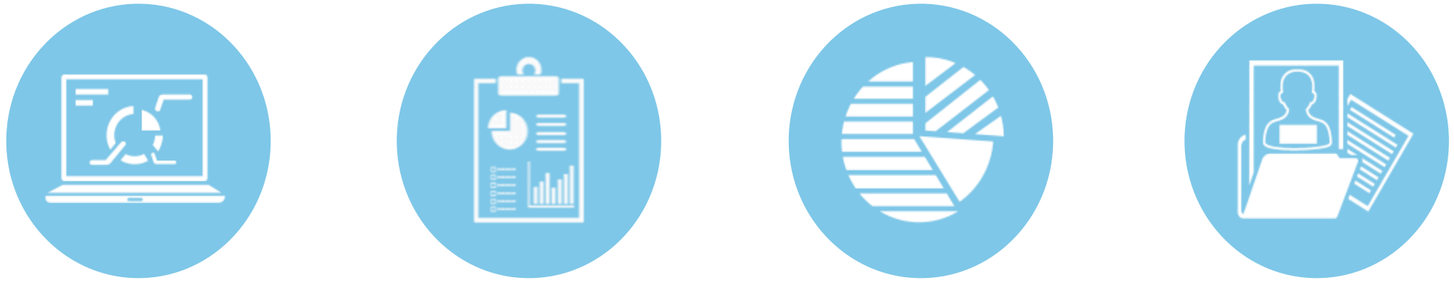
DevOps Certification Training

Lesson 06: Containerization with Docker



© Simplilearn. All rights reserved.

Learning Objectives

By the end of this lesson, you will be able to:

- Describe the importance of Docker over Virtual Machines
- Demonstrate on Docker images, Containers, and Docker Registry
- Demonstrate on Docker Compose and Docker Hub
- Describe the importance of Docker Networking

©Simplilearn. All rights reserved.

Containerization with Docker

Overview of Docker

©Simplilearn. All rights reserved.

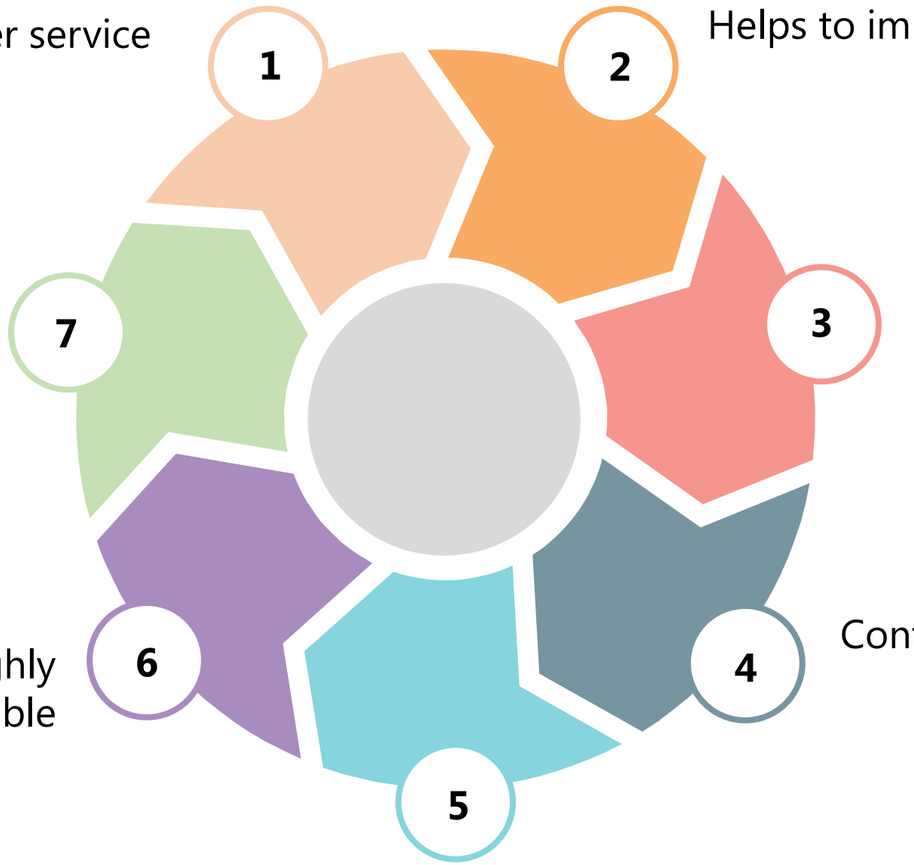
Docker Introduction

Docker is an open-source software that enables microservices and containerization of software applications. It's a light weight container service that consists of source code with dependencies bundled up together.

©Simplilearn. All rights reserved.

Docker Features

1. Light weight container service
2. Helps to implement micro services
3. Source codes with dependencies are bundled up together inside container
4. Containers runs in isolation
5. Helps to resolve the versioning problem with software
6. Fast, reliable and highly scalable
7. Is independent of the operating system and supports platforms based on both Windows and Unix



©Simplilearn. All rights reserved.

Docker Workflow

Pull or Build Docker image from Docker hub, bundle source with dependencies inside this Docker image.

Runs pulled Docker image on Docker host and start container on the host.

Container will be running in background and we can kill or stop container anytime we want to do so.

We can access applications hosted in Docker container using host machine IP and port details.

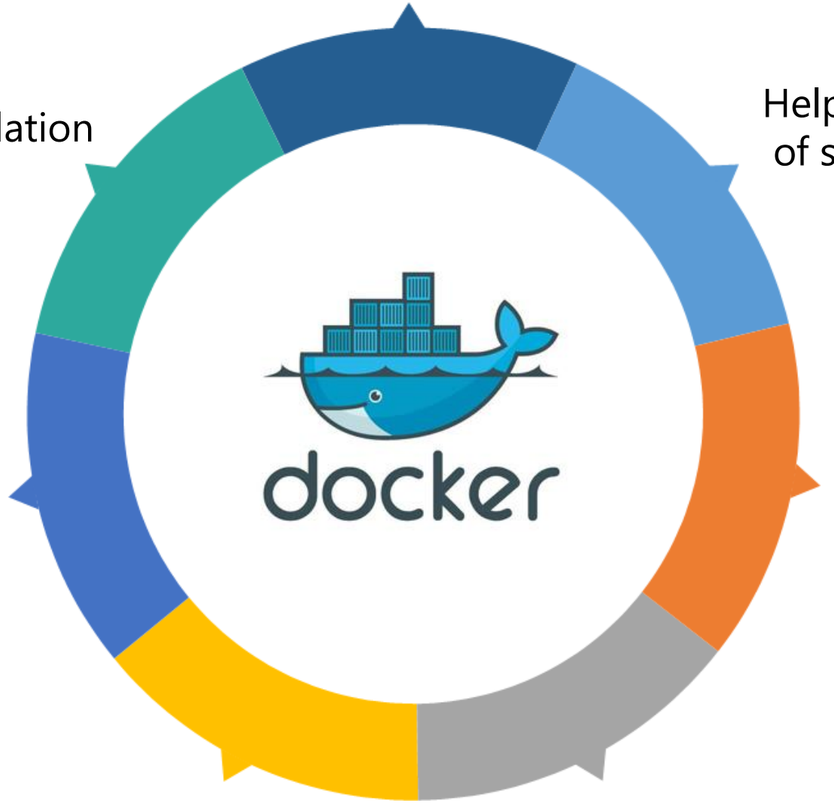
©Simplilearn. All rights reserved.

Docker Benefits

Infrastructure cost can be saved

Applications run in total isolation

Helps to establish standardization of software applications



Can be built once and deploy to multiple environments and operating systems

Eliminates versioning and dependency issues related to hosting source code on server

Is a better solution for CI/CD implementation of software applications

Helps in fast configurations and building applications

©Simplilearn. All rights reserved.

Containerization with Docker

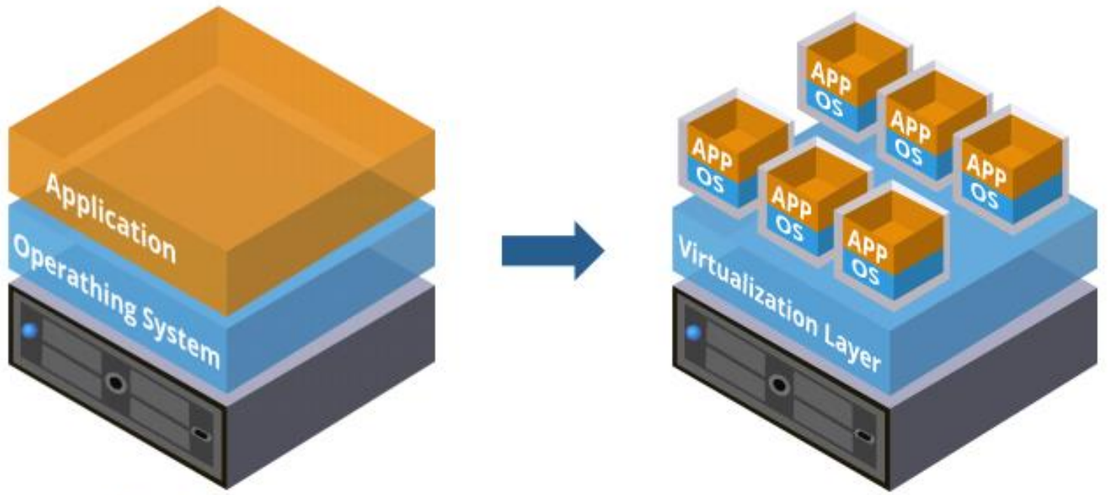
Overview of Virtualization

©Simplilearn. All rights reserved.

Virtualization

Virtualization refers to creating virtual resources like server, CPU, storage or network. You can create multiple sub-machines that can be used to host software applications.

- It helps to create multiple environment out of the same physical system
- It helps in optimal utilization of physical resources and helps in reducing expenses
- It helps to host multiple operating systems isolated from each other



Traditional Architecture

Virtual Architecture

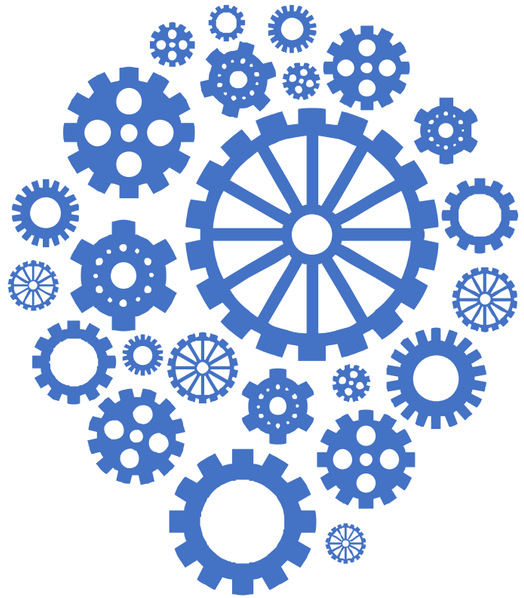
©Simplilearn. All rights reserved.

Virtualization Advantages

Creates multiple environments on the same physical system

Helps to reduce the infrastructure cost margin

Easy to clone virtual machines with software



Allow multiple machines run at same time in total isolation

Able to host different operating systems on the same physical machine

©Simplilearn. All rights reserved.

Comparison between Docker and Virtual Machines

	Docker	Virtual Machine
Operating System	Basic Operating system based images	Any
Startup Time	1-2 s	1-2 min
Disk Usage	Small	Large
Memory Usage	Small	Large
Configuration setup	Easy and simple	Complex
Hardware Requirement	Less	More
Security	Not Matured	Mature and Better

©Simplilearn. All rights reserved.

Containerization with Docker

Docker installation on Multiple OS

©Simplilearn. All rights reserved.

Docker on Linux

Docker was initially launched for Unix-based environment. Later, it was released for other platforms. Docker installation is available for Ubuntu, centos, Red hat and other Unix based platforms.

```
root@docker:~# apt update
Hit:1 http://us-east1.gce.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://us-east1.gce.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://us-east1.gce.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Hit:4 http://archive.canonical.com/ubuntu bionic InRelease
Get:5 http://us-east1.gce.archive.ubuntu.com/ubuntu bionic-updates/main Sources [211 kB]
Get:6 http://us-east1.gce.archive.ubuntu.com/ubuntu bionic-updates/universe Sources [97.5 kB]
Get:7 http://us-east1.gce.archive.ubuntu.com/ubuntu bionic-updates/multiverse Sources [3212 B]
Get:8 http://us-east1.gce.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [437 kB]
Get:9 http://us-east1.gce.archive.ubuntu.com/ubuntu bionic-updates/main Translation-en [164 kB]
Get:10 http://us-east1.gce.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [575 kB]
Get:11 http://us-east1.gce.archive.ubuntu.com/ubuntu bionic-updates/universe Translation-en [154 kB]
Get:12 http://us-east1.gce.archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages [5696 B]
Get:13 http://security.ubuntu.com/ubuntu bionic-security InRelease [83.2 kB]
Get:14 http://security.ubuntu.com/ubuntu bionic-security/main Sources [61.0 kB]
Get:15 http://security.ubuntu.com/ubuntu bionic-security/universe Sources [23.0 kB]
Get:16 http://security.ubuntu.com/ubuntu bionic-security/multiverse Sources [1332 B]
Get:17 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [205 kB]
Get:18 http://security.ubuntu.com/ubuntu bionic-security/main Translation-en [80.9 kB]
Get:19 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [96.6 kB]
Get:20 http://security.ubuntu.com/ubuntu bionic-security/universe Translation-en [54.6 kB]
Get:21 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [1448 B]
Fetched 2417 kB in 1s (2053 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
20 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@docker:~# apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
docker.io is already the newest version (18.06.1-0ubuntu1~18.04.1).
The following package was automatically installed and is no longer required:
  grub-pc-bin
Use 'apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 20 not upgraded.
root@docker:~#
```

©Simplilearn. All rights reserved.

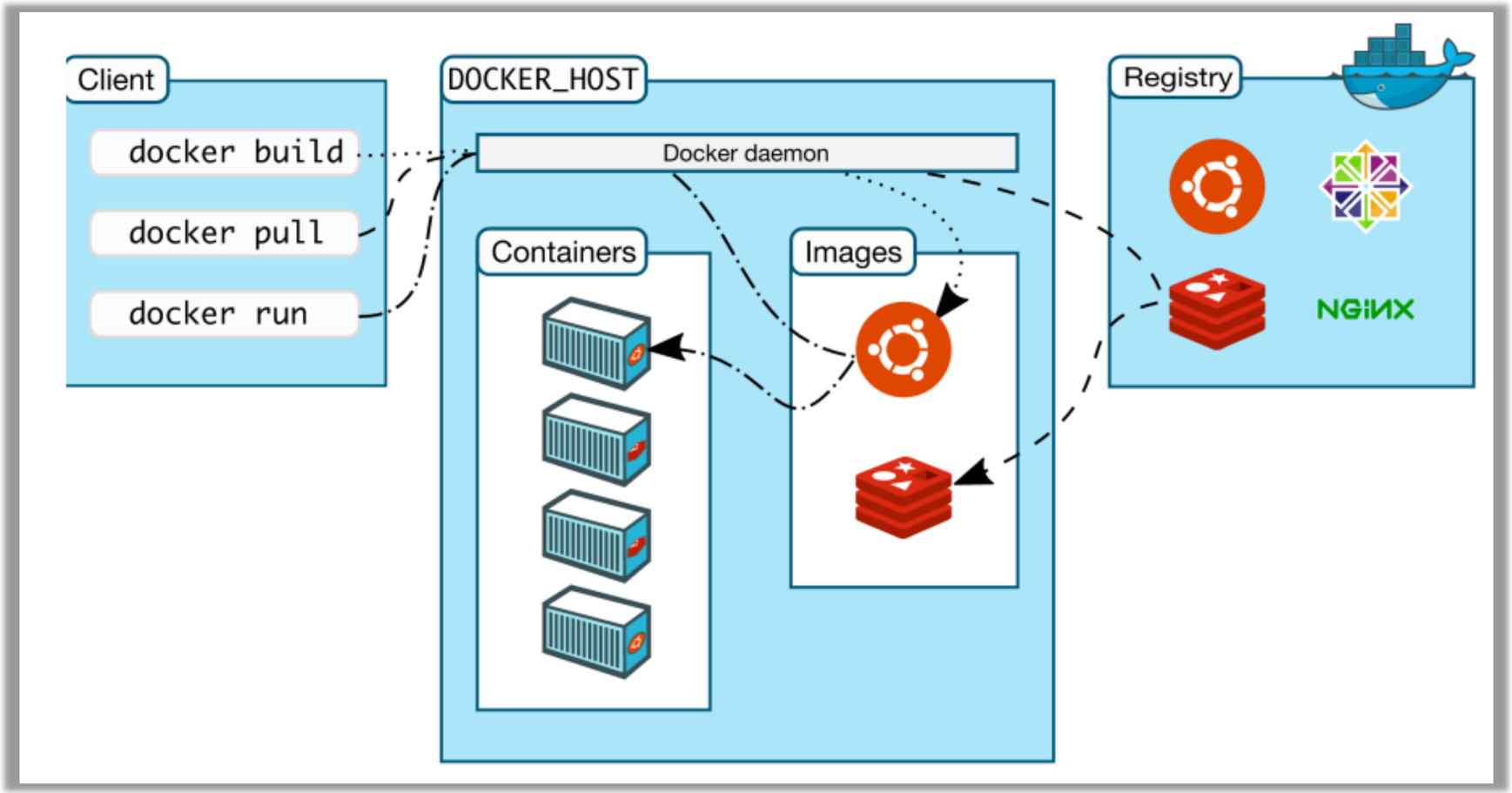
Docker on Desktop

- Supports desktop installations for Mac and Windows platforms
- Availability of .DMG and .EXE file support for Mac and Windows respectively
- Requires minimum 4 GB of RAM for both Mac and Windows
- Needs xhyve hypervisor in Mac and Hyper V in Windows to implement Docker
- Windows starter and Windows home versions are not supported by Docker

©Simplilearn. All rights reserved.

Docker Architecture

Docker is lightweight container service and the architecture of Docker is quiet simple. Docker follow client-server architecture.



©Simplilearn. All rights reserved.

Docker Architecture (Contd.)

- Docker Client
  - It is a command-line interface connected to Docker daemon using REST API.
  - It is used to manage images and containers on Docker host.
- Docker Daemon
  - Docker Daemon is kind of background process running on server and responsible for keeping applications and Docker images up and running.
  - In case, it is down, all Docker-hosted applications will also be down.

©Simplilearn. All rights reserved.

Docker Architecture (Contd.)

- Docker Registry
  - Registry is used to store public and private Docker images.
  - You can also download already published Docker images from Docker Hub.
  - You can pull enterprise Docker images from Docker Store such as Oracle, IBM Web Sphere or other proprietary software.

©Simplilearn. All rights reserved.

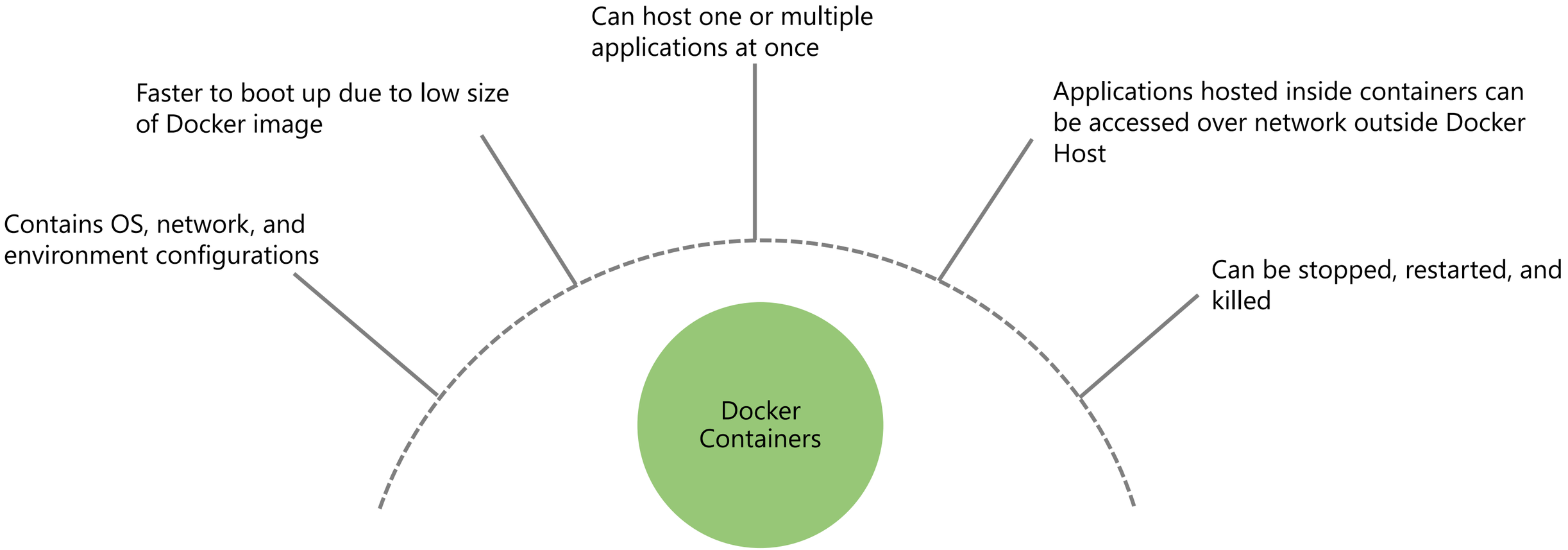
Docker Images

1. Read only template combination of multiple layers
2. Comprises of basic Operating System binaries
3. We can store Docker images on Docker hub or private repositories
4. Hosts software applications, web containers, application servers, and even databases
5. Exists for various open source tools in Docker hub and its free to pull those images
6. Images can be Built using custom Dockerfile
7. Smaller in size which helps to store and run Docker images easily

©Simplilearn. All rights reserved.

Docker Containers

An instance of Docker image is called Docker Container. Docker containers host different types of applications inside them.



©Simplilearn. All rights reserved.

Execution on Docker Images

- The command to check all existing Docker images on Docker hosts is “docker images”.
- The command to pull a Docker image from either Docker hub or private repository is docker pull “docker pull <image_name>:<tag>”.
- You can get the image name from Docker hub or you can also search it using the command “docker search <image_name>”.
- Docker pull command will first check the image on local system. In case it is not found there, Docker image will be pulled from Docker hub.

©Simplilearn. All rights reserved.

Execution on Docker Images (Contd.)

```
root@docker:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
root@docker:~# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
473ede7ed136: Pull complete
c46b5fa4d940: Pull complete
93ae3df89c92: Pull complete
6b1eed27cade: Pull complete
Digest: sha256:29934af957c53004d7fb6340139880d23fb1952505a15d69a03af0d1418878cb
Status: Downloaded newer image for ubuntu:latest
root@docker:~# docker pull ubuntu:18.04
18.04: Pulling from library/ubuntu
Digest: sha256:29934af957c53004d7fb6340139880d23fb1952505a15d69a03af0d1418878cb
Status: Downloaded newer image for ubuntu:18.04
root@docker:~# docker pull ubuntu:18.10
18.10: Pulling from library/ubuntu
72f45ff89b78: Pull complete
9f034a33b165: Pull complete
386fee7ab4d3: Pull complete
f941b4ac6aa8: Pull complete
Digest: sha256:ef5d78b10fcada65484b2dd9cd6e0704fba36021a101b2b52023a1566c4ebad4
Status: Downloaded newer image for ubuntu:18.10
root@docker:~# docker pull centos
Using default tag: latest
latest: Pulling from library/centos
aeb7866da422: Pull complete
Digest: sha256:67dad89757a55bfdfabec8abd0e22f8c7c12a1856514726470228063ed86593b
Status: Downloaded newer image for centos:latest
root@docker:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              18.10               7e10e8cb09ba        3 weeks ago         73.7MB
ubuntu              18.04               ea4c82dcd15a        3 weeks ago         85.8MB
ubuntu              latest              ea4c82dcd15a        3 weeks ago         85.8MB
centos              latest              75835a67d134        5 weeks ago         200MB
root@docker:~#
```

©Simplilearn. All rights reserved.

Execution on Docker Images (Contd.)

- You can remove Docker image using command below:

  docker rmi <image_name>:<tag>

- Docker images consumes local disk space, you can have to continuously work on maintenance.

- You can also export Docker images to tar file so that they can be transferred from one server to another.

  docker save <image_name>:<tag> -o <tar_filename>

- Docker image can be imported with all layers from tar file using command below. All previous details regarding image will be loaded automatically.

  docker load –I <tar_filename>

©Simplilearn. All rights reserved.

Execution on Docker Images (Contd.)

```
root@docker:~# docker images
oREPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              18.10               7e10e8cb09ba        3 weeks ago         73.7MB
ubuntu              18.04               ea4c82dcd15a        3 weeks ago         85.8MB
ubuntu              latest              ea4c82dcd15a        3 weeks ago         85.8MB
centos              latest              75835a67d134        5 weeks ago         200MB
root@docker:~# docker rmi ubuntu
Untagged: ubuntu:latest
root@docker:~# docker rmi ubuntu:18.04
Untagged: ubuntu:18.04
Untagged: ubuntu@sha256:29934af957c53004d7fb6340139880d23fb1952505a15d69a03af0d1418878cb
Deleted: sha256:ea4c82dcd15a33e3e9c4c37050def20476856a08e59526fbe533cc4e98387e39
Deleted: sha256:2ac9356b41d2d032dc980b6ee2b2a911790a47b59b4fcfd92e52a0729a389403
Deleted: sha256:f19c7e29a7e3fbaf44997bab14791e3d3d26d689bc3e2e720a9d5d8a77f68d6c
Deleted: sha256:b951bb1959dc1a5dfeda46229b36b33c4cc01e3f682ee1b77af8dab1cc7cf8a3
Deleted: sha256:102645f1cf722254bbfb7135b524db45fbbac400e79e4d54266c000a5f5bc400
root@docker:~# docker save centos -o centos.tar
root@docker:~# docker rmi centos
Untagged: centos:latest
Untagged: centos@sha256:67dad89757a55bfdfabec8abd0e22f8c7c12a18565147264702280 63ed86593b
Deleted: sha256:75835a67d1341bdc7f4cc4ed9fa1631a7d7b6998e9327272afea342d90c4ab6d
Deleted: sha256:f972d139738dfcd1519fd2461815651336ee25a8b54c358834c50af094bb262f
root@docker:~# docker load -i centos.tar
f972d139738d: Loading layer [==================================================>]  208.8MB/208.8MB
Loaded image: centos:latest
root@docker:~# docker images | grep centos
centos              latest              75835a67d134        5 weeks ago         200MB
root@docker:~#
```

©Simplilearn. All rights reserved.

simplilearn

Execution on Docker Containers

- The ‘run’ command is used to create an instance of Docker image.

  docker run -it <image_name> bash

  - -i or --interactive keeps STDIN open
  - -t or --tty allocates a pseudo TTY

- You can also run Docker image in detached mode using flag ‘d’.

  docker run -d <image_name>

- Docker container can be easily removed from Docker host using command below:

  docker rm <container_id> -f

- Docker container can be stopped, killed, and restarted using commands below:

  docker stop <container_id>
  docker restart <container_id>
  docker kill <container_id>

©Simplilearn. All rights reserved.

Execution on Docker Containers (Contd.)

```
root@docker:~# docker run -it centos bash
[root@4a5119324fbd /]# uname -a
Linux 4a5119324fbd 4.15.0-1023-gcp #24-Ubuntu SMP Wed Oct 10 13:28:59 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux
[root@4a5119324fbd /]# exit
exit
root@docker:~# docker run -d jenkins
47b089a3c311f34d72e3f65b2d18b7e2f2b8663aedbb8779c80b78bd48f3f32e
root@docker:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                       PORTS
47b089a3c311        jenkins             "/bin/tini -- /usr/l…"   10 seconds ago      Up 10 seconds                8080/tcp, 5
4a5119324fbd        centos              "bash"                   36 seconds ago      Exited (0) 29 seconds ago
ky
0f896a8a9c81        centos              "bash"                   5 minutes ago       Exited (0) 5 minutes ago
root@docker:~# docker rm 47b089a3c311 4a5119324fbd 0f896a8a9c81 -f
47b089a3c311
4a5119324fbd
0f896a8a9c81
root@docker:~# docker run -d jenkins
5a149d38a29d7673b75624a147acec3deb3371d98cb55369e34181cf614a65fd
root@docker:~# docker stop 5a149d38a29d7673b75624a147acec3deb3371d98cb55369e34181cf614a65fd
5a149d38a29d7673b75624a147acec3deb3371d98cb55369e34181cf614a65fd
root@docker:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                          PORTS
5a149d38a29d        jenkins             "/bin/tini -- /usr/l…"   26 seconds ago      Exited (143) 10 seconds ago
root@docker:~# docker start 5a149d38a29d
5a149d38a29d
root@docker:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
5a149d38a29d        jenkins             "/bin/tini -- /usr/l…"   48 seconds ago      Up 4 seconds        8080/tcp, 50000/tcp
root@docker:~#
```

©Simplilearn. All rights reserved.

Execution on Docker Containers (Contd.)

- You can connect back to Docker container running in detached mode using command below:

  docker exec -it <container_id> bash

  - -i or --interactive keeps STDIN open
  - -t or --tty allocates a pseudo TTY

- Docker container can be paused and unpaused using the commands below:

  docker pause <container_id>

  docker unpause <container_id>

©Simplilearn. All rights reserved.

Execution on Docker Containers (Contd.)

```
root@docker:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
5a149d38a29d        jenkins             "/bin/tini -- /usr/l…"   8 minutes ago       Up 7 minutes        8080/tcp, 50000/tcp
root@docker:~# docker exec -it 5a149d38a29d bash
jenkins@5a149d38a29d:/$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
jenkins      1     0  0 15:36 ?        00:00:00 /bin/tini -- /usr/local/bin/jenkins.sh
jenkins      6     1  5 15:36 ?        00:00:28 java -jar /usr/share/jenkins/jenkins.war
jenkins     14     6  0 15:36 ?        00:00:00 [jenkins.sh] <defunct>
jenkins     72     0  0 15:44 pts/0    00:00:00 bash
jenkins     79    72  0 15:44 pts/0    00:00:00 ps -ef
jenkins@5a149d38a29d:/$ exit
exit
root@docker:~# docker pause 5a149d38a29d
5a149d38a29d
root@docker:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                     PORTS
5a149d38a29d        jenkins             "/bin/tini -- /usr/l…"   9 minutes ago       Up 8 minutes (Paused)      8080/tcp, 50000/t
root@docker:~# docker unpause 5a149d38a29d
5a149d38a29d
root@docker:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
5a149d38a29d        jenkins             "/bin/tini -- /usr/l…"   9 minutes ago       Up 8 minutes        8080/tcp, 50000/tcp
root@docker:~#
```

©Simplilearn. All rights reserved.

simplilearn

Assisted Practice

Duration: 30 mins

MySQL Database in Docker Container

Problem Statement: You are given a project to demonstrate the execution of MySQL database in Docker container.

Access: Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

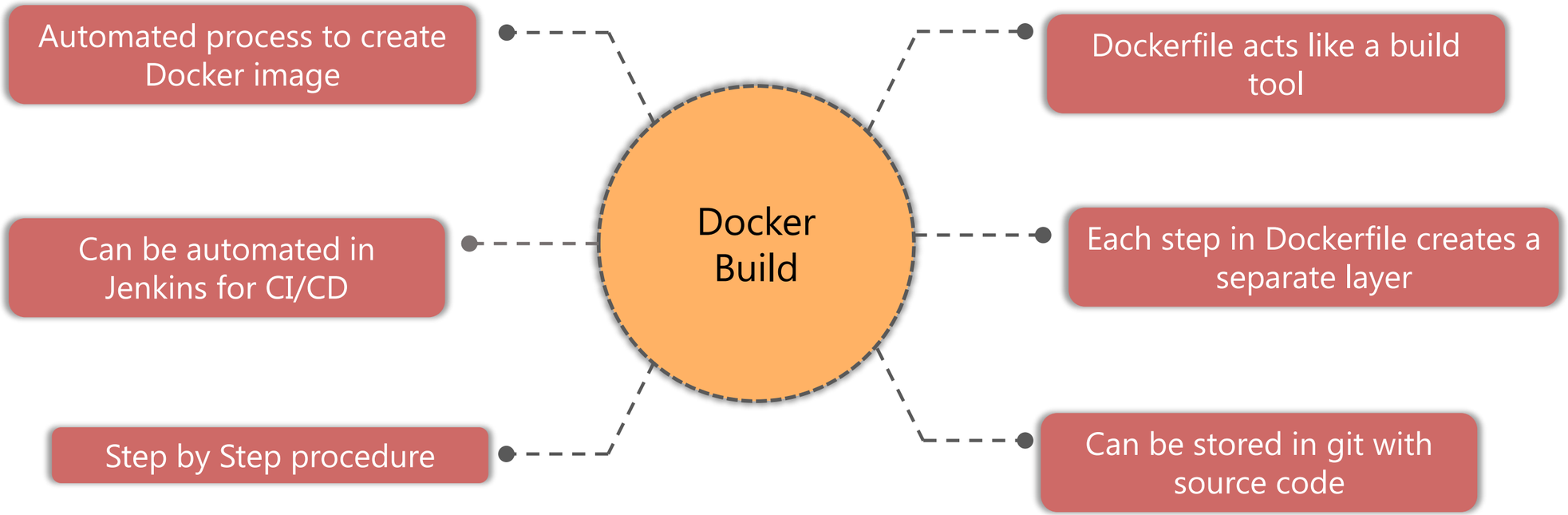
©Simplilearn. All rights reserved.

Assisted Practice: Guidelines to Demonstrate MySQL in Docker

1. Login to your Ubuntu Lab and open the terminal.
2. Create a folder to store all the files generated during the demonstration.
3. Clone the repository from Git and navigate inside the directory.
4. Grant the read, write, and execute permissions to the “runserver_first” file.
5. Execute the client and log the data in the database.
6. Execute the MySQL database command and enter the login password to use it.
7. Explore the MySQL database and confirm that it works efficiently.
8. Exit the MySQL Client.

©Simplilearn. All rights reserved.

Docker Automated Build



©Simplilearn. All rights reserved.

Dockerfile

- Dockerfile is used to build custom Docker images. You can add the application source or configure and install any application software in Docker image.
- There are a few attributes that can be used in Dockerfile:
  - FROM → Used to define base image information
  - MAINTAINER → Used to display the name and e-mail details of build owner
  - SHELL → Used to set the default shell
  - COPY → Used to transfer the local file to Docker container

©Simplilearn. All rights reserved.

Dockerfile (Contd.)

- ADD → This is similar to COPY, but you can use URL instead of local file. Archived files get unachieved automatically once transferred inside container to a specific folder
- RUN → Run command used to execute command inside Docker image
- ENV → Used to create environment variables inside Docker image
- COMMAND → Command defines the executable to be execute while running Docker image
- WORKDIR → Used to configure working DIR during executing Docker build command
- EXPOSE → Exposes port on which application will be hosted

©Simplilearn. All rights reserved.

Dockerfile (Contd.)

```
root@docker:~# docker build -t docker_apache .
Sending build context to Docker daemon  212.2MB
Step 1/12 : FROM ubuntu:18.04
 ---> ea4c82dcd15a
Step 2/12 : MAINTAINER SimpliLearn
 ---> Using cache
 ---> 70ac7fca0837
Step 3/12 : RUN apt-get update && apt-get install -y apache2 && apt-get clean && rm -rf /var/lib/apt/lists/*
 ---> Using cache
 ---> 1fc155264ef1
Step 4/12 : ENV APACHE_RUN_USER www-data
 ---> Using cache
 ---> 91f16ae7f5a8
Step 5/12 : ENV APACHE_RUN_GROUP www-data
 ---> Using cache
 ---> bb40bb196433
Step 6/12 : ENV APACHE_LOG_DIR /var/log/apache2
 ---> Using cache
 ---> a461eef9c4e0
Step 7/12 : WORKDIR /var/www/html
 ---> Using cache
 ---> 9ae2833a4176
Step 8/12 : COPY index.html /var/www/html
 ---> Using cache
 ---> cf7fafb6b846
Step 9/12 : ADD *.html /var/www/html
 ---> Using cache
 ---> e613dd7ed4d4
Step 10/12 : EXPOSE 80
 ---> Using cache
 ---> b8dd45cb6cb5
Step 11/12 : SHELL ["/bin/sh", "-c"]
 ---> Using cache
 ---> 33bbc46aa9de
Step 12/12 : CMD ["/usr/sbin/apache2", "-D", "FOREGROUND"]
 ---> Using cache
 ---> eb2d5063ce91
Successfully built eb2d5063ce91
Successfully tagged docker_apache:latest
root@docker:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
docker_apache       latest              eb2d5063ce91        12 seconds ago      182MB
ubuntu              18.04               ea4c82dcd15a        3 weeks ago         85.8MB
jenkins             latest              cd14cecfdb3a        4 months ago        696MB
root@docker:~#
```

©Simplilearn. All rights reserved.

Dockerfile Build

- Dockerfile is used to create custom Docker image using Docker build command.
- Docker build command follows syntax below:

  docker build –t

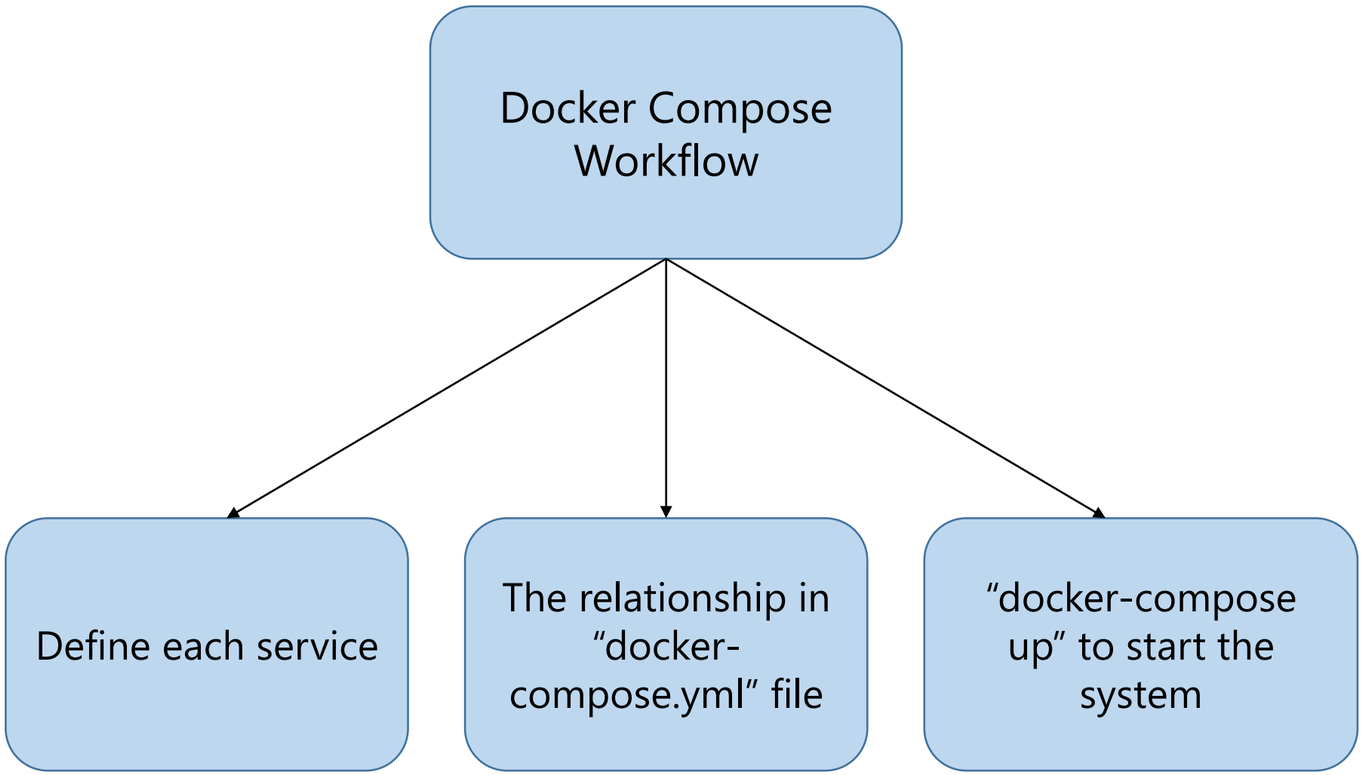
  <image_name>:<tag> .
- This command creates an intermediate container, and that container will automatically be removed once image build is complete.

```
root@docker:~# cat Dockerfile
FROM ubuntu:18.04

MAINTAINER SimpliLearn

RUN apt-get update && apt-get install -y apache2 && apt-get clean && rm -rf /var/lib/apt/lists/*

ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2

WORKDIR /var/www/html

COPY index.html /var/www/html
ADD *.html /var/www/html

EXPOSE 80

SHELL ["/bin/sh", "-c"]
CMD ["/usr/sbin/apache2", "-D", "FOREGROUND"]
root@docker:~#
```

©Simplilearn. All rights reserved.

Docker Compose

- Docker Compose is used to define and run complex applications in Docker.
- The main functions of Docker Compose are building images, scaling containers, and rerunning containers that have stopped.
- It has a similar set of commands like Docker.
- Entire configuration can be stored and versioned along with the project.



©Simplilearn. All rights reserved.

Assisted Practice

Duration: 30 mins

Using Docker Compose to Manage a Container

Problem Statement: You are given a project to demonstrate the use of Docker Compose to manage a container.

Access: Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

©Simplilearn. All rights reserved.

Assisted Practice: Guidelines to Demonstrate the Use of Docker Compose

1. Login to your Ubuntu Lab and open the terminal.

2. Create a Dockerfile and build the Docker image with it.

3. Run the image and connect to the SSH server.

4. Exit the container once you confirm that you have logged in.

5. Create a “docker-compose.yml” file and connect to the server.

6. Stop the server and exit the container.

©Simplilearn. All rights reserved.

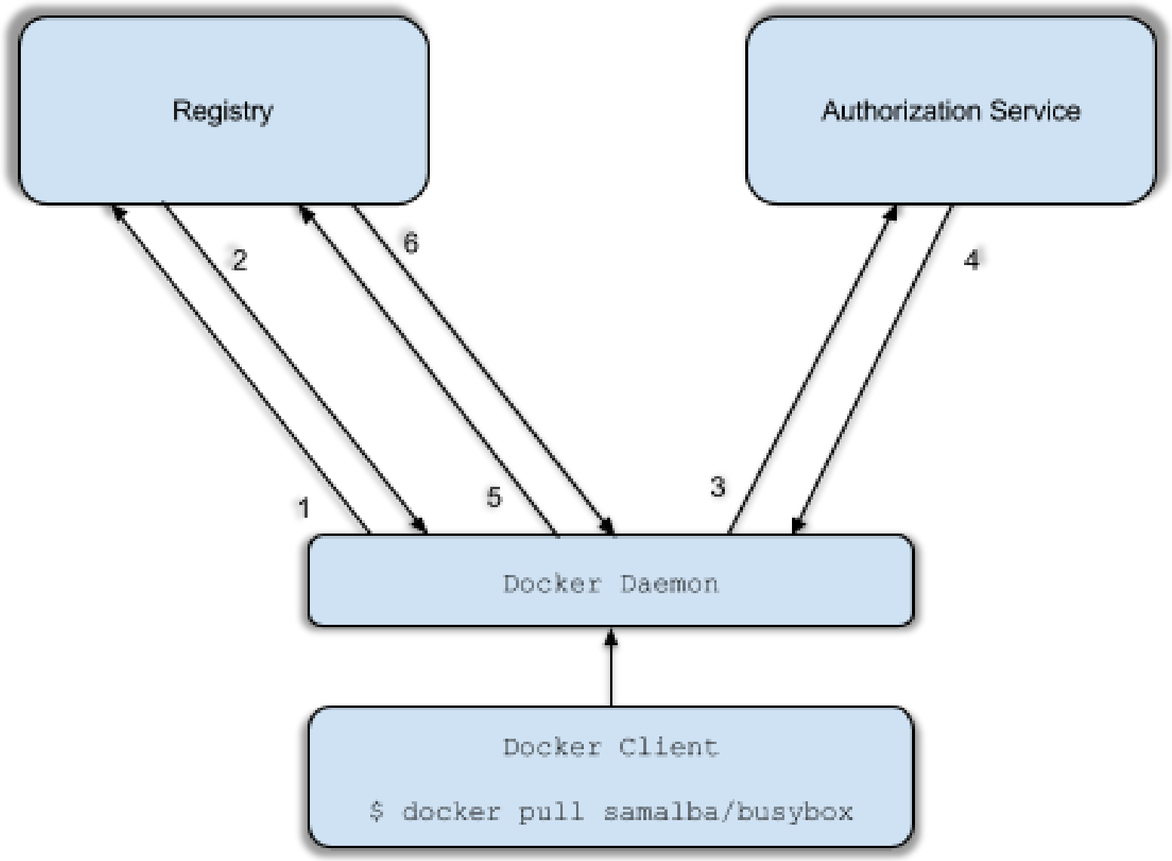
Containerization with Docker

Docker Registry

©Simplilearn. All rights reserved.

Docker Registry

- Docker Registry helps you share Docker images.
- You can also have different collaborators to a single Docker image.
- Both public and private both Docker images can be uploaded to Docker hub.
- Docker store consists of enterprise and trusted Docker images.



©Simplilearn. All rights reserved.

Docker Registry (Contd.)

- You can upload various versions of the same Docker image.
- The free account of Docker hub supports unlimited public Docker images and only one private Docker image.
- Some of the private Docker Registries, other than Docker hub and Docker store, are listed as below:
  - Google Container Registry
  - Amazon ECR
  - Jfrog Artifactory
  - Nexus Artifactory

©Simplilearn. All rights reserved.

Assisted Practice

Duration: 30 mins

Run Docker Registry with Centos

Problem Statement: You are given a project to demonstrate and run a Docker registry from an official registry image.

Access: Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

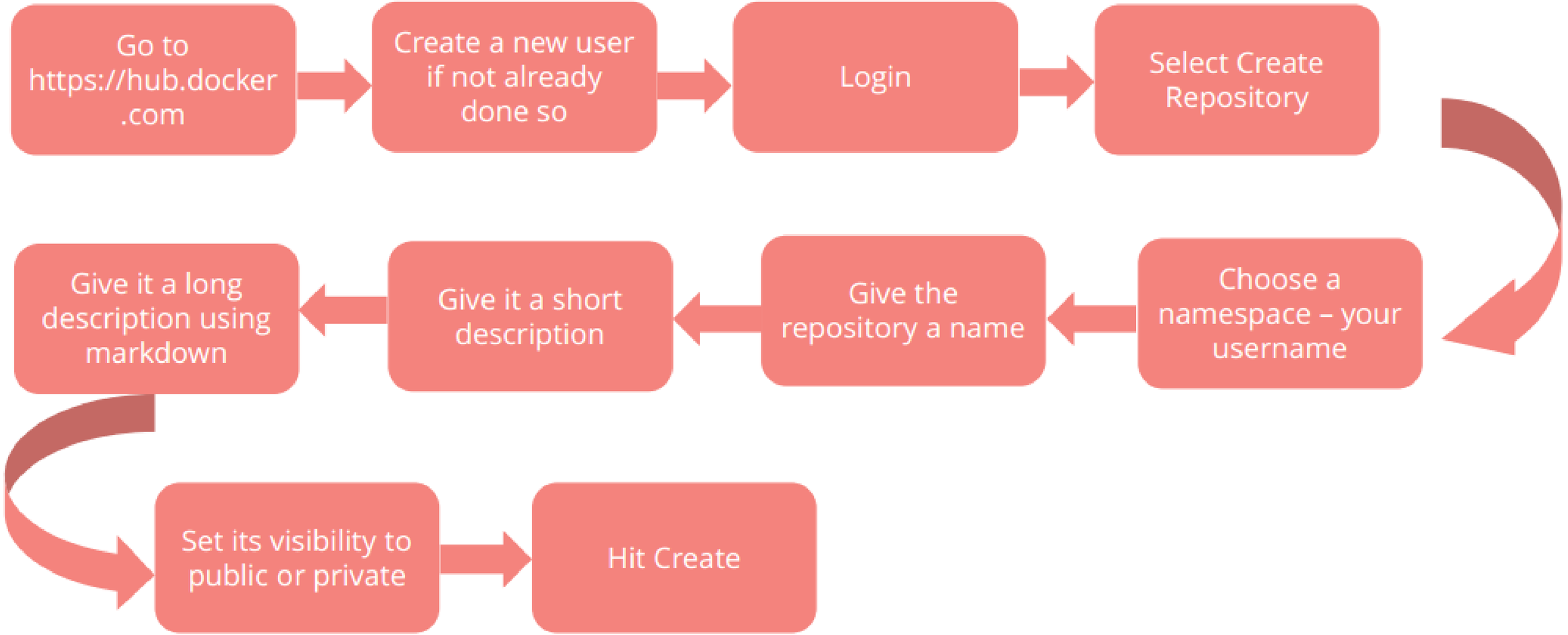
©Simplilearn. All rights reserved.

Assisted Practice: Guidelines to Run Docker Registry with Centos

1. Login to your Ubuntu Lab and open the terminal.
2. Pull a recent version of the Centos Linux container.
3. Run the registry in a new Docker container. Pull a new image from Docker Hub.
4. Push the image to the local registry.
5. Remove the image from the local cache.
6. Pull the image from the local repository.
7. Run the new Docker image.
8. Exit the container once the image is confirmed.

©Simplilearn. All rights reserved.

Docker Hub Configuration: Step by Step Procedure



©Simplilearn. All rights reserved.

Docker Hub Login

Docker login command is used to connect to Docker hub from Docker client CLI. Before this can be done, you must tag existing image with Docker image concatenated with Docker username.

docker tag <image_name><tag> <docker_hub_username>/ <image_name><tag>
docker push <docker_hub_username>/ <image_name><tag>

```
root@docker:~# docker tag docker_apache anujsharma1990/docker_apache
root@docker:~# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: anujsharma1990
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@docker:~# docker push anujsharma1990/docker_apache
The push refers to repository [docker.io/anujsharma1990/docker_apache]
e7e6e790a930: Pushed
735b1c81b582: Pushed
76c033092e10: Mounted from library/ubuntu
2146d867acf3: Mounted from library/ubuntu
ae1f631f14b7: Mounted from library/ubuntu
102645f1cf72: Mounted from library/ubuntu
latest: digest: sha256:7d8cccf25ef3b9f1578f0efb2d26a0c67cacc7c168709c12310c0ad47ad821e9 size: 1776
root@docker:~#
```

©Simplilearn. All rights reserved.

Containerization with Docker

Docker Networking

©Simplilearn. All rights reserved.

Docker Networking

Docker Networking helps Docker container to interact with other microservices over the network. Access to multiple applications hosted inside the Docker container is possible through Docker Networking.

There are four types of Docker networks:

- Bridge Mode: It is the default network available for Docker containers. It helps to access Docker container using same Docker host IP address and port.

©Simplilearn. All rights reserved.

Docker Networking (Contd.)

- Host Mode: This will inherit all network resources directly from host and can become exposed directly to public network. This is faster than Bridge mode, but there might be some security issues.

- Container Mode: This mode helps to have a unique IP address for each Docker container. You can not access the application through external server, but you can access it using Docker container IP through internal server.

- None: in this mode, networking gets switched off and Docker containers could not be able to connect to the network anymore. This helps to setup the applications network is not needed.

©Simplilearn. All rights reserved.

Execution on Docker Networking

You can use commands below to host an application in Docker container and also access it using a specific port from an external server. You can easily access application from anywhere through the internet.

```
root@docker:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
root@docker:~# docker run -d -p 8080:8080 --name jenkins jenkins
092942ab1619b3c853ed2c701b6060750c624562a0f3e9f4b2cb2607ab88f35d
root@docker:~# curl localhost:8080
<html><head><meta http-equiv='refresh' content='1;url=/login?from=%2F'/><script>window.location.replace('/login?from=%2F');</script></head><body style='bac
kground-color:white; color:white;'>


Authentication required
<!--
You are authenticated as: anonymous
Groups that you are in:

Permission you need to have (but didn't): hudson.model.Hudson.Administer
-->

</body></html>

    root@docker:~#
```

©Simplilearn. All rights reserved.

Assisted Practice

Duration: 45 mins

Demonstrate Docker Networking with Two SSHs

Problem Statement: You are given a project to demonstrate Docker networking and its configuration in a container.

Access: Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

©Simplilearn. All rights reserved.

Assisted Practice: Guidelines to Demonstrate Docker Networking

1. Login to your Ubuntu Lab and open the terminal.
2. Create a Centos container and install the net tools.
3. Confirm the IP address, hostname, and exit the container.
4. Commit the container to an image.
5. Create a bridge network and find its IP range.
6. Connect the network from the second SSH window.
7. Confirm the IP addresses and hostnames.
8. Disconnect the connection and exit the container.

©Simplilearn. All rights reserved.

Key Takeaways

You are now able to:

- Describe the importance of Docker over Virtual Machines
- Demonstrate on Docker images, Containers, and Docker Registry
- Demonstrate on Docker Compose and Docker Hub
- Describe the importance of Docker Networking

©Simplilearn. All rights reserved.

Knowledge Check

©Simplilearn. All rights reserved.

Knowledge Check

1

Which one of the following platforms is supported by Docker?

a. Linux

b. Windows

c. Mac OS

d. All of the Above

©Simplilearn. All rights reserved.

Knowledge Check 1	**Which one of the following platforms is supported by Docker?**

a. Linux

b. Windows

c. Mac OS

d. All of the Above

The correct answer is **d. All of the Above**

Docker is supported by all major platforms, including Linux, windows, and MAC OS. You can download and install Docker on any of these platforms.

©Simplilearn. All rights reserved.

Knowledge Check 2	**Docker containers running on the same host share operating system kernel, and this makes them fast.**

a. True

b. False

©Simplilearn. All rights reserved.

Knowledge Check

2

Docker containers running on the same host share operating system kernel, and this makes them fast.

a. True

b. False

The correct answer is **a. True**

Docker uses host machine kernel; this helps containers to load faster and effectively.

©Simplilearn. All rights reserved.

Knowledge Check

3

Which one of below is a tool used for defining and creating multiple containers?

a. Docker Swarm

b. Docker Hub

c. Docker Cloud

d. Docker Compose

©Simplilearn. All rights reserved.

Knowledge Check

3

Which one of below is a tool used for defining and creating multiple containers?

a. Docker Swarm

b. Docker Hub

c. Docker Cloud

d. Docker Compose

The correct answer is **d. Docker Compose**

Docker compose is the tool that can be used to define multiple containers; the containers can run using Docker compose command.

©Simplilearn. All rights reserved.

Knowledge Check

4

Which of the following is a Cloud Service provided by AWS for hosting Docker images?

a. Docker Hub

b. AWS ECR

c. Google Container Registry

d. Jfrog Artifactory

©Simplilearn. All rights reserved.

Knowledge Check

4

Which of the following is a Cloud Service provided by AWS for hosting Docker images?

a. Docker Hub

b. AWS ECR

c. Google Container Registry

d. Jfrog Artifactory

The correct answer is **b. AWS ECR**

Amazon ECR service is used to create private repositories; it will help you host your application in Docker images easily.

©Simplilearn. All rights reserved.

Lesson-End Project

Build a Docker Image and Deploy to the Docker Hub

Duration: 40 mins

Problem Statement:

Perform the following actions:

- Create a GitHub repository to save the code and to build the image.
- Create an account on Docker Hub.
- Install Docker on your machine.
- Login to Docker Hub.

Access: Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

©Simplilearn. All rights reserved.

Thank You

©Simplilearn. All rights reserved.