

Laborator 6

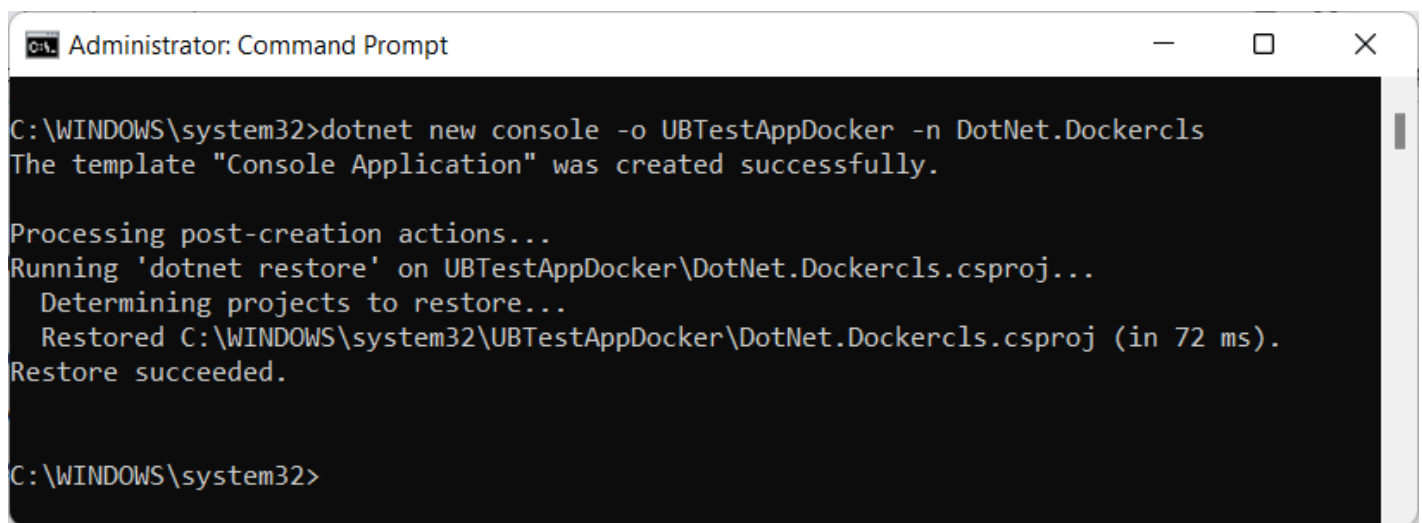
Aplicații .NET și Docker

1. Deschideți PowerShell sau Command Prompt/Terminal

Observație: Vom crea o aplicație simplă în .NET tip consolă pentru a ilustra principalele elemente fundamentale pentru crearea containerelor utilizând Docker.

2. În PowerShell/Terminal/Command Prompt introduceți comanda

```
dotnet new console -o UBTestAppDocker -n DotNet.Docker
```



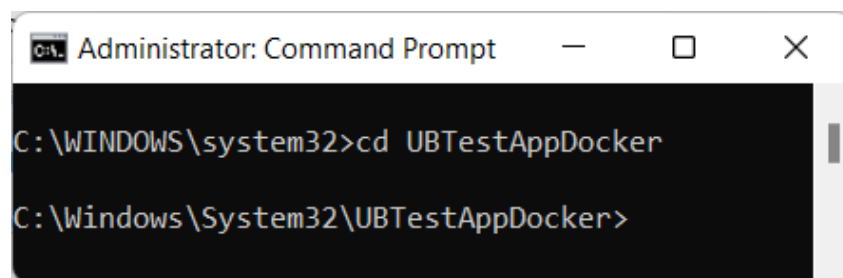
```
Administrator: Command Prompt

C:\WINDOWS\system32>dotnet new console -o UBTestAppDocker -n DotNet.Dockercls
The template "Console Application" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on UBTestAppDocker\DotNet.Dockercls.csproj...
  Determining projects to restore...
  Restored C:\WINDOWS\system32\UBTestAppDocker\DotNet.Dockercls.csproj (in 72 ms).
Restore succeeded.

C:\WINDOWS\system32>
```

3. Examinați aplicația creată și fișierele care o alcătuiesc. Locația aplicației poate să difere de la calculator la calculator.
4. Navigați la locația aplicației și intrați în folderul aplicației utilizând comanda `cd UBTestAppDocker`

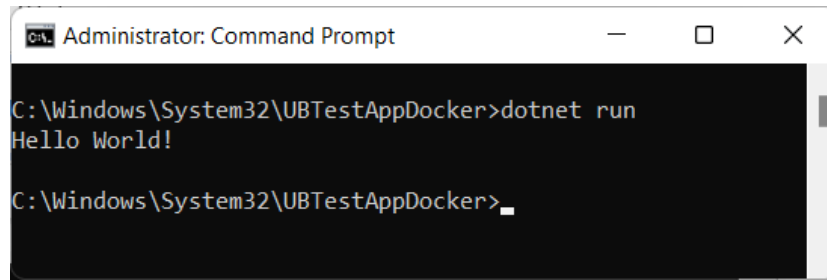


```
Administrator: Command Prompt

C:\WINDOWS\system32>cd UBTestAppDocker

C:\Windows\System32\UBTestAppDocker>
```

5. Rulați în terminal comanda `dotnet run` și verificați că aplicația rulează cu succes. După rularea cu succes va apărea în consolă mesajul `Hello World!`, un mesaj care este generat automat la crearea aplicației de către Visual Studio.

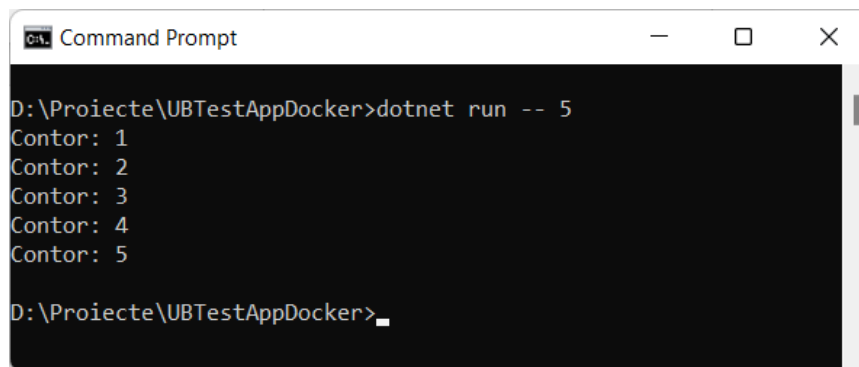


```
Administrator: Command Prompt
C:\Windows\System32\UBTestAppDocker>dotnet run
Hello World!
C:\Windows\System32\UBTestAppDocker>_
```

6. Deschideți folderul aplicației și identificați fișierul `Program.cs`.
7. Identificați linia `Console.WriteLine("Hello World!");`
8. Înlocuiți linia respectivă cu următorul fragment de cod:

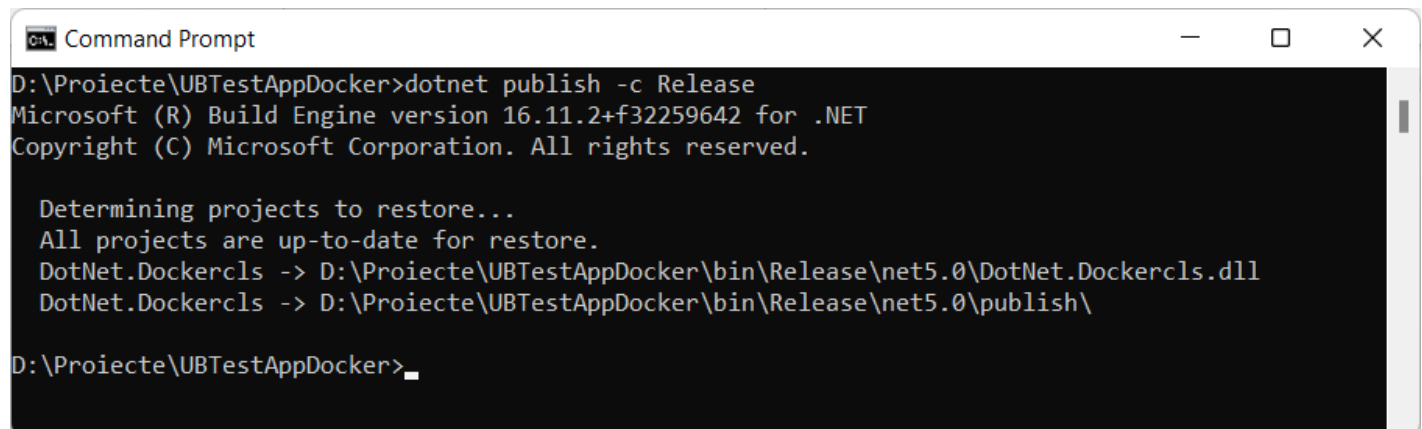
```
var contor = 0;
var valoare = args.Length != 0 ? Convert.ToInt32(args[0]) : -1;
while (valoare == -1 || contor < valoare)
{
    Console.WriteLine($"Contor: {++contor}");
}
```

9. În terminal rulați comanda `dotnet run -- 5` și comparați rezultatul obținut.



```
Command Prompt
D:\Proiecte\UBTestAppDocker>dotnet run -- 5
Contor: 1
Contor: 2
Contor: 3
Contor: 4
Contor: 5
D:\Proiecte\UBTestAppDocker>_
```

10. Înainte să adăugăm aplicația în container Docker, trebuie ca aplicația să fie publicată. Acest lucru se realizează rulând comanda: `dotnet publish -c Release`

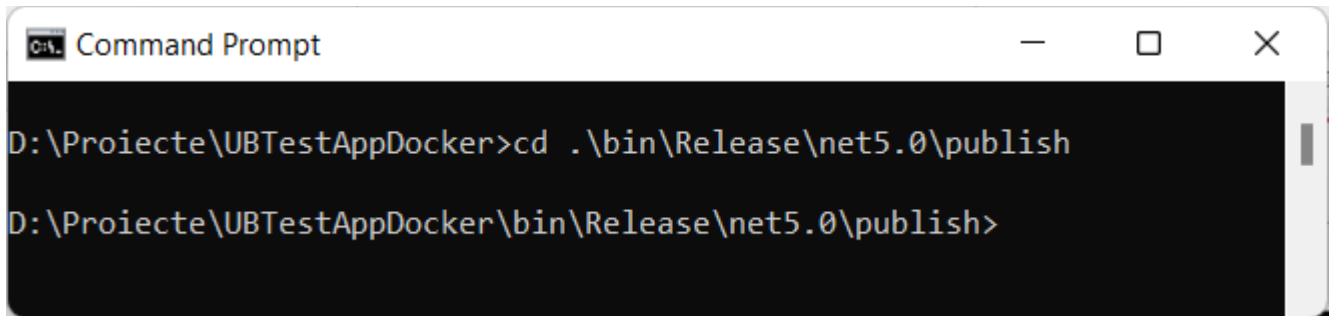


```
Command Prompt
D:\Proiecte\UBTestAppDocker>dotnet publish -c Release
Microsoft (R) Build Engine version 16.11.2+f32259642 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

Determining projects to restore...
All projects are up-to-date for restore.
DotNet.Dockercls -> D:\Proiecte\UBTestAppDocker\bin\Release\net5.0\DotNet.Dockercls.dll
DotNet.Dockercls -> D:\Proiecte\UBTestAppDocker\bin\Release\net5.0\publish\
D:\Proiecte\UBTestAppDocker>_
```

11. Navigați la locația `.\bin\Release\net5.0\publish` utilizând comanda

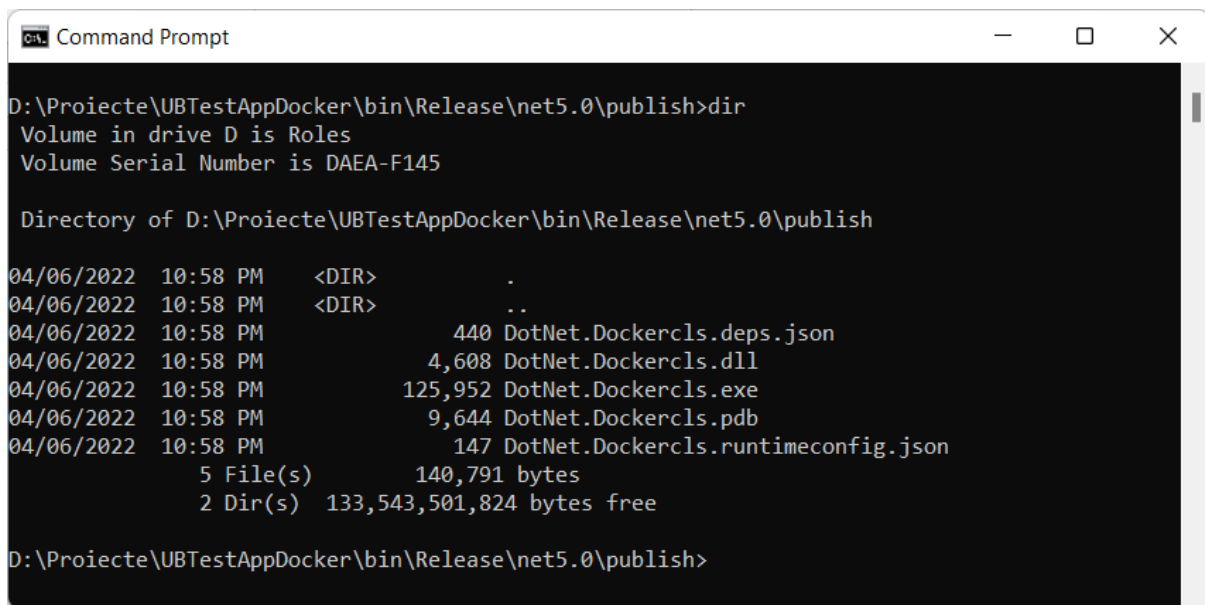
```
cd .\bin\Release\net5.0\publish
```



În cazul meu întreaga cale este

`D:\Proiecte\UBTestAppDocker\bin\Release\net5.0\publish`

12. Listați conținutul cu comanda `dir` locației și identificați că aveți același conținut ca cel din imagine.



13. Creați un fișier intitulat `Dockerfile` cu conținutul de mai jos și salvați fișierul în locația în care se află fișierul cu extensia `.csproj` (`DotNet.Dockercls.csproj`).

```
FROM mcr.microsoft.com/dotnet/sdk:5.0 AS build-env
```

```
WORKDIR /app
```

```
# Copiem fisierele respective
```

```
COPY . ./
```

```
# Restore as distinct layers
```

```
RUN dotnet restore
```

```
# Build and publish a release
```

```
RUN dotnet publish -c Release -o out
```

```
# Construim imaginea runtime
```

```
FROM mcr.microsoft.com/dotnet/aspnet:5.0
```

```
WORKDIR /app
```

```
COPY --from=build-env /app/out .
```

```
ENTRYPOINT ["dotnet", "DotNet.Docker.dll"]
```

Exemplul propus în cadrul acestui laborator utilizează imaginea de tip runtime specifică pentru ASP.NET Core (care conține imagine .NET de tip runtime) care corespunde cu aplicația consolă .NET. Imaginea ASP.NET Core este utilizată în mod intenționat în acest exemplu, astfel imaginea `mcr.microsoft.com/dotnet/runtime:6.0` ar putea fi folosită la fel.

- MCR – Microsoft Container Registry
- FROM – necesită un nume de imagine container Docker.
- MCR – `mcr.microsoft.com` este un nume pentru hub-ul docker – care găzduiește în mod public containerele accesibile.
- Porțiunea `dotnet` este containerul, unde segmentul `sdk` sau `aspnet` reprezintă numele imaginii container
- Imaginea are ca etichetă `6.0`, aspect foarte important și utilizat cu scopul versionării
- Prin urmare, `mcr.microsoft.com/dotnet/aspnet:6.0` reprezintă .NET 6.0 runtime.
- Verificați că aveți versiunea runtime care se potrivește cu runtime-ul targetat de către SDK. De exemplu, aplicația creată utilizează .NET 6.0 SDK și imaginea de bază menționată în *Dockerfile* este etichetată cu `6.0`

14. În terminal, rulați comanda `docker build -t counter-image -f Dockerfile`.
Verificați output-ul:

```

Command Prompt

D:\Proiecte\UBTestAppDocker>docker build -t counter-image -f Dockerfile .
[+] Building 35.9s (14/14) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 415B                                              0.0s
=> [internal] load .dockerignore                                                 0.0s
=> => transferring context: 2B                                                  0.0s
=> [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:6.0             1.1s
=> [internal] load metadata for mcr.microsoft.com/dotnet/sdk:6.0               1.1s
=> [internal] load build context                                                 0.2s
=> => transferring context: 884.75kB                                           0.1s
=> [stage-1 1/3] FROM mcr.microsoft.com/dotnet/aspnet:6.0@sha256:26ef9dc4aa354cc4aa4ae533c97f92d0d72c5e848f6968660be51d9fc1ce092e 13.2s
=> => resolve mcr.microsoft.com/dotnet/aspnet:6.0@sha256:26ef9dc4aa354cc4aa4ae533c97f92d0d72c5e848f6968660be51d9fc1ce092e 0.0s
=> => sha256:2ea2b9152ff024eb5d612c2a368894a35beb84336856e63d1fac442dac3bdfb 1.37kB / 1.37kB 0.0s
=> => sha256:683c561135969c88cbb30026752ac418886514d6b31fea0ecb202fb49df9e01b 3.37kB / 3.37kB 0.0s
=> => sha256:345ba88e9cec878c19fe799552330864563aefca68acdc4937543b4079bb1999 14.97MB / 14.97MB 2.7s
=> => sha256:b8df06e47c01c3c5c04367f8c512539c03fcf2c97a6d0c9ac56027b0109e9232 31.62MB / 31.62MB 5.8s
=> => sha256:26ef9dc4aa354cc4aa4ae533c97f92d0d72c5e848f6968660be51d9fc1ce092e 2.17kB / 2.17kB 0.0s
=> => sha256:c229119241af7b23b121052a1cae4c03e0a477a72ea6a7f463ad7623ff8f274b 31.38MB / 31.38MB 6.0s
=> => sha256:ea1c72bb96b40af8fa92638582bab7165cd2dc35444088feff5ee0fc6e0880fb 156B / 156B 2.9s
=> => sha256:aaa524629324e13aa3a6842e31eea95d892be78e8b480c1360ac464816fcbfff 9.45MB / 9.45MB 4.8s
=> => extracting sha256:c229119241af7b23b121052a1cae4c03e0a477a72ea6a7f463ad7623ff8f274b 2.8s
=> => extracting sha256:345ba88e9cec878c19fe799552330864563aefca68acdc4937543b4079bb1999 0.8s
=> => extracting sha256:b8df06e47c01c3c5c04367f8c512539c03fcf2c97a6d0c9ac56027b0109e9232 24.2s
=> => extracting sha256:ea1c72bb96b40af8fa92638582bab7165cd2dc35444088feff5ee0fc6e0880fb 0.0s
=> => extracting sha256:aaa524629324e13aa3a6842e31eea95d892be78e8b480c1360ac464816fcbfff 0.5s
=> [build-env 1/5] FROM mcr.microsoft.com/dotnet/sdk:6.0@sha256:a2a8f968b043349b8faa0625c5405ac33da70b3274ff9e17109430f16aa9a3ee 22.6s
=> => resolve mcr.microsoft.com/dotnet/sdk:6.0@sha256:a2a8f968b043349b8faa0625c5405ac33da70b3274ff9e17109430f16aa9a3ee 0.0s
=> => sha256:345ba88e9cec878c19fe799552330864563aefca68acdc4937543b4079bb1999 14.97MB / 14.97MB 2.7s
=> => sha256:a2a8f968b043349b8faa0625c5405ac33da70b3274ff9e17109430f16aa9a3ee 2.17kB / 2.17kB 0.0s
=> => sha256:845edd05175a1625df3f854f6f92e8d70ee8c95bc02c99e53b99eae139234d3 2.01kB / 2.01kB 0.0s
=> => sha256:2d1a2c7481f5a7fd2894c2bf87f5486812e75136388dac6ce32c04803f0c4de 7.30kB / 7.30kB 0.0s
=> => sha256:c229119241af7b23b121052a1cae4c03e0a477a72ea6a7f463ad7623ff8f274b 31.38MB / 31.38MB 6.0s
=> => sha256:b8df06e47c01c3c5c04367f8c512539c03fcf2c97a6d0c9ac56027b0109e9232 31.62MB / 31.62MB 5.8s
=> => sha256:ea1c72bb96b40af8fa92638582bab7165cd2dc35444088feff5ee0fc6e0880fb 156B / 156B 2.8s
=> => sha256:aaa524629324e13aa3a6842e31eea95d892be78e8b480c1360ac464816fcbfff 9.45MB / 9.45MB 4.8s
=> => sha256:bc180deca390379b4cebd74906b4cf3e1be6cc361771b6ab2f9fd3aa99ba555 25.37MB / 25.37MB 7.6s
=> => sha256:0a37482a5789270445554321ceaa2aef3a21f4d19ebe3dfbc7d3299728ad85d 139.14MB / 139.14MB 15.9s
=> => sha256:384ec9ec6797a86e3285efce7ddad35ba71c5b4d13c67230d8df547a8f2d721d 13.37MB / 13.37MB 8.1s
=> => extracting sha256:c229119241af7b23b121052a1cae4c03e0a477a72ea6a7f463ad7623ff8f274b 28.2s
=> => extracting sha256:345ba88e9cec878c19fe799552330864563aefca68acdc4937543b4079bb1999 25.1s
=> => extracting sha256:b8df06e47c01c3c5c04367f8c512539c03fcf2c97a6d0c9ac56027b0109e9232 1.5s
=> => extracting sha256:ea1c72bb96b40af8fa92638582bab7165cd2dc35444088feff5ee0fc6e0880fb 0.0s
=> => extracting sha256:aaa524629324e13aa3a6842e31eea95d892be78e8b480c1360ac464816fcbfff 0.5s
=> => extracting sha256:bc180deca390379b4cebd74906b4cf3e1be6cc361771b6ab2f9fd3aa99ba555 2.9s
=> => extracting sha256:0a37482a5789270445554321ceaa2aef3a21f4d19ebe3dfbc7d3299728ad85d 5.5s

Command Prompt

=> => extracting sha256:384ec9ec6797a86e3285efce7ddad35ba71c5b4d13c67230d8df547a8f2d721d 0.6s
=> [stage-1 2/3] WORKDIR /app                                                    1.8s
=> [build-env 2/5] WORKDIR /app                                                  2.3s
=> [build-env 3/5] COPY . /                                                      0.3s
=> [build-env 4/5] RUN dotnet restore                                           5.6s
=> [build-env 5/5] RUN dotnet publish -c Release -o out                       3.4s
=> [stage-1 3/3] COPY --from=build-env /app/out .                               0.1s
=> exporting to image                                                            0.2s
=> => exporting layers                                                            0.1s
=> => writing image sha256:ff662f3df6a624ca659e61de12819fc5e2380c156f92d08deb642a33eafcd0df 0.0s
=> => naming to docker.io/library/counter-image                                0.0s

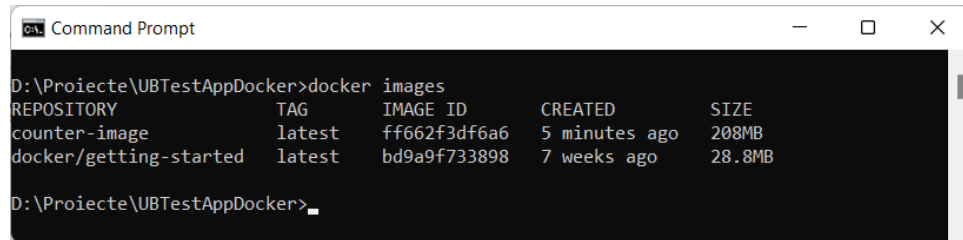
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

D:\Proiecte\UBTestAppDocker>

```

Docker va procesa fiecare linie din fișierul *Dockerfile*. Punctul (.) din comandă setează contextul de build al imaginii. Flag-ul -f reprezintă calea către *Dockerfile*. Comanda construiește imaginea și crează un repository local numit *counter-image* care trimite către imaginea respective.

15. Listați imaginile instalate cu ajutorul comenzii `docker images`



```
Command Prompt
D:\Proiecte\UBTestAppDocker>docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
counter-image        latest          ff662f3df6a6   5 minutes ago   208MB
docker/getting-started latest          bd9a9f733898   7 weeks ago     28.8MB
D:\Proiecte\UBTestAppDocker>
```

Repository-ul *counter-image* reprezintă numele imaginii. Eticheta *latest* este eticheta utilizată pentru identificarea imaginii. ID-ul imaginii este `ff662f3df6a6`. Timestamp-ul '5 minutes ago' este timpul la care a fost creată imaginea. Dimensiunea este redată de coloana *SIZE* și este 208MB.

Pasul final din fișierul *Dockerfile* este să creeze un container din imagine și să ruleze aplicația, să copieze aplicația publicată în container, și să definească punctul de intrare (entry point).

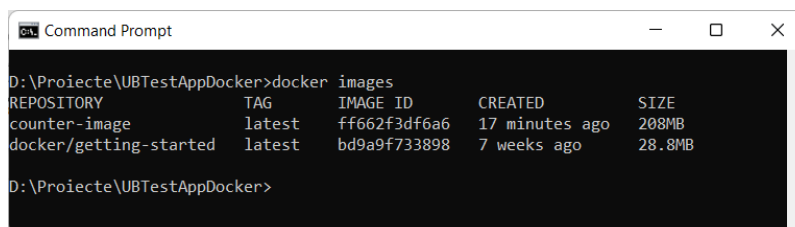
```
FROM mcr.microsoft.com/dotnet/aspnet:5.0
WORKDIR /app
COPY --from=build-env /app/out .
ENTRYPOINT ["dotnet", "DotNet.Docker.dll"]
```

Comanda COPY îi spune lui Docker să copieze folderul specificat de pe calculator către un folder din container. În exemplul nostru, folderul *publish* este copiat întrun folder numit *app* din container.

Comanda WORKDIR schimbă directorul current în *app* în interiorul containerului.

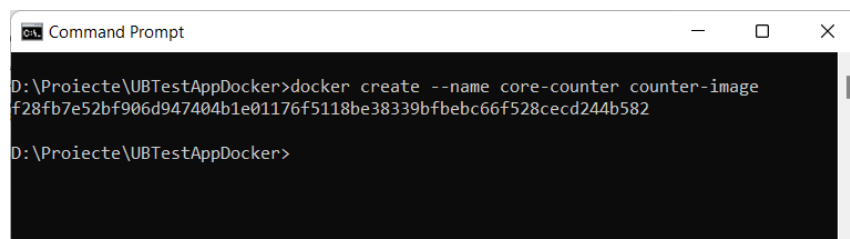
Comanda ENTRYPOINT îi spune Dockerului să configureze containerul să ruleze ca un executabil. Când containerul pornește, comanda ENTRYPOINT pornește. Când comanda ajunge la final, containerul se oprește din rulare.

16. În terminal rulați comanda `docker images`



```
Command Prompt
D:\Proiecte\UBTestAppDocker>docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
counter-image        latest          ff662f3df6a6   17 minutes ago   208MB
docker/getting-started latest          bd9a9f733898   7 weeks ago     28.8MB
D:\Proiecte\UBTestAppDocker>
```

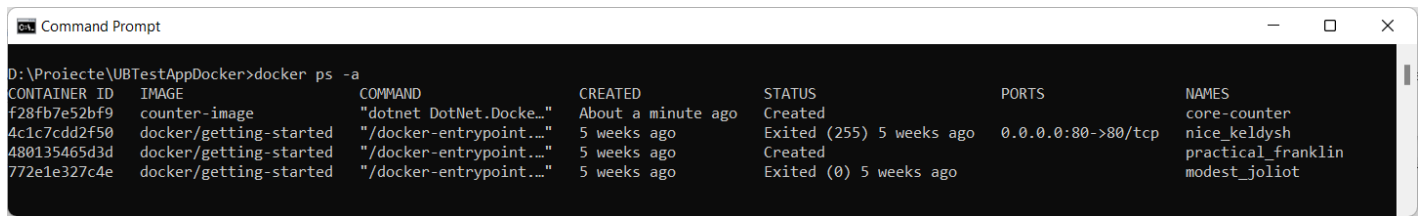
17. Avem imaginea! Următorul pas este generăm un container pentru aplicația respectivă. Rulați în terminal comanda `docker create --name core-counter counter-image`



```
Command Prompt
D:\Proiecte\UBTestAppDocker>docker create --name core-counter counter-image
f28fb7e52bf906d947404b1e01176f5118be38339bfbebc66f528cecd244b582
D:\Proiecte\UBTestAppDocker>
```

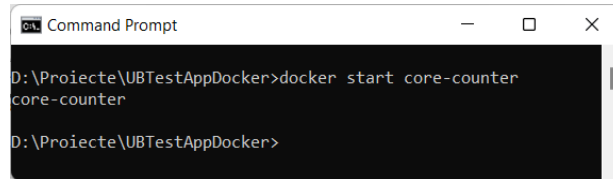
Comanda `docker create` va genera containerul bazat pe imaginea *counter-image*. Outputul reprezintă ID-ul containerului.

18. Listați toate containerele utilizând comanda `docker ps -a`



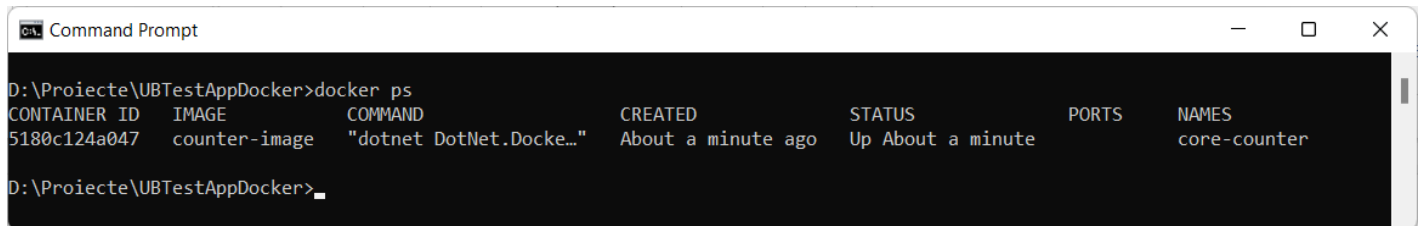
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f28fb7e52bf9	counter-image	"dotnet DotNet.Docke..."	About a minute ago	Created		core-counter
4c1c7cdd2f50	docker/getting-started	"/docker-entrypoint..."	5 weeks ago	Exited (255) 5 weeks ago	0.0.0.0:80->80/tcp	nice_keldysh
480135465d3d	docker/getting-started	"/docker-entrypoint..."	5 weeks ago	Created		practical_franklin
772e1e327c4e	docker/getting-started	"/docker-entrypoint..."	5 weeks ago	Exited (0) 5 weeks ago		modest_joliot

19. Pentru pornirea/rularea containerului utilizați comanda `docker start core-counter`



```
D:\Proiecte\UBTestAppDocker>docker start core-counter
core-counter
D:\Proiecte\UBTestAppDocker>
```

20. Verificați imaginea ca este pornită



CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5180c124a047	counter-image	"dotnet DotNet.Docke..."	About a minute ago	Up About a minute		core-counter

21. Pentru oprirea containerului utilizați comanda `docker stop core-counter`.

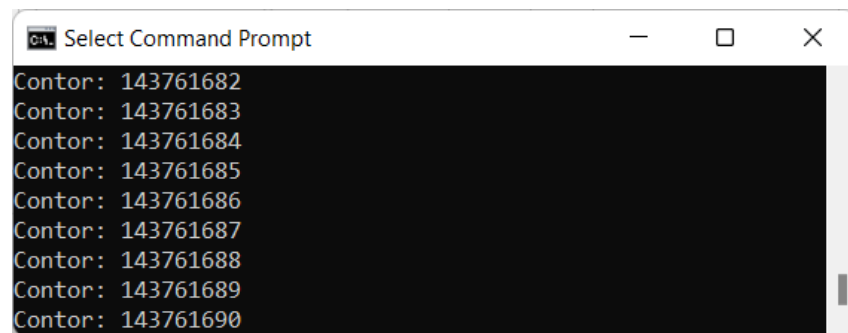
22. Pentru ștergerea unui container se utilizează comanda `docker rm core-counter`

23. Pentru a ne conecta la containerul `core-counter` se utilizează comenzile:

```
docker start core-counter
```

```
docker attach --sig-proxy=false core-counter
```

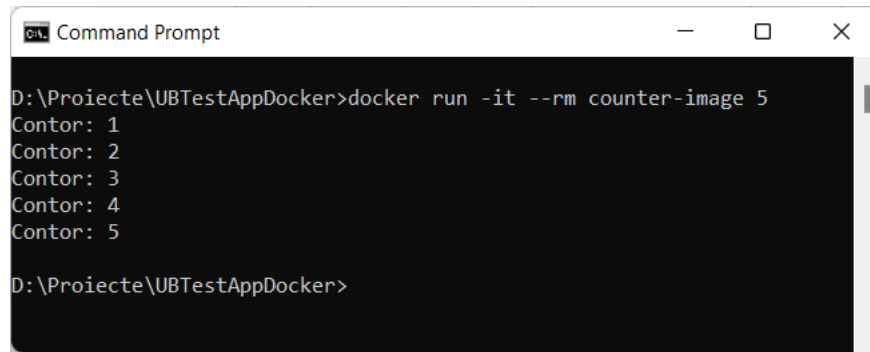
În acest caz se va rula la infinit bucla `while` din cod.



```
Select Command Prompt
Contor: 143761682
Contor: 143761683
Contor: 143761684
Contor: 143761685
Contor: 143761686
Contor: 143761687
Contor: 143761688
Contor: 143761689
Contor: 143761690
```

24. Pentru a putea rula containerul respective împreună cu imaginea atașată pentru argumente, vom utiliza comanda:

```
docker run -it --rm counter-image 5
```

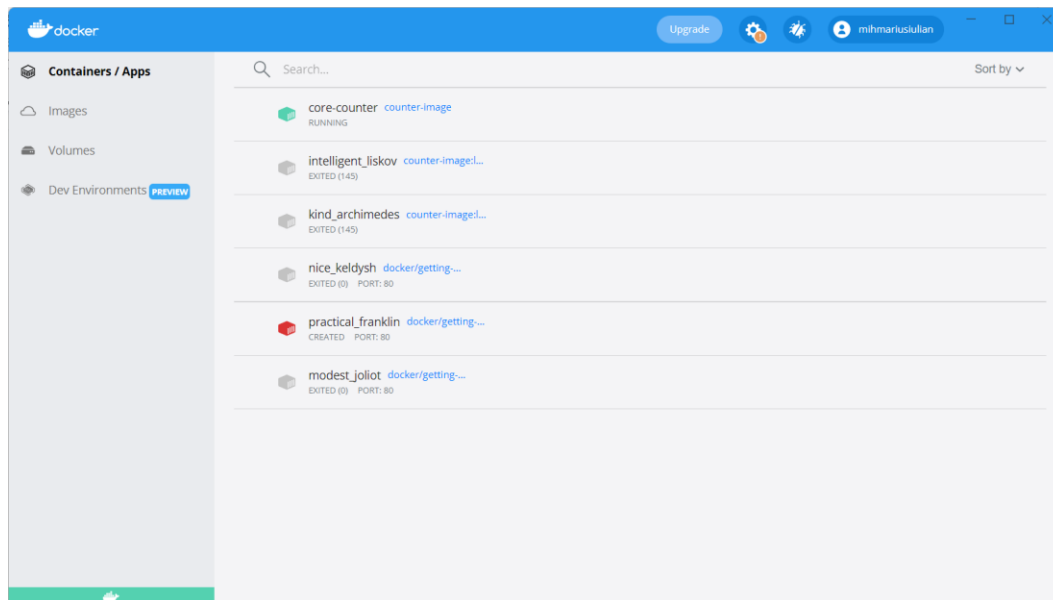


```
Command Prompt

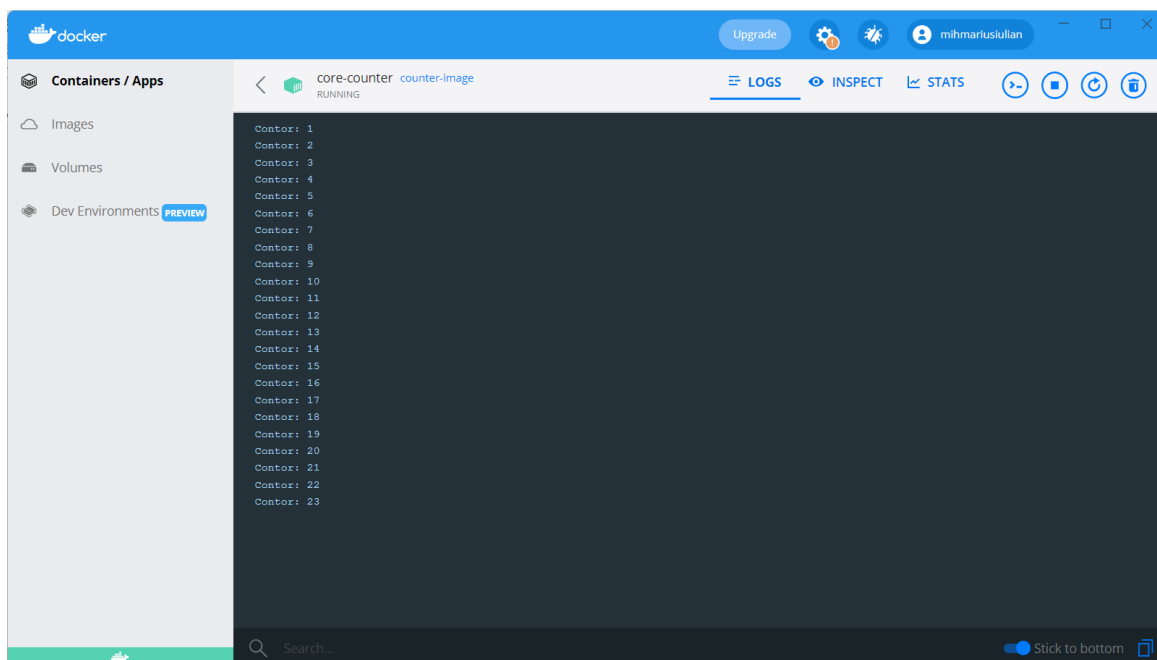
D:\Proiecte\UBTestAppDocker>docker run -it --rm counter-image 5
Contor: 1
Contor: 2
Contor: 3
Contor: 4
Contor: 5

D:\Proiecte\UBTestAppDocker>
```

25. Deschideți DockerHub și verificați că aveți imaginea core-counter create.



26. Deschideți imaginea respective și observați procesul de generare



Aplicații de laborator

Pentru fiecare dintre aplicațiile¹ de mai jos, parcurgeți pașii de mai sus și adăugați aplicațiile respective în Docker. Aplicațiile sunt elementare și se pot dezvolta în aplicații tip console.

1. Implementați o aplicație care să calculeze factorial dintrun număr.
2. Implementați o aplicație care să calculeze numărul de vocale dintrun cuvânt/propoziție.
3. Implementați o aplicație care să afișeze frecvența de apariție pentru fiecare caracter dintr-o propoziție sau paragraf.
Exemplu: Fie propoziția “Creez aplicații cu Docker”
C – 1
r – 2
e – 3
z – 1
a – 2
etc...
4. Implementați o aplicație tip joc Spânzurătoarea.
5. Implementați o aplicație care să permită inventarul unor cărți/produse (opțional se poate realiza cu bază de date sau doar un fișier text/XML/json etc.)

¹ Puteți să vă alegeți doar o singură aplicație sau pe toate și să le dezvoltați pe parcursul semestrului.