Command Line Interface: Docker Engine

Killian Anderson and James Haas

October 2024

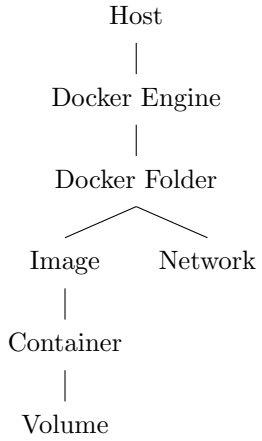
1 Introduction

The 'Docker Engine' is open ended software intended to run or service products in development. This allows for easy deployment as a 'Container' for many applications related to web-hosting. The following documentation is general vocabulary, command lines, and references for the end users reading, most of which is for educational purposes as part of a sampler. We assume there is no knowledge of 'Docker', but that the user has some knowledge of 'Unix' based systems from which to deploy the attached sampler. You may find that this guide is a summary of documentation found on their website `https://docs.docker.com/`.

2 Key Terminology

Generally, we may describe some basic terminology to help the reader better understand how Docker allows a user to easily develop and deploy executables.
Host: A host is a machine where the Docker Engine resides and functions.
Docker Engine: A Docker Engine performs many of the functions to run an instance of an application in an easily deployable format.
Docker Folder: Typically, a project will reside in a folder where all instances of Docker will run.
Network: A network is where the engine works to connect containers from an image to infrastructure or other containers.
Image: An image is set of rules that are read only, typically libraries or source code, from which the engine will follow.
Container: A container is a running instance of Docker loaded from an image.
Volume: A volume is the memory of a running container from the image.

The diagram should be useful in visualizing the key definitions introduced here in the text. ‘



3 Docker Engine Download

It is assumed here that you do not have any previous engine installed on your local machine. Now, if you do you must remove any prior files before installation of the latest versions. You may refer to the documentation regarding installation here: `https://docs.docker.com/engine/install/`.

Generally, we have summarized the installation process based on your Operating System's (OS) respective package manager. Note, replace OS_NAME with the name of the OS you are utilizing for your docker installation in lowercase lettering.

3.1 Installation Commands

Any OS using the 'APT' package manager;

```
sudo install -m 0755 -d /etc/apt/keyrings
```

```
sudo curl -fsSL
https://download.docker.com/linux/OS_NAME/gpg -o /etc/apt/keyrings/-
docker.asc
```

```
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

```
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

You may need to change VERSION_CODENAME with UBUNTU_CODENAME depending on which distribution of linux you are using.

```
echo \
"deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/OS_NAME \
$(. /etc/os-release && echo"$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
sudo apt-get update
```

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin
```

Any OS using the 'YUM' package manager;

```
sudo yum-config-manager --add-repo
https://download.docker.com/linux/OS_NAME/docker-ce.repo
```

```
sudo yum install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
```

Any OS using the 'DNF' package manager;

```
sudo dnf-3 config-manager --add-repo
https://download.docker.com/linux/OS_NAME/docker-ce.repo
```

```
sudo dnf install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
```

Once the installation has been completed you may use the command to start the engine.

```
sudo systemctl start docker
```

3.2 Non-root User Management

These steps are for users that want to manage docker without super permissions as root. You may reference the post installation page, `https://docs.docker.com/engine/install/linux-postinstall`, on the the Docker website.

You need to add a group under which the engine will run.

```
sudo groupadd docker
```

Subsequently, add the user (or your USER).

```
sudo usermod -aG docker $USER
```

You must force the new group to take effect with the user changes.

```
newgrp docker
```

Reminder, the user running the engine will need proper permissions to execute.

```
sudo chown "$USER":"$USER" /home/"$USER"/.docker -R
sudo chmod g+rwx "$HOME/.docker" -R
```

Additionally, you may force the system to boot the engine every time it starts up. This is best practice to do under a non-root user.

```
sudo systemctl enable docker.service
sudo systemctl enable containerd.service
```

You may stop the forced system start behavior with the commands.

```
sudo systemctl disable docker.service
sudo systemctl disable containerd.service
```

4 Documentation Regarding Command Line Interface (CLI)

It is noted that this is a summary of references from `https://docs.docker.com/reference/#command-line-interfaces-clis` for the management of a 'Container'.

It shall be noted that 'Docker' is a platform intended for servicing products in development and easy deployment as a 'container.' In essence, Docker may function as a virtual machine. You will need to install the Docker engine on your local machine in order to host a service. This tutorial is dedicated to the CLI necessary to manage a container deployed as a service.

All commands typically will begin with the word 'docker' typed into the CLI once the engine has been installed.

```
docker [option] [command] [arguments]
```

4.1 Management Commands

Usage is to retrieve a list of containers.

```
docker ps
```

```
docker ps -a
```

Usage is to create an image.

```
docker create [image]
```

Usage is to create an image interactively (with prompts).

```
docker create -it [image]
```

Usage is to rename a container

```
docker rename
```

Usage is to remove a container.

```
docker rm
```

```
docker rm -f
```

Usage is to retrieve logs of an executed container.

```
docker logs[container]
```

Usage is to retrieve logs of an executed container with an interval with duration or end date.

```
docker logs -f –until=[interval] [container]
```

Usage is to retrieve (command) events from the server.

```
docker events [container]
```

Usage is to update a container dynamically.

```
docker update [container]
```

Usage is to list a mapping a port of a container.

```
docker port [container]
```

Usage is to retrieve processes from a container.

```
docker top [container]
```

Usage is to retrieve a data stream of a executing container.

```
docker stats [container]
```

Usage is to retrieve a list of changed files and directories in a container.

```
docker diff [container]
```

Usage is to copy files and/or folders between a container and the local directory.

```
docker cp [file path] CONTAINER:[path]
```

4.2 Run, Start, and Stop Commands

Usage is to create and run a new image.

```
docker run [image] [command]
```

Usage is to create and run a container from an image.

```
docker run –name [container] [image]
```

Usage is to publish a container from an image with the respective port.

```
docker run -p [host port]:[container port] [image]
```

Usage is to remove an image with associated containers and volumes.

```
docker run –rm [image]
```

Usage is to run an image in background.

```
docker run -d [image]
```

Usage is to run image with interactive prompt.

```
docker run -it [image]
```

Usage is to start an executable container.

```
docker start [container]
```

Usage is to stop an executable container processes.

```
docker stop [container]
```

Usage is to restart an executable container processes.

```
docker restart [container]
```

Usage is to pause (or rather suspend) an executable container processes.

```
docker pause [container]
```

Usage is to wait or block an executable container until an exit print has been made.

```
docker wait [container]
```

Usage is to attach an executable container. This includes the terminal input, output, and/or errors of an executed container.

```
docker attach [container]
```

Usage is to execute a container with interactive prompt and the shell (which most certainly will be bash).

```
docker exec -it [container] [shell]
```

Usage is to build and/or rebuild services in containers.

```
docker compose build
```

Usage is to create services for a container.

```
docker compose create
```

Usage is to stop and remove containers and networks.

```
docker compose down
```

Usage is to create and start containers.

```
docker compose up
```

Usage is to execute a command in a running container.

```
docker compose exec
```

4.3 Image commands

Usage is to build an image based on instructions in the Dockerfile.

```
docker build [docker file path]
```

Usage is to build a container from an image.

```
docker build [container] (or . for current directory)
```

Usage is to build from a file path by an applied tag.

```
docker build -t [name]:[tag] [docker file path]
```

Usage is to build from a specific Dockerfile.

```
docker build -f [docker file path]
```

Usage is to download an image from a registry.

```
docker pull [image]
```

Usage is to upload an image to a registry.

```
docker push [image]
```

Usage is to create an image from a tarball (downloadable) contents.

```
docker import [url/file name]
```

Usage is to create a new image from container changes. Useful for debugging or development.

```
docker commit [container] [new image]
```

Usage is to create a target image tag for an identifier.

```
docker tag [image] [image]:[tag]
```

Usage is to list all images.

```
docker images
```

Usage is to retrieve history of an image.

```
docker history [image]
```

Usage is to remove more than one image.

```
docker rmi [image]
```

Usage is to load an image from tarball (downloadable) file.

```
docker load –image [tar file]
```

Usage is to save an image as a tarball (downloadable) file.

```
docker save [image] > [tar file]
```

Usage is to remove unused containers, networks, images.

```
docker image prune
```

4.4 Networking Commands

Usage is to list all networks for the hosts.

```
docker network ls
```

Usage is to remove a network from the host. Any container must be disconnected beforehand.

```
docker network rm [network]
```

Usage is to display information regarding a network or networks.

```
docker network inspect [network]
```

Usage is to connect a container to a network.

```
docker network connect [network] [container]
```

Usage is to disconnect a container from a network.

```
docker network disconnect [network] [container]
```