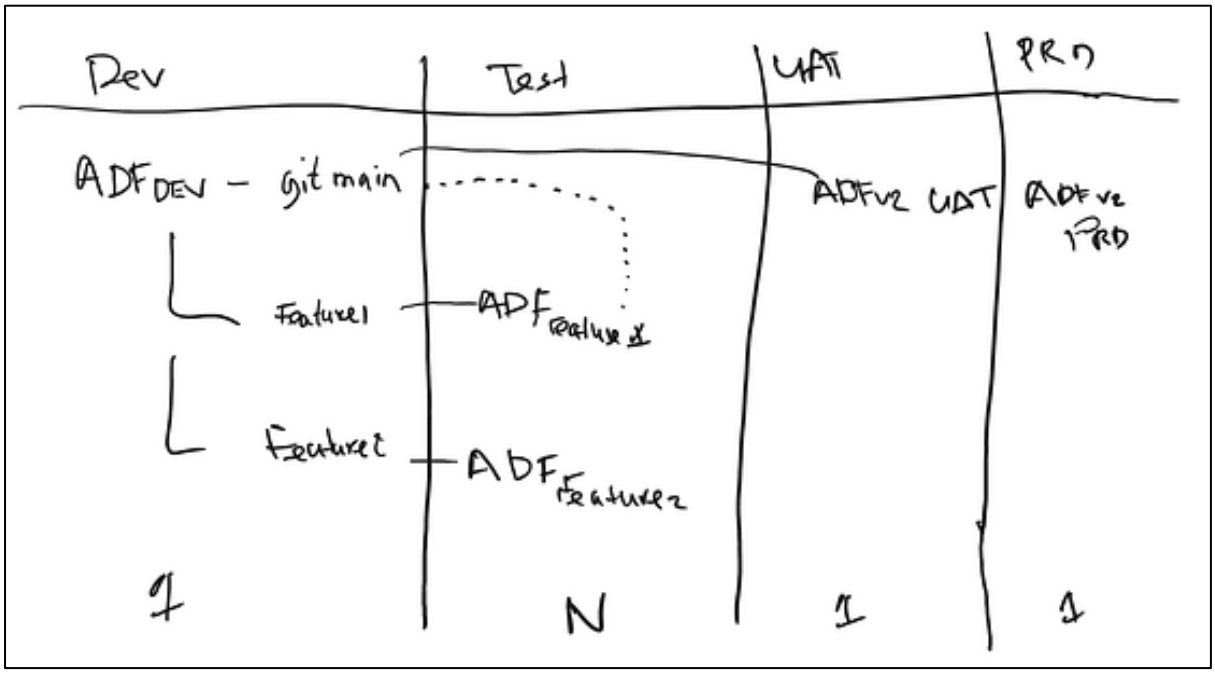
End to End Process, ADF From DEV to PROD

1. Introduction

In software projects, DEV/TST/UAT environments are used to develop and test code. After that, code is released to the PRD environment. Azure Data Factory (ADF) projects follow the same phased approach. However, the following ADF challenges need to be taken care of:

- ADF project can only be deployed as a whole rather than deploying a single pipeline. In this, it shall be prevented that untested code ends up in production.
- ADF supports branching and pull requests, however, it is hard to review changes in JSON. Reviewing changes can only be done in ADF itself, either using a different branch in DEV or run in TST
- ADF pipelines cannot be tested in isolation, however, by fixed data sources and pytest libraries, unit and integration testing can be achieved. See my previous blog.



In this, the main idea is as follows:

- 1 ADF instance is used by multiple developers to create features. Each feature gets its own branch. In case too many developers are working on same ADF instance (>10), it can be decided to create a new instance for a group of developers.
- Once a feature is ready to be tested, it is deployed to its own ADF instance using an ARM template. In case N features need to be tested, this results in N ADF instances. This way a new feature can be tested in isolation without having to merge to the main branch first.
- Once a feature is successfully, it is merged to the main branch and subsequently, the main branch is automatically deployed to the UAT environment. In the UAT environment, integration tests can be done. It could also be decided to merge to a dedicated UAT (pre release) branch, that is merged to main after testing in UAT is successfully.
- Once integration tests are successful, main branch is deployed to PRD. Notice that deployment to PRD always happens from main branch rather than feature branches. This is to prevent that merge conflicts can occur after PRD deployment, which, worst case, can result in revoking features already in PRD.

In the remaining of this blog, we discuss the detailed process to deploy to the different environments. Finally, notice that this blog follows the MSFT recommended new ADF CI/CD flow. In this flow, the NPM Azure Data Factory utilities library rather than ADF publish button is leveraged to propagate to different environments.

2. DEV

In the remainder of this blog, the project is deployed using the following steps:

- 2.1 Prerequisites
- 2.2 Setup Azure DevOps project
- 2.3 Setup ADF DEV instance
- 2.4 Create feature branch in ADF DEV

2.1 Prerequisites

The following resources are required in this tutorial:

- Azure Account
- Azure DevOps
- Azure CLI (recommended, also for troubleshooting)

Subsequently, go to the Azure portal and create a resource group in which all Azure resources will be deployed. This can also be done using the following Azure CLI command:

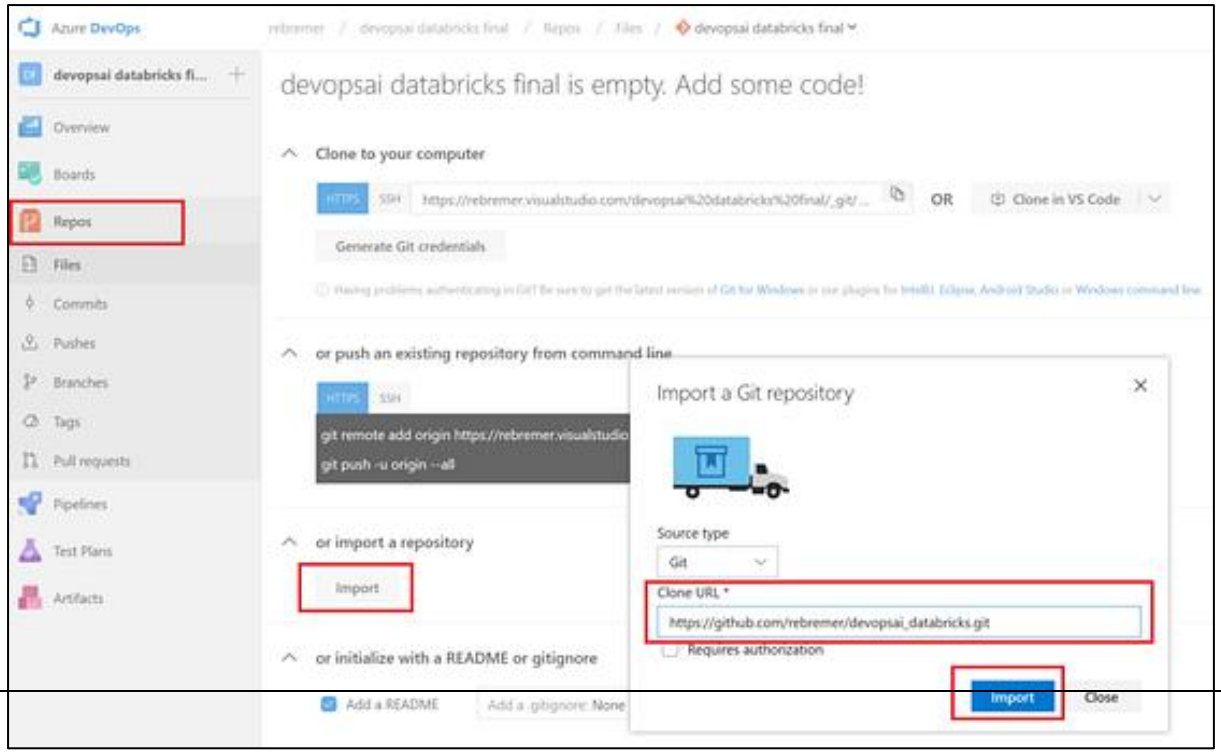
az group create -n <<your resource group>> -l <<your location>>

2.2 Create Azure DevOps project

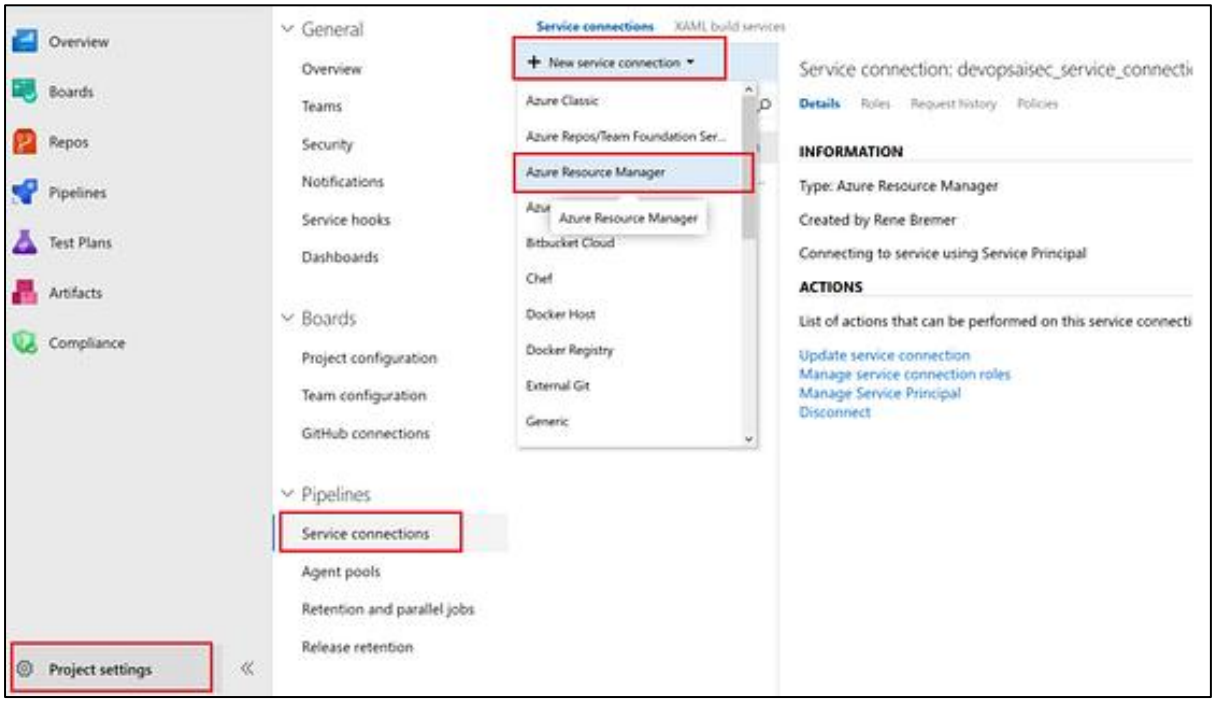
Azure DevOps is the tool to continuously build, test, and deploy your code to any platform and cloud. Once you created a new project, click on the repository folder and select to import the following repository:

- https://github.com/rebremer/azure-data-factory-cicd-feature-release

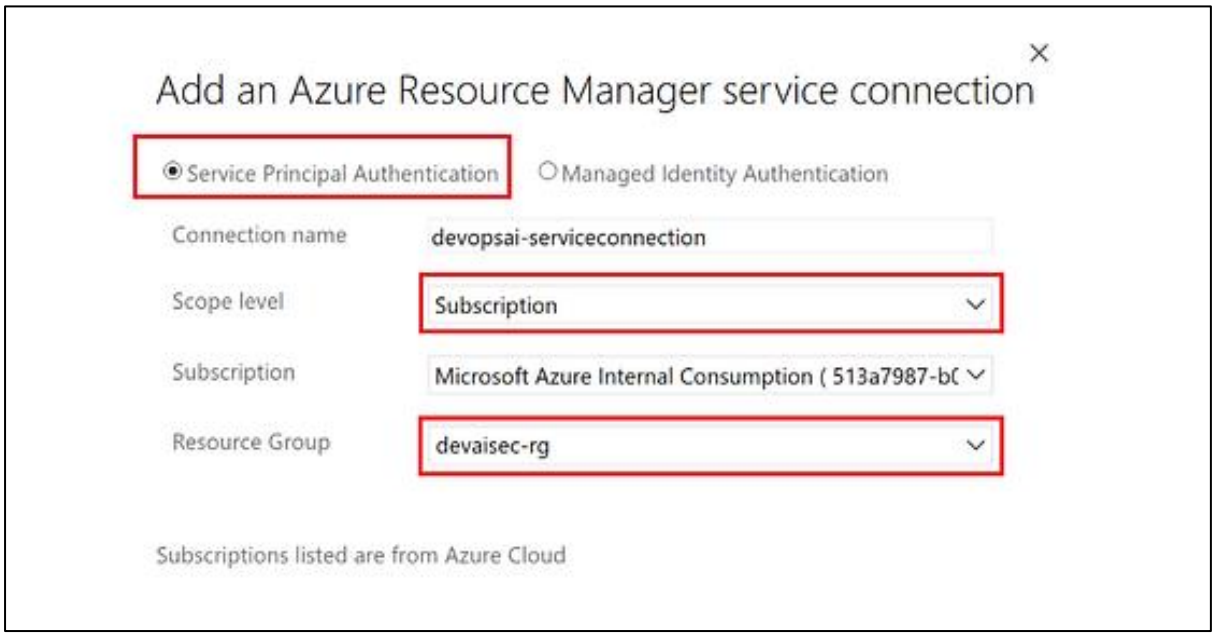
See also the picture below, in which a devops repo is created using the git repo of this blog:



Subsequently, the Service connection is needed to access the resources in the resource group fom Azure DevOps. Go to project settings, service connection and then select Azure Resource Manager, see also picture below.



Select Service Principal Authentication and limit scope to your resource group which you created earlier, see also picture below.

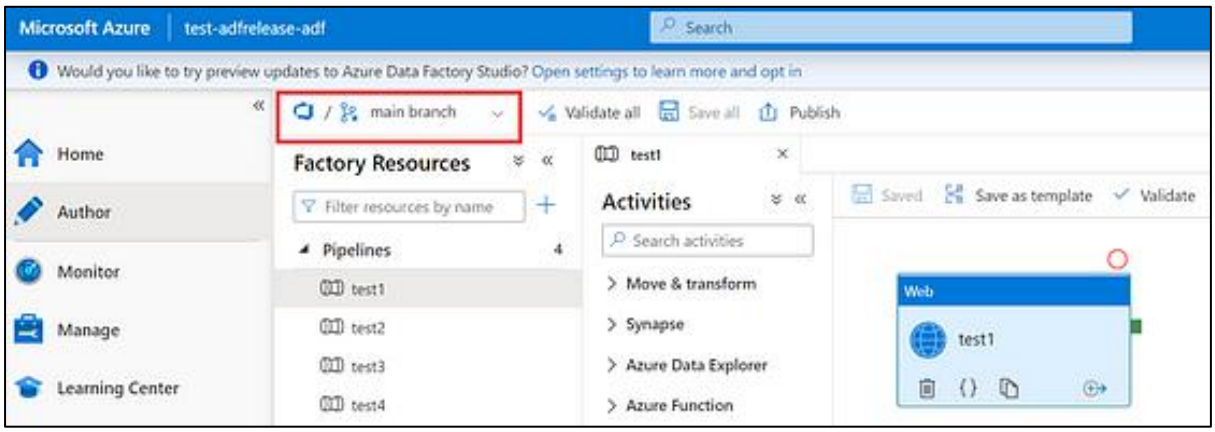


2.3 Setup ADF DEV instance

In this paragraph, the Azure Data Factory is created that is used for development. This ADF DEV instance shall use the Azure DevOps git repo that was created in step 2.2.

- In this link, it is explained how to create an Azure Data Factory instance in the portal
- In this link, it is explained how a code repository can be added to ADF. Make sure the devops project git repo of 2.2 is added.

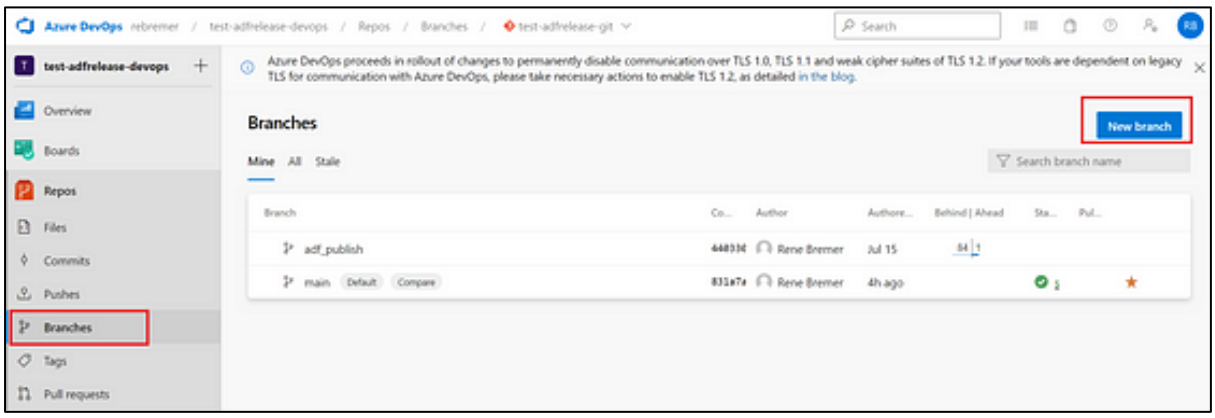
Alternatively, Azure CLI can also be used to create an data factory including adding code repository, see here. See below for a successfully deployed ADF instance that is linked to Azure DevOps git repo:



2.4 Create feature branch in ADF DEV

After everything is setup, development can start. A best practice in software development is to create development branches from the main branch. In this development branch, features are created by developers.

In this blog, it is key that new branches are created in Azure DevOps rather than ADF itself. This is because that Azure DevOps Pipeline yml file shall also be included in the new branch (ADF does not do that). Go to your Azure DevOps project, select branches and create new branch, see also below.



2.4 Create new branch in Azure DevOps

A developer can now also find the branch in ADF and can start coding in the branch. After a developer finished his coding is his branch, it shall be reviewed by other developers and tested. This is discussed in the next sub paragraph.

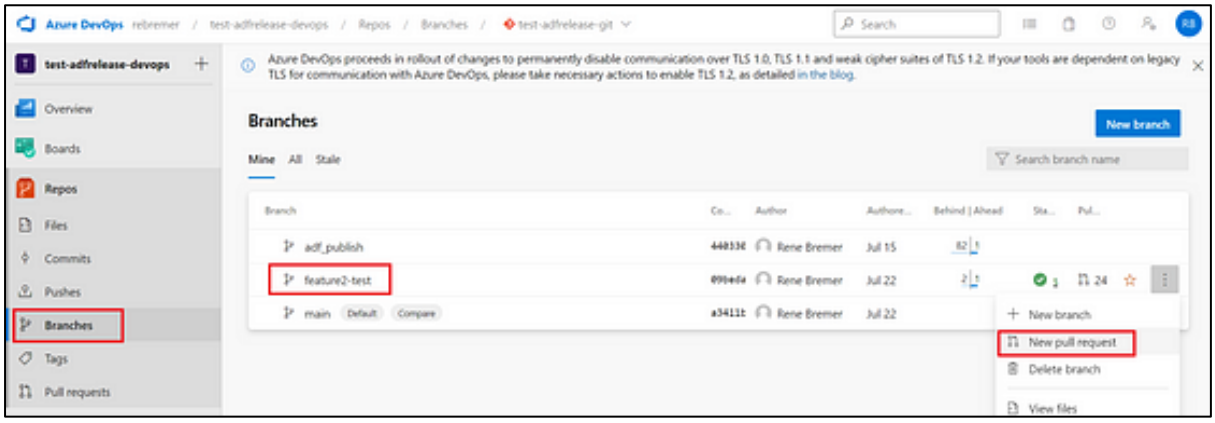
3. TST

In this chapter, a feature branch is reviewed and deployed in ADF instance for testing. The following steps are executed:

- 3.1 Create Pull Request (PR)
- 3.2 Deploy feature branch to a new ADF TST instance

3.1 Create pull request (PR)

After a developer finished its branch, a PR shall be created such that other people can review. Go to the Azure DevOps project, go to branches, select your feature branch and select new pull request, see below



Fill in the form and fill in the names of developers that shall review the changes. An email will be sent to these developers to notify them.

3.2 Deploy feature branch to a new ADF TST instance

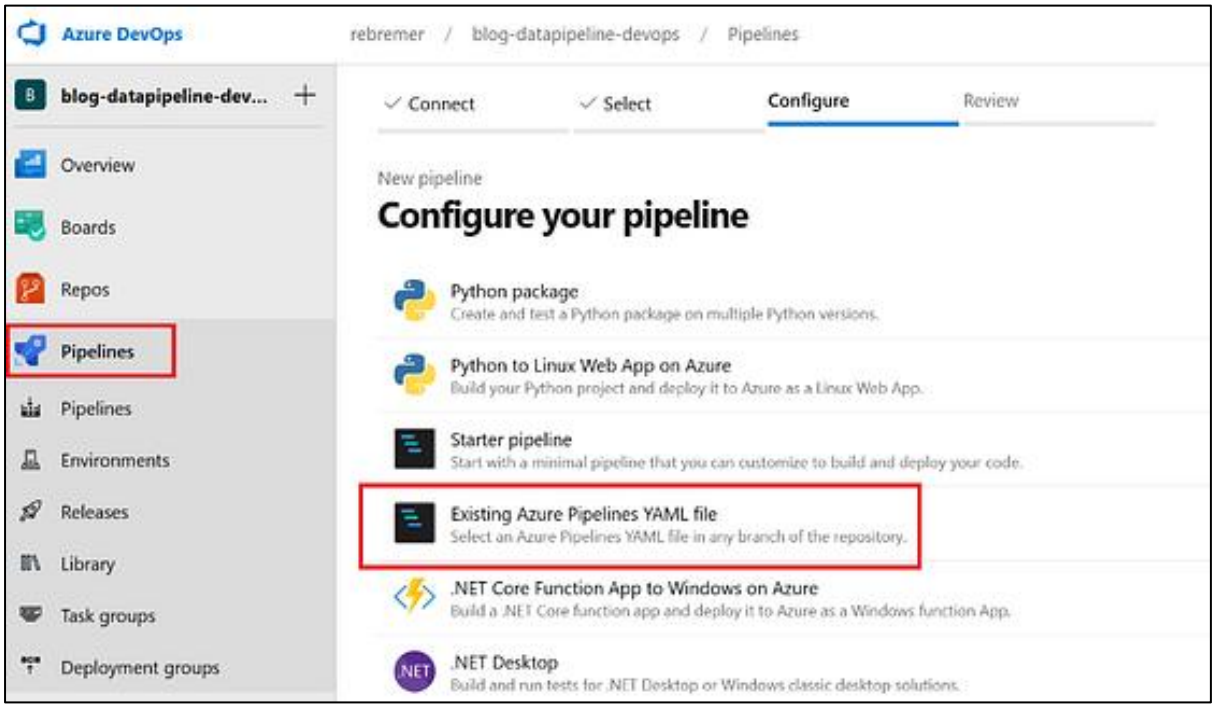
As discussed in the introduction, it is hard to review changes in ADF pipelines from JSON. A PR shall always be reviewed in ADF instance itself. Three different ways to do this are as follows:

- Go to ADF DEV instance, select the branch and review the changes. Testing can be a challenge, since branch is still in DEV environment
- Deploy changes to a generic, existing ADF TST instance. Deploying code to existing pipeline will overwrite the existing ADF instance.
- Deploy changes to a new, dedicated ADF TST instance having the name of the feature branch. Changes can also be tested in isolation.

The new, dedicated ADF TST instance approach is taken in this sub paragraph. Go to your Azure DevOps project, to your repo, find azure-pipelines-release.yml and adapt the values to point to your workspace, see also below.

```
variables:
  adfdev : "<<your dev ADF instance>>"
  adftst : $(Build.SourceBranchName)
  subiddev: "<<your dev subscription id>>"
  subidtst: "<<your tst subscription id>>"
  rgdev : "<<your dev resource group>>"
  rgtst : "<<your tst resource group>>"
  location : "<<your location>>"
  AzureServiceConnectionId: "<<your AzureServiceConnectionId>>"
```

Now select Pipelines in your DevOps Project and click “New pipeline”. Go to the wizard, select the Azure Repos Git and the git repo you created earlier. In the tab configure, choose “Existing Azure Pipelines YAML file” and then azure-pipelines-release.ymlthat can be found in the git repo, see also below.



Once the pipeline is created, it is run immediatelly, and when ran successfully, a new data factory instance is created with the name of the feature branch. After a the feature branch is reviewed and tested correctly, it can be decided to merge it to main again. This will be discussed in the next paragraph.

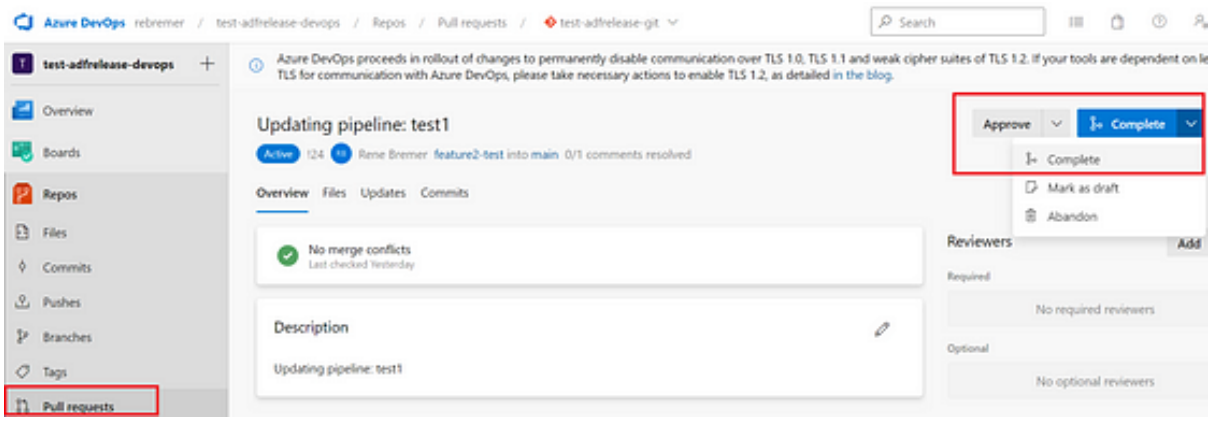
4. UAT and PRD

In the remainder of this blog, the project is deployed using the following steps:

- 4.1 Merge feature branch to main branch
- 4.2 Deploy main branch to ADF UAT
- 4.3 Deploy main branch to ADF PRD

4.1 Merge feature branch to main branch

After the feature branch is reviewed and tested correctly, it shall be merged with the main again. Select the pull request, add a comment and mark the PR as complete, see also image below.



The feature branch is now merged to the main branch. As a next step, regression test shall be run in the UAT environment on the main branch. That will be discussed in the next paragraph.

4.2 Deploy main branch to ADF UAT

The main branch is always to deploy to PRD. This is done to make sure that only 1 code base exists that is used in production. However, after merging a feature to main done in chapter, a couple of regression tests shall be run first to make sure the main branch does not contain conflicting changes. In this paragraph, the main branch deployed to UAT.

Go to your Azure DevOps project, to your repo, find azure-pipelines-release.txt and adapt the values to point to your workspace, see also below

```
variables:
  adfdev : "<<your dev ADF instance>>"
  adfuat : "<<your uat ADF instance>>"
  subiddev: "<<your dev subscription id>>"
  subiduat: "<<your uat subscription id>>"
  rgdev : "<<your dev resource group>>"
  rguat : "<<your uat resource group>>"
  location : "<<your location>>"
AzureServiceConnectionId: "<<your AzureServiceConnectionId>>"
```

Now take similar steps described in paragraph 3.2 to create and run the pipeline. Two additional remarks:

- azure-pipelines-release.yml contains the following snippet of code: trigger: main . This implies that whenever there is a change in the main branch (e.g. when a feature branch is merged), the pipeline is run and tests can be executed.
- Creating a Azure DevOps pipeline to deploy to UAT only needs to be once

When the tests are executed successfully, the code is ready to be deployed in PRD. This is discussed in the next chapter

4.3 Deploy main branch to ADF PRD

When regression tests were successful in UAT, the main branch is ready to deployed in PRD. This can simply be done by adding an deploy ADF to PRD task in the pipeline deployed in 4.2. Two addition remarks on regression testing and deployment to PRD:

- Automated testing ADF can be challenge. In my previous blog How to build unit tests for Azure Data Factory, it is extensively discussed how to setup a test framework using DevOps and pytest
- To prevent that a release is automatically deployed to PRD, a manual validation step can be added to the pipeline. This is also discussed in my previous blog and can be found in accompanied azure pipeline example

5. Conclusion

In a typical software project, DEV/TST/UAT environments are used to develop and test code. After that, it is released to the PRD environment. Azure Data Factory (ADF) pipelines follow the same release cycle. However, an ADF project can only be deployed as a whole and there is a risk that untested code is deployed into production. In this blog and git repoazure-data-factory-cicd-feature-release , it is described how ADF releases can be managed, see also overview below.