Deckhouse
ФЛАНТ

Высоконагруженные системы мониторинга

Deckhouse
ФЛАНТ

Владимир Гурьянов

Технический директор Deckhouse
Observability

- С **2011** занимаюсь эксплуатацией приложений
- С **2021** работаю в компании «Флант»
- С **2023** занимаюсь развитием направления Observability

Deckhouse Observability Platform DOP

Единая платформа вместо зоопарка систем мониторинга

10+

тысяч серверов на мониторинге

100+

миллионов метрик в минуту

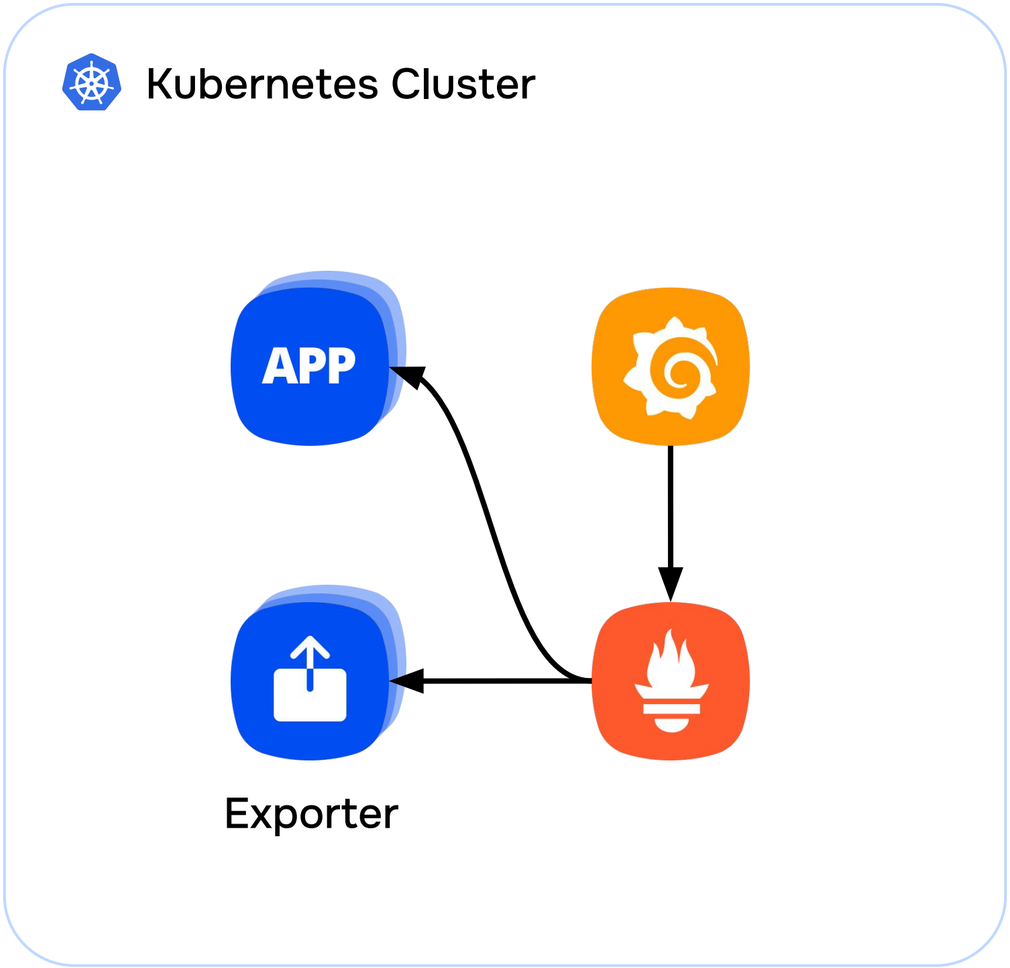
22+

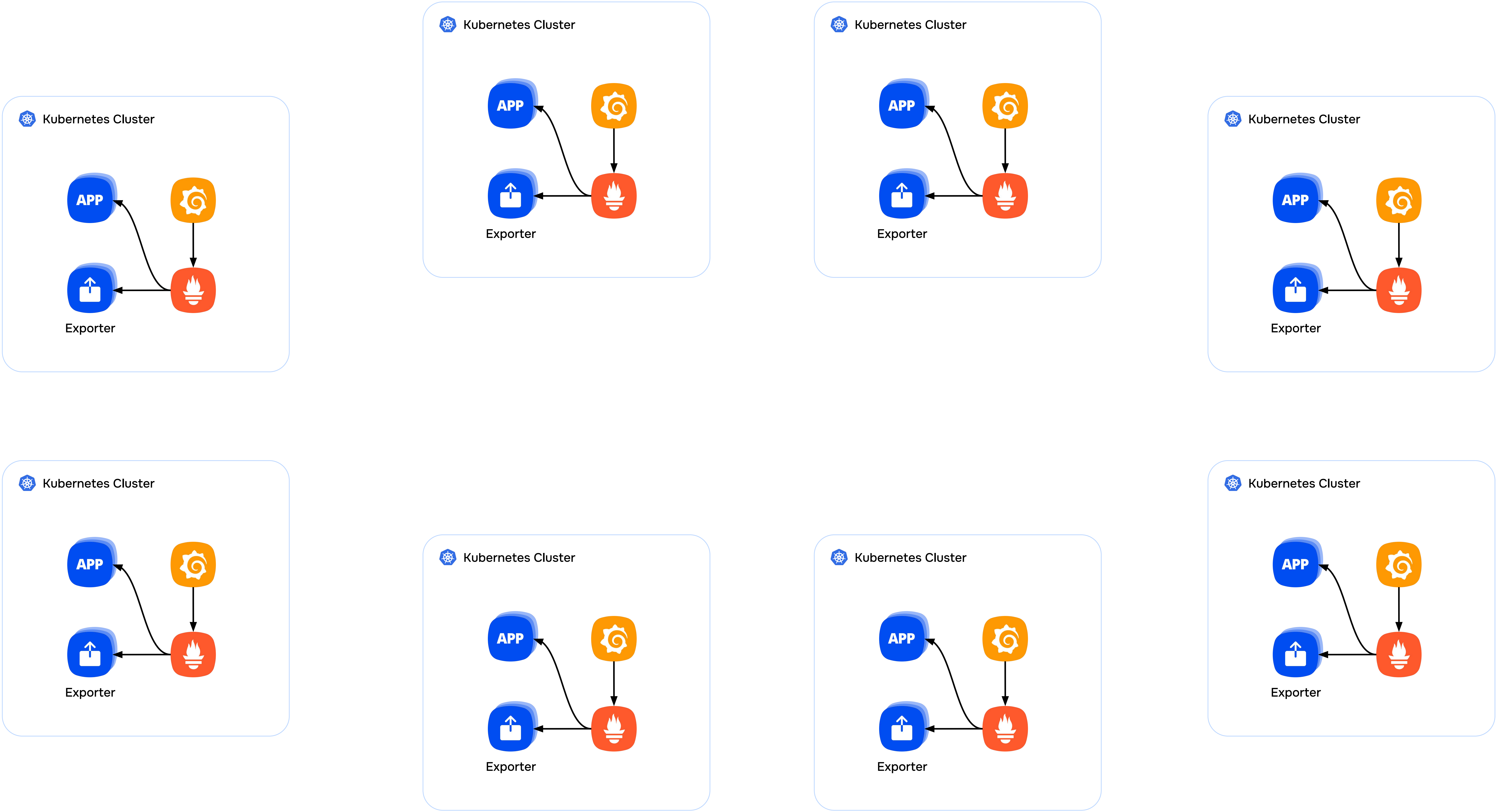
тысяч приложений в production

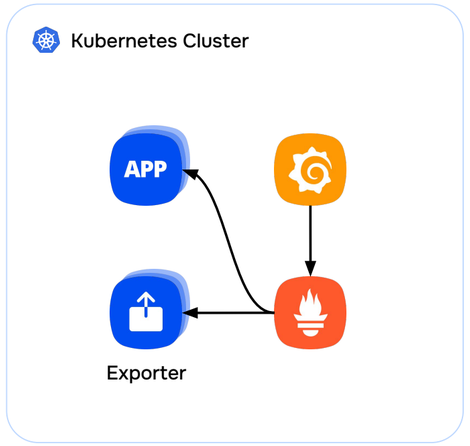
600+

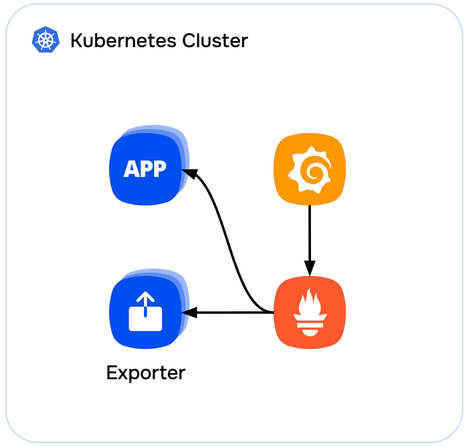
кластеров на мониторинге

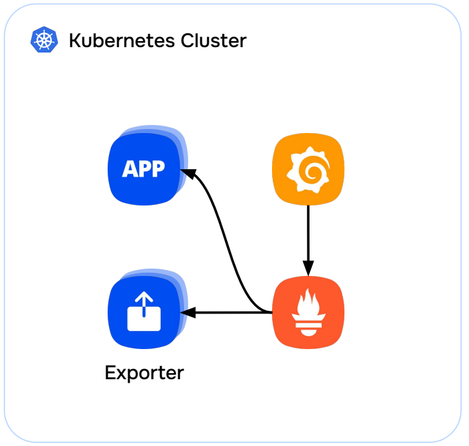
Deckhouse Observability Platform

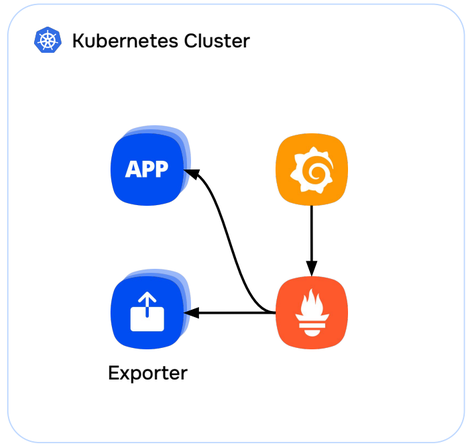

Kubernetes Cluster
APP
Exporter


Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter

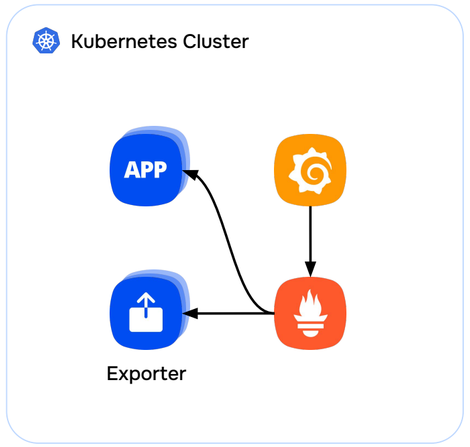


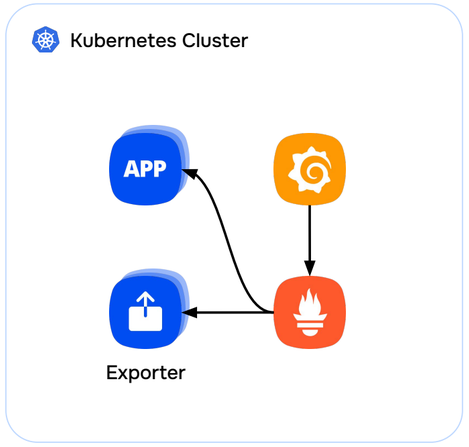


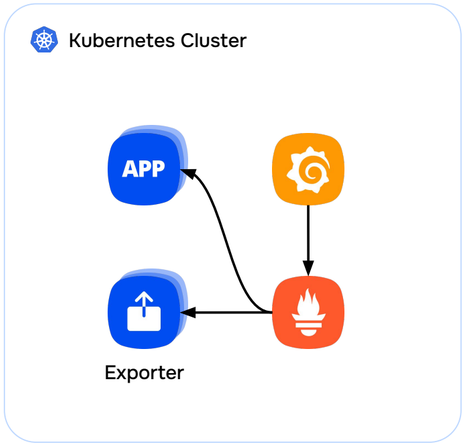


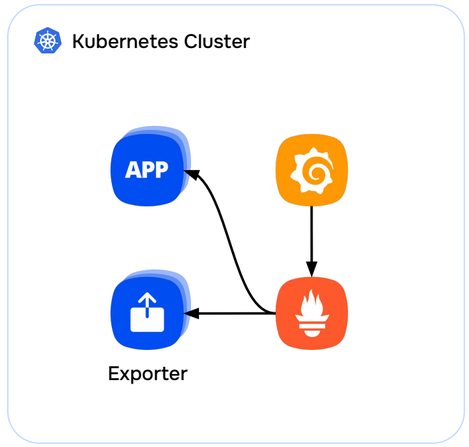


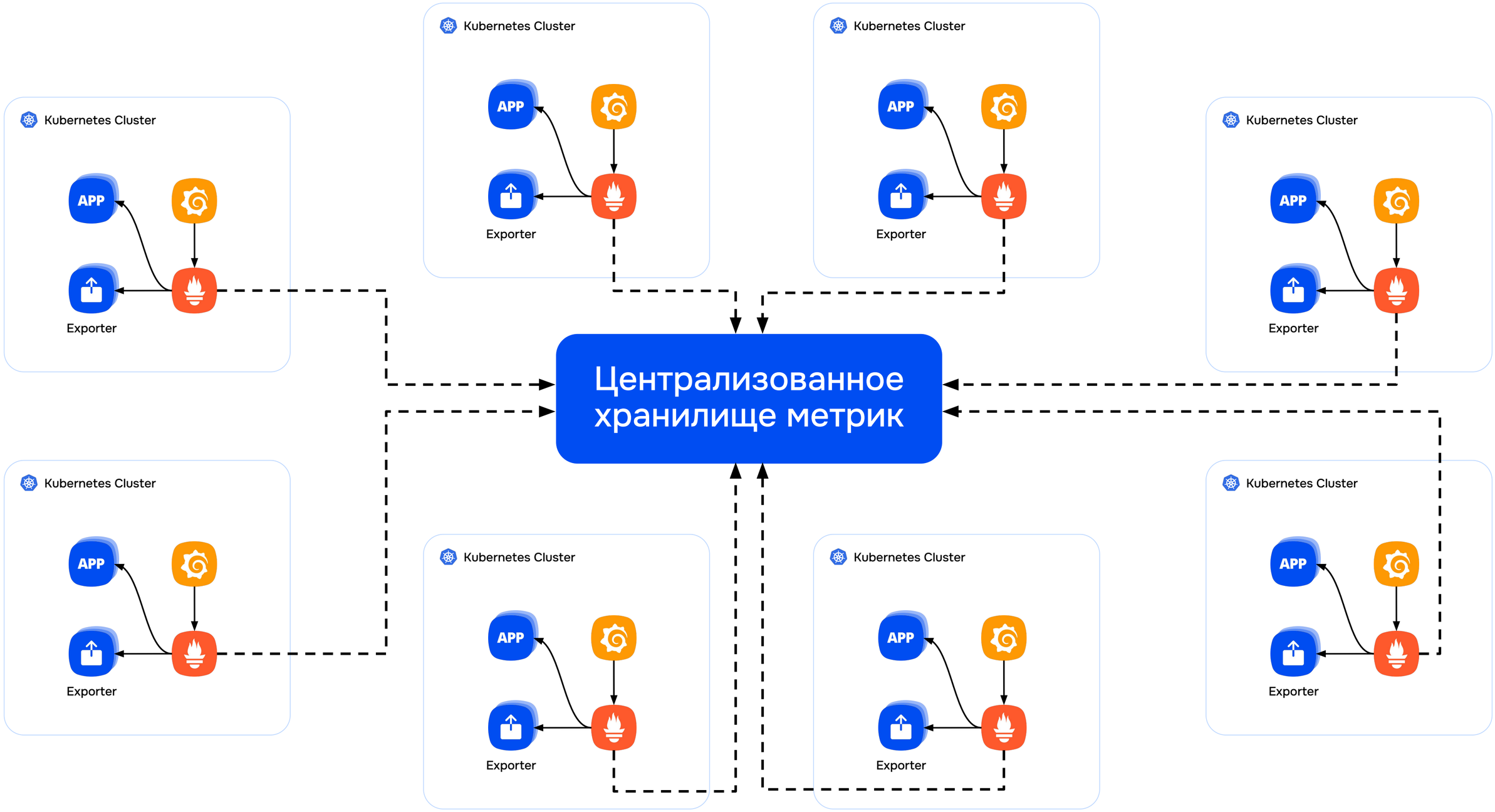
Централизованное хранилище метрик









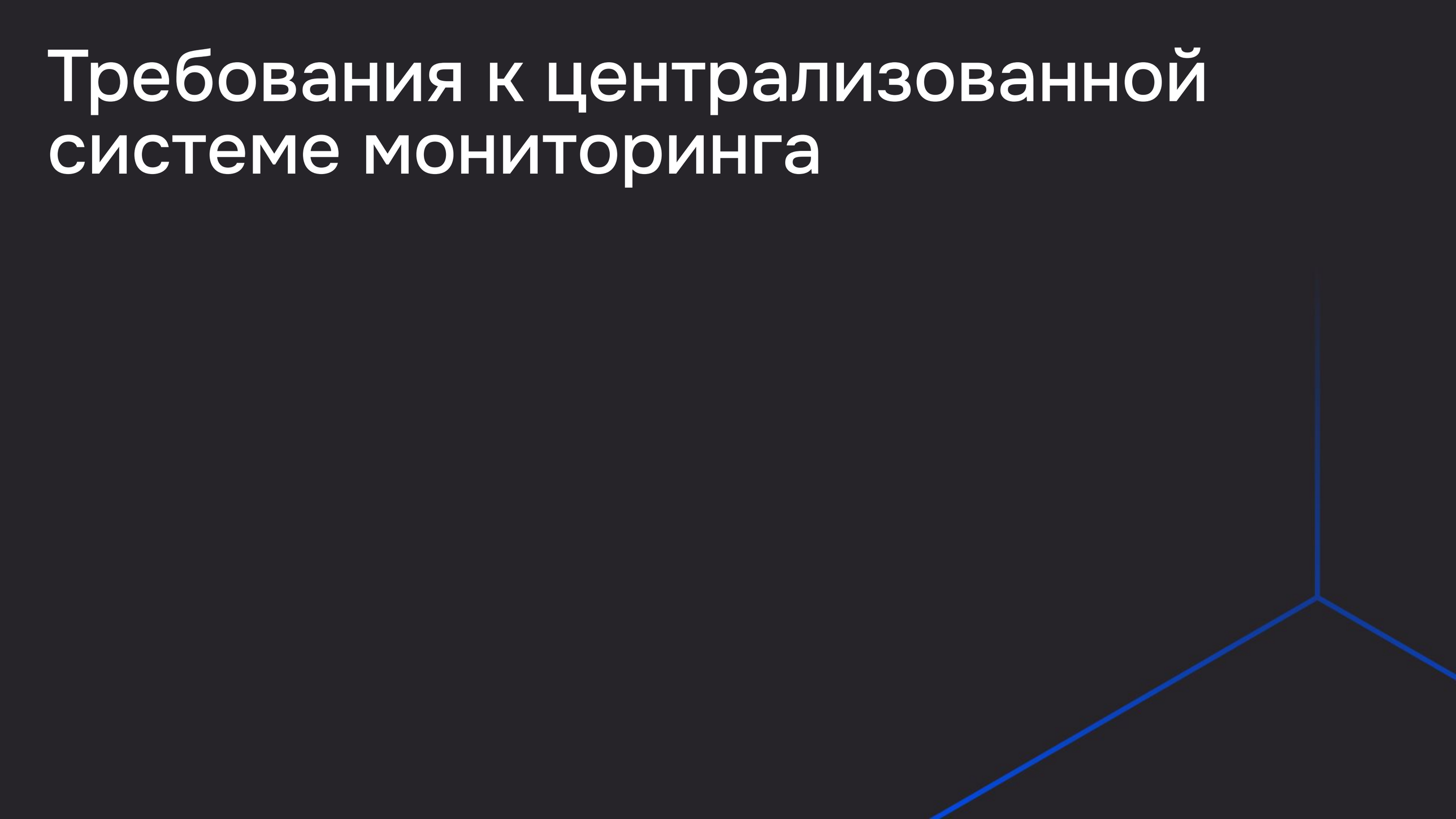

Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter
Централизованное хранилище метрик
Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter

1 маленький кластер → 800 000+

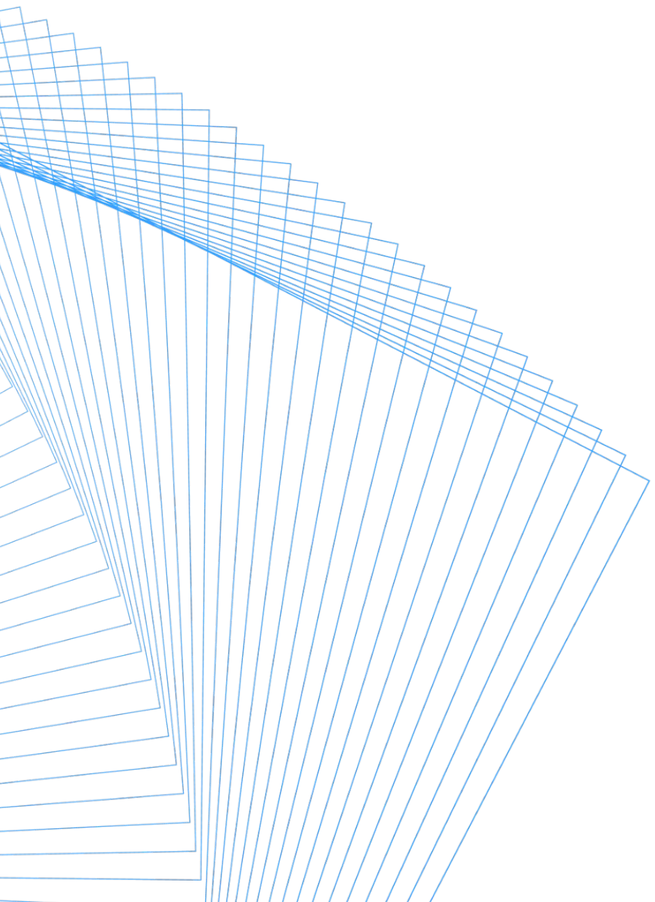
1 маленький кластер → 800 000+

20 кластеров = 800 000 × 20 = 16 000 000

Требования к централизованной системе мониторинга

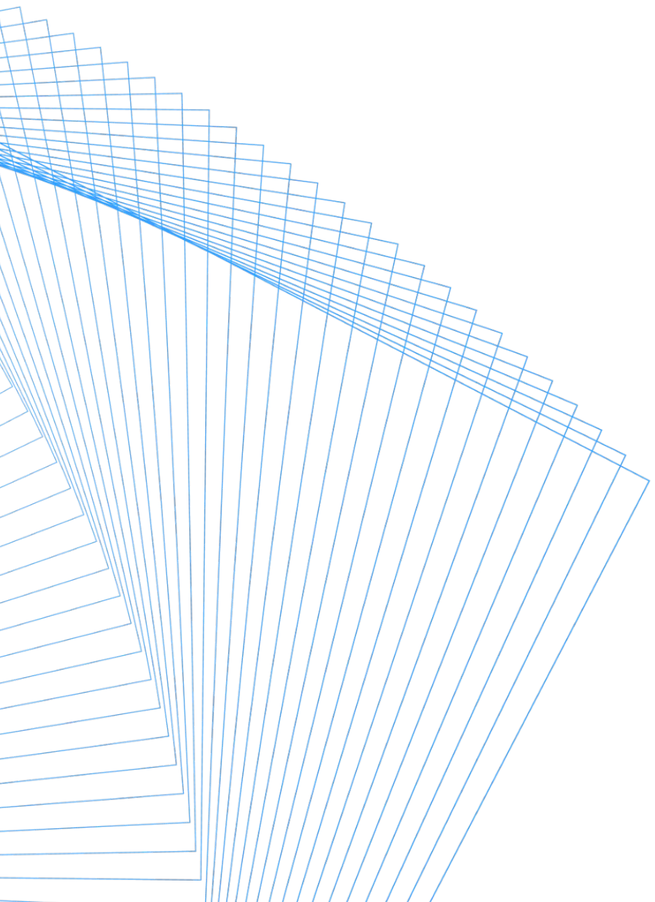


Требования



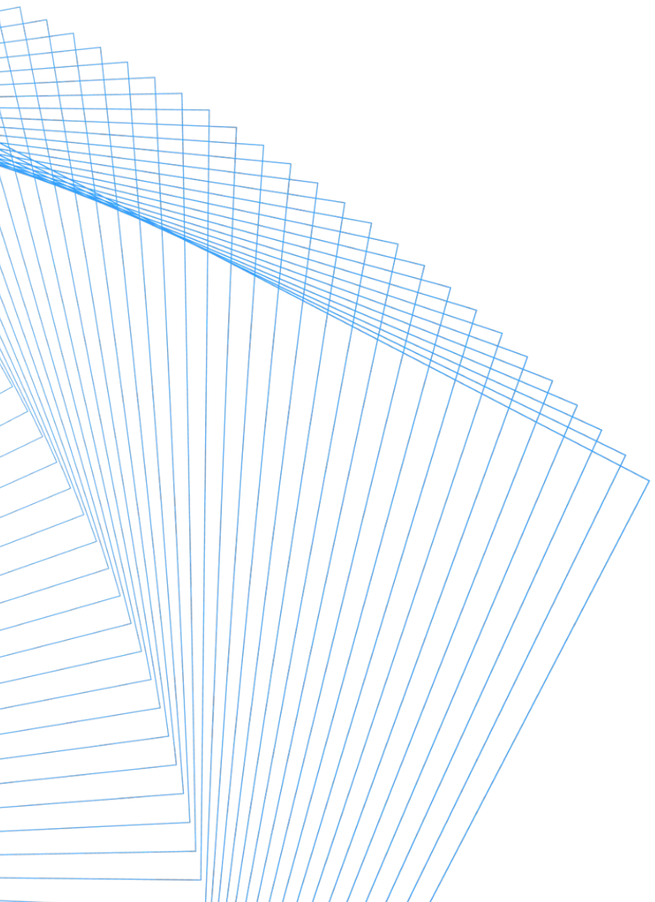
1 Отказоустойчивость

Требования



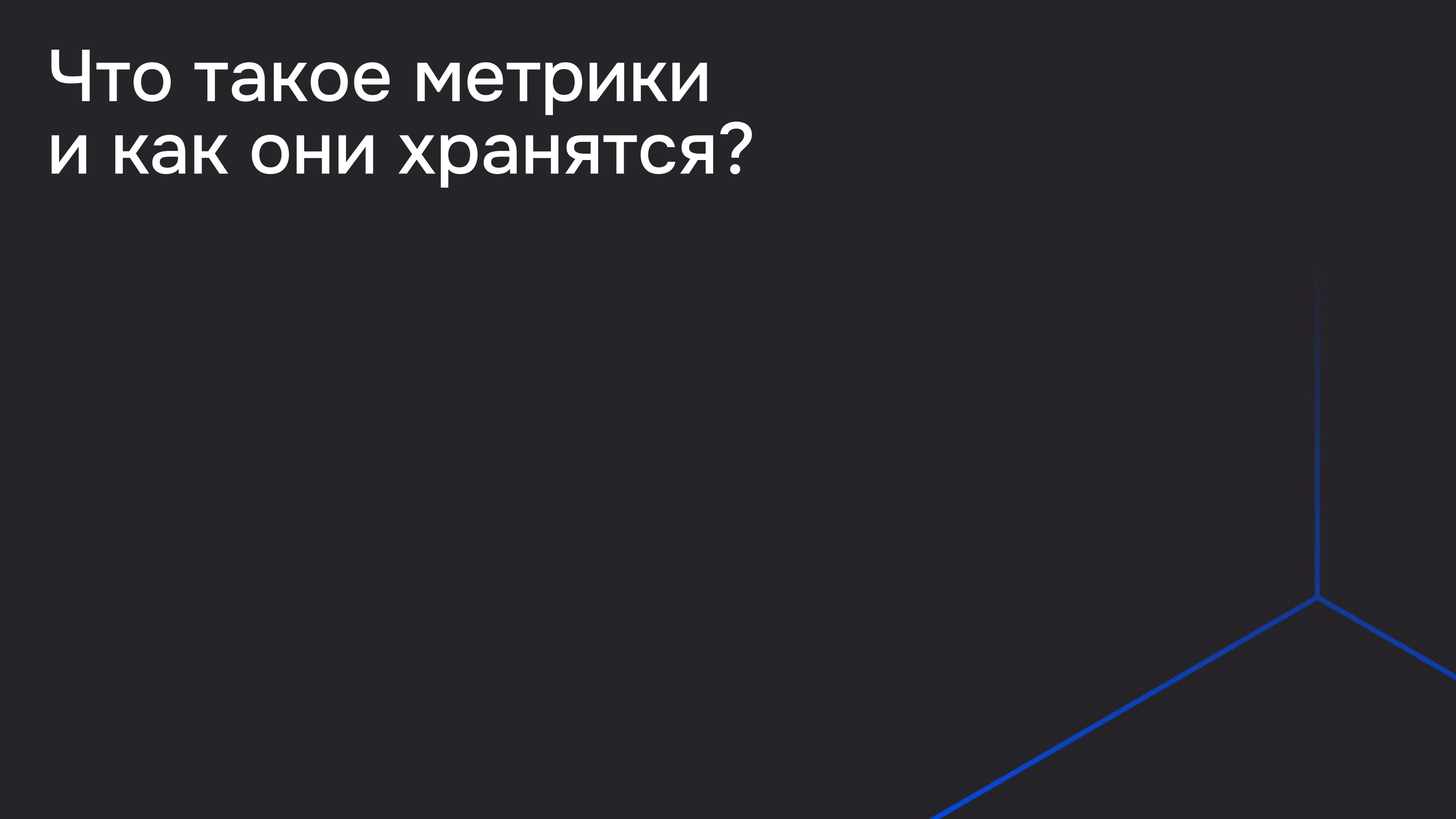
1. Отказоустойчивость
2. Шардирование данных

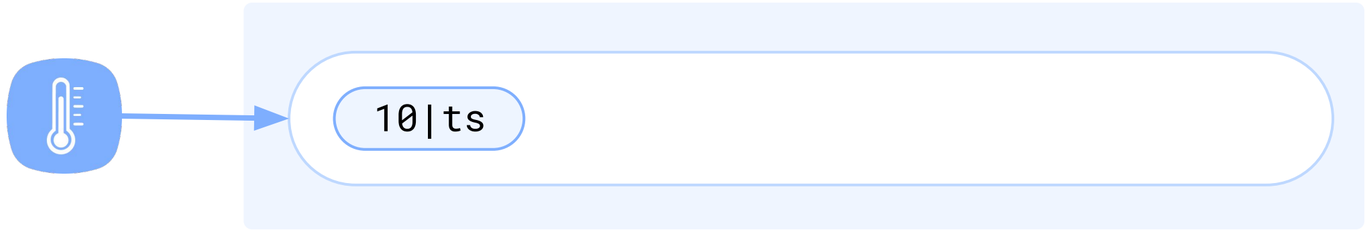
Требования



1. Отказоустойчивость
2. Шардирование данных
3. Минимальная стоимость

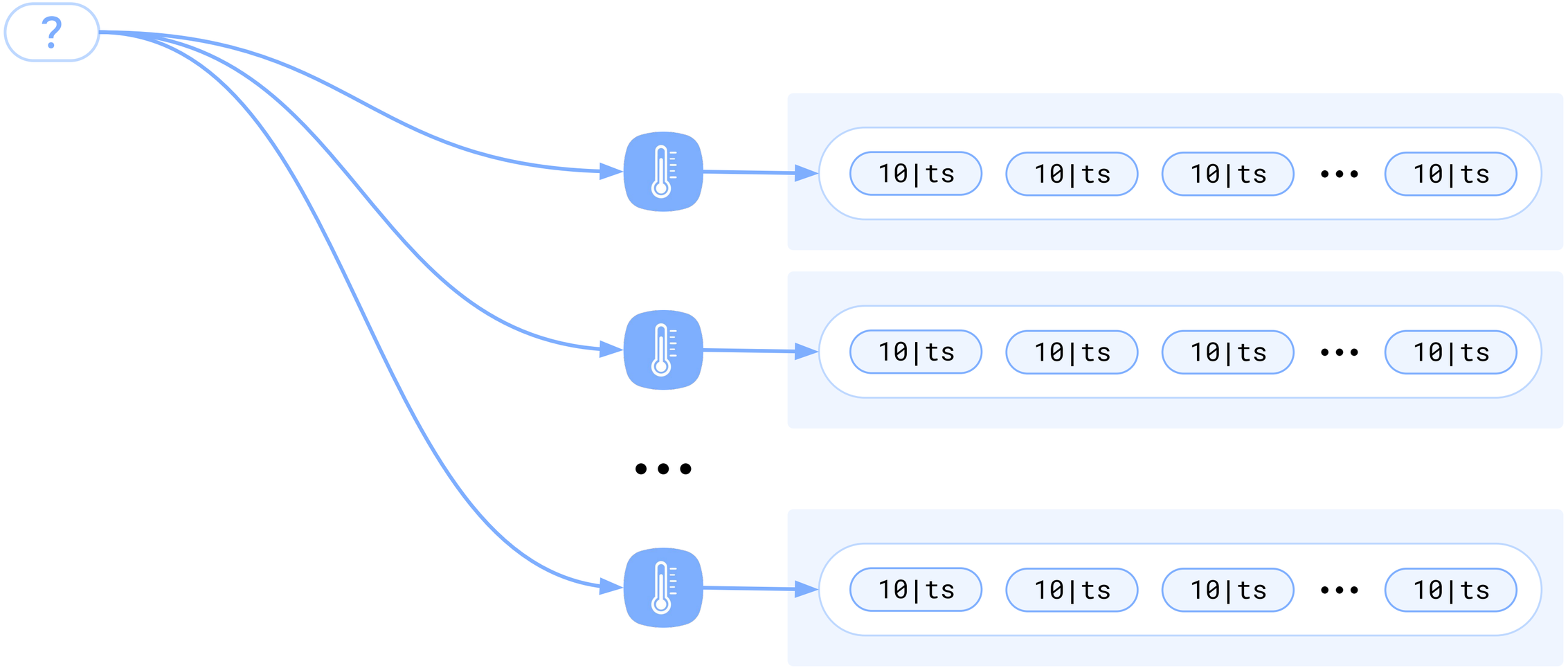
Что такое метрики и как они хранятся?




10|ts


10|ts
10|ts
10|ts
•••
10|ts


10|ts
10|ts
10|ts
10|ts
10|ts
10|ts
10|ts
10|ts
10|ts
10|ts
10|ts
10|ts
10|ts
10|ts


?
10|ts
10|ts
10|ts
10|ts
10|ts
10|ts
10|ts
10|ts
10|ts
10|ts

LabelSet

LabelSet

```
{
  ключ:  значение,
  ключ:  значение,
  …
}
```

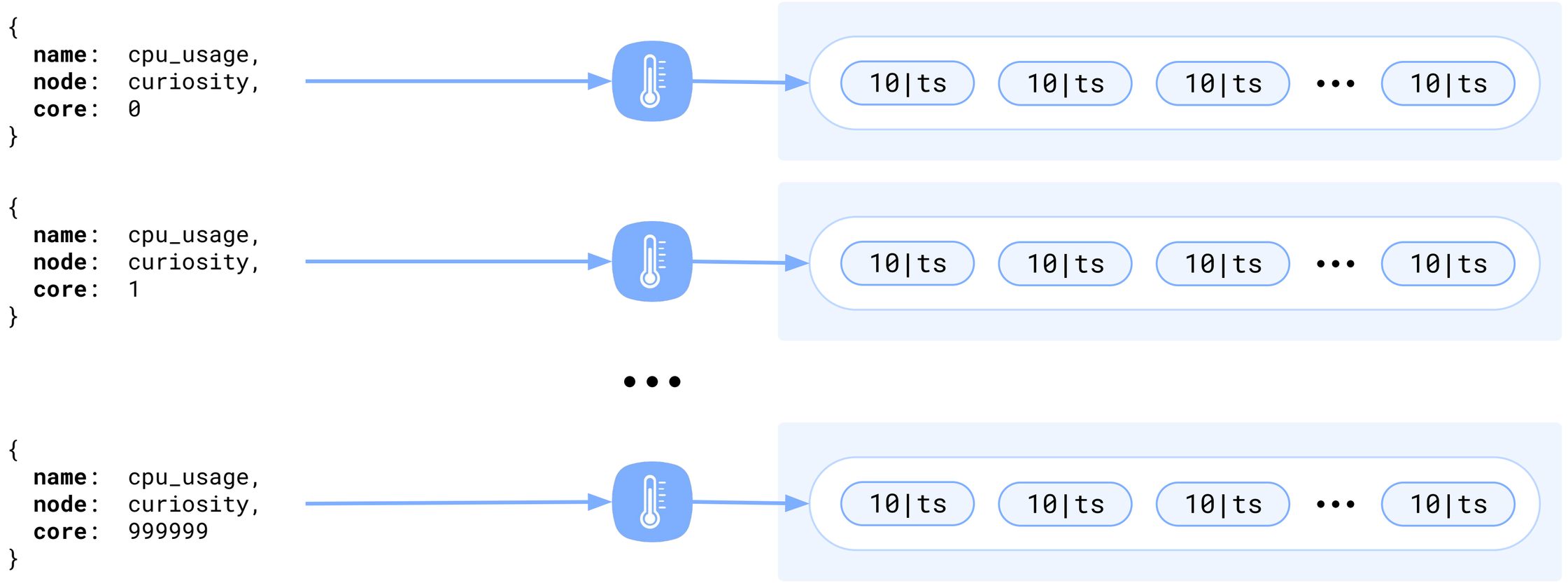
LabelSet

```
{
  ключ:  значение,
  ключ:  значение,
  …
}
```

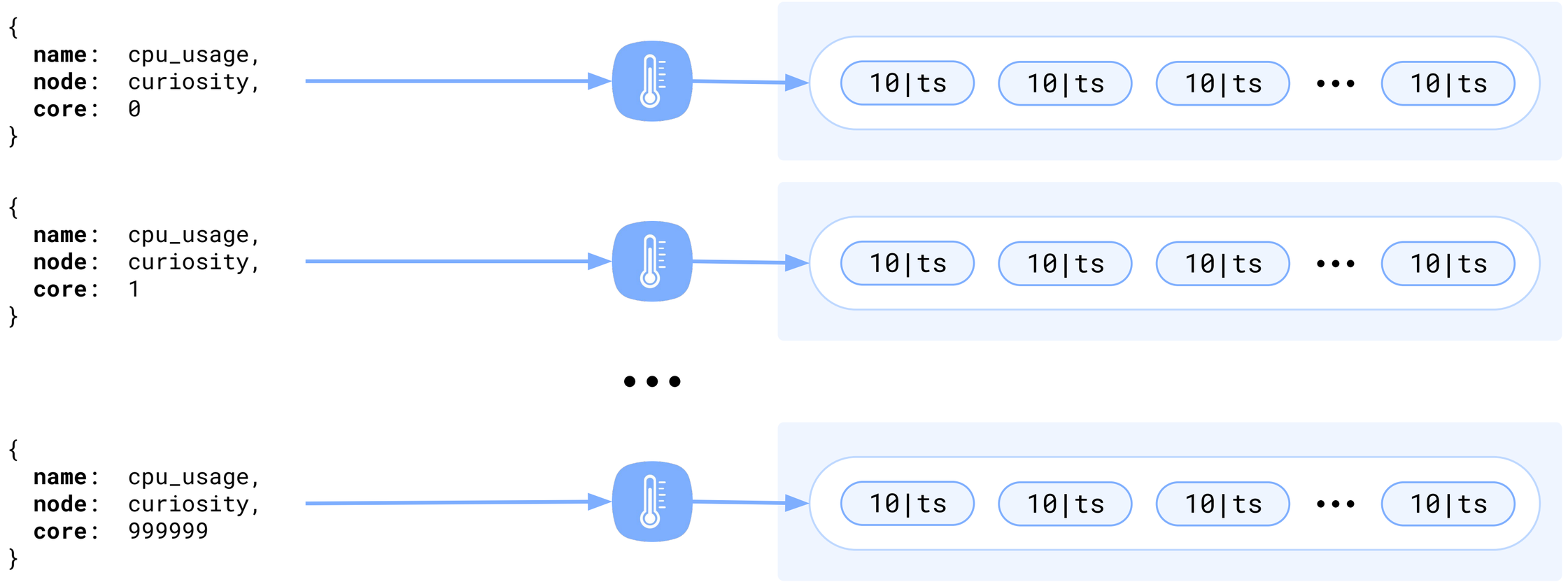
```
{
  name:  cpu_usage,
  node:  curiosity,
  core:  4
}
```

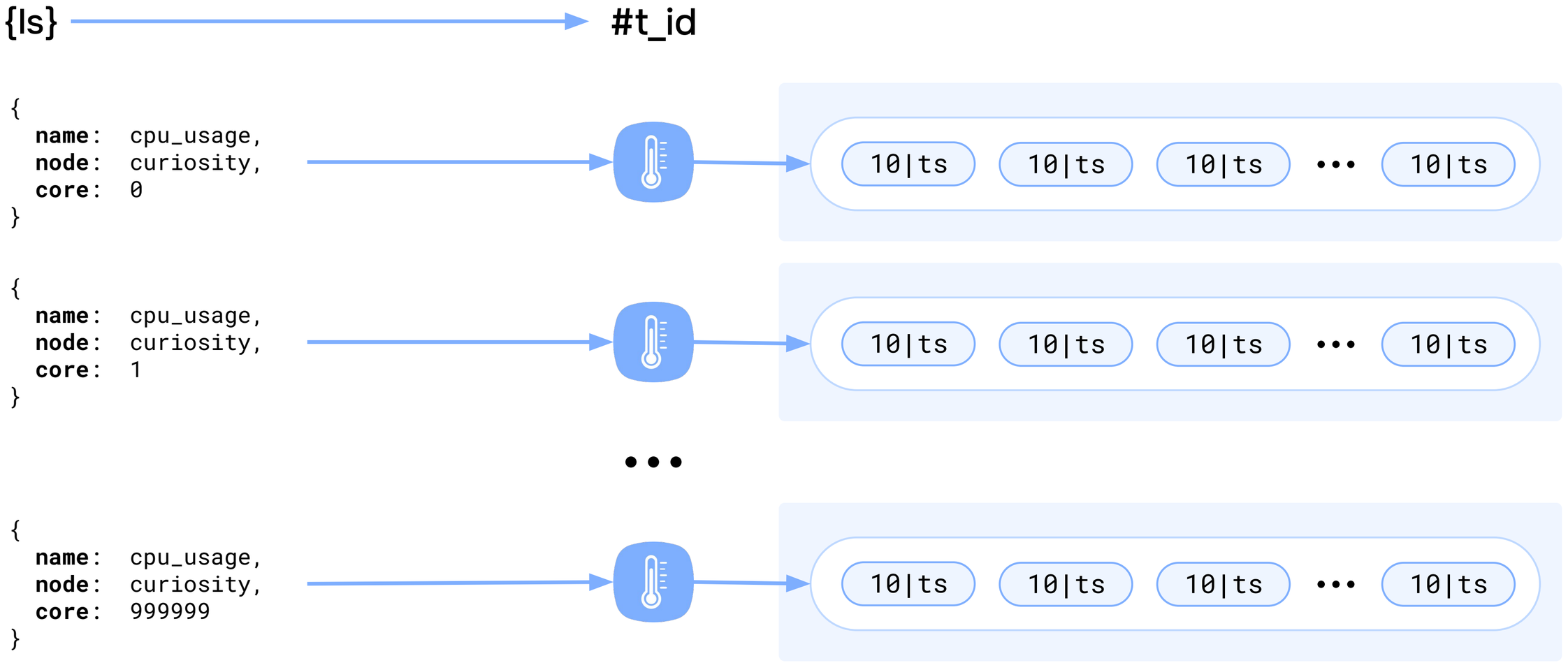
LabelSet

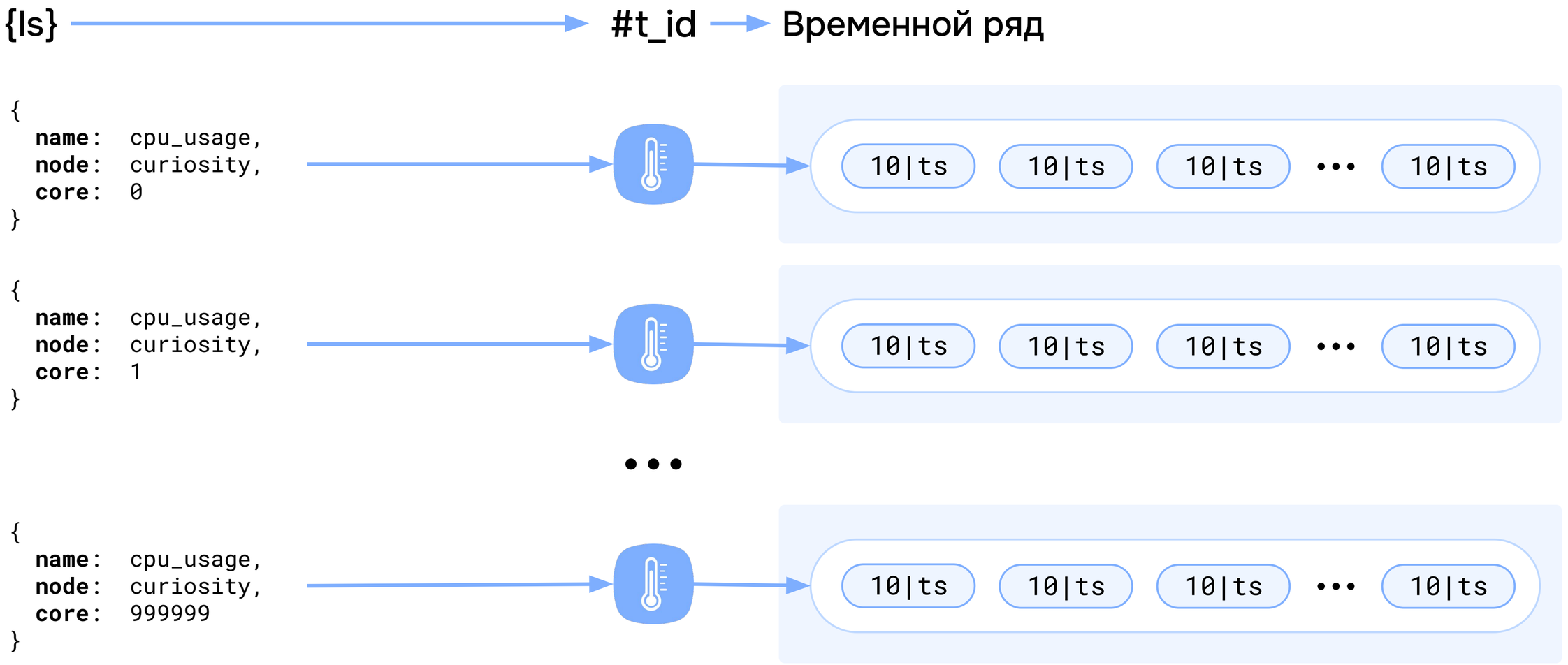
LabelSet


{
name: cpu_usage,
node: curiosity,
core: 0
}
10|ts
10|ts
10|ts
10|ts
{
name: cpu_usage,
node: curiosity,
core: 1
}
10|ts
10|ts
10|ts
10|ts
{
name: cpu_usage,
node: curiosity,
core: 999999
}
10|ts
10|ts
10|ts
10|ts

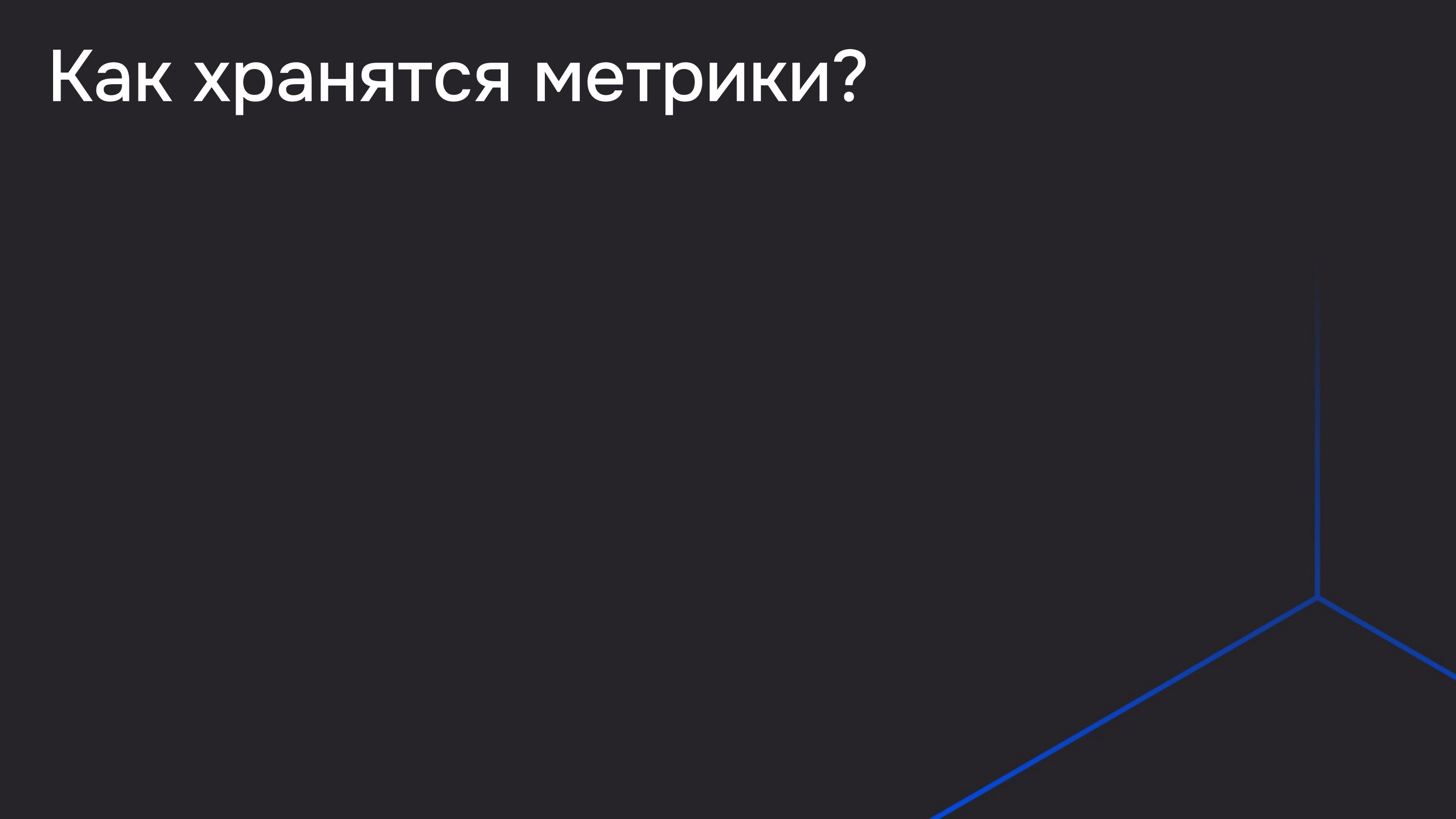
{ls}

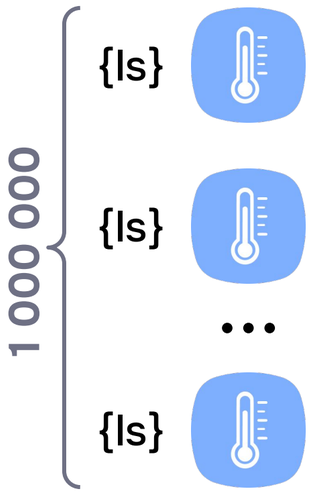

{
name: cpu_usage,
node: curiosity,
core: 0
}
10|ts
10|ts
10|ts
•••
10|ts
{
name: cpu_usage,
node: curiosity,
core: 1
}
10|ts
10|ts
10|ts
•••
10|ts
{
name: cpu_usage,
node: curiosity,
core: 999999
}
10|ts
10|ts
10|ts
•••
10|ts


{ls}
#t_id
{
name: cpu_usage,
node: curiosity,
core: 0
}
10|ts
10|ts
10|ts
•••
10|ts
{
name: cpu_usage,
node: curiosity,
core: 1
}
10|ts
10|ts
10|ts
•••
10|ts
•••
{
name: cpu_usage,
node: curiosity,
core: 999999
}
10|ts
10|ts
10|ts
•••
10|ts


{ls}
#t_id
Временной ряд
{
name: cpu_usage,
node: curiosity,
core: 0
}
{
name: cpu_usage,
node: curiosity,
core: 1
}
{
name: cpu_usage,
node: curiosity,
core: 999999
}
10|ts
10|ts
10|ts
10|ts

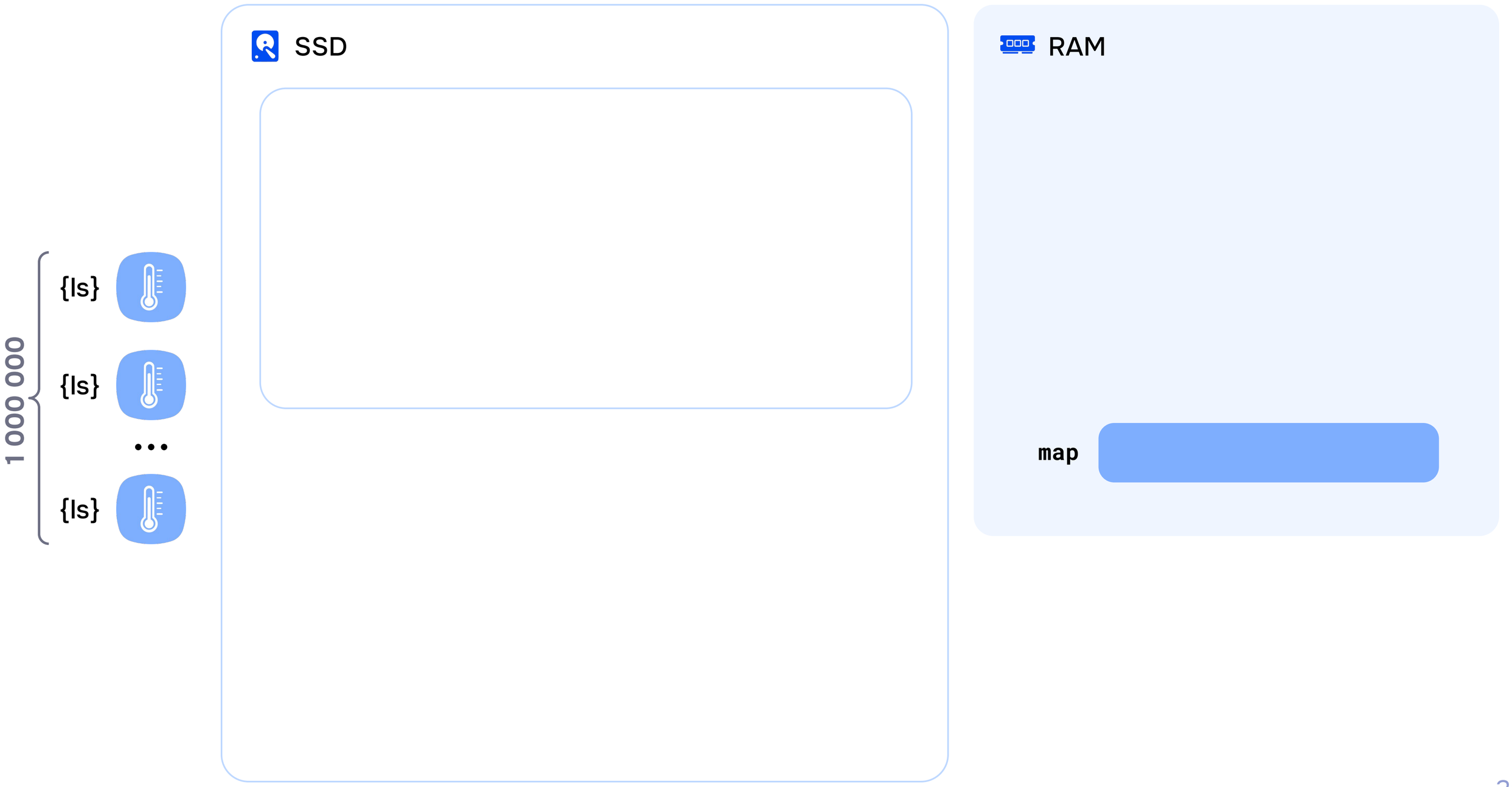
Как хранятся метрики?





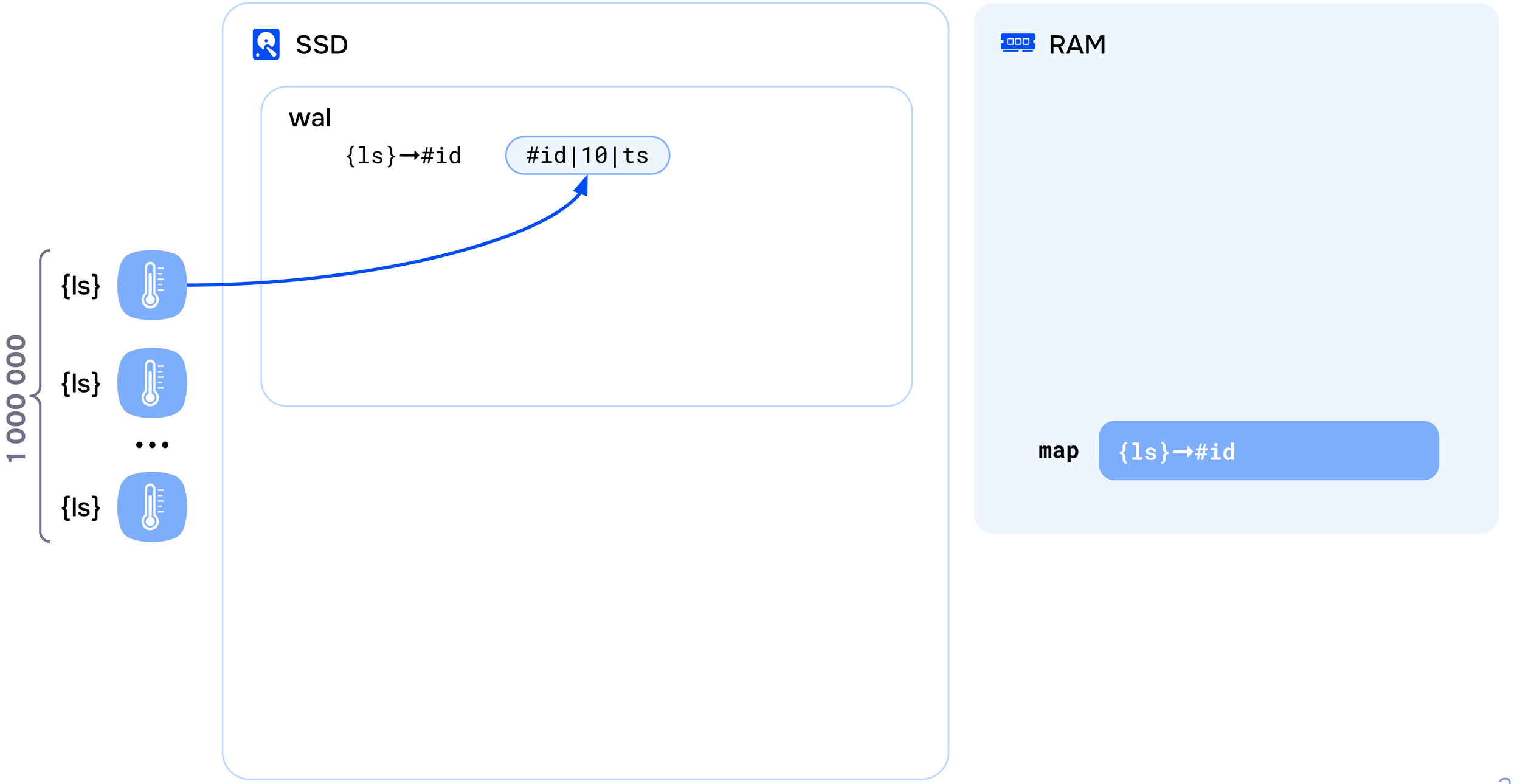
SSD

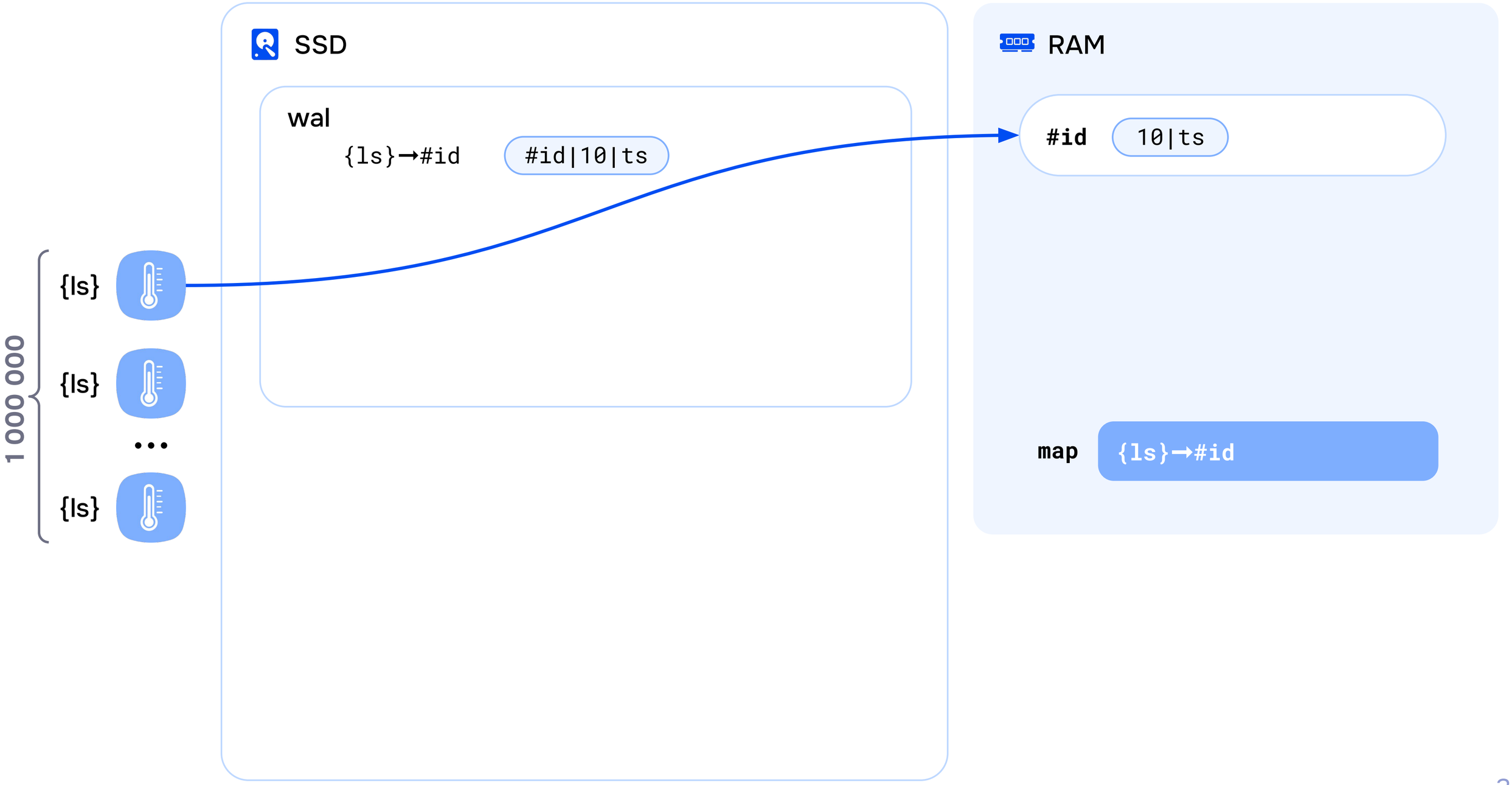
RAM


SSD
RAM
map
{ls}
{ls}
{ls}
1 000 000


SSD
RAM
{ls}
{ls}
{ls}
1 000 000
map
{ls}→#id


SSD
wal
{ls}→#id
{ls}
{ls}
{ls}
1 000 000
RAM
map
{ls}→#id


SSD
wal
{ls}→#id
#id|10|ts
RAM
map
{ls}→#id
1 000 000
{ls}
{ls}
{ls}


SSD
wal
{ls}→#id
#id|10|ts
RAM
#id
10|ts
map
{ls}→#id
1 000 000
{ls}
{ls}
{ls}

1 000 000

{ls}

{ls}

...

{ls}

SSD

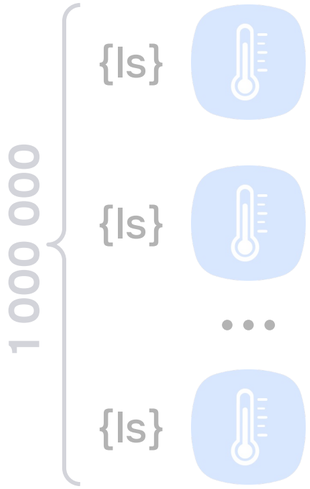
wal

{ls}→#id #id|10|ts

RAM

#id 10|ts

map {ls}→#id



SSD

wal

{ls}→#id #id|10|ts

RAM

#id 10|ts

map {ls}→#id ••• {ls}→#id

1 000 000

{ls}

{ls}

•••

{ls}

SSD

wal

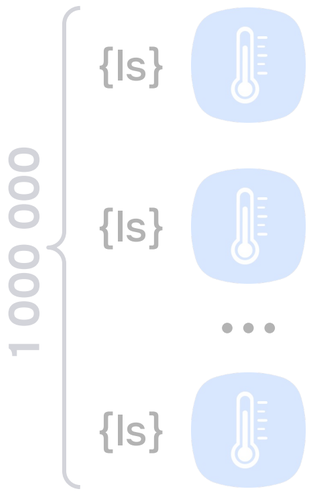
{ls}→#id #id|10|ts •••

{ls}→#id #id|10|ts

RAM

#id 10|ts

map {ls}→#id ••• {ls}→#id



SSD

wal

{ls}→#id #id|10|ts •••

{ls}→#id #id|10|ts

RAM

#id 10|ts

#id 10|ts

•••

#id 10|ts

map {ls}→#id ••• {ls}→#id

1 000 000

{ls}

{ls}

•••

{ls}

SSD

wal

{ls}→#id #id|10|ts •••

{ls}→#id #id|10|ts #id|10|ts

#id|10|ts #id|10|ts •••

#id|10|ts

RAM

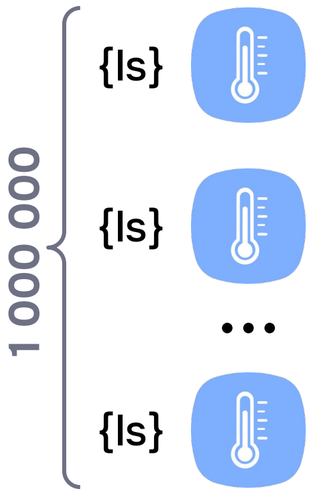
#id 10|ts ••• 10|ts

#id 10|ts ••• 10|ts

• • •

#id 10|ts ••• 10|ts

map {ls}→#id ••• {ls}→#id



SSD

wal

{ls}→#id #id|10|ts •••

{ls}→#id #id|10|ts #id|10|ts

#id|10|ts #id|10|ts •••

#id|10|ts

RAM

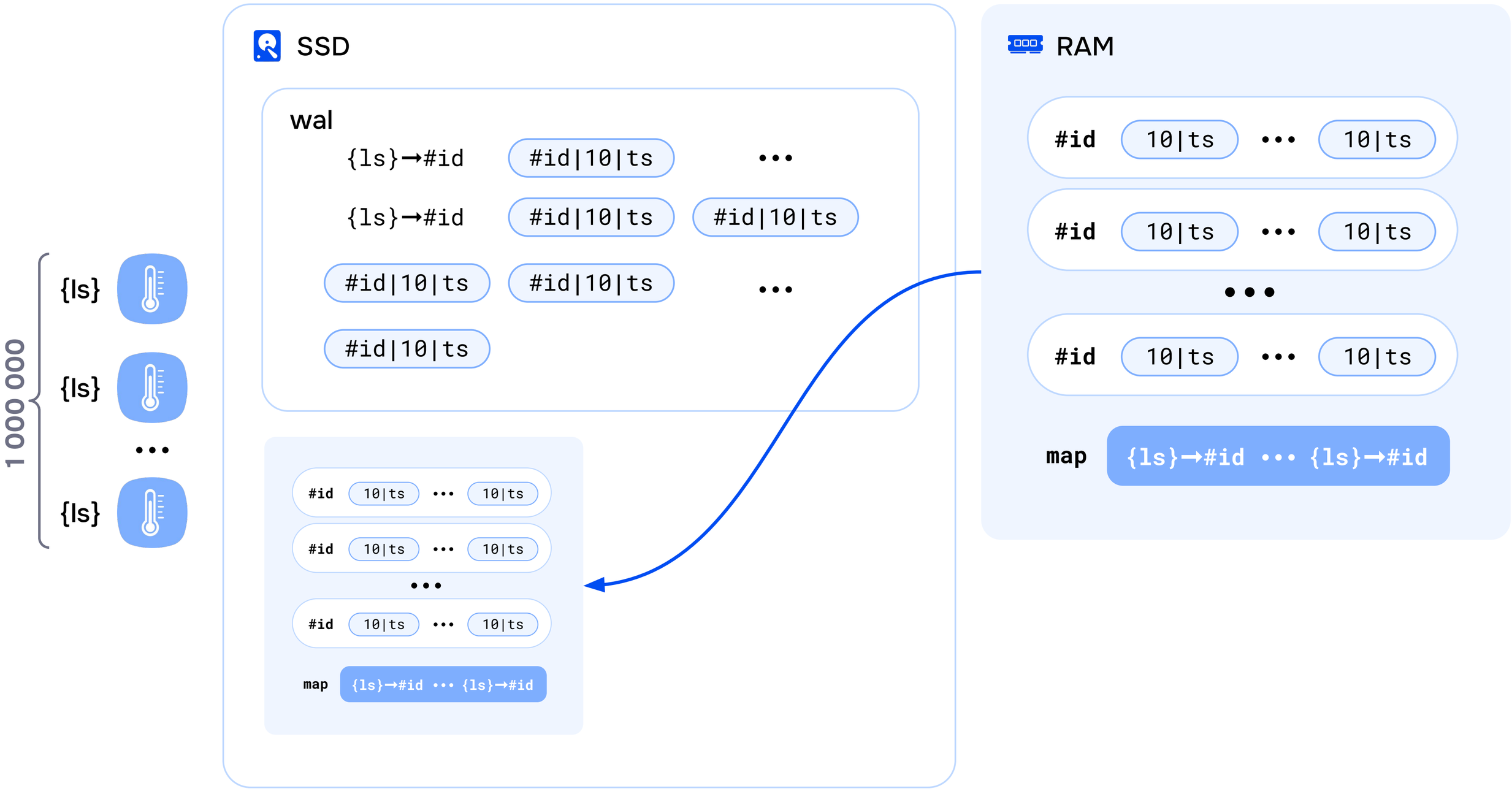
#id 10|ts ••• 10|ts

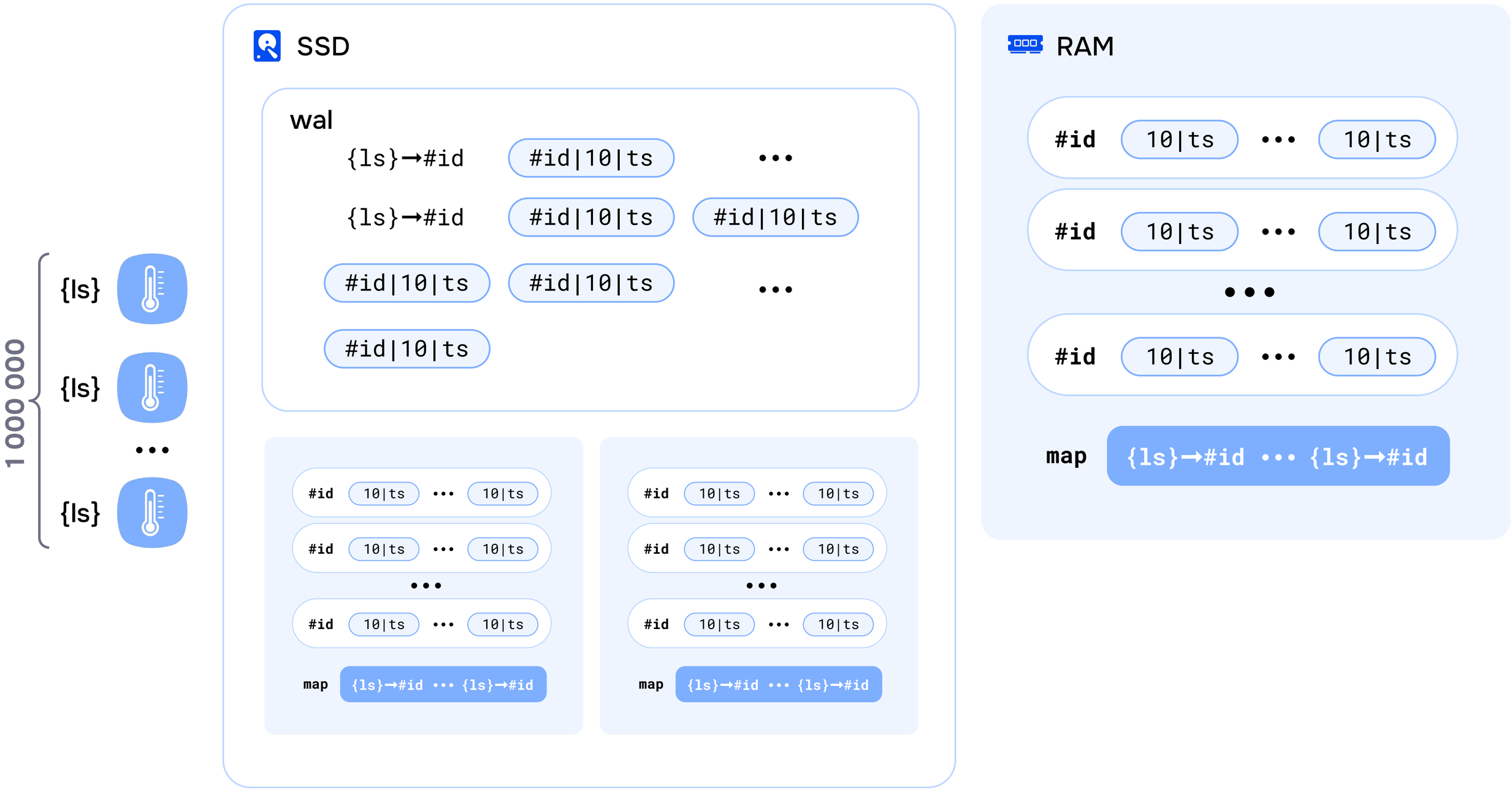
#id 10|ts ••• 10|ts

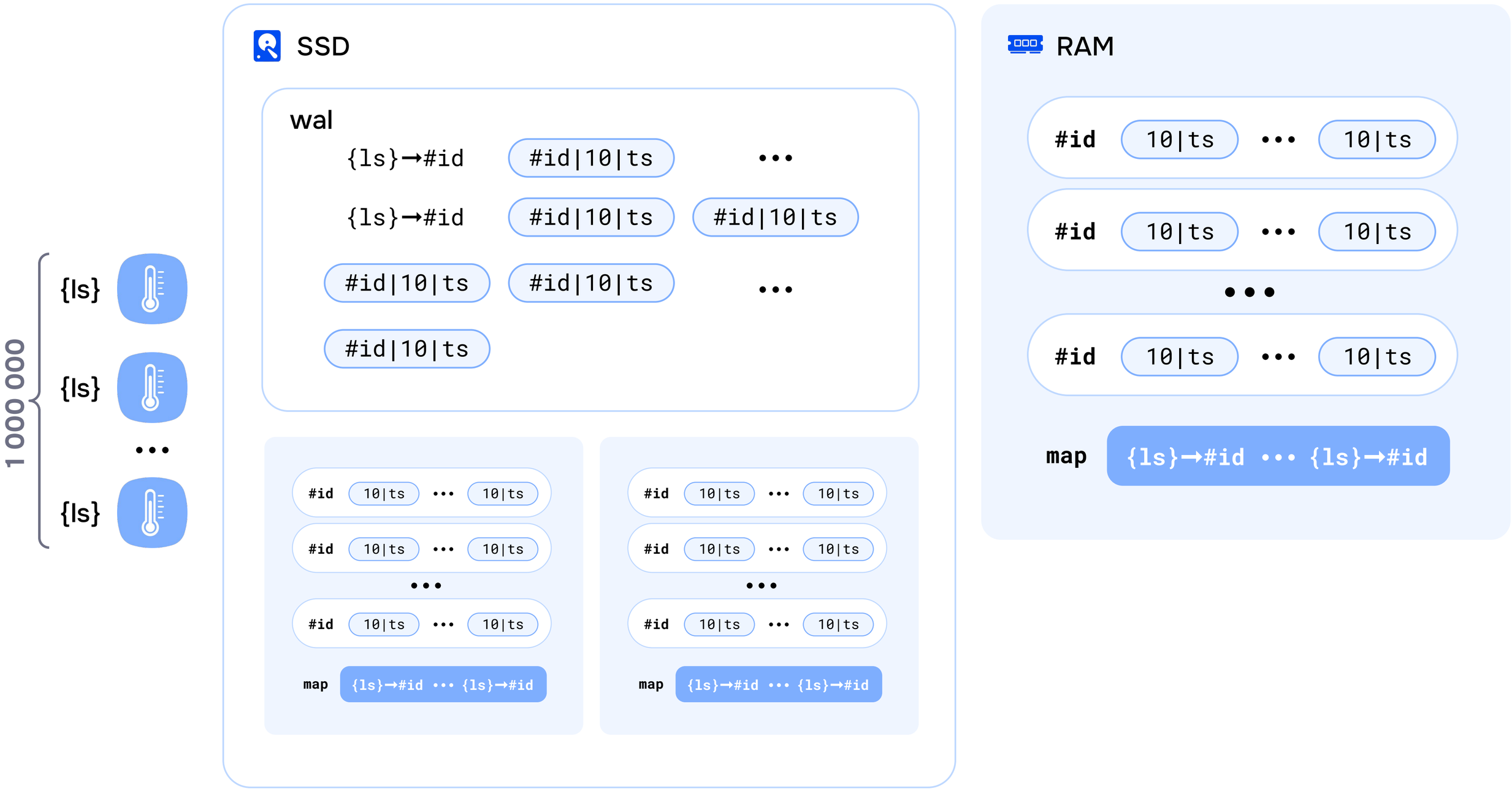
•••

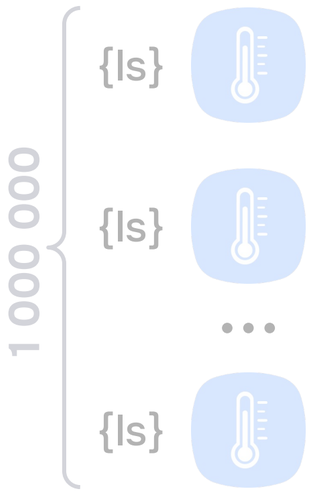
#id 10|ts ••• 10|ts

map {ls}→#id ••• {ls}→#id


SSD
wal
{ls}→#id
#id|10|ts
{ls}→#id
#id|10|ts
#id|10|ts
#id|10|ts
#id|10|ts
#id|10|ts
#id|10|ts
#id
10|ts
10|ts
#id
10|ts
10|ts
#id
10|ts
10|ts
map
{ls}→#id ••• {ls}→#id
RAM
#id
10|ts
10|ts
#id
10|ts
10|ts
#id
10|ts
10|ts
map
{ls}→#id ••• {ls}→#id
1 000 000
{ls}
{ls}
{ls}


1 000 000
{ls}
{ls}
{ls}
SSD
wal
{ls}→#id
#id|10|ts
{ls}→#id
#id|10|ts
#id|10|ts
#id|10|ts
#id|10|ts
#id
10|ts
10|ts
map
{ls}→#id ••• {ls}→#id
RAM


1 000 000
{ls}
{ls}
{ls}
SSD
wal
{ls}→#id
#id|10|ts
{ls}→#id
#id|10|ts
#id|10|ts
#id|10|ts
#id|10|ts
#id
10|ts
10|ts
map
{ls}→#id ••• {ls}→#id
RAM



SSD

wal

{ls}→#id #id|10|ts •••

{ls}→#id #id|10|ts #id|10|ts

#id|10|ts #id|10|ts •••

#id|10|ts

#id 10|ts ••• 10|ts

#id 10|ts ••• 10|ts

•••

#id 10|ts ••• 10|ts

map {ls}→#id ••• {ls}→#id

#id 10|ts ••• 10|ts

#id 10|ts ••• 10|ts

•••

#id 10|ts ••• 10|ts

map {ls}→#id ••• {ls}→#id

RAM

#id 10|ts ••• 10|ts

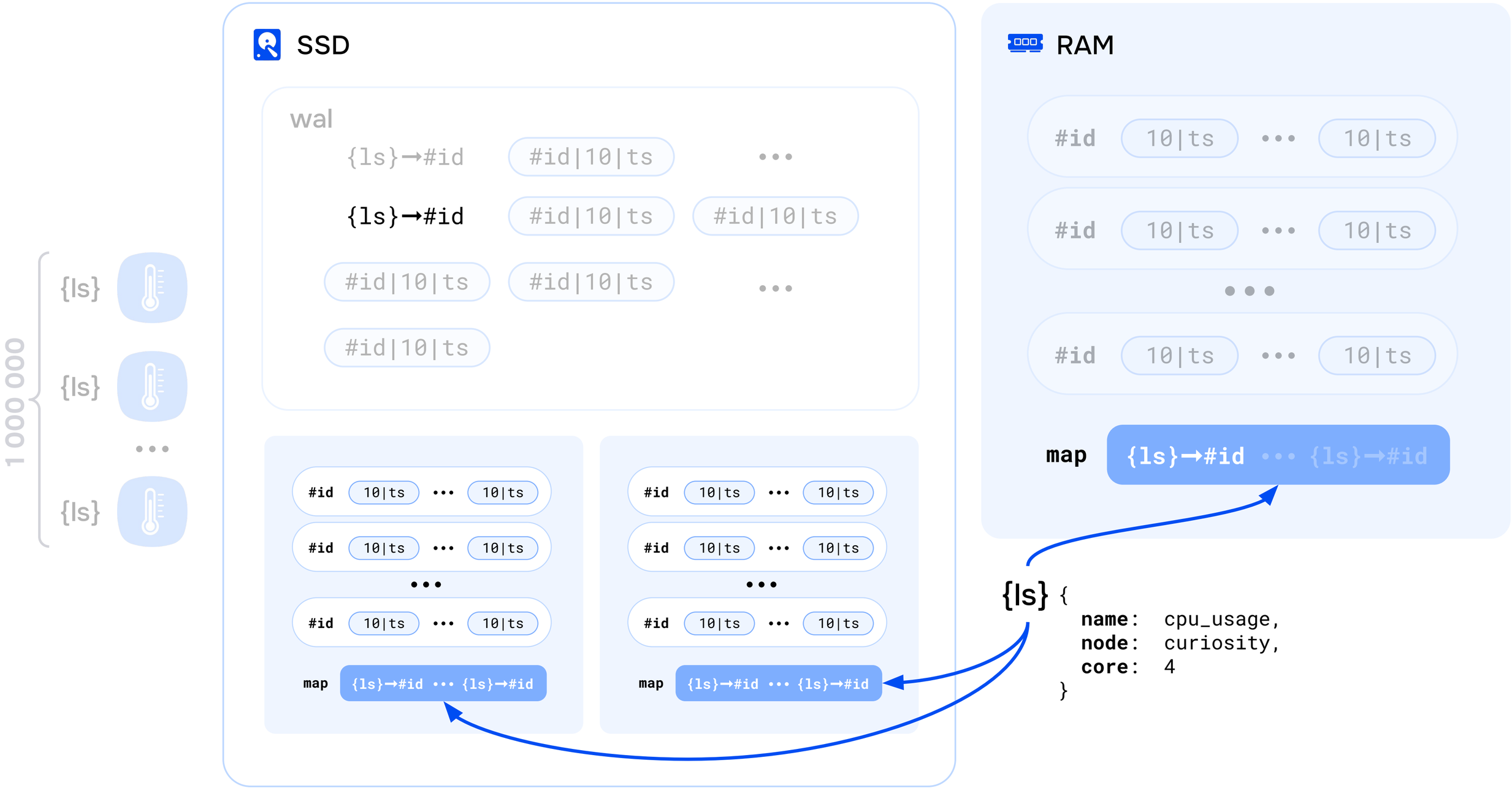
#id 10|ts ••• 10|ts

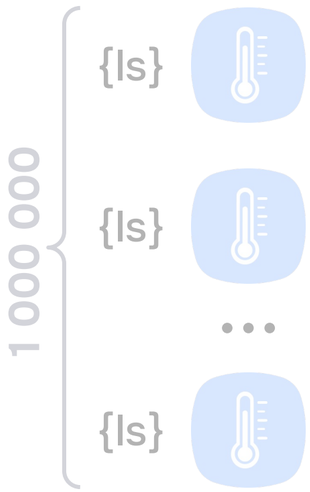
• • •

#id 10|ts ••• 10|ts

map {ls}→#id ••• {ls}→#id

{ls} {
  name: cpu_usage,
  node: curiosity,
  core: 4
}


SSD
wal
{ls}→#id
#id|10|ts
{ls}→#id
#id|10|ts
#id|10|ts
#id|10|ts
#id|10|ts
#id|10|ts
#id
10|ts
10|ts
map
{ls}→#id ••• {ls}→#id
RAM
#id
10|ts
10|ts
map
{ls}→#id ••• {ls}→#id
1 000 000
{ls}
{ls}
{ls}
{ls} {
name: cpu_usage,
node: curiosity,
core: 4
}



SSD

wal

{ls}→#id #id|10|ts •••

{ls}→#id #id|10|ts #id|10|ts

#id|10|ts #id|10|ts •••

#id|10|ts

#id 10|ts ••• 10|ts

#id 10|ts ••• 10|ts

•••

#id 10|ts ••• 10|ts

map {ls}→#id ••• {ls}→#id

#id 10|ts ••• 10|ts

#id 10|ts ••• 10|ts

•••

#id 10|ts ••• 10|ts

map {ls}→#id ••• {ls}→#id

RAM

#id 10|ts ••• 10|ts

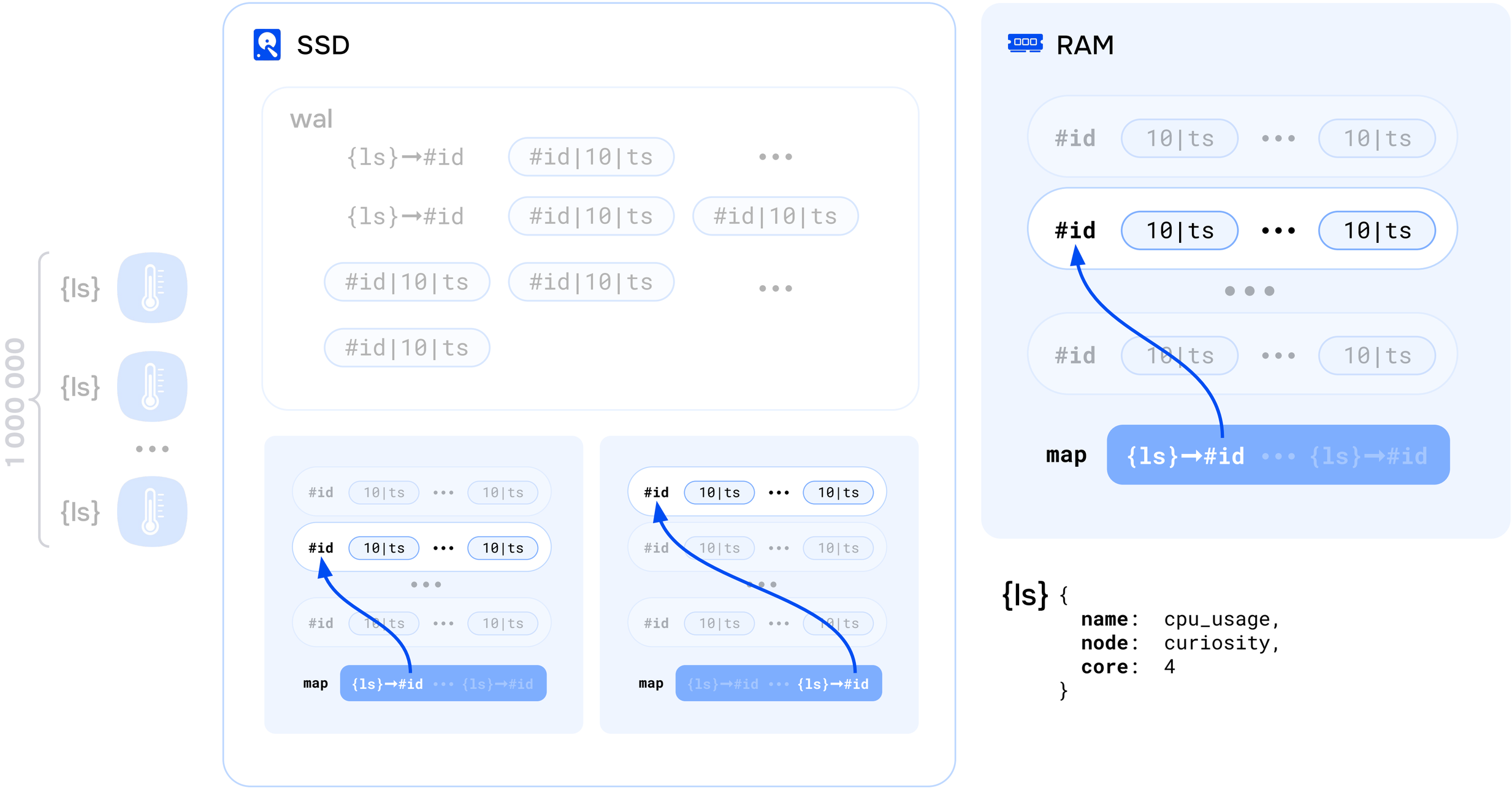
#id 10|ts ••• 10|ts

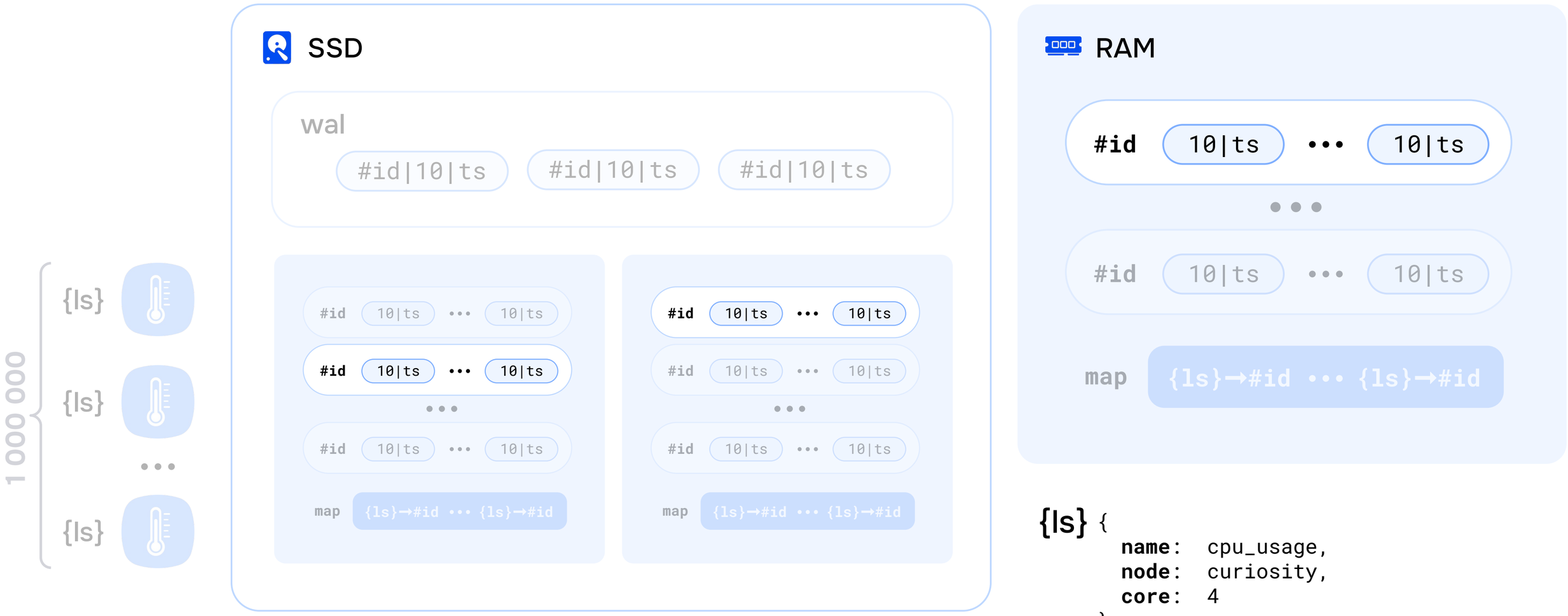
•••

#id 10|ts ••• 10|ts

map {ls}→#id ••• {ls}→#id

```
{ls} {
    name:  cpu_usage,
    node:  curiosity,
    core:  4
}
```

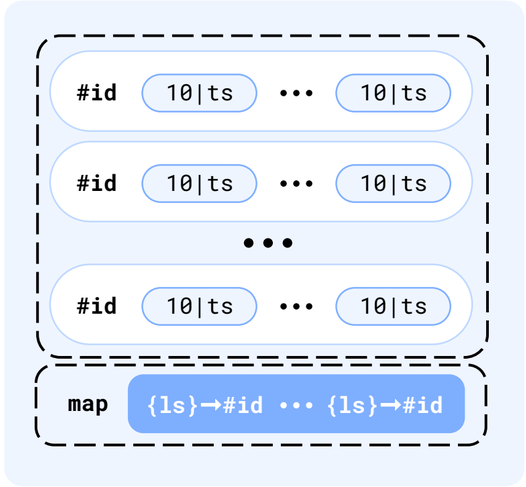

1 000 000
{ls}
{ls}
{ls}
SSD
wal
{ls}→#id
#id|10|ts
{ls}→#id
#id|10|ts
#id|10|ts
#id|10|ts
#id|10|ts
#id|10|ts
#id
10|ts
10|ts
map
{ls}→#id
{ls}→#id
RAM
#id
10|ts
10|ts
#id
10|ts
10|ts
#id
10|ts
10|ts
map
{ls}→#id
{ls}→#id
{ls} {
name: cpu_usage,
node: curiosity,
core: 4
}

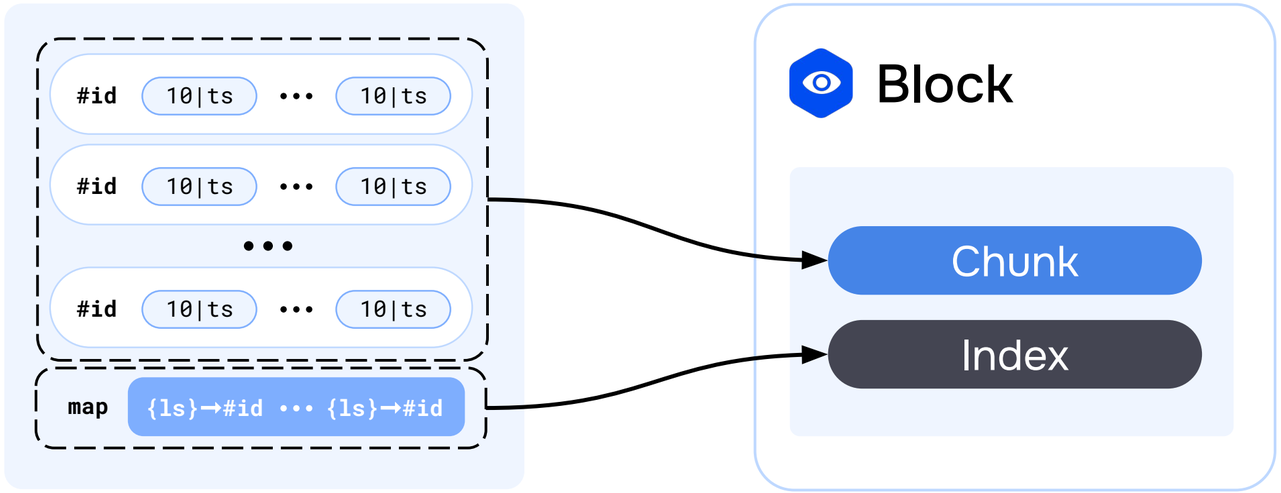


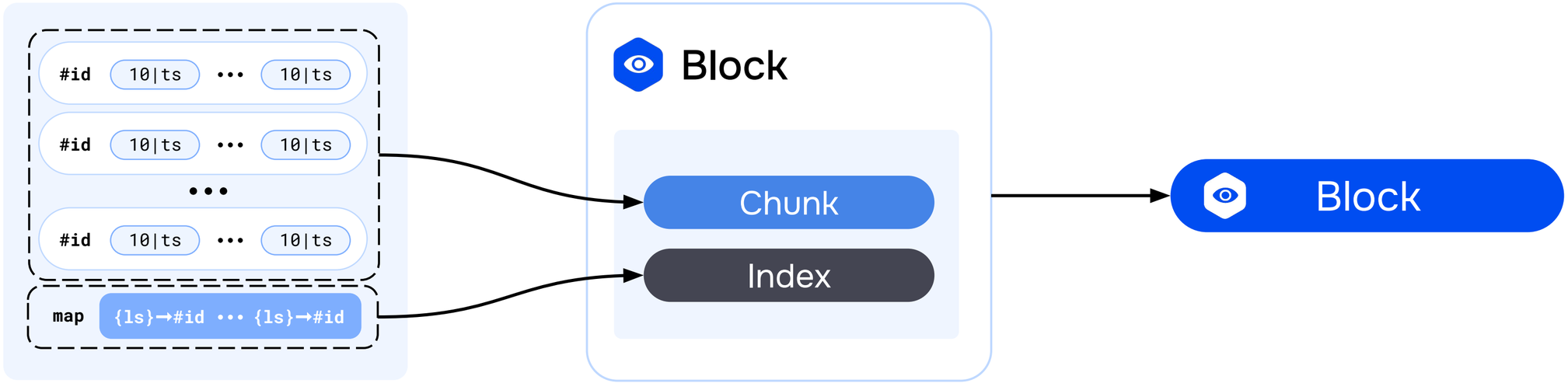
{ls} {
  name: cpu_usage,
  node: curiosity,
  core: 4
}


SSD
wal
#id|10|ts
#id|10|ts
#id|10|ts
#id 10|ts ••• 10|ts
#id 10|ts ••• 10|ts
#id 10|ts ••• 10|ts
map {ls}→#id ••• {ls}→#id
#id 10|ts ••• 10|ts
#id 10|ts ••• 10|ts
#id 10|ts ••• 10|ts
map {ls}→#id ••• {ls}→#id
1 000 000
{ls}
{ls}
{ls}
RAM
#id 10|ts ••• 10|ts
#id 10|ts ••• 10|ts
map {ls}→#id ••• {ls}→#id
{ls} {
name: cpu_usage,
node: curiosity,
core: 4
}
10|ts
10|ts
10|ts
10|ts
10|ts
10|ts
10|ts


SSD
wal
#id|10|ts
#id|10|ts
#id|10|ts
#id 10|ts ••• 10|ts
#id 10|ts ••• 10|ts
#id 10|ts ••• 10|ts
map {ls}→#id ••• {ls}→#id
#id 10|ts ••• 10|ts
#id 10|ts ••• 10|ts
#id 10|ts ••• 10|ts
map {ls}→#id ••• {ls}→#id
RAM
#id 10|ts ••• 10|ts
#id 10|ts ••• 10|ts
map {ls}→#id ••• {ls}→#id
1 000 000
{ls}
{ls}
{ls}
{ls} {
name: cpu_usage,
node: curiosity,
core: 4
}
10|ts 10|ts 10|ts 10|ts 10|ts 10|ts ••• 10|ts

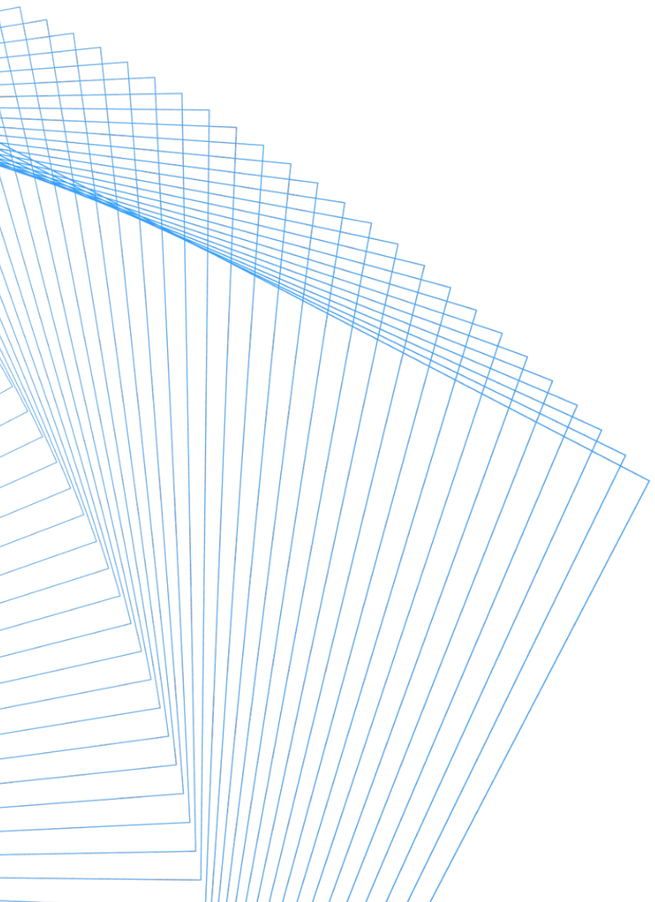

#id
10|ts
•••
10|ts
#id
10|ts
•••
10|ts
•••
#id
10|ts
•••
10|ts
map
{ls}→#id ••• {ls}→#id


#id
10|ts
•••
10|ts
#id
10|ts
•••
10|ts
•••
#id
10|ts
•••
10|ts
map
{ls}→#id ••• {ls}→#id
Block
Chunk
Index

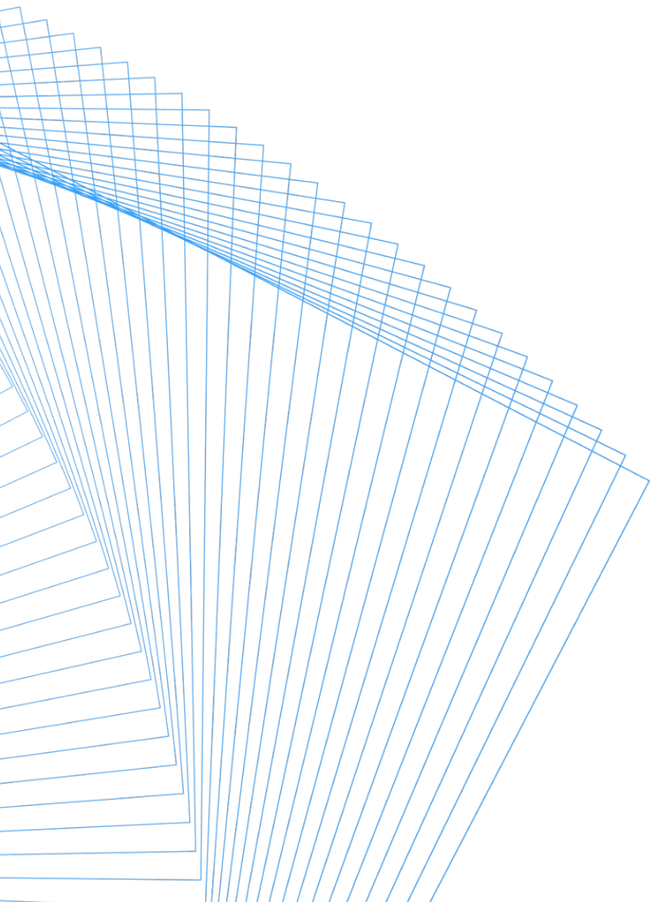

#id
10|ts
•••
10|ts
#id
10|ts
•••
10|ts
•••
#id
10|ts
•••
10|ts
map
{ls}→#id ••• {ls}→#id
Block
Chunk
Index
Block

Важно

01 Блок не меняется после записи на диск



Важно

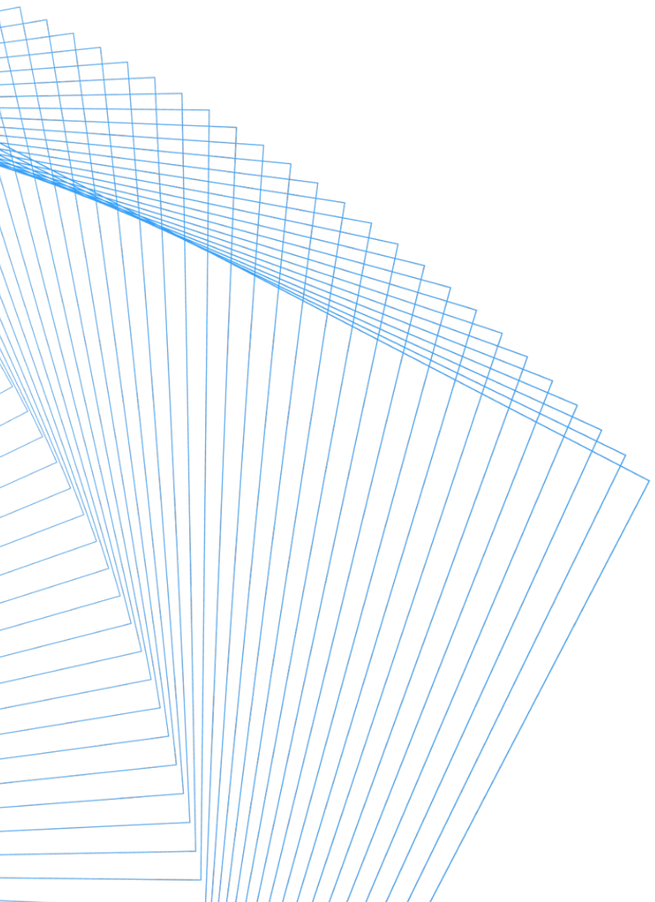


01 Блок не меняется после записи на диск

02 Блок состоит из двух частей:

- Index — содержит данные о LabelSet
- Chunk — содержит значения метрик

Важно



01 Блок не меняется после записи на диск

02 Блок состоит из двух частей:

- Index — содержит данные о LabelSet
- Chunk — содержит значения метрик

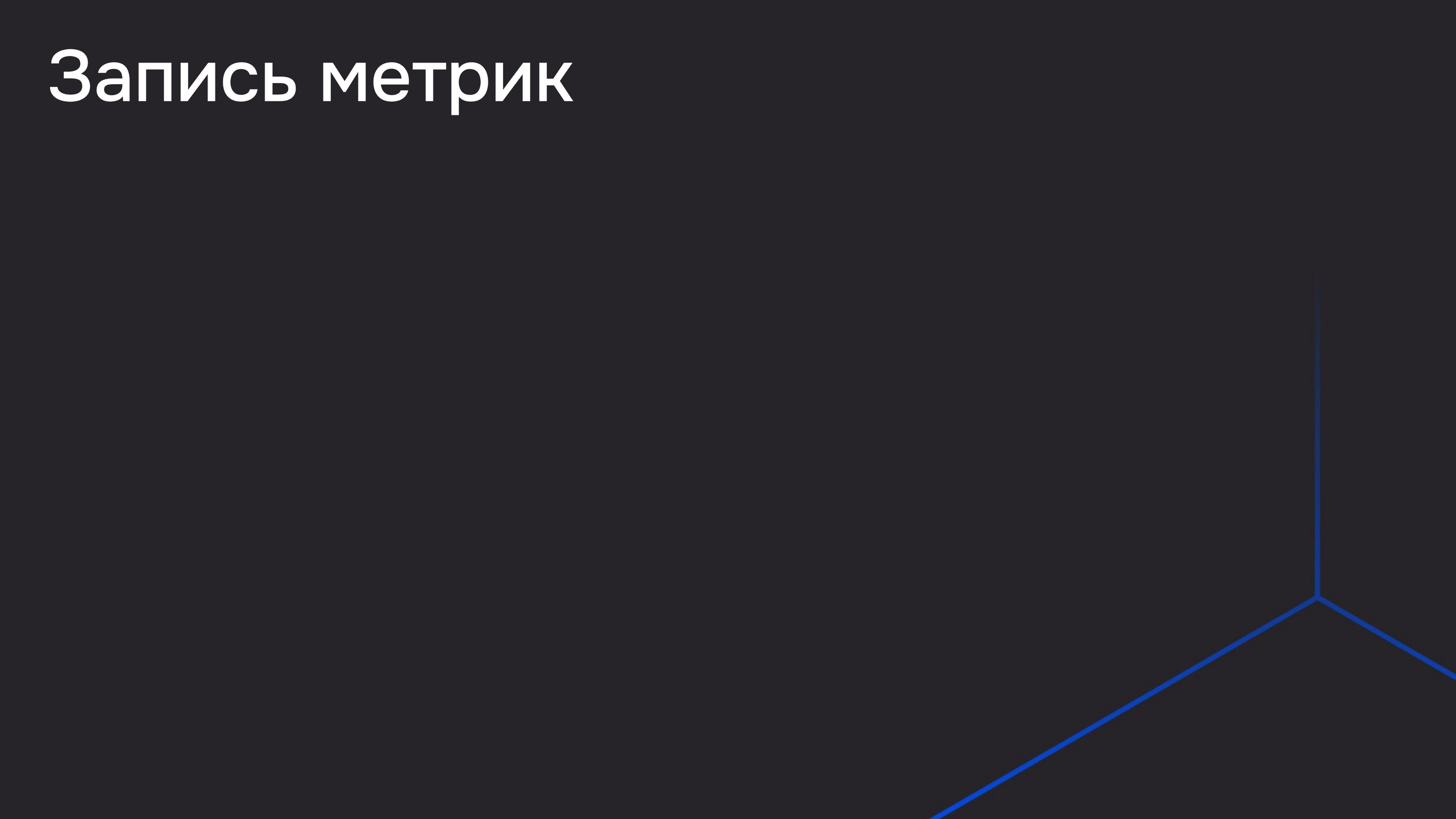
03 Index — примерно 10 % от размера всего блока

Мониторинг

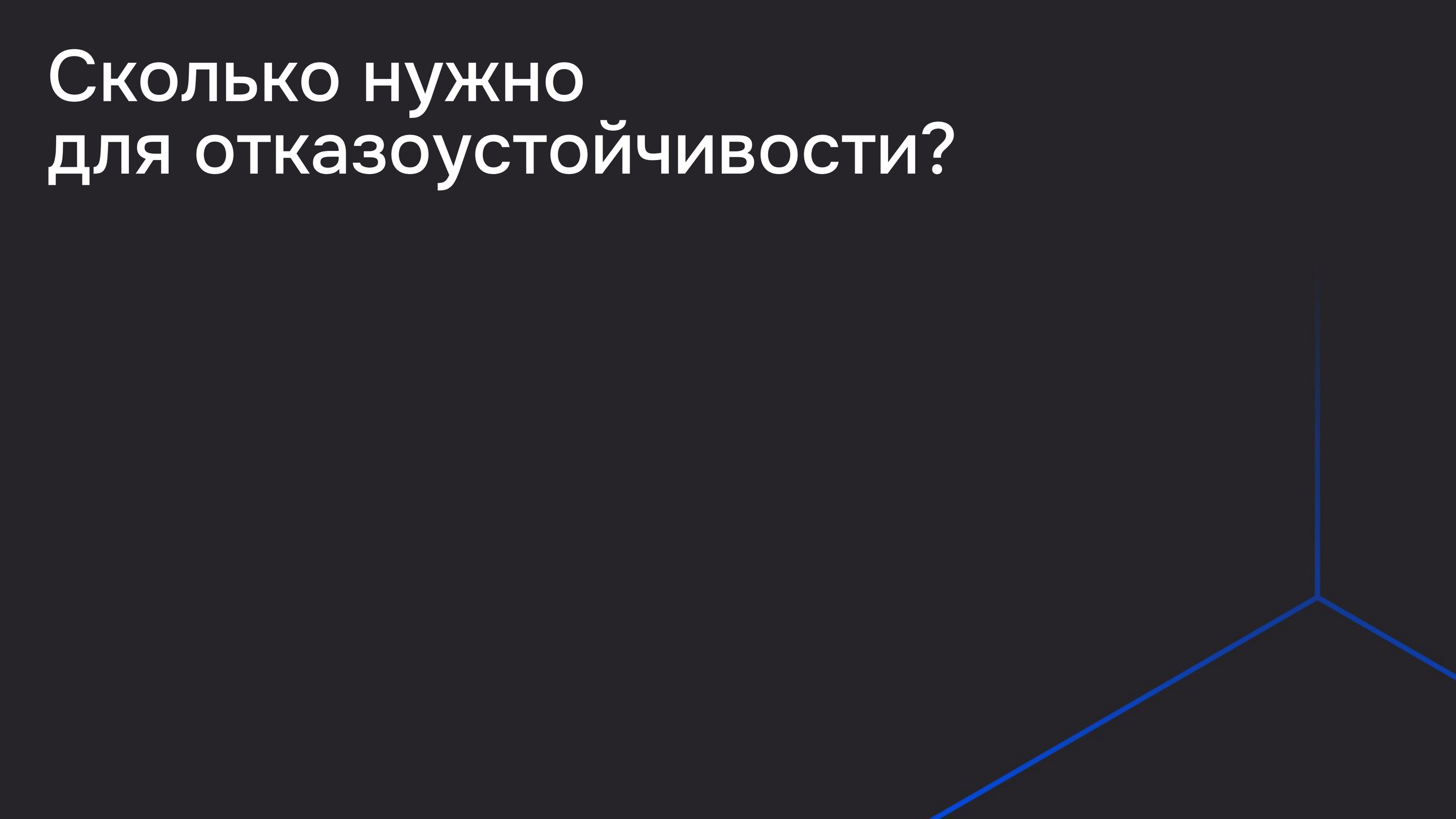
Статья «Мимо тёщиного дома я без метрик не хожу»

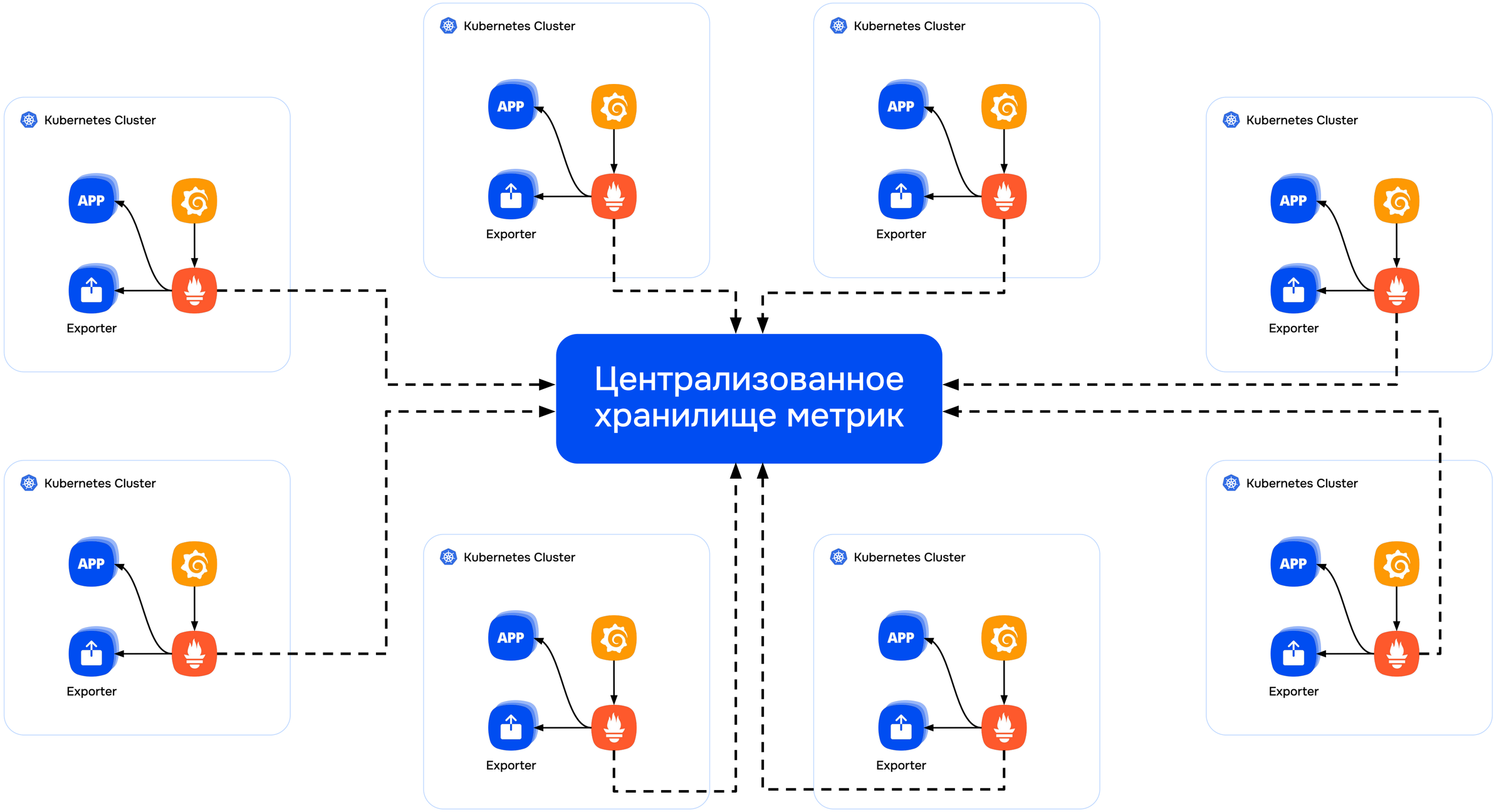
Подробнее

Запись метрик

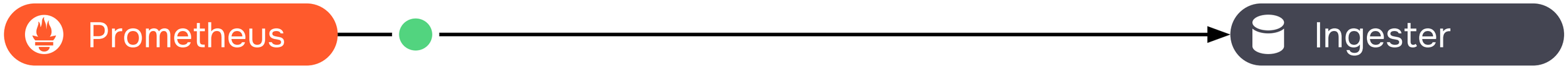


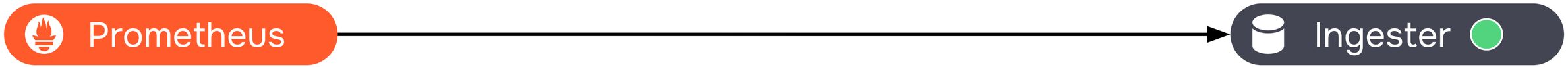
Сколько нужно для отказоустойчивости?

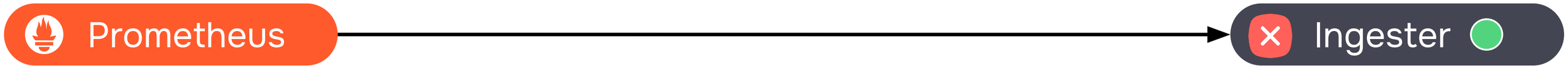



Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter
Централизованное хранилище метрик
Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter
Kubernetes Cluster
APP
Exporter


Prometheus


Prometheus
Ingester

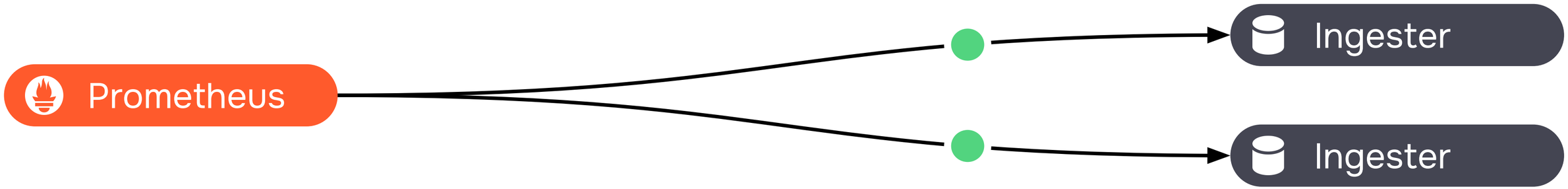

Prometheus
Ingester

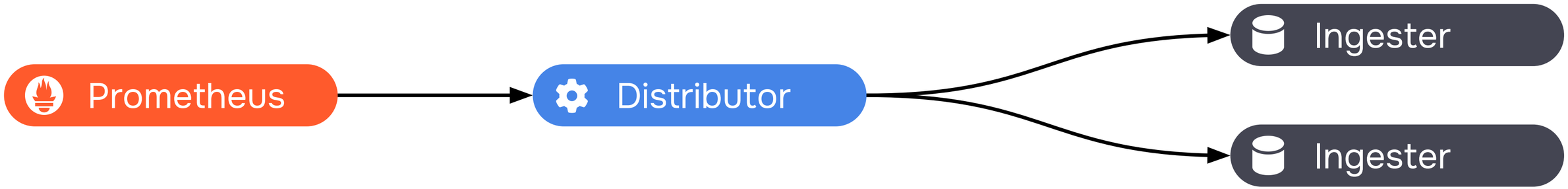

Prometheus
Ingester

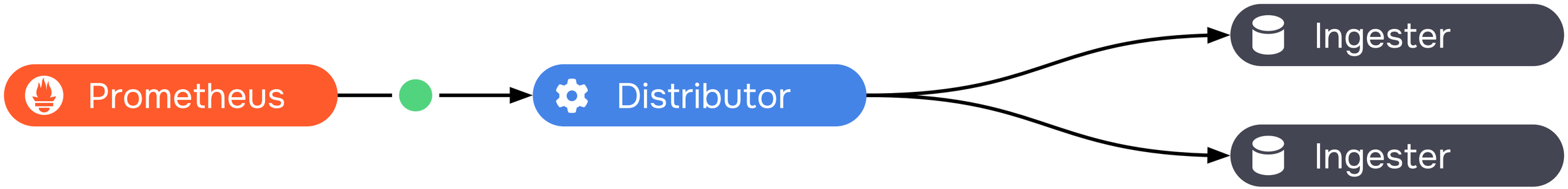

Prometheus

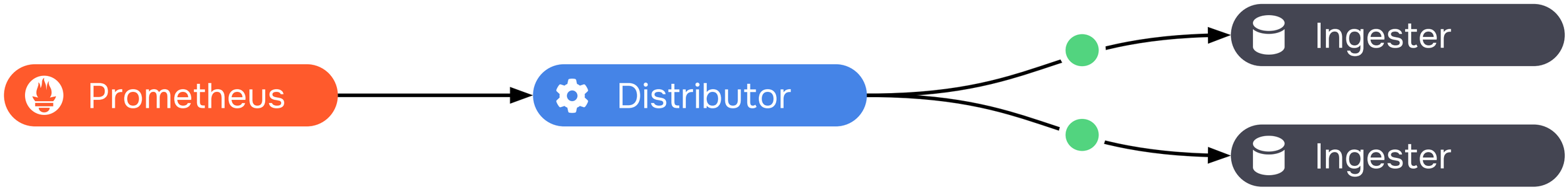

Ingester

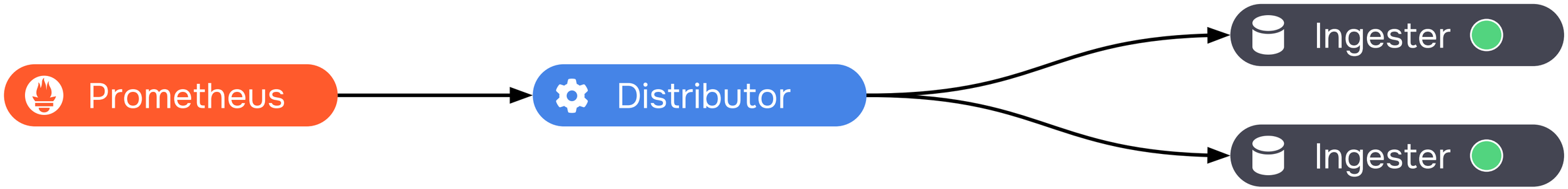

Ingester

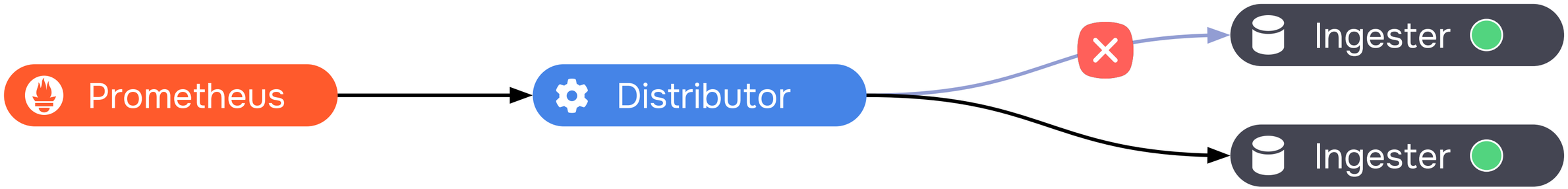

Prometheus
Ingester
Ingester

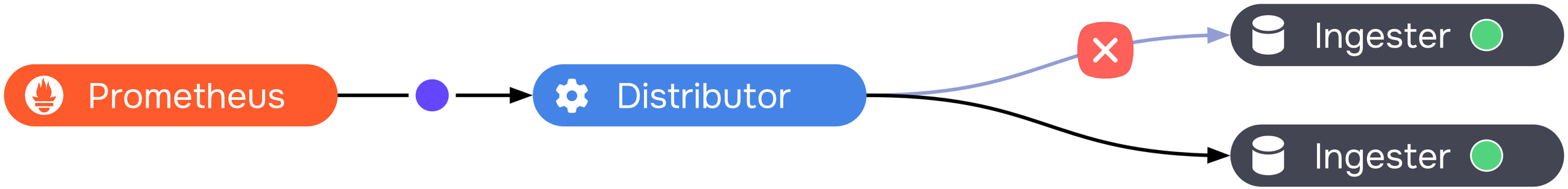

Prometheus
Distributor
Ingester
Ingester

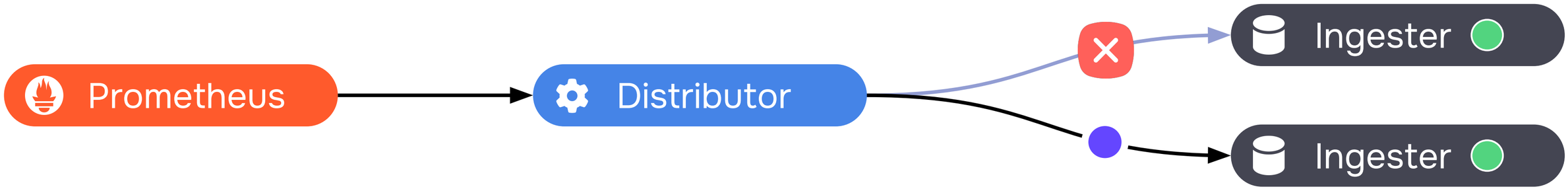

Prometheus
Distributor
Ingester
Ingester

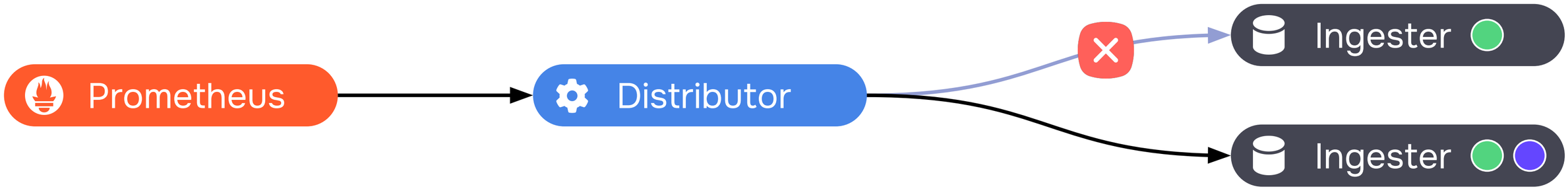

Prometheus
Distributor
Ingester
Ingester

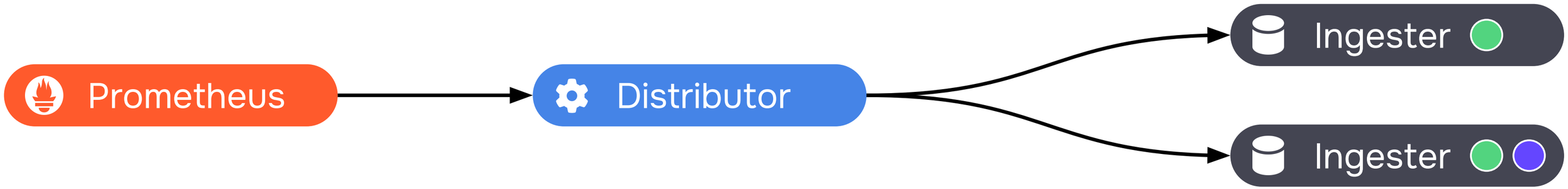

Prometheus
Distributor
Ingester
Ingester

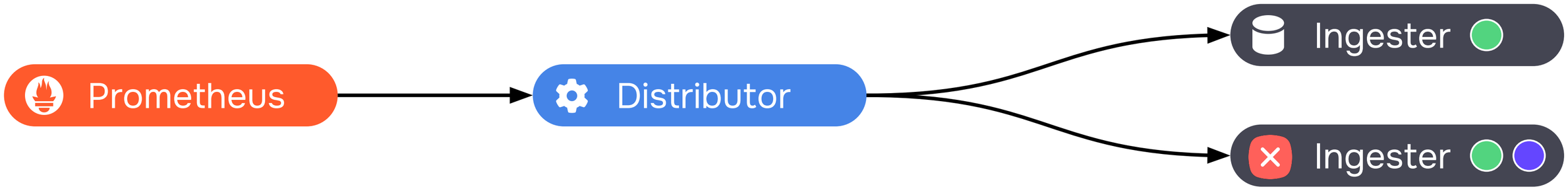

Prometheus
Distributor
Ingester
Ingester

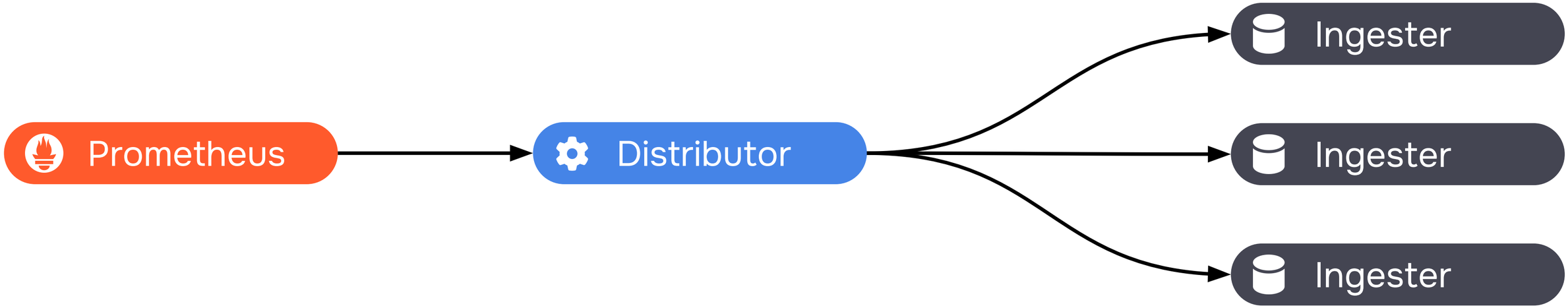

Prometheus
Distributor
Ingester
Ingester

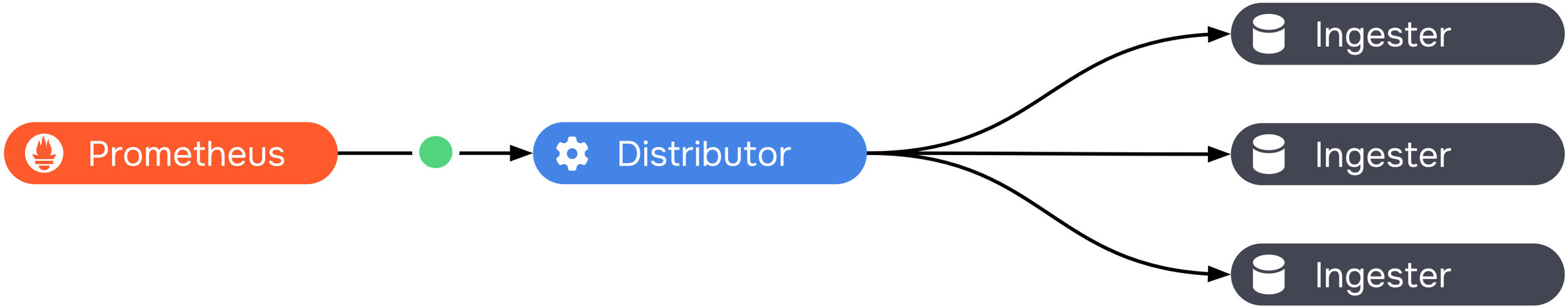

Prometheus
Distributor
Ingester
Ingester

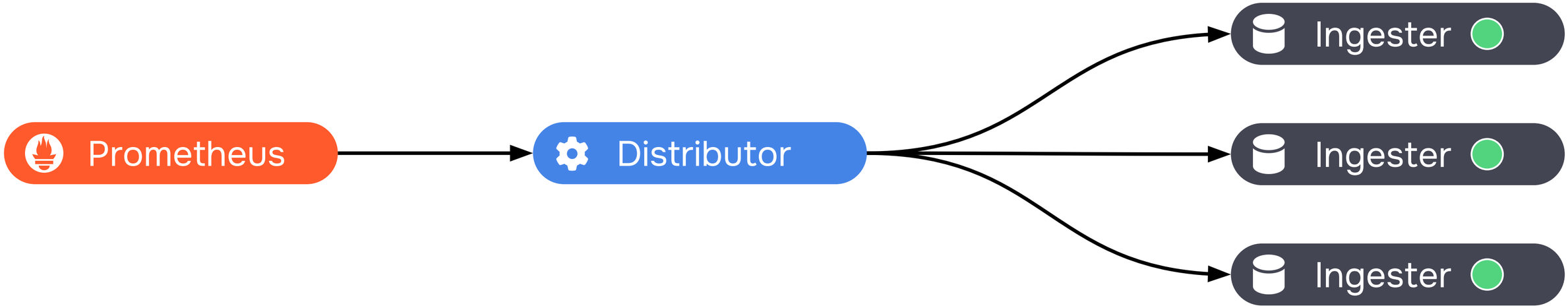

Prometheus
Distributor
Ingester
Ingester

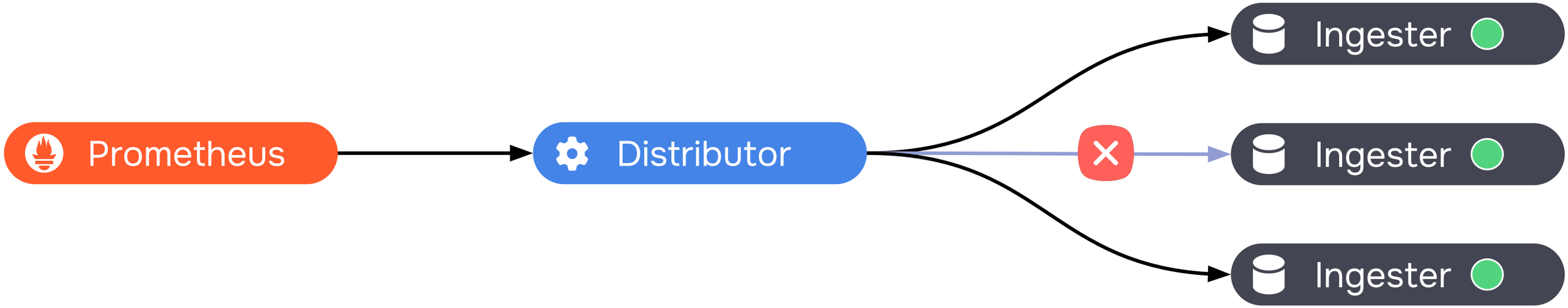

Prometheus
Distributor
Ingester
Ingester

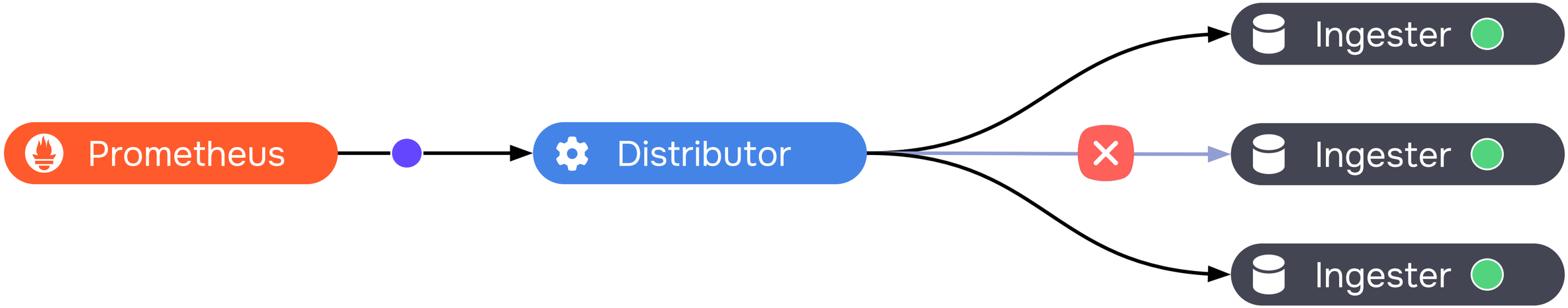

Prometheus
Distributor
Ingester
Ingester

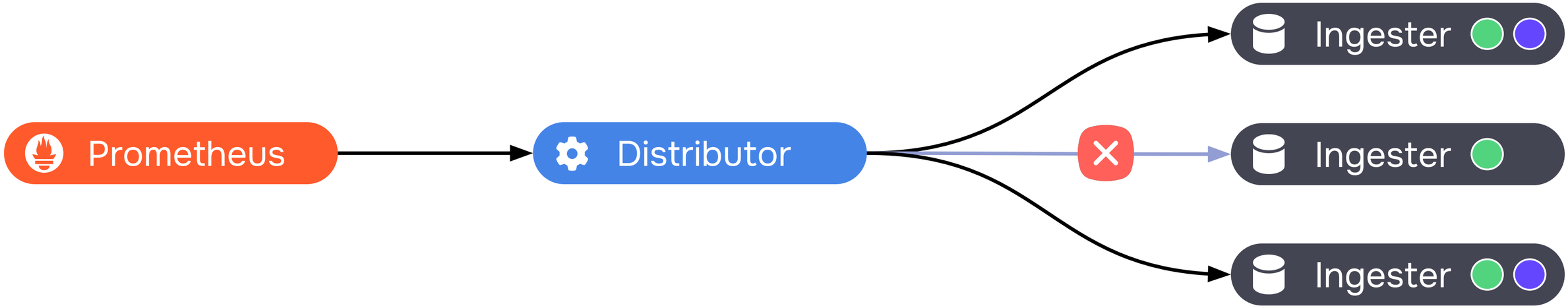

Prometheus
Distributor
Ingester
Ingester
Ingester

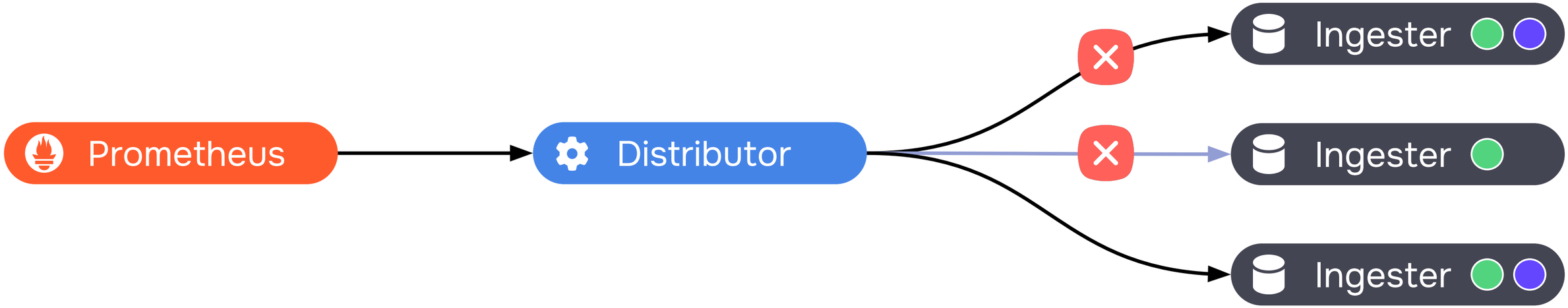

Prometheus
Distributor
Ingester
Ingester
Ingester

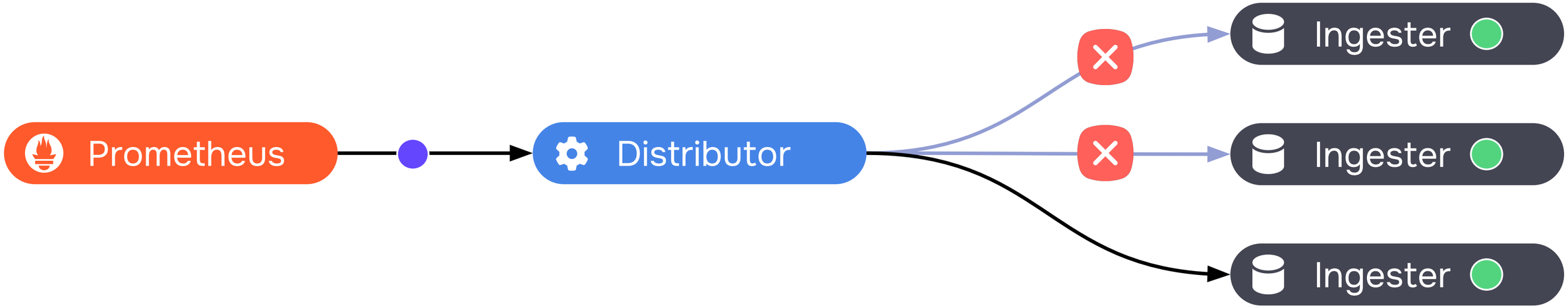

Prometheus
Distributor
Ingester
Ingester
Ingester

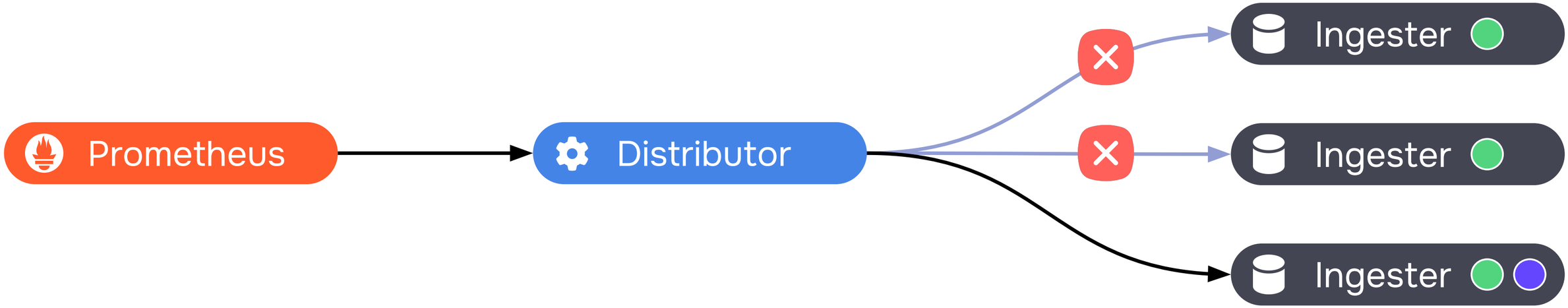

Prometheus
Distributor
Ingester
Ingester
Ingester

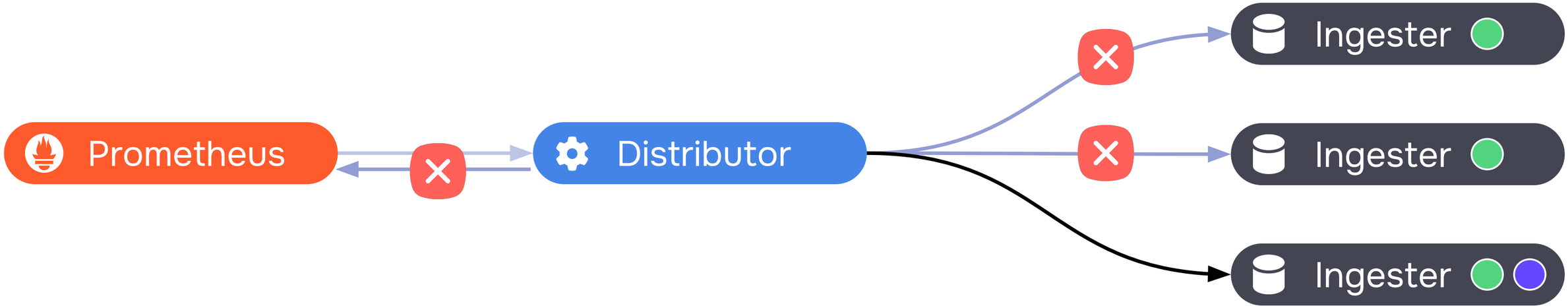

Prometheus
Distributor
Ingester
Ingester
Ingester

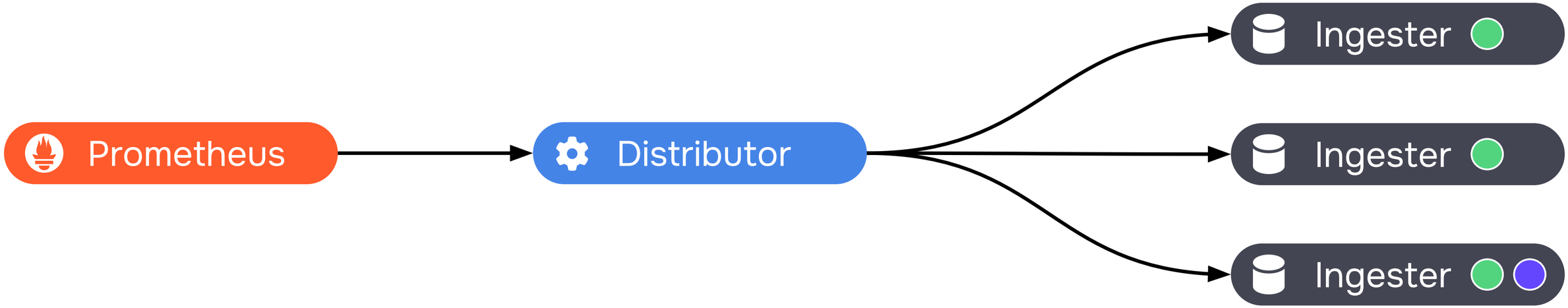

Prometheus
Distributor
Ingester
Ingester
Ingester

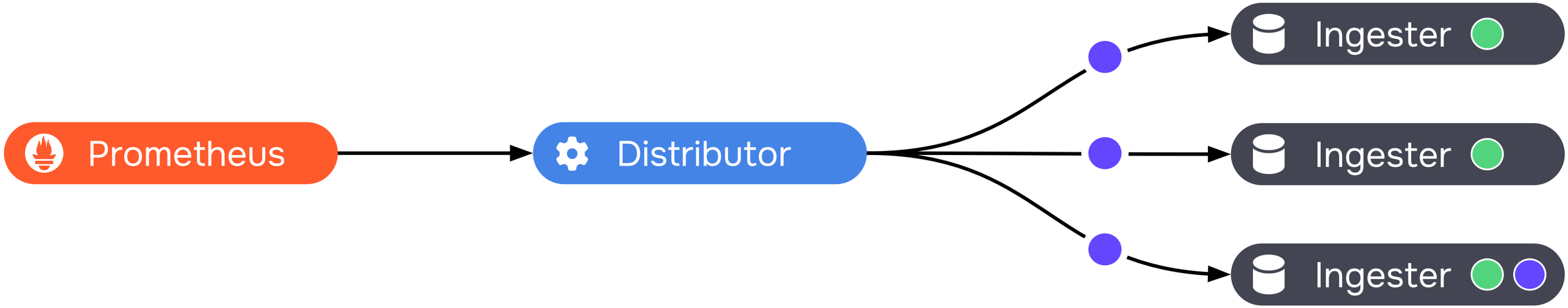

Prometheus
Distributor
Ingester
Ingester
Ingester

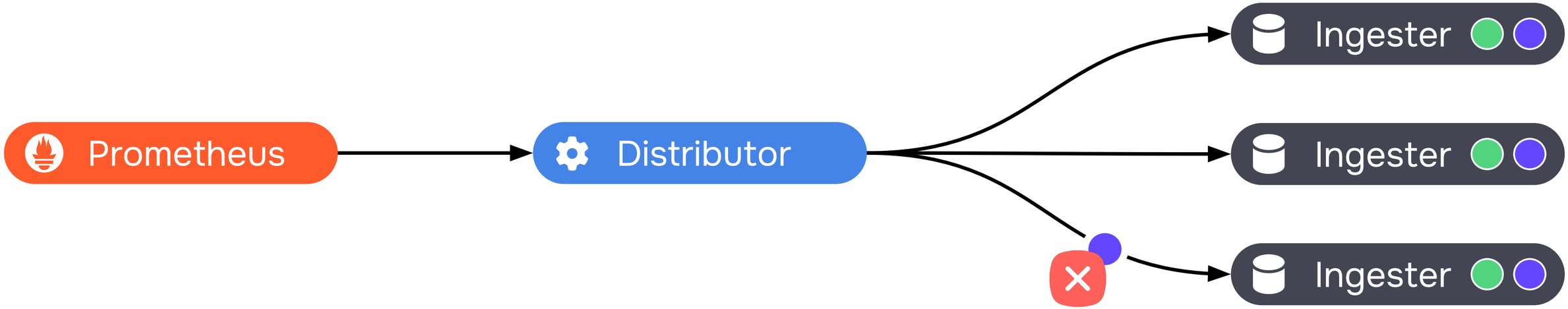

Prometheus
Distributor
Ingester
Ingester
Ingester

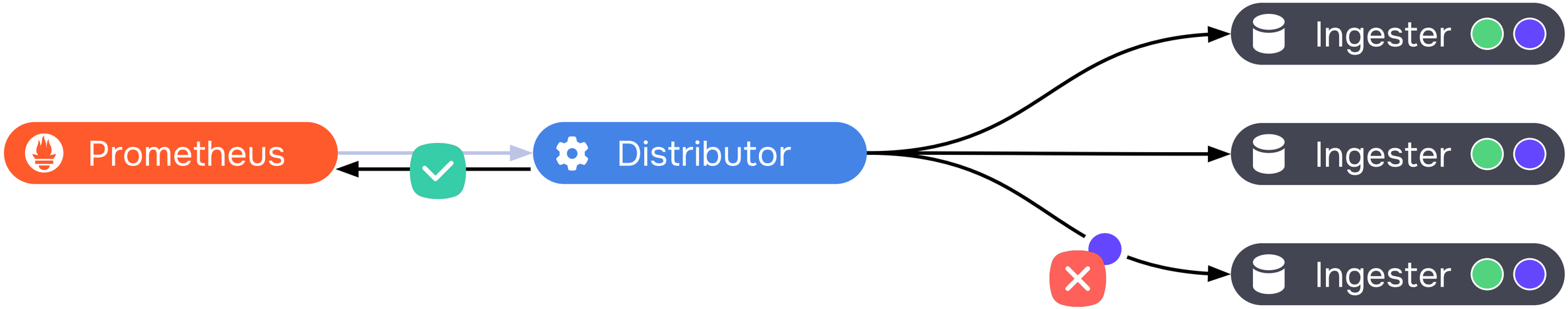

Prometheus
Distributor
Ingester
Ingester
Ingester


Prometheus
Distributor
Ingester
Ingester
Ingester


Prometheus
Distributor
Ingester
Ingester
Ingester

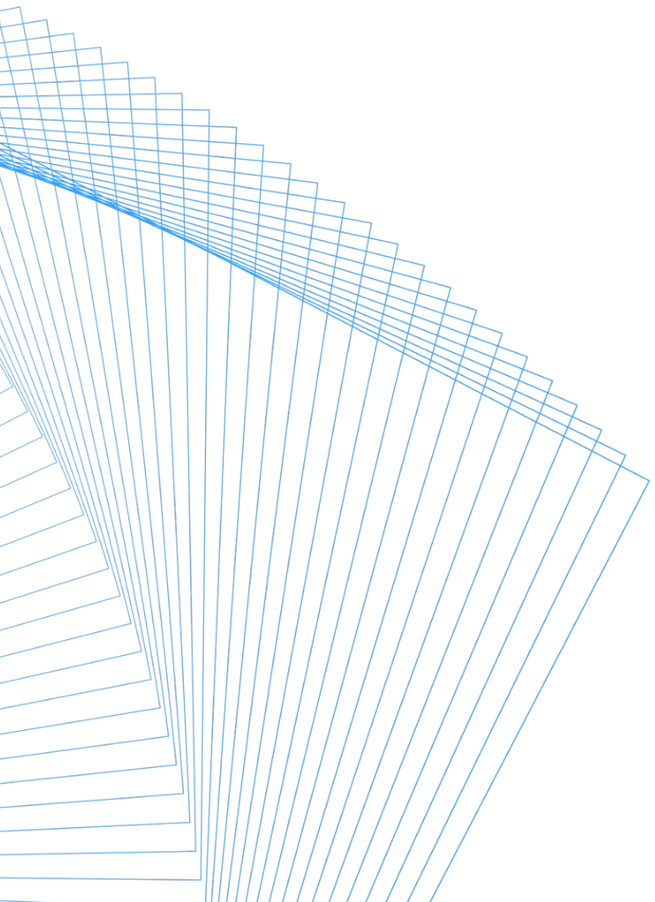

Prometheus
Distributor
Ingester
Ingester
Ingester


Prometheus
Distributor
Ingester
Ingester
Ingester


Prometheus
Distributor
Ingester
Ingester
Ingester

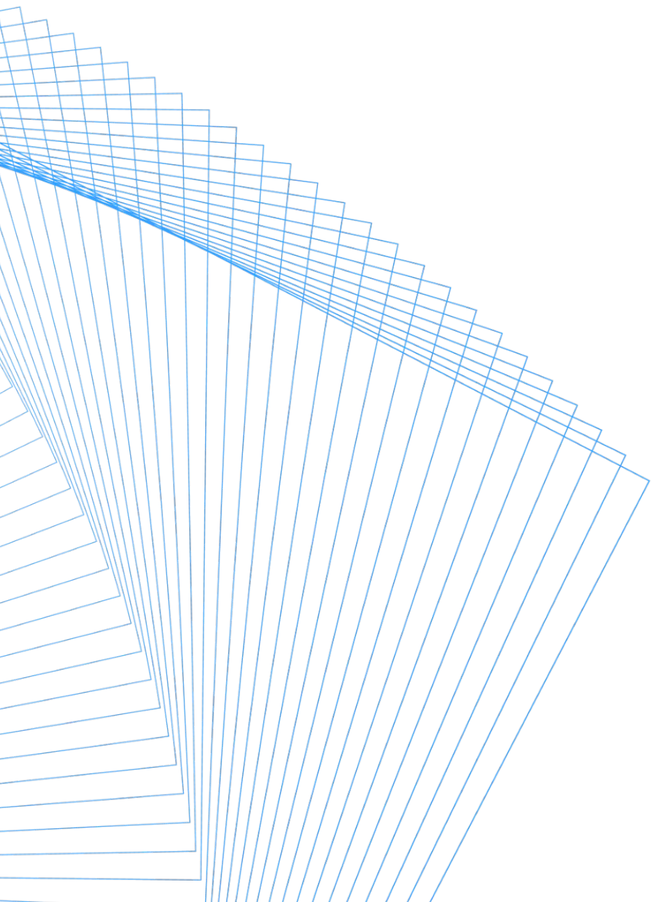
Централизованная система мониторинга

- Отказоустойчивость на запись: WC = RF/2 + 1

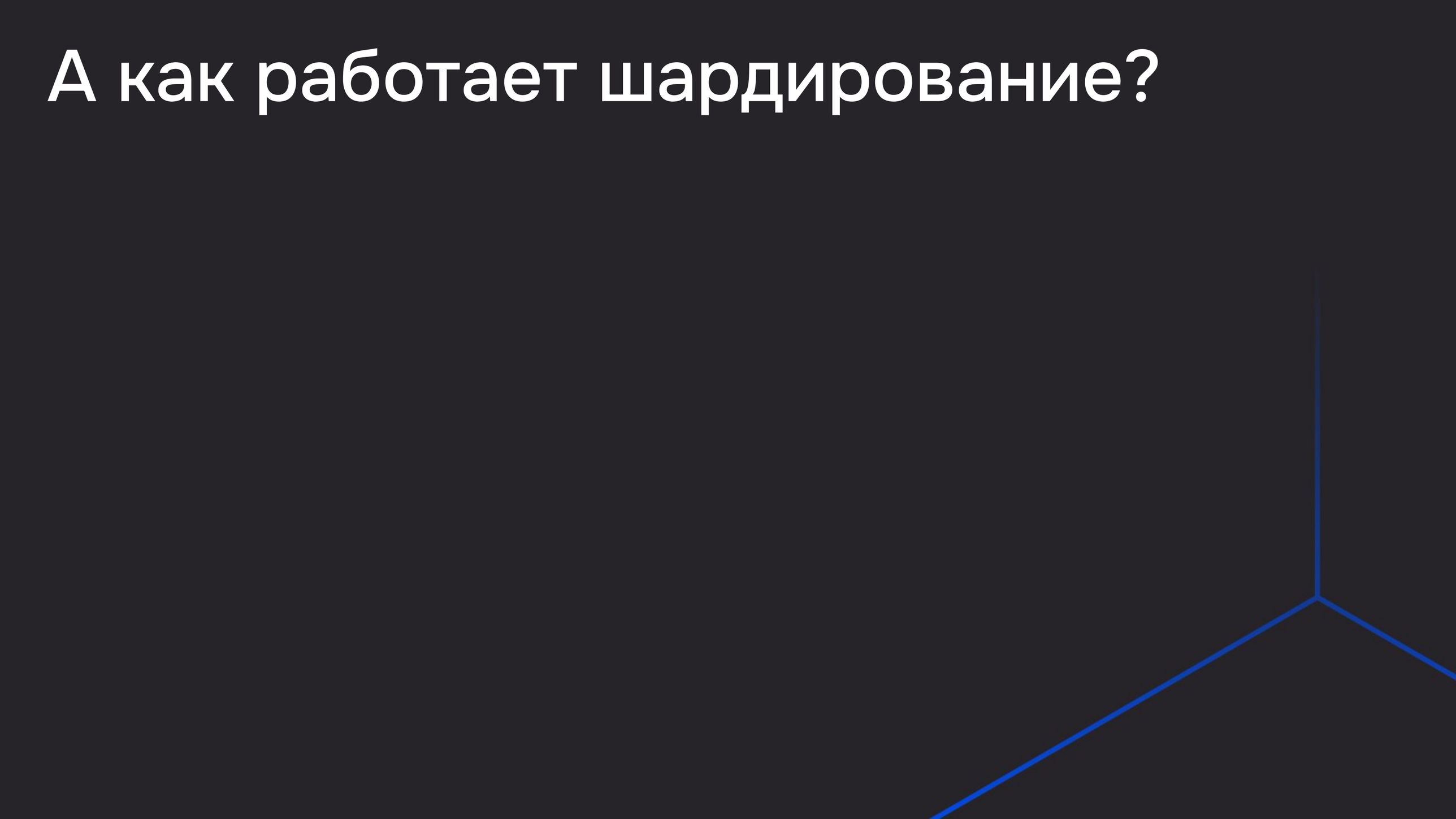


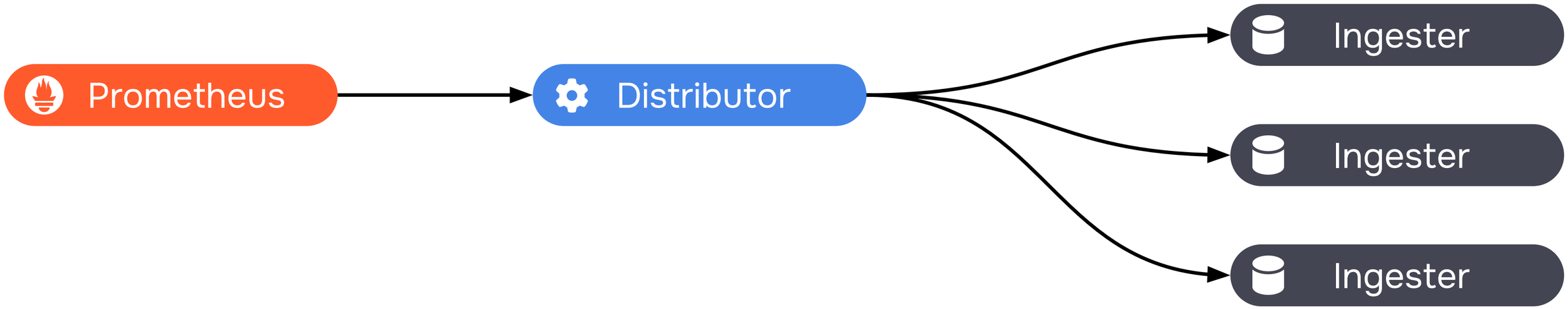
Централизованная система мониторинга

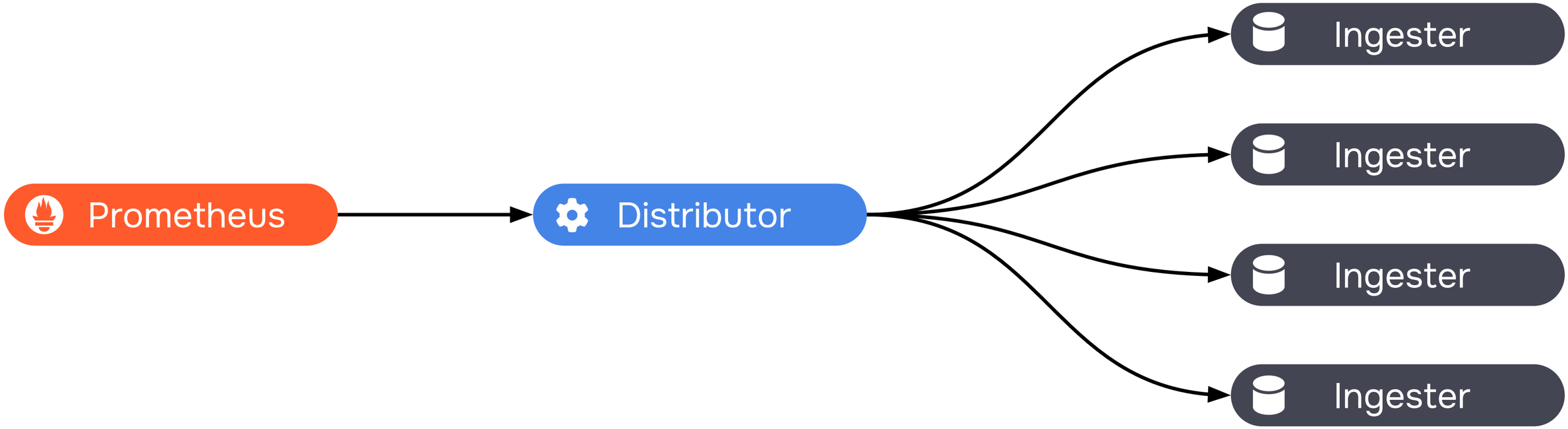
- Отказоустойчивость на запись: WC = RF/2 + 1
- Отказоустойчивость на чтение: RC = RF - WC + 1

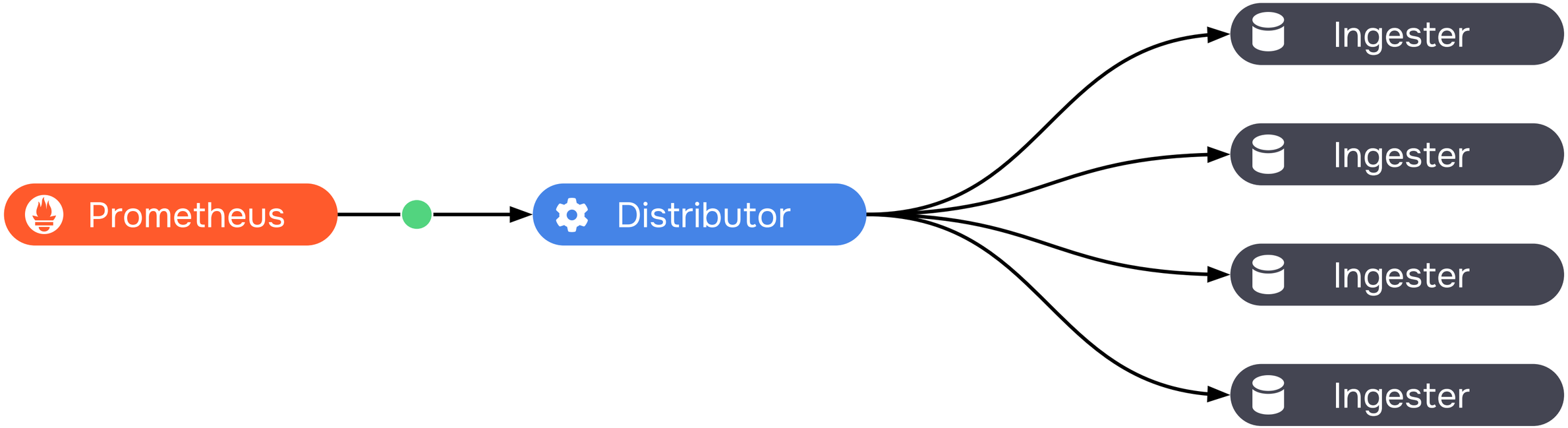


А как работает шардирование?



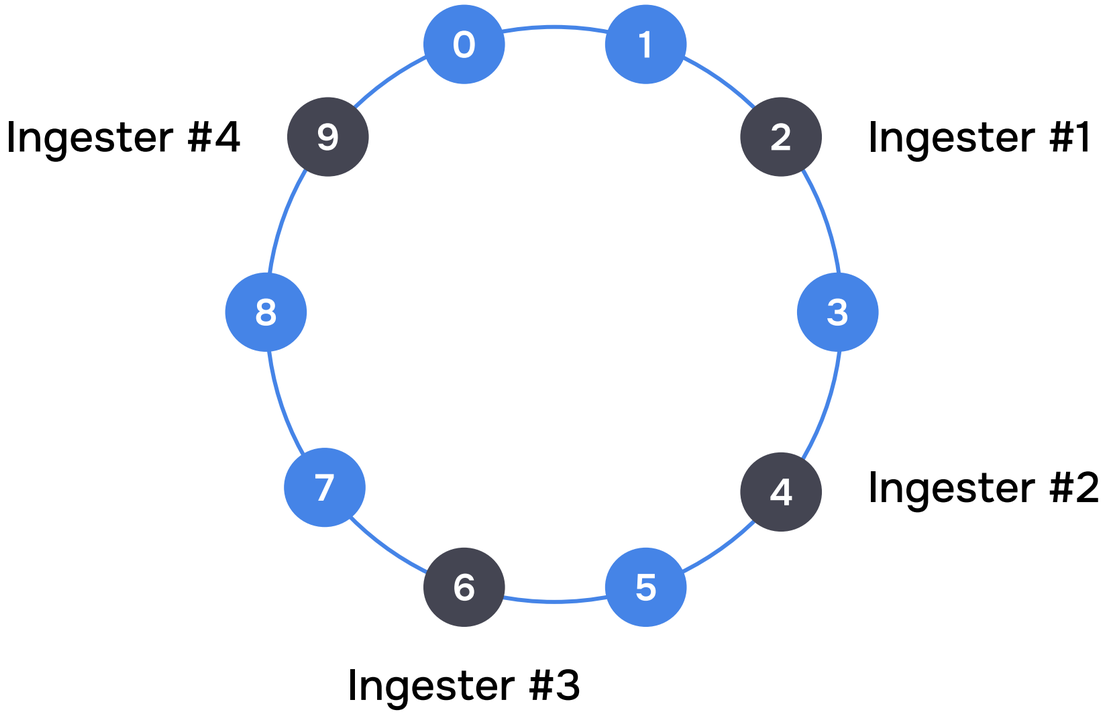

Prometheus
Distributor
Ingester
Ingester
Ingester


Prometheus
Distributor
Ingester
Ingester
Ingester
Ingester


Prometheus
Distributor
Ingester
Ingester
Ingester
Ingester

Хеш-ринг

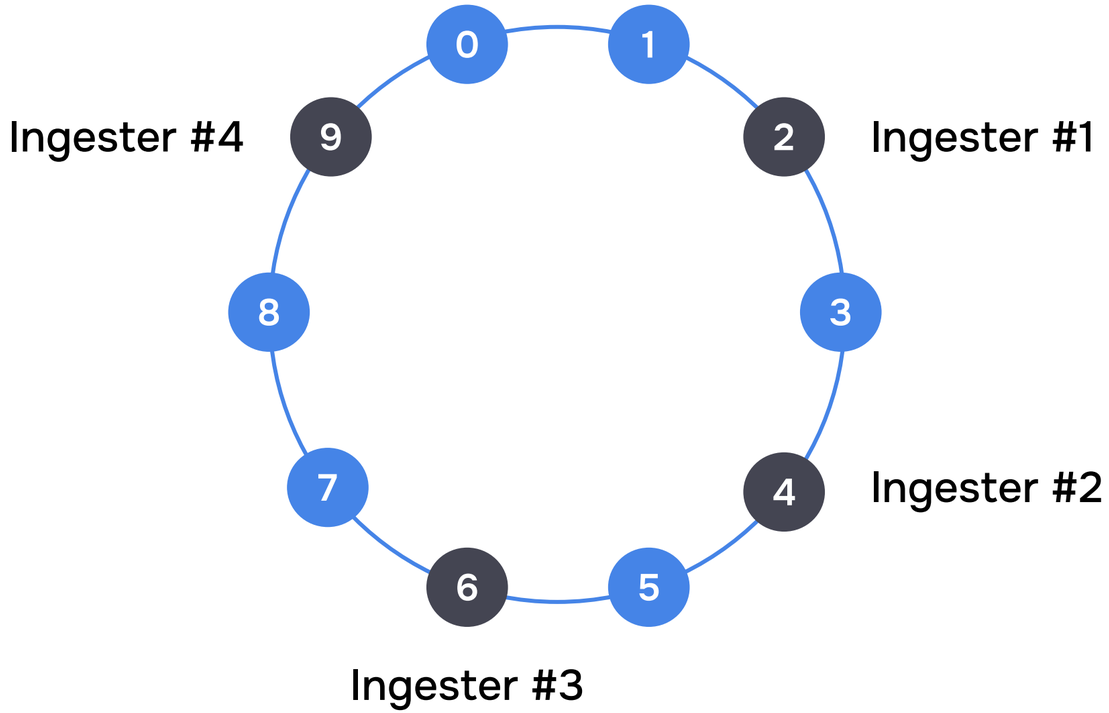
Хеш-ринг



Хеш-ринг

Series

```
{
   __name__="cpu_seconds_total,
   instance: "1.1.1.1"
}
```

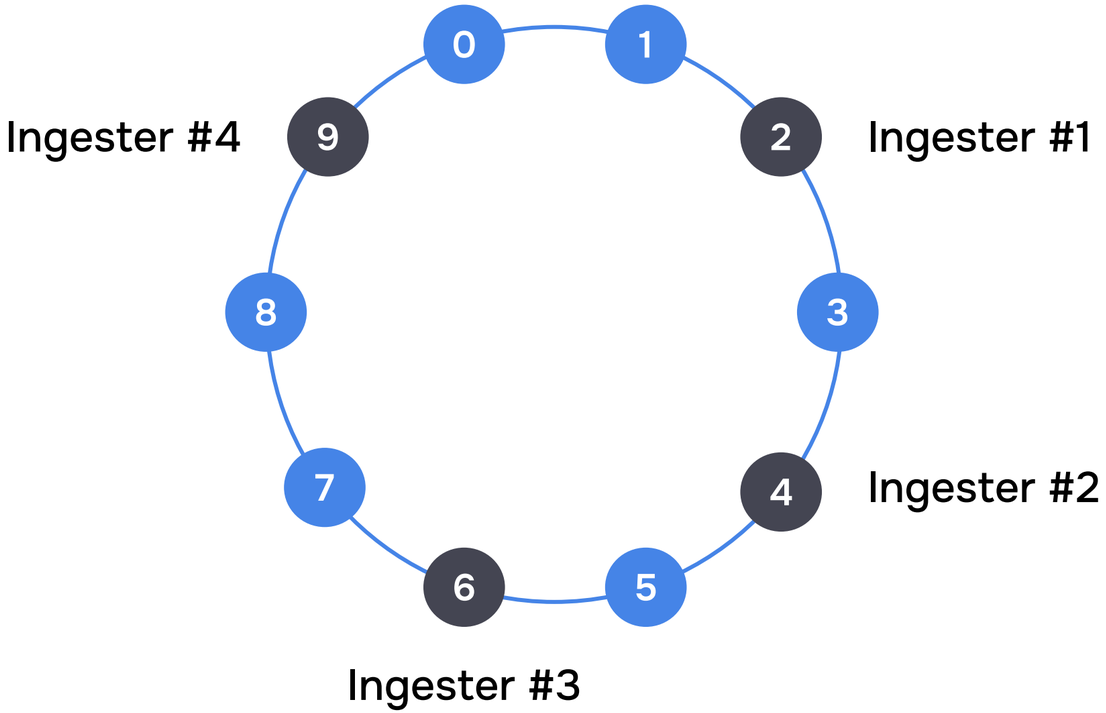


Хеш-ринг

Series

```
{
   __name__="cpu_seconds_total,
   instance: "1.1.1.1"
}
```

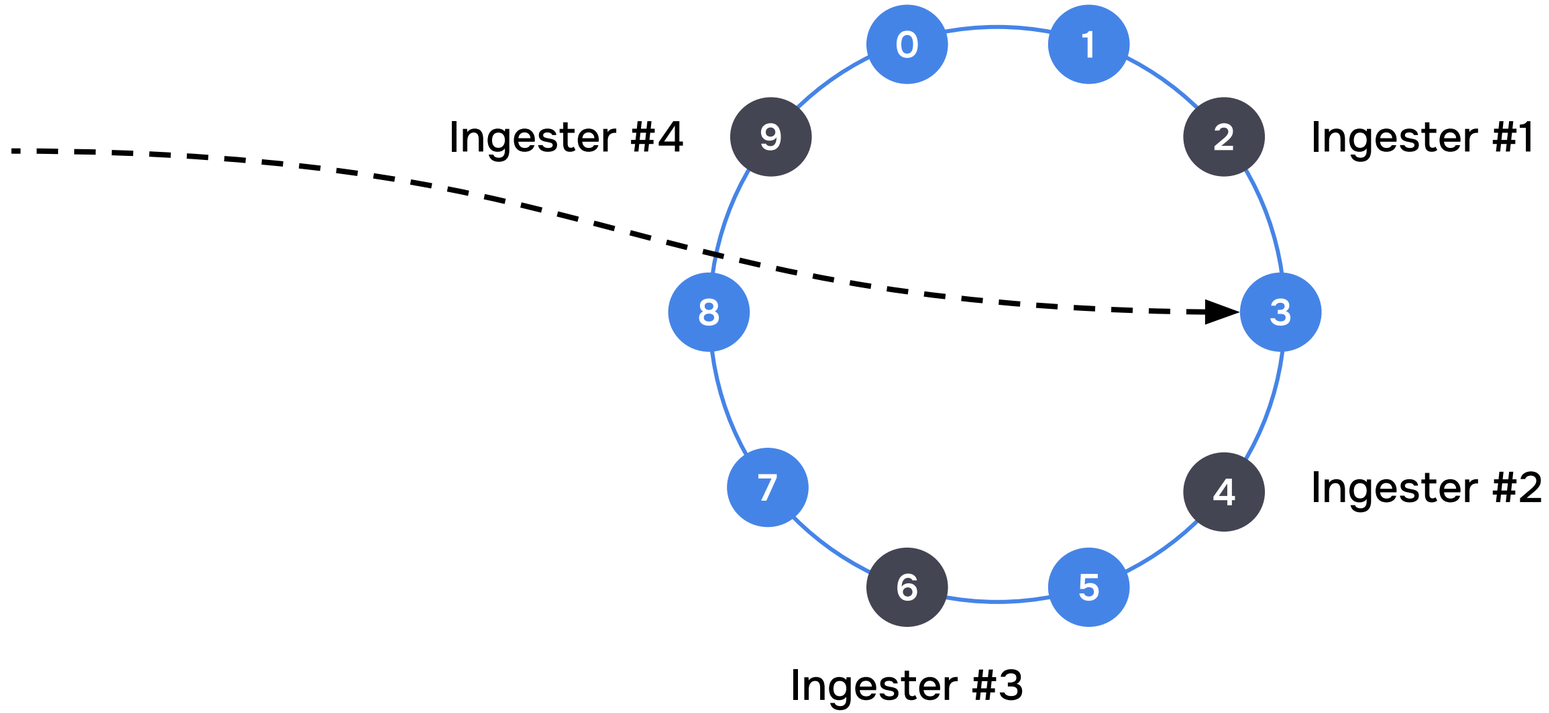
-> token = 3



Хеш-ринг

Series

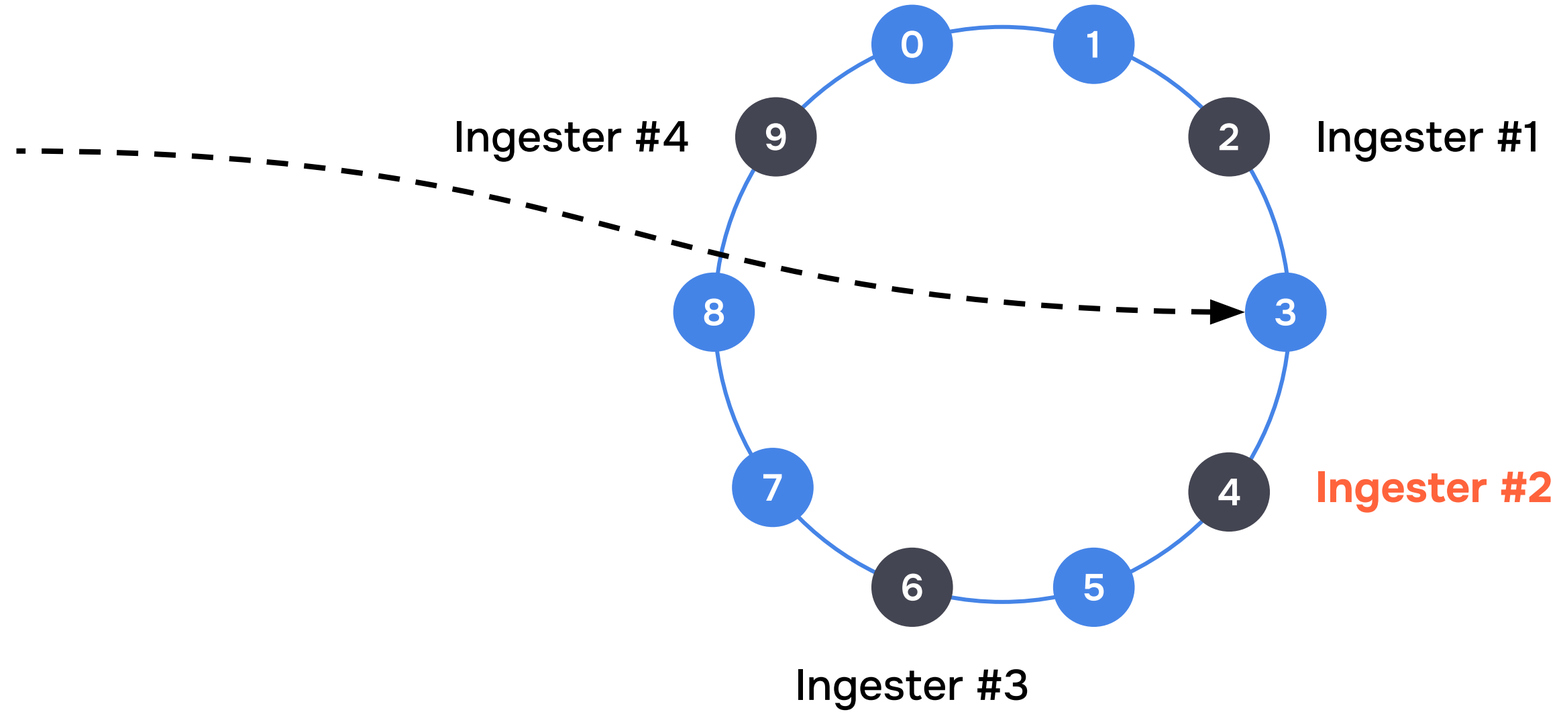
```
{
   __name__="cpu_seconds_total,
   instance: "1.1.1.1"
}
-> token = 3
```



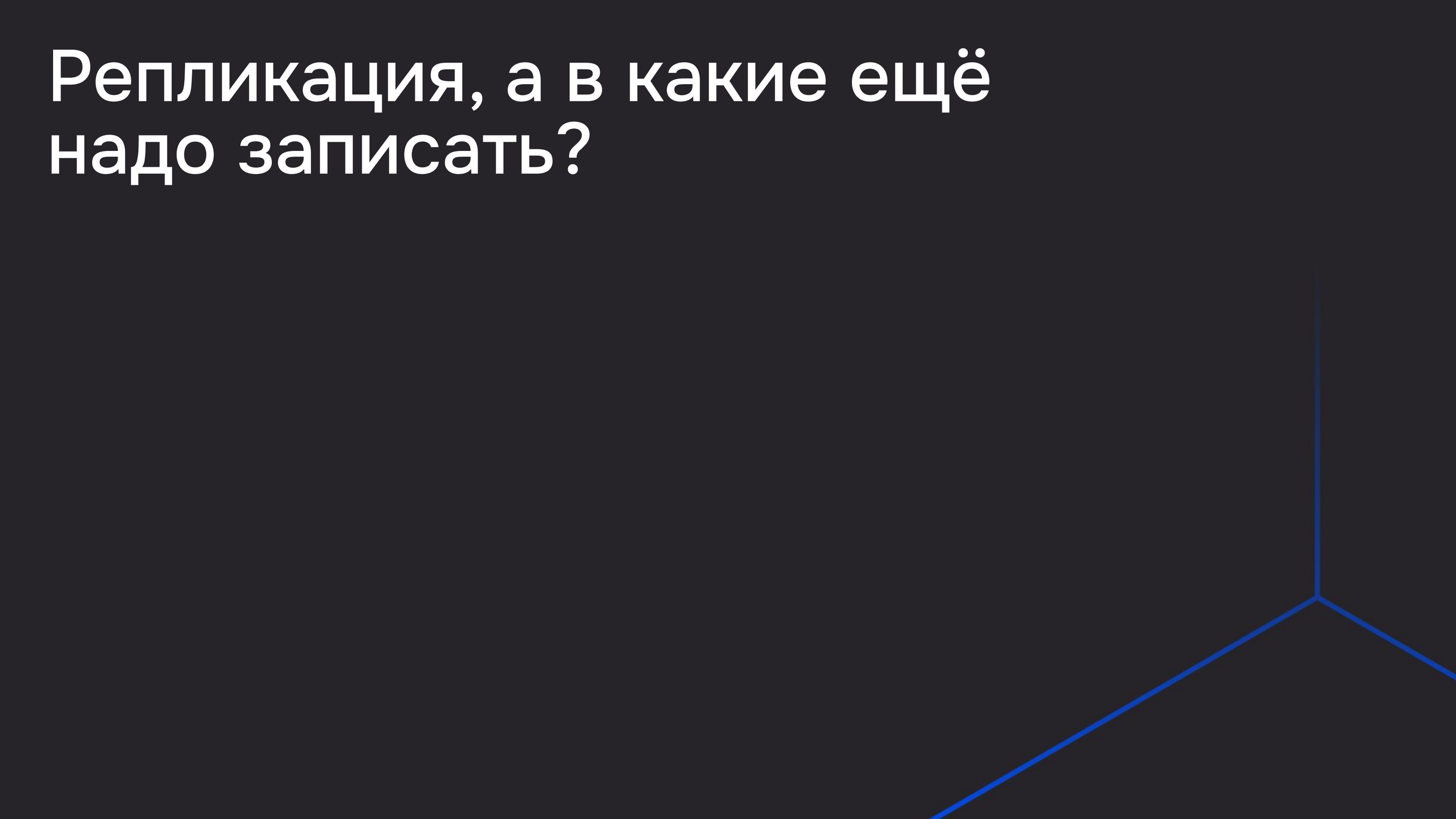
Хеш-ринг

Series

```
{
   __name__="cpu_seconds_total,
   instance: "1.1.1.1"
}
-> token = 3
```



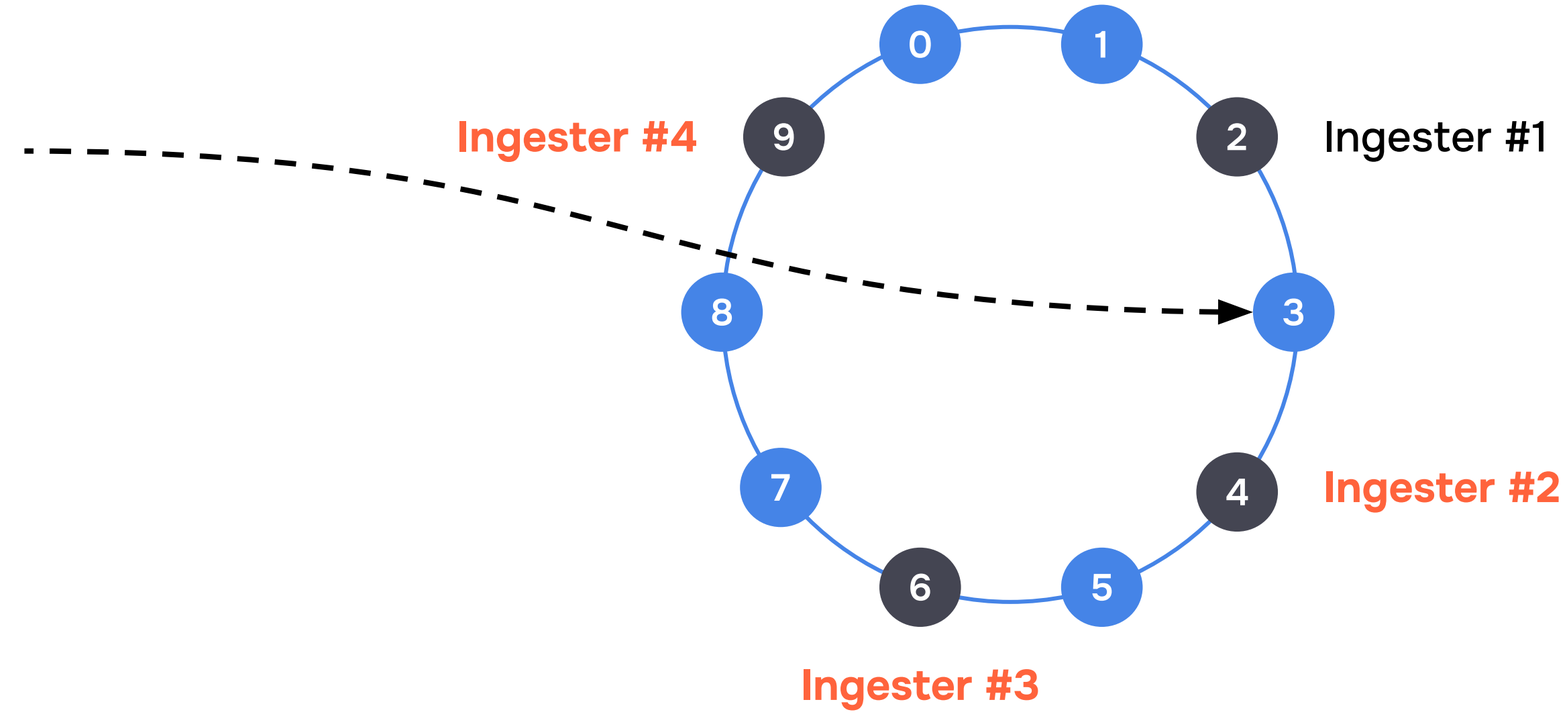
Репликация, а в какие ещё надо записать?



Хеш-ринг

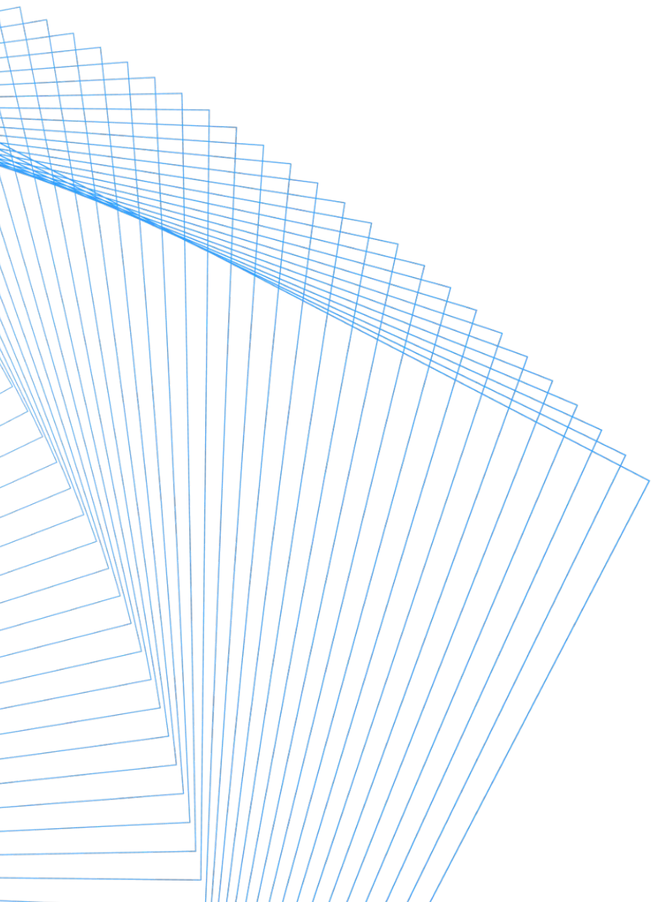
Series

```
{
   __name__="cpu_seconds_total,
   instance: "1.1.1.1"
}
-> token = 3
```

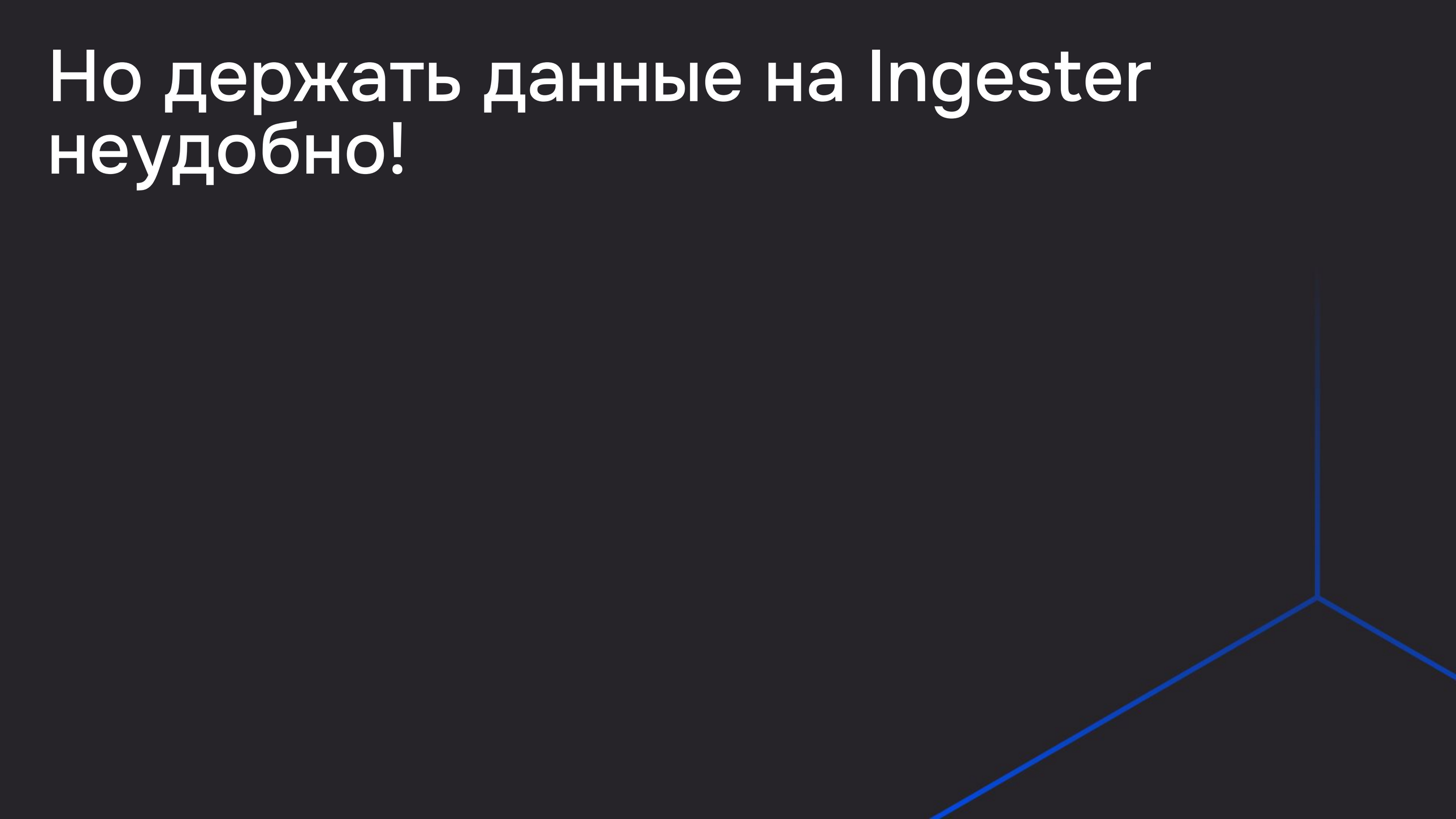


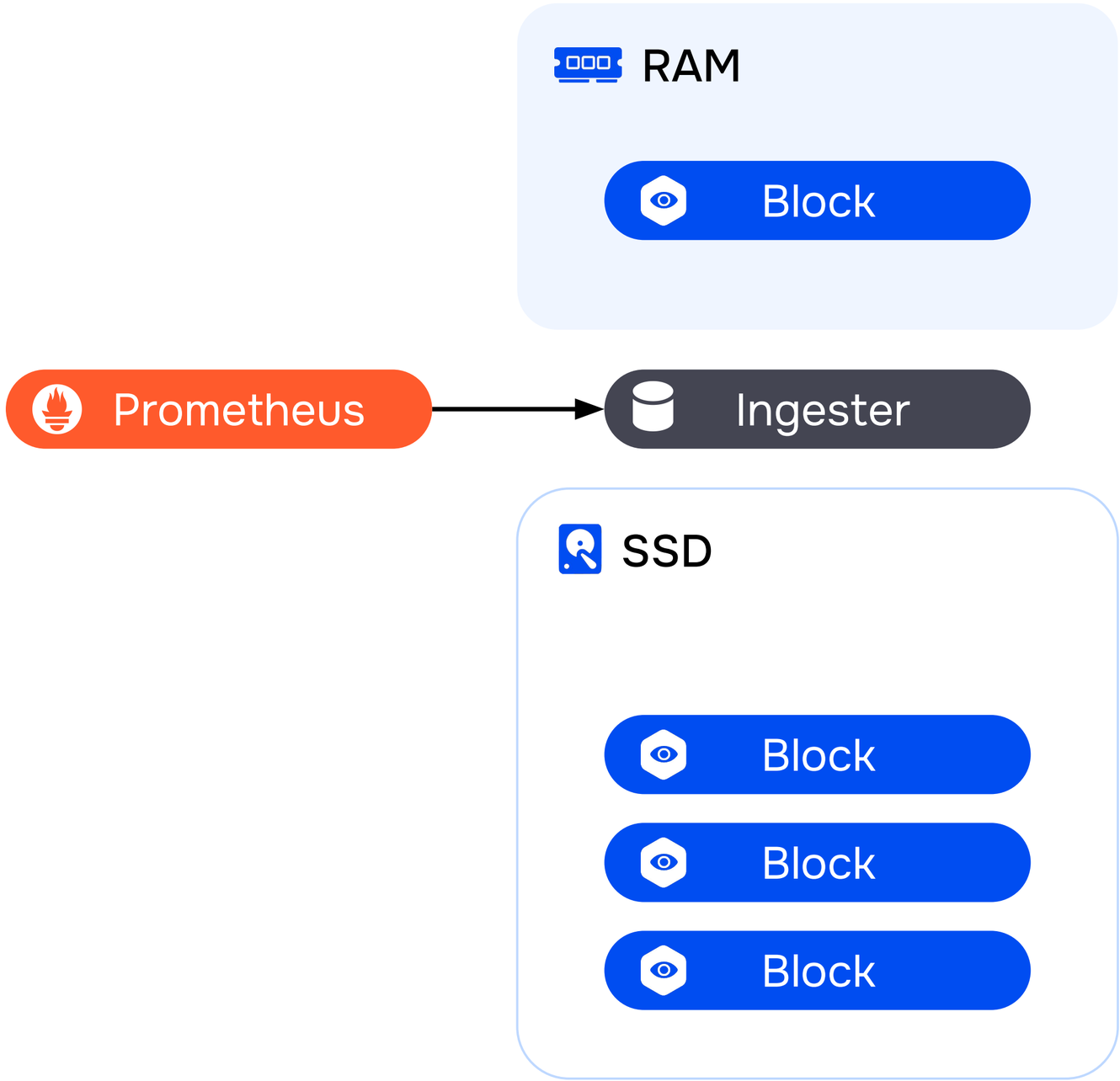
Централизованная система мониторинга

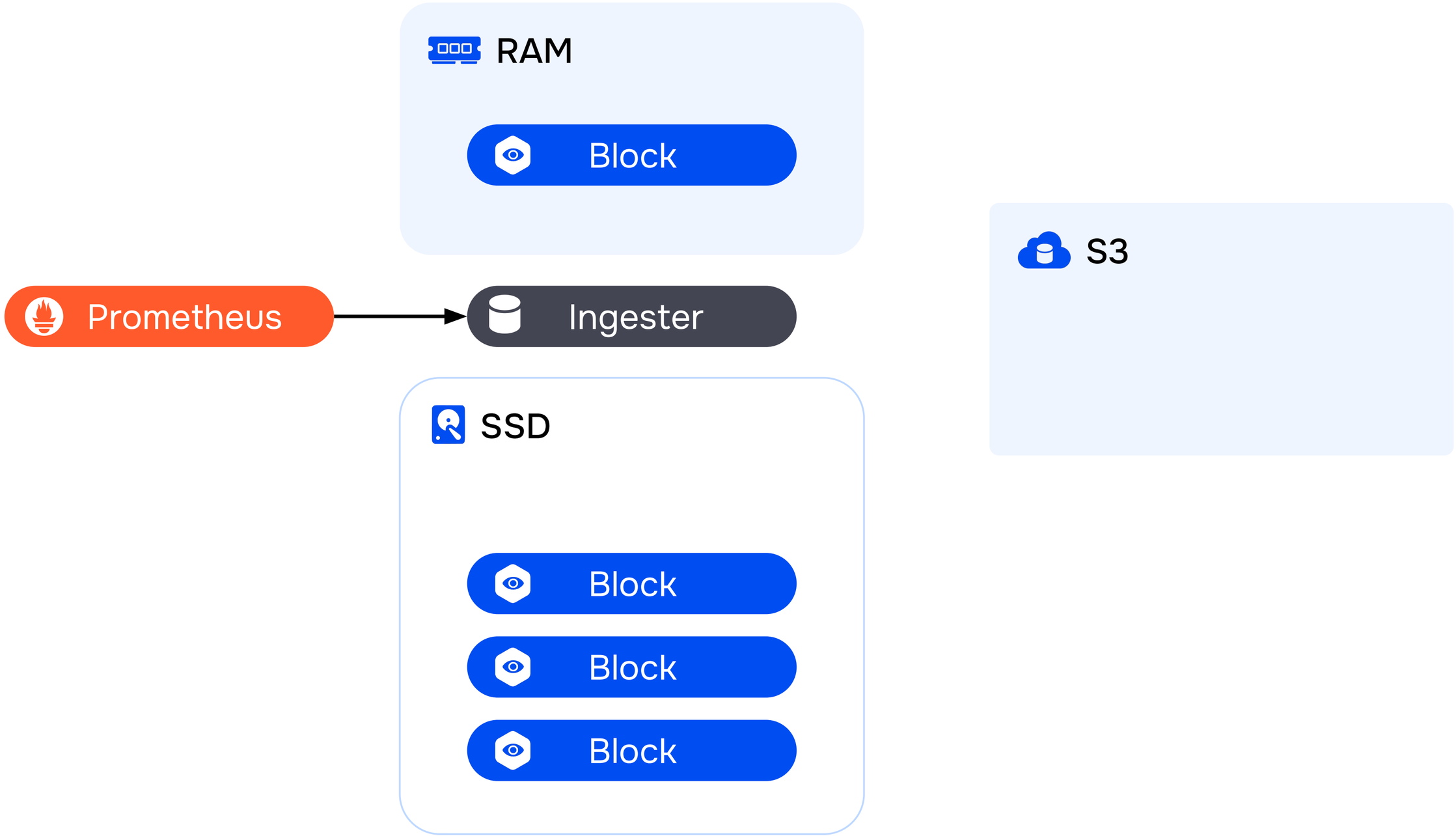
- Отказоустойчивость на запись: WC = RF/2 + 1
- Отказоустойчивость на чтение: RC = RF - WC + 1
- Шардирование на запись на основе хеш-ринга

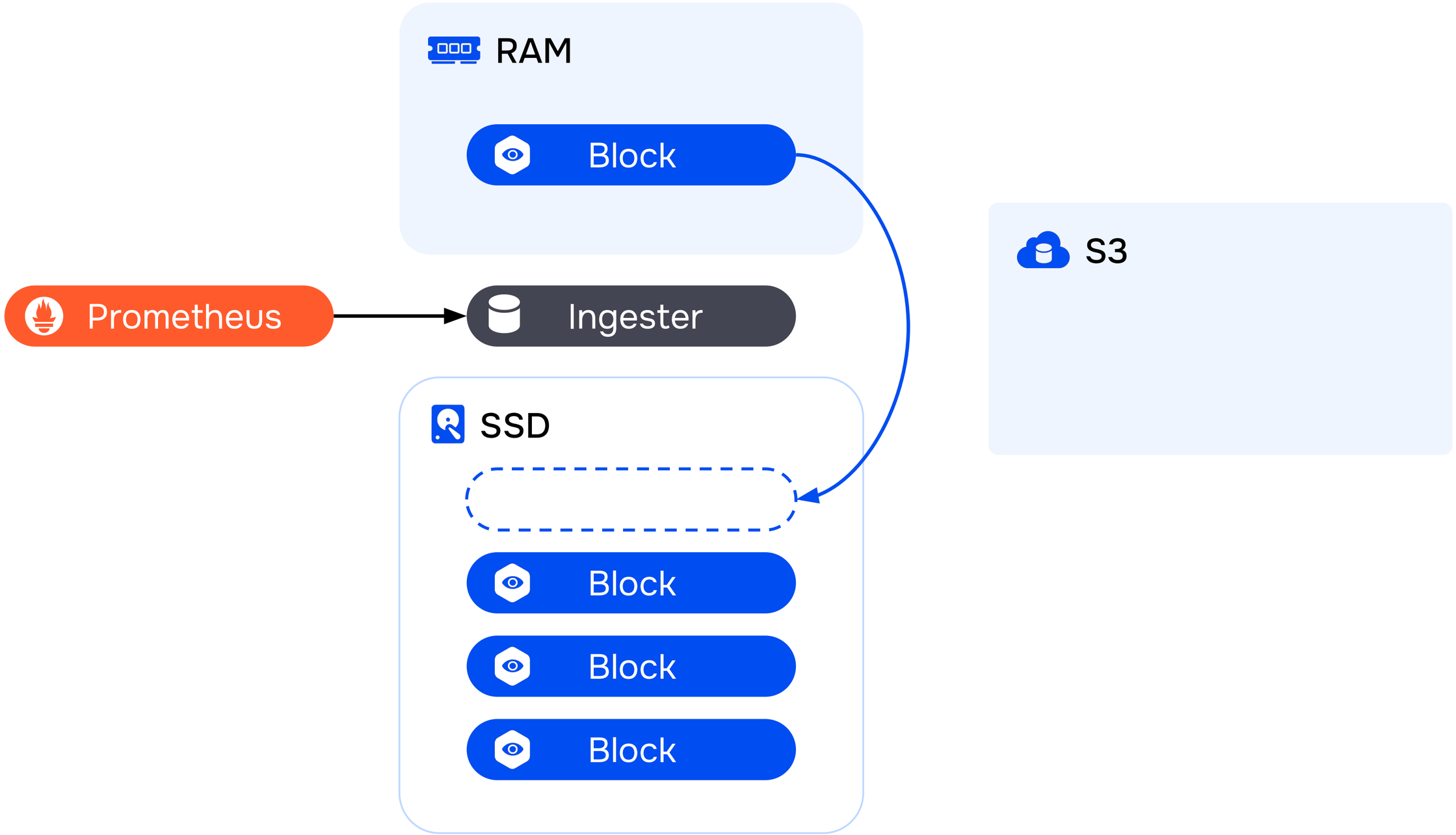


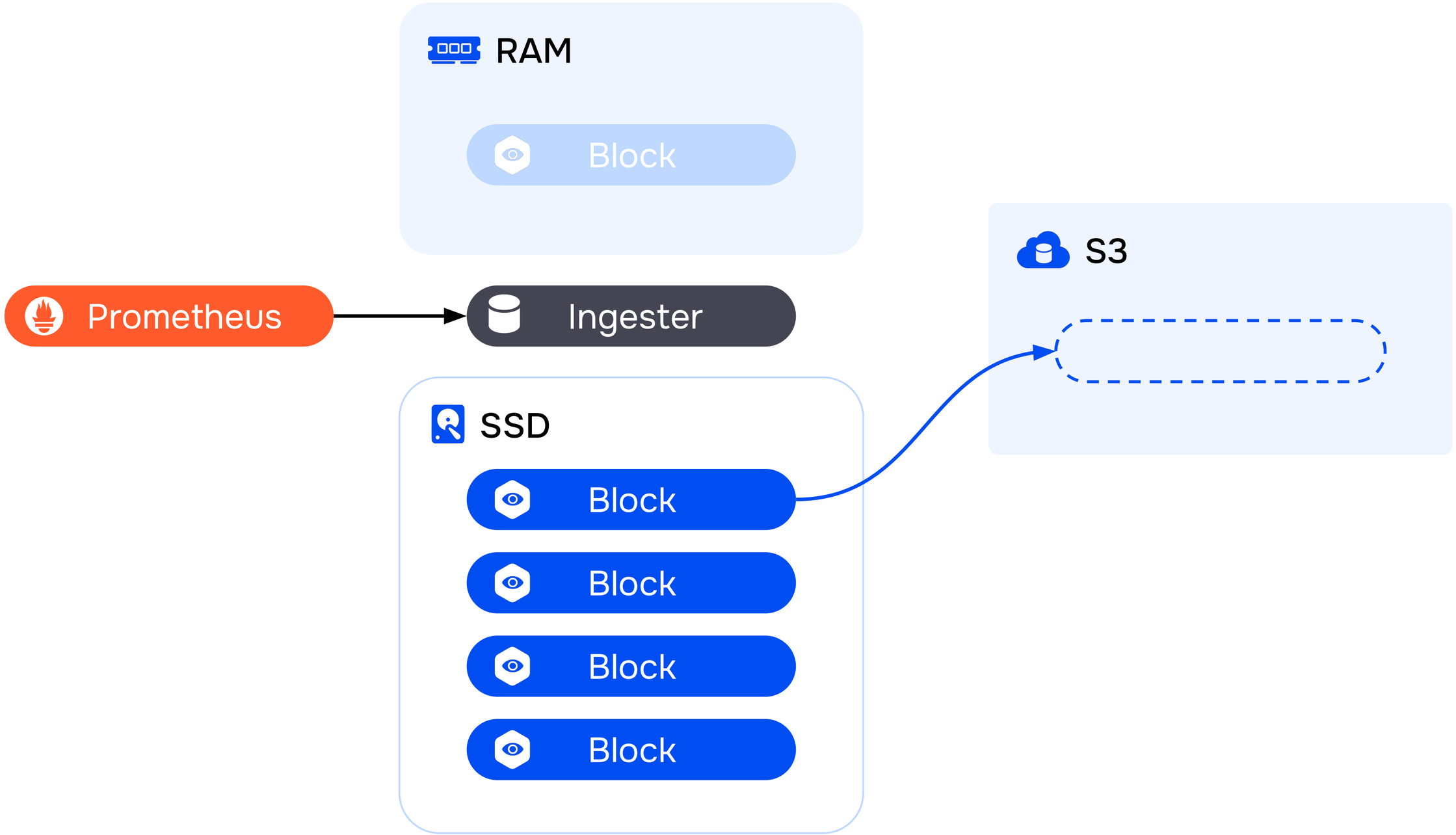
Но держать данные на Ingester неудобно!

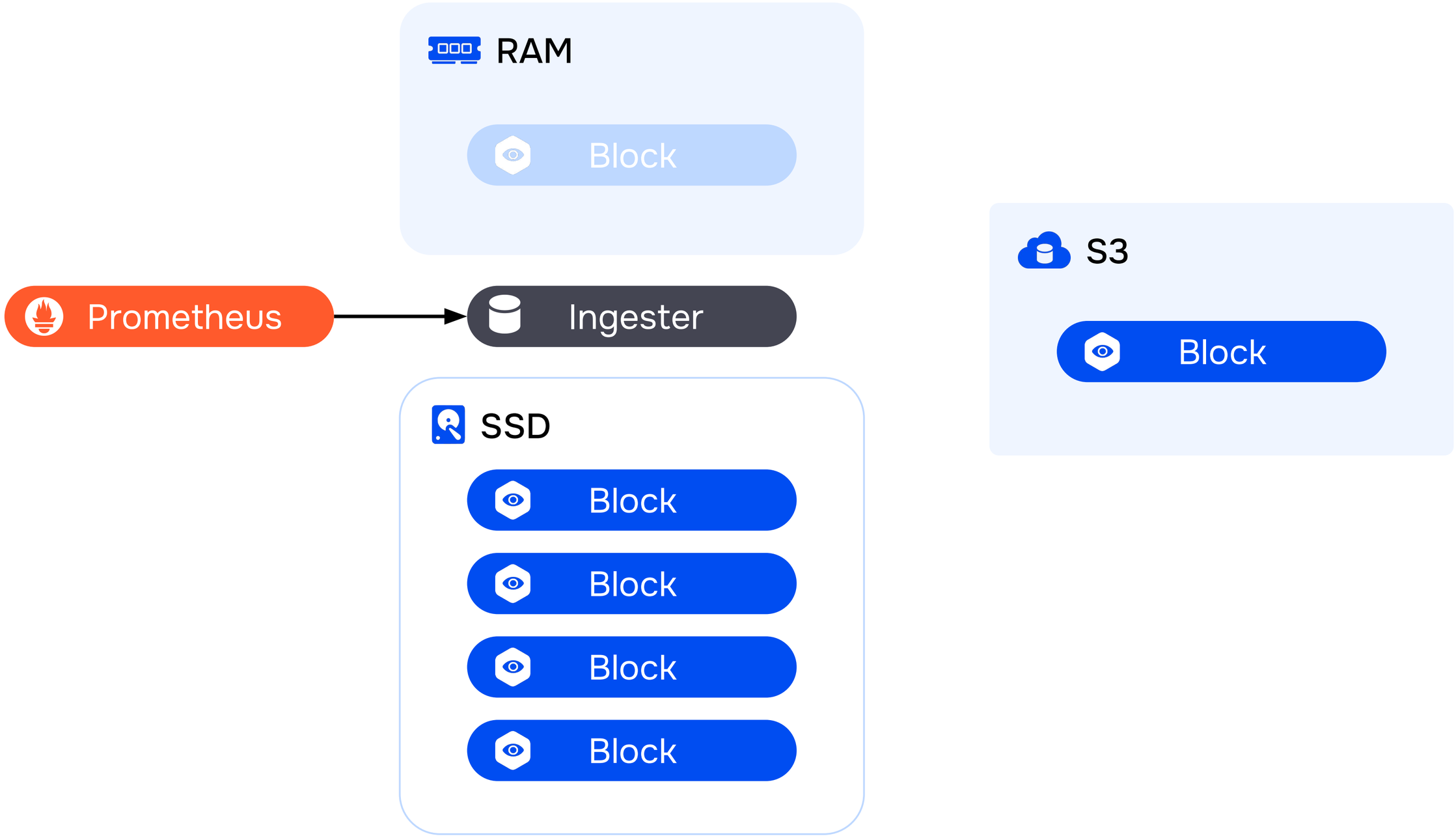


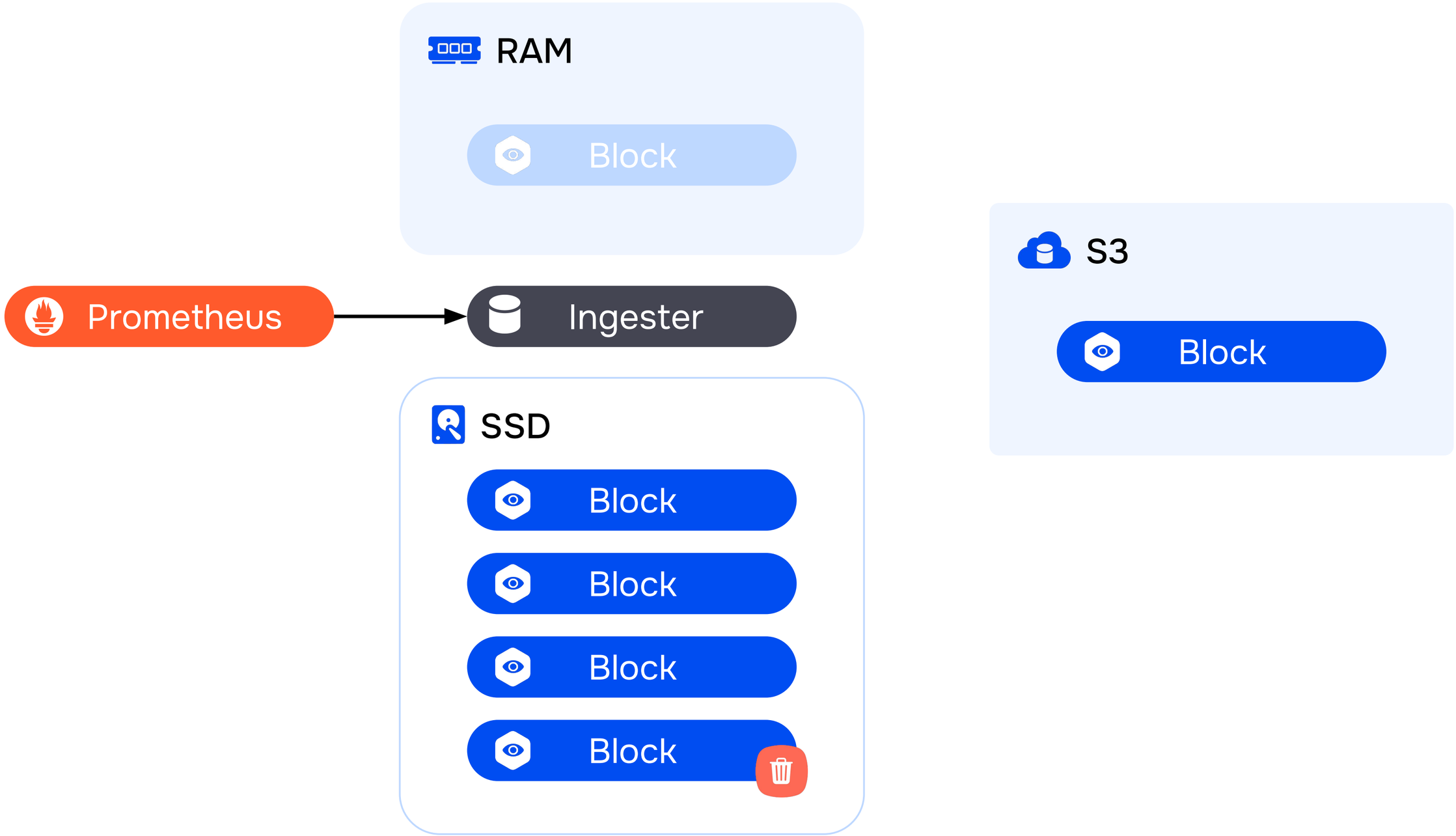

RAM
Block
Prometheus
Ingester
SSD
Block
Block
Block

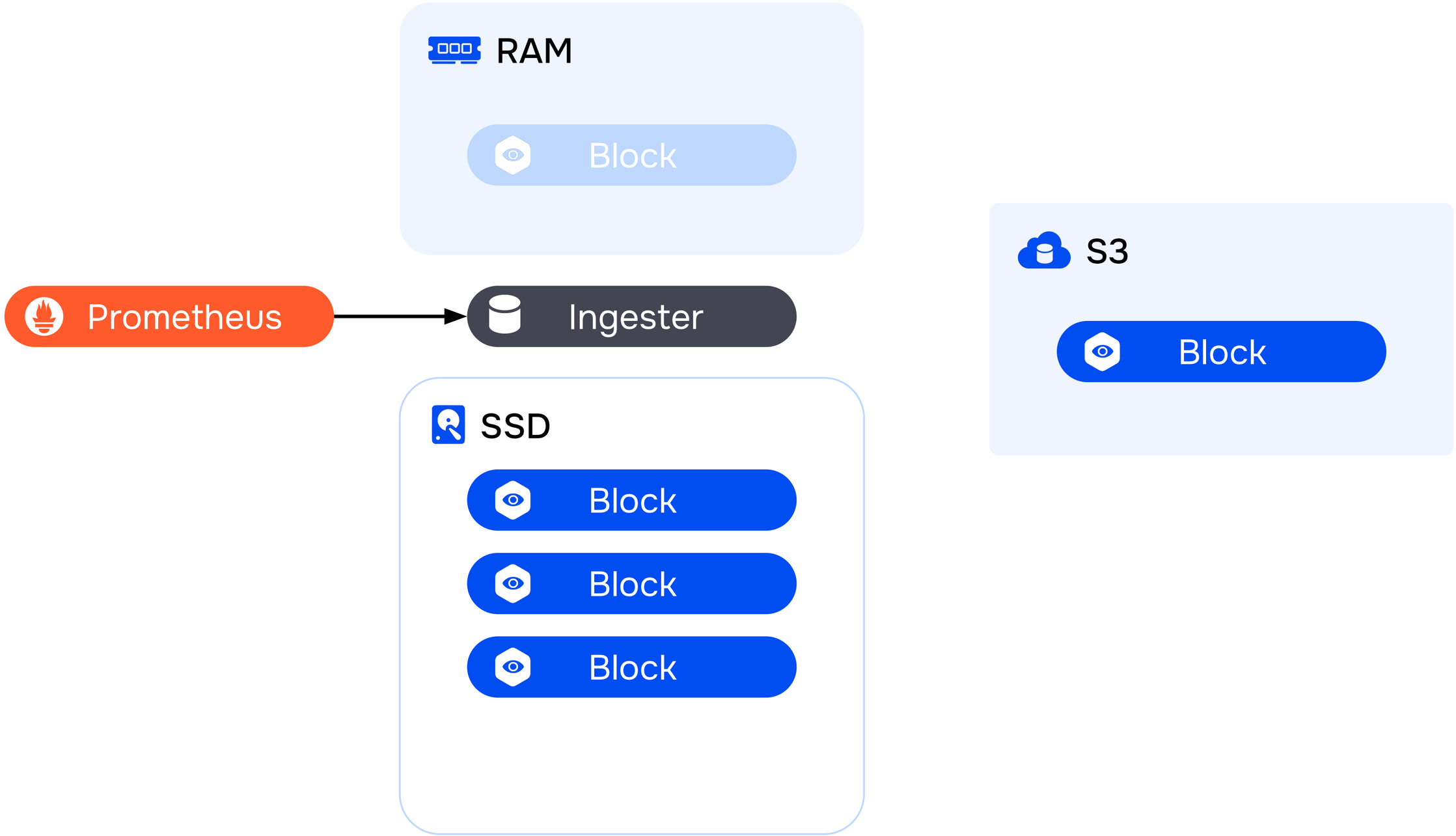

RAM
Block
Prometheus
Ingester
S3
SSD
Block
Block
Block

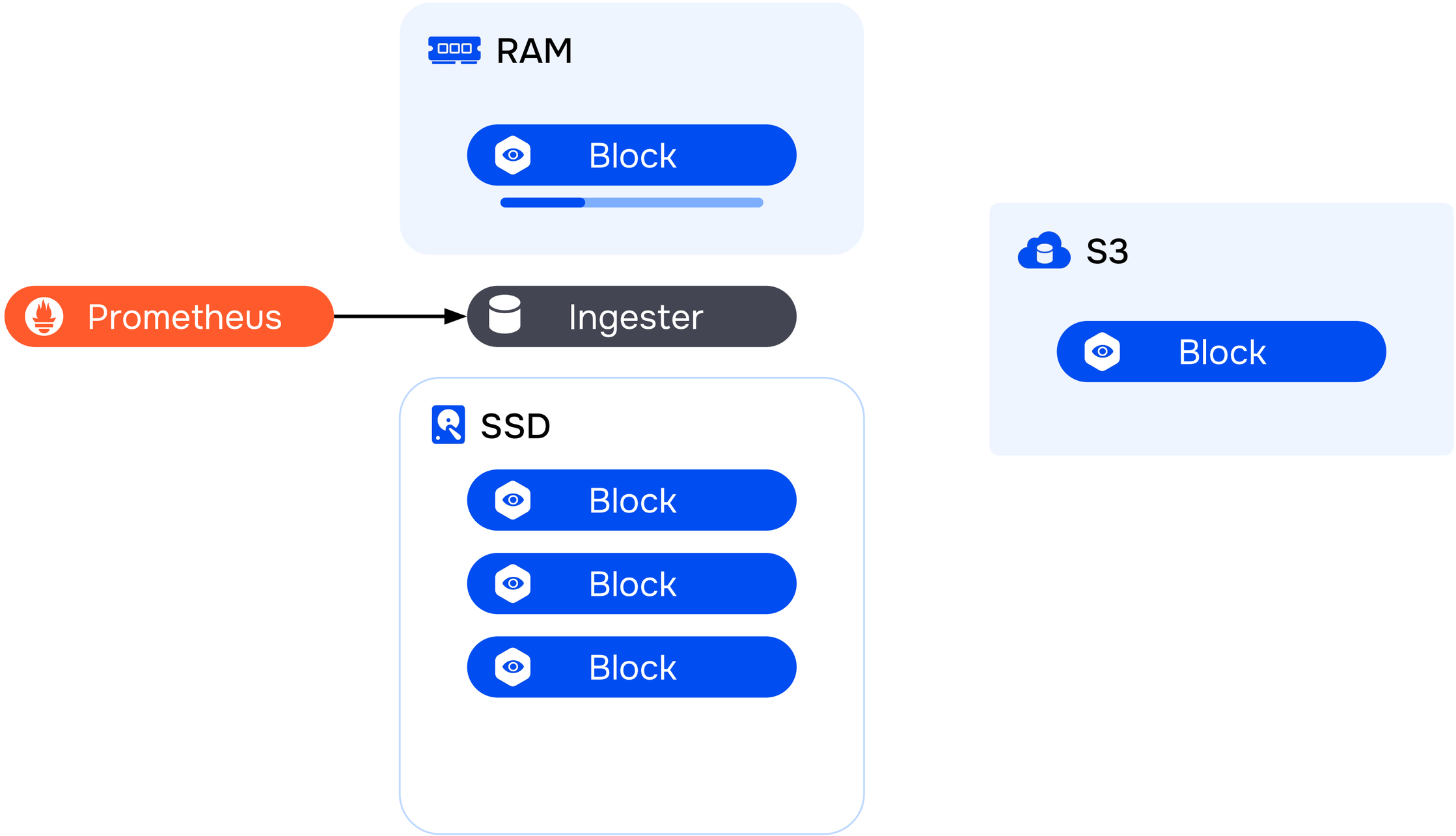

RAM
Block
Prometheus
Ingester
S3
SSD
Block
Block
Block

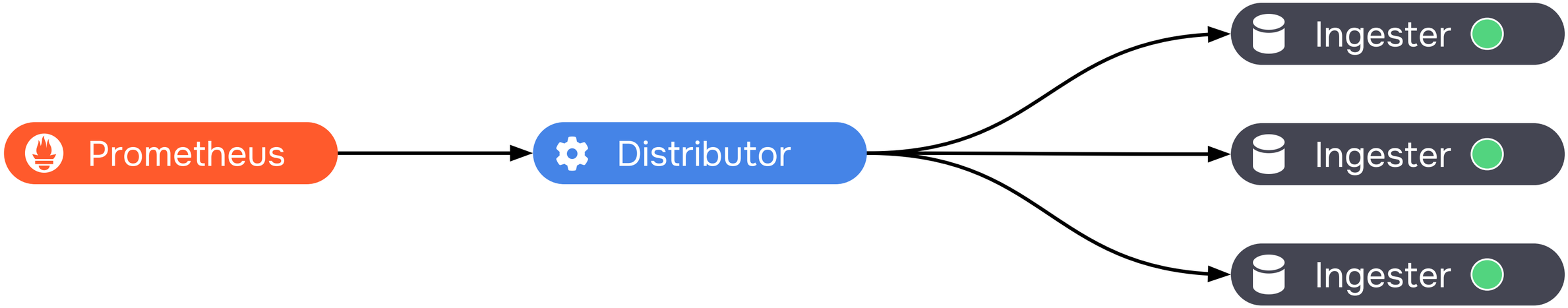

RAM
Block
Prometheus
Ingester
S3
SSD
Block
Block
Block
Block


RAM
Block
Prometheus
Ingester
S3
Block
SSD
Block
Block
Block
Block


RAM
Block
Prometheus
Ingester
S3
Block
SSD
Block
Block
Block
Block

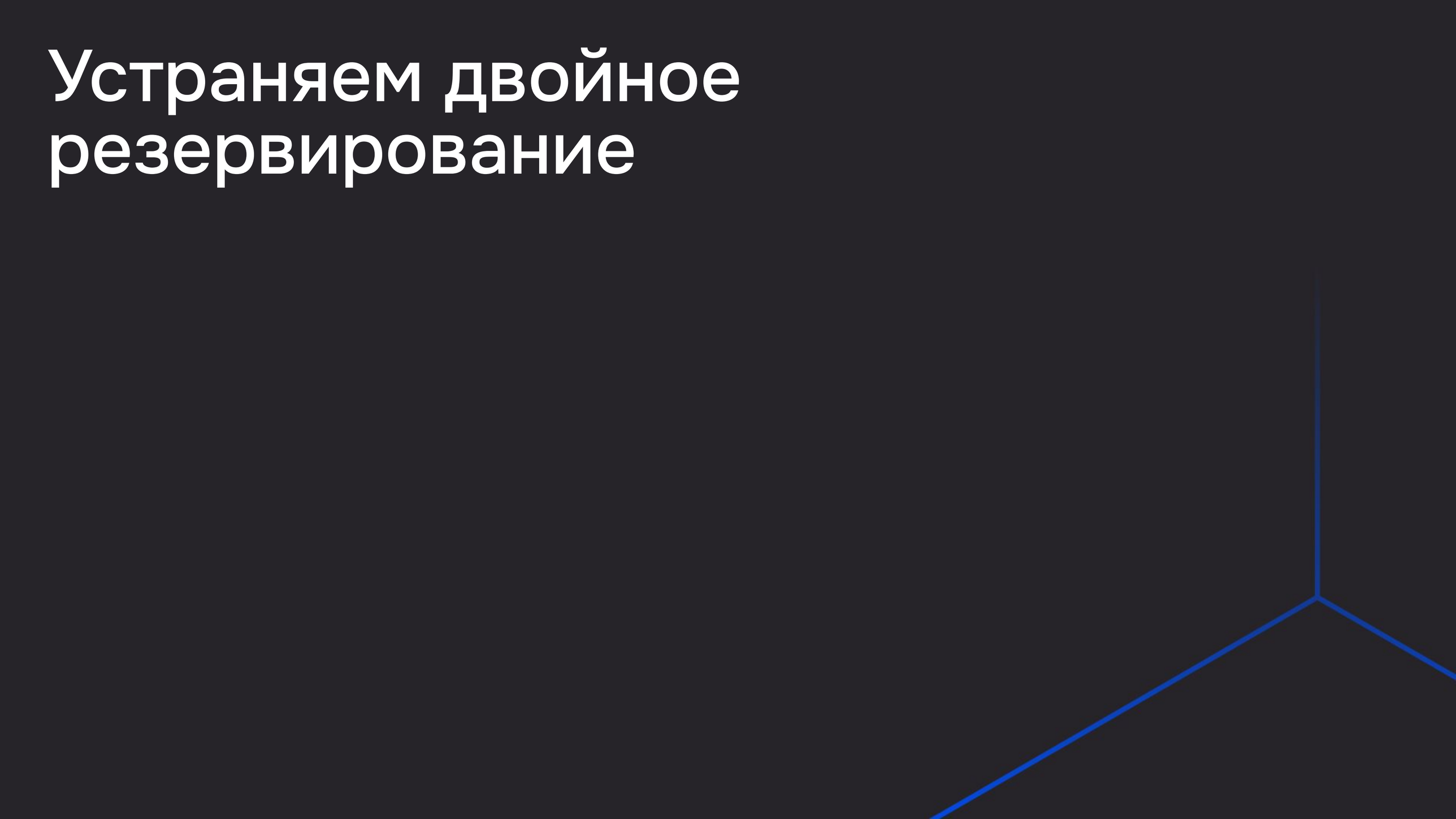

RAM
Block
Prometheus
Ingester
S3
Block
SSD
Block
Block
Block

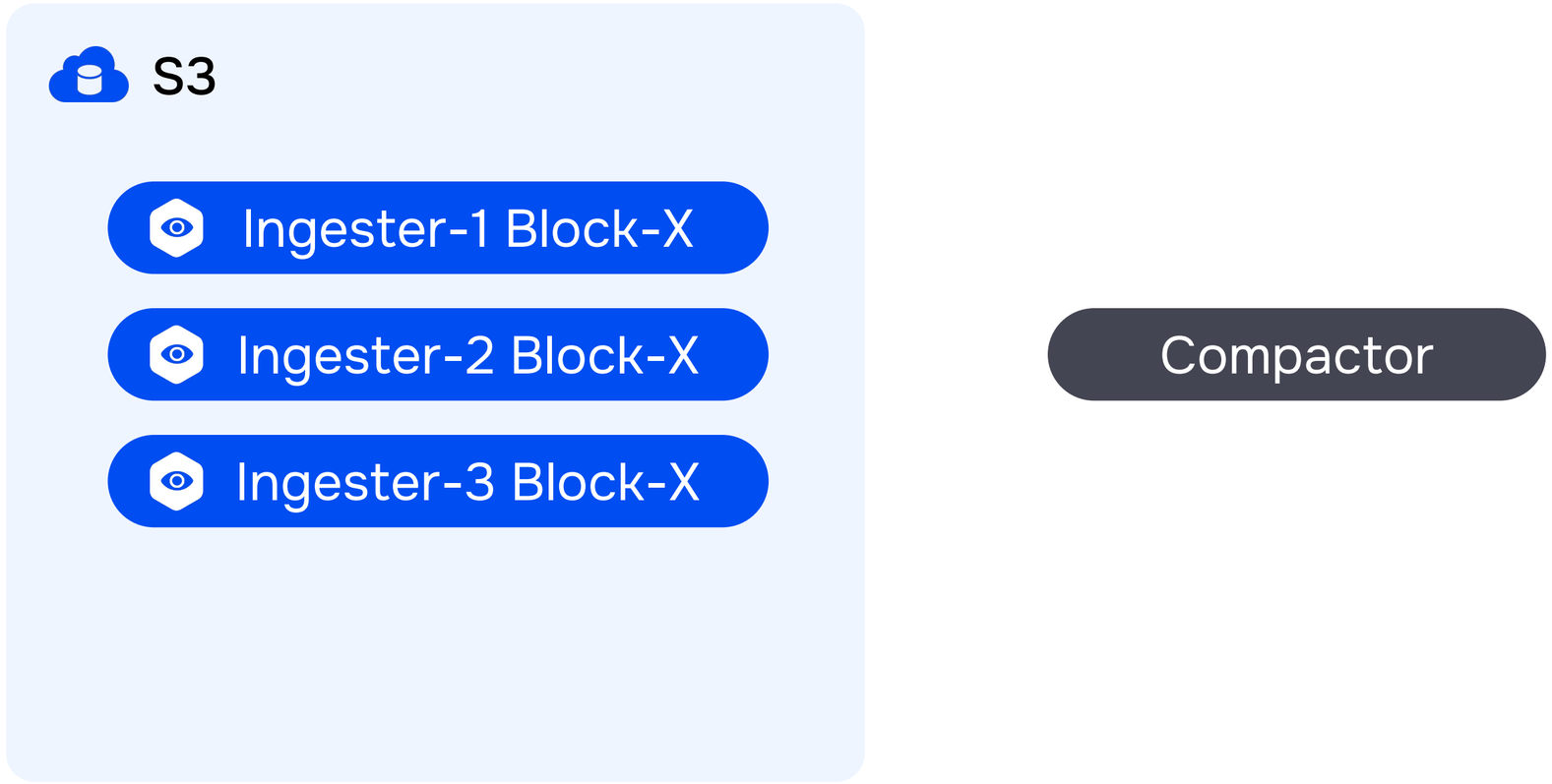

RAM
Block
Prometheus
Ingester
S3
Block
SSD
Block
Block
Block

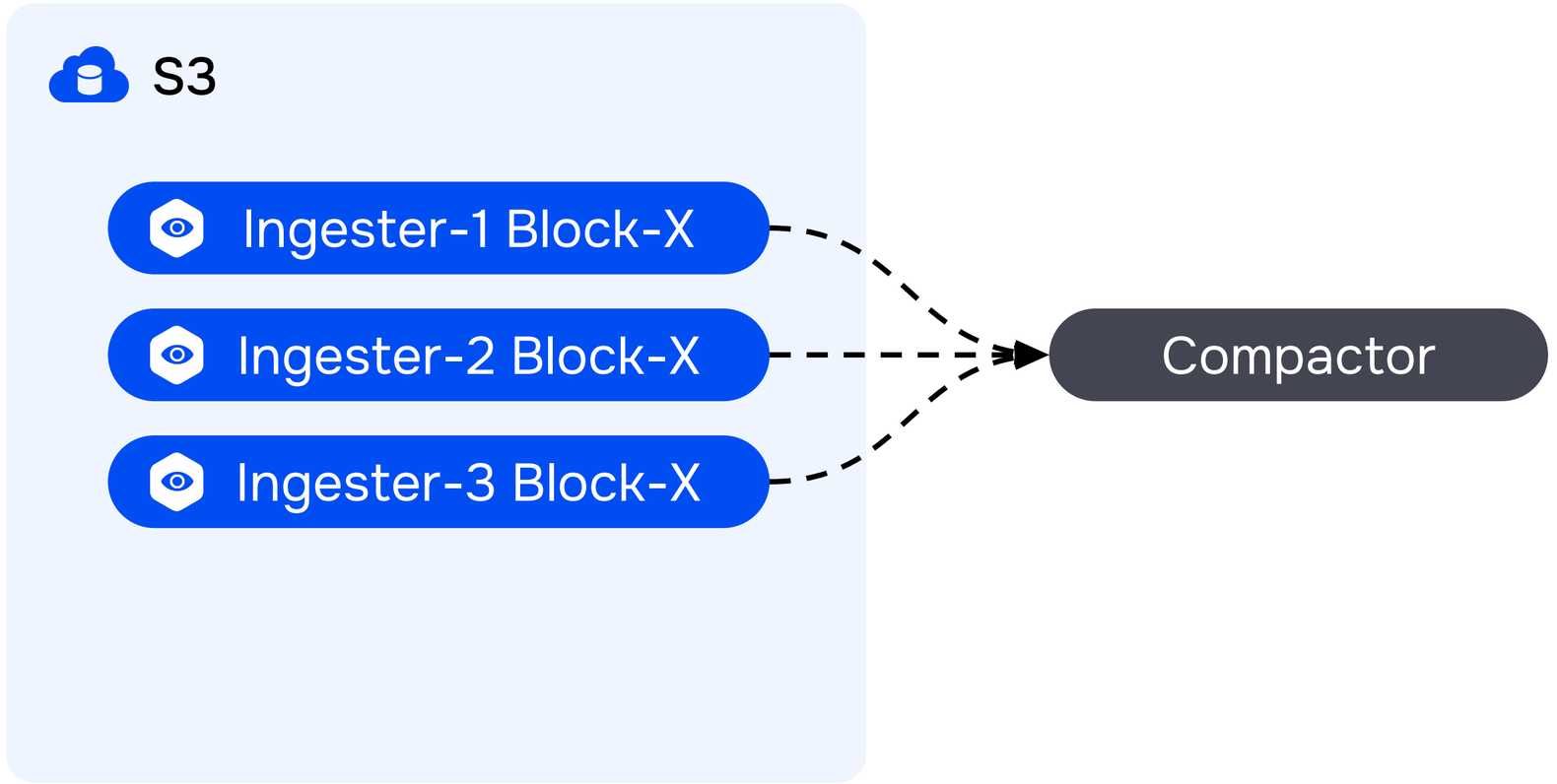

Prometheus
Distributor
Ingester
Ingester
Ingester

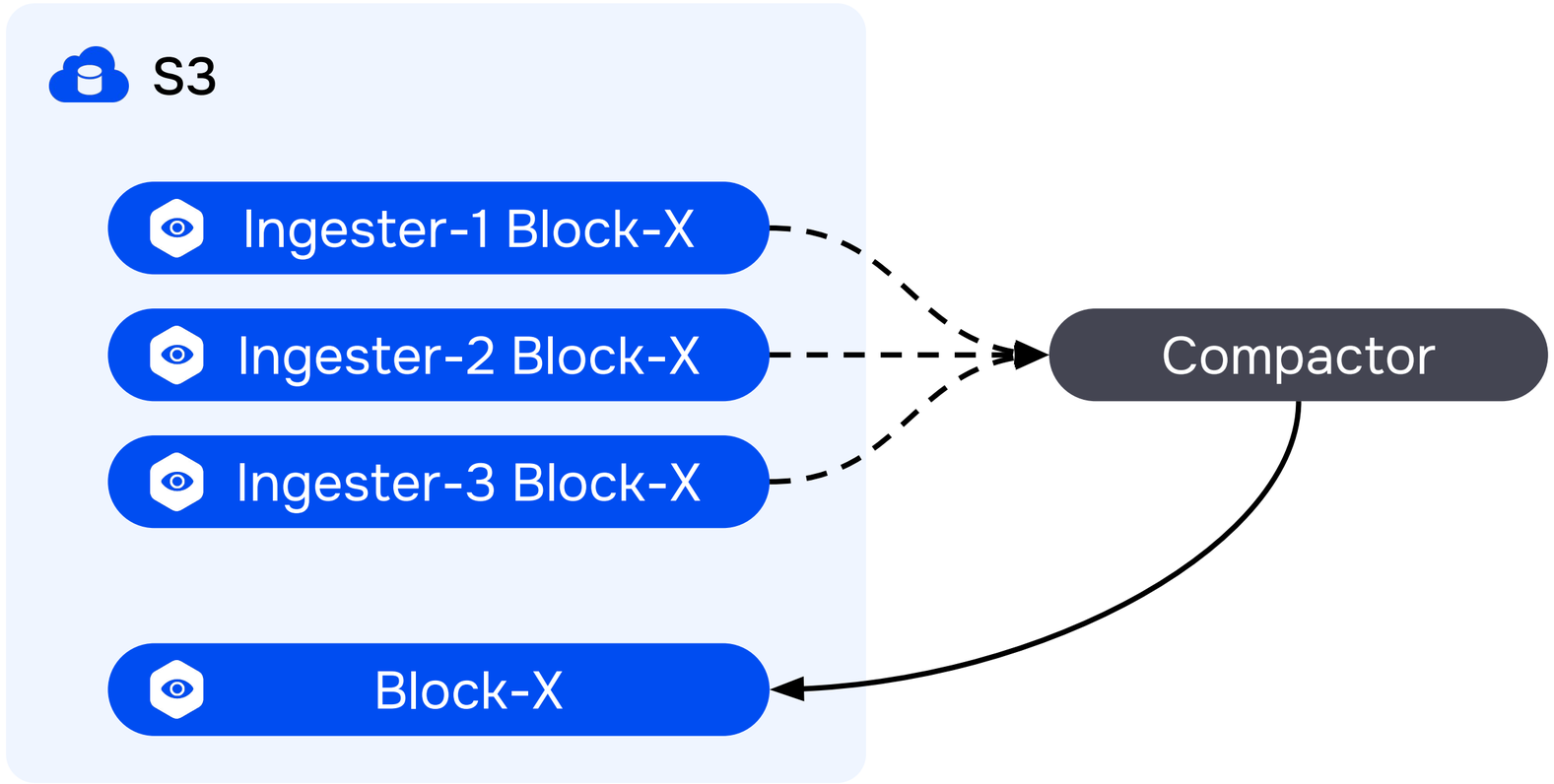

S3
Ingester-1 Block-X
Ingester-2 Block-X
Ingester-3 Block-X

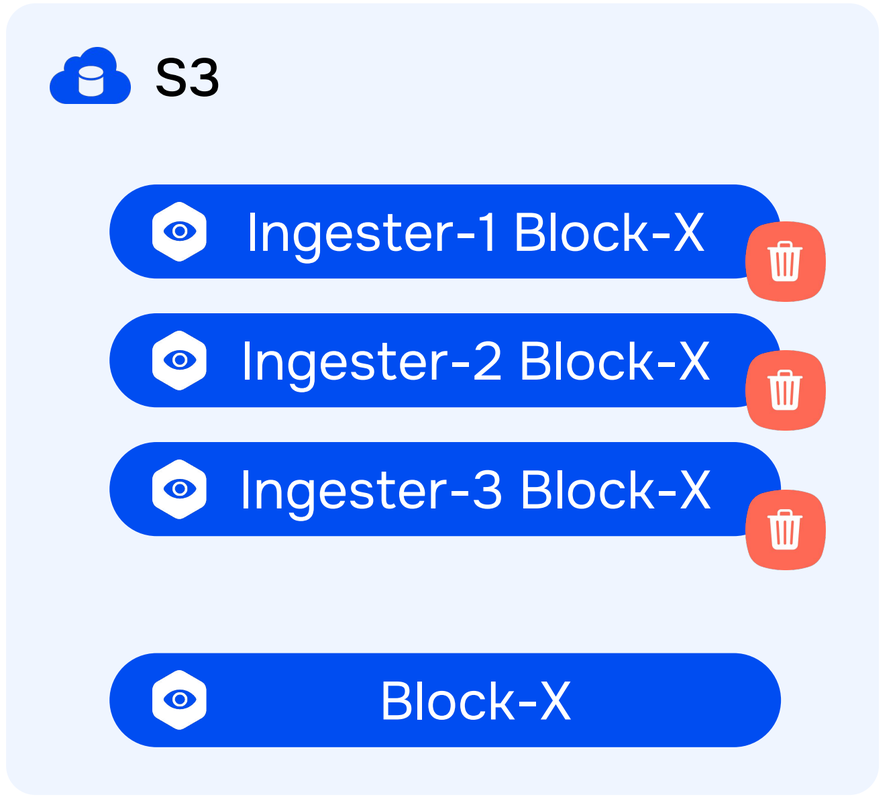
Устраняем двойное резервирование



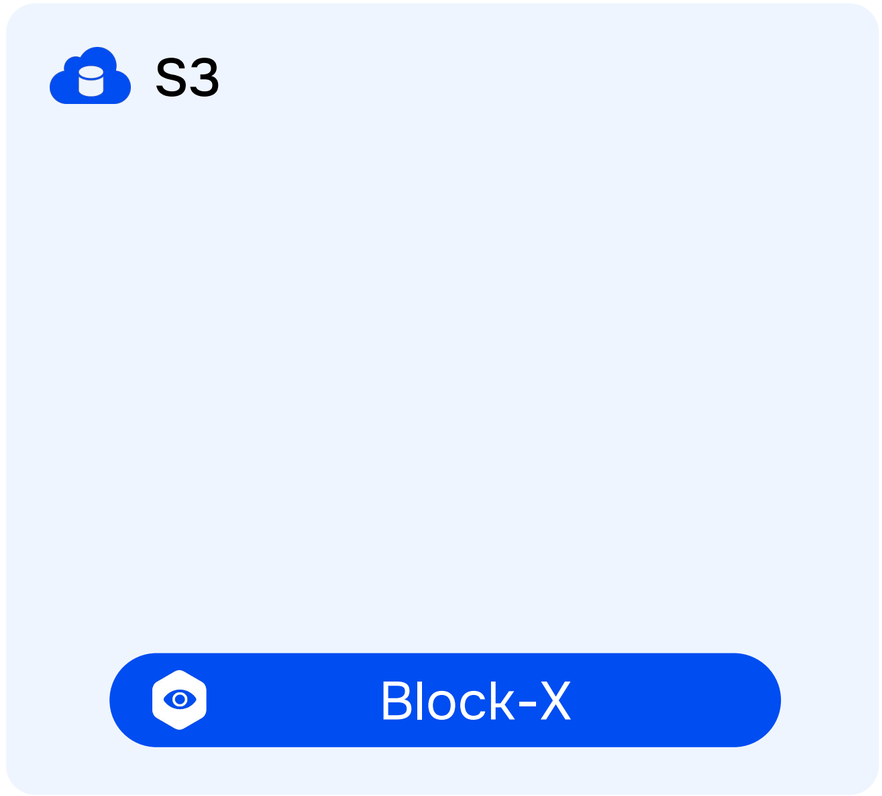

S3
Ingester-1 Block-X
Ingester-2 Block-X
Ingester-3 Block-X
Compactor


S3
Ingester-1 Block-X
Ingester-2 Block-X
Ingester-3 Block-X
Compactor


S3
Ingester-1 Block-X
Ingester-2 Block-X
Ingester-3 Block-X
Block-X
Compactor


S3
Ingester-1 Block-X
Ingester-2 Block-X
Ingester-3 Block-X
Block-X

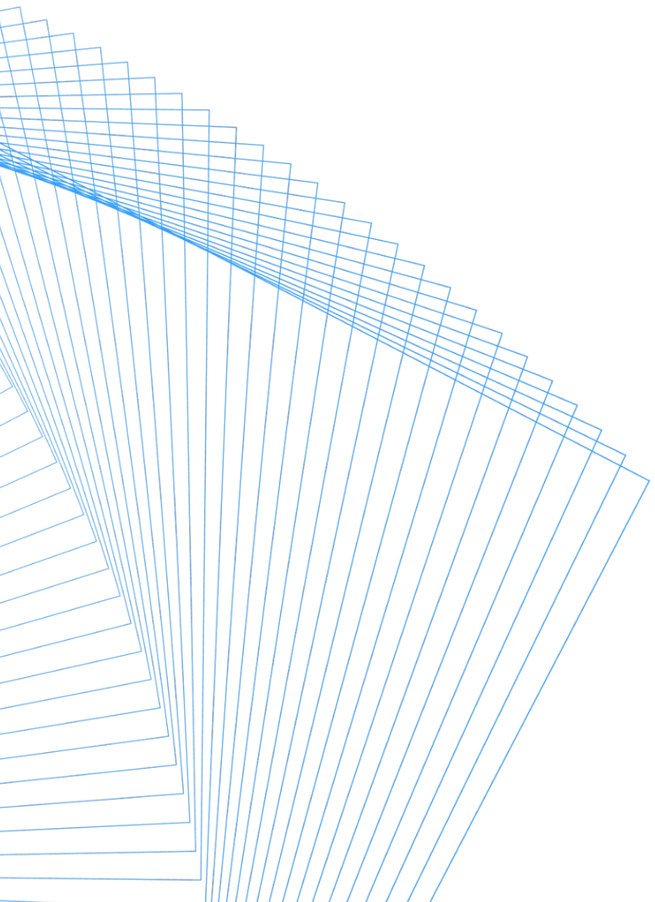
Compactor



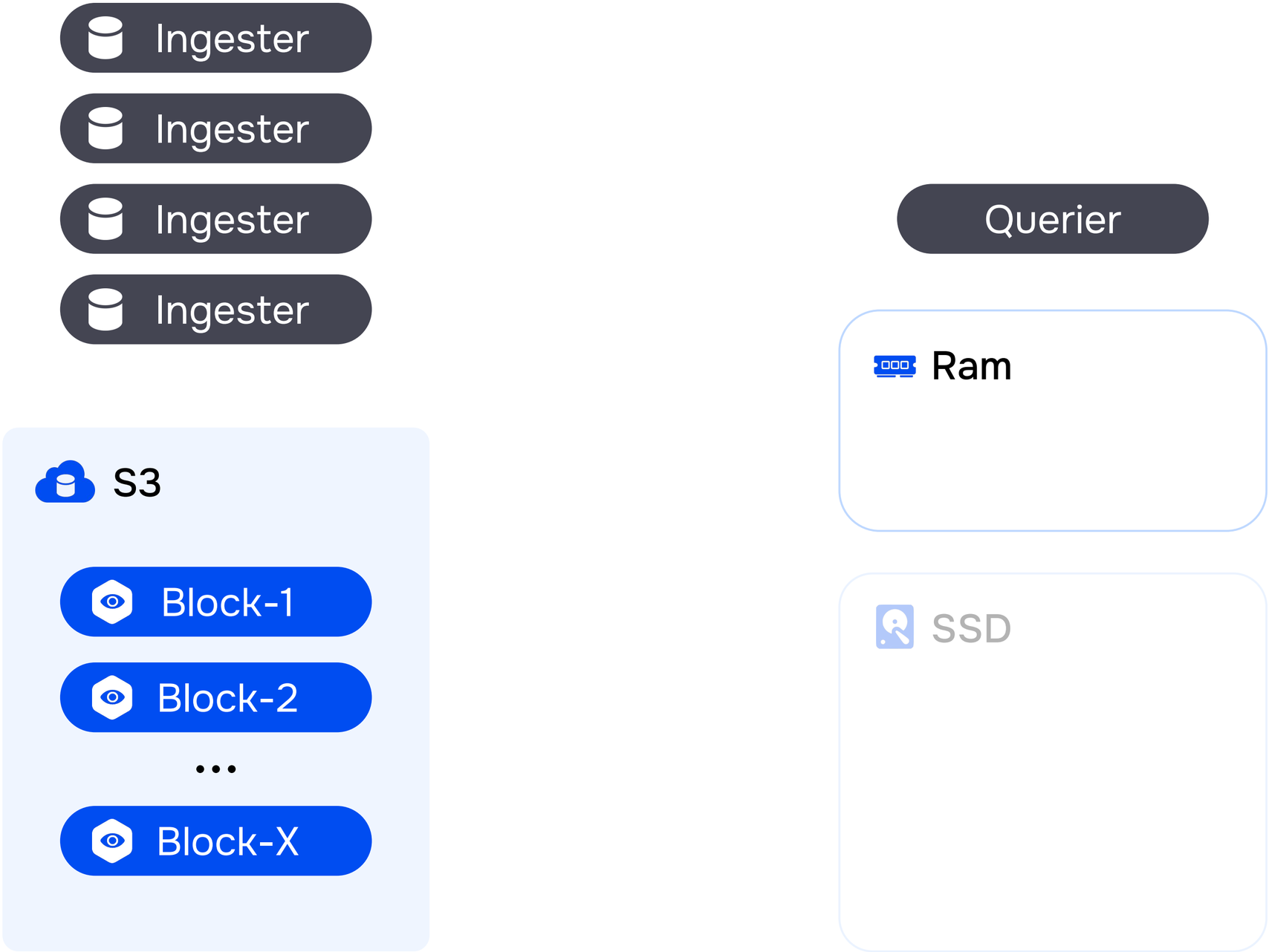
Compactor

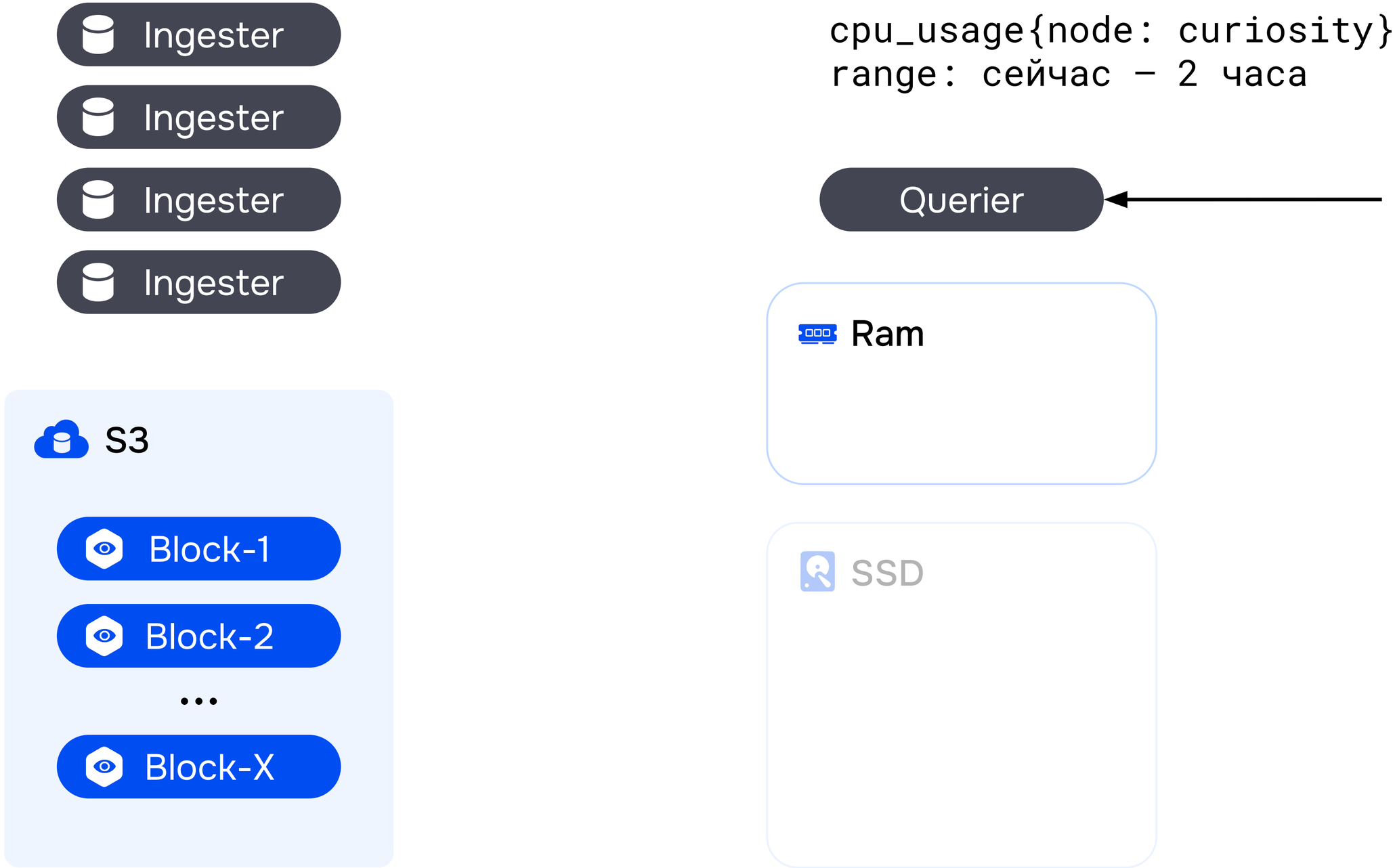
Централизованная система мониторинга

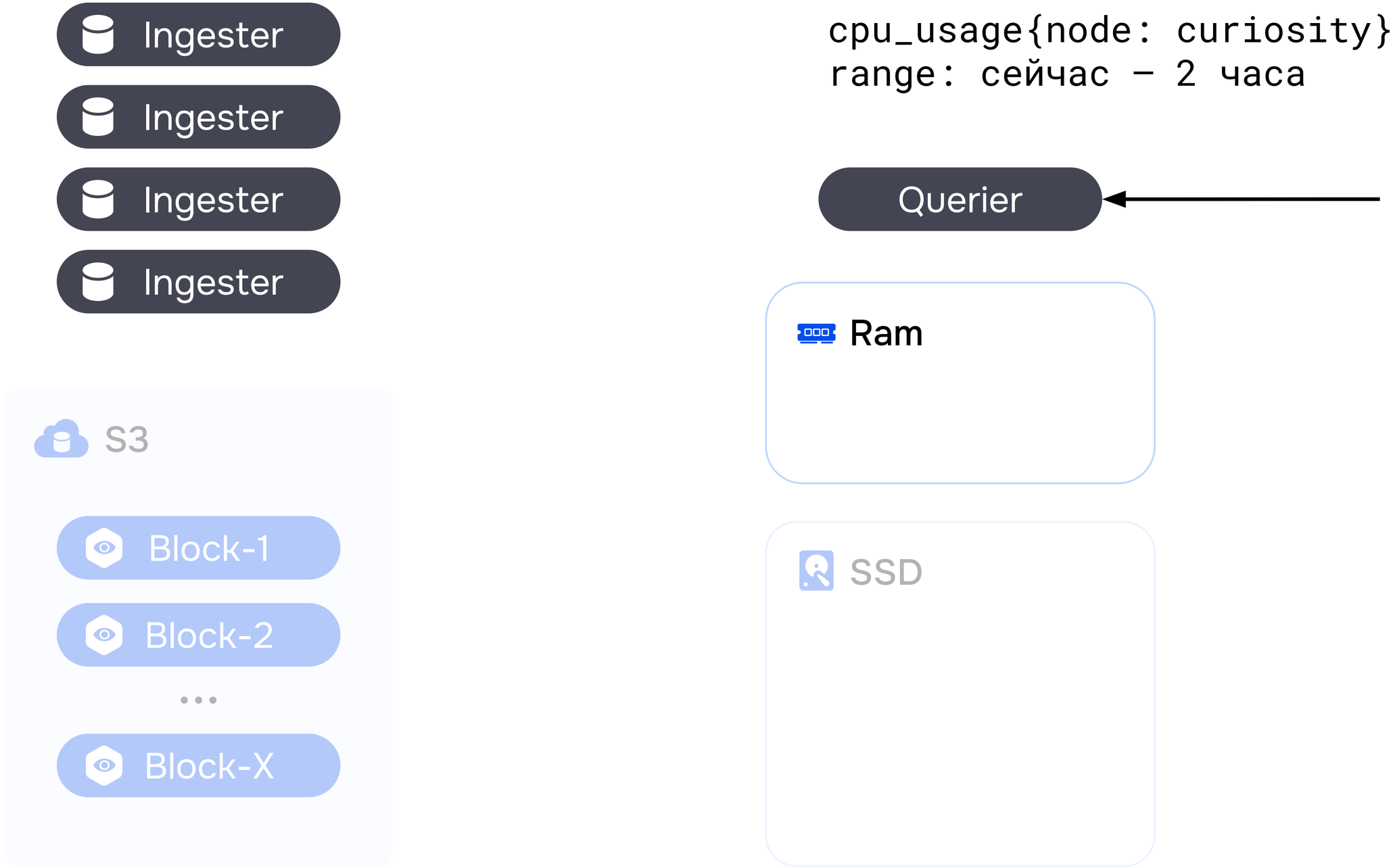
- Отказоустойчивость на запись: WC = RF/2 + 1
- Отказоустойчивость на чтение: RC = RF - WC + 1
- Шардирование на запись на основе хеш-ринга
- Исторические данные хранятся в S3

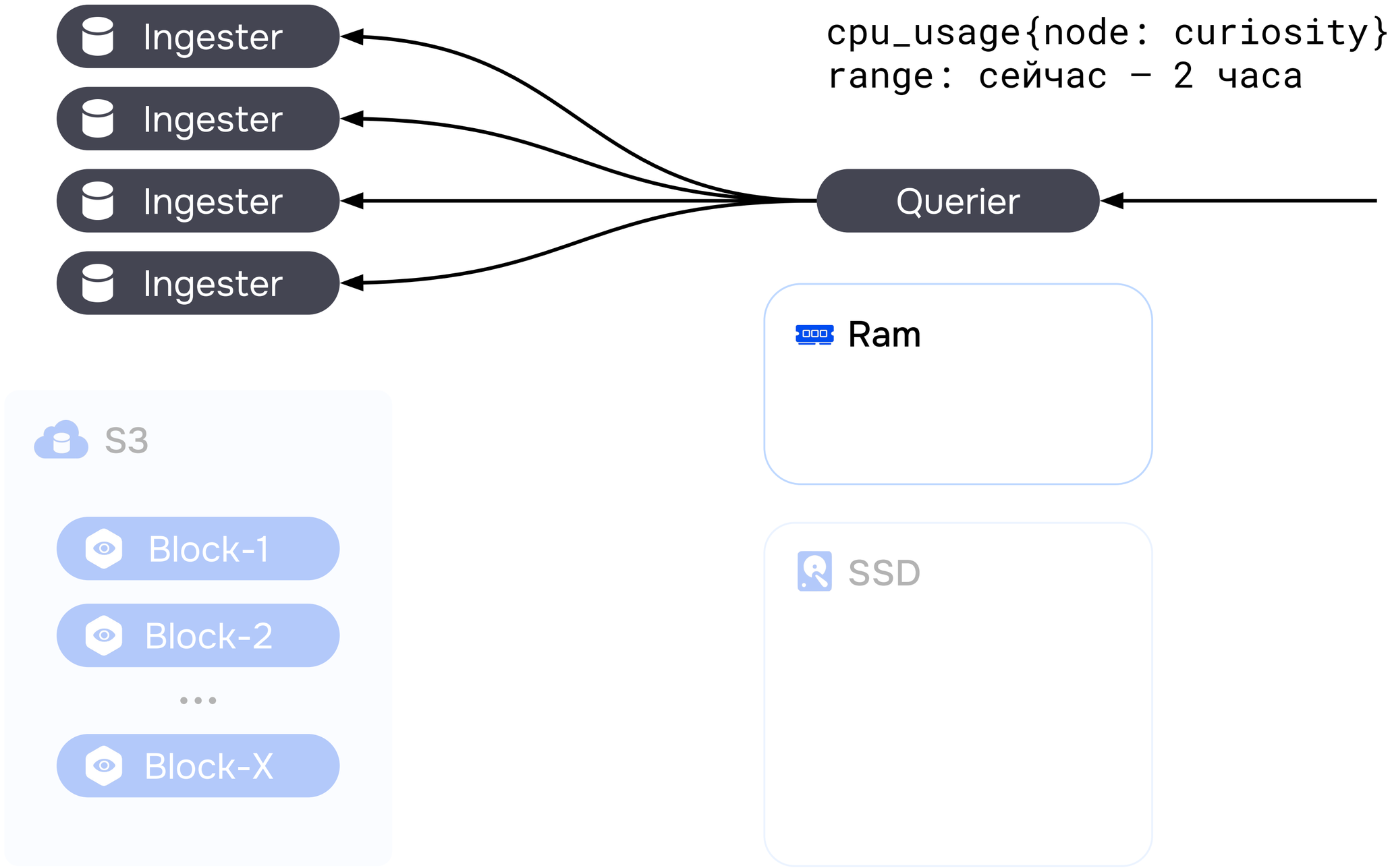


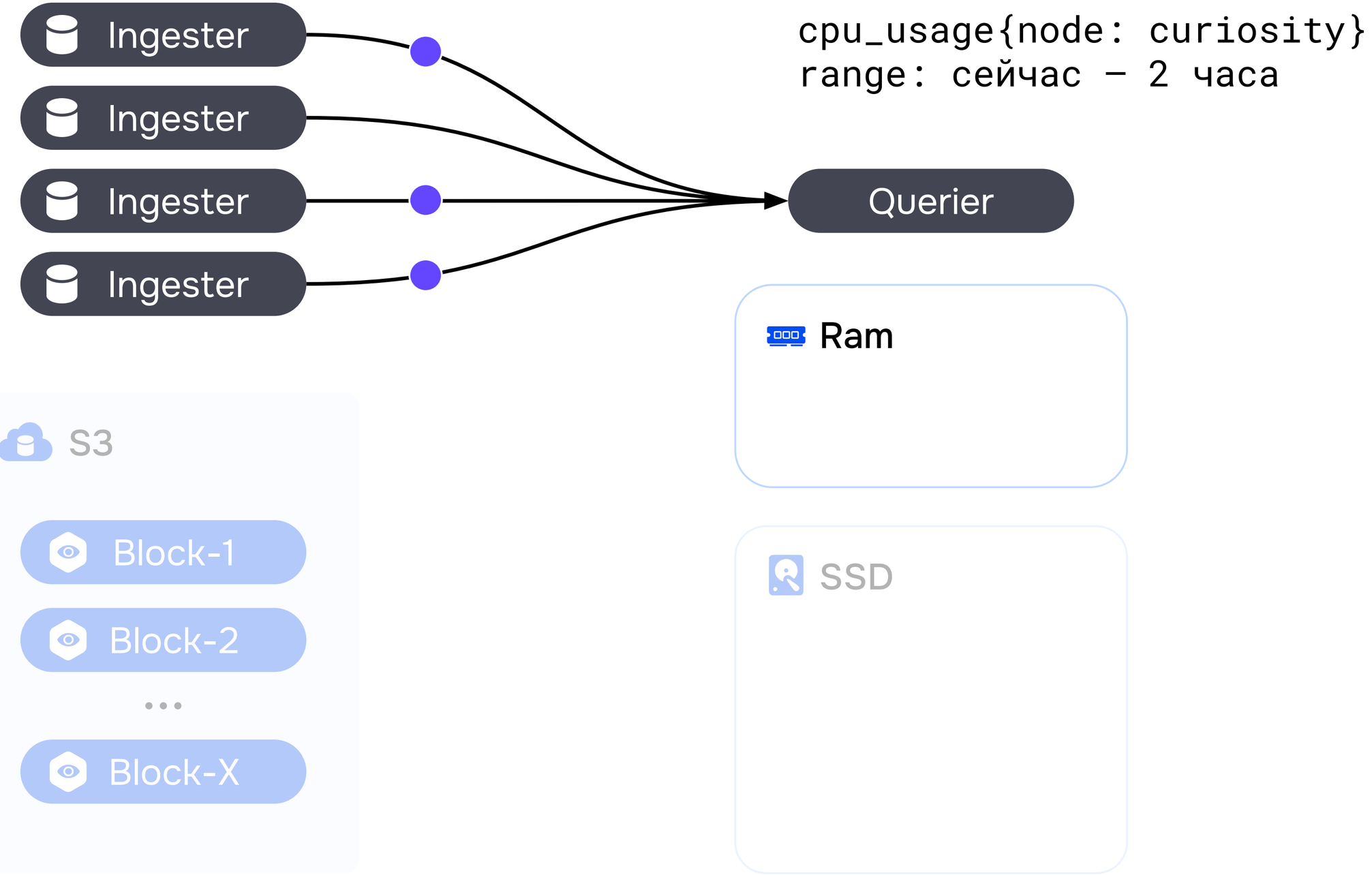
Как читать?

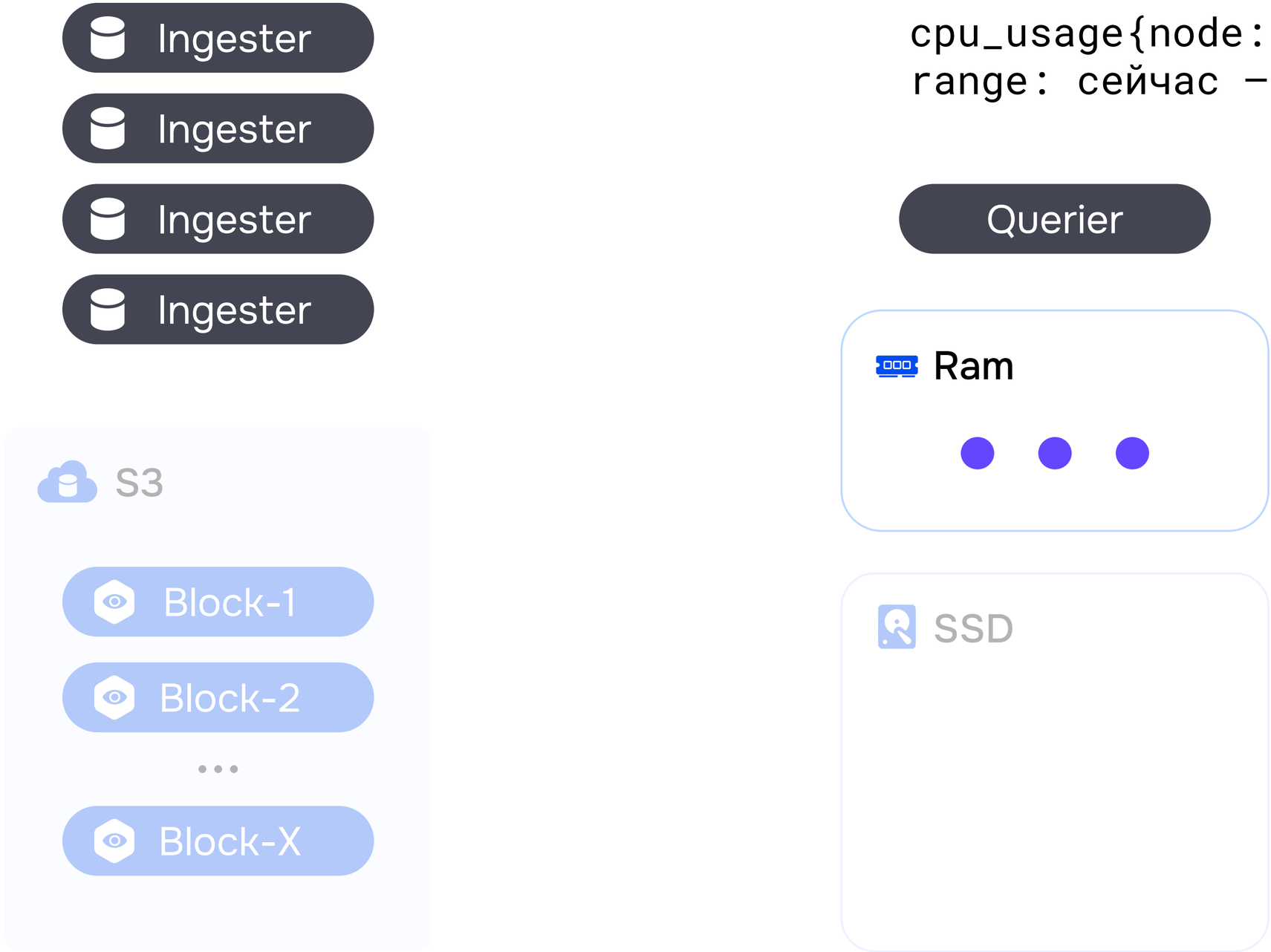

Ingester
Ingester
Ingester
Ingester
Querier
Ram
S3
Block-1
Block-2
...
Block-X
SSD

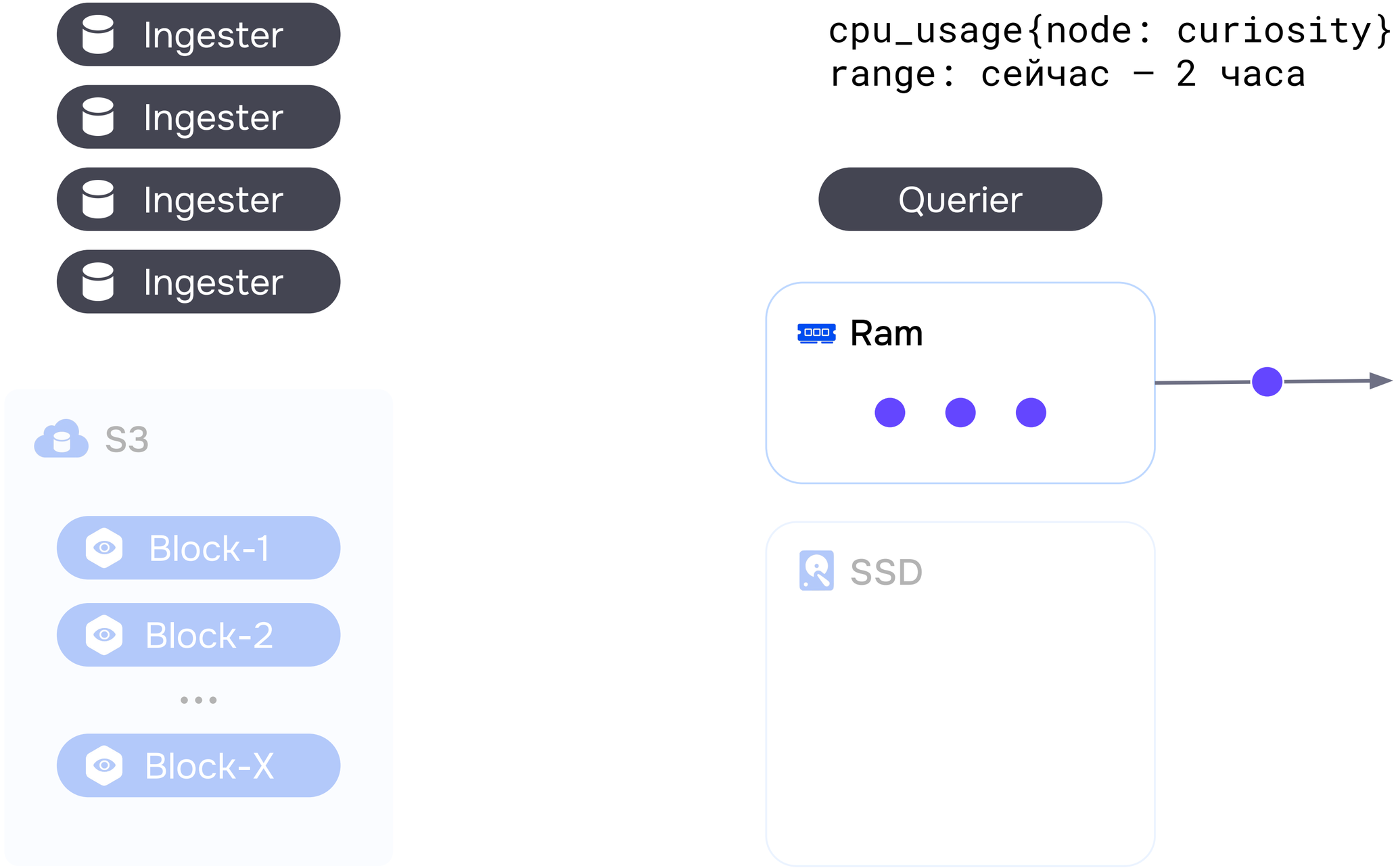

Ingester
Ingester
Ingester
Ingester
cpu_usage{node: curiosity}
range: сейчас – 2 часа
Querier
Ram
S3
Block-1
Block-2
...
Block-X
SSD

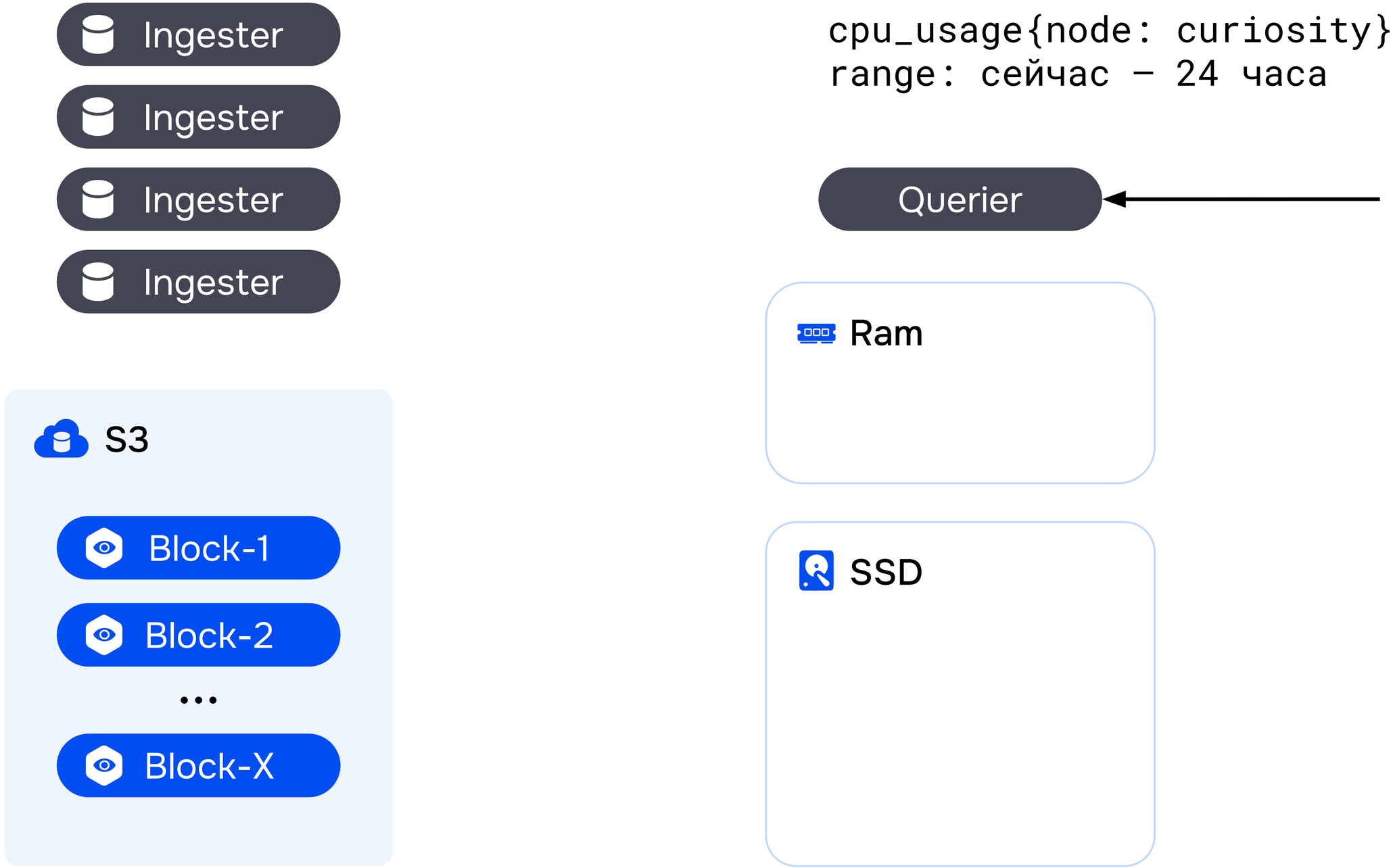

Ingester
Ingester
Ingester
Ingester
cpu_usage{node: curiosity}
range: сейчас – 2 часа
Querier
Ram
S3
Block-1
Block-2
...
Block-X
SSD

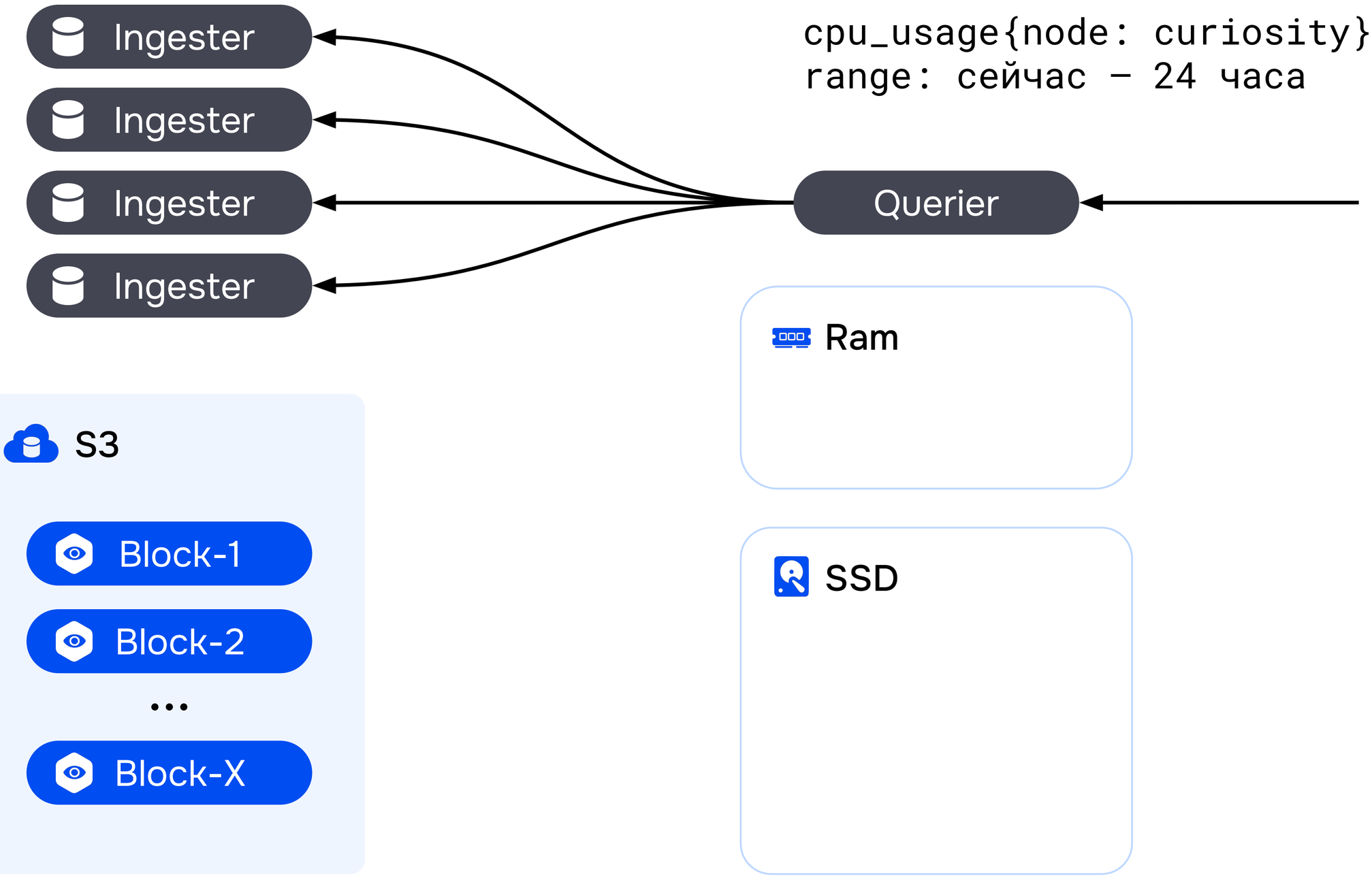

Ingester
Ingester
Ingester
Ingester
cpu_usage{node: curiosity}
range: сейчас – 2 часа
Querier
Ram
S3
Block-1
Block-2
...
Block-X
SSD

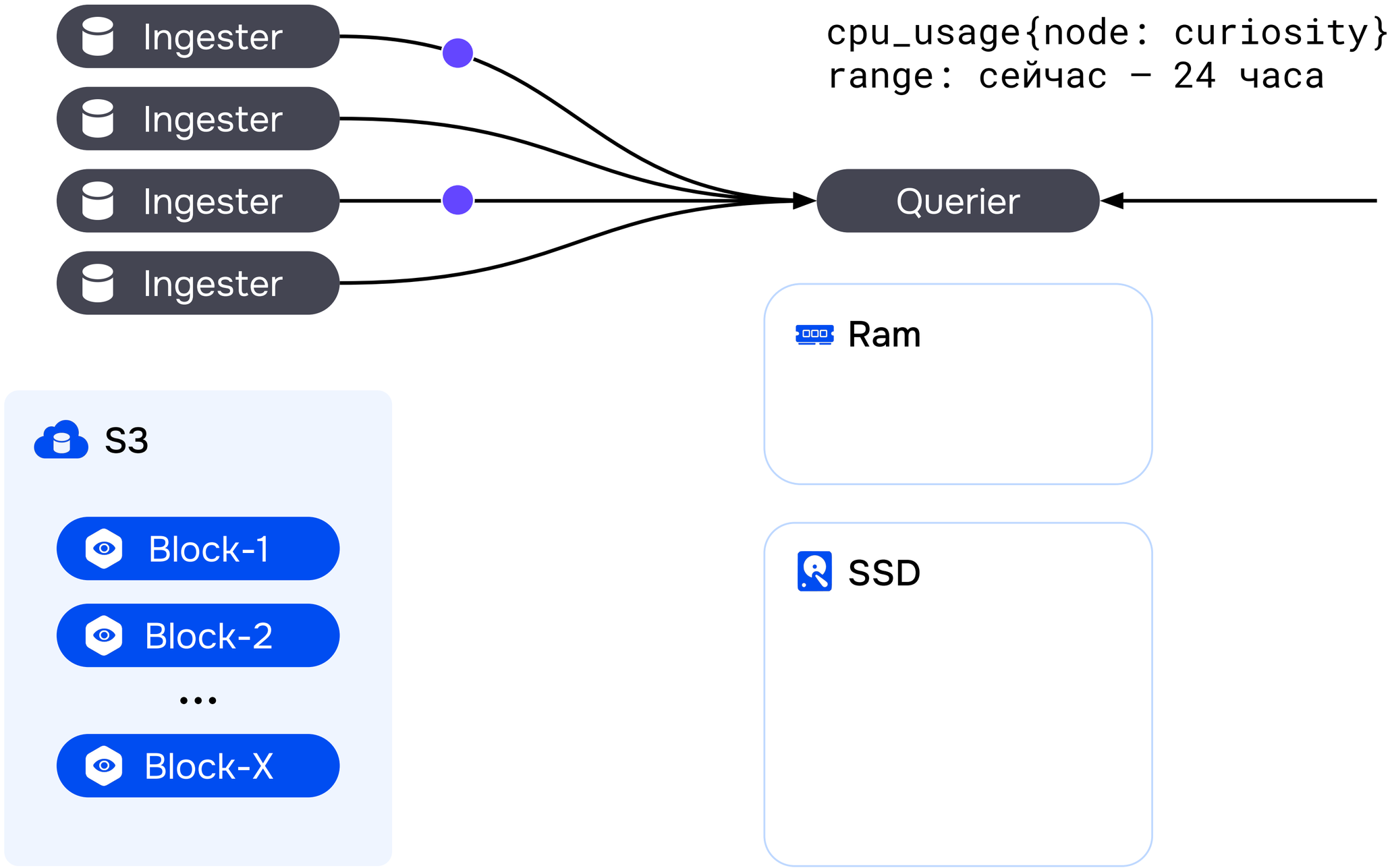

Ingester
Ingester
Ingester
Ingester
cpu_usage{node: curiosity}
range: сейчас – 2 часа
Querier
Ram
S3
Block-1
Block-2
...
Block-X
SSD

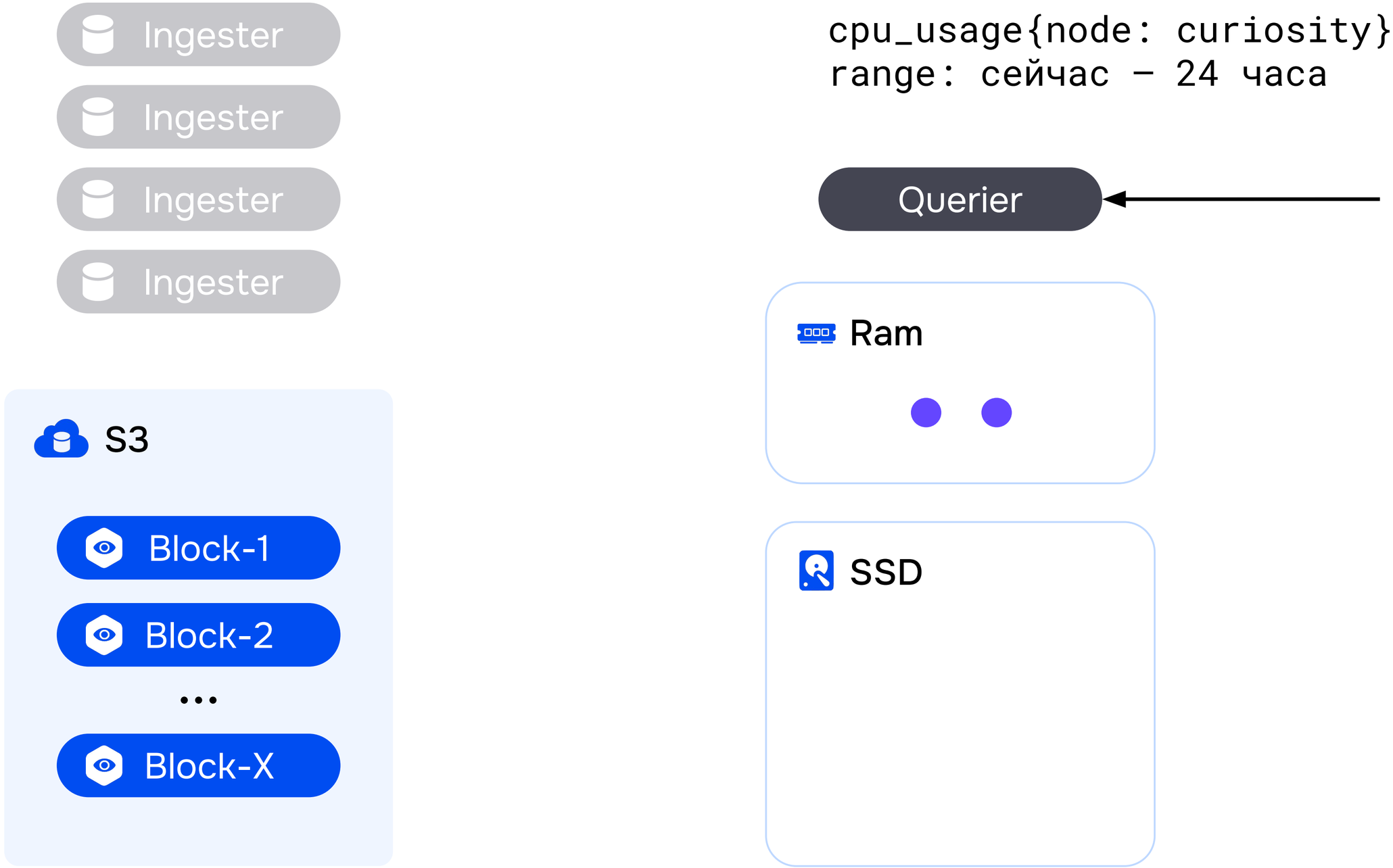

Ingester
Ingester
Ingester
Ingester
cpu_usage{node: curiosity}
range: сейчас – 2 часа
Querier
Ram
S3
Block-1
Block-2
Block-X
SSD

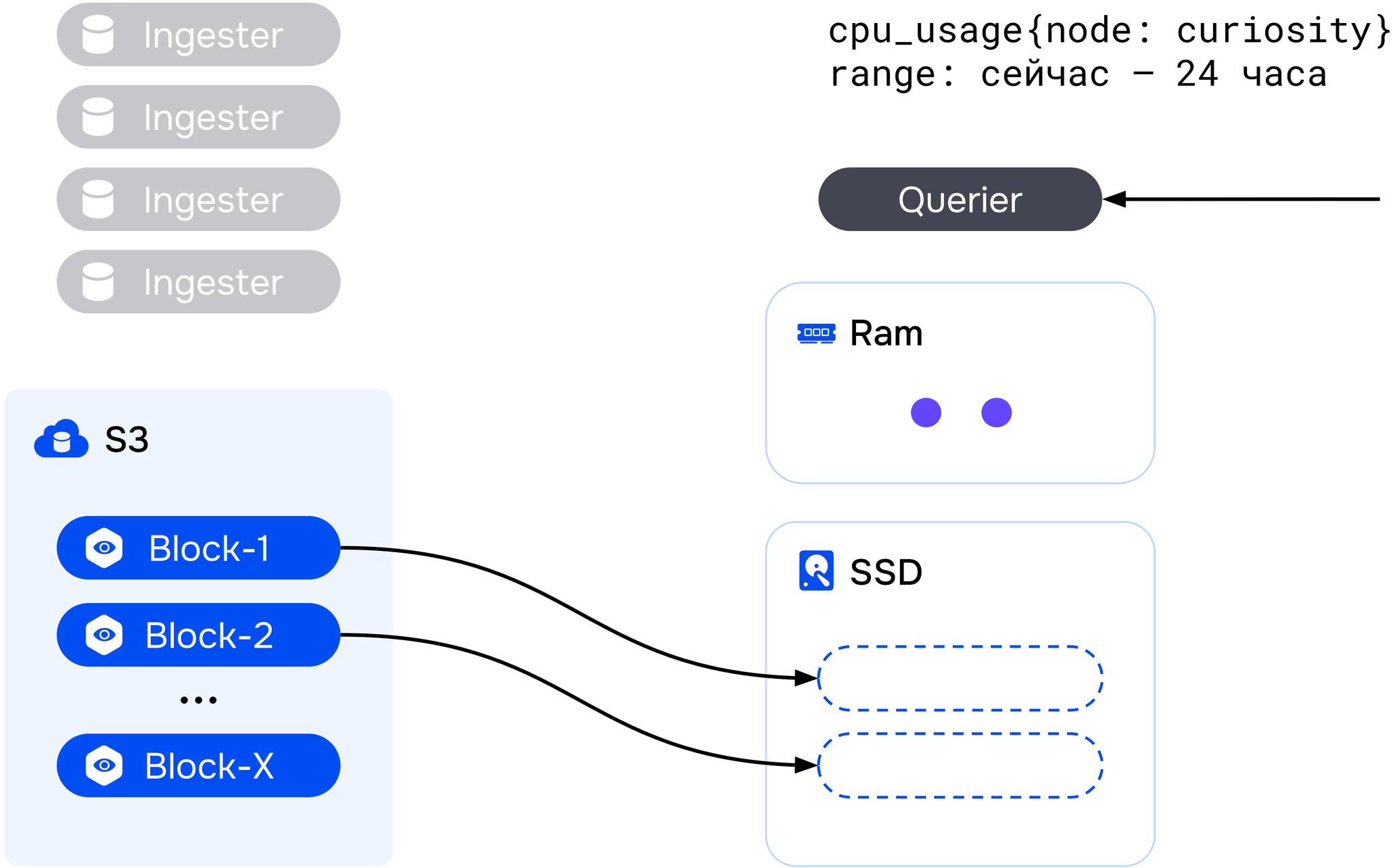

Ingester
Ingester
Ingester
Ingester
S3
Block-1
Block-2
Block-X
cpu_usage{node: curiosity}
range: сейчас – 2 часа
Querier
Ram
SSD

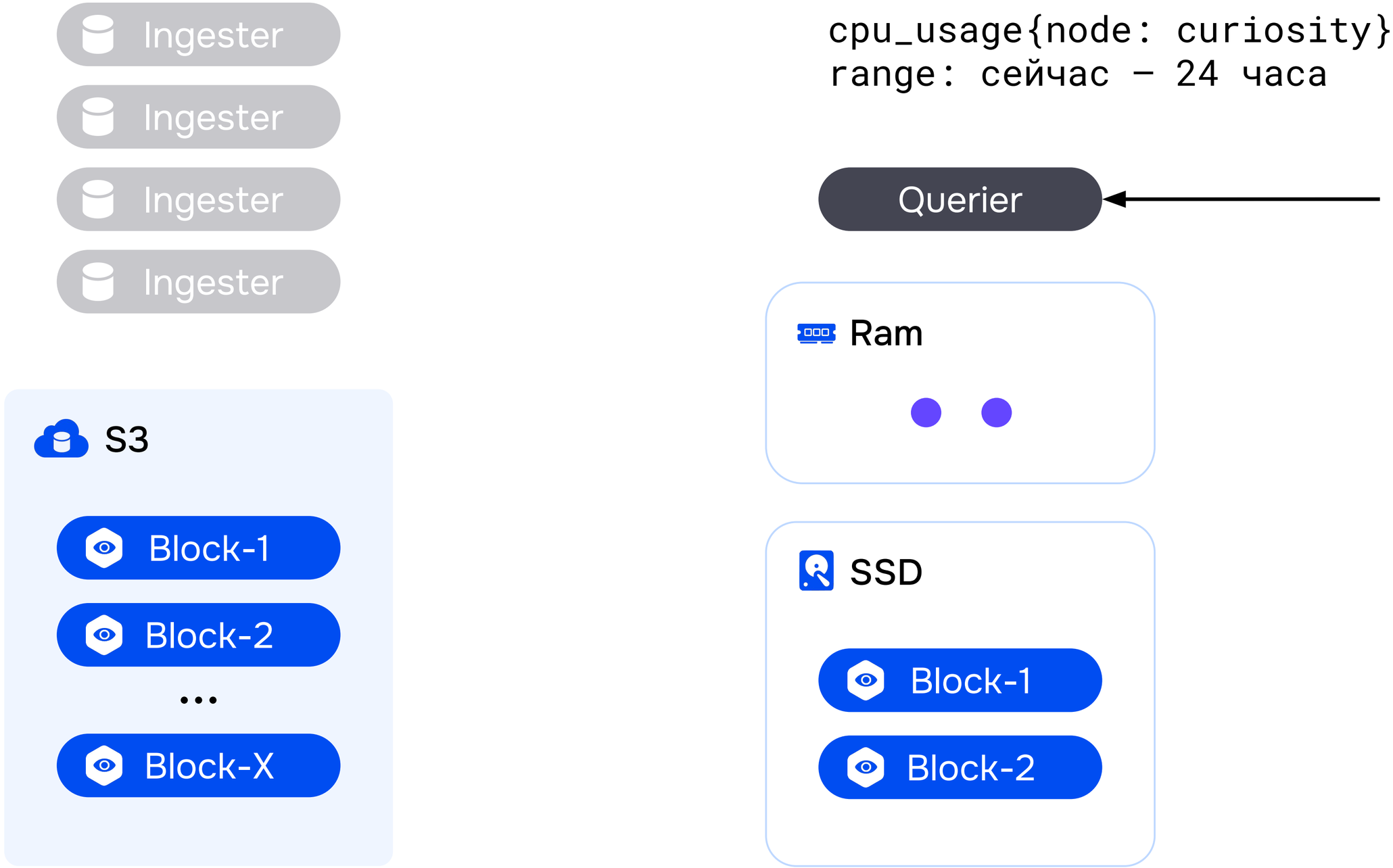

Ingester
Ingester
Ingester
Ingester
cpu_usage{node: curiosity}
range: сейчас – 24 часа
Querier
Ram
S3
Block-1
Block-2
...
Block-X
SSD

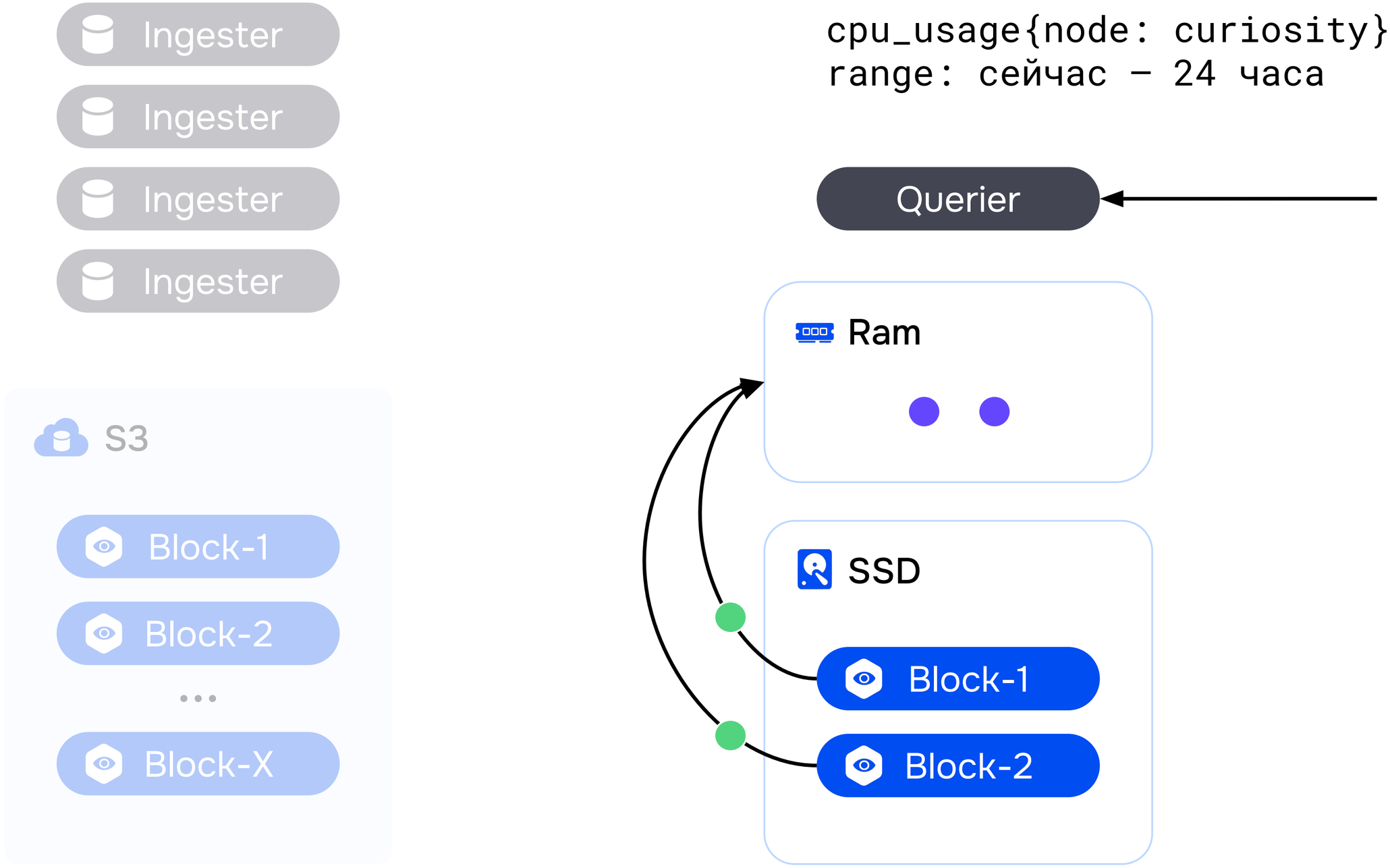

Ingester
Ingester
Ingester
Ingester
cpu_usage{node: curiosity}
range: сейчас – 24 часа
Querier
Ram
SSD
S3
Block-1
Block-2
...
Block-X

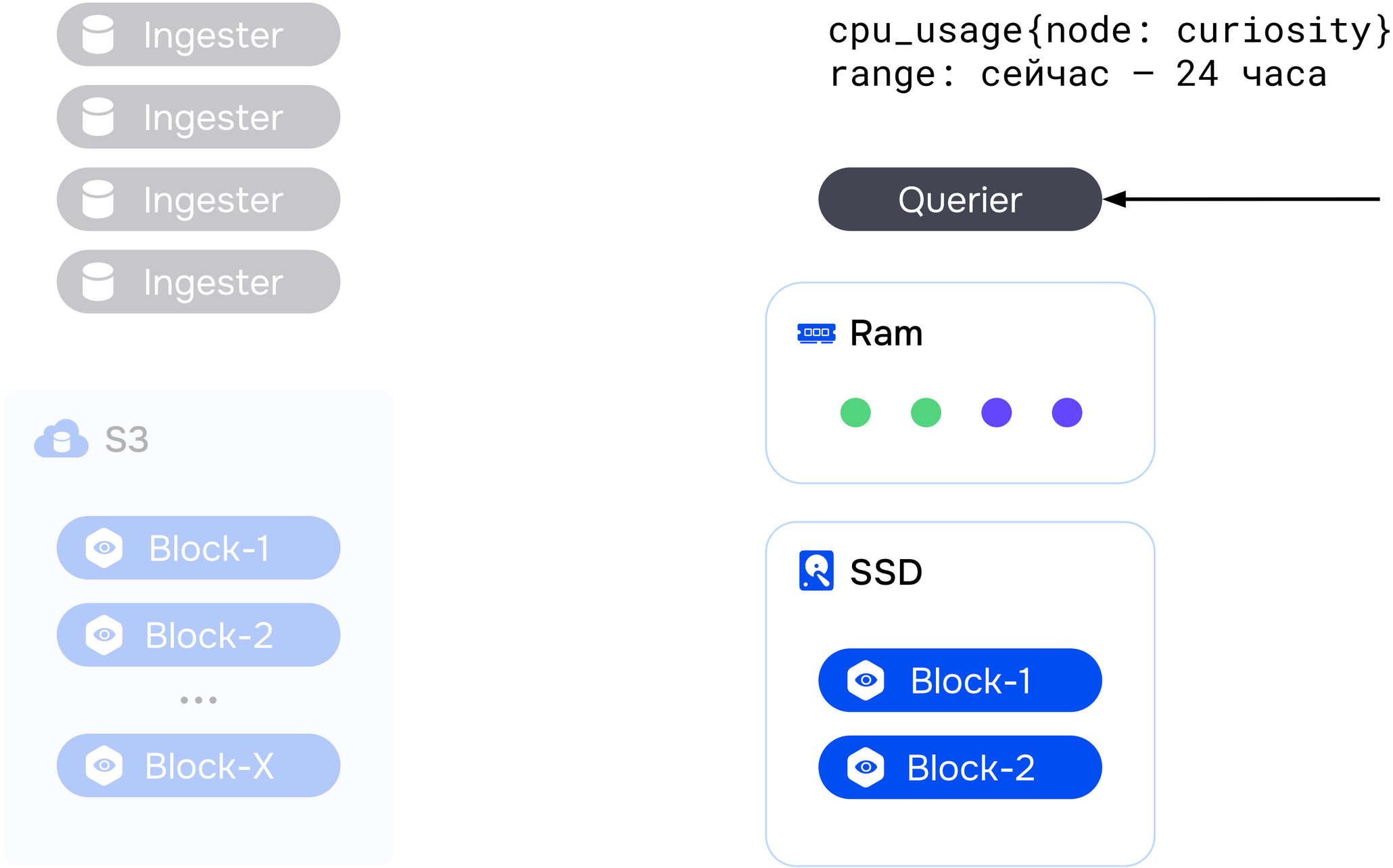

Ingester
Ingester
Ingester
Ingester
cpu_usage{node: curiosity}
range: сейчас – 24 часа
Querier
Ram
SSD
S3
Block-1
Block-2
...
Block-X

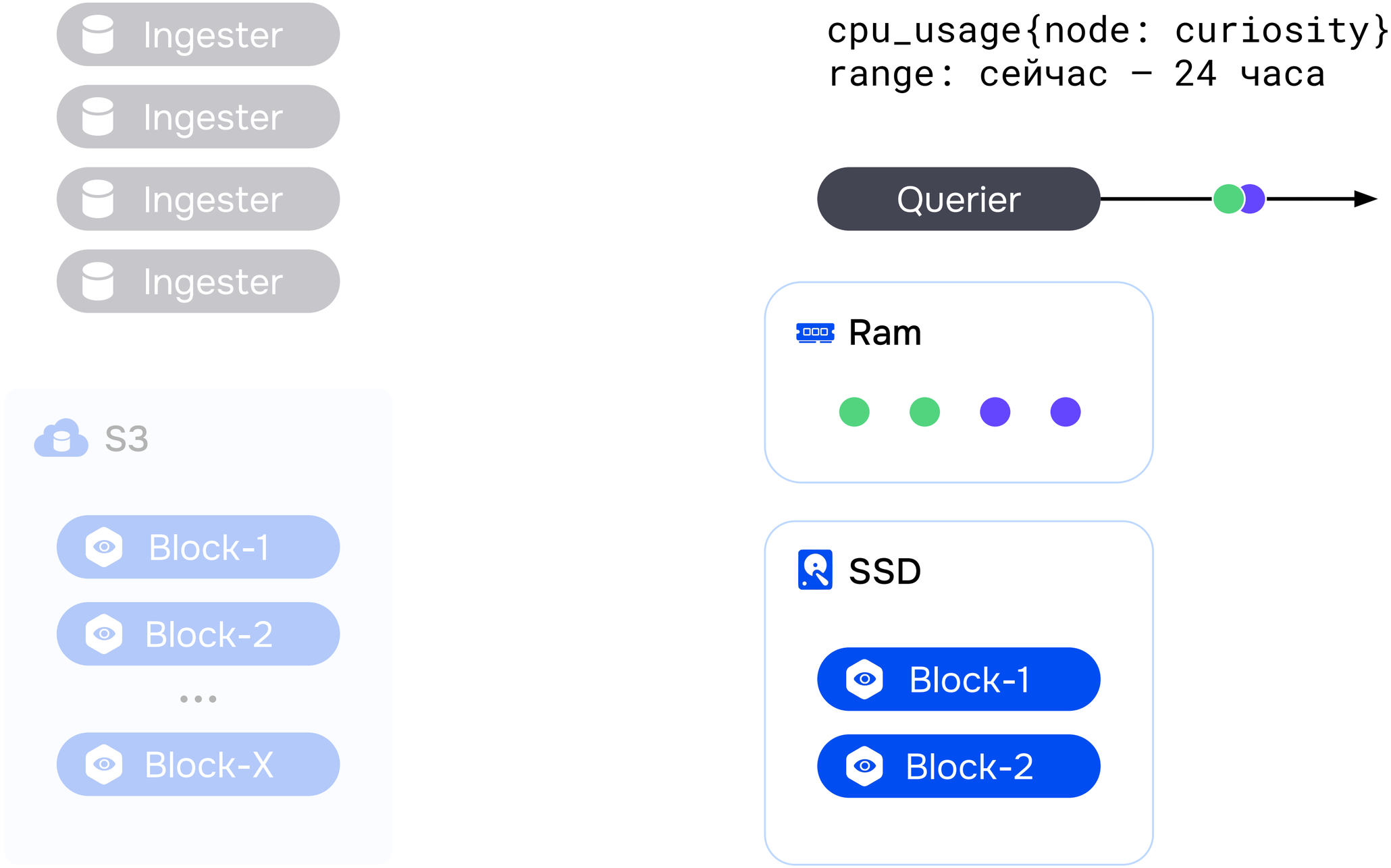

Ingester
Ingester
Ingester
Ingester
cpu_usage{node: curiosity}
range: сейчас – 24 часа
Querier
Ram
S3
Block-1
Block-2
...
Block-X
SSD


Ingester
Ingester
Ingester
Ingester
cpu_usage{node: curiosity}
range: сейчас – 24 часа
Querier
Ram
S3
Block-1
Block-2
...
Block-X
SSD

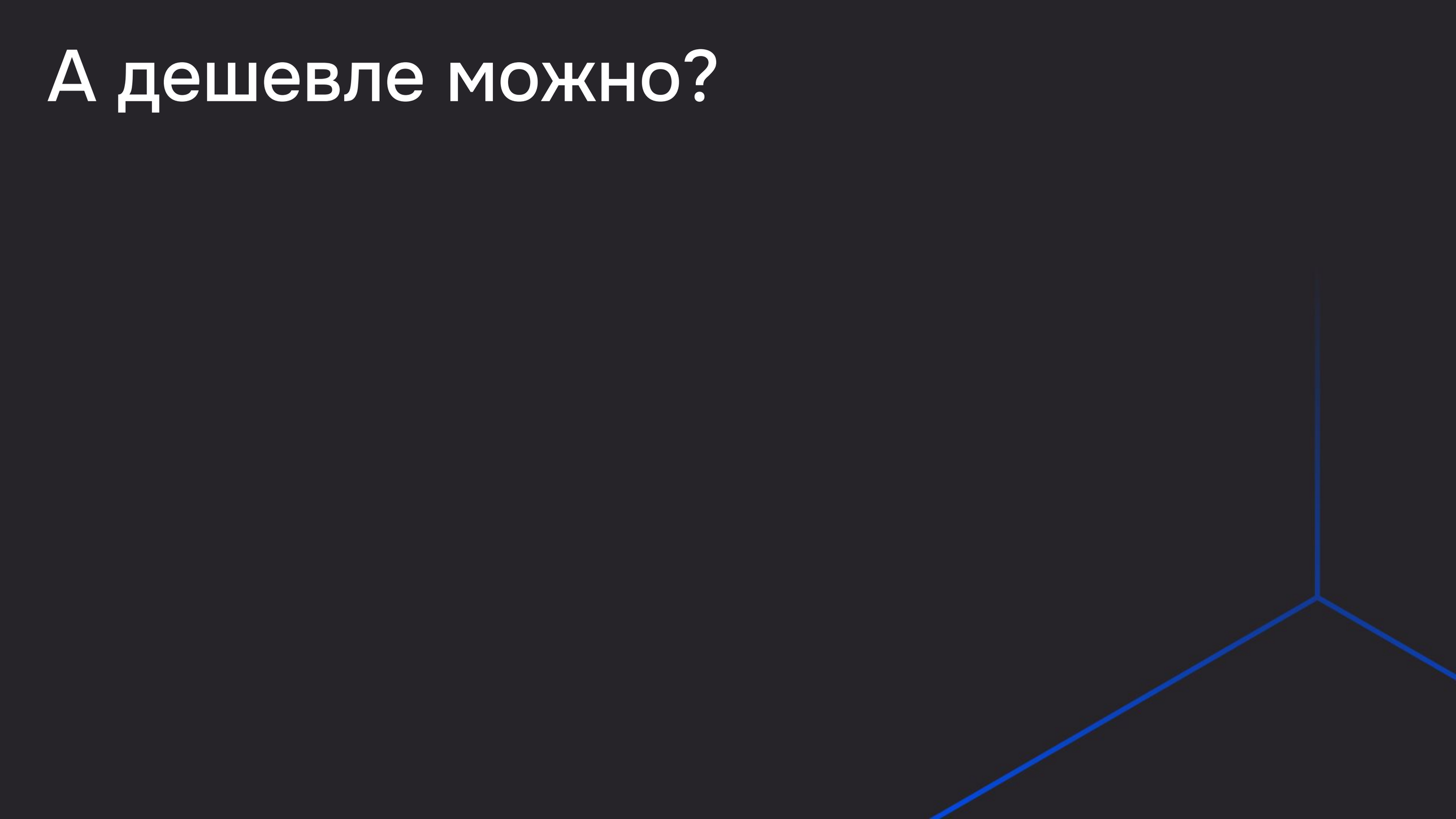

Ingester
Ingester
Ingester
Ingester
cpu_usage{node: curiosity}
range: сейчас – 24 часа
Querier
Ram
S3
Block-1
Block-2
...
Block-X
SSD
Block-1
Block-2

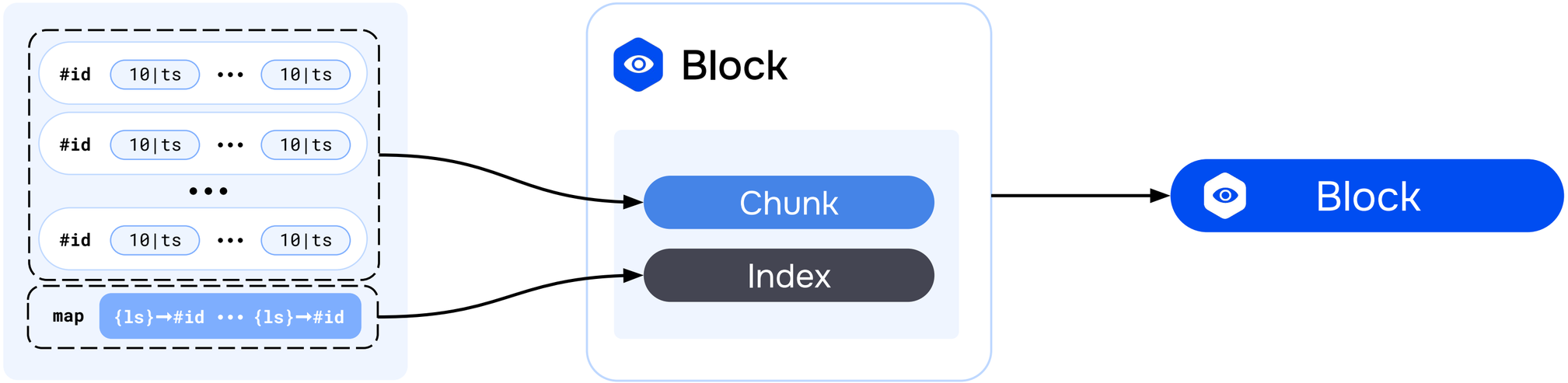

Ingester
Ingester
Ingester
Ingester
cpu_usage{node: curiosity}
range: сейчас – 24 часа
Querier
Ram
S3
Block-1
Block-2
...
Block-X
SSD
Block-1
Block-2

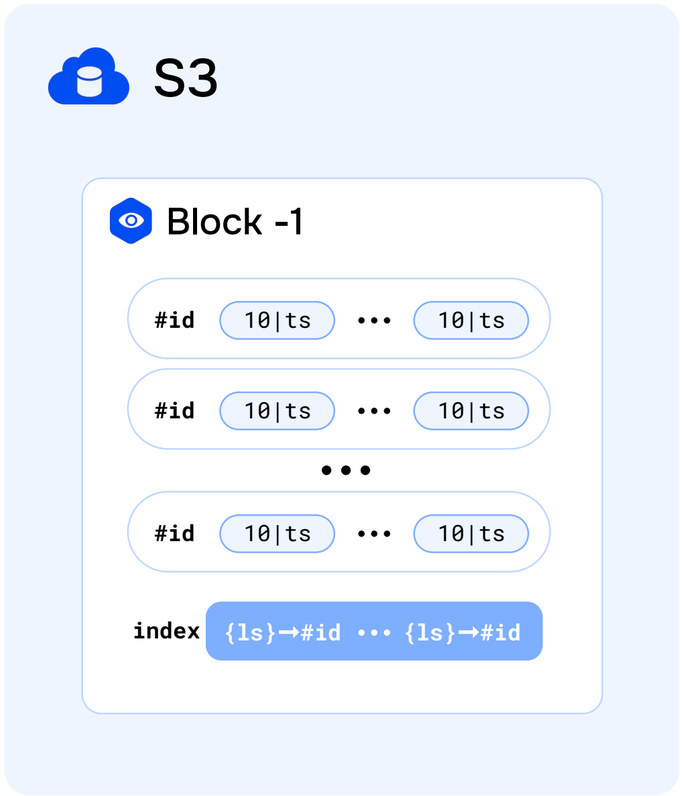

Ingester
Ingester
Ingester
Ingester
cpu_usage{node: curiosity}
range: сейчас – 24 часа
Querier
Ram
S3
Block-1
Block-2
...
Block-X
SSD
Block-1
Block-2


Ingester
Ingester
Ingester
Ingester
cpu_usage{node: curiosity}
range: сейчас – 24 часа
Querier
Ram
S3
Block-1
Block-2
...
Block-X
SSD
Block-1
Block-2

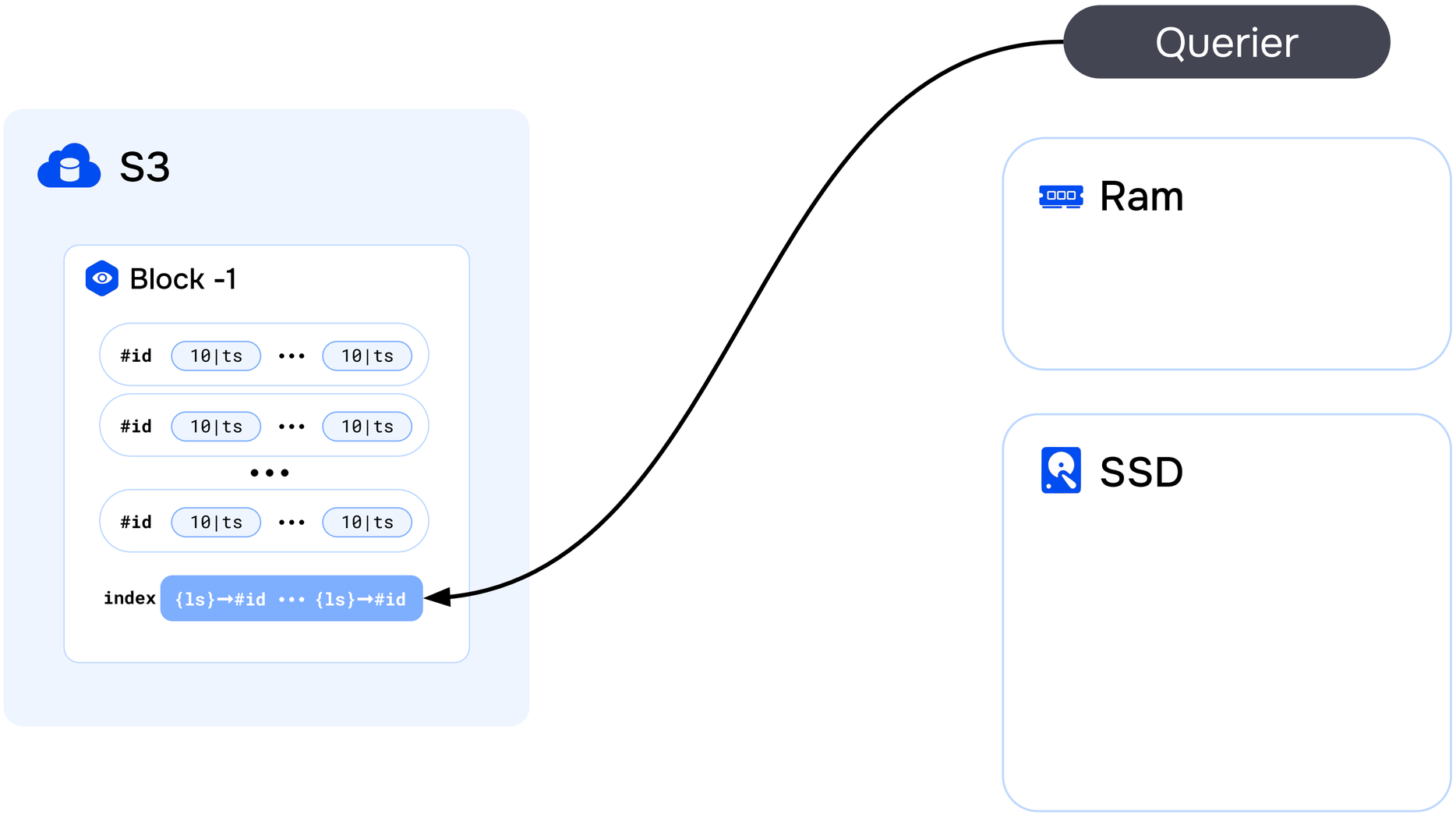
А дешевле можно?

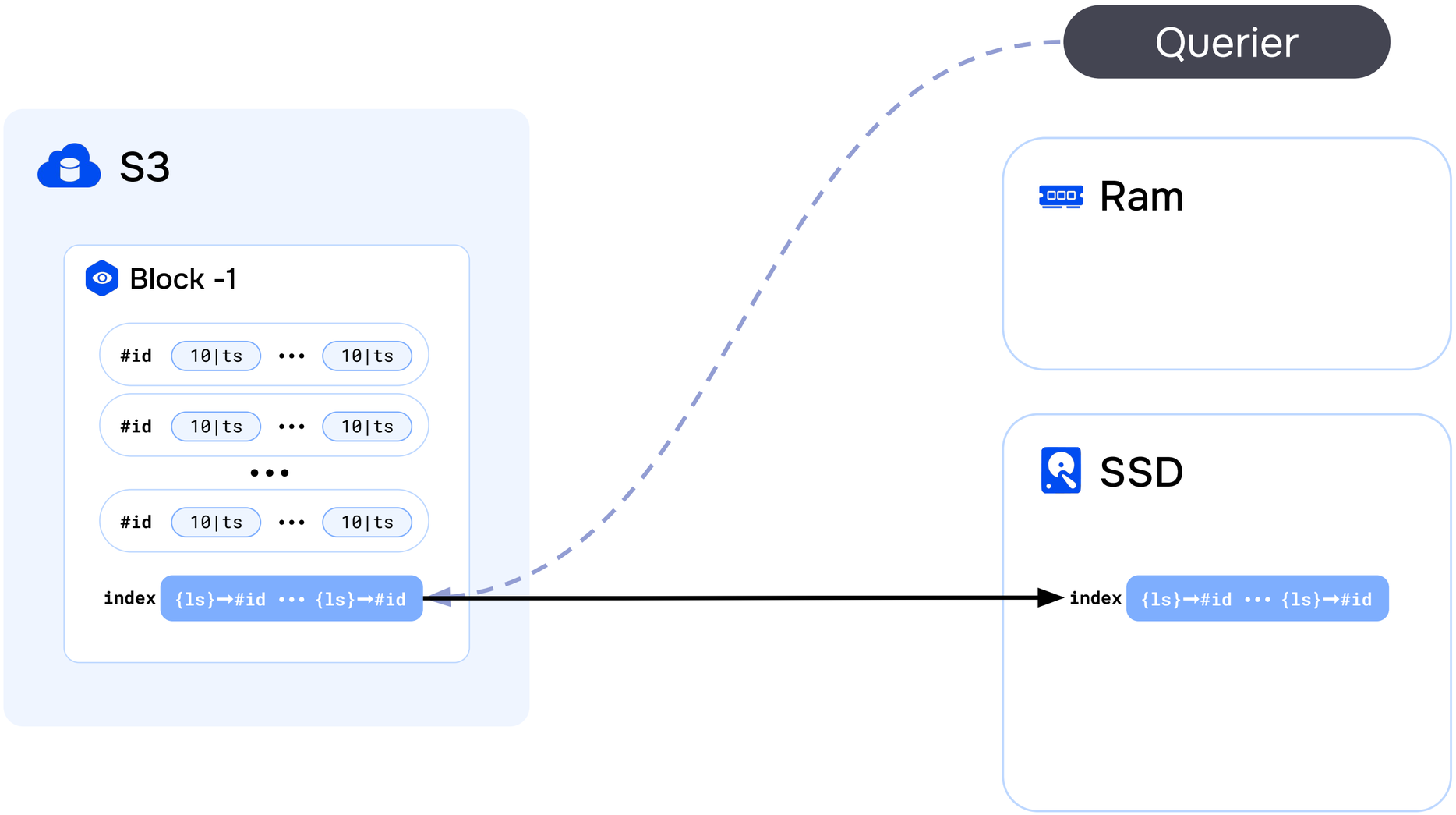


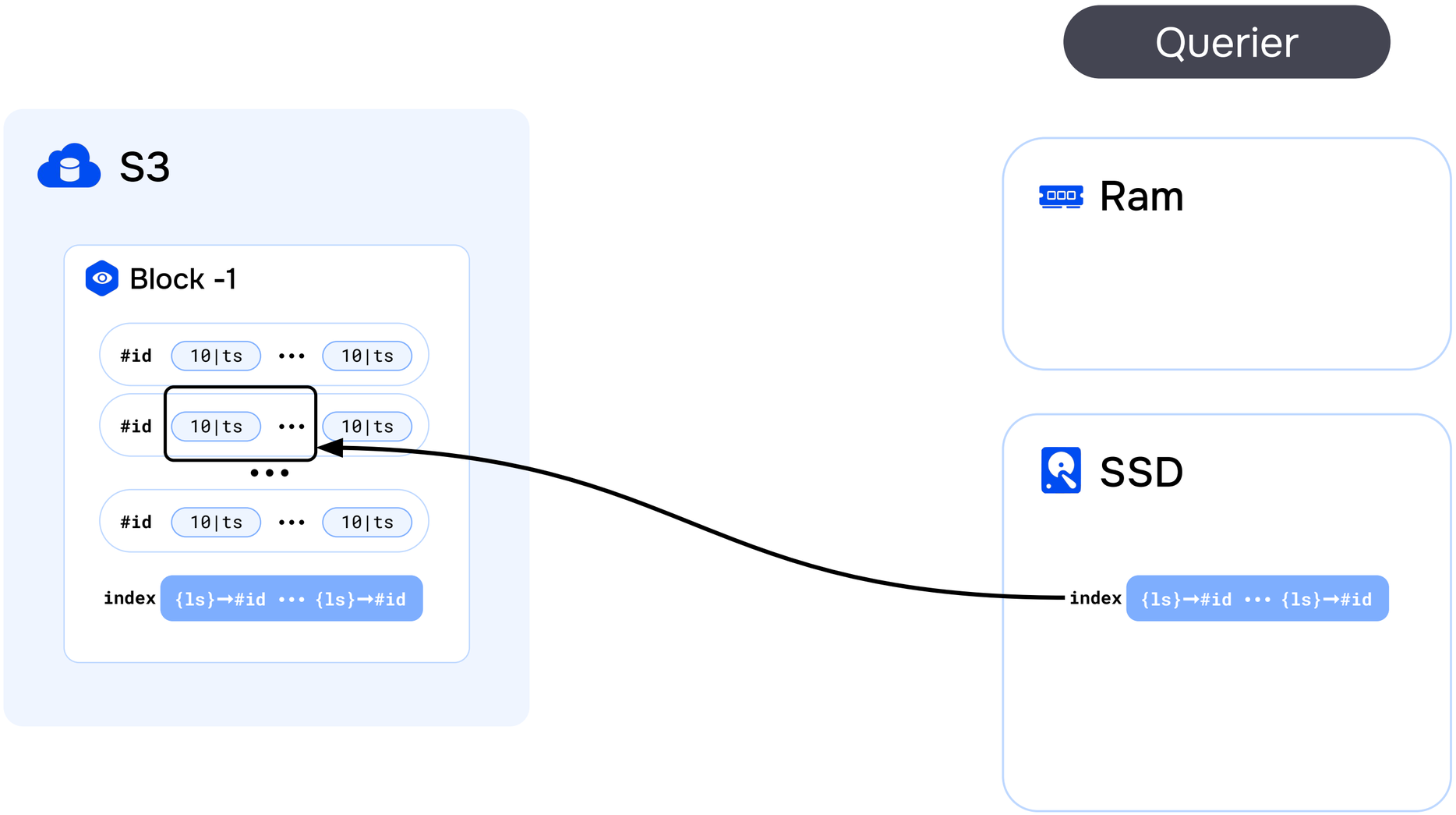

#id
10|ts
•••
10|ts
#id
10|ts
•••
10|ts
•••
#id
10|ts
•••
10|ts
map
{ls}→#id ••• {ls}→#id
Block
Chunk
Index
Block

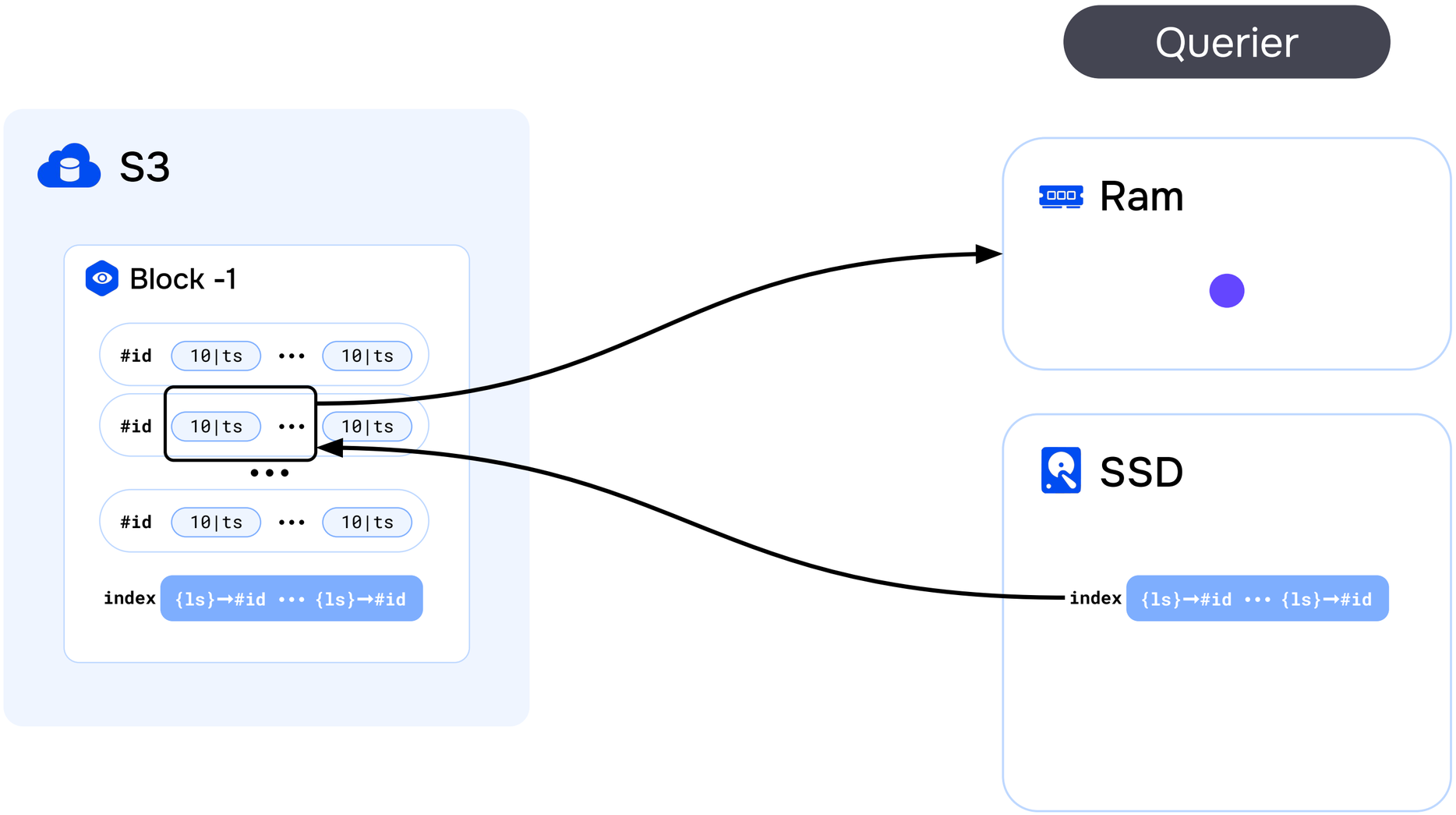

S3
Block -1
#id 10|ts ••• 10|ts
#id 10|ts ••• 10|ts
•••
#id 10|ts ••• 10|ts
index {ls}→#id ••• {ls}→#id

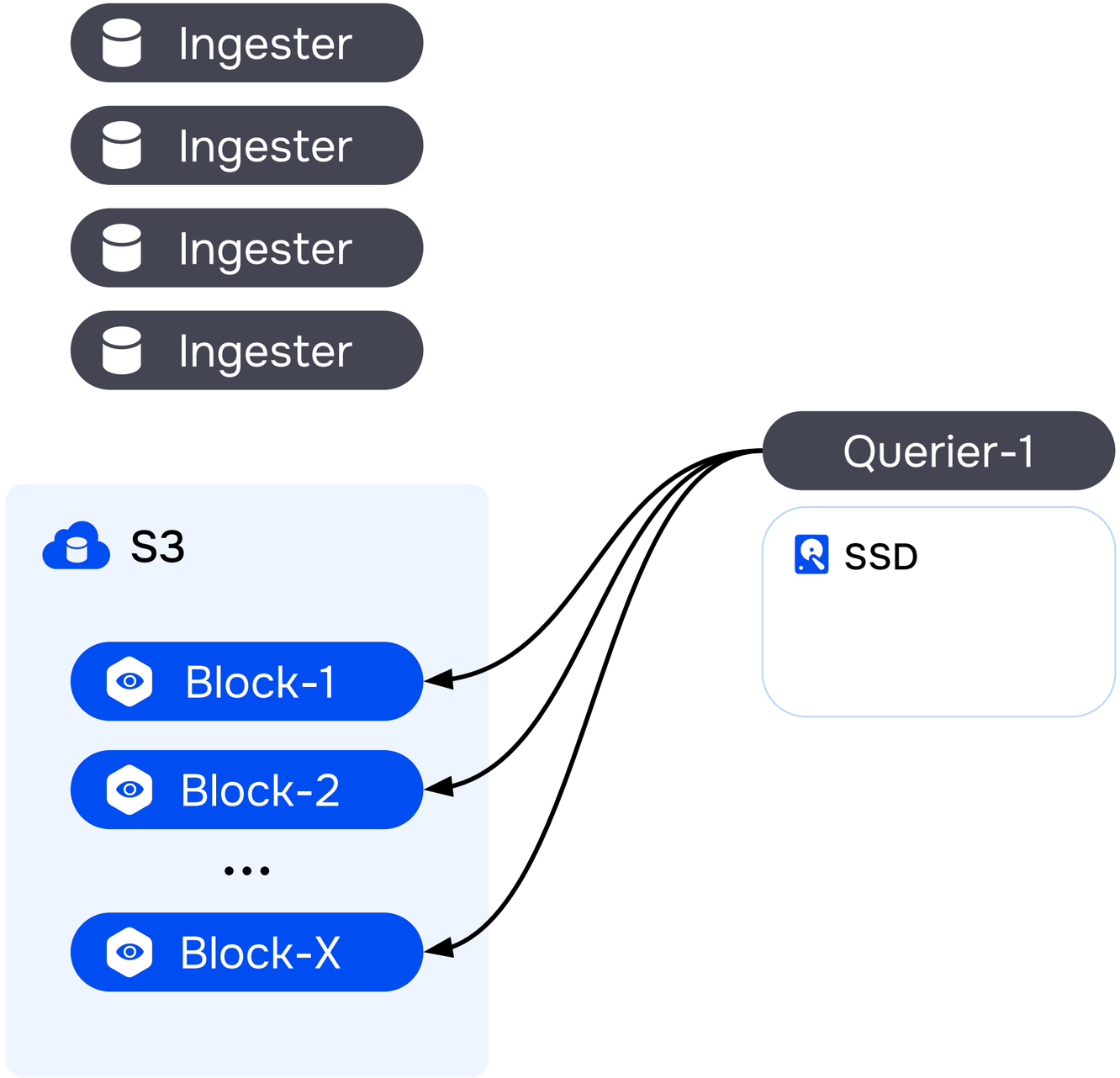

Querier
Ram
SSD

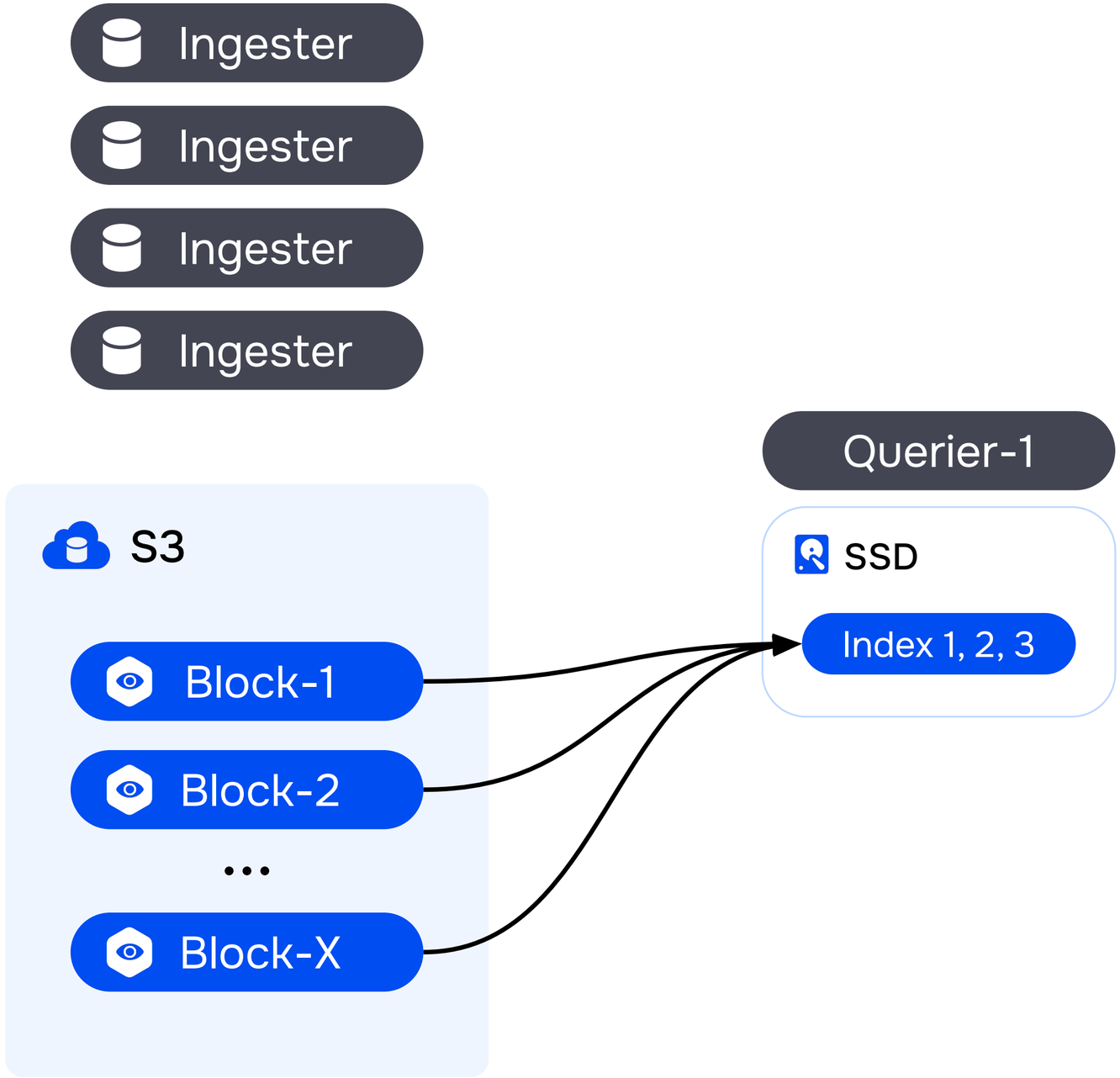

Querier
S3
Block -1
#id
10|ts
•••
10|ts
#id
10|ts
•••
10|ts
•••
#id
10|ts
•••
10|ts
index
{ls}→#id ••• {ls}→#id
Ram
SSD

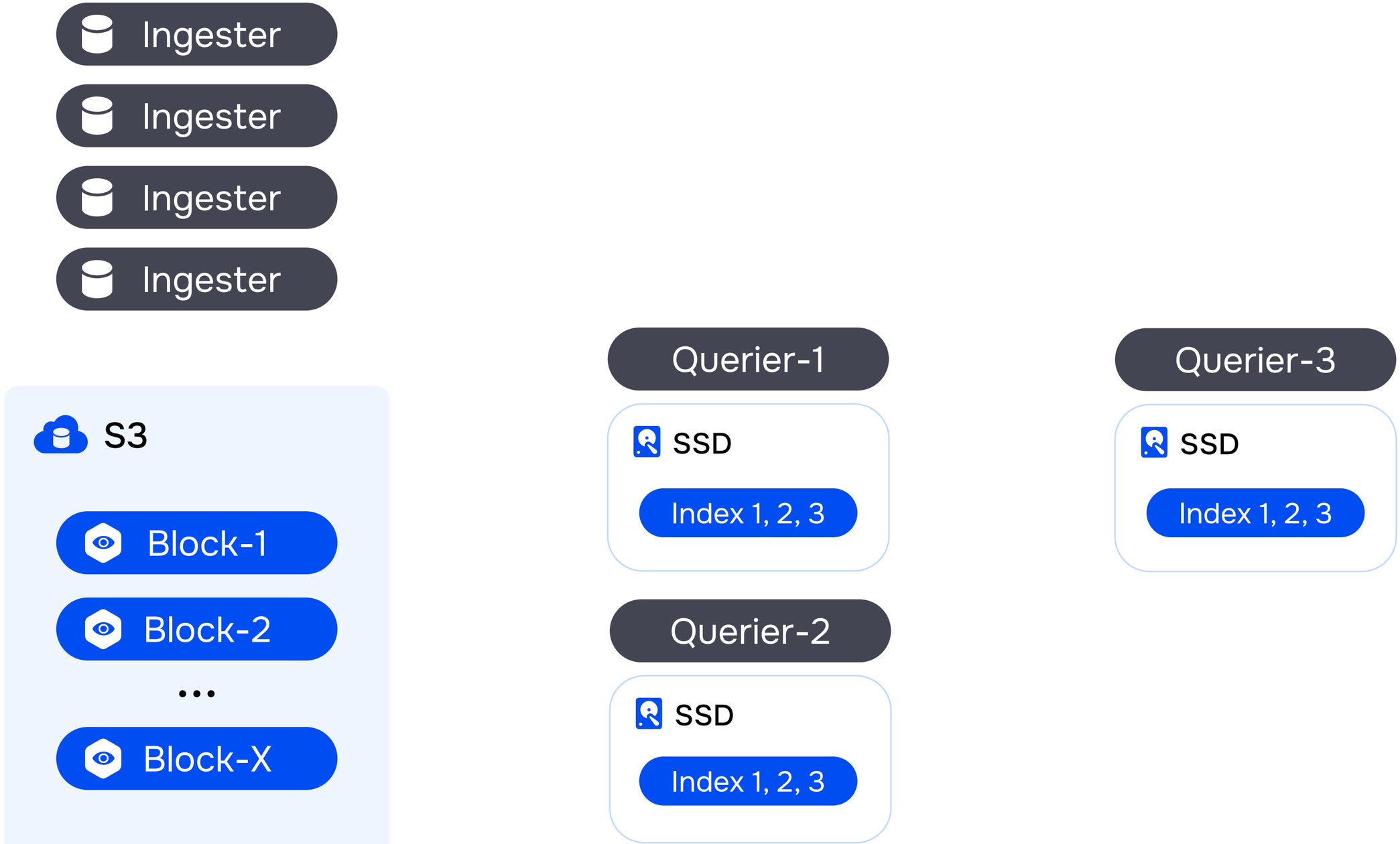

Querier
S3
Block -1
#id 10|ts ••• 10|ts
#id 10|ts ••• 10|ts
•••
#id 10|ts ••• 10|ts
index {ls}→#id ••• {ls}→#id
Ram
SSD
index {ls}→#id ••• {ls}→#id

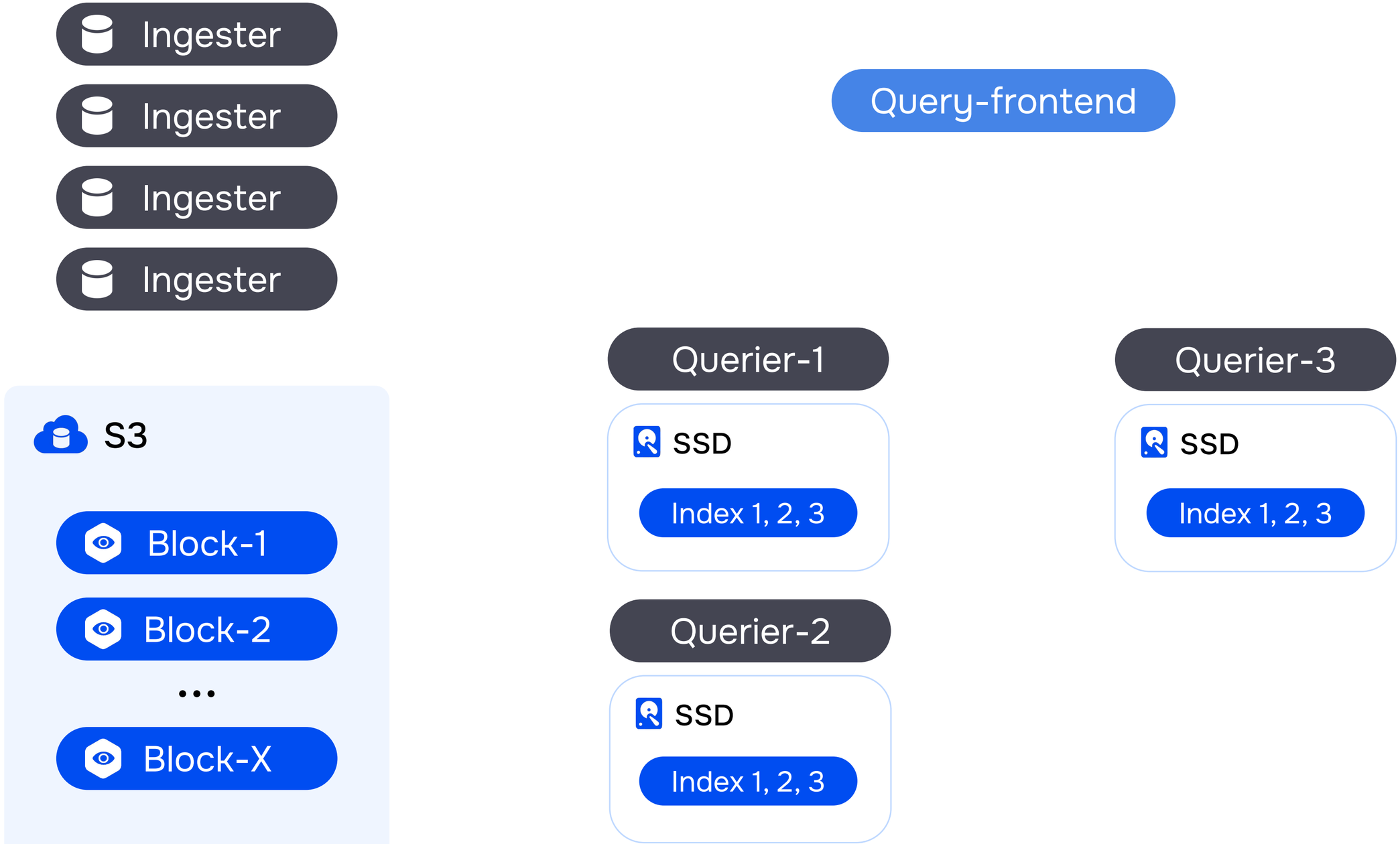

Querier
S3
Block -1
#id
10|ts
10|ts
#id
10|ts
10|ts
#id
10|ts
10|ts
index
{ls}→#id ••• {ls}→#id
Ram
SSD
index
{ls}→#id ••• {ls}→#id

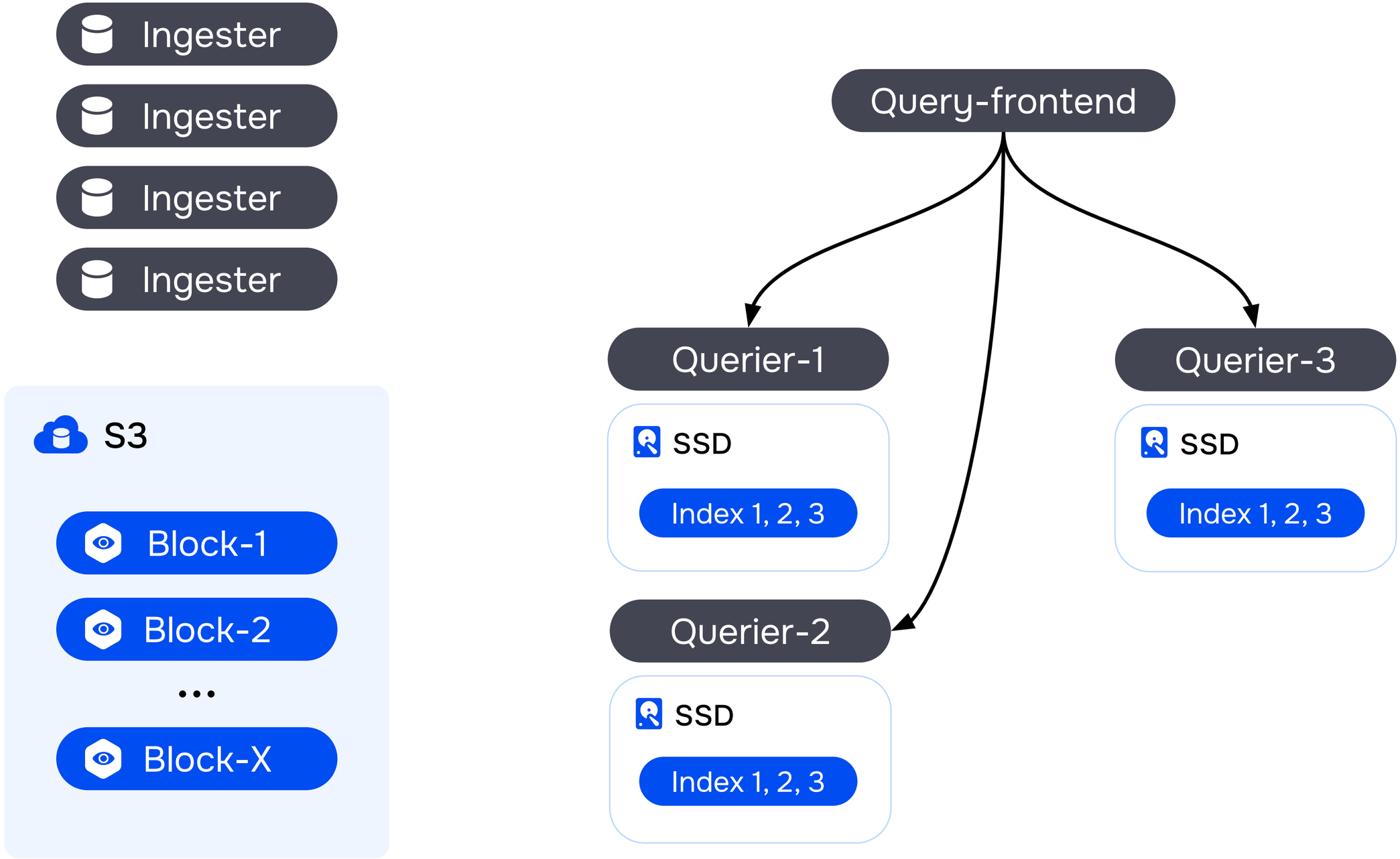

Querier
S3
Block -1
#id
10|ts
10|ts
#id
10|ts
10|ts
#id
10|ts
10|ts
index
{ls}→#id ••• {ls}→#id
Ram
SSD
index
{ls}→#id ••• {ls}→#id


Ingester
Ingester
Ingester
Ingester
Querier-1
S3
SSD
Block-1
Block-2
...
Block-X


Ingester
Ingester
Ingester
Ingester
Querier-1
S3
SSD
Index 1, 2, 3
Block-1
Block-2
...
Block-X

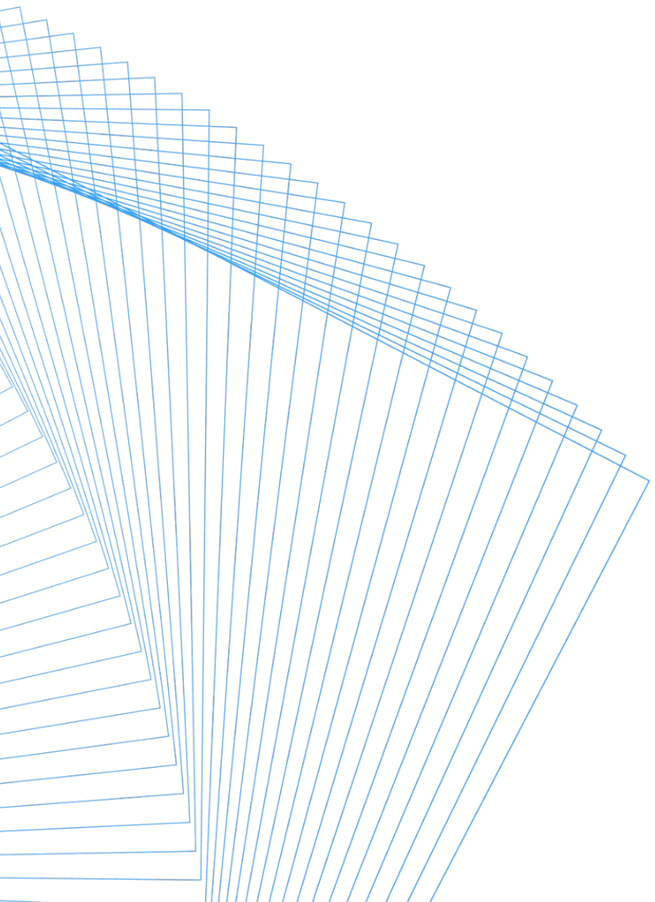

Ingester
Ingester
Ingester
Ingester
S3
Block-1
Block-2
...
Block-X
Querier-1
SSD
Index 1, 2, 3
Querier-2
SSD
Index 1, 2, 3
Querier-3
SSD
Index 1, 2, 3


Ingester
Ingester
Ingester
Ingester
Query-frontend
Querier-1
Querier-3
S3
SSD
Index 1, 2, 3
SSD
Index 1, 2, 3
Block-1
Block-2
...
Block-X
Querier-2
SSD
Index 1, 2, 3

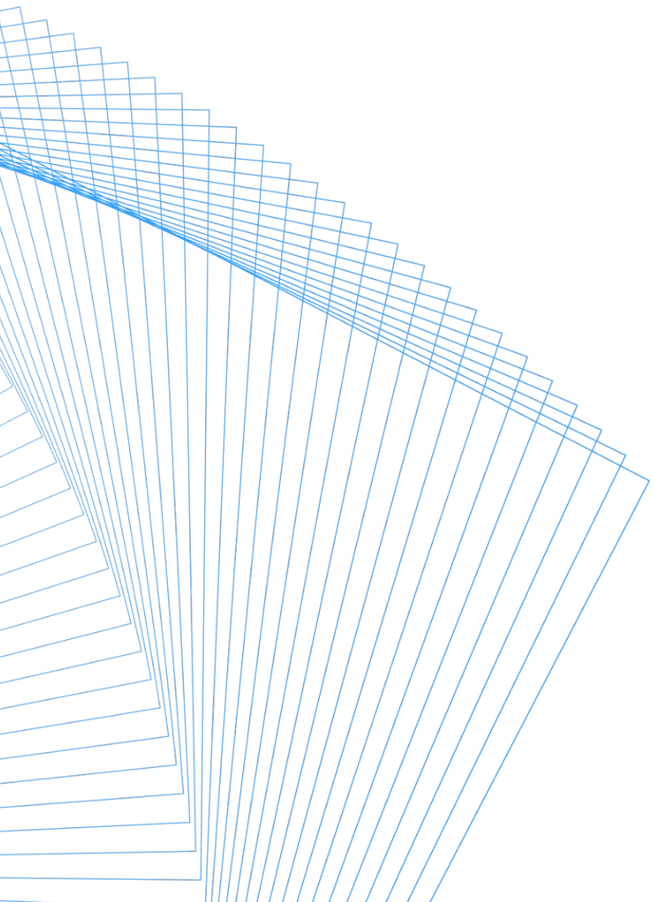

Ingester
Ingester
Ingester
Ingester
S3
Block-1
Block-2
...
Block-X
Query-frontend
Querier-1
SSD
Index 1, 2, 3
Querier-3
SSD
Index 1, 2, 3
Querier-2
SSD
Index 1, 2, 3

Централизованная система мониторинга

- Отказоустойчивость на запись: WC = RF/2 + 1
- Отказоустойчивость на чтение: RC = RF - WC + 1
- Шардирование на запись на основе хеш-ринга
- Исторические данные хранятся в S3
- При запуске читаем из S3 только индекс

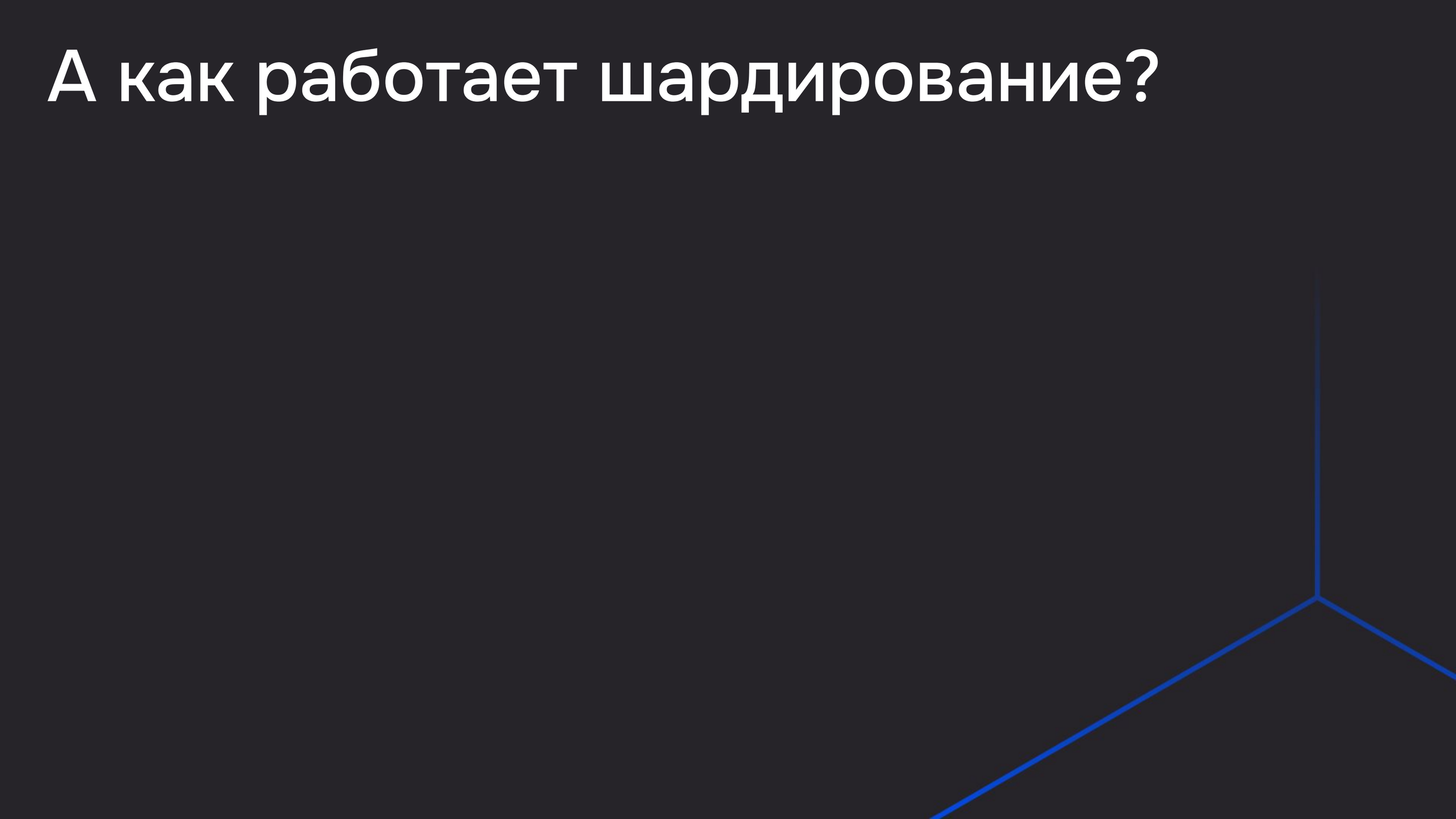


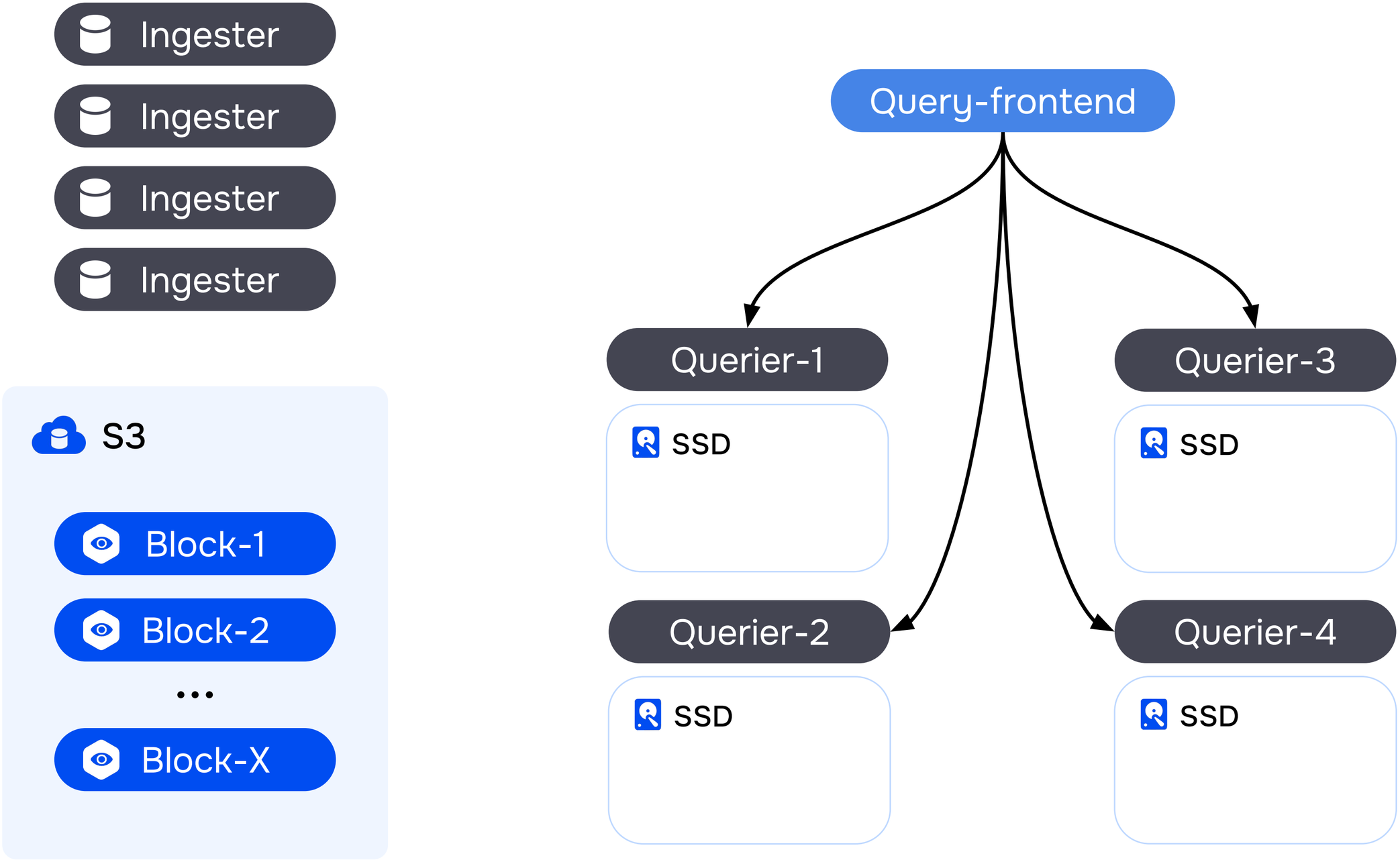
Централизованная система мониторинга



- Отказоустойчивость на запись: WC = RF/2 + 1
- Отказоустойчивость на чтение: RC = RF - WC + 1
- Шардирование на запись на основе хеш-ринга
- Исторические данные хранятся в S3
- При запуске читаем из S3 только индекс
- Читаем из чанков только нужные данные

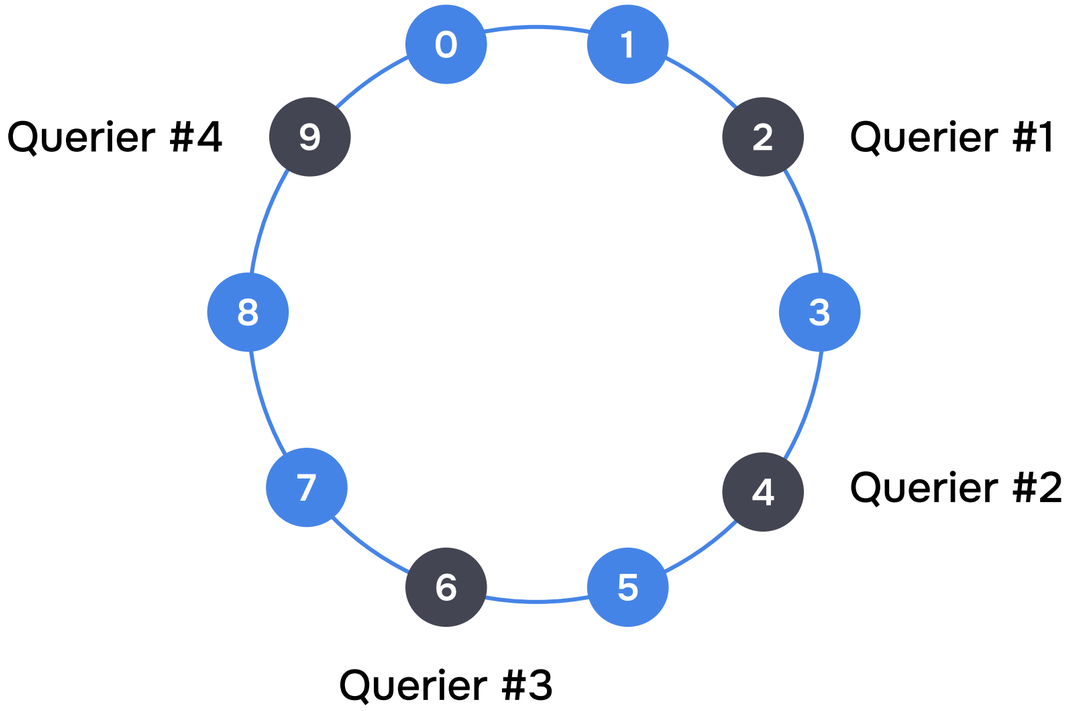
А как работает шардирование?




Ingester
Ingester
Ingester
Ingester
S3
Block-1
Block-2
...
Block-X
Query-frontend
Querier-1
SSD
Querier-3
SSD
Querier-2
SSD
Querier-4
SSD

Хеш-ринг

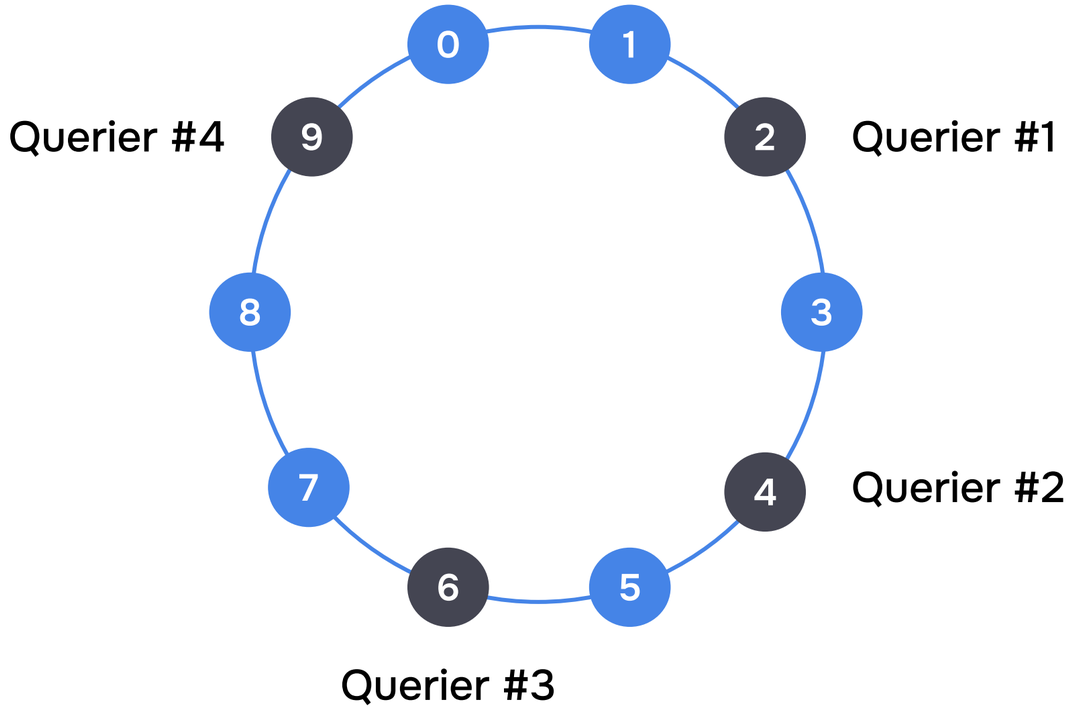
Хеш-ринг



Хеш-ринг

Block

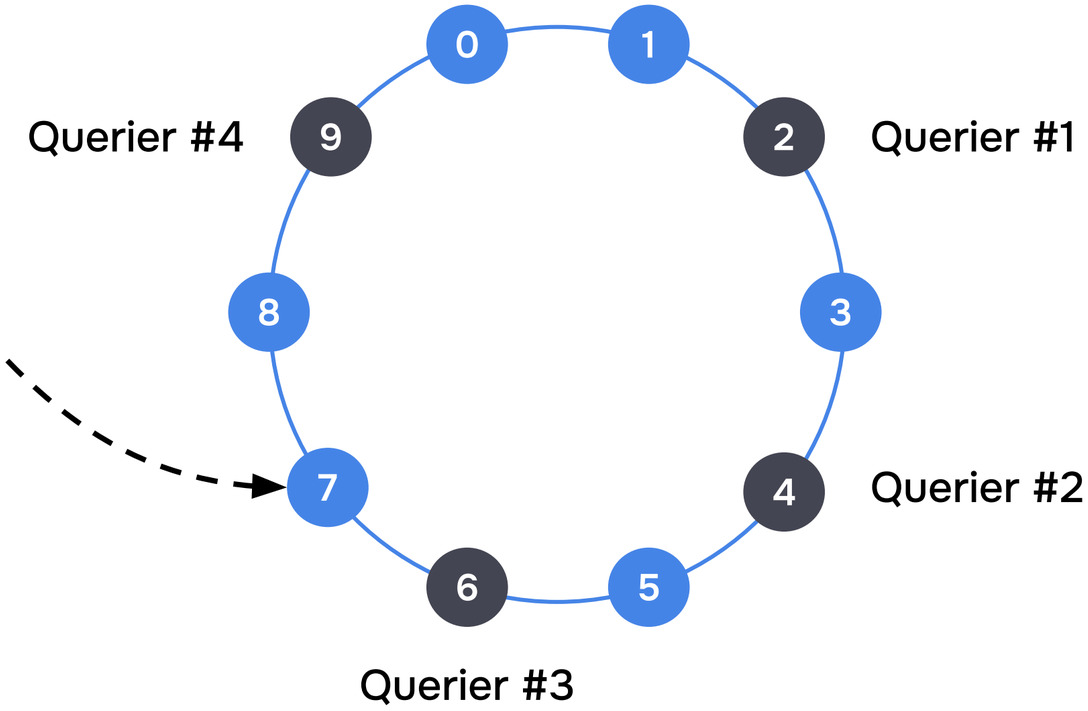
UID: 01HZN9V7YZSC3F2Q0X8D0DYRZC



Хеш-ринг

Block

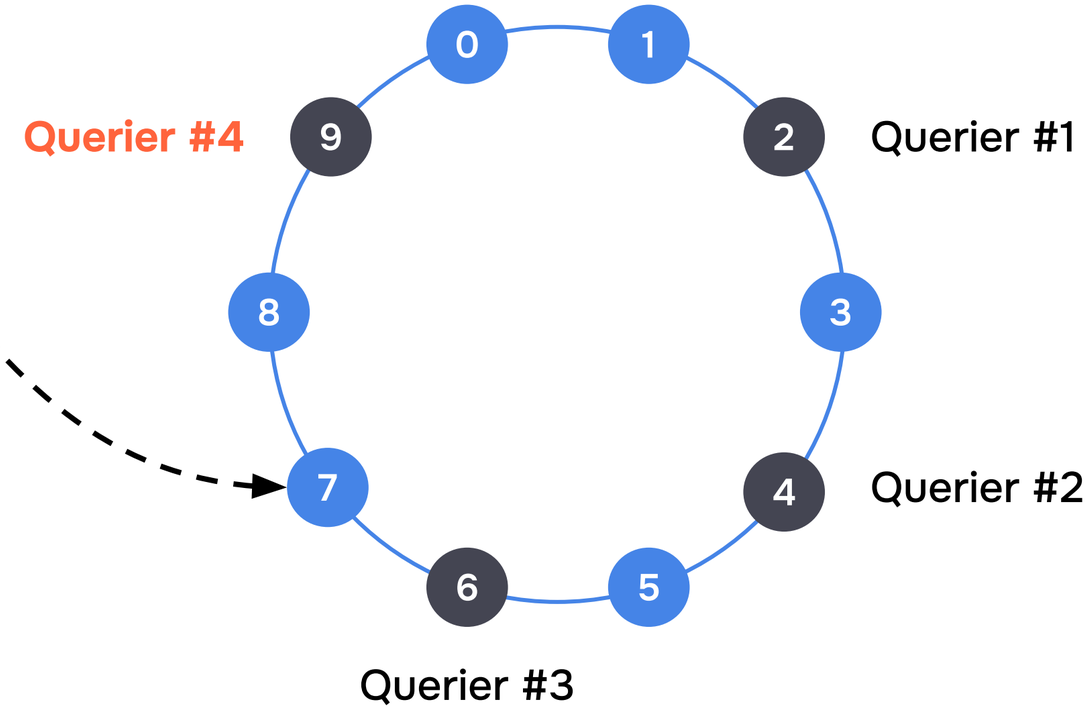
```
UID: 01HZN9V7YZSC3F2Q0X8D0DYRZC
-> token = 7
```



Хеш-ринг

Block

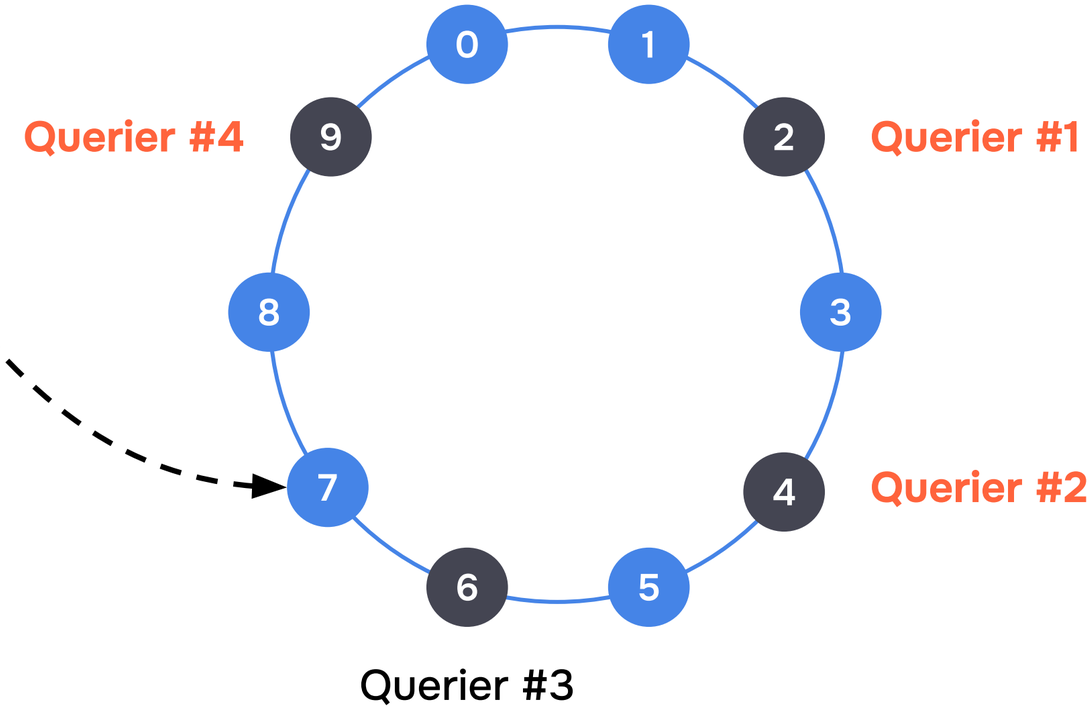
```
UID: 01HZN9V7YZSC3F2Q0X8D0DYRZC
-> token = 7
```

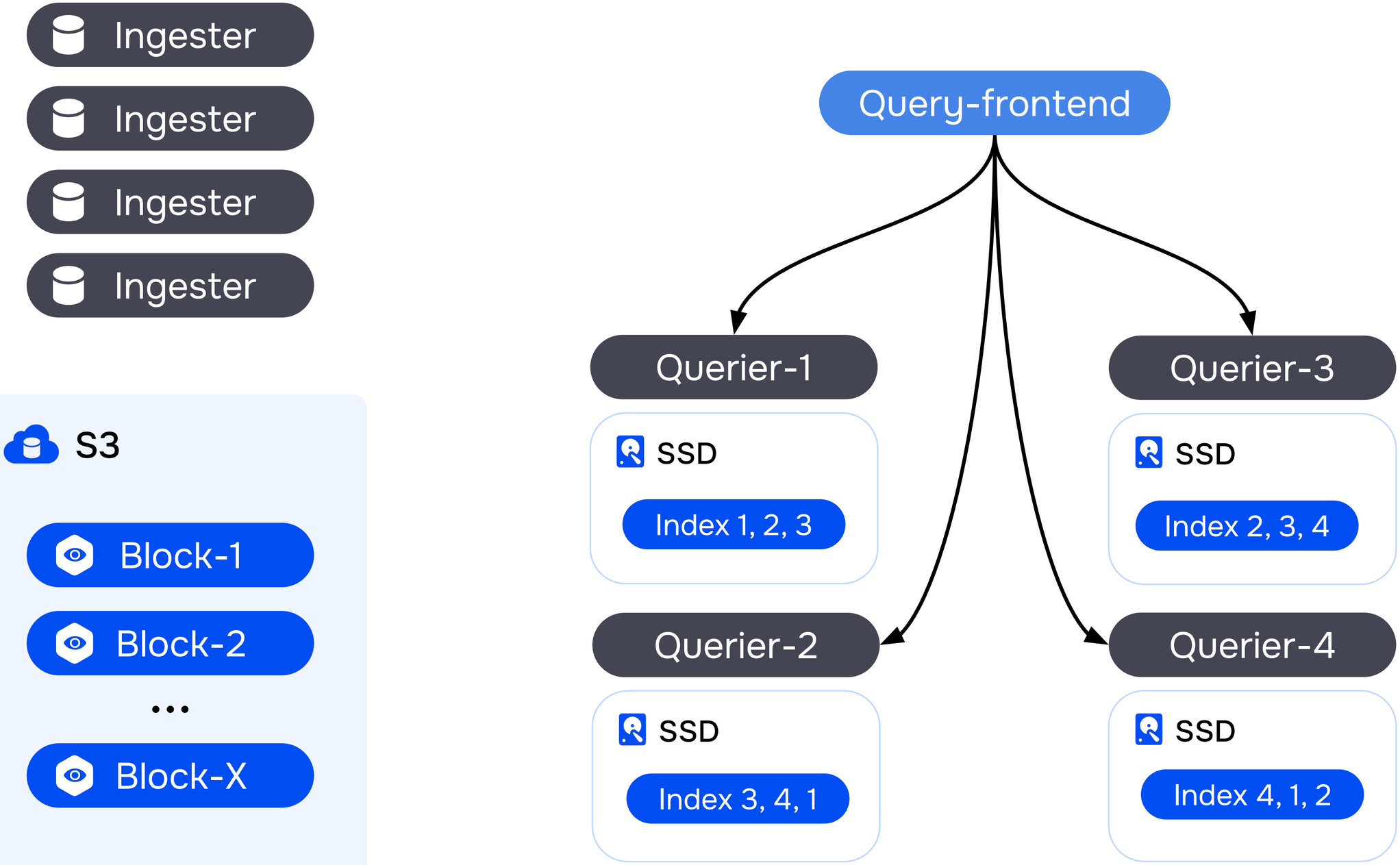


Хеш-ринг

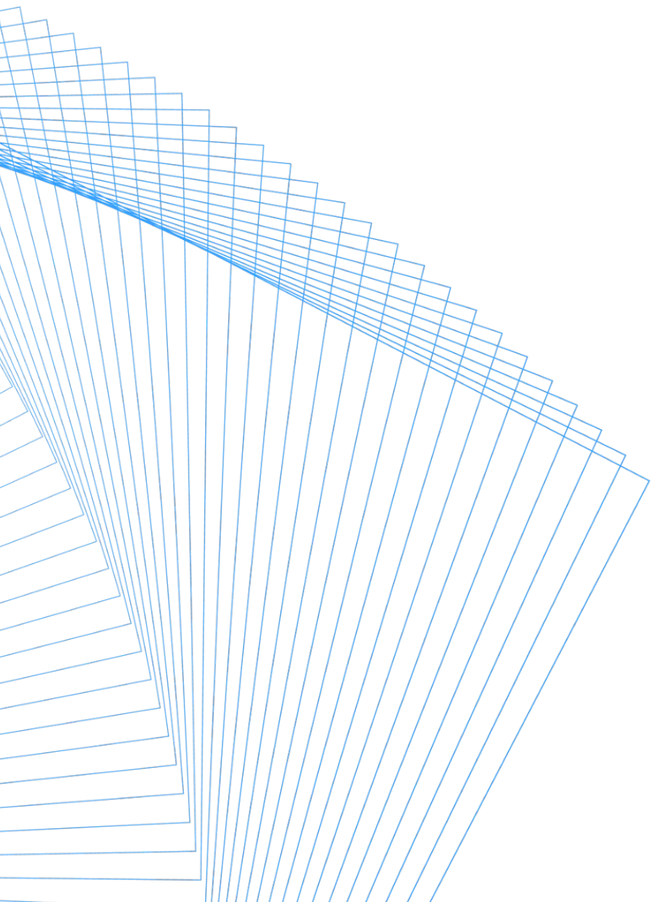
Block

`UID: 01HZN9V7YZSC3F2Q0X8D0DYRZC`
`-> token = 7`



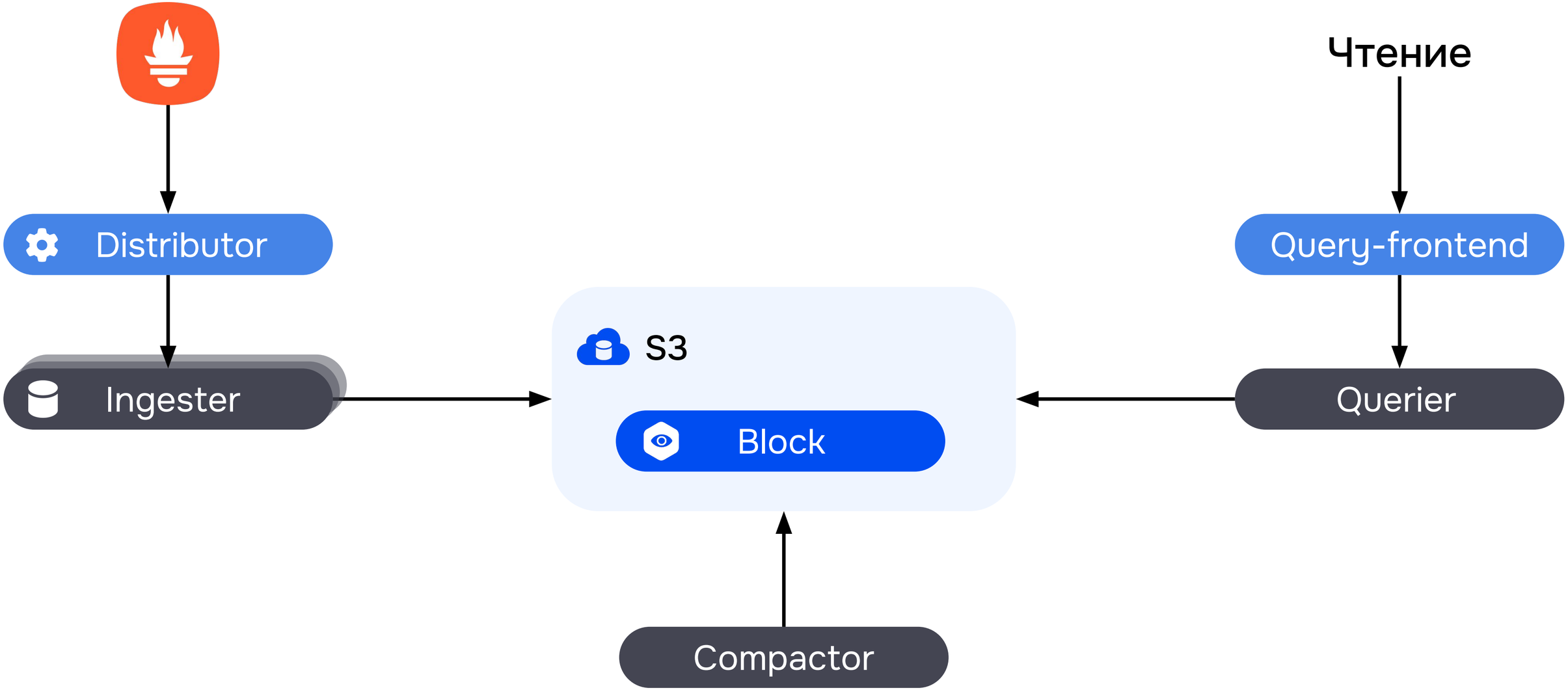

Ingester
Ingester
Ingester
Ingester
S3
Block-1
Block-2
...
Block-X
Query-frontend
Querier-1
SSD
Index 1, 2, 3
Querier-3
SSD
Index 2, 3, 4
Querier-2
SSD
Index 3, 4, 1
Querier-4
SSD
Index 4, 1, 2

Централизованная система мониторинга

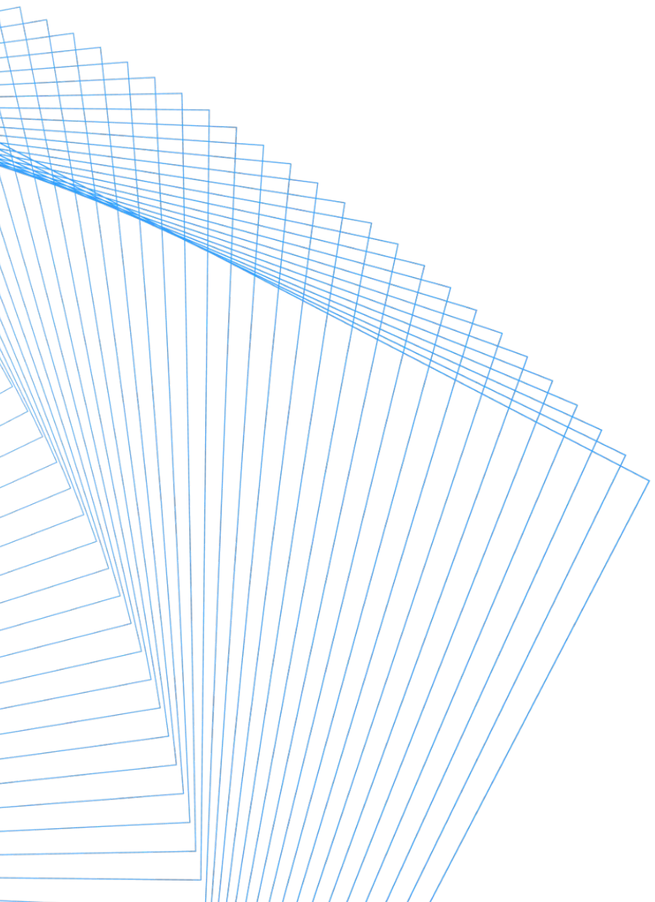


- Отказоустойчивость на запись: WC = RF/2 + 1
- Отказоустойчивость на чтение: RC = RF - WC + 1
- Шардирование на запись на основе хеш-ринга
- Исторические данные хранятся в S3
- При запуске читаем из S3 только индекс
- Читаем из чанков только нужные данные
- Шардирование на чтение на основе хеш-ринга

Итоги

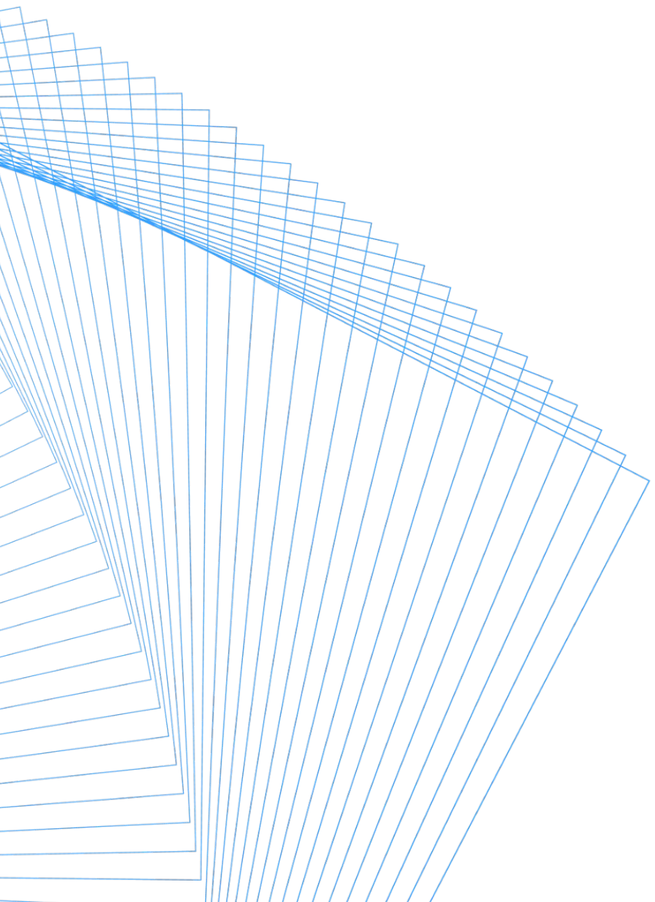

Чтение
Distributor
Query-frontend
S3
Ingester
Block
Querier
Compactor

Итоги



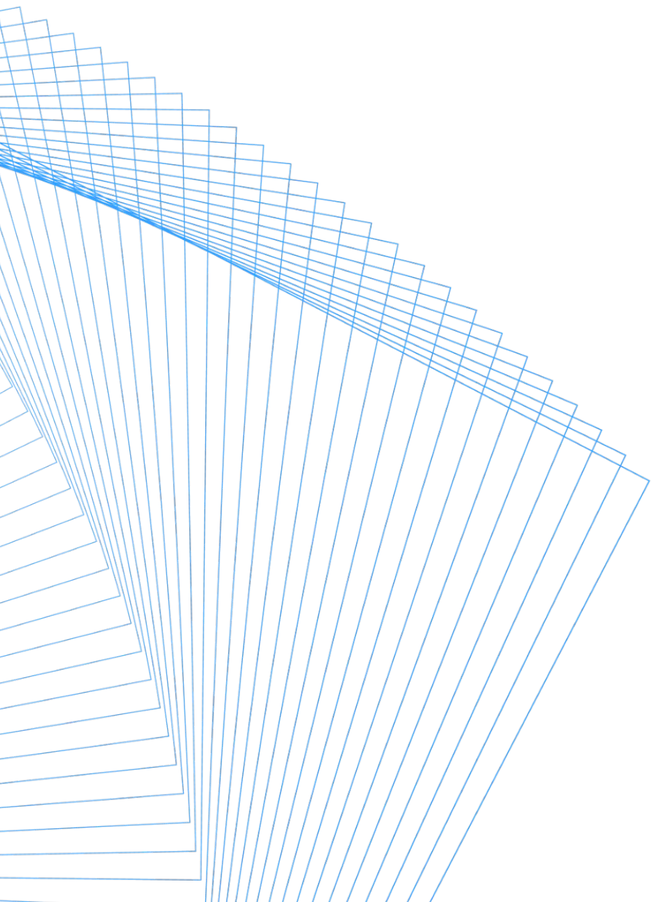
1 Отказоустойчивая система мониторинга

Итоги



1. Отказоустойчивая система мониторинга
2. Горизонтально масштабируемая система мониторинга

Итоги



1. Отказоустойчивая система мониторинга
2. Горизонтально масштабируемая система мониторинга
3. С хранением данных в S3

Deckhouse
ФЛАНТ

Буду рад ответить на ваши вопросы!

Telegram

YouTube

Habr

@Magvai69

github.com/deckhouse/deckhouse

github.com/deckhouse/prompp