Install Docker on Linux ( Redhat or Centos )

**To install Docker on an Amazon Linux instance**

1. Launch an instance with the Amazon Linux AMI. For more information, see [Launching an Instance](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/launching-instance.html) in theAmazon EC2 User Guide for Linux Instances.
2. Connect to your instance. For more information, see [Connect to Your Linux Instance](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstances.html) in the Amazon EC2 User Guide for Linux Instances.
3. Update the installed packages and package cache on your instance.

[ec2-user ~]$ **sudo yum update -y**

1. Install Docker.

[ec2-user ~]$ **sudo yum install -y docker**

1. Start the Docker service.

[ec2-user ~]$ **sudo service docker start**

Starting cgconfig service: [ OK ]

Starting docker: [ OK ]

1. Add the ec2-user to the docker group so you can execute Docker commands without using sudo.

[ec2-user ~]$ **sudo usermod -a -G docker ec2-user**

1. Log out and log back in again to pick up the new docker group permissions.
2. Verify that the ec2-user can run Docker commands without sudo.

[ec2-user ~]$ **docker info**

Containers: 2

Images: 24

Storage Driver: devicemapper

Pool Name: docker-202:1-263460-pool

Pool Blocksize: 65.54 kB

Data file: /var/lib/docker/devicemapper/devicemapper/data

Metadata file: /var/lib/docker/devicemapper/devicemapper/metadata

Data Space Used: 702.3 MB

Data Space Total: 107.4 GB

Metadata Space Used: 1.864 MB

Metadata Space Total: 2.147 GB

Library Version: 1.02.89-RHEL6 (2014-09-01)

Execution Driver: native-0.2

Kernel Version: 3.14.27-25.47.amzn1.x86\_64

Operating System: Amazon Linux AMI 2014.09

## Sign up for a Docker Hub Account

 go to [https://hub.docker.com](https://hub.docker.com/) and sign up

Docker Command Line

* Docker Command is used to manage the docker
* Docker command can be used to manage the local host or remote host by “docker –H host
* By default, the Docker command line stores its configuration files in a directory called .docker within your HOME directory. However, you can specify a different location via the DOCKER\_CONFIG environment variable or the--config command line option.
* If both are specified, then the --config option overrides the DOCKER\_CONFIG environment variable.

**Run a Docker Container**

* Run a container using Command “docker run”

docker run ubuntu /bin/**echo** 'Hello world‘

Hello world

* The docker run combination *runs* containers.
* Next we specified an image: ubuntu.
* This is the source of the container we ran.
* Docker calls this an image.
* In this case we used the Ubuntu operating system image.
* When you specify an image, Docker looks first for the image on your Docker host.
* If it can’t find it then it downloads the image from the public image registry: [Docker Hub](https://hub.docker.com/).
* Next we told Docker what command to run inside our new container:/bin/**echo** 'Hello world'

**An interactive container**

* You can get the console of the container by running a shell

docker run -t -i ubuntu /bin/bash

root@af8bae53bdd3:/*#*

-t flag assigns a pseudo-tty or terminal inside our new container and

the -i flag allows us to make an interactive connection by grabbing the standard in (STDIN) of the container.

root@af8bae53bdd3:/*#*

root@af8bae53bdd3:/*# pwd* /

root@af8bae53bdd3:/*# ls*

bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var

root@af8bae53bdd3:/*# exit*

**A daemonized Hello world**

$docker run -d ubuntu /bin/sh -c "while true; do echo hello world; sleep 1; done“

* -d flag tells Docker to run the container and put it in the background, to daemonize it.
* The docker ps command queries the Docker daemon for information about all the containers it knows about.

$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 1e5535038e28 ubuntu /bin/sh -c 'while tr 2 minutes ago Up 1 minute insane\_babbage

$ docker ps -a

To List all container including stopped one

**Container Logs**

The docker logs command looks inside the container and returns its standard output: in this case the output of our command hello world.

* $ docker logs insane\_babbage

hello world

hello world

hello world

. . .

**Stop a Docker Contianer**

* The docker stop command tells Docker to politely stop the running container. If it succeeds it will return the name of the container it has just stopped.

$ docker **stop** insane\_babbage

$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

**Run a Web Server in a Container using Nginx**

* Create a Website Directory to host

$mkdir -p ~/docker-nginx/html

$cd ~/docker-nginx/html

$echo “My Docker Website”> index.html

* Start the web server container

sudo docker run --name docker-nginx -p 80:80 -d -v ~/docker-nginx/html:/usr/share/nginx/html nginx

* -v specifies that we're linking a volume
* -p denotes which port the host will listen and which port container will listen
* the part to the left of the : is the location of our file/directory on our virtual machine (~/docker-nginx/html)
* the part to the right of the : is the location that we are linking to in our container (/usr/share/nginx/html)
* Verify
* Open Browser in your machine and <http://public> IP or hostname

**Working with Images**

* Docker images will  list the images you have locally on our host.

$ docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

ubuntu 14.04 1d073211c498 3 days ago 187.9 MB

busybox latest 2c5ac3f849df 5 days ago 1.113 MB

training/webapp latest 54bb4e8718e8 5 months ago

* A repository potentially holds multiple variants of an image. In the case of ourubuntu image you can see multiple variants covering Ubuntu 10.04, 12.04, 12.10, 13.04, 13.10 and 14.04
* Tag is represt by name:tag ( ubuntu:12:04 )

$ docker **run** -t -i ubuntu:12.04 /bin/bash

* Docker pull pulls the image from Repository
* By default it pulls from docker hub
* You can specify your own registry as well

$ docker pull centos

Pulling repository centos

b7de3133ff98: Pulling dependent layers

5cc9e91966f7: Pulling fs layer

511136ea3c5a: Download complete

ef52fb1fe610: Download complete . . .

Status: Downloaded newer image for centos

* Docker Search to search in the repository
* In Docker hub

**Searching images in registry**

$docker login

$docker search sinatra

NAME DESCRIPTION STARS OFFICIAL AUTOMATED

training/sinatra Sinatra training image 0 [OK]

marceldegraaf/sinatra Sinatra test app 0

**Pulling our image**

* Docker pull to pull images
* In Docker hub

$docker pull training/sinatra

**Building a sample app**

**To create a Docker image of a PHP web application**

1. Install **git** and use it to clone the simple PHP application from your GitHub repository onto your system.
   1. Install git.

[ec2-user ~]$ **sudo yum install -y git**

* 1. Clone the simple PHP application onto your system.

[ec2-user ~]$ **git clone https://github.com/awslabs/ecs-demo-php-simple-app**

1. Change directories to the ecs-demo-php-simple-app folder.

[ec2-user ~]$ **cd ecs-demo-php-simple-app**

1. Examine the Dockerfile in this folder. A Dockerfile is a manifest that describes the base image to use for your Docker image and what you want installed and running on it. For more information about Dockerfiles, go to the[Dockerfile Reference](https://docs.docker.com/reference/builder/).

[ec2-user ecs-demo-php-simple-app]$ **cat Dockerfile**

FROM ubuntu:12.04

# Install dependencies

RUN apt-get update -y

RUN apt-get install -y git curl apache2 php5 libapache2-mod-php5 php5-mcrypt php5-mysql

# Install app

RUN rm -rf /var/www/\*

ADD src /var/www

# Configure apache

RUN a2enmod rewrite

RUN chown -R www-data:www-data /var/www

ENV APACHE\_RUN\_USER www-data

ENV APACHE\_RUN\_GROUP www-data

ENV APACHE\_LOG\_DIR /var/log/apache2

EXPOSE 80

CMD ["/usr/sbin/apache2", "-D", "FOREGROUND"]

This Dockerfile uses the Ubuntu 12.04 image. The RUN instructions update the package caches, install some software packages for the web server and PHP support, and then add your PHP application to the web server's document root. The EXPOSE instruction exposes port 80 on the container, and the CMD instruction starts the web server.

1. Build the Docker image from your Dockerfile. Substitute *my-dockerhub-username* with your Docker Hub user name.

[ec2-user ecs-demo-php-simple-app]$ **docker build -t *my-dockerhub-username*/amazon-ecs-sample .**

1. Run **docker images** to verify that the image was created correctly and that the image name contains a repository that you can push to (in this example, your Docker Hub user name).

[ec2-user ecs-demo-php-simple-app]$ docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

*my-dockerhub-username*/amazon-ecs-sample latest 43c52559a0a1 12 minutes ago 258.1 MB

ubuntu 12.04 78cef618c77e 3 weeks ago 133.7 MB

1. Run the newly built image. The -p 80:80 option maps the exposed port 80 on the container to port 80 on the host system. For more information about **docker run**, go to the [Docker run reference](https://docs.docker.com/reference/run/).
2. [ec2-user ecs-demo-php-simple-app]$ **docker run -p 80:80 *my-dockerhub-username*/amazon-ecs-sample**

apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2 for ServerName

**Note**

Output from the Apache web server is displayed in the terminal window. You can ignore the "Could not reliably determine the server's fully qualified domain name" message.

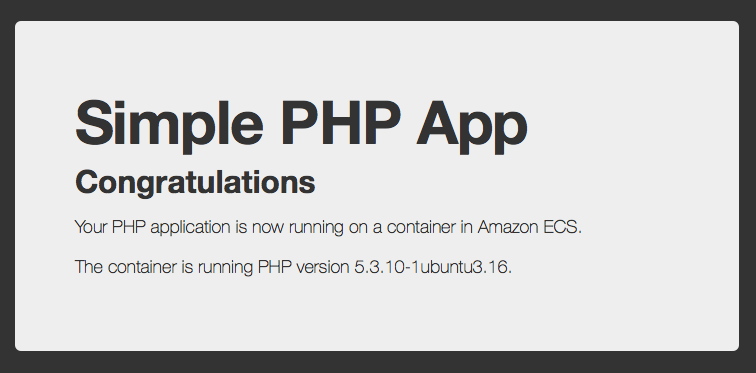
1. Open a browser and point to the server that is running Docker and hosting your container.
   1. If you are using an EC2 instance, this is the **Public DNS** value for the server, which is the same address you use to connect to the instance with SSH. Make sure that the security group for your instance allows inbound traffic on port 80.
   2. If you are running Docker locally on a Linux computer, point your browser to <http://localhost/>.
   3. If you are using **boot2docker** on a Windows or Mac computer, find the IP address of the VirtualBox VM that is hosting Docker with the **boot2docker ip** command.
   4. $ **boot2docker ip**

192.168.59.103

* 1. If you are using **docker-machine** on a Windows or Mac computer, find the IP address of the VirtualBox VM that is hosting Docker with the **docker-machine ip** command, substituting *machine-name* with the name of the docker machine you are using.
  2. $ **docker-machine ip *machine-name***

192.168.59.103

1. You should see a web page running the simple PHP app.



1. Stop the Docker container by typing **Ctrl**+**c**.

## Java Hello World for Docker

In this example, we are going to run the typical Java Hello World example in docker. We shall use Maven to setup a new project for Hello word example. The contents of our maven project descriptor should be as follows:

pom.xml

|  |  |  |
| --- | --- | --- |
|  | <project xmlns="<http://maven.apache.org/POM/4.0.0>"xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>" | |
|  | | xsi:schemaLocation="<http://maven.apache.org/POM/4.0.0> <http://maven.apache.org/xsd/maven-4.0.0.xsd>"> |

|  |  |
| --- | --- |
|  | <modelVersion>4.0.0</modelVersion> |
|  | <groupId>com.javacodegeeks</groupId> |

|  |  |
| --- | --- |
|  | <artifactId>helloworld</artifactId> |
|  | <version>1.0</version> |

|  |  |
| --- | --- |
|  | </project> |

Now, add a Java class into the source folder (**src/main/java** directory) with the following code.

HelloWorld.java

|  |  |
| --- | --- |
|  | public class HelloWorld { |
|  | public static void main(String[] args) throws Exception{ |

|  |  |
| --- | --- |
|  | System.out.println("Hello World"); |
|  | } |

|  |  |
| --- | --- |
|  | } |

After that, create a file called Dockerfile into the root directory of the project and copy the following contents.

Dockerfile

|  |  |
| --- | --- |
|  | FROM ubuntu:14.04 |
|  | MAINTAINER javacodegeeks |

|  |  |
| --- | --- |
|  |  |
|  | RUN apt-get update && apt-get install -y python-software-properties software-properties-common |

|  |  |
| --- | --- |
|  | RUN add-apt-repository ppa:webupd8team/java |
|  |  |

|  |  |
| --- | --- |
|  | RUN echo "oracle-java8-installer shared/accepted-oracle-license-v1-1 boolean true" | debconf-set-selections |
|  |  |

|  |  |
| --- | --- |
|  | RUN apt-get update && apt-get install -y oracle-java8-installer maven |
|  |  |

|  |  |
| --- | --- |
|  | ADD . /usr/local/helloworld |
|  | RUN cd /usr/local/helloworld && mvn install |

|  |  |
| --- | --- |
|  | CMD ["java", "-cp", "/usr/local/helloworld/target/helloworld-1.0.jar", "HelloWorld"] |

## 4. Running the example

Now, we are going to build the Docker image. Open a prompt and go to the directory where the Dockerfile is located. Now, you just need to execute the following command:

Docker build command

|  |  |
| --- | --- |
| 1 | docker build -t javacodegeeks/helloworld:1.0 . |

Once the command ends, the last line should be similar to this one:

Docker build output

|  |  |
| --- | --- |
| 1 | Successfully built 29336bbbfa5c |

At this moment, Docker has created an image called **javacodegeeks/helloworld:1.0**. Now, let’s create and start a container for this image with the docker run command.

Docker run command

|  |  |
| --- | --- |
| 1 | docker run --name helloworldcontainer javacodegeeks/helloworld:1.0 |
| 2 | HelloWorld |

## BUILDING A MULTI-NODE CLUSTER USING THE DEFAULT PORTS

The following TCP port are used by Galera:

* 3306-MySQL port
* 4567-Galera Cluster
* 4568-IST port
* 4444-SST port

Before we start, we need to stop enforcing AppArmor for Docker:

$ aa-complain /etc/apparmor.d/docker

Building a multi-node cluster using the default ports is not complicated. Besides mapping the ports 1:1, we also need to set `–wsrep-node-address` to the IP address of the host.

We assume following 3 nodes

* nodea 10.10.10.10
* nodeb 10.10.10.11
* nodec 10.10.10.12

A simple cluster setup would look like this:

nodea$ docker run -d -p 3306:3306 -p 4567:4567 -p 4444:4444 -p 4568:4568 --name nodea erkules/galera:basic --wsrep-cluster-address=gcomm:// --wsrep-node-address=10.10.10.10

nodeb$ docker run -d -p 3306:3306 -p 4567:4567 -p 4444:4444 -p 4568:4568 --name nodeb erkules/galera:basic --wsrep-cluster-address=gcomm://10.10.10.10 --wsrep-node-address=10.10.10.11

nodec$ docker run -d -p 3306:3306 -p 4567:4567 -p 4444:4444 -p 4568:4568 --name nodec erkules/galera:basic --wsrep-cluster-address=gcomm://10.10.10.10 --wsrep-node-address=10.10.10.12

nodea$ docker exec -t nodea mysql -e 'show status like "wsrep\_cluster\_size"'

+--------------------+-------+

| Variable\_name | Value |

+--------------------+-------+

| wsrep\_cluster\_size | 3 |

+--------------------+-------+

### **Installation in Windows**

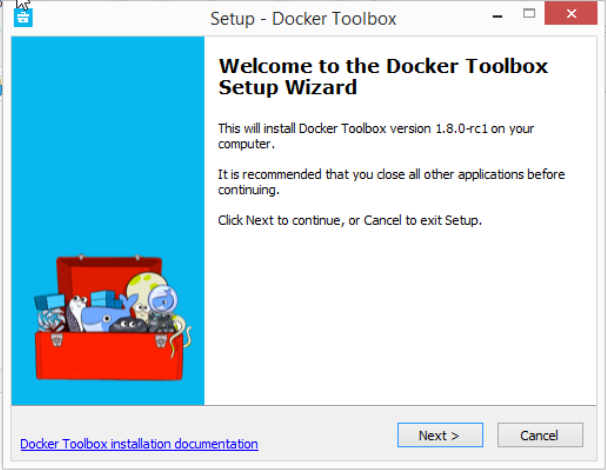
If you have VirtualBox running, you must shut it down before running the installer.

1. Download the Docker tool box from the below URL

https://www.docker.com/products/docker-toolbox

1. Click the installer link to download.
2. Install Docker Toolbox by double-clicking the installer.

The installer launches the “Setup - Docker Toolbox” dialog.



1. Press “Next” to install the toolbox.

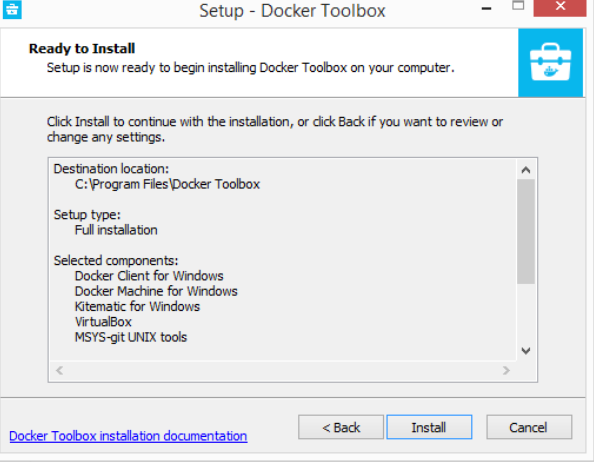
The installer presents you with options to customize the standard installation. By default, the standard Docker Toolbox installation:

* + installs executables for the Docker tools inC:\Program Files\Docker Toolbox
  + install VirtualBox; or updates any existing installation
  + adds a Docker Inc. folder to your program shortcuts
  + updates your PATH environment variable
  + adds desktop icons for the Docker Quickstart Terminal and Kitematic

This installation assumes the defaults are acceptable.

1. Press “Next” until you reach the “Ready to Install” page.

The system prompts you for your password.



1. Press “Install” to continue with the installation.

When it completes, the installer provides you with some information you can use to complete some common tasks.



1. Press “Finish” to exit.

## Running a Docker Container

To run a Docker container, you:

* create a new (or start an existing) Docker virtual machine
* switch your environment to your new VM
* use the docker client to create, load, and manage containers

Once you create a machine, you can reuse it as often as you like. Like any VirtualBox VM, it maintains its configuration between uses.

There are several ways to use the installed tools, from the Docker Quickstart Terminal or [from your shell](https://docs.docker.com/engine/installation/windows/#from-your-shell).

### **Using the Docker Quickstart Terminal**

1. Find the Docker Quickstart Terminal icon on your Desktop and double-click to launch it.

The application:

* + opens a terminal window
  + creates a default VM if it doesn’t exist, and starts the VM after
  + points the terminal environment to this VM

Once the launch completes, you can run docker commands.

1. Verify your setup succeeded by running the hello-world container.

$ docker run hello-world

Unable to find image 'hello-world:latest' locally

511136ea3c5a: Pull complete

31cbccb51277: Pull complete

e45a5af57b00: Pull complete

hello-world:latest: The image you are pulling has been verified.

Important: image verification **is** a tech preview feature **and** should **not** be

relied on to provide security.

Status: Downloaded newer image **for** hello-world:latest

Hello **from** Docker.

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.

2. The Docker daemon pulled the "hello-world" image **from** the Docker Hub.

(Assuming it was **not** already locally available.)

3. The Docker daemon created a new container **from** that image which runs the

executable that produces the output you are currently reading.

4. The Docker daemon streamed that output to the Docker client, which sent it

to your terminal.

To **try** something more ambitious, you can run an Ubuntu container **with**:

$ docker run -it ubuntu bash

For more examples **and** ideas, visit:

http://docs.docker.com/userguide/

### **Using Docker from Windows Command Prompt (cmd.exe)**

Launch a Windows Command Prompt (cmd.exe).

The docker-machine command requires ssh.exe in your PATHenvironment variable. This .exe is in the MsysGit bin folder.

Add this to the %PATH% environment variable by running:

**set** **PATH**=%**PATH**%;"c:\Program Files (x86)\Git\bin"

Create a new Docker VM.

docker-machine **create** *--driver virtualbox my-default*

Creating VirtualBox VM...

Creating SSH **key**...

**Starting** VirtualBox VM...

**Starting** VM...

**To** see how **to** **connect** Docker **to** this machine, run: docker-machine env my-**default**

The command also creates a machine configuration in theC:\USERS\USERNAME\.docker\machine\machines directory. You only need to run the create command once. Then, you can usedocker-machine to start, stop, query, and otherwise manage the VM from the command line.

* List your available machines.

C:\Users\mary> docker-machine ls

NAME ACTIVE DRIVER STATE URL SWARM

my-**default** \* virtualbox Running tcp:*//192.168.99.101:2376*

If you have previously installed the deprecated Boot2Docker application or run the Docker Quickstart Terminal, you may have a dev VM as well.

* Get the environment commands for your new VM.

C:\Users\mary> docker-machine **env** --shell **cmd** my-default

* Connect your shell to the my-default machine.

C:\Users\mary> eval "$(docker-machine env my-default)"

* Run the hello-world container to verify your setup.

C:\Users\mary> docker **run** hello-world

### **Using Docker from PowerShell**

1. Launch a Windows PowerShell window.
2. Add ssh.exe to your PATH:

PS C:\Users\mary> $Env:Path = "${Env:Path};c:\Program Files (x86)\Git\bin"

1. Create a new Docker VM.

PS C:\Users\mary> docker-machine create --driver virtualbox my-**default**

1. List your available machines.

C:\Users\mary> docker-machine ls

NAME ACTIVE DRIVER STATE URL SWARM

my-**default** \* virtualbox Running tcp:*//192.168.99.101:2376*

1. Get the environment commands for your new VM.

C:\Users\mary> docker-machine env --shell powershell my-**default**

1. Connect your shell to the my-default machine.

C:\Users\mary> eval "$(docker-machine env my-default)"

1. Run the hello-world container to verify your setup.

C:\Users\mary> docker **run** hello-world