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

MVP
Microsoft Most Valuable Professional

4tecture©
empower your software solutions

Our Products:

Multi-Tenant OpenID Connect Identity Provider

ProAuth

www.proauth.net

Enterprise Application Framework for .NET

ReaFx

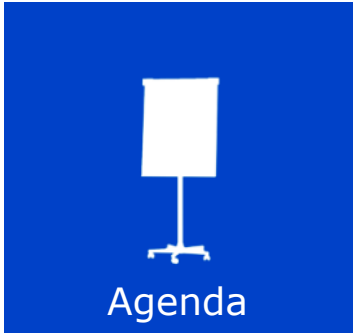
www.reafx.net

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

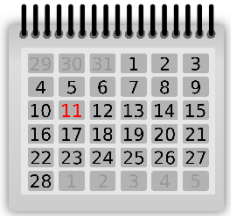


Chapter 1/8

Intro

4tecture
empower your software solutions

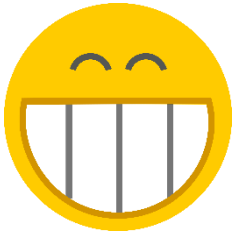
Write Code



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

Chapter 2/8

Database as Code with SSDT

4tecture
empower your software solutions

SSDT - Characteristic

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

SSDT - Characteristic

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

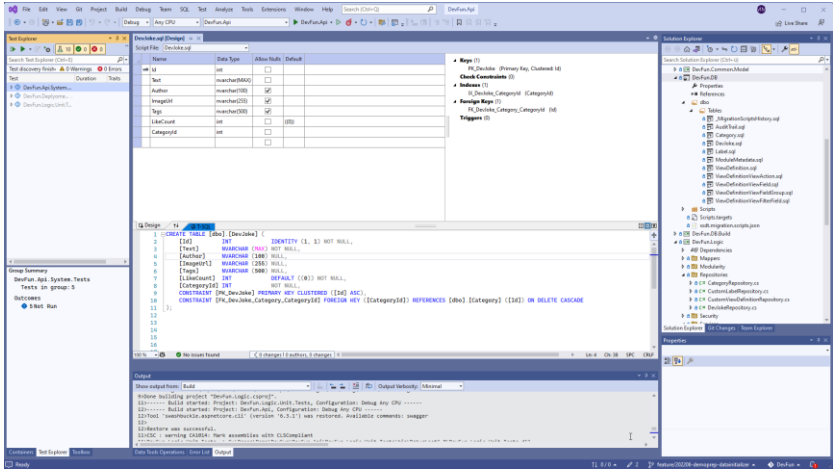
DB Schema Migrations (Static & Dynamic SQL)

Single Pre- and Post Script Logic

Microsoft.Data.Tools.Msbuild
(NuGet)

SSDT - Features

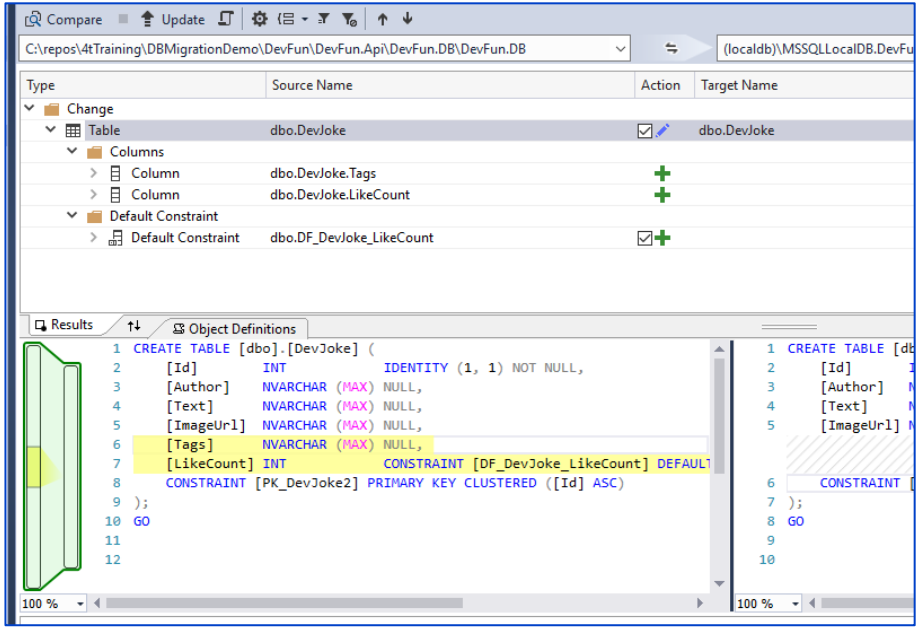
- Build time validation / IntelliSense Support



- Bidirectional Scheme Comparison (SSDT ⇔ DB)
- Bidirectional Scheme Synchronization (SSDT ⇔ DB)
- Versioned migration and schemes artifact (DACPAC)
- Code-base integration / Change tracking (GIT)

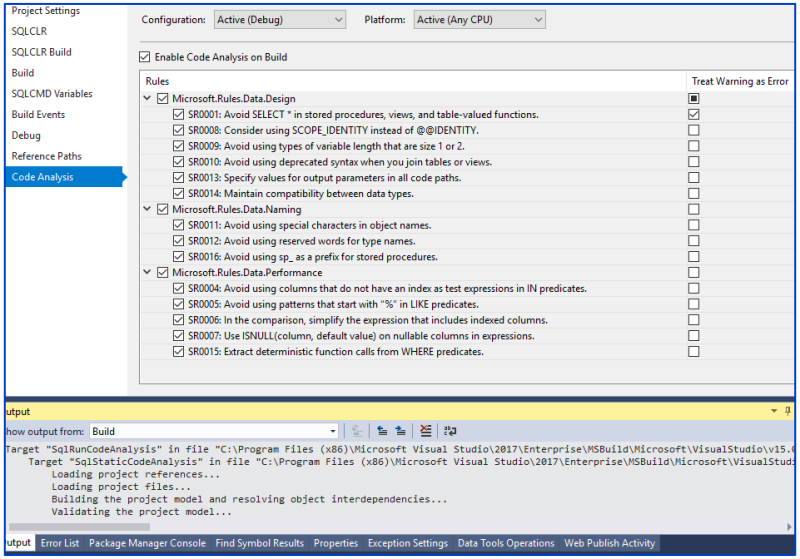
Schema Compare

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



Code Analysis

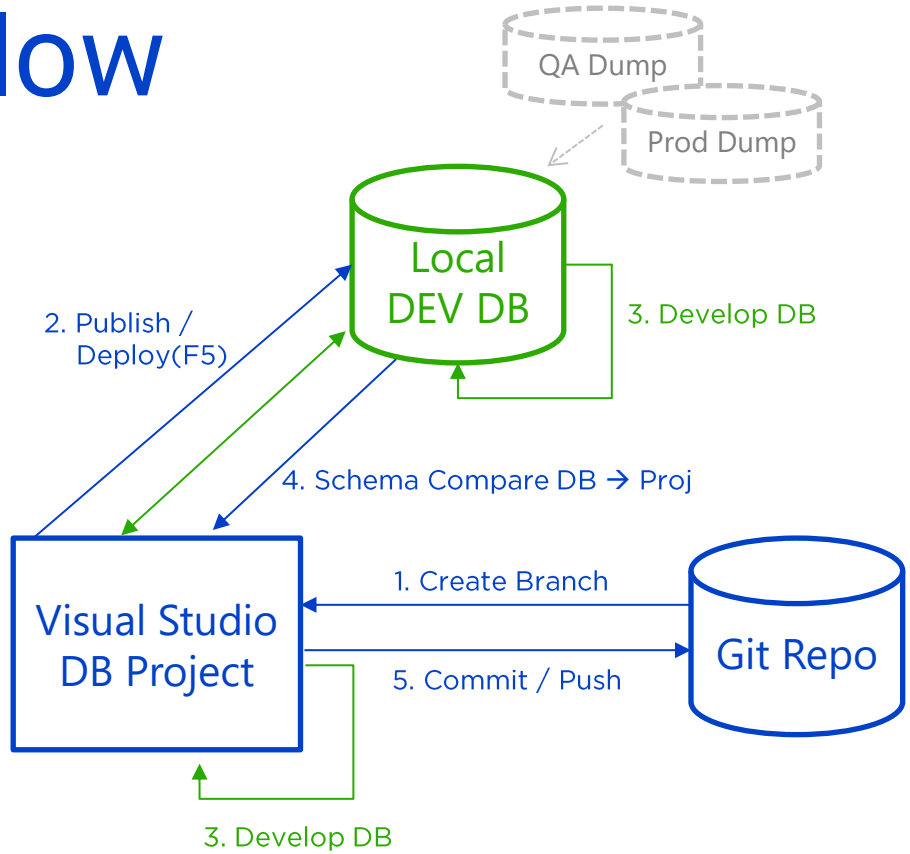
- Standardized Design Patterns
- Code Quality
- Reduce Code Smells
- Supports Static-/ and Dynamic SQL
- Tables, SP, UDDT, Views…




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

4tecture
empower your software solutions

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

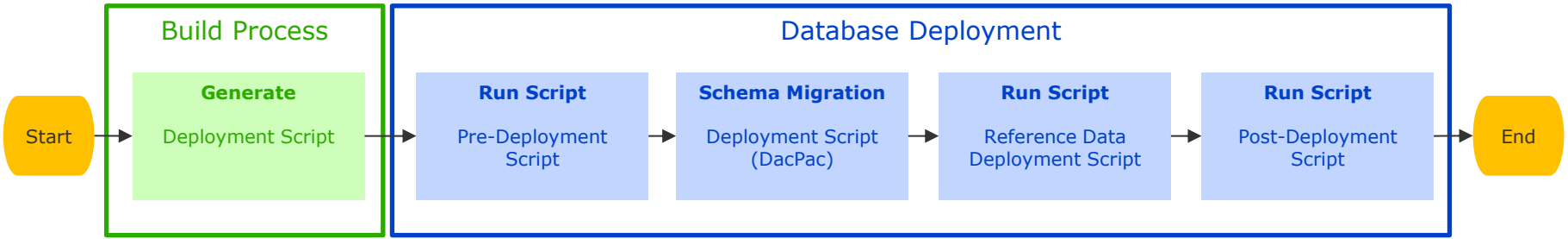
SSDT can easily be extended….

Concept

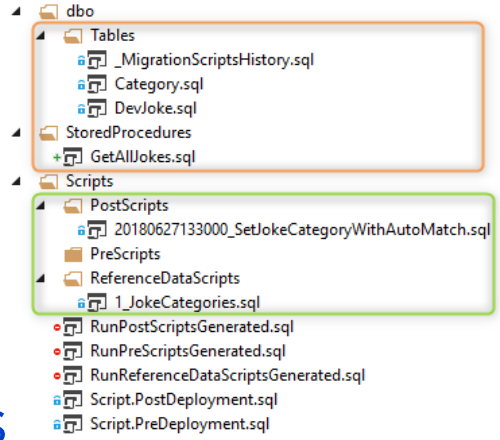
- Logical separation for Migration Scripts
- Traceability (dev-phase)
- Automatic migration management
- Managed Pre-/Post Scripts
- Managed Reference Data Scripts
- Error- and Transaction Handling
- Pre-Validation / Syntax Check
- History Management

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

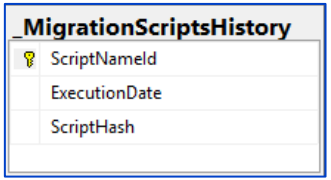
- Manage Pre-, Post and Reference Data Scripts
- Manage declarative scheme changes

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



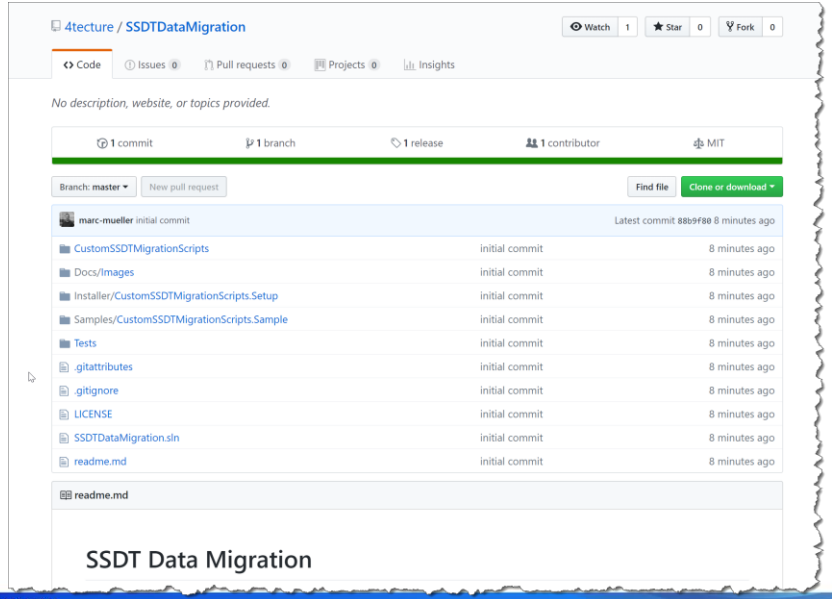
	ScriptNameId	ExecutionDate	ScriptHash
1	PostScripts\20180627133000_SetJokeCategoryWithAu...	2018-06-26 13:35:05.0366667	rLdsdu8U65wWDXQbL99aAA==
2	PostScripts\20180627134200_HashPlainPasswords.sql	2018-06-26 13:35:05.0366667	gsknD7VbLNUMVgnr5SodmQ==

- 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



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
  },
```

Chapter 4/8

Migration Scenarios

4tecture
empower your software solutions

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

```
--MERGE generated by 'sp_generate_merge' stored procedure, Version 0.93
--Originally by Vyas (http://vyaskn.tripod.com): sp_generate_inserts (build 22)
--Adapted for SQL Server 2008/2012 by Daniel Nolan (http://danere.com)

SET NOCOUNT ON

MERGE INTO [dbo].[PaymentMethod] AS Target
USING (VALUES
  (N'creditcard',N'The credit card payment method',N'Credit Card')
 ,(N'debitcard',N'The debit card payment method',N'Debit Card')
) AS Source ([Identifier],[Description],[Name])
ON (Target.[Identifier] = Source.[Identifier])
WHEN MATCHED AND (
  NULLIF(Source.[Description], Target.[Description]) IS NOT NULL OR NULLIF(Target.[Description], Source
  NULLIF(Source.[Name], Target.[Name]) IS NOT NULL OR NULLIF(Target.[Name], Source.[Name]) IS NOT NULL
 UPDATE SET
  [Description] = Source.[Description],
  [Name] = Source.[Name]
WHEN NOT MATCHED BY TARGET THEN
 INSERT([Identifier],[Description],[Name])
 VALUES(Source.[Identifier],Source.[Description],Source.[Name])
WHEN NOT MATCHED BY SOURCE THEN
 DELETE
;

DECLARE @mergeError int
 , @mergeCount int
SELECT @mergeError = @@ERROR, @mergeCount = @@ROWCOUNT
IF @mergeError != 0
 BEGIN
 PRINT 'ERROR OCCURRED IN MERGE FOR [dbo].[PaymentMethod]. Rows affected: ' + CAST(@mergeCount AS VARC
 END
ELSE
 BEGIN
 PRINT '[dbo].[PaymentMethod] rows affected by MERGE: ' + CAST(@mergeCount AS VARCHAR(100));
 END
```

Chapter 5/8

DevOps Process

Challenges

Downtime

Roll Forward / Rollback

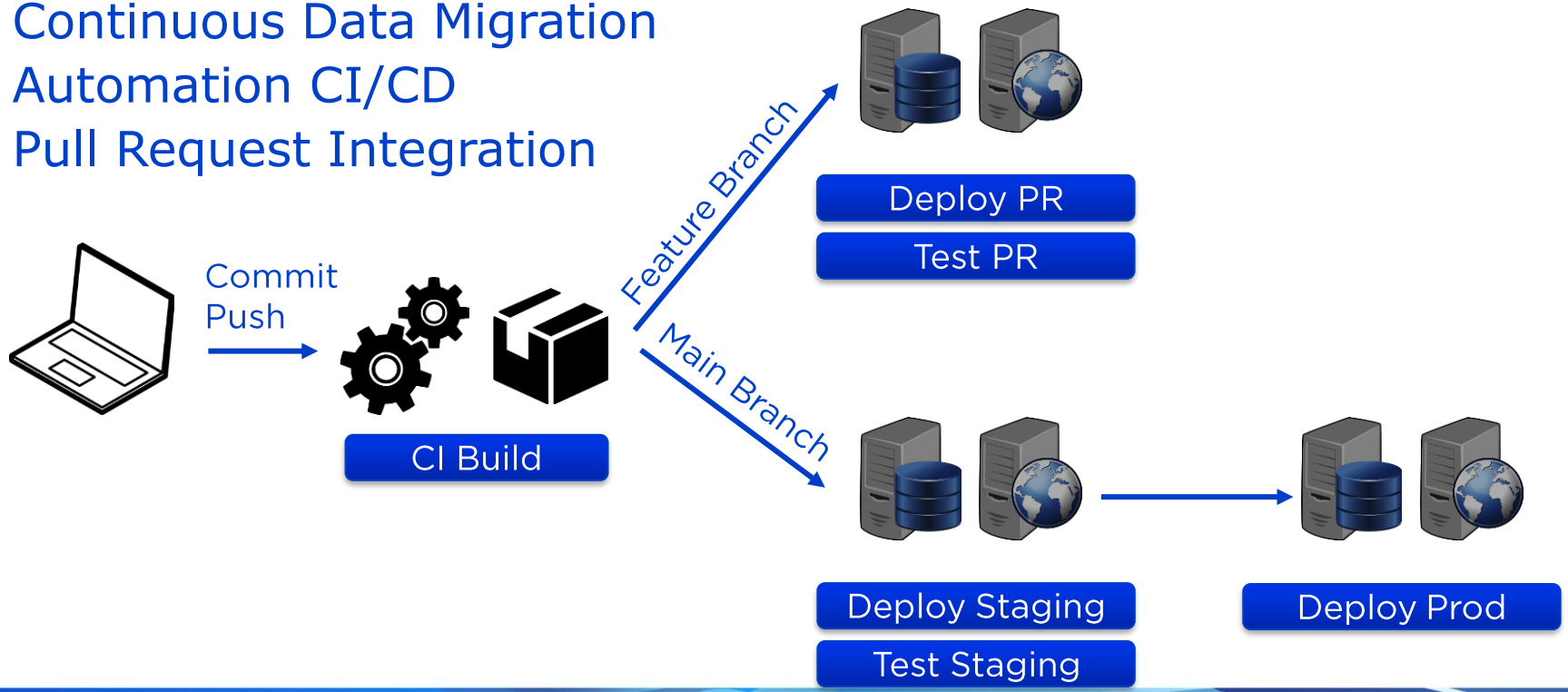
Non-Breaking Changes / Breaking Changes

Deployment Order: DB First / Code First

Handling multiple Schema Versions

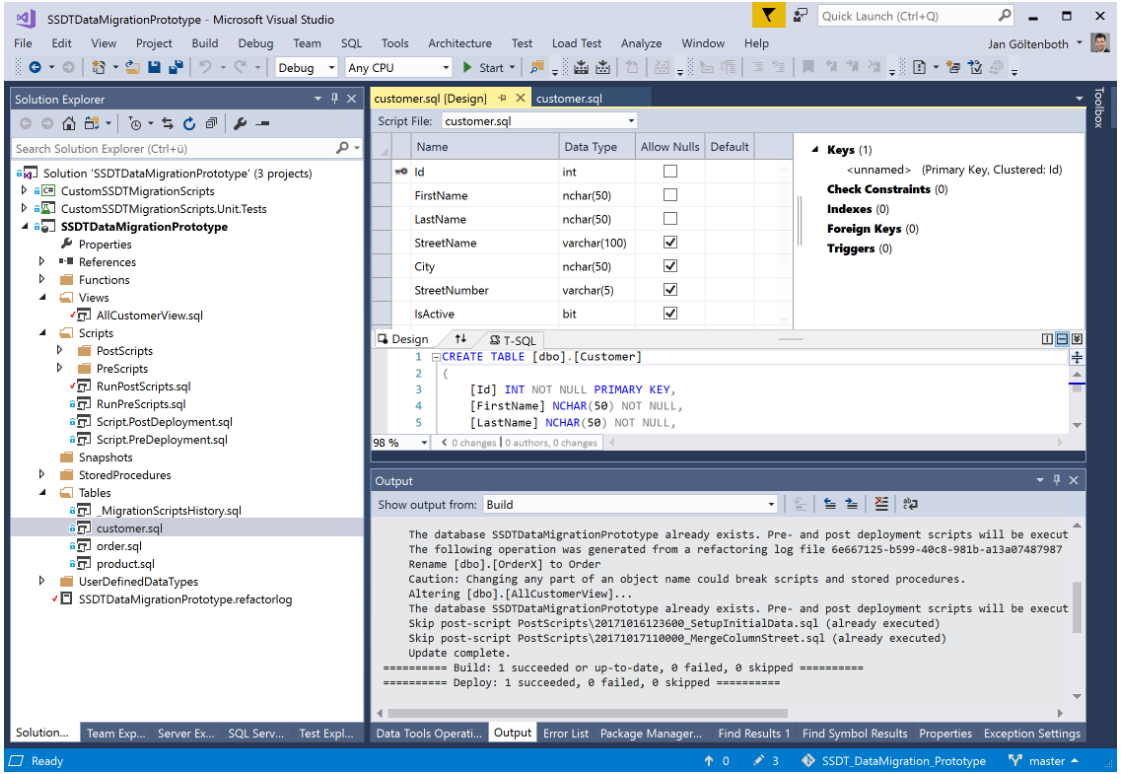
DevOps Process Integration

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

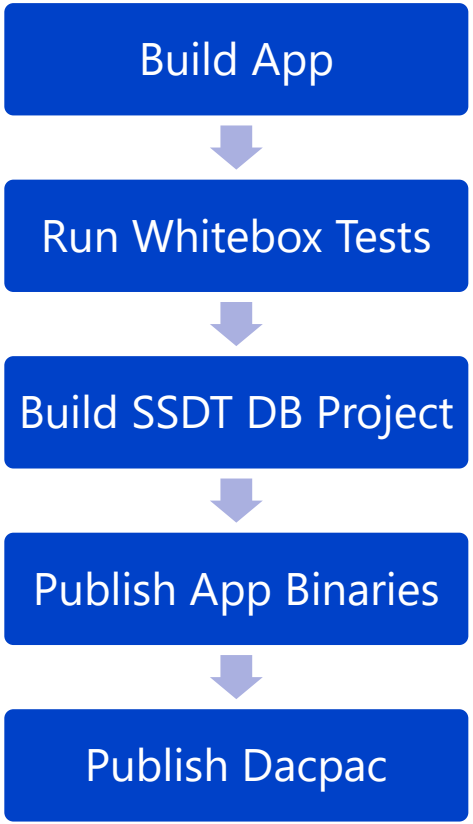


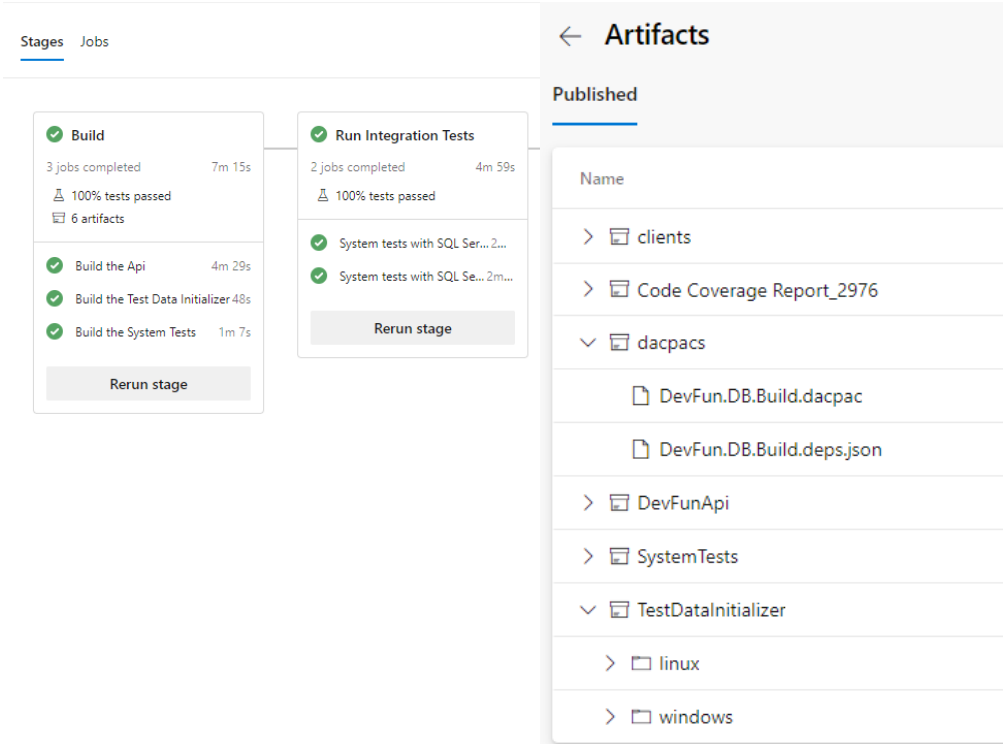
Visual Studio Integration

- Design
- Validate
- Build
- Deploy (local dev)



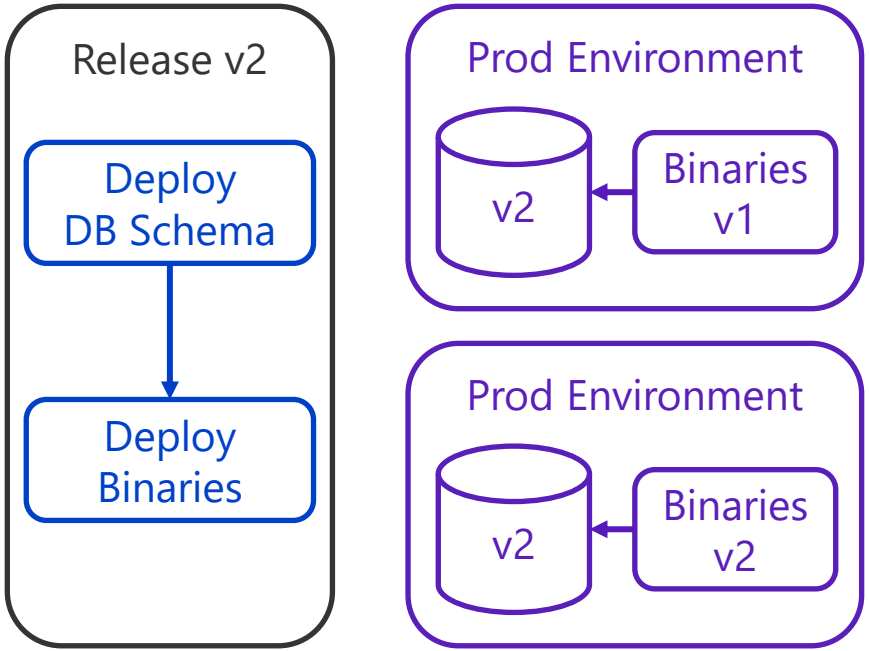
Azure DevOps CI Build



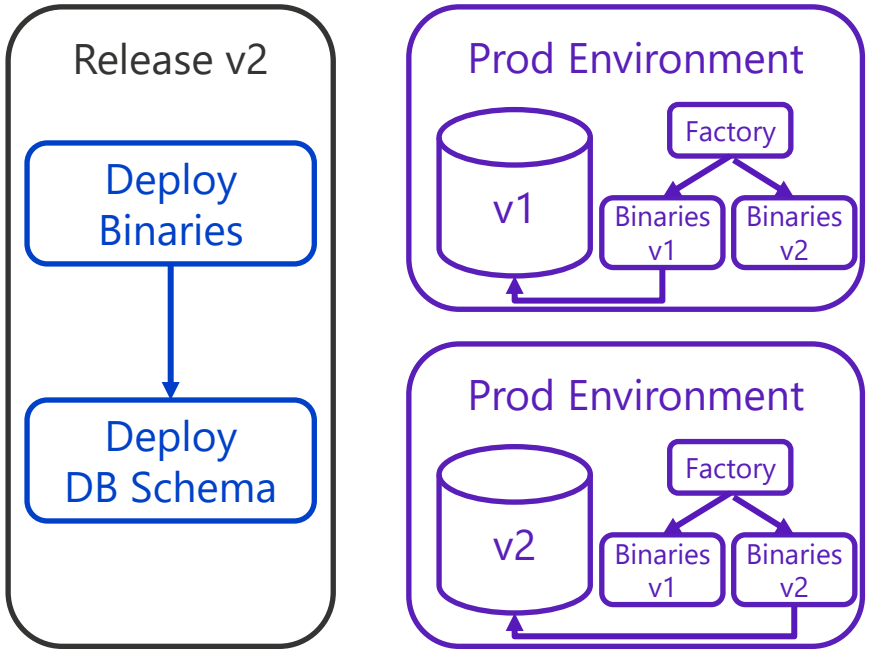


Deployment Approaches

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

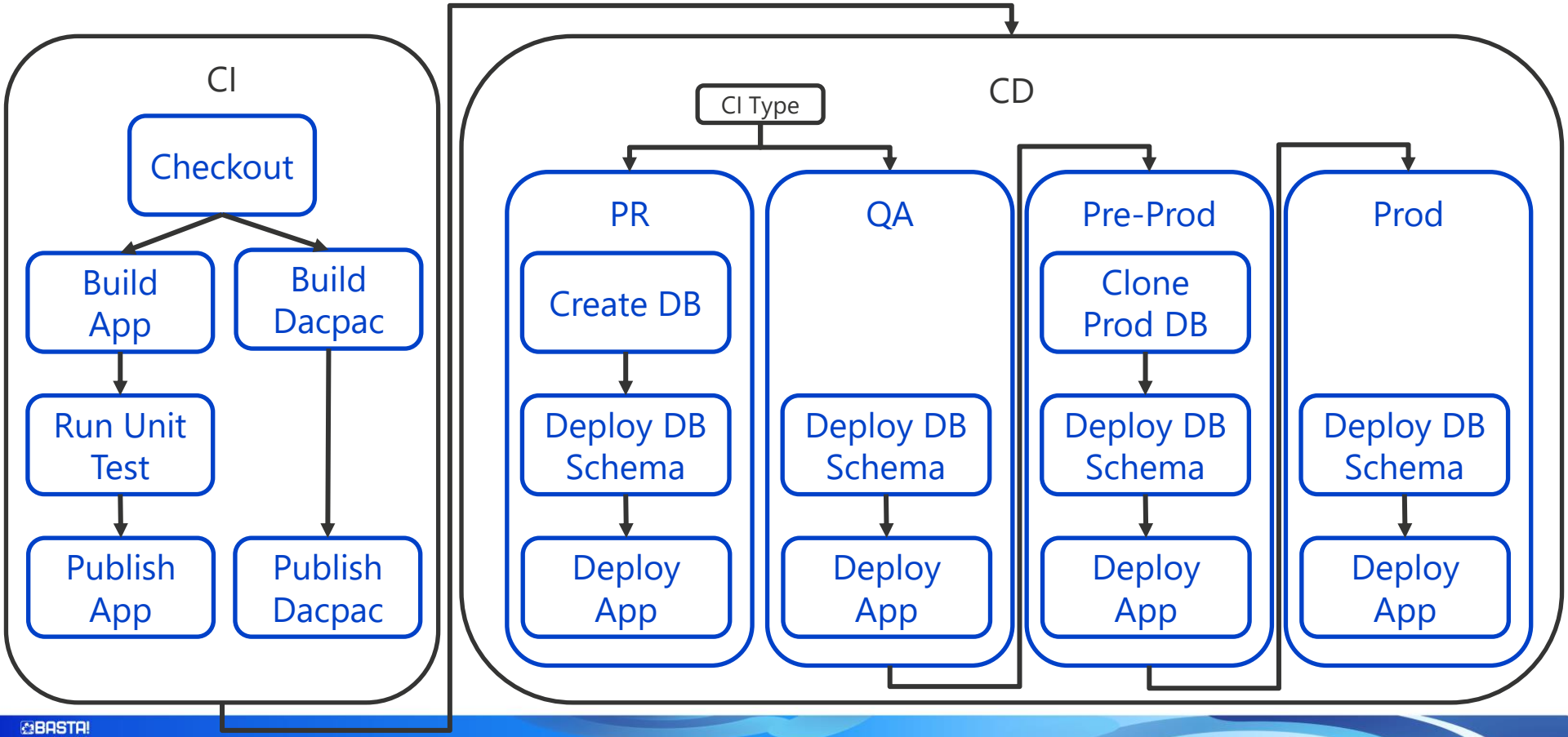
Support Rollback Scenarios

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

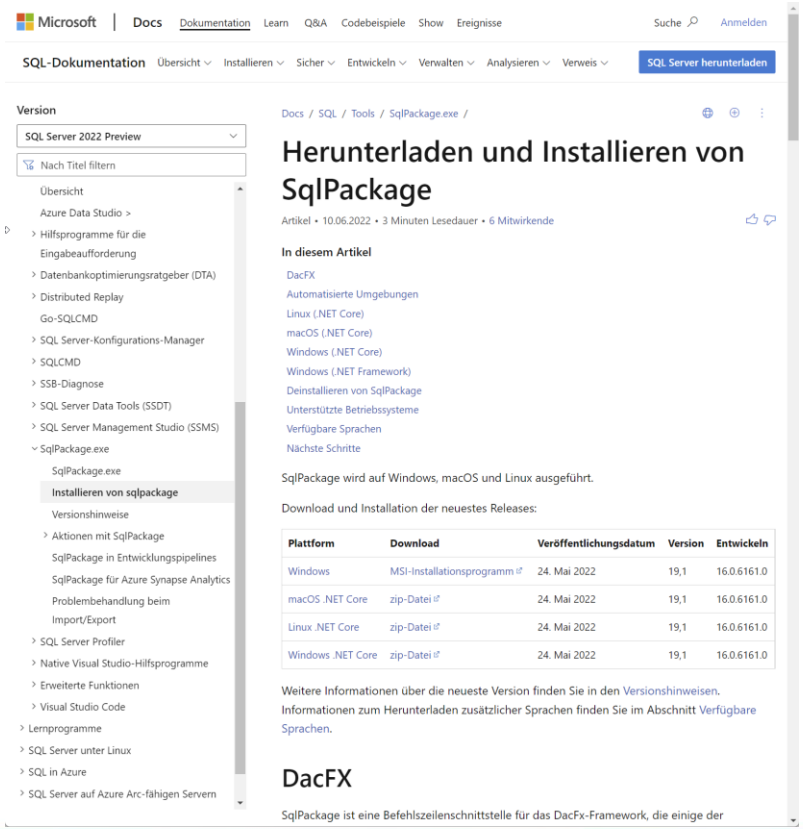
Implement Rollback logic only if needed

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

CI / CD Pipeline



SQLPackage




```yaml
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

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

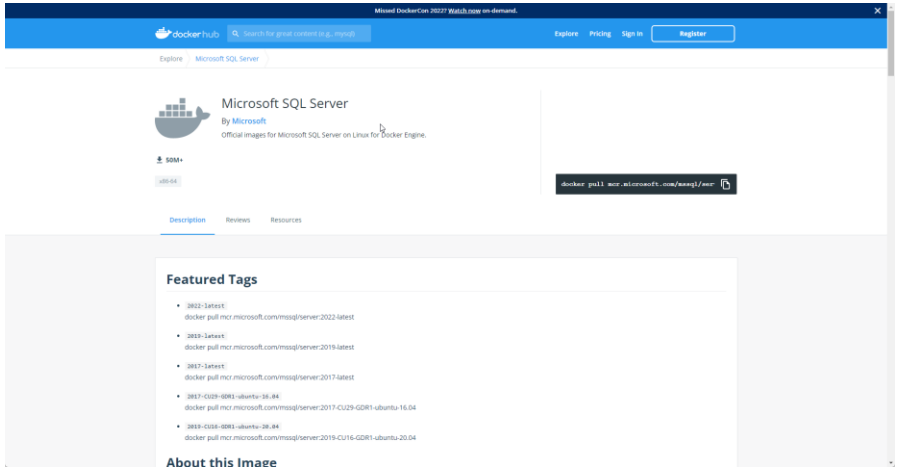
Containers

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

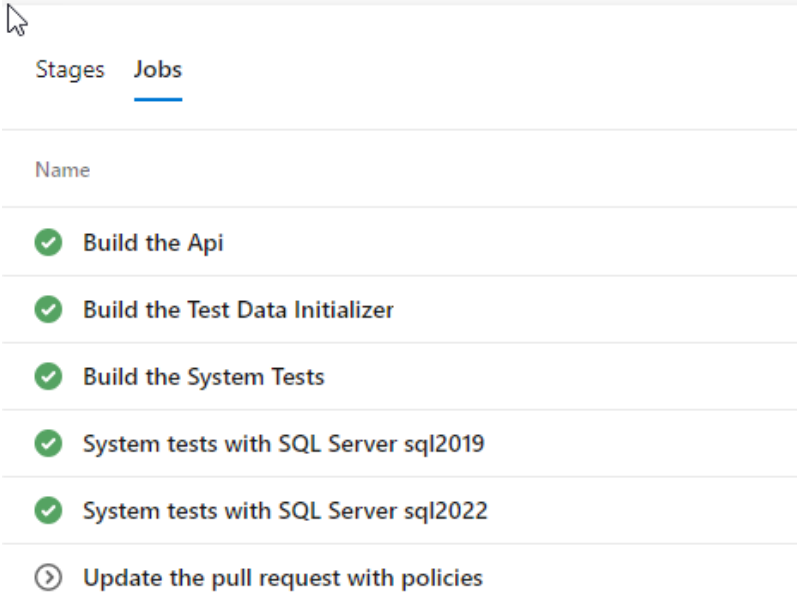

Demo
SQL Server in Container

Local Development Environment

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

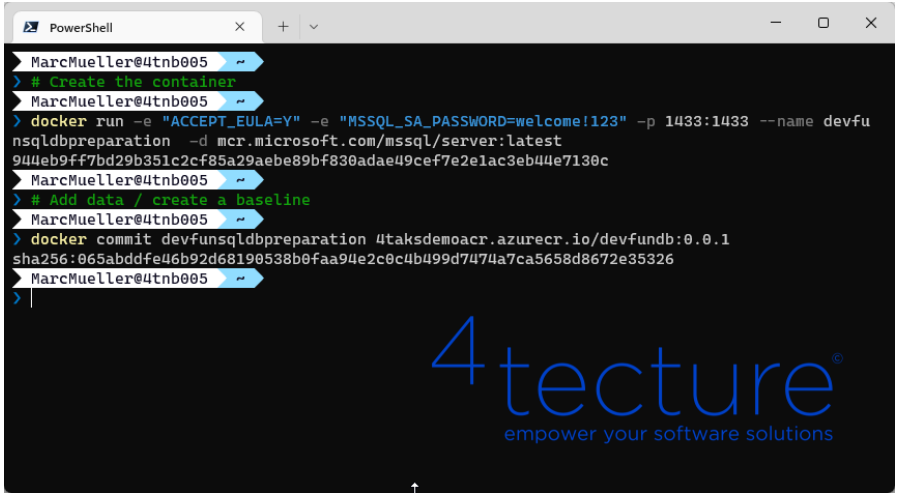
Test Multiple Database Versions

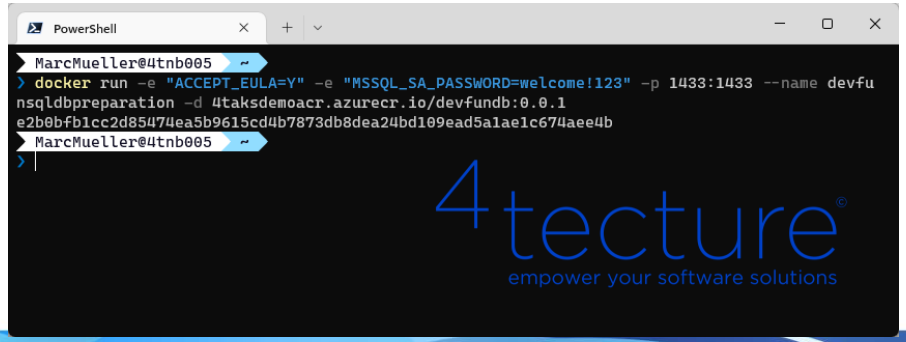
- Locally
- Azure Pipelines



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





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”

```
FROM mcr.microsoft.com/dotnet/sdk:6.0

RUN curl -sL https://aka.ms/InstallAzureCLIDeb | bash

RUN apt-get update && \
    apt-get install -y --no-install-recommends unzip && \
    rm -rf /var/lib/apt/lists/* && \
    wget -q -O /opt/sqlpackage.zip https://go.microsoft.com/fwlink/?linkid=2157202 \
    && unzip -qq /opt/sqlpackage.zip -d /opt/sqlpackage \
    && chmod +x /opt/sqlpackage/sqlpackage && rm -f /opt/sqlpackage.zip

RUN apt-get update && \
    apt-get install -y curl gnupg apt-transport-https && \
    curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add - && \
    sh -c 'echo "deb [arch=amd64] https://packages.microsoft.com/repos/microsoft-debian-stretch-prod stretch main" > /etc/apt/sources.list.d/microsoft.list' && \
    apt-get update && \
    apt-get install -y powershell && \
    curl -fsSL https://deb.nodesource.com/setup_16.x | bash - && \
    apt-get install -y nodejs

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"]
```

Chapter 8/8

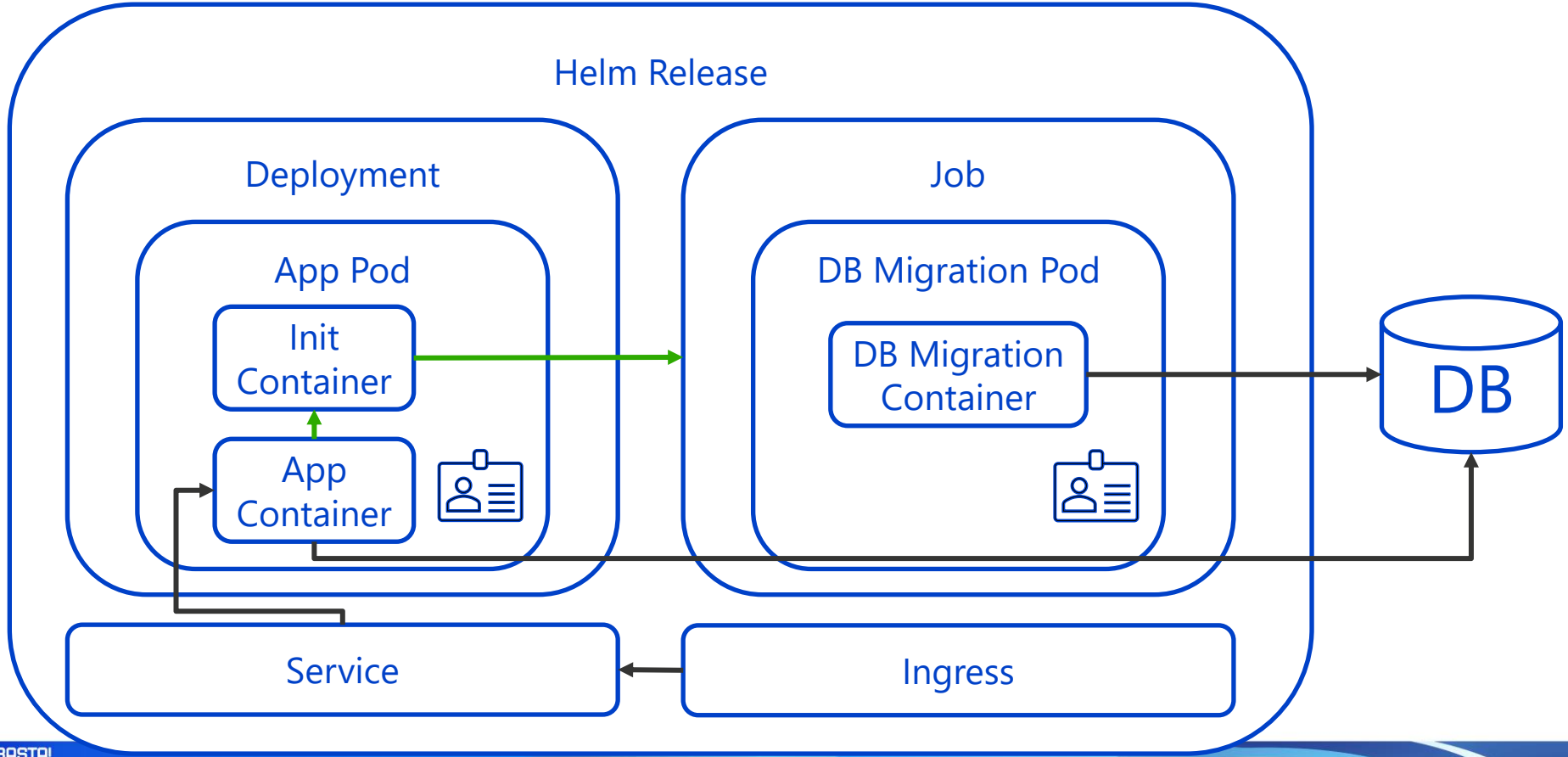
Autonomous Deployment

4tecture®
empower your software solutions

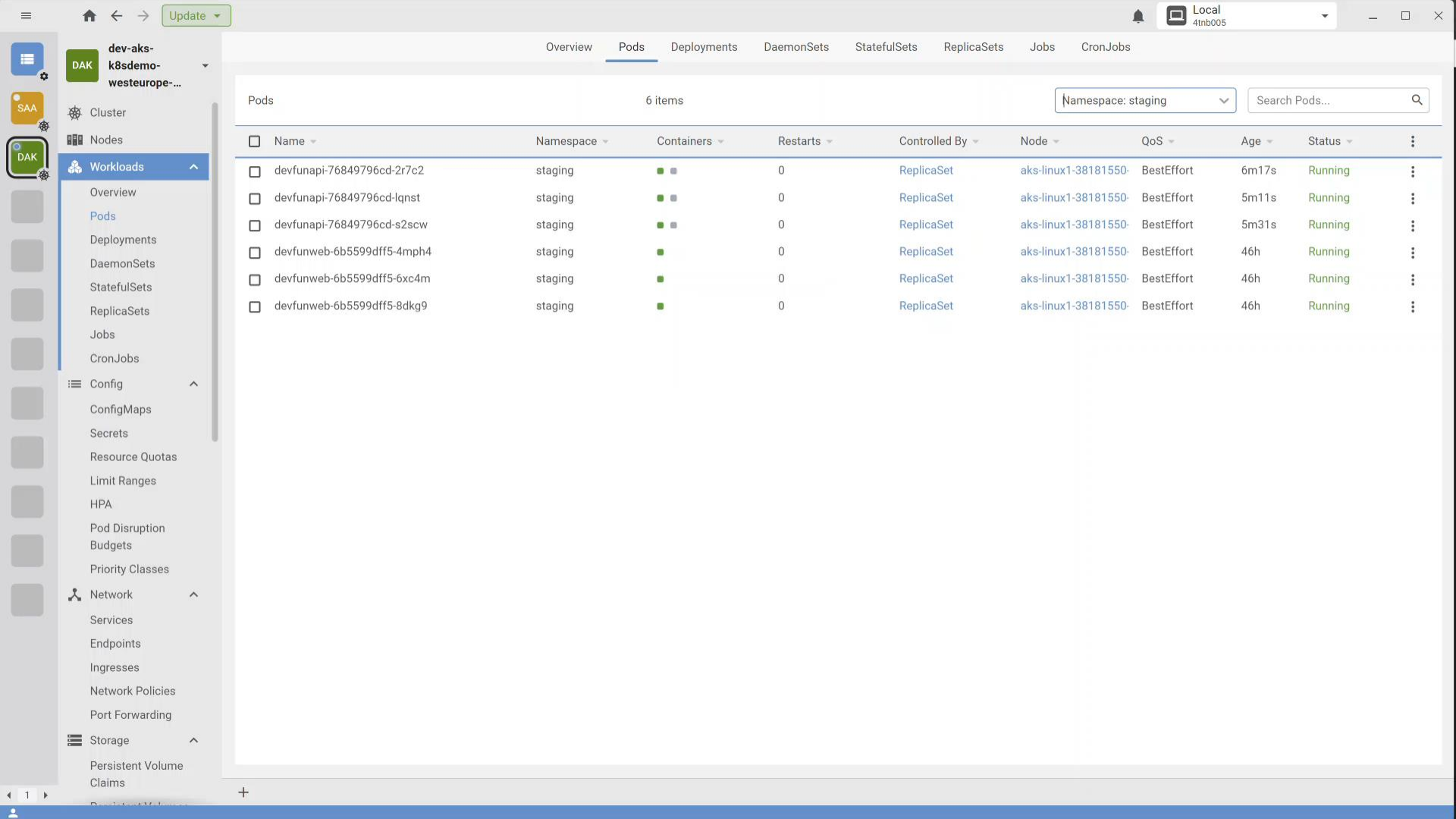
Autonomous Application Packages

- 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


Update
Local
4tnb005
dev-aks-k8sdemo-westeurope-...
Cluster
Nodes
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 0 ReplicaSet aks-linux1-38181550- BestEffort 6m17s Running
devfunapi-76849796cd-lqnst staging 0 ReplicaSet aks-linux1-38181550- BestEffort 5m11s Running
devfunapi-76849796cd-s2scw staging 0 ReplicaSet aks-linux1-38181550- BestEffort 5m31s Running
devfunweb-6b5599dff5-4mph4 staging 0 ReplicaSet aks-linux1-38181550- BestEffort 46h Running
devfunweb-6b5599dff5-6xc4m staging 0 ReplicaSet aks-linux1-38181550- BestEffort 46h Running
devfunweb-6b5599dff5-8dkg9 staging 0 ReplicaSet aks-linux1-38181550- BestEffort 46h Running

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:

4tecture GmbH
Industriestrasse 25
CH-8604 Volketswil

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

Marc Müller
Principal Consultant

www.powerofdevops.com

4tecture
empower your software solutions

ProAuth

ReaFx


4tecture©
empower your software solutions