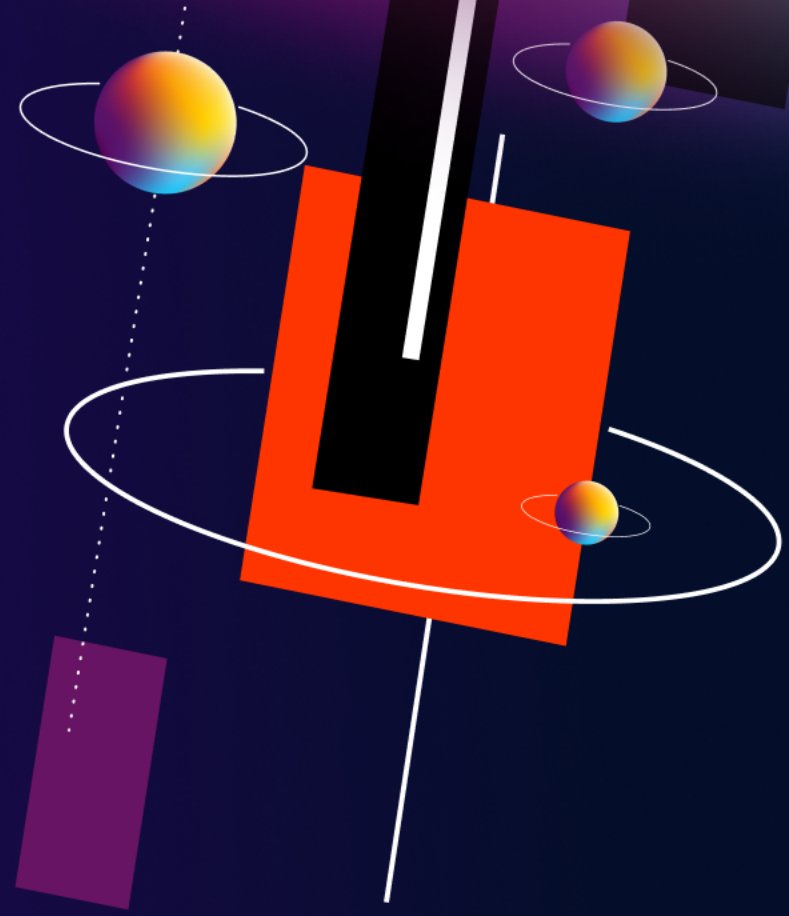
Distributed Observability

Максим Залысин
Positive Technologies



DevOps Conf 2024

О спикере

Максим Залысин

Head of DevOps
Positive Technologies

- Полжизни в IT
  - Руковожу
  - Администрирую
  - Разрабатываю
  - Выступаю

О компании

22 года

опыта исследований
и разработок

2200+

сотрудников: инженеров по ИБ,
разработчиков, аналитиков
и других специалистов

250+

экспертов в нашем
исследовательском
центре безопасности

20+

продуктов в области
информационной
безопасности

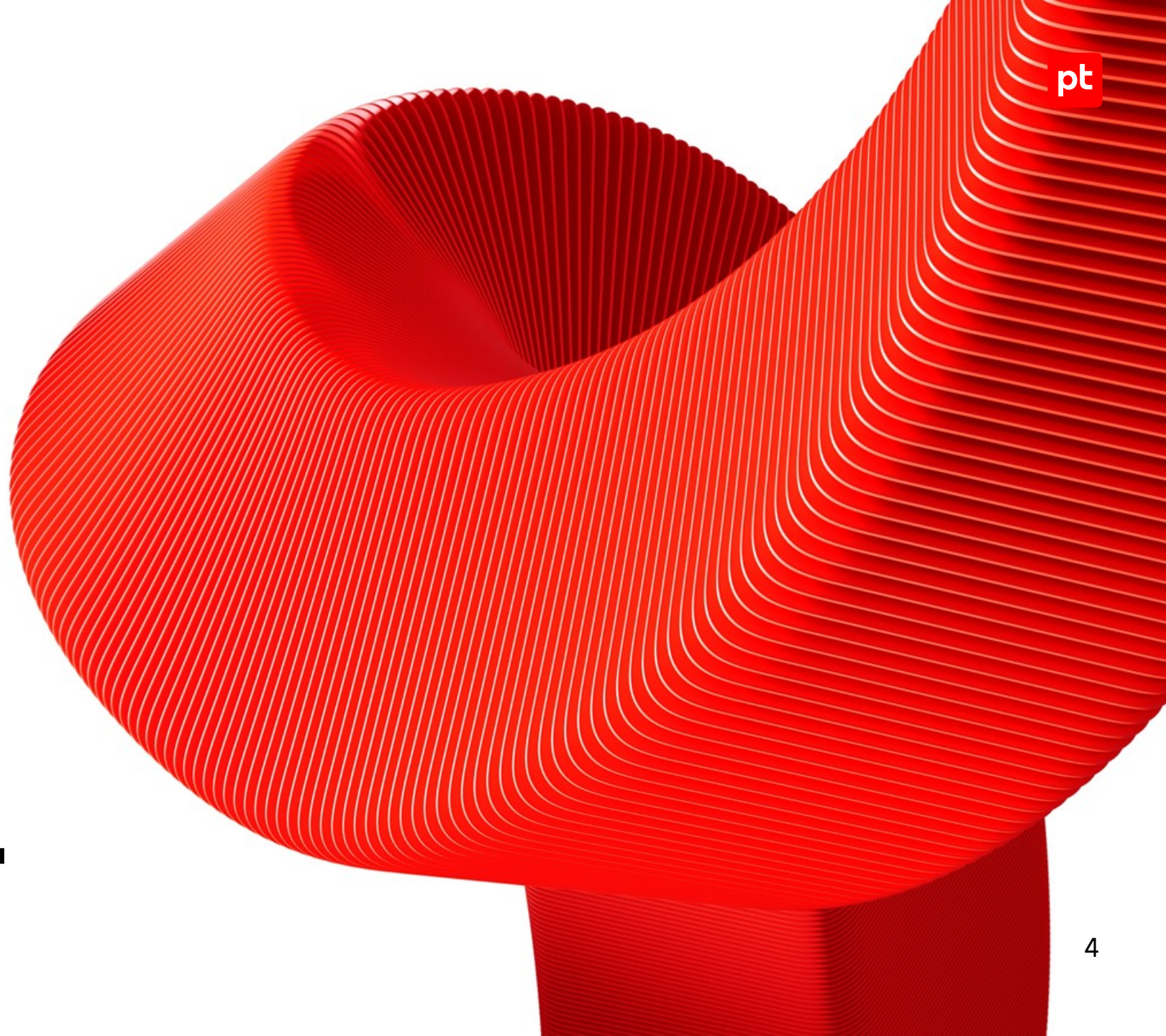
900+

экспертов в R&D

50%

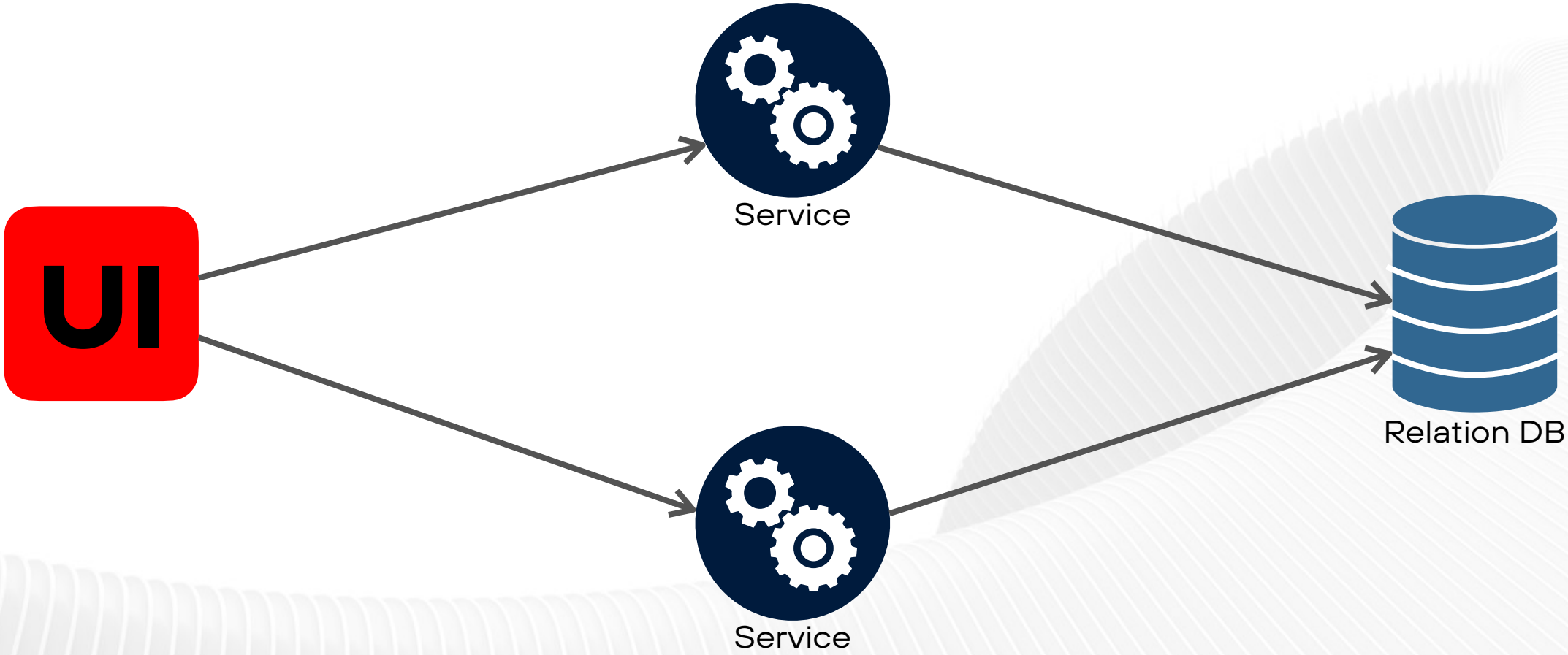
всех уязвимостей
в промышленности
и телекомах обнаружили наши
эксперты

- Создаем продукты и решения
- Проводим аудиты безопасности
- Расследуем инциденты
- Исследуем угрозы

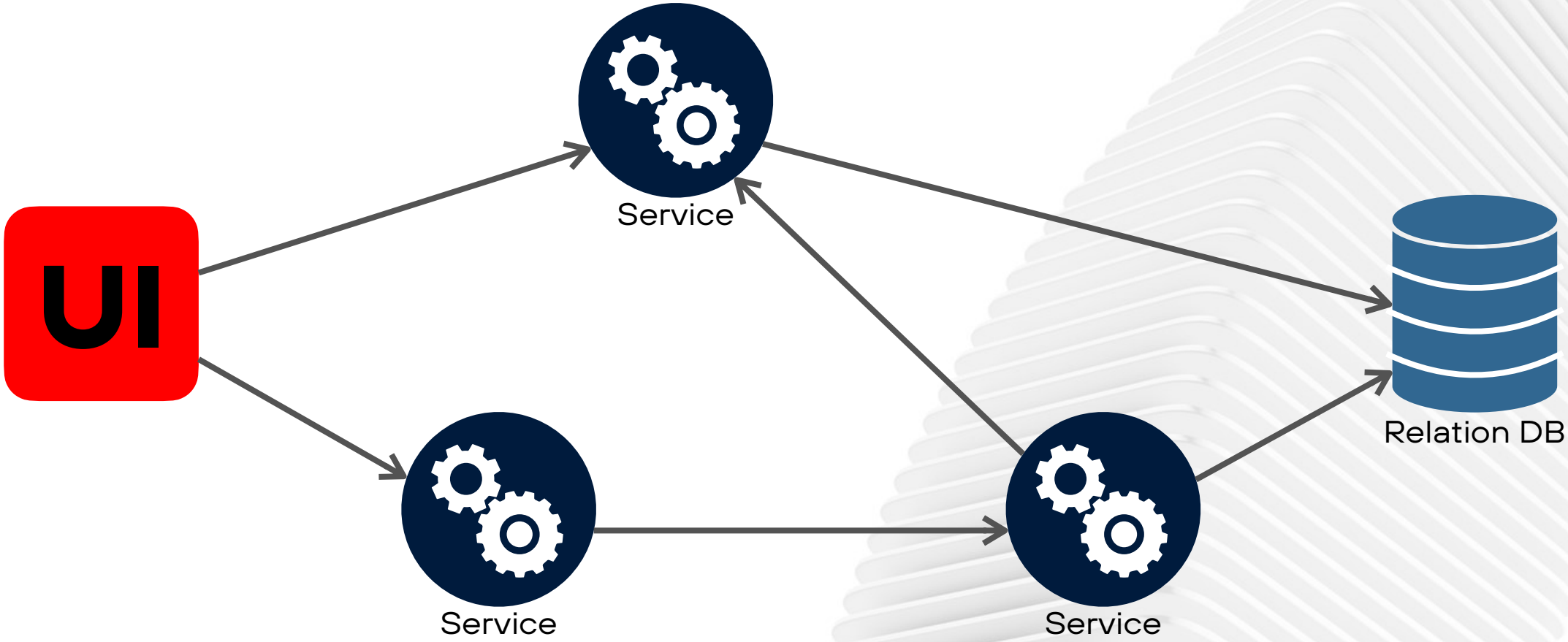


Продукт

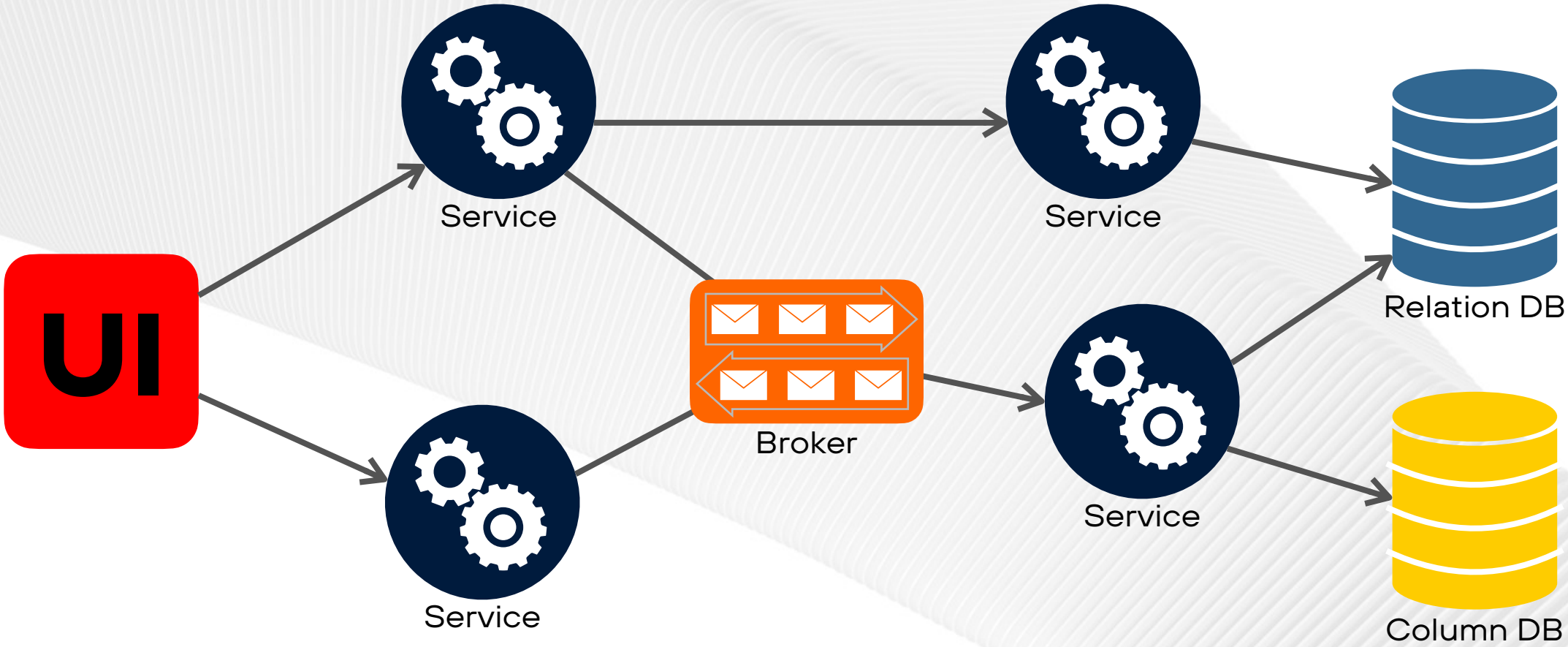
WEB-сервис



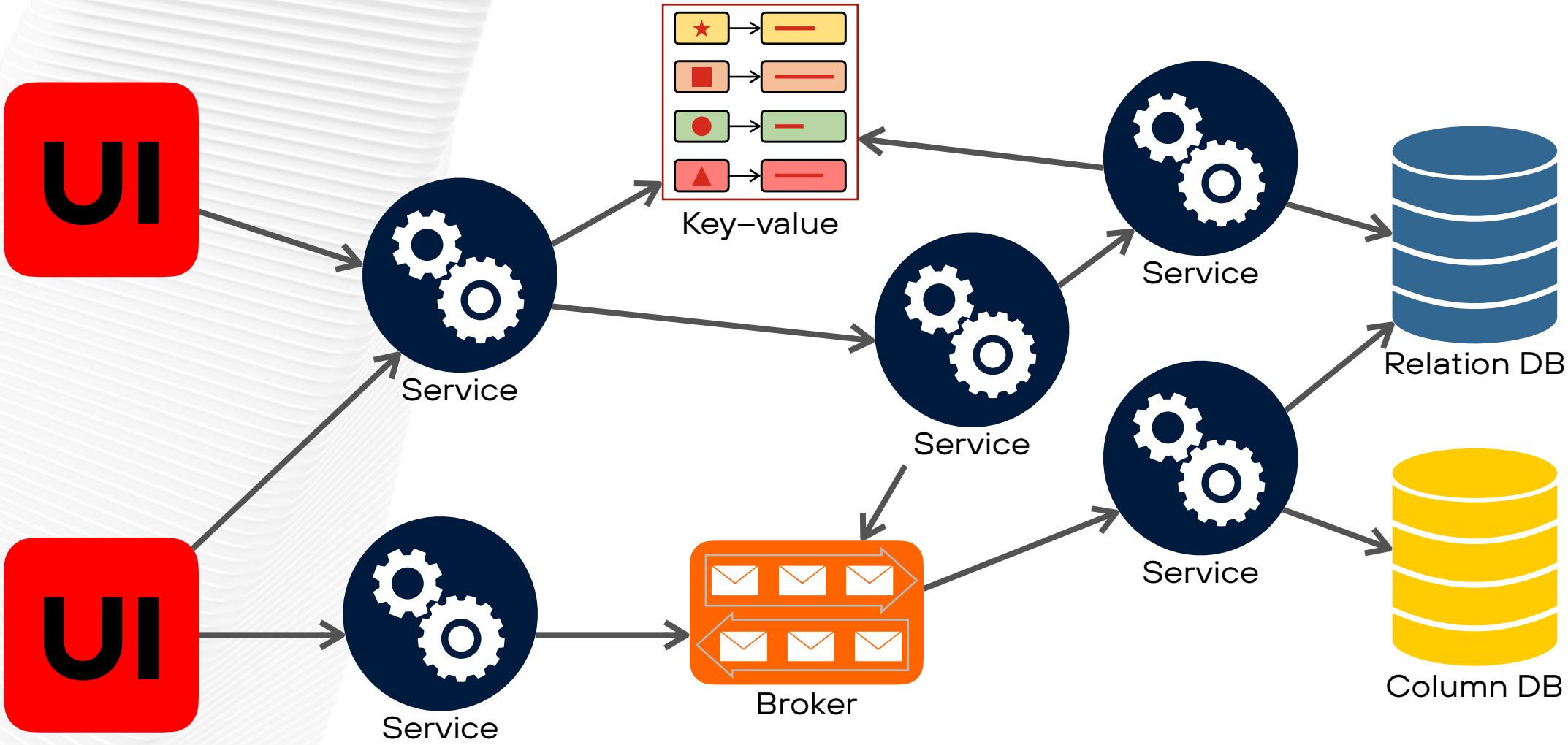
WEB-сервис



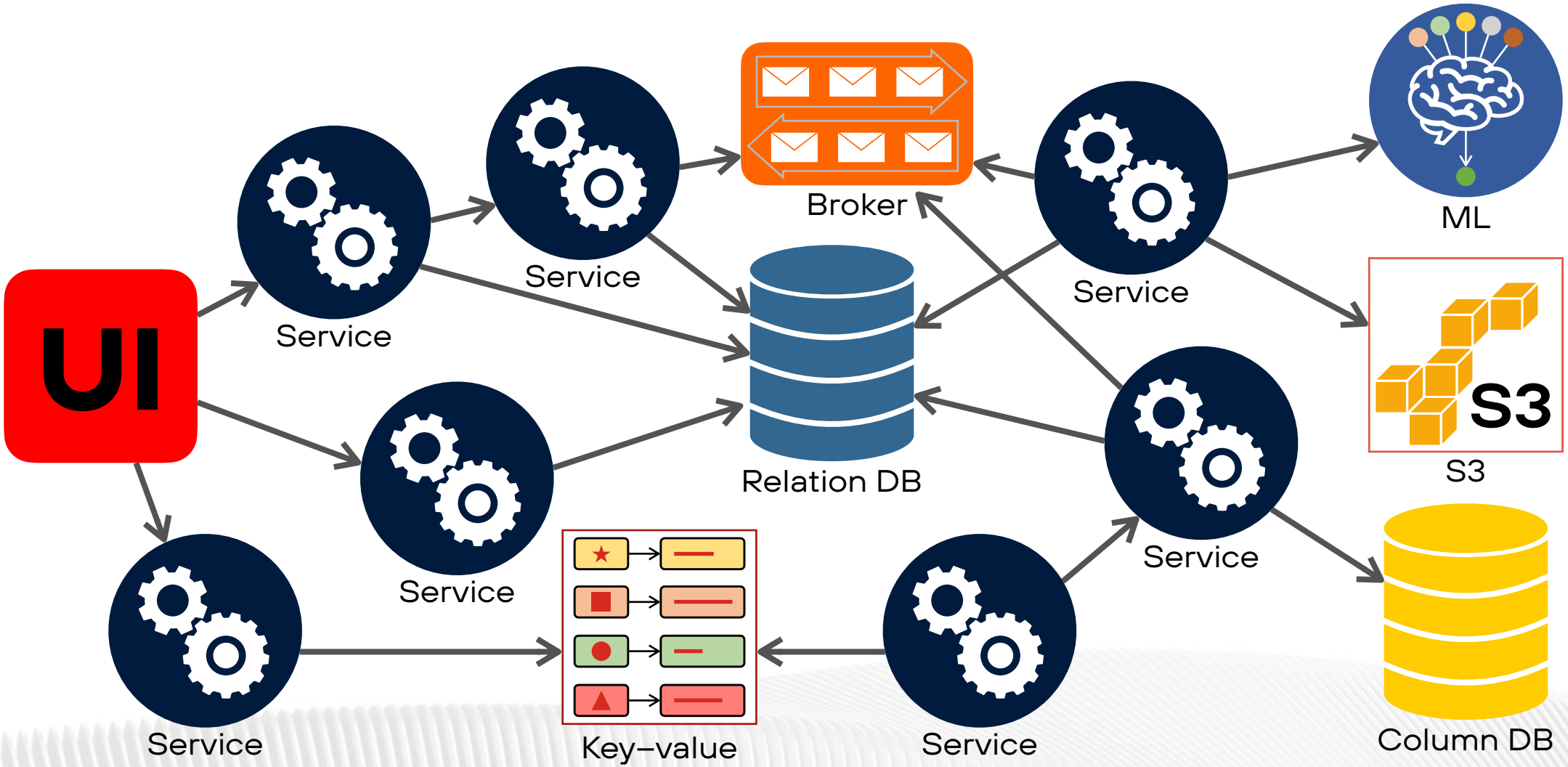
WEB-сервис


UI
Service
Service
Service
Service
Broker
Relation DB
Column DB

Коробочный продукт



Коробочный продукт

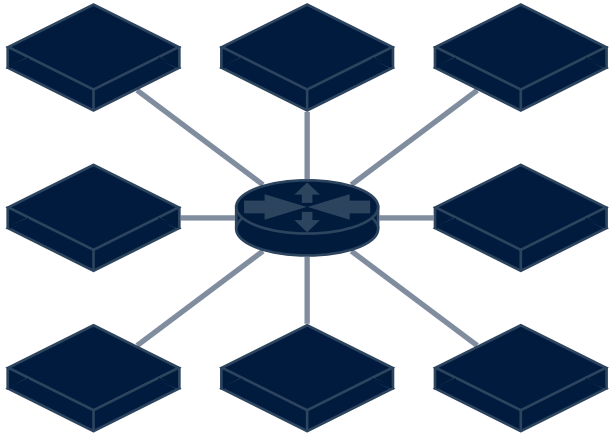


Инфраструктура

Сеть

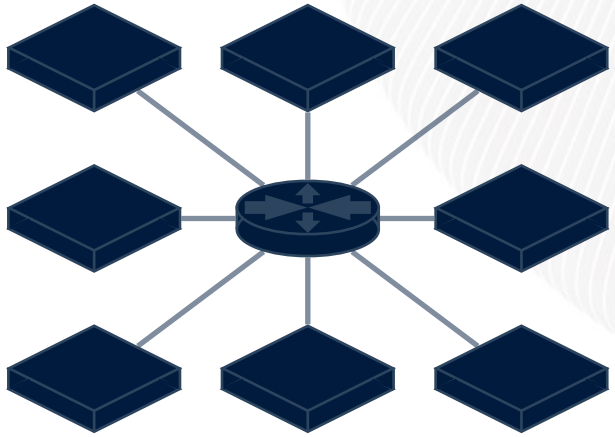
Сеть

Сеть

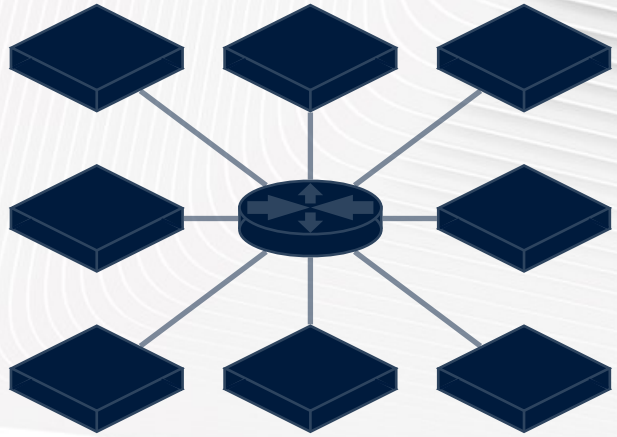


infra

stand

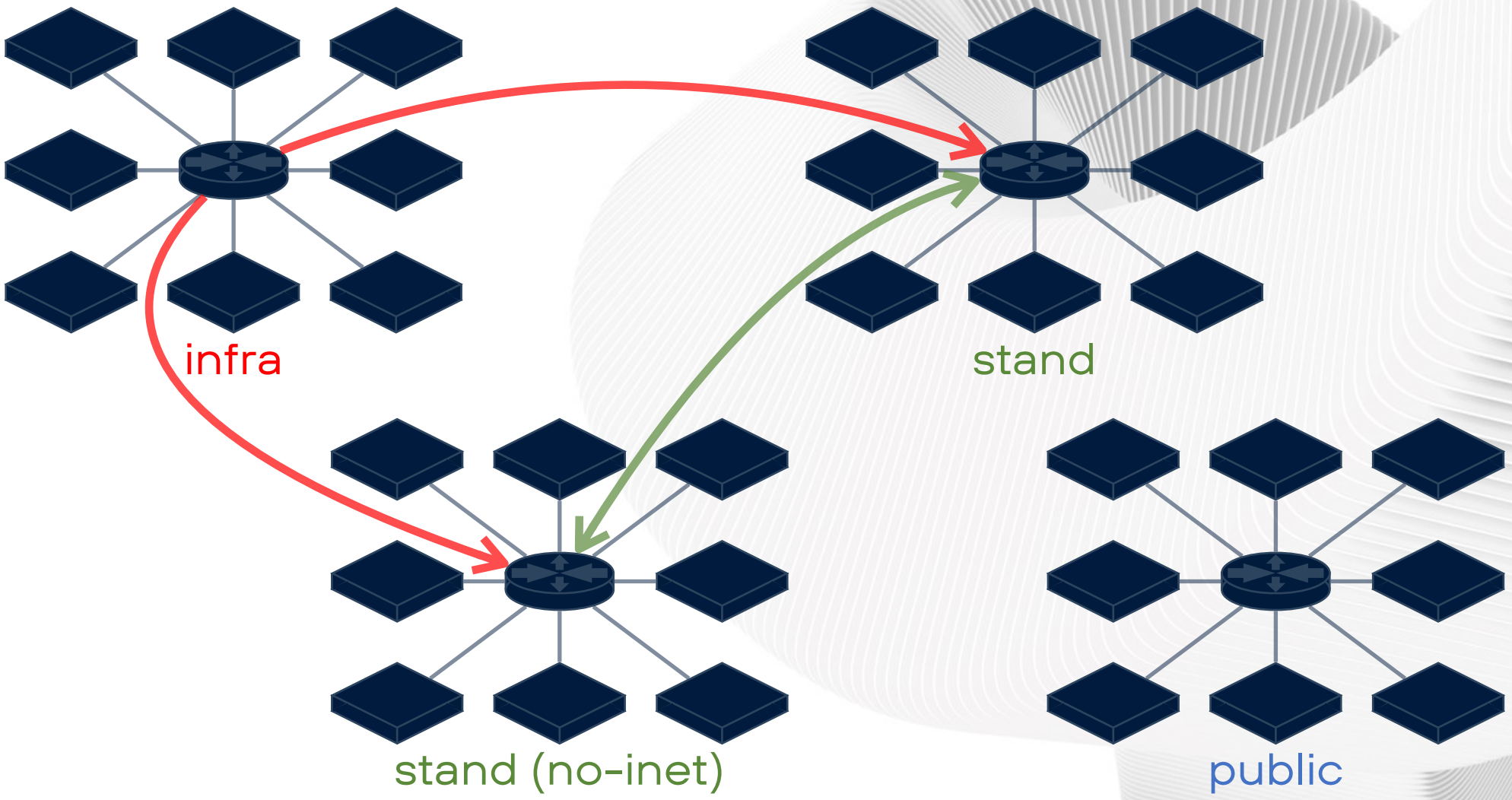


stand (no-inet)

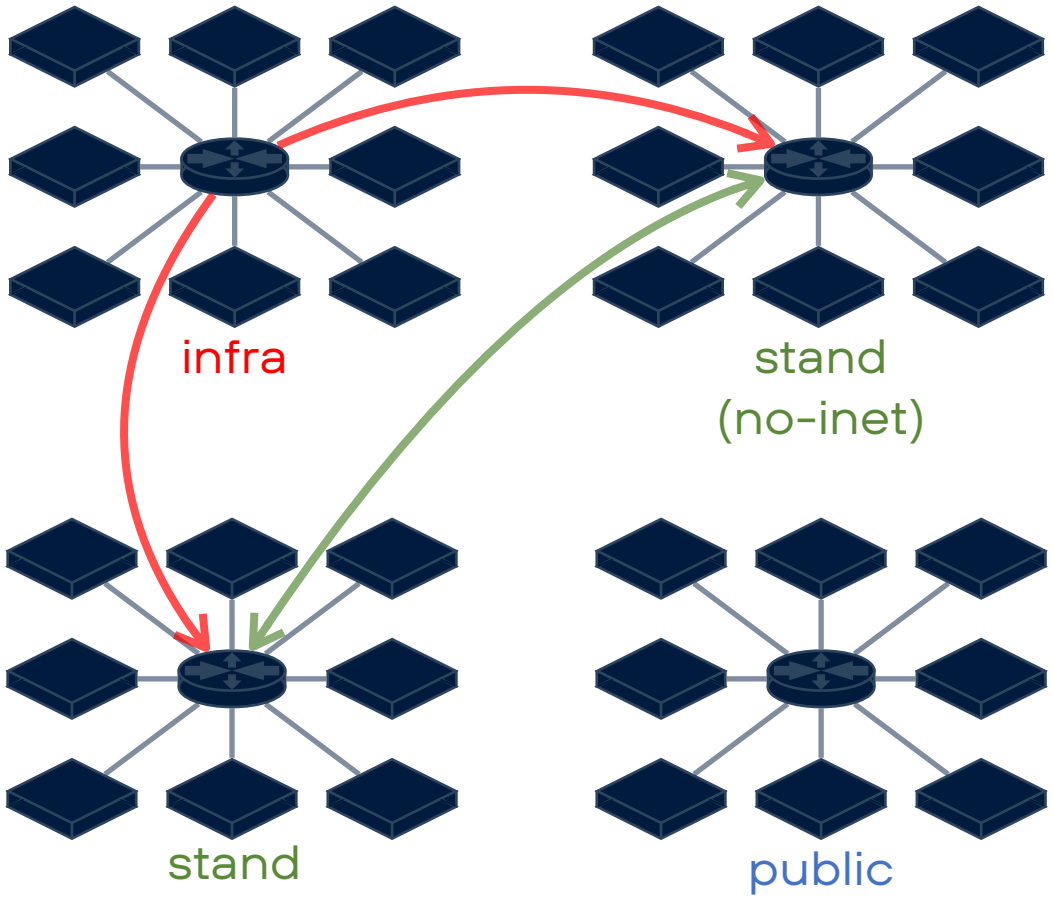


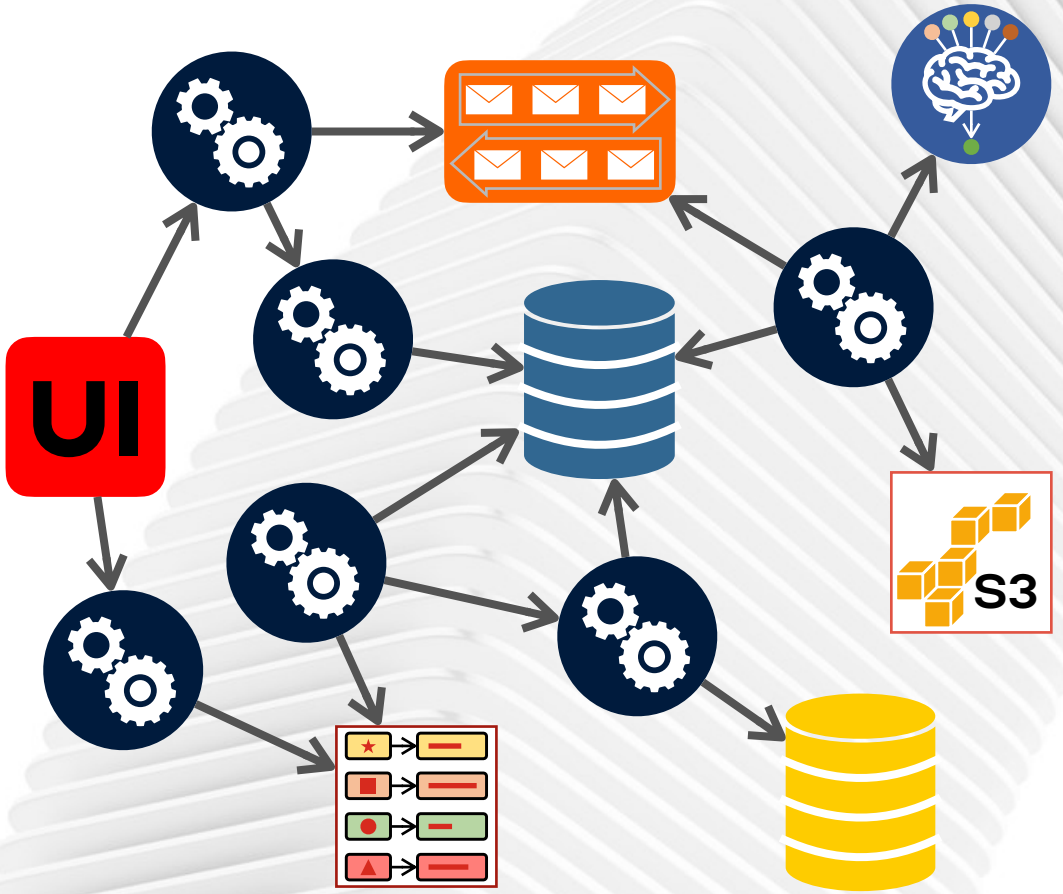
public

Сеть

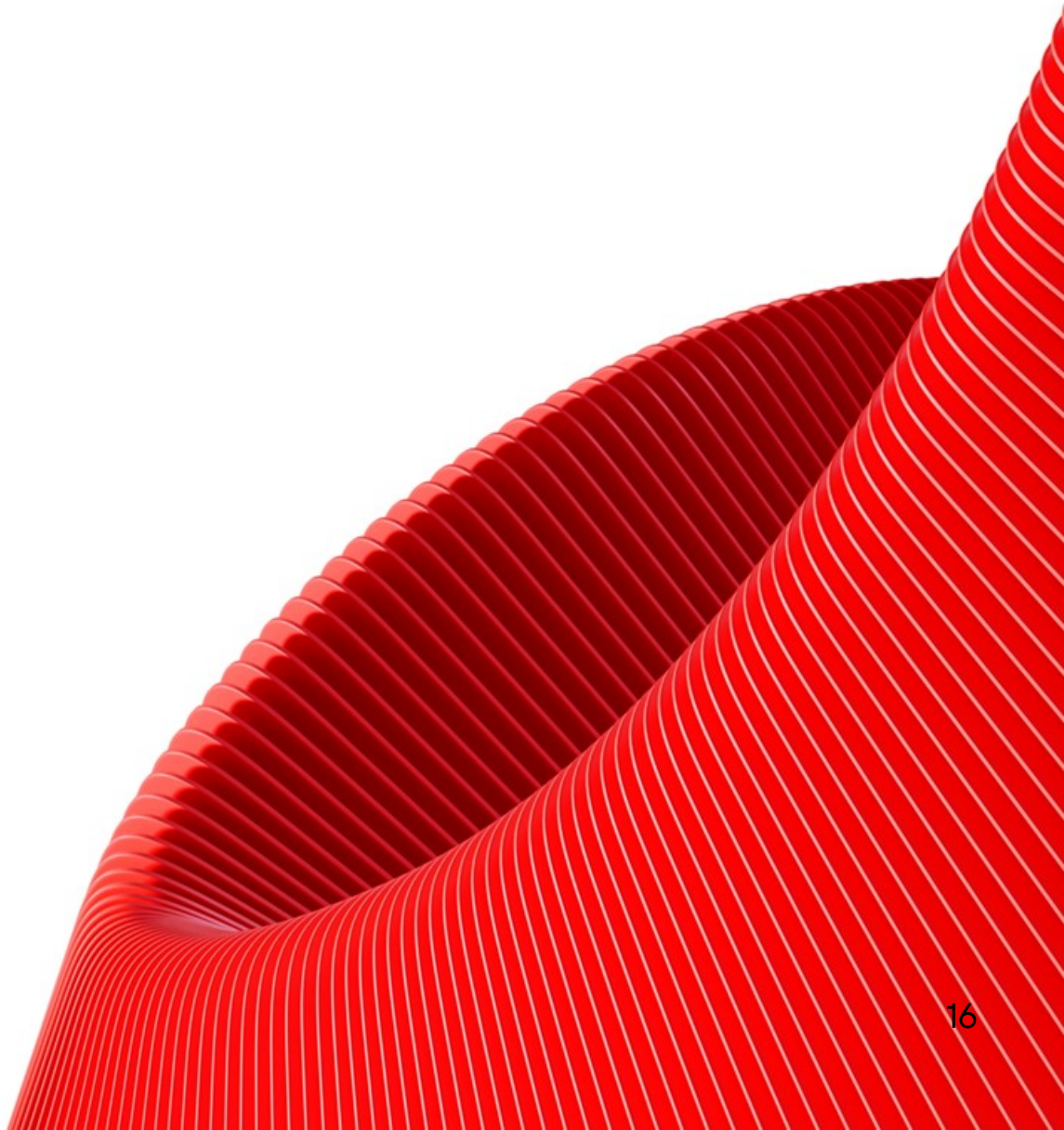


Требования



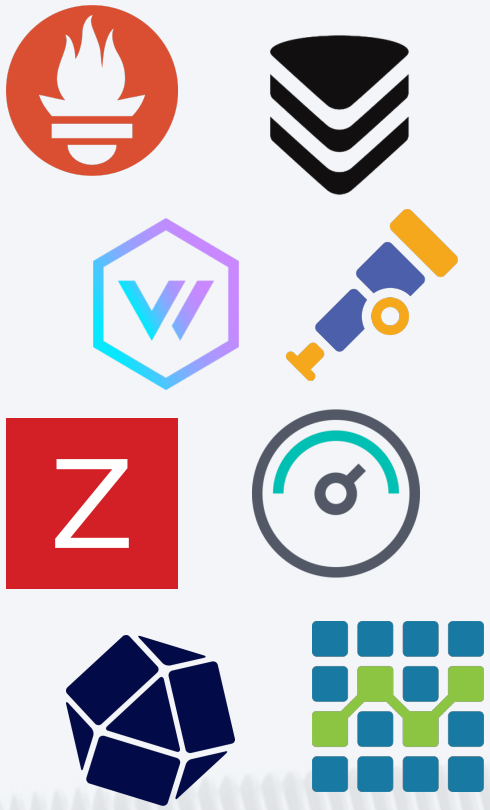


Решения

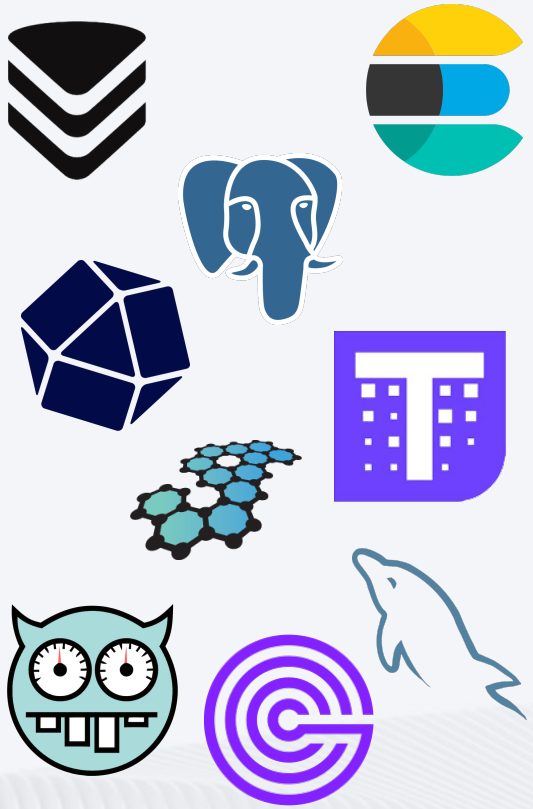


Метрики

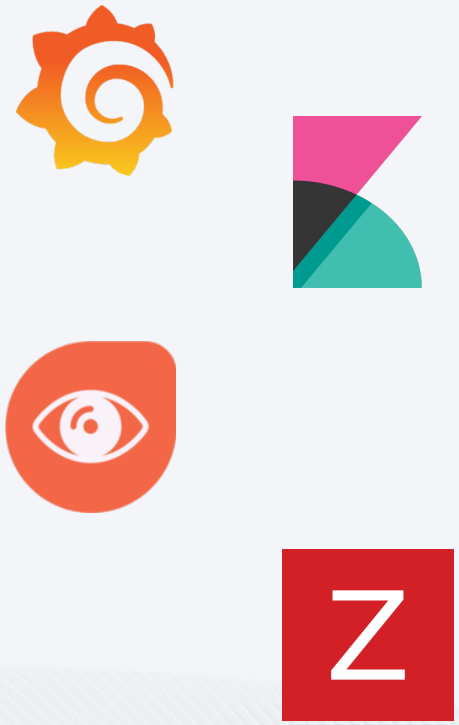
Сбор



Хранение

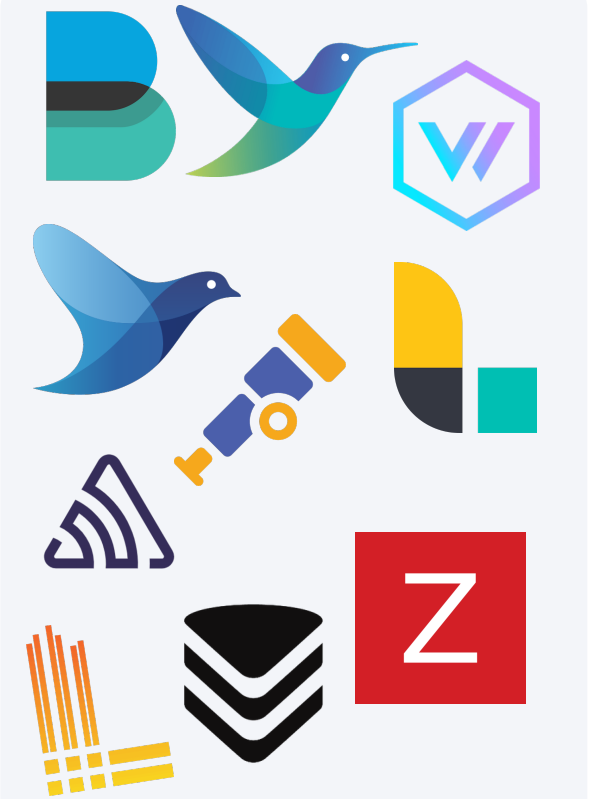


Вывод

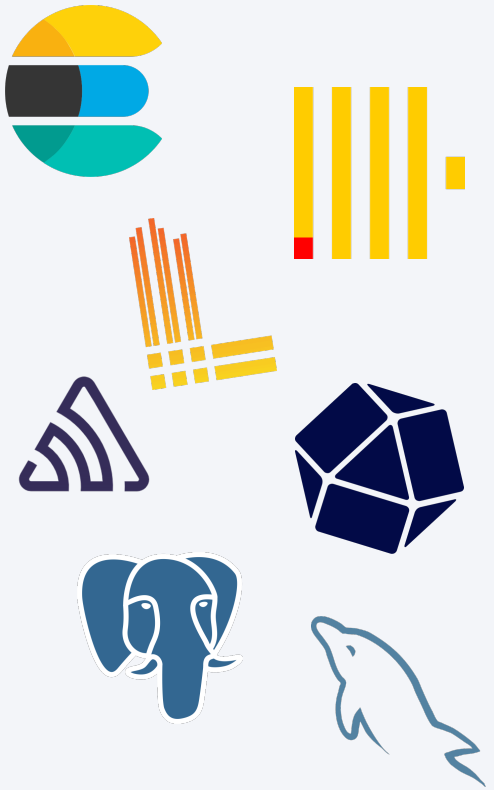


Логи

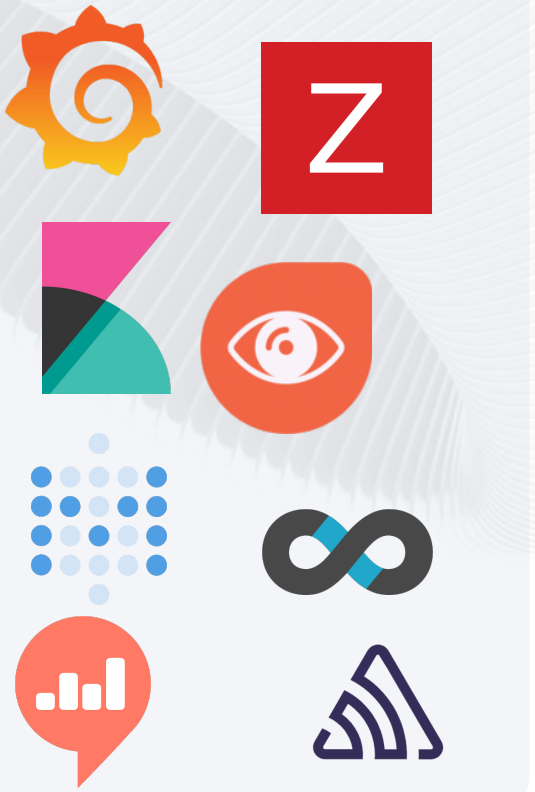
Сбор



Хранение



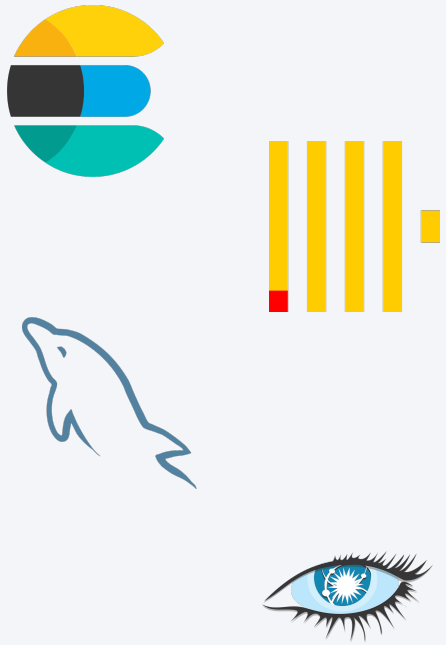
Вывод



Трейсы

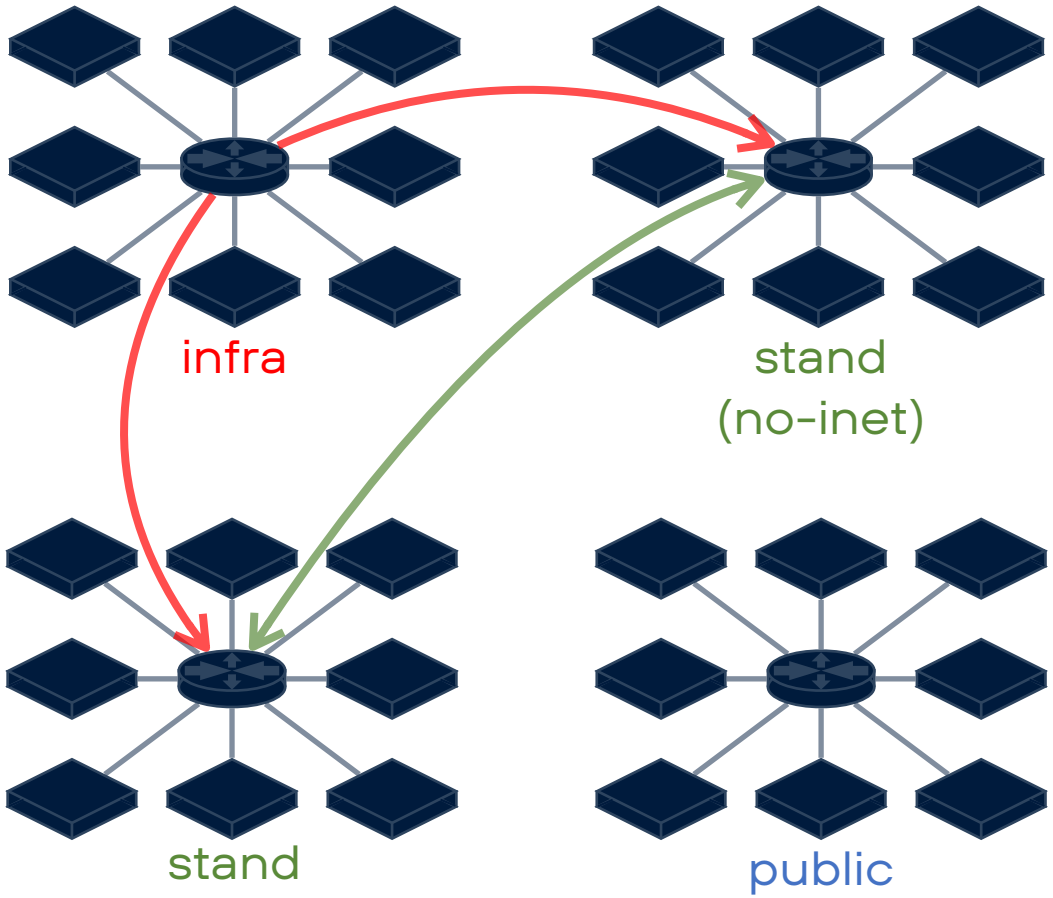
Сбор

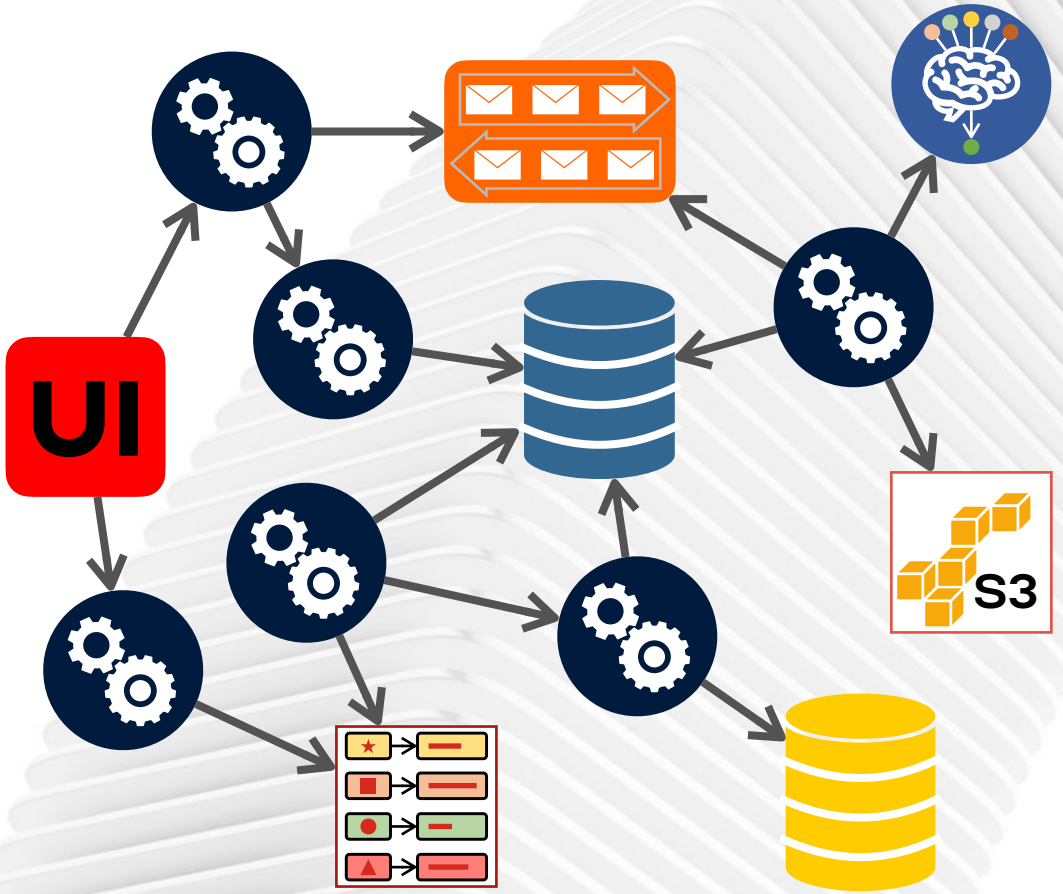
Хранение



Вывод

Требования





Observability

Сбор

Хранение

Вывод

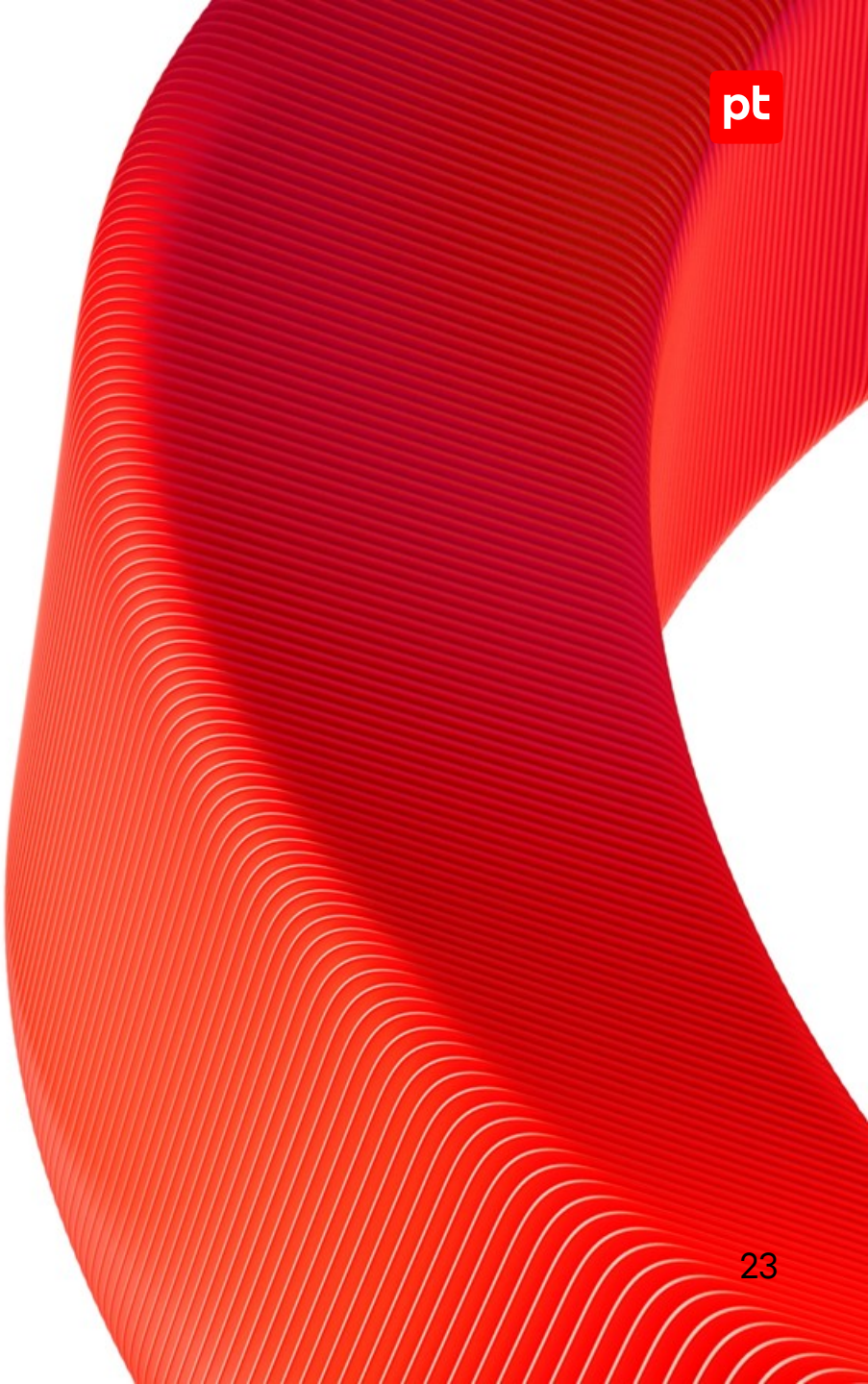
Observability

Сбор

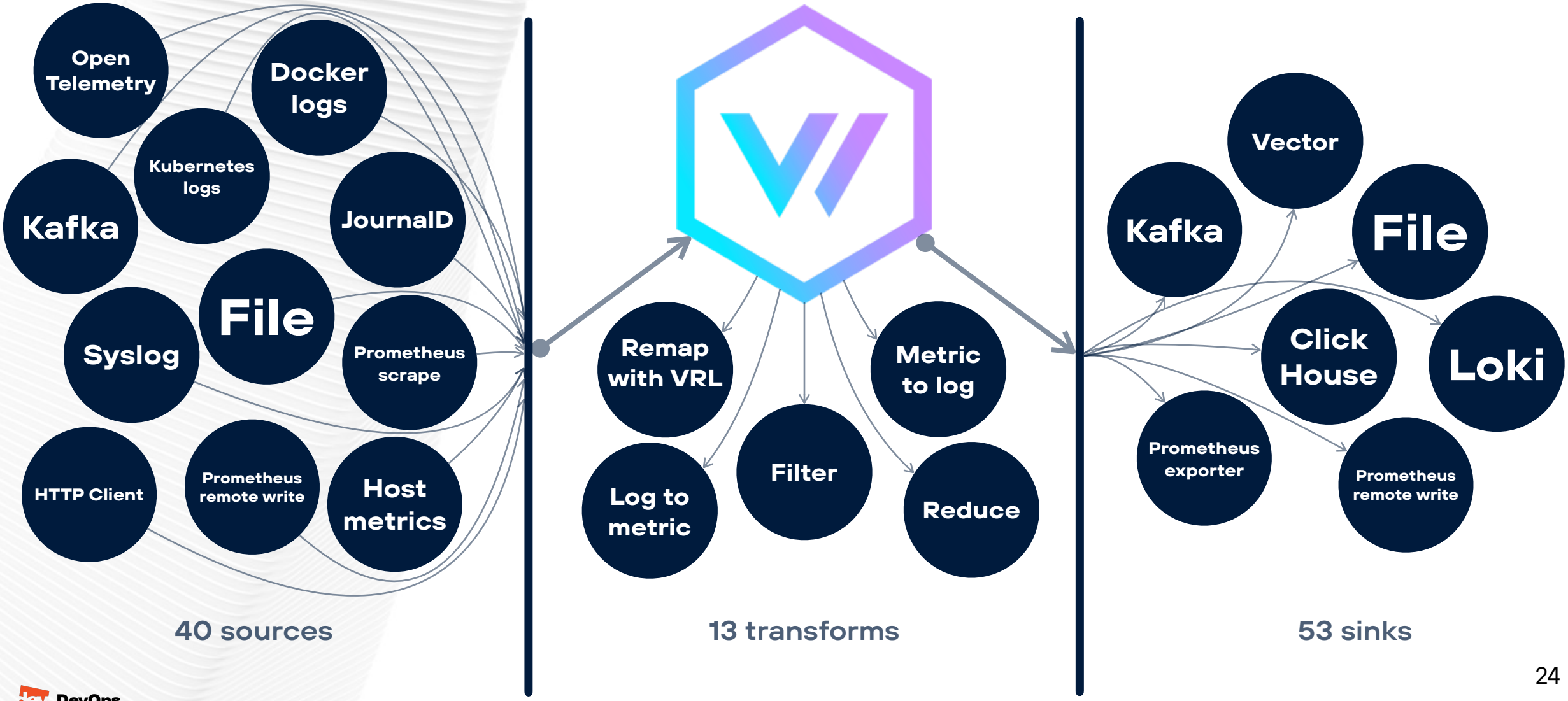
Хранение

Вывод

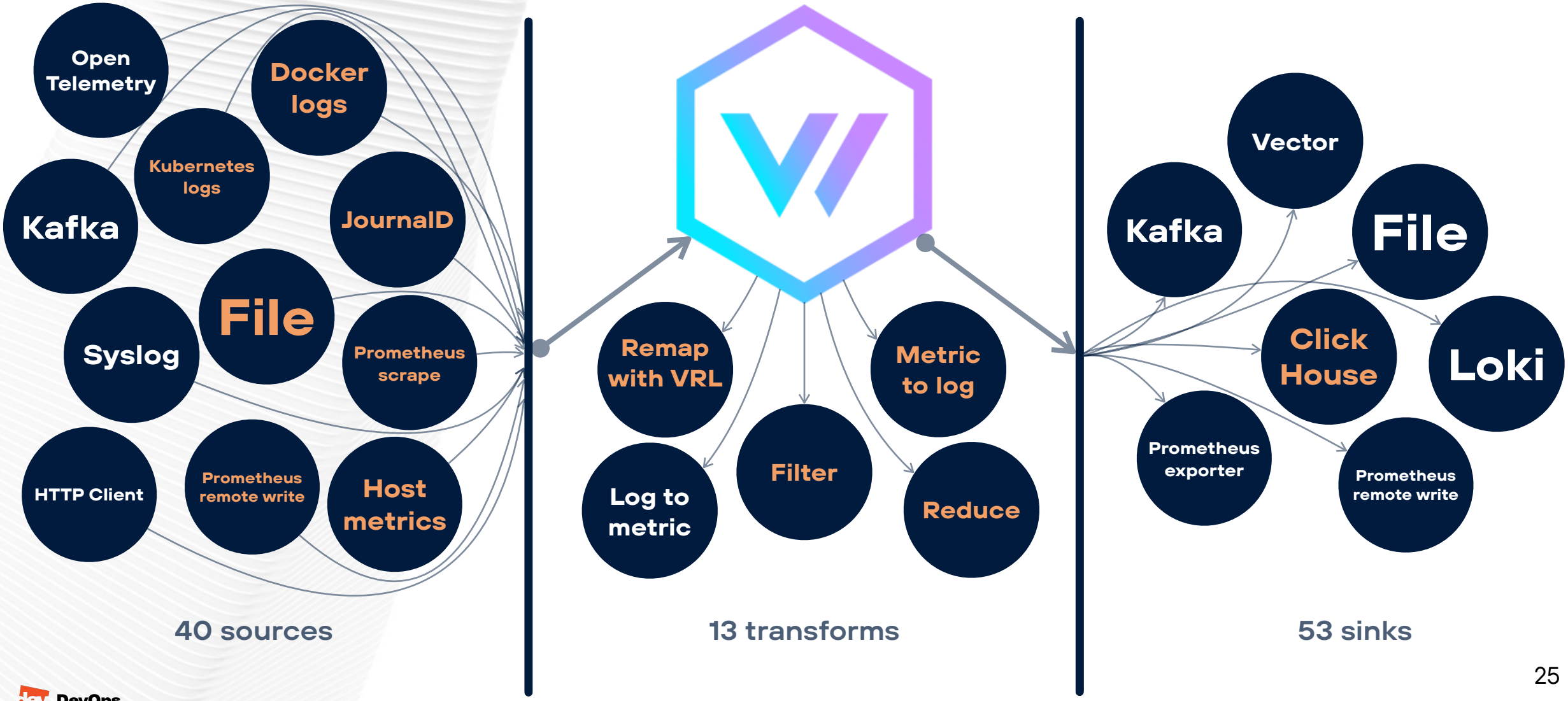
Vector



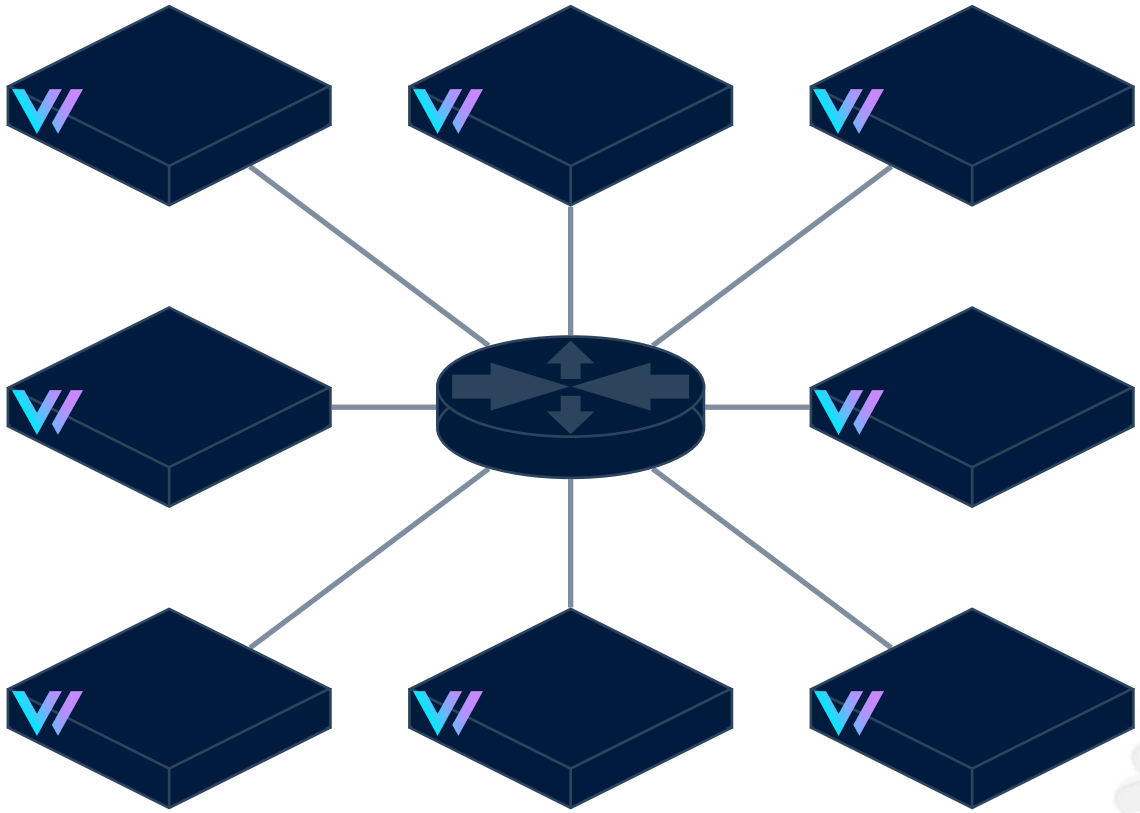
Vector by Datadog



Vector by Datadog



Vector



— Vector

youtrack_logs.toml

```
[sources.youtrack_logs]
  type = "file"
  include = [ "/host/opt/youtrack/logs/*.log" ]
  [sources.youtrack_logs.multiline]
    start_pattern = '^\[[^\]]+\]\s+\d{2}:\d{2}:\d{2},\d{3}\s+[A-Z]+\s+'
    condition_pattern = '^\[[^\]]+\]\s+\d{2}:\d{2}:\d{2},\d{3}\s+[A-Z]+\s+'
    mode = "halt_before"
[transforms.youtrack_logs_remap]
  type = "remap"
  inputs = [ "youtrack_logs" ]
  source = '''...'''
```

dmesg_logs.toml

```
[sources.dmesg_logs]
  type = "journald"
   journal_directory = "/host/var/log/journal"
   extra_args = [ "--identifier=kernel" ]
[transforms.dmesg_logs_remap]
  type = "remap"
   inputs = [ "dmesg_logs" ]
   source = '''...'''
```

services_logs.toml

```
[sources.services_logs]
  type = "docker_logs"
   include_containers = [ "frontend", "backend" ]
 [transforms.services_logs_remap]
   type = "remap"
  inputs = [ "services_logs" ]
   source = '''...'''
```

logs_to_clickhouse.toml

```
[sinks.logs_to_clickhouse]
  type = "clickhouse"
  endpoint = "http://clickhouse-01.local:8123"
  table = "logs"
  inputs = [ "*_logs_remap" ]
  encoding.timestamp_format = "unix"
  encoding.only_fields = [ "timestamp", "level", "message", "app", "host", "data" ]
```

host_metrics.toml

```
[sources.host_metrics]
  type = "host_metrics"
  scrape_interval_secs = 10
  collectors = [ "cpu", "disk", "filesystem", "load", "host", "memory", "network" ]
  network.devices.includes = [ "eth*" ]
  filesystem.filesystems.includes = [ "ext*" ]
  filesystem.mountpoints.includes = [ "/host" ]
```

node_exporter_metrics.toml

```
[sources.node_exporter_metrics]
  type = "prometheus_scrape"
  scrape_interval_secs = 10
  endpoints = [ "http://127.0.0.1:9100/metrics" ]
  instance_tag = "instance"
  endpoint_tag = "endpoint"
```

postgres_metrics.toml

```
[sources.postgres_metrics]
  type = "postgres_metrics"
  scrape_interval_secs = 10
  endpoints = [ "postgresql://vector:vector@127.0.0.1:5432/postgres" ]
```

metrics_to_clickhouse.toml

```
[transforms.metrics_to_log]
  type = "metric_to_log"
  inputs = [ "*_metrics" ]
[transforms.metrics_remap]
  type = "remap"
  inputs = [ "metrics_to_log" ]
  source = '''...'''
[sinks.metrics_to_clickhouse]
  type = "clickhouse"
  endpoint = "http://clickhouse-01.local:8123"
  table = "metrics"
  inputs = [ "metrics_remap" ]
  encoding.timestamp_format = "unix"
  encoding.only_fields = [ "timestamp", "name", "type", "value", "buckets", "quantiles", "host", "tags" ]
```

metrics_to_clickhouse.toml

```
[transforms.metrics_to_log]
  type = "metric_to_log"
  inputs = [ "*_metrics" ]
[transforms.metrics_remap]
  type = "remap"
  inputs = [ "metrics_to_log" ]
  source = '''...'''
[sinks.metrics_to_clickhouse]
  type = "clickhouse"
  endpoint = "http://clickhouse-01.local:8123"
  table = "metrics"
  inputs = [ "metrics_remap" ]
  encoding.timestamp_format = "unix"
  encoding.only_fields = [ "timestamp", "name", "type", "value", "buckets", "quantiles", "host", "tags" ]
```

Плюсы и минусы

Плюсы

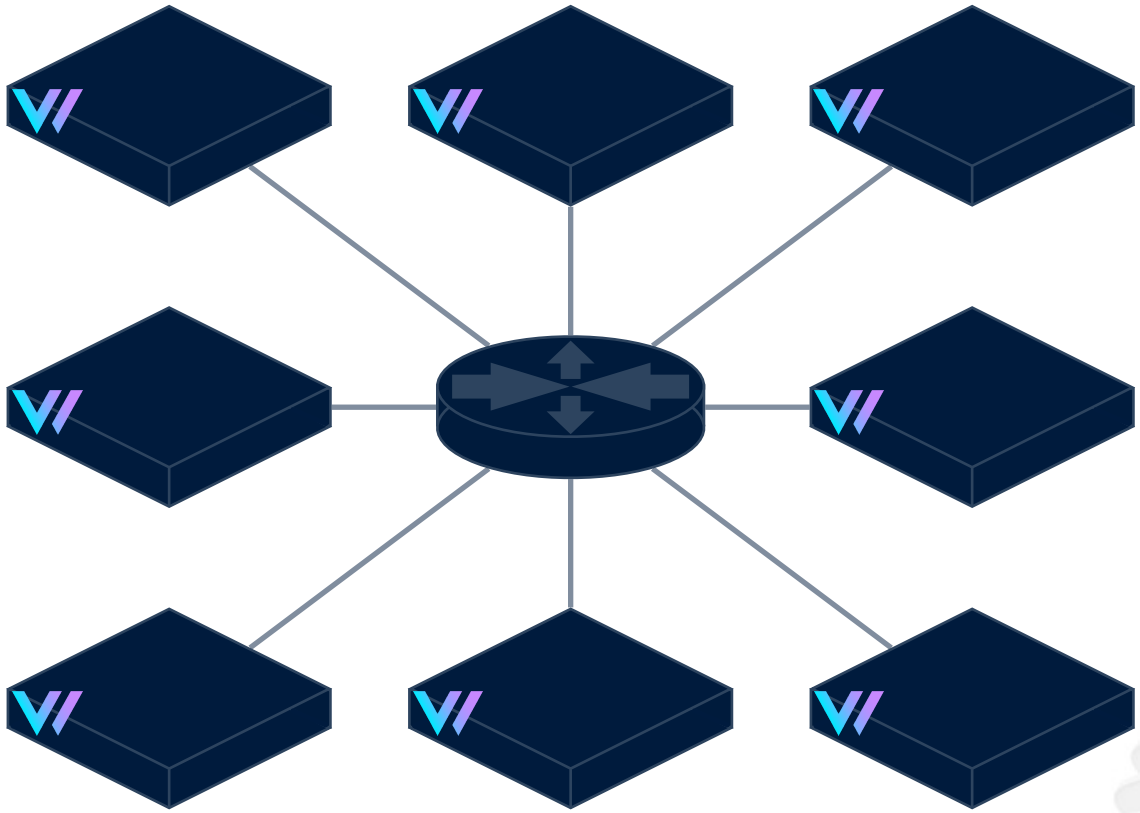
1. Rust – современно и быстро
2. VRL (Vector Remap Language)
3. 40 sources, 13 transforms, 53 sinks
4. All-in-One-транспорт

Минусы

1. Нет service discovery
2. Нет транспорта трейсов
3. Не поддерживает multiline с journald

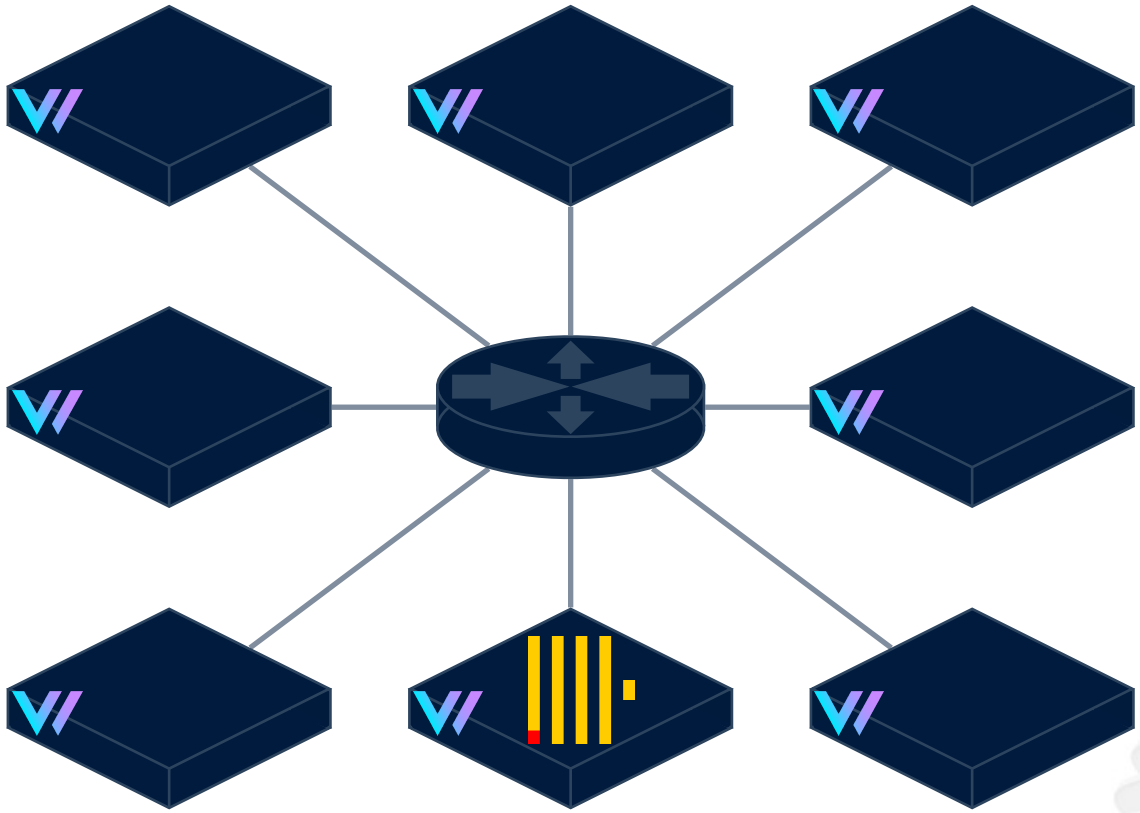
ClickHouse

Vector



— Vector

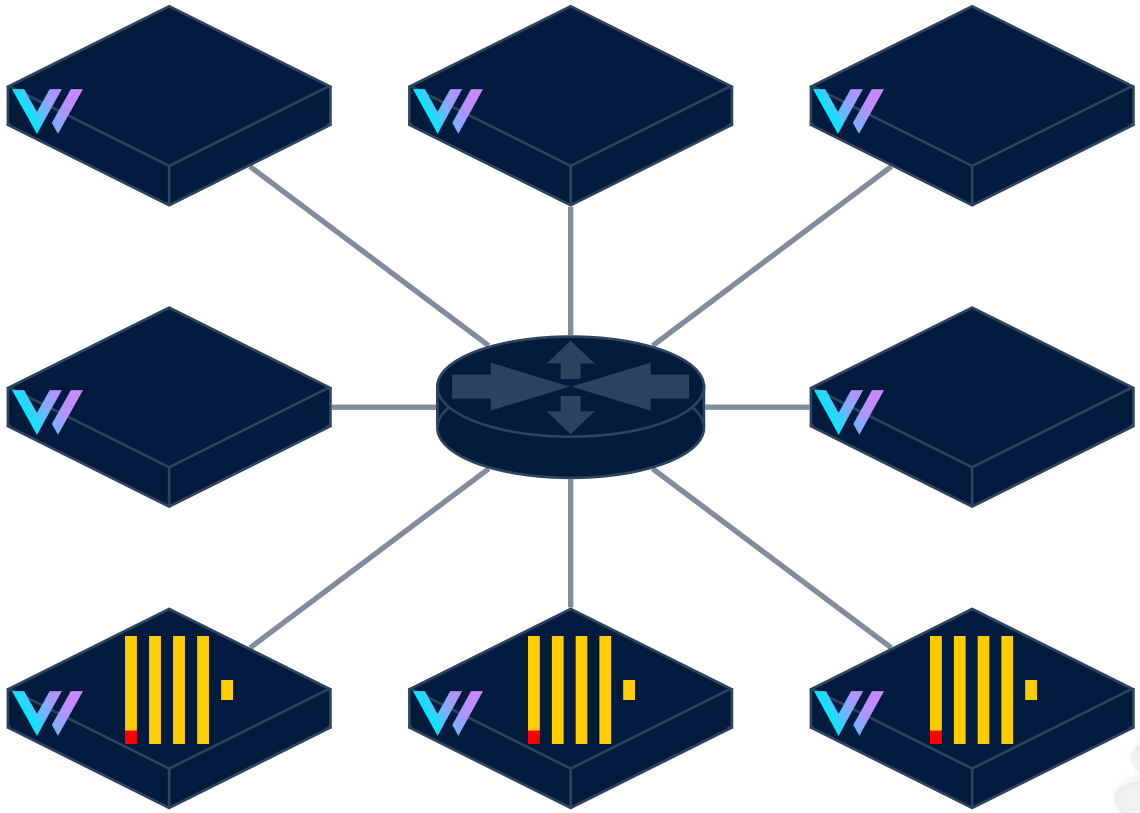
Vector > ClickHouse



— Vector

— ClickHouse

Vector > ClickHouse cluster



— Vector

— ClickHouse

bind.xml

```
<clickhouse>
    <listen_host>0.0.0.0</listen_host>
    <listen_try>1</listen_try>
    <http_port>8123</http_port>
    <tcp_port_secure>9440</tcp_port_secure>
    <openSSL>
        <server>
            <caConfig>/etc/clickhouse/ssl/ca.crt</caConfig>
            <certificateFile>/etc/clickhouse/ssl/server.crt</certificateFile>
            <privateKeyFile>/etc/clickhouse/ssl/server.key</privateKeyFile>
            <dhParamsFile>/etc/clickhouse/ssl/dhparam.pem</dhParamsFile>
            <verificationMode>strict</verificationMode>
        </server>
    </openSSL>
</clickhouse>
```

bind.xml

```
<clickhouse>
    <listen_host>0.0.0.0</listen_host>
    <listen_try>1</listen_try>
    <http_port>8123</http_port>
    <tcp_port_secure>9440</tcp_port_secure>
    <openSSL>
        <server>
            <caConfig>/etc/clickhouse/ssl/ca.crt</caConfig>
            <certificateFile>/etc/clickhouse/ssl/server.crt</certificateFile>
            <privateKeyFile>/etc/clickhouse/ssl/server.key</privateKeyFile>
            <dhParamsFile>/etc/clickhouse/ssl/dhparam.pem</dhParamsFile>
            <verificationMode>strict</verificationMode>
        </server>
    </openSSL>
</clickhouse>
```

CREATE TABLE metrics

```
CREATE TABLE IF NOT EXISTS metrics
(
  `timestamp` DateTime('UTC') CODEC(DoubleDelta),
  `name` LowCardinality(String),
  `type` LowCardinality(String),
  `value` Float64 CODEC(Gorilla),
  `buckets` Map(LowCardinality(String), Float64),
  `quantiles` Map(LowCardinality(String), Float64),
  `host` LowCardinality(String),
  `tags` Map(LowCardinality(String), String)
)
ENGINE = MergeTree
PARTITION BY toDate(`timestamp`)
ORDER BY (`timestamp`, `name`)
TTL `timestamp` + INTERVAL 3 MONTH DELETE
SETTINGS ttl_only_drop_parts = 1;
```

CREATE TABLE logs

```
CREATE TABLE IF NOT EXISTS logs
(
  `timestamp` DateTime('UTC') CODEC(T64),
  `level` LowCardinality(String),
  `message` String,
  `app` LowCardinality(String),
  `host` LowCardinality(String),
  `data` Map(LowCardinality(String), String)
)
ENGINE = MergeTree
PARTITION BY toDate(`timestamp`)
ORDER BY `timestamp`
TTL `timestamp` + INTERVAL 1 MONTH DELETE
SETTINGS ttl_only_drop_parts = 1;
```

CREATE USER ...

```
CREATE USER IF NOT EXISTS vector IDENTIFIED WITH sha256_password BY '...';
GRANT INSERT ON metrics TO vector;
GRANT INSERT ON logs TO vector;

CREATE USER IF NOT EXISTS grafana IDENTIFIED WITH sha256_password BY '...';
GRANT SELECT ON metrics TO grafana;
GRANT SELECT ON logs TO grafana;
```

Плюсы и минусы

Плюсы

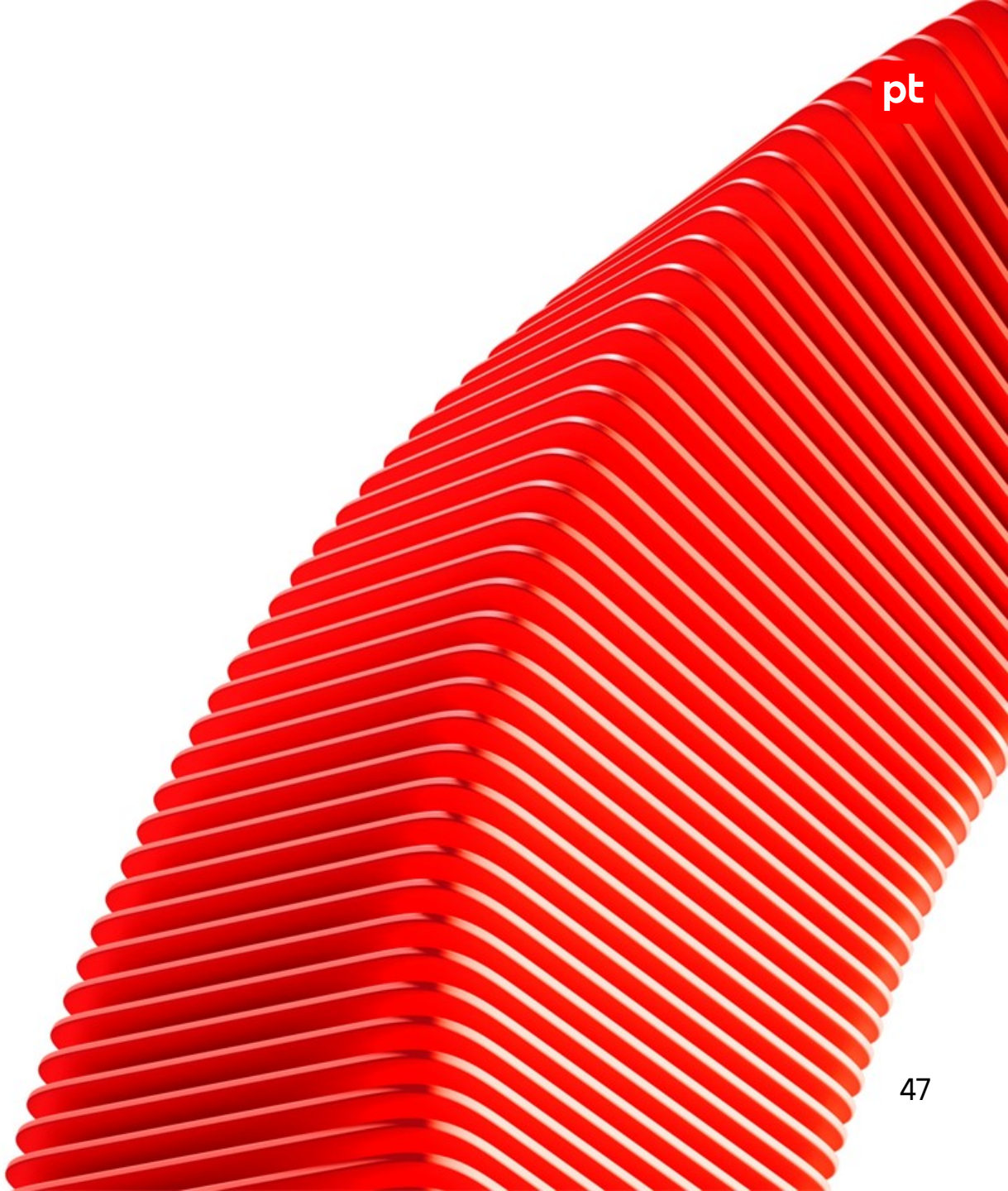
1. SQL с массой полезных функций
2. Эффективней OpenSearch в потреблении ресурсов
3. Native Interface (TCP) с Mutual TLS

Минусы

1. Непросто работать с шардированными данными
2. Нужно тюнить под структуру и объемы данных

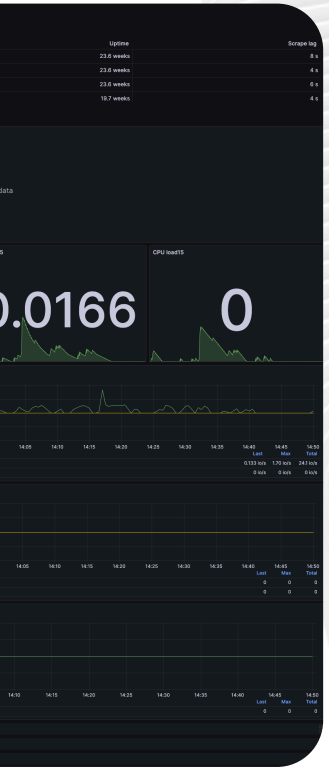
pt

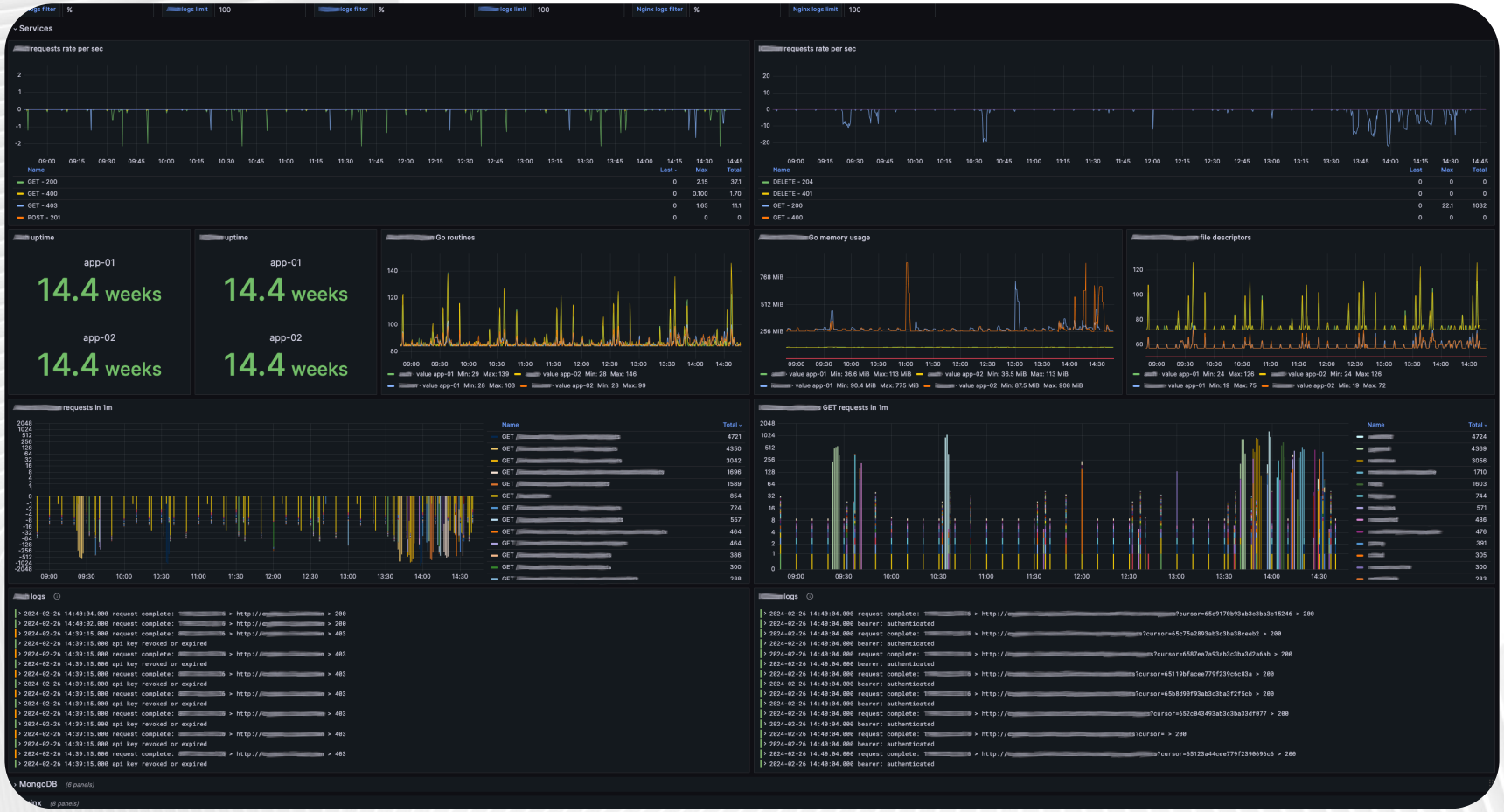
Grafana

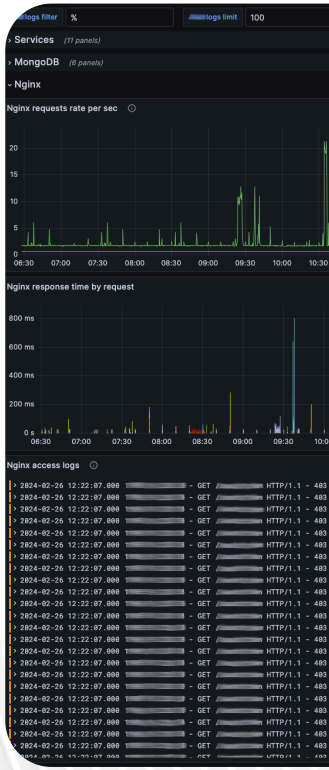


Dashboard «Hosts»

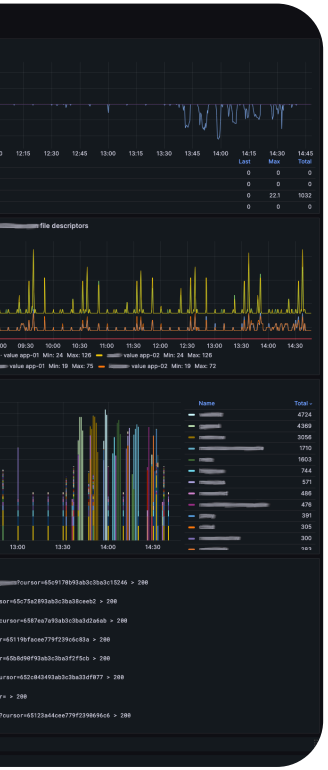
Dashboard «Product»

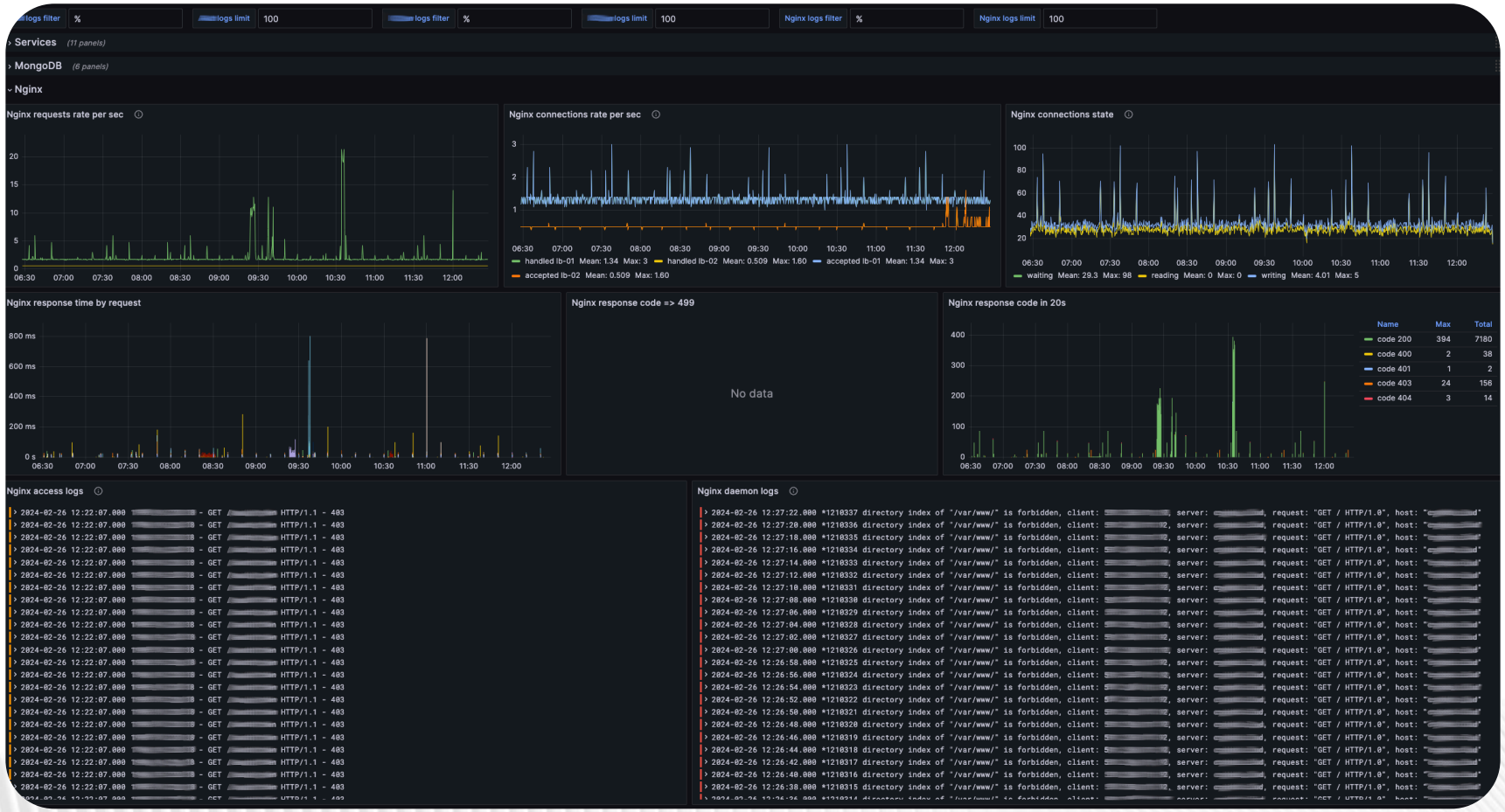






Dashboard «Product»





dmesg logs

```
SELECT timestamp AS log_time, message AS log, level, data
FROM logs
WHERE $__timeFilter(timestamp) AND app = 'dmesg'
ORDER BY timestamp DESC
LIMIT ${dmesg_logs_limit};
```

Nginx access logs

```
SELECT log_time, log, log_level AS level, host, data
FROM (
  SELECT timestamp AS log_time, message AS log,
    multiIf(
      toInt32OrZero(data['status']) >= 400 AND toInt32OrZero(data['status']) < 500, 'warning',
      toInt32OrZero(data['status']) >= 500, 'error',
      'info'
    ) AS log_level,
    host, data
  FROM logs
  WHERE $__timeFilter(timestamp) AND app = 'nginx' AND level = 'info'
  AND message LIKE '${nginx_logs_filter}'
  ORDER BY timestamp DESC
  LIMIT ${nginx_logs_limit}
);
```

Services requests

```
SELECT
  $__timeInterval(timestamp) AS timestamp,
  regexpExtract(data['request'], '^([A-Z]+\\s\\/[^?]+)[?\\s]') AS request,
  count(request) AS value
FROM logs
WHERE $__timeFilter(timestamp)
  AND app = 'nginx'
  AND data['request'] LIKE '%/api/%'
GROUP BY timestamp, request
ORDER BY timestamp;
```

Memory usage

```
SELECT $__timeInterval(timestamp) AS timestamp, value FROM metrics
WHERE $__timeFilter(timestamp) AND name = 'memory_used_bytes'
ORDER BY timestamp;

SELECT $__timeInterval(timestamp) AS timestamp, value FROM metrics
WHERE $__timeFilter(timestamp) AND name = 'memory_buffers_bytes'
ORDER BY timestamp;

SELECT $__timeInterval(timestamp) AS timestamp, value FROM metrics
WHERE $__timeFilter(timestamp) AND name = 'memory_cached_bytes'
ORDER BY timestamp;

SELECT $__timeInterval(timestamp) AS timestamp, value FROM metrics
WHERE $__timeFilter(timestamp) AND name = 'memory_shared_bytes'
ORDER BY timestamp;
```

CPU (user) usage rate

```
SELECT $__timeInterval(timestamp) AS timestamp, sum(delta) / $__interval_s AS value
FROM (
  SELECT timestamp, value AS current,
    lagInFrame(current) OVER (ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS previous,
    current - previous AS delta
  FROM (
    SELECT timestamp, sum(value) AS value
    FROM metrics
    WHERE name = 'cpu_seconds_total' AND tags['mode'] = 'user'
      AND timestamp >= addSeconds($__fromTime, -$__interval_s)
      AND timestamp <= toStartOfInterval($__toTime, INTERVAL $__interval_s SECOND)
    GROUP BY timestamp
    ORDER BY timestamp
  )
)
WHERE $__timeFilter(timestamp)
GROUP BY timestamp
ORDER BY timestamp;
```

Плюсы и минусы

Плюсы

1. «ClickHouse data source plugin for Grafana» с поддержкой Mutual TLS
2. Вывод любых SQL-данных практически в любом формате
3. Библиотека панелей
4. Встроенный алертинг через Prometheus Alertmanager

Минусы

1. Недостаточно функционала в работе с логами

Хабр

- Clickhouse, Grafana и 3000 графиков. Как построить систему быстрых дашбордов | 20.11.2023, Валентин Борисов, Ozon Tech
- Vector.dev: затащили в PoC | 07.12.2023, Ефимцева Наталия, ГК ICL

ClickHouse | Blog

- Working with Time Series Data in ClickHouse | 03.01.2023, Camilo Sierra
- Essential Monitoring Queries - part 1 - INSERT Queries | 28.12.2022, Camilo Sierra
- Essential Monitoring Queries - part 2 - SELECT Queries | 03.01.2023, Camilo Sierra
- Building an Observability Solution with ClickHouse - Part 1 - Logs | 11.01.2023, Dale McDiarmid
- Building an Observability Solution with ClickHouse - Part 2 - Traces | 29.03.2023, Dale McDiarmid
- Visualizing Data with ClickHouse - Part 1 - Grafana | 07.10.2023, Dale McDiarmid

ClickHouse | Docs

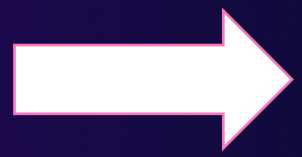
- Integrating Vector with ClickHouse

Материал

github.com/magna-z/distributed-observability

Спасибо!

Оставить отзыв о докладе



Максим Залысин

Head of DevOps | Positive Technologies

DevOps Conf 2024