

7.1 Learning Objectives

Configuration Management

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

- Describe the importance of Docker over virtual machines
- Demonstrate Docker images, containers, and Docker Registry
- Demonstrate Docker Compose and Docker Hub
- Describe the importance of Docker networking

7.2 Overview of Docker

Docker



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

Docker Features



Is a lightweight container service



Helps to resolve the versioning problem with software



Helps to implement microservices



Is fast, reliable, and highly scalable



Has source code with dependencies bundled up together inside container



Is independent of the operating system and supports platforms based on both Windows and Unix



Has containers running in isolation

Docker Workflow

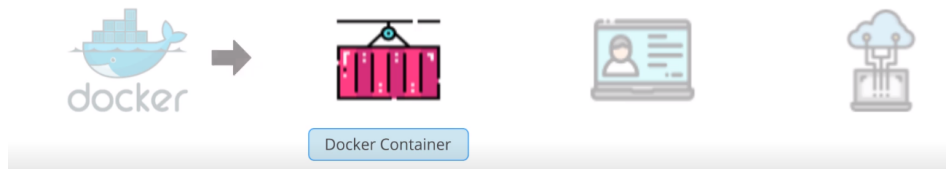
Pull or build a Docker image from the Docker Hub, and bundle source code with dependencies inside this Docker image.



Docker Hub



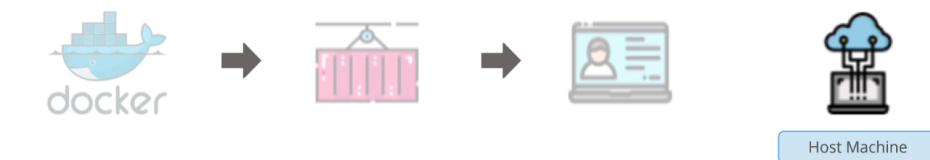
Run the pulled Docker image on a Docker host and start a container on the host.



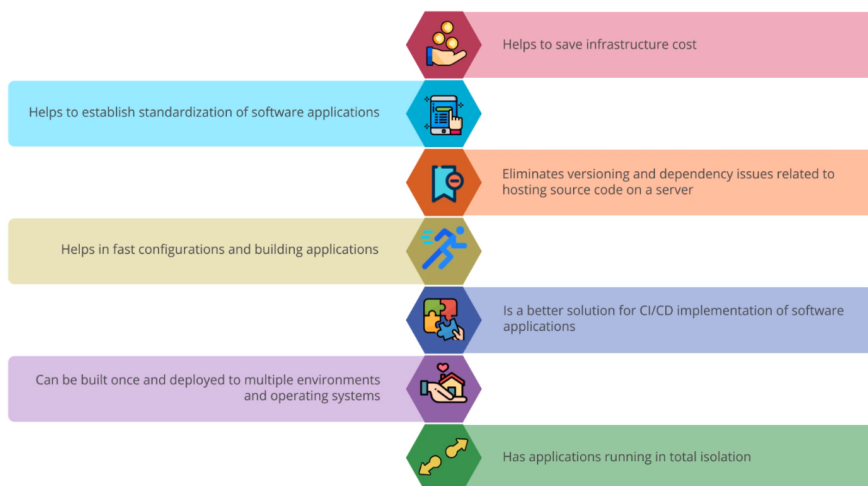
The container will be running in background, and it can be killed or stopped anytime.



Applications hosted in the Docker container can be accessed using the host machine IP address and port address.

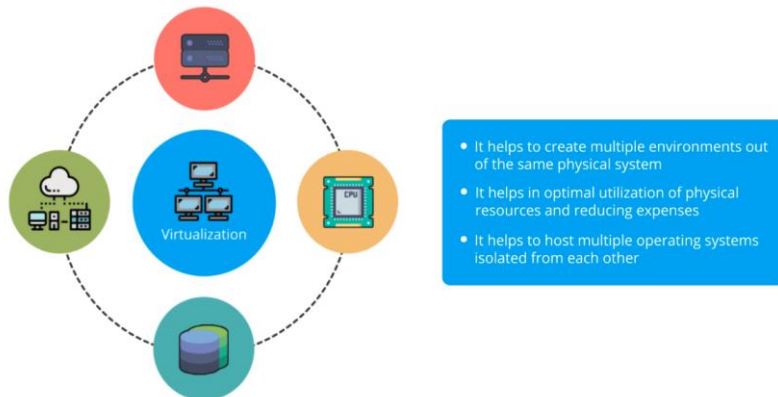


Docker Benefits

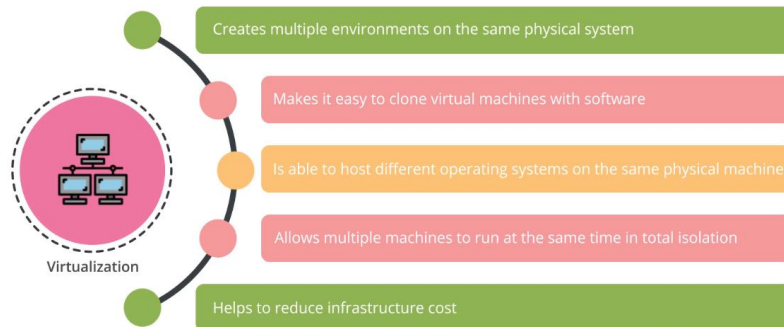


7.3 Overview of Virtualization

Virtualization

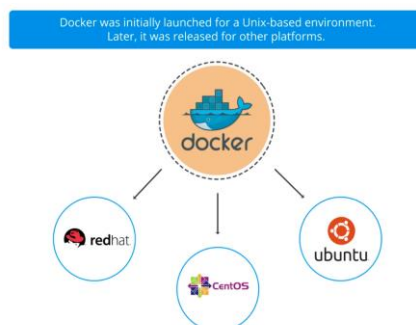


Advantages of Virtualization



7.4 Docker Installation on Multiple OS

Docker on Linux



```
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
```

Docker on Desktop



Supports desktop installations for both Mac and Windows platforms

Makes .DMG and .EXE file support available for Mac and Windows respectively

Requires minimum 4GB of RAM for both Mac and Windows

Needs xhyve hypervisor in Mac and Hyper V in Windows to implement Docker

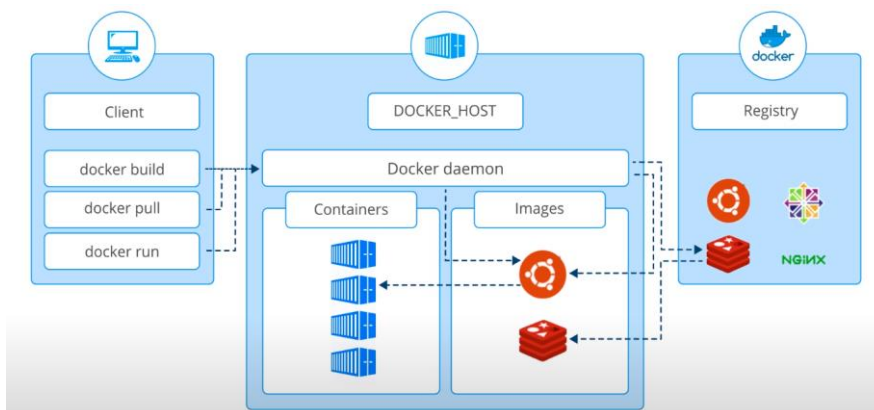
Docker Architecture



Docker is a lightweight container service

Architecture is simple

Docker follows the client-server model



Docker Images



Read-only template combination of multiple layers



Basic operating system binaries



Docker images can be stored on Docker Hub or private repositories



Docker images host software applications, web containers, application servers, and databases



Exist for various open-source tools in the Docker Hub, and it's free to pull those images

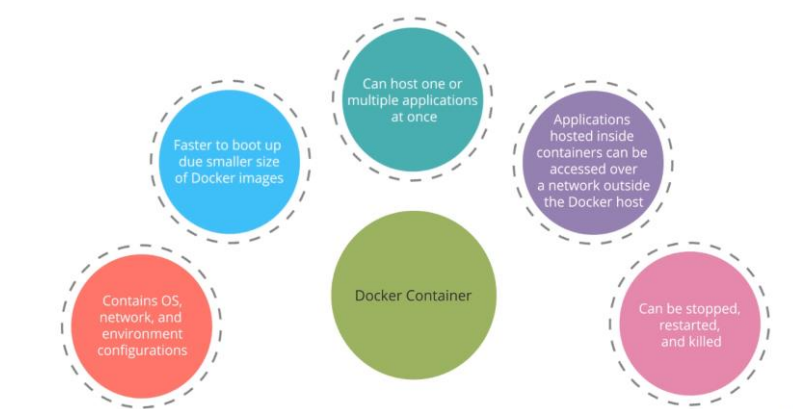


Can be built using a custom Dockerfile



Help users store and run them easily, as they are smaller in size

Docker Containers



Execution of Docker Images

You can check all existing Docker images on Docker hosts using the command:	<code>docker images</code>
You can pull a Docker image from either the Docker Hub or a private repository using the command:	<code>docker pull <image_name>:<tag></code>
You can get the image name from Docker Hub, or you can also search it using the command:	<code>docker search <image_name></code>
You can remove a Docker image using the command:	<code>Docker rmi <image_name>:<tag></code>
You can also export Docker images to a tar file so that they can be transferred from one server to another using the command:	<code>docker save <image_name>:<tag> -o <tar_filename></code>
You can import a Docker image with all the layers from a tar file using the command:	<code>docker load -i <tar_filename></code>

Execution of Docker Containers

An instance of Docker image can be created using the "run" command below:

```
docker run -it <image_name> bash
```

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

A Docker image can also be run in the detached mode using flag "d":

```
docker run -d <image_name>
```

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

```
docker rm <container_id> -f
```

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

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

Users can connect back to a Docker container running in detached mode using the command below:

```
docker exec -it <container_id> bash
```

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

A Docker container can be paused and unpaused using the commands:

```
docker pause <container_id>  
docker unpause <container_id>
```

7.5 MySQL Database in Docker Container

Praxisbeispiel

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

7.6. Docker Installation on Multiple OS

Docker Automated Build



Docker File

- Dockerfile is used to build custom Docker images. You can add the application source code, or configure and install any application software into a 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 email ID of build owner
- SHELL → Used to set the default shell
- COPY → Used to transfer the local file to Docker container

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

Docker File Build

- Dockerfile is used to create a custom Docker image using Docker build command.
- Docker build command follows the 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

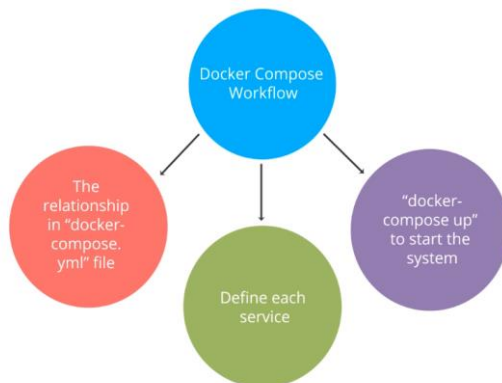
COPY index.html /var/www/html
ADD *.html /var/www/html

EXPOSE 80

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


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.



7.7 Using Docker Compose to Manage a Container

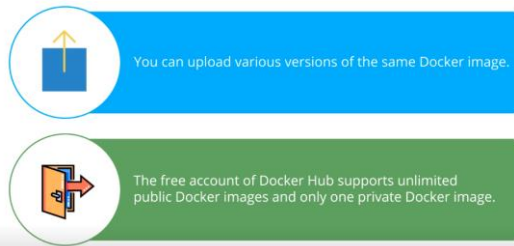
Praxisbeispiel

- Login to your Ubuntu Lab, and open the terminal.
- Create a Dockerfile and build the Docker image with it.
- Run the image and connect to the SSH server.
- Exit the container once you confirm that you have logged in.
- Create a "docker-compose.yml" file and connect to the server.
- Stop the server and exit the container.

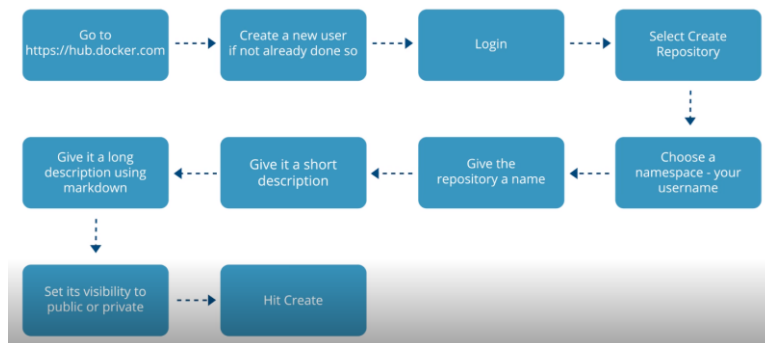
7.8 Docker Registry

Docker Registry

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



Docker Hub Configuration: Step-by-Step Procedure



7.9 Run Docker Registry with Centos

Praxisbeispiel

- Login to your Ubuntu Lab, and open the terminal.
- Pull a recent version of the Centos Linux container.
- Run the registry in a new Docker container and pull a new image from Docker Hub.
- Push the image to the local registry.
- Remove the image from the local cache.
- Pull the image from the local repository.
- Run the new Docker image.
- Exit the container once the image is confirmed.

7.10 Docker Networking

Docker Networking

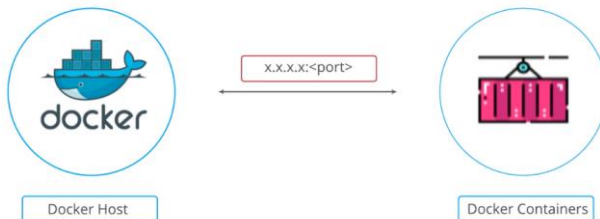


Docker Networking helps Docker container to interact with other microservices over the network. Access to multiple applications hosted inside the Docker container is possible through Docker Networking.

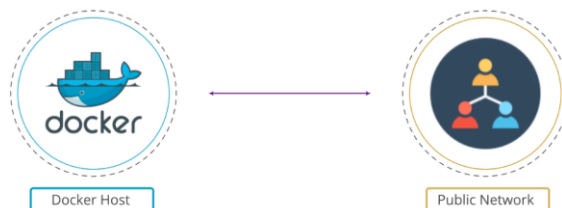
There are four types of Docker networks:

- Bridge Mode
- Host Mode
- Container Mode
- None

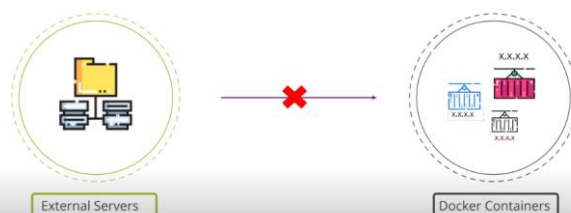
Bridge Mode: It is the default network available for Docker containers. It helps to access Docker container using same Docker host IP address and port.



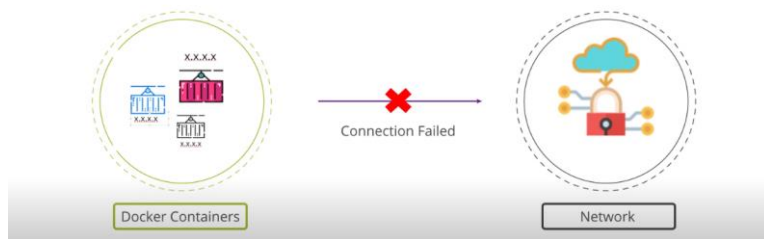
Host Mode: This will inherit all network resources directly from the host and can become exposed directly to public network. This is faster than Bridge mode, but there might be some security issues.



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



None: In this mode, Docker networking gets switched off and containers are not able to connect to the network. This helps to set up the applications network.



7.11 Demonstrate Docker Networking with Two SSHs

Praxisbeispiel

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