Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

# Rootless Podman вместо Docker: миграция без простоя

Как мы переводим контейнеры клиентов на Podman 5.x, quadlet и systemd --user

Июль 2026

**itfresh.ru · ИТ-аутсорсинг для юридических лиц**

О ЧЁМ ЭТОТ РАЗБОР

# Суть проблемы

Демон Docker работает от root, а членство в группе docker равносильно root на хосте: скомпрометированный CI-агент или разработчик получает контроль над машиной целиком. Мы переводим нагрузку на rootless Podman: демона нет, контейнер — процесс пользователя в user namespace, компрометация ограничена его UID и подмонтированными томами.

## Почему это важно бизнесу

- Утечка через один контейнер не становится компрометацией хоста: радиус поражения ограничен UID сервисного пользователя
- Подрядчику и разработчику больше не нужны права, эквивалентные root на продакшн-хосте
- Автозапуск и рестарт описаны в systemd — исчезает класс инцидентов «после перезагрузки сервис не поднялся»
- Единый журнал journalctl вместо docker logs ускоряет разбор инцидента и передачу дежурства
- Разграничение доступа закрывается архитектурой процесса, а не регламентом на бумаге

ЧТО МЫ ЗАКЛАДЫВАЕМ

# Ключевые параметры реализации

**5.0**

с этой версии rootless-сеть по умолчанию — pasta вместо slirp4netns; ниже в проект не берём

*релиз Podman 5.0*

**4.4**

с этой версии есть quadlet: сервис описан в *.container, юнит собирает генератор systemd

*podman-systemd.unit(5)*

**65536**

размер диапазона subuid/subgid на сервисного пользователя в /etc/subuid и /etc/subgid

*наш стандарт*

**1024→80**

штатный порог net.ipv4.ip_unprivileged_port_start и куда опускаем его как исключение

*sysctl, документация rootless-режима*

**00:00**

podman-auto-update.timer: OnCalendar=daily плюс RandomizedDelaySec=900, разброс 15 минут

*podman-auto-update(1)*

**3–5 сут**

держим Docker и Podman параллельно на узле, прежде чем гасить демон

*наш стандарт*

РЕАЛИЗАЦИЯ 1 ИЗ 3

# Подготовка узла и сервисного пользователя под rootless

## Что настраиваем

*Rocky Linux 9 или Ubuntu 24.04, один узел под контейнеры отдела; сервисный пользователь без sudo и без группы docker*

## Как мы это делаем

1. Ставим podman рядом с Docker и сверяем podman info --format '{{.Host.Security.Rootless}}' и podman version — в проект берём 5.x
2. Заводим пользователя app, смотрим grep '^app:' /etc/subuid /etc/subgid; после правки диапазонов — podman system migrate под этим пользователем
3. loginctl enable-linger app: без linger юниты systemctl --user умирают с сессией и не стартуют при загрузке узла
4. Требуем cgroups v2: на v1 rootless-лимиты не работают, для CPU и CPUSET добавляем drop-in user@.service с Delegate=
5. Хранилище ~/.local/share/containers/storage выносим на отдельный раздел с квотой, чтобы образы не съели корень

**РЕЗУЛЬТАТ**

Узел принимает нагрузку без единого процесса от root: демона нет, контейнеры — дочерние процессы пользователя app. Группа docker не нужна никому, место под образы ограничено разделом, а не всем диском.

**КЛЮЧЕВОЙ НЮАНС**

Правки /etc/subuid и /etc/subgid не подхватываются на лету: user namespace держит живым pause-процесс. Пока не выполнен podman system migrate, созданные контейнеры работают со старым маппингом UID.

РЕАЛИЗАЦИЯ 2 ИЗ 3

# Перенос compose-стека на quadlet-юниты systemd

## Что настраиваем

*Стек из 6–10 сервисов — reverse-proxy, приложение, БД, кэш — на одном узле, целиком под systemctl --user*

## Как мы это делаем

1. Сначала поднимаем как есть: systemctl --user enable --now podman.socket и DOCKER_HOST=unix://$XDG_RUNTIME_DIR/podman/podman.sock
2. Переписываем в ~/.config/containers/systemd/*.container: Image полным именем с реестром, PublishPort, Volume, Network= из своего *.network
3. Тома переносим podman volume export vol -o vol.tar; на приёмнике сначала podman volume create, затем podman volume import — сам он том не создаёт
4. Владельца файлов в томе правим podman unshare chown -R; на SELinux-хостах к Volume добавляем :Z, на Ubuntu он не нужен
5. Ставим HealthCmd и HealthInterval (по умолчанию 30s), HealthOnFailure=kill, а перезапуск отдаём [Service] Restart=always

**РЕЗУЛЬТАТ**

Compose-файл перестаёт быть точкой отказа: порядок старта, зависимости и рестарт описаны в systemd, логи идут в journalctl -u, состояние читается из systemctl --user status. После перезагрузки узла стек поднимается сам.

**КЛЮЧЕВОЙ НЮАНС**

Сгенерированный юнит не редактируют: правим *.container и делаем systemctl --user daemon-reload, генератор пересобирает его. Без [Install] WantedBy=default.target сервис не стартует при загрузке.

РЕАЛИЗАЦИЯ 3 ИЗ 3

# Обновления, резервные копии и контроль после перехода

## Что настраиваем

*Все rootless-узлы: политика обновления образов, снятие томов, мониторинг состояния юнитов и healthcheck*

## Как мы это делаем

1. AutoUpdate=registry в *.container плюс systemctl --user enable --now podman-auto-update.timer вместо ручного podman pull
2. Разводим таймер по узлам drop-in'ом: сначала пустая строка OnCalendar=, затем своё время, иначе значение добавится к штатному
3. Перед волной обновления — podman auto-update --dry-run; при неудачном рестарте юнита откат на прежний образ включён по умолчанию
4. Бэкап: podman volume export по каждому тому, каталог ~/.config/containers/systemd — в git, в образах версионный тег вместо latest
5. Мониторим не docker ps, а systemctl --user is-failed и статус healthcheck; алерт на failed-юнит и на unhealthy-контейнер

**РЕЗУЛЬТАТ**

Обновление образов перестаёт быть ручной операцией: в реестре появился новый digest — юнит перезапускается сам, не поднялся — откатывается на прежний образ. Восстановление узла сводится к git clone каталога юнитов и распаковке томов.

**КЛЮЧЕВОЙ НЮАНС**

Политика registry работает только для контейнеров под systemd-юнитом и только с полностью квалифицированным именем образа. Короткое имя ломает auto-update, а тег latest лишает откат ориентира.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

# Подводные камни

- ✕ **Пользователь в группе docker = root**
  Доступ к сокету демона даёт root на хосте — достаточно смонтировать / внутрь контейнера. Группу docker снимаем сразу после миграции узла.

- ✕ **Забыли loginctl enable-linger**
  Без linger юниты systemctl --user гаснут вместе с последней сессией и не стартуют при загрузке. Проверяем loginctl show-user на каждом узле.

- ✕ **Публикация портов 80 и 443 в rootless**
  Rootless не биндит порты ниже 1024. Слушаем 8080/8443 и терминируем TLS на хостовом nginx, а не правим ip_unprivileged_port_start.

- ✕ **Теряется реальный IP клиента**
  При пробросе через rootlessport источник виден как адрес шлюза. Ставим впереди хостовой nginx с X-Forwarded-For либо переводим сеть на pasta.

- ✕ **Права на bind-mount после переезда**
  UID внутри user namespace и на хосте не совпадают. Чиним podman unshare chown -R, а на SELinux-хостах добавляем к тому флаг :Z.

- ✕ **Правка subuid без system migrate**
  Новый диапазон не применится к уже созданным контейнерам, пока жив pause-процесс. После правки выполняем podman system migrate под пользователем.

- ✕ **Ставка на Swarm и Docker-in-Docker**
  Podman не реализует Swarm, а DinD-пайплайны переносятся тяжело. Такие узлы честно оставляем на Docker и не ломаем рабочий процесс.

- ✕ **Сеть rootless под тяжёлой нагрузкой**
  Пользовательский сетевой стек дороже bridge по CPU. Для БД и стриминга берём rootful Podman или host-сеть, полумеры тут не работают.

УРОВНИ РЕАЛИЗАЦИИ

# Как правильно

## МИНИМУМ

- Podman 5.x, отдельный пользователь на сервис, группа docker расформирована
- loginctl enable-linger включён на всех сервисных пользователях
- Порты ниже 1024 не публикуются, наружу смотрит хостовой reverse-proxy
- Docker и Podman работают параллельно 3–5 суток до отключения демона

## НОРМАЛЬНО

- Все сервисы описаны в *.container, compose выведен из продакшена
- В каждом юните HealthCmd, HealthOnFailure и [Service] Restart=always
- podman-auto-update.timer включён, образы заданы полными именами
- Каталог ~/.config/containers/systemd под git с ревью изменений

## ХОРОШО

- Версионные теги вместо latest, обновления через auto-update --dry-run
- Свои сети *.network и секреты через Secret=, а не переменные окружения
- Тома снимаются podman volume export по расписанию с тестом распаковки
- Состояние юнитов и healthcheck заведены в Zabbix с алертом дежурному

ПРОВЕРЬТЕ У СЕБЯ

# Чек-лист самопроверки

- [ ] Есть ли на узлах пользователи в группе docker и знаете ли вы поимённо, кто это?
- [ ] Показывает ли podman info rootless: true и стоит ли на узлах версия 5.x?
- [ ] Включён ли linger и поднимаются ли контейнеры после холодной перезагрузки узла?
- [ ] Заданы ли диапазоны в /etc/subuid и /etc/subgid и выполнялся ли podman system migrate?
- [ ] Описаны ли сервисы quadlet-юнитами и лежат ли эти юниты в системе контроля версий?
- [ ] Есть ли у юнитов [Install] WantedBy=default.target, иначе они не стартуют при загрузке?
- [ ] Настроен ли healthcheck и что делает система при переходе контейнера в unhealthy?
- [ ] Включён ли podman-auto-update.timer, разведён ли по узлам и заданы ли полные имена образов?
- [ ] Проверялось ли восстановление тома из podman volume export на тестовом узле?
- [ ] Есть ли перечень сервисов, которые сознательно остаются на Docker, и обоснование?

*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).

- Инвентаризируем контейнеры, тома и порты, отделяем то, что мигрировать не нужно
- Готовим узел: сервисные пользователи, subuid/subgid, linger, квота на хранилище
- Переписываем compose-стек в quadlet-юниты с healthcheck и автозапуском
- Настраиваем podman auto-update, бэкап томов и откат на предыдущий образ
- Заводим юниты и healthcheck в мониторинг, передаём регламент эксплуатации

**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** podman-systemd.unit(5): ключи [Container] и пути quadlet-юнитов (docs.podman.io — 5.x)

**02** podman-auto-update(1): политики registry и local, таймер, откат (docs.podman.io — 5.x)

**03** podman-run(1): --health-cmd, --health-interval, --health-on-failure (docs.podman.io — 5.x)

**04** podman-system-migrate(1): subuid/subgid и pause-процесс (docs.podman.io — 5.x)

**05** podman-volume-export(1) и podman-volume-import(1): перенос томов (docs.podman.io — 5.x)

**06** Building, running, and managing containers, RHEL 9 и 10 (docs.redhat.com — 2026)

**07** Наш шаблон quadlet-юнита и чек-лист приёмки rootless-узла (itfresh.ru — 2026)


*Основано на официальной документации продуктов и нашей практике внедрения.*