Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Podman: rootless-контейнеры без демона — наша методика

Разворачиваем Podman 5.8/6.0 с Quadlet, subuid-маппингом и auto-update вместо Docker-демона

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Docker-демон работает от root, а членство в группе docker эквивалентно root-доступу к хосту: скомпрометированный CI-раннер или подрядчик с доступом к сокету получает весь сервер. Сам демон — единая точка отказа: его рестарт кладёт все контейнеры разом. Мы переводим серверы клиентов на Podman — rootless-контейнеры под управлением systemd, без демона и без группы docker.

Почему это важно бизнесу

- Компрометация одного сервиса не даёт root на сервере: user namespace держит контейнер под непривилегированным uid
- Нет демона — нет общей точки отказа: сбой движка не останавливает уже работающие контейнеры клиентских сервисов
- Аудиты ИБ заказчиков проходим без исключений в политиках: на хосте нет docker-сокета и контейнерных процессов от root
- Обновления образов по расписанию с авто-откатом по healthcheck — меньше ручных работ и ночных простоев

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

5.8/6.0

версии Podman, которые мы ставим: 6.0 на новые серверы, ветка 5.8.x — на действующих хостах до...

по докам podman.io

65536

размер диапазона subUID/subGID в /etc/subuid: root контейнера маплится в uid 100000+ хоста

наш стандарт / subuid(5)

00:00

podman-auto-update.timer: OnCalendar=daily + RandomizedDelaySec=900 — ежедневная сверка digest...

по докам podman-auto-update(1)

<2%

сетевой оверхед rootless на backend pasta по нашим стендовым замерам, source IP клиента сохран...

наш стенд / passt.top

30 с

HealthInterval в Quadlet: healthcheck обязателен — без него auto-update не сможет откатиться

наш стандарт

0 МБ

резидентного демона нет: контейнеры — потомки systemd через conmon, чистый fork/exec

архитектура Podman

РЕАЛИЗАЦИЯ 1 ИЗ 3

Rootless-базис: subuid, linger, Quadlet

Что настраиваем

Типовой прикладной сервер клиента (Debian 13 / AlmaLinux 9): web-сервисы, обвязка 1С, вспомогательные БД

Как мы это делаем

1. Создаём выделенного пользователя: useradd -r -m svc-app; usermod --add-subuids 300000-365535 --add-subgids 300000-365535 svc-app — свой диапазон на каждый сервис
2. Включаем loginctl enable-linger svc-app — user-юниты живут без интерактивной сессии и стартуют при загрузке сервера
3. Описываем контейнеры Quadlet-файлами в ~/.config/containers/systemd/*.container: Image (FQIN), PublishPort, Volume с :Z, HealthCmd, AutoUpdate=registry
4. systemctl --user daemon-reload && systemctl --user start app.service; логи — штатно в journald: journalctl --user -u app
5. Проверяем маппинг: podman unshare cat /proc/self/uid_map — root контейнера обязан попадать в выделенный диапазон, а не в uid 0

РЕЗУЛЬТАТ

Каждый сервис живёт под своим uid и своим диапазоном subordinate-ID: компрометация web-контейнера не даёт доступа ни к соседней БД, ни к хосту. Управление — штатным systemd (enable, зависимости, Restart=always), без отдельного демона и его конфигов.

КЛЮЧЕВОЙ НЮАНС

Диапазоны subuid разных пользователей не должны пересекаться — ведём их реестр по хосту. После правки /etc/subuid обязателен podman system migrate, иначе в хранилище останется старый маппинг.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Миграция с Docker: сокет-совместимость и CI

Что настраиваем

Серверы с docker-compose-стеками и GitLab CI-раннерами; миграция без переписывания пайплайнов

Как мы это делаем

1. Снимаем инвентарь: docker ps -a, compose-файлы, тома и всех, кто ходит в /var/run/docker.sock (Traefik, CI, агенты мониторинга)
2. Включаем Docker-совместимый API: systemctl --user enable --now podman.socket; клиентам задаём DOCKER_HOST=unix://$XDG_RUNTIME_DIR/podman/podman.sock
3. Ставим пакет podman-docker (команда docker становится обёрткой Podman) — скрипты и Makefile работают без правок
4. Compose-стеки конвертируем в Quadlet (podlet или вручную): .container/.pod/.volume-файлы вместо docker-compose.yml
5. Для портов ниже 1024 задаём sysctl net.ipv4.ip_unprivileged_port_start=80 точечным drop-in в /etc/sysctl.d

РЕЗУЛЬТАТ

CI-пайплайны и утилиты, привязанные к Docker API, продолжают работать через podman.socket, а прод-контейнеры уже крутятся под systemd без root. Docker сносим с хоста только после недели параллельной обкатки на staging.

КЛЮЧЕВОЙ НЮАНС

podman generate systemd объявлен deprecated — новые юниты только через Quadlet (podman-systemd.unit). Аналога Docker Swarm нет: такие стеки переводим в pod-ы или k8s-манифесты через podman kube play.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Автообновление образов с авто-откатом

Что настраиваем

Все rootless-хосты на нашей поддержке; контроль прогонов — journald плюс Telegram-алерт дежурному

Как мы это делаем

1. В Quadlet ставим AutoUpdate=registry и только полные имена образов (docker.io/library/nginx:1.27-alpine) — по коротким именам сверка digest не сработает
2. Включаем systemctl --user enable --now podman-auto-update.timer; расписание переопределяем drop-in-ом OnCalendar под сервисное окно клиента
3. Перед боевым прогоном — podman auto-update --dry-run: видно, каким юнитам приедет новый digest
4. Добавляем Notify=healthy и HealthCmd: если после обновления юнит не выходит в healthy, Podman откатывает его на предыдущий образ
5. Итоги прогонов забираем из journalctl -u podman-auto-update.service и отправляем в Telegram-канал дежурного

РЕЗУЛЬТАТ

Патчи базовых образов доезжают до прода ежедневно без ручных работ; неудачное обновление автоматически откатывается по healthcheck — клиентский сервис не лежит до утра, а у дежурного есть алерт с причиной.

КЛЮЧЕВОЙ НЮАНС

Auto-update сравнивает digest, а не тег: тег фиксируем по минорной ветке (nginx:1.27-alpine), никогда не latest — иначе однажды приедет мажорный релиз с ломающими изменениями.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Короткое имя образа в Quadlet**
  AutoUpdate=registry требует FQIN (docker.io/...): по короткому имени Podman не определит registry для сверки digest — всегда пишем полный путь
- ✕ **Сервисы гаснут после logout**
  Без loginctl enable-linger systemd убивает user-сессию вместе с контейнерами — включаем linger сразу при создании сервис-пользователя
- ✕ **Rootless не биндит порт 80**
  Непривилегированный uid не откроет порт <1024 — задаём net.ipv4.ip_unprivileged_port_start=80 или публикуем 8080 за reverse-proxy
- ✕ **Volume без :Z на SELinux**
  На RHEL-семействе контейнер ловит EACCES к тому — ставим :Z (или :z для общих томов) в Volume=, а :U чинит владельца под user namespace
- ✕ **Хардкод /var/run/docker.sock**
  Сторонний софт ждёт docker.sock — поднимаем podman.socket и отдаём путь через DOCKER_HOST или симлинк, иначе агенты мониторинга слепнут
- ✕ **Теряется source IP клиента**
  Через rootlessport сервис видит 127.0.0.1 вместо адреса клиента — используем сеть pasta: её порт-форвардинг сохраняет исходный IP
- ✕ **generate systemd в новых проектах**
  Команда deprecated и даёт хрупкие юниты — только Quadlet; файлы валидируем quadlet -dryrun (/usr/libexec/podman на RHEL, /usr/lib/podman на Debian)
- ✕ **Диск пухнет от старых образов**
  Auto-update копит вытесненные образы в ~/.local/share/containers — ставим prune-таймер не раньше суток после прогона (старый образ нужен для отката)

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Podman 5.x из репозитория дистрибутива (Debian 13 / AlmaLinux 9), 5.8/6.0 — из сборо…
- Quadlet-файлы вместо podman run в rc.local и cron
- FQIN-имена образов и фиксированные минорные теги вместо latest

НОРМАЛЬНО

- Отдельный subuid/subgid-диапазон на каждый сервис, реестр диапазонов
- podman-auto-update.timer в окне клиента, контроль через --dry-run
- HealthCmd + Notify=healthy для авто-отката неудачных обновлений
- podman.socket для CI и мониторинга, совместимых только с Docker API

ХОРОШО

- Podman 6.0, сеть pasta, journald-логи в центральный сборщик
- Скан образов Trivy в CI до пуша, свой зеркальный registry
- Prune-таймер, квоты на storage, Telegram-алерты о сбоях auto-update
- SELinux enforcing и seccomp-профили по умолчанию, ноль --privileged

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Контейнеры запущены не от root, и на хосте нет группы docker с широким членством?
- [ ] Для каждого сервис-пользователя включён linger и выделен свой диапазон subuid/subgid?
- [ ] Все контейнеры описаны Quadlet-файлами и управляются systemd, а не запускаются руками?
- [ ] В юнитах прописаны полные имена образов (FQIN) и фиксированные теги, а не latest?
- [ ] Включён podman-auto-update.timer и проверен реальный сценарий отката по healthcheck?
- [ ] Порты ниже 1024 решены осознанно: sysctl, reverse-proxy или отдельный root-сервис по исключению?
- [ ] Тома промаркированы для SELinux (:Z/:z), система работает в enforcing?
- [ ] Софт, ожидающий docker.sock, переведён на podman.socket и проверен под нагрузкой?
- [ ] Есть prune-политика и алерт на заполнение хранилища образов?

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).

- Аудит текущего Docker-хозяйства и план миграции на rootless Podman без простоя сервисов
- Конвертация docker-compose-стеков в Quadlet-юниты с healthcheck и автообновлением
- Настройка CI/CD и мониторинга через podman.socket — пайплайны работают без переписывания
- Сопровождение: auto-update с откатом, prune-политики, алерты в Telegram, ежемесячный отчёт

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) — Quadlet (docs.podman.io — 5.8/6.0)

02 podman-auto-update(1) (docs.podman.io — 5.8)

03 Podman 6.0 Release Notes (podman.io — 6.0.0)

04 Rootless Podman tutorial (docs.podman.io — 5.8)

05 pasta — user-mode networking (passt.top — 2026)

06 Наш шаблон Quadlet + чек-лист миграции (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.