Dezvoltarea Aplicațiilor Web utilizând ASP.NET Core MVC
Curs 7 – Docker

Cuprins

Docker 2
Ce este Docker 2
Instalare Docker 3
Windows 3
MACOS 5
Linux 6
Configurarea unui proiect ASP.NET Core 9.0 pentru Docker 7

Docker

Ce este Docker

Docker este o platformă care permite dezvoltatorilor să împacheteze aplicațiile și dependențele acestora în containere, asigurând un comportament consistent în diferite medii. Containerele pot fi rulate pe orice sistem de operare care suportă Docker, indiferent de configurațiile software sau hardware ale sistemului gazdă. Astfel, Docker simplifică procesul de rulare, testare și implementare a aplicațiilor în diferite medii, nemaiexistând probleme de compatibilitate.

Avantajele utilizării Docker:

- **Portabilitate și consistență între mediile de dezvoltare**
  - Docker permite rularea aceleași aplicații, în același mod, pe diferite medii
- **Scalabilitate și rapiditate**
  - Containerele sunt foarte rapide de lansat și de replicat, ceea ce permite scalarea rapidă a aplicațiilor. În comparație cu mașinile virtuale, care pot dura câteva minute să pornească, containerele Docker pot porni în câteva secunde.
- **Managementul dependențelor**
  - Docker permite includerea tuturor dependențelor aplicației (biblioteci, configurații etc.) direct în container.
- **Proces simplificat de dezvoltare și colaborare**
  - Folosind containere Docker, echipele de dezvoltare pot lucra pe aceeași versiune a aplicației și cu aceleași configurații, indiferent de sistemele lor locale.

- **Îmbunătățirea securității**
  - Docker izolează aplicațiile în containere separate, astfel încât aplicațiile nu au același spațiu de memorie și nu au acces direct una la alta. Această izolare crește securitatea.

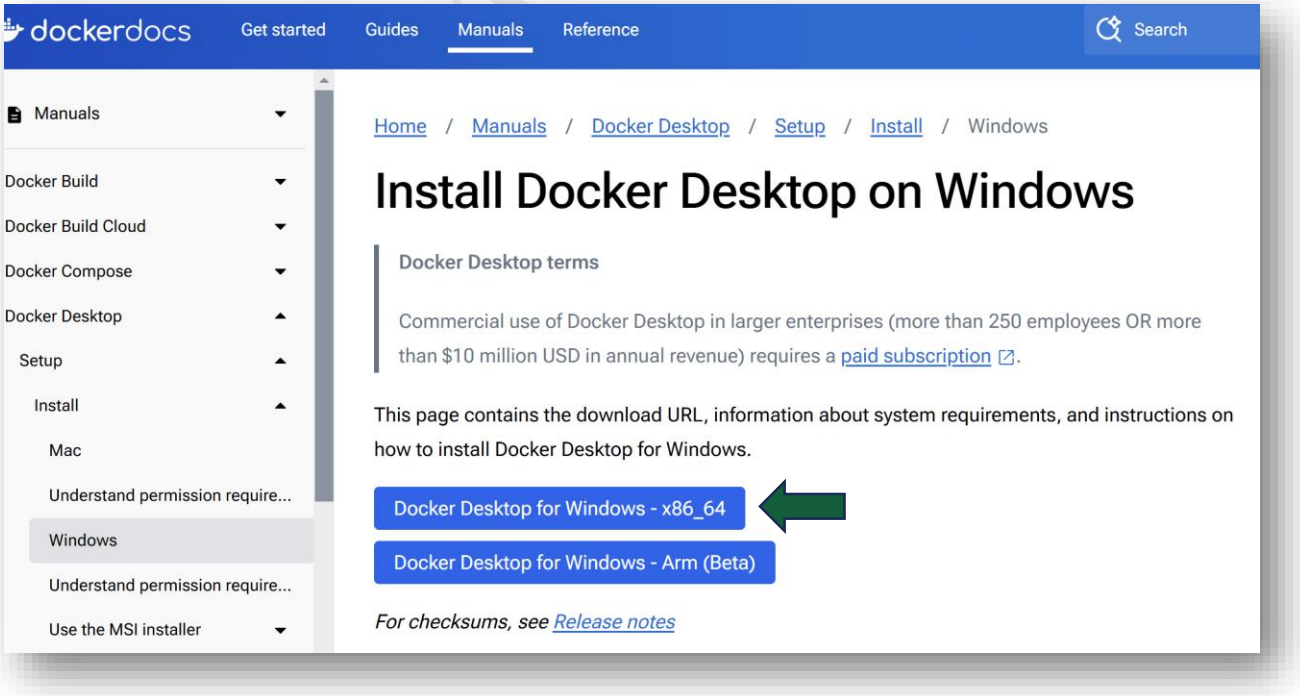
Acest curs cuprinde instalarea Docker, configurarea unui proiect ASP.NET Core 9.0 pentru Docker și rularea proiectului atât pe Windows, cât și pe macOS sau Linux.

Instalare Docker

Windows

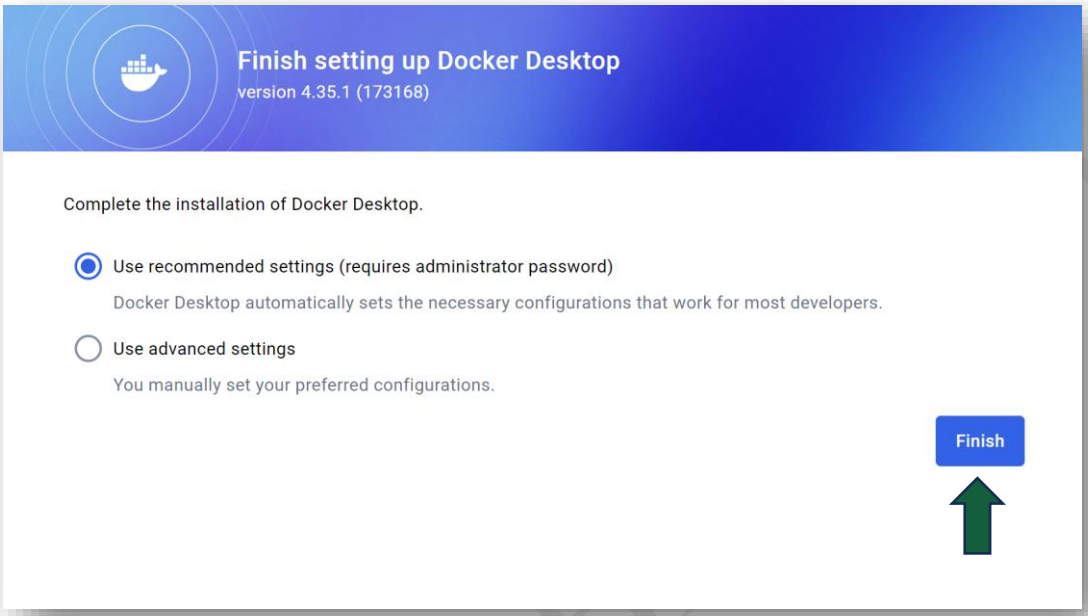
https://docs.docker.com/desktop/setup/install/windows-install/

PASUL 1 – Descărcați Docker Desktop – accesați pagina Docker Desktop pentru **Windows** și descărcați programul de instalare.

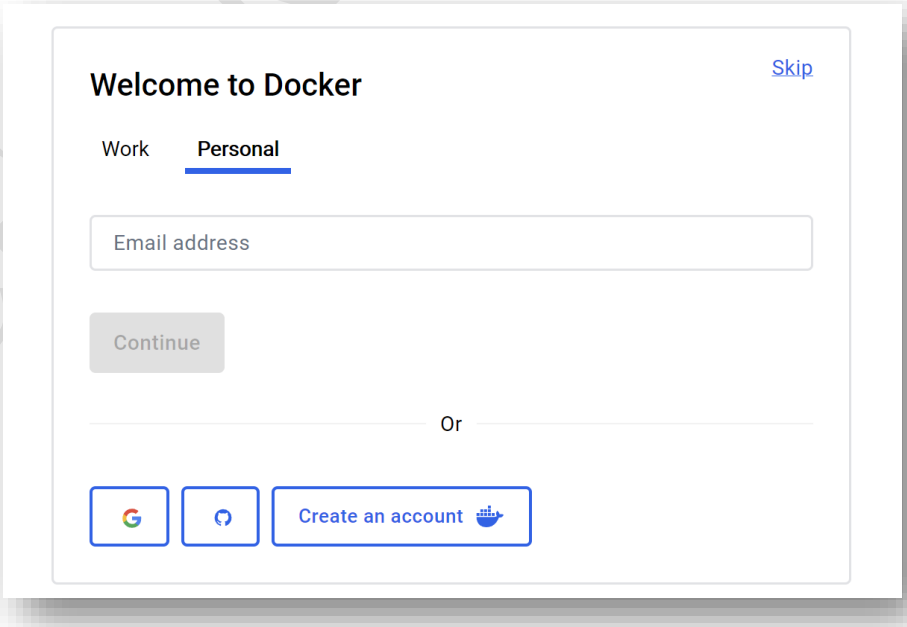


PASUL 2 – Instalați Docker Desktop – rulați programul de instalare descărcat și urmați instrucțiunile de pe ecran. La final o să fie necesară repornirea calculatorului.

După reponire, o să apară următorul ecran:



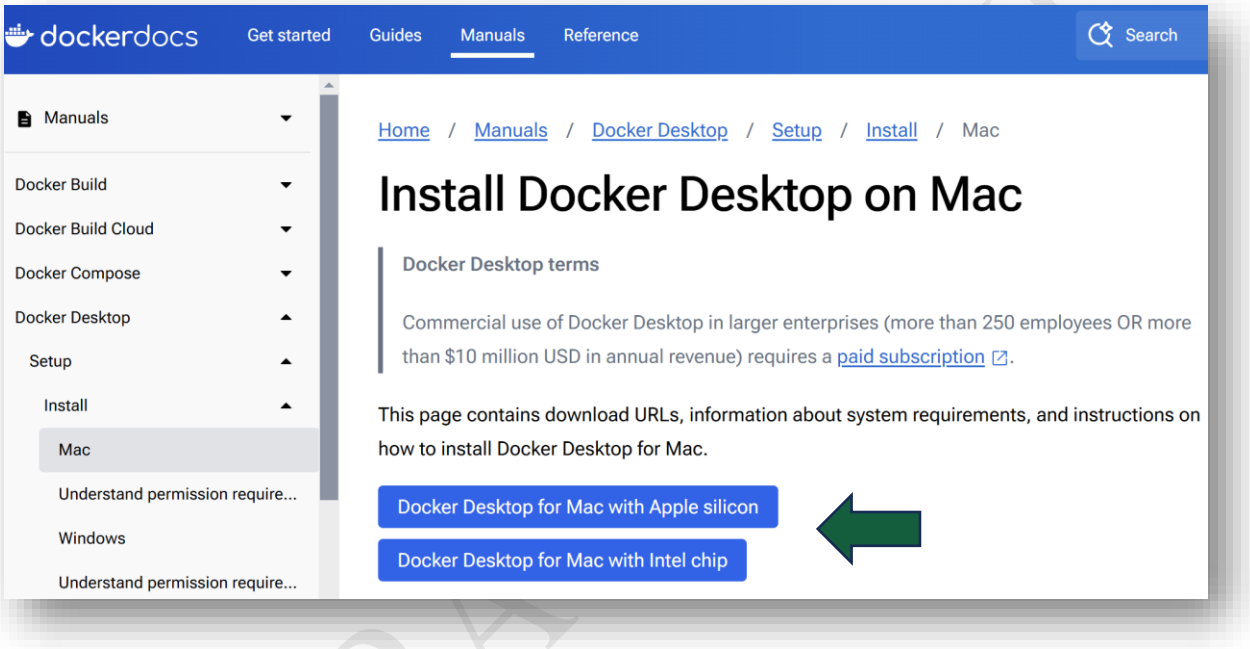
PASUL 3 – Se realizează un cont, utilizând opțiunea ***Personal***



MACOS

https://docs.docker.com/desktop/setup/install/mac-install/

PASUL 1 – Descărcați Docker Desktop - accesați pagina Docker Desktop pentru **MAC** și descărcați programul de instalare potrivit arhitecturii (Intel sau Apple Silicon).



PASUL 2 – Instalați Docker Desktop

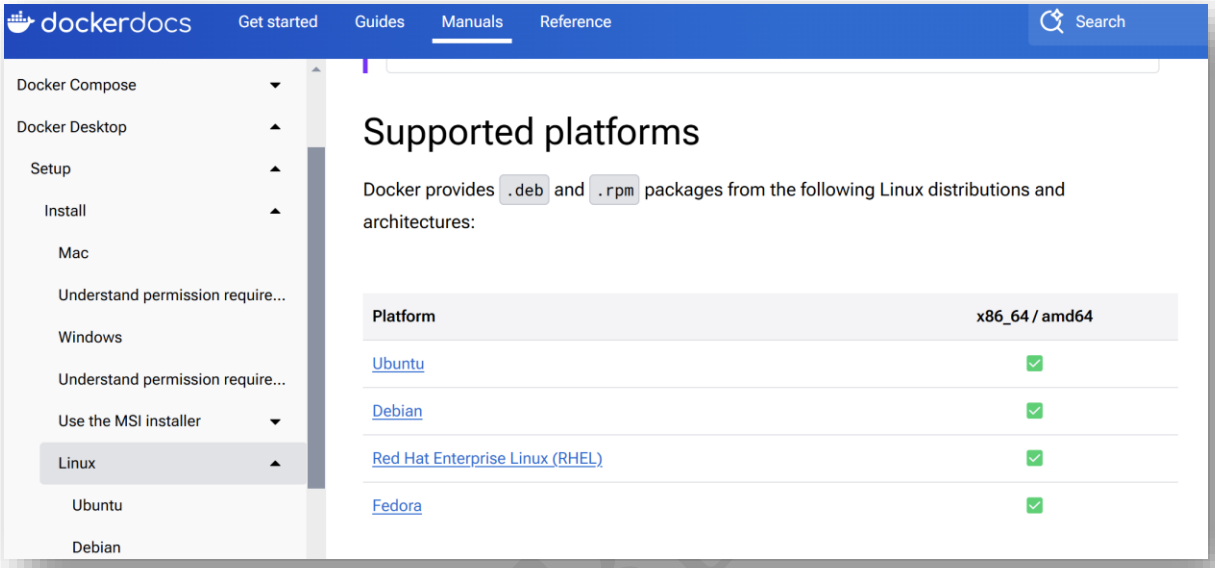
PASUL 3 – Deschideți fișierul .dmg descărcat și mutați pictograma Docker în folderul Applications

PASUL 4 – Lansați Docker Desktop - deschideți Docker din folderul Applications

Linux

https://docs.docker.com/desktop/setup/install/linux/

PASUL 1 – Se alege versiunea de Linux necesară



PASUL 2 – Se descarcă și instalează, utilizând comenzile din terminal, în funcție de distribuția de Linux. Exemplu pentru UBUNTU - https://docs.docker.com/desktop/setup/install/linux/ubuntu/

Configurarea unui proiect ASP.NET Core 9.0 pentru Docker

Se descarcă arhiva de pe site, numită **Folder_Docker.**

Initializare proiect nou

```
docker compose run --rm sdk dotnet new mvc -n DockerProject -o DockerProject -f net9.0 --auth Individual
```

Descriere comenzi:

- `docker compose run` - rulează o comandă one-off în containerul unui serviciu
- `--rm` - șterge automat containerul după ce comanda se termină
- `sdk` - numele serviciului din docker-compose.yml (folosește imaginea .NET SDK 9.0)
- `dotnet new webapp` - comanda .NET CLI pentru crearea unui nou proiect de tip web application
- `-n DockerProject` - numele proiectului
- `-o DockerProject` - directorul de output unde va fi creat proiectul
- `-f net9.0` - framework-ul țintă (.NET 9.0)
- `--auth Individual` - adaugă autentificare cu conturi individuale de utilizator (Identity)

Comanda creează un proiect ASP.NET Core web application cu autentificare în directorul DockerProject, folosind containerul Docker cu .NET SDK 9.0. Containerul se șterge automat după ce comanda se termină.

Rularea initiala a proiectului

Primul pas este ștergerea folderul Migrations din interiorul folderului Data.

Pentru rularea proiectului se folosește comanda: `docker compose up -d`

Instalarea pachetelor

Instalarea pachetelor se face din terminalul aflat în tab-ul **EXEC** din **GUI Docker**. Se navighează la containerul **app** și se acceseaza tab-ul **exec**. În cadrul acestui tab se adaugă pachetele necesare:

```
dotnet tool install --global dotnet-ef --version 9.0.11

dotnet add package Pomelo.EntityFrameworkCore.MySql
```

Deoarece migrațiile au fost create pentru SQLite, acestea o să fie refăcute pentru MySQL.

1. Se configurează **ApplicationDbContext** pentru **MySQL** prin adăugarea următoarelor linii de cod:

```
protected override void OnModelCreating(ModelBuilder builder)
    {
        base.OnModelCreating(builder);

        // Configure string properties for MySQL compatibility
        foreach (var entityType in builder.Model.GetEntityTypes())
        {
            foreach (var property in entityType.GetProperties())
            {
                if (property.ClrType == typeof(string))
                {
                    var maxLength = property.GetMaxLength();
```

```
                    if (maxLength == null)
                    {
                        // Set default max length for string primary keys and
foreign keys
                        if (property.IsKey() || property.IsForeignKey())
                        {
                            property.SetMaxLength(255);
                        }
                    }
                }
            }
        }
```

2. Se schimbă driver-ul cu MySQL

options.UseMySql(connectionString, new MySqlServerVersion(new Version(8, 4))));

și se adaugă **ConnectionString-ul** corect în Program.cs

```
"Database=dockerproject;Host=db;Port=3306;User=dockeruser;P
assword=dockerpass;"
```

Se repornește containerul docker al app.

3. Se generează și se aplică noile migrații, folosind comenzile de mai jos:

```
docker compose stop app

docker compose run --rm app sh -c "cd /src/DockerProject
&& dotnet ef migrations add InitialMigration"

docker compose start app
```

Se execută comanda următoare în containerul **app** – în tab-ul **exec**:

```
dotnet ef database update
```

Rularea comenzilor din CLI

Pentru a rula comenzi din CLI în containerul app, se utilizează comanda:

```
docker compose exec app <comanda>
```

Dockerfile explicați – Explicație Docker Compose & Dockerfile

- **docker-compose.yml** definește trei servicii principale:
  - **sdk**: folosit pentru comenzi administrative .NET (generează/restore proiectul)
  - **app**: rulează aplicația cu hot-reload, expune portul 8080, are legătură la baza de date
  - **db**: pornește un server MySQL accesibil doar celorlalte containere

- Folosește volume pentru a păstra pachetele .NET și datele MySQL și o rețea internă pentru legătura între servicii

- Dockerfile nu este detaliat aici, dar rolul lui este să stabilească mediul de build/rulare pentru aplicația .NET (ex. instalează SDK-ul .NET, copiază codul sursă)

Astfel, cu aceste fișiere, se poate porni rapid un mediu de lucru complet izolat pentru dezvoltare **.NET + MySQL** fără a instala nimic suplimentar pe calculatorul personal.

Utilizare cu un proiect .NET existent

Dacă există deja un proiect .NET creat, se poate utiliza direct cu acest setup Docker Compose urmând pașii:

1. Se cloneaza proiectul într-un folder
2. Se adaugă în .env numele proiectului in, variabila **DOCKER_PROJECT_NAME**

OBS: Valoarea implicită este **DockerProject**, deci dacă proiectul este deja într-un folder numit DockerProject, nu sunt necesare modificări.

Se pornesc serviciile cu:

3. `docker compose up -d`
4. Se accesează containerul app pentru a rula comenzi suplimentare (ex: migrații, restore, build), sau se utilizează workflow-ul sdk după caz (instalare pachete, configurare baza de date, etc.)
5. Proiectul o să pornească automat cu hot-reload pe portul definit (implicit 8080)
6. Apoi, se vor configura baza de date, connection string-ul și se vor rula migratiile, conform pașilor descriși anterior.