

# Setting up Everything

For learning docker you can use the docker playground or you can just install the docker on your host machine. docker support windows mac and of course Linux. but in this tutorial we will use the docker playground

you can find the docker in this url

install docker in linux

```
wget -q0- https://test.docker.com/ | sh
```

this command will download a script and run the script with the bash

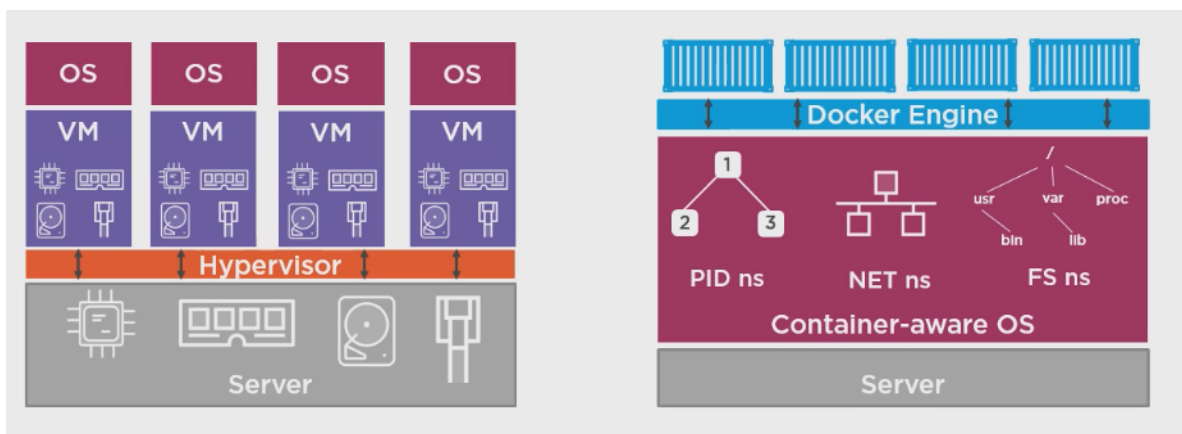
to find the docker version and check if the docker is installed

the command is

```
docker version
```

if everything goes right .this command will show the docker version  
in your computer

## what is Container?



## virtual machine:

when you use the virtualisation manager like the hyper v, or other virtualisation software. it takes the hardware resources and then make the virtual version and then make several virtual version that we called virtual machine and on top of it we install the Operating system then on top of it the application run or whatever we want to do it. it takes too much resources.

## Container:

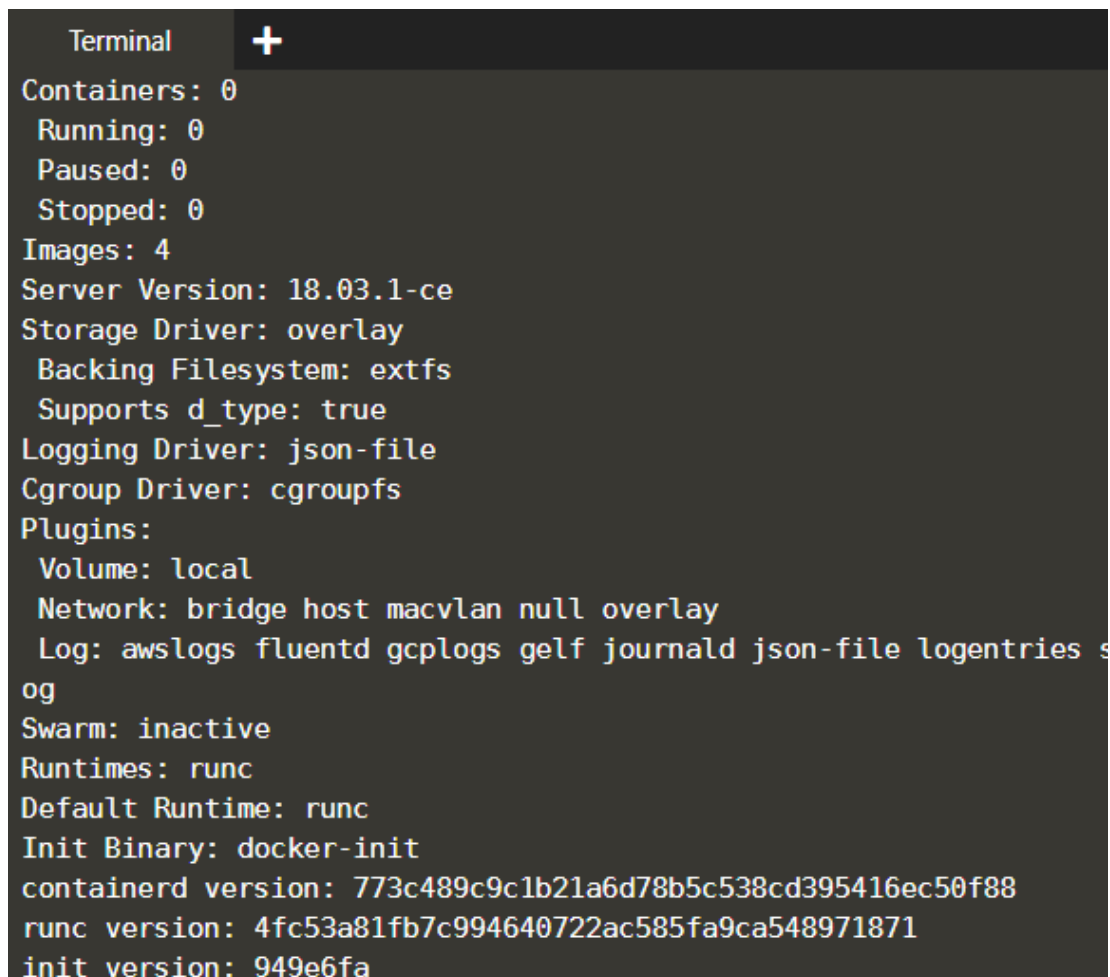
on the other hand the container like docker slices the operating system resources. every container got their root process and the own file system . they are more like operating system and it efficient to use it for it does not take a whole operating system to work.

## work with docker:

- first check if any image is installed although in a fresh install this will show you 0

```
docker info
```

the output is

A terminal window with a dark background and light-colored text. The title bar at the top says 'Terminal' with a plus sign icon. The output of the 'docker info' command is displayed, showing various system and Docker configuration details. The text is as follows:

```
Containers: 0
Running: 0
Paused: 0
Stopped: 0
Images: 4
Server Version: 18.03.1-ce
Storage Driver: overlay
Backing Filesystem: extfs
Supports d_type: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
Volume: local
Network: bridge host macvlan null overlay
Log: awslogs fluentd gcplogs gelf journald json-file logentries s
og
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 773c489c9c1b21a6d78b5c538cd395416ec50f88
runc version: 4fc53a81fb7c994640722ac585fa9ca548971871
init version: 949e6fa
```

- lets install a basic container name **\*\* hello world \*\***

```
docker run hello-world
```

this command will check the local drive if the container exists if there is not then it will fetch a copy from the docker hub and and launch the container

it is a very basic container it just print some messages the output is shown bellow

```
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:6a65f928fb91fcfb963f7aa6d57c8eeb426ad9a20c7ee045538ef34847f44f1
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent
it
to your terminal.
```

- To see the docker status we run this command

```
docker ps -a
```

```
$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED
STATUS        PORTS          NAMES
b9b071c41f0d   hello-world   "/hello"                 4 minutes ago
Exited (0) 4 minutes ago
elegant_jones
$
```

- now if you run the command

```
docker info
```

it will show there is one container in your system

- to the see images name that is downloaded use the command

```
docker images
```

# Theory behind Containers and Images

when we install the docker engine it install the docker daemon and the docker client both

two things are known as the docker engine .

when we run the command `docker run hellow-world`

this 3 thing happened behind the scene

- client make a API call to the daemon
- Daemon implements the Docker remote Api to find the container of it is not here locally
- `docker run` runs the new Container

## Working with Images

to install Alpine linux container

```
docker install Alpine
```

to install ubuntu latest

```
docker install ubuntu:latest
```

to install ubuntu image of a specific version

```
docker install ubuntu:14.04
```

to remove a docker images

```
docker rmi ubuntu:14.04
```

see the list of images

```
docker images
```

## Run a web server in Docker (NGNIX)

```
docker pull nginx  
docker run -d --name web_server -p 80:8080 nginx
```

### Command explanation

**-d** tells you run the command in the detach mode (in background)

**--name** wil give you a name of the container

**-p 80:8080** This will map the port 80 of the container to the port 8080 of the Host machine

```
$ docker run -d --name web_server -p 80:8080 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
afb6ec6fdc1c: Pull complete
b90c53a0b692: Pull complete
11fa52a0fdc0: Pull complete
Digest: sha256:6fff55753e3b34e36e24e37039ee9eae1fe38a6420d8ae16ef37c92d1eb26699
Status: Downloaded newer image for nginx:latest
dd42d14a8a020275b4f7eefb111f08187576097f17144d280b6f6b971dde339d
$
```

lets see the status

`docker ps`

```
$ docker ps
CONTAINER ID   IMAGE    COMMAND                  CREATED        STATUS
dd42d14a8a02   nginx    "nginx -g 'daemon of..." About a minute ago Up 59
seconds       80/tcp, 0.0.0.0:80->8080/tcp web_server
```

lets see if the server is running and we get response from the server

```
curl localhost:8080
```

now if you want to interact with docker container

suppose you want to run bash shell to do that

you can use this command

```
$ docker run -it --name temp1 nginx /bin/bash
root@5e09150fdb01:/#
root@5e09150fdb01:/# cd /usr/share/nginx/
root@5e09150fdb01:/usr/share/nginx# ls
html
root@5e09150fdb01:/usr/share/nginx# cd html/
root@5e09150fdb01:/usr/share/nginx/html# ls
50x.html  index.html
root@5e09150fdb01:/usr/share/nginx/html#
```

to stop any container you use the `docker stop` command

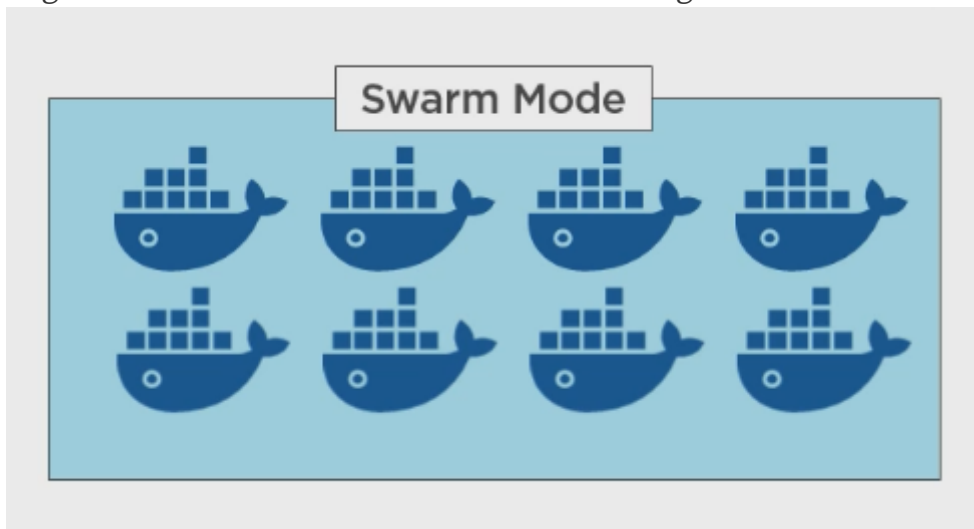
```
docker stop <container id>
```

## Native Clustering in Docker

you can run a cluster of docker for running an app

cluster is also known as swarm

engines in a swarm run in a swarm mode manager node maintain the Swarm



and just remember container also known as task

## Building a Swarm:

we will make three manager nodes and three worker nodes Manager nodes look after the state of the cluster and gives tasks to the worker nodes. you can use multiple manager nodes for redundancy . and that is also recommended that you should have at least three manager nodes in a swarm and only one leader node. manager can be a worker nodes too

Manager nodes maintain the swarm worker nodes executes the task that are assigned to the workers.

you can run your application in the

1) first create a 6 instance three manager and three worker

2) go to manager1 type this command

```
docker swarm init
```

and to add another instance to as manager

run this command

```
docker swarm-join manager
```

it will give you a docker command with a key.

like this

```
docker swarm join --token SWMTKN-1-  
0wwpy6i65t1aw7wl12m4tir3xa7vy95kallufx8vhzszflzas1-  
eobdc3suqm29rs6fdi64oqhXu 172.17.0.86:2377
```

copy the command

3) ssh to manager 2 and execute the command that you copy

4) go to the manager1 and execute the

```
docker swaem-join manager
```

and generate another command

5) ssh to the manager3 and execute this command

6) now for adding the worker use this command

```
docker swarm-join worker
```

7) now do the same process and add the other two worker

[create the command ssh to worker instance and add it]

8) to see the node use this command

```
docker node ls
```

9) remember manager1 will be the leader

10) sets deploy a webserver nginx. you need to deploy it as a service

```
docker service create -p 80:80 --name webserver nginx
```

11) one thing to solve confusion is manager node is also worker node

11) now activate the rest 5

```
docker service scale webserver=5
```

12) now you can show the output with

```
docker service ps
```

**now you can hit any node in a swarm and get your service**