Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

FluxCD: GitOps-доставка в Kubernetes без Jenkins

Как мы строим pull-доставку на Flux 2.9: bootstrap, Kustomization, image automation, SOPS

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Деплой через Jenkins и ручной kubectl apply превращает кластер в «чёрный ящик»: kubeconfig лежит в CI, состояние расходится с Git, откат — перезапуск pipeline, падение CI останавливает релизы. Мы переводим доставку на pull-модель Flux 2.9: контроллеры внутри кластера сами сверяют его с Git и правят дрейф — Jenkins уходит из контура деплоя.

Почему это важно бизнесу

- Релиз ускоряется с 10–15 минут до 1–3: короче окно простоя и быстрее выкатка исправлений безопасности
- Откат = git revert: авария лечится за минуты, а не поиском «кто и что руками накатил»
- kubeconfig с правами admin не хранится в CI — компрометация раннера не отдаёт кластер
- Git-история — полный аудит изменений прод-среды для разбора инцидентов и проверок
- CI-сервер перестаёт быть единой точкой отказа доставки: контроллеры живут в кластере

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

2.9

разворачиваем только поддерживаемую ветку Flux: K8s 1.34–1.36, OpenShift 4.21; 2.6 и старше —...

по докам Flux v2.9

1m / 10m

опрос Git раз в 1m (GitRepository), сверка Kustomization раз в 10m — отклик без шторма API

наш стандарт

prune: true

включаем везде: ресурс, удалённый из Git, удаляется из кластера — никаких «забытых» объектов

kustomize-controller v1

wait + 5m

wait: true + timeout: 5m — релиз «зелёный» только после прохождения health-check всех ресурсов

по докам Kustomization v1

v1 / v2

стабильные API: GitRepository/Kustomization/ImagePolicy — v1, HelmRelease — v2; никаких beta

по докам Flux v2.9

5% → 50%

Flagger-canary: stepWeight 5, maxWeight 50, анализ раз в 1m, откат после 5 фейлов метрик

по докам Flagger

РЕАЛИЗАЦИЯ 1 ИЗ 3

Bootstrap и каркас GitOps-репозитория

Что настраиваем

Кластеры prod/staging клиента (10–30 микросервисов); репозиторий k8s-gitops в GitLab или GitHub

Как мы это делаем

1. flux check --pre → flux bootstrap gitlab --path=clusters/production: контроллеры в ns flux-system, gotk-components.yaml + gotk-sync.yaml — в репозитории
2. Каркас: apps/ (base + overlays), infrastructure/ (ingress-nginx, cert-manager как HelmRelease v2), clusters/<env>/ — по Kustomization на слой
3. dependsOn: apps ждут infrastructure; на слоях — healthChecks по Deployment, wait: true, timeout: 5m, retryInterval: 2m
4. Секреты — SOPS + age: ключ в Secret sops-age (ns flux-system), в Kustomization — spec.decryption.provider: sops; в Git только шифротекст

РЕЗУЛЬТАТ

Кластер поднимается «с нуля» одним bootstrap: любое изменение проходит через merge request, состояние воспроизводимо из Git. Ручные kubectl apply перестают быть источником правды — дрейф Flux откатывает на ближайшей сверке.

КЛЮЧЕВОЙ НЮАНС

Порядок применения задаём только через dependsOn + healthChecks, а не интервалами: без wait: true Kustomization становится Ready до готовности подов, и зависимые слои стартуют раньше времени.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Image automation: релиз без нажатия кнопки

Что настраиваем

Контур доставки образов: registry клиента (Harbor/GitLab Registry) → Git → кластер; CI остаётся только сборкой

Как мы это делаем

1. ImageRepository (interval: 5m) сканирует registry; ImagePolicy выбирает тег по semver range ">=1.0.0 <2.0.0" — никаких latest
2. В deployment.yaml ставим маркер # {"$imagepolicy": "flux-system:orders"} — стратегия Setters правит только помеченные строки
3. ImageUpdateAutomation коммитит в ветку flux-image-updates с messageTemplate; в prod тег заезжает через merge request, в staging — сразу в main
4. Коммиты автоматики подписываем SSH-ключом (verify в GitRepository, Flux 2.9) — в истории видно, что менял робот

РЕЗУЛЬТАТ

Цепочка CI build → push в registry → новый тег в Git → выкладка работает без человека в staging и с одним ревью в prod; от сборки до кластера — минуты, каждая версия зафиксирована коммитом.

КЛЮЧЕВОЙ НЮАНС

ImagePolicy принимает только строгий semver-диапазон: теги «latest»/«build-123» ломают автоматику, поэтому схему тегов в CI приводим к vX.Y.Z до включения image automation.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Наблюдаемость, дрейф и аварийный контур

Что настраиваем

Эксплуатация после внедрения: notification-controller, метрики Prometheus, регламент ручных вмешательств

Как мы это делаем

1. Provider (telegram/slack) + Alert с eventSeverity: error из notification-controller: падение reconcile прилетает дежурному в канал
2. Метрики контроллеров и статусы CR — в Prometheus (kube-state-metrics, gotk_resource_info); алерт: Ready=False дольше 10 минут
3. Регламент хотфикса: flux suspend kustomization apps → ручная правка → flux resume; suspend фиксируем в тикете и снимаем в тот же день
4. Диагностика одним набором: flux get all -A, flux events, flux tree kustomization apps; форс-сверка — flux reconcile kustomization apps --with-source

РЕЗУЛЬТАТ

Дрейф и падения выката видны за минуты, а не при следующем инциденте; ручные правки живут максимум до конца сверки либо оформлены suspend'ом с тикетом. Эксплуатация получает один инструмент диагностики вместо раскопок в Jenkins-логах.

КЛЮЧЕВОЙ НЮАНС

flux suspend не имеет таймера: забытая пауза отключает GitOps насовсем. Вешаем алерт на suspended-ресурсы (gotk_resource_info из kube-state-metrics) и проверяем их в еженедельном обходе.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **prune удаляет живые ресурсы**
  При переносе манифеста между Kustomization prune сносит объект; критичное помечаем kustomize.toolkit.fluxcd.io/prune: disabled и переносим в два MR

- ✕ **Секреты в Git открытым текстом**
  Любой с доступом к репо получает прод-пароли; шифруем SOPS+age до первого коммита: публичный ключ в репо, приватный — только в Secret кластера

- ✕ **Flux затирает хотфиксы**
  Ручной kubectl apply откатится на ближайшей сверке — это не баг, а auto-remediation; срочные правки только через flux suspend или коммит в Git

- ✕ **Война с HPA за replicas**
  Если replicas задан в Git, Flux и HPA перезаписывают друг друга; убираем поле из манифеста либо исключаем его через spec.ignore в Kustomization (Flux…

- ✕ **latest вместо semver**
  ImagePolicy не выбирает «последний latest»: без vX.Y.Z-тегов автоматика молчит; схему тегирования в CI меняем до включения image automation

- ✕ **Один interval на всё**
  GitRepository 1m и Kustomization 10m — разные ручки: короткий interval на больших репо даёт шторм клонирований и троттлинг Kubernetes API

- ✕ **beta-API в манифестах**
  image.toolkit v1beta2 и notification v1beta2 удалены в 2.9; перед апгрейдом мигрируем CR и валидируем манифесты плагином flux schema

- ✕ **Смена дефолта Helm-хуков в 2.9**
  Post-rendering по умолчанию сменился nohooks → combined: чарты с upgrade/test-хуками ведут себя иначе; фиксируем стратегию явно в HelmRelease

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Flux 2.9 через flux bootstrap, отдельный репозиторий k8s-gitops, деплой только из Git
- prune: true, wait: true, timeout: 5m на каждой Kustomization
- SOPS + age для секретов до первого коммита манифестов
- flux check в мониторинге; прямой push в main закрыт, только merge request

НОРМАЛЬНО

- Слои infrastructure/apps с dependsOn и healthChecks, overlays на staging/prod
- Alert из notification-controller в Telegram/Slack дежурному
- Image automation в staging: ImagePolicy semver + коммиты робота в main
- Метрики контроллеров в Prometheus, алерт на Ready=False дольше 10 минут

ХОРОШО

- Prod-теги через MR из ветки flux-image-updates, подпись и verify коммитов
- Flagger-canary на ключевых сервисах: stepWeight 5, maxWeight 50, откат по метрикам
- Multi-cluster: clusters/dev|staging|prod из одного репо, общее — через overlays
- Multi-tenancy: --no-cross-namespace-refs и serviceAccountName на Kustomization

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Кластер разворачивается с нуля из Git одним flux bootstrap — проверяли на стенде, а не в теории?
- [ ] Все изменения прод-манифестов идут через merge request, прямые push в main закрыты?
- [ ] prune и wait включены на всех Kustomization, критичные ресурсы защищены от prune аннотацией?
- [ ] В Git нет ни одного секрета открытым текстом, приватный age-ключ есть в офлайн-резерве?
- [ ] Падение reconcile прилетает дежурному в мессенджер, а не обнаруживается при инциденте?
- [ ] Есть регламент хотфикса: suspend → правка → resume, и алерт на забытые suspended-ресурсы?
- [ ] Теги образов — строгий semver, image automation не ждёт «latest»?
- [ ] Версия Flux в поддерживаемом окне (2.7+), процедура апгрейда расписана и отрепетирована?
- [ ] Откат релиза отрепетирован: git revert доводит кластер до прежней версии за минуты?

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).

- Разворачиваем Flux 2.9 под ключ: bootstrap, каркас репозитория, SOPS, стыковка с вашим CI
- Переносим деплой с Jenkins/скриптов на pull-модель без остановки релизов
- Настраиваем image automation и canary через Flagger под ваши сервисы и метрики
- Заводим мониторинг Flux в Prometheus/Grafana и алерты в Telegram дежурному
- Берём GitOps-контур на поддержку: апгрейды Flux, ревью манифестов, разбор инцидентов

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 — Core Concepts, Bootstrap (fluxcd.io — v2.9)

02 Kustomize Controller — Kustomization API (prune, wait, healthChecks) (fluxcd.io — v1)

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

04 Flux Security — Manage Kubernetes secrets with SOPS and age (fluxcd.io — v2.9)

05 Flux Monitoring — Prometheus metrics, kube-state-metrics (fluxcd.io — v2.9)

06 Flagger — Canary analysis: How it works (docs.flagger.app — 2026)

07 Наш шаблон k8s-gitops: apps/infrastructure/clusters + SOPS (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.