Datenbankentwicklung und Prototyping in Containern

David Schäfer
HMS Analytical-Software GmbH

Wie lange benötigt man für die Beladung?





3 - 4 min pro Container

viele Nerven ~ 20 bis 40 min

Agenda

- Motivation und Einführung
- Prototyping
- Entwicklung
- Entwicklung im Team & Operationalisierung
- Limitierungen & Fazit



Motivation und Einführung



Microsoft und Linux

Mai 2015: „Microsoft loves Linux“

März 2016: Erste Preview von MS SQL auf Linux

August 2016: WSL 1 wird in Windows 10 bereit gestellt

Oktober 2017: MS SQL Server 2017



Warum? Was ist das Problem?

Bisher

- Datenbank-Software hat ***viele Abhängigkeiten und Konfigurationsmöglichkeiten, z.B. Collation***
- Normale Installation ***benötig seine Zeit***
- ***Mehrere Versionen*** unter einem Host können zu Schwierigkeiten führen



Warum? Flexibilität!

- ***Geschwindigkeit:*** Neue Version innerhalb von Minuten verfügbar
- ***Vielfalt:*** Sämtliche MS SQL Versionen inkl. Cumulative Updates sind seit MS SQL 2017 GA verfügbar: [> 130 Versionen]
- ***Parallelität:*** Mehrere MS SQL Container können ausgeführt werden



Warum?
Vergleichbarkeit zwischen Umgebungen

- Wie hat jeder Entwickler die gleiche MS SQL Version?
- Wie verteilt man statische Daten und/oder eine Start-Datenbank?

→ MS SQL on Linux in Containern

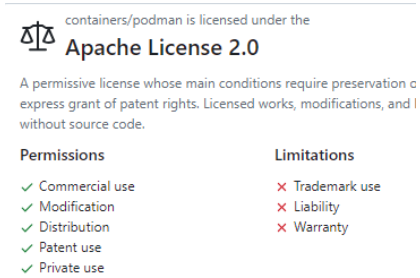


Prototyping



Welche Container Engine?

- Docker vs. Podman?
  - → Beides geht!
- ! Docker Desktop – Lizenzkosten im kommerziellen Umfeld !
- Alternative: Podman
  - Apache 2 Lizenz
  - Befehle analog zu Docker





Umgebungsvariablen für Container – die wichtigsten für den Container

Name	Beschreibung/Mögliche Werte
ACCEPT_EULA	Y
MSSQL_PID	Evaluation, Developer, Express, Web Standard, Enterprise, A product key
MSSQL_SA_PASSWORD	Passwort-Zeichenkette
MSSQL_TCP_PORT	Portnummer
MSSQL_COLLATION	Collation als Zeichenfolge
MSSQL_LCID	Numerische Sprach ID

Configure SQL Server settings with environment variables on Linux

Allgemeine Container Optionen

- **Persistenz:** Volume Mount, z.B.
  -v C:\dbserver\daten:/var/opt/mssql/data
  -v C:\dbserver\transfer:/transfer
- **Port-Mapping:** z.B. -p 1633:1433
- **Daemon-Modus:** -d
- **Benennung des laufenden Containers:** --name sqlpassdemo1

Demo 1 – Starten des Containers

podman run -e "ACCEPT_EULA=Y"
-e "MSSQL_SA_PASSWORD=yourStrong(!)Password"
-e "MSSQL_PID=Evaluation" -p 1633:1433
--name sqlpassdemo1 -d
-v C:\dbserver\daten:/var/opt/mssql/data
-v C:\dbserver\ transfer:/transfer
mcr.microsoft.com/mssql/server:2022-preview-ubuntu-22.04

Verbindung über ***127.0.0.1,1633*** möglich

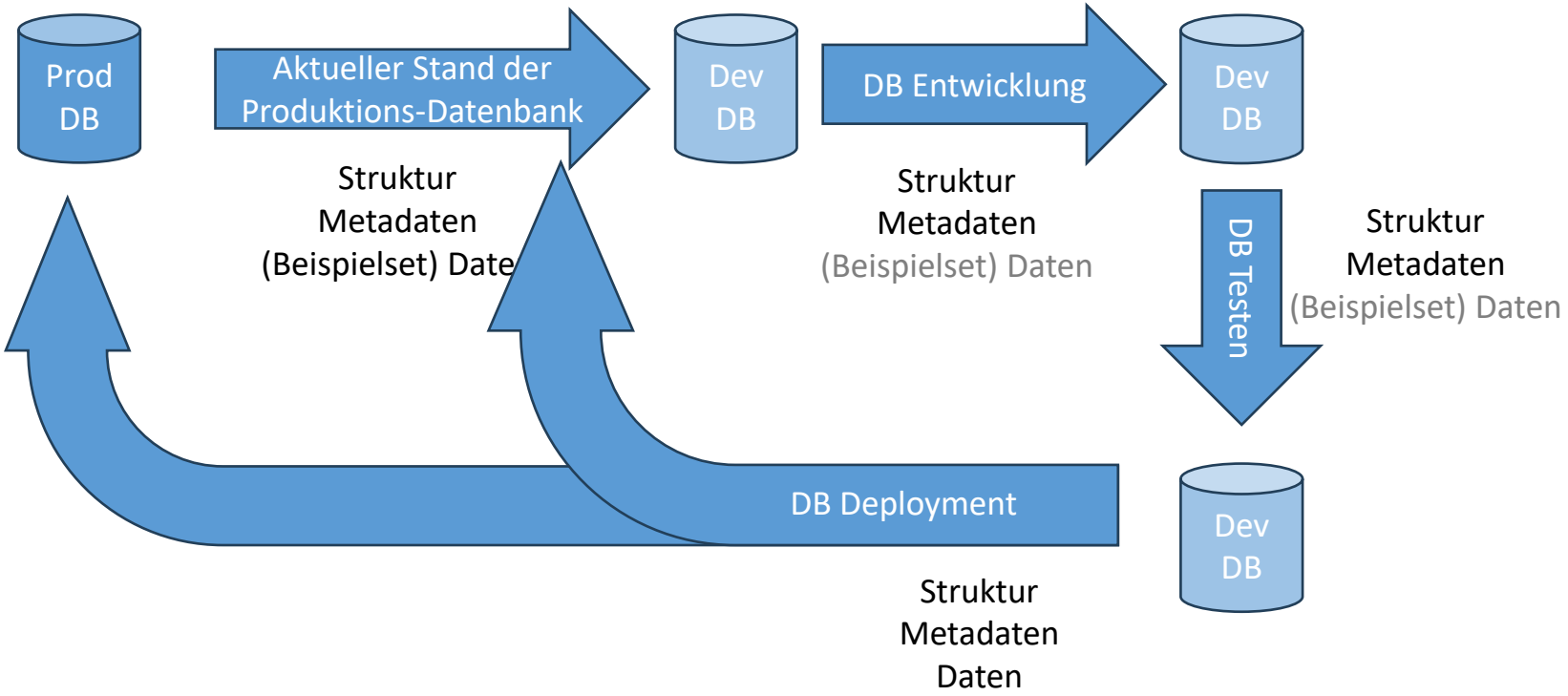
Demo 1 - Ergebnis

- Restore Adventure Works Datenbank
- Ausführen einiger Kommandos

Entwicklung



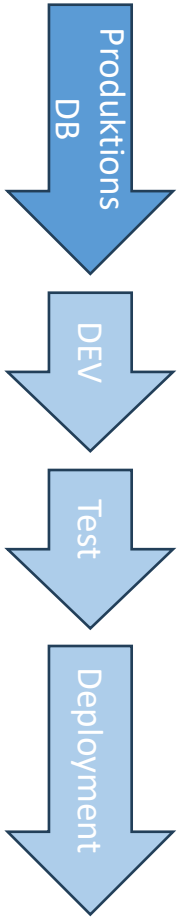
(Vereinfachter) Entwicklungszyklus



Wie bekommt man Daten in Container? Die Klassiker

- **Backup/Restore**
  - Vorteil:
    - 1:1
  - Nachteil:
    - Große Datenmenge
    - Sensible Informationen
    - Andere Konfigurationen?

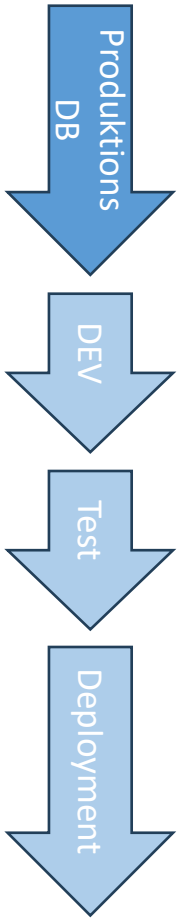
- **Initialisierungsskript mit Daten**
  - Vorteil:
    - Struktur bleibt erhalten
    - Definiertes Set an Daten
  - Nachteil:
    - Datenmenge kann fehlen?
    - Handarbeit bei der Erstellung der Daten



Wie bekommt man Daten in Container? DacPac Tools

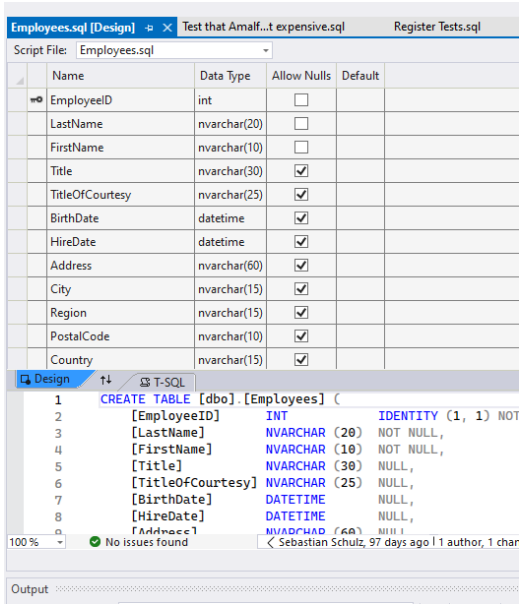
- **BacPac mit sqlpackage export Option erstellen**
  - Vorteil:
    - Moderner Ansatz – state of the art
    - Filter für Daten verfügbar
  - Nachteil:
    - Funktioniert nur für Features, die auch in Azure SQL DBs verfügbar sind

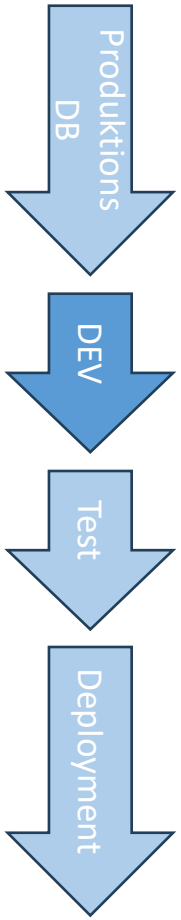
[Weiteführende Informationen]



Datenbank – Entwicklung – DacFx Familie

- Ablauf:
  - Code Entwicklung
  - Erstellung DacPac
  - Deployment DacPac
- Ursprünglich .net Framework basiert
- Lösungen für dot verfügbar

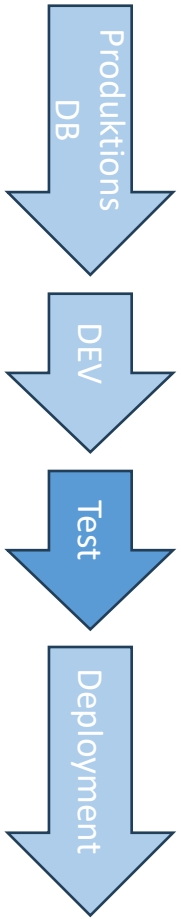




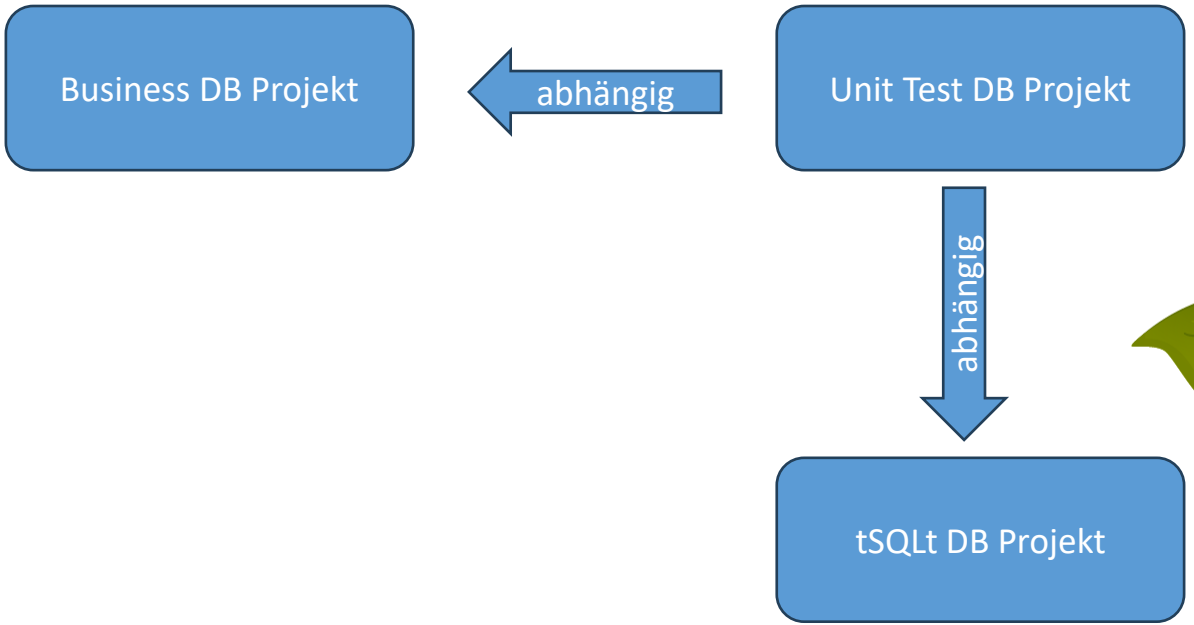
Datenbank – Tests: tSQLt

- Unit Test Framework zum Testen von T-SQL – Code
- Bereitstellung von
  - Assert Funktionen
  - Mock Funktionen
  - Reporting Funktionen
- T-SQL Code in eigenem DB Projekt gliedern
- Deployment über weiteres DacPac

tSQLt
Test Driven Design for SQL Server

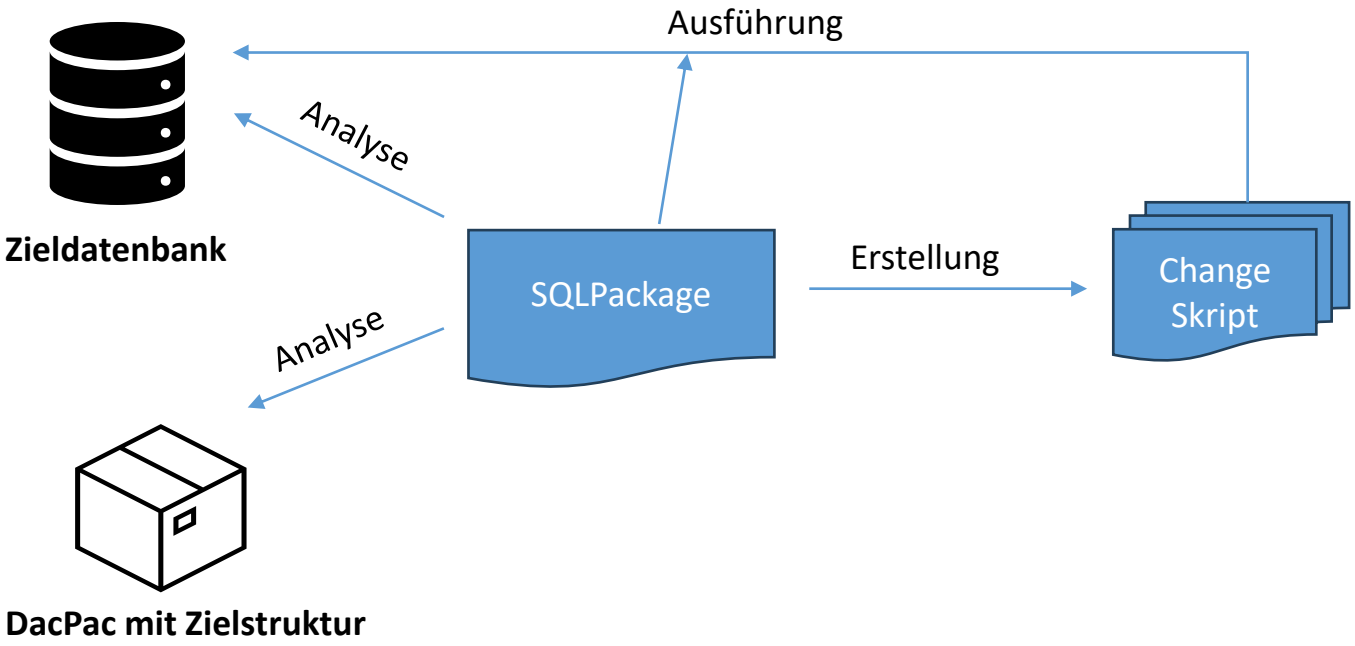


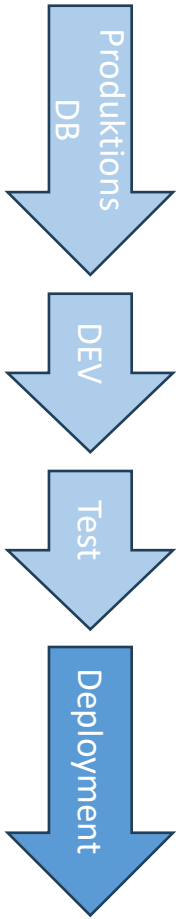
Datenbank Projekt Struktur





Datenbank – Deployment: SQLPackage





Operationalisierung – Bauen mit Dockerfile

- Basis Container
- Umgebungsvariablen
- Operationen, die im Build ausgeführt werden soll
- Kommando beim Start des Images

```
FROM mcr.microsoft.com/mssql/server:2022-latest

ARG MSSQL_COLLATION=SQL_Latin1_General_CP1_CI_AS
ENV ACCEPT_EULA=Y
ENV MSSQL_PID=Developer

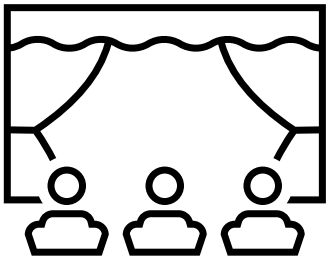

## SQL Package
RUN mkdir ./tools
RUN wget https://aka.ms/sqlpackage-linux -O ./tools/sqlpackage-linux.zip
RUN unzip ./tools/sqlpackage-linux -d ./tools/sqlpackage && \
    chmod a+x ./tools/sqlpackage/sqlpackage

# Start server and create Database
RUN (/opt/mssql/bin/sqlservr & ) | grep -q "Service Broker manager has starte
    ./tools/sqlpackage/sqlpackage /Action:Import /TargetConnectionString:"Serv
    ./tools/sqlpackage/sqlpackage /Action:Publish /TargetConnectionString:"Ser
    /opt/mssql-tools/bin/sqlcmd -S localhost -U ${SA_USER} -P ${SA_PASSWORD} -
    ./tools/sqlpackage/sqlpackage /Action:Publish /TargetConnectionString:"Ser
    ./tools/sqlpackage/sqlpackage /Action:Publish /TargetConnectionString:"Ser

# https://stackoverflow.com/questions/31870222/how-can-i-keep-a-container-runn
CMD (/opt/mssql/bin/sqlservr &) && \
```

DEMO 2

- IDE Unterstützung
- Dockerfile
- PowerShell Skript

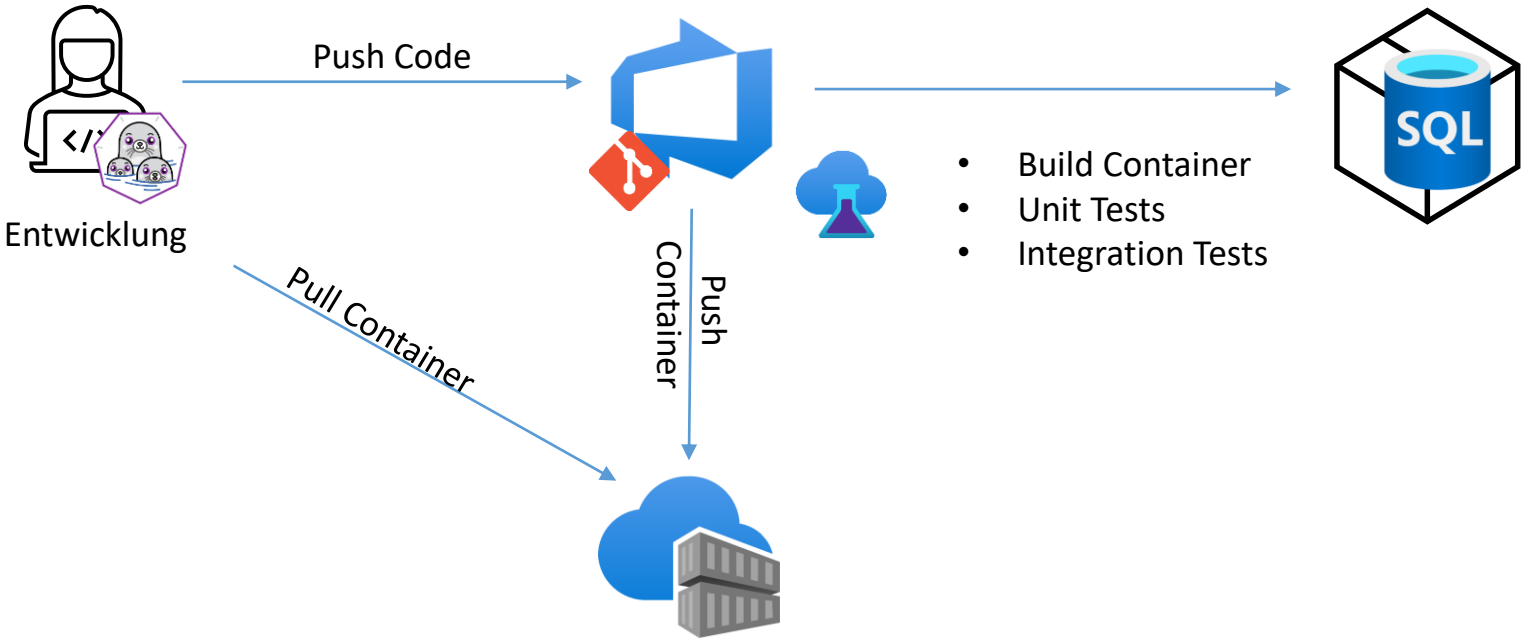




Entwicklung im Team & Operationalisierung

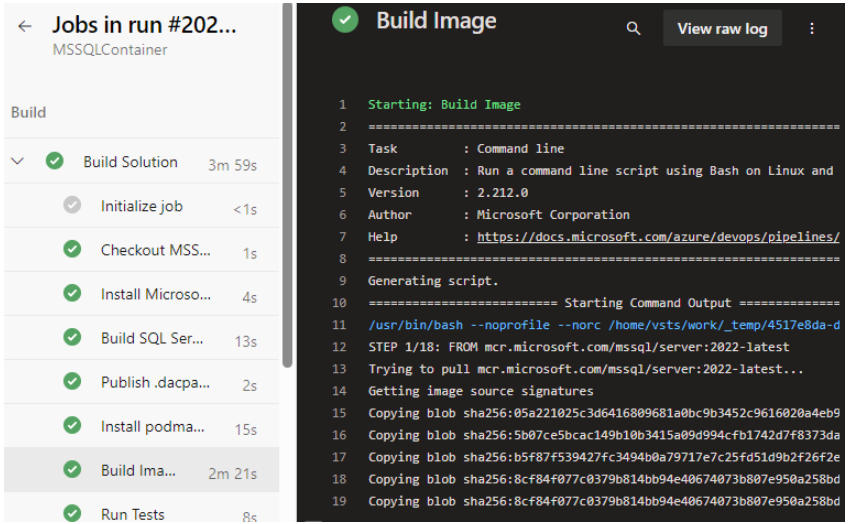


Entwicklungsworkflow



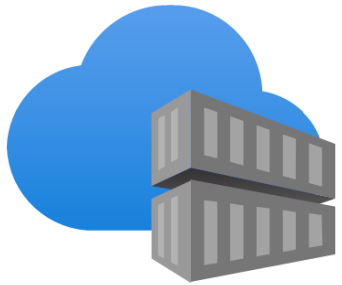
Build Pipeline

- Azure DevOps zum
  - Bauen des DacPacs
  - Erstellen des Containers
  - Ausführen der Tests
  - Veröffentlichen des Containers in einer Container Registry
- Jede andere CI/CD Technologie denkbar, z.B. GitHub Actions



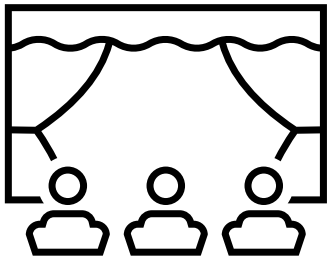
Wie teilt man Container? Container Registry

- Analog zu DockerHub – private Container Registry
- Mögliches Produkt: Azure Container Registry
- Entwickler können von dort Container holen



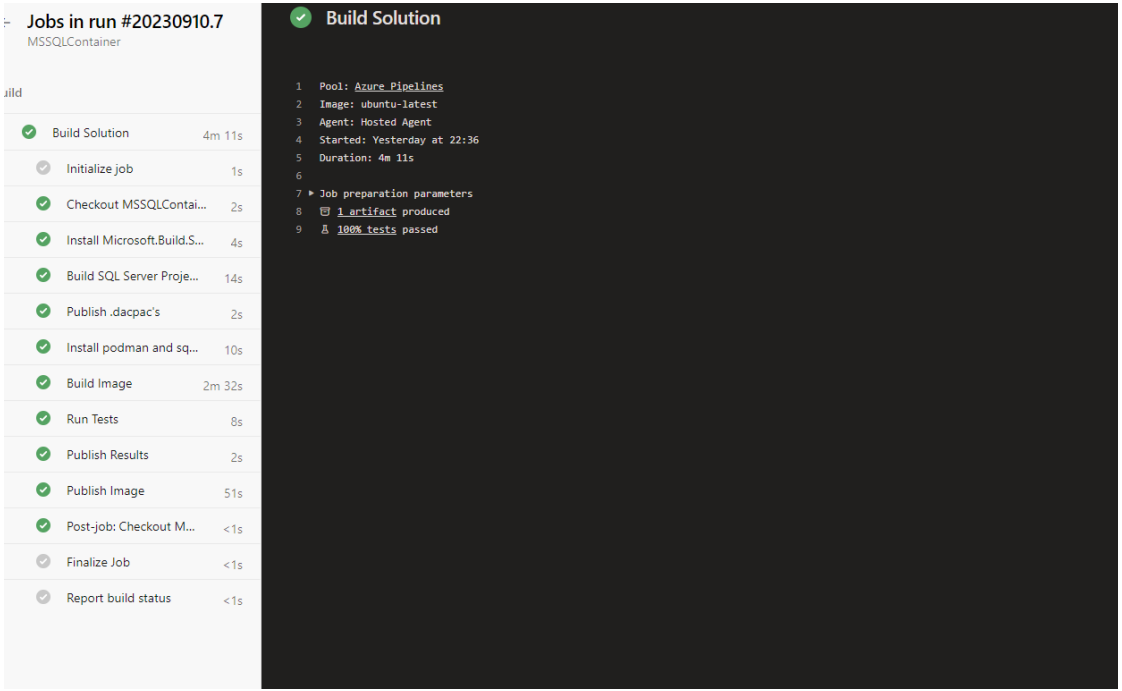
DEMO 3

- Azure DevOps Build Pipeline
- Container Registry



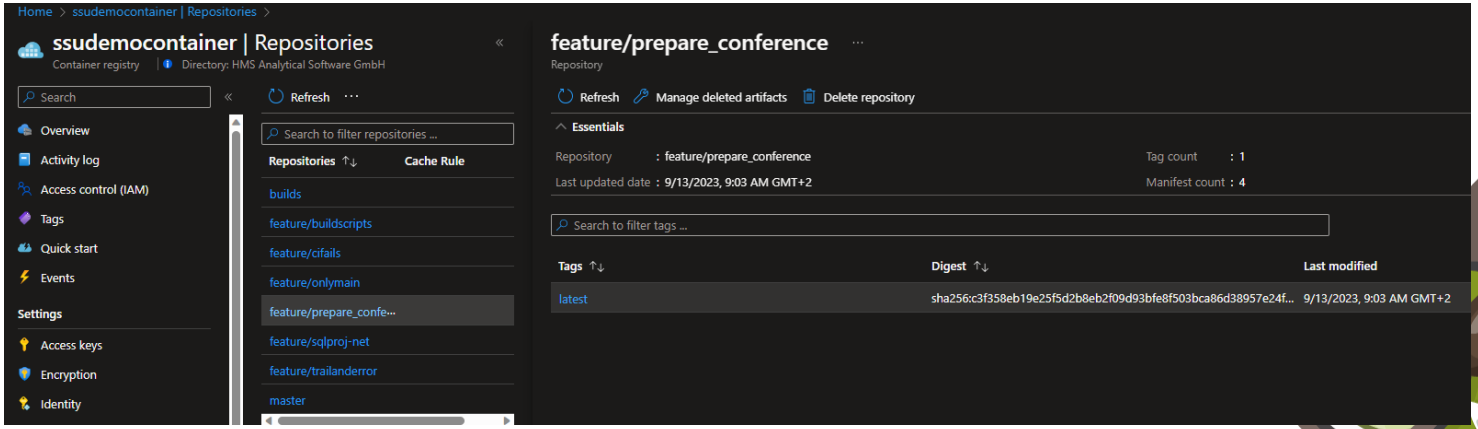


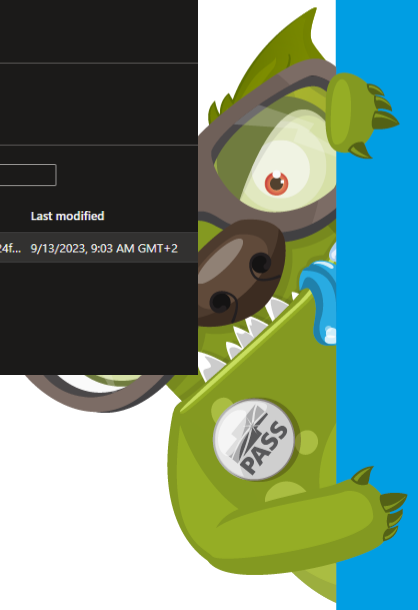
DEMO 3 – DevOps Pipeline





DEMO 3 – Container Registry





Limitierungen & Fazit



Limitierungen

- Nur Database Engine verfügbar
- Operationen, die auf das darunter liegende Dateisystem oder Host System zugreifen sind nicht verfügbar, z.B.
  - xp_cmdshell
  - FileTable & FILESTREAM
- „Linked Server“ gehen nur zu anderen SQL Serven
- …



Fazit

- Mehr Flexibilität
- Mehr Vergleichbarkeit
- Geringere Zeit für Time-to-Develop

- Nicht alle Funktionen werden unterstützt
  Braucht man die?
  In Azure SQL gibt es sie auch nicht…



Feedback!

Vielen Dank für
Euer Feedback!



THANK YOU TO OUR SPONSORS

PASS DEUTSCHLAND e.V.
Microsoft Data Platform Community

pmOne

SOLARWINDS

smart casual datadesign

Mainzer Datenfabrik

b.telligent
smart data. smart decisions.

HMS analytical software

ORAYLIS
turn your data into value

TIMEXTENDER

Microsoft

ITGAIN

SOLISYON
www.solisyon.de

GDS BUSINESS INTELLIGENCE GmbH

Weiterführende Links

- Editions and supported features of SQL Server 2022 (16.x) on Linux
- Configure SQL Server settings with environment variables on Linux
- Docker Hub MS SQL

- Demo Code on GitHub



Thank you very much for your attention.

Vielen Dank für Eure Aufmerksamkeit.

David Schäfer

david.schaefer@analytical-software.de

