FAQs for Docker Desktop for Linux

- Why does Docker Desktop for Linux run a VM?
- How do I enable file sharing?
- How do I use Docker SDKs with Docker Desktop for Linux?
- Where does Docker Desktop store Linux containers?
  - Where is the disk image file?
    - What if the file is too large?
    - How do I move the file to a bigger drive?
    - How do I delete unnecessary containers and images?
    - How do I reduce the maximum size of the file?

Why does Docker Desktop for Linux run a VM?

Docker Desktop for Linux runs a Virtual Machine (VM) for the following reasons:

1. To ensure that Docker Desktop provides a consistent experience across platforms.
   During research, the most frequently cited reason for users wanting Docker Desktop for Linux was to ensure a consistent Docker Desktop experience with feature parity across all major operating systems. Utilizing a VM ensures that the Docker Desktop experience for Linux users will closely match that of Windows and macOS.
2. To make use of new kernel features.
   Because Docker controls the kernel and the OS inside the VM, Docker can roll these out to all users immediately, even to users who are intentionally sticking on an LTS version of their machine OS.
3. To enhance security.
   Container image vulnerabilities pose a security risk for the host environment. There is a large number of unofficial images that are not guaranteed to be verified for known

vulnerabilities. Malicious users can push images to public registries and use different methods to trick users into pulling and running them. The VM approach mitigates this threat as any malware that gains root privileges is restricted to the VM environment without access to the host.
Why not run rootless Docker? Although this has the benefit of superficially limiting access to the root user so everything looks safer in "top", it allows unprivileged users to gain `CAP_SYS_ADMIN` in their own user namespace and access kernel APIs which are not expecting to be used by unprivileged users, resulting in vulnerabilities.

4. To provide the benefits of feature parity and enhanced security, with minimal impact on performance.
   The VM utilized by Docker Desktop for Linux uses `VirtioFS`, a shared file system that allows virtual machines to access a directory tree located on the host. Docker's internal benchmarking shows that with the right resource allocation to the VM, near native file system performance can be achieved with VirtioFS.
   As such, the default memory available to the VM in Docker Desktop for Linux is adjusted. You can tweak this setting to your specific needs by using the **Memory** slider within the **Settings** > **Resources** tab of Docker Desktop.

How do I enable file sharing?

Docker Desktop for Linux uses VirtioFS as the default mechanism to enable file sharing between the host and Docker Desktop VM. Synchronized file shares, a faster, cache-based alternative for bind-mount-heavy workloads like PHP/JS projects, is also available on Linux with a Pro, Team, or Business subscription. For more details, see File sharing/desktop/settings-and-maintenance/settings/#file-sharing.

How do I use Docker SDKs with Docker Desktop for Linux?

Docker Desktop for Linux uses a per-user socket located at `~/.docker/desktop/docker.sock` instead of the system-wide `/var/run/docker.sock`. The Docker CLI handles this automatically through the `desktop-linux` context, but Docker SDKs and other tools that connect directly to the Docker daemon also need the `DOCKER_HOST` environment variable set.

Without setting `DOCKER_HOST`, SDKs attempt to connect to `/var/run/docker.sock` and fail with an error like:

```
Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?
```

To fix this, set the `DOCKER_HOST` environment variable before running your SDK-based application:

```
export DOCKER_HOST=unix://$HOME/.docker/desktop/docker.sock
```

Or dynamically retrieve it from the `desktop-linux` context:

```
export DOCKER_HOST=$(docker context inspect desktop-linux --format '{{ .Endpoints.docker.Host }}')
```

To make this permanent, add the export command to your shell profile (`~/.bashrc`, `~/.zshrc`, or similar):

```
echo 'export DOCKER_HOST=unix://$HOME/.docker/desktop/docker.sock' >> ~/.bashrc
```

Where does Docker Desktop store Linux containers?

Docker Desktop stores Linux containers and images in a single, large "disk image" file in the Linux filesystem. This is different from Docker on Linux, which usually stores containers and images in the `/var/lib/docker` directory on the host's filesystem.

Where is the disk image file?

To locate the disk image file, select **Settings** from the Docker Desktop Dashboard then **Advanced** from the **Resources** tab.

The **Advanced** tab displays the location of the disk image. It also displays the maximum size of the disk image and the actual space the disk image is consuming. Note that other tools might display space usage of the file in terms of the maximum file size, and not the actual file size.

What if the file is too large?

If the disk image file is too large, you can:

- Move it to a bigger drive
- Delete unnecessary containers and images
- Reduce the maximum allowable size of the file

How do I move the file to a bigger drive?

To move the disk image file to a different location:

1. Select **Settings** then **Advanced** from the **Resources** tab.

2. In the **Disk image location** section, select **Browse** and choose a new location for the disk image.
3. Select **Apply** for the changes to take effect.

Do not move the file directly using your file manager, as this can cause Docker Desktop to lose track of the file.

How do I delete unnecessary containers and images?

Check whether you have any unnecessary containers and images. You can see the detailed space usage information by running:

```
$ docker system df -v
```

Alternatively, to list images, run:

```
$ docker image ls
```

To list containers, run:

```
$ docker container ls -a
```

If there are lots of redundant objects, run the command:

```
$ docker system prune
```

This command removes all stopped containers, unused networks, dangling images, and build cache.

It might take a few minutes to reclaim space on the host depending on the format of the disk image file:

- If the file is named `Docker.raw`: space on the host should be reclaimed within a few seconds.
- If the file is named `Docker.qcow2`: space will be freed by a background process after a few minutes.

Space is only freed when images are deleted. Space is not freed automatically when files are deleted inside running containers. To trigger a space reclamation at any point, run the command:

```
$ docker run --privileged --pid=host docker/desktop-reclaim-space
```

Note that many tools report the maximum file size, not the actual file size. To query the actual size of the file on the host from a terminal, run:

```
$ cd ~/.docker/desktop/vms/0/data
$ ls -klsh Docker.raw
2333548 -rw-r--r--@ 1 username  username    64G Dec 13 17:42 Docker.raw
```

In this example, the actual size of the disk is `2333548` KB, whereas the maximum size of the disk is `64` GB.

How do I reduce the maximum size of the file?

To reduce the maximum size of the disk image file:

1. From Docker Desktop Dashboard select **Settings** then **Advanced** from the **Resources** tab.
2. The **Disk image size** section contains a slider that allows you to change the maximum size of the disk image. Adjust the slider to set a lower limit.
3. Select **Apply**.

When you reduce the maximum size, the current disk image file is deleted, and therefore, all containers and images are lost.

□□□ #2
□ □□□ □□□ 2026-10-31 19:14:22 UTC
□ □□□ □□□ 2026-10-31 19:14:23 UTC