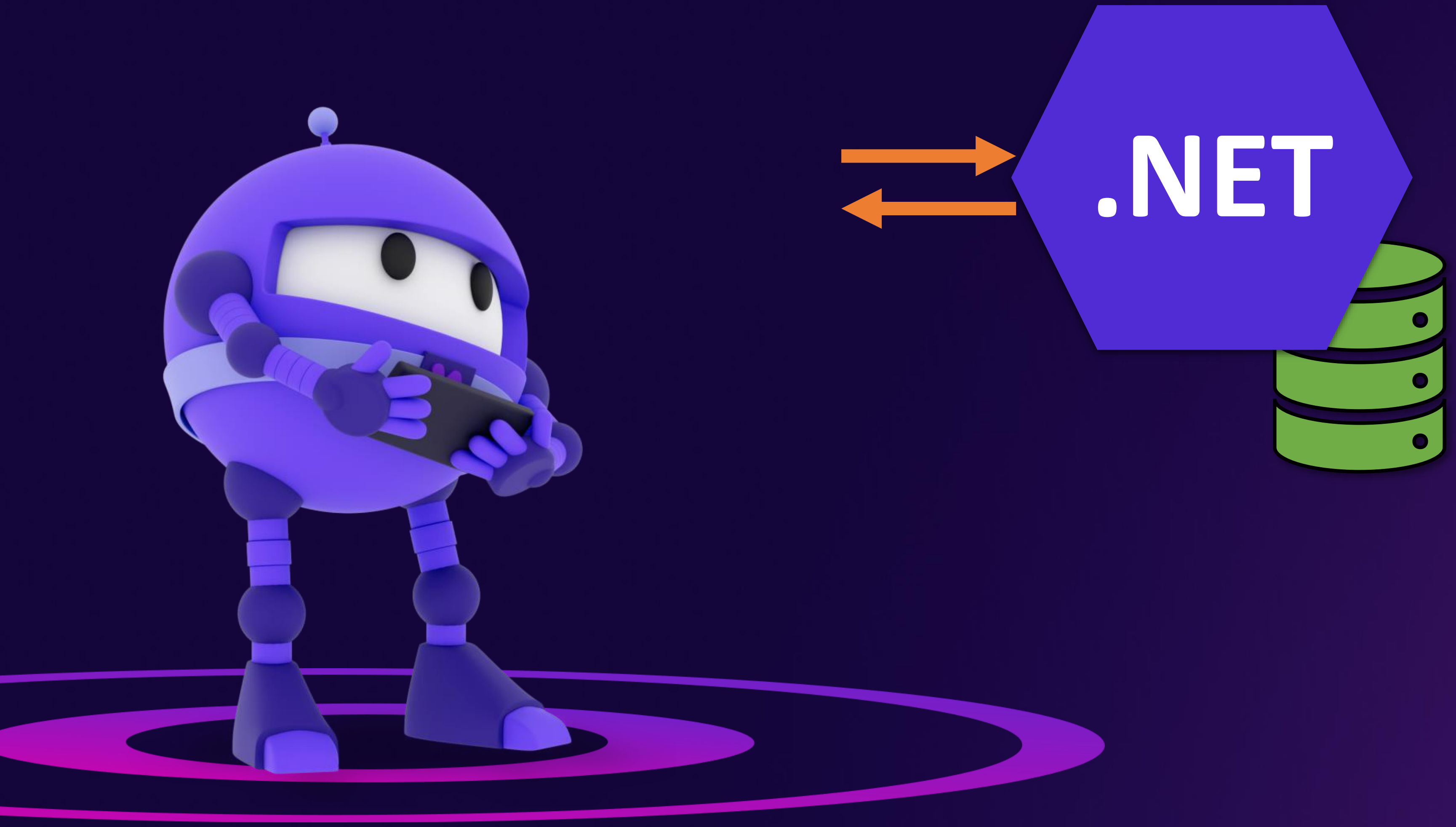
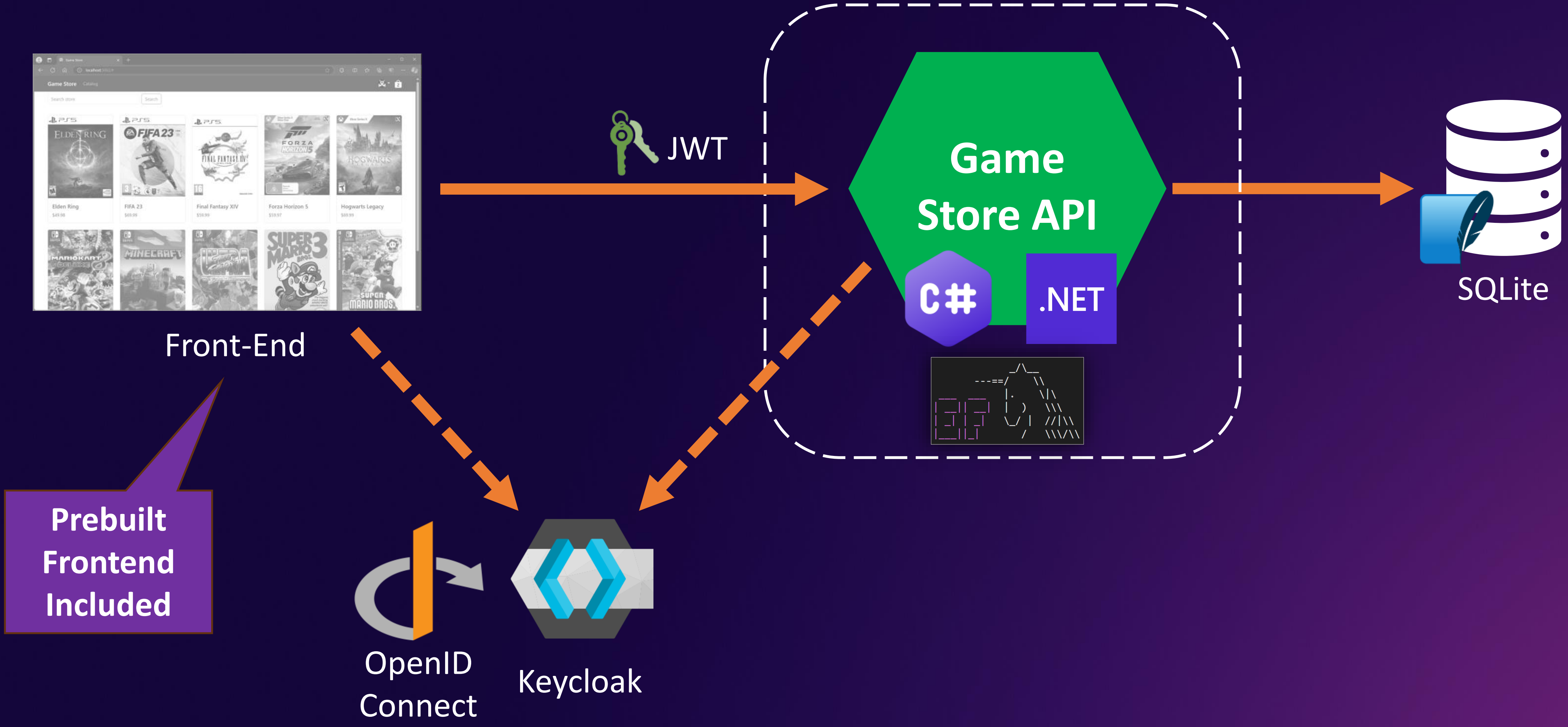
1. ASP.NET Core Essentials



Bootcamp – Stage 1



Front-End

JWT

Game Store API

C#

.NET

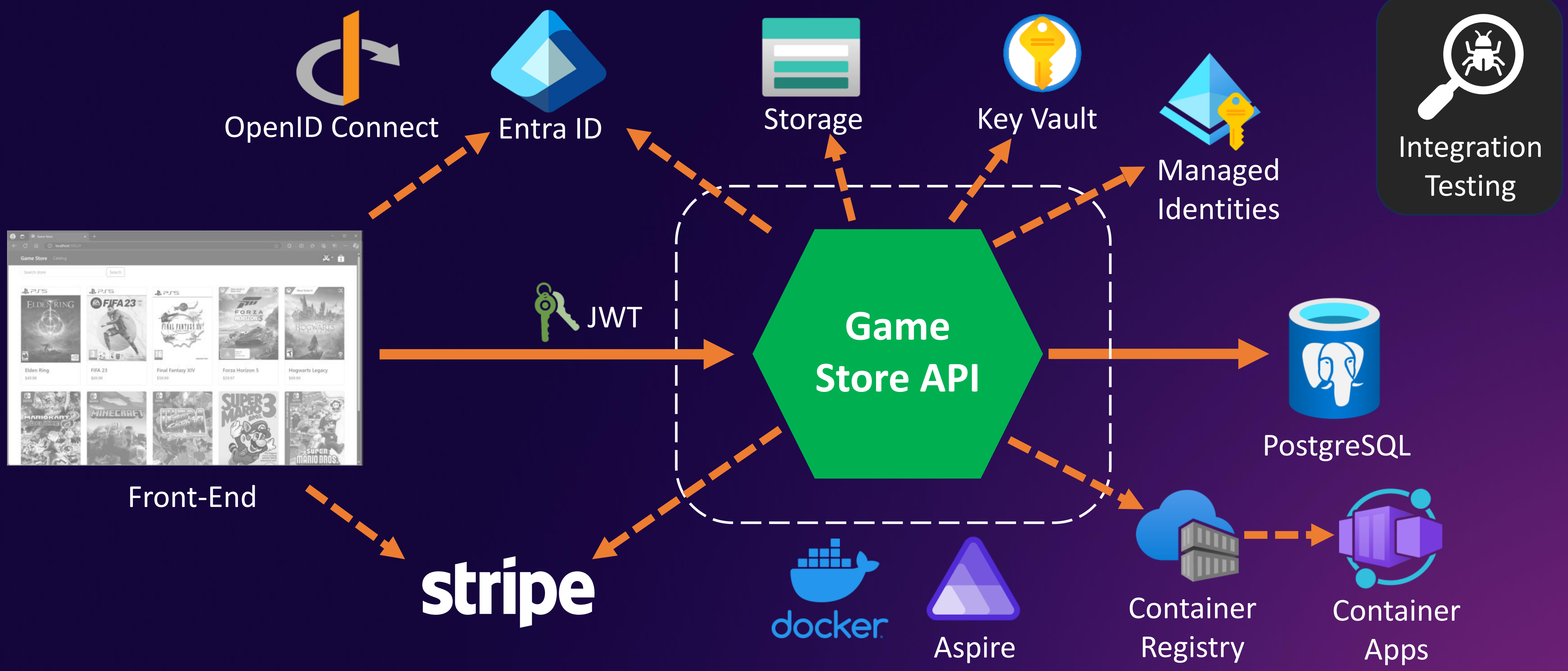
SQLite

Prebuilt Frontend Included

OpenID Connect

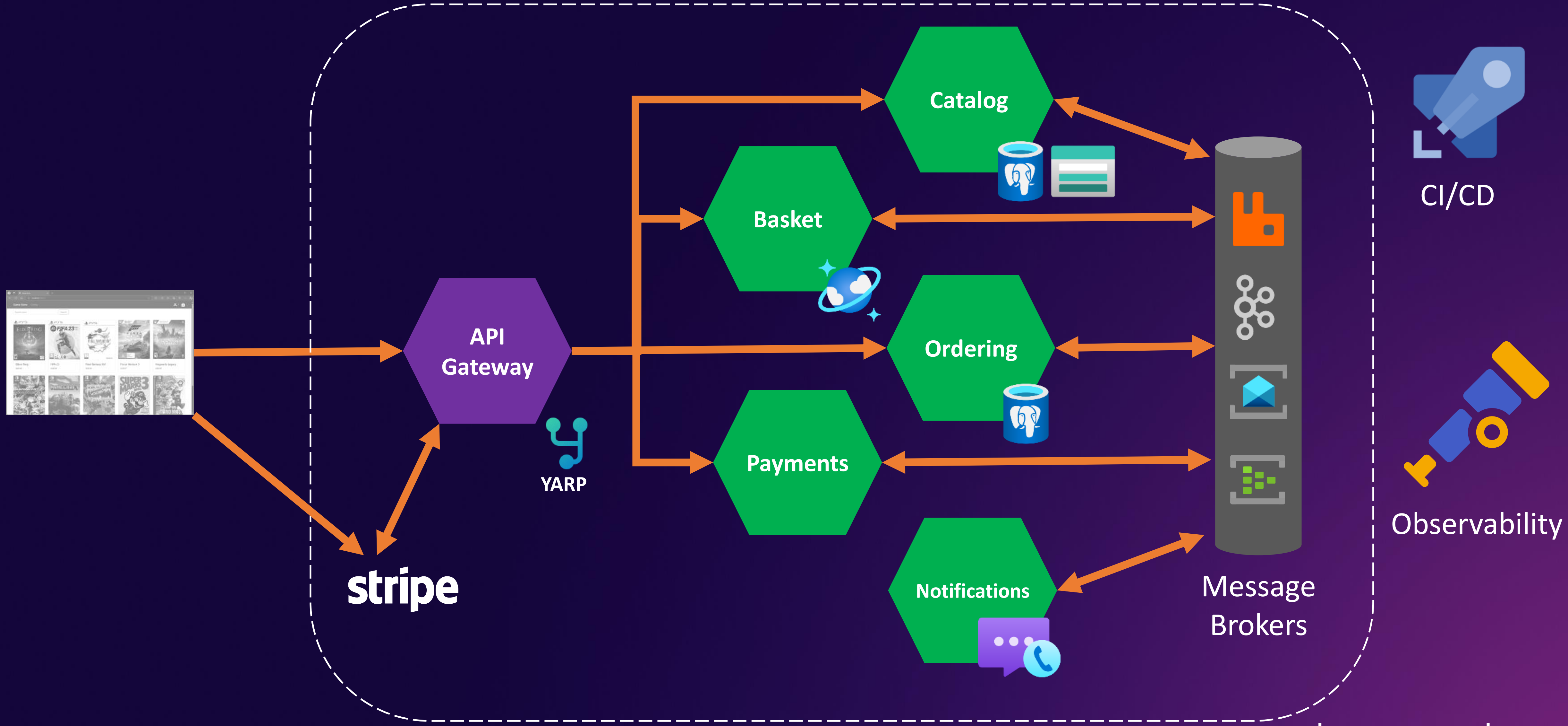
Keycloak

Bootcamp – Stage 2



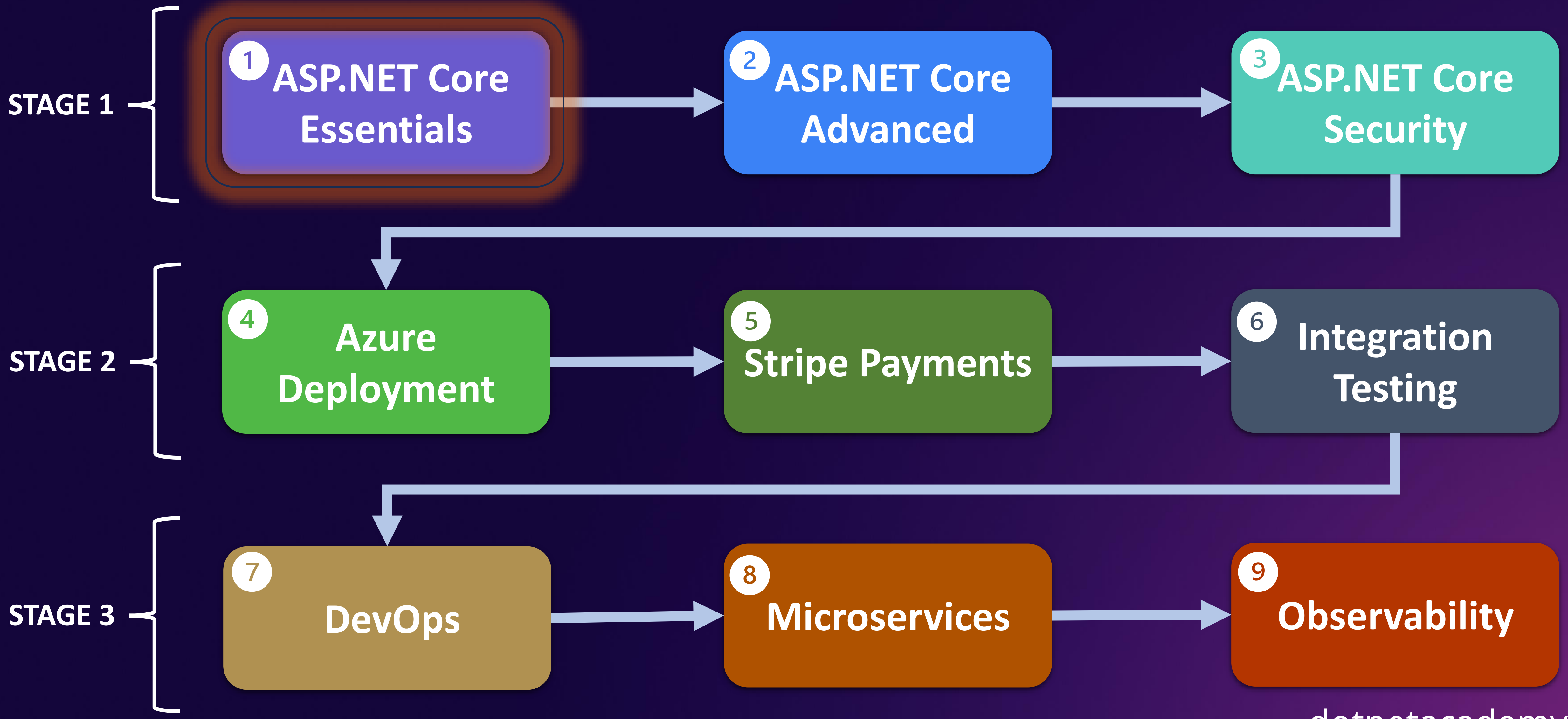
dotnetacademy.io

Bootcamp – Stage 3



dotnetacademy.io

Where are we?



dotnetacademy.io

ASP.NET Core Essentials Topics

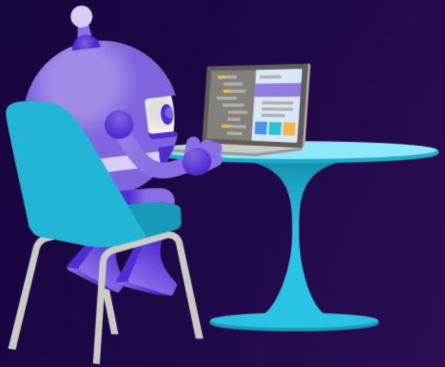
- Create ASP.NET Core Apps
- Understand REST APIs
- Implement CRUD Endpoints
- Input Data Validation
- Data Transfer Objects (DTOs)
- Vertical Slice Architecture
- Dependency Injection
- Working With Data (EF Core)
- Configuration System
- UI Integration

Is this bootcamp for you?

Basic C# or Java Knowledge

Web Development Essentials

Some Database Experience



Beginner Level Course

Software prerequisites

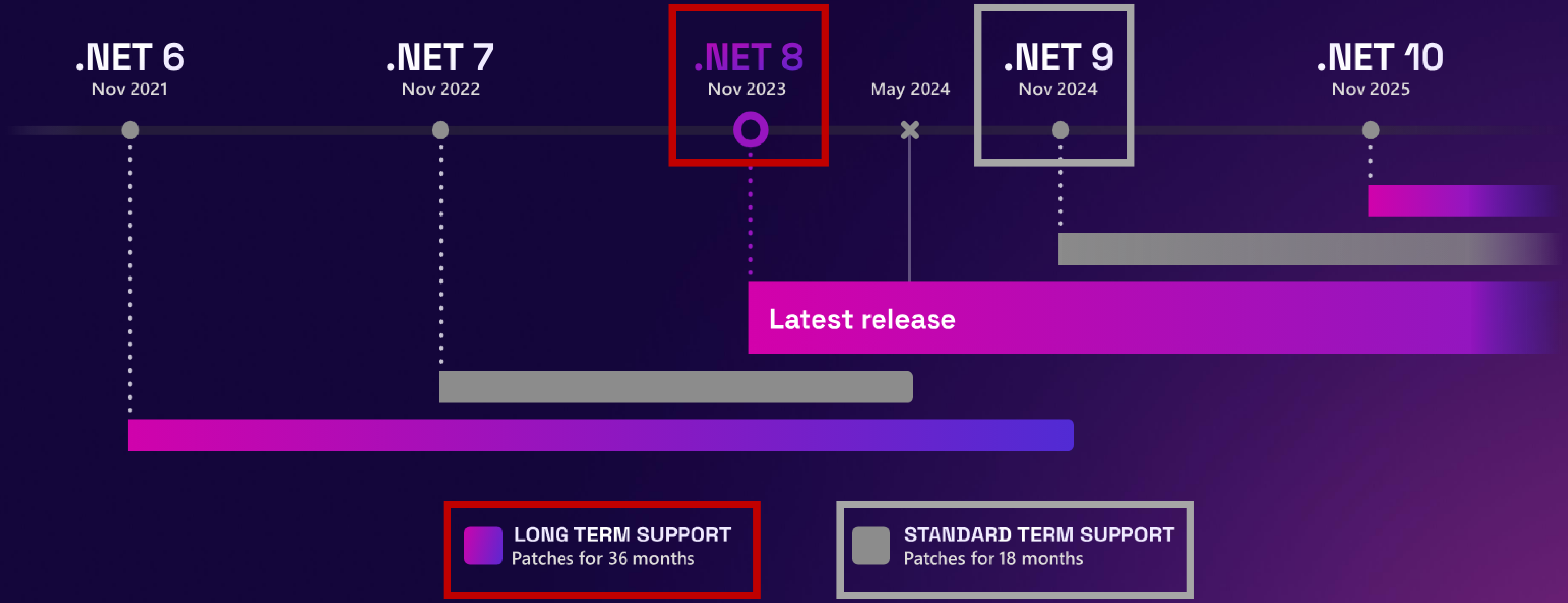
.NET SDK

https://dot.net/download

Visual Studio Code

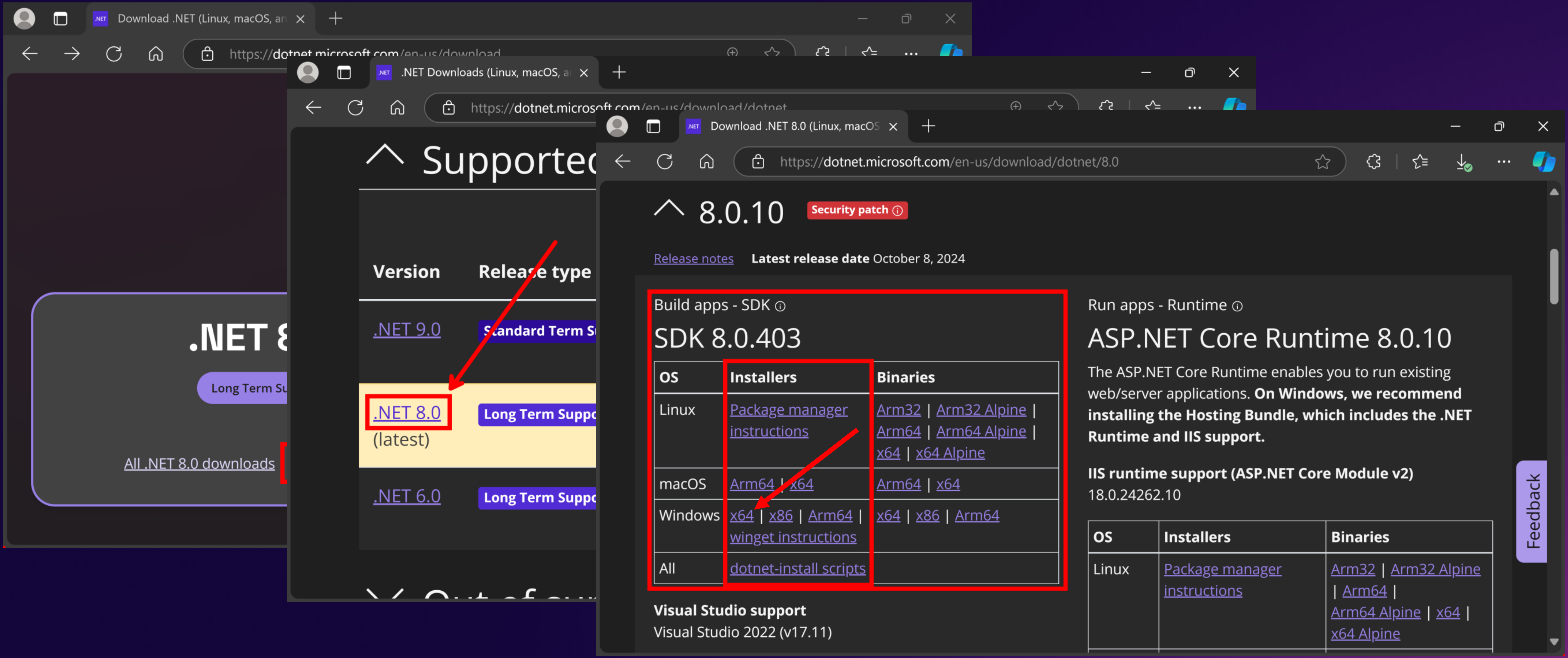
https://code.visualstudio.com

Choosing a .NET version



dotnetacademy.io

Installing the .NET 8 SDK



Download .NET (Linux, macOS, an

https://dotnet.microsoft.com/en-us/download

.NET 8

Long Term Su

All .NET 8.0 downloads

.NET Downloads (Linux, macOS, a

https://dotnet.microsoft.com/en-us/download/dotnet

Supportec

Version	Release type
.NET 9.0	Standard Term S
.NET 8.0 (latest)	Long Term Suppo
.NET 6.0	Long Term Suppo

Download .NET 8.0 (Linux, macOS

https://dotnet.microsoft.com/en-us/download/dotnet/8.0

8.0.10 Security patch

Release notes **Latest release date** October 8, 2024

Build apps - SDK

SDK 8.0.403

OS	Installers	Binaries
Linux	Package manager instructions	Arm32 | Arm32 Alpine | Arm64 | Arm64 Alpine | x64 | x64 Alpine
macOS	Arm64 | x64	Arm64 | x64
Windows	x64 | x86 | Arm64 | winget instructions	x64 | x86 | Arm64
All	dotnet-install scripts	

Visual Studio support
Visual Studio 2022 (v17.11)

Run apps - Runtime

ASP.NET Core Runtime 8.0.10

The ASP.NET Core Runtime enables you to run existing web/server applications. **On Windows, we recommend installing the Hosting Bundle, which includes the .NET Runtime and IIS support.**

IIS runtime support (ASP.NET Core Module v2)
18.0.24262.10

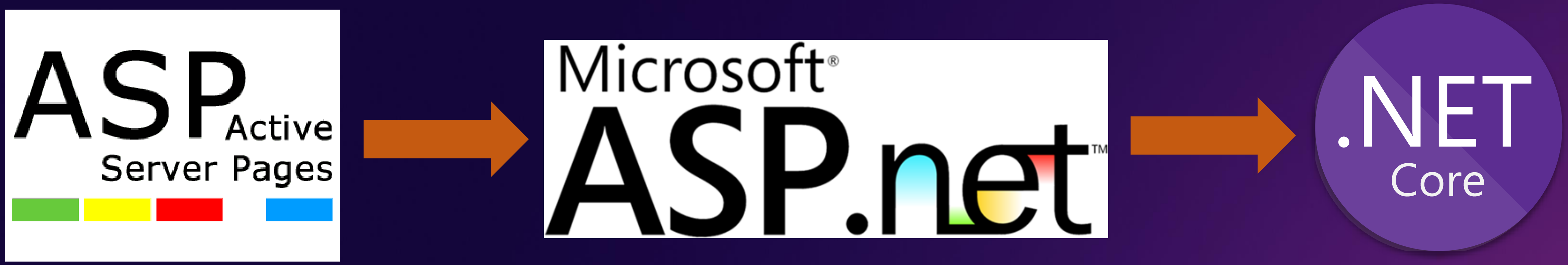
OS	Installers	Binaries
Linux	Package manager instructions	Arm32 | Arm32 Alpine | Arm64 | Arm64 Alpine | x64 | x64 Alpine

dotnetacademy.io

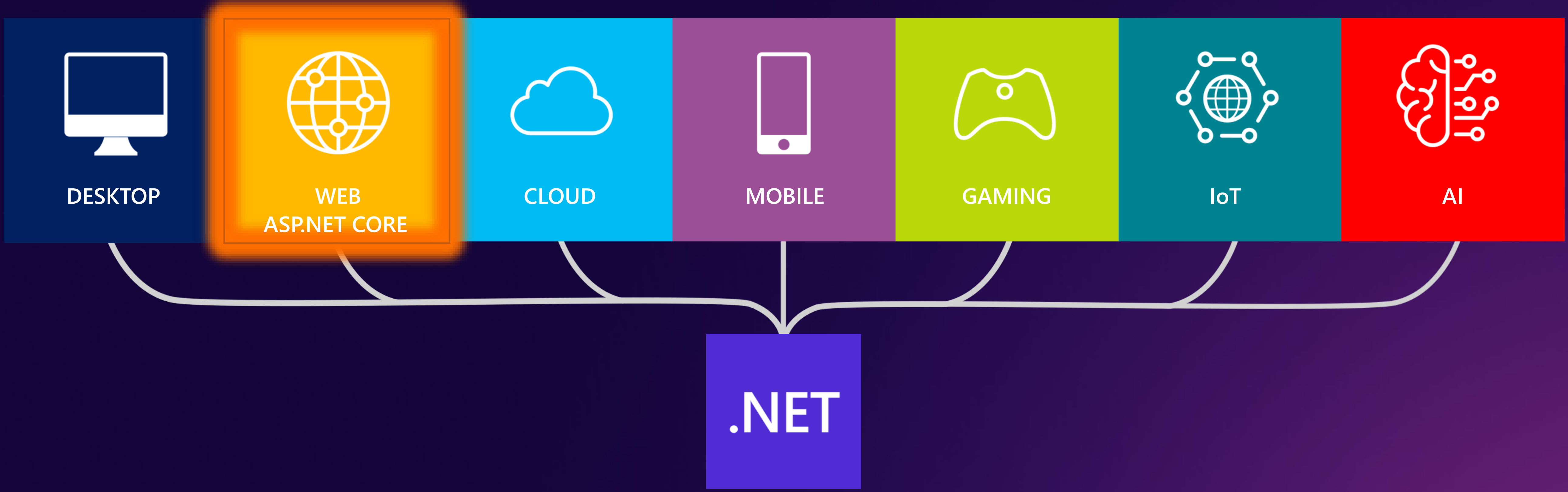
Introduction to ASP.NET Core

What is ASP.NET Core?

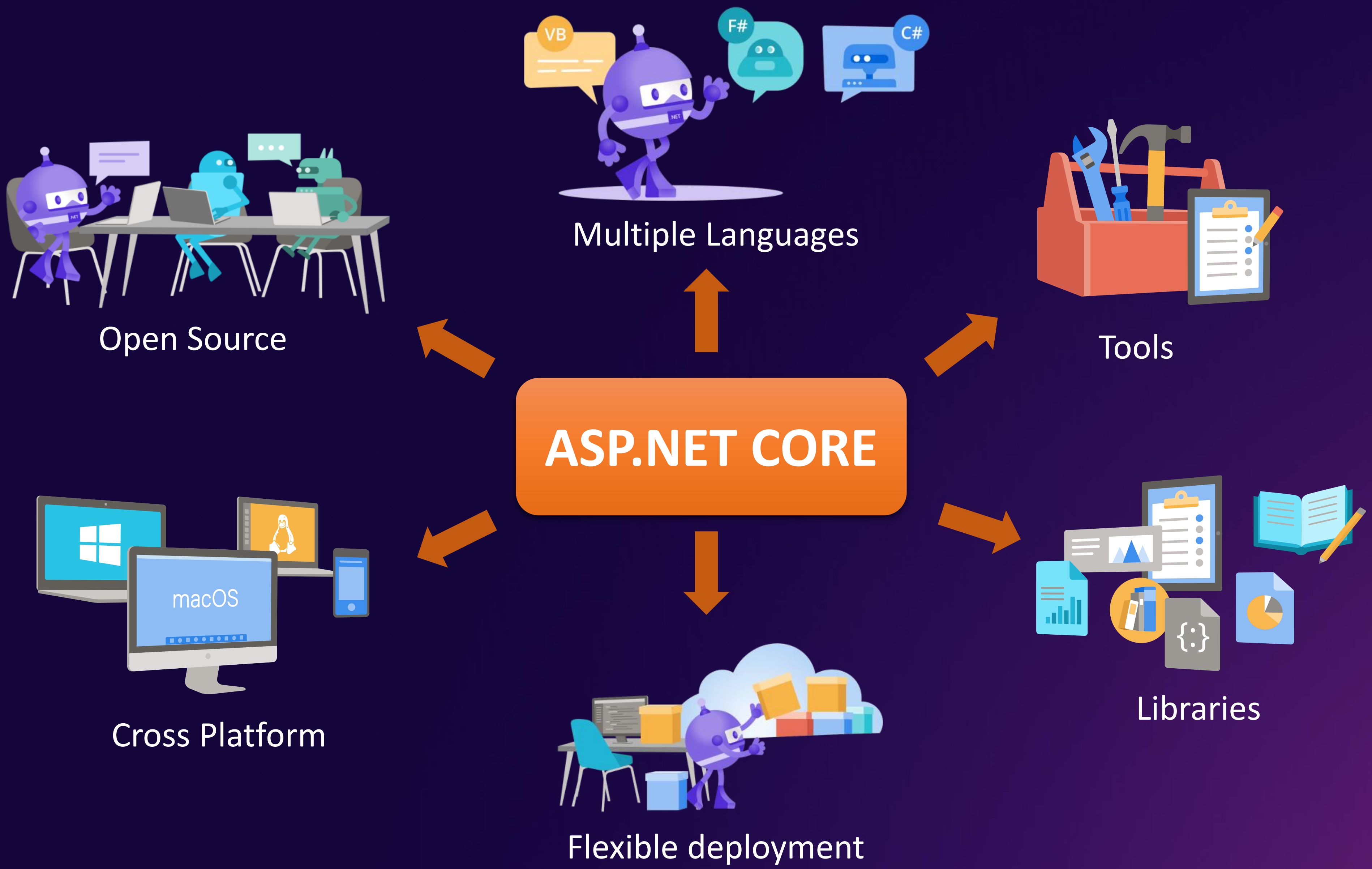
ASP.NET Core is a popular web-development framework for building web apps on the .NET platform



What is.NET?



A free, cross-platform, open source developer platform
for building many different types of applications.



ASP.NET CORE

Multiple Languages

Open Source

Tools

Cross Platform

Libraries

Flexible deployment

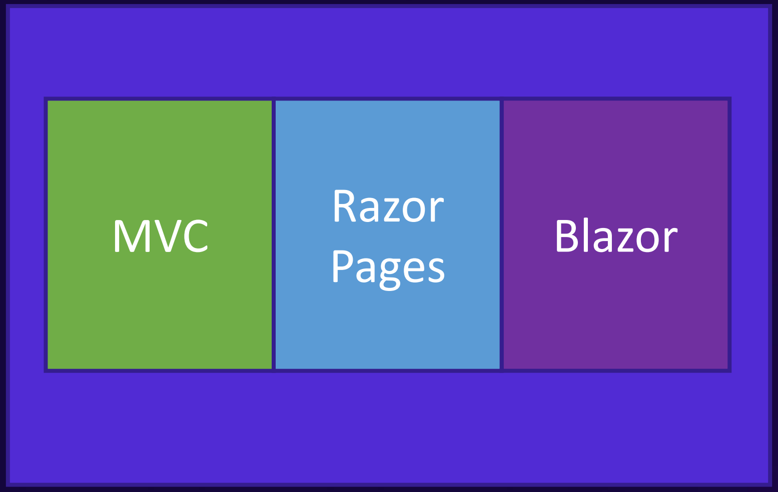
dotnetacademy.io

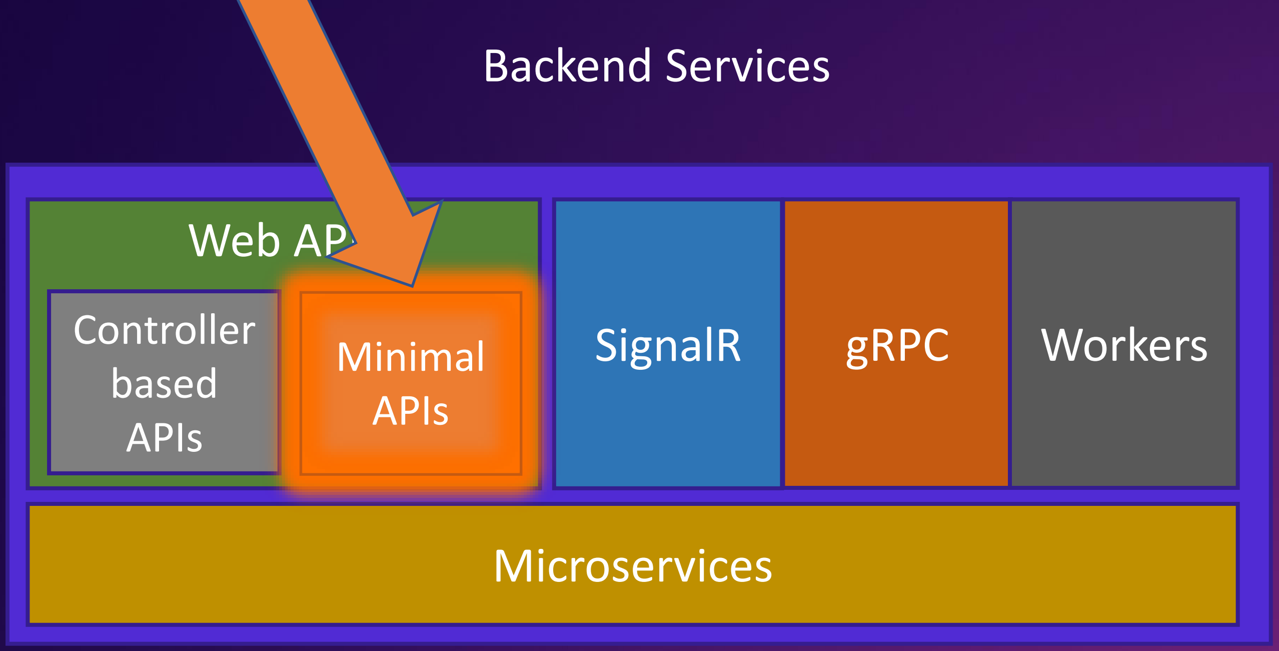
What can you create with ASP.NET Core?

Web UI

THIS COURSE COVERS

Backend Services

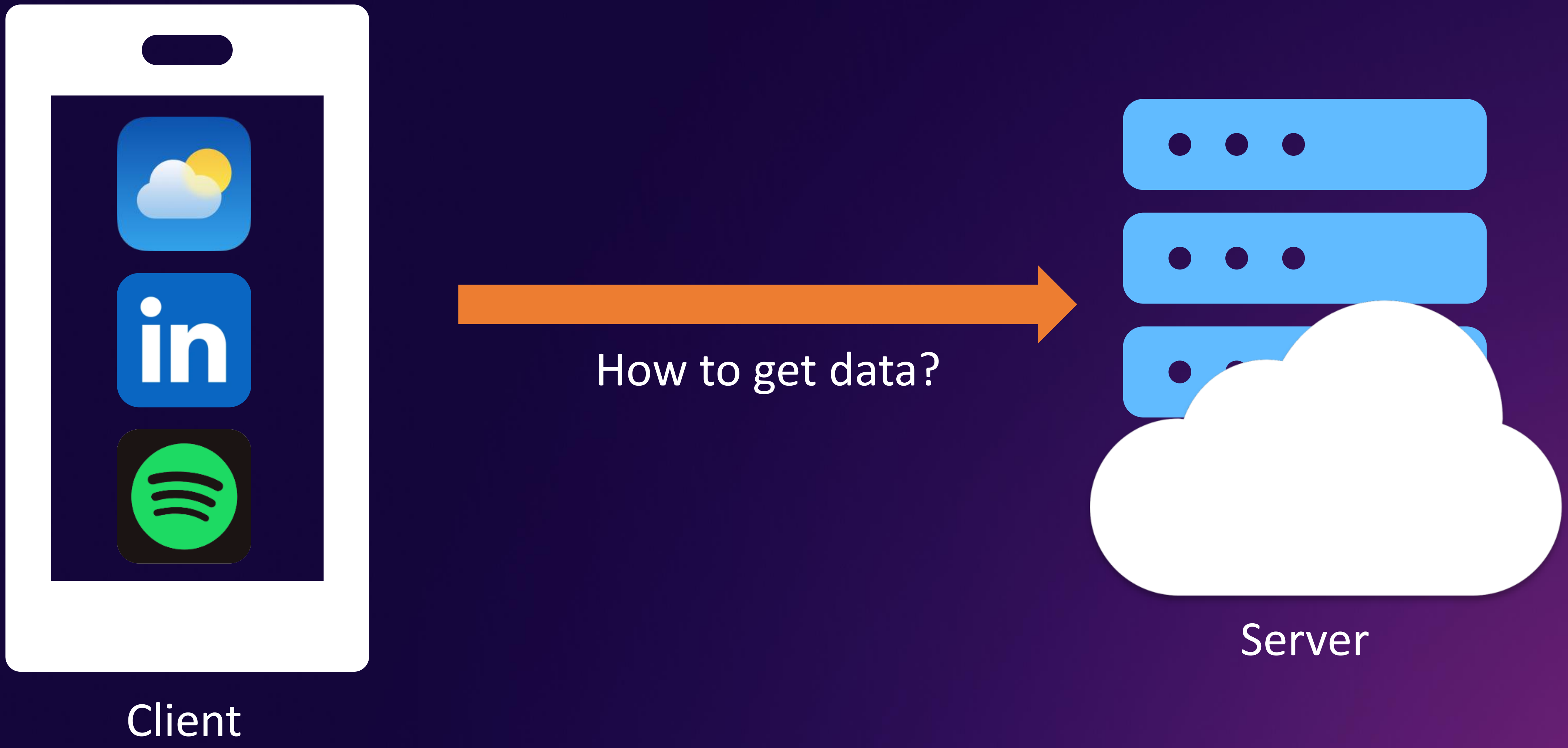




dotnetacademy.io

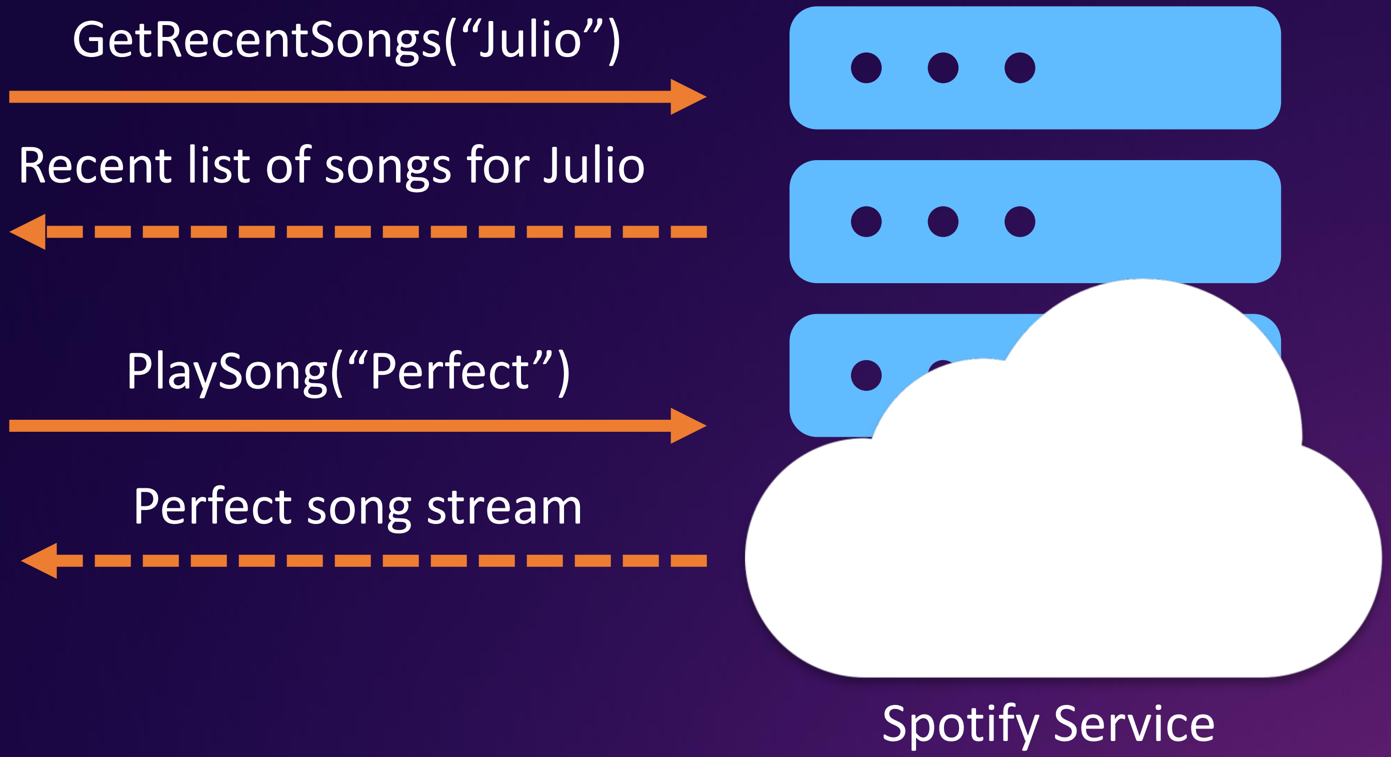
What is a REST API?

Clients and Servers



What is an API?

Application
Programming
Interface



An API helps clients communicate what they want to the service so it can understand and fulfill the request.

What is REST?

REpresentational

State

Transfer

Stateless	Client-Server
Uniform interface	Layered system
Cacheable	Code on demand

A set of guiding principles that impose conditions on how an API should work

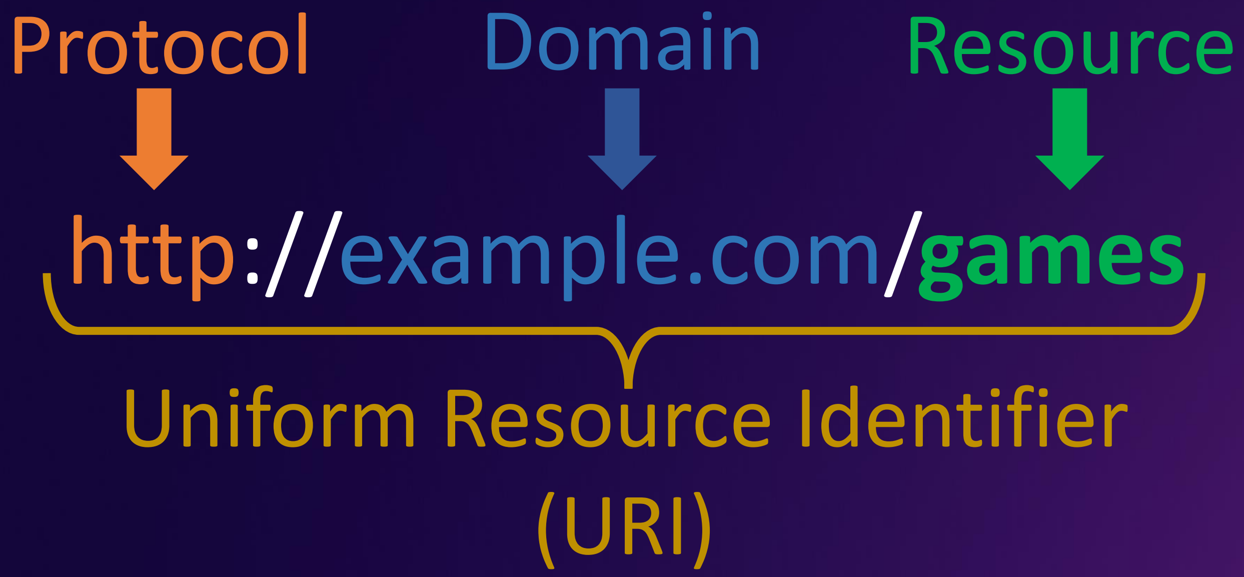
What is a REST API?

A REST or RESTFUL API is one that conforms to the REST architectural style

Interacting with REST APIs

How to identify resources in a REST API?

A resource is any object, document or thing that the API can receive from or send to clients



Protocol

Domain

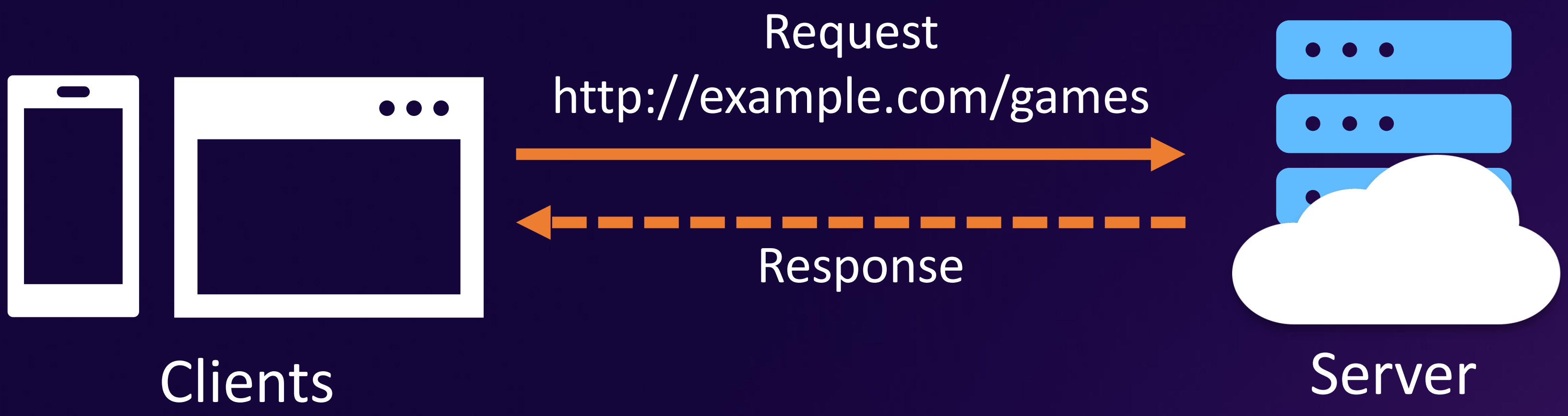
Resource

http://example.com/**games**

Uniform Resource Identifier

(URI)

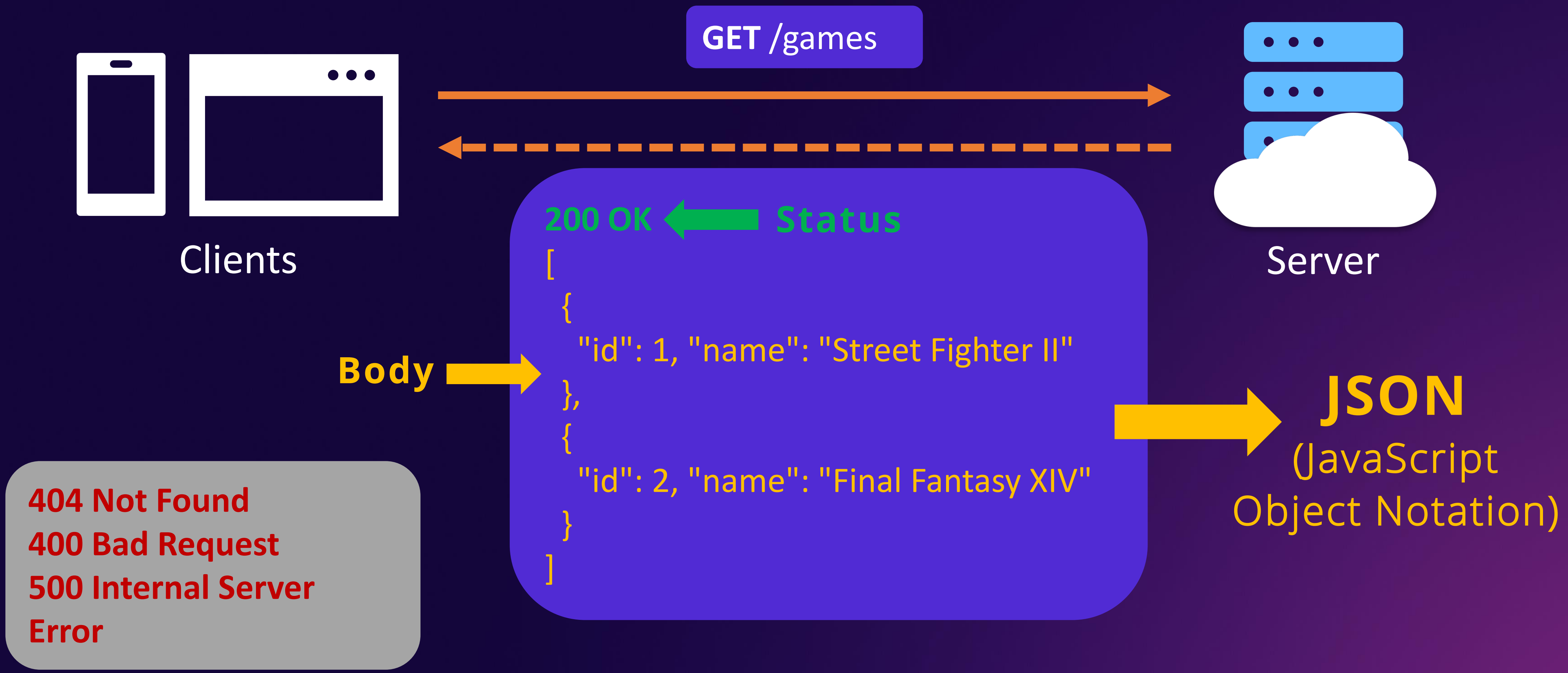
How to interact with a REST API?



HTTP Methods

Create	POST	Creates a new resource
Read	GET	Retrieves the resource representation/state
Update	PUT	Updates an existing resource
Delete	DELETE	Deletes a resource

Get All Games - HTTP GET

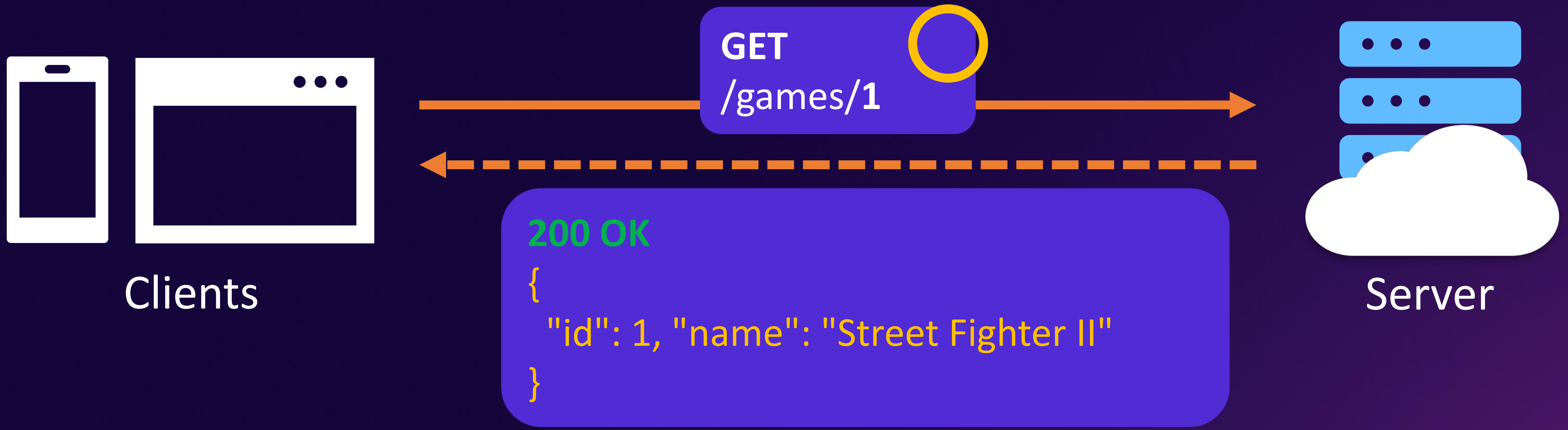


200 OK

```
[
 {
  "id": 1, "name": "Street Fighter II"
 },
 {
  "id": 2, "name": "Final Fantasy XIV"
 }
]
```

404 Not Found
400 Bad Request
500 Internal Server Error

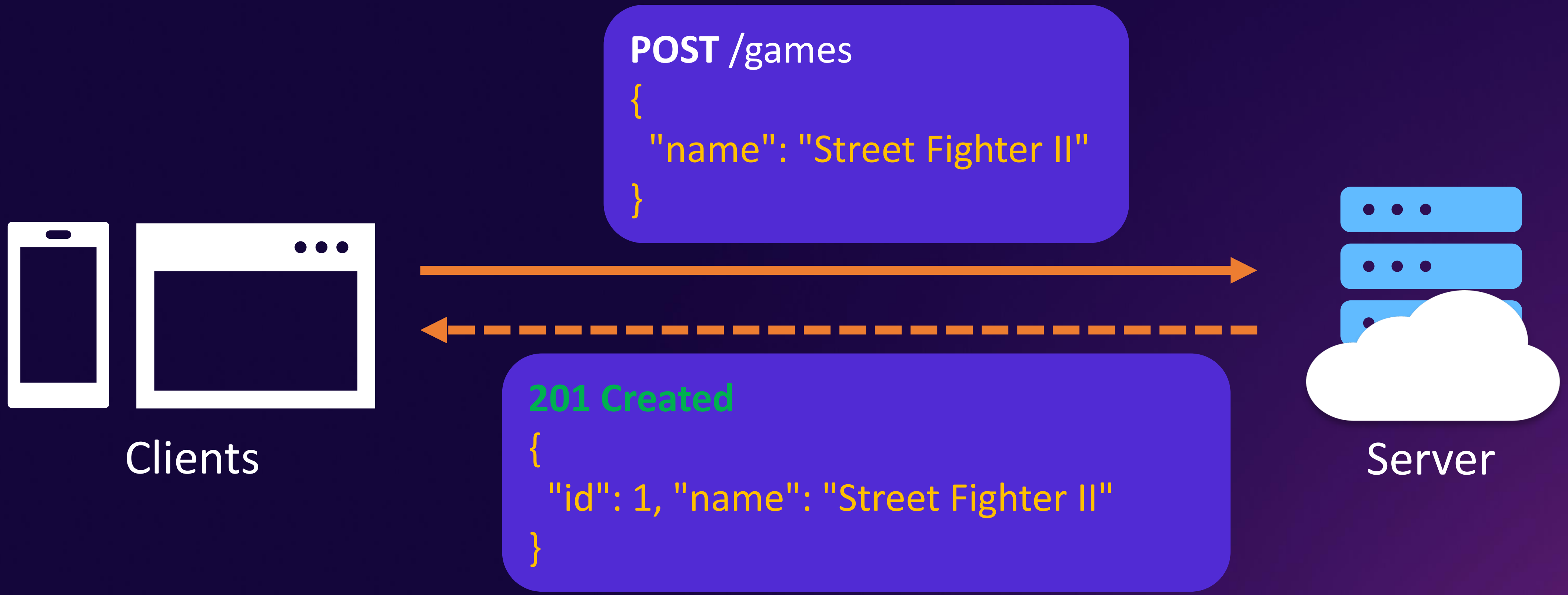
Get A Specific Game - HTTP GET



200 OK

```
{
  "id": 1, "name": "Street Fighter II"
}
```

Create A Game - HTTP POST



POST /games

```
{
  "name": "Street Fighter II"
}
```

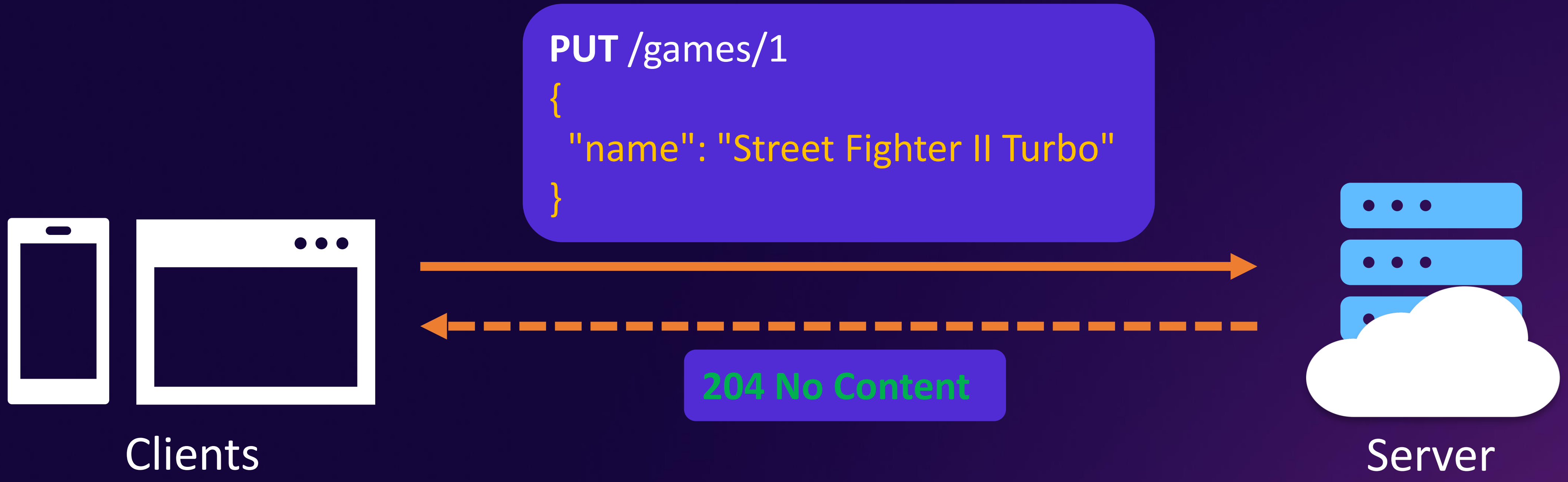
201 Created

```
{
  "id": 1, "name": "Street Fighter II"
}
```

Clients

Server

Update A Game - HTTP PUT

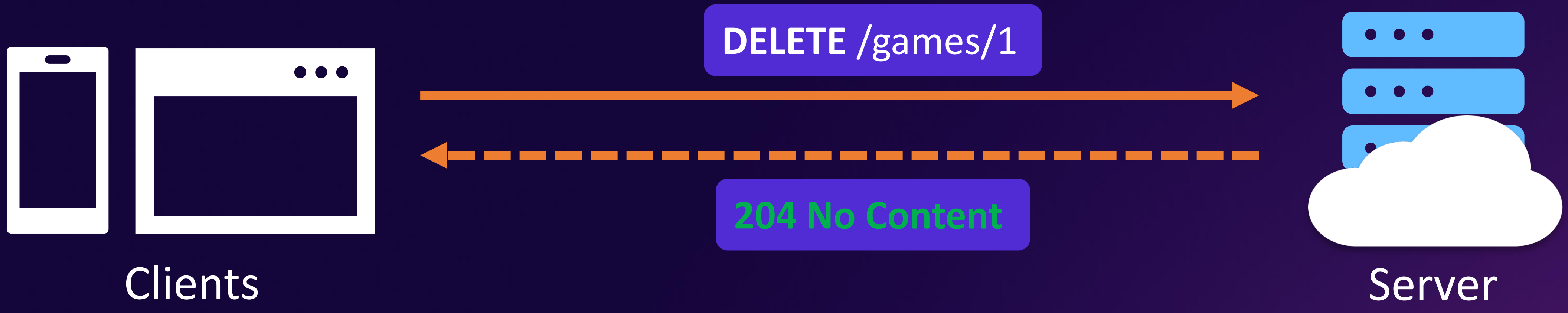


PUT /games/1

```
{
  "name": "Street Fighter II Turbo"
}
```

204 No Content

Delete A Game - HTTP DELETE



Games REST API

GET	/games
GET	/games/1
POST	/games
PUT	/games/1
DELETE	/games/1

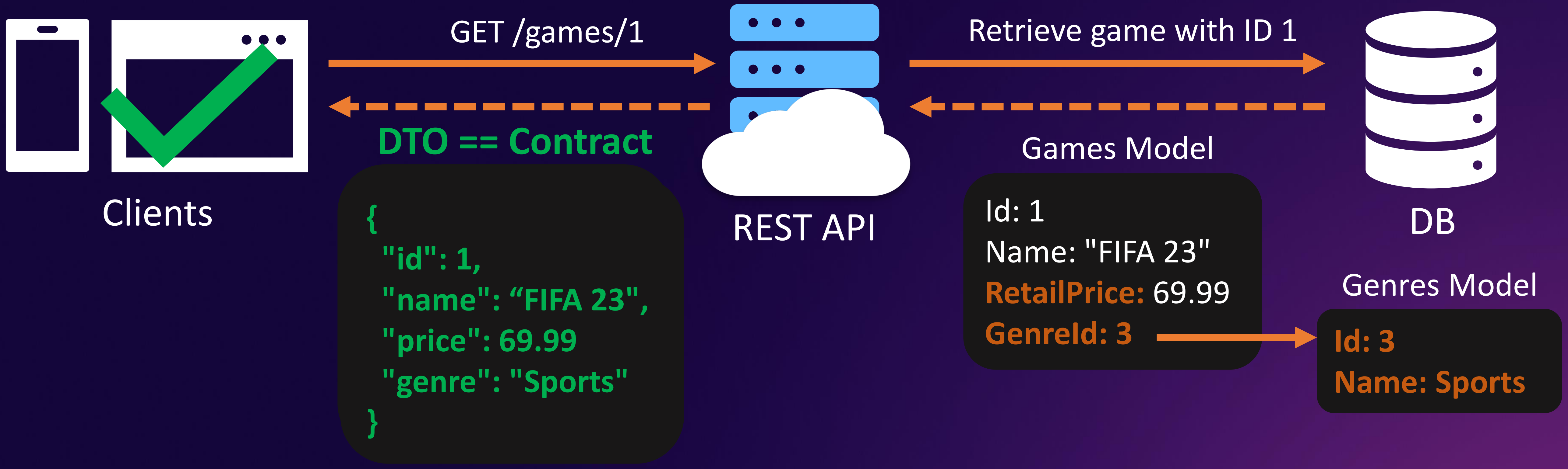
Understanding Data Transfer Objects

What is a Data Transfer Object?

A Data Transfer Object (DTO) is an object that carries data between processes or applications.

In the context of a REST API, a DTO can be considered a contract between the client and server.

Why use Data Transfer Objects?



```
{
  "id": 1,
  "name": “FIFA 23",
  "price": 69.99
  "genre": "Sports"
}
```

Id: 1
Name: "FIFA 23"
RetailPrice: 69.99
GenreId: 3

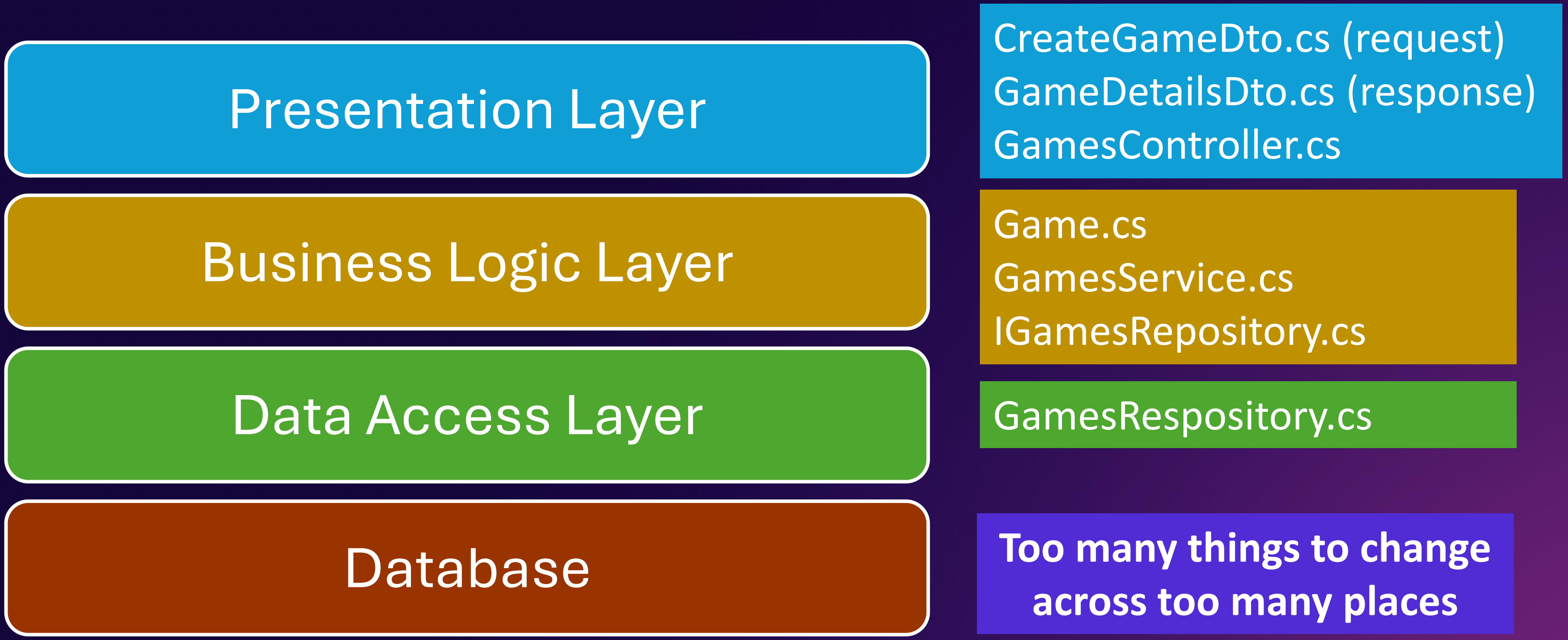
Id: 3
Name: Sports

The DTO acts as a contract that defines the expectations and requirements for how data will be exchanged between client and server

dotnetacademy.io

Vertical Slice Architecture

Structuring code the old way



Presentation Layer

Business Logic Layer

Data Access Layer

Database

Create Game

CreateGameDto.cs (request)
GameDetailsDto.cs (response)
GamesController.cs

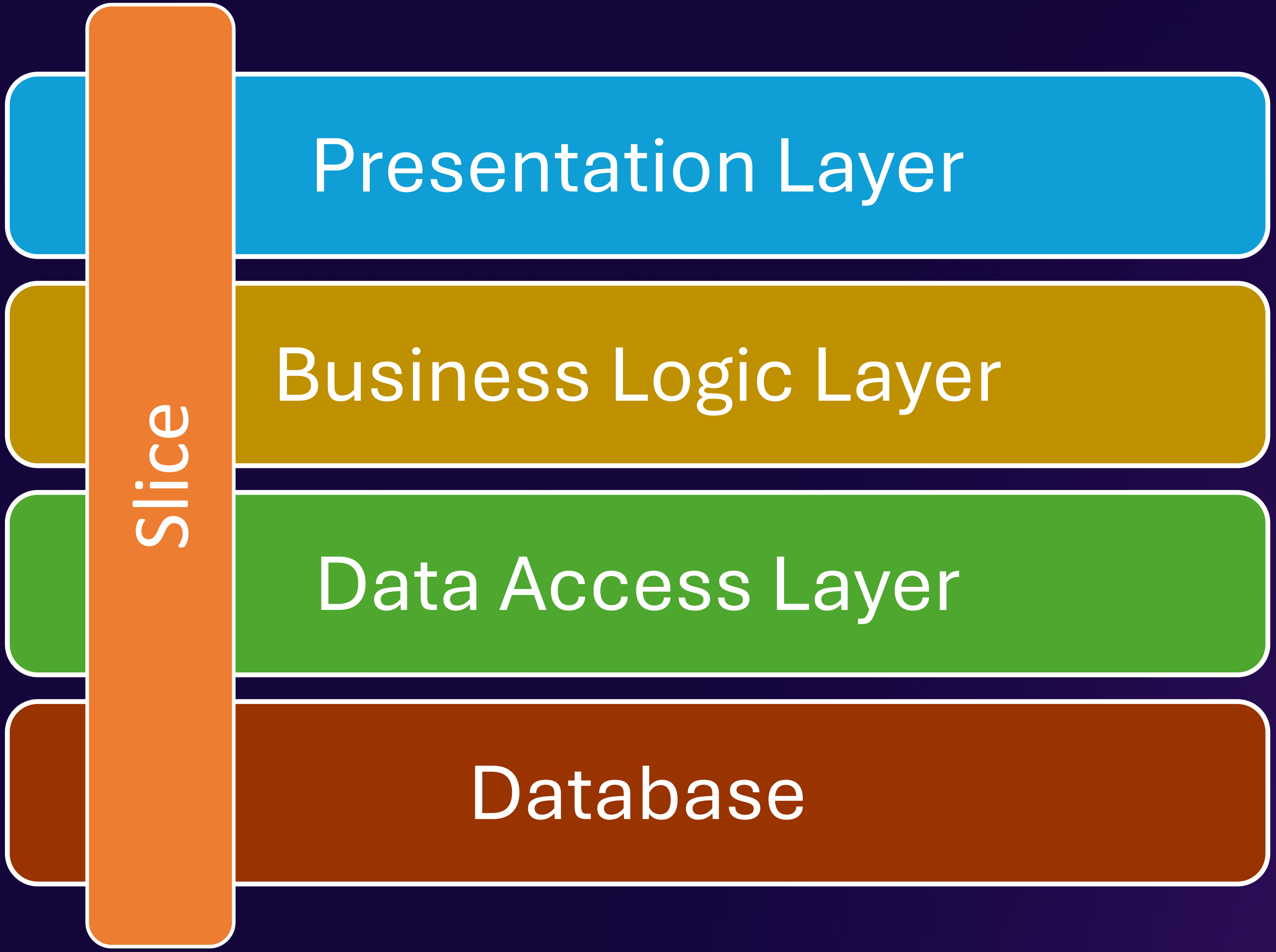
Game.cs
GamesService.cs
IGamesRepository.cs

GamesRespository.cs

Too many things to change across too many places

dotnetacademy.io

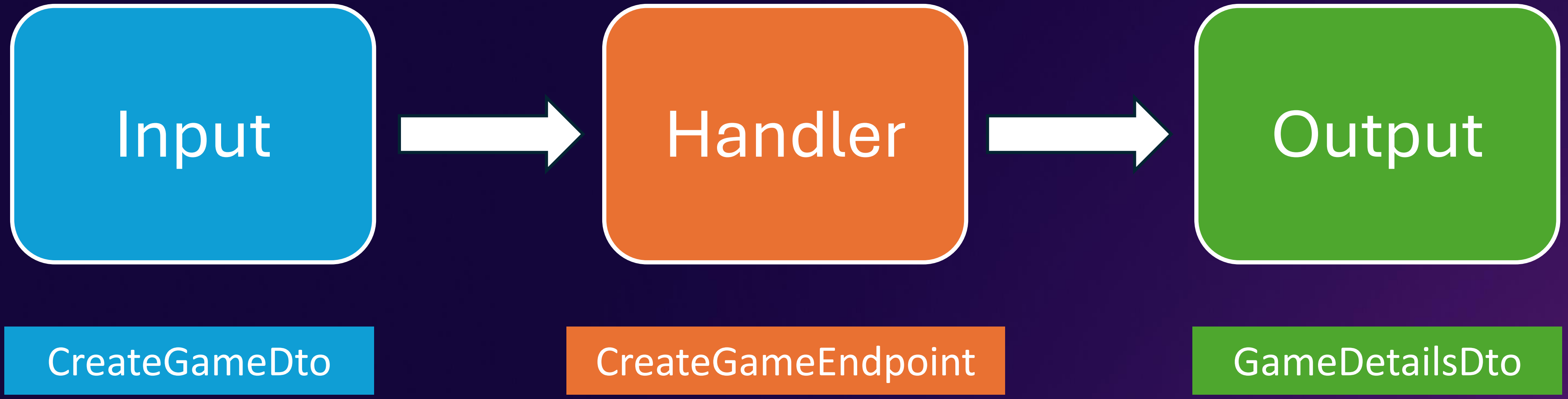
Structuring code around slices



- **Codebase is divided into independent features (slices)**
- **Each slice contains everything needed for a specific feature**

dotnetacademy.io

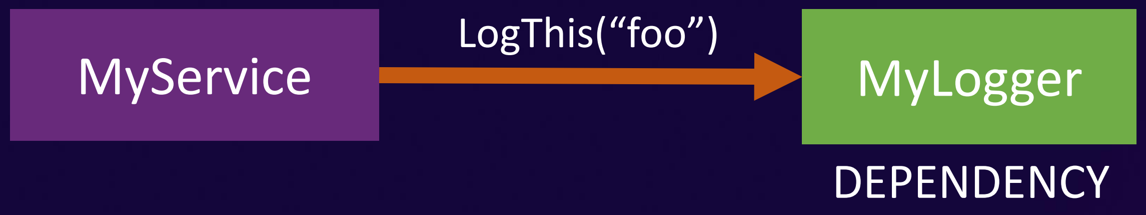
Structuring a slice



- Single cohesive unit where the flow is clear
- The code is simpler to write and maintain
- There's less unnecessary abstraction

Understanding Dependency Injection

What is a Dependency?



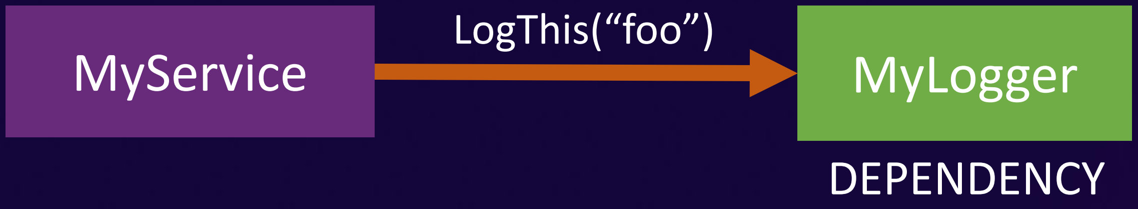
```
public MyService()
{
   var logger = new MyLogger();
   logger.LogThis("I'm Ready!");
}
```

```
public MyService()
{
   var writter = new MyFileWritter("output.log");
   var logger = new MyLogger(writter);
   logger.LogThis("I'm Ready!");
}
```

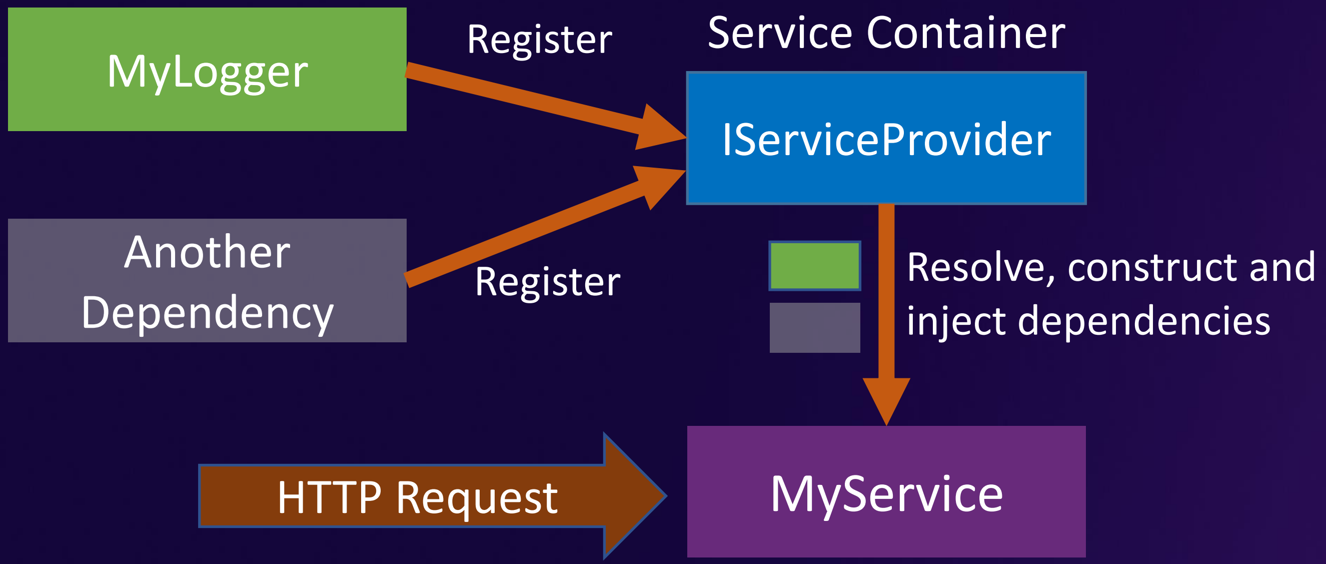
Problems

- MyService is tightly coupled to the Logger dependency. Any changes to MyLogger require changes to MyService.
- MyService needs to know how to construct and configure the MyLogger dependency.
- It's hard to test MyService since the MyLogger dependency cannot be mocked or stubbed.

What is Dependency Injection?



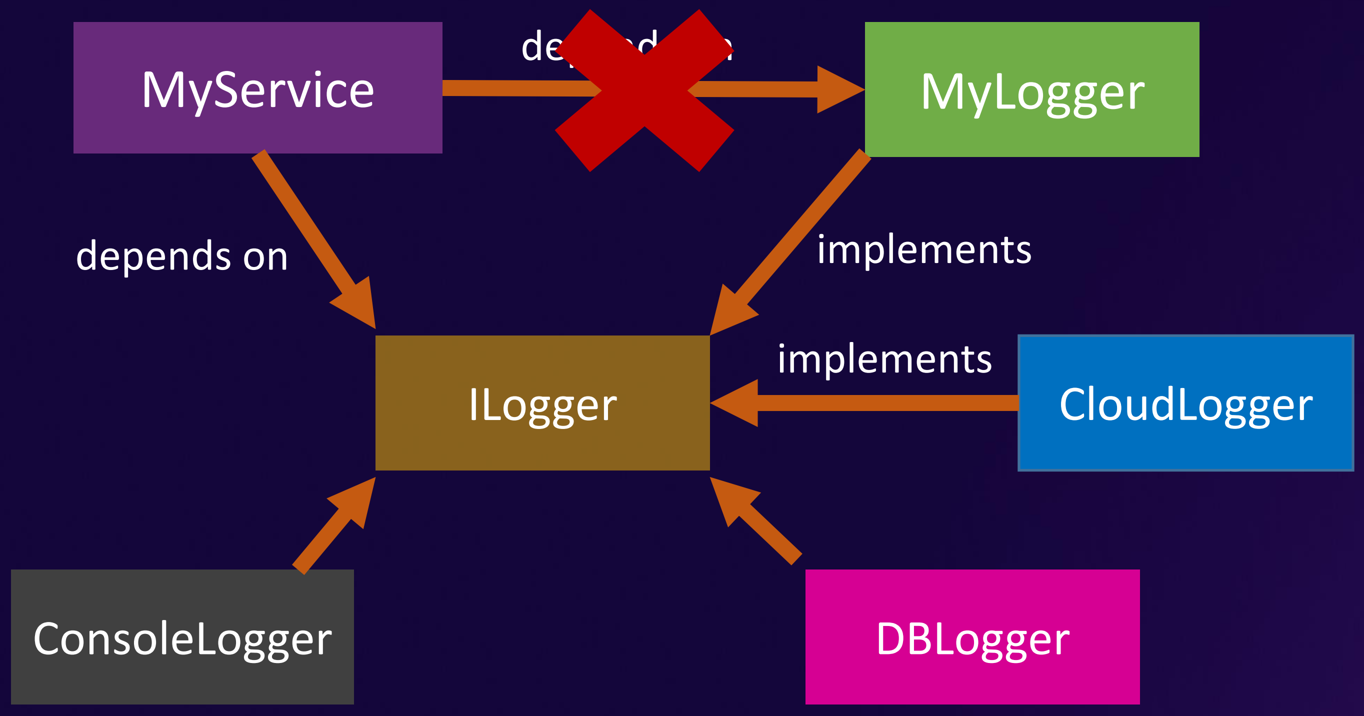
```
public MyService(MyLogger logger)
{
   logger.LogThis(“I’m Ready!”);
}
```



Benefits

- MyService won’t be affected by changes to its dependencies.
- MyService doesn’t need to know how to construct or configure its dependencies.
- Dependencies can also be injected as parameters to minimal API endpoints
- Opens the door to using Dependency Inversion

Using Dependency Inversion



```
public MyService(ILogger logger)
{
   logger.LogThis("I'm Ready!");
}
```

The Dependency Inversion Principle

"Code should depend on abstractions as opposed to concrete implementations."

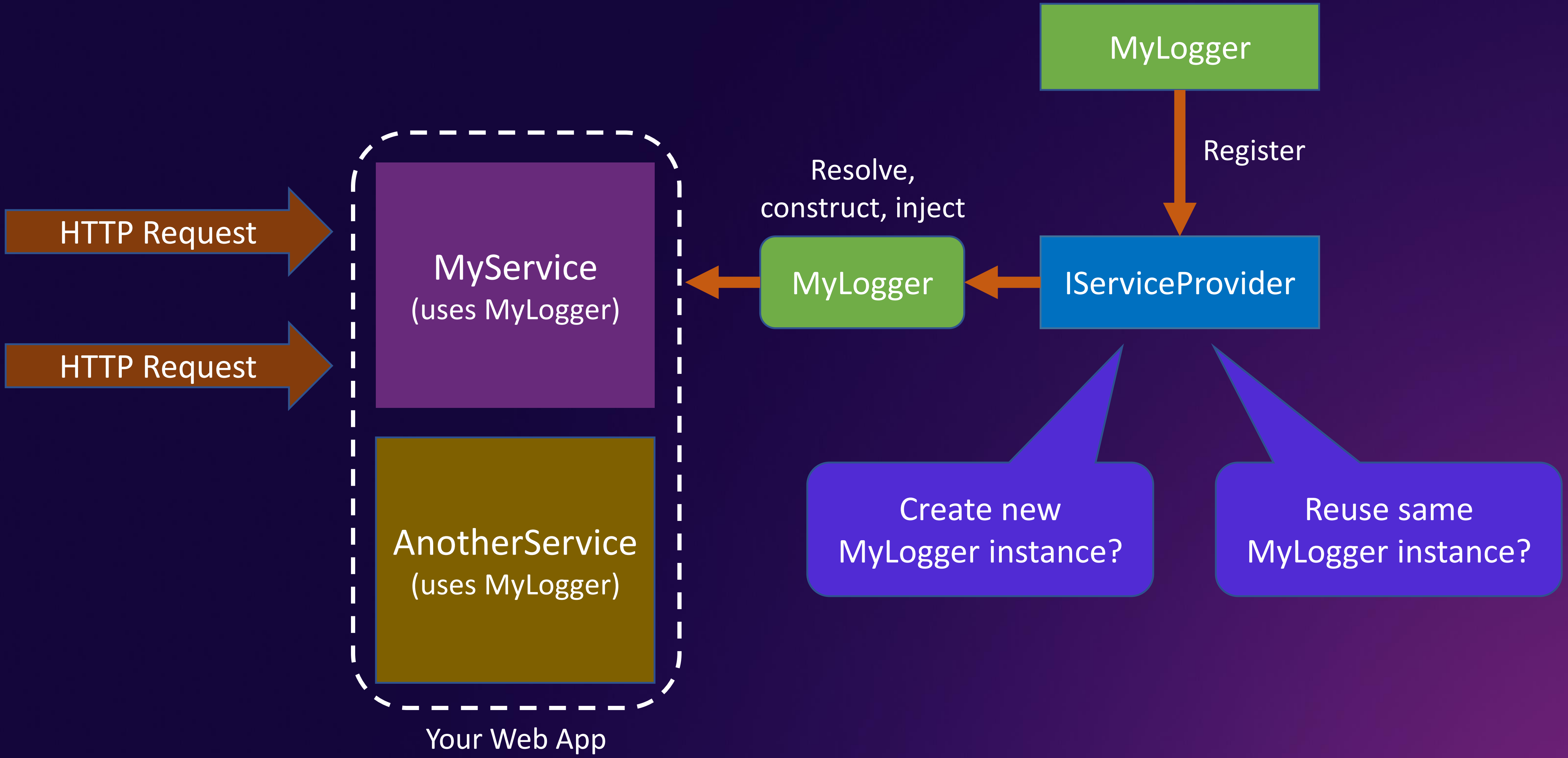
Benefits

- The logger dependency can be swapped out for a different implementation without modifying MyService
- It's easier to test MyService since the logger dependency can be mocked or stubbed
- Code is cleaner, easier to modify and easier to reuse

dotnetacademy.io

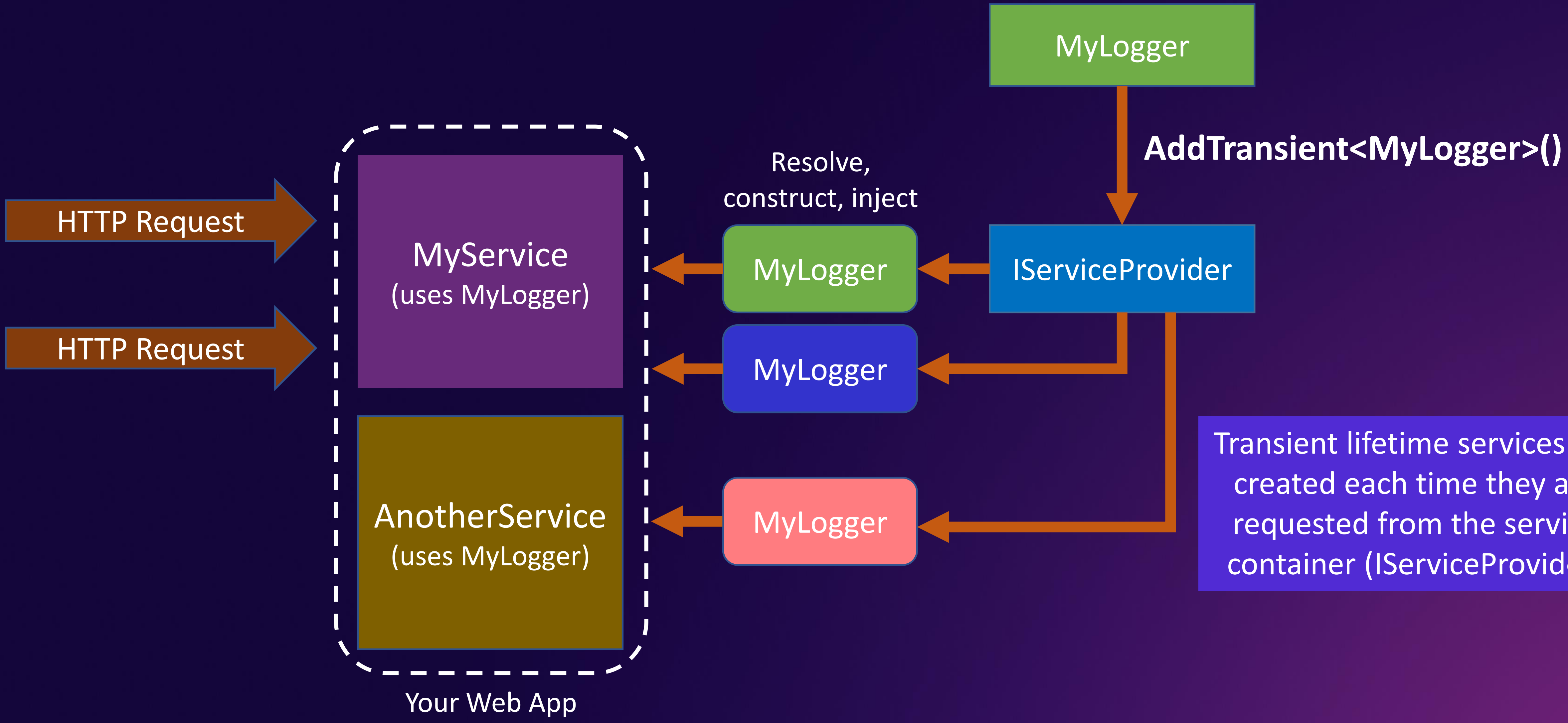
Understanding Service Lifetimes

When should instances be created?



dotnetacademy.io

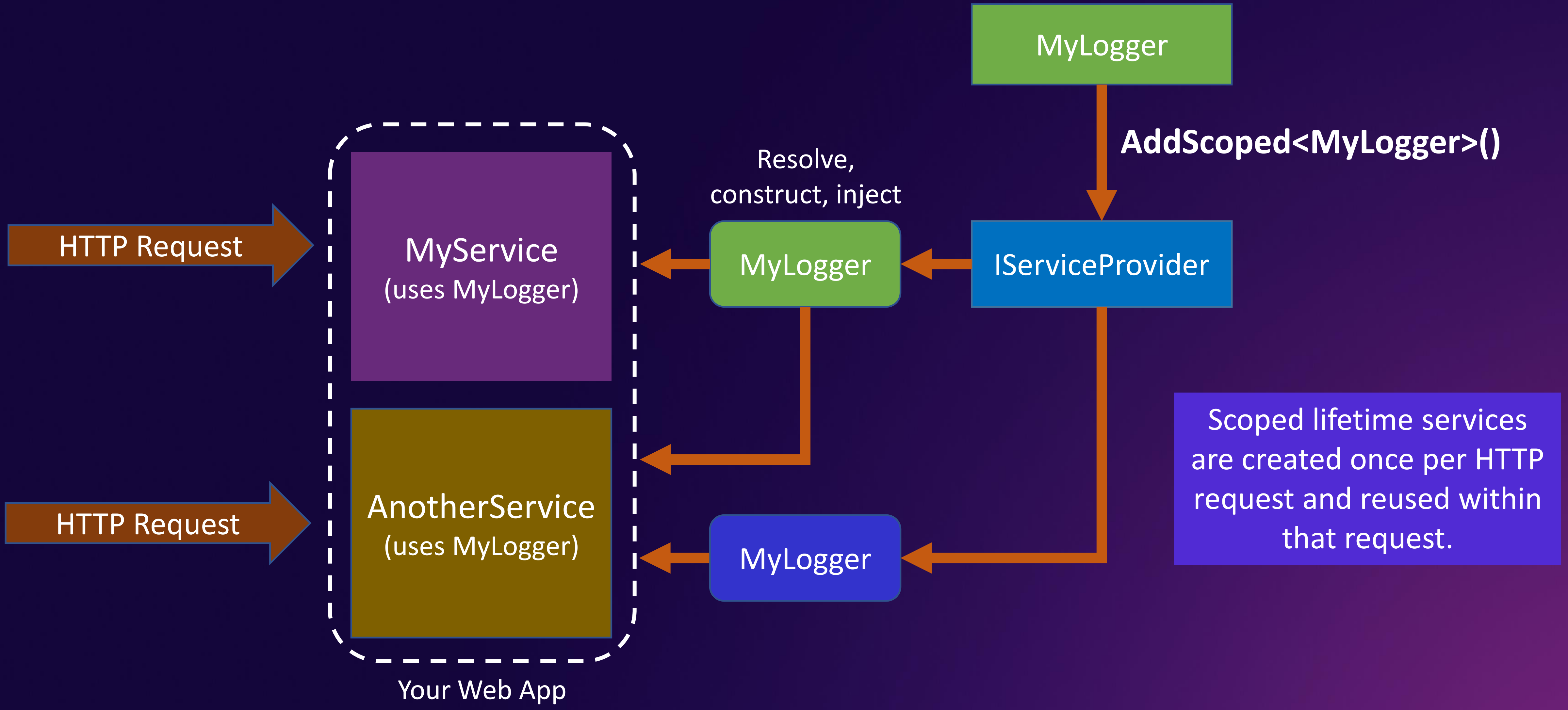
The Transient Service Lifetime



Transient lifetime services are created each time they are requested from the service container (IServiceProvider)

dotnetacademy.io

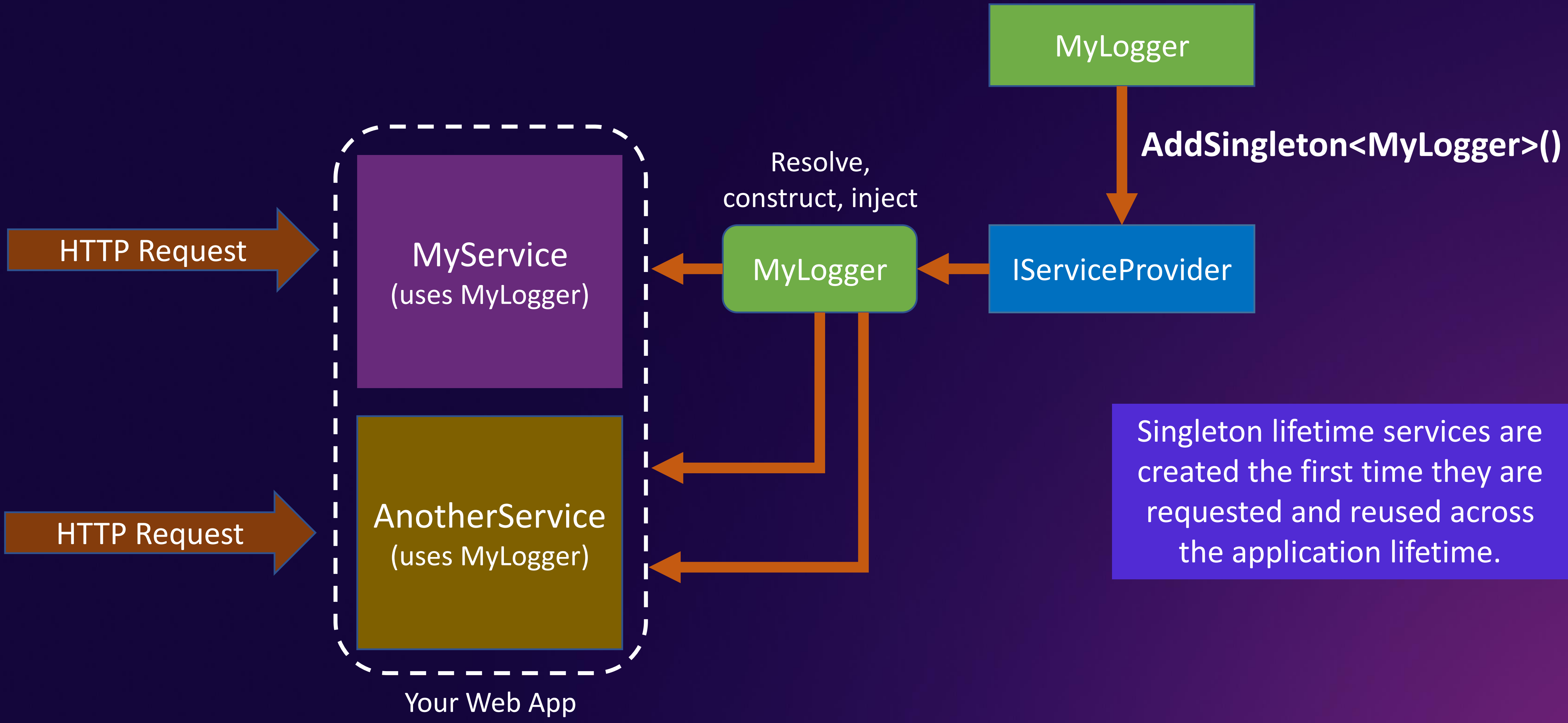
The Scoped Service Lifetime



Scoped lifetime services are created once per HTTP request and reused within that request.

dotnetacademy.io

The Singleton Service Lifetime

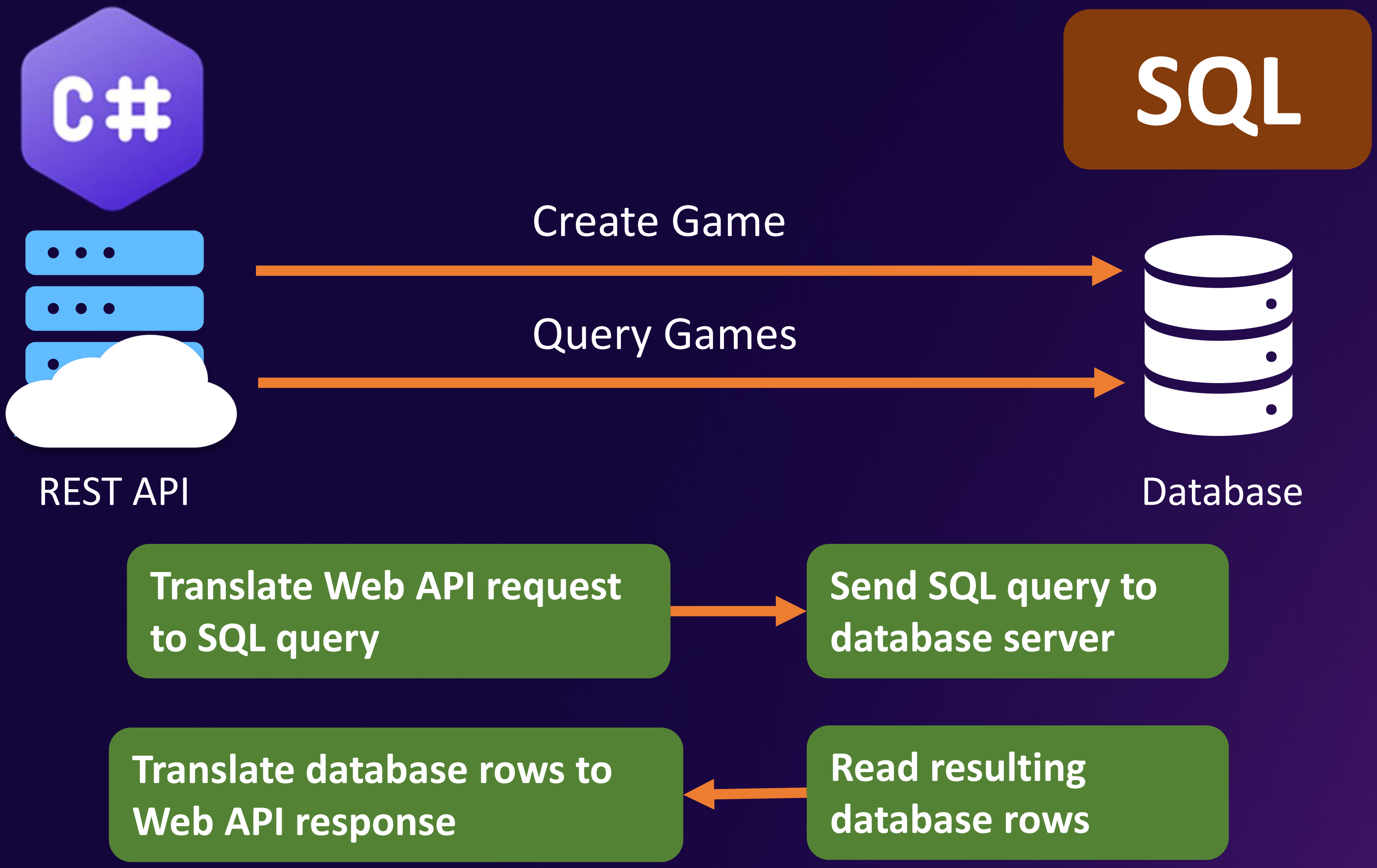


Singleton lifetime services are created the first time they are requested and reused across the application lifetime.

dotnetacademy.io

Introduction to Entity Framework Core

The Need For Object-Relational Mapping (O/RM)



SQL

Create Game

Query Games

REST API

Database

Translate Web API request to SQL query

Send SQL query to database server

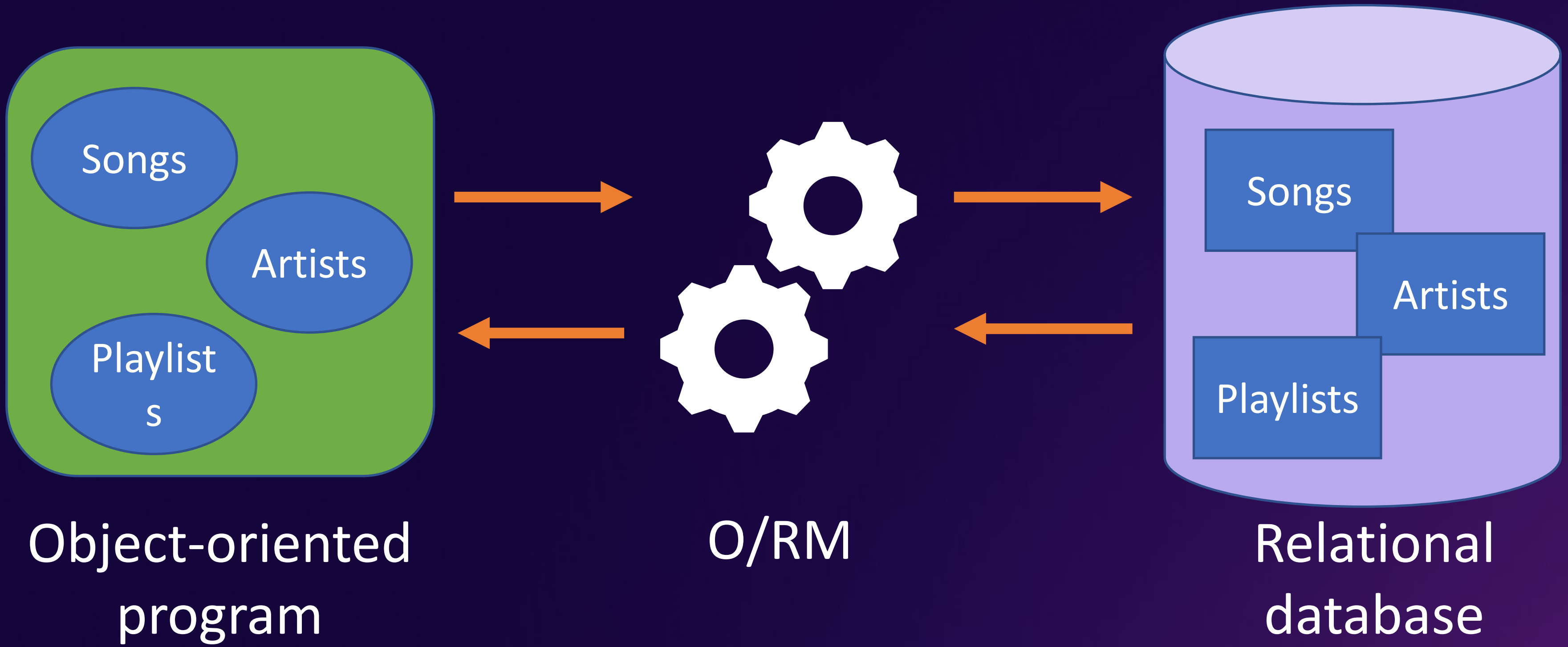
Read resulting database rows

Translate database rows to Web API response

Problems

- Need to learn new language
- Need a lot of additional data-access code
- Error prone
- Need to manually keep C# models in sync with DB tables

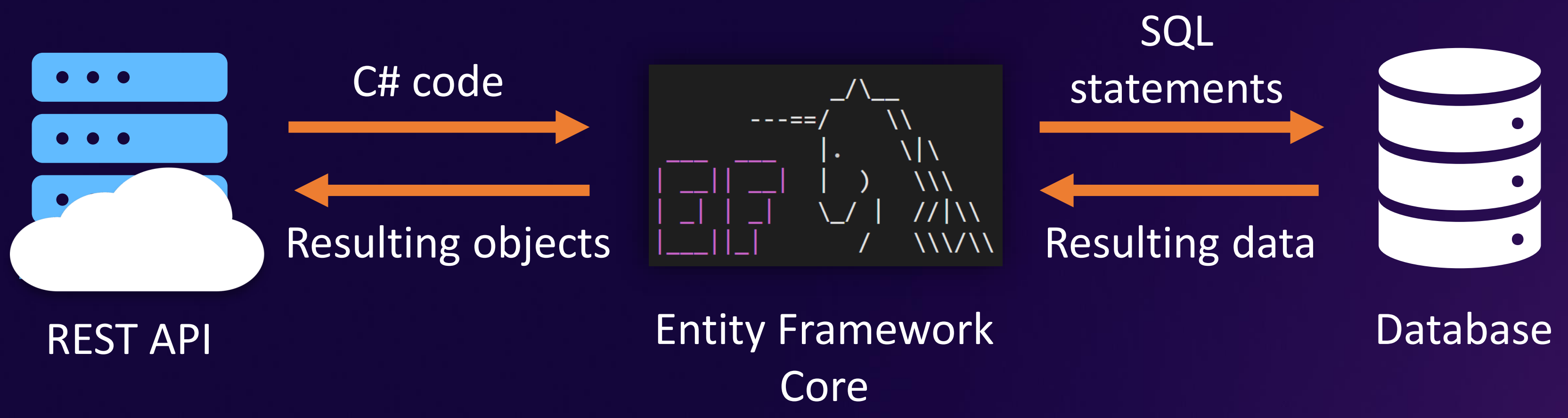
What is Object-Relational Mapping (O/RM)?



A technique for converting data between a relational database and an object-oriented program

What is Entity Framework Core?

A lightweight, extensible, open source and cross-platform object-relational mapper for .NET



```
                     _/\__
               ---==/    \\
         ___  ___   |.    \|\
        | __|| __|  |  )    \\\
        | _| | _|   \_/ |  //|\\
        |___||_|       /   \\\/\\
```

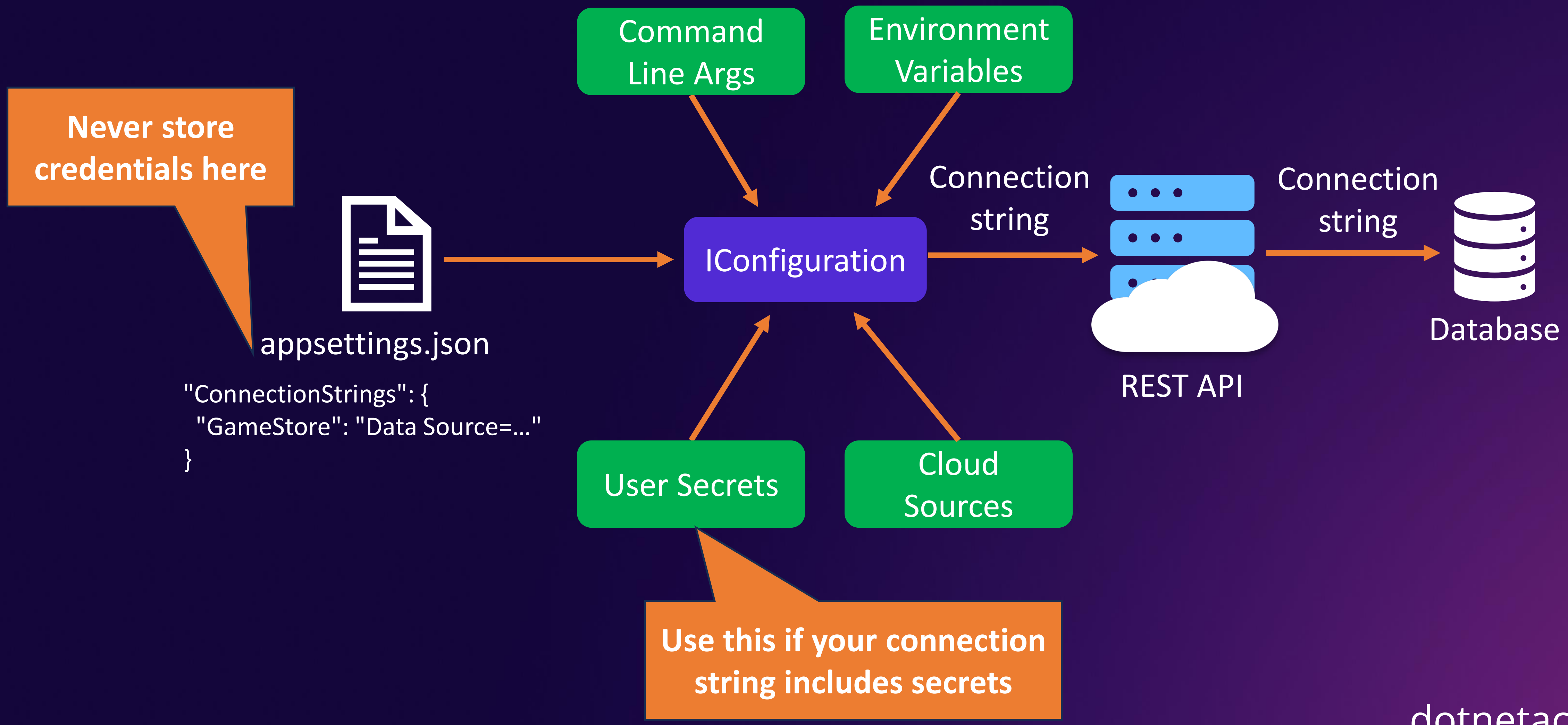
Benefits

- No need to learn a new language
- Minimal data-access code (LINQ)
- Tooling to keep C# models in sync with DB tables
- Change tracking
- Multiple database providers

dotnetacademy.io

ASP.NET Core Configuration

The ASP.NET Core Configuration System



Command Line Args

Environment Variables

Never store credentials here

appsettings.json

```
"ConnectionStrings": {
  "GameStore": "Data Source=..."
}
```

IConfiguration

Connection string

REST API

Connection string

Database

User Secrets

Cloud Sources

Use this if your connection string includes secrets

dotnetacademy.io