

Azure DevOps Build, Deploy, Pull Requests and Testing for your Data Warehouse

13. September 2023
SQL Konferenz Hanau
Felix Möller, noventum consulting GmbH



Über mich

Felix Möller

- Service Lead Microsoft Intelligent Data Platform bei noventum
- Erfahrungen seit gut 6 Jahren mit der Azure Data Platform
- Seit 11 Jahren Microsoft BI
- Konferenzsprecher
 - 2021: Understanding Costs of your Data Estate with Power BI
 - 2021: Chocolatey for the Data Platform Guy
 - 2019: Advanced Modelling – Power Platform World Tour, München ([Slides](#))
 - 2019: CI/CD for your Cloud Data Platform, München ([Slides](#))
 - 2018: TFS / VSTS for SSIS projects, St. Augustin ([Slides](#))
 - 2018: Power BI Dashboard in a Day, Köln
 - 2015: Einführung in BIML metadatengetriebene DWH-Entwicklung, Hannover ([Slides](#))
- Fokusthemen
 - Data Platform DevOps
 - Metadata-driven DWH-Entwicklung
 - SAP-Datenextraktion
- E-Mail: felix.moeller@noventum.de



data &
analytics
@noventum
consulting

AZURE DATAGUY

<https://azdataguy.com/>

Publikumsfragen



Agenda

- 1 Überblick und Zielsetzung
- 2 Zusammenarbeit mit Git
- 3 Build und Deploy
 - 3.1 Integration Services
 - 3.2 SQL Server Database Projects
 - 3.3 Tabular Models
 - 3.4 Azure Data Factory
- 4 Testing
- 5 Zusammenfassung
- 6 Fragen / Diskussion

Qualität erhöhen, effizient zusammenarbeiten

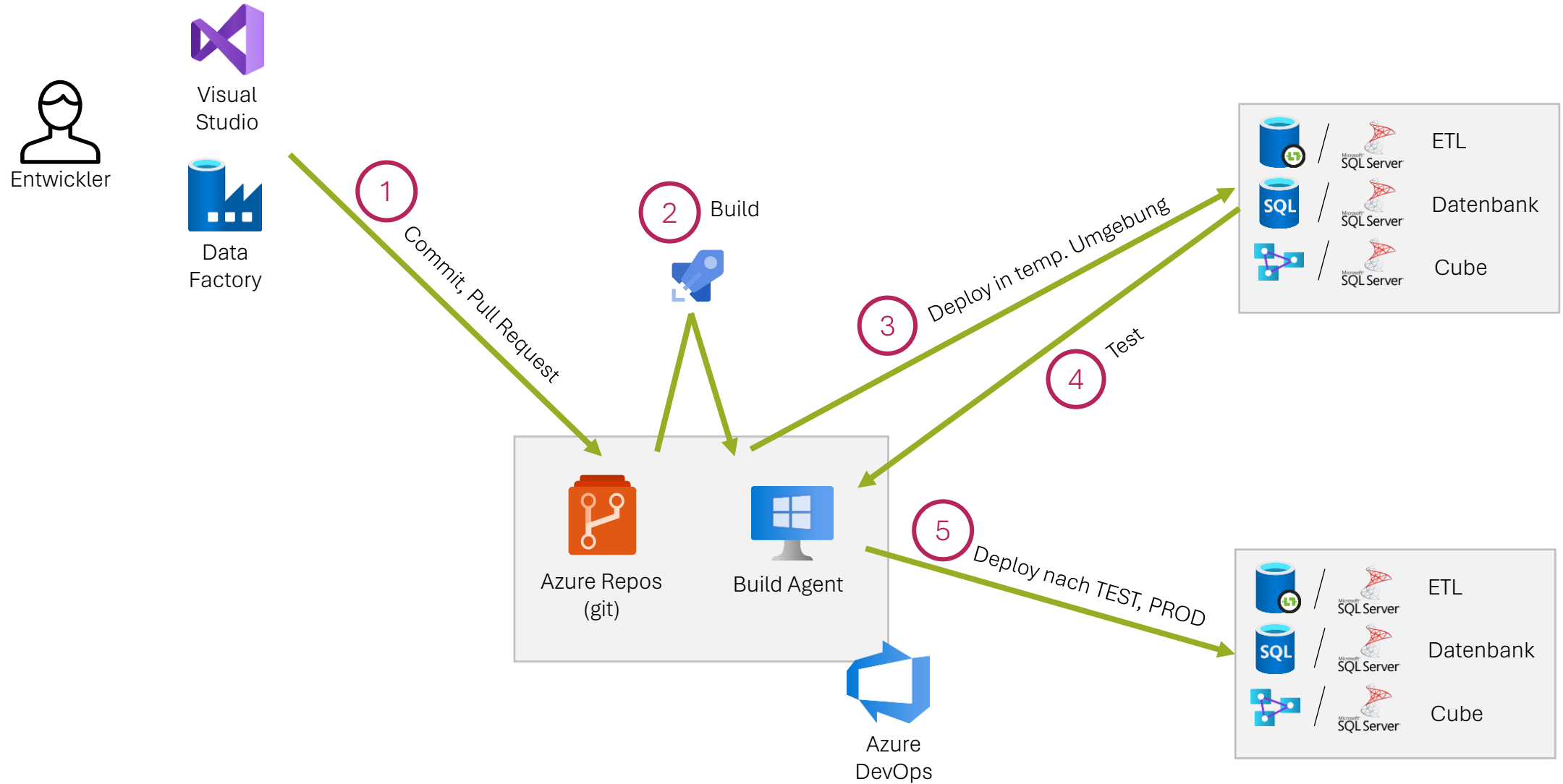
Zielsetzung

- Qualität erhöhen, schneller bereitstellen
 - Build – eigener Agent und mit MSFT-hosted Agent
 - Deployment – onPrem und Azure PaaS
 - Integrationstest
 - Datentests
- Vollständige Dokumentation
 - Infrastruktur
 - Build
 - Deployment
- Skalierung für Zusammenarbeit mehrerer Entwickler
 - Parallele Entwicklungen mit Pull Requests
- Vereinfachung des Deployments
 - Deployen in jede Umgebung ohne manuelle Schritte

The screenshot displays the Azure DevOps pipeline interface for a pipeline named 'Validate RepositoryVorlage (op01-onPrem)'. The pipeline is triggered by a pull request from Felix Möller. The interface shows the following details:

- Summary:** Pull request by Felix Möller. Repository: RepositoryVorlage (242, 605493ab). Time started and elapsed: Today at 13:57 (5m 17s). Related: 0 work items, 1 published. Tests and coverage: 100% passed, Setup code coverage.
- Stages:**
 - Build Stage:** 1 job completed (1m 34s). 100% tests passed, 1 artifact.
 - Build Job:** 1m 28s.
 - Deploy the database:** 3 jobs completed (1m 47s). Jobs include: SSOT Deployment to Ad... (1m ...), SSIS Deployment to Adhoc (16s), SSAS Deployment to Adhoc (11s).
 - Process data platform:** 1 job completed (42s). Job: Execute SSIS on Adhoc (31s).
 - Test data platform:** 1 job completed (28s). 100% tests passed. Jobs include: Execute Test Cases (22s).
 - Destroy elements of L...:** 3 jobs completed (40s). Jobs include: Drop Database (9s), Remove SSIS project from S... (6s), Remove SSAS Model (6s).

Die Schritte des DevOps-Prozesses



Agenda

1 Überblick und Zielsetzung

2 Zusammenarbeit mit Git

3 Build und Deploy

3.1 Integration Services

3.2 SQL Server Database Projects

3.3 Tabular Models

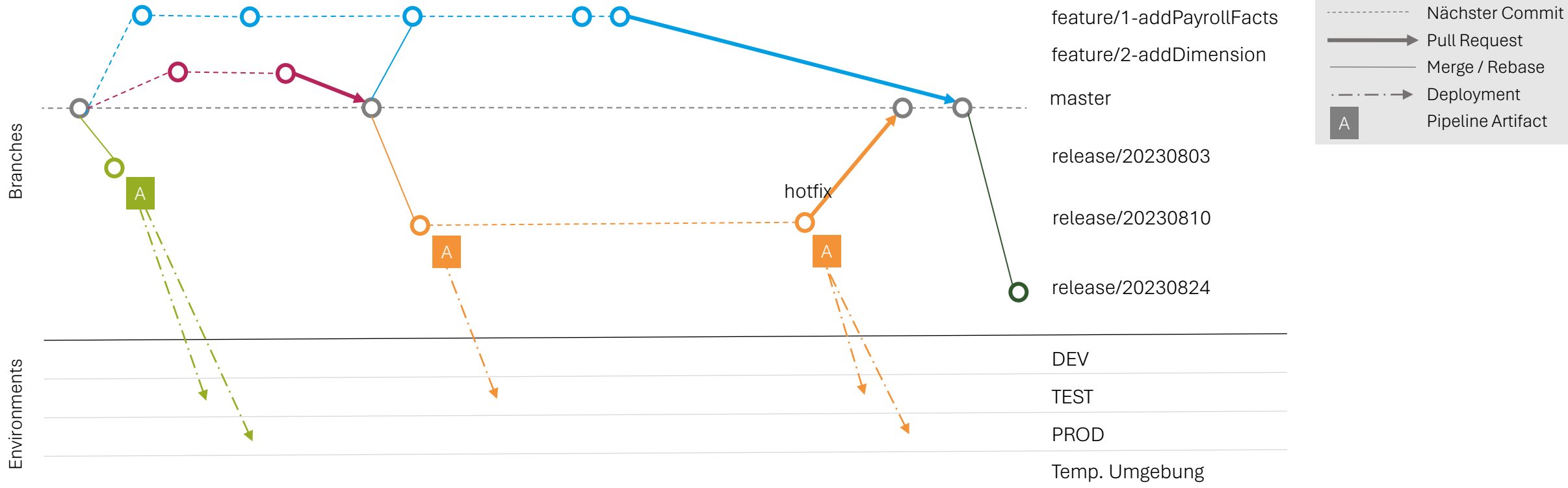
3.4 Azure Data Factory

4 Testing

5 Zusammenfassung

6 Fragen / Diskussion

Branching: basierend auf MS Release Flow



Beschreibung

- Es werden Releases erzeugt, aus diesen entstehen Artefakte, die in Umgebungen deployt werden
- Details unter [How Microsoft develops with DevOps - Azure DevOps | Microsoft Learn](#), [Release Flow: How We Do Branching on the VSTS Team - Azure DevOps Blog \(microsoft.com\)](#)

Bewertung

- Vorteil: Eine führende Branch (master), keine Drift zwischen Branches möglich
- Vorteil: Gleicher Code in alle Umgebungen released
- Vorteil: Entkoppelung von Deployment und Pull Request Merge

Pull Request validiert sich automatisch

The screenshot displays the Azure DevOps interface for a pull request titled "adjustment of publish profile". The "Checks" section shows three required checks: "Ensure Branch can be built" (succeeded), "Ensure it can be deployed to onPrem" (succeeded), and "Ensure it can be deployed to Azure" (queued). A "Checks" modal window is open, showing the details of the "Ensure it can be deployed to Azure" check, which is currently in a "Queue" state. A callout box points to the "Checks" section with the text "Pull Request führt automatisch Checks aus".

The "Deployment" section shows a pipeline named "90 Deployment" with several stages, including "platformproductenvironment", "platformproducts", "platforms", "scripts", and "stages". A callout box points to this section with the text "YAML Pipelines".

The "Pull request" details section shows the pull request by Felix Möller, with repository "RepositoryVorlage" and commit "248". The "Summary" section shows the pull request was created today at 21:11, with 0 work items and 1 published item. The "Stages" section shows a pipeline with five stages: "Build Stage" (1 job completed, 1m 45s), "Deploy the platform" (3 jobs completed, 2m 13s), "Process data platform" (1 job completed, 42s), "Test data platform" (1 job completed, 33s), and "Destroy elements of ..." (3 jobs completed, 41s). A callout box points to the "Build Stage" with the text "Ein Check testet das Deployment".

Ein Check testet das Deployment

YAML Pipelines

Live Demo



```
feature/sqlkonferenz-broken  into master
Overview Files 1 Commits 2
1 changed file
RepositoryVorlage
70 SSAS/SSAS_DWH
Model.bim
Model.bim
-----
2619 2619      "name": "dim_GLAccount_HIST",
2620 2620      "columns": [
2621 2621      {
2622 2622      - "name": "GL Account HIST - Valid From",
2623 2623      + "name": "GL Account HIST - Valid From",
2624 2624      - "dataType": "dateTime",
2625 2625      + "dataType": "dateTime",
2626 2626      - "sourceColumn": "GL Account HIST - Valid From",
2627 2627      + "sourceColumn": "GL Account HIST - Valid From",
2628 2628      - "sourceProviderType": "date",
2629 2629      + "sourceProviderType": "date",
2630 2630      - "displayFolder": "00 GL Account HIST",
2631 2631      + "displayFolder": "00 GL Account HIST",
2632 2632      - "summarizeBy": "none",
2633 2633      + "summarizeBy": "none",
2634 2634      },
2635 2635      {
2636 2636      - "name": "GL Account HIST - Valid To",
2637 2637      + "name": "GL Account HIST - Valid To",
2638 2638      - "dataType": "dateTime",
2639 2639      + "dataType": "dateTime",
2640 2640      - "sourceColumn": "GL Account HIST - Valid To",
2641 2641      + "sourceColumn": "GL Account HIST - Valid To",
2642 2642      - "sourceProviderType": "date",
2643 2643      + "sourceProviderType": "date",
2644 2644      - "displayFolder": "00 GL Account HIST",
2645 2645      + "displayFolder": "00 GL Account HIST",
2646 2646      - "summarizeBy": "none",
2647 2647      + "summarizeBy": "none",
2648 2648      },
2649 2649      ],
2650 2650      "summarizeBy": "none"
2651 2651      },
2652 2652      "SELECT",
2653 2653      "\t [GLAccount_HIST_ID]",
2654 2654      - "\t,[GL Account HIST - Valid From]",
2655 2655      + "\t,[GL Account HIST - Valid To]",
2656 2656      - "\t,[GL Account HIST - Valid To]",
2657 2657      + "\t,[GL Account HIST - Valid To]",
2658 2658      - "\t,[GL Account HIST - Concat]",
2659 2659      + "\t,[GL Account HIST - Concat]",
2660 2660      - "\t,[GL Account HIST - Desc]",
2661 2661      + "\t,[GL Account HIST - Desc]",
2662 2662      - "\t,[GL Account HIST - is Active]",
2663 2663      + "\t,[GL Account HIST - is Active]",
2664 2664      ]
-----
```

Anpassung am View vergessen
[feature/sqlkonferenz-broken](#)

```
Overview Files Updates Commits
All Changes Filter 2 changed files
RepositoryVorlage
50 SSDT/SSDT_DWH/DVL_SSAS_DWH/Vi...
dim_GLAccount_HIST.sql
70 SSAS/SSAS_DWH
Model.bim
dim_GLAccount_HIST.sql -2+2
/50 SSDT/SSDT_DWH/DVL_SSAS_DWH/Views/dim_GLAccount_HIST.sql
View
-----
2 2      SELECT
3 3      [GLAccount_HIST_ID]
4 4      , [GL Account HIST]
5 5      - , [GL Account HIST - Valid From]
6 6      + , [GL Account HIST - Valid To] AS [GL Account HIST - Valid From]
7 7      - , [GL Account HIST - Valid To] AS [GL Account HIST - Valid To]
8 8      + , [GL Account HIST - Concat]
9 9      , [GL Account HIST - Desc]
-----
Model.bim -6+6
/70 SSAS/SSAS_DWH/Model.bim
View
-----
2619 2619      "name": "dim_GLAccount_HIST",
2620 2620      "columns": [
2621 2621      {
2622 2622      - "name": "GL Account HIST - Valid From",
2623 2623      + "name": "GL Account HIST - Valid From",
2624 2624      - "dataType": "dateTime",
2625 2625      + "sourceColumn": "GL Account HIST - Valid From",
2626 2626      - "sourceProviderType": "date",
2627 2627      + "displayFolder": "00 GL Account HIST",
2628 2628      - "summarizeBy": "none",
2629 2629      },
2630 2630      {
2631 2631      - "name": "GL Account HIST - Valid To",
2632 2632      + "name": "GL Account HIST - Valid To",
2633 2633      - "dataType": "dateTime",
2634 2634      + "sourceColumn": "GL Account HIST - Valid To",
2635 2635      - "sourceProviderType": "date",
2636 2636      + "displayFolder": "00 GL Account HIST",
2637 2637      - "summarizeBy": "none",
2638 2638      },
2639 2639      ],
2640 2640      "summarizeBy": "none"
2641 2641      },
2642 2642      "SELECT",
2643 2643      "\t [GLAccount_HIST_ID]",
2644 2644      - "\t,[GL Account HIST]",
2645 2645      + "\t,[GL Account HIST - Valid From]",
2646 2646      - "\t,[GL Account HIST - Valid To]",
2647 2647      + "\t,[GL Account HIST - Valid To]",
2648 2648      - "\t,[GL Account HIST - Concat]",
2649 2649      + "\t,[GL Account HIST - Concat]",
2650 2650      - "\t,[GL Account HIST - Desc]",
2651 2651      + "\t,[GL Account HIST - Desc]",
2652 2652      - "\t,[GL Account HIST - is Active]",
2653 2653      + "\t,[GL Account HIST - is Active]",
2654 2654      ]
-----
```

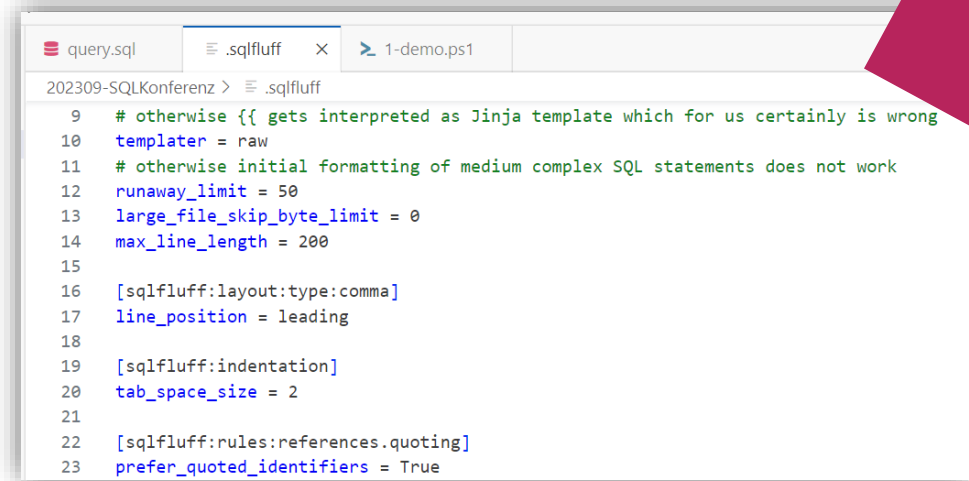
Ganzheitliche Anpassung
[feature/sqlkonferenz-good](#)

Gleiche Formatierung I/III

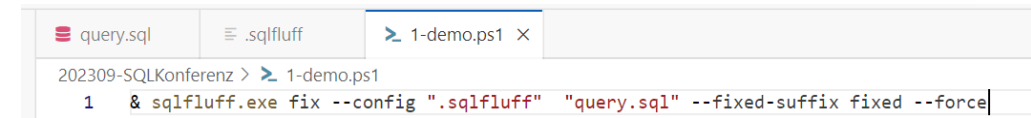
- Bei paralleler Entwicklung entstehen Konflikte, diese sollten vermieden werden
- Pull Request lassen sich nur effizient reviewen, wenn der diff möglichst klein ist
- Git unterstützt „pre-commit“ hooks, um dies zu automatisieren
- Pre-commit <https://pre-commit.com/>
 - Framework für Tools, die vor einem Commit ausgeführt werden
 - Bspw. Leerzeichen statt Tabs, einheitliche Windows Zeilenumbrüche, einheitlich UTF-8, XML-Formatierung, YAML-Validierung
 - Wird mit `pre-commit install` in einem git Repository aktiviert

Gleiche Formatierung II/III

- Sqlfluff <https://github.com/sqlfluff/sqlfluff>
 - Formatiert SQL Code für fast jede Datenbank
 - Sehr feingranular konfigurierbar
 - Einheitliches Casting (**CAST** vs **CONVERT**)
 - einheitliches Aliasing (**AS** vs „“)
 - einheitliche Reihenfolge in **JOINS**
 - Einheitliche Kommas
 - Perfekte Einrückung
- Leider nicht sehr schnell



```
query.sql | .sqlfluff | 1-demo.ps1
202309-SQLKonferenz > .sqlfluff
9 # otherwise {{ gets interpreted as Jinja template which for us certainly is wrong
10 templater = raw
11 # otherwise initial formatting of medium complex SQL statements does not work
12 runaway_limit = 50
13 large_file_skip_byte_limit = 0
14 max_line_length = 200
15
16 [sqlfluff:layout:type:comma]
17 line_position = leading
18
19 [sqlfluff:indentation]
20 tab_space_size = 2
21
22 [sqlfluff:rules:references.quoting]
23 prefer_quoted_identifiers = True
```



```
query.sql | .sqlfluff | 1-demo.ps1 X
202309-SQLKonferenz > 1-demo.ps1
1 & sqlfluff.exe fix --config ".sqlfluff" "query.sql" --fixed-suffix fixed --force
```

```
SELECT
  c.Customer_KEY
  , m.Material_KEY,
  CONVERT(int, Quantity) AS Quantity,
  Price * Quantity Amount
  , case when MaterialType_KEY = 'ZC' THEN 'A' ELSE CASE WHEN
MaterialType_KEY = 'ZD' THEN 'B' ELSE 'C' END END AS
MaterialCategory_KEY
FROM DPL.fact_sales s
left join DPL.dim_customer c ON (c.Customer_ID = f.Customer_ID)
LEFT JOIN DPL.dim_material as m on (f.Material_ID = m.Material_ID)
WHere s.Fiscalyear BETWEEN 2012 and 2023
```

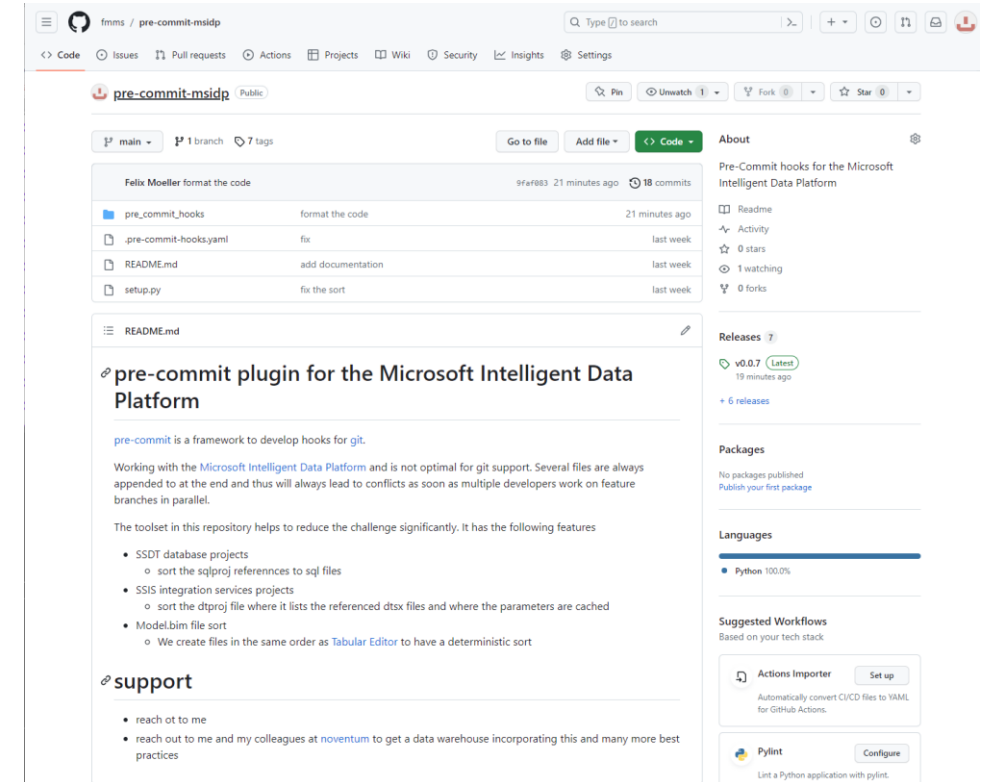
vorher

```
SELECT
  c.Customer_KEY
  , m.Material_KEY
  , CAST (Quantity AS int) AS Quantity
  , Price * Quantity AS Amount
  , CASE
  WHEN MaterialType_KEY = 'ZC' THEN 'A'
  WHEN MaterialType_KEY = 'ZD' THEN 'B'
  ELSE 'C'
  END AS MaterialCategory_KEY
FROM DPL.fact_sales f
LEFT JOIN DPL.dim_customer c ON (c.Customer_ID = f.Customer_ID)
LEFT JOIN DPL.dim_material m ON (m.Material_ID = f.Material_ID)
WHERE s.Fiscalyear BETWEEN 2012 AND 2023
```

nachher

Gleiche Formatierung III/III

- Größtes Konfliktpotential in Projektdateien für SSIS, SSDT
 - Neue Artefakte immer unten angehängt
 - Immer ein Merge-Konflikt
- Intelligent Data Platform hooks <https://github.com/fmms/pre-commit-msidp>
 - sortiert alle Einträge im **Build** und **Folder** der Projektdatei (**.sqlproj**)
 - Sortiert alle Referenzen auf SSIS-Pakete in Projektdatei (**.dtproj**)
 - Sortiert alle Objekte in **Model.bim** entsprechend der Tabular Editor Serialisierung – effizientere Reviews



```
Sorting files
PS C:\Users\fmoller\Desktop\HRReportingRepository\70 SSAS\SSAS_HRA_DWH> pre-commit run --all
Sorting the files referenced in SSDT-SSDT project.....Passed
Sorting the files referenced in SSDT-SSIS project.....Passed
Sorting the datasources, tables, relationships etc in Model.bim.....Passed
trim trailing whitespace.....Passed
fix end of files.....Passed
check yaml.....Passed
check for added large files.....Passed
mixed line ending.....Passed
fix utf-8 byte order marker.....Passed
PS C:\Users\fmoller\Desktop\HRReportingRepository\70 SSAS\SSAS_HRA_DWH> |
```

Agenda

1 Überblick und Zielsetzung

2 Zusammenarbeit mit Git

3 Build und Deploy

3.1 Integration Services

3.2 SQL Server Database Projects

3.3 Tabular Models






3.4 Azure Data Factory

4 Testing

5 Zusammenfassung

6 Fragen / Diskussion

Was passiert im Build-Vorgang

Technologie	Source Code	Deployment Artefakt	
 Integration Services	*.dtsx	*.ispac	1
 SQL Datenbank	*.sql	*.dacpac	2
 Analysis Services	Model.bim	Model.asdatabase	3
 Data Factory	*.json	ARM-Template *.json mit azure.datafactory.tools	4
 Power BI	*.pbip, *.pbix	*.pbix	

Agenda

1 Überblick und Zielsetzung

2 Zusammenarbeit mit Git

3 Build und Deploy

3.1 Integration Services

3.2 SQL Server Database Projects

3.3 Tabular Models

3.4 Azure Data Factory

4 Testing

5 Zusammenfassung

6 Fragen / Diskussion

Integration Services

Failed flatfile extractions do not stop the overall process (because of security mechanisms within the file extraction)

Checkpoints	
CheckpointFileName	
CheckpointUsage	Never
SaveCheckpoints	False
Execution	
DelayValidation	False
Disable	False
DisableEventHandlers	False
FailPackageOnFailure	False
FailParentOnFailure	False
MaxConcurrentExecutables	-1
Name	
Specifies the name of the object.	

Integration Services 1.1
In Visual Studio 2022

1 Integration Services

Build

- Git Integration seit Visual Studio 2017
- SSIS Pakete sind XML-Dateien
 - Durch enthaltene Metadaten sehr umfangreich
 - Layout Informationen sorgen für Konflikte
- Pre-commit kann verwendet werden um Layoutinformationen zu entfernen, entfernt aber auch Kommentare ...

Deploy

- SSIS normalerweise deployt mit `ISDeploymentWizard.exe`
- Deployment in Azure-SSIS Runtime mit AccessToken nur möglich über SMO (PowerShell) ([Deploy an SSIS project with PowerShell - SQL Server Integration Services \(SSIS\) | Microsoft Learn](#))
- PowerShell Deployment funktioniert nicht auf MSFT-hosted build agents (<https://github.com/actions/runner-images/issues/8174>)
- Konfiguration über Stored Procedures in der SSISDB (e.g. `[catalog].[set_object_parameter_value]`)

Learnings

- Visual Studio auf Microsoft Hosted Agents ist SEHR langsam – etwa 4 – 9-fache Geschwindigkeit mit Self-Hosted Agent
- `ISDeploymentWizard.exe` unterstützt keine AccessToken-Authentifizierung
- Deployment mit PowerShell über SMO auf onPrem Server ist sehr langsam.

Agenda

1 Überblick und Zielsetzung

2 Zusammenarbeit mit Git

3 Build und Deploy

3.1 Integration Services

3.2 SQL Server Database Projects

3.3 Tabular Models

3.4 Azure Data Factory

4 Testing

5 Zusammenfassung

6 Fragen / Diskussion

2

Datenbankobjekte werden im Datenbankprojekt verwaltet

The screenshot shows the Visual Studio 2022 interface with a SQL script editor on the left and a 'Umbenennen' (Rename) dialog box in the center. The script contains SQL code for a MERGE statement and manual values. The dialog box is used to rename a table from 'INTERNAL_ETL.task' to 't'. The background shows the Project Explorer with a database project structure.

```

10      ('DPL_dim_GLAccount_HIST', 1, 'SP', 'INTERNAL_ETL.load
11      ('DPL_fact_AccountsPayableLineItem', 1, 'SP', 'INTERNAL
12      ('DAL_SAP_md_OGL_ACCOUNT_ATTR', 1, 'SSIS', 'DAL_SAP_md.
13      ('DAL_SAP_md_OGL_ACCOUNT_TEXT', 1, 'SSIS', 'DAL_SAP_md.
14      ('DAL_SAP_td_0FI_AP_4', 1, 'SSIS', 'DAL_SAP_td_0FI_AP_4
15      ('DAL_SAP_md_OGL_ACCOUNT_T011_HIER', 0, 'SSIS', 'DAL_S
16      ) manualvalues ([Task], [isEnabled], [TaskType], [Executab
17      )
18
19      MERGE [INTERNAL_ETL].[task] t
20      USING manualvalues
21      ON (
22      t.[Task] = s.
23      )
24      WHEN NOT MATCHED E
25      THEN
26      INSERT (
27      [Task]
28      , [isEnabled]
29      , [TaskType]
30      , [Executable]
31      , [Layer]

```

The screenshot shows the SQL Server Enterprise Manager interface. The 'Table Properties' window is open for the table 'md_OGL_ACCOUNT_ATTR'. The 'Columns' tab shows the table structure with columns like 'tst', 'JobID', 'JobTST', 'KTOPL', 'SAKNR', 'BILKT', 'GVTYP', 'VBUND', 'XBILK', 'SAKAN', 'ERDAT', and 'KTOKS'. The 'Scripts' tab shows the CREATE TABLE script for the table.

Move	Name	Type	Primary Key	Allow Null
	tst	datetime2(3)	<input type="checkbox"/>	<input type="checkbox"/>
	JobID	bigint	<input type="checkbox"/>	<input type="checkbox"/>
	JobTST	datetime2(7)	<input type="checkbox"/>	<input type="checkbox"/>
	KTOPL	nvarchar(4)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	SAKNR	nvarchar(10)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	BILKT	nvarchar(10)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	GVTYP	nvarchar(2)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	VBUND	nvarchar(6)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	XBILK	nvarchar(1)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	SAKAN	nvarchar(10)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	ERDAT	date	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	ERNAM	nvarchar(12)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	KTOKS	nvarchar(4)	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Datenbankprojekte in Azure Data Tools basieren auf neuem Projekttyp, aber haben noch sehr viele fehlende Funktionen. (vgl. [SQL Database Projects - add refactoring \(which is available in SSDT\) - Issue #13920 - microsoft/azuredatstudio \(github.com\)](#)).

SSDT in Visual Studio 2022/2019 Unterstützt Umbenennungen, Refactorings, etc.

2 Datenbankobjekte

Build

- Git Integration direkt seit Visual Studio 2017
- Datenbankobjekte werden als SQL-Datei abgespeichert
 - `.refactorlog` für Umbenennungen
- Postdeployment-Skripte
- Ergebnis ist eine `dacpac`-Datei

Deploy

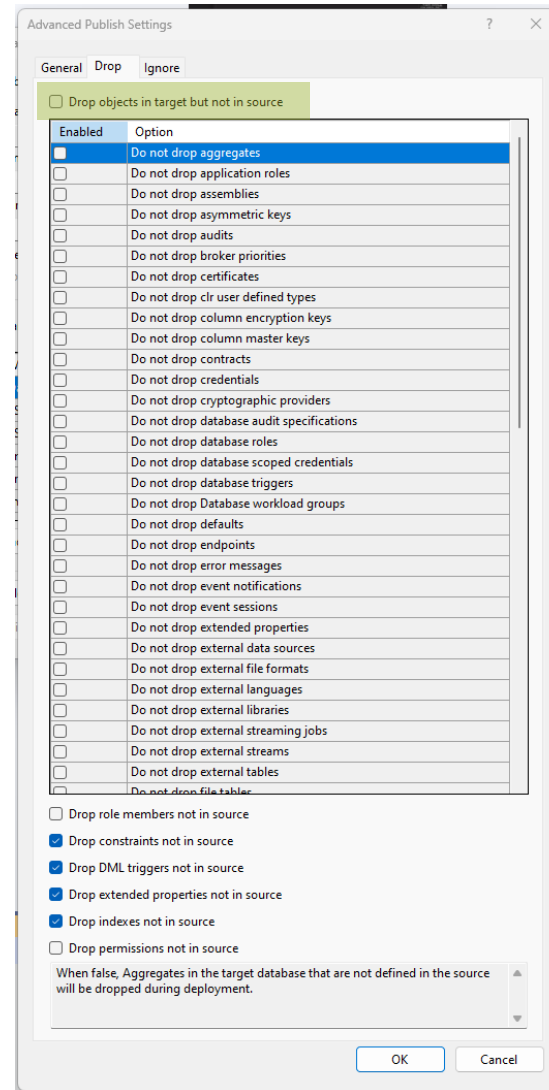
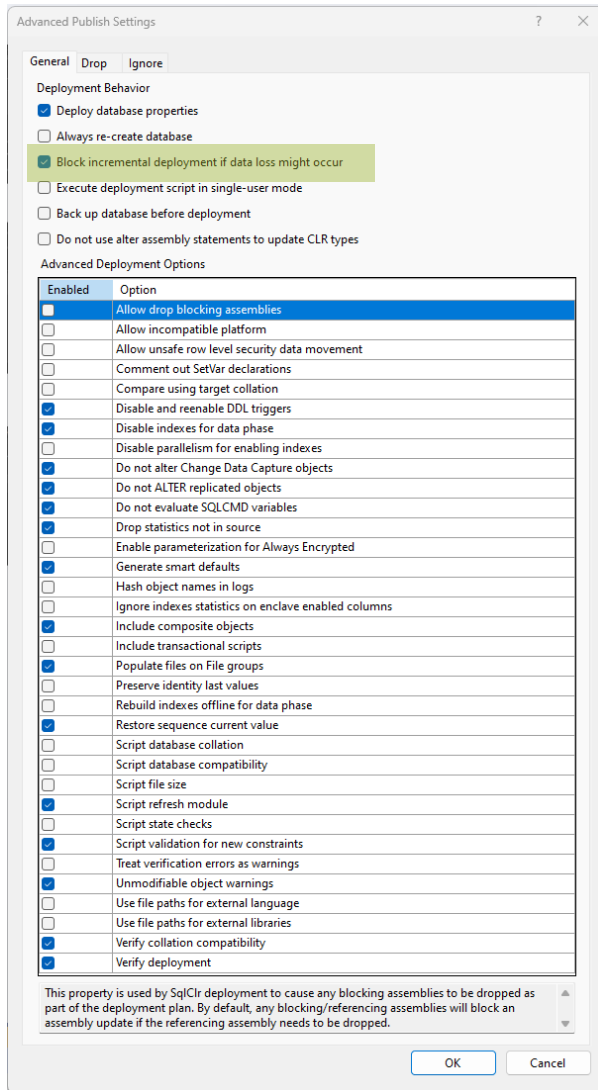
- Dacpac gibt den Sollzustand vor, Datenbank wird per `sqlpackage.exe` entsprechend angepasst
- Allgemeine Deployment Optionen
 - State-basierend: SSDT, [dbForge Source Control for SQL Server](#)
 - Migration-basierend: [DbUp](#)

Learnings

- Ausführungsreihenfolge von `sqlpackage.exe`
 1. Vergleich DacPac zu Datenbank
 2. Erzeugen eines Deploymentscripts
 3. Ausführen des Pre-Deploymentscripts
 4. Ausführen des Deploymentscripts (vgl. Schritt 2)
 5. Ausführen des Post-Deployment
- Pre-Pre-Deployment-Skript notwendig

Vgl. <https://blogs.msdn.microsoft.com/gertd/2009/09/14/pre-deployment-scripts/>
und <https://the.agilesql.club/2015/07/pre-compare-pre-deployment-scripts-to-ssdt/>

2 Hunderte Optionen im Publish-Profil



Block on Possible Data Loss

- Sicherstellen, dass es zu keinem Verlust kommt

Generate Smart Defaults

- Hilfreich für non-nullable columns mit Default Constraints

Drop objects not in source

- Notwendig um im Ziel nicht immer weitere Objekte anzusammeln

Ignore Whitespace

- Notwendig um nicht mit sqlfluff Formatierungen in Endlosschleife zu landen.

Script database collation

- Project auf Case Sensitive, Datenbank kann auf Case insensitive bleiben

Agenda

- 1 Überblick und Zielsetzung
- 2 Zusammenarbeit mit Git
- 3 Build und Deploy
 - 3.1 Integration Services
 - 3.2 SQL Server Database Projects
 - 3.3 Tabular Models
 - 3.4 Azure Data Factory
- 4 Testing
- 5 Zusammenfassung
- 6 Fragen / Diskussion

3 Tabular Models

Build

- Entwicklung in Tabular Editor 2 (<https://github.com/TabularEditor/TabularEditor>)
- Microsoft AS Designer vernachlässigt ([Microsoft Analysis Services Projects - Visual Studio Marketplace](#))
 - Visual Studio 2022 letztes Update April 2022
 - Visual Studio 2019 letztes Update Juli 2021
- Provider- statt Structured-DataSources zu bevorzugen (vgl. [\[1\]](#) und [\[2\]](#))

Learnings

- Authentifizierung gegen Azure Analysis Services ist möglich mit der Service Connection aus Azure DevOps: `User ID= ;Password=AccessToken`

Deploy

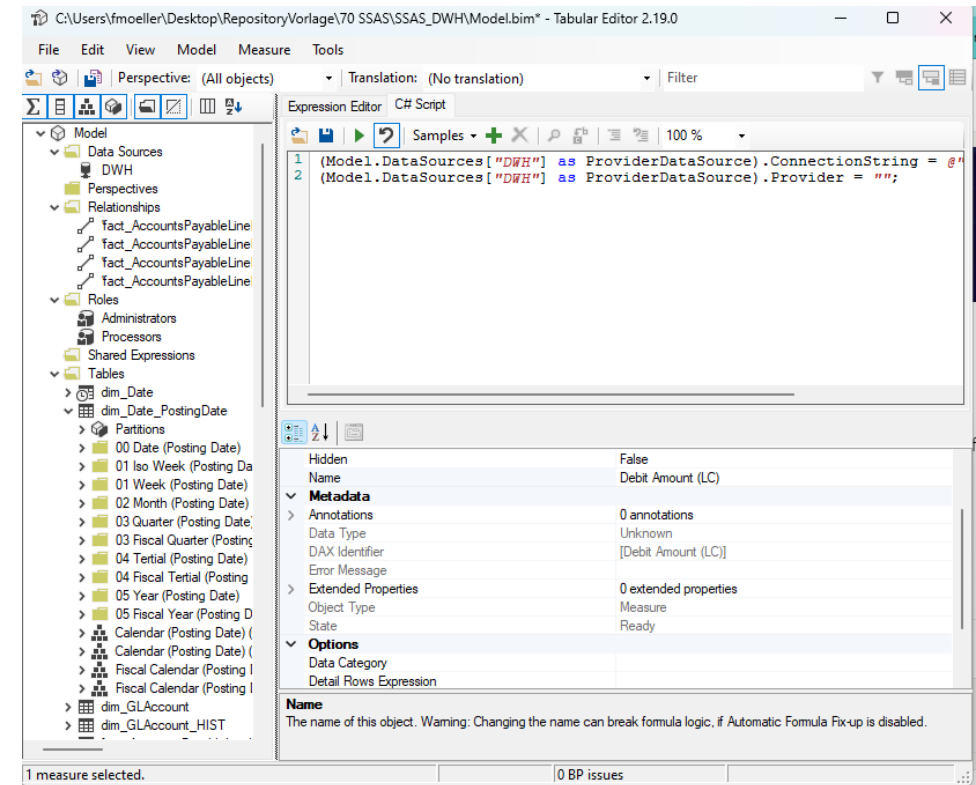
- `Microsoft.AnalysisServices.Deployment.exe` Teil von SSMS nicht vorhanden auf MSFT-hosted Agents
- Tabular Editor erlaubt sehr bequemes Deployment, Datasources können im Deployment über ein C#-Skript aktualisiert werden.
- XMLA-Endpunkte (Premium) ermöglicht Deployment von Power BI Datasets

```
$saasato = $(Get-AzAccessToken -Resource "https://westeurope.asazure.windows.net").Token
$modelbim = "$(Agent.BuildDirectory)\mybuild\arti1\cube\Model.asdatabase"
$sasModel = "RepositoryVorlage"
$connstr = "Provider=MSOLAP;Data Source=$(asServerFQDN);User ID=;Password=$saasato"

$cmd = "C:\Program Files (x86)\Tabular Editor\TabularEditor.exe"
$params = @(
  "$modelbim",
  "-DEPLOY", "$connstr", "$sasModel",
  "-ROLES",
  "-MEMBERS",
  "-OVERWRITE",
  "-CONNECTIONS",
  "-SCRIPT", "model.cs",
  "-VSTS"
)
$p = Start-Process -Wait -NoNewWindow -PassThru -FilePath $cmd -ArgumentList $params
exit($p.ExitCode)
```


3 Tabular Editor

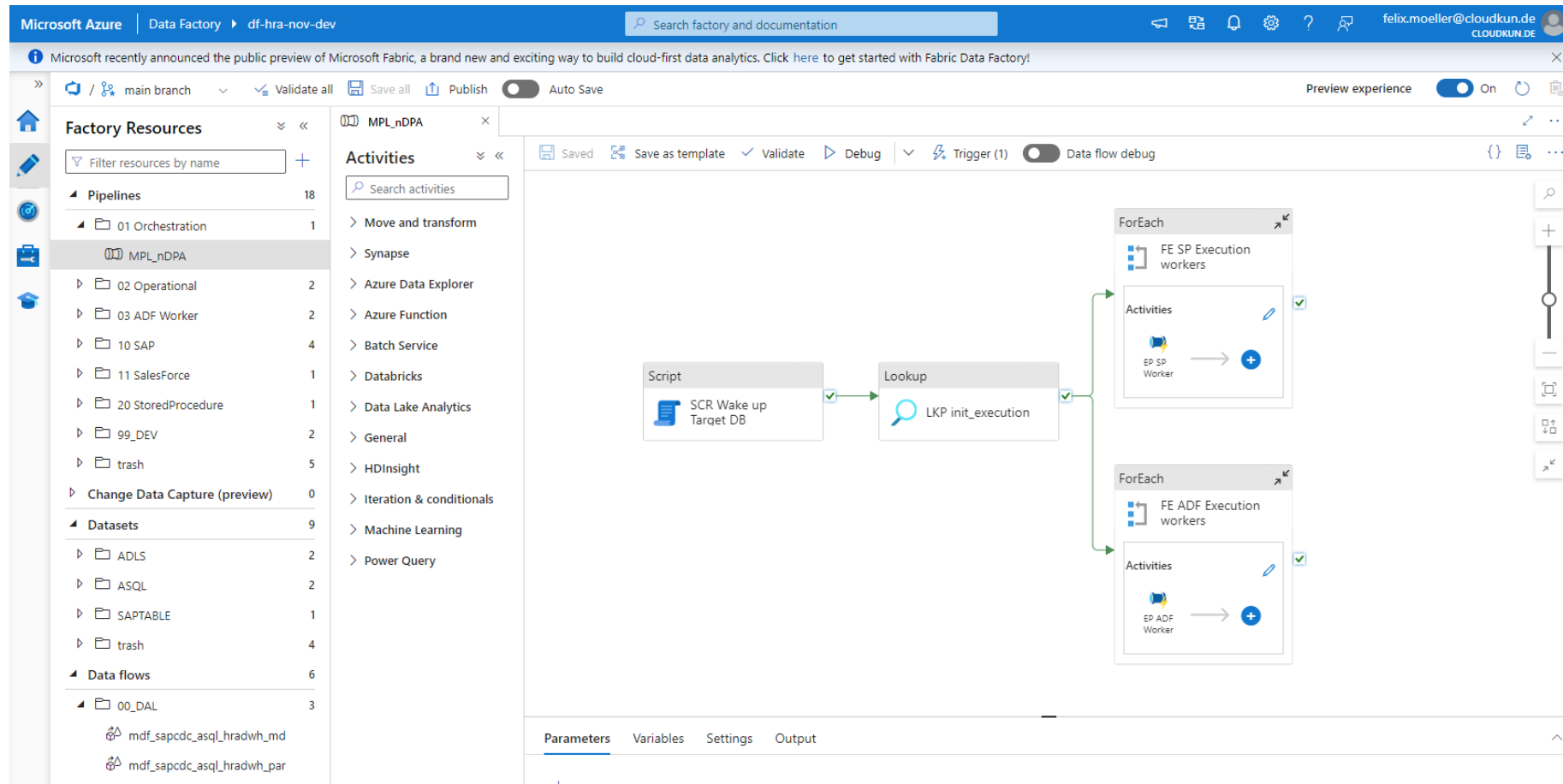
- Entwicklung von Daniel Otykier
- Viel schneller und stabiler als SSAS in VS
- Erlaubt es Best Practices zu validieren (vgl. [GitHub - TabularEditor/BestPracticeRules: An official collection of standard Rules for use with Tabular Editor's Best Practice Analyzer.](#))
- Integration mit DaxFormatter (vgl. [DAX Formatter by SQLBI](#))
- C# Skripte möglich zur Automatisierung
- Kommandozeile bietet viele Optionen zur Automatisierung
- Kann als Ersatz für Visual Studio verwendet werden, wenn man auf PowerQuery/Structured-Datasources verzichtet
- [Tabular Editor 3](#) – Entwicklungsumgebung mit u.a. DAX Intellisense – 90 EUR/Benutzer/Monat



Agenda

- 1 Überblick und Zielsetzung
- 2 Zusammenarbeit mit Git
- 3 Build und Deploy
 - 3.1 Integration Services
 - 3.2 SQL Server Database Projects
 - 3.3 Tabular Models
 - 3.4 Azure Data Factory
- 4 Testing
- 5 Zusammenfassung
- 6 Fragen / Diskussion

Azure Data Factory



The screenshot displays the Azure Data Factory web interface. The top navigation bar shows 'Microsoft Azure | Data Factory | df-hra-nov-dev' and a search bar. Below the navigation bar, there's a notification about Microsoft Fabric. The main interface is divided into several sections:

- Factory Resources:** A sidebar on the left with a search filter and a tree view showing folders like '01 Orchestration', 'MPL_nDPA', '02 Operational', etc.
- Activities:** A central sidebar with a search filter and a list of activity categories such as 'Move and transform', 'Synapse', 'Azure Data Explorer', etc.
- Pipeline Design:** The main workspace showing a pipeline with the following sequence of activities:
 - Script:** 'SCR Wake up Target DB'.
 - Lookup:** 'LKP init_execution'.
 - ForEach (top):** 'FE SP Execution workers' containing an 'EP SP Worker' activity.
 - ForEach (bottom):** 'FE ADF Execution workers' containing an 'EP ADF Worker' activity.
- Bottom Panel:** Tabs for 'Parameters', 'Variables', 'Settings', and 'Output'.

ADF
Branching und git Integration direkt im
Browser.

Data Factory

Build

- Git Integration direkt im ADF Studio
- Azure Data Factory Pipelines werden als JSON abgespeichert
- Zwei Wege zum ARM-Template
 - ARM-Templates in adf_publish Branch (vgl. [Continuous integration and delivery - Azure Data Factory | Microsoft Learn](#))
 - NPM basiertes Build (vgl. [Automated publishing for continuous integration and delivery - Azure Data Factory | Microsoft Learn](#))

Deploy

- Zwei Optionen des Deployments
 - ARM JSON mit Standardtools
 - ADF JSON mit azure.datafactory.tools (vgl. [GitHub - Azure-Player/azure.datafactory.tools: Tools for deploying Data Factory \(v2\) in Microsoft Azure](#))

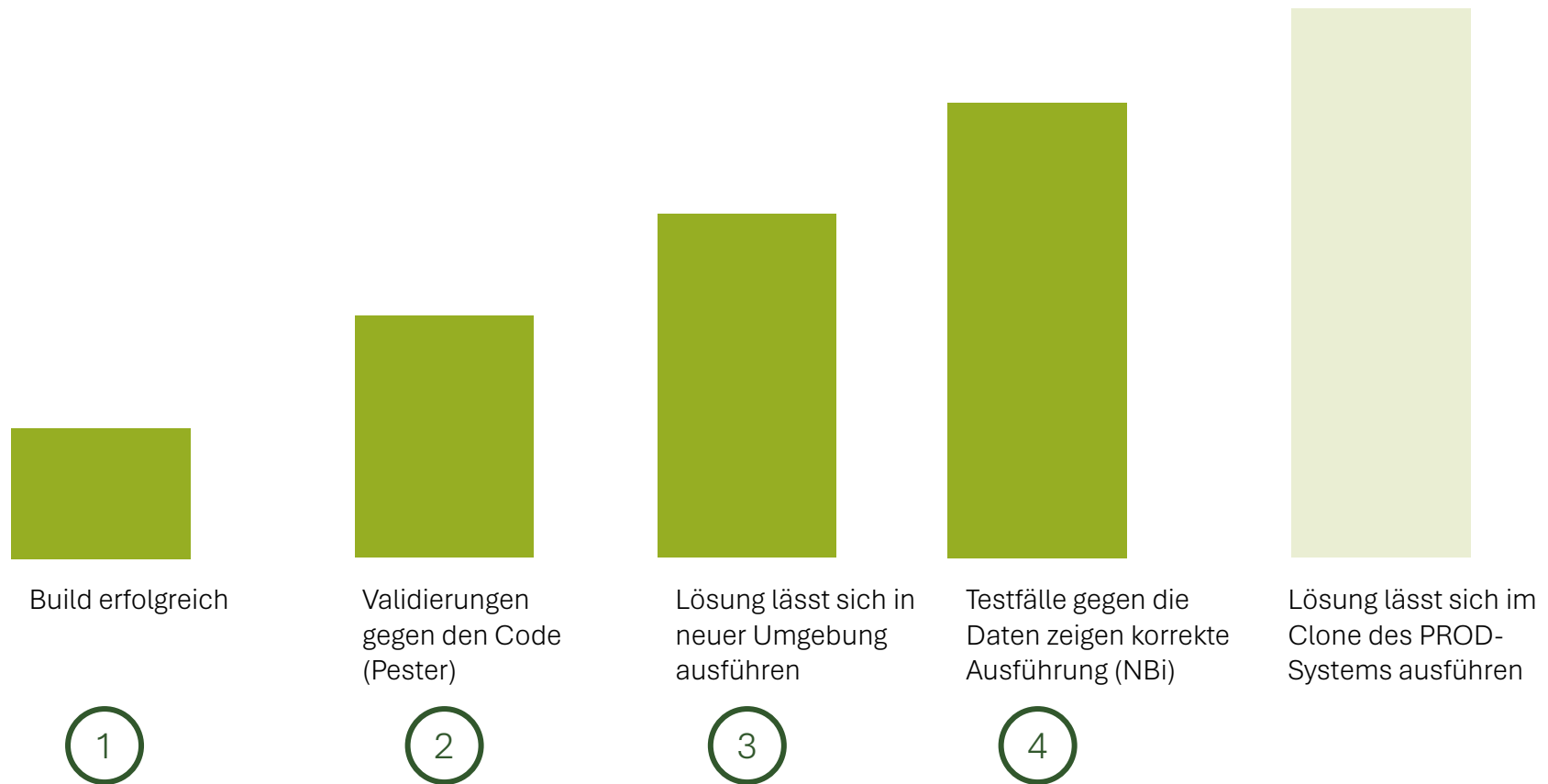
Learnings

- azure.datafactory.tools erlaubt wesentlich bequemeres Deployment

Agenda

- 1 Überblick und Zielsetzung
- 2 Zusammenarbeit mit Git
- 3 Build und Deploy
 - 3.1 Integration Services
 - 3.2 SQL Server Database Projects
 - 3.3 Tabular Models
 - 3.4 Azure Data Factory
- 4 Testing
- 5 Zusammenfassung
- 6 Fragen / Diskussion

Verschiedene Stufen des Testens



1 Build des Projekts ist ein guter erster Test

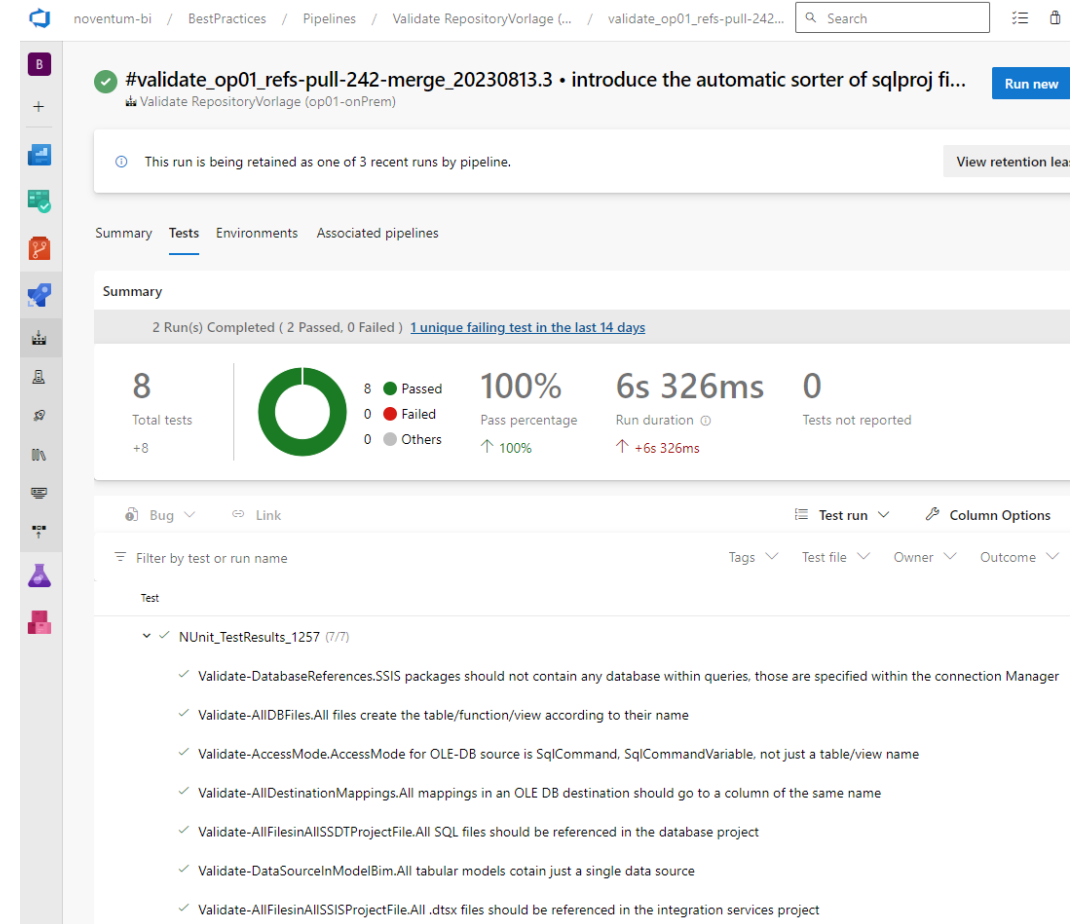
- SSIS
 - fehlende Pakete im Projekt
- SSDT
 - Fehlende Abhängigkeiten – referenzierter View oder Tabelle existiert nicht
 - Syntaxfehler (bspw. SELET statt SELECT)

2 Pester das PowerShell Testing Framework

- Pester ist ein Framework um PowerShell Testfälle zu definieren
 - Kann lokal ausgeführt werden
 - Integriert sich mit Azure DevOps

- SSIS
 - Keinerlei Datenbankqualifizierter Queries in Sources
 - Stets ein SELECT-Statement statt „Table or View“

- Datenbankprojekt
 - Dateinamen passen zum zugehörigen SQL Objekt
 - Alle Dateien sind auch Teil der sqlproj-Datei

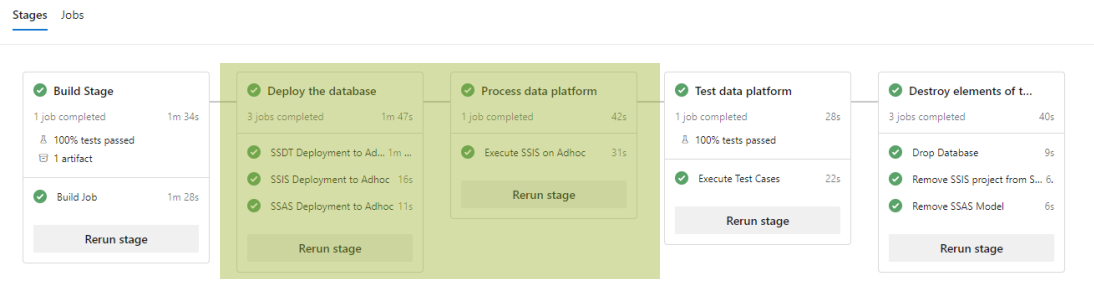


The screenshot shows the results of a pipeline run in Azure DevOps. The run is titled "#validate_op01_refs-pull-242-merge_20230813.3 • introduce the automatic sorter of sqlproj fi...". The summary indicates that 2 runs were completed, with 2 passed and 0 failed. A green donut chart shows 100% pass percentage. The run duration was 6s 326ms. Below the summary, a list of test results is shown, all of which passed:

- ✓ Validate-DatabaseReferences.SSIS packages should not contain any database within queries, those are specified within the connection Manager
- ✓ Validate-AllDBFiles.All files create the table/function/view according to their name
- ✓ Validate-AccessMode.AccessMode for OLE-DB source is SqlCommand, SqlCommandVariable, not just a table/view name
- ✓ Validate-AllDestinationMappings.All mappings in an OLE DB destination should go to a column of the same name
- ✓ Validate-AllFilesInAllSSDTProjectFile.All SQL files should be referenced in the database project
- ✓ Validate-DataSourceInModelBim.All tabular models contain just a single data source
- ✓ Validate-AllFilesInAllSSISProjectFile.All .dtsx files should be referenced in the integration services project

3 Integrationstest

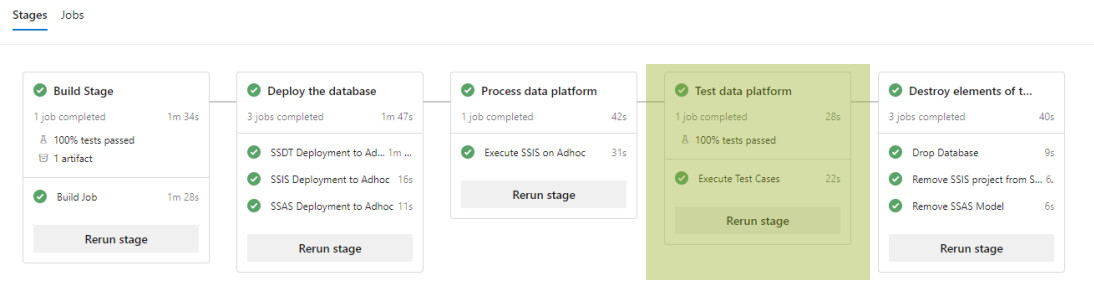
- Viele Fehler verursacht durch Toolbrüche
 - AS greift auf Spalte zu, die es nicht gibt
 - ADF nutzt Stored Procedure, die umbenannt wurde
 - Datentypen zwischen SSIS und der Datenbank weichen voneinander ab
- Umsetzung
 - Deployment in temporäre Umgebung
 - Azure: weitere Ressourcen per bicep (vgl. <https://github.com/Azure/bicep>)
 - onPrem: weitere DBs / Projekte
 - Deaktivieren der Quellsystemanbindung
 - Ausführung des kompletten ETL-Prozesses inkl. Cube-Verarbeitung



4 Datentest

- Einige Fehler brauchen Daten
 - Irgendwo zu viele Daten gefiltert
 - Veränderung der Geschäftslogik

- Umsetzung
 - Samplen von Testdaten aus bestehender Quellsystem-Extraktion
 - Testdaten als Post-Deployment-Skript
 - Validieren von Testfällen nach der Ausführung
 - NBI (vgl. [NBI - Testing framework for Business Intelligence](#)) integriert sich in Azure DevOps



4 Testen mit NBi

- Seit NBi 1.24 können Connection String Informationen in Umgebungsvariablen übergeben werden
- Die Ergebnisse integrieren sich in Azure DevOps

```
- task: PowerShell@2
  displayName: "Download NBi.Framework"
  inputs:
    targetType: 'inline'
    script: |
      $ProgressPreference = 'SilentlyContinue'
      [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::TLS12
      $amo = "https://github.com/Seddryck/NBi/releases/download/v1.24/NBi-Framework-1.24.0.zip"
      $amopath = Join-Path $env:TEMP "NBi.Framework.zip"
      Invoke-WebRequest -Uri $amo -OutFile $amopath
      Expand-Archive $amopath -Force -DestinationPath "$(Agent.BuildDirectory)\mybuild\arti1\tests\Framework"
- task: AzurePowerShell@5
  displayName: 'Get Access Token (Azure)'
  inputs:
    azureSubscription: ${ variables.azureconnectionName }
    azurePowerShellVersion: LatestVersion
    ScriptType: 'InlineScript'
    Inline: |
      $ato = $(Get-AzAccessToken -Resource "https://database.windows.net/").Token;
      Write-Host "##vso[task.setvariable variable=ato]$ato"
- task: PowerShell@2
  displayName: 'Run NBi test suite'
  inputs:
    targetType: 'inline'
    script: |
      $nunit = "$(Agent.BuildDirectory)\mybuild\arti1\tests\NBiTests.nunit"
      & "C:\Program Files (x86)\NUnit 2.7.1\bin\nunit-console.exe" $nunit
  env:
    DWHConnectionString: "Data Source=$(sqlServerFQDN);Initial
Catalog=$(databaseName);Provider=MSOLEDBSQL;Access Token=$(ato);"
- task: PublishTestResults@2
  displayName: "Publish NBi tests"
  condition: always()
  inputs:
    testResultsFormat: 'NUnit'
    testResultsFiles: '**/TestResult.xml'
    failTaskOnFailedTests: true
```

```
<?xml version="1.0" encoding="UTF-8"?>
<testSuite xmlns="http://NBi/TestSuite" name="NBi Test Suite">
  <settings>
    <reference name="DWH">
      <connection-string><environment name="DWHConnectionString"/></connection-
string>
    </reference>
  </settings>
  <test name="Validate Equality">
    <system-under-test>
      <result-set>
        <query connection-string="@DWH">SELECT 42 AS n</query>
      </result-set>
    </system-under-test>
    <assert>
      <equal-to>
        <result-set>
          <row>
            <cell>42</cell>
          </row>
        </result-set>
      </equal-to>
    </assert>
  </test>
</testSuite>
```

Agenda

- 1 Überblick und Zielsetzung
- 2 Zusammenarbeit mit Git
- 3 Build und Deploy
 - 3.1 Integration Services
 - 3.2 SQL Server Database Projects
 - 3.3 Tabular Models
 - 3.4 Azure Data Factory
- 4 Testing
- 5 Zusammenfassung
- 6 Fragen / Diskussion

Status der Pipeline

✓ #validate_op01_refs-pull-264-merge_20230903.2 • try that

Validate RepositoryVorlage (op01-onPrem)

Run new

This run is being retained as one of 3 recent runs by pipeline.

View retention leases

Summary Tests Environments Associated pipelines

Pull request by  Felix Möller

View 2 changes

Repository and version

RepositoryVorlage

264 1df6b793

Time started and elapsed

Today at 17:55

6m 18s

Related

0 work items

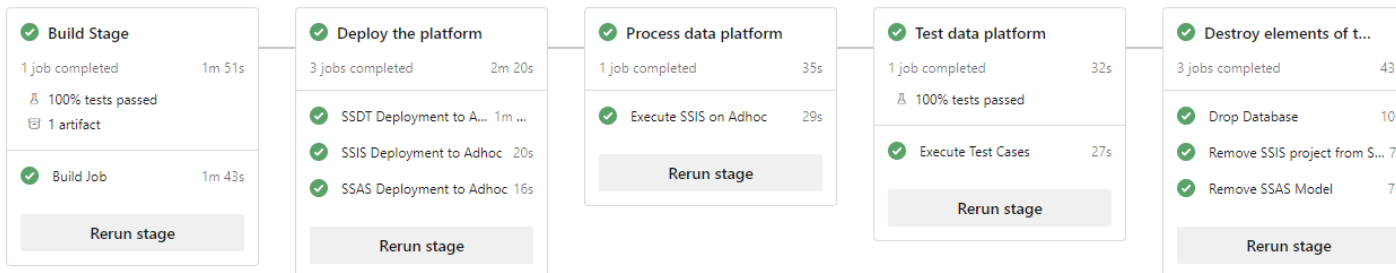
1 published

Tests and coverage

100% passed

Setup code coverage

Stages Jobs



- **Branching** ist essentiell für Zusammenarbeit
- Pre-commit erlaubt Zusammenarbeit mit möglichst wenig Konflikten
- Alle Komponenten ADF / SSIS / SSDT / SSAS erlauben automatisches **Build** und **Deployment**
- Alles gezeigte funktioniert in der Cloud und onPrem
 - Build Agents: **Microsoft-hosted** vs **Self-Hosted Build Agents**
 - Zielumgebung: **onPrem SQL Server** vs. **Azure PaaS Services**
- Testen erlaubt sowohl Qualität im Projekt aufrechtzuerhalten als auch Regressionen zu vermeiden.
- Eine automatische Validation Pipeline findet fast alle technischen Fehler automatisch

Agenda

- 1 Überblick und Zielsetzung
- 2 Zusammenarbeit mit Git
- 3 Build und Deploy
 - 3.1 Integration Services
 - 3.2 SQL Server Database Projects
 - 3.3 Tabular Models
 - 3.4 Azure Data Factory
- 4 Testing
- 5 Zusammenfassung
- 6 Fragen / Diskussion

Start Building your pipeline today!



Thank you very much for your attention.

Vielen Dank für Eure Aufmerksamkeit.