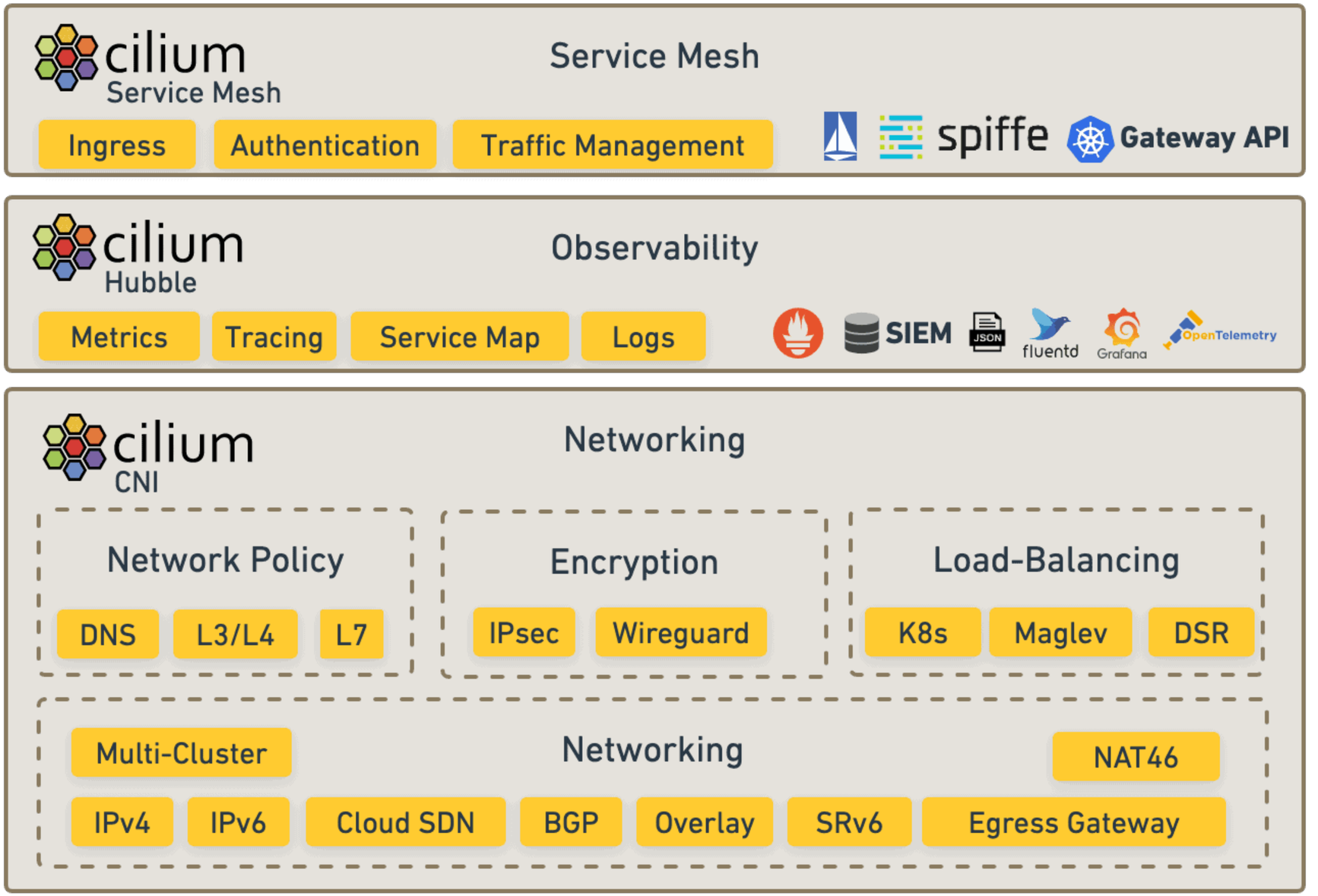
Cilium: сетевые политики уровня приложений

Кожемякин Александр, Lead SRE

Содержание

Cilium


cilium
Service Mesh
Service Mesh
Ingress
Authentication
Traffic Management
spiffe
Gateway API
cilium
Hubble
Observability
Metrics
Tracing
Service Map
Logs
SIEM
JSON
fluentd
Grafana
OpenTelemetry
cilium
CNI
Networking
Network Policy
DNS
L3/L4
L7
Encryption
IPsec
Wireguard
Load-Balancing
K8s
Maglev
DSR
Networking
Multi-Cluster
NAT46
IPv4
IPv6
Cloud SDN
BGP
Overlay
SRv6
Egress Gateway

Cilium для сетевой безопасности

Cilium — платформа для обеспечения сетевой безопасности, использующая eBPF для высокопроизводительного контроля сети:

- Поддержка L3/L4 и L7 политик.
- Возможность инспекции и фильтрации трафика на уровне приложений (L7).
- Полноценная поддержка Service и Cluster Mesh.

Основные преимущества:

- Высокая производительность благодаря eBPF.
- Расширяемость и гибкость политик.
- Совместимость с облачными и локальными средами.

Большой минус:

- Высокий порог входа (благодаря eBPF)

Понимание L7 политик

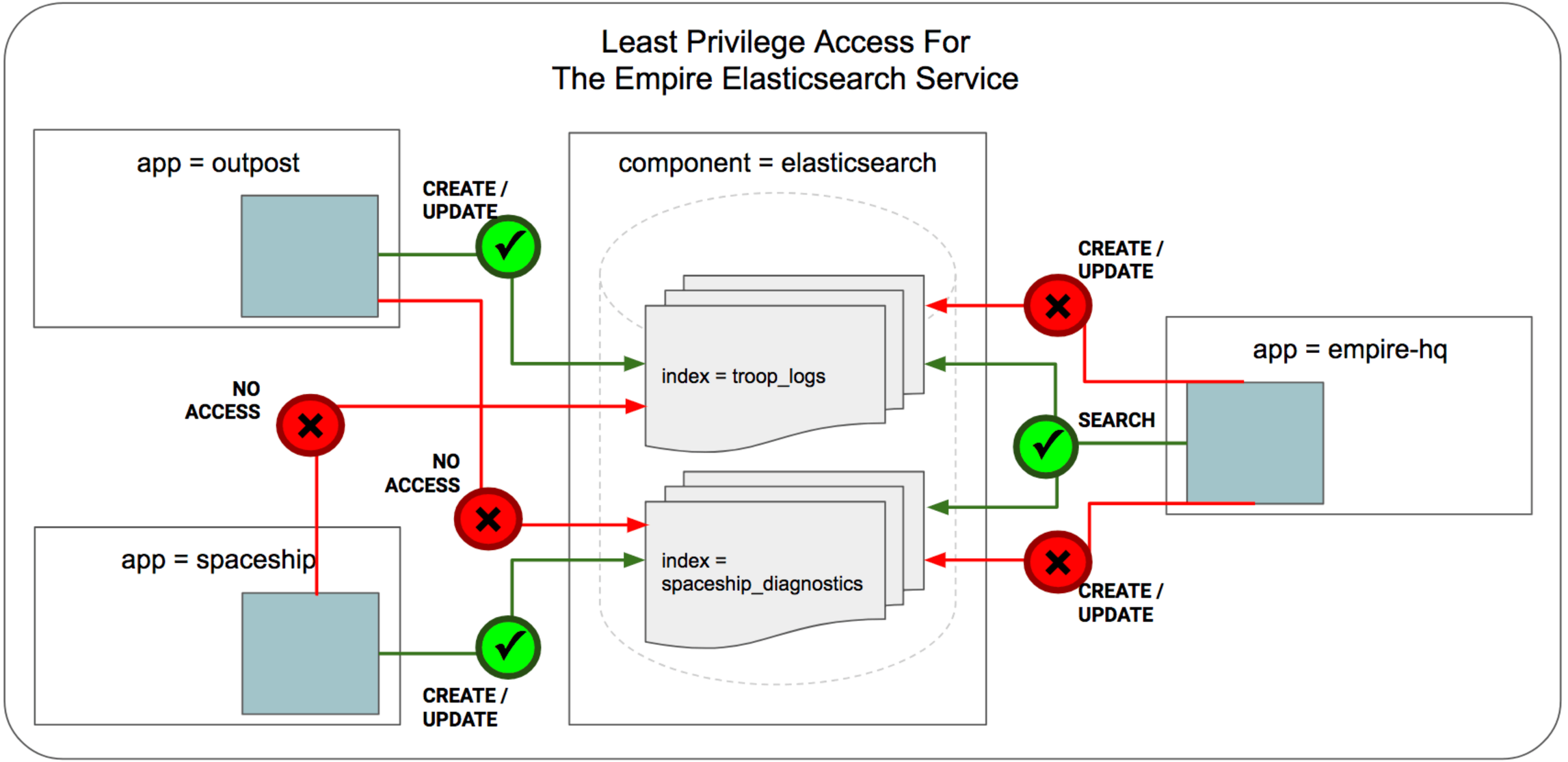
1) L7 политики работают на уровне приложений, контролируя трафик на основе содержимого данных.

2) В Cilium L7 политики основаны на eBPF, обеспечивая высокую производительность и глубокую интеграцию с ядром ОС.

3) Расширение возможностей:
- Управление HTTP-запросами: фильтрация URI, заголовков и методов.
- Контроль доступа к специфическим API-эндоинтам в gRPC.
- Управление командами и доступом к topic в Kafka, разграничение в Elastic.

4) Основные преимущества:
- Повышенная безопасность за счет детального контроля трафика.
- Гибкость в управлении доступом на уровне приложений.
- Уменьшение атак на уровне приложений, таких как SQL-инъекции или XSS.

Схематически выглядит так



Микросервисы и безопасность

1) L7 политики работают на уровне приложений, контролируя трафик на основе содержимого данных.

2) В Cilium L7 политики основаны на eBPF, обеспечивая высокую производительность и глубокую интеграцию с ядром ОС.

3) Расширение возможностей:
- Управление HTTP-запросами: фильтрация URI, заголовков и методов.
- Контроль доступа к специфическим API-эндоинтам в gRPC.
- Управление командами и доступом к topic в Kafka, разграничение в Elastic.

4) Основные преимущества:
- Повышенная безопасность за счет детального контроля трафика.
- Гибкость в управлении доступом на уровне приложений.
- Уменьшение атак на уровне приложений, таких как SQL-инъекции или XSS.

Kafka, особенности безопасности

Как устроено

Topic - как основная единица хранения всех сообщений очереди.

Broker - управляет сообщениями и репликацией

Consumer, Producer - основные потребители сообщений

Все вместе позволяют построить быструю и надежную передачу сообщений между микросервисами.

Как защитили

Шифрование - TLS

Аутентификация - SASL

Авторизация доступов - ACL

Типичные угрозы:
- mitm
- несанкционированная запись/чтение

GRPC

Как устроено

Работает поверх HTTP/2, обеспечивая двухстороннюю потоковую передачу и механизм приоритизации.

Поддерживает различные методы RPC:

- обычные
- серверные
- клиентские
- двухсторонние потоковые передачи.

Такая архитектура способствует высокой производительности и низкой latency.

Как защитили

Шифрование - TLS

Аутентификация - mTLS, OAuth2, JWT, etc.

Авторизация - простая интеграция со сторонними системами управления правами доступа

Типичные угрозы:

- mitm
- Несанкционированный доступ к сервисам
- Неправильное управление сертификатами и ключами

А теперь сделаем безопасно

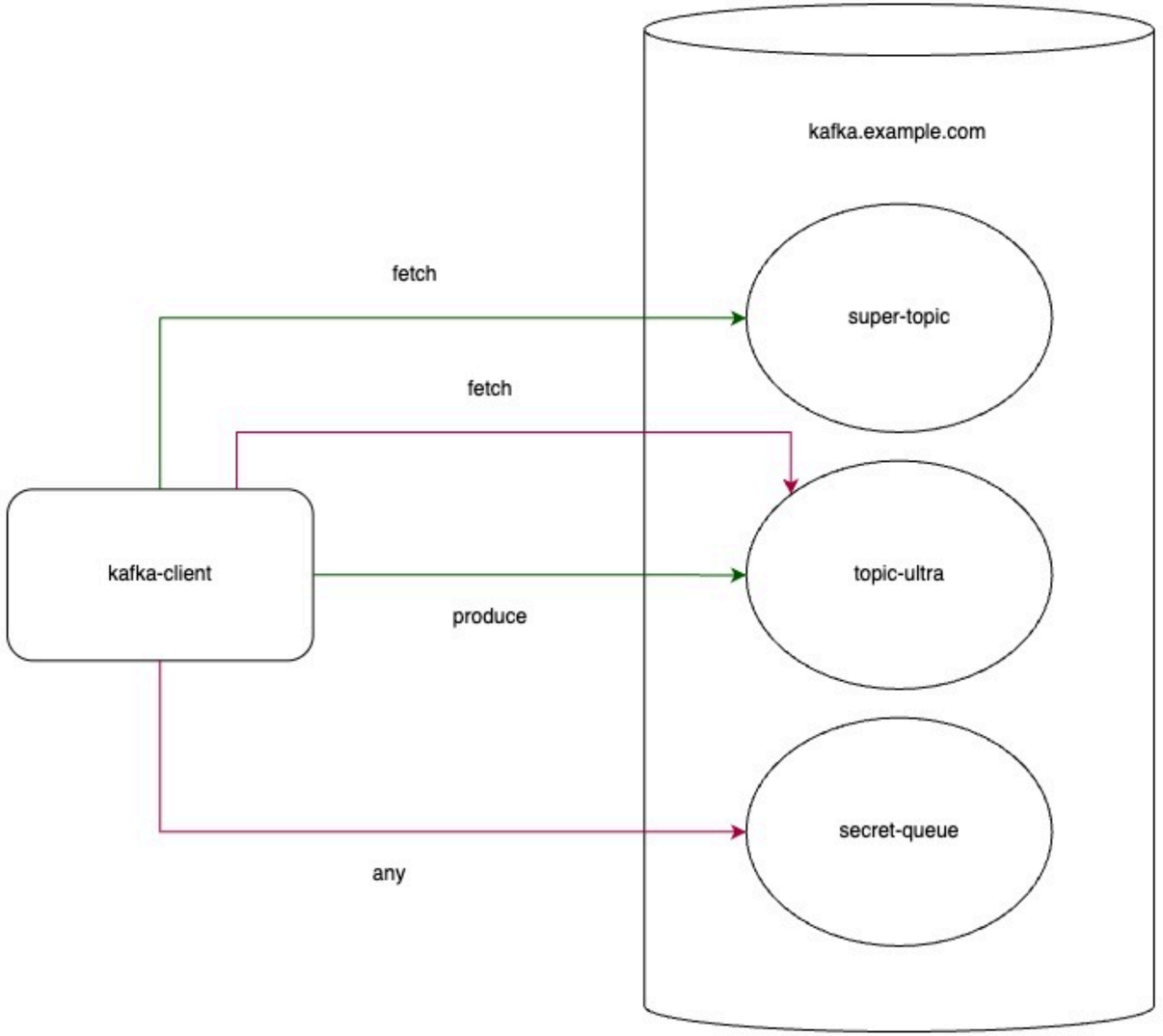
Напишем простую политику

```
spec:
  endpointSelector:
    matchLabels:
      app: kafka-client #Лейблы приложения, которому разрешаем доступ
  ingress:
  - fromEndpoints:
    - matchLabels:
        app: kafka-broker #Лейблы нашего Kafka broker
      ……
      rules:
        kafka:
        - apiKey: "fetch" #Разрешаем команды fetch
          topic: “super-topic" #Указываем конкретную тему, доступную для чтения
        - apiKey: "produce" #Разрешаем команды produce
          topic: "topic-ultra" #Указываем, в какую тему можно записывать
```

И теперь для внешней Kafka

```
spec:
  endpointSelector:
    matchLabels:
      app: kafka-client #Лейблы приложения, которому разрешаем доступ
  egress:
  - toFQDNs:
    - matchName: “kafka.example.com” #Заменяем Kafka на внешнюю
      ……
      rules:
        kafka:
           #Правила, как и на предыдущем слайде
  labels:
    - key: "purpose"
      value: “external-kafka-access”
      #Использование подобных лейблов не обязательно, но сильно упрощает жизнь в рамках
      больших инфраструктур
```

Что получилось



Еще больше безопасности

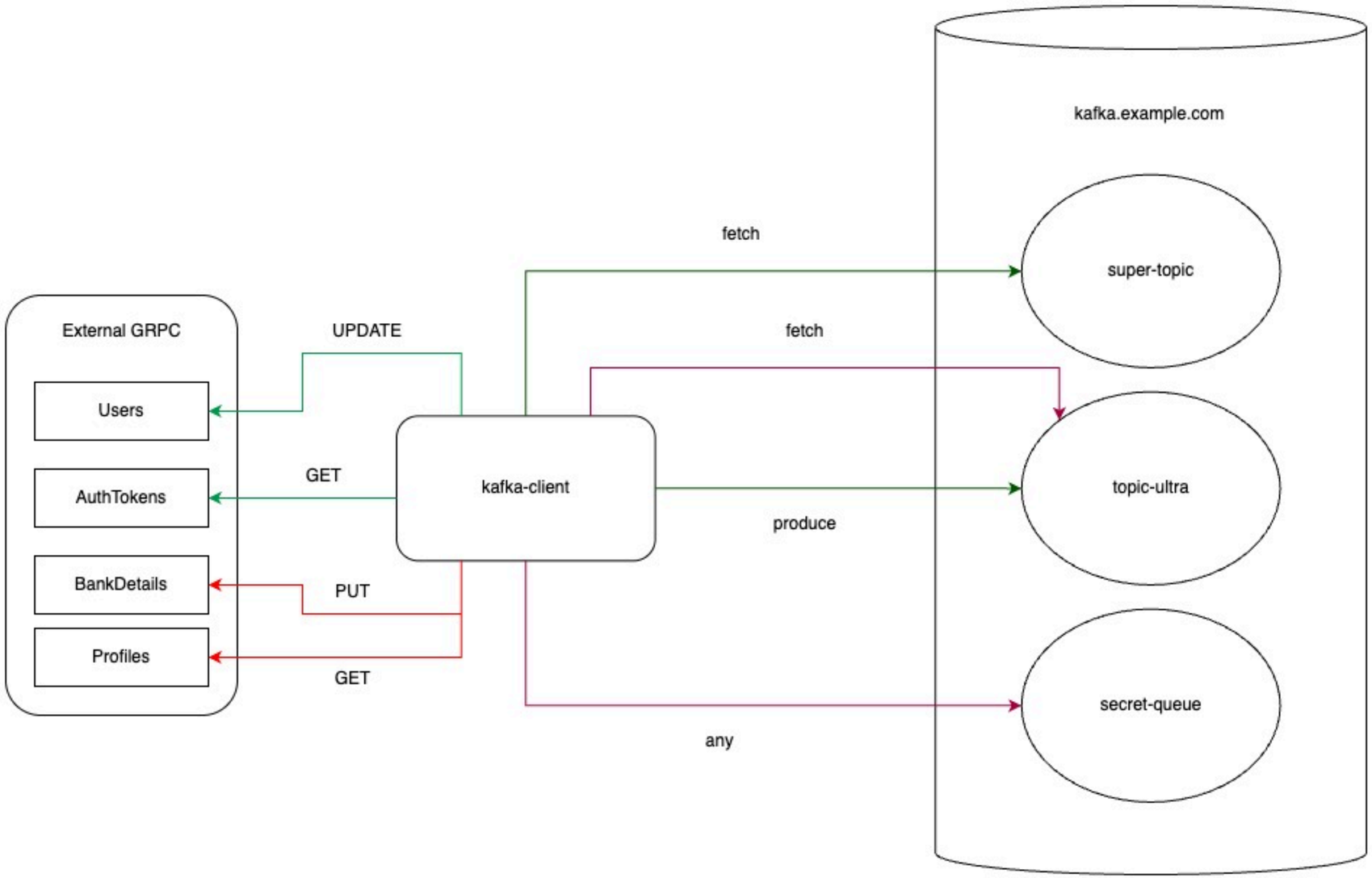
Политики для grpc

```
spec:
    endpointSelector:
        matchLabels:
          app: kafka-client #Лейбл приложения, которому разрешен доступ к gRPC серверу
    egress:
      - toEndpoints:
        - matchLabels:
            app: grpc-server #Определяет gRPC серверы
        toPorts:
            - ports:
              - port: "50051" #Порт, который использует ваш gRPC сервер
        protocol: "TCP"
      rules:
        http:
        - method: "GET"
          path: "/com.example.GrpcApp/AuthTokens" #Разрешенный метод gRPC
        - method: "UPDATE"
          path: "/com.example.GrpcApp/Users" #Другой разрешенный метод gRPC
```

Можно и для внешнего мира

```
spec:
  endpointSelector:
    matchLabels:
      app: kafka-client #Опять наше приложение
  egress:
  - toFQDNs:
    - matchName: "external-grpc.example.com" #DNS-имя внешнего gRPC сервиса
  ……….
  rules:
        http:
           #Правила, как и на предыдущем слайде
  labels:
    - key: "purpose"
      value: “external-grpc-server”
      #Использование подобных лейблов не обязательно, но сильно упрощает жизнь в рамках
      больших инфраструктур
```

А теперь все вместе



Зачем заморачиваться

Единое средство контроля:
- Точка правды для политик - репозиторий
- Любое изменение проходит ревью
- Данные о сетевом взаимодействии выгружаются в SIEM

Повышает понимание окружения:
- Нужно изучить один инструмент
- Повышает контроль за разнородной инфраструктурой
- Помогает строить схему сетевых взаимодействий

НО:
- Не отменяет других средств контроля
- Придется договариваться с множеством команд
- На продуктовых контурах не получится “влететь с ноги”

Делать или нет

- Попробовать однозначно стоит

- Даже если не выкатите в прод, гораздо лучше узнаете сетевое окружение

- А если выкатите в рабочие среды, получите много интересного опыта

- Как бонус, структурируете хранение правил в удобный вид

Давайте делать сети безопаснее вместе

Кожемякин Александр,
Lead SRE