Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

# eBPF-observability для Kubernetes: Cilium, Hubble, Tetragon

Как мы строим сетевую видимость и runtime-контроль кластера без sidecar-прокси

Июль 2026

**itfresh.ru · ИТ-аутсорсинг для юридических лиц**

О ЧЁМ ЭТОТ РАЗБОР

# Суть проблемы

Кластер Kubernetes работает, но что происходит в сети между подами — не видно: почему запрос из одного namespace не доходит до другого и какой процесс полез в чужой каталог. Типовой ответ — sidecar-прокси на каждый под: плюс CPU, память и точки отказа. Мы закрываем видимость в ядре: Cilium как CNI, Hubble как поток сетевых событий, Tetragon как runtime-контроль процессов.

## Почему это важно бизнесу

- Диагностика сетевого сбоя в кластере без видимости растягивается на часы, и всё это время сервис недоступен клиентам
- Sidecar-прокси на каждом поде съедают CPU и память, за которые компания платит в счёте за инфраструктуру
- Без runtime-контроля запуск постороннего процесса внутри контейнера остаётся незамеченным до последствий
- NetworkPolicy без наблюдаемости пишут «на глаз»: либо разрешено всё, либо ломается продуктивная нагрузка
- Нет журналов сетевых потоков — нечем разобрать инцидент задним числом и нечего предъявить аудиту

ЧТО МЫ ЗАКЛАДЫВАЕМ

# Ключевые параметры реализации

**1.20.x**

стабильная ветка Cilium, которую разворачиваем; параллельно поддерживаются 1.19 и 1.18

*docs.cilium.io, Cilium 1.20*

**5.10+**

минимальное ядро Linux для текущих релизов Cilium; на RHEL 8.10 достаточно 4.18

*System Requirements, Cilium 1.20*

**v1.7.0**

версия Helm-чарта и образа Tetragon, которую фиксируем в values перед раскаткой

*Tetragon 1.7.0, Helm chart reference*

**4095**

hubble.eventBufferCapacity по умолчанию: буфер flow на агент, 2^n-1, максимум 65535

*Helm reference cilium/cilium*

**16384**

bpf.policyMapMax по умолчанию — потолок записей policy-map на endpoint

*Helm reference cilium/cilium*

**9965**

порт метрик Hubble; агент 9962, оператор 9963, Envoy 9964 — все четыре в Prometheus

*Monitoring & Metrics, Cilium 1.20*

РЕАЛИЗАЦИЯ 1 ИЗ 3

# Cilium CNI как фундамент стека: установка и приёмка

## Что настраиваем

*Все ноды кластера: DaemonSet cilium-agent и Deployment cilium-operator в namespace kube-system*

## Как мы это делаем

1. Сверяем ядро и BTF: 5.10+ (RHEL 8.10 — 4.18), CONFIG_BPF_SYSCALL=y, CONFIG_DEBUG_INFO_BTF=y, наличие /sys/kernel/btf/vmlinux
2. Ставим чарт с жёстко зафиксированной версией: helm install cilium cilium/cilium --version 1.20.1 -n kube-system -f values.yaml
3. Открываем между нодами 4240/TCP (health), 4244/TCP (Hubble), 4245/TCP (Relay), 8472/UDP (VXLAN), 51871/UDP (WireGuard), ICMP echo
4. При отказе от kube-proxy задаём kubeProxyReplacement=true вместе с k8sServiceHost и k8sServicePort — иначе агент не найдёт API-сервер
5. Приёмка до перевода нагрузки: cilium status --wait, затем cilium connectivity test в отдельном тестовом namespace

**РЕЗУЛЬТАТ**

Кластер получает единый datapath на eBPF: политики, балансировка сервисов и наблюдаемость живут в одном компоненте, без параллельного набора агентов. Connectivity test даёт документированное подтверждение, что pod-to-pod, сервисы и DNS работают ещё до перевода продуктивной нагрузки.

**КЛЮЧЕВОЙ НЮАНС**

Требования к ядру у дистрибутивов разные: 5.10+ у большинства, но на RHEL 8.10 хватает 4.18. Сверяем полный список CONFIG_* из раздела System Requirements, а не только вывод uname -r.

РЕАЛИЗАЦИЯ 2 ИЗ 3

# Hubble: поток L3/L4/L7 и метрики в Prometheus и Grafana

## Что настраиваем

*Hubble server внутри cilium-agent, плюс hubble-relay и hubble-ui; метрики на 9965/TCP, ServiceMonitor в monitoring*

## Как мы это делаем

1. Включаем hubble.enabled, hubble.relay.enabled, hubble.ui.enabled; 4244/TCP обязателен на каждой ноде, иначе Relay не соберёт поток
2. Задаём hubble.metrics.enabled="{dns,drop,tcp,flow,icmp,httpV2}" с sourceContext и destinationContext — контекст напрямую задаёт кардинальность
3. Скрейп: hubble.metrics.serviceMonitor.enabled=true (interval 10s), prometheus.serviceMonitor.enabled=true, дашборды — dashboards.enabled=true
4. На нагруженных нодах поднимаем hubble.eventBufferCapacity с 4095 до 16383 или 65535 — допустимы только значения вида 2^n-1
5. Разбор инцидента: cilium hubble port-forward, затем hubble observe --namespace <ns> --verdict DROPPED -f до конкретной политики

**РЕЗУЛЬТАТ**

Инженер видит поток L3/L4/L7 в реальном времени и отвечает на вопрос «почему не ходит трафик» за минуты: hubble observe показывает verdict DROPPED и политику-виновника. Метрики drop, dns и http ложатся в Grafana как штатные графики кластера.

**КЛЮЧЕВОЙ НЮАНС**

Hubble по умолчанию держит события в кольцевом буфере на ноде и ничего не пишет на диск. Нужна ретроспектива — включаем hubble.export.static с fieldMask и вывозим файл наружу, иначе история теряется.

РЕАЛИЗАЦИЯ 3 ИЗ 3

# Tetragon: runtime-контроль процессов, файлов и эскалаций

## Что настраиваем

*DaemonSet tetragon и tetragon-operator в kube-system; TracingPolicy на кластер, TracingPolicyNamespaced — на namespace*

## Как мы это делаем

1. helm repo add cilium https://helm.cilium.io/ и helm install tetragon cilium/tetragon -n kube-system с фиксацией версии чарта 1.7.0
2. Пишем TracingPolicy на kprobe security_file_open с matchArgs operator Prefix по /etc/shadow, /etc/passwd и каталогам с секретами
3. Шум режем в ядре: matchBinaries, matchNamespaces, matchWorkloads и действие Post с rateLimit и rateLimitScope, а не фильтрами в userspace
4. Экспорт: exportFilename, exportFileMaxSizeMB=10 и exportFileMaxBackups=5 (дефолты чарта), exportRateLimit; JSON забирает агент логов ноды
5. Проверка: kubectl exec -n kube-system ds/tetragon -c tetragon -- tetra getevents -o compact; метрики Tetragon на порту 2112

**РЕЗУЛЬТАТ**

Служба эксплуатации получает поток событий exec, доступа к файлам и смены привилегий с привязкой к поду и namespace. Посторонний запуск в контейнере виден в момент события, а не при разборе последствий; политику можно перевести из наблюдения в блокировку.

**КЛЮЧЕВОЙ НЮАНС**

Enforcement (Sigkill, Override) включаем только после периода наблюдения: широко написанная политика убьёт легитимный процесс. Сначала Post с rateLimit, затем сужение селекторами, и лишь потом блокировка.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

# Подводные камни

- ✕ **Установка без проверки ядра и BTF**
  Без CONFIG_DEBUG_INFO_BTF=y часть eBPF-программ и Tetragon не загрузятся. Проверяем /sys/kernel/btf/vmlinux и список CONFIG_* до раскатки чарта

- ✕ **Закрытый 4244/TCP — Hubble «пустой»**
  Relay собирает поток с агентов по 4244/TCP. При закрытом порту UI показывает ноль флоу, а причину ищут в конфигурации Hubble, а не в фаерволе нод

- ✕ **Метрики Hubble с контекстом pod**
  sourceContext и destinationContext со значением pod плодят серию на каждый под: Prometheus раздувается, дашборды тормозят. Начинаем с namespace

- ✕ **Дефолтные BPF-map под высокой нагрузкой**
  Дефолты policyMapMax 16384 и ctTcpMax 524288 не тянут высокий pps. Поднимаем bpf.policyMapMax и bpf.ctTcpMax либо включаем bpf.mapDynamicSizeRatio

- ✕ **natMax больше 2/3 суммы CT-таблиц**
  Агент валидирует: bpf-nat-global-max не больше 2/3 суммы bpf-ct-global-tcp-max и bpf-ct-global-any-max. Иначе cilium-agent падает на старте

- ✕ **TracingPolicy без селекторов**
  kprobe без matchArgs и matchBinaries отдаёт в userspace каждый вызов: растут CPU и объём лога. Фильтруем в ядре селекторами и ограничиваем rateLimit

- ✕ **Миграция с iptables-CNI «на живую»**
  Cilium считает политики в eBPF, поведение NetworkPolicy отличается от iptables. Переносим через новый node pool с постепенным drain старых нод

- ✕ **helm install без --version**
  Без явной версии следующий апгрейд подтянет новый мажор Cilium со сменой дефолтов. Версии чарта и образов фиксируем в values.yaml в Git

УРОВНИ РЕАЛИЗАЦИИ

# Как правильно

## МИНИМУМ

- Ядро 5.10+ с BTF на всех нодах, версия чарта Cilium зафиксирована в Git
- cilium status --wait и cilium connectivity test как обязательная приёмка
- Hubble включён, порт 4244/TCP открыт между всеми нодами кластера
- prometheus.enabled=true: метрики агента (9962) и Hubble (9965) в Prometheus

## НОРМАЛЬНО

- ServiceMonitor для agent, operator и Hubble; дашборды через dashboards.enabled
- Контекст метрик Hubble — namespace, алерты по hubble_drop_total
- Tetragon с TracingPolicy на доступ к /etc/shadow и каталогам секретов
- BPF-map отстроены под pps: policyMapMax, ctTcpMax, mapDynamicSizeRatio

## ХОРОШО

- Hubble exporter в файл с fieldMask и allowList, вывоз событий в SIEM
- hubble.redact для URL query и заголовков — секреты не попадают во flow
- TracingPolicyNamespaced на продуктивные namespace, Post с rateLimit
- Апгрейд Cilium сначала на стенде с connectivity test, затем на проде

ПРОВЕРЬТЕ У СЕБЯ

# Чек-лист самопроверки

- [ ] Ядро на всех нодах 5.10+ (или 4.18 на RHEL 8.10) и BTF доступен в /sys/kernel/btf/vmlinux?
- [ ] Версии чарта Cilium и Tetragon зафиксированы в values.yaml, а не берутся как latest?
- [ ] Порты 4240, 4244, 4245/TCP и 8472/UDP (6081/UDP при Geneve) открыты между всеми нодами?
- [ ] cilium connectivity test прогоняется после каждого апгрейда, а не только при установке?
- [ ] Метрики 9962, 9963, 9965 попадают в Prometheus и есть алерт на рост drop?
- [ ] hubble.eventBufferCapacity поднят на нагруженных нодах до значения вида 2^n-1?
- [ ] Размеры BPF-map (policyMapMax, ctTcpMax, natMax) пересчитаны под реальную нагрузку?
- [ ] Есть TracingPolicy на доступ к чувствительным файлам и на запуск процессов в контейнерах?
- [ ] События Tetragon ротируются по размеру и вывозятся за пределы ноды в хранилище логов?
- [ ] Чувствительные данные в сетевых потоках скрыты через hubble.redact?

*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 кластера и нод: ядро, BTF, открытые порты, совместимость с текущим CNI
- Развёртывание Cilium 1.20.x с Hubble Relay и UI, фиксация версий и values в Git
- Настройка Prometheus и Grafana: ServiceMonitor, дашборды, алерты по drop и BPF-map
- Внедрение Tetragon: TracingPolicy под ваш периметр, экспорт событий в SIEM
- Миграция с iptables-CNI через новый node pool с проверкой всех NetworkPolicy

**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 System Requirements: ядро, CONFIG_*, порты нод (docs.cilium.io — 1.20)

02 Monitoring & Metrics: порты 9962/9963/9964/9965, ServiceMonitor (docs.cilium.io — 1.20)

03 Setting up Hubble Observability и Configuring Hubble exporter (docs.cilium.io — 1.20)

04 Helm reference cilium/cilium: bpf.*, hubble.*, dashboards (docs.cilium.io — 1.20.1)

05 eBPF Maps: sizing и bpf-map-dynamic-size-ratio (docs.cilium.io — 1.20)

06 Tracing Policy, selectors и Helm chart values Tetragon (tetragon.io — 1.7.0)

07 Наш шаблон values.yaml и приёмочный чек-лист кластера (itfresh.ru — 2026)

*Основано на официальной документации продуктов и нашей практике внедрения.*