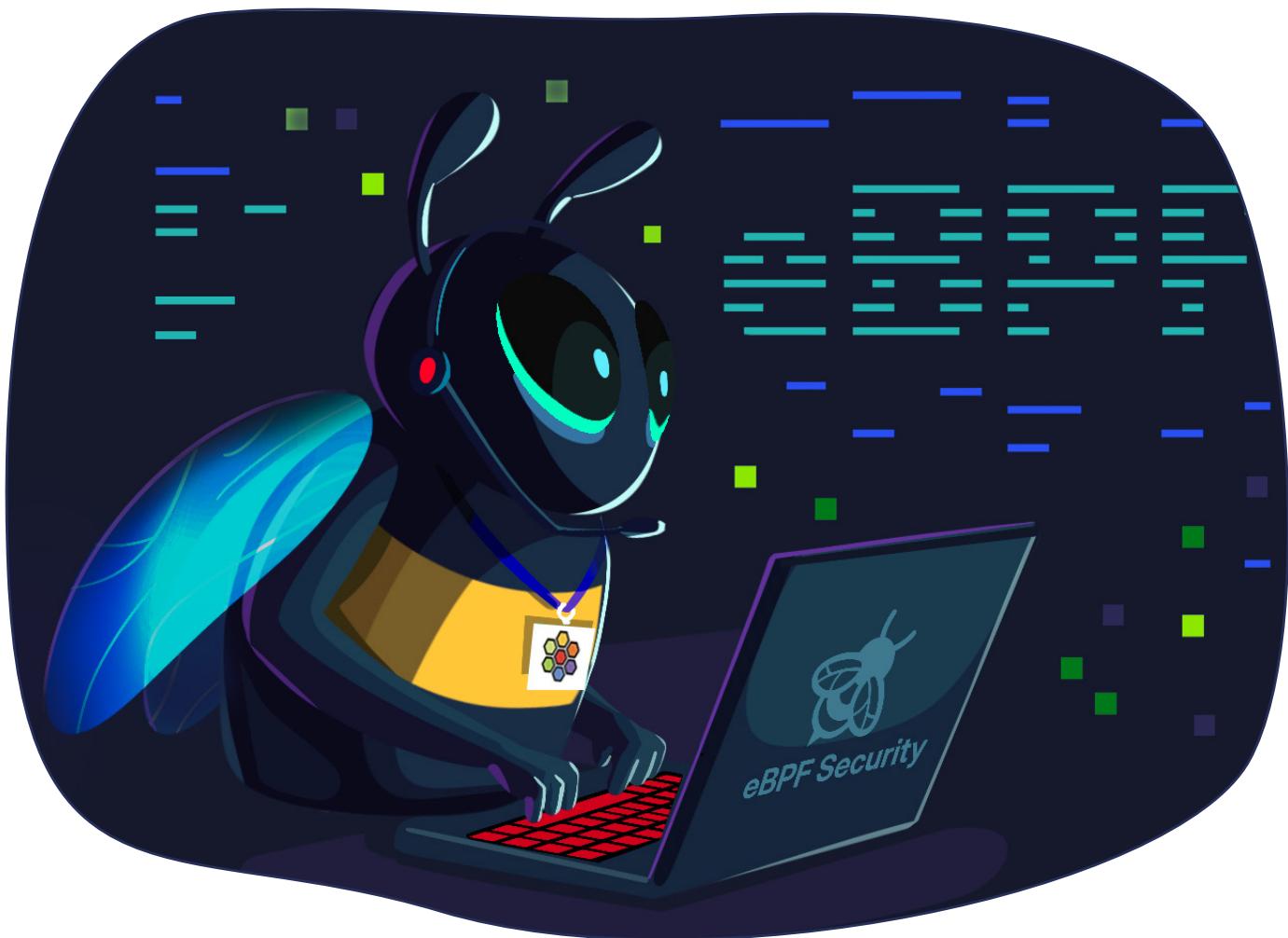


Правильная настройка сетевых политик Kubernetes



Авторы

Дин Льюис – старший инженер по техническому маркетингу в Isovalent с разнообразным опытом работы в области ИТ, архитектурного проектирования и облачных технологий.

Дин фокусируется на внедрении облачных инноваций, помогая компаниям использовать корпоративные решения **Isovalent**, включая Cilium, Hubble и Tetragon, для обеспечения безопасности и масштабирования своей IT-инфраструктуры. Больше информации можно узнать из его [персонального блога¹](#) или послушать его выступления на технологических форумах.

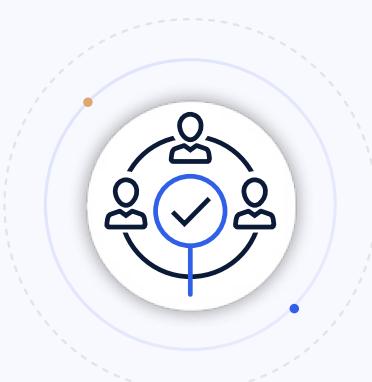
Рэймонд де Йонг является техническим директором Isovalent в регионе EMEA. Сотрудничает с заказчиками и сотрудниками, добиваясь успеха в критически важных средах, обеспечивая бесперебойное внедрение и высокую производительность Cilium.

Обладая почти десятилетним опытом, Рэймонд увлечен изучением уникальных вариантов использования ПО клиентами и разработкой инновационных решений, отвечающих их растущим потребностям. Помимо мира технологий, занимается теннисом, ездой на велосипеде, азартными играми и сочинением музыки на гитаре или фортепиано.

С особой благодарностью рецензентам этой книги:

Нико Виберт Даффи
Кули Роланд Уолтерс
Анна Капушинская

Филип Шмид
Энтони Берк
Джереми Колвин
Джо Стрингер



Содержание

Вступление	4
Каковы основные проблемы, с которыми сталкиваются предприятия, внедряющие сетевые политики?	5
Чему Вы научитесь?	6
Что такое сетевые политики?	7
Единая система безопасности и наблюдаемости для современных рабочих нагрузок при помощи Cilium	8
Безопасное межсервисное взаимодействие	9
- Безопасный доступ к внешним сервисам и из них	11
- Прозрачная защита API-интерфейсов	12
Подходы Kubernetes к внедрению сетевой политики	13
Стратегия дизайна Сетевых политик	18
Команды разработчиков и Мультитенантность	18
Ключевые угрозы сетевой безопасности Kubernetes	20
Оптимизация доступов	23
Анализ рисков	24
- Определение приоритетов	24
- Инвентаризация неиспользуемого доступа	24
- Последствия избыточности	26
Инициализация стратегии Сетевых политик	27
Приоритизация работы с пространствами имён	32
Глобальные сетевые политики кластера	33
Трудности реализации сетевых политик без необходимых инструментов	35
От теории к практике: реальный пример	36
Внедрение архитектуры Zero Trust	38
Hubble	39
- Hubble CLI	40
- Hubble UI	43
- Hubble Timescape	44
- Редактор сетевых политик	46
- Метрики Hubble	51
- Дашборды Isovalent Enterprise для Cilium и Hubble	53
Заключение	54
Приложение	55

Вступление

В современных условиях регулирование безопасности — не просто необходимость, а регулярная задача. Предприятия сталкиваются с необходимостью защищать критически важные приложения, не отставая при этом от быстрых циклов разработки. Ставки высоки: один неверный шаг может привести к утечке данных, нарушению соответствия требованиям или простою.

В Isovalent мы видели, как организации решают эту задачу, внедряя передовые сетевые политики, шифрование и взаимную аутентификацию. Но как достичь идеального баланса между надежной защитой и оперативностью? В этой электронной книге рассматриваются технические аспекты и реальные стратегии разработки и внедрения сетевых политик с использованием Cilium на базе eBPF в высокозащищенных средах.



Каковы основные проблемы, с которыми сталкиваются предприятия, внедряющие сетевые политики?

Создание критически важной безопасной платформы - задача не из легких. С чего начать? Как обеспечить надежную защиту и при этом не отставать от стремительных темпов разработки приложений? Рассмотрим эти насущные проблемы:

- ● **Баланс между безопасностью и гибкостью:** Как внедрять строгие меры безопасности, не препятствуя инновациям? Могут ли разработчики легко внедрять свои приложения, обеспечивая при этом безопасность?
- ● **Адаптация к постоянным изменениям:** Поскольку облачные приложения быстро развиваются, как успевать за новыми сервисами или функциями, которые не поддерживаются текущими сетевыми политиками? Как избежать "белых пятен" в политике, не останавливая разработку?
- ● **Управление сложностью в масштабе:** Как при наличии нескольких команд и служб обеспечить согласованное применение политик во всей инфраструктуре, не создавая узких мест в работе?
- ● **Подтверждение соответствия требованиям:** Являются ли меры безопасности достаточно комплексными, чтобы соответствовать нормативным требованиям? Можно ли гарантировать, что платформа соответствует отраслевым стандартам, таким как GDPR, PCI-DSS или HIPAA, и способны ли вы эффективно продемонстрировать соответствие требованиям?
- ● **Реагирование на угрозы безопасности:** Можете ли вы быстро выявлять инциденты безопасности и реагировать на них до того, как они приведут к несанкционированному доступу или утечке данных? Достаточно ли надежны ваши системы обнаружения и реагирования, чтобы предотвратить потенциальный ущерб?

Эти проблемы не возникают сами по себе — они накладываются друг на друга, создавая трения между командами и технологиями. Перед лицом такой сложности ваша стратегия должна обеспечивать безопасность и соответствие требованиям вашей платформы, позволяя разработчикам быстро продвигаться вперед и внедрять инновации.

Чему Вы научитесь?

В этом руководстве мы расскажем об основных стратегиях принятия и реализации эффективных сетевых политик в средах **Kubernetes**. Вы получите четкое представление о том, как сбалансировать безопасность и оперативную гибкость, используя передовые инструменты, такие как **Cilium** и **Hubble**, для обеспечения масштабируемых, детализированных политик.

Независимо от того, отвечаете ли вы за защиту критически важных приложений, управление кластерами **Kubernetes** или обеспечение соответствия требованиям, этот документ предоставит полезную информацию по следующим вопросам:

- ● Разработка сетевых политик, которые защищают вашу инфраструктуру, не препятствуя разработке приложений.
- ● Использование наблюдаемости сети для улучшения и оптимизации обеспечения безопасности.
- ● Внедрение архитектуры с нулевым уровнем доверия для минимизации рисков безопасности в вашем кластере.
- ● Используйте **Cilium** и **Hubble** для упрощения управления политиками, обеспечения соответствия требованиям и масштабной защиты рабочих нагрузок **Kubernetes**.

К концу обучения вы будете обладать знаниями и инструментами, позволяющими уверенно применять сетевые политики, соответствующие вашим целям в области безопасности и бизнеса, эффективно устранивая разрыв между стратегическим пониманием и технической реализацией.



Что такое сетевые политики?

В средах **Kubernetes** сетевые политики выступают в качестве критического уровня безопасности, контролируя потоки трафика между подами, сервисами и внешними объектами. На высоком уровне сетевые политики позволяют вам определять, что разрешено — или что заблокировано — при передаче данных на уровне IP или портов (уровни OSI 3 и 4).

Например, вы можете захотеть обеспечить, чтобы только определенные поды в вашем кластере могли взаимодействовать с определенными сервисами, или ограничить доступ к уязвимым областям вашей инфраструктуры из внешних источников. Сетевые политики дают вам возможность определять и применять эти правила, помогая обезопасить ваши приложения и защитить их от несанкционированного доступа.

Однако определение эффективных сетевых политик может быть сложной задачей. Ниже приведен ключевые моменты, участвующие в создании сетевых политик:

- ● **Взаимодействие подов:** Укажите, какие поды могут взаимодействовать с другими, гарантируя, что будет осуществляться только одобренное взаимодействие.
- ● **Управление неймспейсами:** Определите правила на уровне неймспейса, чтобы ограничить или разрешить трафик между различными частями вашего приложения или среды.
- ● **Правила блокировки IP-адресов:** Установите политики на основе диапазонов IP-адресов, определяя, каким внешним IP-адресам разрешен или заблокирован доступ к вашему кластеру.

Сетевые политики реализуются с помощью сетевого плагина, такого как **Cilium**, который обеспечивает улучшенную видимость и безопасность среды **Kubernetes**. В то время как стандартные сетевые политики **Kubernetes** охватывают базовые элементы управления уровня 3 и уровня 4, **Cilium** расширяет их, включая более продвинутые политики, позволяющие контролировать трафик уровня 7 и включая политики всего кластера, с помощью специализированного формата общих политик *CiliumClusterwideNetworkPolicy*. Такая степень детализации особенно важна в современных облачных средах, где безопасность должна масштабироваться наряду с быстро развивающимися приложениями.

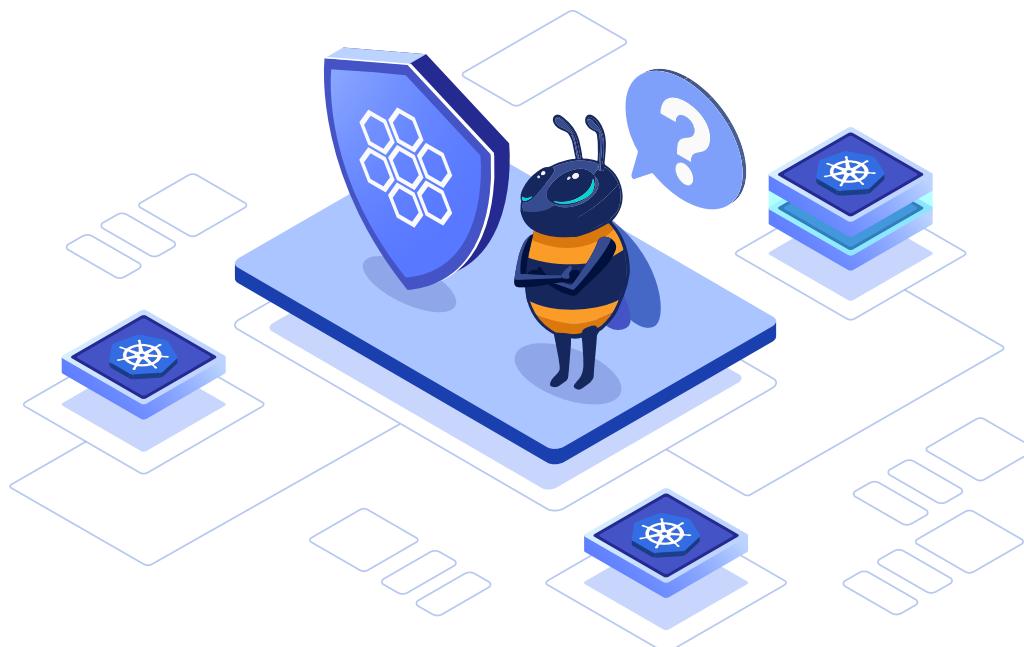
Используя **Cilium**, предприятия могут внедрять сетевые политики с уверенностью в том, что они не только применяются, но и масштабируемые, безопасны и готовы поддерживать даже самые требовательные облачные рабочие нагрузки.

Единая система безопасности и наблюдаемости для современных рабочих нагрузок при помощи Cilium

Cilium обеспечивает беспрецедентную наглядность и контроль над системами и приложениями с высоким уровнем детализации и эффективности. Это мощное сочетание работает прозрачно, не требуя изменений приложений. **eBPF** превосходно справляется с современными контейнеризированными рабочими нагрузками и без проблем поддерживает традиционные рабочие нагрузки, включая виртуальные машины и стандартные процессы **Linux**. С помощью **Cilium** и **eBPF** вы достигаете повышенной безопасности, производительности и гибкости всей вашей инфраструктуры, будь то облачная или легаси инфраструктура.

eBPF – это революционная технология, основанная на ядре Linux, которая позволяет запускать изолированные программы в привилегированном контексте, например, в ядре операционной системы. Она используется для безопасного и эффективного расширения возможностей ядра без изменения исходного кода ядра или загрузки модулей ядра.

Платформа **Isovalent**, включая **Cilium** и **Tetragon**, построена на основе технологии.



Безопасное межсервисное взаимодействие

Современные распределенные приложения используют контейнерные технологии для быстрого развертывания и масштабируемых операций. Однако традиционные подходы к защите этих рабочих нагрузок, такие как брандмауэры на базе IP, могут стать неэффективными и медленными. Некоторые клиенты сообщают, что время запуска контейнеров растягивается до нескольких минут, поскольку брандмауэры необходимо постоянно обновлять при каждом запуске новых контейнеров. Это создает проблемы как с масштабируемостью, так и с эксплуатационными характеристиками, особенно при одновременном запуске нескольких контейнеров.

Традиционные брандмауэры защищают рабочие нагрузки, фильтруя трафик на основе IP-адресов источника и назначения и портов. При каждом запуске контейнера необходимо обновлять правила брандмауэра в кластере, что может замедлить развертывание и усложнить операции по мере увеличения количества контейнеров.

Cilium решает эти проблемы, управляя безопасностью на уровне ендпоинтов. В терминологии Cilium ендпоинты – это группа контейнеров приложений, которые используют общий IP-адрес.

В Kubernetes метки – это пары ключ-значение, присваиваемые таким ресурсам, как поды и неймспейсы. Эти метки позволяют вам проще группировать ресурсы и управлять ими, особенно при определении правил безопасности. Cilium присваивает идентификатор безопасности группам подов (конечных точек), которые используют один и тот же набор меток, чтобы упростить применение политики и преодолеть ограничения, связанные с фильтрацией на основе IP.



Идентификатор безопасности прикрепляется ко всем сетевым пакетам, отправляемым подами, и проверяется на принимающем узле. Тем самым, Cilium обеспечивает масштабируемое и эффективное обеспечение безопасности без необходимости постоянного обновления брандмауэра. Управление этими идентификационными данными безопасности еще больше упрощается благодаря хранилищу ключей и значений, и делает его более гибким.

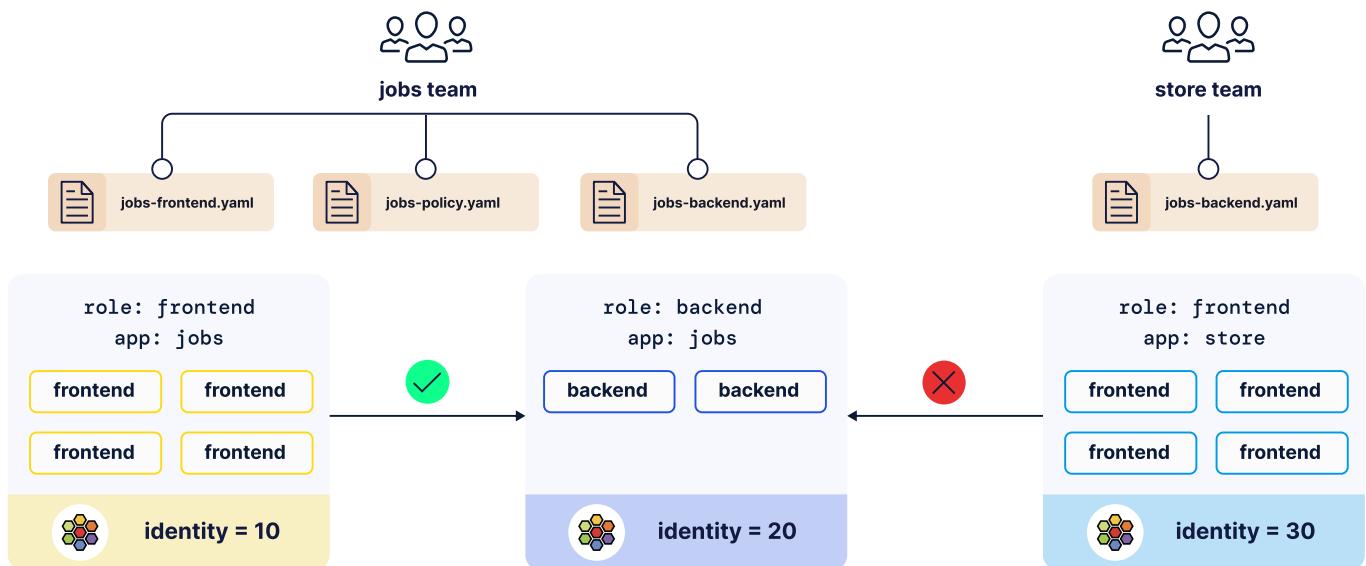


Рисунок 1. Идентификаторы Cilium с сетевыми политиками



Безопасный доступ к внешним сервисам и из них

Безопасность на основе меток является предпочтительным инструментом для управления внутренним контролем доступа к трафику, исходящему из кластеров. Кроме того, при рассмотрении вопроса о защите трафика, поступающего от внешних служб, **Cilium** поддерживает традиционные политики безопасности на основе **CIDR** как для входящего, так и для исходящего трафика, что позволяет ограничить доступ к определенным диапазонам IP-адресов.

Кроме того, **Cilium** повышает гибкость, поддерживая политики безопасности на основе **DNS**. Эти политики позволяют создавать правила входа и выхода на основе полных доменных имен (FQDN), включая использование подстановочных знаков и шаблонов соответствия. Политики, основанные на **DNS**, особенно полезны, когда удаленные IP-адреса являются динамическими или заранее неизвестными, или когда использование **DNS** упрощает управление.

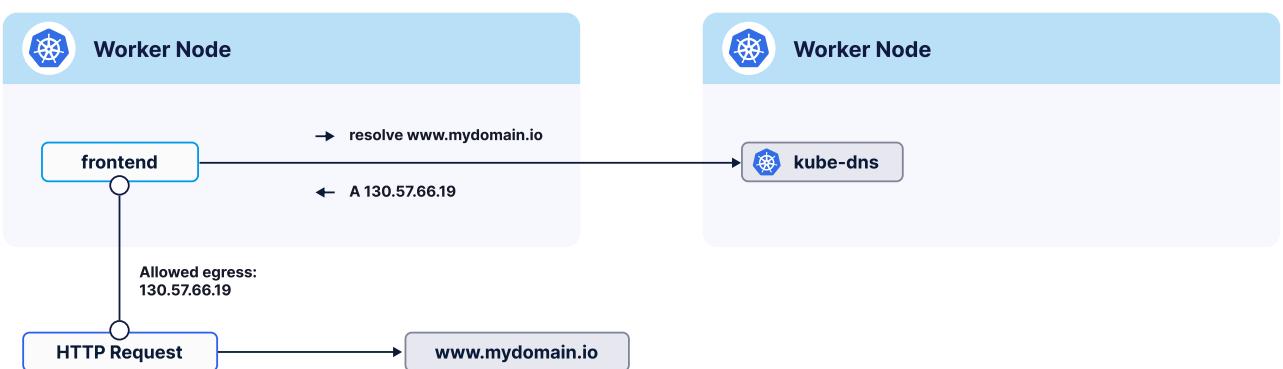


Рисунок 2. Политики Cilium в отношении DNS

Прозрачная защита API-интерфейсов

Cilium позволяет вам защищать современные прикладные протоколы, такие как HTTP, gRPC и Kafka, с беспрецедентной степенью детализации.

Традиционные брандмауэры работают на уровнях 3 и 4, где протоколу, запущенному на определенном порту, либо полностью доверяют, либо он полностью блокируется. В отличие от этого, **Cilium** обеспечивает поддержку уровня 7 с возможностью фильтрации запросов отдельных прикладных протоколов, обеспечивая точное управление трафиком. Для примера:

- **HTTP-запросы:** Разрешать все HTTP-запросы с использованием метода `GET` и пути `/public/.*`, отклоняя все остальные запросы.
- **Сообщения в Kafka:** Разрешить службе `service1` создавать в топике Kafka `topic1` и `service2` использовать в `topic2`, отклоняя все остальные сообщения Kafka.
- **Заголовки HTTP:** Требуется, чтобы заголовок HTTP `X-Token: [0-9]+` присутствовал во всех вызовах REST.

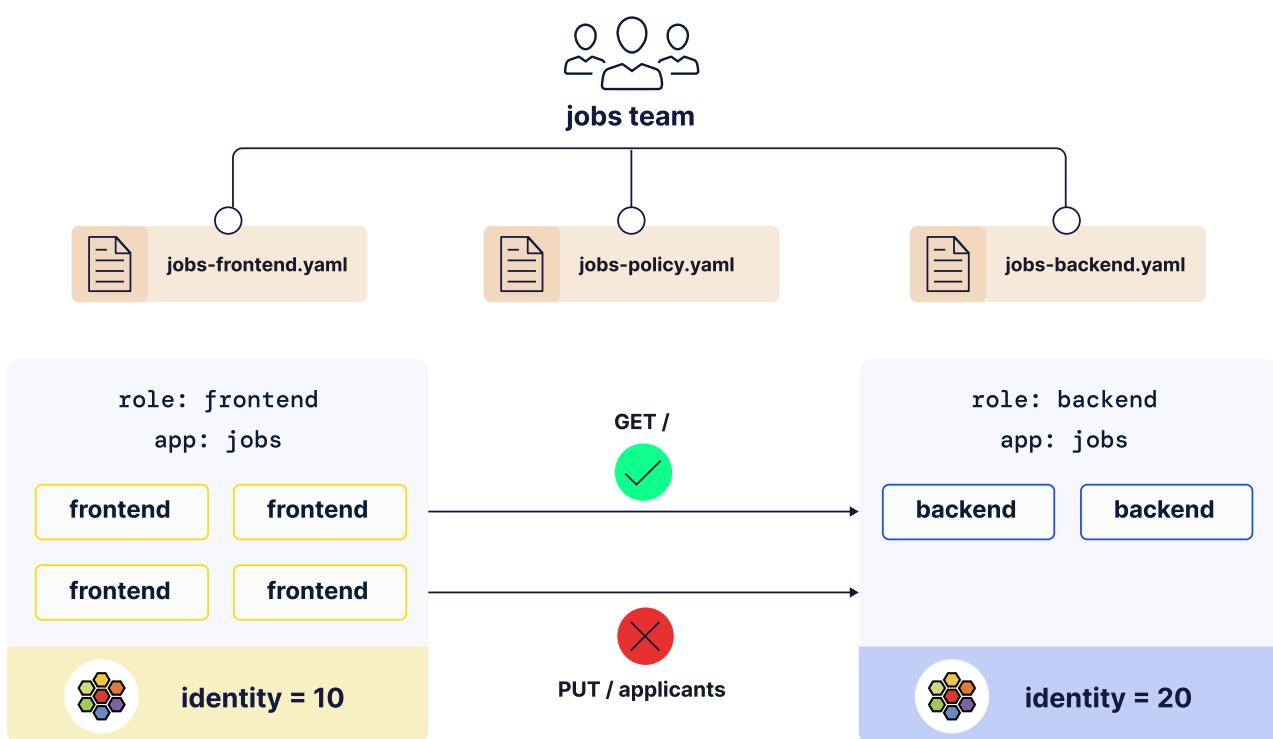


Рисунок 3. Правило 7-го уровня Cilium на примере HTTP

Подходы Kubernetes к внедрению сетевой политики

Существуют различные подходы к внедрению сетевых политик. Однако каждый из них потребует тщательного рассмотрения и будет иметь определённые последствия.

Сначала Мониторинг, только потом Политики

Эффективной стратегией внедрения сетевых политик является подход «сначала мониторинг, затем применение политик». В этом случае инструменты мониторинга за сетью, такие как Hubble (инструмент мониторинга в Cilium), отслеживают трафик и позволяют владельцу собирать данные о взаимодействии сервисов, выявлять закономерности и обнаруживать потенциальные проблемы, не нарушая трафик предварительным применением сетевых политик.

Благодаря этой информации, можно постепенно применять правила — разрешать трафик там, где это имеет смысл, и ужесточать политику в случае необходимости. Благодаря первоначальному мониторингу появляется эталонный шаблон потоков трафика и рабочих нагрузок, с которым можно сравнивать ситуацию до и после применения политик.

Уведомляйте, но не блокируйте

Стратегия “уведомлять, но не блокировать” снижает риск неправильных настроек и простоев в работе, позволяя командам получать информацию перед применением ограничительных политик. Она особенно полезна на ранних этапах внедрения сетевых политик или при быстром развитии приложений.

При развертывании Cilium на платформе Kubernetes в агент Cilium можно добавить следующую конфигурацию `--policy-audit-mode=true`. Когда включен режим [Policy Audit Mode²](#), сетевые политики не применяются, поэтому этот параметр не рекомендуется использовать для продового окружения. Режим аудита политик поддерживает аудит сетевых политик, реализованных на сетевых уровнях 3 и 4.

После активации можно проверить результат применения политики, как показано в примере ниже:

```
# hubble observe flows -t policy-verdict --last 1
Feb 7 12:53:39.168: default/tiefighter:54134 (ID:31028) ->
default/deathstar-6fb5694d48-5hmds:80 (ID:16530) policy-verdict:none AUDITED
(TCP Flags: SYN)
```

В приведенном выше примере мы можем видеть, что Под `deathstar-6fb5694d48-5hmds` получил трафик от Пода `tiefighter` который не соответствует ни одной из настроенных политик (вердикт по политике: проверка не проводилась, `policy-verdict:none AUDITED`).



Запрещать или не запрещать по умолчанию

Использование подхода “запрещать по умолчанию” гарантирует, что каждый случай межсервисного взаимодействия должен быть явно разрешен с помощью сетевой политики, поскольку в противном случае весь трафик по умолчанию будет запрещен. Несмотря на то, что этот подход является наиболее безопасным, т.к. приоритетное внимание уделяется безопасности, он сопряжен со значительным риском неправильных настроек. Такие настройки, например использование некорректной метки для идентификации рабочих нагрузок, к которым применяется политика, могут привести к простою или недоступности приложения. Поэтому применять такой подход к использованию на начальном этапе, когда у вас в кластере уже есть рабочие нагрузки, не рекомендуется.

Кроме того, такой подход создает большие трудности для разработчиков при внедрении и поддержании изменений в разрабатываемых приложениях. Настоятельно рекомендуется ознакомиться со всеми необходимыми сервисами, работающими в вашем кластере Kubernetes, и ознакомиться с документацией, касающейся необходимой настройки сетевых политик Cilium.

По умолчанию Cilium разрешает весь входящий и исходящий трафик между ендпоинтами, если он явно не ограничен. Однако, когда сетевая политика применяется к ендпоинту, она переходит в состояние запрета по умолчанию, в котором разрешен только явно разрешенный трафик. Это поведение применяется для каждого направления:

- ● **Входящий трафик:** Если правило выбирает конечную точку (endpoint) и содержит раздел ingress, эта точка переходит в режим «запрещено по умолчанию» для входящего трафика.
- ● **Исходящий трафик:** Если правило применяется к конечной точке (endpoint) и содержит секцию egress, эта точка переходит в режим «запрещено по умолчанию» для исходящего трафика.

По сути, ендпоинты ничем изначально неограничены, но как только любая политика к ним применена, они переключаются в режим запрета по умолчанию в направлении, указанном политикой (ingress или egress).

Существуют сценарии, в которых вы можете захотеть применить политики, не запуская режим запрета по умолчанию для выбранных ендпоинтов. Это регулируется при помощи поля `EnableDefaultDeny`. Если это поле отключено, правило будет применено без перевода конечной точки в режим запрета по умолчанию.

Например, можно реализовать политику, которая перехватывает весь DNS-трафик, разрешая при этом другие виды трафика, даже если это первоначальный набор политик для ендпоинта. Эта стратегия позволяет администраторам поддерживать наблюдаемость, например, отслеживать запросы DNS, без опасности непреднамеренного блокирования трафика при развертывании первой политики в кластере. Однако, как и при любом развертывании сетевой политики, в том числе для обеспечения наблюдаемости, сохраняется риск неправильной настройки, которая может привести к непреднамеренной блокировке трафика.

```
apiVersion: cilium.io/v2
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: intercept-all-dns
spec:
  endpointSelector:
    matchExpressions:
      - key: "io.kubernetes.pod.namespace"
        operator: "NotIn"
        values:
          - "kube-system"
      - key: "k8s-app"
        operator: "NotIn"
        values:
          - "kube-dns"
  enableDefaultDeny:
    egress: false
    ingress: false
  egress:
    - toEndpoints:
        - matchLabels:
            io.kubernetes.pod.namespace: kube-system
            k8s-app: kube-dns
        toPorts:
          - ports:
              - port: "53"
                protocol: TCP
              - port: "53"
                protocol: UDP
        rules:
          dns:
            - matchPattern: "*"
```

Компания Schuberg Philis добилась соответствия стандартам PCI-DSS при помощи решений от Isovalent, Cilium и соблюдения принципов Zero Trust

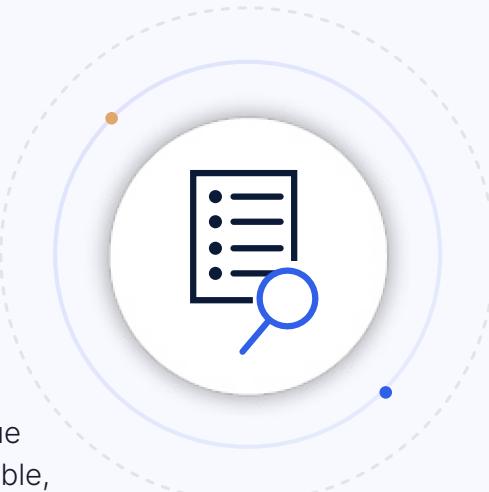
Специалисты Schuberg Philis из команды Критически важных систем помогли клиенту перенести инфраструктуру, из локального окружения в облако. Ключевой сложностью было создание облачного решения, которое бы обеспечивало соответствие требованиям PCI-DSS с минимальными трудозатратами на поддержку.

Ключевые требования:

- ● **Сетевая модель Zero Trust**
- ● **Мультитенантность**
- ● **Простота эксплуатации**

Пример из практики³ →

чтобы изучить опыт успешного внедрения Isovalent Enterprise for Cilium в инфраструктуру AWS, которое обеспечило применение политик безопасности "по умолчанию запрещено", возможность работы в режиме мультитенантности, сбор метрик и мониторинг через Hubble, маршрутизация трафика с использованием Cilium Egress Gateway.



Управление рисками

Основанный на оценке рисков подход к сетевой безопасности позволяет сосредоточиться на защите наиболее уязвимых областей платформы без чрезмерного усложнения операций. Вместо повсеместного применения строгих политик необходимо начинать с мониторинга структуры трафика и определения приоритетов критически важных служб, представляющих наибольший риск.

Чтобы это реализовать, начните определять ключевые показатели безопасности и использовать такие инструменты, как Cilium и Hubble, для мониторинга трафика. Отслеживайте, какие сервисы подключены к внешним сетям, анализируйте потоки трафика и выявляйте закономерности, которые могут указывать на потенциальные уязвимости.

На практике это может выглядеть как мониторинг клиентского сервиса, который взаимодействует с внешними API. Можно применить более строгие политики, чтобы ограничить доступ к доверенным конечным точкам, оставив менее важные сервисы с более широкими разрешениями. Со временем уточняете эти политики на основе наблюдаемого трафика и эффективности политики.

Чтобы сделать эту стратегию эффективной, отслеживайте такие показатели, как количество доступных сервисов, объемы входящего и исходящего трафика и вердикты по потокам (разрешено/запрещено). Они помогут выявить ошибки в настройках и указать, где необходимо ужесточить политики безопасности, не перегружая инфраструктуру.

Стратегия дизайна Сетевых политик

Команды разработчиков и Мультитенантность

В типичной среде **Kubernetes** может работать несколько команд — от разработчиков до пользователей приложений — они совместно используют кластеры, каждый из которых управляет своими приложениями в своём пространстве имён. Применение сетевых политик для пространства имен (**namespace scope**) часто является основополагающим шагом в обеспечению безопасности рабочих нагрузок, поскольку позволяет изолировать ресурсы и контролировать входящий и исходящий трафик в пределах этого пространства имен. Однако ответственность за определение и применение более широких стандартов безопасности обычно ложится на инженеров команды платформы или администраторов кластера, которые должны обеспечивать соблюдение требований безопасности и нормативных положений во всем кластере.

В то время как инженеры команды платформы контролируют политики в масштабах всего кластера, команды разработчиков приложениями сосредоточены непосредственно на создании и развертывании своих сервисов. Эти команды часто полагаются на администраторов, которые обеспечивают необходимые ограничения и средства контроля доступа. Задача состоит в том, чтобы поддерживать строгие политики безопасности, не препятствуя гибкости разработчиков. Такой баланс гарантирует, что разработчики смогут быстро внедрять приложения, соблюдая при этом стандарты безопасности и соответствия требованиям.

Хотя сетевые политики ограниченные пространством имён предоставляют надежный начальный уровень безопасности, политики же прикладного уровня (**application-level policies**) могут быть применены с использованием меток и детализированных средств их выбора. Политики могут быть применены к определенным ресурсам приложениям (например `app=store`) с помощью соответствующих меток, которые позволяют осуществлять детальный контроль внутри пространства имен. Это гарантирует, что, хотя область действия сетевой политики обычно привязана к пространству имен, приложения в пределах этого пространства имен могут быть защищены на основе уникальных требований каждой службы, используя мощные возможности Cilium по внедрению политик на основе меток.

Сочетая политики, ориентированные на область пространства имен, с настройками меток на уровне приложений, инженеры команды платформы могут обеспечить согласованную всеобъемлющую защиту для всех потребителей, позволяя при этом каждой команде поддерживать безопасность своих конкретных ресурсов приложения.

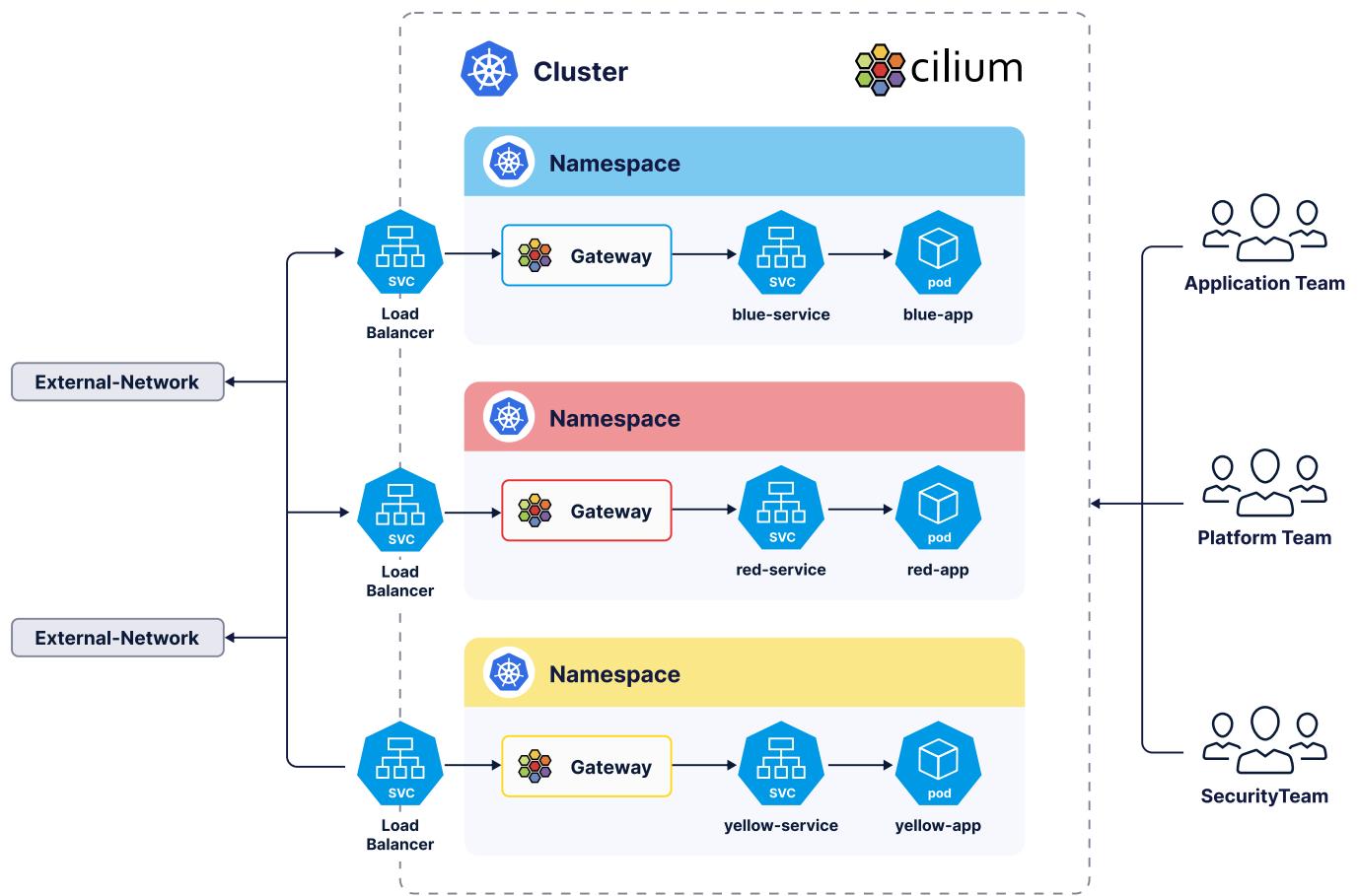


Рисунок 4. Мультитенантный режим в Kubernetes при помощи Cilium



Ключевые угрозы сетевой безопасности Kubernetes

Мы выделяем четыре основных типа рисков, связанных с сетью **Kubernetes**. Эти типы рисков важны для разработки эффективных сетевых политик вашей инфраструктуры. Эти четыре типа рисков показаны на следующей диаграмме.

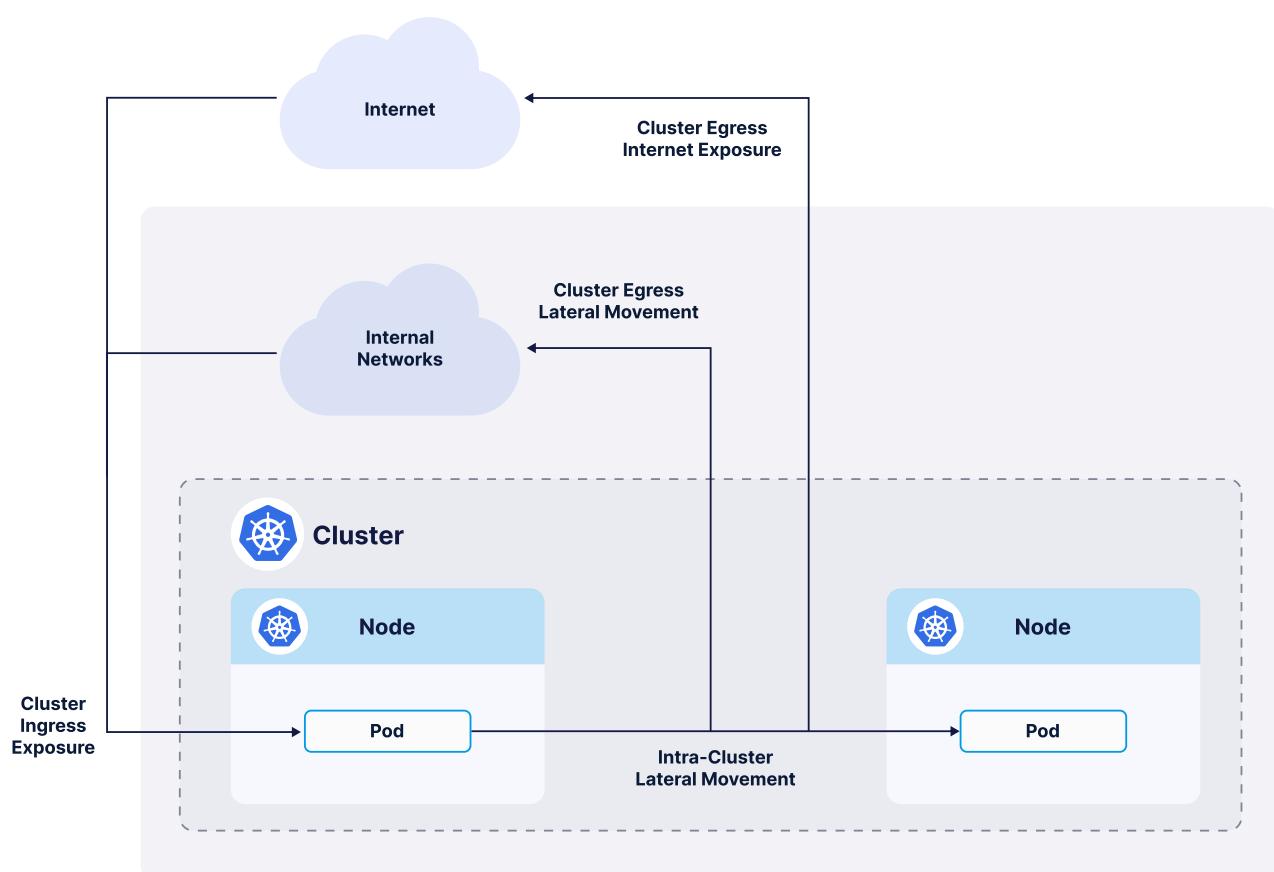


Рисунок 5. Мультитенантность в Kubernetes с помощью Cilium

1

Доступ к кластеру через Ingress:

- ● К этой категории относятся все сервисы, доступные из сети интернет, представляющие прямой канал для внешних атак.
- ● Сервисы, доступные из внешних сетей внутри ЦОД или Виртуального Частного Облака (VPC), могут потенциально привести к несанкционированному доступу.
- ● Случайное использование LoadBalancer или NodePort вместо ClusterIP для внутренних сервисов может непреднамеренно расширить зону потенциальных атак.
- ● Маршрутизация между подами внутри сети, разрешающая прямое подключение рабочих нагрузок, расширяет поверхность для потенциальных атак.

2

Доступ к кластеру через Egress:

- ● Возможность установки исходящих подключений в интернет создает риск утечки данных, запуска криптомайнинга и взаимодействие с управляемыми серверами злоумышленников — все это распространенные векторы атак.



3

Боковое перемещение внутри кластера:

- По умолчанию **Kubernetes** разрешает весь трафик внутри кластера, что позволяет злоумышленникам перемещаться по кластеру в поперечном направлении, если они получают доступ.

4

Cluster Egress Lateral Movement:

- Разрешенные исходящие подключения к виртуальным машинам, физическим серверам в сети, метаданным облака, API-серверам **Kubernetes** или другим кластерам **Kubernetes** создает дополнительные возможности для перемещения злоумышленников внутри сети. Это может привести к компрометации других систем и повышению привилегий в сети.

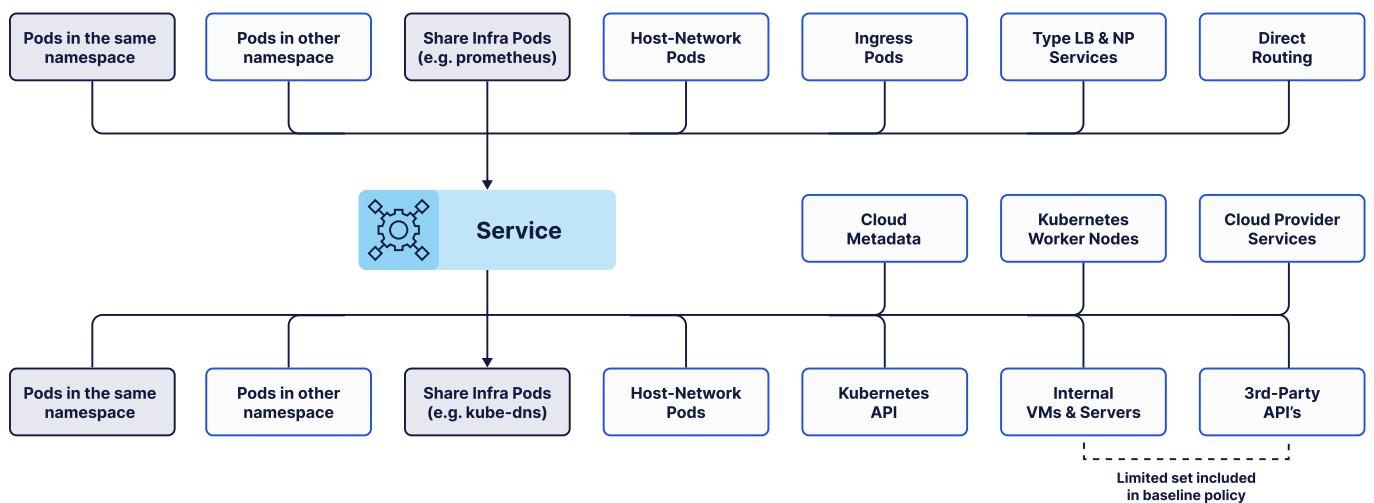


Рисунок 6. Виды угроз безопасности

Оптимизация доступов

Наша основная цель – обеспечить корректную работу приложений, сократив при этом неиспользуемый доступ к сети, который может привести к ненужному риску. Сводя к минимуму избыточный доступ, мы можем ограничить уязвимости, не влияя на сервисы, необходимые приложению.

Эффективный метод минимизации риска заключается в сужении потенциального радиуса распространения в кластерах Kubernetes. По умолчанию Kubernetes разрешает неограниченную связь между кластером и подключением к нему, а также между модулями в разных пространствах имен. Этот параметр по умолчанию может значительно повысить риск масштабного воздействия на ваш кластер, поскольку один скомпрометированный под потенциально может привести к атаке, затрагивающей все рабочие нагрузки. Сокращение масштабов взаимодействия имеет важное значение для повышения уровня безопасности вашей среды.

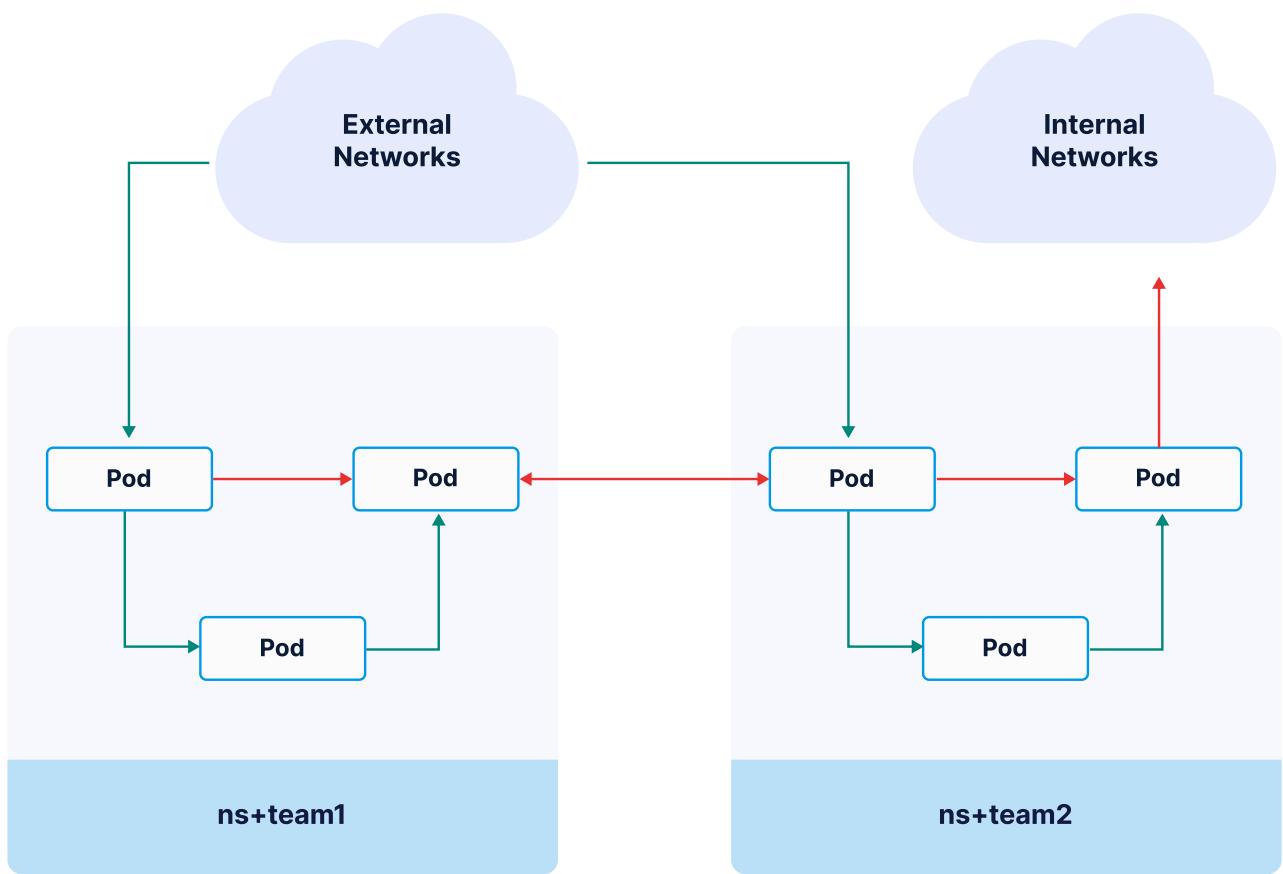


Рисунок 7. Ограничение радиуса поражения внутри пространств имен и между ними

Анализ рисков

Хотя сегментация на уровне Namespace, эффективно снижает потенциальные риски, управление растущим объемом сетевого трафика остается сложной задачей. Для эффективного управления рисками необходимо провести аудит всех сетевых подключений, оценить степень их уязвимости и на основе этого приоритизировать их конфигурирование.

Определение приоритетов

Для приоритизации необходимо изучить конкретные критерии:

- 1 Количество сервисов доступных извне через NodePort, LoadBalancer, или ExternalIP
- 2 Количество сервисов доступных через Ingress/Gateway API
- 3 Количество сервисов доступных из других Namespace
- 4 Количество сервисов имеющих разрешение на соединения с внешними сетями или интернетом

Инвентаризация неиспользуемого доступа

Использование современных инструментов наблюдения, таких как **Hubble**, позволяет выявлять и устранять неиспользуемый доступ. Подобные системы анализируют расхождения между сетевыми правилами и реальным трафиком, показывая, какие политики отсутствуют или избыточны. Такой подход повышает безопасность, оставляя только необходимые соединения.

Но не каждый доступ — это уязвимость. Риск зависит от роли сервиса и его данных. Например, доступ **kube-dns** ко всем подам — это необходимость, а аналогичный доступ для сервиса с данными — критическая уязвимость. Фронтенд должен быть доступен извне, а бэкенд **Redis** — нет. Даже необходимый внешний доступ (egress) следует сужать до конкретных целей, чтобы снизить поверхность атаки.

Воспользуемся аналогией:

“

Представим оживленный город, где каждое здание представляет собой отдельный прикладной сервис в нашем кластере Kubernetes. В этом городе есть определенные здания, такие как центральное почтовое отделение, которые должны быть доступны каждому. Это похоже на службу `kube-dns`, которая должна быть доступна для любой рабочей нагрузки в кластере. Ее использование является как намеренным, так и необходимым и не представляет угрозы безопасности.

Как бы то не было не все здания должны быть открыты для всех. Представим многоквартирный дом с незапертными дверями, в которые может войти любой житель города. Такой неограниченный доступ представляет угрозу безопасности, как если бы служба приложений была доступна из всех модулей кластера.

Теперь представим главную библиотеку города, которая должна быть открыта для общественности, чтобы предоставлять информацию. Это похоже на фронтенд приложение, подключенное через порт 443 к внешней сети. Ее доступность необходима для функционирования города. Другой пример, представим хранилище с высокой степенью защиты, в котором хранятся конфиденциальные документы. Это хранилище представляет собой сервер `Redis`, используемый для кэширования результатов работы базы данных. В отличие от библиотеки, хранилище не должно быть доступно для всех, и его двери должны оставаться надежно запертими.

Наконец, рассмотрим возможность использования городской курьерской службы для доставки посылок. Хотя эта служба необходима, она не должна иметь права свободно перемещаться куда бы то ни было. Аналогичным образом, может потребоваться аналогичная служба внутри кластера, которая подключается к внешним сетям, но доступ к ней должен быть ограничен определенными пунктами назначения, чтобы свести к минимуму ненужное воздействие.

В таком городе крайне важно поддерживать баланс между доступностью и безопасностью. Тщательно определяя, какие здания доступны и для кого, мы обеспечиваем бесперебойную работу городских служб без ущерба для безопасности. Аналогичным образом, продуманные сетевые политики помогают защитить критически важные службы, сохраняя при этом необходимый доступ в кластере Kubernetes.

”

Последствия избыточности

Использование инструментов сетевого наблюдения, таких как Hubble, позволяет преобразовывать отслеживаемые потоки трафика в конкретные правила или сетевые политики Kubernetes. Однако прямое преобразование этих потоков в политики может привести к сложности и нестабильности работы. Эта проблема подчеркивается динамическим характером IP-адресов и портов для служб как внутри, так и за пределами нашего кластера Kubernetes. К счастью, при использовании Hubble для создания сетевых политик он использует ярлыки рабочих нагрузок Kubernetes как часть идентификаторов Cilium, рассмотренных ранее. Кроме того, приложения часто зависят от инфраструктуры кластера и общих служб, таких как kube-dns, служб логгирования и хранилищ данных. Частые обновления приложений и служб командой разработчиков также требуют тщательного рассмотрения, что еще больше усложняет эффективное управление сетевыми политиками.

Представьте себе управление бизнес-системой, разработчики которого постоянно насыщают её новыми микросервисами и новыми версиями этих микросервисов. Создание парных правил, для внутреннего и внешнего трафика, для каждого микросервиса может привести к значительной сложности в работе и появлению огромного количества сетевых политик. Такой подход может привести к сбоям в работе при запуске новых приложений, поскольку для правильной работы каждого нового или обновленного приложения требуются соответствующие модификации сетевых политик. Огромное количество парных сетевых политик может стать неуправляемым, что затруднит их оперативное обслуживание. Этот пример подчеркивает необходимость более стратегического подхода, который обеспечит баланс между требованиями безопасности и операционной эффективностью, обеспечивая надежную защиту и бесперебойную разработку приложений.



Инициализация стратегии Сетевых политик

Чтобы избежать сложностей, связанных с чрезмерно конкретными правилами, и при этом обеспечить эффективную безопасность, рекомендуется начать с общей или “крупнозернистой” стратегии сетевой политики, организованной на основе пространств имён.

Такая политика означает установление более общих правил, применимых к большим группам ресурсов, а не конкретных, подробных правил для каждого отдельного приложения. Такой подход сводит к минимуму необходимость частой модификации политик, даже по мере развития приложений и изменения их сетевого поведения. Начав с более широких политик, вы обеспечите баланс между надежной безопасностью и гибкостью в эксплуатации. Как только будет разработана общая политика для данного приложения, вы сможете постепенно внедрять более подробные, детализированные политики для дальнейшего повышения безопасности.

В следующих шагах описано, как эффективно перейти от этой первоначальной настройки к более конкретным сетевым политикам.



1

Коммуникации внутри пространства имен:

- Начните с открытых политик:** изначально разрешите весь входящий и исходящий трафик в пространстве имен, чтобы обеспечить плавную адаптацию приложения.
- Наблюдайте за потоками трафика:** используйте инструменты и метрики Hubble observability для мониторинга трафика и определения необходимых связей между службами в пространстве имен.

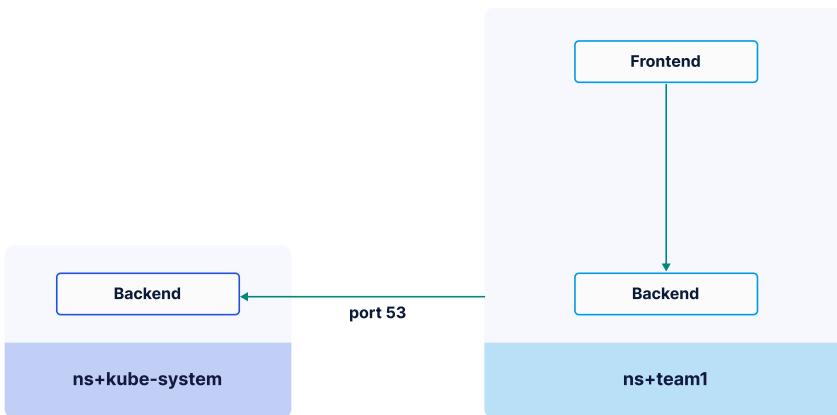


Рисунок 8. Изначальная схема трафика внутри пространства имён

2

Внутриклusterный трафик:

- Взаимодействие в пределах кластера:** Разрешите внутриклusterный трафик для поддержки взаимодействия внутренних сервисов и их зависимостей. Как правило, этот трафик связан с работой служб мониторинга или инфраструктуры.

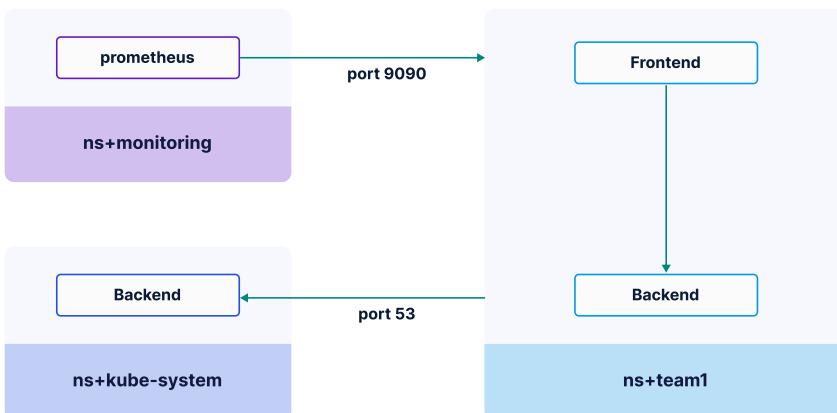


Рисунок 9: Разрешение трафика для служб мониторинга

3

Внешний доступ:

- Разрешение внешних подключений: Разрешайте входящие подключения в кластере через соответствующие сервисы, такие как LoadBalancer, NodePort или Ingress. Настройте сетевые политики с необходимыми правилами для входящего трафика (Ingress). По возможности ограничивайте область источников, используя IP-адреса, префиксы CIDR или FQDN. [Правила FromFQDN сетевых политик были представлены в Cilium Enterprise 1.13.](#)⁴

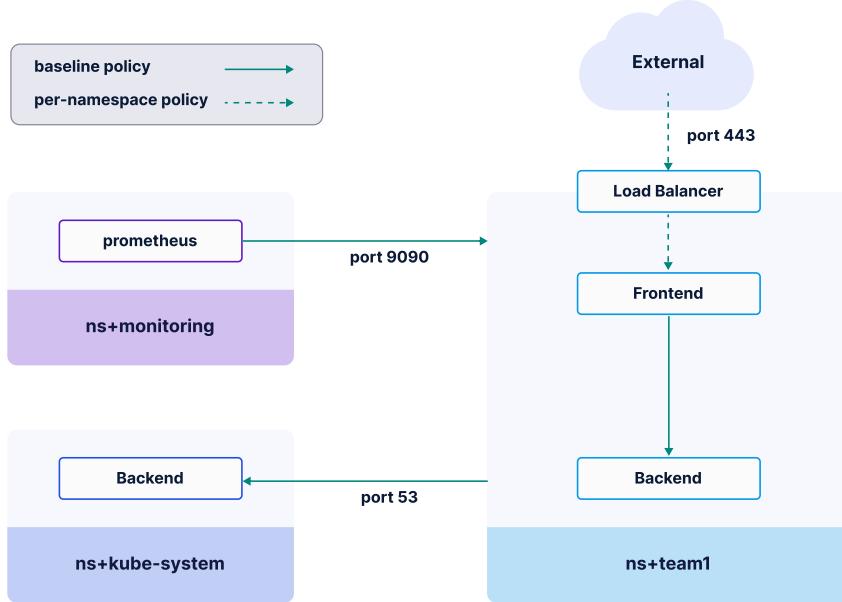


Рисунок 10: Разрешение внешних подключений

- Разрешите исходящие подключения к внешним сервисам, необходимым для приложения или службы. По возможности ограничивайте адреса назначения, используя IP, префиксы CIDR или FQDN. Это, например, снизит риск потенциальных попыток утечки данных.

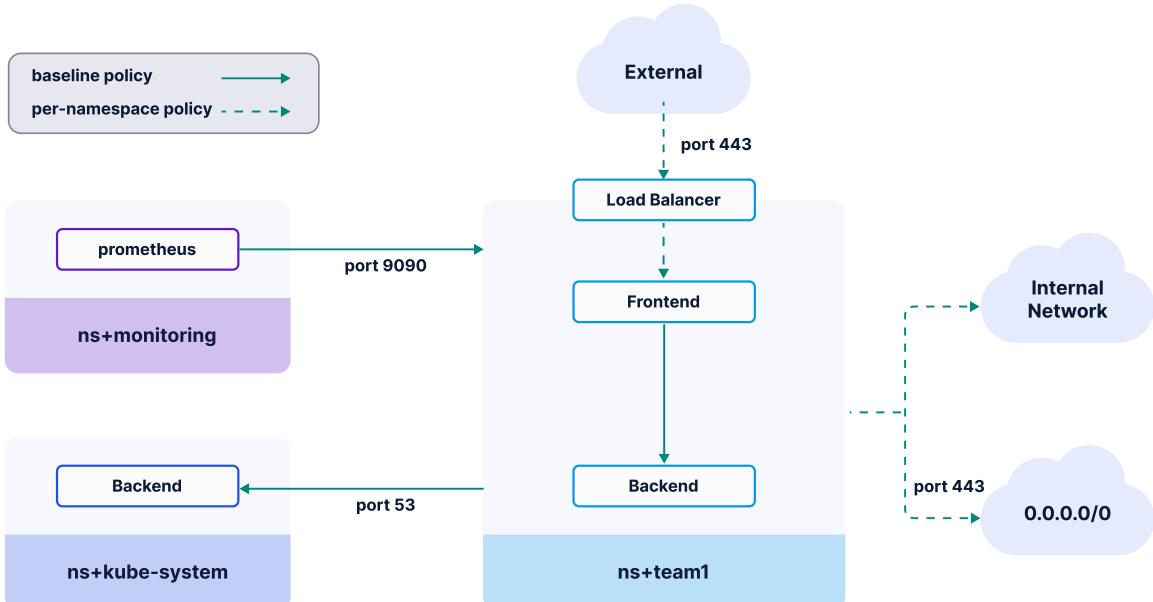


Рисунок 11: Начальные исходящие подключения

- Повысьте безопасность, настроив правила уровня 7 (L7) для разрешения только конкретных HTTP/API-вызовов, путей и методов.
 - Автоматизация процессов: Оптимизируйте рабочие процессы реагирования на инциденты и формирования отчетов, для экономии времени и ресурсов.
 - Ограничьте доступ к определенным URL-путям, чтобы гарантировать доступность только необходимым подключениям.

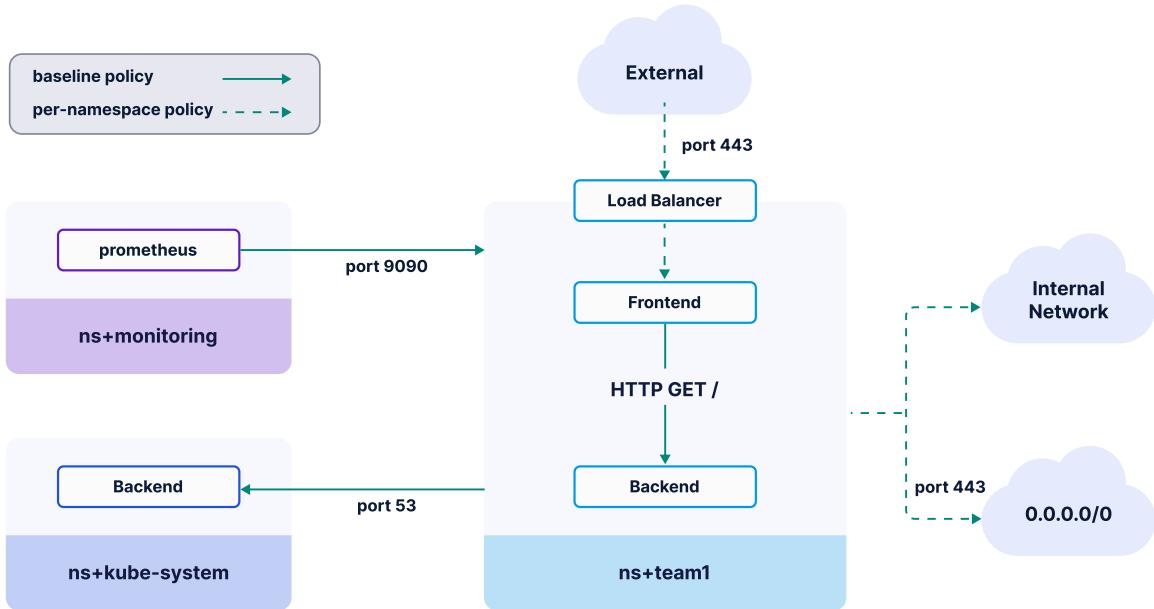


Рисунок 12: Защита HTTP-запросов и взаимодействий между клиентской и серверной частью приложения.

- Используйте ресурсы Ingress или Gateway API в сочетании с сетевыми политиками для ограничения внешнего доступа к сервисам приложений.

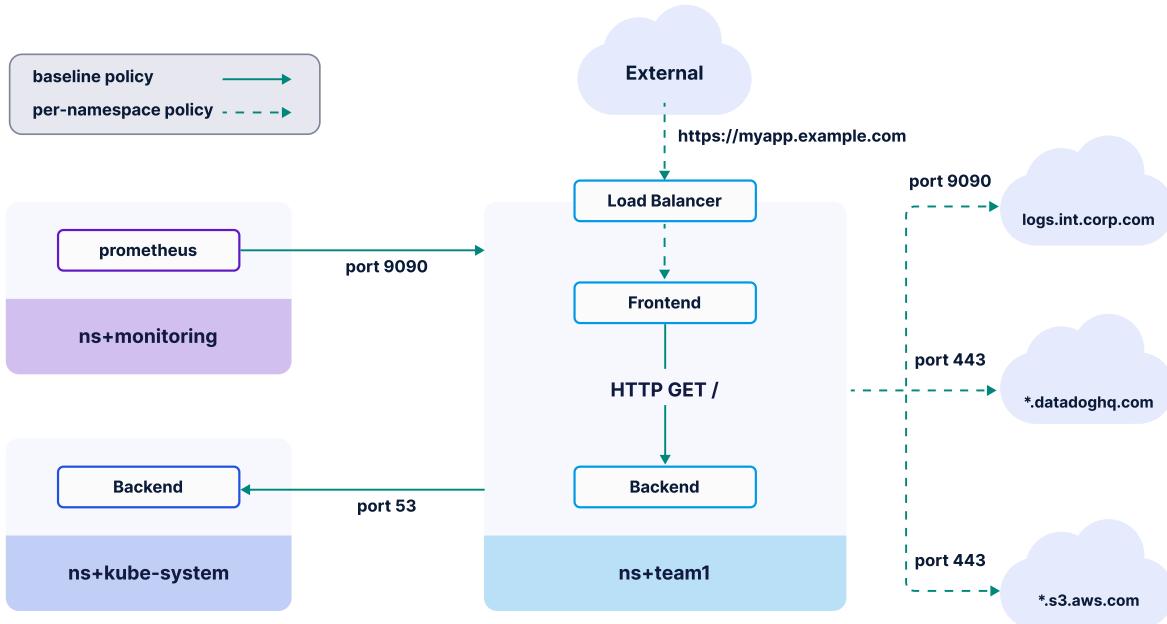
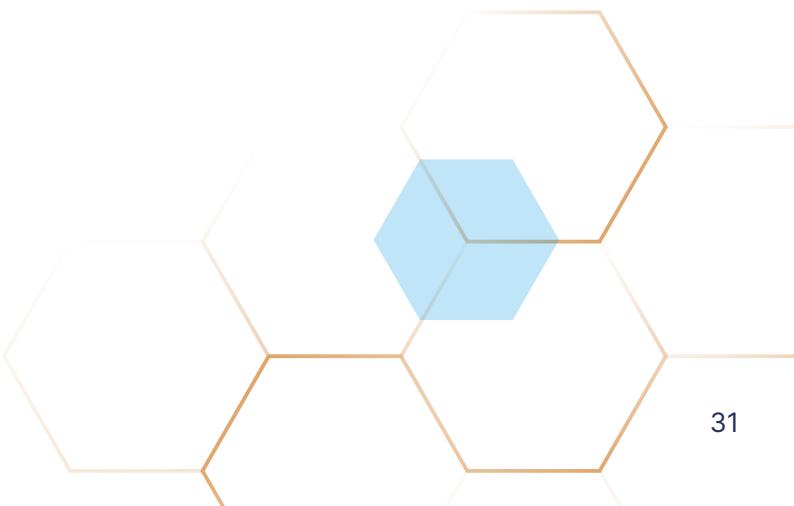


Рисунок 13. Детализированные правила исходящего трафика, основанные на FQDN и портах подключения.

Опциональное взаимодействие между пространствами имён:

- ● Определите требования к сервисам: выявите службы, которым необходимо взаимодействовать между пространствами имён, и конкретные порты, которые они используют.
- ● Разрешите необходимый трафик: при необходимости настройте сетевые политики, разрешающие требуемое взаимодействие между пространствами имён, чтобы гарантировать бесперебойную работу основных сервисов.



Приоритизация работы с пространствами имён

Обычно в различных пространствах имён запущено несколько приложений. Каждое из них имеет разный уровень риска экспозиции или доступ к ресурсам с конфиденциальными данными. Важно определить приоритетность пространств имён для первоочередной защиты.

Оцените и расставьте приоритеты для пространств имён в соответствии со следующими критериями:

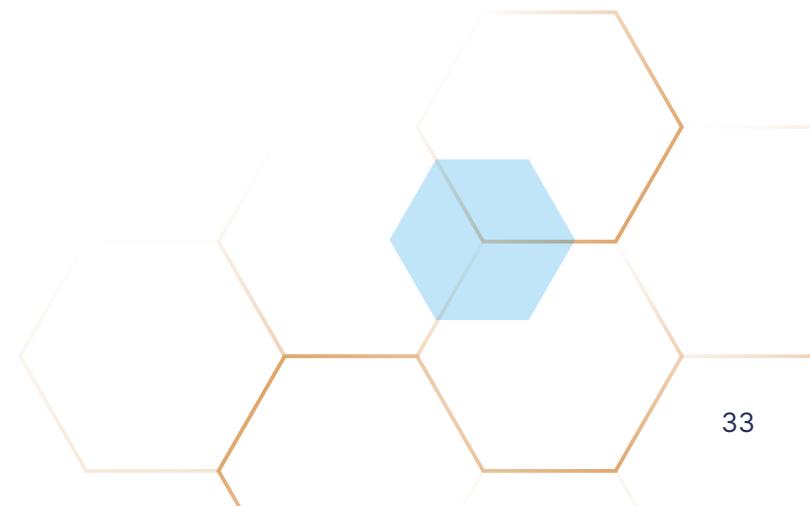
- ● **Уровень безопасности:** Выявите приложения, наиболее уязвимые к нарушениям безопасности исходя из характера их данных, критичности для бизнеса или их роли в инфраструктуре.
- ● **Внешняя доступность и потребность в сетевом взаимодействии:** Проанализируйте текущий уровень доступности сервиса в сравнении с фактическими потребностями в подключениях, чтобы выявить службы, где сетевой доступ можно минимизировать без ущерба для доступности приложения.



Глобальные сетевые политики кластера

Следующим шагом в защите ваших кластеров Kubernetes является настройка глобальных сетевых политик ([Cilium Cluster Wide Network⁵](#)) для обеспечения единого базового уровня безопасности и защитных механизмов. Типичными примерами правил в глобальных сетевых политиках являются:

- ● **Доступ внутри пространства имён:** Разрешите весь входящий и исходящий трафик в пределах одного пространства имён, чтобы обеспечить бесперебойное внутреннее взаимодействие, это также сохранит общую безопасность кластера.
- ● **Исходящий трафик к внешним сервисам:** Обязательно настройте необходимые правила для исходящего трафика к общедоступным сервисам всего кластера, вроде kube-dns и Prometheus. Кроме того, разрешите трафик к общим внешним инфраструктурным ресурсам, определённым через CIDR или FQDN, таким как сервисы логирования, мониторинга и хранилище секретов (vault).
- ● **Входящий трафик к внутренним сервисам:** Если требуется, настройте сетевые политики, разрешающие входящий трафик из внешних источников. Для приёма трафика используйте ресурсы типа LoadBalancer. Идеальным решением будет настройка Gateway (например, с использованием HTTPRoute), чтобы точно определить, какой трафик уровня L7 и по каким URL-маршрутам должен быть доступен извне.
- ● **Блокировка по умолчанию для входящего/исходящего трафика (Default Deny):** Внедрите политику "запрещено по умолчанию" (default deny) для всего входящего и исходящего трафика, чтобы явно было разрешено только указанное в правилах взаимодействие. Этот подход обеспечивает, что в кластер и из него передаётся исключительно авторизованный трафик.



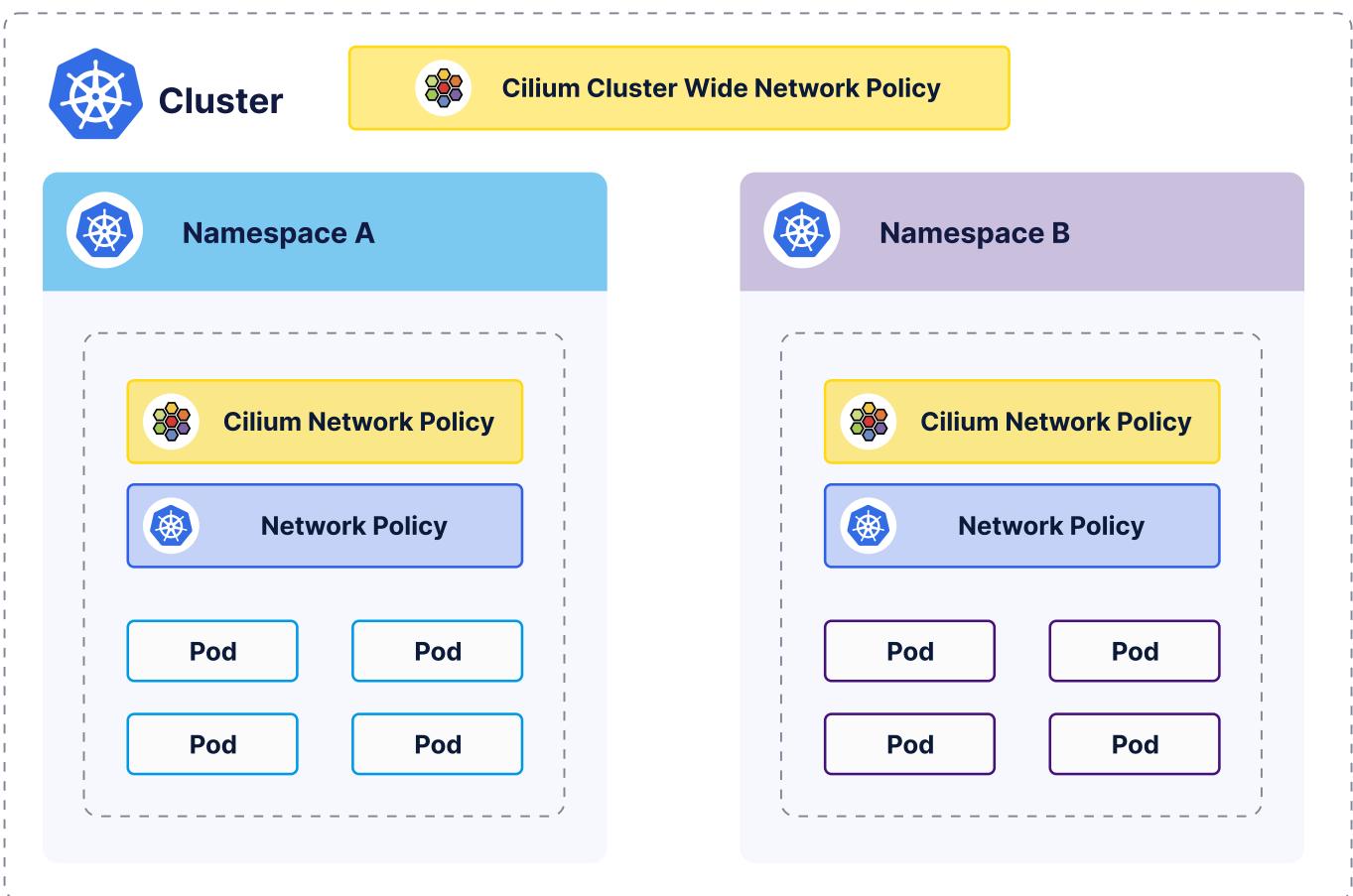


Рисунок 14. Глобальные сетевые политики Cilium

Ключевой сценарий использования глобальных политик — явное блокирование трафика к запрещенным адресатам, например, к IP-адресам, связанным с ограниченными странами, или возможность блокировки известных вредоносных IP-адресов. Внедрение таких политик обеспечивает соответствие нормативным и корпоративным стандартам и помогает защитить инфраструктуру от несанкционированного доступа. Этот подход, интегрированный в общую структуру безопасности, играет важную роль в выполнении требований соответствия, таких как "Общий регламент по защите данных" (GDPR), и предоставляет дополнительный уровень контроля над исходящим трафиком.



Трудности реализации сетевых политик без необходимых инструментов

Даже при наличии продуманной стратегии сетевых политик, её практическое внедрение может представлять серьёзную трудность, особенно при использовании традиционных инструментов или ручных процессов. С увеличением масштабов предприятия, растёт и сложность управления сетевыми политиками в разнородных средах, что делает ручные методы работы подверженными ошибкам и неэффективными.

Основные сложности, с которыми сталкиваются компании при реализации сетевых политик без надлежащих инструментов:

- ● **Недостаток наблюдаемости:** При отсутствии детального представления о том, как трафик движется между сервисами, трудно понять, какие политики действительно нужны и не блокируют ли существующие настройки важные коммуникации.
- ● **Сложность управления:** Ручное создание, обновление и применение сетевых политик для разнородных и постоянно меняющихся приложений увеличивает сложность, что ведёт к ошибкам конфигурации или слепым зонам, способным подвергнуть инфраструктуру угрозам.
- ● **Проблемы масштабируемости:** По мере масштабирования приложений растёт и количество взаимосвязей, это делает управление сетевыми политиками практически невозможным без средств автоматизации.
- ● **Трудности диагностики:** При неожиданной блокировке трафика сетевыми политиками, традиционные инструменты не обеспечивают достаточной наблюдаемости в реальном времени для быстрой диагностики и устранения проблем без простоя.

Эти вызовы наглядно демонстрируют важность использования инструментов сетевой наблюдаемости и безопасности. В следующем разделе мы покажем, как платформа Isovalent, построенная на основе Cilium и Hubble, предоставляет необходимые средства и полную видимость для эффективного внедрения сетевых политик и преодоления перечисленных трудностей.

От теории к практике: реальный пример

Теперь, когда мы рассмотрели подходы и стратегии внедрения сетевых политик, мы сосредоточимся на реализации реального примера с использованием платформы Isovalent на основе Cilium и Hubble.

В качестве примера используется демо-приложение на основе микросервисов от Google ([Micro-Services Demo provided by Google⁶](#)).

Онлайн Бутик состоит из 11 микросервисов, написанных на разных языках, каждый со своими шаблонами взаимодействия и зависимостями. Ниже представлена высокоуровневая архитектурная схема приложения;

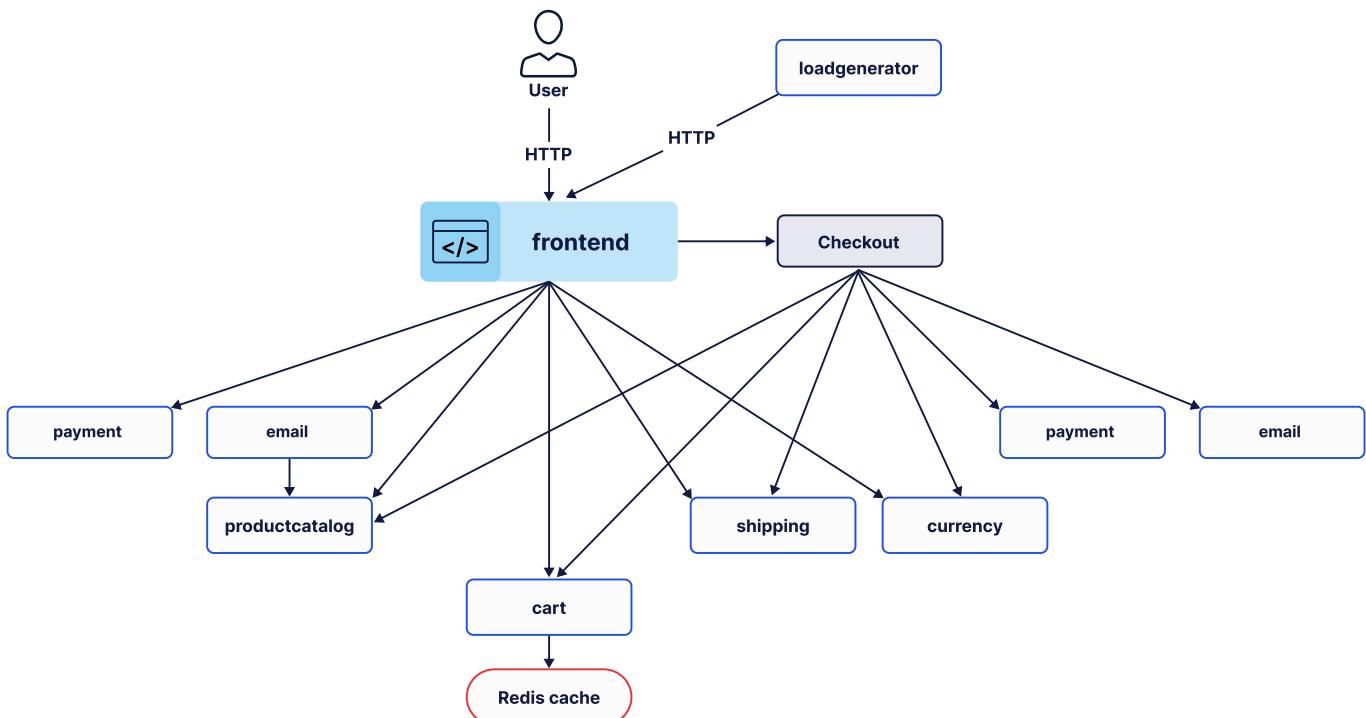


Рисунок 15. Схема подключения приложения для онлайн-магазина

В приведённой ниже таблице ресурсы описаны более подробно:

Сервис	Язык	Описание
<u>frontend⁷</u>	Go	HTTP-сервер для обслуживания веб-сайта. Не требует регистрации и аутентификации, автоматически генерирует идентификаторы сессий для всех пользователей.
<u>cartservice⁸</u>	C#	Хранит товары в корзине покупок пользователя в Redis и обеспечивает доступ к ним.
<u>productcatalogservice⁹</u>	Go	Отвечает за предоставление каталога товаров в виде JSON-файла, включая функции поиска и получения данных об отдельных продуктах.
<u>currencyservice¹⁰</u>	Node.js	Выполняет конвертацию валют, получаемых от Европейского центрального банка. Является сервисом с самой высокой нагрузкой (наибольшее количество запросов в секунду, QPS).
<u>paymentservice¹¹</u>	Node.js	Выполняет (тестовое) списание средств с предоставленной кредитной карты на указанную сумму и возвращает идентификатор транзакции.
<u>shippingservice¹²</u>	Go	Рассчитывает стоимость доставки на основе содержимого корзины. Осуществляет (тестовую) отправку товаров по указанному адресу.
<u>emailservice¹³</u>	Python	Отправляет пользователям (тестовое) письмо с подтверждением заказа.
<u>checkoutservice¹⁴</u>	Go	Управляет процессом оформления заказа: получает корзину, управляет оплатой, доставкой и уведомлением.
<u>recommendationservice¹⁵</u>	Python	Рекомендует другие товары на основе содержимого корзины.
<u>adservice¹⁶</u>	Java	Предоставляет текстовую рекламу на основе заданных контекстных ключевых слов.
<u>loadgenerator¹⁷</u>	Python/ Locust	Имитирует поведение пользователей, отправляя запросы по сценариям покупок.

Внедрение архитектуры Zero Trust

Модель безопасности Zero Trust основана на принципах проверки каждого соединения и отсутствия автоматического доверия к любой сущности — как внутри, так и за пределами вашей сети. Этот подход минимизирует поверхность атаки, разрешая только явно авторизованное и постоянно проверяемое взаимодействие.

Применяя принципы Zero Trust в Kubernetes, мы сосредотачиваемся на сокращении поверхности атаки на нескольких уровнях. Сетевые политики, такие как те, что обеспечиваются Cilium, позволяют детально контролировать, какие сервисы могут взаимодействовать как внутри кластера, так и с внешними ресурсами. Это гарантирует, что даже внутренний трафик проходит строгую проверку, снижая риск горизонтального перемещения злоумышленников.

В этом примере приложение изначально небезопасно, разрешая доступ к микросервисам в пространстве имён как внешнему, так и внутреннему трафику. При переходе на архитектуру Zero Trust цель разрешить только ожидаемые и легитимные сетевые потоки приложения, блокируя несанкционированный или потенциально вредоносный трафик.

Проблемы безопасности приложения включают:

- ● Фронтенд может обращаться к любым сервисам в интернете, что создаёт риск утечек или взломов данных.
- ● Неконтролируемая внутренняя коммуникация между микросервисами, это увеличивает риск горизонтального перемещения злоумышленника при компрометации одного из сервисов.
- ● Отсутствие контроля за входящим трафиком, это позволяет внешним источникам свободно обращаться к микросервисам без проверки.
- ● Критически важные сервисы, такие как платёжный сервис, не имеют специальных политик, ограничивающих их внешние подключения, что делает их уязвимыми.

Существенно снизить вектор атак можно, ограничив сетевые коммуникации только теми, что действительно необходимы — как внутри кластера, так и с внешними системами.

Далее мы поэтапно разберём процесс применения этих сетевых политик к демо-приложению, продемонстрировав, как построить защищённую и устойчивую инфраструктуру без ущерба для гибкости, требуемой современными динамичными средами.

Hubble

Hubble предоставляет высокую наблюдаемость, позволяя подробно наблюдать коммуникацию и поведение сервисов, а также состояние сетевой инфраструктуры. В основе — прозрачность, обеспечиваемая технологией eBPF. Hubble обеспечивает детальный мониторинг на уровне узла, кластера и даже между кластерами в распределённой среде (Cluster Mesh). Это помогает оптимизировать производительность, быстро находить и устранять проблемы, а также обеспечивать безопасность всей сети. Использование Hubble даёт необходимую информацию для принятия решений, повышает операционную эффективность и помогает поддерживать стабильную и защищённую среду Kubernetes.

Hubble UI предоставляет пользователям интерфейс как командной строки (CLI), так и веб-интерфейс (UI).

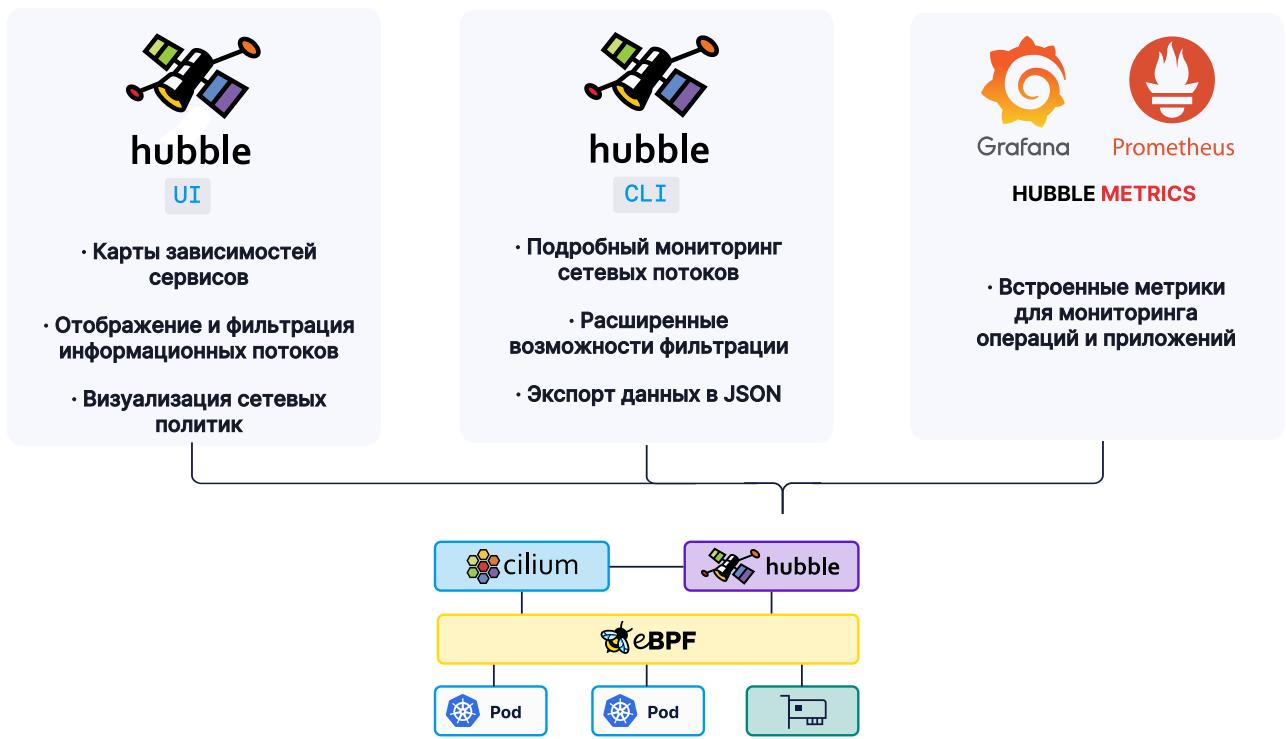


Рисунок 16. Обзор Hubble

Hubble CLI

Интерфейс командной строки Hubble CLI расширяет возможности, предоставляемые стандартными командами `kubectl`, добавляя детали сетевого уровня для запросов, такие как их статус и идентификаторы безопасности источника и назначения.

Интерфейс командной строки позволяет наблюдать сетевые потоки от агентов Cilium. А пользователи могут отслеживать эти потоки со своей локальной рабочей станции для диагностики проблем или мониторинга.

Ниже приведён пример вывода при мониторинге приложения, запущенного в пространстве имён `microservices-demo`, до применения каких-либо сетевых политик. Результат показан в свёрнутом виде, но есть возможность вывести полную информацию в формате JSON.

```
# hubble observe -n microservices-demo
Sep 13 12:10:11.267:
microservices-demo/loadgenerator-856dc76c97-ggjbn:38151
(ID:63434) <- kube-system/coredns-76f75df574-5gt56:53 (ID:31390)
to-endpoint FORWARDED (UDP)
Sep 13 12:10:11.268:
microservices-demo/loadgenerator-856dc76c97-ggjbn:43932
(ID:63434) -> microservices-demo/frontend-6d9488c857-xcnsq:8080
(ID:3374) to-overlay FORWARDED (TCP Flags: ACK, PSH)
```

В этих данных мы видим трафик между сервисом `loadgenerator` и сервисом `frontend` через порт 8080, который отвечает за имитацию пользовательской активности. Важно отметить, что весь трафик в настоящее время разрешён, так как не применяются сетевые политики для ограничения этого взаимодействия. Без политик такая открытая схема трафика представляет потенциальный риск для безопасности, поскольку сервисы вроде `loadgenerator` могут взаимодействовать с несанкционированными конечными точками.

Полный вывод в формате JSON содержит дополнительные детали, такие как HTTP-методы, коды ответов и другие данные уровня L7, которые особенно полезны для диагностики сложных сетевых шаблонов. Все потоки, фиксируемые Hubble, сохраняются в формате JSON и могут быть просмотрены с помощью CLI или UI Hubble. Эта детальная информация включает данные прикладного уровня (L7), такие как задержки, HTTP-коды ответов, методы, URL-адреса и заголовки, обеспечивая глубокую видимость поведения сервисов.



```
"17": {
    "type": 1,
    "latency_ns": 0,
    "record": {
        "oneofKind": "http",
        "http": {
            "code": 0,
            "method": "POST",
            "url": "http://recommendationservice:8080/hipstershop.RecommendationService/
ListRecommendations",
            "protocol": "HTTP/2",
            "headers": [
                {
                    "key": ":scheme",
                    "value": "http"
                },
                {
                    "key": "Content-Type",
                    "value": "application/grpc"
                },
                {
                    "key": "Grpc-Accept-Encoding",
                    "value": "gzip"
                },
                {
                    "key": "Te",
                    "value": "trailers"
                },
                {
                    "key": "User-Agent",
                    "value": "grpc-go/1.65.0"
                },
                {
                    "key": "X-Envoy-Internal",
                    "value": "true"
                },
                {
                    "key": "X-Request-Id",
                    "value": "75dc5bd9-c925-4770-93b5-a59c6c6ab4d3"
                }
            ]
        }
    }
},
```

Кроме того, с конкретным потоком может быть связана сетевая политика Cilium, что также фиксируется в JSON-выводе. В приведённых ниже данных поле `ingress_allowed_by` содержит информацию о том, какая сетевая политика разрешает трафик для данного потока.

```
{  
    "ingress_allowed_by": [  
        {  
            "name": "allow-ingress",  
            "namespace": "microservices-demo",  
            "labels": [  
                "k8s:io.cilium.k8s.policy.derived-from=CiliumNetworkPolicy",  
                "k8s:io.cilium.k8s.policy.name=allow-ingress",  
                "k8s:io.cilium.k8s.policy.namespace=microservices-demo",  
                "k8s:io.cilium.k8s.policy.uid=40768eb8-effe-440d-b3fb-9a937c4fb07c"  
            ],  
            "revision": "2"  
        }  
    ]  
}
```



Hubble UI

Hubble UI предоставляет графический интерфейс с возможностью визуализации и анализа на основе сетевых потоков, захваченных с платформы, на которой работает Cilium. Ниже можно увидеть карту сервисов (Service map), отображающую маршруты взаимодействия сервисов, запущенных в пространстве имён [microservices-demo](#).

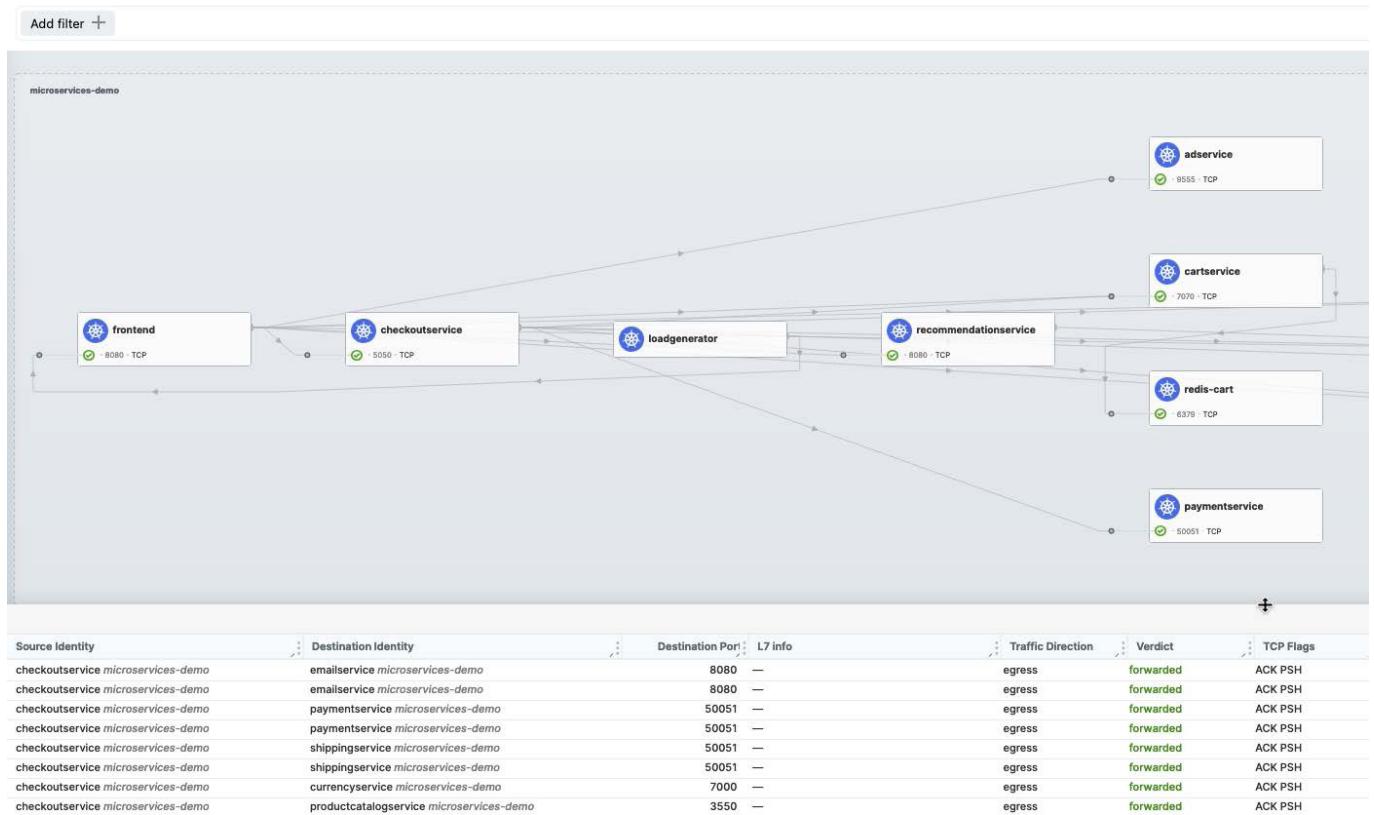


Рисунок 17: Hubble UI

Каждый сетевой поток, представленный в таблице под графиком карты сервисов, можно выбрать для просмотра дополнительной информации и метаданных, включая необработанный вывод в формате JSON, аналогично тому, как это делается в CLI Hubble.

Hubble, как часть платформы Isovalent, предоставляет возможность самостоятельного доступа для нескольких пользователей (Multi-Tenant Self-Service Access) через интеграцию с OpenID Connect (OIDC). Это позволяет подключить существующие платформы идентификации и авторизации, такие как Okta, Auth0 и другие.

В Hubble реализовано управление доступом на основе ролей (RBAC), это позволяет контролировать, какие данные могут быть доступны и кому внутри организации. Это также даёт отдельным командам разработки возможность работать с сетевыми потоками и создавать политики специально для своих пространств имён, не имея доступа к данным других пространств имён. Такая возможность позволяет командам независимо управлять безопасностью своих приложений, разрабатывая соответствующие политики и при этом сохраняя общий уровень безопасности и конфиденциальности в организации.

Hubble Timescape

В высоконагруженных средах иногда бывает сложно обеспечить безопасность приложения с помощью правильных сетевых политик Kubernetes, особенно при изменении шаблонов трафика или обнаружении неожиданных потоков. Функция Hubble Timescape решает эту проблему, позволяя отслеживать весь сетевой поток на протяжении жизненного цикла рабочих нагрузок и анализировать историю сетевого поведения и взаимодействий между сервисами.

На пути к обеспечению подхода Zero Trust эта функция бесцenna при доработке сетевых политик, поскольку позволяет наблюдать, как вело себя приложение до и после их применения. С Hubble Timescape можно уверенно тестировать, корректировать и применять политики для защиты приложения, сводя к минимуму риск сбоев или ошибочных конфигураций.

На изображении ниже показано приложение `microservices-demo` для которого загружен определённый временной интервал с соответствующими сетевыми потоками.

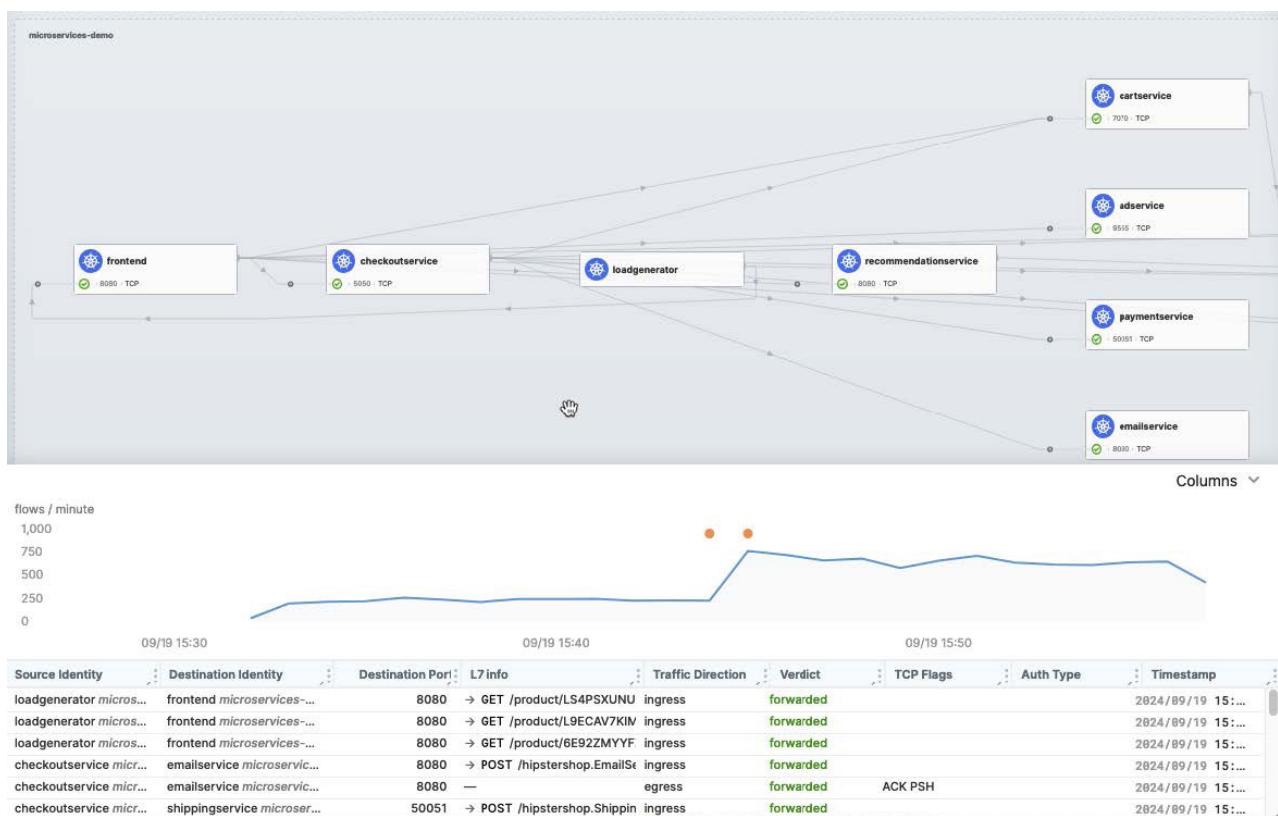


Рисунок 18: Hubble Timescape

Выше таблицы с потоками расположена линейный график. Он показывает количество зафиксированных информационных потоков за выбранный период и все изменения сетевых политик, применённых в кластере Kubernetes. Оранжевые точки на временной шкале отмечают эти события: создание, изменение или удаление сетевых политик.

При нажатии на оранжевую точку на графике времени отобразятся данные о зафиксированном изменении сетевой политики, а также появится возможность просмотреть эту политику в Редакторе сетевых политик Hubble (компонент Isovalent Enterprise).

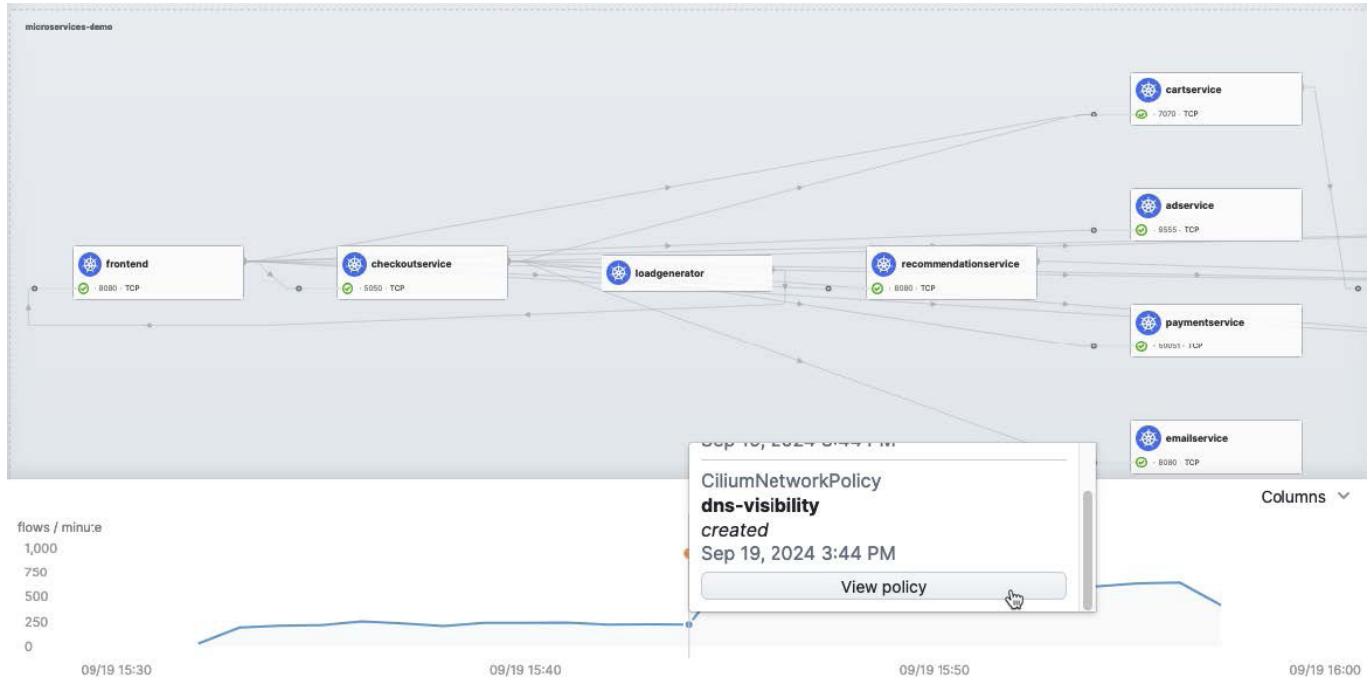


Рисунок 19. Hubble Timescape с просмотром сетевых политик (Network Policy Viewer)

Интерфейс Hubble UI фиксирует изменения сетевых политик и наглядно показывает различия между их версиями, что позволяет оператору при необходимости откатиться к предыдущему состоянию. На приведённом ниже скриншоте демонстрируются сравнение версий политики, используемой для мониторинга HTTP-трафика.

The screenshot shows the Network Policy Viewer comparing two versions of a CiliumNetworkPolicy. The left column lists line numbers (30, 31, 32, 33, 27, 28, 29, 30, 31, 42, 43, 44, 45, 46, 47, 48, 49) and the right column shows the corresponding policy code. A context menu is open at line 31, showing options: 'Restore current version' and 'updated Sep 19, 2024 4:42 PM'. The policy code includes sections for 'spec:', 'ingress', 'rules:', and 'status:'. A red box highlights the deletion of port 9200 in the 'ingress' section.

```

+ New ⚙️ + updated Sep 19, 2024 4:42 PM ⚙️ K8S Cilium
30 + ?:54Z
31 + Filter policy versions...
32 + Restore current version
33 + updated Sep 19, 2024 4:42 PM
27 34 created Sep 19, 2024 3:45 PM
28 35 spec:
29 36   endpointSelector: {}
30 37   ingress:
31 38     - fromEntities:
      Expand 10 lines ...
42 49       protocol: TCP
43 50     - port: "9200"
44 51       protocol: TCP
45 52     - port: "8000"
46 53       protocol: TCP
47 54     rules:
48 55       http:
49 56         - {}
50 57     enableDefaultDeny: {}
58   status: {}

```

Рисунок 20. Сравнение версий сетевых политик

Редактор сетевых политик

Редактор сетевых политик — это компонент, встроенный в Hubble как часть Isovalent Enterprise. Он позволяет не только визуализировать существующие сетевые политики, но и создавать новые, с нуля или на основе информационных потоков, наблюдаемых на вашей платформе.

Использование редактора сетевых политик гарантирует, что создаваемые политики соответствуют стандартам Cilium, это снижает трудозатраты, необходимые для быстрого внедрения сетевых стратегий сетевых, рассмотренных ранее в этом документе, на пути к модели Zero Trust.

Редактор сетевых политик поддерживает как стандартные сетевые политики Kubernetes, так и расширенные политики Cilium. Последние предоставляют дополнительные возможности для фильтрации на основе данных о трафике и приложениях, включая правила уровня 7 (Layer 7).

Ниже мы видим, что политика с именем “dns-visibility” загружена в редактор из кластера Kubernetes. Эта политика предназначена для разрешения захвата DNS-запросов, выполняемых приложениями, с целью повышения наблюдаемости.

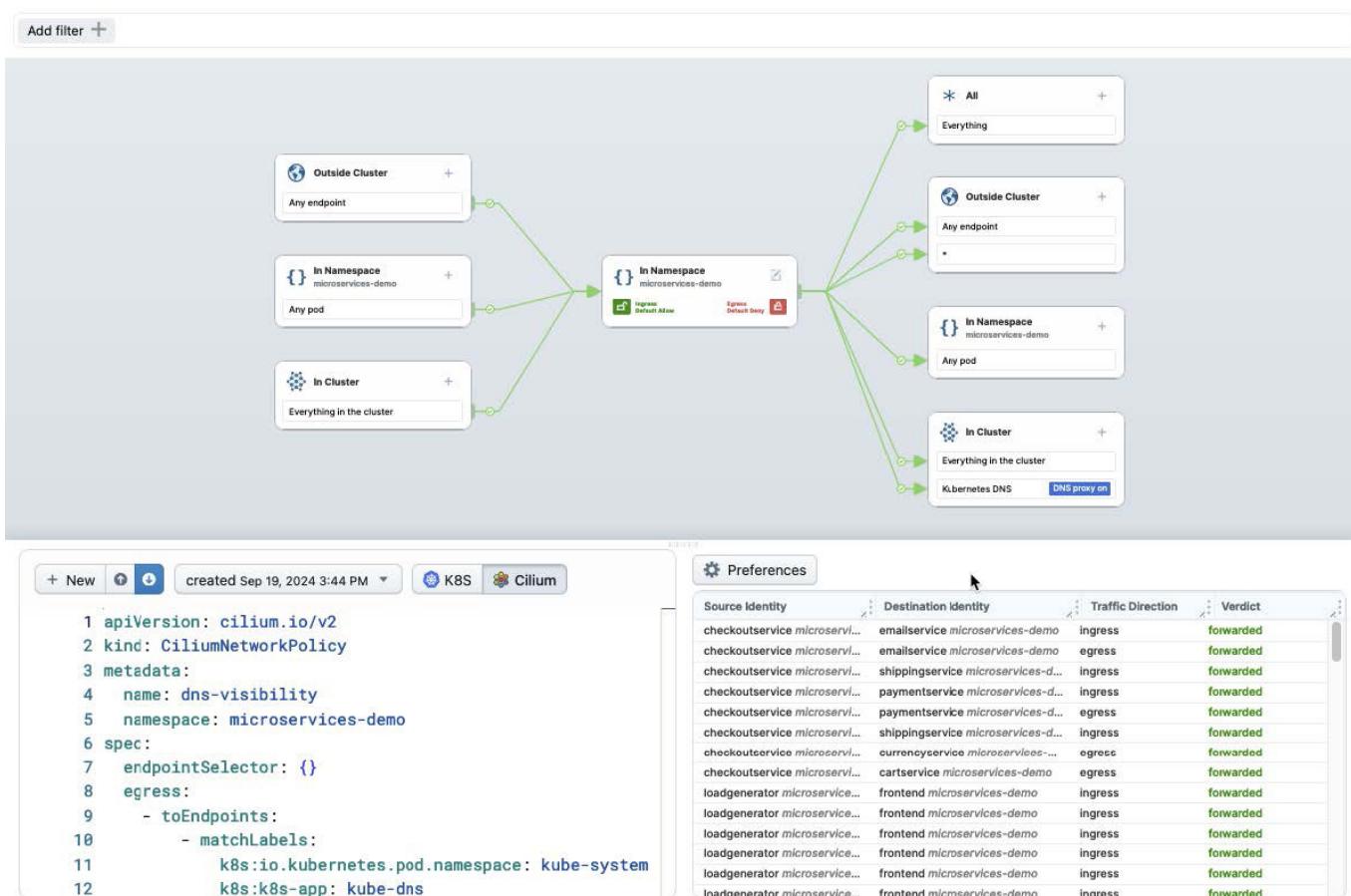


Рисунок 21. Редактор сетевых политик

Владельцы платформы могут легко отслеживать трафик, направленный на внешние FQDN, в таблице потоков. С её помощью можно выбрать определённый трафик для добавления во вновь создаваемые политики и применить их в Cilium.

Следуя указанным подходам, первой сетевой политикой, которую необходимо создать для защиты приложений в пространстве имён `microservices-demo`, будет политика, разрешающая весь трафик внутри этого пространства имён.

Это выполняется путем настройки доступа “от/к любым подам с помощью селекторов “In Namespaces” на карте сервисов. После применения остальные селекторные поля автоматически отобразятся как красные маршруты трафика, указывая на то, что трафик заблокирован — например, из пространства имён к любым конечным точкам за пределами кластера (верхнее правое поле).

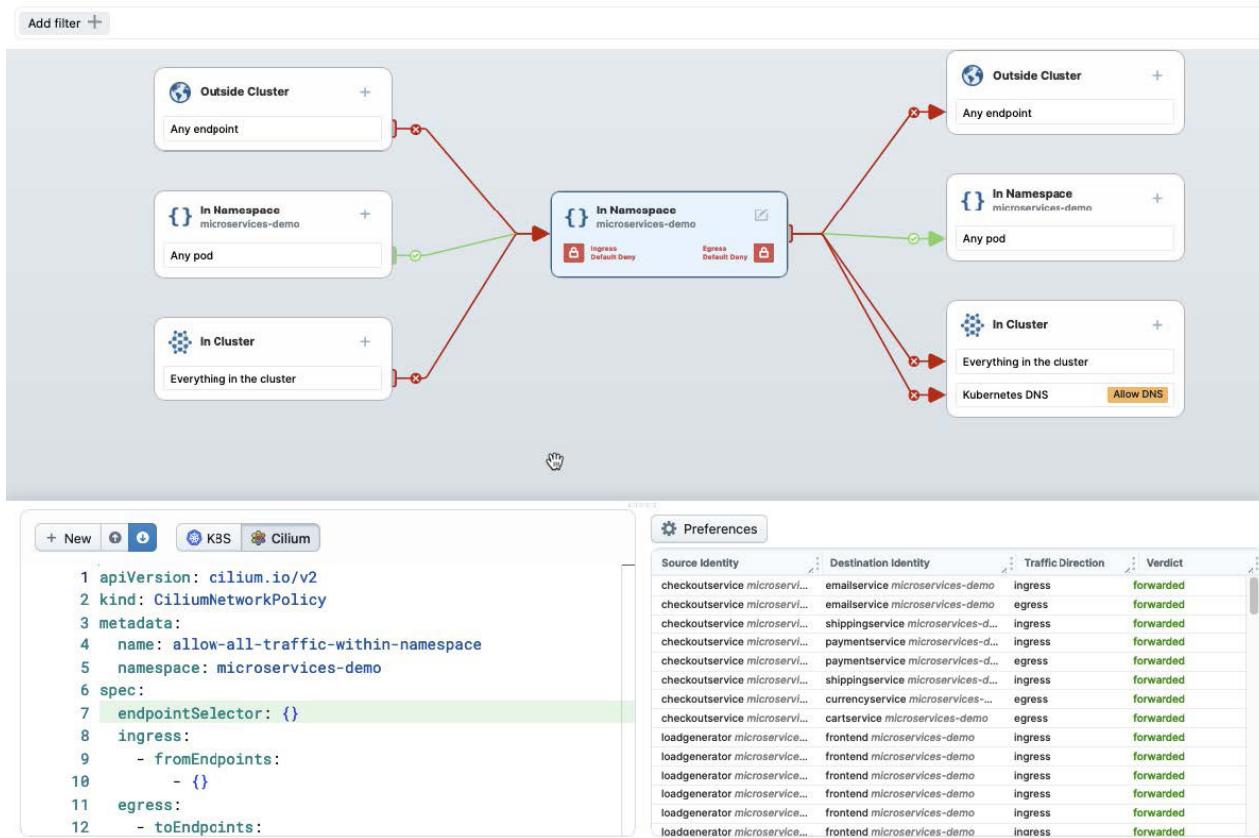


Рисунок 22. Создание нового правила с помощью Редактора сетевых политик

Ниже приведён вывод YAML из Редактора сетевых политик.

```

apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: allow-all-traffic-within-namespace
  namespace: microservices-demo
spec:
  endpointSelector: {}
  ingress:
    - fromEndpoints:
        - {}
  egress:
    - toEndpoints:
        - {}

```

На основе этой конфигурации можно построить набор правил для конкретных сервисов внутри пространства имён. После настройки всех необходимых правил исходное широкое правило, разрешающее всё взаимодействие внутри пространства имён, можно будет удалить.

На скриншоте ниже создана новая политика, которая будет применяться ко всем потокам трафика к сервису `checkoutservice` и от него. Сервис идентифицируется по метке Kubernetes, назначеннной данному deployment; весь остальной трафик запрещён, это отображено красными линиями "denied" на карте сервисов.

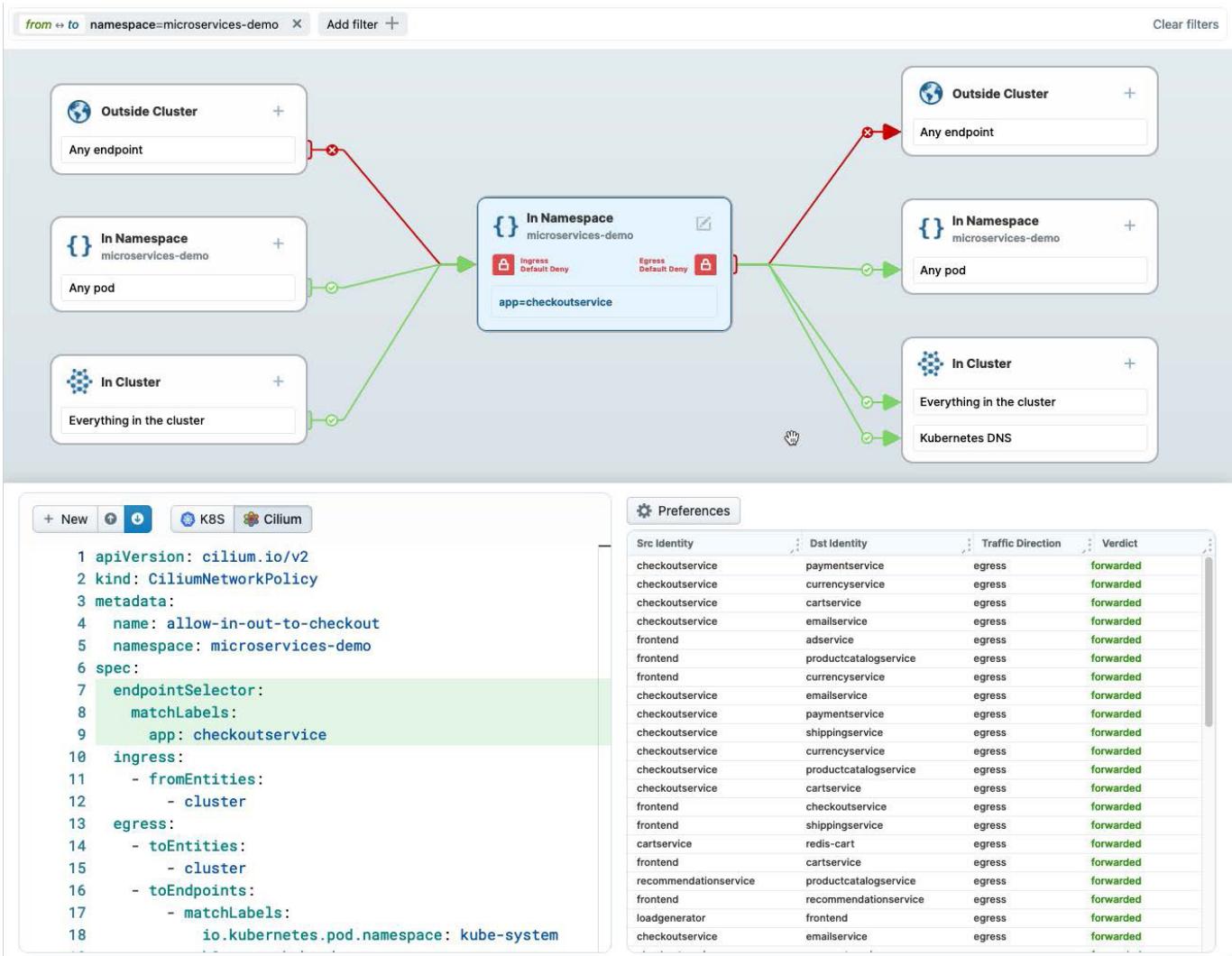


Рисунок 23. Создание правила с использованием endpointSelector

Для настройки этого правила в селекторе конечных точек (endpointSelector) используется метка приложения в Kubernetes `app=checkoutservice`. В сервисной карте для селекторов "In Cluster" были выбраны опции "Everything in the cluster" и "Everything in the cluster" + "Kubernetes DNS", эти правила разрешают взаимодействие со всеми ресурсами внутри, а также явно разрешают доступ к сервису DNS кластера, это критически важно для работы большинства приложений.

Эти действия генерируют приведённый ниже вывод в формате YAML. Используя политики, которые применяются к определённым `endpointSelectors` в пространстве имен, можно формировать в своей среде всё более детализированные правила.

```
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: allow-in-out-to-checkout
  namespace: microservices-demo
spec:
  endpointSelector:
    matchLabels:
      app: checkoutservice
  ingress:
    - fromEntities:
        - cluster
  egress:
    - toEntities:
        - cluster
    - toEndpoints:
        - matchLabels:
            io.kubernetes.pod.namespace: kube-system
            k8s-app: kube-dns
        toPorts:
          - ports:
              - port: "53"
                protocol: UDP
            rules:
              dns:
                - matchPattern: "*"
```

Правила можно создавать напрямую из таблицы сетевых потоков Hubble. Для этого достаточно выбрать любой зафиксированный поток и добавить его в новую сетевую политику (или удалить из неё). При выборе потока для создания правила система автоматически подставит метки Kubernetes (Labels) соответствующих рабочих нагрузок (workloads) в качестве селекторов `matchLabels` в секцию `toEndpoints` в зависимости от направления трафика (входящий Ingress или исходящий Egress).

На скриншоте ниже показано правило, созданное на основе сетевого потока. Оно разрешает входящий трафик (ingress) к `checkoutservice` идентифицируемому по его метке в Kubernetes. Конфигурация, полученная из этого сетевого потока для генерации правила, может быть удалена.

С помощью Редактора сетевых политик можно использовать несколько потоков для построения набора правил внутри политики Cilium.

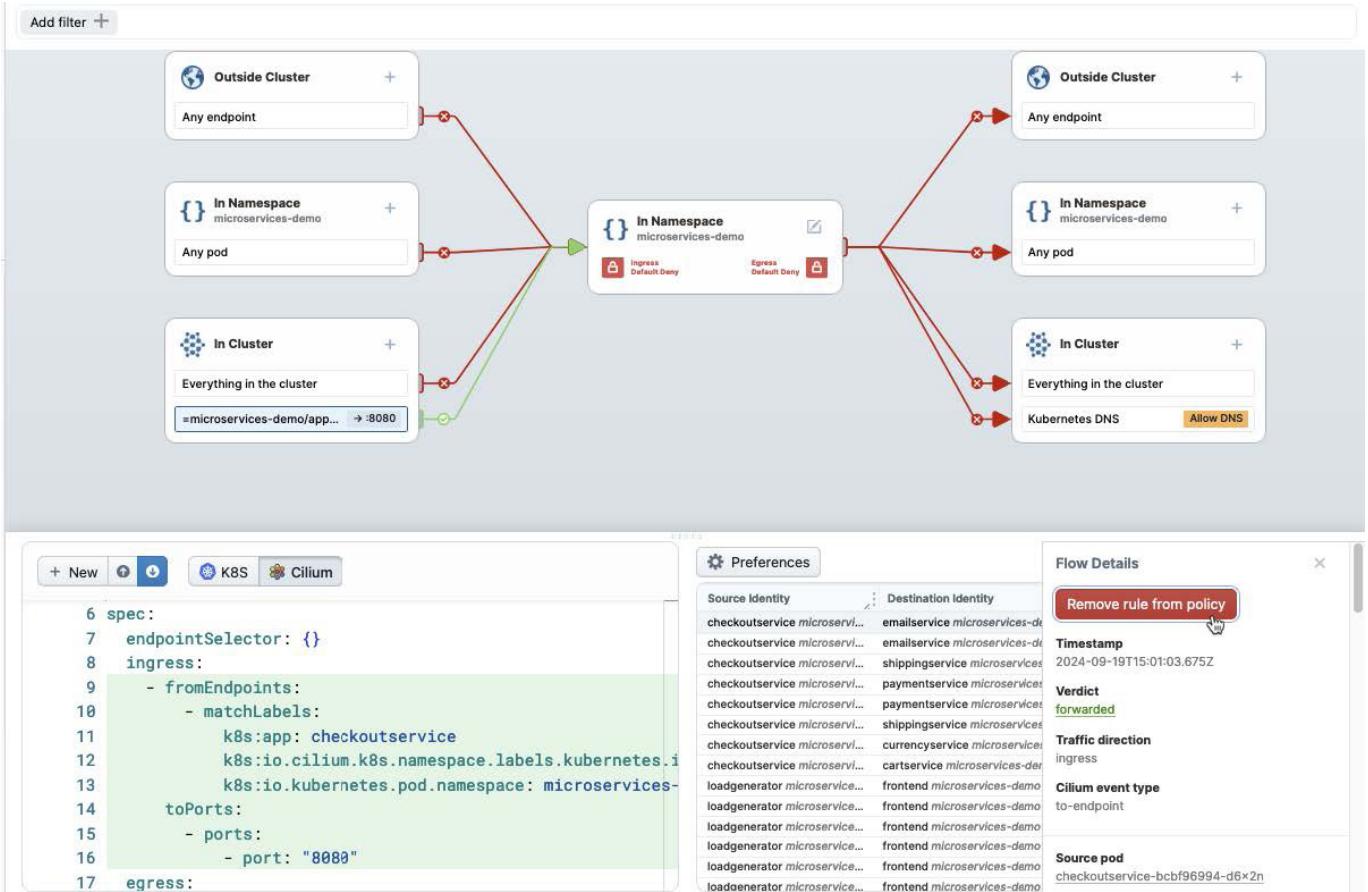


Рисунок 24. Удаление правила с помощью редактора сетевой политики

Метрики Hubble

Cilium и Hubble предоставляют метрики в формате OpenMetrics. Метрики могут собираться и визуализироваться при помощи широкого спектра инструментов. Частно используются Prometheus для сбора и Grafana для визуализации, но существует множество других вариантов. Можно сочетать Prometheus с альтернативными решениями для визуализации, использовать Grafana с различными системами хранения или интеграция со сторонними платформами мониторинга.

Метрики Cilium и Hubble можно использовать и включать по отдельности. Это позволяет гибко настраивать мониторинг и отслеживать только те данные о сети и безопасности, которые действительно необходимы. Такой подход даёт полную наблюдаемость, точно соответствующую задачам. Метрики Hubble обеспечивают видимость всех потоков, обрабатываемых Cilium, включая данные о сбросе пакетов (drops), где и почему трафик блокируется, результатах применения политик (policy verdicts), какие правила сетевых политик сработали, а также протоколы уровня 4 (L4) и уровня 7 (L7).

На приведённом ниже дашборде Grafana используется метрика

`hubble_policy_verdicts_total` вместе с метаданными Kubernetes, захваченными в потоках Hubble, чтобы обеспечить более лучшую видимость сетевых политик, применённых в кластере. На скриншоте первый раздел, Ingress Policy Verdict Rate per Minute by Match Type, показывает, что из применённых сетевых политик большая часть трафика обрабатывается на уровне L7. В правом нижнем блоке видно, что около 50% потоков перенаправляются; это означает, что действуют политики, которые направляют трафик через прокси Cilium. В данном демонстрационном примере это используется для обеспечения видимости DNS, о которой говорилось ранее.

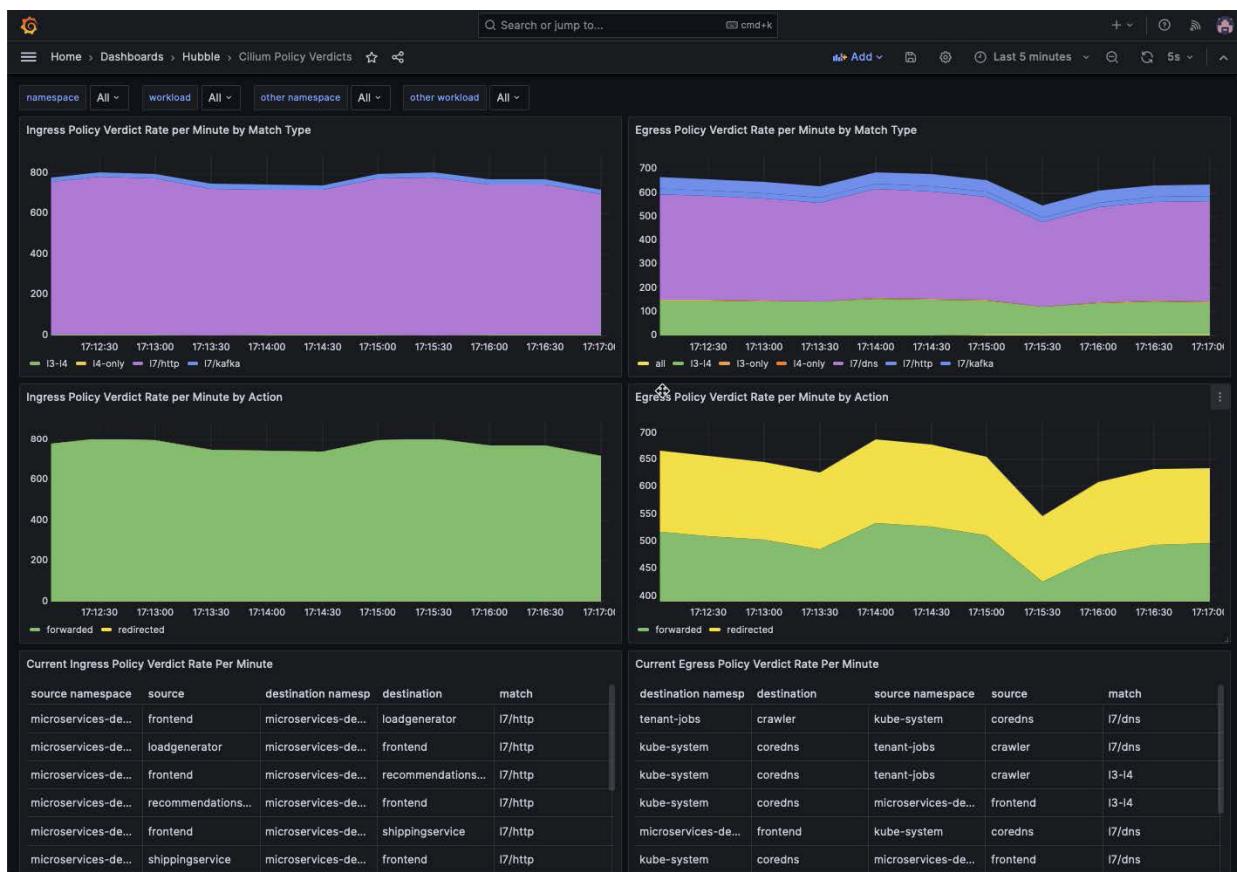


Рисунок 25. Метрики результатов применения политик Hubble

Используя метрики, предоставляемые Hubble, можно собирать и визуализировать данные уровня приложения (Layer 7) для мониторинга и диагностики. Объём запросов, процент успешных выполнений и длительность запросов — вот лишь некоторые из доступных пользователям метрик. Они предоставляются через Hubble Metrics как часть механизма политик, который используется для выбора потоков данных и их перенаправления через прокси Cilium, обеспечивая тем самым дополнительную наблюдаемость.

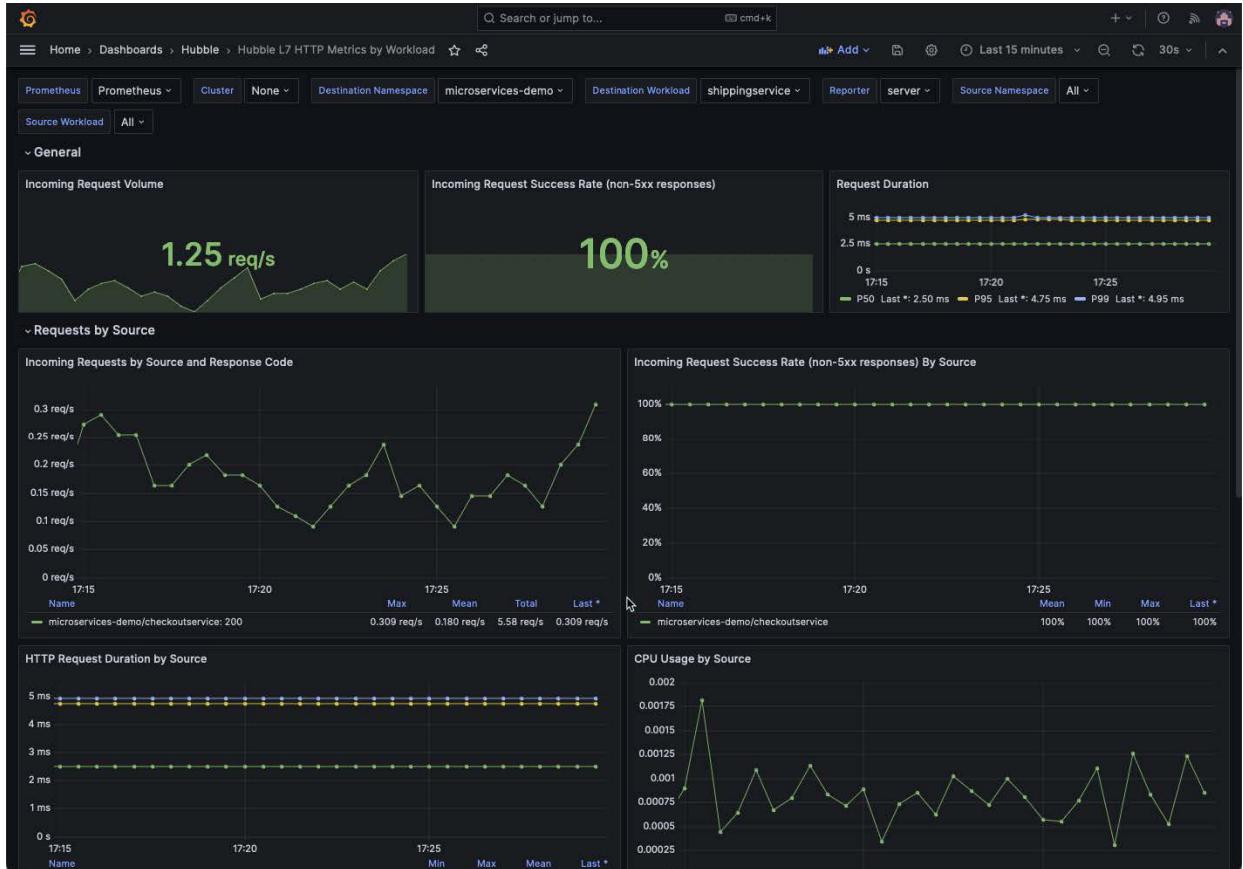


Рисунок 26: Метрики HTTP уровня 7 (L7) в Hubble

Дашборды Isovalent Enterprise для Cilium и Hubble

Iovalent сосредоточен на том, чтобы снять операционную нагрузку с эксплуатации облачных платформ в масштабе. Это возможно благодаря нашему опыту как создателей Cilium, готовым корпоративным решениям, а также выдающейся поддержке и сопровождению клиентов.

Клиенты Isovalent Enterprise получают доступ к набору готовых корпоративных дашбордов, специально разработанных для операторов платформы. Эти панели мониторинга фокусируются на работоспособности и эффективности работы Cilium в вашей инфраструктуре.

Используя дашборд «Высокоуровневое состояние» (High-Level Health), можно оценить общую работоспособность Cilium. На приведённом ниже скриншоте внимание сфокусировано на разделе «Применение политик» (Policy Enforcement), который показывает ключевые показатели, отслеживающие сигналы в механизме политик Cilium.



Рисунок 27. Дашборд "Высокоуровневое состояние" (High-Level Health Dashboard)

Заключение

Внедрение модели Zero Trust критически важно для крупных организаций, стремящихся минимизировать риски безопасности и обеспечить соответствие требованиям в сложных средах. Неэффективное управление сетевой безопасностью может привести к пробелам в политиках, слепым зонам и повышенной уязвимости к атакам. Используя платформу Isovalent Enterprise с её компонентами Cilium и Tetragon, предприятия могут применять единые политики безопасности во всех кластерах Kubernetes, эффективно снижая риски и сохраняя операционную эффективность. Такой подход не только усиливает защиту, но и даёт командам разработки возможность внедрять инновации без ущерба для соответствия стандартам или производительности.

В Isovalent мы сопровождаем клиентов на этом пути с помощью наших корпоративных решений для Cilium и Tetragon. Наша команда архитекторов решений и инженеров по надёжности (Customer Reliability Engineers), экспертов в области облачных технологий Kubernetes и сетей, предоставляет индивидуальную поддержку, усиление функций и тестовые среды для клиентов. Как создатели Cilium, ведущего CNI для Kubernetes, экспертиза Isovalent гарантирует, что клиенты могут снижать риски, повышать безопасность и ускорять развёртывание сетевых политик.

С помощью расширенных возможностей Cilium для сетевой наблюдаемости и обеспечения безопасности, а также комплексного мониторинга и контроля Tetragon в реальном времени, организации могут уверенно внедрять архитектуру Zero Trust, которая адаптируется к динамичной среде современных рабочих нагрузок. Платформа Isovalent предлагает необходимую гибкость и мощь для защиты вашей инфраструктуры, будь то контейнерные среды, виртуальные машины или гибридные системы. От создания детализированных сетевых политик до мониторинга и их совершенствования в реальном времени — Isovalent гарантирует, что ваша среда Kubernetes остаётся безопасной, масштабируемой и готовой поддерживать инновации на любом уровне.



Приложение

¹<https://veducate.co.uk/>

²<https://docs.cilium.io/en/stable/security/policy-creation/>

³<https://isovalent.com/blog/post/achieving-pci-dss-compliance-with-isovalent-cilium-and-zero-trust/>

⁴<https://isovalent.com/blog/post/isovalent-enterprise-1-13/>

⁵<https://docs.cilium.io/en/stable/network/kubernetes/policy/#ciliumclusterwidernetworpolicy>

⁶<https://github.com/GoogleCloudPlatform/microservices-demo>

⁷<https://github.com/GoogleCloudPlatform/microservices-demo/tree/main/src/frontend>

⁸<https://github.com/GoogleCloudPlatform/microservices-demo/tree/main/src/cartservice>

⁹<https://github.com/GoogleCloudPlatform/microservices-demo/tree/main/src/productcatalogservice>

¹⁰<https://github.com/GoogleCloudPlatform/microservices-demo/tree/main/src/currencyervice>

¹¹<https://github.com/GoogleCloudPlatform/microservices-demo/tree/main/src/paymentservice>

¹²<https://github.com/GoogleCloudPlatform/microservices-demo/tree/main/src/shippingservice>

¹³<https://github.com/GoogleCloudPlatform/microservices-demo/tree/main/src/emailservice>

¹⁴<https://github.com/GoogleCloudPlatform/microservices-demo/tree/main/src/checkoutservice>

¹⁵<https://github.com/GoogleCloudPlatform/microservices-demo/tree/main/src/recommendationservice>

¹⁶<https://github.com/GoogleCloudPlatform/microservices-demo/tree/main/src/adservice>

¹⁷<https://github.com/GoogleCloudPlatform/microservices-demo/tree/main/src/loadgenerator>

eBPF-based unified cloud-native solutions