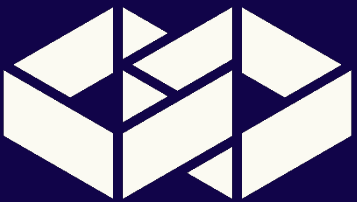


Code First, Review Later: Making EF Core Work for DBAs

Tonie Huizer

Consultant



promicro

Tonie Huizer

He/him

Freelance

DevOps consultant



 [linkedin.com/in/toniehuizer](https://www.linkedin.com/in/toniehuizer)

 github.com/promicroNL/events

 www.promicro.nl





SeaQL.nl

Data, DevOps & SQL @ Sea!

17 and 18 April 2026, Ouddorp NL

The agenda for today

- The good and the bad of EF Code First
- Three Hybrids workflow for working with EF Core
 - With demos!
- Wrap-up: Take aways + Q&A

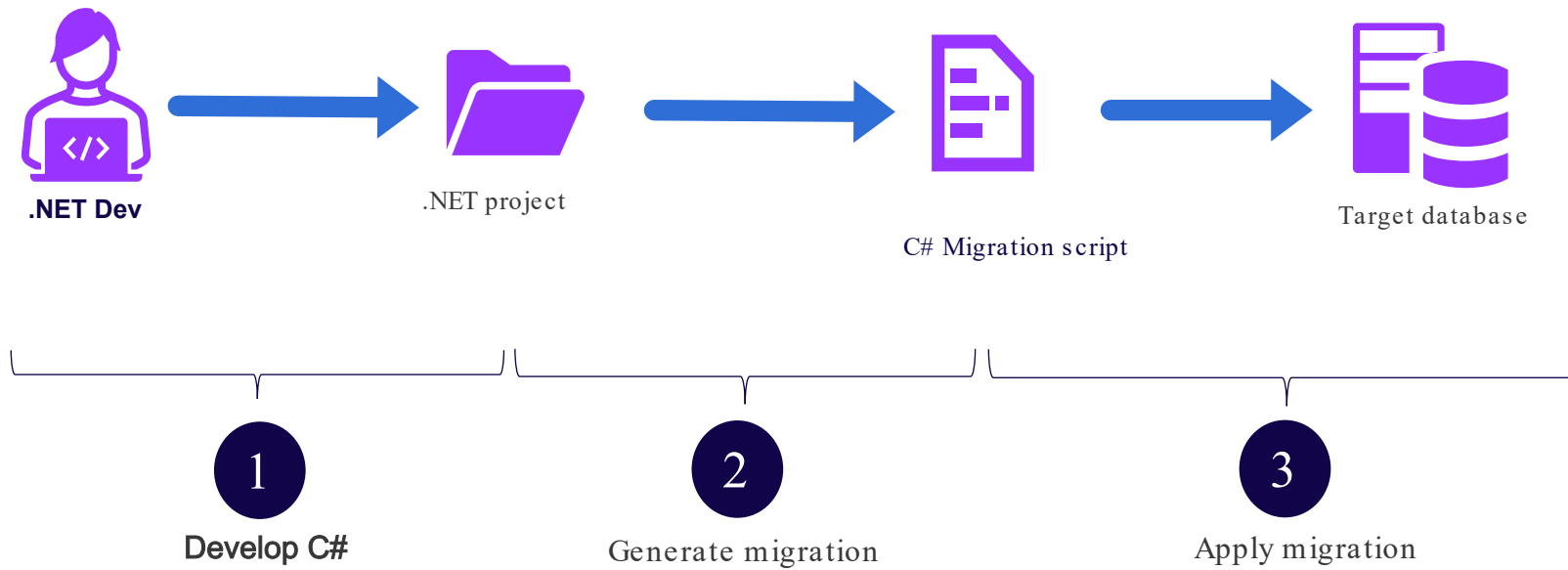
The good and the bad of EF Code First

Click to edit Master title style



EF Core workflow

EF Core workflow



“That's all there is to it – your application is ready to run on your new database, and you didn't need to write a single line of SQL.”

Source: Microsoft documentation

Note that this way of applying migrations is ideal for local development but... is less suitable for production environments.

Migrations Overview - EF Core | Microsoft Learn

https://learn.microsoft.com/en-us/ef/core/...

Create your database and schema

At this point you can have EF create your database and create your schema from the migration. This can be done via the following:

.NET CLI

Visual Studio

.NET CLI

```
dotnet ef database update
```

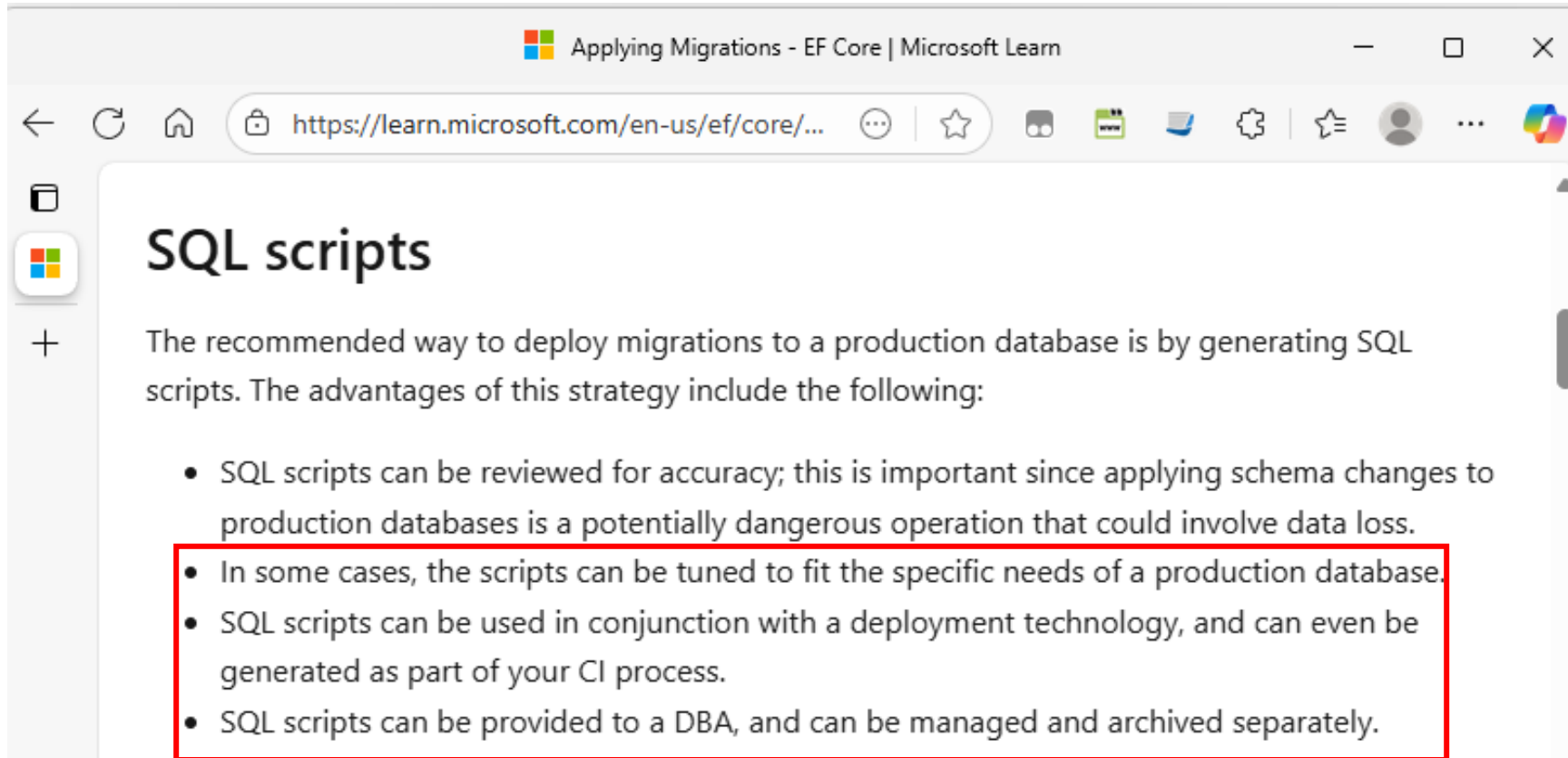
Copy

That's all there is to it - your application is ready to run on your new database, and you didn't need to write a single line of SQL. Note that this way of applying migrations is ideal for local development, but is less suitable for production environments - see the [Applying Migrations page](#) for more info.

Note that this way of applying migrations is ideal for local development but... is less suitable for production environments.

**“The recommended way to
deploy migrations to a
production database is by
generating SQL scripts.”**

Source: Microsoft documentation



The screenshot shows a web browser window with the title 'Applying Migrations - EF Core | Microsoft Learn'. The address bar shows the URL 'https://learn.microsoft.com/en-us/ef/core/...'. The page content is titled 'SQL scripts' and includes a paragraph and a bulleted list. The list is highlighted with a red rectangular box.

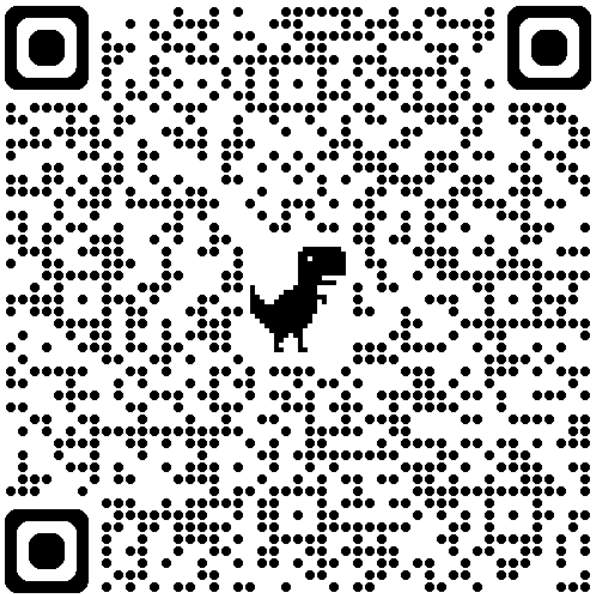
SQL scripts

The recommended way to deploy migrations to a production database is by generating SQL scripts. The advantages of this strategy include the following:

- SQL scripts can be reviewed for accuracy; this is important since applying schema changes to production databases is a potentially dangerous operation that could involve data loss.
- In some cases, the scripts can be tuned to fit the specific needs of a production database.
- SQL scripts can be used in conjunction with a deployment technology, and can even be generated as part of your CI process.
- SQL scripts can be provided to a DBA, and can be managed and archived separately.

Source: Microsoft documentation

“When it’s time to move from development and QA to production, letting EF Core update the database for you is generally not the best approach.”



Julie Lerman @ The Data Farm

The situation with EF

- Developers write C# and let EF handle the SQL
- These migrations aren't suitable for production deployments.
- DBAs struggle with auto-generated SQL
- Limited visibility into (schema) changes

(the DBAs) Limited visibility

```
56 public class Customer
57 {
58     public int Id { get; set; }
59     public string? FirstName { get; set; }
60     public string? LastName { get; set; }
61     public DateTime DateOfBirth { get; set; }
62 - public string? Address { get; set; }
63 }
64
```

```
1 CREATE TABLE [dbo].[Customers]
2 (
3     [Id] [int] NOT NULL IDENTITY(1, 1),
4     [FirstName] [nvarchar] (max) NULL,
5     [LastName] [nvarchar] (max) NULL,
6     [DateOfBirth] [datetime2] NOT NULL,
7 - [Address] [nvarchar] (max) NULL
8 )
9 GO
10 ALTER TABLE [dbo].[Customers] ADD CONSTRAINT [PK_Customers] PRIMARY KEY CLUSTERED ([Id])
11 GO
12
```

```
56 public class Customer
57 {
58     public int Id { get; set; }
59     public string? FirstName { get; set; }
60     public string? LastName { get; set; }
61     public DateTime DateOfBirth { get; set; }
62 + public string? Address { get; set; }
63 + public string? Remark { get; set; }
64 +
65 }
```

```
1 CREATE TABLE [dbo].[Customers]
2 (
3     [Id] [int] NOT NULL IDENTITY(1, 1),
4     [FirstName] [nvarchar] (max) NULL,
5     [LastName] [nvarchar] (max) NULL,
6     [DateOfBirth] [datetime2] NOT NULL,
7 + [Address] [nvarchar] (max) NULL,
8 + [Remark] [nvarchar] (max) NULL
9 )
10 GO
11 ALTER TABLE [dbo].[Customers] ADD CONSTRAINT [PK_Customers] PRIMARY KEY CLUSTERED ([Id])
12 GO
13
```


Hybrids workflow

For working with

Click to edit Master title style

EF Core





October 2019

Volume 34 Number 10

[Data Points]

Hybrid Database Migrations with EF Core and Flyway

By Julie Lerman



During development, it's common to use EF Core's migration commands to not only



Flyway Explained

- Flyway “versions” the database
- Flyway applies SQL scripts to the database:
 - Versioned scripts (V) run once
 - Repeatable scripts (R) run on every upgrade
 - Undo scripts (U) allow rollback
- Tracks changes in flyway_schema_history
- Flyway works with multiple RDMSs

```
migrations/  
V1__initial_schema.sql  
V2__add_customer_table.sql  
R_refresh_views.sql  
U2__add_customer_table.sql
```

C# migration vs SQL migration

```
public partial class AddedAddress : Migration
{
    /// <inheritdoc />
    protected override void Up (MigrationBuilder
migrationBuilder)
    {
        migrationBuilder.AddColumn<string>(
            name: "Address",
            table: "Customers",
            type: "nvarchar(max)",
            nullable: false,
            defaultValue: "");
    }
    ...
}
```

```
ALTER TABLE [Customers] ADD [Address]
varchar(max) NOT NULL DEFAULT N'';
GO
```

```
INSERT INTO [__EFMigrationsHistory]
([MigrationId], [ProductVersion])
VALUES (N'20230110134534_AddedAddress',
N'7.0.1');
GO
...
```

C# migration vs SQL migration

```
public partial class AddedAddress : Migration
{
    /// <inheritdoc />
    protected override void Up (MigrationBuilder
migrationBuilder)
    {
        migrationBuilder.AddColumn<string>(
            name: "Address",
            table: "Customers",
            type: "nvarchar(max)",
            nullable: false,
            defaultValue: "");
    }
    ...
}
```

```
ALTER TABLE [Customers] ADD [Address]
varchar(max) NOT NULL DEFAULT N'';
GO

INSERT INTO [__EFMigrationsHistory]
([MigrationId], [ProductVersion])
VALUES (N'20230110134534_AddedAddress',
N'7.0.1');
GO
...
```

SQL migration vs Flyway migration

C# migration file name:

20230110134534_AddedAddress

Flyway migration file name:

V003_20230110134534__AddedAddress

SQL migration vs Flyway migration

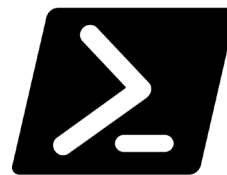
C# migration file name:

20230110134534_AddedAddress

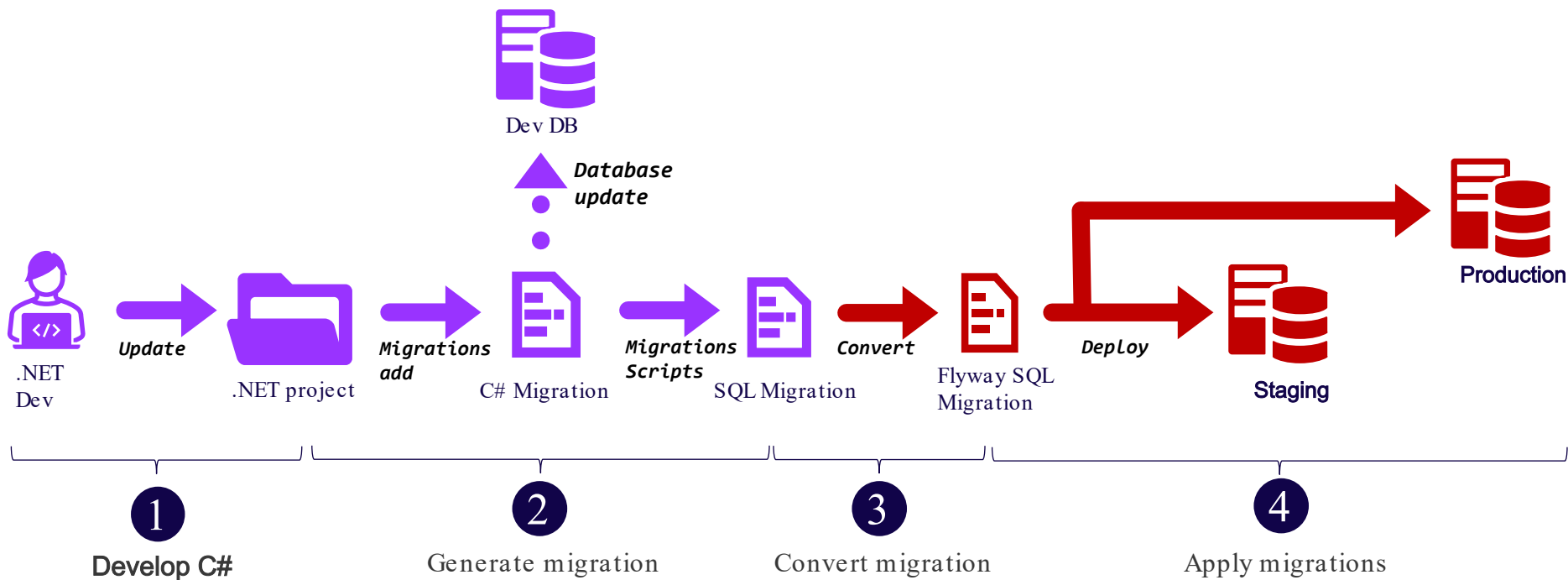
Flyway migration file name:

V003_20230110134534__AddedAddress

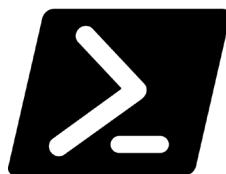
Simple Hybrid workflow



Simple Hybrid workflow

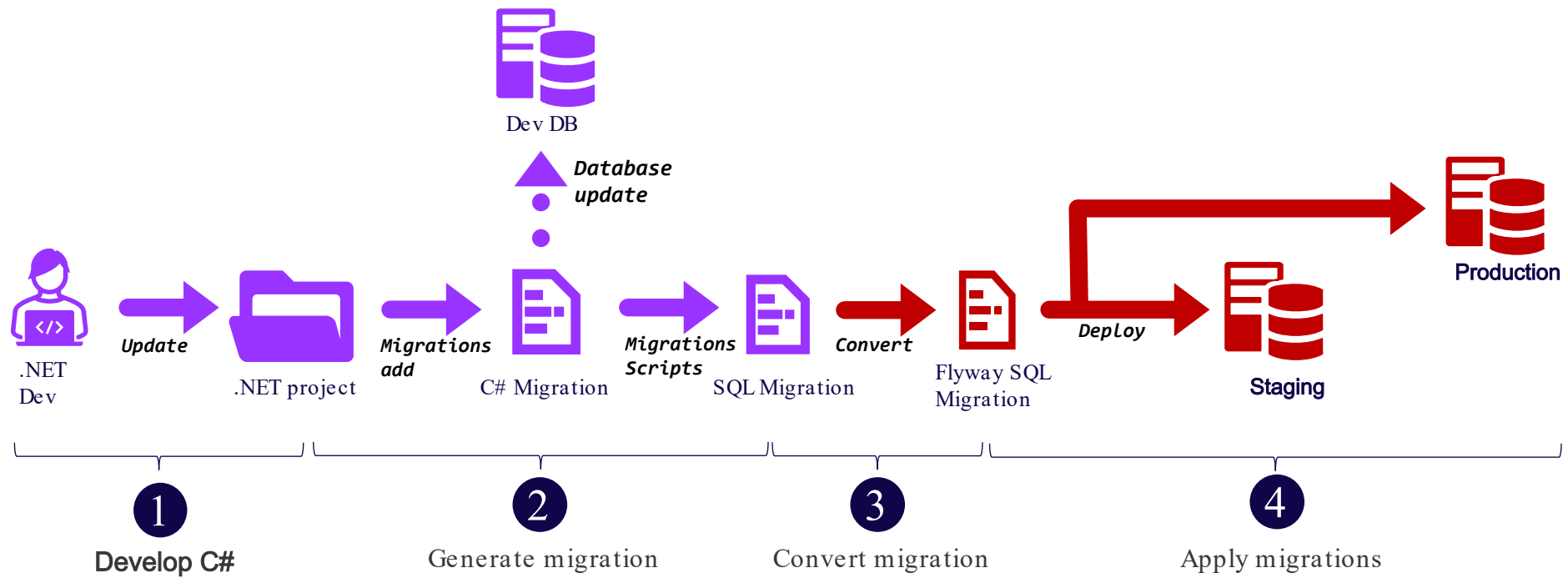


EF-managed Development

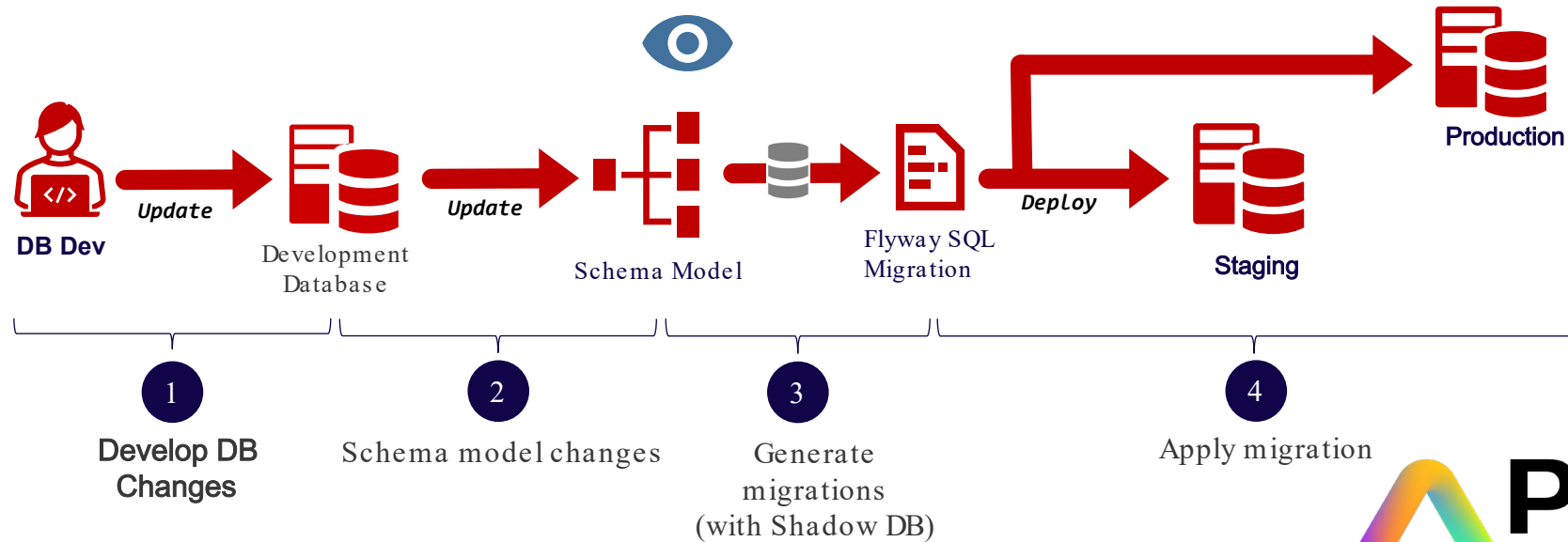


Demo time

Simple Hybrid workflow

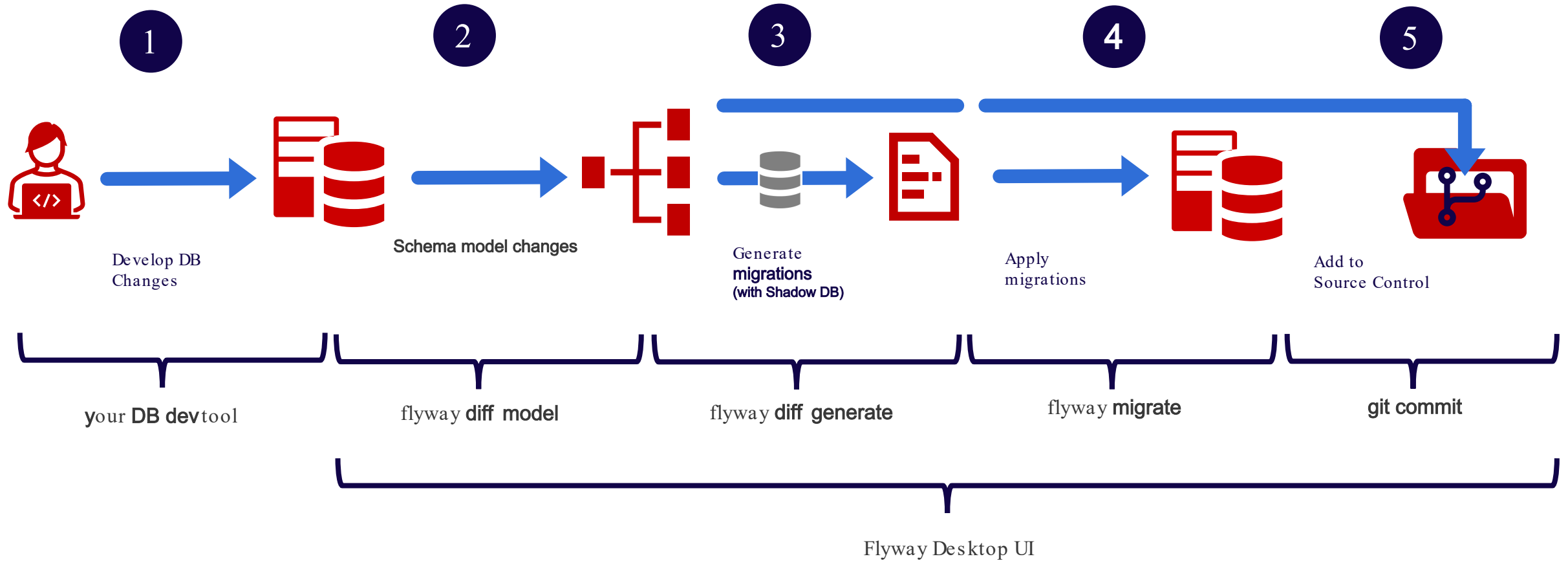


Flyway workflow Desktop

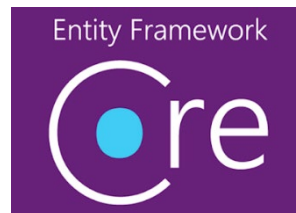




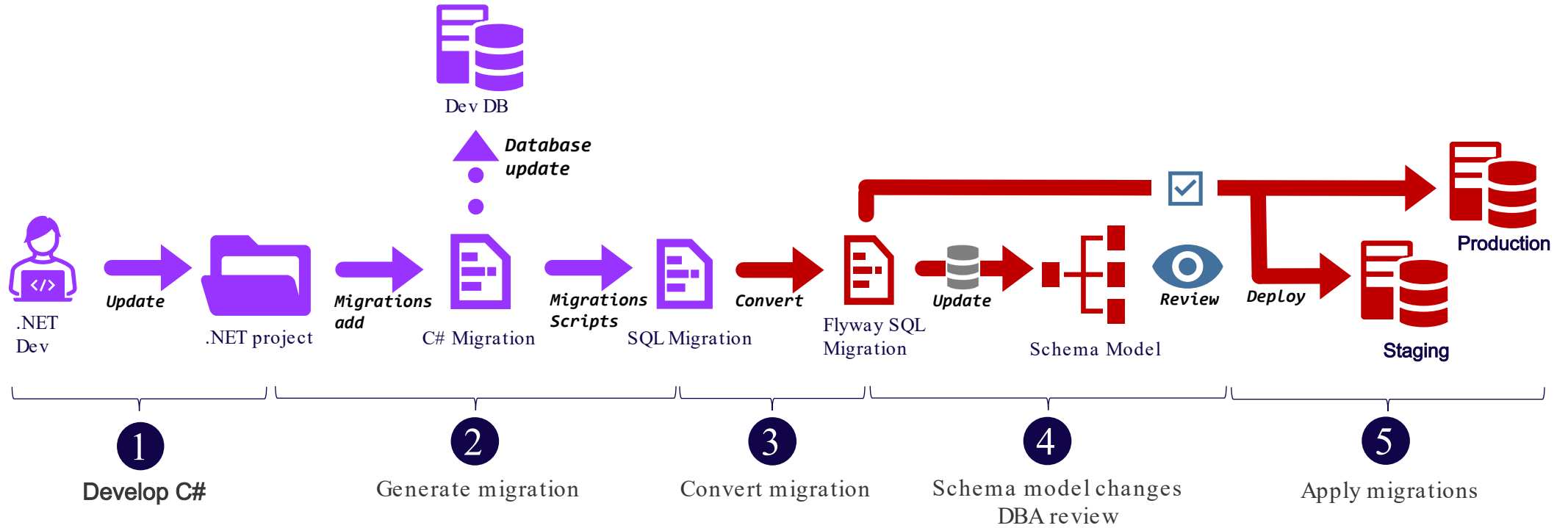
Migration based Flyway Desktop Workflow via CLI



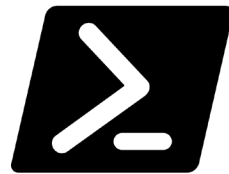
Inverted Hybrid workflow



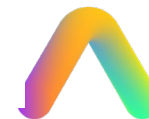
Inverted Hybrid workflow



A
EF-managed Development



B
Flyway-managed Deployments



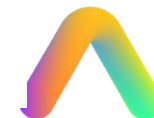
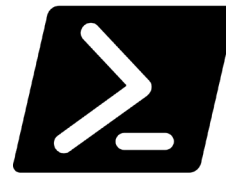
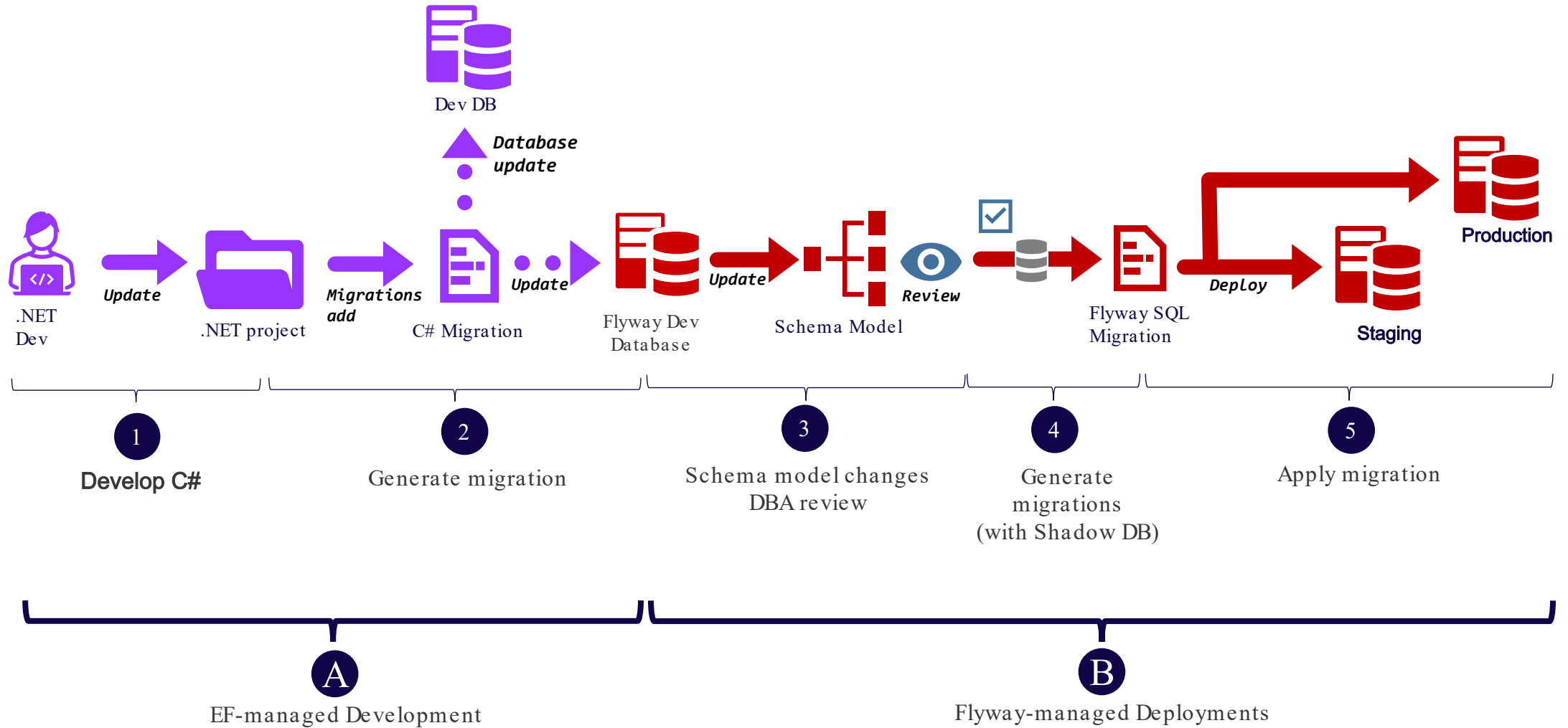
PASS
Data Community Summit

Demo time

Full Hybrid workflow



Hybrid workflow



PASS
Data Community Summit

Demo time

Comparison

Aspect	EF Core Code First Workflow	Simple Hybrid Workflow	Inverted Hybrid Workflow	Hybrid Workflow
Schema Model Review	✗	✗	✓	✓
Converted & Linked migrations	✗	✓	✓	✗
Generated migrations	✗	✗	✗	✓
Strength	Fastest to deploy; minimal tooling	Faster than DBA-reviewed hybrids	DBA catches issues before production	DBA catches issues before production; Cleanest SQL output
Weakness	No staging/prod separation; no DBA check	No DBA quality check	Two-step tooling adds overhead	Two-step tooling adds overhead

Next steps: Automation

- Choose your workflow
- Create CI trigger on the EF changes
- Use pipelines, but first...
 - Start local to automate
 - Use verbose logging
 - Avoid inline PowerShell

Demo time

Wrap-up: Take aways & Q&A

[Click to edit Master title style](#)



Take aways

- It's the process not the tool
- Automate the chosen hybrid workflow
- Generate Migrations early to catch issues early
- Always include undo scripts

Questions?



Thank you

I appreciate the time you spent with me.
Please reach out if you have any questions!

Tonie Huizer

 linkedin.com/in/toniehuizer

 github.com/promicroNL/events

 www.promicro.nl

Your feedback is important to us



Evaluate this session at:

passdatacommunitysummit.com/evaluations