Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

GitOps 2.0: ArgoCD или Flux v2 — как мы строим доставку

Разбираем архитектуру обоих стеков и наш регламент внедрения GitOps в кластеры заказчиков

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Релизы в Kubernetes у заказчика едут вручную: kubectl apply и helm upgrade с ноутбуков инженеров, конфигурация кластера расходится с репозиторием, и никто точно не скажет, что сейчас крутится в проде. Откат — археология по чатам, уход DevOps-инженера — потеря процесса. Мы переводим доставку на GitOps: единственный источник истины — Git, кластер сводит себя к нему сам.

Почему это важно бизнесу

- Простой при неудачном релизе: откат через git revert занимает минуты, вручную — часы
- Аудит для ИБ и заказчиков: каждая правка прода — коммит с автором, датой и review
- Нет зависимости от одного админа: процесс деплоя описан кодом, а не в голове
- DR: кластер восстанавливается bootstrap-ом из Git, а не по памяти инженера
- Меньше человеческих ошибок: рука в прод не ходит, selfHeal убирает ручной drift

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

3.5

текущая ветка Argo CD: ставим из HA-манифестов, обновляем строго по minor-цепочке — патчи полу…

Argo CD 3.5, argo-cd.readthedocs.io

2.9

текущая ветка Flux: image automation GA с 2.7, Helm v4 — с 2.8; ставим командой flux bootstrap

Flux 2.9, fluxcd.io

180 с

timeout.reconciliation в argocd-cm — период опроса Git; вниз не крутим, подключаем webhook

по докам Argo CD

1м / 10м

интервалы GitRepository и Kustomization после bootstrap: источник — 1 мин, применение — 10 мин

по докам Flux

6

контроллеров Flux: source, kustomize, helm, notification, image-reflector, image-automation

Flux 2.9, fluxcd.io

5

retry.limit в syncPolicy автосинка: пять попыток с экспоненциальным backoff до алерта дежурному

наш стандарт

РЕАЛИЗАЦИЯ 1 ИЗ 3

Argo CD: HA-ядро, SSO и multi-tenancy

Что настраиваем

Management-кластер заказчика: argocd-server, repo-server, application-controller, Redis HA

Как мы это делаем

1. Ставим из manifests/ha/install.yaml в namespace argocd: по 2 реплики server и repo-server, application-controller как StatefulSet, Redis HA с sentinel
2. Подключаем SSO через Dex/OIDC (Keycloak), в argocd-rbac-cm пишем policy.csv: роли по проектам и группам, default-роль — readonly
3. Каждой команде — свой AppProject с белым списком sourceRepos, destinations и типов ресурсов; project: default в проде запрещён
4. Автосинк включаем поэтапно: syncPolicy.automated с prune и selfHeal, retry.limit 5; окна деплоя режем через syncWindows проекта
5. Порядок выката задаём аннотацией argocd.argoproj.io/sync-wave и хуками PreSync/PostSync: CRD и миграции БД — в ранних волнах

РЕЗУЛЬТАТ

Единая точка управления релизами с UI: видно health и sync каждого Application, откат — git revert или History & Rollback. Команды изолированы проектами, доступ по SSO-группам, все операции в audit-логе argocd-server. Ручной kubectl в прод больше не нужен.

КЛЮЧЕВОЙ НЮАНС

С ветки 3.x RBAC ужесточён по умолчанию — политики прописываем явно до миграции. timeout.reconciliation не занижаем: мгновенную реакцию даёт webhook Git-провайдера на /api/webhook, а не частый поллинг.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Flux v2: bootstrap, fleet-репозиторий, image automation

Что настраиваем

Кластеры без выделенного management-звена: namespace flux-system + репозиторий fleet-config

Как мы это делаем

1. flux bootstrap github --path=clusters/<имя>: Flux сам коммитит манифесты контроллеров в Git и дальше обновляет себя из репозитория
2. Раскладываем репо: apps/base + apps/<env> как Kustomize-overlays, infrastructure/ — отдельной Kustomization, связка через dependsOn
3. В Kustomization включаем prune, wait, timeout 5m и healthChecks — CRD и операторы гарантированно встают раньше приложений
4. HelmRelease (helm.toolkit.fluxcd.io/v2): driftDetection.mode=enabled, remediation.retries=3 с автоматическим rollback релиза
5. Image automation: ImageRepository сканирует registry, ImagePolicy отбирает тег по semver-range, ImageUpdateAutomation коммитит его в Git

РЕЗУЛЬТАТ

Кластер полностью описан в Git: новая площадка поднимается bootstrap-ом за десятки минут, обновления образов едут без человека — от registry до коммита. Drift по Helm-релизам ловит driftDetection, откаты неудачных релизов выполняет remediation.

КЛЮЧЕВОЙ НЮАНС

Берём только GA-API: GitRepository v1, Kustomization v1, HelmRelease v2, image.toolkit v1 — beta-CRD в манифестах не оставляем, перед апгрейдом ветки гоняем flux migrate. Для multi-tenancy включаем --no-cross-…

РЕАЛИЗАЦИЯ 3 ИЗ 3

Масштабирование и наблюдаемость GitOps-контура

Что настраиваем

Парк кластеров: ApplicationSet в Argo CD, шардирование контроллера, метрики gotk_*/argocd_* в Prometheus

Как мы это делаем

1. ApplicationSet с генераторами cluster+git: приложение на каждый кластер по matchLabels, values-файл подставляется из label окружения
2. При десятках кластеров шардируем application-controller: кластеры делятся между репликами StatefulSet, алгоритм — ARGOCD_CONTROLLER_SHARDING_ALGORITHM=consistent-ha…
3. Prometheus собирает argocd_app_info и gotk_resource_info (customResourceState в kube-state-metrics); алерты: OutOfSync дольше 10 мин, ready=False, рост длительности…
4. Вместо агрессивного поллинга — вебхуки: /api/webhook у argocd-server, Receiver у notification-controller Flux
5. События синка шлём дежурному: argocd-notifications и Alert/Provider Flux в Telegram-канал эксплуатации

РЕЗУЛЬТАТ

Один шаблон обслуживает весь парк: добавление кластера — это регистрация и labels, приложения приезжают сами. Деградацию синка видим в Grafana раньше жалоб пользователей, дежурный получает алерт с именем Application и причиной.

КЛЮЧЕВОЙ НЮАНС

Смотрим не «успех последнего синка», а время нахождения вне Synced/Ready: gotk_resource_info{ready="False"} и sync_status у argocd_app_info — первичные SLI GitOps-контура.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Два GitOps-движка на одних ресурсах**
  Argo CD и Flux спорят за одни объекты и бесконечно перекатывают их. Делим зоны ответственности: один движок на кластер или на класс ресурсов.

- ✕ **Секреты закоммичены в Git**
  GitOps тянет всё из репо — секрет попадает в историю навсегда. Ставим External Secrets Operator или SOPS+age, в Git только шифртекст.

- ✕ **selfHeal против ручных hotfix**
  Контроллер молча откатывает kubectl edit: инженер «чинит», через минуты всё возвращается. Правки — только через Git, авралы — через sync windows.

- ✕ **prune без защитных аннотаций**
  prune=true сносит ресурс, случайно выпавший из рендера. На CRD, namespace и PVC вешаем Prune=false и kustomize.toolkit.fluxcd.io/prune: disabled.

- ✕ **Интервалы 30 с на всех источниках**
  Десятки GitRepository с interval 30s упираются в rate limit Git-провайдера. Источникам — 1–5 мин, мгновенность дают webhook и Receiver.

- ✕ **repo-server без лимитов ресурсов**
  helm template и kustomize build на больших монорепо съедают память repo-server. Ставим limits и reposerver.parallelism.limit, монорепо дробим.

- ✕ **Все приложения в project: default**
  Любая команда может деплоить в любой кластер и namespace. С первого дня нарезаем AppProject с ограничением destinations и sourceRepos.

- ✕ **Kustomization без wait и healthChecks**
  Flux считает apply успехом до готовности подов — зависимые релизы падают. Включаем wait: true, timeout и dependsOn между слоями.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Один движок на кластер: Argo CD 3.5 или Flux 2.9, установка из официальных манифестов
- Весь деплой — только из Git: ветка main защищена, review обязателен
- Секреты вне Git: SOPS+age или External Secrets Operator

НОРМАЛЬНО

- Автосинк с prune/selfHeal и retry.limit, порядок — sync waves или dependsOn
- SSO по OIDC и RBAC по командам: AppProject либо tenant-namespace во Flux
- Webhook вместо частого поллинга, метрики контроллеров в Prometheus с алертами

ХОРОШО

- Multi-cluster: ApplicationSet по labels кластеров, шардирование контроллера
- Image automation: ImagePolicy по semver, автокоммит новых тегов в Git
- Progressive delivery: Argo Rollouts или Flagger с анализом метрик canary
- DR-регламент: подъём кластера с нуля bootstrap-ом из Git за контрольное время

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Можно ли восстановить кластер с нуля только из Git-репозитория, без ручных kubectl apply?
- [ ] Защищена ли ветка main: обязательный review, запрет force-push, подписанные коммиты?
- [ ] Проверяли ли историю Git сканером секретов — нет ли там паролей и токенов открытым текстом?
- [ ] Ограничено ли, кто и куда деплоит: destinations и sourceRepos нарезаны по командам?
- [ ] Придёт ли алерт, если приложение висит OutOfSync или Ready=False дольше 10 минут?
- [ ] Задан ли порядок выката (sync waves / dependsOn), чтобы CRD и операторы вставали первыми?
- [ ] Сколько минут занимает откат неудачного релиза и когда это проверяли на практике?
- [ ] Обновляются ли сами GitOps-контроллеры по регламенту: minor-цепочка, CVE-патчи?
- [ ] Есть ли sync windows: может ли автосинк выкатить релиз в ночь перед отчётным днём?

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 процесса деплоя и выбор движка: Argo CD или Flux под ваш кластер и команду
- Внедрение под ключ: HA-установка, SSO/RBAC, структура Git-репозитория, sync-политики
- Перевод секретов на External Secrets Operator или SOPS без простоя приложений
- Мониторинг GitOps-контура: drift, ошибки синка, алерты дежурному в Telegram
- Сопровождение: обновления контроллеров, DR-учения bootstrap-из-Git, 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 Argo CD Operator Manual: High Availability, RBAC, Sync Options (argo-cd.readthedocs.io — 3.5)

02 Argo CD ApplicationSet Controller: Generators (argo-cd.readthedocs.io — 3.5)

03 Flux Documentation: Kustomization и HelmRelease API (fluxcd.io — 2.9)

04 Flux Guide: Image Update Automation (GA) (fluxcd.io — 2.7+)

05 Flux: Multi-tenancy lockdown и flux bootstrap (fluxcd.io — 2.9)

06 Шаблон ITfresh: структура fleet-репозитория GitOps (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.