Docker:

Build, Ship, Run, Any App, Anywhere

https://www.simplilearn.com/tutorials/docker-tutorial/docker-vs-virtual-machine

Docker:
Docker is a container-based model where containers are software packages used for executing an application on any operating system
In Docker, the containers share the host OS kernel . Here, multiple workloads can run on a single OS
It can start multiple containers at a time on the Docker engine

Virtual Machine: (Traditional approach of shipping and deploying the software packages) :
It is not a container-based model; they use user space along with the kernel space of an OS
It does not share the host kernel
Each workload needs a complete OS or hypervisor
It can start only a single VM on a VMX

-To avoid dependency hell,

- Container Runtime (Docker Engine)
- Create a docker image(like class, template)
- Create container (like object) which

— Docker Architecture
  - Docker client
  - docker daemon
  - docker hub registry

- Docker clientcommands
  - docker build - to
  - Docker pull - to get image from the registry
  - Docker run - to create container from the image

— docker playground
https://labs.play-with-docker.com/
Hub.docker.com
Docker id: **sukumarrajamouli790**

password: Y@s@sw1n1

hub.docker.com
harinikokinti/H@rini790

— To create and build docker image

1. Base OS
2. Install JRE
3. Copy JAR
4. Configure
5. Expose Ports

Sample docker file:
Dockerfile

— To create the docker image, go to the project path where Dockerfile exists

```
docker build -t hello-world-docker .     (Don't forget dot)
```

— To create a container(instance of the image)

```
docker run -dp 8080:8080 --name=container1 hello-world-docker
```

```
docker run -dp 8080:8761 --name=bankingRegistry banking-registry-service
```

```
docker run -d --name=registry-service -p 8761:8761 banking-registry-service
```

— to create mysql image

```
docker run -dp 3306:3306 --name mysql1 -v mysql-vol:/var/lib/mysql -e
MYSQL_ROOT_PASSWORD=root -e MYSQL_USER=training -e
MYSQL_PASSWORD=training mysql:8.0.29
```

```
docker run -dp 3306:3306 --name mysql1 -v mysql-vol:/var/lib/mysql -e
MYSQL_ROOT_PASSWORD=root -e MYSQL_USER=mysql -e
MYSQL_PASSWORD=mysql mysql:8.0.29
```

Docker file:

```
#Base Image (Linux)
#Install JRE/JDK
#Copy JAR file
#Start App
#Expose Port
```

```
FROM openjdk:11
MAINTAINER saravana@docker.com
#ARG JAR_FILE
#COPY ${JAR_FILE} app.jar
COPY  ./target/*.jar app.jar
ENTRYPOINT ["java","-jar","app.jar"]
EXPOSE 8080
```

— To publish in docker hub

1. Login into hub.docker.com
2. Create repository
3. Build docker image
4. Open command prompt and login to docker account -> docker login
5. Create alias for the image with repository name same as in hub.docker.com
   ```
   docker tag hello-world-docker sukumarrajamouli790/hello-world-docker
   ```
6. Push the docker image into the hub
   ```
   docker push sukumarrajamouli790/hello-world-docker
   ```

7. Run docker image

```
Docker run -p 8080:8080 —name=hello1 sukumarrajamouli790/hello-world-docker
```

```
docker run -dp 8080:8080 --name=hello1 dockrtraining/hello-world-docker
```

```
docker run -dp 8080:8080 --name=hello1 sukumarrajamouli790/hello-world-docker
```

— Docker commands

1. To run and publish the image to the registry in background

```
docker run -dp 80:80 --name=demo1 docker/getting-started
```

2. To get the images

```
docker images
```

3. To get the containers

```
docker ps
```

4. To enter into the container (linux kernel)

```
docker exec -it demo1 /bin/sh
```

You can do some changes if needed and exit to come out of the container

5. To pull the docker image from the registry

docker pull <docker-image>
docker pull docker/getting-started

5. To run the docker image foreground

docker run docker/getting-started

[6:00 PM] Saravana (Guest)
apk add --no-cache openjdk11

[6:02 PM] Saravana (Guest)
chmod 777 mvnw

[6:02 PM] Saravana (Guest)
./mvnw clean install

— Universal Artifact manager
https://www.sonatype.com/products/nexus-repository

https://jfrog.com/artifactory/

mac
====
REFERENCE: https://docs.docker.com/docker-for-mac/install/

STEP 1: Download Docker Desktop for Mac
    https://hub.docker.com/editions/community/docker-ce-desktop-mac/

STEP 2: Install and run Docker Desktop on Mac
    Double-click Docker.dmg to open the installer, then drag the Docker icon to the Applications folder.

STEP 3: Double-click Docker.app in the Applications folder to start Docker
    The Docker menu in the top status bar indicates that Docker Desktop is running, and accessible from a terminal

STEP 4: Verify Docker Installation
    Open terminal and enter below command

```
		sudo docker version		(=> shows docker client and server details)
```

windows
=======
REFERENCE: https://docs.docker.com/docker-for-windows/install/

STEP 1: Download Docker Desktop for Windows
https://hub.docker.com/editions/community/docker-ce-desktop-windows/

STEP 2: Double-click Docker Desktop Installer.exe to run the installer

STEP 3: When prompted, ensure the Enable Hyper-V Windows Features option is selected on the Configuration page

STEP 4: Follow the instructions on the installation wizard to authorize the installer and proceed with the install
When the installation is successful, click Close to complete the installation process

STEP 5: Start Docker Desktop
To start Docker Desktop, search for Docker, and select Docker Desktop in the search results
When the whale icon in the status bar stays steady, Docker Desktop is up-and-running, and is accessible from any terminal window

STEP 4: Verify Docker Installation
Open command prompt and enter below command

```
		docker version		(=> shows docker client and server details)
```

*nix (ubuntu)
========
REFERENCE: https://docs.docker.com/engine/install/ubuntu/

STEP 1: Remove existing docker installation

```
		sudo apt-get remove docker docker-engine docker.io containerd runc
```

STEP 2: Install pre-requisite softwares

```
		sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
```

STEP 3: Add Docker Repo

```
		curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

STEP 4: Setup staple repository

```
        sudo add-apt-repository "deb [arch=amd64] https://
download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

STEP 5: Docker Installation

```
        sudo apt-get update
        sudo apt-get install docker-ce docker-ce-cli containerd.io
```

STEP 6: Verify Docker Installation

```
        sudo docker version        (=> shows docker client and server details)
```

———————————

Docker	
	- Introduction
	- Virtual Machine vs Container
	- What is Docker
	- Docker Architecture
	- Docker Engine
	- Docker Components
	- Image => read only template to create containers. Build and Ship. ~JAR
	- Container => running instance of image. Can run one or more containers from an image. Run JAR -> JVM -> mem /resources

	- Registry (Local [/var/lib/docker] | Public (Docker Hub) | Private => Nexus | Artifactory)
	- Docker Server (Daemon) (dockerd)
	- Docker Client
	- CLI
	- REST API
	- SDK
	- Plugins/Extentions
	- Docker Compose => enables to run multiple docker containers in single unit
	- Docker Swarm | Kubernetes => container management platform / orchestration platform
	- Docker Installation and Setup
	- Docker Desktop for Windows (10 Prof/Entp)
	- Docker for Windows 7 => Docker Toolbox
	- Docker for Linux
	- Docker for Mac
	- Docker Community Edition vs Docker Enterprise Edition
	- Docker Enterprise sold to Mirandis (2019)
	- Docker CLI
	- docker basic commands
	- docker image
	- docker container
	- Dockerfile
	step 1: install base os
	step 2: install nodejs / java
	step 3: copy application build code

	step 4: npm start | java -jar app.jar
	step 5: expose ports - 3000
	- Build Docker Image
	- Launch Docker Container
	- Save/Load Docker Image locally
	- Export/Import Docker Container locally
	- Working with Public Registry
	- Create Account in https://hub.docker.com
	- Docker Login
	- Tag Image and Push Image
	- Docker Logout
	- Connect to Running Docker Container
	- Make changes in Container and Commit
	- Eclipse Docker Tooling plugin
	- Docker Compose
	- Dockerizing microservices (ShoppingCart)
	- Docker Storage Drivers
	- AUFS
	- Overlay2
	- Devicemapper
	- Btrfs
	- ZFS

	- Docker Volumes
	- bind mount => persist the data into host file system. source directory should be created and available.
	- volumes => persist the data into docker storage area within host machine. managed by docker.
	- tmpfs mount => temporarily persist data into host memory. tmpfs mount removed once container stops.
	- Docker Networking
	- bridge => best when you need multiple containers to communicate on the same Docker host
	- host => best when the network stack should not be isolated from the Docker host, but you want other aspects of the container to be isolated
	- overlay => best when you need containers running on different Docker hosts to communicate using Swarm services
	- macvlan => best when you are migrating from a VM setup or need your containers to look like physical hosts on your network, each with a unique MAC address
	- none => no networking
	- Docker Linking (Deprecated)
	- Docker Swarm
	- Storage Drivers

	AUFS
	This is a stable driver; can be used for production-ready applications.
	It has good memory usage and is good for ensuring a smooth Docker experience for containers.
	There is a high-write activity associated with this driver which should be considered.
	It’s good for systems which are of Platform as a service type work.
	Overlay2
	This is a stable driver and it is in line with the main Linux kernel functionality.
	It has a good memory usage.
	This driver is good for testing applications in the lab. Also in production.
	Devicemapper
	This is a stable driver; ensures a smooth Docker experience.
	This driver is good for testing applications in the lab.
	This driver is in line with the main Linux kernel functionality.
	Btrfs
	This driver is in line with the main Linux kernel functionality.
	There is a high-write activity associated with this driver which should be considered.

	This driver is good for instances where you maintain multiple build pools.
	ZFS
	This is a stable driver and it is good for testing applications in the lab.
	It’s good for systems which are of Platform-as-a-Service type work.