**Docker file Parameters**

**FROM:** Initializes a new build stage and sets the Base Image

**RUN:** Will execute any commands in a new layer

**CMD:** Provides a default for an executing container. There can only be one CMD instruction in a Docker file

**LABEL:** Adds metadata to an image

**EXPOSE:** Informs Docker that the container listens on the specified network ports at runtime

**ENV:** Sets the environment variable <key> to the value <value>

**ADD:** Copies new files, directories or remote file URLs from <src> and adds them to the filesystem of the image at the path <dest>.

**COPY:** Copies new files or directories from <src> and adds them to the filesystem of the container at the path <destui**ENTRYPOINT:** Allows for configuring a container that will run as an executable

**VOLUME:** Creates a mount point with the specified name and marks it as holding externally mounted volumes from native host or other containers

**USER:** Sets the username (or UID) and optionally the user group (or GID) to use when running the image and for any RUN, CMD, and ENTRYPOINT instructions that follow it in the docker file

**WORKDIR:** Sets the working directory for any RUN, CMD, ENTRYPOINT, COPY, and ADD instructions that follow it in the Docker file

**ARG:** Defines a variable that users can pass at build-time to the builder with the docker build command, using the --build-arg <varname>=<value> flag

**ONBUILD:** Adds a trigger instruction to the image that will be executed at a later time, when the image is used as the base forced another build

**HEALTHCHECK:** Tells Docker how to test a container to check that it is still working

**SHELL:** Allows the default shell used for the shell form of commands to be overridden

**Building image by piping the Docker file through STDIN:**

docker image build -t <NAME>:<TAG> -<<EOF

Build instructions

EOF

Example:

docker image build -t linuxacademy/nginx:stind --rm -<<EOF

FROM nginx: latest

VOLUME ["/usr/share/nginx/html/"]

EOF

**Building an image using a URL:**

docker image build -t <NAME>:<TAG> <GIT\_URL>#<REF>

docker image build -t <NAME>:<TAG> <GIT\_URL>#:<DIRECTORY>

docker image build -t <NAME>:<TAG> <GIT\_URL>#<REF>:<DIRECTORY>

Example:

docker image build -t linuxacademy/weather-app:github https://github.com/linuxacademy/content-weather-app.git#remote-build

**Building an image from a zip file:**

docker image build -t <NAME>:<TAG> - < <FILE>.tar.gz

Example:

cd docker\_images

mkdir tar\_image

cd tar\_image

git clone https://github.com/linuxacademy/content-weather-app.

cd content-weather-app

git checkout remote-build

tar -zcvf weather-app.tar.gz Dockerfile

docker image build -t linuxacademy/weather-app:from-tar - < weather-app.tar.g

**Docker Machine Overview**

Docker Machine is a tool that lets you install Docker Engine on virtual hosts and manage the hosts with **“docker-machine”** commands. You can use Machine to create Docker hosts on your local Mac or Windows box, on your company network, in your data center, or on cloud providers like Azure, AWS, or Digital Ocean.

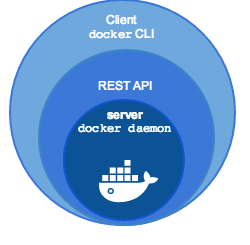
Using docker-machine commands, you can start, inspect, stop, and restart a managed host, upgrade the Docker client and daemon, and configure a Docker client to talk to your host.



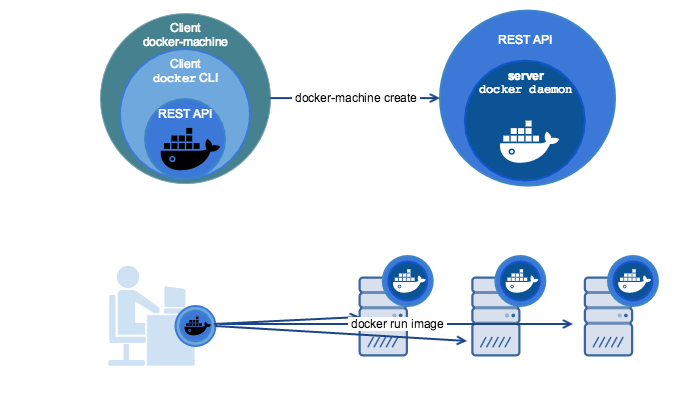
Above diag show the machine trying to run docker-engine on remote machines

Below diagram illustrates the difference between the docker engine and docker machine:

**Docker Engine:** The client-server application made up of the Docker daemon, a REST API that specifies interfaces for interacting with the daemon, and a command line interface (CLI) client that talks to the daemon (through the REST API wrapper). Docker Engine accepts **docker commands from the CLI**, such as docker run <image>, docker ps to list running containers, docker image ls to list images, and so on.



**Docker Machine** is a tool for provisioning and managing your Dockerized hosts (hosts with Docker Engine on them). Typically, you install Docker Machine on your local system. Docker Machine has its own command line client docker-machine and the Docker Engine client, docker. You can use Machine to install Docker Engine on one or more virtual systems. These virtual systems can be local (as when you use Machine to install and run Docker Engine in VirtualBox on Mac or Windows) or remote (as when you use Machine to provision Dockerized hosts on cloud providers). The Dockerized hosts themselves can be thought of, and are sometimes referred to as, managed “machines”.



**How to install docker machine on the nodes?**

First ensure that docker is installed on the machine before we proceed further, now follow the details further mentioned

$ base=https://github.com/docker/machine/releases/download/v0.14.0 &&

curl -L $base/docker-machine-$(uname -s)-$(uname -m) >/tmp/docker-machine &&

sudo install /tmp/docker-machine /usr/local/bin/docker-machine

Once installation is done, execute the below command to verify the version

$ docker-machine version

docker-machine version 0.14.0, build 9371605

