**Docker**

[**1. Introduction to docker**](http://www.vogella.com/tutorials/Docker/article.html#introduction-to-docker)**:-**

Docker is a light weight container, allowing to run pre-configured system images on another operating system. Compared to other virtualization technologies, Docker allows to run many containers per VM and uses less memory/CPU resources. Docker does this, as does not require an hypervisor for each container.

[**2. Use cases**](http://www.vogella.com/tutorials/Docker/article.html#use-cases)**:-**

Docker allows to solve the following use cases

* You can to use a specific version of a operating system for testing
* You want to run your software tests in an controlled environment
* You want to configure a system and share the configuration

[**3**. **Docker components**](http://www.vogella.com/tutorials/Docker/article.html#docker-components)**:-**

Docker consists of the following components:

* Images
* Containers
* Daemon
* Clients
* Registries

[**3.1. Images**](http://www.vogella.com/tutorials/Docker/article.html#images)**:-**

Images are read-only templates which provide functionality for running an instance of this image (container). An example for a image is the latest release of Ubuntu. Images are defined as layers, for example, you can add Java to the Ubuntu image and get another image based on this.

The Docker hub provides lot of pre-configured images. You can modify existing images and save these modifications as new image.

**3.2** [**Containers**](http://www.vogella.com/tutorials/Docker/article.html#containers)**:-**

Container are the started components based on images. They contain the actual application and dependencies but share the same kernel. They can be started, stopped, paused, deleted. Containers are immutable and disposable.

### 3.3 [Docker Daemon](http://www.vogella.com/tutorials/Docker/article.html#docker-daemon):-

Is used to manage the container. It runs natively on Linux and inside a VM on Windows and Mac OS X. To start it use the docker command.

### 3.4 [Docker Clients](http://www.vogella.com/tutorials/Docker/article.html#docker-clients):-

Clients (CLI, IDE) run on host VM. They provide the tools to interact with container, i.e., to start them.

### [3.5 Registries](http://www.vogella.com/tutorials/Docker/article.html#registries):-

Images are saved in a registry and have an ID with consists of a repository and a tag. For example, *fedora:22*, is an image which contains the Fedora 22 OS from the fedora repository.

To use an image you have to pull it from a registry, to share an image with others you have to push it to one. The default Docker registry is the Docker Hub. You can upload your personal images to Github, in this case you add your user name as prefix to the image, e.g.,*vogella/fedore:22*

### [3.6 Dockerfile](http://www.vogella.com/tutorials/Docker/article.html#dockerfile):-

A dockerfile allows to modify existing images. It contains the image that should be used and the adjustments to this image, e.g., which packages should be installed, system configuration etc.

# Installing Docker on Ubuntu 16.04 LTS

1 Check the Kernel Version

ubuntu@ubuntu:~$ uname -r

Expected Response

4.4.0-42-generic

2 Update package information, ensure that APT works with the https method, and that CA certificates are installed.

$ sudo apt-get update

3 Add the keys from the keyserver

$ sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 \

--recv-keys 58118E89F3A912897C070ADBF76221572C52609D

Expected Output

Executing: /tmp/tmp.VEHRG1bkmN/gpg.1.sh --keyserver

hkp://p80.pool.sks-keyservers.net:80

--recv-keys

58118E89F3A912897C070ADBF76221572C52609D

gpg: requesting key 2C52609D from hkp server p80.pool.sks-keyservers.net

gpg: key 2C52609D: public key "Docker Release Tool (releasedocker)

<docker@docker.com>" imported

gpg: Total number processed: 1

gpg: imported: 1 (RSA: 1)

1. Add end point for *ubuntu-xenial* to /etc/apt/sources.list.d

$ sudo vi /etc/apt/sources.list.d/docker.list

For Ubuntu xenial , add following line to the file

deb https://apt.dockerproject.org/repo ubuntu-xenial main

1. For aufs use linux-image-extra

$ sudo apt-get install linux-image-extra-$(uname -r)

1. Install docker binary

$ sudo apt-get install docker-engine

$ sudo service docker start

Check the docker version

$ docker --version

Docker version 1.12.2, build bb80604

1. Optional : Add current user to docker group to avoid using sudo

$ sudo usermod -aG docker ubuntu

Give Permissions to current user to execute docker binary

$ sudo chmod 755 /usr/bin/docker

# Getting Started

## Basic Commands

### Info command

This command gives information about the docker setup on your machine/vm

$ docker info

Output of the command will look similar to the listing below

Containers: 0

Images: 301

Server Version: 1.9.0

Storage Driver: aufs

Root Dir: /var/lib/docker/aufs

Backing Filesystem: extfs

Dirs: 301

Dirperm1 Supported: false

Execution Driver: native-0.2

Logging Driver: json-file

Kernel Version: 3.13.0-32-generic

Operating System: Ubuntu 14.04.1 LTS

CPUs: 4

Total Memory: 3.141 GiB

Name: ubuntu

ID: CS3B:XJ7B:DEWA:NRXG:VGDY:HYWO:N4NA:FGXM:WYME:FW5C:WYJN:IHVJ

Username: rajdeepd

Registry: https://index.docker.io/v1/

WARNING: No swap limit support

### Create a Container and enter its shell

$ sudo docker run -i -t debian /bin/bash

This should give you a new command prompt inside the container, very similar to if you had ssh‘ed into a remote machine. In this case the flags -i and -t tell Docker we want an interactive session with a tty attached. The command /bin/bash gives a bash shell. When you exit the shell the container will stop — containers only run as long as their main process

$ docker run debian echo hello-world

hello-world

### Create a container with a name

You can use -h command line parameter to specify a container name.

$ docker run -h CONTAINER1 -i -t debian /bin/bash

Output of the command above will open a tty inside the container

root@CONTAINER1:/#

### Create a container with a Networking mode

Container mode can be specified using the flag :code:[`](http://containertutorials.com/get_started/index.html#id1)-net=<NETWORK\_MODE> where

$ docker run -h CONTAINER2 -i -t --net="bridge" debian /bin/bash

### List of docker containers running

$ docker ps -a

### Inspect a Container

$ docker inspect hopeful\_pare

Output will be a JSON file.

### Start a Stopped Container

$ docker start hopeful\_pare

where hopeful\_pare is the container name.

**Enter the Shell of a Started Container**

$ docker attach hopeful\_pare

where hopeful\_pare is the container name.

### Detach from a Container

docker run -t -i → can be detached with ^P^Q and reattached with docker attach

docker run -i → cannot be detached with ^P^Q; will disrupt stdin

docker run → cannot be detached with ^P^Q;

can SIGKILL client; can reattach with docker attach

### Docker Logs

If you run this command with the name of your container, you should get a list of commands executed in the container.

$ docker logs hopeful\_pare

where hopeful\_pare is the container name.

### Removing a Single Container

$ docker rm hopeful\_pare

### Removing all the Containers

$ docker rm `docker ps --no-trunc -aq`

# Getting Started with Alpine[¶](http://containertutorials.com/alpine/get_started.html#getting-started-with-alpine)

Alpine is a lightweight linux distribution based on musl libc and busybox. There is a docker image based on Alpine which is an easy way of getting started with Alpine

## Alpine Docker Image

Based on Alpine kernel, this is a lightweight image of 5mb

Pull the alpine image,

$ docker pull alpine

Check IP Address of the container

$ docker run alpine ifconfig

Launching a bash shell

$ docker run -i -t alpine /bin/bash

This will give an error, as bash is not supported in alpine

exec: "/bin/bash": stat /bin/bash: no such file or directory

docker: Error response from daemon: Container command not found or does not exist..

Getting inside the container

$ docker run -it alpine /bin/sh

/ #

Detaching from the container without stopping Ctrl-P Ctrl-Q

Check the docker container is still running

$ docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

8647ce2b84a5 alpine "/bin/sh" About a minute ago Up About a min

# Static site on Apache server from Docker

In this lab we learn how to host a static site running on Apache server hosted by Docker

1. Create a Dockerfile
2. FROM smebberson/alpine-apache
3. ADD ./public-html/myindex.html /var/www/localhost/htdocs
4. Create a directory public\_html with the following content in myindex.html
5. <html>
6. <body>
7. Hi There - Static page served by Apache Server
8. </body>
9. </html>
10. Your directory should look like this
11. $ tree .
12. .
13. ├── Dockerfile
14. └── public-html
15. └── myindex.html
16. Create a Docker image
17. $ docker build -t my-apache2-alpine .

This will create a my-apache2 image.

1. Create a Docker Container running this image
2. docker run -p 80:80 --name my-apache2-alpine-1 my-apache2-alpine
3. Open browser of the host at http://localhost:80, you will see the website up and running
4. Open the browser at the following url to see your custom page http://localhost:80/myindex.html

# Docker Images

**Show images**

$ sudo docker images

**Specifying a Variant**

$ sudo docker run -t -i ubuntu:14.04 /bin/bash

**Pull an Image**

$ sudo docker pull debian

**Create your own image**

1. Create a Dockerfile with the following content
2. FROM debian:wheezy
3. RUN apt-get update && apt-get install -y cowsay fortune
4. Go to the directory container Dockerfile and execute the following command to build a image
5. $ docker build -t test/cowsay-dockerfile .

You will see output as shown below

Sending build context to Docker daemon 2.048 kB

Sending build context to Docker daemon

Step 0 : FROM debian:wheezy

wheezy: Pulling from debian

7a3e804ed6c0: Pull complete

b96d1548a24e: Already exists

Status: Downloaded newer image for debian:wheezy

---> b96d1548a24e

Step 1 : RUN apt-get update && apt-get install -y cowsay fortune

---> Running in 4404353a3643

Get:1 http://security.debian.org wheezy/updates Release.gpg [1554 B]

Get:2 http://security.debian.org wheezy/updates Release [102 kB]

Get:3 http://httpredir.debian.org wheezy Release.gpg [2390 B]

.....

Setting up perl (5.14.2-21+deb7u2) ...

update-alternatives: using /usr/bin/prename to provide /usr/bin/rename

---> ca3618d10f2a

Removing intermediate container 4404353a3643

Successfully built ca3618d10f2a

1. Check that image has been created
2. $ docker images
3. REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE
4. test/cowsay-dockerfile latest ca3618d10f2a 3 minutes ago 126.9 MB
5. docker-dev dry-run-test-2 db155754d7fc 6 days ago 1.571 GB
6. <none> <none> b01392d005bb 6 days ago 1.571 GB
7. debian wheezy b96d1548a24e 7 days ago 84.97 MB
8. debian latest df2a0347c9d0 7 days ago 125.2 MB
9. dockerswarm/dind-master latest bb4cd757411e 7 days ago 159 MB
10. <none> <none> f672d2db20f6 7 days ago 1.571 GB
11. <none> <none> 1fe07c1fdf52 8 days ago 1.571 GB
12. dockerswarm/swarm-test-env latest 01e6a0da0825 2 weeks ago 515.5 MB
13. ubuntu 14.04 07f8e8c5e660 3 weeks ago 188.3 MB
14. hello-world latest 91c95931e552 5 weeks ago 910 B
15. busybox latest 8c2e06607696 5 weeks ago 2.433 MB
16. Run the cowsay program using the built image
17. $ docker run test/cowsay-dockerfile /usr/games/cowsay "Hi!"

This will execute and show the output

\_\_\_\_\_

< Hi! >

-----

\ ^\_\_^

\ (oo)\\_\_\_\_\_\_\_

(\_\_)\ )\/\

||----w |

|| ||

1. Removing a Docker Image : Docker image can be removed using the following command

$ docker rmi test/cowsay-dockerfile