**CICD in Synapse SQL**

**How to deliver your database objects across multiple environments**

The purpose of this document is to provide a step-by-step guide on how to set up continuous integration and continuous delivery for a Synapse workspace using Azure DevOps. This includes creating a new project, installing the Synapse Workspace Deployment extension, committing changes to the repository, creating a build pipeline, and setting up a release pipeline to deploy changes to the target SQL pool.

1. **CI Stage: Continuous Integration**
2. Make sure you have an Azure DevOps Organization (<https://aex.dev.azure.com/>) that is connected to your Tenant’s AAD:  
     
   thumbnail image 1 of blog post titled 
    
    
     
    
    
    
       
     
      
       
         
          How to use CI/CD integration to automate the deploy of a Synapse Workspace to multiple environments
          
         
        
      
     
    
      
    
    
    
    
    
   
3. Create a new project under your Azure DevOps Organization. Name the project “**synapse-cicd-demo**” and for demonstration purposes, keep the project visibility “**Private**”. Select “**Create**” to provision your new project.  
   thumbnail image 2 of blog post titled 
    
    
     
    
    
    
       
     
      
       
         
          How to use CI/CD integration to automate the deploy of a Synapse Workspace to multiple environments
          
         
        
      
     
    
      
    
    
    
    
    
   
4. Make sure you have installed the **"Synapse Workspace Deployment" Microsoft**extension in your Azure DevOps Organization. You can check this by navigating to your "**Azure DevOps**" home page and by clicking on "**Organization settings**" on the bottom left corner. From the left-menu, under "**General**" select "**Extensions**". Look under the "**Installed**" extensions. If you don't have the Synapse Workspace Deployment extension installed, just hit the "**Browse Marketplace**" button and search for **"Synapse Workspace Deployment"**(include double quotes in the search bar to narrow your search). Follow the installation instructions.
5. After the project is created, from **Repos** page in the Azure DevOps portal, initialize & click **Clone in Visual Studio**. When prompted, click **Allow**. This will start **Visual Studio** with the new project loaded.

A screenshot of a computer

Description automatically generated

1. Once the project loads, you will be prompted to clone from the server to your local path. Change the local path if required and click **Clone**.

A screenshot of a computer

Description automatically generated

1. Next, click **File** in the **New Project** from the menu bar. This will display the New Project dialog box. From there, click **SQL Server** in the list of installed templates, ensure to select **Add to Solution**, and click **Create**. This will create **Database1** project.

A screenshot of a computer

Description automatically generated

1. Once the solution is created, click **Solution Explorer**. In the **Solution Explorer** window, right click **Database1,** in the drop-down menu, click properties. Under Project Settings select **Microsoft Azure SQL Data Warehouse.**

A screenshot of a computer

Description automatically generated

1. Click **Solution Explorer**. In the **Solution Explorer** window, right click **Database1**, in the drop-down menu, click **Add** and click **Table**.

A screenshot of a computer

Description automatically generated

1. In the **Add New Item – Database 1** dialog box, ensure that **Table (Data Warehouse)** is selected and click Add.

*We will accept the default table definition, since our intention is to demonstrate the process of continuous integration and continuous deployment, however, in real-life scenarios, your database structure would obviously be more elaborate. The process of configuring described in this lab would not change in this case.*

1. Click **File** and, in the **File** menu, click **Save All**.

A screenshot of a computer

Description automatically generated

1. To commit the project to your repository, switch to the **Git Changes** window, enter a message describing the changes you made and click **Commit All**.

A screenshot of a computer

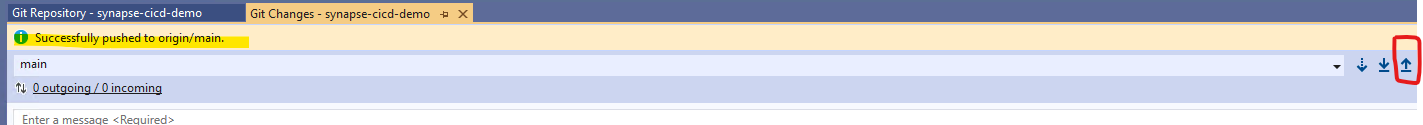
Description automatically generated

1. Next, click the **Sync** link and then, on Git Changes Explorer window.

A screenshot of a computer

Description automatically generated

1. Next, click **Push** to push changes to **Azure DevOps**.



1. Switch back to the Azure DevOps interface and refresh the Repos page. You Should now see Table1.

A screenshot of a computer

Description automatically generated

1. Under the **Pipelines** section, click on **Create Pipeline**. Select “Native Azure DevOps Experience” to switch to classic editor.

A white text on a white background

Description automatically generated

thumbnail image 14 of blog post titled 
 
 
  
 
 
 
    
  
   
    
      
       CICD in Synapse SQL: How to deliver your database objects across multiple environments
       
      
     
   
  
 
   
 
 
 
 
 


1. Select the source that is hosting your Git repo, in my case I'm selecting "**Azure Repos Git**"

thumbnail image 15 of blog post titled 
 
 
  
 
 
 
    
  
   
    
      
       CICD in Synapse SQL: How to deliver your database objects across multiple environments
       
      
     
   
  
 
   
 
 
 
 
 


1. After selecting "**Continue**", select the "**.NET Deskop**" template and hit the "**Apply**" button

thumbnail image 16 of blog post titled 
 
 
  
 
 
 
    
  
   
    
      
       CICD in Synapse SQL: How to deliver your database objects across multiple environments
       
      
     
   
  
 
   
 
 
 
 
 


1. Name your pipeline (In my case: "Synapse DacPac Automation-CI") and make sure you remove all template tasks **except these below**:

thumbnail image 17 of blog post titled 
 
 
  
 
 
 
    
  
   
    
      
       CICD in Synapse SQL: How to deliver your database objects across multiple environments
       
      
     
   
  
 
   
 
 
 
 
 


1. You can use these variables when configuring your tasks:

**Build:**

**A screenshot of a computer

Description automatically generated**

**Copy Files to:**

**A screenshot of a computer

Description automatically generated**

**Publish Artifact:**

**A screenshot of a computer

Description automatically generated**

1. Select "**Save & Queue**" and when prompted to run the pipeline, select "**Save and Run**".

thumbnail image 18 of blog post titled 
 
 
  
 
 
 
    
  
   
    
      
       CICD in Synapse SQL: How to deliver your database objects across multiple environments
       
      
     
   
  
 
   
 
 
 
 
 


1. Wait for the pipeline to run to finish and then click in your agent job name (in my case "Agent job 1") under the "**Jobs**" blade to get more details about the pipeline execution.

thumbnail image 19 of blog post titled 
 
 
  
 
 
 
    
  
   
    
      
       CICD in Synapse SQL: How to deliver your database objects across multiple environments
       
      
     
   
  
 
   
 
 
 
 
 


1. From the left menu, you can see the pipeline tasks and you can click in the "**Build solution**" task to get more details about this build. The highlighted code represents the Dacpac file that was generated.

A screenshot of a computer program

Description automatically generated

1. Once you confirm that the dacpac file has been generated, you can move forward to the third and final stage: the CD stage.

**B. CD Stage: The Continuous Delivery**

Once you have completed all the steps above to generate your dacpac file, you will use the Azure DevOps Release Pipeline to set up the Continuous Delivery process and publish the generated Dacpac file to your target SQL pool.

1. From your Azure DevOps project menu, select "**Pipelines**" and then select "**Releases**" to create a new Release Pipeline. Create a New Pipeline.

thumbnail image 21 of blog post titled 
 
 
  
 
 
 
    
  
   
    
      
       CICD in Synapse SQL: How to deliver your database objects across multiple environments
       
      
     
   
  
 
   
 
 
 
 
 


1. When prompted to select a template, select "**Empty job**"

thumbnail image 23 of blog post titled 
 
 
  
 
 
 
    
  
   
    
      
       CICD in Synapse SQL: How to deliver your database objects across multiple environments
       
      
     
   
  
 
   
 
 
 
 
 


1. Name your release pipeline (in this case: Dacpac Automation CD) and then select "**+ Add an artifact**". This will define a source location to be used by your release tasks.

thumbnail image 24 of blog post titled 
 
 
  
 
 
 
    
  
   
    
      
       CICD in Synapse SQL: How to deliver your database objects across multiple environments
       
      
     
   
  
 
   
 
 
 
 
 


1. When configuring your artifacts, select the source type as "**Build**". Then select your build pipeline that was created in the previous stage (in this case Dacpac Automation - CI ). Hit the "**Add**" button to create your artifact.

thumbnail image 25 of blog post titled 
 
 
  
 
 
 
    
  
   
    
      
       CICD in Synapse SQL: How to deliver your database objects across multiple environments
       
      
     
   
  
 
   
 
 
 
 
 


1. Configure your release stage1 by selecting the "**1 job , 0 task**" link under your stage name.

thumbnail image 26 of blog post titled 
 
 
  
 
 
 
    
  
   
    
      
       CICD in Synapse SQL: How to deliver your database objects across multiple environments
       
      
     
   
  
 
   
 
 
 
 
 


1. From your Agent Job task, click on the plus (+) sign to add a new task to your Release Pipeline.

thumbnail image 27 of blog post titled 
 
 
  
 
 
 
    
  
   
    
      
       CICD in Synapse SQL: How to deliver your database objects across multiple environments
       
      
     
   
  
 
   
 
 
 
 
 


1. Search for "**Data Warehouse**" and select "**Add**" to add the Azure SQL Data Warehouse deployment task to your release.

thumbnail image 28 of blog post titled 
 
 
  
 
 
 
    
  
   
    
      
       CICD in Synapse SQL: How to deliver your database objects across multiple environments
       
      
     
   
  
 
   
 
 
 
 
 


1. Before configuring this task, you can create some variables to avoid exposing some task parameter values. Select the "**Variables**" tab and add the following variables to your release pipeline:

A screenshot of a computer

Description automatically generated

**Note:** you can secure any password in DevOps by using the locker icon when adding the variable. This will change the variable type to secret.

1. You can use these variables when configuring your task. Just type $(variable\_name). Choose "**SQL DACPAC file**" as your deploy type and browse your .dacpac file from your artifact location.

A screenshot of a computer

Description automatically generated

1. When browsing your dacpac file you should be looking into a path similar to this one below:

thumbnail image 32 of blog post titled 
 
 
  
 
 
 
    
  
   
    
      
       CICD in Synapse SQL: How to deliver your database objects across multiple environments
       
      
     
   
  
 
   
 
 
 
 
 


1. Save your release and then select the "**Create Release**" option. Hit the "**Create**" button to trigger a new release.

thumbnail image 33 of blog post titled 
 
 
  
 
 
 
    
  
   
    
      
       CICD in Synapse SQL: How to deliver your database objects across multiple environments
       
      
     
   
  
 
   
 
 
 
 
 


1. You can check the release progress by selecting the "**Logs**" option.

thumbnail image 34 of blog post titled 
 
 
  
 
 
 
    
  
   
    
      
       CICD in Synapse SQL: How to deliver your database objects across multiple environments
       
      
     
   
  
 
   
 
 
 
 
 


1. Once the release is successfully finished, you can log into your target database to confirm the objects that have been deployed.

thumbnail image 35 of blog post titled 
 
 
  
 
 
 
    
  
   
    
      
       CICD in Synapse SQL: How to deliver your database objects across multiple environments
       
      
     
   
  
 
   
 
 
 
 
 
