Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Flux CD: GitOps-доставка приложений в Kubernetes

Наша методология: bootstrap Flux 2.8, монорепо, SOPS-секреты, Helm-релизы и алерты деплоев

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

У заказчиков с 5+ сервисами в Kubernetes деплой «руками» через kubectl apply и самописные скрипты даёт дрейф stg/prod, неотслеживаемые правки в кластере и откаты «по памяти инженера». Мы переводим доставку на Flux CD: git-репозиторий — единственный источник истины, контроллеры сами доводят кластер до описанного состояния, откат — это git revert.

Почему это важно бизнесу

- Каждая ручная правка в кластере — недокументированное изменение: после сбоя конфигурацию никто не восстановит по памяти
- Дрейф stg/prod превращает релиз в лотерею: тестировали одну конфигурацию, в прод уехала другая
- Откат через git revert занимает минуты вместо часов простоя на «вспомнить, как было настроено»
- git log закрывает аудит и передачу дел: смена подрядчика или уход админа не парализует эксплуатацию
- Воспроизводимость кластера с нуля по репозиторию — это план аварийного восстановления, а не обещание

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

2.8

Версия Flux, которую разворачиваем: Helm v4 с server-side apply и CEL-healthchecks

fluxcd.io, релиз 24.02.2026

1 мин

interval GitRepository flux-system — быстрый подхват коммитов без перегрузки git-сервера

наш стандарт (bootstrap)

10 мин

interval боевых Kustomization: баланс скорости наката и нагрузки на API-сервер

наш стандарт

x3

install.remediation.retries в HelmRelease — авторемедиация сломанного релиза до эскалации

наш стандарт (API helm-controller v2)

3

минорных версий Kubernetes поддерживает Flux — апгрейды кластера планируем синхронно с ним

release policy fluxcd.io

v1

стабильные API GitRepository/Kustomization/OCIRepository; beta-CRD добиваем через flux migrate

по докам fluxcd.io 2.7+

РЕАЛИЗАЦИЯ 1 ИЗ 3

Bootstrap Flux и структура GitOps-монорепо

Что настраиваем

Кластер заказчика (managed K8s или наш стенд), репозиторий в GitLab/Gitea заказчика, каталог clusters/<env>

Как мы это делаем

1. flux check --pre — сверяем кластер с окном поддержки (три последних минорных K8s), затем flux bootstrap gitlab --path=./clusters/prod
2. Монорепо: clusters/{prod,stg} + infrastructure/ (ingress-nginx, cert-manager, monitoring) + apps/; окружения различаются только Kustomize-оверлеями
3. Два Kustomization: infra и apps с dependsOn на infra — приложения не применяются, пока не готовы CRD и ingress-класс
4. В prod включаем prune: true, wait: true, timeout: 5m и healthChecks по Deployment; interval 5m на stg и 10m на prod

РЕЗУЛЬТАТ

Кластер воспроизводим с нуля одной командой bootstrap: авария, переезд или новый стенд — это git clone и 20–30 минут реконсиляции, а не восстановление по памяти. Каждое изменение проходит через merge request с ревью, история наката полная.

КЛЮЧЕВОЙ НЮАНС

Порядок применения фиксируем только через dependsOn: без него CRD от cert-manager приходят позже своих потребителей и Kustomization уходит в retry-цикл на ровном месте.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Секреты через SOPS/age и управляемые Helm-релизы

Что настраиваем

Все Secret-манифесты репозитория + внешние чарты (ingress-nginx, cert-manager) через HelmRepository/HelmRelease

Как мы это делаем

1. age-keygen: приватный ключ — в Secret sops-age в flux-system, публичный — в .sops.yaml с creation_rules и encrypted_regex ^(data|stringData)$
2. В Kustomization включаем decryption: provider: sops — расшифровка на лету в кластере, в git лежит только шифротекст
3. HelmRelease v2: чарт пиним диапазоном версий, install.remediation.retries: 3, upgrade.remediation.retries: 2, driftDetection: mode: enabled
4. Резервную копию age-ключа кладём в офлайн-хранилище паролей — отдельно и от кластера, и от репозитория

РЕЗУЛЬТАТ

Секреты живут в том же git-потоке, что и манифесты: ревью, история, откат — без открытых паролей в репозитории. Сломанный Helm-апгрейд Flux откатывает сам по remediation, а ручные правки релиза в обход git затираются drift detection.

КЛЮЧЕВОЙ НЮАНС

encrypted_regex обязателен: без него SOPS шифрует YAML целиком, diff в merge request становится нечитаемым, и ревьюеры перестают смотреть изменения секретов.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Уведомления о деплоях и автообновление образов

Что настраиваем

notification-controller + image-reflector/image-automation на prod-кластере; алерты в Telegram-канал дежурной смены

Как мы это делаем

1. Provider type: telegram + Alert по всем Kustomization и HelmRelease — каждый накат и каждая ошибка реконсиляции видны в канале
2. ImageRepository сканирует registry каждые 5m, ImagePolicy отбирает теги по semver-диапазону, ImageUpdateAutomation коммитит бамп тега обратно в git
3. api.telegram.org из РФ-кластера недоступен, а провайдер telegram игнорирует поле address — трафик выводим через HTTP-прокси в spec.proxySecretRef провайдера
4. Диагностика по регламенту: flux get all -A, flux events --for, метрики контроллеров скрейпим Prometheus-ом

РЕЗУЛЬТАТ

Команда видит каждый деплой и каждый Failed в реальном времени, релизы приложений едут без участия инженера: от push тега в registry до обновлённого Deployment — один цикл реконсиляции, и сам бамп версии остаётся коммитом в истории.

КЛЮЧЕВОЙ НЮАНС

eventSeverity: info на живом кластере шумит — дежурным заводим отдельный Alert только на error, иначе канал замыливается и реальные сбои пропускаются.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **prune: true без страховки**
  Удалённый из git ресурс исчезает из кластера. Критичные объекты (PVC, namespace) помечаем аннотацией kustomize.toolkit.fluxcd.io/prune: disabled

- ✕ **Ручные правки в кластере**
  Flux затирает kubectl edit при следующей реконсиляции — «чинили, само сломалось». Все хотфиксы только коммитом, срочный накат — flux reconcile

- ✕ **Циклический dependsOn**
  Kustomization A ждёт B, B ждёт A — вечный Progressing. Строим явный граф infra → platform → apps без обратных рёбер

- ✕ **Деплой «зелёный», а поды лежат**
  Без wait и healthChecks Kustomization становится Ready сразу после apply. Включаем wait: true, для кастомных ресурсов — CEL-выражения в healthCheckEx...

- ✕ **Потеря age-ключа SOPS**
  Ключ только в кластере = после аварии секреты нечитаемы. Копию храним в офлайн-хранилище паролей, отдельно от git и кластера

- ✕ **Устаревшие beta-API CRD**
  v1beta1/v2beta1 удалены из CRD в Flux 2.7, v1beta2 — в 2.8: перед апгрейдом прогоняем flux migrate, иначе контроллеры перестанут видеть ресурсы

- ✕ **Слишком частые интервалы**
  interval: 1m на десятках Kustomization давит на git-сервер и API. Источник — 1m, применение — 5–10m; срочное — flux reconcile

- ✕ **Один Kustomization на всё**
  Ошибка в любом манифесте блокирует весь накат. Дробим по доменам: меньше blast radius, независимые ретраи и healthChecks

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Flux 2.8 bootstrap в prod, монорепо clusters/<env>, деплой только через git
- SOPS + age для всех Secret, резервная копия ключа вне репозитория и кластера
- prune: true, wait: true и healthChecks на всех боевых Kustomization

НОРМАЛЬНО

- Разбивка infra/apps с dependsOn, Kustomize-оверлеи для stg и prod
- HelmRelease с remediation.retries и driftDetection для всех чартов
- Alert на error-события всех Kustomization и HelmRelease дежурным
- Регламент апгрейда Flux: flux migrate + окно трёх минорных версий K8s

ХОРОШО

- Image automation: ImagePolicy по semver и автокоммит бампа тегов в git
- CEL-healthchecks (healthCheckExprs) для кастомных ресурсов, метрики контроллеров в P…
- Поток изменений: MR → авто-деплой в stg → промоушен в prod через git
- OCIRepository для артефактов с проверкой подписи cosign (spec.verify)

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Все изменения в кластере проходят через git-коммит, прямой kubectl apply на проде исключён?
- [ ] flux check проходит чисто, версия Flux и кластера — в окне поддержки трёх минорных K8s?
- [ ] Секреты в репозитории зашифрованы SOPS, резервная копия age-ключа лежит вне кластера и вне git?
- [ ] На боевых Kustomization включены prune, wait и healthChecks — «зелёный» деплой значит живые поды?
- [ ] Порядок infra → apps закреплён через dependsOn и в графе нет циклов?
- [ ] У каждого HelmRelease настроен remediation.retries — сломанный апгрейд откатывается без дежурного?
- [ ] Ошибки реконсиляции прилетают в канал дежурных, а не лежат молча в flux events?
- [ ] Восстановление кластера с нуля по репозиторию проверено учениями на чистом стенде?
- [ ] Перед апгрейдом Flux прогоняется flux migrate для устаревших API?
- [ ] Бамп версий приложений автоматизирован (image automation) и виден в истории git?

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).

- Внедрение Flux под ключ: bootstrap, структура монорепо, перевод манифестов на Kustomize и Helm за 3–5 дней
- Миграция секретов на SOPS/age или External Secrets Operator с регламентом хранения и ротации ключей
- Алерты деплоев в Telegram/Slack (включая обход блокировок из РФ-кластеров) и метрики Flux в ваш мониторинг
- Обучение команды и runbook: как катить, как откатывать, как разбирать Failed-реконсиляции
- Сопровождение: апгрейды Flux по циклу CNCF, flux migrate, регулярное ревью GitOps-репозитория

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 Flux Documentation — Get Started / Bootstrap (flux2) (fluxcd.io — 2.8)

02 Kustomize Controller — Kustomization API (fluxcd.io — v1)

03 Helm Controller — HelmRelease API (fluxcd.io — v2)

04 Flux Guides — Manage Kubernetes secrets with SOPS (fluxcd.io — 2026)

05 Notification Controller — Provider / Alert API (fluxcd.io — v1beta3)

06 Image Automation Controllers — ImagePolicy, ImageUpdateAutomation (fluxcd.io — v1)

07 Flux release policy — supported versions (fluxcd.io — 2026)

08 Наш шаблон GitOps-монорепо clusters/{env} + infrastructure + apps (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.