+What you’ll learn and do

You’ll learn how to:

* install Docker software for your platform
* run a software image in a container
* browse for an image on Docker Hub
* create your own image and run it in a container
* create a Docker Hub account and an image repository
* create an image of your own
* push your image to Docker Hub for others to use

### What is Docker?

### Docker is an [open-source](https://en.wikipedia.org/wiki/Open-source) project that automates the deployment of [Linux](https://en.wikipedia.org/wiki/Linux) [applications](https://en.wikipedia.org/wiki/Application_software) inside [software containers](https://en.wikipedia.org/wiki/Software_container).

Docker is a technology that wraps Linux's operating-system-level virtualization.

This is an extremely verbose way of saying that Docker is a tool that quickly creates isolated environments for you to develop and deploy applications in.

Another magical feature of Docker is that it allows you to run ANY linux-based operating system within the container - offering even greater flexibility.

If you need to run one application with CentOS and another with Ubuntu - no problem!

### Containers and Images

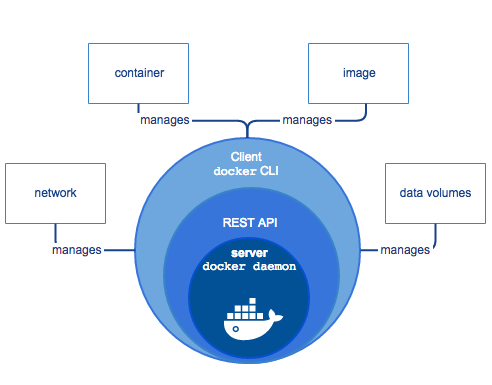
Docker uses the term Containers to represent an actively running environment that can run almost any piece of software; whether it be a web application or a service daemon.

Docker Images are the "building blocks" that act as the starting point for our containers. Images become containers, and containers can be made into images. Images are typically base operating system images (such as Ubuntu), but they can also be highly-customized images containing the base OS plus any additional dependencies you install on it.

What is Docker Engine?

*Docker Engine* is a client-server application with these major components:

* A server which is a type of long-running program called a daemon process.
* A REST API which specifies interfaces that programs can use to talk to the daemon and instruct it what to do.
* A command line interface (CLI) client.



The CLI uses the Docker REST API to control or interact with the Docker daemon through scripting or direct CLI commands. Many other Docker applications use the underlying API and CLI.

**The daemon creates** and manages Docker *objects*, such as images, containers, networks, and data volumes.

Note: Docker is licensed under the open source Apache 2.0 license.

What can I use Docker for?

***Fast, consistent delivery of your applications***

Docker can streamline the development lifecycle by allowing developers to work in standardized environments using local containers which provide your applications and services. You can also integrate Docker into your continuous integration and continuous deployment (CI/CD) workflow.

Consider the following example scenario. Your developers write code locally and share their work with their colleagues using Docker containers. They can use Docker to push their applications into a test environment and execute automated and manual tests. When developers find problems, they can fix them in the development environment and redeploy them to the test environment for testing. When testing is complete, getting the fix to the customer is as simple as pushing the updated image to the production environment.

***Responsive deployment and scaling***

Docker’s container-based platform allows for highly portable workloads. Docker containers can run on a developer’s local host, on physical or virtual machines in a data center, in the Cloud, or in a mixture of environments.

Docker’s portability and lightweight nature also make it easy to dynamically manage workloads, scaling up or tearing down applications and services as business needs dictate, in near real time.

***Running more workloads on the same hardware***

Docker is lightweight and fast. It provides a viable, cost-effective alternative to hypervisor-based virtual machines, allowing you to use more of your compute capacity to achieve your business goals. This is useful in high density environments and for small and medium deployments where you need to do more with fewer resources.

What is Docker’s architecture?

Docker uses a client-server architecture. The Docker *client* talks to the Docker *daemon*, which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon *can* run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client

and daemon communicate using a REST API, over UNIX sockets or a network interface.

The Docker daemon

The Docker daemon runs on a host machine. The user uses the Docker client to interact with the daemon.

The Docker client

The Docker client, in the form of the docker binary, is the primary user interface to Docker. It accepts commands and configuration flags from the user and communicates with a Docker daemon. One client can even communicate with multiple unrelated daemons.

Inside Docker

To understand Docker’s internals, you need to know about *images*, *registries*, and *containers*.

Docker images

A Docker *image* is a read-only template with instructions for creating a Docker container. For example, an image might contain an Ubuntu operating system with Apache web server and your web application installed. You can build or update images from scratch or download and use images created by others. An image may be based on, or may extend, one or more other images. A docker image is described in text file called a *Dockerfile*, which has a simple, well-defined syntax. For more details about images, see [How does a Docker image work?](https://docs.docker.com/engine/understanding-docker/#how-does-a-docker-image-work).

Docker images are the build component of Docker.

Docker containers

A Docker container is a runnable instance of a Docker image. You can run, start, stop, move, or delete a container using Docker API or CLI commands. When you run a container, you can provide configuration metadata such as networking information or environment variables. Each container is an isolated and secure application platform, but can be given access to resources running in a different host or container, as well as persistent storage or databases. For more details about containers, see [How does a container work?](https://docs.docker.com/engine/understanding-docker/#how-does-a-container-work).

Docker containers are the run component of Docker.

Docker registries

A docker registry is a library of images. A registry can be public or private, and can be on the same server as the Docker daemon or Docker client, or on a totally separate server. For more details about registries, see [How does a Docker registry work?](https://docs.docker.com/engine/understanding-docker/#how-does-a-docker-registry-work)

Docker registries are the distribution component of Docker.

Docker management commands

| Command | Description |
| --- | --- |
| [dockerd](https://docs.docker.com/engine/reference/commandline/dockerd/) | Launch the Docker daemon |
| [info](https://docs.docker.com/engine/reference/commandline/info/) | Display system-wide information |
| [inspect](https://docs.docker.com/engine/reference/commandline/inspect/) | Return low-level information on a container or image |
| [version](https://docs.docker.com/engine/reference/commandline/version/) | Show the Docker version information |

Image commands

| Command | Description |
| --- | --- |
| [build](https://docs.docker.com/engine/reference/commandline/build/) | Build an image from a Dockerfile |
| [commit](https://docs.docker.com/engine/reference/commandline/commit/) | Create a new image from a container’s changes |
| [history](https://docs.docker.com/engine/reference/commandline/history/) | Show the history of an image |
| [images](https://docs.docker.com/engine/reference/commandline/images/) | List images |
| [import](https://docs.docker.com/engine/reference/commandline/import/) | Import the contents from a tarball to create a filesystem image |
| [load](https://docs.docker.com/engine/reference/commandline/load/) | Load an image from a tar archive or STDIN |
| [rmi](https://docs.docker.com/engine/reference/commandline/rmi/) | Remove one or more images |
| [save](https://docs.docker.com/engine/reference/commandline/save/) | Save images to a tar archive |
| [tag](https://docs.docker.com/engine/reference/commandline/tag/) | Tag an image into a repository |

Container commands

| Command | Description |
| --- | --- |
| [attach](https://docs.docker.com/engine/reference/commandline/attach/) | Attach to a running container |
| [cp](https://docs.docker.com/engine/reference/commandline/cp/) | Copy files/folders from a container to a HOSTDIR or to STDOUT |
| [create](https://docs.docker.com/engine/reference/commandline/create/) | Create a new container |
| [diff](https://docs.docker.com/engine/reference/commandline/diff/) | Inspect changes on a container’s filesystem |
| [events](https://docs.docker.com/engine/reference/commandline/events/) | Get real time events from the server |
| [exec](https://docs.docker.com/engine/reference/commandline/exec/) | Run a command in a running container |
| [export](https://docs.docker.com/engine/reference/commandline/export/) | Export a container’s filesystem as a tar archive |
| [kill](https://docs.docker.com/engine/reference/commandline/kill/) | Kill a running container |
| [logs](https://docs.docker.com/engine/reference/commandline/logs/) | Fetch the logs of a container |
| [pause](https://docs.docker.com/engine/reference/commandline/pause/) | Pause all processes within a container |
| [port](https://docs.docker.com/engine/reference/commandline/port/) | List port mappings or a specific mapping for the container |
| [ps](https://docs.docker.com/engine/reference/commandline/ps/) | List containers |
| [rename](https://docs.docker.com/engine/reference/commandline/rename/) | Rename a container |
| [restart](https://docs.docker.com/engine/reference/commandline/restart/) | Restart a running container |
| [rm](https://docs.docker.com/engine/reference/commandline/rm/) | Remove one or more containers |
| [run](https://docs.docker.com/engine/reference/commandline/run/) | Run a command in a new container |
| [start](https://docs.docker.com/engine/reference/commandline/start/) | Start one or more stopped containers |
| [stats](https://docs.docker.com/engine/reference/commandline/stats/) | Display a live stream of container(s) resource usage statistics |
| [stop](https://docs.docker.com/engine/reference/commandline/stop/) | Stop a running container |
| [top](https://docs.docker.com/engine/reference/commandline/top/) | Display the running processes of a container |
| [unpause](https://docs.docker.com/engine/reference/commandline/unpause/) | Unpause all processes within a container |
| [update](https://docs.docker.com/engine/reference/commandline/update/) | Update configuration of one or more containers |
| [wait](https://docs.docker.com/engine/reference/commandline/wait/) | Block until a container stops, then print its exit code |

Hub and registry commands

| Command | Description |
| --- | --- |
| [login](https://docs.docker.com/engine/reference/commandline/login/) | Register or log in to a Docker registry |
| [logout](https://docs.docker.com/engine/reference/commandline/logout/) | Log out from a Docker registry |
| [pull](https://docs.docker.com/engine/reference/commandline/pull/) | Pull an image or a repository from a Docker registry |
| [push](https://docs.docker.com/engine/reference/commandline/push/) | Push an image or a repository to a Docker registry |
| [search](https://docs.docker.com/engine/reference/commandline/search/) | Search the Docker Hub for images |

**Practical:**

1 .Docker search imagesname

2. docker pull imagename

3. docker images(docker images –a)

4. docker run –it imagename /bin/bash

5. docker ps(crtl+p+q)

6. docker ps –a(exit)

7.docker \* login

**Modifie\*d image name:**

8.docker tag imageid newimage

or

9. docker tag imageid registry:5000 or repository/imagename.

**Check:**

10.docker images

**Remove images:**

11.docker rmi imagename or image id

**Delete all images:**

12.docker rmi $(docker images –q)

**Remove containers:**

13.docker rm –f containerid

**Remove all containers:**

14.docker rm $(docker ps –a –q)

**Build** :

15.docker build –t anyname .

**Check:**

16.docker images

**Build and push and pull and run:**

17.before push we want create a repository in dockerhub

18.docker push repositorynameor registry:5000/imagename

Ex: docker images

…..docker tag imageid username(dockerhub login username)/imagename

…..docker images

……docker push username(dockerhub login username)/imagename

**Check:**

20.login dockerhub account and check it created repository

21.docker pull repository or registry:5000/imagename

22.docker images

23.docker run –it imagename /bin/bash

**Check information about images and containers:**

24.docker info

**Run background:(dittach mode)**

**25.**docker run –it -d imagename /bin/bash

**Check:**

26.docker ps

**Attach mode:**

27.docker ps

28.docker attach (docker ps containerid)

Or

29.docker attach containerid(background)

**Start:**

**(**foreground to background)

**first we can run background mode and ofter we change to attach mode above two steps**

**now we start foreground to backgroung asuse.**

30.docker ps –a

31.docker start containerid(docker ps –a id)

**Ckeck:**

32.docker ps

**Stop:**

**Stop means stop the running container**

**We want stop container (background running) before removing that container.**

33.docker stop containerid(docker ps containerid)

**History:**

It will shows the **history of image**

34.docker history imagename

**Inspect:**

Return low-level information of container

35. docker inspect imagename

**Logs:**

Logs of container

36.docker logs containerid and docker logs –t containerid

**Kill:**

Kill the running container

37.docker ps

38.docker kill containerid

**Check:**

**39.docker ps**

**Diff:**

**List the changed files and directories in a container᾿s filesystem There are 3 events that are listed in the diff:**

**40.docker diff containerid**

1. A - Add
2. D - Delete
3. C – Change

**41.docker run –it imagename /bin/bash**

**/#.mkdir sekhar**

**/#exit**

**43.docker ps –a**

**44.docker diff cintainerid (docker ps –a containerid)**

1. For example:

$ docker diff 7bb0e258aefe

C /dev

A /dev/kmsg

C /etc

A /etc/mtab

A /**go**

A /**go**/src

A /**go**/src/github.com

A /**go**/src/github.com/docker

A /**go**/src/github.com/docker/docker

A /**go**/src/github.com/docker/docker/.git

....