Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Долгий старт Pod с большим PVC: чиним fsGroup chown

Разбираем OnRootMismatch, CSI fsGroupPolicy и mount-time fsGroup

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Образ скачан, том примонтирован, а Pod минуты висит в ContainerCreating: kubelet перед стартом рекурсивно делает chown/chmod каждого файла на томе под fsGroup из securityContext. Время растёт не с гигабайтами, а с числом файлов — на PVC с миллионами мелких файлов (аплоады, логи, датасеты) это минуты к каждому старту и ретраю, срывая SLA готовности сервиса и окна деплоя.

Почему это важно бизнесу

- Каждый рестарт Pod с большим PVC — это минуты простоя сервиса сверх обычного времени старта
- HPA/rolling update растягивается: новые реплики не проходят readiness, пока идёт chown тома
- Ретраи по liveness/OOM превращаются в цикл долгих re-mount и повторный chown на каждом старте
- Инженеры тратят часы на разбор 'Pod висит в ContainerCreating', принимая chown за сбой CSI
- Неправильный fsGroupPolicy драйвера ломает доступ приложения к данным после миграции тома

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

1.23 GA

fsGroupChangePolicy стабилен в securityContext Pod (beta с 1.20)

kubernetes.io — Security Context

1.26 GA

VOLUME_MOUNT_GROUP: CSI сам применяет fsGroup при mount, kubelet chown не делает

kubernetes-csi.github.io — FSGroup Support

O(files)

время recursive chown зависит от числа inode тома, а не от его объёма в ГБ

наш стандарт

1.36 GA

User Namespaces (hostUsers: false) — idmap тома вместо chown на диске

kubernetes.io — User Namespaces

3 режима

CSIDriver.fsGroupPolicy: None / File / ReadWriteOnceWithFSType — сверяем перед rollout

kubernetes.io — CSIDriver v1 (GA с 1.23)

1 раз

полный chown неизбежен только при первом провижининге пустого тома — закладываем в план

наш стандарт

РЕАЛИЗАЦИЯ 1 ИЗ 3

Аудит fsGroupPolicy CSI-драйвера перед включением OnRootMismatch

Что настраиваем

StatefulSet на CSI-томе ReadWriteOnce (ext4/xfs) от 200 ГБ и от сотен тысяч файлов

Как мы это делаем

1. kubectl get csidriver <name> -o jsonpath='{.spec.fsGroupPolicy}' — проверяем режим до правки Pod
2. Если File или ReadWriteOnceWithFSType — добавляем securityContext.fsGroupChangePolicy: OnRootMismatch
3. Проверяем NodeGetCapabilitiesResponse драйвера на VOLUME_MOUNT_GROUP — это отдаёт chown в CSI
4. Сравниваем время ContainerCreating до/после по kubectl get events --field-selector involvedObject.name
5. На новом томе выставляем целевой GID один раз при первом провижининге, а не на каждом старте

РЕЗУЛЬТАТ

Повторные старты Pod на уже размеченном томе перестают ждать полный chown — root каталога хватает, чтобы kubelet пропустил рекурсивный проход и Pod ушёл в Running за секунды, а не минуты.

КЛЮЧЕВОЙ НЮАНС

OnRootMismatch сверяет права ТОЛЬКО корня тома: если приложение меняет владельца отдельных подкаталогов вручную, root останется 'верным' и chown не подтянет их — проверяем консистентность заранее.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Перенос chown в CSI-драйвер: mount-time fsGroup вместо kubelet

Что настраиваем

Кластеры 1.26+ на CSI-драйверах с поддержкой VOLUME_MOUNT_GROUP (блочные и сетевые тома)

Как мы это делаем

1. Проверяем версию CSI-драйвера и его NodeServiceCapability: VOLUME_MOUNT_GROUP в спеке
2. kubelet передаёт fsGroup в NodeStageVolumeRequest/NodePublishVolumeRequest вместо цикла chown
3. Отключаем избыточный initContainer с busybox chown -R, который держали как обходной путь
4. Прогоняем canary StatefulSet на staging: сверяем владельца файлов внутри контейнера после старта
5. Фиксируем в Helm values.yaml требование fsGroupPolicy != None для этого storageClass

РЕЗУЛЬТАТ

kubelet вообще не трогает файлы тома — время примонтирования не зависит от числа inode, старт Pod становится предсказуемым независимо от того, сколько файлов накопилось на PVC за месяцы работы.

КЛЮЧЕВОЙ НЮАНС

Поддержка VOLUME_MOUNT_GROUP — capability драйвера, не гарантия по умолчанию: старые сборки CSI-плагина заявляют её в README, но не отдают в NodeGetCapabilities — проверяем вызовом, а не текстом.

РЕАЛИЗАЦИЯ 3 ИЗ 3

User Namespaces (hostUsers: false) как обход chown на 1.36

Что настраиваем

Пилотные workload без привилегий на узлах с ядром 6.3+ и containerd/runC с idmap

Как мы это делаем

1. Проверяем idmapped mounts на узле: ядро от 6.3 и ФС ext4/xfs/btrfs/overlayfs/tmpfs
2. Ставим hostUsers: false в PodSpec, оставляя fsGroup/runAsUser как обычно внутри контейнера
3. Кладём том через idmap-mount: владелец на диске не меняется, UID/GID транслирует ядро на лету
4. Сверяем инциденты: процесс в контейнере видит UID 0 как ожидается, хостовый UID остаётся непривилегированным
5. Оставляем OnRootMismatch как fallback для CSI-драйверов без поддержки VOLUME_MOUNT_GROUP/idmap

РЕЗУЛЬТАТ

Для новых нагрузок chown тома исчезает как проблема класса, а не как оптимизация: файлы физически не переписываются, а UID/GID транслируются ядром при каждом обращении к тому.

КЛЮЧЕВОЙ НЮАНС

hostUsers: false не заменяет fsGroupChangePolicy там, где ФС или CSI-плагин не умеют idmap: совмещаем обе настройки и проверяем совместимость драйвера.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Ждём эффект OnRootMismatch на новом томе**
  При первом провижининге root ещё не размечен под fsGroup — полный chown случится один раз, закладываем это в план миграции, а не считаем сбоем.
- ✕ **fsGroupPolicy: None игнорирует политику**
  Если CSIDriver.spec.fsGroupPolicy = None, fsGroupChangePolicy не действует вовсе — права выставляем сами через initContainer, иначе доступа не будет.
- ✕ **OnRootMismatch не видит правки подкаталогов**
  Проверяется только владелец корня тома: если владелец подпапок менялся вручную позже, chown их не подтянет — держим права консистентными от корня.
- ✕ **Ждём эффект на emptyDir/secret/configMap**
  fsGroupChangePolicy не применяется к эфемерным томам — оптимизация имеет смысл только для PVC/CSI-томов с персистентными данными.
- ✕ **supplementalGroupsPolicy Strict рвёт легаси**
  Fine-grained SupplementalGroups (GA с 1.35): Strict убирает подхват групп из /etc/group — легаси-образы теряют доступ, тестируем на staging.
- ✕ **Верим README драйвера, не capability-ответу**
  Поддержка VOLUME_MOUNT_GROUP заявлена в доках плагина, но не всегда реализована в сборке — проверяем ответ NodeGetCapabilities, а не README.
- ✕ **volumeMode: Block как автофикс**
  Блочный том убирает файловую систему и chown целиком, но приложение должно работать с raw-устройством — архитектурное решение, не быстрая настройка.
- ✕ **Путаем volumeBindingMode с fsGroup**
  WaitForFirstConsumer решает topology-aware провижининг тома, а не время рекурсивного chown — разные механизмы, один не лечит проблему другого.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- fsGroupChangePolicy: OnRootMismatch во всех Pod с PVC от 50k+ файлов
- kubectl get csidriver -o yaml — фиксируем fsGroupPolicy каждого драйвера
- Мониторим время ContainerCreating через kubectl events на каждом деплое

НОРМАЛЬНО

- Проверяем VOLUME_MOUNT_GROUP драйвера и отключаем ручной chown в initContainer
- Обновляем кластер до 1.26+, а CSI-драйверы — до сборок с VOLUME_MOUNT_GROUP
- Замеряем корреляцию числа inode тома со временем старта в перф-тестах перед релизом
- Документируем целевой GID тома на этапе первого провижининга в шаблоне StorageClass

ХОРОШО

- Пилотируем hostUsers: false (User Namespaces, 1.36 GA) на workload без привилегий
- Строим дашборд 'ContainerCreating > N с' по namespace с алертом на аномальный рост
- Тестируем supplementalGroupsPolicy: Strict (GA 1.35) в staging на легаси-образах
- Закладываем volumeMode: Block для нагрузок, которые сами управляют файловой системой

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] securityContext.fsGroupChangePolicy: OnRootMismatch выставлен для всех Pod с большим PVC?
- [ ] Проверен fsGroupPolicy используемого CSIDriver (None/File/ReadWriteOnceWithFSType)?
- [ ] Известно, поддерживает ли CSI-драйвер VOLUME_MOUNT_GROUP по факту, а не по README?
- [ ] Есть ли лишние initContainer с ручным chown -R, которые можно убрать?
- [ ] Замерено время ContainerCreating на проде до и после включения OnRootMismatch?
- [ ] Проверена совместимость легаси-образов с supplementalGroupsPolicy: Strict (GA 1.35)?
- [ ] Оценена применимость volumeMode: Block для конкретной нагрузки?
- [ ] Рассмотрен hostUsers: false для новых workload на узлах с поддержкой idmap?
- [ ] Есть алерт на аномальный рост времени ContainerCreating по namespace?

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).

- Аудитируем CSIDriver.fsGroupPolicy и securityContext всех Deployment/StatefulSet кластера
- Настраиваем и тестируем fsGroupChangePolicy: OnRootMismatch с замером ContainerCreating
- Обновляем CSI-драйверы до версий с GA-поддержкой mount-time fsGroup (VOLUME_MOUNT_GROUP)
- Пилотируем User Namespaces (hostUsers: false, GA в 1.36) на узлах с ядром 6.3+
- Строим мониторинг времени старта Pod и алерты на рост ContainerCreating по namespace

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 Configure a Security Context for a Pod or Container (fsGroupChangePolicy) (kubernetes.io — 1.23 GA)

02 CSIDriver v1 — spec.fsGroupPolicy (kubernetes.io — 1.23 GA)

03 FSGroup Support — VOLUME_MOUNT_GROUP (kubernetes-csi.github.io — 1.26 GA)

04 Persistent Volumes (volumeMode Block/Filesystem) (kubernetes.io — 2026)

05 Storage Classes (volumeBindingMode) (kubernetes.io — 2026)

06 Init Containers (kubernetes.io — 2026)

07 Pod Lifecycle (ContainerCreating) (kubernetes.io — 2026)

08 User Namespaces (hostUsers, idmapped mounts) (kubernetes.io — 1.36 GA)

Основано на официальной документации продуктов и нашей практике внедрения.