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:

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

[Line 1]: `FROM <image>

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

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

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

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

Q: What do we expect to happen?

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

```
$ 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:

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

```
COPY <local path> <container path>
```

```
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?

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

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

You can change the working directory:

```
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**:

```bash
$ 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:**

```
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:**

```
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?