Installing Datawarehouse reporting for **ROGER365.io**

Written by Erwin Wiegman

Date 21-12-2023
Version 1.5

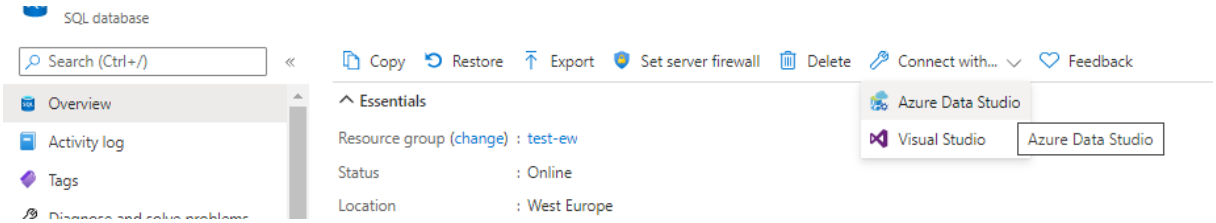
Index

High-level overview of the steps

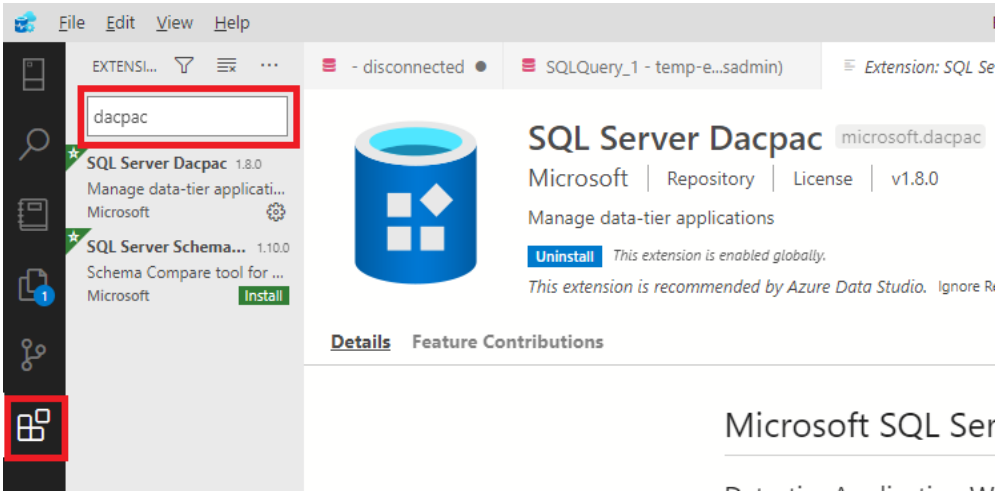
1) Create an Azure SQL database
2) Deploy reporting API schema to database
3) Install the SQL connector for logic apps
4) Install the Logic Apps

Install or update SQL schema with dacpac file

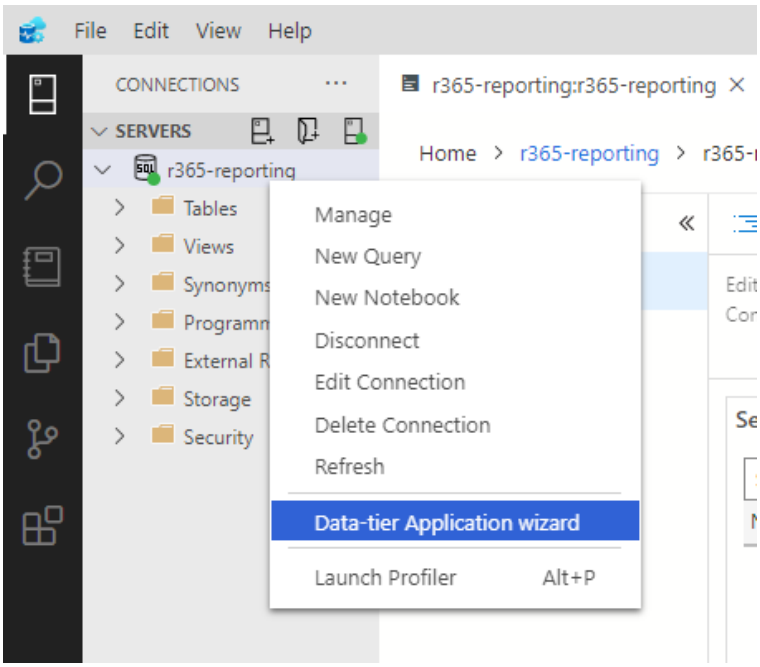
First create an Azure SQL database and connect to it with Azure data studio. This can be done from the azure portal on the Azure SQL server.



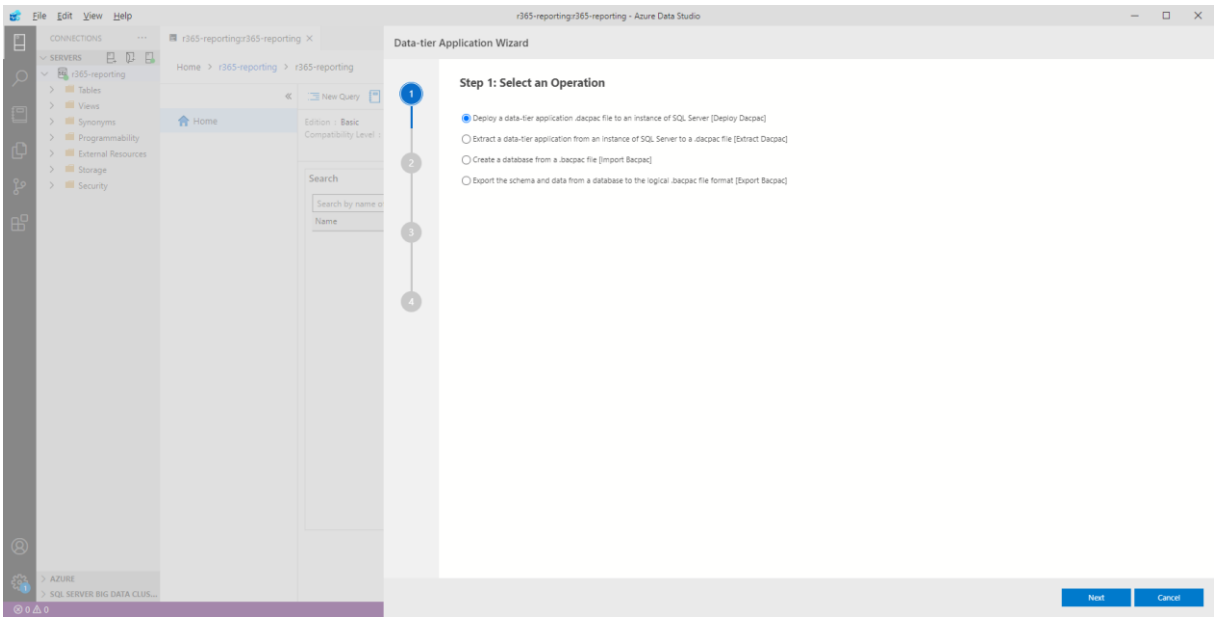
In Azure Data Studio the SQL Server Dacpac extension must be installed (one time). The extension menu item can be found on the left menu. Type dacpac in the search box, and install the extension.



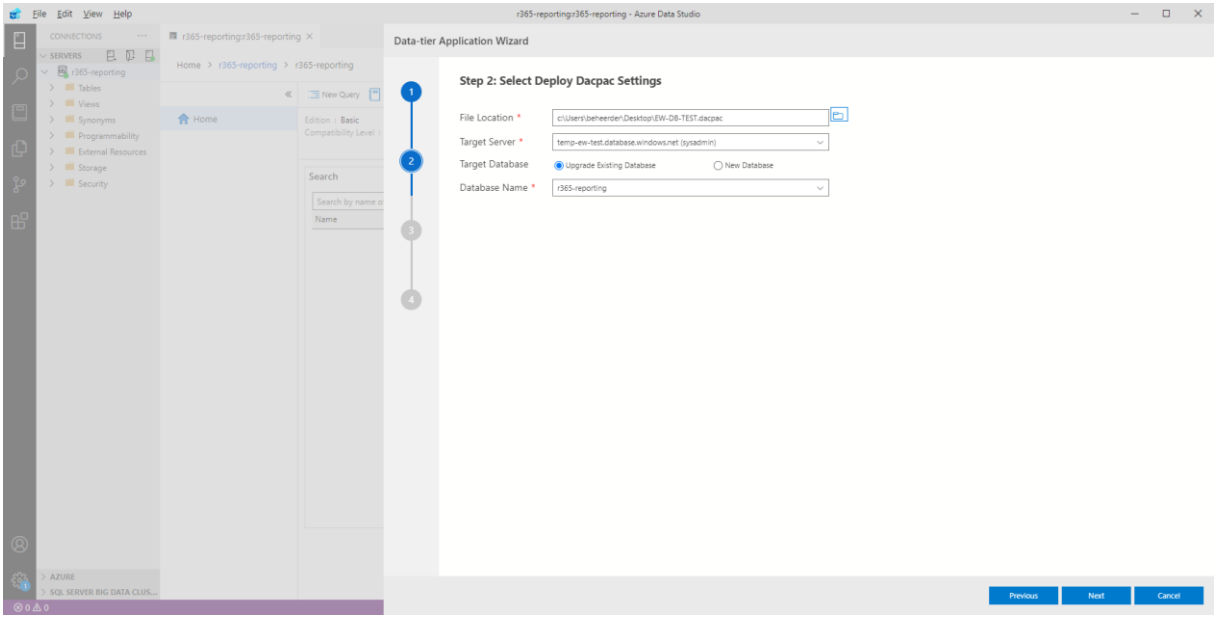
Connect you database to Azure Data Studio in the server section and open your reporting database



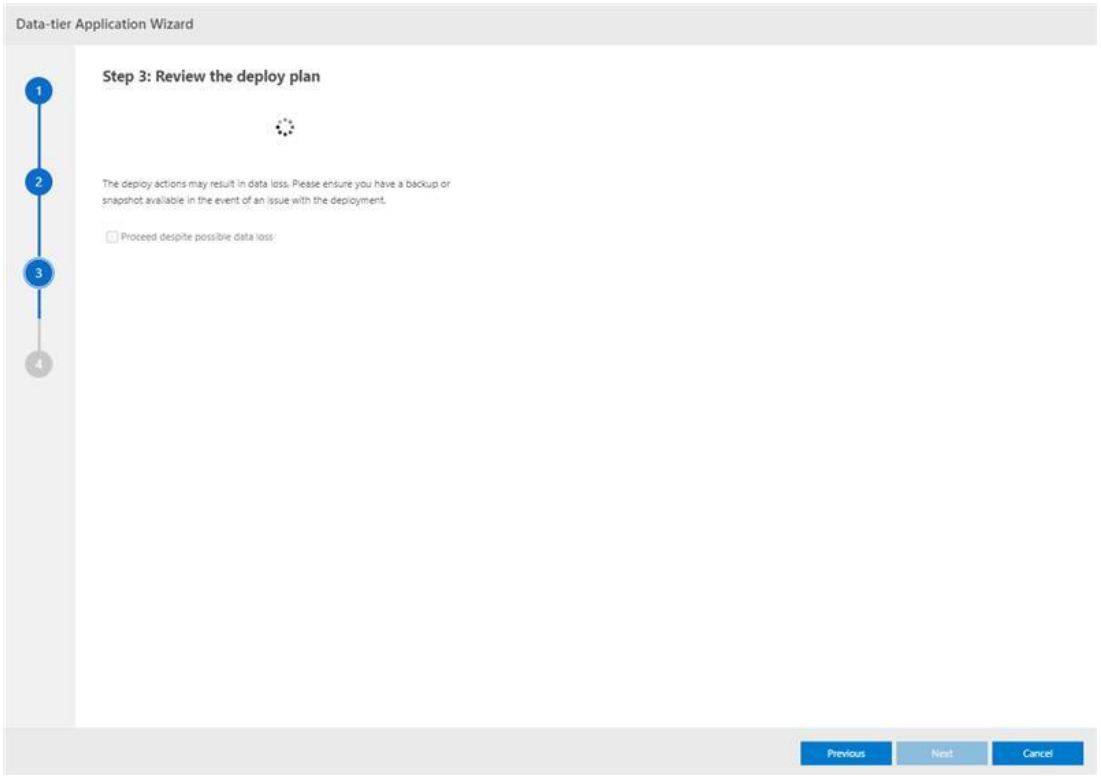
Choose the Data-tier Application wizard and in the wizard pop-up choose deploy a data-tier application .dacpac file to an instance of SQL server

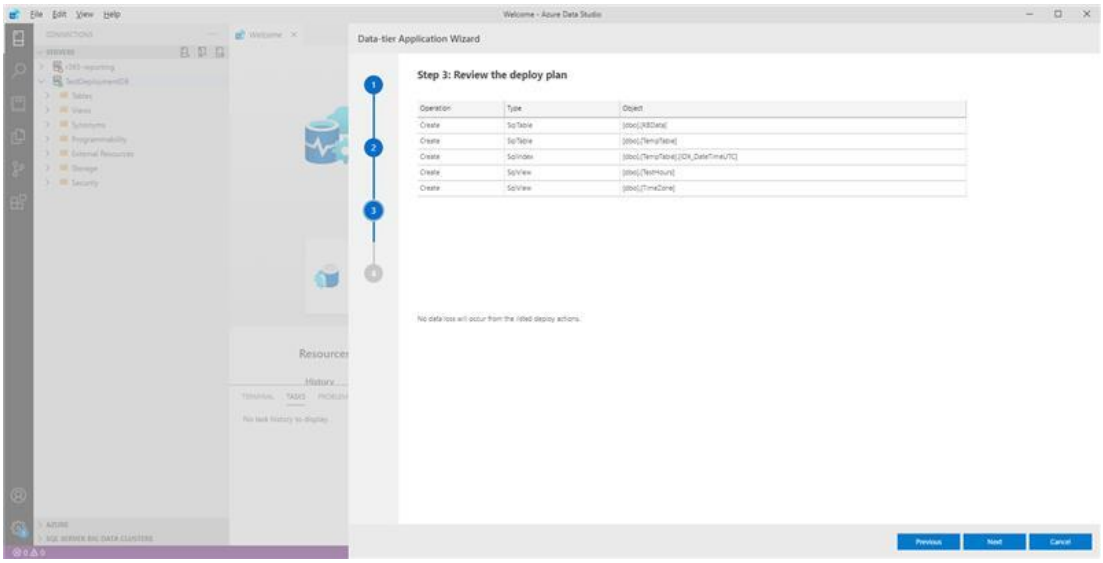


Point the wizard to the dacpac file. Which will create the database schema (or update it, because this is the same procedure).



The loading of the DACPAC can take some while. The process is checking the current database and the new version of the database and will display the changes that will be made after it is finished processing.



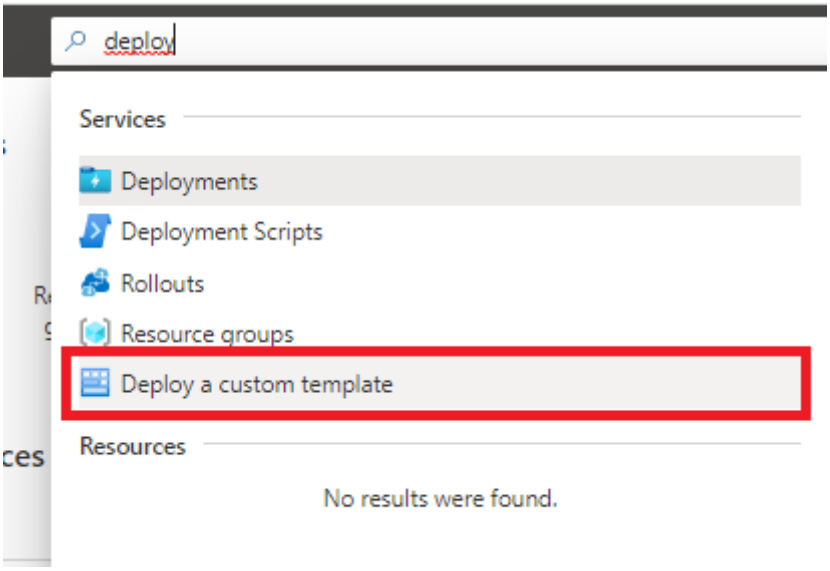


Go the through the wizard and deploy the dacpac file. After is has been deployed successful, you can refresh your database connection and all changes should be applied.

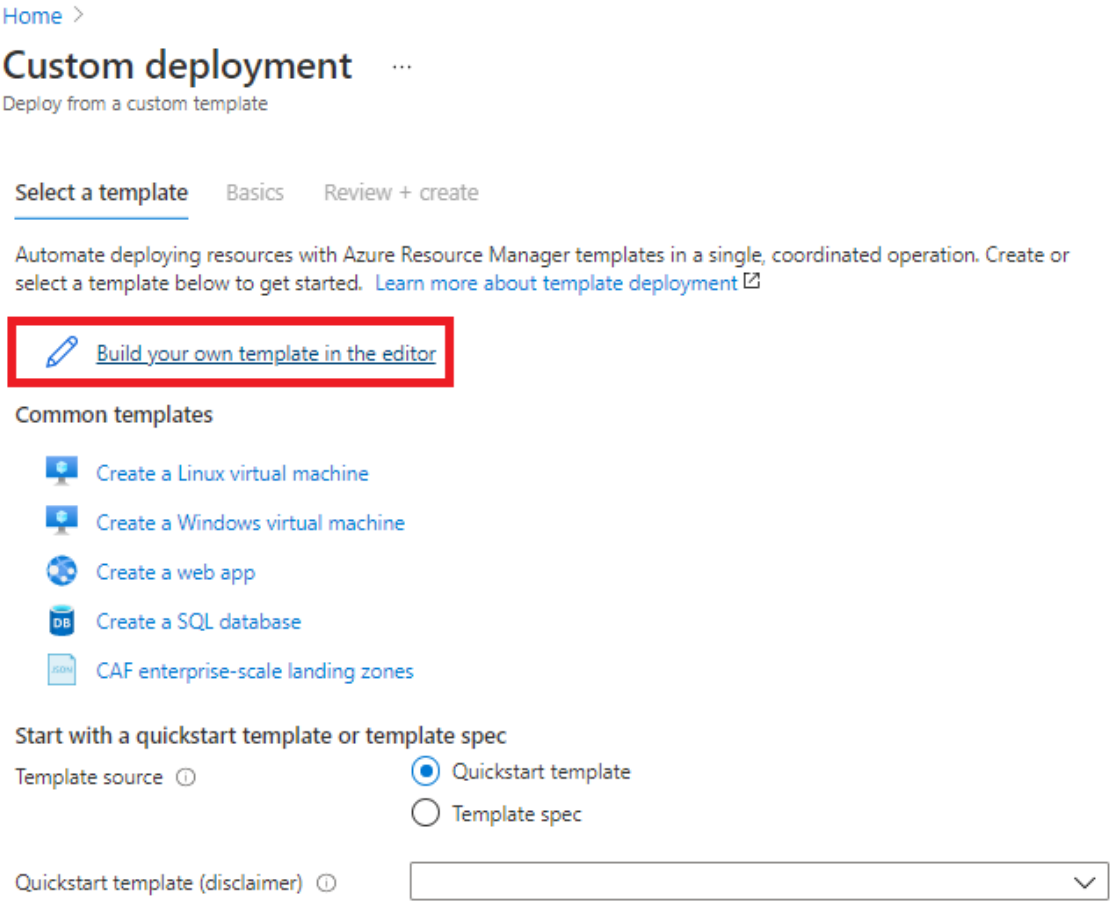
Install the SQL connector for logic apps

Before you start you need the location where you want the connector installed (for example westeurope) and you need the GUID for your azure subscription (this can be found under the subscriptions in azure).

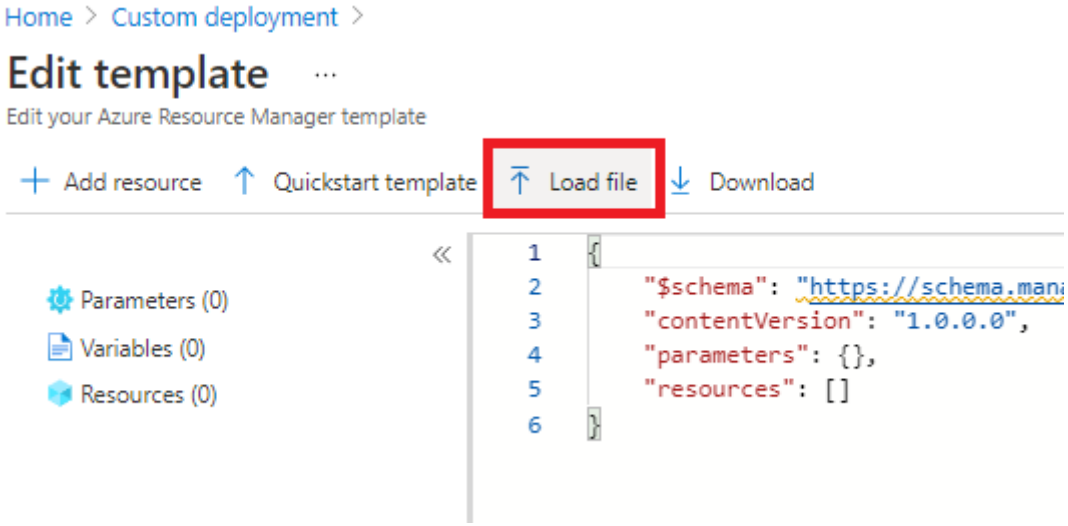
Create a resource group in your Azure tenant and choose to deploy a custom template. (type deploy in the azure search box).



Choose build your own template in the editor

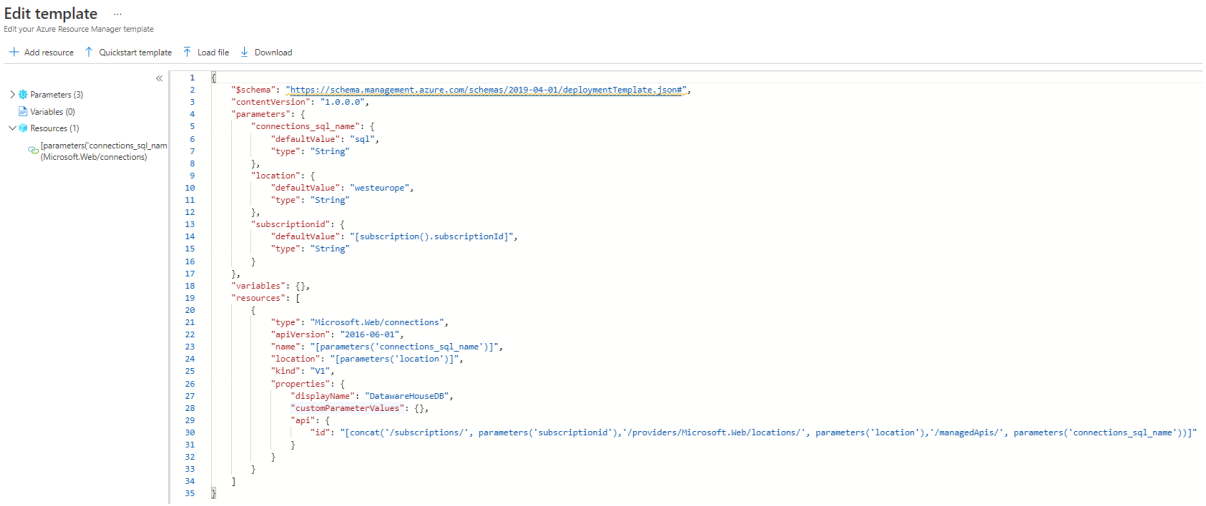


Choose load file and choose the logic app template and click save.

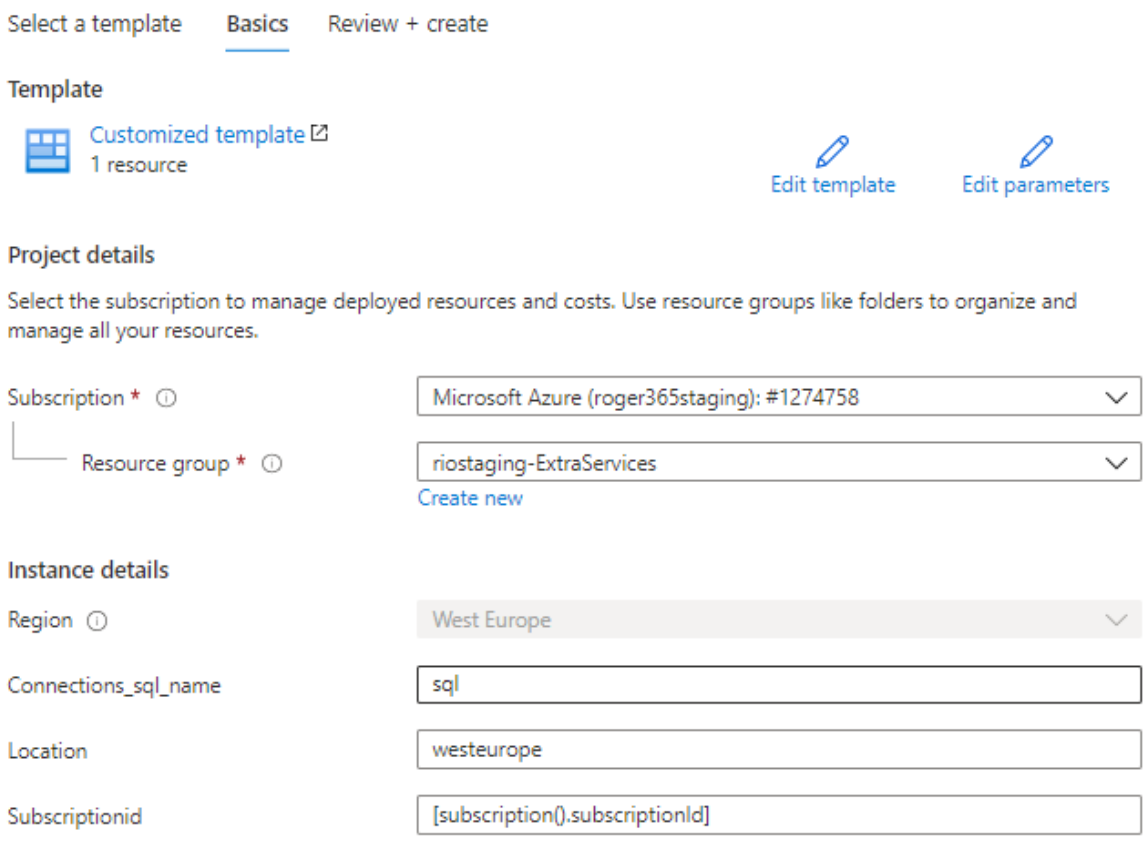


Choose the template.json from the API connection SQL folder (provided with the ZIP file)

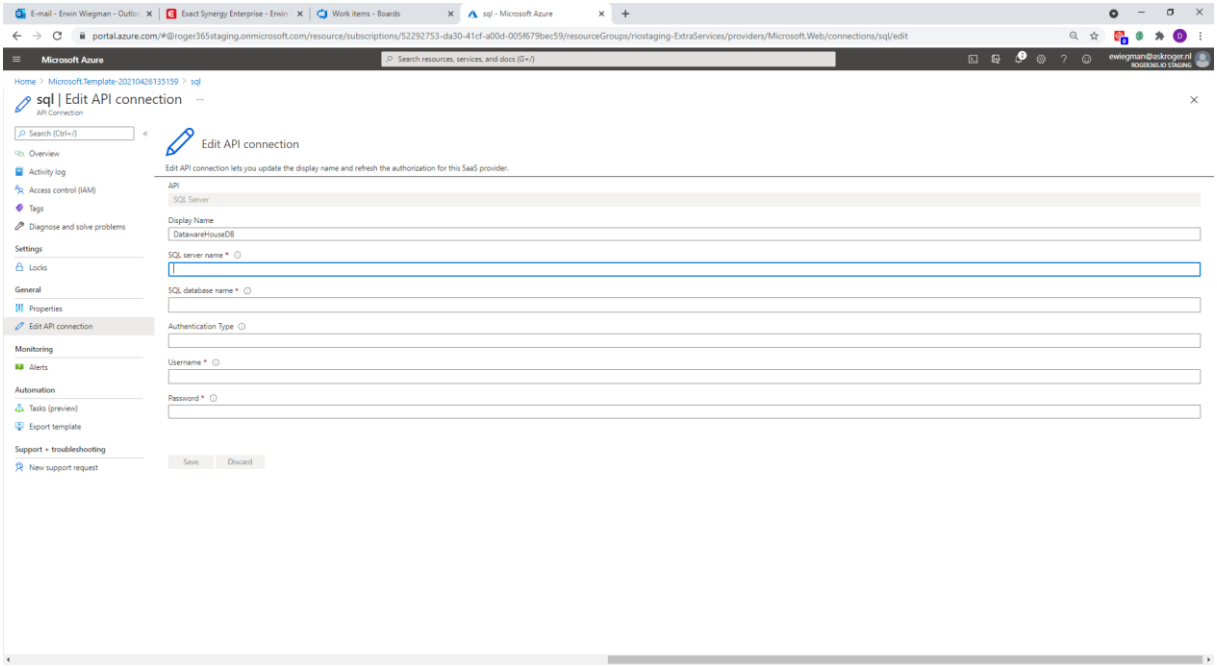
Save the template to get the deployment started



Only edit the Subscriptionid with your subscriptionid and press the review+create button (and afterwards the create button to create the resource).



After you deploy the connector, edit the connector and add the datawarehouseDB credentials (database you created earlier in which the dacpac file has been deployed) to the API settings.

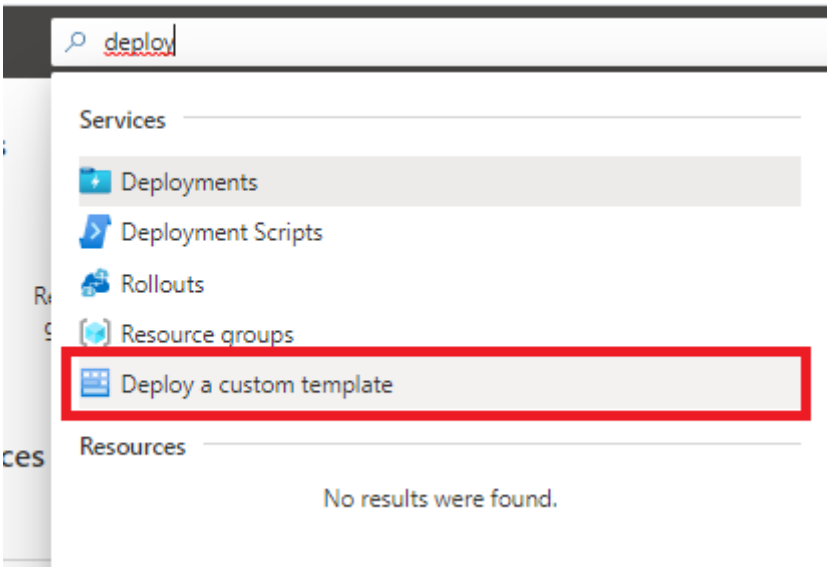


Item	Description
SQL servername	Your Azure SQL FQDN name in following syntax <<sqlservername>>.database.windows.net
SQL databasename	Your database name on the SQL server in which the schema is loaded
Authentication Type	is left empty
Username	Username to access the database (minimum of read/write permissions + execute stored procedure
Password	Password for user

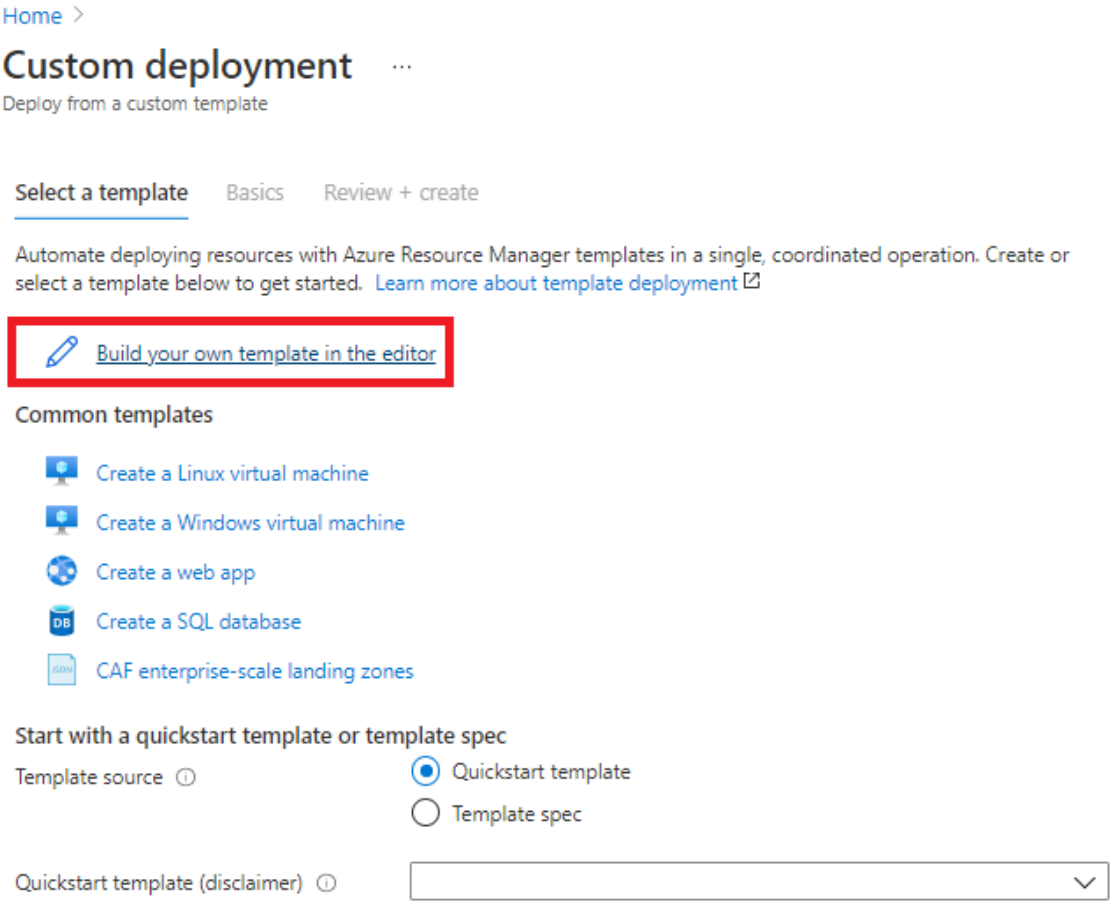
Import logic app to your azure tenant

First get the Resource Id from the SQL API created earlier. This must be used as an input for the logic apps

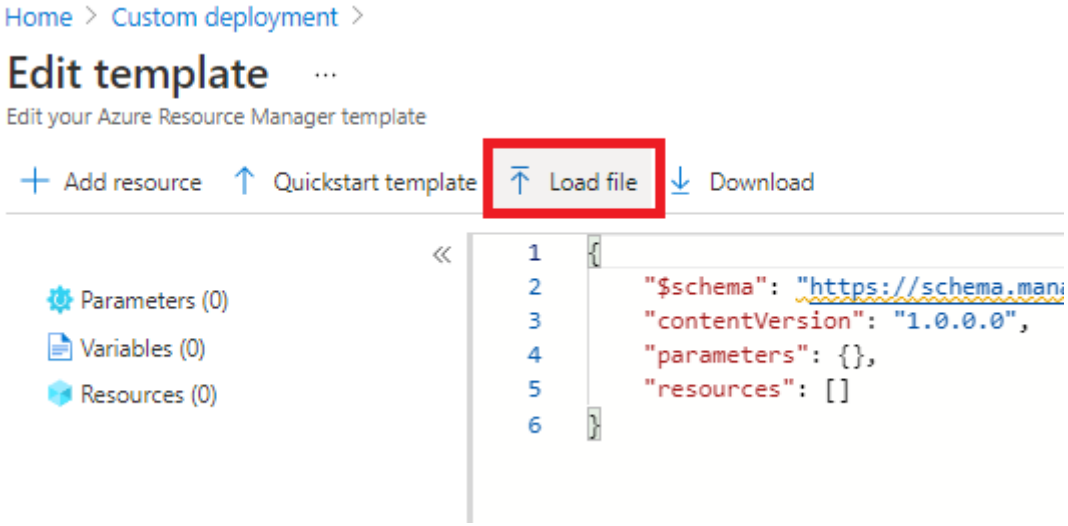
You can use the resource group created earlier. Choose to deploy a custom template. (type deploy in the azure search box).



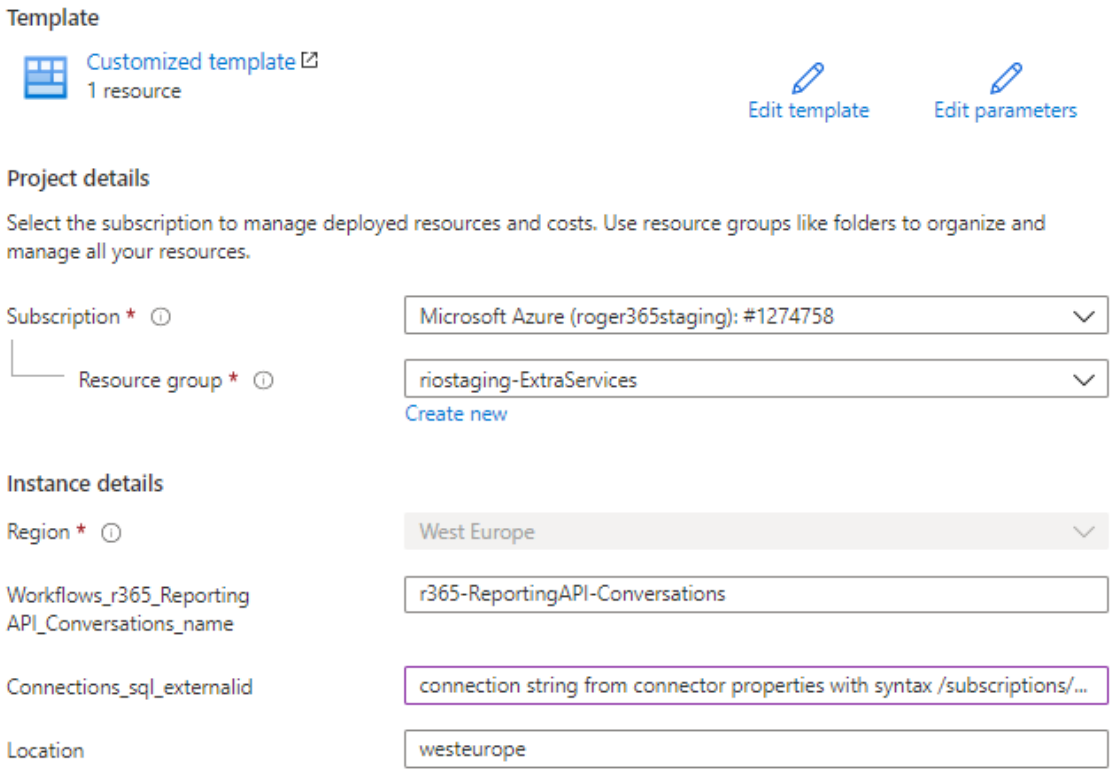
Choose build your own template in the editor



Choose load file and choose the logic app template and click save.

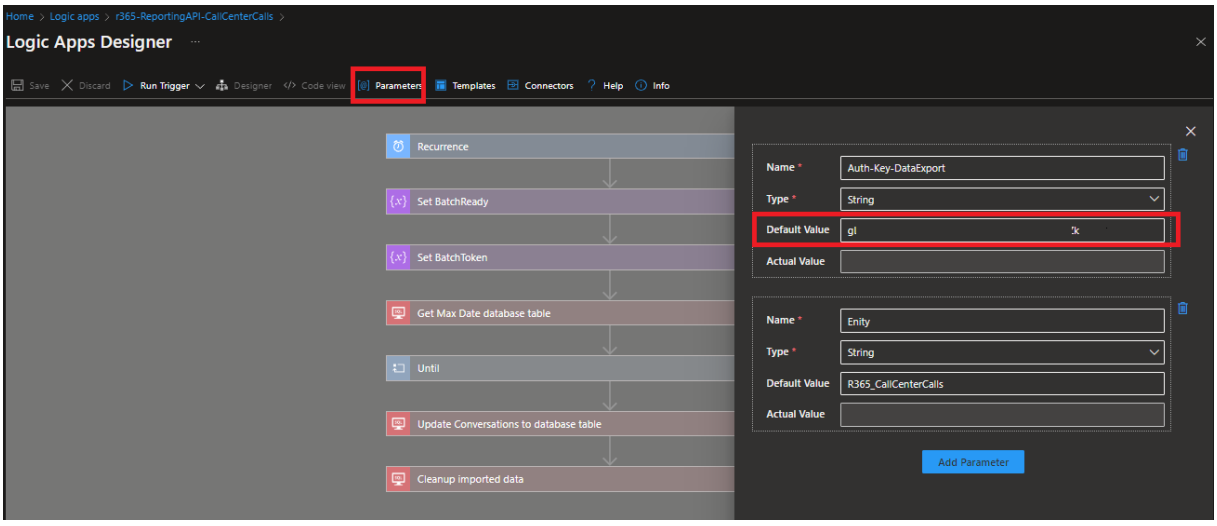


Use the connection id from the SQL connector (on the properties page of the resource saved earlier) and use this as the Connections_sql_externalid value.



Do this import for all logic app templates provided

After you have imported the logic apps. They should be edited to provide the right type of API key to get the data from the ROGER365.io platform. The value of the key can be stored in the parameters on the logic app.



The type of key can be found in the default Value description. This should be overwritten by the actual key. The key can be created in the ROGER365.io portal.

Installing a new version of the database

New features and bugfixes will be added to the solutions over time. So the database has to be updated. There are two scenarios for updating the database and it depends on your situation which one you have to use to update the database. Before updating the database always make sure you have a recent backup.

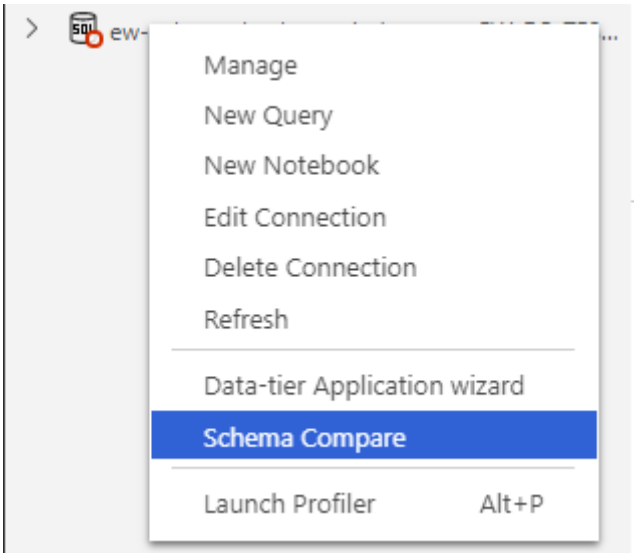
No custom object in the reporting database

If there are no custom object added, like views then the normal procedure can be used (same as how the initial database is deployed as previously descripted in this document.

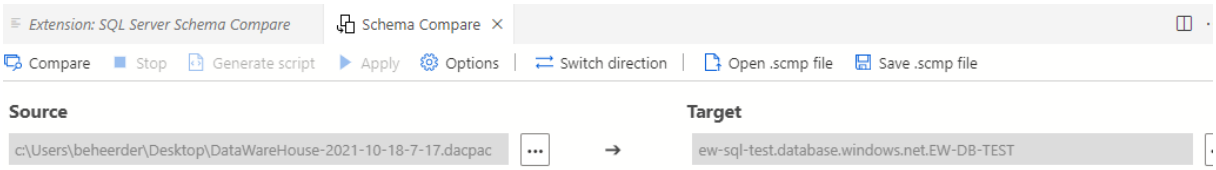
With custom object in the reporting database

When there a custom objects added to the reporting database, the normal update doesn't work, because this update would remove all the custom object created. In this case we have to do a schema compare in Azure Data Studio and unselect the deleting of the object.

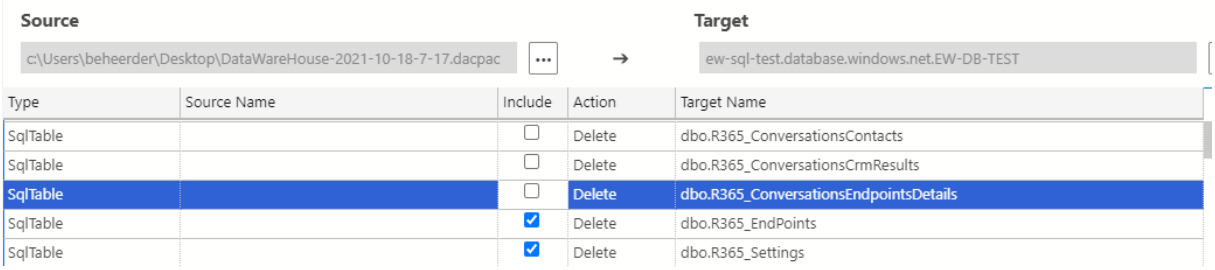
For this the extension SQL Server Schema Compare should be installed in Azure data studio. After installing the extension you can right click on the database and select the schema compare option.



Select the new dacpac file on the source and the reporting database as the target and click the compare button.



After some time the differences between the dacpac file and the running database is shown. For all the custom object in the database which should NOT be deleted, the check mark in the include column should be off.



When you deselected all the custom objects the changes can be applied to the reporting database with the apply button at the top.

Database explanation

At the time of writing the reporting database has the following tables:

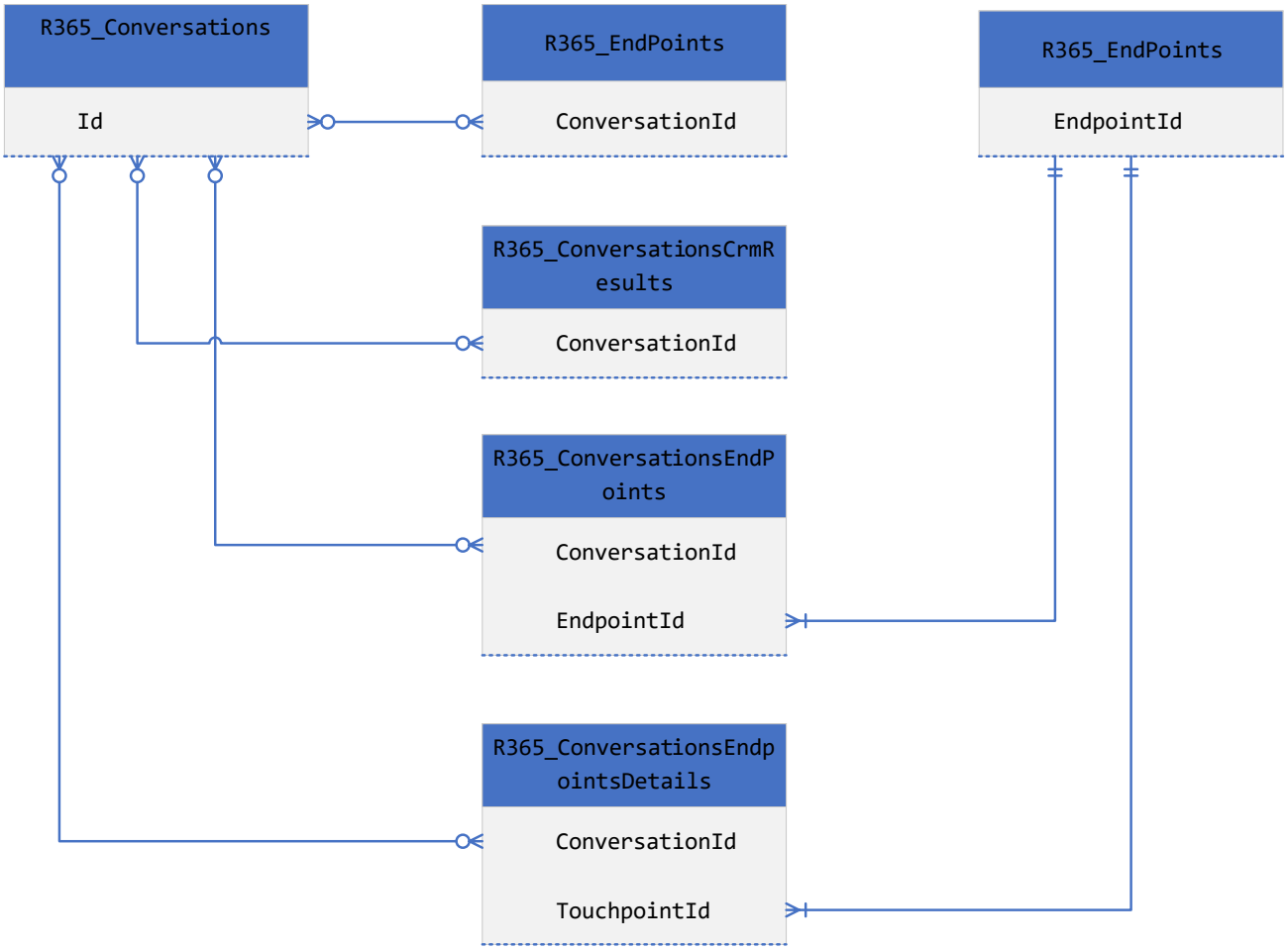
Table	Description
R365_CallCenterAgents	Contact center Agent information is stored in this table. This also includes the AzureAd ObjectId
R365_CallCenterAgentsEventLog	Log of events from the contact center agents
R365_CallCenterAgentsWorkEntries	Information stored about the after worktime of the contact center agents.
R365_CallCenterCalls	This is the primary table of the reporting solution in which the information is stored about the call center calls
R365_CallCenterCallStreamChanges	Information about audio stream changes received from the teams platform. Can be used to determine how long a call has been put on hold.
R365_CallCenterExecutionResults	The execution results from a contact center call
R365_CallCenterQueues	The contact center queues which are configured
R365_CallCenterRequests	Logs the Hunt requests. Every time a call enters a queue, an entry in this table is created and used for determining the agents to hunt.
R365_Conversations	This is the primary table of the reporting solution in which the information is stored about the closed conversations
R365_ConversationsContacts	Contacts which we active part on the conversation. If know by the CRM replicators, the CRM name is also shown. The items shown here, are the items which are picked in the conversation itself (If there are multiple CRM results).
R365_ConversationsCrmResultsBulk	All CRM results that apply to the conversation. These also include the results which are not selected in the conversation itself.
R365_ConversationsEndPoints	Link table for all endpoints which are used in the conversation
R365_ConversationsEndpointsDetails	Details of the conversation endpoint during the conversation. (for example Webchat session Id or Twitter username)
R365_EndPoints	Endpoints names which are configured in the ROGER365.io portal (including deleted endpoints).
R365_Transcripts	Table with the actual send messages in the conversation. For this to be filled, the option for transcript must be enabled on the touchpoint in the ROGER365.io portal
Sys_Staging_JSON	Table used for supporting the ELT process from the logic apps. Should not be changed or used for reporting
R365_Settings	Table for storing settings for the reporting solution

And these views:

View	Description
vw_CallCenterCalls	Best view for reporting on the call center calls.
vw_Conversations	Best view for reporting on the conversations, this view aggregates the tables to one reporting entity
vw_Hours	View that can be used to join data for reporting based on every hour.

Please keep in mind when creating PowerBI reports, the best wat is to create these reports on the view because of maintenance.

Social messsaging schema



Contact center schema