DevOops

2022

Linkerd: the Hard Way

@aatarasoff

@aatarasoff

@aatarasoff

@aatarasoff


Sputnik V
Vaccinated


LINKERD HERO

Минздрав предупреждает

Мнение докладчика может не совпадать с официальной позицией кого бы то ни было.

Все представленные в докладе сведения, примеры, выводы и другую информацию вы можете использовать на свой страх и риск. За все ваши действия ответственность несёте только вы сами.

No Service Mesh 101 :)

Может ли Service Mesh “просто работать”?

- **Minimalistic**
- **Easy to Use**
- **Ultralight**

```
linkerd install --crds | kubectl apply -f -
```

```
linkerd install --crds | kubectl apply -f -

linkerd install        | kubectl apply -f -
```

```
linkerd install --crds | kubectl apply -f -

linkerd install --ha   | kubectl apply -f -
```

И это на самом деле работает

	Name ↑	Status	Type	Pods	Namespace
	linkerd-destination	OK	Deployment	3/3	linkerd
	linkerd-heartbeat	OK	Cron Job	0/0	linkerd
	linkerd-identity	OK	Deployment	3/3	linkerd
	linkerd-proxy-injector	OK	Deployment	3/3	linkerd

Установка

- cli
  - на попробовать
- helm charts
  - production-окружение

Production Ready

- HA Configuration
- Separated Node Pool
  - Control Plane
- Proxy Resources
  - requests
  - limits

Linkerd Configuration

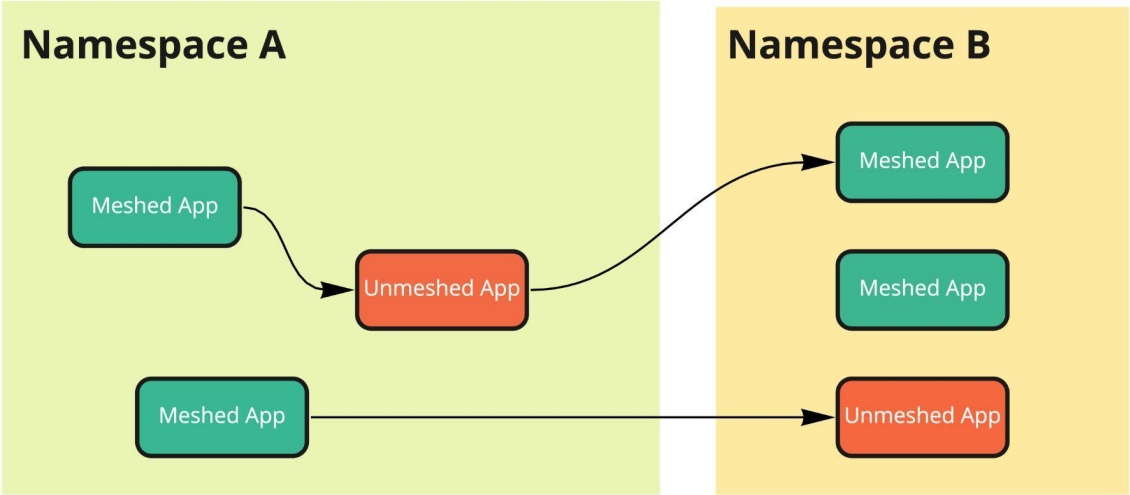
```
nodeSelector:
  cloud.google.com/gke-nodepool: "linkerd-pool"
tolerations:
- key: "linkerd"
  operator: "Equal"
  value: "true"
  effect: "NoSchedule"
...
proxy:
  ...
  resources:
    cpu:
      limit: 1024m
      request: 5m
    memory:
      limit: 250Mi
      request: 8Mi
```

Linkerd Configuration

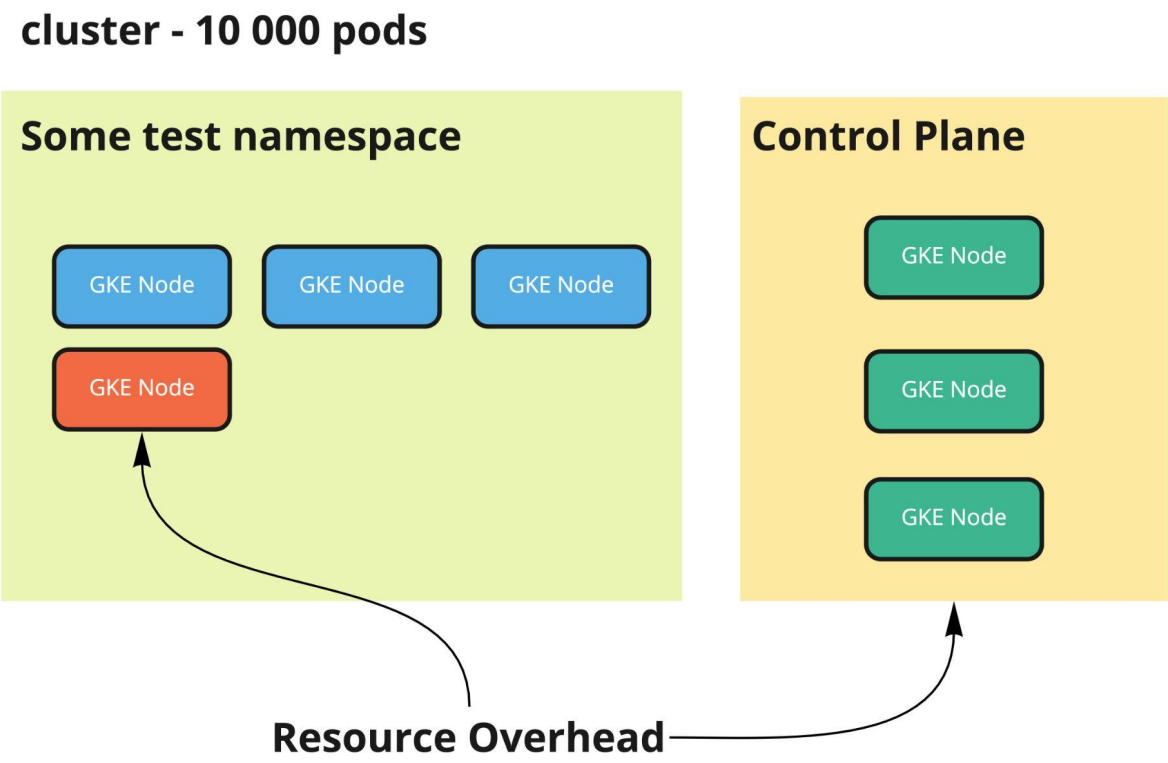
```
nodeSelector:
  cloud.google.com/gke-nodepool: "linkerd-pool"
tolerations:
- key: "linkerd"
  operator: "Equal"
  value: "true"
  effect: "NoSchedule"
...
proxy:
  ...
  resources:
    cpu:
      limit: 1024m
      request: 5m
    memory:
      limit: 250Mi
      request: 8Mi
```

Миграция на Linkerd

```
# POD or Namespace definition
annotations:
  linkerd.io/inject: enabled
```

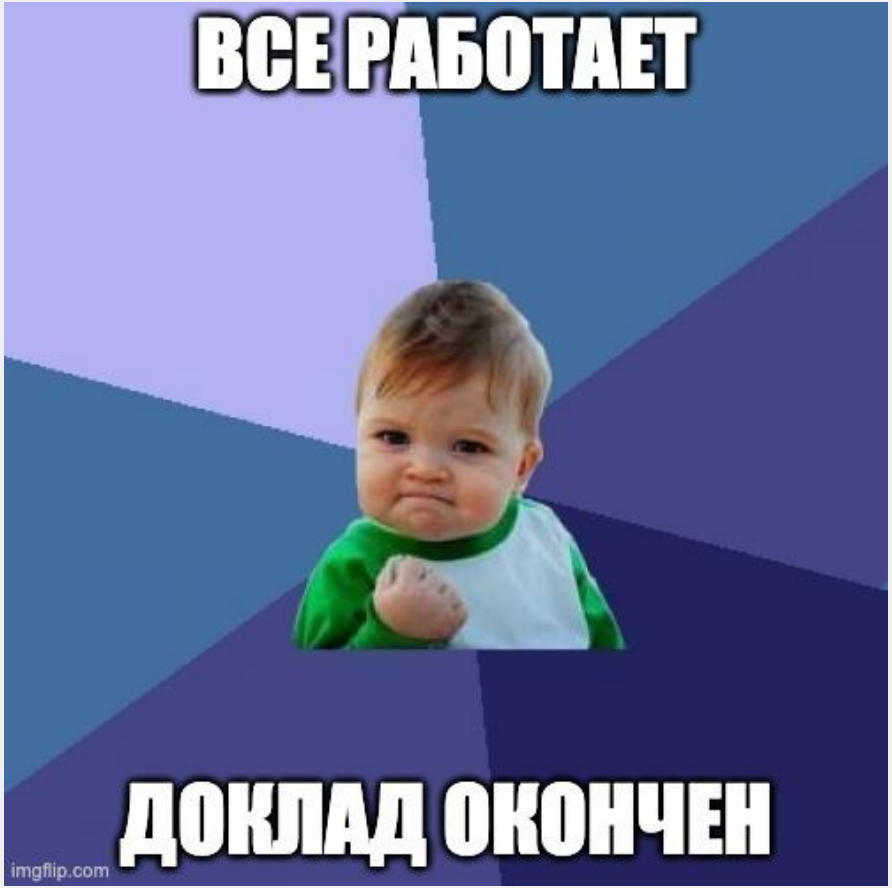


CPU/Memory Linkerd



CPU/Memory Linkerd vs Istio

https://linkerd.io/2021/05/27/linkerd-vs-istio-benchmarks/


ВСЕ РАБОТАЕТ
ДОКЛАД ОКОНЧЕН
imgflip.com

А что у нас работает?

Linkerd Core

Работает

- mTLS + политики авторизации
- Load Balancing
- Traffic Split
- Ingress

Не работает)

- Meshed Cron Jobs
- Observability
- Distributed Tracing
- Политики авторизации

Linkerd Extensions

- linkerd-viz
- linkerd-jaeger
- ~~linkerd-multicluster~~
- linkerd-easyauth

Linkerd: the Hard Way

Текущее состояние

Работает

- mTLS + политики авторизации
- Load Balancing
- Traffic Split
- Ingress

Не работает

- Meshed Cron Jobs
- Observability
- Distributed Tracing
- Политики авторизации

Текущее состояние

Работает

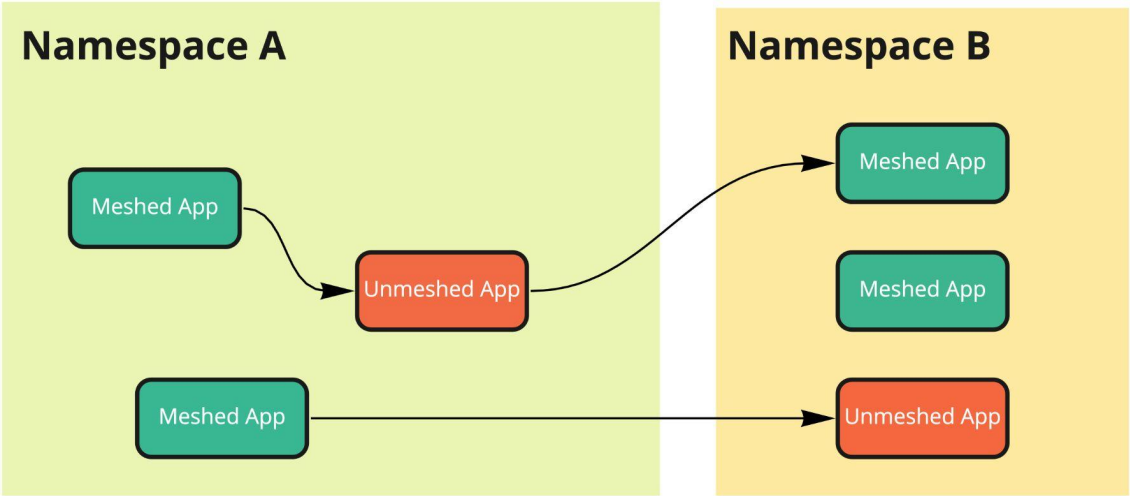
- mTLS + политики авторизации
- Load Balancing
- Traffic Split
- Ingress

Не работает

- Meshed Cron Jobs
- Observability
- Distributed Tracing
- Политики авторизации

Такие простые аннотации

```
# POD or Namespace definition
annotations:
  linkerd.io/inject: enabled
```



Проблема Jobs/CronJobs

Containers

Job is completed

Name	Status	Image	Restart count	Logs
-upload	Completed		0	View logs
linkerd-proxy	Running	'linkerd-proxy:stable-2.11.0	0	View logs

Простой вариант решения проблемы

Containers

Job is completed

Name	Status	Image	Restart count	Logs
-upload	Completed			
linkerd-proxy	Running			

```
jobTemplate:
 spec:
   template:
     metadata:
       annotations:
         linkerd.io/inject: disabled
```

Проблема Jobs/CronJobs

Не работает, если вам нужно делать Mesh для джобов

Containers

Job is completed

Name	Status	Image	Restart count	Logs
-upload	Completed			
linkerd-proxy	Running			

```
jobTemplate:
 spec:
   template:
     metadata:
       annotations:
         linkerd.io/inject: disabled
```

Проблема Jobs/CronJobs

```
command: ["/bin/sh"]
args: ["-c",
       "<job_run_command_here>",
       "; CODE=$?; wget --post-data hello=shutdown
http://localhost:4191/shutdown; exit $CODE;"]
```

Containers

Name	Status			Logs
-upload	Completed		0	View logs
linkerd-proxy	Running	'linkerd-proxy:stable-2.11.0	0	View logs

Проблема Jobs/CronJobs

https://github.com/linkerd/linkerd-await

Containers

Job is completed

Name	Status	Image	Restart count	Logs
-upload	Completed		0	View logs
linkerd-proxy	Running	'linkerd-proxy:stable-2.11.0	0	View logs

Проблема Jobs/CronJobs

https://github.com/linkerd/linkerd-await

```
command: ["/linkerd-await"]
args: ["--shutdown", "--", "/coolcommand"]
```

Containers

Name				ount	Logs
-upload	Completed			0	View logs
linkerd-proxy	Running		'linkerd-proxy:stable-2.11.0	0	View logs

Проблема Jobs/CronJobs

https://github.com/linkerd/linkerd-await

```
command: ["/linkerd-await"]
args: ["--shutdown", "--", "/coolcommand"]
```

Containers

Name			ount	Logs
-upload	Completed		0	View logs
linkerd-proxy	Running	'linkerd-proxy:stable-2.11.0	0	View logs

А как нам положить этот бинарь во все наши джобы???

Проблема Jobs/CronJobs

https://github.com/linkerd/linkerd-await

```
volumes:
- name: linkerd-await
  emptyDir: {}
```

Containers

Job is completed

Name	Status	Image	Restart count	Logs
-upload	Completed		0	View logs
linkerd-proxy	Running	'linkerd-proxy:stable-2.11.0	0	View logs

Проблема Jobs/CronJobs

https://github.com/linkerd/linkerd-await

```
volumes:
- name: linkerd-await
  emptyDir: {}
```

Containers

Name	Status	Image
-upload	Completed	
linkerd-proxy	Running	

```
initContainers:
- name: linkerd-await
  image: "<image>/linkerd-await:0.2.4"
  command: ["cp"]
  args: ["/tmp/linkerd-await", "/linkerd-await"]
  volumeMounts:
  - mountPath: /linkerd
    name: linkerd-await
```

Проблема Jobs/CronJobs

https://github.com/linkerd/linkerd-await

```
volumes:
- name: linkerd-await
  emptyDir: {}
```

Containers

Name	Status	Image
-upload	Completed	
linkerd-proxy	Running	

```
initContainers:
- name: linkerd-await
  image: "<image>/linkerd-await:0.2.4"
  command: ["cp"]
  args: ["/tmp/linkerd-await", "/linkerd-await"]
  volumeMounts:
  - mountPath: /linkerd
    name: linkerd-await
```

```
volumeMounts:
- mountPath: /linkerd
  name: linkerd-await
command: ["/linkerd-await"]
args: ["--shutdown", "--", "/coolcommand"]
```

https://itnext.io/three-ways-to-use-linkerd-with-kubernetes-jobs-c12ccc6d4c7c


LINKERD

linkerd / **linkerd-await**

Текущее состояние

Работает

- mTLS + политики авторизации
- Load Balancing
- Traffic Split
- Ingress
- Meshed CronJobs

Не работает

- Observability
- Distributed Tracing
- Политики авторизации

Linkerd Viz

Linkerd Dashboard

Метрики по неймспейсам

HTTP Metrics

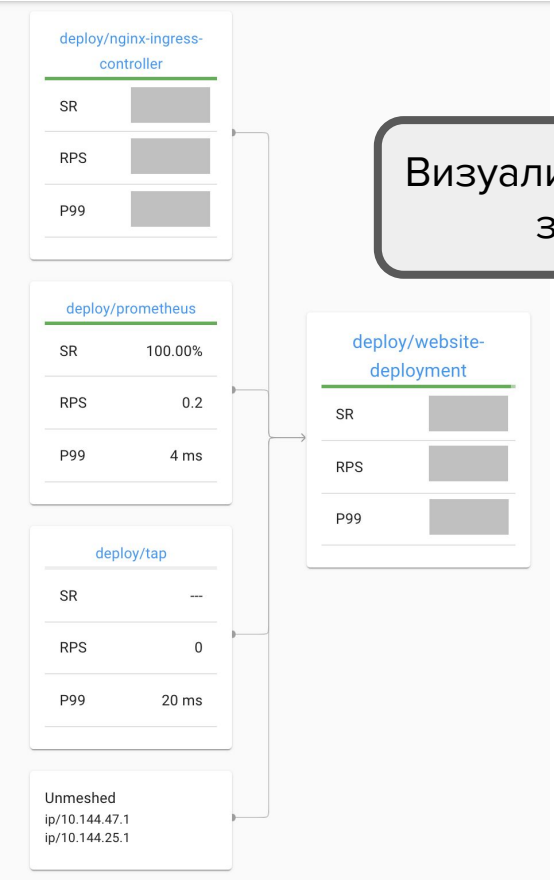
Namespace	Meshed	Success Rate	RPS	P50 Latency	P95 Latency	P99 Latency	Grafana
banking	99/99	99.88%			71 ms	203 ms	
production	273/273	99.97%			40 ms	163 ms	
assistance	41/41	100.00%			27 ms	98 ms	
common	2/2	100.00%			1 ms	1 ms	
compliance	23/23	100.00%			89 ms	186 ms	
datascience							

Real-Time метрики для сервиса

LIVE CALLS | ROUTE METRICS

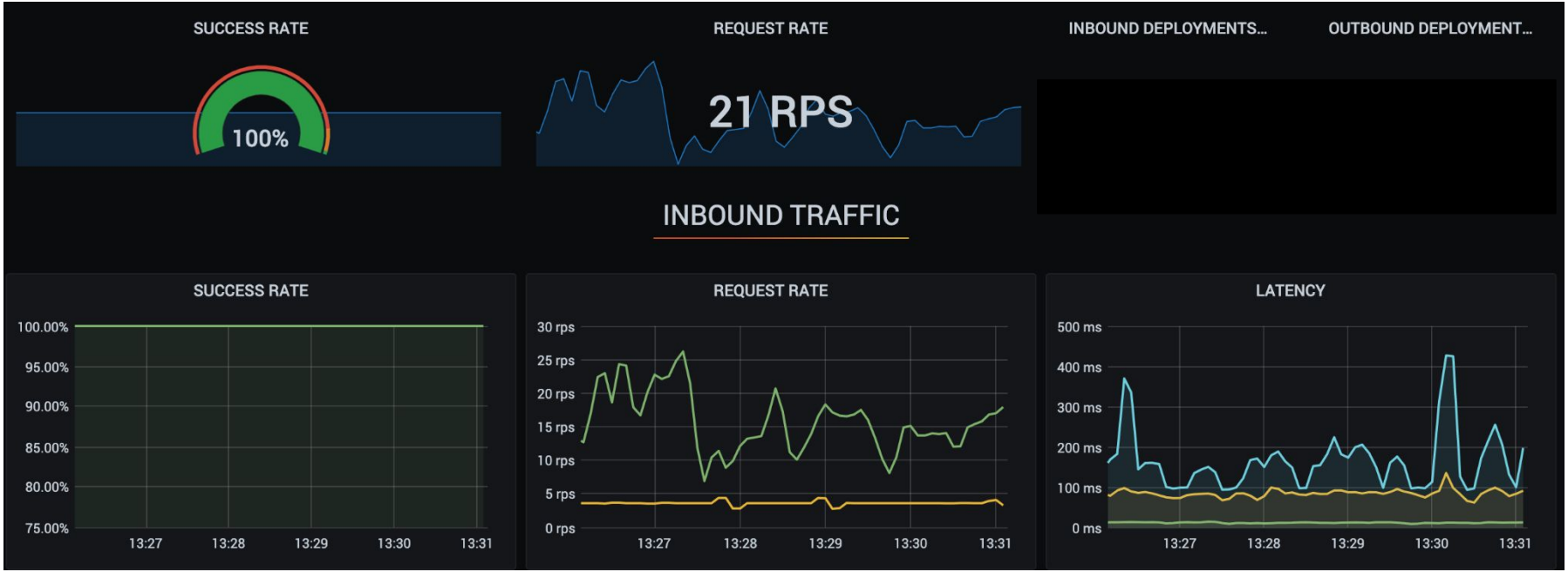
	Name	Method	Path	Count	Best	Worst	Last	Success Rate	Tap
FROM	deploy/banking-deployment	POST	/api/c	8	52 ms	95 ms	60 ms	100.00%	
FROM	deploy/account-statement-deployment	GET	/api/t	3	17 ms	34 ms	21 ms	100.00%	

Linkerd Dashboard



Визуализация потоков запросов

Унифицированные метрики



Prometheus + Metrics Overload

- Не бесплатно
  - 200-300 ворклоадов
  - 2 млрд. сэмплов за день
  - high cardinality

Prometheus + Metrics Overload

- Не бесплатно
  - 200-300 ворклоадов
  - 2 млрд. сэмплов за день
  - high cardinality

```
Типичная метрика linkerd содержит:
- pod source
- pod destination

> my-deployment-sh5559f-dxj
```

Prometheus + Metrics Overload

- Не бесплатно
  - 200-300 ворклоадов
  - 2 млрд. сэмплов за день
  - high cardinality
- Real-Time Only
  - держать данные в течение часа
  - 16 GB и более оперативной памяти
  - и всё равно стабильность и латенси не очень

Prometheus + Metrics Overload

- Не бесплатно
  - 200-300 ворклоадов
  - 2 млрд. сэмплов за день
  - high cardinality
- Real-Time Only
  - держать данные в течение часа
  - 16 GB и более оперативной памяти
  - и всё равно стабильность и латенси не очень
- Stackdriver (Monarch by Google)
  - дорого и относительно медленно
  - может и не заработать (ограничение на 55 лейблов)

Например будет так

Managed Service for Prometheus

✓ 1 hour | 6 hours | 1 day | 1 week | 1 month | 6 weeks | CUSTOM

While in Preview, this page can only query metrics sent via Google Cloud Managed Service for Prometheus. LEARN MORE

PromQL Query

request_total

RUN QUERY

Query duration

Series returned

LINE | STACKED AREA

generic::invalid_argument: One or more errors parsing the query.

А теперь добро пожаловать в OpenSource!

Prometheus Config. Reduce Labels

```
...
# Copy all pod labels to tmp labels
- action: labelmap
  regex: __meta_kubernetes_pod_label_(.+)
  replacement: __tmp_pod_label_$1

... # some other action here


# Copy tmp labels into real labels
- action: labelmap
  regex: __tmp_pod_label_(.+)
```

Prometheus Config. Reduce Labels

```
...
# Copy all pod labels to tmp labels
- action: labelmap
  regex: __meta_kubernetes_pod_label_(.+)
  replacement: __tmp_pod_label_$1

... # some other action here


# Copy tmp labels into real labels
- action: labelmap
  regex: __tmp_pod_label_(.+)
```

Prometheus Config. Reduce Labels

```
...
# Copy all pod labels to tmp labels
- action: labelmap
  regex: __meta_kubernetes_pod_label_(.+)
  replacement: __tmp_pod_label_$1

... # some other action here


# Copy tmp labels into real labels
- action: labelmap
  regex: __tmp_pod_label_(.+)
```

Prometheus Config. Reduce Labels

```
...
# Copy all pod labels to tmp labels
- action: labelmap
  regex: __meta_kubernetes_pod_label_(.+)
  replacement: __tmp_pod_label_$1

... # some other action here


# Copy tmp labels into real labels
- action: labelmap
  regex: __tmp_pod_label_(.+)
```

Deep Dive into Metrics

```
topk(10, count by (__name__)({__name__=~".+"}))
```

Deep Dive into Metrics

```
topk(10, count by (__name__)({__name__=~".+"}))
```

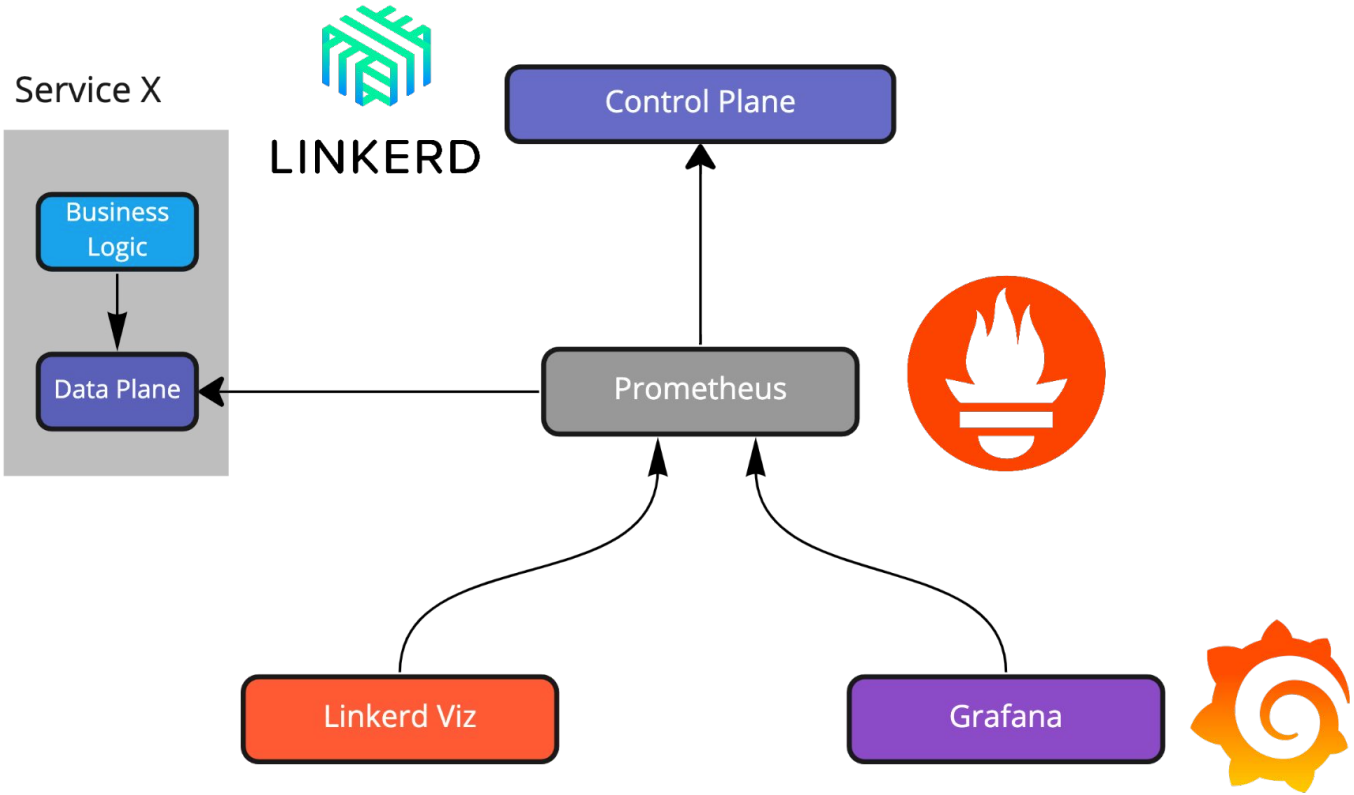
```
response_latency_ms_bucket:
 -  histogram
 -  > 250K cardinality
 -  additional buckets: 2/3/4/5…/40000/50000
```

Prometheus Config. Reduce Cardinality

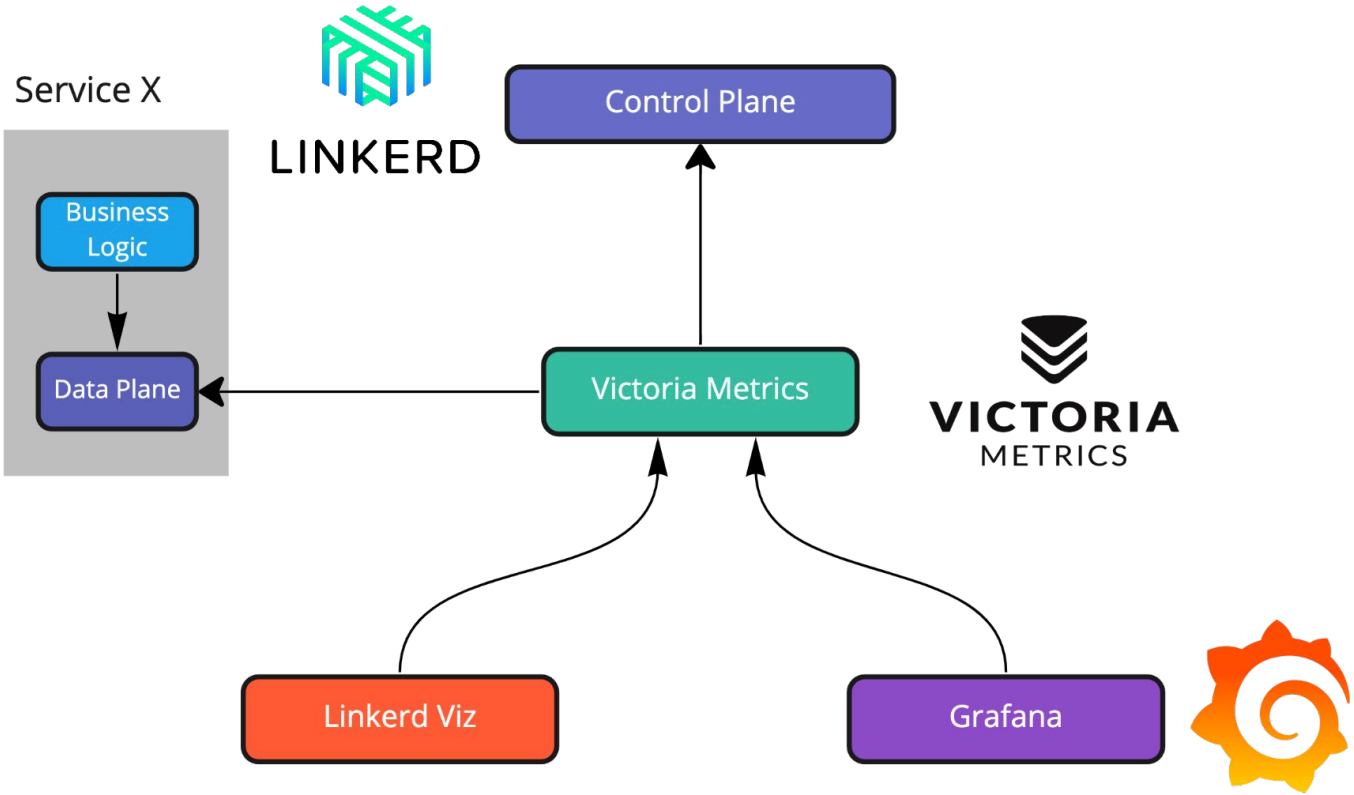
```
... # some other action here

metric_relabel_configs:
  - action: drop
    source_labels: [le]
    regex: "2.*|3.*|4.*|5.*"

... # some other action here
```

```
response_latency_ms_bucket:
 -  250K -> 70K cardinality
```

Prometheus -> Victoria Metrics



Prometheus -> Victoria Metrics



VM Config. Reuse Prometheus

```
- args:
  - -promscrape.config.strictParse=true
  - -promscrape.config=/etc/prometheus/prometheus.yml
  - -promscrape.maxScrapeSize=256MB
  - -storageDataPath=/data
  - -retentionPeriod=30d
  - -httpListenAddr=:9090
  - -search.maxSeries=100000
  - -search.maxUniqueTimeseries=1000000
```

VM Config. Reuse Prometheus

```
- args:
  - -promscrape.config.strictParse=true
  - -promscrape.config=/etc/prometheus/prometheus.yml
  - -promscrape.maxScrapeSize=256MB
  - -storageDataPath=/data
  - -retentionPeriod=30d
  - -httpListenAddr=:9090
  - -search.maxSeries=100000
  - -search.maxUniqueTimeseries=1000000
```

~~global:~~
~~ evaluation_interval: 10s~~
~~rule_files:~~
~~ - /etc/prometheus/*_rules.yml~~
~~ - /etc/prometheus/*_rules.yaml~~

VM Config. Reuse Prometheus

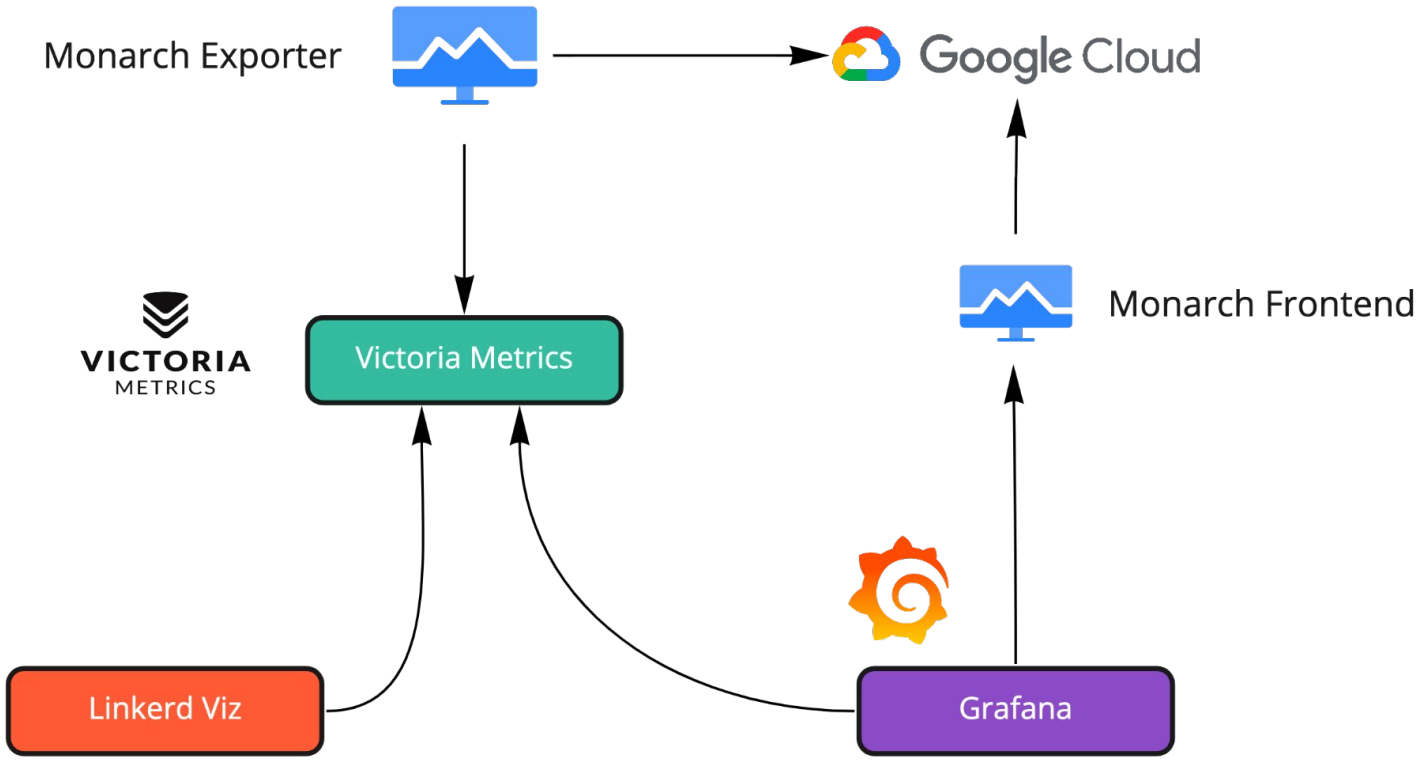
```
- args:
  - -promscrape.config.strictParse=true
  - -promscrape.config=/etc/prometheus/prometheus.yml
  - -promscrape.maxScrapeSize=256MB
  - -storageDataPath=/data
  - -retentionPeriod=30d
  - -httpListenAddr=:9090
  - -search.maxSeries=100000
  - -search.maxUniqueTimeseries=1000000
```

VM Config. Reuse Prometheus

```
- args:
  - -promscrape.config.strictParse=true
  - -promscrape.config=/etc/prometheus/prometheus.yml
  - -promscrape.maxScrapeSize=256MB
  - -storageDataPath=/data
  - -retentionPeriod=30d
  - -httpListenAddr=:9090
  - -search.maxSeries=100000
  - -search.maxUniqueTimeseries=1000000
```

В целом норм, но давайте больше?

Victoria Metrics -> Monarch



Monarch Config. Scraping

```
data:
  evaluation_rules.yml: |-
    groups:
    - name: monarch
      rules:
      - record: job:response_latency_ms_bucket:sum
        expr: sum by (deployment, namespace, dst_deployment,
direction, le) (response_latency_ms_bucket)
      - record: job:response_total:sum
        expr: sum by (deployment, namespace, dst_deployment,
direction, status_code, classification) (response_total)
```

Только нужные метрики

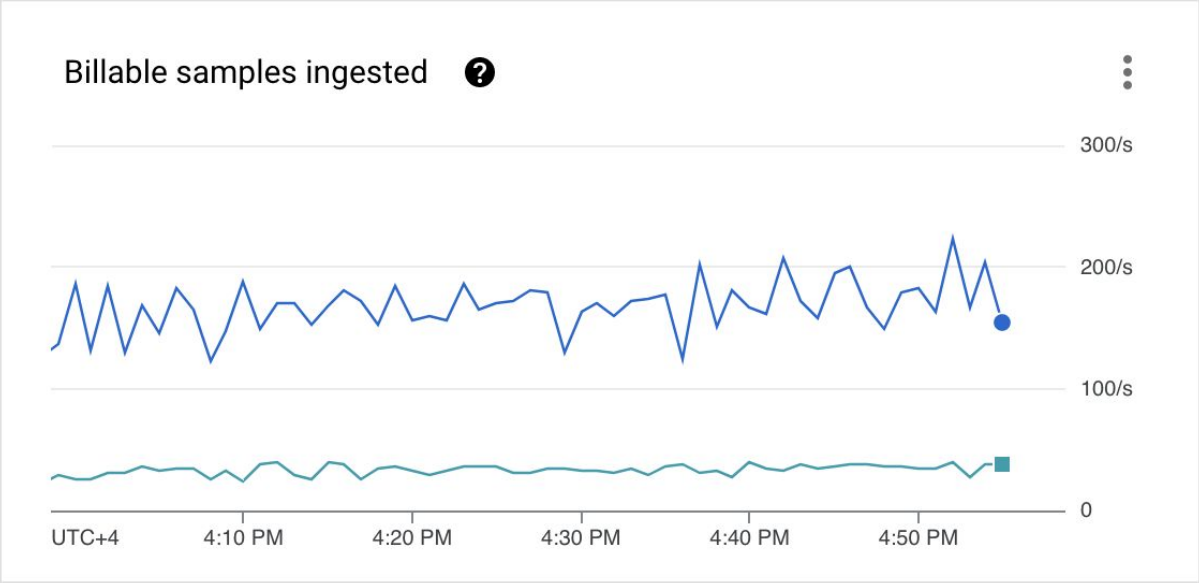
Monarch Resource Consumption

Billable bytes ingested

0 B

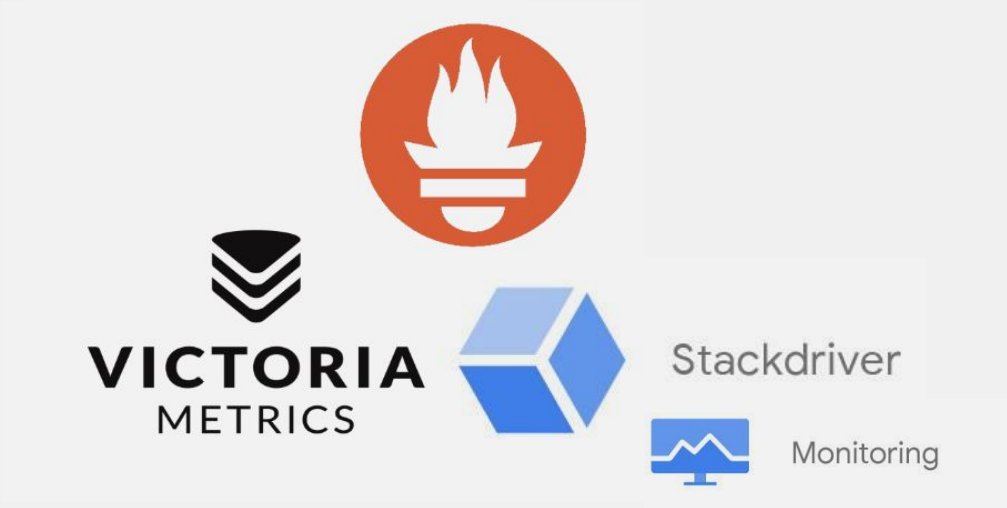
Billable samples ingested

688.57K



linkerd / **linkerd-await**

linkerd / **linkerd-await**



Текущее состояние

Работает

- mTLS + политики авторизации
- Load Balancing
- Traffic Split
- Ingress
- Meshed CronJobs
- Observability

Не работает

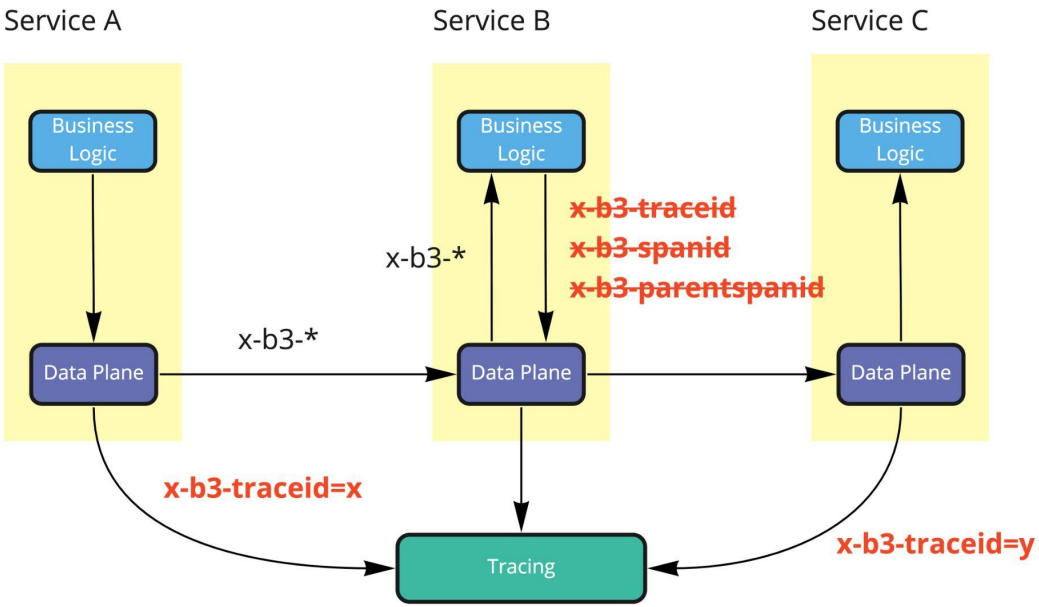
- Distributed Tracing
- Политики авторизации

Distributed Tracing a la Linkerd Jaeger

Linkerd Distributed Tracing Problem

x-b3-traceid
x-b3-spanid
x-b3-parentspanid
x-b3-sampled
x-b3-flags

should be propagated (as x-request-id)



Решение проблемы трейсов

- Инструментация
  - opentelemetry
  - linkerd-proxy supports x-b3-* only
- Проброс заголовков
  - x-b3-*

Решение проблемы трейсов

- ~~Инструментация~~
  - ~~opentelemetry~~
  - ~~linkerd-proxy supports x-b3-* only~~
- Проброс заголовков
  - x-b3-*

Схема сбора трейсов

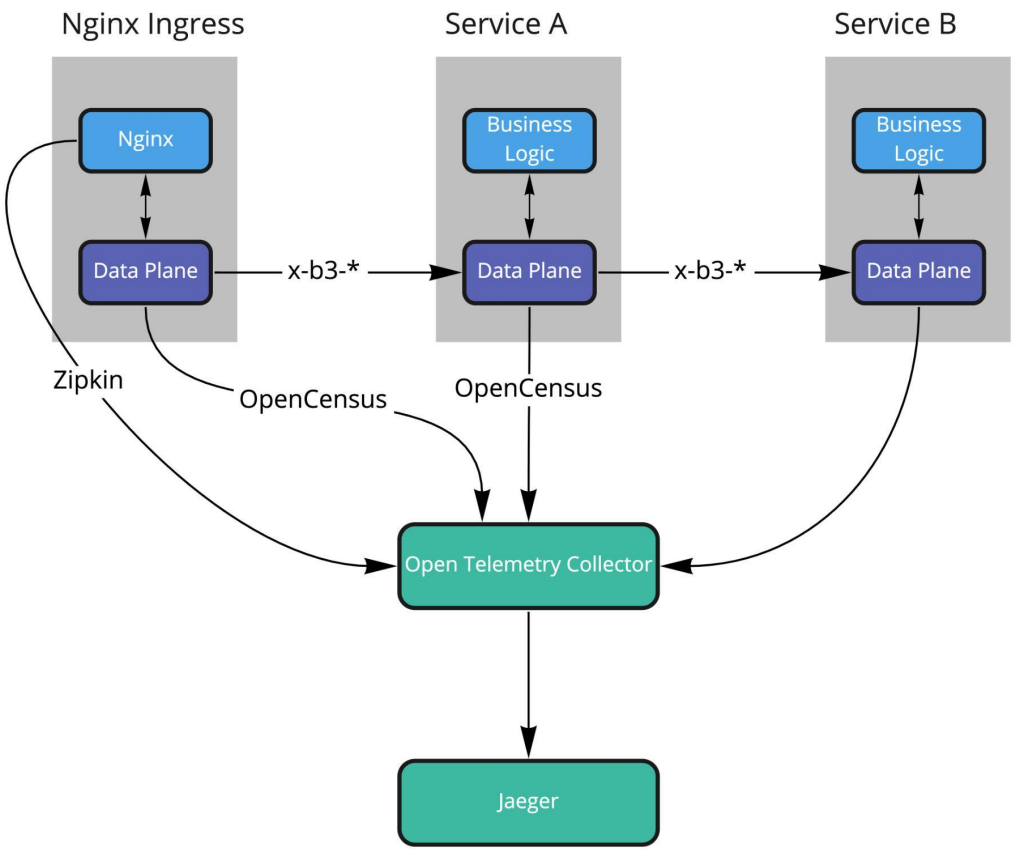


Схема сбора трейсов

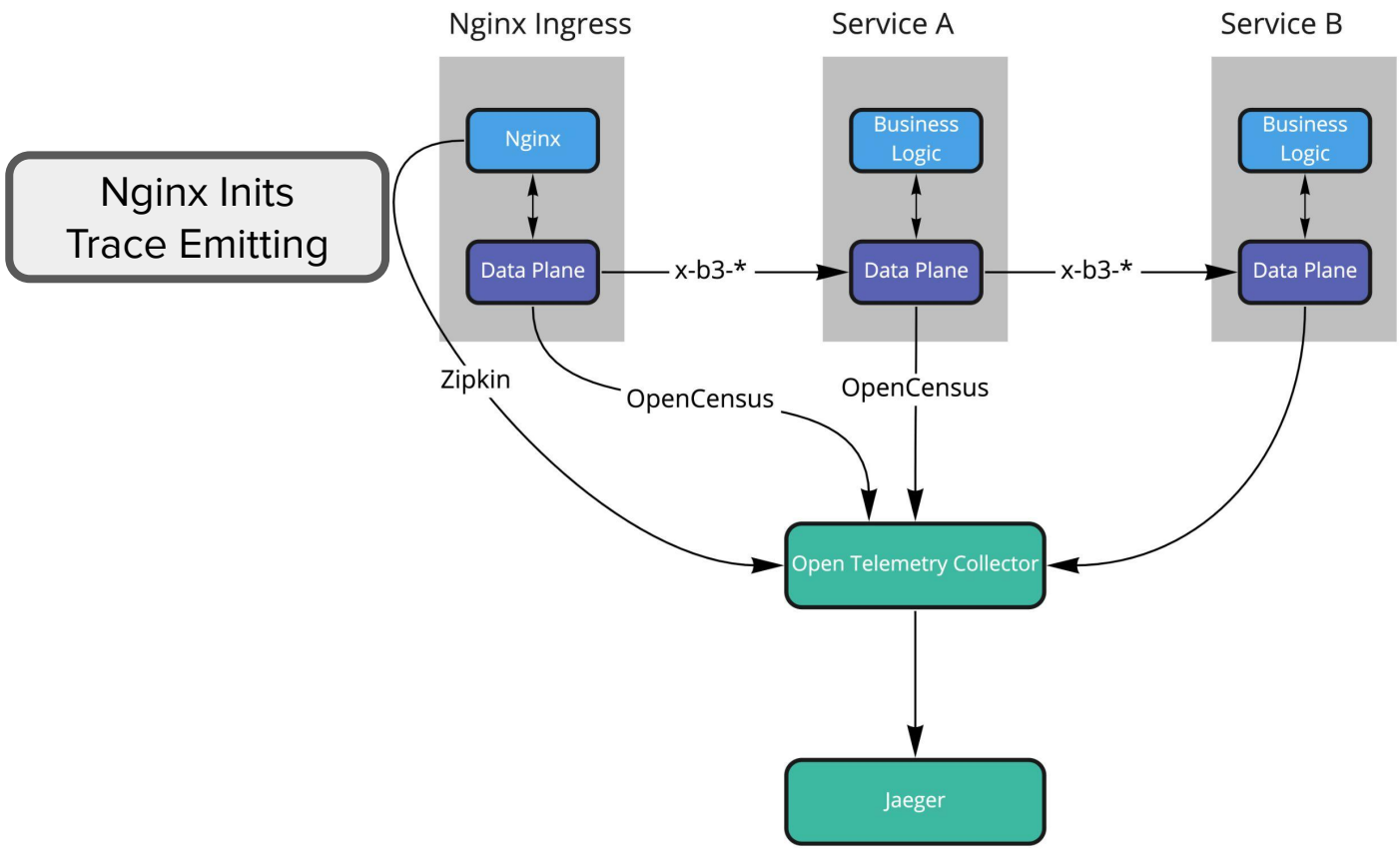


Схема сбора трейсов

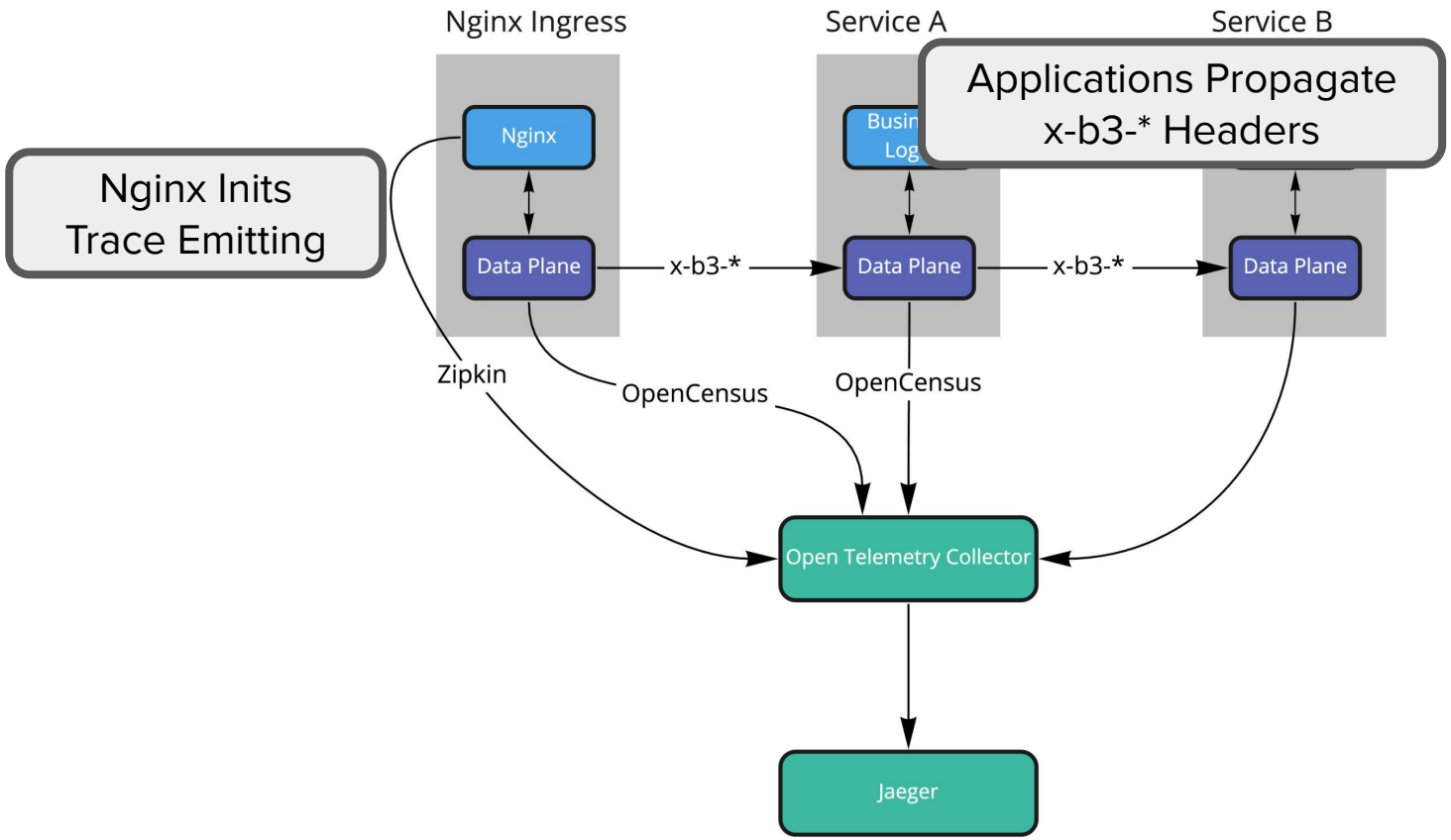


Схема сбора трейсов

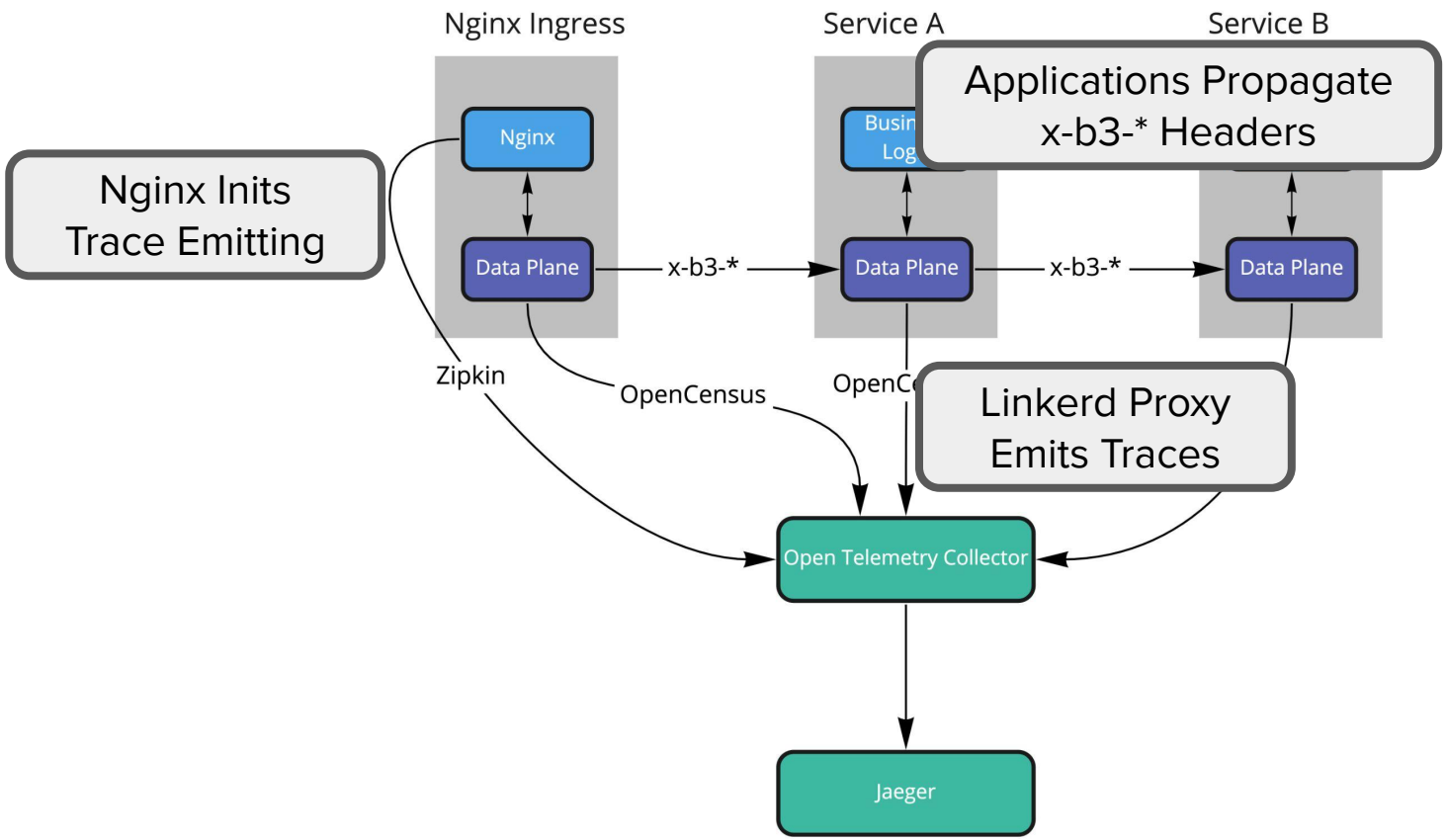


Схема сбора трейсов

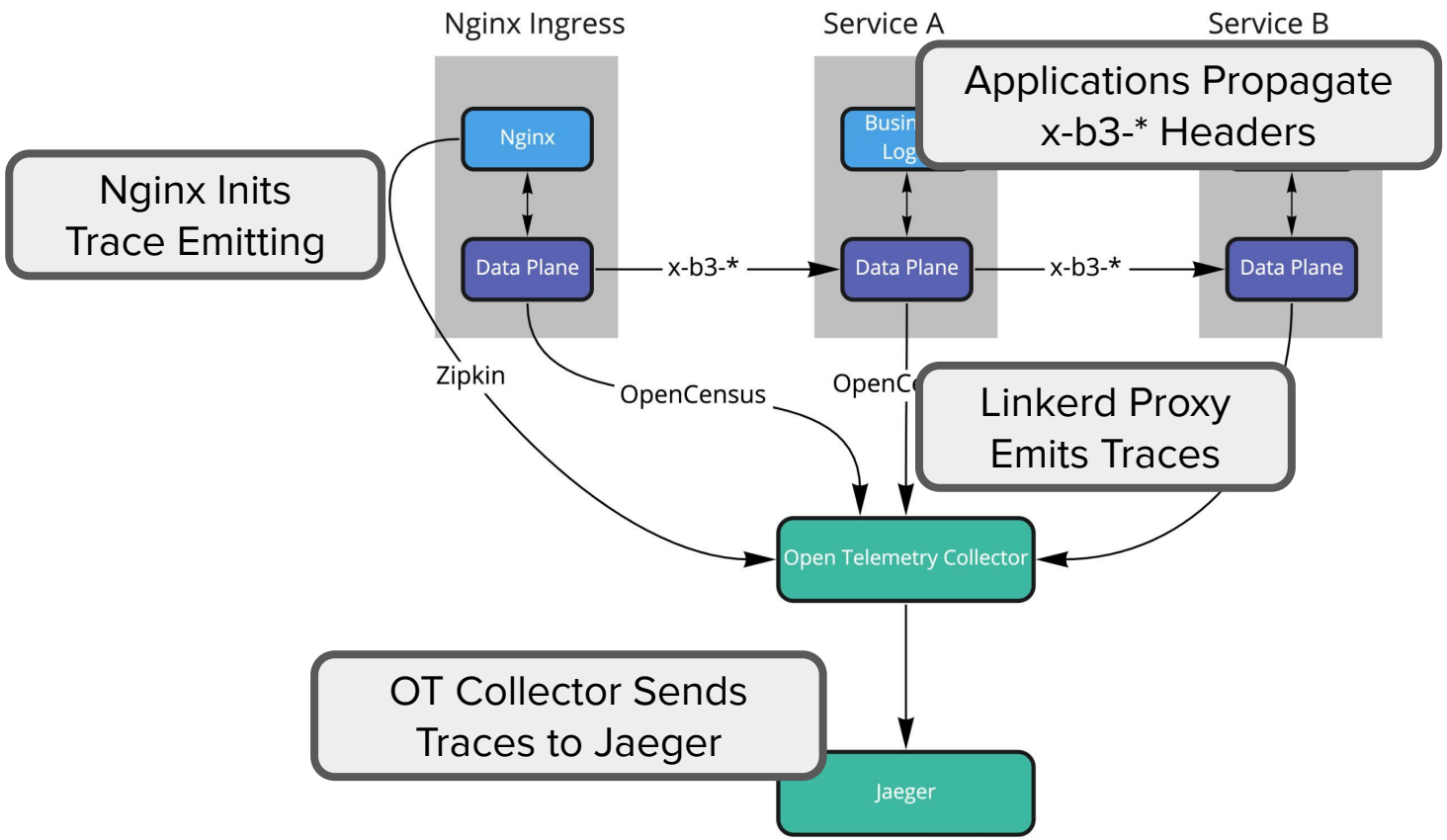
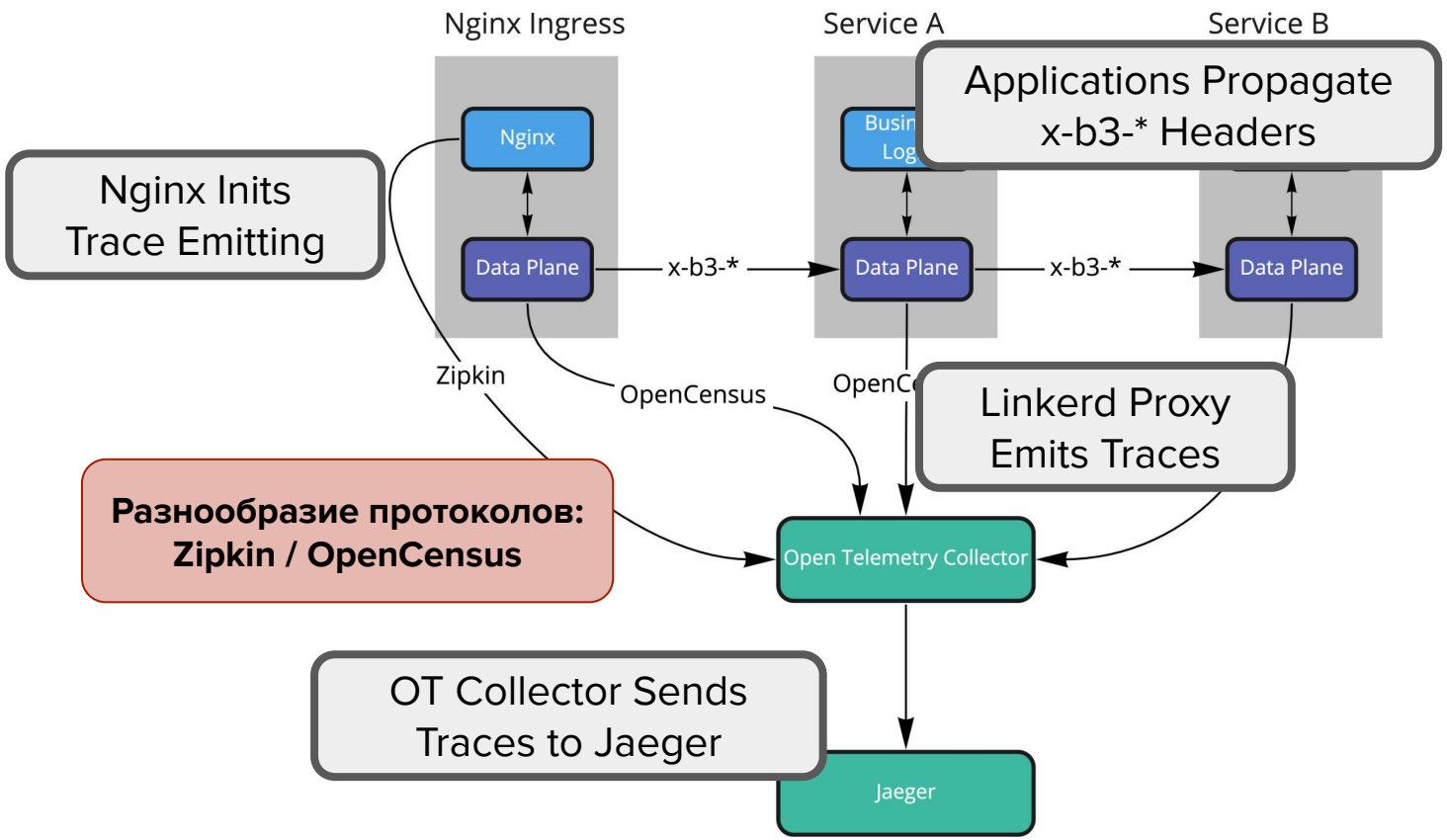


Схема сбора трейсов



Nginx Ingress Config

```
data:
  enable-opentracing: "true"
  zipkin-collector-host: collector.linkerd-jaeger
  zipkin-sample-rate: "0.2"
  ...
kind: ConfigMap
```

Nginx Ingress Config

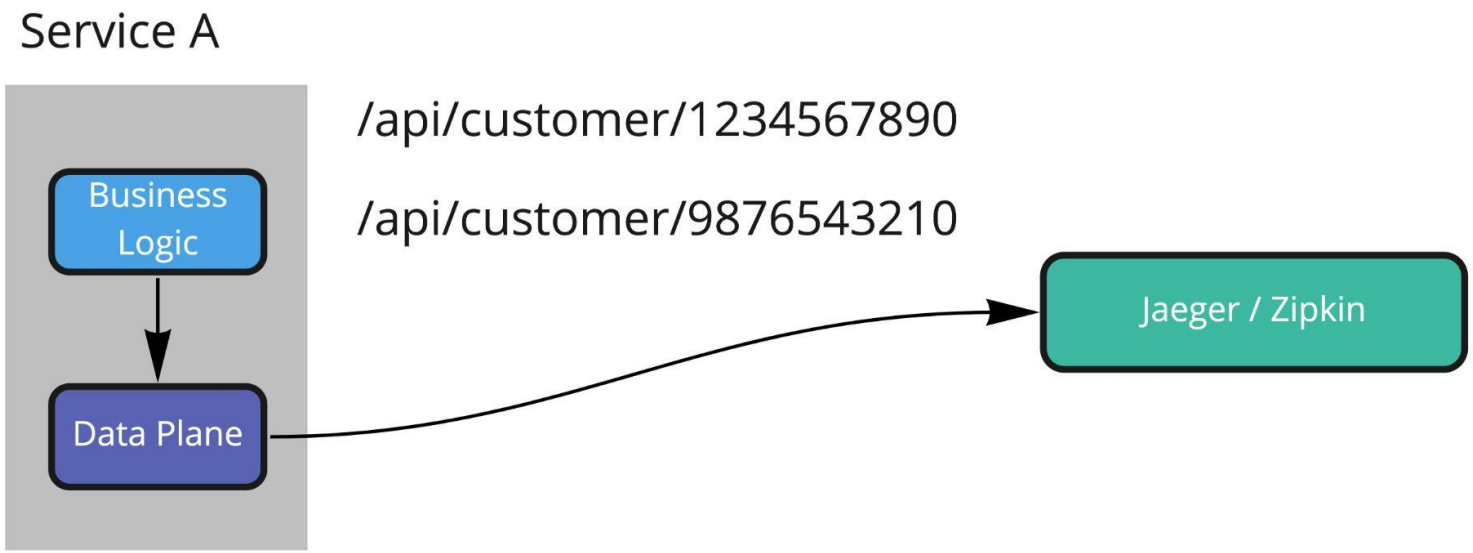
```
data:
  enable-opentracing: "true"
  zipkin-collector-host: collector.linkerd-jaeger
  zipkin-sample-rate: "0.2"
  ...
kind: ConfigMap
```

Разнообразие протоколов:
Zipkin / Jaeger

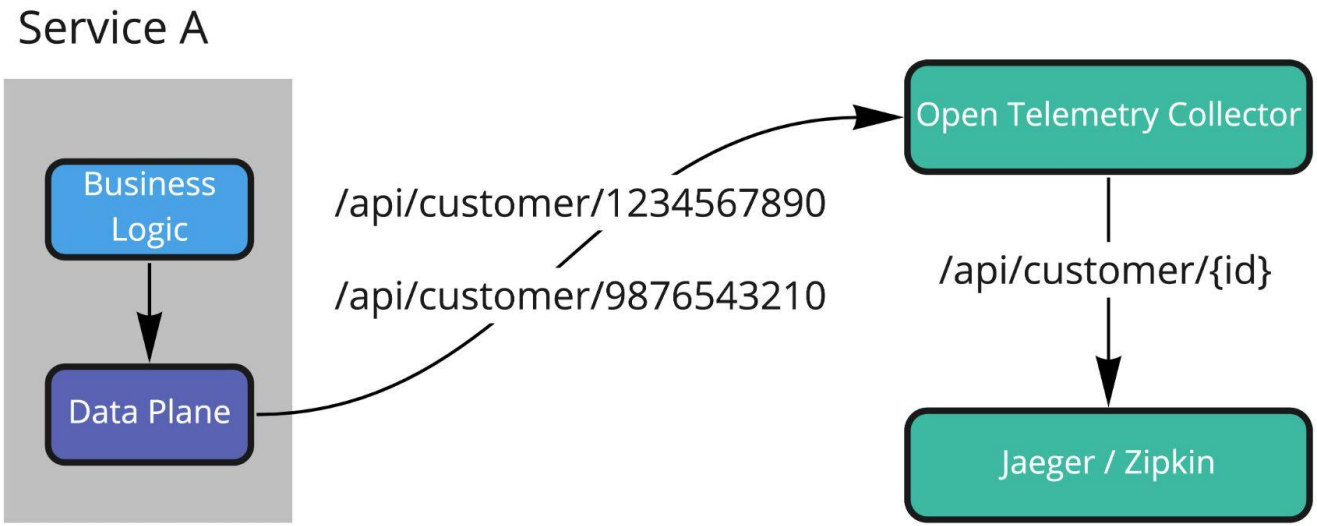
Nginx Ingress Config

```
data:
  enable-opentracing: "true"
  zipkin-collector-host: collector.linkerd-jaeger
  zipkin-sample-rate: "0.2"
  location-snippet: |
    opentracing_tag request-id $sent_http_x_request_id;
    opentracing_tag customer-id $sent_http_x_customer_id;
  ...
kind: ConfigMap
```

Проблема сбора трейсов



Open Telemetry Collector



OpenTelemetry Configuration

```
...
service:
  ...
  pipelines:
    traces:
      receivers:
      - opencensus
      - zipkin
      processors:
      - batch
      - resource
      - attributes
      - span/name_from_path
      - span/extension
      - span/specific
      - span/id
      - span/nodigits
      exporters:
      - jaeger
```

Pipeline обработки трейсов

OpenTelemetry Configuration

```
...
service:
  ...
  pipelines:
    traces:
      receivers:
      - opencensus
      - zipkin
      processors:
      - batch
      - resource
      - attributes
      - span/name_from_path
      - span/extension
      - span/specific
      - span/id
      - span/nodigits
      exporters:
      - jaeger
```

Opencensus - протокол, который использует Linkerd

Zipkin - протокол, который использует Nginx

OpenTelemetry Configuration

```
...
service:
  ...
  pipelines:
    traces:
      receivers:
      - opencensus
      - zipkin
      processors:
      - batch
      - resource
      - attributes
      - span/name_from_path
      - span/extension
      - span/specific
      - span/id
      - span/nodigits
      exporters:
      - jaeger
```

```
resource:
  attributes:
  - key: service.name
    from_attribute: app
    action: upsert
attributes:
  actions:
  - key: "http.url"
    pattern: ^.*//.*example\.com(?P<http_path>/.*)\?.*$
    action: extract
  - key: "http.path"
    pattern: ^(?P<http_path>/.*)\?.*$
    action: extract
  - key: "http.path"
    from_attribute: http_path
    action: upsert
  - key: "http.url"
    action: delete
  - key: "http_path"
    action: delete
```

Ставим корректное имя сервиса для linkerd-спанов

OpenTelemetry Configuration

```
...
service:
  ...
  pipelines:
    traces:
      receivers:
      - opencensus
      - zipkin
      processors:
      - batch
      - resource
      - attributes
      - span/name_from_path
      - span/extension
      - span/specific
      - span/id
      - span/nodigits
      exporters:
      - jaeger
```

```
resource:
  attributes:
  - key: service.name
    from_attribute: app
    action: upsert
attributes:
  actions:
  - key: "http.url"
    pattern: ^.*//.*example\.com(?P<http_path>/.*)\?.*$
    action: extract
  - key: "http.path"
    pattern: ^(?P<http_path>/.*)\?.*$
    action: extract
  - key: "http.path"
    from_attribute: http_path
    action: upsert
  - key: "http.url"
    action: delete
  - key: "http_path"
    action: delete
```

Обфускация урлов

OpenTelemetry Configuration

```
...
service:
  ...
  pipelines:
    traces:
      receivers:
      - opencensus
      - zipkin
      processors:
      - batch
      - resource
      - attributes
      - span/name_from_path
      - span/extension
      - span/specific
      - span/id
      - span/nodigits
      exporters:
      - jaeger
```

span/* - обработка спанов

OpenTelemetry Configuration

```
...
service:
  ...
  pipelines:
    traces:
      receivers:
      - opencensus
      - zipkin
      processors:
      - batch
      - resource
      - attributes
      - span/name_from_path
      - span/extension
      - span/specific
      - span/id
      - span/nodigits
      exporters:
      - jaeger
```

```
span/name_from_path:
  exclude:
    match_type: strict
    services: ["nginx"]
  name:
    from_attributes: [http.path]
```

```
span/nodigits:
  name:
    to_attributes:
      rules:
      - ^.*/(?P<id>\d+).*$
```

OpenTelemetry Configuration

```
...
service:
  ...
  pipelines:
    traces:
      receivers:
      - opencensus
      - zipkin
      processors:
      - batch
      - resource
      - attributes
      - span/name_from_path
      - span/extension
      - span/specific
      - span/id
      - span/nodigits
      exporters:
      - jaeger
```

Экспортируем в Jaeger

Проблемы с дефолтным Jaeger

- all-in-one (in-memory)
- ограниченное количество сэмплов
- нужно тащить БД
  - Cassandra
  - Elasticsearch

Replace Jaeger to StackDriver

```
...
service:
  ...
  pipelines:
    traces:
      receivers:
      - opencensus
      - zipkin
      processors:
      - batch
      - resource
      - attributes
      - span/name_from_path
      - span/extension
      - span/specific
      - span/id
      - span/nodigits
      exporters:
      - googlecloud
```

Экспортируем в stackdriver

Пример полученных трейсов

Selected trace details

Selected trace ID
0000000000000000011b24b5fd78ae84

No Logs found for this trace EXPAND ALL

0 10 20 30 40 50

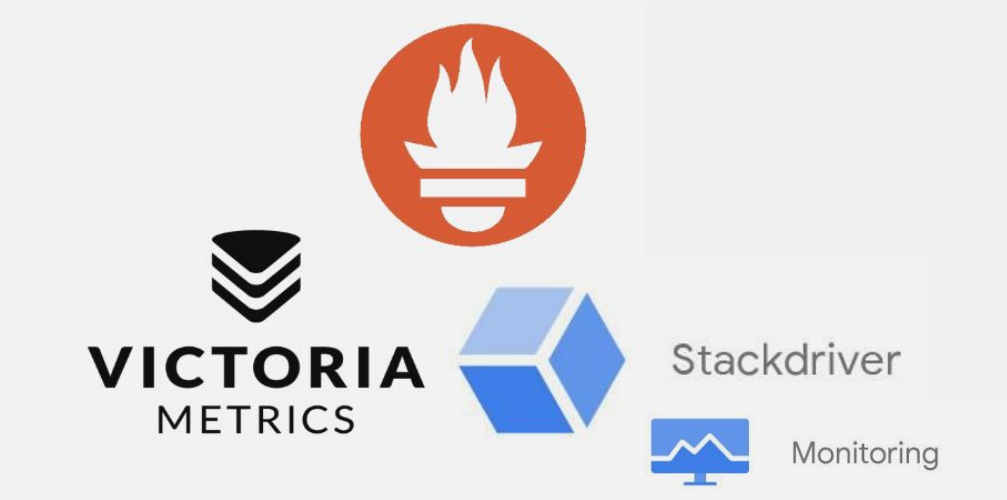
^/ (36.254 ms)
^/ (35.886 ms)
/api/customer/ (35.4 ms)
/api/customer/ (33.663 ms)
/api/customer/ (32.564 ms)
/api/customer/ (32.466 ms)

/api/customer/
Start Time: @3.13 ms
Timestamp: 2022-10-18 (11:31:24.203) +0300

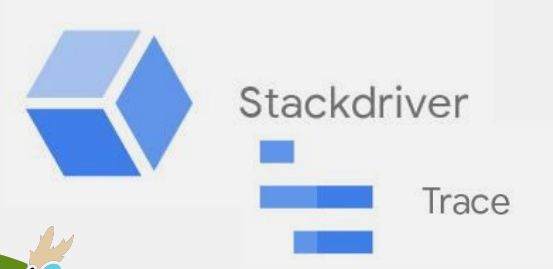
Details

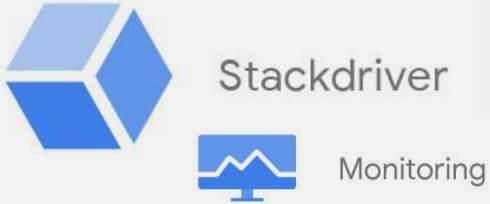
Trace logs	View |
Service	api
HTTP Method	GET
HTTP Path	/api/customer/
HTTP Status Code	200

linkerd / **linkerd-await**



linkerd / linkerd-await





Текущее состояние

Работает

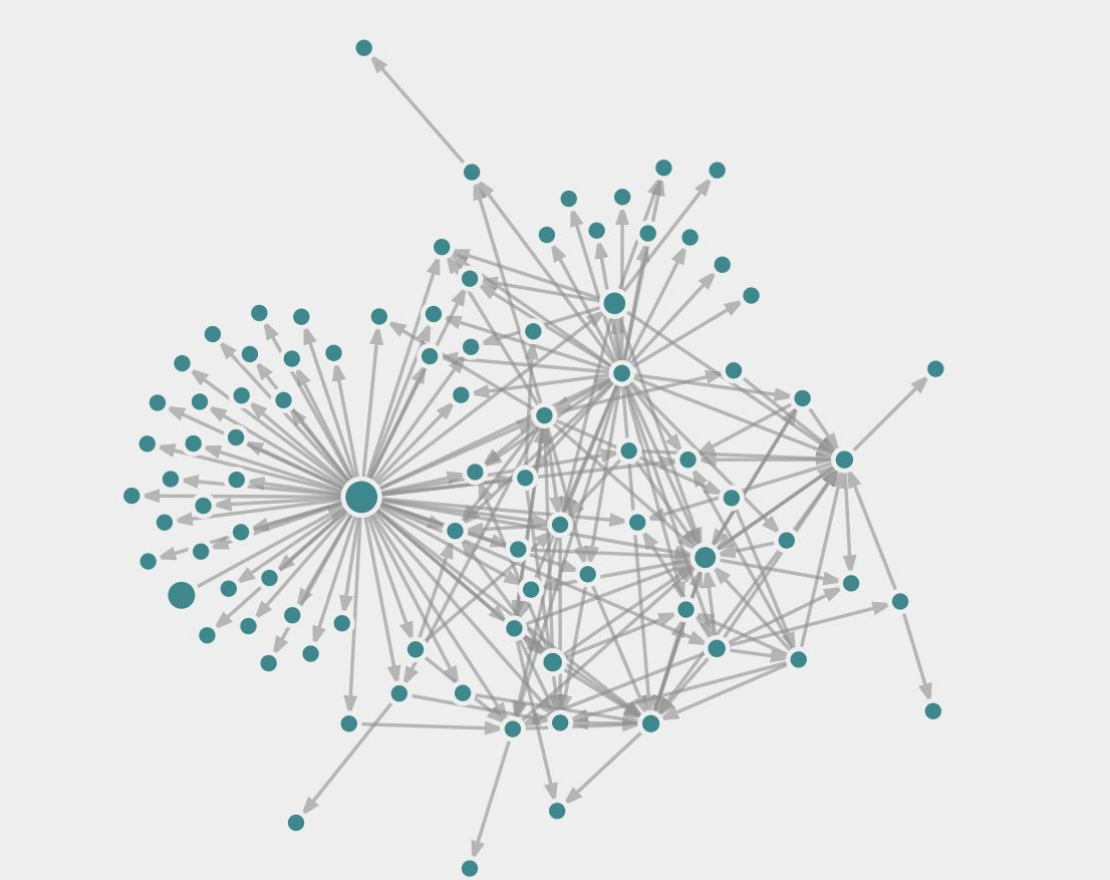
- mTLS + политики авторизации
- Load Balancing
- Traffic Split
- Ingress
- Meshed CronJobs
- Observability
- Distributed Tracing

Не работает

- Политики авторизации

Authorization Policies

Типичная MSA



Уровень защиты определяется самым слабым звеном

Backend A

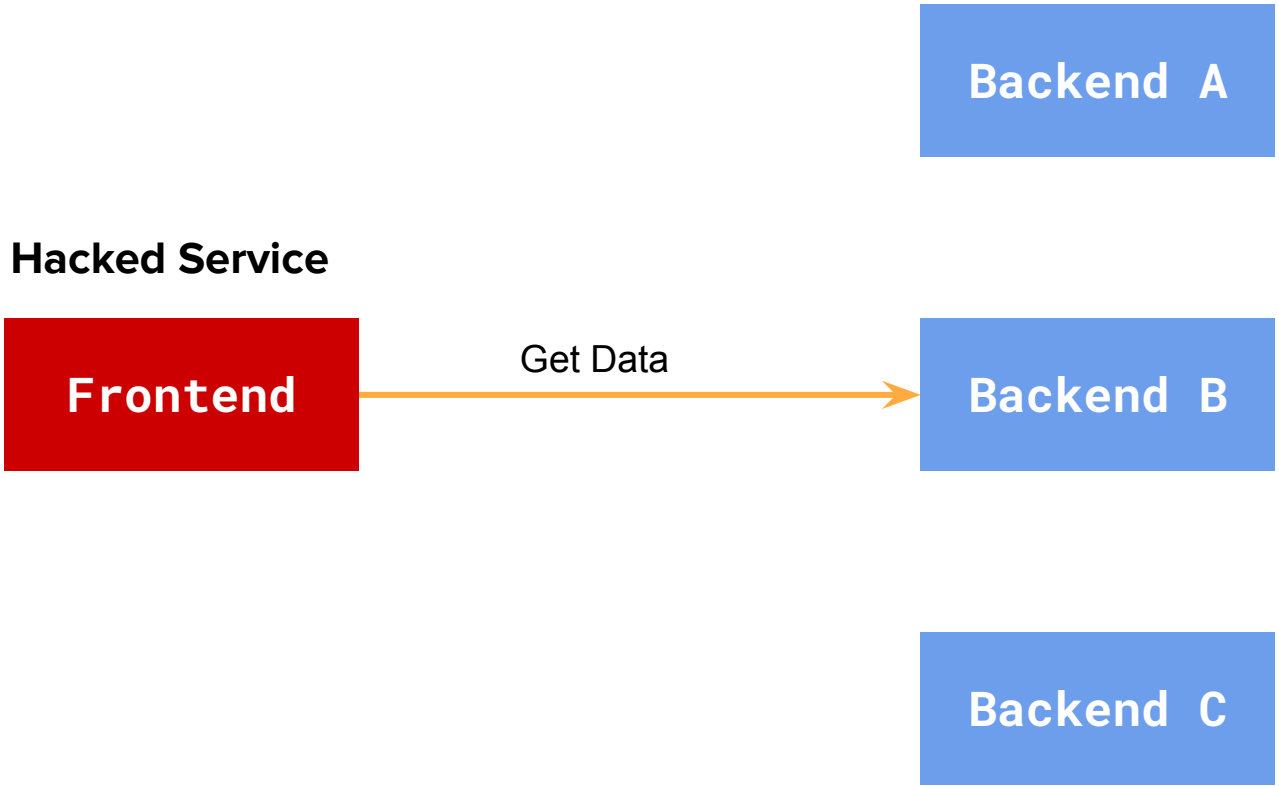
Hacked Service

Frontend

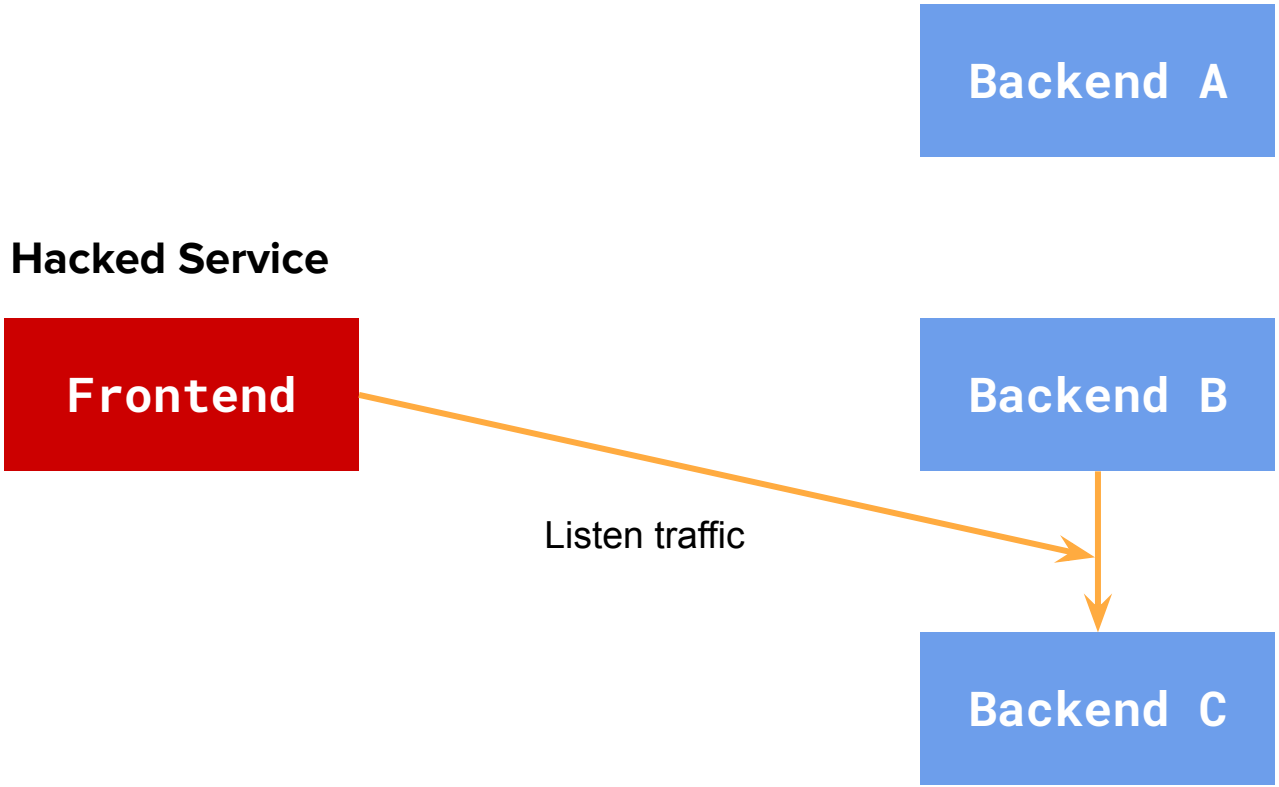
Backend B

Backend C

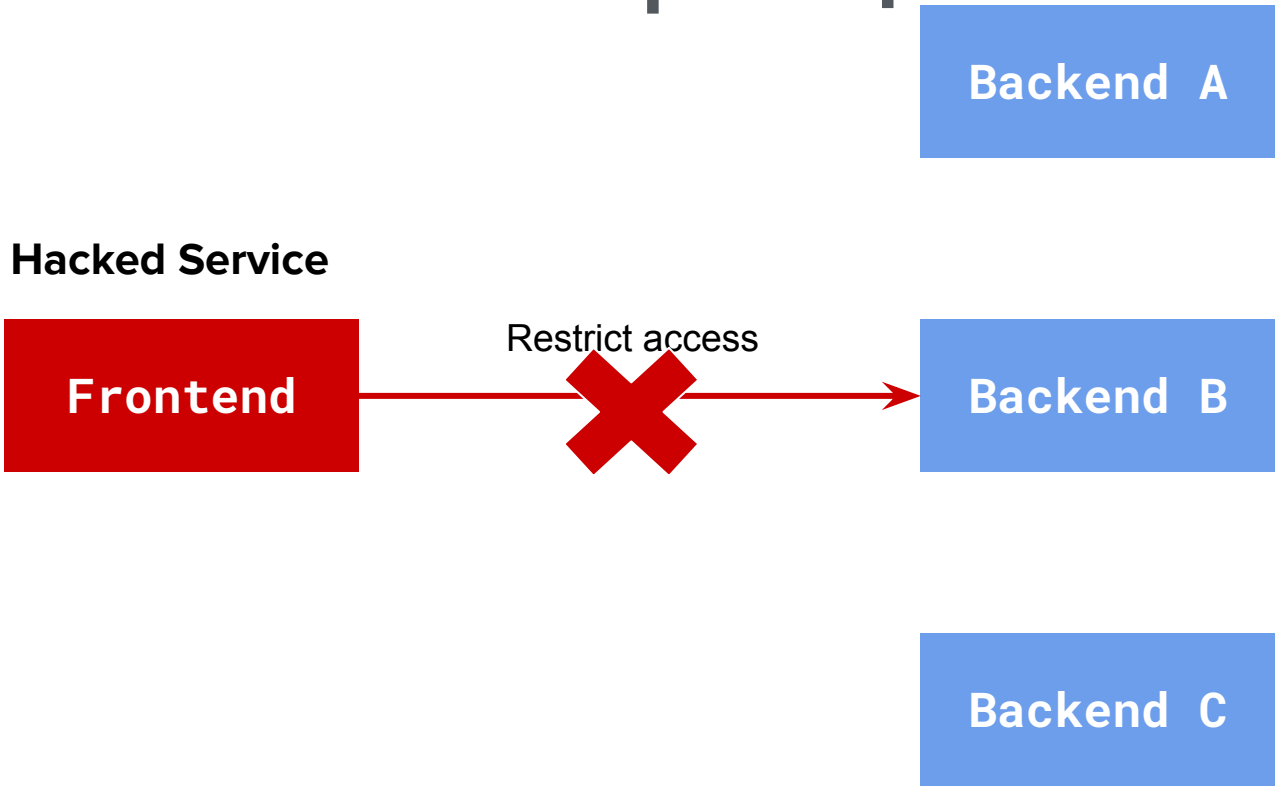
Без политик авторизации



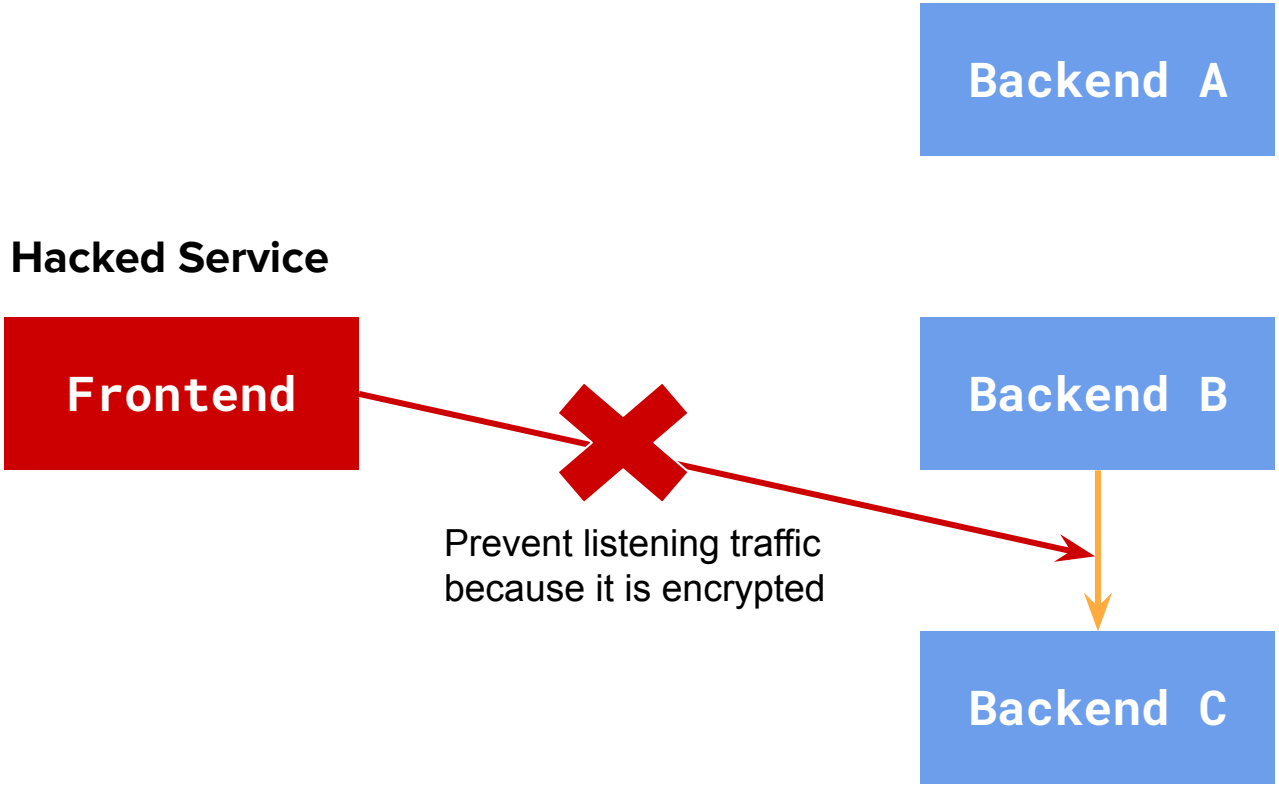
Без шифрования трафика



Ограничение доступа по политикам авторизации



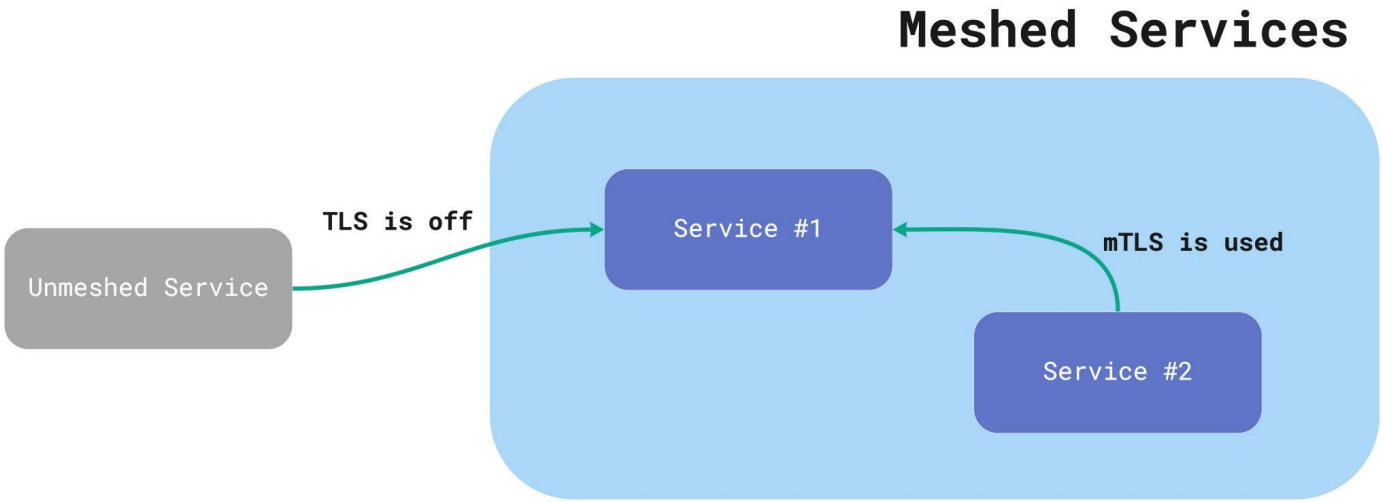
Использование mTLS



Linkerd Authorization Policies 2.11.x

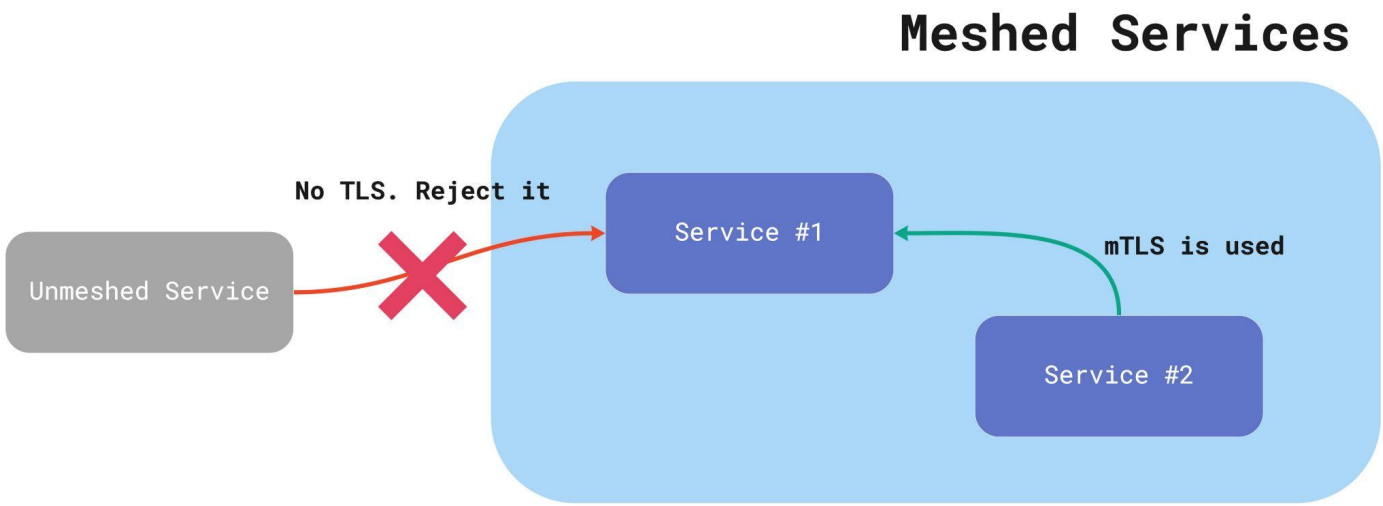
- Server
  - порт приложения
- ServerAuthorization
  - политика для Server

mTLS + Authorization. Default



mTLS - permissive
Authorization Policy - allow all

mTLS + Authorization. Strict mTLS



mTLS - strict
Authorization Policy - allow all

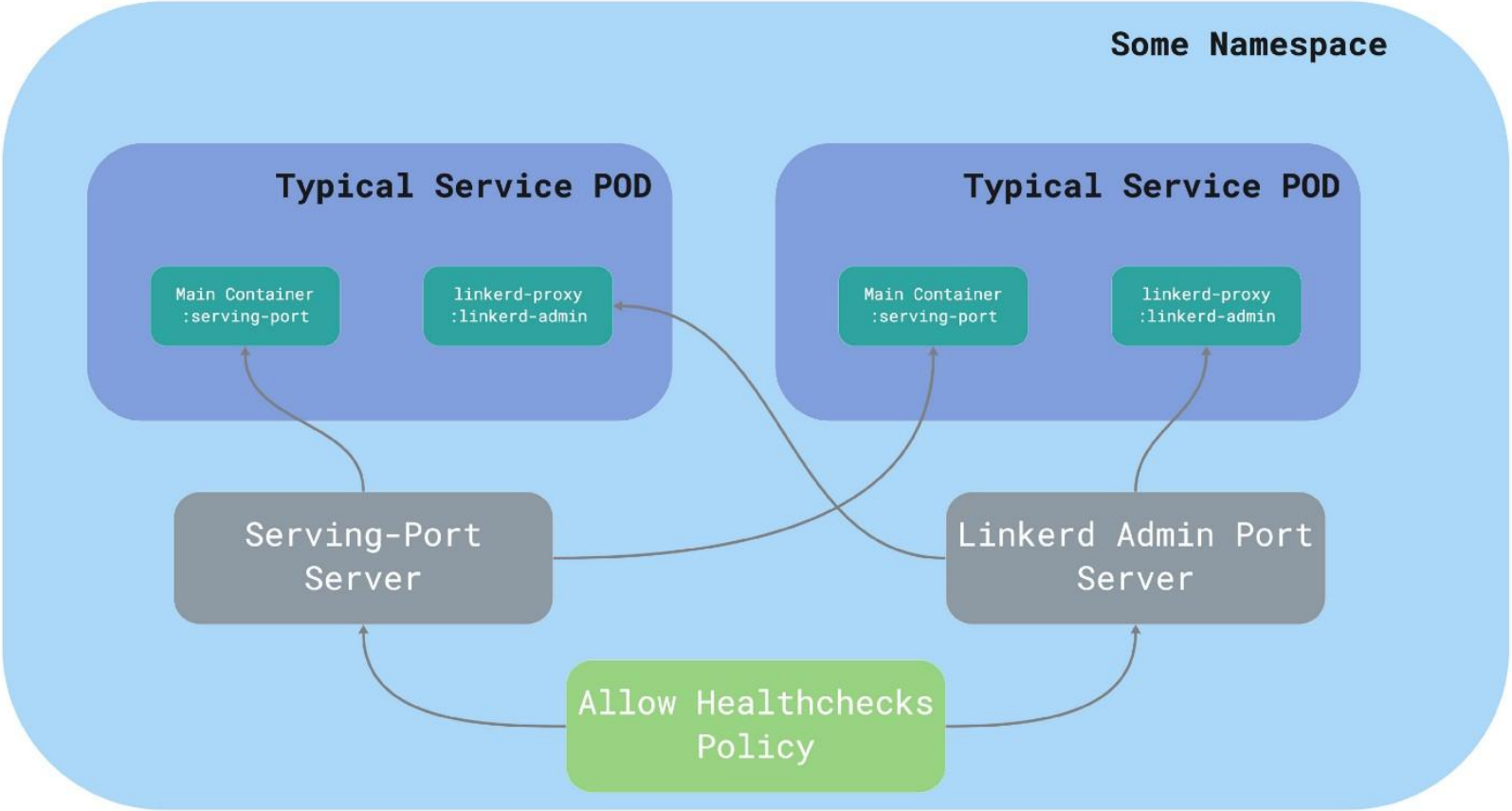
2.11.x Health Check Issue

DETAILS EVENTS LOGS YAML

Message	Reason
Readiness probe failed: HTTP probe failed with statuscode: 403	Unhealthy
Container api failed liveness probe, will be restarted	Killing
Liveness probe failed: HTTP probe failed with statuscode: 403	Unhealthy

403 - because kubelet is not authorized

Решение проблемы с хелсчеками



Решение проблемы с хелсчеками

```yaml
---
apiVersion: policy.linkerd.io/v1beta1
kind: Server
metadata:
  namespace: coolns
  name: linkerd-admin-port (serving-port for app)
  labels:
    linkerd.io/server-type: common
spec:
  podSelector:
    matchLabels:
      linkerd.io/control-plane-ns: linkerd
   port: linkerd-admin (serving-port for app)
```

Как минимум нужно сделать политики для порта приложения и для порта linkerd-proxy

Политики для kubelet

```
---
apiVersion: policy.linkerd.io/v1beta1
kind: ServerAuthorization
metadata:
  namespace: coolns
  name: allow-healthchecks
spec:
  server:
    selector:
      matchLabels:
        linkerd.io/server-type: common
  client:
    unauthenticated: true
    networks:
# kubelet CIDRs here, for example with templating
{% for n in range(x, y) %}
{% for m in range(0, 256) %}
      - cidr: 10.{{ n }}.{{ m }}.1/32
{% endfor %}
{% endfor %}
```

Матчинг по типу Server

Политики для kubelet

```
---
apiVersion: policy.linkerd.io/v1beta1
kind: ServerAuthorization
metadata:
  namespace: coolns
  name: allow-healthchecks
spec:
  server:
    selector:
      matchLabels:
        linkerd.io/server-type: common
  client:
    unauthenticated: true
    networks:
# kubelet CIDRs here, for example with templating
{% for n in range(x, y) %}
{% for m in range(0, 256) %}
      - cidr: 10.{{ n }}.{{ m }}.1/32
{% endfor %}
{% endfor %}
```

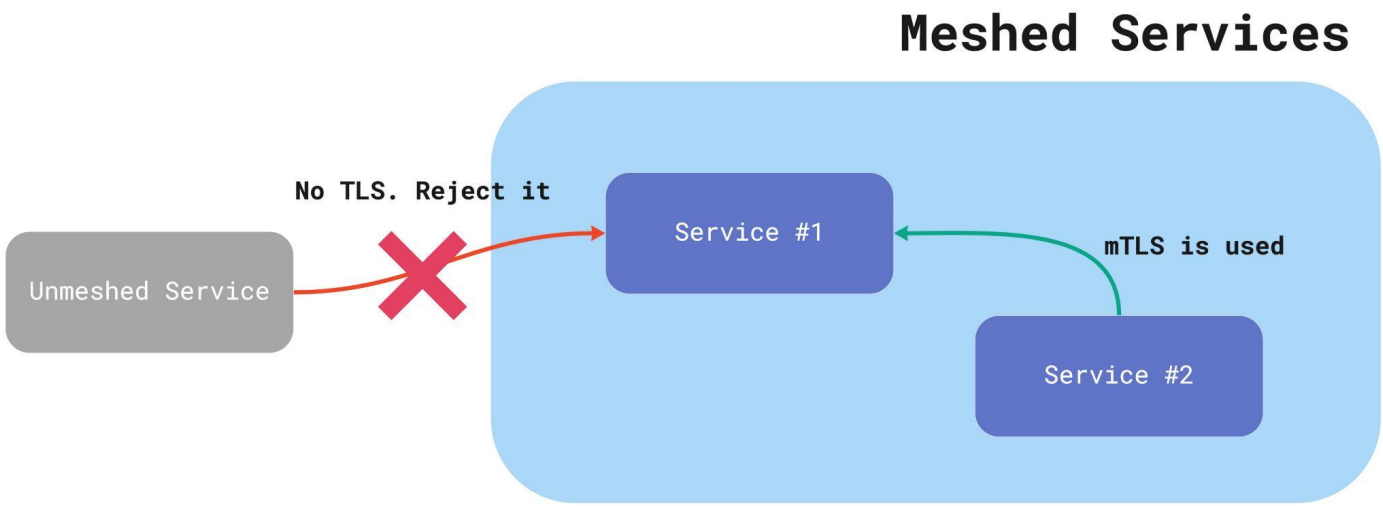
Авторизуем kubelet
GKE имеет схему вида
10.n.m.1/32

Политики авторизации

```yaml
---
apiVersion: policy.linkerd.io/v1beta1
kind: ServerAuthorization
metadata:
  namespace: coolns
  name: allow-authenticated
spec:
  server:
    selector:
      matchLabels:
        linkerd.io/server-type: common
  client:
    meshTLS:
      identities:
        - "*"
```

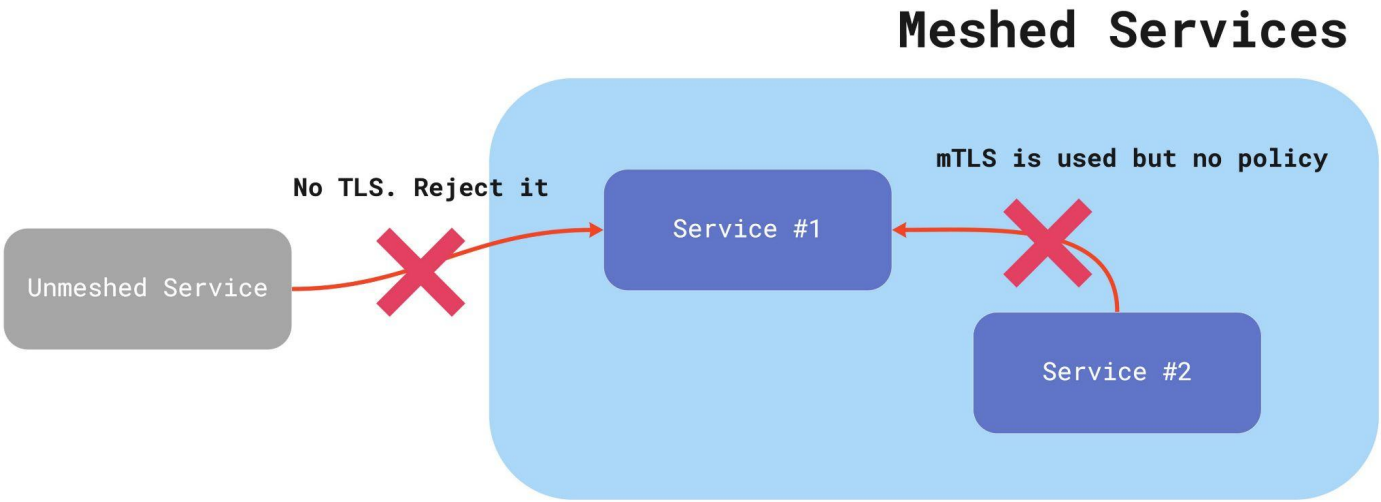
Авторизуем все mTLS коннекции

mTLS + Authorization. Strict mTLS



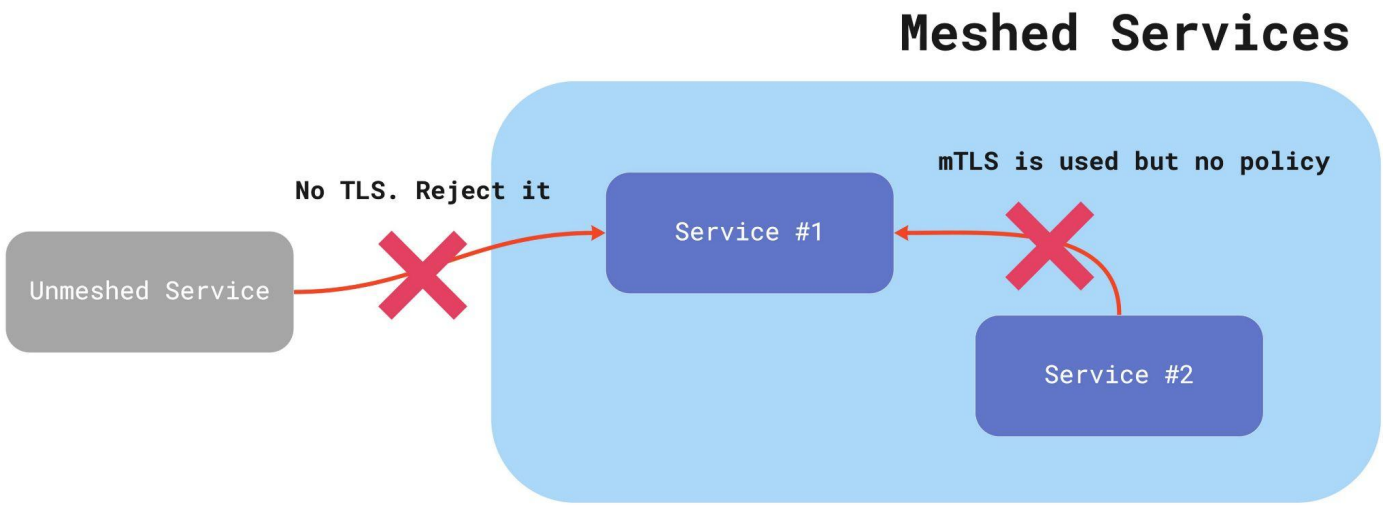
Be careful. Nginx Ingress or Prometheus could be out of mesh

mTLS + Authorization. Deny mode



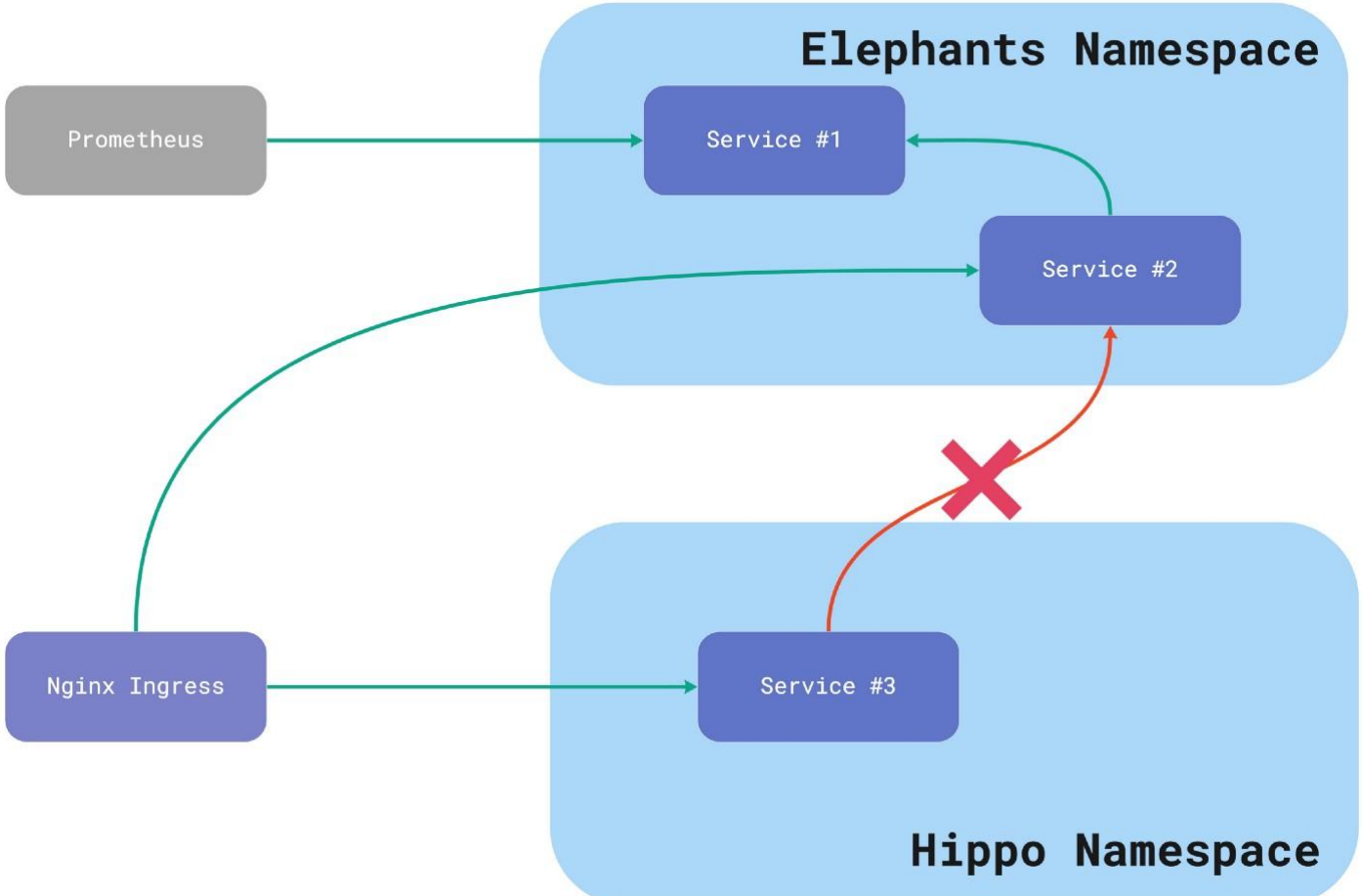
mTLS - strict
Authorization Policy - deny all

mTLS + Authorization. Deny mode

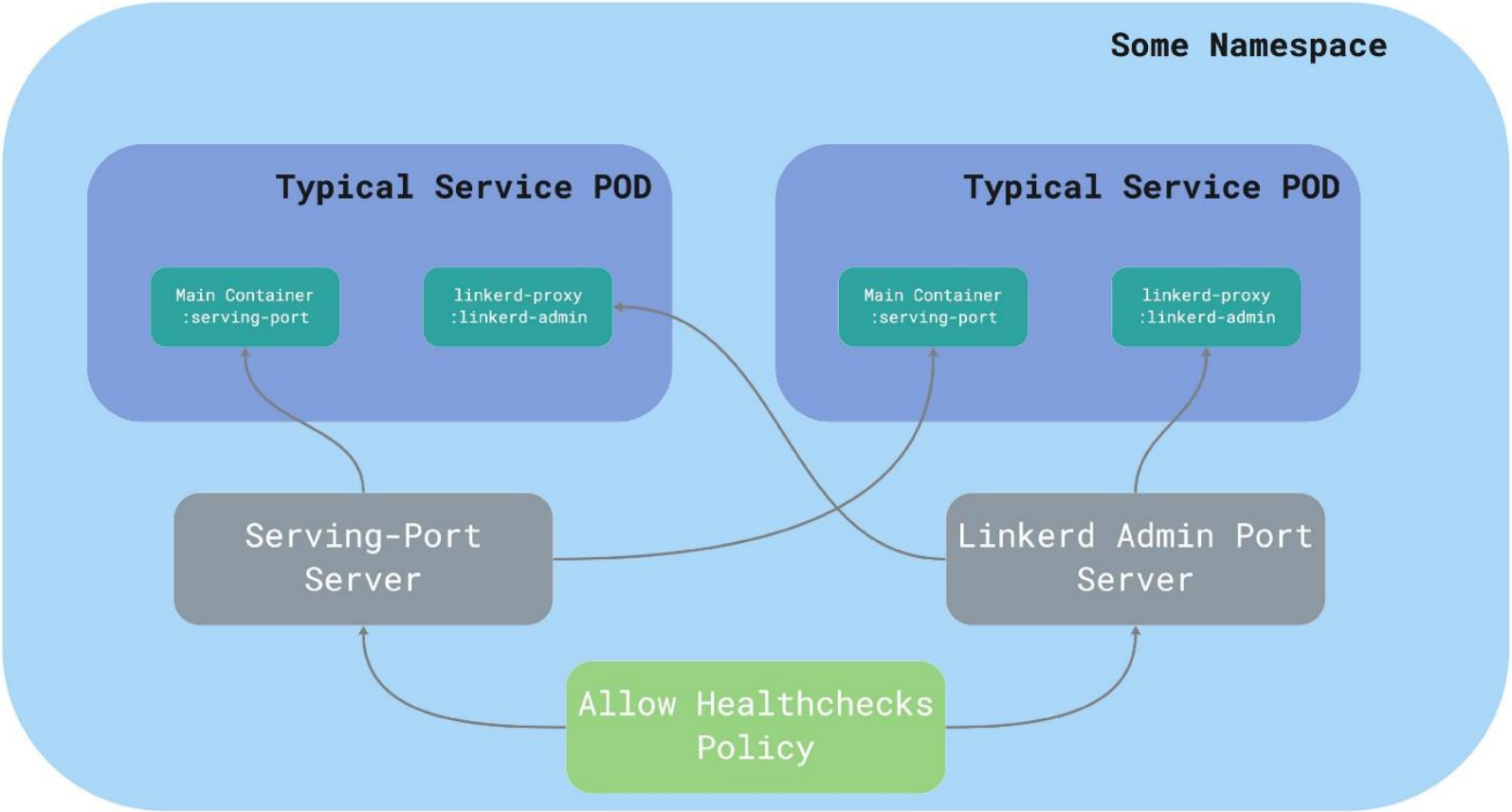


Стоит включать после полной настройки всех политик

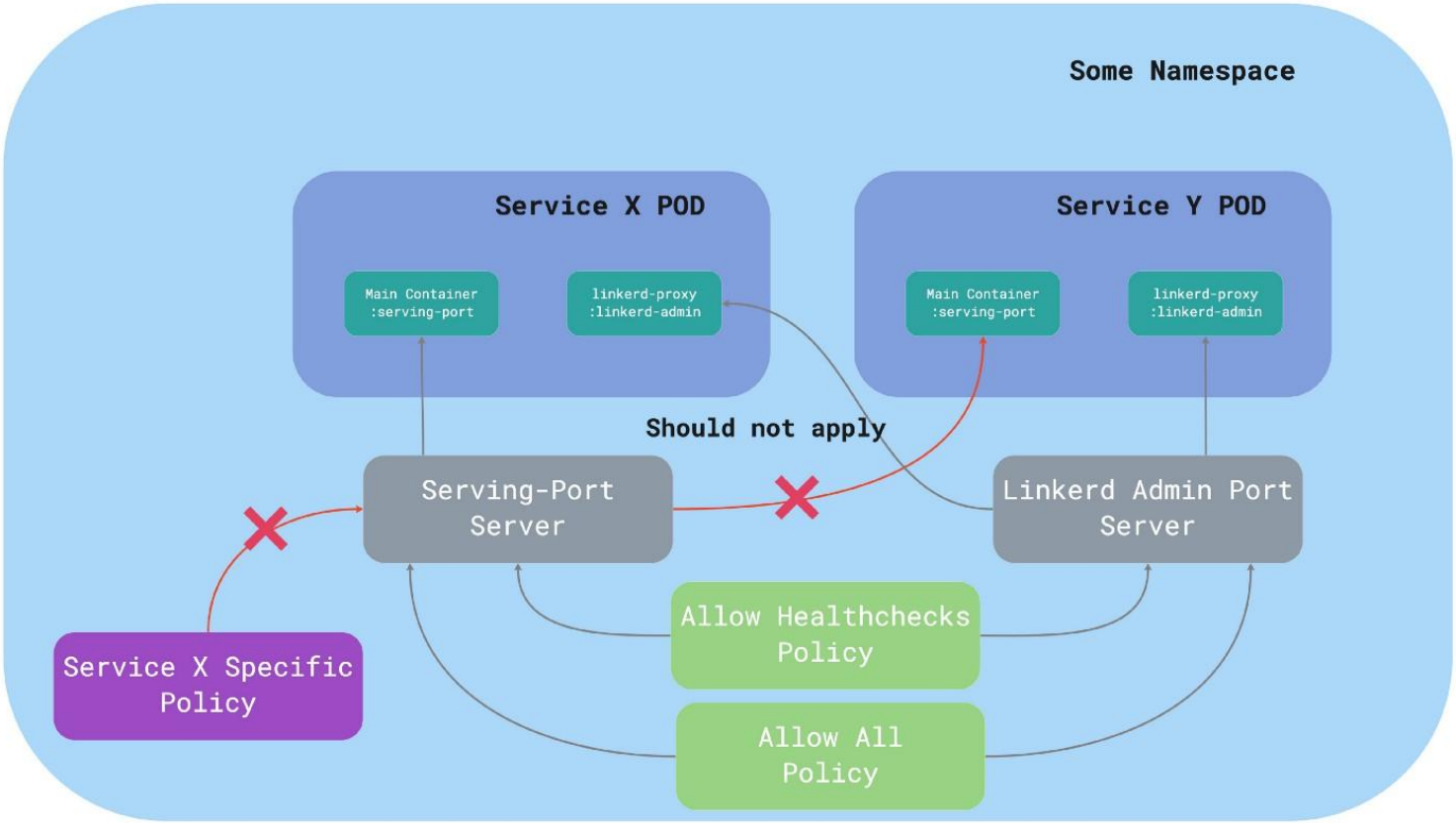
Реальный нереальный пример



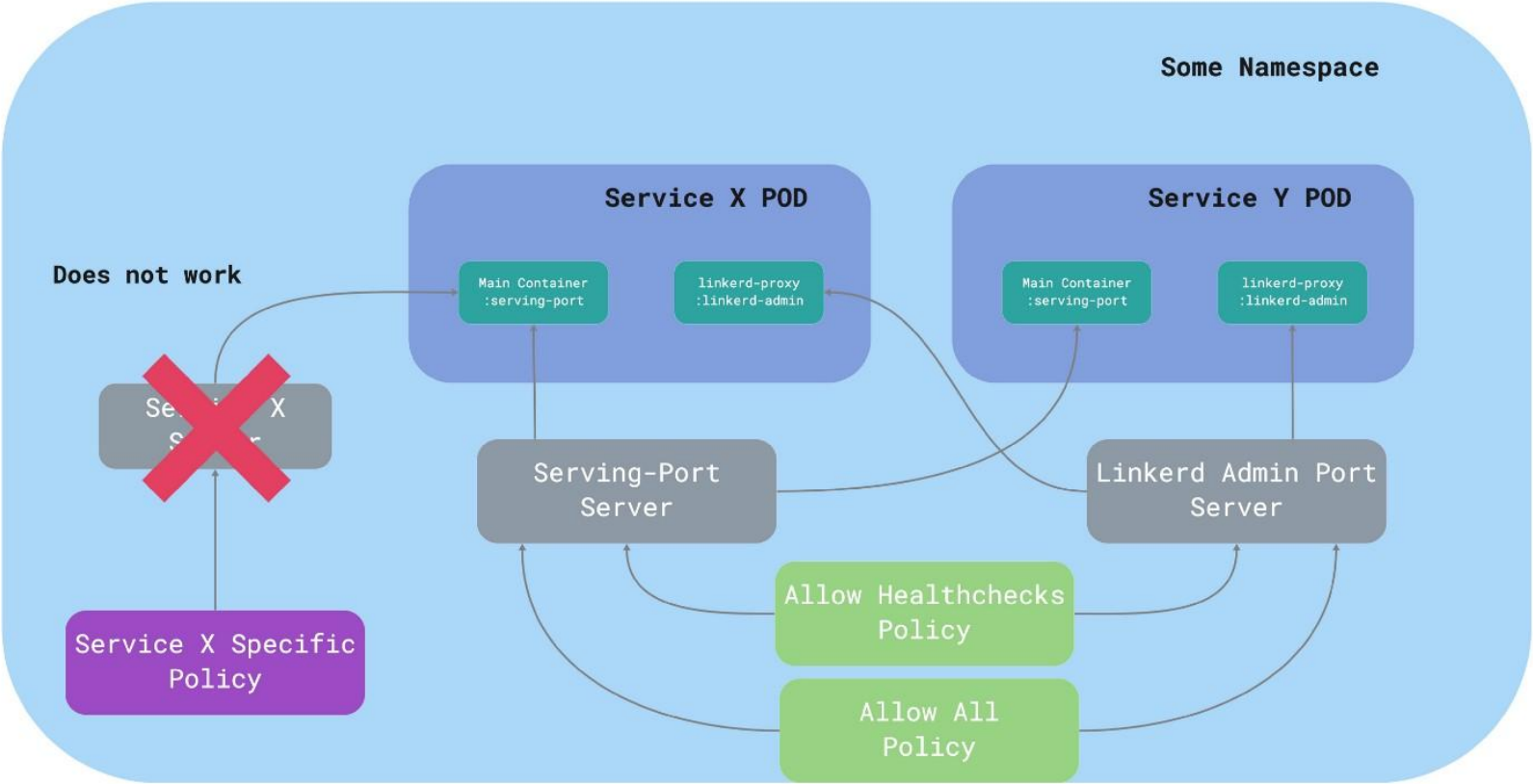
Помните нашу схему?



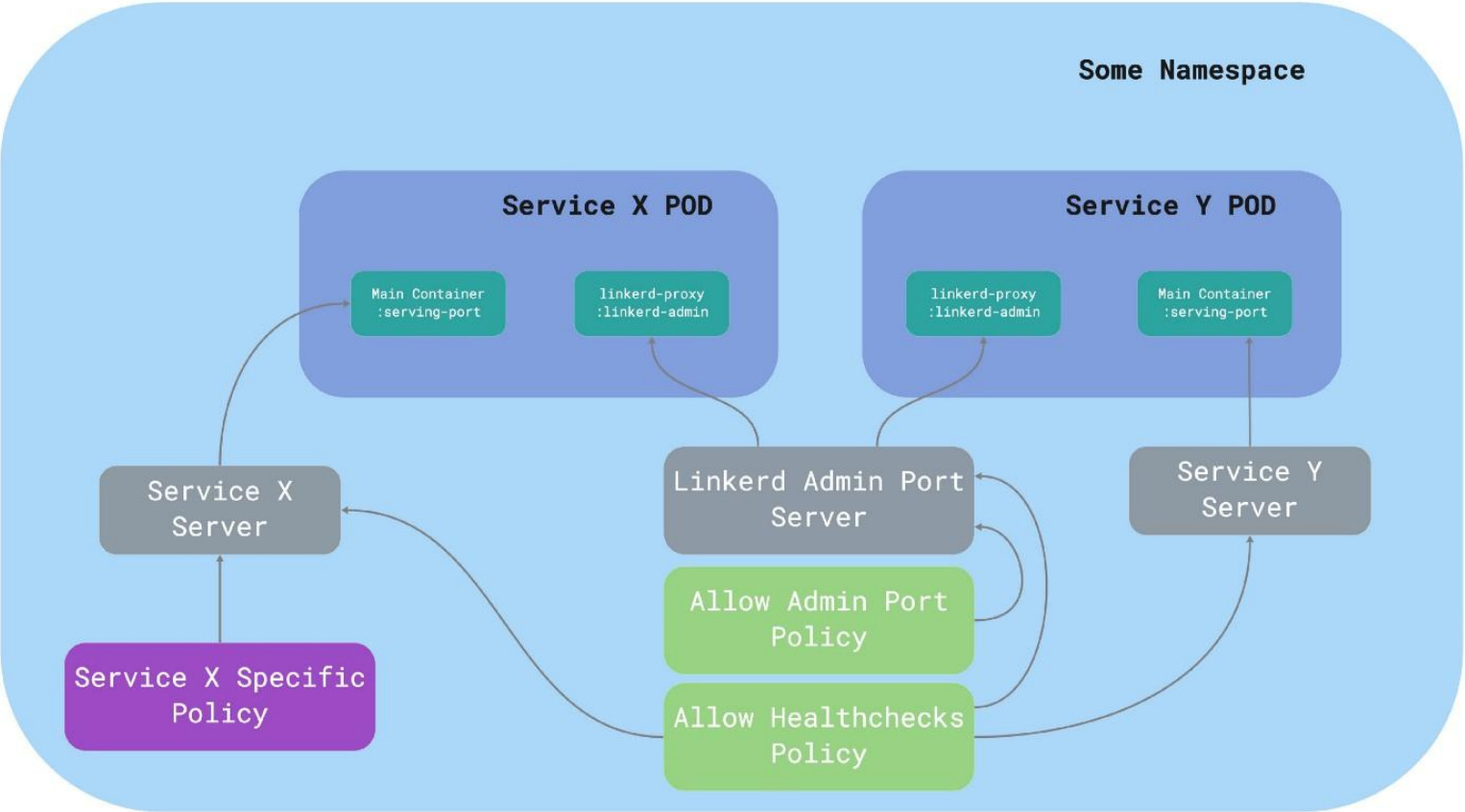
Накидываем специфичные политики



One Port - One Server



One Port - One Server



Prometheus Policies

```yaml
---
apiVersion: policy.linkerd.io/v1beta1
kind: ServerAuthorization
metadata:
  name: linkerd-prometheus-allow
spec:
  server:
    selector:
      matchLabels:
        linkerd.io/server-type: common
  client:
    meshTLS:
      identities:
        - "prometheus.linkerd-viz.serviceaccount.identity.linkerd.cluster.local"
```

Авторизуем Prometheus, чтобы получать метрики

Nginx Policies

```yaml
---
apiVersion: policy.linkerd.io/v1beta1
kind: ServerAuthorization
metadata:
  name: linkerd-nginx-allow
spec:
  server:
    selector:
      matchLabels:
        linkerd.io/nginx: enabled
  client:
    meshTLS:
      identities:
        - "*.nginx-ingress.serviceaccount.identity.linkerd.cluster.local"
```

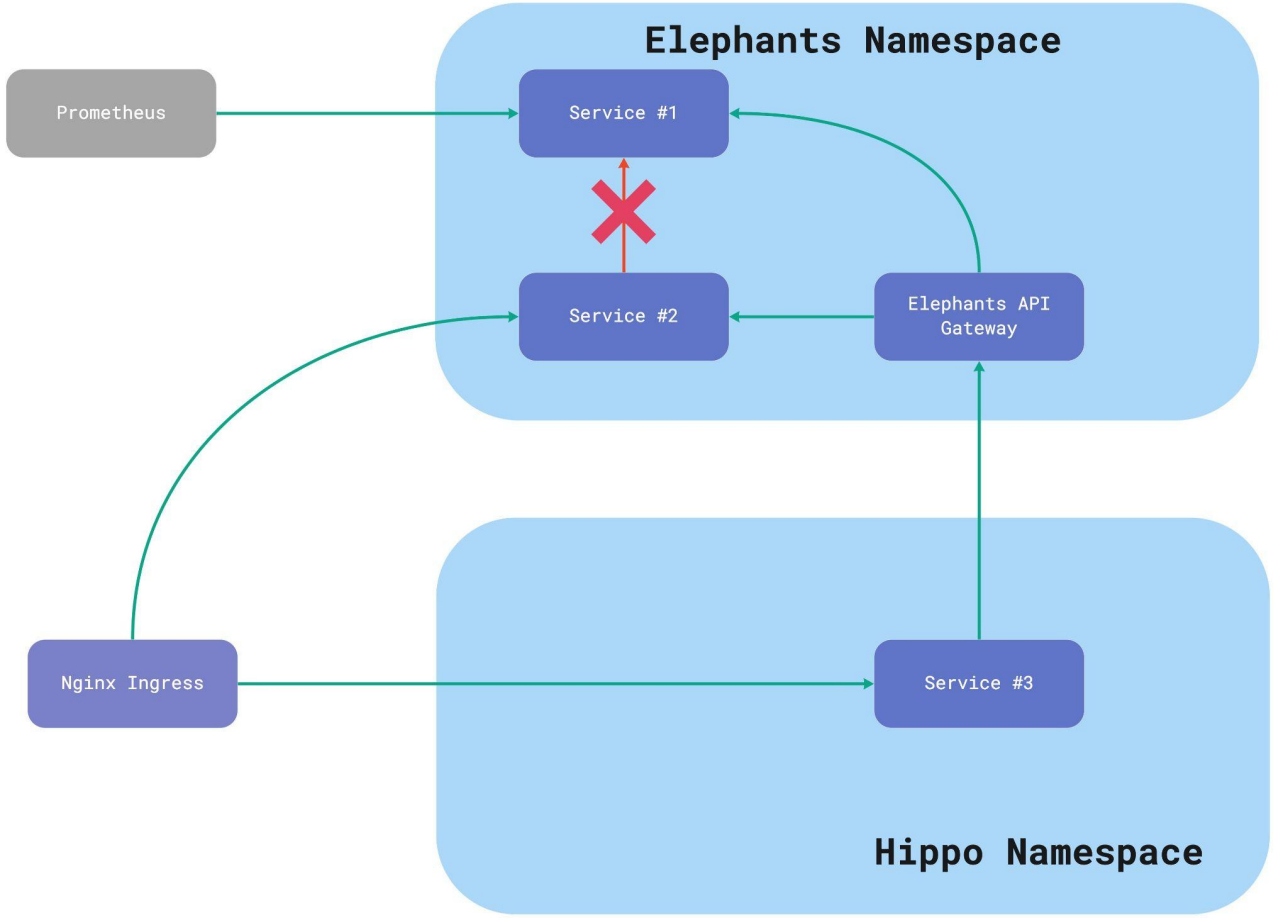
Авторизуем только те приложения, которые выставлены через ingress наружу

Namespace Allow Policies

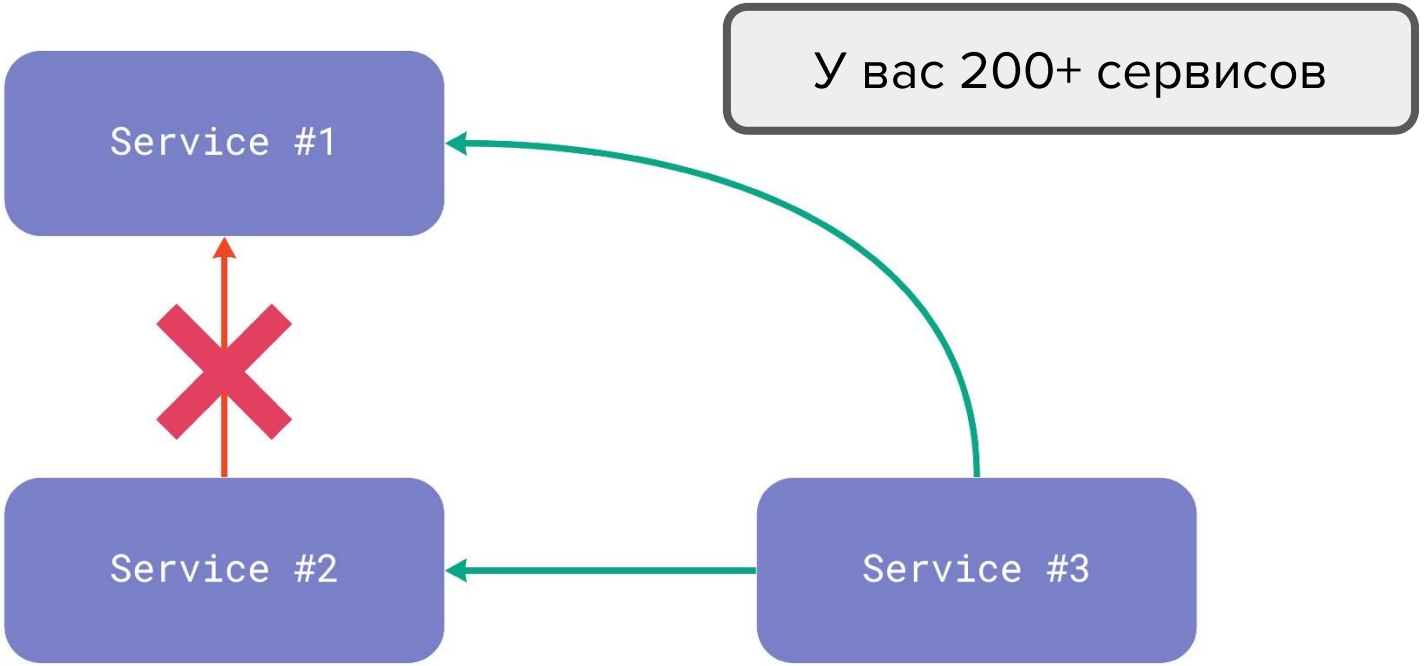
```yaml
---
apiVersion: policy.linkerd.io/v1beta1
kind: ServerAuthorization
metadata:
  name: linkerd-elephants-allow
spec:
  server:
    selector:
      matchLabels:
        linkerd.io/server-ns: elephants
  client:
    meshTLS:
      identities:
        - "*.elephants.serviceaccount.identity.linkerd.cluster.local"
```

Авторизуем запросы внутри неймспейса

Усложним наш пример



Организация политик авторизации



Helm. Explicit Configuration

```
#no one allowed
allowedServiceAccounts: []

#some services is allowed
allowedServiceAccounts:
  - namespace: elephants
    name: service3
  - namespace: hippo
    name: service1
```

Легко понять

Легко шаблонизировать

Но! Нужно идентифицировать всех клиентов

Explicit Configuration Policy

```yaml
---
apiVersion: policy.linkerd.io/v1beta1
kind: ServerAuthorization
metadata:
  name: service2-allow
spec:
  server:
    selector:
      matchLabels:
        linkerd.io/server-ns: elephants
  client:
    meshTLS:
      identities:
        - "service3.elephants.serviceaccount.identity.linkerd.cluster.local"
        - "service1.hippo.serviceaccount.identity.linkerd.cluster.local"
```

Helm. Native Configuration

```
#no dependencies
dependencies: []

#service #3 depends on #1 and #2
dependencies:
  - namespace: elephants
    name: service1
  - namespace: elephants
    name: service2
```

Более естественный способ описания

Но! Нужно решить обратную задачу (обратный граф)

Native Configuration Policy

```
{{- range .Values.approvedNamespaces }}
---
apiVersion: policy.linkerd.io/v1beta1
kind: ServerAuthorization
metadata:
  name: service3-deps
  namespace: {{ . }}
spec:
  server:
    selector:
      matchExpressions:
        - {key: "linkerd.io/authorization-policy", operator: In, values: {{ $.Values.dependencies }}}
  client:
    meshTLS:
      identities:
        - "{{ $.Values.name }}.{{ $.Values.namespace }}.serviceaccount.identity.linkerd.cluster.local"
{{ end }}
```

Создадим политику зависимостей для сервиса service3

Native Configuration Policy

```
{{- range .Values.approvedNamespaces }}
---
apiVersion: policy.linkerd.io/v1beta1
kind: ServerAuthorization
metadata:
  name: service3-deps
  namespace: {{ . }}
spec:
  server:
    selector:
      matchExpressions:
        - {key: "linkerd.io/authorization-policy", operator: In, values: {{ $.Values.dependencies }}}
  client:
    meshTLS:
      identities:
        - "{{ $.Values.name }}.{{ $.Values.namespace }}.serviceaccount.identity.linkerd.cluster.local"
{{ end }}
```

Политики работают только в рамках одного неймспейса

Native Configuration Policy

```
{{- range .Values.approvedNamespaces }}
---
apiVersion: policy.linkerd.io/v1beta1
kind: ServerAuthorization
metadata:
  name: service3-deps
  namespace: {{ . }}
spec:
  server:
    selector:
      matchExpressions:
        - {key: "linkerd.io/authorization-policy", operator: In, values: {{ $.Values.dependencies }}}
  client:
    meshTLS:
      identities:
        - "{{ $.Values.name }}.{{ $.Values.namespace }}.serviceaccount.identity.linkerd.cluster.local"
{{ end }}
```

Политика вешается
на зависимые сервисы,
а не на того, кто вызывает

Native Configuration Policy

```
{{- range .Values.approvedNamespaces }}
---
apiVersion: policy.linkerd.io/v1beta1
kind: ServerAuthorization
metadata:
  name: service3-deps
  namespace: {{ . }}
spec:
  server:
    selector:
      matchExpressions:
        - {key: "linkerd.io/authorization-policy", operator: In, values: {{ $.Values.dependencies }}}
  client:
    meshTLS:
      identities:
        - "{{ $.Values.name }}.{{ $.Values.namespace }}.serviceaccount.identity.linkerd.cluster.local"
{{ end }}
```

Авторизуем целевой сервис

Держать в голове

- **Неймспейсозависимость**
  - **политики применяются только в рамках одного неймспейса**
  - **требуется дупликация политик**
- **Один порт - один Server**
- **Платформо-специфические вещи**
  - **kubelet CIDR**
  - **Internal Load Balancers**

Держать в голове

- **Неймспейсозависимость**
  - политики применяются только в рамках одного неймспейса
  - требуется дупликация политик
- **Один порт - один Server**
- **Платформо-специфические вещи**
  - kubelet CIDR
  - Internal Load Balancers

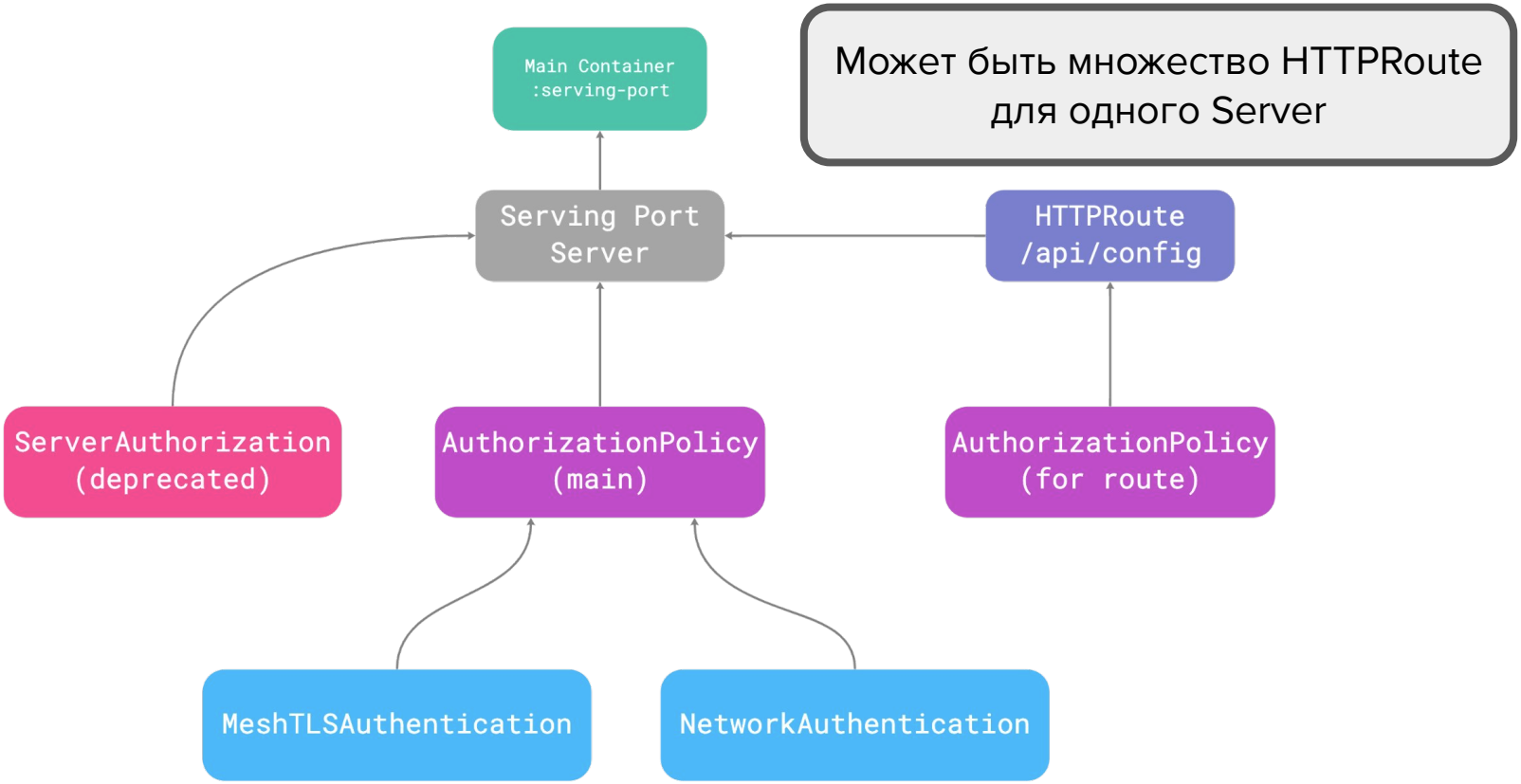
Держать в голове

- Неймспейсозависимость
  - политики применяются только в рамках одного неймспейса
  - требуется дупликация политик
- Один порт - один Server
- Платформо-специфические вещи
  - kubelet CIDR
  - Internal Load Balancers

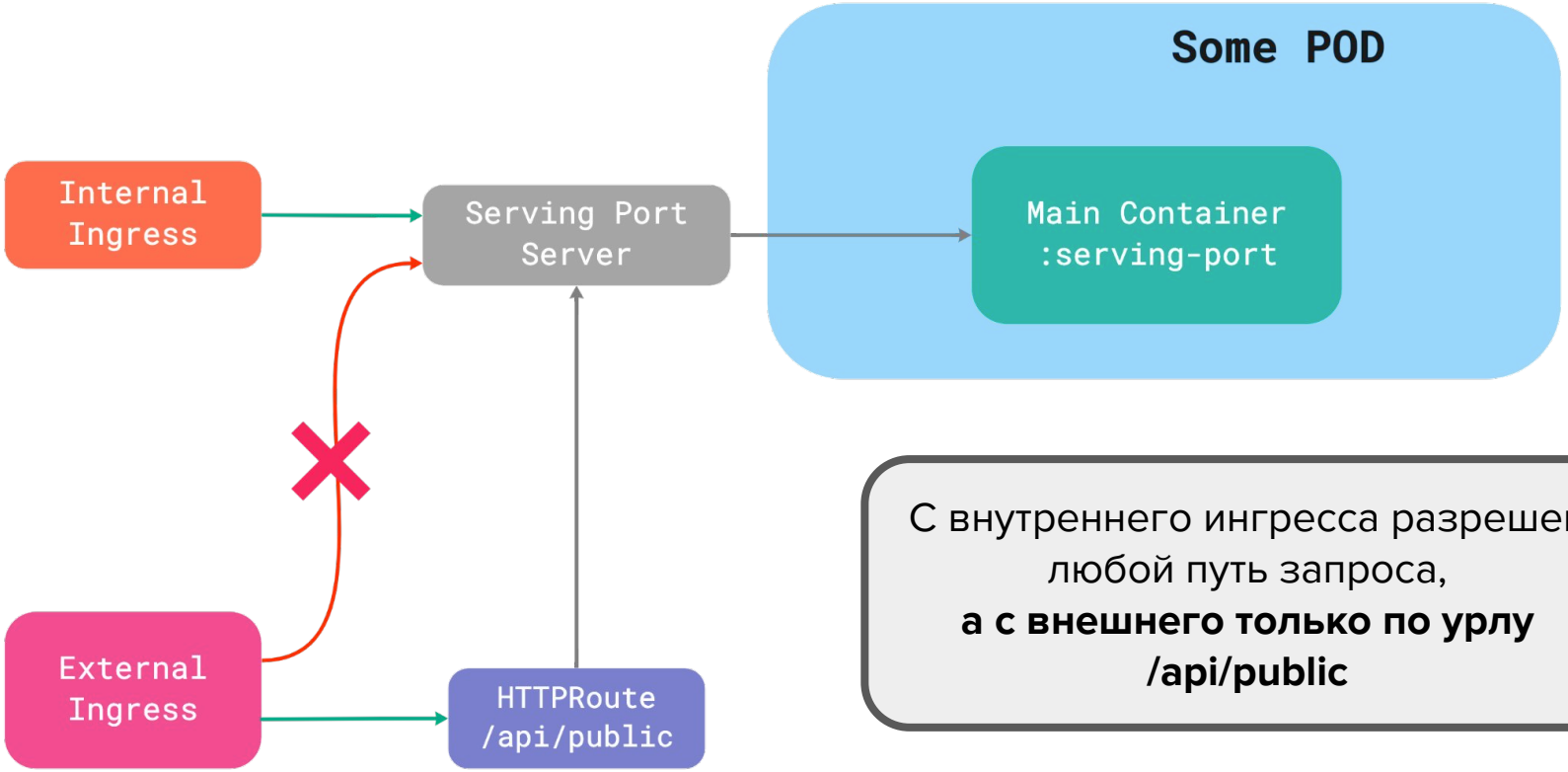
Linkerd Authorization Policies 2.12.x

- HTTPRoute
  - path на порту
- MeshTLSAuthentication
- NetworkAuthentication
- AuthorizationPolicy
  - вместо ServerAuthorization

Новая схема политик



HTTPRoute в действии



С внутреннего ингресса разрешен любой путь запроса,
а с внешнего только по урлу /api/public

2.12.x Health Check Issue

DETAILS EVENTS LOGS YAML

Message	Reason
Readiness probe failed: HTTP probe failed with statuscode: 403	Unhealthy
Container api failed liveness probe, will be restarted	Killing
Liveness probe failed: HTTP probe failed with statuscode: 403	Unhealthy

403 - because kubelet is not authorized if HTTPRoute is defined :)

Решение проблемы с хелсчеками 2

```
---
apiVersion: policy.linkerd.io/v1alpha1
kind: NetworkAuthentication
metadata:
  name: cluster-network-authn
  namespace: {{ . }}
spec:
  networks:
# kubelet CIDRs here, for example with templating
{% for n in range(x, y) %}
{% for m in range(0, 256) %}
    - cidr: 10.{{ n }}.{{ m }}.1/32
{% endfor %}
{% endfor %}
```

Авторизуем kubelet
GKE имеет схему вида
10.n.m.1/32

Решение проблемы с хелсчеками 2

```yaml
---
apiVersion: policy.linkerd.io/v1beta1
kind: HTTPRoute
metadata:
  name: website-healthcheck
spec:
  parentRefs:
  - group: policy.linkerd.io
    kind: Server
    name: website-srv
  rules:
  - matches:
    - method: GET
      path:
        type: PathPrefix
        value: /healthz
```

Добавляем эксплицитный HTTPRoute на хелсчек

Решение проблемы с хелсчеками 2

```yaml
---
apiVersion: policy.linkerd.io/v1beta1
kind: HTTPRoute
metadata:
  name: website-healthcheck
spec:
  parentRefs:
  - group: policy.linkerd.io
    kind: Server
    name: website-srv
  rules:
  - matches:
    - method: GET
      path:
        type: PathPrefix
        value: /healthz
```

Добавляем эксплицитный HTTPRoute на хелсчек

Решение проблемы с хелсчеками 2

```yaml
---
apiVersion: policy.linkerd.io/v1alpha1
kind: AuthorizationPolicy
metadata:
  name: website-healthcheck-allow
spec:
  requiredAuthenticationRefs:
  - group: policy.linkerd.io
    kind: NetworkAuthentication
    name: cluster-network-authn
  targetRef:
    group: policy.linkerd.io
    kind: HTTPRoute
    name: website-healthcheck
```

Добавляем полиси которое разрешает доступ из сети кластера к хелсчеку

Всё классно, но!

А что такое targetRef?

```yaml
---
apiVersion: policy.linkerd.io/v1alpha1
kind: AuthorizationPolicy
metadata:
  name: website-healthcheck-allow
spec:
  requiredAuthenticationRefs:
  - group: policy.linkerd.io
    kind: NetworkAuthentication
    name: cluster-network-authn
  targetRef:
    group: policy.linkerd.io
    kind: HTTPRoute
    name: website-healthcheck
```

targetRef

- HTTPRoute
- Server
- Namespace

А что такое targetRef?

```yaml
---
apiVersion: policy.linkerd.io/v1alpha1
kind: AuthorizationPolicy
metadata:
  name: website-healthcheck-allow
spec:
  requiredAuthenticationRefs:
  - group: policy.linkerd.io
    kind: NetworkAuthentication
    name: cluster-network-authn
  targetRef:
    group: policy.linkerd.io
    kind: HTTPRoute
    name: website-healthcheck
```

Нет селекторов! AAAAAA!!!!!

Helm. Native Configuration

```
#no dependencies
dependencies: []

#service #3 depends on #1 and #2
dependencies:
  - namespace: elephants
    name: service1
  - namespace: elephants
    name: service2
```

Сложно сделать.
Много дупликаций политик - сложный менеджмент

А что ещё?

Новые проблемы использования авторизационных политик

- Obsolete resources
- Observability
- Monitoring

Obsolete Resources

```
linkerd easyauth authcheck -n ns
```

https://github.com/aatarasoff/linkerd-easyauth

Observability

```
linkerd authz -n ns deployment/cool-deployment
```

```
ROUTE   SERVER       AUTHORIZATION_POLICY      SERVER_AUTHORIZATION
*       website-amb                            linkerd-healthchecks-allow
*       website-srv                            linkerd-healthchecks-allow
*       website-amb  linkerd-ingress-allow
*       website-srv  linkerd-ingress-allow
*       website-amb  linkerd-monitoring-allow
*       website-srv  linkerd-monitoring-allow
```

Observability

```
linkerd authz -n ns deployment/cool-deployment
```

```
linkerd easyauth authz -n ns deployment/cool-deployment
```

Up to 10x faster

https://github.com/aatarasoff/linkerd-easyauth

AlertManager Config. Monitoring

```
- alert: Denied tcp requests by auth policies for 1m
  annotations:
     message: "Deployment: {{ '{{' }} $labels.deployment }},
               Server: {{ '{{' }} $labels.srv_name }}"
  expr: (sum(increase(inbound_tcp_authz_deny_total[1m])) by (deployment, srv_name) > 0)
  ...


- alert: Denied http requests by auth policies for 1m
  annotations:
    message: "Deployment: {{ '{{' }} $labels.deployment }},
              Server: {{ '{{' }} $labels.srv_name }}"
  expr: (sum(increase(inbound_http_authz_deny_total[1m])) by (deployment, srv_name) > 0)
  ...
```

AlertManager Config. Monitoring

```
- alert: Denied tcp requests by auth policies for 1m
  annotations:
     message: "Deployment: {{ '{{' }} $labels.deployment }},
               Server: {{ '{{' }} $labels.srv_name }}"
  expr: (sum(increase(inbound_tcp_authz_deny_total[1m])) by (deployment, srv_name) > 0)
  ...


- alert: Denied http requests by auth policies for 1m
  annotations:
    message: "Deployment: {{ '{{' }} $labels.deployment }},
              Server: {{ '{{' }} $labels.srv_name }}"
  expr: (sum(increase(inbound_http_authz_deny_total[1m])) by (deployment, srv_name) > 0)
  ...
```

[FIRING:1] Denied http requests by auth policies for 1m

🚦 **Alerts Firing:**

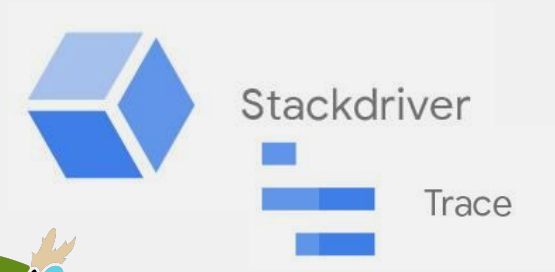
- **[WARNING]** Deployment: website-deployment, Server: website-srv

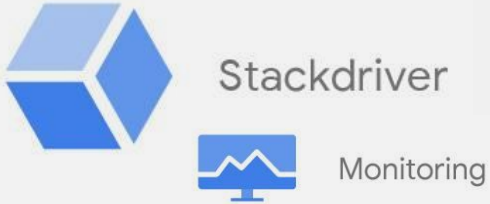
Текущее состояние

Работает

- mTLS + политики авторизации
- Load Balancing
- Traffic Split
- Ingress
- Meshed CronJobs
- Observability
- Distributed Tracing

linkerd / linkerd-await





LINKERD

linkerd / linkerd-await

aatarasoff / linkerd-easyauth

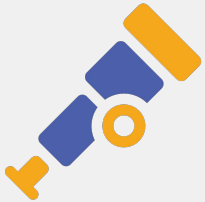
Stackdriver

Trace

VICTORIA
METRICS

Stackdriver

Monitoring



OpenTelemetry

Выводы

Выводы

Linkerd Core “просто работает”

Базовые вещи, такие как mTLS,
балансировка запросов
работают из коробки

Выводы

Observability требует изучения и экспертизы в области third-party решений

Victoria Metrics для real-time метрик и стабильности дашборда

Stackdriver или другой long-term storage для исторических метрик

Выводы

Распределенная трассировка требует доработки приложений

OpenTelemetry Collector требует настройки для красивых трейсов

Jaeger или другое решение нужно сетапить самому

Выводы

Авторизационные политики работают в простых случаях

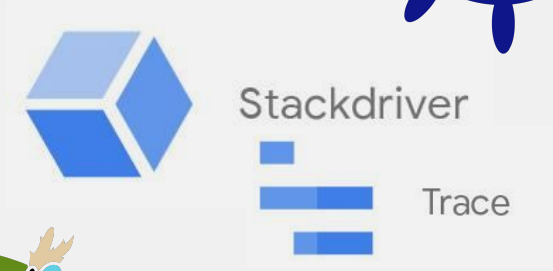
Требуют кропотливой работы по унификации и мониторинга

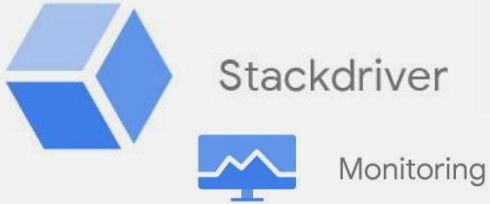
Имеют ряд ограничений и двойственности

LINKERD

linkerd / linkerd-await

aatarasoff / linkerd-easyauth





Ваши вопросы?

@aatarasoff

@aatarasoff

@aatarasoff

@aatarasoff