Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Btrfs-снапшоты и откат Linux-сервера: как мы это внедряем

Схема субтомов, snapper с хуками apt, откат через default subvolume, репликация send/receive

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Обновление пакетов, правка конфигов или неудачная миграция на Linux-сервере (nginx/PHP, сервер 1С, PostgreSQL, файловый шлюз) ломает сервис, а инженер часами восстанавливает состояние по apt-логам. Мы закладываем Btrfs-снапшоты со snapper: pre/post-снимок на каждую транзакцию apt, откат за одну перезагрузку, а данные и логи в отдельных субтомах при этом не теряются.

Почему это важно бизнесу

- Неудачный apt upgrade на сервере 1С или сайта — это часы простоя всего офиса, а не «пять минут переустановки».
- Ручной откат по apt-истории непредсказуем: старые версии зависимостей уже недоступны в репозитории, итог — «полуживая» система.
- Снимок перед каждым изменением делает обновления безопасными — их можно ставить регулярно, а не откладывать годами.
- Инкрементальный send/receive на резервный узел даёт восстановление за минуты без ночного копирования гигабайт по rsync.
- Откат одного конфига из снимка занимает секунды и не требует перезагрузки — меньше простоев из-за человеческих ошибок.

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

6.12 / 6.8

Ядро Debian 13 и Ubuntu 24.04 LTS — база наших внедрений: space_cache=v2 и discard=async штатно

по докам btrfs.readthedocs.io

0.10.6

snapper в Debian 13 / Ubuntu 24.04 (upstream 0.13.2): таймеры timeline/cleanup, хук 80snapper

snapper-configs(5), Debian tracker

zstd:3

Уровень сжатия, дефолт zstd по btrfs(5): 1,5–2x на текстах и логах без заметной задержки записи

btrfs(5) / наш стандарт

0.5 / 0.2

SPACE_LIMIT / FREE_LIMIT в snapper: снимки ≤50% ФС, свободно ≥20%; SPACE_LIMIT требует QGROUP

snapper default-config

10/10/8/6

TIMELINE_LIMIT hourly/daily/weekly/monthly, yearly=0; NUMBER_LIMIT=50 для pre/post-пар apt

наш стандарт

30 дней

btrfs scrub по таймеру раз в месяц (по btrfs-scrub(8)): сверка csum данных и метаданных

по докам btrfs-scrub(8)

РЕАЛИЗАЦИЯ 1 ИЗ 3

Разметка и монтирование: субтома под безопасный откат

Что настраиваем

Корневой раздел SSD/NVMe сервера на Debian 13 / Ubuntu 24.04; при двух дисках — mdadm RAID-1 или btrfs raid1 под корнем

Как мы это делаем

1. mkfs.btrfs -L system /dev/nvme0n1p2; субтома верхнего уровня @, @home, @var_log, @var_cache, @var_lib_docker, @tmp, @snapshots — не вложенные друг в друга
2. fstab: корень без subvol= (грузимся через default subvolume); остальные — subvol=@home,compress=zstd:3,noatime,ssd,discard=async; /.snapshots — субтом @snapshots
3. Избыточность — mdadm RAID-1 под Btrfs или btrfs -m raid1 -d raid1 на двух дисках; raid5/raid6 средствами Btrfs (write hole, статус unstable) не применяем
4. Каталоги PostgreSQL / кластера 1С — отдельный субтом @pgdata с chattr +C до записи данных (nodatacow); сжатие и контрольные суммы там отключаются
5. Docker — overlay2 на @var_lib_docker (не storage-driver btrfs): снимок корня не тянет сотни слоёв, вывод snapper list остаётся читаемым

РЕЗУЛЬТАТ

Откат корня не трогает /home, логи, кэши и БД; снимки корня остаются компактными (десятки ГБ на 60 снимков при 500 ГБ данных), сжатие zstd:3 экономит 1,5–2x на текстах и логах без заметной нагрузки на CPU.

КЛЮЧЕВОЙ НЮАНС

Снимок не включает вложенные субтома — они превращаются в пустые каталоги; поэтому все субтома создаём на верхнем уровне ФС и собираем дерево только через fstab.

РЕАЛИЗАЦИЯ 2 ИЗ 3

snapper: pre/post на apt, timeline и cleanup

Что настраиваем

Все Linux-хосты на Btrfs: конфиг root обязателен, home — по необходимости; ротация снимков в /.snapshots

Как мы это делаем

1. apt install snapper; snapper -c root create-config /; в /etc/snapper/configs/root правим TIMELINE_LIMIT_* (10/10/8/6/0), NUMBER_LIMIT=50, NUMBER_LIMIT_IMPORTANT=20
2. Хук 80snapper из пакета делает pre/post на каждую транзакцию apt; в /etc/default/snapper держим DISABLE_APT_SNAPSHOT=no, в описание снимка пишем команду apt
3. systemctl enable --now snapper-timeline.timer snapper-cleanup.timer; контроль — snapper -c root list и journalctl -u snapper-cleanup
4. FREE_LIMIT=0.2 работает по statvfs без квот; SPACE_LIMIT считает через QGROUP — полные qgroup на боевых хостах не включаем: удаление снимка может повесить хост
5. Перед своими работами — ручной снимок: snapper -c root create -d 'before nginx tls' --cleanup-algorithm number; после — status и diff между номерами

РЕЗУЛЬТАТ

Каждое изменение пакетов зафиксировано парой pre/post; snapper status 42..43 показывает, что именно изменилось; откат одного файла — undochange 42..0 /etc/nginx/nginx.conf без перезагрузки; старые снимки чистятся сами по лимитам.

КЛЮЧЕВОЙ НЮАНС

Snapper rollback требует btrfs и корень, смонтированный через default subvolume: в fstab без subvol=, в GRUB без rootflags=subvol=@ — иначе после reboot грузится старый корень.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Откат и репликация: rollback, grub-btrfs, send/receive

Что настраиваем

Продовый сервер + резервный узел или NAS с Btrfs; ночная инкрементальная репликация снимков по SSH

Как мы это делаем

1. Откат: snapper -c root rollback 42 — ro-снимок текущего состояния + rw-снимок из #42, set-default; reboot. Старый корень остаётся в /.snapshots для разбора
2. Файловый откат без reboot: snapper -c root status 42..43, затем undochange 42..0 /etc/nginx/nginx.conf; либо cp из /.snapshots/N/snapshot/
3. grub-btrfs + grub-btrfsd (следит за /.snapshots) добавляют снимки в меню GRUB — загрузка в снимок без live-USB, если система после обновления не стартует
4. btrbk: snapshot_preserve 14d 8w, target ssh://backup/.../host1 с target_preserve 30d 6m; в ход идут только ro-снимки, инкремент через -p parent
5. Регламент: btrfs scrub start по таймеру раз в 30 дней, btrfs device stats в Zabbix; balance -dusage=50 только при нехватке нераспределённого места, не по крону

РЕЗУЛЬТАТ

Полный откат — три команды и около трёх минут вместо часов разбора apt-истории; ночной инкремент 500-ГБ тома уходит на резервный узел за минуты; при отказе диска разворачиваем последний принятый снимок на другом хосте.

КЛЮЧЕВОЙ НЮАНС

Последний общий parent-снимок нельзя удалять на источнике раньше, чем приёмник его получил, — иначе следующий send станет полным; проверяем received_uuid через btrfs subvolume show.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **subvol=@ в fstab и rollback**
  rollback меняет default subvolume, а fstab/GRUB держат subvol=@ — грузится старое. Корень монтируем без subvol= либо подменяем @ вручную из снимка.

- ✕ **Вложенные субтома внутри корня**
  В снимке они становятся пустыми каталогами: откат «теряет» /var/lib/docker или /home. Все субтома — на верхнем уровне, сборка дерева через fstab.

- ✕ **qgroup на боевом сервере**
  Полные квоты пересчитывают ссылки экстентов; создание/удаление снимка может повесить хост. Не включаем; при нужде — quota enable --simple (ядро 6.7+).

- ✕ **RAID5/6 средствами Btrfs**
  Write hole без журнала, в документации статус unstable. Избыточность — mdadm RAID-1/10 под Btrfs или btrfs raid1 для данных и метаданных.

- ✕ **База данных под CoW**
  PostgreSQL/1C фрагментируют файлы при CoW, задержки растут. Отдельный субтом + chattr +C до записи; снимки работают, но без сжатия и csum.

- ✕ **«Нет места» при свободном df**
  Место держат снимки и незанятые чанки. FREE_LIMIT=0.2 в snapper, cleanup all, затем balance -dusage=50 возвращает чанки; сжатие тут не лечит.

- ✕ **Удалили parent на источнике**
  Следующий btrfs send без -p идёт полным — гигабайты по SSH за ночь не успеют. btrbk держит parent через snapshot_preserve_min; сверяем received_uuid.

- ✕ **Timeline копит снимки годами**
  Дефолт TIMELINE_LIMIT_YEARLY=10 и MONTHLY=10 забивает /.snapshots. Ставим hourly 10 / daily 10 / weekly 8 / monthly 6 / yearly 0 и NUMBER_LIMIT=50.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Корень на Btrfs, субтома @, @home, @var_log, @snapshots; noatime, compress=zstd:3
- snapper root-конфиг + apt-хук 80snapper; таймеры snapper-timeline/-cleanup включены
- Ручной снимок snapper create перед своими работами; runbook с командой rollback
- Лимиты TIMELINE_LIMIT_* и NUMBER_LIMIT заданы, дефолт yearly=10 убран

НОРМАЛЬНО

- Корень через default subvolume, rollback проверен на тесте; grub-btrfs в меню GRUB
- btrfs scrub по таймеру раз в 30 дней; device stats и место под снимки в Zabbix
- Отдельные субтома @var_cache, @var_lib_docker, @tmp; БД на nodatacow-субтоме
- SPACE_LIMIT/FREE_LIMIT выставлены, алерт при заполнении ФС выше 80% с учётом снимков

ХОРОШО

- btrbk: ночной send/receive на второй узел с Btrfs, ротация 30d/6m на приёмнике
- Квартальный тест: загрузка из снимка и разворот тома с приёмника на другом хосте
- mdadm RAID-1 или btrfs raid1 под корнем; ошибки scrub/device stats — триггеры Zabbix
- Runbook отката для дежурного: команды, что восстанавливается и что нет (/home, БД)

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Корень сервера на Btrfs с отдельными субтомами для /home, /var/log, /var/cache и /.snapshots?
- [ ] Перед каждой транзакцией apt создаётся пара pre/post (snapper list показывает тип pre/post с описанием apt)?
- [ ] Таймеры snapper-timeline/-cleanup активны, лимиты TIMELINE_LIMIT_* заданы осознанно, а не по дефолту?
- [ ] Откат snapper rollback + reboot реально проверен на этом сервере, а не только описан в документации?
- [ ] Есть загрузка в снимок из меню GRUB (grub-btrfs) на случай, если система после обновления не стартует?
- [ ] Квоты qgroup на боевом сервере выключены, RAID5/6 средствами Btrfs не используется?
- [ ] btrfs scrub запускается по расписанию, ошибки device stats и scrub уходят в мониторинг?
- [ ] Снимки реплицируются send/receive на другой узел, parent-снимок не удаляется раньше, чем его принял приёмник?
- [ ] Задокументировано, что rollback НЕ восстанавливает (/home, /var/log, БД), и как их откатывать отдельно?
- [ ] Свободное место контролируется: FREE_LIMIT задан, есть алерт при заполнении выше 80% с учётом снимков?

Если хотя бы на два вопроса ответ «нет» или «не знаю» — тема требует внимания.

ЧТО МЫ ДЕЛАЕМ

Как поможет ITFresh

ITFresh — ИТ-аутсорсинг для юридических лиц до 50 рабочих мест в Москве и области. 15+ лет практики, собственная инфраструктура в дата-центре МТС (8 серверов Dell Xeon Platinum).

- Проектируем разметку Btrfs под сервер (субтома, fstab, RAID) и переносим ext4-серверы с миграцией данных
- Разворачиваем snapper с apt-хуками, таймерами и лимитами; проверяем rollback и загрузку из снимка на тесте
- Настраиваем btrbk send/receive на резервный узел или NAS, ротацию на обеих сторонах, квартальный тест
- Подключаем scrub, device stats и место под снимки к Zabbix 7.0; пишем runbook отката для дежурного инженера
- Сопровождаем обновления Linux-серверов по регламенту «снимок — обновление — проверка — откат при сбое»

15+
лет в ИТ-поддержке

50
рабочих мест — наш профиль

МТС
дата-центр, Москва

КОНТАКТЫ

Обсудить вашу задачу

Сайт	**itfresh.ru**
Телефон	**+7 903 729-62-41**
Telegram	**@ITfresh_Boss**

Бесплатно посмотрим вашу инфраструктуру по этому чек-листу и скажем, где тонко — без обязательств.

itfresh.ru

ДОКУМЕНТАЦИЯ И НАШ СТЕК

Техническая база

01 BTRFS documentation: btrfs(5) mount options, Compression, Status (btrfs.readthedocs.io — progs 7.1)

02 btrfs-subvolume(8), btrfs-send(8), btrfs-scrub(8), btrfs-quota(8) (btrfs.readthedocs.io — 2026)

03 snapper(8), snapper-configs(5), data/default-config (snapper.io — 0.13.2)

04 SLES 15 SP7: System recovery and snapshot management with Snapper (documentation.suse.com — 15 SP7)

05 btrbk(1), btrbk.conf(5): retention, ssh targets, incremental send (digint.ch — 2026)

06 grub-btrfs(8) и grub-btrfsd: снимки в меню GRUB (github.com/Antynea/grub-btrfs — 2026)

07 Наш шаблон: fstab, snapper root-config, btrbk.conf (Debian 13/Ubuntu 24.04) (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.