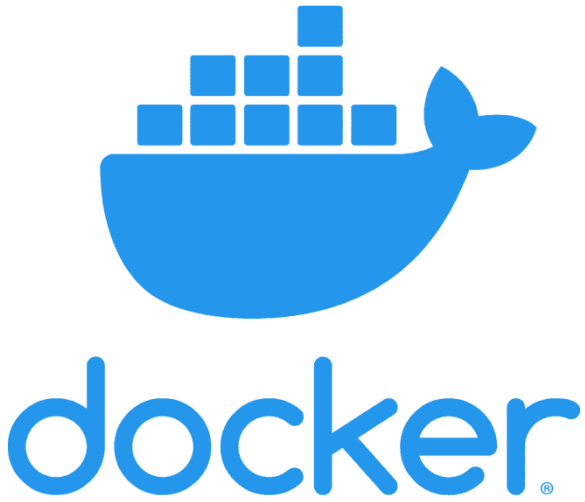


PROF. P. M. JADAV

ASSOCIATE PROFESSOR

COMPUTER ENGINEERING DEPARTMENT

FACULTY OF TECHNOLOGY

DHARMSINH DESAI UNIVERSITY, NADIAD

Content

Container Introduction

- A container is a standard unit of software that packages up
  - code and
  - all its dependencies
- so the application runs quickly and reliably among different computing environment.

Docker Introduction

- Docker is a Linux-based, open-source container management service
- “Develop once and run anywhere”
- Develop applications, ship them into container and deploy it anywhere
- Developed by Solomon Hykes
- First released in March 2013
- Written in Go language

Components of a Docker Architecture

Docker Image

- A Docker image is a
  - lightweight,
  - standalone,
  - executable package of software
- It includes everything needed to run an application:
  - code, runtime, system tools, system libraries and settings

Docker Image vs. Container

- Docker images become containers when they run on Docker Engine
- Available for both Linux and Windows (Docker Desktop), containerized software will always run the same, regardless of the infrastructure
- Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging

Docker Image vs. Container

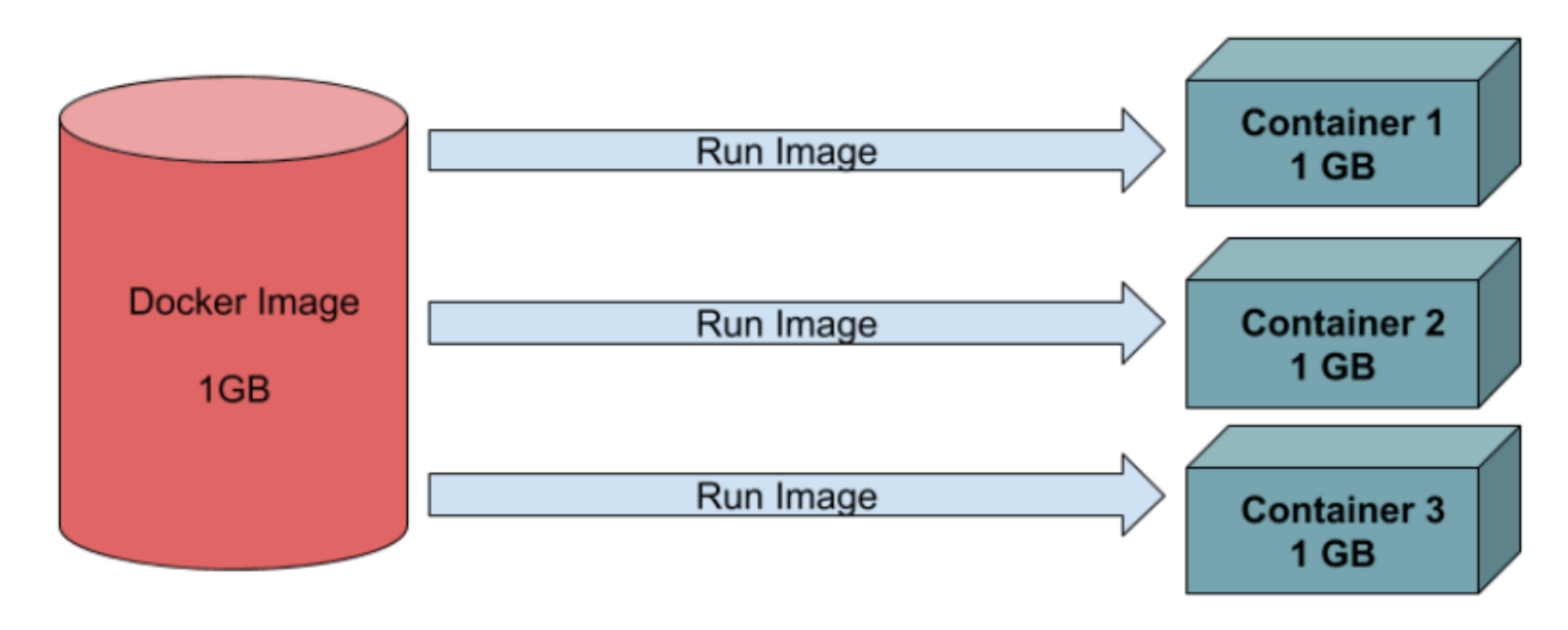
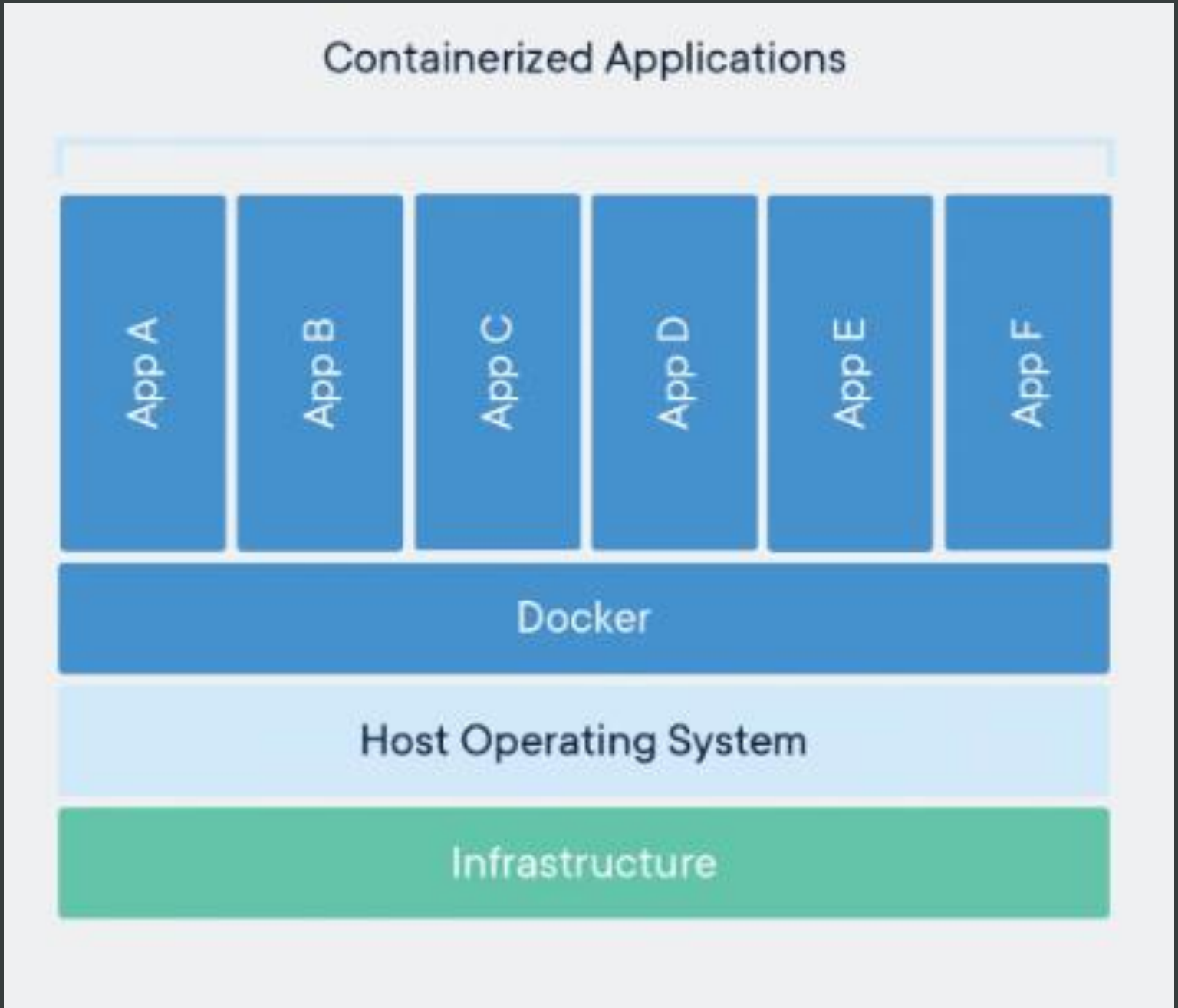


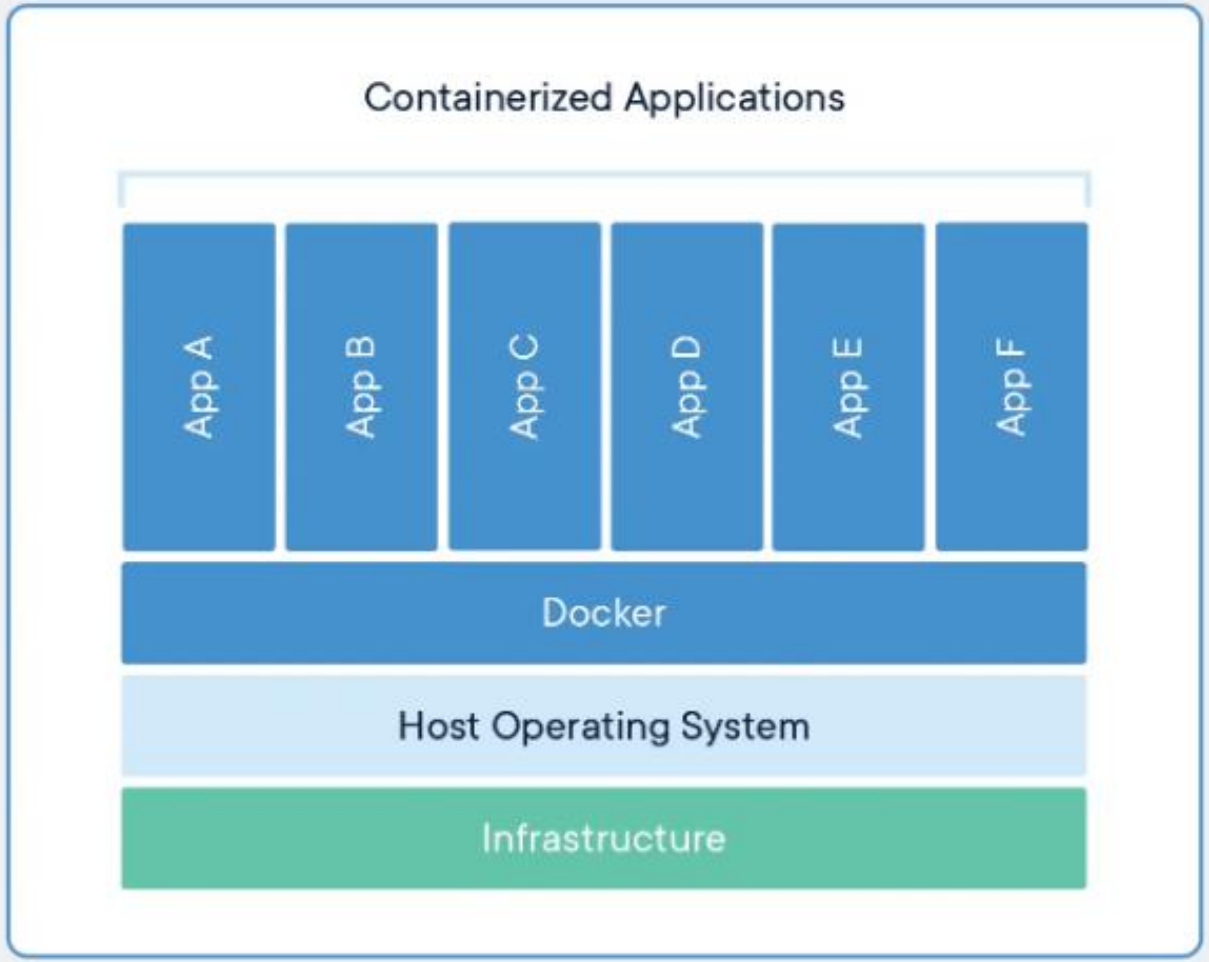
Image Source: https://davetang.github.io/reproducible_bioinformatics/docker.html

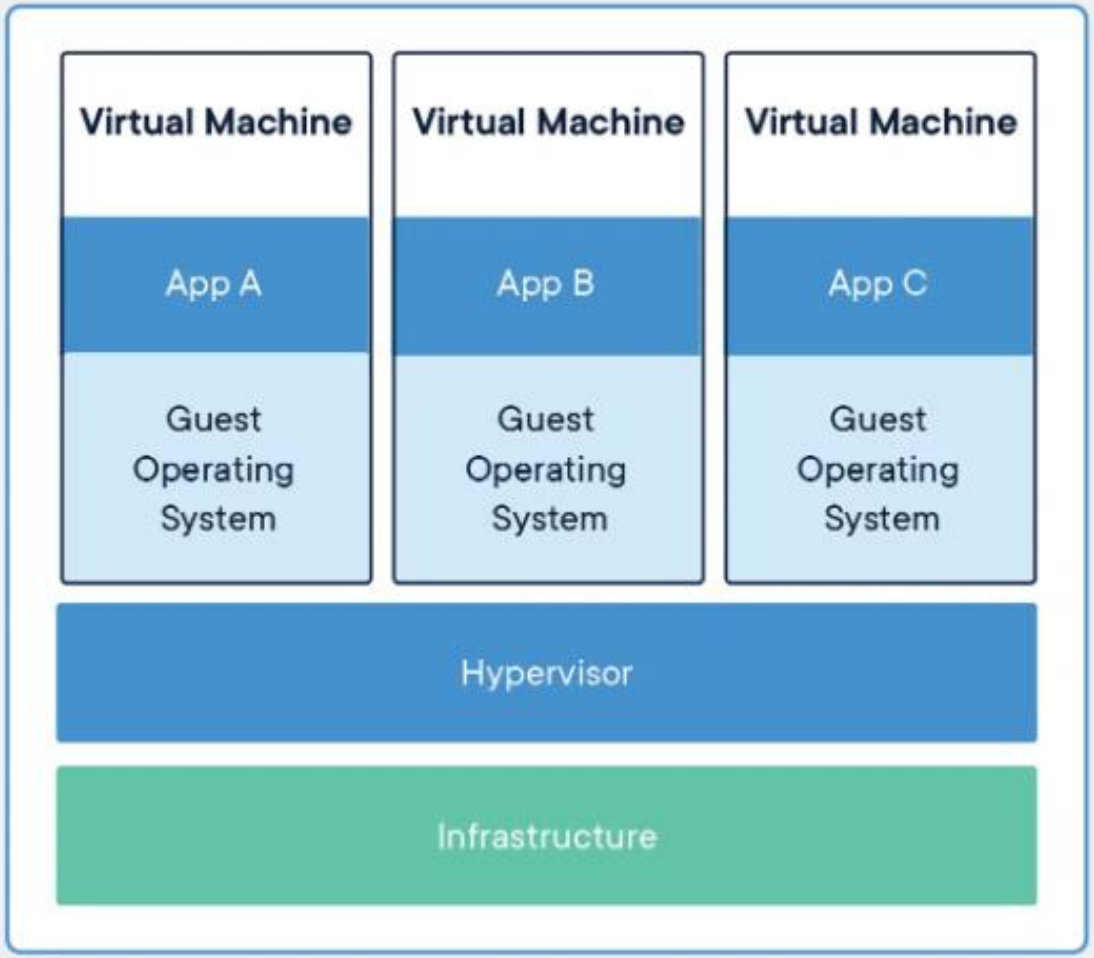

Containerized Applications
App A
App B
App C
App D
App E
App F
Docker
Host Operating System
Infrastructure

Virtual Machine

- A Virtual Machine (VM) is a compute resource that uses s/w instead of a physical computer to run programs and deploy apps.
- One or more virtual "guest" machines run on a physical "host" machine.
- Each virtual machine runs its own operating system and functions separately from the other VMs, even when they are all running on the same host.

Container vs. Virtual Machine





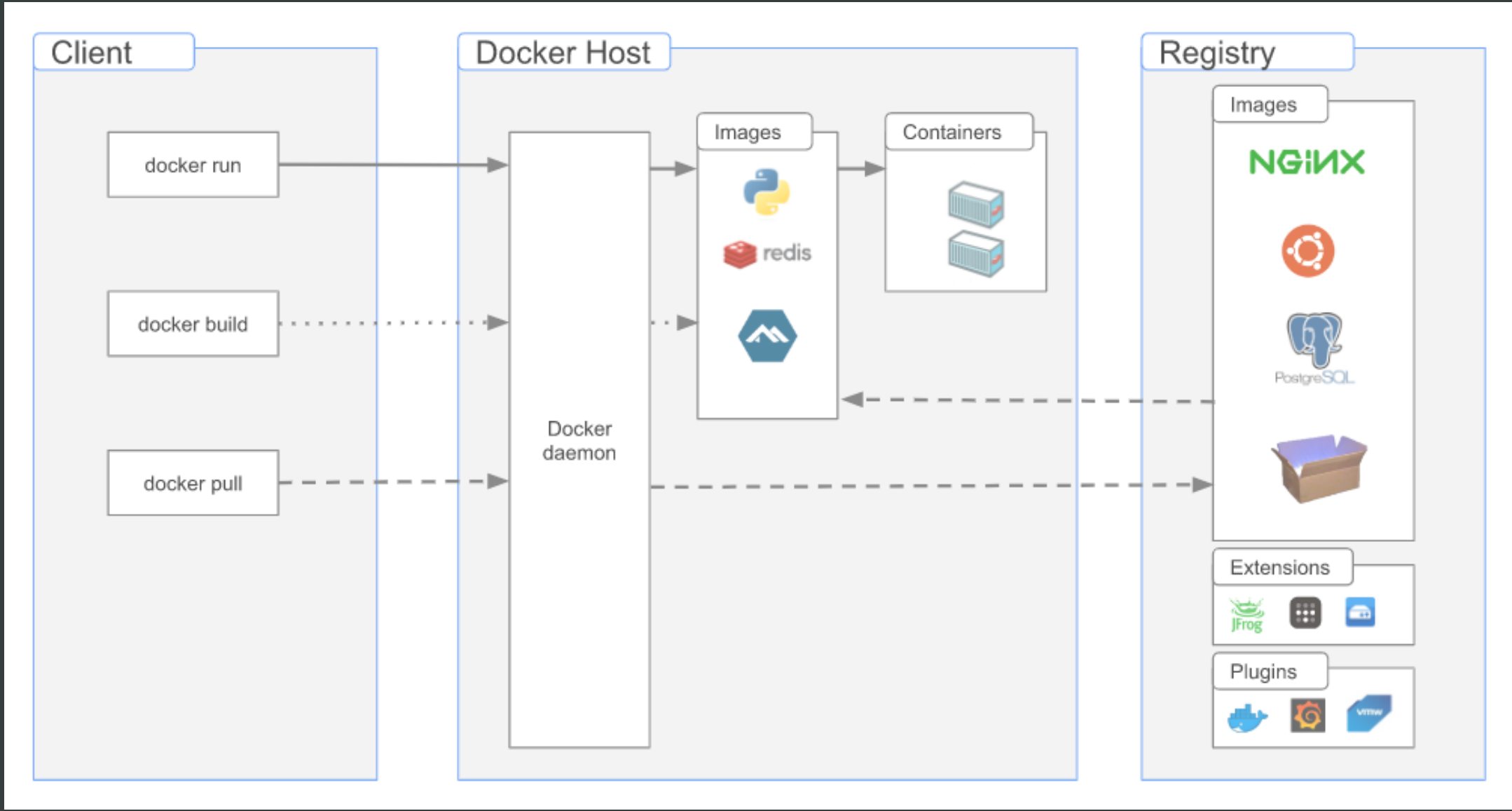
Container Benefits

- Multiple containers can run on the same machine
- They share the OS kernel with other containers
- Each runs as isolated processes in user space
- Containers take up less space (in MBs) than VMs (in GBs)
- It can handle more applications and require fewer VMs and Operating systems.

Virtual Machines Characteristics

- VMs are an abstraction of physical hardware turning one server into many servers.
- The hypervisor allows multiple VMs to run on a single machine
- Each VM includes a full copy of an OS, the application, necessary binaries and libraries (taking more space, in GBs)
- VMs can also be slow to boot.

Docker Architecture



The Docker Daemon

- The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as
  - images, containers, networks, and volumes.
- A daemon can also communicate with other daemons to manage Docker services.

The Docker Client

- The Docker client (docker) is the primary way that many Docker users interact with Docker
- Using commands such as docker run, the client sends these commands to dockerd, which carries them out.
- The docker command uses the Docker API.
- The Docker client can communicate with more than one daemon.

The Docker Desktop

- Docker Desktop is an easy-to-install application for Mac, Windows or Linux environment
- It enables you to build and share containerized applications and microservices
- Docker Desktop includes the Docker daemon (dockerd), the Docker client (docker), Docker Compose, Docker Content Trust, Kubernetes, and Credential Helper.

The Docker Registries

- It stores Docker images.
- Docker Hub is a public registry that anyone can use
- Docker is configured to look for images on Docker Hub by default
- You can even run your own private registry.
- When you use the docker pull or docker run commands, the required images are pulled from your configured registry.
- When you use the docker push command, your image is pushed to your configured registry.

Images

- An image is a read-only template with instructions for creating a Docker container
- Often, an image is based on another image, with some additional customization.
- For example, you may build an image which is based on the ubuntu image, but installs the Node.js server and your application, as well as the configuration details needed to make your application run.

Images

- You might create your own images
- You might only use those created by others and published in a registry
- To build your own image, you create a Dockerfile with a simple syntax for defining the steps needed to create the image and run it.
- Each instruction in a Dockerfile creates a layer in the image
- When you change the Dockerfile and rebuild the image, only those layers which have changed are rebuilt
- This is part of what makes images so lightweight, small, and fast, when compared to other virtualization technologies

Containers

- A container is a runnable instance of an image
- You can create, start, stop, move, or delete a container using the Docker API or CLI
- You can
  - connect a container to one or more networks,
  - attach storage to it, or
  - create a new image based on its current state

Containers

- Container is relatively well isolated from other containers and its host machine
- You can control how isolated a container's network, storage, or other underlying subsystems are from other containers or from the host machine
- A container is defined by its image as well as any configuration options you provide to it when you create or start it
- When a container is removed, any changes to its state that are not stored in persistent storage disappear.

Installing Docker Desktop on Windows 10/11 (pre-requisite)

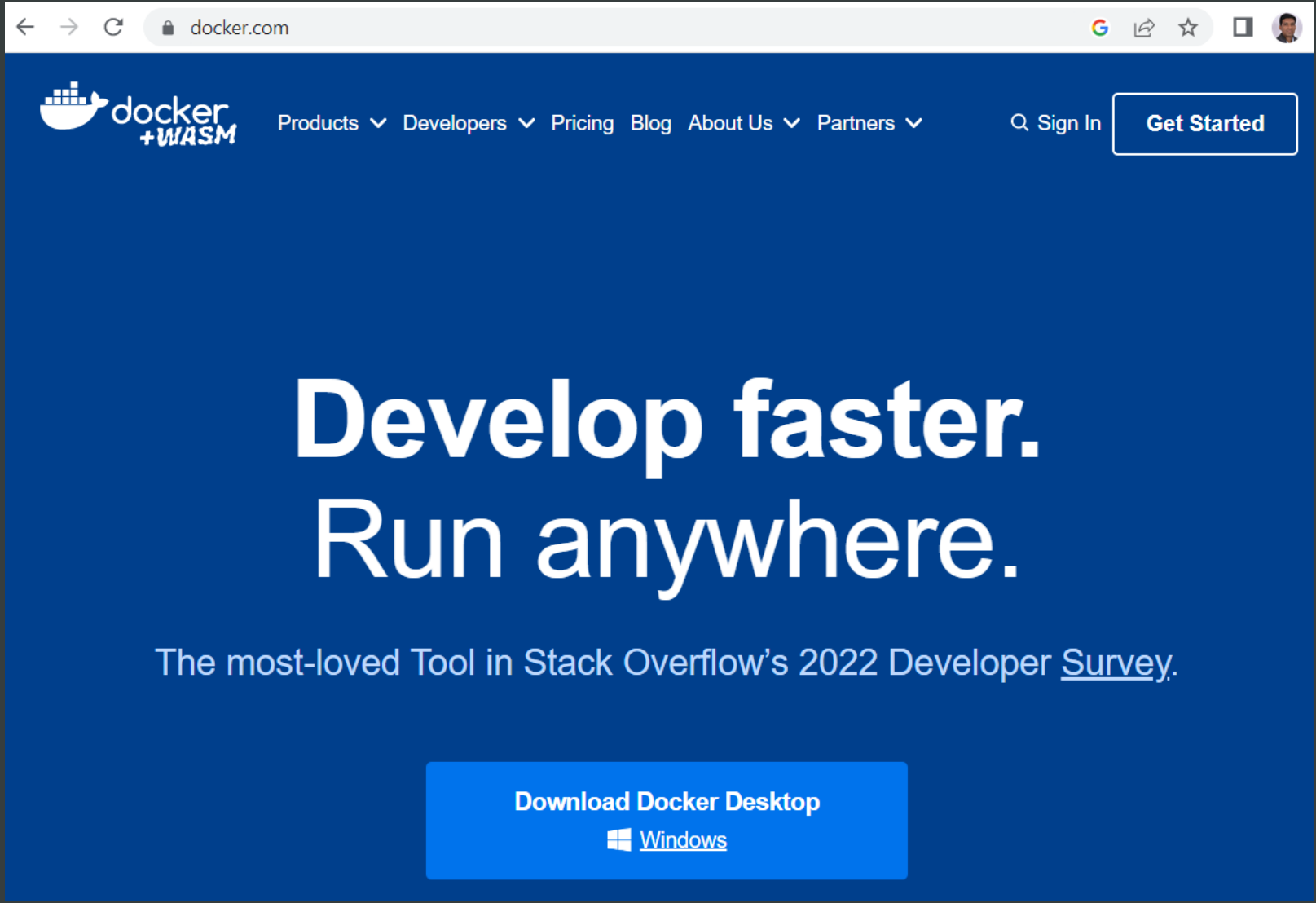
- Windows Subsystem for Linux (WSL) is required before installing Docker Desktop on Windows 10/11
- Open PowerShell or Windows Command Prompt in administrator mode by right-clicking and selecting "Run as administrator", enter the

  wsl --install

- command, then restart your machine
- If you're running an older build, or just prefer not to use the install command and would like step-by-step directions, see

  https://learn.microsoft.com/en-us/windows/wsl/install-manual

Installing Docker Desktop (docker.com)



Installing Docker Desktop (docker.com)

```
C:\Users\PMJ\Dropbox\subject\WSD\Docker>docker version
Client:
 Cloud integration: v1.0.31
 Version:           20.10.23
 API version:       1.41
 Go version:        go1.18.10
 Git commit:        7155243
 Built:             Thu Jan 19 17:43:10 2023
 OS/Arch:           windows/amd64
 Context:           default
 Experimental:      true

Server: Docker Desktop 4.17.0 (99724)
 Engine:
  Version:          20.10.23
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.18.10
  Git commit:       6051f14
  Built:            Thu Jan 19 17:32:04 2023
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.18
  GitCommit:        2456e983eb9e37e47538f59ea18f2043c9a73640
 runc:
  Version:          1.1.4
  GitCommit:        v1.1.4-0-g5fd4c4d
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

Dockerfile (a text file without any extension)

```
FROM ubuntu
```

Base Image (ubuntu)

```
MAINTAINER pmj.ce@ddu.ac.in

RUN apt-get update

RUN apt-get install –y nginx

CMD ["echo", "Image created"]
```

Creating an Image using Dockerfile

- Create a Dockerfile in the current directory
- Run the following command:

```
$ docker build .
```

Creating an Image using Dockerfile

```
[+] Building 116.7s (7/7) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.2s
 => => transferring dockerfile: 193B                                                                    0.0s
 => [internal] load .dockerignore                                                                       0.1s
 => => transferring context: 2B                                                                         0.0s
 => [internal] load metadata for docker.io/library/ubuntu:latest                                        3.8s
 => [1/3] FROM docker.io/library/ubuntu@sha256:2adf22367284330af9f832ffefb717c78239f6251d9d0f58de50b86229ed1427   4.7s
 => => resolve docker.io/library/ubuntu@sha256:2adf22367284330af9f832ffefb717c78239f6251d9d0f58de50b86229ed1427   0.0s
 => => sha256:74f2314a03de34a0a2d552b805411fc9553a02ea71c1291b815b2f645f565683 2.30kB / 2.30kB          0.0s
 => => sha256:76769433fd8a87dd77a6ce33db12156b1ea8dad3da3a95e7c9c36a47ec17b24c 29.53MB / 29.53MB        3.2s4
 => => sha256:2adf22367284330af9f832ffefb717c78239f6251d9d0f58de50b86229ed1427 1.13kB / 1.13kB          0.0se
 => => sha256:b2175cd4cfdd5cdb1740b0e6ec6bbb4ea4892801c0ad5101a81f694152b6c559 424B / 424B              0.0s
 => => extracting sha256:76769433fd8a87dd77a6ce33db12156b1ea8dad3da3a95e7c9c36a47ec17b24c               1.2s
 => [2/3] RUN apt-get update                                                                           35.7s
 => [3/3] RUN apt-get install -y nginx                                                                 71.6s
 => exporting to image                                                                                  0.5s
 => => exporting layers                                                                                 0.5s
 => => writing image sha256:fcecdc464af6e4d00c5693de08eb3a7723401c1ab12d13522d6c7903c152fcbd            0.0s
```

Viewing and Removing a Docker Image

```
C:\Users\PMJ\Dropbox\subject\WSD\Docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
<none>       <none>    fcecdc464af6   4 minutes ago    176MB

C:\Users\PMJ\Dropbox\subject\WSD\Docker>docker rmi fcecdc464af6
Deleted: sha256:fcecdc464af6e4d00c5693de08eb3a7723401c1ab12d13522d6c7903c152fcbd

C:\Users\PMJ\Dropbox\subject\WSD\Docker>docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

Assigning a Name and Tag to an Image

```
C:\Users\PMJ\Dropbox\subject\WSD\Docker>docker build -t myimage:0.1 .
[+] Building 3.1s (7/7) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 32B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/ubuntu:latest
 => [1/3] FROM docker.io/library/ubuntu@sha256:2adf22367284330af9f832ffefb717c78239f6251d9d0f58de50b86229ed1427
 => CACHED [2/3] RUN apt-get update
 => CACHED [3/3] RUN apt-get install -y nginx
 => exporting to image
 => => exporting layers
 => => writing image sha256:fcecdc464af6e4d00c5693de08eb3a7723401c1ab12d13522d6c7903c152fcbd
 => => naming to docker.io/library/myimage:0.1

C:\Users\PMJ\Dropbox\subject\WSD\Docker>docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
myimage      0.1       fcecdc464af6   12 minutes ago   176MB
```

Inside Docker Container

```
C:\Users\PMJ\Dropbox\subject\WSD\Docker>docker run -it myimage:0.1 /bin/bash
root@23e58beaed4c:/# pwd
/
root@23e58beaed4c:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run
root@23e58beaed4c:/# exit
exit
```

Listing Containers

```
C:\Users\PMJ\Dropbox\subject\WSD\Docker>docker ps -a
CONTAINER ID   IMAGE          COMMAND   CREATED          STATUS    PORTS     NAMES
8e0ebb79d53b   myimage:0.1    "-d"      15 minutes ago   Created           cool_poincare
```

Containerizing Node.js App (index.js)

```
var express = require('express');
var path = require('path');
var app = express();
var bodyParser = require("body-parser");

app.use(bodyParser.urlencoded({ extended: false }));

app.get('/', function (req, res) {
    var options = {
        root: path.join(__dirname)
    };

    var fileName = 'index.html';
    res.sendFile(fileName, options, function (err) {
        if (err) {
            throw err;
        } else {
            console.log('Sent:', fileName);
        }
    });
});
```

Containerizing Node.js App (index.js)

```
app.post('/submit', function (req, res) {
    var name = req.body.firstName + ' ' + req.body.lastName;

    res.send(name + ' Submitted Successfully!');
});

var server = app.listen(8001, function () {
});

console.log('Node server is running on port 8001..');
```

Containerizing Node.js App (index.html)

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Node.js Demo Application</title>
</head>
<body>
    <form action="/submit" method="post">
        First Name: <input name="firstName" type="text" /> <br/>
        Last Name: <input name="lastName" type="text" /> <br/>
        <input type="submit" />
    </form>
</body>
</html>
```

Containerizing Node.js App (.dockerignore)

```
.git
.gitignore
node_modules
npm-debug.log
Dockerfile*
docker-compose*
README.md
LICENSE
.vscode
```

Docker Registry (hub.docker.com)

hub.docker.com/_/node

node

Explore Pricing Sign In Register

Explore > Official Images > node

node DOCKER OFFICIAL IMAGE · 1B+ · 10K+

Node.js is a JavaScript-based platform for server-side and networking applications.

```
docker pull node
```

Node Images With Different Versions

- 19-alpine3.16 , 19.7-alpine3.16 , 19.7.0-alpine3.16 , alpine3.16 , current-alpine3.16
- 19-alpine , 19-alpine3.17 , 19.7-alpine , 19.7-alpine3.17 , 19.7.0-alpine , 19.7.0-alpine3.17 , alpine , alpine3.17 , current-alpine , current-alpine3.17
- 19 , 19-bullseye , 19.7 , 19.7-bullseye , 19.7.0 , 19.7.0-bullseye , bullseye , current , current-bullseye , latest
- 19-bullseye-slim , 19-slim , 19.7-bullseye-slim , 19.7-slim , 19.7.0-bullseye-slim , 19.7.0-slim , bullseye-slim , current-bullseye-slim , current-slim , slim
- 19-buster , 19.7-buster , 19.7.0-buster , buster , current-buster
- 19-buster-slim , 19.7-buster-slim , 19.7.0-buster-slim , buster-slim , current-buster-slim
- 18-alpine3.16 , 18.14-alpine3.16 , 18.14.2-alpine3.16 , hydrogen-alpine3.16 , lts-alpine3.16
- 18-alpine , 18-alpine3.17 , 18.14-alpine , 18.14-alpine3.17 , 18.14.2-alpine , 18.14.2-alpine3.17 , hydrogen-alpine , hydrogen-alpine3.17 , lts-alpine , lts-alpine3.17
- 18 , 18-bullseye , 18.14 , 18.14-bullseye , 18.14.2 , 18.14.2-bullseye , hydrogen , hydrogen-bullseye , lts , lts-bullseye , lts-hydrogen
- 18-bullseye-slim , 18-slim , 18.14-bullseye-slim , 18.14-slim , 18.14.2-bullseye-slim , 18.14.2-slim , hydrogen-bullseye-slim , hydrogen-slim , lts-bullseye-slim , lts-slim
- 18-buster , 18.14-buster , 18.14.2-buster , hydrogen-buster , lts-buster
- 18-buster-slim , 18.14-buster-slim , 18.14.2-buster-slim , hydrogen-buster-slim , lts-buster-slim
- 16-alpine3.16 , 16.19-alpine3.16 , 16.19.1-alpine3.16 , gallium-alpine3.16

Node Images With Different Versions

- `16-alpine`, `16-alpine3.17`, `16.19-alpine`, `16.19-alpine3.17`, `16.19.1-alpine`, `16.19.1-alpine3.17`, `gallium-alpine`, `gallium-alpine3.17`
- `16-bullseye`, `16.19-bullseye`, `16.19.1-bullseye`, `gallium-bullseye`
- `16-bullseye-slim`, `16.19-bullseye-slim`, `16.19.1-bullseye-slim`, `gallium-bullseye-slim`
- `16`, `16-buster`, `16.19`, `16.19-buster`, `16.19.1`, `16.19.1-buster`, `gallium`, `gallium-buster`
- `16-buster-slim`, `16-slim`, `16.19-buster-slim`, `16.19-slim`, `16.19.1-buster-slim`, `16.19.1-slim`, `gallium-buster-slim`, `gallium-slim`
- `14-alpine3.16`, `14.21-alpine3.16`, `14.21.3-alpine3.16`, `fermium-alpine3.16`
- `14-alpine`, `14-alpine3.17`, `14.21-alpine`, `14.21-alpine3.17`, `14.21.3-alpine`, `14.21.3-alpine3.17`, `fermium-alpine`, `fermium-alpine3.17`
- `14-bullseye`, `14.21-bullseye`, `14.21.3-bullseye`, `fermium-bullseye`
- `14-bullseye-slim`, `14.21-bullseye-slim`, `14.21.3-bullseye-slim`, `fermium-bullseye-slim`
- `14`, `14-buster`, `14.21`, `14.21-buster`, `14.21.3`, `14.21.3-buster`, `fermium`, `fermium-buster`
- `14-buster-slim`, `14-slim`, `14.21-buster-slim`, `14.21-slim`, `14.21.3-buster-slim`, `14.21.3-slim`, `fermium-buster-slim`, `fermium-slim`

Containerizing Node.js App (.Dockerfile)

```
FROM node:19-alpine
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
RUN chown -R node /usr/src/app/node_modules
COPY . .
EXPOSE 8001
CMD ["npm", "start"]
```

Creating a Docker Image

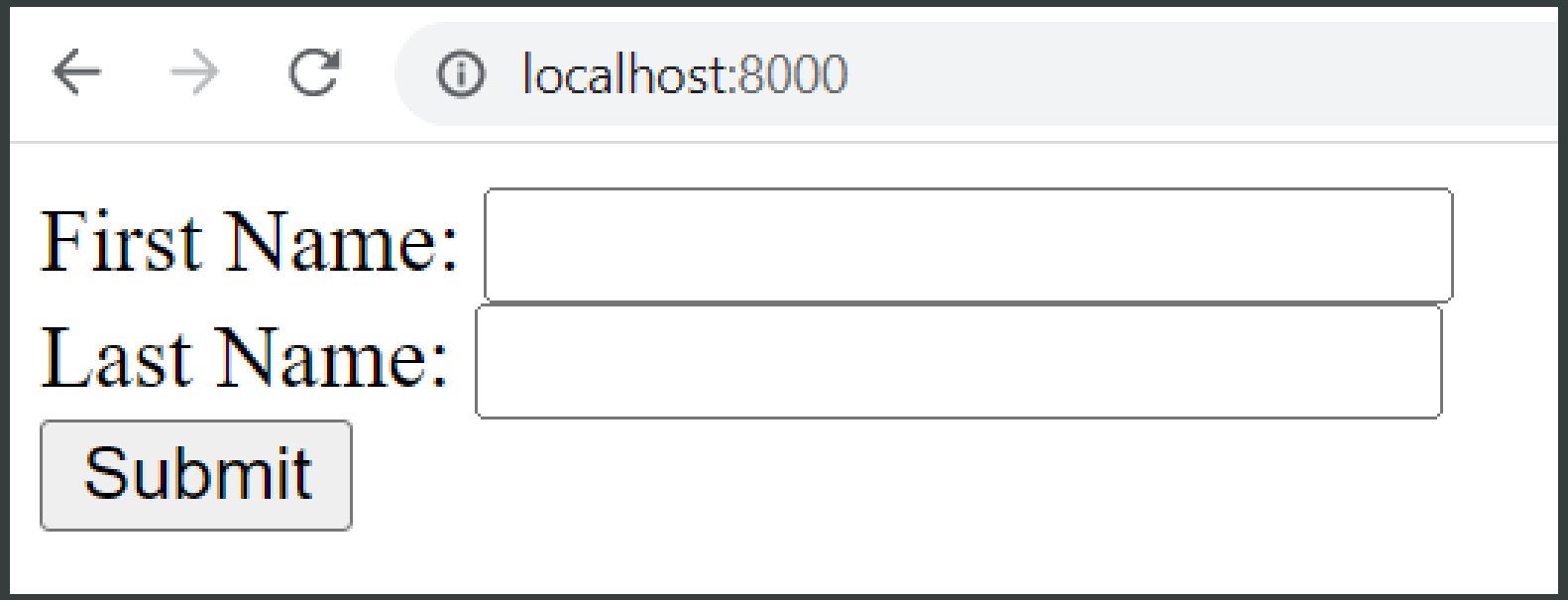
```
C:\Users\PMJ\Dropbox\subject\WSD\Docker\demo>docker build -t pmj/node-app .
[+] Building 7.2s (11/11) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 212B
 => [internal] load .dockerignore
 => => transferring context: 34B
 => [internal] load metadata for docker.io/library/node:19-alpine
 => [1/6] FROM docker.io/library/node:19-alpine@sha256:155e324802ebfdd3f508340dcb0cd4a7510f8594802a2e53150f171ae8aa2462
 => [internal] load build context
 => => transferring context: 193B
 => CACHED [2/6] WORKDIR /usr/src/app
 => CACHED [3/6] COPY package*.json ./
 => CACHED [4/6] RUN npm install
 => [5/6] RUN chown -R node /usr/src/app/node_modules
 => [6/6] COPY . .
 => exporting to image
 => => exporting layers
 => => writing image sha256:7d90190fcc89519602aad802b3536e62a0e956ac6bc0a1869b8c416d424b69fb
 => => naming to docker.io/pmj/node-app

C:\Users\PMJ\Dropbox\subject\WSD\Docker\demo>docker images
REPOSITORY     TAG       IMAGE ID       CREATED          SIZE
pmj/node-app   latest    7d90190fcc89   12 seconds ago   186MB
```

Creating Container from the Docker Image

```
C:\Users\PMJ\Dropbox\subject\WSD\Docker\demo>docker run -d -p 8000:8001 --name test pmj/node-app
26cf546547e7bd3a37372a8943db0440725433cf9d85648e64ab4434cef4e8e5
```

```
C:\Users\PMJ\Dropbox\subject\WSD\Docker\demo>docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                    NAMES
26cf546547e7   pmj/node-app   "docker-entrypoint.s…"   6 seconds ago   Up 4 seconds   0.0.0.0:8000->8001/tcp   test

C:\Users\PMJ\Dropbox\subject\WSD\Docker\demo>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                    NAMES
26cf546547e7   pmj/node-app   "docker-entrypoint.s…"   13 seconds ago   Up 12 seconds   0.0.0.0:8000->8001/tcp   test
```

Running the Node.js App from the Host

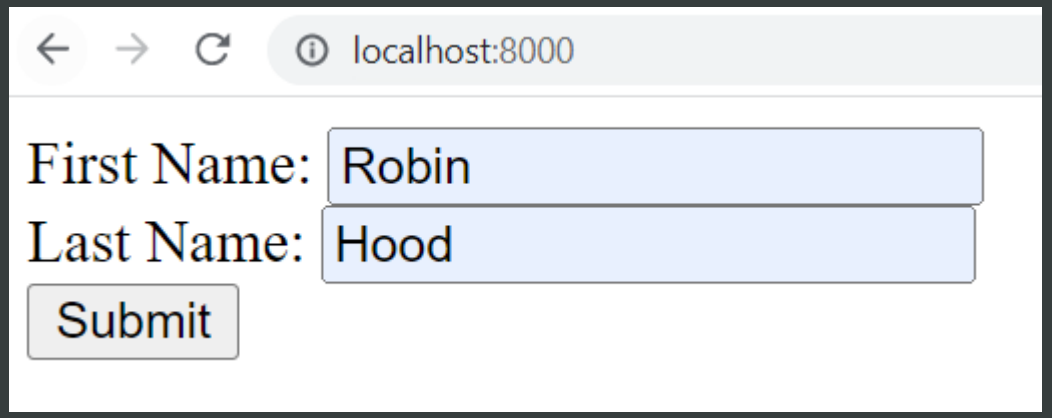


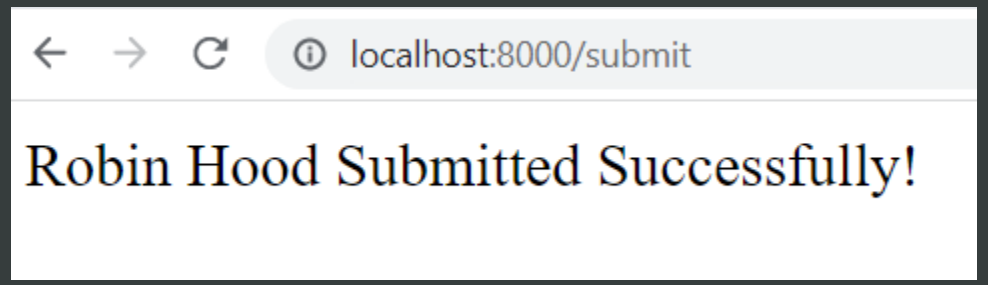
Viewing logs of a Docker Container

This command is useful when there is an error in the container and the container exits.

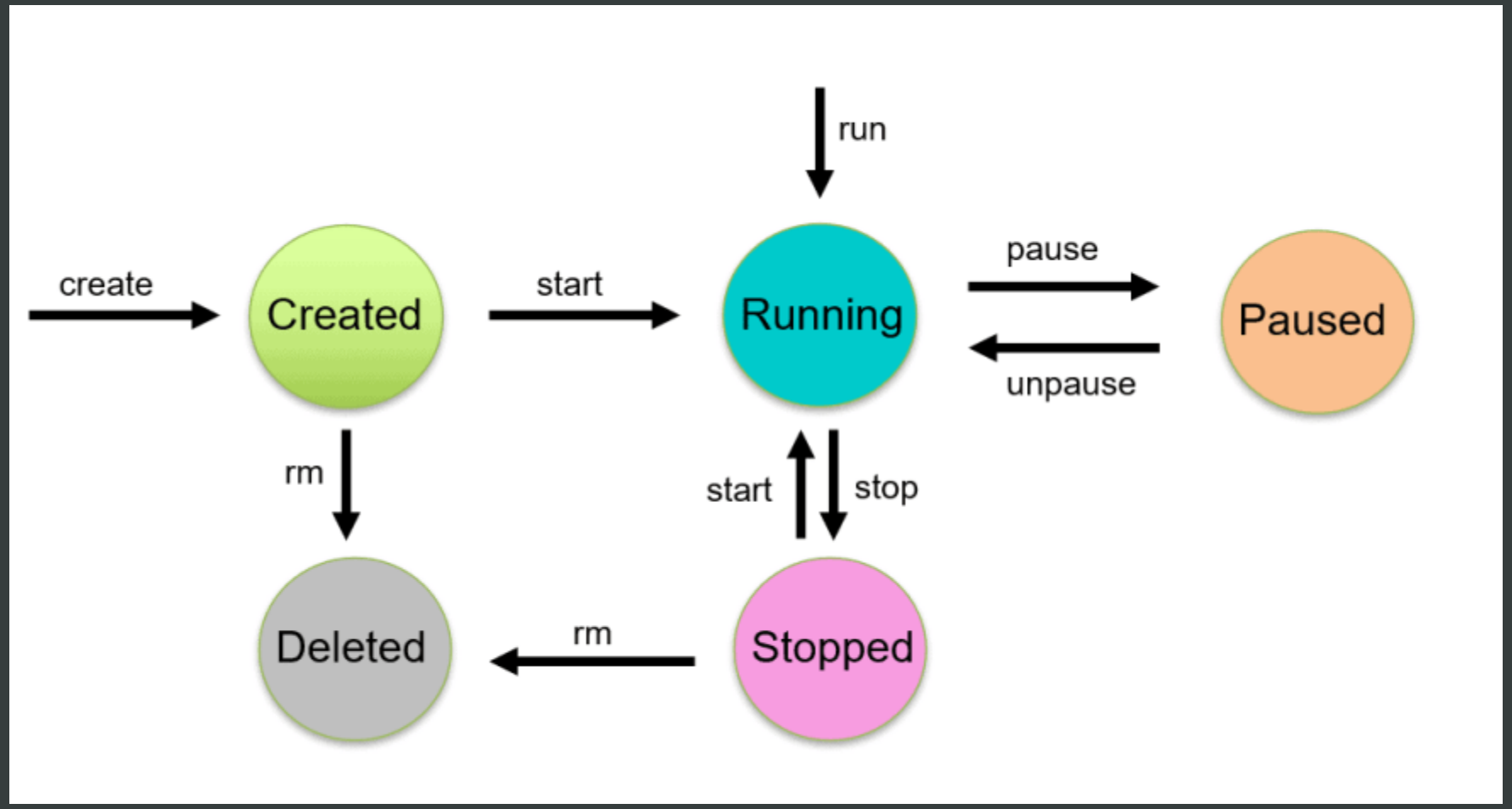
```
C:\Users\PMJ\Dropbox\subject\WSD\Docker\demo>docker logs test

> demo@1.0.0 start
> node index.js

Node server is running on port 8001..
Sent: index.html
```

Running the Node.js App from the Host





Docker Container Lifecycle



Source: https://dev.to/docker/docker-architecture-life-cycle-of-docker-containers-and-data-management-1a9c

Docker Commands (Start, Stop, Restart, Pause, Unpause, and Kill)

```
C:\Users\PMJ\Dropbox\subject\WSD\Docker\demo>docker pause test
test

C:\Users\PMJ\Dropbox\subject\WSD\Docker\demo>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                    PORTS                     NAMES
26cf546547e7   pmj/node-app   "docker-entrypoint.s…"   16 minutes ago   Up 16 minutes (Paused)    0.0.0.0:8000->8001/tcp    test

C:\Users\PMJ\Dropbox\subject\WSD\Docker\demo>docker stop test
test

C:\Users\PMJ\Dropbox\subject\WSD\Docker\demo>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

C:\Users\PMJ\Dropbox\subject\WSD\Docker\demo>docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                     PORTS     NAMES
26cf546547e7   pmj/node-app   "docker-entrypoint.s…"   16 minutes ago   Exited (1) 9 seconds ago             test

C:\Users\PMJ\Dropbox\subject\WSD\Docker\demo>docker start test
test

C:\Users\PMJ\Dropbox\subject\WSD\Docker\demo>docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS         PORTS                     NAMES
26cf546547e7   pmj/node-app   "docker-entrypoint.s…"   16 minutes ago   Up 3 seconds   0.0.0.0:8000->8001/tcp    test
```

Login to your Docker Hub Account

```
C:\Users\PMJ\Dropbox\subject\WSD\Docker\demo>docker login
Login with your Docker ID to push and pull images from Docker Hub.
o create one.
Username: username
Password:
Login Succeeded
```

Push Your Image to your Docker Hub Repository

```
C:\Users\PMJ\Dropbox\subject\WSD\Docker\demo>docker tag pmj/node-app username/node-app:v1.0

C:\Users\PMJ\Dropbox\subject\WSD\Docker\demo>docker push username/node-app:v1.0
The push refers to repository [docker.io/username/node-app]
591b258e3607: Pushed
857aa744ca8b: Pushed
0754f209159a: Pushed
6ff4307eb90a: Pushed
5ca1ae2b31b6: Pushed
12e09510884d: Pushed
1194a4b7d675: Pushed
25927219b4b7: Pushed
7cd52847ad77: Pushed
v1.0: digest: sha256:1368ac7b3012eebfac6e0dc69508733b98b55a39c0ba7f73ee9a0ef98007ad9c size: 2204
```

username is your docker username

Pull your Image from Docker Hub Repository

```
C:\Users\CEDDIT>docker pull username /node-app:v1.0
v1.0: Pulling from username /node-app
63b65145d645: Pull complete
d5bdfeed6dfa: Pull complete
75ec85813c14: Pull complete
a6f48326f540: Pull complete
93c93c493eef: Pull complete
f327f8e2fb6f: Pull complete
793e5531187a: Pull complete
aa2aabedb1ec: Pull complete
e95f74760bec: Pull complete
Digest: sha256:1368ac7b3012eebfac6e0dc69508733b98b55a39c0ba7f73ee9a0ef98007ad9c
Status: Downloaded newer image for username/node-app:v1.0
docker.io/username/node-app:v1.0
```

Running the downloaded Image

```
C:\Users\CEDDIT>docker run -d -p 8000:8001 --name demo pmjadav/node-app:v1.0
266ef1404fe4184f854643e6b3cf88aea29edc33958ed6bae0af2697cf95aa2c

C:\Users\CEDDIT>docker ps
CONTAINER ID   IMAGE                   COMMAND                  CREATED         STATUS         PORTS                    NAMES
266ef1404fe4   pmjadav/node-app:v1.0   "docker-entrypoint.s…"   6 seconds ago   Up 5 seconds   0.0.0.0:8000->8001/tcp   demo

C:\Users\CEDDIT>docker ps -a
CONTAINER ID   IMAGE                   COMMAND                  CREATED          STATUS          PORTS                    NAMES
266ef1404fe4   pmjadav/node-app:v1.0   "docker-entrypoint.s…"   15 seconds ago   Up 15 seconds   0.0.0.0:8000->8001/tcp   demo

C:\Users\CEDDIT>docker ps -a -q
266ef1404fe4
```

Execute a Command in a running Container

```
C:\Users\CEDDIT>docker exec -it demo ps aux
PID   USER     TIME  COMMAND
    1 root      0:00 npm start
   18 root      0:00 node index.js
   34 root      0:00 ps aux
```

```
C:\Users\CEDDIT>docker exec -it demo /bin/sh
/usr/src/app # ls
history.txt        index.html        index.js        node_modules      package-lock.json  package.json
/usr/src/app # exit
```

Docker pause, unpause, stop, start commands

```
C:\Users\CEDDIT>docker ps
CONTAINER ID   IMAGE                   COMMAND                  CREATED         STATUS         PORTS                     NAMES
266ef1404fe4   pmjadav/node-app:v1.0   "docker-entrypoint.s…"   9 minutes ago   Up 9 minutes   0.0.0.0:8000->8001/tcp    demo

C:\Users\CEDDIT>docker pause demo
demo

C:\Users\CEDDIT>docker ps
CONTAINER ID   IMAGE                   COMMAND                  CREATED         STATUS                   PORTS                     NAME
S
266ef1404fe4   pmjadav/node-app:v1.0   "docker-entrypoint.s…"   10 minutes ago   Up 10 minutes (Paused)   0.0.0.0:8000->8001/tcp    demo

C:\Users\CEDDIT>docker unpause demo
demo

C:\Users\CEDDIT>docker ps
CONTAINER ID   IMAGE                   COMMAND                  CREATED         STATUS          PORTS                     NAMES
266ef1404fe4   pmjadav/node-app:v1.0   "docker-entrypoint.s…"   10 minutes ago   Up 10 minutes   0.0.0.0:8000->8001/tcp    demo

C:\Users\CEDDIT>docker stop demo
demo

C:\Users\CEDDIT>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

C:\Users\CEDDIT>docker start demo
demo

C:\Users\CEDDIT>docker ps
CONTAINER ID   IMAGE                   COMMAND                  CREATED         STATUS         PORTS                     NAMES
266ef1404fe4   pmjadav/node-app:v1.0   "docker-entrypoint.s…"   10 minutes ago   Up 2 seconds   0.0.0.0:8000->8001/tcp    demo
```

Modify the Container

docker exec -it demo /bin/sh

```
var express = require('express');^M
var path = require('path');^M
var app = express();^M
var moment = require('moment');
var bodyParser = require("body-parser");^M
^M
app.use(bodyParser.urlencoded({ extended: false }));^M
^M
app.get('/', function (req, res) {^M
        var options = {^M
        root: path.join(__dirname)^M
    };^M
     ^M
    var fileName = 'index.html';^M
    res.sendFile(fileName, options, function (err) {^M
        if (err) {^M
            next(err);^M
        } else {^M
            console.log('Sent:', fileName);^M
        }^M
    });^M
});^M
^M
app.post('/submit', function (req, res) {^M
    var name = req.body.firstName + ' ' + req.body.lastName;^M
    ^M
    res.send(name + ' Submitted Successfully on ' + moment().format('Do MMMM YYYY'));^M
});^M
^M
```

Modify the Container

```
/usr/src/app # npm i moment

added 1 package, and audited 95 packages in 2s

10 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
/usr/src/app # cat package.json
{
  "dependencies": {
    "express": "^4.18.2",
    "moment": "^2.29.4",
    "nodemon": "^2.0.21",
    "path": "^0.12.7"
  },
```

Commit changes to an Image

```
C:\Users\CEDDIT>docker ps
CONTAINER ID   IMAGE                   COMMAND                  CREATED          STATUS          PORTS                    NAMES
266ef1404fe4   pmjadav/node-app:v1.0   "docker-entrypoint.s…"   37 minutes ago   Up 26 minutes   0.0.0.0:8000->8001/tcp   demo

C:\Users\CEDDIT>docker commit 266ef1404fe4 pmjadav/node-app:v1.1
sha256:874d5605562d5e829f4e9f55d8fc4155cf47096e7324715883de8f37a8b9c9a6

C:\Users\CEDDIT>docker images
REPOSITORY         TAG       IMAGE ID       CREATED          SIZE
pmjadav/node-app   v1.1      874d5605562d   11 seconds ago   186MB
pmjadav/node-app   v1.0      7d90190fcc89   11 hours ago     186MB
```

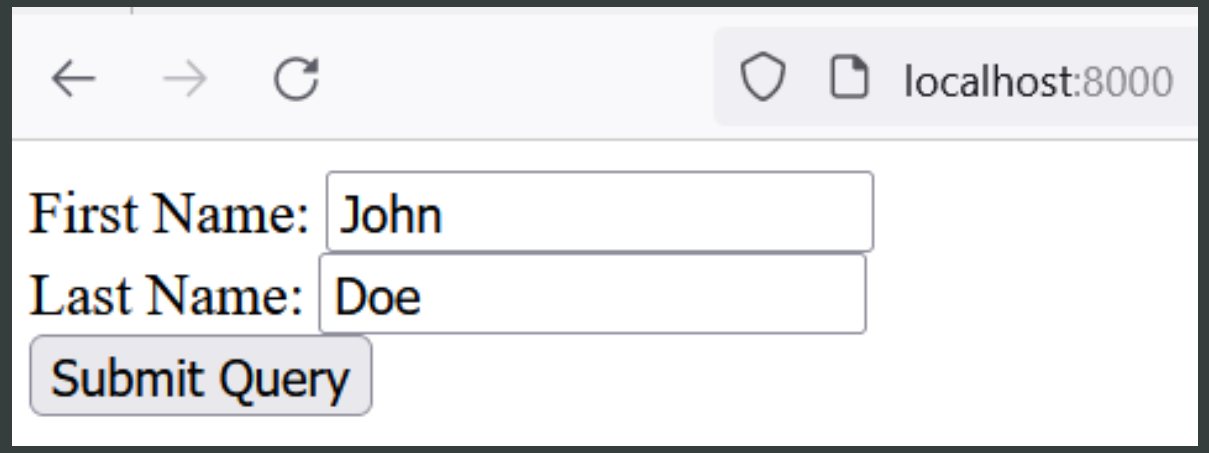
Stopping and Removing the Container

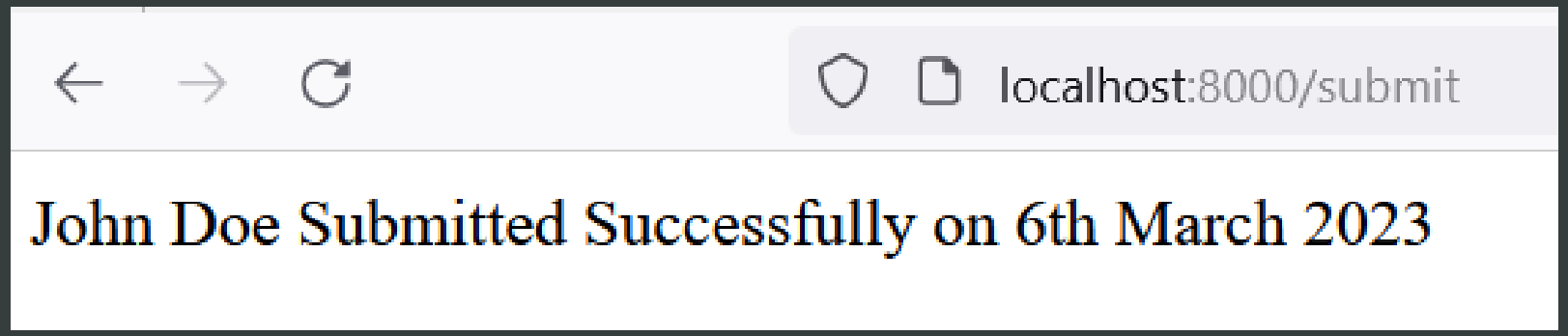
```
C:\Users\CEDDIT>docker stop demo
demo

C:\Users\CEDDIT>docker ps -a
CONTAINER ID   IMAGE                   COMMAND                  CREATED          STATUS                      PORTS     NAMES
266ef1404fe4   pmjadav/node-app:v1.0   "docker-entrypoint.s…"   41 minutes ago   Exited (1) 3 seconds ago              demo

C:\Users\CEDDIT>docker rm demo
demo

C:\Users\CEDDIT>docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

Running a Container with Modified Image

```
C:\Users\CEDDIT>docker run -d -p 8000:8001 --name demo1 pmjadav/node-app:v1.1
f0ecfd6e56adde00011ba0fc409853fc046f9cb7597338bc22b0d78b4bb402a3
```

Running a Container with Modified Image





Pushing an Updated Image to Docker Registry

```
C:\Users\CEDDIT>docker push pmjadav/node-app:v1.1
The push refers to repository [docker.io/pmjadav/node-app]
9eb26d233340: Pushed
591b258e3607: Layer already exists
857aa744ca8b: Layer already exists
0754f209159a: Layer already exists
6ff4307eb90a: Layer already exists
5ca1ae2b31b6: Layer already exists
12e09510884d: Layer already exists
1194a4b7d675: Layer already exists
25927219b4b7: Layer already exists
7cd52847ad77: Layer already exists
v1.1: digest: sha256:49c6867997f951f37ea5a4c86ef85e740fc0e25b3a662b4f5f4ee69597722c80 size: 2415
```

Pushing an Updated Image to Docker Registry

pmjadav / node-app

Description

This repository does not have a description

Last pushed: 5 minutes ago

Tags

IMAGE INSIGHTS INACTIVE
Activate

This repository contains 2 tag(s).

Tag	OS	Type	Pulled	Pushed
v1.1		Image	---	5 minutes ago
v1.0		Image	4 hours ago	12 hours ago

See all

Go to Advanced Image Management

Saving username in a file (username.txt)

```
/usr/src/app # ls
index.html          index.js            node_modules        package-lock.json   package.json
/usr/src/app # mkdir user
/usr/src/app # touch user/username.txt
```

```
var fs = require('fs');
```

```
app.post('/submit', function (req, res) {^M
    var name = req.body.firstName + ' ' + req.body.lastName + '\n';^M
    ^M
    fs.appendFile('./user/username.txt', name, function(err) {
        if (err) throw err;
        console.log(name + ' saved in the file.');
    });
    res.send(name + ' Submitted Successfully on ' + moment().format('Do MMMM YYYY'));
});^M
```

username.txt file is lost every time we run the container again. The data do not persist.

Committed the modified image as pmjadav/node-app:v1.2

Saving username in a file (username.txt)

```
C:\Users\PMJ>docker commit demo11 pmjadav/node-app:v1.2
sha256:5789eccac8ad22858775b4f5dd15c8fad7a0a7c7700094618dc4d651e34c66e4

C:\Users\PMJ>docker images
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
pmjadav/node-app    v1.2      5789eccac8ad   6 seconds ago    191MB
pmjadav/node-app    v1.1      51cda4a53c19   47 hours ago     191MB

C:\Users\PMJ>docker ps -a
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS          PORTS                     NAMES
ec3a4b7d93ee   pmjadav/node-app:v1.1    "docker-entrypoint.s…"   6 minutes ago    Up 6 minutes    0.0.0.0:8000->8001/tcp    demo11

C:\Users\PMJ>docker stop demo11
demo11

C:\Users\PMJ>docker rm demo11
demo11
```

Saving username in a file (username.txt)

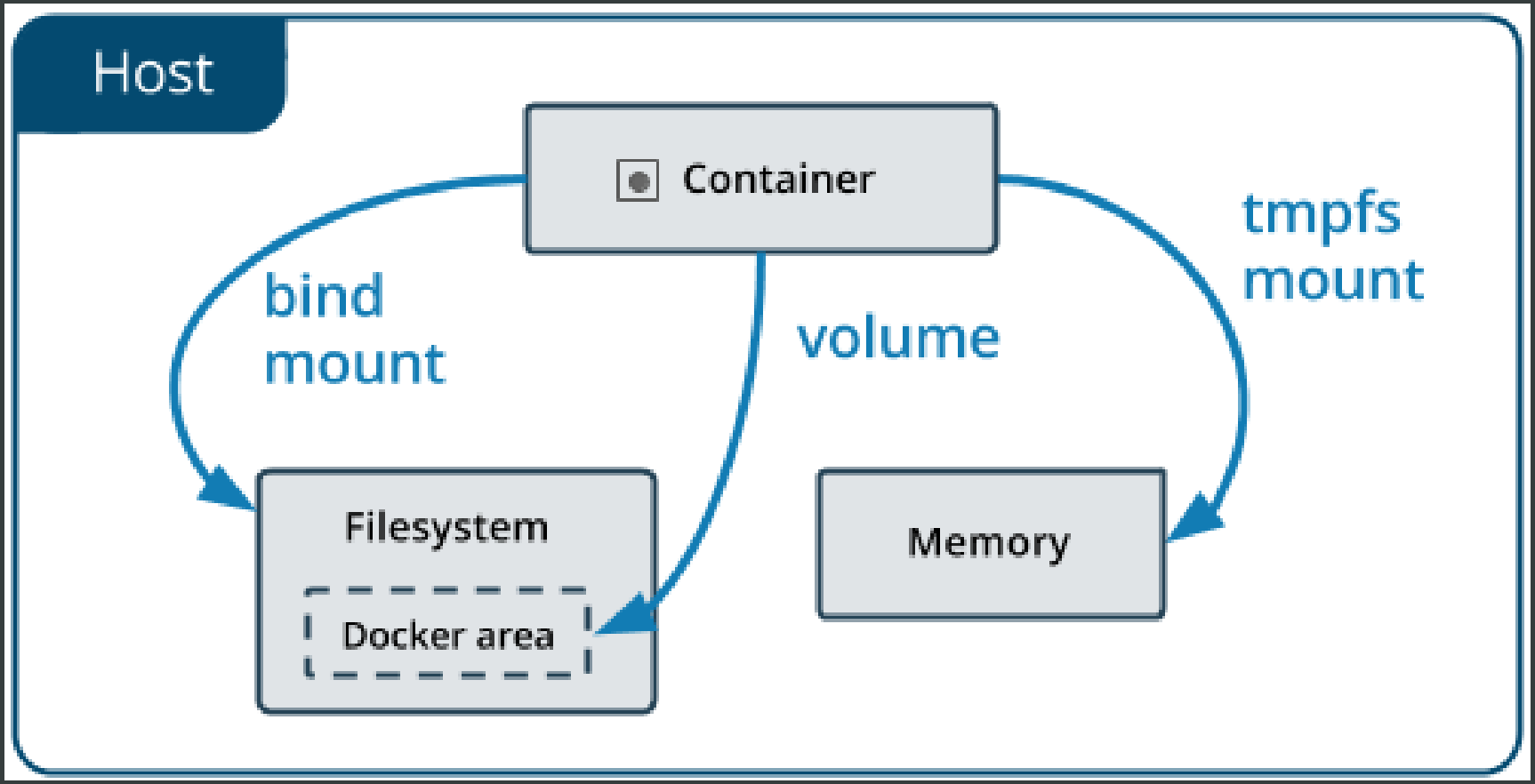
```
C:\Users\PMJ>docker images
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
pmjadav/node-app    v1.2      5789eccac8ad   23 seconds ago   191MB
pmjadav/node-app    v1.1      51cda4a53c19   47 hours ago     191MB

C:\Users\PMJ>docker run -dp 8000:8001 --name demo12 pmjadav/node-app:v1.2
934d5a4c7a51442a7b9d6e108d262664036656dfc928f15428814c3305430abb

C:\Users\PMJ>docker ps -a
CONTAINER ID   IMAGE                    COMMAND                   CREATED         STATUS         PORTS                     NAMES
934d5a4c7a51   pmjadav/node-app:v1.2    "docker-entrypoint.s…"    8 seconds ago   Up 7 seconds   0.0.0.0:8000->8001/tcp    demo12

C:\Users\PMJ>docker exec -it demo12 /bin/sh
/usr/src/app # ls
index.html          index.js            node_modules        package-lock.json   package.json        user
/usr/src/app # cat user/username.txt
Prashant Jadav
Robin Hood
```

Viewing Logs of running Container

```
C:\Users\PMJ>docker logs demo12

> demo@1.0.0 start
> node index.js

Node server is running on port 8001..
Sent: index.html
Prashant Jadav
 saved in the file.
Robin Hood
 saved in the file.
```

username.txt file data is lost every time we run new instance of the container.
The data do not persist.

Manage Data in Docker



Manage Data in Docker

- Volumes are stored in a part of the host filesystem which is managed by Docker (/var/lib/docker/volumes/ on Linux). Non-Docker processes should not modify this part of the filesystem. Volumes are the best way to persist data in Docker.

- Bind mounts may be stored anywhere on the host system. They may even be important system files or directories. Non-Docker processes on the Docker host or a Docker container can modify them at any time.

- tmpfs mounts are stored in the host system's memory only, and are never written to the host system's filesystem.

Volumes

- Volumes are the preferred mechanism for persisting data generated by and used by Docker containers
- While bind mounts are dependent on the directory structure and OS of the host machine.
- Volumes are completely managed by Docker.

Advantages of Volumes over Bind mounts

1) Volumes are easier to back up or migrate than bind mounts.
2) You can manage volumes using Docker CLI commands or the Docker API.
3) Volumes work on both Linux and Windows containers.
4) Volumes can be more safely shared among multiple containers.
5) Volume drivers let you store volumes on remote hosts or cloud providers, to encrypt the contents of volumes, or to add other functionality.
6) New volumes can have their content pre-populated by a container.

Advantages of Volumes over Bind mounts

7) Volumes on Docker Desktop have much higher performance than bind mounts from Mac and Windows hosts.
8) In addition, volumes are often a better choice than persisting data in a container's writable layer, because a volume does not increase the size of the containers using it
9) The volume's contents exist outside the lifecycle of a given container.

Creating a Volume

```
C:\Users\PMJ>docker volume create usernames
usernames

C:\Users\PMJ>docker volume ls
DRIVER    VOLUME NAME
local     usernames
```

```
C:\Users\PMJ>docker volume inspect usernames
[
    {
        "CreatedAt": "2023-03-08T08:01:36Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/usernames/_data",
        "Name": "usernames",
        "Options": {},
        "Scope": "local"
    }
]
```

Use of Volume while running the Container

docker run -dp 8000:8001
--name demo12 --mount
type=volume,src=usernames,target=/usr/src/app/user
pmjadav/node-app:v1.2

```
C:\Users\PMJ>docker run -dp 8000:8001 --name demo12 --mount type=volume,src=usernames,target=/usr/src/app/user pmjadav/node-app:v1.2
ab5b6704c32f0616f32bb2f32cb0de3c0659990c25b834d941f6a2ffb6984901
```

Persists the user names

```
C:\Users\PMJ>docker ps -a
CONTAINER ID   IMAGE                    COMMAND                   CREATED         STATUS         PORTS                     NAMES
ab5b6704c32f   pmjadav/node-app:v1.2    "docker-entrypoint.s…"    3 minutes ago   Up 3 minutes   0.0.0.0:8000->8001/tcp    demo12

C:\Users\PMJ>docker exec -it demo12 /bin/sh
/usr/src/app # cat user/username.txt
/usr/src/app # cat user/username.txt
Prashant Jadav
Robin Hood
/usr/src/app # exit

C:\Users\PMJ>docker stop demo12
demo12

C:\Users\PMJ>docker rm demo12
demo12

C:\Users\PMJ>docker run -dp 8000:8001 --name demo12 --mount type=volume,src=usernames,target=/usr/src/app/user pmjadav/node-app:v1.2
35ff9fd9e7f1e404e021f6e9f7032ad1e31e2e3bb73bd3125d74ee6b09e63ebb

C:\Users\PMJ>docker exec -it demo12 /bin/sh
/usr/src/app # cat user/username.txt
Prashant Jadav
Robin Hood
/usr/src/app # cat user/username.txt
Prashant Jadav
Robin Hood
John Doe
/usr/src/app #
```

After inserting user names via http://localhost:8000/

After inserting user names via http://localhost:8000/

Container Networking

- Containers, by default, run in isolation
- Container don't know about other processes or containers on the same machine
- To allow one container to talk to another networking is required
- If you place the two containers on the same network, they can talk to each other.

Create a Network (Node-app)

```
PS C:\Users\PMJ> docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
20d64d9bfcf2   bridge    bridge    local
87ac43b33d50   host      host      local
d02628085f8b   none      null      local
PS C:\Users\PMJ> docker network

Usage:  docker network COMMAND

Manage networks

Commands:
  connect     Connect a container to a network
  create      Create a network
  disconnect  Disconnect a container from a network
  inspect     Display detailed information on one or more networks
  ls          List networks
  prune       Remove all unused networks
  rm          Remove one or more networks

Run 'docker network COMMAND --help' for more information on a command.
PS C:\Users\PMJ> docker network create node-app
b0cad0dc2c23eea3c02127831ecf94dee6c6af3cbe30cd6102aa4c42b394133d
```

```
PS C:\Users\PMJ> docker run -d `
>>      --network node-app --network-alias mysql `
>>  --name mysql8 `
>>      -v node-mysql-data:/var/lib/mysql `
>>      -e MYSQL_ROOT_PASSWORD=secret `
>>      -e MYSQL_DATABASE=todos `
>>      mysql:8.0
Unable to find image 'mysql:8.0' locally
8.0: Pulling from library/mysql
b4ddc423e046: Pull complete
b338d8e4ffd1: Pull complete
b2b1b06949ab: Pull complete
daf393284da9: Pull complete
1cb8337ae65d: Pull complete
f6c2cc79221c: Pull complete
4cec461351e0: Pull complete
ab6bf0cba08e: Pull complete
8df43cafbd11: Pull complete
c6d0aac53df5: Pull complete
b24148c7c251: Pull complete
Digest: sha256:d8dc78532e9eb3759344bf89e6e7236a34132ab79150607eb08cc746989aa047
Status: Downloaded newer image for mysql:8.0
e1ef30abd32f3801af036a563b3130de17c9d79ac5fd25ca91133a28d75b486e
```

Run Mysql Container on Node-app Network

Renaming the tag (repository name)

```
PS C:\Users\PMJ> docker tag mysql:8.0 pmjadav/mysql:8.0
PS C:\Users\PMJ> docker images
REPOSITORY          TAG       IMAGE ID       CREATED        SIZE
pmjadav/node-app    v1.2      5789eccac8ad   3 hours ago    191MB
pmjadav/node-app    v1.1      51cda4a53c19   2 days ago     191MB
mysql               8.0       4f06b49211c0   12 days ago    530MB
pmjadav/mysql       8.0       4f06b49211c0   12 days ago    530MB
```

Running the Mysql Container on the node-app network

```
PS C:\Users\PMJ> docker run -d `
>>      --network node-app --network-alias mysql `
>>  --name mysql8 `
>>      -v node-mysql-data:/var/lib/mysql `
>>      -e MYSQL_ROOT_PASSWORD=secret `
>>      -e MYSQL_DATABASE=todos `
>>      pmjadav/mysql:8.0
6f4c39d698d374c3d0c1a113db1e4430a281a994117fb225fbd6a2cc92ced1ce
PS C:\Users\PMJ> docker ps -a
CONTAINER ID   IMAGE               COMMAND                  CREATED         STATUS         PORTS                   NAMES
6f4c39d698d3   pmjadav/mysql:8.0   "docker-entrypoint.s…"   4 seconds ago   Up 3 seconds   3306/tcp, 33060/tcp   mysql8
```

```
PS C:\Users\PMJ> docker exec -it mysql8 mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.32 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| todos              |
+--------------------+
5 rows in set (0.01 sec)

mysql> use todos;
Database changed
mysql> show tables;
Empty set (0.00 sec)
```

secret

Allowing remote client to connect to Mysql Server

```
mysql> alter user 'root'@'%' identified with mysql_native_password by 'secret';
Query OK, 0 rows affected (0.01 sec)
```

Connecting to Mysql from Node-app Container

```
PS C:\Users\PMJ> docker run -dp 8000:8001 --name demo12 `
>> --network node-app pmjadav/node-app:v1.2
5cf5409e910aa82a592cc6e4911fb34e7819c796698adbf41dbd7d355905b2c6
PS C:\Users\PMJ> docker exec -it demo12 /bin/sh
/usr/src/app # ping mysql
PING mysql (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.274 ms
64 bytes from 172.18.0.2: seq=1 ttl=64 time=0.093 ms
64 bytes from 172.18.0.2: seq=2 ttl=64 time=0.157 ms
64 bytes from 172.18.0.2: seq=3 ttl=64 time=0.124 ms
^C
--- mysql ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.093/0.162/0.274 ms
```

Installing My-sql Client on Node-app Container

```
/usr/src/app # apk add mysql-client
(1/5) Installing mariadb-common (10.6.12-r0)
(2/5) Installing ncurses-terminfo-base (6.3_p20221119-r0)
(3/5) Installing ncurses-libs (6.3_p20221119-r0)
(4/5) Installing mariadb-client (10.6.12-r0)
(5/5) Installing mysql-client (10.6.12-r0)
Executing busybox-1.35.0-r29.trigger
OK: 41 MiB in 22 packages
```

```
PS C:\Users\PMJ> docker exec -it demo12 /bin/sh
/usr/src/app # mysql -u root -p  -h 172.18.0.2 -D todos
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 15
Server version: 8.0.32 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [todos]> select * from tab;
ERROR 1146 (42S02): Table 'todos.tab' doesn't exist
MySQL [todos]> show tables;
Empty set (0.002 sec)

MySQL [todos]> create table todos(id int, name varchar(50), iscomplete bool);
Query OK, 0 rows affected (0.046 sec)

MySQL [todos]> insert into todos values (1, 'Lecture', 0);
Query OK, 1 row affected (0.013 sec)

MySQL [todos]> insert into todos values (2, 'Meeting', 0);
Query OK, 1 row affected (0.010 sec)
```

```
MySQL [todos]> select * from todos;
+------+---------+------------+
| id   | name    | iscomplete |
+------+---------+------------+
|    1 | Lecture |          0 |
|    2 | Meeting |          0 |
+------+---------+------------+
2 rows in set (0.001 sec)
```

Docker Compose

- Docker Compose is used to run multiple containers as a single service
- In the Node.js application which required Node-app and MySQL, we can create one file which would start both the containers as a service without the need to start each one separately
- We can create a YAML file to define the services and with a single command, can spin everything up or tear it all down.

Docker Compose Advantages

- We can
  - define our application stack in a file,
  - keep it at the root of our project repo, and
  - easily enable someone else to contribute to our project
- Someone would only need to clone our repo and start the compose app

Docker Compose Version

```
C:\Users\PMJ>docker compose version
Docker Compose version v2.15.1
```

Creating Compose File

- At the root of the node-app project folder create a file named compose.yml
- In the compose file start defining the list of services (or containers) we want to run as part of our application

Define the Node-app and Mysql services (compose.yml)

```yaml
services:
    node1:
        build:  .
        ports:
            - 8000:8001
        volumes:
            - ./usernames:/usr/src/app/user
    mysql:
        image: mysql:8.0
        volumes:
            - ./node-mysql-data:/var/lib/mysql
        ports:
            - 3306:3306
            - 33060:33060
        environment:
            MYSQL_HOST: mysql
            MYSQL_ROOT_PASSWORD: secret
            MYSQL_DATABASE: users
```

Build from the Dockerfile in the current directory

Use of Docker Compose to run the Containers

```
PS E:\docker\demo> docker compose up -d
[+] Running 12/12
 - mysql Pulled
   - 767a87c58327 Pull complete
   - cbd6d17e71a0 Pull complete
   - 9b17ad003fbc Pull complete
   - 410b54c19b6b Pull complete
   - c6192cec9415 Pull complete
   - f7be351756ff Pull complete
   - ae2d1ab519ee Pull complete
   - 119cfaa7dea0 Pull complete
   - 7176b3cc6ba1 Pull complete
   - 2eb39e909e2b Pull complete
   - e935886e1025 Pull complete
[+] Building 6.6s (12/12) FINISHED
```

```
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 212B
=> [internal] load .dockerignore
=> => transferring context: 177B
=> [internal] load metadata for docker.io/library/node:19-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [1/6] FROM docker.io/library/node:19-alpine@sha256:4a3a2ccfa80
=> => resolve docker.io/library/node:19-alpine@sha256:4a3a2ccfa80
=> [internal] load build context
=> => transferring context: 37.42kB
=> CACHED [2/6] WORKDIR /usr/src/app
=> CACHED [3/6] COPY package*.json ./
=> CACHED [4/6] RUN npm install
=> CACHED [5/6] RUN chown -R node /usr/src/app/node_modules
=> [6/6] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:a3eea8f6fed63f76a819349535e41be45b01da
=> => naming to docker.io/library/demo-node1
[+] Running 3/3
 - Network demo_default    Created
 - Container demo-mysql-1  Started
 - Container demo-node1-1  Started
```

Listing the Containers

```
PS E:\docker\demo> docker ps -a
CONTAINER ID   IMAGE        COMMAND                  CREATED         STATUS         PORTS
                            NAMES
9fb525bfbd35   demo-node1   "docker-entrypoint.s…"   6 seconds ago   Up 4 seconds   0.0.0.0:8000->8001/tcp
                            demo-node1-1
ab2936266967   mysql:8.0    "docker-entrypoint.s…"   6 seconds ago   Up 4 seconds   0.0.0.0:3306->3306/tcp,
 0.0.0.0:33060->33060/tcp   demo-mysql-1
```

Using Pre-existing Network

- If you want your containers to join a pre-existing network, use the external option

```
services:
    # ...
networks:
    network1:
        name: my-pre-existing-network
        external: true
```

References

- https://www.vmware.com/topics/glossary/content/virtual-machine.html
- https://www.docker.com/resources/what-container/
- https://docs.docker.com/get-started/overview/