS_TOKEN	[redacted]

CLI Profiles

If you have multiple databricks workspaces to administer, you can define CLI *"profiles"* as an alias for each different workspace

```
C:\> databricks configure --profile AdvAnalytics
Databricks Host: https://uksouth.azuredatabricks.net
Personal access token: [Token]



# We can then run any CLI command using and input the --profile as a parameter, without having
to change any configuration

C:\> databricks clusters list --profile AdvAnalytics
ID                     Name         State
0625-124059-taupe49    Processing   TERMINATED
```

Common CLI Tasks

List all clusters in a workspace:

```
databricks clusters list
```

Start a cluster:

```
databricks clusters start --clusterid
```

List folders from the Workspace:

```
databricks workspace list "/"
```

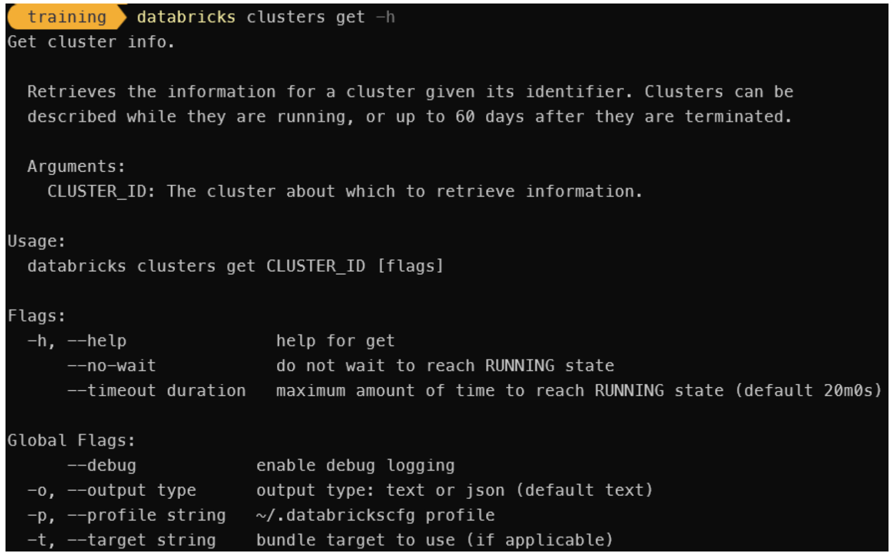
Upload directory of notebooks to a workspace:

```
databricks workspace import_dir SOURCE_PATH TARGET_PATH
```

Help!

You can write any command followed by -h to return the help text for that command – if you're not sure what a specific command does, just type it with -h!

```
databricks clusters get -h
```



```
 training  databricks clusters get -h
Get cluster info.

  Retrieves the information for a cluster given its identifier. Clusters can be
  described while they are running, or up to 60 days after they are terminated.

  Arguments:
    CLUSTER_ID: The cluster about which to retrieve information.

Usage:
  databricks clusters get CLUSTER_ID [flags]

Flags:
  -h, --help                 help for get
      --no-wait              do not wait to reach RUNNING state
      --timeout duration     maximum amount of time to reach RUNNING state (default 20m0s)

Global Flags:
      --debug              enable debug logging
  -o, --output type        output type: text or json (default text)
  -p, --profile string     ~/.databrickscfg profile
  -t, --target string      bundle target to use (if applicable)
```

VSCode & Databricks Connect

Common Scenarios

We want to be able to run our notebooks locally against our Databricks Environment

We want to be able to debug from any IDE

We want to be able to embed Databricks into custom applications

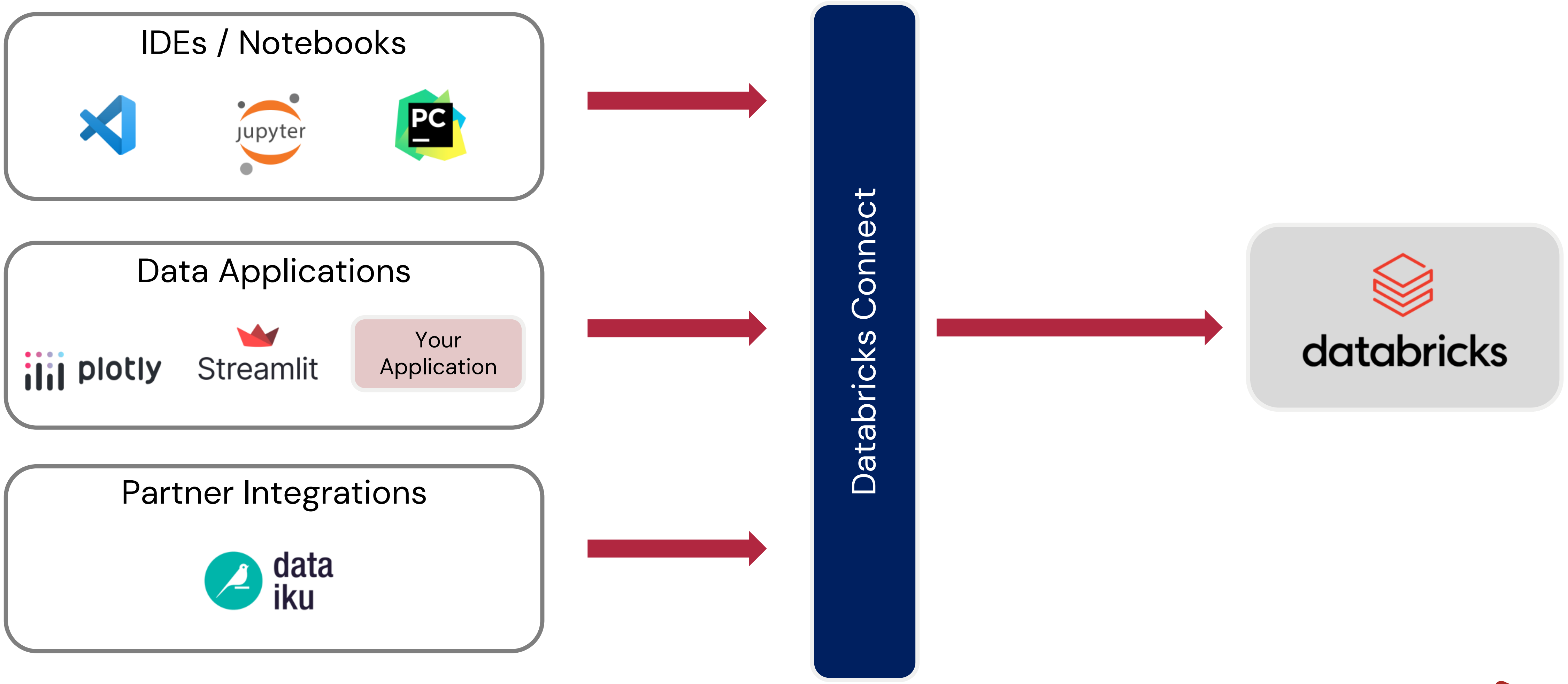
ADVANCING ANALYTICS

What is Databricks Connect?

What is Databricks Connect

- Allows you to connect popular IDEs and custom applications to Databricks clusters
- Allows you to write custom Databricks Python code using Databricks Python Libraries: SDK and Databricks Connect
- Allows you to write code using Spark APIs and run them remotely on a Databricks cluster instead of in the local Spark session

What is Databricks Connect



How to Utilize Databricks Connect

There are two ways to Utilize Databricks Connect:

- Python Libraries (directly in code)
- VSCode Extension (IDE)

Python Libraries - Install

- Install Databricks CLI: https://docs.databricks.com/en/dev-tools/cli/install.html
- Authenticate and set profile name (used in your code):

```
databricks configure --profile test
```

- Install Python Packages:

```
pip install pyspark == [version]
pip install databricks.connect == [version]
pip install databricks.sdk == [version]
```

The Databricks Connect major and minor **package version** should match your Databricks Cluster **Runtime version**, if not you will experience errors

Python Libraries – Configure & Usage

Once installed it is easy to quickly start writing Python, and have it integrated with your Databricks workspace of choice

Import Libraries

```
from dash import Dash, dash_table
from databricks.connect.session import DatabricksSession as SparkSession
from databricks.sdk.core import Config

config = Config(profile=“[local Databricks Profile]",
cluster_id=“[cluster id]")

spark = SparkSession.builder.sdkConfig(config).getOrCreate()
df = spark.table("samples.nyctaxi.trips").limit(10)
```

Set Configuration

Read Data

ADVANCING ANALYTICS

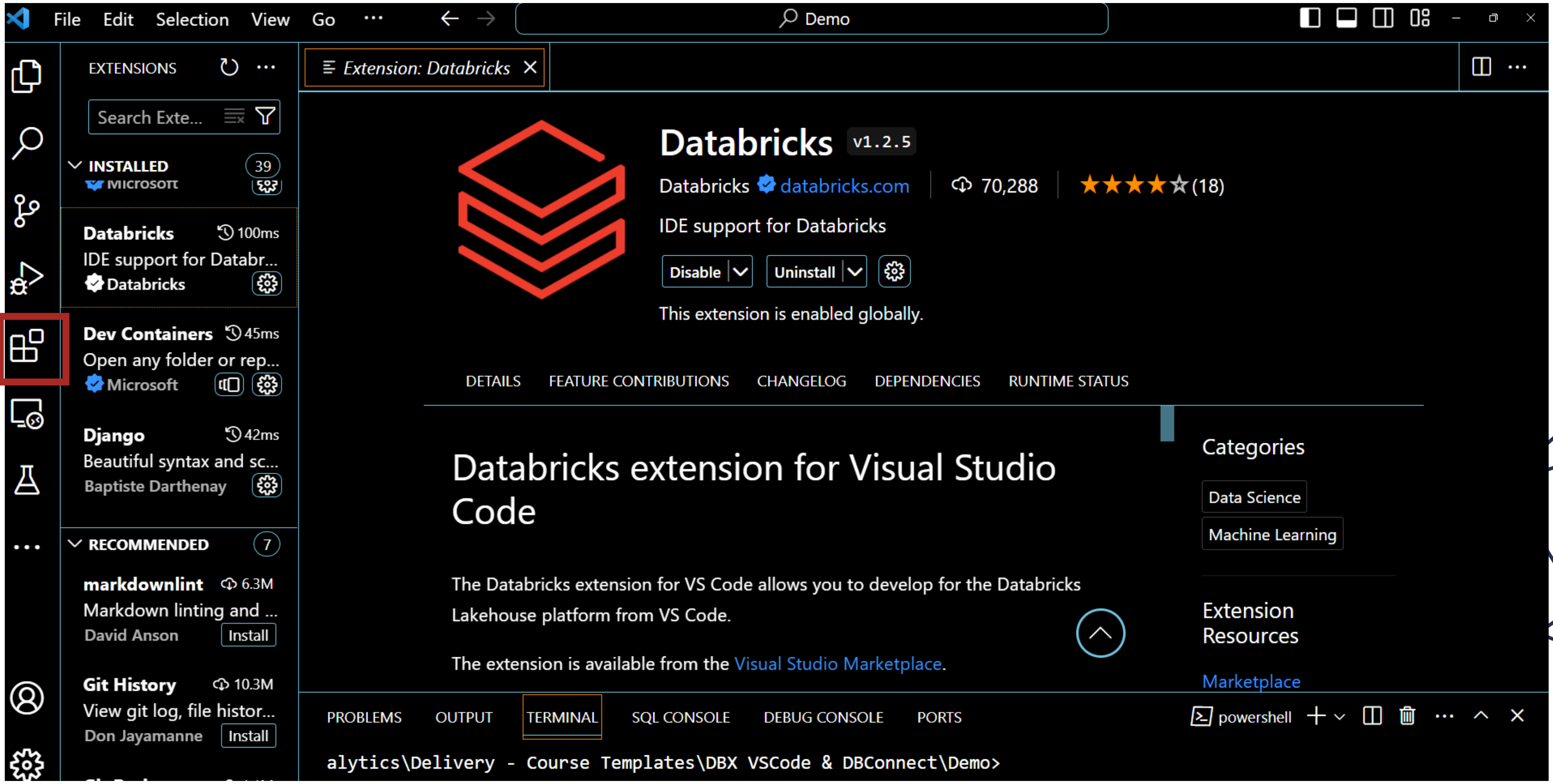
VSCode Extension

- Exclusive to VSCode, other IDEs require the use of the Python packages instead
- Allows developers to smoothly integrate their Databricks Workspace with local Folders/Repos
- Creates a synced folder in the Databricks Workspace (within Repos)
- Allows developers to test Notebooks locally

ADVANCING ANALYTICS

VSCode Extension - Install

Extensions tab



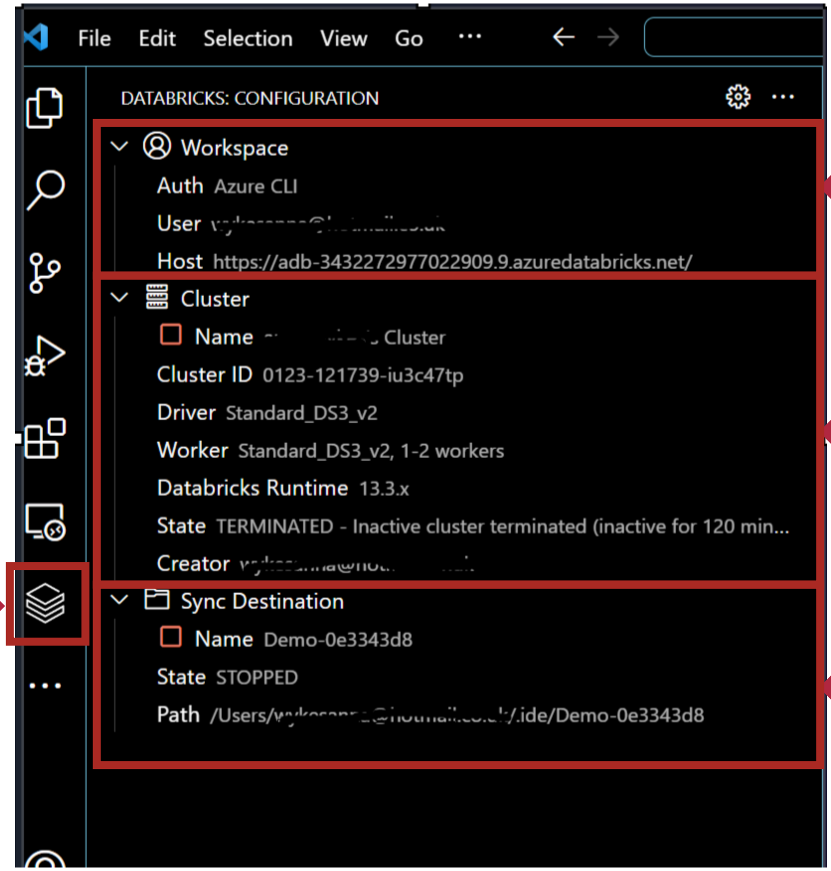
Databricks extension for Visual Studio Code

The Databricks extension for VS Code allows you to develop for the Databricks Lakehouse platform from VS Code.

The extension is available from the Visual Studio Marketplace.

```
alytics\Delivery - Course Templates\DBX VSCode & DBConnect\Demo>
```

VSCode Extension – Configure



DATABRICKS: CONFIGURATION

Workspace
Auth Azure CLI
User [redacted]
Host https://adb-3432272977022909.9.azuredatabricks.net/

Cluster
Name [redacted] Cluster
Cluster ID 0123-121739-iu3c47tp
Driver Standard_DS3_v2
Worker Standard_DS3_v2, 1-2 workers
Databricks Runtime 13.3.x
State TERMINATED - Inactive cluster terminated (inactive for 120 min...
Creator [redacted]

Sync Destination
Name Demo-0e3343d8
State STOPPED
Path /Users/[redacted]/.ide/Demo-0e3343d8

Databricks Option Added

Set Workspace Details

Set Cluster Details

Set Sync Destination

ADVANCING ANALYTICS

Demo: Databricks Connect

Recap

Recap

- Use **Databricks Git integration** to version control our notebooks
- Install **python package dependencies** at the cluster leve
- Code distribution make code more **accessible, maintainable** & **scalable**
- Python Wheels is to **organize and distribute code** quickly as a **single object**
- Databricks Asset Bundles (DABs) allows you to package the **entire Databricks project**
- Manage **secrets and secrets scopes**, through the CLI and using dbutils.
- Use the **Databricks CLI** for regular tasks such as **managing clusters and workspaces**
- Use **Databricks Connect** to work with Databricks **notebooks in VSCode**

ADVANCING ANALYTICS

ADVANCING
ANALYTICS