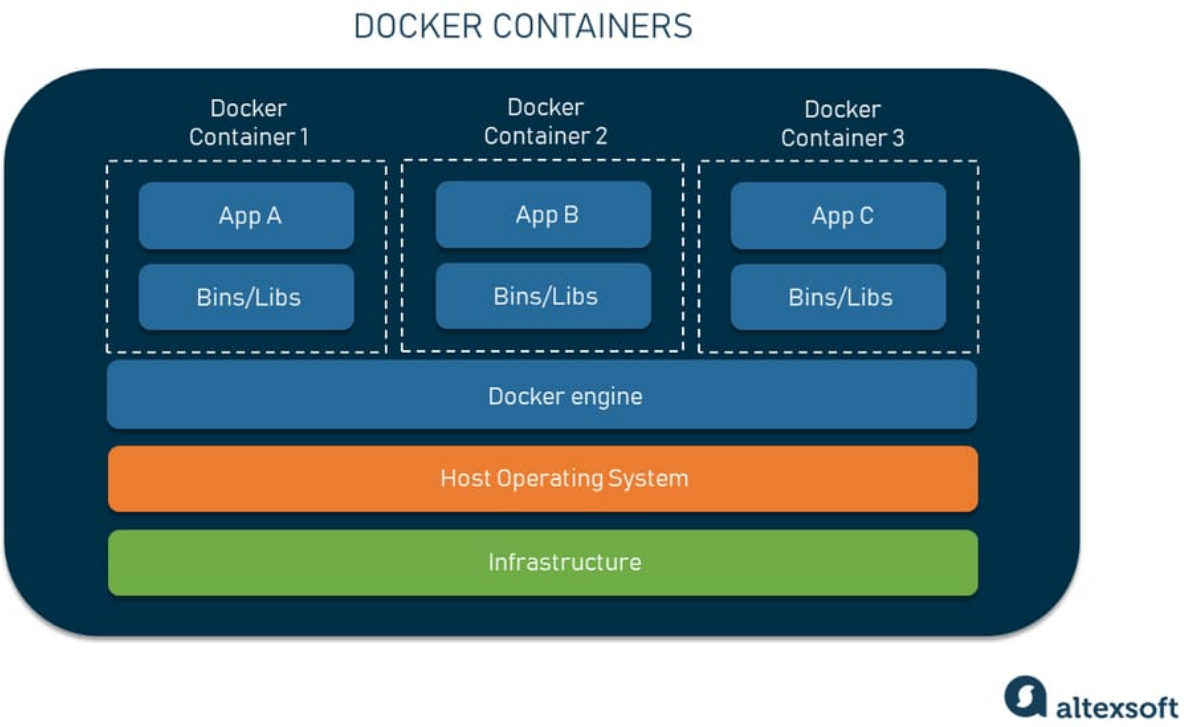
What is Docker & What problem it solves

- Packages the application along with the dependencies.
- Let's say you made a project with gradle, java or some jdk version and when you transferred the project to some other machine It needs those exact dependencies to run
- What if we developed on local, put on stage and then on prod. All these environments will have different dependencies and thus our project will not able to run
- To ship these containers we need to make some docker image to take a container up



DockerFile, Image and Container

- Docker file includes commands to make an image. Let's say if I want to build something then for that I will run some commands

first and then using this file an image (file) will be created.

- Docker container runs the entire application and comes up using an image. Running image can be called a container.

  DockerFile -- (build) --> Image -- (run) --> Container

> Install Docker for Windows/Linux/MacOS from their documentation

Test Docker installation & its commands

```
docker pull hello-world // can be pulled from docker
hub (hub.docker.com)

docker run hello-world
docker pull openjdk:18
docker pull python:x
docker images
docker search mysql
docker run python
docker ps -a
docker run --env MYSQL_ROOT_PASSWORD ...
docker run --detach
docker run -d, docker run -e

docker run --name pythonContainer -d <image-id>
docker ps -a

docker run --name pythonContainer -it -d <image-
name/image-id>
docker ps -a

docker exec -it <container-id>
>> you are inside the container
```

```
>> print("Hello world")

docker inspect <container-id>

docker run --name javaContainer -it -d <image-
name/image-id>
docker ps -a
docker exec -it javaContainer <container-id/command>
>> run System.out.Println("Hello world");

>> run mysql and try to connect from inside and outside
of the VM

docker exect -it mysql bash // run mysql commands
inside

docker pull nginx
// explore from docker hub, about exposing port and
running

docker stop <container-id/container-name/container-id-
starting-letters>
docker ps -a

docker stop <c-1> <c-2> <c-3> .....

docker rmi <image-name>

docker login
docker commit
docker push
docker copy
```

```
docker logs
docker volume
```

Building docker image

```
FROM ubuntu // base image
RUN apt update
CMD ["echo", "hello people, my first image"]
```

```
docker build -t myimage . // same folder
docker run --name myubuntuimage myimage
```

Java application using Docker

```
FROM openjdk:11 // This image itself will be based on
some OS like it is based on debian distribution of
linux
WORKDIR /usr/src/myapp
COPY . /usr/src/myapp
RUN javac Test.java
CMD ["java", "Test"]
```

```
docker build -t myjavaapp .
```

Note:- It's worth noting that the operating system inside the container is isolated from the host operating system, meaning that the host operating system could be different from Debian Linux. However, within the container, your Java application will run on Debian Linux.

Dockerizing Spring boot application

```
java -jar jarFileInBuildJar // to run any jar
```

```
FROM openjdk:17
WORKDIR /usr/src/myapp
COPY . /usr/src/myapp
CMD ["java", "-jar", "xyz.jar"]

EXPOSE 9590
```

```
docker build -t myjavaappimage .
docker run --name myproject -it -d -p 9595:9590
myjavaappimage
docker ps -a
docker logs myproject
docker run
```

> Run in local system to see if application is running or not

---

Docker Compose

It allows us to run multiple docker containers or applications in an environment. It is a one stop solution being used to run a number of applications.

Docker Networks

It allows docker container to connect with each other and the outer world. Let's try without docker compose:-

```
docker network create mongo-network
docker network ls

docker run -d \
-p 27017:27017 \
-e MONGO_INITDB_ROOT_USERNAME=admin \
-e MONGO_INITDB_ROOT_PASSWORD=password \
--network mongo-network \
--name mongodb \
mongo

docker run -d \
-p 8081:8081 \
-e ME_CONFIG_MONGODB_ADMINUSERNAME=admin \
-e ME_CONFIG_MONGODB_ADMINPASSWORD=password \
-e ME_CONFIG_MONGODB_SERVER=mongodb \
--network mongo-network \
--name mongo-express \
mongo-express

docker logs <container-id>
```

Docker compose file

```
version: '3'
services:
        mongodb:
                image: mongo:x
                ports:
                        - 27017:27017
```

```
                environment:
                        -
MONGO_INITDB_ROOT_USERNAME=admin
                        -
MONGO_INITDB_ROOT_PASSWORD=password
        mongo-express:
                image: mongo-express:x
                ports:
                        - 8081:8081
                environment:
                        -
ME_CONFIG_MONGODB_ADMINUSERNAME=admin
                        -
ME_CONFIG_MONGODB_ADMINPASSWORD=password
                        -
ME_CONFIG_MONGODB_SERVER=mongodb
                depends_on:
                        mongodb
```

```
docker compose -f xyz.yml up -d
docker compose -f xyz.yml down
```

Note:-

- Docker compose will create a shared network for the services mentioned in the compose file
- use depends-on attribute if some service is dependent on other service till that starts
- Always setup the volumes if you want to persist the data after removing the container. But data will still be there if we use stop or re-start the containers

```
docker compose -f xyz.yml stop
docker compose -f xyz.yml start
```

Docker compose variables

```
version: '3'
services:
        xuz:
                image: alphadecodex112/test:1.0
                ports:
                        - 9591:9590
                environment:
                        - MYSQL_HOST=root
                        - MONGO_INITDB_ROOT_PASSWORD=
${MONGO_ADMIN_PASS}
        mongo-express:
                image: mongo-express:x
                ports:
                        - 8081:8081
                environment:
                        -
ME_CONFIG_MONGODB_ADMINUSERNAME=${MONGO_ADMIN_USER}
                        -
ME_CONFIG_MONGODB_ADMINPASSWORD=${MONGO_ADMIN_PASS}
                        -
ME_CONFIG_MONGODB_SERVER=mongodb
                depends_on:
                        mongodb
```

Docker compose Repository

```
docker build -t alphadecodex/my-app:1.0 .
docker images
docker login
docker push alphadecodex/my-app:1.0
```