Миграция с ext4 на Btrfs: пошагово

1. Введение: зачем переходить на Btrfs

Btrfs — это современная файловая система с поддержкой *copy-on-write*, снапшотов, встроенного сжатия и расширенных возможностей управления данными. В отличие от ext4, Btrfs предлагает более гибкие механизмы хранения, защиты и восстановления, что делает её идеальной для современных сценариев эксплуатации.

✨ **Преимущества Btrfs**

- **Снапшоты, откаты, COW:** защита от сбоев, возможность моментального восстановления состояния системы.
- **Встроенное сжатие и deduplication:** экономия дискового пространства и снижение нагрузки на хранилище.
- **Удобная работа с субтомами:** логическая изоляция `/home`, `/var`, `/opt` и др., интеграция с GRUB для выбора снапшота при загрузке.

Переход на Btrfs становится особенно актуальным в следующих ситуациях:

Когда миграция оправдана

- 🔄 **Частые обновления** — Btrfs позволяет реализовать атомарные обновления и безопасный откат.
- 🎁 **Работа с логами, контейнерами, образами** — за счёт CoW, reflink и тонкого управления квотами.
- 🛡️ **Необходимость защиты от сбоев** — снапшоты и субтомы позволяют вернуть систему в рабочее состояние за секунды.

В этой статье мы рассмотрим, как грамотно и безопасно перейти с ext4 на Btrfs, не потеряв данные и обеспечив стабильную работу системы.

2. Ограничения и риски миграции

Переход с **ext4** на **Btrfs** — это не просто изменение параметров монтирования. Файловая система `ext4` не поддерживает «на лету» конвертацию в `btrfs`, как, например, ext2 → ext3 или ext3 → ext4.

❗ Нет прямой конверсии: для перехода на Btrfs требуется полное **переформатирование** раздела и последующий перенос данных.

Это значит, что перед миграцией необходимо:

- 📦 **Создать полный резервный архив** всех данных с ext4-раздела (rsync, tar, btrfs send в будущем).
- 💾 **Переформатировать раздел:** использовать `mkfs.btrfs` с правильными опциями.
- 🔄 **Восстановить данные из бэкапа** на вновь созданную Btrfs-систему.

⚠️ **Дополнительные риски и детали, которые нужно учесть:**

- **Загрузчик:** GRUB должен поддерживать загрузку с Btrfs (в современных системах — поддержка есть).
- **UUID и LABEL:** после форматирования UUID изменится — необходимо обновить записи в `/etc/fstab` и `/boot/grub/grub.cfg` (или выполнить `update-grub`).
- 💤 **Раздел под swap:** Btrfs не поддерживает swap-файлы по-умолчанию без специальных настроек. Рекомендуется использовать отдельный swap-раздел или ZRAM.

Эти ограничения делают миграцию чувствительной к ошибкам. Подходите к переходу на Btrfs осознанно, тестируйте на стенде и всегда имейте рабочий бэкап.

3. Подготовка к миграции

Перед форматированием и переходом на **Btrfs** важно грамотно подготовить систему: проверить исходный раздел, сохранить данные и учесть конфигурационные нюансы.

🔑 Определение целевого раздела

Убедитесь, какой именно раздел используется под корневую файловую систему (`/`) или нужный каталог (`/home`, `/data`). Для этого можно использовать:

```
lsblk -f
mount | grep "^/dev"
```

Проверка целостности файловой системы ext4

Перед копированием данных **обязательно** проверьте исходную файловую систему на ошибки:

```
sudo fsck.ext4 /dev/sdX1
```

💾 Создание полного резервного копирования

Резервная копия должна быть сделана **вне форматируемого раздела** (например, на внешний диск, второй раздел или сетевой ресурс).

Вариант 1: rsync с сохранением прав и ACL:

```
sudo rsync -aAXv /mnt/ext4/ /mnt/backup/
```

- `-aAX` сохраняет права, владельцев, extended attributes
- `-v` — подробный вывод

Вариант 2: Создание сжатого архива:

```
sudo tar -cpzf /mnt/backup/full-backup.tgz /
```

🎁 Подходит для хранения всей системы в одном файле (особенно удобно при переносе или архивации).

🎁 Сохранение списка пакетов и конфигураций

Для удобства восстановления системы или переноса:

```
rpm -qa > installed-packages.txt
```

Для систем на базе Debian/Ubuntu: `dpkg --get-selections > pkg-list.txt`

💡 **Совет:** дополнительно сохраните содержимое `/etc`, `/home` и `/var` отдельно, чтобы при необходимости избирательно восстанавливать конфиги и данные.

3. Подготовка к миграции

Перед переходом с **ext4** на **Btrfs** необходимо провести тщательную подготовку: определить целевой раздел, проверить целостность файловой системы и создать полную резервную копию всех данных и конфигураций.

📌 Определение целевого раздела

Сначала определите, какой раздел содержит систему, которую вы хотите перенести на Btrfs. Используйте команду:

```
lsblk -f
```

или:

```
findmnt /
```

Обратите внимание на файловую систему (`ext4`) и имя устройства (например, `/dev/sda2`).

Проверка целостности ext4

Перед копированием файлов нужно убедиться, что текущая файловая система не содержит ошибок:

```
sudo fsck.ext4 /dev/sdX1
```

💾 Создание полного резервного копирования

Все данные **будут утеряны при форматировании**, поэтому необходимо заранее сохранить всю важную информацию. Доступны два способа:

◈ Вариант 1: Rsync (с сохранением прав и атрибутов)

```
sudo rsync -aAXv /mnt/ext4/ /mnt/backup/
```

- `-a` — архивный режим (все файлы, права, даты);
- `-A` — сохранять ACL;
- `-X` — сохранять xattrs (SELinux, AppArmor и др.);
- `-v` — подробный вывод.

🔹 Вариант 2: Архив в .tar.gz

```
sudo tar -cpzf /mnt/backup/full-backup.tgz /
```

Подходит, если нужно сохранить всю систему в одном сжатом архиве.

📦 Сохранение списка пакетов и конфигураций

Это поможет при последующей переустановке системы и ПО:

```
rpm -qa > installed-packages.txt
```

Для Debian-подобных дистрибутивов:

```
dpkg --get-selections > pkg-list.txt
```

💡 **Рекомендуется:** отдельно сохранить содержимое каталогов `/etc`, `/home`, `/var` и других нестандартных путей, если в них содержатся важные настройки или данные.

4. Переформатирование раздела в Btrfs

После создания резервной копии можно приступить к **переформатированию ext4-раздела в Btrfs**. Это приведёт к удалению всех данных, поэтому убедитесь, что копия надёжно сохранена.

📤 Размонтирование целевого раздела

Перед форматированием необходимо размонтировать раздел, если он смонтирован:

```
sudo umount /dev/sdX1
```

Если раздел используется системой, это можно сделать из LiveCD или с другого загрузочного носителя.

💾 Форматирование в Btrfs

Выполните команду для создания новой файловой системы Btrfs на целевом разделе:

```
sudo mkfs.btrfs -f /dev/sdX1
```

- `-f` — принудительное форматирование, даже если на разделе есть данные;
- `/dev/sdX1` — замените на имя вашего устройства.

⚙️ Дополнительные параметры (по желанию)

Вы можете указать дополнительные флаги для тонкой настройки:

```
sudo mkfs.btrfs -f -L rootfs -d single -m single /dev/sdX1
```

- `-L rootfs` — установка метки тома (полезно для `fstab`);
- `-d single` — одиночное (не RAID) хранение данных (по умолчанию, если один диск);
- `-m single` — одиночное размещение метаданных (также дефолт на одном устройстве).

❗ **Важно:** если вы планируете использовать несколько устройств (RAID), указывайте соответствующие параметры `-d raid1`, `-m raid1` и т. д. Подробнее — в [официальной документации Btrfs](#).

После успешного форматирования раздел готов для монтирования, создания подтомов и восстановления данных.

5. Развёртывание данных из бэкапа

После форматирования раздела в Btrfs необходимо восстановить содержимое системы или данных из ранее созданной резервной копии. Это может быть **архив `tar`** или **директория, скопированная через `rsync`**.

📎 Монтирование нового Btrfs-раздела

Сначала смонтируйте новый раздел во временную точку:

```
sudo mount /dev/sdX1 /mnt
```

Если вы создавали подтомы — укажите нужный сабтом в параметрах монтирования (`-o subvol=@`).

♻️ Восстановление из `rsync`

Если бэкап был сделан с помощью `rsync`, используйте обратную команду для восстановления:

```
sudo rsync -aAXv /mnt/backup/ /mnt/
```

- `-aAX` — сохраняет права, владельцев, ACL, xattrs;
- `-v` — подробный вывод (опционально);
- `/mnt/backup/` — путь до бэкапа;
- `/mnt/` — путь, куда разворачиваем.

🎁 Восстановление из `tar`-архива

Если вы сохранили систему в архив, разверните её в новый раздел так:

```
sudo tar -xpf full-backup.tgz -C /mnt
```

Флаги `-xpf` означают: eXtract, Preserve права, read From-файл.

🔍 Проверка после восстановления

Убедитесь, что восстановлены:

- Права доступа (`ls -l`), владельцы (`chown`);
- ACL и extended attributes (`getfacl`, `getfattr`);
- Символические ссылки, специальные файлы (через `ls -l`, `file`).

ℹ️ **Совет:** если вы переносите root-раздел, не забудьте проверить наличие `/proc`, `/sys`, `/dev` — они монтируются отдельно.

6. Настройка загрузки и `fstab`

После восстановления данных необходимо обновить конфигурацию монтирования и загрузчика, чтобы система корректно загружалась с нового Btrfs-раздела.

📄 Обновление `/etc/fstab`

Убедитесь, что в `/etc/fstab` указан актуальный UUID и параметры Btrfs. Пример записи для подтома `@`:

```
UUID=<новый_UUID>  /  btrfs  defaults,noatime,compress=zstd,subvol=@  0 1
```

- `noatime` — снижает количество лишних записей;
- `compress=zstd` — включение сжатия (можно заменить на `lzo`);
- `subvol=@` — если вы используете подтом `@` в качестве корня.

🔍 Проверка UUID

Для получения актуального UUID выполните:

```
sudo blkid /dev/sdX1
```

Замените `/dev/sdX1` на ваш раздел, где теперь размещена файловая система Btrfs.

Обновление загрузчика GRUB

Чтобы загрузчик увидел новую файловую систему и корень, необходимо обновить конфигурацию GRUB:

Debian / Ubuntu:

```
sudo update-grub
```

RHEL / Fedora / Rocky / ALTLinux:

```
sudo grub2-mkconfig -o /boot/grub2/grub.cfg
```

Также проверьте наличие `initramfs` и, при необходимости, пересоздайте его:

```
sudo update-initramfs -u
sudo dracut -f
```

⚠️ **Важно:** Если загрузчик установлен в другой раздел или вы используете UEFI, проверьте, чтобы путь к корневой ФС соответствовал новому устройству и UUID.

7. Создание субтомов и снапшотов (опционально)

Одним из ключевых преимуществ Btrfs является возможность логического разделения файловой системы на **субтомы** (subvolumes). Это удобно для изоляции разных частей системы и управления снапшотами.

📄 Пример: создание субтомов

Смонтируйте корень и создайте подтомы для системы и данных:

```
sudo mount /dev/sdX1 /mnt
sudo btrfs subvolume create /mnt/@
sudo btrfs subvolume create /mnt/@home
sudo btrfs subvolume create /mnt/@log
sudo btrfs subvolume create /mnt/@cache
```

При необходимости создайте дополнительные подтомы под `/opt`, `/srv` и т. д.

🔁 Перемонтирование с подтомами

После переноса данных в нужные сабтомы, настройте `/etc/fstab` с параметром `subvol=` для каждого:

```
UUID=<uuid>  /      btrfs  defaults,noatime,compress=zstd,subvol=@      0 1
UUID=<uuid>  /home   btrfs  defaults,noatime,compress=zstd,subvol=@home   0 2
UUID=<uuid>  /var/log btrfs defaults,noatime,compress=zstd,subvol=@log    0 2
UUID=<uuid>  /var/cache btrfs defaults,noatime,compress=zstd,subvol=@cache 0 2
```

💡 **Совет:** Использование отдельных субтомов позволяет делать выборочные снапшоты, например только `@`, исключая `@home` или `@log`.

🕒 Настройка снапшотов (Snapper или Timeshift)

- **Snapper:** инструмент для системных снапшотов с поддержкой откатов и интеграцией с YaST/Zypper, apt/dnf.
- **Timeshift:** простой инструмент для десктопов с GUI, отлично работает с Btrfs под Ubuntu, Mint и Manjaro.

Перед использованием этих утилит важно, чтобы структура подтомов была согласована с требованиями выбранного инструмента.

8. Проверка и тестовый запуск

После завершения переноса данных, настройки fstab и загрузчика, выполните перезагрузку системы и убедитесь, что всё работает корректно.

🔁 Перезагрузка

Перезагрузите систему:

```
sudo reboot
```

🔍 Проверка после загрузки

- ✅ Убедитесь, что корень и подтомы смонтированы корректно:

  ```
  mount | grep btrfs
  ```

- 📊 Посмотреть использование пространства:

  ```
  btrfs filesystem df /
  ```

- 📋 Список всех подтомов:

  ```
  btrfs subvolume list /
  ```

✅ **Если всё работает корректно**, вы успешно завершили переход с ext4 на Btrfs. Теперь можно настраивать автоматические снапшоты, сжатие, scrub и другие функции.

9. Советы по оптимизации Btrfs

После успешной миграции на **Btrfs** рекомендуется провести дополнительную настройку для повышения производительности, надёжности и удобства обслуживания.

⚙️ Рекомендуемые опции монтирования (в `fstab`)

Добавьте или уточните параметры монтирования для каждого подтома:

```
UUID=xxxx-xxxx  /  btrfs  defaults,noatime,compress=zstd,autodefrag,subvol=@  0 1
```

- `compress=zstd` — эффективное и быстрое сжатие (можно заменить на `lzo` или `zlib`);
- `autodefrag` — автоматически дефрагментирует часто изменяемые файлы (например, логи, базы данных);
- `noatime` — снижает количество записей на диск.

💡 **Совет:** для рабочих нагрузок с базами данных или VM лучше отключить `autodefrag` и CoW на уровне отдельных файлов.

🔄 Настройка регулярного `btrfs scrub`

Scrub — это проверка целостности данных и метаданных, аналог проверки с ECC:

```
sudo btrfs scrub start -Bd /
```

Чтобы запускать scrub ежемесячно, активируйте systemd-таймер:

```
sudo systemctl enable --now btrfs-scrub@-.timer
```

🕒 Инструменты для снапшотов и бэкапов

- **Snapper:** автоматическое создание снапшотов до/после обновлений, интеграция с dnf/apt/zypper.
- **btrbk:** инструмент для резервного копирования на основе снапшотов, поддерживает SSH, удалённые копии, ротацию.

📋 Мониторинг состояния устройства

Даже при использовании Btrfs, важно следить за состоянием диска (особенно SSD):

```
sudo smartctl -a /dev/sdX
```

Также полезно анализировать логи:

```
journalctl -u btrfs-scrub@-.service
dmesg | grep -i btrfs
```

Резюме: правильные параметры монтирования + регулярный `scrub` + система

снапшотов — основа стабильной и надёжной работы Btrfs в течение долгого времени.

10. Заключение

Миграция с **ext4** на **Btrfs** требует внимательности, понимания и тестирования — но взамен вы получаете современную, гибкую и мощную файловую систему.

✅ **Btrfs особенно выгоден:**

- 📈 В системах с высокой скоростью изменений: журналы, контейнеры, кэш.
- 🔁 Там, где важна откатность: снапшоты, откаты, тестирование.
- ⚡ На NVMe, SSD и гибридных серверах: сжатие, параллельные операции, CoW.

Тем не менее, **ext4** остаётся надёжной, стабильной и проверенной временем файловой системой. Она отлично подходит для минималистичных, встраиваемых и критически стабильных систем, где "магия" не нужна.

Выбор между ext4 и Btrfs — это не вопрос «что лучше», а вопрос **подходящей задачи**. При правильной настройке Btrfs предлагает больше контроля, гибкости и современных возможностей.

📎 Приложение: пример `/etc/fstab` после миграции

```
UUID=xxxx-xxxx  /       btrfs  rw,noatime,compress=zstd,subvol=@       0 1
UUID=xxxx-xxxx  /home   btrfs  rw,noatime,compress=zstd,subvol=@home   0 2
```

Здесь предполагается, что вы создали подтомы `@` (для `/`) и `@home` (для `/home`) при миграции.

Итог: Btrfs подойдёт тем, кто хочет современную систему хранения с поддержкой снапшотов, сжатия и простого отката. Ext4 — тем, кто ценит стабильность и простоту.