2.5 Configuring Docker Storage

The Docker Engine is configured to use `overlay2` as the default storage driver to manage Docker containers. This provides a performance and scalability improvement on earlier releases that used the device mapper as the default storage driver, but the technology is new and should be tested properly before use in production environments. For more information on `overlay2`, see https://docs.docker.com/engine/userguide/storagedriver/overlayfs-driver/.

Overlay file systems can corrupt when used in conjunction with any file system that does not have dtype support enabled. For Oracle Linux 7 Update 4 or earlier the root partition is automatically formatted with `-n ftype=0`(disabling dtype support), where XFS is selected as the file system, the package installer checks the filesystem for dtype support and if this is not enabled the default storage driver is set to use `devicemapper`. This check is only performed on a fresh installation of Docker. The configuration of an existing Docker installation is unaffected during upgrade.

This configuration change allows Docker to function on a default Oracle Linux 7 system without any additional configuration required, immediately after install. However, this configuration is not recommended for production environments. Performance and scalability can be compromised by this configuration. Therefore, it is important to consider using dedicated storage for Docker and to change the storage driver to use either `btrfs` or `overlay2`.

Important
If you continue to use `devicemapper` as the storage driver, you should be aware that some Docker images, such as the image for Oracle Database, require that the base device size is set to 25GB or more. The default base device size for `devicemapper` is updated to 25GB, but this only meets a minimum requirement for some containers. Where additional capacity may be required, the base device size can be changed by setting the `dm.basesize` start option for a container or, globally, for the Docker Engine.

You can change this value globally, by adding it to the `storage-opts` configuration parameter in `/etc/docker/daemon.json`, for example:

```
{
...
"storage-opts" : [ "dm.basesize" : "50G" ],
...
}
```

Note that the base device size is sparsely allocated, so an image may not initially use all of this space. You can check how much space is allocated to the `Base Device Size` by running the **docker info**command.

See https://docs.docker.com/engine/reference/commandline/dockerd/#storage-driver-options for more information on storage driver options.

Oracle recommends using btrfs as a more stable and mature technology than overlayfs.

In most cases, it is advisable to create a dedicated file system to manage Docker containers. This file system can be mounted at `/var/lib/docker` at boot time, before the Docker service is started.

Any unused block device that is large enough to store several containers is suitable. The suggested minimum size is 1GB but you might require more space to implement complex Docker applications. If the system is a virtual machine, Oracle recommends that you create, partition, and format a new virtual disk. Alternatively, convert an existing ext3 or ext4 file system to btrfs.
See https://docs.oracle.com/cd/E52668_01/E54669/html/ol7-use-case7-btrfs.html in the *Oracle Linux Administrator's Guide for Release 7*. If an LVM volume group has available space, you can create a new logical volume and format it as a btrfs file system.

Important
XFS file systems must be created with the `-n ftype=1` option enabled for use as an overlay. The root partition on Oracle Linux 7 is automatically formatted with `-n ftype=0` where XFS is selected as the file system. Therefore, if you intend to use the `overlay2` storage driver in this environment, you must format a separate device for this purpose.

Using the **docker-storage-config** Utility to Automatically Configure Docker Storage

As of Oracle Container Runtime for Docker 17.06, the `docker-engine` package includes a utility that can help you to configure storage correctly for a new Docker deployment. The **docker-storage-config** utility can format a new block device, set up the mount point and correctly configure the Docker Engine to run with the appropriate storage driver so that your storage configuration follows Oracle guidelines.

For usage instructions, run **docker-storage-config** with the `-h` option:

```
# docker-storage-config -h
```

The **docker-storage-config** utility requires that you provide the path to a valid block device to use for Docker storage. Note that this script formats the device with a new file system. This can be a destructive operation. Any existing data on the device may be lost. Use the **lsblk** command to help you correctly identify block devices currently attached to the system.

To automatically set up your Docker storage, before installation, run **docker-storage-config** as root:

```
# docker-storage-config -s btrfs -d /dev/sdb1
```

Substitute ***/dev/sdb1*** with the path to the block device that you attached as dedicated storage.

You can substitute ***btrfs*** with `overlay2` if you would prefer to use this storage driver. If you do this, the block device is formatted with XFS and dtype support is enabled.

To overwrite an existing configuration, you can use the `-f` flag. If your Docker installation has already been used to set up images and containers, this option is destructive and may make these images and containers inaccessible to you, so the option should be used with caution.

Manually Preparing a Dedicated File System to Manage Docker Containers

1. To configure the Docker Engine to use btrfs as the storage driver to manage containers:
    a. Use **yum** to install the `btrfs-progs` package.

```
# yum install btrfs-progs
```

b. If the root file system is not configured as a btrfs file system, create a btrfs file system on a suitable device or partition such as `/dev/sdb1` in this example:

```
# mkfs.btrfs /dev/sdb1
```

2. To configure the Docker Engine to use a block device formatted with XFS in conjunction with the `overlay2` storage driver to manage containers:
   a. Format the block device with the XFS file system, for example to format a partition `/dev/sdb1`:

```
# mkfs -t xfs -n ftype=1 /dev/sdb1
```

It is essential that you use the `-n ftype=1` option when you create the file system or you will not be able to use overlayfs. To check if a mounted XFS partition has been formatted correctly, run the following command and check the output to make sure that `ftype=1`:

```
# xfs_info /dev/sdb1 | grep ftype
```

3. Use the **blkid** command to display the UUID and TYPE for the new file system and make a note of this value, for example:

```
# blkid /dev/sdb1
/dev/sdb1: UUID="26fece06-e3e6-4cc9-bf54-3a353fdc5f82" TYPE="xfs" \
  PARTUUID="ee0d0d72-dc97-40d8-8cd9-39e29fbc660e"
```

The UUID for the file system on the device `/dev/sdb1` in this example is the `UUID` value `26fece06-e3e6-4cc9-bf54-3a353fdc5f82`. You can ignore the `PARTUUID` value, which is the UUID of the underlying partition. The TYPE of file system in this example is the `TYPE` value `xfs`.

6. Create an entry in your fstab to ensure that the file system is mounted at boot. Open `/etc/fstab` in an editor and add a line similar to the following:

```
UUID=UUID_value /var/lib/docker   fstype     defaults        0 0
```

Replace ***UUID_value*** with the UUID value that you found in step 3. Replace ***fstype*** with the file system TYPE reported in step 3.

Note

Previous versions of Docker required that dedicated storage used by Docker was mounted via a Systemd mount target and a Systemd drop-in file for the Docker service. This requirement was related to an issue where the storage was automatically unmounted when the Docker service was stopped. This issue no longer applies. If your storage is currently mounted using these methods, consider simplifying your environment by removing the Systemd drop-in and mount target and replacing this with an fstab entry.

This entry defines a mount for the file system on `/var/lib/docker`. You might need to create this directory if you are performing a fresh installation:

```
# mkdir /var/lib/docker
```

You must mount the file system to start using it:

```
# mount /var/lib/docker
```

Manually Configure Docker to Use a Specified Storage Driver

1. Create or edit `/etc/docker/daemon.json`.

   If you are creating this file from scratch, it should look like this:

   ```
   {
     "storage-driver": "btrfs"
   }
   ```

   Replace ***btrfs*** with your preferred storage driver. If you are using an XFS, ext3 or ext4 file system, you might replace ***btrfs*** with `overlay2`.

   If this file already exists and contains other entries, be careful that adding a line for the `storage-driver` configuration variable conforms with typical JSON formatting.

   For more information on the format and options for this configuration file, see https://docs.docker.com/engine/reference/commandline/dockerd/#daemon-configuration-file.

2. Check that the `--storage-driver` option is not being invoked as a command line switch when starting the Docker Engine daemon.

   You should check that this option does not appear in either the `/etc/sysconfig/docker` or `/etc/sysconfig/docker-storage` files. These files are deprecated and may be removed in future releases. If these files contain any other configuration parameters, consider whether you could move these into `/etc/docker/daemon.json` to future-proof your configuration.

   Also check that this option does not appear in any systemd drop-in files in `/etc/systemd/system/docker.service.d/`. While this is a supported configuration option, it is preferable to keep all Docker Engine configuration consolidated and in the same place, where possible.

3. Once you have started the Docker Engine and it is running, check that it is using the storage driver that you have configured:

   ```
   # docker info | grep Storage
   ```

   You can run the **docker info** command on its own to get a more thorough view of the configuration.