

Optimaliseren van gegevensverwerking in Microsoft Azure: Een vergelijkende analyse tussen Azure Data Factory en Azure Databricks voor het implementeren van Extractie, Transformatie en Laden (ETL).

Optionele ondertitel.

Lievens Loeka.

Scriptie voorgedragen tot het bekomen van de graad van
Professionele bachelor in de toegepaste informatica

Promotor: Dhr. Bosteels Gertjan

Co-promotor: Dhr. Van Damme Koen

Academiejaar: 2023–2024

Eerste examenperiode

Departement IT en Digitale Innovatie .

**HO
GENT**

Woord vooraf

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Samenvatting

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Inhoudsopgave

Lijst van figuren	vii
1 Inleiding	1
1.1 Probleemstelling	1
1.2 Onderzoeksvraag	2
1.3 Onderzoeksdoelstelling	2
1.4 Opzet van deze bachelorproef	2
2 Stand van zaken	3
2.1 Wat zijn ETL's of ELT's?	3
2.2 Welke soorten ETL tools bestaan er?	3
2.3 Populairste cloud-based ETL tools	4
2.4 Apache Spark	4
2.5 Continuous Integration (CI) en Continuous Deployment (CD)	5
2.6 Microsoft Azure	6
2.6.1 Subscription	7
2.6.2 Data Lake	7
2.6.3 Key Vault	8
2.6.4 App Registration	8
2.6.5 Cost Management	8
2.6.6 Azure Data Factory (ADF)	9
2.6.7 Azure Databricks	9
3 Methodologie	11
3.1 Literatuuronderzoek	11
3.2 Vergelijkingscriteria	11
3.3 Proof-of-concepts	11
3.4 Uitvoeren van pipelines en verzamelen van resultaten	12
3.5 Conclusie	12
4 Vergelijkingscriteria	13
4.1 Makkelijk meetbaar	13
4.2 Verschillen aantoonbaar aan de hand van een voorbeeld	13
4.3 Eigen ervaring	13

5	Proof of Concepts	14
5.1	Azure Data Factory (ADF)	14
5.1.1	Opzet van resources	14
5.1.2	Collaboration en source control	15
5.1.3	IDE integratie	16
5.1.4	Infrastructure as code (IaC)	17
5.1.5	Ophalen van data uit Azure Data Lake	17
5.1.6	Mogelijkheid tot debuggen	23
5.1.7	Belangrijkste transformaties	24
5.1.8	Schrijven van data naar Azure Data Lake	40
5.2	Azure Databricks	40
5.2.1	Opzet van resources	40
5.2.2	Collaboration en source control	43
5.2.3	IDE integratie	47
5.2.4	Infrastructure as code (IaC)	47
5.2.5	Ophalen van data uit Azure Data Lake	48
5.2.6	Mogelijkheid tot debuggen	56
5.2.7	Belangrijkste transformaties	56
5.2.8	Schrijven van data naar Azure Data Lake	63
5.3	Validatie van output	64
6	Uitvoeren van pipelines en verzamelen van resultaten	65
6.1	Azure Data Factory	66
6.2	Azure Databricks	67
6.3	Grafieken	69
6.3.1	Kosten	69
6.3.2	Performantie	70
7	Conclusie	72
7.1	Aan de hand van vergelijkingscriteria	72
7.2	Eigen ervaring	73
A	Onderzoeksvoorstel	74
A.1	Inleiding	74
A.2	Literatuurstudie	75
A.3	Methodologie	76
A.4	Verwacht resultaat, conclusie	77
	Bibliografie	78

Lijst van figuren

2.1	Spark stack (Holden Karau & Zaharia, 2015)	5
2.2	Verschillen tussen IaaS, PaaS en SaaS (Stoenescu, 2021)	6
5.1	Aanmaken van Azure Data Factory	14
5.2	Toewijzen van Data Factory Contributor Role	15
5.3	Configuratie van Azure DevOps in Azure Data Factory	16
5.4	Configuratie van source transformation	18
5.5	Configuratie van Linked Service	19
5.6	Configuratie van source options	20
5.7	Configuratie van source options	21
5.8	Importeren van schema voor data flow source	21
5.9	Voorbeeld geïmporteerd schema van data flow source	22
5.10	Data preview in data flow	23
5.11	Join van de tabel "new_year" op de tabel "new_syndicalpremiumrequest"	24
5.12	Derive "EntryYear", "CalendarYear" en "RefYear" op de tabel "new_syndicalpremiumrequest"	
5.13	Hernoemen van kolommen op "new_syndicalpremiumrequest" en splitting in twee aparte branches	26
5.14	Inner join van "new_organizationyear" op de tabel "new_syndicalpremiumrequest"	26
5.15	Selecteren en hernoemen van kolommen op de tabel "new_syndicalpremiumrequest"	27
5.16	Custom (cross) join van "new_membership" op de tabel "new_syndicalpremiumrequest"	28
5.17	Selecteren en hernoemen van kolommen op de tabel "new_syndicalpremiumrequest"	29
5.18	Union van twee branches	30
5.19	Group by "spr_id", "Group" en "RefYear" in "new_syndicalpremiumrequest"	31
5.20	Group by "spr_id", "Group" en "RefYear" in "new_syndicalpremiumrequest"	31
5.21	Joinen van de tabel "new_membership" op de tabel "new_syndicalpremiumrequest"	32
5.22	Selecteren van de juiste kolommen in "new_syndicalpremiumrequest"	33
5.23	Herhalen van transformaties	33
5.24	Herhalen van transformaties	33
5.25	Bepalen van "Gr" in "new_syndicalpremiumrequest"	34
5.26	Conditional split op de tabel "new_syndicalpremiumrequest"	35
5.27	Derive "AntheaNumber" en "MembershipNumber" op de tabel "new_syndicalpremiumrequest"	
5.28	Join van de tabel "new_membership" op de tabel "new_syndicalpremiumrequest" aan de hand van "new_groupid", "new_personid" en "CalendarYear"	36
5.29	Select op de tabel "new_syndicalpremiumrequest"	36

5.30 Join van de tabel “new_membership” op de tabel “new_syndicalpremiumrequest” aan de hand van “new_groupid”, “new_personid” en “EntryYear”	37
5.31 Join van de tabel “new_membership” op de tabel “new_syndicalpremiumrequest” aan de hand van “new_groupid”, “new_personid” en “RefYear”	37
5.32 Derive van kolommen “AntheaNumber” en “MembershipNumber” op de tabel “new_syndicalpremiumrequest”	38
5.33 Verwijderen van onnodige kolommen op de tabel “new_syndicalpremiumrequest”	39
5.34 Union van twee streams	39
5.35 Sink in Azure Data Factory	40
5.36 Configuratie van Azure Databricks	41
5.37 Configuratie van compute resource	42
5.38 Installatie van “spark-cdm-connector”	43
5.39 Gebruikers toevoegen/verwijderen in Azure Databricks	43
5.40 Rechten wijzigen van gebruikers in Azure Databricks	44
5.41 Groepen toevoegen of verwijderen in Azure Databricks	44
5.42 Clone Git repository in Databricks folders	45
5.43 Aanmaken van een notebook in databricks	45
5.44 Commit en push in Databricks	45
5.45 Commit en push in Databricks	46
5.46 Aanmaken van app registration	48
5.47 Aanmaken client secret	49
5.48 Role assignment toevoegen aan storage account	49
5.49 Job function role kiezen voor role assignment	49
5.50 Members kiezen voor role assignment	50
5.51 Aanmaken van key vault	51
5.52 Permission model van key vault	52
5.53 Networking configuratie van key vault	52
5.54 Toevoegen van een secret in key vault	53
5.55 Opzoeken van properties van key vault	53
5.56 Aanmaken van secret scope in databricks	54
5.57 Debuggen in Azure Databricks	56
5.58 Voorbeeld van hoe de functie explode werkt.	59
6.1 Data flow runtimes voor het uitvoeren van de Azure Data Factory pipeline.	65
6.2 Clusters voor het uitvoeren van de Azure Databricks pipeline.	65
6.3 Consumption voor het uitvoeren van de pipeline met 4 Worker Cores + 4 Driver Cores	66
6.4 Consumption voor het uitvoeren van de pipeline met 8 Worker Cores + 8 Driver Cores	66
6.5 Kost en tijd voor het uitvoeren van de pipeline met 4 Worker Cores + 4 Driver Cores.	67

6.6	Kost en tijd voor het uitvoeren van de pipeline met 8 Worker Cores + 8 Driver Cores.	67
6.7	Kost en tijd voor het uitvoeren van de pipeline met 4 Worker Cores + 4 Driver Cores.	68
6.8	Kost en tijd voor het uitvoeren van de pipeline met 8 Worker Cores + 8 Driver Cores.	68
6.9	Prijzen voor pipeline met 4 Worker Cores + 4 Driver Cores	69
6.10	Prijzen voor pipeline met 8 Worker Cores + 8 Driver Cores	69
6.11	Uitvoeringstijden voor pipeline met 4 Worker Cores + 4 Driver Cores . .	70
6.12	Uitvoeringstijden voor pipeline met 8 Worker Cores + 8 Driver Cores . .	70
6.13	Cluster startup tijden voor pipeline met 4 Worker Cores + 4 Driver Cores	71
6.14	Cluster startup tijden voor pipeline met 8 Worker Cores + 8 Driver Cores	71

1

Inleiding

De inleiding moet de lezer net genoeg informatie verschaffen om het onderwerp te begrijpen en in te zien waarom de onderzoeksvraag de moeite waard is om te onderzoeken. In de inleiding ga je literatuurverwijzingen beperken, zodat de tekst vlot leesbaar blijft. Je kan de inleiding verder onderverdelen in secties als dit de tekst verduidelijkt. Zaken die aan bod kunnen komen in de inleiding (Pollefliet, 2011):

- context, achtergrond
- afbakenen van het onderwerp
- verantwoording van het onderwerp, methodologie
- probleemstelling
- onderzoeksdoelstelling
- onderzoeksvraag
- ...

1.1. Probleemstelling

Uit je probleemstelling moet duidelijk zijn dat je onderzoek een meerwaarde heeft voor een concrete doelgroep. De doelgroep moet goed gedefinieerd en afgelijnd zijn. Doelgroepen als “bedrijven,” “KMO’s”, systeembeheerders, enz. zijn nog te vaag. Als je een lijstje kan maken van de personen/organisaties die een meerwaarde zullen vinden in deze bachelorproef (dit is eigenlijk je steekproefkader), dan is dat een indicatie dat de doelgroep goed gedefinieerd is. Dit kan een enkel bedrijf zijn of zelfs één persoon (je co-promotor/opdrachtgever).

1.2. Onderzoeksvraag

Wees zo concreet mogelijk bij het formuleren van je onderzoeksvraag. Een onderzoeksvraag is trouwens iets waar nog niemand op dit moment een antwoord heeft (voor zover je kan nagaan). Het opzoeken van bestaande informatie (bv. “welke tools bestaan er voor deze toepassing?”) is dus geen onderzoeksvraag. Je kan de onderzoeksvraag verder specificeren in deelvragen. Bv. als je onderzoek gaat over performantiemetingen, dan

1.3. Onderzoeksdoelstelling

Wat is het beoogde resultaat van je bachelorproef? Wat zijn de criteria voor succes? Beschrijf die zo concreet mogelijk. Gaat het bv. om een proof-of-concept, een prototype, een verslag met aanbevelingen, een vergelijkende studie, enz.

1.4. Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk 7, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2

Stand van zaken

Bedrijven slaan veel data op. Doordat deze data nuttig kan zijn voor het identificeren van nieuwe kansen is het belangrijk om deze data klaar te maken voor business analytics. Dit is het proces van verzamelen, organiseren, analyseren en interpreteren van gegevens om inzichten te krijgen. Er kan bijvoorbeeld gekeken worden naar klantgegevens om zo patronen en trends te vinden in het gedrag van de klant. (J., [2023](#))

Doordat bedrijven vaak werken met veel verschillende soorten data dat op verschillende plekken opgeslaan wordt, is het vaak belangrijk dat deze data eerst opgekuist, getransformeerd en georganiseerd moet worden. Dit is waarbij het implementeren van ETL's en ELT's van pas komt. (Inmon, [2023](#))

2.1. Wat zijn ETL's of ELT's?

ETL's en ELT's zijn processen die organisaties gebruiken voor het verzamelen en samenvoegen van data uit meerdere bronnen. Bij ETL's wordt de data getransformeerd voor het naar de doelopslagplaats geladen wordt, terwijl dit bij ELT's pas achteraf gebeurt. Daardoor staat ETL voor Extract, Transform and Load en ELT voor Extract, Load and Transform. (Bartley, [2023](#))

2.2. Welke soorten ETL tools bestaan er?

Er bestaan verschillende soorten ETL tools. Zo zijn er de cloud-based ETL tools. Deze worden gehost op cloud infrastructure, zijn zeer schaalbaar en bieden pay-as-you-go prijs modellen aan. (Ethan, [2024](#))

Daarnaast zijn er ook on-premises ETL tools. Deze worden gehost op de infrastructuur van het bedrijf waardoor het bedrijf er de volledige controle over heeft. (Ethan, [2024](#))

Afhankelijk van wat men nodig heeft kan er ook gekozen worden voor hybrid ETL

tools. Dit is een combinatie van het gebruik van cloud-based tools met het gebruik van on-premises tools. (Ethan, 2024)

Ten slotte zijn er ook open source ETL tools. Dit zijn gratis ETL tools. Voorbeelden hiervan zijn Portable, Apache NiFi, AWS Glue, Airbyte en Informatica. (Ethan, 2024)

2.3. Populairste cloud-based ETL tools

Zoals te zien in de enquête van Vines en Tanasescu (2023) is Microsoft Azure, gevolgd door Amazon Web Services (AWS) en Google Cloud Services, de populairste cloud provider. Deze cloud-providers bieden dan ook de meest populaire cloud-based ETL tools aan.

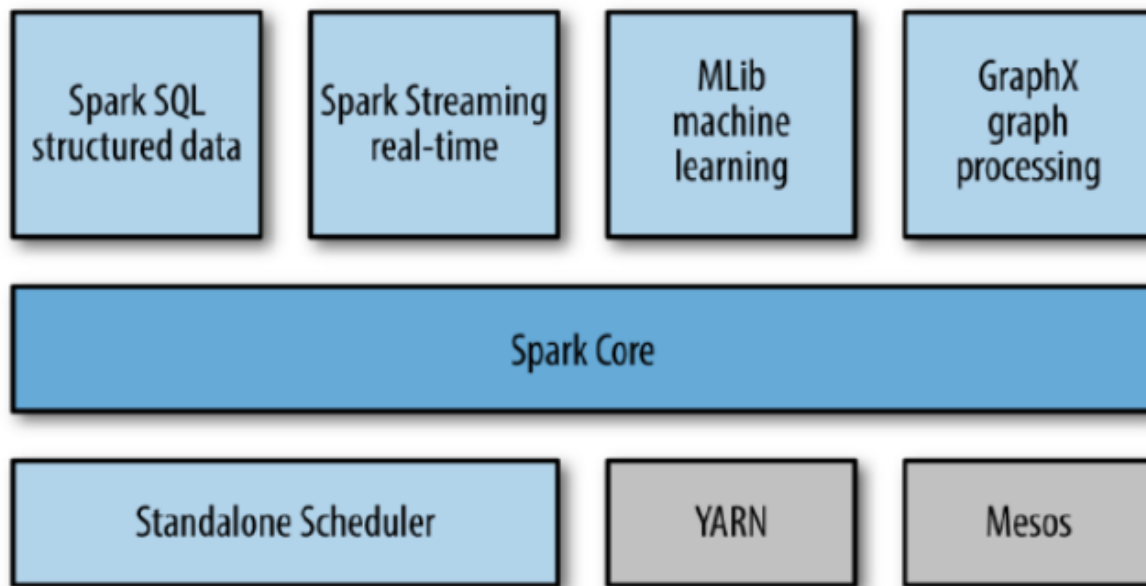
Microsoft biedt bijvoorbeeld Azure Data Factory en Azure Databricks aan. Binnen Azure Data Factory kan er gebruik gemaakt worden van Mapping Data Flows, dit is een code-vrije manier waarmee ETL's opgebouwd kunnen worden. De logica achter de ETL kan hierna makkelijk getest worden op live data en samples. (Kromer, 2022b)

Daarnaast biedt Azure ook Azure Databricks aan. Het verschil hierbij is dat de ETL's worden geïmplementeerd via code terwijl dat bij Azure Data Factory via de UI tools gebeurt. Azure Databricks is gebaseerd op het Apache Spark opensource project. Het grote voordeel is dat het platform het toelaat om makkelijker te kunnen samen werken. Daarnaast is Apache Spark niet enkel gelimiteerd tot het maken van ETL's maar kan het ook gebruikt worden voor real-time analytics, machine learning, graph processing, etc. (Etaati, 2019)

Ook Amazon Web Services (AWS) en Google Cloud Services bieden ETL tools aan. Zo heeft AWS bijvoorbeeld AWS Glue (Khan e.a., 2024) en Google Cloud heeft Google Data Fusion. (Jaiswal, 2022)

2.4. Apache Spark

Apache Spark is een gedistribueerd computing systeem dat het meest actieve Apache open source project is geworden. Het stelt zich in staat om data te verwerken in grote hoeveelheden met een relatief eenvoudig te gebruiken API voor zowel Python, Java, Scala en SQL. Het is in staat om om gegevenstransformaties en machine learning algoritmes te schrijven die relatief systeemafhankelijk zijn. (Karau & Warren, 2017)

**Figuur (2.1)**

Spark stack (Holden Karau & Zaharia, 2015)

Als eerst hebben we Spark Core, dit is het fundament van Apache Spark en biedt de basisfunctionaliteit voor het verdelen en verwerken van gegevens in een gedistribueerde omgeving. Daarnaast hebben we Apache SQL, dit is een module voor het verwerken van gestructureerde gegevens met behulp van SQL-achtige queries. Spark Streaming aan de andere kant is een uitbreiding van de core Spark API die real-time gegevensverwerking mogelijk maakt. MLib is dan weer de machine learning library van Apache Spark. GraphX is de bibliotheek binnen Apache Spark voor het verwerken van grafiekgegevens en het uitvoeren van grafiekanalyses. En ten slotte zijn er Cluster Managers, deze zijn verantwoordelijk voor het beheren van resources en het toewijzen van taken binnen een Spark-cluster. Voorbeelden hiervan zijn Apache Hadoop YARN, Apache Mesos en ingebouwde standalone-clustermodus van Spark. (Holden Karau & Zaharia, 2015)

2.5. Continuous Integration (CI) en Continuous Deployment (CD)

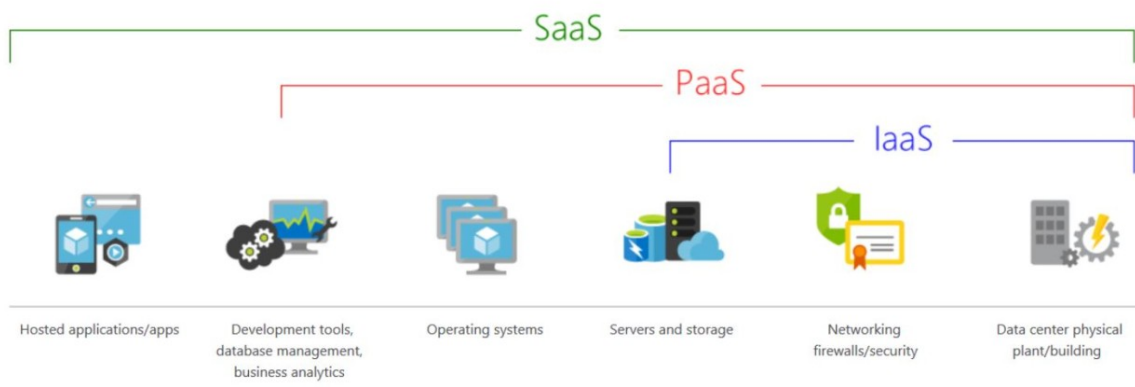
Bij Continuous Integration (CI) wordt de code van één of meerdere developers samen gevoegd, getest en gebouwd. Dit gebeurt continu. Een voorbeeld hiervan is dat een developer de code van zijn/haar GitHub branch merged op de “main” (of “master”) branch van het project. Het doel van CI is dat deze “main” (of “master”) branch gezond blijft en dat nieuwe aanpassingen door meerdere developers niet zorgt voor fouten in de code die resulteren in het falen van een build. (Jackson, 2020)

Continuous Delivery (CD) is een uitbereiding van Continuous Integration. Het is het automatiseren van het release process. Het zorgt er voor dat aanpassingen frequent gedeployed kunnen worden. (Jackson, 2020)

Ten slotte is er ook Continuous Deployment. Dit is een uitbereiding op Continuous Delivery en zorgt er voor dat code changes automatisch naar productie worden gepusht. (Jackson, 2020)

2.6. Microsoft Azure

Microsoft Azure is een cloud computing-platform die door Microsoft wordt aangeboden. Het biedt een breed scala aan cloudgebaseerde oplossingen en services waarmee bedrijven en ontwikkelaars applicaties kunnen bouwen, implementeren en beheren.



Figuur (2.2)

Verschillen tussen IaaS, PaaS en SaaS (Stoenescu, 2021)

Met Azure kunnen bedrijven gebruikmaken van Infrastructure as a Service (IaaS), waardoor ze onder andere virtuele machines, opslag en netwerken kunnen huren op basis van hun behoeften. Dit geeft organisaties de flexibiliteit om snel op te schalen zonder dat ze fysieke servers hoeven aan te schaffen. Daarnaast biedt het platform ook een Platform as a Service (PaaS), waarmee ontwikkelaars zich kunnen concentreren op het ontwikkelen van applicaties zonder zich zorgen te hoeven maken over de onderliggende infrastructuur. Ten slotte is er ook Software as a Service. Hierbij wordt er een applicatie gehost en beschikbaar gemaakt voor de gebruiker op basis van een subscription. Voorbeelden hier van zijn bijvoorbeeld officetools (Microsoft Office 365). (Suneetha, 2024)

Azure is sterk in dataopslag en -beheer, waarbij veilige en schaalbare opties worden aangeboden voor zowel gestructureerde als ongestructureerde gegevens. Boven-

dien kunnen organisaties gebruikmaken van analyse- en big data-services zoals Azure Synapse Analytics en HDInsight om inzichten te verkrijgen die strategische beslissingen ondersteunen. (Awati, [2023a](#))

Darnaast biedt Azure ook encryptie, toegangsbeheer en compliance-certificeringen om gegevens veilig te houden. (Siddiqui, [2023](#))

2.6.1. Subscription

Een Azure Subscription helpt bij het beheren van toegang tot Azure-services en bijhorende kosten. Binnen dit abonnement wordt de toegang tot resources beheerd en worden kosten bijgehouden. Elk abonnement wordt gekoppeld aan een factureringseenheid, zoals een creditcard of een zakelijke overeenkomst. Een Azure Subscription heeft vaak een gelaagde structuur. De belangrijkste elementen hiervan zijn:

Resource Groups

Dit zijn logische containers waarin je vergelijkbare resources kunt plaatsen om het beheer te vereenvoudigen. Bijvoorbeeld, alle resources die nodig zijn voor een specifieke applicatie kunnen binnen één resourcegroep worden gegroepeerd. (Microsoft, [2024f](#))

Roles en Role-Based Access Control (RBAC)

Gebruikers en groepen kunnen rollen toegewezen krijgen die hun toegang tot resources binnen het abonnement bepalen. Deze rollen kunnen variëren van lezer tot eigenaar, afhankelijk van de benodigde toegangsrechten. (Microsoft, [2024e](#))

Beleidsregels

Organisaties kunnen beleidsregels instellen om te controleren welke resources kunnen worden gebruikt en hoe ze worden ingezet. (Microsoft, [2024i](#))

2.6.2. Data Lake

Azure Data Lake is een cloudgebaseerde gegevensopslag ontworpen om bedrijven te helpen met het beheren en analyseren van grote hoeveelheden gestructureerde en ongestructureerde gegevens. De technologie achter Azure Data Lake bestaat uit drie hoofdelementen: Azure HDInsight, Azure Data Lake Storage en Azure Data Lake Analytics. (Awati, [2023b](#))

HDInsight is een open source analytics platform voor het beheren van big data. (Awati, [2023b](#))

Daarnaast is Data Lake Storage een veilige en schaalbare opslagservice die gegevens in hun oorspronkelijke formaat kan bewaren, ongeacht het type. Het is geoptimaliseerd voor big data-workloads, waardoor het grote hoeveelheden data van

verschillende bronnen snel kan verwerken en opslaan. (Awati, [2023b](#))

Aan de andere kant biedt Azure Data Lake Analytics een serverloze analyseomgeving waarin ontwikkelaars complexe queries kunnen uitvoeren op de opgeslagen gegevens. Door gebruik te maken van U-SQL, een querytaal die de kracht van SQL en C# combineert, kunnen gebruikers gemakkelijk aangepaste analyses uitvoeren op grote datasets. Bovendien betaalt men alleen voor de verbruikte rekenkracht, waardoor dit model schaalbaar en kostenefficiënt is. (Awati, [2023b](#))

Een belangrijk aspect van Azure Data Lake is dat kan integreren met andere Azure-services, zoals bijvoorbeeld Azure Synapse Analytics of Power BI.

2.6.3. Key Vault

Azure Key Vault maakt het mogelijk om veilig cryptografische sleutels en andere gevoelige informatie, zoals certificaten, API-sleutels en wachtwoorden op te slaan. De service biedt strenge toegangscontrole en encryptie om ervoor te zorgen dat gevoelige informatie privé en beschermd blijft. Met behulp van Azure Key Vault kunnen bijvoorbeeld secrets opgeslaan worden die dan via HTTPS toegankelijk zijn binnen beveiligde applicaties. Toegangsbeheer binnen Azure Key Vault wordt geregeld via Azure Active Directory (AAD), waardoor beheerders controle hebben over wie toegang heeft tot de Key Vault en wat ze kunnen doen met de opgeslagen geheimen. Bij het uitvoeren van ETL-processen (Extract, Transform, Load) is het van cruciaal belang dat gegevensbronnen en bestemmingen veilig met elkaar communiceren. Azure Key Vault speelt hierin een sleutelrol door als centrale opslagplaats voor alle gevoelige verbindingsinformatie te dienen, zoals database wachtwoorden en API-sleutels. Dit minimaliseert het risico op datalekken en verhoogt de veiligheid van het ETL-proces. (Microsoft, [2024a](#))

2.6.4. App Registration

Met behulp van Azure App Registration kan een applicatie de nodige credentials krijgen tot Azure services en API's. Het is een soort paspoort voor een specifieke applicatie. (Anaparthi, [2023](#))

2.6.5. Cost Management

In Azure is Microsoft Cost Management ontworpen voor het analyseren, monitoren en optimaliseren van kosten. Dankzij Cost Management kunnen rapporten en analyses gemaakt worden in de Azure Portal of in Power BI. Daarnaast kunnen er ook budgetten en waarschuwingen ingesteld worden. Ook zijn er verschillende hulpmiddelen beschikbaar die helpen bij het inschatten van cloudkosten. Kort samen gevat bevat het dus alle nodige tools voor het beheren, analyseren, inschatten en optimaliseren van kosten. (Microsoft, [2023](#))

2.6.6. Azure Data Factory (ADF)

Azure Data Factory is een Platform as a Service (PaaS) voor het implementeren van ETL's en ELT's. Zowel on-premises als cloudgegevensbronnen worden hierbij ge ondersteund voor het verplaatsen van gegevens. (Rawat & Narain, [2018](#))

Onderdelen

Azure Data Factory is opgebouwd uit verschillende onderdelen. Als eerste hebben we een pipeline. Dit is een groep van activiteiten die een reeks processen uitvoert zoals bijvoorbeeld het extraheren of transformeren van gegevens. Een voorbeeld van een activity is een Mapping Data Flow. Hiermee kan men logica voor datatransformaties ontwikkelen zonder code te schrijven. Daarnaast zijn er ook datasets. Dit is een representatie of verwijzing naar de daadwerkelijke gegevens in gegevensopslag. Een dataset is steeds gekoppeld aan een linked service. Deze slaan de informatie op die Azure Data Factory nodig heeft voor het connecteren naar een externe dataopslag. Een pipeline wordt uitgevoerd door een trigger. Er zijn veel verschillende soorten triggers voor veel verschillende soorten events. Daarnaast kan er voor een pipeline parameters gedefinieerd worden. Dit zijn read-only key-value pairs die een configuratie vormen. Ook kunnen er variables gebruikt worden om tijdelijk waardes op te slaan. In combinatie met parameters kunnen dan waardes tussen pipelines, data flows, en andere activiteiten door gegeven worden. Wanneer een pipeline wordt uitgevoerd zal er een pipeline run aangemaakt worden. (Microsoft, [2024g](#))

Mapping Data Flows

Met behulp van Mapping Data Flows kunnen ETL's geïmplementeerd worden zonder hiervoor gebruik te moeten maken van code. De resulterende data flows worden uitgevoerd als activiteiten binnen een Azure Data Factory pipeline dat gebruik maakt van Apache Spark Clusters. Deze Mapping Data Flows maken gebruik van data flow scripts. Dit zijn artifacten die gegenereerd worden door de UI. Het is een taal die de data transformatie beschrijft dat de Spark Cluster zal moeten uitvoeren. De UI van Azure Data Factory beheert het data flow script maar het script kan ook bekeken en handmatig bewerkt worden. (Kromer, [2022a](#))

In Mapping Data Flows kunnen verschillende soorten transformaties gedaan worden zoals bijvoorbeeld het joinen, het filteren of selecteren van data.

2.6.7. Azure Databricks

Azure Databricks is een geavanceerd platform voor data-analyse dat zich integreert met Azure services. Het biedt een complete omgeving voor het ontwikkelen, implementeren en delen van krachtige data-analyses en AI-toepassingen op grote schaal. Het integreert zich ook met de opensource-community zoals bijvoorbeeld Delta Lake, Delta Sharing, MLflow, Apache Spark en Redash. Veel voorkomende use cases van Azure Databricks zijn het bouwen van een data lakehouse voor on-

dernemingen, het implementeren van ETL's, gebruik van machine learning en dergelijke. (Microsoft, [2024c](#))

Notebooks

Met behulp van Azure Databricks Notebook kan er een collaboratieve, interactieve interface gebruikt worden die het mogelijk maakt om live code, tekst en visualisaties te combineren. Er kan bijvoorbeeld code geschreven worden in Python, SQL, Scala en R. Deze kan dan uitgevoerd worden met behulp van Apache Spark. Doordat Databricks opgericht is door de makers van Apache Spark kan er gebruik gemaakt worden van een geoptimaliseerde versie van Apache Spark. Deze heeft meer functionaliteiten en een betere performantie. Dit maakt Databricks interessant voor bedrijven die zich willen focussen op implementaties die vooral enkel Spark gebruiken. (Hill, [2023](#))

Clusters

Met behulp van Databricks Clusters kunnen jobs en notebooks uitgevoerd worden. Er zijn twee verschillende soorten clusters, All-purpose Clusters en Job Clusters. All-purpose Clusters worden manueel gebruikt in notebooks. Job Clusters aan de andere kant worden gebruikt binnen jobs. Wanneer een job wordt uitgevoerd zal deze cluster opgestart worden en wanneer de job klaar is met uitvoeren zal deze terug afgesloten worden. (Samuel, [2021](#))

Clusters kunnen één driver node en één of meerdere worker nodes hebben. De driver houdt de status van alle notebooks in het cluster bij en stuurt werk naar de worker nodes. (Samuel, [2021](#))

Jobs

Met behulp van Azure Databricks jobs kunnen applicaties zoals bijvoorbeeld notebooks uitgevoerd worden. Dit kan bijvoorbeeld via een scheduling systeem of wanneer nieuwe bestanden arriveren op een bepaalde externe locatie. Jobs kunnen aangemaakt worden via de Jobs UI, de Databricks CLI of via de Jobs API. (Microsoft, [2024h](#))

Data lakehouse

Delta Lake

Delta Lake is het standaard opslagformaat voor alle operaties binnen Azure Databricks. Het maakt gebruik van Parquet data bestanden met een file-based transaction log voor ACID transactions. (Microsoft, [2024d](#))

Delta Live Tables

Delta Live Tables is een framework voor het bouwen van processing pipelines. Hierbij wordt er gebruik gemaakt van streaming tables en materialized views. Streaming tables zijn Delta tables waarbij er extra support is voor streaming of incre-

mental data processing en materialized views zijn views waarbij de resultaten pre-computed zijn. (Microsoft, [2024b](#))

3

Methodologie

3.1. Literatuuronderzoek

In het literatuuronderzoek zijn we in detail op zoek gegaan naar de mogelijkheden die er zijn om ETL's te gaan implementeren. Hierbij houden we rekening dat Net IT met Microsoft producten werkt waardoor er vooral naar Azure gekeken wordt. Er is dus ook verder in detail gegaan op Azure Data Factory en Azure Databricks doordat deze verder onderzocht zullen worden. Ten slotte is er ook in detail gegaan op andere Azure services. Dit komt doordat we deze gebruiken in de proof-of-concepts.

3.2. Vergelijkingscriteria

Er zullen vergelijkingscriteria opgesteld worden om zo de prestaties van zowel Azure Data Factory als Azure Databricks gestructureerd te beoordelen.

3.3. Proof-of-concepts

Binnen Net IT wordt data van Microsoft 365 Customer Engagement geëxporteerd naar CSV bestanden en in Azure Data Lake geplaatst. Deze bestanden moeten minstens één keer per dag opgesplitst worden per groep per jaar en zullen moeten doorgestuurd worden naar de klant. Hiervoor moet er dus een ETL geïmplementeerd worden. Doordat er onderzocht wordt naar wat de beste mogelijkheid is voor het implementeren van deze ETL zal er dus een proof-of-concept uitgewerkt worden voor zowel Azure Data Factory en Azure Databricks. Voor het implementeren van deze proof-of-concepts zal er een pipeline gemaakt worden die ook bij de klant gebruikt wordt. Belangrijk hierbij is dat er voor deze bachelorproef gebruik gemaakt wordt van dummy data.

De tabellen uit data lake die gebruikt worden zijn “new_syndicalpremiumrequest”,

“new_person”, “new_bankaccount”, “new_year”, “new_membership”, “new_group” en “new_organizationyear”. Voor zowel Data Factory als Databricks wordt er eerst gekeken naar hoe deze opgezet kunnen worden. Vervolgens wordt er gekeken hoe er samen gewerkt kan worden en hoe source control geïmplementeerd kan worden. Ook gaan we kijken of deze kunnen geïntegreerd worden in bepaalde IDE's en of Infrastructure as Code gebruikt kan worden. Daarnaast wordt er ook gekeken hoe men data kan ophalen uit data lake met behulp van het Common Data Model en wordt er gekeken hoe de pipeline gedebugged kan worden. Vervolgens worden de belangrijkste transformaties van de pipeline overlopen. En ten slotte wordt er getoond hoe de data terug naar Azure Data Lake geschreven wordt. Op deze manier kan Data Factory en Databricks makkelijk vergeleken worden. Belangrijk is ook dat er getoond wordt hoe de output van beide pipelines met elkaar vergeleken kan worden om te valideren dat de output hetzelfde is.

Er wordt steeds gewerkt vanuit de tabel “new_syndicalpremiumrequest”. Dit zijn de premies die geëxporteerd worden vanuit Microsoft 365 Customer Engagement. Deze premies worden opgesplitst per groep, per jaar. Voor de huidige proof-of-concepts worden deze premies naar slechts één CSV bestand geëxporteerd. Dit zodat het geëxporteerde CSV bestand van Data Factory dan makkelijk vergeleken kan worden met het geëxporteerde CSV bestand van Databricks. Het resultaat van de ETL is dus een CSV bestand waarbij elke rij een premie is, hierbij heeft elke premie een groep en referentiejaar.

3.4. Uitvoeren van pipelines en verzamelen van resultaten

In dit deel zal er een testopstelling opgezet worden voor beide pipelines voor het verkrijgen van meetbare vergelijkingscriteria. Aan de hand van de resulterende data zullen grafieken opgesteld worden en interpretaties van deze grafieken genoteerd worden.

3.5. Conclusie

Aan de hand van de vergelijkingscriteria zullen er conclusies opgesteld worden met aanbevelingen voor Azure Data Factory of Azure Databricks.

4

Vergelijkingscriteria

4.1. Makkelijk meetbaar

- Kostprijs
- Performantie

Kostprijs en performantie zijn makkelijk meetbaar door de twee pipelines uit te voeren. De data is vergelijkbaar met elkaar dus aan de hand van grafieken kan er gekeken worden welke optie bijvoorbeeld het goedkoopst of snelst is.

4.2. Verschillen aantoonbaar aan de hand van een voorbeeld

- Mogelijkheid tot debuggen
- Source control
- ARM templates

Bovenstaande vergelijkingscriteria zijn minder makkelijk meetbaar maar kunnen wel aan de hand van een voorbeeld vergeleken worden. Conclusies trekken op basis van deze vergelijkingscriteria zal waarschijnlijk moeilijker zijn tenzij er een groot verschil tussen de twee zou zijn.

4.3. Eigen ervaring

Ten slotte is eigen ervaring ook zeer belangrijk. Dit kan helpen bij het begrijpen van subjectieve aspecten zoals bijvoorbeeld gebruiksgemak, esthetiek en gebruikerservaring. Ook is het belangrijk om te begrijpen waarom mijn voorkeur naar een bepaalde technologie zou gaan.

5

Proof of Concepts

5.1. Azure Data Factory (ADF)

5.1.1. Opzet van resources

Door in Microsoft Azure naar Data Factories te navigeren kan een nieuwe data factory aangemaakt worden.

The screenshot shows the 'Basics' tab of the Azure Data Factory creation wizard. At the top, there are tabs for 'Basics', 'Git configuration', 'Networking', 'Advanced', 'Tags', and 'Review + create'. Below the tabs, there is a link to 'Try it' and a section for 'Project details' with instructions to select a subscription and resource group. The 'Subscription' field is set to 'Visual Studio Enterprise Subscription – MPN - 2024'. The 'Resource group' field is set to 'LoekaBP', with a 'Create new' link below it. The 'Instance details' section includes 'Name' (data-factory-bp-loeka), 'Region' (West Europe), and 'Version' (V2).

Field	Value
Subscription *	Visual Studio Enterprise Subscription – MPN - 2024
Resource group *	LoekaBP
Name *	data-factory-bp-loeka
Region *	West Europe
Version *	V2

Figuur (5.1)

Aanmaken van Azure Data Factory

Bij het aanmaken van een data factory moet er een subscription en resource group gekozen worden. Er kan een nieuwe resource group aangemaakt worden of een

reeds bestaande gekozen worden. Daarnaast moet er een naam, gewenste regio en versie voor Data Factory gekozen worden. Git configuratie komt later aan bod. Wanneer de resource is aangemaakt kan Azure Data Factory opgestart worden.

5.1.2. Collaboration en source control

Binnen Azure Data Factory kan er op 2 manieren samen gewerkt worden.

Roles en permissions

Add role assignment ...

[Role](#) [Members](#) [Conditions](#) [Review + assign](#)

A role definition is a collection of permissions. You can use the built-in roles or you can create your own custom roles. [Learn more](#)

[Job function roles](#) [Privileged administrator roles](#)

Grant access to Azure resources based on job function, such as the ability to create virtual machines.

×
Type : All
Category : All

Name ↑↓	Description ↑↓
Data Factory Contributor	Create and manage data factories, as well as child resources within them.

Showing 1 - 1 of 1 results.

Figuur (5.2)

Toewijzen van Data Factory Contributor Role

Door de bij de resource group van de data factory de Data Factory Contributor role toe te wijzen kan men toegang geven tot volgende zaken:

- Het aanmaken, wijzigen en verwijderen van data factories en child resources
- Deployment van Resource Manager templates
- Het managen van App Insight alerts voor Data Factory
- Het aanmaken van support tickets

Source control

Azure Data Factory laat het toe om een Git repository te configureren via Azure Repos of GitHub. We kiezen hier voor Azure DevOps doordat er binnen Net IT hier mee gewerkt word.

Configure a repository

Net IT NV (118054c8-f9ce-4058-a30c-77ee43ef2918)

Specify the settings that you want to use when connecting to your repository.

☒ Select repository ☐ Use repository link

Azure DevOps organization name * ⓘ

loekalievans

Project name * ⓘ

Example Data Factory

Repository name * ⓘ

Example Data Factory

Collaboration branch * ⓘ

main

Publish branch * ⓘ

adf_publish

Root folder * ⓘ

/

Custom comment

☒ Use custom comment

Import existing resources

☒ Import existing resources to repository

Import resource into this branch ⓘ

Figuur (5.3)

Configuratie van Azure DevOps in Azure Data Factory

De collaboration branch is de enigste branch waarbij de publish knop zichtbaar zal zijn. Door te werken met feature branches en hiermee pull requests te maken op de collaboration branch kan er dus samen gewerkt worden. De publish branch is de branch waar alle ARM templates van de gepubliceerde factory opgeslaan worden.

5.1.3. IDE integratie

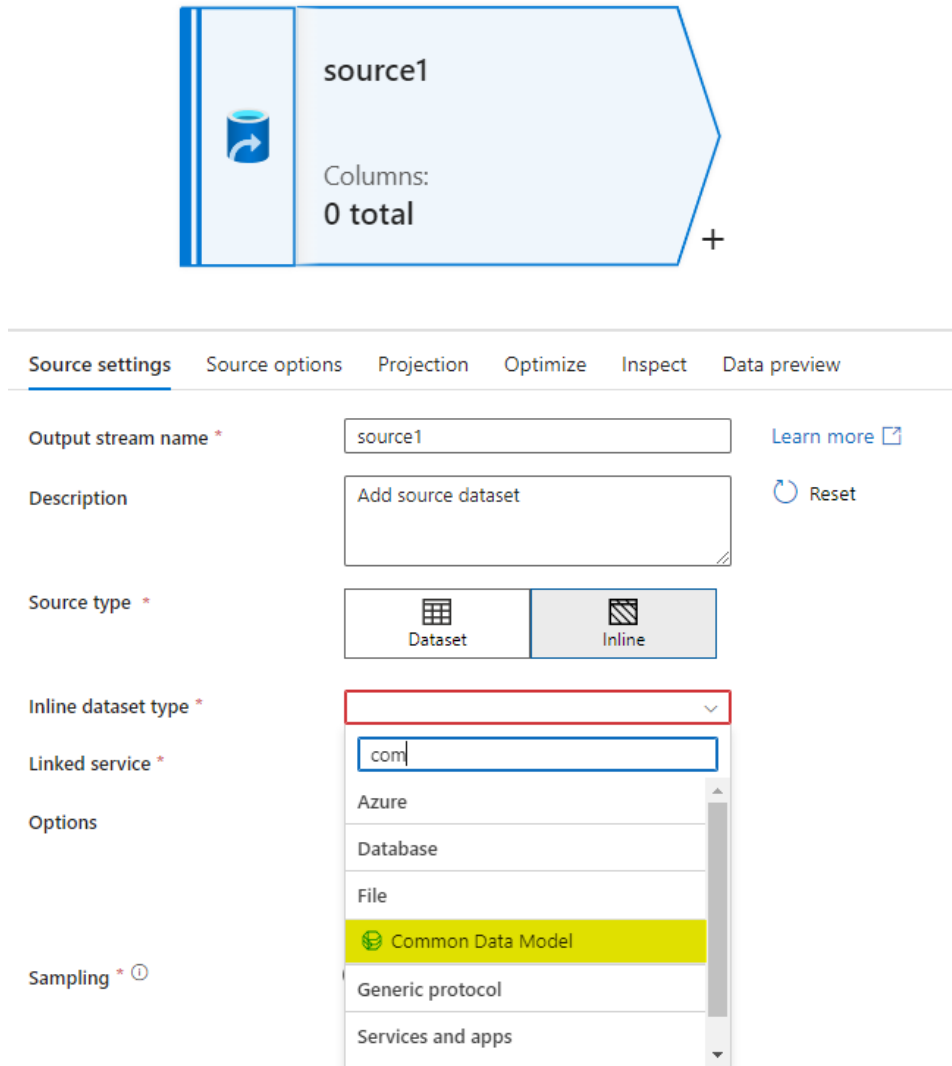
Met behulp van de “Azure Data Factory tools for Visual Studio” extensie kunnen data pipelines ontwikkeld worden in Visual Studio. Ook nu kunnen deze visueel ontwikkeld worden of kunnen de Azure Data Factory JSON bestanden met behulp van IntelliSense en schema-validatie bewerkt worden.

5.1.4. Infrastructure as code (IaC)

Azure Data Factory maakt gebruik van Azure Resource Manager (ARM) templates voor het opslaan van configuratie van verschillende ADF entiteiten zoals bijvoorbeeld pipelines, datasets, dataflows, enzovoort. Hiermee kan een data factory verplaatst worden naar een andere omgeving. Dit kan automatisch met behulp van Azure Pipelines of handmatig door een ARM template te uploaden via de Data Factory UX. Zoals te zien in sectie 5.1.2 is Git integreerbaar in Azure Data Factory. Hierin worden de ARM templates opgeslaan op de publish branch. Met behulp van Continuous Integration (CI) en Continuous Deployment (CD) via DevOps-pipelines kunnen aanpassingen die gebeuren op een bepaalde Git-repository dan automatisch gevalideert, getest en uitgerold worden naar een doelomgeving.

5.1.5. Ophalen van data uit Azure Data Lake

Het ophalen van data uit Data Lake in Data Flow gebeurt steeds op dezelfde manier.



The image shows the configuration interface for a source transformation in Azure Data Factory. At the top, there is a visual representation of the transformation: a blue arrow pointing right, labeled 'source1', with 'Columns: 0 total' below it. Below this is a tabbed interface with the following settings:



- Source settings** (selected):
 - Output stream name ***: source1
 - Description**: Add source dataset
 - Source type ***: Dataset (selected), Inline
 - Inline dataset type ***: dropdown menu showing options: Azure, Database, File, Common Data Model (highlighted), Generic protocol, Services and apps
 - Linked service ***: com
 - Options**: (empty)
 - Sampling ***: (empty)
- Source options**
- Projection**
- Optimize**
- Inspect**
- Data preview**


Figuur (5.4)

Configuratie van source transformation

Als source type wordt er steeds gekozen voor "inline". Dit doordat we slechts werken met één enkele dataflow en geen gedeelde datasets nodig hebben. Als inline data set type kiezen we voor Common Data Model.


New linked service

 Azure Data Lake Storage Gen2 [Learn more](#) 


 To avoid publishing immediately to Data Factory, please use Azure Key Vault to retrieve secrets securely. [Learn more](#) [here](#)

Name *


Description


Connect via integration runtime * 


Authentication type

Account selection method 

☒ From Azure subscription ☐ Enter manually

Azure subscription 

Storage account name * 


Test connection 


☒ To linked service ☐ To file path

Annotations

+ New

> Parameters

> Advanced 

 Test connection

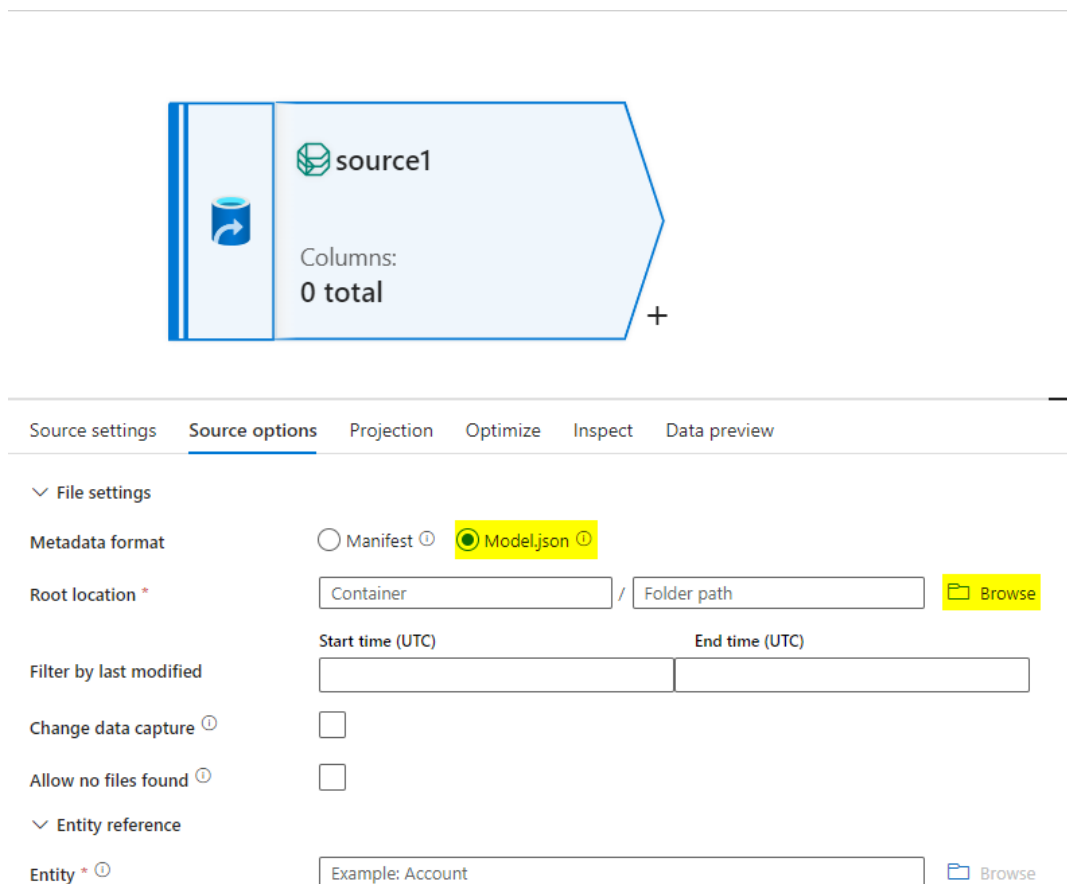
Figuur (5.5)

Configuratie van Linked Service

Er zal éénmalig een Linked Service aangemaakt moeten worden. Hierbij kiezen we voor Azure Data Lake Storage Gen2. We kunnen makkelijk gaan koppelen met de juiste data lake door een Azure Subscription en Storage account name aan te duiden. Door op “Test connection” te klikken kunnen we kijken of de connectie met data lake is gelukt. Door op “Create” te klikken hebben we nu een Linked Service die steeds bij elke Source gebruikt kan worden.

Let op: Doordat Git geen secrets opslaat is het aanbevolen om gebruik te maken

van Azure Key Vault voor het opslaan van connection strings of passwords.



The screenshot shows the 'Source options' tab for a connector named 'source1'. The connector icon is a blue cylinder with a white arrow. The text 'Columns: 0 total' is displayed. Below the connector name, there is a '+' sign. The 'Source options' tab is selected, and the 'File settings' section is expanded. The 'Metadata format' is set to 'Model.json' (highlighted in yellow). The 'Root location' is set to 'Container' / 'Folder path', with a 'Browse' button (highlighted in yellow). The 'Filter by last modified' section has two input fields for 'Start time (UTC)' and 'End time (UTC)'. The 'Change data capture' and 'Allow no files found' checkboxes are unchecked. The 'Entity reference' section has an 'Entity' field with the value 'Example: Account' and a 'Browse' button.

Figuur (5.6)

Configuratie van source options

Onder "Source options" wordt "Model.json" aangeduid. Door hierna op "Browse" te klikken kan er aangeduid worden waar het Model.json bestand te vinden is in Data Lake. Dit JSON bestand beschrijft hoe de data in Data Lake er uit ziet.

source1

Columns:
0 total

+

Source settings Source options Projection Optimize Inspect Data preview ●

File settings

Metadata format ☐ Manifest ☒ Modeljson

Root location * mydatalake / Folder path Browse

Filter by last modified Start time (UTC) End time (UTC)

Change data capture ☐

Allow no files found ☐

Entity reference

Entity * new_year Browse

Figuur (5.7)

Configuratie van source options

De gewenste entiteit kan nu geselecteerd worden door op “Browse” naast “Entity” te klikken. Let op: hier voor zal Data flow debug aan moeten staan.

source1

Columns:
0 total

+

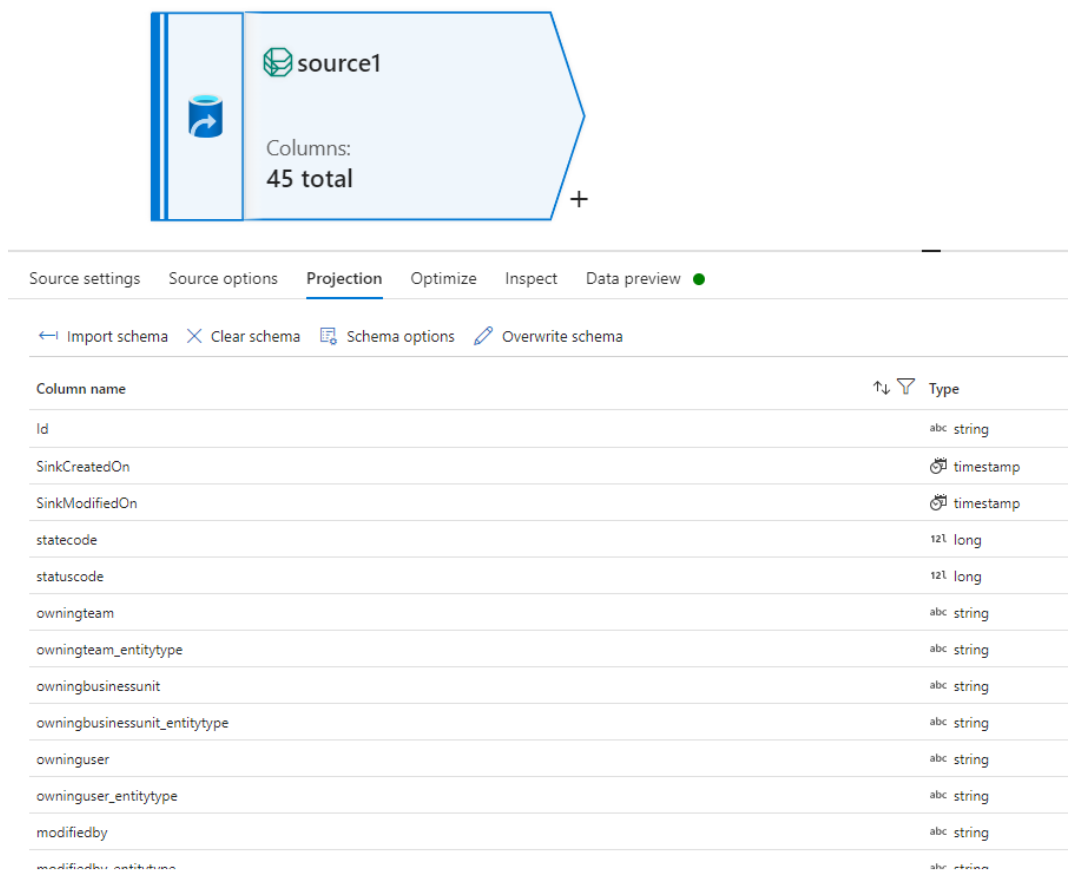
Source settings Source options Projection Optimize Inspect Data preview ●

Import schema Clear schema Schema options

Figuur (5.8)

Importeren van schema voor data flow source

Onder “Projection” kan er nu op “Import schema” geklikt worden om de verschillende kolommen met bijhorende types op te halen.



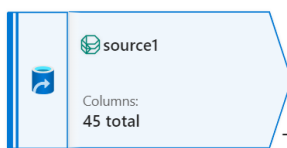
The screenshot shows the 'Projection' tab in the ADF interface. At the top, there's a card for 'source1' with a database icon and the text 'Columns: 45 total'. Below this, a horizontal menu contains 'Source settings', 'Source options', 'Projection' (selected), 'Optimize', 'Inspect', and 'Data preview'. Under the 'Projection' tab, there are four buttons: 'Import schema' (with a left arrow icon), 'Clear schema' (with an 'X' icon), 'Schema options' (with a list icon), and 'Overwrite schema' (with a pencil icon). Below these buttons is a table with two columns: 'Column name' and 'Type'. The table lists 14 columns with their respective types.

Column name	Type
Id	abc string
SinkCreatedOn	timestamp
SinkModifiedOn	timestamp
statecode	121 long
statuscode	121 long
owningteam	abc string
owningteam_entitytype	abc string
owningbusinessunit	abc string
owningbusinessunit_entitytype	abc string
owninguser	abc string
owninguser_entitytype	abc string
modifiedby	abc string
modifiedby_entitytype	abc string

Figuur (5.9)

Voorbeeld geïmporteerd schema van data flow source

De foto hierboven toont een voorbeeld van een geïmporteerd schema.



source1
Columns:
45 total

Source settings Source options Projection Optimize Inspect **Data preview**

Number of rows **INSERT** 17 **UPDATE** 0

Refresh Typecast Modify Map drifted Statistics Remove Export to CSV

Id	SinkCreatedOn	SinkModifiedOn	statecode	statuscode	owningteam	owningteam_e...	owningbusine...	owningbusine...
1465bcc5-7b4c...	NULL	NULL	0	1	NULL	NULL	134eee16-8827...	businessunit
1665bcc5-7b4c...	NULL	NULL	0	1	NULL	NULL	134eee16-8827...	businessunit
1865bcc5-7b4c...	NULL	NULL	0	1	NULL	NULL	134eee16-8827...	businessunit
1a65bcc5-7b4c...	NULL	NULL	0	1	NULL	NULL	134eee16-8827...	businessunit
1c65bcc5-7b4c...	NULL	NULL	0	1	NULL	NULL	134eee16-8827...	businessunit
1e65bcc5-7b4c...	NULL	NULL	0	1	NULL	NULL	134eee16-8827...	businessunit
2065bcc5-7b4c...	NULL	NULL	0	1	NULL	NULL	134eee16-8827...	businessunit
2265bcc5-7b4c...	NULL	NULL	0	1	NULL	NULL	134eee16-8827...	businessunit
2465bcc5-7b4c...	NULL	NULL	0	1	NULL	NULL	134eee16-8827...	businessunit
2a65bcc5-7b4c...	NULL	NULL	0	1	NULL	NULL	134eee16-8827...	businessunit
2665bcc5-7b4c...	NULL	NULL	0	1	NULL	NULL	134eee16-8827...	businessunit

Figuur (5.10)

Data preview in data flow

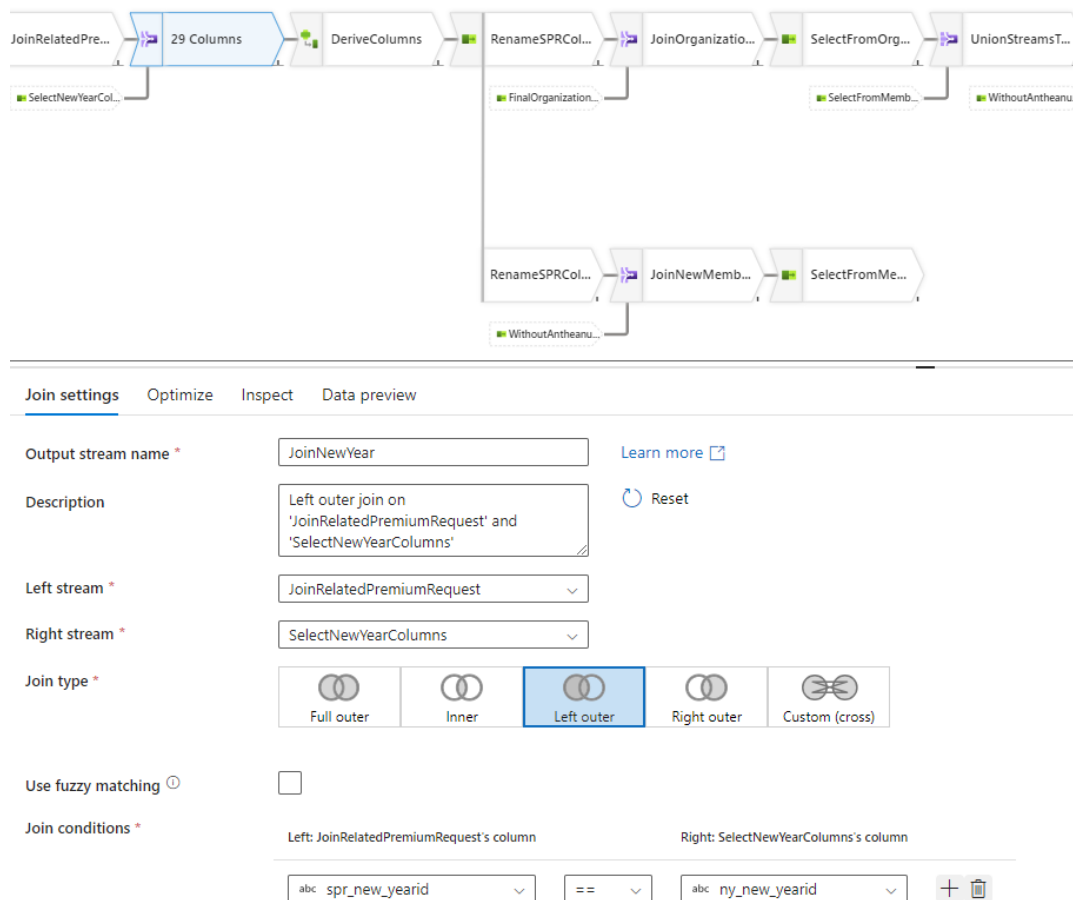
Door naar “Data preview” te navigeren kan er een preview bekeken worden van de data uit de gekozen tabel. Dit kan bij elke transformatie die er in data flow gebeurt. Ook hier voor moet Data flow debug aan staan.

5.1.6. Mogelijkheid tot debuggen

Zoals te zien op Figuur 5.10 kan er per transformatie een data preview getoond worden. Dit is een tabel die toont hoe de data na de transformatie er uit ziet. Belangrijk om hierbij te weten is dat “Data flow debug” aan zal moeten staan en er dus ook kosten in rekening gebracht zullen worden. Per tabel zal er standaard ook slechts 1000 rijen opgehaald worden. Voor kleine pipelines zal dit dus geen probleem zijn maar voor deze pipeline moest deze limiet verhoogd worden. Daarnaast kan dit soms heel traag worden wanneer de transformaties complex worden.

5.1.7. Belangrijkste transformaties

Determinatie van welke groepen de premie in hun bestand krijgen



Figuur (5.11)

Join van de tabel "new_year" op de tabel "new_syndicalpremiumrequest"

De tabel "new_syndicalpremiumrequest" heeft een kolom "spr_new_yearid". Om te gaan bepalen wat het referentiejaar van deze premie is zal dus de tabel "new_year" op de tabel "new_syndicalpremiumrequest" gejoind moeten worden aan de hand van dit id.

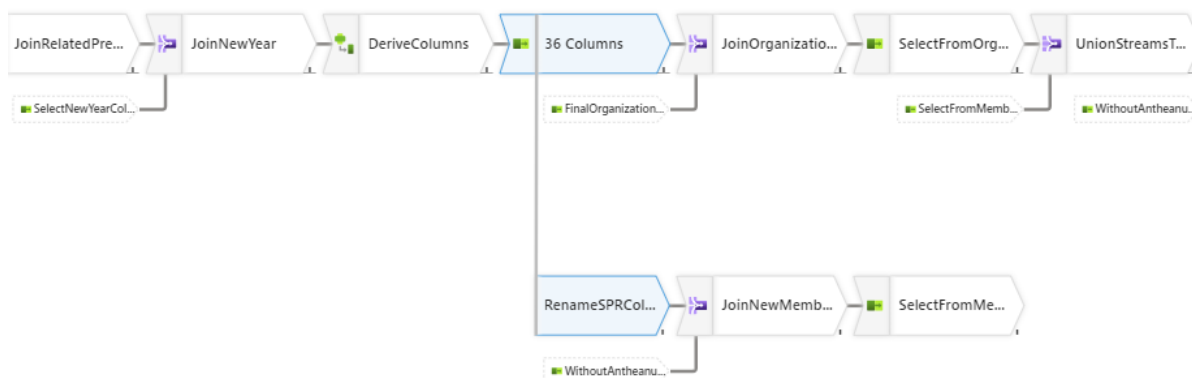
The screenshot displays a data pipeline with the following steps: JoinRelatedPre..., JoinNewYear, 36 Columns, RenameSPRCol..., JoinOrganizatio..., SelectFromOrg..., UnionStreamsT..., and JoinToSt. Below the pipeline, the 'Derived column's settings' panel is open for the column 'spr_new_bankaccountid'. The incoming stream is 'JoinNewYear'. The columns table is as follows:

Column	Expression	
Type	iif(spr_new_scannedrequestcode == 55201000... abc	+ -
Form	iif(spr_new_requesttypeid == "d026f6f9-fc82-... abc	+ -
Status	iif(rpr_new_treatmentstate == 552010004,case(s... abc	+ -
AvalonRemark	iif(spr_new_feedback == true(), 'FEEDBACK', spr_... abc	+ -
EntryYear	year(toDate(spr_createdon, "yyyy-MM-dd"))	123 + -
CalendarYear	year(currentUTC())	123 + -
RefYear	toInteger(ny_new_year)	123 + -

Figuur (5.12)

Derive "EntryYear", "CalendarYear" en "RefYear" op de tabel "new_syndicalpremiumrequest"

Voor het bepalen van de groepen hebben we 3 nieuwe kolommen nodig. Als eerste hebben we het "EntryYear" nodig, dit is het jaartal van "spr_createdon", de datum wanneer de record is aangemaakt. Daarnaast hebben we "CalendarYear" nodig, dit is het jaartal van de huidige datum. En ten slotte hebben we "RefYear" nodig, dit is het jaartal van de tabel "new_year" die net gejoind is geweest.

**Figuur (5.13)**

Hernoemen van kolommen op “new_syndicalpremiumrequest” en splitsing in twee aparte branches

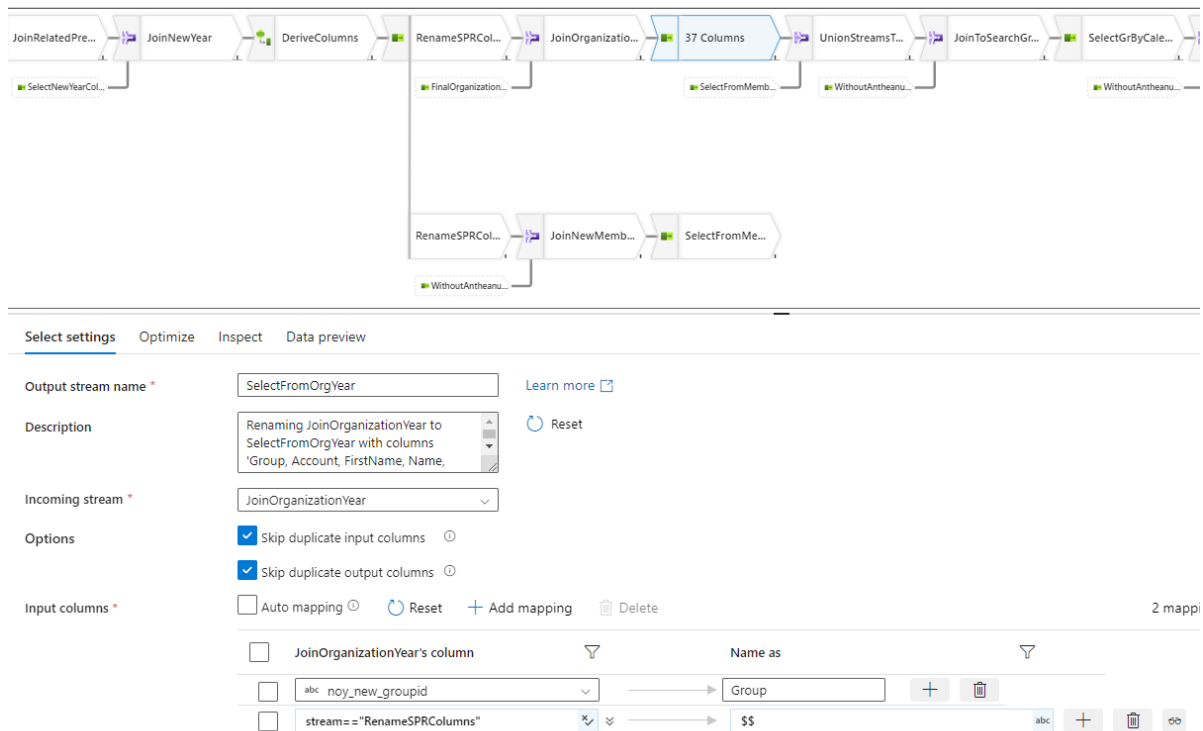
Vervolgens worden bepaalde kolommen van naam hernoemt. Welke kolommen dit zijn is onbelangrijk voor deze transformatie. Wat wel belangrijk is dat de pipeline zich nu opsplijst in twee aparte branches. Dit doordat er 2 inner joins zullen gebeuren.

Join settings	Optimize	Inspect	Data preview
Description	Inner join on 'RenameSPRColumns' and 'FinalOrganizationYearSelect'		Reset
Left stream *	RenameSPRColumns		
Right stream *	FinalOrganizationYearSelect		
Join type *	<div>Full outer</div> <div>Inner</div> <div>Left outer</div> <div>Right outer</div> <div>Custom (cross)</div>		
Use fuzzy matching ⓘ	<input type="checkbox"/>		
Join conditions *	<div>Left: RenameSPRColumns's column</div> <div>Right: FinalOrganizationYearSelect's column</div> <div> <div>abc spr_new_idadm</div> <div>==</div> <div>abc noy_new_idadmaux</div> <div>+</div> <div>abc spr_new_yearid</div> <div>==</div> <div>abc noy_new_yearid</div> <div>+</div> </div>		

Figuur (5.14)

Inner join van “new_organizationyear” op de tabel “new_syndicalpremiumrequest”

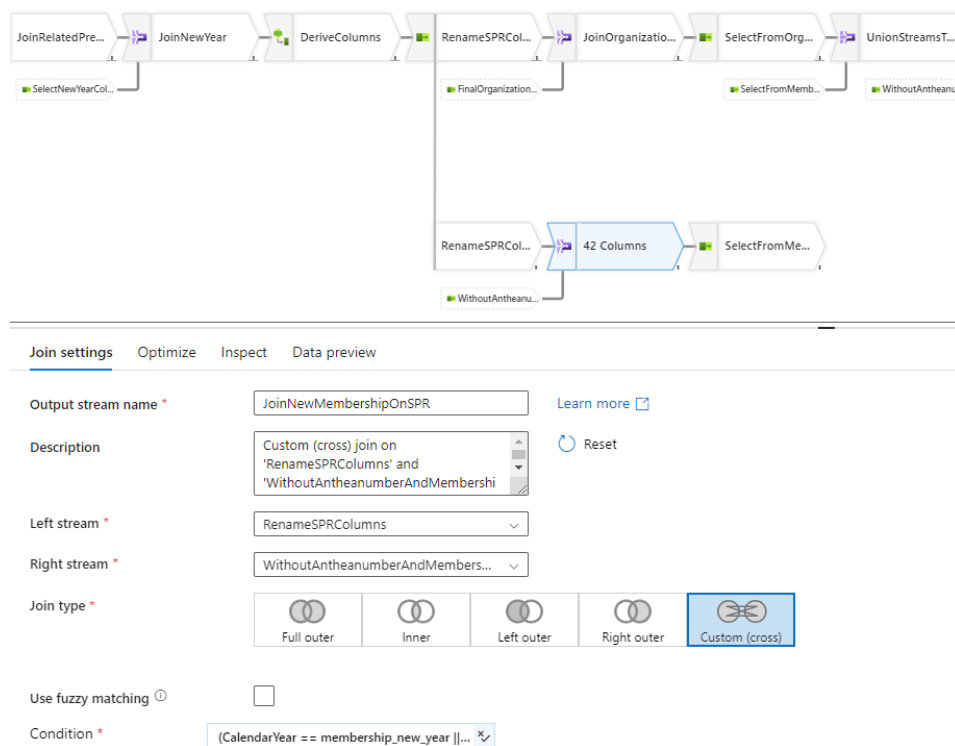
Er gebeurt nu een inner join van de tabel “new_organizationyear” op de tabel “new_syndicalpremiumrequest”. Hierbij wordt er aan de hand van IDADM en het id van het referentiejaar gejoind.



Figuur (5.15)

Selecteren en hernoemen van kolommen op de tabel “new_syndicalpremiumrequest”

Vervolgens worden alle kolommen, die er voor de join waren, geselecteerd. Daarnaast wordt er één kolom “noy_new_groupid” geselecteerd en hernoemd naar “Group”.

**Figuur (5.16)**

Custom (cross) join van “new_membership” op de tabel “new_syndicalpremiumrequest”

Bij de tweede branch wordt de tabel “new_membership” gejoind op de tabel “new_syndicalpremiumrequest”. Er wordt gebruik gemaakt van een custom (cross) join doordat er OR condities gebruikt worden. De custom (cross) join is de enigste join die gebruik kan maken van OR condities binnen Azure Data Factory. Door deze condities, zal deze join werken net zoals een inner join.

```

1  (CalendarYear = membership_new_year || EntryYear =
    membership_new_year || RefYear = membership_new_year)
2  && spr_new_personid = membership_new_personid

```

In de conditie van de custom (cross) join wordt er vergeleken of CalendarYear, EntryYear of RefYear overeenkomt met het jaartal van de membership. Daarnaast wordt er ook gekeken of personid overeen komt. Het is belangrijk dat deze vergelijking op personid niet wordt vergeten aangezien de pipeline dan anders oneindig zal blijven runnen.

The top part of the image shows a data pipeline with the following transformations: JoinRelatedPre..., JoinNewYear, DeriveColumns, RenameSPRCol..., JoinOrganizatio..., SelectFromOrg..., UnionStreamsT..., JoinToSearchGr..., and SelectGrByCale... Below this, a branch leads to another RenameSPRCol... transformation, followed by JoinNewMemb... and a final output of 38 Columns.

The bottom part shows the configuration for the 'SelectFromMemberships' transformation:

- Output stream name:** SelectFromMemberships
- Description:** Renaming JoinNewMembershipOnSPR to SelectFromMemberships with columns 'Group, ng_new_groupprefix,
- Incoming stream:** JoinNewMembershipOnSPR
- Options:**
 - ☒ Skip duplicate input columns
 - ☒ Skip duplicate output columns
- Input columns:**
 - ☐ Auto mapping
 - ☐ Reset
 - ☐ Add mapping
 - ☐ Delete

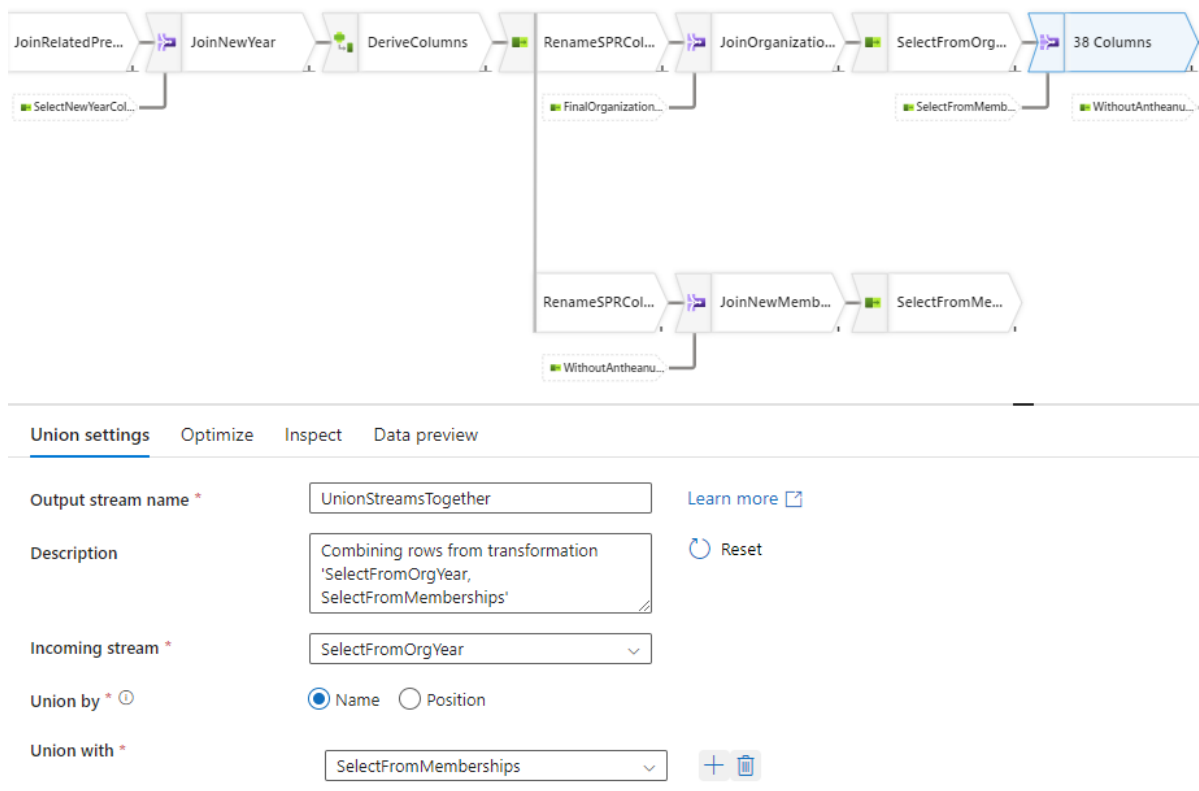
The input column mapping table is as follows:

JoinNewMembershipOnSPR's column	Name as
abc membership_new_groupid	Group
abc ng_new_groupprefix	ng_new_groupprefix
stream == "RenameSPRColumns"	\$\$

Figuur (5.17)

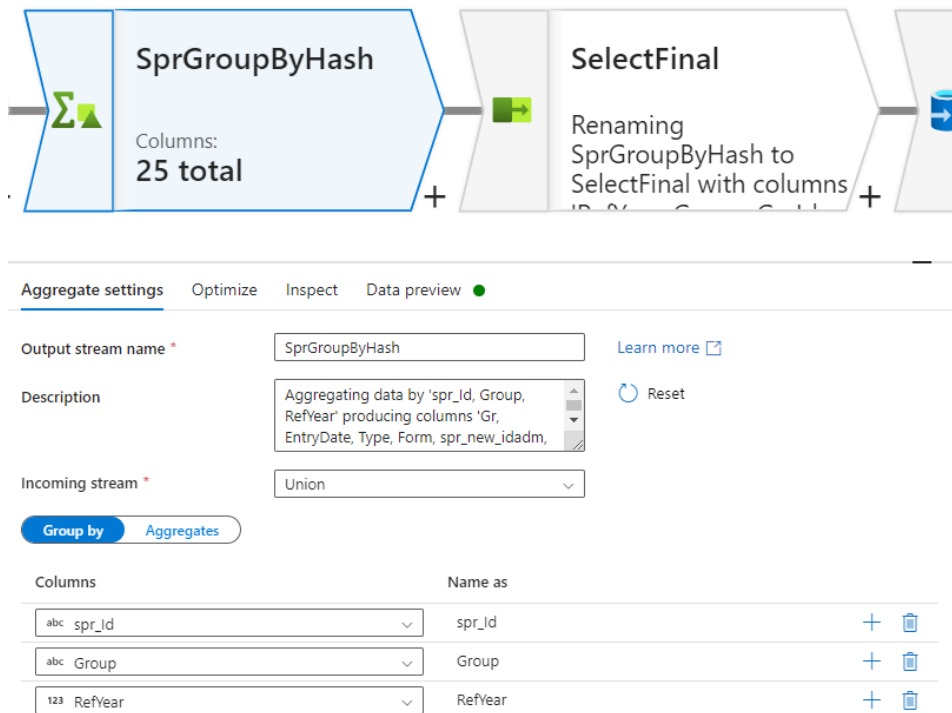
Selecteren en hernoemen van kolommen op de tabel "new_syndicalpremiumrequest"

Ook na deze join worden alle kolommen die er voor de join waren geselecteerd. Daarnaast wordt er één kolom "membership_new_groupid" geselecteerd en hernoemd naar "Group". Ten slotte wordt de kolom "ng_new_groupprefix" geselecteerd maar dit heeft te maken met een andere transformatie.

**Figuur (5.18)**

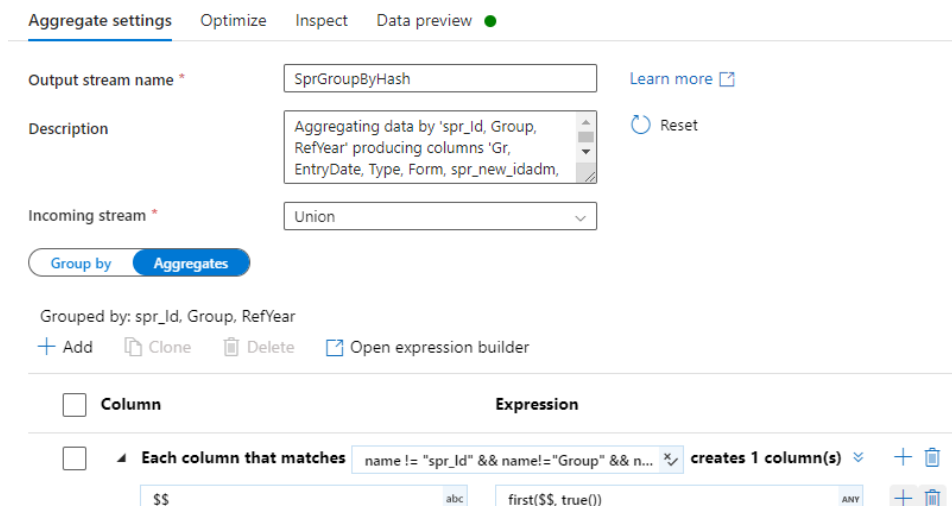
Union van twee branches

Beide branches hebben nu dezelfde kolommen met een extra kolom "Group". Daarnaast heeft de onderste branch nog één extra kolom "ng_new_groupprefix". De beide branches worden samen gevoegd met behulp van een union. De bovenste branch die de kolom "ng_new_groupprefix" niet heeft zal voor deze kolom de waarde "NULL" krijgen in de records komende van deze branch.

**Figuur (5.19)**

Group by "spr_Id", "Group" en "RefYear" in "new_syndicalpremiumrequest"

Om te voorkomen dat een premie twee keer in het zelfde bestand terecht komt voor een bepaalde groep en referentiejaar zal er op het einde van de pipeline gegroepeerd worden op basis van id van de premie, groep en referentiejaar.

**Figuur (5.20)**

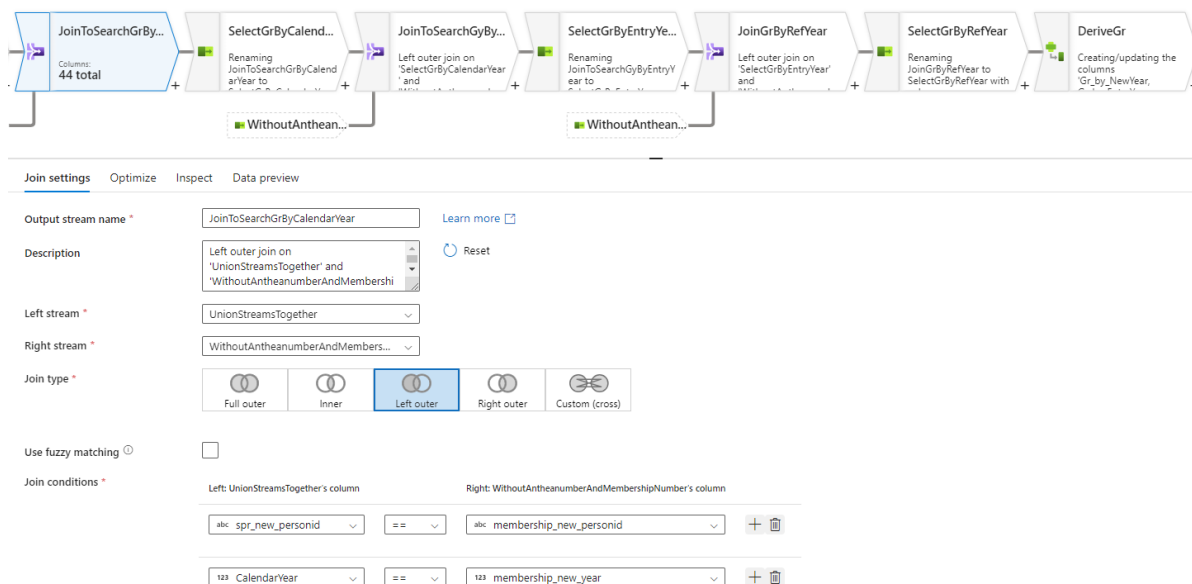
Group by "spr_Id", "Group" en "RefYear" in "new_syndicalpremiumrequest"

1 `name ≠ "spr_Id" && name ≠ "Group" && name ≠ "RefYear"`

Voor alle kolommen behalve de kolommen die in de “Group by” gebruikt worden zal de aggregatiefunctie “first” gebruikt worden. De tweede parameter “true()” wordt gebruikt om aan te geven dat “NULL” waardes genegeerd moeten worden. Dit zal dus resulteren in de eerste waarde die niet “NULL” is van de kolom groep. Indien alle waardes “NULL” zijn zal dit wel in “NULL” resulteren.

De premie heeft nu een id, een groep en een referentiejaar. Op basis hier van kan bepaald worden welke premie naar naar welk exportbestand voor een bepaalde groep en referentiejaar moet geëxporteerd worden.

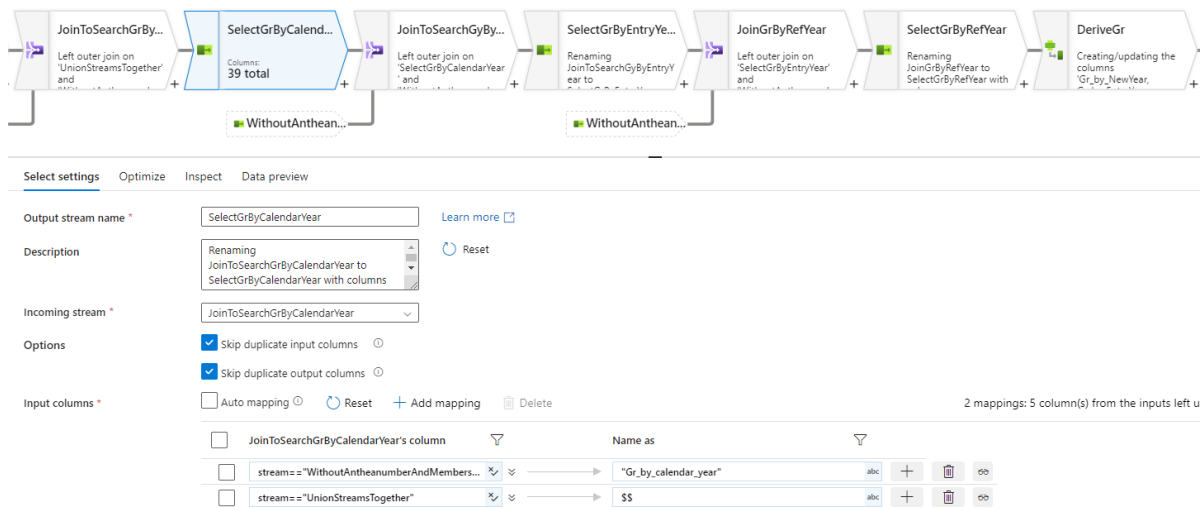
Bepalen van de kolom “Gr” voor een premie



Figuur (5.21)

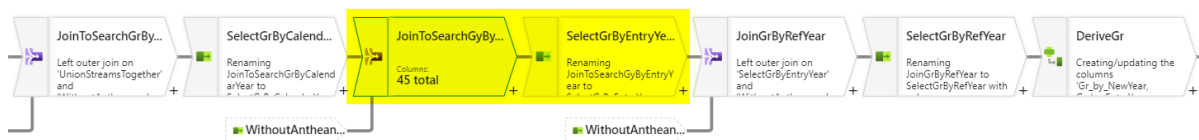
Joinen van de tabel “new_membership” op de tabel “new_syndicalpremiumrequest”

De tabel “new_membership” wordt gejoind op de tabel “new_syndicalpremiumrequest”. Dit gebeurt aan de hand van “new_personid” en het kalenderjaar van de premie.

**Figuur (5.22)**

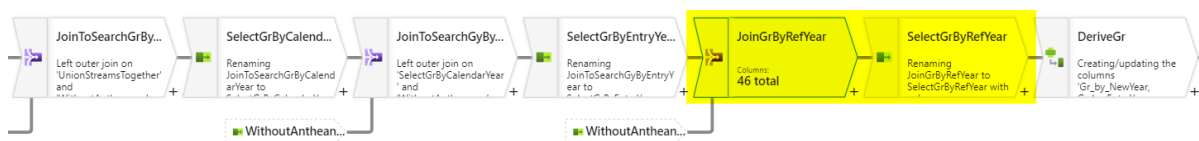
Selecteren van de juiste kolommen in “new_syndicalpremiumrequest”

De kolommen die reeds bestonden voor de join worden geselecteerd. Daarnaast wordt de kolom “ng_new_groupprefix” geselecteerd en hernoemd naar “Gr_by_calendar_year”.

**Figuur (5.23)**

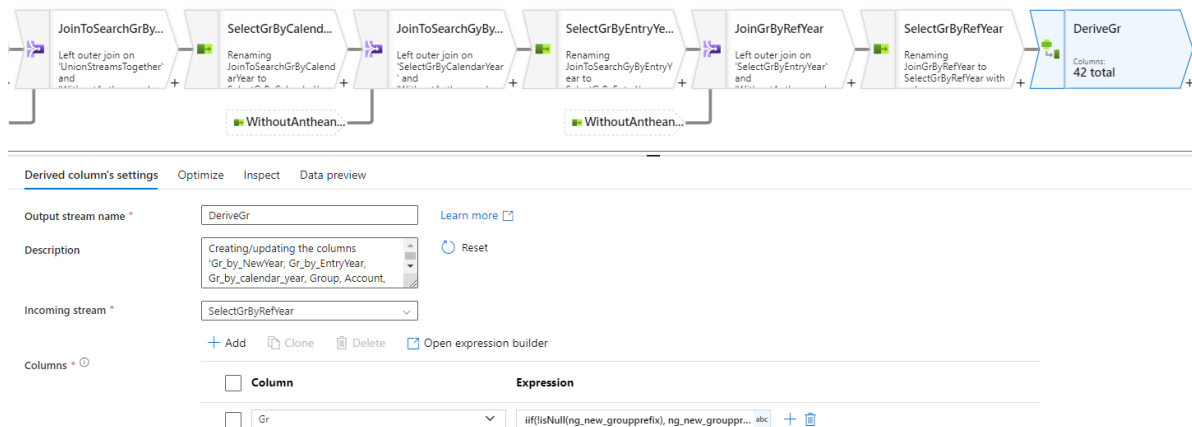
Herhalen van transformaties

Dezelfde transformaties gebeuren opnieuw maar deze keer aan de hand van “EntryYear”. Ook hier worden de reeds bestaande kolommen geselecteerd en wordt de nieuwe kolom “ng_new_groupprefix” hernoemd naar “Gr_by_EntryYear”.

**Figuur (5.24)**

Herhalen van transformaties

Ook voor “RefYear” zullen deze transformaties gebeuren, resulterend in een nieuwe kolom “Gr_by_RefYear”.

**Figuur (5.25)**

Bepalen van “Gr” in “new_syndicalpremiumrequest”

We hebben nu een group prefix reeds komende uit een andere transformatie 5.17 en group prefixes komende uit de joins die we hebben uitgevoerd.

```

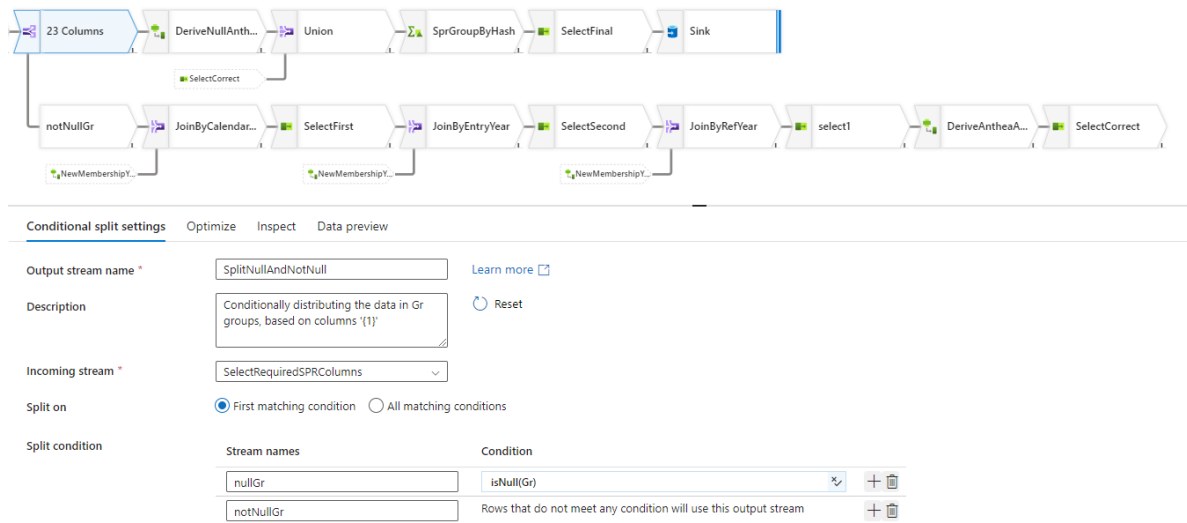
1 iif(!isNull(ng_new_groupprefix), ng_new_groupprefix,
2   iif(!isNull(Gr_by_calendar_year), Gr_by_calendar_year,
3     iif(!isNull(Gr_by_EntryYear), Gr_by_EntryYear,
4       iif(!isNull(Gr_by_NewYear), Gr_by_RefYear,
         toString(null())))))

```

Om de kolom “Gr” te bepalen zal er gezocht worden naar de eerste kolom die niet “NULL” is startende van uit “ng_new_groupprefix” waarbij er vervolgens gekeken wordt naar “Gr_by_calendar_year”, “Gr_by_EntryYear” en “Gr_by_NewYear”.

Ook voor deze transformatie is het belangrijk dat er gegroepeerd wordt zoals bij Figuur 5.19 om te voorkomen dat dezelfde premie twee keer in een groep voor een bepaald referentiejaar terecht komt.

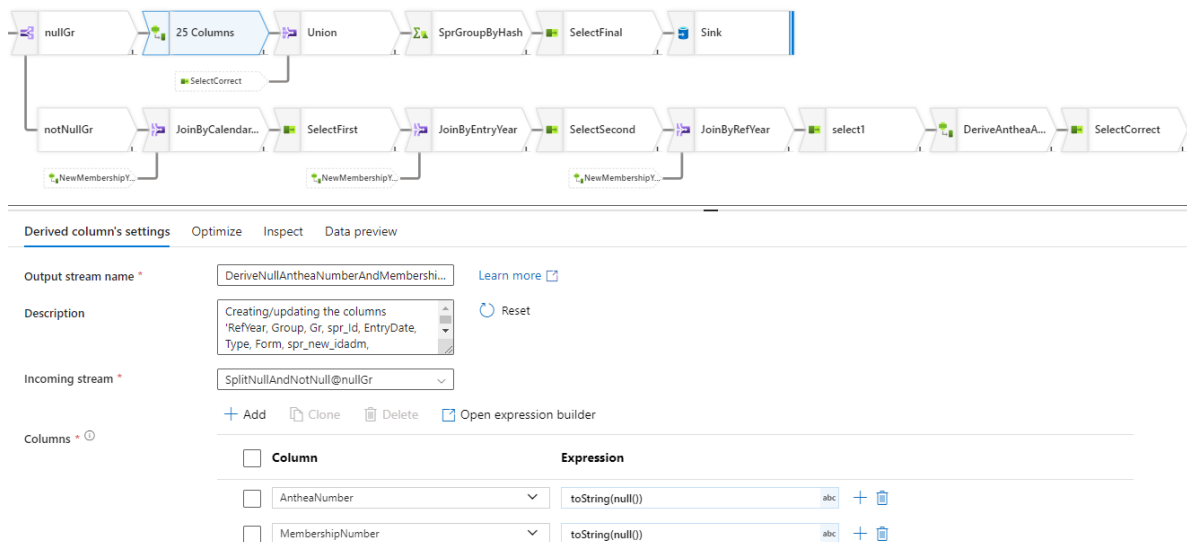
Bepalen van de kolommen “AntheaNumber” en “MembershipNumber” voor een premie



Figuur (5.26)

Conditional split op de tabel “new_syndicalpremiumrequest”

De pipeline zal splitten in twee branches met behulp van een conditional split. Alle rijen waarbij de kolom “Gr” “NULL” is, zullen boven verder gaan. Alle rijen waarbij de kolom “Gr” niet “NULL” zijn, zullen onderaan verder gaan.



Figuur (5.27)

Derive “AntheaNumber” en “MembershipNumber” op de tabel “new_syndicalpremiumrequest”

Voor de rijen waarbij de kolom “Gr” “NULL” was, zullen de kolommen “AntheaNumber” en “MembershipNumber” ook “NULL” zijn.

Join settings Optimize Inspect Data preview

Description: Left outer join on 'SplitNullAndNotNull@notNullGr' and 'NewMembershipYearToInteger' [Reset](#)

Left stream *: SplitNullAndNotNull@notNullGr

Right stream *: NewMembershipYearToInteger

Join type *: ☒ Full outer ☐ Inner ☒ Left outer ☐ Right outer ☐ Custom (cross)

Use fuzzy matching [?](#) ☐

Join conditions *

Left: SplitNullAndNotNull@notNullGr's column		Right: NewMembershipYearToInteger's column
abc spr_new_groupid	==	abc membership_new_groupid
123 CalendarYear	==	123 membership_new_year
abc spr_new_personid	==	abc membership_new_personid

Figuur (5.28)

Join van de tabel “new_membership” op de tabel “new_syndicalpremiumrequest” aan de hand van “new_groupid”, “new_personid” en “CalendarYear”

Voor de rijen waarbij de kolom “Gr” niet “NULL” was, zal de tabel “new_membership” hier op gejoind worden. Hierbij wordt er gebruik gemaakt van “group_id”, “new_personid” en “CalendarYear”.

Select settings Optimize Inspect Data preview

Output stream name *: SelectFirst [Learn more](#)

Description: Renaming JoinByCalendarYear to SelectFirst with columns 'antheanumber_first' [Reset](#)

Incoming stream *: JoinByCalendarYear

Options: ☒ Skip duplicate input columns [?](#) ☒ Skip duplicate output columns [?](#)

Input columns *

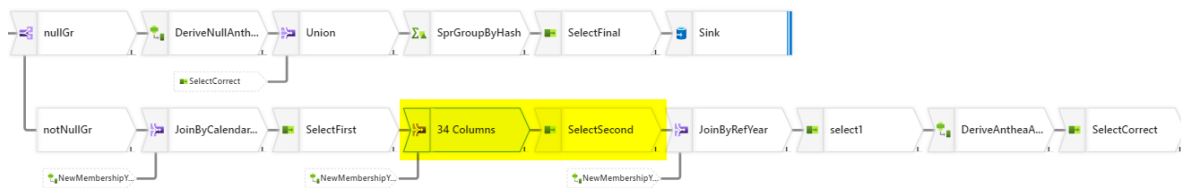
☐ Auto mapping [?](#) [Reset](#) [Add mapping](#) [Delete](#) 3 mappings: 7 column(s) from

JoinByCalendarYear's column	Name as
abc membership_new_antheanumber	antheanumber_first
abc membership_new_newmembershipnumber	membershipnumber_first
stream == "SplitNullAndNotNull@notNullGr"	\$\$

Figuur (5.29)

Select op de tabel “new_syndicalpremiumrequest”

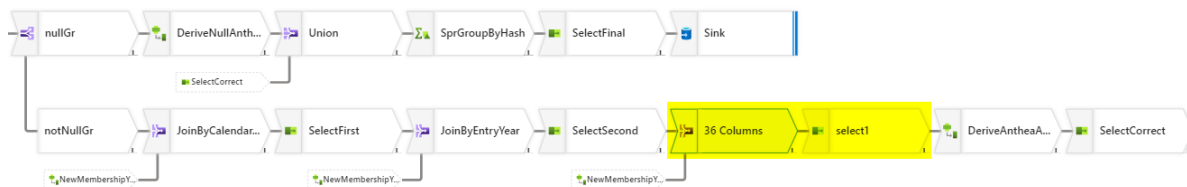
Alle kolommen die er voor de join waren zullen opnieuw geselecteerd worden. Daarnaast worden “membership_new_antheanumber” en “membership_new_newmembershipnumber” geselecteerd en hernoemd naar “antheanumber_first” en “membershipnumber_first”.



Figuur (5.30)

Join van de tabel “new_membership” op de tabel “new_syndicalpremiumrequest” aan de hand van “new_groupid”, “new_personid” en “EntryYear”

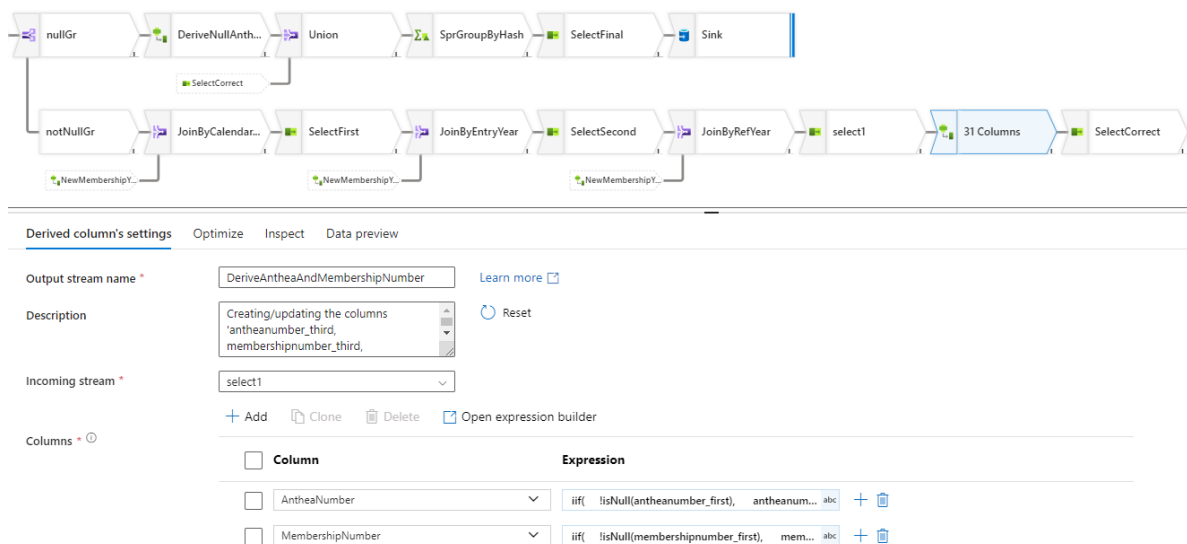
De vorige twee transformatie gebeuren opnieuw maar deze keer wordt er gebruik gemaakt van “EntryYear” in plaats van “CalendarYear”. Daarnaast worden de kolommen “membership_new_antheanumber” en “membership_new_newmembershipnumber” deze keer hernoemd naar “antheanumber_second” en “membershipnumber_second”.



Figuur (5.31)

Join van de tabel “new_membership” op de tabel “new_syndicalpremiumrequest” aan de hand van “new_groupid”, “new_personid” en “RefYear”

Ook zal voor “RefYear” in plaats van “CalendarYear” deze transformaties opnieuw gebeuren. Deze keer met de kolommen “antheanumber_third” en “membershipnumber_third” als resultaat.

**Figuur (5.32)**

Derive van kolommen “AntheaNumber” en “MembershipNumber” op de tabel “new_syndicalpremiumrequest”

De kolommen “AntheaNumber” en “MembershipNumber” worden nu berekent.

```

1 iif(!isNull(antheanumber_first), antheanumber_first,
2   iif(!isNull(antheanumber_second), antheanumber_second,
3     iif(!isNull(antheanumber_third), antheanumber_third,
        toString(null()))))

```

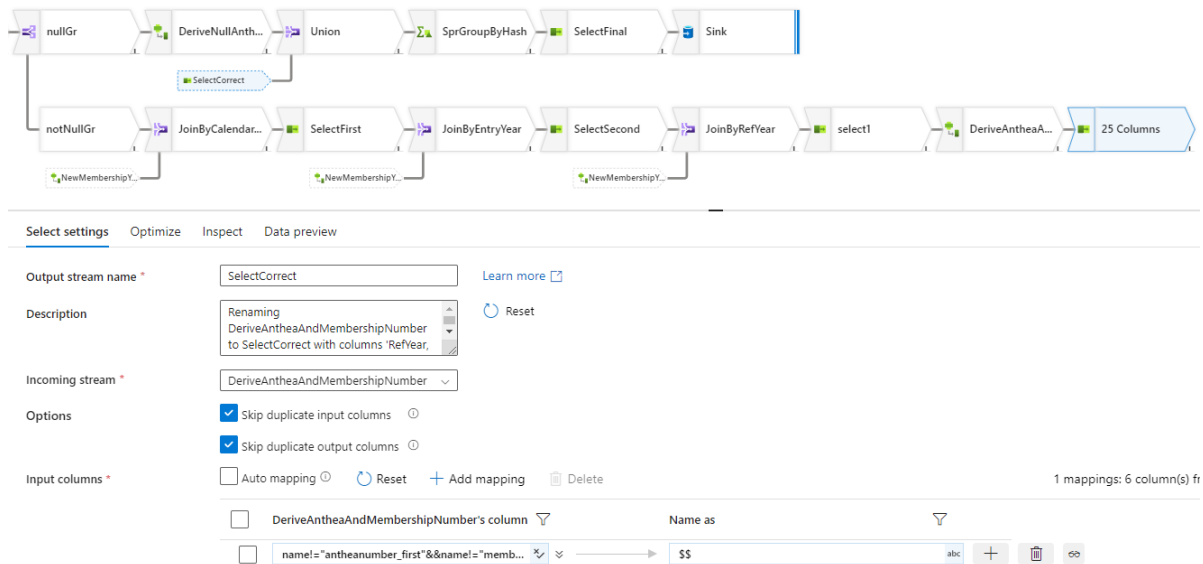
Wanneer het antheanummer van de eerste join niet “NULL” is zal deze gebruikt worden, anders zal er gekeken worden of de tweede niet “NULL” is. Als de tweede “NULL” is zal de derde gebruikt worden. Indien de derde ook “NULL” is zal dit resulteren in “NULL” als waarde voor de kolom “AntheaNumber”.

```

1 iif(!isNull(membershipnumber_first), membershipnumber_first,
2   iif(!isNull(membershipnumber_second), membershipnumber_second,
3     iif(!isNull(membershipnumber_third), membershipnumber_third,
        toString(null()))))

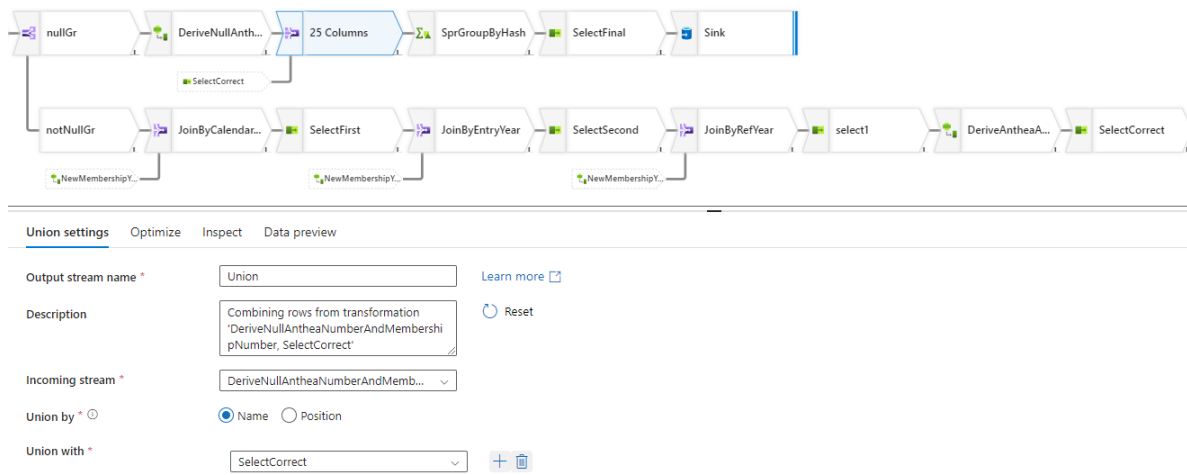
```

Ook de kolom “MembershipNumber” wordt op dezelfde manier berekent.

**Figuur (5.33)**

Verwijderen van onnodige kolommen op de tabel “new_syndicalpremiumrequest”

De kolommen die niet nodig zijn worden verwijderd.

**Figuur (5.34)**

Union van twee streams

Beide streams worden nu samen gevoegd met behulp van een union.

En ten slotte is het ook voor deze transformatie belangrijk dat er gegroepeerd wordt zoals bij Figuur 5.19 om te voorkomen dat dezelfde premie twee keer in een groep voor een bepaald jaartal terecht komt.

5.1.8. Schrijven van data naar Azure Data Lake

Sink Settings Errors Mapping Optimize Inspect Data preview

i This sink currently has Single partition set in Optimize. This will make your data flow execution longer. The recommended setting is Multiple partitions.

Output stream name * Sink [Learn more](#)

Description Add sink dataset [Reset](#)

Incoming stream * SelectFinal

Sink type * Dataset **Inline** Cache

Inline dataset type * DelimitedText

Linked service * AzureDataLakeStorage3 [Test connection](#) [Edit](#) [New](#)

Skip line count

Figuur (5.35)

Sink in Azure Data Factory

Voor het schrijven van data naar Azure Data Lake wordt er gebruik gemaakt van een sink. Deze maakt ook gebruik van de Linked Service aangemaakt in Figuur 5.5. Als “Inline dataset type” is hierbij gekozen voor “DelimitedText” zodat de resulterende output een CSV bestand is. Belangrijk is dat voor de proof-of-concepts één enkel CSV bestand geschreven wordt, hierdoor staat “Partition option” op “Single partition”.

5.2. Azure Databricks

5.2.1. Opzet van resources

Door in Microsoft Azure naar Databricks te navigeren kan er een nieuwe Azure Databricks workspace aangemaakt worden.

Basics Networking Encryption Security & compliance Tags Review + create

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Visual Studio Enterprise Subscription – MPN - Spoke 2
Resource group *	LoekaBP
	Create new

Instance Details

Workspace name *	databricks-loeka-standard
Region *	West Europe
Pricing Tier *	Standard (Apache Spark, Secure with Microsoft Entra ID)
Managed Resource Group name	databricks-loeka-standard-mrg

Figuur (5.36)

Configuratie van Azure Databricks

Bij het aanmaken van databricks moet er een subscription en resource group gekozen worden. Er kan een nieuwe resource group aangemaakt worden of een reeds bestaande gekozen worden. Daarnaast moet er een naam, gewenste regio en pricing tier gekozen worden. Als pricing tier kiezen we hier voor “Standard” doordat we geen gebruik gaan maken van Premium features. Ten slotte kan er ook een Managed Resource Group name gekozen worden. Deze resource group houdt alle resource bij die databricks nodig heeft, zoals bijvoorbeeld virtual machines, storage accounts en virtual networks.

[Compute](#) > [New compute](#) >

Loeka Lievens's Cluster

☒ Multi node ☐ Single node

Access mode ⓘ Single user access ⓘ

Single user | ▾

Loeka Lievens | ▾

Performance

Databricks runtime version ⓘ

Runtime: 11.3 LTS (Scala 2.12, Spark 3.3.0) | ▾

☒ Use Photon Acceleration ⓘ

Worker type ⓘ

Standard_DS3_v2 | 14 GB Memory, 4 Cores | ▾

Min workers

2

Max workers

8

☐ Spot instances ⓘ

Driver type

Same as worker | 14 GB Memory, 4 Cores | ▾

☒ Enable autoscaling ⓘ☒ Terminate after

60

 minutes of inactivity ⓘ

Tags ⓘ

Add tags

Key

Value

Add

> Automatically added tags

▸ Advanced options

Create compute

Cancel

Figuur (5.37)

Configuratie van compute resource

Voordat notebooks en jobs uitgevoerd kunnen worden zal er eerst een compute resource aangemaakt moeten worden. Hierbij zal de databricks runtime version op 11.3 LTS gezet moeten worden zodat we Apache Spark 3.3.0 kunnen gebruiken. Dit omdat we gebruik gaan maken van “spark-cdm-connector”. Ook hebben we ingesteld dat de cluster zichzelf zal uitschakelen na 60 minuten om onnodige kosten te voorkomen.

Install library
Send feedback for library
X

Library Source ⓘ

☐ Workspace
☐ File Path/ADLS
☐ PyPI
☐ Maven
☐ CRAN
☒ DBFS

Library files stored in DBFS can be modified by any workspace user. If you want to prevent changes, you should load libraries from [Workspace Files](#) or [ADLS](#). DBFS libraries are supported in Databricks Runtime 14.3 LTS and below.

Library Type

☒ JAR
☐ Python Whl

spark-cdm-connector-assembly-synapse-
spark3.2

Remove file

Cancel
Install

Figuur (5.38)

Installatie van “spark-cdm-connector”

Ten slotte moet de [jar file](#) van “spark-cdm-connector” geïnstalleerd worden in het aangemaakte compute resource om gebruik te kunnen maken van het Common Data Model in onze pipeline.

5.2.2. Collaboration en source control

Management en permissions

Workspace settings / Identity and access / Users

Status	Email	Name	
	koen.vandamme@net-it.be	Koen Van Damme	⋮
	loeka.lievens@net-it.be	Loeka Lievens	⋮
	loekalievens@hotmail.com		⋮

< Previous
 Next >
 20 / page

Figuur (5.39)

Gebruikers toevoegen/verwijderen in Azure Databricks

Gebruikers die behoren tot het AAD directory van de Azure Databricks environment kunnen makkelijk toegevoegd worden met behulp van het e-mailadres van de gebruiker.

Workspace settings / Identity and access / Users /

Edit user

Email loekalievens@hotmail.com

[Remove user](#)

Entitlements

Admin access
Can manage this workspace and its users, groups, resources, and settings On ☒

Workspace access
Can access data engineering and ML environments On ☒

Databricks SQL access
Can access SQL environment On ☒

Unrestricted cluster creation
Can create clusters; when disabled, the user is restricted to access granted by cluster policies On ☒

Figuur (5.40)

Rechten wijzigen van gebruikers in Azure Databricks

Rechten van gebruikers kunnen gewijzigd worden.

Workspace settings / Identity and access /

Groups

[Add group](#)

Name	Members	Source
admins	3	System ⓘ
users	3	System ⓘ

< Previous Next > 20 / page

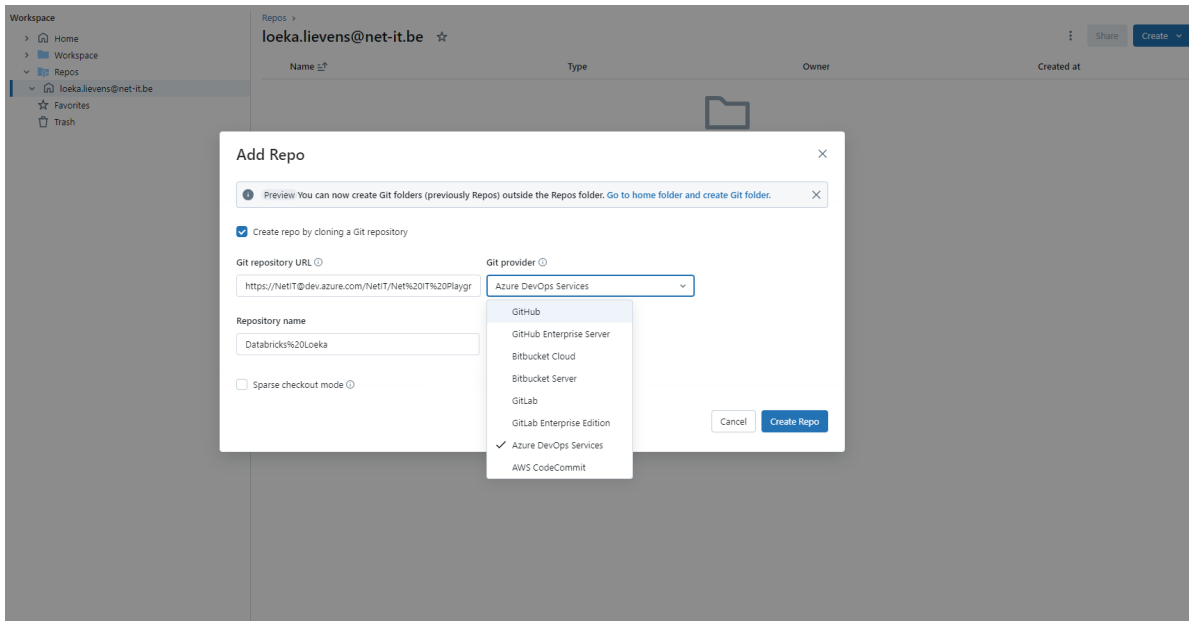
Figuur (5.41)

Groepen toevoegen of verwijderen in Azure Databricks

Er kunnen groepen toegevoegd of verwijderd worden aan Azure Databricks. Gebruikers kunnen dan aan deze groepen toegevoegd worden. Dit vereenvoudigt het om toegang toe te wijzen aan werkruimten, gegevens en andere objecten.

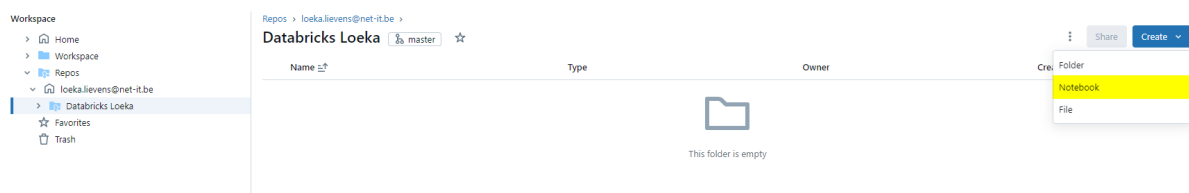
Source control

Databricks folders is een visuele Git client binnen Azure Databricks om gebruik te kunnen maken van source control.

**Figuur (5.42)**

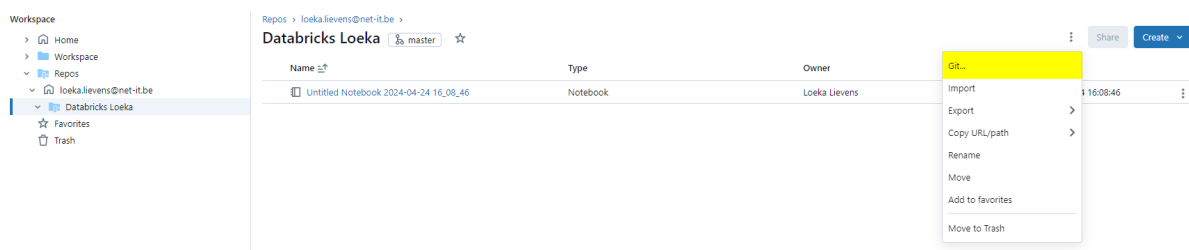
Clone Git repository in Databricks folders

Het geeft de optie om verschillende Git providers te gaan gebruiken. Doordat er binnen Net IT gewerkt wordt met Azure DevOps Services zal hier voor gekozen worden.

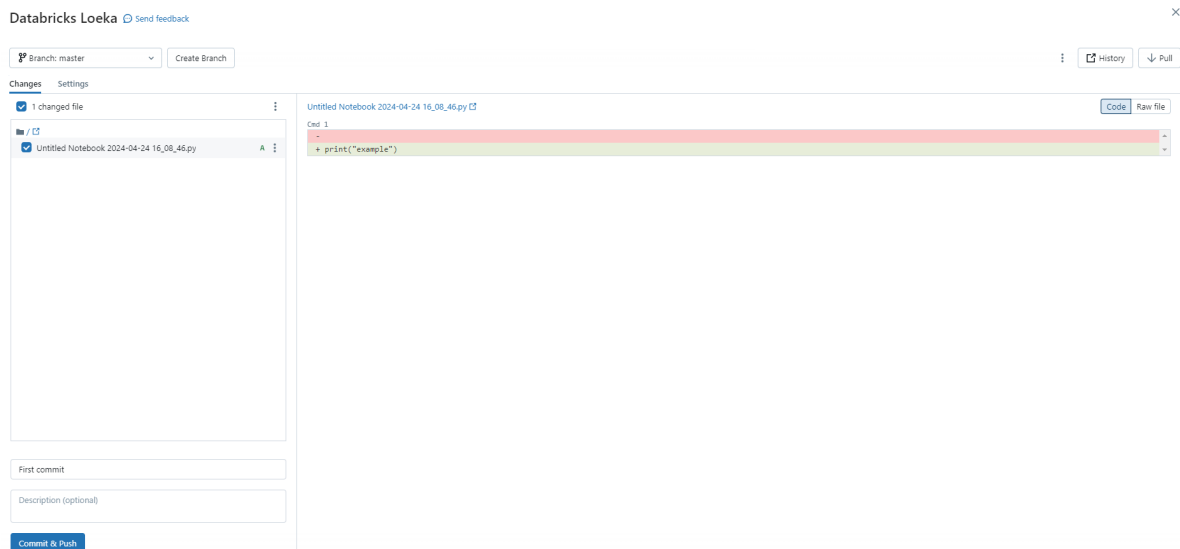
**Figuur (5.43)**

Aanmaken van een notebook in databricks

Ter illustratie zal er een voorbeeld notebook aangemaakt worden.

**Figuur (5.44)**

Commit en push in Databricks

**Figuur (5.45)**

Commit en push in Databricks

Deze notebook kan dan gecommited worden naar Git.

5.2.3. IDE integratie

Met behulp van Databricks Connect kan er vanuit Visual Studio Code, PyCharm, RStudio Desktop, IntelliJ IDEA, notebook servers en andere applicaties geconnecteerd worden met Databricks clusters. Hierdoor kan bijvoorbeeld Python-code zowel lokaal als op Databricks clusters uitgevoerd worden. Daarnaast kan code ook gesynchroniseerd worden met een Databricks workspace. Ook kan er makkelijk gedebugged worden met behulp van Databricks Connect.

5.2.4. Infrastructure as code (IaC)

Ook Azure Databricks kan met behulp van Azure Resource Manager (ARM) templates aangemaakt worden. Hierbij is het grote verschil wel dat bijvoorbeeld het aanmaken van clusters niet ondersteund worden. Er kan wel gebruik gemaakt worden van terraform om dit te implementeren. Daarnaast kan er ook gebruik gemaakt worden van Databricks Asset Bundles (DABs). Deze maakt het mogelijk om CI/CD te gaan implementeren met behulp van YAML syntax. Een bundle bevat dus de nodige cloud infrastructuur en workspace configurations, source files zoals bijvoorbeeld notebooks en Python files, definities voor Databricks resources en unit tests en integration tests. Met behulp van de Databricks CLI kunnen bundles gepubliceerd worden van uit de command line.

5.2.5. Ophalen van data uit Azure Data Lake

Register an application ...

* Name

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

bp-loeka-2



Supported account types

Who can use this application or access this API?

- ☒ Accounts in this organizational directory only (Net IT NV only - Single tenant)
- ☐ Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant)
- ☐ Accounts in any organizational directory (Any Microsoft Entra ID tenant - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- ☐ Personal Microsoft accounts only

[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.

Select a platform



e.g. <https://example.com/auth>

Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from [Enterprise applications](#).

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

Register

Figuur (5.46)

Aanmaken van app registration

Door in Microsoft Azure naar "App registrations" te navigeren kan er een nieuwe app registration aangemaakt worden.

Add a client secret



Description

secret

Expires

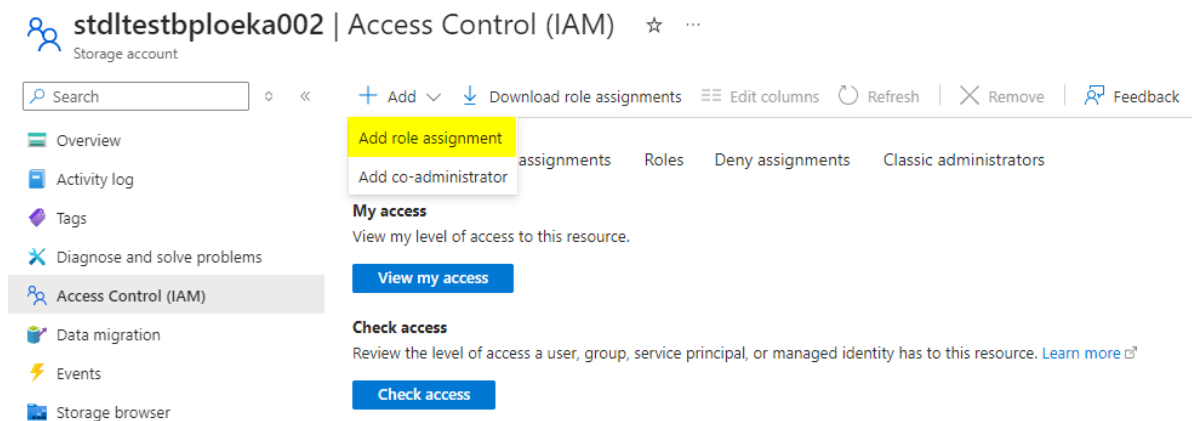
730 days (24 months)



Figuur (5.47)

Aanmaken client secret

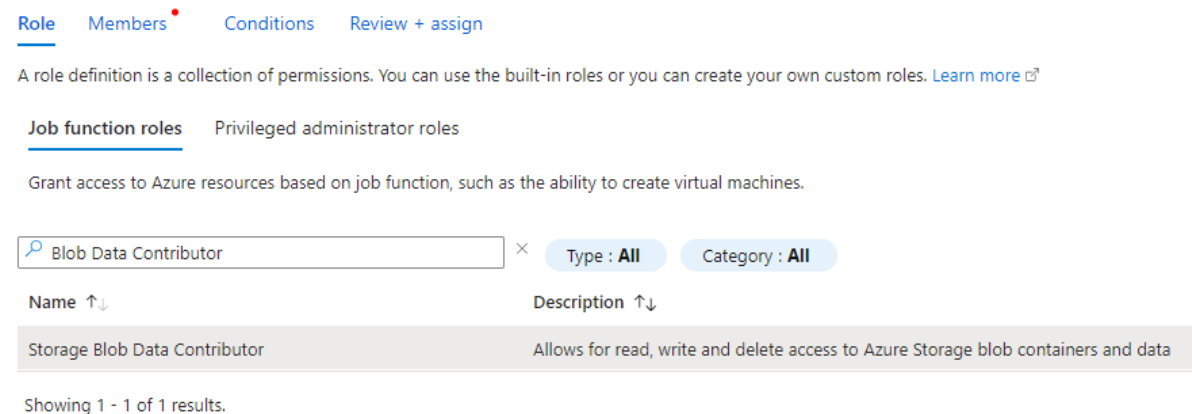
Door naar “Certificates & secrets” in de app registration te navigeren kan er een client secret aangemaakt worden. Het is belangrijk om deze client secret op te slaan.



Figuur (5.48)

Role assignment toevoegen aan storage account

Door naar “Access Control (IAM)” te navigeren van het storage account waarmee we een connectie willen maken, kunnen we nu een role assignment gaan toevoegen.



Figuur (5.49)

Job function role kiezen voor role assignment

Als job function role moet er voor “Storage Blob Data Contributor” gekozen worden.

Role **Members** Conditions Review + assign


Members

Showing a filtered list of roles because your permissions include a condition. [Learn more](#)
[View my access](#)

Selected role Storage Blob Data Contributor

Assign access to
☒ User, group, or service principal
☐ Managed identity

Members [+ Select members](#)

Name	Object ID	Type	
bp-loeka-2	6a06fcfb-1331-42d6-975b-8bec1b86928a	App	

Description

Optional

Figuur (5.50)

Members kiezen voor role assignment

Bij “Members” kiezen we voor de service principal die we net hebben aangemaakt. Dit zorgt er voor dat de app registration die we net hebben aangemaakt toegang heeft tot het data lake.

Basics Access configuration Networking Tags Review + create

Azure Key Vault is a cloud service used to manage keys, secrets, and certificates. Key Vault eliminates the need for developers to store security information in their code. It allows you to centralize the storage of your application secrets which greatly reduces the chances that secrets may be leaked. Key Vault also allows you to securely store secrets and keys backed by Hardware Security Modules or HSMs. The HSMs used are Federal Information Processing Standards (FIPS) 140-2 Level 2 validated. In addition, key vault provides logs of all access and usage attempts of your secrets so you have a complete audit trail for compliance.

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Visual Studio Enterprise Subscription – MPN - Spoke 2
Resource group *	LoekaBP
	Create new

Instance details

Key vault name * ⓘ	bp-loeka-kv
Region *	West Europe
Pricing tier * ⓘ	Standard

Recovery options

Soft delete protection will automatically be enabled on this key vault. This feature allows you to recover or permanently delete a key vault and secrets for the duration of the retention period. This protection applies to the key vault and the secrets stored within the key vault.

To enforce a mandatory retention period and prevent the permanent deletion of key vaults or secrets prior to the retention period elapsing, you can turn on purge protection. When purge protection is enabled, secrets cannot be purged by users or by Microsoft.

Figuur (5.51)

Aanmaken van key vault

Doordat we de client secret die we net hebben aangemaakt niet in onze databricks code willen zetten zal er een key vault aangemaakt worden. Dit kan door in Microsoft Azure naar “Key vaults” te navigeren. Bij het aanmaken van een key vault moet er een subscription en resource group gekozen worden. Er kan een nieuwe resource group aangemaakt worden of een reeds bestaande gekozen worden. Daarnaast moet er een naam, gewenste regio en pricing tier gekozen worden.

Permission model

Grant data plane access by using a [Azure RBAC](#) or [Key Vault access policy](#)

- ☐ [Azure role-based access control \(recommended\)](#) ⓘ
- ☒ [Vault access policy](#) ⓘ

Figuur (5.52)

Permission model van key vault

Onder “Access configuration” wordt er gekozen voor “Vault access policy” als permission model.

Basics Access configuration **Networking** Tags Review + create

You can connect to this key vault either publicly, via public IP addresses or service endpoints, or privately, using a private endpoint.

Enable public access ☒

Public Access

Allow access from:

☐ All networks

☒ Selected networks

ⓘ Only networks you choose can access this key vault. [Learn more](#)

Virtual networks

Allow selected virtual networks to connect to your resource securely and directly using service endpoints [Learn more](#)

+ Add a virtual network ▾

Virtual network	Subnet	Resource group	Subscription
No virtual networks are selected.			

Exception

Enabling access to resources requires you allow trusted Microsoft services to bypass firewall.

☒ Allow trusted Microsoft services to bypass this firewall

ⓘ This setting is related to firewall only. In order to access this key vault, the trusted service must also be given explicit permissions the Access Policies section. [Learn more](#)

Figuur (5.53)

Networking configuratie van key vault

Er kan gekozen worden om enkel geselecteerde networks public access te geven of om public access uit te schakelen. Belangrijk is dat “Allow trusted Microsoft services to bypass this firewall” aangevinkt staat.

Upload options: Manual

Name: AppKey ✓

Secret value: ***** ✓

Content type (optional):

Set activation date: ☐

Set expiration date: ☐

Enabled: Yes

Tags: 0 tags

Figuur (5.54)

Toevoegen van een secret in key vault

Door naar “Secrets” te navigeren in de key vault kan het client secret toegevoegd worden.

bp-loeka-kv | Properties

Search: Save Discard changes Refresh

Overview Activity log Access control (IAM) Tags Diagnose and solve problems Access policies Events Objects Keys Secrets Certificates Settings Access configuration Networking Microsoft Defender for Cloud

Properties

Locks

Name	bp-loeka-kv
Sku (Pricing tier)	Standard
Location	westeurope
Vault URI	https://bp-loeka-kv.vault.azure.net/
Resource ID	/subscriptions/7d6782ce-b3be-4d79-8b60-d81b84184144/resourceGroups/LoekaBP/providers/Microsoft.KeyVault/vaults/bp-loeka-kv
Subscription ID	7d6782ce-b3be-4d79-8b60-d81b84184144
Subscription Name	Visual Studio Enterprise Subscription – MPN - Spoke 2
Directory ID	118054c8-f9ce-4058-a30c-77ee43ef2918
Directory Name	Net IT NV

Soft delete Soft delete has been enabled on this key vault

Days to retain deleted vaults: 90

Purge protection: ☒ Disable purge protection (allow key vault and objects to be purged during retention period) ☐ Enable purge protection (enforce a mandatory retention period for deleted vaults and vault objects)

Figuur (5.55)

Opzoeken van properties van key vault

Door naar “Properties” te navigeren kunnen we nu “Vault URI” en “Resource ID” terugvinden. Deze zijn belangrijk voor de volgende stap.

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 ?

KeyVault

Manage Principal ?

All Users

Azure Key Vault ?

DNS Name

https://bp-loeka-kv.vault.azure.net/

Resource ID

/subscriptions/7d6782ce-b3be-4d79-8b60-d81b84184144/resourceGroups/LoekaBF

Figuur (5.56)

Aanmaken van secret scope in databricks

Door naar “https://<databricks-instance>#secrets/createScope” te navigeren kunnen we nu een secret scope aanmaken in databricks. Hiervoor hebben we “Vault URI” en “Resource ID” nodig uit de vorige stap.

```
1 app_key = dbutils.secrets.get(scope = "KeyVault", key = "AppKey")
```

We kunnen nu in onze notebooks met bovenstaande code secrets zoals bijvoorbeeld “AppKey” ophalen uit key vault.

```
1 storage_account = "<storage-account>"
2 app_id = "<application-id>"
3 app_key = dbutils.secrets.get(scope = "KeyVault", key = "AppKey")
4 tenant_id = "<directory-id>"
```

Om een connectie te kunnen maken met data lake hebben we “storage_account”, “app_id”, “app_key” en “tenant_id” nodig. Dit kunnen we doen door het volgende te vervangen:

- “<storage-account>” met de naam van het Azure storage account
- “<application-id>” met het Application (client) ID voor de Microsoft Entra ID application
- “<directory-id>” met het Directory (tenant) ID voor de Microsoft Entra ID application

```

1 df = spark.read.format("com.microsoft.cdm") \
2   .option("appId", app_id) \
3   .option("appKey", app_key) \
4   .option("tenantId", tenant_id) \
5   .option("storage", f"{storage_account}.dfs.core.windows.net") \
6   .option("manifestPath", "sourcedata/model.json") \
7   .option("mode", "permissive")

```

We kunnen nu gebruik maken van “spark-cdm-connector” door “com.microsoft.cdm” mee te geven. Het is belangrijk dat “mode” op “permissive” staat, dit zorgt er voor dat er “NULL” values assigned worden wanneer een CSV rij minder aantal kolommen heeft dan het entity schema.

```

1 spr = df.option("entity", "new_syndicalpremiumrequest") \
2   .load()

```

Met bovenstaande code kan een entiteit ingeladen worden in een variabele. Dit zal dus ook gebeuren met alle entiteiten die we hebben.

```

1 spr \
2   .select("Id", "new_personid", "new_bankaccountid",
3         "new_scannedrequestcode", "new_isdeclarationonhonour",
4         "new_requesttypeid", "new_formnumber",
5         "new_treatmentstate", "new_feedback",
6         "new_reasonforcontrol", "new_yearid", "createdon",
7         "new_idadm", "new_contributionamount", "new_premiumamount",
8         "new_paymentdate", "new_remarkgroupfinal", "new_groupid") \
9   .createOrReplaceTempView("new_syndicalpremiumrequest")

```

Voor elke entiteit die we hebben ingeladen in een variabele zullen we nu de nodige kolommen gaan selecteren. Hierna kan er een temporary view aangemaakt worden voor de entiteiten. Daardoor kunnen we later queries gaan uitvoeren in SQL om zo transformaties uit te voeren.

5.2.6. Mogelijkheid tot debuggen

SQL: pyspark.sql.dataframe.DataFrame = [RefYear: integer, Group: string ... 21 more fields]

Table + New result table: ON

	RefYear	Group	Gr	Id	EntryDate	Type	Form	.00 Contr
1	2018	f4a8fede-b94c-e911-a848-000d3a2ddcbb	zz5bs	045eb4b9-fb9c-ea11-a812-000d3a2d0171	2019-07-21T22:00:00.000+00:...	FORM	6x8e-5c02n7-wl	
2	2022	f4a8fede-b94c-e911-a848-000d3a2ddcbb	zz5bs	6d18cd1e-ece4-ed11-a7c7-6045bd90585b	2023-04-27T11:10:32.000+00:...	QR	nuu11	
3	2017	f4a8fede-b94c-e911-a848-000d3a2ddcbb	zz5bs	0013c7ca-6c9c-ea11-a812-000d3a2d0171	2018-07-23T22:00:00.000+00:...	FORM	hpox-6dj8ge-fc	
4	2018	f4a8fede-b94c-e911-a848-000d3a2ddcbb	zz5bs	38fa7b0d-f89c-ea11-a812-000d3a2c0643	2019-08-13T22:00:00.000+00:...	FORM	k0x4-4t37bz-t0	
5	2020	f4a8fede-b94c-e911-a848-000d3a2ddcbb	zz5bs	0785000f-36f0-eb11-bacb-0022487efb50	2021-07-29T06:27:34.000+00:...	FORM	4lb9-1wj0ow-8p	
6	2012	f4a8fede-b94c-e911-a848-000d3a2ddcbb	zz5bs	f07135e5-ac98-ea11-a812-000d3a2d0171	2013-09-02T22:00:00.000+00:...	FORM	8slh-dmyw4j-xe	
7	2013	f4a8fede-b94c-e911-a848-000d3a2ddcbb	zz5bs	aa0183a2-149a-ea11-a812-000d3a2d0171	2014-06-03T22:00:00.000+00:...	FORM	9rpf-zpryrf-2d	
8	2010	f4a8fede-b94c-e911-a848-000d3a2ddcbb	zz5bs	c1332c44-b397-ea11-a811-000d3a2c0643	2011-08-24T22:00:00.000+00:...	FORM	03qt-akrude-ne	
9	2010	f4a8fede-b94c-e911-a848-000d3a2ddcbb	zz5bs	255ab30c-a997-ea11-a811-000d3a2d0171	2011-10-20T22:00:00.000+00:...	FORM	99zg-uycc2o-5z	
10	2012	f4a8fede-b94c-e911-a848-000d3a2ddcbb	zz5bs	6002a0cd-ab98-ea11-a812-000d3a2c0643	2013-07-25T22:00:00.000+00:...	FORM	fs7p-ajicy6-7o	
11	2011	f4a8fede-b94c-e911-a848-000d3a2ddcbb	zz5bs	aa0c858a-3798-ea11-a812-000d3a2c0643	2012-08-05T22:00:00.000+00:...	FORM	46p5-t6xtut-lp	
12	2015	f4a8fede-b94c-e911-a848-000d3a2ddcbb	zz5bs	8c99aa8d-6e9b-ea11-a812-000d3a2d0171	2016-04-26T22:00:00.000+00:...	FORM	4vn7-xf3ca4-74	
13	2011	f4a8fede-b94c-e911-a848-000d3a2ddcbb	zz5bs	5d45313a-2e98-ea11-a812-000d3a2d0171	2012-07-08T22:00:00.000+00:...	FORM	a4zj-ugtr9g-gy	
14	2018	f4a8fede-b94c-e911-a848-000d3a2ddcbb	zz5bs	fcd1871f-f99c-ea11-a812-000d3a2c0643	2019-08-05T22:00:00.000+00:...	FORM	jqdc-mrizzg-1e	

6,335 rows | Truncated data | 1.71 minutes runtime | Refreshed 6 minutes ago

Figuur (5.57)

Debuggen in Azure Databricks

In Azure Databricks kan er telkens geprint worden welke output bepaalde code geeft. Bij SQL statements zal dit bijvoorbeeld resulteren in een tabel. Op deze tabel kan er gefilterd worden of kunnen zelfs visualisaties gemaakt worden. Het maximum aantal resultaten dat hier bij getoond zal worden is 10.000 rijen of 2MB. In Databricks Runtime 12.2 LTS kan dit maximum verhoogd worden. Daarnaast is er ook een interactieve debugger in “Public Preview”. Hier gaan we verder niet op ingaan aangezien deze Databricks Runtime version 13.3 LTS of hoger nodig heeft en wij een oudere versie gebruiken om Common Data Model te gebruiken.

5.2.7. Belangrijkste transformaties

Determinatie van welke groepen de premie in hun bestand krijgen

```

1 CREATE
2 OR REPLACE TEMPORARY VIEW spr_stage_1 AS
3 SELECT
4     YEAR(spr.createdon) EntryYear,
5     YEAR(GETDATE()) CalendarYear,
6     CAST(ny.new_year AS INTEGER) RefYear,
7     IF(spr.new_scannedrequestcode = 552010000,
8         IF(spr.new_isdeclarationonhonour, "DOH", "FORM"),
9         IF(spr.new_scannedrequestcode = 552010001, "QR",
10            "UNKNOWN")) Type,
11     IF(spr.new_requesttypeid =
12         "d026f6f9-fc82-ea11-a811-000d3a2d0171",
13         CONCAT(SUBSTRING(new_formnumber, 1, 4), "-",
14             SUBSTRING(new_formnumber, 5, 6), "-",
15             SUBSTRING(new_formnumber, 11, 20)), spr.new_formnumber)
16     Form,

```

```

10
11  IF(rpr.new_treatmentstate = 552010004,
12      CASE
13          WHEN spr.new_treatmentstate = 552010000 THEN "1"
14          WHEN spr.new_treatmentstate = 552010001 THEN "10"
15          WHEN spr.new_treatmentstate = 552010002 THEN "1"
16          WHEN spr.new_treatmentstate = 552010003 THEN "10"
17          WHEN spr.new_treatmentstate = 552010004 THEN "5"
18          WHEN spr.new_treatmentstate = 552010005 THEN "11"
19          WHEN spr.new_treatmentstate = 552010006 THEN "1"
20          WHEN spr.new_treatmentstate = 552010007 THEN "10"
21          WHEN spr.new_treatmentstate = 552010008 THEN "8"
22          ELSE ""
23      END,
24      CASE
25          WHEN spr.new_treatmentstate = 552010000 THEN "1"
26          WHEN spr.new_treatmentstate = 552010001 THEN "2"
27          WHEN spr.new_treatmentstate = 552010002 THEN "1"
28          WHEN spr.new_treatmentstate = 552010003 THEN "2"
29          WHEN spr.new_treatmentstate = 552010004 THEN "5"
30          WHEN spr.new_treatmentstate = 552010005 THEN "11"
31          WHEN spr.new_treatmentstate = 552010006 THEN "1"
32          WHEN spr.new_treatmentstate = 552010007 THEN "2"
33          WHEN spr.new_treatmentstate = 552010008 THEN "8"
34          ELSE ""
35      END
36  ) Status,
37  IF(spr.new_feedback, "FEEDBACK", spr.new_reasonforcontrol)
38      AvalonRemark,
39  spr.new_personid,
40  spr.new_idadm,
41  spr.new_yearid,
42  spr.id,
43  spr.createdon,
44  spr.new_contributionamount,
45  spr.new_premiumamount,
46  spr.new_paymentdate,
47  ba.new_bankaccount,
48  spr.new_remarkgroupfinal,
49  p.new_firstname,

```

```

50     p.new_lastname,
51     p.new_dateofbirth
52 FROM new_syndicalpremiumrequest spr
53 LEFT JOIN new_person p ON spr.new_personid = p.new_personid
54 LEFT JOIN new_bankaccount ba ON spr.new_bankaccountid =
    ba.new_bankaccountid
55 LEFT JOIN related_premium_requests rpr ON spr.Id =
    rpr.new_syndicalpremiumrequestid
56 LEFT JOIN new_year ny ON spr.new_yearid = ny.new_yearid;

```

De tabel “new_year” wordt opnieuw gejoind op de tabel “new_syndicalpremiumrequest” met behulp van de kolom “new_yearid”. Daarnaast worden ook hier de drie kolommen “EntryYear”, “CalendarYear” en “RefYear” berekent. Deze worden op dezelfde manier berekent als bij de transformatie in Azure Data Factory.

```

1 CREATE
2 OR REPLACE TEMPORARY VIEW spr_stage_2 AS
3 SELECT
4     explode(array(nm.new_groupid, noy.new_groupid)) new_groupid,
5     spr.*,
6     nm.new_groupprefix new_groupprefix
7 FROM spr_stage_1 spr
8 LEFT JOIN new_membership_stage_1 nm
9     ON (spr.CalendarYear = nm.new_year OR spr.EntryYear =
        nm.new_year OR RefYear = nm.new_year) AND spr.new_personid
        = nm.new_personid
10 LEFT JOIN new_organizationyear_stage_1 noy
11     ON spr.new_idadm = noy.new_idadmaux AND spr.new_yearid =
        noy.new_yearid;

```

De tabellen “new_membership” en “new_organizationyear” worden gejoind op de tabel “new_syndicalpremiumrequest” om de groepen te gaan bepalen. Zoals te zien in de code snippet hierboven noemen deze tabellen “new_membership_stage_1” en “new_organizationyear_stage_1”. Dit komt doordat er reeds andere transformaties zijn geweest.

Bij de tabel “new_membership” wordt er, net zoals in Azure Data Factory, gekeken of “CalendarYear”, “EntryYear” of “RefYear” overeenkomt met het jaartal van de membership. Daarnaast wordt er ook gekeken of personid overeen komt.

Bij de tabel “new_organizationyear” wordt er gejoind met behulp van IDADM en het id van het referentiejaar.

spr.id	nm.new_groupid	noy.new_groupid
101e4523-7c60-45ff-a928-087ca139d8f5	2ad31fb8-5117-4627-a2a3-a5fdbee5b9ae	null
f5ac28c0-6b80-412a-b0e6-f9a82feff9a1	f1015dd0-3096-478d-ae12-4226c101dd54	a2974e74-be93-4326-9d18-f4bb2abc0842

↓

spr.id	new_groupid
101e4523-7c60-45ff-a928-087ca139d8f5	2ad31fb8-5117-4627-a2a3-a5fdbee5b9ae
101e4523-7c60-45ff-a928-087ca139d8f5	null
f5ac28c0-6b80-412a-b0e6-f9a82feff9a1	f1015dd0-3096-478d-ae12-4226c101dd54
f5ac28c0-6b80-412a-b0e6-f9a82feff9a1	a2974e74-be93-4326-9d18-f4bb2abc0842

Figuur (5.58)

Voorbeeld van hoe de functie explode werkt.

Deze twee joins resulteren in twee nieuwe kolommen “nm.new_groupid” en “noy.new_groupid”. Met behulp van de “explode” functie zorgen we er voor dat deze twee kolommen één kolom zullen worden. Één rij zal dan bijvoorbeeld twee rijen worden. Deze twee rijen zijn dan volledig hetzelfde met het verschil dat de kolom “new_groupid” voor de eerste rij de waarde van “nm.new_groupid” heeft en voor de tweede rij de waarde van “noy.new_groupid”.

```

1 CREATE
2 OR REPLACE TEMPORARY VIEW spr_stage_3 AS
3 SELECT
4     FIRST(id) id,
5     FIRST(new_groupid) new_groupid,
6     FIRST(new_personid) new_personid,
7     FIRST(CalendarYear) CalendarYear,
8     FIRST(EntryYear) EntryYear,
9     FIRST(RefYear) RefYear,
10    FIRST(new_groupprefix) new_groupprefix,
11    FIRST(createdon) createdon,
12    FIRST(new_contributionamount) new_contributionamount,
13    FIRST(new_premiumamount) new_premiumamount,

```

```

14     FIRST(new_paymentdate) new_paymentdate,
15     FIRST(new_bankaccount) new_bankaccount,
16     FIRST(new_remarkgroupfinal) new_remarkgroupfinal,
17     FIRST(AvalonRemark) AvalonRemark,
18     FIRST(new_firstname) new_firstname,
19     FIRST(new_lastname) new_lastname,
20     FIRST(new_dateofbirth) new_dateofbirth,
21     FIRST(Type) Type,
22     FIRST(Form) Form,
23     FIRST(Status) Status,
24     FIRST(new_idadm) new_idadm
25 FROM spr_stage_2
26 WHERE id IS NOT NULL AND new_groupid IS NOT NULL AND RefYear IS NOT
    NULL
27 GROUP BY (id, new_groupid, RefYear);

```

Ten slotte wordt er een “GROUP BY” clause gebruikt om er voor te zorgen dat een premie niet twee keer naar dezelfde groep voor dat referentiejaar wordt gestuurd. Daarnaast wordt er ook gefilterd zodat er geen premies zijn zonder id, groupid of referentiejaar. Met behulp van de “FIRST” aggregate function selecteren we steeds de eerste waarde voor de nodige kolommen. We hebben nu een tabel met alle premies waarvan we weten naar welke groep voor welk referentiejaar ze gestuurd moeten worden.

Bepalen van de kolom “Gr” voor een premie

```

1 CREATE
2 OR REPLACE TEMPORARY VIEW nm_grouped_by AS
3 SELECT
4     FIRST(new_groupprefix, TRUE) new_groupprefix,
5     FIRST(new_personid) new_personid,
6     FIRST(new_year) new_year
7 FROM new_membership_stage_1
8 WHERE new_groupprefix IS NOT NULL
9 GROUP BY (new_personid, new_year);
10
11 CREATE
12 OR REPLACE TEMPORARY VIEW spr_stage_4 AS
13 SELECT
14     spr.*,
15     CASE

```

```

16      WHEN spr.new_groupprefix IS NOT NULL THEN
          spr.new_groupprefix
17      WHEN nmCalendar.new_groupprefix IS NOT NULL THEN
          nmCalendar.new_groupprefix
18      WHEN nmEntry.new_groupprefix IS NOT NULL THEN
          nmEntry.new_groupprefix
19      WHEN nmRef.new_groupprefix IS NOT NULL THEN
          nmRef.new_groupprefix
20      ELSE NULL
21  END Gr
22 FROM spr_stage_3 spr
23 LEFT JOIN nm_grouped_by nmCalendar ON spr.new_personid =
    nmCalendar.new_personid AND spr.CalendarYear =
    nmCalendar.new_year
24 LEFT JOIN nm_grouped_by nmEntry ON spr.new_personid =
    nmEntry.new_personid AND spr.EntryYear = nmEntry.new_year
25 LEFT JOIN nm_grouped_by nmRef ON spr.new_personid =
    nmRef.new_personid AND spr.RefYear = nmRef.new_year;

```

Voor het bepalen van de kolom “Gr” in databricks wordt er eerst een temporary view “nm_grouped_by” aangemaakt waarbij er gegroepeerd wordt op “new_personid” en “new_year”. Dit zal er voor zorgen dat de joins die we uitvoeren zullen resulteren in één enkele match (of geen). De tweede parameter in de functie “FIRST” (aangeduid in code snippet) zorgt er voor dat “NULL” values genegeerd worden. De eerste waarde die niet “NULL” is voor “new_groupprefix” zal dus geselecteerd worden. De temporary view die we hebben aangemaakt wordt nu drie keer gejoind op de tabel “spr_stage_3” aan de hand van “new_personid” en “CalendarYear”, “EntryYear” of “RefYear”. “Gr” zal bepaald worden door eerst te kijken of “spr.new_groupprefix” niet “NULL” is, wanneer deze wel “NULL” is zal er gekeken worden naar “nmCalendar.new_groupprefix”, “nmEntry.new_groupprefix” en ten slotte naar “nmRef.new_groupprefix”.

Bepalen van de kolommen “AntheaNumber” en “MembershipNumber” voor een premie

```

1 %sql
2 CREATE
3 OR REPLACE TEMPORARY VIEW nm_grouped_by_2 AS
4 SELECT
5     FIRST(new_antheanumber, TRUE) new_antheanumber,
6     FIRST(new_newmembershipnumber, TRUE) new_newmembershipnumber,
7     FIRST(new_groupid) new_groupid,

```

```

8      FIRST(new_year) new_year,
9      FIRST(new_personid) new_personid
10 FROM new_membership_stage_1
11 WHERE new_antheanumber IS NOT NULL OR new_newmembershipnumber IS
      NOT NULL
12 GROUP BY (new_groupid, new_year, new_personid);
13
14 CREATE
15 OR REPLACE TEMPORARY VIEW spr_stage_5 AS
16 SELECT
17     spr.*,
18     CASE
19         WHEN Gr IS NULL THEN NULL
20         WHEN nmCalendar.new_antheanumber IS NOT NULL THEN
21             nmCalendar.new_antheanumber
22         WHEN nmEntry.new_antheanumber IS NOT NULL THEN
23             nmEntry.new_antheanumber
24         WHEN nmRef.new_antheanumber IS NOT NULL THEN
25             nmRef.new_antheanumber
26         ELSE NULL
27     END AntheaNumber,
28     CASE
29         WHEN Gr IS NULL THEN NULL
30         WHEN nmCalendar.new_newmembershipnumber IS NOT NULL THEN
31             nmCalendar.new_newmembershipnumber
32         WHEN nmEntry.new_newmembershipnumber IS NOT NULL THEN
33             nmEntry.new_newmembershipnumber
34         WHEN nmRef.new_newmembershipnumber IS NOT NULL THEN
35             nmRef.new_newmembershipnumber
36         ELSE NULL
37     END MembershipNumber
38 FROM spr_stage_4 spr
39 LEFT JOIN nm_grouped_by_2 nmCalendar ON spr.new_groupid =
      nmCalendar.new_groupid AND spr.CalendarYear =
      nmCalendar.new_year AND spr.new_personid =
      nmCalendar.new_personid
40 LEFT JOIN nm_grouped_by_2 nmEntry ON spr.new_groupid =
      nmEntry.new_groupid AND spr.EntryYear = nmEntry.new_year AND
      spr.new_personid = nmEntry.new_personid
41 LEFT JOIN nm_grouped_by_2 nmRef ON spr.new_groupid =
      nmRef.new_groupid AND spr.RefYear = nmRef.new_year AND
      spr.new_personid = nmRef.new_personid;

```


Voor het bepalen van de kolommen “AntheaNumber” en “MembershipNumber” wordt er eerst een temporary view “nm_grouped_by_2” aangemaakt waarbij er gegroepeerd wordt op “new_groupid”, “new_year” en “new_personid”. Dit zal er voor zorgen dat de joins die uitgevoerd worden resulteren in één enkele match (of geen). De functie “FIRST” zorgt er weer voor dat “NULL” values genegeerd worden. De aangemaakte temporary view wordt nu drie keer gejoind op de tabel “spr_stage_4” aan de hand van “new_groupid”, “new_personid” en “CalendarYear”, “EntryYear” of “RefYear”. Wanneer de kolom “Gr”, “NULL” is, zal “AntheaNumber” en “MembershipNumber” ook “NULL” zijn. Wanneer dit niet zo is wordt eerst gekeken of de waarde van de join aan de hand van “CalendarYear” niet “NULL” is. Wanneer dit wel “NULL” is zal er vervolgens gekeken worden naar de join aan de hand van “EntryYear” en vervolgens naar de join aan de hand van “RefYear”.

5.2.8. Schrijven van data naar Azure Data Lake

```
1 SELECT
2     RefYear,
3     new_groupid `Group`,
4     Gr,
5     id Id,
6     createdon EntryDate,
7     Type,
8     Form,
9     new_contributionamount Contribution,
10    new_premiumamount Prem,
11    Status,
12    new_paymentdate PayDate,
13    new_bankaccount Account,
14    new_remarkgroupfinal Remark,
15    AvalonRemark,
16    new_lastname `Name`,
17    new_firstname FirstName,
18    new_dateofbirth Birthday,
19    CalendarYear,
20    EntryYear,
21    AntheaNumber,
22    MembershipNumber,
23    new_idadm IdAdm,
24    new_groupprefix IDADM_Gr
25 FROM spr_stage_5;
```

De laatste SQL statement die de data bevat die naar Azure Data Lake geschreven moet worden wordt niet in temporary view gestoken. Databricks zal de output van deze SQL statement automatisch in een PySpark data frame genaamd “_sqldf” steken.

```

1 spark.conf.set(f"fs.azure.account.auth.type.{storage_account}.dfs.core.windows.net
    "OAuth")
2 spark.conf.set(f"fs.azure.account.oauth.provider.type.{storage_account}.dfs.core.w
    "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")
3 spark.conf.set(f"fs.azure.account.oauth2.client.id.{storage_account}.dfs.core.wind
    app_id)
4 spark.conf.set(f"fs.azure.account.oauth2.client.secret.{storage_account}.dfs.core
    app_key)
5 spark.conf.set(f"fs.azure.account.oauth2.client.endpoint.{storage_account}.dfs.co
    f"https://login.microsoftonline.com/{tenant_id}/oauth2/token")
6
7 _sqldf.coalesce(1).write.option("header",
    "true").mode("overwrite").csv(f"abfss://target@{storage_account}.dfs.core.wind

```

Met behulp van “_sqldf” in Python en de variabelen die we eerder hebben geïnitieerd, zoals te zien in sectie 5.2.5, wordt er terug naar Azure Data Lake geschreven. Hierbij wordt er gebruik gemaakt van “.coalesce(1)” om aan te geven dat er naar slechts één CSV bestand geschreven moet worden.

5.3. Validatie van output

Het exportbestand voor zowel Azure Data Factory als Azure Databricks kan in Microsoft Power BI ingeladen worden om te vergelijken of de data overeenkomt.

```

1 = Table.RemoveMatchingRows("#ADF", Table.ToRecords("#DATABRICKS"),
    { "Id", "Group", "RefYear" })

```

Bovenstaande Power Query gaat een tabel aanmaken met rijen die geen matches hebben voor de kolom “Id”, “Group” en “RefYear”. Wanneer dit dus een lege tabel geeft betekent dit dus dat beide exportbestanden even veel rijen hebben waarbij voor elke rij van Azure Data Factory een rij is in het bestand van Azure Databricks waarbij deze overeen komen. Er kunnen ook andere kolommen toegevoegd worden om deze te gaan valideren of het laatste argument kan weg gelaten worden om de volledige CSV bestanden te gaan valideren.

6

Uitvoeren van pipelines en verzamelen van resultaten

Voor het vergelijken van de kostprijs en performantie van beide pipelines in Azure Data Factory en Azure Databricks zullen de pipelines op verschillende manieren dagelijks uitgevoerd worden.

Compute Size	Core count
Small	4 Worker Cores + 4 Driver Cores
Medium	8 Worker Cores + 8 Driver Cores

Figuur (6.1)

Data flow runtimes voor het uitvoeren van de Azure Data Factory pipeline.

Worker type + Driver type	Core count	Memory
Standard_D4ads_v5	4 Worker Cores + 4 Driver Cores	16GB Worker Memory + 16GB Driver Memory
Standard_D8ads_v5	8 Worker Cores + 8 Driver Cores	16GB Worker Memory + 16GB Driver Memory

Figuur (6.2)

Clusters voor het uitvoeren van de Azure Databricks pipeline.

Zowel de Azure Data Factory pipeline als de Azure Databricks pipeline zullen dagelijks gerund worden op twee verschillende compute sizes. Hierbij wordt er telkens gekeken naar “4 Worker Cores + 4 Driver Cores” en “8 Worker Cores + 8 Driver Cores”.

6.1. Azure Data Factory

Het vinden van de kosten voor Azure Data Factory kan in Microsoft Cost Management terug gevonden worden. Hierbij kan er gefilterd worden per dag om de dagelijkse kosten voor het uitvoeren van de pipelines terug te vinden. In Azure Data Factory zelf kan er gekozen worden om een billing report op pipeline of factory level te tonen. Maar doordat er gebruik gemaakt wordt van Azure Data Flow in Azure Data Factory komen de kosten van beide pipelines nog steeds terecht in één enkele resource in het billing report. Hierdoor kan enkel de totale kostprijs voor het uitvoeren van de twee pipelines voor een bepaalde dag gevonden worden. Gelukkig kan in Azure Data Factory zelf de consumption per pipeline gevonden worden. Aan de hand hier van kunnen de kosten voor het uitvoeren van deze pipeline berekend worden.

Een voorbeeld hiervan is dat de totale kostprijs voor Azure Data Factory in het billing report op 5 mei €0,72 bedraagt. Wanneer we in Azure Data Factory naar de consumption per pipeline gaan kijken vinden we het volgende terug:

	Quantity	Unit
Pipeline orchestration		
Activity runs	1	Activity runs
Data flow		
Data Flow	1.1259	vCore-hour

Figuur (6.3)

Consumption voor het uitvoeren van de pipeline met 4 Worker Cores + 4 Driver Cores

	Quantity	Unit
Pipeline orchestration		
Activity runs	1	Activity runs
Data flow		
Data Flow	1.7344	vCore-hour

Figuur (6.4)

Consumption voor het uitvoeren van de pipeline met 8 Worker Cores + 8 Driver Cores

Beide pipelines hebben één activity run. Doordat de prijs per 1.000 activity runs slechts €0,937 bedraagt betekent dit dat de prijs voor één enkele activity run onder de €0,01 zit. Hierdoor zullen we hier geen rekening mee houden.

Wat wel belangrijk is, is het aantal vCore-hour er verbruikt is in Azure Data Flow. De kostprijs per vCore-hour in Azure Data Flow bedraagt €0.251. Aan de hand hiervan kunnen we dus de kostprijs per pipeline berekenen:

$$1,1259 \text{ vCore-hour} \times €0.251 = €0,2826009 \approx €0,28$$

$$1,7344 \text{ vCore-hour} \times €0.251 = €0,4353344 \approx €0,44$$

De kosten voor het uitvoeren van de pipeline met 4 Worker Cores + 4 Driver Cores bedraagt dus €0,28 en voor de pipeline met 8 Worker Cores + 8 Driver Cores bedraagt dit €0,44. Wanneer we deze kosten optellen komen we uit op de eerdere gevonden totale kost van €0,72. Elke dag zal dus de kostprijs op deze manier berekend worden. Ook de tijd die nodig is voor het uitvoeren van een pipeline en de cluster startup tijd vinden we terug in Azure Data Factory.

4 Worker Cores + 4 Driver Cores

Datum	Kost	Totale tijd	Cluster start-up tijd	Uitvoeringstijd
05/05	€ 0,28	10m	3m 11s	6m 49s
06/05	€ 0,29	10m 48s	2m 57s	7m 51s
07/05	€ 0,29	10m 2s	3m 9s	6m 53s

Figuur (6.5)

Kost en tijd voor het uitvoeren van de pipeline met 4 Worker Cores + 4 Driver Cores.

8 Worker Cores + 8 Driver Cores

Datum	Kost	Totale tijd	Cluster start-up tijd	Uitvoeringstijd
05/05	€ 0,44	7m 53s	3m 6s	4m 47s
06/05	€ 0,47	8m 6s	3m 24s	4m 42s
07/05	€ 0,43	7m 56s	2m 42s	5m 14s

Figuur (6.6)

Kost en tijd voor het uitvoeren van de pipeline met 8 Worker Cores + 8 Driver Cores.

6.2. Azure Databricks

Voor het vinden van de kostprijs voor de pipelines in Azure Databricks zijn er twee Databricks instances aangemaakt. Één voor de pipeline met 4 Worker Cores + 4 Driver Cores en één voor de pipeline met 8 Worker Cores + 8 Driver Cores. Hierdoor kan er in het billing report gekeken worden naar de kosten voor de pipelines apart. Belangrijk om hierbij te vermelden is dat, zoals te zien in figuur 5.36, bij het

aanmaken van Azure Databricks, er een Managed Resource Group name gekozen kan worden. Aan de hand van deze naam kan er gekeken worden naar de kosten van bijvoorbeeld virtual machines, storage accounts en virtual networks van de pipeline. Al deze kosten optellen voor een bepaalde dag zal dus de totale kost voor het uitvoeren van een pipeline uitkomen. De tijd zelf voor het uitvoeren van de pipelines vinden we terug in Databricks. Hierbij staat er geen cluster startup tijd maar deze kan teruggevonden worden met behulp van de Databricks REST API of Databricks CLI.

4 Worker Cores + 4 Driver Cores

Datum	Kost	Totale Tijd	Cluster start-up tijd	Uitvoeringstijd
05/05	€ 0,22	10m 44s	3m 48s	6m 56s
06/05	€ 0,21	10m 16s	3m 45s	6m 31s
07/05	€ 0,21	10m 20s	4m 8s	6m 12s

Figuur (6.7)

Kost en tijd voor het uitvoeren van de pipeline met 4 Worker Cores + 4 Driver Cores.

8 Worker Cores + 8 Driver Cores

Datum	Kost	Totale Tijd	Cluster start-up tijd	Uitvoeringstijd
05/05	€ 0,42	7m 17s	3m 22s	3m 55s
06/05	€ 0,42	7m 43s	3m 44s	3m 59s
07/05	€ 0,46	10m 30s	6m 27s	4m 3s

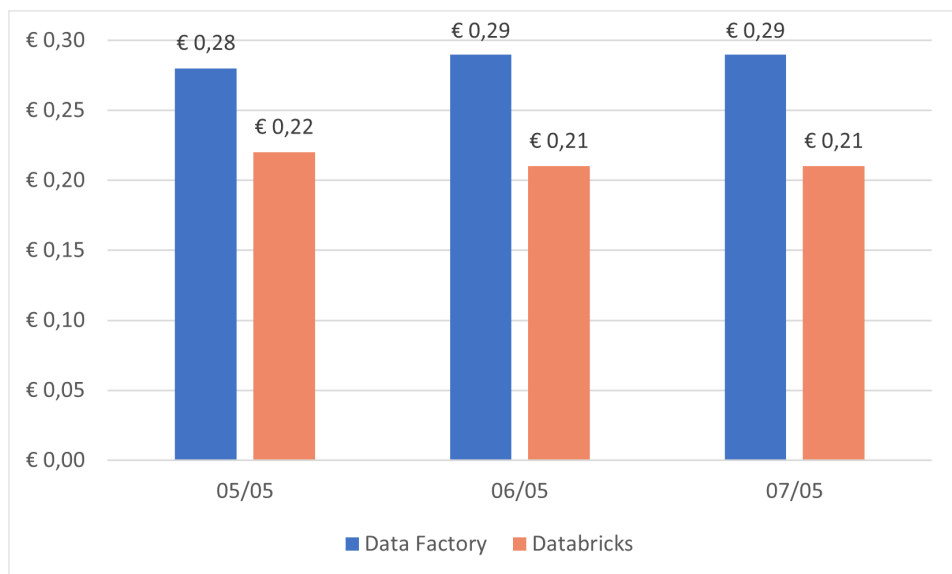
Figuur (6.8)

Kost en tijd voor het uitvoeren van de pipeline met 8 Worker Cores + 8 Driver Cores.

Zoals te zien in de rode tekst is de totale tijd voor het uitvoeren van de pipeline bij één van de runs veel groter dan bij de andere. Dit komt doordat de cluster startup tijd hierbij veel groter was.

6.3. Grafieken

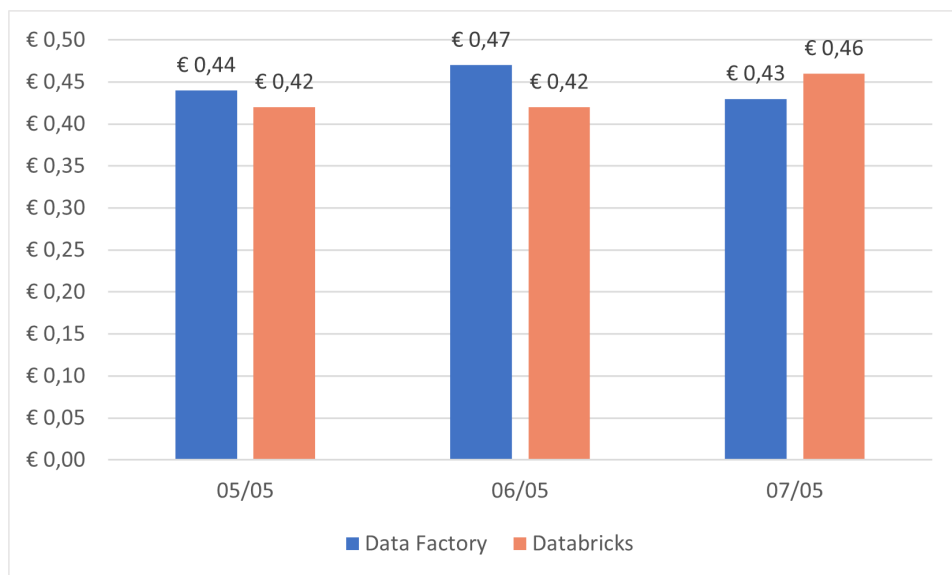
6.3.1. Kosten



Figuur (6.9)

Prijzen voor pipeline met 4 Worker Cores + 4 Driver Cores

Deze grafiek toont de kosten per run van Data Factory en Databricks voor de pipeline met 4 Worker Cores + 4 Driver Cores over drie dagen: 5 mei, 6 mei en 7 mei. Uit de gegevens blijkt dat de kosten per run voor Data Factory op al deze dagen iets hoger liggen dan die van Databricks.



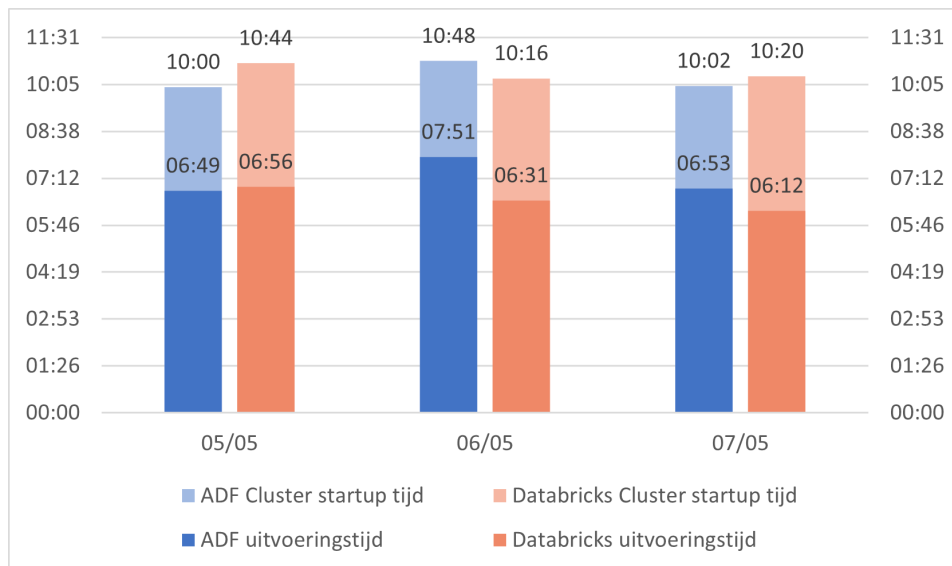
Figuur (6.10)

Prijzen voor pipeline met 8 Worker Cores + 8 Driver Cores

Ook hier liggen de kosten voor de pipeline met 8 Worker Cores + 8 Driver Cores iets

hoger bij Data Factory, met uitzondering dat op 7 mei de kosten van databricks hoger lagen. Dit komt waarschijnlijk doordat de cluster startup tijd ook hoger lag.

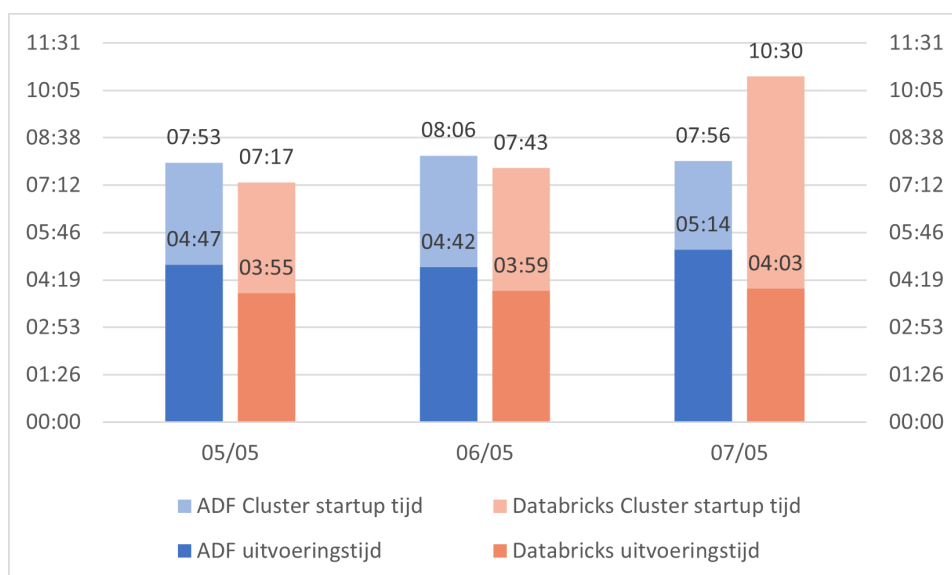
6.3.2. Performantie



Figuur (6.11)

Uitvoeringstijden voor pipeline met 4 Worker Cores + 4 Driver Cores

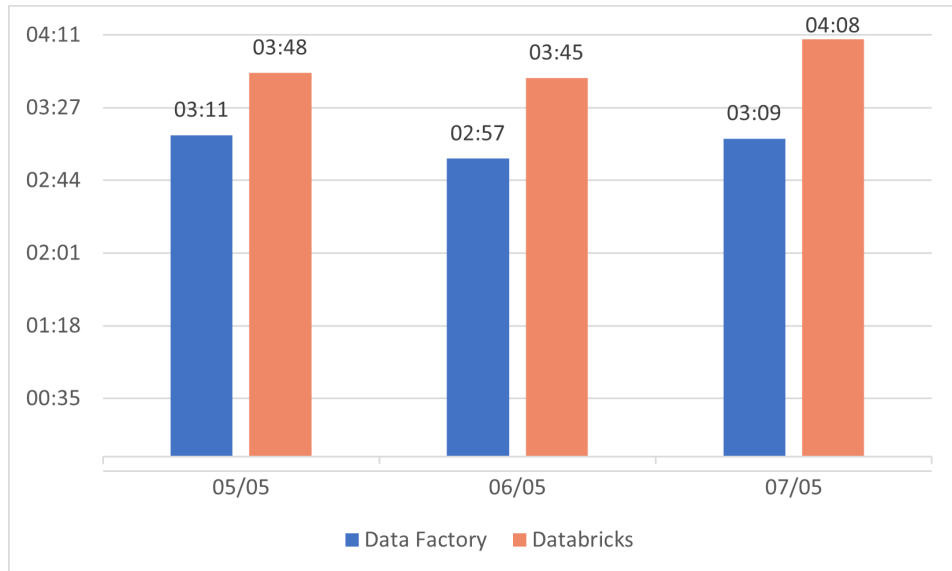
In deze grafiek wordt per datum de uitvoeringstijd getoond voor de pipeline met 4 Worker Cores + 4 Driver Cores. De extra tijd die er bij komt is de cluster startup tijd die resulteert in een totale tijd. In de grafiek is het moeilijk te zien of Azure Data Factory of Azure Databricks de beste performantie heeft.



Figuur (6.12)

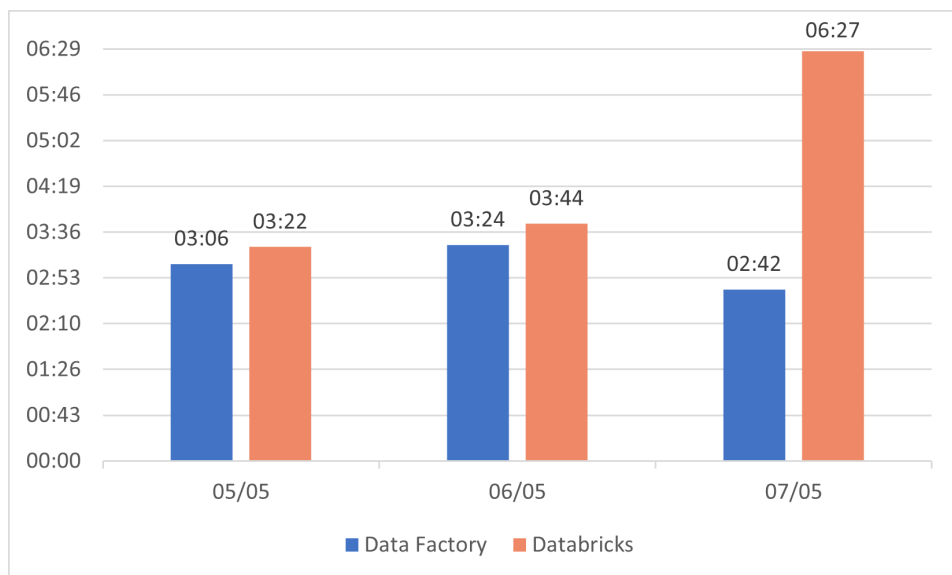
Uitvoeringstijden voor pipeline met 8 Worker Cores + 8 Driver Cores

Ook hier wordt de uitvoeringstijd per datum opnieuw getoond, maar deze keer voor de pipeline met 8 Worker Cores + 8 Driver Cores. Wanneer we kijken naar de uitvoeringstijd zien we dat Azure Databricks performanter is. Ook bij de totale tijd kunnen we dit zien, met uitzondering dat op 7 mei de cluster startup time hoger lag dan normaal.



Figuur (6.13)

Cluster startup tijden voor pipeline met 4 Worker Cores + 4 Driver Cores



Figuur (6.14)

Cluster startup tijden voor pipeline met 8 Worker Cores + 8 Driver Cores

Wat wel opvallend is, is dat voor beide pipelines de cluster startup tijd voor Azure Data Factory opvallend lager is.

7

Conclusie

In dit onderzoek werd onderzocht welke optie tussen Azure Data Factory en Azure Databricks het beste was voor het implementeren van een ETL.

7.1. Aan de hand van vergelijkingscriteria

Als we gaan kijken naar kostprijs en performantie zien we dat Azure Data Factory consistentere resultaten heeft gepaard met iets hogere kosten. Daarnaast zien we dat Databricks performanter is dan Data Factory wanneer een hoger aantal Cores gebruikt wordt. Wanneer we kijken naar de cluster startup tijden zien we dat deze bij Databricks hoger liggen dan bij Data Factory. Wanneer er dus gewerkt wordt met een pipeline die vaker uitgevoerd moet worden, waardoor het cluster ingeschakeld kan blijven, zal dit dus resulteren in snellere uitvoeringstijden dan bij Azure Data Factory. Op vlak van kostprijs en performantie is het dus moeilijk om te zeggen welke optie de beste is. Afhankelijk van use case zal dit enorm hard verschillen.

Debuggen in Azure Databricks is beter dan bij Azure Data Factory. De tabel die getoond wordt is makkelijker te gebruiken en toont de data gestructureerd weer. In Data Factory werd de volledige kolom naam vaak niet volledig getoond en was het moeilijker om de kolom breder te maken. Daarnaast kon deze tabel ook niet gefilterd worden of konden hier geen visualisaties voor gemaakt worden. In Databricks was dit wel het geval waardoor het implementeren van de pipeline hier makkelijker werd.

Zowel Azure Data Factory als Azure Databricks hebben beide de mogelijkheid om source control te gebruiken. Bij Azure Data Factory kan ook de infrastructuur opgeslaan worden in source control. Dit gebeurt in de vorm van Azure Resource Ma-

nager (ARM) templates. Bij Databricks gaat dit iets moeilijker, de opzet van een Databricks omgeving kan in ARM templates aangemaakt worden maar specifieke dingen zoals bijvoorbeeld clusters kunnen hier niet mee gecreëerd worden. Hiervoor zal er dus gebruik gemaakt moeten worden van Terraform of Databricks Asset Bundles (DABs). Het zal dus iets moeilijker zijn om Infrastructure as Code (IaC) te gebruiken in Databricks waardoor op dit vlak Data Factory aan te raden valt.

7.2. Eigen ervaring

Persoonlijk gaat mijn voorkeur naar Azure Databricks. Azure Data Factory was makkelijker om te gaan gebruiken maar doordat de uitgewerkte pipeline zeer complex was werd dit vaak minder overzichtelijk dan in Azure Databricks. Door mijn eigen kennis en ervaring in SQL vond ik bij het implementeren van de pipeline in Azure Databricks fouten in Azure Data Factory. Persoonlijk lijkt mij Databricks dus voor een developer een betere keus dan Data Factory. Voor iemand met minder programmeerervaring zou ik voor Azure Data Factory gaan hanteren.



Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.1. Inleiding

Het implementeren van ETL's en ELT's speelt een kritieke rol in de development binnen Net IT. Net IT is een bedrijf gespecialiseerd in Customer Relationship Management (CRM). Een CRM-systeem is een applicatie om voeling te houden met klanten en om processen te stroomlijnen en meer winst te genereren. Doordat Net IT een Microsoft Partner is zullen ze CRM-toepassingen implementeren met Microsoft Dynamics 365 en intelligente bedrijfstoeepassingen met Microsoft Power Platform. Binnen Microsoft 365 Dynamics Customer Engagement worden er CSV bestanden geëxporteerd naar Azure Data Lake. Deze data wordt minstens dagelijks opgekuist, aangevuld en opgesplitst om naar de klant te kunnen doorsturen via SFTP of e-mail. Momenteel wordt dit gedaan door ETL's en ELT's te implementeren in Azure Data Factory, gebruik makend van de UI tools die aangeboden worden. Doordat dit niet de enige mogelijkheid is om dit te implementeren zal er gekeken worden naar de verschillende opties die er zijn. Aangezien Net IT een Microsoft Partner is zal er vooral gekeken worden binnen Microsoft Azure. Deze verschillende mogelijkheden zullen vergeleken worden op basis van performantie, kostprijs (voor dezelfde performantie), complexiteit, moeilijkheidsgraad in opzet en configuratie van de resources, verschil in implementatietijd en mogelijkheden van de tool. De methode die gebruikt zal worden is een gemengde aanpak gebaseerd op literatuuronderzoek en het opstellen van proof-of-concepts voor de gegeven use case van Net IT. Dit zal resulteren in een gedetailleerd vergelijkingsrapport, inclusief aanbeveling voor de meest geschikte aanpak voor het implementeren van

ETL's of ELT's voor de gegeven use case.

A.2. Literatuurstudie

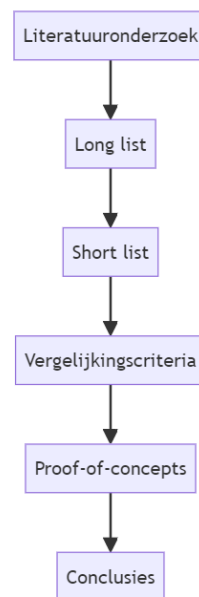
Als data engineer krijgt men data in veel verschillende vormen. Het is dus noodzakelijk om deze data klaar te maken voor business analytics. (Kromer, [2022b](#)) Hiervoor maakt men gebruik van ETL's en ELT's. Dit zijn processen die organisaties gebruiken voor het verzamelen en samenvoegen van data uit meerdere bronnen. Bij ETL's wordt de data getransformeerd voor het naar de doelopslagplaats geladen wordt, terwijl dit bij ELT's pas achteraf gebeurt. Daardoor staat ETL voor Extract, Transform and Load en ELT voor Extract, Load and Transform. (Bartley, [2023](#))

In Azure zijn er meerdere mogelijkheden voor het implementeren van ETL's en ELT's. Één van deze mogelijkheden is Azure Data Factory. Zoals te zien in de enquête van Sreemathy e.a. ([2021](#)) is dit de meest populaire data integratie service die aangeboden wordt door de cloud providers. Binnen Azure Data Factory kan er gebruik gemaakt worden van Mapping Data Flows, dit is een codevrije manier waarmee ETL's opgebouwd kunnen worden. De logica achter de ETL kan hierna makkelijk getest worden op live data en samples. (Kromer, [2022a](#))

Daarnaast biedt Azure ook Azure Databricks aan. Het verschil hierbij is dat de ETL's worden geïmplementeerd via code terwijl dat bij Azure Data Factory via de UI tools kan gebeuren. Azure Databricks is gebaseerd op het Apache Spark open source project. Het grote voordeel is dat het platform het toelaat om makkelijker te kunnen samenwerken. Daarnaast is Apache Spark niet enkel gelimiteerd tot het maken van ETL's maar kan het ook gebruikt worden voor real-time analytics, machine learning, graph processing, etc. (Etaati, [2019](#))

Azure is niet de enigste cloud provider die ETL tools aanbiedt. Zo heeft AWS bijvoorbeeld AWS Glue (Khan e.a., [2024](#)) en Google Cloud heeft Google Data Fusion. (Jaiswal, [2022](#))

A.3. Methodologie



De eerste fase is het literatuuronderzoek. Hierbij wordt er gekeken naar welke mogelijkheden er zijn voor het implementeren van ETL's en ELT's. Er zal vooral gefocust worden op de mogelijkheden binnen Azure maar ook andere opties zullen bekeken worden. Dit zal gedaan worden met behulp van academische onderzoekstools zoals Google Scholar en andere relevante databanken en bronnen. Ook de documentatie van Azure, casestudies van bedrijven die gebruik maken van Azure en blogs van experts op dit vakgebied zullen hier zeker bij van pas komen. Deze fase zal ook zeker in samenwerkingen met Net IT gebeuren zodat de huidige gebruikte technologieën voor het implementeren van ETL's en ELT's zeker niet uitgesloten worden. Dit onderdeel zal naar verwachting vier weken in beslag nemen.

In de tweede fase zullen de resultaten van het literatuuronderzoek samengevat worden in een long list. Dit onderdeel zal naar verwachting twee weken duren.

In de derde fase zullen de meest interessante opties uit de long list samengevat worden in een short list. Hierbij zal gekeken worden naar wat het interessantst is voor Net IT. Doordat dit een kleine fase is zal dit bij de tijd van de tweede fase horen. In de vierde fase zullen er vergelijkingscriteria opgesteld worden voor de gegeven use case door Net IT. Het is belangrijk om te weten wat er moet vergeleken worden. Belangrijk om op te merken is dat niet alles even meetbaar zal zijn. Er zal dus goed gekeken worden naar hoe de vergelijkingscriteria gemeten kunnen worden. Dit onderdeel zal naar verwachting één week duren.

In de vijfde fase zal er op basis van de short list en vergelijkingscriteria een proof-of-concept uitgewerkt worden voor elke mogelijkheid dat er binnen Azure is. Er zal een situatie (gegeven door Net IT) opgezet worden met dummy data in een data lake. Het doel is dat er op basis van deze data export bestanden zullen gemaakt worden. Dit zal de langste fase zijn en zal dus vier weken in beslag nemen.

De zesde en laatste fase, die naar verwachting één week zal duren, is de evaluatie van de opties die we hebben onderzocht. Het doel is om te tonen welke optie het beste is. Daardoor zal het resultaat van deze analyse een gedetailleerd vergelijkingsrapport zijn en een aanbeveling voor welke optie het beste is.

A.4. Verwacht resultaat, conclusie

Het resultaat zal een gedetailleerd vergelijkingsrapport zijn, inclusief aanbevelingen voor de meest geschikte aanpak voor het implementeren van ETL's of ELT's voor de gegeven use case. Daarnaast zal er ook een resultaat per vergelijkingscriteria zijn zodat Net IT zelf ook een dieper inzicht krijgt in bijvoorbeeld operationele implicaties, kostenstructuur, etc.

Bibliografie

- Anaparthi, S. (2023, november 14). *Azure App Registration: Azure Cloud Made Easy* (S. Anaparthi, Red.). <https://medium.com/@srijaanaparthi/azure-app-registration-azure-cloud-made-easy-ba44a6ea8953#:~:text=Azure%20App%20Registration%20is%20the,services%20more%20secure%20and%20accessible>.
- Awati, R. (2023a, november 1). *Microsoft Azure Data Lake* (R. Awati, Red.). <https://www.techtarget.com/searchcloudcomputing/definition/Microsoft-Azure-Data-Lake>
- Awati, R. (2023b, november 1). *Microsoft Azure Data Lake* (R. Awati, Red.). <https://www.techtarget.com/searchcloudcomputing/definition/Microsoft-Azure-Data-Lake>
- Bartley, K. (2023, januari 2). *What is the Difference Between ETL and ELT?* <https://rivery.io/blog/etl-vs-elt/>
- Etaati, L. (2019, juni 13). *Azure Databricks*. In *Machine Learning with Microsoft Technologies: Selecting the Right Architecture and Tools for Your Project* (pp. 159–171). Apress. https://doi.org/10.1007/978-1-4842-3658-1_10
- Ethan. (2024, januari 14). *100+ Best ETL Tools List Software (January 2024 Update)*. <https://portable.io/learn/best-etl-tools>
- Hill, J. (2023, augustus 23). *Notebooks in Azure Databricks* (J. Hill, Red.). <https://endjin.com/blog/2023/08/notebooks-in-azure-databricks>
- Holden Karau, P. W., Andy Konwinski, & Zaharia, M. (2015, januari 26). *Learning Spark*. https://books.google.be/books?id=2eptBgAAQBAJ&printsec=frontcover&redir_esc=y#v=onepage&q&f=false
- Inmon, B. (2023, maart 31). *Understanding the Necessity of ETL in Data Integration*. <https://www.integrate.io/blog/etl-in-data-integration/>
- J., P. K. (2023, maart 6). *What is Business Analytics? Definition, Importance Examples*. <https://www.linkedin.com/pulse/what-business-analytics-definition-importance-examples-jha/>
- Jackson, L. (2020, september 27). *The CI/CD Pipeline*. In *The Complete ASP.NET Core 3 API Tutorial: Hands-On Building, Testing, and Deploying*. Apress. https://doi.org/10.1007/978-1-4842-6255-9_12
- Jaiswal, N. (2022). *Data Fusion Basics*. <https://medium.com/google-cloud/data-fusion-basic-concepts-c40b09efd695>
- Karau, H., & Warren, R. (2017, mei 22). *High Performance Spark*. <https://books.google.be/books?hl=nl&lr=&id=90glDwAAQBAJ&oi=fnd&pg=PP1&dq=apache+>

spark&ots=FB4MS_Wiyb&sig=PYBA6rEGqwRMvkKogOdVgkpCqHA#v=onepage&q=apache%20spark&f=false

Khan, B., Jan, S., Khan, W., & Chughtai, M. I. (2024). An Overview of ETL Techniques, Tools, Processes and Evaluations in Data Warehousing. https://cdn.techscience.cn/files/jbd/2024/TSP_JBD-6/TSP_JBD_46223/TSP_JBD_46223.pdf

Kromer, M. (2022a). Common ETL Pipeline Practices in ADF with Mapping Data Flows. In *Mapping Data Flows in Azure Data Factory: Building Scalable ETL Projects in the Microsoft Cloud*. https://doi.org/10.1007/978-1-4842-8612-8_5

Kromer, M. (2022b, augustus 26). Introduction to Mapping Data Flows. In *Mapping Data Flows in Azure Data Factory: Building Scalable ETL Projects in the Microsoft Cloud* (pp. 27–50). Apress. https://doi.org/10.1007/978-1-4842-8612-8_3

Microsoft. (2023, juli 8). *What is Microsoft Cost Management* (Microsoft, Red.). <https://learn.microsoft.com/en-us/azure/cost-management-billing/costs/overview-cost-management>

Microsoft. (2024a, januari 30). *Azure Key Vault basic concepts* (Microsoft, Red.). <https://learn.microsoft.com/en-us/azure/key-vault/general/basic-concepts>

Microsoft. (2024b, maart 1). *What is Delta Live Tables?* (Microsoft, Red.). <https://learn.microsoft.com/en-us/azure/databricks/delta-live-tables/>

Microsoft. (2024c, maart 7). *What is Azure Databricks?* <https://learn.microsoft.com/en-gb/azure/databricks/introduction/>

Microsoft. (2024d, maart 7). *What is Delta Lake?* (Microsoft, Red.). <https://learn.microsoft.com/en-us/azure/databricks/delta/>

Microsoft. (2024e, maart 12). *What is Azure role-based access control (Azure RBAC)?* (Microsoft, Red.). <https://learn.microsoft.com/en-us/azure/role-based-access-control/overview>

Microsoft. (2024f, maart 19). *Manage Azure resource groups by using the Azure portal* (Microsoft, Red.). <https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/manage-resource-groups-portal>

Microsoft. (2024g, maart 19). *What is Azure Data Factory?* (Microsoft, Red.). <https://learn.microsoft.com/en-us/azure/data-factory/introduction>

Microsoft. (2024h, april 12). *Introduction to Azure Databricks Workflows* (Microsoft, Red.). <https://learn.microsoft.com/en-us/azure/databricks/workflows/>

Microsoft. (2024i, april 17). *What is Azure Policy?* (Microsoft, Red.). <https://learn.microsoft.com/nl-nl/azure/governance/policy/overview>

Pollefliet, L. (2011). *Schrijven van verslag tot eindwerk: do's en don'ts*. Academia Press.

Rawat, S., & Narain, A. (2018, december 19). Introduction to Azure Data Factory. In *Understanding Azure Data Factory: Operationalizing Big Data and Advan-*

- ced Analytics Solutions (pp. 13–56). Apress. https://doi.org/10.1007/978-1-4842-4122-6_2
- Samuel, N. (2021, november 8). *Databricks Clusters: Types 2 Easy Steps to Create Manage* (N. Samuel, Red.). <https://hevodata.com/learn/databricks-clusters/>
- Siddiqui, Z. (2023, maart 1). *Azure Security: Protecting Your Data in the Cloud* (Z. Siddiqui, Red.). <https://www.linkedin.com/pulse/azure-security-protecting-your-data-cloud-zeeshan-siddiqui>
- Sreemathy, J., Brindha, R., Nagalakshmi, M. S., Suvekha, N., Ragul, N. K., & Praveen-nandha, M. (2021). Overview of etl tools and talend-data integration. *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*. <https://intapi.sciendo.com/pdf/10.2478/picbe-2023-0182>
- Stoenescu, R. B. (2021, februari 25). *Azure IaaS, PaaS, SaaS – what's the difference?* (R. B. Stoenescu, Red.). <https://www.linkedin.com/pulse/azure-iaas-paas-saas-whats-difference-roxana-beatrice-stoenescu->
- Suneetha. (2024, augustus 24). *Microsoft Azure – SaaS, PaaS or IaaS* (Suneetha, Red.). <https://www.winwire.com/blog/microsoft-azure-saas-paas-or-iaas/>
- Vines, A., & Tanasescu, L. (2023). An Overview of ETL Cloud Services: An Empirical Study Based on User's Experience. *Proceedings of the International Conference on Business Excellence*, 17(1), 2085–2098. <https://intapi.sciendo.com/pdf/10.2478/picbe-2023-0182>