November 9-11, 2021

www.dotnetconf.net


# .NET Conf

Discover the world of .NET

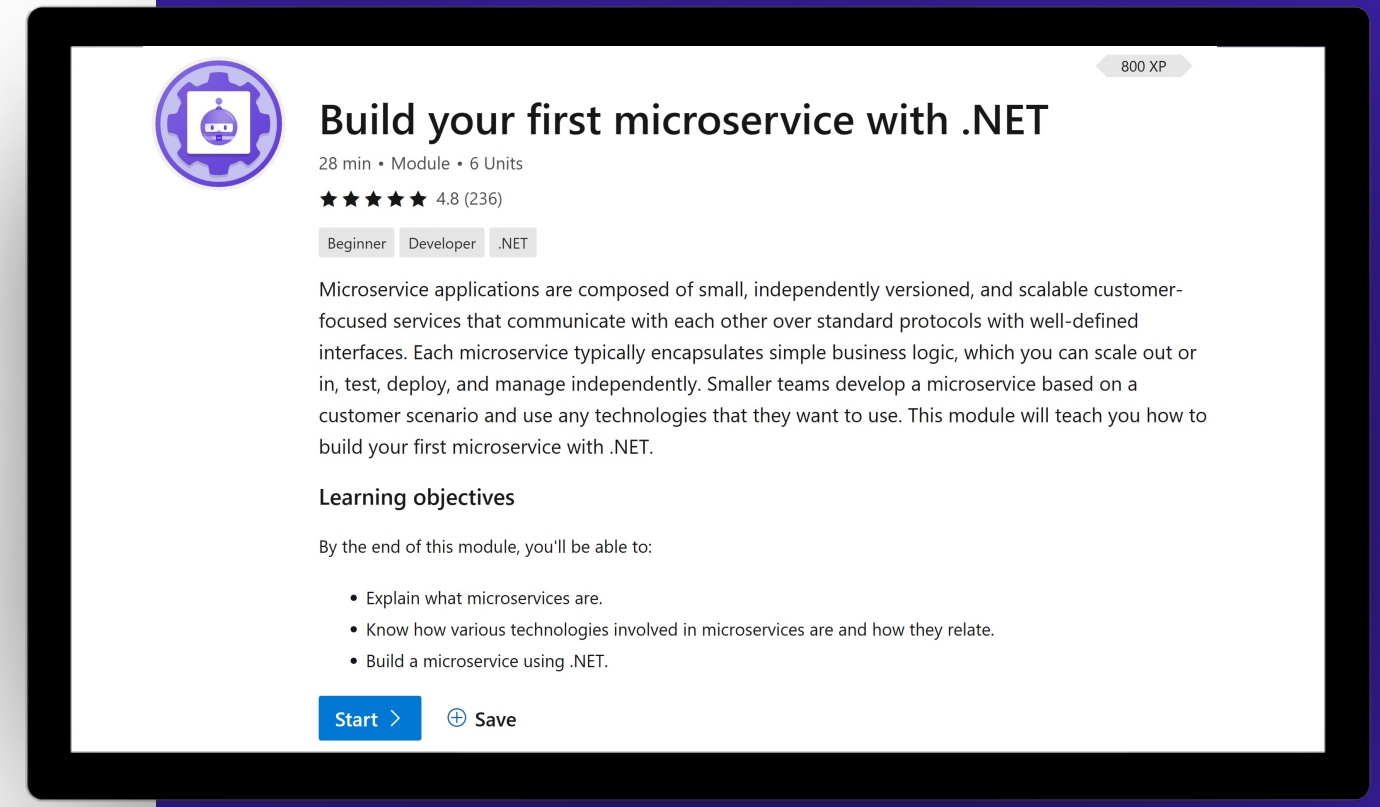
# Build your first microservice with .NET

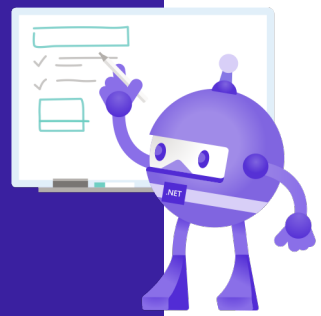
Nish Anil & Matt Soucoup

# /learn

**https://aka.ms/dotnet-conf-microservice**

Complete interactive learning exercises, watch videos, and practice and apply your new skills.





# Learning objectives

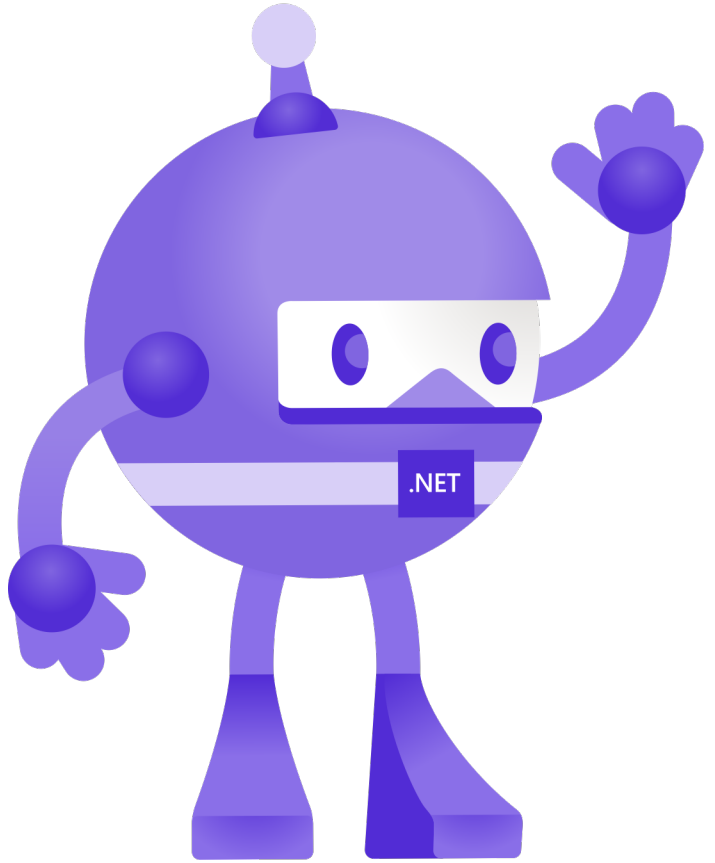
Explain what microservices are.

Know how various technologies involved in microservices are and how they relate.

Build a microservice using .NET.

# Live & interactive

Say “hi” and ask questions in the chat



# What are microservices?

Modern cloud applications need to be fast, agile, massively scalable, and reliable.

# What is a microservice architecture?

A microservices architecture is an approach in which a large application is decomposed into a set of smaller services.

Each microservice has a separate codebase, which can be managed by a small development team.

Microservices are small, independent, and loosely coupled.

Microservices are deployed independently. A team can update an existing microservice without rebuilding and redeploying the entire application.

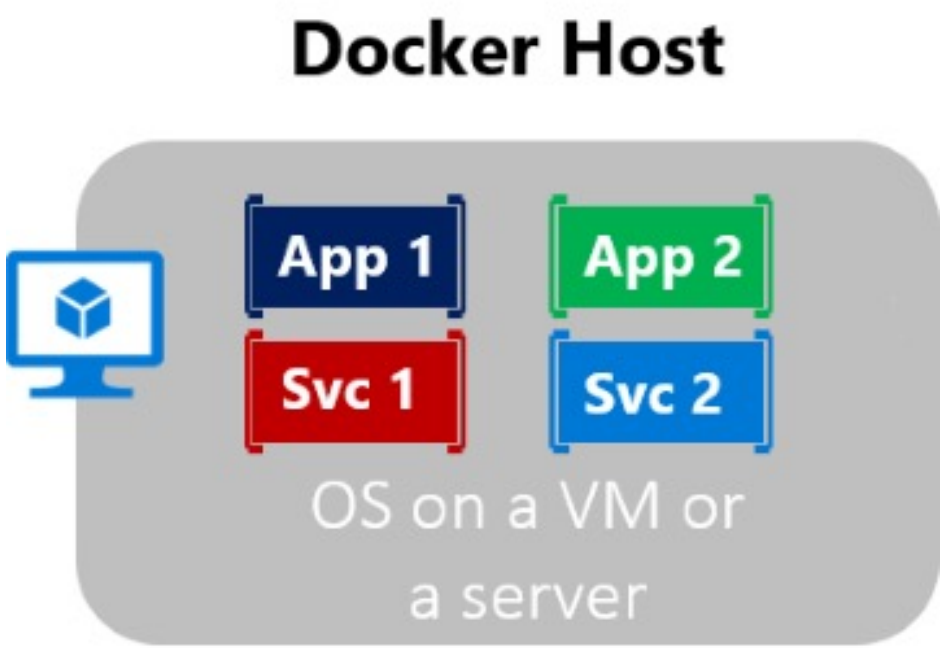
Microservices are responsible for persisting their data or external state in their respective databases. Unlike the monolithic architecture, microservices do not share databases.

Microservices communicate with each other by using well-defined APIs. Internal implementation details of each service are hidden from other services.

Supports polyglot programming. For example, microservices don't need to share the same technology stack, libraries, or frameworks.

# What role do containers play?

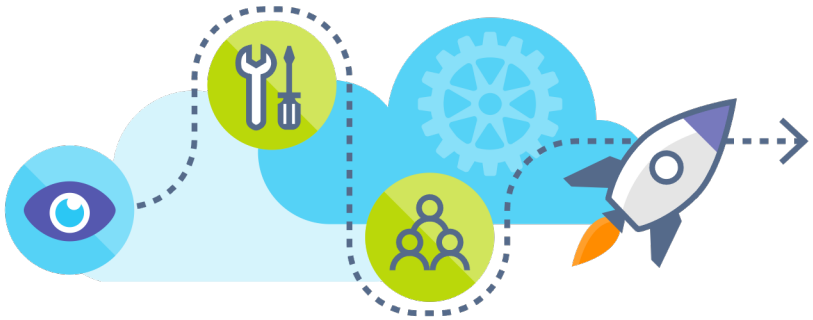
Containerization is an approach to software development in which an application or service, its dependencies, and its configuration (abstracted as deployment manifest files) are packaged together as a container image.




Containerizing an application is not the only way to deploy microservices. You could deploy microservices as individual services in Azure App Service, or via virtual machines, or any number of ways. However, containers are the tool that we'll use for the rest of this module to deploy our microservices in.

# Docker

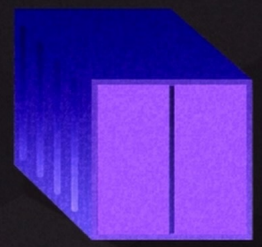
Docker is an open-source project for automating the deployment of applications as portable, self-sufficient containers that can run on the cloud or on-premises.



https://www.docker.com/

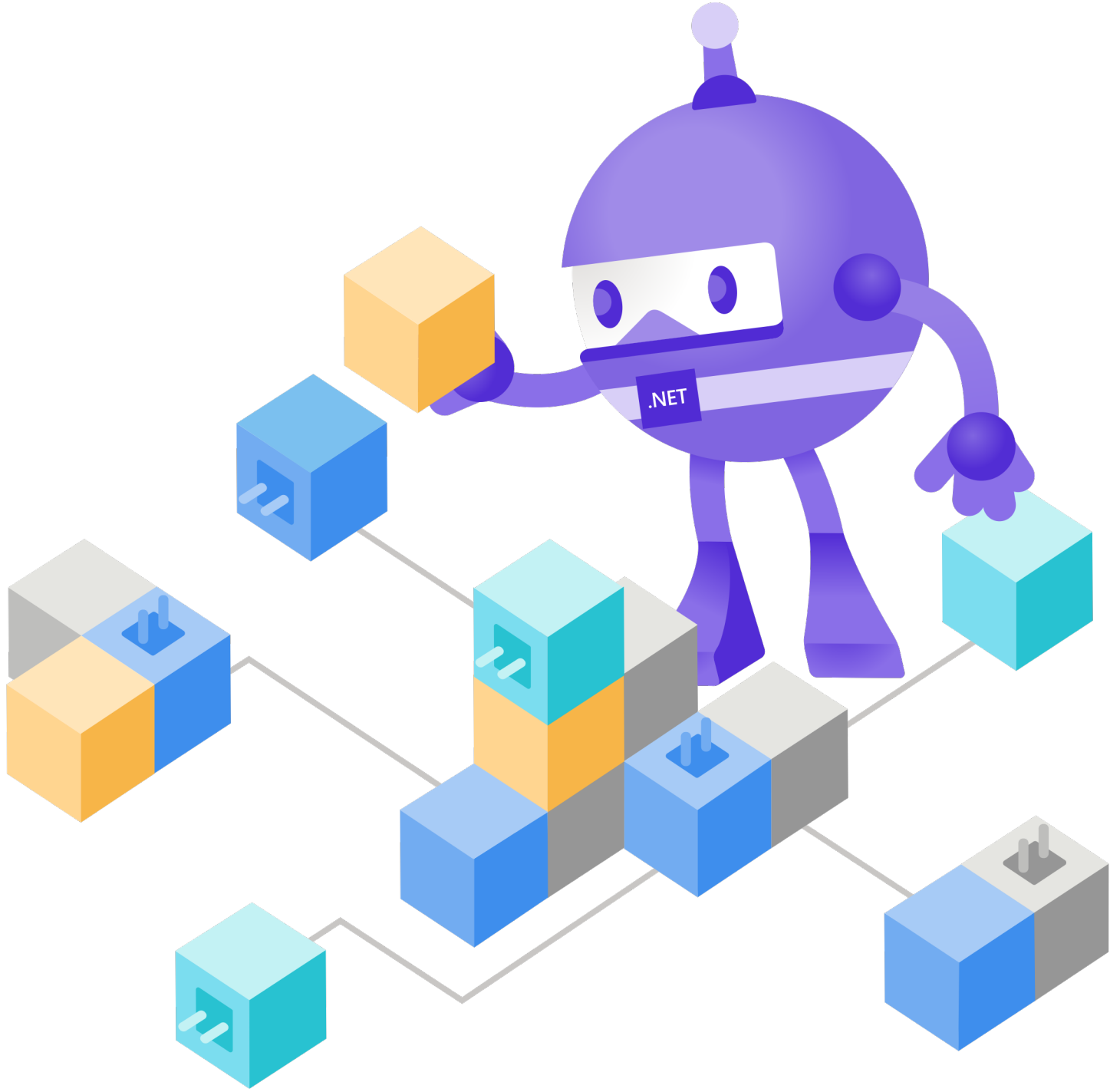
# What is an image?

When using Docker, a developer creates an app or service and packages it and its dependencies into a container image.



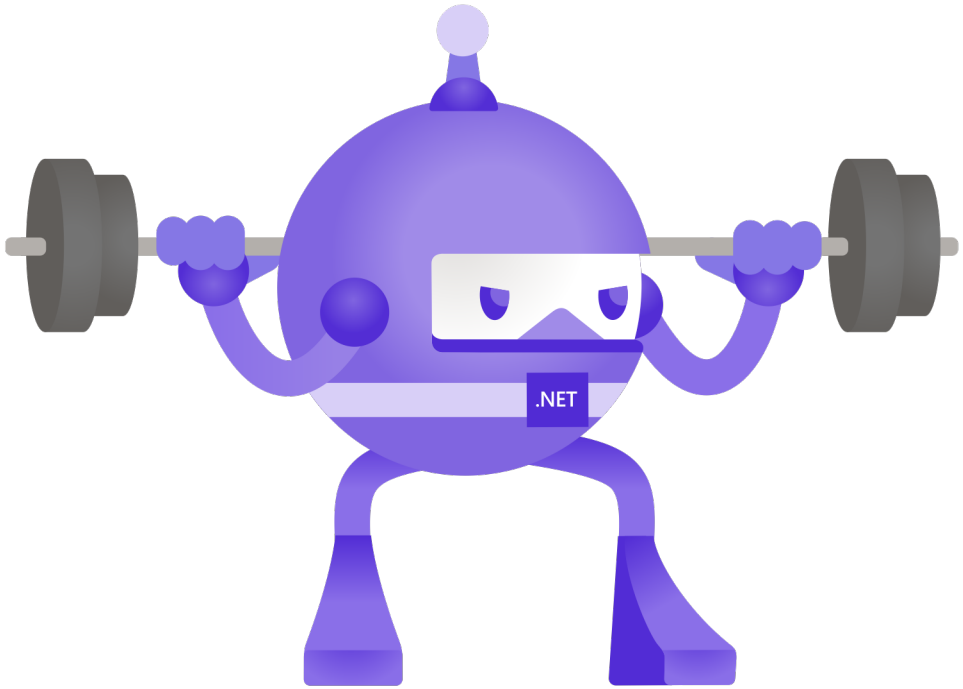
# Why build microservices in .NET?

Starting with .NET Core and continuing all the way through the present iterations, .NET was built to be cloud native first.



# Exercise

Build a Dockerfile for your microservice



# Build a Dockerfile for your microservice

In this exercise you will learn to create a microservice endpoint and containerize it using docker.

- Clone the code
- Create the Dockerfile for the back-end web API

Exercise - Build a Dockerfile for your microservice - Learn | Microsoft Docs

https://review.docs.microsoft.com/en-us/lear...

Build your first microservice with .NET / Exercise - Build a Dockerfile for your microservice

Previous | Unit 3 of 6 | Next

100 XP

# Exercise - Build a Dockerfile for your microservice

5 minutes

In this exercise you will learn to create a microservice endpoint and containerize it using docker.

> **Important**
>
> To complete this exercise, please download and install both the .NET SDK and Docker. You will also need a text editor, such as Visual Studio Code.

## Clone the code

The website and API have already been created for you. Clone the repository from GitHub to retrieve the code.

1. Open up a command prompt.
2. Change to a directory that you want the code to be downloaded to.
3. Run the following command to download, or clone, the sample repository:

```
Bash                                                    Copy
git clone https://github.com/MicrosoftDocs/mslearn-dotnetmicroservices
```

The code will download into a new folder named **mslearn-dotnetmicroservices**.

File Edit Selection View Go Run Terminal Help mslearn-dotnetmicroservices - Visual Studio Code

EXPLORER

MSLEARN-DOTNETMICROSERVICES

- .vscode
- backend
- finished-files
- frontend
- .dockerignore
- .gitignore
- CODE_OF_CONDUCT.md
- docker-compose.yml
- LICENSE
- LICENSE-CODE
- microservices.sln
- README.md
- SECURITY.md

OUTLINE

TIMELINE

Show All Commands Ctrl + Shift + P

Go to File Ctrl + P

Find in Files Ctrl + Shift + F

Start Debugging F5

Toggle Terminal Ctrl + `

migrate-net-6* | 0 0 | Nish | Live Share | microservices.sln | [Azurite Blob Service] | Spell

# What is a Dockerfile?

A Dockerfile is a text file that contains instructions on how to build a Docker image.

Dockerfile

```
FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
WORKDIR /src
COPY backend.csproj .
RUN dotnet restore
COPY . .
RUN dotnet publish -c release -o /app

FROM mcr.microsoft.com/dotnet/aspnet:6.0
WORKDIR /app
EXPOSE 80
EXPOSE 443
COPY --from=build /app .
ENTRYPOINT ["dotnet", "backend.dll"]
```

Exercise - Build a Dockerfile for your microservice - Learn | Microsoft Docs

https://review.docs.microsoft.com/en-us/lear...

Dockerfile Copy

```
FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
WORKDIR /src
COPY backend.csproj .
RUN dotnet restore
COPY . .
RUN dotnet publish -c release -o /app
```

This will perform the following steps sequentially when invoked:

- Pull the `mcr.microsoft.com/dotnet/sdk:6.0` image and name the image `build`.
- Set the working directory within the image to `/src`.
- Copy the file named **backend.csproj** found locally to the `/src` directory that was just created.
- Calls `dotnet restore` on the project.
- Copy everything in the local working directory to the image.
- Calls `dotnet publish` on the project.

4. Directly below the last line, enter this code:

Dockerfile Copy

```
FROM mcr.microsoft.com/dotnet/aspnet:6.0
WORKDIR /app
EXPOSE 80
EXPOSE 443
COPY --from=build /app .
ENTRYPOINT ["dotnet", "backend.dll"]
```

This will perform the following steps sequentially when invoked:

- Pull the `mcr.microsoft.com/dotnet/aspnet:6.0` image.
- Set the working directory within the image to `/app`.
- Exposes port 80 and 443.
- Copy everything from the `/app` directory of the **build** image created above into the app

File Edit Selection View Go Run Terminal Help Dockerfile - mslearn-dotnetmicroservices - Visual Studio...

Dockerfile M

backend > Dockerfile > ...

You, seconds ago | 1 author (You)

```
1  FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
2  WORKDIR /src
3  COPY backend.csproj .
4  RUN dotnet restore
5  COPY . .
6  RUN dotnet publish -c release -o /app
7
8  FROM mcr.microsoft.com/dotnet/aspnet:6.0
9  WORKDIR /app
10 EXPOSE 80
11 EXPOSE 443
12 COPY --from=build /app .
13 ENTRYPOINT ["dotnet", "backend.dll"]      You, seconds ago • Uncommitted chang
```

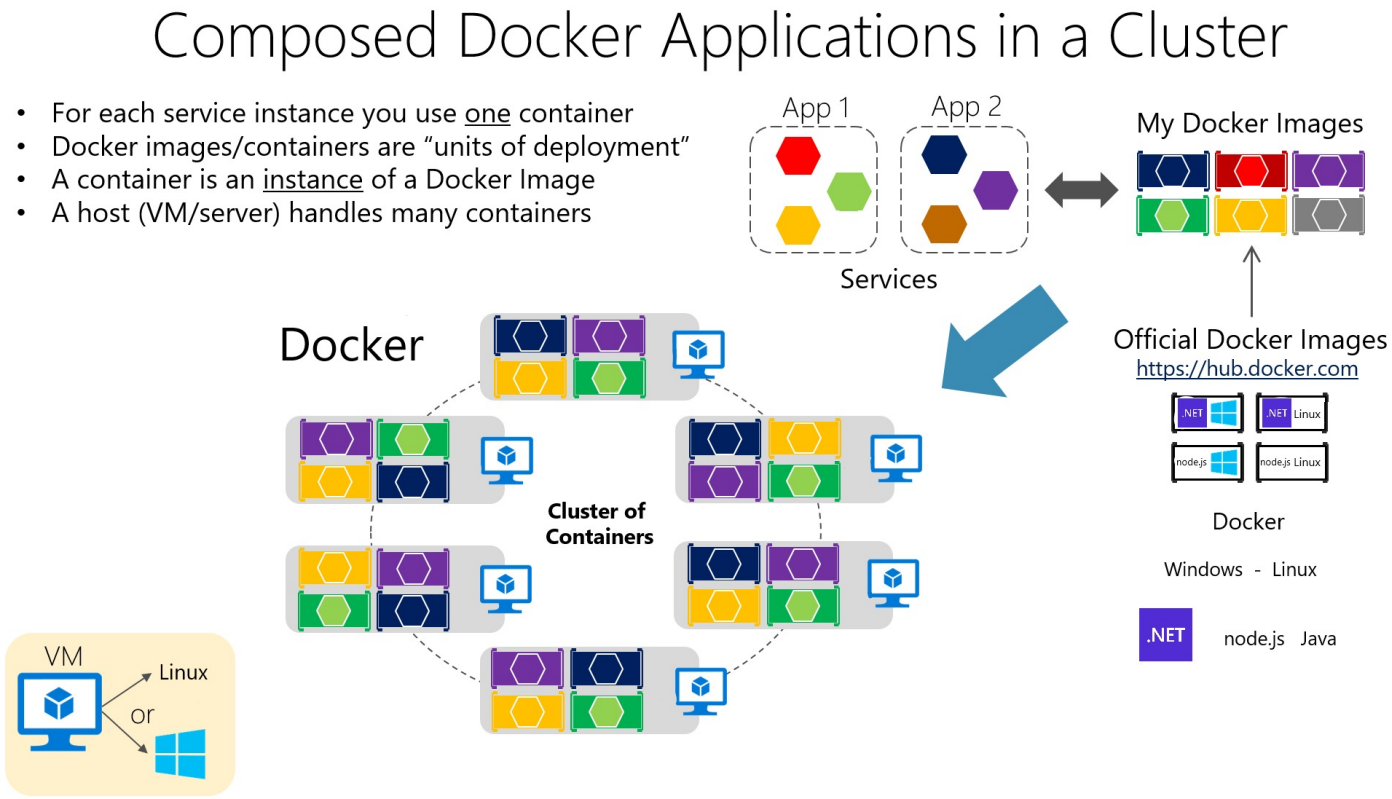
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL pwsh

```
PowerShell 7.1.5
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

Loading personal and system profiles took 759ms.
 mslearn-dotnetmicroservices  migrate-net-6 ≡ +1 ~1  6.0.100
```

migrate-net-6* 0 0 Nish Live Share microservices.sln UTF-8 CRLF Dockerfile Spell

# What is an orchestrator?

Using orchestrators for production-ready applications is essential if your application is based on microservices or simply split across multiple containers.



# Docker Compose

A full-featured orchestrator might be too much for the simple website that the Contoso Pizza is dealing with.

Docker compose tool gets installed by default with the Docker for desktop which makes it easier to run all containers along with their dependencies locally. For production scenarios, we recommend using Orchestrators that gives finer control over automatic deployment, scaling and management of containers.

# Create a Docker Compose file

The Contoso Pizza has two services that they'd like to group together to build and deploy as a single unit.

- Create the docker-compose file

Microservices orchestration - Learn | Microsoft Docs

https://review.docs.microsoft.com/en-us/lear...

Build your first microservice with .NET / Microservices orchestration

Previous | Unit 4 of 6 | Next

Build your first microservice with .NET

| | |
|---|---|
| Introduction | 1 min |
| What are microservices? | 7 min |
| Exercise - Build a Dockerfile for your microservice | ✓ |
| Microservices orchestration | 7 min |
| Exercise - Create a Docker Compose file | 5 min |
| Summary | 3 min |

200 XP

# Microserv

7 minutes

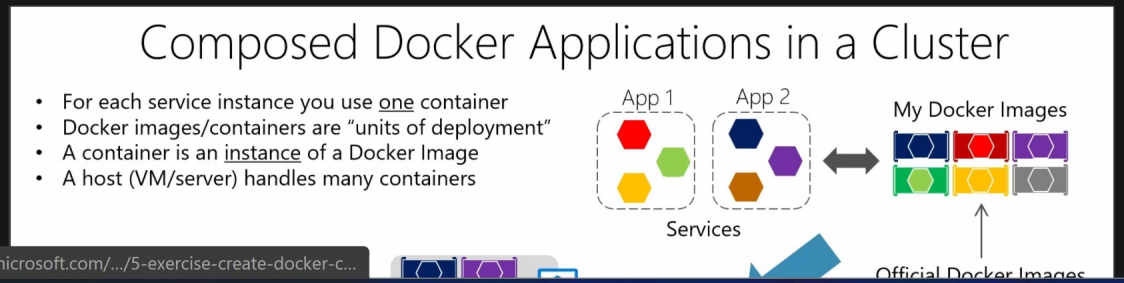
The Contoso Pizza lov so far. The overall web applicatio

As more and more ser plex to scale out and manage. Orchestr

## What is an orchestrator?

Using orchestrators for production-ready applications is essential if your application is based on microservices or simply split across multiple containers. As introduced previously, in a microservice-based approach, each microservice owns its model and data so that it will be autonomous from a development and deployment point of view. These kinds of systems are complex to scale out and manage; therefore, you absolutely need an orchestrator if you want to have a production-ready and scalable multi-container application.

The image below illustrates deployment into a cluster of an application composed of multiple microservices (clusters).




...microsoft.com/.../5-exercise-create-docker-c...

File Edit Selection View Go Run Terminal Help Dockerfile - mslearn-dotnetmicroservices - Visual Studio...

Dockerfile M

backend > Dockerfile > ...

You, seconds ago | 1 author (You)

```
1  FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
2  WORKDIR /src
3  COPY backend.csproj .
4  RUN dotnet restore
5  COPY . .
6  RUN dotnet publish -c release -o /app
7
8  FROM mcr.microsoft.com/dotnet/aspnet:6.0
9  WORKDIR /app
10 EXPOSE 80
11 EXPOSE 443
12 COPY --from=build /app .
13 ENTRYPOINT ["dotnet", "backend.dll"]      You, seconds ago • Uncommitted chang
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL pwsh

```
=> CACHED [build 6/6] RUN dotnet publish -c release -o /app                    0.0s
=> CACHED [stage-1 3/3] COPY --from=build /app .                               0.0s
=> exporting to image                                                          0.0s
=> => exporting layers                                                         0.0s
=> => writing image sha256:6e6402db9a2654e1b8f609dca95c201295ed39c8e0f4b1b8559d50d  0.0s
=> => naming to docker.io/library/pizzabackend                                 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how t
o fix them
backend migrate-net-6 ≡ +1 ~1 6.0.100 docker images
REPOSITORY     TAG       IMAGE ID       CREATED          SIZE
pizzabackend   latest    6e6402db9a26   3 minutes ago    212MB
backend migrate-net-6 ≡ +1 ~1 6.0.100 docker run -it --rm -p 5200:80
 --name pizzabackendcontainer pizzabackend
{"EventId":14,"LogLevel":"Information","Category":"Microsoft.Hosting.Lifetime","Message":"
Now listening on: http://[::]:80","State":{"Message":"Now listening on: http://[::]:80","a
ddress":"http://[::]:80","{OriginalFormat}":"Now listening on: {address}"}}
{"EventId":0,"LogLevel":"Information","Category":"Microsoft.Hosting.Lifetime","Message":"A
pplication started. Press Ctrl\u002BC to shut down.","State":{"Message":"Application start
ed. Press Ctrl\u002BC to shut down.","{OriginalFormat}":"Application started. Press Ctrl\u
002BC to shut down."}}
{"EventId":0,"LogLevel":"Information","Category":"Microsoft.Hosting.Lifetime","Message":"H
osting environment: Production","State":{"Message":"Hosting environment: Production","envN
ame":"Production","{OriginalFormat}":"Hosting environment: {envName}"}}
{"EventId":0,"LogLevel":"Information","Category":"Microsoft.Hosting.Lifetime","Message":"C
ontent root path: /app","State":{"Message":"Content root path: /app","contentRoot":"/app",
"{OriginalFormat}":"Content root path: {contentRoot}"}}
```

migrate-net-6* ⊗ 0 ⚠ 0 Nish Live Share microservices.sln UTF-8 CRLF Dockerfile Spell

# What is a Docker-Compose file?

A config file that docker-compose uses to build and run multiple images.

**docker-compose.yml**

```yaml
version: '3.4'

services:

  frontend:
    image: pizzafrontend
    build:
      context: frontend
      dockerfile: Dockerfile
    environment:
      - backendUrl=http://backend
    ports:
      - "5902:80"
    depends_on:
      - backend
  backend:
    image: pizzabackend
    build:
      context: backend
      dockerfile: Dockerfile
    ports:
      - "5900:80"
```

Exercise - Create a Docker Compose file - Learn | Microsoft Docs

https://review.docs.microsoft.com/en-us/lear...

# Create the docker-compose file

1. Use Visual Studio Code to open the **docker-aspnet-pizza** folder you cloned.

2. In the topmost folder (the same folder with README.md), open then named **docker-compose.yml**. This file will be empty.

3. Inside of that file, add the following code:

yml Copy

```yml
version: '3.4'

services:

  frontend:
    image: pizzafrontend
    build:
      context: frontend
      dockerfile: Dockerfile
    environment:
      - backendUrl=http://backend
    ports:
      - "5902:80"
    depends_on:
      - backend
  backend:
    image: pizzabackend
    build:
      context: backend
      dockerfile: Dockerfile
    ports:
      - "5900:80"
```

This code does several things:

- First it creates the frontend website, naming it **pizza frontend**. It tells Docker to build it, pointing to the Dockerfile found in the **frontend** folder. Then it sets an environment variable for the website: `backendUrl=http://backend`. Finally it opens a port and declares it depends on

File Edit Selection View Go Run Terminal Help Startup.cs - mslearn-dotnetmicroservices - Visual Studio...

EXPLORER

MSLEARN-DOTNETMICR...

- .vscode
- backend
- finished-files
- frontend
  - bin
  - obj
  - Pages
  - Properties
  - wwwroot
  - appsettings.Development.json
  - appsettings.json
  - Dockerfile
  - frontend.csproj
  - PizzaClient.cs
  - PizzaInfo.cs
  - Program.cs
  - Startup.cs
- .dockerignore
- .gitignore
- CODE_OF_CONDUCT.md
- docker-compose.yml
- LICENSE
- LICENSE-CODE
- microservices.sln
- README.md
- SECURITY.md

OUTLINE

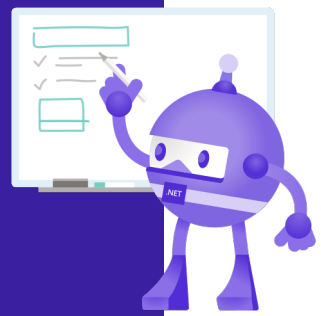
TIMELINE

erfile M | docker-compose.yml | Startup.cs

frontend > Startup.cs > ...

```cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using System.Net.Http;

namespace frontend
{
    public class Startup
    {
        public Startup(IConfiguration configuratic
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get;

        // This method gets called by the runtime.
        public void ConfigureServices(IServiceColl
        {
            services.AddRazorPages();

            services.AddHttpClient<PizzaClient>(cl
            {
                var baseAddress = new Uri(Configur

                client.BaseAddress = baseAddress;
            });

        }

        // This method gets called by the runtime.
        public void Configure(IApplicationBuilder
        {
            if (env.IsDevelopment())
            {
```

migrate-net-6* 0 0 Nish Live Share microservices.sln UTF-8 CRLF C# Spell



# Summary

Explain what microservices are.

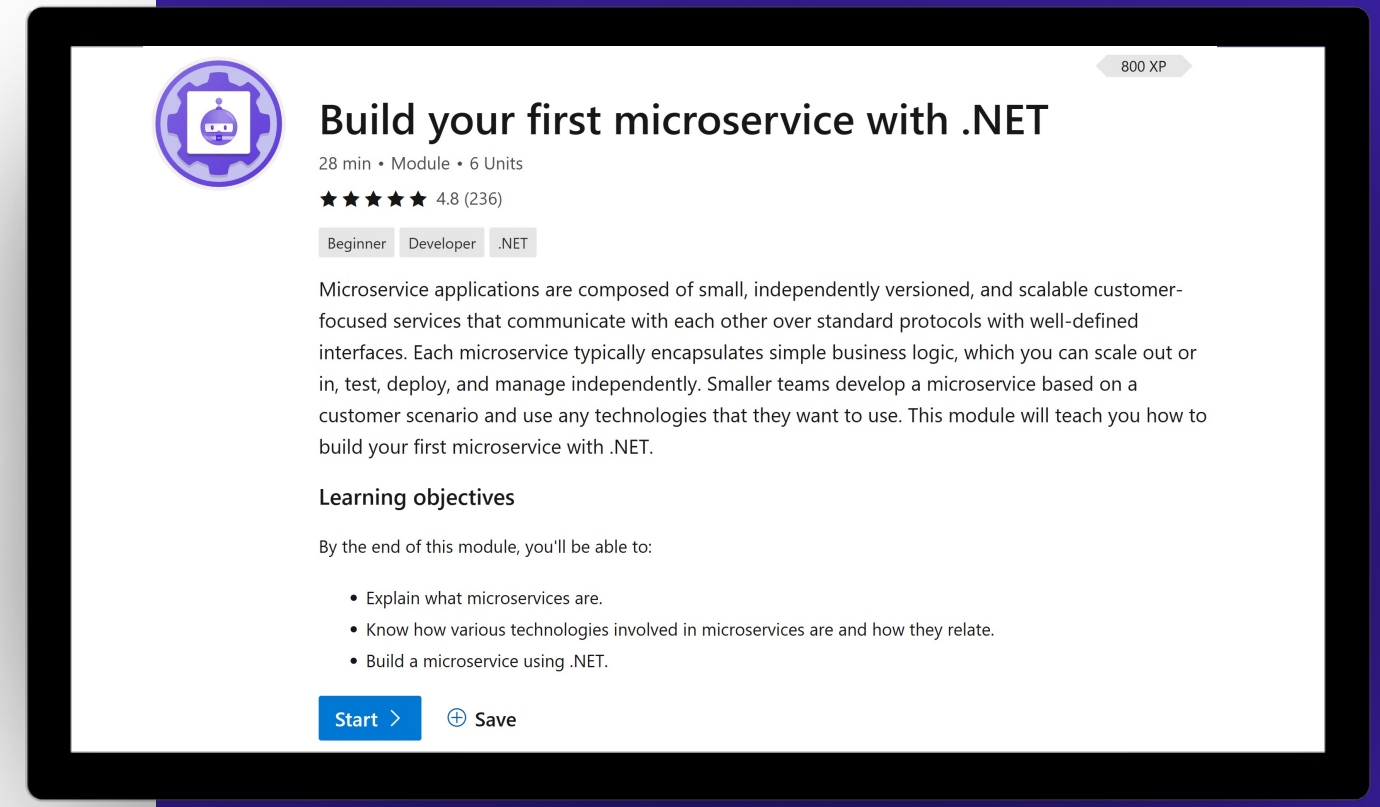
Know how various technologies involved in microservices are and how they relate.

Build a microservice using .NET.

# /learn

**https://aka.ms/TBD**

Complete interactive learning exercises, watch videos, and practice and apply your new skills.



# Resources

## References

aka.ms/learn-live-microservices

LEARN Live

# Modern Web Development with .NET 6

5-part series, starting Monday, November 15 @ 2:00pm Pacific

**Register Now :**

**https://aka.ms/LearnLive-MWD-withdotNET6**

Scott Hanselman

Maira Wenzel

Jeff Fritz

Jon Galloway

James Montemagno

Matt Soucoup

# Q&A

# Thank you!

Nish Anil & Matt Soucoup