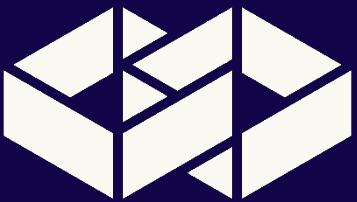


Code First, Review Later: Making EF Core Work for DBAs

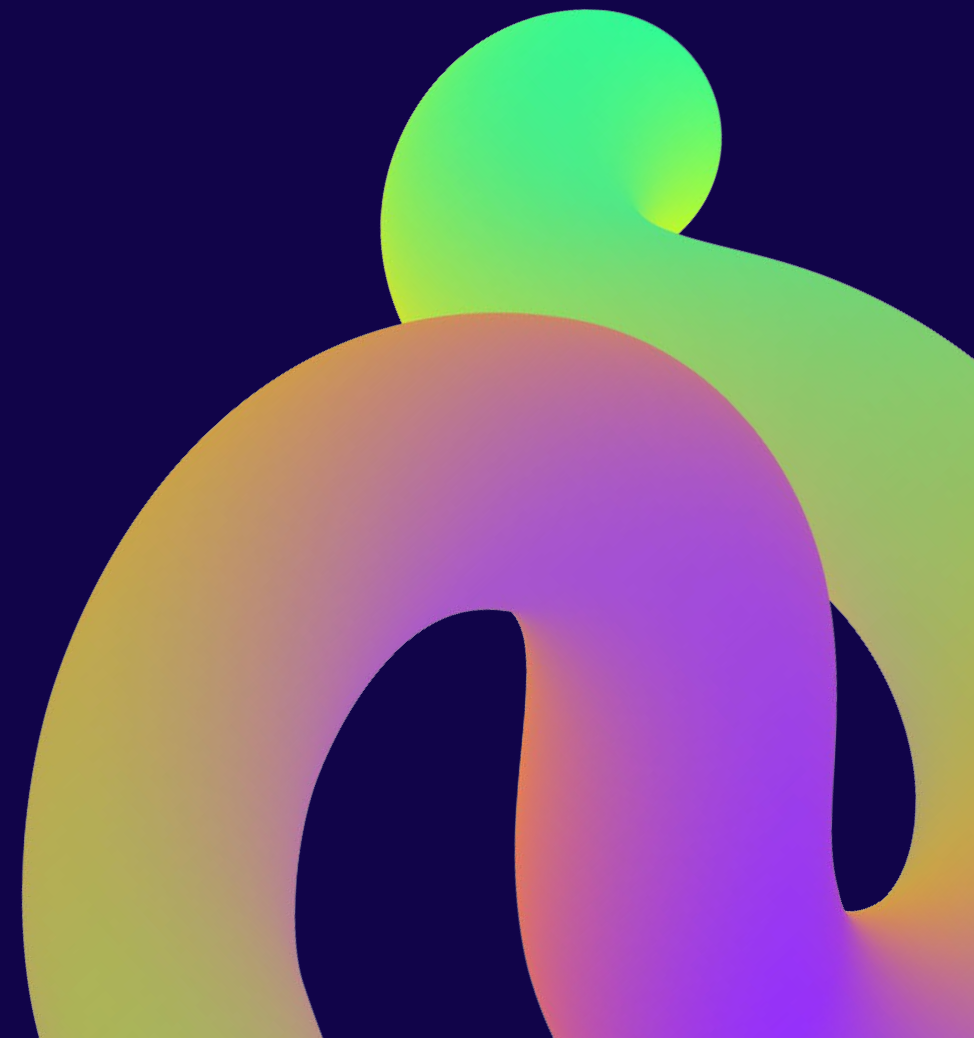
Tonie Huizer

Consultant



promicro

ON TOUR | NEW YORK
APASS



Tonie Huizer

He/him

Freelance

DevOps consultant



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

 github.com/promicroNL/events

 www.promicro.nl



The agenda for today

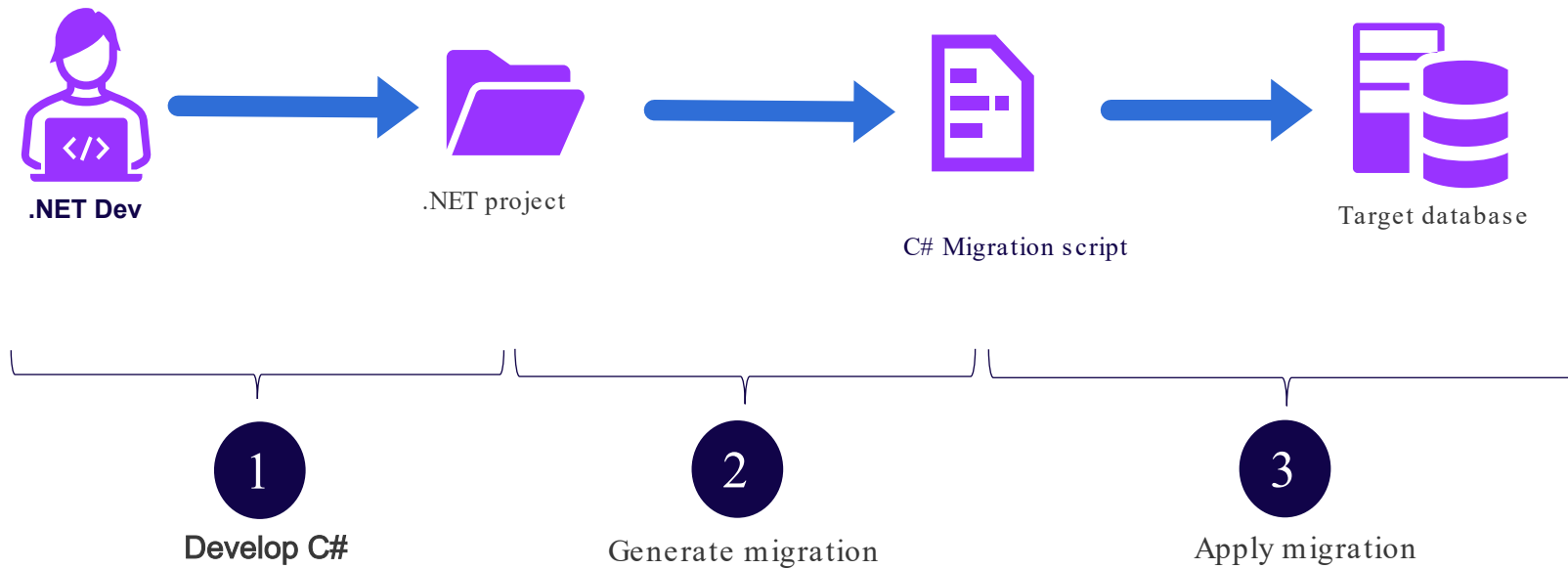
- The good and the bad of EF Code First
- Three Hybrids workflow for working with EF Core
 - With demos!
- Wrap-up: Take aways + Q&A

The good and the bad of EF Code First



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.”

Someone @ Microsoft Docs

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
```

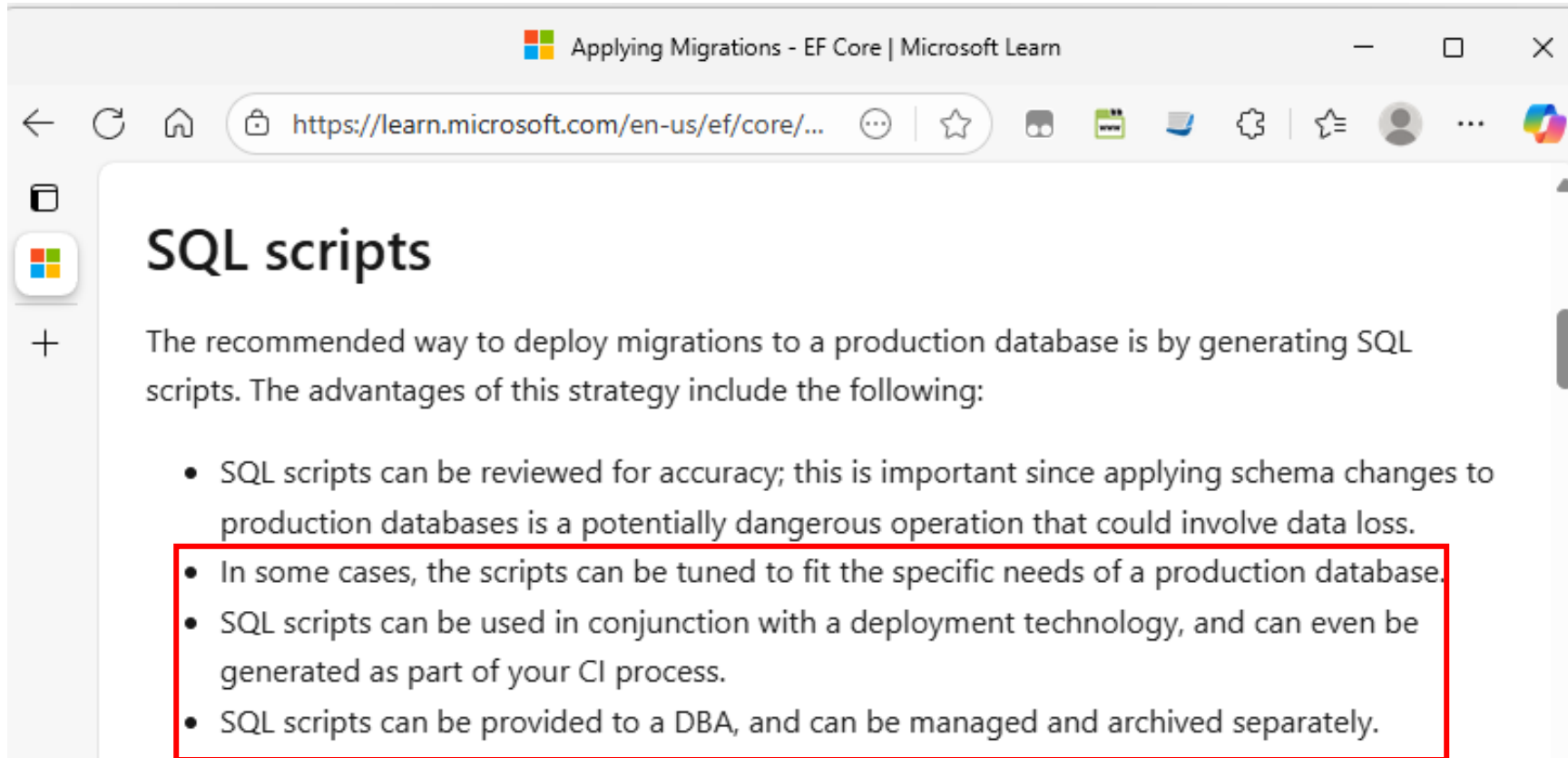
Copy

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.”**

Someone @ Microsoft Docs



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: 'The recommended way to deploy migrations to a production database is by generating SQL scripts. The advantages of this strategy include the following:'. Below this is a bulleted list of four points. The second, third, and fourth points are enclosed in a red rectangular box.

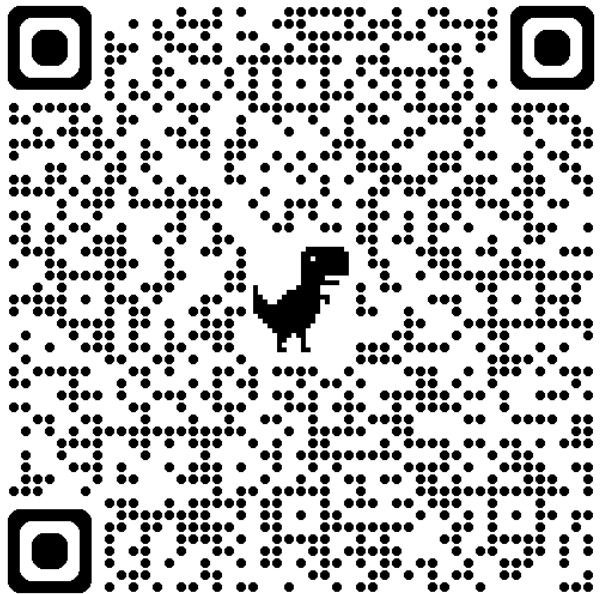
SQL scripts

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.

Someone @ Microsoft Docs

“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

Hybrids workflow For working with 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

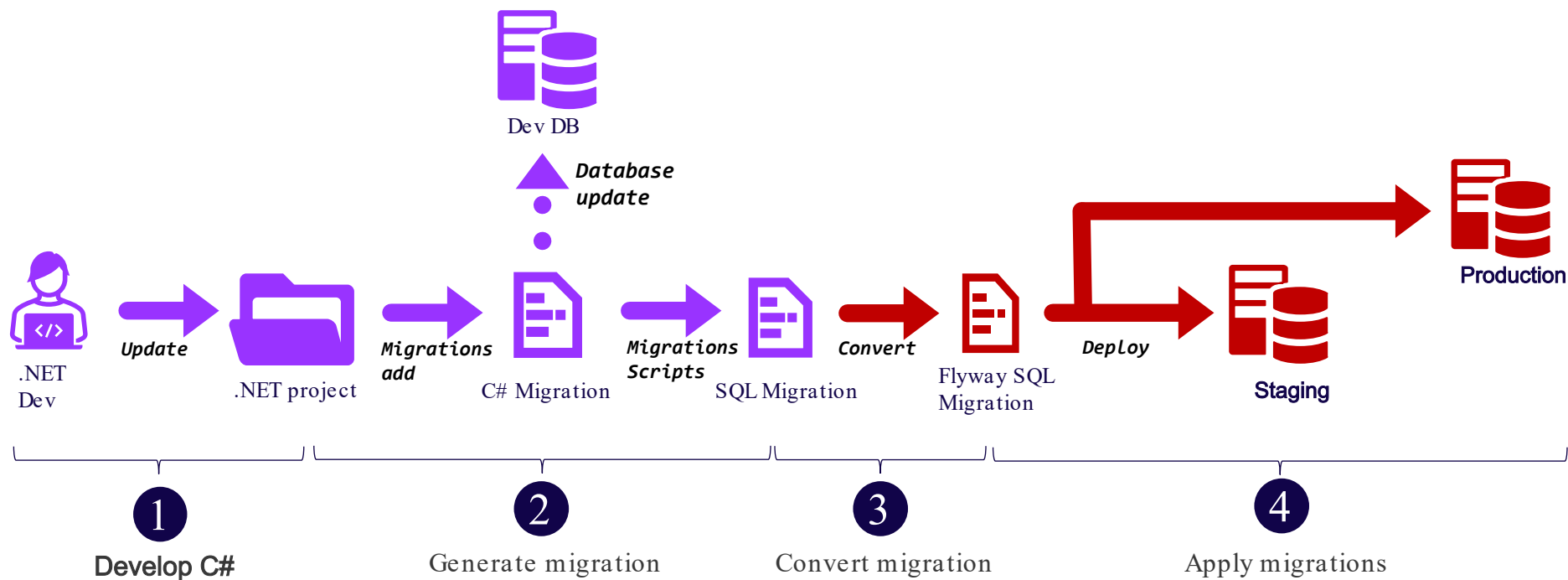
- Flyway “versions” the database
- Flyway applies SQL scripts to the database:
 - Versioned scripts (V) run once
 - Repeatable scripts (R) run on every upgrade
 - Undo scripts (U) allow rollback
- Tracks changes in flyway_schema_history
- Flyway works with multiple RDMSs

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

Simple Hybrid workflow



Simple Hybrid workflow



A
EF-managed Development

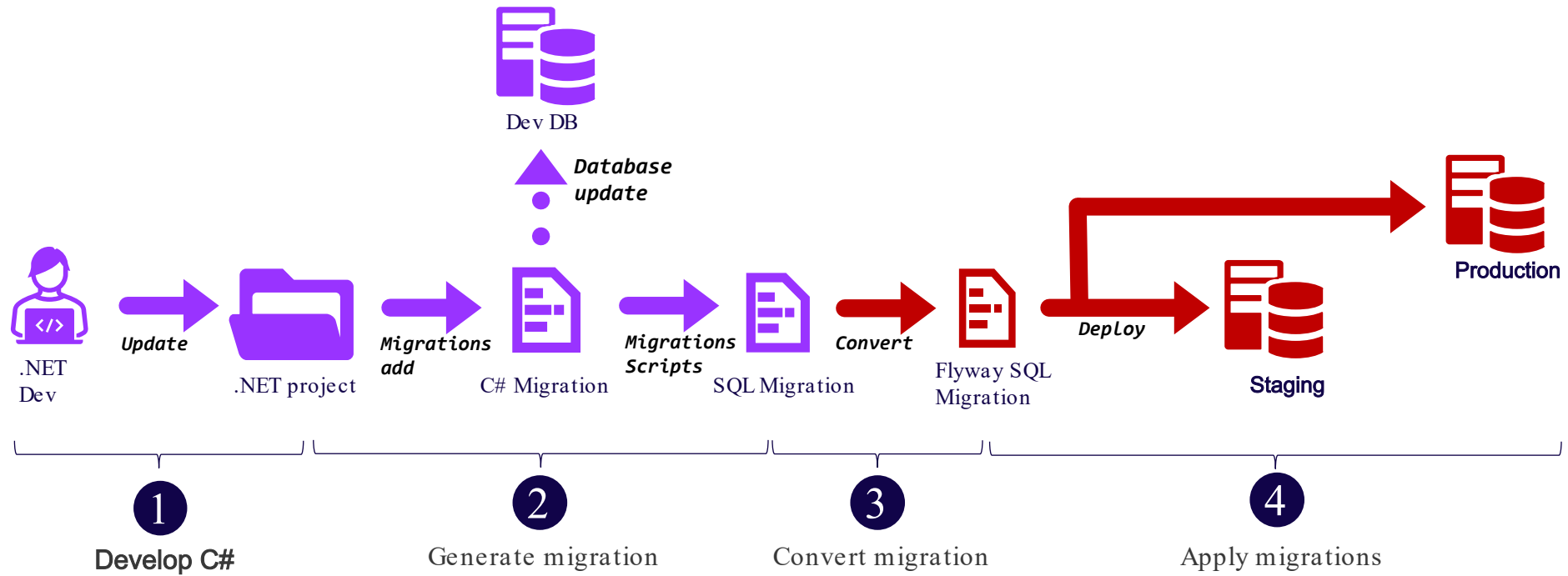


B
Flyway-managed Deployments

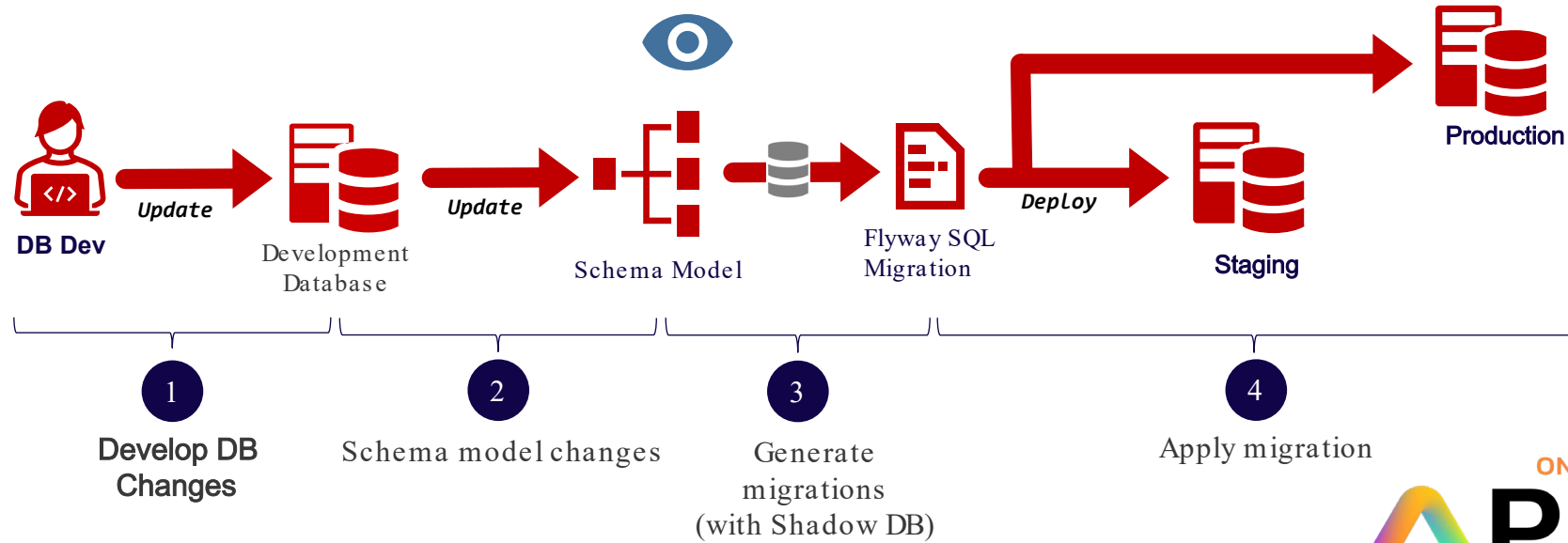


Demo time

Simple Hybrid workflow

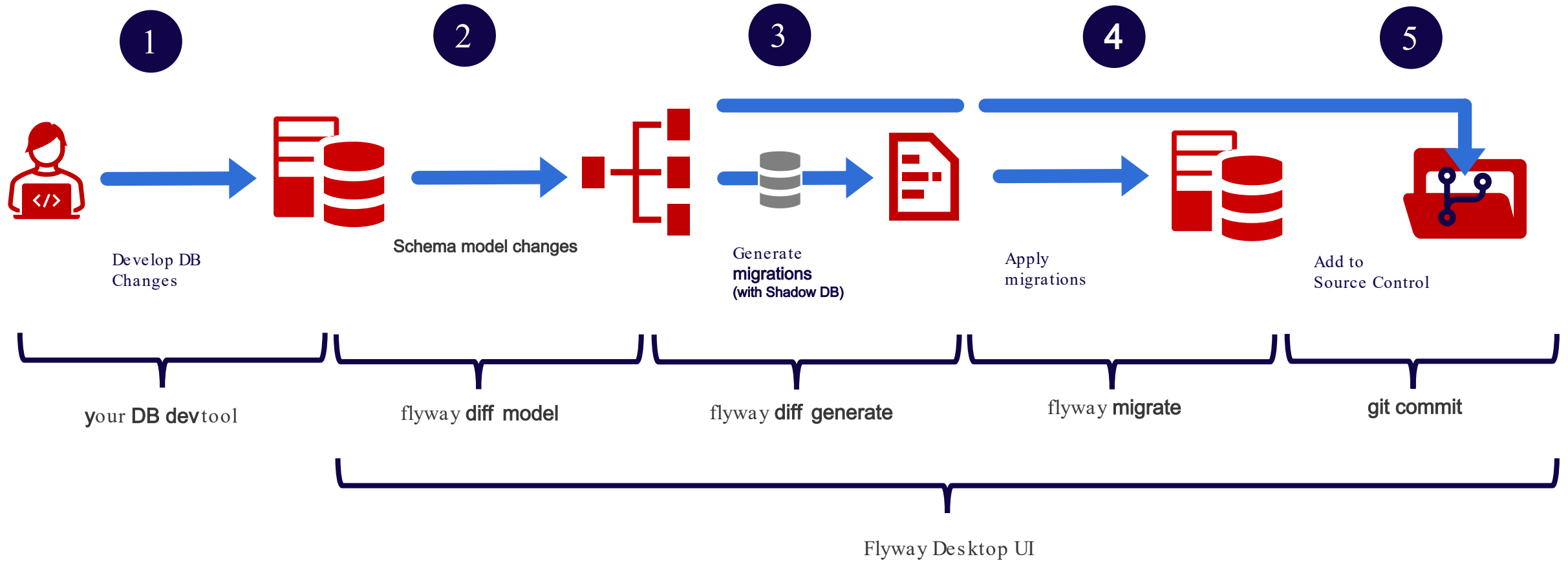


Flyway workflow Desktop

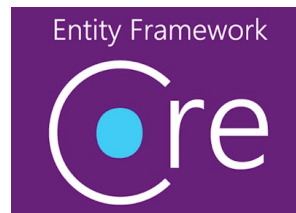




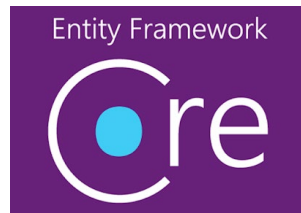
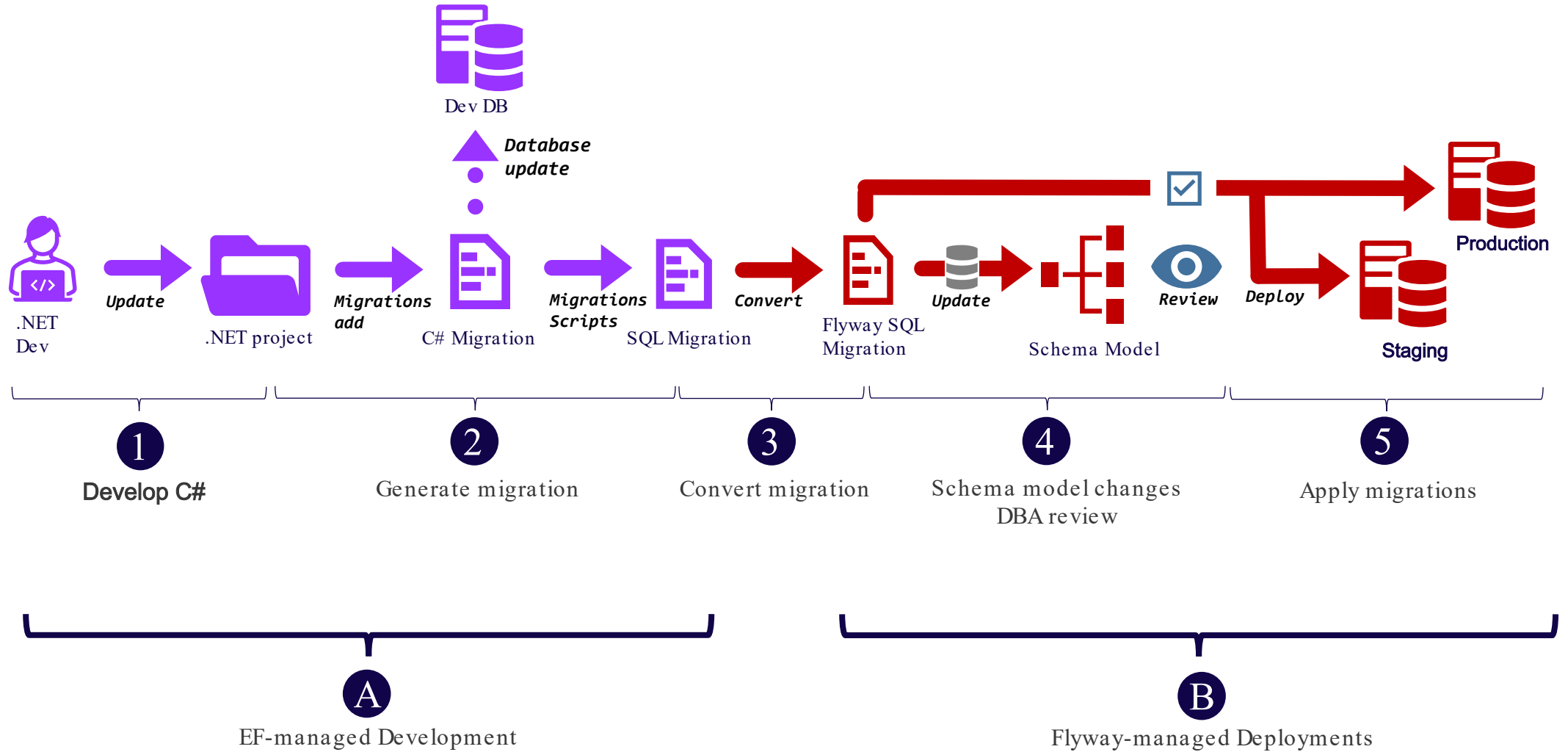
Migration based Flyway Desktop Workflow via CLI



Inverted Hybrid workflow

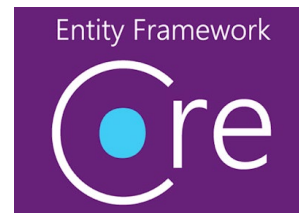


Inverted Hybrid workflow

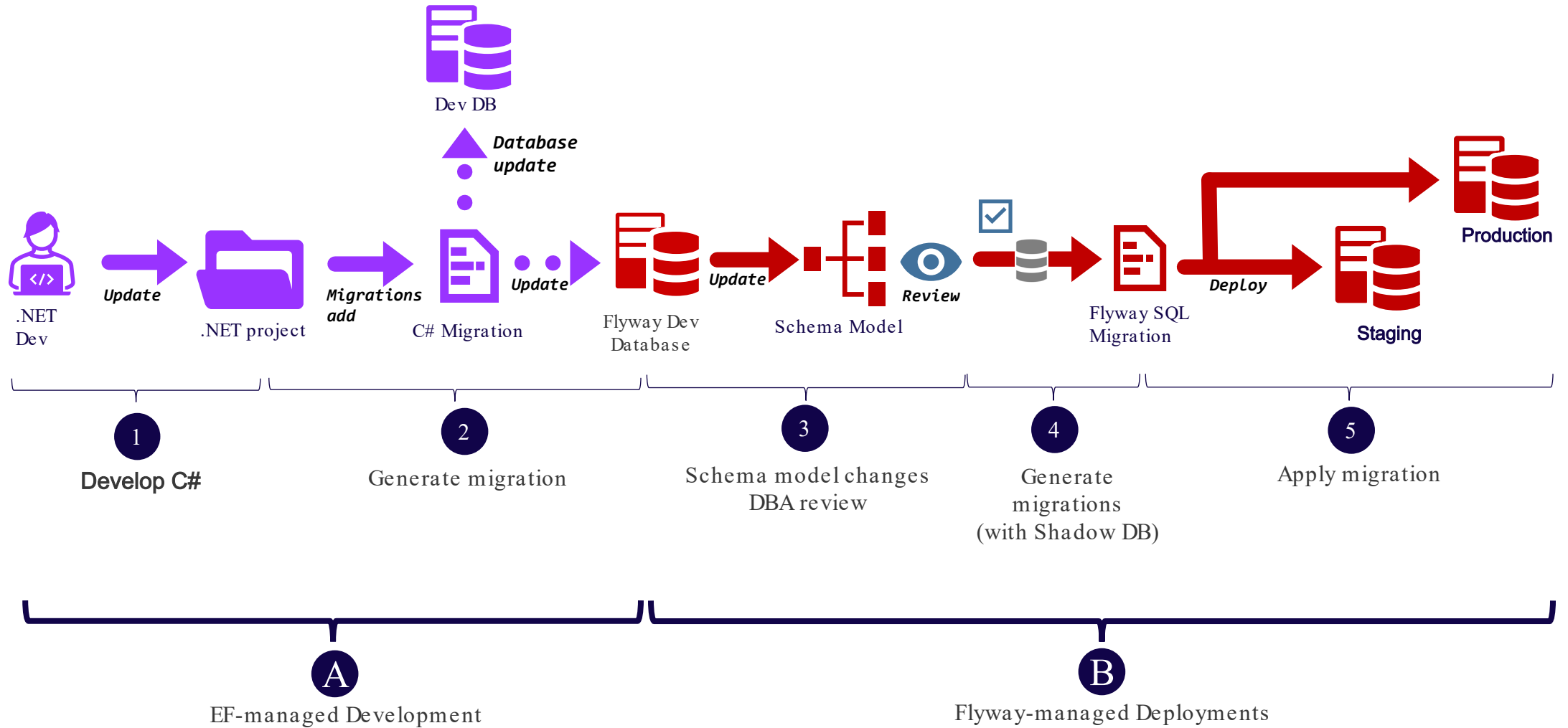


Demo time

Full Hybrid workflow



Hybrid workflow



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

Wrap-up: **Take aways & Q&A**



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