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>]`

Running a docker container:

```
$ docker run test
```

Q: What happens?

- Fix:

Attempt #2:

```
$ docker run test
```

Q: What happens?

- Clean Up:

Attempt #3:

```
$ docker run test
```

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

```
FROM alpine
COPY cs240 /inside-of-docker-filesystem
ENTRYPOINT ["/bin/sh"]
```

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

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

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

<table>
<thead>
<tr>
<th>16/Dockerfile-03</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 FROM alpine</td>
</tr>
<tr>
<td>2 COPY cs240 /inside-of-docker-filesystem</td>
</tr>
<tr>
<td>3 RUN /inside-of-docker-filesystem/create.sh</td>
</tr>
<tr>
<td>4 ENTRYPOINT [&quot;/bin/sh&quot;]</td>
</tr>
</tbody>
</table>

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

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

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

You can change the working directory:

<table>
<thead>
<tr>
<th>16/Dockerfile-04</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 FROM alpine</td>
</tr>
<tr>
<td>2 COPY cs240 /inside-of-docker-filesystem</td>
</tr>
<tr>
<td>3 WORKDIR /inside-of-docker-filesystem</td>
</tr>
<tr>
<td>4 RUN create.sh</td>
</tr>
<tr>
<td>5 ENTRYPOINT [&quot;/bin/sh&quot;]</td>
</tr>
</tbody>
</table>

**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).

**cs240-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"]
```

---

**Using Host System Resources**

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 24000:24000```

Many docker images are open-source and available via DockerHub (ex: GitHub but for docker) -- however, you can host private images or download them directly!