Docker is supported on these Ubuntu operating systems:

- Ubuntu Xenial 16.04 (LTS)
- Ubuntu Wily 15.10
- Ubuntu Trusty 14.04 (LTS)
- Ubuntu Precise 12.04 (LTS)

This page instructs you to install using Docker-managed release packages and installation mechanisms. Using these packages ensures you get the latest release of Docker. If you wish to install using Ubuntu-managed packages, consult your Ubuntu documentation.

> **Note**: Ubuntu Utopic 14.10 and 15.04 exist in Docker's `APT` repository but are no longer officially supported.

Prerequisites

Docker requires a 64-bit installation regardless of your Ubuntu version. Additionally, your kernel must be 3.10 at minimum. The latest 3.10 minor version or a newer maintained version are also acceptable.

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.

To check your current kernel version, open a terminal and use `uname -r` to display your kernel version:

```
$ uname -r
3.11.0-15-generic
```

> **Note**: If you previously installed Docker using `APT`, make sure you update your `APT` sources to the new Docker repository.

Update your apt sources

Docker's `APT` repository contains Docker 1.7.1 and higher. To set `APT` to use packages from the new repository:

1. Log into your machine as a user with `sudo` or `root` privileges.
2. Open a terminal window.
3. Update package information, ensure that APT works with the `https` method, and that CA certificates are installed.

   ```
   $ sudo apt-get update
   $ sudo apt-get install apt-transport-https ca-certificates
   ```

4. Add the new `GPG` key.

   ```
   $ sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80
   ```

```
--recv-keys 58118E89F3A912897C070ADBF76221572C52609D
```

5. Open the `/etc/apt/sources.list.d/docker.list` file in your favorite editor.

   If the file doesn't exist, create it.

6. Remove any existing entries.

7. Add an entry for your Ubuntu operating system.

   The possible entries are:

   - On Ubuntu Precise 12.04 (LTS)

     ```
     deb https://apt.dockerproject.org/repo ubuntu-precise main
     ```

   - On Ubuntu Trusty 14.04 (LTS)

     ```
     deb https://apt.dockerproject.org/repo ubuntu-trusty main
     ```

   - Ubuntu Wily 15.10

     ```
     deb https://apt.dockerproject.org/repo ubuntu-wily main
     ```

   - Ubuntu Xenial 16.04 (LTS)

     ```
     deb https://apt.dockerproject.org/repo ubuntu-xenial main
     ```

   Note: Docker does not provide packages for all architectures. You can find nightly built binaries in https://master.dockerproject.org. To install docker on a multi-architecture system, add an `[arch=...]` clause to the entry. Refer to the Debian Multiarch wiki for details.

8. Save and close the `/etc/apt/sources.list.d/docker.list` file.

9. Update the APT package index.

   ```
   $ sudo apt-get update
   ```

10. Purge the old repo if it exists.

    ```
    $ sudo apt-get purge lxc-docker
    ```

11. Verify that APT is pulling from the right repository.

    ```
    $ apt-cache policy docker-engine
    ```

    From now on when you run `apt-get upgrade`, APT pulls from the new repository.

Prerequisites by Ubuntu Version

- Ubuntu Xenial 16.04 (LTS)
- Ubuntu Wily 15.10
- Ubuntu Trusty 14.04 (LTS)

For Ubuntu Trusty, Wily, and Xenial, it’s recommended to install the `linux-image-extra` kernel package. The `linux-image-extra` package allows you use the `aufs` storage driver.

To install the `linux-image-extra` package for your kernel version:

1. Open a terminal on your Ubuntu host.
2. Update your package manager.

   ```
   $ sudo apt-get update
   ```

3. Install the recommended package.

   ```
   $ sudo apt-get install linux-image-extra-$(uname -r)
   ```

4. Go ahead and install Docker.

If you are installing on Ubuntu 14.04 or 12.04, `apparmor` is required. You can install it using: `apt-get install apparmor`

Ubuntu Precise 12.04 (LTS)

For Ubuntu Precise, Docker requires the 3.13 kernel version. If your kernel version is older than 3.13, you must upgrade it. Refer to this table to see which packages are required for your environment:

linux-image-generic-lts-trusty	Generic Linux kernel image. This kernel has AUFS built in. This is required to run Docker.
linux-headers-generic-lts-trusty	Allows packages such as ZFS and VirtualBox guest additions which depend on them. If you didn’t install the headers for your existing kernel, then you can skip these headers for the”trusty” kernel. If you’re unsure, you should include this package for safety.
xserver-xorg-lts-trusty	Optional in non-graphical environments without Unity/Xorg. **Required** when running Docker on machine with a graphical environment.
libgl1-mesa-glx-lts-trusty	To learn more about the reasons for these packages, read the installation instructions for backported kernels, specifically the LTS Enablement Stack — refer to note 5 under each version.

To upgrade your kernel and install the additional packages, do the following:

1. Open a terminal on your Ubuntu host.
2. Update your package manager.

   ```
   $ sudo apt-get update
   ```

3. Install both the required and optional packages.

   ```
   $ sudo apt-get install linux-image-generic-lts-trusty
   ```

   Depending on your environment, you may install more as described in the preceding table.
4. Reboot your host.

   ```
   $ sudo reboot
   ```

5. After your system reboots, go ahead and install Docker.

Install

Make sure you have installed the prerequisites for your Ubuntu version.

Then, install Docker using the following:

1. Log into your Ubuntu installation as a user with `sudo` privileges.
2. Update your `APT` package index.

   ```
   $ sudo apt-get update
   ```

3. Install Docker.

   ```
   $ sudo apt-get install docker-engine
   ```

4. Start the `docker` daemon.

   ```
   $ sudo service docker start
   ```

5. Verify `docker` is installed correctly.

   ```
   $ sudo docker run hello-world
   ```

   This command downloads a test image and runs it in a container. When the container runs, it prints an informational message. Then, it exits.

Optional configurations

This section contains optional procedures for configuring your Ubuntu to work better with Docker.

- Create a docker group
- Adjust memory and swap accounting
- Enable UFW forwarding
- Configure a DNS server for use by Docker
- Configure Docker to start on boot

Create a Docker group

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 other users can access it with `sudo`. For this reason, `docker` daemon always runs as the `root` user.

To avoid having to use `sudo` when you use the `docker` command, create a Unix group called `docker` and add users to it. When the `docker` daemon starts, it makes the ownership of the Unix socket read/writable by the `docker` group.

> **Warning**: The `docker` group is equivalent to the `root` user; For details on how this impacts security in your system, see *Docker Daemon Attack Surface* for details.

To create the `docker` group and add your user:

1. Log into Ubuntu as a user with `sudo` privileges.

   This procedure assumes you log in as the `ubuntu` user.

2. Create the `docker` group.

   ```
   $ sudo groupadd docker
   ```

3. Add your user to `docker` group.

   ```
   $ sudo usermod -aG docker ubuntu
   ```

4. Log out and log back in.

   This ensures your user is running with the correct permissions.

5. Verify your work by running `docker` without `sudo`.

   ```
   $ docker run hello-world
   ```

   If this fails with a message similar to this:

   ```
   Cannot connect to the Docker daemon. Is 'docker daemon' running on this
   ```

```
host?
```

Check that the DOCKER_HOST environment variable is not set for your shell. If it is, unset it.

Adjust memory and swap accounting

When users run Docker, they may see these messages when working with an image:

```
WARNING: Your kernel does not support cgroup swap limit. WARNING: Your
kernel does not support swap limit capabilities. Limitation discarded.
```

To prevent these messages, enable memory and swap accounting on your system. Enabling memory and swap accounting does induce both a memory overhead and a performance degradation even when Docker is not in use. The memory overhead is about 1% of the total available memory. The performance degradation is roughly 10%.

To enable memory and swap on system using GNU GRUB (GNU GRand Unified Bootloader), do the following:

1. Log into Ubuntu as a user with sudo privileges.
2. Edit the /etc/default/grub file.
3. Set the GRUB_CMDLINE_LINUX value as follows:

   ```
   GRUB_CMDLINE_LINUX="cgroup_enable=memory swapaccount=1"
   ```

4. Save and close the file.
5. Update GRUB.

   ```
   $ sudo update-grub
   ```

6. Reboot your system.

Enable UFW forwarding

If you use UFW (Uncomplicated Firewall) on the same host as you run Docker, you’ll need to do additional configuration. Docker uses a bridge to manage container networking. By default, UFW drops all forwarding traffic. As a result, for Docker to run when UFW is enabled, you must set UFW’s forwarding policy appropriately.

Also, UFW’s default set of rules denies all incoming traffic. If you want to reach your containers from another host allow incoming connections on the Docker port. The Docker port defaults to 2376 if TLS is enabled or 2375 when it is not. If TLS is not enabled, communication is unencrypted. By default, Docker runs without TLS enabled.

To configure UFW and allow incoming connections on the Docker port:

1. Log into Ubuntu as a user with sudo privileges.
2. Verify that UFW is installed and enabled.

   ```
   $ sudo ufw status
   ```

3. Open the `/etc/default/ufw` file for editing.

   ```
   $ sudo nano /etc/default/ufw
   ```

4. Set the `DEFAULT_FORWARD_POLICY` policy to:

   ```
   DEFAULT_FORWARD_POLICY="ACCEPT"
   ```

5. Save and close the file.
6. Reload UFW to use the new setting.

   ```
   $ sudo ufw reload
   ```

7. Allow incoming connections on the Docker port.

   ```
   $ sudo ufw allow 2375/tcp
   ```

Configure a DNS server for use by Docker

Systems that run Ubuntu or an Ubuntu derivative on the desktop typically use `127.0.0.1` as the default `nameserver` in `/etc/resolv.conf` file. The NetworkManager also sets up `dnsmasq` to use the real DNS servers of the connection and sets up `nameserver 127.0.0.1` in /`etc/resolv.conf`.

When starting containers on desktop machines with these configurations, Docker users see this warning:

```
WARNING: Local (127.0.0.1) DNS resolver found in resolv.conf and containers
can't use it. Using default external servers : [8.8.8.8 8.8.4.4]
```

The warning occurs because Docker containers can't use the local DNS nameserver. Instead, Docker defaults to using an external nameserver.

To avoid this warning, you can specify a DNS server for use by Docker containers. Or, you can disable `dnsmasq` in NetworkManager. Though, disabling `dnsmasq` might make DNS resolution slower on some networks.

The instructions below describe how to configure the Docker daemon running on Ubuntu 14.10 or below. Ubuntu 15.04 and above use `systemd` as the boot and service manager. Refer to control and configure Docker with systemd to configure a daemon controlled by `systemd`.

To specify a DNS server for use by Docker:

1. Log into Ubuntu as a user with `sudo` privileges.
2. Open the `/etc/default/docker` file for editing.

   ```
   $ sudo nano /etc/default/docker
   ```

3. Add a setting for Docker.

   ```
   DOCKER_OPTS="--dns 8.8.8.8"
   ```

   Replace `8.8.8.8` with a local DNS server such as `192.168.1.1`. You can also specify multiple DNS servers. Separated them with spaces, for example:

```
--dns 8.8.8.8 --dns 192.168.1.1
```

> **Warning**: If you’re doing this on a laptop which connects to various networks, make sure to choose a public DNS server.

4. Save and close the file.
5. Restart the Docker daemon.

   ```
   $ sudo service docker restart
   ```

Or, as an alternative to the previous procedure, disable `dnsmasq` in NetworkManager (this might slow your network).

1. Open the `/etc/NetworkManager/NetworkManager.conf` file for editing.

   ```
   $ sudo nano /etc/NetworkManager/NetworkManager.conf
   ```

2. Comment out the `dns=dnsmasq` line:

   ```
   dns=dnsmasq
   ```

3. Save and close the file.
4. Restart both the NetworkManager and Docker.

   ```
   $ sudo restart network-manager
   $ sudo restart docker
   ```

Configure Docker to start on boot

Ubuntu uses `systemd` as its boot and service manager `15.04` onwards and `upstart` for versions `14.10` and below.

For `15.04` and up, to configure the `docker` daemon to start on boot, run

```
$ sudo systemctl enable docker
```

For `14.10` and below the above installation method automatically configures `upstart` to start the docker daemon on boot

Upgrade Docker

To install the latest version of Docker with `apt-get`:

```
$ sudo apt-get upgrade docker-engine
```

Uninstallation

To uninstall the Docker package:

```
$ sudo apt-get purge docker-engine
```

To uninstall the Docker package and dependencies that are no longer needed:

```
$ sudo apt-get autoremove --purge docker-engine
```

The above commands will not remove images, containers, volumes, or user created configuration files on your host. If you wish to delete all images, containers, and volumes run the following command:

```
$ rm -rf /var/lib/docker
```

You must delete the user created configuration files manually.