Docker: Up and Running

Docker Images

Release 2020-03-08-1013

Instructor

Sean P. Kane

@spkane

Follow Along Guide

Textual Slides

Copyright © 2015-2020, Sean P. Kane (@spkane)

Prerequisites

- A recent computer and OS
  - Recent Linux, OS X, or Windows 10
  - root/admin rights
  - Sufficient resources to run one 2 CPU virtual machine (VM)
  - CPU Virtualization extensions MUST be enabled in your BIOS/EFI
  - Reliable and fast internet connectivity
- Docker Community Edition

Copyright © 2015-2020, Sean P. Kane (@spkane)

Prerequisites

- A graphical web browser
- A text editor
- A software package manager
- Git client
- General comfort with the command line will be helpful.
- [optional] tar, wget, curl, jq, SSH client

Copyright © 2015-2020, Sean P. Kane (@spkane)

A Note for Windows Users

This class was written from a largely Unix based perspective, but everything can be made to work in Windows with very little effort.

- Unix Variables
  - `export MY_VAR=test`
  - `echo ${MY_VAR}`
- Windows 10 Variables (powershell)
  - `$env:my_var = "test"`
  - `Get-ChildItem Env:my_var`

Copyright © 2015-2020, Sean P. Kane (@spkane)

A Note About Proxies

Proxies can interfere with some Docker activities if they are not configured correctly.

If required, you can configure a proxy in Docker: Community Edition via the preferences.

Copyright © 2015-2020, Sean P. Kane (@spkane)

Instructor Environment

- **Operating System**: Mac OS X (v10.15.X+)
- **Terminal**: iTerm2 (Build 3.X.X+) - https://www.iterm2.com/
- **Shell Customization**: Bash-it - https://github.com/Bash-it/bash-it
- **Shell Prompt Theme**: Starship - https://starship.rs/
- **Shell Prompt Font**: Fira Code - https://github.com/tonsky/FiraCode
- **Text Editor**: Visual Studio Code (v1.X.X+) - https://code.visualstudio.com/

Copyright © 2015-2020, Sean P. Kane (@spkane)

Docker in Translation

- **Docker client**
  - The docker command used to control most of the Docker workflow and talk to remote Docker servers.
- **Docker server**
  - The dockerd command used to launch the Docker daemon. This turns a Linux system into a Docker server that can have containers deployed, launched, and torn down via a remote client.

Copyright © 2015-2020, Sean P. Kane (@spkane)

Docker in Translation

- **Virtual Machine**
  - In general, the docker server can be only directly run on Linux. Because of this, it is common to utilize a Linux virtual machine to run Docker on other development platforms. Docker Community Edition makes this very easy.

Copyright © 2015-2020, Sean P. Kane (@spkane)

Docker in Translation

- **Docker images**
  - Docker images consist of one or more filesystem layers and some important metadata that represent all the files required to run a Dockerized application. A single Docker image can be copied to numerous hosts. A container will typically have both a name and a tag. The tag is generally used to identify a particular release of an image.

Copyright © 2015-2020, Sean P. Kane (@spkane)

Docker in Translation

- **Linux Containers**
  - A Linux Container is a single instantiation of a Docker or OCI-standard image. A specific container can only exist once; however, you can easily create multiple containers from the same image.
  - OCI - Open Container Initiative

Copyright © 2015-2020, Sean P. Kane (@spkane)

Testing the Docker Setup

```
$ docker images
$ docker run -d --rm --name quantum -p 18080:8080 \
    spkane/quantum-game:latest
$ docker ps
```

- In a web browser, navigate to port 18080 on your Docker server.
  - (e.g.) http://127.0.0.1:18080/

```
$ docker stop quantum
$ docker ps
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

Exploring the Dockerfile

```
$ cd ${HOME}
$ mkdir class-docker-images
$ cd ${HOME}/class-docker-images
$ git clone https://github.com/spkane/balance_game.git \
    --config core.autocrlf=input
$ cd balance_game
```

- Open & explore Dockerfile in your text editor

Full Documentation:
https://docs.docker.com/engine/reference/builder/

Copyright © 2015-2020, Sean P. Kane (@spkane)

Registering with Docker Hub

Create an account at: https://hub.docker.com/

```
$ docker login
$ cat ~/.docker/config.json
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

Registry Authentication

```
{
    "auths": {
        "https://index.docker.io/v1/": {
            "auth": "q378t348q7tb78bfs387b==",
            "email": "me@example.com"
        }
    }
}
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

Create Your Image Repository

- **Login**: `https://hub.docker.com/`
- **Click**: `Create Repository+`
- **Enter name**: `balance_game`
- **Set visibility**: `public`
- **Click**: `Create`

Copyright © 2015-2020, Sean P. Kane (@spkane)

Building Your First Image

```
$ export HUB_USER=${USER}
$ docker build -t ${HUB_USER}/balance_game:latest .
$ docker push ${HUB_USER}/balance_game:latest
$ docker search ${HUB_USER}
$ docker run -d --rm --name balance_game -p 18081:80 \
    ${HUB_USER}/balance_game:latest
$ docker stop balance_game
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

Docker Hub API Examples

```
$ curl -s -S \
    "https://registry.hub.docker.com/v2/repositories/library/alpine/tags/"
    | jq '."results"[]["name"]' | sort
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

A Typical Build Process

1. Base image pulled, if required
2. New intermediate container created from base image
   - or empty container created, if using `FROM scratch`
3. Dockerfile command executed inside intermediate container
4. New image/layer created from intermediate container
5. If there is another step, a new intermediate container is created from the last step, and then the build goes back to step 3.

Copyright © 2015-2020, Sean P. Kane (@spkane)

Advanced Dockerfile Techniques

Copyright © 2015-2020, Sean P. Kane (@spkane)

Keep it Small

- Each and every layer's size matters
- Don't install unnecessary files

```
$ cd ${HOME}/class-docker-images
$ docker run --rm -d --name outyet-small -p 8090:18088 \
    spkane/outyet:1.9.4-small
$ docker export outyet-small -o export.tar
$ tar -tvf export.tar
$ docker stop outyet-small
$ rm export.tar
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

Debugging an Image

- If your image has a shell installed you can access it using a command like this:

```
$ docker images
$ docker run --rm -ti spkane/outyet:1.9.4-small /bin/sh
```

But without a shell in the image this will fail.

Copyright © 2015-2020, Sean P. Kane (@spkane)

Debugging an Image

So, let's fix this:

```
$ git clone https://github.com/spkane/outyet.git \
    --config core.autocrlf=input
$ cd outyet
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

Multi-Stage Images

```
FROM golang:1.9.4

COPY . /go/src/outyet
WORKDIR /go/src/outyet

ENV CGO_ENABLED=0
ENV GOOS=linux

RUN go get -v -d && \
    go install -v && \
    go test -v && \
    go build -ldflags "-s" -a -installsuffix cgo -o outyet .
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

Multi-Stage Images

```
# To support debugging, let's use alpine instead of scratch
FROM alpine:latest

# Since we are using alpine we can simply install these
RUN apk --no-cache add ca-certificates

WORKDIR /
COPY --from=0 /go/src/outyet/outyet .

EXPOSE 18088

CMD ["/outyet", "-version", "1.9.4", "-poll", "600s", "-http", ":18088"]
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

Building the Improved Image

```
$ docker build -f Dockerfile -t outyet:1.9.4-local .
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

Debugging an Image

- Now that we have a shell, let's try this again:

```
$ docker images
$ docker run --rm -ti outyet:1.9.4-local /bin/sh
```

- Once inside the new container:

```
$ ls -lFa /outyet
$ exit
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

Debugging a Broken Build (1 of 2)

- Break the `Dockerfile` and then try building it again.

```
RUN apc --no-cache add ca-certificates
```

```
$ docker build -f Dockerfile .
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

Debugging a Broken Build (2 of 2)

- Let's debug the last successful image in that build

```
$ docker run --rm -ti ${IMAGE_ID} /bin/sh
```

- Once inside the new container:

```
$ apc --no-cache add ca-certificates
$ apk --no-cache add ca-certificates
$ exit
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

Smart Layering

- Each and every layer's size matters
- Clean up, inside of each step.

```
$ cd ${HOME}/class-docker-images
$ cd balance_game
$ docker build -f Dockerfile.fedora .
$ docker tag ${IMAGE_ID} size${#}
$ docker history size${#}
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

Smart Layering

- edit `Dockerfile.fedora`, build, and re-examine size

```
RUN dnf install -y httpd
RUN dnf clean all
```

```
RUN dnf install -y httpd && \
    dnf clean all
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

Timing commands in Windows

- In the next exercise we will be timing commands using a Unix utility. If you are on Windows and want to try to time these commands locally, you can try something like this in Powershell.

```
PS C:\> $t = Measure-Command { docker build --no-cache . }
PS C:\> Write-Host That command took $t.TotalSeconds to complete.
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

Order Matters

- Keep commands that change towards the end of your Dockerfile.

```
$ cd ${HOME}/class-docker-images
$ cd balance_game
$ time docker build --no-cache .
$ time docker build .
```

- edit `start.sh`

```
$ time docker build .
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

Order Matters

- Add to the top of Dockerfile:

```
ADD start.sh /
```

- And then remove the same line from the bottom of the Dockerfile.

```
$ time docker build --no-cache .
$ time docker build .
$ vi start.sh
$ time docker build .
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

Private Registry

```
$ git clone https://github.com/spkane/docker-registry.git \
    --config core.autocrlf=input
$ cd docker-registry
$ docker-compose up -d
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

Private Registry

```
$ docker login 127.0.0.1:5000
$ docker image ls registry
$ docker tag $(docker image ls registry --format "{{.ID}}") \
    127.0.0.1:5000/registry
$ docker push 127.0.0.1:5000/registry
$ docker-compose down
```

Copyright © 2015-2020, Sean P. Kane (@spkane)

What We Have Learned

- What a Dockerfile is
- Building a Docker image
- Using Docker Hub
- Keeping Images Small
- Keeping Builds Fast
- Multi-Stage Dockerfiles
- Debugging Images
- Creating a Private Registry

Copyright © 2015-2020, Sean P. Kane (@spkane)

Additional Reading

- The 12-Factor App
  - http://12factor.net/
- Official Docker Documentation
  - https://docs.docker.com/
- Docker: Up and Running
  - http://shop.oreilly.com/product/0636920153566.do

Copyright © 2015-2020, Sean P. Kane (@spkane)

Additional Learning Resources

https://learning.oreilly.com/

Copyright © 2015-2020, Sean P. Kane (@spkane)

Student Survey

Please take a moment to fill out the class survey linked to in the chat channel.

O’Reilly and I value your comments about the class.

Thank you!

Copyright © 2015-2020, Sean P. Kane (@spkane)

Any Questions?

Copyright © 2015-2020, Sean P. Kane (@spkane)