Docker

Kommandon för att hantera Docker Containrar och avbilder

Vanliga kommandon för att lista och ta bort containrar och avbilder

- docker ps
  - Listar alla containrar som körs
- docker ps **-a**
  - Listar all containrar oavsett om de körs eller inte
- docker **rm** *<container namn eller container id>*
  - Tar bort en container(Viktigt att containern är stoppad)

- docker images
  - Listar alla avbilder som finns på den lokala datorn
- docker **rmi** *<image id>*
  - Tar bort en image (För att kunna ta bort en avbild så måste containern som är skapad utifrån avbilden först tas bort)

Kommandon för att skapa en container

docker **run** -d --name *<namn man vill ge sin container>* -p 8080:80 *<namn eller id på avbilden som skall användas av containern>*

- -d
  - betyder att containern kommer att köras i bakgrunden
- --name
  - här kan man ge containern ett namn, annars skapar docker ett namn åt containern
- -p
  - -p står för *publish*, vilket enkelt uttryckt betyder vilken extern port skall knytas till den interna porten
- --rm
  - Kommer att ta bort containern efter det att den stoppats

För en fullständig lista av inställningar/flaggor som går att sätta för run kommandot se följande länk: https://docs.docker.com/engine/reference/commandline/run/

Kommandon för att skapa en avbild

För att skapa en avbild av kod så måste vi ha skapa fil som döps till ***Dockerfile*** utan filändelse. I denna fil skapar vi instruktioner för hur avbilden skall skapas utifrån koden i vår applikation.

Exempel:

Här skapas en container ifrån en .NET 6.0 Razor Pages applikation

```
# Steg 1. Hämta bas avbilden(image) för aspnet 6.0...
# Vi är beroende av sdk för att kunna bygga vår applikation i avbilden
FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build-env

# Steg 2. Ange vad katalogen skall heta i avbilden dit vi skall kopiera och köra vår kod ifrån
WORKDIR /app

# Steg 3. Kopiera all källkod till docker...
# COPY <src>mellanslag<dest>
COPY . ./
# Kör dotnet restore för att återställa paket som är definierade i *.csproj filen
RUN dotnet restore

# Steg 4. Skapa ett publiceringspaket
# publish skapar ett optimerat publiceringspaket
# -c anger vilken konfiguration vi ska använda
# Release anger att det är ett bygg paket som är optimerat
# -o anger vart vill vi placera detta paket -> out katalogen
RUN dotnet publish -c Release -o out

# Steg 5. Skapa en byggprocess av applikationen...
# Vi kommer att skapa en aspnet applikation
# Vår applikation är beroende av aspnet version 6
FROM mcr.microsoft.com/dotnet/aspnet:6.0
# Ange vart i avbilden som filer ska placeras
WORKDIR /app

# COPY använder sig av build-env inställningarna ovan
# och kopierar filer och paket ifrån katalogen
# /app/out. Observera mellanslaget innan sista punkten.
COPY --from=build-env /app/out .

# Steg 6. Ange hur applikation ska startas upp...
# I detta fall använder vi kommandot dotnet och startar applikationens dll fil.
ENTRYPOINT [ "dotnet", "westcoastcars-app.dll" ]
```

Bygga/skapa avbilden

För att skapa avbilden ifrån ovanstående Dockerfile, kan vi köra följande kommando
docker build **-t** *<image-namn>* **-f** *Dockerfile* .

- -t sker oss möjlighet att ge vår avbild ett namn/tagg
- -f anger vart instruktionerna för att bygga vår avbild finns (*Dockerfile*). Om Dockerfile finns i samma katalog som vi kör kommandot ifrån, då kan vi utelämna detta kommando.
- . Sista punkten anger var koden finns för att skapa avbilden finns. Punkten betyder i aktuell katalog.

Starta och stoppa en container

För att starta en container som är skapad kör kommandot
docker start *<container namn/container id>*

För att stoppa en container som körs använd kommandot
docker stop *<container namn/container id>*