Docker storage drivers

Estimated reading time: 10 minutes

Ideally, very little data is written to a container's writable layer, and you use Docker volumes to write data. However, some workloads require you to be able to write to the container's writable layer. This is where storage drivers come in.

Docker supports several different storage drivers, using a pluggable architecture. The storage driver controls how images and containers are stored and managed on your Docker host.

After you have read the storage driver overview (https://docs.docker.com/storage/storagedriver/), the next step is to choose the best storage driver for your workloads. In making this decision, there are three high-level factors to consider:

If multiple storage drivers are supported in your kernel, Docker has a prioritized list of which storage driver to use if no storage driver is explicitly configured, assuming that the storage driver meets the prerequisites.

Use the storage driver with the best overall performance and stability in the most usual scenarios.

Docker supports the following storage drivers:

- `overlay2` is the preferred storage driver, for all currently supported Linux distributions, and requires no extra configuration.
- `aufs` is the preferred storage driver for Docker 18.06 and older, when running on Ubuntu 14.04 on kernel 3.13 which has no support for `overlay2` .
- `devicemapper` is supported, but requires `direct-lvm` for production environments, because `loopback-lvm` , while zero-configuration, has very poor performance. `devicemapper` was the recommended storage driver for CentOS and RHEL, as their kernel version did not support `overlay2` . However, current versions of CentOS and RHEL now have support for `overlay2` , which is now the recommended driver.
- The `btrfs` and `zfs` storage drivers are used if they are the backing filesystem (the filesystem of the host on which Docker is installed). These filesystems allow for advanced options, such as creating "snapshots", but require more maintenance and setup. Each of these relies on the backing filesystem being configured correctly.

- The `vfs` storage driver is intended for testing purposes, and for situations where no copy-on-write filesystem can be used. Performance of this storage driver is poor, and is not generally recommended for production use.

Docker's source code defines the selection order. You can see the order at the source code for Docker Engine - Community 18.09 (https://github.com/docker/docker-ce/blob/18.09/components/engine/daemon/graphdriver/driver_linux.go#L50)

If you run a different version of Docker, you can use the branch selector at the top of the file viewer to choose a different branch.

Some storage drivers require you to use a specific format for the backing filesystem. If you have external requirements to use a specific backing filesystem, this may limit your choices. See Supported backing filesystems (/storage/storagedriver/select-storage-driver/#supported-backing-filesystems).

After you have narrowed down which storage drivers you can choose from, your choice is determined by the characteristics of your workload and the level of stability you need. See Other considerations (/storage/storagedriver/select-storage-driver/#other-considerations) for help in making the final decision.

> *NOTE*: Your choice may be limited by your Docker edition, operating system, and distribution. For instance, `aufs` is only supported on Ubuntu and Debian, and may require extra packages to be installed, while `btrfs` is only supported on SLES, which is only supported with Docker Enterprise. See Support storage drivers per Linux distribution (/storage/storagedriver/select-storage-driver/#supported-storage-drivers-per-linux-distribution) for more information.

Supported storage drivers per Linux distribution

At a high level, the storage drivers you can use is partially determined by the Docker edition you use.

In addition, Docker does not recommend any configuration that requires you to disable security features of your operating system, such as the need to disable `selinux` if you use the `overlay` or `overlay2` driver on CentOS.

Docker Engine - Enterprise and Docker Enterprise

For Docker Engine - Enterprise and Docker Enterprise, the definitive resource for which storage drivers are supported is the Product compatibility matrix (https://success.docker.com/Policies/Compatibility_Matrix). To get commercial support from Docker, you must use a supported configuration.

Docker Engine - Community

For Docker Engine - Community, only some configurations are tested, and your operating system's kernel may not support every storage driver. In general, the following configurations work on recent versions of the Linux distribution:

Linux distribution	Recommended storage drivers	Alternative drivers
Docker Engine - Community on Ubuntu	`overlay2` or `aufs` (for Ubuntu 14.04 running on kernel 3.13)	`overlay` [1], `devicemapper` [2], `zfs` , `vfs`
Docker Engine - Community on Debian	`overlay2` (Debian Stretch), `aufs` or `devicemapper` (older versions)	`overlay` [1], `vfs`
Docker Engine - Community on CentOS	`overlay2`	`overlay` [1], `devicemapper` [2], `zfs` , `vfs`
Docker Engine - Community on Fedora	`overlay2`	`overlay` [1], `devicemapper` [2], `zfs` , `vfs`

[1]) The `overlay` storage driver is deprecated in Docker Engine - Enterprise 18.09, and will be removed in a future release. It is recommended that users of the `overlay` storage driver migrate to `overlay2` .

[2]) The `devicemapper` storage driver is deprecated in Docker Engine 18.09, and will be removed in a future release. It is recommended that users of the `devicemapper` storage driver migrate to `overlay2` .

When possible, `overlay2` is the recommended storage driver. When installing Docker for the first time, `overlay2` is used by default. Previously, `aufs` was used by default when available, but this is no longer the case. If you want to use `aufs` on new installations going forward, you need to explicitly configure it, and you may need to install extra packages, such as `linux-image-extra` . See aufs (https://docs.docker.com/storage/storagedriver/aufs-driver/).

On existing installations using `aufs`, it is still used.

When in doubt, the best all-around configuration is to use a modern Linux distribution with a kernel that supports the `overlay2` storage driver, and to use Docker volumes for write-heavy workloads instead of relying on writing data to the container's writable layer.

The `vfs` storage driver is usually not the best choice. Before using the `vfs` storage driver, be sure to read about its performance and storage characteristics and limitations (https://docs.docker.com/storage/storagedriver/vfs-driver/).

> **Expectations for non-recommended storage drivers: Commercial support is not available for Docker Engine - Community, and you can technically use any storage driver that is available for your platform. For instance, you can use `btrfs` with Docker Engine - Community, even though it is not recommended on any platform for Docker Engine - Community, and you do so at your own risk.**
>
> The recommendations in the table above are based on automated regression testing and the configurations that are known to work for a large number of users. If you use a recommended configuration and find a reproducible issue, it is likely to be fixed very quickly. If the driver that you want to use is not recommended according to this table, you can run it at your own risk. You can and should still report any issues you run into. However, such issues have a lower priority than issues encountered when using a recommended configuration.

Docker Desktop for Mac and Docker Desktop for Windows

Docker Desktop for Mac and Docker Desktop for Windows are intended for development, rather than production. Modifying the storage driver on these platforms is not possible.

Supported backing filesystems

With regard to Docker, the backing filesystem is the filesystem where `/var/lib/docker/` is located. Some storage drivers only work with specific backing filesystems.

Storage driver	Supported backing filesystems
`overlay2`, `overlay`	`xfs` with ftype=1, `ext4`
`aufs`	`xfs`, `ext4`
`devicemapper`	`direct-lvm`

Storage driver	Supported backing filesystems
`btrfs`	`btrfs`
`zfs`	`zfs`
`vfs`	any filesystem

Other considerations

Suitability for your workload

Among other things, each storage driver has its own performance characteristics that make it more or less suitable for different workloads. Consider the following generalizations:

- `overlay2`, `aufs`, and `overlay` all operate at the file level rather than the block level. This uses memory more efficiently, but the container's writable layer may grow quite large in write-heavy workloads.
- Block-level storage drivers such as `devicemapper`, `btrfs`, and `zfs` perform better for write-heavy workloads (though not as well as Docker volumes).
- For lots of small writes or containers with many layers or deep filesystems, `overlay` may perform better than `overlay2`, but consumes more inodes, which can lead to inode exhaustion.
- `btrfs` and `zfs` require a lot of memory.
- `zfs` is a good choice for high-density workloads such as PaaS.

More information about performance, suitability, and best practices is available in the documentation for each storage driver.

Shared storage systems and the storage driver

If your enterprise uses SAN, NAS, hardware RAID, or other shared storage systems, they may provide high availability, increased performance, thin provisioning, deduplication, and compression. In many cases, Docker can work on top of these storage systems, but Docker does not closely integrate with them.

Each Docker storage driver is based on a Linux filesystem or volume manager. Be sure to follow existing best practices for operating your storage driver (filesystem or volume manager) on top of your shared storage system. For example, if using the ZFS storage driver on top of a shared storage system, be sure to follow best practices for operating ZFS filesystems on top of that specific shared storage system.

Stability

For some users, stability is more important than performance. Though Docker considers all of the storage drivers mentioned here to be stable, some are newer and are still under active development. In general, `overlay2`, `aufs`, `overlay`, and `devicemapper` are the choices with the highest stability.

Test with your own workloads

You can test Docker's performance when running your own workloads on different storage drivers. Make sure to use equivalent hardware and workloads to match production conditions, so you can see which storage driver offers the best overall performance.

Check your current storage driver

The detailed documentation for each individual storage driver details all of the set-up steps to use a given storage driver.

To see what storage driver Docker is currently using, use `docker info` and look for the `Storage Driver` line:

```
$ docker info

Containers: 0
Images: 0
Storage Driver: overlay2
 Backing Filesystem: xfs
<output truncated>
```

To change the storage driver, see the specific instructions for the new storage driver. Some drivers require additional configuration, including configuration to physical or logical disks on the Docker host.

> **Important**: When you change the storage driver, any existing images and containers become inaccessible. This is because their layers cannot be used by the new storage driver. If you revert your changes, you can access the old images and containers again, but any that you pulled or created using the new driver are then inaccessible.

Related information