# Controlling Database Creation and Schema Changes with Migrations

**Julie Lerman**
MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK

@julielerman thedatafarm.com

# Module Overview

Overview of EF Core Migrations API

Create and inspect a migration file

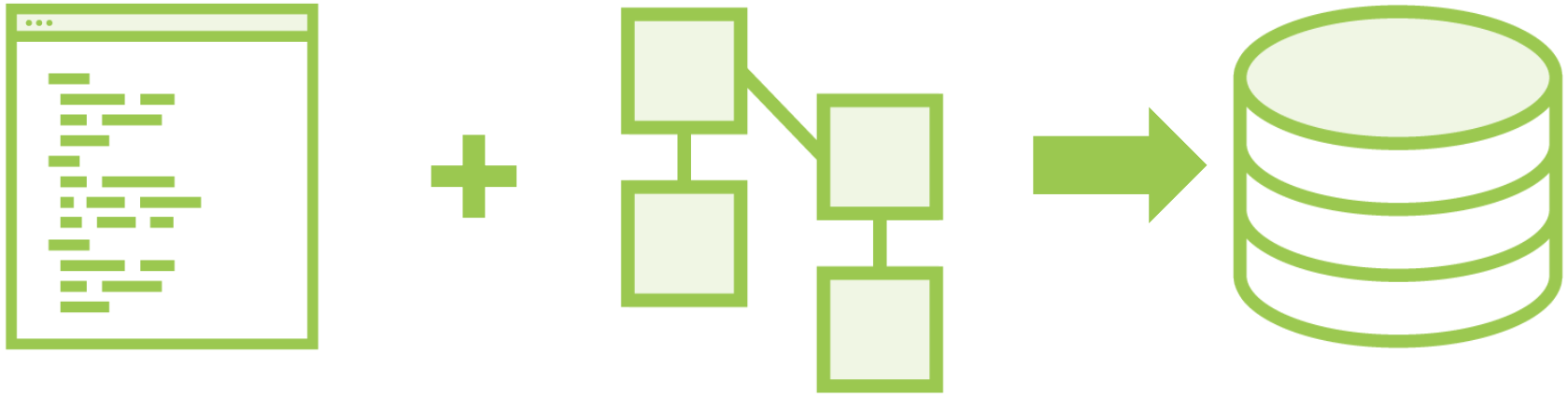
Using EF Core Migrations to create a database or database scripts

Reverse engineer and existing database into classes and DbContext

# Understanding EF Core Migrations

# EF Core Basic Migrations Workflow

# Mapping Your Data Model to the Database



EF Core Migrations

are

**source-control friendly**

# Adding Your First Migration

# NuGet Packages for Migrations

## Migrations Commands

Powershell: Microsoft.EntityFrameworkCore.Tools
dotnet CLI: Microsoft.EntityFrameworkCore.Tools.Dotnet

## Migrations APIs

Microsoft.EntityFrameworkCore.Design
(installed as a dependency of Tools)

# Runtime Needed to Run Migration Commands

## What if DbContext is in a class library project?

**Class Library**
Add Tools package
Default PMC project

**Executable Project**
Make it startup project
Reference class library
Add Design package

**Class Library**
Default PMC project

**Executable Project**
Make it startup project
Reference class library
Add Tools package
(Design comes in via NuGet dependency)

**Advanced Setup**

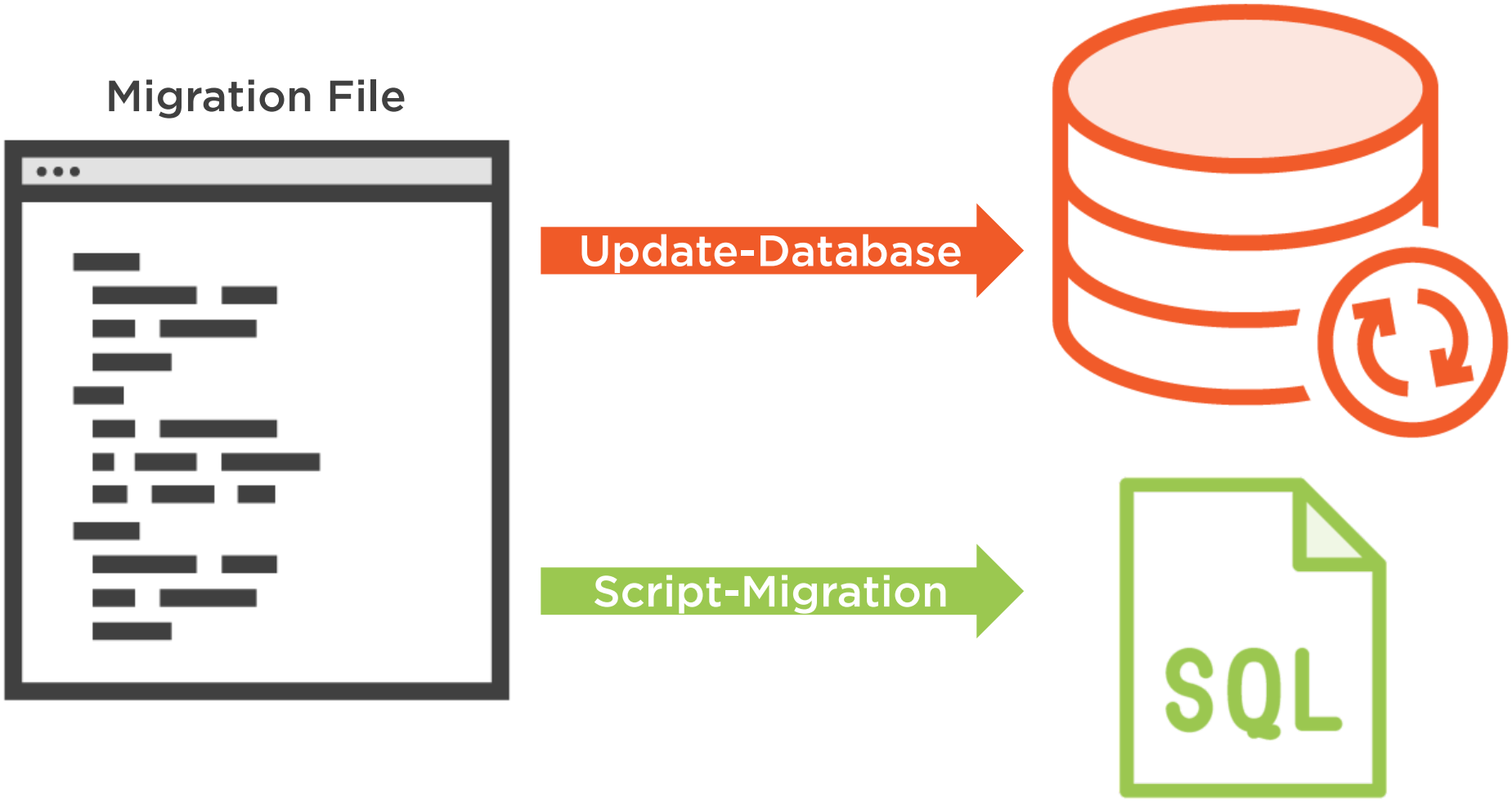
**Class Library**
Default PMC project
Mods to csproj file
Add Tools package
(Design comes in via NuGet dependency)

**Executable Project**
Not needed

# Inspecting Your First Migration

# Using Migrations to Script or Directly Create the Database

# Applying Migrations



# Migrations Recommendation

**Development database**

update-database

**Production database**

script-migration

# What If Database Does Not Exist?

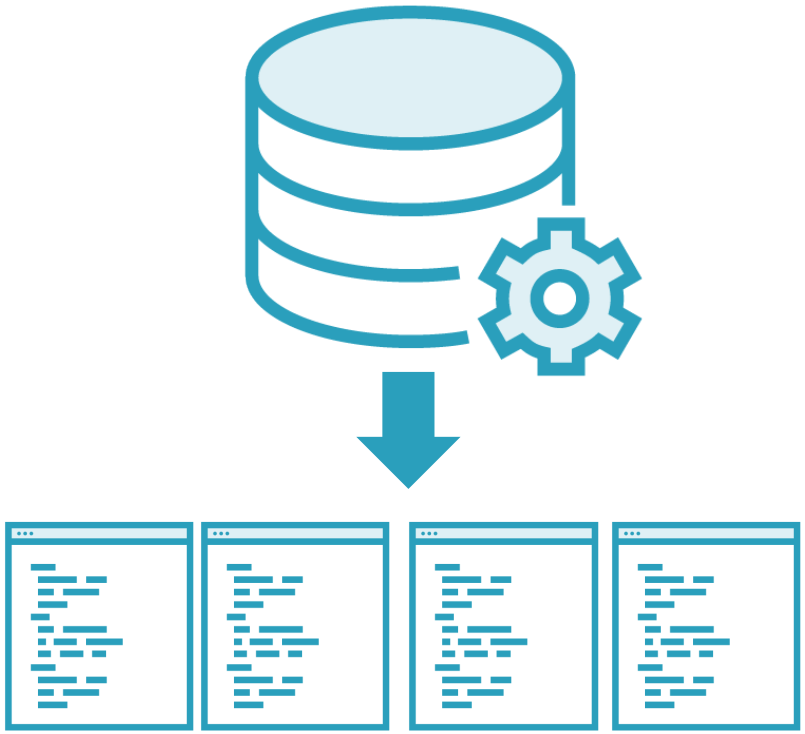
## update-database

API's internal code will create the database

## script-migration

You must create the database before running the script

# Reverse Engineering an Existing Database



Create DbContext & classes from database

Updating model is not currently supported

Transition to migrations is not pretty ... look for helpful link in resources

PowerShell command: scaffold-dbcontext

# How EF Core Determines Mappings to DB

## Conventions

Default assumptions

```
property name=column name
```

## Override with Fluent Mappings

Apply in DbContext using Fluent API

```
modelBuilder.Entity<Quotes>()
  .Property(q => q.Text)
  .HasColumnName("Line");
```

## Override with Data Annotations

Apply in entity

```
[Column("Line")]
public string Text{get;set;}
```

# Review

- Workflow of how EF Core determines database schema
- Where Migrations API fits in
- PowerShell or CLI commands for creating and executing migrations
- Explored API used in a migrations file
- Used migrations commands to generate script or create a new database directly
- Reverse engineer existing database into code

# Resources

Entity Framework Core on GitHub github.com/aspnet/entityframework

EF Core Documentation docs.microsoft.com/ef

EF Core Power Tools Extension (model visualizer and more):
github.com/ErikEJ/SqlCeToolbox/wiki/EF-Core-Power-Tools

EF Core migrations with existing database schema
cmatskas.com/ef-core-migrations-with-existing-database-schema-and-data

# Controlling Database Creation and Schema Changes with Migrations

**Julie Lerman**
MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK

@julielerman thedatafarm.com