CLOUD CONCEPTS

(DEV-OPS TOOL: AWS DOCKER)

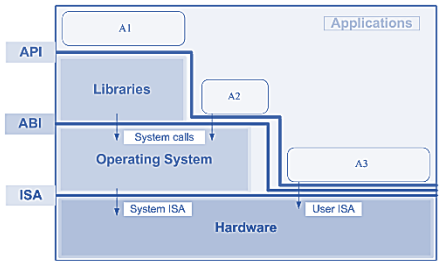
Concept of Virtualization in Cloud: Virtualization is a basic enabler of Cloud Computing; it simplifies the management of physical resources for the three abstractions.

For example, the state of a virtual machine (VM) running under a virtual machine monitor (VMM) can be saved and migrated to another server to balance the load. For example, virtualization allows users to operate in environments they are familiar with, rather than forcing them to specific ones.

- Virtualization abstracts the underlying resources; simplifies their use; isolates users from one another; and supports replication which increases the elasticity of a system.
- Cloud resource virtualization is important for:
    - Performance isolation (we can dynamically assign and account for resources across different applications)
    - System security: (it allows isolation of services running on the same hardware)
    - Performance and reliability: (it allows applications to migrate from one platform to another)
    - The development and management of services offered by a provider.

- Virtualization simulates the interface to a physical object by:
  - Multiplexing: creates multiple virtual objects from one instance of a physical object. Many virtual objects to one physical. Example - a processor is multiplexed among a number of processes or threads.
  - Aggregation: creates one virtual object from multiple physical objects. One virtual object to many physical objects. Example - a number of physical disks are aggregated into a RAID disk.
  - Emulation: constructs a virtual object of a certain type from a different type of a physical object. Example - a physical disk emulates a Random Access Memory (RAM).
  - Multiplexing and emulation. Examples - virtual memory with paging multiplexes real memory and disk; a virtual address emulates a real address.

- Layering – a common approach to manage system complexity:
  - Simplifies the description of the subsystems; each subsystem is abstracted through its *interfaces* with the other subsystems
  - Minimises the interactions among the subsystems of a complex system
  - With layering we are able to design, implement, and modify the individual subsystems independently
- Layering in a computer system:
  - Hardware
  - Software
    - Operating system
    - Libraries
    - Applications



Application Programming Interface (API), Application Binary Interface (ABI), and Instruction Set Architecture (ISA). An application uses library functions (A1), makes system calls (A2), and executes machine instructions (A3) (from book)

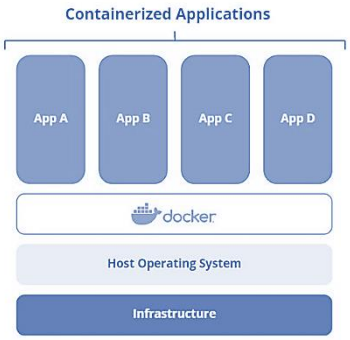
Tools of Dev-Ops: Chef, Ansible, Jenkins, Terraform, Docker

Docker: It is a software development platform to deploy apps that create containers (like a virtual machine).

- Developed in 2013
- It runs natively on Linux operating system only.
- It provides containerization.
- Apps are packaged in containers that can be run on any operating system.
- App behavior would be the same regardless of where they run.

- Any machine
- No compatibility issues
- Predictable behavior
- Less work
- Easier to maintain and deploy
- Works with any language, any OS, any technology

Say you developed a website using PHP (latest version) on Apache server version V.X. The operating system you used while building the website is Linux. After developing the website, you shared your website with the testing team before deployment and the website did not run properly. Possible problems could be:



- Version of the software
- Library, DLL files or dependency missing.
- Different operation systems
- Environment incompatible

In short, the environment where the app is developed is upgraded while the app is tested on old production environment.

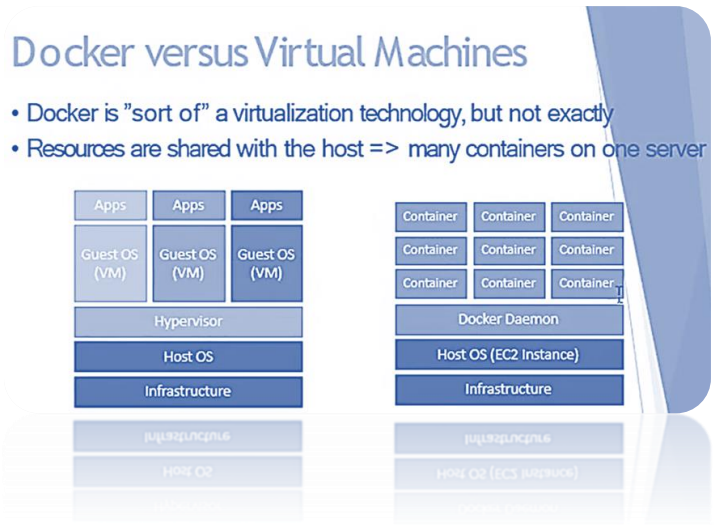
Problem statement: Your app is working fine in your environment while in production environment after stagging app did not install properly in testing phase.

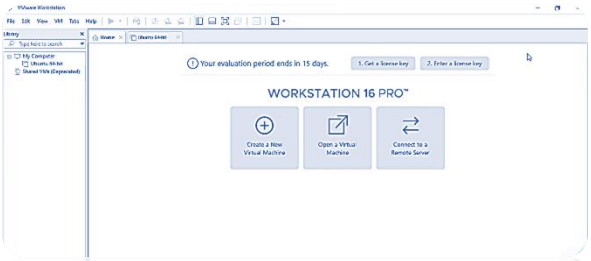
Solution: (one of many) is Docker

1. The developer will create a docker file. This file will have the instructions for the items that are to be executed or run whenever docker file runs.
2. Create and place the image on docker hub.
    a. Run docker image using the docker file. Docker image is placed on Docker Hub (Just like Git Hub tool where code is placed and pulled whenever code is to be run)
        i. Docker hub is a public registry where different images are placed. You can also place your created image on docker hub.
3. Stagging environment will pull that image from docker hub and run the docker container.
4. Same Image is pulled by testing environment from docker hub and run the docker container.

5. In both cases there will be no issues or errors.

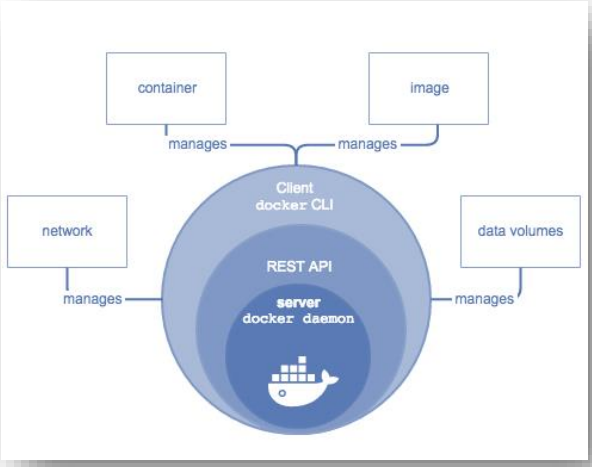
- **Virtualization and Containerization:**
    - In virtualization every application runs on a virtual machine. This virtual machine is created using Type II hypervisor (e.g., VMware workstation) if runs on the host operating system. Many enterprises use the Type I hypervisor known as bare metal hypervisor (e.g., EXSI).
        - Resources once provided to any VM can't be used by other VM. (Resource underutilization).
    - In case of containerization resources can be shared (i.e., shared operating system)
        - Everything like bin files is already installed.

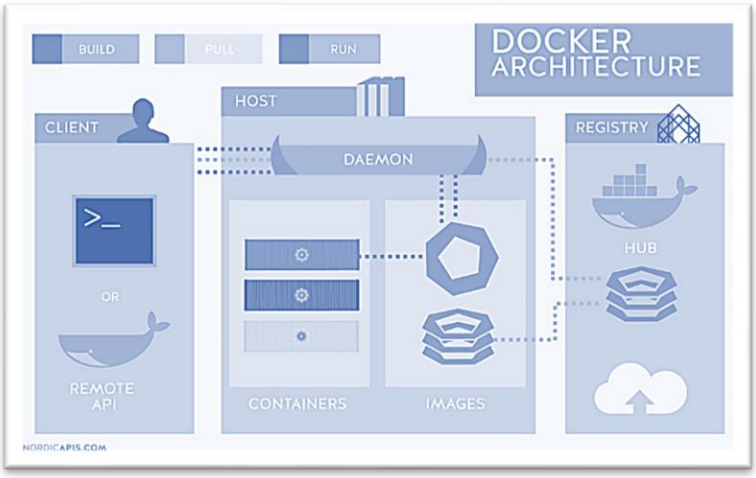


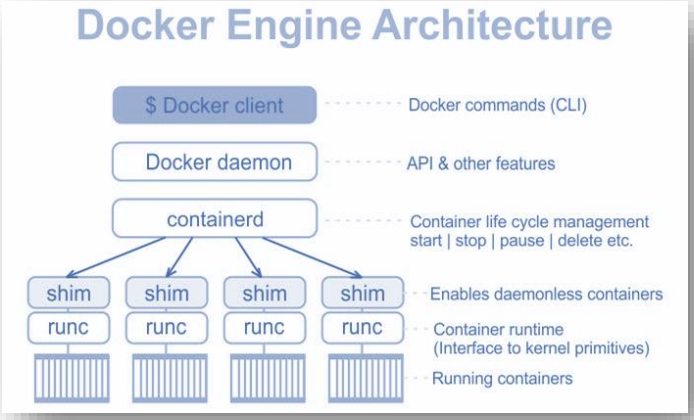


Docker Architecture:

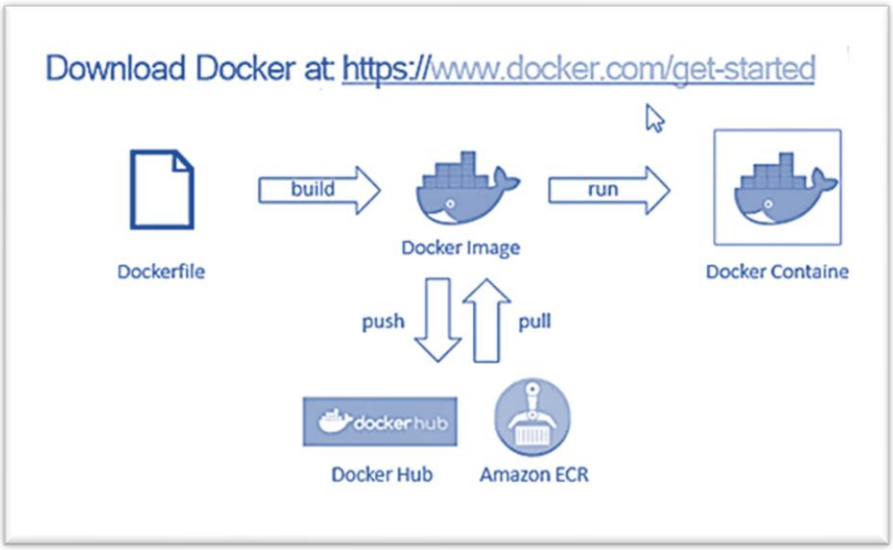
- Docker engine contains:
    - Docker demon (manages networking and volumes)
    - Rest API
    - Docker client (create file/images and run)
- The Docker client is **YOU.**
- Server is **Docker Demon**.

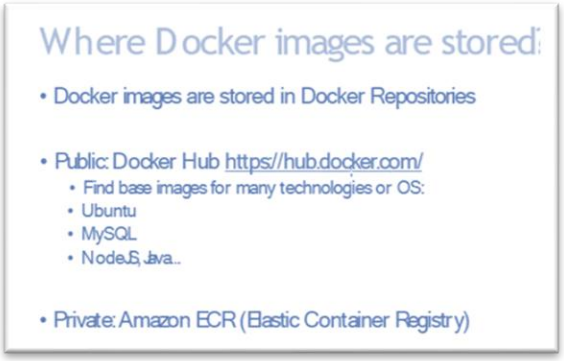




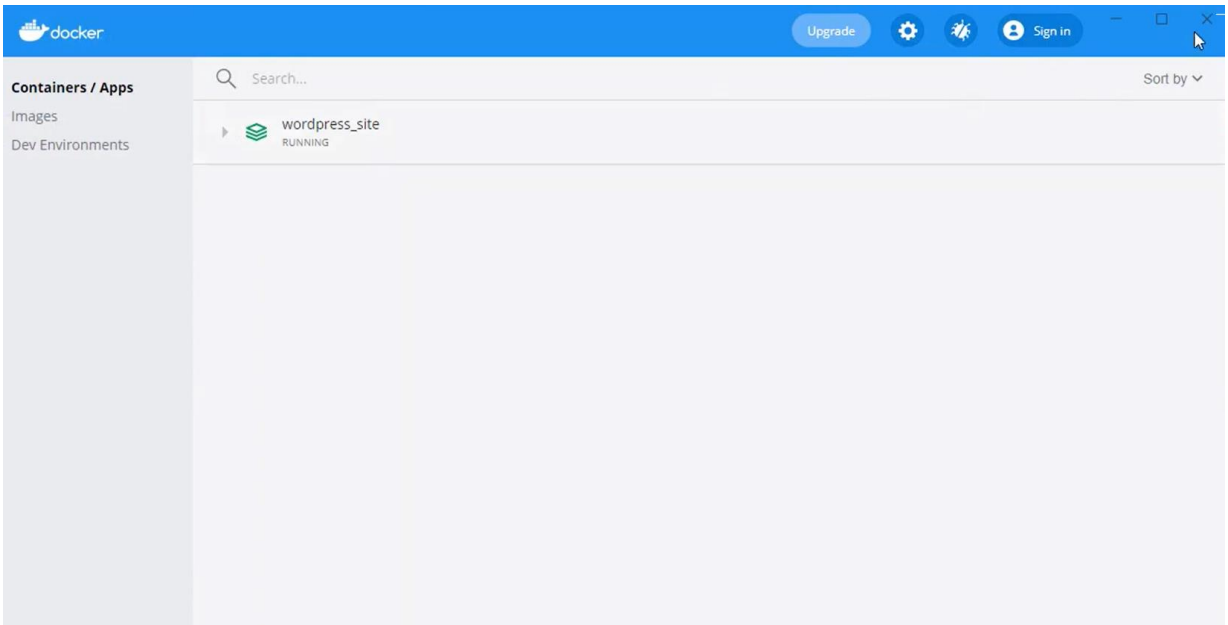


- **Working:**
  1. Docker clients make requests to DD.
  2. DD runs the instructions and creates images successfully and launches a container.
  3. If you want to save an image, save it on docker hub or on private registry.
  4. Next time a new image is not created but pulled from registry and run.





- **Docker Desktop App:**
  https://docs.docker.com/desktop/install/windows-install/



Lab (Docker- Containers)

1. Launch an EC2 Instance on Amazon and SSH it through PowerShell or Putty.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\HP> cd downloads
PS C:\Users\HP\downloads> ssh -i "db09keypairdocker.pem" ec2-user@ec2-54-162-150-130.compute-1.amazonaws.com
The authenticity of host 'ec2-54-162-150-130.compute-1.amazonaws.com (54.162.150.130)' can't be established.
ED25519 key fingerprint is SHA256:y909jjcK0Bhx+d1Hd4KOLQEo+J4YKZJpdwx8Pp29Vhc.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'ec2-54-162-150-130.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
       #_
       ~\_  ####_        Amazon Linux 2
      ~~  \_#####\
      ~~     \###|       AL2 End of Life is 2025-06-30.
      ~~       \#/ ___
       ~~       V~' '->
        ~~~         /    A newer version of Amazon Linux is available!
          ~~._.   _/
             _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
           _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-32-15 ~]$
[ec2-user@ip-172-31-32-15 ~]$ sudo su
[root@ip-172-31-32-15 ec2-user]# sudo yum update -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
No packages marked for update
```

2. Run the following command to see if docker is installed.

```
[root@ip-172-31-32-15 ec2-user]# which docker
/usr/bin/which: no docker in (/sbin:/bin:/usr/sbin:/usr/bin)
```

3. Install Docker.

```
[root@ip-172-31-32-15 ec2-user]# amazon-linux-extras install docker
Installing docker
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-kernel-5.10
17 metadata files removed
6 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
amzn2extra-docker
amzn2extra-kernel-5.10
```

```
Installed:
  docker.x86_64 0:20.10.25-1.amzn2.0.3

Dependency Installed:
  containerd.x86_64 0:1.7.2-1.amzn2.0.1      libcgroup.x86_64 0:0.41-21.amzn2      pigz.x86_64 0:2.3.4-1.amzn2.0.1
  runc.x86_64 0:1.1.7-4.amzn2
```

4. Check if Docker is installed.

```
[root@ip-172-31-32-15 ec2-user]# which docker
/bin/docker
```

5. Check the version.

```
[root@ip-172-31-32-15 ec2-user]# docker -v
Docker version 20.10.25, build b82b9f3
```

6. Check the information.

```
[root@ip-172-31-32-15 ec2-user]# docker info
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Docker Buildx (Docker Inc., v0.0.0+unknown)
```

7. Check Docker service status.

```
[root@ip-172-31-32-15 ec2-user]# service docker status
Redirecting to /bin/systemctl status docker.service
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
     Docs: https://docs.docker.com
```

8. Start Docker service and check the status.

```
[root@ip-172-31-32-15 ec2-user]# service docker start
Redirecting to /bin/systemctl start docker.service
```

```
[root@ip-172-31-32-15 ec2-user]# service docker status
Redirecting to /bin/systemctl status docker.service
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
   Active: active (running) since Thu 2023-11-30 18:26:17 UTC; 31s ago
     Docs: https://docs.docker.com
  Process: 3530 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
  Process: 3528 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
 Main PID: 3533 (dockerd)
    Tasks: 7
   Memory: 20.6M
   CGroup: /system.slice/docker.service
           └─3533 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=3276...
```

9. Check docker information again.

```
[root@ip-172-31-32-15 ec2-user]# docker info
Client:
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Docker Buildx (Docker Inc., v0.0.0+unknown)

Server:
 Containers: 0
  Running: 0
  Paused: 0
  Stopped: 0
 Images: 0
 Server Version: 20.10.25
 Storage Driver: overlay2
  Backing Filesystem: xfs
  Supports d_type: true
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: cgroupfs
 Cgroup Version: 1
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
 Swarm: inactive
 Runtimes: runc io.containerd.runc.v2 io.containerd.runtime.v1.linux
 Default Runtime: runc
 Init Binary: docker-init
 containerd version: 0cae528dd6cb557f7201036e9f43420650207b58
 runc version: f19387a6bec4944c770f7668ab51c4348d9c2f38
 init version: de40ad0
 Security Options:
  seccomp
   Profile: default
 Kernel Version: 5.10.199-190.747.amzn2.x86_64
 Operating System: Amazon Linux 2
 OSType: linux
 Architecture: x86_64
```

10. Check the images and any docker containers (stopped or running).

a. docker images [to check any container images]
b. docker ps [to see the running containers]
c. docer ps -a [to see the running and stopped containers]

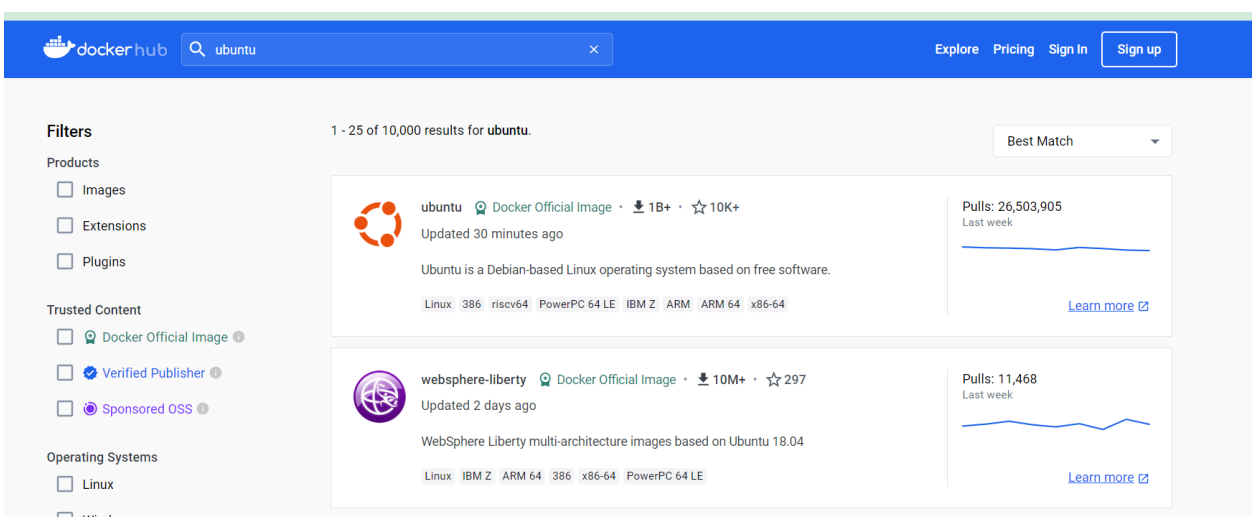
```
[root@ip-172-31-32-15 ec2-user]# docker images
REPOSITORY    TAG        IMAGE ID    CREATED    SIZE
[root@ip-172-31-32-15 ec2-user]# docker ps
CONTAINER ID    IMAGE      COMMAND    CREATED    STATUS     PORTS      NAMES
[root@ip-172-31-32-15 ec2-user]# docker ps -a
CONTAINER ID    IMAGE      COMMAND    CREATED    STATUS     PORTS      NAMES
```

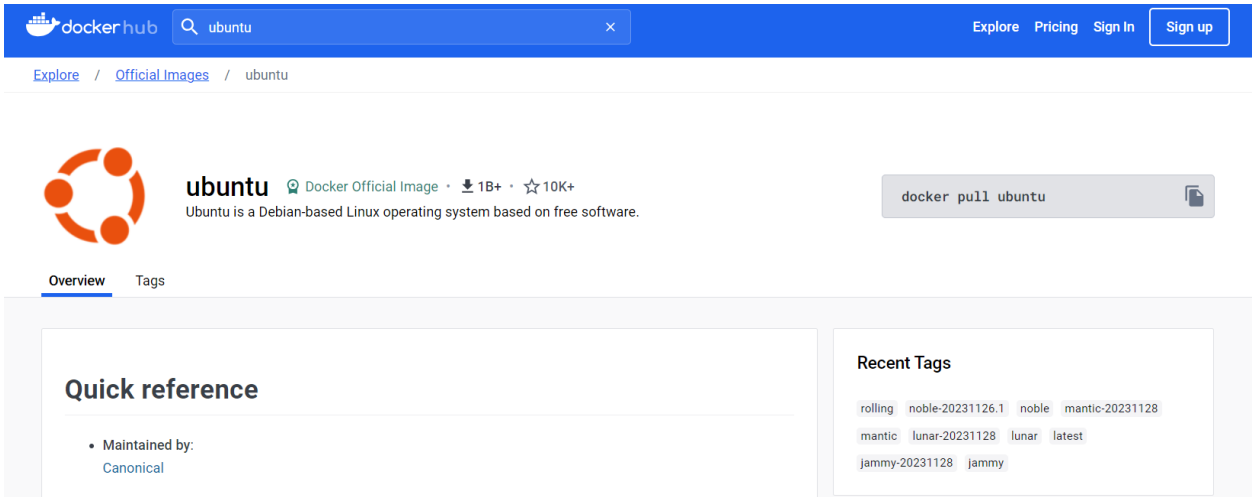
11. Install the docker container using the command as shown. The container will start running after executing the command. Use ls command to see the directory structure. To exit the running container, use the exit command.
    a. Ubuntu image available at the public repository of the docker.
    b. Name must be the same while pulling.

```
[root@ip-172-31-32-15 ec2-user]# docker run -it ubuntu /bin/bash
```

```
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
aece8493d397: Pull complete
Digest: sha256:2b7412e6465c3c7fc5bb21d3e6f1917c167358449fecac8176c6e496e5c1f05f
Status: Downloaded newer image for ubuntu:latest
root@5a31ef7973cf:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
```

```
root@5a31ef7973cf:/# exit
exit
```





12. Run the following commands to see the available images and containers.

```
[root@ip-172-31-32-15 ec2-user]# docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED         STATUS                      PORTS     NAMES
5a31ef7973cf   ubuntu    "/bin/bash"   3 minutes ago   Exited (127) 7 seconds ago            agitated_perlman
[root@ip-172-31-32-15 ec2-user]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    e4c58958181a   8 weeks ago   77.8MB
```

13. Run the image again and exit it. You will see that two containers are created from that image (the image is only one). You can run as many containers as possible from that image.

```
[root@ip-172-31-32-15 ec2-user]# docker run -it ubuntu /bin/bash
root@ac61e25a825c:/# exit
exit
[root@ip-172-31-32-15 ec2-user]# docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED             STATUS                      PORTS     NAMES
ac61e25a825c   ubuntu    "/bin/bash"   About a minute ago  Exited (0) 5 seconds ago            gallant_feistel
5a31ef7973cf   ubuntu    "/bin/bash"   5 minutes ago       Exited (127) 2 minutes ago          agitated_perlman
```

```
[root@ip-172-31-32-15 ec2-user]# docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    e4c58958181a   8 weeks ago   77.8MB
```

14. Installing and running other docker images.

```
[root@ip-172-31-32-15 ec2-user]# docker run -it kalilinux/kali-rolling /bin/bash
```

```
┌──(root㉿0955ee0199d1)-[/]
└─# cat /etc/os-release
PRETTY_NAME="Kali GNU/Linux Rolling"
NAME="Kali GNU/Linux"
VERSION_ID="2023.3"
VERSION="2023.3"
VERSION_CODENAME=kali-rolling
ID=kali
ID_LIKE=debian
HOME_URL="https://www.kali.org/"
SUPPORT_URL="https://forums.kali.org/"
BUG_REPORT_URL="https://bugs.kali.org/"
ANSI_COLOR="1;31"

┌──(root㉿0955ee0199d1)-[/]
└─# exit
exit
```

```
[root@ip-172-31-32-15 ec2-user]# docker run -it jenkins/jenkins:lts /bin/bash
Unable to find image 'jenkins/jenkins:lts' locally
lts: Pulling from jenkins/jenkins
```

15. Just pulling the images from the docker hub.

```
[root@ip-172-31-32-15 ec2-user]# docker pull centos
Using default tag: latest
latest: Pulling from library/centos
a1d0c7532777: Pull complete
```

16. Finding the machine image without visiting the hub.

```
[root@ip-172-31-32-15 ec2-user]# docker search chef
NAME                                DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
chef/chef                           A systems integration framework, built to br…   67
chef/inspec                         InSpec auditing and testing framework.          15
chef/chefworkstation                                                                3
chefes/buildkite-windows                                                            1
chefes/a2-integration                                                               1
cincproject/auditor                 Cinc community rebuild of Chef InSpec           1
chefdemo/compliance-loader-fail                                                     0
chefdemo/automate-nginx                                                             0
chefes/buildkite                                                                    0
chefes/centos-systemd                                                               0
cincproject/cinc                    Community rebuild of Chef Infra Client          0
chefes/expeditor                                                                    0
osuosl/chef-zero                                                                    0
chefdemo/compliance-loader-pass                                                     0
chefes/expeditor-www                                                                0
cincproject/workstation             Cinc community rebuild of Chef Workstation      0
chefdemo/workflow-server                                                            0
chefdemo/logstash                                                                   0
chefdemo/compliance                                                                 0
chefdemo/postgresql                                                                 0
chefdemo/rabbitmq                                                                   0
chefdemo/notifications                                                              0
osuosl/chef-s390x                                                                   0
chefdemo/postgresql-data                                                            0
chefes/a1-buildkite                                                                 0
```

17. Setting the name of the container.

```
[root@ip-172-31-32-15 ec2-user]# docker run -it --name drakhshan ubuntu /bin/bash
root@e1cf0df75b1f:/# cat /etc/os-release
PRETTY_NAME="Ubuntu 22.04.3 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.3 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
root@e1cf0df75b1f:/# exit
exit
```

18. Starting and stopping the docker by name.

```
[root@ip-172-31-32-15 ec2-user]# docker start drakhshan
drakhshan
[root@ip-172-31-32-15 ec2-user]# docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED              STATUS          PORTS     NAMES
e1cf0df75b1f   ubuntu    "/bin/bash"   About a minute ago   Up 21 seconds             drakhshan
[root@ip-172-31-32-15 ec2-user]# docker attach drakhshan
root@e1cf0df75b1f:/# exit
exit
[root@ip-172-31-32-15 ec2-user]# docker stop drakhshan
drakhshan
```

19. See the running and stopped containers.

```
[root@ip-172-31-32-15 ec2-user]# docker ps -a
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS                        PORTS     NAMES
e1cf0df75b1f   ubuntu                   "/bin/bash"              4 minutes ago    Exited (0) 46 seconds ago
  drakhshan
78dbfcbd648c   jenkins/jenkins:lts      "/usr/bin/tini -- /u…"   8 minutes ago    Exited (0) 7 minutes ago
  ecstatic_leakey
0955ee0199d1   kalilinux/kali-rolling   "/bin/bash"              11 minutes ago   Exited (0) 10 minutes ago
  elated_goodall
ac61e25a825c   ubuntu                   "/bin/bash"              17 minutes ago   Exited (0) 16 minutes ago
  gallant_feistel
5a31ef7973cf   ubuntu                   "/bin/bash"              21 minutes ago   Exited (127) 18 minutes ago
```

20. Use the following command to remove the container.

```
[root@ip-172-31-32-15 ec2-user]# docker rm drakhshan
drakhshan
```

21. See the docker images after removing the docker.

```
[root@ip-172-31-32-15 ec2-user]# docker ps -a
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS                        PORTS     NAMES
78dbfcbd648c   jenkins/jenkins:lts      "/usr/bin/tini -- /u…"   8 minutes ago    Exited (0) 8 minutes ago
  ecstatic_leakey
0955ee0199d1   kalilinux/kali-rolling   "/bin/bash"              12 minutes ago   Exited (0) 11 minutes ago
  elated_goodall
ac61e25a825c   ubuntu                   "/bin/bash"              18 minutes ago   Exited (0) 16 minutes ago
  gallant_feistel
5a31ef7973cf   ubuntu                   "/bin/bash"              22 minutes ago   Exited (127) 19 minutes ago
  agitated_perlman
```