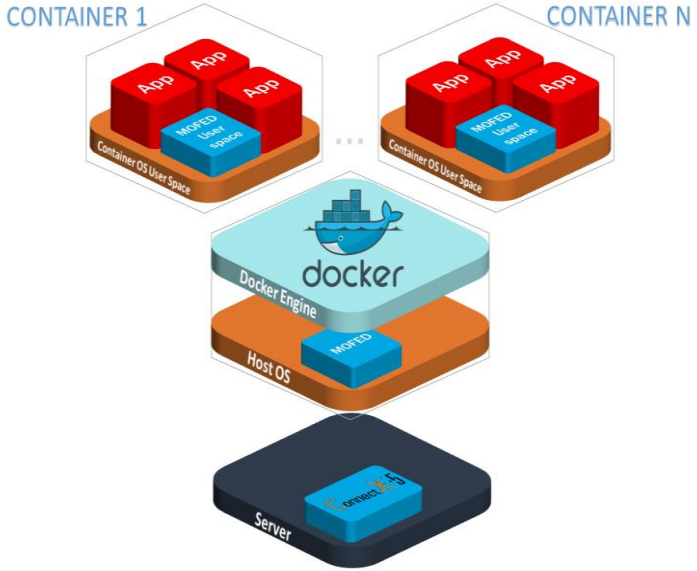
Docker and Containerization Basics

Package App + Dependencies + Configurations as Containers



Technical Trainers

SoftUni Team

Software University

SoftUni

https://softuni.bg

Software University

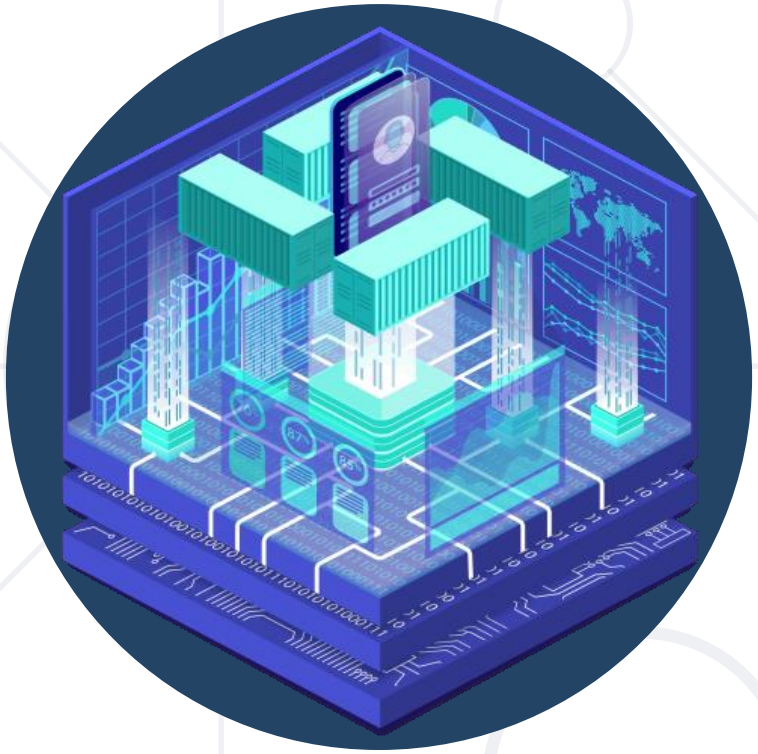
Have a Question?

Software University

sli.do

#Dev-Ops

Table of Contents

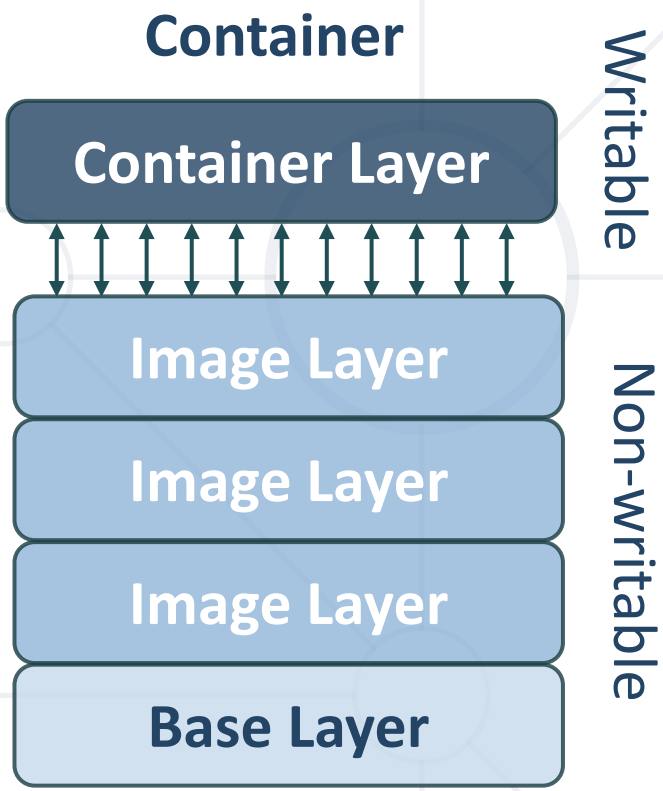


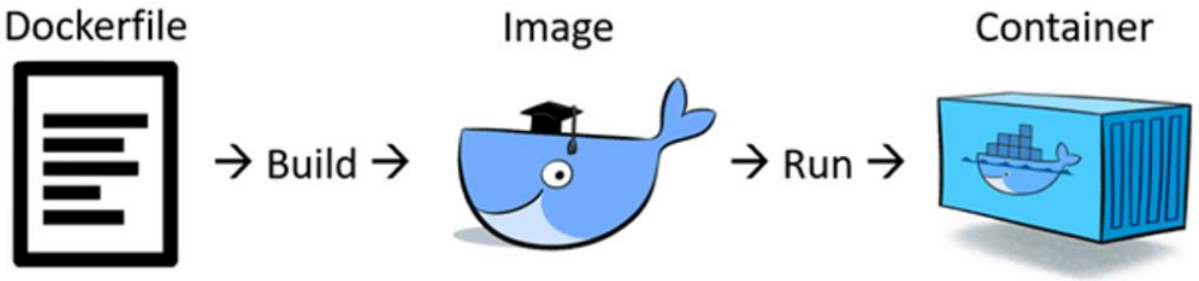
Containerization Overview

What is it, Advantages, VMs VS Containers

Containerization

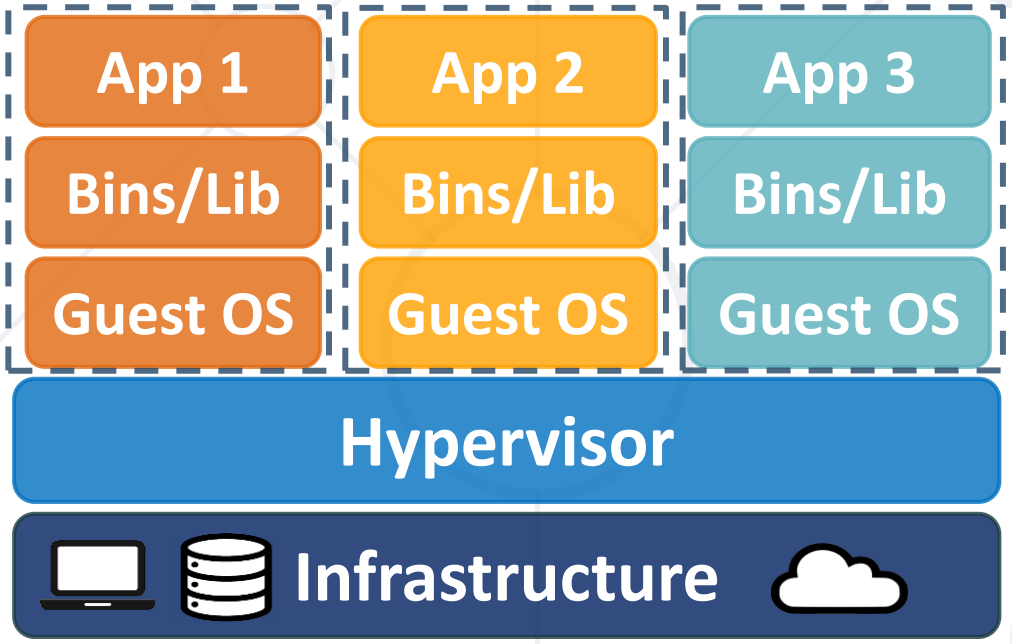
- **Containerization** == approach in which an **app or service** is packaged as a container
- **Image** == read-only template that contains a set of instructions for creating a container
  - It contains software, packaged with its dependencies and configuration
  - Designed to run in a virtual environment
- **Container** == a runnable instance of an image





VMs vs Containers

- **VMs** virtualize the hardware
- Complete isolation
- Complete OS installation. Requires more resources

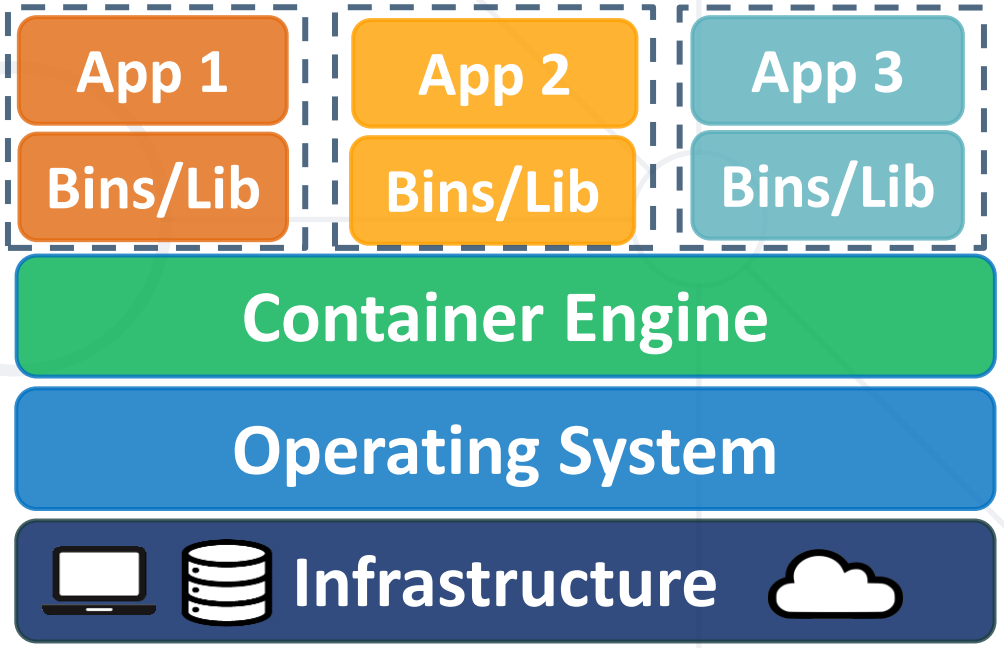


Utilization

Size

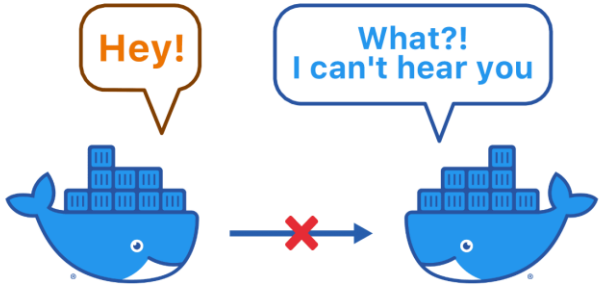
Boot Up Time

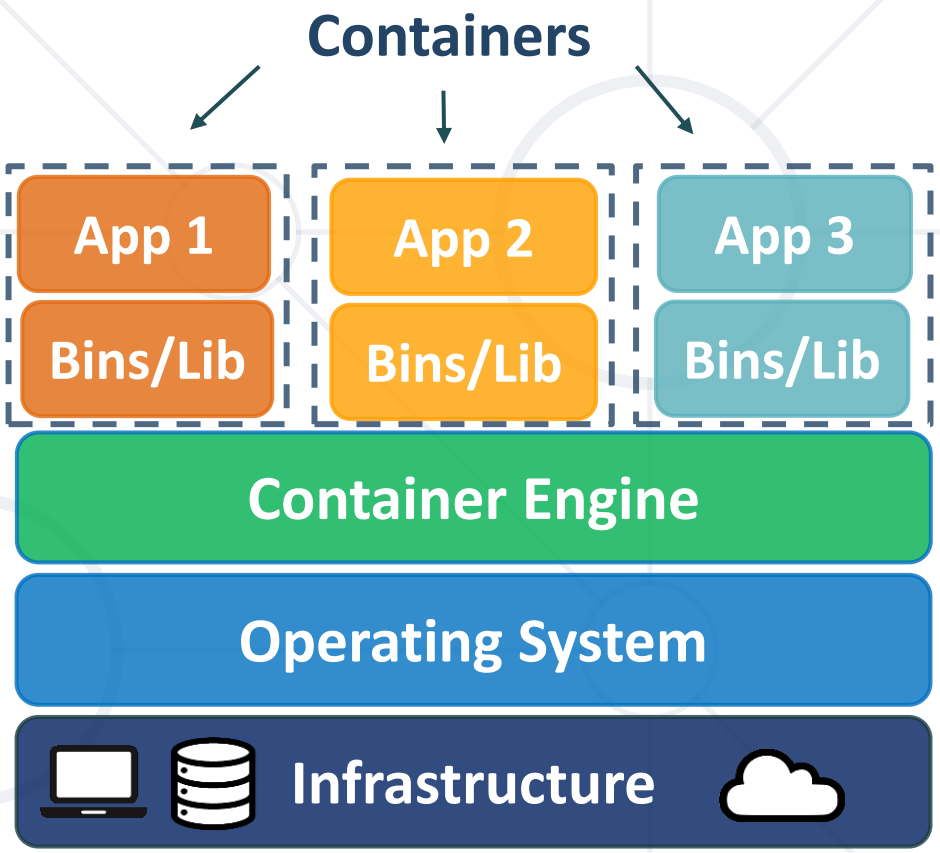
- **Containers** virtualize the OS
- Lightweight isolation
- Shared kernel. Requires fewer resources



Containerization – Advantages (1)

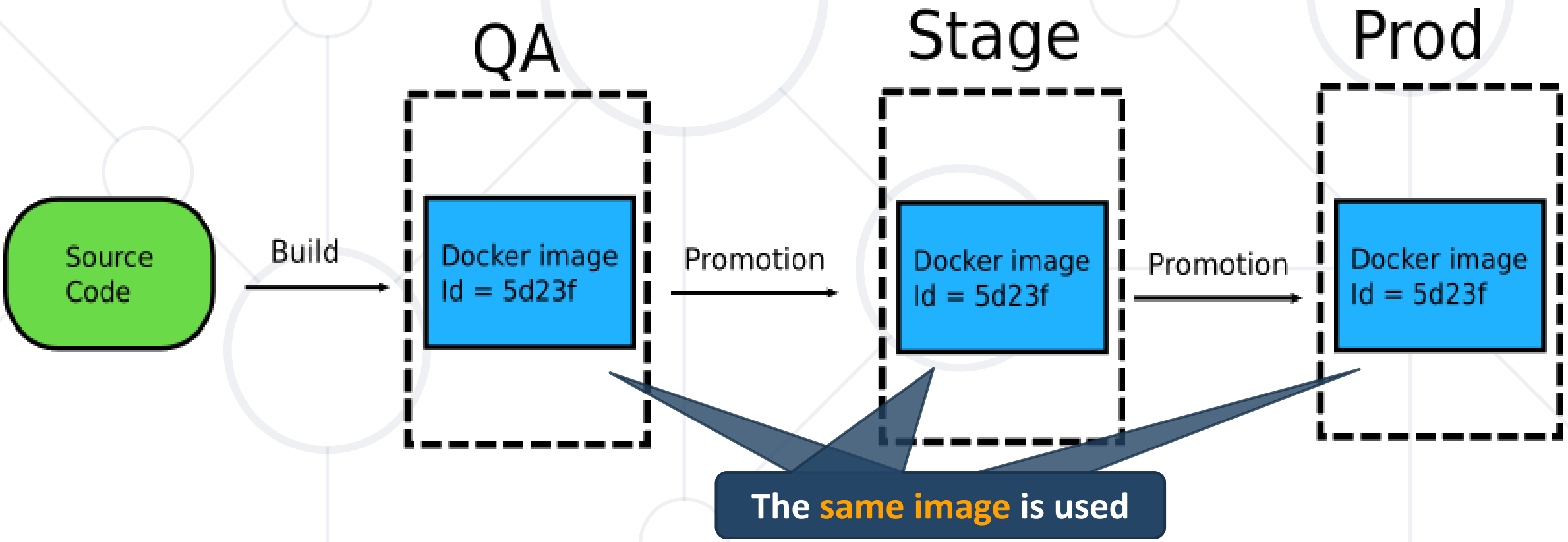
- A containerized app can be **tested** and **deployed** as a **unit** to the host OS
- Containers **isolate apps from each other** on a shared OS
  - Containers run on a container host
  - The container host runs on an OS (Linux / Windows / macOS)
  - Thus, having a smaller footprint than virtual machines

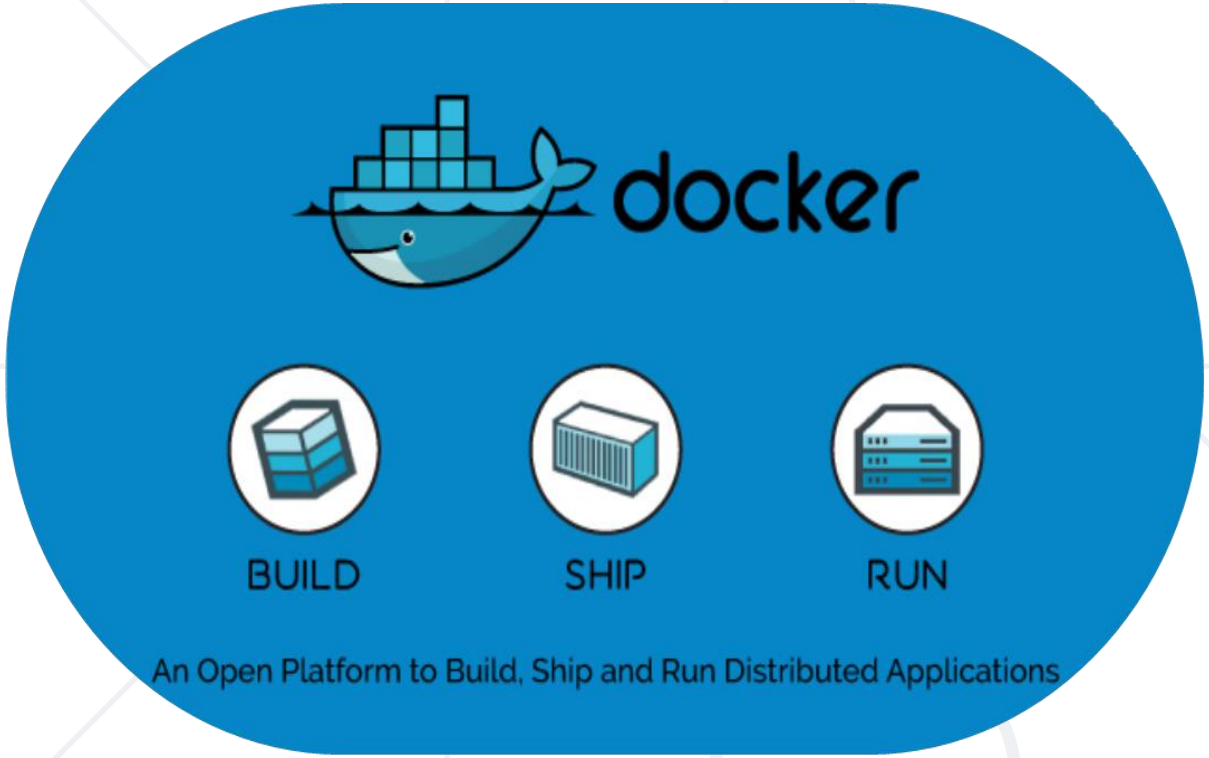




Containerization – Advantages (2)

- Containerization allows easy **deployments across environments** with little or no modification





Docker

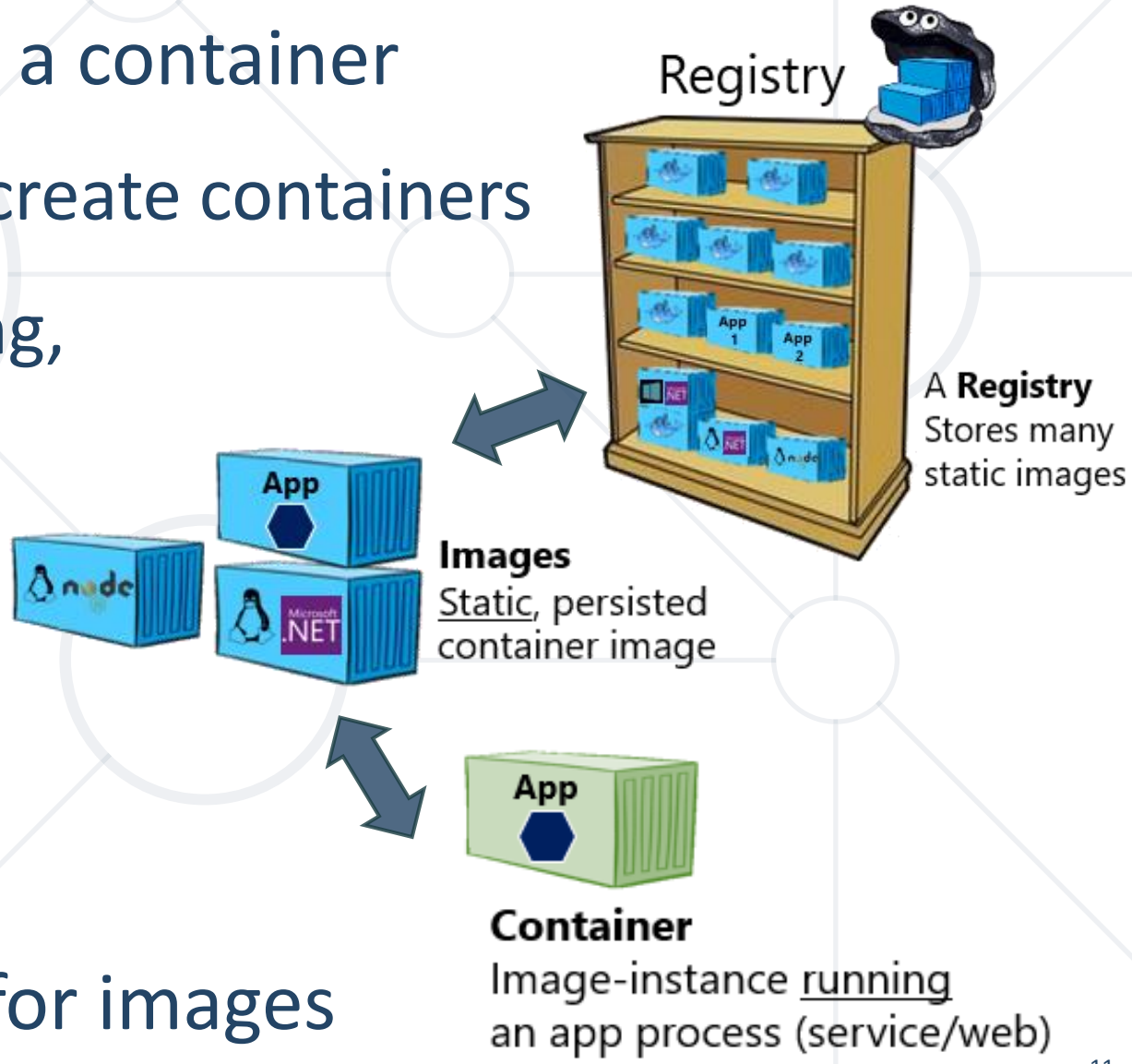
Docker Images, Containers, Software Development

Docker

- **Docker** == lightweight, open-source, secure **containerization platform**
- It simplifies **building**, **shipping**, and **running applications**
  - On different environments
- Runs natively on Linux or Windows servers
- Runs on Windows or Mac development machines
- Relies on **images** and **containers**

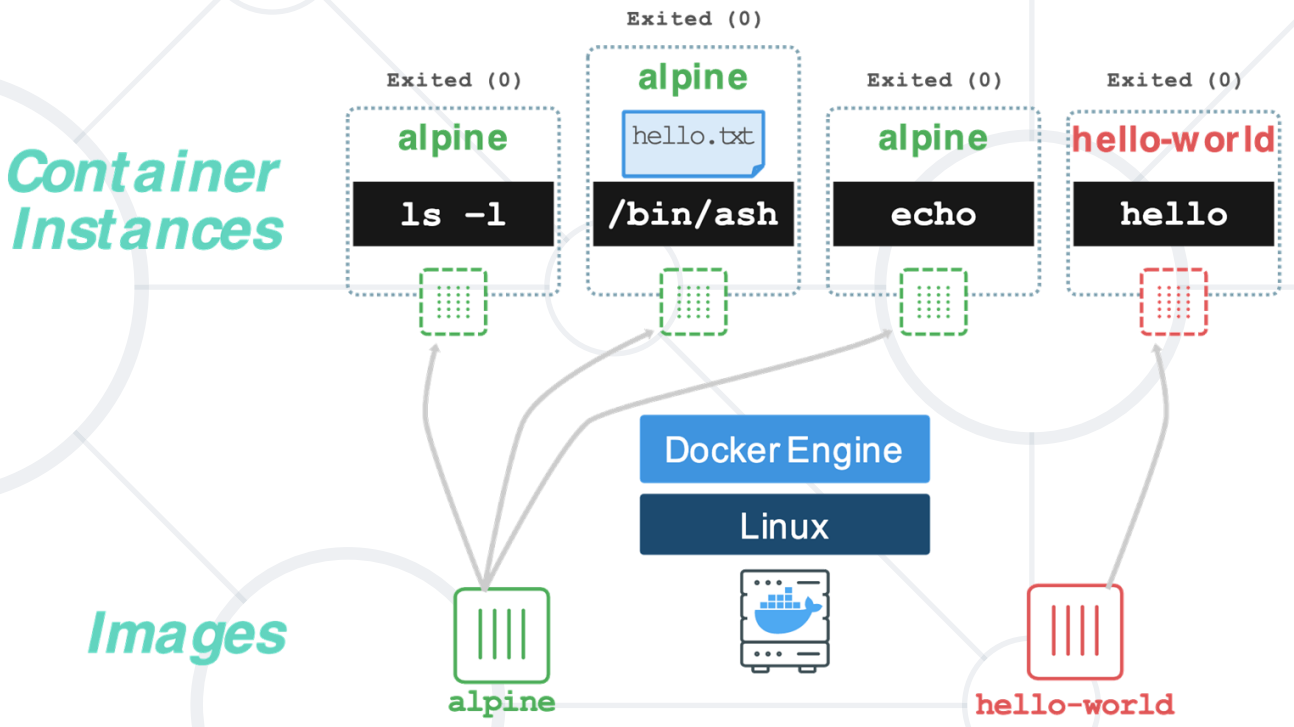
Docker Image

- A **Docker image** == blueprint for a container
  - A **read-only template**, used to create containers
  - If you want to change something, you should create a new image
  - Holds app/service/ other software
  - **Framework**, **dependencies** and **code** are "described" here
- **Docker registry** == a repository for images



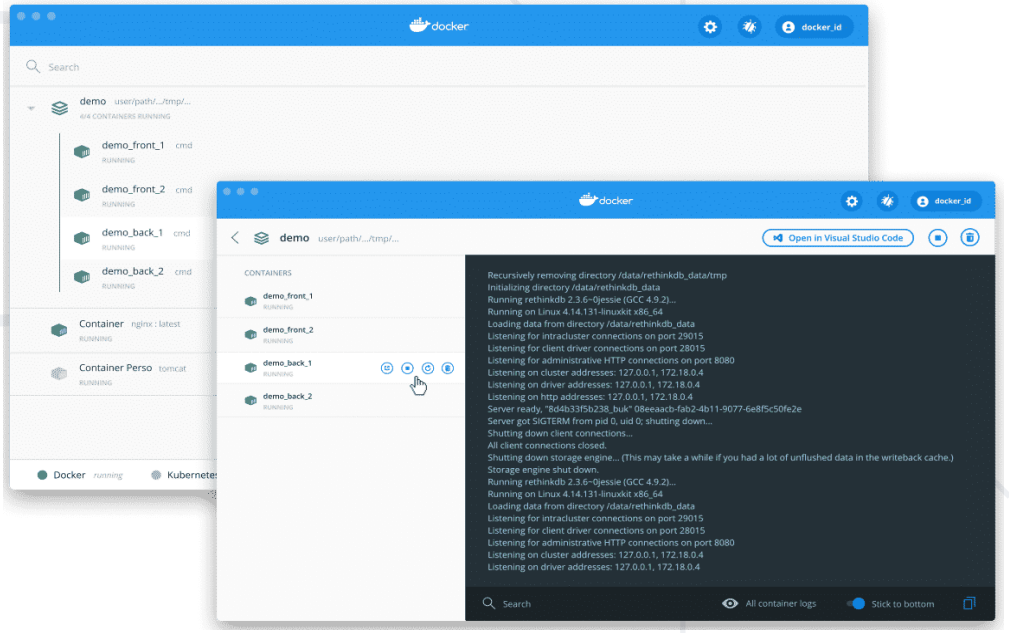
Docker Container

- Built **from the image**
  - Images become containers at **runtime**
- It is the actual **running environment** for your app
- **Isolated** and **secured**
- It can be started/stopped/deleted
- Different app components may reside in separate containers
  - Database, back-end, front-end, caching, messaging, etc.



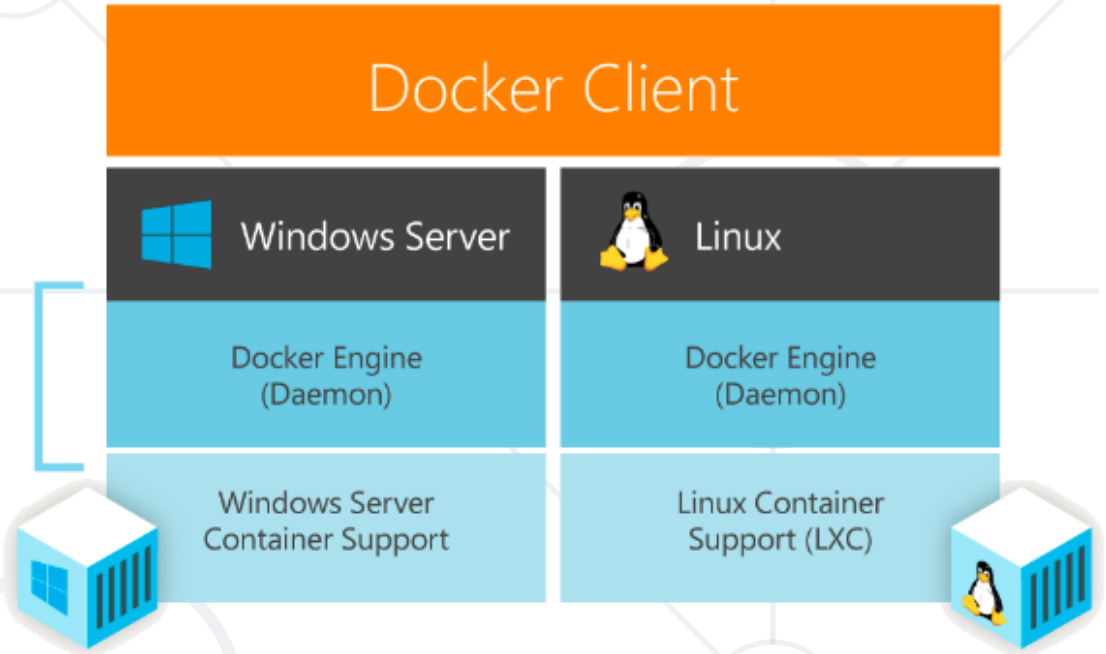
Docker Desktop

- Runs on Windows or Mac development machines
- Includes **Docker Engine**, **CLI** and **Kubernetes**
- Complete Docker **development environment**
- Containerize and share any application
  - https://www.docker.com/products/docker-desktop



Docker Desktop (2)

- On Windows
  - Ability to switch between **Linux** and **Windows Server environments**
  - Typically runs Linux containers through **WSL2** technology (Windows Subsystem for Linux)
  - https://docs.docker.com/desktop/install/windows-install

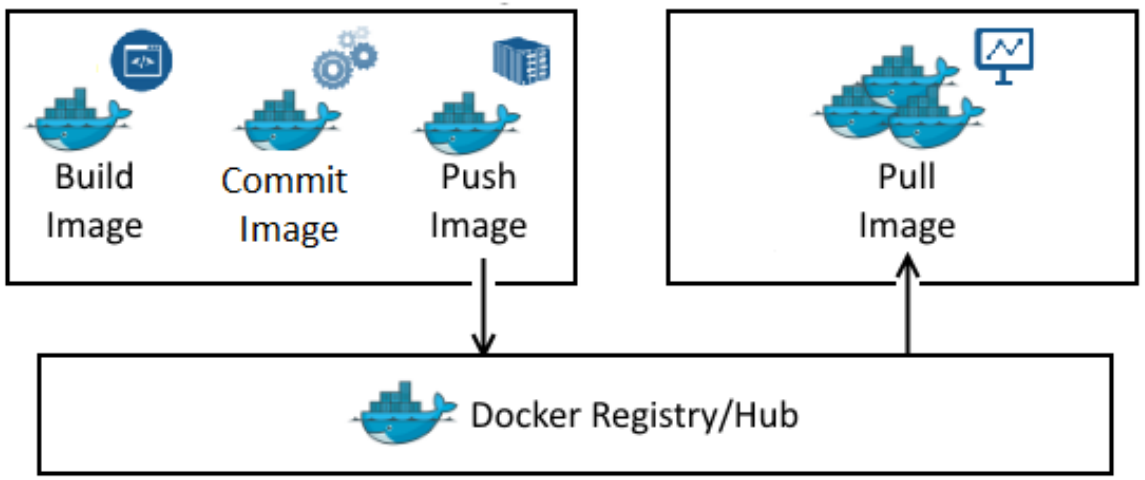


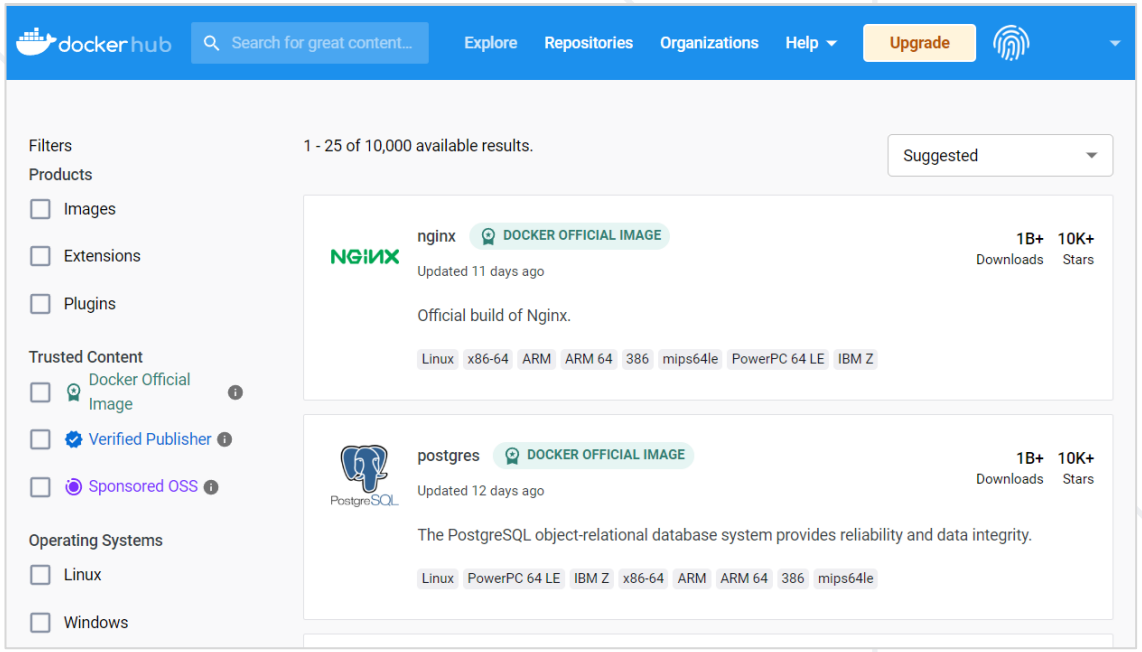
- There are third-party solutions for Linux – DockStation, CairoDock, and more…

Docker Hub

- **Docker Hub** == cloud-based **image repository** (registry)
- Used for easy **finding** and **sharing images**
- Supports **public and private repositories**
- Automated builds and webhooks

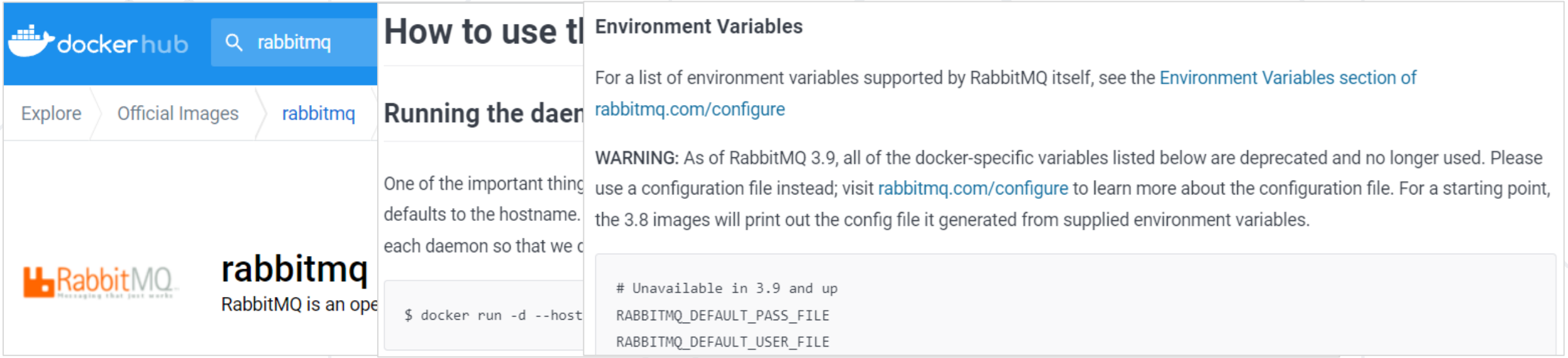
https://hub.docker.com





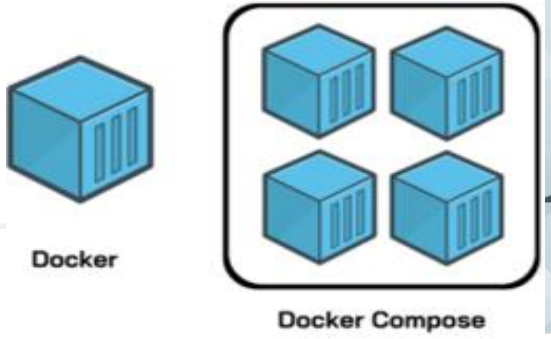
Docker Hub Image Overview

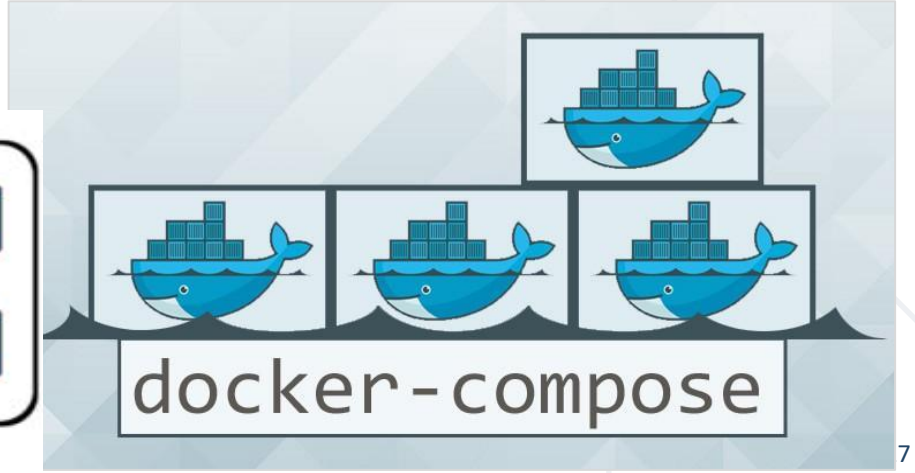
- For every tool we use in Docker, it is recommended that we **read its documentation** first
  - As sometimes we need to perform configurations to work with the tool
  - For example, for RabbitMQ: https://hub.docker.com/_/rabbitmq



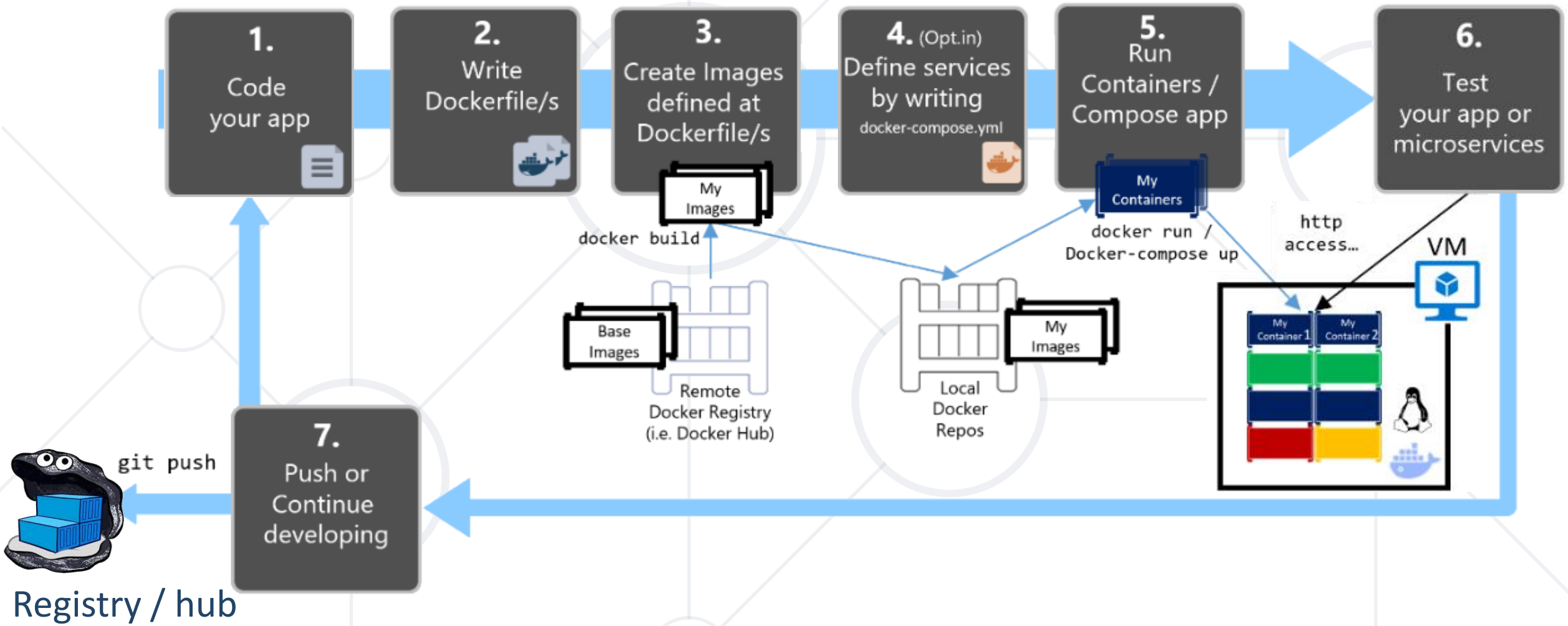
Docker Compose

- Some apps combine multiple components
  - e.g., WordPress requires Linux + NGINX + PHP + MySQL
  - Each component may run in a separate Docker container
- To run **multiple connected containers**, we use **Docker Compose**

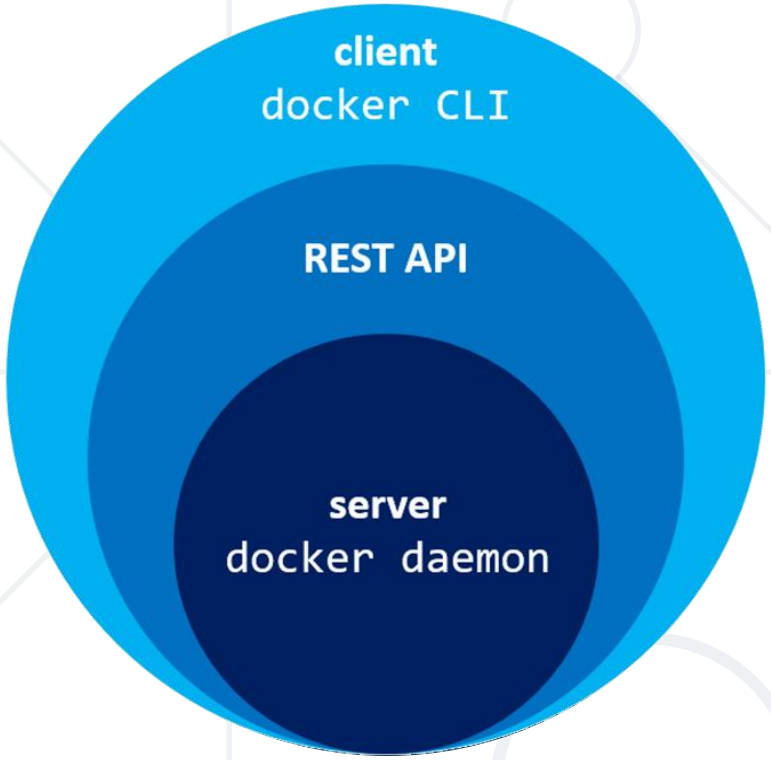




Development Workflow for Docker Apps



Registry / hub



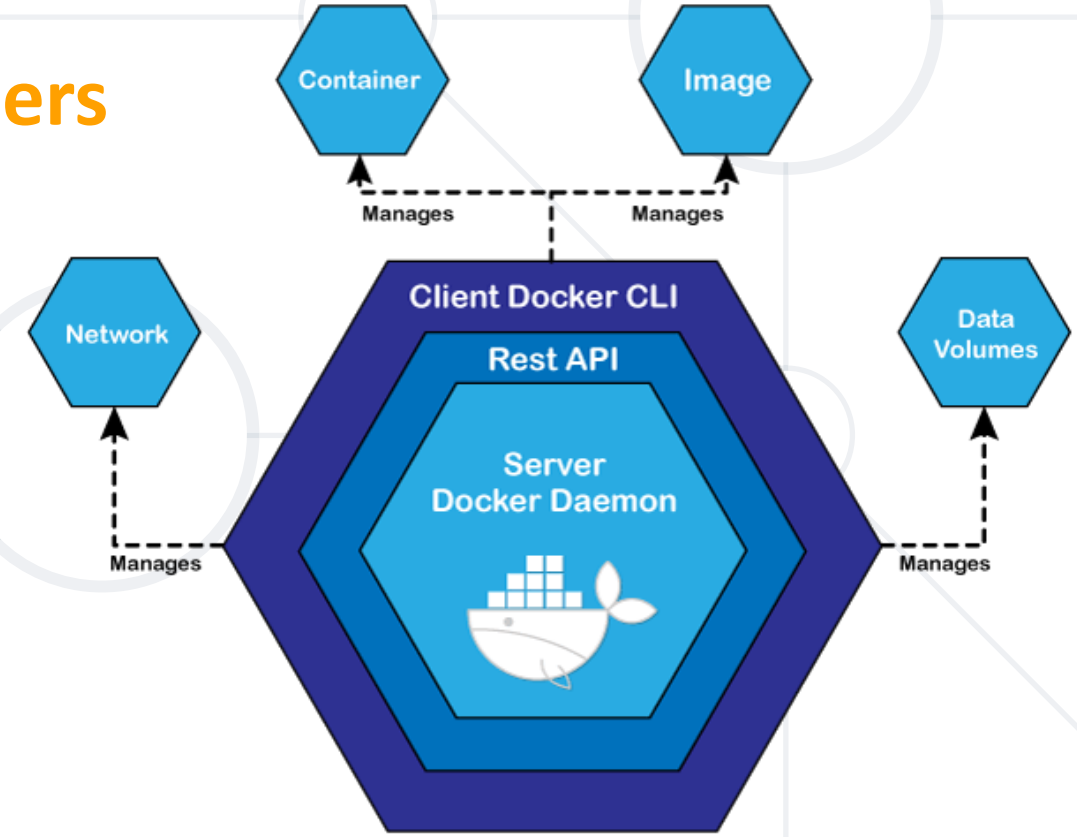
Docker CLI

Command Line Tool to Talk to the Docker Daemon

Docker CLI

- **Docker CLI** allows working with the **Docker Engine**
  - Build and manage **images**
  - Run and manage **containers**
- Example commands

```
docker pull [image]
docker run [image]
docker images
docker ps
docker logs [container]
```



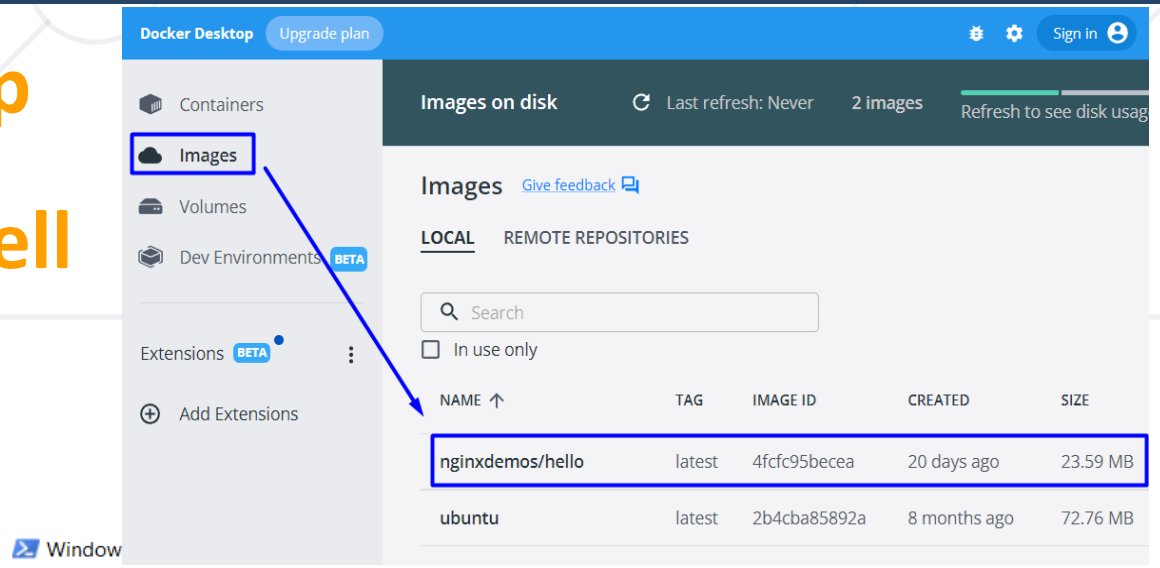
Docker CLI Demo

- First, install and run **Docker Desktop**
- Open a **CLI** – for example, **PowerShell**
- Let's **download** a sample **NGINX** server

```
docker pull nginxdemos/hello
```

- See **all images**

```
docker images
```




```
PS C:\Users\PC> docker pull nginxdemos/hello
Using default tag: latest
latest: Pulling from nginxdemos/hello
213ec9aee27d: Pull complete
ae98275d0ecb: Pull complete
121e2d9f6af2: Pull complete
6a07d505af0f: Pull complete
3e8957b70867: Pull complete
2806408d582e: Pull complete
843ea801d698: Pull complete
                 Pull complete
                 complete
a28dbd7b5e9c74b3221291e8b2cbbd507e292ca716a
 newer image for nginxdemos/hello:latest
os/hello:latest
```

By default, the **latest image version** is installed

```
PS C:\Users\PC> docker images
REPOSITORY        TAG       IMAGE ID       CREATED        SIZE
nginxdemos/hello  latest    4fcfc95becea   2 weeks ago    23.6MB
ubuntu            latest    2b4cba85892a   8 months ago   72.8MB
```

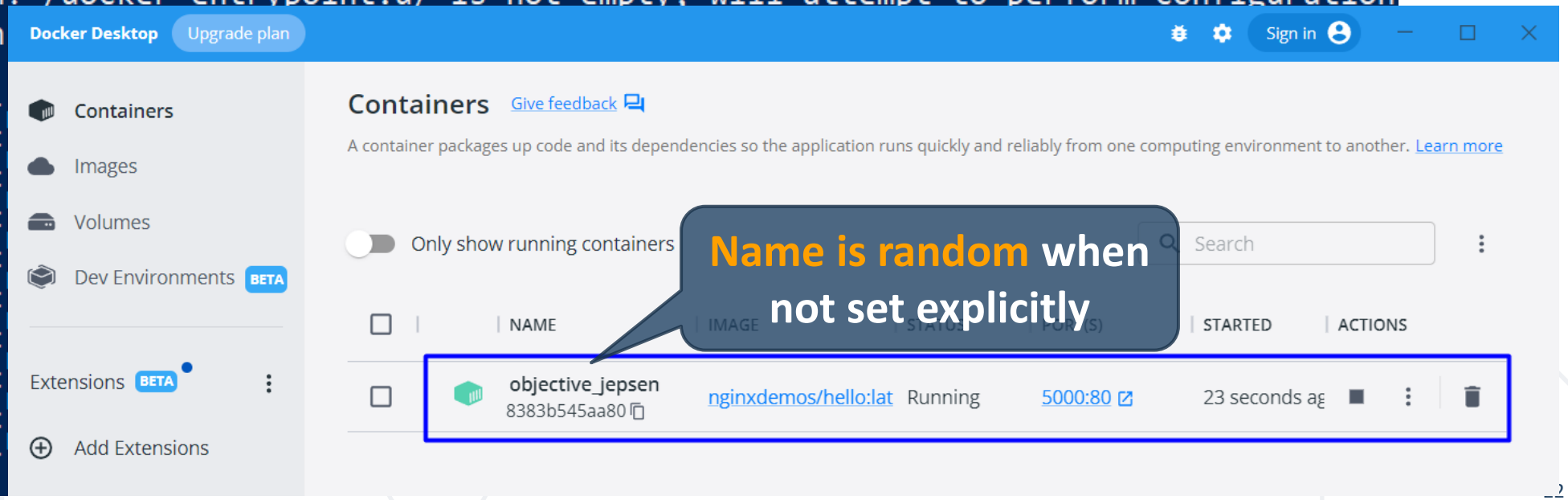
Docker CLI Demo (2)

- To run NGINX, we need to **expose ports** from the container
  - As the server runs on **port 80**, which is isolated and inaccessible

```
docker run -p 5000:80 nginxdemos/hello
```

You can choose **any free port**, not only 5000

```
PS C:\Users\PC> docker run -p 5000:80 nginxdemos/hello
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh
...
2022/11/08 13:03:48 [
2022/11/08 13:03:48 [
2022/11/08 13:03:48 [
2022/11/08 13:03:48 [
2022/11/08 13:03:48 [
2022/11/08 13:03:48 [
2022/11/08 13:03:48 [
2022/11/08 13:03:48 [
2022/11/08 13:03:48 [
...
```



Name is random when not set explicitly

Docker CLI Demo (3)

- See the server on **localhost:5000**

- Run it in **detached mode** with a **name**

```
docker run -p 5000:80 -d --name
code_it_up nginxdemos/hello
```

```
PS C:\Users\PC> docker run -p 5000:80 -d --name code_it_up nginxd
cd35b688a0ed9a6751bd98f2f1d8002f1f9e77dd9fb451a5ed6448afe1b2511c
```

- **Connecting** to a running container's console

```
docker attach {id} / {name}
```

```
PS C:\Users\PC> docker attach code_it_up
172.17.0.1 - - [13/Dec/2022:13:17:09 +0000] "GET /hello HTTP/1.1" 200 7236 "-" "Mozilla/5.0 (Windo
ws NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36" "-"
```

Docker CLI Demo (4)

- To see **container logs**

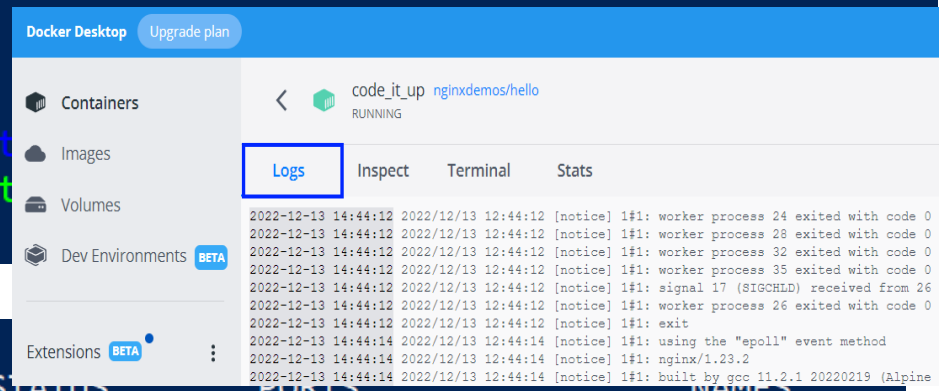
```
docker logs {id} / {name}
```

```
PS C:\Users\PC> docker logs code_it_up
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
```

- To open a **command shell** into a running container

```
docker exec -it code_it_up /bin/sh
```

```
PS C:\Users\PC> docker exec -it code_it_up /bin/sh
/ # ls
bin
dev
docker-ent
docker-ent
etc
```



- To see **all running containers**

```
docker ps
```

```
PS C:\Users\PC> docker ps
CONTAINER ID  IMAGE             COMMAND                  CREATED        STATUS        PORTS                    NAMES
e0be2118bd40  nginxdemos/hello  "/docker-entrypoint.…"  10 seconds ago  Up 9 seconds  0.0.0.0:5000->80/tcp  code_it_up
```

- To **stop the container**

```
docker stop {id} / {name}
```

```
PS C:\Users\PC> docker stop e0
e0
```

You can use only the **first two symbols** of the id

Docker CLI Demo (5)

- To show **all ran containers** `docker ps -a`

```
PS C:\Users\PC> docker ps -a
CONTAINER ID  IMAGE            COMMAND                  CREATED            STATUS                     PORTS  NAMES
e0be2118bd40  nginxdemos/hello "/docker-entrypoint.…"  26 seconds ago     Exited (0) 25 seconds ago         code_it_up
8383b545aa80  nginxdemos/hello "/docker-entrypoint.…"  About a minute ago Exited (0) 49 seconds ago         objective_jepsen
```

- **Delete the container**

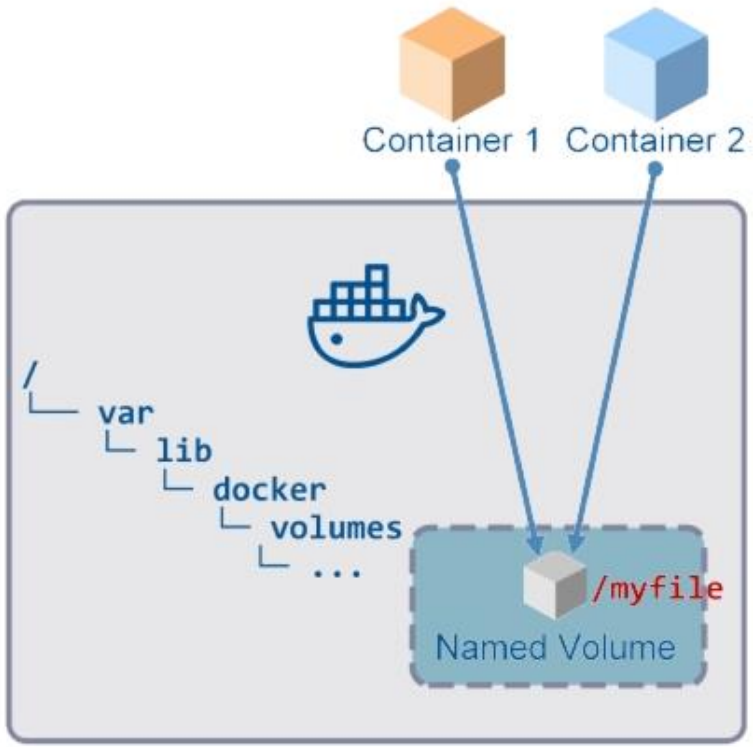
`docker rm {id} / {name}`

```
PS C:\Users\PC> docker rm e0
e0
```

	NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
	objective_jepsen 8383b545aa80	nginxdemos/hello:lat	Exited	5000:80		
	code_it_up e0be2118bd40	nginxdemos/hello:lat	Running	5000:80	34 seconds ag	

- **Delete the image** from local disk `docker rmi nginxdemos/hello`

```
PS C:\Users\PC> docker rmi nginxdemos/hello
Untagged: nginxdemos/hello:latest
Untagged: nginxdemos/hello@sha256:c0ba28dbd7b5e9c74b3221291e8b2cbbd507e292ca71df35ae5e0a4a0ed4436a
Deleted: sha256:4fcfc95beceab8ed406c0e818cd2eaac8d80a738ad00bf56dfe4cdddd1f98201
```

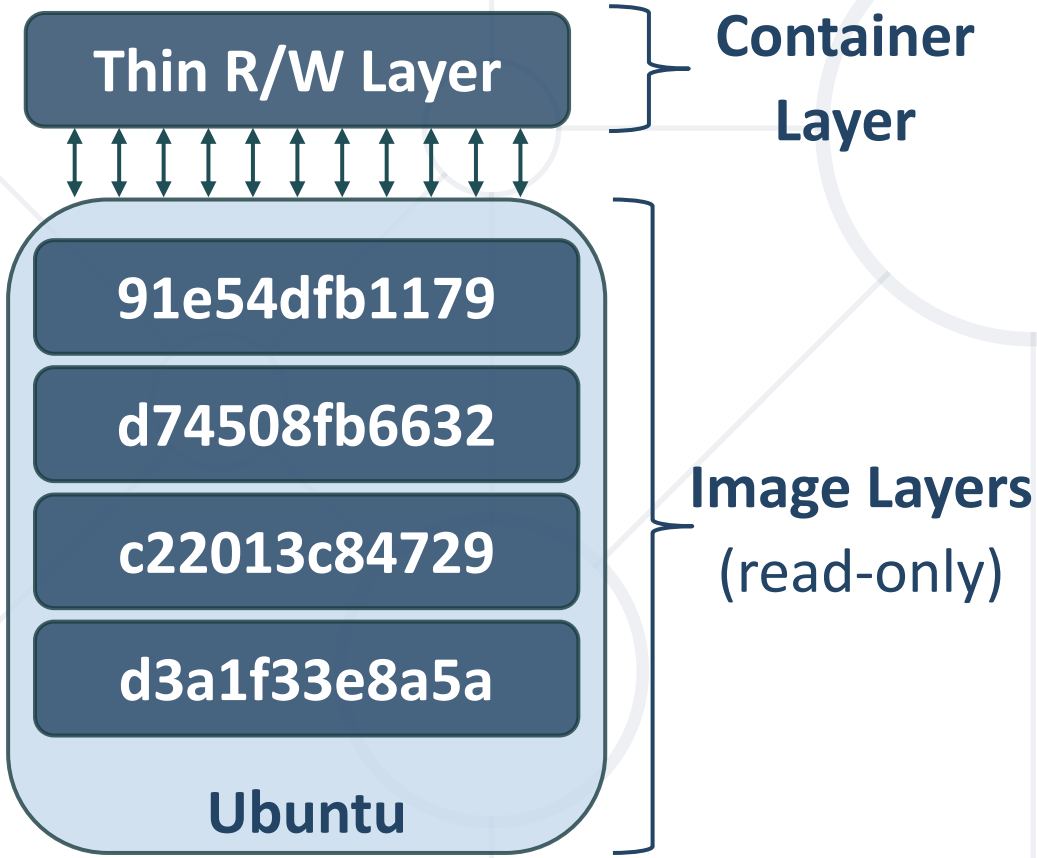


File System and Volume

Data in Docker Containers

Layered File System

- Each **image** has **file system layers**, which are read-only and isolated



Container, based on Ubuntu

- Image layers are **reused** in different images

Image

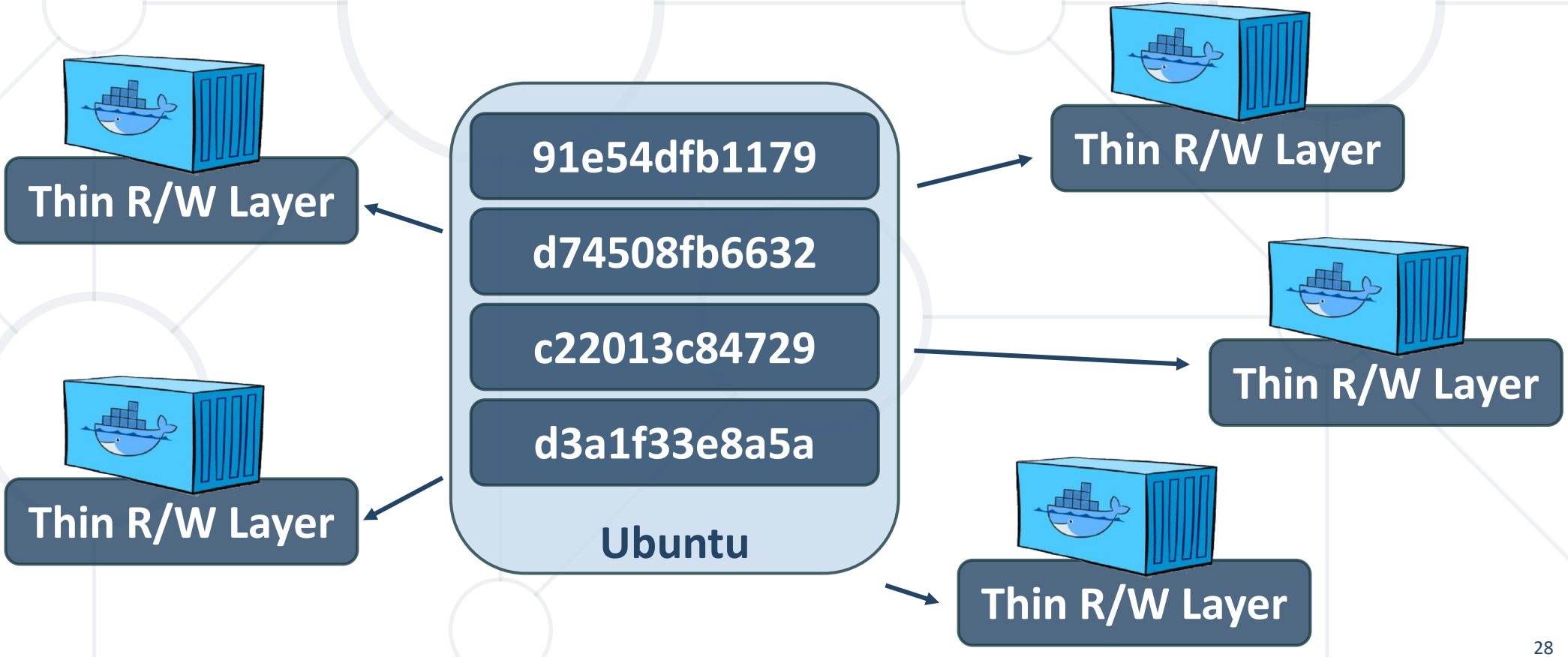
```
PS C:\Users\PC> docker pull node
Using default tag: latest
latest: Pulling from library/node
17c9e6141fdb: Already Exists
de4a4c6caea8: Already Exists
4edced8587e6: Pull complete
a7969cffbf46: Pull complete
74fbfde6af91: Downloading [=======>                    ]  77.08MB/196.9MB
babbacf2d498: Download complete
bd2943500448: Downloading [=================>          ]  24.78MB/45.38MB
98eaae01c196: Download complete
61339482de65: Download complete
```

Layer exists from another image

Image layers

Layered File System

- **Images** share **layers**
  - Therefore they load faster once you have them



Container Isolation

- Each container is **isolated** and has its **own writable file system**
  - By default, file system is deleted after you delete the container
  - Which is not very suitable for persistence operations

Delete old container and **create** a new one

```
PS C:\Users\PC> docker exec -it code_it_up /bin/sh
/ # touch test.txt
/ # ls
bin
dev
docker-e
docker-e
etc
home
lib
```

```
PS C:\Users\PC> docker exec -it code_it_up /bin/sh
/ # ls
bin                    media                  srv
dev                    mnt                    sys
docker-entrypoint.d    opt                    tmp
docker-entrypoint.sh   proc                   usr
etc                    root                   var
home                   run
lib                    sbin
```

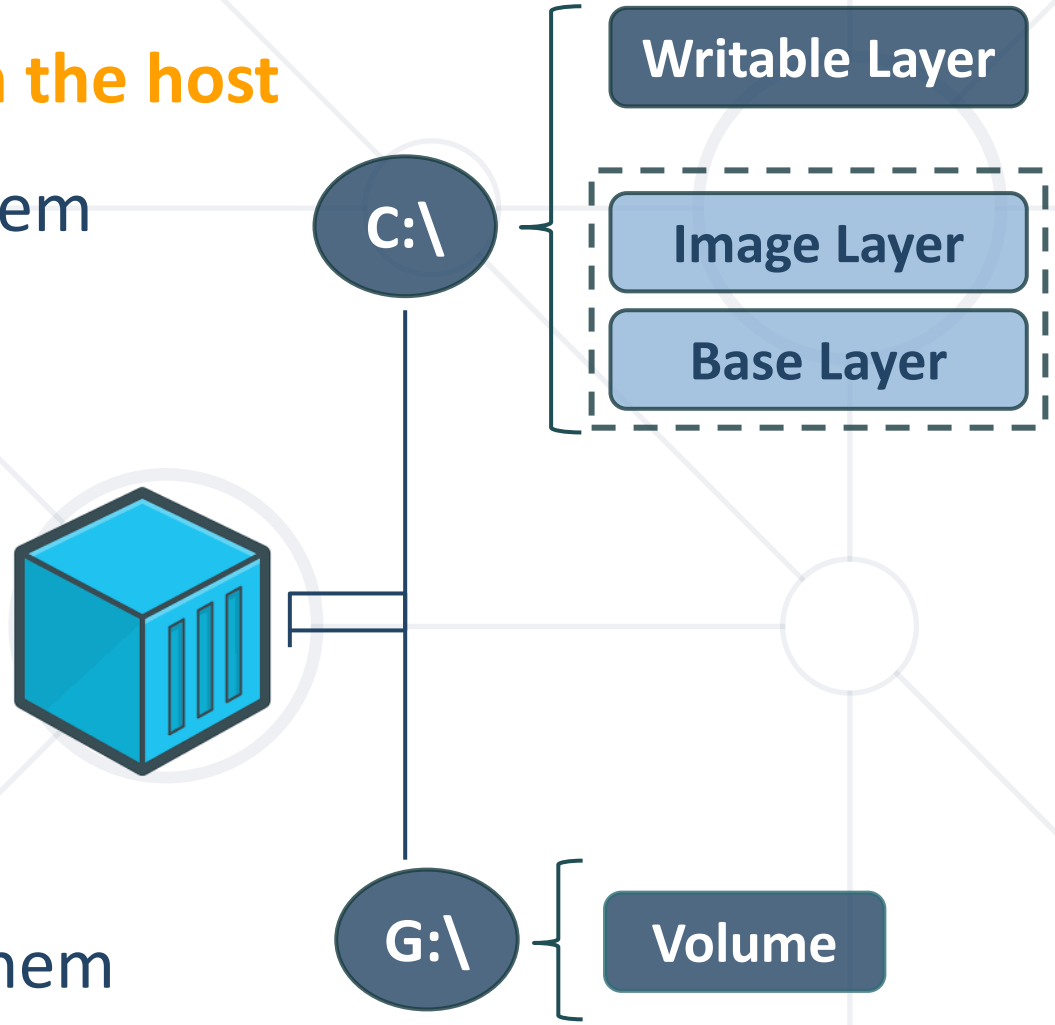
code_it_up
7fbae24f31a3

code_it_up
774cdfc8a290

test.txt file is missing

Volumes

- To persist data, use **volumes**
  - Special type of **directory on the host**
  - Mapped to the real file system
  - Can be shared and reused among containers
  - Image updates won't affect volumes
  - Persisted even after the container is deleted
  - You have full control over them



Attach Local Folder as Volume

- Attach **local folder as volume** to a container

```
docker run -p 5001:80 -d -v c:\users:/app nginxdemos/hello
```

```
PS C:\Users\PC> docker run -p 5001:80 -d -v c:\users:/app nginxdemos/hello
fff523c5c1b81e457a53d51ee5afa963553c8523766846f906002053a695d157
```

busy_shaw
fff523c5c1b8

- Examine mapped container's **/app** folder

```
PS C:\Users\PC> docker exec -it busy_shaw /bin/sh
/ # cd /app
/app # ls -al
total 4
dr-xr-xr-x    1 root     root     4096 Nov  5  2021 .
drwxr-xr-x    1 root     root     4096 Dec 14 08:50 ..
lrwxrwxrwx    1 root     root       23 Dec  7  2019 All Users -> /mnt/host/c/ProgramData
dr-xr-xr-x    1 root     root     4096 Nov  6  2021 Default
lrwxrwxrwx    1 root     root       25 Dec  7  2019 Default User -> /mnt/host/c/Users/Default
drwxrwxrwx    1 root     root     4096 Dec 12 12:09 PC
drwxrwxrwx    1 root     root     4096 Nov  5  2021 Public
-r-xr-xr-x    1 root     root      174 Dec  7  2019 desktop.ini
```

/app has files from **c:\users**

C:\Users

Name

Default
PC
Public

Creating and Using Volumes

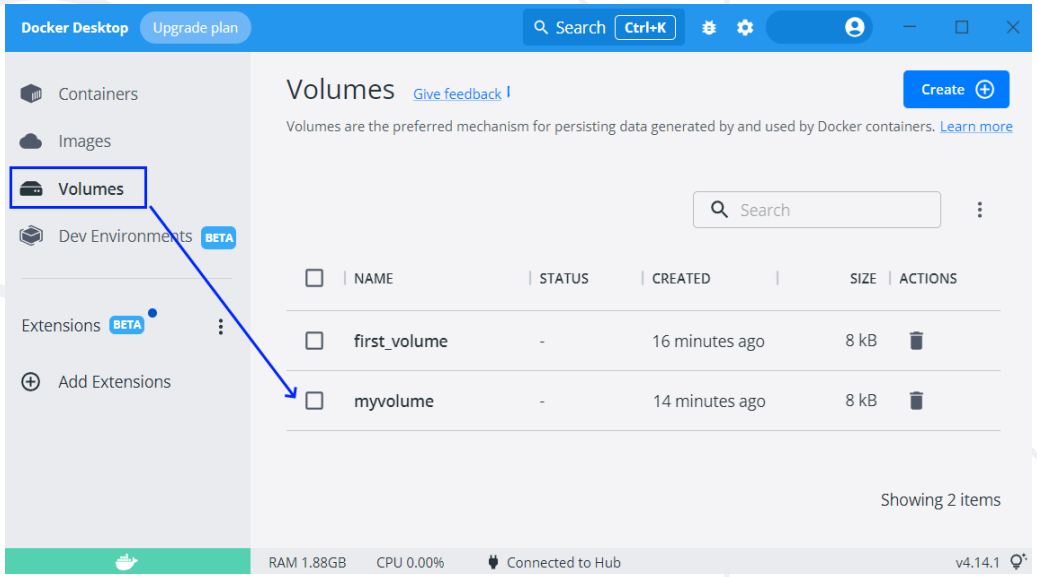
- **Create** a volume `docker volume create myvolume`
- **List** all volumes `docker volume ls`
- **Inspect** volume

```
docker volume inspect myvolume
```

```
PS C:\Users\PC> docker volume create myvolume
myvolume
```

```
PS C:\Users\PC> docker volume ls
DRIVER    VOLUME NAME
local     first_volume
local     myvolume
```

```
PS C:\Users\PC> docker volume inspect myvolume
[
    {
        "CreatedAt": "2022-12-14T08:14:20Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/myvolume/_data",
        "Name": "myvolume",
        "Options": {},
        "Scope": "local"
    }
]
```



Creating and Using Volumes (2)

- **Mount volume** to container

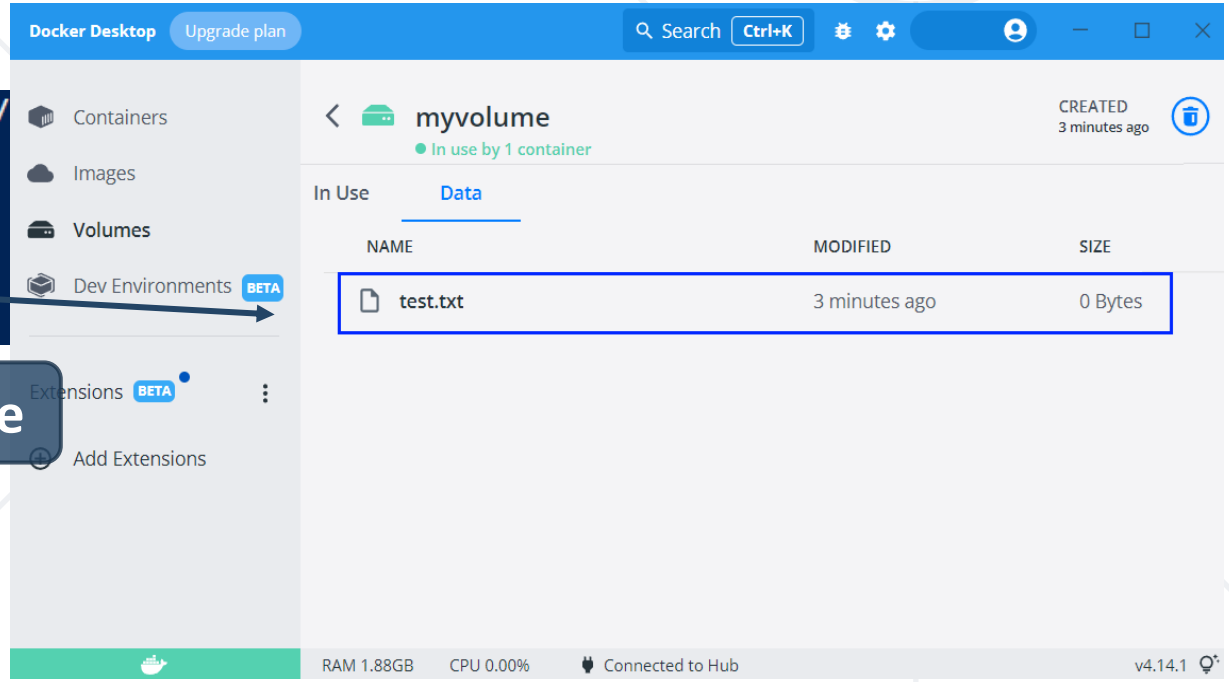
```
docker run -p 5000:80 -d -v myvolume:/myapp nginxdemos/hello
```

```
PS C:\Users\PC> docker run -p 5000:80 -d -v myvolume:/myapp nginxdemos/hello
061e1027c3830bb7321485258e9b1573967be98d998a241c5dfbc1bb30b923f4
```

zen_rhodes
061e1027c383

- Create a **file** in the **/myapp** folder

```
PS C:\Users\PC> docker exec -it zen_rhodes /bin/
/ # cd /myapp
/myapp # touch test.txt
/myapp # ls
test.txt
```



- **Remove** volume

Should not be in use

```
docker volume rm myvolume
```

```
PS C:\Users\PC> docker volume rm myvolume
myvolume
```

Vue.js App in a Container

Set Up and Run the App

Create App

- Create a new **Vue.js** application

```
vue init browserify MyWebsite
```

- Navigate to the application folder

```
cd MyWebsite
```

```
PS C:\Users\     > vue init browserify MyWebsite

? Project name my-website
? The version of the package 0.1.0
? Project description A Vue.js project
? Author [redacted]
? Vue build standalone
? Use ESLint to lint your code? Yes
? Setup unit tests with Karma + Jasmine? No

   vue-cli · Generated "MyWebsite".

   To get started:

     cd MyWebsite
     npm install
     npm run dev

PS C:\Users\     > cd MyWebsite
PS C:\Users\     \MyWebsite> npm install
```

Set and Run the App on Default Port

- Pull the **node.js image**

```
docker pull node:16
```

- Set the application's **default IP** in **package.json**

- Run the application locally first and go to **https://localhost:8080**

```
npm run dev
```

Image version

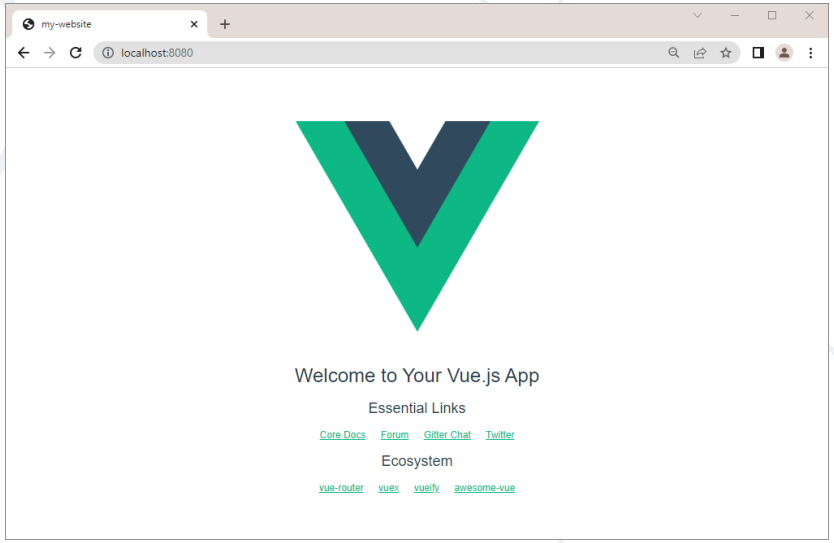
```
PS C:\Users\     \MyWebsite> npm run dev

> my-website@0.1.0 dev
> npm-run-all --parallel watchify serve


> my-website@0.1.0 watchify
> watchify -vd -p browserify-hmr -e src/main.js -o dist/build.js


> my-website@0.1.0 serve
> http-server -o -c 1 -a localhost

Starting up http-server, serving ./
Available on:
  http://localhost:8080
Hit CTRL-C to stop the server
```



Run the App In a Container

- Run container with **interactive shell** to enter inside it
- Map **external port 8080** to **internal port 8080**
- Create **volume** and map current directory to container's **/app** directory
- Set **/app** as working directory when container is started
- Inside the container, run the **Vue.js app**

It may be any directory

```
docker run -it `
  -p 8080:8080 `
  -v ${PWD}:/app `
  -w /app `
  node:16
  npm run serve
```

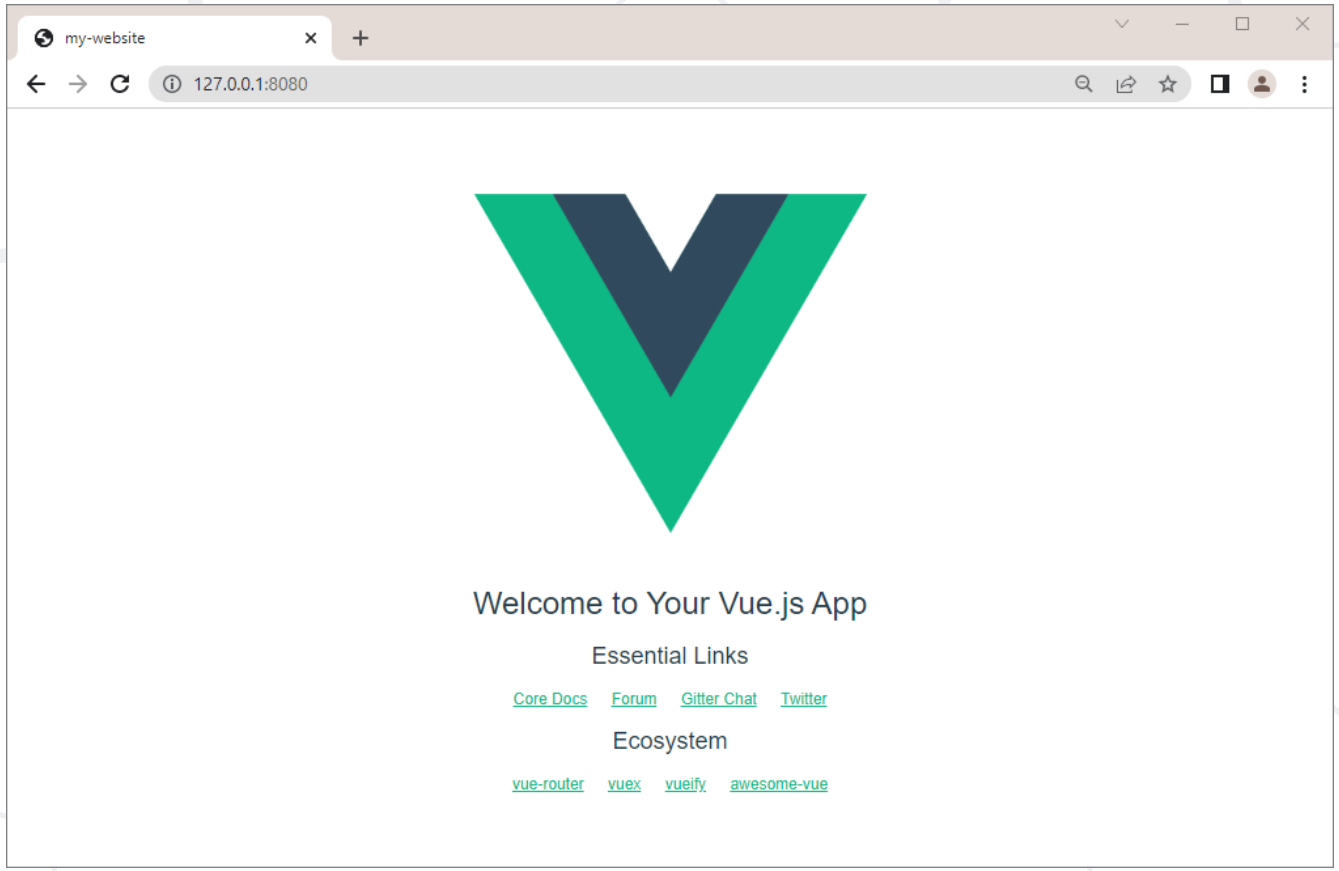
Image for container

```
PS C:\Users\     \MyWebsite> docker run -it `
>> -p 8080:8080 `
>> -v ${PWD}:/app `
>> -w /app `
>> node:16 `
>> npm run serve
```

You can chain commands

Run the App In a Container – Result

- The app should be built and started
- Go to **127.0.0.1:8080** to validate it is running



Database in a Container

Docker Container with MongoDB

MongoDB Image

- Pull latest **MongoDB image**

```
docker pull mongo
```

- Examine the **documentation** on how to use the image

https://hub.docker.com/_/mongo

```
PS C:\Users\     \MyWebsite> docker pull mongo
Using default tag: latest
latest: Pulling from library/mongo
1bc677758ad7: Pull complete
7eb83bb7be98: Pull complete
e95121721c4c: Pull complete
799041b403ca: Pull complete
1828e70ef29a: Pull complete
8e3781beae9e: Pull complete
5d5753162333: Pull complete
44dd404b40f4: Pull complete
44599c9d5d1b: Pull complete
Digest: sha256:928347070dc089a596f869a22a4204c0feace3eb03470a6a2de6814f11fb7309
Status: Downloaded newer image for mongo:latest
docker.io/library/mongo:latest
```

Run a Database Container

- **Run the container** with the following command

```
docker run \
> -p 27017:27017 \
> -v ${PWD}/data:/etc/mongo \
> -d \
> -e MONGO_INITDB_ROOT_USERNAME=mongoadmin \
> -e MONGO_INITDB_ROOT_PASSWORD=mongoadminpass \
> mongo
```

You can connect on **localhost:27017**

Set **admin username**

Set **admin password**

- **Disable** host's **MongoDB Server instances** or **use another port**!
- **Admin password** should follow rules from documentation
- When MongoDB Server container is started, other apps can log in to it and use the database

Run a Database Container with Volume

- To persist data after container is stopped, **create a volume**

```
PS C:\Users\vikto> docker run `
>> -p 27017:27017 `
>> -v data:/etc/mongo `
>> -d `
>> -e MONGO_INITDB_ROOT_USERNAME=mongoadmin `
>> -e MONGO_INITDB_ROOT_PASSWORD=mongoadminpass `
>> mongo
```

```
docker run \
> -p 27017:27017 \
> -v data:/etc/mongo \
> -d \
> -e MONGO_INITDB_ROOT_USERNAME=mongoadmin \
> -e MONGO_INITDB_ROOT_PASSWORD=mongoadminpass \
> mongo
```

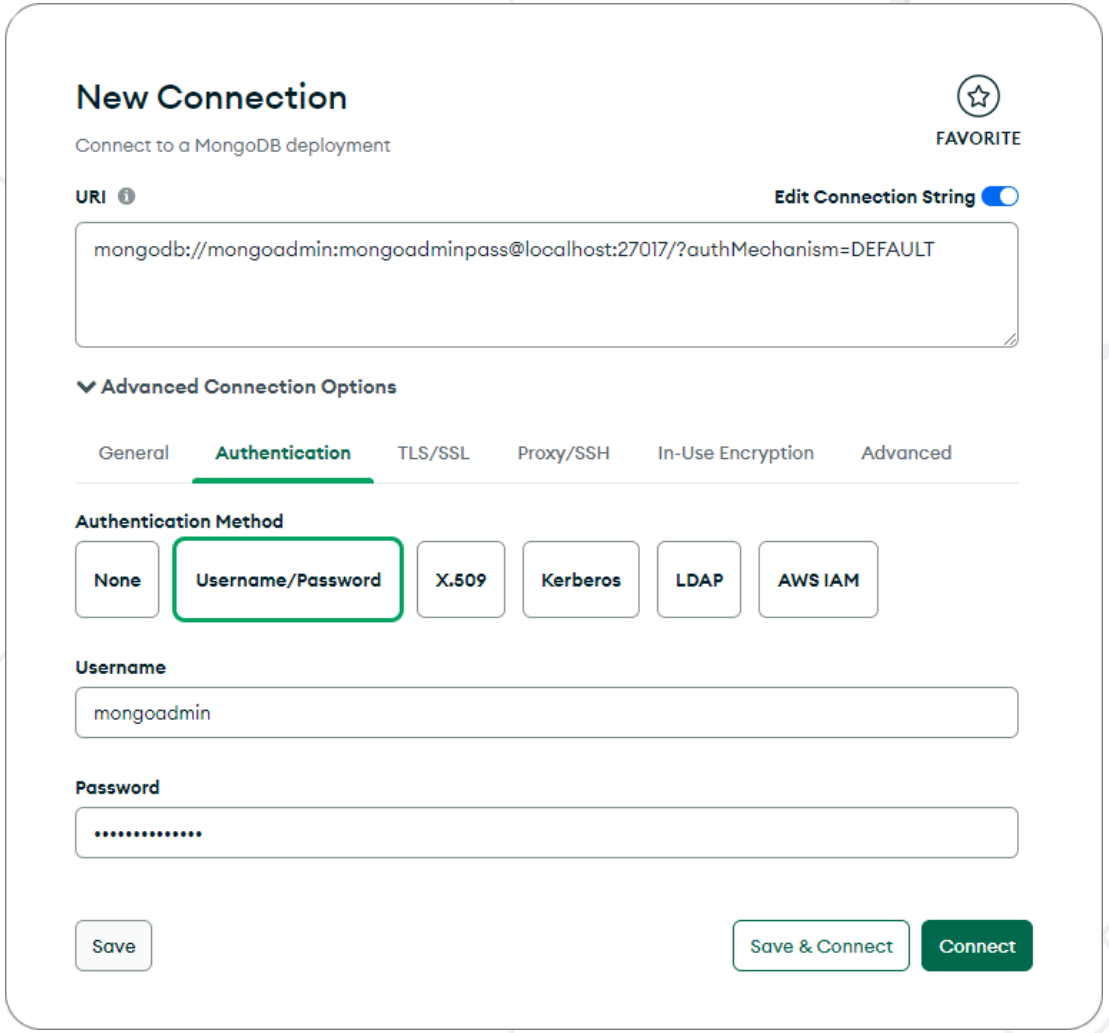
Volume name

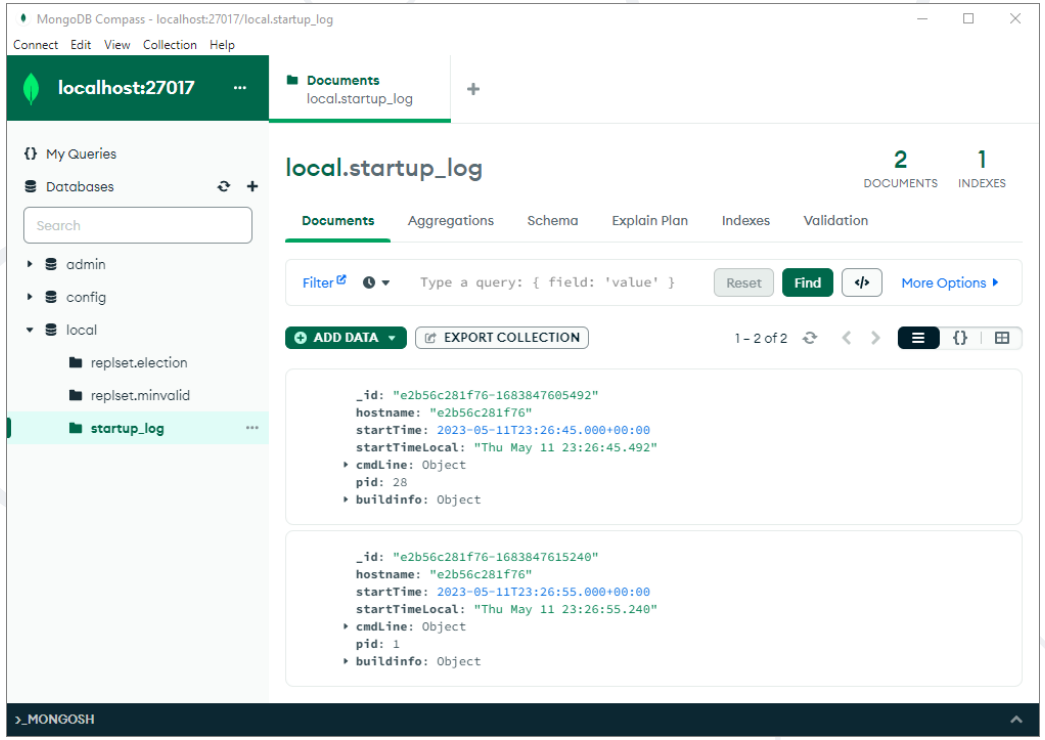
All MongoDB Server data is in /etc/mongo

- You can then easily **backup** or **restore** the **data** from the volume

Container's Database in MongoDB Compass

- You can connect to the container database in **MongoDB Compass**

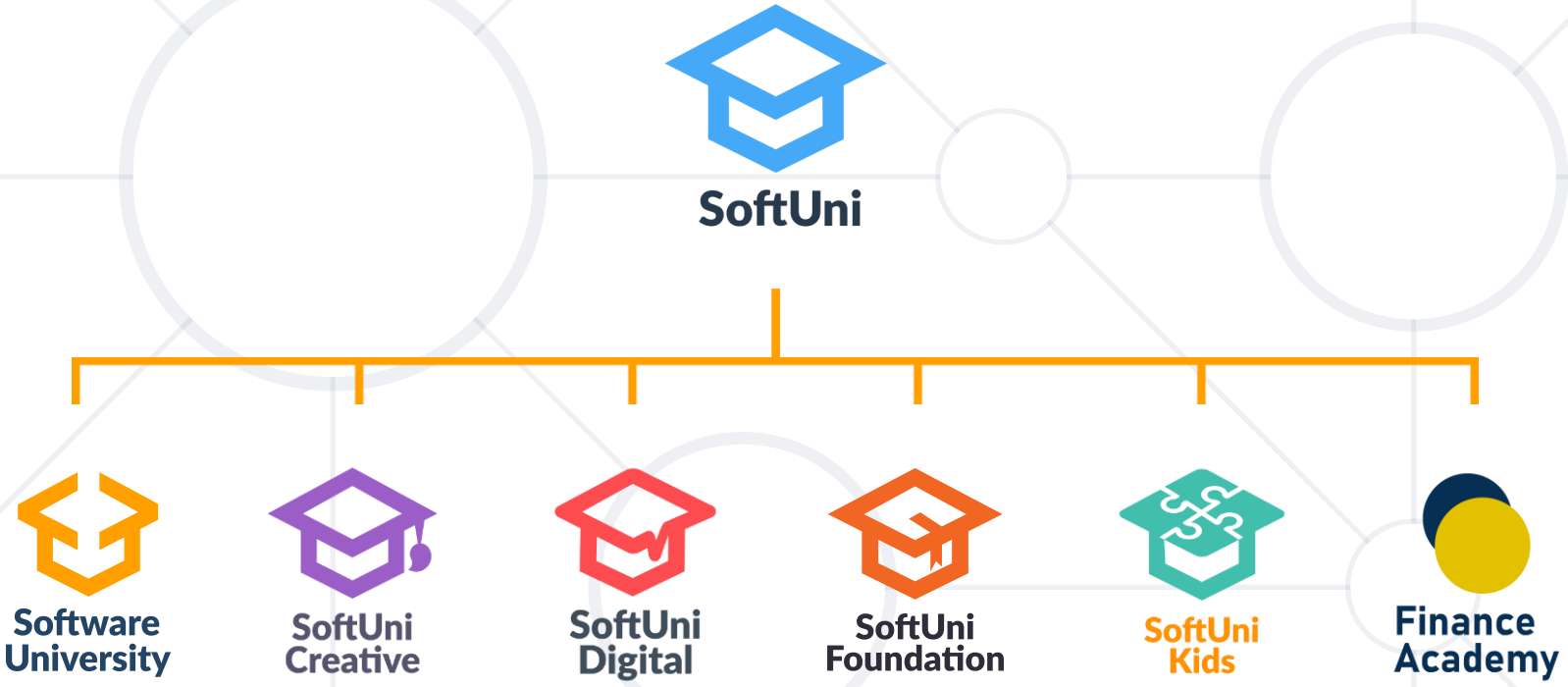




Summary

- With **Docker** we can create and manage **images**, **containers**, **volumes**, etc.
  - **Image** == read-only template with instructions for creating a Docker container
  - **Container** == a runnable instance of an image
  - **Volumes** == the preferred mechanism for persisting data
- We can **run apps in containers**
- We can also have a working **database in a container**

Questions?



SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

POKERSTARS
POKER | CASINO | SPORTS
a Flutter International brand

INDEAVR
Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

createX

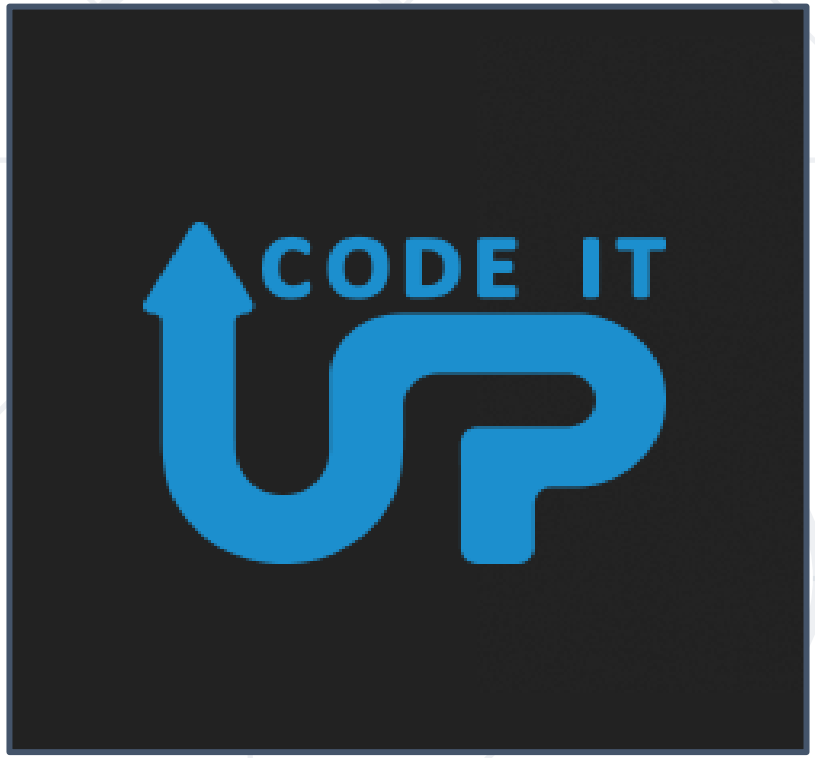
Postbank
Решения за твоето утре

BOSCH

DXC TECHNOLOGY

SmartIT

Educational Partners



License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg