



MARINFO Manual Infrastructure Design

Version 1.0.0

Date: 05/05/2023

Table of Contents

1. Introduction	5
1.1 Summary	5
1.2 Project Description.....	5
2. Solution architecture	7
2.1 Data Factory	7
2.1.1 General information	7
2.1.2 Implementations	8
2.2 Logic App	10
2.3 Azure Data Lake	11
2.4 Databricks.....	12
2.5 Azure SQL Database.....	13
3. Azure prerequisites, permissions and platform	15
3.1 Azure Devops	15
3.1.1 Azure Devops project site	16
3.1.2 Extensions	16
3.2 Azure Subscription.....	17
3.3 Resource Groups.....	17
4. Functional Design.....	19
4.1 MARINFO data	19
4.1.1 Data Ingestion	19
4.1.2 Data validation.....	21
4.1.2.1 File types	24
4.1.2.1 MARINFO Synchronization	25
4.1.3 Data Consumption.....	25
4.1.4 Serve	25
4.1.4.1 Azure SQL DB.....	26
4.1.4.2 ODBC & JDBC	26
4.1.4.3 SSN Exports.....	27
4.1.4.4 Web Services	28
4.1.4.5 Power BI	35
4.1.4.6 Functional Monitoring	35
4.2 MARINFO History	35
4.3 MARINFO Verification	37
5. Data governance and security.....	38
5.1 Azure governance.....	38
5.1.1 Naming conventions	38
5.1.1.1 Azure resources	38
5.1.1.2 Service principals	38
5.1.2 Logging, monitoring and alerting	38
5.1.3 Availability.....	39

5.2	Azure component security	39
5.2.1	Azure Log Analytics Workspace.....	39
5.2.1.1	General.....	39
5.2.1.2	Security	40
5.2.1.3	High-availability and disaster recovery.....	40
5.2.2	Data Storage data lake.....	40
5.2.2.1	Security	41
5.2.3	High-availability and disaster recovery	41
5.3	Azure Key Vault	41
5.3.1	Access Policies.....	42
5.3.2	Security.....	42
5.3.3	High-availability and disaster recovery	42
5.4	Azure Databricks	42
5.4.1	General	42
5.4.2	Databricks cluster	43
5.4.3	Manual actions	43
5.4.4	Permissions	43
5.4.5	Security.....	43
5.4.6	High-availability and disaster recovery	43
5.5	Virtual Networks.....	43
5.6	Azure Data Factory.....	44
5.6.1	General.....	44
5.6.2	Post-Setup Actions	44
5.6.3	Security.....	44
5.6.4	High-availability and disaster recovery	45
5.7	Azure SQL	45
5.7.1	Azure SQL Server	45
5.7.2	Azure SQL database	46
5.7.3	Permissions	46
5.7.4	Security.....	47
5.7.4.1	Azure SQL User Accounts	47
5.7.4.2	Advanced Threat Protection.....	47
5.7.4.3	Transparent Data Encryption	48
5.7.4.4	Azure SQL Firewall	49
5.7.5	High-availability and disaster recovery	49
5.8	Disaster Recovery	49
5.8.1	General strategy	49
5.8.2	Microsoft documentation	51
6.	Appendix.....	52
6.1	Azure Devops Service principal Azure AD permission configuration	52
6.2	Create an Azure Key Vault-backed secret scope.....	57
6.3	Datalake Access guide	59
6.3.1	Azure storage explorer	59
6.4	Adding manual files to the datalake.....	63
6.5	Data Factory guide	64
6.5.1	General information	64
6.5.2	Manual data factory trigger.....	65
6.6	Database upgrade	68
6.7	Create read-only user in MARINFO database (using SQL Server Management Studio): ..	69
6.8	Add IP range to access MARINFO database (using Azure portal):	71

1. Introduction

1.1 Summary

The purpose of MARINFO is to provide ship related information and statistics for all ships assigned with an IMO number. The goal of this project is to make MARINFO cloud native and migrate all existing historical data to the cloud.

This document describes the infrastructure architecture of the Databricks Marinfo project data platform deployed for the European Maritime Safety Agency (EMSA). This encompasses the necessary Azure components, network connectivity, security, availability and scalability of the solution.

This document will cover the prerequisites, permissions and platform on which the solution is built, the used Azure components, how to access them and their relationships, the data flow through the components and the Azure governance model governing the entire implementation.

1.2 Project Description

The European Maritime Safety Agency (hereinafter called EMSA) provides various maritime information services. One of these maritime information services is MARINFO. The purpose of MARINFO is to provide ship related information and statistics for all ships assigned with an IMO number. MARINFO was developed by EMSA in 2007 with Talend for ETL, Perl scripts and an Oracle database.

The goal of this project is to make MARINFO cloud-native. This new solution will pick up data, validate and transform it, and make it available to end users and systems. All historical information needs to be transferred to the new solution as well.

Over time, there have been multiple versions of the MARINFO data. Before this project version 3 of the MARINFO data is used by EMSA. This will be the version that is used to implement the project.

Besides the MARINFO database, the following specific operational features are also in scope of the project:

MARINFO History: Register changes

A database schema that stores automatically all changes registered to the MARINFO data, to allow the business users to recreate the shipping universe at any given date in the past. For every data change (insert or update) an extra row is added to the table. Every row has a record start date and a record end date to indicate the validity of this information. In literature this modelling technique is commonly called a slowly changing dimension type 2. More information about this database is given in section 4.2.

Example: if a ship is 'recycled', a new record for that ship is created in the SHIP table of MARINFO HISTORY, with a timestamp to identify that from that moment on, the ship is no longer Active. The previously existing ship record is then closed (record end date) with the same timestamp, allowing the database users to assess the ships' situation in any given moment in time.

MARINFO Verification: Data uploading troubleshooting

Logging of problems during data uploading helps data contractors to identify issues and work on corrective measures. The verification information consists of loading history (LOAD_ID + timestamp), loading results (success/failure) and loading results causes.

In case of loading errors, the MARINFO Administrators and the Data-Provider contractor receive a complete report of the data loading failure.

MARINFO Views:

A number of informational views have been set up in order to harmonize and consolidate data from the data providers in a unified way:

- **MAR_REF:**

Aims to create reference tables for different dimensions like shiptypes, company's, ports, countries,... An additional layer refreshes these tables, new references are automatically added when they are available in the source data.

- **MAR_HISTORY:**

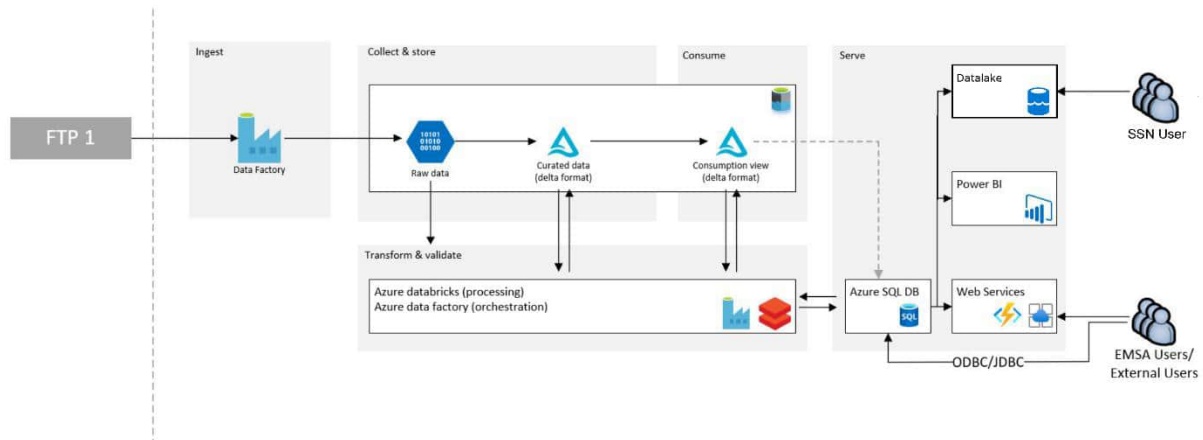
Aims to create history tables of certain characteristics of the underlying MARINFO data. The ship's flag, status, crew,.. are recorded with start and end date in these weekly updated history tables.

- **MAR_DONA:**

Key indicators that describe movement and other key statistics about a dynamic "active fleet universe".

2. Solution architecture

In this chapter the main components of the data platform architecture are discussed.



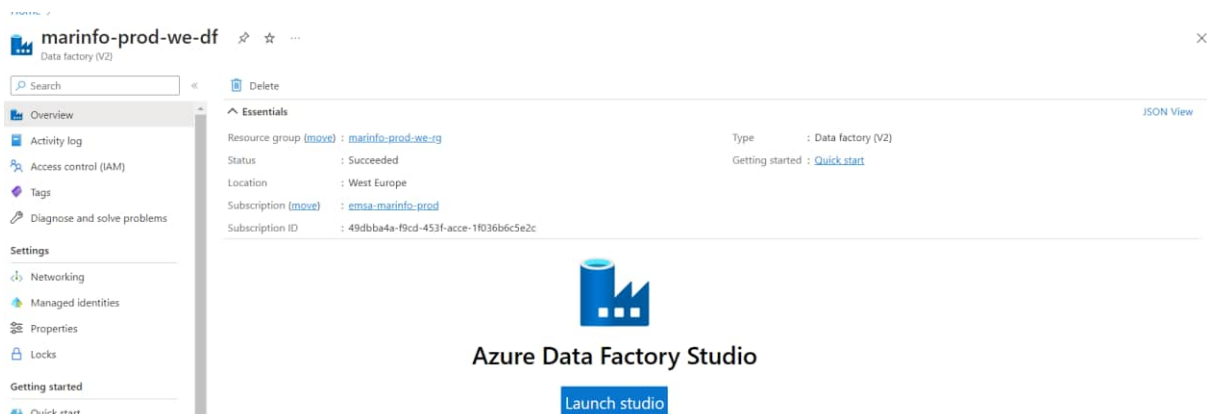
2.1 Data Factory

2.1.1 General information

Azure data factory will be the main orchestration component for this project. Azure Data Factory is Azure's cloud ETL service for scale-out serverless data integration and data transformation. It offers a code-free UI for intuitive authoring and single-pane-of-glass monitoring and management.

The relevant data factory marinfo-prod-we-df can be found by clicking on the link below, or by navigating in the azure portal as seen in below picture.

<https://adf.azure.com/en/home?factory=%2Fsubscriptions%2F49dbba4a-f9cd-453f-acce-1f036b6c5e2c%2FresourceGroups%2Fmarinfo-prod-we-rg%2Fproviders%2FMicrosoft.DataFactory%2Ffactories%2Fmarinfo-prod-we-df>



Since data factory will be the orchestration component, the build-in alerting mechanism is used to have automated processes to monitor the data flows.

Each time a pipeline in data factory is triggered, its status will be visible on the monitoring tab in the data factory UI. This tab can be consulted to check the status of a specific run and to see more details in case of an error.

Pipeline name	Run start	Run end	Duration	Triggered by	Status	Parameters	Annotations	Error
LoadData	9/14/20, 2:15:03 PM	---	00:04:25	5b59efc2-6025-43e0-ab...	In progress			
ScheduleEventHub	9/14/20, 2:15:01 PM	---	00:04:27	06-24_Every_Fifteen_Min	In progress			
LoadData	9/14/20, 2:00:05 PM	9/14/20, 2:00:30 PM	00:00:25	ba00c3ac-e191-4779-99...	Failed			
LoadData	9/14/20, 2:00:00 PM	9/14/20, 2:05:26 PM	00:05:25	6b557320-7270-4a4d-ab...	Succeeded			
LoadData	9/14/20, 2:00:00 PM	9/14/20, 2:00:33 PM	00:00:34	47f6def9-982b-4bd2-911...	Succeeded			
ScheduleEventHub	9/14/20, 2:00:00 PM	9/14/20, 2:08:31 PM	00:08:30	06-24_Every_Fifteen_Min	Failed			
ScheduleFailedExport	9/14/20, 1:59:59 PM	9/14/20, 2:08:33 PM	00:08:33	Hourly-BusinessHours	Succeeded			
ScheduleFailedStart	9/14/20, 1:59:59 PM	9/14/20, 2:05:26 PM	00:05:26	Hourly-BusinessHours	Succeeded			
LoadData	9/14/20, 1:45:03 PM	9/14/20, 1:49:49 PM	00:04:46	06434973-6617-47b4-a8...	Succeeded			
ScheduleEventHub	9/14/20, 1:45:02 PM	9/14/20, 1:49:50 PM	00:04:48	06-24_Every_Fifteen_Min	Succeeded			
LoadData	9/14/20, 1:30:03 PM	9/14/20, 1:34:54 PM	00:04:51	48f62764-2885-4d6b-bc...	Succeeded			
ScheduleEventHub	9/14/20, 1:30:00 PM	9/14/20, 1:34:53 PM	00:04:53	06-24_Every_Fifteen_Min	Succeeded			

The monitor tab can also be found by clicking this link below:

<https://adf.azure.com/en/monitoring/pipelinerruns?factory=%2Fsubscriptions%2F49dbba4a-f9cd-453f-acce-1f036b6c5e2c%2FresourceGroups%2Fmarinfo-prod-we-rq%2Fproviders%2FMicrosoft.DataFactory%2Ffactories%2Fmarinfo-prod-we-df>

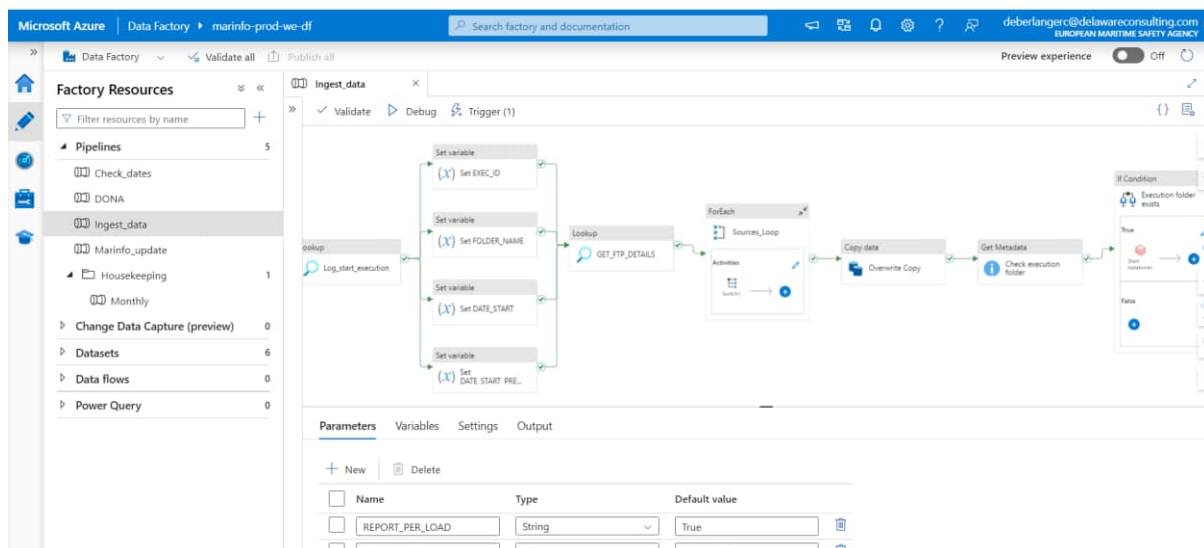
It is possible to setup automatic alerts based on the results present in the monitoring tab. Alerts can be automatically sent if a pipeline in data factory failed to execute (failed pipeline) or if the trigger to start the pipeline did not start (failed trigger). There are different mechanisms to be alerted and these can be configured for each alert. These are easily configurable alerts within Azure pipelines are general alerts that can for example notify when a pipeline didn't finish properly. So, there is less customization available here then for example with the ingest_data alerts, but can on the other hand send a generic alert for all other pipelines. Currently, a custom email is sent each time the ingest_data pipeline runs.

Creating alerts will ensure 24/7 monitoring of your data integration projects and make sure that you are notified of issues.

The mails from data factory are basic e-mails, simply stating a failure or success of a pipeline. Since reporting of failure requires more advanced e-mails to be sent for EMSA, a logic app will be used. The logic app is explained in the next section.

2.1.2 Implementations

Pipelines:



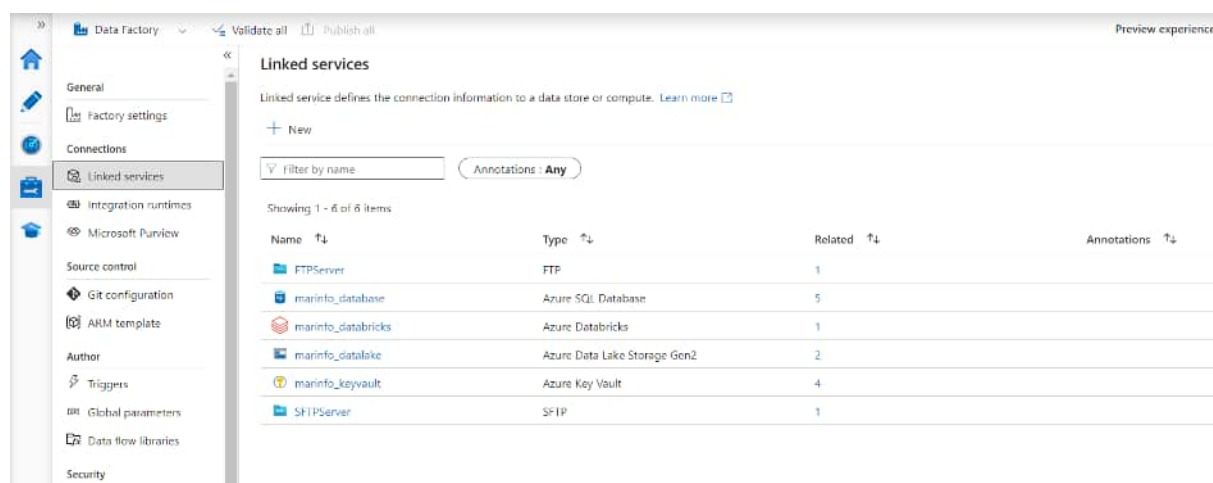
For different processes we have different pipelines setup. The main pipeline that ingests data from IHS is shown in the picture above. Each cell represents an action that is triggered in sequence. These tasks include a copy data tasks which moves data from one place to another, a databricks trigger that will process the data and store it into SQL or creates the report, a logic app trigger that will prepare the automated mail... Similarly the DONA and Marinfo_Update pipeline are there to orchestrate when certain SQL views, stored procedures need to be refreshed/materialized.

Triggers:

The screenshot shows the 'Triggers' management page in the Microsoft Azure Data Factory interface. The page displays a list of triggers: DONA, MarinfoWeekly, Monthly, and Weekly, all of which are Schedule triggers and are in a 'Started' status. The 'Edit trigger' panel on the right shows the configuration for the 'Weekly' trigger, including its name, description, type (ScheduleTrigger), start date, time zone, recurrence (Every 1 week), and advanced recurrence options (Run on these days: Sun, Mon, Tue, Wed, Thu, Fri, Sat).

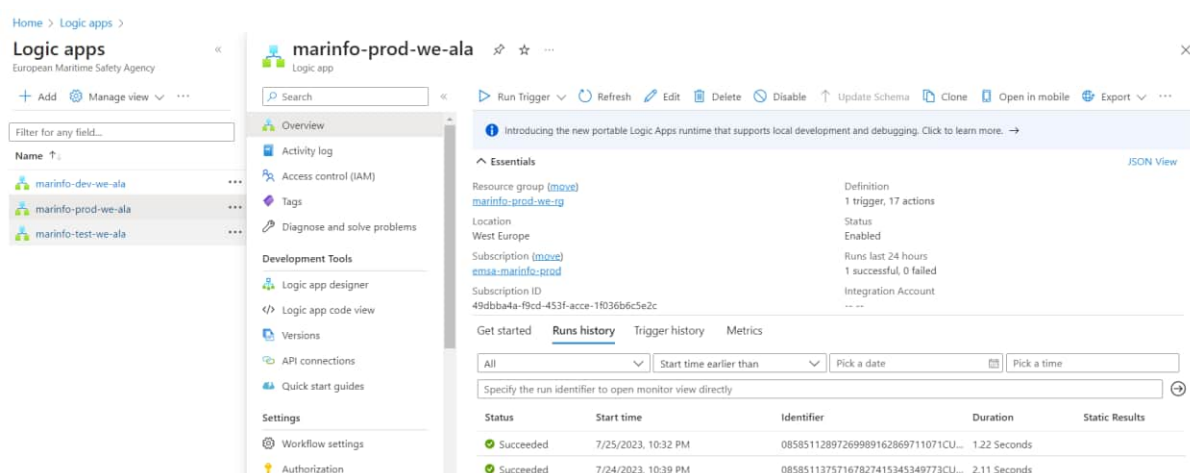
In the manage tab, we have the option to modify our triggers. These triggers will be linked to our pipelines and determine when/how often they should be triggered.

Linked Services:



In the manage tab we also find our linked services. These are predefined services that Data Factory needs access to in order to complete the pipeline.

2.2 Logic App

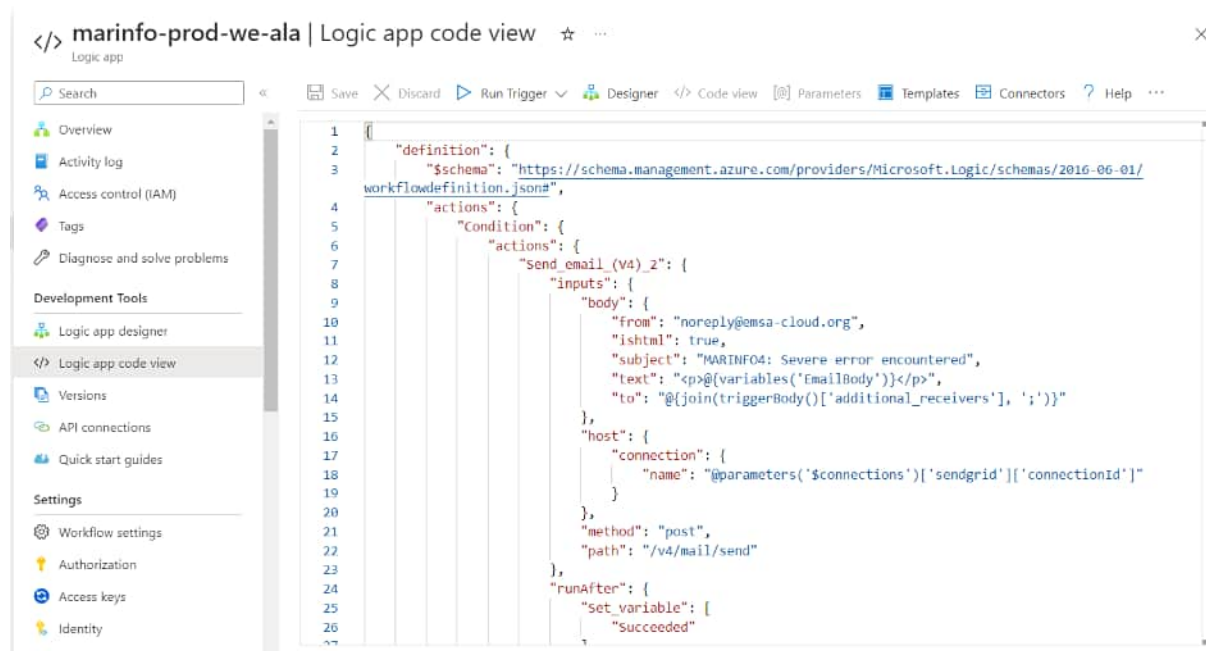


Azure Logic Apps is a cloud service that helps you schedule, automate, and orchestrate tasks, business processes, and workflows when you need to integrate apps, data, systems, and services across enterprises or organizations. Logic Apps simplifies how you design and build scalable solutions for app integration, data integration, system integration, enterprise application integration (EAI), and business-to-business (B2B) communication. For this project, a logic app will be used to send e-mail reports with office365 when events happen during the process flow. The content of the email can be customized for EMSA.

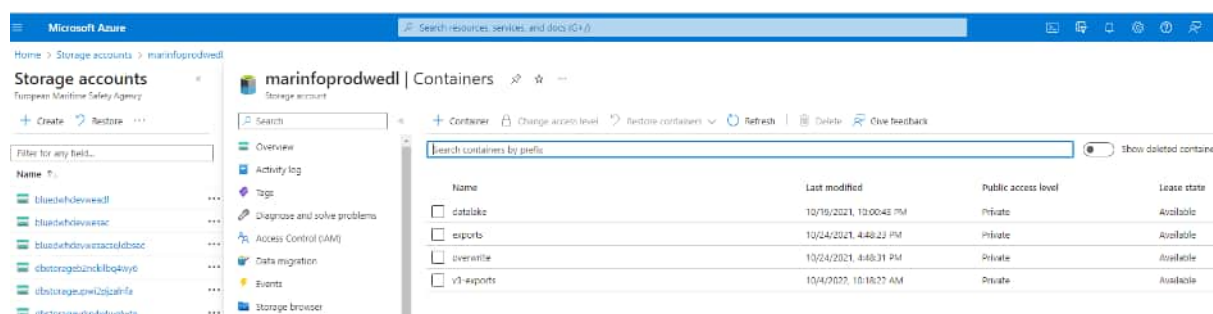
The relevant logic app for the production environment can be accessed/configured by following the link below:

<https://portal.azure.com/#/@emsaeuropa.onmicrosoft.com/resource/subscriptions/49dbba4a-f9cd-453f-acce-1f036b6c5e2c/resourceGroups/marinfo-prod-we-rg/providers/Microsoft.Logic/workflows/marinfo-prod-we-ala/logicApp>

The code of the logic email app can be found by accessing the logic app code view tab on the left.



2.3 Azure Data Lake



Azure Data Lake Storage Gen2 is a set of capabilities dedicated to big data analytics, built on Azure Blob storage. Data Lake Storage Gen2 is the result of converging the capabilities of two existing storage services, Azure Blob storage and Azure Data Lake Storage Gen1. Features from Azure Data Lake Storage Gen1, such as file system semantics, directory, and file level security and scale are combined with low-cost, tiered storage, high availability/disaster recovery capabilities from Azure Blob storage.

The data lake is where all raw data is stored. The Azure data factory will copy all new data from the FTP server to the data lake where it can be picked up by databricks to process the data. Besides raw data, intermediate results will be stored in parquet files (delta lake) which are also stored on the data lake.

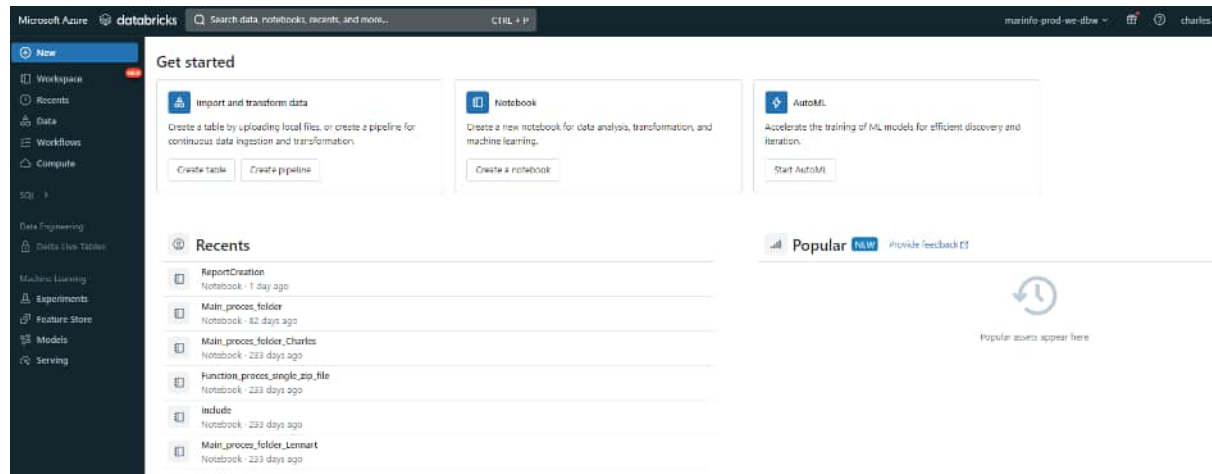
The delta lake, which was previously mentioned, is an open source storage layer that brings reliability to data lakes. Delta Lake provides ACID transactions, scalable metadata handling, and unifies streaming and batch data processing. Delta Lake runs on top of your existing data lake and is fully compatible with Apache Spark APIs.

Delta Lake on Databricks allows you to configure Delta Lake based on your workload patterns. Databricks also includes Delta Engine, which provides optimized layouts and indexes for fast interactive queries.

The relevant storage account for the production environment can be accessed by clicking the link below.

<https://portal.azure.com/#/@emsaeuropa.onmicrosoft.com/resource/subscriptions/49dbba4a-f9cd-453f-acce-1f036b6c5e2c/resourceGroups/marinfo-prod-we-rq/providers/Microsoft.Storage/storageAccounts/marinfoprodwedl/containersList>

2.4 Databricks



Azure Databricks is an Apache Spark-based analytics platform optimized for the Microsoft Azure cloud services platform. Designed with the founders of Apache Spark, Databricks is integrated with Azure to provide one-click setup, streamlined workflows, and an interactive workspace that enables collaboration between data scientists, data engineers, and business analysts. Databricks will be used for transformation and validation of the data and for transferring the data from CSV files to the relational database.

For the purpose of ingesting data into the SQL server and initiating the checks that need to be performed, as well as automatic report creation, databricks notebooks written in the python programming language are used.

datalake_folder

execution_id

MARINFO4

250

96

log_end_load(ctl_id, 'FAIL')

97

98

print('number successful loads:', len(succeeded_loads))

99

print('successful loads:', succeeded_loads)

100

print()

101

print('number of failed loads:', len(failed_loads))

102

print('failed loads:', failed_loads)

103

104

"""Once something goes wrong before a the process order gets determined we log an error and stop the processing of the script.

105

""" We throw an error to notify DataFactory that the load has failed.

106

except Exception as e:

107

except FailureInDatabase as e:

108

log_error(execution_id, 'NULL', 'NULL', e)

109

raise Exception('The load has failed')

110

111

""" (H) Spark Jobs

112

113

loading file: tblNullMaterialCodes

114

successfully written to STAGING.LOT2A_CODE_NULL_MATERIAL

115

Index: 14

116

loading file: tblFlagHistory

117

successfully written to STAGING.LOT2A_HISTORY_FLAG

118

Index: 3

119

loading file: tblCallSignedHMSHistory

120

successfully written to STAGING.LOT2A_HISTORY_CALL_SIGN_HMSI

121

Index: 1

122

loading file: tblAdditionalShipsData

123

successfully written to STAGING.LOT2A_ADDITIONAL_SHIPS_DATA

124

EXEC [MAR_VERT4]. [SPR_ScreeningGeneric] @EXEC_ID=250, @LOAD_ID=564, @LOAD_DETAIL_ID=12241, @FILES_LOT_ID=5638.0, @TABLE_NAME='LOT2A_CODE_FLAG', @SCHEMA_NAME='MARINFO4', @debug=0

125

EXEC [MAR_VERT4]. [SPR_ScreeningGeneric] @EXEC_ID=250, @LOAD_ID=564, @LOAD_DETAIL_ID=12241, @FILES_LOT_ID=5638.0, @TABLE_NAME='LOT2A_CODE_CLASS', @SCHEMA_NAME='MARINFO4', @debug=0

126

EXEC [MAR_VERT4]. [SPR_ScreeningGeneric] @EXEC_ID=250, @LOAD_ID=564, @LOAD_DETAIL_ID=12241, @FILES_LOT_ID=5644.0, @TABLE_NAME='LOT2A_CODE_NULL', @SCHEMA_NAME='MARINFO4', @debug=0

127

EXEC [MAR_VERT4]. [SPR_ScreeningGeneric] @EXEC_ID=250, @LOAD_ID=564, @LOAD_DETAIL_ID=12241, @FILES_LOT_ID=5641.0, @TABLE_NAME='LOT2A_CODE_SHEFTYPE', @SCHEMA_NAME='MARINFO4', @debug=0

128

EXEC [MAR_VERT4]. [SPR_ScreeningGeneric] @EXEC_ID=250, @LOAD_ID=564, @LOAD_DETAIL_ID=12241, @FILES_LOT_ID=5654.0, @TABLE_NAME='LOT2A_CODE_STATUS', @SCHEMA_NAME='MARINFO4', @debug=0

129

EXEC [MAR_VERT4]. [SPR_ScreeningGeneric] @EXEC_ID=250, @LOAD_ID=564, @LOAD_DETAIL_ID=12241, @FILES_LOT_ID=5647.0, @TABLE_NAME='LOT2A_CODE_PANDIT', @SCHEMA_NAME='MARINFO4', @debug=0

130

EXEC [MAR_VERT4]. [SPR_ScreeningGeneric] @EXEC_ID=250, @LOAD_ID=564, @LOAD_DETAIL_ID=12241, @FILES_LOT_ID=5632.0, @TABLE_NAME='LOT2A_COMPANY_ALL', @SCHEMA_NAME='MARINFO4', @debug=0

131

EXEC [MAR_VERT4]. [SPR_ScreeningGeneric] @EXEC_ID=250, @LOAD_ID=564, @LOAD_DETAIL_ID=12241, @FILES_LOT_ID=5634.0, @TABLE_NAME='LOT2A_COMPANY', @SCHEMA_NAME='MARINFO4', @debug=0

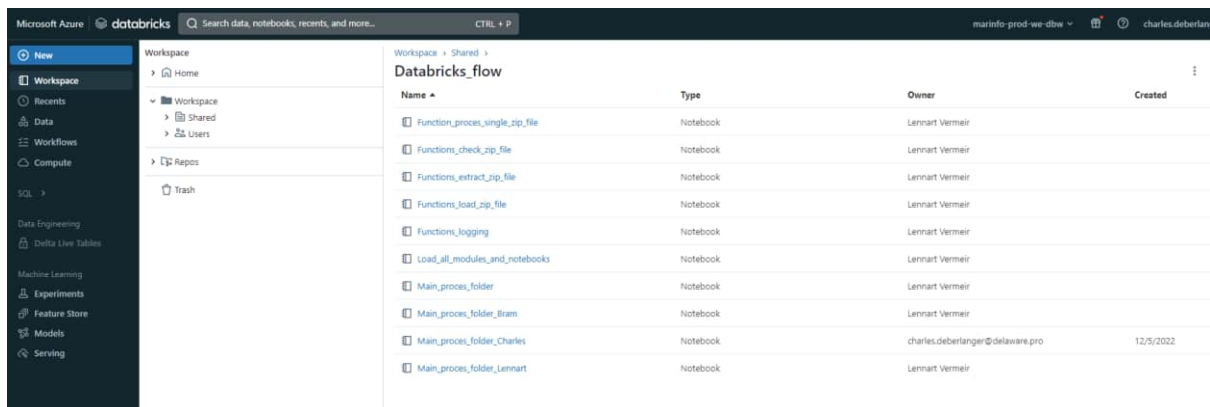
132

EXEC [MAR_VERT4]. [SPR_ScreeningGeneric] @EXEC_ID=250, @LOAD_ID=564, @LOAD_DETAIL_ID=12241, @FILES_LOT_ID=5643.0, @TABLE_NAME='LOT2A_HULL_SHAPE', @SCHEMA_NAME='MARINFO4', @debug=0

133

EXEC [MAR_VERT4]. [SPR_ScreeningGeneric] @EXEC_ID=250, @LOAD_ID=564, @LOAD_DETAIL_ID=12241, @FILES_LOT_ID=5677.0, @TABLE_NAME='LOT2A_HULLER_DETAILS', @SCHEMA_NAME='MARINFO4', @debug=0

These notebooks can be found in the Workspace tab on the left:



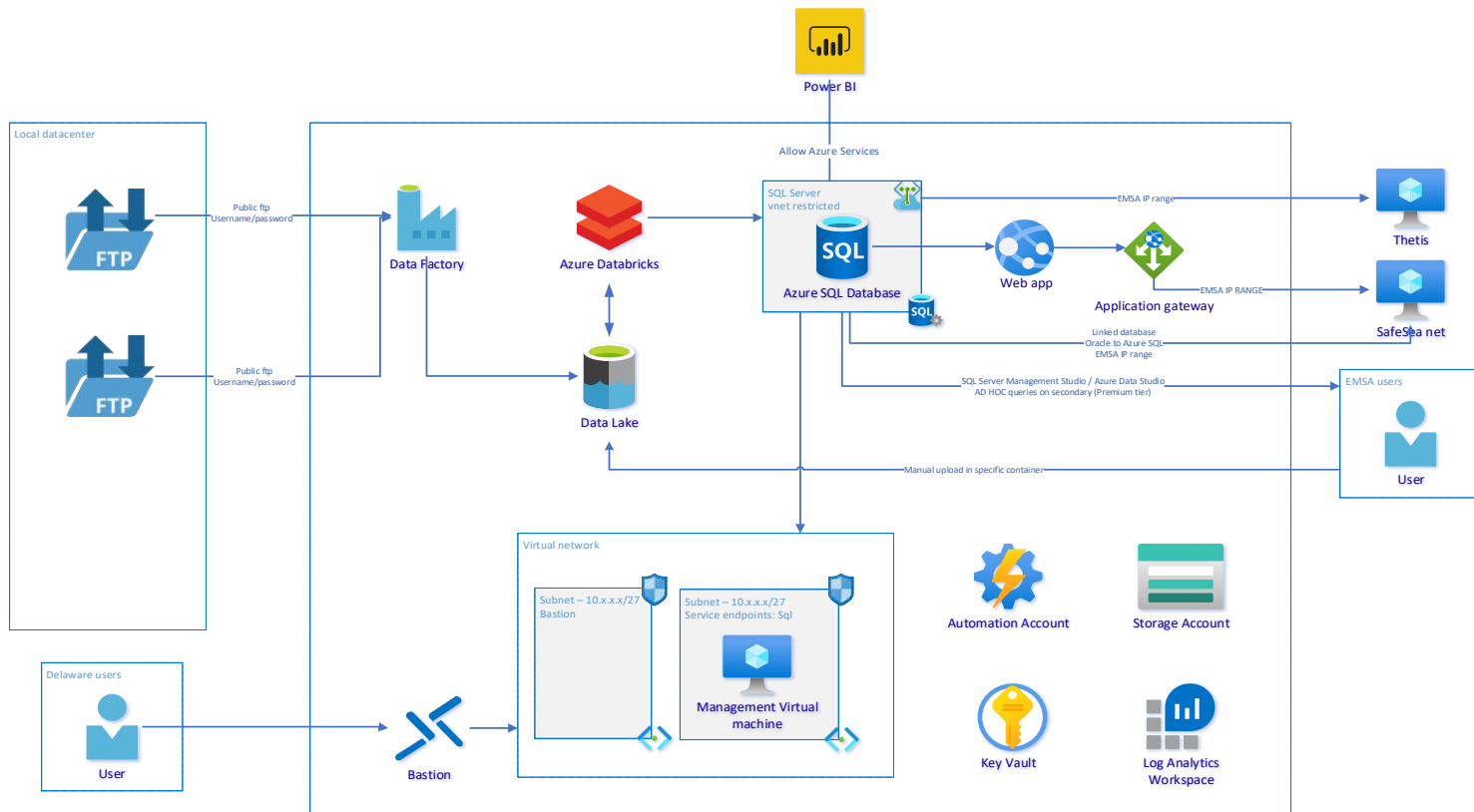
2.5 Azure SQL Database

After the input files arrive in the Azure Data Lake they are picked up by Azure Databricks where the files undergo the necessary processing steps before writing the end result to both the Azure Data lake and an Azure SQL database.

From then onwards the data is available in the Azure SQL database for all direct data consumers described below as well as the web services.

Azure SQL Database is the intelligent, scalable, cloud database service that provides the broadest SQL Server engine compatibility.

In the MARINFO database, we have constructed tables where the MARINFO data is stored, views that generate transformations based on the original IHS data and stored procedures that help with ingesting data, performing checks or materializing the views.



3. Azure prerequisites, permissions and platform

The Marinfo project is built on Microsoft Azure and will make use of the existing Azure infrastructure to integrate with the data sources that currently reside on-premises. Specific information on the used Azure DevOps project, the Azure Tenant, subscription, resource groups and provided networks are described in this section.

3.1 Azure DevOps

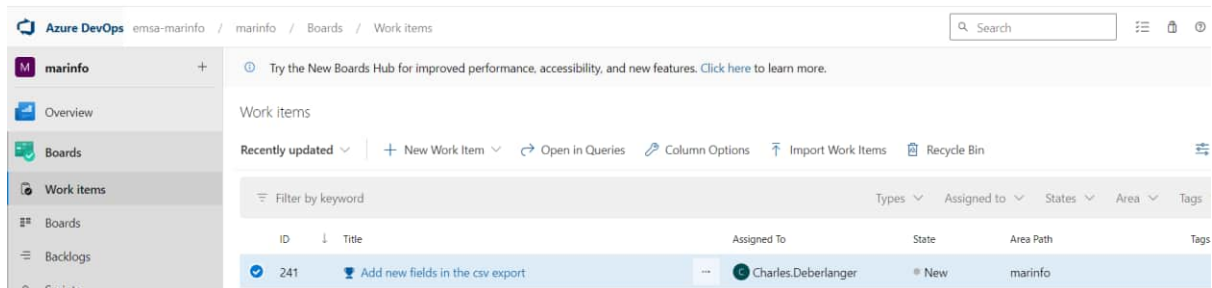
An Azure DevOps project is used to store both the Azure platform as well as the application code. Keeping the complete application in source control allows for quick recovery from failure, modular expansion of the solution and guarantees that the different environments are configured in exactly the same way. The Azure DevOps project for the Marinfo project can be found here:

<https://dev.azure.com/emsa-marinfo/marinfo>

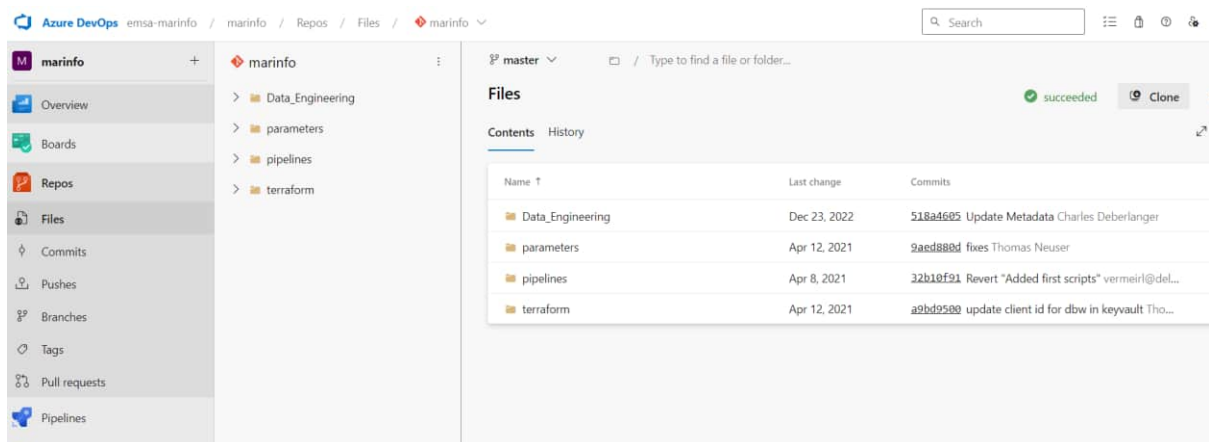
Property	Value
Project name	MARINFO
Description	This Azure DevOps project hosts the code and build/release pipelines for the EMSA Marinfo project
Visibility	Private
Version Control	Git
Work Item Process	template

Table 1: Azure DevOps project properties

The Boards tab shows open tickets for the MARINFO development.



The Repos tab contains all the GIT managed code for each of the Azure components:



3.1.1 Azure Devops project site

The Azure Devops can be accessed by clicking the link below:


<https://dev.azure.com/emsa-marinfo/>

3.1.2 Extensions

The following free Azure Devops extensions are added to the Azure Devops project.

Module	Publisher	Url
Terraform	Microsoft DevLabs	https://marketplace.visualstudio.com/items?itemName=ms-devlabs.custom-terraform-tasks

3.2 Azure Subscription

Subscription name	Subscription ID
emsa-marinfo-dev	
emsa-marinfo-preprod	
emsa-marinfo-prod	
emsa-marinfo-test	

3.3 Resource Groups

The data platform solution in Azure is divided in multiple resource groups depending on the chosen number of tiers and regions. In each resource group, some resources necessary for management of the environment and BI development are deployed. The resource groups follow a generic naming convention.

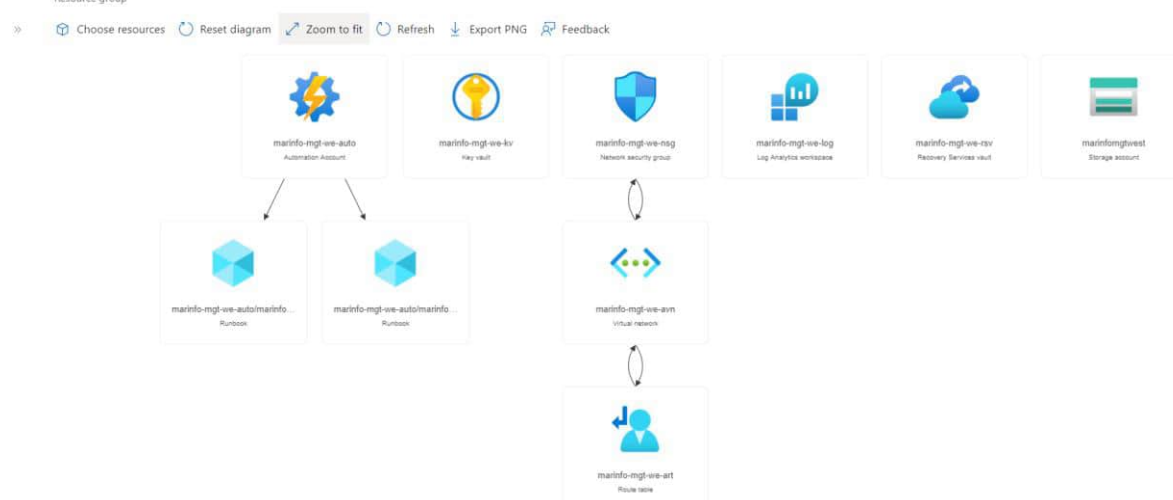
NAME	Subscription	Location
marinfo-test-we-rg	emsa-marinfo-test	West Europe
marinfo-test-we-dbw-arg	emsa-marinfo-test	West Europe
marinfo-prod-we-rg	emsa-marinfo-prod	West Europe
marinfo-prod-we-dbw-arg	emsa-marinfo-prod	West Europe
marinfo-mgt-we-rg	emsa-marinfo-prod	West Europe
marinfo-dev-we-rg	emsa-marinfo-dev	West Europe
marinfo-dev-we-dbw-arg	emsa-marinfo-dev	West Europe

marinfo-dev-we-rg, marinfo-test-we-rg and marinfo-prod-we-rg are the respectively dev, test and prod environments where all day-to-day MARINFO activity takes place. Resources such as the Databricks service, Azure Data Factory, SQL server, Azure Datalake,... can be found here.

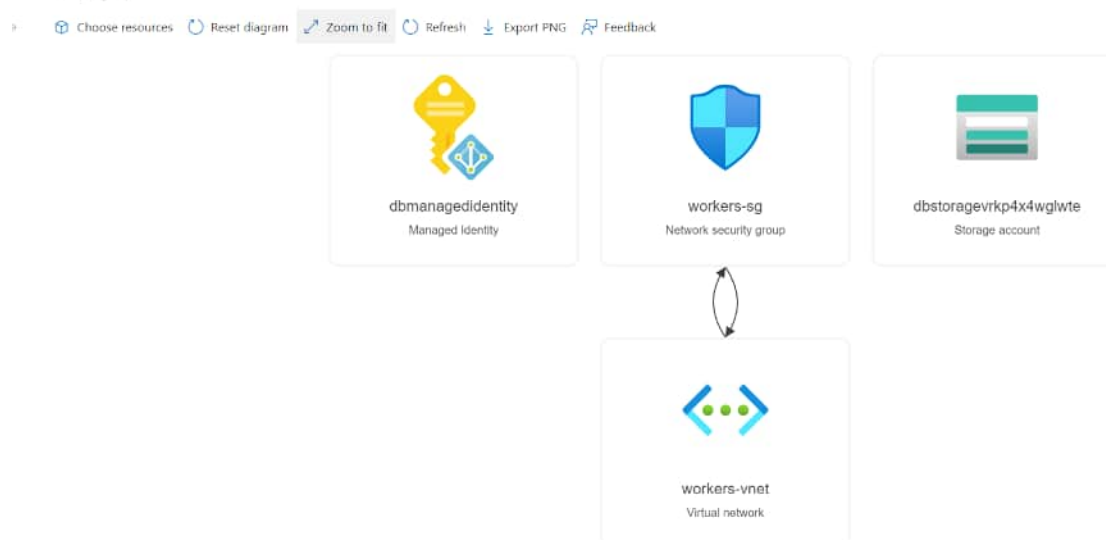


The remaining resource groups `marinfo-dev-we-dbw-arg`, `marinfo-test-we-dbw-arg` and `marinfo-mgt-we-rg` contain little resources and have mainly infra, back-up and management related purposes.

marinfo-mgt-we-rg | Resource visualizer



marinfo-prod-we-dbw-arg | Resource visualizer



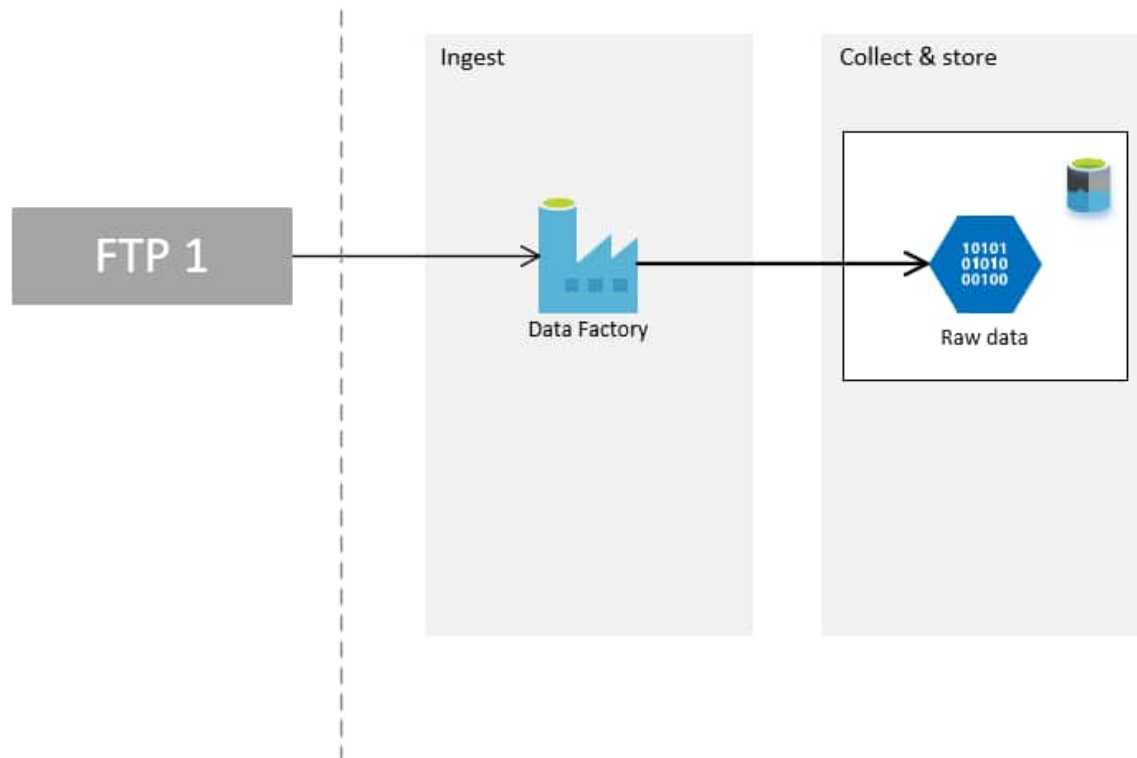
4. Functional Design

In the following sections, each step in the process is explained in more detail:

4.1 MARINFO data

4.1.1 Data Ingestion

Ingestion of data is done by the Data Factory. Based on the trigger schedule, data is pulled from the FTP servers and stored on an azure data lake. The schedule is configurable in data factory. All raw data is stored, which is the starting point for databricks to begin transformations on the data. No retention period is specified, hence all data will be stored indefinitely. It is however possible to configure the retention period of the data.



The raw data consists of ZIP files which are uploaded each day by external contractors. At the moment only one FTP server is used as shown in above figure. It should however be possible to add new FTP or SFTP servers in the future that are simultaneously checked for new data. The current FTP server is reachable from the public internet.

The filename of the ZIP files contains the date on which it was uploaded on the FTP server (lotX_YYYYMMDD.zip where X reflects the destination lot). This makes it possible to only process the most recent files. Each file that has no record in the database or that has a record with status *failed*, has not been successfully processed and can be handled. Note that files should be processed in order, oldest files first based on the date in the filename. In case of a failure, the process is not blocked and more recent files will be processed nevertheless. The order in which files should be processed from the same day is determined by the dependencies between the different tables in the MARINFO database. Each file that is processed is added to the MARINFO_VERIFICATION schema together with its processing status. Note that failures during the ingestion step consist only of failed copy tasks (for whatever reason possible) or systems that are not available/reachable, because no data validation is done yet. Hence, an e-mail is sent only to EMSA users in case something happened during this step.

Note that no data is removed from the FTP server by EMSA users. The external contractor may however delete files from the FTP server.

Step-by-step approach

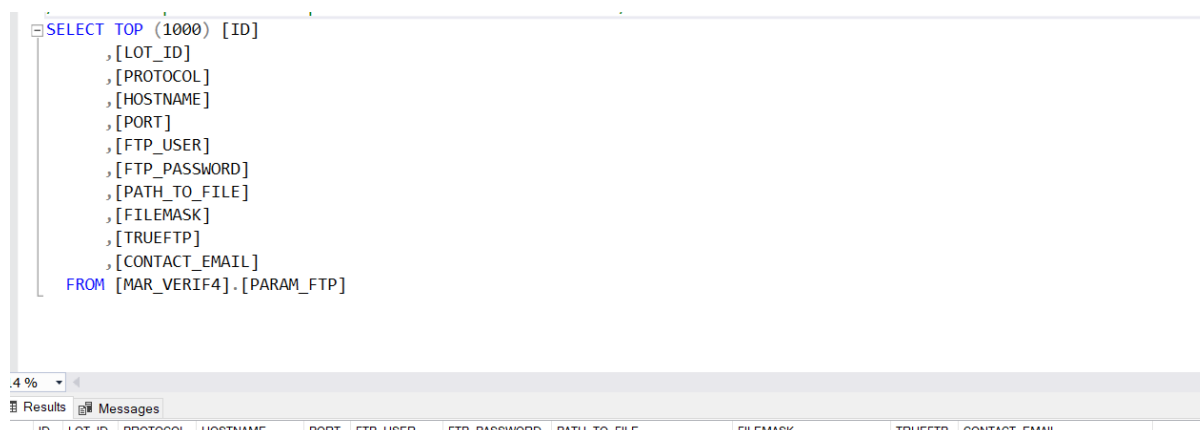
1. Connect to the FTP server
2. List the files on the remote directory which comply the expected pattern for the lot
3. Identify the files which are more recent than the latest successfully processed load for the specified lot.
4. Copy the identified files to the raw data area in the data lake.

Failure handling

In case of a failure when processing the data in the ZIP file, the full zip is NOT processed. Since data factory is used for orchestration, the build-in mechanisms for alerting are used. A mail will be sent to EMSA users stating the ingestion failed. Note that failures due to incorrect data or data types mismatches etc. are explained in more detail later.

FTP Configurations

The configurations on how to access the FTP server are stored into the SQL database. The information can be found / adjusted in the following table: MAR_VERIF4.PARAM_FTP



```

SELECT TOP (1000) [ID]
      ,[LOT_ID]
      ,[PROTOCOL]
      ,[HOSTNAME]
      ,[PORT]
      ,[FTP_USER]
      ,[FTP_PASSWORD]
      ,[PATH_TO_FILE]
      ,[FILEMASK]
      ,[TRUEFTP]
      ,[CONTACT_EMAIL]
FROM [MAR_VERIF4].[PARAM_FTP]
  
```

ID	LOT_ID	PROTOCOL	HOSTNAME	PORT	FTP_USER	FTP_PASSWORD	PATH_TO_FILE	FILEMASK	TRUEFTP	CONTACT_EMAIL
----	--------	----------	----------	------	----------	--------------	--------------	----------	---------	---------------

At the time of writing, the data can be found on the hostname: mft.ihsmarkit.com

Raw Data

The raw data can then be found on the marinfoprodwedl storage account, in the datalake container and opening the MARINFO4 folder. Here, they are stored in zip files.

Storage accounts > marinfoprodwedl | Containers >

atalake

Container

Upload Add Directory Refresh Rename Delete Change tier Acquire lease Break lease Give feedback

Authentication method: Access key (Switch to Azure AD User Account)
Location: datalake / MARINFO4 / executionid=100

Search blobs by prefix (case-sensitive) Show

Name	Modified	Access tier	Archive status	Blob type	Size
[-]					
lot1a_berthcallings_upd_20220228.zip	4/5/2022, 10:01:23 PM	Hot (Inferred)		Block blob	7.56 MB
lot1a_berthcallings_upd_20220307.zip	4/5/2022, 10:01:27 PM	Hot (Inferred)		Block blob	7.67 MB
lot1a_berthcallings_upd_20220314.zip	4/5/2022, 10:01:18 PM	Hot (Inferred)		Block blob	7.63 MB
lot1a_berthcallings_upd_20220321.zip	4/5/2022, 10:01:32 PM	Hot (Inferred)		Block blob	7.46 MB

4.1.2 Data validation

The collect & store step takes the ZIP files from the raw data area and processes them. The detailed architecture of this step is shown in below figure.

tblEvents.all	6.107.802	698.560	ALL File	27/04/2020 17:16	7AF3BE64
lot1_casualtydata.all	10.757.825	2.914.554	ALL File	27/04/2020 17:15	BE56AB93
Checksum.txt	140	125	Text Document	27/04/2020 17:16	1640AFB6

The ZIP files from the data ingestion step will be extracted and each CSV file that was contained in the ZIP file will be processed. The CSV file contains the file type in its filename. There are 4 possible file types (all, upd, nul and del). They are explained in more detail in the following subsections. The figure above shows an example of the contents of a zip file. It consists of multiple CSV files containing the data for the operations. Besides, a checksum file *Checksum.txt* is present to verify the authenticity of the data.

The first step in the collect & store step is to verify the authenticity of the data by using the checksum file. This file should contain a valid header (FILENAME|NBRECORDS|MD5) and the information in the checksum file should be correct (Number of records for each file + MD5 hash of each file + no other files in the ZIP other than the ones present in the checksum file). Below example shows the content of a *Checksum.txt* file.

```
FILENAME|NBRECORDS|MD5
lot1_casualtydata.all|19665|EC6BDA147C99778719CAAF95748D1EF0
tblEvents.all|31526|4D0E97ED3CE8665ECF2800851254DE3F
```

The next step is to validate the data against the correct data types. The final step is to validate all primary and foreign keys. The primary key should be unique and the foreign key should be already present in the data. Before storing the data, circular dependencies should be checked.

In case a failure occurs due to wrong data types or incorrect keys/dependencies, a mail is sent to the users of EMSA and the external contractors. The mail is not processed automatically at both sides, hence the content of the report should not be equal to the present situation as long as the errors are pinpointed so the contractors know where the error occurred and why. The contractors will upload the corrected data to the FTP servers and the corrected data will be processed again. The corrected data should not have the same filename as the file in which the error was found. If a file fails, next run, more recent files will be processed. An example of an e-mail report is shown below. Depending on the lot, the mail to the external contractor will have different recipients.

MARINFO4: COMPLETED - Load of zipfile lot1a_berthcallings_upd_20230605.zip



MARINFO4: FAIL - Load of zipfile lot2a_shipdata_20230530.all.zip



Besides the reporting mail, the loading process should also append a new line to the MARINFO Verification database with the timestamp when the loading process occurs, a status of the process (success/failure) and some extra information about the exact error and why the error happened. This step is explained in more detail later.

Step-by-step approach

The following steps are orchestrated within the databricks 'Main_Process' script.

<https://adb-1166721739537560.0.azuredatabricks.net/?o=1166721739537560#notebook/2869800246450272/commmand/2869800246450273>

The checks performed are either validation logic executed in databricks itself or validation logic directly in the SQL database.

1. Validate the checksum file
 - a. Headers equals FILENAME|NBRECORDS|MD5
 - b. Number of records in checksum file should be exactly the number of files in the zip file
2. Validate file types (one of all, upd, nul and del)
3. Validate information in the checksum file
 - a. Filenames should be correct based on files in the zip
 - b. Number of records for each file should be correct
 - c. MD5 hash for each file should be correct
4. Validate data types for each file
5. Load data
 - a. Disable circular dependencies/foreign keys
 - b. Validate header of file and primary key field
 - c. Enable circular dependencies/foreign keys and check

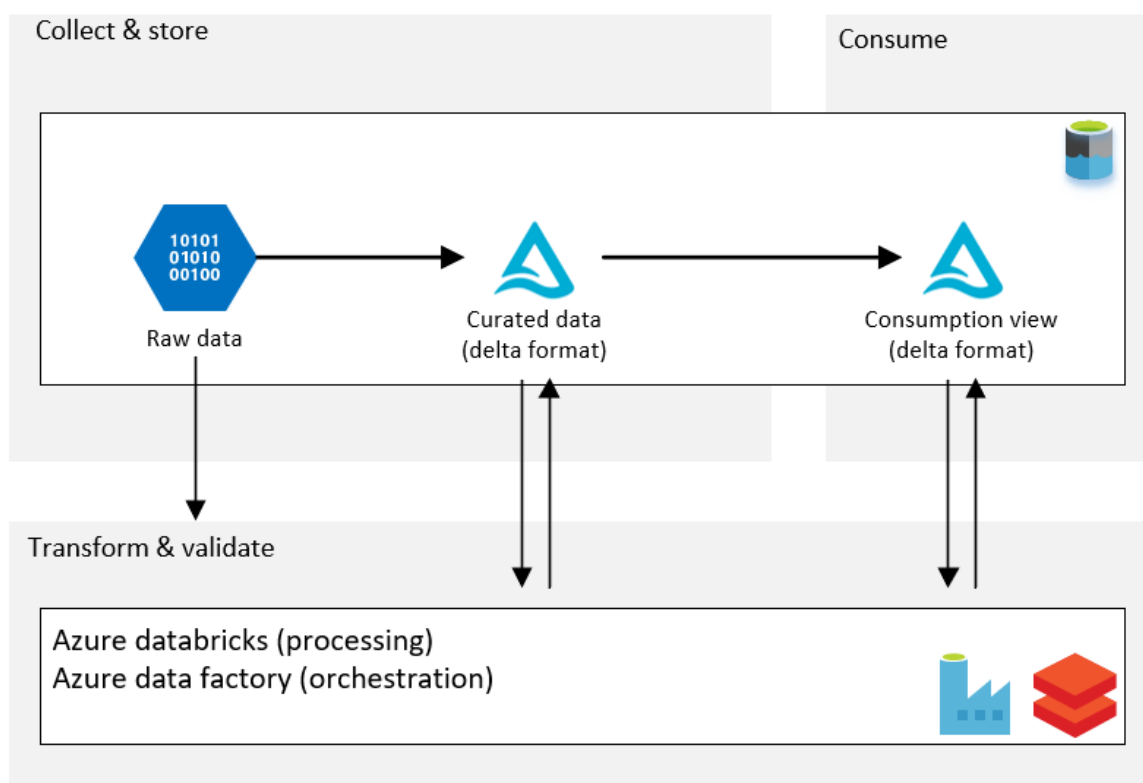
Failure Handling

In case of a failure during data validation, the following process flow is in place:

1. All changes done up until this step are reverted so the database is in the same state as it was before processing the file.
2. Error mail is sent to EMSA users and to the contact person of the contractor providing the data (can differ per lot).
3. In MARINFO_VERIFICATION, information about the failed file is added and the status is set to *failed*. The ZIP file always contains information about a lot.
4. One of both options is applicable:

<p>a. Contractor corrects the data and re-uploads it to the FTP server. Note that this is the full ZIP again as no data is processed upon a failure. The checksum file will also change when changing the data itself (This is a manual step at the contractor side)</p>	<p>b. EMSA users correct the data and upload it directly to the datalake in a separate container. A file uploaded by EMSA users will always have precedence over the contractor's file. Note that this is the full ZIP again as no data is processed upon a failure. The checksum file will also change when changing the data itself. Changes made to the data should be communicated to the contractor as they will have to change their data as well to keep database consistency when processing a .all file in the future. (This is a manual step at EMSA side)</p>
---	--

5. Next run, the corrected file will be picked-up again and be processed because the record of this file has a *failed* status in the MARINFO_VERIFICATION database. Note that if no correction file is available on the FTP server, but there are more recent files, the newer files are processed, nevertheless. This way, a failure will not block the overall process.



4.1.2.1 File types

Each file should be one of four defined types (*all*, *upd*, *del*, *nul*). The details of each type are described below.

- **All**

A file type of type *all* will contain *.all* in its filename. Files of this type will contain all data for the table. All data needs to be replaced by the data contained in the CSV file (delete all and insert). This file type is for example used for synchronization between the external contractor and the data in the database of EMSA.
- **Upd**

A file type of type *upd* will contain *.upd* in its filename. Files of this type contain records that need to be updated or inserted. An ID will be provided in the data in order to know exactly which record needs to be updated or, if the ID is not present, inserted.
- **Del**

A file type of type *del* will contain *.del* in its filename. Files of this type contain records that need to be deleted. This file type is only used occasionally. Note that a cascade delete is in place, meaning that all records to which a reference is made, should also be deleted from the tables. However, the deletes from the referenced tables should also be present in *.del* files for the referenced tables. If this is not the case, the failure mechanism should take over.
- **Nul**

A file type of type *nul* will contain *.nul* in its filename. Files of this type should not be processed.

Nothing should be done with these files. They are however present on the FTP server and should be taken into account when calculating the validity of the checksum file.

4.1.2.1 MARINFO Synchronization

It is possible that once a year, the full database may need to be compared to the data supplier contractor's database, to repair potential database errors/mistakes cumulated along the update loads. This is called synchronization.




The external contractor will send a "*.all" file to overwrite all current data in the database (to fix potential database errors/mistakes cumulated along the update loads). All existing data will be deleted and data in the "*.all" file will be inserted. Note that the history of data is only changed if a change/deletion was in place, otherwise the current version of the history of the specific record will remain.

Backups are automatically taken for the database. In case of a failure, it will be possible to roll-back a previous version of the database. At the moment, backups are stored on geo-redundant storage in Azure.

The geo-redundant storage can be found in the azure portal – storage account tab:

<https://portal.azure.com/#view/HubsExtension/BrowseResource/resourceType/Microsoft.Storage%2FStorageAccounts>

The geo-redundant storage accounts are as seen below:

<input type="checkbox"/> Name ↑↓	Type ↑↓	Kind ↑↓	Resource group ↑↓	Location ↑↓
<input type="checkbox"/>  dbstorageeb2nckilbq4wy6	Storage account	BlobStorage	marinfo-test-we-dbw-arg	West Europe
<input type="checkbox"/>  dbstorageupwi2pjzalnfa	Storage account	BlobStorage	marinfo-dev-we-dbw-arg	West Europe
<input type="checkbox"/>  dbstorageevrpk4x4wglwte	Storage account	BlobStorage	marinfo-prod-we-dbw-arg	West Europe

4.1.3 Data Consumption

It could be that there are specific use cases/needs in which transformed or aggregated data is queried regularly. In that case the data can be precalculated and stored in the required transformed format in the consumption layer. The consumption layer holds data that is derived from the curated data for specific (reusable) use cases. Since the cost in data lake storage is low, this 'extra' layer can even be used for 'small' transformations. For example, the data harmonisations that are considered in the technical specifications could be done in this layer with mapping tables. By applying this in the extra layer, corrective actions and recalculation based on the curated data are always possible if the harmonization rule changes or in case of a bug. At the moment, this layer will contain an exact copy of the curated data.

4.1.4 Serve

This part contains the components that can be used for consumption of the data. In the figure below, the architecture of the serving part together with adjacent parts (Consume & Transform and Validate) is depicted. This layer consists of several components which are described in more detail in the below subsections.

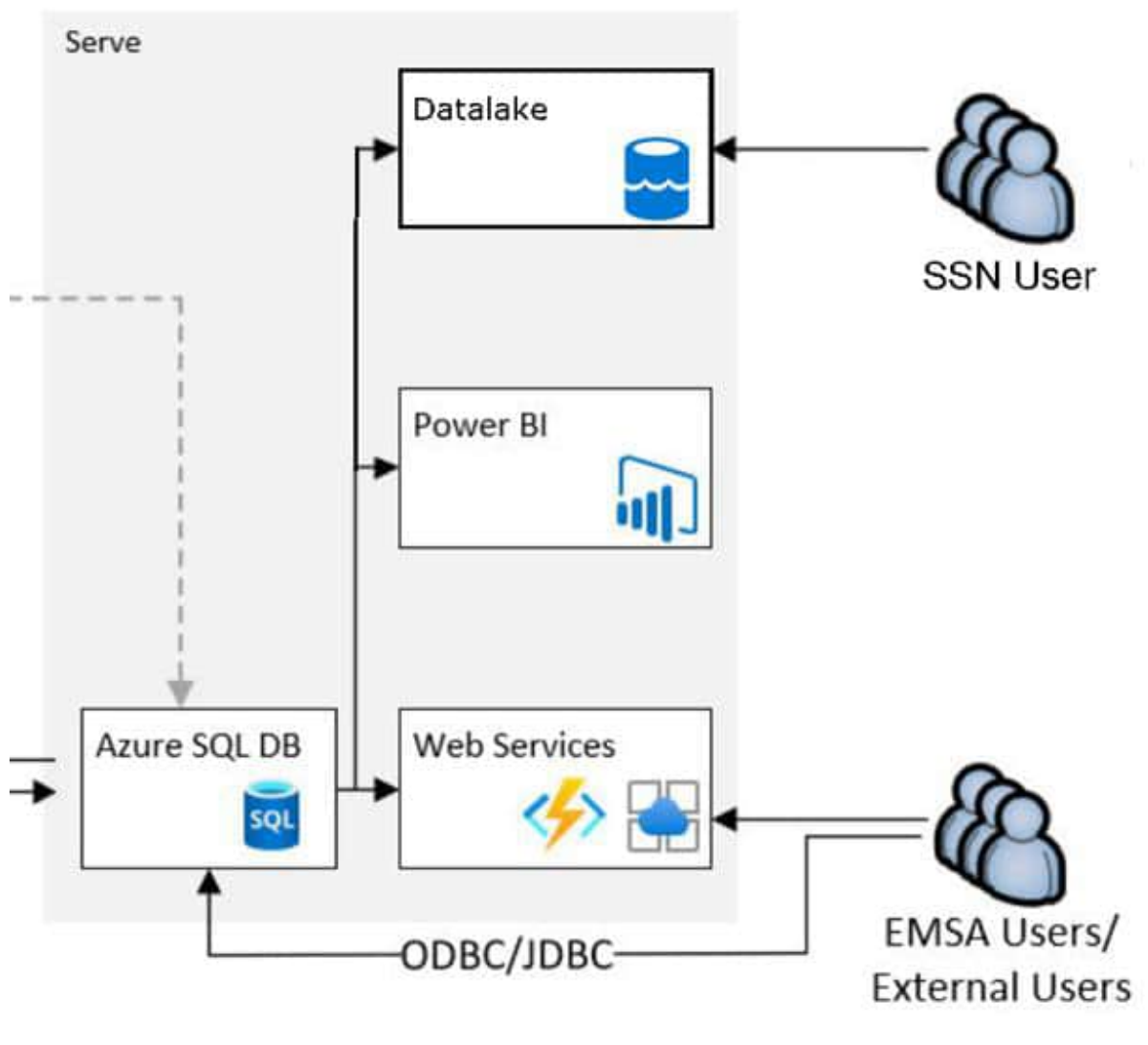
4.1.4.1 Azure SQL DB

The data tables are made available for consumption in the SQL database. The data is copied over from either the “Curated Data” or the “Curated” data lake layer. This is done via Databricks. Next to serving the data the SQL DB will also fulfil the next functions: keeping functional logs (loading history, loading results, loading result causes, # records,...). This database can be queried by end-users through ODBC/JDBC connection, web services or Power BI.

Please refer to the EMSA MARINFO DATA LIBRARY document for more in-depth information regarding SQL tables, SQL views and stored procedures.

4.1.4.2 ODBC & JDBC

Users that need access to the MARINFO database use JDBC or ODBC connections at the moment. By using the connections, users are able to get read-only access to the database. Since ODBC and JDBC drivers are by default included in Azure SQL databases, these connections can still be used. Using these connections allow for an easy go-live scenario.



Current users and the schemas they have access too: (Please find login details in the key vault)

- AnalysisUser [MAR_ANALYSIS, MAR_API4, MAR_HISTORY4, MAR_VERIF4, MARINFO4, SSN, THETIS, MAR_DONA]
- CrewlistUser [Crewlist]

- Thetis_User [MARINFO4]
- DONA_User [MAR_DONA]
- Mss_user [MARINFO4/SSL]
- CSDReadOnlyUser [MAR_DONA]

You can access the key vault by clicking the link below:

<https://portal.azure.com/#@emsaeuropa.onmicrosoft.com/resource/subscriptions/49dbba4a-f9cd-453f-acce-1f036b6c5e2c/resourceGroups/marinfo-prod-we-rg/providers/Microsoft.KeyVault/vaults/marinfo-prod-we-kv/secrets>

4.1.4.3 SSN Exports

Home > Storage accounts > marinfoprodwedl | Containers >

exports ...

Container

Search

Upload Add Directory Refresh Rename Delete Change tier

Authentication method: Access key (Switch to Azure AD User Account)

Location: exports / SSN

Search blobs by prefix (case-sensitive)

Name	Modified	Access tier
<input type="checkbox"/> [.]		
<input type="checkbox"/> LOTA_MAX_MV		
<input type="checkbox"/> PORTSDATA		

Settings

- Shared access tokens
- Manage ACL
- Access policy
- Properties

The SSN exports: LOTA_MAX_MV and PORTSDATA can be found by using the following link:

https://portal.azure.com/#view/Microsoft_Azure_Storage/ContainerMenuBlade/~/overview/storageAccountId/%2Fsubscriptions%2F49dbba4a-f9cd-453f-acce-1f036b6c5e2c%2FresourceGroups%2Fmarinfo-prod-we-rg%2Fproviders%2FMicrosoft.Storage%2FstorageAccounts%2Fmarinfoprodwedl/path/exports/etag/%220x8D996FD547EF415%22/defaultEncryptionScope/%24account-encryption-key/denyEncryptionScopeOverride~/false/defaultId//publicAccessVal/None

To toggle on/off exports to the datalake, configurations can be stored/adjusted in the following SQL table:

```
FROM [MAR_VERIF4].[VIEWS]
```

VIEW_ID	VIEW_NAME	VIEW_SCHEMA	TABLE_NAME	TABLE_SCHEMA	STAGING_SCHEMA	TO_BE_MATERIALIZED	TO_BE_EXPORTED	TO_BE_EXPORTED_PATH
1	V_Incidents	MAR_API4	Incidents	MAR_API4	STAGING	1	0	NULL
2	V_Inspections	MAR_API4	Inspections	MAR_API4	STAGING	1	0	NULL
3	Company_Current	MAR_API4	NULL	NULL	NULL	0	0	NULL
4	Company_Details	MAR_API4	NULL	NULL	NULL	0	0	NULL
5	Crew	MAR_API4	NULL	NULL	NULL	0	0	NULL
6	FLAG_HISTORY_WITH_END_DATE	MAR_API4	NULL	NULL	NULL	0	0	NULL
7	MANAGER_HISTORY_WITH_END_DATE	MAR_API4	NULL	NULL	NULL	0	0	NULL
8	NAME_HISTORY_WITH_END_DATE	MAR_API4	NULL	NULL	NULL	0	0	NULL
9	OWNER_HISTORY_WITH_END_DATE	MAR_API4	NULL	NULL	NULL	0	0	NULL
10	V_Port_Calls	MAR_API4	Port_Calls	MAR_API4	STAGING	0	0	NULL
11	V_Company_Ownership_Information	MAR_API4	Company_Ownership_Information	MAR_API4	STAGING	1	0	NULL
12	Vessel_Flag_History	MAR_API4	NULL	NULL	NULL	0	0	NULL
13	Vessel_Name_History	MAR_API4	NULL	NULL	NULL	0	0	NULL
14	LOTA_MAX_MV	SSN	NULL	NULL	NULL	0	1	exports/SSN/LOTA_MAX_MV
15	V_PORTSADATA	SSN	NULL	NULL	NULL	0	1	exports/SSN/PORTSDATA

TO_BE_EXPORTED should be 1, and a path needs to be configured.

4.1.4.4 Web Services

MARINFO has web services available to get data from the database. RESTful services are developed that deliver information in JSON format. All APIs uses pagination and request throttling. The use cases for the web services are explained in more detail below. Some of the data is available in multiple lots (Lot1C and lot2 overlap in terms of data). Note that for the web services where data is available in multiple places, data from one lot will be used.

Vessel Name History

Via a webservice, it is possible to retrieve a list of names which were used by a set of specified vessels, with the dates corresponding to when the name was used by the vessel(s). Multiple IMO numbers can be given as input and the number of names that the webservice can return per vessel is a one to many relationship.

Parameters:

- IMO number(s) of the vessel(s)

Used Tables:

- Lot1C_History_Name

Return value:

- List
 - IMO Number
 - Name
 - Effective Date

Pagination:

Yes, (maximum of 100 vessels)

Vessel Flag History

Via a webservice, it shall be possible to retrieve a list of flags which were used by a set of vessels, with the dates corresponding to when the flag was used by the vessel(s). Multiple IMO numbers can be given as input and the number of names that the webservice can return per vessel is a one to many relation.

Parameters:

- IMO number(s) of the vessel(s)

Used Tables:

- Lot1C_History_Flag

Return value:

- List
 - IMO Number
 - Flag
 - Effective Date

Pagination:

Yes, (maximum of 100 vessels)

Company Current

Via a webservice, it shall be possible to retrieve the current ownership and management details of a set of specified vessels. Multiple IMO numbers can be given as input.

Parameters:

- IMO number(s) of the vessel(s)

Used Tables:

- Lot1C_Ship

Return value:

- List
 - IMO Number
 - Registered Owner (full name)
 - Group Beneficial Owner (full name)
 - Ship Manager (full name)
 - Operator (full name)
 - Technical Manager (full name)
 - DOC Company (full name)

Pagination:

Yes, (maximum of 100 vessels)

Company Previous

Via a webservice, it shall be possible to retrieve the previous ownership and management details of a specified vessel, with the corresponding dates. Multiple IMO numbers can be given as input and the number of entries for previous ownership and management details that the webservice can return per vessel is a one to many relation.

Parameters:

- IMO number(s) of the vessel(s)

Used Tables:

- Lot2_History_Owner
- Lot2_History_BenefOwner
- Lot2_History_Manager
- Lot2_History_Operator
- Lot2_History_TechManager
- Lot2_History_DOC

Return value:

- List
 - IMO Number
 - Registered Owner (full name)
 - Registered Owner Start Date
 - Registered Owner End Date
 - Group Beneficial Owner (full name)
 - Group Beneficial Owner Start Date
 - Group Beneficial Owner Start Date
 - Ship Manager (full name)
 - Ship Manager Start Date
 - Ship Manager Start Date
 - Operator (full name)
 - Operator Start Date
 - Operator Start Date
 - Technical Manager (full name)
 - Technical Manager Start Date
 - Technical Manager Start Date
 - DOC Company (full name)
 - DOC Company Start Date
 - DOC Company Start Date

Company Details

Via a webservice, it shall be possible to retrieve the current details of a company. Multiple IMO numbers or company names can be given as input. Wildcards (*) should be accepted.

Parameters:

- Company IMO number(s)

OR

- Company Name(s)

Used Tables:

- Lot1C_Company
- Lot1C_Company_all

Return value:

- List
 - Company IMO number
 - Company Name
 - Country Name
 - Nationality of Registration
 - Nationality of Control
 - Company Status
 - Founded Date

Pagination:

Yes, (maximum of 100 vessels)

Company Fleet

Via a webservice, it shall be possible to retrieve a list of vessel IMO numbers connected, within a predefined range of dates, with a company, i.e. the company has one of the following relationships to the vessel:

- Registered Owner
- Group Beneficial Owner
- Ship Manager
- Operator
- Technical Manager
- DOC Company

Wildcards (*) should be accepted.

Parameters:

- Company IMO number(s)

OR

- Company Name(s)

AND

- Start Date
- End Date

Used Tables:

- Lot2_History_Owner
- Lot2_History_BenefOwner
- Lot2_History_Manager
- Lot2_History_Operator
- Lot2_History_TechManager
- Lot2_History_DOC

Return value:

- List
 - Company IMO Number
 - Company IMO Name
 - IMO numbers of vessels connected to the company

Crew

Via a webservice, it shall be possible to retrieve the details of the crew of a set of vessels. Multiple IMO numbers can be given as input and the number of entries for the details of the crew that the webservice can return per vessel is a one to many relation.

Parameters:

- IMO number(s) of the vessel(s)

Used Tables:

- Lot1C_Crew_List

Return value:

- List
 - Vessel IMO number
 - Crew List Date
 - Nationality
 - Total Crew
 - Total Ratings
 - Total Officers

Pagination:

Yes, (maximum of 100 vessels)

Incidents

Via a webservice, it shall be possible to retrieve the casualty and events data for a set of vessels. Multiple IMO numbers can be given as input and the number of entries for casualty and events data that the webservice can return per vessel is a one to many relation.

Parameters:

- IMO number(s) of the vessel(s)

Used Tables:

- Lot1E_Casualties
- Lot1E_Events
- Lot1C_Ship
- Lot1C_History_Flag

Return value:

- List
 - Vessel IMO number
 - Incident Date
 - Vessel Name At Time Of Incident
 - Vessel Flag At Time Of Incident
 - Vessel Type At Time Of Incident (if not, current Type)
 - Vessel GT At Time Of Incident (if not, current GT)
 - Vessel DWT At Time Of Incident (if not, current DWT)
 - Vessel Date of Build or Age
 - Severity Indicator
 - Total Loss Indicator
 - Casualty type
 - Precis Text
 - Complimentary Text
 - Number Killed
 - Number Missing
 - Other vessel involved IMO Number 1
 - Other vessel involved IMO Number 2
 - Other vessel involved IMO Number 3

Pagination:

Yes, (maximum of 100 vessels)

Inspections

Via a webservice, it shall be possible to retrieve the details of Port State Control inspections performed for a set of vessels. Multiple IMO numbers can be given as input and the number of entries for the details of Port State Control inspections that the webservice can return per vessel is a one to many relation.

Parameters:

- IMO number(s) of the vessel(s)

Used Tables:

- Lot1C_Ship
- Lot1C_History_Flag
- Lot1F_Inspections

Return value:

- List
 - Vessel IMO number
 - Vessel Name when inspected
 - Vessel Flag when inspected
 - Vessel Owner when inspected
 - Vessel Manager when inspected
 - Vessel Type when inspected (if not, current Type)
 - Vessel GT when inspected (if not, current GT)

- Vessel DWT when inspected (if not, current DWT)
- Vessel Date of Build or Age
- Inspection Date
- Authorisation
- Inspection Port
- Country
- Number of Defects
- Ship Detained Indicator
- Number of Days Detained
- Number of Part Days Detained
- Follow up Inspection
- Expanded Inspection Indicator
- Other Inspection Type
- Release Date

Pagination:

Yes, (maximum of 100 vessels)

Port Calls

Via a webservice, it shall be possible to retrieve the port calls and ship transits for a set of vessels for a determined time period. Multiple IMO numbers can be given as input and the number of entries for port calls that the webservice can return per vessel is a one to many relation.

Parameters:

- IMO number(s) of the vessel(s)
- Start Date
- End Date

Used Tables:

- Lot1A_Movements

Return value:

- List
 - Vessel IMO number
 - Port Name (if a port call)
 - Port ID (if a port call)
 - Port Country (if a port call)
 - Arrival Date/Transit date
 - Sail Date (if a port call)
 - Destination (if a port call)
 - ETA (if a port call)
 - Previous port
 - Movement Type (port call or transit)

Pagination:

Yes, (maximum of 100 vessels)

4.1.4.5 Power BI

EMSA users are also using Power BI to access the data in the Azure SQL DB to create relevant reports and dashboards.

4.1.4.6 Functional Monitoring

The functional monitoring part is a joint result of several Azure technologies. An Azure data factory is in place for orchestration purposes. The data factory has some built-in mechanisms that can be used for monitoring. It is also this tool that EMSA users will use to manually trigger a run of the flow. Logs of the data factory will be stored in a database, and it will be possible to query these results.

The second technology that is used is a logic app. The logic app will be in place to send the customized reports to the end-users (both contractors and EMSA users). A report will be sent in various cases:

1. When one of the systems is unavailable, an e-mail will be sent to EMSA staff stating the unavailable systems (can be FTP server, SQL Server, storage account, databricks).
2. When the checksum validation fails, meaning that there are missing files, incomplete files or the MD5 hash is not correct, an e-mail will be sent to both EMSA staff and the contact person of the lot where the problem occurred.
3. When data validation failed due to data type mismatches or missing/incorrect key references, an e-mail will be sent to both EMSA staff and the contact person of the lot where the error occurred.

Since end-users should be able to run SQL Queries on the MARINFO database, Power BI is also added as a part of the monitoring. It will be able to write custom queries and execute them on the MARINFO database. A basic (example) report will be built with information about the run and eventual errors during the run. The information in the report will be the same information that is contained in the e-mail that is sent in case of an error during the flow. EMSA users will get a training about the use of Power BI which will enable end-users to create their own dashboards and execute their own queries going forward.

4.2 MARINFO History

This table will contain all changes that were ever made to the MARINFO data, to allow the business users to recreate the shipping universe at any given date in the past. For every data change (insert or update) an extra row is added to the table. Every row has a record start date and a record end date to indicate the validity of this information. In literature this modelling technique is commonly called a slowly changing dimension type 2.

The MARINFO History is an exact copy of the whole MARINFO database, but for each record, a begin and end date are stored. In the picture below, an example of the history tables are shown for lot 1. The names of the tables are exactly the same as in the MARINFO database, but with a prefix *H*. It also shows the fields that are present in one of the tables. The fields are the same as for the MARINFO database extended with an ETL_ID, ETL_VERSION, ETL_DEBUT and ETL_FIN.

The screenshot shows a database management tool interface. On the left, a tree view displays the database structure, including a table named 'MARINFO4.LOT1A_MOVEMENTS (System-Versioned)' and its history table 'MAR_HISTORY4.LOT1A_MOVEMENTS (History)'. The columns for the history table are listed: CALLID (int, not null), IMO (nvarchar(7), not null), SHIPNAME (nvarchar(50), not null), SHIPTYPE (nvarchar(255), not null), MOVEMENTTYPECODE (nvarchar(1), null), COUNTRY (nvarchar(40), null), OLDPORTID (nvarchar(6), null), PORTID (int, null), PORTNAME (nvarchar(40), null), PORTGEOID (int, null), ZONENAME (nvarchar(100), null), ANCHORAGEPARENTID (int, null), ANCHORAGEPARENTNAME (nvarchar(40), null), ARRIVALDATE (date, null), ARRIVALDATEFULL (datetime, null), SAILDATE (date, not null), SAILDATEFULL (datetime, not null), HOURSINPORT (int, null), ARRIVALDRAUGHT (decimal(5,3), null), DEPARTUREDRAUGHT (decimal(5,3), null), DESTINATION (nvarchar(255), null), ETA (datetime, null), LASTPORTOFCALLID (int, null), LASTPORTOFCALLCOUNTRYCODE (nvarchar(10), null), LASTPORTOFCALLCRYNAME (nvarchar(40), null), LASTPORTOFCALLNAME (nvarchar(100), null), LASTPORTOFCALLARRIVALDATE (datetime, null), LASTPORTOFCALLSAILDATE (datetime, null), LASTPORTOFCALLCODE (int, null), MOVEMENTTYPE (nvarchar(310), null), and DATECREATED (datetime, not null).

On the right, a SQL query is shown, selecting various fields from the 'MAR_HISTORY4.LOT1A_MOVEMENTS' table. The query is as follows:

```

, [LASTPORTOFCALLCRYNAME]
, [LASTPORTOFCALLNAME]
, [LASTPORTOFCALLARRIVALDATE]
, [LASTPORTOFCALLSAILDATE]
, [LASTPORTOFCALLCODE]
, [MOVEMENTTYPE]
, [DATECREATED]
, [LOAD_ID]
, [ETL_ID]
, [ETL_VERSION]
, [ETL_DEBUT]
, [ETL_FIN]
FROM [MAR_HISTORY4].[H_LOT1A_MOVEMENTS]

```

Below the query, a table of results is displayed, showing columns: LARRIVALDATE, LASTPORTOFCALLSAILDATE, LASTPORTOFCALLCODE, MOVEMENTTYPE, DATECREATED, LOAD_ID, ETL_ID, ETL_VERSION, ETL_DEBUT, and ETL_FIN. The table contains 24 rows of data, with the first row having a value of 14485 in the LASTPORTOFCALLCODE column and 16 in the LOAD_ID column.

The ETL_DEBUT and ETL_FIN are timestamp fields for the beginning and end time of a record and are based on the time at which a record was processed. A record is updated (.upd file type) if one of its fields are changed in comparison with the current record. If this is the case, the time at that moment (ETL time) is used as the end time of the previous record and as the start time of the new (current version) record. The end time of the new record is set to a default value. Besides the start and end time, an ETL_ID and an ETL_VERSION are stored (a composite primary key is created for these two fields). These fields are automatically filled in when creating a new record. The ETL version will increase by one for each update of a specific record in the history.

- MARINFO4.LOT2A_HISTORY_BENEFOWNER (System-Versioned)**
- MARINFO4.LOT2A_HISTORY_CALL_SIGN_MMSI (System-Versioned)**
- MARINFO4.LOT2A_HISTORY_CLASS (System-Versioned)**
- MARINFO4.LOT2A_HISTORY_DOC (System-Versioned)**
- MARINFO4.LOT2A_HISTORY_FLAG (System-Versioned)**
- MARINFO4.LOT2A_HISTORY_FLAG_DEC (System-Versioned)**
- MARINFO4.LOT2A_HISTORY_MANAGER (System-Versioned)**
- MARINFO4.LOT2A_HISTORY_NAME (System-Versioned)**
- MARINFO4.LOT2A_HISTORY_OPERATOR (System-Versioned)**
- MARINFO4.LOT2A_HISTORY_OWNER (System-Versioned)**
- MARINFO4.LOT2A_HISTORY_PANDI (System-Versioned)**
- MARINFO4.LOT2A_HISTORY_STATUS (System-Versioned)**
- MARINFO4.LOT2A_HISTORY_TECHMANAGER (System-Versioned)**

If a record is deleted (.del file type), the end time at which the record is processed is used as the end date for that record in the history and no new record is created. When a synchronization occurs (.all file type), only updated and deleted records are changed in the history database.

Note that changes to the MARINFO_HISTORY database depend on the data in the MARINFO_HISTORY database itself and do not depend on the MARINFO database. However, the MARINFO_HISTORY should contain all data present in the MARINFO database and hence both should be in sync regarding the most recent data.

	Columns	Data	Model	Constraints	Grants	Statistics	Flashback	Dependencies	Details	Partitions	Indexes	SQL	
						Filter:							▼ Actions...
		ETL_ID	ETL_VERSION	ETL_DEBUT	ETL_FIN	LRIMOSHIPNO	MMSI	SHIP_NAME	SHIP_TYPE_NAME_LV5	SHIP_TYPE_CODE_LV5			
25		205474	1	25-JAN-17	28-SEP-18								
26		205475	1	25-JAN-17	30-MAY-19								
27		205476	1	25-JAN-17	01-JAN-00								
28		205477	1	25-JAN-17	01-JAN-00								
29		205478	1	25-JAN-17	01-JAN-00								
30		205479	1	25-JAN-17	01-JAN-00								
31		205480	1	25-JAN-17	01-JAN-00								
32		205481	1	25-JAN-17	01-JAN-00								
33		205482	1	25-JAN-17	21-FEB-18								
34		205321	1	25-JAN-17	27-NOV-18								
35		205322	1	25-JAN-17	01-JAN-00								
36		205323	1	25-JAN-17	01-JAN-00								
37		205324	1	25-JAN-17	01-JAN-00								
38		205325	1	25-JAN-17	01-JAN-00								
39		205326	1	25-JAN-17	21-FEB-18								
40		205327	1	25-JAN-17	21-FEB-18								
41		205328	1	25-JAN-17	01-JAN-00								
42		205329	1	25-JAN-17	01-JAN-00								

4.3 MARINFO Verification

The MARINFO Verification database will hold all data and metadata regarding uploading of the data. It contains the logs of the loading process and can help in identifying issues with the data. Each record in the MARINFO Verification database consists of a `LOAD_ID` and a timestamp when the operation was executed and a status of the operation (success/failure/pending). Some other extra fields can be added for more clarification (e.g. the filename, the number of files inside the zip, ...)

Besides, tables containing logs are stored in this database for each run.

5. Data governance and security

5.1 Azure governance

5.1.1 Naming conventions

In this paragraph the naming conventions used for all resources are indicated.

5.1.1.1 Azure resources

The following naming conventions are used for all Azure resources

application-tier-[location]-resource

In the above structure the components are listed in the table below

Property	Value
Location	Westeurope → we
Application	marinfo
Tier	Dev, test, mgt or prod
Resource	An abbreviation of the Azure resource

Remarks:

- For subresources of virtual machines like a network interface card (nic) the machine name is taken and the subresource abbreviation is put after the virtual machine name.
For example: marinfo-dev-we-nic.

5.1.1.2 Service principals

The same structure is followed where svp is the abbreviation for service principal

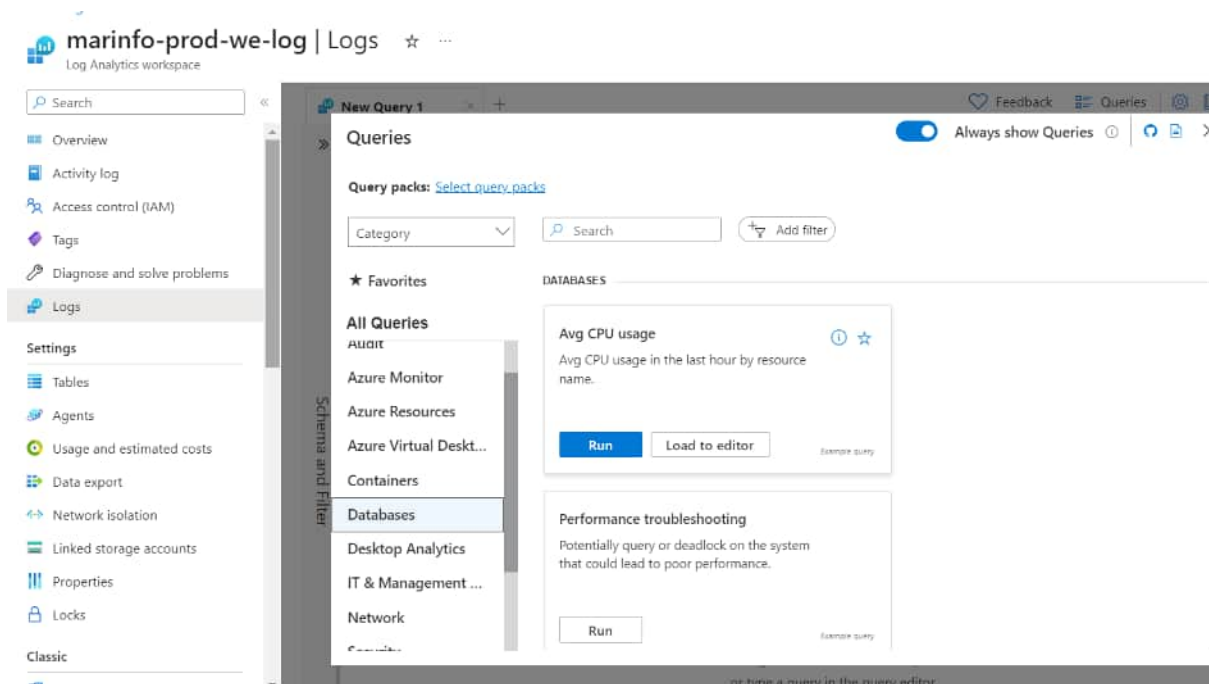
5.1.2 Logging, monitoring and alerting

All resources which are capable of sending their logs to an Azure Log Analytics Workspace have this enabled in their diagnostic settings where relevant. All relevant available logs are sent to enable proper troubleshooting and tracing back events when necessary.

To access the different Log Analytics Workspaces (One for each environment) you can follow the link below:

<https://portal.azure.com/#view/HubsExtension/BrowseResource/resourceType/Microsoft.Operationallnsights%2Fworkspaces>

These 'Logs' can be queried in the Logs tab of the Log Analytics workspace. Here, default queries are made available to inquire certain popular indicators. The picture below shows how to for example get Avg CPU usage for the databases in the MARINFO-PROD environment.



5.1.3 Availability

All components within the solution that are used for processing the data are Platform as a service components providing the necessary high-availability. The uptime service level agreements (SLA) for these components is presented in the table below.

A disaster recovery environment is currently not incorporated in this design.

Component	Uptime SLA
Azure Storage Account	Reads: 99.9% (LRS, ZRS and GRS) 99.99% (RA-GRS)
	Writes: 99.9%
Azure Automation Account	99.9%
Azure SQL Database	99.99%
Azure VM	99.9%
Azure Key Vault	99.9%
Azure Databricks	99.95%
Azure Data Factory	99.9%
Azure Log Analytics Workspace	99.9%

5.2 Azure component security

5.2.1 Azure Log Analytics Workspace

5.2.1.1 General

A Log Analytics workspace is a unique environment for Azure Monitor log data. Each workspace has its own data repository and configuration, and data sources and solutions are configured to store their data in a particular workspace. A log analytics workspace is deployed in each environment. Diagnostic logs of the resources are written to the workspace of the corresponding environment.

Please find the logs by following the link below:

<https://portal.azure.com/#view/HubsExtension/BrowseResource/resourceType/Microsoft.OperationalInsights%2Fworkspaces>

The pricing of this component depends on the amount of data (considered under the Azure monitor component) that is written to the workspace (billing per GB in Pay-As-You-Go model).

Parameter	Development	Test	Production
Location	West Europe	West Europe	West Europe
Pricing Tier	Pay-as-you-go	Pay-as-you-go	Pay-as-you-go
Tag description	Log Analytics workspace for monitoring and alerting of the environment	Log Analytics workspace for monitoring and alerting of the environment	Log Analytics workspace for monitoring and alerting of the environment
Lock	Delete lock		Delete lock

5.2.1.2 Security

An Azure log analytics workspace is deployed for each environment which causes the data of the environments to be entirely separated in location as well as in access permissions. The Azure log analytics workspace does not contain sensitive data and logging data is only accessible for users with the monitoring data readers or higher permissions.

5.2.1.3 High-availability and disaster recovery

The log analytics workspace is not critical for the workings of the application but offers an SLA of 99.9%. Together with Azure monitor which is by default available in the Azure subscription the monitoring and the alerting within the environment is managed.

A separate log analytics workspace is deployed in a disaster recovery environment in case this is foreseen to capture the logging of the disaster recovery components.

5.2.2 Data Storage data lake

An Azure Data lake storage account is the same as a normal Azure storage account where the hierarchical namespace property is enabled. Data processed in Azure Databricks is stored in this Data lake. It also serves as the landing zone for the data.

Parameter	Development	Test	Production
Performance	Standard	Standard	Standard
Account Kind	StorageV2	StorageV2	StorageV2
Replication	LRS	LRS	LRS
Access tier	Hot	Hot	Hot
Connectivity method	Allow trusted Microsoft services	Allow trusted Microsoft services	Allow trusted Microsoft services
Secure Transfer required	Enabled	Enabled	Enabled
Large file shares	Disabled	Disabled	Disabled
Soft delete	Disabled (not possible)	Disabled (not possible)	Disabled (not possible)
Hierarchical namespace	Enabled	Enabled	Enabled
Containers	<ul style="list-style-type: none"> - databricks - exports - overwrite - datalake 	<ul style="list-style-type: none"> - databricks - exports - overwrite - datalake 	<ul style="list-style-type: none"> - databricks - exports - overwrite - datalake
Tag description	Data Storage	Data Storage	Data Storage
Lock	Delete lock	Delete lock	Delete lock

Next to this resource configuration the Databricks service principal is added to the Storage contributor role on this storage account.

5.2.2.1 Security

A number of storage accounts contain sensitive data which is handled with the necessary care.

- Secure transfer required is enabled on all accounts allowing no unencrypted connections to the storage accounts
- Stored data is automatically encrypted on all storage accounts using 256-bit AES encryption. This is done with the Microsoft managed key.
- All storage accounts containing data have their firewall configured to be only accessible from the virtual network
- All Azure files storage accounts have their firewall configured to be only accessible from the virtual network. They are accessed by end-users outside the virtual network using a private endpoint which ensures that all traffic goes over the local network and Expressroute connection to Azure rather than the public internet.

5.2.3 High-availability and disaster recovery

The storage accounts form a critical part of the application and offer a 99.9% SLA.

In a disaster recovery set-up the redundancy of the data storage would be changed to the RA-GRS level making the data available in the paired Azure data center. In the case of Azure West-Europe this is Azure North-Europe. Data can be processed there without any issues and failover is completely transparent for the application. Non-date storage is just deployed in the same way in the disaster recovery tier. Azure files does currently not have the RA-GRS redundancy level. In a disaster recovery scenario a second Azure Files storage account would need to be provided.

5.3 Azure Key Vault

An Azure key vault is deployed in each environment to store all secrets, keys and certificates that are used in the corresponding environment.

Parameter	Development	Test	Production
Pricing Tier	Premium	Premium	Premium
Soft delete	Enable	Enable	Enable
Retention period (days)	90	90	90
Purge protection	Disabled	Disabled	Disabled
Azure Virtual Machines access	Yes	Yes	Yes
Azure Resource Manager access	Yes	Yes	Yes
Azure Disk Encryption Access	Yes	Yes	Yes
Tag description	Azure Key Vault for storing of secrets	Azure Key Vault for storing of secrets	Azure Key Vault for storing of secrets
Lock	Delete lock	Delete lock	Delete lock
Diagnostic Settings	Log Analytics workspace Log: <ul style="list-style-type: none"> - JobLogs - JobStreams - DscNodeStatus Metric: <ul style="list-style-type: none"> - AllMetrics 	Log Analytics workspace Log: <ul style="list-style-type: none"> - JobLogs - JobStreams - DscNodeStatus Metric: AllMetrics	Log Analytics workspace Log: <ul style="list-style-type: none"> - JobLogs - JobStreams - DscNodeStatus Metric: AllMetrics

Table 2: Azure Key Vault configuration

5.3.1 Access Policies

The following access policies are in place for the key vaults:

Key Vault	Permissions
<u>marinfo-dev-we-kv</u>	Secret: Get, List, set, delete
<u>marinfo-test-we-kv</u>	Secret: Get, List, set, delete
<u>marinfo-prod-we-kv</u>	Secret: Get, List, set, delete

5.3.2 Security

The Azure Key Vault contains a number of configuration data and credentials that are handled with the necessary care.

- All key vaults have their firewall configured to be only accessible from the virtual network. When accessing the key vault even from the Azure Portal in the browser it is not available if the connection isn't coming from within the virtual network
- Access policies are restricted and indicated in the access policies configuration table
- Passwords in the vault are automatically generated and placed in the key vault automatically without manual intervention.

5.3.3 High-availability and disaster recovery

The Azure key vault forms a critical part of the application and offers a 99.9% SLA.

Currently, there is no disaster recovery component active for the Azure Key Vault. In a disaster recovery set-up a second key vault is required for the disaster recovery environment.

5.4 Azure Databricks

5.4.1 General

Parameter	Development	Test	Production
Pricing Tier	Standard	Standard	Standard
Deploy in own vnet	Yes	Yes	Yes
Virtual network	N/A	N/A	N/A
Public subnet name	N/A	N/A	N/A
Public subnet CIDR Range	N/A	N/A	N/A
Private subnet name	N/A	N/A	N/A
Private subnet CIDR Range	N/A	N/A	N/A
Tag description	Azure Databricks workspace for processing of the data	Azure Databricks workspace for processing of the data	Azure Databricks workspace for processing of the data
Lock	Delete lock	Delete lock	Delete lock

Table 3: Azure Databricks workspace configuration

5.4.2 Databricks cluster

Currently, each environment is equipped with autoscaling cluster pools. These will only be active when a job run is triggered.

Parameter	Development	Test	Production
Pool	ADFpool	ADFpool	ADFpool
Enable autoscaling	True	True	True
Worker Type	Standard_DS3_v2	Standard_DS3_v2	Standard_DS3_v2
Workers	0 - 35	0 - 35	0 - 35
Driver Type	Standard_DS3_v2	Standard_DS3_v2	Standard_DS3_v2

5.4.3 Manual actions

The creation of an Azure Key Vault backed secret scope for Azure Databricks is described in 6.2 Create an Azure Key Vault-backed secret scope. This will create an access policy on the Azure Key Vault for retrieving the secrets. This access policy is included in the key vault access policies.

5.4.4 Permissions

After the initial deployment specific permissions for the users will be determined in the different environments. Accessing the Azure Databricks workspace is done using Azure AD.

5.4.5 Security

The Azure Databricks workspace contains and orchestrates the jobs for processing the data. It does not store that data itself. Storing is the responsibility of the Azure Data Lake. For the processing of the data by Azure Databricks the following security actions are implemented.

- Databricks makes use of the Azure key vault and stores no secrets by itself
- Access to the Azure Key Vault for the secrets and Azure Data Lake for the data is done by using a service principal and access to that service principal is explicitly added on the Data Lake and Key Vault.
- Deployment of Azure Databricks is done within the virtual network rather than using the default virtual network provided by Azure Databricks itself. It ensures the data never leaves the private network during processing.

5.4.6 High-availability and disaster recovery

The Azure Databricks workspace forms a critical part of the application and offers a 99.95% SLA.

In a disaster recovery set-up a second Azure Databricks workspace is set up and is capable of running the required workload.

5.5 Virtual Networks

A virtual network is deployed for each environment and forms a spoke in the network hub-spoke model which is peered with the hub virtual network. This hub network is connected to the on-premises data sources and end-users residing on the sites and in the local datacenter.

NSGs and route tables are created for each subnet but currently left at the default values.

<input type="checkbox"/> Name ↑↓	Resource group ↑↓
<input type="checkbox"/>  marinfo-dev-we-avn	marinfo-dev-we-rg
<input type="checkbox"/>  marinfo-mgt-we-avn	marinfo-mgt-we-rg
<input type="checkbox"/>  marinfo-prod-we-avn	marinfo-prod-we-rg
<input type="checkbox"/>  marinfo-test-we-avn	marinfo-test-we-rg
<input type="checkbox"/>  workers-vnet	marinfo-dev-we-dbw-arg
<input type="checkbox"/>  workers-vnet	marinfo-prod-we-dbw-arg
<input type="checkbox"/>  workers-vnet	marinfo-test-we-dbw-arg

5.6 Azure Data Factory

5.6.1 General

Azure Data Factory is Azure's cloud ETL service for scale-out serverless data integration and data transformation. It offers a code-free UI for intuitive authoring and single-pane-of-glass monitoring and management. It orchestrates the data movement from the on-premises data sources to the Azure Data lake where the data is then processed further.

Parameter	Development	Test	Production
Version	V2	V2	V2
Enable Git	True	True	True
Git URL	https://dev.azure.com/ems-a-marinfo/marinfo/_git/DataFactory	https://dev.azure.com/ems-a-marinfo/marinfo/_git/DataFactory	https://dev.azure.com/ems-a-marinfo/marinfo/_git/DataFactory
Repo Name	DataFactory	DataFactory	DataFactory
Branch Name	master	master	master
Root Folder	Factory	Factory	factory
Tag description	Azure Data Factory for data movement orchestration	Azure Data Factory for data movement orchestration	Azure Data Factory for data movement orchestration
Lock	Delete lock	Delete lock	Delete lock

5.6.2 Post-Setup Actions

CI/CD setup for pipelines: More information regarding the CI/CD setup of data factory can be found in section

The data factory must be able to access on-premises data sources. For this purpose, a self-hosted integration runtime must be installed on an Azure VM.

for a complete list of all access policies per environment.

5.6.3 Security

Azure data factory is an orchestrator and does not contain any data. It instructs the self-hosted integration runtime machine to move the data from the on-premises data sources to the Azure Data

lake and instructs Azure Databricks to execute its workload. The following security actions are implemented.

- Azure Data Factory accesses the key vault using its own service principal
- Access to the on-premises data sources is only possible through the self-hosted integration runtime, ensuring that no data leaves the internal network. In the case of public ftp this is not the case.

5.6.4 High-availability and disaster recovery

Azure Data Factory forms a critical part of the application and offers a 99.9% availability SLA. It also depends on the availability of the self-hosted integration runtime.

In a disaster recovery set-up another Azure Data factory is deployed in the secondary region.

5.7 Azure SQL

Azure SQL Database is the intelligent, scalable, cloud database service that provides the broadest SQL Server engine compatibility.

5.7.1 Azure SQL Server

Some parameters must be configured during setup of the Azure SQL Database. The following applies for the SQL Server:

Parameter	Development	Production
Server admin login	sqladmin	N/A
Password	Generated automatically Secret: sqladmin-pwd	N/A
Allow access to Azure services	Enabled	N/A
Advanced data security	Enabled	N/A
Tag description	Azure SQL Server to host databases	N/A
Lock	Delete lock	N/A

Additional configurations of the SQL Server

The storage account of the corresponding environment is used to store the results of the vulnerability scan.

All subnets from the virtual network in the management environment must be added to the firewall rules. Advanced data security is enabled and all options are configured

- Advanced threat protection mailing suspicious logins
- Weekly vulnerability assessment assessing changes from the benchmark firewall rules and SQL Server permissions

SQL Server auditing must be enabled on server level with destination the log analytics workspace of the environment.

The Azure AD administrator is the group MIF_INFRA_DEV for development.

Advanced data security will scan all databases on the SQL Server for vulnerabilities every 7 days. Scans will be performed every Sunday at approximately 12:00 AM UTC. The results are mailed to the subscription owners:

The subscription owners can be found/alterred in each subscription by checking the access control.

emsa-marinfo-prod | Access control (IAM) ☆ ...

Subscription

Search « + Add Download role assignments Edit columns Refresh Remove Feedback

Check access **Role assignments** Roles Deny assignments Classic administrators

Number of role assignments for this subscription ⓘ

16 4000

Search by name or email Assignment type: All Type: All Role: All

8 items (4 Users, 3 Groups, 1 Service Principals)

Name	Type	Role
▼ Contributor		
[Redacted]	User	Contributor ⓘ
[Redacted]	Group	Contributor ⓘ
▼ Owner		
[Redacted]	User	Owner ⓘ
[Redacted]	Group	Owner ⓘ
[Redacted]	Group	Owner ⓘ
[Redacted]	App	Owner ⓘ
[Redacted]	User	Owner ⓘ
[Redacted]	User	Owner ⓘ

Or by following this link: (For production)

<https://portal.azure.com/#@emsaeuropa.onmicrosoft.com/resource/subscriptions/49dbba4a-f9cd-453f-acce-1f036b6c5e2c/users>

The credentials of the SQL Admin are stored in the key vault (see section 5.3) of the corresponding environment.

5.7.2 Azure SQL database

Parameter	Development	Test	Production
SQL Elastic pool	No	No	No
Size	S3	S3	S3
Tag description	Azure SQL Database for data storage	Azure SQL Database for data storage	Azure SQL Database for data storage
Lock	Delete lock	Delete lock	Delete lock

An automation runbook to do the SQL Maintenance is deployed to the automation account (see section **Error! Reference source not found.**) to which a weekly schedule is attached. The script that is used to provide the SQL Maintenance is the SQL Server Index and Statistics Maintenance from Ola Hallengren. This script is licensed under the [MIT license](#) and can be found [online](#).

5.7.3 Permissions

Depending on the specific data, the amount and types of schemas the specific permissions for all users and service principals will be determined.

5.7.4 Security

The following measures are taken for ensuring the security of the data in the Azure SQL database.

5.7.4.1 Azure SQL User Accounts

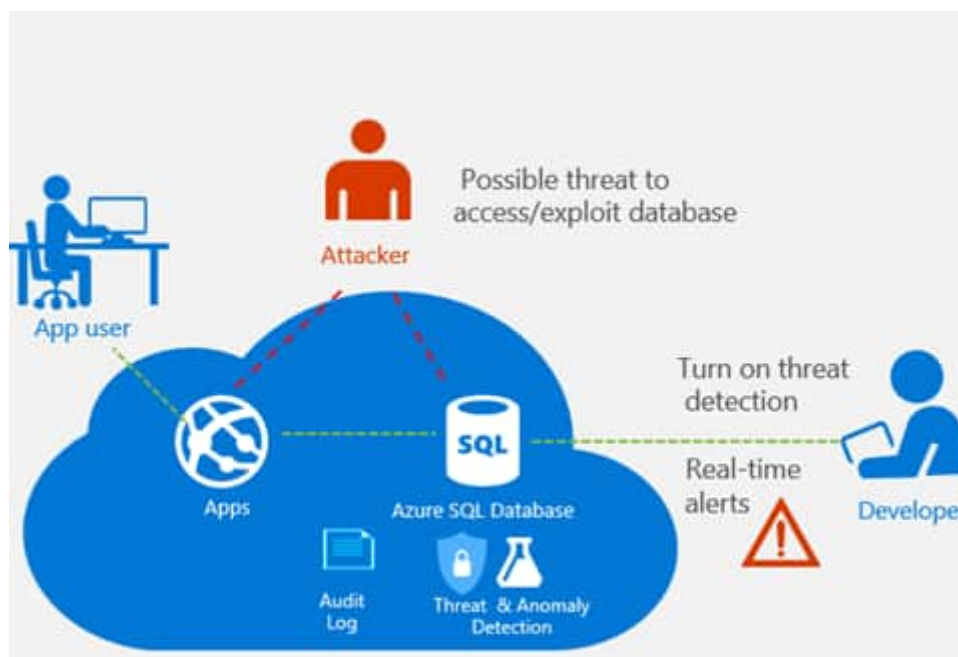
When accessing the database, a user account needs to be provided. This can be a SQL Server user, an Azure AD user or a service principal. Each user has its own permissions and is set to the minimum amount of privileges to execute the necessary tasks. Each Azure SQL Server has an admin user with permissions on the whole database. Next to that there is an Azure SQL AD administrator with the same permissions. This user manages the Azure AD users and their permissions. Whenever possible Azure AD authentication will be used. For technical access service principals are used for authentication. SQL users are not used.

- Azure Data factory connects with its own service principal if necessary
- Azure Databricks connects with its own service principal
- Azure automation account connect with its own service principal
- PowerBI connects with a technical Azure AD user
- The connection is done with its Azure AD users added to the subscription by using the Azure AD group

5.7.4.2 Advanced Threat Protection

Advanced threat protection consists of three components to elevate the security of the Azure SQL database at the cost of € 12.65 per server and will be enabled (included in the estimation above). It contains the components described below and sends weekly vulnerability assessments and security threats to a provided mail address.

- Data Discovery & classification
 - To make sure that all sensitive data in the database is identified it is possible to classify tables and columns. This makes it easier to identify the necessary permissions and apply security features like dynamic data masking
- Vulnerability assessment
 - Using a large range of checks the SQL Server vulnerability assessment identifies possible security risks of the Azure SQL Server like missing firewall settings, database role memberships, ...
- Threat detection
 - Threat detection identifies potential threats like unusual locations from where the Azure SQL database is accessed, brute force credential attacks, SQL injection and more to ensure that action can be taken quickly

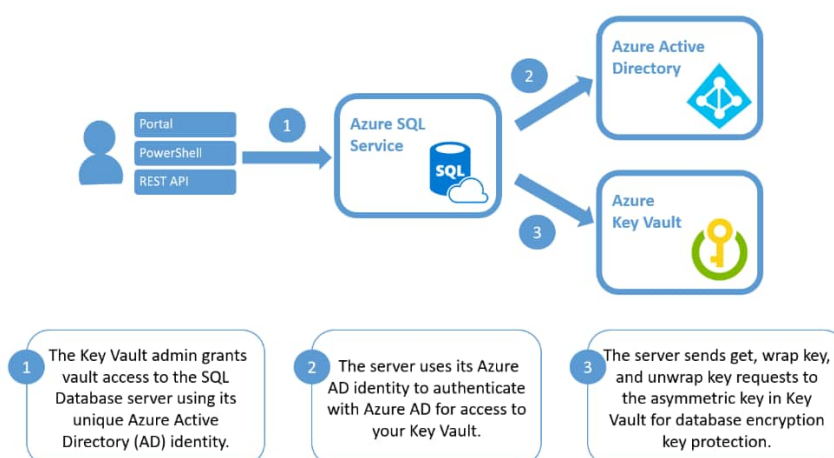


5.7.4.3 Transparent Data Encryption

Transparent data encryption (TDE) makes sure that all data-at-rest is encrypted. In practice this means that when attackers get hold of the database files, transaction log or any database backups these files are encrypted and cannot be restored without the encryption key.

TDE is enabled by default on an Azure SQL database and only adds a small additional cost to the back-ups of the database. The Azure SQL database includes 7 days of point-in-time back-ups being available. If this would not suffice it can be expanded for a longer retention time at a higher cost.

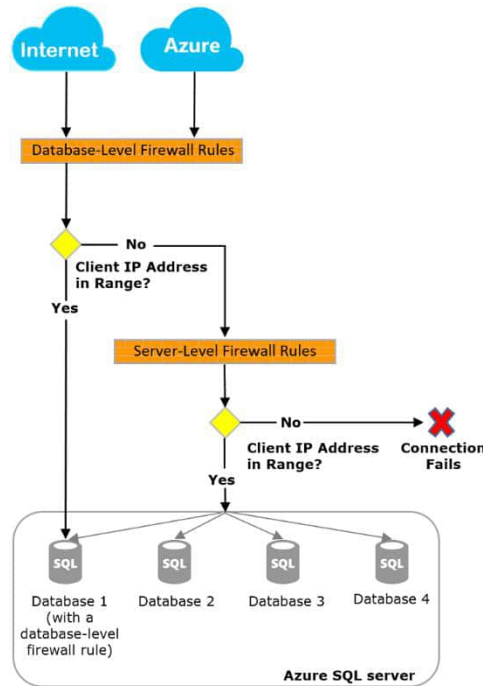
By default, this key is managed by Microsoft but Bring Your Own Key (BYOK) support has been added recently. This feature makes it possible to store your own encryption key in an Azure Key vault and use it to encrypt the database files rather than using the one provided by Microsoft. . **Error! Reference source not found.** shows how this mechanism works.



This feature is activated by default and will not be disabled. The key provided by Microsoft is used.

5.7.4.4 Azure SQL Firewall

While Azure SQL is a service accessible from the internet traffic needs to be explicitly allowed before a connection is possible. Next to that it is possible to deny or allow access to the Azure SQL Database from other Azure services. When enabled no specific access needs to be specified for other Azure services.



5.7.5 High-availability and disaster recovery

Azure SQL database forms a critical part of the application and offers a 99.99% availability SLA.

In a disaster recovery set-up this database will depending on the disaster recovery design be set up in a geo-redundant way or a second Azure SQL database is deployed.

5.8 Disaster Recovery

5.8.1 General strategy

Handling a disaster recovery situation in your Azure data platform solution requires a systematic approach to ensure that you can restore services and data with minimal disruption. Here are the general steps to follow when one or more components fail:

- **Activate the Disaster Recovery Plan (DRP):**

Upon detecting a failure or disaster, the designated personnel should immediately initiate the pre-defined disaster recovery plan for the affected component(s). Ensure that all team members are aware of their roles and responsibilities.

- **Assess the Situation:**

Conduct an initial assessment to determine the scope and severity of the failure. Identify which component(s) are affected and the potential impact on data and services.

- **Isolate the Issue:**

If possible, isolate the issue to prevent further damage or data loss. For example, if a component is compromised due to a security breach, isolate the compromised resources.

- **Restore Services and Data:**

Azure Data Factory: In the case of Azure Data Factory, initiate a failover to a secondary instance if available. Activate predefined pipelines to restore data and functionality. Verify data integrity and pipeline performance.

Azure Data Lake Storage: Restore data from backups to the primary Data Lake Storage account. Ensure that access controls and permissions are reconfigured as necessary.

Azure SQL Database: Restore the SQL Database from backups to the desired point in time. Reconfigure database connections and applications to point to the recovered database.

Azure Databricks: Depending on the severity of the issue, you may need to restore Databricks workspace from snapshots or backups. Ensure that notebooks and job configurations are reinstated.

Azure Key Vault: If Key Vault is compromised, restore secrets and keys from backups. Update access policies and rotate compromised keys as necessary.

- **Testing and Verification:**

After restoring services and data, perform thorough testing to ensure that the recovered components are functioning as expected. Verify data integrity and validate the performance of critical processes.

- **Communication:**

Keep stakeholders, including management and affected users, informed about the situation and the progress of the recovery efforts. Provide realistic estimates of when services will be fully restored.

- **Monitor Continuously:**

Continuously monitor the recovered components and services for any signs of issues or anomalies. Configure alerts and monitoring solutions to detect and respond to potential problems.

- **Root Cause Analysis:**

After the situation is stabilized, conduct a detailed root cause analysis to determine why the failure occurred. This analysis will help you identify areas for improvement in your disaster recovery plan and overall system resilience.

- **Documentation and Reporting:**

Document the entire incident, including the timeline of events, actions taken, and lessons learned. This documentation will be valuable for post-incident analysis and for making improvements to your disaster recovery plan.

- **Review and Update the Disaster Recovery Plan:**

Based on the findings from the root cause analysis and the lessons learned, update your disaster recovery plan to address any shortcomings or areas for improvement. Regularly review and revise the plan to ensure its effectiveness.

Remember that effective communication, rapid response, and well-documented processes are essential during a disaster recovery situation. Being well-prepared and having a tested plan in place will help minimize downtime and data loss in the event of component failures or disasters.

5.8.2 Microsoft documentation

Backup and disaster recovery for Azure applications

<https://learn.microsoft.com/en-us/azure/well-architected/resiliency/backup-and-recovery>

Disaster recovery: Azure databricks

<https://learn.microsoft.com/en-us/azure/databricks/administration-guide/disaster-recovery>

Azure Backup service documentation

<https://learn.microsoft.com/en-us/azure/backup/>

6. Appendix

6.1 Azure DevOps Service principal Azure AD permission configuration

Dashboard > Default Directory | App registrations

Default Directory | App registrations
Azure Active Directory

Search (Ctrl+/) << **+ New registration** Endpoints Troubleshooting Got feedback?

Overview
Getting started
Diagnose and solve problems

Welcome to the new and improved App registrations (now Generally Available). See what's new.

All applications **Owned applications** Applications from personal account

Start typing a name or Application ID to filter these results

Dashboard > Default Directory | App registrations > Register an application

Register an application

* Name
The user-facing display name for this application (this can be changed later).
 ✓

Supported account types
Who can use this application or access this API?

☒ Accounts in this organizational directory only (Default Directory only - Single tenant)
☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant)
☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)

[Help me choose...](#)

Redirect URI (optional)
We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Web

By proceeding, you agree to the [Microsoft Platform Policies](#)

Register

Dashboard > Default Directory | App registrations > DevOpsServicePrincipal

DevOpsServicePrincipal

Search (Ctrl+/)

Overview

Quickstart

Manage

- Branding
- Authentication
- Certificates & secrets
- Token configuration (preview)
- API permissions**
- Expose an API
- Owners
- Roles and administrators (Preview)
- Manifest

Support + Troubleshooting

- Troubleshooting
- New support request

Delete Endpoints

Got a second? We would love your feedback on Microsoft identity platform (previously Azure AD for developer). →

Display name : DevOpsServicePrincipal

Application (client) ID : 0bd7c487-9307-415f-9173-6727990e2560

Directory (tenant) ID : 33ab4e20-221c-480b-a45a-eeadb9a0e4305

Object ID : 31adedfb-2056-4552-9b4d-5c48a5ad1ed1

Supported account types : My organization only

Redirect URIs : [Add a Redirect URI](#)

Application ID URI : [Add an Application ID URI](#)

Managed application in ... : [DevOpsServicePrincipal](#)

Welcome to the new and improved App registrations. Looking to learn how it's changed from App registrations (Legacy)? [Learn more](#)

Call APIs

Build more powerful apps with rich user and business data from Microsoft services and your own company's data sources.

[View API permissions](#)

Documentation

- Microsoft identity platform
- Authentication scenarios
- Authentication libraries
- Code samples
- Microsoft Graph
- Glossary
- Help and Support

Sign in users in 5 minutes

Use our SDKs to sign in users and call APIs in a few steps

[View all quickstart guides](#)

Dashboard > Default Directory | App registrations > DevOpsServicePrincipal | API permissions

DevOpsServicePrincipal | API permissions

Search (Ctrl+/)

Refresh

Overview

Quickstart

Manage

- Branding
- Authentication
- Certificates & secrets
- Token configuration (preview)
- API permissions**
- Expose an API
- Owners
- Roles and administrators (Preview)
- Manifest

Support + Troubleshooting

- Troubleshooting
- New support request














Configured permissions

Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. [Learn more about permissions and consent](#)



[+ Add a permission](#) [Grant admin consent for Default Directory](#)

API / Permissions name	Type	Description	Admin Consent Requ...	Status
Microsoft Graph (1)				...
User.Read	Delegated	Sign in and read user profile	-	...

Request API permissions

	protected content	your products
 Data Export Service for Microsoft Dynamics 365 Export data from Microsoft Dynamics CRM organization to an external destination	 Dynamics 365 Business Central Programmatic access to data and functionality in Dynamics 365 Business Central	 Dynamics CRM Access the capabilities of CRM business software and ERP systems
 Dynamics ERP Programmatic access to Dynamics ERP data	 Flow Service Embed flow templates and manage flows	 Intune Programmatic access to Intune data
 OneNote Create and manage notes, lists, pictures, files, and more in OneNote notebooks	 Power BI Service Programmatic access to Dashboard resources such as Datasets, Tables, and Rows in Power BI	 PowerApps Runtime Service Powerful data storage, modeling, security and integration capabilities
 SharePoint Interact remotely with SharePoint data	 Skype for Business Integrate real-time presence, secure messaging, calling, and conference capabilities	 Speech Create powerful speech-enabled features using speech to text and text to speech conversion
 Yammer Access resources in the Yammer web interface (e.g. messages, users, groups etc.)		

Supported legacy APIs

 Azure Active Directory Graph Programmatic access to directory data and objects	 Exchange A powerful, easy-to-use way to access and manipulate Exchange data
--	---

11

[← All APIs](#)

What type of permissions does your application require?

Your application needs to access the API as the signed-in user.

Your application runs as a background service or daemon without a signed-in user.

expand all

Admin Consent Required

Application (1)

Yes

Yes

> Device

▼ Directory (1)

Yes

Yes

> Domain

> Member

[Policy](#)

Discard

Dashboard > Default Directory | App registrations > DevOpsServicePrincipal | API permissions

DevOpsServicePrincipal | API permissions

- Search (Ctrl+J) << Refresh
- Overview
- Quickstart
- Manage
- Branding
- Authentication
- Certificates & secrets
- Token configuration (preview)
- API permissions
- Expose an API
- Owners
- Roles and administrators (Prev...
- Manifest
- Support + Troubleshooting
- Troubleshooting
- New support request

You are editing permission(s) to your application, users will have to consent even if they've already done so previously.

Configured permissions

Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. [Learn more about permissions and consent](#)

+ Add a permission Grant admin consent for Default Directory

API / Permissions name	Type	Description	Admin Consent Requ...	Status
Azure Active Directory Graph (2)				
Application.ReadWrite.OwnedBy	Application	Manage apps that this app creates or owns	Yes	Not granted for Default ...
Directory.Read.All	Application	Read directory data	Yes	Not granted for Default ...
Microsoft Graph (1)				
User.Read	Delegated	Sign in and read user profile	-	

Dashboard > Default Directory | App registrations > DevOpsServicePrincipal | API permissions

DevOpsServicePrincipal | API permissions

- Search (Ctrl+J) << Refresh
- Overview
- Quickstart
- Manage
- Branding
- Authentication
- Certificates & secrets
- Token configuration (preview)
- API permissions
- Expose an API
- Owners
- Roles and administrators (Prev...
- Manifest
- Support + Troubleshooting
- Troubleshooting
- New support request

Do you want to grant consent for the requested permissions for all accounts in Default Directory? This will update any existing admin consent records this application already has to match what is listed below.

Yes No

all the permissions the application needs. [Learn more about permissions and consent](#)

+ Add a permission Grant admin consent for Default Directory

API / Permissions name	Type	Description	Admin Consent Requ...	Status
Azure Active Directory Graph (2)				
Application.ReadWrite.OwnedBy	Application	Manage apps that this app creates or owns	Yes	Not granted for Default ...
Directory.Read.All	Application	Read directory data	Yes	Not granted for Default ...
Microsoft Graph (1)				
User.Read	Delegated	Sign in and read user profile	-	

Dashboard > Default Directory | App registrations > DevOpsServicePrincipal | API permissions

DevOpsServicePrincipal | API permissions

Search (Ctrl+/)
Refresh

Overview
Quickstart

Manage
Branding
Authentication
Certificates & secrets
Token configuration (preview)
API permissions
Expose an API
Owners
Roles and administrators (Previ...
Manifest
Support + Troubleshooting
Troubleshooting
New support request

Successfully granted admin consent for the requested permissions.

Configured permissions

Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. [Learn more about permissions and consent](#)

+ Add a permission Grant admin consent for Default Directory

API / Permissions name	Type	Description	Admin Consent Requir...	Status
▼ Azure Active Directory Graph (2)				...
Application.ReadWrite.OwnedBy	Application	Manage apps that this app creates or owns	Yes	<div> ✔ Granted for Default Dire... </div>
Directory.Read.All	Application	Read directory data	Yes	<div> ✔ Granted for Default Dire... </div>
▼ Microsoft Graph (1)				...
User.Read	Delegated	Sign in and read user profile	-	<div> ✔ Granted for Default Dire... </div>

6.2 Create an Azure Key Vault-backed secret scope

- Verify that you have Owner permission on the Azure Key Vault instance that you want to use to back the secret scope.
If you do not have a Key Vault instance, follow the instructions in [Quickstart: Create a Key Vault using the Azure portal](#).
- Go to `https://<your_azure_databricks_url>#secrets/createScope` (for example, `https://westus.azuredatabricks.net#secrets/createScope`).

Microsoft Azure

HomePage / Create Secret Scope

Create Secret Scope

[Cancel](#) [Create](#)

A store for secrets that is identified by a name and backed by a specific store type. [Learn more](#)

Scope Name ?

key-vault-secrets

Manage Principal ?

✓ Creator
All Users

Azure Key vault ?

DNS Name

https://databrickskv.vault.azure.net/

Resource ID

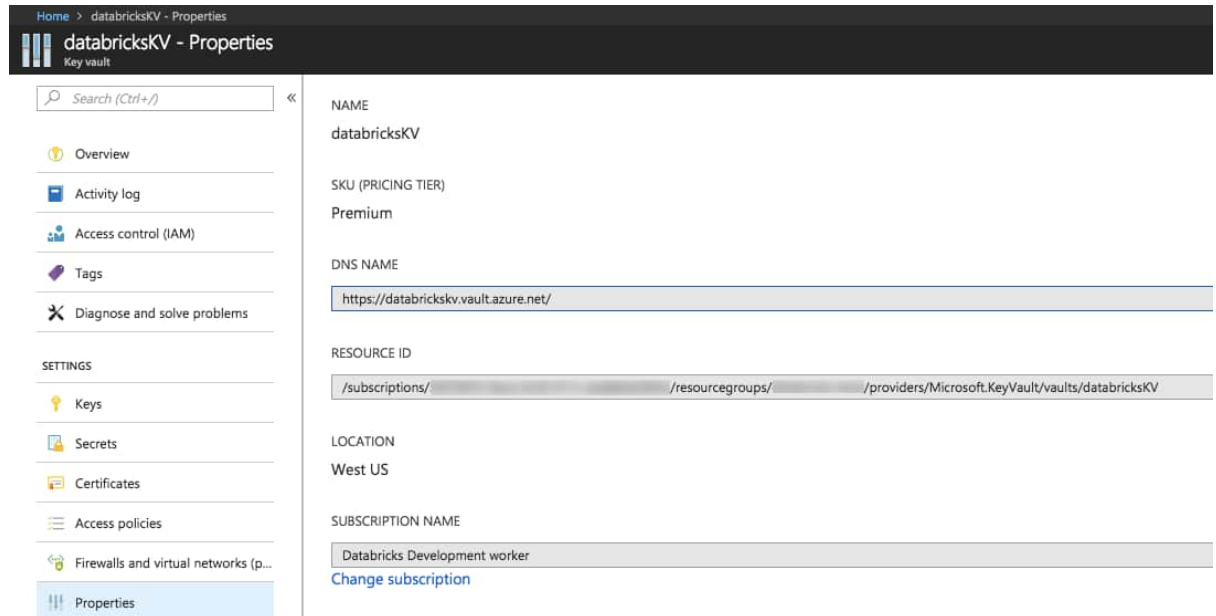
/subscriptions/c.../resourcegroups/databric

3. Enter the name of the secret scope. Secret scope names are case insensitive.
4. Use the **Manage Principal** drop-down to specify whether *All Users* have **MANAGE** permission for this secret scope or only the *Creator* of the secret scope (that is to say, you).
MANAGE permission allows users to read and write to this secret scope, and, in the case of accounts on the [Azure Databricks Premium Plan](#), to change permissions for the scope. Your account must have the [Azure Databricks Premium Plan](#) for you to be able to select *Creator*. This is the recommended approach: grant MANAGE permission to the *Creator* when you create the secret scope, and then assign more granular access permissions after you have tested the scope. For an example workflow, see [Secret workflow example](#).
If your account has the Standard Plan, you must set the MANAGE permission to the "All Users" group. If you select *Creator* here, you will see an error message when you try to save the scope.
For more information about the MANAGE permission, see [Secret access control](#).
5. Enter the **DNS Name** (for example, `https://databrickskv.vault.azure.net/`) and **Resource ID**, for example:

Copy

```
/subscriptions/xxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourcegroups/databricks-
rg/providers/Microsoft.KeyVault/vaults/databricksKV
```

These properties are available from the **Properties** tab of an Azure Key Vault in your Azure portal.



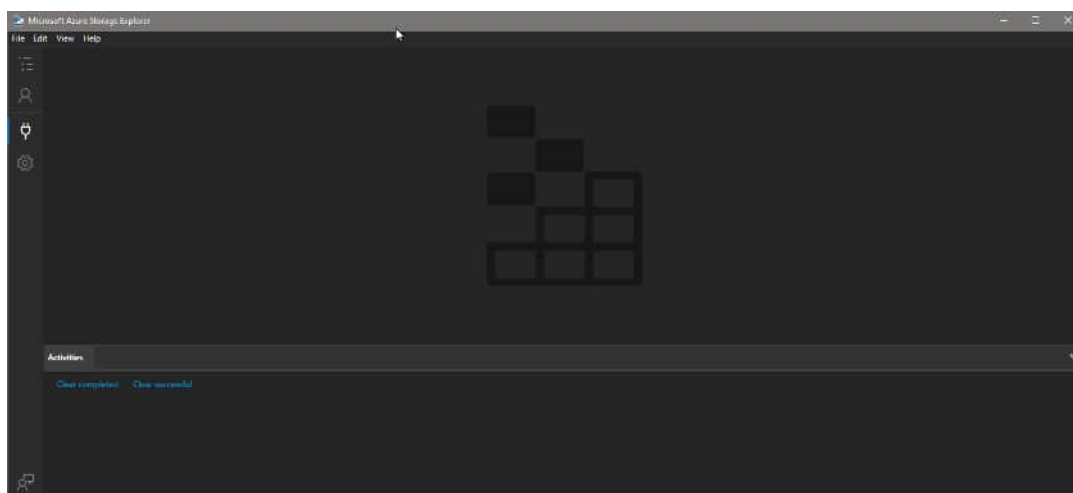
6. Click the **Create** button.
7. Use the [Databricks CLI](#) `databricks secrets list-scopes` command to verify that the scope was created successfully.

6.3 Datalake Access guide

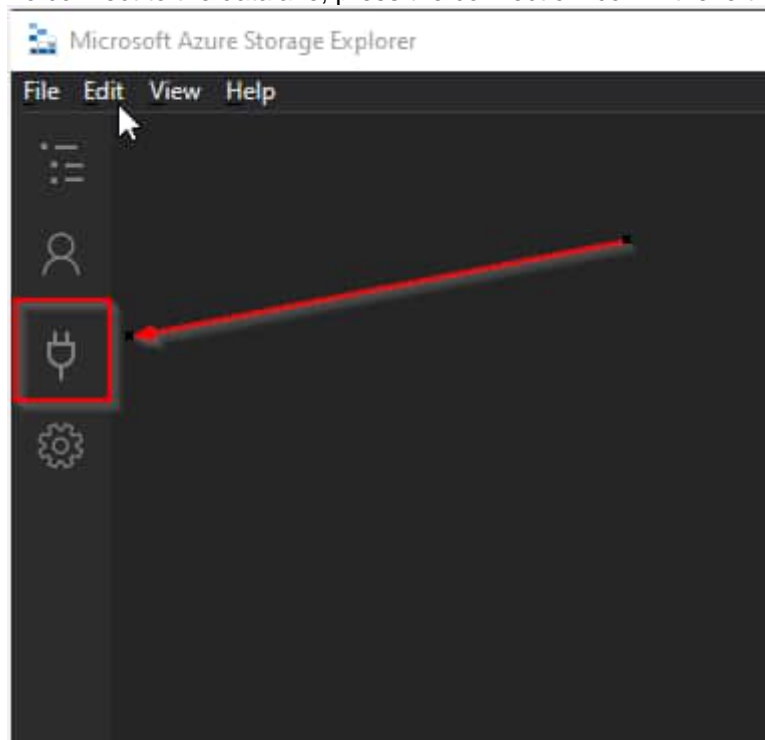
6.3.1 Azure storage explorer

The data lake is what is used in the backend to store all the data from the FTP servers before it is merged to the SQL database. For manual interventions, files can be loaded onto this datalake so they will be processed in the next run. For easy access to the data lake, the Azure storage explorer tool is used. The tool can be downloaded from the microsoft website: <https://azure.microsoft.com/en-us/features/storage-explorer/#features>

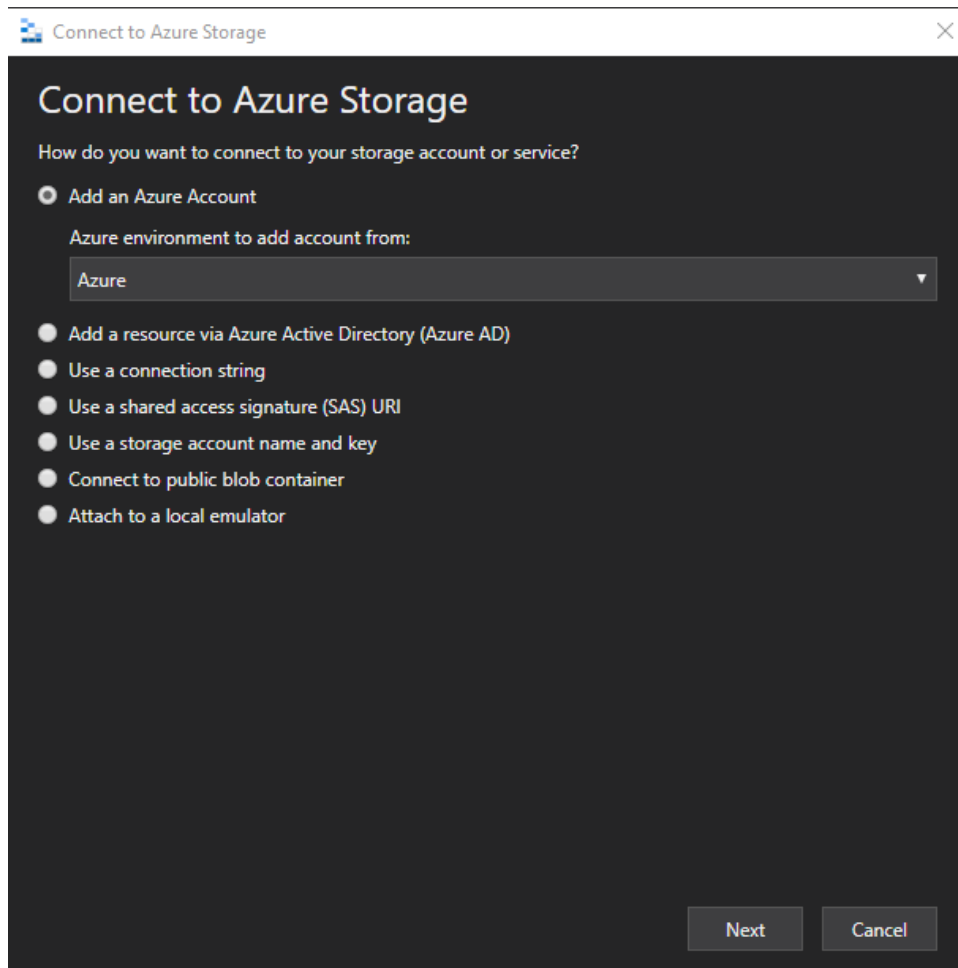
Once the installation is completed, open the Azure Storage Explorer. The following screen will be shown:



To connect to the datalake, press the connection icon in the left menu bar:



The connect dialog will appear:



Choose the option ***Use a storage account name and key*** and click next.

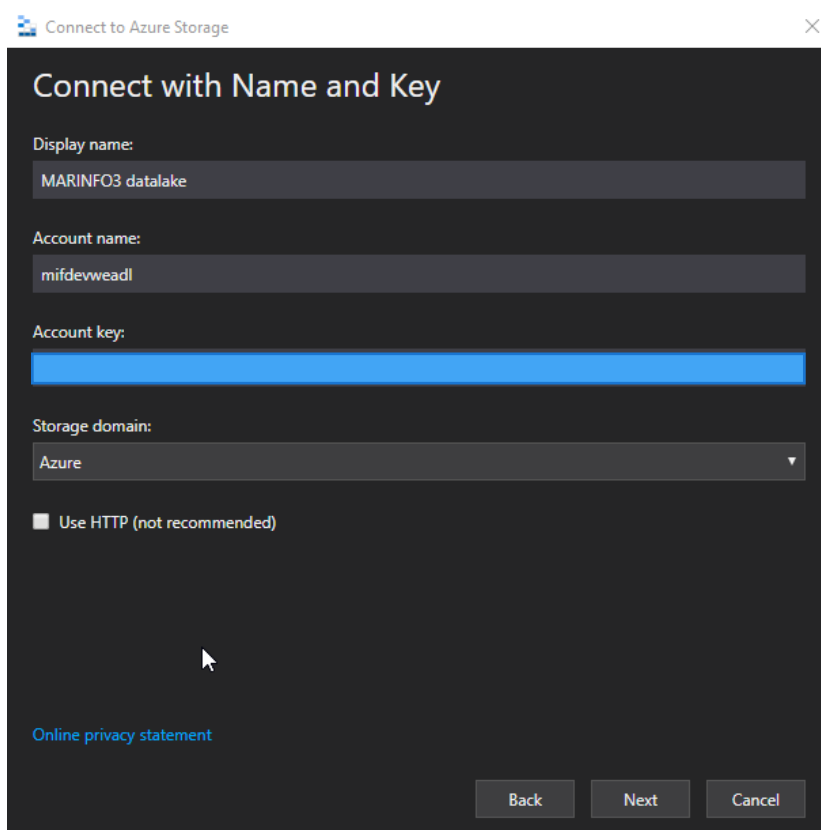
Fill in the fields:

Display name: MARINFO4 datalake (this name can be chosen freely)

Account name: marinfoprodwedl

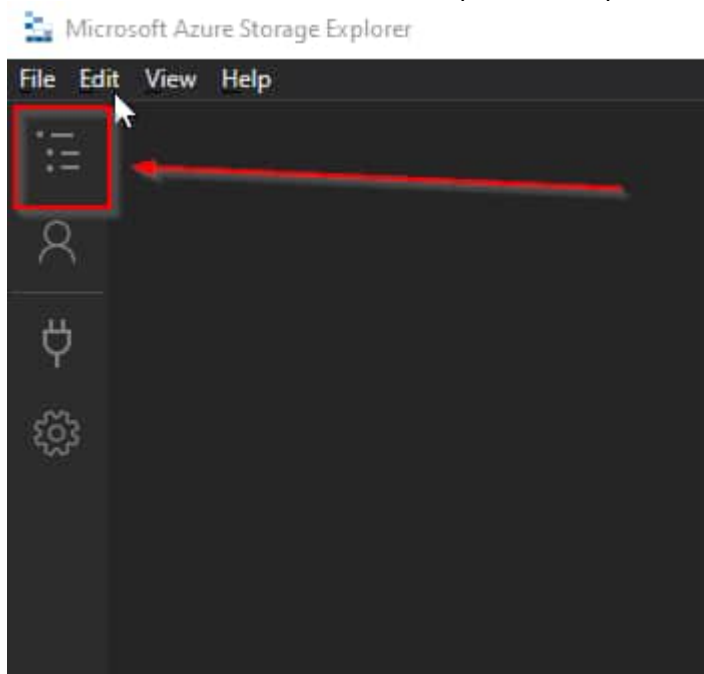
Access key:

Storage domain: Azure

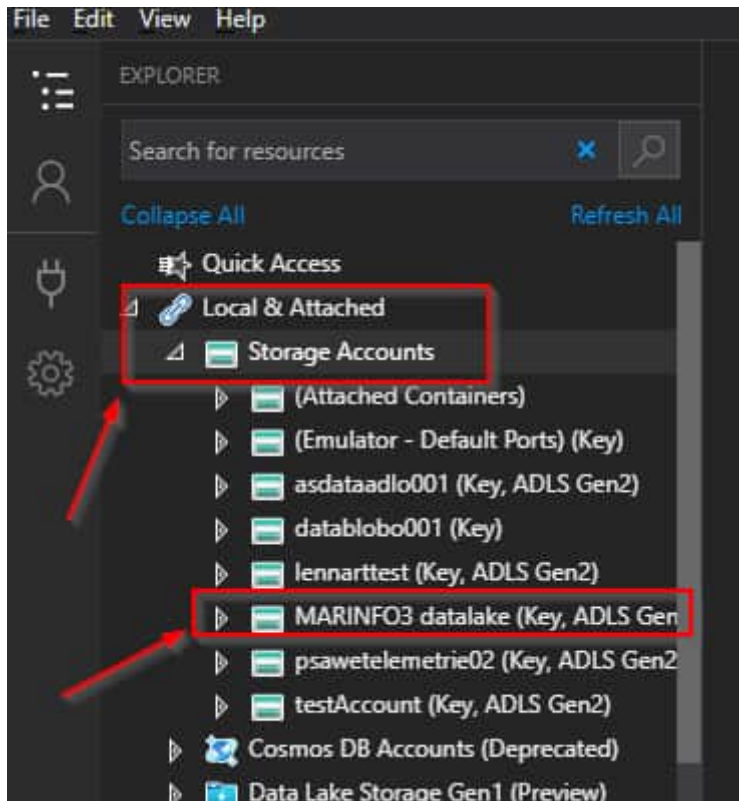


Click next and afterwards click connect. The connection will be added to the Azure Storage explorer.

To view the datalake and its contents, press the explorer button in the left sidebar:

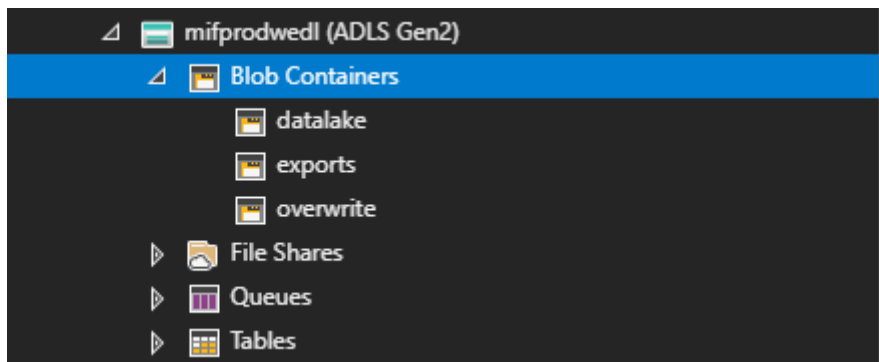


All your connections are shown here. The newly added connection will be visible under local & attached > Storage Accounts



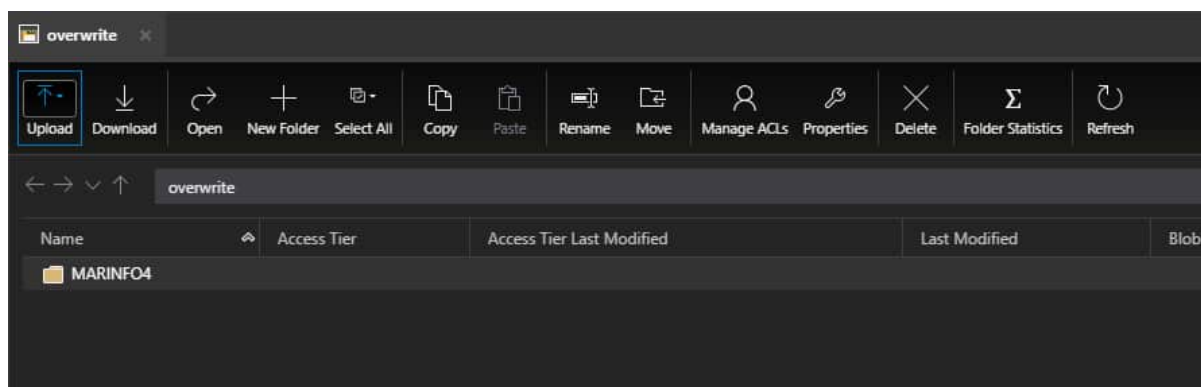
6.4 Adding manual files to the datalake

To add manual files to the data lake, a container has been made where manual files can be dropped to be processed in the next run. Under the MARINFO4 connection in Azure Storage explorer, there will be a *Blob Container* tab. Open the tab and two containers will show up (datalake and overwrite):



The **datalake container** is used by the backend. The reports will be saved in this container together with the extracted data from the zip files, partitioned by execution ID and load ID. Nothing should be changed to this container.

The **overwrite container** is used for manual files. Open the container by double clicking it. The container will open and contains one folder: MARINFO4



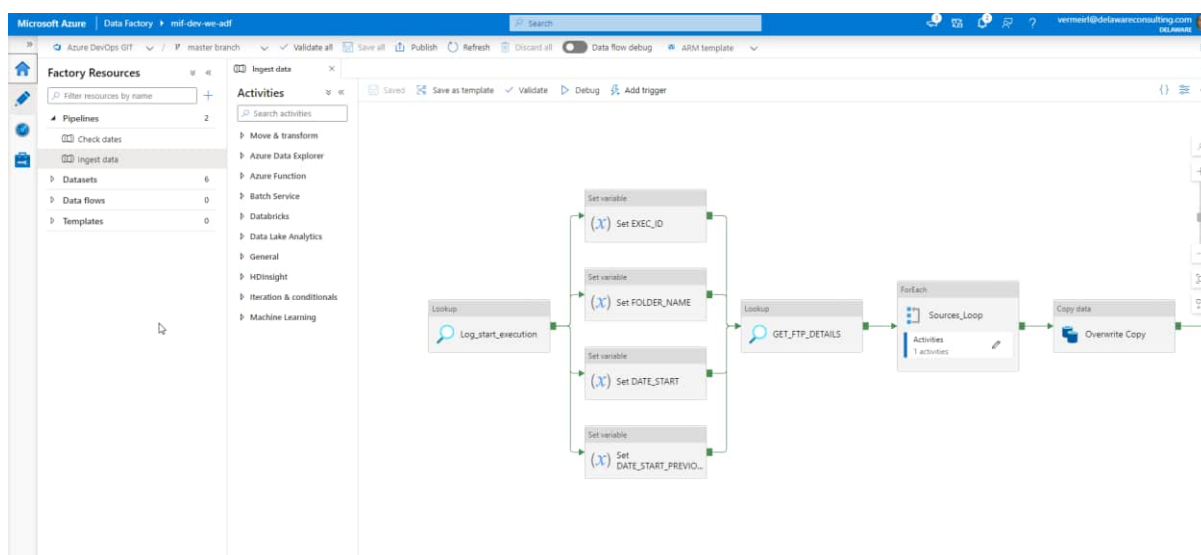
Open the folder by double clicking it. Manual files can be uploaded in this folder. This can be done by dragging and dropping zip files from your computer or by pressing the *upload* button at the top.

Note: each run, the new files are taken from this folder. Whether a file is new or not is based on the last modified date. If this date is before the date of the last successful run, the file will not be processed again. So old files do not have to be removed (they can however).

6.5 Data Factory guide

6.5.1 General information

The data factory is the orchestration tool for the runs. A run can be started and it will complete the full flow (get the data, process it, update logging database for PowerBI report, send emails). The general interface for this tool is visible in the picture below:

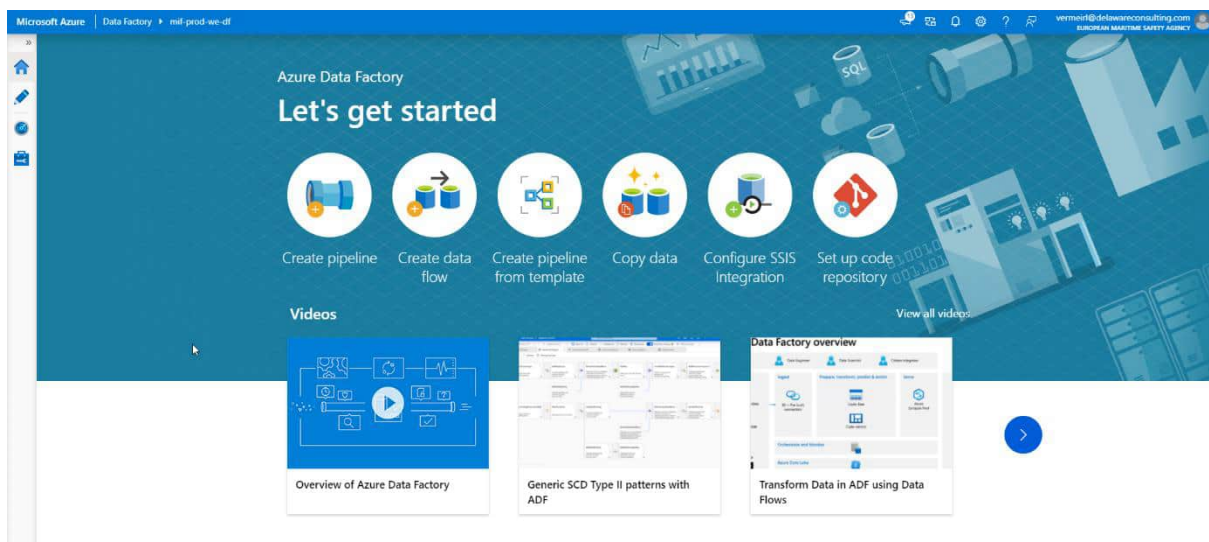


EMSA users that have access to this interface, will be allowed to manually trigger a run of the full flow. To open the data factory interface, go to the following link: <https://adf.azure.com/en-us/home?factory=%2Fsubscriptions%2F26a4f5fc-013a-4eed-b1c5-48384da8c385%2FresourceGroups%2Fmif-prod-we-rq%2Fproviders%2FMicrosoft.DataFactory%2Ffactories%2Fmif-prod-we-df>

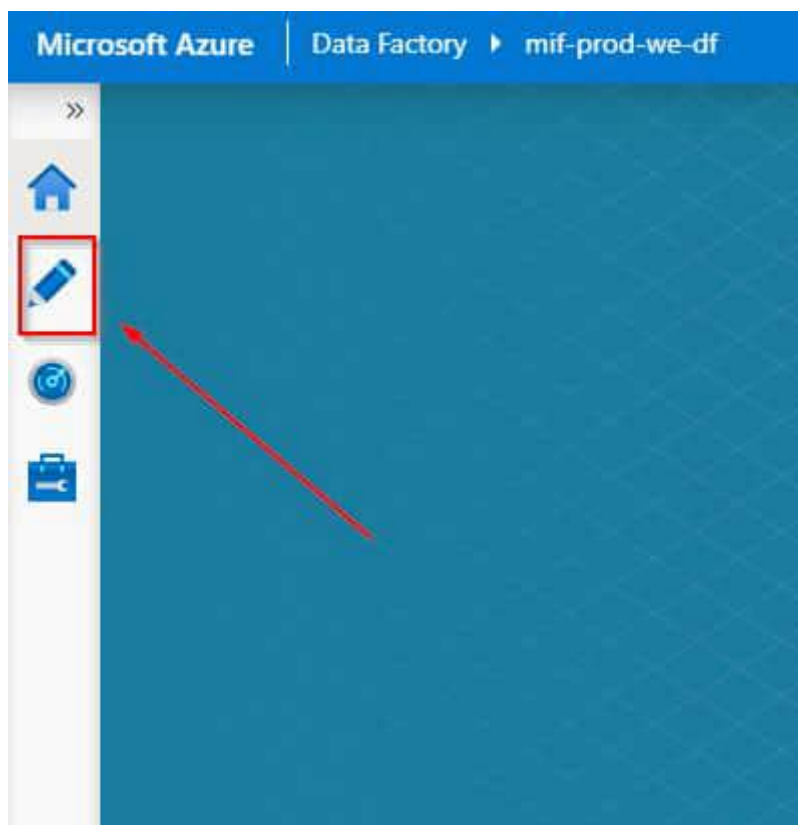
For the flow, a schedule can be created when the flow should run. There are various possibilities. At this moment, a schedule is created for the flow to run twice each week on mon at 21h lisbon time and tue at 21h lisbon time.

6.5.2 Manual data factory trigger

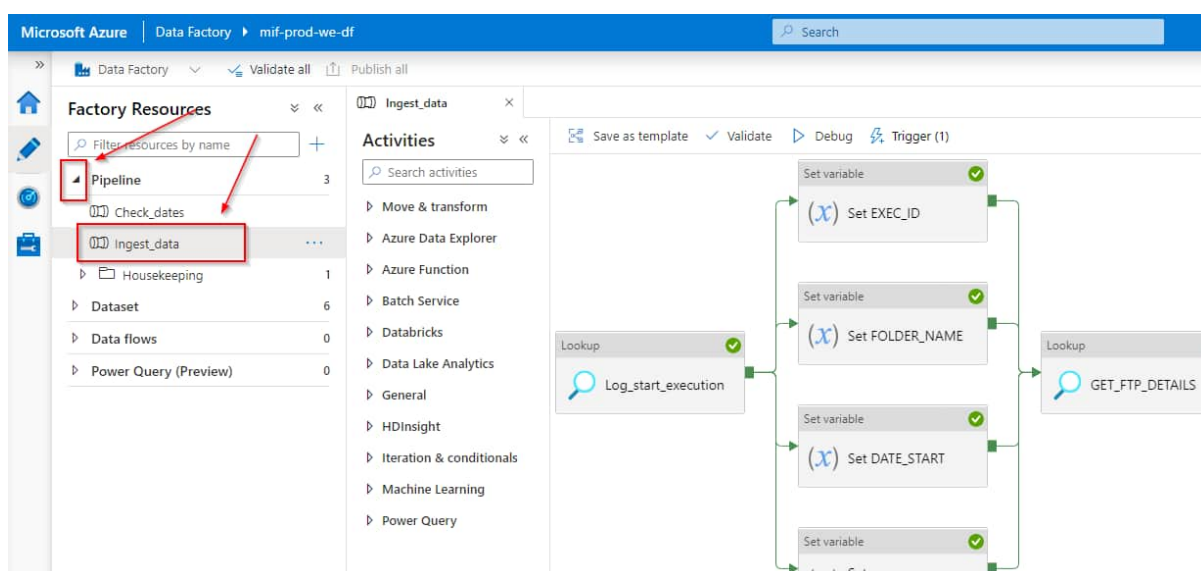
For EMSA users, to be able to trigger the flow when they want, they can login to data factory and do a manual trigger from there. When clicking on the link provided above, the following home page appears:



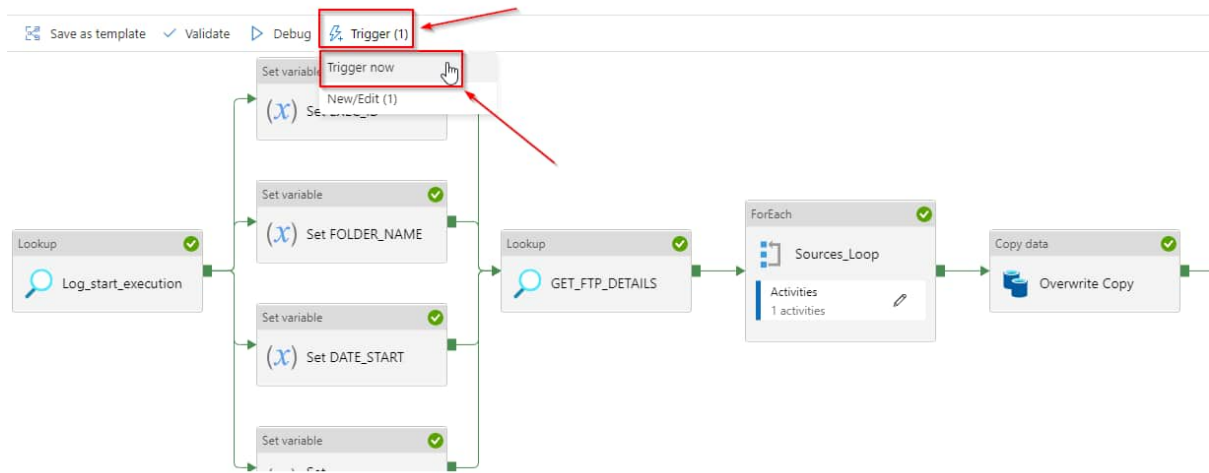
Click on the pencil at the right bar:



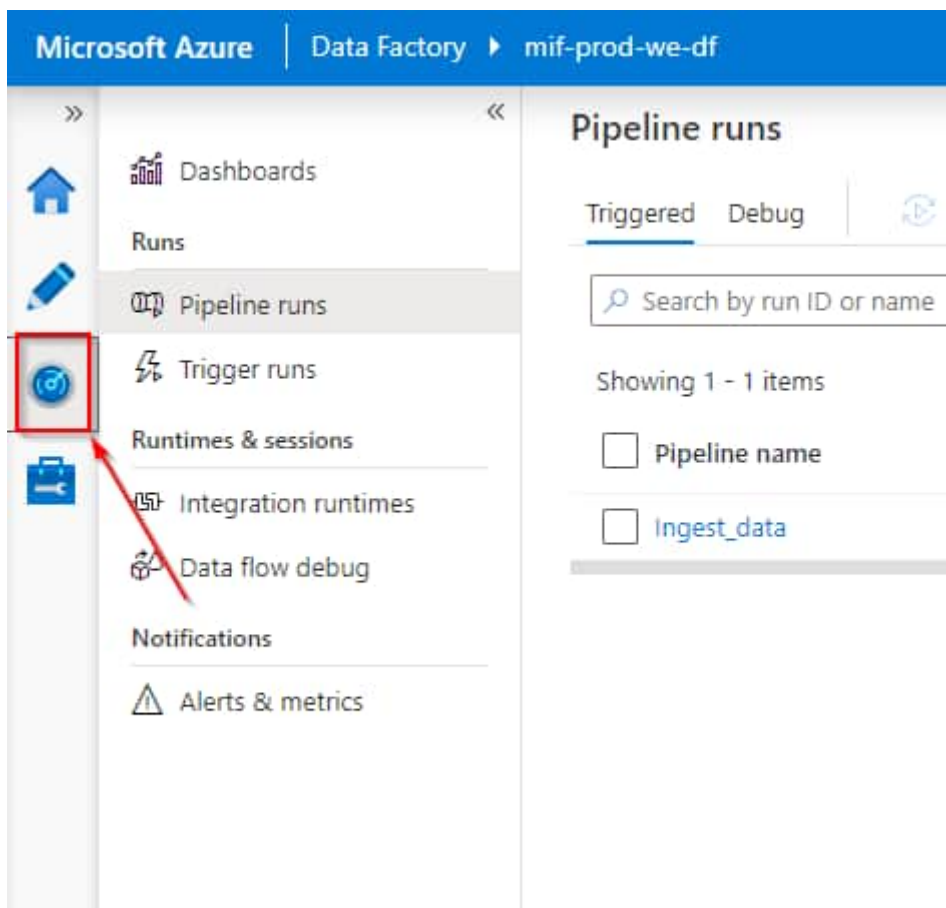
The data factory GUI will open. Now, open the pipelines tab and click on the “Ingest_data” pipeline:



The pipeline will open and at the top, you will find the trigger button. Press this button and press “Trigger Now” and this will start the manual run of the pipeline.



The progress can then be followed up in the monitoring tab of data factory:

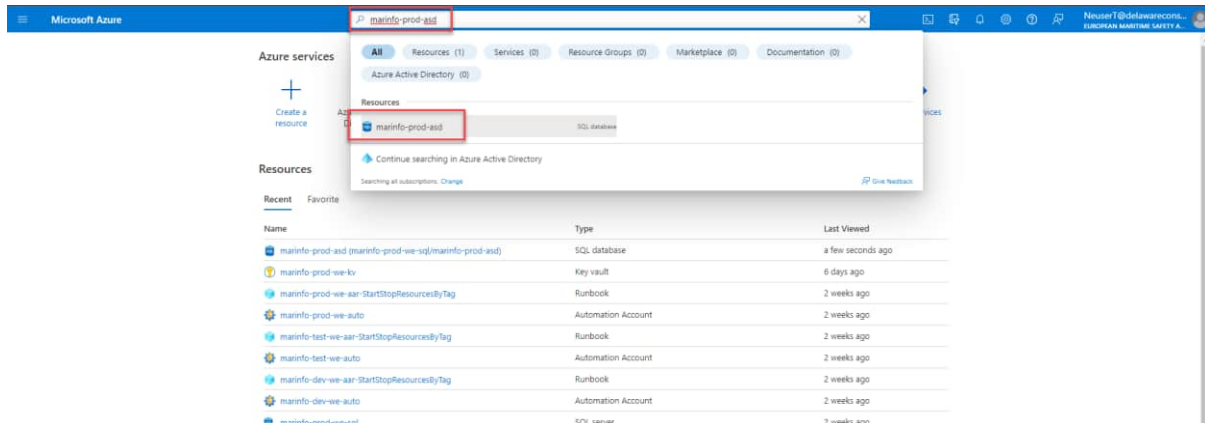


Note that it might take some time, depending on the amount of files that need to be processed, before the run completes. It is possible to check the current status by querying the logging tables in the MARINFO database.

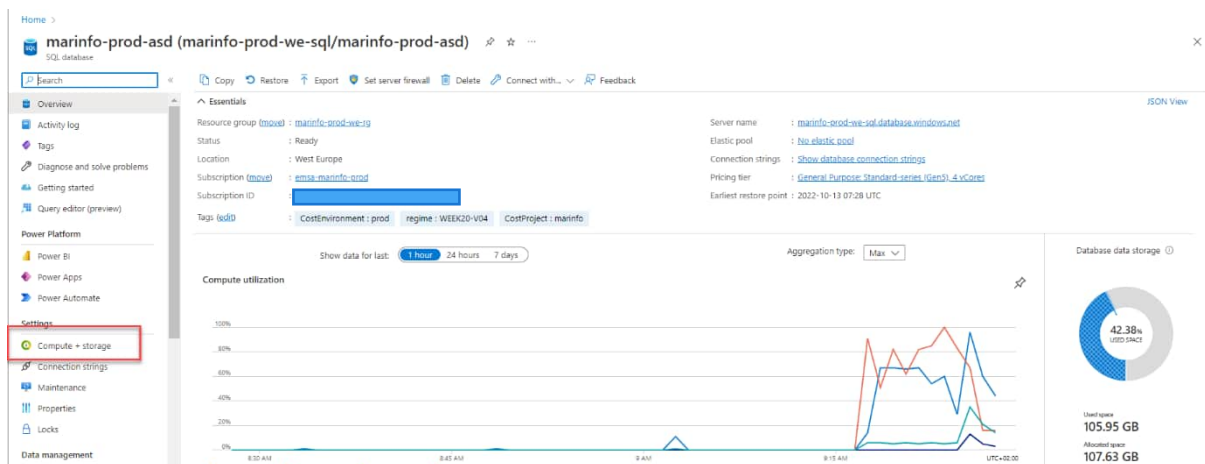
Note: if the contractor added new files to the FTP servers since last run, they will also be processed when triggering the data factory flow.

6.6 Database upgrade

Step 1: Log into the Azure portal and select the correct resource: marinfo-prod-asd



Step 2: Select “Compute + storage”



Step 3: Adapt the slider, or enter a higher number in the input box.

The screenshot shows the Azure portal interface for the 'marinfo-prod-sql' database. The left sidebar contains navigation links for Overview, Activity log, Tags, Diagnose and solve problems, Getting started, Query editor (preview), Power Platform, Power BI, Power Apps, Power Automate, Settings, Connection strings, Maintenance, Properties, Locks, Data management, Replicas, Sync to other databases, and Integrations. The main content area is titled 'Service and compute tier' and includes sections for Service tier (General Purpose), Compute tier (Provisioned), Compute Hardware (Standard-series (Gen5)), and Save money (No). The 'vCores' slider is highlighted with a red box and set to 4. The 'Data max size (GB)' is set to 250. The '75 GB LOG SPACE ALLOCATED' bar is visible at the bottom. On the right, there is a 'Database utilization' graph showing CPU percentage (Max) and a 'Cost summary' table.

Cost summary	
General Purpose (GP_Gen5_4)	
Cost per vCore (in EUR)	139.94
vCores selected	x 4
Cost per GB (in EUR)	0.10
Max storage selected (in GB)	x 325
ESTIMATED COST / MONTH	591.61 EUR

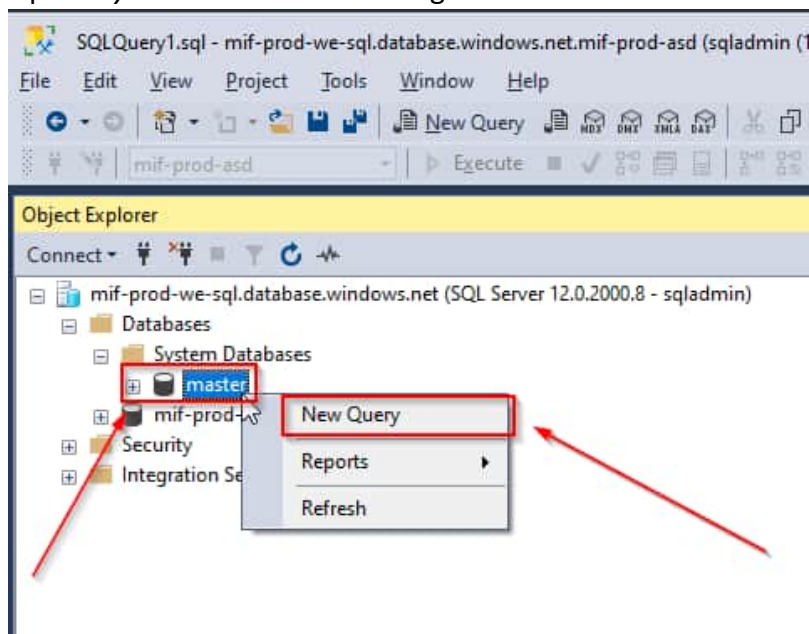
Note: Every night at 8pm CET, the number of VCores is reset to 4.

6.7 Create read-only user in MARINFO database (using SQL Server Management Studio):

1. Login to the database on which to create the read-only user (for this example the PRD database is used) via SSMS:

The screenshot shows the 'Connect to Server' dialog box in SQL Server Management Studio. The 'Server type' is set to 'Database Engine'. The 'Server name' is 'mif-prod-we-sql.database.windows.net'. The 'Authentication' is set to 'SQL Server Authentication'. The 'Login' and 'Password' fields are present, and the 'Remember password' checkbox is checked. The 'Connect' button is highlighted.

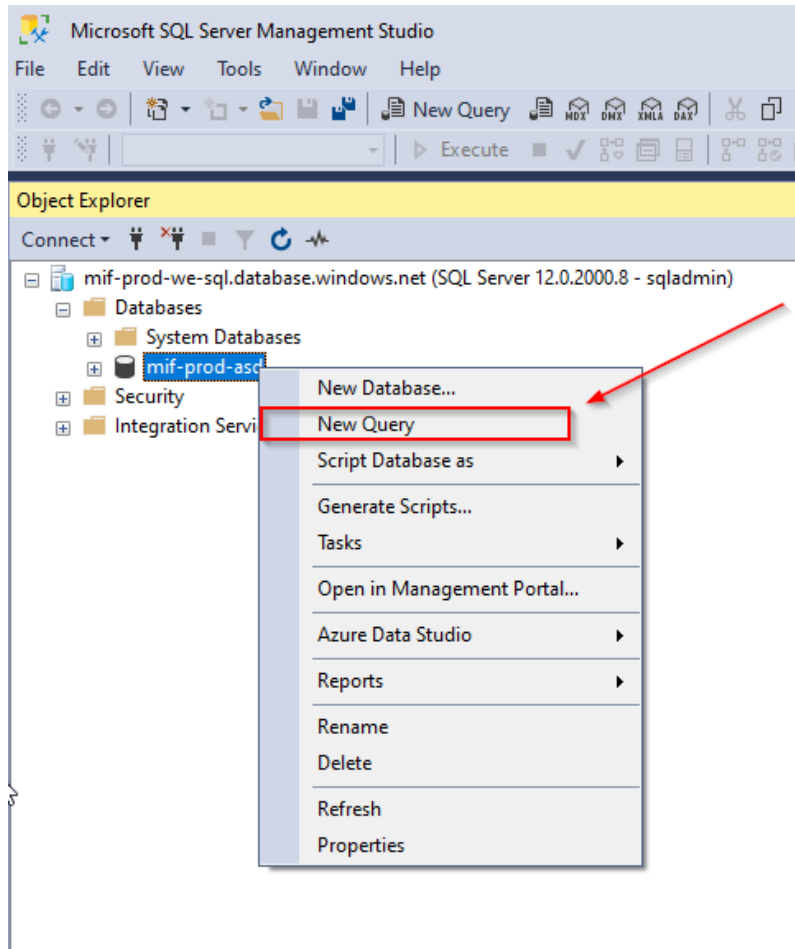
2. Open “System Databases” and right click the “master” database. Click “New Query”:



3. Type in the following query in the editor and change <user> and <password> with the username and password respectively. These can be chosen freely. Do note to keep track of the password:

```
CREATE LOGIN [<user>] WITH PASSWORD=N'<password>'
GO
```

4. Next, right click the database you want to grant access to and select “New Query”:



Type in the following query and change <user> with the username created in the previous steps:

```
CREATE USER [<user>] FOR LOGIN [<user>]
GO
EXEC sp_addrolemember db_datareader, <user>
GO
```

6.8 Add IP range to access MARINFO database (using Azure portal):

1. Login to Azure portal (portal.azure.com)
2. Go to SQL servers, and select the server on the correct environment:

[Home](#) >

SQL servers ...

European Maritime Safety Agency

[+ Create](#) [⚙️ Manage view](#) [↺ Refresh](#) [⬇ Export to CSV](#)

Filter for any field...

Subscription == 3 of 6 selected

Showing 1 to 3 of 3 records.

☐ Name ↑↓☐  marinfo-dev-we-sql☐  marinfo-prod-we-sql☐  marinfo-test-we-sql

3. Select “Show firewall settings”

^ Essentials

Resource group ([move](#))
[marinfo-dev-we-rg](#)

Status
Available

Location
West Europe

Subscription ([move](#))
[emsa-marinfo-dev](#)

Subscription ID

Server admin
sqladmin

Firewalls and virtual networks
[Show firewall settings](#)

Active Directory admin
[neusert@delawareconsulting.com](#)

Server name
marinfo-dev-we-sql.database.windows.net

4. Add a “Rule Name”, “Start IP” and “End IP”, and click on “Save”

[Save](#) [✕ Discard](#) [+ Add client IP](#)

☐ Deny public network access

Minimum TLS Version ⓘ
1.0 1.1 **1.2**

Connection Policy ⓘ
Default Proxy Redirect

Allow Azure services and resources to access this server ⓘ
Yes No

Client IP address 178.118.172.2

Rule name	Start IP	End IP
<input type="text"/>	<input type="text"/>	<input type="text"/> ...
<div></div> ...		

European Maritime Safety Agency

Praça Europa 4
1249-206 Lisbon, Portugal
Tel +351 21 1209 200
Fax +351 21 1209 210
emsa.europa.eu

