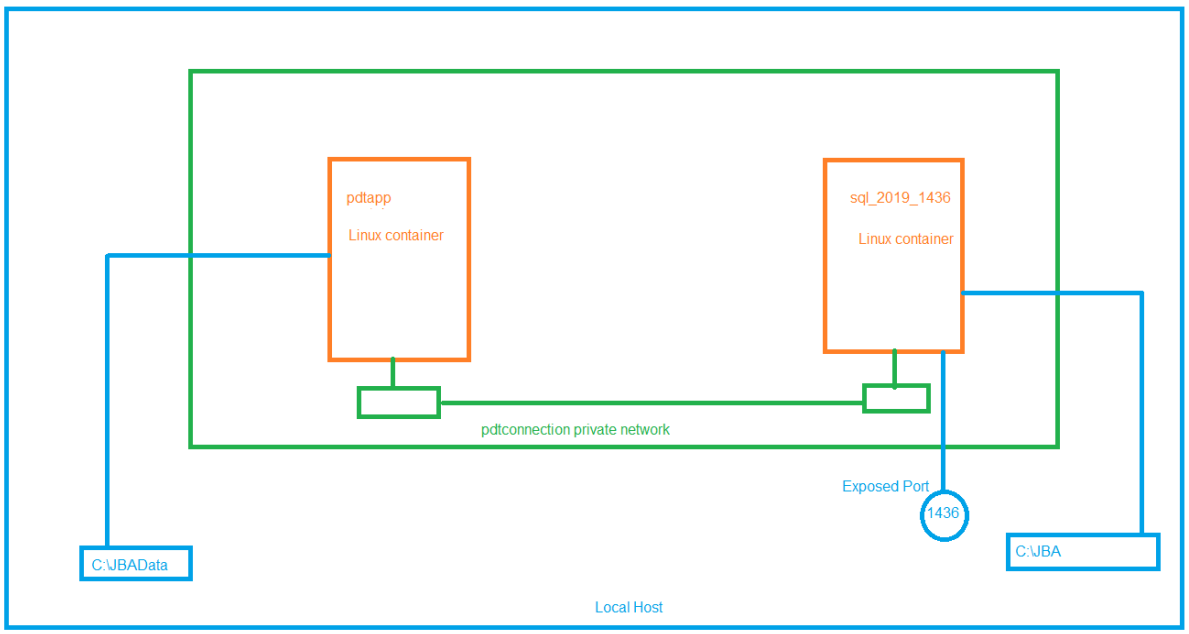
Challenge Solution

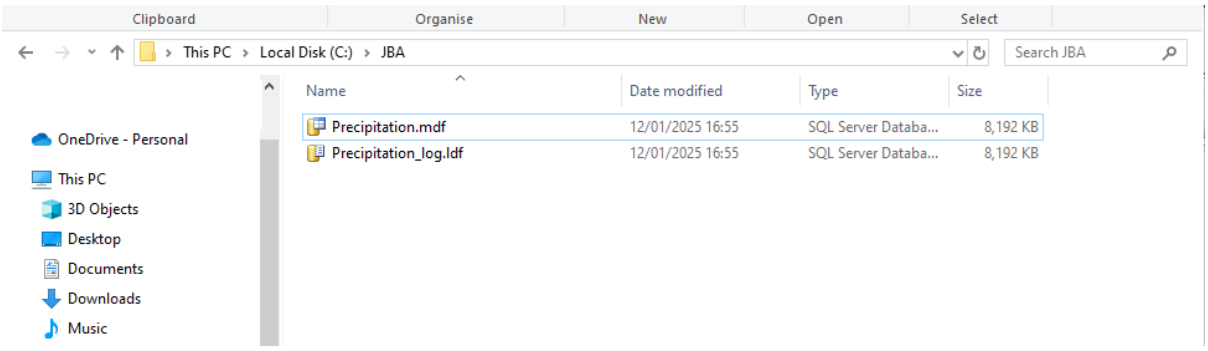
This document contains details for the coding exercise challenge. The following sections detail the solution design and give an introduction to running the application.

1. Solution Design

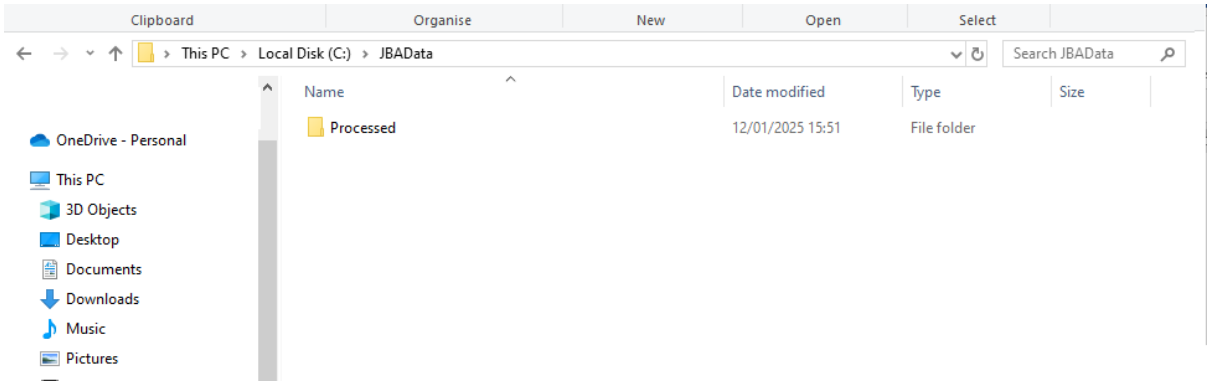
The solution of the challenge exercise Precipitation Data Transform, has been implemented with Linux containers. The architecture of the solution application is as shown in the following image:



It consists of two Linux containers. The pdtapp Linux container and the SQL Server sql_2019_1436 Linux container. The solution binds to two local host folders. On a Windows host machine, these are the C:\JBA folder and contain the solution SQL Server database. See the following image:



And the folder C:\JBAData, which contains the precipitation data files to import to the database. Imported files will be archived into a subfolder named “processed”. See the following image:

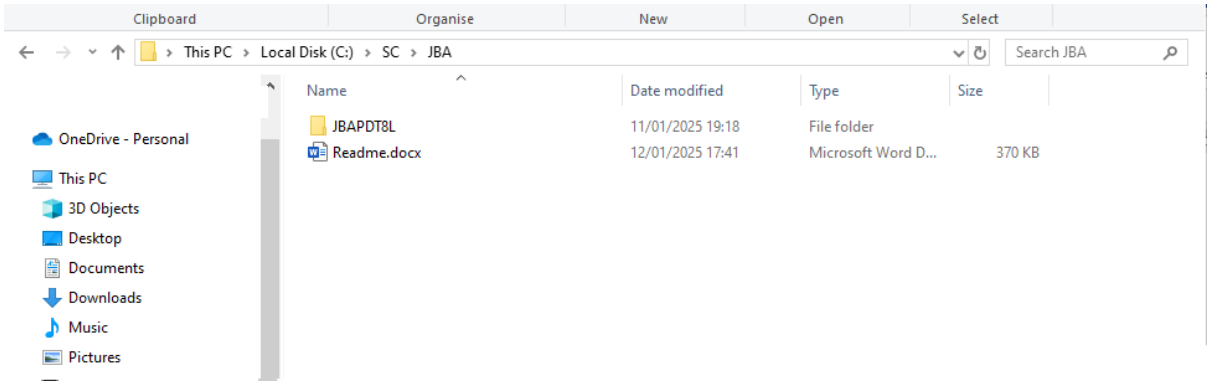


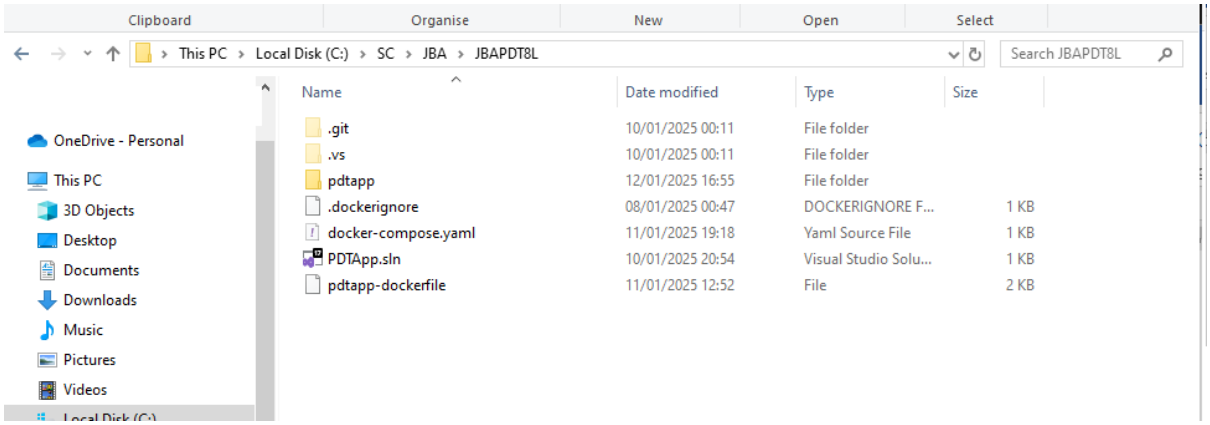
Once the application is running, dropping a file into the C:\JBADat will be picked up by the pdtapp application and the data contained will be imported into the “Precipitation” database located in the C:\JBA folder.

2. Running the “pdtapp” application

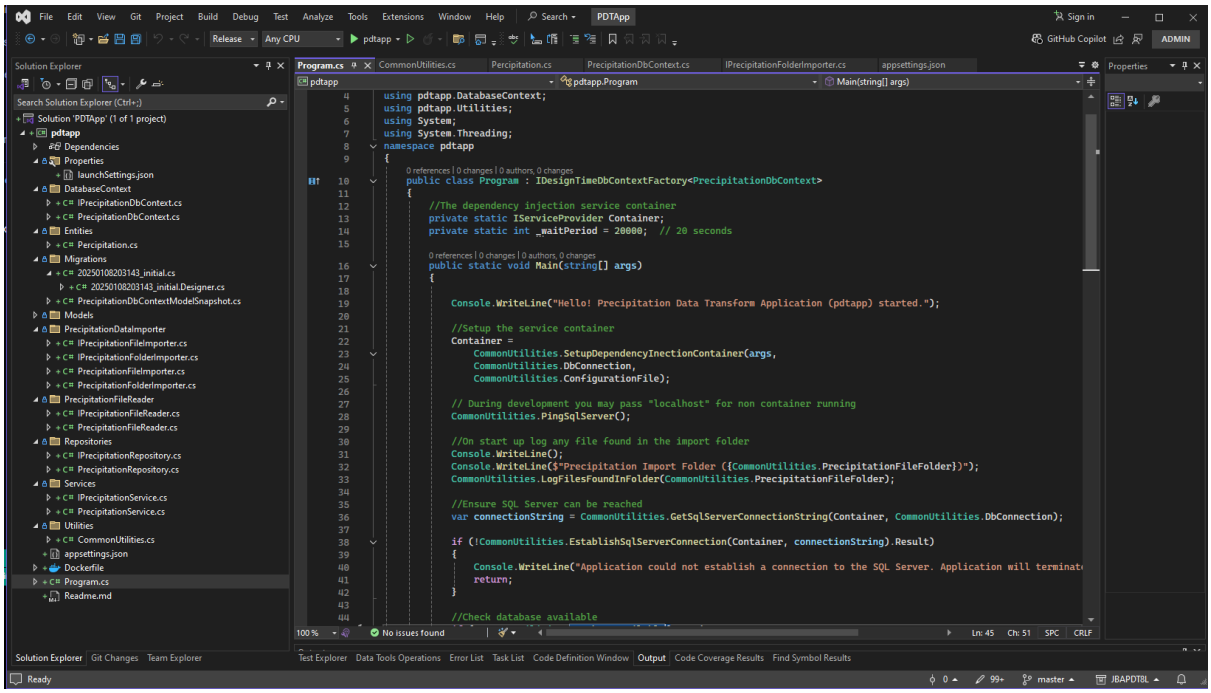
The application code is written in .Net 8 and needs to be copied to the C:\ drive. The code is found in the supplied zip folder. A Readme.md file can be found in the source folder to aid in the development and running of the application. Open the file in your preferred editor (VS Code) and browse the information therein.

The JBA code folder found in the zip file should be copied to the C:\ drive. Once copied code should be located in the subfolder C:\JBA\JBAPTD8L and as shown on the following image;

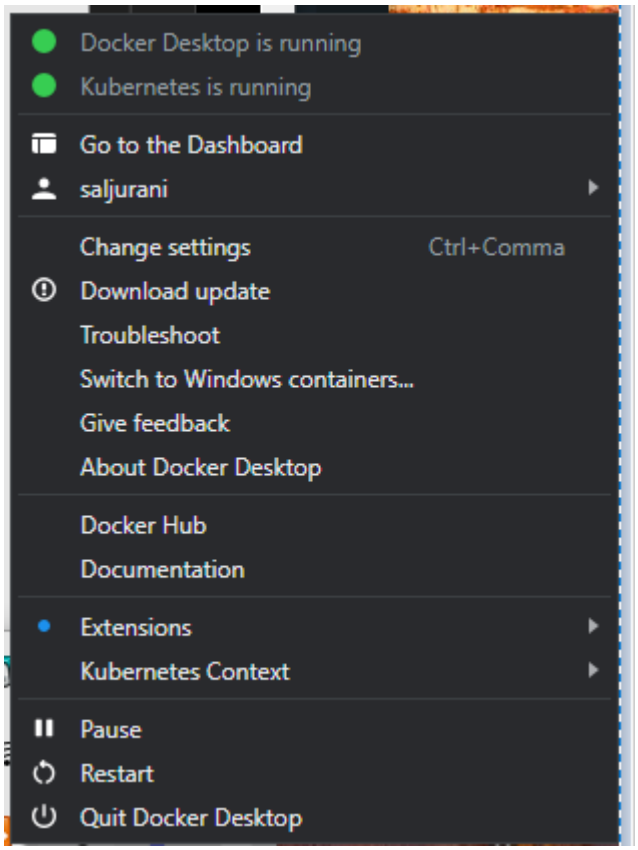




The structure of the solution is shown on the following image:



The code subfolder contains a docker-compose file “docker-compose.yaml” file, which will be used to deploy the solution application to Docker Desktop environment. The Docker Desktop should be switched to the Linux containers , if not click on the menu item to switch to Linux containers See The following image:



To run the “pdtapp” application perform the following steps:

- Open a cmd console with admin privileges and navigate to C:/SC/JBA/JBAPDT8L

- To Build the application and see if all is well, copy the following docker command to the cmd wind and press Enter:
  docker compose -f "C:/SC/JBA/JBAPDT8L/docker-compose.yaml" --project-directory "C:/SC/JBA/JBAPDT8L" build

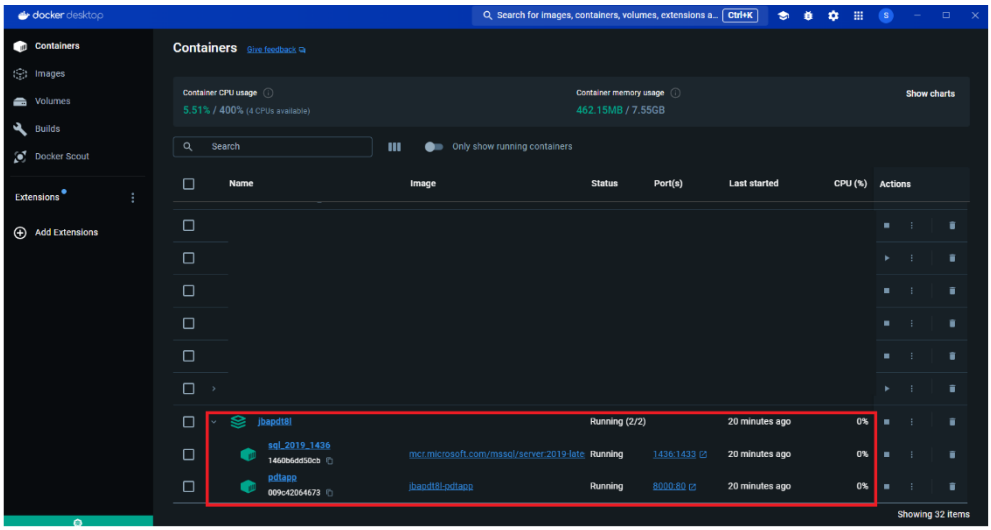
- To Build the application and run it on Docker Desktop environment, copy and execute the following docker-compose up command:

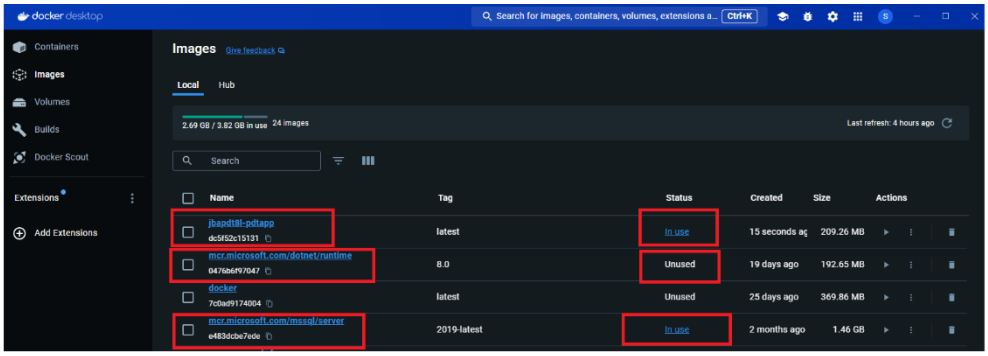
  docker-compose -f "C:/SC/JBA/JBAPDT8L/docker-compose.yaml" up --force-recreate –detach

  This will build and deploy the application into the Docker Desktop platform.

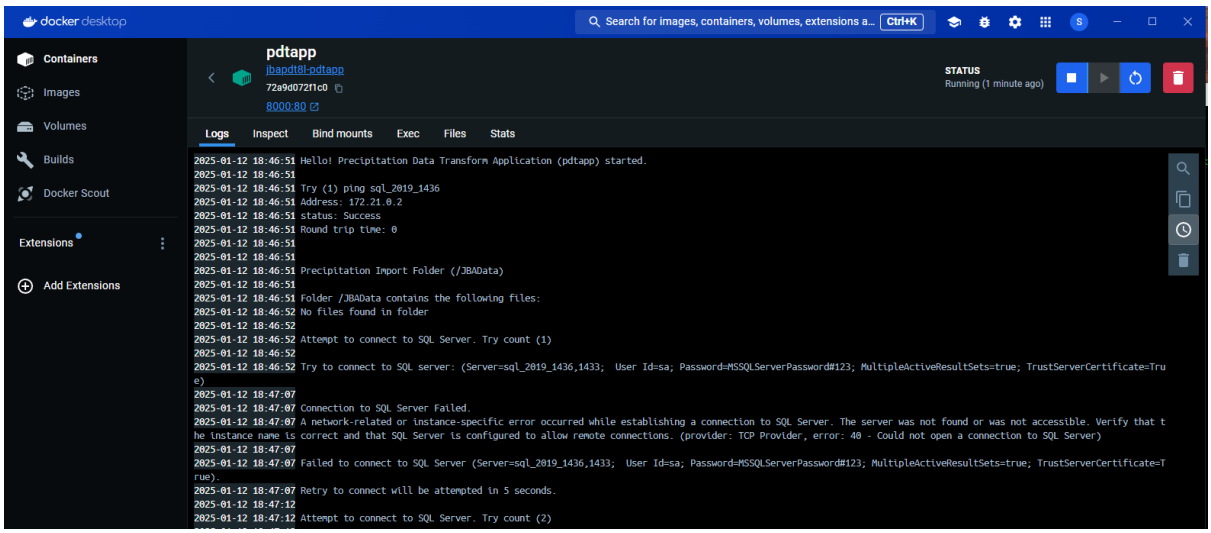
```
C:\SC\JBA\JBAPDT8L>docker-compose -f "C:/SC/JBA/JBAPDT8L/docker-compose.yaml"  up --force-recreate --detach
time="2025-01-12T18:12:46Z" level=warning msg="C:\\SC\\JBA\\JBAPDT8L\\docker-compose.yaml: `version` is obsolete"
[+] Building 117.2s (18/18) FINISHED
 => [pdtapp internal] load build definition from pdtapp-dockerfile
 => => transferring dockerfile: 1.45kB
 => [pdtapp internal] load metadata for mcr.microsoft.com/dotnet/sdk:8.0
 => [pdtapp internal] load metadata for mcr.microsoft.com/dotnet/runtime:8.0
 => [pdtapp internal] load .dockerignore
 => => transferring context: 464B
 => [pdtapp build 1/7] FROM mcr.microsoft.com/dotnet/sdk:8.0@sha256:f25e4f51fa06e3b14af1a1135013a3e96055b76caa0e76afc0096d64a77879fd
 => [pdtapp base 1/2] FROM mcr.microsoft.com/dotnet/runtime:8.0
 => [pdtapp internal] load build context
 => => transferring context: 52.73kB
 => CACHED [pdtapp build 2/7] WORKDIR /src
 => [pdtapp build 3/7] COPY [pdtapp/pdtapp.csproj, pdtapp/]
 => [pdtapp build 4/7] RUN dotnet restore "./pdtapp/pdtapp.csproj"   --interactive
 => [pdtapp build 5/7] COPY . .
 => [pdtapp build 6/7] WORKDIR /src/pdtapp
 => [pdtapp build 7/7] RUN dotnet build "./pdtapp.csproj" -c %BUILD_CONFIGURATION% -o /app/build
 => [pdtapp publish 1/1] RUN dotnet publish "./pdtapp.csproj" -c %BUILD_CONFIGURATION% -o /app/publish /p:UseAppHost=false
 => CACHED [pdtapp base 2/2] WORKDIR /app
 => CACHED [pdtapp final 1/2] WORKDIR /app
 => [pdtapp final 2/2] COPY --from=publish /app/publish .
 => [pdtapp] exporting to image
 => => exporting layers
 => => writing image sha256:dc5f52c151316a16028ed59df8777086c8ac5646075a9c68b3b634afdd595c35
 => => naming to docker.io/library/jbapdt8l-pdtapp
[+] Running 3/3
 ⠿ Network jbapdt8l_pdtconnection  Created
 ⠿ Container sql_2019_1436         Started
 ⠿ Container pdtapp                Started
```

- Open the Docker Desktop Console and confirm the application is running as expected. Confirm the image is created, see the highlighted boxes on the following images:

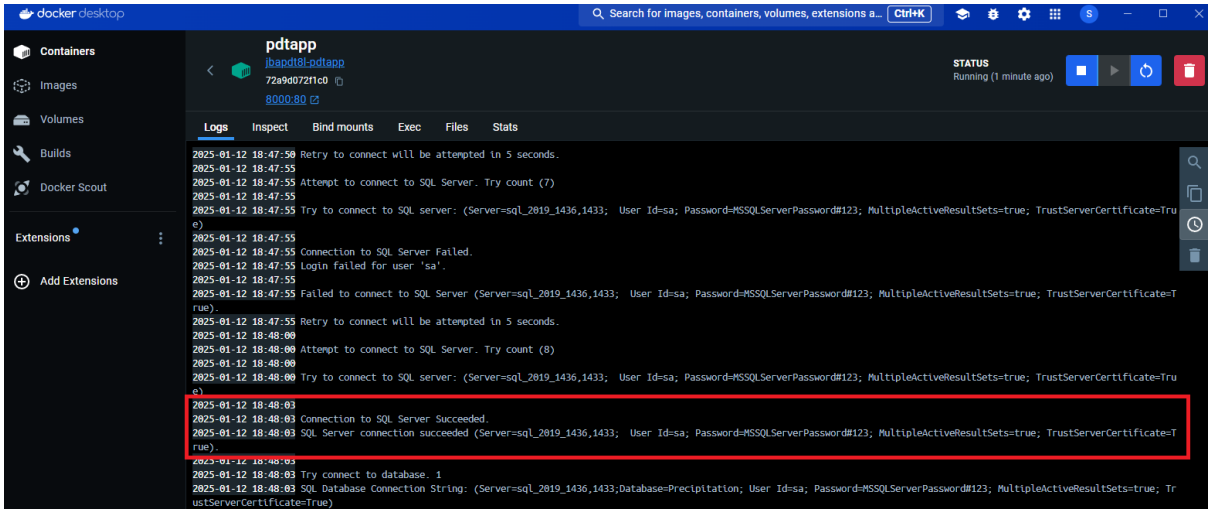




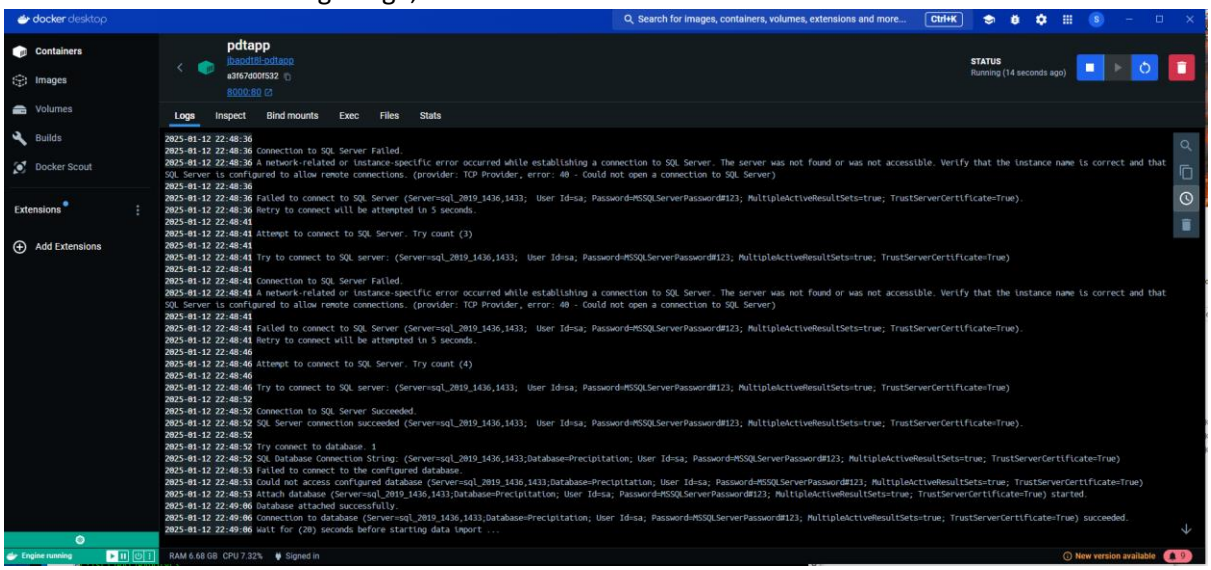
- Open the pdtapp container and navigate to the Logs view. This is done by clicking on the pdtapp container as shown on the previous images and opening up the Logs view. See the following image.

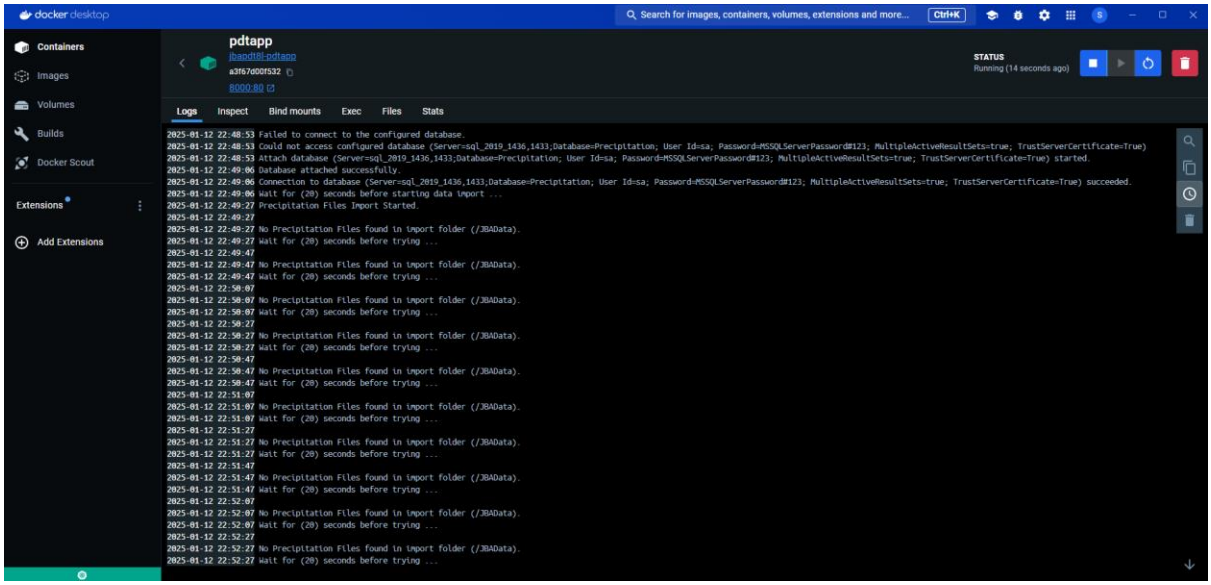


Initially, the SQL Server might not be available and connection to the SQL Server will fail. The application will keep trying to connect until it succeeds. See the following image:

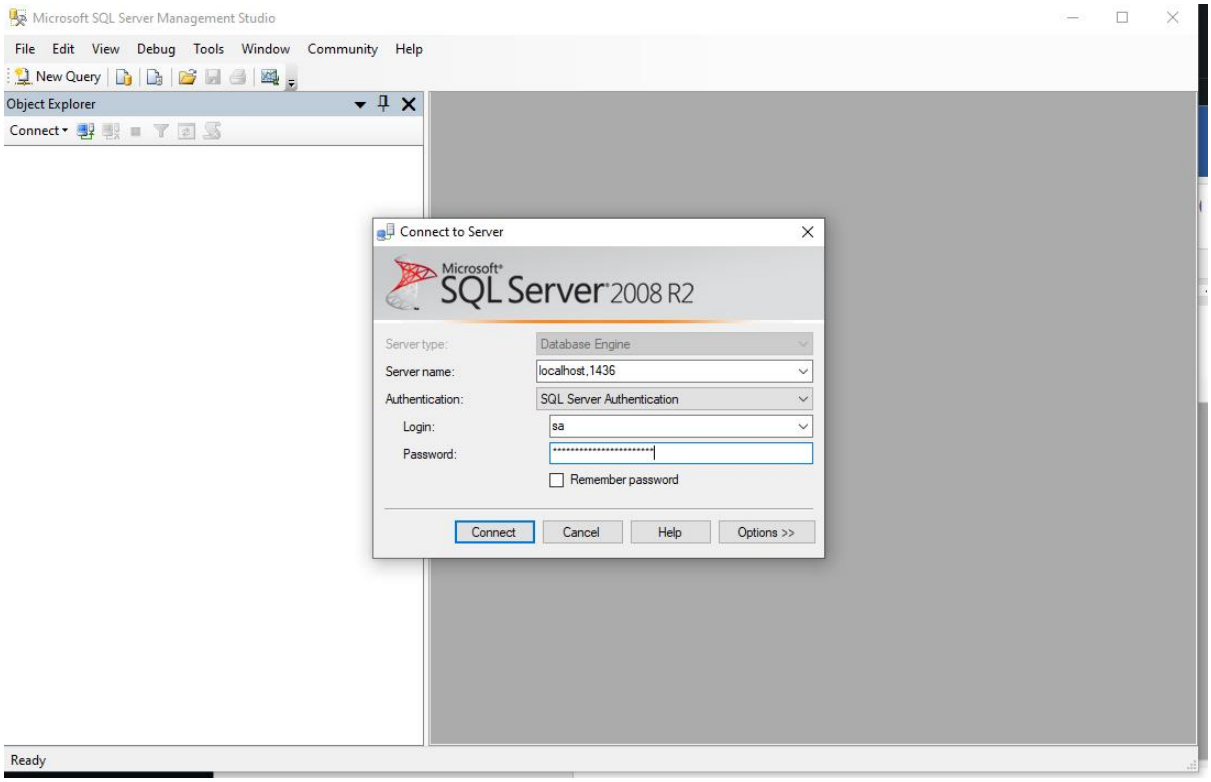


Once a connection to the SQL Server succeeded, the application tries to establish a connection to the configured database “Precipitation”. On the first run, the database may not have been attached and the connection will fail. The application will try to attach the database until it succeeds. See the following image;

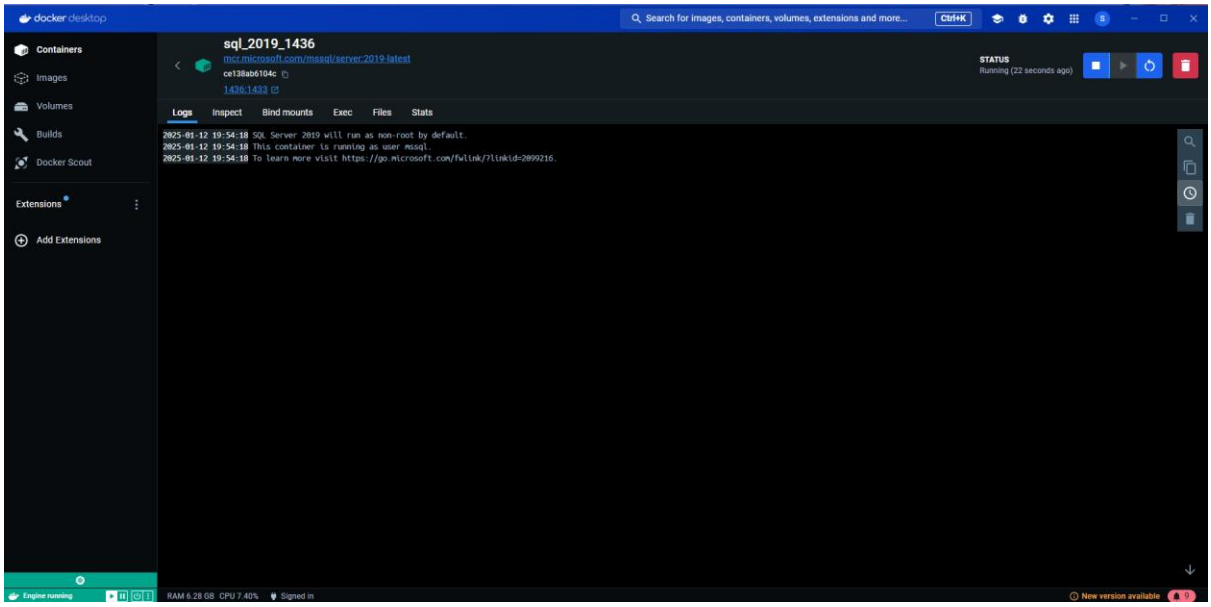




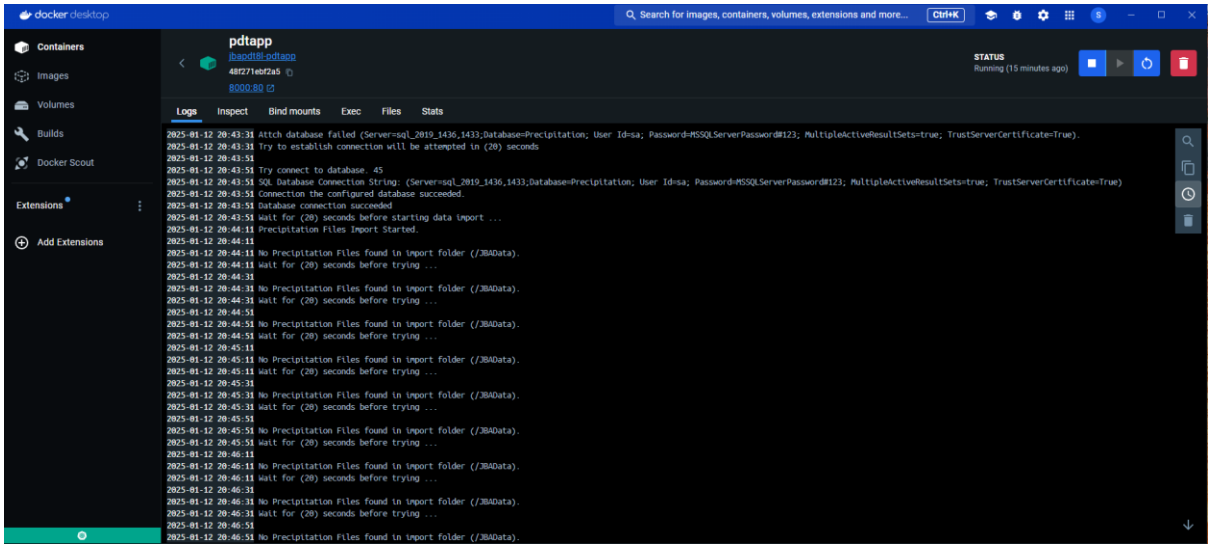
The database can be also be attached manually from the local host using a MS SQL Server Management Studio. See the following image. Use the following password: MSSQLServerPassword#123



- Open the sql_2019_1436 container Logs view and confirm the SQL Server is running as expected. Initially, it will start up by initializing the server and it might take some time to setup the SQL Server and be ready to serve incoming sql quires.

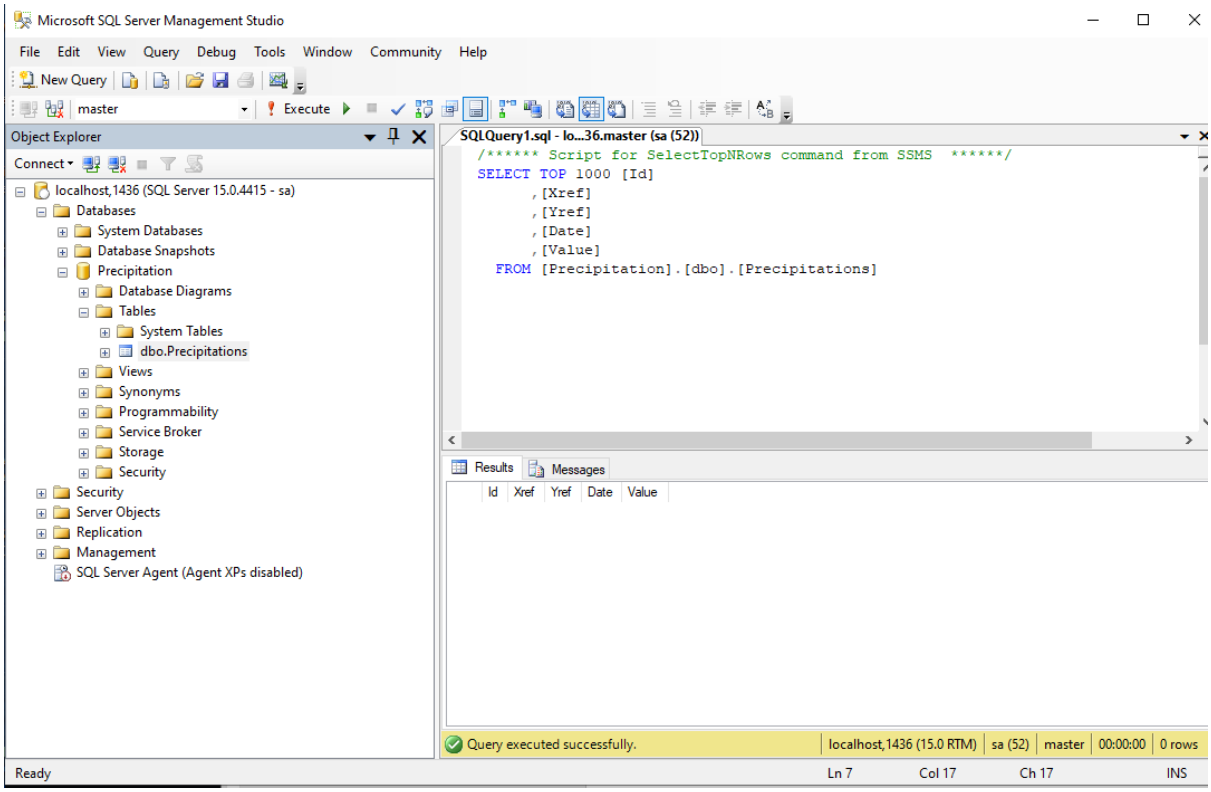


Once the application connects successfully and the database is attached it should be possible to drop files into the C:\JBAData folder and observe the processing of the precipitation files. See the following image for the pdtapp logs view:

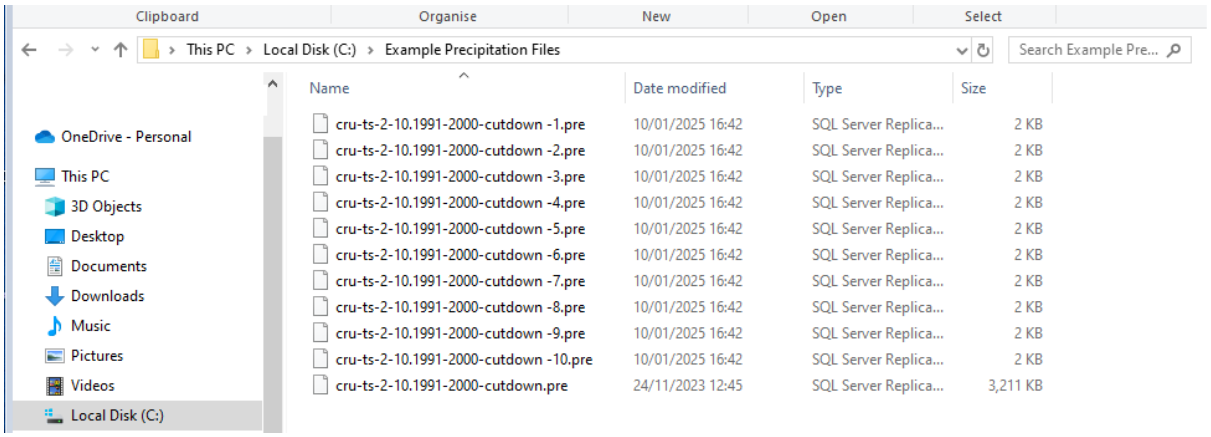


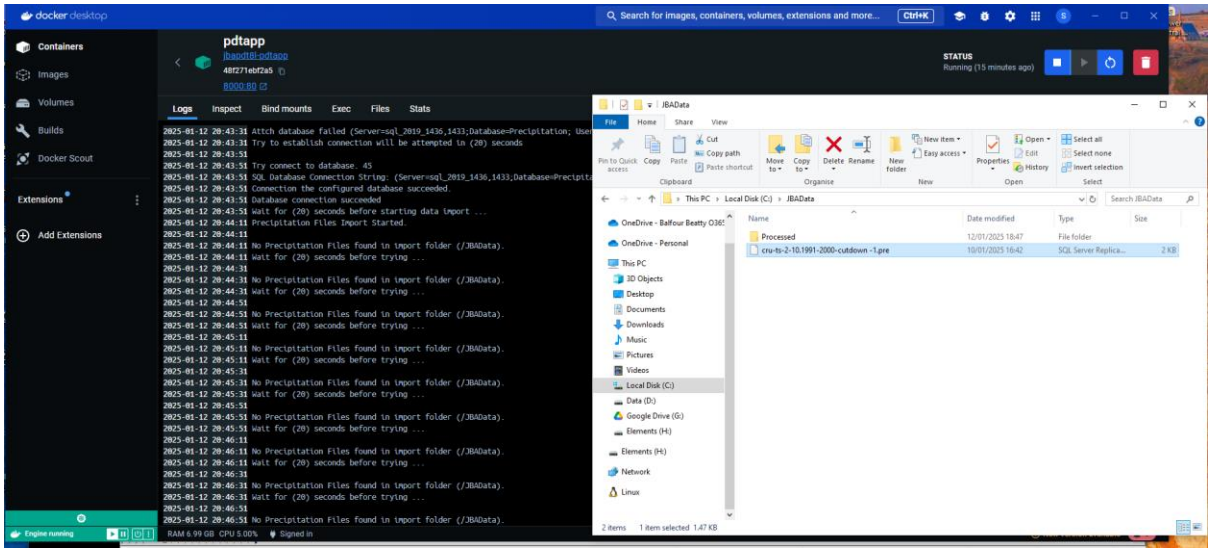
- Drop a Precipitation file into the C:\JBAData and watch the “pdtapp” from the container Logs console consume the file and import the data into the database. Example files can be found in the zipped folder under the subfolder “C:\Example Precipitation Files”.

  Initially, the database is empty and no data is found in the precipitations table.

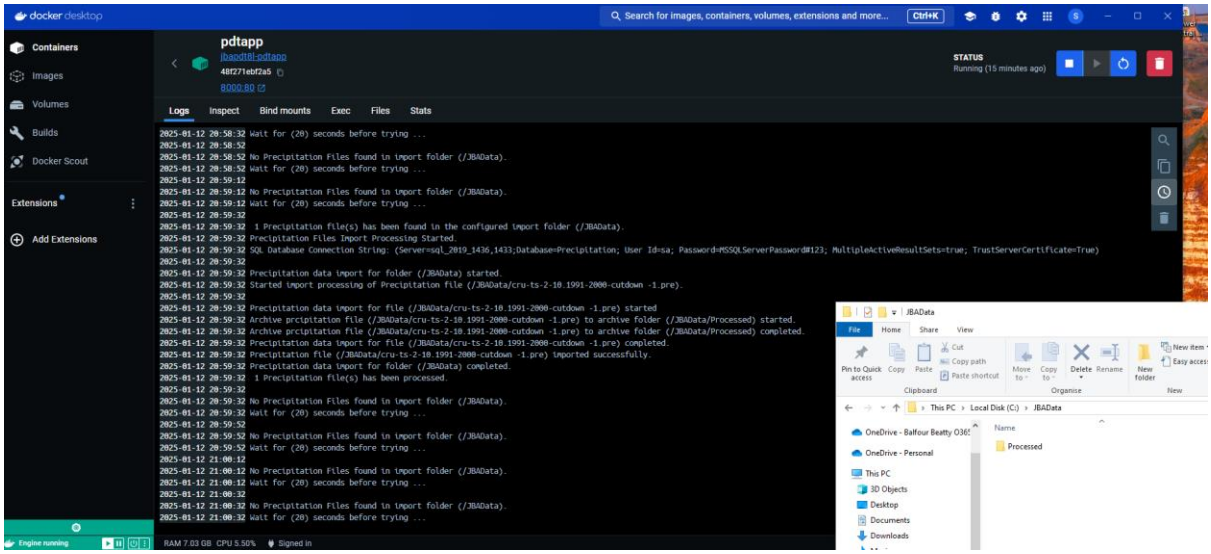


Select file “cru-ts-2-10.1991-2000-cutdown -1.pre” and drop it on the C:\JBAData folder and watch the pdtapp Logs view from the Docker Desktop. It should show a similar view to the following:

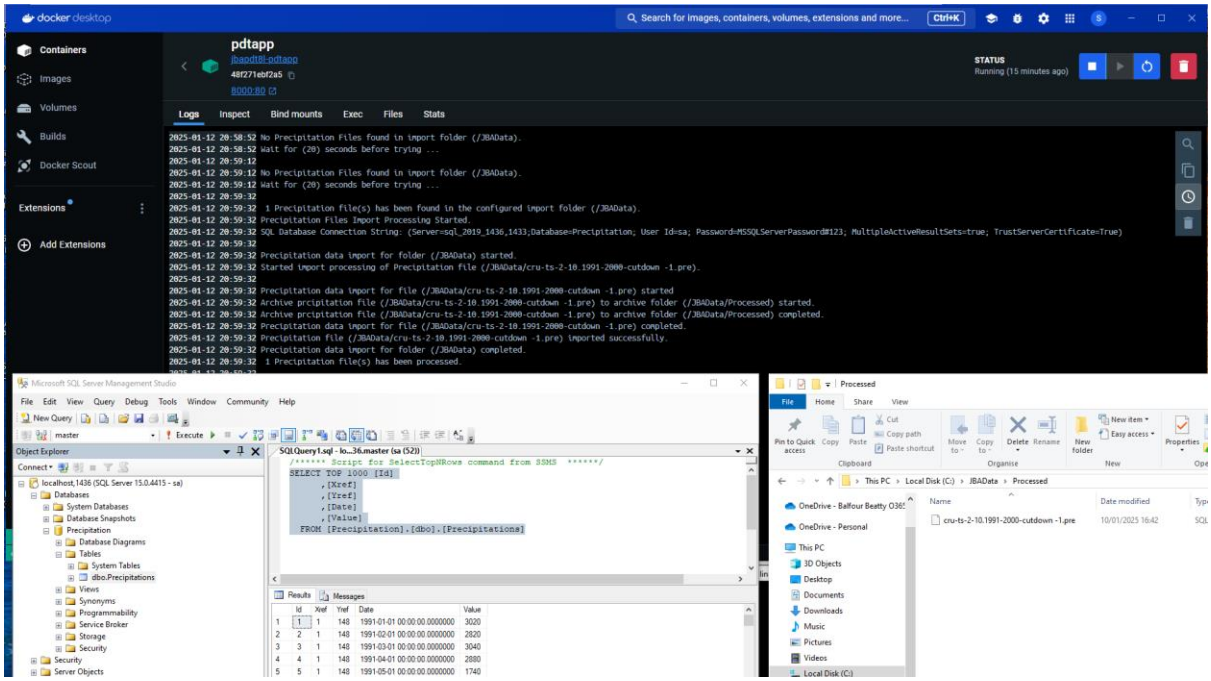




After a period of time the precipitation file is processed and the Logs view shows an output similar to the following:

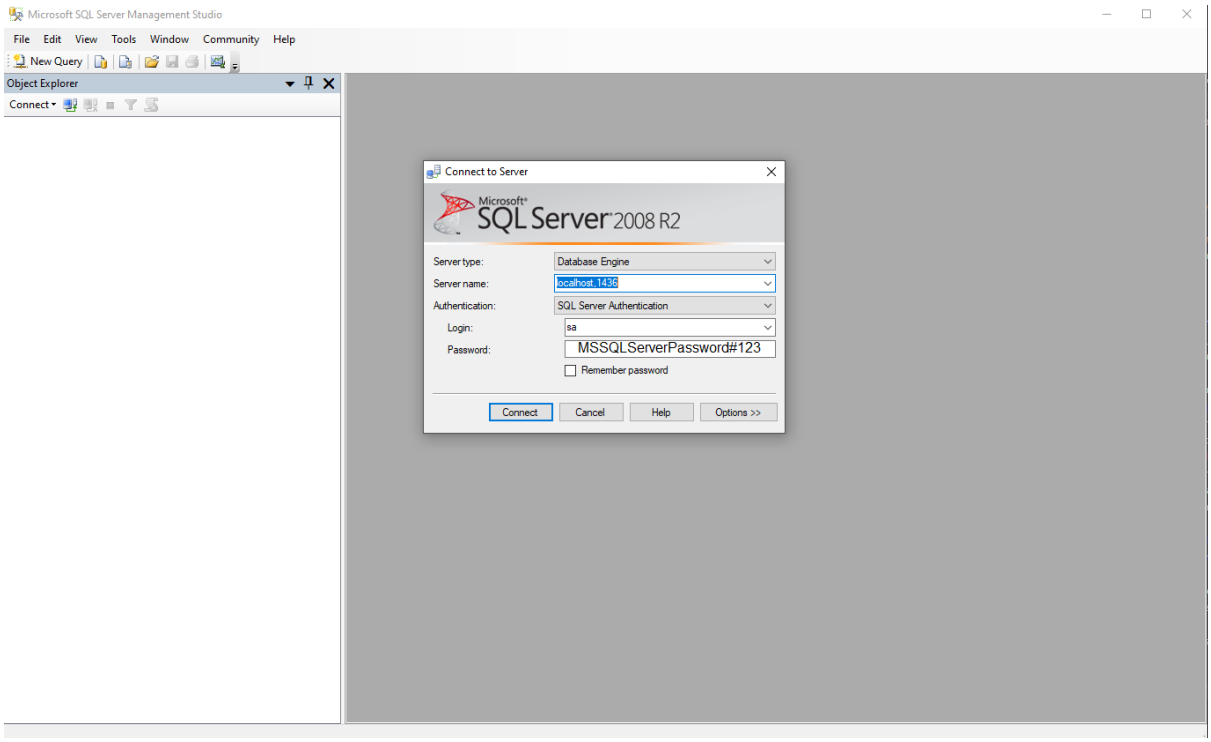


The file is processed and archived into the C:\JBAData\Procssed folder and data is imported into the Precipitation database.

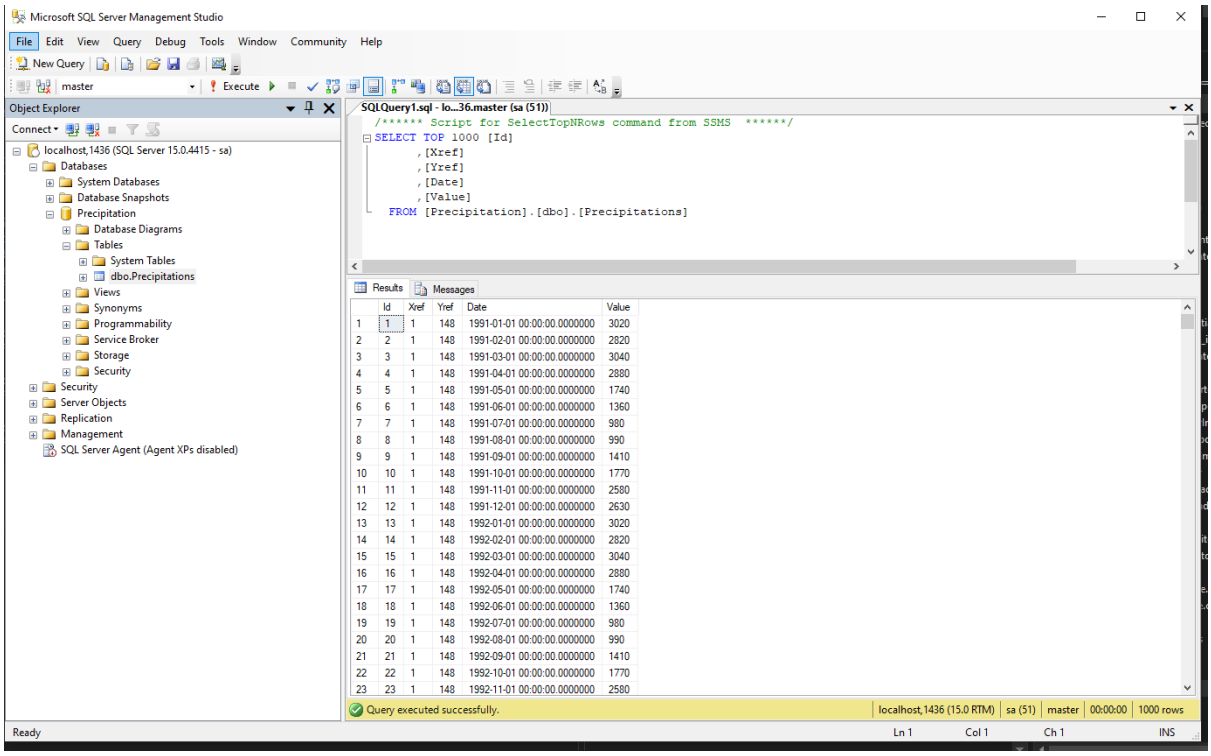


- To connect the sql_2019_1436 database from the local host, open a “SQL Server Management Studio” with admin privileges. Create a database connection and when prompted for credentials fill in the dialogue boxes as follows:
  - Server type: Database Engine
  - Server name: localhost,1436
  - Authentication: SQL Server Authentication
  - Login: sa
  - Password: MSSQLServerPassword#123

See image below.



Once connected navigate to the Precipitations table on the Precipitation database. Perform a selection query on the table. It should show something similar to the following:



- To Stop the application copy and execute the following docker-compose down command:

  docker-compose -f "C:/SC/JBA/JBAPDT8L/docker-compose.yaml" down

  This should produces an output similar to the following image:

```
C:\SC\JBA\JBAPDT8L>docker-compose -f "C:/SC/JBA/JBAPDT8L/docker-compose.yaml"  down
time="2025-01-12T18:41:41Z" level=warning msg="C:\\SC\\JBA\\JBAPDT8L\\docker-compose.yaml: `version` is obsolete"
[+] Running 3/3
 ⠿ Container pdtapp                Removed
 ⠿ Container sql_2019_1436         Removed
 ⠿ Network jbapdt8l_pdtconnection  Removed

C:\SC\JBA\JBAPDT8L>
```