Analytics in Azure – week 2

In de tweede week van het leertraject Analytics in Azure gaan we dieper in op het verwerken van batch-data in de cloud. We hebben het gehad over onderwerpen als:

* Delta Lake architecturen
* Orchestratie
* ETL, ELT en moderne data-laadvarianten

Tooling-wise maken we kennis met diverse tools in het Azure-platform die kunnen helpen bij de batch-verwerking van data:

* Azure Data Factory (ADF) voor orchestratie
* Azure Synapse voor een “totaaloplossing” met:
  + Pipelines als geïntegreerde ADF
  + Serverless SQL Pools als SQL-queries die je direct op je Data Lake kunt uitvoeren
  + SQL Pools als een horizontaal schaalbaar SQL Data Warehouse
  + Spark Pools als geïntegreerde Big Data-oplossing voor alles waar Spark voor gebruikt kan worden
    - Interactieve queries (Spark SQL)
    - Machine Learning (Spark MLib)
    - Graph computation (Spark GraphX)
    - .. en de Python- en Scala- interfaces
* Azure Databricks als “Managed Spark” met heel veel handige verbeteringen en integraties
* Azure SQL Database wanneer je een klein of voordelig Data Warehouse in de cloud nodig hebt (hier hebben we het vorige week over gehad)

In de tijd tot volgende week is het belangrijkste om je Proof of Concept verder uit te bouwen met inhoud. Probeer een enkele stroom op te zetten die zich van de bron tot aan je “warehouse” laag uitbreidt.

Ook is er deze week weer veel verbreding en verdieping. Bedenk goed: dit zijn geen verplichte voorbereidingen! Het belangrijkste is dat je je PoC uitbreidt met de nieuwe kennis.

# Voorbereiding week 3

In week 3 gaan we met Azure DevOps aan de slag. Om hiermee goed aan de slag te kunnen is het belangrijk dat je je voorbereidt door **zo vroeg mogelijk** een DevOps *organization* aan te maken met een persoonlijk account (d.w.z. dat je alle rechten hebt en je zelf de organization hebt aangemaakt).

Microsoft heeft namelijk recent een beperking doorgevoerd op Azure DevOps Pipelines, die je kunt laten opheffen door het invullen van een formulier

Daarom is het belangrijk vooraf (eerder = beter) twee stappen uit te voeren:

1. Een nieuwe Azure DevOps organization aanmaken met een *persoonlijk* account

Dit kan eenvoudig via [Azure DevOps Services | Microsoft Azure](https://azure.microsoft.com/en-us/services/devops/?nav=min). Open de pagina in een private window, en kies voor “Start free”. Volg de stappen om een Azure DevOps-organization aan te maken. Het zijn slechts drie stappen, en deze omgeving ga je gedurende de training gebruiken om op te werken.

1. Een formulier invullen bij Microsoft voor het aanvragen van CI/CD “parallellism”

Nadat je *organization* is aangemaakt, vul je het formulier in op [Azure DevOps Parallelism Request (office.com)](https://forms.office.com/pages/responsepage.aspx?id=v4j5cvGGr0GRqy180BHbR63mUWPlq7NEsFZhkyH8jChUMlM3QzdDMFZOMkVBWU5BWFM3SDI2QlRBSC4u). Voor de vierde en laatste vraag: het betreft een *private* projects-aanvraag.

Wanneer je deze twee stappen hebt genomen, ben je qua omgevingen goed voorbereid op de training.

# Oefeningen

In deze week kun je de volgende oefeningen doen om meer ervaring op te doen. Aan te raden is om één kant te kiezen en die uit te diepen. Houd daarbij rekening met je beschikbare tijd, startniveau en wat voor jouw organisatie / klant / PoC de meest interessante techniek is.

In alle gevallen geldt: zorg ervoor dat je een stuk ETL ontwikkelt, waarbij data getransformeerd en/of geïntegreerd wordt. Waar het kan: neem een eenvoudige dataset uit je organisatie als voorbeeld. Als dat niet lukt: neem bijvoorbeeld de AdventureWorks-database als basis, en zorg dat deze in twee stappen verder ingeladen wordt:

1. Ingest (laden “as-is” naar Data Lake, bijvoorbeeld in Parquet-formaat)
2. Process (verwerken van data in je Data Lake naar een andere “laag”)

Over de aanpak:

* De “ingest” is het eenvoudigst met Azure Data Factory te doen – het *copy* statement is een heel krachtige manier
* Wanneer je voor je PoC data van on-prem wilt laden (dit kan met een self-hosted integration runtime in ADF), voeg dit dan later toe: zorg dat eerst de structuur staat
* De “Process” kun je op diverse manieren aanpakken:
  + Grafisch: Met ADF (of Synapse Pipelines) een Data Flow inrichten. Niet vergeten uit te zetten!
  + In DataBricks: gebruik DataBricks SQL of DataBricks Python-codes om de data te laden en transformeren
  + In Synapse Serverless SQL: SQL-code schrijven om data te transformeren en terug te schrijven naar het Data Lake
  + In Synapse SQL Pools: SQL-code schrijven om data te transformeren en te schrijven naar een SQL-cluster
* Wat betreft de locatie: Zowel Databricks als ADF kunnen de data zowel naar het Data Lake (tip: delta formaat) als naar Synapse SQL Pools efficiënt wegschrijven. Voor kleinere workloads is Azure SQL Database hier ook erg geschikt.

Hieronder vind je per tool nog wat extra materiaal voor naslag of verder leren:

## Azure Data Factory

Voor Azure Data Factory hebben we twee tutorials waarmee je meer kennis maakt ADF:

[Create a data factory pipeline](https://docs.microsoft.com/en-us/azure/data-factory/tutorial-copy-data-portal) helpt je om de basics nog eens te doorlopen: wat is er allemaal nodig om ADF simpelweg te gebruiken om data te verplaatsen? Prettig is dat hier alle diensten nog eens de revue passeren.

[Delta lake ETL with data flows](https://docs.microsoft.com/en-us/azure/data-factory/tutorial-data-flow-delta-lake) gaat een stapje verder, door het gebruiken van het Delta Lake formaat.

Voor alle genoemde formaten kun je ten slotte kijken hoe Power BI hiermee verbindt:

* Kun je Parquet-bestanden inladen in Power BI?
* Kun je Delta Lake bestanden inladen in Power BI?

Als je wat meer “los” bent op Power Query kun je ook eens nadenken over / kijken naar de impact van deze formaten: is *query folding* bijvoorbeeld nog steeds mogelijk?

## Databricks

In de video [Getting Started with Databricks SQL - YouTube](https://www.youtube.com/watch?v=OMjxlqIqSqs) laat Simon Whiteley zien hoe je Data Warehousing in Databricks kunt “regelen”. In de [Tutorial - Perform ETL operations using Azure Databricks | Microsoft Docs](https://docs.microsoft.com/en-us/azure/databricks/scenarios/databricks-extract-load-sql-data-warehouse) wordt dit verder uitgewerkt op een praktijkgerichte basis.

Ook kun je het volgende Databricks *notebook* importeren:

* Klik binnen Databricks op Workspace -> Users -> jouw username
* Kies hier het pijltje dat naar beneden wijst naast je naam, kies *import*
* Geef aan dat je het Notebook wilt downloaden van het volgende adres:  
  <https://github.com/MicrosoftDocs/mslearn-perform-basic-data-transformation-in-azure-databricks/blob/master/DBC/05.1-Basic-ETL.dbc?raw=true>

Zin om nóg een tandje dieper te gaan? Er is ook een “advanced transformations” notebook beschikbaar dat je kunt importeren:

<https://github.com/MicrosoftDocs/mslearn-perform-advanced-data-transformation-in-azure-databricks/blob/master/DBC/05.2-Advanced-ETL.dbc?raw=true>

## Synapse Serverless SQL

Microsoft zet Synapse Serverless SQL momenteel vooral in de markt voor “Exploratory Data Analysis”. Het is echter ook mogelijk om logische Data Warehouses te maken met behulp van Synapse.

Srini Velamakantie laat in zijn blog [Data Lakehouse Architecture — Azure Synapse Serverless SQL Pools | by Srini Velamakanti | Analytics Vidhya | Medium](https://medium.com/analytics-vidhya/data-lakehouse-architecture-azure-synapse-serverless-sql-pools-78a7a142ca37) zien hoe dat er qua ontwerp uitziet.

Wanneer je voor jouw PoC hier een stuk van wilt gebruiken, dan heeft hij ook een tweede artikel geschreven met daarin de technische “tussenstappen” uitgeschreven: [Build a new-age Cost-Effective Business Intelligence System using Azure Data Lakehouse and Synapse Serverless SQL Pool | by Srini Velamakanti | Analytics Vidhya | Medium](https://medium.com/analytics-vidhya/build-a-new-age-cost-effective-business-intelligence-system-using-azure-data-lakehouse-and-synapse-46d361ba33e6)

## Synapse Spark

Synapse Spark biedt met name voordelen wanneer je Machine Learning-modellen wilt integreren in je dataverwerking. Het operationaliseren van ML modellen is in diverse (grotere) organisaties een belangrijke rol voor Data Engineers. Wanneer dit in jouw organisatie niet het geval is, is een PoC in een andere techniek waarschijnlijk nuttiger.

Als je hiermee aan de slag wilt, heeft Microsoft een tutorial die zich specifiek richt op *data wrangling* met Spark: [Data wrangling with Apache Spark pools (preview) - Azure Machine Learning | Microsoft Docs](https://docs.microsoft.com/en-us/azure/machine-learning/how-to-data-prep-synapse-spark-pool). Deze tutorial staat sterk in de context van Machine Learning Operations (MLOps).

## Synapse SQL Pools

Voor grote DW-oplossingen is SQL Pools de aangewezen Microsoft-tool binnen het Azure-platform. Je kunt de data op diverse manieren in je SQL Pool krijgen. In drie artikelen wordt hier meer over uitgelegd:

* [Ingest data into a dedicated SQL pool - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/data-integration/data-integration-sql-pool) legt uit hoe je data de SQL Pool in krijgt met behulp van Synapse Pipelines
* [Quickstart: Bulk load data using a single T-SQL statement - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/quickstart-bulk-load-copy-tsql) geeft je een quick start om met behulp van het COPY statement (Synapse-specifiek) data eenvoudig in te laden
* [Quickstart: Bulk load data with a dedicated SQL pool - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/quickstart-load-studio-sql-pool) geeft je een tip over de wizard die je kunt gebruiken om het COPY-statement voor je te laten genereren

Wat betreft de snelste manier om data te laden is er momenteel wat onduidelijkheid. In het verleden waren er “best practices” over het laden met COPY of CREATE TABLE AS SELECT – manieren die logischerwijs aansloten bij een parallelle manier van data verwerken. Hier wordt in de huidige documentatie nauwelijks meer over gesproken. En het zou goed te verklaren zijn als Microsoft deze logica over het slim aanpakken van dataloads inmiddels verwerkt heeft in een “regulier” INSERT..SELECT statement.

Wanneer je hier op schaal mee wilt gaan werken, is het dus zaak om dit te testen. Gelukkig kan dat vrij eenvoudig in de cloud.

# Verdieping en verbreding

Hieronder vind je diverse onderwerpen als je meer verbreding of verdieping zoekt.

## Best Practices voor Data Lakes

Door de gedistribueerde aanpak van je Data Lake loont het om ook je bestanden slim te distribueren. Per bestandsformaat gelden daar soms verschillende *practices*. Microsoft heeft enkele algemene practices in het artikel [Best practices for using Azure Data Lake Storage Gen2 | Microsoft Docs](https://docs.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-best-practices) gezet. Eén advies is bijvoorbeeld om bestanden tussen de 256 MB en 100 GB te houden.

Wanneer je Delta Lake als bestandsformaat gebruikt, gebeurt een deel van die optimalisatie voor jou. Delta Lake werkt op de achtergrond namelijk met Parquet-bestanden, en de Delta Engine (die dus Delta bestanden wegschrijft) bepaalt hoe de data verdeeld moet worden. Jij helpt daarbij door een partitioneringskolom aan te geven. Voor de liefhebbers hier opnieuw een [paper](https://databricks.com/wp-content/uploads/2020/08/p975-armbrust.pdf) van, voor de techneuten een [onder-de-motorkap van het protocol](https://github.com/delta-io/delta/blob/master/PROTOCOL.md).

## ADF

Voor een stukje verdieping en verbreding zou je in Azure Data Factory kunnen kijken naar twee onderwerpen:

1. De *wrangling Data Flow*
2. Data laden met behulp van een For-loop

### Wrangling Data Flow

Met de Wrangling Data Flow krijg je feitelijk Power Query beschikbaar op grote schaal (let wel: het is nog vrij beperkt in vergelijking met wat je van Power Query gewend bent!). In de onderstaande “Getting started” krijg je hier een eerste kennismaking mee.

[Getting started with wrangling data flow in Azure Data Factory - Azure Data Factory | Microsoft Docs](https://docs.microsoft.com/en-us/azure/data-factory/wrangling-tutorial)

Bedenk je goed dat in een “Getting started” vooral wordt laten zien wat er wél kan! Veel zaken waar Power Query normaal in uitblinkt (gekke datastructuren, kolommen naar rijen en andersom, etc.) zijn juist niet mogelijk. Experimenteer daarom veel met andere Power Query voorbeelden om te zien wat er wel / niet werkt hierin.

### Data laden met behulp van een For-loop

Wanneer je in de Data Factory-oefeningen twee tabellen hebt ingeladen met de *copy data*-activiteit, vraag je je al snel af hoe je dit ooit voor 100 tabellen schaalbaar gaat doen. Goed nieuws: je kunt eenvoudig een *loop* schrijven. Op de GitHub vind je het stappenplan hoe je hiermee aan de slag kunt: [analytics-in-azure/04-copy-data-stg-loop.md at main · vstrien/analytics-in-azure (github.com)](https://github.com/vstrien/analytics-in-azure/blob/main/week-2/opdrachten-en-thuiswerk/verdieping-adf/04-copy-data-stg-loop.md)

## Data Lakehouse op Synapse

Wanneer je je Data Lakehouse op Synapse wilt opzetten, krijg je met nieuwe uitdagingen te maken. Denk bijvoorbeeld aan de manier waarop een sterschema is opgezet: je bent wellicht gewend elke dimensie een surrogaatsleutel toe te kennen – en die vervolgens in je fact op te nemen. Dat gaat niet meer “zomaar” op een Lakehouse – er is geen centrale database die bijhoudt welke id’s er al wel en niet zijn uitgegeven.

Soms kun je uitgaan van de sleutels in de bron. In het artikel [Creating a Logical Data Warehouse with Synapse Serverless SQL: Part 4 of 4 – Connecting Power BI to the Dimensional Model – Serverless SQL](https://www.serverlesssql.com/logical-data-warehouse/creating-a-logical-data-warehouse-with-synapse-serverless-sql-part-4-of-4-connecting-power-bi-to-the-dimensional-model/) wordt er dieper op ingegaan hoe je dit kunt aanpakken.

Een alternatief is dan om te gaan voor hash-sleutels: je genereert een hash op basis van de onderliggende data. Dus met de data waarmee je eerder een lookup deed om je fact table te vullen met de juiste dimensie-id’s (SELECT id FROM dimension WHERE eigenschap = ‘a’ AND eigenschap2 = ‘b’) maak je nu simpelweg een hash (bijv. SELECT SHA1(a & ‘separator’ & b)). Bijkomend voordeel is dat je feittabel nu al gevuld kan worden vóórdat je dimensie bijgewerkt is.

Een laatste afweging is om een Data Lakehouse zoveel mogelijk zonder UPDATE statements te houden. Zoals vorige week benoemd, is een parallel gedistribueerd systeem vaak gebaat bij zo min mogelijk communicatie tussen de *workers*. In het artikel [Building A Modern Batch Data Warehouse Without UPDATEs | by Daniel Mateus Pires | Towards Data Science](https://towardsdatascience.com/building-a-modern-batch-data-warehouse-without-updates-7819bfa3c1ee) wordt hier dieper op ingegaan.

## Wat doet DataBricks eigenlijk onder water?

Zoals we op de trainingsdag zagen heeft DataBricks ook een SQL-interface. Maar wanneer je hier een tabel aanmaakt, dan blijk je die ook te kunnen benaderen met je Python-code. Of met R. Hoe werkt dat onder water?

DataBricks slaat alles in een groot filesystem op. Ze noemen dit “DBFS” – DataBricks File System. En feitelijk maken ze hier ook een soort “external tables” in aan, waarin jij zelf kunt aangeven in welk formaat je welke data wilt aankoppelen.

Als je wilt inzien wat er onder water gebeurt, is het handig om de “DBFS browser” aan te zetten. Hier staat beschreven hoe dat moet: [Manage the DBFS file browser | Databricks on AWS](https://docs.databricks.com/administration-guide/workspace/dbfs-browser.html)

Vervolgens kun je bekijken hoe dit filesystem eruit ziet: [Browse DBFS using the UI](https://docs.microsoft.com/en-us/azure/databricks/data/databricks-file-system#browse-dbfs-using-the-ui).

## Meer over DataBricks, Delta Lake en Synapse

Simon Whiteley heeft twee interessante video’s die je nog wat meer achtergrond geven. Het zijn opnames van sessies op SQLBits:

* [Databricks, Delta Lake and You - Simon Whiteley - YouTube](https://www.youtube.com/watch?v=y91r_DLMEq8) (48 min.)
* [The Azure Spark Showdown - Databricks VS Synapse Analytics - Simon Whiteley - YouTube](https://www.youtube.com/watch?v=FjsnVueXijQ) (49 min.)