



BASTA!

by entwickler.de

Automatisiertes Datenbank-Deployment im DevOps-Prozess

Marc Müller
Principal Consultant



marc.mueller@4tecture.ch
@muellermarc
www.4tecture.ch

4tecture[®]
empower your software solutions



About me:

Marc Müller
Principal Consultant
@muellermarc



4tecture[®]
empower your software solutions

Our Products:

Multi-Tenant OpenID Connect Identity Provider



www.proauth.net

Enterprise Application Framework for .NET



www.reafx.net

Slide Download



<https://www.4tecture.ch/events/basta24dbdeployment>

Agenda

- Intro
- Database as Code with SSDT
- Schema Migrations
- Migration Scenarios
- DevOps Process
- Deployment Security Considerations
- Containers
- Autonomous Deployment





Chapter 1/8

Intro

4tecture®
empower your software solutions

Write Code



Customer use the code



Write Code



Customer use the code



100 deployments
per day!



Being ready for 100 deployments a day

Fully automated process

- Build Automation
- Deployment Automation
- Test Automation

Small and frequent releases

- Reduce Complexity
- Daily Business

There is no place like production

- Testing in Production
- Zero Downtime
- Feature Flags

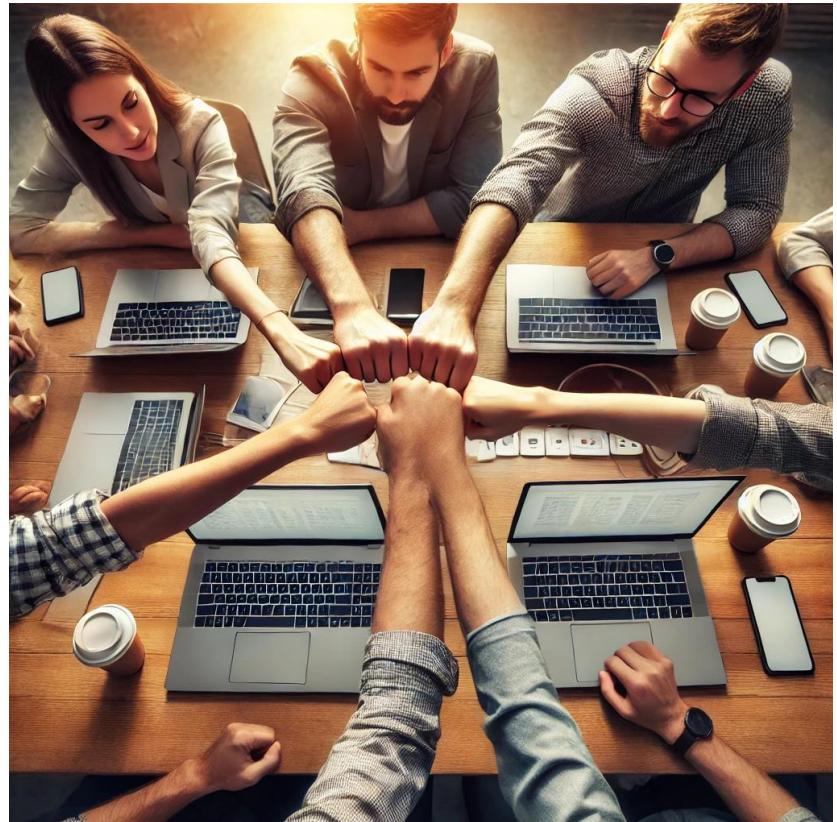


What about the DB development?

Database development
is fully integrated

No manual schema
changes

Automated deployment
of schema changes



Reality?

Different Teams

DB development not integrated / manual

Schema mismatch between dev and prod



Challenges

DB schema and code change
belong together

Dry-Run on (production) data

Data Migrations / Reference
Data

Zero Downtime Deployment



A close-up, low-angle shot of several rowers in a racing shell. Their legs and the oars are visible, with the water spray from their strokes creating a misty, dynamic background.

Chapter 2/8

Database as Code with SSDT

SSDT - Characteristic

- SSDT Project Type for relational Database Development
- Integrated in Visual Studio IDE
- Others: SSMS, Redgate, DDL/DML Scripts
- SSDT Advantages:

IDE
MSBuild

IntelliSense
Validation

Code Base
Consistency

Design
Compare

CI
CD

SSDT - Characteristic

- Officially Supported since VS 2015
- 1:1 Database Representation
- SSDT Deployment / Prerequisites:

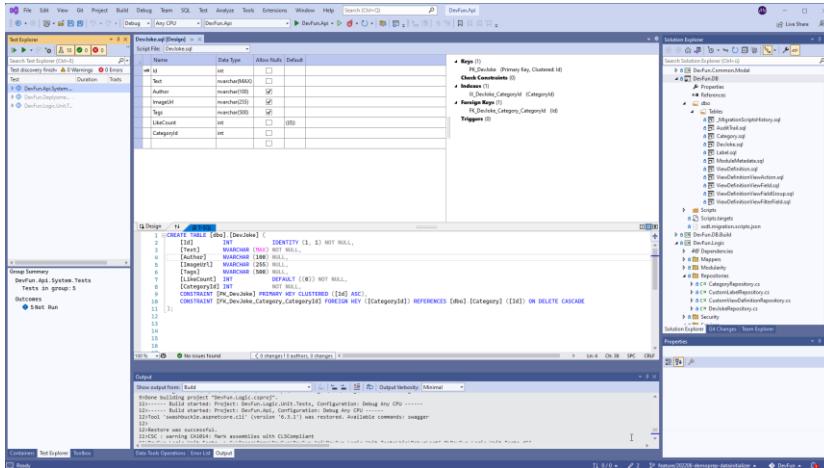
DB Schema Migrations (Static & Dynamic SQL)

Single Pre- and Post Script Logic

Microsoft.Data.Tools.Msbuild
(NuGet)

SSDT - Features

- Build time validation / IntelliSense Support

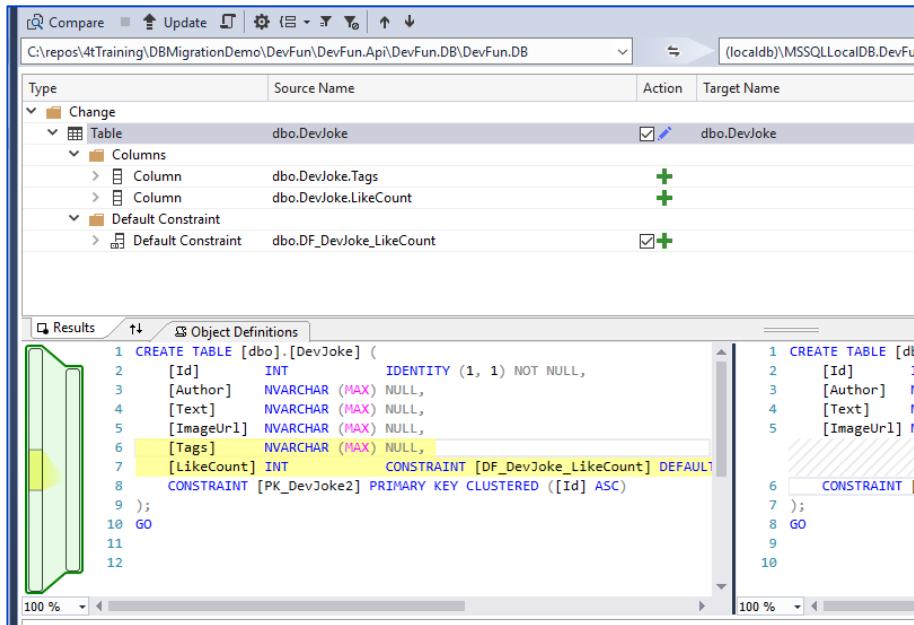


- Bidirectional Scheme Comparison
- Bidirectional Scheme Synchronization
- Versioned migration and schemes artifact
- Code-base integration / Change tracking

(SSDT ⇄ DB)
(SSDT ⇄ DB)
(DACPAC)
(GIT)

Schema Compare

- Schema Compare
- Local Development
- Bidirectional Sync
- Choose your favorite IDE
- Prevent data loss: rename in SSDT



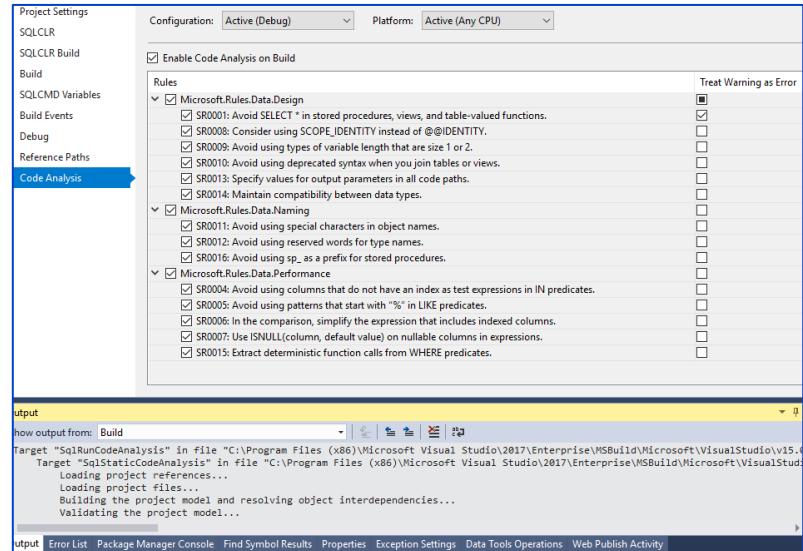
The screenshot shows the SSDT Schema Compare tool comparing two database schemas. The top pane displays a tree view of changes, with the 'Change' node expanded to show 'Table' (dbo.DevJoke), 'Columns' (Column: dbo.DevJoke.Tags, Column: dbo.DevJoke.LikeCount), and 'Default Constraint' (Default Constraint: dbo.DF_DevJoke_LikeCount). The bottom pane shows the 'Results' tab with the 'Object Definitions' tab selected, displaying the CREATE TABLE script for the dbo.DevJoke table. The 'Tags' column is highlighted in yellow, indicating it is a changed column. The right side of the results pane shows the original and target code for comparison.

```
1 CREATE TABLE [dbo].[DevJoke] (
2     [Id] INT IDENTITY (1, 1) NOT NULL,
3     [Author] NVARCHAR (MAX) NULL,
4     [Text] NVARCHAR (MAX) NULL,
5     [ImageUrl] NVARCHAR (MAX) NULL,
6     [Tags] NVARCHAR (MAX) NULL,
7     [LikeCount] INT CONSTRAINT [DF_DevJoke_LikeCount] DEFAULT(0),
8     CONSTRAINT [PK_DevJoke2] PRIMARY KEY CLUSTERED ([Id] ASC)
9 );
10 GO
```

```
1 CREATE TABLE [dbo].[DevJoke] (
2     [Id] INT IDENTITY (1, 1) NOT NULL,
3     [Author] NVARCHAR (MAX) NULL,
4     [Text] NVARCHAR (MAX) NULL,
5     [ImageUrl] NVARCHAR (MAX) NULL,
6     [Tags] NVARCHAR (MAX) NULL,
7     [LikeCount] INT CONSTRAINT [DF_DevJoke_LikeCount] DEFAULT(0),
8     CONSTRAINT [PK_DevJoke2] PRIMARY KEY CLUSTERED ([Id] ASC)
9 );
10 GO
```

Code Analysis

- Standardized Design Patterns
- Code Quality
- Reduce Code Smells
- Supports Static-/ and Dynamic SQL
- Tables, SP, UDDT, Views...



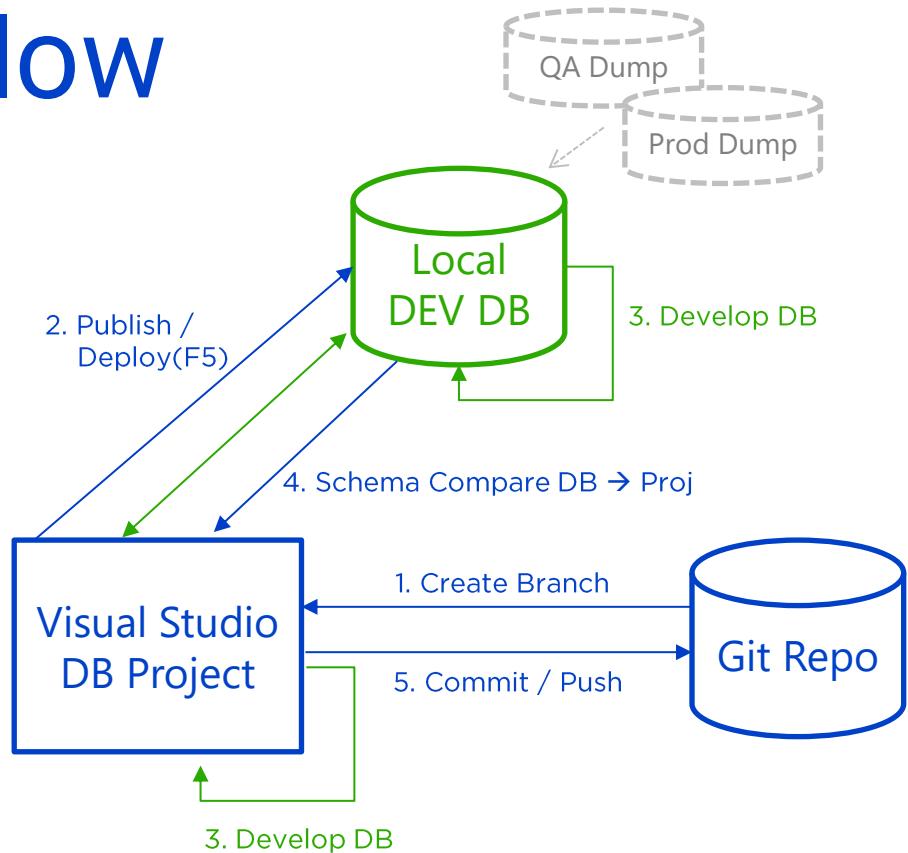
Demo



SSDT in Visual Studio

Developer Workflow

1. Create a Feature Branch from Development
2. Publish/Deploy (F5) Database Project
3. Develop Database changes (Renames have to performed in SSDT)
4. Perform a Schema Compare from DB to Database Project, Sync.
5. Commit > PR > Review



Developer Workflow - Details

Action for every branch change

- After a branch is updated / created we need to set the database to the correct state
- Publish / Deploy (F5) the database project
 - Publish / Deploy (F5) not Schema Compare
 - Pre-, Post- and Reference-Data Scripts are only executed during publish / deploy
 - Use a predefined Publish Profile (in source code) / make sure debug settings meet the production deployment plan
- **Publish the database every time you**
 - Create a new feature branch
 - Update your current branch from a remote (Pull)
 - Switch between branches



Chapter 3/8

Schema Migrations

SSDT is nice, but...

SSDT supports basic script extensibility

- Single Pre-Script
- Single Post-Script

Enterprise-grade migrations imply complexity

- Extended Script Management is needed
- State Tracking of Custom Migrations
- «DB Version» Tracking

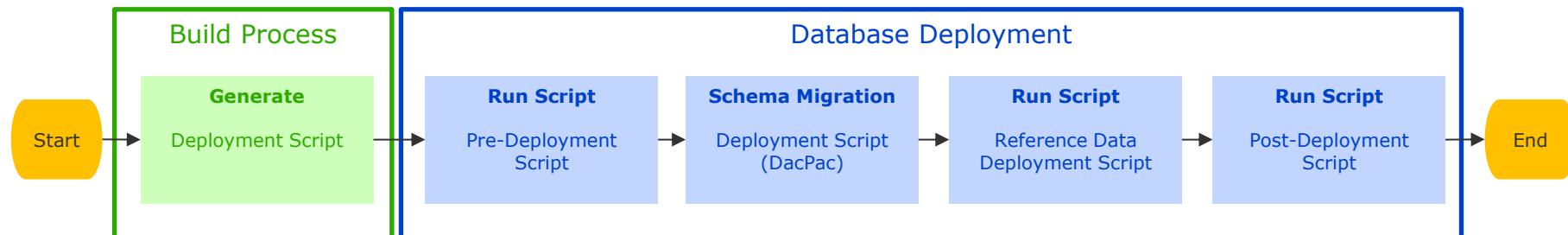
SSDT can easily be extended....

Concept

- Logical separation for Migration Scripts
- Traceability (dev-phase)
- Automatic migration management
- Managed Pre-/Post Scripts
- Managed Reference Data Scripts
- Error- and Transaction Handling
- Pre-Validation / Syntax Check
- History Management

Deployment Process Flow

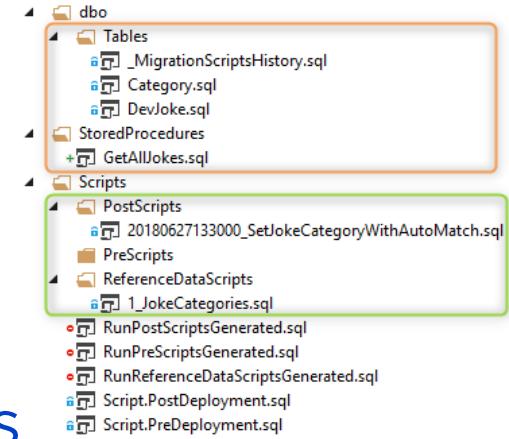
1. Pre-Deployment (*undefined DB-state*)
2. SSDT Schema Deployment
3. Reference Data Deployment
4. Post-Deployment (*Defined schema*)



Responsibilities

Developer / DBA

- Manage Pre-, Post and Reference Data Scripts
- Manage declarative schema changes

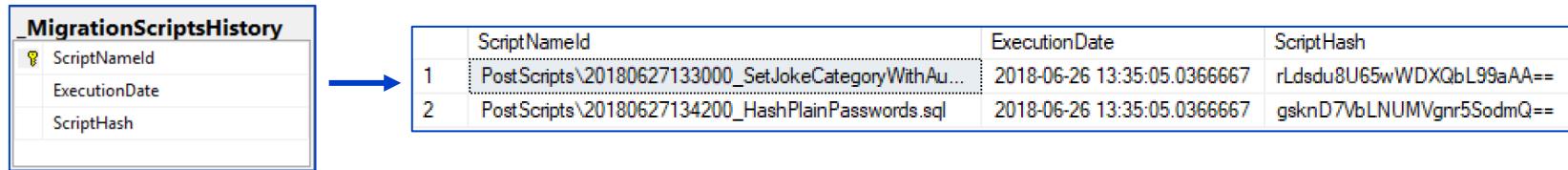


Migration Engine (SSDT / Extension)

- Manage, validate - transactional Migration Scripts
- Extensible, configurable module
- Serves as self-managed Blackbox

Migration History

- Prevents duplicate runs on target database
- Migration scripts kept simple - less complexity

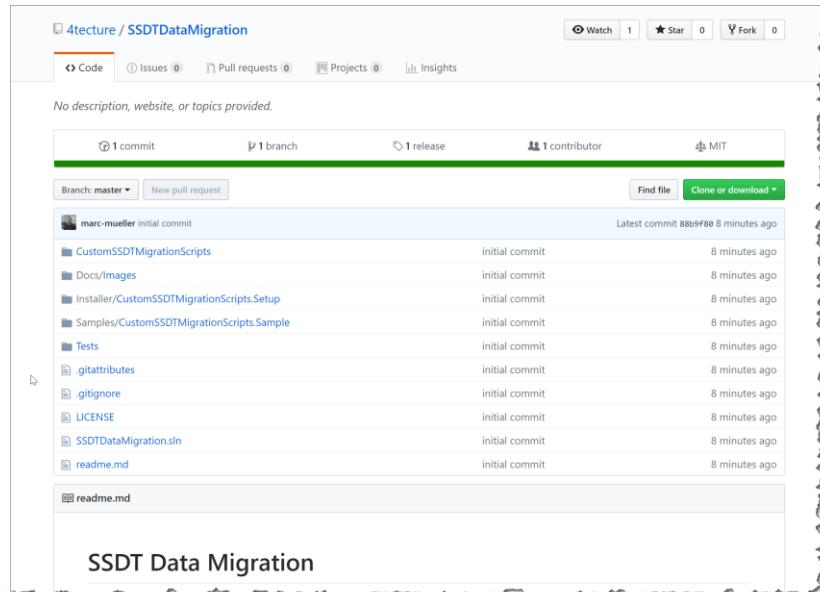


- Process specific meta-configuration
 - Traceability (Execution Date)
 - Predefined execution order based on patterns
 - Managed script execution and change detection
 - Versioning

Our Learnings combined...

We created an open-source extension providing a simple starting point for your projects.

<https://github.com/4tecture/SSDTDataMigration>



The screenshot shows the GitHub repository page for 'SSDTDataMigration'. The repository has 1 commit, 1 branch, 1 release, and 1 contributor. The license is MIT. There are no issues, pull requests, or projects. The repository description is empty. The code tab is selected, showing a single commit from 'marc-mueller' titled 'initial commit' made 8 minutes ago. The repository structure includes 'CustomSSDTMigrationScripts', 'Docs/Images', 'Installer/CustomSSDTMigrationScripts.Setup', 'Samples/CustomSSDTMigrationScripts.Sample', 'Tests', and several configuration files like '.gitattributes', '.gitignore', 'LICENSE', 'SSDTDataMigration.sln', and 'readme.md'. The README file contains the text 'SSDT Data Migration'.

Features

- Configurable Setup and Naming Convention
- Logging / Full Transactional Scripts
- Custom execution filters
- Fully configurable Extension

```
{  
  "PreScripts": {  
    "ScriptBaseDirectory": ".\\Scripts\\PreScripts",  
    "ScriptNamePattern": "^(\d{14})_(.*).sql",  
    "ScriptRecursiveSearch": true,  
    "GeneratedScriptPath": ".\\Scripts\\RunPreScriptsGenerated.sql",  
    "ExecutionFilterMode": "days", // "all", "count", "days", "date"  
    "ExecutionFilterValue": "100",  
  
    "TreatScriptNamePatternMismatchAsError": true,  
    "TreatHashMismatchAsError": true  
  },
```

A dynamic photograph showing a team of rowers from behind, their bodies leaning forward in a powerful stroke. The water is splashing around the oars, creating a sense of motion. The rowers are wearing blue and red athletic gear.

Chapter 4/8

Migration Scenarios

Migration Scenarios

- Add new nullable field to table
- Add new not nullable field to table
- Delete existing field
- Change data type (float to decimal)
- Change existing data type from nullable to not nullable
- Change existing stored procedure
- Add new stored procedure
- Add new function
- Add user defined type (UDT)
- Change existing stored procedure

Responsibility Matrix

Scenario	Pre-Scripts	SSDT Engine	Reference Data Scripts	Post-Scripts
Add (not) nullable field		Supported		
Change to not nullable	Value Migration (optional)	Supported (Smart Defaults)		
Delete field	Data Migration (optional)	Supported		Data Migration (optional)
Change data type (float to dec)	Data Migration (Custom Rounding)	Supported		
Add stored procedure (SP)		Supported		
Change SP		Supported		
Delete objects	Data Migration (optional)	Supported (Enable Data Loss)		
Rename objects		Supported		
Add user defined type (UDT)		Supported		
Add business reference data			Use merge scripts	

Reference Data

- Data needed by application logic
 - Extendable Business Logic
 - Selections
 - Configurations
 - Translations
- Should be consistent with application binaries
- Mainly Merge-Statements

```
1 --MERGE generated by 'sp_generate_merge' stored procedure, Version 0.93
2 --Originally by Vyas (http://vyaskn.tripod.com): sp_generate_inserts (build 22)
3 --Adapted for SQL Server 2008/2012 by Daniel Nolan (http://danere.com)
4
5 SET NOCOUNT ON
6
7 MERGE INTO [dbo].[PaymentMethod] AS Target
8 USING (VALUES
9   ('N'creditcard','N'The credit card payment method','N'Credit Card')
10  ,(N'debitcard','N'The debit card payment method','N'Debit Card')
11 ) AS Source ([Identifier],[Description],[Name])
12 ON (Target.[Identifier] = Source.[Identifier])
13 WHEN MATCHED AND (
14   NULLIF(Source.[Description], Target.[Description]) IS NOT NULL OR NULLIF(Target.[Description], Source.
15   [Name]) IS NOT NULL OR NULLIF(Target.[Name], Source.[Name]) IS NOT NULL
16 ) UPDATE SET
17   [Description] = Source.[Description],
18   [Name] = Source.[Name]
19 WHEN NOT MATCHED BY TARGET THEN
20   INSERT([Identifier],[Description],[Name])
21   VALUES(Source.[Identifier],Source.[Description],Source.[Name])
22 WHEN NOT MATCHED BY SOURCE THEN
23   DELETE
24 ;
25
26 DECLARE @mergeError int
27 , @mergeCount int
28 SELECT @mergeError = @@ERROR, @mergeCount = @@ROWCOUNT
29 IF @mergeError != 0
30 BEGIN
31 PRINT 'ERROR OCCURRED IN MERGE FOR [dbo].[PaymentMethod]. Rows affected: ' + CAST(@mergeCount AS VARCHAR(100))
32 END
33 ELSE
34 BEGIN
35 PRINT '[dbo].[PaymentMethod] rows affected by MERGE: ' + CAST(@mergeCount AS VARCHAR(100));
36 END
37
```

A close-up, low-angle shot of several rowers in a racing shell. Their hands are gripping yellow and black oars, which are angled downwards. The water is visible at the bottom, creating a sense of motion. The rowers are wearing blue and red athletic gear.

Chapter 5/8

DevOps Process

Challenges

Downtime

Roll Forward / Rollback

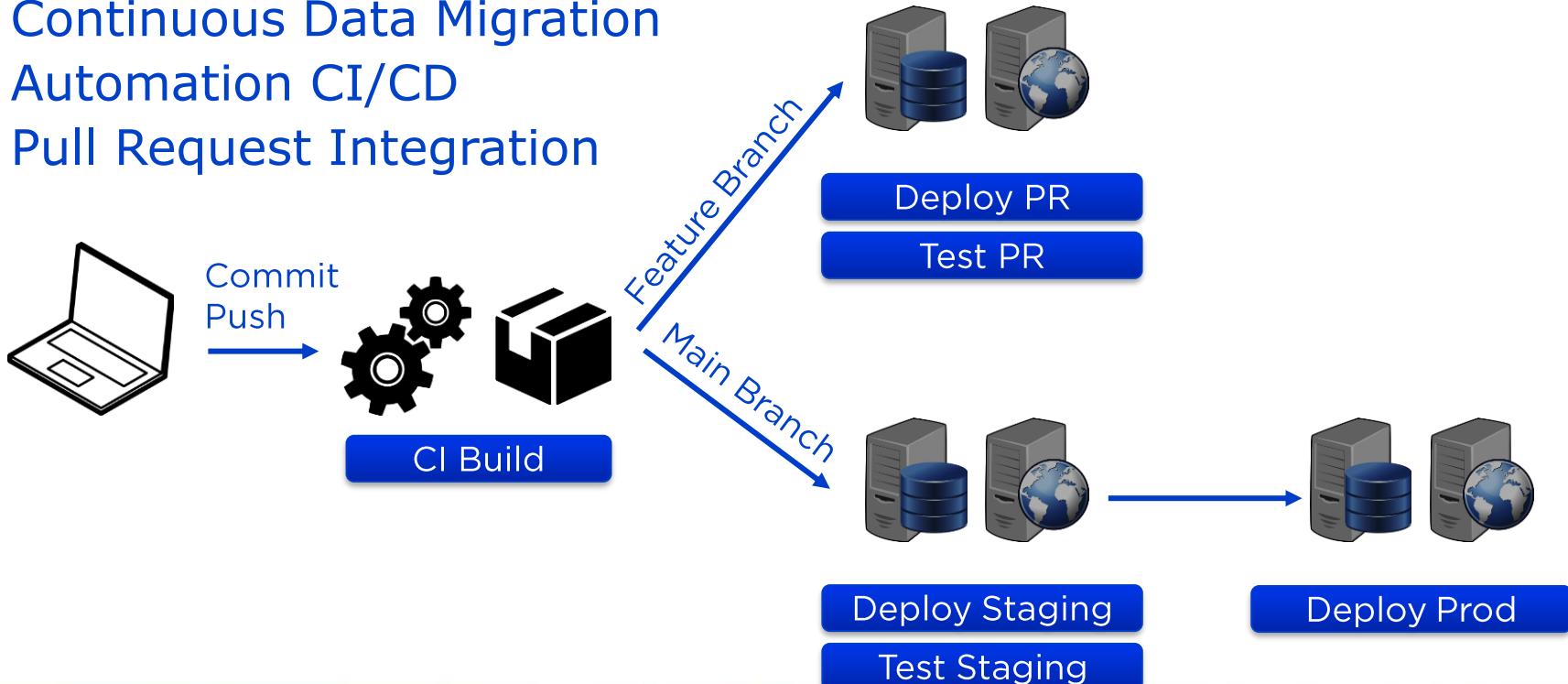
Non-Breaking Changes / Breaking Changes

Deployment Order: DB First / Code First

Handling multiple Schema Versions

DevOps Process Integration

- DevOps Process Integration
- Continuous Data Migration
- Automation CI/CD
- Pull Request Integration



Visual Studio Integration

- Design



- Validate



- Build

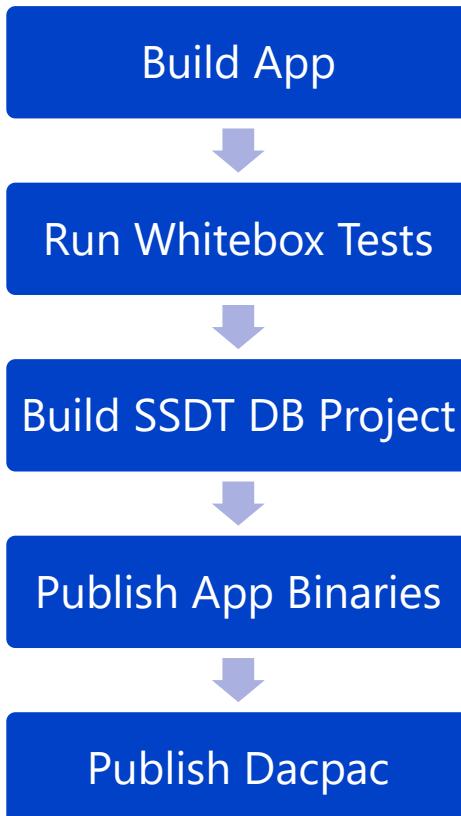


- Deploy
(local dev)

The database SSDTDataMigrationPrototype already exists. Pre- and post deployment scripts will be executed.
The following operation was generated from a refactoring log file 6e6667125-b599-40c8-a13a@7487987
Rename [dbo].[OrderX] to Order
Caution: Changing any part of an object name could break scripts and stored procedures.
Altering [dbo].[AllCustomerView]...
The database SSDTDataMigrationPrototype already exists. Pre- and post deployment scripts will be executed
Skip post-script PostScripts_20171016123600_SetupInitialData.sql (already executed)
Skip post-script PostScripts_2017101710000_MergeColumnStreet.sql (already executed)
Update complete.
===== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ======

===== Deploy: 1 succeeded, 0 failed, 0 skipped ======

Azure DevOps CI Build



Stages Jobs

Stage	Jobs	Time	Status
Build	3 jobs completed Build the Api Build the Test Data Initializer Build the System Tests	7m 15s	100% tests passed
Run Integration Tests	2 jobs completed System tests with SQL Ser... System tests with SQL Se...	4m 59s	100% tests passed

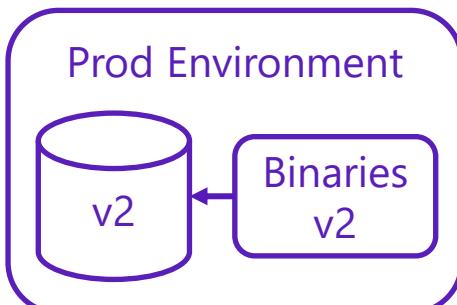
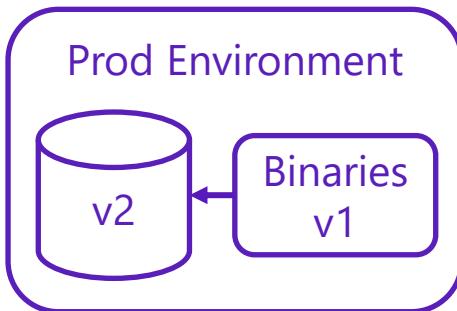
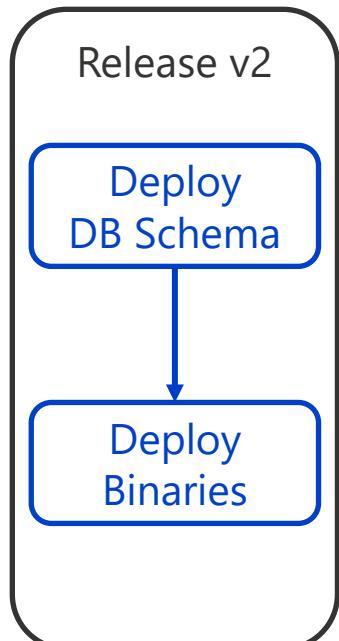
Artifacts

Published

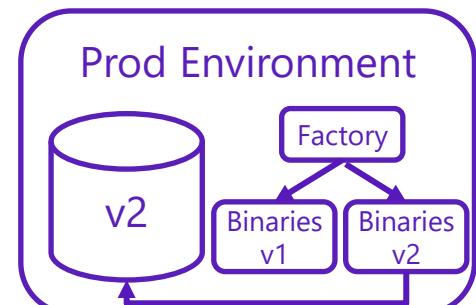
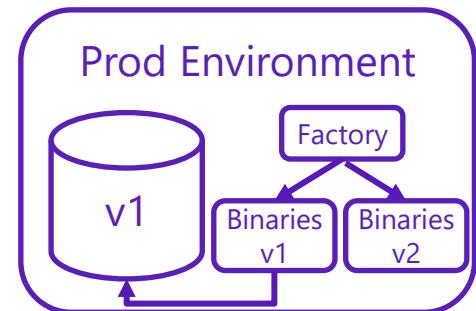
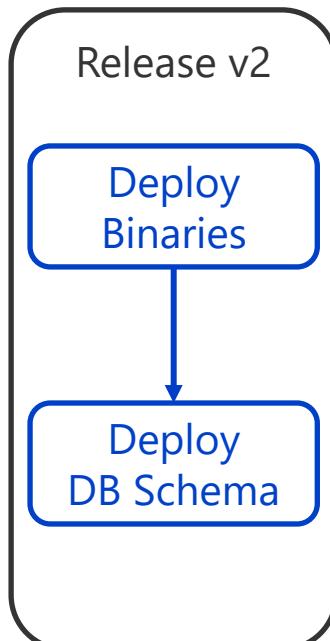
Name
clients
Code Coverage Report_2976
dacpacs
DevFun.DB.Build.dacpac
DevFun.DB.Build.deps.json
DevFunApi
SystemTests
TestDatalInitializer
linux
windows

Deployment Approaches

DB First



Code First



Where to put the fallback logic?

Database (SQL)

- Use views / triggers to support old schema

Advantages

- Old code just works during deployment

Disadvantages

- Have a lot of if statement in database logic
- Harder to test

Code (C#)

- Use factory to determine the implementation for the current database version
- Couple database version to features / implementation

Advantages

- Code is easier to test

Disadvantages

- More complexity in code
- Factory / Toggles needed

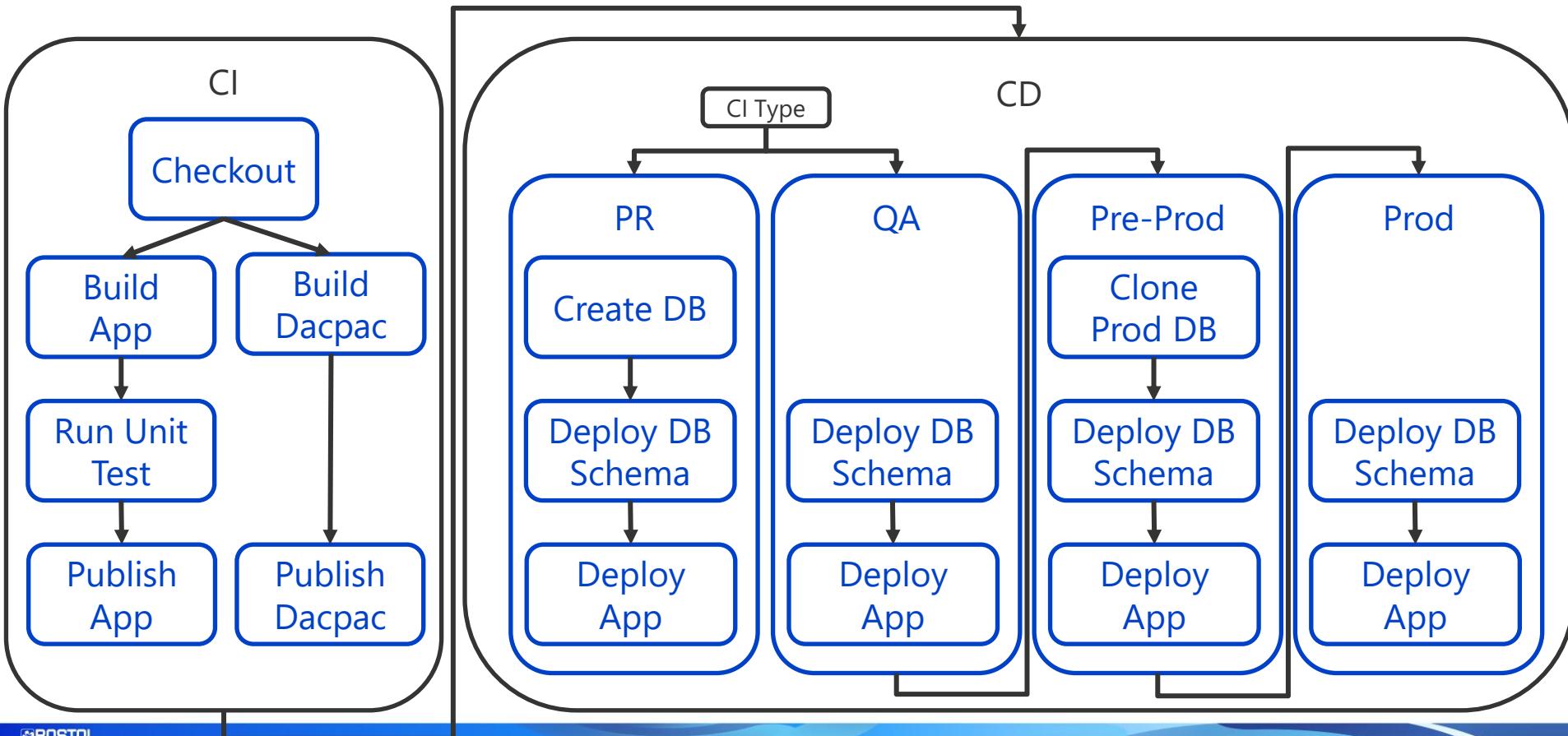
Support Rollback Scenarios

«If you can't get upgrade right, what leads you to believe you could get rollback right as well?» – *Buck Hodges*

Implement Rollback logic only if needed

- DB deployment is often complex and multi-step
- Hopefully never used – wasted time for implementation and testing?

CI / CD Pipeline



SQLPackage

Microsoft | Docs Dokumentation Learn Q&A Codebeispiele Show Ereignisse

Suche Anmelden

SQL-Dokumentation Übersicht Installieren Sicher Entwickeln Verwalten Analysieren Verweis

SQL Server herunterladen

Version

SQL Server 2022 Preview

Nach Titel filtern

Übersicht

Azure Data Studio >

> Häufige Programme für die Eingabeaufforderung

> Datenbankoptimierungsrategeber (DTA)

> Distributed Replay

Go-SQLCMD

> SQL Server-Konfigurations-Manager

> SQLCMD

> SSB-Diagnose

> SQL Server Data Tools (SSDT)

> SQL Server Management Studio (SSMS)

> SqlPackage.exe

SqlPackage.exe

Installieren von sqlpackage

Versionshinweise

> Aktionen mit SqlPackage

SqlPackage in Entwicklungspipelines

SqlPackage für Azure Synapse Analytics

Problembehandlung beim Import/Export

> SQL Server Profiler

> Native Visual Studio-Hilfsprogramme

> Erweiterte Funktionen

> Visual Studio Code

> Lernprogramme

> SQL Server unter Linux

> SQL in Azure

> SQL Server auf Azure Arc-fähigen Servern

Herunterladen und Installieren von SqlPackage

Artikel • 10.06.2022 • 3 Minuten Lesedauer • 6 Mitwirkende

In diesem Artikel

DacFX

Automatisierte Umgebungen

Linux (.NET Core)

macOS (.NET Core)

Windows (.NET Core)

Windows (.NET Framework)

Deinstallieren von SqlPackage

Unterstützte Betriebssysteme

Verfügbare Sprachen

Nächste Schritte

SqlPackage wird auf Windows, macOS und Linux ausgeführt.

Download und Installation der neuestes Releases:

Plattform	Download	Veröffentlichungsdatum	Version	Entwickeln
Windows	MSI-Installationsprogramm	24. Mai 2022	19.1	16.0.6161.0
macOS .NET Core	zip-Datei	24. Mai 2022	19.1	16.0.6161.0
Linux .NET Core	zip-Datei	24. Mai 2022	19.1	16.0.6161.0
Windows .NET Core	zip-Datei	24. Mai 2022	19.1	16.0.6161.0

Weitere Informationen über die neueste Version finden Sie in den Versionshinweisen.
Informationen zum Herunterladen zusätzlicher Sprachen finden Sie im Abschnitt Verfügbare Sprachen.

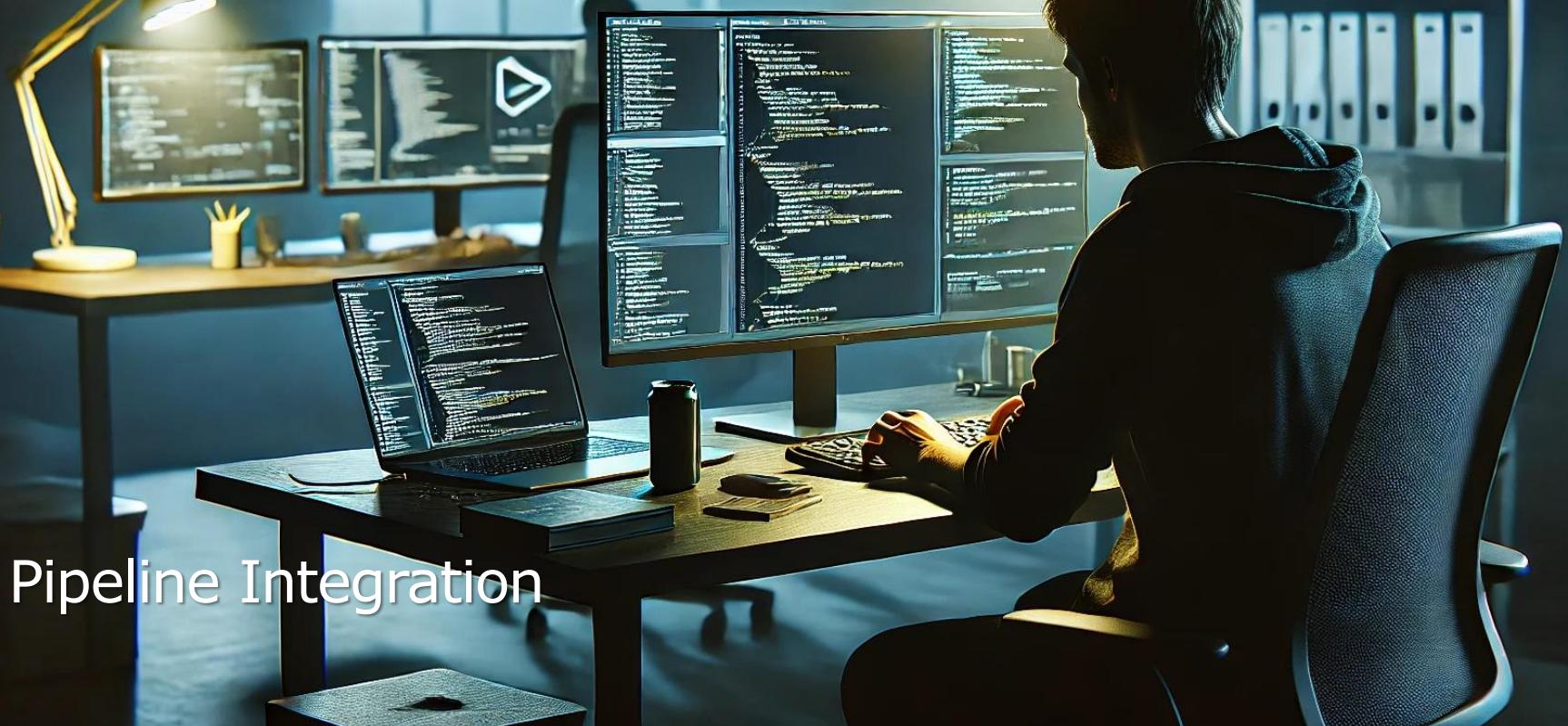
DacFX

SqlPackage ist eine Befehlszeilschnittstelle für das DacFx-Framework, die einige der

steps:

```
- pwsh: |
  /opt/sqlpackage/sqlpackage
  /a:Publish
  /p:BlockOnPossibleDataLoss=
    ${{ parameters.blockOnDataLoss }}
  /p:GenerateSmartDefaults=True
  /tcs:"${{ parameters.connection }}"
  /sf:"${{ parameters.dacpacFile }}"
  displayName: 'Deploy DACPACs'
```

Demo



Pipeline Integration

A close-up, low-angle shot of several rowers in a racing shell. Their legs and torsos are visible as they pull on their oars. The water is choppy, creating white spray at the bow. The rowers are wearing blue and red athletic gear.

Chapter 6/8

Deployment Security Considerations

Security Considerations

- Dedicated deployment agents
- Strict security boundary between dev/test and prod
- Use dedicated users for each database / service
- Use dedicated users for
 - Schema deployment with DDL
 - Application / service with read/write permissions



Chapter 7/8

Containers

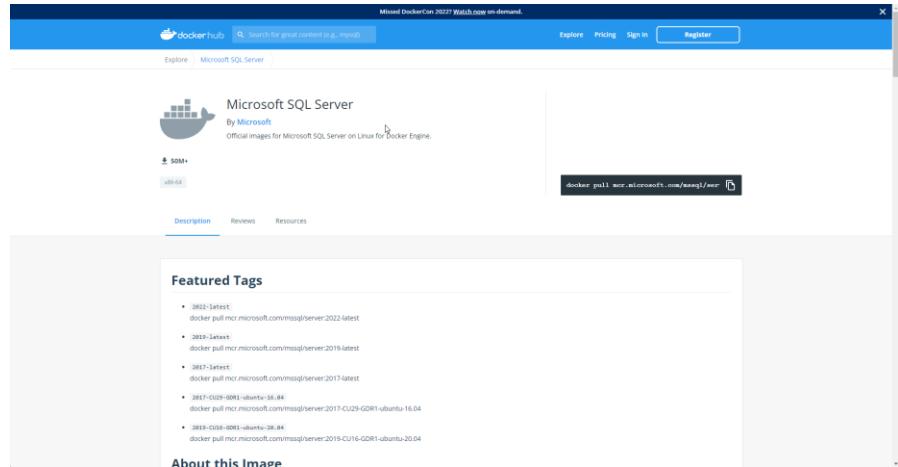
4tecture®
empower your software solutions

Containers are your friends

- Local development environment
- Test multiple database versions
- Fast baselines for testing
- Ideal artifact for DB schema deployment tools

SQL Server Container Images

- Linux Container
- Versions
 - 2022
 - 2019
 - 2017



https://hub.docker.com/_/microsoft-mssql-server

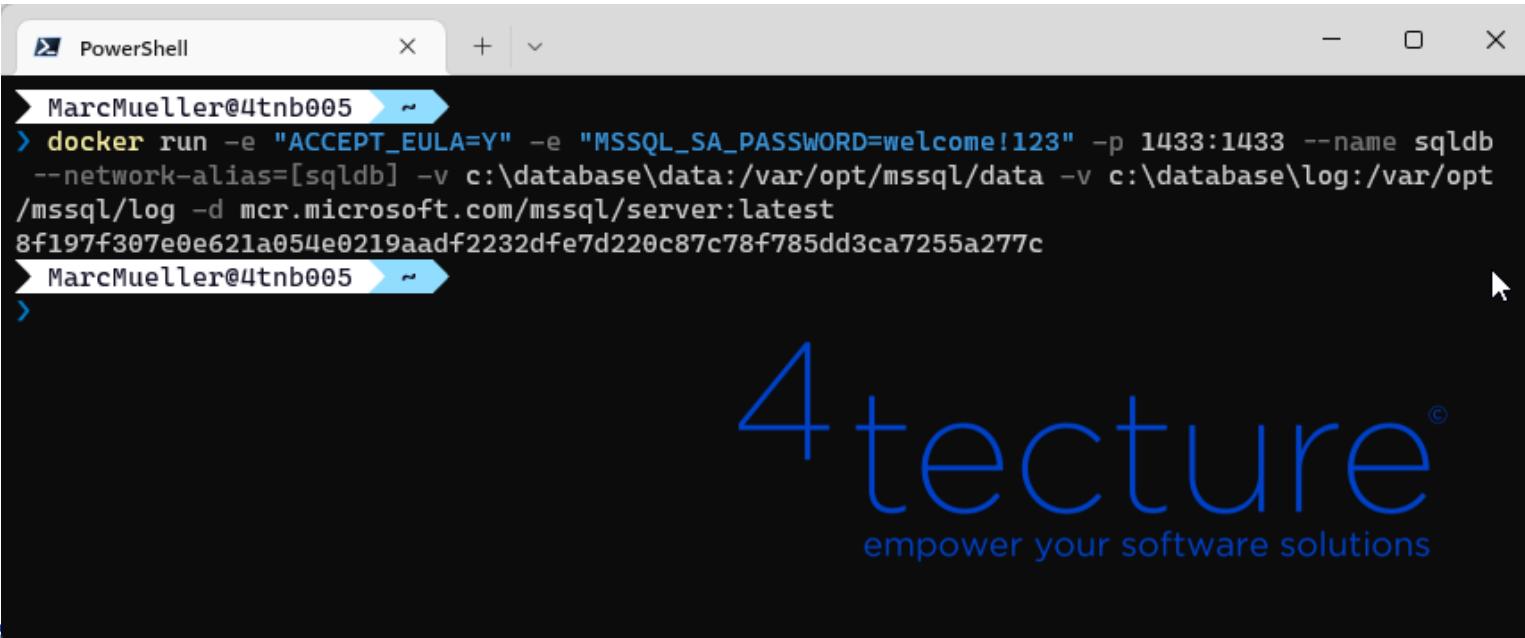
Demo



SQL Server in Container

Local Development Environment

- Fast
- Multiple versions side-by-side
- External persistency



A screenshot of a Windows PowerShell window titled "PowerShell". The window shows a command being run to start a Docker container for Microsoft SQL Server. The command is:

```
MarcMueller@4tnb005 ~
> docker run -e "ACCEPT_EULA=Y" -e "MSSQL_SA_PASSWORD=welcome!123" -p 1433:1433 --name sqldb
--network-alias=[sqldb] -v c:\database\data:/var/opt/mssql/data -v c:\database\log:/var/opt
/mssql/log -d mcr.microsoft.com/mssql/server:latest
8f197f307e0e621a054e0219aadf2232dfe7d220c87c78f785dd3ca7255a277c
> MarcMueller@4tnb005 ~
```

The container ID is displayed as 8f197f307e0e621a054e0219aadf2232dfe7d220c87c78f785dd3ca7255a277c.

4tecture[®]
empower your software solutions

Test Multiple Database Versions

- Locally
- Azure Pipelines

A screenshot of the Azure Pipelines interface. On the left, there's a sidebar with a cursor icon pointing to the 'Jobs' tab, which is currently selected. Below the tabs, there's a section titled 'Name' with a list of pipeline stages. Each stage has a green checkmark icon and a brief description:

- Build the Api
- Build the Test Data Initializer
- Build the System Tests
- System tests with SQL Server sql2019
- System tests with SQL Server sql2022
- Update the pull request with policies

A screenshot of the .azure-pipelines/devfunapi-ci.yml YAML file. The file defines a multi-stage pipeline named 'DevFunApi-CI'. It includes stages for building the API and initializing test data. The pipeline also performs system tests using different versions of Microsoft SQL Server (sql2019, sql2022) running in Docker containers. The configuration uses environment variables like SA_PASSWORD and ACCEPT_EULA, and specifies connection strings for the databases.

```
25 - name: containerdbserver
26 | value: 'sql'
27 - name: Dacpacfile
28 | value: '$(Pipeline.Workspace)/dacpacs/DevFun.DB.Build.dacpac'
29
30 resources:
31   containers:
32     - container: sql2022
33       image: mcr.microsoft.com/mssql/server:2022-latest
34       env:
35         - SA_PASSWORD: "$(containerdbpassword)"
36         - ACCEPT_EULA: "Y"
37     - container: sql2019
38       image: mcr.microsoft.com/mssql/server:2019-latest
39       env:
40         - SA_PASSWORD: "$(containerdbpassword)"
41         - ACCEPT_EULA: "Y"
42     - container: sql2017
43       image: mcr.microsoft.com/mssql/server:2017-latest
44       env:
45         - SA_PASSWORD: "$(containerdbpassword)"
46         - ACCEPT_EULA: "Y"
47     - container: api
48       image: '$(ImageName):$(Build.BuildNumber)'
49       endpoint: 4taksDemoAcr
50       env:
51         - DEVFUNOPTIONS__DEPLOYMENTENVIRONMENT: "Development"
52         - CONNECTIONSTRINGS__DEVFUNDATABASE: 'Server=tcp:$(containerdbserver),1433;D...
53         - LICENSE__LICENSEDATA: "$(LicenseData)"
54     - container: worker
55       image: 4taksdemocracy.azurecr.io/linuxworker:latest
```

Fast Baseline for Testing

- Backup / Restore may increase execution time drastically
- Container images can be built with database baselines
- Multiple application data versions for migration testing
- Specific data baseline for corresponding tests



```
MarcMueller@4tnb005 ~
> # Create the container
MarcMueller@4tnb005 ~
> docker run -e "ACCEPT_EULA=Y" -e "MSSQL_SA_PASSWORD=welcome!123" -p 1433:1433 --name devfunqlfdbpreparation -d mcr.microsoft.com/mssql/server:latest
94ueb9ff7bd29b351c2cf85a29aeb89bf830adaee49cef7e2e1ac3eb44e7130c
MarcMueller@4tnb005 ~
> # Add data / create a baseline
MarcMueller@4tnb005 ~
> docker commit devfunqlfdbpreparation 4taksdemoacr.azurecr.io/devfundb:0.0.1
sha256:065abddfe46b92d68190538b0faa94e2c0c4b499d7474a7ca5658d8672e35326
MarcMueller@4tnb005 ~
```



```
MarcMueller@4tnb005 ~
> docker run -e "ACCEPT_EULA=Y" -e "MSSQL_SA_PASSWORD=welcome!123" -p 1433:1433 --name devfunqlfdbpreparation -d 4taksdemoacr.azurecr.io/devfundb:0.0.1
e2b0fb1cc2d85474ea5b9615cd4b7873db8dea24bd109ead5ala1c674aee4b
MarcMueller@4tnb005 ~
```

Containerize your DB schema deployment

- **Tooling**
 - Put SQL tools into a “worker container”
(SQL Package, Powershell Modules)
 - Create a custom CLI for schema deployment
- **Dacpac**
 - Include specific dacpac files into container
(versioned image)
 - Volume mount the dacpac files into container at deployment time

“Worker Container”

```
1 FROM mcr.microsoft.com/dotnet/sdk:6.0
2
3 RUN curl -sL https://aka.ms/InstallAzureCLIDeb | bash
4
5 RUN apt-get update && \
6     apt-get install -y --no-install-recommends unzip && \
7     rm -rf /var/lib/apt/lists/* && \
8     wget -q -O /opt/sqlpackage.zip https://go.microsoft.com/fwlink/?linkid=2157202 \
9     && unzip -qq /opt/sqlpackage.zip -d /opt/sqlpackage \
10    && chmod +x /opt/sqlpackage/sqlpackage && rm -f /opt/sqlpackage.zip
11
12 RUN apt-get update && \
13     apt-get install -y curl gnupg apt-transport-https && \
14     curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add - && \
15     sh -c "deb [arch=amd64] https://packages.microsoft.com/repos/microsoft-debian-stretch-prod stretch main" > /etc/apt/sources.list.d/microsoft.list" && \
16     apt-get update && \
17     apt-get install -y powershell && \
18     curl -fsSL https://deb.nodesource.com/setup\_16.x | bash - && \
19     apt-get install -y nodejs
20
21 RUN pwsh -c "Install-Module SqlServer -Scope AllUsers -Force"
```

Custom CLI for Schema Deployment

2 references | Marc Müller, 273 days ago | 1 author, 1 change

```
class DbSchemaDeployer
{
    private DacOperationStatus operationStatus;

    1 reference | Marc Müller, 273 days ago | 1 author, 1 change
    public Task<bool> DeployDatabase(string server, string database,
        string user, string password, string dacpacFile)
    {
        var csBuilder = new SqlConnectionStringBuilder();
        csBuilder.DataSource = server;
        csBuilder.InitialCatalog = database;
        if (!string.IsNullOrWhiteSpace(user))
        {
            csBuilder.UserID = user;
        }

        if (!string.IsNullOrWhiteSpace(password))
        {
            csBuilder.Password = password;
        }

        return DeployDatabase(csBuilder.ConnectionString, dacpacFile);
    }
}
```

2 references | Marc Müller, 273 days ago | 1 author, 1 change

```
public async Task<bool> DeployDatabase(string connectionString, string dacpacFile)
{
    var dacService = NeedsAccessToken(connectionString) ? new DacServices(connectionString, new AccessTokenProvider()) : new DacServices(connectionString);
    dacService.Message += MessageHandler;
    dacService.ProgressChanged += ProgressChangedHandler;
    dacService.Publish(DacPackage.Load(dacpacFile), (new SqlConnectionStringBuilder(connectionString)).InitialCatalog,
        new PublishOptions() { DeployOptions = new DacDeployOptions() { BlockOnPossibleDataLoss = false, GenerateSmartDefaults = true } });

    while (operationStatus == DacOperationStatus.Pending || operationStatus == DacOperationStatus.Running)
    {
        await Task.Delay(500).ConfigureAwait(false);
    }

    return operationStatus == DacOperationStatus.Completed;
}
```

```
1 FROM mcr.microsoft.com/dotnet/runtime:6.0 AS base
2 WORKDIR /app
3
4 FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
5 WORKDIR /src
6 COPY ["*.props", "./"]
7 COPY ["DevFun.DbMigration.Cli/DevFun.DbMigration.Cli.csproj", "DevFun.DbMigration.Cli/"]
8 COPY ["DevFun.DB.Build/DevFun.DB.Build.csproj", "DevFun.DB.Build/"]
9 RUN dotnet restore "DevFun.DbMigration.Cli/DevFun.DbMigration.Cli.csproj"
10 COPY .
11 WORKDIR "/src/DevFun.DbMigration.Cli"
12 RUN dotnet build "DevFun.DbMigration.Cli.csproj" -c Release -o /app/build
13
14 FROM build AS publish
15 RUN dotnet publish "DevFun.DbMigration.Cli.csproj" -c Release -o /app/publish
16 RUN cp /app/build/DevFun.DB.Build.dacpac /app/publish
17
18 FROM base AS final
19 WORKDIR /app
20 COPY --from=publish /app/publish .
21 ENTRYPOINT ["dotnet", "DevFun.DbMigration.Cli.dll"]
```

A close-up, low-angle shot of several rowers in a racing shell. Their hands are gripping yellow and black oars, which are angled downwards. The water is visible at the bottom, creating a sense of motion. The rowers are wearing blue and red athletic gear.

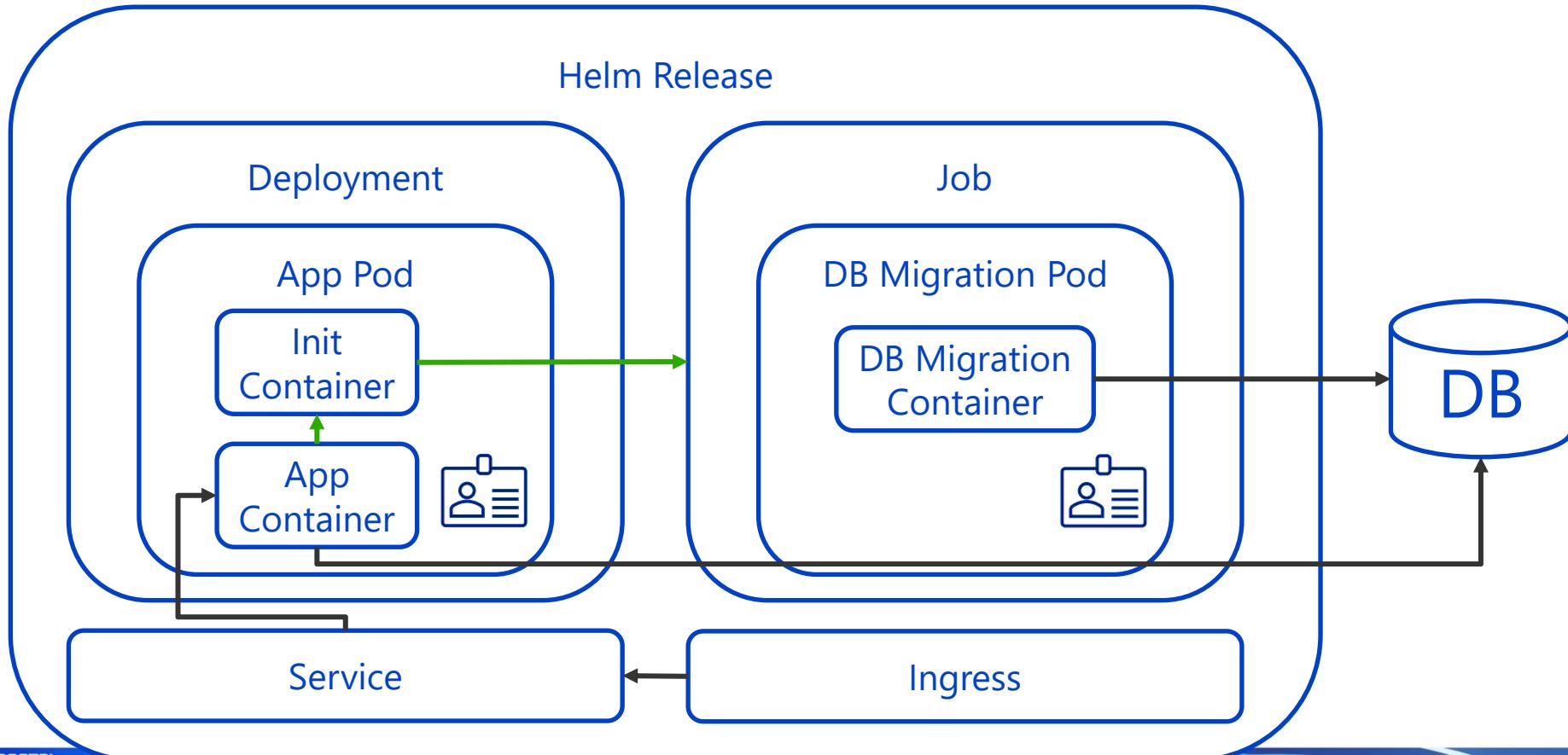
Chapter 8/8

Autonomous Deployment

Autonomous Application Packages

- CI/CD pipelines work great for internal services
- If an application package is distributed, the schema deployment should be part of it
- Logic from the CI/CD pipeline is moved to the application package
- CI/CD pipelines can be simplified

Helm Release



Demo



Kubernetes Rollout

DAK dev-aks-k8sdemo-westeurope-...

SAA Cluster

Nodes

DAK

Workloads

- Overview
- Pods
- Deployments
- DaemonSets
- StatefulSets
- ReplicaSets
- Jobs
- CronJobs

Config

- ConfigMaps
- Secrets
- Resource Quotas
- Limit Ranges
- HPA
- Pod Disruption
- Budgets
- Priority Classes

Network

- Services
- Endpoints
- Ingresses
- Network Policies
- Port Forwarding

Storage

- Persistent Volume
- Claims

Overview Pods Deployments DaemonSets StatefulSets ReplicaSets Jobs CronJobs

Pods 6 items Namespace: staging Search Pods...

Name	Namespace	Containers	Restarts	Controlled By	Node	QoS	Age	Status	⋮
devfunapi-76849796cd-2r7c2	staging	2	0	ReplicaSet	aks-linux1-38181550-	BestEffort	6m17s	Running	⋮
devfunapi-76849796cd-lqnst	staging	2	0	ReplicaSet	aks-linux1-38181550-	BestEffort	5m11s	Running	⋮
devfunapi-76849796cd-s2scw	staging	2	0	ReplicaSet	aks-linux1-38181550-	BestEffort	5m31s	Running	⋮
devfunweb-6b5599dff5-4mph4	staging	2	0	ReplicaSet	aks-linux1-38181550-	BestEffort	46h	Running	⋮
devfunweb-6b5599dff5-6xc4m	staging	2	0	ReplicaSet	aks-linux1-38181550-	BestEffort	46h	Running	⋮
devfunweb-6b5599dff5-8dkg9	staging	2	0	ReplicaSet	aks-linux1-38181550-	BestEffort	46h	Running	⋮

A close-up, low-angle shot of several rowers in a racing shell. Their legs and the oars are visible, with the water spray from the oars creating a misty effect. The rowers are wearing blue and red athletic gear.

Automated DB Schema Deployments

Q & A

4tecture®
empower your software solutions

Recap

- Database Development fully integrated into development process
- No manual schema changes in deployment process
- Handle corner cases with custom migration scripts integrated in automated deployment
- Database Deployment might be part of application package vs. pipeline automation

Thank you for your attention!

If you have any questions do not hesitate to contact us:

4ecture GmbH
Industriestrasse 25
CH-8604 Volketswil

+41 44 508 37 00
info@4ecture.ch
www.4ecture.ch

Marc Müller
Principal Consultant

www.powerofdevops.com

4ecture®
empower your software solutions



A close-up photograph of several hands reaching towards a central wooden puzzle piece. The puzzle piece is light-colored wood with a dark green and red section. The hands belong to different people, suggesting collaboration. The background is blurred.

4tecture[©]

empower your software solutions