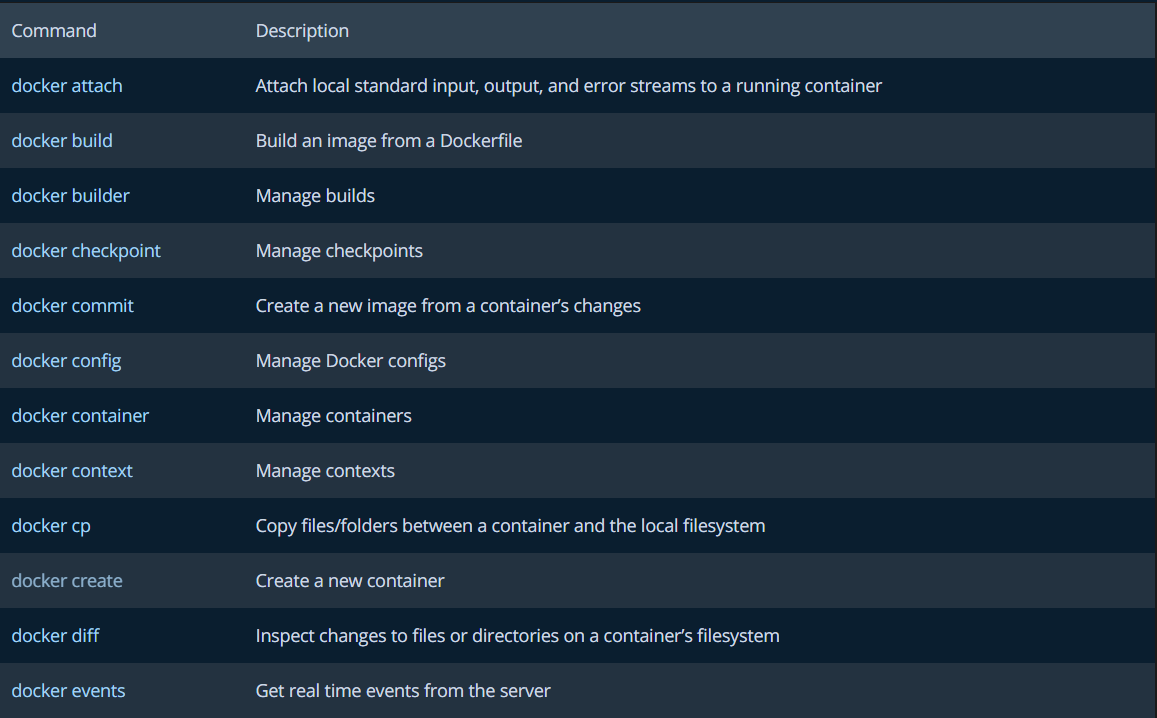
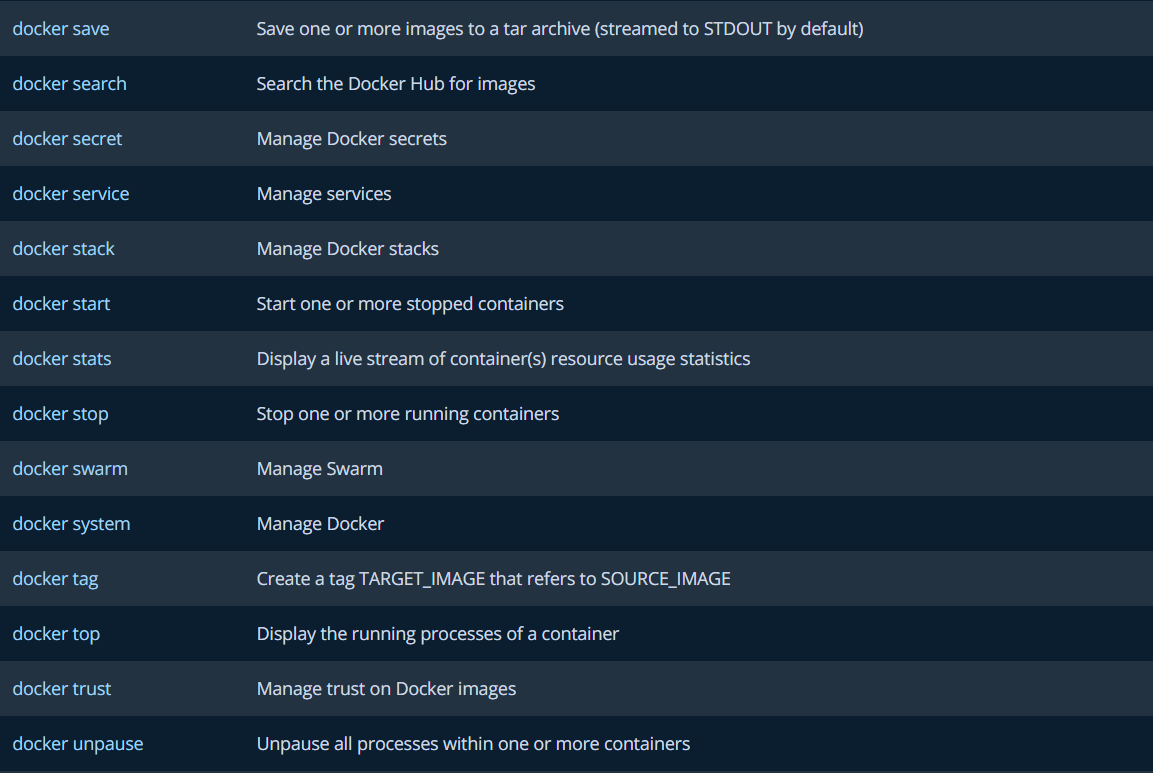
Busy box in docker is a very lightweight image with few Linux commands. We will use as a test subject to understand the Image concept in Docker.

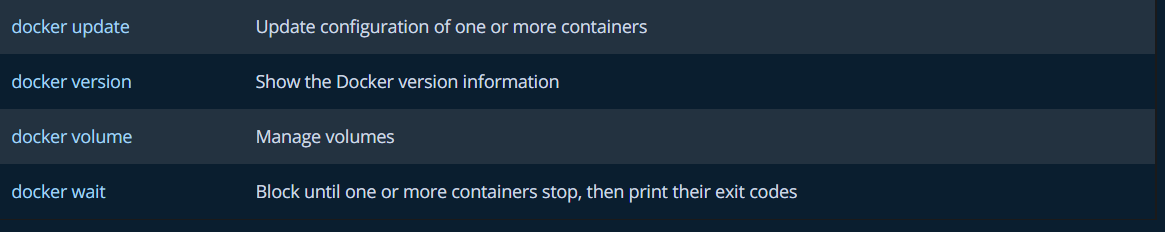
**Child commands**











Run Basic Commands

**Fetch the Busybox image from the Docker registry and save it to your system.**

Command- docker pull busybox

**To get list of Images**

Command- docker images or docker images –all

**Run a Docker container based on this image.**

Command - docker run busybox

**Run linux commands over the image**

Command - docker run busybox ls

Command- docker run busybox echo "hello from busybox"

**Run a container from the image and enter the BusyBox shell:**

docker run -it --rm busybox

**Now you can run the shell commands like ls or uname or date**

More Commands to test

Working with Dockerfiles using Busybox

## dockerfiles

The first step to creating an image of your tool is to write a file called Dockerfile. Each line in a Dockerfile is a set of instructions that are to build an image. Here's the most simple Dockerfile you can have. Begin by creating your own Dockerfile with these contents.

FROM busybox

LABEL Ramanpreet, preetraman500@gmail.com

ENTRYPOINT ["date"]

Each line in the file must begin with a recognised Dockerfile command. In this case the commands are FROM and MAINTAINER. The first described a base Docker image that we'll build our own on top of. Here we use the 'busybox' image. On the second line we specify the name and email address person who is responsible for this image. Once you've created this image you can build it with the command:

You can test this example Dockerfile by building it into an image using docker build. The --tag specifies what the image should be called. In this case we'll call the image "my\_image". Run the following command in the same directory you have placed your Dockerfile.

docker build --tag my\_image .

If this builds successfully you should have a Docker image called 'my\_image' Docker image available on your system. You list all your images with the command.

docker images

## running a container

When you're building a container it's useful to be able to log into it so you can experiment. You can create a container from this image and log in using the --interactive and --tty flags. You should specify a shell command to run also, in this case we use /bin/bash which is the command to start the bash shell. If you're not too familiar with what bash is, it's fine to use this command, without understanding the details, whenever you want to log into a container.

docker run --interactive --tty my\_image /bin/bash

Once you've logged into a container you can experiment with listing the internal file system or the current version of Linux.

ls

uname

The output of both these commands comes from the operating system inside the container, not your host computer.

## an image to do something

You will want to create an image to do something useful that you can share with others. We'll create an example Docker that returns the current time when run.

DOCKERFILE:FROM busybox

LABEL Ramanpreet, preetraman500@gmail.com

ENTRYPOINT ["date"]

This uses the ENTRYPOINT directive to specify this command gets run when the container gets run. In this case we specify the date should be run. Build and run this container, and you should see your image tells you the current date and time.

docker build --tag timepiece .

docker run timepiece

You can specify different date output formats by passing these as an argument to the docker run command. For example you can pass a format string and the date will be returned in this format.

SHELL:docker run timepiece +"%Y-%m-%d"

**Show all containers that are currently running.**

Command - docker ps – a

**Delete images that you no longer need.**

Command - docker rm #containerid