scieneers
DRIVEN BY DATA

Microsoft Fabric

Development- & Deployment-Optionen

Milo Sikora

BI Consultant, Karlsruhe

- Seit 4 Jahren BI Consultant bei scieneers GmbH
- M.Sc. Wirtschaftsingenieurwesen - Karlsruher Institut für Technologie
- Seit 2021 Microsoft Certified Trainer (MCT) für Power BI
- Haupttechnologien in Projekten
  - Azure Data Platform Stack
  - Power BI
  - *Neu: Fabric*
- DIY'ler (3D-Druck, mechanische Tastaturen…)

Fragestellung

scieneers
DRIVEN BY DATA

>> Welche Development- und Deployment-Optionen bietet Fabric und für wen eignen diese sich aufgrund ihrer Vor- und Nachteile?

scieneers
DRIVEN BY DATA

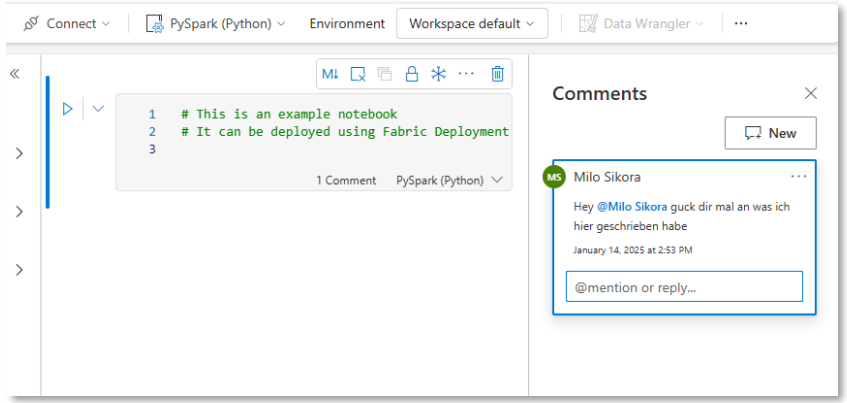
Der Development Prozess

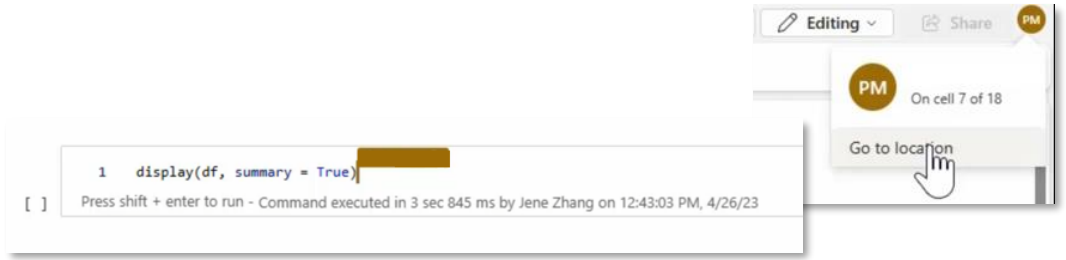
Möglichkeiten in Fabric zu entwicklen

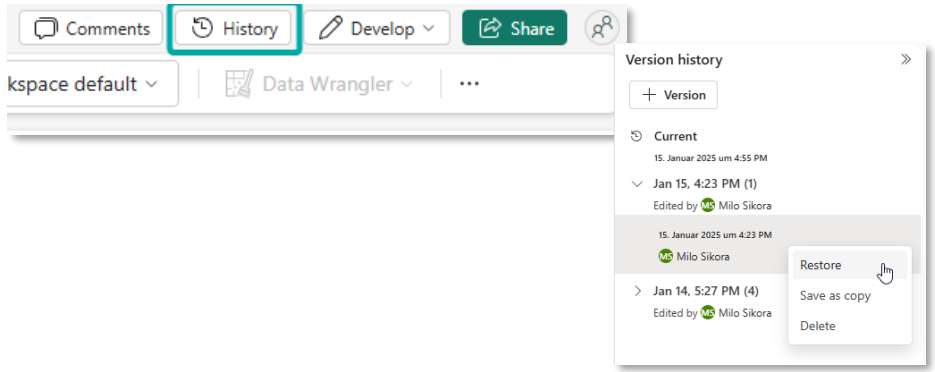
Dev-Prozess 1: Live-Entwicklung in der Fabric GUI

- Der **Standard-Prozess**
- Reine Entwicklung in der **Fabric-Web-Oberfläche**
- Änderungen sind **direkt "live"**
- Notebooks bieten **integrierte Versionierung** inkl. Rollback Funktionalitäten
- In der Notebook-GUI ist **parallele** Kollaboration mit mehreren Personen inkl. Kommentaren möglich
  - Cursor-Highlighting etc.
  - Ermöglicht Paarprogrammierungs-, Remotedebugging- und Tutorenunterrichtsszenarios
- Keine **Isolation**

Es kann **nicht isoliert an Features etc. gearbeitet** werden, ohne das Live System oder den Stand den die anderen Workspace-Mitglieder sehen, zu beeinflussen.

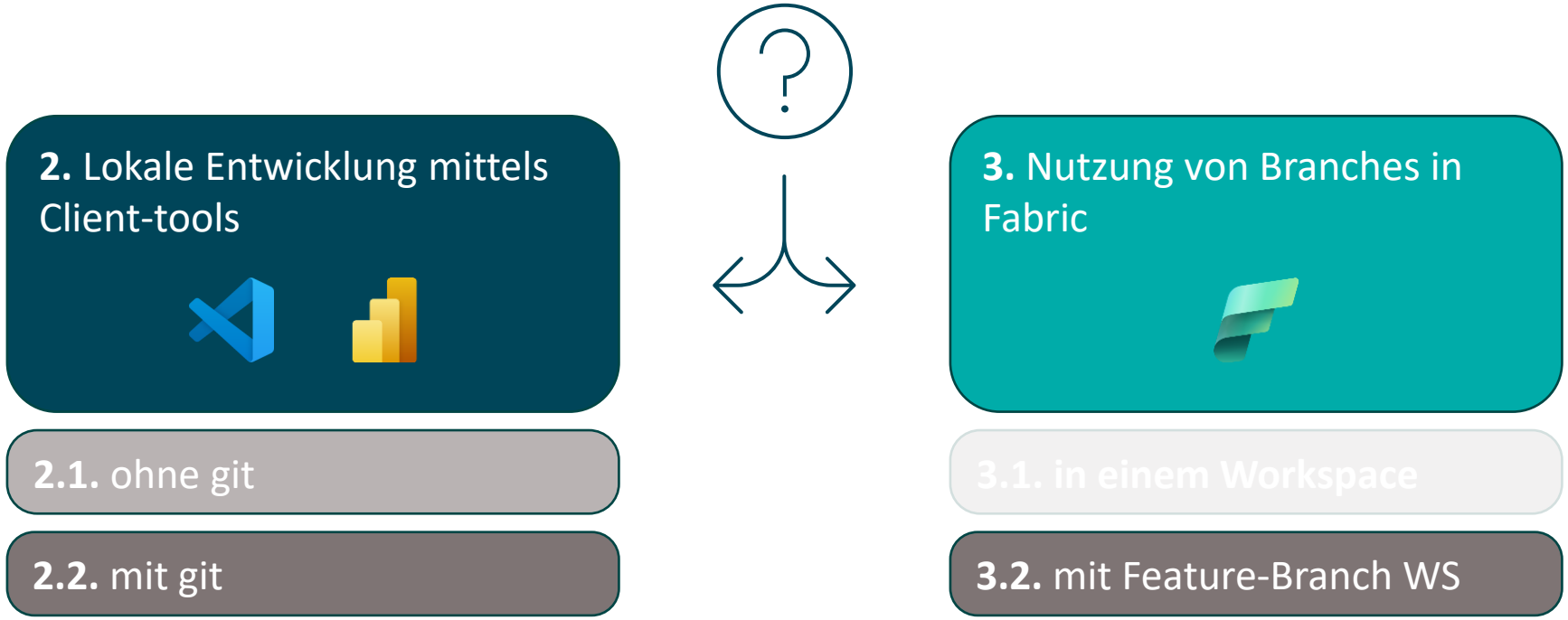






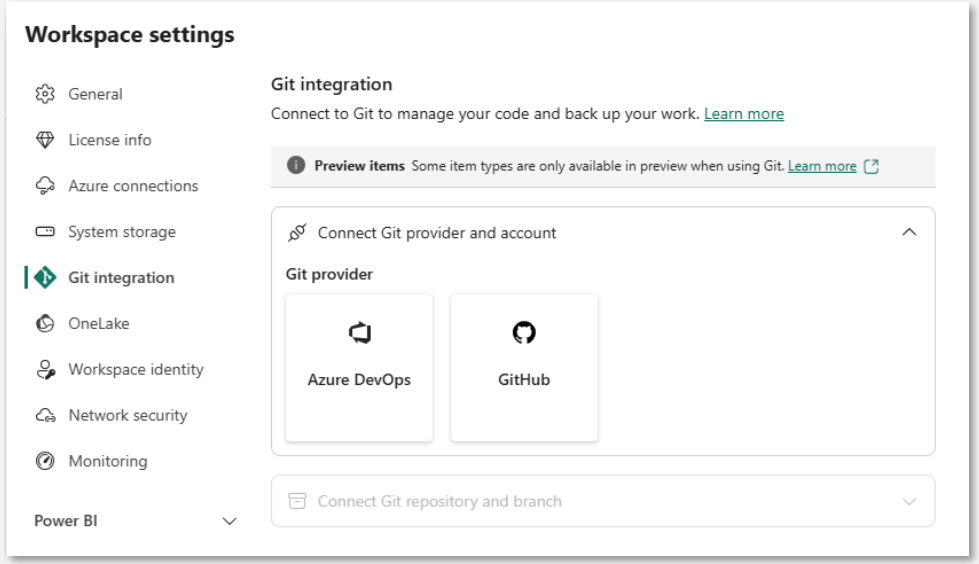
Isolierte Development Prozesse

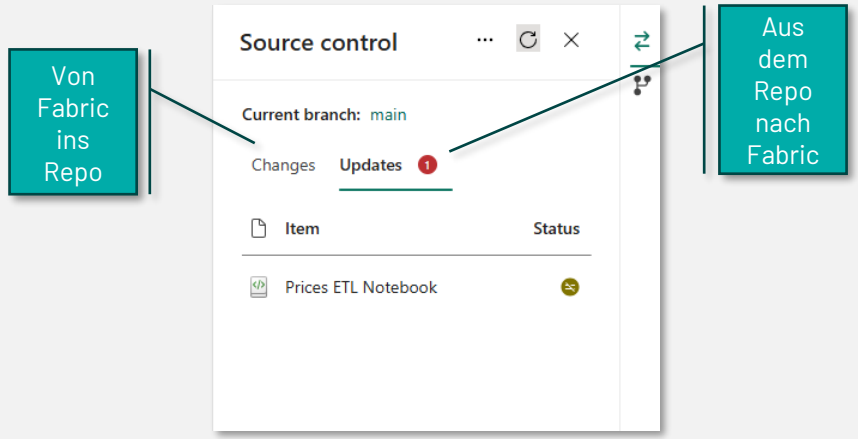
- Es gibt **zwei Möglichkeiten** für einen **isolierten** Development workflow
- Beide Ansätze benötigen (i.d.R.) einen **git-integrierten DEV-workspace**
- Der gewählte **Development-Ansatz** ist grundsätzlich unabhängig vom späteren **Deployment-Modell**



Einschub: Fabric GIT Integration

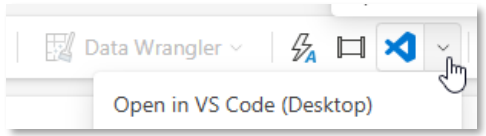
- Fabric Workspace kann an **Azure DevOps** oder **GitHub Repo** angeschlossen werden
- Sync kann in beide Richtungen genutzt werden:
  - Aktuellen Live-Stand **von Fabric ins Repo** sichern
  - Aktuellen Live-Stand mit Änderungen, welche im Repo liegen **überschreiben**

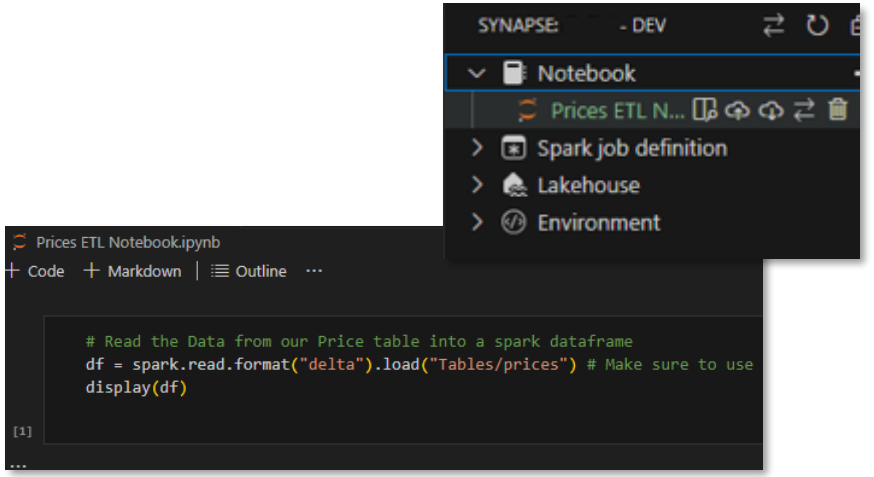




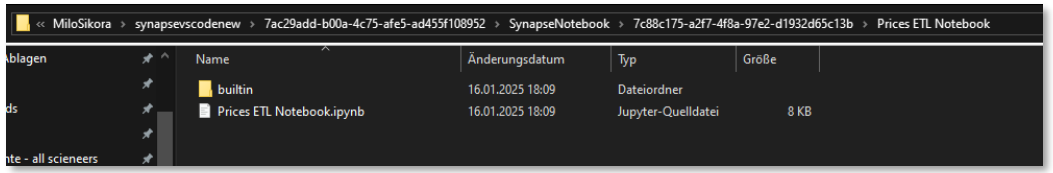
Dev-Prozess 2.1: Lokale Entwicklung ohne git

- Möglich für Fabric-Objekte für die es **lokale Client Tools** gibt
  - .pbix Power BI Report → Power BI Desktop
  - .ipynb Fabric Notebook → VS Code mit Fabric Extension
- Es ist kein **lokales git-repository notwendig** / nutzbar
- Über **VS Code Extension** kann Notebook lokal bearbeitet werden
  - Es wird eine **lokale Kopie** des .ipynb Notebooks in separatem Ordner (nicht git Ordner!) erzeugt
  - Notebook kann lokal mit den üblichen **Jupyter-Notebook Funktionalitäten** [Zellen, Output …] ausgeführt werden
  - Notebook kann nach Fertigstellung der lokalen Änderungen über die VS Code Extension wieder in das Fabric Notebook übertragen werden
  - Ermöglicht die Nutzung von GitHub Copilot etc., sodas man nicht auf den Fabric Copilot (ab F64) angewiesen ist





- **Nach dem Upload** in den Fabric WS, könnte man den change optional dann von dort in das Repo publishen, wenn der Fabric WS git-integrated ist
  - → Git als reine Versionierung, nicht für Branching Funktionalität

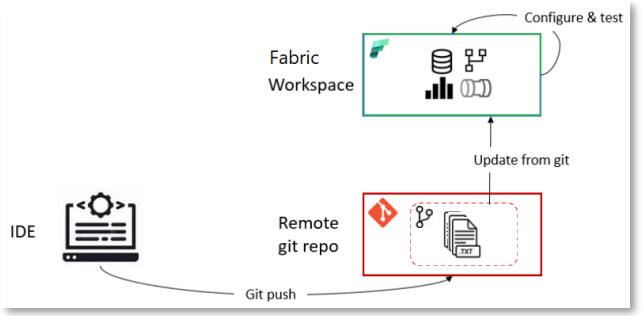


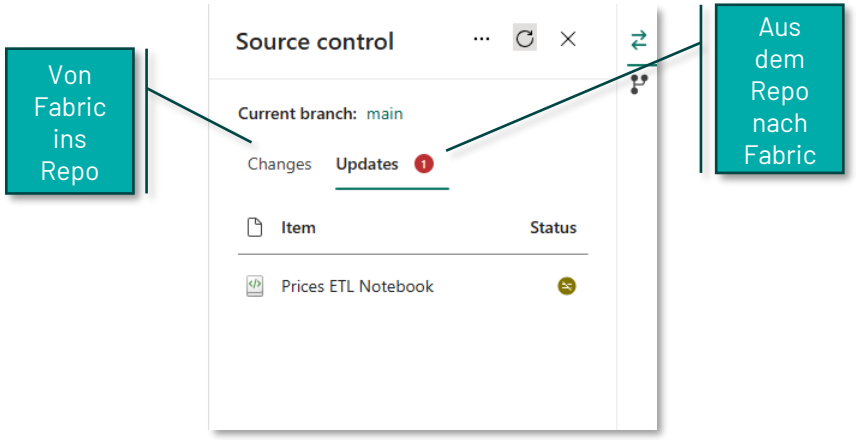
DEMO 2_1

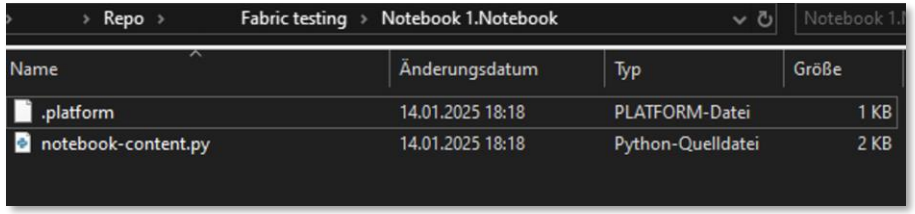
Dev-Prozess 2.2: Lokale Entwicklung mit git

scieneers
DRIVEN BY DATA

- **Ausgangslage**
  - Git-integrierter Fabric Workspace (main branch in Azure DevOps Repository)
  - Man cloned das Repository lokal
- Dateien werden in **gewohnten DevOps Prozessen** bearbeitet
  - Lokale Branches möglich
  - Feature Branch in DevOps pushen und Pull Request in den main branch erstellen
  - Pull requests & Approval Prozesse über DevOps Funktionalitäten abbildbar
- Soll das Feature in den **live Stand** des Fabric Workspace integriert werden:
  - PR in den main branch im DevOps Repo mergen
  - Dann diesen Stand aus Fabric heraus pullen / updaten
  - Weiteres testing etc. in Fabric nachdem Änderungen live sind
- **Nachteil**:
  - Notebooks werden im Repo nicht als .ipynb Datei gespeichert, sondern als reine .py Datei → Development ist sehr viel weniger komfortabel



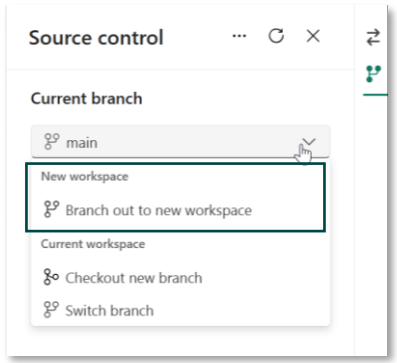


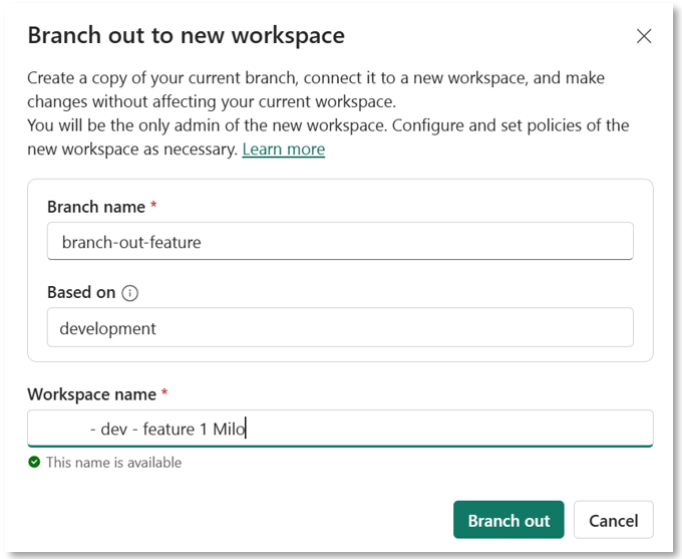


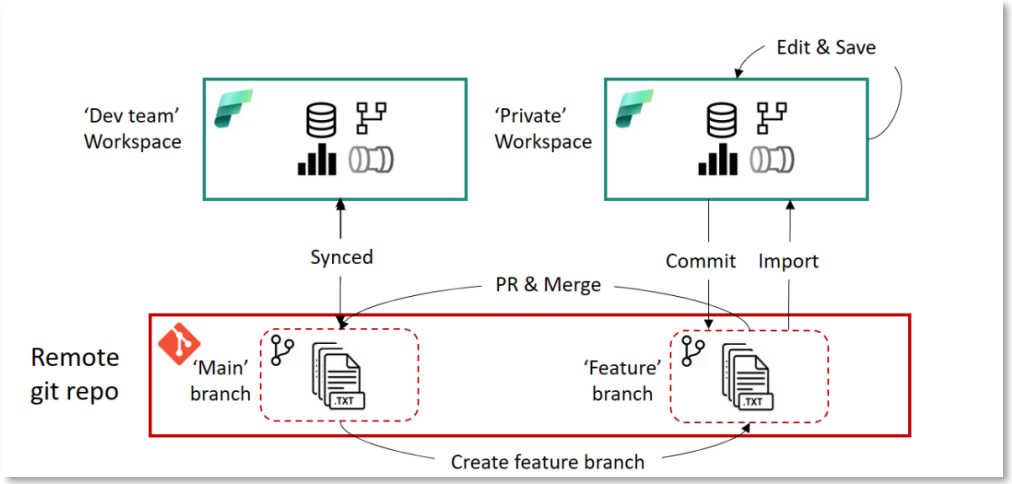
DEMO 2_2

Dev-Prozess 3.2: Feature-Branch workspaces

- Es **wird**
  - ein neuer WS angelegt
  - im Git-Repo ein neuer branch angelegt
  - der neue WS mit dem neuen branch im Git-Repo verbunden
  - der Inhalt des branches in den neuen WS gepulled
- Im neuen WS kann dann **isoliert** an einem Feature gearbeitet werden
  - Notebooks bleiben z.B. mit dem **Lakehouse des Quell-WS** verbunden, da Daten ja nicht git-gesichert sind und somit im neuen Lakehouse auch nicht vorhanden sind
- Pull-requests / merges werden über **Azure DevOps** gelöst
- Der neue Stand wird anschließend aus Fabric heraus **vom Repo gepulled**
  - Das kann **manuell** erfolgen, oder **via API** im **Deployment-Prozess**
- **Probleme**:
  - User müssen die Berechtigung haben WS zu erstellen
  - Feature WS werden nicht automatisch gelöscht







DEMO 3_2

scieneers
DRIVEN BY DATA

Der Deployment Prozess

Möglichkeiten zum Deployment von Fabric Objekten

Deployment Prozesse

scieneers
DRIVEN BY DATA

1. Fabric Deployment Pipelines

2. Azure DevOps Pipelines

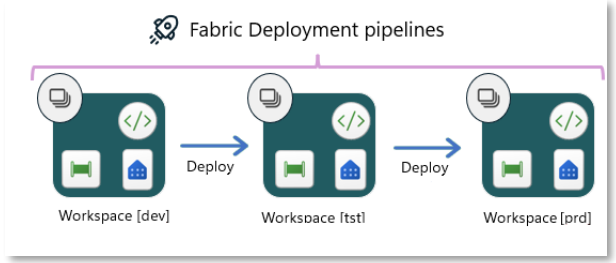
2.1. git-based

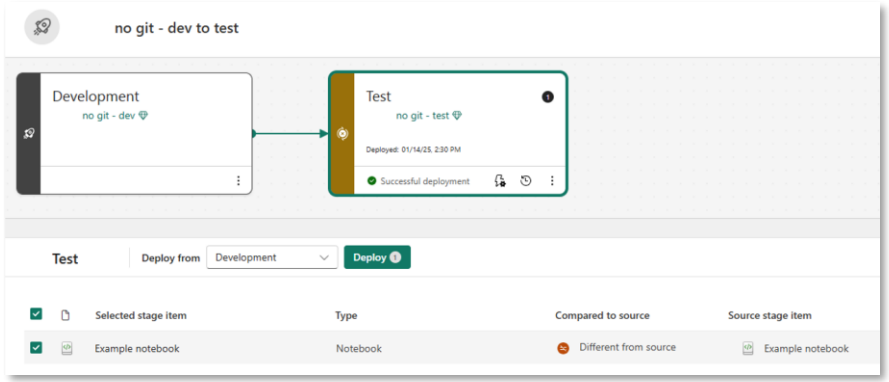
2.2. release pipelines

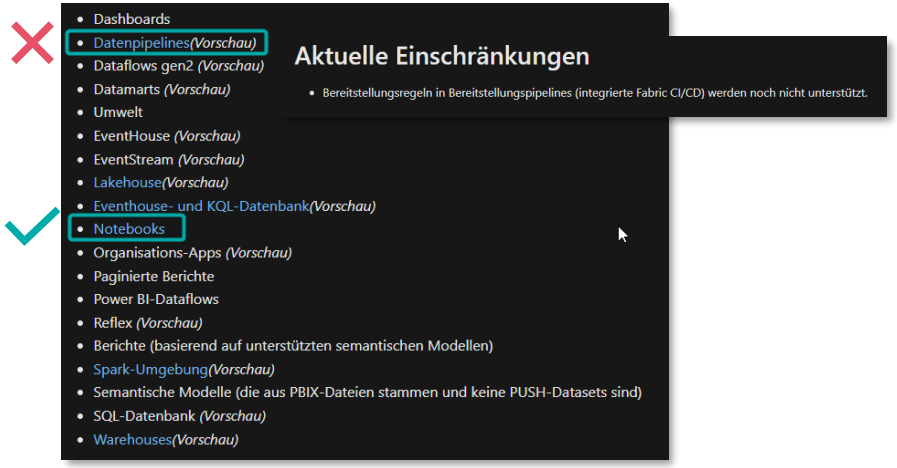
3. Hybride / Kombinierte Ansätze

Deploy-Prozess 1: Fabric Deployment Pipelines

- Keine GIT-Integration **notwendig**, kann aber ergänzend eingesetzt werden
- **Einfachster Deploymentprozess** z.B. für Fachabteilungen
- In Fabric Integriert, **kein weiteres Tool** notwendig
- Es werden noch **nicht alle Fabric Objekte** unterstützt
- Es wird der Stand deployed der im DEV-Workspace **live** ist
- Bis zu **10 Environments** abbildbar
- Es können sog. **Deployment-Rules** definiert werden.
  - Deployment-Rules können je nach zu deployendem Objekt nur bestimmte Aspekte ändern z.B. das default lakehouse bei Notebooks
  - Deployment-Rules für pipelines werden (noch) nicht unterstützt
  - Parameter im Notebook-Code können (noch?) nicht geändert werden
  - Entitäten Namen können nicht geändert werden
  - Changes und Unterschiede zwischen den Environments können nicht ignoriert werden, es gibt aber ein Filter-Funktionalität (z.B. nur geänderte Notebooks anzeigen)



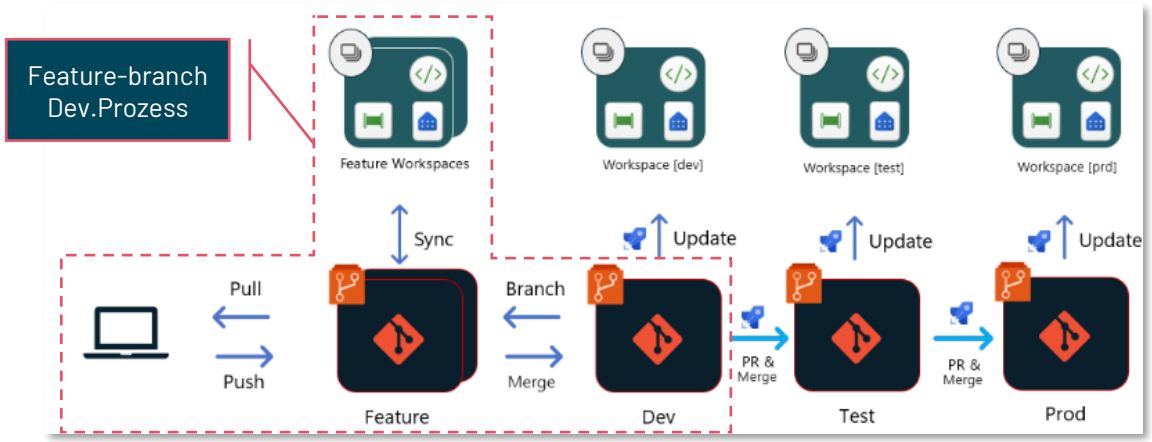


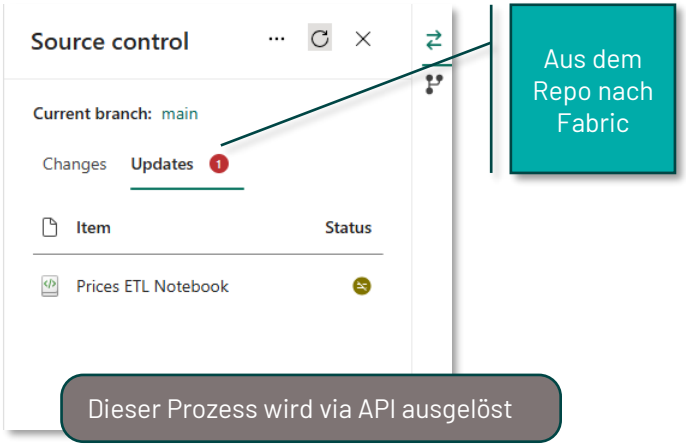


DEMO Deploy_1

Deploy-Prozess 2.1: Azure DevOps Git-based

- Folgt der git-flow branching-strategy
- Beliebiger Dev.Prozess umsetzbar, solange **Dev-WS git-integriert** ist
- Test- und Prod.-WS in Fabric werden an **eigene branches** im gleichen Repo angeschlossen
  - Auf diese sollte nicht direkt committed werden können
  - Änderungen sind nur über pull-requests möglich
- Sobald ein PR gemerged ist, wird in Azure DevOps **automatisiert eine build-pipeline ausgelöst**. Diese…
  - … ersetzt Environment-Parameter
    … dies kann auch über post-deployment API calls direkt auf den Fabric Objekten gelöst werden
  - … kann erweiterte Funktionalitäten (linting, testing, approval etc.) enthalten
  - … speichert Output in einen release-branch
- Ein update im release-branch löst eine **release-pipeline** aus
  - Diese sorgt über die Fabric API dafür, dass der Stand der Objekte im WS basierend auf dem release-branch geupdated wird (pull aus repo!)

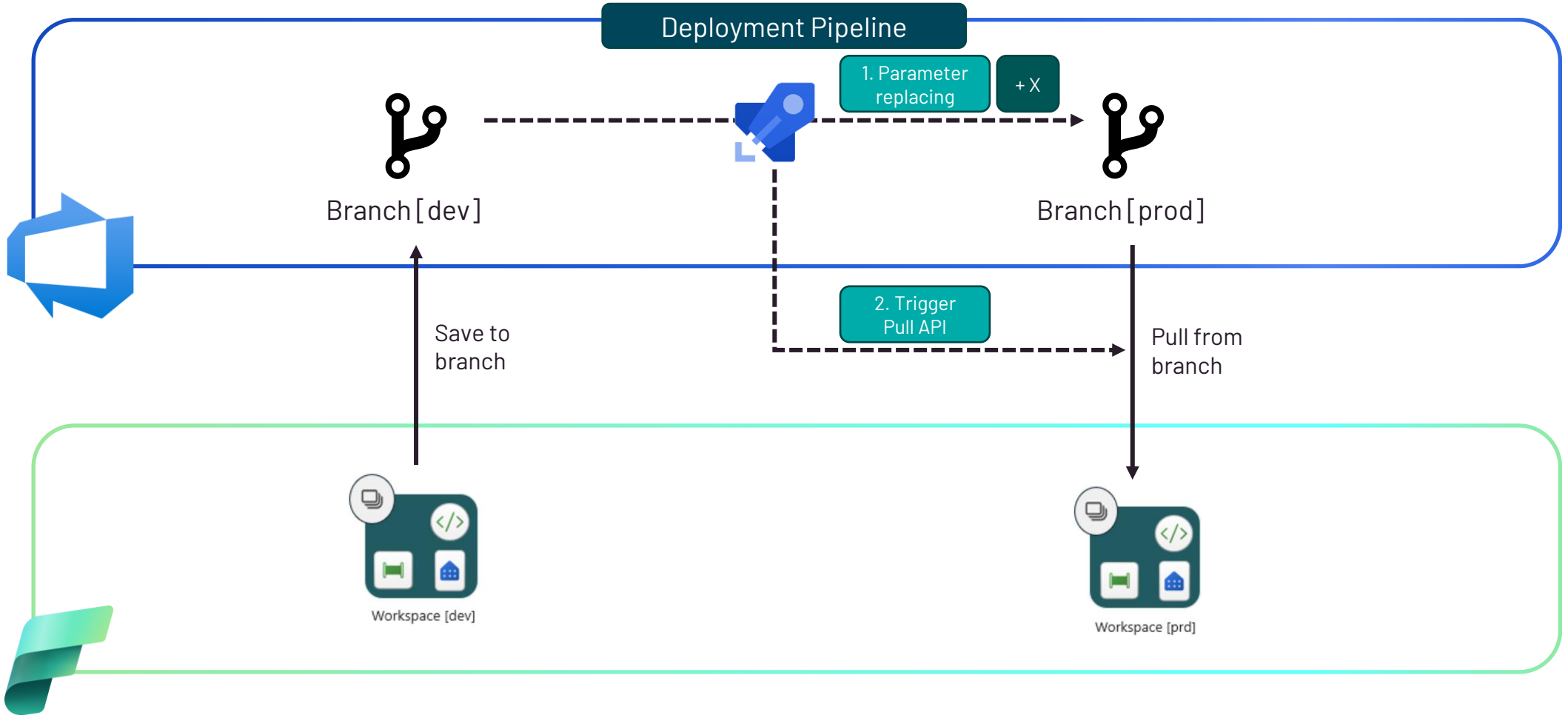




Dieser Prozess wird via API ausgelöst

Aber Achtung: Service Principal bzw. MI Auth wird für diese API noch nicht unterstützt, was Automatisierung unmöglich macht…

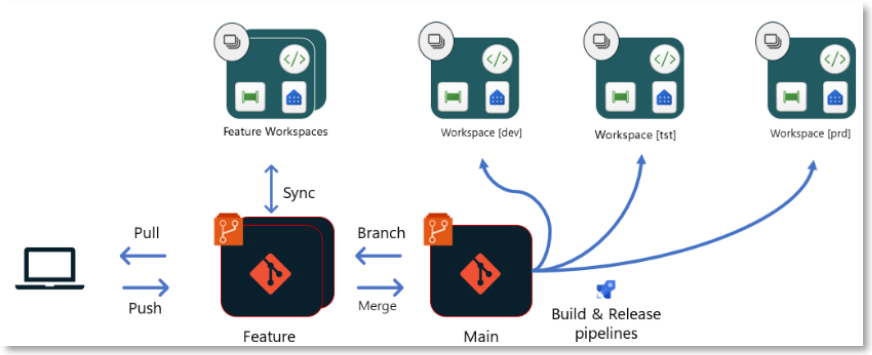
Deploy-Prozess 2.1: Azure DevOps Git-based



DEMO Deploy_2.1

Deploy-Prozess 2.2: DevOps Release Pipelines

- **Kein branch** ist hier direkt mit einem WS verbunden
- Wird ein Feature in den main-branch gemerged, wird eine **Build pipeline** ausgelöst. Diese...
  - ... kann erweiterte Funktionalitäten (linting, testing, approval etc.) enthalten
  - ... speichert Output in einen release-branch oder als Artefakt zwischen
- Eine **release pipelines** nutzt dann diesen Stand, um...
  - ... je nach Environment auf das released werden soll, die Parameter anzupassen
  - ... mittels der Fabric Item API CRUD Operationen auf den Obejekten im jeweiligen WS durchzuführen (push in die WS!)
  - ... verschiedene Libraries können die Fabric Item API abstrahieren, um die Arbeit zu erleichtern



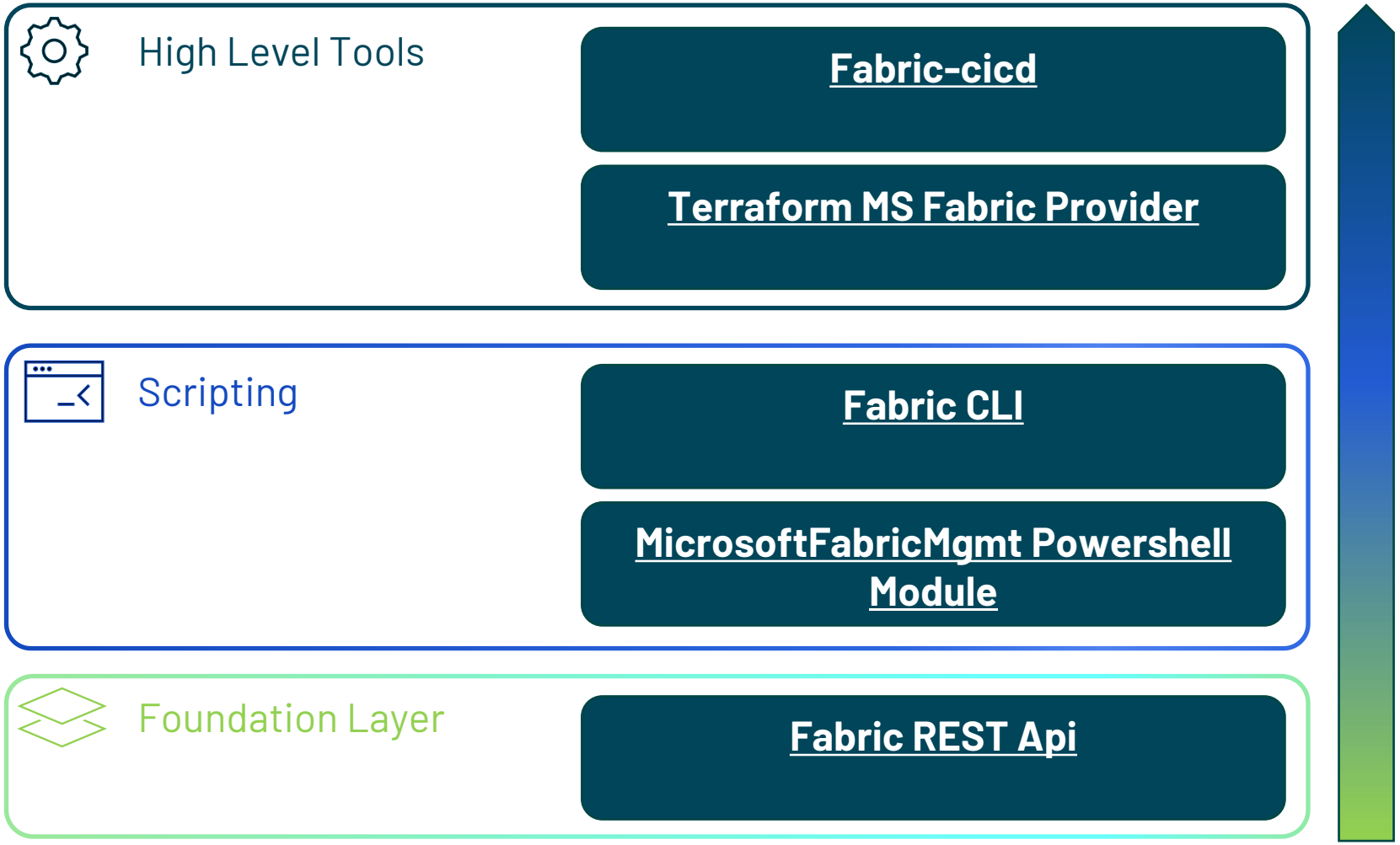

```
HTTP
POST https://api.fabric.microsoft.com/v1/workspaces/{workspaceId}/items
```

z.B. erstelle neues Notebook in diesem WS

```
HTTP
PATCH https://api.fabric.microsoft.com/v1/workspaces/{workspaceId}/items/{itemId}
```

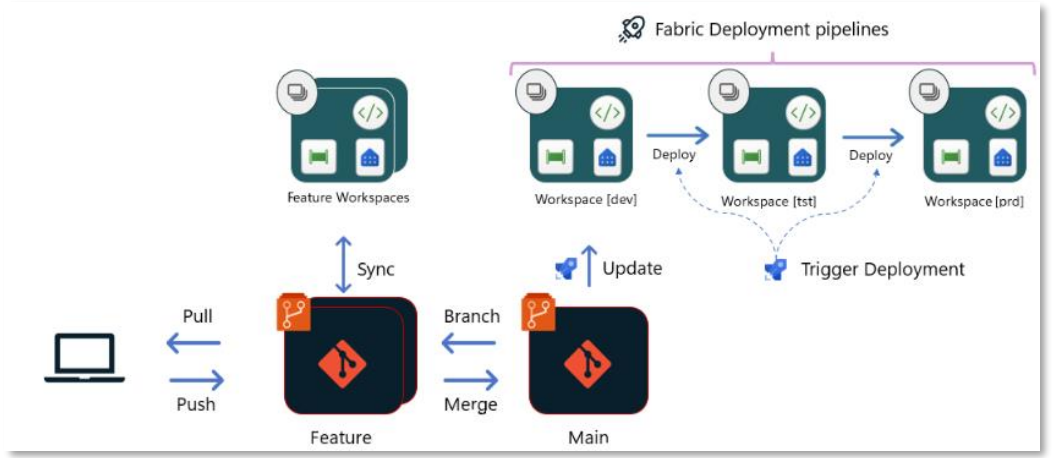
z.B. update das Notebook mit der ID x in diesem WS

Deploy-Prozess 2.2: DevOps Release Pipelines



Deploy-Prozess 3: Kombinierte Lösungen

- Nur der DEV WS ist git-integrated
- Das Deployment findet vom DEV WS aus, über Fabric **Deployment pipelines** statt (analog Deploy-Prozess 1)
- Der Start der Fabric Deployment pipelines kann allerdings über Azure DevOps-trigger und die Deployment pipelines API **automatisiert gestartet** werden
- In die Fabric Deployment pipelines lassen sich i.d.R. **keine weiteren Schritte wie das Ausführen von Notebooks oder das Ansprechen von APIs** etc. einbauen.
- Diese Schritte können aber parallel / sequenziell durch die DevOps Pipeline angestoßen werden



Vor- und Nachteile der einzelnen Ansätze

1. Fabric Deployment	2.1. DevOps git-based	2.2. DevOps Release pipelines	3. Hybrider Ansatz
+ keine git-Erfahrung notwendig + alles in einem Portal	+ Volle Flexibilität + Erlaubt Einbindung weiterer Schritte wie testing etc.	+ Volle Flexibilität + Erlaubt Einbindung weiterer Schritte wie testing etc.	+ Kompromiss aus Flexibilität und einfachem Setup + Es können bedingt weitere Schritte eingebunden werden
– Begrenzte Einbindung von weiteren Schritten wie Code-testing etc. – Abhängigkeit von unterstützten Objekten und dem Umfang der Deployment-Rules pro Objekt – Keine Möglichkeit weitere APIs anzusprechen (z.B. RUN notebook um SQL views upzudaten)	– Erfordert Management verschiedener branches – Management in DevOps, nicht im Fabric Portal – DevOps Lizenzen notwendig – Update bzw. Replace von Env.-Parametern in Ressourcen und Connections aktuell noch "hacky" (Variable library ist aber frisch im preview!)	– Erfordert am meisten Erfahrung mit DevOps Scripting – Ggfs. häufigere Anpassung der Pipelines notwendig – DevOps Lizenzen notwendig – Abhängigkeit von der Fabric Item API / darauf aufbauenden Libraries – Management in DevOps, nicht im Fabric Portal	– DevOps Pipeline wird an zwei verschiedenen Orten gepflegt – DevOps Lizenzen notwendig – Abhängigkeit von unterstützten Objekten und dem Umfang der Deployment-Rules pro Objekt
– **Fachabteilungen** – **Power BI+ Projekte**	– **Professionelle Data Teams**	– **Professionelle Data Teams**	– **Von technisch versierter Fachabteilung bis zentral IT einsetzbar**

scieneers
DRIVEN BY DATA

Gibt es Fragen?

Danke fürs Zuhören!