



# Back End Development 1

## sesi 13

---

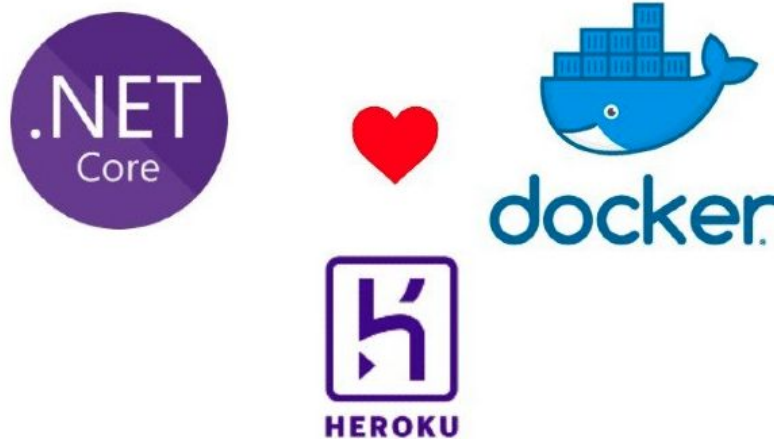


# Deployment ≠ Git Explorations

## DEPLOYMENT + GIT INTEGRATIONS - Sesi 13

### Introduction

Sebagai seorang Developer, kita tentu senang melakukan beberapa proyek sampingan dan juga ingin menunjukkannya kepada teman developer lainnya. Nah dalam kebanyakan kasus, kita hanya bisa menjalankannya secara lokal dan menunjukkannya kepada developer lain, tetapi bagaimana jika kita ingin menyebarkannya ke cloud, maka kita memerlukan beberapa layanan cloud berbayar seperti Azure atau AWS.



Tapi masalahnya, sebagian besar waktu proyek sampingan kami hanyalah POC dan kami tidak ingin mengeluarkan uang sepeserpun untuk itu, jadi dalam skenario ini, Heroku menjadi opsi alternatif yang sangat baik untuk di coba.

## DEPLOYMENT + GIT INTEGRATIONS - Sesi 13

# HEROKU

Heroku is a platform as a service (PaaS) that enables developers to build, run, and operate applications entirely in the cloud.

Source : <https://www.heroku.com/>

Currently, Heroku supports languages Ruby, Java, PHP, Python, Node, Go, Scala and Clojure but unfortunately not .Net or .Net Core. This where Docker comes into the picture. Heroku provides ways to deploy apps with Docker.

Containers are a standardized unit of software that allows developers to isolate their app from its environment, solving the “it works on my machine” headache. For millions of developers today, Docker is the de facto standard to build and share containerized apps — from desktop, to the cloud.

## DEPLOYMENT + GIT INTEGRATIONS - Sesi 13

### Persiapan

Pada dasarnya untuk deployment via heroku, ASP NET CORE 5 membutuhkan docker jadi sebagai persiapan kita bisa lakukan instalasi yang diperlukan terlebih dahulu.

1. .Net Core App
2. Docker
3. Visual Studio or VS Code
4. Heroku free account
5. Heroku CLI

## 1. Containing the ASP.NET Core Application

Hal pertama yang harus kita lakukan dalam aplikasi adalah menambahkannya ke Docker sebagai Container, untuk itu, file Dockerfile harus ditambahkan ke proyek.

Di file inilah kita akan menentukan bagaimana container kita akan dibuat.

Juga disarankan agar file .dockerignore dibuat dalam aplikasi . Seperti .gitignore , file ini mendefinisikan direktori dan file yang harus diabaikan.

Untuk aplikasi ini, script .dockerignore adalah:

```
.dockerignore x
carterapi-master > .dockerignore
1  **/.dockerignore
2  **/.git
3  **/.gitignore
4  **/.vs
5  **/.vscode
6  **/*.proj.user
7  **/bin
8  **/obj
9
```

Masih dalam project yang sama buat file bernama Dockerfile

```
1 FROM mcr.microsoft.com/dotnet/core/aspnet:3.1 AS base
2 WORKDIR /app
3
4 FROM mcr.microsoft.com/dotnet/core/sdk:3.1 AS build
5 WORKDIR /src
6 COPY . .
7 RUN dotnet restore
8 RUN dotnet build --no-restore -c Release -o /app
9
10 FROM build AS publish
11 RUN dotnet publish --no-restore -c Release -o /app
12
13 FROM base AS final
14 WORKDIR /app
15 COPY --from=publish /app .
16
17 CMD ASPNETCORE_URLS=http://*:$PORT dotnet CarterAPI.dll
18
```

File ini berbentuk script yang akan diimplementasi sewaktu kita build dan running docker.

Jalankan docker Build -t carter-api .

```
bash-3.2$ docker build -t carter-api .
Sending build context to Docker daemon 27.65kB
Step 1/13 : FROM mcr.microsoft.com/dotnet/core/aspnet:3.1 AS base
3.1: Pulling from dotnet/core/aspnet
68ced04f60ab: Pull complete
e936bd534ffb: Pull complete
caf64655bcbb: Pull complete
7064c3d93b4a: Pull complete
4156e490f05f: Pull complete
Digest: sha256:bb49293c317ada0f9e2c33fd96740167684528e08d41334cd0ffcddbb4bedc8d
Status: Downloaded newer image for mcr.microsoft.com/dotnet/core/aspnet:3.1
----> e2cd20adb129
Step 2/13 : WORKDIR /app
----> Running in 6fe0ee2304dd
Removing intermediate container 6fe0ee2304dd
----> db26de1d2662
Step 3/13 : FROM mcr.microsoft.com/dotnet/core/sdk:3.1 AS build
3.1: Pulling from dotnet/core/sdk
50e431f79093: Pull complete
dd8c6d374ea5: Pull complete
c85513200d84: Pull complete
55769680e827: Pull complete
053b9b7da68c: Pull complete
7e81b64e966c: Extracting [=====>] 17.27MB/115.9MB
c70b1f2ee510: Download complete
```





## DEPLOYMENT + GIT INTEGRATIONS - Sesi 13


### Login Heroku

Sambil menunggu image project docker selesai, kita persiapkan new app di heroku untuk publish app yang lagi di build ke Heroku.

Create new App pada dashboard Heroku



Create New App


App name



carter-api is available

Choose a region

 United States 

 Add to pipeline...

Create app

Kembali ke app, dan masih dalam project folder di CLI , jalankan command ini :

heroku container:push web -a carter-api

```
bash-3.2$ heroku container:push web -a carter-api
=== Building web (/Users/wlad/Documents/Treinaweb/Projetos/Blog/CarterAPI/Dockerfile)
|
----> fa40ab282a5b
Step 5/13 : COPY . .
----> Using cache
----> fb71d0cb2628
Step 6/13 : RUN dotnet restore
----> Using cache
----> 647905bde87f
Step 7/13 : RUN dotnet build --no-restore -c Release -o /app
----> Using cache
----> 526b431bc380
Step 8/13 : FROM build AS publish
----> 526b431bc380
Step 9/13 : RUN dotnet publish --no-restore -c Release -o /app
----> Using cache
----> c51946d5d521
Step 10/13 : FROM base AS final
----> db26de1d2662
Step 11/13 : WORKDIR /app
----> Using cache
----> d9bb6696ed52
Step 12/13 : COPY --from=publish /app .
----> Using cache
----> bc1000f55665
Step 13/13 : CMD ASPNETCORE_URLS=http://*:$PORT dotnet CarterAPI.dll
----> Using cache
----> 99f72dcf357b
Successfully built 99f72dcf357b
Successfully tagged registry.heroku.com/carter-api/web:latest
=== Pushing web (/Users/wlad/Documents/Treinaweb/Projetos/Blog/CarterAPI/Dockerfile)
|
```



Sekarang Jalankan `container:release`

`heroku container:release web -a carter-api`

```
bash-3.2$ heroku container:release web -a carter-api
Releasing images web to carter-api... done
bash-3.2$ █
```

One berhasil silahkan akses :  
`Carter-api.herokuapp.com`

Versi swagger  
`carter-api.herokuapp.com/openapi/ui/index.html`