Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

HAProxy vs Envoy: какой прокси выбрать для микросервисов

Как мы ставим HAProxy 3.2 LTS на edge и Envoy 1.39 внутри: критерии, параметры, миграция

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

У заказчика 10–40 микросервисов за одним nginx с ручными upstream: деплой требует reload, упавший бэкенд ловят клиенты, нет ретраев и трассировки. Мы ставим управляемый L7-слой: HAProxy на периметре, Envoy внутри. Без него каждый выкат — микропростой, а деградация одного сервиса каскадом валит смежные.

Почему это важно бизнесу

- Каждый reload прокси со сбросом соединений — обрывы платежей и загрузок файлов у клиентов
- Без health-check и ретраев отказ одной ноды виден пользователям, а не только мониторингу
- Без circuit breaker деградация одного сервиса каскадом кладёт весь контур за минуты
- Без метрик per-route разбор инцидента занимает часы вместо минут — SLA под угрозой
- Неверный выбор прокси = переделка контура через год и двойные затраты на внедрение

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

3.2 LTS

Ветка HAProxy, которую мы ставим на edge: фиксы до 2030-Q2, QUIC/HTTP-3, OpenSSL 3.5

по докам haproxy.org

1.39

Ветка Envoy: релизы ежеквартально, поддержка ~12 мес — закладываем апгрейд раз в квартал

по докам envoyproxy.io

2s/3/2

Наш health-check: inter 2s, fall 3, rise 2 — вывод ноды за ~6 с без ложных срабатываний

наш стандарт

50 000

global maxconn HAProxy на edge-паре; согласуем с fd-лимитом и памятью (~50 КБ/соединение TLS)

haproxy.cfg, наш стандарт

1024

Дефолт circuit_breakers.max_requests в Envoy — всегда пересчитываем под реальный RPS кластера

по докам envoyproxy.io

10%

max_ejection_percent в outlier detection: не выкидываем больше 1/10 эндпоинтов разом

по докам envoyproxy.io

РЕАЛИЗАЦИЯ 1 ИЗ 3

Edge-контур на HAProxy 3.2: пара с VRRP и seamless reload

Что настраиваем

Периметр: 2 VM (4 vCPU/8 ГБ) с keepalived, за ними 5–15 бэкенд-групп HTTP/gRPC

Как мы это делаем

1. Ставим haproxy 3.2 LTS из официального репо, режим master-worker в global, stats socket с level admin и expose-fd listeners — reload без обрыва соединений
2. defaults: timeout connect 5s / client 30s / server 30s / http-request 10s; frontend :443 с ssl-min-ver TLSv1.2, маршрутизация ACL path_beg по сервисам
3. backend: balance leastconn, option httpchk GET /health, http-check expect status 200, server ... check inter 2s fall 3 rise 2
4. Динамика: секция resolvers + server-template для DNS/Consul discovery, Data Plane API v3 для правок backend через REST без reload
5. Метрики: встроенный prometheus-exporter (http-request use-service prometheus-exporter на отдельном listener :8405) → Prometheus/Grafana

РЕЗУЛЬТАТ

Выкаты и смена пула бэкендов идут без разрыва клиентских соединений; отказ ноды скрыт от клиентов за ~6 с; edge-пара держит десятки тысяч RPS на 4 vCPU с запасом по CPU x3.

КЛЮЧЕВОЙ НЮАНС

Reload безопасен только в master-worker с expose-fd listeners на stats socket — без этого старый процесс рвёт установленные соединения. Проверяем ещё и согласованность maxconn с ulimit -n.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Межсервисный слой на Envoy: ретраи, circuit breaking, трейсинг

Что настраиваем

Внутренний контур: Envoy как front-proxy/sidecar перед 10–40 сервисами, gRPC и HTTP/2

Как мы это делаем

1. Bootstrap envoy.yaml: admin на 127.0.0.1:9901 (не наружу!), dynamic_resources с ADS по gRPC — конфиг едет через xDS v3 (LDS/RDS/CDS/EDS) без рестартов
2. Кластеры: type EDS (или STRICT_DNS без control plane), health_checks interval 5s / timeout 2s, outlier_detection: consecutive_5xx 5, base_ejection_time 30s
3. route retry_policy: retry_on 5xx,connect-failure, num_retries 2, per_try_timeout 2s + retry_budget в circuit_breakers (budget_percent 20) вместо жёсткого max_retrie…
4. circuit_breakers per-cluster: max_connections/max_requests пересчитываем от нагрузочного прогона, а не оставляем дефолт 1024
5. Наблюдаемость: /stats/prometheus с admin-порта, access log в JSON, трейсинг OpenTelemetry-фильтром с пробросом traceparent

РЕЗУЛЬТАТ

Сбойный эндпоинт исключается outlier detection автоматически, ретраи гасят единичные 5xx, а latency per-route и retry count видны в Grafana — инцидент локализуется по метрикам за минуты без захода на хосты.

КЛЮЧЕВОЙ НЮАНС

Ретраи без per_try_timeout и retry budget превращают деградацию в retry storm: нагрузка умножается на num_retries. Бюджет ретраев считаем от ёмкости бэкенда, лимиты — из нагрузочного теста.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Выбор и миграция: матрица критериев и канареечный перевод трафика

Что настраиваем

Этап пилота: стейдж-стенд, затем боевой периметр заказчика, перевод без окна простоя

Как мы это делаем

1. Инвентаризация: протоколы (gRPC → Envoy нативно), число сервисов (до ~5 — хватает одного HAProxy), нужен ли service mesh/динамический discovery
2. PoC на стейдже: одинаковые маршруты на обоих прокси, прогон wrk/vegeta с рабочим профилем нагрузки, сравнение p99 и RAM/CPU под пиком
3. Канарейка: на старом балансировщике заводим weight 10% на новый контур, растим до 100% по мере чистых метрик 5xx/p99 за 24 ч
4. Фиксируем эксплуатационный регламент: кто и как меняет конфиг (git + CI-валидация haproxy -c / envoy --mode validate), алерты на health-check flap

РЕЗУЛЬТАТ

Заказчик получает не «прокси по моде», а контур под свой профиль: чаще всего HAProxy на edge + Envoy внутри. Перевод трафика обратим на любом шаге — откат за минуты сменой weight.

КЛЮЧЕВОЙ НЮАНС

Валидация конфига до выката обязательна: haproxy -c -f и envoy --mode validate ловят 90% ошибок ещё в CI. Конфиг прокси держим только в git, ручные правки на хосте запрещаем.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Reload рвёт соединения**
  systemctl reload без master-worker и expose-fd listeners сбрасывает клиентов; включаем master-worker + stats socket и проверяем на longpoll-сессиях
- ✕ **DNS резолвится один раз**
  HAProxy без секции resolvers берёт IP на старте; при смене адреса сервиса трафик идёт в пустоту — ставим resolvers + server-template
- ✕ **Дефолтные circuit breakers**
  1024 max_requests у Envoy душит нагруженный кластер или, наоборот, велик для слабого — пересчитываем per-cluster от нагрузочного теста
- ✕ **Retry storm при деградации**
  retry_on 5xx без per_try_timeout и retry_budget умножает нагрузку на падающий сервис; задаём бюджет ретраев и таймаут попытки
- ✕ **Открытый admin-порт Envoy**
  На :9901 есть /quitquitquit и сброс счётчиков; вешаем только на 127.0.0.1, метрики наружу отдаём отдельным listener
- ✕ **Нет timeout http-request**
  Без него HAProxy уязвим к slowloris — медленные заголовки съедают maxconn; ставим timeout http-request 10s в defaults
- ✕ **maxconn выше ulimit**
  global maxconn 50000 при ulimit -n 4096 = отказ под пиком; проверяем LimitNOFILE в unit-файле и память на соединение
- ✕ **Envoy-конфиг пишут руками**
  Статический YAML на 40 сервисов нечитаем и расходится с реальностью; либо генерация из шаблона в CI, либо control plane с xDS

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- HAProxy 3.2 LTS на edge: ACL-маршрутизация, httpchk, таймауты в defaults
- master-worker + stats socket expose-fd — reload без обрыва соединений
- Валидация haproxy -c -f в CI, конфиг в git, запрет ручных правок
- Prometheus-exporter и алерты на рост 5xx и health-check flap

НОРМАЛЬНО

- Пара edge-нод с keepalived (VRRP), синхронизация stick-table между ними
- Секция resolvers + server-template: discovery через DNS/Consul без reload
- Envoy 1.39 на межсервисный трафик: ретраи, outlier detection, gRPC-баланс
- Rate limiting на фронте: stick-table http_req_rate + deny 429 по порогу

ХОРОШО

- Control plane (xDS/ADS): конфиг Envoy обновляется без рестартов из реестра
- Circuit breakers и retry_budget посчитаны от нагрузочного прогона, не дефолт
- Сквозной трейсинг OpenTelemetry: traceparent от edge до последнего сервиса
- Канареечные выкаты весами (weight) и автоматический откат по метрикам p99/5xx

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Reload прокси проходит без разрыва установленных соединений (проверено на живой сессии)?
- [ ] Health-check настроен на каждый бэкенд и выводит ноду быстрее, чем это заметит клиент?
- [ ] Таймауты connect/client/server/http-request заданы явно, а не оставлены по умолчанию?
- [ ] Конфиг прокси хранится в git и валидируется (haproxy -c / envoy --mode validate) до выката?
- [ ] Admin-порт Envoy и stats socket HAProxy недоступны из внешней сети?
- [ ] Ретраи ограничены per_try_timeout и бюджетом — деградация не превратится в retry storm?
- [ ] Метрики per-route (p99, 5xx, retry count) собираются в Prometheus и есть алерты?
- [ ] maxconn согласован с ulimit -n, памятью и лимитами circuit breaker по всей цепочке?
- [ ] Смена IP/состава бэкендов подхватывается без reload (resolvers или xDS)?
- [ ] Есть план отката: перевод трафика весами обратим на любом шаге миграции?

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 текущей балансировки и матрица выбора HAProxy/Envoy под ваш профиль нагрузки
- Внедрение edge-пары HAProxy 3.2 LTS: VRRP, TLS, health-check, seamless reload, метрики
- Настройка Envoy: ретраи, circuit breaking, outlier detection, трейсинг OpenTelemetry
- Канареечная миграция со старого nginx/прокси без окна простоя, с планом отката
- Поддержка контура: обновления веток LTS, нагрузочные прогоны, дашборды и алерты

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 HAProxy Configuration Manual (ветка 3.2) (haproxy.org — 3.2)

02 HAProxy Management Guide: master-worker, seamless reload (haproxy.org — 3.2)

03 HAProxy Data Plane API Reference (haproxy.com — v3)

04 Envoy xDS protocol (LDS/RDS/CDS/EDS, ADS) (envoyproxy.io — 1.39)

05 Envoy Circuit breaking & Outlier detection (envoyproxy.io — 1.39)

06 Envoy Admin interface & statistics (envoyproxy.io — 1.39)

07 Шаблон ITfresh: edge-пара haproxy.cfg + envoy bootstrap (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.