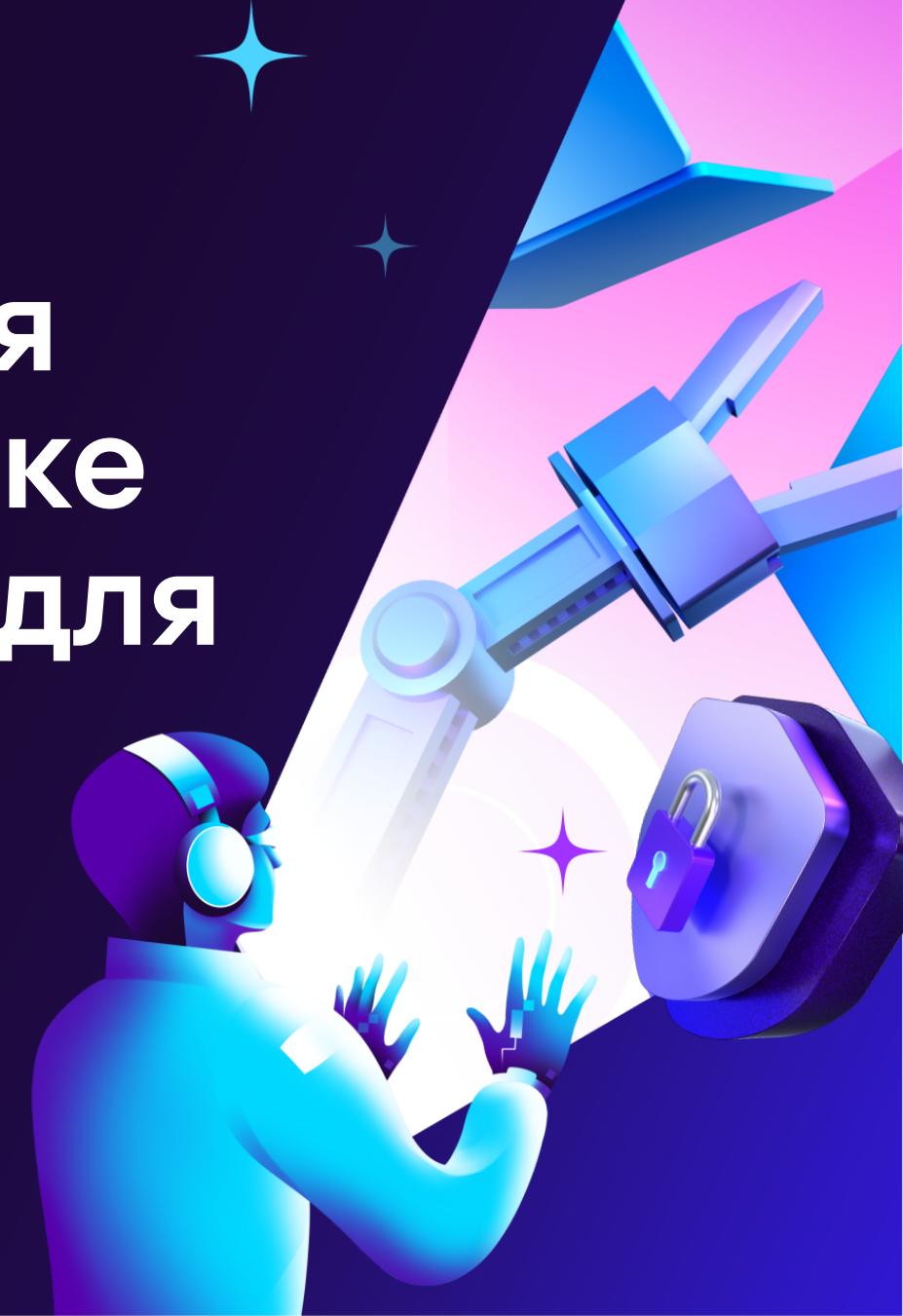
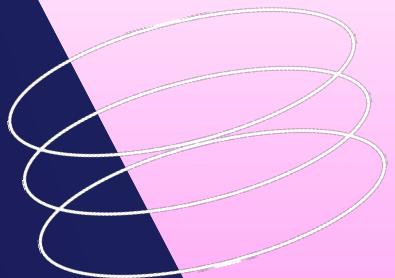


Алексей Княжев

Старший инженер, Авито @ DBaaS

# Как мы снизили до нуля операционку по приемке Kubernetes-кластеров для баз данных





- › В IT больше 5 лет.
- › С приходом в Авито в октябре 2023 года переобулся в DBRE.
- › Разрабатываю платформу DBaaS.
- › Интересуюсь Kubernetes, базами данных, пивом.

**Алексей Княжев**

@muhomorfus

Старший инженер в команде DBaaS, Авито.



# План доклада



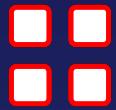
01

## Платформа DBaaS

Что такое DBaaS?

Что из себя представляет DBaaS  
в Авито в 2025?





## DBaaS – это

Базы данных как услуга.



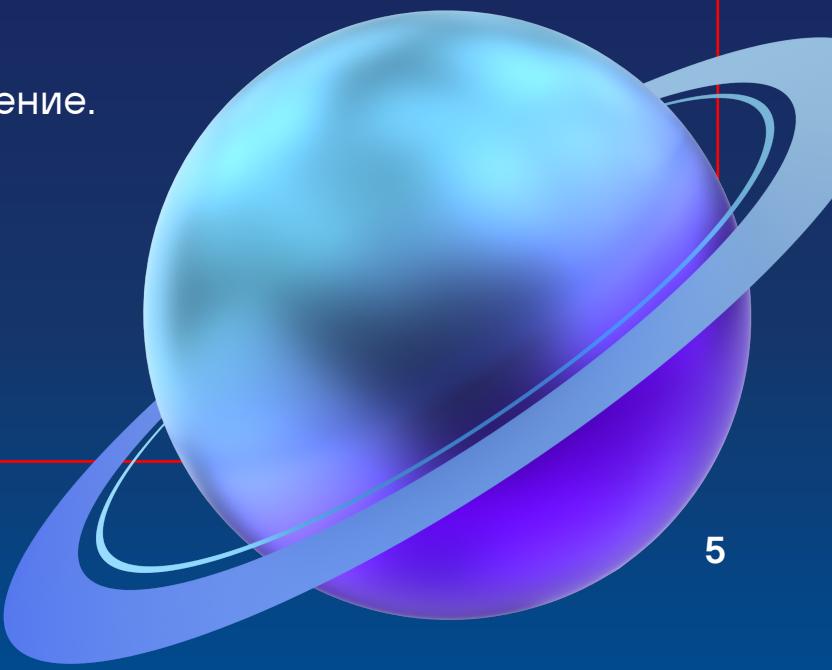
## DBaaS – это

Полная автоматизация жизненного цикла базы данных:  
от развёртывания до удаления и отправки данных на архивное хранение.



## DBaaS – это

Снятие паразитной когнитивной нагрузки по работе с базами  
данных с продуктовых разработчиков.



# DBaaS + Kubernetes = ❤



В качестве среды исполнения платформа DBaaS использует Kubernetes. Это позволяет использовать достаточно разнообразной автоматики, идущей «в комплекте».



Автоматизация запуска приложений



Простота обслуживания железа



Наличие готовых решений в opensource

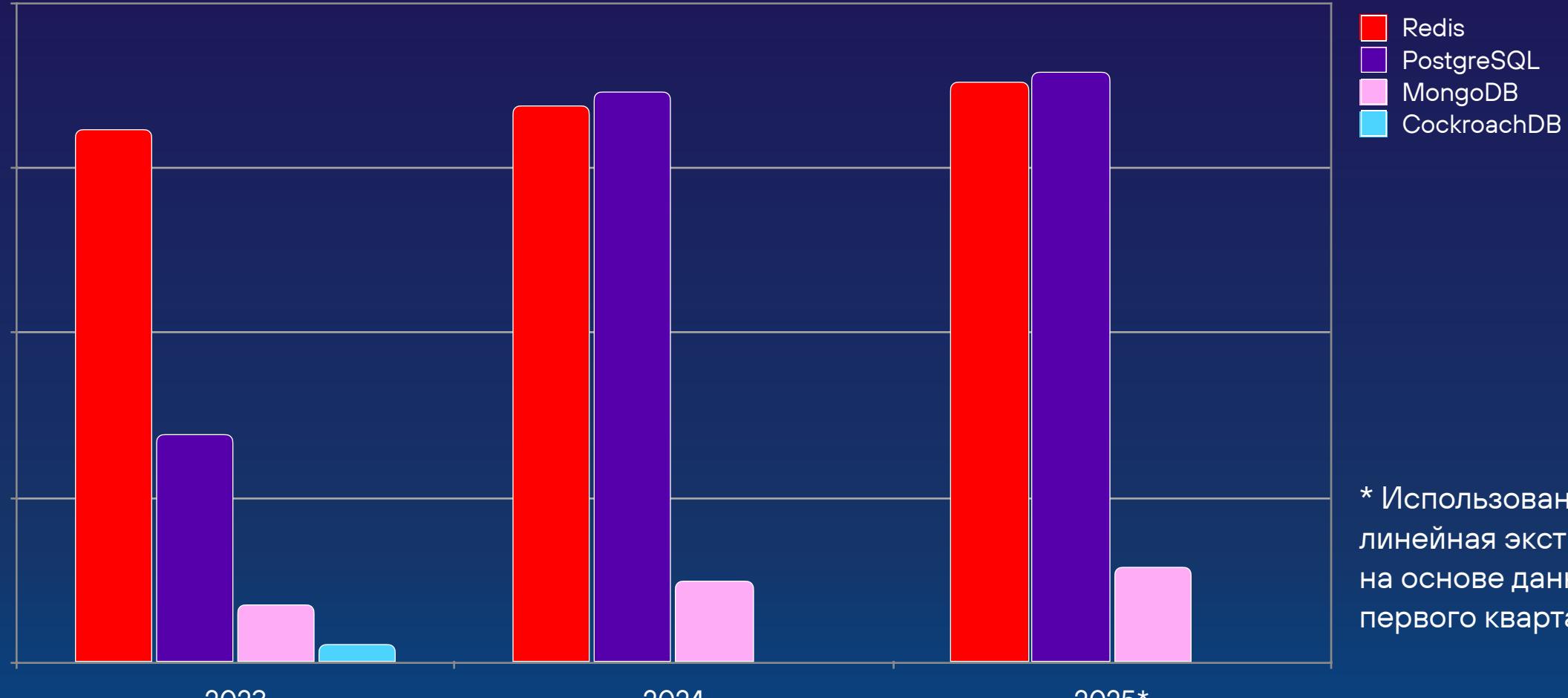


Независимость от окружения

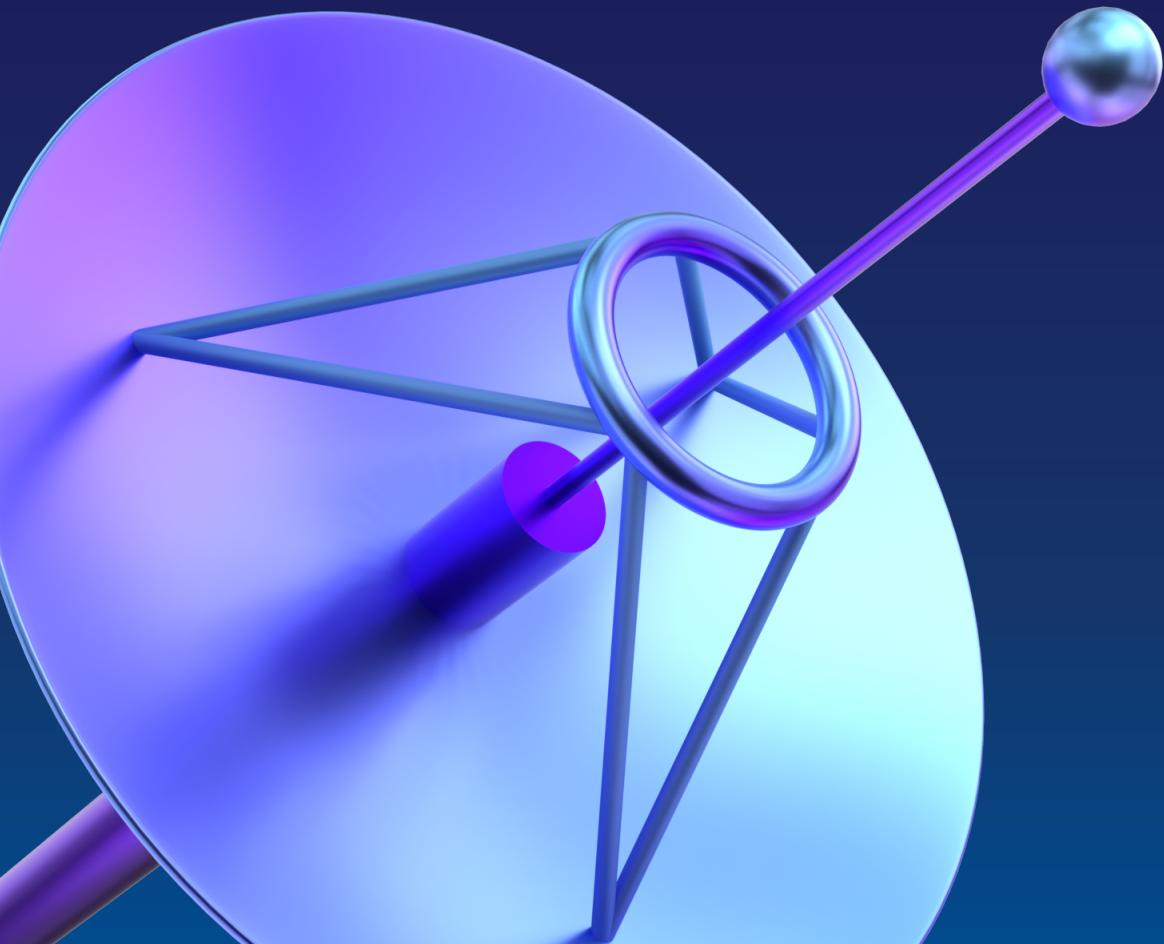
# DBaaS в цифрах



Количество создаваемых хранилищ



# DBaaS в цифрах



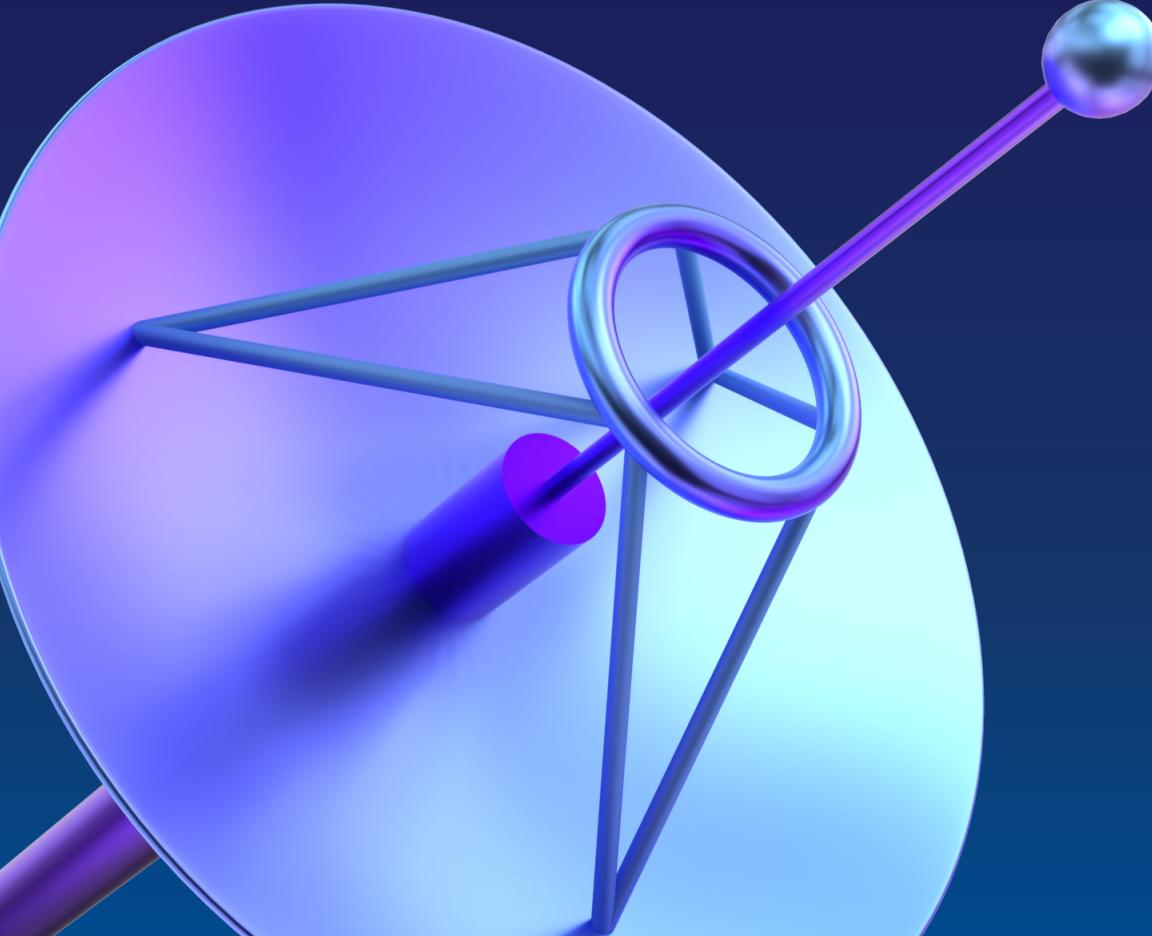
**~ 4 600** Хранилищ

**4 (7)** Технологий

**> 1 000** Worker-нод

**11** Kubernetes-кластеров

# DBaaS в планах



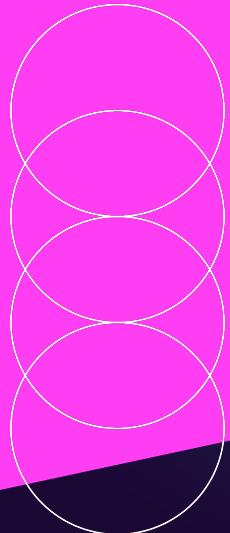
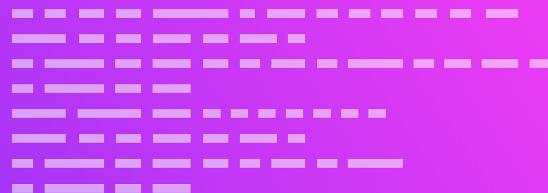
Безразмерные БД  
(YugabyteDB, Valkey)

Zero-downtime DB

Autoscaling DB

И многое чего еще...

02



## Базы в K8s?!

Да-да, это законно...

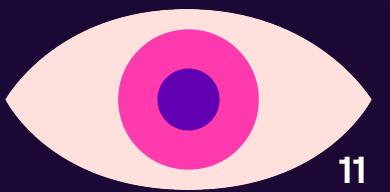
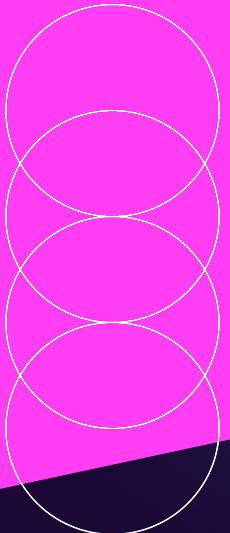
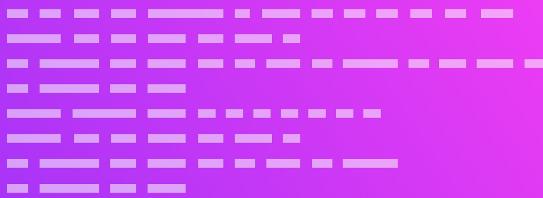


10

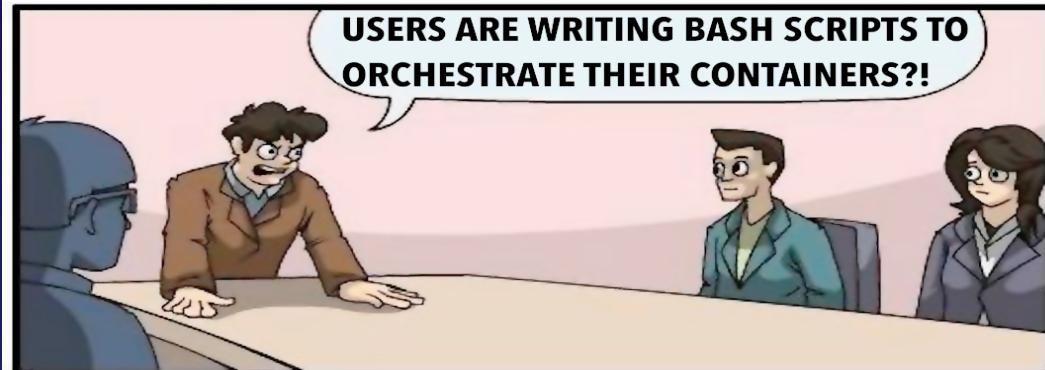


Я не буду  
рассказывать подробности  
работы в Multi-DC.

Не влезем в тайминг (мне лень).



# Docker, Inc 2014



Как хранить данные персистентно для приложений, запускаемых в Kubernetes?

В качестве плагина CSI для платформы DBaaS в Авито мы используем TopoLVM.

-  Создаёт persistent volumes с использованием LVM (Logical Volume Manager).
-  Поддерживает dynamic provisioning и volume expansion без даунтайма.
-  Изолирует volumes на уровне блочных устройств.
-  Умеет в интеграцию с планировщиком Kubernetes, учитывая объем свободного диска.

# Проблема «шумных соседей»

Запущенные на одной ноде поды разделяют ее ресурсы. Из-за этого, если не ограничивать разделяемые ресурсы, может возникнуть проблема **«шумных соседей»** — когда чрезмерное потребление какого-то из разделяемых ресурсов подом приводит к деградации других подов.



# Проблема «шумных соседей»



Для решения проблемы «шумных соседей» по сетевому I/O разработали **netochka**.

- ➡ Базируется на утилите tc.
- ➡ Работает в режиме сайдкар-контейнера.
- ➡ Экспортирует все нужные нам метрики по Network I/O.
- ➡ Читает конфиг из YAML-файла, на основе него генерирует правила для tc.

# Проблема «шумных соседей»



ALLO IOVA ETO TI??

Для решения проблемы «шумных соседей» по дисковому I/O разработали **ioba**.

- 💻 Базируется на механизме ядра cgroup v2.
- ernetes Работает в режиме DaemonSet-a.
- 📊 Ограничения хранятся в Custom Resource.
- ⚡ Реагирует на создание подов на ноде.
- 📈 Экспортирует все нужные нам метрики по дисковому I/O.

# Обслуживание оборудования



Когда нужно вывести оборудование в обслуживание, необходимо освободить Kubernetes-ноду. При этом, нативная автоматика Kubernetes не позволяет выселять поды с ноды с учетом сохранения гарантий доступности платформы.

Для решения проблемы мы разработали **dbaas-descheduler**.

- 🔧 Интегрирован с kubectl drain и Eviction API.
- 💀 При попытке выселения пода срабатывает Validation Webhook, который заводит задачи на выселение подов БД с ноды.
- ✚ Поды выселяются с учетом гарантий доступности платформы, выселенные реплики наполняются за счет средств репликации.

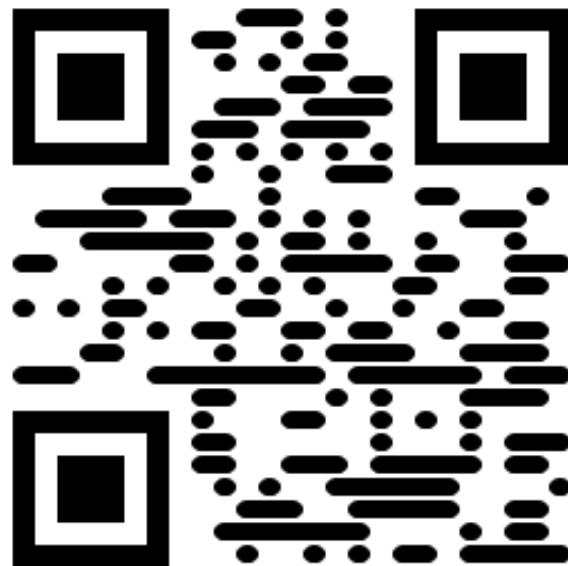
# Обслуживание оборудования



Для решения проблемы мы разработали **dbaas-descheduler**.

- 🔧 Интегрирован с kubectl drain и Eviction API.
- 💀 При попытке выселения пода срабатывает Validation Webhook, который заводит задачи на высечение подов БД с ноды.
- ➕ Поды выселяются с учетом гарантий доступности платформы, выселенные реплики наполняются за счет средств репликации.

# Avito Database Meetup #2



[t.me/avitotech](https://t.me/avitotech)

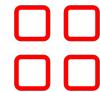
03

# Приемка кластера

Анатомия нашего K8s-кластера и процесс его приемки



# Анатомия нашего K8s-кластера



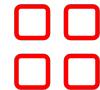
db-operators

Поднимают базы со  
всеми необходимыми  
зависимостями

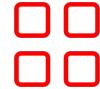


yummy-beer

# Анатомия нашего K8s-кластера



db-operators



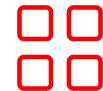
ioba

• Ограничивает  
дисковый ввод-вывод  
для реплик хранилищ



yummy-beer

# Анатомия нашего K8s-кластера



db-operators



ioba



dbaas-descheduler



yummy-beer

- Выселяет реплики с ноды с учетом гарантий доступности

# Анатомия нашего K8s-кластера



db-operators



ioba



dbaas-descheduler



yummy-beer

Нужен для работы  
database discovery



consul-agent

# Анатомия нашего K8s-кластера



db-operators



ioba



dbaas-descheduler



consul-agent



yummy-beer

Плагин CSI



topolvm

# Анатомия нашего K8s-кластера



db-operators



ioba



dbaas-descheduler



yummy-beer



consul-agent



topolvm



vm-agent

Собирает метрики  
хранилищ

# Анатомия нашего K8s-кластера



db-operators



ioba



dbaas-descheduler



consul-agent



topolvm



vm-agent



yummy-beer

# Анатомия нашего K8s-кластера



db-operators



ioba



dbaas-descheduler



yummy-beer



consul-agent



topolvm



vm-agent

Ожидаем уже готовыми

# Кроме того, нужно, чтобы в кластере



была рабочая интеграция с общим DNS

# Кроме того, нужно, чтобы в кластере



была рабочая интеграция с общим DNS

были проставлены емкости нод по сети

# Кроме того, нужно, чтобы в кластере



была рабочая интеграция с общим DNS

были проставлены емкости нод по сети

были проставлены емкости нод по дисковому I/O

# Кроме того, нужно, чтобы в кластере



была рабочая интеграция с общим DNS

были проставлены емкости нод по сети

были проставлены емкости нод по дисковому I/O

на нодах были проставлены необходимые taints

# Кроме того, нужно, чтобы в кластере



была рабочая интеграция с общим DNS

были проставлены емкости нод по сети

были проставлены емкости нод по дисковому I/O

на нодах были проставлены необходимые taints

taint-ы соответствовали конфигурации оборудования

# Кроме того, нужно, чтобы в кластере



была рабочая интеграция с общим DNS

были проставлены емкости нод по сети

были проставлены емкости нод по дисковому I/O

на нодах были проставлены необходимые taints

ноды были правильно настроены

taint-ы соответствовали конфигурации оборудования

# Кроме того, нужно, чтобы в кластере



была рабочая интеграция с общим DNS

были проставлены емкости нод по сети

валидно генерировались SAN-сертификаты хранилищ

были проставлены емкости нод по дисковому I/O

на нодах были проставлены необходимые taints

ноды были правильно настроены

taint-ы соответствовали конфигурации оборудования

# Кроме того, нужно, чтобы в кластере



была рабочая интеграция с общим DNS

отсутствовало расхождение часов

были проставлены емкости нод по сети

валидно генерировались SAN-сертификаты хранилищ

были проставлены емкости нод по дисковому I/O

на нодах были проставлены необходимые taints

ноды были правильно настроены

taint-ы соответствовали конфигурации оборудования

# Про генерацию SAN-сертификатов



Для генерации секретов хранилищ, пользовательских учеток, сертификатов и прочего мы используем Vault, взаимодействуем с которым через плагин vault-database-credentials (VDC).

Доменное имя реплик хранилища зависит от кластеров, в которых они запущены.

Соответственно, в конфигурации плагина содержится список кластеров, и на основе этого списка генерируется SAN-сертификат для реплики хранилища.

# Про генерацию SAN-сертификатов



Для генерации секретов хранилищ, пользовательских учеток, сертификатов и прочего мы используем Vault, взаимодействуем с которым через плагин vault-database-credentials (VDC).

Доменное имя реплик хранилища зависит от кластеров, в которых они запущены.

Соответственно, в конфигурации плагина содержится список кластеров, и на основе этого списка генерируется SAN-сертификат для реплики хранилища.

*redis1-rs-rs001-0.redis1-rs-rs001.redis1-rs-rs001.svc.yummy-beer.k8s*

# Про генерацию SAN-сертификатов



Для генерации секретов хранилищ, пользовательских учеток, сертификатов и прочего мы используем Vault, взаимодействуем с которым через плагин vault-database-credentials (VDC).

Доменное имя реплик хранилища зависит от кластеров, в которых они запущены.

Соответственно, в конфигурации плагина содержится список кластеров, и на основе этого списка генерируется SAN-сертификат для реплики хранилища.

Имя хранилища

`redis1-rs-rs001-0.redis1-rs-rs001.redis1-rs-rs001.svc.yummy-beer.k8s`

# Про генерацию SAN-сертификатов



Для генерации секретов хранилищ, пользовательских учеток, сертификатов и прочего мы используем Vault, взаимодействуем с которым через плагин vault-database-credentials (VDC).

Доменное имя реплик хранилища зависит от кластеров, в которых они запущены.

Соответственно, в конфигурации плагина содержится список кластеров, и на основе этого списка генерируется SAN-сертификат для реплики хранилища.

redis1-rs-rs001-0.redis1-rs-rs001.redis1-rs-rs001.svc.yummy-beer.k8s

Имя репликасета

# Про генерацию SAN-сертификатов



Для генерации секретов хранилищ, пользовательских учеток, сертификатов и прочего мы используем Vault, взаимодействуем с которым через плагин vault-database-credentials (VDC).

Доменное имя реплик хранилища зависит от кластеров, в которых они запущены.

Соответственно, в конфигурации плагина содержится список кластеров, и на основе этого списка генерируется SAN-сертификат для реплики хранилища.

Имя кластера

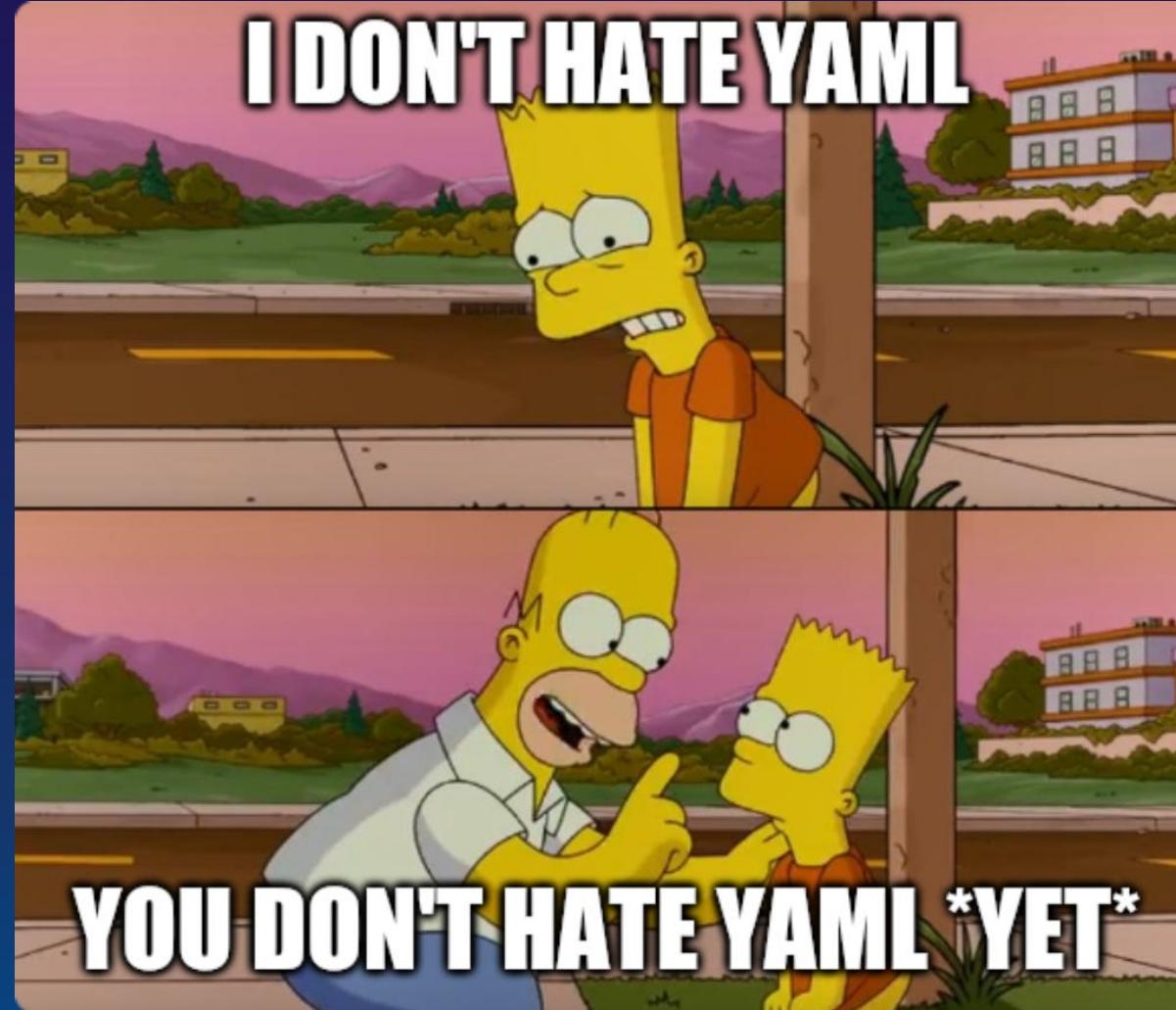
`redis1-rs-rs001-0.redis1-rs-rs001.redis1-rs-rs001.svc.yummy-beer.k8s`

# Процесс приемки кластера



№	Что делаем?
1	Проверяем базовую готовность кластера: простановку пулов, ёмкостей нод, наличие TopoLVM.
2	Заводим Jira-таски на добавление кластера в конфигурации плагинов для Vault, ждем выполнения.
3	Устанавливаем наши компоненты: операторы, ioba, consul, descheduler.
4	Регистрируем кластер в системе.
5	Перегенерируем SAN-сертификаты для всех хранилищ на платформе.
6	Переносим тестовые хранилища в новый кластер, проверяем работу.

# Процесс приемки кластера



# Что тут не так?



Весь процесс приемки очень труднозатратен, нужно создавать много Pull-Request-ов с изменениями манифестов, много что проверить руками.

Есть очень много требований, которые относятся только к нашим Kubernetes-кластерам.

## Закон Мерфи

Если что-нибудь может пойти не так, оно пойдёт не так.

Периодически какие-то из требований к кластерам или worker-нодам не выполняются, и у нас начинаются проблемы...

Что тут не так?  
phdX  
Было бы  
лучше  
Было бы  
лучше  
Было бы  
лучше  
Было бы  
лучше

Наша команда принимает кластера достаточно часто.  
Сейчас у нас **11** кластеров, минимум **3** на подходе. А мы...

- Тратим целый человеко-спринт на приемку кластеров.
- Тратим много нервов при возникновении проблем.
- Ошибаемся при копипасте манифестов.
- В процессе эксплуатации напарываемся на тяжело отлаживаемые проблемы.

# Нужно пилить автоматизацию!

phd X pt



04



# Автоматизация 1/3

Как мы автоматизировали проверки кластера и нод?

# Существующие решения



## **Kuberhealthy** ([clc.to/gM9pNA](https://clck.ru/gM9pNA))

- Неудобно делать проверки отдельных worker-нод.
- Умеет все что нам нужно для проверки кластера в целом.

## **Node problem detector (NPD)** ([clc.to/ct\\_bEA](https://clck.ru/ct_bEA))

- Предназначен для проверок уровня нод.
- Все очень плохо с кастомизацией проверок в рантайме.

# Существующие решения



## Kuberhealthy ([clc.to/gM9pNA](https://clc.to/gM9pNA))

- Неудобно делать проверки отдельных worker-нод.
- Умеет все что нам нужно для проверки кластера в целом.



## Node problem detector (NPD) ([clc.to/ct\\_bEA](https://clc.to/ct_bEA))

- Предназначен для проверок уровня нод.
- Все очень плохо с кастомизацией проверок в рантайме.



# А как же проверки worker-нод?



Для проверок нод кластера мы решили создать свое решение с блекджеком, кастомизируемыми проверками и экспортом всех нужных нам метрик.

# А как же проверки worker-нод?



Для проверок нод кластера мы решили создать свое решение с блекджеком, кастомизируемыми проверками и экспортом всех нужных нам метрик.

**node-validator**

# А как же проверки worker-нод?



Для проверок нод кластера мы решили создать свое решение с блекджеком, кастомизируемыми проверками и экспортом всех нужных нам метрик.

## node-validator

При этом для такого механизма одним из важнейших требований была блокировка запуска реплик хранилища на неготовой ноде. Это помогло бы избежать ситуаций, когда хранилища запускаются на нодах, которые были неправильно настроены.

# Первая попытка



# Первая попытка



**worker node**

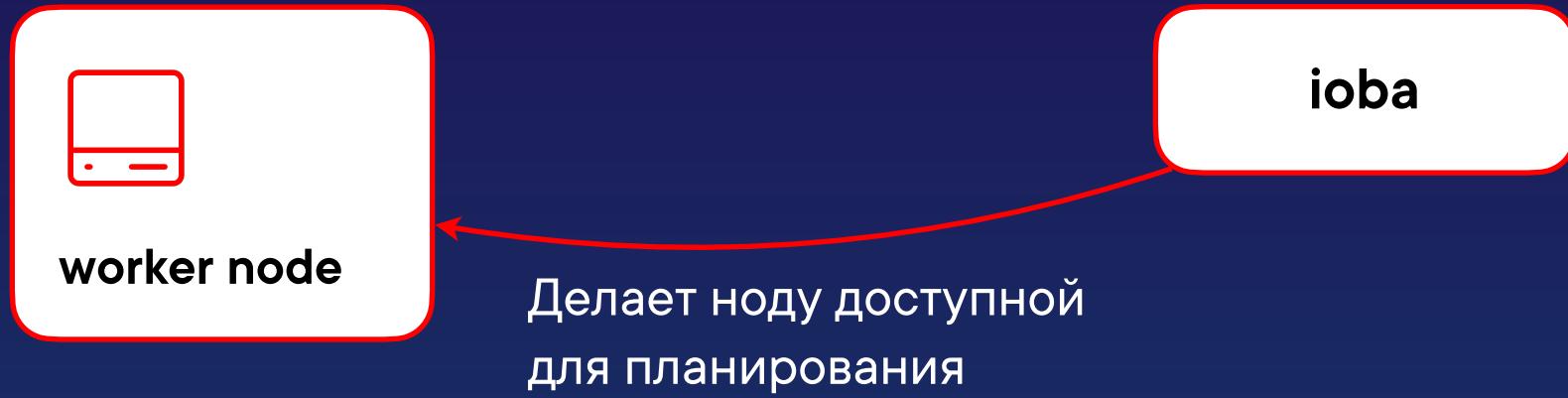
**ioba**

Проводит бенчмарк  
ноды  
по дисковому I/O

# Первая попытка



# Первая попытка



# Первая попытка



**worker node**

**ioba**

Такая схема смогла хорошо себя зарекомендовать: в кластерах, которые поддерживает ioba, хранилища на добавляемых нодах запускаются только в том случае, если на нодах проставлена емкость по дисковому вводу-выводу.

# Собираем полноценный инструмент



```
apiVersion: node-validator.dbaas.avito.ru/v1alpha1
kind: NodeValidation
metadata:
  name: check-pool-is-presented
  namespace: node-validator
  labels:
    node-validator.dbaas.avito.ru/priority: true
spec:
  nodesQuery: |
    [.spec.taints[] | .key] | index("node-role.kubernetes.io/control-plane") != null
  runInterval: 1m
  metric: |
    kube_node_spec_taint{cluster=~"{{ .Cluster }}", node="{{ .Node }}", key="dbaas.pool"}
  operator: eq
  value: 1
```

# Собираем полноценный инструмент



```
apiVersion: node-validator.dbaas.avito.ru/v1alpha1
kind: NodeValidation
metadata:
  name: check-pool-is-presented
  namespace: node-validator
  labels:
    node-validator.dbaas.avito.ru/priority: true
spec:
  nodesQuery: |
    [.spec.taints[] | .key] | index("node-role.kubernetes.io/control-plane") != null
  runInterval: 1m
  metric: |
    kube_node_spec_taint{cluster=~"{{ .Cluster }}", node="{{ .Node }}", key="dbaas.pool"}
  operator: eq
  value: 1
```

Метка приоритетности проверки

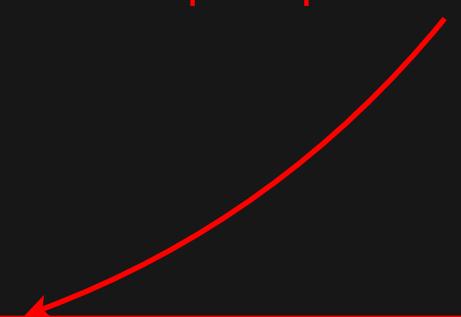


# Собираем полноценный инструмент



```
apiVersion: node-validator.dbaas.avito.ru/v1alpha1
kind: NodeValidation
metadata:
  name: check-pool-is-presented
  namespace: node-validator
  labels:
    node-validator.dbaas.avito.ru/priority: true
spec:
  nodesQuery: |
    [.spec.taints[] | .key] | index("node-role.kubernetes.io/control-plane") != null
  runInterval: 1m
  metric: |
    kube_node_spec_taint{cluster=~"{{ .Cluster }}", node="{{ .Node }}", key="dbaas.pool"}
  operator: eq
  value: 1
```

JQ-выражение для выборки проверяемых нод

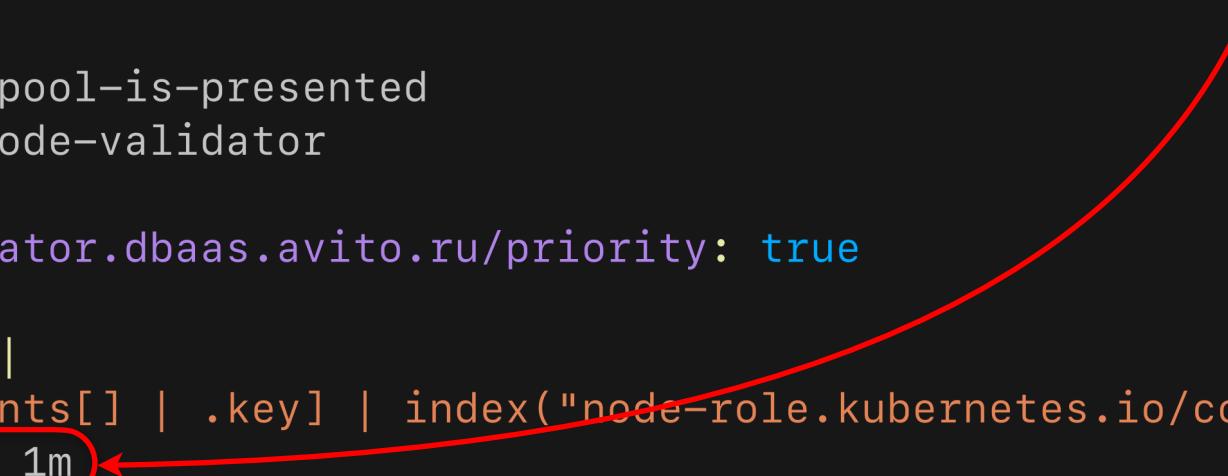


# Собираем полноценный инструмент



```
apiVersion: node-validator.dbaas.avito.ru/v1alpha1
kind: NodeValidation
metadata:
  name: check-pool-is-presented
  namespace: node-validator
  labels:
    node-validator.dbaas.avito.ru/priority: true
spec:
  nodesQuery: |
    [.spec.taints[] | .key] | index("node-role.kubernetes.io/control-plane") != null
  runInterval: 1m
  metric: |
    kube_node_spec_taint{cluster=~"{{ .Cluster }}", node="{{ .Node }}", key="dbaas.pool"}
operator: eq
value: 1
```

Интервал проверки



# Собираем полноценный инструмент



```
apiVersion: node-validator.dbaas.avito.ru/v1alpha1
kind: NodeValidation
metadata:
  name: check-pool-is-presented
  namespace: node-validator
  labels:
    node-validator.dbaas.avito.ru/priority: true
spec:
  nodesQuery: |
    [.spec.taints[] | .key] | index("node-role.kubernetes.io/control-plane") != null
  runInterval: 1m
  metric: |
    kube_node_spec_taint{cluster=~"{{ .Cluster }}", node="{{ .Node }}", key="dbaas.pool"}
  operator: eq
  value: 1
```

Условие, по которому  
мы проверяем  
«ЖИВОСТЬ» НОДЫ



# Собираем полноценный инструмент



100

```
apiVersion: node-validator.dbaas.avito.ru/v1alpha1
kind: NodeValidation
metadata:
  name: check-pool-is-presented
  namespace: node-validator
```

Решения с метриками нам оказалось достаточно +  
оно не приводит к росту нагрузки на кластер.

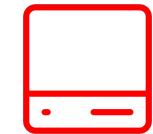
```
    selector: "db-1-plane") != null  
runInterval: 1m  
metric: |  
  kube_node_spec_taint{cluster=~"{{ .Cluster }}", node="{{ .Node }}", key="dbaas.pool"}  
operator: eq  
value: 1
```

# Схема работы

**node-validator**

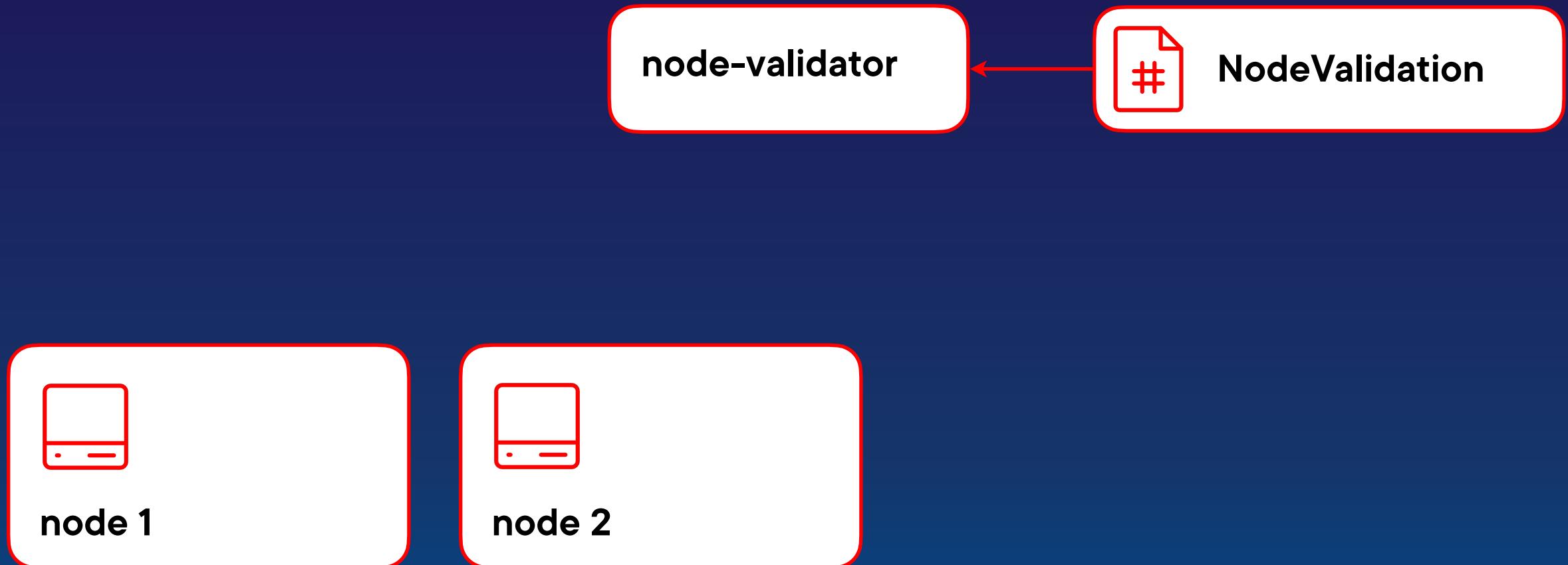


**node 1**



**node 2**

# Схема работы



# Схема работы

Запускает  
периодические  
проверки подходящих  
нод.

• **node-validator**



**NodeValidation**

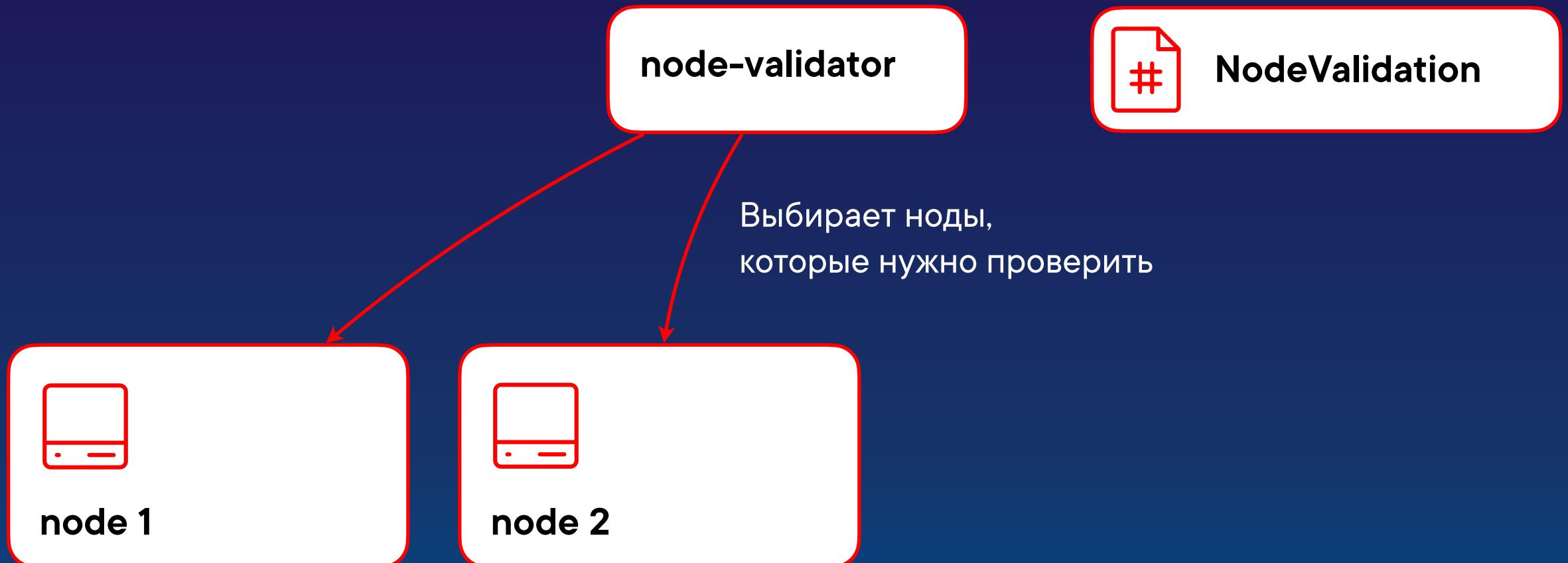


**node 1**

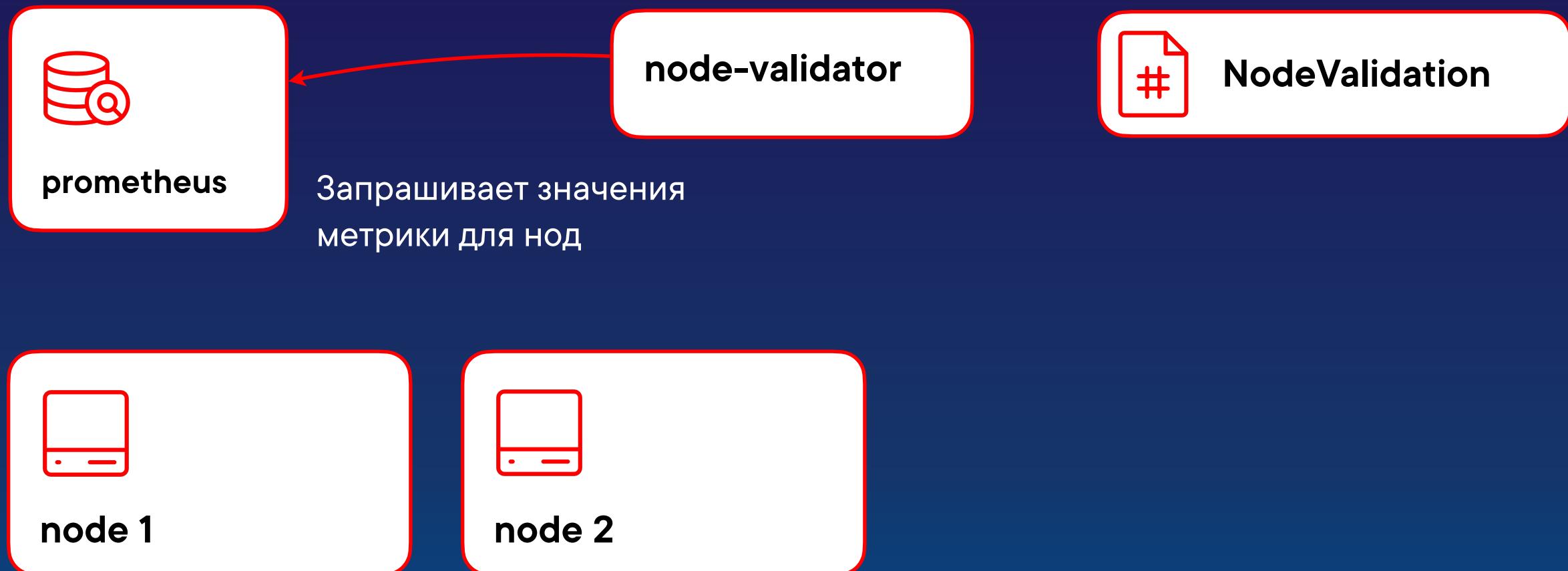


**node 2**

# Схема работы одной проверки



# Схема работы одной проверки



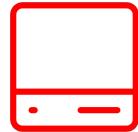
# Схема работы

Сравнивает значение  
метрики с эталоном.

• **node-validator**



**NodeValidation**

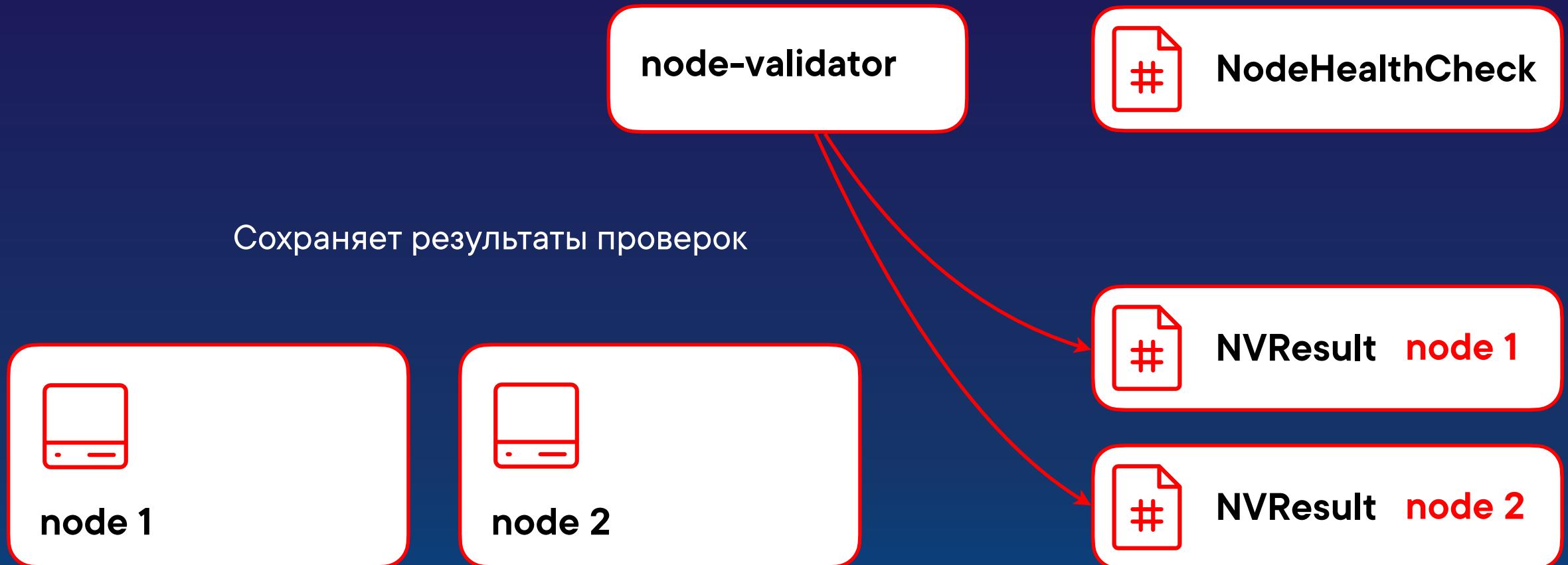


**node 1**



**node 2**

# Схема работы



# Результат проверки



```
apiVersion: node-validator.dbaas.avito.ru/v1alpha1
kind: NodeValidationResult
metadata:
  name: check-pool-is-presented-node1
  namespace: node-validator
  labels:
    node-validator.dbaas.avito.ru/priority: true
spec:
  nodeName: node1
status:
  state: Failed
```

# Результат проверки



```
apiVersion: node-validator.dbaas.avito.ru/v1alpha1
kind: NodeValidationResult
metadata:
  name: check-pool-is-presented-node1
  namespace: node-validator
  labels:
    node-validator.dbaas.avito.ru/priority: true
spec:
  nodeName: node1
status:
  state: Failed
```

Имя проверяемой  
ноды

# Результат проверки



```
apiVersion: node-validator.dbaas.avito.ru/v1alpha1
kind: NodeValidationResult
metadata:
  name: check-pool-is-presented-node1
  namespace: node-validator
  labels:
    node-validator.dbaas.avito.ru/priority: true
spec:
  nodeName: node1
status:
  state: Failed
```

Статус проверки  
Success/Failed

# Обработка результатов



- По результатам проверки на ноде выставляется Condition.
- Если все проверки успешны — с ноды снимается taint.
- Если хотя бы одна приоритетная проверка провалилась — на ноду ставится taint.

Это позволяет обеспечивать выполнение одного из основных требований — на неготовой ноде мы не будем пытаться запустить хранилища.

# Дашбордики



Summary ⓘ

# Issues

Clusters summary ⓘ

23:50 23:55 00:00 00:05 00:10 00:15

Validations summary ⓘ

Validation	Unhealthy nodes	Healthy
node-role-is-presented	All nodes healthy	
pool-is-presented	All nodes healthy	
network-capacity-filled	All nodes healthy	
ioba-running	All nodes healthy	
no-network-errors	10	
topolvm-running	All nodes healthy	
dbaas-pool-is-presented	All nodes healthy	

Nodes with issues ⓘ

Cluster	Node	Failed validations
keen-breeze	keen-breeze-default-worker-v2-kgg9r	no-network-errors
keen-breeze	keen-breeze-default-worker-v2-p58ct	no-network-errors
keen-breeze	keen-breeze-default-worker-v2-pp9ds	no-network-errors
keen-breeze	keen-breeze-default-worker-v2-rhzwh	no-network-errors
keen-breeze	keen-breeze-default-worker-v2-5l4np	no-network-errors
keen-breeze	keen-breeze-default-worker-v2-fp4z6	no-network-errors
keen-breeze	keen-breeze-default-worker-v2-kc7bh	no-network-errors

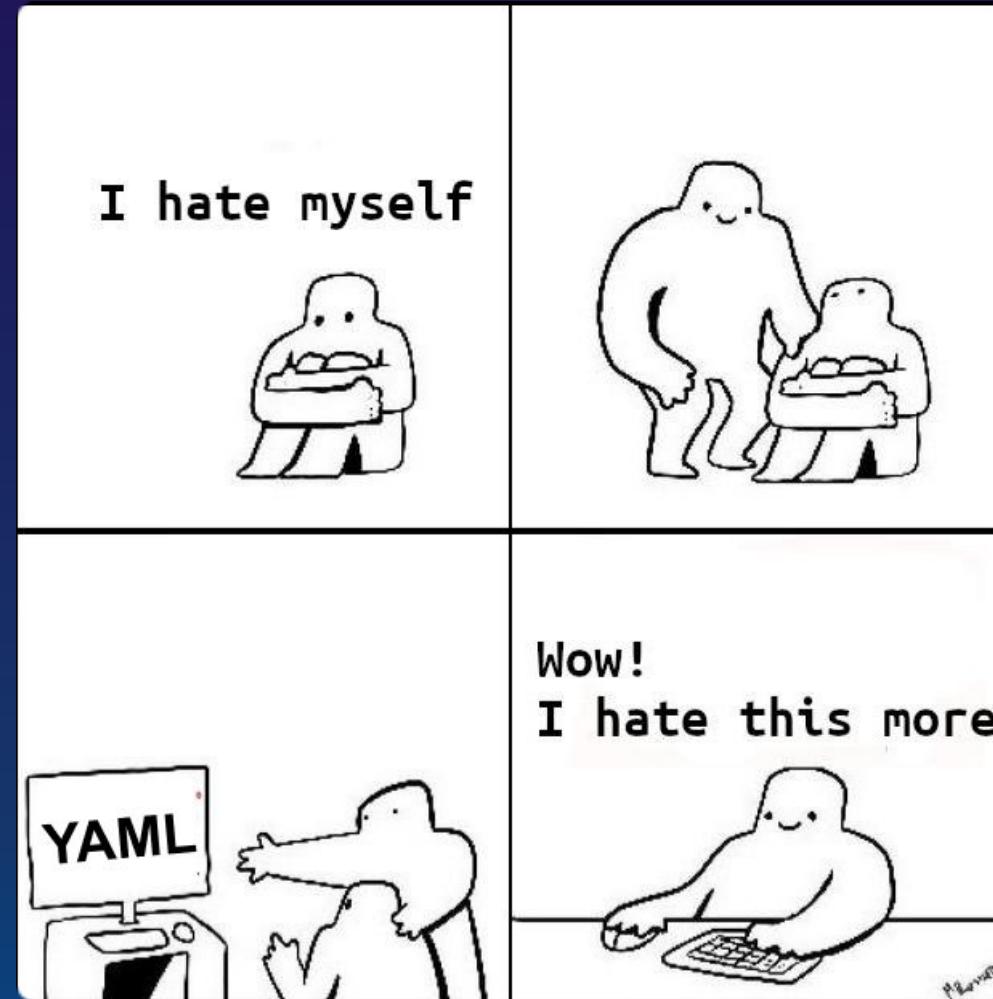
05



# Автоматизация 2/3

Как мы автоматизировали деплой компонентов?

# Зачем вообще автоматизировать?



# Магия ArgoCD



Для развертывания компонентов мы используем ArgoCD. Пишем манифесты в репозитории, создаем проект в ArgoCD и смотрим, как прекрасно все разворачивается

# Магия ArgoCD



Для развертывания компонентов мы используем ArgoCD. Пишем манифесты в репозитории, создаем проект в ArgoCD и смотрим, как прекрасно все разворачивается

Как оказалось, в самом ArgoCD есть все, что нужно для автоматизации развертывания компонентов в мультиклUSTERной среде!



# ApplicationSet



```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: ioba-appset
spec:
  generators:
    - clusters:
        selector:
          matchExpressions:
            - key: dbaas.avito.ru/ioba-enabled
              operator: In
              values: ["true"]
  template:
    metadata:
      name: 'ioba-{{name}}'
    spec:
      syncPolicy:
        automated:
          prune: true
      syncOptions:
        - CreateNamespace=true
    source:
      repoURL: ssh://.../ioba.git
      targetRevision: HEAD
      path: deployments/helm
      helm:
        values: |-
          clusterName: {{name}}
        image:
          tag: "2.2.8"
```

# ApplicationSet



```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: ioba-appset
spec:
  generators:
    - clusters:
        selector:
          matchExpressions:
            - key: dbaas.avito.ru/ioba-enabled
              operator: In
              values: ["true"]
  template:
    metadata:
      name: 'ioba-{{name}}'
    spec:
      syncPolicy:
        automated:
          prune: true
      syncOptions:
        - CreateNamespace=true
    source:
      repoURL: ssh://.../ioba.git
      targetRevision: HEAD
      path: deployments/helm
      helm:
        values: |-
          clusterName: {{name}}
          image:
            tag: "2.2.8"
```

Генераторы, с помощью которых будут создаваться проекты в ArgoCD

# ApplicationSet

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: ioba-appset
spec:
  generators:
    - clusters:
        selector:
          matchExpressions:
            - key: dbaas.avito.ru/ioba-enabled
              operator: In
              values: ["true"]
template:
  metadata:
    name: 'ioba-{{name}}'
  spec:
    syncPolicy:
      automated:
        prune: true
    syncOptions:
      - CreateNamespace=true
  source:
    repoURL: ssh://.../ioba.git
    targetRevision: HEAD
    path: deployments/helm
    helm:
      values: |-
        clusterName: {{name}}
        image:
          tag: "2.2.8"
```

Шаблон проекта

# ApplicationSet



```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: ioba-appset
spec:
  generators:
    - clusters:
        selector:
          matchExpressions:
            - key: dbaas.avito.ru/ioba-enabled
              operator: In
              values: ["true"]
  template:
    metadata:
      name: 'ioba-{{name}}'
    spec:
      syncPolicy:
        automated:
          prune: true
      syncOptions:
        - CreateNamespace=true
  source:
    repoURL: ssh://.../ioba.git
    targetRevision: HEAD
    path: deployments/helm
    helm:
      values: |-
        clusterName: {{name}}
      image:
        tag: "2.2.8"
```

Автоматическое  
создание  
неймспейса!

# Результат

DETAILS DIFF SYNC SYNC STATUS HISTORY AND ROLLBACK DELETE REFRESH ▾

APP HEALTH Healthy

SYNC STATUS Synced to HEAD (e5e14bf)

Auto sync is enabled.

Author: Aleksey Knyazhev <avknyazhev@...>

Comment: DBAAS-4651: fix lint

LAST SYNC Sync OK to e5e14bf

Succeeded 39 minutes ago (Tue Apr 15 2025 15:34:55 GMT+0300)

Author: Aleksey Knyazhev <avknyazhev@...>

Comment: DBAAS-4651: fix lint

+ - ⚡ 🔎 🔎 100%

```
graph LR; ioba[ioba] -- "a month" --> AS[AS applicationset]; AS -- "a month" --> app[9 Applications]
```

06



# Автоматизация 3/3

Как мы автоматизировали работу с сертификатами?

# Добавление кластера в конфигурацию

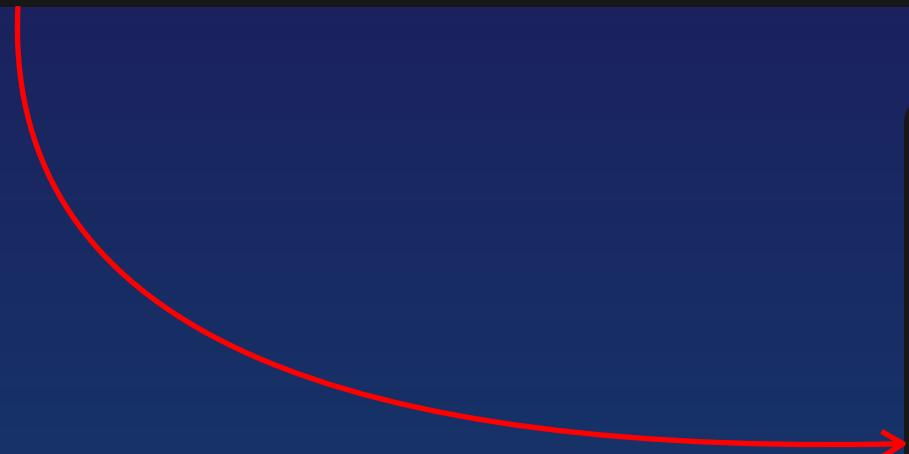


```
vault write -force vdc-redis/dbaas-clusters/yummy-beer
```

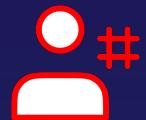


```
vault read vdc-redis/dbaas-clusters
```

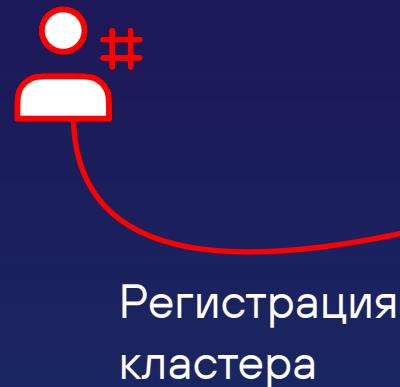
```
first-cluster  
second-cluster  
third-cluster  
yummy-beer
```



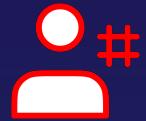
# Добавление кластера в конфигурацию



# Добавление кластера в конфигурацию



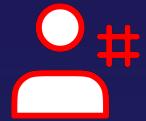
# Добавление кластера в конфигурацию



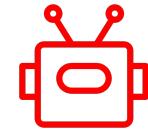
Добавлен новый  
кластер



# Добавление кластера в конфигурацию



dbaas



dbaas-vault

vault write .../dbaas-clusters/...

vault



vdc

# Перегенерация сертификатов



Реплики хранилищ взаимодействуют друг с другом с использованием TLS, мы используем SAN-сертификаты.

```
● ● ●  
Certificate:  
Data:  
    Subject: ..., CN = redis1-rs-rs001  
    ...  
X509v3 extensions:  
    X509v3 Subject Alternative Name:  
        DNS:*.redis1-rs-rs001.redis1-rs-rs001.svc.first-cluster.k8s,  
        DNS:*.redis1-rs-rs001.redis1-rs-rs001.svc.second-cluster.k8s,  
        DNS:*.redis1-rs-rs001.redis1-rs-rs001.svc.third-cluster.k8s  
Signature Algorithm: ...
```

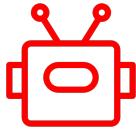
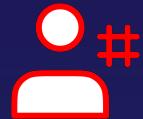
# Перегенерация сертификатов



При переносе реплики хранилище в новый кластер, соответствующее Alternative Name должно быть «зашито» в сертификат.

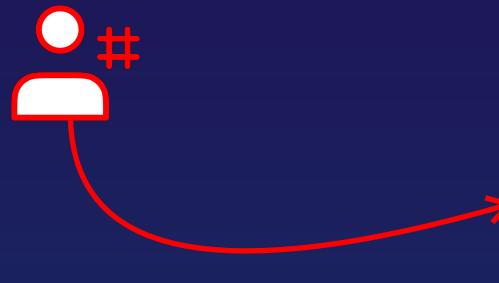
```
Certificate:
Data:
    Subject: ..., CN = redis1-rs-rs001
    ...
X509v3 extensions:
    X509v3 Subject Alternative Name:
        DNS:*.redis1-rs-rs001.redis1-rs-rs001.svc.first-cluster.k8s,
        DNS:*.redis1-rs-rs001.redis1-rs-rs001.svc.second-cluster.k8s,
        DNS:*.redis1-rs-rs001.redis1-rs-rs001.svc.third-cluster.k8s
Signature Algorithm: ...
```

# Перегенерация сертификатов



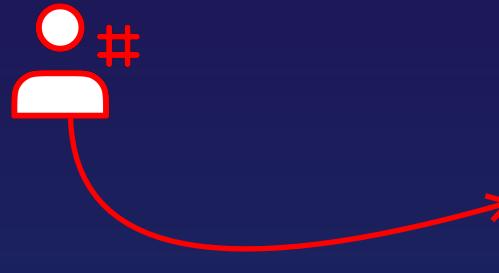
dbaas

# Перегенерация сертификатов



Изменение  
распределения  
реплик

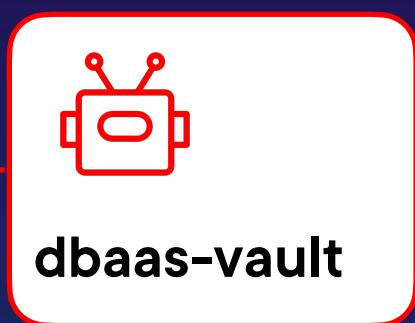
# Перегенерация сертификатов



Изменение  
распределения  
реплик

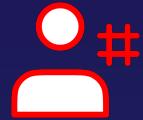
```
name: redis1
replica_sets:
- name: rs001
  subsets:
  - kubernetes_cluster: first-cluster
    replicas: 1
  - kubernetes_cluster: second-cluster
    replicas: 1
  - kubernetes_cluster: third-panel
    replicas: 1
modified_by: avknyazhev
```

# Перегенерация сертификатов

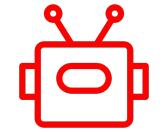


Изменено  
распределение  
реплик

# Перегенерация сертификатов



dbaaS



dbaaS-vault

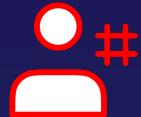
Запустить перегенерацию  
сертификатов

vault



vdc

# Перегенерация сертификатов

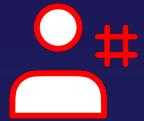


dbaas

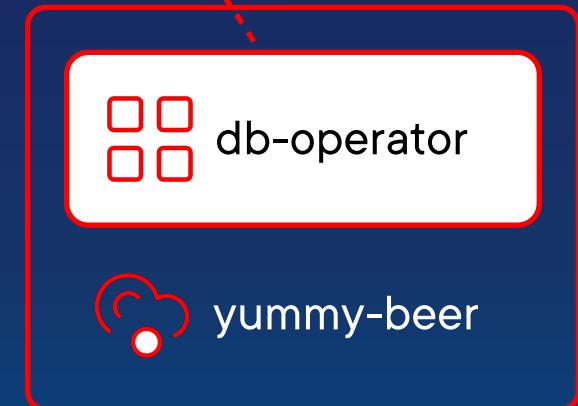
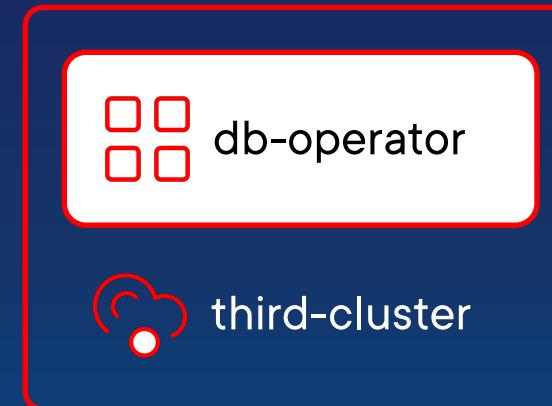
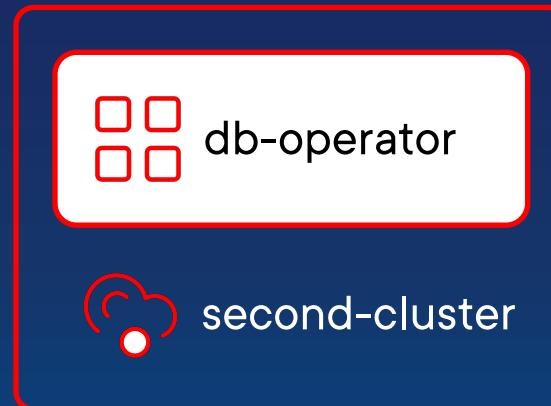
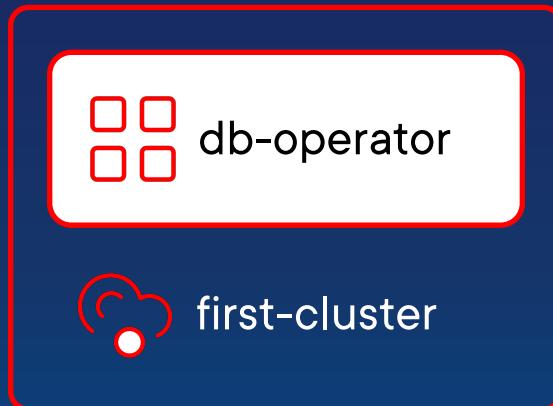


STORAGE_NAME	REPLICA_SET_NAME	SOFTWARE	ENVIRONMENT	K8S_CLUSTER	UPDATE_STATUS	READABLE	WRITABLE	HEALTHY
redis1	rs001	redis	staging	first-cluster	deletion-pending	1/1	0/1	1/1
redis1	rs001	redis	staging	second-cluster	pending	1/1	1/1	1/1
redis1	rs001	redis	staging	third-cluster	pending	1/1	0/1	1/1
redis1	rs001	redis	staging	yummy-beer	in-progress	1/1	0/1	1/1

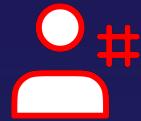
# Перегенерация сертификатов



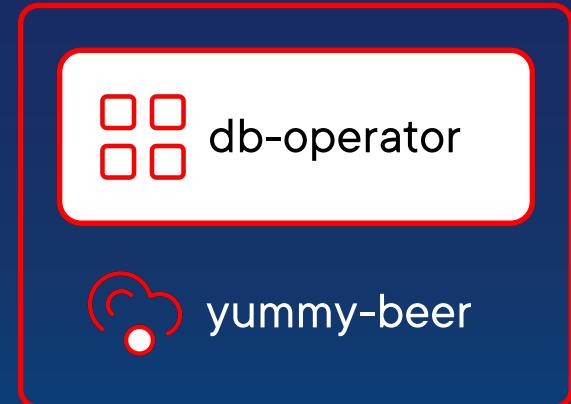
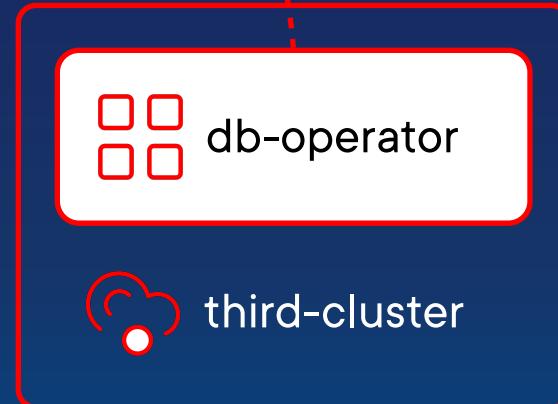
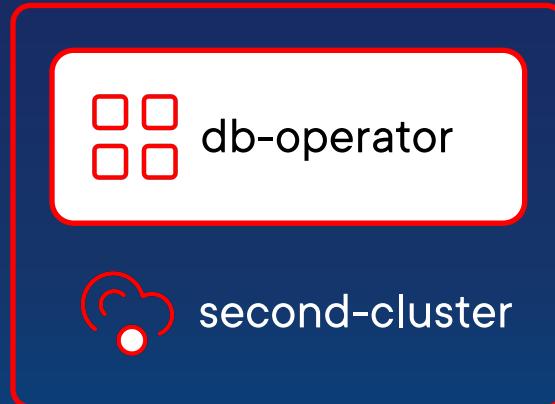
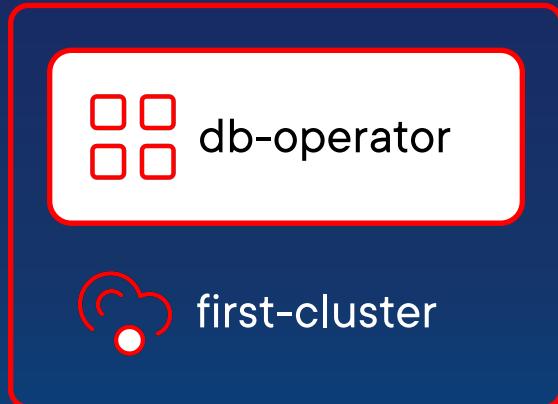
Развернуть новую реплику



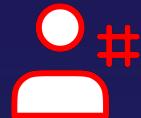
# Перегенерация сертификатов



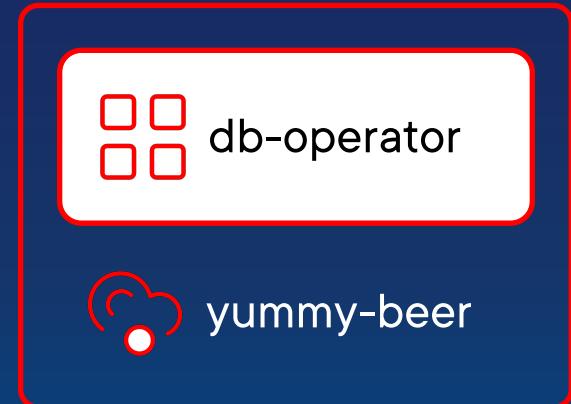
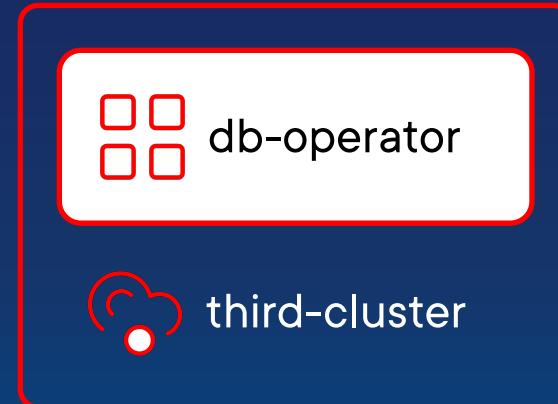
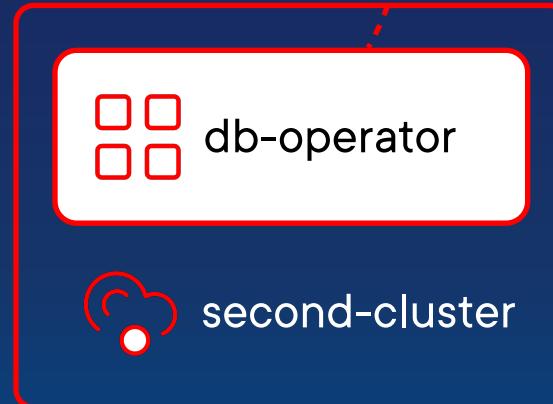
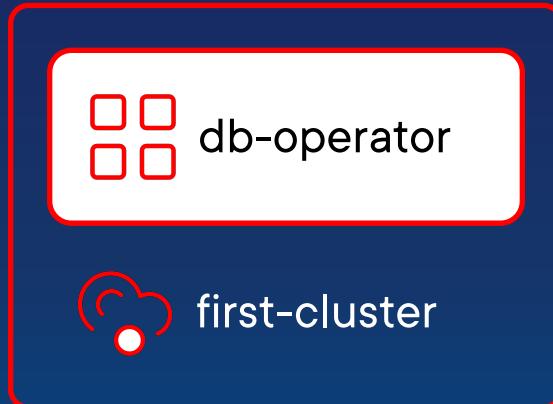
Применить изменения



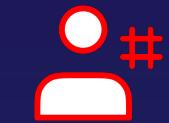
# Перегенерация сертификатов



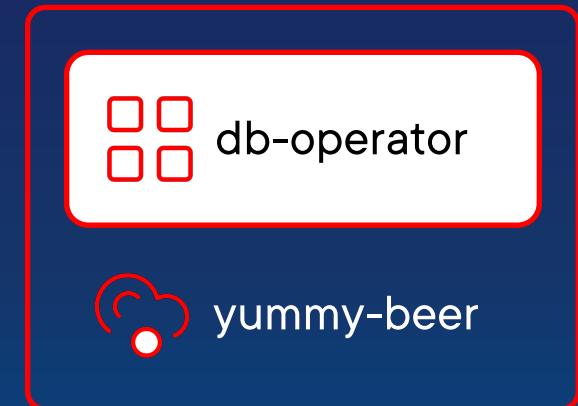
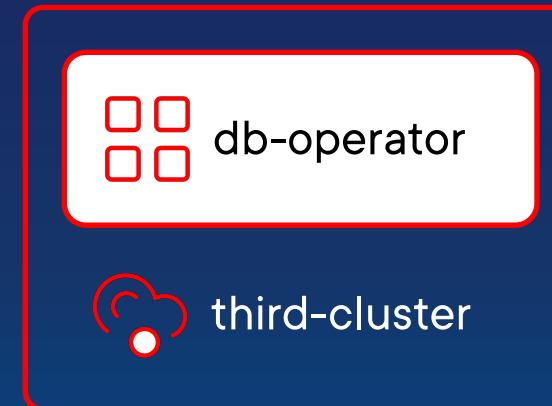
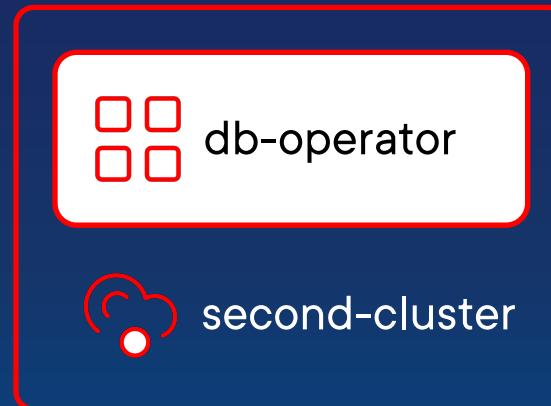
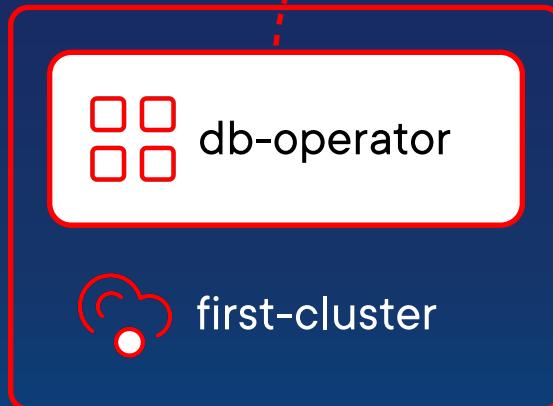
Применить изменения



# Перегенерация сертификатов



Удалить реплику



# Перегенерация сертификатов

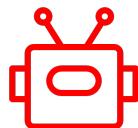


Как не допустить дальнейший ход Rolling Update, пока сертификаты не перегенерируются?

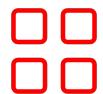
# Перегенерация сертификатов



Как не допустить дальнейший ход Rolling Update, пока сертификаты не перегенерируются?



dbaas



db-operator



yummy-beer



replica

FS

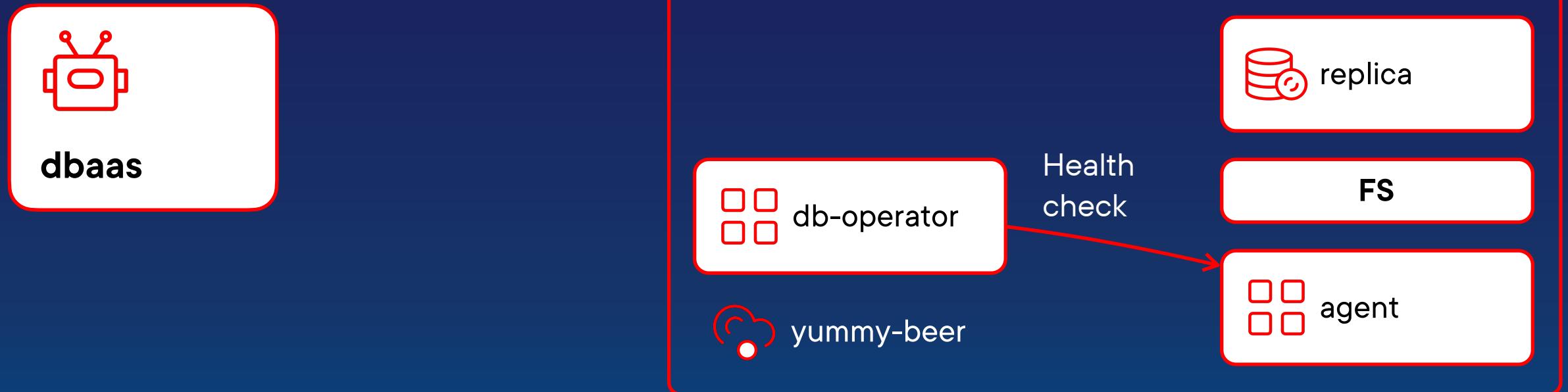


agent

# Перегенерация сертификатов



Как не допустить дальнейший ход Rolling Update, пока сертификаты не перегенерируются?



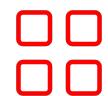
# Перегенерация сертификатов



Как не допустить дальнейший ход Rolling Update, пока сертификаты не перегенерируются?



dbaas



db-operator



yummy-beer

Проверить  
сертификат



replica

FS

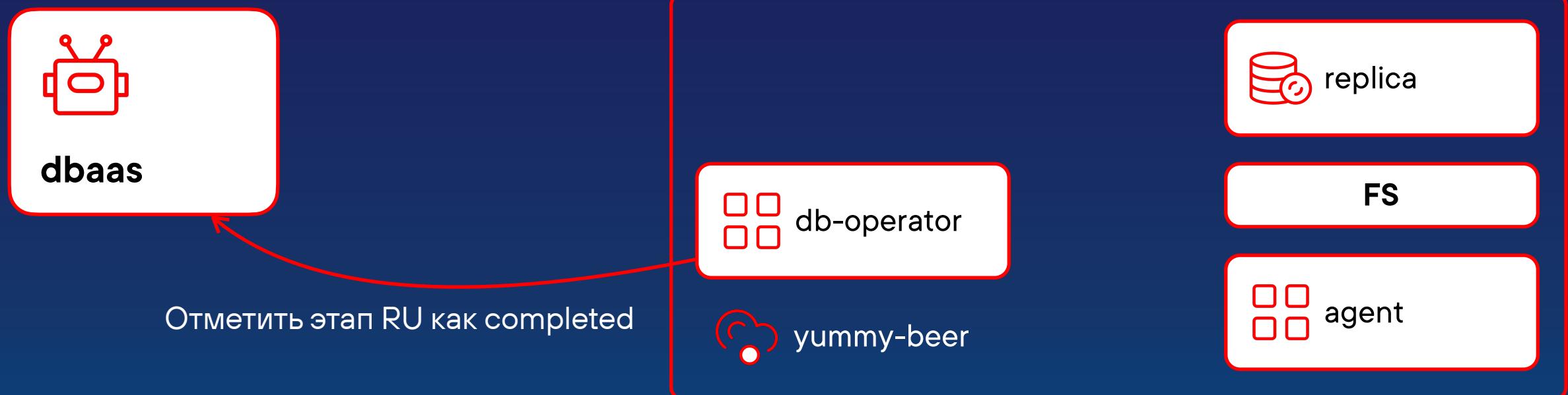


agent

# Перегенерация сертификатов



Как не допустить дальнейший ход Rolling Update, пока сертификаты не перегенерируются?



# Результаты



- 👉 Инструкция по приемке кластера: 18 пунктов → 2.  
Время приемки: 1/3 человека-спринта → почти 0.
- 🚑 Максимально снизили желание дежурного пойти  
к психиатру (но это неточно).
- 📝 Исключили возможные ошибки неверной копипасты манифестов.
- 🃏 Покрыли автоматизированными проверками случаи, которые не  
покрывались ручными проверками.

 Базы данных в Kubernetes это не страшно!

В Kubernetes есть достаточно крутых примитивов для работы со stateful-приложениями.

 Kubernetes — топчик даже для десятков тысяч баз данных.

 ArgoCD — топчик.

 Работать с сертификатами — не топчик.

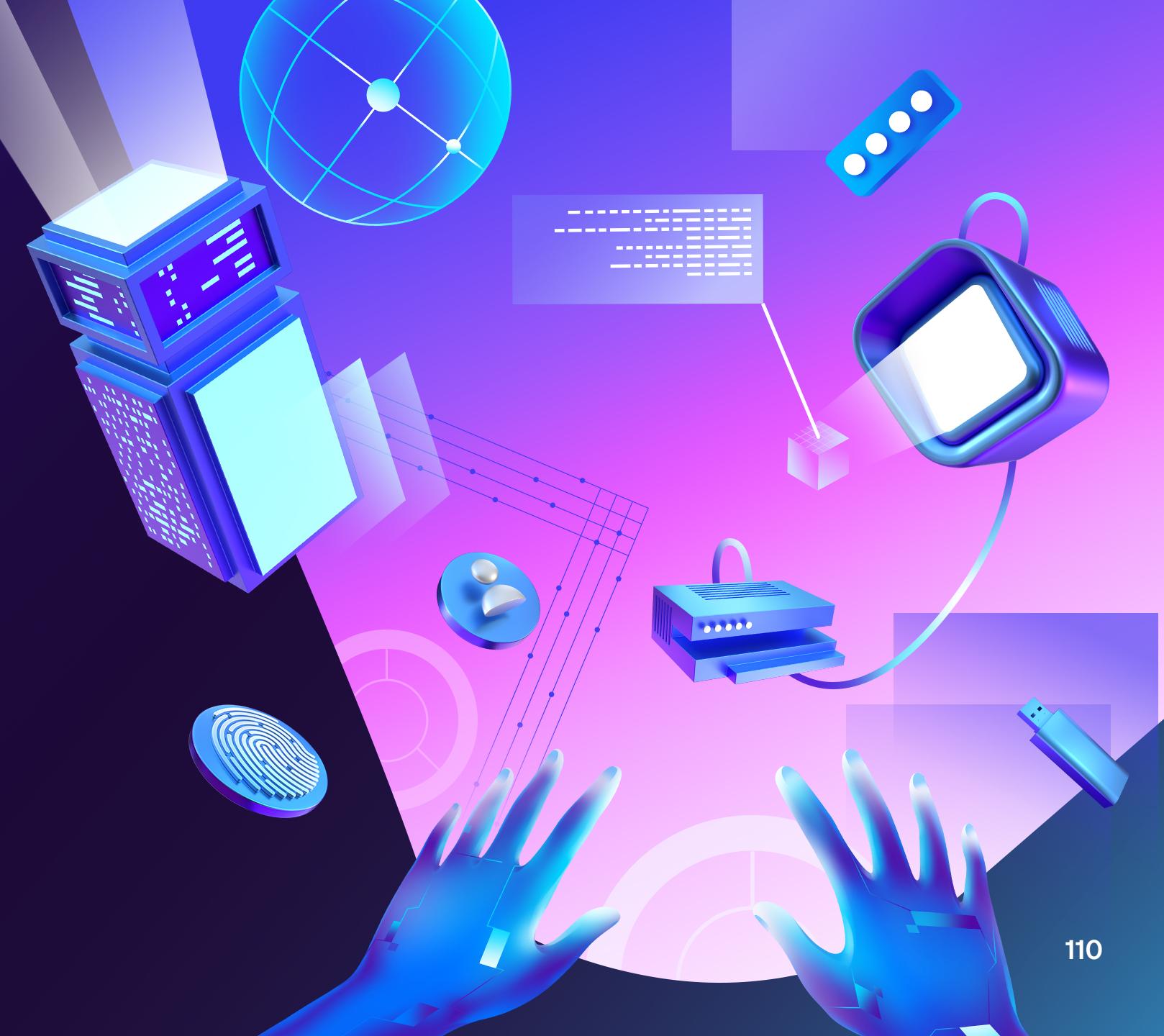
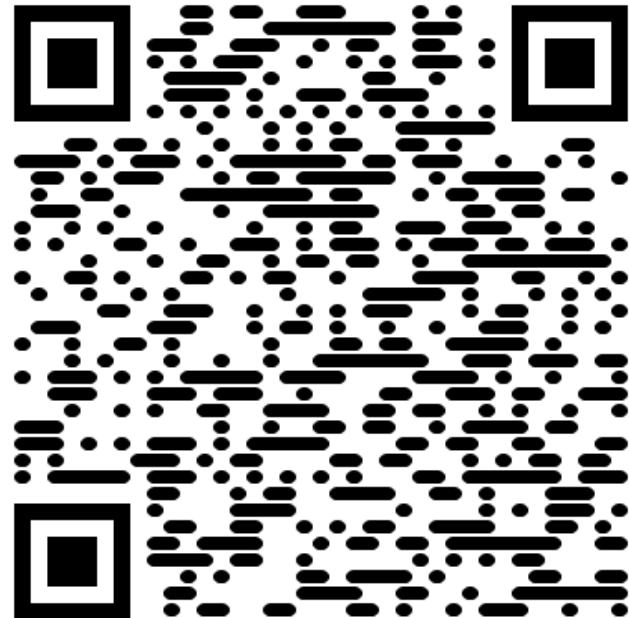
 Много готовых решений.

 Всегда можно сделать самим.

Задавайте вопросы

Telegram: @muhomorfus

Email: avknyazhev@avito.ru



Спасибо!



Оцените доклад

