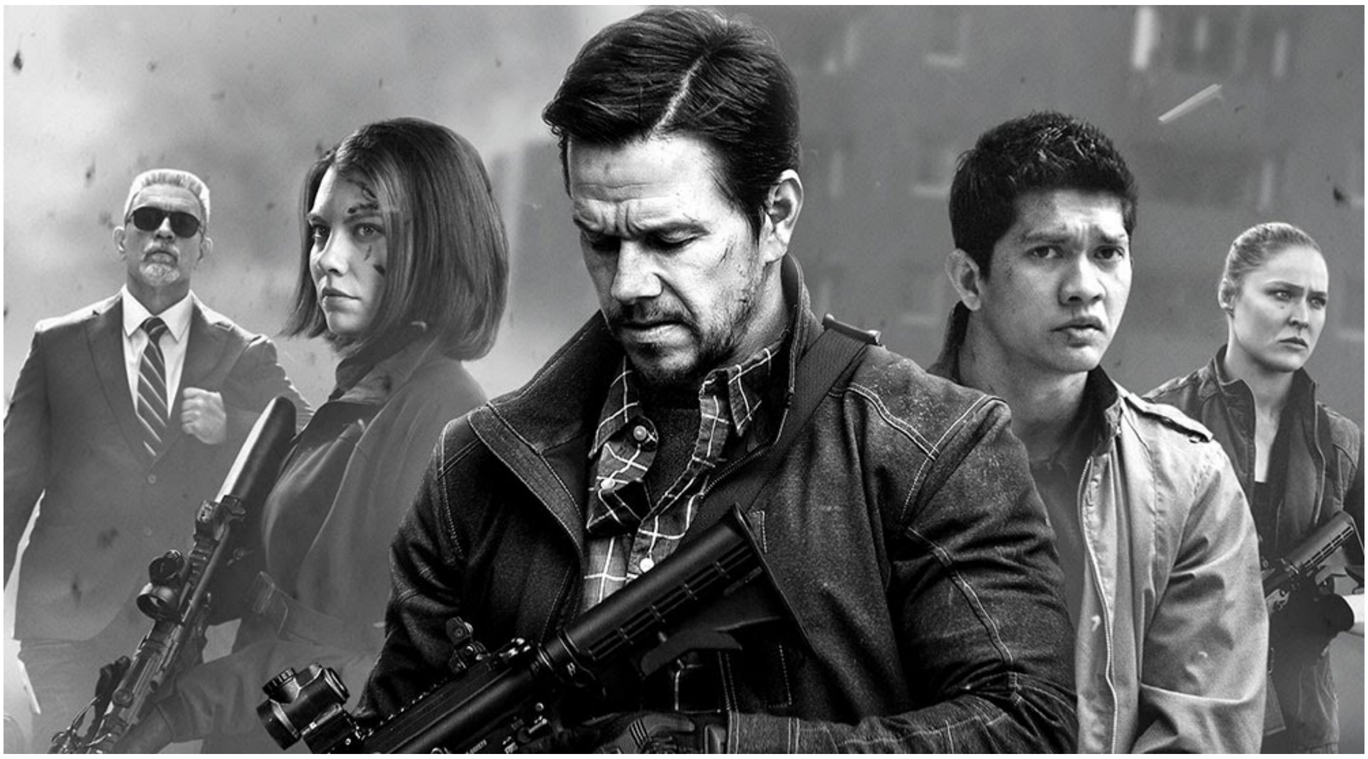
Dockerize WebForm Application
—「最後一公里」之「行百里者半於九十」



0. Prologue

It was a windy Friday night, I laid as if i had been stabbed on my back. My brain awake but i was unable to move. Numb as i was and spirits floating around. Suddenly, a strange idea dawned upon me to *dockerize* ta so as to improve my IT skills.

I thought again and again what I was doing and what was still missing. Procedures and steps to fulfill my plan was vaguely formed. It was an experiment to run real world **WebForm** Application on Docker containers, on Windows platform of course.

The only software I need was Docker Desktop for Windows. The price is free, the value is priceless -- in order to run Windows containers, Hyper-V is a must! That means after installation, virtualization software such as VMWare and VirtualBox will cease to function.

To be honest, Windows container is a barren field (*virgin* was used in draft) that few people dare to challenge with...

`corrigendum`

According to Oracle© VM VirtualBoxAdministrator's Guide for Release 6.0:

> 2.33. Using Hyper-V with Oracle VM VirtualBox
> Oracle VM VirtualBox can be used on a Windows host where Hyper-V is running. This is an experimental feature.

> No configuration is required. Oracle VM VirtualBox detects Hyper-V automatically and uses Hyper-V as the virtualization engine for the host system. The CPU icon in the VM window status bar indicates that Hyper-V is being used.
>
> **Note**
> When using this feature, some host systems might experience significant Oracle VM VirtualBox performance degradation.

---

1. The App

This is the easy part. Since TA is written in such a way that no external dependency exists. All packages are installed via NuGet package manager at application level. It connects to an oracle database as defined in web.config:

```
<add name="conn" connectionString="DATA SOURCE=oracle12-scan/mypdb;USER
ID=myuserid;PASSWORD=mypwd;PERSIST SECURITY INFO=True;Connection
Timeout=120;Max Pool Size=500;"
providerName="Oracle.ManagedDataAccess.Client" />
```

TA uses **.NET v4.5 Classic** application pool runs under **Default Web Site**.

2. The Image

Officially, there are two images capable of running WebForm programs:

```
mcr.microsoft.com/dotnet/framework/aspnet:4.8
```

for .NET framework version 4.8, which is the latest.

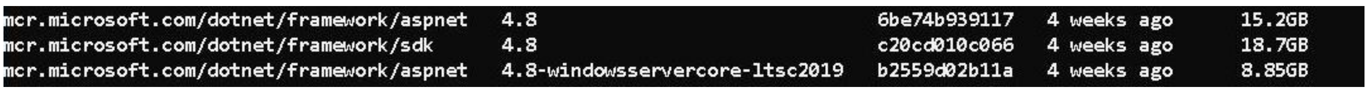
```
mcr.microsoft.com/dotnet/framework/aspnet:3.5
```

for .NET framework version 3.5, which is for older programs.

Around `15G` each! Just spend some time to pull your image in lunch time...

`corrigendum`

Using image **mcr.microsoft.com/dotnet/framework/aspnet:4.8-windowsservercore-ltsc2019** significantly reduces image size.



```
mcr.microsoft.com/dotnet/framework/aspnet   4.8                             6be74b939117   4 weeks ago   15.2GB
mcr.microsoft.com/dotnet/framework/sdk      4.8                             c20cd010c066   4 weeks ago   18.7GB
mcr.microsoft.com/dotnet/framework/aspnet   4.8-windowsservercore-ltsc2019  b2559d02b11a   4 weeks ago   8.85GB
```

3. The Scripts

The hardcore part of dockerizing is to create a Dockerfile.

```
FROM mcr.microsoft.com/dotnet/framework/aspnet:4.8

SHELL ["powershell"]

RUN New-Item -Path 'C:\inetpub\wwwroot\TA' -Type Directory; \
    New-Item -Path 'C:\writable' -Type Directory; \
    Remove-Website -Name 'Default Web Site'; \
    New-Website -Name 'DefaultWebSite' -PhysicalPath 'C:\inetpub\wwwroot' -
Port 80 -Force; \
    New-WebApplication -Name 'TA' -Site 'DefaultWebSite' -PhysicalPath
'C:\inetpub\wwwroot\TA' -ApplicationPool '.NET v4.5 Classic';

EXPOSE 80

RUN Set-ItemProperty -Path
'HKLM:\SYSTEM\CurrentControlSet\Services\Dnscache\Parameters' \
    -Name ServerPriorityTimeLimit -Value 0 -Type DWord

COPY TA /inetpub/wwwroot/TA
```

Everything just like ordinary Dockerfile except the **RUN** command. It create two folders, remove and re-create Default Web Site, create WebApplication just like deploying a website manually.

SHELL ["powershell"] changes to a different shell for the rest of the Dockerfile, so I can run PowerShell cmdlets.

RUN Set-ItemProperty turns off the Windows DNS cache inside the image, so any DNS requests get served by Docker.

Makefile

```
. . .
build:
   docker build --tag albert0i/ta:2.8.1 --no-cache .

up:
   docker run --name ta --rm -d -p 80:80 -v
C:\Docker\TA\writable:c:\writable albert0i/ta:2.8.1

down:
   docker container stop ta
. . .
```

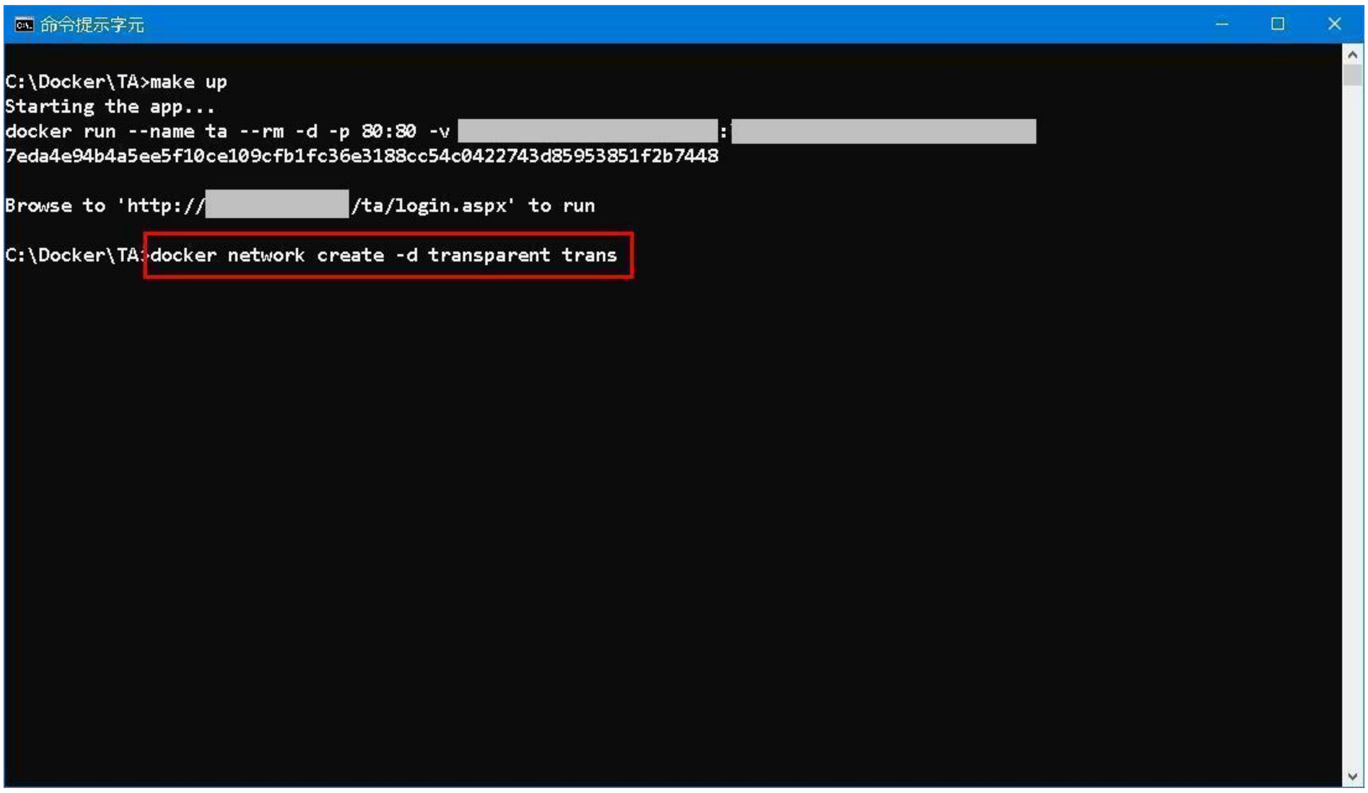
A Makefile is used to facilitate the whole build-and-run lifecycle.

4. Summary

In case your windows container can't reach the database, try to run:

```
docker network create -d transparent trans
```

I don't know why but it works for me...



```
C:\Docker\TA>make up
Starting the app...
docker run --name ta --rm -d -p 80:80 -v [redacted]:[redacted]
7eda4e94b4a5ee5f10ce109cfb1fc36e3188cc54c0422743d85953851f2b7448

Browse to 'http://[redacted]/ta/login.aspx' to run

C:\Docker\TA>docker network create -d transparent trans
```

Goodbye and Good Luck!

5. Reference

1. Docker Desktop for Windows
2. ASP.NET - By Microsoft - Official images for ASP.NET
3. Modernizing Traditional .NET Apps with Docker
4. New-WebApplication
5. Windows docker container cannot ping host

EOF (2021/10/19 Created)

EOF (2022/06/26 Revised)