

## 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:

### 16/Dockerfile-01

```
1 FROM alpine
2 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:

### 16/Dockerfile-02

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

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

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

### 16/Dockerfile-03

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

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

**Q:** What do we expect to happen?

### 16/cs340/create.sh

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

-

You can change the working directory:

```
16/Dockerfile-04
1 FROM alpine
2 COPY cs340 /inside-of-docker-filesystem
3 WORKDIR /inside-of-docker-filesystem
4 RUN create.sh
5 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
```

Natively on Windows:

- Use PowerShell
- Use `-v ${PWD}/mongodb/./data/db` for -v option.

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

```
16/artist.py
1 from pymongo import MongoClient
2 mongo = MongoClient('localhost', 27017)
3 db = mongo["17-artist-database"]
4
5 store = db["waf"]
6 doc = store.find_one({
7     "Favorite Artist": {"$exists": True }
8 })
9
10 if doc:
11     print("Favorite Artist: ")
12     print(doc)
13 else:
14     store.insert_one({
15         "Favorite Artist": "Taylor Swift"
16     })
17     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?