Microsoft

Module 06: Building Containers

Docker Workshop

1

Agenda

- The Dockerfile
- $ docker build
- Dockerfile Instructions
- Lab 05: Building your first image
- Lab 06: Build more complex images
- Best Practices & Common Mistakes

2

The Dockerfile

- Dockerfile (with capital D)
- It's comprised of instructions that define how to build an image
- These instructions get read one at time, from top to bottom
- When we build images from Dockerfile, any other files and directories in the same directory as the Dockerfile were going to get included in the build (build context).

```
FROM ubuntu:14.04
RUN \
  apt-get update && \
  apt-get -y install apache2

VOLUME /myvol
ADD index.html /var/www/html/index.html

EXPOSE 80

CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

3

$ docker build

```
$ docker build -t name:tag  .
```

```
$ docker build github.com/creack/docker-firefox
```

- Create a new Docker image following the Dockerfile instructions
- You can specify a Dockerfile in the filesystem or build an image from a Dockerfile stored in GitHub
- Tags are used to manage image versions

4

Dockerfile – Instructions

FROM

- Sets the Base Image for subsequent instructions
- The image can be any valid image (usually a container start by pulling an image from the Docker Hub)

```
FROM ubuntu:18.04
```

5

Dockerfile – Instructions

LABEL

- Add metadata to the docker image
- Used to indicate things like the image maintainer, version, description, etc

```
FROM ubuntu:18.04
LABEL maintainer="tomer@microsoft.com"
LABEL version="1.0"
LABEL description="my image description"
```

6

Dockerfile – Instructions

RUN

- Used to run commands against our images that we're building
- Every run instruction adds a layer to our image
- Run commands are the image "build steps"

```
FROM ubuntu:18.04
LABEL maintainer="tomer@microsoft.com"
RUN apt-get update
RUN apt-get install -y apache2
RUN apt-get install -y vim
RUN apt-get install -y apache2-utils
```

7

Dockerfile – Instructions

CMD

- Is the command executed anytime we launch a container from this image
- This command can be overridden in the run command
- Two types of sintax:
  - Shell: echo "Hello World"
  - Exec: ["echo", "Hello World"]

```
FROM ubuntu:18.04
LABEL maintainer="tomer@microsoft.com"
RUN apt-get update
RUN  apt-get install -y apache2
RUN  apt-get install -y vim
RUN apt-get install -y apache2-utils
CMD ["echo","Hello World!"]
```

8

Lab 05: Building your first image

Lab

Lab-05.md - Repos (azure.com)

9

Build Cache

- When we build a new image the docker daemon iterates through our Dockerfile executing each instruction.
- As each instruction gets executed, the daemon checks to see whether it's got an image for that instruction already in its build cache
- The build cache store each instruction + linked image
- (Change the docker file invalidates the build cache)

10

Dockerfile – Instructions

EXPOSE

- The EXPOSE instruction informs Docker that the container listens on the specified network ports at runtime.
- You can expose one port number and publish it externally under another number.

```
FROM ubuntu:18.04
LABEL maintainer="tomer@microsoft.com"
RUN  apt-get update  &&  apt-get install -y \
            apache2 \
            vim \
            apache2-utils
EXPOSE  80
CMD  ["apache2ctl","-D","FOREGROUND"]
```

11

Dockerfile – Instructions

ENTRYPOINT

- Is the better method of specifying the default app to run inside of a container
- Anything we do specify at the end of the docker run command at runtime (or CMD instruction) get interpreted as arguments to the entry point instruction

```
FROM ubuntu:18.04
LABEL maintainer="tomer@microsoft.com"
RUN  apt-get update  &&  apt-get install -y \
            apache2 \
            vim \
            apache2-utils
EXPOSE  80
ENTRYPOINT  ["echo" , "hello-world"]
```

12

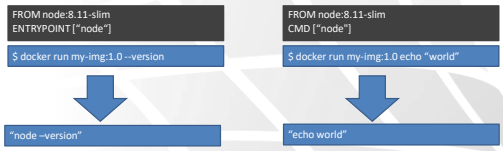
CMD vs. ENTRYPOINT

- You want to allow your users to Override the default behavior (usually utility containers ,or "all-in-one" containers)
- => CMD
- You want to "finalize" the behavior and not allow users to override it with their custom commands
- => ENTRYPOINT

13

CMD vs. ENTRYPOINT

- When you have ENTRYPOINT on your Dockerfile passing "commands" at the end of docker run become the argument list.



14

Dockerfile – Instructions

ENV

- Used to assign environment variables
- Environment variables can be overridden in the docker run command using **-e "var=value"**

```
FROM ubuntu:18.04
LABEL maintainer="tomer@microsoft.com"
RUN apt-get update && apt-get install -y \
        apache2 \
        vim \
        apache2-utils
ENV var1=val1
EXPOSE 80
ENTRYPOINT echo $var1
```

15

Playground

- What does this command mean :
- Docker run --name busy -e Key=Val busybox env
- Why is it working ? Browse the Busybox Docker file here

16

Dockerfile – Instructions

ARG

- The ARG instruction defines variables used at build-time
- Arguments can be passed in the docker build command using the flag **--build-arg <varname>=<value>**

```
FROM ubuntu:18.04
LABEL maintainer="tomer@microsoft.com"
RUN apt-get update && apt-get install -y \
        apache2 \
        vim \
        apache2-utils
ENV var1=val1
ARG var2=val2
RUN echo $var2
EXPOSE 80
ENTRYPOINT echo $var1
```

17

Dockerfile – Instructions

COPY

- Copies new files or directories from <src> and adds them to the filesystem of the container at the path <dest>
- Each <src> may contain wildcards

```
FROM ubuntu:18.04
LABEL maintainer="tomer@microsoft.com"
RUN apt-get update && apt-get install -y \
        apache2 \
        vim \
        apache2-utils
COPY hom* /mydir
ENV var1=val1
ARG var2=val2
RUN echo $var2
EXPOSE 80
ENTRYPOINT ["echo"]
```

18

Dockerfile – Instructions

ADD

- Similar than Copy but:
  - ADD allows <src> to be an URL
  - If the <src> parameter of ADD is an archive in a recognized compression format or a URL, it will be unpacked

```
FROM ubuntu:18.04
LABEL maintainer="tomer@microsoft.com"
RUN apt-get update && apt-get install -y \
            apache2 \
            vim \
            apache2-utils
ADD test.tar.gz /mydir
ENV var1=val1
ARG var2=val2
RUN echo $var2
EXPOSE 80
ENTRYPOINT ["echo"]
```

19

Dockerfile – Instructions

WORKDIR

- Used to set the working directory inside the container

```
FROM ubuntu:18.04
LABEL maintainer="tomer@microsoft.com"
RUN apt-get update && apt-get install -y \
            apache2 \
            vim \
            apache2-utils
WORKDIR /mydir
ADD test.tar.gz .
ENV var1=val1 var2=val2
ARG var2=val2
RUN echo $var2
EXPOSE 80
ENTRYPOINT ["echo"]
```

20

Multi layer Dockerfile

Base build image

Restore (nugget)

Publish

Execute

```
# https://hub.docker.com/_/microsoft-dotnet
FROM mcr.microsoft.com/dotnet/sdk:5.0 AS build
WORKDIR /source

# copy csproj and restore as distinct layers
COPY *.csproj .
RUN dotnet restore

# copy and publish app and libraries
COPY . .
RUN dotnet publish -c release -o /app --no-restore

# final stage/image
FROM mcr.microsoft.com/dotnet/runtime:5.0
WORKDIR /app
COPY --from=build /app .
ENTRYPOINT ["dotnet", "dotnetapp.dll"]
```

https://github.com/dotnet/dotnet-docker/blob/master/samples/dotnetapp/README.md

21

Lab 06: Building more complex images

Lab

Lab-06.md - Repos (azure.com)

22

Dockerfile Common Mistakes

- Using "latest" tag in base images
- Using external services during the build
- Adding EXPOSE and ENV at the top of your Dockerfile
- Add the **ENTIRE** app directory at the beginning of the Dockerfile
- Wrong multiple FROM statements
- Multiple services running in the same container

23

Dockerfile Best Practices

- Use a .dockerignore file
- Containers should be immutable & ephemeral (no data inside)
- Minimize the number of layers / Consolidate instructions
- Avoid installing unnecessary packages
- Sort multi-line arguments
- Use Build cache
- Understand CMD and ENTRYPOINT

24

The alpine base image

- Alpine is a vert tiny Linux image which "weights" only 3.6 MB !

DISTRIBUTION	VERSION	SIZE
Debian	jessie	123MB
CentOS	7	193MB
Fedora	25	231MB
Ubuntu	16.04	118MB
Alpine	3.6	3.98MB

25

The alpine base image

Debian

```
time docker run --rm debian sh -c "apt-get update && apt-get install curl"

real    0m27.928s
user    0m0.019s
sys     0m0.077s
```

Alpine

```
time docker run --rm alpine sh -c "apk update && apk add curl"

real    0m5.698s
user    0m0.008s
sys     0m0.037s
```

26

Docker Builder Pattern

- Pattern used to create small images

Build Container

Compiler
Dev Dependencies
Unit Tests
etc

Runtime Container

Runtime Env
Debug/Monitor
Tooling

- Benefits:
  - Size
  - Performance
  - Security

27

Docker Builder Pattern

700 MB
392 Vulnerabilities

```
FROM go:onbuild
WORKDIR /app
ADD . /app
RUN cd /app && go build -o goapp
EXPOSE 8080
ENTRYPOINT ./goapp
```

VS

12 MB
3 Vulnerabilities

```
FROM golang:alpine AS build-env
WORKDIR /app
ADD . /app
RUN cd /app && go build -o goapp

FROM alpine
RUN apk update && apk add ca-
certificates && rm -rf /var/cache/apk/*
WORKDIR /app
COPY --from=build-env /app/goapp /app

EXPOSE 8080
ENTRYPOINT ./goapp
```

28

What is a better approach?

(A)

```
FROM python:3.6
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
ENTRYPOINT ["python"]
CMD ["ap.py"]
```

(B)

```
FROM python:3.6
WORKDIR /app
COPY requirements.txt /app/requirements.txt
RUN pip install -r requirements.txt
COPY . /app
ENTRYPOINT ["python"]
CMD ["ap.py"]
```

- **B)** The COPY . /app command will invalidate the cache as soon as any file in the current directory is updated.

29

What is a better approach?

(A)

```
FROM centos:7

RUN yum update -y
RUN yum install -y sudo
RUN yum install -y git
RUN yum clean all
```

470 MB

(B)

```
FROM centos:7

RUN yum update -y \
 && yum install -y \
 sudo \
 git \
 && yum clean all
```

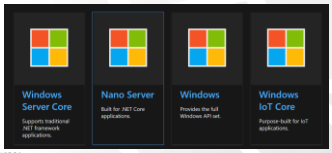
265 MB

- **B)** In "A" many layers were added for nothing. Also the "yum clean all" command is meant to reduce the size of the image but it actually does the opposite by adding a new layer

30

Building Windows Container

- Windows offers 4 base images to build from :



- Source: https://docs.microsoft.com/en-us/virtualization/windowscontainers/manage-containers/container-base-images

31

Build Asp.net windows Container

Lab

Lab-07.md - Repos (azure.com)

32

Questions

33