7 раз отмерь, а потом переделай

как мы храним сетевые политики

Александр Кожемякин

Обо мне

Александр Кожемякин

SRE, VK

О чем эта презентация

1. На какие грабли мы наступали при хранении сетевых политик
2. Как боролись со сложностями, зачем переделывали
3. Как эволюционировал наш подход к хранению политик
4. Что получилось в итоге: плюсы и минусы

Почему появилась презентация

Довольно быстро пришли к необходимости Deny All

С Cilium проделывали трюк в первый раз

Узнали много интересного о работе политик,

Выяснили много новых моментов о собственных

Кому будет полезна

- Если вы только планируете использовать сетевые политики – поможет разобраться в организации и не столкнуться с нашими проблемами
- Если у вас много политик – откроете для себя новые моменты

Как у нас устроена сеть

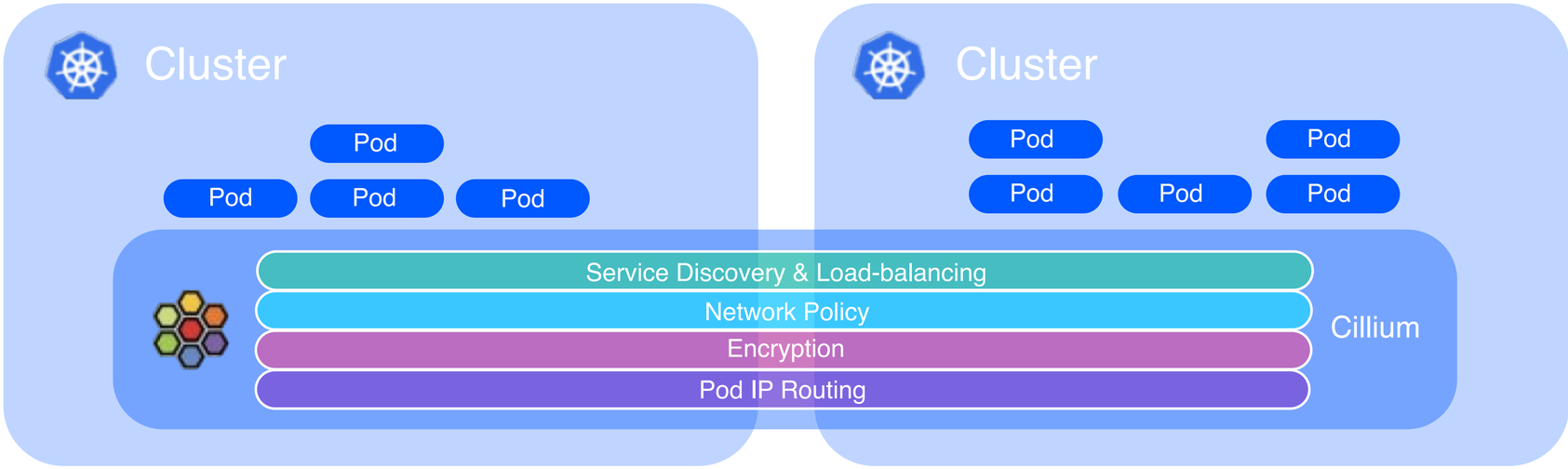
- Плоская l3 сеть везде (реальные адреса у подов)
- Cilium 1.12
- K8S 1.23
- Глобальные доступы выдаются на фаерволах, правила для приложений – в кластерах

Мультикластерная сеть в cilium

Cluster mesh в cilium –
соединение кластеров
в единую сеть на уровне cni

Каждый кластер работает
в своем сетевом пространстве,
но при этом связан на сетевом
уровне с остальными
кластерами в меше

Наши кластера связаны
с помощью cilium
на сетевом уровне



Зачем нам политики

Управление сетевыми доступами

- Гранулярность
- Время открытия
- Гибкость

Удобно отслеживать и быстро менять

- Единая точка правды – репозиторий
- Убедиться, что в кластере все как надо – запуск пайплайна одной кнопкой

Какие политики мы используем

- L3 и L7 политики
- Понятные правила, много фишек с лейблами
- Все для уровня приложений

- Сложно написать хорошую и лаконичную политику
- Не всегда получается переиспользовать – приходится переписывать

Шаблонизация

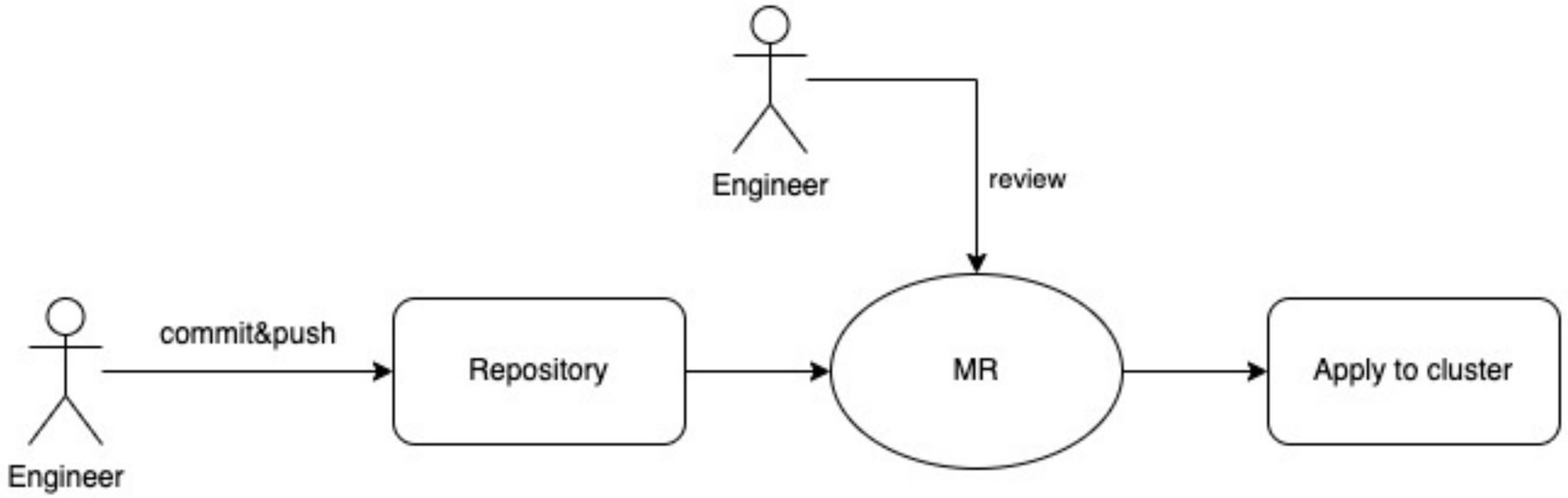
- Все храним в одном репозитории
- Используем helm
- Привыкли к буквам вместо ip

```
- toCIDRSet:
  {{- range .Values.global.networkAddresses.masterAddrs }}
    - cidr: {{ . }}/32
  {{- end }}
  {{- range .Values.global.networkAddresses.masterVips }}
    - cidr: {{ . }}/32
  {{- end }}
```

И запихнем это в кластер

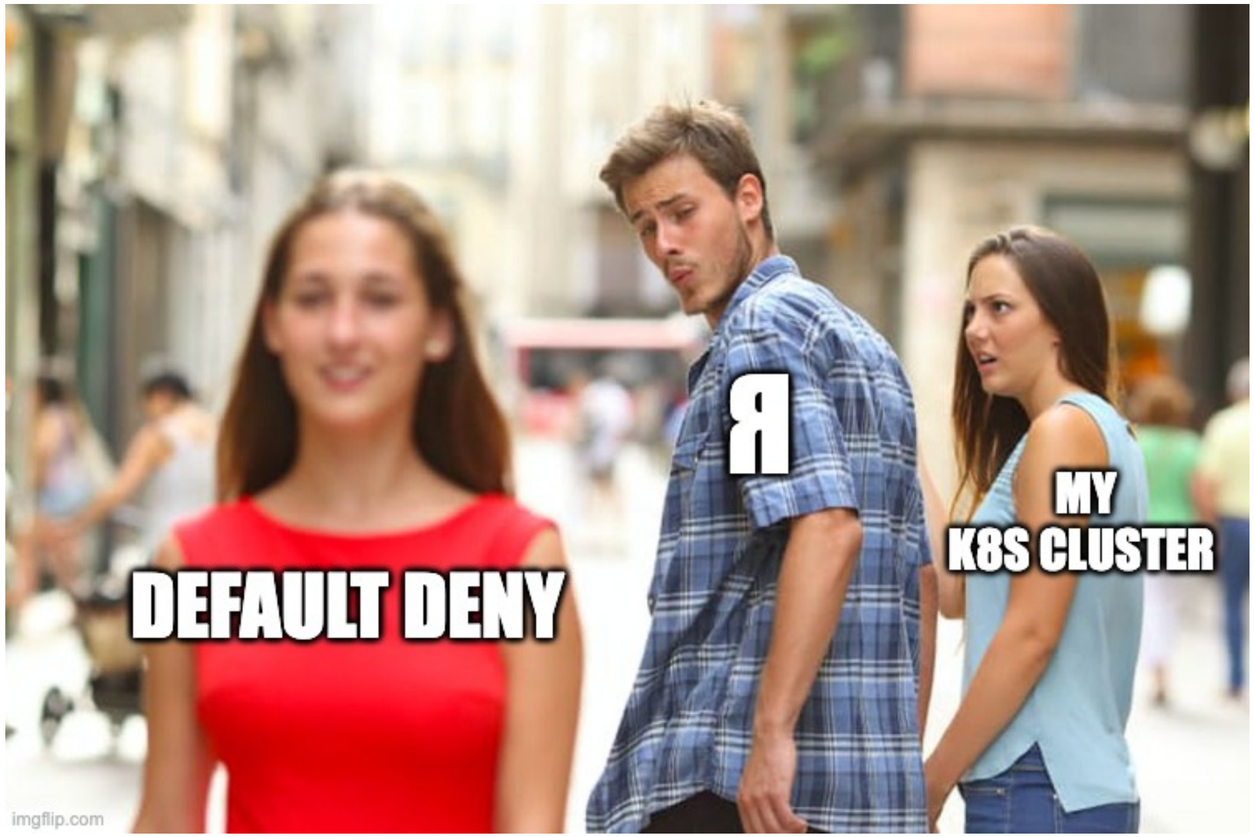
- Пайплайн для внесения изменений на Gitlab-ci
- Можем точечно выбрать кластер для изменений
- Проверка соответствия мастеру одной кнопкой
- Быстро убираем расхождения

Изменения в кластерах



Но после default deny

- У нас получилось, все круто
- Стали добавлять политики для каждого приложения
- Инженеры прокачали свой скил



Красивые схемки, а в репе что?

- На предыдущих слайдах все красиво и понятно
- Так ли было всегда? Да
- А в репозитории – нет

Одно приложение - один файл

- Именно таким подходом мы пользовались вначале

- Это очень удобно когда кластер свежий и в нем мало приложений

- Удобно, если вы только начинаете пользоваться политиками

- Нет необходимости договариваться про именование, расположение и т.д.

- Понятно где искать

- Нет путаницы с файлами

- у вас больше одной политики

Больше политик богу политик

- Берем приложение, собираем политику, кладем в файл

- Делаем вторую политику, кладем в тот же файл


```yaml
---
apiVersion: cilium.io/v2
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: "icmpv6.egress.to.any"
spec:
  endpointSelector: {}
  egress:
  - icmps:
    - fields:
      - family: IPv6
        type: 135
      - family: IPv6
        type: 136

---
apiVersion: cilium.io/v2
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: "icmpv6.ingress.from.any"
spec:
  endpointSelector: {}
  ingress:
  - icmps:
    - fields:
      - family: IPv6
        type: 135
      - family: IPv6
        type: 136
```

И делаем еще политики

– Силиуму не хватит политик только для icmp в ipv6

– Делаем еще политику

- Кладем в тот же файл

Не забыть сделать wc -l

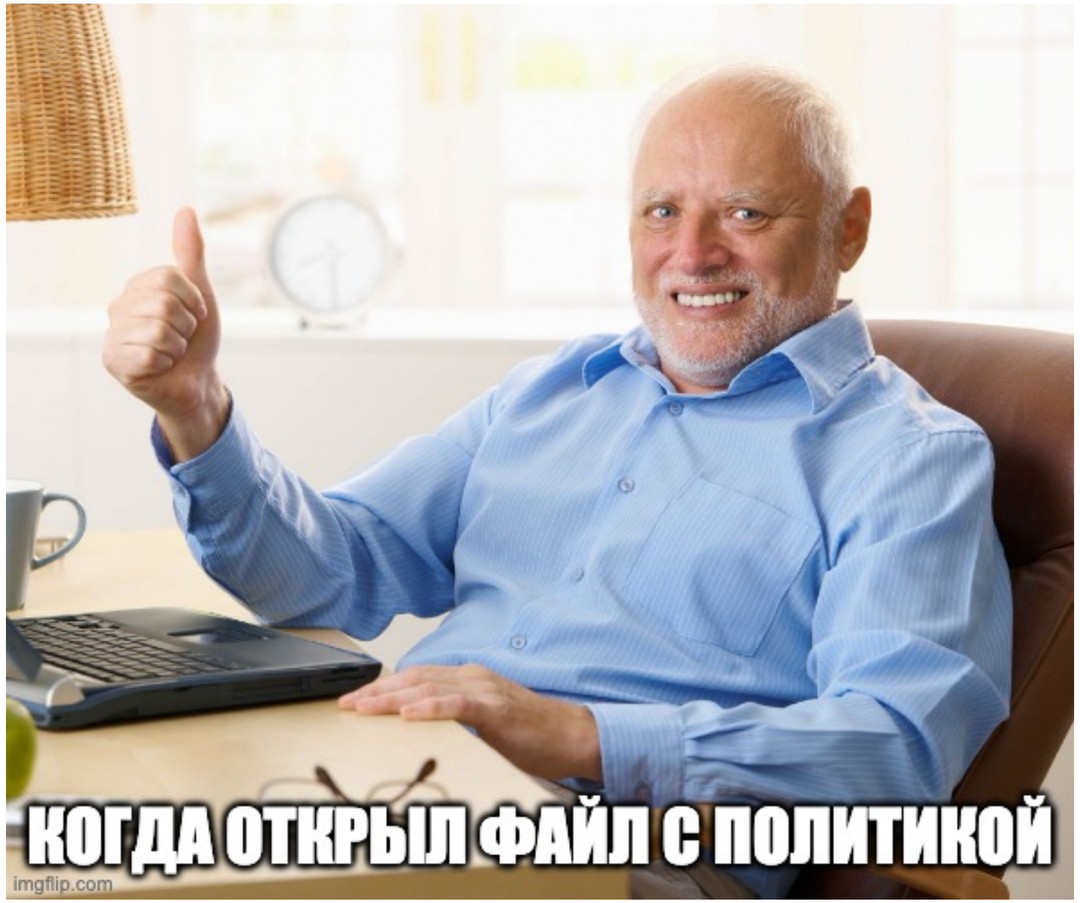
- Не забыли
- $ cat cilium-policies.yaml | wc -l
  $ 1472

Интересный вывод

- Удобно и понятно только на поиграться

- В реальном проде будет сложно

- Шанс, что инженер сойдет с ума и уволится повышается



Мы научились хранить политики в двух файлах

- В двух файлах получается меньше строк, чем в одном
- Их можно делить, например, по направлениям
- Так у нас появились файлы:

  appname-ingress-policy.yaml

  appname-egress-policy.yaml

И стали строить светлое будущее

- Структура репозитория улучшилась
- Понимание кода увеличилось
- Время на поиск и изменение политик снизилось

- Появляются мысли, что мы на правильном пути
- Но не покидает чувство, что где-то есть проблемка...

Смотрим на файлики

Не прокатило, вычеркиваем

-Все так же большой объем файла

-Читаемость падает

- И возникает две интересных проблемы

Больше файлов богу файлов

- Начали пользоваться шаблоном для именования:

  {app}-{component}-{destination}

- Файлы стали выглядеть так

  cilium-hubble-ui-ingress.yaml

Что это дало

- Репозиторий стал упорядоченным
- Перестали появляться файлы на 1к строк
- Поиск по репозиторию стал удобным

- Количество файлов растет
- Необходимо точное понимание, какой компонент править

Курица-яйцо

- У нас появилось приложение, которому нужен ingress
- Приложению нужно 2 простых политики
- Куда какую складывать?

```
- fromEndpoints:
    - matchLabels:
        ingress: istio
  toPorts:
    - ports:
        - port: "5050"
```

```
- toEndpoints:
    - matchLabels:
        io.kubernetes.pod.namespace.labels.infra: "true"
        myapp: super
```

Все к приложению

Обе политики можно сложить в папку с политиками приложения

- Удобно при выкатке
- Удобно править
- Все политики приложения в одном месте

- Крайне сложно отслеживать политики для ingress’ов
- Ломается принцип «политики компонента в одном месте»

Разнести все отдельно

Кладем ingress-политику к нашему приложению
egress – к политикам istio-ingress

- Все на своих местах
- Политику можно легко поправить

- Потерять политику становится очень просто
- Нет консистентных выкаток

Лейблы к нам спешат

Лейблы на ns

- Вешаем лейблы на неймспейсы, например, pfm-admins=true
- Делаем простую политику
- Во всех ns с правильным лейблом автоматически будет открываться доступ с ingress

```yaml
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: istio-ingressgateway-infra-istio-ingressgateway.egress.to.external-namespace-pfm-admins
  namespace: pfm-istio
spec:
  description: access from istio-ingressgateway to everyone with NS label pfm-admins=true
  egress:
  - toEndpoints:
    - matchExpressions:
      - key: io.kubernetes.pod.namespace
        operator: Exists
      matchLabels:
        k8s:io.cilium.k8s.namespace.labels.pfm-admins: "true"
  endpointSelector:
    matchLabels:
      k8s:app.kubernetes.io/instance: istio-ingressgateway-infra
      k8s:app.kubernetes.io/name: istio-ingressgateway
      k8s:io.kubernetes.pod.namespace: pfm-istio
```

И как оно?

- Не создаем лишних политик
- Приложения сразу получают ingress

- Открытие доступа сразу на весь ns
- Сложно выдать гранулярный доступ
- Порог вхождения выше

Немного вспомним, что уже обсудили

- Разобрались с файлами
- Поняли как жить с ingress’ами
- Пока очень удобно, если у нас только один кластер
- А если много – надо все шаблонизировать!

Шаблончики для удобства

```yaml
metadata:
  name: "cluster.{{ .Release.Name }}.all.egress.to.{{ .Release.Namespace }}-{{ .Release.Name }}"
spec:
  endpointSelector: {}
  egress:
    - toEndpoints:
      - matchLabels:
        k8s:io.kubernetes.pod.namespace: "{{ .Release.Namespace }}"
        k8s:app.kubernetes.io/instance: "{{ .Release.Name }}"
        k8s:app.kubernetes.io/name: "{{ .Release.Name }}"
```

- Helm сам подставит нужные значения
- В кластере будет понятное имя политики
- И это просто красиво

Ну круто же

- В кластерах понятные имена политик
- Все имена единообразны
- Инженер трогая кластер руками понимает, что это за политика и зачем она нужна

- В чарте больше одного приложения
- Приложение умеет порождать дочерние контейнеры

Gitlab runner

- Выглядит неплохо
- Все необходимые переменные поставятся автоматически

```
metadata:
  name: "{{ .Release.Name }}.egress.to.gitlab"
egress:
  - toCIDRSet:
    {{- range .Values.global.networkAddresses.gitlab }}
    - {{ . }}
    {{- end }}
    toPorts:
    - ports:
      - port: "443"
        protocol: TCP
endpointSelector:
    matchLabels:
      k8s:app.kubernetes.io/instance: "{{ .Release.Name }}"
      k8s:app.kubernetes.io/name: "{{ .Release.Name }}"
      k8s:io.kubernetes.pod.namespace: "{{ .Release.Namespace }}"
```

Gitlab runner все сломал

- В кластере больше 1 раннера
- Они разные
- Раннеры порождают дочерние процессы, которым необходим другой доступ

- На этом этапе произошла трагедия

Как решить проблему

- Стали использовать больше понятных обозначений для имени политики

- Поменяли соглашение об именовании:
  {Release}-{sub-process}-{destination}-to-{service}

Все равно думаем, как можно улучшить

Стоит ли все шаблонизировать

- Читаемо, понятно, меньше опечаток
- Глаза привыкают к структуре, удобно искать в кластере
- Единообразие во всех кластерах

- Приложения с дочерними процессами
- Несколько приложений в релизе

Что получается

Что мы имеем:

- Политики шаблонизируются
- Именование одинаковое во всех кластерах
- Довольно удобно хранить в файлах

Что забыли:

- Как хранить в репозитории

Где сложности

- Разные cni

- Должно быть удобно всем

- В кластер не попадают лишние элементы

Как решили

– В репозитории вынесли выше уровня кластера

– Отдельная структура каталогов:

network-policies:

> calico

> cilium

- Основной минус такого подхода: необходимо указывать тип cni для каждого кластера

И что же вышло

- Политики для разных cni в одном репозитории
- Можно править независимо
- Выбор типа политик делается один раз на уровне кластера

- Нет гарантии «одинаковости» политик в легаси и новом кластере
- Нужны дополнительные переменные

Начнем подводить итоги

Что было вначале:

- Полный бардак в репозитории
- Сложности с шаблонизацией
- Отсутствие единого именования
- Боль и страх в глазах инженеров

Как стало

- Привели имена политик к единому виду
- Упорядочили структуру файлов
- Упорядочили расположение политик в репозитории
- Инженеры стали больше улыбаться

Что точно можно порекомендовать

Если только начинаете использовать политики:

- Заранее договоритесь о именовании
- Продумайте структуру репозитория и файлов

Если уже используете:

- Заведите страницу для болей
- Проводите ревью раз в пару месяцев

Пользуйтесь сетевыми политиками

Делайте свой кластер безопаснее и удобнее

Спасибо!

Contacts:

Email: avernusalex@gmail.com

Tg: @kozhemyash