**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Ansible**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***Ansible**

* Ansible is a configuration management tool.
* It is an open-source, community-driven product.
* In Ansible, there is a concept of control node and managed nodes.
* The server where the Ansible package is installed is called the control node.
* The control node and managed nodes are connected using the SSH protocol on port 22.
* Ansible is agentless, meaning we only need to install the Ansible package on the control node.
* There's no need to install anything on the managed nodes.

**Playbook in Ansible**

* Ansible uses a concept called a playbook.
* Playbooks are written on the control node.
* Whatever is written in the playbook is executed on the managed nodes connected to the control node.
* Playbooks are written in YAML (Yet Another Markup Language).
* A playbook contains plays, and plays contain tasks.
* Tasks written in a playbook are executed in a parallel way across all managed nodes.

**For example**,

* if we have 3 managed nodes and 2 tasks in a playbook:
* The first task will run on all 3 machines in parallel.
* Then the second task will run on all 3 machines in parallel.
* The configuration file for Ansible is located at: /etc/ansible/ansible.cfg

1. **amazon-linux-extras enable ansible2**
2. **yum install -y ansible**
3. **vi /home/ec2-user/keypair2.pem**
4. **chmod 400 /home/ec2-user/keypair2.pem**
5. **vi /etc/ansible/hosts**
6. **ansible -m ping webservers**
7. **vi install\_httpd.yml**
8. **ansible-playbook install\_httpd.yml**

* **vi /etc/ansible/hosts**

[webservers]

host1 ansible\_host=1.2.3.4 ansible\_user=ec2-user ansible\_ssh\_private\_key\_file=/home/ec2-user/keypair1.pem

host2 ansible\_host=5.6.7.8 ansible\_user=ec2-user ansible\_ssh\_private\_key\_file=/home/ec2-user/keypair2.pem

* **vi install\_httpd.yml**

---

- name: Install and start Apache on webservers

hosts: webservers

become: yes

tasks:

- name: Install httpd

yum:

name: httpd

state: present

- name: Start and enable httpd

service:

name: httpd

state: started

enabled: yes

- name: Create index.html

copy:

content: "<h1>Welcome to {{ inventory\_hostname }}</h1>"

dest: /var/www/html/index.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Virtualization & Hypervisor**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Virtualization**

* Virtualization is the process of creating multiple virtual machines using a

hypervisor

**Hypervisor**

* A hypervisor is software that allows us to create multiple virtual machines.

**Advantages of Virtualization**

1. Reduced capital and operating costs.
2. Minimized or eliminated downtime.
3. Increased IT productivity and efficiency.

**Disadvantages of Virtualization**

1. It takes time.
2. Wastage of resources.
3. Software/applications that run in one environment may not run in another environment.

**Types of Virtualization**

1) Desktop Virtualization

2) Application Virtualization

3) Server Virtualization

4) Storage Virtualization

5) Network Virtualization

6) OS Virtualization

7) Hardware Virtualization

**Types of Hypervisors**

**Type 1 Hypervisor**

* (Also called a bare-metal hypervisor)
* Type 2 Hypervisor

**Type 1 Hypervisor**

* If hypervisor software is directly installed on hardware, it is called a Type 1 hypervisor.
* For a Type 1 hypervisor, an OS is needed.

**Examples**

1. VMware ESXi

2. KVM (Kernel-Based VM)

3. Xen

**Type 2 Hypervisor**

* If hypervisor software is installed on an operating system (OS), it is called a Type 2 hypervisor.
* For a Type 2 hypervisor, an OS is needed.

**Example:**

* Oracle VirtualBox

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Docker**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Docker is an open platform for developing, shipping, and running applications.
* It is an open-source software platform used to create, deploy, and manage virtualized application containers on a common operating system (OS).
* Docker was developed in 2013.
* Docker is used to run containers.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Container**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Container**

* A container is a runnable instance of an image.
* It is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.
* Containers are lightweight and contain everything needed to run an application, so we don’t need to rely on what is currently installed on the host.
* A container is a fully packaged and portable computing environment. Everything an application needs to run, such as binaries, libraries, configuration files, and dependencies, is encapsulated and isolated within the container.
* We can create, start, stop, move, or delete a container using the Docker API or CLl.
* Containers are isolated from each other.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Docker Registries**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Docker Registries**

* Docker images are stored in a Docker registry.

**There are two types of registries:**

* Public registry
* Private registry

**Docker Hub is a public registry that anyone can use.**

* When we use the docker pull command, the required images are pulled from the registry.
* When we use the docker push command, the required images are pushed to the registry.

**Advantages of Docker**

1) No pre-allocation of RAM.

2) Lower cost.

3) Lightweight.

4) Takes very little time to create containers.

5) We can reuse images.

**Disadvantages of Docker**

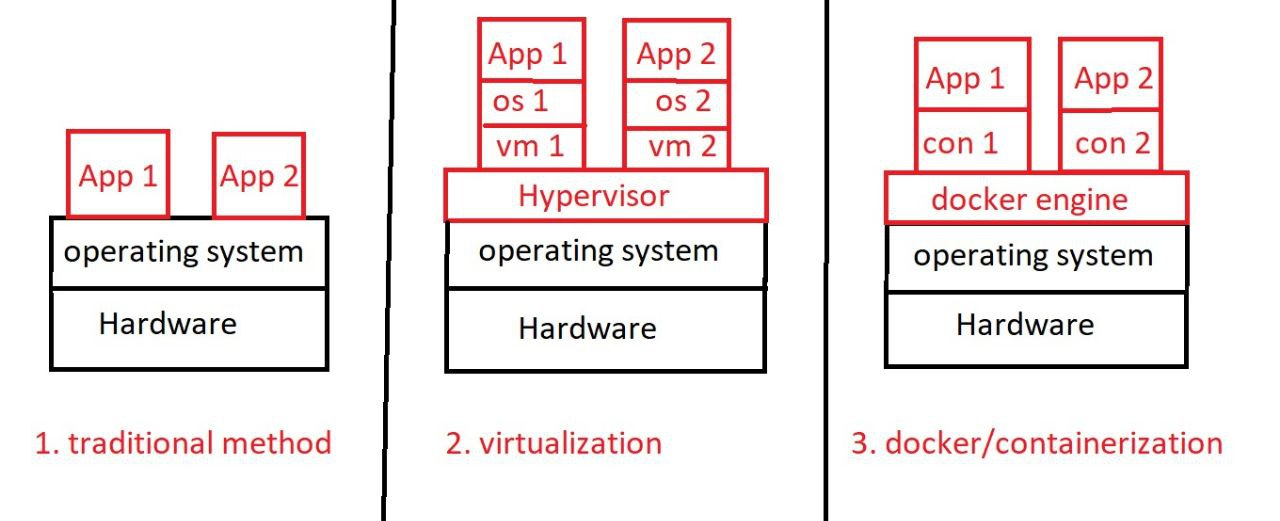
* Docker is not suitable for applications that require a rich GUI.
* Difficult to manage a large number of containers.
* Docker does not support cross-platform compatibility. If an application is designed to run in a Docker container on Windows, it cannot run on Linux, and vice versa.
* Docker is only suitable when the developer’s OS and tester’s OS are the same. If the OS is different, a VM is required.

**A developer writes code in a Dockerfile**.

* A Dockerfile contains all dependencies and software required to run an application.

**For example,**

* it may specify that:
* The container should have Red Hat OS.
* Git should be installed.
* Certain files and directories should be present.
* When we run this Dockerfile on the Docker engine, it creates an image, and from that image, it creates a container.
* We can upload this image to Docker Hub (registry).



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Installation of Docker**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Installation of Docker**

**To install Docker**

* yum install -y docker

**To check the Docker version**

* docker --version
* (OR)
* docker -v

**To check the status of Docker**

* systemctl status docker

**To start Docker**

* systemctl start docker

**Go to \*Docker Hub\* → \*Explore\* → (You can see multiple Docker images here)**

**To list all images:**

* docker images
* (OR)
* docker image ls

**To pull an image from Docker Hub:**

* docker pull <image-name>

**Example:**

* docker pull ubuntu
* docker pull centos

**To pull a specific version of an image:**

* docker pull ubuntu:22.04

**To list running containers:**

* docker ps
* (OR)
* docker container ls

**To list all containers (including stopped ones):**

* docker ps -a
* (OR)
* docker container ls -a

**To create a container from an image:**

* docker run <image-name>
* (The container will be in a stopped state.)

**Example:**

* docker run ubuntu

**To create/run a container from a specific image version:**

* docker run <image-name>:<tag>
* (The container will be in a stopped state.)

**Example**:

* docker run ubuntu:20.04

***We can create multiple containers using the same image.***

**To delete an image:**

* docker rmi -f <image-name>

**Example:**

* docker rmi -f ubuntu

**(If we delete an image, it does not affect existing containers.)**

**To delete a stopped container:**

* docker container rm <container-id>

**To delete a running or stopped container:**

* docker rm -f <container-id>

**To delete multiple containers:**

* docker rm -f <container-id-1> <container-id-2>

**To delete all stopped containers and images:**

* docker system prune -a -f

**To create and enter a container:**

* docker run -it <image-name> bash

**Example:**

* docker run -it ubuntu bash

**To exit from the container:**

* exit
* (OR)
* Ctrl + d
* (The container will be in a stopped state.)

**To detach from the container (keep it running):**

* Ctrl + p + q
* (The container will remain in a running state.)

**To start a container:**

* docker start <container-id>
* (OR)
* docker container start <container-id>

**To stop a container:**

* docker stop <container-id>
* (OR)
* docker container stop <container-id>

**To stop multiple containers:**

* docker stop <container-id> <container-id>

**To enter a running container:**

* (Container must be in a running state.)
* docker exec -it <container-id> bash
* (OR)
* docker attach <container-id>

**To search for an image**

* docker search httpd
* docker search ubuntu

**To rename an existing container**

* docker rename old-container-name new-container-name

**To create a container with a custom name**

* docker run --name containername imagename
* docker run --name lekharaj5 ubuntu
* docker run -it --name sumit ubuntu

**To see information about containers**

* docker info

**To see CPU and memory utilization by a container**

* docker stats container-id

**To view container logs**

* docker logs -f container-id

**Docker port expose/mapping/binding**

* docker run -itp 80:80 ubuntu
* apt-get update -y
* apt-get install apache2 -y
* service apache2 status
* service apache2 start
* cat > /var/www/html/index.html
* If we have already created a container and forgot to set up port mapping, we can't modify the port mapping for the existing container.
* But we can delete that container and create a new one using the same image and set up the correct port mapping by using the below command,
* docker run -itp 80:80 imagename

**Docker volume**

* To sync a directory on a local machine with a directory in a container, we use Docker volume:
* docker run -itv machine\_dir:container\_dir ubuntu

**Port mapping & docker volume**

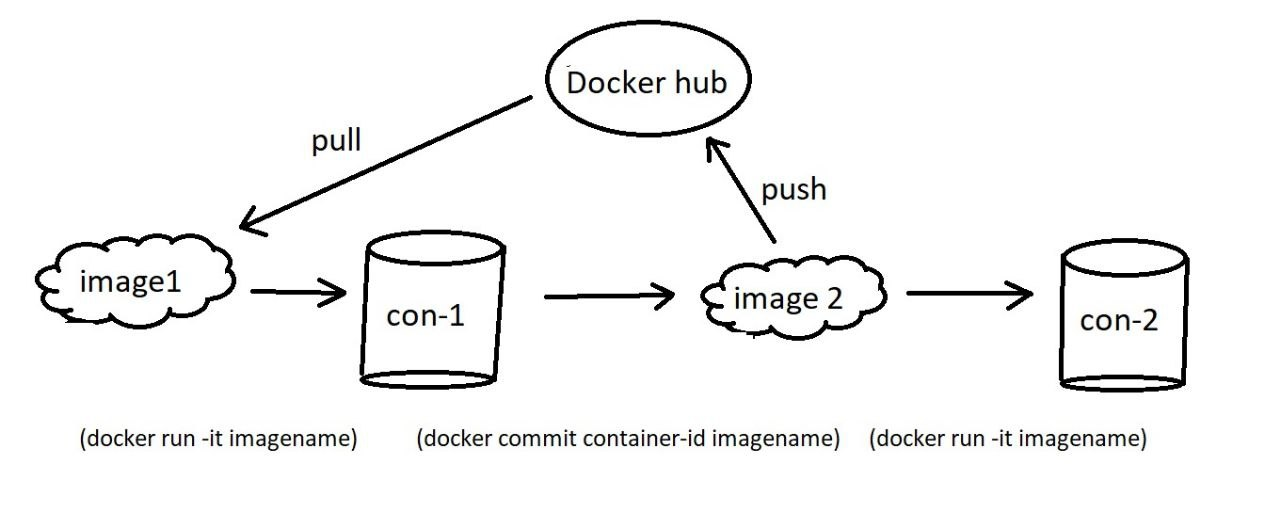
* docker run -itp 80:80 -v machine\_dir:container dir imagename

**To create a container from an image**

* docker run -it imagename

**To create an image from a container**

* docker commit container-name imagename



**To push an image to docker hub**

* docker login
* docker push imagename

**To check network connectivity**

* ping 8.8.8.8
* ping google.com
* ping localhost
* ping 8.8.8.8 -c 5
* ping google.com -c 5
* ping -c 5 google.com
* ping localhost -c 5

**To run a CentOS container**

* docker run -it centos:centos7 bash

**To run a CentOS container and execute a command**

* docker run -it centos:centos7
* Ping 8.8.8.8
* docker run -it centos:centos7 ping 8.8.8.8
* docker run -itd centos:centos7 ping 8.8.8.8

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Docker Compose**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Docker Compose**

* Docker Compose is a tool for defining and running multiple containers.
* It uses a YAML file where all the configuration steps are defined.
* YAML stands for YAML Ain’t Markup Language (a recursive acronym originally standing for Yet Another Markup Language").
* Docker Compose treats each container as a service.
* With Docker Compose, we can create and manage multiple containers easily.
* Docker Compose configuration is written in YAML format.
* YAML files use a key-value format:
* key: value

**The Docker Compose file should be named either:**

* docker-compose.yml or docker-compose.yaml
* There are multiple versions of the Docker Compose file format.
* We need to specify the version you are using at the beginning of the file.

**A commonly used version is 3.**

* yum install -y docker
* systemctl start docker
* To install Docker Compose
* curl -L --fail https://github.com/docker/compose/releases/download/1.29.2/run.sh -o /usr/local/bin/docker-compose
* chmod +x /usr/local/bin/docker-compose

**Ubuntu and CentOS containers using Docker Compose**

**vim docker-compose.yaml**

version: '3'

services:

one:

image: 'ubuntu'

command: tail -f /dev/null

two:

image: 'centos:centos7'

command: tail -f /dev/null

**To run a Docker Compose file**

* docker-compose up -d
* docker ps
* docker exec -it container\_name\_or\_id bash

**To run a particular service**

* docker-compose up servicename
* docker-compose up -d one
* docker-compose up -d two

**Creating an Ubuntu web server, binding port 80 of the host to port 80 of the Ubuntu container, and syncing /dir1 on the host with /dir2 in the container using Docker Compose.**

mkdir /dir1

cd /dir1

touch file1

cd /root

**vim docker-compose.yaml**

version: '3.8'

services:

ubuntu-apache:

image: ubuntu:22.04

container\_name: ubuntu\_apache\_server

ports:

- "80:80"

volumes:

- /dir1:/dir2

command: >

bash -c "apt-get update &&

apt-get install -y apache2 &&

echo 'ServerName localhost' >> /etc/apache2/apache2.conf &&

echo 'hello all' > /var/www/html/index.html &&

apachectl -D FOREGROUND"

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Dockerfile**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* A Dockerfile is a simple text file with a set of instructions.
* The name of the file should be "Dockerfile" (with a capital D).
* vim Dockerfile
* Component names should be in capital letters.
* Components of a Dockerfile:

**1. FROM**

* Specifies the base image to be downloaded from Docker Hub.

**2. MAINTAINER**

* Defines the author or owner of the Dockerfile.

**3. EXPOSE**

* Specifies the ports that need to be opened.

**4. WORKDIR**

* Sets the directory to log in to within the container.

**5. ENV**

* Sets environment variables inside the container.

**6. ARG**

* Passes an argument to the container.

**7. COPY**

* Copies data from the local machine to the container.

**8. ADD**

* Similar to COPY but also extracts files into the container from the local machine.
* It can also download data from the internet.

**9. RUN**

* Executes commands while creating the image.

**10. CMD**

* Executes commands while creating the container.
* Multiple CMD instructions can be used.

**11. ENTRYPOINT**

* Similar to CMD but has a higher priority than CMD.
* A Dockerfile can have only one ENTRYPOINT.
* The first command executed in the container is from ENTRYPOINT, followed by the commands mentioned in CMD.

**cd /root**

**vim Dockerfile**

FROM ubuntu

MAINTAINER Lekharaj

WORKDIR /tmp

RUN echo "Hello all" > /file1

**To create an image from a Dockerfile**

* docker build -t custom-imagename path-of-Dockerfile
* docker build -t image1 /root

**To create a container from an image**

* docker run -it imagename bash
* docker run -it image1 bash

**cd /root**

**vim Dockerfile**

FROM ubuntu

WORKDIR /tmp

RUN echo "Hello all" > /home/file20

RUN mkdir /dir1

COPY file60 /dir1

ADD file5.tar /tmp

**To create an image from a Dockerfile**

* docker build -t custom-imagename path-of-Dockerfile
* docker build -t image1 /root

**create a container from an image**

* docker run -it imagename
* docker run -it image1
* FROM ubuntu
* RUN apt-get update -y
* RUN apt-get install tree -y
* RUN apt-get install git -y
* RUN apt-get install apache2 -y

**To check whether a particular package is installed on an Ubuntu machine**

* apt list --installed pkgname

**To optimize a Dockerfile**:

* FROM ubuntu
* RUN apt-get update -y && apt-get install tree git apache2 -y

**Types of formats to write a Dockerfile**

**1) Shell format**

**2) Exec format**

**1) Shell format**

FROM ubuntu

RUN mkdir /dir1

RUN touch /file1

**2) Exec format**

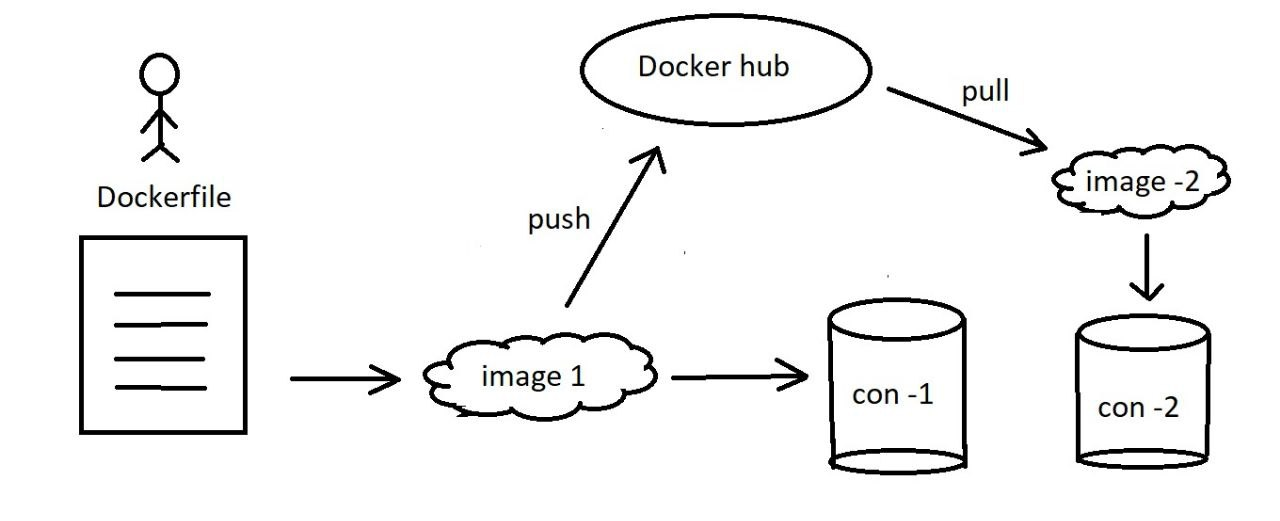
FROM ubuntu

RUN [ "mkdir", "/dir1" ]

RUN [ "touch", "/file1" ]

**How to push a custom image to Docker Hub**

* Create a Dockerfile
* Build the image
* docker build -t dockerhub-username/imagename path-of-Dockerfile
* Log in to Docker Hub
* docker login
* Push the image
* docker push account-name/imagename



**Docker EXP**

* Having knowledge of Docker to create isolated environments for development and testing.
* Understanding how to manage Docker images and versions using Docker Hub.
* Knowledge of creating and running Docker containers to ensure consistent application performance.
* Understanding how to write Dockerfiles to build and configure application images.
* Understanding Docker Compose to manage multi-container applications.

**Roles & responsibilities in Docker**

* Used Docker to create isolated environments for development and testing
* Managed Docker images and their versions using Docker Hub
* Created and managed Docker containers to ensure applications run the same in any environment.
* Created Dockerfiles to build and configure application images