Continuous Integration and Continuous Delivery with Azure Data Factory and Azure DevOps

Azure Labs by Roque Daudt (rdaudt@yahoo.com)

05 – Continuous Integration

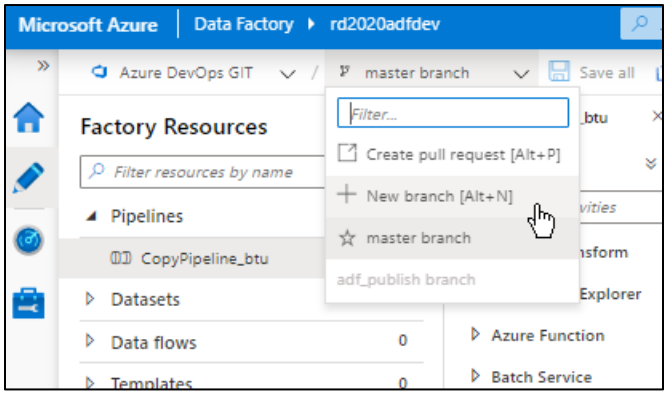
In the previous lab we set up a repo for ADF dev. Now, we will review the process to apply changes to an ADF pipeline and integrate them back to the main branches, **master** (collaboration) and **adf_publish**.

If you have closed your browser/tabs, open ADF Dev again as well as Azure Devops.

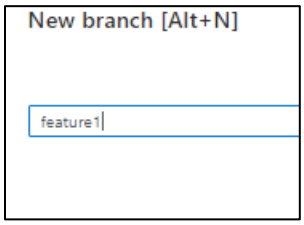
Create a feature branch

When developers need to apply changes to an ADF pipeline they start by creating a new branch out of master in the development ADF. Navigate to ADF Dev in order to run this part.

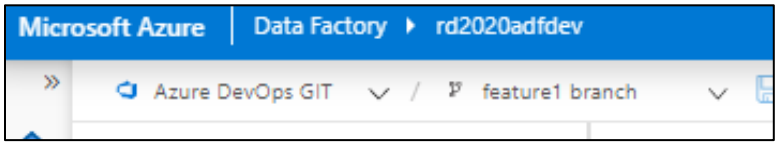
Open the dropdown list below.



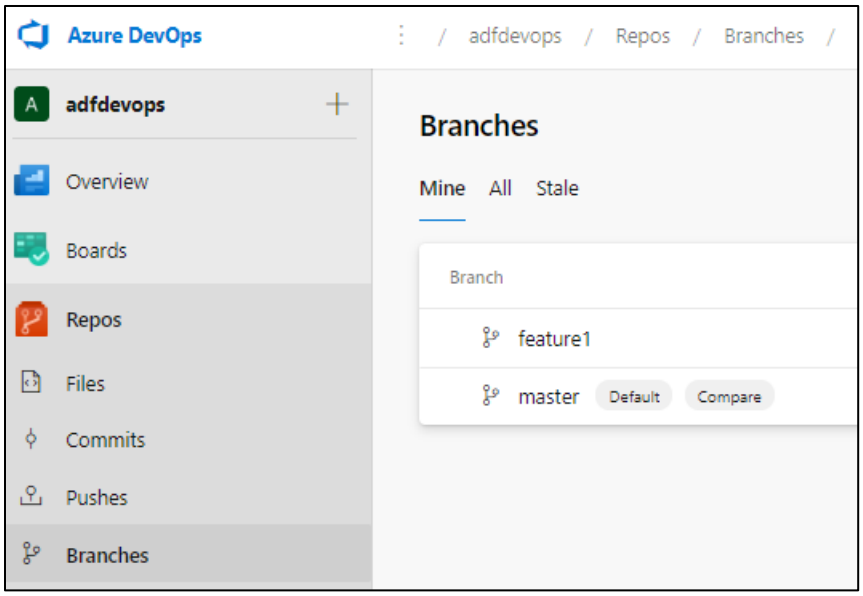
Give the new branch a name, **feature1**.



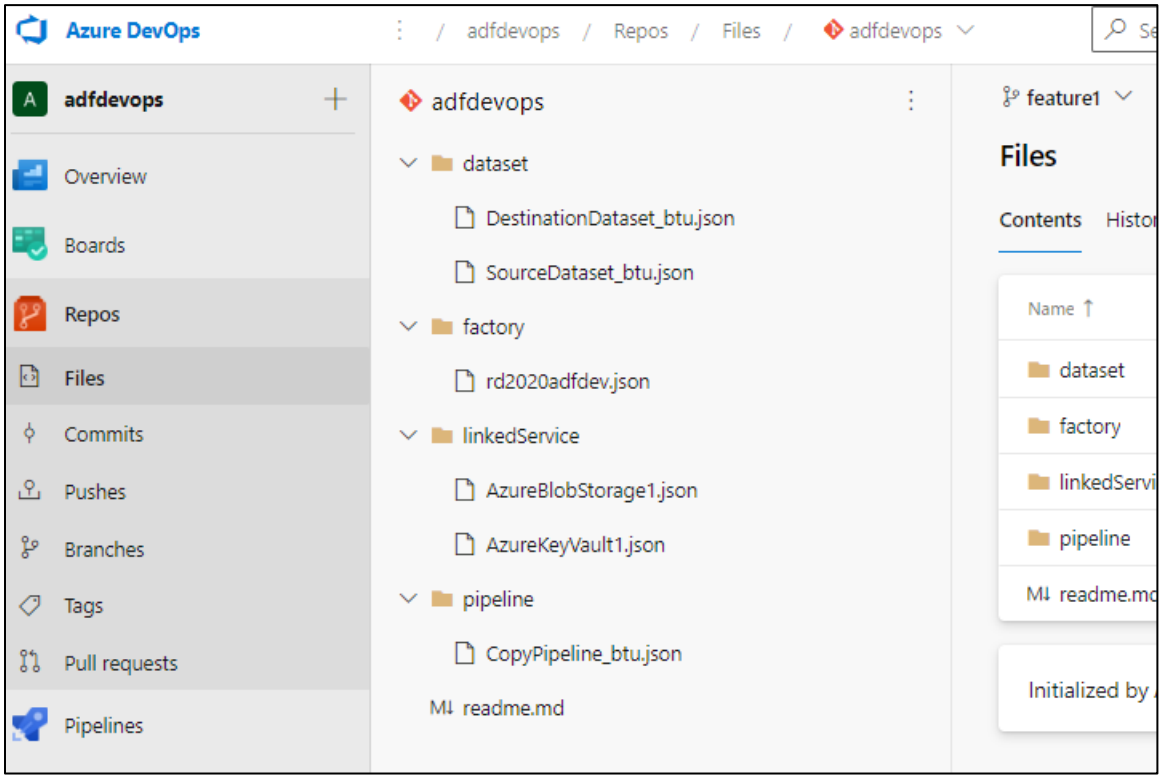
Enter a branch name and click **Save**. The new branch is created, and it becomes the current branch.



Go back to Azure DevOps. Now we see both **master** and **feature1** branches.



Click the **feature1** branch. You can then see the same folders and files that exist in master.

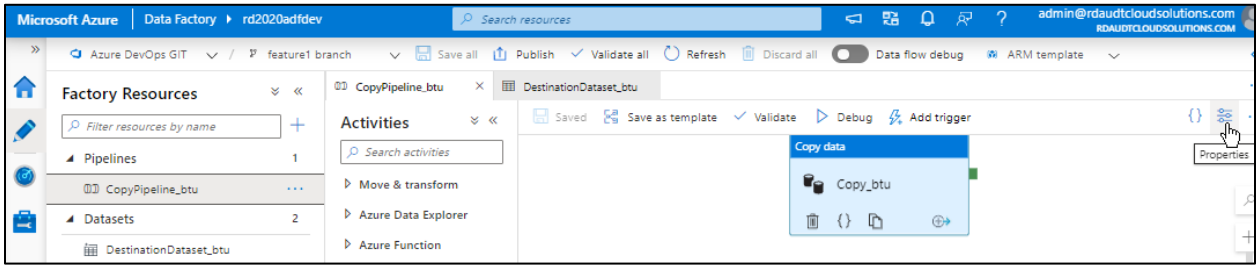


Leave Azure DevOps open and go back to ADF Dev.

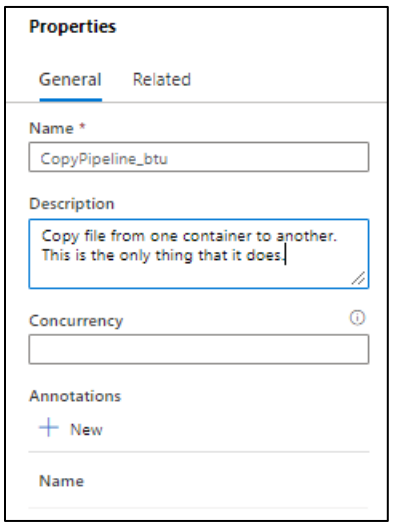
Update the ADF pipeline

Now we want to apply a simple change to the pipeline in the **feature1** branch and learn how to integrate it to the main branches. We will simply update the pipeline's description.

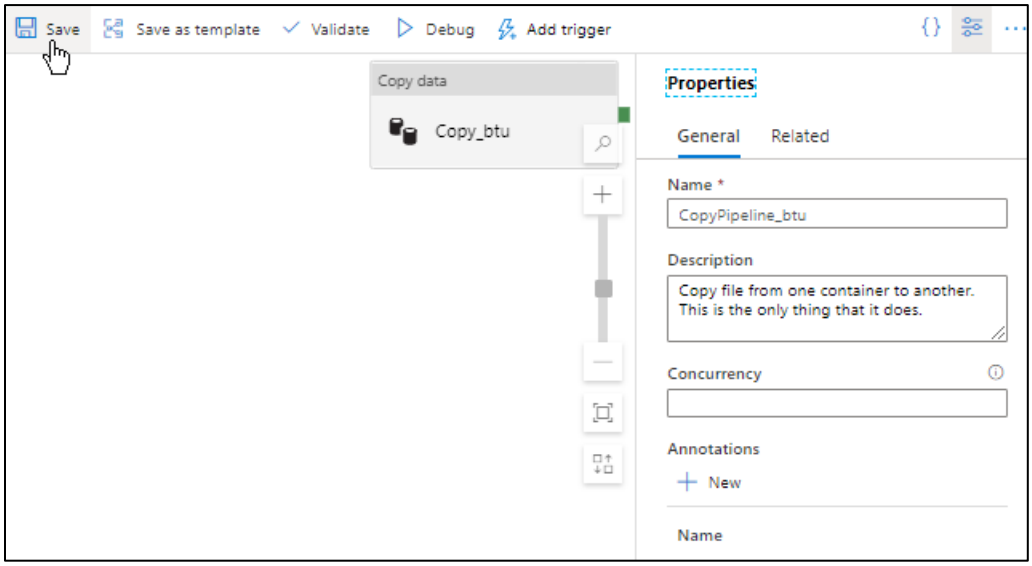
Go back to the tab where ADF dev is, click **CopyPipeline_btu**, and click **Properties**.



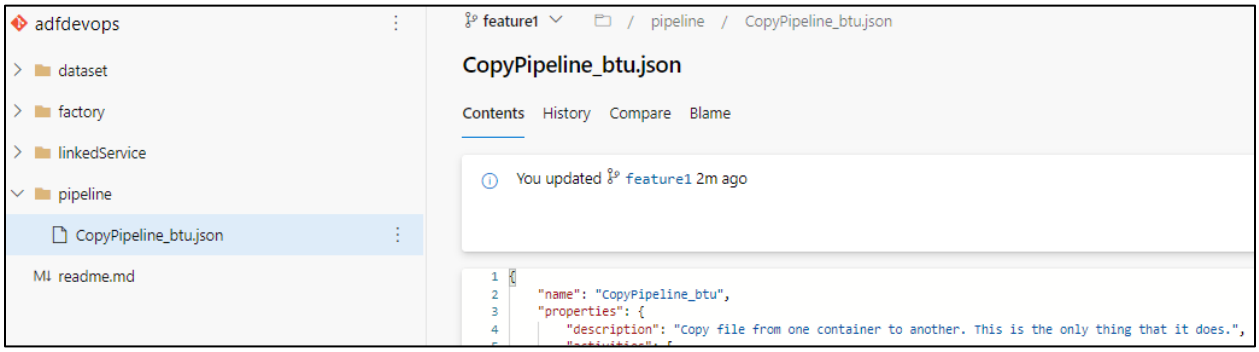
In the **Properties** panel, update the **Description** as shown below, or enter any additional text.



Click **Save**.



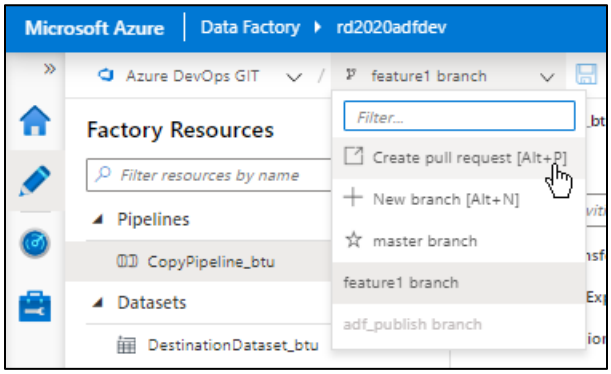
Go to the tab where Azure DevOps is open. Verify that the change saved in ADF Dev was indeed saved in the **feature1** branch.



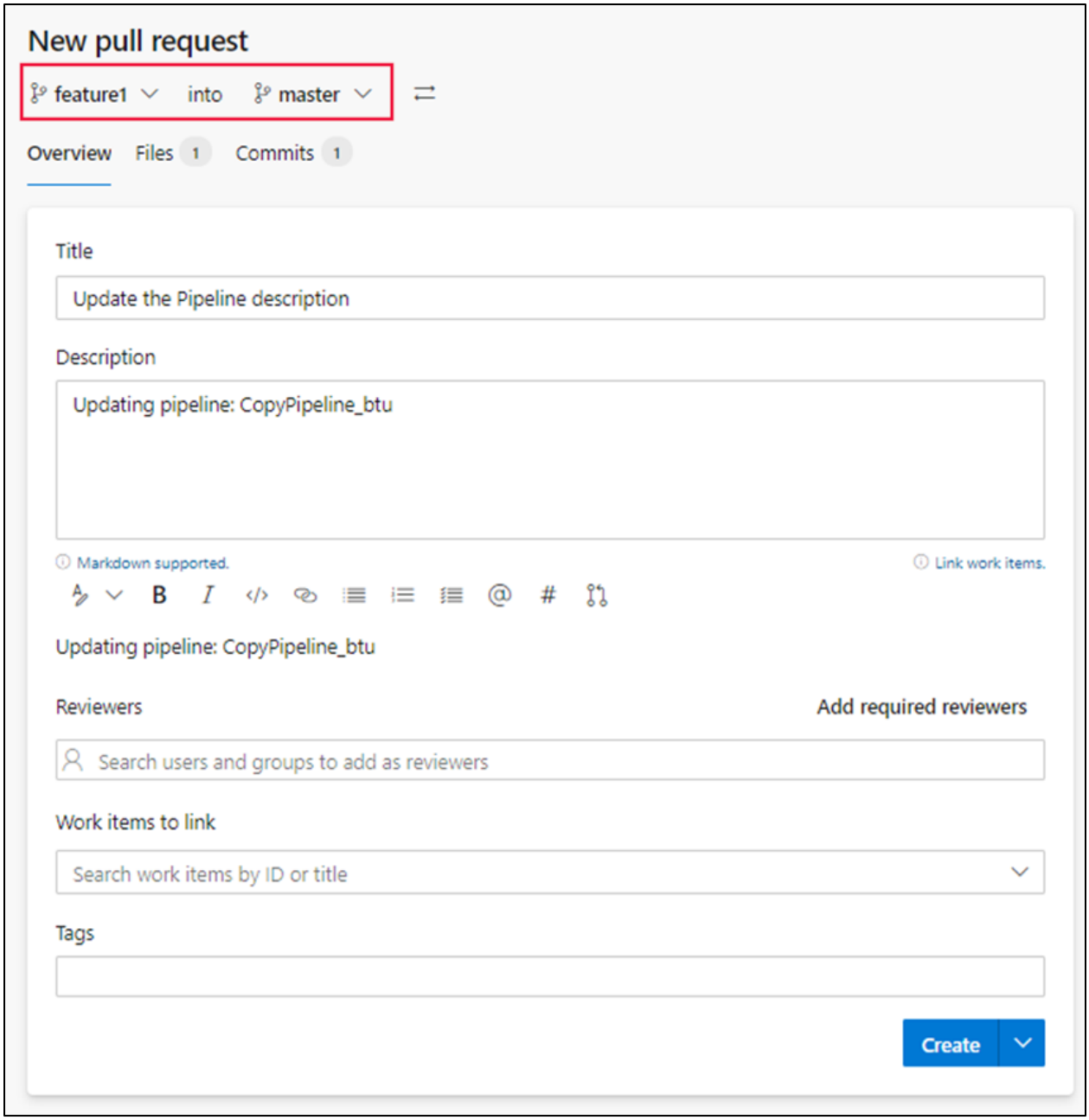
Merge the changes to the master branch

Go back to ADF Dev.

In order to start the process to merge changes back to the master branch the developer **creates a pull request** from the feature1 branch.

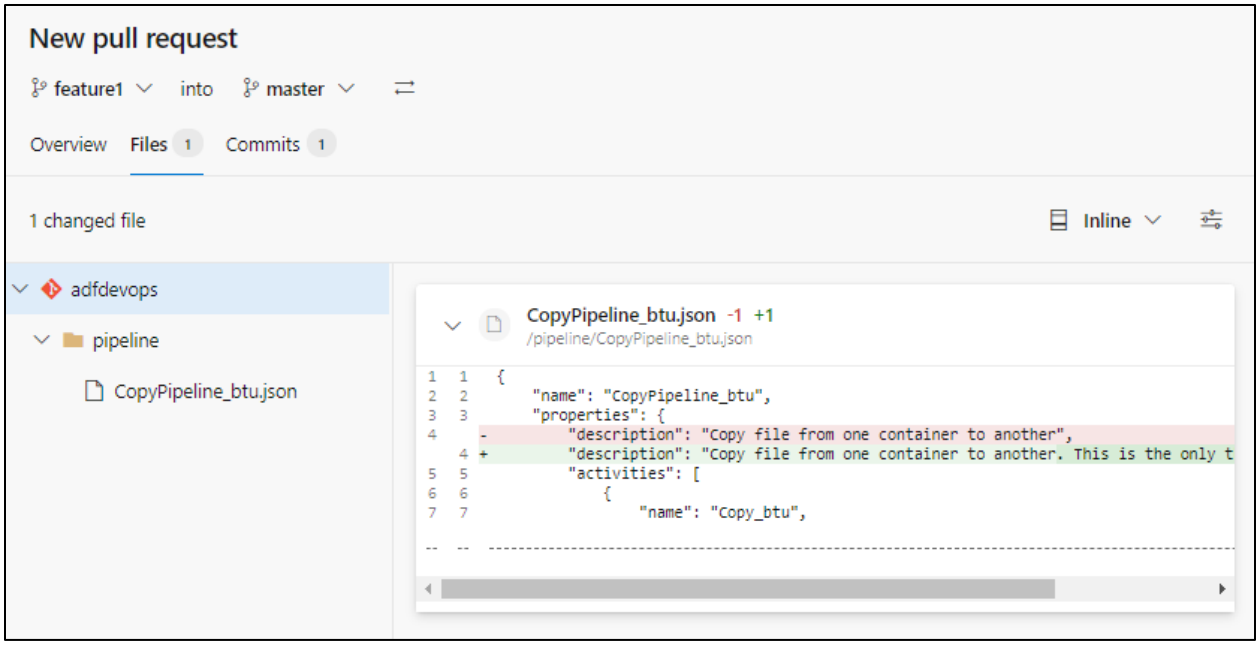


This takes the developer to the **New pull request** form in Azure DevOps.

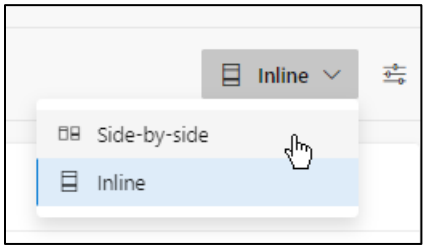


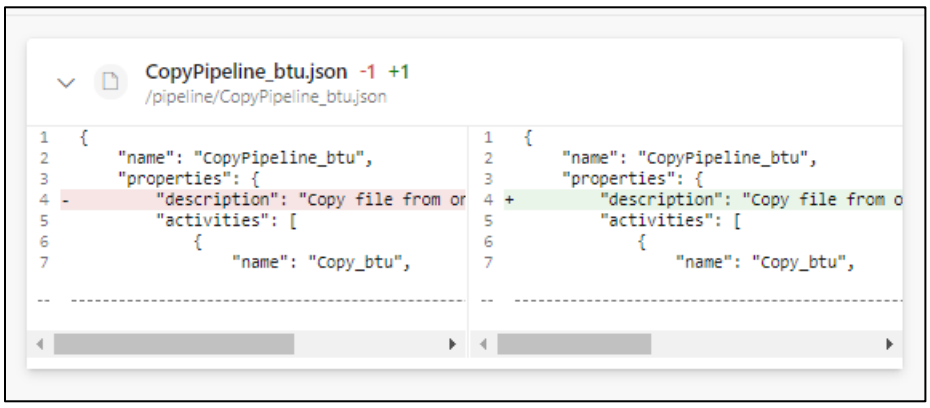
Make sure that the request is **from feature1 branch to the master branch**, as shown above in red.

Click the **Files** tab and review the change that will be merged.

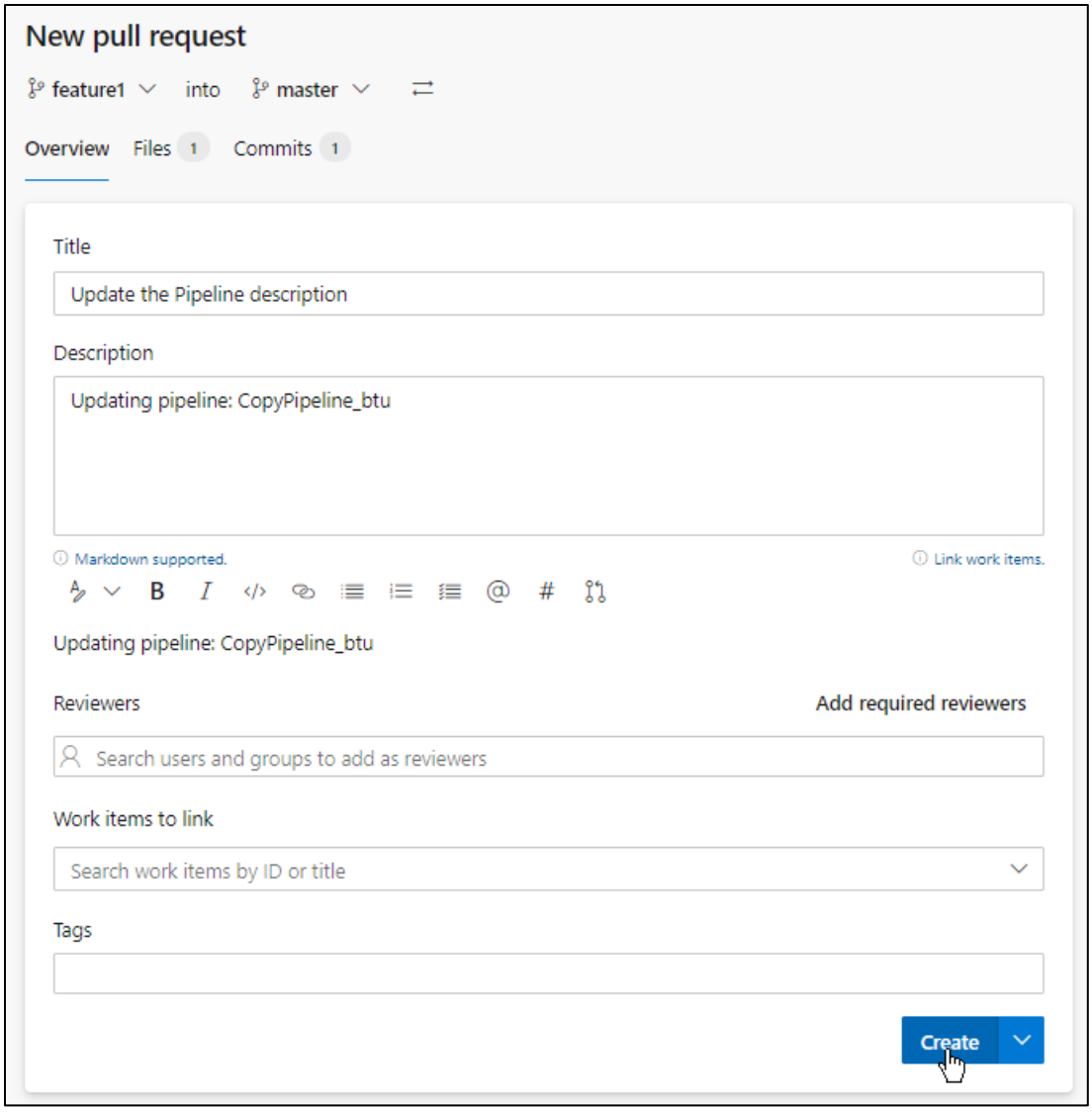


You have a choice of showing the files (current and changed) in side-by-side mode.

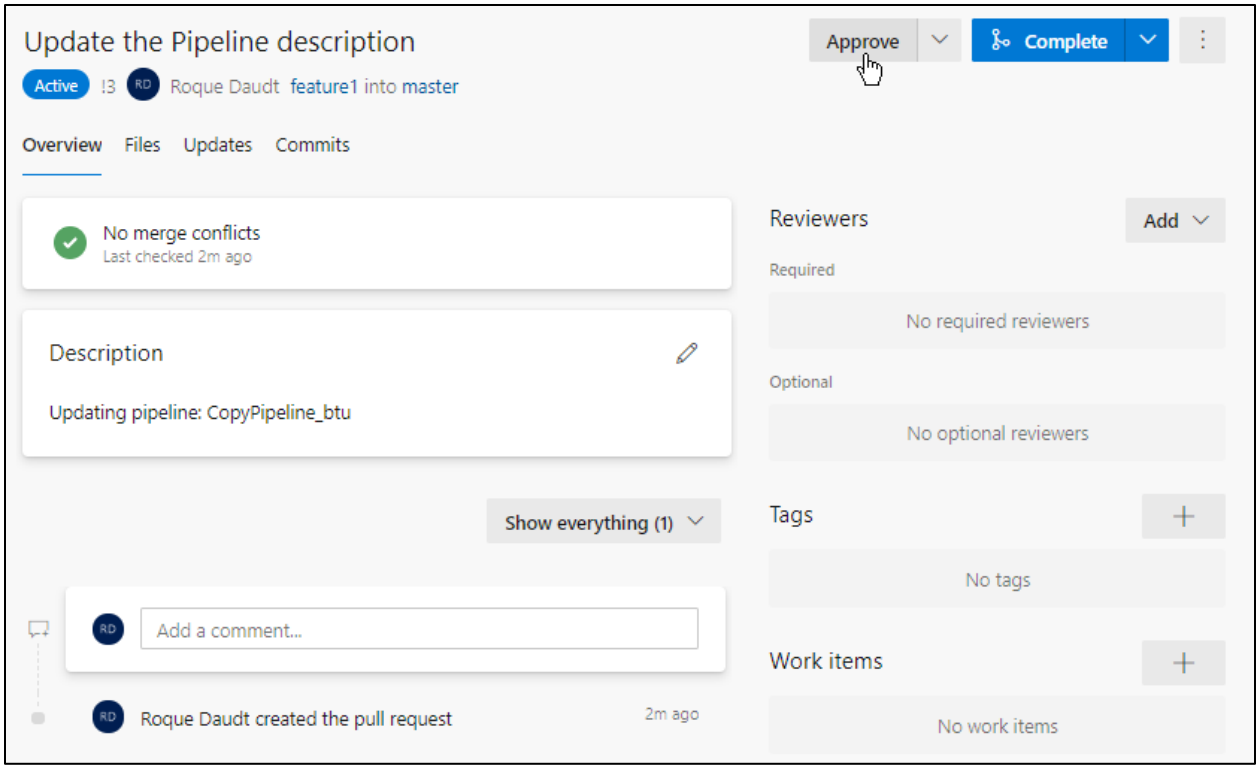




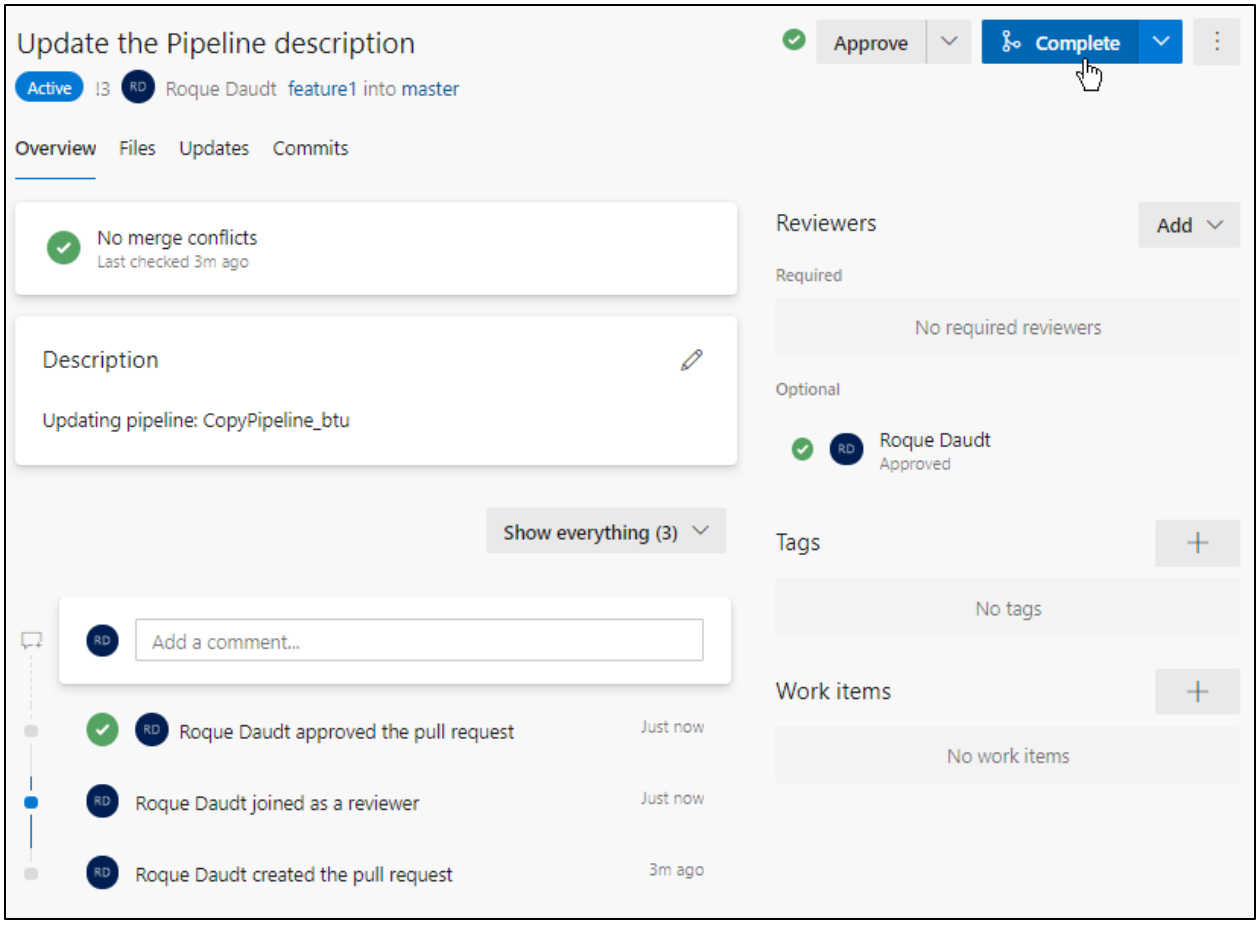
Go back to **Overview** tab and enter a more meaningful Title or leave as it is. Click **Create**.



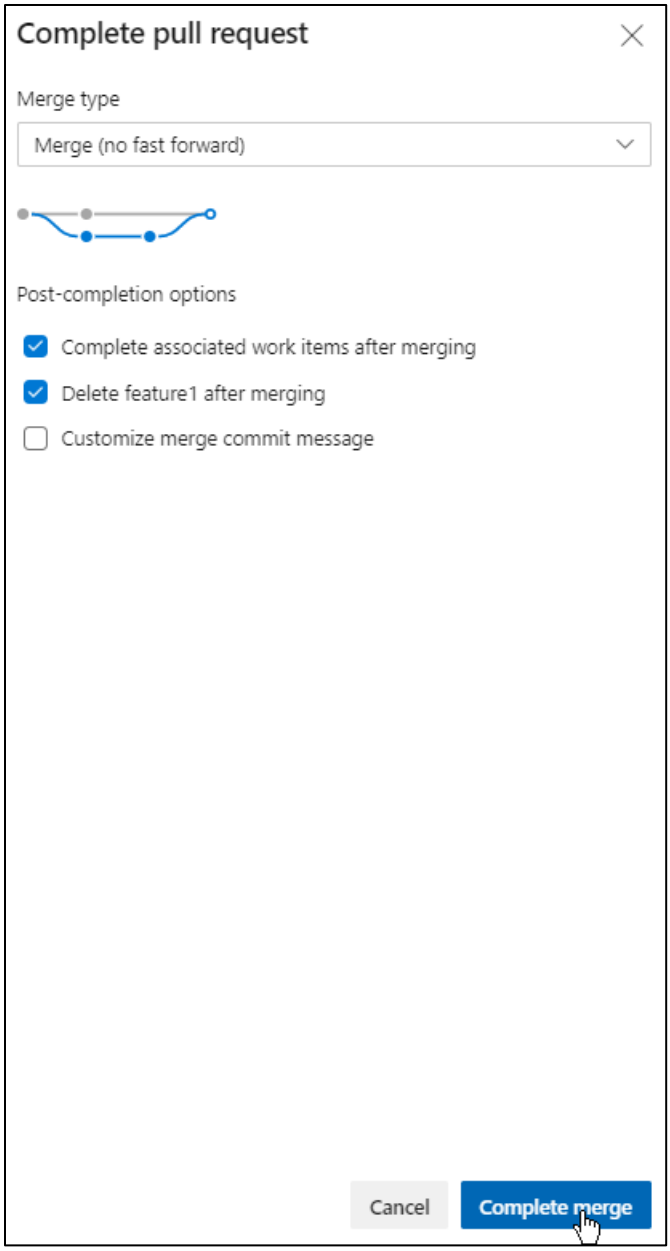
You are taken to the **Update the Pipeline description** panel. Typically, the approver would be the one managing this part of the process. We will use the same account all the time.



Click **Approve**.

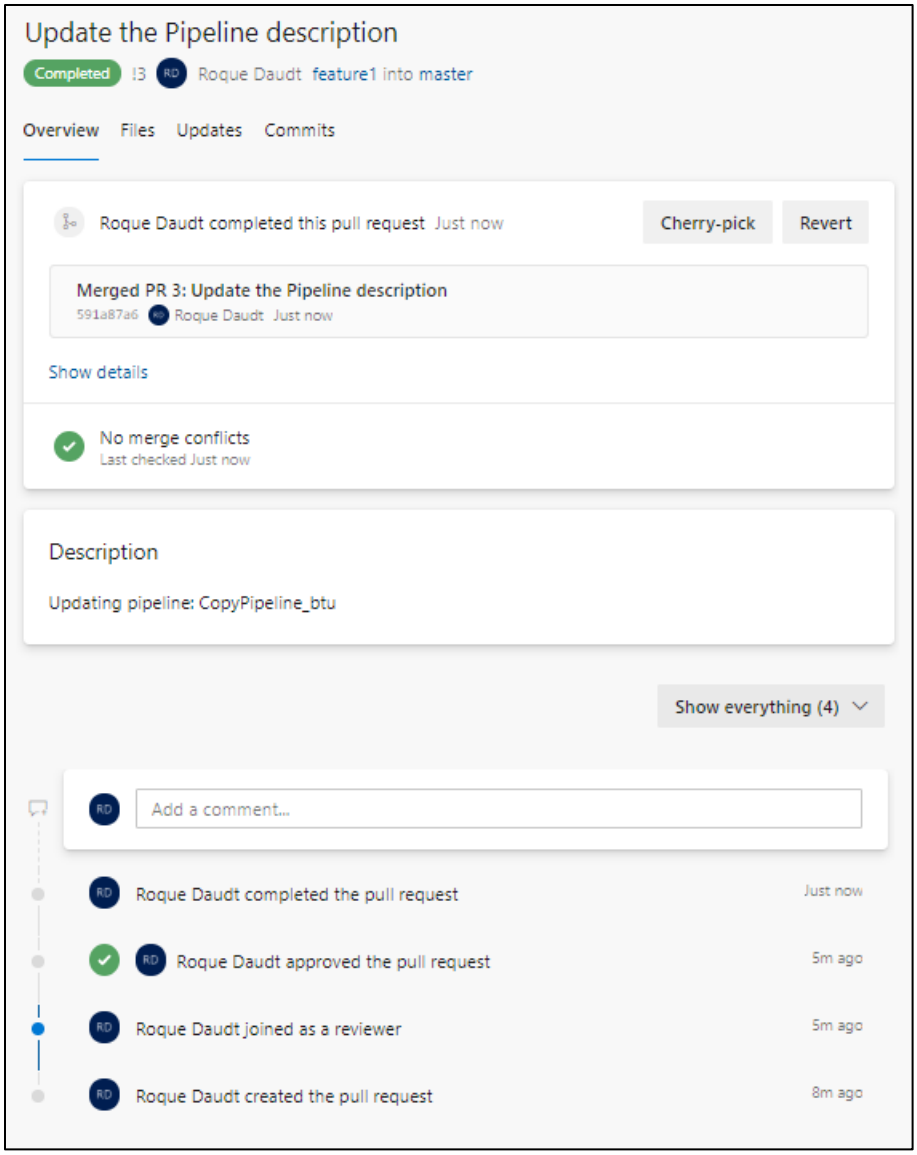


Click **Complete**.

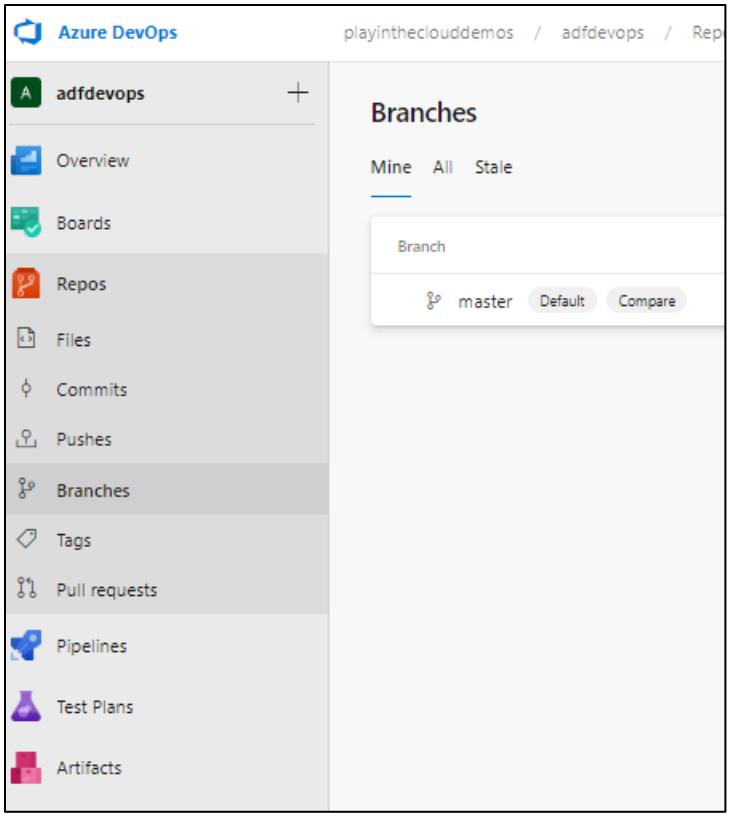


Leave the options as they are but note the option to delete or not the feature1 branch after the merge.

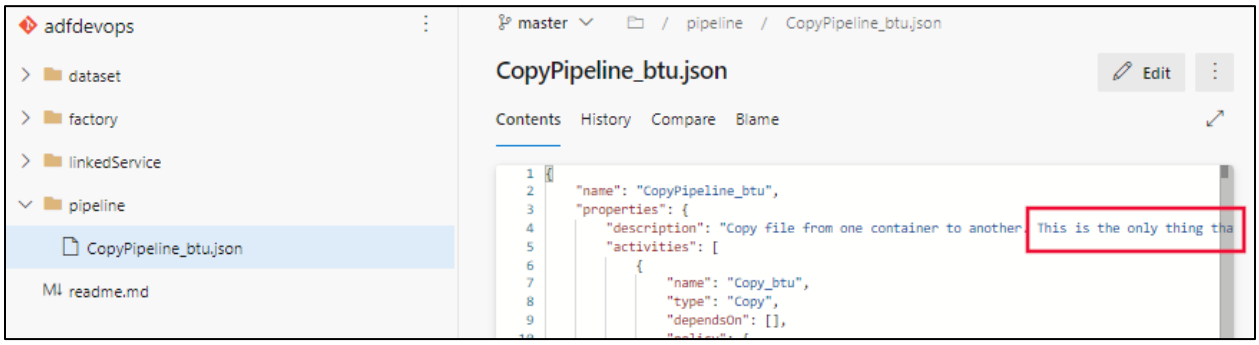
Click **Complete merge**. Upon a successful merge, this is what you see.



Now, if we click **Branches** in Azure DevOps, we only see the **master** branch, because we requested **feature1** branch to be deleted after the merge.



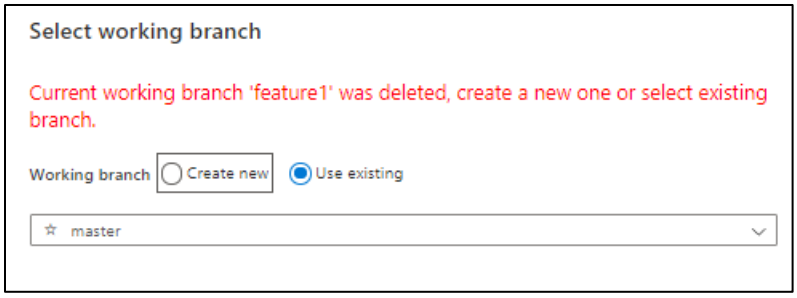
If we click on **Files**, we see the changed file in the master branch.



Publish to adf_publish branch

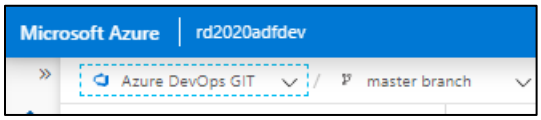
Next, we need to publish the master branch to the adf_publish.

Go back to ADF Dev. If you deleted the feature1 branch you should see a panel like this.

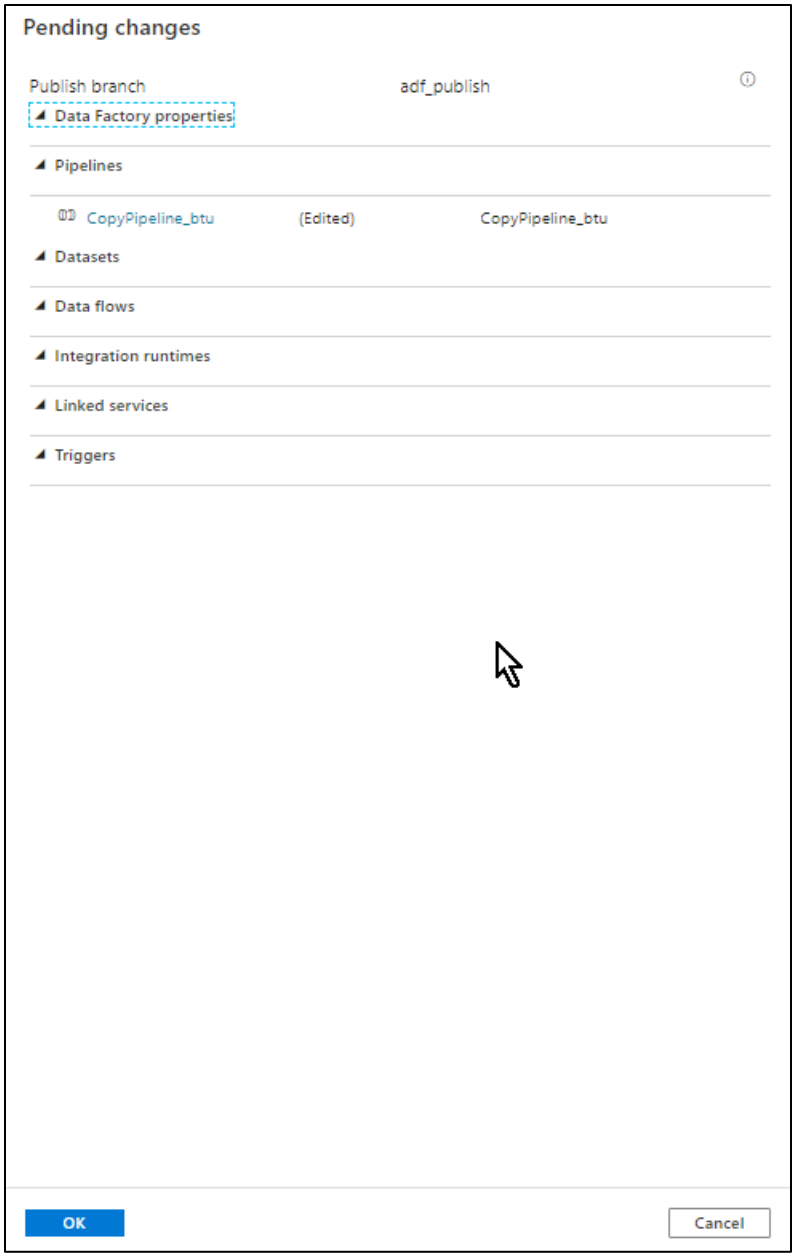


Select **Use existing** and click **Save** to set master as the current branch in your ADF Dev. It is important to understand that **you can only publish to the adf_publish branch from the master branch**.

You can confirm that you are in the **master** branch by checking ADF's top bars.

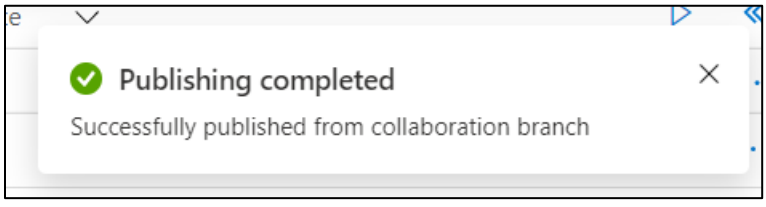


Click **Publish**. ADF opens the panel below.

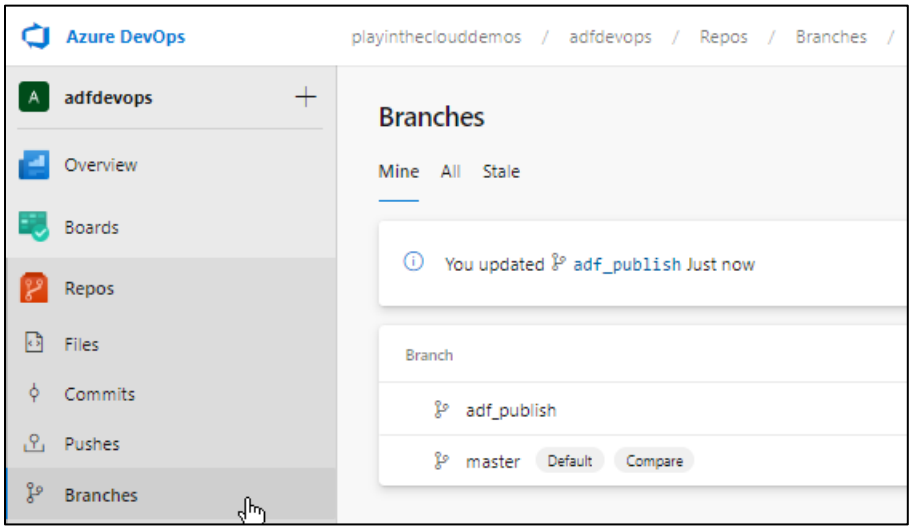


Click OK.

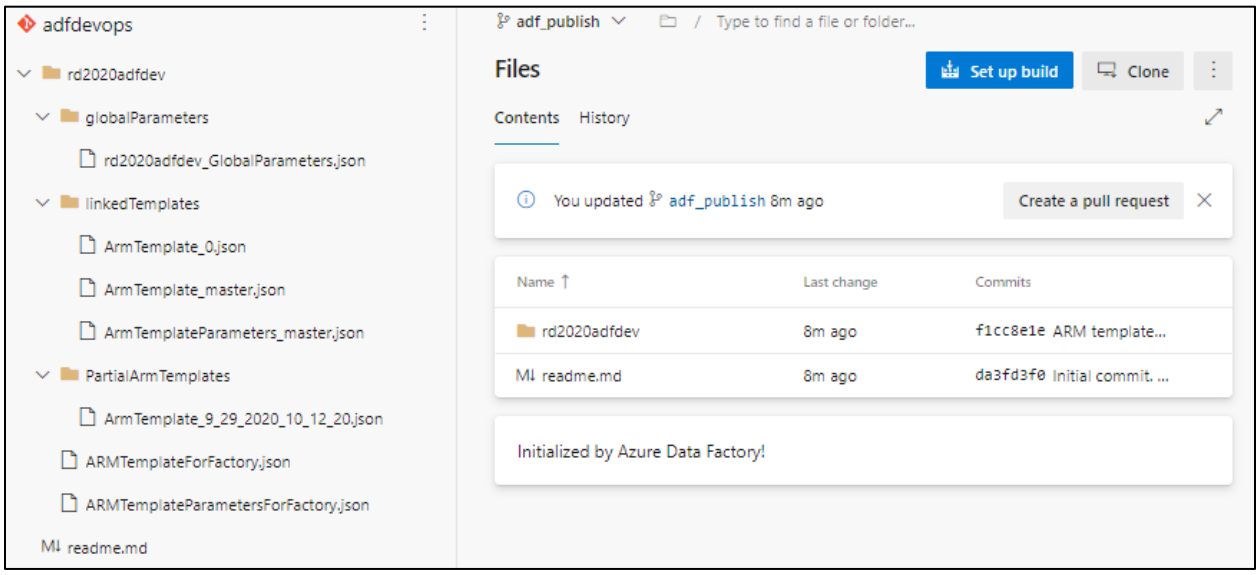
You will se several pop-up message on the top right of the ADF window. Hopefully, the messages indicate a successful operation, such as the one below.



Go back to Azure DevOps and click Branches. Now, for the first time we see the **adf_publish** branch. It is created the first time that we publish.

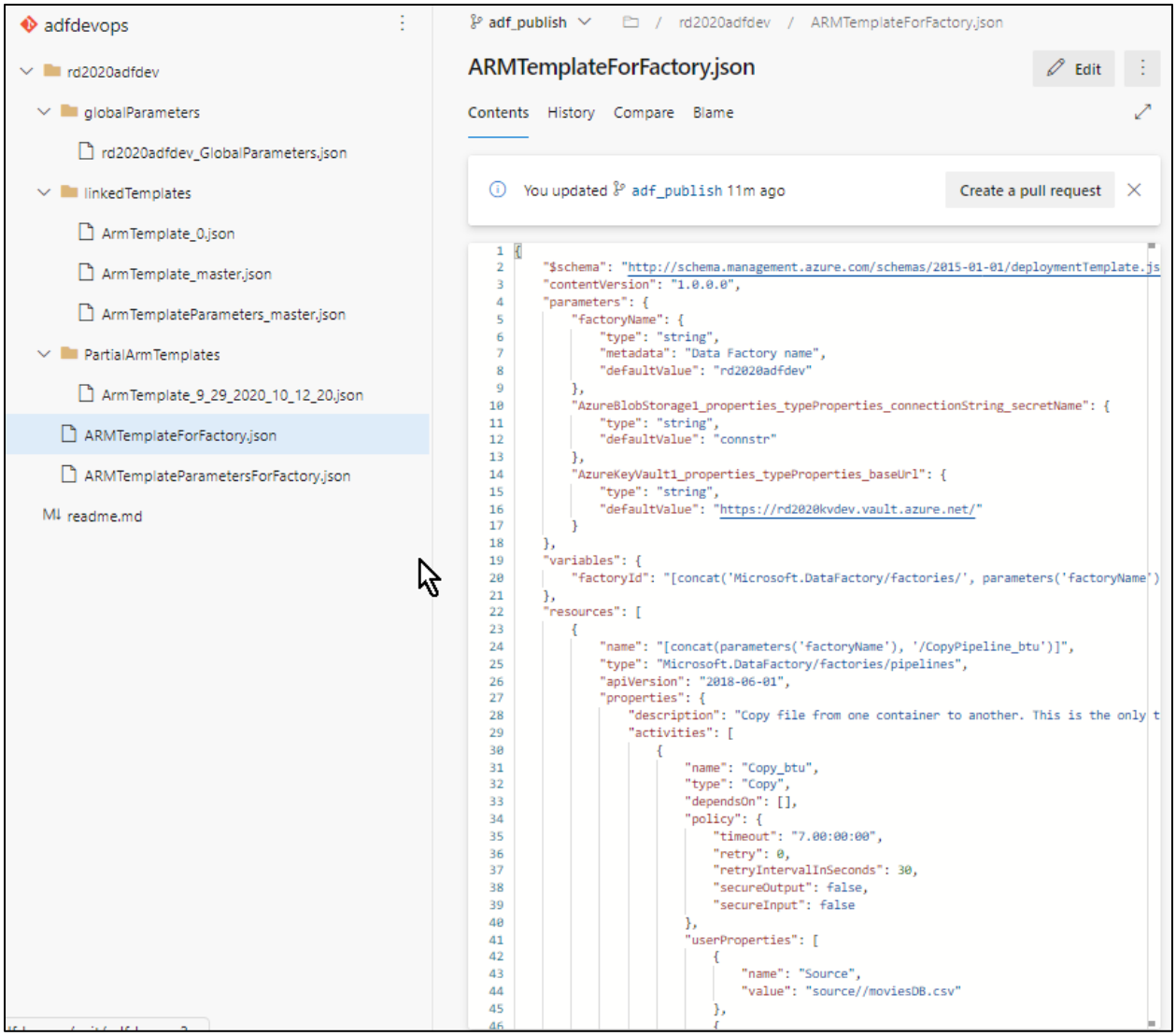


Click the **adf_publish** branch and explore the files. You see that it is a different folder structure and a different set of files, when compared to the master branch. This so because what happened as per our publish request is that a set of ARM templates were created based on existing files in the master branch. They are different things and there is not a one-to-one relation between them.



These labs are not about ARM but it helps to explore the newly-created ARM templates a bit in order to understand what happen in the next labs.

First, you want to note that there is a file named **ARMTemplateForFactory.json**. This is the main template; it describes the resources that will be created when these templates are processed in the higher environments.

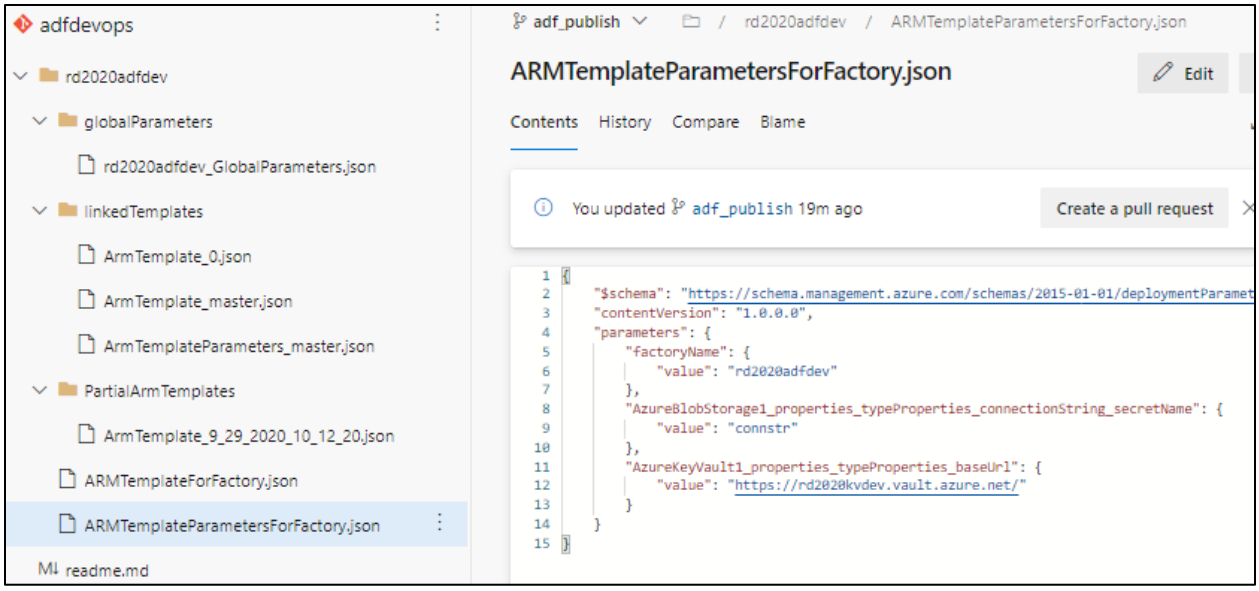


In the image above, the description of the pipeline object, for an instance, starts at line 24. If you scroll down the file you will descriptions like that for all other objects: datasets, linked services, etc.

The initial 18 lines are important, too. They present parameters that need to be passed to this template at runtime. As you can see, there are three parameters:

- **Factoryname**
- **AzureBlobStorage1_properties_typeProperties_connectionString_secretName**
- **AzureKeyVault1_properties_typeProperties_baseUrl**

Click now on the **ARMTemplateParametersForFactory.json** file. Unsurprisingly, you will find there the three parameters discussed in the paragraph above.



We will be back talking about these parameters in the next lab.

One last note about the process described in this lab is that Microsoft's "official" method for integration with ADF actually requires some manual intervention, as experienced in the lab.

Next

Now that **adf_publish** is configured we will explore how to create a release pipeline to deploy it to the test and production environments. Go To **06 – Continuous Delivery**.