

Implement continuous integration and continuous delivery (CI/CD) in Microsoft Fabric

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/implement-cicd-in-fabric/1-introduction>

Introduction

Completed

As a Fabric data engineer or developer, you're responsible for designing, building, and maintaining Fabric items that store, process and, analyze data. You collaborate with your team to release changes from development through production in different Fabric workspaces. With multiple engineers working on projects, there's a need to incrementally release and integrate Fabric item changes and move those changes through a deployment process where they can be tested and quickly released. Lifecycle management in Fabric lets you do this.

Lifecycle management in Fabric uses deployment pipelines and source control integration to help you achieve continuous integration and continuous delivery (CI/CD). You can integrate Fabric with version control systems like GitHub and Azure DevOps and improve your delivery process, and integrate automated testing.

By the end of this module, you'll know how to work with source control in Fabric, how to use Fabric deployment pipelines and how Fabric REST APIs can be used for CI/CD automation.

2. Understand Continuous Integration and Continuous Delivery (CI/CD)

Understand Continuous Integration and Continuous Delivery (CI/CD)

Completed

When you and members of your team are each responsible developing and maintaining different parts of your Fabric environment, a best practice is to work in isolated development environments until you're ready to combine your development efforts and publish your changes to a particular pre-production environment. When you're ready to publish your changes, you need to make sure that your changes don't break existing code or interfere with changes made by other developers. There's also a need to ensure that code changes are saved and can be reverted if there are issues. The built-in continuous integration and continuous delivery capabilities in Fabric can help facilitate this.

Continuous integration and continuous delivery is a process for integrating code contributions from multiple developers into a main codebase. Contributions are frequently committed, and automated processes build and test the new code. Code is continuously moving into production, reducing feature development time.

Continuous integration

If developers work on separate code branches on their local machines for long periods of time and only merge their changes to the main codebase once their work is finished, this increases the likelihood of conflicts and bugs that might only be identified in later development stages and can slow down delivery of features to users.

Continuous integration (CI) helps you avoid bugs and code failures and lets you continuously develop and release functionality. In CI, you frequently commit code to a shared code branch or trunk in a version control system and once it's merged, changes are validated by a build process and automated testing. Conflicts between new and existing code are identified earlier in the development process and are easier and faster to fix.

Continuous delivery

Continuous delivery happens after continuous integration. Once CI is complete, code is deployed to a staging environment where more automated testing is performed before code is released into production.

Continuous deployment

Continuous deployment is a process that automatically releases updates into production environments through structured deployment stages, once they pass automated tests.

Use CI/CD in Fabric

Managing the lifecycle of Fabric items using CI/CD has two parts: integration and deployment. Integration is implemented using Git. Deployment is implemented using Fabric deployment pipelines. Automation of deployment or integration is implemented using Fabric REST APIs.

- **Git:** Lets your team collaborate using branches, and provides version control. It helps you manage incremental code changes, and see code history.
- **Deployment pipelines:** Lets you promote code changes to different environments like development, test, and production.
- **Fabric REST APIs:** Enables automation and lets you programmatically manage CI/CD processes.

3. Implement version control and Git integration

<https://learn.microsoft.com/en-us/training/modules/implement-cicd-in-fabric/3-implement-version-control-and-git-integration>

Implement version control and Git integration

Completed

To support continuous integration, you frequently merge your code changes into a shared repository. The shared repository is part of a version control system like GitHub or Azure DevOps. Version control is a way of managing changes to code over time. It lets you track code revisions, contribute collaboratively to code development, and revert to prior versions of code if needed.

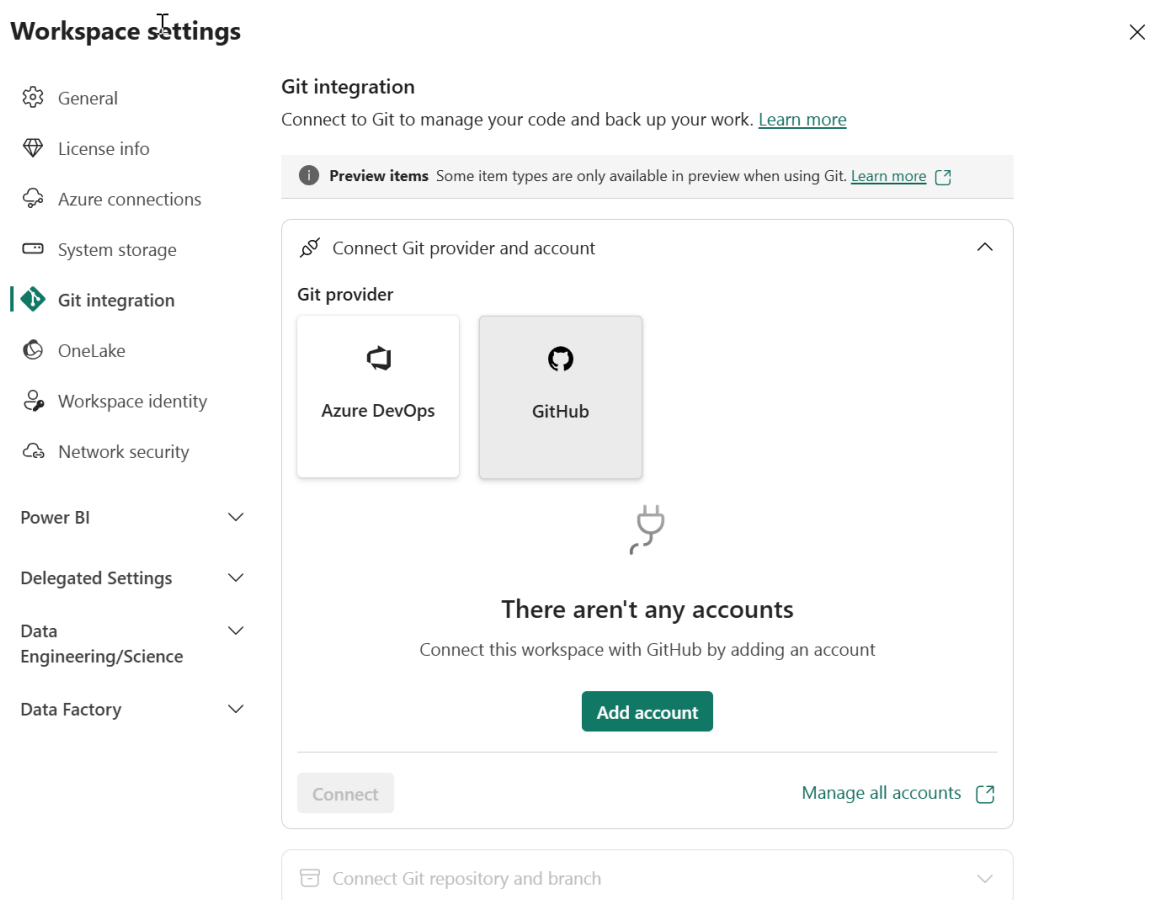
GitHub and Azure DevOps are the version control systems that are supported in Fabric. These version control systems allow you to create a copy of a code repository that's called a branch. You can use the branch to work on your own code independently from the main version of your team's code. When you have changes to submit, you can **commit** them to the repository and merge your changes with a main code branch.

Integration with version control is at the workspace level in Fabric. You can version items you develop within a workspace.

Connect to a Git Repository

A Fabric workspace is shared environment that accesses live items. Any changes made directly in the workspace overrides and affect all other workspace users. A best practice is for you to develop in an isolated workspace, outside of a shared, live workspace. In your own protected workspace, you can connect to your own branch and sync content from the live workspace into your protected workspace, and then commit your changes back to your branch or the main branch.

1. **Set up a git repository:** The first step in implementing Git integration is to set up a Git repository in either GitHub or Azure DevOps. The repository is the central location for storing and managing items.
2. **Connect a Fabric workspace to a Git repository:** Next, within the workspace that you want to connect to your repository, establish a connection to the repository from the **Git integration** option in workspace settings.



When you connect a workspace to Git, you **create or select an existing a Git repository branch** to sync with. Fabric syncs the content between the workspace and Git so they have the same content.

Workspace settings



- General
- License info
- Azure connections
- System storage
- Git integration**
- OneLake
- Workspace identity
- Network security

- Power BI
- Delegated Settings
- Data Engineering/Science
- Data Factory

Git integration

Connect to Git to manage your code and back up your work. [Learn more](#)

Preview items Some item types are only available in preview when using Git. [Learn more](#)

Connect Git provider and account

Provider

GitHub

Fabric Testing

<https://github.com/github-acct/FabricTesting>

Log out

[Manage all accounts](#)

Connect Git repository and branch

Repository URL

<https://github.com/github-acct/FabricTesting>

Branch *

Demo

Git folder

Enter name of folder

Connect and sync

Cancel

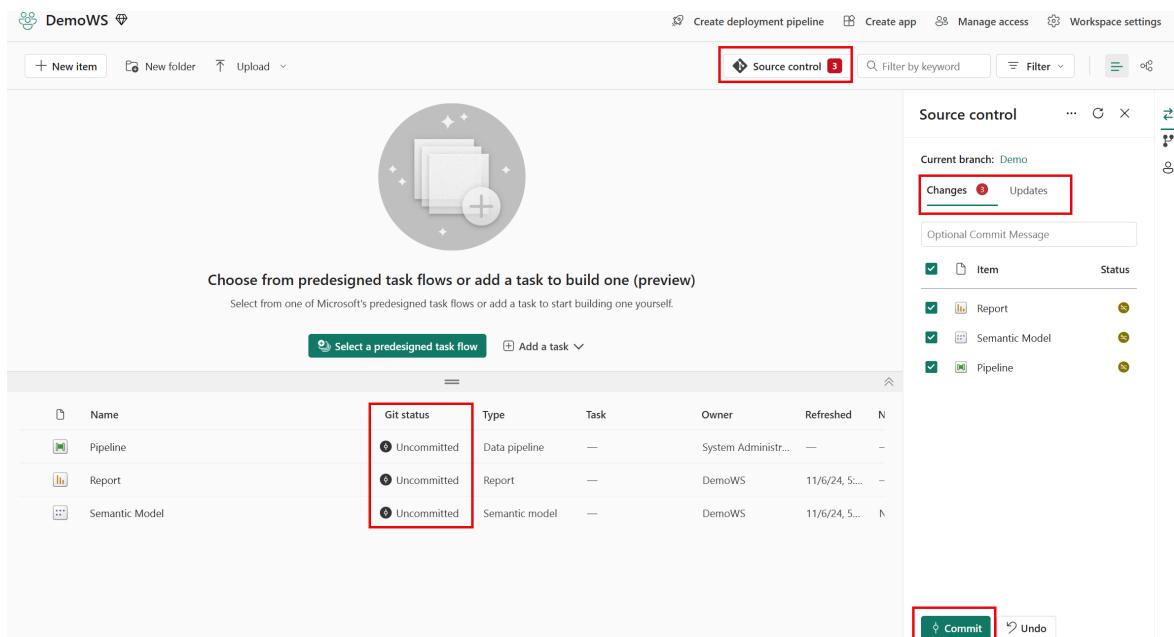
Commit and update the Fabric workspace and Git repository

After you connect to the repository, the workspace shows a Git status column indicating the sync state of items in the workspace, compared to the items in the remote branch.

The source control icon shows the number of items that are different between the workspace and the repository.

To synchronize the workspace and repository:

- When you make workspace changes, synchronize them with the Git branch using the **Changes** selection in the **Source control** window.
- When new commits are made in the Git branch, synchronize them with your workspace using the **Updates** selection in the **Source control** window.



Branching scenarios

Changes that you make to a workspace when you're doing development work affect all other workspace users so it's a best practice to work in isolation outside of shared workspaces. To keep your development work isolated from shared workspaces, you can develop using:

- A separate, isolated workspace
- Client tools like Power BI Desktop for reports and semantic models or VS Code for Notebooks.

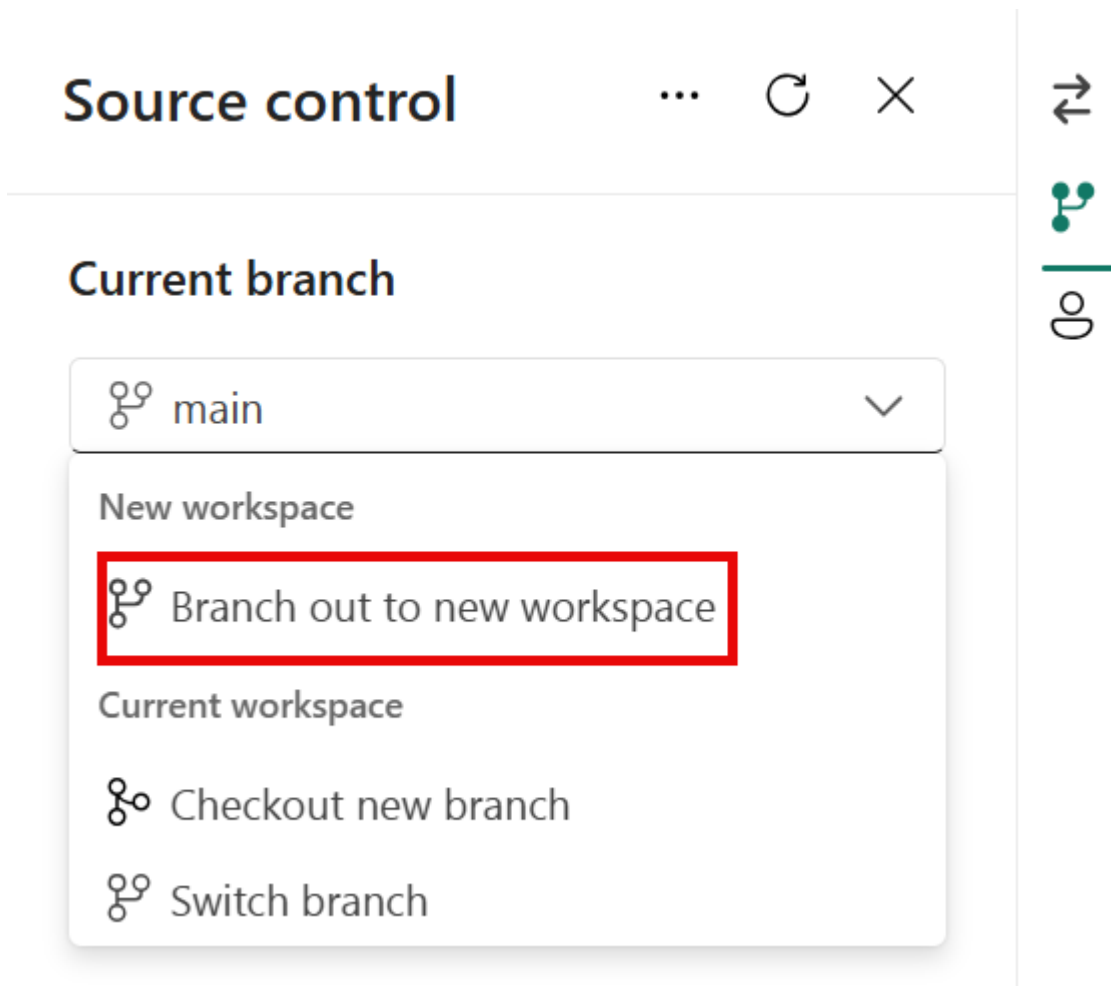
In both scenarios, your feature development work should take place in a dedicated branch instead of the main code branch. This makes it easy for multiple developers to work on a feature without affecting the main branch.

Create a dedicated branch, issue pull requests and sync a workspace with Git

Create a dedicated branch and issue pull requests to pull changes from your branch into the main branch by following these steps:

For development using a separate, isolated workspace:

1. Connect a development workspace to the main branch, following the instructions in the section on this page entitled "Connect to a Git Repository".
2. If you're a developer who works in the Fabric web interface, create an isolated branch for your work from the development workspace that's connected to the main branch by selecting **Source control** and **Branch out to new workspace**. Name the branch and associate it with another workspace. The new workspace syncs with the new branch you create and become an isolated work environment for your work.



3. Makes changes in your branch, then commit them to your isolated branch via the **Source control** interface in Fabric.
4. Then, *in Git*, create a **Pull Request (PR)** to pull the changes from your isolated branch into the main branch.
5. The main branch in Git will be updated once the PR is merged to the main branch. When you open the *shared* development workspace, you'll be prompted to synchronize the new content from Git with the shared development workspace.

When using client tools for development, the process is similar to that when developing in the Fabric web interface.

1. Connect a development workspace to the main branch, following the instructions in the section on this page entitled "Connect a Fabric workspace to a Git repository".
2. Clone the repository on your local machine.
3. Push the changes to the remote repository when you're ready to test in Fabric. Test the changes by connecting your isolated branch to a separate workspace.
4. Issue a PR *in Git* to merge your changes into the main branch.
5. When you open the shared workspace associated with the main branch, you'll be prompted to sync the changes from the repository into the workspace.

4. Implement deployment pipelines

<https://learn.microsoft.com/en-us/training/modules/implement-cicd-in-fabric/4-implement-deployment-pipelines>

Implement deployment pipelines

Completed

Pipelines enable a continuous integration/continuous deployment (CI/CD) approach that ensures content is updated, tested, and regularly refreshed. Pipelines are a way to automate the movement of content through the development, test, and production stages of the content development lifecycle.

What are deployment pipelines?

Fabric deployment pipelines help you deploy your Fabric items across different environments like development, test, and production. They let you develop and test content in Fabric before it reaches end users.

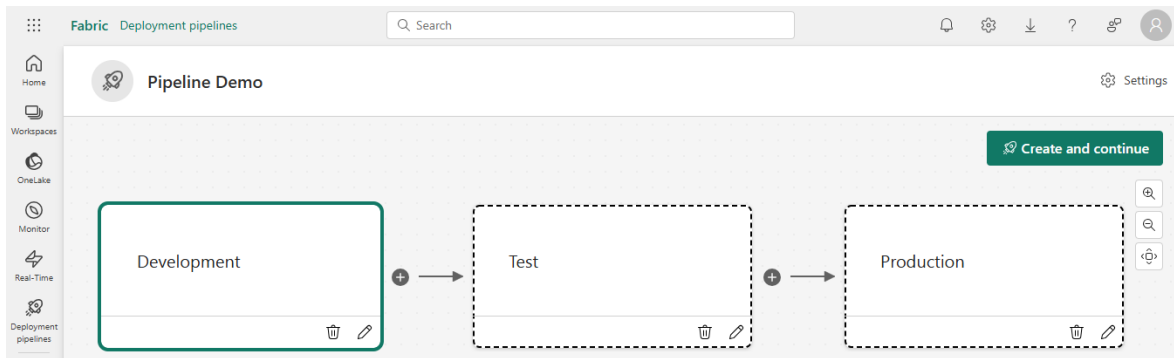
Create a deployment pipeline

Deployment pipelines can be created using two different methods:

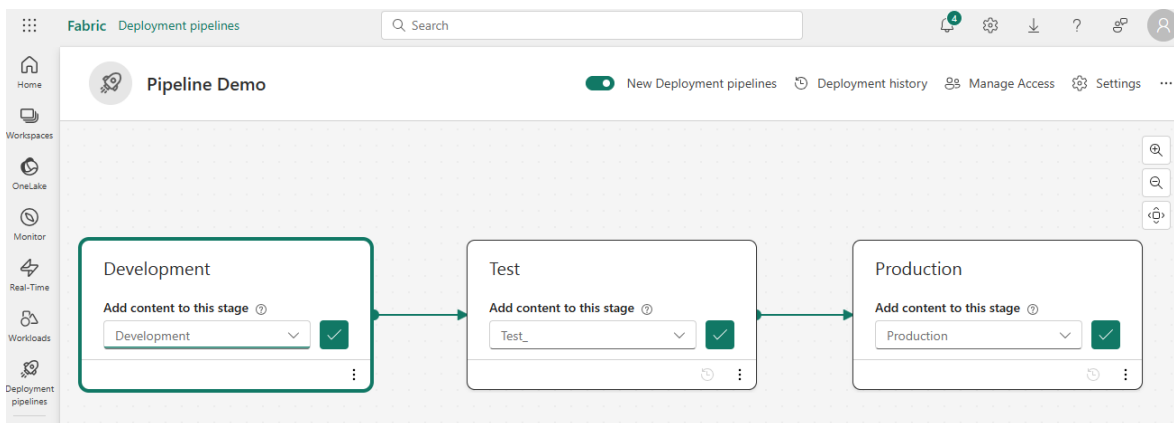
- Using the **Workspaces** icon on the left navigation pane in Fabric.
- Using the **Create deployment pipeline** icon at the top of a workspace

Follow these steps to create a deployment pipeline:

1. Select the **Workspaces** icon on the left navigation pane, then **Deployment pipelines**.
2. Select **New pipeline**. Then name the pipeline and select **Next**.
3. Define, and name the stages in your pipeline. Then select **Create and continue**.



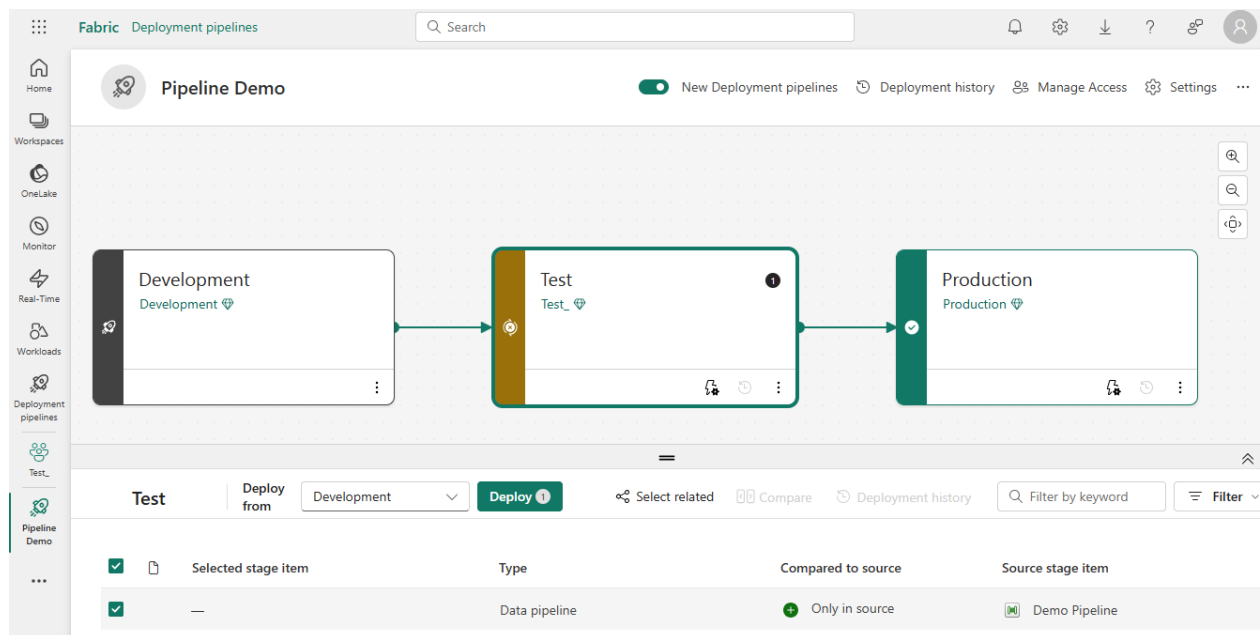
4. Assign a workspace to a stage. Then select the green check mark next to the stage. Then you're ready to deploy content to the pipeline.



Deploy content to a pipeline stage

The deployment process lets you clone content from one stage in the pipeline to another, typically from development to test, and from test to production.

To deploy content between stages, select the stage to deploy to, and then select the stage in the **Deploy from** drop-down box, and then select the **Deploy** button. The deployment process copies all of the workspace content into the target stage. In the following image, there's a data pipeline that only exists in the development stage that will be moved to the test stage when **Deploy** is selected in the development stage.



Use deployment pipelines with Git

Deployment pipelines can be used with Git integration to maintain version control and automate deployments. There are different approaches to combining these tools, depending on your workflow needs.

One common approach is to connect only the Development workspace to Git. With this approach, Git integration is used for version control during development, while deployment pipelines handle the promotion of content to Test and Production environments. This avoids potential Git synchronization conflicts when deploying content across multiple stages.

To use deployment pipelines with Git using this approach:

1. Follow the instructions in the section on this page entitled "Create a deployment pipeline" to create a deployment pipeline and assign each stage to a workspace.
2. Connect the Development workspace to a Git repository and branch in **Git integration** in **Workspace settings**.

Workspace settings



General

License info

Azure connections

System storage

Git integration

OneLake

Workspace identity

Network security

Power BI

Delegated Settings

Data
Engineering/Science

Data Factory

Git integration

Connect to Git to manage your code and back up your work. [Learn more](#)

Preview items Some item types are only available in preview when using Git. [Learn more](#)

Connect Git provider and account

Git provider



Azure DevOps



GitHub



There aren't any accounts

Connect this workspace with GitHub by adding an account

Add account

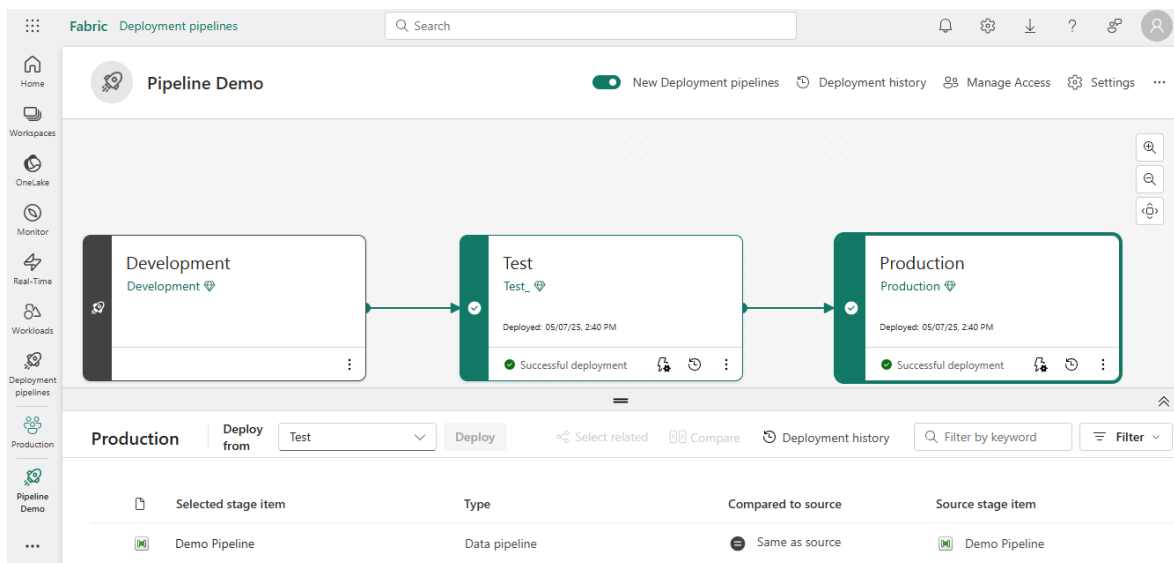
Connect

Manage all accounts

Connect Git repository and branch

3. Make your content changes in the Development workspace and commit them to Git using the **Source control** panel.
4. Promote content between staging environments using the deploy button in the pipeline as described in the **Deploy content to a pipeline stage** section on this page. This moves content between environments in Fabric. The deployment pipeline copies content from Development to Test and Production workspaces.

In the image below, the checkmark in the deployment stage box indicates that a data pipeline item exists in all three staging environments of the deployment pipeline *in Fabric* and that the Fabric stages are synchronized.



Tip

For more information about different CI/CD workflow options in Fabric, including alternative approaches for combining Git integration with deployment pipelines, see [Choose the best Fabric CI/CD workflow option for you](#).

5. Automate CI/CD using Fabric APIs

<https://learn.microsoft.com/en-us/training/modules/implement-cicd-in-fabric/5-automate-cicd-using-fabric-apis>

Automate CI/CD using Fabric APIs

Completed

Fabric REST APIs allow you to automate Fabric procedures and processes, improving efficiency and productivity. REST API stands for Representational State Transfer Application Programming Interface. Azure REST APIs are used to manage and interact with various Azure services.

Some of the advantages of using the Fabric REST APIs are:

- Automating repeat processes with consistency, making it easier to perform data processing on an ongoing basis.
- Seamless integration with other systems and applications, providing a streamlined and efficient data pipeline.

Fabric CI/CD REST APIs are available for deployment pipelines and Git integration.

- [Deployment pipelines REST APIs](#)
- [Git REST APIs](#)

Use the Fabric REST APIs for CI/CD to automate processes

You can use the Fabric CI/CD REST APIs to:

- Commit the changes made in the workspace to the connected remote branch.
- Update the workspace with commits pushed to the connected branch.
- See which items have incoming changes and which items have changes that weren't yet committed to Git with the Git status API.
- List Deployment Pipeline Stage Items: Returns the supported items from the workspace assigned to the specified stage of the specified deployment pipeline.
- Deploy Stage Content: Deploys items from the specified stage of the specified deployment pipeline.

6. Exercise: Implement deployment pipelines in Microsoft Fabric

<https://learn.microsoft.com/en-us/training/modules/implement-cicd-in-fabric/6-exercise-implement-deployment-pipelines>

Exercise: Implement deployment pipelines in Microsoft Fabric

Completed

Now it's your chance to use CI/CD in Microsoft Fabric.

In this exercise, you learn how to use Fabric deployment pipelines to automate the process of copying changes between environments like development, test, and production.

Note

You need a Microsoft Fabric trial license with the Fabric preview enabled in your tenant. See [Getting started with Fabric](#) to enable your Fabric trial license.

Launch the exercise and follow the instructions.

7. Module assessment

<https://learn.microsoft.com/en-us/training/modules/implement-cicd-in-fabric/7-knowledge-check>

Module assessment

Completed

8. Summary

<https://learn.microsoft.com/en-us/training/modules/implement-cicd-in-fabric/8-summary>

Summary

Completed

In this module, you learned how to work with version control in Fabric, how to use Fabric deployment pipelines and how Fabric REST APIs can be used for CI/CD automation.

To learn more, see:

- [Tutorial: End to end lifecycle management](#)
- [Exploring CI/CD Capabilities in Microsoft Fabric: A Focus on Data Pipelines](#)
- [Using the Microsoft Fabric REST APIs](#)