# 

Fuel Watch RSS

| As-Built Documentation

Authors: Sandeep Jami

Date: 01st September 2022

Version: V1

**Table of Contents**

[1 OVERVIEW 3](#_Toc112957839)

[2 SOLUTION ARCHITECTURE 4](#_Toc112957840)

[2.1 Components 4](#_Toc112957841)

[3 Solution Delivery 6](#_Toc112957842)

[4 RESOURCES DEPLOYMENT 7](#_Toc112957843)

[4.1 Azure Resource Group 7](#_Toc112957844)

[4.2 Azure Data Lake 11](#_Toc112957845)

[4.3 Azure SQL Server Database 13](#_Toc112957846)

[4.4 Azure Data Factory 15](#_Toc112957847)

[4.5 Azure Key Vault 17](#_Toc112957848)

[4.6 Azure Data Factory Pipeline Deployment 19](#_Toc112957849)

# OVERVIEW

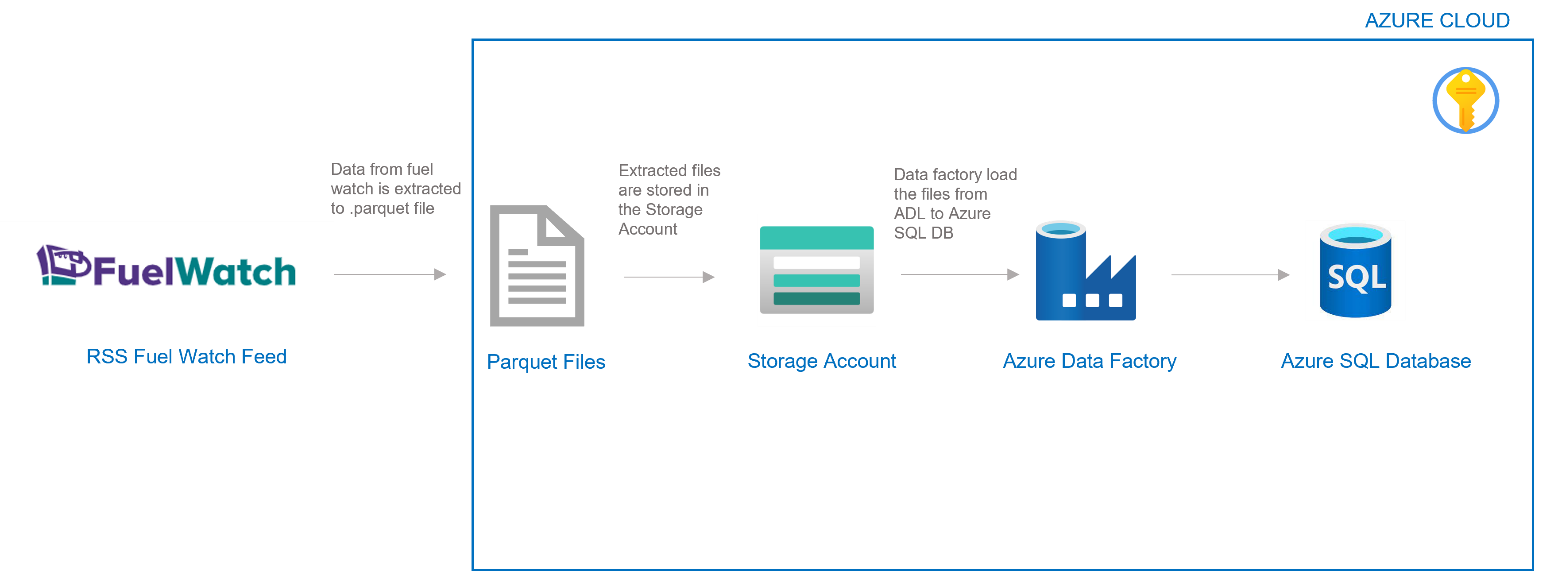
Form the given scenario, data from Fuel Watch is loaded in to Azure SQL database using different Azure components which is shown in the document below.

As part of the requirement, all resources are deployed to Azure via Azure Resource Manager (ARM) template and the templates are provided in the .zip format.

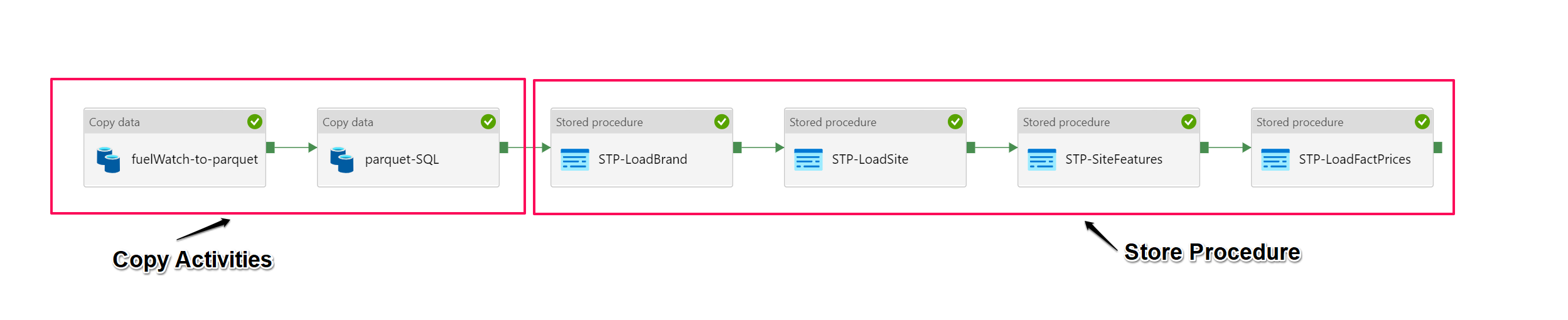
This document outlines the architecture about the implementation of Rss Fuel Watch feed to Azure SQL database such as creating, deploying and running the scripts.

# SOLUTION ARCHITECTURE

The following diagram depicts the different Azure resources and components involved in the set-up and the deployment.



The data is extracted from the Fuel Watch source system in the format of Parquet file, and stored in Azure Storage Account. Azure Data Factory will extract the files from the Storage Account and ingest the data in to Azure SQL database.



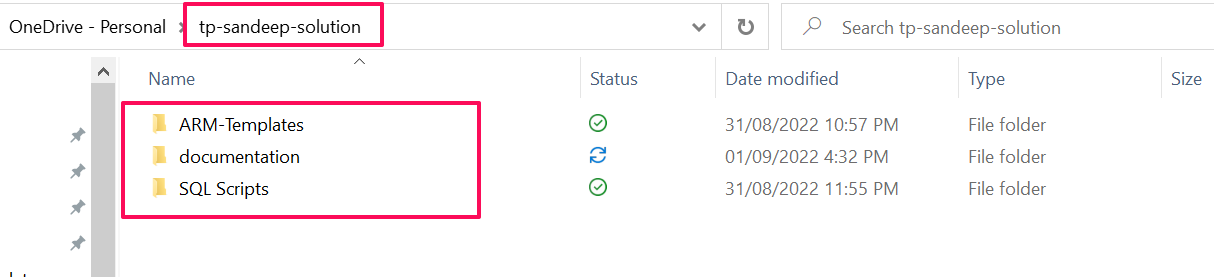
## Components

Below is the list of the components which are used in developing the solution.

| Azure Service | Description | Configuration |
| --- | --- | --- |
| Azure Resource Group | Containers for all Azure resources are listed below |  |
| Azure Data Factory | Azure's cloud ETL service for scale-out serverless data integration and data transformation | V2 |
| Azure Storage Account | A storage account is automatically created when setting up an ML studio | LRS - StorageV2 |
| Azure SQL Database | Azure SQL Database is a fully managed platform as a service (PaaS) database | General Purpose - Serverless: Standard-series (Gen5), 1 vCore |
| Azure Key Vault | Key Vault is used to store the keys and secrets. | Standard |

# Solution Delivery

The solution is delivered in .zip format (tp-sandeep-solution), where all the files and scripts are in one folder.

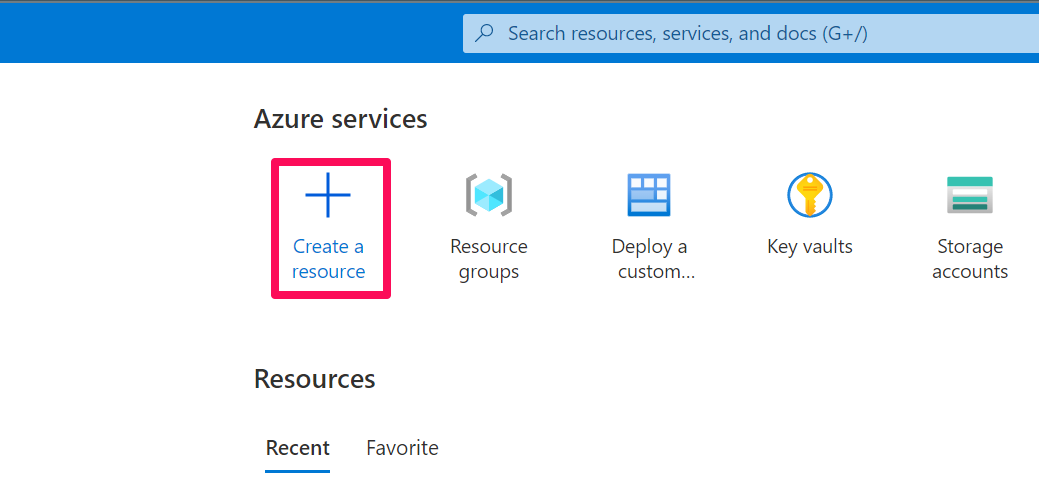


# RESOURCES DEPLOYMENT

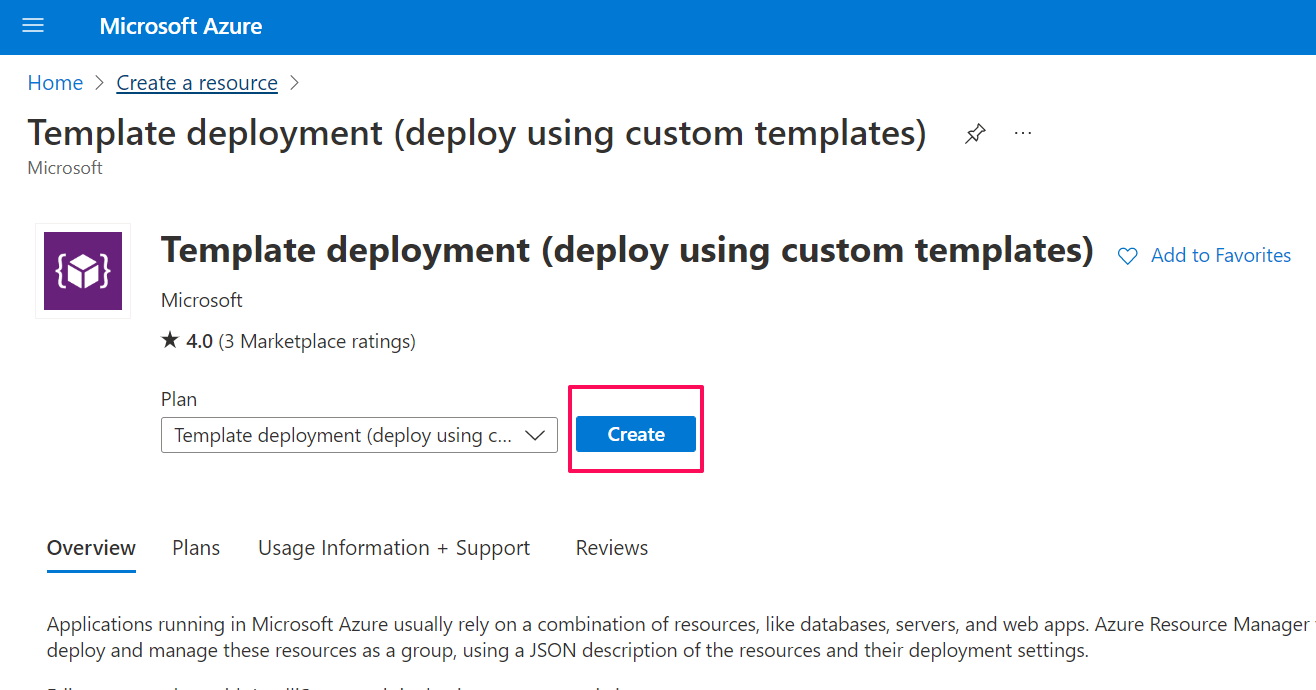
The ARM templates have been provided to deploy the Azure resources in the given folder.

## Azure Resource Group

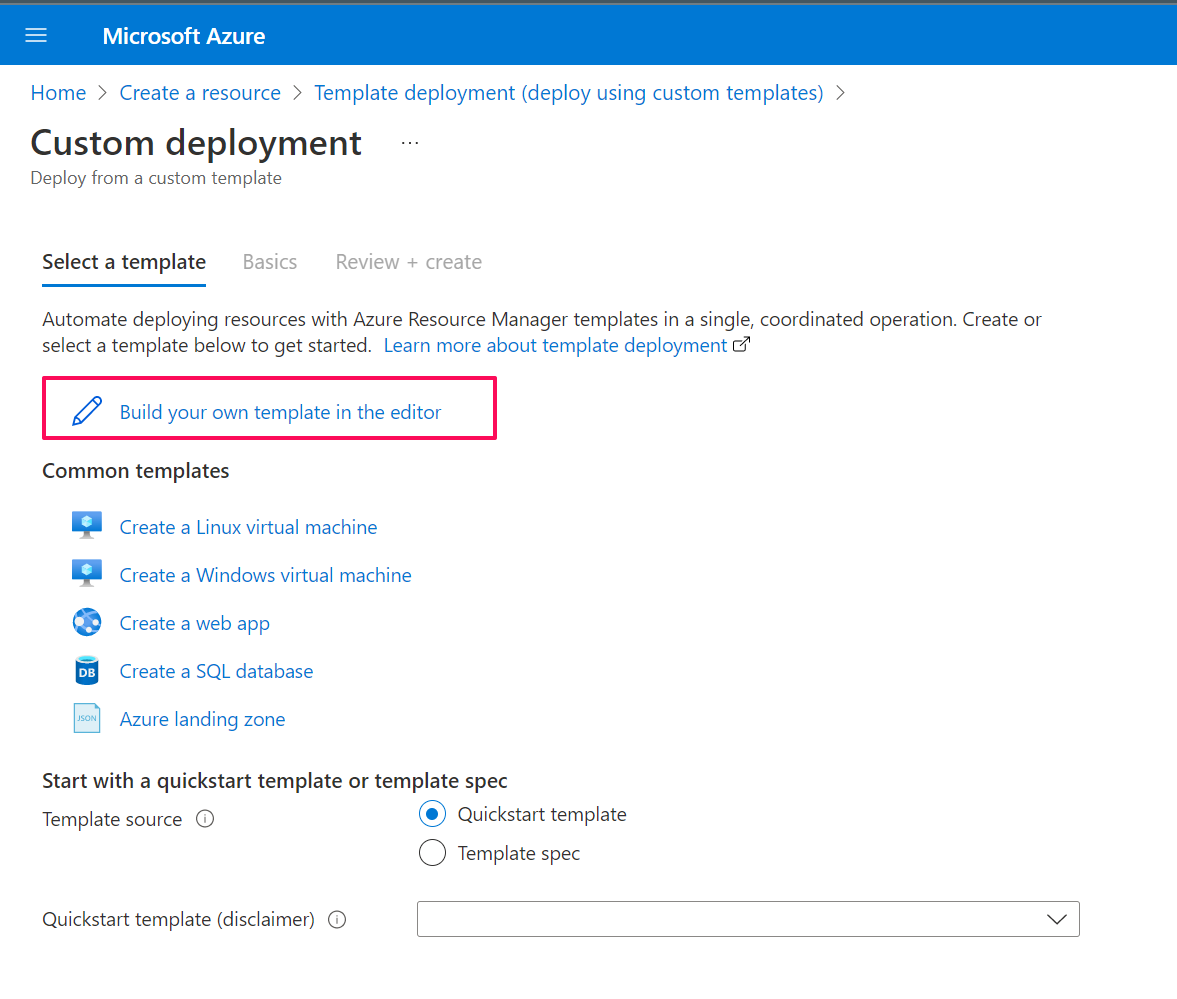
Once the user opens the Azure services, click on “create a source”.



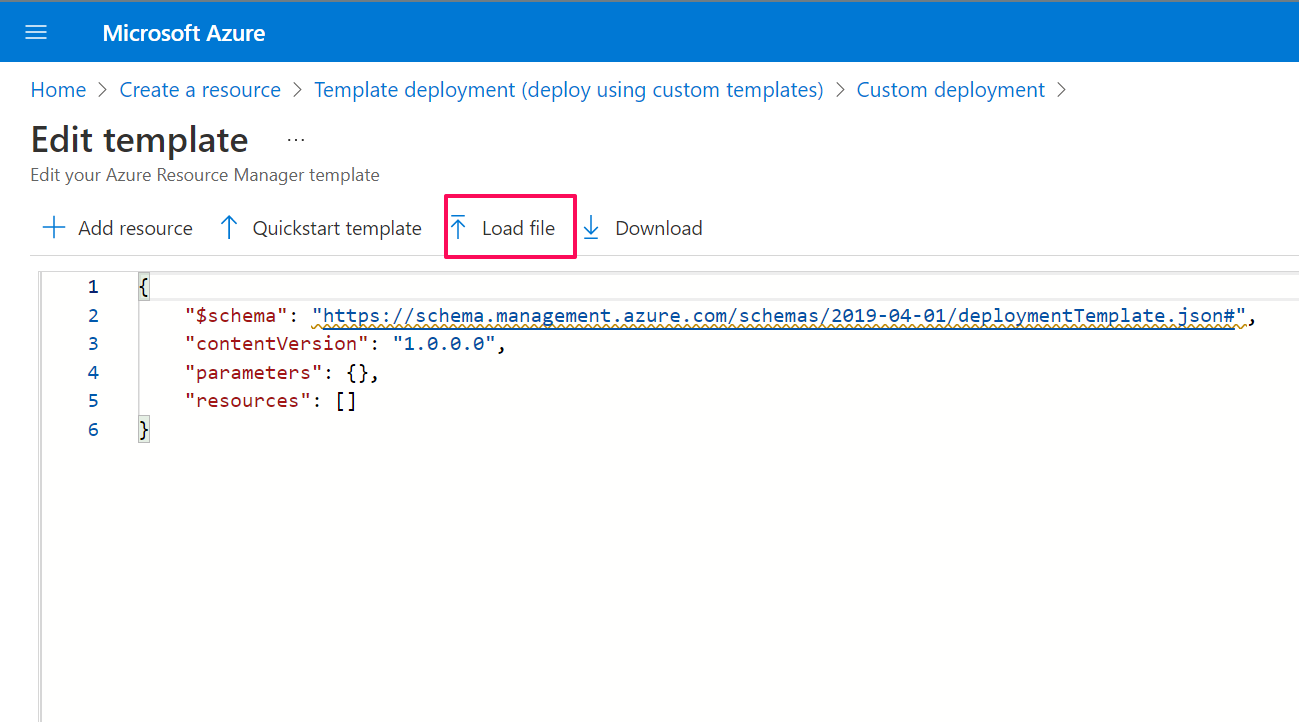
The user can then see the template deployment page, then click on “create”.



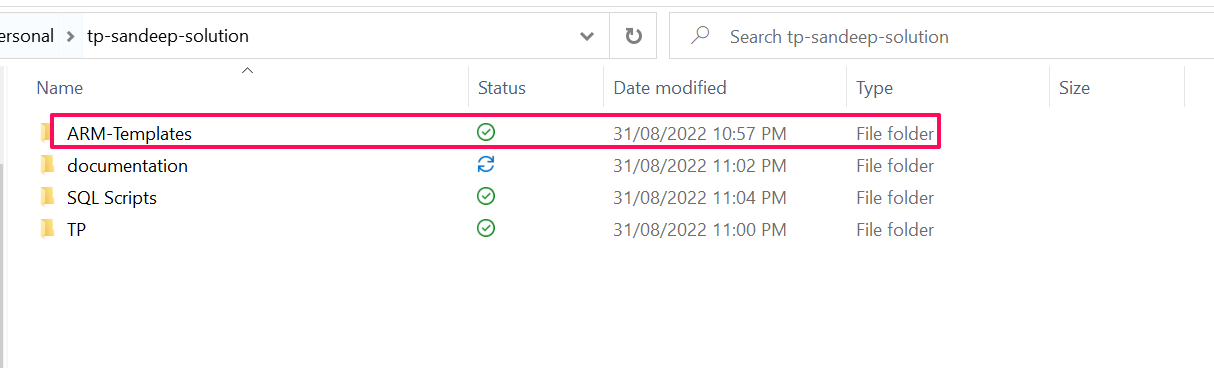
Once the user clicks on create, the custom deployment page is opened in which you can build your own template in the editor.

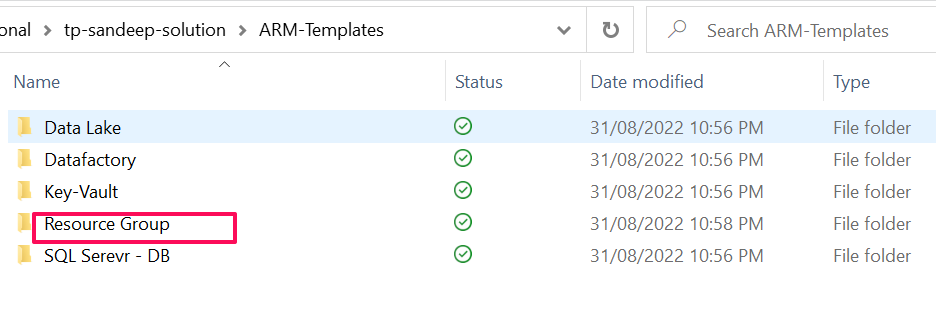


The user then loads the file.

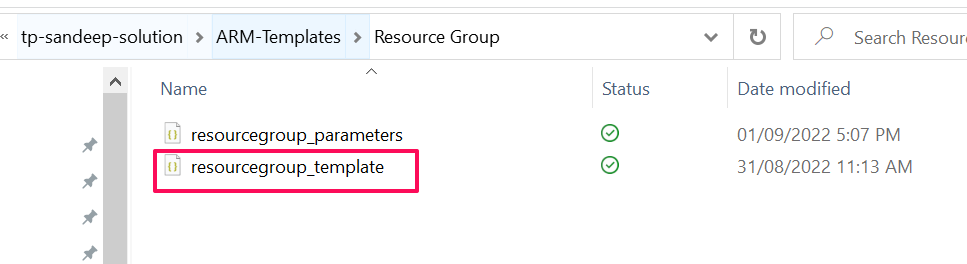


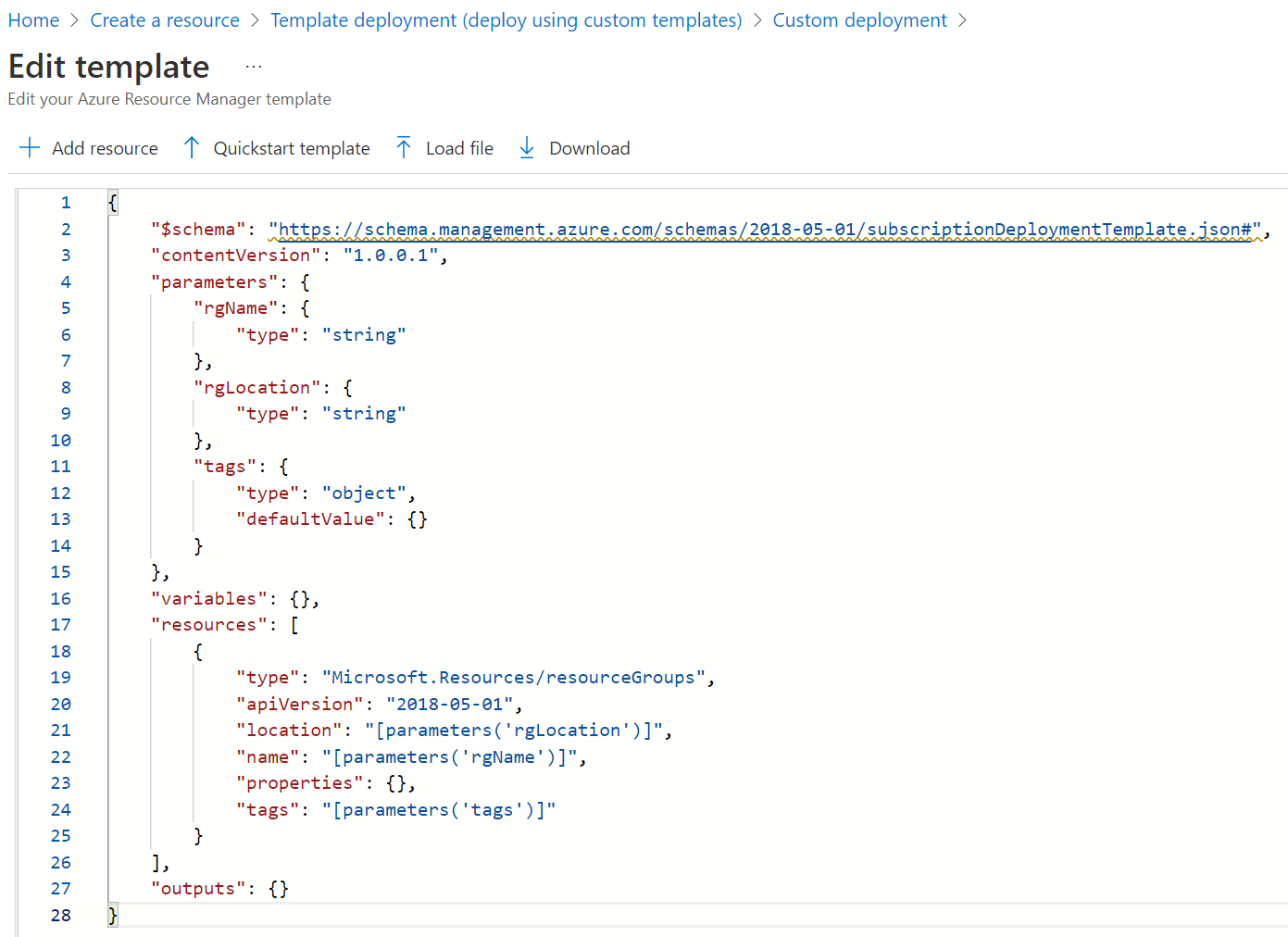
Navigate to ARM-templates and then select Resource Group.





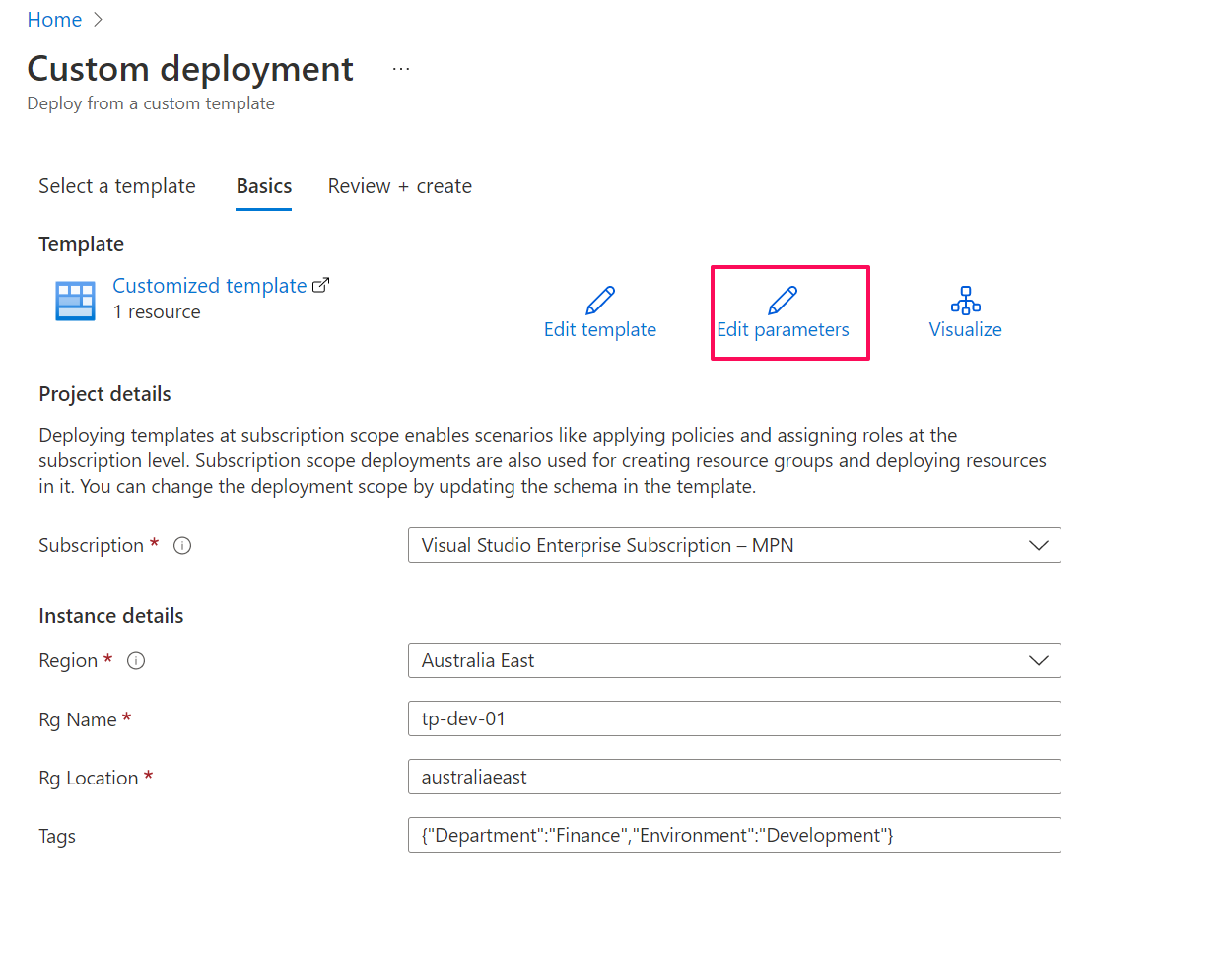
Upload the template file followed by parameter file.



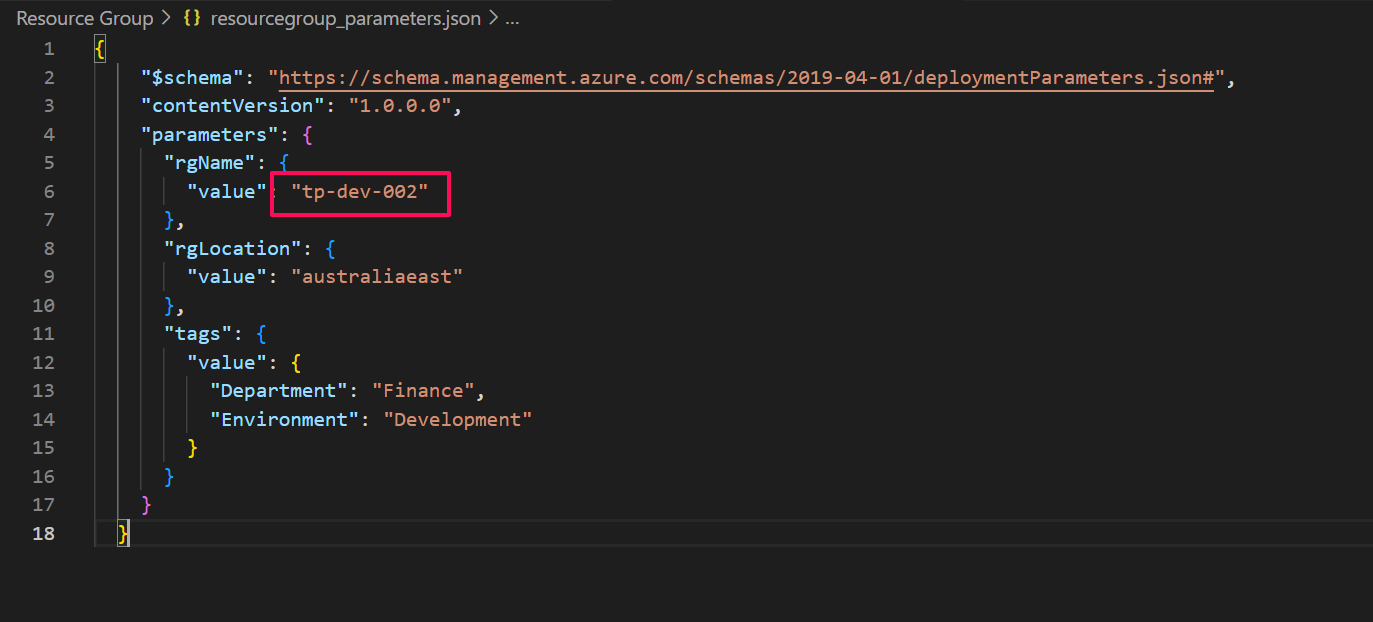


Load Parameters

You can edit the parameters at this point.

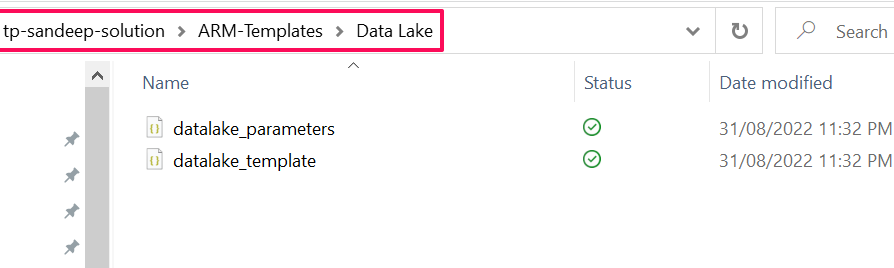


Any naming changes can be done in the parameter file.

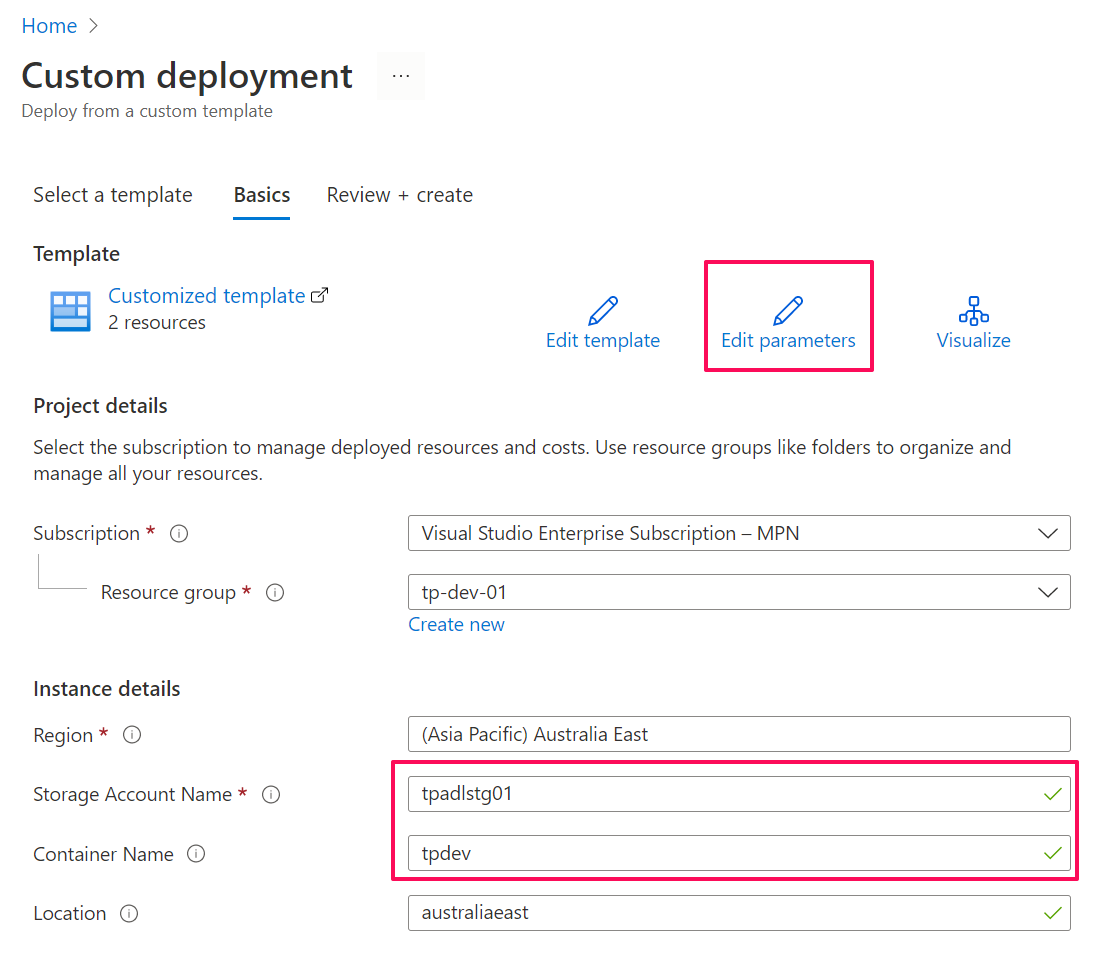


## Azure Data Lake

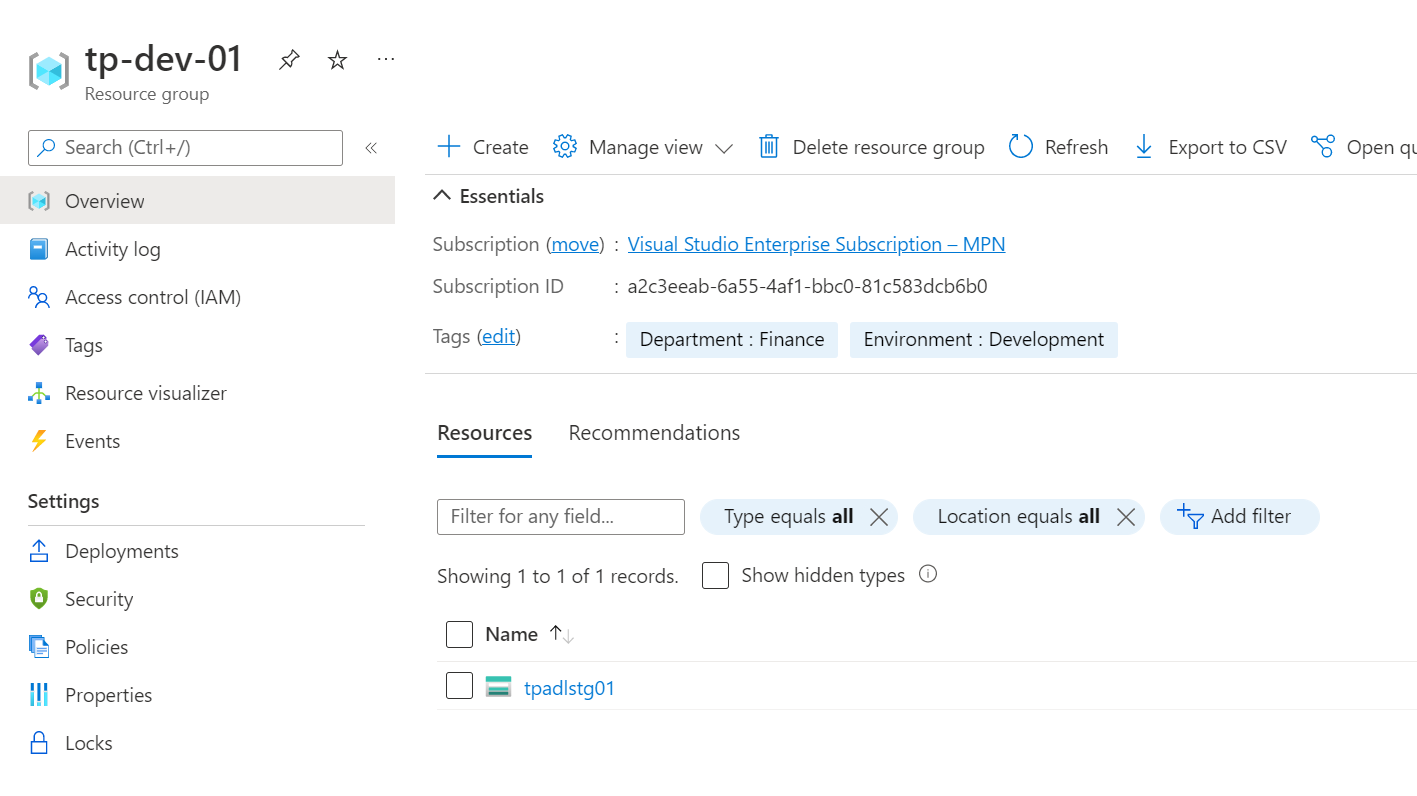
Navigate to ARM-template in the tp-sandeep-solution and select Data Lake. Upload the template file followed by the parameter file.



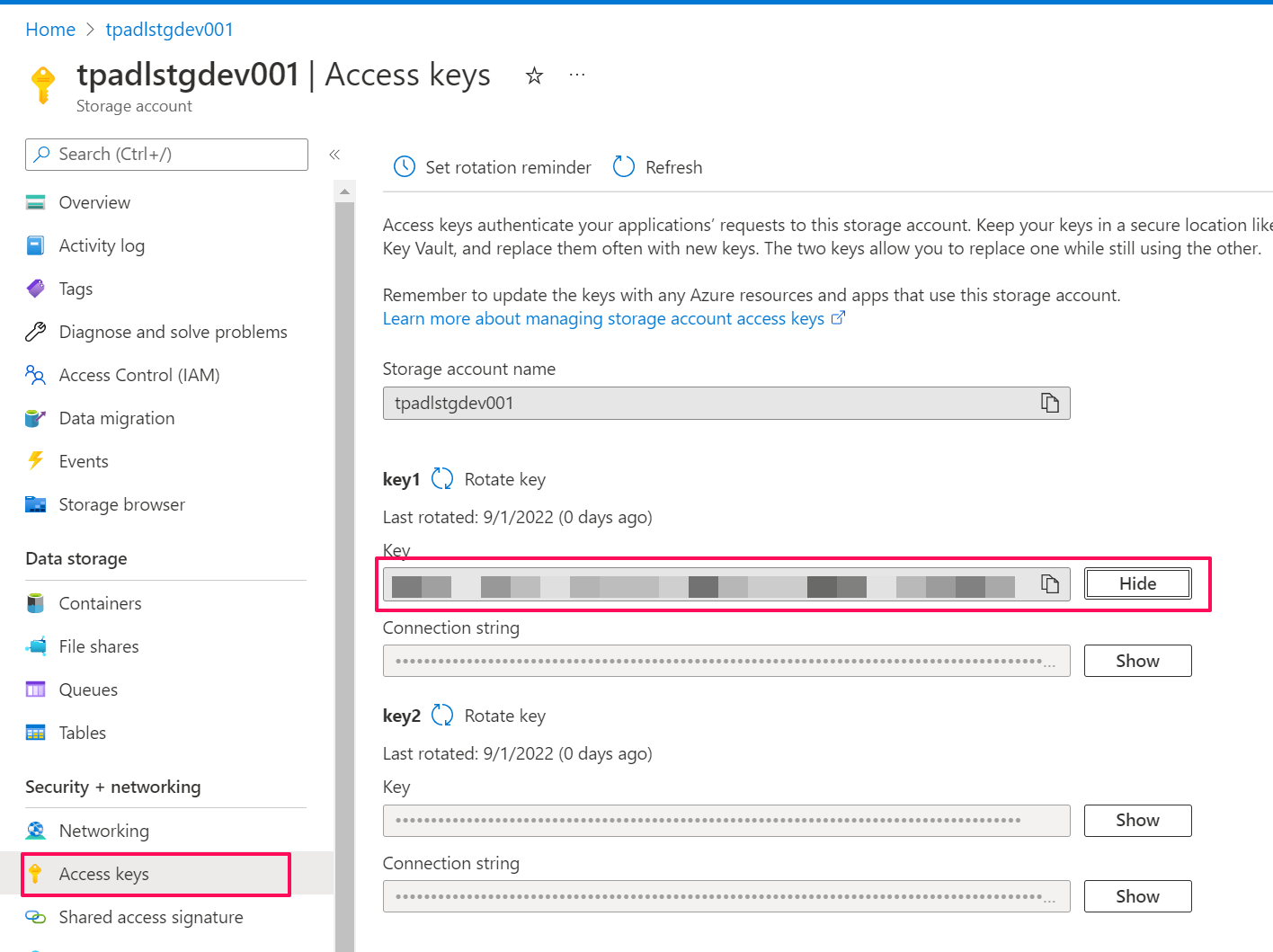
The user can edit the parameters at this point.





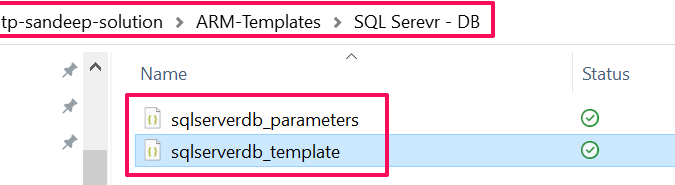


Once the data lake is created, copy the access key as this key has to be used while deploying the key vault.

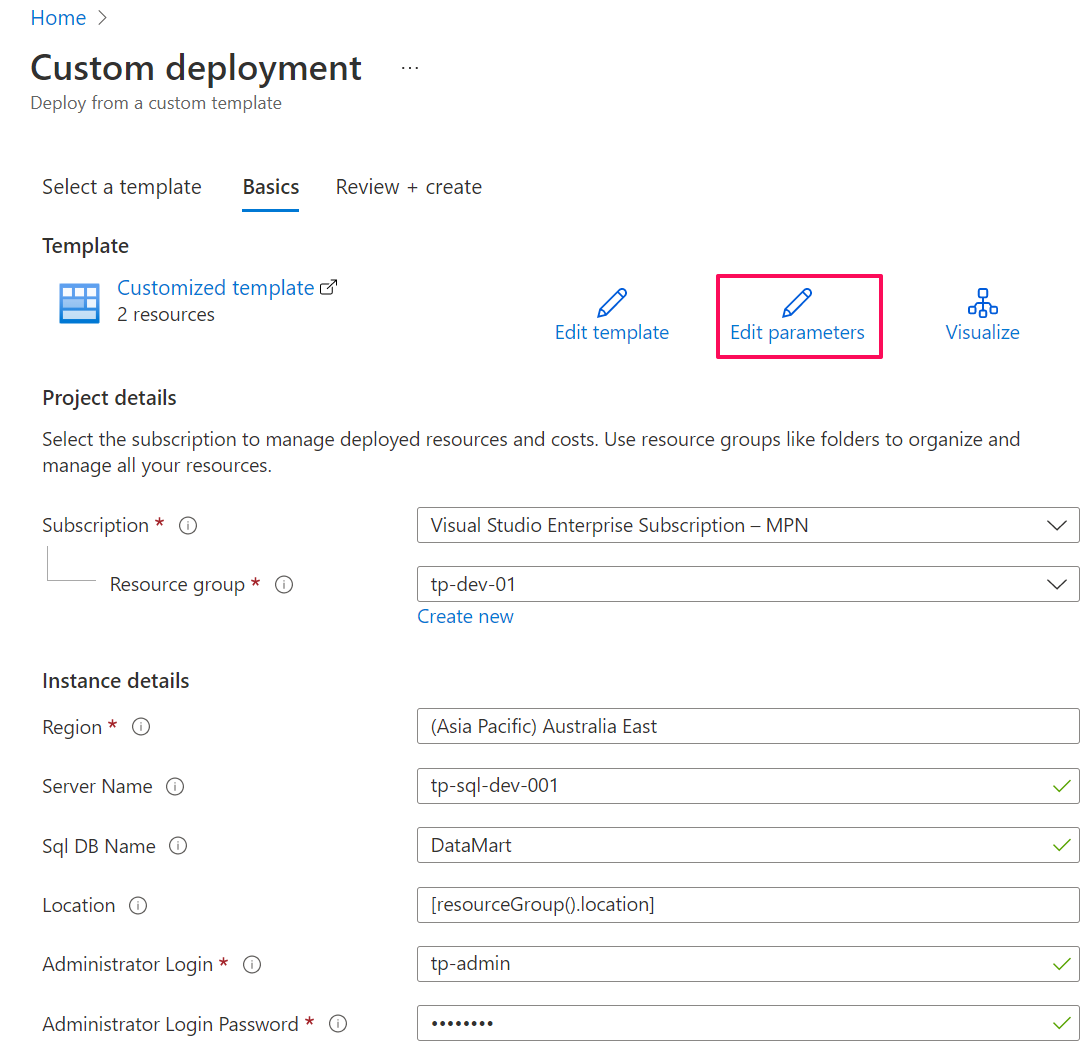


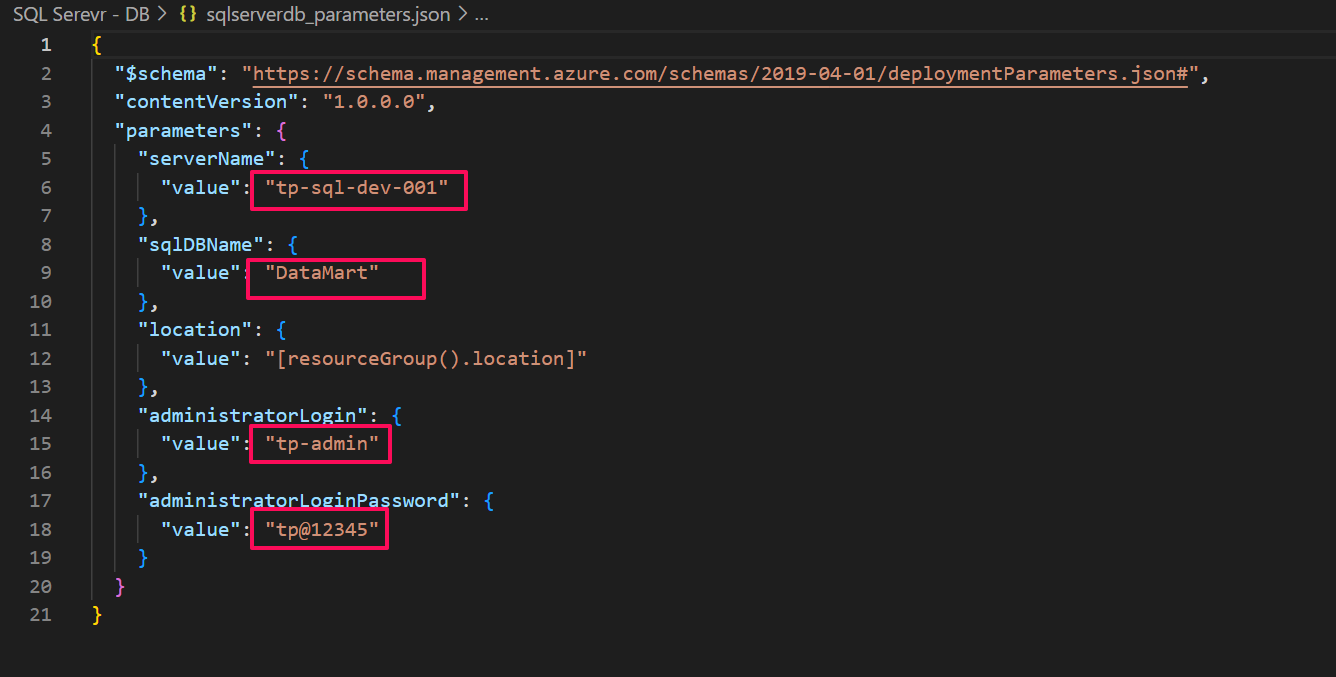
## Azure SQL Server Database

Navigate to ARM-template in the tp-sandeep-solution and select SQL Server - DB. Upload the template file followed by the parameter file.

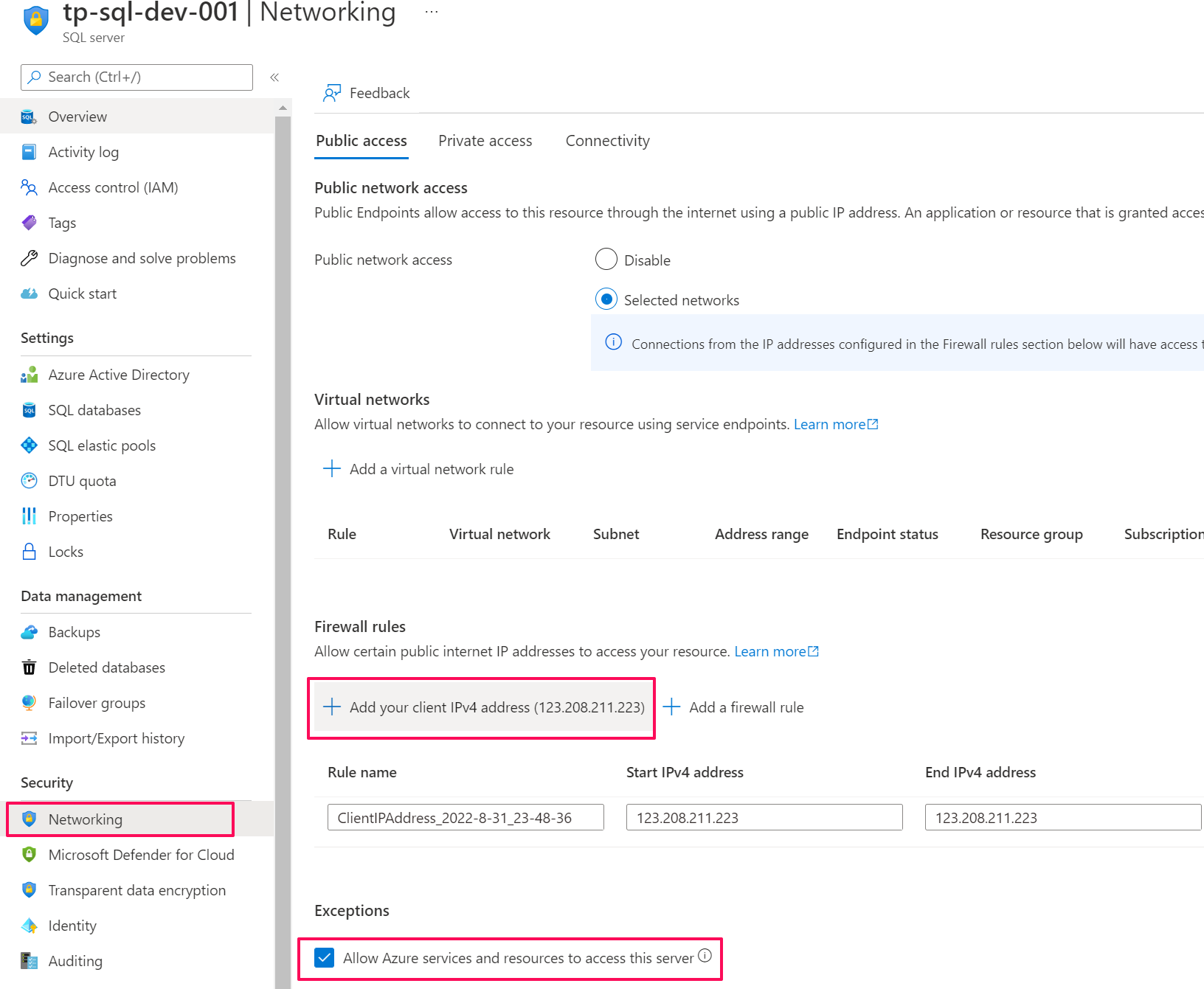


The user can edit the parameters at this point.

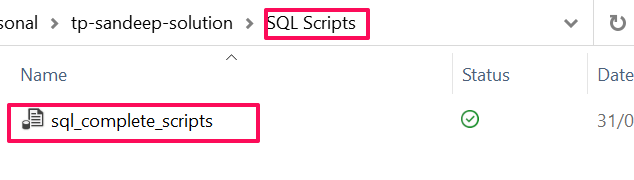


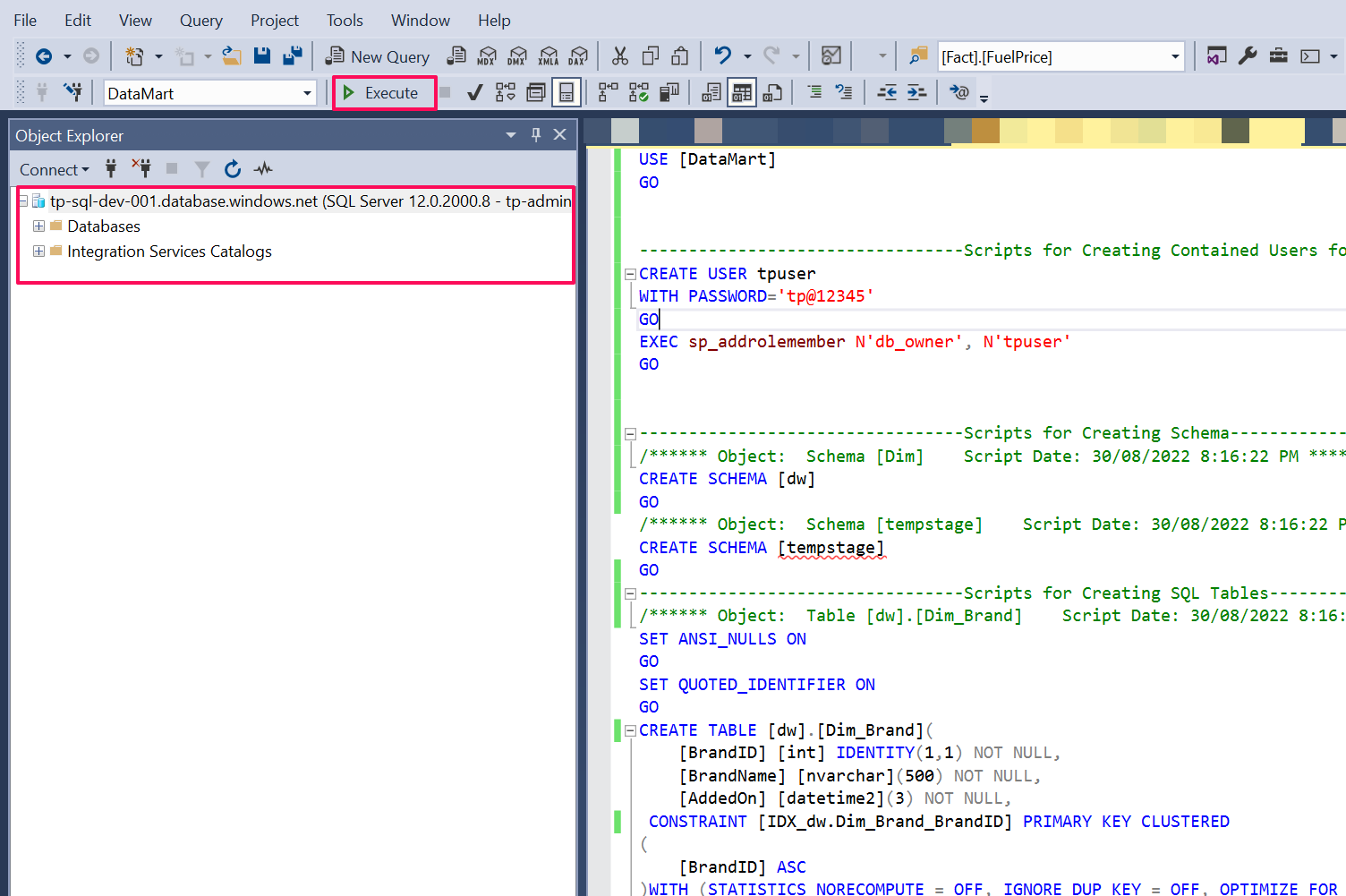


Once the server and database are created, navigate to networking in the server and allow azure services and resources to access this server. At this point, you have to add your IP as shown in the below image.

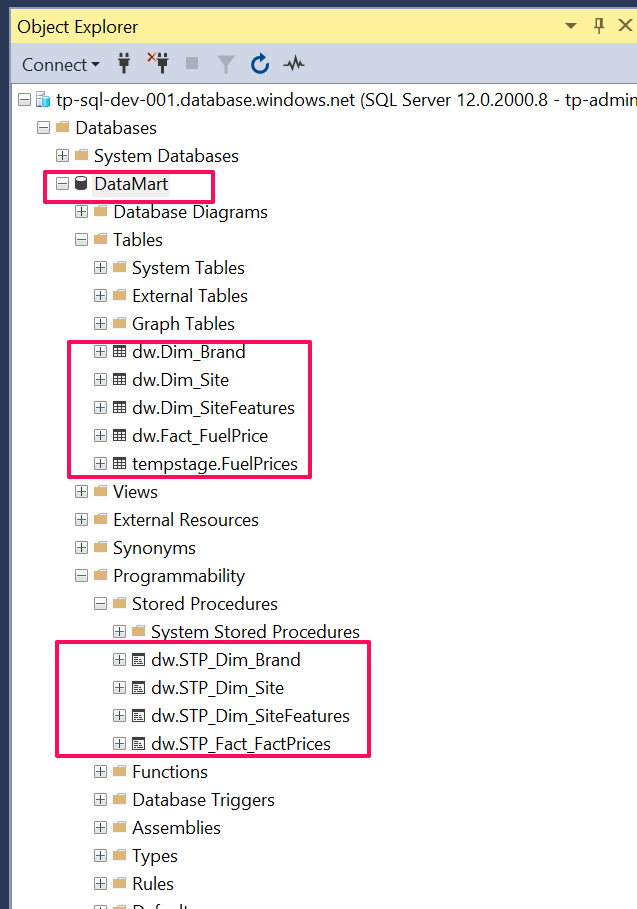


Now login to this server using SQL Server Management Studio (SSMS) with tp-admin user and run the sql\_complete\_scripts files as shown below.





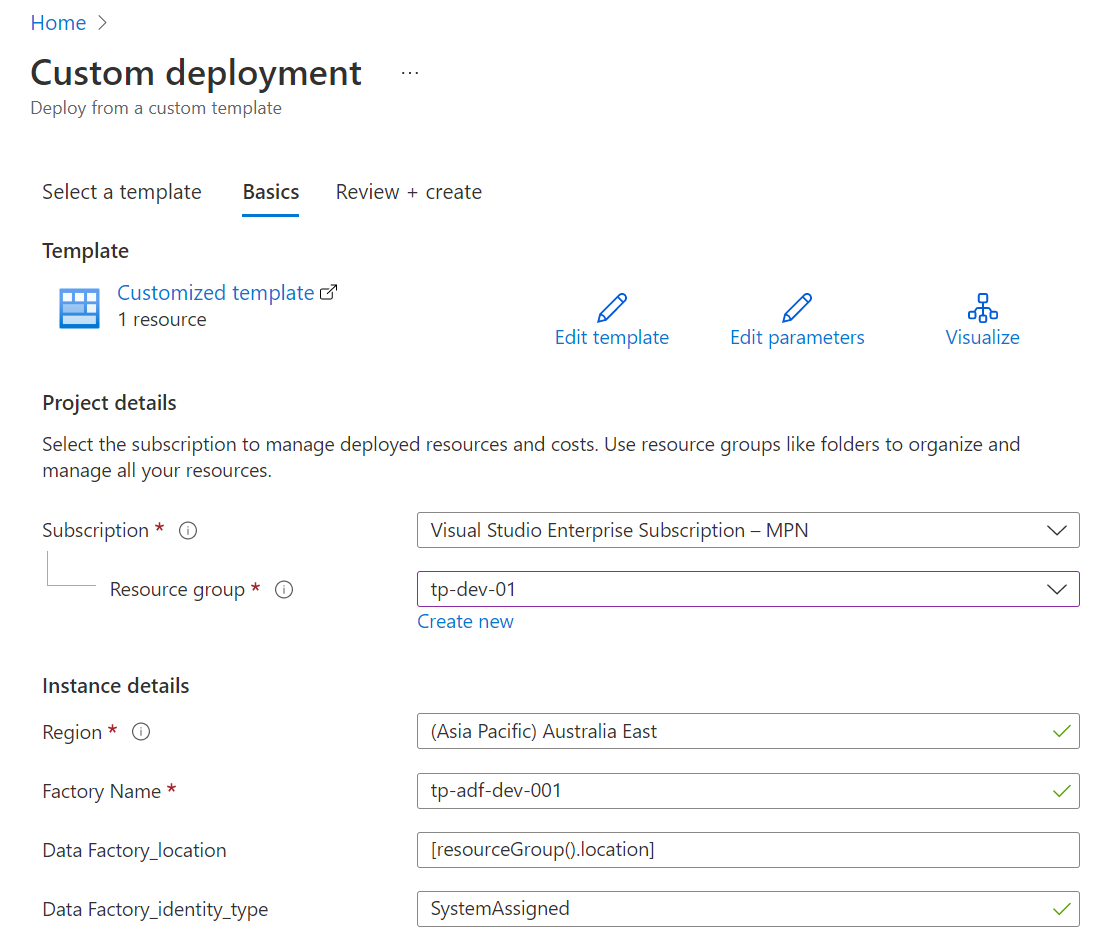
Once the scripts have been executed you should be able to see the tables and store procedures.

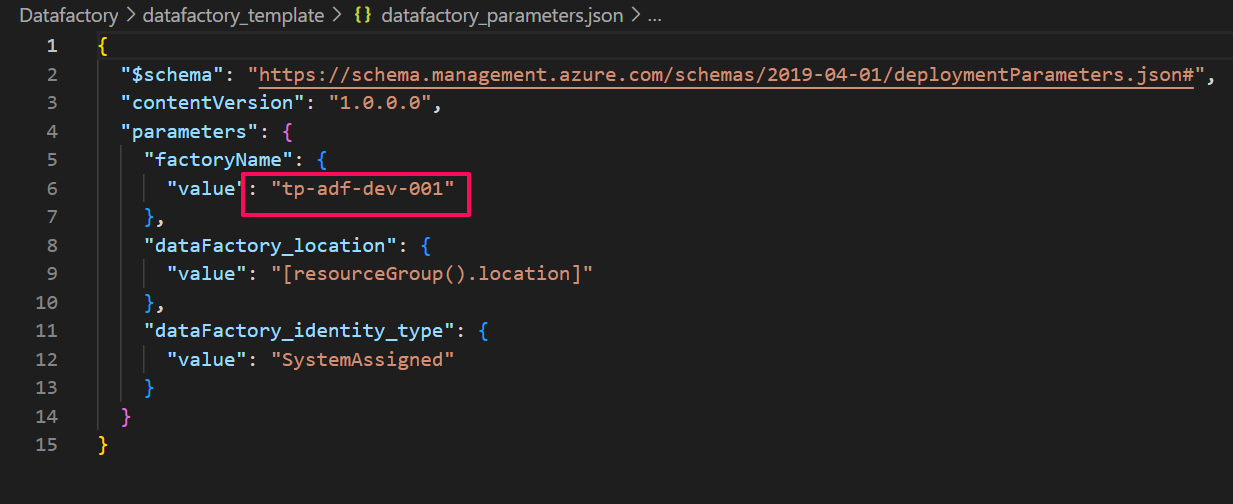


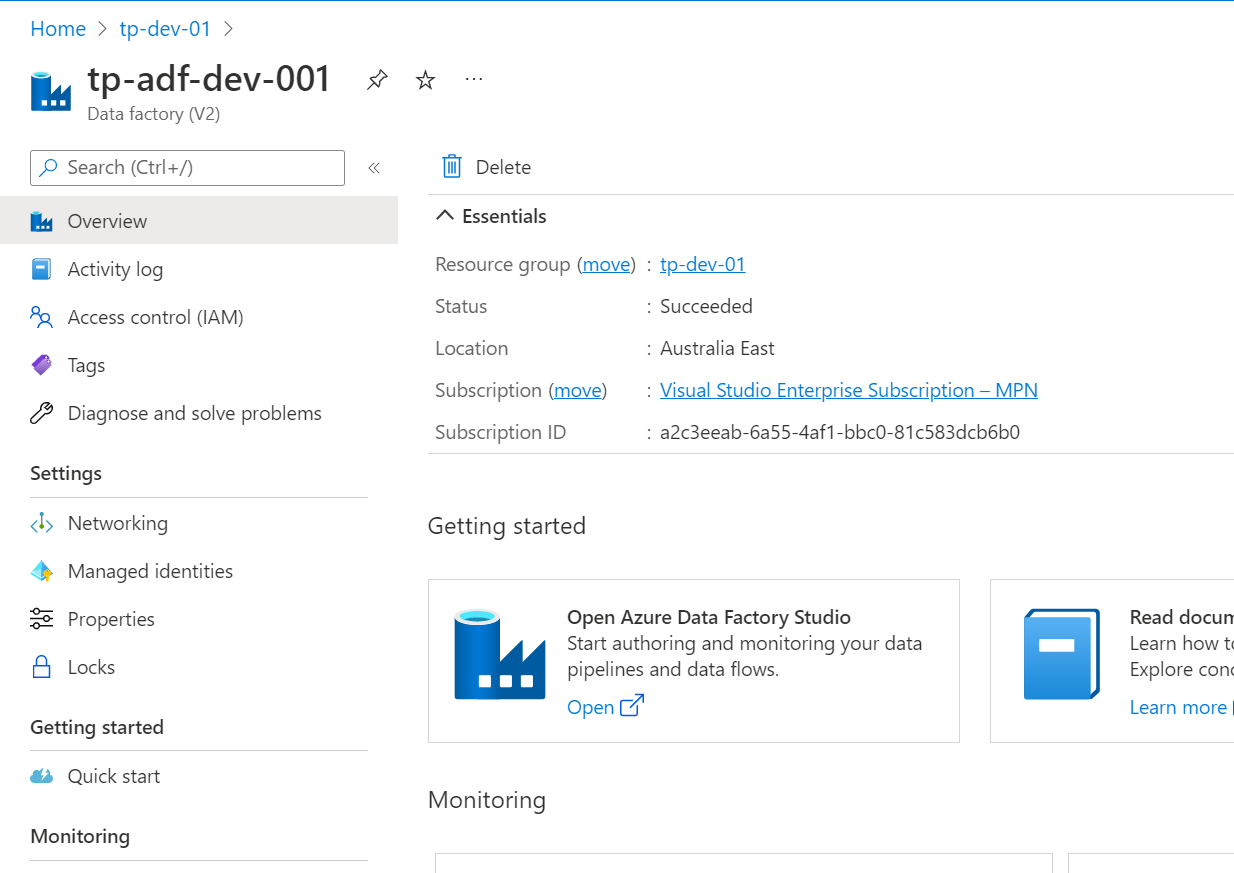
## Azure Data Factory

In this section, we are deploying the data factory and the pipelines are deployed after the Azure key vault creation.

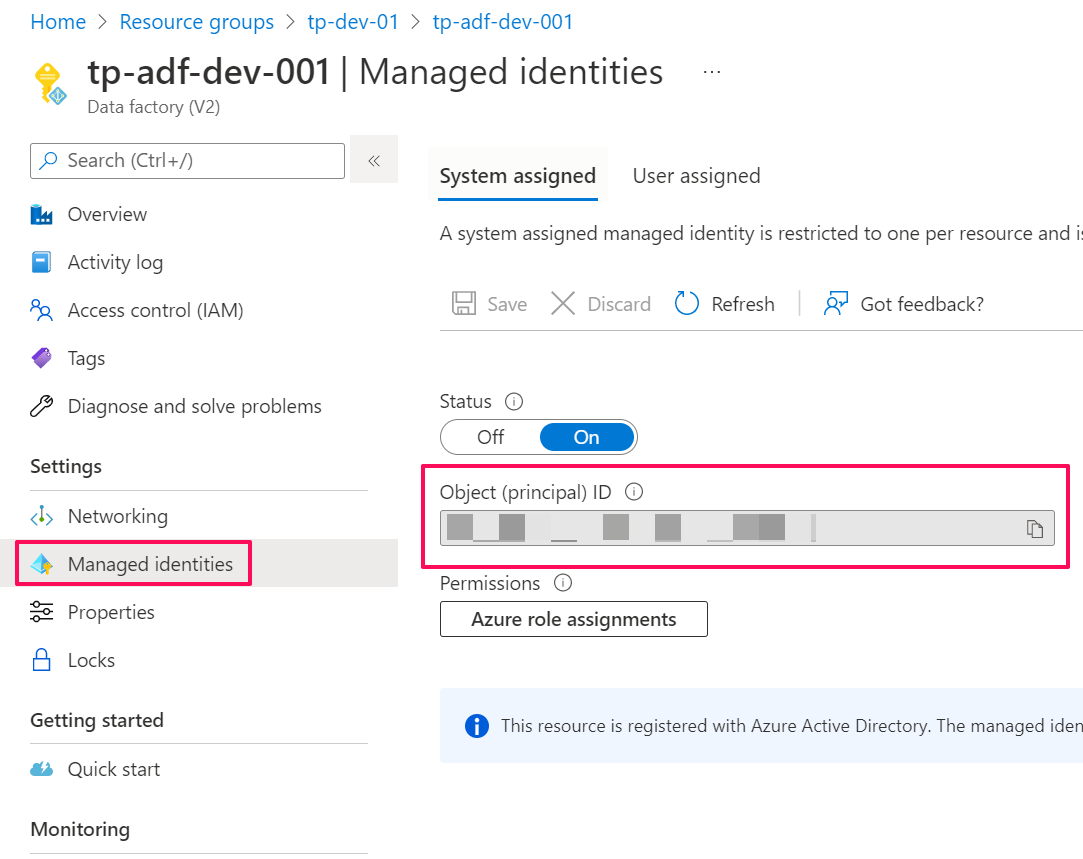
The user can edit the parameters at this point.





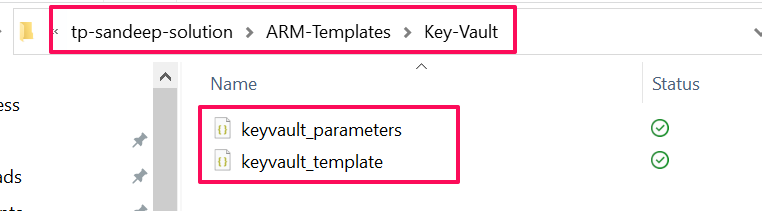


Once the data factory is created, we have to copy the Object ID. This Object ID is used in deploying the Azure key vault.

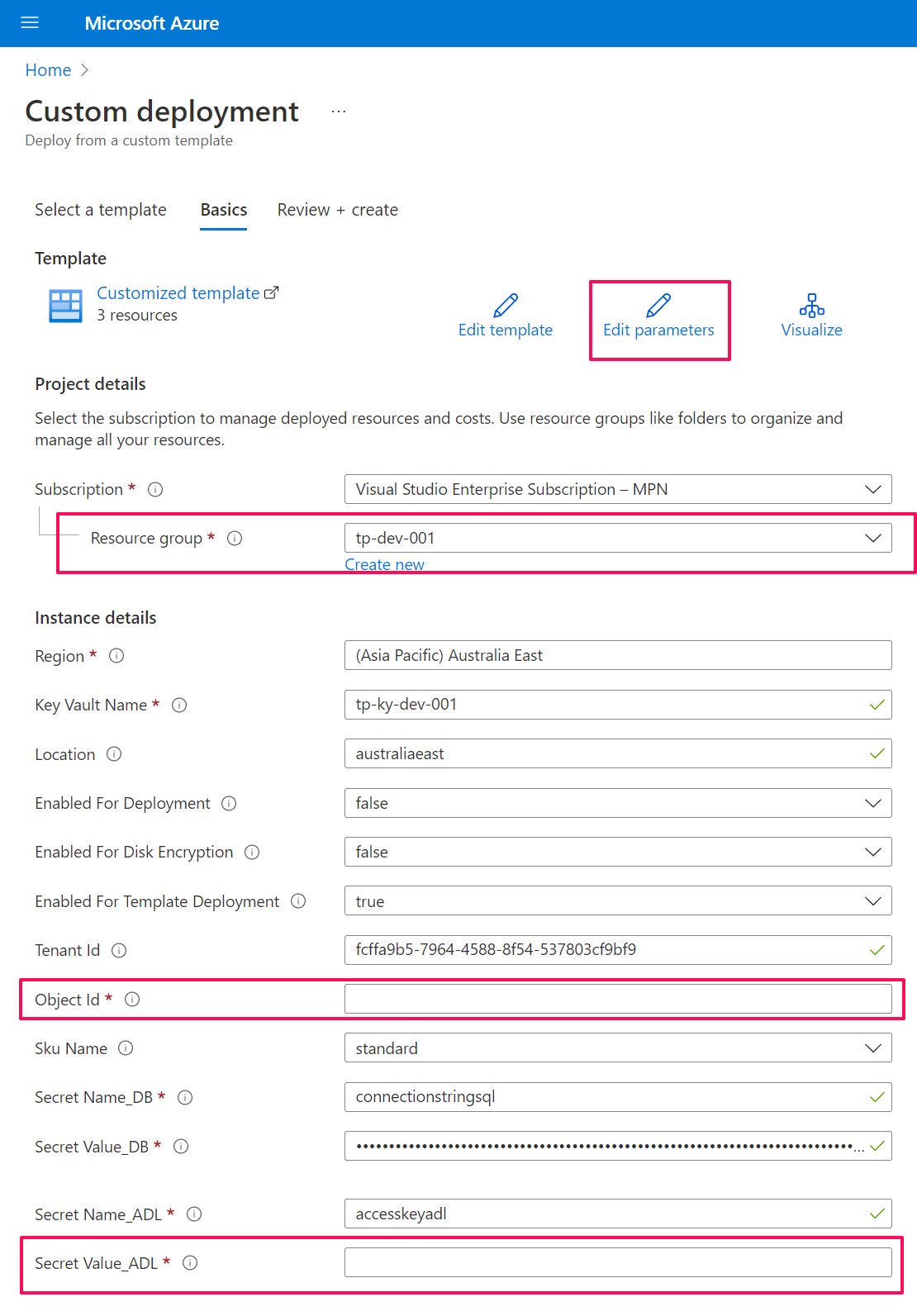


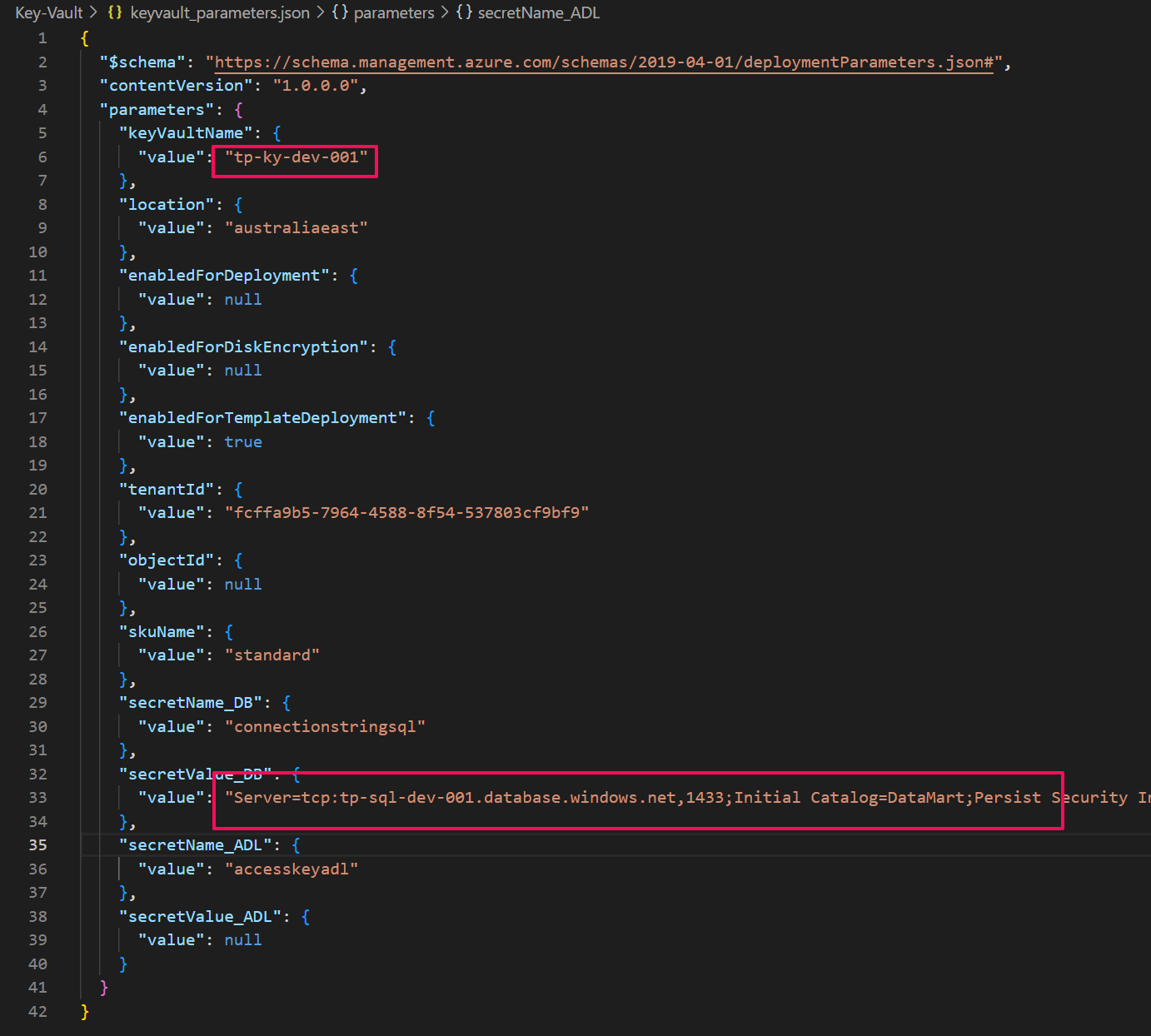
## Azure Key Vault

Navigate to ARM-template in the tp-sandeep-solution and select Key-Vault. Upload the template file followed by the parameter file.



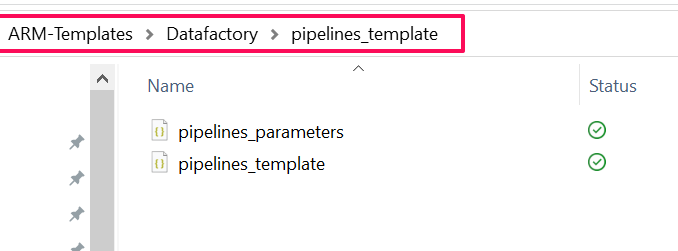
At this point, we have to use the access key of data lake and Object ID of data factory that we copied earlier. The user can also edit the parameters.





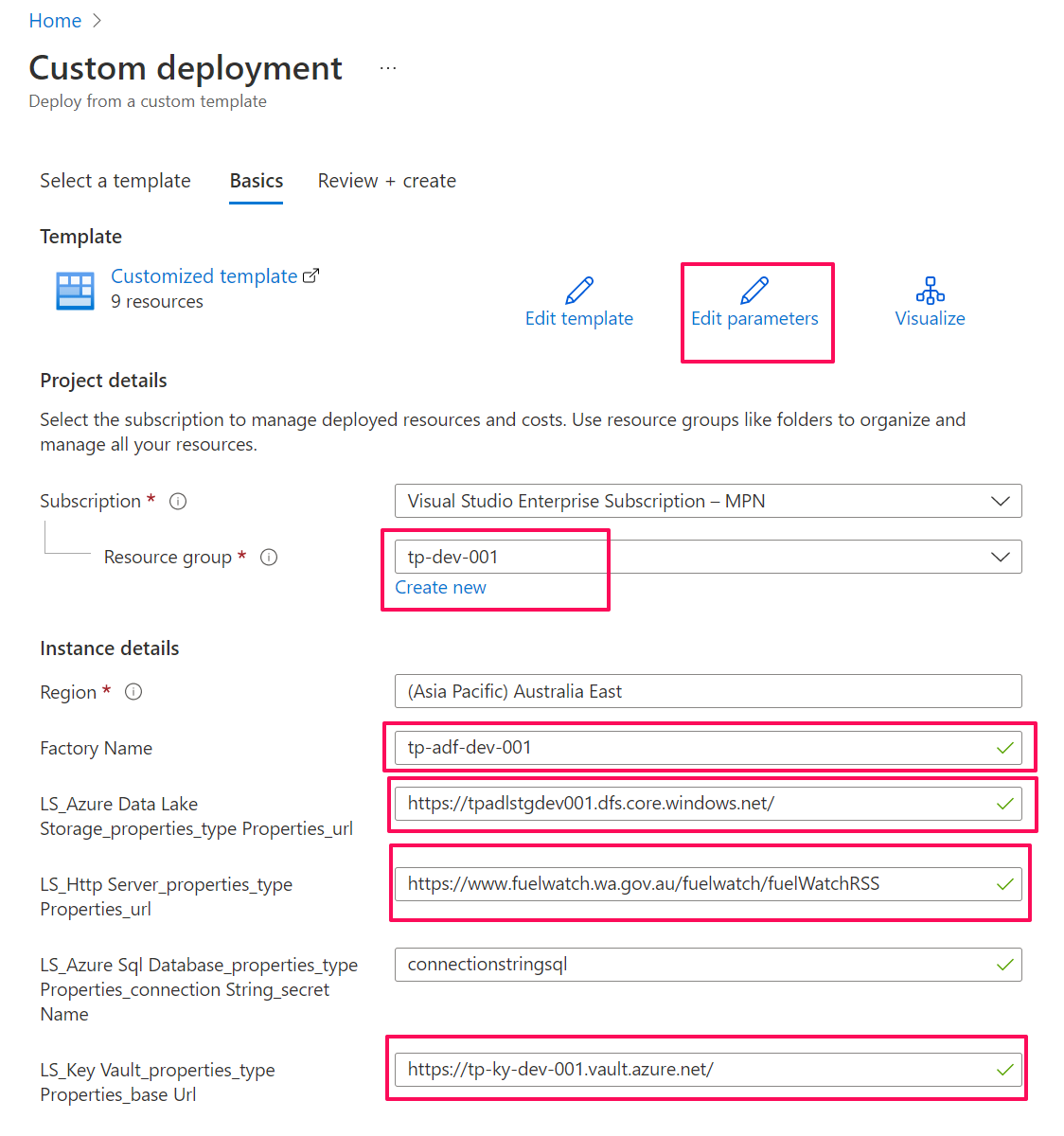
## Azure Data Factory Pipeline Deployment

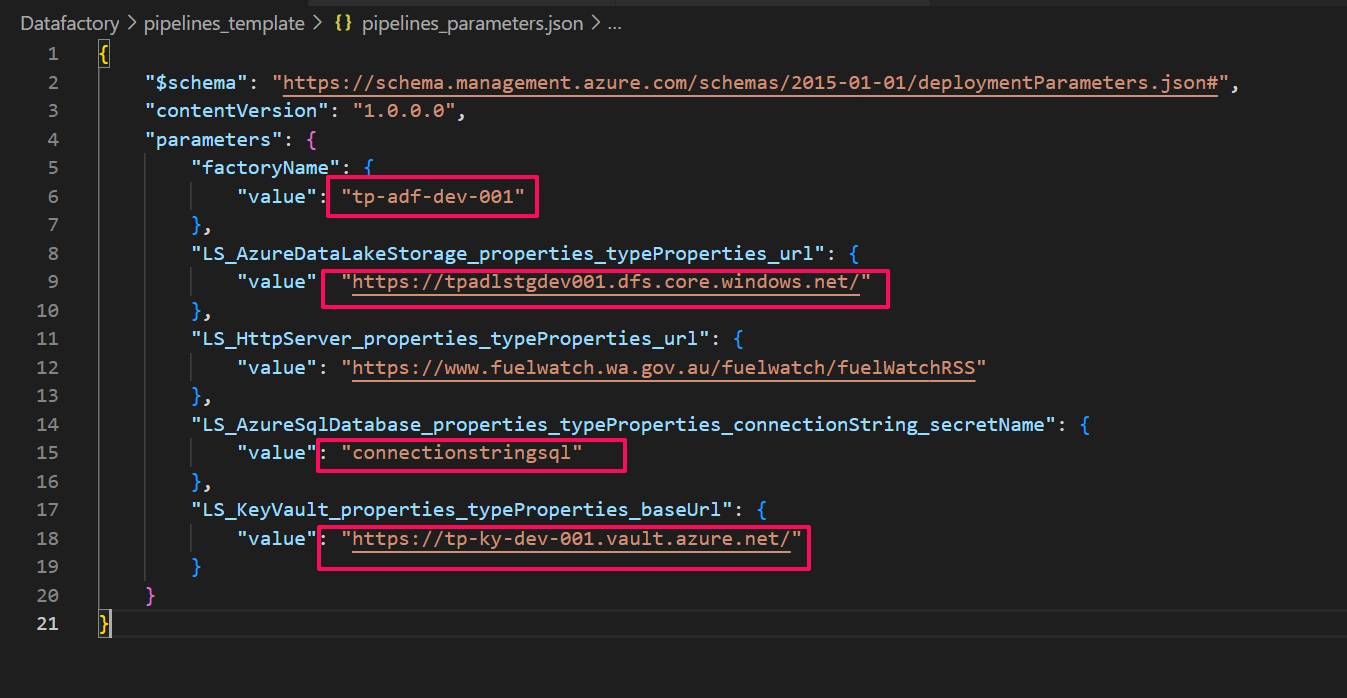
Navigate to ARM-template in the tp-sandeep-solution and select Data factory. The user should select pipelines\_template. Upload the template file followed by the parameter file.



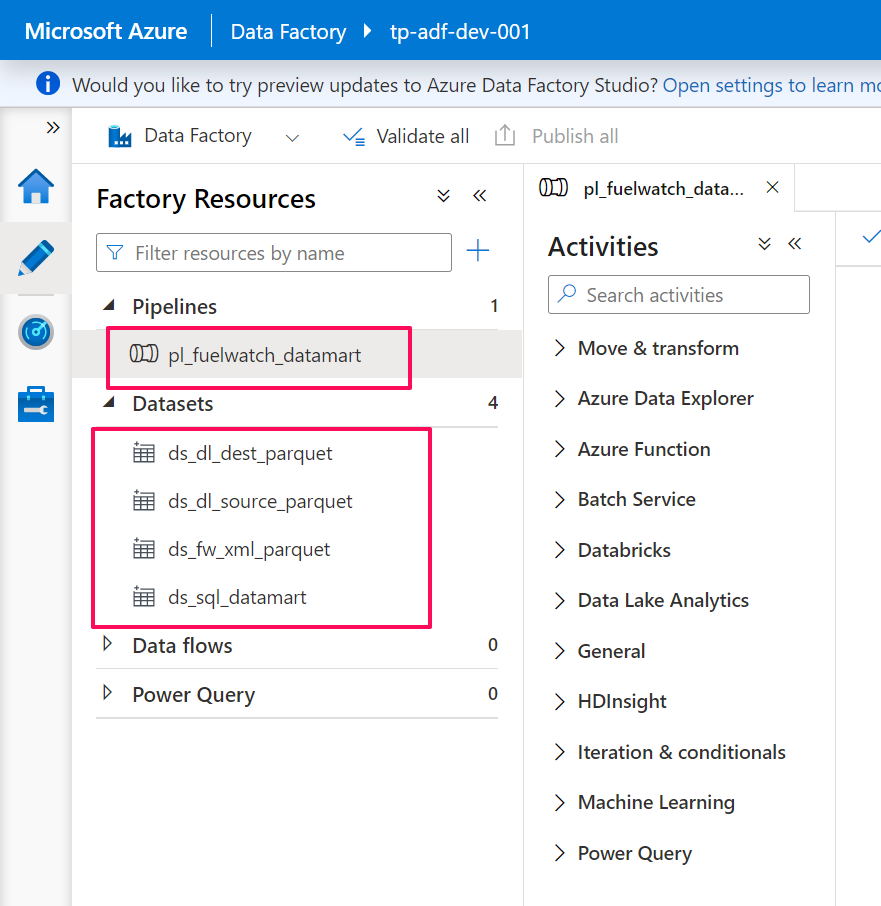
As the data factory has already been created, now we are deploying the pipelines as shown in the below image.

The user can edit the naming parameters at this point.

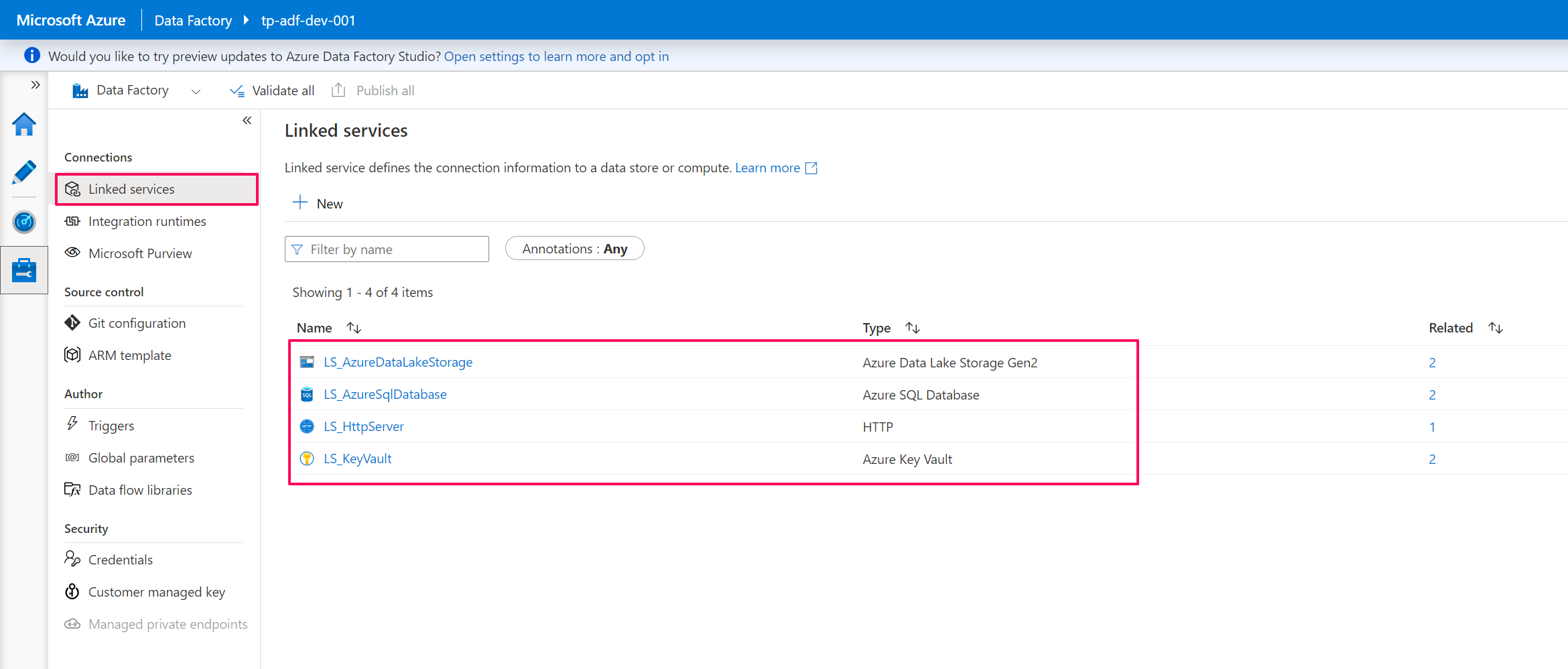


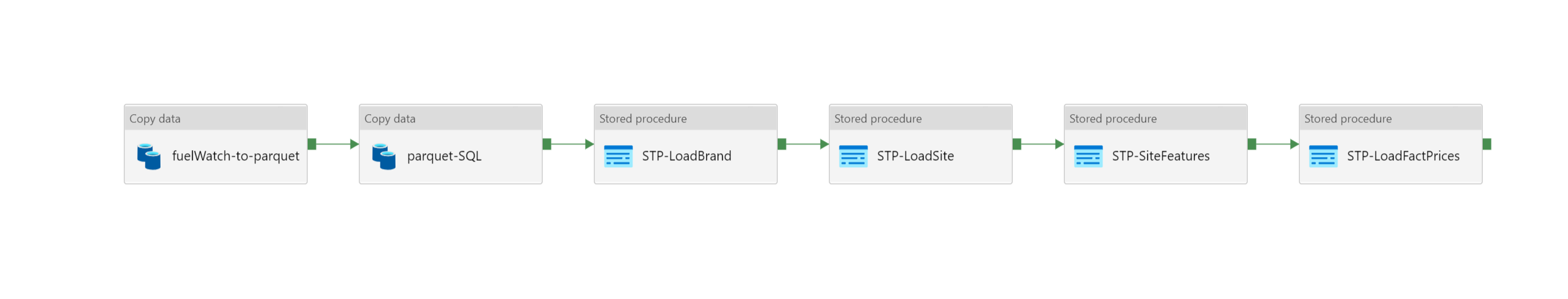


Once the pipelines are deployed, the user should be able to see the resources in the data factory.



Make sure to test the linked services connections. Once the connections are successful, please run the pipeline.





Once the pipelines are successful, the user should be able to see the data in the tables.

