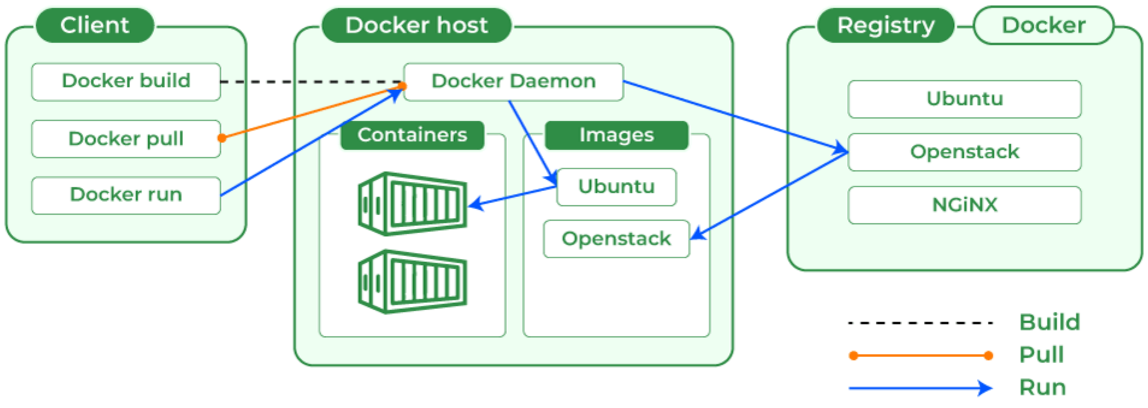
Docker architecture

Docker makes use of client-server architecture. The Docker client talks with the docker daemon which helps in building, running, and distributing the docker containers. The Docker client runs with the daemon on the same system or we can connect the Docker client with the Docker daemon remotely. With the help of REST API over a UNIX socket or a network, the docker client and daemon interact with each other.



Dockefie basic

A Dockerfile is a text document in which you can lay down all the instructions that you want for an image to be created. The first entry in the file specifies the base image, which is a pre-made image containing all the dependencies you need for your application. Then, there are commands you can send to the Dockerfile to install additional software, copy files, or run scripts.

The result is a Docker image: a self-sufficient, executable file with all the information needed to run an application.

FROM - Represents the base image (OS), which is the command that is executed first before any other commands.
Syntax:
***FROM* <Image Name>**

COPY - The copy command is used to copy the file/folders to the image while building the image.
Syntax:
COPY <Source> <Destination>

ADD - While creating the image, we can download files from distant HTTP/HTTPS destinations using the ADD command.
Syntax: ***ADD <URL>***

RUN - Scripts and commands are run with the RUN instruction. The execution of RUN commands or instructions will take place while you create an image on top of the prior layers (Image).
Syntax
RUN < Command + ARGS>

CMD - The main purpose of the CMD command is to start the process inside the container and it can be overridden.
Syntax
CMD [command + args]

ENTRYPOINT - A container that will function as an executable is configured by ENTRYPOINT. When you start the Docker container, a command or script called ENTRYPOINT is executed.
It can’t be overridden. The only difference between cmd and entery point CMD can be overridden and ENTRYPOINT can’t.
Syntax
ENTRYPOINT [command + args]

MAINTAINER - By using the MAINTAINER command we can identify the author/owner of the Docker file and we can set our own author/owner for the image.
Syntax:
MAINTAINER <NAME>

Docker build image

The docker build command is used to build a Docker image from a Dockerfile and a build context. It's a fundamental command for creating container images that can be used to run applications.

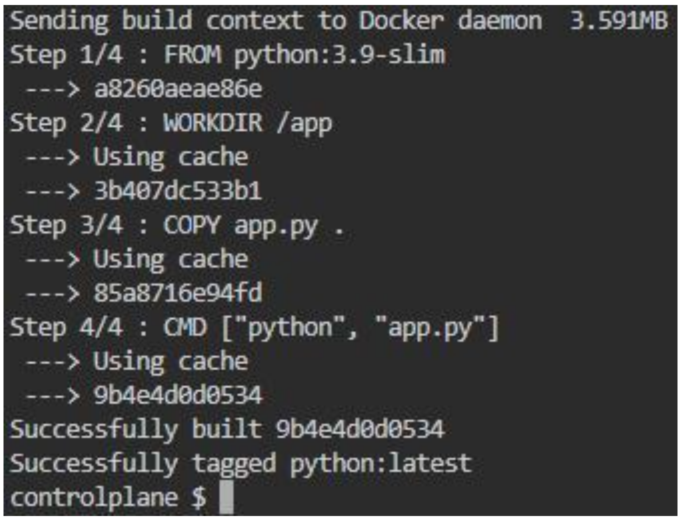
docker build: The command to build the Docker image.

Syntax:

docker build [OPTIONS] PATH | URL

-t myapp:v1: Specifies the name and tag for the image **docker build -t myapp:v1 .**

. :Indicates that the current directory is the build context.



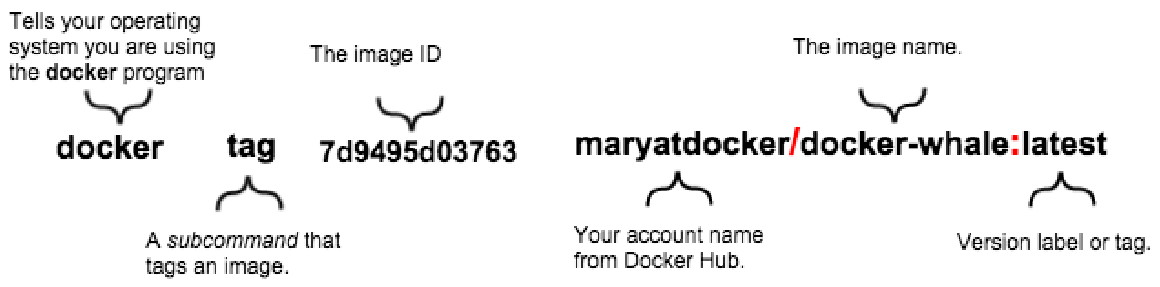
```
Sending build context to Docker daemon  3.591MB
Step 1/4 : FROM python:3.9-slim
 ---> a8260aeae86e
Step 2/4 : WORKDIR /app
 ---> Using cache
 ---> 3b407dc533b1
Step 3/4 : COPY app.py .
 ---> Using cache
 ---> 85a8716e94fd
Step 4/4 : CMD ["python", "app.py"]
 ---> Using cache
 ---> 9b4e4d0d0534
Successfully built 9b4e4d0d0534
Successfully tagged python:latest
controlplane $
```

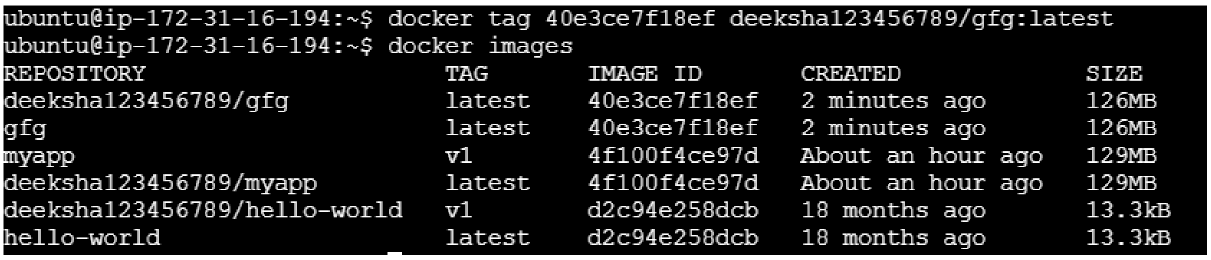
Docker tag image

The docker tag command in Docker assigns a new name and optional tag to an existing Docker image. It's used to rename an image for various reasons, like changing the tag to a specific version, or environment, or even for pushing to a registry with a different name and tag.

Syntax:

docker image tag <SOURCE_IMAGE[:TAG]> <TARGET_IMAGE[:TAG]>


Tells your operating system you are using the docker program
The image ID
The image name.
docker
tag
7d9495d03763
maryatdocker/docker-whale:latest
A subcommand that tags an image.
Your account name from Docker Hub.
Version label or tag.



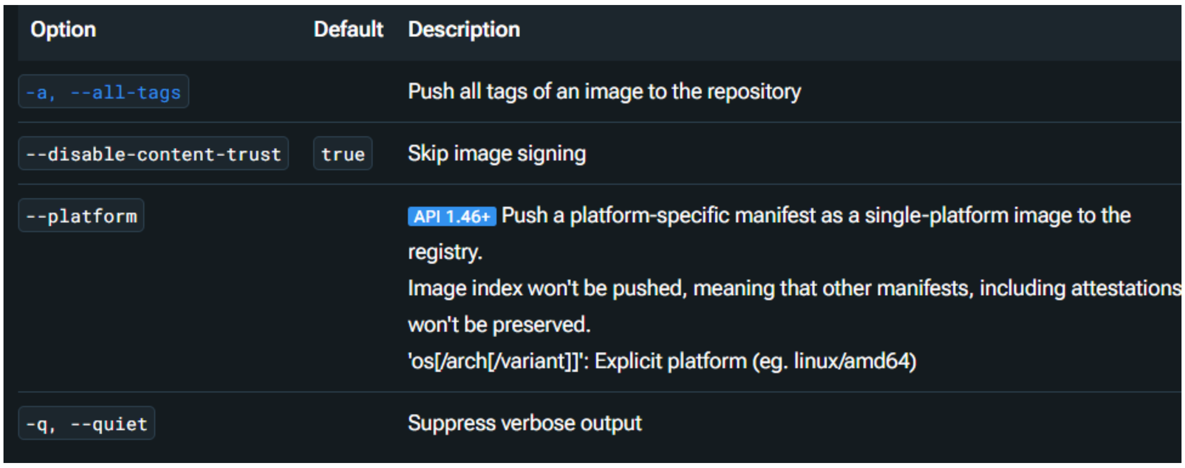
```
ubuntu@ip-172-31-16-194:~$ docker tag 40e3ce7f18ef deeksha123456789/gfg:latest
ubuntu@ip-172-31-16-194:~$ docker images
REPOSITORY                     TAG       IMAGE ID       CREATED            SIZE
deeksha123456789/gfg           latest    40e3ce7f18ef   2 minutes ago      126MB
gfg                            latest    40e3ce7f18ef   2 minutes ago      126MB
myapp                          v1        4f100f4ce97d   About an hour ago  129MB
deeksha123456789/myapp         latest    4f100f4ce97d   About an hour ago  129MB
deeksha123456789/hello-world   v1        d2c94e258dcb   18 months ago      13.3kB
hello-world                    latest    d2c94e258dcb   18 months ago      13.3kB
```

Docker push image

The docker push command is used to upload a tagged Docker image to a remote registry like Docker Hub or a private registry. It's the opposite of docker pull, which downloads images from a registry.

syntax:

docker push <image_name:tag>



Option	Default	Description
`-a, --all-tags`		Push all tags of an image to the repository
`--disable-content-trust`	`true`	Skip image signing
`--platform`		API 1.46+ Push a platform-specific manifest as a single-platform image to the registry. Image index won't be pushed, meaning that other manifests, including attestations won't be preserved. 'os[/arch[/variant]]': Explicit platform (eg. linux/amd64)
`-q, --quiet`		Suppress verbose output