Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Immutable-инфраструктура: серверы не чинят, а пересоздают

Конвейер golden-образов Packer и rolling-замена узлов через Terraform — в облаке и на Proxmox

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Серверы заказчиков годами живут «снежинками»: ручные правки по SSH, дрейф конфигураций, эталона нет. Каждое обновление — рулетка, откат — 30–60 минут, восстановление после сбоя или шифровальщика — дни. Мы переводим stateless-слой на immutable-модель: сервер собирается из версионного golden-образа, при обновлении пересоздаётся, откат — предыдущий образ за 2–3 минуты.

Почему это важно бизнесу

- Откат неудачного релиза вручную — 30–60 минут простоя; переключение на предыдущий образ — 2–3 минуты
- Сервер с историей ручных правок невозможно быстро воспроизвести после сбоя или шифровальщика — эталона нет
- Дрейф конфигураций = регулярные инциденты «после обновления всё сломалось» и часы инженеров на разбор
- Ноль SSH в production закрывает целый класс атак и человеческих ошибок, каждый доступ — аудируемый
- Патч критической уязвимости раскатывается на весь парк за 10–15 минут, а не за вечер ручной работы

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

1.15.x

Packer: один HCL-шаблон даёт идентичные образы для AWS (amazon-ebs) и Proxmox (proxmox-iso)

developer.hashicorp.com, Packer 1.15

66%

min_healthy_percentage в instance_refresh: минимум 2 из 3 узлов живы во время rolling-замены

registry.terraform.io, AWS provider 6.x

120 с

instance_warmup + health_check_grace_period: новый узел обязан пройти ELB health check до гаше...

наш стандарт

1.13.x

Talos Linux для K8s-нод: без shell и SSH, управление через API, A/B-апгрейд с автооткатом

talos.dev, Talos 1.13

0

SSH-сессий в production: доступ через SSM Session Manager или debug-клон из того же образа

наш стандарт

5 мин

полный цикл деплоя: сборка образа + rolling-замена; откат = предыдущий тег образа за 2–3 минуты

наш стандарт

РЕАЛИЗАЦИЯ 1 ИЗ 3

Конвейер golden-образов: Packer в CI

Что настраиваем

Сборочный этап: единый репозиторий образов app-tier заказчика — AWS и Proxmox VE из одного кода

Как мы это делаем

1. Шаблон app-server.pkr.hcl: source_ami_filter по debian-12-amd64-* с явным owner-ID; для Proxmox — builder proxmox-iso с ISO Debian 12 и cloud-init драйвом
2. Provisioner shell: apt upgrade, nginx, приложение из артефакта CI; в финале удаляем authorized_keys Packer и machine-id — образ обезличен
3. Версионирование: имя shop-app-{{timestamp}} + git SHA в тегах; manifest post-processor отдаёт ID образа следующему шагу пайплайна — Terraform
4. Сборка в GitHub Actions/GitLab CI по тегу релиза: 4–6 минут на образ; retention-скрипт хранит последние 5 версий + релизные, остальное чистит

РЕЗУЛЬТАТ

Любой узел воспроизводим: один и тот же образ в тесте и в проде, идентичность гарантирована кодом, а не памятью инженера. Инциденты «на этом сервере почему-то иначе» исчезают как класс — расследуем не сервер, а конкретную версию образа.

КЛЮЧЕВОЙ НЮАНС

По докам Packer пиним required_plugins (amazon >= 1.3) и owners в source_ami_filter: без пина сборка в CI однажды притянет другой базовый образ, и «идентичные» образы перестанут быть идентичными.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Rolling-замена узлов: Terraform + ASG instance refresh

Что настраиваем

Прод-контур app-tier: aws_launch_template + autoscaling group, 3–10 инстансов за ALB

Как мы это делаем

1. aws_launch_template без key_name (SSH-ключа нет), image_id = вывод Packer; секреты приложение тянет на старте из Secrets Manager/Vault, не из образа
2. instance_refresh: strategy=Rolling, в preferences min_healthy_percentage=66 и instance_warmup=120; health_check_type=ELB — узел «жив», только если отвечает target g...
3. terraform apply -var="app_ami_id=..." — ASG сам поднимает узлы из нового образа, ждёт health check и гасит старые, деплой без даунтайма
4. Для рискованных релизов — blue-green: две ASG, переключение min_size переменной blue_active; откат = один apply, 2–3 минуты

РЕЗУЛЬТАТ

Деплой новой версии — около 5 минут без даунтайма и без входа на серверы. Откат — apply с предыдущим ID образа. Инциденты «сломали при обновлении» уходят в ноль: непроверенный узел просто не получает трафик от балансировщика.

КЛЮЧЕВОЙ НЮАНС

min_healthy_percentage по умолчанию 90 — на трёх узлах rolling так не пойдёт: считаем порог от фактического min_size (2/3 = 66%). И всегда health_check_type=ELB, иначе ASG признает живым мёртвое приложение.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Immutable без облака: Proxmox VE и Talos Linux

Что настраиваем

On-prem: гипервизоры Proxmox VE 8.x заказчика + Kubernetes-ноды на Talos

Как мы это делаем

1. Packer proxmox-iso собирает VM-шаблон с qemu-guest-agent и cloud_init=true; Terraform-провайдер bpg/proxmox клонирует боевые узлы из шаблона
2. Идентичность узла (hostname, IP, ключи) задаём только через cloud-init user-data при клонировании — сам шаблон руками не правим никогда
3. K8s-ноды — Talos 1.13: без shell и SSH, конфиг декларативно через talosctl apply-config, обновление talosctl upgrade — атомарный A/B с автооткатом
4. Логи и метрики уезжают с узла с первой секунды: Grafana Alloy → Loki, node_exporter → Prometheus — диагностика без захода на сервер

РЕЗУЛЬТАТ

Immutable работает и без облака: узлы клонируются из шаблона за минуты, K8s-ноды обновляются атомарно с автооткатом при неудаче. Восстановление стенда после аварии — пересоздание из кода и образов, а не реанимация уникальной машины по памяти.

КЛЮЧЕВОЙ НЮАНС

cloud_init=true в proxmox-iso лишь добавляет пустой cloud-init CDROM к шаблону — сетевые параметры задаются при клонировании через Terraform, а не запекаются в образ, иначе все клоны поднимутся с одним IP.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

✕ **Секреты запечены в образ**
Пароли и ключи в AMI утекают с каждым снапшотом. Передаём их на старте узла: cloud-init user-data + Vault/Secrets Manager, в образе — только код.

✕ **Immutable для баз данных**
Stateful-сервисы при пересоздании теряют данные. БД и очереди выносим на managed-сервисы или отдельные persistent-узлы с собственным бэкапом.

✕ **«Быстро поправлю по SSH»**
Одна ручная правка — и узел снова снежинка. Правило: любой фикс только через новый образ; для срочной отладки — SSM или debug-клон из того же образа.

✕ **Незапинованный базовый образ**
most_recent=true без фильтра owners однажды притянет чужой или сломанный образ. Фиксируем owner-ID и маску имени, обновление базы — коммит в шаблон.

✕ **Health check уровня EC2**
Проверка «инстанс запущен» пропускает мёртвое приложение. Ставим health_check_type=ELB и endpoint /healthz с проверкой зависимостей (БД, кэш).

✕ **Логи умирают вместе с узлом**
После пересоздания разбирать нечего. Логи и метрики шлём наружу с первой секунды жизни узла (CloudWatch/Loki, Prometheus), иначе immutable ослепляет.

✕ **Толстый user-data**
Сотни строк логики на старте = медленный и хрупкий запуск. Всё тяжёлое — в образ на этапе Packer, в user-data только параметры окружения.

✕ **Нет retention образов**
Сотни AMI и снапшотов копят счёт и мусор. Храним последние 5 версий плюс релизные, остальное чистим по расписанию скриптом в CI.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Packer-шаблон базового образа: ОС + патчи + агент мониторинга, сборка вручную
- Версионирование образов: имя с timestamp + git SHA, храним последние 5
- Логи и метрики наружу (Loki/CloudWatch + Prometheus) до отключения SSH

НОРМАЛЬНО

- Сборка образа в CI по тегу релиза, деплой через ASG instance refresh
- min_healthy_percentage=66, health_check_type=ELB, instance_warmup 120 с
- Секреты вне образа: Vault/Secrets Manager, доступ — SSM вместо SSH
- Terraform-state в удалённом backend с блокировками

ХОРОШО

- Blue-green: две среды, переключение и откат одним terraform apply
- Патчи безопасности конвейером: от коммита до прода 10–15 минут
- K8s-ноды на Talos: без shell, атомарный A/B-апгрейд с автооткатом
- Smoke-тесты образа в пайплайне до выпуска в прод (goss/Terratest)

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Можем ли пересоздать любой app-сервер из образа за 5 минут без ручных шагов?
- [ ] Остался ли в production SSH-доступ и кто им пользовался за последний месяц?
- [ ] Секреты передаются на старте узла, а не лежат внутри образа и его снапшотов?
- [ ] Откат на предыдущую версию — одна команда и меньше 5 минут?
- [ ] Health check проверяет приложение (endpoint /healthz), а не только факт запуска VM?
- [ ] Логи и метрики переживают уничтожение узла — уходят во внешнее хранилище?
- [ ] Stateful-компоненты (БД, очереди) отделены от immutable-слоя и бэкапятся отдельно?
- [ ] Базовый образ пересобирается с патчами безопасности не реже раза в месяц?
- [ ] Для каждого боевого узла известны версия и git SHA образа, из которого он поднят?
- [ ] Старые образы и снапшоты чистятся по retention-политике, а не копятся годами?

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 инфраструктуры и план перевода stateless-слоя на immutable: что пересоздаём, что оставляем persistent
- Конвейер под ключ: Packer-шаблоны, CI-сборка образов, Terraform с rolling- и blue-green-деплоем
- On-prem вариант: шаблоны Proxmox VE + cloud-init, кластер Kubernetes на Talos без SSH
- Наблюдаемость до отключения SSH: Loki/Prometheus/CloudWatch и доступ через SSM Session Manager
- Сопровождение: ежемесячная пересборка образов с патчами, retention и контроль дрейфа

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 Packer Docs: HCL2 templates, builder amazon-ebs (developer.hashicorp.com — 1.15)

02 Packer Proxmox plugin: proxmox-iso builder (developer.hashicorp.com — 1.2.x)

03 Terraform AWS provider: aws_autoscaling_group / instance_refresh (registry.terraform.io — 6.x)

04 Amazon EC2 Auto Scaling User Guide: Instance refresh (docs.aws.amazon.com — 2026)

05 Talos Linux Docs: Upgrades, talosctl (talos.dev — 1.13)

06 Proxmox VE Admin Guide: Cloud-Init Support (pve.proxmox.com — 8.x)

07 Наш шаблон: itf-golden-image pipeline (Packer + CI + retention) (наш стандарт — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.