

Create a CI/CD process with Azure DevOps

Note: In this reading you can see the steps involved in the process of creating a CI/CD process with Azure DevOps.

Spark Queries for Cosmos DB Core (SQL) API

Now that we have covered the definition of CI/CD, let's put those principles in practice by following the steps below.

Requirements

The following services are needed

- Azure DevOps
- Two Azure Databricks Workspaces: DEV and PROD

End goal scenario

1. Develop a notebook in DEV workspace

2. Commit this to Azure DevOps (Master branch of the repo)

3. Once the commit is successful, this notebook will automatically be deployed into PROD workspace

Note: *Ideally, a user wants to create a feature branch first and work there. Once it's reviewed by peers using 'pull request', this feature branch can be committed to master branch. Then, it'll be automatically deployed into higher environment like Staging for production testing or PROD directly.*

Unit Pre-requisites

Microsoft Azure Account: You will need a valid and active Azure account for the Azure labs. If you do not have one, you can sign up for a [free trial](#)

- If you are a Visual Studio Active Subscriber, you are entitled to Azure credits per month. You can refer to this [link](#) to find out more, including how to activate and start using your monthly Azure credit.
- If you are not a Visual Studio Subscriber, you can sign up for the FREE [Visual Studio Dev Essentials](#) program to create Azure free account.

Create the required resources

To complete this lab, you will need to deploy two Azure Databricks workspaces in your Azure subscription.

Deploy a DEV Azure Databricks workspace

1. Click the following button to open the Azure Resource Manager template in the Azure portal. [Deploy Databricks from the Azure Resource Manager Template](#)
2. Provide the required values to create your Azure Databricks workspace:
 - **Subscription:** Choose the Azure Subscription in which to deploy the workspace.
 - **Resource Group:** Leave at Create new and provide a name for the new resource group.
 - **Location:** Select a location near you for deployment. For the list of regions supported by Azure Databricks, see [Azure services available by region](#).
 - **Workspace Name:** Provide a name for your workspace that includes the word **DEV** in the name.
 - **Pricing Tier:** Ensure **premium** is selected.
3. Accept the terms and conditions.
4. Select Purchase.
5. The workspace creation takes a few minutes. During workspace creation, move on to the next step to deploy the PROD Azure Databricks workspace.

Deploy a PROD Azure Databricks workspace

1. Click the following button to open the Azure Resource Manager template in the Azure portal. [Deploy Databricks from the Azure Resource Manager Template](#)
2. Provide the required values to create your Azure Databricks workspace:
 - **Subscription:** Choose the Azure Subscription in which to deploy the workspace.
 - **Resource Group:** Leave at Create new and provide a name for the new resource group.
 - **Location:** Select a location near you for deployment. For the list of regions supported by Azure Databricks, see [Azure services available by region](#).
 - **Workspace Name:** Provide a name for your workspace that includes the word **PROD** in the name.
 - **Pricing Tier:** Ensure **premium** is selected.
3. Accept the terms and conditions.
4. Select Purchase.
5. The workspace creation takes a few minutes. During workspace creation, the portal displays the Submitting deployment for Azure Databricks tile on the right side. You may need to scroll

right on your dashboard to see the tile. There is also a progress bar displayed near the top of the screen. You can watch either area for progress.

Create a cluster

Optional: *Creating a cluster is optional for this module. It is only needed if you plan on running the sample notebook. The notebook is only used for demonstration purposes of the CI/CD process. There is no requirement to run its cells to complete this unit.*

1. When your Azure Databricks workspace creation is complete, select the link to go to the resource.
2. Select **Launch Workspace** to open your Databricks workspace in a new tab.
3. In the left-hand menu of your Databricks workspace, select **Clusters**.
4. Select **Create Cluster** to add a new cluster.

Create Cluster

New Cluster

CancelCreate Cluster

0 Workers: 0.0 GB Memory, 0 Cores, 0 DBU

1 Driver: 14.0 GB Memory, 4 Cores, 0.75 DBU ?

Cluster Name



Cluster Mode ?



Pool ?



Databricks Runtime Version ?

[Learn more](#)New

This Runtime version supports only Python 3.

Autopilot Options

☒ Terminate after minutes of inactivity ?

Node Type ?

14.0 GB Memory, 4 Cores, 0.75 DBU



► Advanced Options

The create cluster page.

5. Enter a name for your cluster. Use your name or initials to easily differentiate your cluster from your coworkers.

6. Select the **Cluster Mode: Single Node**.

7. Select the **Databricks RuntimeVersion: Runtime: 7.3 LTS (Scala 2.12, Spark 3.0.1)**.

8. Under **Autopilot Options**, leave the box **checked** and in the text box enter **45**.

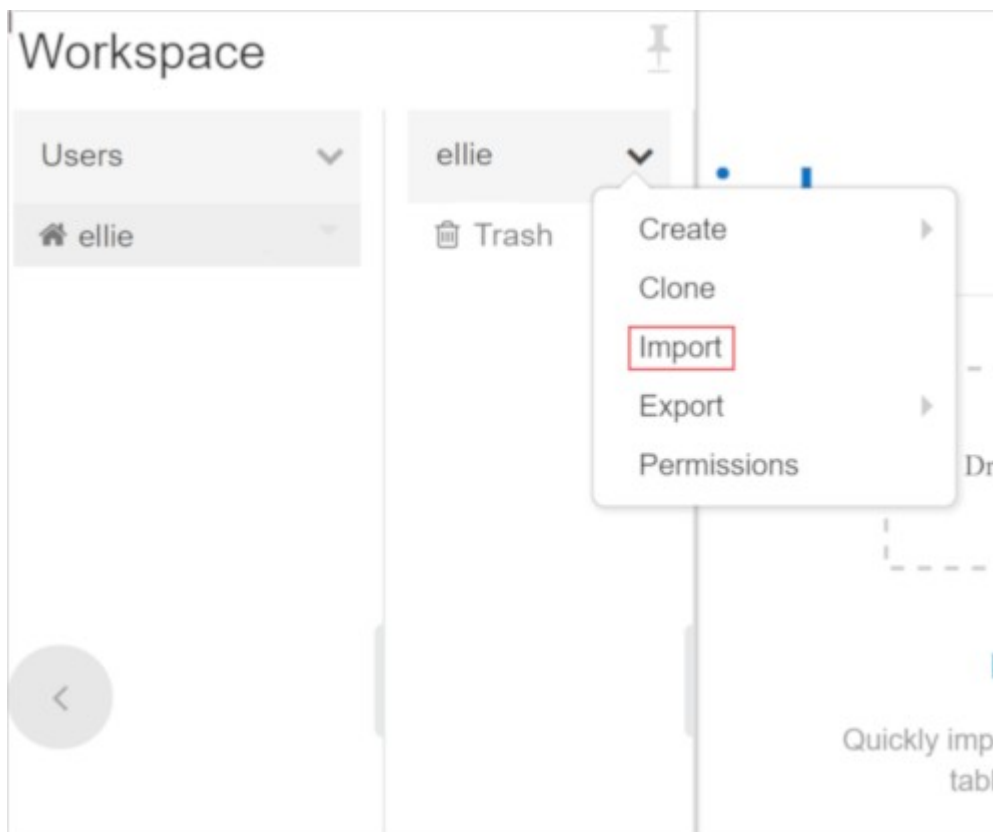
9. Select the **Node Type: Standard_DS3_v2**.

10. Select **Create Cluster**.

Repeat the above steps for the PROD workspace.

Clone the Databricks archive in the DEV workspace

1. If you do not currently have your Azure Databricks workspace open: in the Azure portal, navigate to your deployed Azure Databricks workspace and select **Launch Workspace**.
2. In the left pane, select **Workspace > Users**, and select your username (the entry with the house icon).
3. In the pane that appears, select the arrow next to your name, and select **Import**.



The menu option to import the archive.

4. In the **Import Notebooks** dialog box, select the URL and paste in the following URL:

<https://github.com/solliancenet/microsoft-learning-paths-databricks-notebooks/blob/master/data-engineering/DBC/13-CI-CD-with-Azure-Devops.dbc?raw=true>

5. Select **Import**.

High Level Steps

At a high level, setting up CI/CD on Azure Databricks with Azure DevOps consists of 4 steps:

1. Setting up Azure DevOps Repo
2. Have your Azure Workspace and notebook configured to use Azure DevOps
3. Azure DevOps - Create a build pipeline (CI)
4. Azure DevOps - Create a release pipeline (CD)

Step 1: Set up Azure DevOps Repo

1. Go to <https://aex.dev.azure.com>. Make sure that you are logged in under the correct account.
2. If your account has access to multiple Azure subscriptions, make sure that you're in the same directory as your Databricks workspaces.
3. Create a project.
4. Complete the new project form by providing a unique project name. Set visibility to **Private**, expand the Advanced section on the bottom, then select **Git** for version control. The work item process can remain at its default setting.

Create new project

Project name *

Databricks_CICD

✓

Description

Demo

G

Visibility

Public ⓘ

Anyone on the internet can view the project. Certain features like TFVC are not supported.

Private

Only people you give access to will be able to view this project.

Public projects are disabled for your organization. You can turn on public visibility with [organization policies](#).

^ Advanced

Version control ⓘ

Git

▼

Work item process ⓘ

Agile

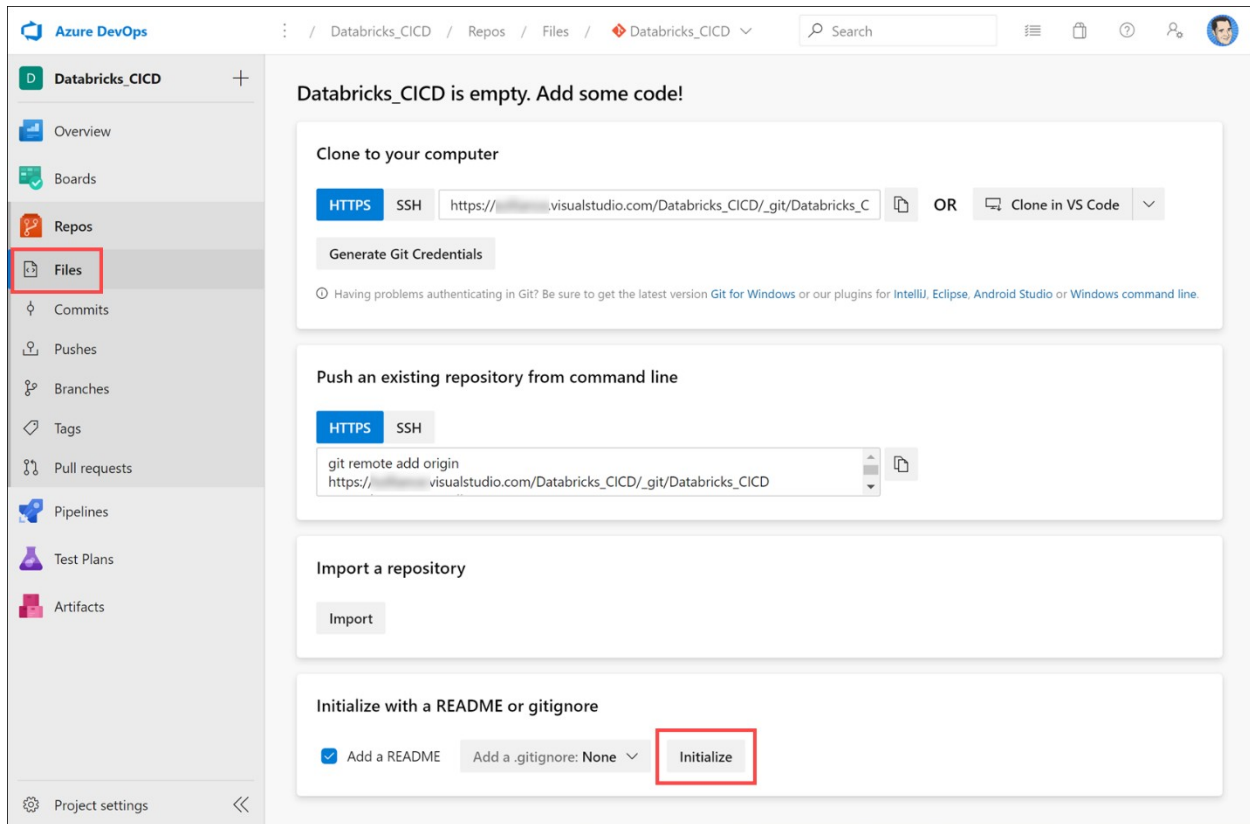
▼

Cancel

Create

Create new project form.

5.Once Project is created, go to **Repos**, then **Initialize** your repo.

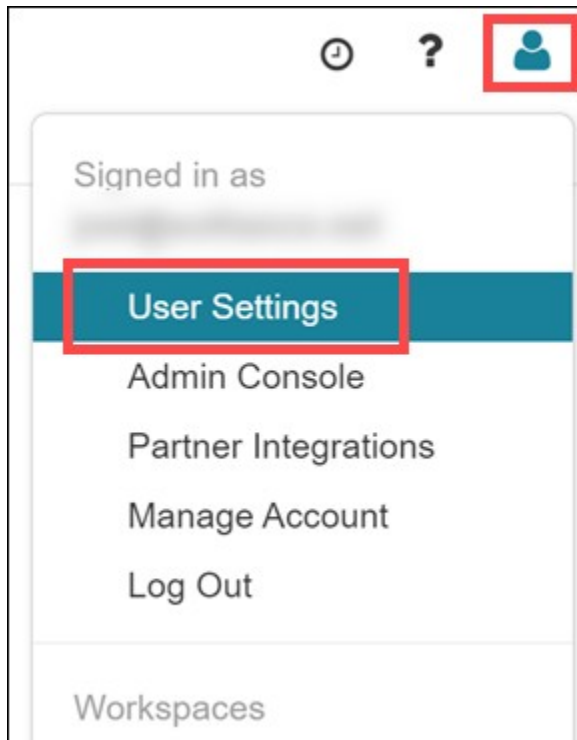


Initialize repo.

Step 2: Set up Azure Databricks Workspace and your notebook

Go to DEV Azure Databricks workspace and setup your Git integration provider to Azure DevOps Services by following these steps:

1. Select the user icon on the top-right of the workspace, then select **User Settings**.



User Settings menu link.

2. Select the **Git Integration** tab. Make certain the Git provider is set to **Azure DevOps Services**.

User Settings

[Access Tokens](#) **Git Integration** [Notebook Settings](#)

Git integration

Databricks supports notebook version control integration with GitHub, Bitbucket Cloud, or Azure DevOps Services. To connect to git repositories and make commits, we need a personal access token (for GitHub) or an app password (for Bitbucket Cloud). Azure DevOps Services requires no extra authentication.

Generating tokens

To generate a GitHub personal access token, follow the [GitHub documentation](#). When using GitHub, the token must have the "repo" permission.

To generate a Bitbucket Cloud app password, follow the [Bitbucket Cloud documentation](#). When using Bitbucket Cloud, the app password must have "read" and "write" permission under repository.

For more information on Git integration, see the Databricks documentation on [GitHub integration](#), [Bitbucket Cloud integration](#), or [Azure DevOps Services integration](#).

Git provider

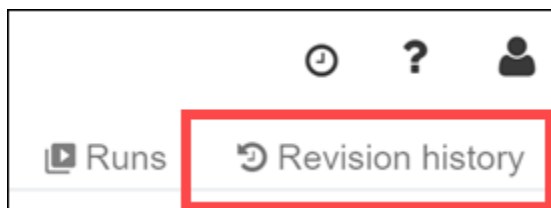
Azure DevOps Services ▼

Save

The Azure DevOps Services git provider is selected.

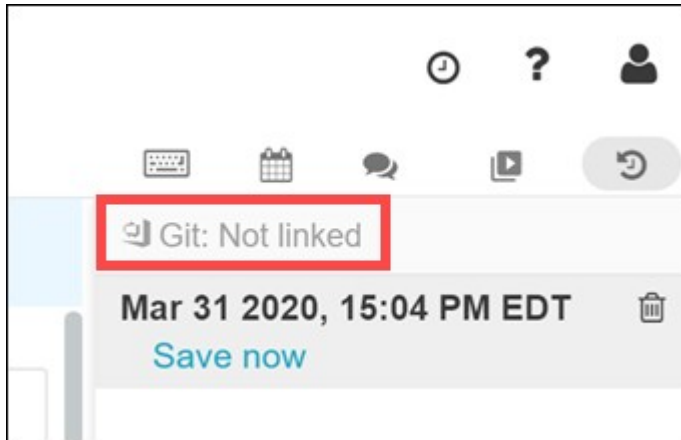
3. Navigate to the **13-CI-CD-with-Azure-Devops** folder in your Workspace. If you imported the Databricks notebook for this lab, this is the directory that was created during that process.

4. Open the **Distinct-Articles** notebook. Select **Revision history** on the right-hand side above the notebook.



The Revision history link is highlighted.

5. Select **Git: Not linked** at the top of the revision history. This opens a Git configuration dialog.



The Git configuration dialog

6. In Git Preferences, use the URL scheme **https://dev.azure.com/<myOrg>/<myProject>/_git/<myRepo>** in the **Link** field to link Azure DevOps and Azure Databricks to the same Azure AD tenant.

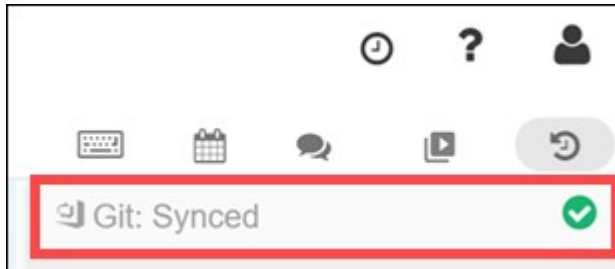
A screenshot of the 'Git Preferences' dialog box. At the top, it has the title 'Git Preferences'. Below the title, there are two radio buttons for 'Status': 'Link' (selected) and 'Unlink'. Below this, there is a red-bordered box containing a question mark icon and the text 'Link', followed by a text input field containing the URL 'https://dev.azure.com/[redacted]/Databricks_CICD/_git/Databrickl'. Below the red box, there is a 'Branch' dropdown menu set to 'master'. Below that, there is a 'Path in Git Repo' text input field containing 'notebooks/Users/joel@[redacted]/13-CI-CD-with-Azure-Dev'. At the bottom, there is a tip: 'Tip: You can also import/export multiple notebooks or an entire folder through Workspace API to your computer and check-in to your favorite version control system.' and two buttons: 'Close' and 'Save'.

The Git Preferences dialog is displayed.

If your Azure DevOps organization is **org.visualstudio.com**, open **dev.azure.com** in your browser and navigate to your repository. Copy the URL from the browser and paste that URL in the Link field. **Note:** *format of URL is:*

`http://dev.azure.com/<myOrg>/<myProject>/_git/<myRepo>` which differs from the URL of the repo displayed in Azure DevOps:
`http://dev.azure.com/<myOrg>/_git/<myRepo>`

7. Select **Save** to finish linking your notebook. You should see that the Git repo is now synced.



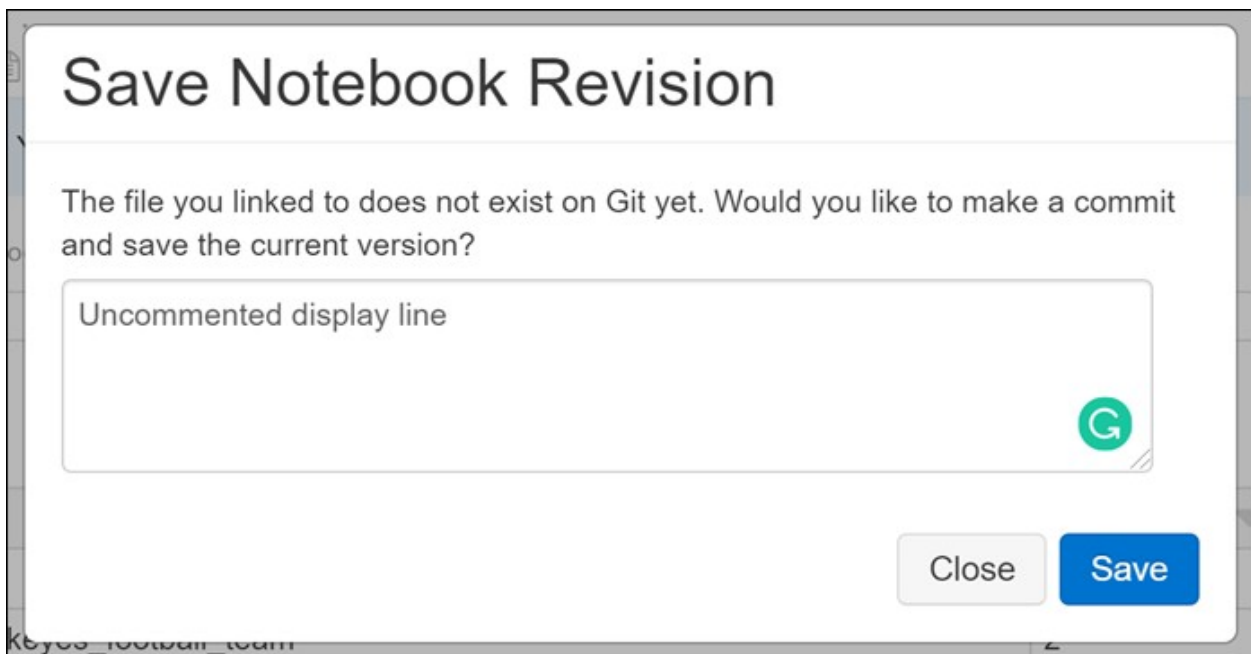
Git: Synced

8. Close the Revision History sidebar, then scroll down to the bottom of the notebook. Uncomment cell #9 so it looks like the following:

```
display(df.select("article"))
```

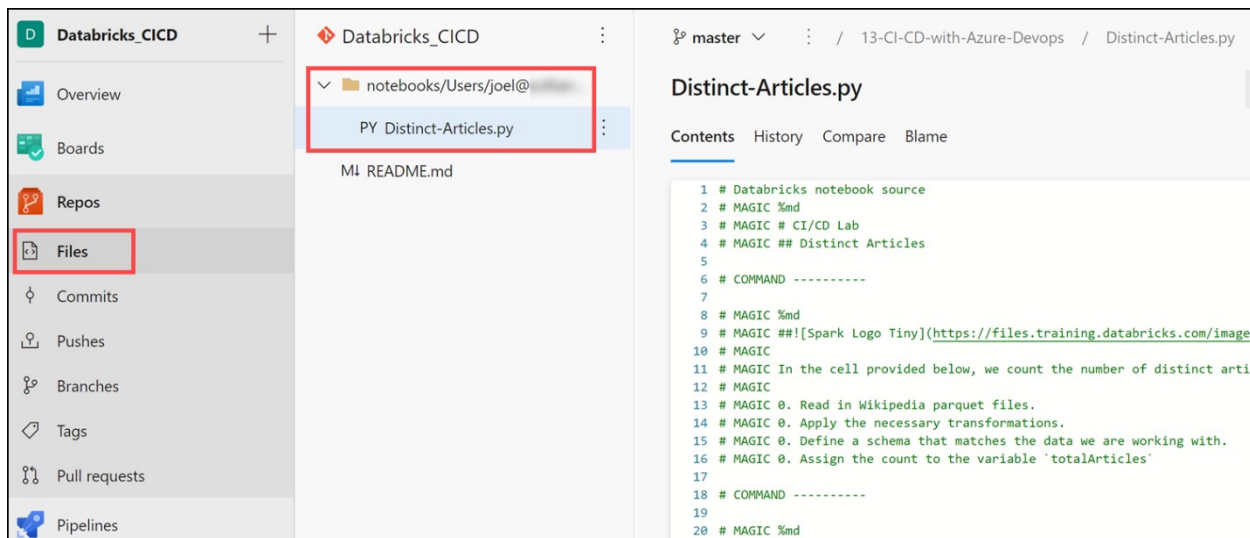
1

9. Open the Revision History sidebar once again. When prompted to **Save Notebook Revision**, enter a revision description, then **Save**.



Save Notebook Revision dialog.

10. Go back to your repo in Azure DevOps and refresh the files list. You should see your notebook in the repo. If you look under Commits, you should see your commit message as well.



The committed notebook is displayed in Azure DevOps.

Step 3: Retrieve Access Token from PROD workspace

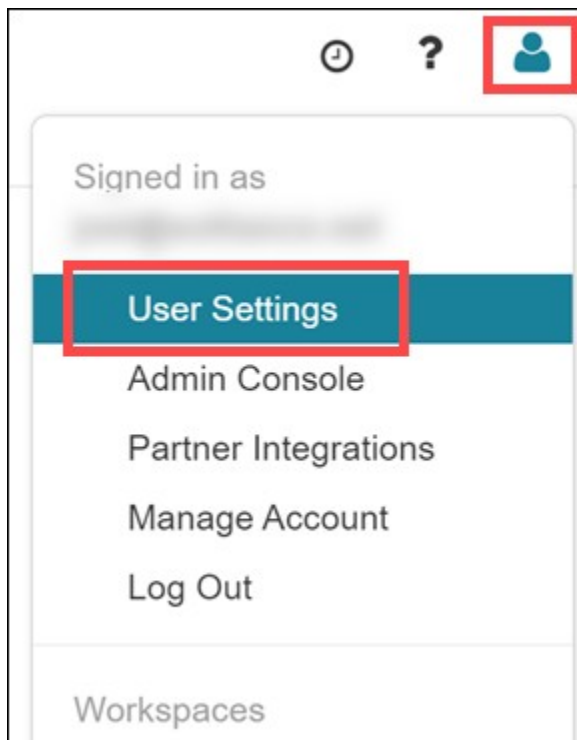
Go to PROD Azure Databricks workspace and generate a user access token for the Azure DevOps release pipeline you will create later, by following these steps:

1. Navigate to your production (PROD) Azure Databricks workspace.
2. When you launch and sign in to the workspace, take note of the URL. It is in the form of **https://<location>.azuredatabricks.net**. Copy the **location** portion of the URL (for example, westus2) and save it to a text editor for later reference.



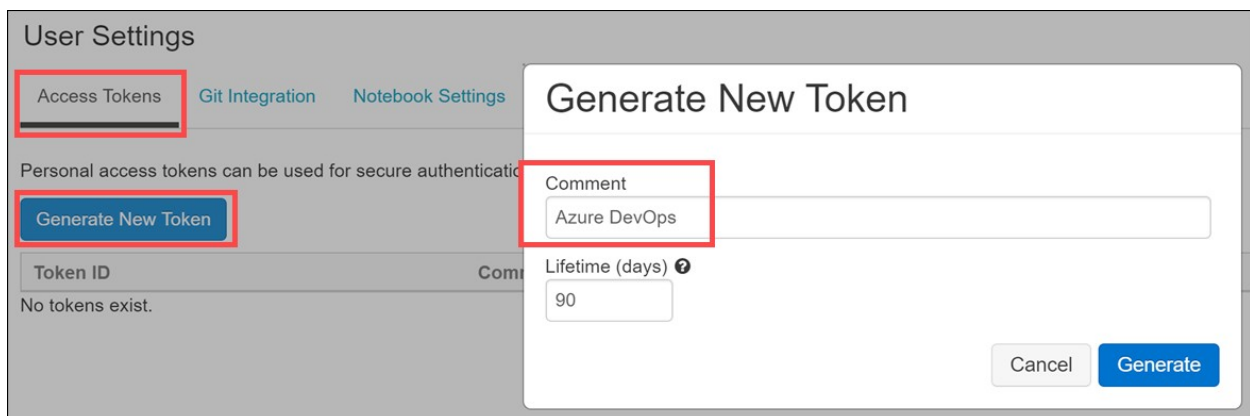
The Azure Databricks URL

3. Select the user icon on the top-right of the workspace, then select **User Settings**.



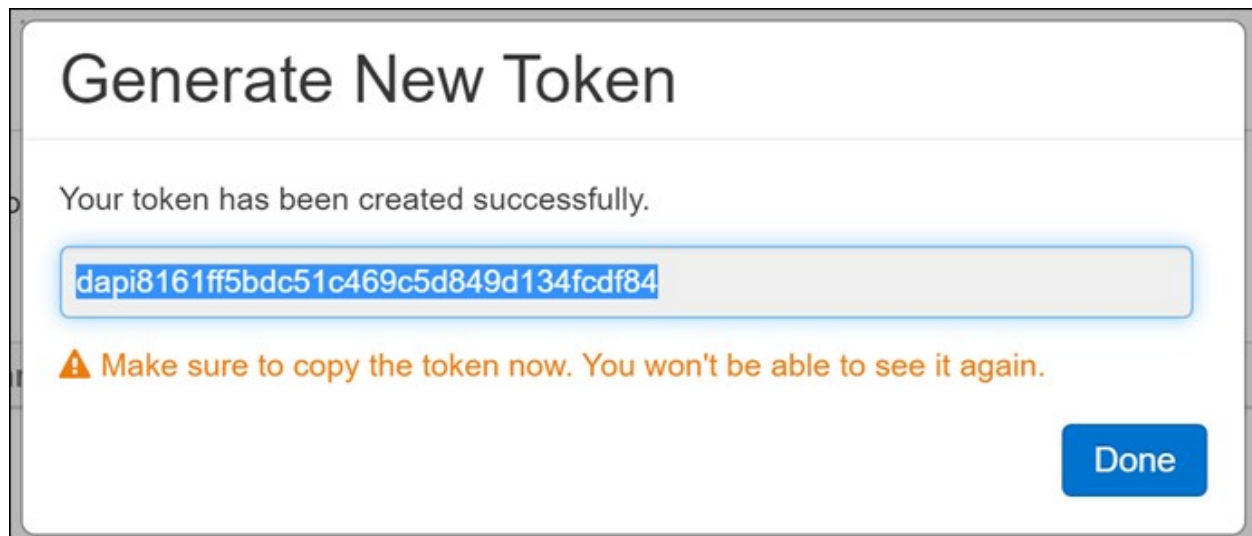
User Settings menu link.

4. Under the Access Tokens tab, select **Generate New Token**. In the Generate New Token dialog, add **Azure DevOps** for the Comment, then select **Generate**.



Generate New Token form is displayed.

5. Copy the new token and save it to a text editor for later reference. This is only displayed once.

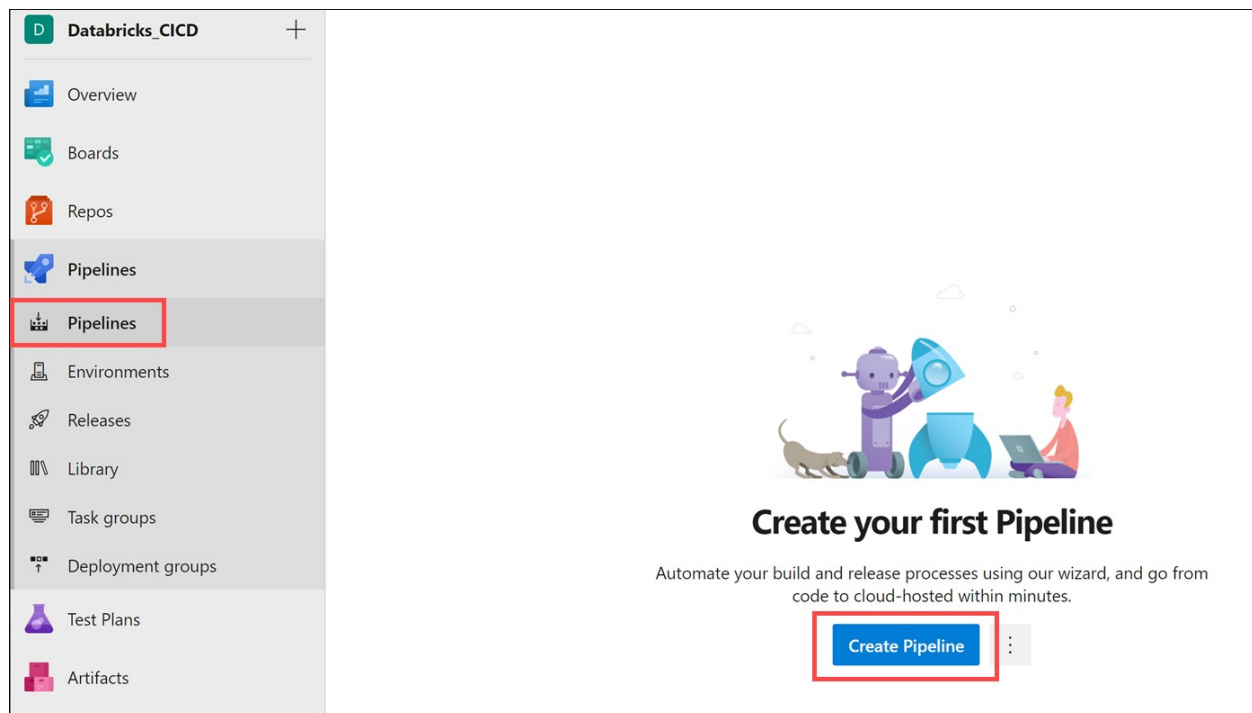


New Token is displayed.

Step 4: Azure DevOps - Create a build pipeline (CI)

A Build pipeline provides the **CI** portion of CI/CD.

1. Navigate back to Azure DevOps. Within your project, expand **Pipelines** in the left-hand menu, then select **Pipelines**. Select **Create Pipeline**.



Create Pipeline is highlighted.

2. Select **Use the classic editor** link under "Where is your code?"

New pipeline

Where is your code?



Azure Repos Git YAML

Free private Git repositories, pull requests, and code search



Bitbucket Cloud YAML

Hosted by Atlassian



GitHub YAML

Home to the world's largest community of developers



GitHub Enterprise Server YAML

The self-hosted version of GitHub Enterprise



Other Git

Any generic Git repository



Subversion


Centralized version control by Apache


[Use the classic editor](#) to create a pipeline without YAML.


The classic editor link is highlighted.


3. Select your project, repository, and the **master** branch for manual and scheduled builds, then select **Continue**.


Select a source



Azure Repos Git


GitHub


GitHub Enterprise Server


Subversion


Bitbucket Cloud


Other Git

Team project

 Databricks_CICD

Repository

 Databricks_CICD

Default branch for manual and scheduled builds


 master

Continue

The build source form is displayed.


4. Under Select a template, select the start with an **Empty job** link.

Select a template

Or start with an  **Empty job**

Search


Configuration as code



YAML


Looking for a better experience to configure your pipelines using YAML files? Try the new YAML pipeline creation experience. [Learn more](#)

Featured



.NET Desktop

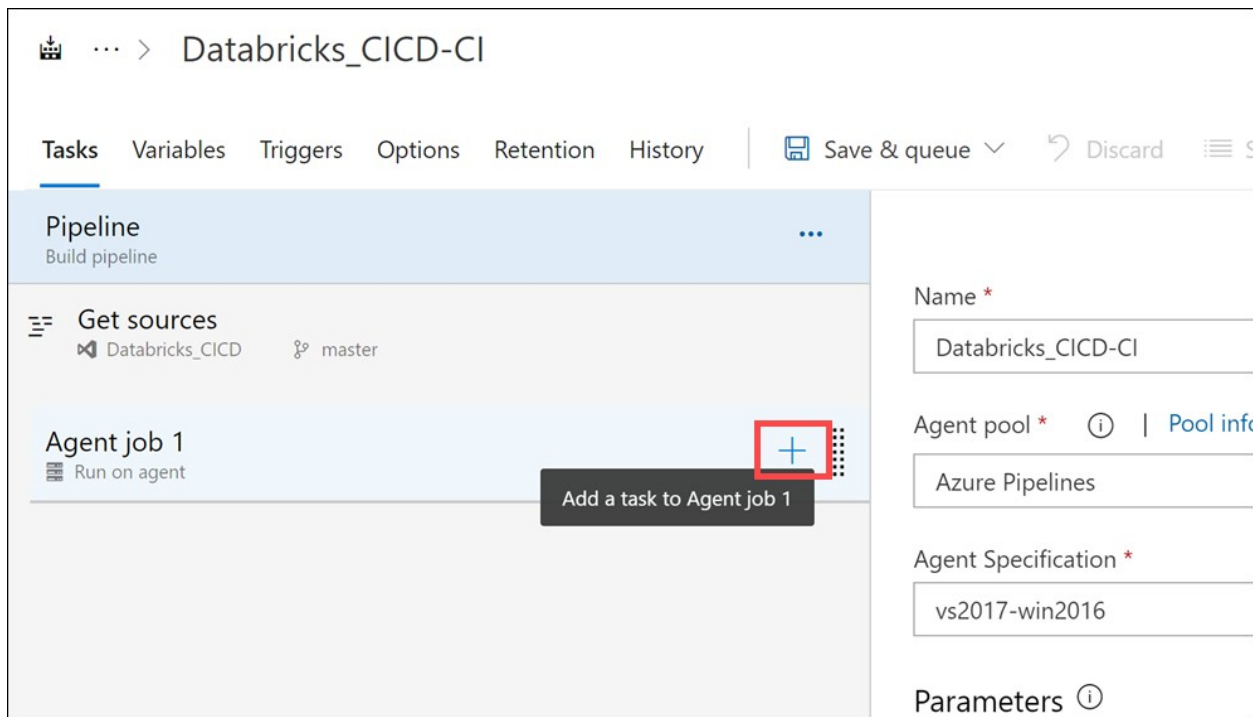
Build and test a .NET or Windows classic desktop solution.



Android

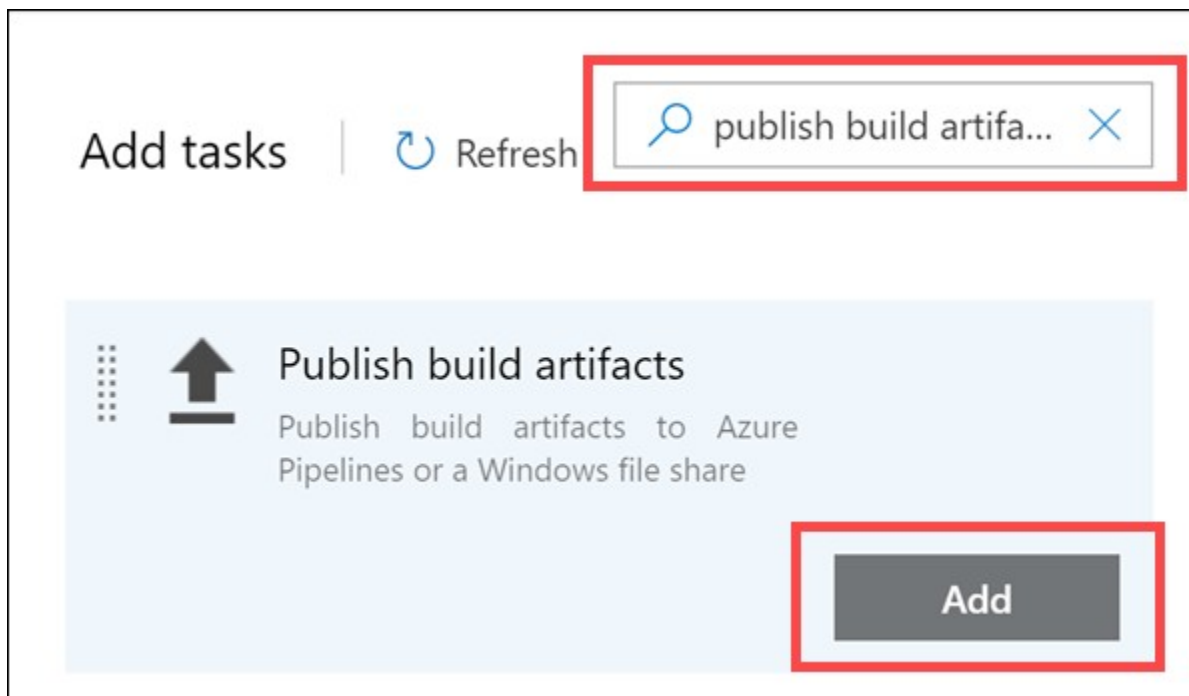
Empty job link is highlighted.

5. Select the **+** link on **Agent job 1** to add a task.



The Add Task link is highlighted.

6. Search for the "**Publish Build Artifacts**" task and add it to a Agent job.



The search box has the publish build artifacts search term, and the Publish Build Artifacts item is selected.

7. Select added task, enter **notebooks** for the **Path to publish** and enter **DEV build** for the **Artifact name**.

The screenshot shows the configuration for the 'Publish build artifacts' task. The left sidebar lists the pipeline steps: 'Get sources' (Databricks_CICD, master) and 'Agent job 1' (Run on agent). The 'Publish Artifact: DEV build' task is selected. The right pane shows the task configuration: 'Task version' is 1.*, 'Display name' is 'Publish Artifact: DEV build', 'Path to publish' is 'notebooks', 'Artifact name' is 'DEV build', and 'Artifact publish location' is 'Azure Pipelines'. The 'Triggers' tab is highlighted in the top navigation bar.

Pipeline
Build pipeline

Get sources
Databricks_CICD master

Agent job 1
Run on agent

Publish Artifact: DEV build
Publish build artifacts

Publish build artifacts ⓘ Link settings View Y

Task version 1.*

Display name *
Publish Artifact: DEV build

Path to publish * ⓘ
notebooks

Artifact name * ⓘ
DEV build

Artifact publish location * ⓘ
Azure Pipelines

Advanced ▾

The Publish Build Artifacts properties are shown.

8. Select the **Triggers** tab and check **Enable continuous integration**. This will automatically trigger a build whenever you commit your code to the repo.

The screenshot shows the 'Triggers' tab configuration. The left sidebar shows 'Continuous integration' with 'Databricks_CICD' (Enabled) and 'Scheduled' (No builds scheduled). The right pane shows the 'Databricks_CICD' trigger configuration: 'Enable continuous integration' is checked, 'Batch changes while a build is in progress' is unchecked, and 'Branch filters' are set to 'Include' and '*/ master'. The 'Triggers' tab is highlighted in the top navigation bar.

Tasks Variables **Triggers** Options Retention History | Save & queue ▾ Discard Summary ...

Continuous integration

Databricks_CICD
Enabled

Scheduled
No builds scheduled

Build completion
Build when another build completes

Databricks_CICD

☒ Enable continuous integration

☐ Batch changes while a build is in progress

Branch filters

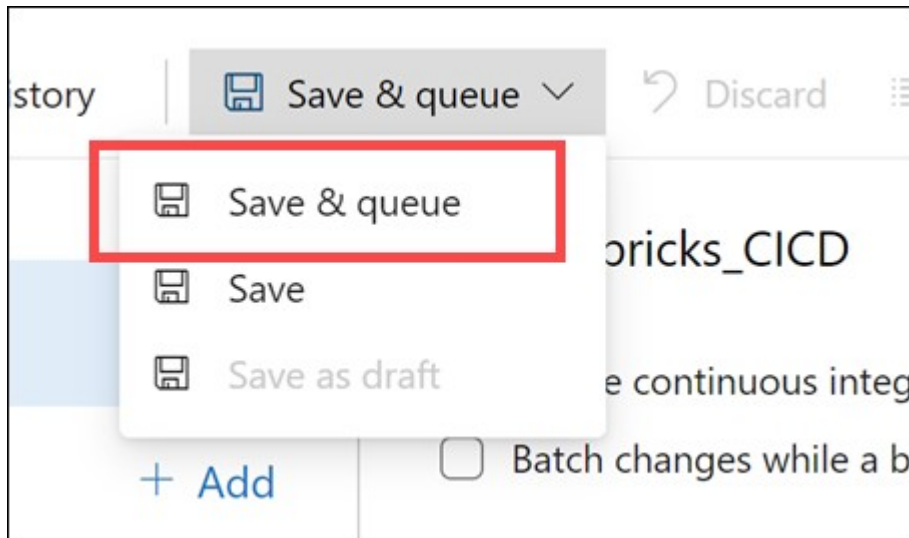
Type Branch specification

Include ▾ */ master ▾

+ Add

The Triggers tab is selected and enable continuous integration is checked.

9. Select **Save & queue** to continue.



Save and queue.

10. In the Run pipeline dialog that appears, enter a save comment, then select **Save and run**.

Run pipeline

×

Select parameters below and manually run the pipeline

Save comment

Initial setup


Agent pool

Azure Pipelines

Agent Specification *

vs2017-win2016

Branch/tag

 master

Select the branch, commit, or tag

Advanced options

Variables

1 variable defined

>

Demands

This pipeline has no defined demands

>

☐ Enable system diagnostics

Cancel

Save and run

The Run pipeline dialog is displayed.

11. Verify that your build pipeline was created and successfully run.

The screenshot shows the Azure DevOps interface for a pipeline named "Databricks_CICD". The left-hand menu is expanded to "Pipelines", and the "Summary" tab is selected. The pipeline is in a "Retained" state. The "Jobs" section shows a single job, "Agent job 1", which completed successfully in 9 seconds.

Repository and version	Time started and elapsed	Related	Tests and coverage
Databricks_CICD master 3bd648e	Today at 12:17 PM 12s	0 work items 1 published	Get started

Name	Status	Duration
Agent job 1	Success	9s

Successful pipeline run.

Step 5: Azure DevOps - Create a release pipeline (CD)

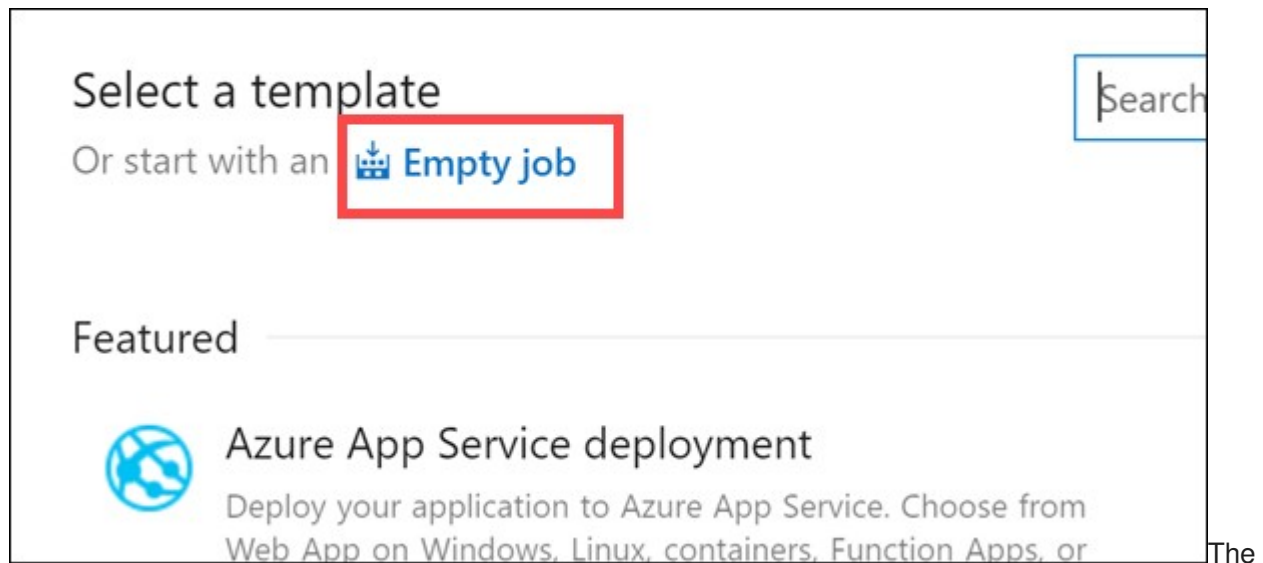
A release pipeline provides the **CD** portion of CI/CD.

1. Within your Azure DevOps project, expand **Pipelines** in the left-hand menu, then select **Releases**. Select **New pipeline**.

The screenshot shows the Azure DevOps interface for the "Databricks_CICD" project. The left-hand menu is expanded to "Releases", which is highlighted with a red box. The main area displays a message: "No release pipelines found. Automate your release process in a few easy steps with a new pipeline." Below this message is a blue button labeled "New pipeline", which is also highlighted with a red box.

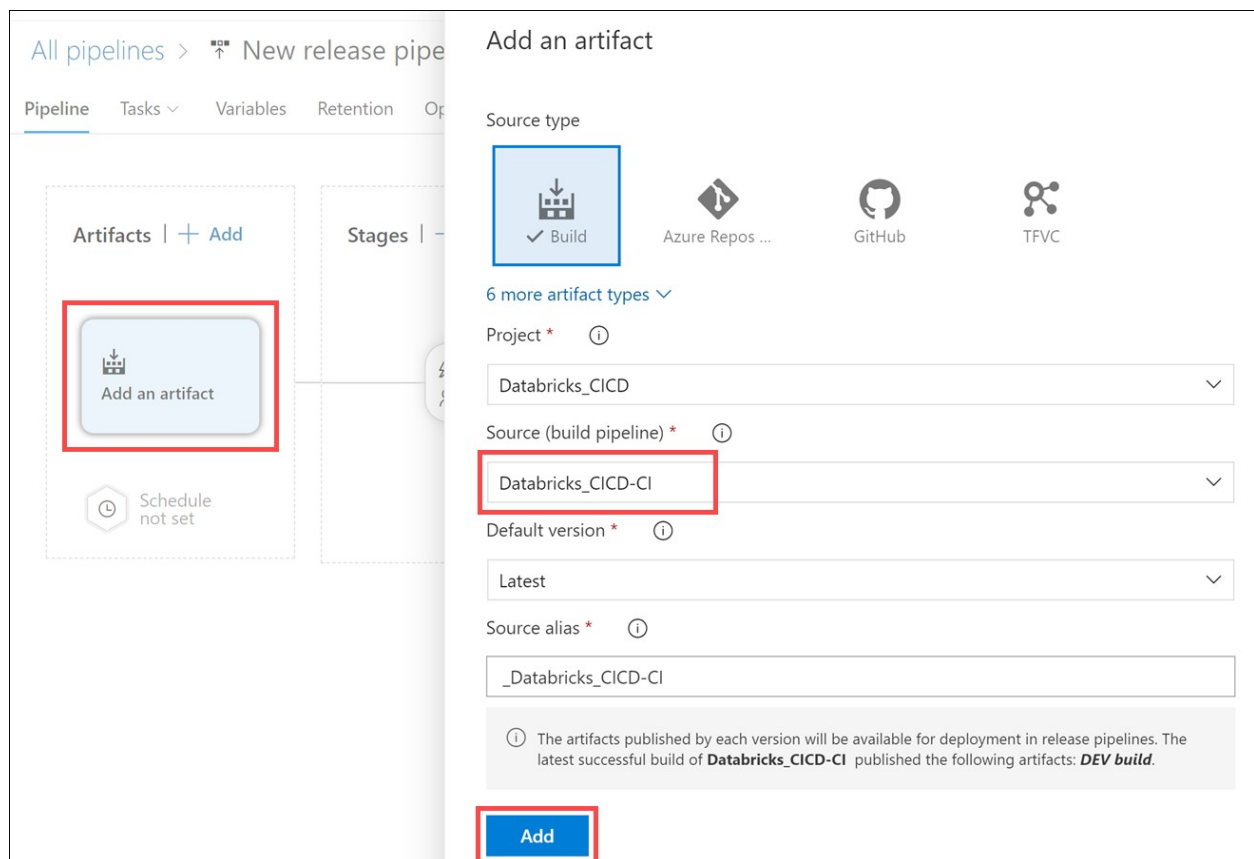
New pipeline is highlighted.

2. As you did when creating the previous pipeline, select the start with an **Empty job** link.



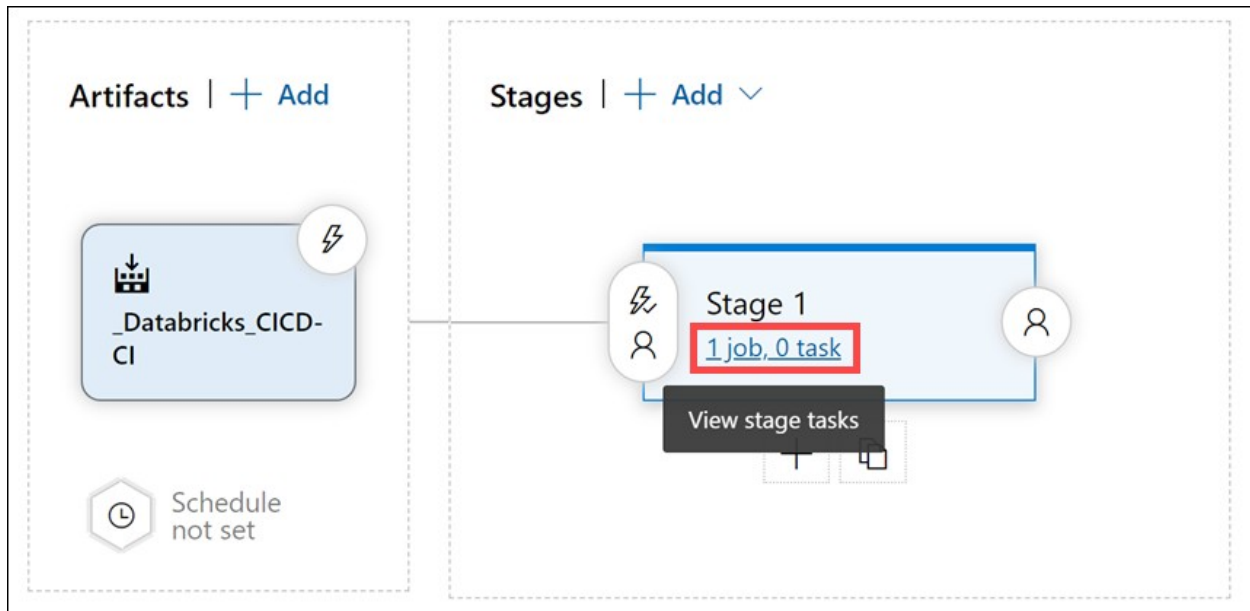
Empty job link is highlighted for release pipeline.

3. Select **Add an artifact**. Set the source type to **Build**, then select your build pipeline you created in the previous step as the **Source**. Select **Add** to apply your changes.



The add an artifact dialog is displayed.

4. View the tasks for Stage 1 by selecting the **1 job, 0 task** link.



add a task

5. Select the **+** link on **Agent job** to add a task. Search "Databricks", then add **Databricks Deploy Notebooks** **Note:** *If this is the first time adding this task, you first have to install "Databricks Script Deployment Task by Data Thirst", then the displayed Databricks tasks will become available. This package is provided by third-party.*

The screenshot shows the "Add tasks" interface in the pipeline configuration tool. The left sidebar shows "Stage 1" and "Agent job" with a red box around the "+" icon. The main area displays a list of tasks under the "Add tasks" header, with a search bar containing "databricks".

- Databricks Deploy Cluster:** Configures a Databricks Cluster (Analytics Workload) in your workspace.
- Databricks Deploy Notebooks:** Deploys a folder of Notebooks to your Databricks workspace. This task is highlighted with a blue background and has a red box around the "Add" button.
- Databricks Deploy Secret:** Deploy a secret to your Databricks workspace.
- Databricks files to DBFS:** Deploy files (py, jar, csv etc) to DBFS.
- Databricks Create Bearer Token:** Databricks Create Bearer Token and outputs back to a variable called \$(BearerToken) which you can reuse in later steps.

At the bottom, there is a "Marketplace" section with the task "Databricks Script Deployment Task by Data Thirst" listed.

The Databricks Deploy Notebooks task is displayed.

6. Once the task is added, select it and then fill the required parameters, then **Save**:

- **Azure Region:** Enter the region of your production (PROD) Azure Databricks workspace that you obtained from the URL in a previous step.
- **Source files path:** Browse to and select the **Users** subfolder.
- **Target files path:** Enter **/Users**.
- **Databricks bearer token:** Paste the Azure Databricks Access Key you copied in an earlier step.

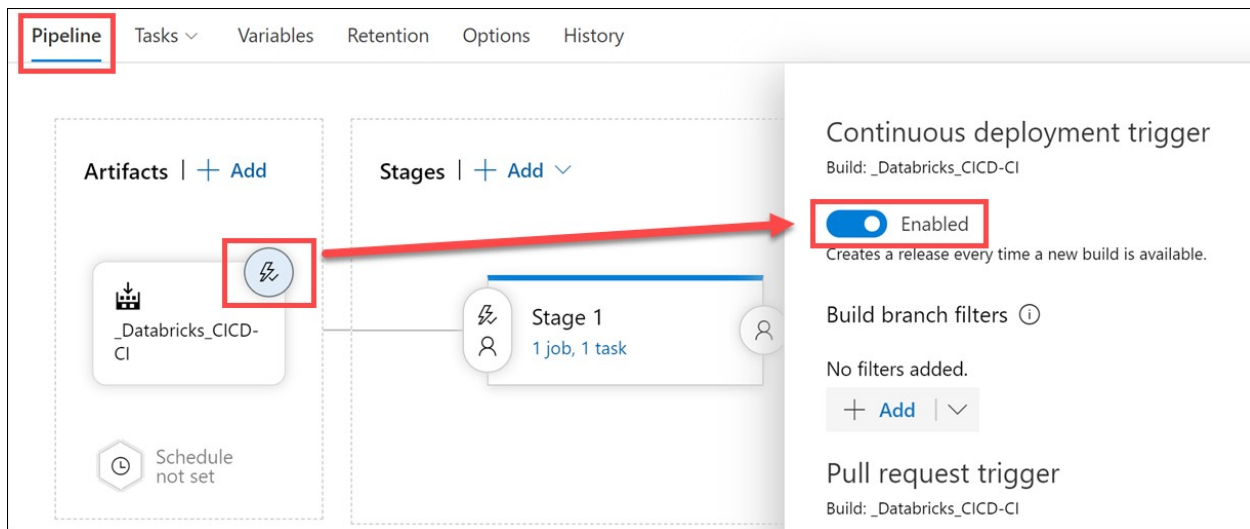
Note: The Databricks bearer token has to be obtained from PROD workspace for access permission.

The screenshot shows the 'New release pipeline' configuration page in Azure DevOps. The 'Tasks' tab is selected, and a task named 'Databricks Notebooks deployment' is configured. The configuration fields on the right are:

- Display name: Databricks Notebooks deployment
- Azure Region: westus2
- Source files path: \$(System.DefaultWorkingDirectory)/_Databricks_CICD-CI/DEV build/Users
- Target files path: /Users
- Clean Workspace Folder: ☐
- Security: ☒ Bearer Token, ☐ Service Principal
- Databricks bearer token: dapi8161ff5bdc51c469c5d849d134cdf84
- Control Options: ☐
- Output Variables: ☐

Databricks Notebooks deployment step form is displayed.

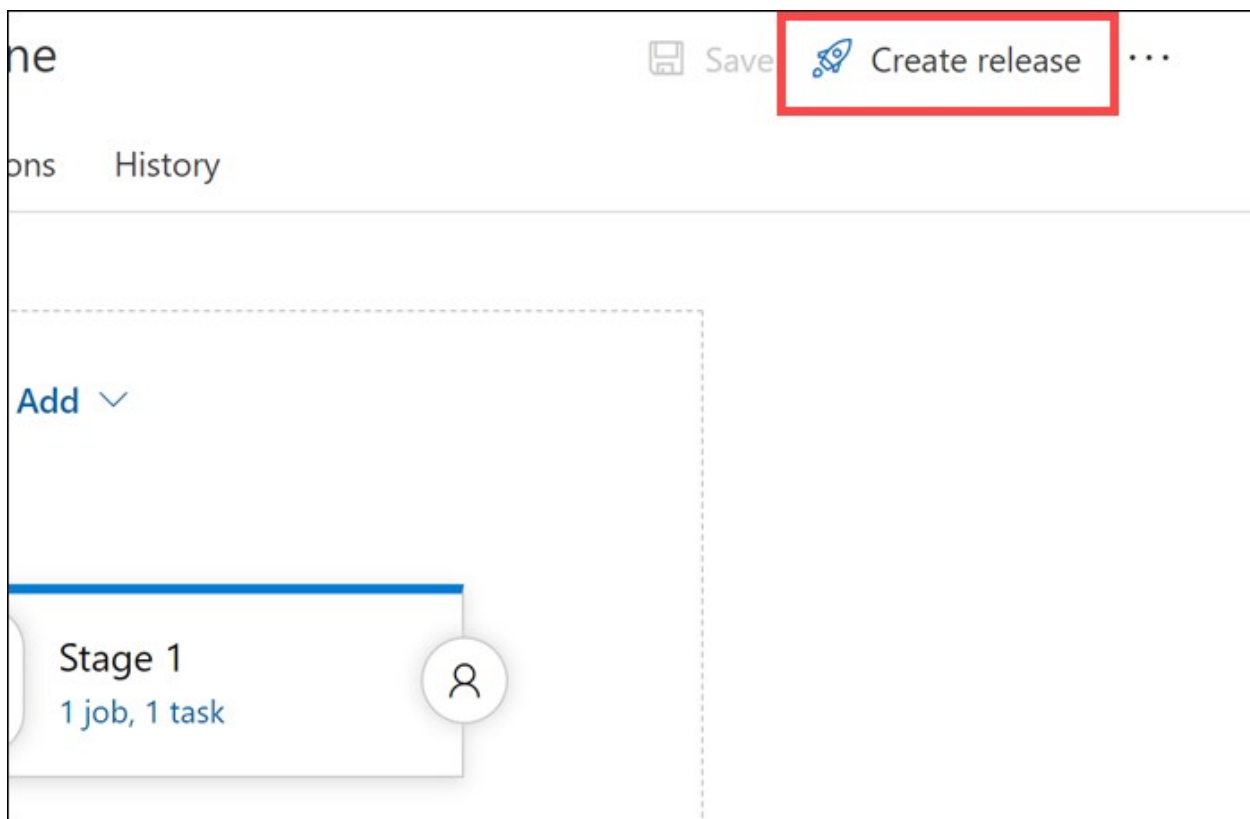
7. Select the **Pipeline** tab. Select the **Continuous deployment trigger** on the artifact, then **Enable** the continuous deployment trigger. This will create a release every time a new build is available.



The continuous deployment trigger is displayed.

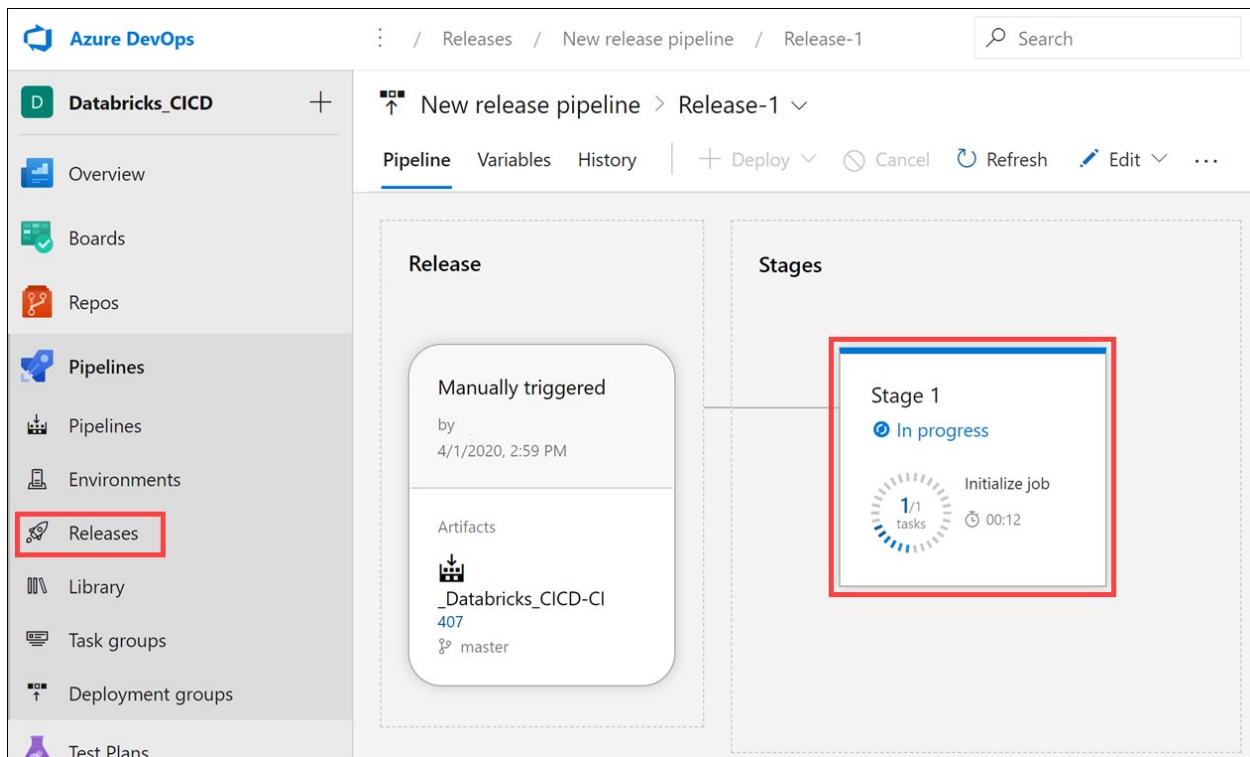
8. Select **Save** to save your pipeline changes.

9. Finally, create a release by selecting **Create release** at the top of the pipeline blade. When the **Create a new release** form displays, select **Create**.



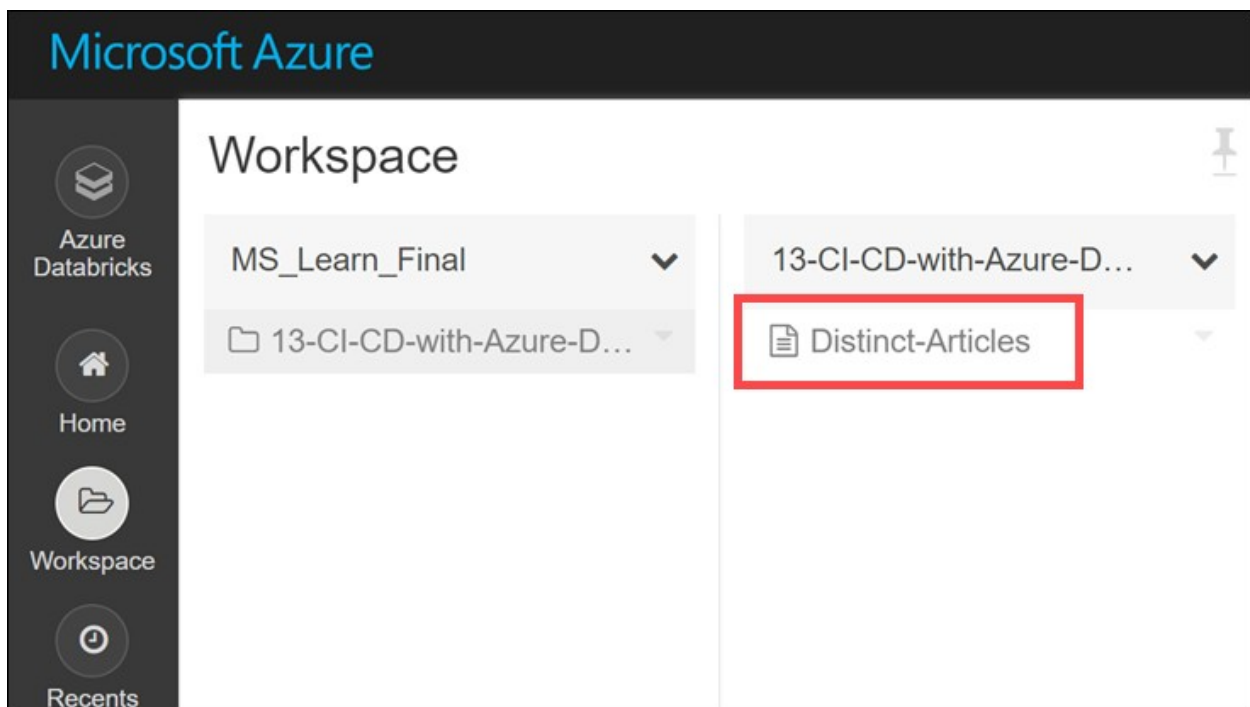
Create release is highlighted.

10. Navigate back to **Releases** under the Pipelines section of the left-hand menu. Select the release you just created. When it opens, you should see that it is either in progress or completed.



The release is in progress.

11. Navigate back to your production (PROD) Azure Databricks workspace. If it is already open, refresh the page. Navigate to your user folder under the workspace. You should see your notebook. This was saved to your workspace by your release pipeline.



Notebook in production workspace.

CI/CD setup is now completed. If you commit your code from the DEV workspace to the repo (master branch), the same notebook should be available in PROD.

Experiment with making changes to your notebook in DEV, then committing those changes. You will be able to see your build and release pipelines execute and the notebook in the PROD workspace automatically update to reflect those changes.