avito.tech

Москва — 2023

Service Mesh-авторизация с Istio и OpenPolicyAgent

Антон Губарев

Инженер PaaS

Антон Губарев

Инженер PaaS

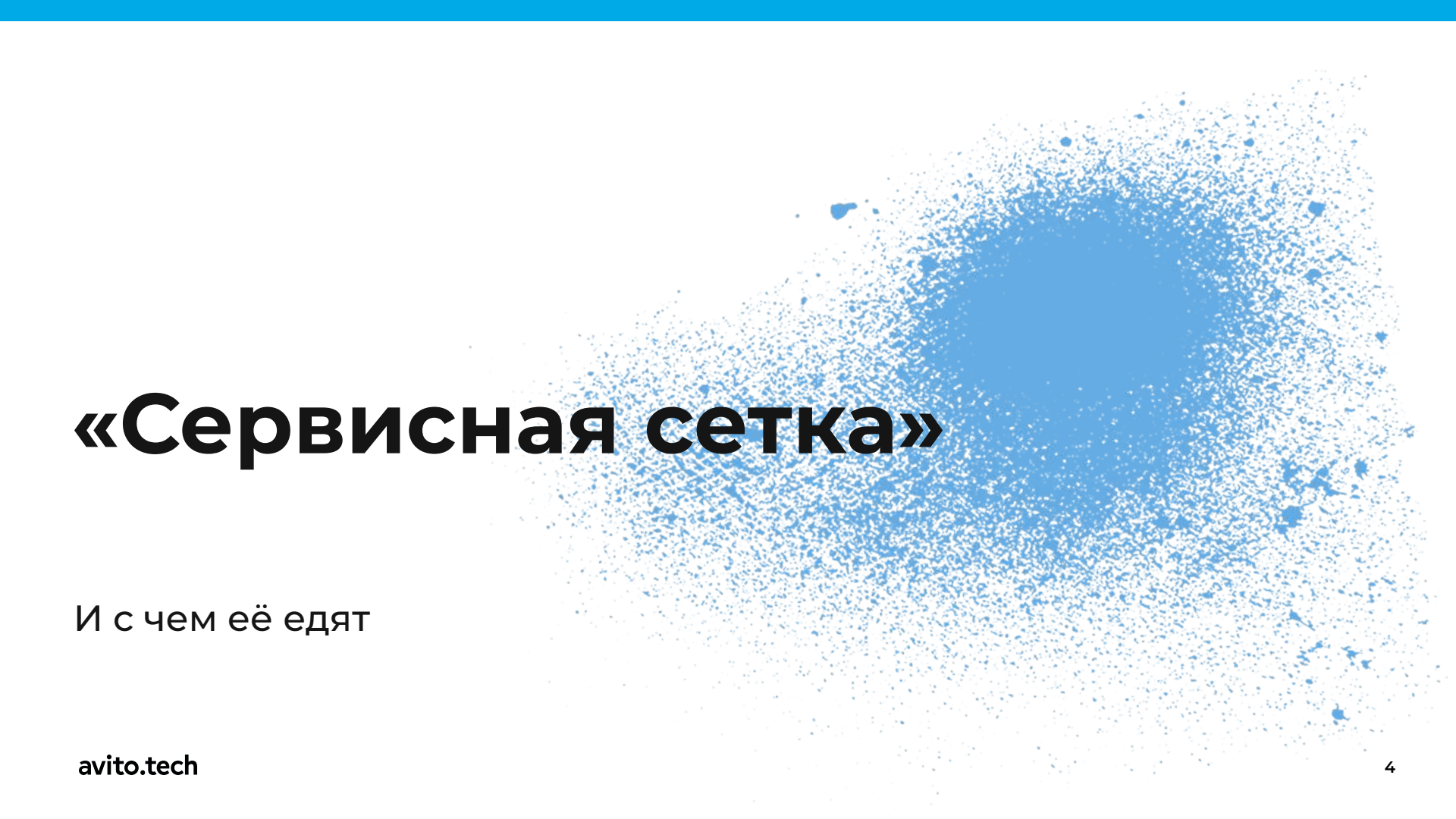
- Участвую в развитии Авито.PaaS.
- Веду проект межсервисной авторизации.
- Преподавал и руководил обучающими программами.
- Занимался архитектурой нагруженного проекта.

О чём доклад

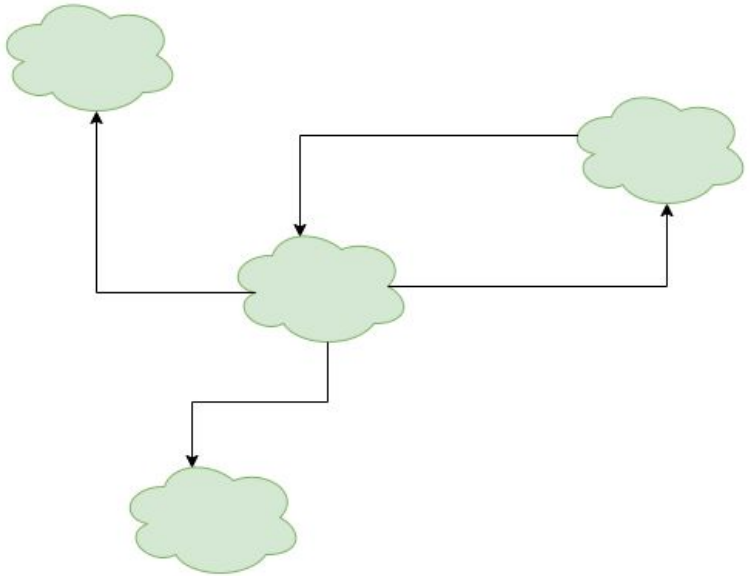
- Service Mesh
- Как он устроен в Авито.PaaS
- Авторизация
- Выводы

«Сервисная сетка»

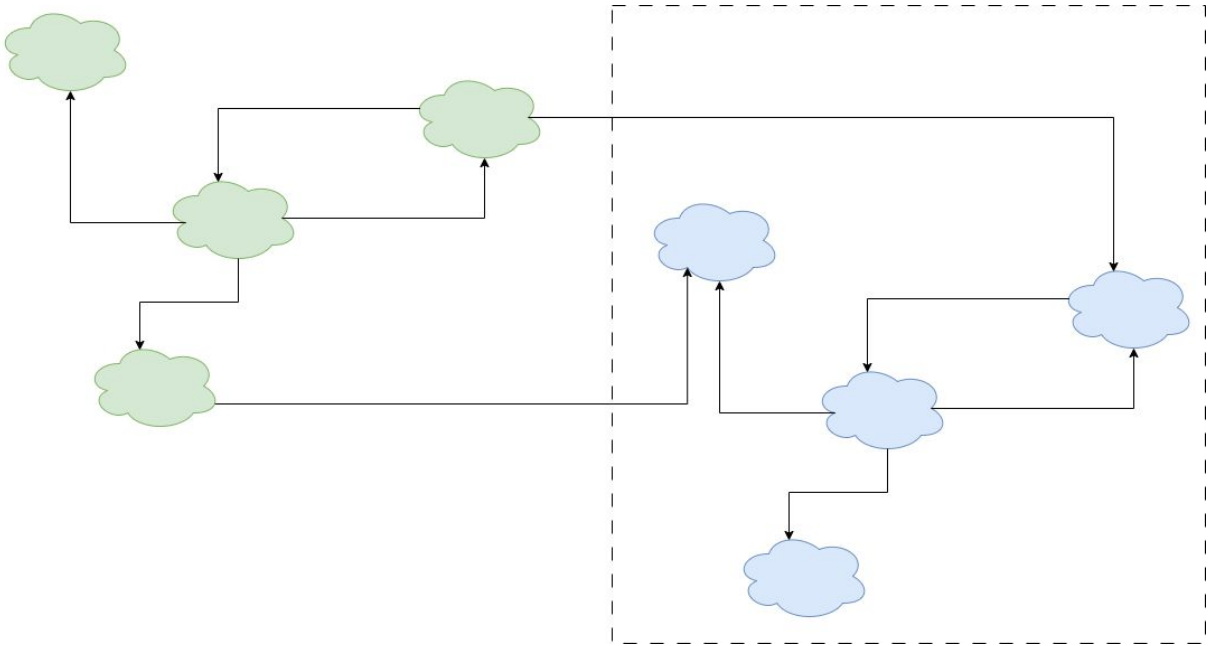
И с чем её едят



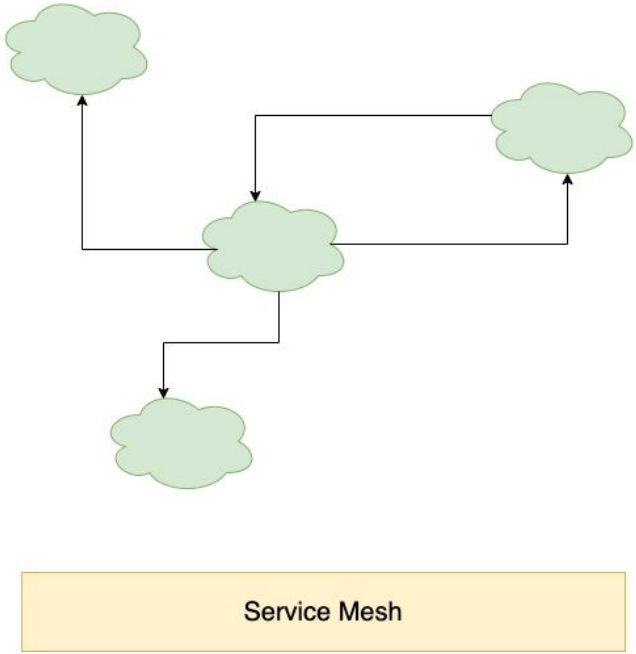
Решаемые проблемы



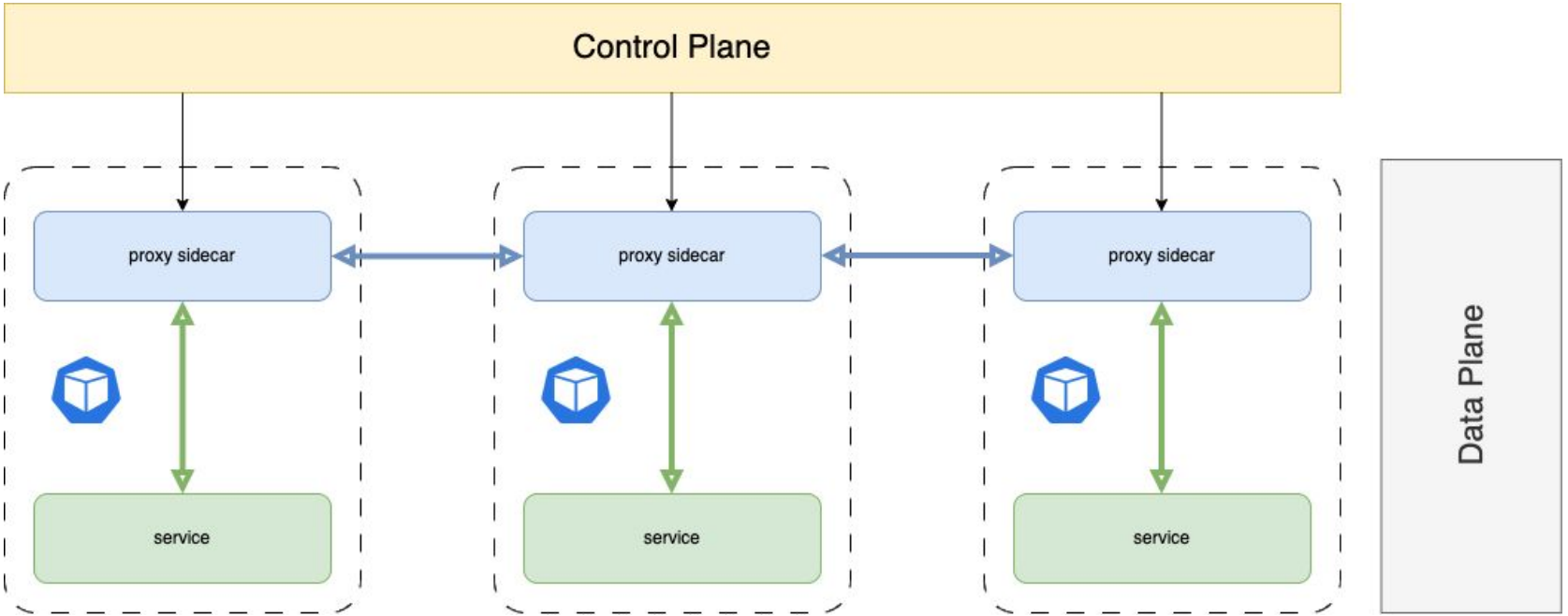
Решаемые проблемы



Дополнительный слой



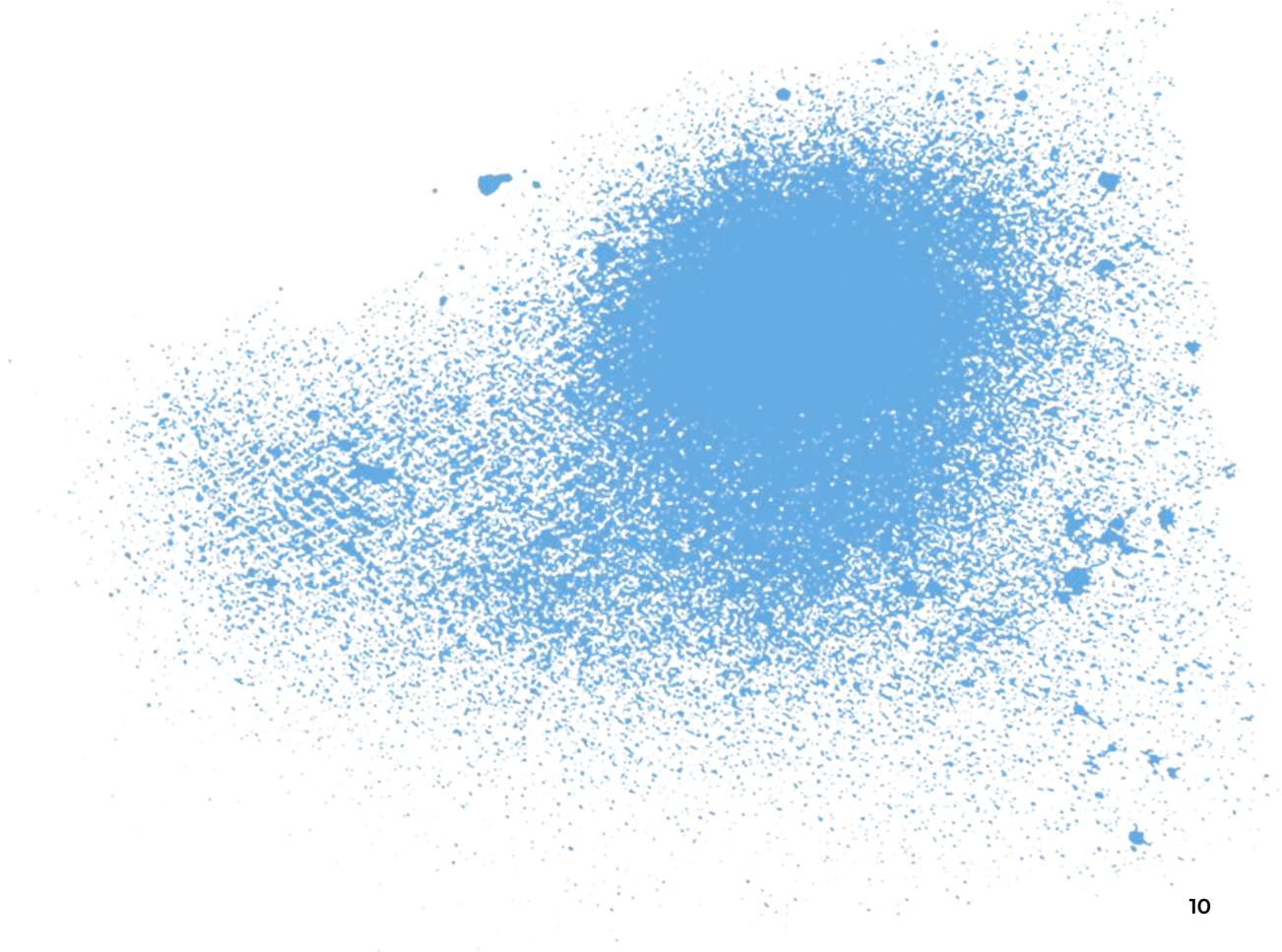
Data/Control plane



Примеры

- LinkerD
- Istio
- Traefik Mesh
- Nginx Mesh

Istio



Istio

- 33K звёзд на GitHub
- CNCF graduated
- Envoyproxy
- Google (и не только)

Envoyproxy

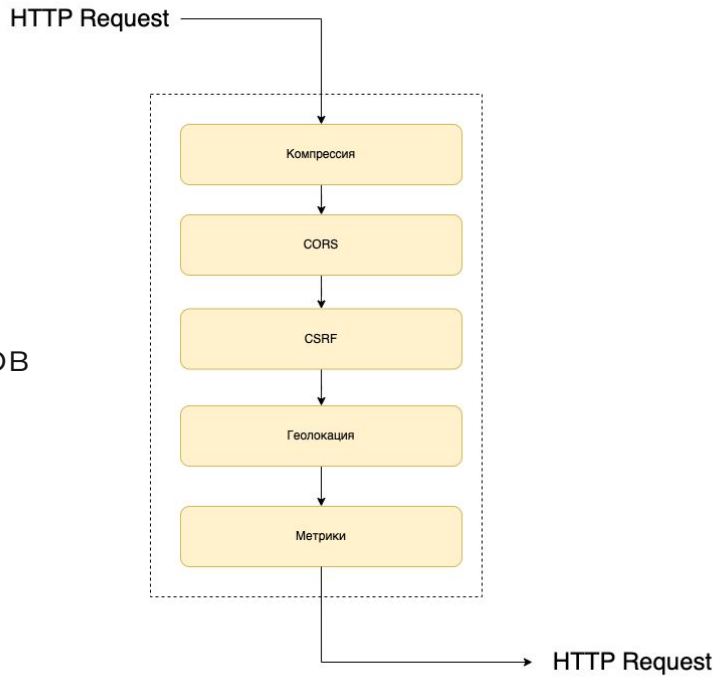
- Прокси-сервер.
- Написан на C++.
- Поддерживает различные продвинутые механики.
- Конфигурируется на лету (xDS).
- Расширяемый (c++/wasm/go/lua).

Envoyproxy

```yaml
admin:
  address:
    socket_address: { address: 127.0.0.1, port_value: 9901 }

static_resources:
  listeners:
  - name: listener_0
    address:
      socket_address: { address: 127.0.0.1, port_value: 10000 }
    filter_chains:
    - filters:
      - name: envoy.filters.network.http_connection_manager
        typed_config:
          "@type": type.googleapis.com/envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionManager
          stat_prefix: ingress_http
          codec_type: AUTO
          route_config:
            name: local_route
            virtual_hosts:
            - name: local_service
              domains: ["*"]
              routes:
              - match: { prefix: "/" }
                route: { cluster: some_service }
          http_filters:
          - name: envoy.filters.http.router
            typed_config:
              "@type": type.googleapis.com/envoy.extensions.filters.http.router.v3.Router
  clusters:
  - name: some_service
    connect_timeout: 0.25s
    type: STATIC
    lb_policy: ROUND_ROBIN
    load_assignment:
      cluster_name: some_service
      endpoints:
      - lb_endpoints:
        - endpoint:
            address:
              socket_address:
                address: 127.0.0.1
                port_value: 1234
```

Envoy

Запрос проходит
по цепочке фильтров

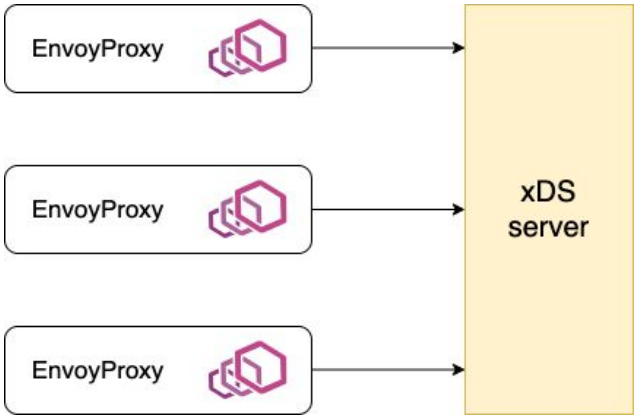


HTTP filters

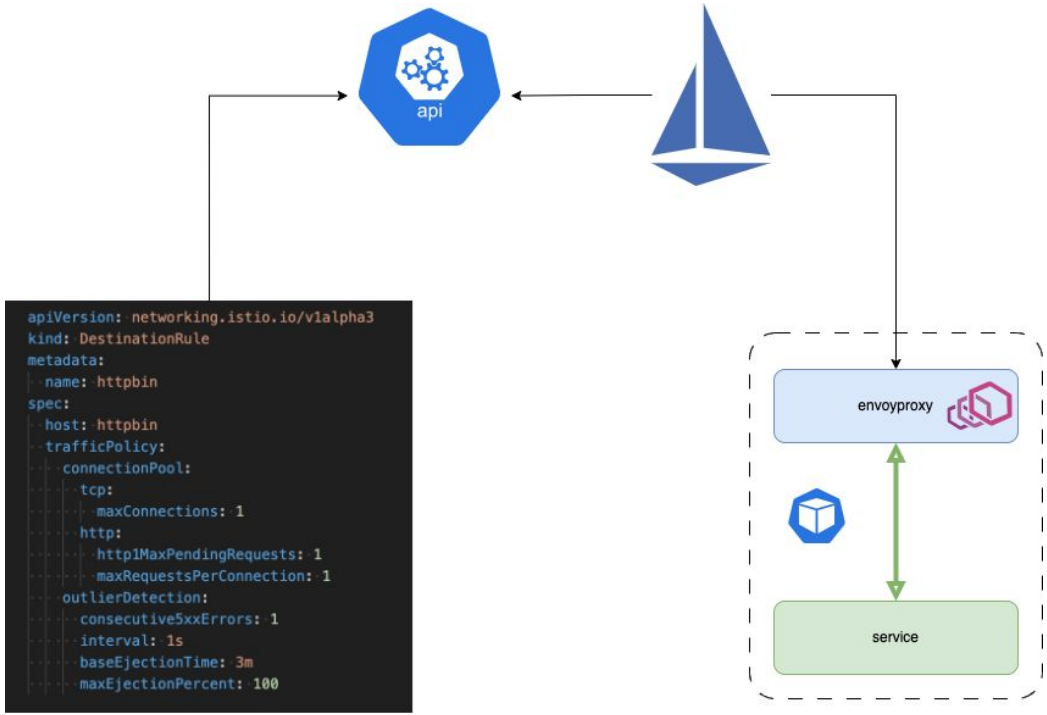
- Adaptive Concurrency
  - Overview
  - Concurrency Controllers
  - Limitations
  - Example Configuration
  - Runtime
  - Statistics
- Admission Control
  - Overview
  - Example Configuration
  - Statistics
- AWS Lambda
  - Example configuration
  - Credentials
  - Statistics
- AWS Request Signing
  - Example configuration
  - Credentials
  - Statistics
- Bandwidth limit
  - Example configuration
  - Statistics
  - Runtime
- Buffer
  - Per-Route Configuration
- Cache filter
  - Example configuration

xDS

- LDS (Listeners)
- VHDS (Virtual Hosts)
- RDS (Route)
- CDS (Cluster)
- EDS (Endpoint)



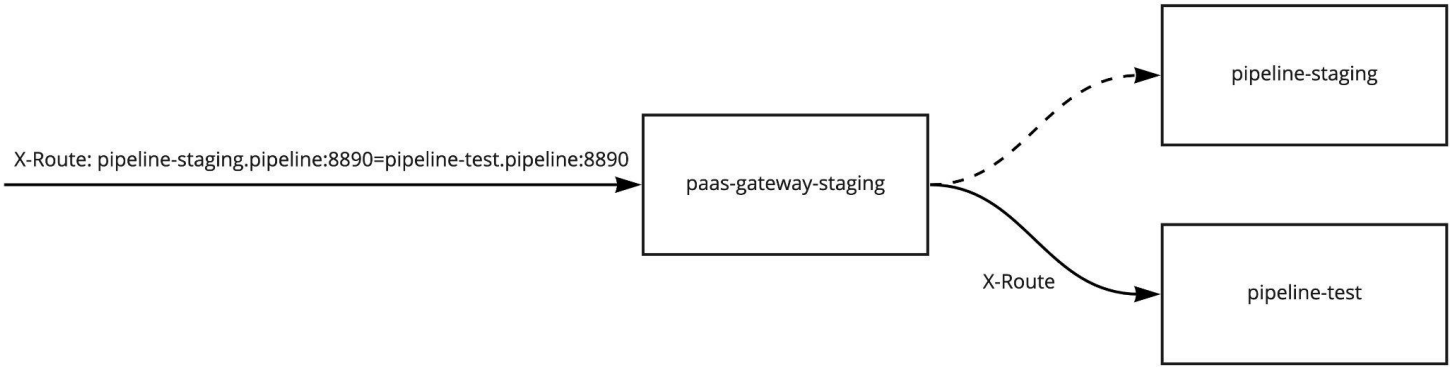
Istio + Envoy



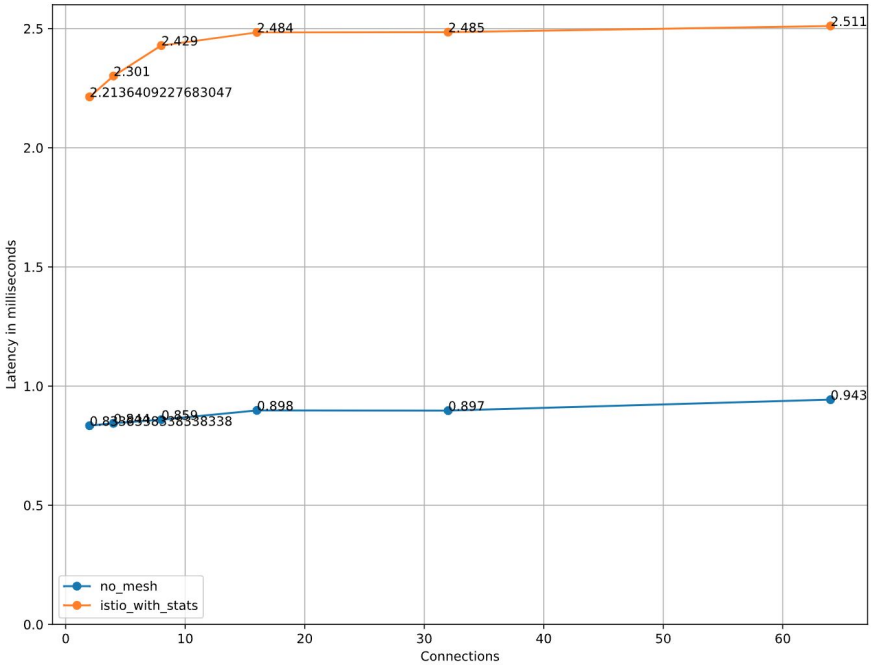

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: httpbin
spec:
  host: httpbin
  trafficPolicy:
    connectionPool:
      tcp:
        maxConnections: 1
      http:
        http1MaxPendingRequests: 1
        maxRequestsPerConnection: 1
    outlierDetection:
      consecutive5xxErrors: 1
      interval: 1s
      baseEjectionTime: 3m
      maxEjectionPercent: 100
```

Пример

- Rate limiter
- Трейсинг
- mTLS
- X-Route



Latency (99.9)


2.5
2.0
1.5
1.0
0.5
0.0
Latency in milliseconds
0
10
20
30
40
50
60
Connections
2.511
2.484
2.485
2.429
2.301
2.2136409227683047
0.943
0.898
0.897
0.859
no_mesh
istio_with_stats

Межсервисная авторизация

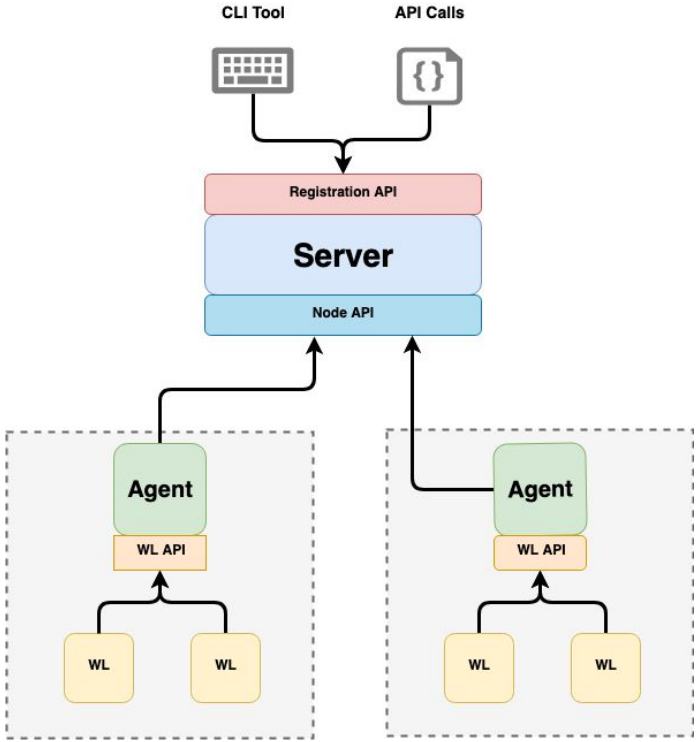
Как пример возможностей Service Mesh

brief

```
service "opa-istio-test"

idempotent rpc sum (SumIn) SumOut `A sum method`
rpc countWords (CountWordsIn) CountWordsOut `Counts words in text`

message SumIn {
    a    int    `A first number`
    b    int    `A second number`
}

message SumOut {
    sum    int    `A sum of the numbers`
}

message CountWordsIn {
    text string
}

message CountWordsOut {
    words int
}
```

mTLS

- Spiffe
- Spire

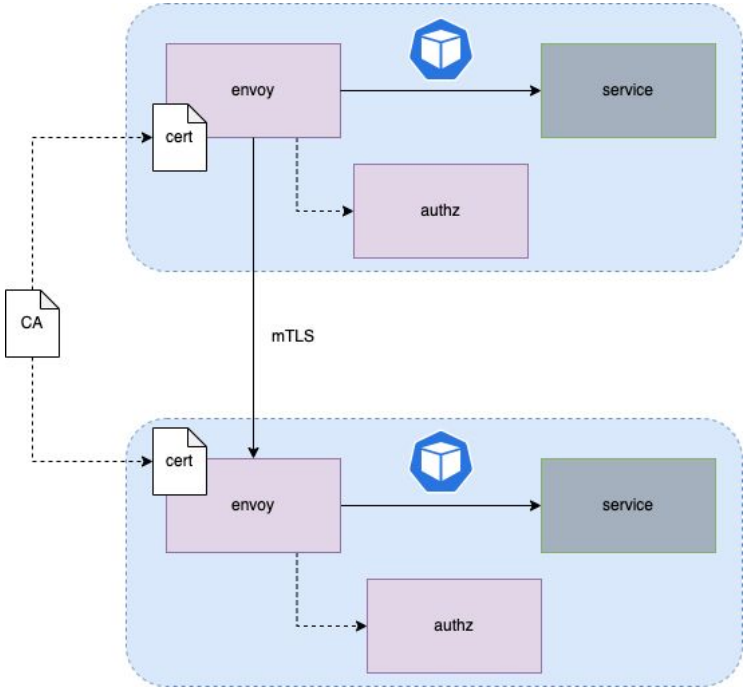


authz-фильтр

```yaml
http_filters:
  - name: envoy.filters.http.ext_authz
    typed_config:
      "@type": type.googleapis.com/envoy.extensions.filters.http.ext_authz.v3.ExtAuthz
      grpc_service:
        envoy_grpc:
          cluster_name: ext-authz
      with_request_body:
        max_request_bytes: 1024
        allow_partial_message: true
        pack_as_bytes: true
```

```yaml
clusters:
  - name: ext-authz
    connect_timeout: 0.25s
    type: logical_dns
    lb_policy: round_robin
    load_assignment:
      cluster_name: ext-authz
      endpoints:
      - lb_endpoints:
        - endpoint:
            address:
              socket_address:
                address: 127.0.0.1
                port_value: 10003
```

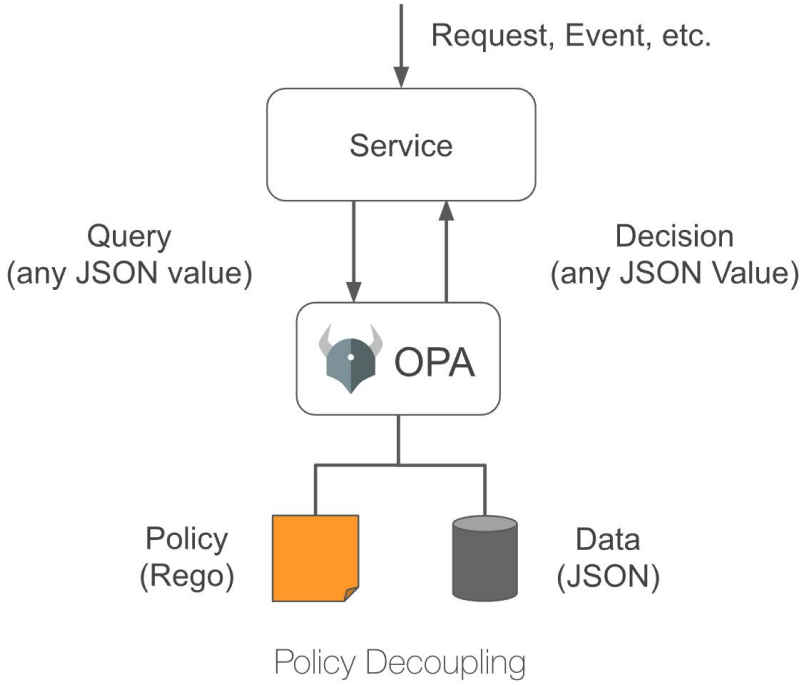
mTLS + authz



Open Policy Agent

- CNCF graduated
- Стартовал в 2016 году
- Service mesh — одно из назначений
- Использует rego для политик

Архитектура



Policy Decoupling

Policy

Rego
Декларативный язык для политик

- Вложенные структуры
- Читабельность
- Расширяемость
- Тестирование и дебаг

```
package indexed

default allow := false

allow {
    some user
    input.method == "GET"
    input.path = ["accounts", user]
    input.user == user
}

allow {
    input.method == "GET"
    input.path == ["accounts", "report"]
    roles[input.user][_] == "admin"
}

allow {
    input.method == "POST"
    input.path == ["accounts"]
    roles[input.user][_] == "admin"
}

roles := {
    "bob": ["admin", "hr"],
    "alice": ["procurement"],
}
```

Читабельность

```python
def hostnames(sites):
    result = []
    for site in sites:
        for server in site.servers:
            result.append(server.hostname)
    return result
```

```
sites[_].servers[_].hostname
```

auth.toml

```toml
version = "0.2"

[default]
description = "Клиенты, которым разрешен доступ ко всем ручкам, для которых не указаны отдельные policy"
clients=[
    # сервисы
    "*",

    # пользователи
    "user:*",

    # боты
    "ext:*"
]

[[policy]]
description = "Ограничить доступ к ручкам get и getAll всем кроме указанного набора клиентов"
endpoints=["rpc:get", "rpc:getAll"]
clients=[
   # сервисы
   "atlas",
   "paas-api",

   # пользователи
   "user:ipivanov",

   # боты
   "ext:paas-bot"
]
```

Структура
- default
- policy
- policy
- policy
- ...

version — на этапе беты

[default]

- Обязательная секция (может быть единственной),
- Срабатывает для всех ручек по умолчанию.

```
[default]
description = "Клиенты, которым разрешен доступ ко всем ручкам, для которых не указаны отдельные policy"
clients=[
    # сервисы
    "*",

    # пользователи
    "user:*",

    # боты
    "ext:*"
]
```

Клиенты

paas-сервисы

- * — все сервисы,
- <service_name> — paas-api (без префикса).

Пользователи (user:)

- user:* — все пользователи,
- user:apgubarev — конкретный пользователь из ldap.

Внешние зависимости (все остальные mtls-серты)

- ext:* — все внешние зависимости,
- ext:paas-bot — интеграция с CN = paas-bot.

Пустой массив — запрещено всем!

[default] Примеры

```
[default]
description = "Клиенты, которым разрешен доступ ко всем ручкам, для которых не указаны отдельные policy"
clients=[
    # сервисы
    "*",

    # пользователи
    "user:*",

    # боты
    "ext:*"
]
```

```
version = "0.2"

[default]
description = "По умолчанию доступ закрыт всем"
clients=[]
```

[[policy]]

- Описывает политику для одной или нескольких ручек
- Количество политик в одном auth.toml не ограничено.
- Полностью перекрывает default для указанных ручек.
- Необходимо перечислить всех клиентов полностью.
- Один и тот же endpoint — только один раз в auth.toml.

Пример

```
version = "0.2"

[default]
description = "Закрыть доступ всем"
clients=[]

[[policy]]
description = "Ограничить доступ к get"
endpoints=["rpc:get"]
clients=[
   "atlas"
]

[[policy]]
description = "Ограничить доступ к getAll"
endpoints=["rpc:getAll"]
clients=[
   "paas-api",
   "user:*",
]
```

Транслятор

- Оказался проще, чем думали.
- Удобно просматривать правила в проде.
- Можно дебажить чтением.

```
package envoy.authz

import future.keywords
import input.attributes.request.http as http_request

# defines
default allow := false

path := trim(http_request.path, "/")

x_source := http_request.headers["x-source"]

x_source_ingress := http_request.headers["x-source-ingress"]

source := x_source_ingress if {
        startswith(x_source_ingress, "user:")
} else := x_source_ingress if {
        startswith(x_source_ingress, "ext:")
} else := x_source

source_is_user := startswith(source, "user:")

source_is_ext := startswith(source, "ext:")

source_is_service if {
        not source_is_user
        not source_is_ext
        count(source) > 0
}

# maintenance
allow if {
        path == "_info"
}

# policies

# title: default
# description: Generated from [default], allow all ext
allow if {
        source_is_ext
}

# title: default
# description: Generated from [default]
allow if {
        source in ["some-service"]
}
```

И снова платить

Не было grpc-метрик, теперь есть (github)

Кэширование на envoy

- Ключ — клиент и ручка, значение — ответ OPA.
- Инвалидация вместе с редеплоем.
- Реализовали фильтр на lua.
- Дало ускорение сайдкара — максимум 2 мс.

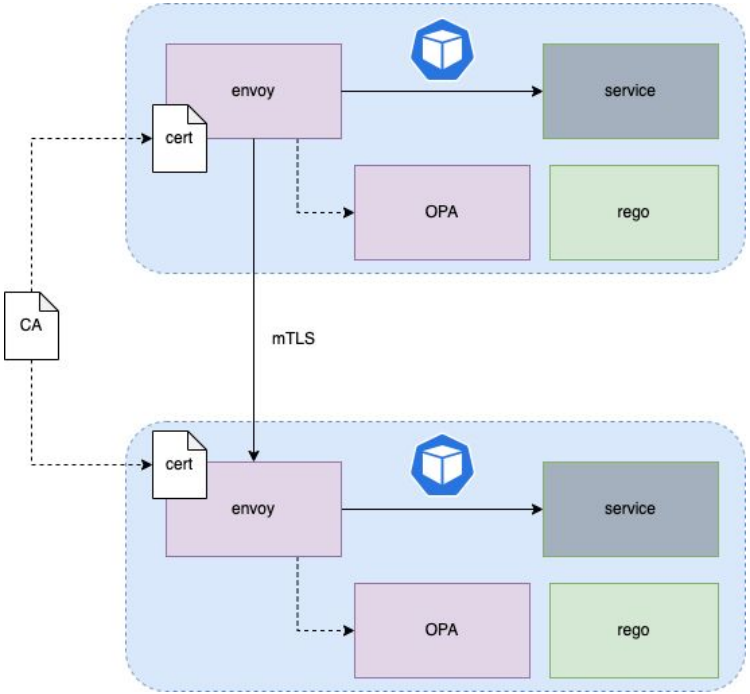
Istio-авторизация

У Istio есть своя авторизация (в базовых случаях достаточно).

Нам не хватило:

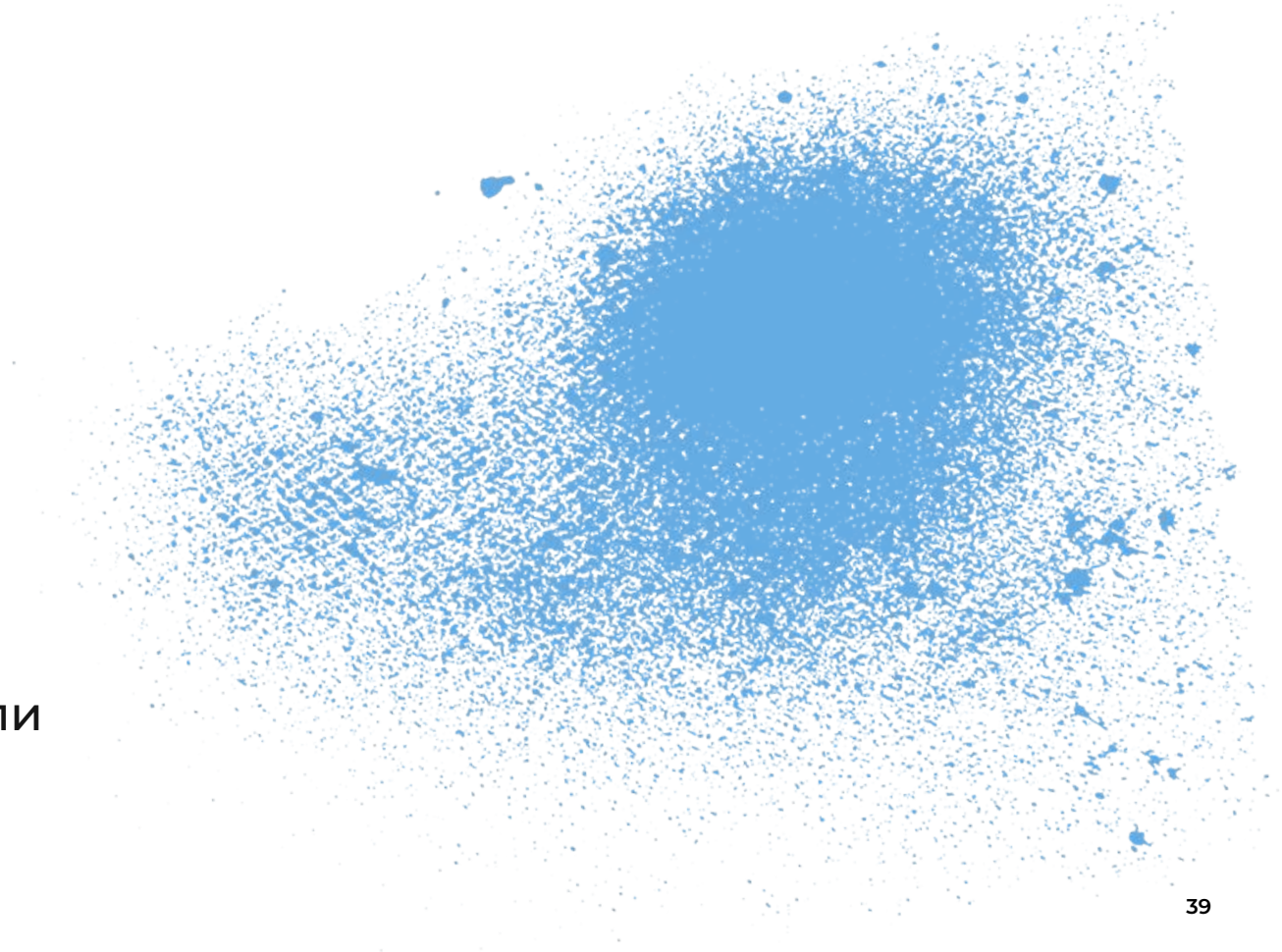
- Реализовали валидацию и тестирование rego-правил.
- Смогли повлиять на latency.

Что получилось



Выводы

Какой опыт получили



Выводы

- Service Mesh даёт профит, но на больших объёмах.
- Service Mesh это дорого.
- Реализовали платформенную аутентификацию и авторизацию.
- OPA/rego — мощный инструмент, но нужен для сложных случаев.
- Недостаточно просто включить авторизацию.

avito.tech

Москва — 2023

Service Mesh-авторизация с Istio и OpenPolicyAgent

Антон Губарев

antgubarev.dev@gmail.com

antgubarev.tech

GitHub/antgubarev