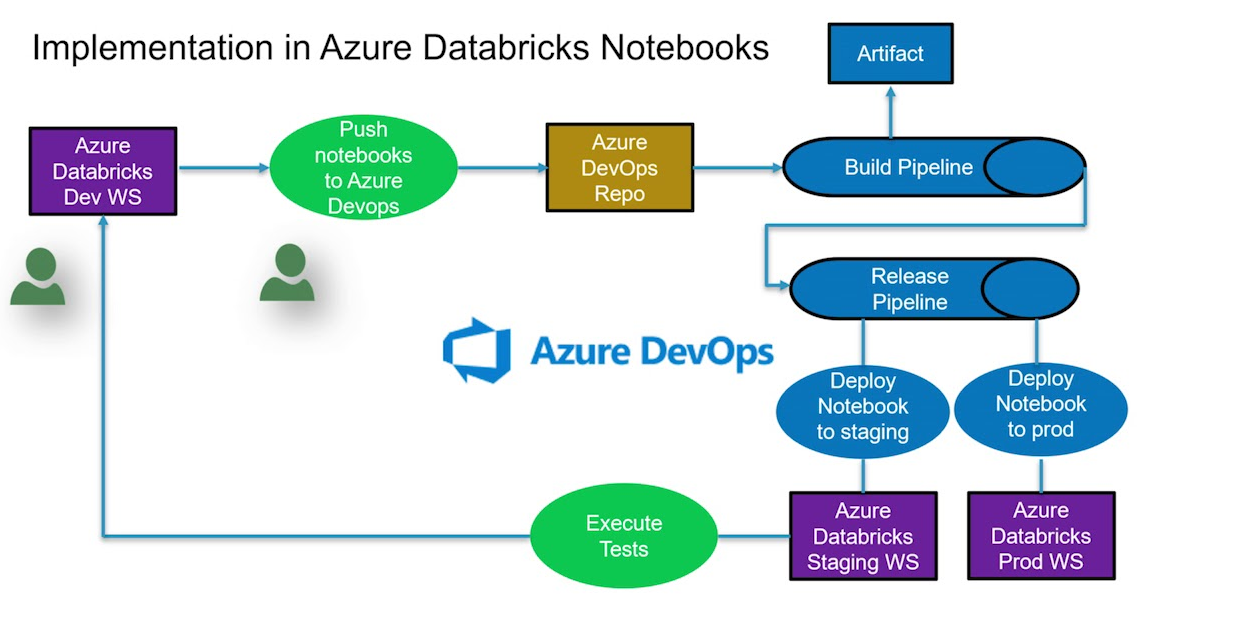
**CI/CD on Azure Databricks using Azure DevOps**

Continuous integration and continuous delivery (CI/CD) refers to the process of developing and delivering software in short, frequent cycles through the use of automation pipelines.

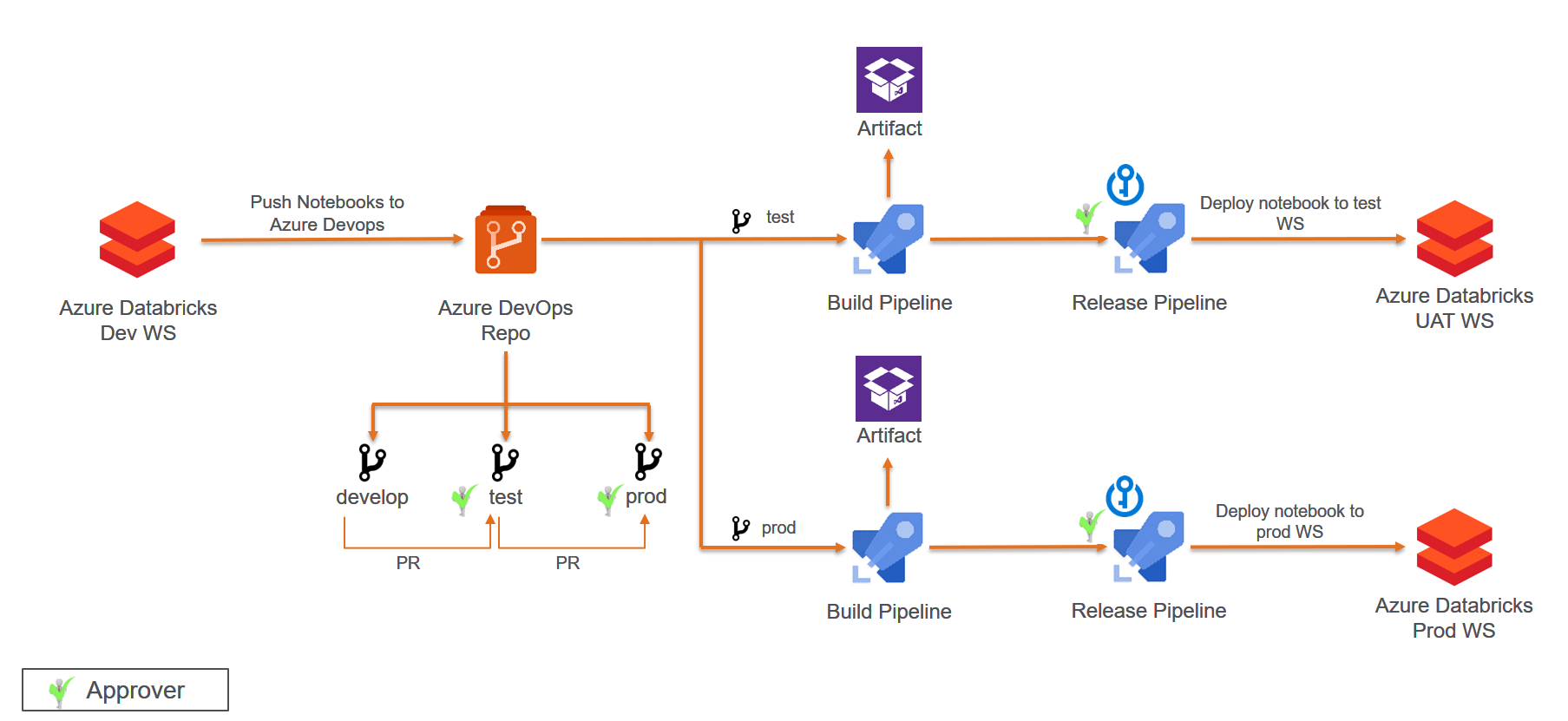


**Continuous integration:**

1. Code
   1. Develop code and unit tests in an Azure Databricks notebook or using an external IDE.
   2. Manually run tests.
   3. Commit code and tests to a git branch.
2. Build
   1. Gather new and updated code and tests.
   2. Run automated tests.
   3. Build libraries and non-notebook Apache Spark code.
3. Release: Generate a release artifact.

**Continuous delivery:**

1. Deploy
   1. Deploy notebooks.
   2. Deploy libraries.
2. Test: Run automated tests and report results.
3. Operate: Programmatically schedule data engineering, analytics, and machine learning workflows.



* Develop and commit your code in develop branch
* Push code from develop branch to → test branch → master branch
* Deploy Notebooks in different environments Dev →Test →Prod using CI/CD pipelines in Azure DevOps

**Move data in Databricks**

There are different way to move the data from one layers to other layers in Databricks.

**Method1: Using Databricks CLI**

The DBFS command-line interface (CLI) uses the DBFS API to expose an easy to use command-line interface to DBFS. Using this client, you can interact with DBFS using commands similar to those you use on a Unix command line. For example:

# List files in DBFS

dbfs ls

# Put local file ./apple.txt to dbfs:/apple.txt

dbfs cp ./apple.txt dbfs:/apple.txt

# Get dbfs:/apple.txt and save to local file ./apple.txt

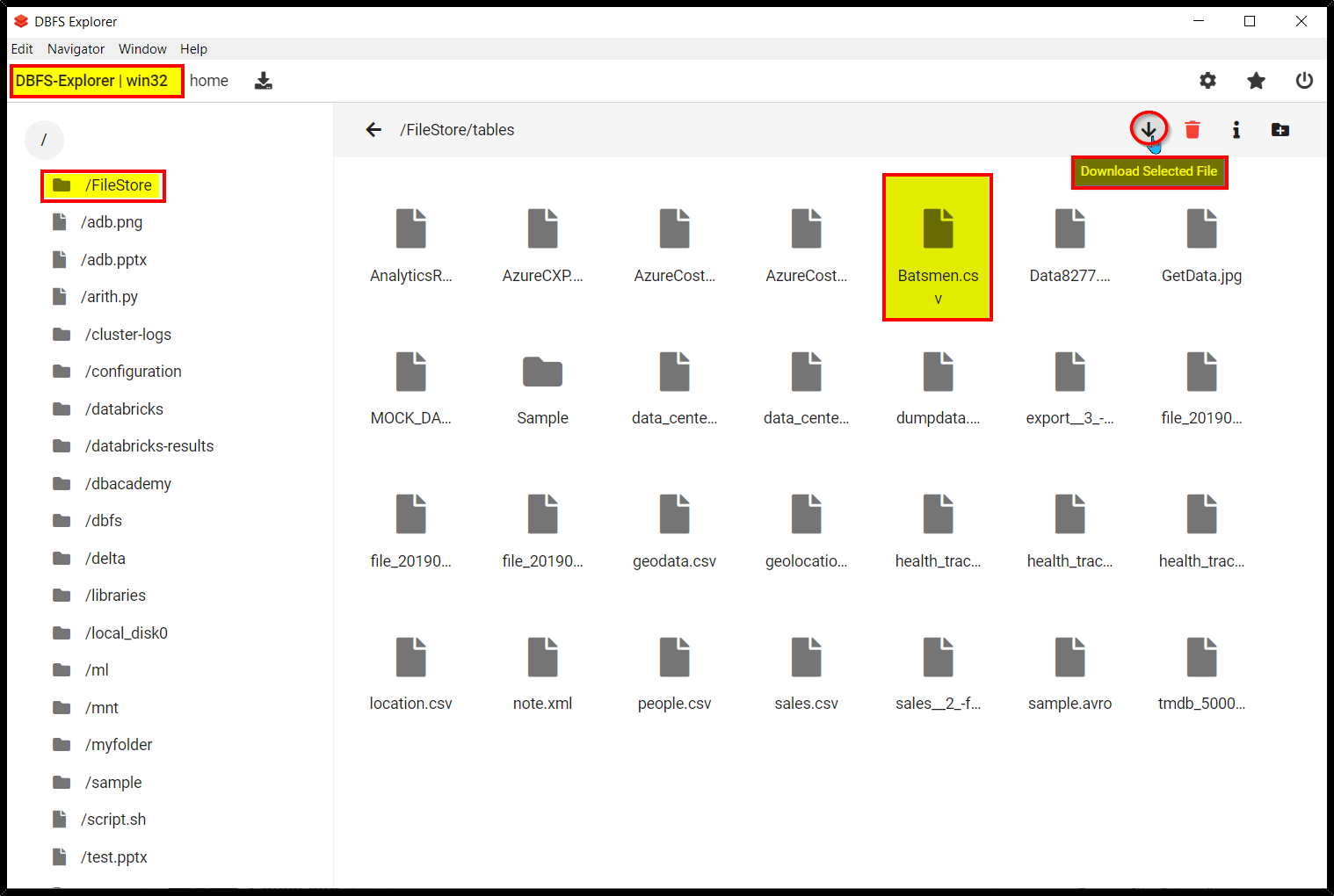
dbfs cp dbfs:/apple.txt ./apple.txt

# Recursively put local dir ./banana to dbfs:/banana

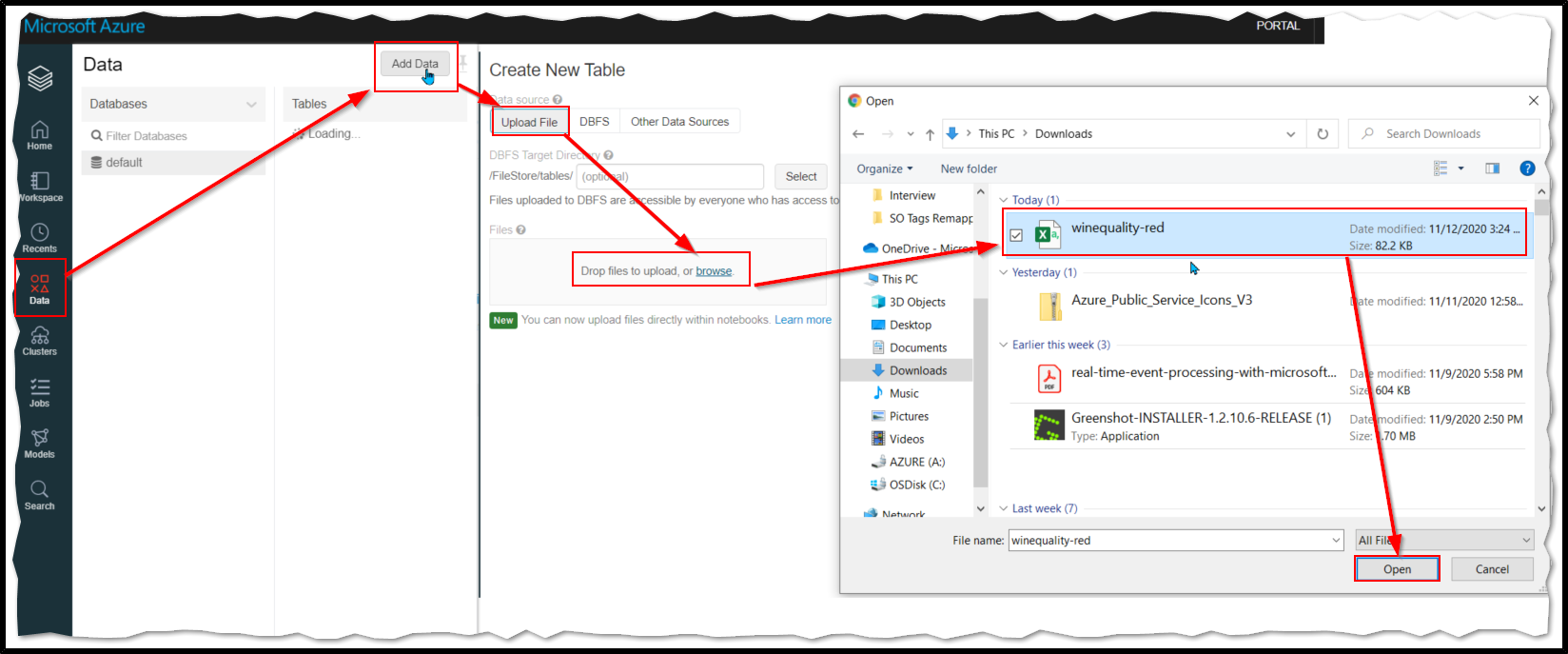
dbfs cp -r ./banana dbfs:/banana

**Method2: Using third-party tool named DBFS Explorer**

[DBFS Explorer](https://datathirst.net/projects/dbfs-explorer) was created as a quick way to upload and download files to the Databricks filesystem (DBFS). This will work with both AWS and Azure instances of Databricks. You will need to create a bearer token in the web interface in order to connect.



**Method3: Using the Azure Databricks portal.**



**https://databricks.com/blog/2020/03/06/connect-90-data-sources-to-your-data-lake-with-azure-databricks-and-azure-data-factory.html**