INTRODUCTION TO DOCKER FOR .NET DEVELOPERS

BRENDON MATHESON

Your Presenter

- Brendon Matheson
- Australian
- 11yr Bangkok Resident

Currently Working On

- Healthcare (Architect at Orion Health)
- Cloud / Multi-Tenant / SaaS
- Functions-as-a-Service (FaaS)


DOCKER

Agenda

- What is Docker?
- Exercise 1 – hello-world
- Exercise 2 – Externalities
- Exercise 3 – Build a .NET Core app on Linux
- Exercise 4 – Dockerize a .NET Core app on Linux
- Exercise 5 – Customize the dotnetcore SDK container
- Exercise 6 – Dockerize nginx and CIFS on Linux
- Exercise 7 – Serve static content in IIS on Windows
- Exercise 8 – Serve web app in IIS on Windows

Virtualization vs Containerization

Docker is a cool n virtualization techno y

What is Docker?

Virtualization

- Virtual hardware
  - CPU
  - Disk
  - Memory
  - Devices
- Guest OS and software installed into VM

VM's => System-Oriented

Containerization

- Native hardware – no hypervisor
  - Allocate resources with control groups (on Linux)
- Host kernel is used by containerized process

Containers => Service-Oriented

References

Background Reading

- https://docs.docker.com/engine/docker-overview/#what-can-i-use-docker-for
- http://www.haifux.org/lectures/299/netLec7.pdf

Installation

- https://docs.docker.com/docker-for-windows/install/

Exercise 1 – hello-world

- Run it!

```
docker run hello-world
```

- Review https://hub.docker.com/_/hello-world/
- Pull

```
docker pull debian:9
```

- Check C:\Users\Public\Documents\Hyper-V\Virtual hard disks

Exercise 1 – hello-world

- Run an interactive session in Debian 9

```
docker run –i -t debian:9 /bin/bash
```

- Run a detached nginx instance

```
docker run –d nginx
```

- Launch a bash process in the detached nginx instance

```
docker exec -it <id> /bin/bash
```

Exercise 1 – hello-world

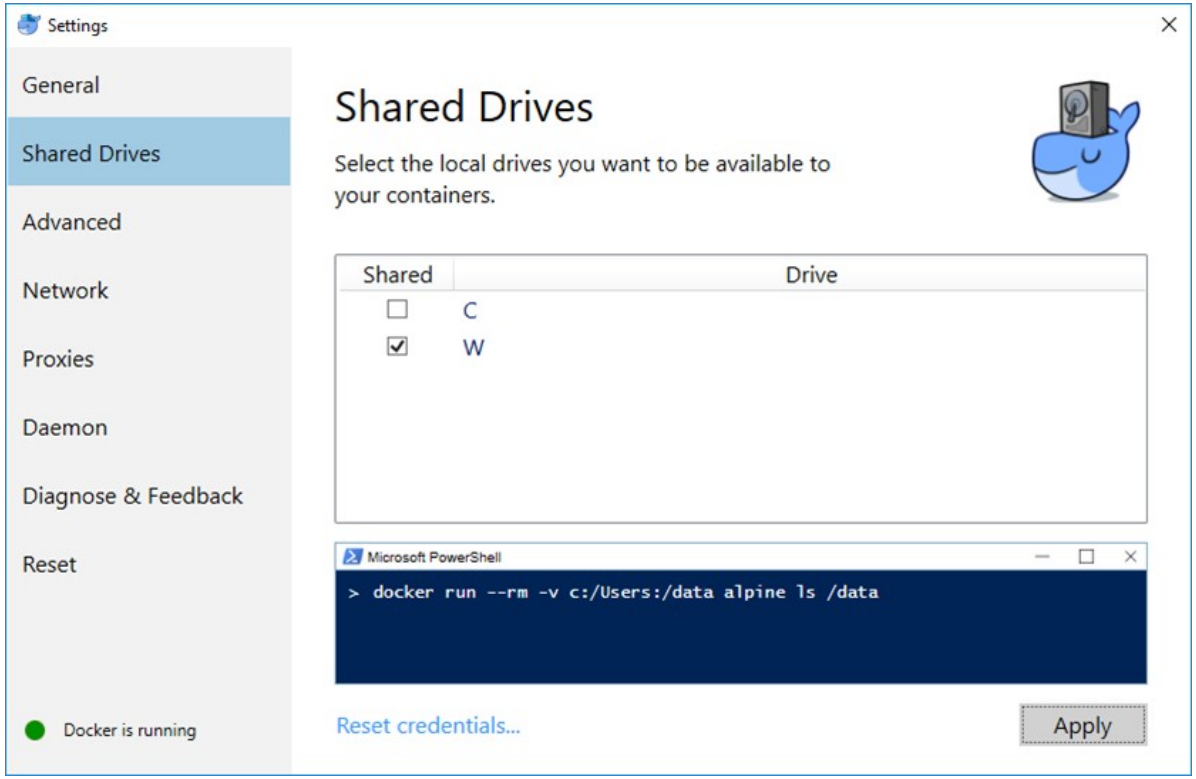
- Attach to the detached nginx instance

```
docker attach <id>
```

- Housekeeping commands

```
docker stop
docker rm
docker images
docker rmi
```

Exercise 2 – Externalities



Exercise 2 – Externalities

- Mounting file system volumes

```
docker run –it –v W:\data:/data debian:9 /bin/bash
```

- Exposing ports

```
docker run -it -p 8080:80 nginx
```

- Environment variables

```
docker run -it -e "FOO=bar" debian:9 /bin/bash
root@8e035b9c48d9:/# echo $FOO
```

Exercise 3 – Build a dotnetcore app

- Launch a build environment

```
docker run –it –v W:\wrk\bjm_str_px_docker_dotnet\hello:/hello-world
microsoft/dotnet:2-sdk /bin/bash
```

- Navigate to the mounted project directory

```
cd /hello-world
```

Exercise 3 – Build a dotnetcore app

- Build as a FDD (Framework Dependent Deployment)

```
dotnet build -c Release hello.csproj
dotnet publish -c Release hello.csproj
```

- References:
  - https://docs.microsoft.com/en-us/dotnet/core/deploying/index
  - https://docs.microsoft.com/en-us/dotnet/core/tools/dotnet-build
  - https://docs.microsoft.com/en-us/dotnet/core/tools/dotnet-publish
  - https://docs.microsoft.com/en-us/dotnet/core/rid-catalog

Exercise 3 – Build a dotnetcore app

- Run the app

```
dotnet bin/Release/netcoreapp2.0/publish/hello.dll
```

- Quit the container

```
docker ps -a
```

- Clean up the container

```
docker rm <id>
```

Exercise 4 – Dockerize a dotnetcore app

- Create the Dockerfile

```
FROM microsoft/dotnet:2-runtime

RUN mkdir -p /hello-world/

COPY bin/Release/netcoreapp2.0/publish/* /hello-world/

CMD ["dotnet", "/hello-world/hello.dll"]
```

- Build the image

```
docker build –t bren/hello .
```

Exercise 4 – Dockerize a dotnetcore app

- Run it

```
docker run bren/hello
```

Exercise 5 – Customize the dotnetcore SDK container

- Create the Dockerfile

```
FROM microsoft/dotnet:2-sdk

RUN apt-get update -y; apt-get upgrade -y;
RUN apt-get install -y \
     nano \
     vim
```

- Build and run your custom SDK environment

```
docker build –t bren/dotnetsdk .
docker run –it bren/dotnetsdk /bin/bash
```

b@bren.cc

http://u.bren.cc/github

brendon.matheson

http://u.bren.cc/linkedin

http://u.bren.cc/youtube

http://u.bren.cc/twitter