docker info

To view detailed information about your Docker installation

```
[node1] (local) root@192.168.0.23 ~
$ docker info
Client:
 Version:    27.3.1
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Docker Buildx (Docker Inc.)
    Version:  v0.17.1
    Path:     /usr/local/libexec/docker/cli-plugins/docker-buildx
  compose: Docker Compose (Docker Inc.)
    Version:  v2.29.7
    Path:     /usr/local/libexec/docker/cli-plugins/docker-compose
  scout: Docker Scout (Docker Inc.)
    Version:  v1.0.9
    Path:     /usr/lib/docker/cli-plugins/docker-scout

Server:
```


creating hello world image

```
[node1] (local) root@192.168.0.23 ~
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:7e1a4e2d11e2ac7a8c3f768d4166c2defeb09d2a750b010412b6ea13de1efb19
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 try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

docker container ls

lists all **running** Docker containers on your system.

```
$ docker container ls
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[node1] (local) root@192.168.0.23 ~
```

docker container ls –a

lists **all Docker containers** on your system — both **running and stopped**.

```
$ docker container ls -a
CONTAINER ID   IMAGE         COMMAND    CREATED         STATUS                     PORTS     NAMES
52026bf0168c   hello-world   "/hello"   6 minutes ago   Exited (0) 6 minutes ago             laughing_buck
[node1] (local) root@192.168.0.23 ~
$
```

docker container start laughing_buck

This command starts an **already created and exited container** named laughing_buck.

Docker confirms it by echoing back the container name.

docker images

Shows the available Docker images on your system.

```
[node1] (local) root@192.168.0.23 ~
$ docker container start laughing_buck
laughing_buck
[node1] (local) root@192.168.0.23 ~
$ docker container ls -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                      PORTS     NAMES
ebe668f5590d   hello-world   "/hello"   6 minutes ago    Exited (0) 6 minutes ago              unruffled_shtern
52026bf0168c   hello-world   "/hello"   14 minutes ago   Exited (0) 4 seconds ago              laughing_buck
[node1] (local) root@192.168.0.23 ~
$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    74cc54e27dc4   2 months ago   10.1kB
[node1] (local) root@192.168.0.23 ~
$
```


docker run -d -p 9090:80 nginx

Explanation:

- docker run: Start a new Docker container.
- -d: Run it in **detached mode** (in the background).
- -p 9090:80: **Map port 9090** on your **host machine** to port **80 inside the container** (where nginx serves web content).
- nginx: The **image name**. You're asking Docker to run a container using the nginx image.

```
[node1] (local) root@192.168.0.23 ~
$ docker run -d -p 9090:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
6e909acdb790: Pull complete
5eaa34f5b9c2: Pull complete
417c4bccf534: Pull complete
e7e0ca015e55: Pull complete
373fe654e984: Pull complete
97f5c0f51d43: Pull complete
c22eb46e871a: Pull complete
Digest: sha256:124b44bfc9ccd1f3cedf4b592d4d1e8bddb78b51ec2ed5056c52d3692baebc19
Status: Downloaded newer image for nginx:latest
90d5925a4a283bf2dbb9ecd0db460837b352fa5f8d48a2aab494ae407fbe02d7
[node1] (local) root@192.168.0.23 ~
$
```

What Docker did:

- It **couldn't find the nginx image locally**, so it pulled the latest version from Docker Hub (library/nginx).

- You can see multiple image layers being downloaded (Pull complete).
- Finally, it **downloaded and started the nginx container**.

```
[node1] (local) root@192.168.0.23 ~
$ docker run -d -p 9090:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
6e909acdb790: Pull complete
5eaa34f5b9c2: Pull complete
417c4bccf534: Pull complete
e7e0ca015e55: Pull complete
373fe654e984: Pull complete
97f5c0f51d43: Pull complete
c22eb46e871a: Pull complete
Digest: sha256:124b44bfc9ccd1f3cedf4b592d4d1e8bddb78b51ec2ed5056c52d3692baebc19
Status: Downloaded newer image for nginx:latest
90d5925a4a283bf2dbb9ecd0db460837b352fa5f8d48a2aab494ae407fbe02d7
[node1] (local) root@192.168.0.23 ~
$
```

docker ps

```
[node1] (local) root@192.168.0.23 ~
$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                    NAMES
90d5925a4a28   nginx     "/docker-entrypoint.…"   52 seconds ago   Up 51 seconds   0.0.0.0:9090->80/tcp     strange_yalow
[node1] (local) root@192.168.0.23 ~
```


Now click on open port and type 9090

Nginx will start

Pulling the Ubuntu image

docker pull Ubuntu

Pulling the latest **Ubuntu** image using Docker

```
$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
5a7813e071bf: Pull complete
Digest: sha256:72297848456d5d37d1262630108ab308d3e9ec7ed1c3286a32fe09856619a782
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
[node1] (local) root@192.168.0.23 ~
$
```

lets check the image created or not using

docker images

```
$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
nginx         latest    53a18edff809   8 weeks ago     192MB
ubuntu        latest    a04dc4851cbc   2 months ago    78.1MB
hello-world   latest    74cc54e27dc4   2 months ago    10.1kB
[node1] (local) root@192.168.0.23 ~
```

Now we are running an Ubuntu container in **detached mode**

```
[node1] (local) root@192.168.0.23 ~
$ docker run -d --name my-linux ubuntu
ea4d7a602514448d9d5da78fd99289194157a0e3c88c9ee8ec519ac43fa87842
[node1] (local) root@192.168.0.23 ~
$
```

-d: Detached mode (runs the container in the background).

--name my-linux: You named the container **my-linux**.

ubuntu: You're using the Ubuntu image.

Now:

docker rm my-linux

```
[node1] (local) root@192.168.0.23 ~
$ docker rm my-linux
my-linux
```

- **`docker rm`**: Removes (deletes) a container.
- **`my-linux`**: That's the name of the container you created earlier.

Since your container was already stopped (because it exited automatically), Docker let you remove it right away.

docker run -itd --name my-linux ubuntu

It runs an Ubuntu container in the background with an interactive terminal and names it my-linux.

```
[node1] (local) root@192.168.0.23 ~
$ docker run -itd --name my-linux ubuntu
b55d379f04bb1b783a45886913daf708de4020371429f11c7e816dc5560463f3
[node1] (local) root@192.168.0.23 ~
$
```

docker ps
listed running containers
docker exec -it my-linux /bin/bash
accessed the running container interactively (bash shell)

```
[node1] (local) root@192.168.0.23 ~
$ docker run -itd --name my-linux ubuntu
b55d379f04bb1b783a45886913daf708de4020371429f11c7e816dc5560463f3
[node1] (local) root@192.168.0.23 ~
$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                    NAMES
b55d379f04bb   ubuntu    "/bin/bash"              58 seconds ago   Up 58 seconds                            my-linux
90d5925a4a28   nginx     "/docker-entrypoint.…"   14 minutes ago   Up 14 minutes   0.0.0.0:9090->80/tcp     strange_yalow
[node1] (local) root@192.168.0.23 ~
$ docker exec -it my-linux /bin/bash
root@b55d379f04bb:/#
```


ps –a

ps -a lists running processes inside the container.

```
root@b55d379f04bb:/# ps -a
  PID TTY          TIME CMD
   20 pts/1    00:00:00 ps
root@b55d379f04bb:/#
```

docker ps command shows the list of running containers

```
[node1] (local) root@192.168.0.23 ~
$ docker ps
CONTAINER ID   NAME           CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O       PIDS
b55d379f04bb   my-linux       0.00%     4.449MiB / 31.42GiB   0.01%     0B / 0B           0B / 0B         1
90d5925a4a28   strange_yalow  0.00%     10.2MiB / 31.42GiB    0.03%     12.4kB / 5.36kB   0B / 12.3kB     9
```

docker network ls command on your Docker host

docker network ls

```
[node1] (local) root@192.168.0.23 ~
$ docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
15f0209fd885   bridge            bridge    local
8256b700b19f   docker_gwbridge   bridge    local
6cd70168e135   host              host      local
9fce0d65c2d8   none              null      local
```

docker volume ls command lists all Docker volumes available on your system.

```
[node1] (local) root@192.168.0.23 ~
$ docker volume ls
DRIVER    VOLUME NAME
[node1] (local) root@192.168.0.23 ~
$
```

`docker volume create mydata` creates a new Docker volume named `mydata`

```
[node1] (local) root@192.168.0.23 ~
$ docker volume create mydata
mydata
[node1] (local) root@192.168.0.23 ~
$ docker volume ls
DRIVER    VOLUME NAME
local     mydata
[node1] (local) root@192.168.0.23 ~
$
```

docker run -it busybox
Starts an interactive container using the busybox image.
echo "some data" > /data/file.txt
Initially failed because the /data directory didn't exist.
mkdir data
You created a new directory named data.
echo "some data" > /data/file.txt
This time, it succeeded since /data now exists.
ls and cd data/ && ls
Confirmed that file.txt was created inside /data.

```
[node1] (local) root@192.168.0.23 ~
$ docker run -it busybox
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
97e70d161e81: Pull complete
Digest: sha256:37f7b378a29ceb4c551b1b5582e27747b855bbfaa73fa11914fe0df028dc581f
Status: Downloaded newer image for busybox:latest
/ # echo "some data" > /data/file.txt
sh: can't create /data/file.txt: nonexistent directory
/ # mkdir data
/ # echo "some data" > /data/file.txt
/ # ls
bin    data   dev    etc    home   lib    lib64  proc   root   sys    tmp    usr    var
/ # cd data/
/data # ls
file.txt
```

You created a `/data` directory inside a **`busybox`** container.
Wrote "some data" into `/data/file.txt`.
Used `cat file.txt` to display the contents of the file.

```
/data # cat file.txt
some data
/data #
```

docker run -it -v mydata:/data busybox

This mounts a **named volume** mydata to /data inside the container.

echo "entering data again" > data/file1.txt

cd data/

ls –lrt

file1.txt was created with contents: entering data again

ls -lrt confirms the file was created with proper permissions and timestamp

```
[node1] (local) root@192.168.0.23 ~
$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                    NAMES
b55d379f04bb   ubuntu    "/bin/bash"              16 minutes ago   Up 16 minutes                            my-linux
90d5925a4a28   nginx     "/docker-entrypoint.…"   29 minutes ago   Up 29 minutes   0.0.0.0:9090->80/tcp     strange_yalow
[node1] (local) root@192.168.0.23 ~
$ docker run -it -v mydata:/data busybox
/ # ls
bin    data   dev    etc    home   lib    lib64  proc   root   sys    tmp    usr    var
/ # echo "entering data again" > data/file1.txt
/ # cd data
/data # ls
file1.txt
/data # ls -lrt
total 4
-rw-r--r--    1 root     root            20 Apr  6 06:56 file1.txt
/data #
```


now you're exploring the results of running different containers using

docker volume ls:

Lists existing Docker volumes.

```
[node1] (local) root@192.168.0.23 ~
$ docker volume ls
DRIVER    VOLUME NAME
local     mydata
```

docker run -it -v mydata:/data busybox

docker run: Runs a new container.

-it: Interactive terminal mode.

-v mydata:/data: Mounts the Docker volume named mydata to the /data directory inside the container.

busybox: Lightweight Linux container image used here.

cat data/file1.txt

Displays the content of /data/file1.txt

```
[node1] (local) root@192.168.0.23 ~
$ docker run -it -v mydata:/data busybox
/ # cat data/file1.txt
entering data again
/ #
```

The file file1.txt exists inside the mydata volume and has the content "entering data again". This confirms that:

- The Docker volume mydata is being persistently used across containers.
- Data written to mydata is accessible when mounted again.