Analytics in Azure – week 3

In de derde week van het leertraject Analytics in Azure gaan we aan de slag met ontwikkeltechniek: Git, Azure DevOps Repos en Azure DevOps Pipelines.

Volgende week gaan we aan de slag met het opzetten van je infrastructuur. Dan bouwen we voort op alle kennis en ervaring die we nu hebben opgedaan met bijvoorbeeld Azure DevOps, en versterken dat met een stuk Infra-as-Code.

In de tijd tot volgende week is het nu zaak om de *inhoud* van je Proof of Concept geautomatiseerd te kunnen uitrollen:

* Als je SQL Databases gebruikt: door een database-project op te zetten.
  + Wanneer je ervaring hebt met [Visual Studio](https://visualstudio.microsoft.com/vs/features/ssdt/), is dit het eenvoudigste. Je maakt een nieuw (leeg) database-project aan, en [importeert het bestaande schema](https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-data-tools/hh864423(v=vs.103)). Koen Verbeeck heeft hier een handig samenvattend artikel over geschreven: [SSDT Top Features for Database Projects (mssqltips.com)](https://www.mssqltips.com/sqlservertip/6749/ssdt-top-features-database-projects/)
  + Als je ervaring hebt met Visual Studio Code kun je hier ook een Database Project aanmaken, en [de structuur importeren van een bestaande database](https://docs.microsoft.com/en-us/sql/azure-data-studio/extensions/sql-database-project-extension-getting-started?view=sql-server-ver15).
* Azure Data Factory:
  + Je ontwikkel-ADF kun je koppelen aan een Git-repository aan de achterzijde.
  + Elke keer dat je op “publish” drukt, wordt hier een nieuwe ARM-template gemaakt
  + Gebruik hiervoor het patroon dat beschreven is op [Continuous integration and delivery - Azure Data Factory | Microsoft Docs](https://docs.microsoft.com/en-us/azure/data-factory/continuous-integration-delivery)
  + In project ‘04’ van je cursus-*organization* staat een voorbeeld van de uitgebreide workflow van ADF. Het *deployment*-deel ervan bevat echter enkele nuttige hints over wat nodig is voor de ADF-deployment (zoals het pad naar het ARM-bestand)

# Verdieping / verbreding

We hebben deze week gekeken naar Git, Azure DevOps Repos en Azure DevOps Pipelines. Op alle gebieden is nog veel verdieping en achtergrond mogelijk.

Ook hebben we kort gekeken naar de achtergronden van DevOps – ook hier is nog verdere verdieping en verbreding mogelijk – bijvoorbeeld hoe dit zich kan vertalen naar de data-wereld.

## DevOps en DataOps

Wanneer je je begrip van DevOps wilt vergroten en combineren met het lezen van een leuk boek, kun je The Phoenix Project van Gene Kim eens opzoeken. Er is ook een opvolger (The Unicorn Project), die wat moderner is en vanuit een ander perspectief beschreven.

Als je vooral benieuwd bent naar de vertaalslag naar het data-gebied, zijn de artikelen van DataKitchen over DataOps erg interessant. Aanrader: [DataOps is NOT Just DevOps for Data | by DataKitchen | data-ops | Medium](https://medium.com/data-ops/dataops-is-not-just-devops-for-data-6e03083157b7)

## Git

Wat we momenteel van Git hebben gezien, is het “remote” samenwerken. Wat we nog níet hebben gezien, is de manier waarop je branches kunt gebruiken en inzetten. Doordat branches in Git lichtgewicht en eenvoudig zijn, worden ze veelvuldig ingezet om een bepaalde workflow te handhaven.

* Atlassian heeft een handig artikel met bijbehorende video over wat branches eigenlijk zijn: [Git Branch | Atlassian Git Tutorial](https://www.atlassian.com/git/tutorials/using-branches)
* Ook voor de Git Workflows heeft Atlassian wat verdiepende en leesbare artikelen:
  + [Comparing Git Workflows | Atlassian Git Tutorial](https://www.atlassian.com/git/tutorials/comparing-workflows)
  + [Git Feature Branch Workflow | Atlassian Git Tutorial](https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow)

Ook Microsoft heeft haar ervaringen beschreven:

* [How Microsoft develops modern software with DevOps - Azure DevOps | Microsoft Docs](https://docs.microsoft.com/en-us/devops/develop/how-microsoft-develops-devops). Dit artikel vereist iets meer voorkennis over hoe branches werken, maar geeft je meer achtergrond over hoe Microsoft zelf zaken aanpakt
* [Git patterns and anti-patterns for successful developers - YouTube](https://www.youtube.com/watch?v=t_4lLR6F_yk) is een filmpje van 20 min. over do en don’t in het gebruik van Git en branches

## Azure DevOps Repos

Met de kennis over Git branching, is het mogelijk om in Azure DevOps Repos een branching-aanpak te handhaven. Er zitten namelijk tools in om de branches te “beschermen”:

### *Keep a high quality, up-to-date main branch*

*The code in your main branch should pass tests, build cleanly, and always be current. Your main branch needs these qualities so that feature branches created by your team start from a known good version of code.*

*Set up a [branch policy](https://docs.microsoft.com/en-us/azure/devops/repos/git/branch-policies?view=azure-devops) for your main branch that:*

* *Requires a pull request to merge code. This approach prevents direct pushes to the main branch and ensures discussion of proposed changes.*
* *Automatically adds reviewers when a pull request is created. The added team members review the code and comment on the changes in the pull request.*
* *Requires a successful build to complete a pull request. Code merged into the main branch should build cleanly.*

*(bron:* [*Git branching guidance - Azure Repos | Microsoft Docs*](https://docs.microsoft.com/en-us/azure/devops/repos/git/git-branching-guidance?view=azure-devops#keep-a-high-quality-up-to-date-main-branch)*)*

## Azure DevOps Pipelines

In Azure DevOps Pipelines zitten ook tools om je releases nog beter te structureren:

* Door een “echte” deployment aan te duiden, kun je *environment* instellingen doen. Bijvoorbeeld op welke omgeving (dev, test, prod) een release mag plaatsvinden onder welke voorwaarden. Meer informatie op [Deployment jobs - Azure Pipelines | Microsoft Docs](https://docs.microsoft.com/en-us/azure/devops/pipelines/process/deployment-jobs?view=azure-devops)
* Je kunt ADF nog een stap verder automatiseren met [Automated publishing for continuous integration and delivery - Azure Data Factory | Microsoft Docs](https://docs.microsoft.com/en-us/azure/data-factory/continuous-integration-delivery-improvements). Feitelijk is dit een eenvoudige manier om van de handmatige “publish”-stap af te komen.

## Synapse en Databricks

Een andere mogelijke verdieping is om na te denken over hoe Synapse en/of Databricks nu in versiebeheer onder te brengen zijn. Beiden hebben een “diepe” Git-integratie, maar een wat technische eenvoudige CI/CD aanpak:

### Databricks

* [Continuous integration and delivery on Azure Databricks using Azure DevOps - Azure Databricks | Microsoft Docs](https://docs.microsoft.com/en-us/azure/databricks/dev-tools/ci-cd/ci-cd-azure-devops)
* [How to Implement CI/CD on Databricks Using Databricks Notebooks and Azure DevOps - The Databricks Blog](https://databricks.com/blog/2021/09/20/part-1-implementing-ci-cd-on-databricks-using-databricks-notebooks-and-azure-devops.html)

### Synapse

* [Continuous integration & delivery in Azure Synapse Analytics - Azure Synapse Analytics | Microsoft Docs](https://docs.microsoft.com/en-us/azure/synapse-analytics/cicd/continuous-integration-delivery)
* Voor Synapse zit er in project 04 ook een voorbeeld-pipeline (momenteel niet aangesloten op jullie huidige code en Azure-omgeving, dus nog wel enig uitzoekwerk)