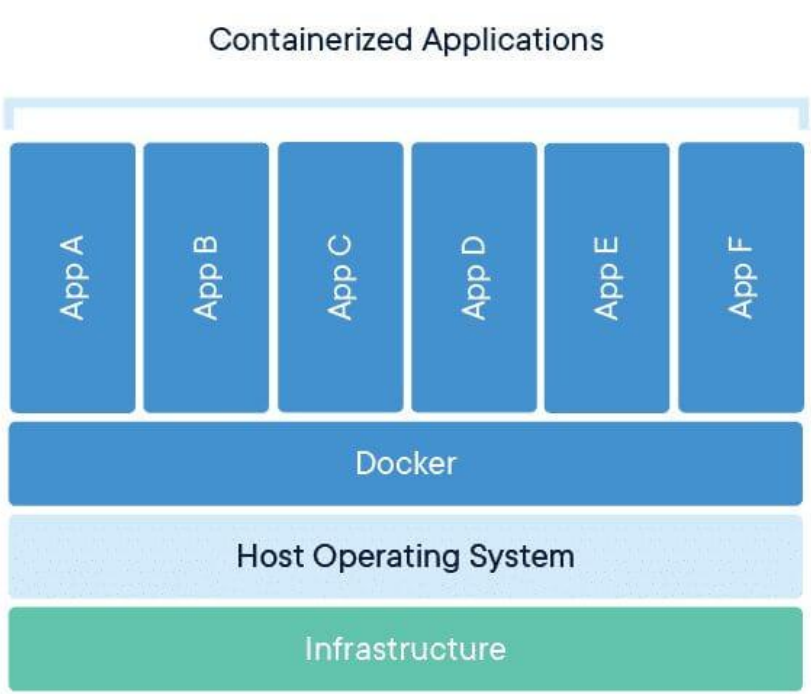
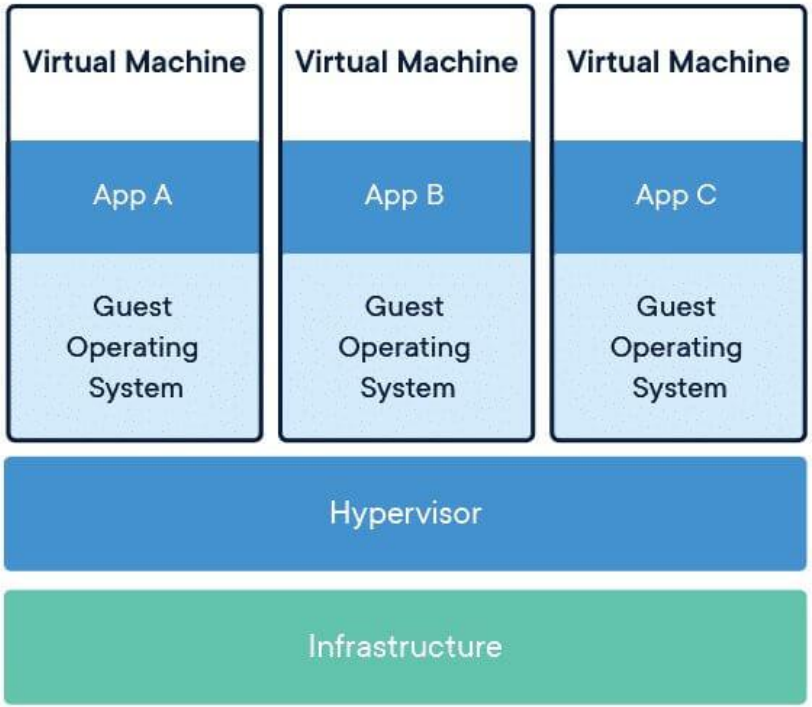
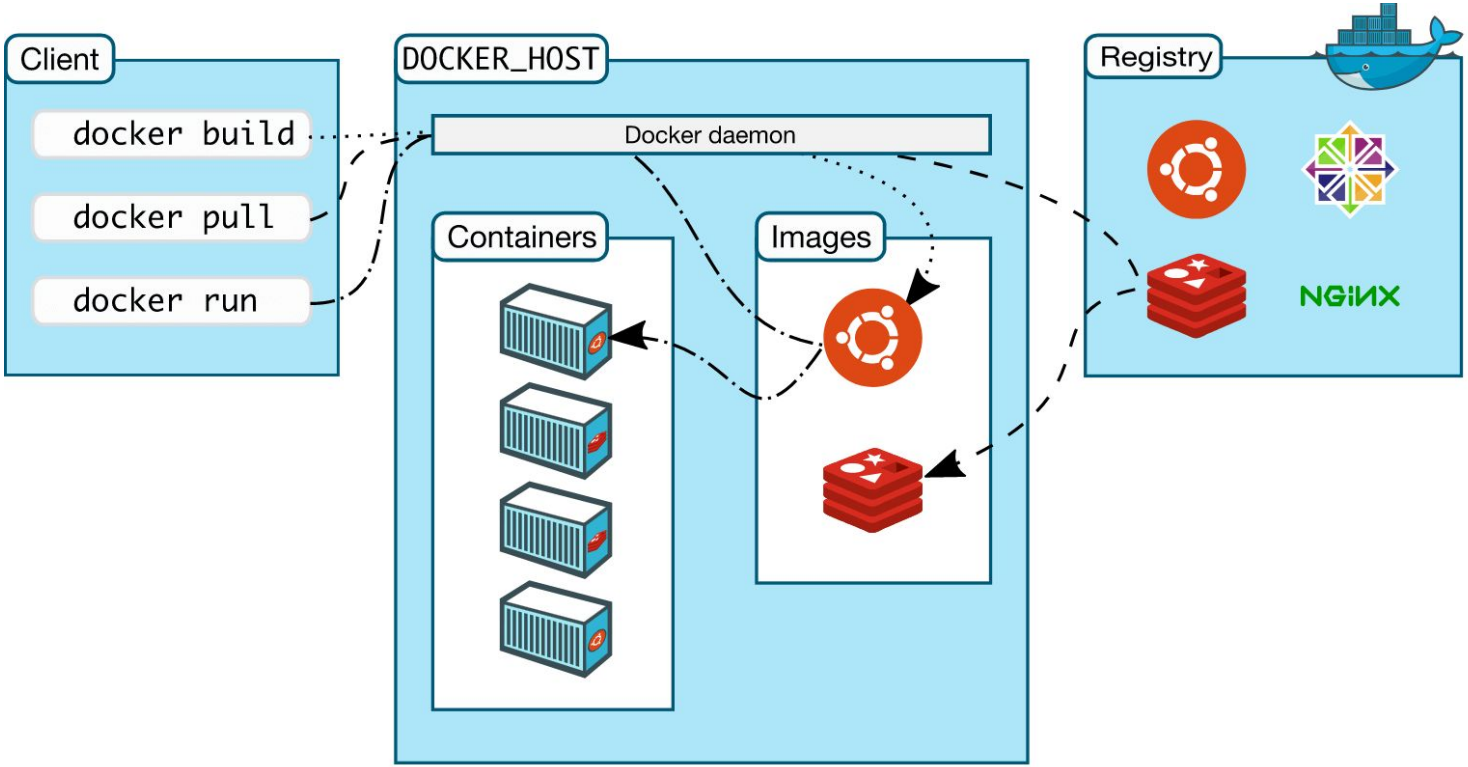
- Docker is a platform for consistently building, running, and shipping applications.





- A virtual machine is an abstraction of hardware resources. Using hypervisors we can create and manage virtual machines. The most popular hypervisors are VirtualBox, VMware and Hyper-v (Windows-only).
- A container is an isolated environment for running an application. It's essentially an operating-system process with its own file system.
- Using Docker, we can bundle an application into an image. Once we have an image, we can run it on any machine that runs Docker.
- An image is a bundle of everything needed to run an application. That includes a cutdown OS, a runtime environment (eg Node, Python, etc), application files, third-party libraries, environment variables, etc.
- To bundle an application into an image, we need to create a Dockerfile. A Docker file contains all the instructions needed to package up an application into an image.
- We can share our images by publishing them on Docker registries. The most popular Docker registry is Docker Hub.

Docker Architecture/Flow



- Install docker
- Make a directory and make a `app.js` file
- Make a Dockerfile:
  - `FROM node:alpine`
  - `WORKDIR /app`
  - `COPY . /app`
  - `CMD node /app.js`
- `docker build -t hello-world:<tag-name> .`
- `docker images` or `docker image ls`
- `docker run hello-world`
- `Publish`
- `Docker lab`

Running containers

```
docker run <image>
docker run -d <image>                  # run in the background
docker run —name <name> <image>        # to give a custom name
docker run —p 3000:3000 <image>        # to publish a port HOST:CONTAINER
```

Listing containers

```
docker ps                              # to list running containers
docker ps -a                           # to list all containers
```

Viewing the logs

```
docker logs <containerID>
docker logs -f <containerID>           # to follow the log
docker logs —t <containerID>           # to add timestamps
docker logs —n 10 <containerID>        # to view the last 10 lines
```

Executing commands in running containers

```
docker exec <containerID> <cmd>
docker exec -it <containerID> sh                # to start a shell
```

Starting and stopping containers

```
docker stop <containerID>
docker start <containerID>
```

Removing containers

```
docker container rm <containerID>
docker rm <containerID>
docker rm -f <containerID>                      # to force the removal
docker container prune                          # to remove stopped containers
```

- **Concepts of volumes (Please explore yourself)**

Copying files between the host and containers

docker cp <containerID>:/app/log.txt .
docker cp secret.txt <containerID>:/app

FOR OUR REACT APP

- Inside Dockerfile:

```
FROM node:<node-version>-alpine<alpine-version>
WORKDIR /app
COPY . .
RUN npm install
EXPOSE 3000
CMD [“npm”, “start”]
```

- For optimization(caching concept)
  - Copy package*.json .
  - Run npm install
  - Copy . .

- Use the concept of .dockerignore
- docker build -t react-app .
- docker run -it react-app sh
- docker run -p 3000:3000 <image-name>