



*Docker Machine Guide by John Cusey*

*January 24, 2019*



Contents

[2 Introduction 3](#_Toc536095533)

[2.1 What is Docker Machine? 3](#_Toc536095534)

[2.2 Why should I use it? 4](#_Toc536095535)

[2.3 WHAT IS the difference between Docker Engine and Docker Machine? 5](#_Toc536095536)

[3 Command Line 6](#_Toc536095537)

[3.1 Create Machine 6](#_Toc536095538)

[3.2 Remove (RM) Machine 7](#_Toc536095539)

[3.3 restart machine 7](#_Toc536095540)

[3.4 stop machine 7](#_Toc536095541)

[3.5 machine environment variablies (env) 8](#_Toc536095542)

[3.6 Switching which machine is active 8](#_Toc536095543)

# Introduction

You can use Docker Machine to:

* Install and run Docker on Mac or Windows
* Provision and manage multiple remote Docker hosts
* Provision Swarm clusters

## What is Docker Machine?

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.

Point the Machine CLI at a running, managed host, and you can run docker commands directly on that host. For example, run docker-machine env default to point to a host called default, follow on-screen instructions to complete env setup, and run docker ps, docker run hello-world, and so forth.

Machine was the only way to run Docker on Mac or Windows before Docker v1.12. Starting with the beta program and Docker v1.12, Docker Desktop for Mac and Docker Desktop for Windows are available as native apps and the better choice for this use case on newer desktops and laptops. We encourage you to try out these new apps. The installers for Docker Desktop for Mac and Docker Desktop for Windows include Docker Machine, along with Docker Compose.

## Why should I use it?

Docker Machine enables you to provision multiple remote Docker hosts on various flavors of Linux.

Additionally, Machine allows you to run Docker on older Mac or Windows systems, as described in the previous topic.

Docker Machine has these two broad use cases.

* **I have an older desktop system and want to run Docker on Mac or Windows**



If you work primarily on an older Mac or Windows laptop or desktop that does not meet the requirements for the new Docker Desktop for Mac and Docker Desktop for Windows apps, then you need Docker Machine run Docker Engine locally. Installing Docker Machine on a Mac or Windows box with the Docker Toolbox installer provisions a local virtual machine with Docker Engine, gives you the ability to connect it, and run docker commands.

* **I want to provision Docker hosts on remote systems**

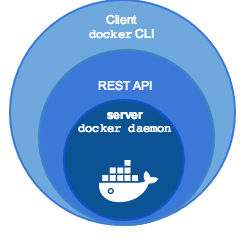


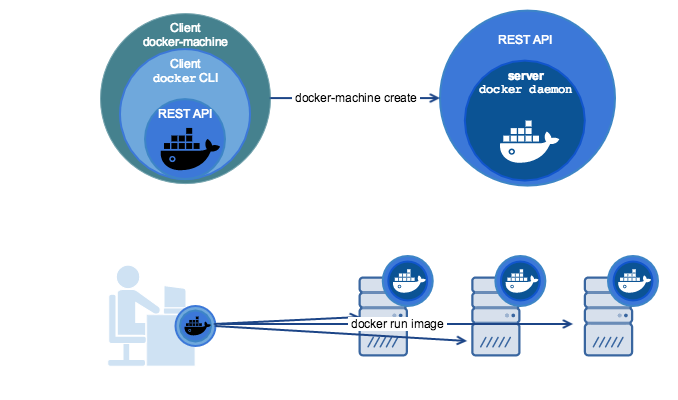
Docker Engine runs natively on Linux systems. If you have a Linux box as your primary system, and want to run docker commands, all you need to do is download and install Docker Engine. However, if you want an efficient way to provision multiple Docker hosts on a network, in the cloud or even locally, you need Docker Machine.

Whether your primary system is Mac, Windows, or Linux, you can install Docker Machine on it, use docker-machine commands to provision, and manage large numbers of Docker hosts. It automatically creates hosts, installs Docker Engine on them, and then configures the docker clients. Each managed host (“**machine**”) is the combination of a Docker host and a configured client.

## WHAT IS the difference between Docker Engine and Docker Machine?

When people say “Docker” they typically mean **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**”.Command 

# Command Line

## Create Machine

<https://docs.docker.com/machine/reference/create/>

Syntax

$ docker-machine create --driver virtualbox <machine-name>

Usage

$ docker-machine create --driver virtualbox dev

## Remove (RM) Machine

<https://docs.docker.com/machine/reference/rm/>

Syntax

Get the machine name

$ docker-machine ls

Remove machine

$ docker-machine rm <machine-name>

Usage

$ docker-machine rm baz

## restart machine

<https://docs.docker.com/machine/reference/restart/>

Syntax

$ docker-machine restart <machine-name>

Usage

$ docker-machine restart dev

## stop machine

<https://docs.docker.com/machine/reference/stop/>

Syntax

Get the machine name

$ docker-machine ls

Remove machine

$ docker-machine stop <machine-name>

Usage

$ docker-machine stop dev

## machine environment variablies (env)

<https://docs.docker.com/machine/reference/env/>

Syntax

Get the environment variables

$ docker-machine env <machine-name>

Usage

Get the environment variables

$ docker-machine env temp.sysadmin.local

export DOCKER\_TLS\_VERIFY=”1”

export DOCKER\_HOST = “tcp://192.168.99.100:2376”

export DOCKER\_CERT\_PATH = “/home/peter/.docker/machine/machines/ temp.sysadmin.local”

# Run this command to configure your shell:

# eval $(docker-machine env temp.sysadmin.local)

Set the environment variables on your computer

$ eval $(docker-machine env temp.sysadmin.local)

Check to see if environments are set

$ echo DOCKER\_HOST

“tcp://192.168.99.100:2376”

See which is the running docker machine (the star mean that machine is active)

$ docker-machine ls

NAME ACTIVE DRIVER STATE URL SWARM DOCKER ERROR

temp.sysadmin.local \* virtualbox Running tcp://192.168.99.100:2376 v1.11.2

## Switching which machine is active

Usage

Get the environment variables

$ docker-machine env temp.sysadmin.local

export DOCKER\_TLS\_VERIFY=”1”

export DOCKER\_HOST = “tcp://192.168.99.100:2376”

export DOCKER\_CERT\_PATH = “/home/peter/.docker/machine/machines/ temp.sysadmin.local”

# Run this command to configure your shell:

# eval $(docker-machine env temp.sysadmin.local)

Setting the environemt variables THIS WILL CHANGE THIS MACHINE TO THE ACTIVE MACHINE

Set the environment variables on your computer

$ eval $(docker-machine env temp.sysadmin.local)