Install Docker Engine on Ubuntu

*Estimated reading time: 11 minutes*

To get started with Docker Engine on Ubuntu, make sure you [meet the prerequisites](https://docs.docker.com/engine/install/ubuntu/#prerequisites), then [install Docker](https://docs.docker.com/engine/install/ubuntu/#installation-methods).

Prerequisites

OS requirements

To install Docker Engine, you need the 64-bit version of one of these Ubuntu versions:

* Ubuntu Focal 20.04 (LTS)
* Ubuntu Bionic 18.04 (LTS)
* Ubuntu Xenial 16.04 (LTS)

Docker Engine is supported on x86\_64 (or amd64), armhf, and arm64 architectures.

Uninstall old versions

Older versions of Docker were called docker, docker.io, or docker-engine. If these are installed, uninstall them:

$ sudo apt-get remove docker docker-engine docker.io containerd runc

It’s OK if apt-get reports that none of these packages are installed.

The contents of /var/lib/docker/, including images, containers, volumes, and networks, are preserved. If you do not need to save your existing data, and want to start with a clean installation, refer to the [uninstall Docker Engine](https://docs.docker.com/engine/install/ubuntu/#uninstall-docker-engine) section at the bottom of this page.

Supported storage drivers

Docker Engine on Ubuntu supports overlay2, aufs and btrfs storage drivers.

Docker Engine uses the overlay2 storage driver by default. If you need to use aufs instead, you need to configure it manually. See [use the AUFS storage driver](https://docs.docker.com/storage/storagedriver/aufs-driver/)

Installation methods

You can install Docker Engine in different ways, depending on your needs:

* Most users [set up Docker’s repositories](https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository) and install from them, for ease of installation and upgrade tasks. This is the recommended approach.
* Some users download the DEB package and [install it manually](https://docs.docker.com/engine/install/ubuntu/#install-from-a-package) and manage upgrades completely manually. This is useful in situations such as installing Docker on air-gapped systems with no access to the internet.
* In testing and development environments, some users choose to use automated [convenience scripts](https://docs.docker.com/engine/install/ubuntu/#install-using-the-convenience-script) to install Docker.

Install using the repository

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

SET UP THE REPOSITORY

1. Update the apt package index and install packages to allow apt to use a repository over HTTPS:
2. $ sudo apt-get update
3. $ sudo apt-get install \
4. apt-transport-https \
5. ca-certificates \
6. curl \
7. gnupg-agent \
8. software-properties-common
9. Add Docker’s official GPG key:
10. $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

Verify that you now have the key with the fingerprint 9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88, by searching for the last 8 characters of the fingerprint.

$ sudo apt-key fingerprint 0EBFCD88

pub rsa4096 2017-02-22 [SCEA]

9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88

uid [ unknown] Docker Release (CE deb) <docker@docker.com>

sub rsa4096 2017-02-22 [S]

1. Use the following command to set up the **stable** repository. To add the **nightly** or **test** repository, add the word nightly or test (or both) after the word stable in the commands below. [Learn about **nightly** and **test** channels](https://docs.docker.com/engine/install/).

**Note**: The lsb\_release -cs sub-command below returns the name of your Ubuntu distribution, such as xenial. Sometimes, in a distribution like Linux Mint, you might need to change $(lsb\_release -cs) to your parent Ubuntu distribution. For example, if you are using Linux Mint Tessa, you could use bionic. Docker does not offer any guarantees on untested and unsupported Ubuntu distributions.

* + x86\_64 / amd64
  + armhf
  + arm64

$ sudo add-apt-repository \

"deb [arch=amd64] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) \

stable"

INSTALL DOCKER ENGINE

1. Update the apt package index, and install the *latest version* of Docker Engine and containerd, or go to the next step to install a specific version:
2. $ sudo apt-get update
3. $ sudo apt-get install docker-ce docker-ce-cli containerd.io

**Got multiple Docker repositories?**

If you have multiple Docker repositories enabled, installing or updating without specifying a version in the apt-get install or apt-get update command always installs the highest possible version, which may not be appropriate for your stability needs.

1. To install a *specific version* of Docker Engine, list the available versions in the repo, then select and install:

a. List the versions available in your repo:

$ apt-cache madison docker-ce

docker-ce | 5:18.09.1~3-0~ubuntu-xenial | https://download.docker.com/linux/ubuntu xenial/stable amd64 Packages

docker-ce | 5:18.09.0~3-0~ubuntu-xenial | https://download.docker.com/linux/ubuntu xenial/stable amd64 Packages

docker-ce | 18.06.1~ce~3-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable amd64 Packages

docker-ce | 18.06.0~ce~3-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable amd64 Packages

...

b. Install a specific version using the version string from the second column, for example, 5:18.09.1~3-0~ubuntu-xenial.

$ sudo apt-get install docker-ce=<VERSION\_STRING> docker-ce-cli=<VERSION\_STRING> containerd.io

1. Verify that Docker Engine is installed correctly by running the hello-world image.
2. $ 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 and exits.

Docker Engine is installed and running. The docker group is created but no users are added to it. You need to use sudo to run Docker commands. Continue to [Linux postinstall](https://docs.docker.com/engine/install/linux-postinstall/) to allow non-privileged users to run Docker commands and for other optional configuration steps.

UPGRADE DOCKER ENGINE

To upgrade Docker Engine, first run sudo apt-get update, then follow the [installation instructions](https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository), choosing the new version you want to install.

Install from a package

If you cannot use Docker’s repository to install Docker Engine, you can download the .deb file for your release and install it manually. You need to download a new file each time you want to upgrade Docker.

1. Go to <https://download.docker.com/linux/ubuntu/dists/>, choose your Ubuntu version, then browse to pool/stable/, choose amd64, armhf, or arm64, and download the .deb file for the Docker Engine version you want to install.

**Note**: To install a **nightly** or **test** (pre-release) package, change the word stable in the above URL to nightly or test. [Learn about **nightly** and **test** channels](https://docs.docker.com/engine/install/).

1. Install Docker Engine, changing the path below to the path where you downloaded the Docker package.
2. $ sudo dpkg -i /path/to/package.deb

The Docker daemon starts automatically.

1. Verify that Docker Engine is installed correctly by running the hello-world image.
2. $ 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 and exits.

Docker Engine is installed and running. The docker group is created but no users are added to it. You need to use sudo to run Docker commands. Continue to [Post-installation steps for Linux](https://docs.docker.com/engine/install/linux-postinstall/) to allow non-privileged users to run Docker commands and for other optional configuration steps.

UPGRADE DOCKER ENGINE

To upgrade Docker Engine, download the newer package file and repeat the [installation procedure](https://docs.docker.com/engine/install/ubuntu/#install-from-a-package), pointing to the new file.

Install using the convenience script

Docker provides convenience scripts at [get.docker.com](https://get.docker.com/) and [test.docker.com](https://test.docker.com/) for installing edge and testing versions of Docker Engine - Community into development environments quickly and non-interactively. The source code for the scripts is in the [docker-install repository](https://github.com/docker/docker-install). **Using these scripts is not recommended for production environments**, and you should understand the potential risks before you use them:

* The scripts require root or sudo privileges to run. Therefore, you should carefully examine and audit the scripts before running them.
* The scripts attempt to detect your Linux distribution and version and configure your package management system for you. In addition, the scripts do not allow you to customize any installation parameters. This may lead to an unsupported configuration, either from Docker’s point of view or from your own organization’s guidelines and standards.
* The scripts install all dependencies and recommendations of the package manager without asking for confirmation. This may install a large number of packages, depending on the current configuration of your host machine.
* The script does not provide options to specify which version of Docker to install, and installs the latest version that is released in the “edge” channel.
* Do not use the convenience script if Docker has already been installed on the host machine using another mechanism.

This example uses the script at [get.docker.com](https://get.docker.com/) to install the latest release of Docker Engine - Community on Linux. To install the latest testing version, use [test.docker.com](https://test.docker.com/) instead. In each of the commands below, replace each occurrence of get with test.

**Warning:**

Always examine scripts downloaded from the internet before running them locally.

$ curl -fsSL https://get.docker.com -o get-docker.sh

$ sudo sh get-docker.sh

<output truncated>

If you would like to use Docker as a non-root user, you should now consider adding your user to the “docker” group with something like:

sudo usermod -aG docker your-user

Remember to log out and back in for this to take effect!

**Warning:**

Adding a user to the “docker” group grants them the ability to run containers which can be used to obtain root privileges on the Docker host. Refer to [Docker Daemon Attack Surface](https://docs.docker.com/engine/security/#docker-daemon-attack-surface) for more information.

Docker Engine - Community is installed. It starts automatically on DEB-based distributions. On RPM-based distributions, you need to start it manually using the appropriate systemctl or service command. As the message indicates, non-root users can’t run Docker commands by default.

**Note:**

To install Docker without root privileges, see [Run the Docker daemon as a non-root user (Rootless mode)](https://docs.docker.com/engine/security/rootless/).

Rootless mode is currently available as an experimental feature.

UPGRADE DOCKER AFTER USING THE CONVENIENCE SCRIPT

If you installed Docker using the convenience script, you should upgrade Docker using your package manager directly. There is no advantage to re-running the convenience script, and it can cause issues if it attempts to re-add repositories which have already been added to the host machine.

Uninstall Docker Engine

1. Uninstall the Docker Engine, CLI, and Containerd packages:
2. $ sudo apt-get purge docker-ce docker-ce-cli containerd.io
3. Images, containers, volumes, or customized configuration files on your host are not automatically removed. To delete all images, containers, and volumes:
4. $ sudo rm -rf /var/lib/docker

You must delete any edited configuration files manually.

Next steps

* Continue to [Post-installation steps for Linux](https://docs.docker.com/engine/install/linux-postinstall/).
* Review the topics in [Develop with Docker](https://docs.docker.com/develop/) to learn how to build new applications using Docker.

[requirements](https://docs.docker.com/search/?q=requirements), [apt](https://docs.docker.com/search/?q=apt), [installation](https://docs.docker.com/search/?q=installation), [ubuntu](https://docs.docker.com/search/?q=ubuntu), [install](https://docs.docker.com/search/?q=install), [uninstall](https://docs.docker.com/search/?q=uninstall), [upgrade](https://docs.docker.com/search/?q=upgrade), [update](https://docs.docker.com/search/?q=update)

This section contains optional procedures for configuring Linux hosts to work better with Docker.

Manage Docker as a non-root user

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 only access it using sudo. The Docker daemon always runs as the root user.

If you don’t want to preface the docker command with sudo, create a Unix group called docker and add users to it. When the Docker daemon starts, it creates a Unix socket accessible by members of the docker group.

**Warning**

The docker group grants privileges equivalent to the root user. For details on how this impacts security in your system, see [*Docker Daemon Attack Surface*](https://docs.docker.com/engine/security/#docker-daemon-attack-surface).

**Note:**

To run Docker without root privileges, see [Run the Docker daemon as a non-root user (Rootless mode)](https://docs.docker.com/engine/security/rootless/).

Rootless mode is currently available as an experimental feature.

To create the docker group and add your user:

1. Create the docker group.
2. $ sudo groupadd docker
3. Add your user to the docker group.
4. $ sudo usermod -aG docker $USER
5. Log out and log back in so that your group membership is re-evaluated.

If testing on a virtual machine, it may be necessary to restart the virtual machine for changes to take effect.

On a desktop Linux environment such as X Windows, log out of your session completely and then log back in.

On Linux, you can also run the following command to activate the changes to groups:

$ newgrp docker

1. Verify that you can run docker commands without sudo.
2. $ 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 and exits.

If you initially ran Docker CLI commands using sudo before adding your user to the docker group, you may see the following error, which indicates that your ~/.docker/ directory was created with incorrect permissions due to the sudo commands.

WARNING: Error loading config file: /home/user/.docker/config.json -

stat /home/user/.docker/config.json: permission denied

To fix this problem, either remove the ~/.docker/ directory (it is recreated automatically, but any custom settings are lost), or change its ownership and permissions using the following commands:

$ sudo chown "$USER":"$USER" /home/"$USER"/.docker -R

$ sudo chmod g+rwx "$HOME/.docker" -R

Configure Docker to start on boot

Most current Linux distributions (RHEL, CentOS, Fedora, Ubuntu 16.04 and higher) use [systemd](https://docs.docker.com/engine/install/linux-postinstall/#systemd) to manage which services start when the system boots. Ubuntu 14.10 and below use [upstart](https://docs.docker.com/engine/install/linux-postinstall/#upstart).

systemd

$ sudo systemctl enable docker

To disable this behavior, use disable instead.

$ sudo systemctl disable docker

If you need to add an HTTP Proxy, set a different directory or partition for the Docker runtime files, or make other customizations, see [customize your systemd Docker daemon options](https://docs.docker.com/config/daemon/systemd/).

upstart

Docker is automatically configured to start on boot using upstart. To disable this behavior, use the following command:

$ echo manual | sudo tee /etc/init/docker.override

chkconfig

$ sudo chkconfig docker on

Use a different storage engine

For information about the different storage engines, see [Storage drivers](https://docs.docker.com/storage/storagedriver/). The default storage engine and the list of supported storage engines depend on your host’s Linux distribution and available kernel drivers.

Configure default logging driver

Docker provides the [capability](https://docs.docker.com/config/containers/logging/) to collect and view log data from all containers running on a host via a series of logging drivers. The default logging driver, json-file, writes log data to JSON-formatted files on the host filesystem. Over time, these log files expand in size, leading to potential exhaustion of disk resources. To alleviate such issues, either configure an alternative logging driver such as Splunk or Syslog, or [set up log rotation](https://docs.docker.com/config/containers/logging/configure/#configure-the-default-logging-driver) for the default driver. If you configure an alternative logging driver, see [Use docker logs to read container logs for remote logging drivers](https://docs.docker.com/config/containers/logging/dual-logging/).

Configure where the Docker daemon listens for connections

By default, the Docker daemon listens for connections on a UNIX socket to accept requests from local clients. It is possible to allow Docker to accept requests from remote hosts by configuring it to listen on an IP address and port as well as the UNIX socket. For more detailed information on this configuration option take a look at “Bind Docker to another host/port or a unix socket” section of the [Docker CLI Reference](https://docs.docker.com/engine/reference/commandline/dockerd/) article.

**Secure your connection**

Before configuring Docker to accept connections from remote hosts it is critically important that you understand the security implications of opening docker to the network. If steps are not taken to secure the connection, it is possible for remote non-root users to gain root access on the host. For more information on how to use TLS certificates to secure this connection, check this article on [how to protect the Docker daemon socket](https://docs.docker.com/engine/security/https/).

Configuring Docker to accept remote connections can be done with the docker.service systemd unit file for Linux distributions using systemd, such as recent versions of RedHat, CentOS, Ubuntu and SLES, or with the daemon.json file which is recommended for Linux distributions that do not use systemd.

**systemd vs daemon.json**

Configuring Docker to listen for connections using both the systemd unit file and the daemon.json file causes a conflict that prevents Docker from starting.

Configuring remote access with systemd unit file

1. Use the command sudo systemctl edit docker.service to open an override file for docker.service in a text editor.
2. Add or modify the following lines, substituting your own values.
3. [Service]
4. ExecStart=
5. ExecStart=/usr/bin/dockerd -H fd:// -H tcp://127.0.0.1:2375
6. Save the file.
7. Reload the systemctl configuration.
8. $ sudo systemctl daemon-reload
9. Restart Docker.
10. $ sudo systemctl restart docker.service
11. Check to see whether the change was honored by reviewing the output of netstat to confirm dockerd is listening on the configured port.
12. $ sudo netstat -lntp | grep dockerd
13. tcp 0 0 127.0.0.1:2375 0.0.0.0:\* LISTEN 3758/dockerd

Configuring remote access with daemon.json

1. Set the hosts array in the /etc/docker/daemon.json to connect to the UNIX socket and an IP address, as follows:
2. {
3. "hosts": ["unix:///var/run/docker.sock", "tcp://127.0.0.1:2375"]
4. }
5. Restart Docker.
6. Check to see whether the change was honored by reviewing the output of netstat to confirm dockerd is listening on the configured port.
7. $ sudo netstat -lntp | grep dockerd
8. tcp 0 0 127.0.0.1:2375 0.0.0.0:\* LISTEN 3758/dockerd

Enable IPv6 on the Docker daemon

To enable IPv6 on the Docker daemon, see [Enable IPv6 support](https://docs.docker.com/config/daemon/ipv6/).

Troubleshooting

Kernel compatibility

Docker cannot run correctly if your kernel is older than version 3.10 or if it is missing some modules. To check kernel compatibility, you can download and run the [check-config.sh](https://raw.githubusercontent.com/docker/docker/master/contrib/check-config.sh) script.

$ curl https://raw.githubusercontent.com/docker/docker/master/contrib/check-config.sh > check-config.sh

$ bash ./check-config.sh

The script only works on Linux, not macOS.

Cannot connect to the Docker daemon

If you see an error such as the following, your Docker client may be configured to connect to a Docker daemon on a different host, and that host may not be reachable.

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

To see which host your client is configured to connect to, check the value of the DOCKER\_HOST variable in your environment.

$ env | grep DOCKER\_HOST

If this command returns a value, the Docker client is set to connect to a Docker daemon running on that host. If it is unset, the Docker client is set to connect to the Docker daemon running on the local host. If it is set in error, use the following command to unset it:

$ unset DOCKER\_HOST

You may need to edit your environment in files such as ~/.bashrc or ~/.profile to prevent the DOCKER\_HOST variable from being set erroneously.

If DOCKER\_HOST is set as intended, verify that the Docker daemon is running on the remote host and that a firewall or network outage is not preventing you from connecting.

IP forwarding problems

If you manually configure your network using systemd-network with systemd version 219 or higher, Docker containers may not be able to access your network. Beginning with systemd version 220, the forwarding setting for a given network (net.ipv4.conf.<interface>.forwarding) defaults to *off*. This setting prevents IP forwarding. It also conflicts with Docker’s behavior of enabling the net.ipv4.conf.all.forwarding setting within containers.

To work around this on RHEL, CentOS, or Fedora, edit the <interface>.network file in /usr/lib/systemd/network/ on your Docker host (ex: /usr/lib/systemd/network/80-container-host0.network) and add the following block within the [Network] section.

[Network]

...

IPForward=kernel

# OR

IPForward=true

...

This configuration allows IP forwarding from the container as expected.

DNS resolver found in resolv.conf and containers can't use it

Linux systems which use a GUI often have a network manager running, which uses a dnsmasq instance running on a loopback address such as 127.0.0.1 or 127.0.1.1 to cache DNS requests, and adds this entry to /etc/resolv.conf. The dnsmasq service speeds up DNS look-ups and also provides DHCP services. This configuration does not work within a Docker container which has its own network namespace, because the Docker container resolves loopback addresses such as 127.0.0.1 to **itself**, and it is very unlikely to be running a DNS server on its own loopback address.

If Docker detects that no DNS server referenced in /etc/resolv.conf is a fully functional DNS server, the following warning occurs and Docker uses the public DNS servers provided by Google at 8.8.8.8 and 8.8.4.4 for DNS resolution.

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]

If you see this warning, first check to see if you use dnsmasq:

$ ps aux |grep dnsmasq

If your container needs to resolve hosts which are internal to your network, the public nameservers are not adequate. You have two choices:

* You can specify a DNS server for Docker to use, **or**
* You can disable dnsmasq in NetworkManager. If you do this, NetworkManager adds your true DNS nameserver to /etc/resolv.conf, but you lose the possible benefits of dnsmasq.

**You only need to use one of these methods.**

Specify DNS servers for Docker

The default location of the configuration file is /etc/docker/daemon.json. You can change the location of the configuration file using the --config-file daemon flag. The documentation below assumes the configuration file is located at /etc/docker/daemon.json.

1. Create or edit the Docker daemon configuration file, which defaults to /etc/docker/daemon.json file, which controls the Docker daemon configuration.
2. $ sudo nano /etc/docker/daemon.json
3. Add a dns key with one or more IP addresses as values. If the file has existing contents, you only need to add or edit the dns line.
4. {
5. "dns": ["8.8.8.8", "8.8.4.4"]
6. }

If your internal DNS server cannot resolve public IP addresses, include at least one DNS server which can, so that you can connect to Docker Hub and so that your containers can resolve internet domain names.

Save and close the file.

1. Restart the Docker daemon.
2. $ sudo service docker restart
3. Verify that Docker can resolve external IP addresses by trying to pull an image:
4. $ docker pull hello-world
5. If necessary, verify that Docker containers can resolve an internal hostname by pinging it.
6. $ docker run --rm -it alpine ping -c4 <my\_internal\_host>
7. PING google.com (192.168.1.2): 56 data bytes
8. 64 bytes from 192.168.1.2: seq=0 ttl=41 time=7.597 ms
9. 64 bytes from 192.168.1.2: seq=1 ttl=41 time=7.635 ms
10. 64 bytes from 192.168.1.2: seq=2 ttl=41 time=7.660 ms
11. 64 bytes from 192.168.1.2: seq=3 ttl=41 time=7.677 ms

DISABLE DNSMASQ

Ubuntu

If you prefer not to change the Docker daemon’s configuration to use a specific IP address, follow these instructions to disable dnsmasq in NetworkManager.

1. Edit the /etc/NetworkManager/NetworkManager.conf file.
2. Comment out the dns=dnsmasq line by adding a # character to the beginning of the line.
3. # dns=dnsmasq

Save and close the file.

1. Restart both NetworkManager and Docker. As an alternative, you can reboot your system.
2. $ sudo restart network-manager
3. $ sudo restart docker

RHEL, CentOS, or Fedora

To disable dnsmasq on RHEL, CentOS, or Fedora:

1. Disable the dnsmasq service:
2. $ sudo service dnsmasq stop
3. $ sudo systemctl disable dnsmasq
4. Configure the DNS servers manually using the [Red Hat documentation](https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/s1-networkscripts-interfaces.html).

Allow access to the remote API through a firewall

If you run a firewall on the same host as you run Docker and you want to access the Docker Remote API from another host and remote access is enabled, you need to configure your firewall to allow incoming connections on the Docker port, which defaults to 2376 if TLS encrypted transport is enabled or 2375 otherwise.

Two common firewall daemons are [UFW (Uncomplicated Firewall)](https://help.ubuntu.com/community/UFW) (often used for Ubuntu systems) and [firewalld](https://firewalld.org/) (often used for RPM-based systems). Consult the documentation for your OS and firewall, but the following information might help you get started. These options are fairly permissive and you may want to use a different configuration that locks your system down more.

* **UFW**: Set DEFAULT\_FORWARD\_POLICY="ACCEPT" in your configuration.
* **firewalld**: Add rules similar to the following to your policy (one for incoming requests and one for outgoing requests). Be sure the interface names and chain names are correct.
* <direct>
* [ <rule ipv="ipv6" table="filter" chain="FORWARD\_direct" priority="0"> -i zt0 -j ACCEPT </rule> ]
* [ <rule ipv="ipv6" table="filter" chain="FORWARD\_direct" priority="0"> -o zt0 -j ACCEPT </rule> ]
* </direct>

Your kernel does not support cgroup swap limit capabilities

On Ubuntu or Debian hosts, You may see messages similar to the following when working with an image.

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

This warning does not occur on RPM-based systems, which enable these capabilities by default.

If you don’t need these capabilities, you can ignore the warning. You can enable these capabilities on Ubuntu or Debian by following these instructions. Memory and swap accounting incur an overhead of about 1% of the total available memory and a 10% overall performance degradation, even if Docker is not running.

1. Log into the Ubuntu or Debian host as a user with sudo privileges.
2. Edit the /etc/default/grub file. Add or edit the GRUB\_CMDLINE\_LINUX line to add the following two key-value pairs:
3. GRUB\_CMDLINE\_LINUX="cgroup\_enable=memory swapaccount=1"

Save and close the file.

1. Update GRUB.
2. $ sudo update-grub

If your GRUB configuration file has incorrect syntax, an error occurs. In this case, repeat steps 2 and 3.

The changes take effect when the system is rebooted.

Next steps

* Take a look at the [Get started](https://docs.docker.com/get-started/) training modules to learn how to build an image and run it as a containerized application.
* Review the topics in [Develop with Docker](https://docs.docker.com/develop/) to learn how to build new applications using Docker.