CS 340
Computer Systems

#17: Containers, CaaS, and Docker

October 20, 2022 · Wade Fagen-Ulmschneider

IaaS vs. CaaS

When we use IaaS, a blank operating system with only the default software is provided.

- As an IaaS user:
- As a container developer:
- As a container consumer:

Containers are **isolated environments** that have their own dedicated RAM, CPU access, disks, network ports, etc.

A Dockerfile specifies how a container should be built:

17/Dockerfile-01

```
FROM alpine
ENTRYPOINT ["/bin/sh"]
```

[Line 1]: **FROM** `<image>`

[Line 2]: **ENTRYPOINT** `[<command>]`

```
$ docker build -t test -f Dockerfile-01 .
```

Running a docker container:

```
$ docker run test
```

Q: What happens?

- Fix:

Common **`docker run`** arguments:

```
$ docker run                 test
```

One of the most important things to do is to add your files into your container:

17/Dockerfile-02

```
FROM alpine
COPY cs340 /inside-of-docker-filesystem
ENTRYPOINT ["/bin/sh"]
```

[Line 2]: **COPY** `<local path> <container path>`

You may need to run a command on **building** the image:

17/Dockerfile-03

```
FROM alpine
COPY cs340 /inside-of-docker-filesystem
RUN /inside-of-docker-filesystem/create.sh
ENTRYPOINT ["/bin/sh"]
```

[Line 3]: **RUN** `<command>`

Q: What do we expect to happen?

17/cs340/create.sh

```
echo "Bye" >bye.txt
```

-

You can change the working directory:

17/Dockerfile-04

```
FROM alpine
COPY cs340 /inside-of-docker-filesystem
WORKDIR /inside-of-docker-filesystem
RUN create.sh
ENTRYPOINT ["/bin/sh"]
```

Bridging Resources with the Host System

If you want the use of any host system resources, you must **explicitly** give them to the docker when you **launch the container**:

```
$ docker run --rm -it -v `pwd`:/mount test
```

```
$ docker run --rm -it -p 34000:34000
```

Docker Images as Building Blocks

Every dockerfile starts with a `FROM <image>` -- all the way down to `FROM scratch` (an image that contains no starting environment).

`cs340-mp6` image:

```
FROM python:3.9   [...]
```

`python:3.9` image:

```
FROM buildpack-deps:buster   [...]
```

`buildpack-deps:buster` image:

```
FROM buildpack-deps:buster-scm   [...]
```

`buildpack-deps:buster-scm` image:

```
FROM buildpack-deps:buster-curl   [...]
```

`buildpack-deps:buster-curl` image:

```
FROM debian:buster   [...]
```

`debian:buster` image:

```
FROM scratch
ADD rootfs.tar.xz /
CMD ["bash"]
```

Developer Uses of Containers

Containers allow us to fully configured services quickly, immediately, and without any concerns about the system runtime.

Example:

```
$ docker run -it --rm
    -p 27017:27017
    -v `pwd`/mongodb/:/data/db
    mongo
```

Windows PowerShell: Use `-v ${PWD}/mongodb/:/data/db` instead.

When the Docker is running, we can start programming using Mongo:

17/artist.py

```python
from pymongo import MongoClient
mongo = MongoClient('localhost', 27017)
db = mongo["17-artist-database"]

store = db["waf"]
doc = store.find_one({
  "Favorite Artist": {"$exists": True }
})

if doc:
  print("Favorite Artist: ")
  print(doc)
else:
  store.insert_one({
    "Favorite Artist": "Taylor Swift"
  })
  print("Added Favorite Artist!")
```

Q: What happens if we restart the docker container after running this program several times?

Q: What happens if we remove the -v flag in our run command?