**[Docker install on Ubuntu 14.04](http://www.bogotobogo.com/DevOps/Docker/Docker_Install_On_Ubuntu_14.php)**

**Install on Ubuntu 14.04**

Docker is an open-source project that makes creating and managing Linux containers really easy. Containers are like extremely lightweight VMs - they allow code to run in isolation from other containers but safely share the machine's resources, all without the overhead of a hypervisor.

In this article, we'll install Docker using Docker-managed release packages as well as Ubuntu managed packages. Using Docker-managed release packages ensures us get the latest release of Docker.

Docker requires a 64-bit installation regardless of our Ubuntu version.

Ubuntu Trusty comes with a 3.13.0 Linux kernel, and a docker.io package which installs Docker 0.9.1 and all its prerequisites from Ubuntu's repository.

Ubuntu (and Debian) contain a much older KDE3/GNOME2 package called docker, so the package and the executable are called docker.io.

To install Docker properly, our kernel must be 3.10 at minimum. The latest 3.10 minor version or a newer maintained version are also acceptable. That's because Kernels older than 3.10 lack some of the features required to run Docker containers. These older versions are known to have bugs which cause data loss and frequently panic under certain conditions.

We can check our current kernel version:

$ uname -r

3.13.0-40-generic

So, we met the [Prerequisites](https://docs.docker.com/installation/ubuntulinux/) (<https://docs.docker.com/engine/installation/linux/ubuntu/>)!

Let's start.

First, update our package manager. Downloads the package lists from the repositories and "updates" them to get information on the newest versions of packages and their dependencies by synchronizing the package index files fetched from /etc/apt/sources.list.

$ sudo apt-get update

**Install using Ubuntu-managed packages**

To install Docker, we need to use:

$ sudo apt-get install docker.io

To make the shell easier to use, we need to create a symlink since /usr/local/bin is for normal user programs not managed by the distribution package manager. The following command overwrites the link (/usr/local/bin/docker):

$ sudo ln -sf /usr/bin/docker.io /usr/local/bin/docker

To enable tab-completion of Docker commands in BASH, either restart BASH or:

$ source /etc/bash\_completion.d/docker.io

Or

$ source /etc/bash\_completion.d/docker

To check if Docker is running:

$ ps aux | grep docker

root 8725 0.0 0.2 362736 9352 ? Ssl 10:05 0:00 /usr/bin/docker.io -d

If we want to run docker as root user, we should add a user (in my case, 'k') to the docker group:

$ sudo usermod -aG docker k

$ id k

uid=1000(k) gid=1000(k) groups=1000(k),4(adm),24(cdrom),27(sudo),30(dip),33(www-data),46(plugdev),108(lpadmin),124(sambashare),1005(svn),131(docker)

**Install using Docker-managed release packages**

We may want to uninstall the Docker if it's already there:

$ sudo apt-get remove docker.io

Then, get the Docker package, and install it from the Docker script:

$ curl -sSL https://get.docker.com/ | sh

or

$ curl -sSL -qo- https://get.docker.com/ | sh

The vertical line pipe character before the **sh** pipes the output from standard output(**-o-**) to the terminal to the **sh** command which says to execute the file that was downloaded in the terminal as a program, assuming that the file that was downloaded first has its permissions set to allow executing the file as a program.

Let's run daemon:

$ sudo docker -d

To verify docker is installed correctly:

$ sudo docker run hello-world

This command downloads a test image and runs it in a container as we can see from the output:

$ sudo docker run hello-world

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

latest: Pulling from library/hello-world

535020c3e8ad: Pull complete

af340544ed62: Already exists

library/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.

Digest: sha256:d5fbd996e6562438f7ea5389d7da867fe58e04d581810e230df4cc073271ea52

Status: Downloaded newer image for hello-world:latest

Hello from Docker.

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

**Checking Docker version**

To see which version of Docker is installed:

$ docker -v

Docker version 1.8.1, build d12ea79

**Install latest Docker from Ubuntu Package**

To install the latest version, we need to add the Docker repository key to our local keychain:

$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 36A1D7869245C8950F966E92D8576A8BA88D21E9

[sudo] password for k:

Executing: gpg --ignore-time-conflict --no-options --no-default-keyring --homedir /tmp/tmp.0gMbMUENwN --no-auto-check-trustdb --trust-model always --keyring /etc/apt/trusted.gpg --primary-keyring /etc/apt/trusted.gpg --keyring /etc/apt/trusted.gpg.d/nagiosinc-ppa.gpg --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 36A1D7869245C8950F966E92D8576A8BA88D21E9

gpg: requesting key A88D21E9 from hkp server keyserver.ubuntu.com

gpg: key A88D21E9: "Docker Release Tool (releasedocker) " not changed

gpg: Total number processed: 1

gpg: unchanged: 1

Add the Docker repository to our apt sources list:

$ sudo sh -c "echo deb http://get.docker.io/ubuntu docker main > /etc/apt/sources.list.d/docker.list"

$ cat /etc/apt/sources.list.d/docker.list

deb http://get.docker.io/ubuntu docker main

Then, update and install the lxc-docker package:

$ sudo apt-get update

$ sudo apt-get install lxc-docker

$ sudo ln -sf /usr/bin/docker /usr/local/bin/docker

$ docker -v

Docker version 1.9.1, build a34a1d5

We can check if our Docker is installed by running the following command as well:

$ docker info

Containers: 12

Images: 77

Server Version: 1.9.1

Storage Driver: aufs

Root Dir: /var/lib/docker/aufs

Backing Filesystem: extfs

Dirs: 101

Dirperm1 Supported: false

Execution Driver: native-0.2

Logging Driver: json-file

Kernel Version: 3.13.0-40-generic

Operating System: Ubuntu 14.04.4 LTS

...

To verify that everything has worked as expected, we can check if Docker downloads the ubuntu image, and then start bash in a container:

$ sudo docker run -i -t ubuntu /bin/bash

...

Status: Downloaded newer image for ubuntu:latest

# exit

exit

k@laptop:/etc/apt/sources.list.d$

We were able to start bash in a container. And we can check if the image for ubuntu is there:

$ sudo docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

ubuntu latest 5506de2b643b 4 weeks ago 199.3 MB

**permission denied for docker.sock**

We may get the following message when we use docker without sudo:

$ docker info

2014/11/21 19:59:03 Get http:///var/run/docker.sock/v1.15/info: dial unix /var/run/docker.sock: permission denied

That's because my user name "k" does not belong "docker" group!

So, let's resolve the issue. First, we need to check if there is docker group:

k@laptop:~$ cat /etc/group

root:x:0:

daemon:x:1:

bin:x:2:

sys:x:3:

adm:x:4:syslog,k

...

docker:x:131:

k@laptop:~$

Ok, docker is one of the names in the group. So, I need to add my username "k" to the docker group:

k@laptop:~$ sudo gpasswd -a k docker

[sudo] password for k:

Adding user k to group docker

Or:

$ sudo usermod -aG docker k

Restart the Docker daemon:

k@laptop:~$ sudo service docker restart

Now, there is no more permission error when we issue docker info without sudo:

k@laptop:~$ docker info

Containers: 1

Images: 7

Storage Driver: aufs

Root Dir: /var/lib/docker/aufs

Dirs: 9

Execution Driver: native-0.2

Kernel Version: 3.13.0-35-generic

Operating System: Ubuntu 14.04.1 LTS

WARNING: No swap limit support

Note: In my case, since I added myself to the group, I had to log out and log back in.

To check the groups I belong:

k@laptop:~$ groups

k adm cdrom sudo dip plugdev lpadmin sambashare docker svn

**Why we get permission denied for docker.sock**

This section is from:  
<https://docs.docker.com/installation/ubuntulinux/#giving-non-root-access>.

**Giving non-root access**

The docker daemon always runs as the **root** user, and since Docker version 0.5.2, the docker daemon binds to a **Unix socket** instead of a **TCP port**. By default that Unix socket is owned by the user root, and so, by default, you can access it with **sudo**.

Starting in version 0.5.3, if you (or your Docker installer) create a Unix group called **docker** and add users to it, then the docker daemon will make the ownership of the Unix socket read/writable by the **docker group** when the daemon starts. The docker daemon must always run as the root user, but if you run the docker client as a user in the docker group then you don't need to add sudo to all the client commands. From Docker 0.9.0 you can use the **-G** flag to specify an alternative group.

Warning: The docker group (or the group specified with the -G flag) is root-equivalent; see [Docker Daemon Attack Surface](https://docs.docker.com/articles/security/" \l "dockersecurity-daemon" \t "_blank) for details.

**Example:**

# Add the docker group if it doesn't already exist.

$ sudo groupadd docker

# Add the connected user "${USER}" to the docker group. # Change the user name to match your preferred user. # You may have to logout and log back in again for # this to take effect.

$ sudo gpasswd -a ${USER} docker

# Restart the Docker daemon. # If you are in Ubuntu 14.04, use docker.io instead of docker

$ sudo service docker restart

**chkconfig or sysv-rc-conf**

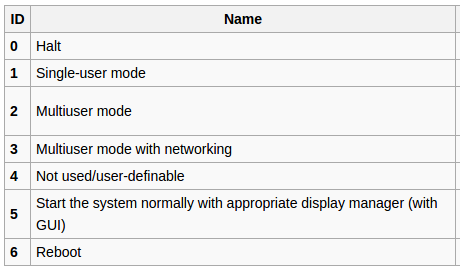
The **chkconfig** utility is a command-line tool that allows us to specify in which **runlevel** to start a selected service, as well as to list all available services along with their current setting.

On CentOS, we can simply turn it on to start it automatically:

$ sudo chkconfig docker on

On Ubuntu > 14, the equivalent to chkconfig is **sysv-rc-conf**. So, we use this instead of chkconfig for Run-level configuration. The **sysv-rc-conf** an easily communicate and managing with **/etc/rc{runlevel}.d/** symlinks.

See the **runlevel** table below:



We may want to install the **sysv-rc-conf** package using apt-get command, and then run **sysv-rc-conf**:

$ sudo apt-get install sysv-rc-conf

$ sudo sysv-rc-conf --list

acpid

anacron

apache2 0:off 1:off 2:on 3:on 4:on 5:on 6:off

...

docker

docker.io

...

vmware-works 0:off 2:on 3:on 4:on 6:off

x11-common S:on

$ sudo sysv-rc-conf docker on

$ sudo sysv-rc-conf --list

...

docker 2:on 3:on 4:on 5:on

docker.io

...