Oracle Linux 8

Managing the Btrfs File System

G26929-01
July 2025

Contents

Preface

Oracle Linux 8: Managing the Btrfs File System provides information about managing the Btrfs file system on Oracle Linux 8 systems.

Documentation License

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1 About the Btrfs File System

The Btrfs file system is designed to meet the expanding scalability requirements of large storage subsystems. Because the Btrfs file system uses B-trees in its implementation, its name is derived from the name of those data structures, although it's not a true acronym. A *B-tree* is a tree-like data structure that enables file systems and databases to efficiently access and update large blocks of data, irrespective of how large the tree grows.

The Btrfs file system provides the following important features:

- **Snapshots**: Btrfs uses Copy-On-Write (COW) functionality so that you can take near instant readable and writable snapshots. Snapshots can enable you to roll back a file system to a previous state, even after converting it from an `ext3` or `ext4` file system.
- **Instant file copy**: Similar to snapshots, you can use the reflink feature available in Btrfs to perform instant copies of large files on the file system.
- **Logical Volume Management**: Btrfs includes integrated logical volume management that feature built-in data redundancy, software RAID functionality, and the ability to dynamically add and remove storage capacity.
- **Online maintenance**: Btrfs includes features for online repair and maintenance operations, which minimizes downtime.
- **Corruption detection and correction**: Btrfs verifies checksums each time a data block is read from disk to help ensure data integrity.
- **Compression and Defragmentation**: Btrfs includes automatic and transparent defragmentation for improved performance. Also, Btrfs can apply compression to the file system, individual subvolumes, or to specific files and directories. Compression can provide significant disk space savings.
- **Remote send and receive**: Btrfs includes a send and receive function that can be used to efficiently perform remote incremental backups to save bandwidth and to easily create a remote mirror of a subvolume.
- **File system seeding**: Btrfs can be used to create a read-only file system that acts as a template to seed other Btrfs file systems, similar in functionality to other union mount file system types such as overlayfs or qcow2.

For more information, visit https://btrfs.readthedocs.io/.

For an overview of local file system management, see Oracle Linux 8: Performing File System Administration.

Note:

In Oracle Linux, the Btrfs file system type, and all the features that are documented are supported on the Unbreakable Enterprise Kernel (UEK) release *only*. Working with Btrfs file system features requires that you boot the system by using UEK R6 or later.

2 Installing the Btrfs Utilities

Btrfs on Oracle Linux requires that the system is booted with UEK and that you install the `btrfs-progs` package to perform file system administration tasks on the file system.

Btrfs isn't available for RHCK, so you need to ensure that the system has UEK installed and is booted into the correct kernel. Also, install the `btrfs-progs` package that provides many Btrfs administration tools, including the `mkfs.btrfs` command that you use to format a device with the Btrfs file system.

1. Enable the appropriate UEK yum repository or ULN channel for the system, if this hasn't already been done.

   For example, to enable the `ol8_UEKR7` repository, run:

   ```
   sudo dnf config-manager --enable ol8_UEKR7
   ```

2. Install UEK on the system, if it isn't already installed.

   ```
   sudo dnf install -y kernel-uek
   ```

   If the system isn't already booted into UEK, you might need to reboot the system.

3. Validate that the `btrfs` kernel module is available for the booted kernel.

   ```
   modinfo btrfs
   ```

4. Install the `btrfs-progs` package.

   ```
   sudo dnf install -y btrfs-progs
   ```

3
Setting Up and Administering a Btrfs File System

This section describes procedures to create a Btrfs file system as well as administering it for a more efficient use.

Creating and Mounting a Btrfs File System

Use the `mkfs.btrfs` command to create a Btrfs file system that's laid out across one or more block devices.

Btrfs can be used on a single block device, or you can use the built-in RAID functionality to format the file system across more than one block device.

When formatting more than one block device, Btrfs defaults to mirror the file system metadata across the devices. If you specify a single device, the metadata is duplicated on that device, unless you specify to use only one copy of the metadata.

The devices can be whole block device(s), disk partitions, files, loopback devices, multipath devices, or LUNs that implement RAID in hardware.

See the `mkfs.btrfs(8)` manual page for more detailed information about the `mkfs.btrfs` command the various Btrfs configurations that you can create.

1. Identify the target device, partition, or file that you want to format with Btrfs.

   Typically, you might use the `lsblk` command to list block devices and partitions that are available to the system.

   Note that formatting a file system is a destructive operation and erases any data on the target device. Ensure that the following steps use the correct target device or file paths.

2. Use the `mkfs.btrfs` command to format the device or devices.

   ```
   sudo mkfs.btrfs [options] <device> [<device>...]
   ```

   To format a single block device, `/dev/sdb`, with Btrfs, you can run:

   ```
   sudo mkfs.btrfs /dev/sdb
   ```

   To format several block devices, in a RAID10 configuration, you can run:

   ```
   sudo mkfs.btrfs -d raid10 -m raid10 /dev/sdb /dev/sdc /dev/sdb /dev/sde
   ```

   Note that in the example, RAID10 requires a minimum of four block devices and stripes data across mirrored pairs. The appropriate number of devices required to achieve the specified RAID configuration must be used when formatting the file system.

   The `mkfs.btrfs` command automatically scans the new Btrfs devices into the kernel. To scan and assemble all the relevant devices of the volume in the kernel you need to run the `btrfs device scan` command. You can undo this action by using the `btrfs device scan --forget` command. To assemble all the relevant devices of the volume, you might need to run the `btrfs device scan` command.

3. Mount the file system.

   For example, you can run:

   ```
   sudo mount /dev/sdb /mnt
   ```

   If you have formatted the file system across several devices, you can specify any of the component devices to perform the mount.

   Note that the default subvolume is mounted unless an alternative subvolume ID or path is specified. See Mounting Btrfs Subvolumes for more information.

4. View the disk space information for the mounted Btrfs file system, by using the `btrfs filesystem df` command.

   For example, you can run:

   ```
   btrfs filesystem df /mnt
   ```

   The `btrfs filesystem df` command displays more exact information about the space that's used by a Btrfs file system than the `df` command.

   For more detailed information about the file system, use the `btrfs filesystem usage` command. For example:

   ```
   btrfs filesystem usage /mnt


   Overall:
       Device size: 1.95TiB
       Device allocated: 5.03GiB
       Device unallocated: 1.95TiB
       Device missing: 0.00B
       Used: 256.00KiB
       Free (estimated): 999.48GiB (min: 999.48GiB)
       Data ratio: 2.00
       Metadata ratio: 2.00
       Global reserve: 3.25MiB (used: 0.00B)

   Data,RAID10: Size:2.00GiB, Used:0.00B (0.00%)
      /dev/nvme0n1p6 1.00GiB
      /dev/nvme0n1p7 1.00GiB
      /dev/nvme0n1p8 1.00GiB
      /dev/nvme0n1p9 1.00GiB

   Metadata,RAID10: Size:512.00MiB, Used:112.00KiB (0.02%)
      /dev/nvme0n1p6 256.00MiB
      /dev/nvme0n1p7 256.00MiB
      /dev/nvme0n1p8 256.00MiB
      /dev/nvme0n1p9 256.00MiB

   System,RAID10: Size:16.00MiB, Used:16.00KiB (0.10%)
      /dev/nvme0n1p6 8.00MiB
   ```

```
      /dev/nvme0n1p7 8.00MiB
      /dev/nvme0n1p8 8.00MiB
      /dev/nvme0n1p9 8.00MiB

   Unallocated:
      /dev/nvme0n1p6 498.74GiB
      /dev/nvme0n1p7 498.74GiB
      /dev/nvme0n1p8 498.74GiB
      /dev/nvme0n1p9 498.74GiB
```

Changing Btrfs File System Devices

You can use the `btrfs device` command to add, remove, or replace devices, and the `btrfs rebalance` command to rebalance the layout of the file system data and metadata across devices. The following table describes each of the commands that can use to perform these tasks.

Command	Description
`btrfs device add` *device mountpoint*	Add a device to the file system that's mounted on the specified mount point, for example: `sudo btrfs device add /dev/sdd /mnt`
`btrfs device delete` *device mountpoint*	Remove a device from a mounted file system, for example: `sudo btrfs device delete /dev/sde /mnt`
`btrfs device delete missing` *mountpoint*	Remove a failed device from the file system that's mounted in degraded mode, for example: `sudo btrfs device remove missing /mnt` To mount a file system in degraded mode, specify the `-o degraded` option to the `mount` command. For a RAID configuration, if the number of devices would become less than the minimum number that are required, you must add the replacement device before removing the failed device.

Command	Description
`btrfs device replace start` *source_dev target_dev mountpoint*	Replace devices on a live file system without unmounting the file system or stopping any tasks that are using the file system, for example: `sudo btrfs device replace start /dev/sdb /dev/sdk /mnt` If the system crashes or loses power while the replacement is taking place, the operation resumes when the system next mounts the file system. The target device must be the same size or larger than the source device. If the source device is no longer available, or you specify the `-r` option, the data is reconstructed by using redundant data that's obtained from other devices, such as another available mirror. The source device is removed from the file system when the operation is complete. Use the `btrfs replace status` *mountpoint* command to check the progress of the replacement operation and the `btrfs replace cancel` *mountpoint* command to cancel the operation.
`btrfs filesystem balance` *mountpoint*	After adding or removing devices, redistribute the file system data and metadata across the available devices.

Defragmenting and Compressing a Btrfs File System

You can defragment the file system to increase I/O performance. You can also compress a Btrfs file system to increase its effective capacity.

You can defragment the file system or any files and directories within the file system by running the `btrfs filesystem defragment` command. You can automatically run defragmentation on the file system by setting the `autodefrag` option when you mount it. However, note that automatic defragmentation isn't recommended for large databases or for images of virtual machines.

Note:

Defragmenting a file or a subvolume with a copy-on-write copy breaks the link between the file and its copy. For example, if you defragment a subvolume that has a snapshot, the disk usage by the subvolume and its snapshot increases because the snapshot is no longer a copy-on-write image of the subvolume.

You can apply compression during a defragmentation, but compression can also be set as a mount option so that it's applied to the whole file system, or can be set as a property on a subvolume so that it's applied to a particular subvolume.

Three different compression types are supported:

- `zlib`: General-purpose and widely used compression with balanced speed and compression ratio.
- `lzo`: Rapid compression and decompression with a lower compression ratio than other options.
- `zstd`: High compression ratio and rapid compression and decompression.

You can enable compression at any point and only the new writes are compressed unless compression is applied when defragmentation is run.

- To defragment a Btrfs file system, run:

  ```
  sudo btrfs filesystem defragment /mnt
  ```

- To mount a Btrfs file system with automatic defragmentation enabled, run:

  ```
  sudo mount -o autodefrag /dev/sdb /mnt
  ```

- To apply compression to a Btrfs file system during a defragmentation operation, run:

  ```
  sudo btrfs filesystem defragment -czlib /mnt
  ```

- To apply compression to a subvolume by setting a compression property on the subvolume, use the `btrfs property set` command.

  ```
  sudo btrfs property set /mnt/subvol1 compression zstd
  ```

- To apply compression when mounting the file system, run:

  ```
  sudo mount -o compress=lzo /dev/sdb /mnt
  ```

Resizing a Btrfs File System

You can use the `btrfs filesystem resize` command to resize a mounted Btrfs file system.

You can use the `btrfs` command to increase the size of a mounted Btrfs file system if enough space exists on the underlying devices to accommodate the change. You also use the `btrfs` command to decrease its size, if the file system has enough available free space. Note that running the command doesn't have any effect on the layout or size of the underlying devices.

- To increase the size of a Btrfs file system by a specified amount, such as 2 GB, run:

  ```
  sudo btrfs filesystem resize +2g /mybtrfs1
  ```

- To decrease the size of a Btrfs file system by a specified amount, such as 4 GB, run:

  ```
  sudo btrfs filesystem resize -4g /mybtrfs2
  ```

- To specify the size of a Btrfs file system to a specified size, such as 20GB, run:

  ```
  sudo btrfs filesystem resize 20g /mybtrfs3
  ```

4

Managing Subvolumes and Snapshots

The top level of a Btrfs file system is a subvolume consisting of a named B-tree structure containing directories, files, and possibly further subvolumes that are also named B-trees, each of which also contains directories and files, and so on.

Snapshots are a type of subvolume that record the contents of their parent subvolumes at the time that you took the snapshot. If you take a snapshot of a Btrfs file system and don't write to it, the snapshot records the state of the original file system and forms a stable image from which you can make a backup. If you make a snapshot writable, you can treat it as a alternative version of the original file system. The copy-on-write functionality of a `btrfs` file system means that snapshots are created quickly and consume little disk space initially.

Taking snapshots of a subvolume isn't a recursive process. If you create a snapshot of a subvolume, every subvolume or snapshot that the subvolume contains is mapped to an empty directory of the same name inside that snapshot.

NOT_SUPPORTED:

Snapshots record the state of the file system at a moment in time. As such, it's not possible to guarantee file system integrity for transactional processes that might have been in operation at the time when a snapshot was taken. While utilities such as the `snapper` command might help capture before and after snapshots for particular operations, such as when using the `dnf` command, these snapshots are still unaware of other processes that might be running on the system at the same time. If you have processes that have intensive I/O or memory usage, such as database or middleware applications, stop these or ensure that all activity is complete before taking a snapshot to reduce the likelihood of data integrity or file system corruption issues within the snapshot.

The `snapper` command to be used to create and manage Btrfs snapshots and provides automation facilities to ensure that snapshots are taken at key system events such as during software installation or upgrade, see Automating File System Snapshots With the Snapper Utility.

You can mount a Btrfs subvolume as though it were a disk device. If you mount a snapshot instead of its parent subvolume, you effectively roll back the state of the file system to when the snapshot was taken. By default, the operating system mounts the parent Btrfs volume, which has an ID of `5`, unless you use the `set-default` option to change the default subvolume. If you set a new default subvolume, the system mounts that subvolume going forward. Y

After you have rolled back a file system by mounting a snapshot, you can take snapshots of the snapshot to record its state as it changes.

Deleting a subvolume deletes all subvolumes under it in the B-tree hierarchy. For this reason, you can't remove the topmost subvolume of a Btrfs file system, which has an ID of `5`.

Creating Btrfs Subvolumes and Snapshots

Subvolumes and snapshots are key features within Btrfs. Subvolumes can be mounted anywhere within the file system. Snapshots are types of subvolumes that create Copy-On-Write (COW) copies of the data within a subvolume at a moment in time, thereby providing incremental backup and rollback functionality within the file system.

1. Use the `btrfs subvolume create` command to create a subvolume.

   To create a subvolume, specify the path to the subvolume that you want to create on the mounted Btrfs file system:

   ```
   btrfs subvolume create /mnt/subvolume1
   ```

   The subvolume is created and appears as an empty subdirectory within the directory on the mounted Btrfs file system where you created it.

2. Use the `btrfs subvolume snapshot` command to create a snapshot of a subvolume.

   To create a snapshot of a subvolume, specify the path to the subvolume that you want to create a snapshot of and the path on the Btrfs file system where you want to store snapshots. For example, you might run:

   ```
   btrfs subvolume snapshot /mnt/subvolume1 /mnt/subvolume1_$(date -I)
   ```

   A snapshot of *subvolume1* is created using the current date as part of its name. The snapshot operation is nearly instant because of the COW nature of the file system. The snapshot contains all the data from the subvolume that it was created from at the moment in time that it was created. The snapshot also doesn't use a significant amount of extra disk space.

   > **Important:**
   >
   > Taking snapshots of a subvolume isn't a recursive process. If you create a snapshot of a subvolume, every subvolume or snapshot that the subvolume contains is mapped to an empty directory of the same name inside that snapshot.

   You can automate snapshots of the file system by using the Snapper utility. See Automating File System Snapshots With the Snapper Utility for more information.

3. Use the `btrfs subvolume list` command to list all the subvolumes within a btrfs file system.

   List the subvolumes within a Btrfs file system at a mount point, for example:

   ```
   btrfs subvolume list /mnt
   ```

   All the subvolumes from the top-level subvolume in the file system at the specified path are listed, regardless of whether the path is mounted onto a subvolume deeper into the tree of subvolumes. Snapshots are also listed, because snapshots are also subvolumes.

Note:

You can use this command to find the ID values of the available subvolumes or snapshots. The subvolume ID value can be useful when mounting an alternative subvolume or snapshot for a rollback.

4. Use the `btrfs subvolume delete` command to delete a subvolume when you no longer need it.

   For example, to delete a snapshot or subvolume, you can run:

   ```
   btrfs subvolume delete /mnt/snapshots/subvol_2025-01-24
   ```

Mounting Btrfs Subvolumes

You can mount subvolumes by subvolume ID or path. This facility makes it easy to access and switch out the subvolumes that are used at different mount points. Most notably, you can use the ability to mount specified subvolumes to easily rollback a mount point to use a particular snapshot.

1. Mount a subvolume to a specified mount point by using the `mount` command and specify the `subvolid` option or by using the `subvol` option.

   You can mount any subvolume or snapshot by specifying its subvolume ID or subvolume path as an option when running the mount command. For example, to mount using the subvolume ID:

   ```
   sudo mount -o "subvolid=256" /dev/sdb /home
   ```

   You can get the subvolume ID for any subvolume or snapshot by using the `btrfs subvolume list` command.

   To mount a subvolume by specifying its subvolume path, you can run:

   ```
   sudo mount -o "subvol=/snapshots/home_2025-01-24" /dev/sdb /home
   ```

   Note:

   When mounting the subvolume by using the `subvol` option, the subvolume path must exist within the root subvolume of the Btrfs file system, even if that location is nested. You can't mount a subvolume by path if it's a subvolume of another subvolume. If you need to mount a nested subvolume, mount it by using the `subvolid` option.

2. Set the default ID for a Btrfs file system to use when it's mounted by using the `btrfs subvolume set-default` command.

You can change the default subvolume ID that the system uses when mounting a Btrfs file system. This operation is often useful when you're trying to rollback a system mount to a previous snapshot. To change the default subvolume ID, run:

```
sudo btrfs subvolume set-default 256 /mnt
```

When the file system is next mounted, the subvolume specified by the value that you set is loaded as the default subvolume and attached to the mount point.

You can check the current default subvolume ID for the file system by running the `btrfs subvolume get-default` command.

Setting the default subvolume ID is useful if trying to configure a root file system to use a snapshot to rollback to a previous point in time. See Working With a Btrfs root File System for more information.

Managing Quotas for Btrfs Subvolumes With Quota Groups

Note:

Be aware that the quota groups feature is available as a Technology Preview *only* in Oracle Linux. Working with this feature requires that you boot the system by using UEK R6 or later.

1. Use the `btrfs quota enable` command to enable quotas on a volume:

   Enable quotas by running following command on a newly created Btrfs file system before any creating any subvolumes:

   ```
   sudo btrfs quota enable volume
   ```

2. Assign a quota-group limit to a subvolume by using the `btrfs qgroup limit` command:

   The following example shows how you would use this command:

   ```
   sudo btrfs qgroup limit 1g /myvol/subvol1
   ```

   ```
   sudo btrfs qgroup limit 512m /myvol/subvol2
   ```

3. To find out the quota usage for a subvolume, use the `btrfs qgroup show` command.

   For example, run:

   ```
   btrfs qgroup show /myvol/subvol1
   ```

Automating File System Snapshots With the Snapper Utility

The Snapper utility can be used to automate the management of file system snapshots. The utility can make it easier to create and delete snapshots, while enabling users to compare the differences between snapshots and revert changes at the file level. For information about the Snapper utility, visit the upstream project page at http://snapper.io/.

NOT_SUPPORTED:

Snapshots record the state of the file system at a moment in time. As such, it's not possible to guarantee file system integrity for transactional processes that might have been in operation at the time when a snapshot was taken. While utilities such as the `snapper` command might help capture before and after snapshots for particular operations, such as when using the `dnf` command, these snapshots are still unaware of other processes that might be running on the system at the same time. If you have processes that might have intensive I/O or memory usage, such as database or middleware applications, stop these or ensure that all activity is complete before taking a snapshot to help reduce the likelihood of data integrity or file system corruption issues within the snapshot.

Installing Snapper

If not already installed, you can install the Snapper utility from the `ol8_UEKR6` yum repository.

- Use the `dnf` command to install Snapper.

  ```
  sudo dnf install -y snapper
  ```

Creating a Snapper Configuration for a Subvolume

You can use the `snapper` command to create and manage snapshots of Btrfs subvolumes. You must create a configuration entry for each Btrfs subvolume that you want to use with Snapper. If a subvolume is mounted at a directory that's outside of the default SELinux policy configuration, you might need to update SELinux file contexts for that directory to use Snapper to create snapshots of that directory.

1. Create a Snapper configuration for an existing mounted Btrfs subvolume.

   To set up the Snapper configuration for an existing mounted Btrfs subvolume:

   ```
   sudo snapper -c config_name create-config -f btrfs fs_name
   ```

   In the previous command, *config_name* is the name of the configuration and *fs_name* is the path of the mounted Btrfs subvolume. Running the command does the following:

   - Adds an entry for *config_name* to the `/etc/sysconfig/snapper` file.
   - Creates the configuration file `/etc/snapper/configs/`*`config_name`* .
   - Sets up a `.snapshots` subvolume for the snapshots.

   For example, the following command sets up the Snapper configuration for a Btrfs `root` file system:

   ```
   sudo snapper -c root_config create-config -f btrfs /
   ```

2. Use the `snapper list-configs` command to list all the existing configurations:

   ```
   sudo snapper list-configs
   ```

   ```
   Config      | Subvolume
   ------------+----------
   home_config | /home
   root_config | /
   ```

3. Update SELinux file contexts for any directories outside of the default policy.

   The default Snapper SELinux policy allows Snapper to manage snapshots in the `/`, `/etc`, `/mnt`, `/usr`, `/var` and `HOME_ROOT` (usually `/home`). If you create a directory, for example `/data` or `/srv`, you might need to set the SELinux file context for that directory so that Snapper can create and manage snapshots for that directory.

   For example to enable Snapper to manage snapshots on the `/data` directory, you can run:

   ```
   sudo semanage fcontext -a -t snapperd_data_t "/data/\.snapshots(/.*)?"
   sudo restorecon -R -v /data
   ```

Understanding Different Types of Snapshots

You can create the following three types of snapshots by using the `snapper` command:

single
You use a *single snapshot* to record the state of a subvolume but it doesn't have any association with other snapshots of the subvolume.

pre
You use a *pre snapshot* to record the state of a subvolume before a modification. A pre snapshot is always paired with a *post snapshot* that you take immediately after you have completed the modification.

post
You use a *post snapshot* to record the state of a subvolume after a modification. A post snapshot is always paired with a *pre snapshot* that you take immediately before you make the modification.

To create a single snapshot of a subvolume, use the `snapper create` command, for example:

```
sudo snapper -c root_config create --description "Regular daily snapshot"
```

Single snapshots are useful for periodic backup purposes and can also be used to create a back-up timeline, as described in Configuring Automatic Snapper Snapshots.

For actions that are likely to result in specific file system modifications that you might need to roll back, you can use pre and post snapshots to capture snapshots of the file system before and after a transaction.

For example, the following commands create pre and post snapshots of a subvolume:

```
sudo snapper -c root_config create -t pre -p
```

After you have created the pre snapshot, perform some action that makes a change to the file system. Then run the following command to create a post snapshot.

```
sudo snapper -c root_config create -t post --pre-num N -p
```

Specifying the `-p` option with the `snapper` command displays the number of the snapshot so that you can reference it when creating the post snapshot or when comparing the contents of the `pre` and `post` snapshots.

The action of creating a pre and post snapshot can be condensed by using the `--command` option with the `snapper create` command to wrap a specified command line operation with `pre` and `post` snapshots. For example:

```
snapper -c root create --command "cd /tmp/build; make install" \
    --description "Installing a home built binary"
```

Pre and post snapshots are often used when performing system changes that might be too complex to revert manually, such as when installing or upgrading packages.

The DNF Snapper plugin that's described in Configuring Automatic Snapper Snapshots uses pre and post snapshots in exactly this way and uses the description field to store the DNF transaction that triggered the snapshot.

For example, in the following set of snapshots, you can identify periodic, single snapshots that are triggered as part of a timeline and then a pre and post snapshot that's triggered by the DNF Snapper plugin when the `vim` package is installed.

```
$ sudo snapper -c root list


 # | Type   | Pre # | Date                            | User | Cleanup  |
Description              | Userdata
---+--------+-------+-------------------------------+------+----------
+-------------------------+---------
0  | single |       |                               | root |          |
current                  |
1  | single |       | Wed 25 Nov 2020 07:00:30 EST | root | timeline |
timeline                 |
2  | single |       | Wed 25 Nov 2020 08:00:01 EST | root | timeline |
timeline                 |
3  | single |       | Wed 25 Nov 2020 09:00:01 EST | root | timeline |
timeline                 |
4  | pre    |       | Wed 25 Nov 2020 09:07:21 EST | root | number
| /usr/bin/dnf install vim |
5  | post   |     4 | Wed 25 Nov 2020 09:07:25 EST | root | number
| /usr/bin/dnf install vim |
6  | single |       | Wed 25 Nov 2020 10:00:01 EST | root | timeline |
timeline                 |
```

Configuring Automatic Snapper Snapshots

Snapper can be configured to perform automatic periodic backups to create a system timeline that you can use to roll the system back to a moment in time. You can also use a DNF plugin with Snapper so that pre and post backups are created for every DNF transaction after the plugin has been installed.

Automated periodic backups are provided as a key feature included with Snapper. These backups take advantage of Snapper's single backup feature and use a systemd timer unit to automatically take backups on an hourly schedule. A second systemd timer unit is included to prune backups to reduce the number of snapshots stored on the system. The timeline options for each Snapper configuration can be customized for each subvolume.

See Automating Single Backups To Create a Snapper Timeline for more information about Snapper timeline configuration.

The DNF plugin for Snapper is a powerful tool that can be used to quickly roll a system back to the moment before you performed a software installation or upgrade. This tool is helpful if a software installation or upgrade is likely to perform significant changes to the file system or to current system configuration. See https://dnf-plugins-extras.readthedocs.io/en/latest/snapper.html for more information. SeeAutomating Pre And Post Backups For DNF Transactions to learn how to enable this feature on the system.

Automating Single Backups To Create a Snapper Timeline

Snapper can be configured to perform periodic single backups to create a Snapper timeline that you can roll back to for any of the subvolumes that you have configured on the system.

The following steps describe how to enable the systemd units that control Snapper timeline management, and how to configure the number of snapshots to store within the timeline. See the `snapper(8)` and `snapper-configs(5)` manual pages for more information.

1. Enable the `snapper-timeline` systemd unit.

   Automatic snapshots are triggered by a systemd timer unit that you must enable to so that the timeline can be created:

   ```
   sudo systemctl enable --now snapper-timeline.timer
   ```

2. Enable the `snapper-cleanup` systemd unit to automatically remove stale snapshots and keep snapshots manageable.

   ```
   sudo systemctl enable --now snapper-cleanup.timer
   ```

   By default, the Snapper timeline configuration keeps 10 hourly, 10 daily, 10 monthly, and 10 yearly snapshots. Snapshots are pruned by the cleanup timer.

3. Review subvolume Snapper configurations to customize Snapper timeline configuration.

   Each Snapper configuration contains settings for a periodic backup, which is controlled by the `TIMELINE_CREATE` configuration variable in the `/etc/snapper/configs/`*`config_name`* file.

   When the systemd timer units are enabled, periodic snapshot events trigger automatically for every Snapper configuration that has the `TIMELINE_CREATE` variable enabled. To disable periodic snapshots for a particular subvolume configuration, change the variable value to `no` in the configuration file.

You can also edit the configuration to control the number of timeline snapshots that are kept by the `snapper-cleanup` process. Set these values by changing the following configuration variables:

```
TIMELINE_LIMIT_HOURLY="10"
TIMELINE_LIMIT_DAILY="10"
TIMELINE_LIMIT_WEEKLY="10"
TIMELINE_LIMIT_MONTHLY="10"
TIMELINE_LIMIT_YEARLY="10"
```

Automating Pre And Post Backups For DNF Transactions

You can install the DNF Snapper plugin on a system to automatically trigger pre and post snapshots for DNF transactions.

This feature can help you roll back changes in cases where system package upgrades cause a failure that you need to debug or to enable you to analyze which files were changed during an installation or upgrade. Note that this plugin requires no user configuration or interaction to work. Pre and post snapshots are created automatically for every DNF transaction after the plugin is installed on the system.

- Use the dnf command to install the plugin:

  ```
  sudo dnf install -y python3-dnf-plugin-snapper
  ```

Working With Btrfs Snapshots by Using Snapper

Use this tabulated summary of commonly performed actions as a reference when you're working with Snapper to manage Btrfs snapshots. For more information, see the `snapper(8)` manual page.

Table 4-1 Commonly Used Snapper Command Reference

Action	Command and Description
Create a single snapshot.	To create a single snapshot of a subvolume, use the `snapper create` command, for example: `sudo snapper -c` *`config_name`* `create --description "`*`description`*`"`

Create pre and post snapshots for a command.	Use the `--command` option with the `snapper create` command to wrap an operation with `pre` and `post` snapshots. For example:
	`snapper -c` *`config_name`* `create --command "`*`command`*`" \`
	`    --description "`*`description`*`"`

Table 4-1 (Cont.) Commonly Used Snapper Command Reference

Action	Command and Description
List snapshots for a configuration.	To list the snapshots that exist for a Snapper configuration or subvolume, run: `sudo snapper -c` *`config_name`* `list`

View file and directory changes between snapshots.	To display the files and directories that have been added, removed, or changed between two snapshots, use the `status` subcommand and specify the numbers of the two snapshots that you want to compare:
	
	`sudo snapper -c` *`config_name`* `status`
	`N` `..` *`N'`*

Show the differences within files between snapshots.	To display the differences between the contents of all the files between two snapshots, use the `diff` subcommand:
	
	`sudo snapper -c` *`config_name`* `diff`
	`N` `..` *`N'`*
	
	You can also display the difference in a single file over two snapshots by providing the full path to the file:
	
	`sudo snapper -c` *`config_name`* `diff`
	`N` `..` *`N'`* *`/path/to/file`*

Delete a snapshot.	To delete a snapshot, specify its number to the `delete` subcommand:
	
	`sudo snapper -c` *`config_name`* `delete`
	`N''`

Table 4-1 (Cont.) Commonly Used Snapper Command Reference

Action	Command and Description
Undo the changes made to files between a pre and a post snapshot.	To undo the changes in the subvolume from post snapshot *N'* to pre snapshot *N'*: `sudo snapper -c config_name undochange N .. N'` Note that undoing a change doesn't revert the file system to the previous snapshot but it reverts modifications made to existing files in the snapshot. This means that files created after the snapshot was taken continue to remain after an `undochange` operation. The `undochange` subcommand doesn't check data integrity for its changes. Be careful of using this command without clearly evaluating the implications of the changes that it's likely to make.

Mounting Snapper Snapshots

You can mount any snapshot generated by Snapper in the same way as any other Btrfs snapshot. You might need to correlate the snapshot volume id with the Snapper snapshot number to work out which snapshot to mount or restore.

1. Run the `snapper list` command to identify the number of the snapshot to roll back to.

   For example, to see all pre and post snapshots, to roll back to a snapshot from before a DNF package update was run:

   ```
   sudo snapper -c root list -t pre-post
   ```

   Running the previous command might produce the following output:

   ```
   Pre # | Post # | Pre Date                         | Post
   Date                         | Description                          | Userdata
   ------+--------+-----------------------------
   +------------------------------+----------------------------------+---------
      4  |      5   | Wed 25 Nov 2020 09:07:21 EST | Wed 25 Nov 2020 09:07:25
   EST | /usr/bin/dnf install vim         |
    127  |    128   | Mon 30 Nov 2020 08:25:42 EST | Mon 30 Nov 2020 08:30:57
   EST | /usr/bin/dnf update              |
   ```

   Note that the snapshot number of the pre snapshot that we intend to mount is 127 in this case.

2. Use the `btrfs subvolume list` command to obtain the subvolume ID for the Snapper snapshot.

For example, to get the subvolume ID of the snapshot with the Snapper snapshot number of 127.:

```
sudo btrfs subvolume list /|grep .snapshots.*127
```

The output of the previous command is as follows:

```
ID 521 gen 11533 top level 268 path .snapshots/127/snapshot
```

Note that the subvolume ID is listed as 521 for this snapshot.

3. Use the `mount` command with the `subvolid` option to specify the subvolume ID of the snapshot to mount:

   ```
   sudo mount -o subvolid=521 /dev/sda2 /mnt
   ```

   You can also use this information to boot into a snapshot of the root file system. See Mounting a Snapshot as the Root File System for more information.

5 Using Data Block Sharing to Copy Files

Use the `cp` command with the `--reflink` option to create lightweight copies of a file within the same subvolume of a Btrfs file system.

The `--reflink` option takes advantage of the copy-on-write mechanism to save disk space and to perform almost instantaneous copy operations. The Btrfs file system creates a new inode that shares the same disk blocks as the existing file, rather than creating a complete copy of the file's data or creating a link that points to the file's inode. The resulting file appears to be a copy of the original file, but the original data blocks aren't duplicated. If you write to one of the files after you have copied using the `--reflink` option, the Btrfs file system makes copies of the blocks before they're written to, preserving the other file's content.

To create a lightweight copy of a file named `foo` to a file named `bar`, run:

```
cp --reflink foo bar
```

The resulting file, `bar`, doesn't use any extra disk space and is created instantaneously, regardless of the size of the original file, `foo`. Disk space is used as each of the files is written to.

6

Creating Backups and Using the Btrfs Send/ Receive Feature

> **Note:**
>
> Working with the Btrfs send/receive feature requires that you boot the system by using UEK R6 or later.

The send operation compares two subvolumes and writes a description of how to convert one subvolume, the *parent* subvolume, into the other subvolume, which is the *sent* subvolume. You would usually direct the output to a file for later use or pipe it to a receive operation for immediate use.

The simplest form of the send operation writes a complete description of a subvolume, for example:

```
sudo btrfs send [-v] [-f sent_file] ... subvol
```

You can specify many instances of the `-v` option to display increasing amounts of debugging output. The `-f` option is used to save the output to a file. Note that both of these options are implicit in the following usage examples.

The following form of the send operation writes a complete description of how to convert one subvolume to another subvolume:

```
sudo btrfs send -p parent_subvol sent_subvol
```

If a subvolume such as a snapshot of the parent volume, known as a *clone source*, will be available during the receive operation from which some data can be recovered, you can specify the clone source to reduce the size of the output file:

```
sudo btrfs send [-p parent_subvol] [-c clone_src] ... subvol
```

You can specify the `-c` option for each of the clone source that exist. If you don't specify the parent subvolume, `btrfs` chooses a suitable parent from the clone sources.

Use the receive operation to regenerate the sent subvolume at a specified path, for example:

```
sudo btrfs receive [-f sent_file] mountpoint
```

Creating a Reference Backup in Preparation for Creating an Incremental Backup

The following procedure describes how to create a reference backup, which is a prerequisite to setting up an incremental backup and restore process for a subvolume by using the send/ receive feature.

1. Create a read-only snapshot of the subvolume to serve as an initial reference point for the backup.

   ```
   sudo btrfs subvolume snapshot -r /vol /vol/backup_0
   ```

2. Ensure that the snapshot has been written to disk by running the `sync` command.

   ```
   sudo sync
   ```

3. Create a subvolume or directory on a Btrfs file system as a backup area to receive the snapshot, for example, `/backupvol`.
4. Send the snapshot to `/backupvol`.

   ```
   sudo btrfs send /vol/backup_0 | btrfs receive /backupvol
   ```

   The previous command creates the `/backupvol/backup_0` subvolume.

   After creating the reference backup, you can then create incremental backups, as needed. See Creating an Incremental Backup.

Creating an Incremental Backup

The following instructions describe how to create an incremental backup by using the send/ receive feature. Note that before creating an incremental backup, you must first create a reference backup. See Creating a Reference Backup in Preparation for Creating an Incremental Backup.

To create an incremental backup:

1. Create a snapshot of the subvolume.

   ```
   sudo btrfs subvolume snapshot -r /vol /vol/backup_1
   ```

2. Ensure that the snapshot has been written to disk by running the `sync` command.

   ```
   sudo sync
   ```

3. Send only the differences between the reference backup and the new backup to the backup area.

   For example:, run

   ```
   sudo btrfs send -p /vol/backup_0 /vol/backup_1 | btrfs receive /backupvol
   ```

   Running the previous command creates the `/backupvol/backup_1` subvolume.

7

Working With a Btrfs root File System

Important:

In Oracle Linux, the Btrfs file system and all the features that are documented in this chapter are supported in the Unbreakable Enterprise Kernel (UEK) release *only*. Working with Btrfs file system features requires that you boot the system by using UEK R6 or later.

You can create a Btrfs root file system during an installation. To do so, you must boot the system by using UEK R6 or later.

Although Oracle Linux uses the default top level ID set with an ID of 5, the root file system is created as a subvolume within the top level file system on a system that's installed with Btrfs as the root file system. Therefore, when you view the subvolume list, you might see output similar to the following:

```
ID 256 gen 1591 top level 5 path boot
ID 258 gen 1591 top level 5 path root
ID 259 gen 1514 top level 5 path home
ID 262 gen 1514 top level 258 path var/lib/portables"
```

In the output, the installation `root` file system subvolume has an ID of 258. The subvolume with ID 258 (`root`) is mounted as `/`. The default subvolume (`root`) with ID 258 is mounted as the active root file system. For example, the `mount` command shows the device that's mounted as the `root` file system and indicates the subvolume ID (258):

Note that the top-level file system isn't mounted by default. To mount the top-level file system volume, use the `subvolid` option to specify the subvolume ID as 5. For example, you might run:

```
sudo mount -o subvolid=5 /dev/sda2 /mnt
```

If you list the contents of `/mnt`, you can view each of the subvolumes within the file system volume, including the `root` subvolume.

By setting the default subvolume to use at the root file system and keeping the top-level file system unmounted, we achieve cleaner separation between actively used subvolumes and underlying file system maintenance such as the storage of snapshots.

By mounting the top level file system, you can easily create a snapshot of the root file system. If you need to rollback to a snapshot, you can change the default subvolume for the file system to match the subvolume ID of the snapshot that you want to roll back to.

Creating Snapshots of the root File System

Creating snapshots of a Btrfs root file system is no different to creating a snapshot of any other subvolume, but to keep snapshots separated from the root subvolume, we store them at the top level of the file system.

1. Mount the top-level subvolume ID on a suitable mount point.

   ```
   sudo mount -o subvolid=5 /dev/sda2 /mnt
   ```

2. Create a directory in the file system to store any snapshots, if this doesn't already exist.

   ```
   sudo mkdir -p /mnt/.snapshots
   ```

3. Take a snapshot of the file system.

   ```
   sudo btrfs subvolume snapshot / /mnt/.snapshots/root_snapshot_1
   ```

4. Unmount the top level of the file system.

   ```
   sudo umount /mnt
   ```

5. Verify that the list of subvolumes includes the newly created snapshot.

   ```
   sudo btrfs subvolume list /


   ID 256 gen 1332 top level 5 path boot
   ID 258 gen 1349 top level 5 path root
   ID 259 gen 1309 top level 5 path home
   ID 261 gen 1309 top level 258 path var/lib/portables
   ID 264 gen 1348 top level 5 path .snapshots/root_snapshot_1
   ```

Mounting a Snapshot as the Root File System

To roll back changes to the system, you can mount a snapshot as the root file system by specifying its ID as the default subvolume.

1. Find the subvolume ID for the snapshot that you want to use to replace the root file system.

   For example, to get a listing of subvolumes and snapshots with their IDs, run:

   ```
   sudo btrfs subvolume list /
   ```

2. Use the `btrfs subvolume set-default` command to change the default subvolume ID to use for the `/` mount point.

   ```
   sudo btrfs subvolume set-default 264 /
   ```

3. Update the system GRUB configuration to ensure that the subvolume default isn't overwritten by a kernel boot argument.

For example, run:

```
default_kernel=$(sudo grubby --default-kernel);
sudo grubby --remove-args="rootflags=subvol=root" --update-
kernel $default_kernel
```

4. Reboot the system for the changes to take effect.

```
sudo reboot
```

5. After the system has booted, validate that the snapshot subvolume is mounted on the root file system at `/`.

   You can review the information returned by the mount command:

```
sudo mount|grep 'on / '
```

   The command might return output similar to the following:

```
/dev/sda2 on / type btrfs
(rw,relatime,seclabel,space_cache,subvolid=264,subvol=/.snapshots/root-
snapshot1)
```

   Or run `btrfs inspect-internal rootid` to return the subvolume ID for the subvolume mounted at the `/` mount point.

```
sudo btrfs inspect-internal rootid /
```

Deleting Snapshots of the root File System

Note:

A snapshot can't be deleted if it's set as the default ID for a subvolume. Deleting a snapshot while it's in use as the root file system might cause system failure and requires a hard physical reset. Before deleting a snapshot that's set as the default subvolume for the root File System, change the default ID and reboot the system, for example:

```
sudo btrfs subvolume set-default 258 /
reboot
```

To delete a snapshot, do the following:

1. Mount the top level of the file system.

   For example, run:

```
sudo mount -o subvolid=5 /dev/sda2 /mnt
```

2. Use the `btrfs subvolume delete` command to delete the snapshot.

   ```
   sudo btrfs subvolume delete /mnt/.snapshots/root-snapshot1
   ```

3. Unmount the top level of the file system:

   ```
   sudo umount /mnt
   ```

   The list of subvolumes relative to the root file system now doesn't include `snapshots/root-snapshot1`.

   ```
   sudo btrfs subvolume list /
   ```

   ```
   ID 256 gen 1332 top level 5 path boot
   ID 258 gen 1349 top level 5 path root
   ID 259 gen 1309 top level 5 path home
   ID 261 gen 1309 top level 258 path var/lib/portables
   ```

8 Creating Swap Files on a Btrfs File System

Swap space is used in Oracle Linux when the amount of physical memory (RAM) is full. If the system needs more memory resources, and the RAM is full, inactive pages in memory are moved to the swap space. Although swap space is helpful for systems with a small amount of RAM, don't use swap space as a replacement for more RAM. You can allocate swap space to a dedicated swap partition, which is the recommended method. Or, you can use a swap file; or, you can combine the use of swap partitions and swap files.

Swap files in Btrfs are supported with the following limitations:

- A swap file can't be on a snapshotted subvolume. Instead, we recommend that you create a subvolume on which to place the swap file.
- Btrfs doesn't support swap files on file systems that span several devices.

The following are step-by-step instructions for creating a swap file in Btrfs. Before creating the new swap file, calculate the size of the swap file in MB. Then, multiply that number by 1024 to find the number of blocks the file requires. For example, the block size of a 64 MB swap file is 65536.

1. Use the `dd` command to create an empty file to use as the swap file.

   ```
   sudo dd if=/dev/zero of=/swapfile bs=1024 count=65536
   ```

2. Set up the swap file by using the `mkswap` command:

   ```
   sudo mkswap /swapfile
   ```

3. Change the permissions on the file so that it's not world readable:

   ```
   sudo chmod 0600 /swapfile
   ```

4. Enable the swap file at boot time by editing the `/etc/fstab` file.

   The `/etc/fstab` file must contain a line similar to the following:

   ```
   /swapfile swap swap defaults 0 0
   ```

5. Regenerate the mount units and register the new configuration in the `/etc/fstab` file.

   ```
   sudo systemctl daemon-reload
   ```

6. Activate the new swap file by using the `swapon` command:

   ```
   sudo swapon /swapfile
   ```

   Running the `swapon` command activates the new swap file immediately.

You can run the following commands to test whether the new swap file was successfully created and by inspecting the active swap space:

```
sudo cat /proc/swaps
sudo free -h
```

9

Converting an Ext File System to a Btrfs File System

You can use the `btrfs-convert` utility to convert an `ext` file system to a Btrfs file system. The utility preserves an image of the original file system in a snapshot named `ext`*`N`*`_saved`, such as `ext4_saved`. With this snapshot, you can roll back the conversion, even if you have changed the `btrfs` file system.

Note that you can't convert a root file system or a bootable partition, such as `/boot`, to Btrfs.

Note:

The conversion to Btrfs isn't supported on the Arm (aarch64) platform. If you're running Oracle Linux on the aarch64 platform, you can migrate data from one file system to another file system.

Converting an Ext File System to a Btrfs File System

You can perform the conversion of an Ext file system to Btrfs on file systems that aren't used as the root partition or the boot partition.

Caution:

Although the conversion tooling stores an image of the Ext file system metadata that you can use to roll back to soon after converting to Btrfs, ensure that you have a full backup of the file system in case the conversion fails or the rollback image is removed and you want to revert to using the Ext file system.

To convert an Ext file system other than the root file system to Btrfs:

1. Unmount the file system.

   ```
   sudo umount mountpoint
   ```

2. Run the correct version of `fsck` (for example, `fsck.ext4`) on the underlying device to check and correct the integrity of file system.

   ```
   sudo fsck.extN -f device
   ```

3. Convert the file system to a btrfs file system.

   ```
   sudo btrfs-convert device
   ```

4. Edit the file `/etc/fstab`, and change the file system type of the file system to `btrfs`. For example:

   ```
   /dev/sdb               /myfs          btrfs    defaults  0 0
   ```

5. Mount the converted file system on the old mount point.

   ```
   sudo mount device mountpoint
   ```

6. Verify the file system and review the Ext backup image.

   Check that the Btrfs file system is correctly initialized and that the data is still available on the file system. For example:

   ```
   sudo btrfs filesystem usage mountpoint
   ls -lah mountpoint
   ```

   Note that the file system includes a subvolume labeled `ext2_saved`:

   ```
   sudo btrfs subvolume list
   ```

   The subvolume contains an image of the Ext file system that can be used to roll the file system back to its original Ext file system type. You can roll the file system back by running:

   ```
   sudo umount mountpoint
   sudo btrfs-convert -r device
   ```

   The file system is rolled back to its original Ext file system type.

   The Ext file system image takes up disk space and might no longer be usable or required after you perform other disk or file system operations on the Btrfs file system. You can remove this backup by running:

   ```
   btrfs subvolume delete mountpoint/ext2_saved
   ```