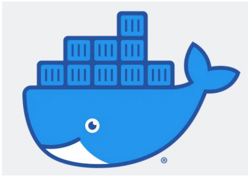
CS200
Software Tools & Technologies Lab II

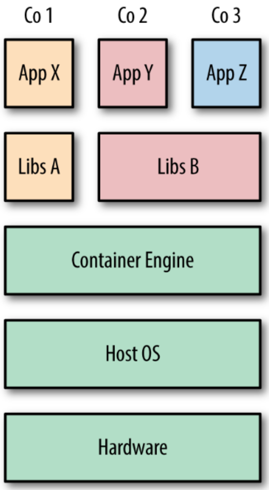
Session 10

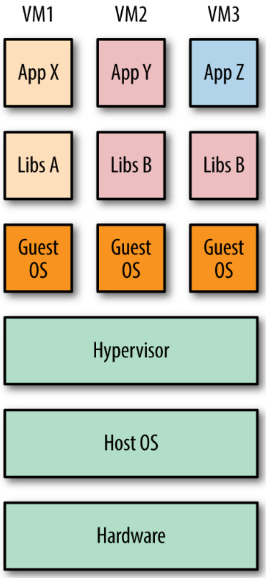
Docker Layers and Container Monitoring



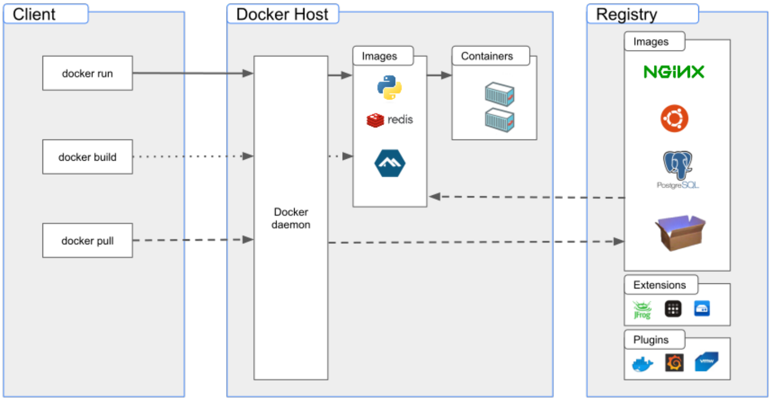
Instructor
Dr. Dhiman Saha

Containers Vs Virtual Machines

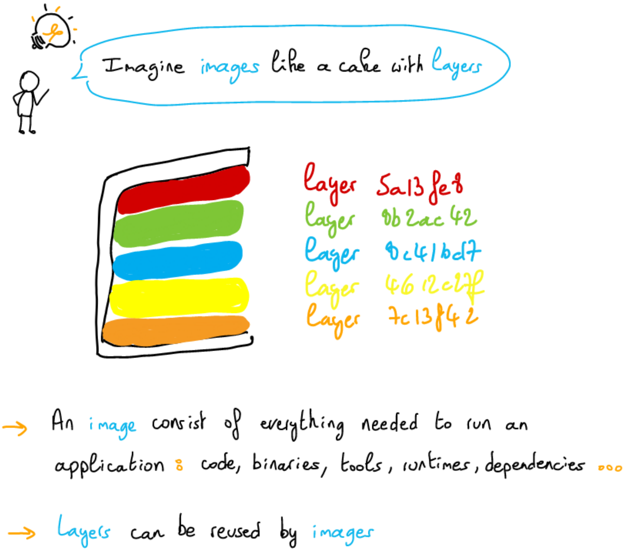




Docker Architecture



Layers in Docker



→ An image consist of everything needed to run an application : code, binaries, tools, runtimes, dependencies ...

→ Layers can be reused by images

Image source: Aurlie Vache - Understanding Docker in a visual way-leanpub.com (2021)

Dockerfile

```
FROM debian
RUN apt-get update
RUN apt-get install -y python3
RUN apt-get install -y python3-pip
RUN apt-get clean all

RUN pip3 install flask

ADD hello.py /tmp/hello.py

EXPOSE 5000

CMD ["python3","/tmp/hello.py"]
```

Fire it Up!!!

```
docker build -t flask .
```

```
docker run -d -P flask
```

docker ps

```
CONTAINER ID   IMAGE      COMMAND              CREATED          STATUS                        PORTS                                     NAMES
680d0759eee5   flask    "python3 /tmp/hello."  5 seconds ago  Up 4 seconds   0.0.0.0:49154– >5000/tcp, :::49154– >5000/tcp   condescending_goldwasser
```

- PORTS shows a mapping between port 5000 of the container and port 49154 of the Docker host[1].
- See it in action from Docker host!
  - Simple curl to `http://localhost:49154/hi`
  - Or open your browser to the same url.

[1] This might be a different port in your case.

EXP-15 Monitoring containers

- `date +''%s''` Save the value returned for later use.

Recall ping docker Experiment

```
docker images

REPOSITORY                TAG      IMAGE ID       CREATED
nginx                     latest   6efc10a0510f   6 days ago
flask                     latest   ef6a4063e7a5   8 days ago
<none>                    <none>   7c68670f4c42   8 days ago
<none>                    <none>   038f7d09b03a   9 days ago
test/ping-dockerfile      latest   6a717afb191b   9 days ago
debian                    ping     7a11db7ba1c0   9 days ago
alpine                    latest   9ed4aefc74f6   2 weeks ago
debian                    latest   f5b06fd90040   3 weeks ago
```

- `docker inspect test/ping-dockerfile`
- Can you **reverse-engineer** the Dockerfile?

Monitoring containers

- `docker run -d -p 80:80 nginx`
- `docker ps`

```
CONTAINER ID   IMAGE   COMMAND                   CREATED          STATUS          PORTS
d1dba430021b   nginx   "/docker-entrypoint."     34 minutes ago   Up 34 minutes   0.0.0.0:80->80/tcp, :::8
```

- docker stats

```
CONTAINER ID   NAME                  CPU %     MEM USAGE / LIMIT     MEM %     NET I/O       BLOCK I/O
d1dba430021b   trusting_heisenberg   0.00%     8.734MiB / 7.685GiB   0.11%     159kB / 0B    0B / 8.19kB
```

Run — ping image

- Check the docker stats after running the following command

```
docker run -d test/ping-dockerfile ping google.com
```

EXP-17 Monitoring containers

Problem You want to monitor Docker events on your host

- Recall the output of date command at the start of the class
- `docker events --since 1481876338`
- Interpret the output

docker logs

`docker logs <container-name/ID>`

- Continuous tracking
  `docker logs -f <container-name/ID>`

- Process monitoring inside container: `docker top`

The sen Github Project

- Spawn a container from ping image created earlier
- Run the following command

```
docker run -v /var/run/docker.sock:/run/docker.sock -ti -e TERM tomastomecek/sen
```

- Find out about the layers in the image.
- Find out about the realtime status of the containers

The `dive` Github Project

- `https://github.com/wagoodman/dive`
- `docker pull wagoodman/dive`

```
docker run --rm -it \
     -v /var/run/docker.sock:/var/run/docker.sock \
     wagoodman/dive:latest <dive arguments...>
```

- Find out about the layers in the `ping` image.

EXP-20 Private Registry

- Have you published your image to a public registry like DockerHub?

Your Own Registry

You would like to run your own Docker registry, hosting it on your own infrastructure.

- Pull the official registry image and run it as a detached container. You should then be able to curl `http://localhost:5000` for a quick test that the registry is running.

```
$ docker pull registry:0.9.1
$ docker run -d -p 5000:5000 registry:0.9.1
$ curl http://localhost:5000
```

- Expected output: `"\"docker-registry server\""`

Private Registry

- Next push an image into your private registry
- The registry is running at `http://localhost:5000`
- Prefix your tag with `localhost:5000` and then push this image to the private reg istry

```
$ docker tag flask localhost:5000/flask
$ docker push localhost:5000/flask
The push refers to a repository [localhost:5000/flask] (len: 1)
Sending image list
Pushing repository localhost:5000/flask (1 tags)
511136ea3c5a: Image successfully pushed
...
88d6464d1f42: Image successfully pushed
Pushing tag for rev [88d6464d1f42] on
{http://localhost:5000/v1/repositories/flask/tags/latest}
```