Security Advisory: Critical OpenSSL Vulnerability - learn how (https://www.docker.com/blog/security-advisory-critical-openssl-vulnerability/) to tell if your repository is vulnerable. ✕

Post-installation steps for Linux

Estimated reading time: 7 minutes

These optional post-installation procedures shows you how to configure your Linux host machine to work better with Docker.

Manage Docker as a non-root user

The Docker daemon binds to a Unix socket, not a TCP port. By default it's the `root` user that owns the Unix socket, 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. On some Linux distributions, the system automatically creates this group when installing Docker Engine using a package manager. In that case, there is no need for you to manually create the group.

Give feedback

Warning

The `docker` group grants root-level privileges to the user. For details on how this impacts security in your system, see Docker Daemon Attack Surface (/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) (/engine/security/rootless/).

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

1. Create the `docker` group.

```
$ sudo groupadd docker
```

2. Add your user to the `docker` group.

```
$ sudo usermod -aG docker $USER
```

3. Log out and log back in so that your group membership is re-evaluated.

> If you're running Linux in a virtual machine, it may be necessary to restart the virtual machine for changes to take effect.

You can also run the following command to activate the changes to groups:

```
$ newgrp docker
```

4. Verify that you can run `docker` commands without `sudo`.

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

Give feedback

This command downloads a test image and runs it in a container. When the container runs, it prints a 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:

```
WARNING: Error loading config file: /home/user/.docker/config.json -
stat /home/user/.docker/config.json: permission denied
```

This error indicates that the permission settings for the `~/.docker/` directory are incorrect, due to having used the `sudo` command earlier.

To fix this problem, either remove the `~/.docker/` directory (it's 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 with systemd

Many modern Linux distributions use systemd (/config/daemon/systemd/) to manage which services start when the system boots. On Debian and Ubuntu, the Docker service starts on boot by default. To automatically start Docker and containerd on boot for other Linux distributions using systemd, run the following commands:

```
$ sudo systemctl enable docker.service
$ sudo systemctl enable containerd.service
```

To stop this behavior, use `disable` instead.

```
$ sudo systemctl disable docker.service
$ sudo systemctl disable containerd.service
```

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 (/config/daemon/systemd/).

Give feedback

Configure default logging driver

Docker provides logging drivers (/config/containers/logging/) for collecting and viewing log data from all containers running on a host. 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 avoid issues with overusing disk for log data, consider one of the following options:

- Configure the `json-file` logging driver to turn on log rotation (/config/containers/logging/json-file/)
- Use an alternative logging driver (/config/containers/logging/configure/#configure-the-default-logging-driver) such as the “local” logging driver (/config/containers/logging/local/) that performs log rotation by default
- Use a logging driver that sends logs to a remote logging aggregator.

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’s 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, refer to the dockerd CLI reference (/engine/reference/commandline/dockerd/#bind-docker-to-another-hostport-or-a-unix-socket).

Secure your connection

Before configuring Docker to accept connections from remote hosts it's critically important that you understand the security implications of opening Docker to the network. If steps aren't taken to secure the connection, it's 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 Protect the Docker daemon socket (/engine/security/protect-access/).

You can configure Docker to accept remote connections. This can be done using the `docker.service` systemd unit file for Linux distributions using systemd. Or you can use the `daemon.json` file, if your distribution doesn't 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.

```
[Service]
ExecStart=
ExecStart=/usr/bin/dockerd -H fd:// -H tcp://127.0.0.1:2375
```

3. Save the file.
4. Reload the `systemctl` configuration.

```
 $ sudo systemctl daemon-reload
```

5. Restart Docker.

```
$ sudo systemctl restart docker.service
```

6. Verify that the change has gone through.

```
$ sudo netstat -lntp | grep dockerd
tcp        0      0 127.0.0.1:2375          0.0.0.0:*               LISTEN
```

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:

```
{
  "hosts": ["unix:///var/run/docker.sock", "tcp://127.0.0.1:2375"]
}
```

2. Restart Docker.

3. Verify that the change has gone through.

```
$ sudo netstat -lntp | grep dockerd
tcp        0      0 127.0.0.1:2375          0.0.0.0:*               LISTEN
```

Give feedback

Enable IPv6 on the Docker daemon

To enable IPv6 on the Docker daemon, see Enable IPv6 support (/config/daemon/ipv6/).

Next steps

- Take a look at the Get started (/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 (/develop/) to learn how to build new applications using Docker.

Docker (/search/?q=Docker), Docker documentation (/search/?q=Docker documentation), requirements (/search/?q=requirements), apt (/search/?q=apt), installation (/search/?q=installation), ubuntu (/search/?q=ubuntu), install (/search/?q=install), uninstall (/search/?q=uninstall), upgrade (/search/?q=upgrade), update (/search/?q=update)