Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Self-hosted GitLab CE 19: как мы ставим корпоративный git-сервер

Omnibus-инсталляция, вход из AD, CI-раннеры, бэкапы и регламент обновлений — наша методика

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Код и пайплайны компании живут в облачных сервисах или на ноутбуках разработчиков: аккаунт могут заблокировать, лицензии дорожают, уволившийся сотрудник уносит единственную копию. Мы переносим это на self-hosted GitLab CE: репозитории, code review, CI/CD и реестр контейнеров на контролируемом сервере — с доменным входом, бэкапом в две части и регламентом обновлений.

Почему это важно бизнесу

- Исходный код — актив компании: потеря репозитория с историей не компенсируется, восстановление с нуля стоит месяцы разработки
- Блокировка облачного аккаунта (санкции, оплата) останавливает разработку в один день — свой сервер этот риск снимает
- GitLab CE бесплатен: платите за одну ВМ и сопровождение вместо подписки за каждого разработчика
- Доступ к коду завязан на AD: увольнение = блокировка доменной учётки, отдельные аккаунты не забываются
- Без регламента бэкапов и обновлений git-сервер сам становится единой точкой отказа всей разработки

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

19.3

текущий релиз GitLab CE — патчи безопасности выходят только для трёх последних минорных веток

GitLab 19.3, релиз 08.2026

8 vCPU/16ГБ

single-node до 1000 юзеров (20 RPS) по reference architecture; командам до 50 человек — с запа…

по докам GitLab, ref arch 1k

25%

postgresql['shared_buffers'] от ОЗУ, потолок 8 ГБ — выше выигрыша нет, память отнимается у Puma

наш стандарт по докам Omnibus

2 части

полный бэкап = дамп gitlab-backup create + /etc/gitlab с gitlab-secrets.json; без секретов дам…

по докам GitLab Backup/Restore

636/LDAPS

encryption 'simple_tls' к контроллеру AD, uid sAMAccountName; plaintext 389 не используем даже…

по докам GitLab LDAP

604800 с

backup_keep_time — локальные дампы храним 7 суток, длинная ретенция только на внешнем сторадже

gitlab.rb, наш стандарт

РЕАЛИЗАЦИЯ 1 ИЗ 3

Развёртывание Omnibus на single-node

Что настраиваем

ВМ Ubuntu 24.04 LTS у клиента или на нашей площадке: 8 vCPU / 16 ГБ / SSD 500 ГБ + swap 8 ГБ, DNS-имя git.company.ru

Как мы это делаем

1. Ставим gitlab-ce из официального репозитория (script.deb.sh), инициализация через EXTERNAL_URL=https://git.company.ru — gitlab-ctl reconfigure собирает весь стек
2. TLS: letsencrypt['enable']=true с auto_renew; во внутренних сетях без 80/443 наружу — сертификат своего AD CS в nginx['ssl_certificate']
3. Почта: gitlab_rails['smtp_*'] на корпоративный релей, STARTTLS 587; gitlab_email_from — реальный служебный ящик, покрытый SPF/DKIM
4. Тюнинг: puma['worker_processes'] по ядрам, sidekiq['max_concurrency']=20, time_zone Europe/Moscow; финальная проверка gitlab-rake gitlab:check
5. Сразу меняем пароль из /etc/gitlab/initial_root_password, включаем 2FA для root и закрываем открытую регистрацию (sign-up restrictions)

РЕЗУЛЬТАТ

Через 2–3 часа у клиента рабочий git-сервер по HTTPS с автопродлением сертификата и почтовыми нотификациями. Один узел по reference architecture держит до 1000 пользователей — команде до 50 человек этого запаса хватает на годы без апгрейда железа.

КЛЮЧЕВОЙ НЮАНС

Все параметры живут только в /etc/gitlab/gitlab.rb и применяются gitlab-ctl reconfigure: ручные правки конфигов внутренних сервисов Omnibus перезатирает при первом же reconfigure.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Вход из Active Directory, CI-раннеры и Container Registry

Что настраиваем

GitLab-нода + отдельная ВМ под docker-раннер (4 vCPU / 8 ГБ) + контроллер домена клиента

Как мы это делаем

1. LDAP: gitlab_rails['ldap_servers'] — host DC, port 636, simple_tls, uid 'sAMAccountName', bind-учётка с минимальными правами, user_filter по группе Developers
2. verify_certificates=true: корневой сертификат AD CS кладём в /etc/gitlab/trusted-certs/, самоподписанные исключения не допускаем
3. Раннер: gitlab-runner register с authentication-токеном glrt-* из UI, executor docker, дефолтный образ из локального зеркала; раннер никогда не ставим на GitLab-ноду
4. Registry: registry_external_url 'https://git.company.ru:5050' + cleanup policy на старые теги, чтобы образы не съели диск
5. Privileged-режим docker-executor включаем только выделенному раннеру для доверенных проектов (сборка образов), остальным — запрещён

РЕЗУЛЬТАТ

Разработчики входят доменной учёткой, доступ к коду централизованно закрывается блокировкой в AD. Пайплайны собираются на отдельной ВМ и не конкурируют с GitLab за CPU/IO, образы уезжают в собственный registry вместо внешних хабов.

КЛЮЧЕВОЙ НЮАНС

Регистрация раннеров теперь только через authentication token glrt-* (создаётся в UI Runners): старые registration tokens в актуальных версиях отключены — это первое, что сверяем с докой при апгрейде CI.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Контур бэкапов и регламент обновлений

Что настраиваем

GitLab-нода + внешний бэкап-узел (rsync/restic или S3-совместимый сторадж), сервисное окно 02:00–04:00

Как мы это делаем

1. Крон 02:00: gitlab-backup create CRON=1 STRATEGY=copy — БД, репозитории, uploads, artifacts; 02:30 — tar /etc/gitlab вместе с gitlab-secrets.json
2. Выгрузка наружу restic'ом на независимый узел; локально держим 7 суток (backup_keep_time=604800), длинная ретенция — только вне сервера
3. Раз в квартал — тестовое восстановление на чистой ВМ: gitlab-backup restore + возврат /etc/gitlab, версия пакета строго та же, что у дампа
4. Обновления: маршрут сверяем по официальному Upgrade Path (обязательные required stops), между шагами ждём завершения background migrations
5. Пакет закрыт apt-mark hold gitlab-ce — минорные патчи накатываем вручную раз в месяц в сервисное окно

РЕЗУЛЬТАТ

Инстанс восстанавливается на чистой ВМ за предсказуемое время: дамп + секреты + та же версия пакета. Обновления перестают быть лотереей — маршрут проверен, окно согласовано, план отката — restore вчерашнего дампа.

КЛЮЧЕВОЙ НЮАНС

Restore работает только в ту же версию GitLab, из которой снят дамп, поэтому фиксируем версию в имени архива. STRATEGY=copy убирает ошибки «file changed as we read it» при бэкапе живого инстанса.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Пропуск обязательных остановок апгрейда**
  Прыжок через минорные версии ломает миграции БД. Идём строго по Upgrade Path через required stops и ждём background migrations между шагами.
- ✕ **gitlab-secrets.json вне бэкапа**
  Шифрованные поля (токены, 2FA, CI-переменные) без секретов не расшифровать — restore формально пройдёт, данные потеряны. /etc/gitlab бэкапим всегда о…
- ✕ **Ожидание встроенной Grafana**
  Из Omnibus её убрали в 16.3 (депрекация с 16.0). Остался Prometheus (:9090) — дашборды поднимаем во внешней Grafana либо снимаем метрики нашим Zabbix.
- ✕ **Автообновление пакета gitlab-ce**
  unattended-upgrades может утащить инсталляцию через несколько версий разом и положить миграции. Ставим apt-mark hold и обновляем только по регламенту.
- ✕ **Раннер на самой GitLab-ноде**
  docker-executor с privileged на хосте GitLab даёт джобам путь к секретам инстанса и конкуренцию за IO. Раннеры — только на отдельных ВМ.
- ✕ **Почта GitLab уходит в спам**
  Слать напрямую с ноды бессмысленно — нет SPF/DKIM. Шлём через корпоративный релей, gitlab_email_from заводим как реальный ящик почтовой системы.
- ✕ **Медленный git push списывают на GitLab**
  Чаще виноват сторадж: смотрим iostat и латентность раздела репозиториев; репы держим на отдельном SSD с noatime.
- ✕ **Registry не включён на старте**
  По умолчанию docker push некуда — разработчики тащат образы во внешние хабы. Включаем registry_external_url сразу и настраиваем cleanup policy.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Omnibus на отдельной ВМ 4 vCPU / 8 ГБ, HTTPS через Let's Encrypt, swap 4–8 ГБ
- Ночной gitlab-backup create + архив /etc/gitlab с выгрузкой на внешний узел
- apt-mark hold gitlab-ce, обновления вручную раз в месяц
- Смена initial_root_password, 2FA для root, регистрация пользователей закрыта

НОРМАЛЬНО

- 8 vCPU / 16 ГБ, репозитории на отдельном SSD с noatime
- Вход через AD: LDAPS 636, bind-учётка с минимальными правами, фильтр по группе
- Docker-раннер на отдельной ВМ + Container Registry с cleanup policy
- Метрики встроенного Prometheus заведены во внешний мониторинг с алертами

ХОРОШО

- Ретенция бэкапов на S3-совместимом сторадже + квартальный тест восстановления
- Регламент апгрейдов по Upgrade Path с контролем background migrations
- Object storage (MinIO/S3) под artifacts и LFS вместо локального диска
- Документированный DR-план: процедура restore с целевым RTO 4 часа

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] GitLab доступен только по HTTPS и сертификат продлевается автоматически (Let's Encrypt или свой PKI)?
- [ ] Бэкап состоит из двух частей — дамп данных И /etc/gitlab с gitlab-secrets.json?
- [ ] Копии бэкапов лежат вне GitLab-сервера, восстановление реально проверялось за последний квартал?
- [ ] Пакет gitlab-ce закрыт от автообновления, апгрейды идут строго по Upgrade Path?
- [ ] Сотрудники входят доменной учёткой, блокировка в AD сразу закрывает доступ к коду?
- [ ] CI-раннеры вынесены с GitLab-ноды, privileged-режим ограничен доверенными проектами?
- [ ] Container Registry включён и cleanup policy чистит старые образы?
- [ ] Метрики Prometheus (:9090) заведены во внешний мониторинг с алертами на диск и очереди Sidekiq?
- [ ] У root сменён initial_root_password и включена 2FA?
- [ ] Известно, на какой версии снят последний дамп, и есть план отката на неё?

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 GitLab CE под ключ: сайзинг, Omnibus, TLS, вход из AD, раннеры, registry — на вашей ВМ или нашем жел…
- Мигрируем с GitHub/Bitbucket/Gitea: репозитории с полной историей, пайплайны переписываем на GitLab CI
- Строим контур бэкапов с внешним хранилищем и проводим тестовые восстановления по графику
- Сопровождаем: ежемесячные обновления по Upgrade Path, мониторинг, разбор инцидентов
- Обучаем команду клиента: код-ревью, ветвление, работа с CI/CD и registry

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 Install GitLab: Linux package (Omnibus) (docs.gitlab.com — 19.3)

02 Reference architecture: up to 20 RPS / 1,000 users (docs.gitlab.com — 19.3)

03 Back up and restore GitLab (docs.gitlab.com — 19.3)

04 Upgrade paths and GitLab 19 upgrade notes (docs.gitlab.com — 19.x)

05 Integrate LDAP with GitLab (docs.gitlab.com — 19.3)

06 GitLab Runner: register and configure runners (docs.gitlab.com — 19.3)

07 GitLab Container Registry administration (docs.gitlab.com — 19.3)

08 Чек-лист внедрения git-сервера ITfresh (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.