DOCKER

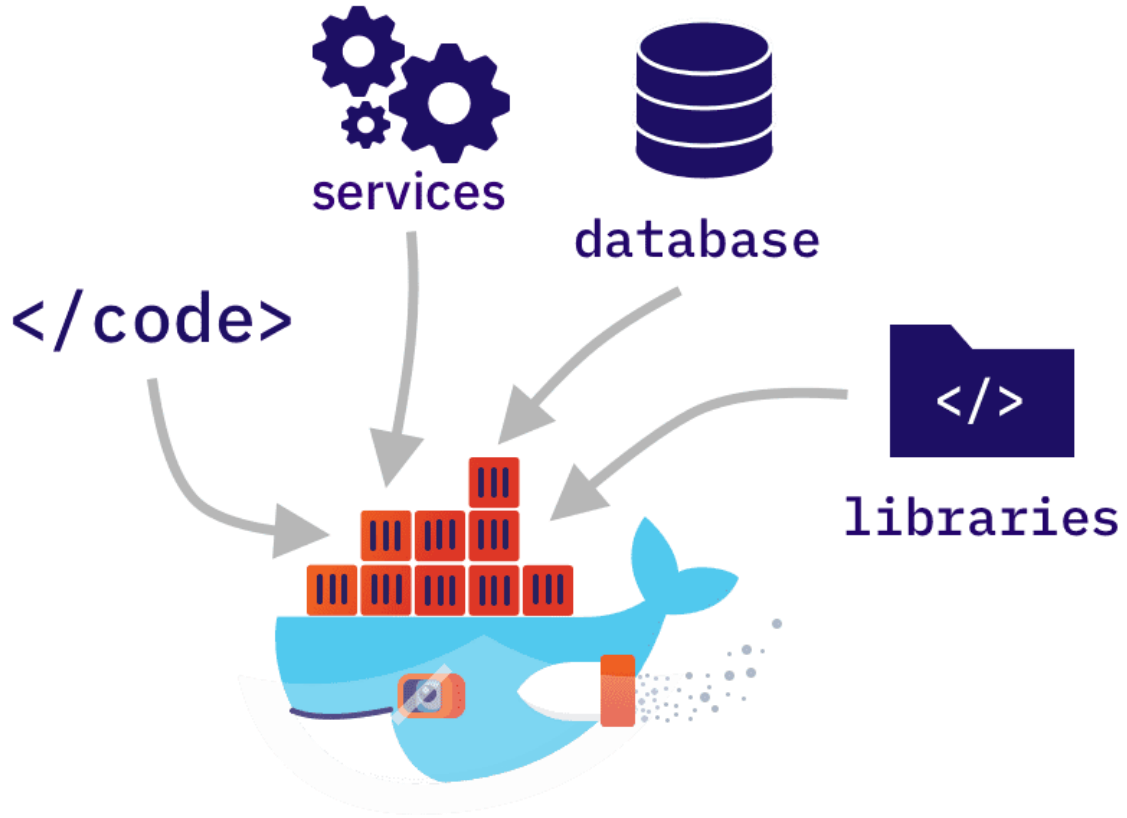
Docker - Introduction

- Tool that allows developers to **bundle all the ingredients of software they create into unified boxes** - called Docker containers.

- The key concern is usually the code, but most apps or websites won’t work without a database, configuration files, runtime libraries or other third-party software.

- Putting these together in a box makes it **easier to launch the app on another developer's computer or server**.

Docker



Docker - History

- Docker was originally created by Solomon Hykes from dotCloud Inc. (later Docker Inc.) as an open-source project and since it's launch in 2013 Docker is available for free.

- In March 2017 the open-source version of Docker was rebranded to Docker CE (Community Edition) and Docker Enterprise Edition (EE) was released.

Docker File

- Docker builds images automatically by reading the instructions from a Dockerfile which is a text file that contains all commands, in order, needed to build a given image.

- A Dockerfile adheres to a specific format and set of instructions which you can find at Dockerfile reference.

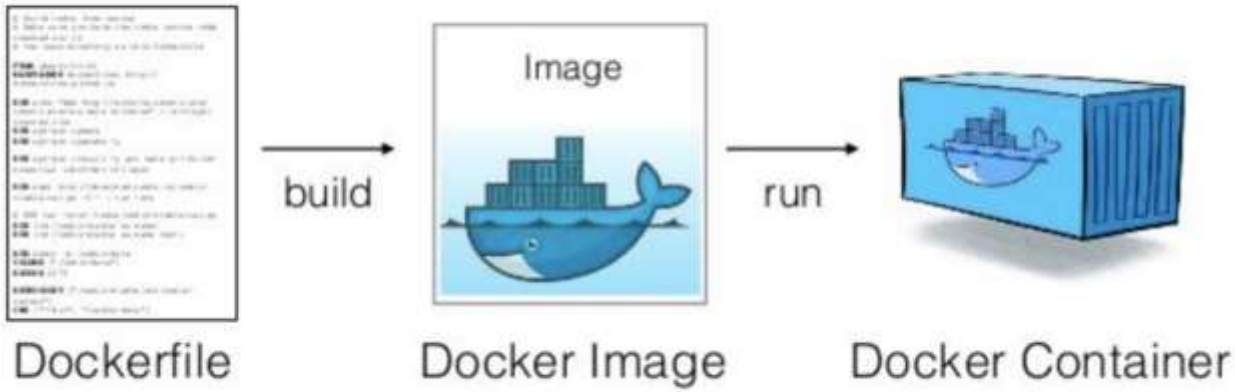
Docker File

Dockerfile - myhttpd:0.1

```
1  # A simple web app served by httpd
2  FROM httpd:2.4
3
4  LABEL AUTHOR=user@example.com
5
6  LABEL VERSION=0.1
7
8  # COPY mypage.html /usr/local/apache2/htdocs/mypage.html
9  # WORKDIR /usr/local/apache2
10
11 COPY mypage.html htdocs/mypage.html
```

Docker - Images

- Docker images are a lightweight, standalone, executable package of software that includes everything needed to run an application, code, runtime, system tools, system libraries and settings.



Docker Container

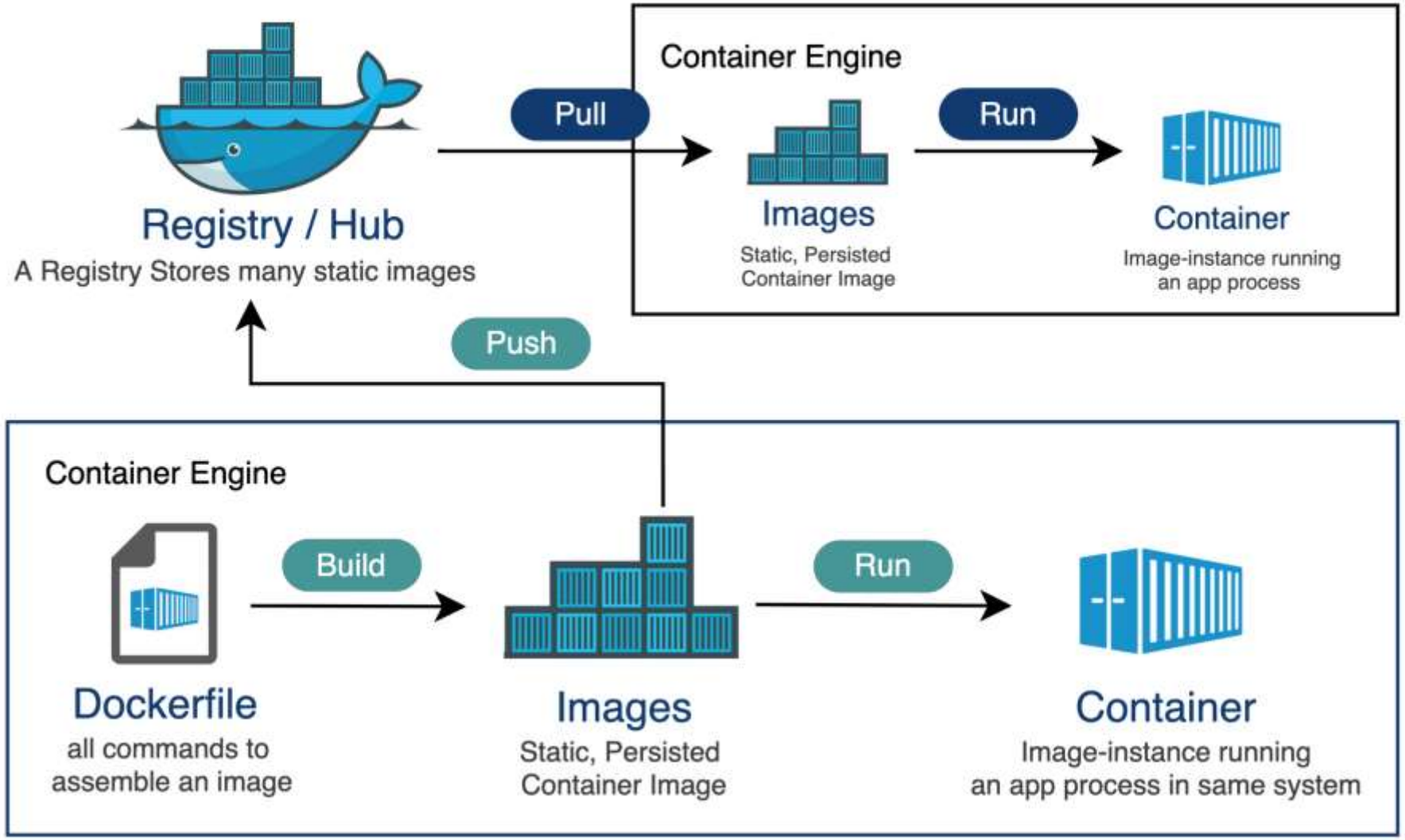
- A container is a runtime instance of a docker image. A container will always run the same, regardless of the infrastructure.

- Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging.

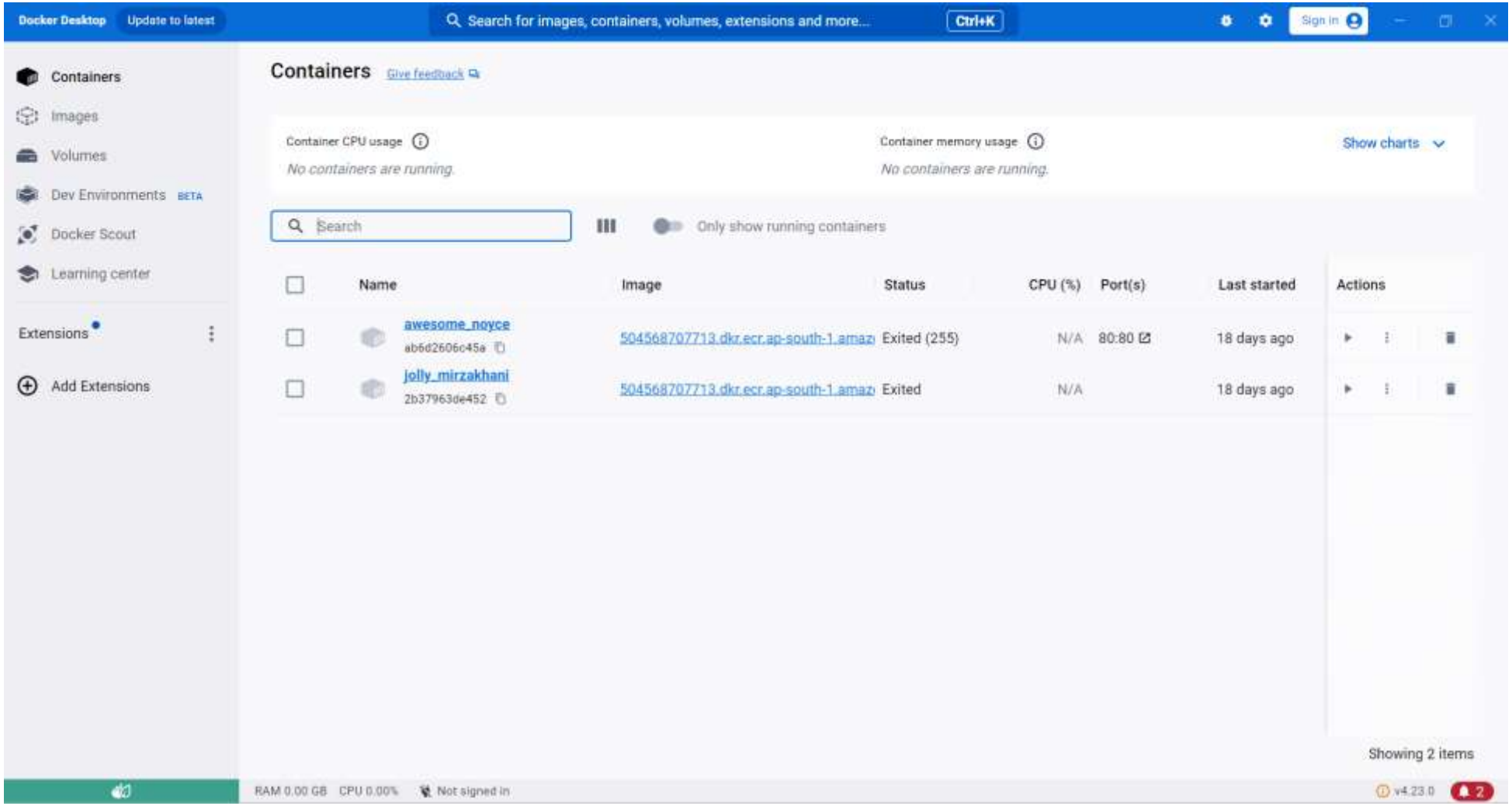
Docker Hub

- Docker Hub is a service provided by Docker for finding and sharing container images with team.
- Readily available images can be found at https://hub.docker.com

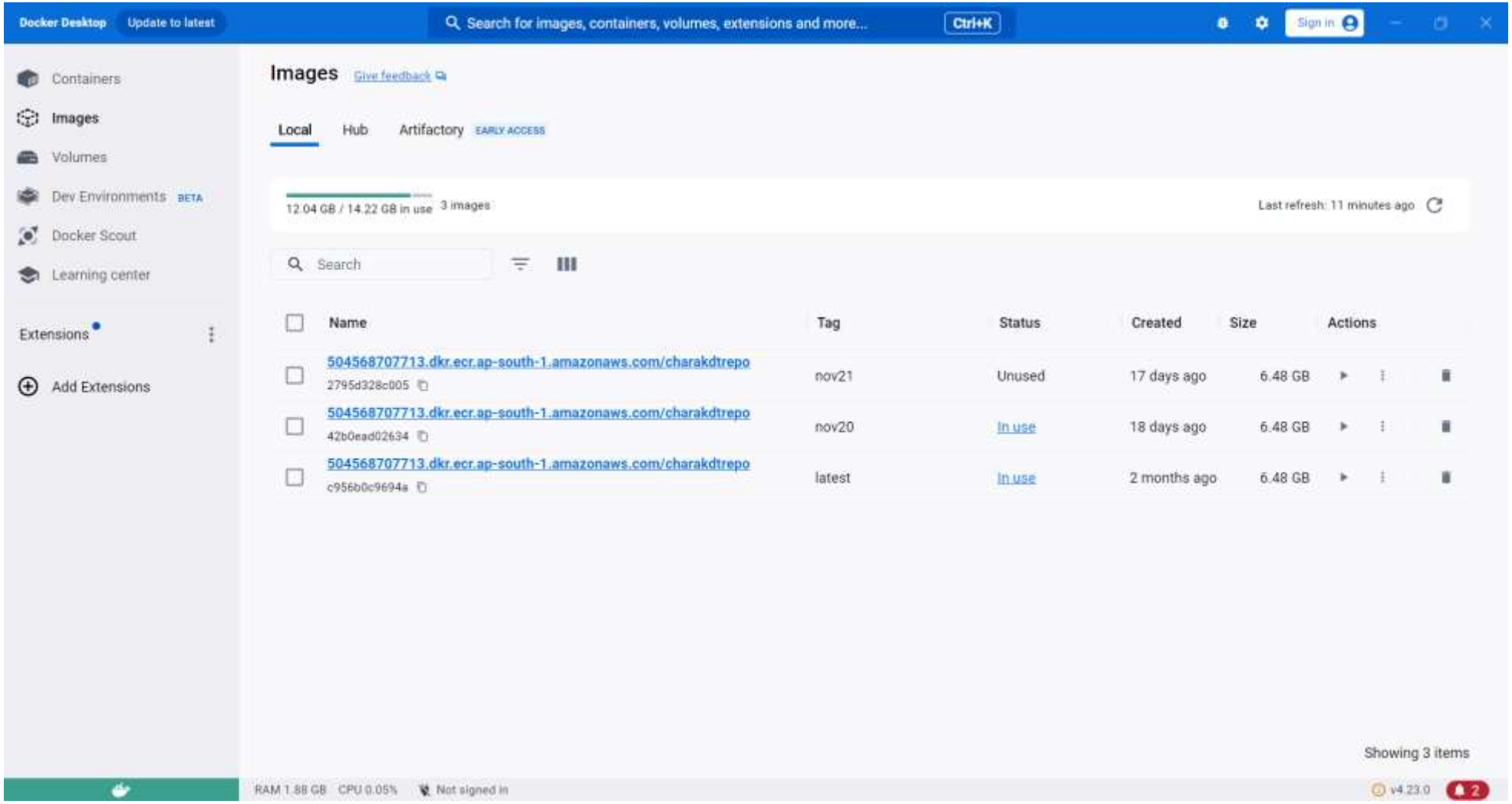
Putting all Docker terminologies together



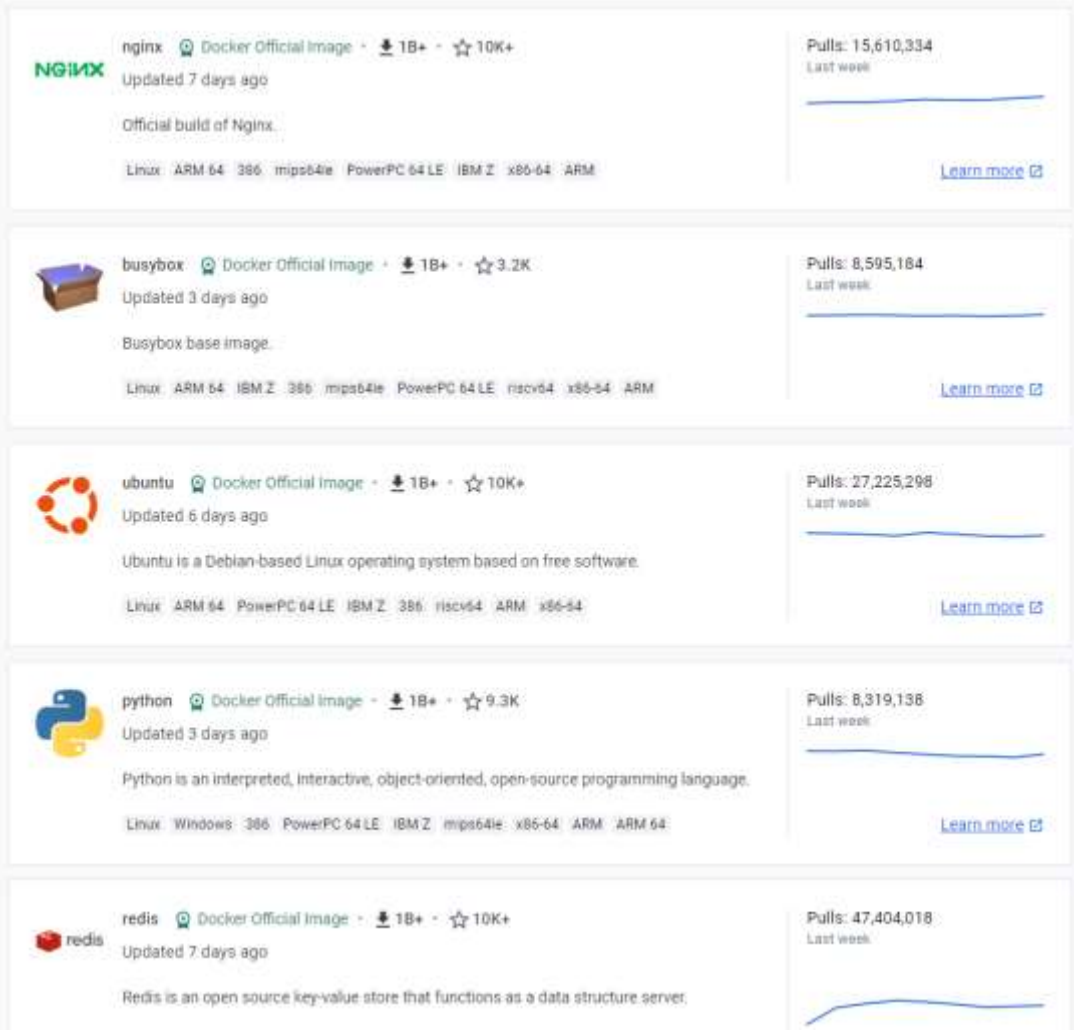
Docker Desktop - Containers



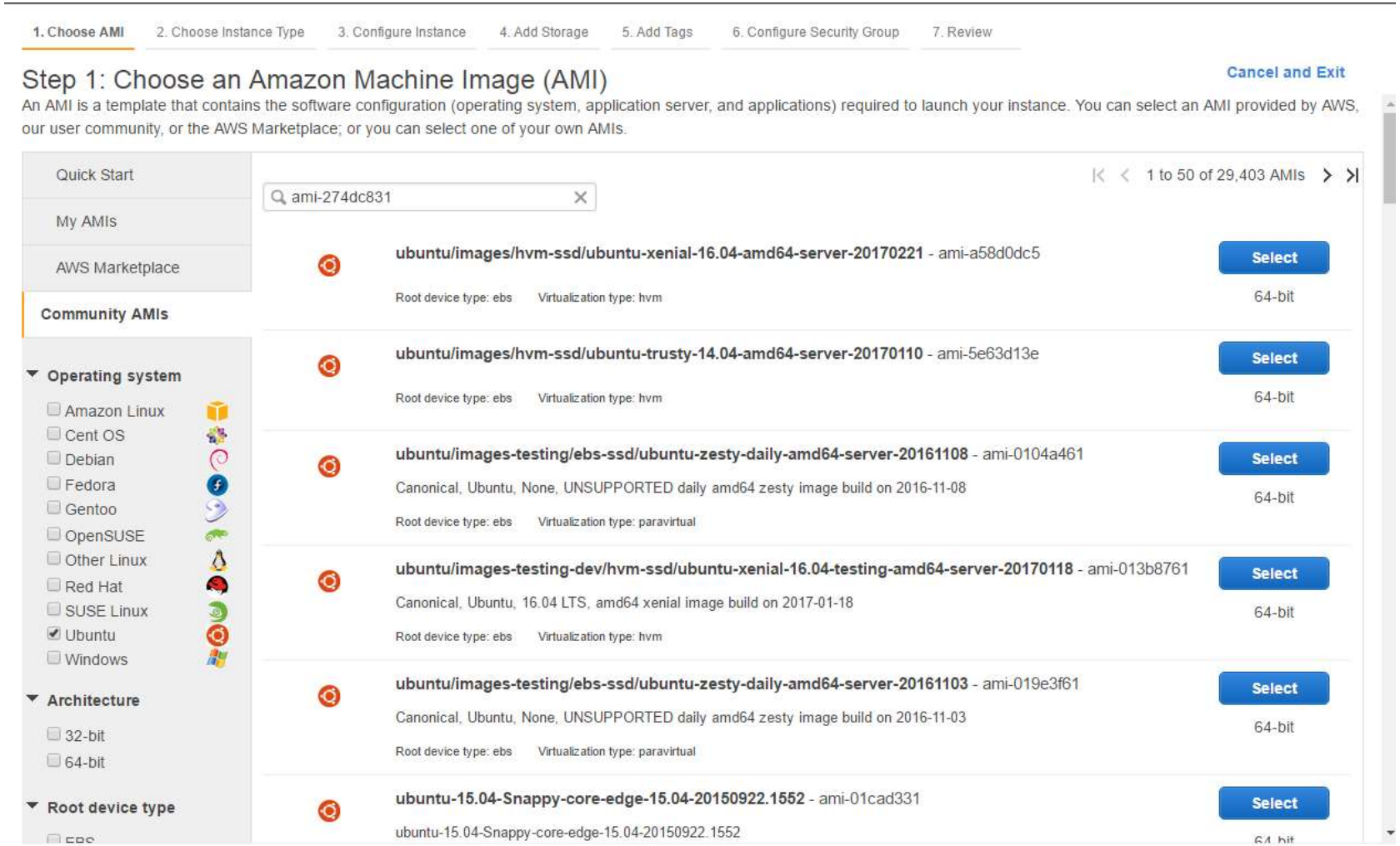
Docker Desktop - Images



Docker Desktop - Images



Amazon Machine Images



Yet another analogy with a difference

Docker Image

Docker Container

Docker - Commands - Images

- **Build an Image from a Dockerfile**

  docker build -t <image_name>
- **Build an Image from a Dockerfile without the cache**

  docker build -t <image_name> . –no-cache
- **List local images**

  docker images
- **Delete an Image**

  docker rmi <image_name>
- **Remove all unused images**

  docker image prune

Docker - Commands - Containers

- **Create and run a container from an image, with a custom name:**
  docker run --name <container_name> <image_name>
- **Run a container with and publish a container's port(s) to the host.**
  docker run -p <host_port>:<container_port> <image_name>
- **Run a container in the background**
  docker run -d <image_name>
- **Start or stop an existing container:**
  docker start|stop <container_name> (or <container-id>)
- **Remove a stopped container:**
  docker rm <container_name>
- **Open a shell inside a running container:**
  docker exec -it <container_name> sh
- **Fetch and follow the logs of a container:**
  docker logs -f <container_name>
- **To inspect a running container:**
  docker inspect <container_name> (or <container_id>)
- **To list currently running containers:**
  docker ps
- **List all docker containers (running and stopped):**
  docker ps --all
- **View resource usage stats**
  docker container stats

Commands - Docker Hub

- **Login into Docker**

  docker login -u <username>
- **Publish an image to Docker Hub**

  docker push <username>/<image_name>
- **Search Hub for an image**

  docker search <image_name>
- **Pull an image from a Docker Hub**

  docker pull <image_name>

Docker - General Commands

- **Start the docker daemon**

  docker -d

- **Get help on Docker. Can also use –help on all subcommands**

  docker --help

- **Display system-wide information**

  docker info

IS DOCKER A ONE-SIZE-FITS-ALL SOLUTION?

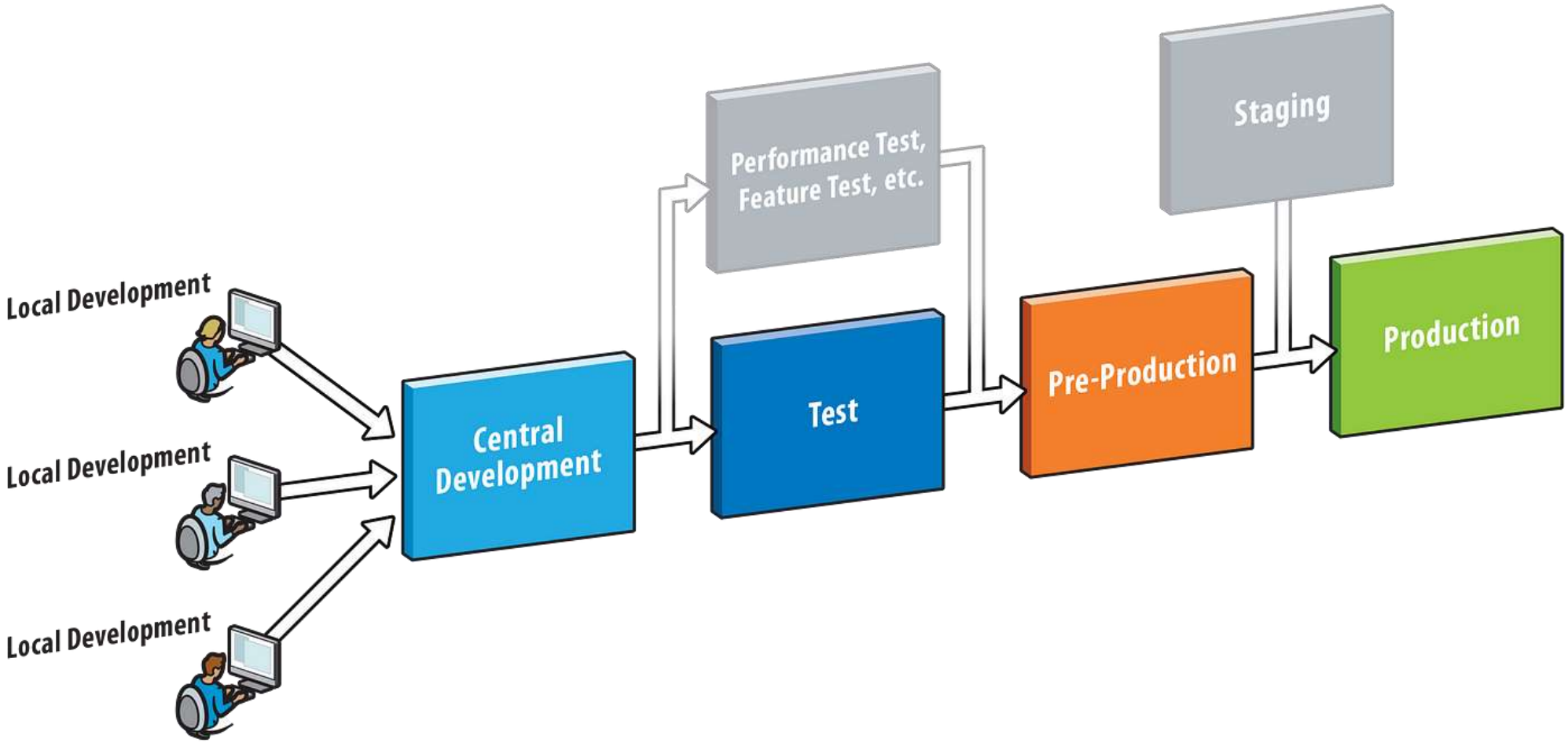
When to Use Docker?

- Development team changes often
- Software runs in different environment
- Software grows and consists of many bits and pieces
- Expecting the software to be scalable and handle more users
- Hosting may change in future and if don’t want to be locked
- Test new technologies for software

Development team changes often

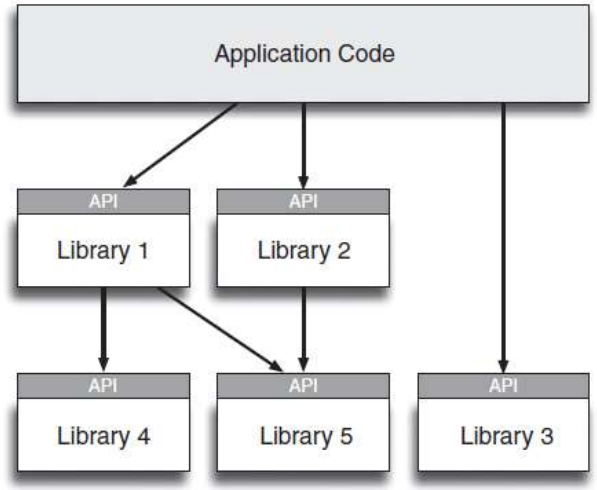
- New developers need to setup their local development environment for the project - e.g. local server, database, or third-party libraries etc.
- It may take hours to many days, depending on complexity and how well is the project setup manually.
- Docker automates this by simply running one command that will prepare the development environment.

Software Runs in Different Env



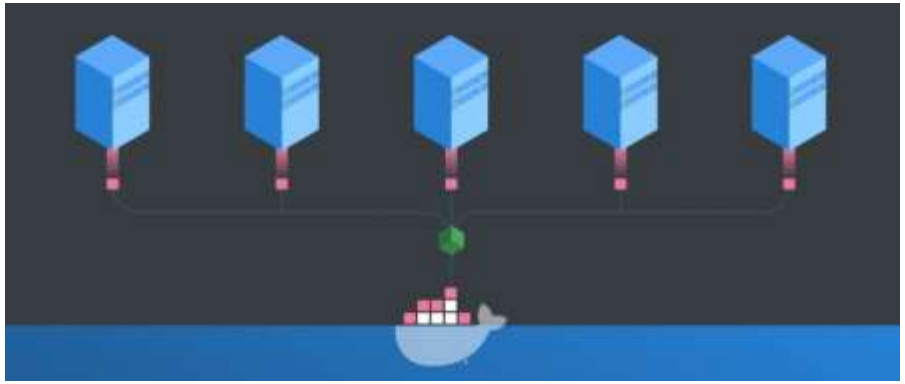
Software Consists of Many Parts

- Developers add new libraries, services, and other dependencies to the software daily.
- All the changes in project setup have to be communicated properly.
- With Docker, all requirements are there in configuration file and if something is specified to this configuration, it is automatically added for every developer working on the project.



Need Software to be Scalable

- Docker containers can be launched in many copies that may run in parallel.
- Docker compose can be used for this.



Docker Compose

```
FROM node:alpine
WORKDIR /backend/
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD [ "node", "app.js" ]
```

```
docker build -t backend .
```

```
docker run -p 3000:3000 backend
```

Backend Image

```
# Use an official Node.js runtime as the base image
FROM node:alpine
RUN npm install -g http-server
# Set the working directory in the container
WORKDIR /frontend
COPY package*.json ./
# Install the application dependencies
RUN npm install
# Copy the entire application to the container
COPY . .
RUN npm run build
EXPOSE 8080
CMD [ "http-server", "dist" ]
```

```
docker build -t frontend .
```

Frontend Image

```
version: "3"
services:
  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
    ports:
      - 3000:3000
  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    ports:
      - 8080:8080
```

```
docker compose build
```

Docker Compose

Hosting may Change in Future

- When a business grows and evolves, so do the server requirements. Also, in fast paced business environment, web infrastructure needs to be flexible enough to adapt quickly.

Test New Technologies for Software

- Docker is quite useful if you want to experiment with different technologies. Want to adopt some new database, programming language, or other tools? Most probably there is already a Docker template for this (called Docker Image) created by a docker community.
- It is especially easy if you use one of the most popular Docker tools - docker compose. It brings all the project infrastructure to just a single file and run docker containers for you. No need for time-consuming installation or debugging compatibility issues.

When not to Use Docker?

- Software product is a desktop application
- Software project is relatively small and simple
- Team consists of only one developer
- Lack of skilled resources for Docker
- If want to speed up the application
- If security is on priority

Need to Speedup the Application

- Docker may speed up your development process significantly, but not necessarily your app itself.
- Although it helps with making your application scalable, so more users will be able to use it, the single instance of your app will usually be just a hint slower than without Docker.

If Security is Priority

- While isolated processes in containers promise improved security, all containers share access to a single host operating system. You risk running Docker containers with incomplete isolation. Any malicious code can get access to your computer memory.