Oracle Linux 8

Managing the XFS File System

G26933-01
July 2025

Oracle Linux 8 Managing the XFS File System,

G26933-01

Contents

7 Backing Up and Restoring an XFS File System

Preface

Oracle Linux 8: Managing the XFS File System provides information about managing the XFS 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 XFS File System

XFS is a high-performance journaling file system that was initially created by Silicon Graphics, Inc. for the IRIX OS and then later ported to Linux. The parallel I/O performance of XFS provides high scalability for I/O threads, file system bandwidth, file, and file system size, even when the file system spans many storage devices.

XFS can be used with the `root` (`/`) or `boot` file systems on Oracle Linux.

XFS has many features that are suitable for deployment in an enterprise-level computing environment that requires the implementation of large file systems:

- Implements journaling for metadata operations.

  Journaling guarantees the consistency of the file system following loss of power or a system failure. XFS records file system updates asynchronously to a circular buffer (the *journal*) before it can commit the actual data updates to disk. The journal can be stored either internally in the data section of the file system, or externally on a separate device to reduce contention for disk access. If the system fails or loses power, it reads the journal when the file system is remounted, and replays any pending metadata operations to ensure the consistency of the file system. The speed of this recovery doesn't depend on the size of the file system.

- Is internally partitioned into allocation groups, which are virtual storage regions of fixed size.

  Any files and directories that you create can span several allocation groups. Each allocation group manages its own set of inodes and free space independently of other allocation groups to provide both scalability and parallelism of I/O operations. If the file system spans many physical devices, allocation groups can optimize throughput by taking advantage of the underlying separation of channels to the storage components.

- Is an extent-based file system.

  To reduce file fragmentation and file scattering, each file's blocks can have variable length extents, where each extent consists of one or more contiguous blocks. XFS's space allocation scheme is designed to efficiently identify free extents that it can use for file system operations. XFS doesn't allocate storage to the holes in sparse files. If possible, the extent allocation map for a file is stored in its inode. Large allocation maps are stored in a data structure maintained by the allocation group.

- Includes the reflink and deduplication features, which provides the following benefits:
  - Each copy can have different file metadata (permissions, and so on) because each copy has its own distinct inode. Only the data extents are shared.
  - The file system ensures that any write causes a copy-on-write, without applications requiring to do anything special.
  - Changing one extent continues to permit all the other extents to remain shared. In this way, space is saved on a per-extent basis. Note, however, that a change to a hard-linked file does require a new copy of the entire file.

- Implements *delayed allocation*

  To reduce fragmentation and increase performance, XFS reserves file system blocks for data in the buffer cache, and allocates the block when the OS flushes that data to disk.

- XFS recognizes extended attributes for files.

  The size of each attribute's value can be up to 64 KB, and each attribute can be allocated to either a root or a user namespace.

- Direct I/O in XFS implements high throughput, non-cached I/O.

  XFS performs DMA directly between an application and a storage device, using the full I/O bandwidth of the device.

- Includes the snapshot facilities that volume managers, hardware subsystems, and databases provide.

  Use the `xfs_freeze` command to suspend and resume I/O for an XFS file system. See Freezing and Unfreezing an XFS File System.

- XFS enables user, group, and project disk quotas on block and inode usage that are initialized when the file system is mounted. Project disk quotas enable you to set limits for individual directory hierarchies within an XFS file system without regard to which user or group has write access to that directory hierarchy.

To maximize throughput for XFS file systems that you create on an underlying striped software or hardware based array, you can use the `su` and `sw` arguments to the `-d` option of the `mkfs.xfs` command to specify the size of each stripe unit and the number of units per stripe. XFS uses the information to align data, inodes, and journal appropriately for the storage. On `lvm` and `md` volumes and some hardware RAID configurations, XFS can automatically select the best stripe parameters for you.

To defragment individual files in an active XFS file system, you can use the `xfs_fsr` command. See Defragmenting an XFS File System.

To grow an XFS file system, you can use the `xfs_growfs` command. See Growing an XFS File System.

To back up and restore a live XFS file system, you can use the `xfsdump` and `xfsrestore` commands. See Backing Up and Restoring an XFS File System.

For more information about XFS, see https://xfs.wiki.kernel.org/.

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

2
Installing XFS Packages

The XFS file system is managed on Oracle Linux by installing the `xfsprogs` and `xfsdump` packages.

XFS is the default file system available during the installation of Oracle Linux. Therefore, most required packages are available without any further installation steps. However, on some systems, where XFS isn't selected as the base file system and where the operating system might be cut back to create a slimmer installation, you might need to install the packages manually.

The XFS packages are available in the core Oracle Linux repositories and channels on the Oracle Linux yum server and on ULN.

1. Use the `dnf` command to install the `xfsprogs` and `xfsdump` packages.

   ```
   sudo dnf install -y xfsprogs xfsdump
   ```

2. Optionally, you can install XFS development packages:

   ```
   sudo dnf install -y xfsprogs-devel
   ```

3

Setting Up and Administering an XFS File System

Understand how to perform common administrative tasks such as creating, mounting, and resizing an XFS file system.

Creating and Mounting an XFS File System

Use the `mkfs.xfs` command to create an XFS file system on a block device, such as a partition, an LVM volume, a disk, or a similar hardware device.

You create XFS file systems by using the `mkfs.xfs` command. The default options for the command are appropriate for most common use cases. For more information, see the `mkfs.xfs(8)` manual page.

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

   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. To create an XFS file system on a device, run:

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

   For example, to format a file system on the device */dev/sdb1* with the default options, run:

   ```
   sudo mkfs.xfs /dev/sdb1
   ```

   Use the appropriate flags to set any custom options on the file system if you need to do so. See Notable XFS Feature Options.

3. Mount the file system.

   For example, you can run:

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

4. Validate and view information about the file system using the `xfs_info` command.

For example, you can run:

```
xfs_info /mnt
```

The command returns information including the device path and the file system geometry. See the `xfs_info(8)` manual page for more information.

Notable XFS Feature Options

You can set some notable options when creating an XFS file system. Options are also described in more detail in the `mkfs.xfs(8)` manual page.

Table 3-1 Notable XFS Feature Options

Option	Description
`lazy-count`	Lazy counters improve I/O performance for application workloads that are metadata intensive because the free-space and inode counters are maintained in parts of the file system other than the superblock. Lazy counters are enabled by default. However, if required, you can disable them by specifying the `-l lazy-count=0` option to the `mkfs.xfs` command.
`logdev`	Using an external log device for the XFS journal can reduce disk contention and improve the file system's I/O throughput. A suitable host device for the journal is a solid-state drive (SSD) device or a RAID device with a battery-backed, write-back cache. The default location for an XFS journal is on the same block device as the data. To reserve an external journal with a specified size when you create an XFS file system, specify the `-l logdev=`*`device`*`,size=`*`size`* option to the `mkfs.xfs` command. If you omit the `size` parameter, `mkfs.xfs` selects a journal size based on the size of the file system. To mount the XFS file system so that it uses the external journal, specify the `-o logdev=`*`device`* option to the `mount` command.

Table 3-1 (Cont.) Notable XFS Feature Options

Option	Description
`bigtime`	The `bigtime` option extends the maximum recorded inode timestamps from 2038 to 2486. This feature requires a kernel later than 5.10 and after it's enabled the file system can no longer be mounted on systems running an earlier kernel. This feature is disabled by default on Oracle Linux 8, because the functionality isn't available for RHCK and UEK R6. To use this feature, upgrade to the latest UEK R7. Older XFS file systems might be formatted without this option enabled. To enable the option on an existing file system, use the `xfs_admin -O bigtime=1` command. See Changing XFS File System Feature Options for more information.
`inobtcount`	The `inobtcount` option reduces mount time on large file systems by speeding up metadata space reservation calculations. This feature requires a kernel later than 5.10 and after it's enabled the file system can no longer be mounted on systems running an earlier kernel. This feature is disabled by default on Oracle Linux 8, because the functionality isn't available for RHCK and UEK R6. To use this feature, upgrade to the latest UEK R7. Older XFS file systems might be formatted without this option enabled. To enable the option on an existing file system, use the `xfs_admin -O inobtcount=1` command. See Changing XFS File System Feature Options for more information.
`reflink`	The `reflink` feature takes advantage of the copy-on-write functionality within XFS to create lightweight copies of a file or directory within an XFS file system. This feature is enabled by default, unless CRC checks are disabled or the XFS file system is used with a DAX mount option. See Copying Files By Using Shared Data Blocks for more information.

Table 3-1 (Cont.) Notable XFS Feature Options

Option	Description
`su`, `sw`	The `su` option, for stripe unit, and the `sw` option, for stripe width, can be specified when formatting an XFS file system for a RAID device or a striped logical volume. Specify the `-d su=<size>,sw=<width>` option to the `mkfs.xfs` command when you need to specify particular device geometry. The stripe unit is specified as a multiple of the file system block size. The stripe width is expressed as a multiplier of the stripe unit value and is usually the same as the number of members in the striped logical volume configuration or number of disks in the RAID device.

Changing XFS File System Feature Options

Use the `xfs_admin` command to update or change XFS file system features on an unmounted device.

The `xfs_admin` command can change feature options on an unmounted XFS file system. For example, you can:

- Enable or disable lazy counters
- Enable or disable other feature options such as `bigtime` and `inobtcount`.
- Change the file system UUID
- Change the file system label

For more information, see the `mkfs_admin(8)` manual page.

> **Note:**
>
> You can't change a mounted XFS file system.

- View and apply a new label to the file system.

  To display the existing label, run:

  ```
  sudo xfs_admin -l /dev/sdb
  ```

  To apply a new label, run:

  ```
  sudo xfs_admin -L "VideoRecords" /dev/sdb
  ```

> **Note:**
>
> The label can be a maximum of 12 characters in length.

- View, generate, or clear the UUID of the file system.

  To display the existing UUID, run:

  ```
  sudo xfs_admin -u /dev/sdb
  ```

  To generate a new UUID, run:

  ```
  sudo xfs_admin -U generate /dev/sdb
  ```

  To clear the UUID altogether, run:

  ```
  sudo xfs_admin -U nil /dev/sdb
  ```

- Disable or enable lazy counters.

  To disable lazy counters, run:

  ```
  sudo xfs_admin -c 0 /dev/sdb
  ```

  To enable lazy counters, run:

  ```
  sudo xfs_admin -c 1 /dev/sdb
  ```

- Enable `bigtime` or `inobtcount`.

  These features are disabled by default on Oracle Linux 8, because the functionality isn't available for RHCK and UEK R6. To use these features, upgrade to the latest UEK R7. Note that you can't mount the file system on earlier kernels after you perform this operation.

  To enable `bigtime` on a file system formatted without this option, run:

  ```
  sudo xfs_admin -O bigtime=1 /dev/sdb
  ```

  To enable `inobtcount` on a file system formatted without this option, run:

  ```
  sudo xfs_admin -O inobtcount=1 /dev/sdb
  ```

Growing an XFS File System

Use the `xfs_growfs` command to increase the size of an XFS file system if space is available on the underlying devices to accommodate the change.

> **Note:**
>
> You can't grow an unmounted XFS file system. Also, no command exists to shrink an XFS file system.

The `xfs_growfs` command doesn't have any effect on the layout or size of the underlying devices.
If you're using Logical Volume Manager (LVM), you can use the `vgextend` command to increase the storage that's available to an LVM volume group and `lvextend` to increase the size of the logical volume that contains the file system.

You can't use the `parted` command to resize a partition that contains an XFS file system. You must instead re-create the partition with a larger size and restore its contents from a backup if you deleted the original partition or from the contents of the original partition if you didn't delete it to free up disk space.

You can either increase the size of the file system to a specified size expressed in file system blocks, or you can increase the size to use the maximum available space on the device.

- Increase the size of the file system to a specified size by using the `xfs_growfs -D` <*size*> command.

  For example, you would increase the size of `/mnt` to 4 TB, assuming a block size of 4 KB, as follows:

  ```
  sudo xfs_growfs -D 1073741824 /mnt
  ```

- Increase the size of the file system to the maximum size that the underlying device supports by using the `xfs_growfs -d` command.

  To increase the size of the file system to the maximum size that the underlying device supports, specify the `-d` option:

  ```
  sudo xfs_growfs -d /mnt
  ```

Checking and Repairing an XFS File System

Use the `xfs_repair` command to check file system consistency and to perform a file system repair.

If you can't mount an XFS file system, you can use the `xfs_repair` command to check its consistency and repair any issues. The command replays the journal log and shows any inconsistencies that might have resulted from the file system not being cleanly unmounted. Unless the file system has an inconsistency, you don't need to perform any repairs, as the journal is replayed every time that you mount an XFS file system.

The following procedure provides the usual steps to take when checking and repairing a file system with a corrupted journal.

1. Backup the file system metadata using the `xfs_metadump` command.

   A metadata image can be helpful in root cause analysis and for identification of software bugs. For example, you can run:

   ```
   sudo xfs_metadump /dev/sdb /tmp/sdb-metadata-image
   ```

   Only run this command on an unmounted or frozen file system, or the dump file could be corrupted or inconsistent. See also Freezing and Unfreezing an XFS File System.

   You can compress the metadata image file and use it when working with a support team to resolve file system issues. You can use the `xfs_mdrestore` command to restore a metadata image to a file system. See the `xfs_metadump(8)` and `xfs_mdrestore(8)` manual pages for more information.

2. Backup file system data, if possible.

   If you can mount the file system and you don't have a suitable backup, you can use the `xfsdump` command to back up the existing file system data. However, note that the command might fail if the file system's metadata has become corrupted.

   For example, to backup the file system mounted at */mnt* onto a device at */dev/sdc*, you could run:

   ```
   sudo xfsdump -l 0 -L "Full backup of /mnt $(date)" -f /dev/sdc /mnt
   ```

   See Backing Up and Restoring an XFS File System for more information.

3. Check file system integrity.

   XFS replays the journal every time the file system is mounted. Therefore. the first file system integrity check is to decide whether the journal is corrupt. If the journal is corrupt, performing a dry run of the repair can help show any corruption and what can be done to remedy the issue.

   a. Replay the file system journal by mounting and unmounting the file system. For example, run:

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

      If you can't mount the file system and a structure needs cleaning error is returned, the journal is corrupt.

   b. Perform a file system consistency check to review the changes that a repair would perform.

      ```
      sudo xfs_repair -n /dev/sdb
      ```

      The `xfs_repair -n` command displays output to indicate changes that would be made to the file system in the case where it would need to complete a repair operation. No changes are made to the file system by running this command.

4. Use the `xfs_repair` command to perform the file system repair.

   a. Run the `xfs_repair -v` command to get verbose output while the repair runs.

For example, you can run:

```
sudo xfs_repair -v /dev/sdb
```

b. Reset the journal as a final resort.

If the journal log has become corrupted, you can reset the log by specifying the `-L` option to `xfs_repair`. For example:

```
sudo xfs_repair -L /dev/sdb
```

> **NOT_SUPPORTED:**
>
> Resetting the log can leave the file system in an inconsistent state, resulting in data loss and data corruption. Unless you're experienced with debugging and repairing XFS file systems by using the `xfs_db` command, we recommend that you instead re-create the file system and restore its contents from a backup.

5. Mount and validate the file system.

For example, run:

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

If you can't mount the file system or you don't have a suitable backup, running the `xfs_repair` command is the only viable option, unless you're experienced in using the `xfs_db` command.

`xfs_db` provides an internal command set for debugging and repairing an XFS file system manually. The commands enable you to perform scans on the file system, and display its data structures. If you specify the `-x` option to enable expert mode, you can change the data structures.

```
sudo xfs_db [-x] device
```

For more information, see the `xfs_db(8)` and `xfs_repair(8)` manual pages, and run the `help` command within `xfs_db`.

Defragmenting an XFS File System

Use the `xfs_fsr` command to defragment whole XFS file systems or individual files within an XFS file system.

Fragmentation occurs when a file is stored on the disk in a non-contiguous manner. Fragmentation can be caused by several factors, including:

- **Frequent Creation and Deletion of Small Files:** When small files are created and deleted often, the free space on the disk becomes scattered across many locations. This dispersed free space can cause new or large files to be stored in non-contiguous chunks.
- **High Levels of File Modification and Growth:** As files are changed and grow, their data might not fit into the originally allocated space. This can lead to fragmentation as the file's data is spread across different parts of the disk.

- **Copy-on-Write Operations:** Features that take advantage of copy-on-write functionality can contribute to fragmentation. For example, when a file is reflinked and its contents are changed, the extents can be split or altered, which can further fragment the file system.

Fragmentation can impact performance resulting in slower access time and increased reflink creation time.

Defragmentation of the file system is managed using the `xfs_fsr` command. For more information, see the `xfs_fsr(8)` manual page.

- To defragment an individual file, use the following command to specify the name of the file as the argument to `xfs_fsr`.

  ```
  sudo xfs_fsr file_path
  ```

- To defragment all mounted XFS file systems, run the `xfs_fsr` command without specifying a path.

  ```
  sudo xfs_fsr
  ```

  Running the `xfs_fsr` command without any options defragments all the mounted and writeable XFS file systems that are listed in `/etc/mtab`. The defragmentation process runs for two hours and defragments the top 10 percent of files with the greatest number of extents. After two hours, the command records its progress in the `/var/tmp/.fsrlast_xfs` file. If you run the command again, the process is resumed from that point.

4 Copying Files By Using Shared Data Blocks

Use the `cp` command with the `--reflink` option to create lightweight copies of a file within an XFS file system.

Note the reflink feature is enabled by default in Oracle Linux when formatting by using the `mkfs.xfs` command.

The `--reflink` option takes advantage of the copy-on-write mechanism to save disk space and to perform almost instantaneous copy operations. The XFS 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 file system makes copies of the blocks before they're written to, preserving the other file's content.

For more information on how the reflink feature works, see https://blogs.oracle.com/linux/xfs-data-block-sharing-reflink.

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.

5

Freezing and Unfreezing an XFS File System

Use the xfs_freeze command to stop write operations when taking hardware-based snapshots of an XFS file system.

> **Note:**
>
> You don't need to explicitly suspend write operations if you use the `lvcreate` command to take an LVM snapshot.

You can also use the `xfs_freeze` command with `btrfs`, `ext3`, and `ext4` file systems.

For more information, see the `xfs_freeze(8)` manual page.

1. To freeze an XFS file system, use the `-f` option with the `xfs_freeze` command:

   ```
   sudo xfs_freeze -f /mnt
   ```

2. To unfreeze an XFS file system, use the `-u` option with the `xfs_freeze` command:

   ```
   sudo xfs_freeze -u /mnt
   ```

6

Managing Quotas on an XFS File System

Use the `xfs_quota` tool to manage quotas on an XFS file system.

Use XFS quotas to set limits on disk space allocation, in terms of blocks, and file creation, in terms of inodes. Quotas can be used to monitor and restrict disk usage at various levels, including individual users, groups, and projects.

Project level quotas can be configured to map directory level hierarchies to particular projects defined in the configuration files at `/etc/projects` and `/etc/projid`. For more information on these configuration files, see the `projects(5)` and `projid(5)` manual pages.

For more information, see the `xfs_quota(8)` manual page.

Enabling Disk Quotas on File Systems

Disk quota types are enabled at mount by specifying a mount option.

Mount Option	Description
`gqnoenforce`	Enable group quotas. Report usage, but don't enforce usage limits.
`gquota`	Enable group quotas and enforce usage limits.
`pqnoenforce`	Enable project quotas. Report usage, but don't enforce usage limits.
`pquota`	Enable project quotas and enforce usage limits.
`uqnoenforce`	Enable user quotas. Report usage, but don't enforce usage limits.
`uquota`	Enable user quotas and enforce usage limits.

Mounting a File System With Quotas Enabled

Mount a file system from the command line with a quota type enabled.

If a file system doesn't have a system mount configured in `/etc/fstab` or the entry doesn't include a quota option, you can enable the quota option when you mount the file system from the command line.

- Mount the file system from the command line using the `-o <quotatype>` option to enable the specified quota.

For example, to enable user quotas, run:

```
sudo mount -o uquota /dev/sdb1 /mnt
```

Replace `uquota` with `uqnoenforce` to enable usage reporting without enforcing any limits.

Editing System Mounts to Use Quotas

Edit /etc/fstab to add quota options to a file system entry, to enable quotas when the file system is remounted.

1. Install the `quota` package on the system, if not already installed.

   ```
   sudo dnf install -y quota
   ```

2. Add the quota type options to the file system's `/etc/fstab` entry.

   For example, to add the `uquota` and `gquota` types, you can add:

   ```
   /dev/sdb1     /home      xfs     defaults,uquota,gquota    0 0
   ```

3. Remount the file system.

   ```
   sudo mount -o remount /home
   ```

Displaying Block Usage Quota Information

Use the `xfs_quota` command to display quota reports and to see file system usage.

To perform this task, you use the specify the `xfs_quota` with the `-x` option to enter expert mode. You can then run quota reporting tasks as required.

- Use the `report` subcommand to see an overview of usage and limits for all users, groups, and projects.

  For example, to display the block usage limits and the current usage for all user quotas in the file system mounted at `/mnt`, use the `report -ubh` subcommand:

  ```
  xfs_quota -x -c 'report -ubh' /mnt
  ```

  Output similar to the following is displayed:

  ```
  User quota on /mnt (/dev/sdc1)
                             Blocks
  User ID      Used   Soft   Hard Warn/Grace
  ---------- --------------------------------
  root            0      0      0  00 [------]
  guest         6.1M   200M   250M  00 [------]
  ```

  You can use the `-i` option to view a summary for inode usage. The `-u`, `-g`, and the `-p` options can be used to view summaries for user, group, and project quotas. The `-h` option returns information in human-readable format.

- Use the `df -h` subcommand to view free and used counts for blocks and inodes.

  ```
  sudo xfs_quota -c 'df -h' /mnt
  ```

  By default, the output displays the block usage:

  ```
  Filesystem     Size   Used  Avail Use% Pathname
  /dev/sdc1    200.0G  32.2M  20.0G   1% /mnt
  ```

  Use the `-i` option to display inode usage:

  ```
  sudo xfs_quota -c 'df -ih' /mnt
  ```

  ```
  Filesystem   Inodes   Used   Free Use% Pathname
  /dev/sdc1     21.0m      4  21.0m   1% /mnt
  ```

Setting Quota Limits

Set soft and hard quota limits using the `xfs_quota` command.

You can set quota limits on inode count and on block usage, effectively limiting the number of files or the amount of disk space that a user, group, or project can create or use.

XFS lets you set a soft limit and a hard limit. The soft limit is used to provide a notification when the quota is exceeded, but to avoid enforcing. A hard limit stops the creation of any more files, or the allocation of any further disk space.

To set limits, you use the specify the `xfs_quota` with the `-x` option to enter expert mode. You can then run the `limit` subcommand to set limits as required.

- Set quota limits for a user.

  Use the `limit` subcommand to set soft and hard limits for block and inode usage by an individual user, for example:

  ```
  sudo xfs_quota -x -c 'limit bsoft=200m bhard=250m isoft=200 ihard=250
  guest' /mnt
  ```

  Note that this command requires that you have mounted the file system with user quotas enabled.

- Set quota limits for a group.

  To set limits for a group on an XFS file system that you have mounted with group quotas enabled, specify the `-g` option to `limit`:

  ```
  sudo xfs_quota -x -c 'limit -g bsoft=5g bhard=6g devgrp' /mnt
  ```

Setting Up Project Quotas

Configure project quotas to apply a quota to individual directory hierarchies.

Project quotas can be set on individual directory hierarchies, which are known as *managed trees*. Each managed tree is uniquely identified by a *project ID* and an optional *project name*. Projects are defined in the `/etc/projects` and `/etc/projid` configuration files. For more information, see the `projects(5)` and `projid(5)` manual pages.

The ability to control the disk usage of a directory hierarchy is useful if you don't otherwise want to set quota limits for a privileged user, for example, `/var/log`, or if many users or groups have write access to a directory, for example, `/var/tmp`.

To define a project and set quota limits for it:

1. Mount the file system with project quotas enabled.

   ```
   sudo mount -o pquota device mountpoint
   ```

   For example, to enable project quotas for the file system mounted at `/mnt`, you would use the following command:

   ```
   sudo mount -o pquota /dev/sdc1 /mnt
   ```

2. Define a unique project ID for the directory hierarchy in the `/etc/projects` file.

   ```
   echo project_ID:mountpoint/directory | sudo tee -a /etc/projects
   ```

   For example, you would set a project ID of 51 for the directory hierarchy `/mnt/testdir` as follows:

   ```
   echo 51:/mnt/testdir | sudo tee -a /etc/projects
   ```

3. Create an entry in the `/etc/projid` file that maps a project name to the project ID.

   ```
   echo project_name:project_ID | sudo tee -a /etc/projid
   ```

   For example, you would map the project name `testproj` to the project with ID 51 as follows:

   ```
   echo testproj:51 | sudo tee -a /etc/projid
   ```

Setting Project Quota Limits

Use the `xfs_quota` command to configure the project quota on a directory tree and then to set soft and hard limits on blocks and inodes.

The instructions that follow here assume that you have already mounted the file system using the `pquota` option and have created a project ID in `/etc/projects`. See Setting Up Project Quotas.

To perform this task, you use the specify the `xfs_quota` with the `-x` option to enter expert mode. You can then run quota configuration tasks as required.

For more information, see the `projects(5)`, `projid(5)`, and `xfs_quota(8)` manual pages.

1. Initialize a project for a managed tree at a specified mountpoint.

   Use the `project` subcommand of `xfs_quota` to define a managed tree in the XFS file system for the project.

   ```
   sudo xfs_quota -x -c 'project -s project_name' mountpoint
   ```

   For example, you would define a managed tree in the `/mnt` file system for the project `testproj`, which corresponds to the directory hierarchy `/mnt/testdir`, as follows:

   ```
   sudo xfs_quota -x -c 'project -s testproj' /mnt
   ```

2. Use the `limit` subcommand with the `-p` option to set limits on the disk usage of the project.

   ```
   sudo xfs_quota -x -c 'limit -p arguments project_name' mountpoint
   ```

   For example, to set a hard limit of 10 GB of disk space for the project `testproj`, you would use the following command:

   ```
   sudo xfs_quota -x -c 'limit -p bhard=10g testproj' /mnt
   ```

7

Backing Up and Restoring an XFS File System

The `xfsdump` package contains the `xfsdump` and `xfsrestore` utilities. The `xfsdump` command examines the files in an XFS file system, identifies files that need to be backed up, and copies them to the storage medium. Any backups that you create by using the `xfsdump` command are portable between systems with different endian architectures. The `xfsrestore` command restores a full or incremental backup of an XFS file system. You can also restore individual files and directory hierarchies from backups.

> **Note:**
>
> Unlike an LVM snapshot, which immediately creates a sparse clone of a volume, `xfsdump` takes time to make a copy of the file system data.

Backups, known as dump sessions, are recorded in an inventory database at `/var/lib/xfsdump/inventory`. The inventory makes it easy to see what backups have been taken, at which points in time, and to which devices. You can view the inventory by either using the `xfsdump -I` or `xfsrestore -I` commands. To interactively prune the inventory, you can use the `xfsinvutil -i` command.

For more information, see the `xfsdump(8)`, `xfsrestore(8)`, and `xfsinvutil(8)` manual pages.

- Creating a Backup of an XFS File System
- Restoring an XFS File System From Backup
- Cloning an XFS File System

Creating a Backup of an XFS File System

Use the `xfsdump` command to create a backup of an XFS file system on a device such as a tape drive or in a backup file on a different file system.

A backup can span several physical media that are written on the same device. Also, you can write several backups to the same medium. Note that you can write only a single backup to a file. The command doesn't overwrite existing XFS backups that are found on physical media. If you need to overwrite any existing backups, you must use the appropriate command to erase a physical medium.

For more information, see the `xfsdump(8)` manual page.

- To create a full backup of an XFS file system, use the `xfsdump -l 0` command.

  For example, the following command writes a level 0 (base) backup of the XFS file system (`/boot`) to the device, `/dev/st0`, and assigns a session label to the backup:

  ```
  sudo xfsdump -l 0 -L "Backup level 0 of /boot $(date)" -f /dev/st0 /boot
  ```

Note that the dump destination needn't be a device, as in the example. You can alternatively specify a file path to store dump files in a directory on a file system. For example, to create an XFS dump file at `/backups/boot.0.xfsdump`, run:

```
sudo xfsdump -l 0 -L "Backup level 0 of /boot $(date)" -f /backups/
boot.0.xfsdump /boot
```

- Make incremental backups using the `xfsdump -l` command, but specify the increment level that you require.

  Make incremental dumps that are relative to an existing backup by using the same command but change the level to indicate the increment level. For example, to create an incremental backup with all the changes after the base (level 0) backup, set the level to 1:

  ```
  sudo xfsdump -l 1 -L "Backup level 1 of /boot $(date)" -f /dev/st0 /boot
  ```

  A level 1 backup records only file system changes after the level 0 backup, a level 2 backup records only the changes after the latest level 1 backup, and so on up to level 9. Note that the previous level backup must exist to perform an incremental backup.

- Resume an interrupted backup by using the `xfsdump -R` command.

  If you interrupt a backup by typing `Ctrl-C` and you didn't specify the `-J` option (suppress the dump inventory) to `xfsdump` , you can resume the dump later by specifying the `-R` option, for example:

  ```
  sudo xfsdump -R -l 1 -L "Backup level 1 of /boot $(date)" -f /dev/st0 /boot
  ```

  The backup session label from the earlier interrupted session is overridden.

Restoring an XFS File System From Backup

Use the `xfsrestore` command to find out information about the backups you have made of an XFS file system or to restore data from a backup.

- Review all available backups by using the `xfsrestore -I` command to view the backup inventory.

  The `xfsrestore -I` command displays information about the available backups, including the session ID and session label.

- Restore a full backup to a target location by using the `xfsrestore` command and specifying either the session ID, by using the `-S` option, or session label, by using the `-L` option, for the backup.

  For example, to restore an XFS file system from a level 0 backup by specifying the session ID, you would use the following command:

  ```
  sudo xfsrestore -f /dev/st0 -S c76b3156-c37c-5b6e-7564-a0963ff8ca8f /mnt
  ```

- Cumulatively restore all data starting from a level 0 backup and working through all available incremental backups, by using the `xfsrestore -r` command.

Specify the `-r` option to cumulatively recover all the data from a level 0 backup, and higher-level backups that are based on that backup:

```
sudo xfsrestore -r -f /dev/st0 -v silent /mnt
```

This command searches for backups in the archive, based on the level 0 backup, and then prompts you to select whether you want to restore each backup, in turn. After restoring the selected backup, the command exits. Note that you must run this command several times, first selecting to restore the level 0 backup, and then later higher-level backups, including the most recent backup that you require to restore the file system data.

> **Note:**
>
> After completing a cumulative restoration of an XFS file system, delete the `housekeeping` directory that the `xfsrestore` command creates in the destination directory.

- Extract a single file or subdirectory from a backup by using the `xfsrestore -s` command.

  For example, to recover the contents of `/var/log` from the backup into the target directory at `/tmp/restored_logs`, you can do:

  ```
  mkdir /tmp/restored_logs
  sudo xfsrestore -s var/log -f /backups/root.0.xfsdump /tmp/restored_logs/
  ```

- Use the `xfsrestore -i` command to run the restore command in interactive mode.

  The interactive restore tool provides subcommands that can be run within the restore shell to navigate around the dump session and to add or remove items that you want to finally extract to the provided location:

  - `cd`: change directory within the dump session.
  - `ls`: list the contents of the current directory in the dump session.
  - `add`: add a file or directory from the current directory to the queue of items to extract.
  - `delete`: delete a file or directory from the queue of items to extract.
  - `extract`: extract the items in the queue to the target location.

  For example to enter interactive mode to recover files from a dump file at `/backups/boot.0.xfsdump` and to extract to the target location at `/tmp/restore_boot`, run:

  ```
  sudo xfsrestore -i -f /backups/boot.0.xfsdump /tmp/restore_boot
  ```

Cloning an XFS File System

Combine the `xfsdump` and `xfsrestore` commands to copy the contents of one XFS file system to another.

This approach to cloning the file system can be performed on a live and mounted file system and is quicker than a block copy of the device. Furthermore, by using piped commands, you can extend this approach to use SSH to restore to a remote system.

- Combine the `xfsdump` and `xfsrestore` commands by using the `-J` option to copy the entire contents of on XFS file system to another.

  The `-J` option suppresses the usual dump inventory housekeeping that the commands perform. for example:

  ```
  sudo xfsdump -J - /mnt | xfsrestore -J - /mnt_clone
  ```

- Use SSH to perform the cloning operation to a remote system.

  For example, you could run a command similar to:

  ```
  sudo xfsdump -J - /home | ssh backup_user@standby.example.com "sudo
  xfsrestore -J - /home"
  ```