* **Index**
  + [**Docker Overview**](#dockeroverview)
    - Docker
    - Images
    - Containers
    - Docker Daemon
    - Docker client
    - Docker hub
    - Dockerfile
    - Dockerfile example
    - Intro to Docker
  + [**Dockers vs Virtual machines.**](#dockervsvms)
  + [**Install Docker on Ubuntu.**](#installdockeronubuntu)
  + [**Install docker on Windows.**](#installdockeronwindows)
  + [**Essential Docker Commands**](#essentaildockercommands)
  + [**Deploy sample Nodejs Backend Application steps.**](#deploynodejsappsteps)
  + [**Some Issues and considerations**](#someissues)
    - Docker and Virtual Machine cannot work at the same time!
    - Fix Issue Container Backend App cannot connect to mongodb server**.**
    - Fix Issue cannot access Container Backend App endpoints.
    - Fix Issue cannot read environment variables in .env file in Container Nodejs Backend App.
    - Fix Issue bycrpt package not working.
    - Check if port is open in Ubuntu.
    - Add inbound rule for port in Ubuntu.
* Docker Overview

**Docker:** is a tool that allows developers and system admins to easily deploy their applications

in a sandbox (called containers) to run on the host operating system.

* It’s also a way to package applications with all necessary dependencies and configurations.
* makes development & deployment more efficient.

**Containers** - Created from Docker images and run the actual application. We create a container using docker run command.

* Container is a layers of images.

**Images** - The blueprints of our application which form the basis of containers.

**Docker Daemon** - The background service running on the host that manages building, running and distributing Docker containers.

* The daemon is the process that runs in the operating system which clients talk to.

**Docker Client** - The command line tool that allows the user to interact with the daemon.

**Docker Hub** - A registry of Docker images.

* You can think of the registry as a directory of all available Docker images.
* If required, one can host their own Docker registries and can use them for pulling images.
* images on docker hub can be public or private.

**Dockerfile -** is a plaintext configuration file used to define the specifications and instructions for building a Docker container image.

* When build Dockerfile its generates a docker Image.
* when modify Dockerfile you should build the image again.

**Dockerfile example**

FROM node:version

RUN npm install -g npm@latest

WORKDIR /home/app

# from source . to destination .

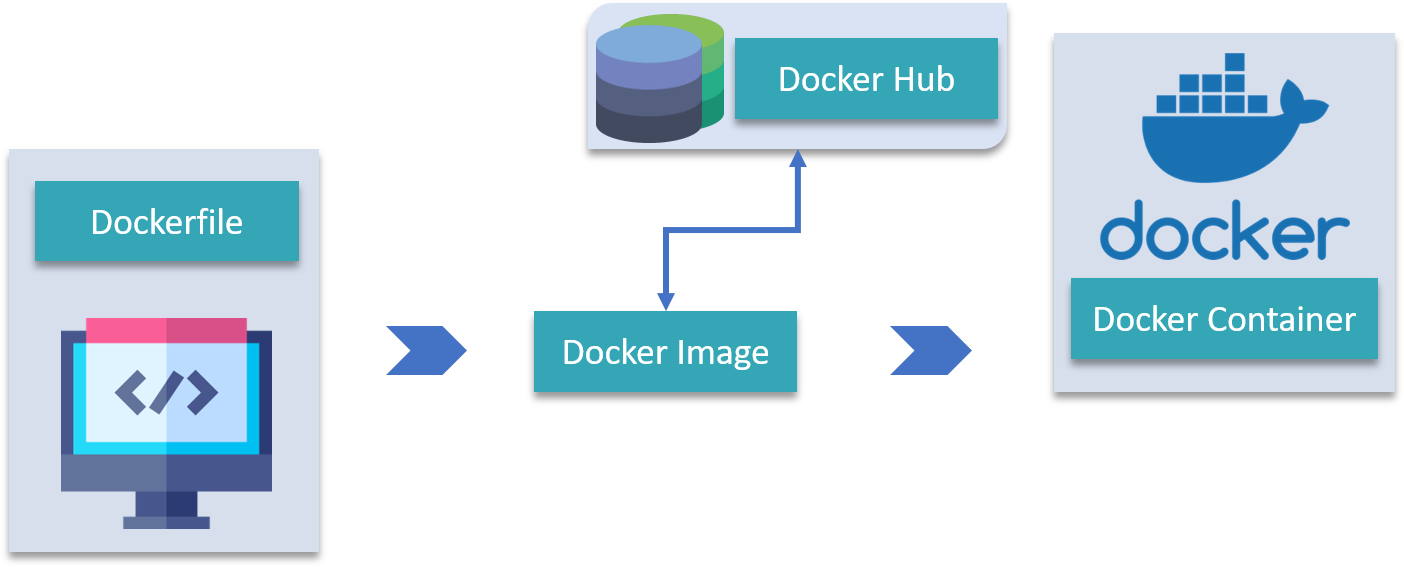
COPY . .

EXPOSE 3000

# refere to /home/app/ where server.js located

CMD ["node","/home/app/server.js"]

- Docker Container creation steps



**Intro to Docker**

* You can make different versions of the same application running without any conflict.
* Containers include (dependencies, application, configurations).
* no configurations needed on the server except (docker runtime)
* Containers offer a logical packaging mechanism in which applications can be abstracted (isolated) from the environment in which they run,

This decoupling allows container-based applications to be deployed easily and consistently.

* Container is a running environment of Image.
* When run image it becomes a container.
* Port Binding
* make sure you don't bind two containers on the same port, but 2 containers can have the same port internally.
* talk to application running inside a container.

**example:**

sudo docker run -d -p 3000:3000 --name forum-app c7e3e0069436

* Docker vs Virtual Machines

|  |  |  |
| --- | --- | --- |
|  | **Dockers** | **Virtual machines** |
| **Computational resources** | containers provide most of the isolation of virtual machines at a fraction of the computing power. | VMs comes at great cost — the computational overhead spent virtualizing hardware for a guest OS to use is substantial. |
| **Virtualization** | Docker virtualize applications Layer. | VMs virtualizes Kernal (OS) and applications |
| **Architecture** | Isolated environments that Share the host os kernel. | VMs use hardware virtualization technology. They run a complete guest operating system on top of a hypervisor, which sits on the host hardware. |
| **Size** | Docker images size is much smaller (Megabytes) |  |
| **Performance & Speed** | Docker containers start and run much faster. | VMs require more resources as they emulate a full OS, VMs can be slower to start and consume more memory and storage. |
| **Compatibility** | Each container can run on a single operating system. | VM of any OS can run on any OS host |
| **Use Cases** | Docker is well-suited for microservices architectures, continuous integration/continuous deployment (CI/CD) pipelines, and packaging applications with their dependencies. | VMs are useful for running applications that require different OS environments, legacy applications, or scenarios where strong isolation is necessary. |
| **Portability** | Docker containers are highly portable and can run consistently across different environments, such as development, testing, and production, as long as the host system supports Docker. | VMs can be less portable due to dependencies on specific hypervisors and configurations. Moving VMs between different virtualization platforms can be more challenging. |
| **Isolation** | Containers provide process-level isolation. While they are isolated from each other, they share the same OS kernel | VMs offer stronger isolation because each VM runs its own OS. |

* Install docker on Ubuntu.

<https://www.simplilearn.com/tutorials/docker-tutorial/how-to-install-docker-on-ubuntu>

1- Open the terminal on Ubuntu.

2- Remove any Docker files that are running in the system

sudo apt-get remove docker docker-engine docker.io

3- Check if the system is up to date.

sudo apt-get update

4- Install Docker.

sudo apt install docker.io

5- Install all the dependency packages.

sudo snap install docker

6- Make sure Docker is Installed and see docker version.

 sudo docker –version

* Install docker on Windows.
* download and install docker from:

<https://docs.docker.com/desktop/install/windows-install/>

* + - follow installation wizard.
    - enable and install WSL on your windows.

 wsl –install

* Essential Docker Commands

docker run hello-world

* run this command in terminal to check if docker is installed successfully.
* If the image hello-world is not exist locally, docker pulls it first from docker hub then run it.
* **Output:**

Hello from Docker This message shows that your installation appears to be working correctly.

docker pull busybox

* + - **Pull**: pulls image from container registry ex: docker hub
    - **Busybox**: image name.

docker run busybox

1- run docker container using busybox pulled image.

2- makes the container (image) run in the background.

Run will pull the image from [docker hub](https://hub.docker.com/search) if it doesn’t exit.

docker images

* to see a list of all images on your system.

docker ps

* command shows you all containers that are currently running.

docker ps -a

* list all containers that previously ran (exited).

docker run -it busybox /bin/bash// or /bin/sh

* + -it : to interactively execute commands available in this container, ex: ls, uptime, …
  + Write exit to exit container interactive commands.

docker rm 8f27b0228131 1a3d16fb59ef

* + to remove a container/s
  + you can enter container name or id.

docker rmi 8f27b0228131 1a3d16fb59ef

* + to remove a image/s
  + enter image id.
    - **Note:** Remove containers and images continuously in order to no to eat up disk space.

docker rm $(docker ps -a -q -f status=exited)

* + delete all non-running containers at once.
  + -q flag, only returns the numeric IDs.
  + -f filters output based on conditions provided.

docker container prune

* + Also to delete all containers at once.

Docker -d run busybox

* + -d means run the container in detached mode.
  + it means that you want to run a Docker container in the background,

detached from the terminal where you executed the command.

* + This allows you to continue using your terminal for other tasks without being tied to the container's console.
  + but you can still interact with the container by using other Docker commands (docker exec, docker logs)

docker run --rm -it prakhar1989/static-site

--rm remove the image after running container from it.

-it specifies an interactive terminal

docker run -d -P --name static-site prakhar1989/static-site

--name static-site to set container name

-P will publish all exposed ports to random ports.

docker port static-site

* + to see this container exposed ports.

docker run -p 8888:80 prakhar1989/static-site

* + bind container to another port.

docker start static-site

docker stop static-site

* + stop the running container, followed by container name or ID.

docker logs static-site

* + to start the container again, followed by container name or ID.
  + to see the logs of this container.

docker pull ubuntu:18.04

* + pull a specific version of image from docker hub
  + **Note:** images can be committed with changes and have multiple versions.

docker stop adoring\_jang | docker rm adoring\_jang | docker rmi d083aee31bd2

* + you can chain commands like this in order to do many things at once here we (stop the container ,delete it, remove its image).

docker build -t app-name:version

**-t** to specify name:tag

- You should first change directory in terminal to project root folder where Dockerfile is located.

docker ps -a | grep my-app

* to filter command output, show my-app line only.
* Deploy Nodejs App steps.
  1. Open terminal
  2. Change directory to project folder
  3. Create Dockerfile file with name `Dockerfile` without any extensions in the root directory of the project, in Dockerfile we write code to describe our images and specify app dependencies and configurations.
  4. We can now build docker image from Dockerfile.

docker build -t forum:1.0 .// name:version

* 1. Get the created Image ID
  + Execute this command to get image ID.
  1. Run the container from image using Image ID from previous step

docker images

docker run -d -p 3000:3000 --name forum-app c7e3e0069436

* + **run** to run container from image.
  + **-p 3000:3000** bind image port to host port 3000
  + **--name forum-app**  set the name of the container
  + **-d** run container in detached mode in the background in order to continue in typing in terminal.
  1. Check if created container is running

docker logs ContainerID/Name // `Name` you set in step 6

docker ps

* 1. To debug container and see logs
* Some Issues and considerations
* Docker and Virtual Machine cannot work at the same time!
* In order to make any of them work you have to change some windows configurations
* **Docker**

**The problem is**: Docker Desktop is unable to detect a Hypervisor.

* + **Enable windows HyberV:**
    - * Open Control panel.
      * Click `Uninstall programs`.
      * then click on `turn windows features on or off`.
      * Turn `Hyber V` feature on.
  + **Turn on Virtualization based security:**
    - Search `Edit group policy` in start menu.
    - Click `Computer Configuration` -> Administrative templates

-> System -> Device Guard -> Turn on Virtualization based security

* + - Double click on it and set the value `Not Configured`
  + **Execute this Cmd command as Administrator:**

bcdedit /set hypervisorlaunchtype auto

* + **Then Restart the computer.**
* **Virtual Machine**

**The problem is:** VMware workstation device credential guard are not compatible.

1. **Disable windows HyberV:**
   * + - Open Control panel.
       - Click `Uninstall programs`.
       - then click on `turn windows features on or off`.
       - Turn `Hyber V` feature off.
2. **Turn on Virtualization based security:**

Search `Edit group policy` in start menu.

Click `

Computer Configuration` -> Administrative templates

-> System -> Device Guard -> Turn on Virtualization based security

Double click on it and set the value `Disabled`

1. **Then Restart the computer.**

* Fix Issue Container Backend App cannot connect to mongodb server.

- to access Mongodb

* + go to C:\Program Files\MongoDB\Server\6.0\bin
  + and edit `mongod.cfg` file then change `127.0.0.1` to `0.0.0.0`
* Fix Issue cannot access Container Backend App endpoints.
  + In Nodejs/Expressjs app you should listen to port and 0.0.0.0 URL to be available to all interfaces.

app.listen(/\*PORT\*/ 3000, "0.0.0.0", () =>

console.log("Express server has started...")

);

* Fix Issue cannot read environment variables in .env file in Container Nodejs Backend App.
  + - In Dockerfile do this to copy the files in order to include hidden file such as .env

WORKDIR /home/app

COPY . .

* Fix Issue bycrpt package not working.
* You can use bcryptjs instead , which is a pure JavaScript package and doesn’t need any dependencies.
* Check if port is open in Ubuntu.

**- using `nmap`**

// Run these commands in Ubuntu terminal

// Install nmap

sudo apt-get install nmap

// SYNTAX: nmap -p <port> <hostname\_or\_ip>

// Example

nmap -p 80 google.com

* + it will show port status like : (Open, Filtered , Closed)
* Add inbound rule for port in Ubuntu.

// Run these commands in Ubuntu terminal

// 1

sudo apt-get install ufw

// 2

sudo ufw enable

// 3- Example (enable inbound rule for port 80)

sudo ufw allow 80/tcp

// 4 – to verify the rules has added

sudo ufw status

// 5- you might need to refresh the rules

sudo ufw reload

* Add outbound rule for the port

// SYNTAX: sudo ufw allow out <port>/tcp

// Example

sudo ufw allow out 80/tcp