

Как достать шумного соседа: ограничение дискового ввода-вывода в *Kubernetes*

Алексей Княжев
Инженер, Авито @ DBaaS

Алексей Княжев @muhomorfus

Инженер в команде DBaaS Авито

В Авито с октября 2023 года: тогда же переобулся из продуктового разработчика в платформенного инженера. Занимаюсь разработкой платформы DBaaS. Интересуюсь Kubernetes, базами данных, ПИВОМ.



План доклада



01

phd 2X 

Платформа **DBaaS**

Что такое DBaaS?

Что из себя представляет платформа DBaaS в Авито сейчас?

DBaaS — Database as a Service



DBaaS — это

Базы данных как услуга.



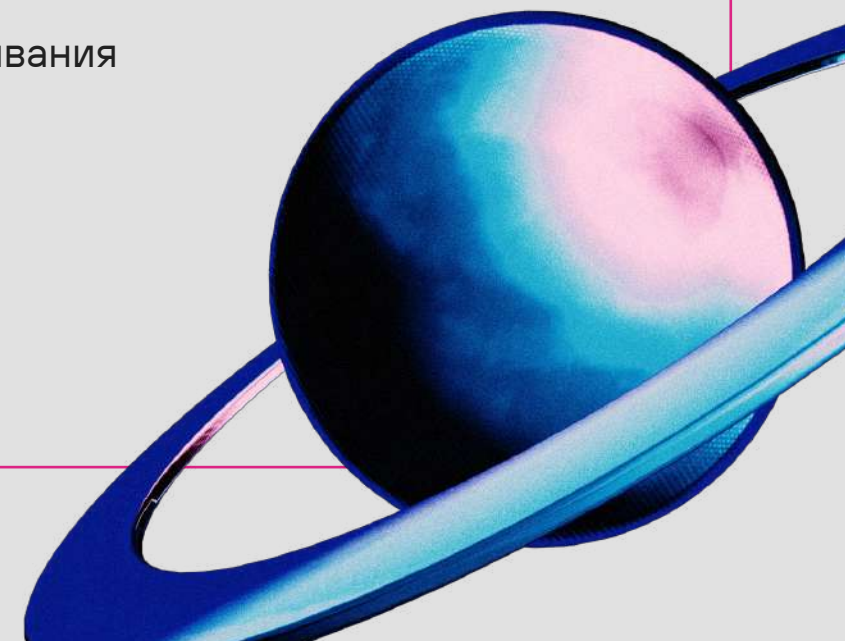
DBaaS — это

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



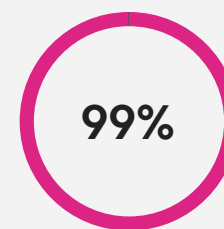
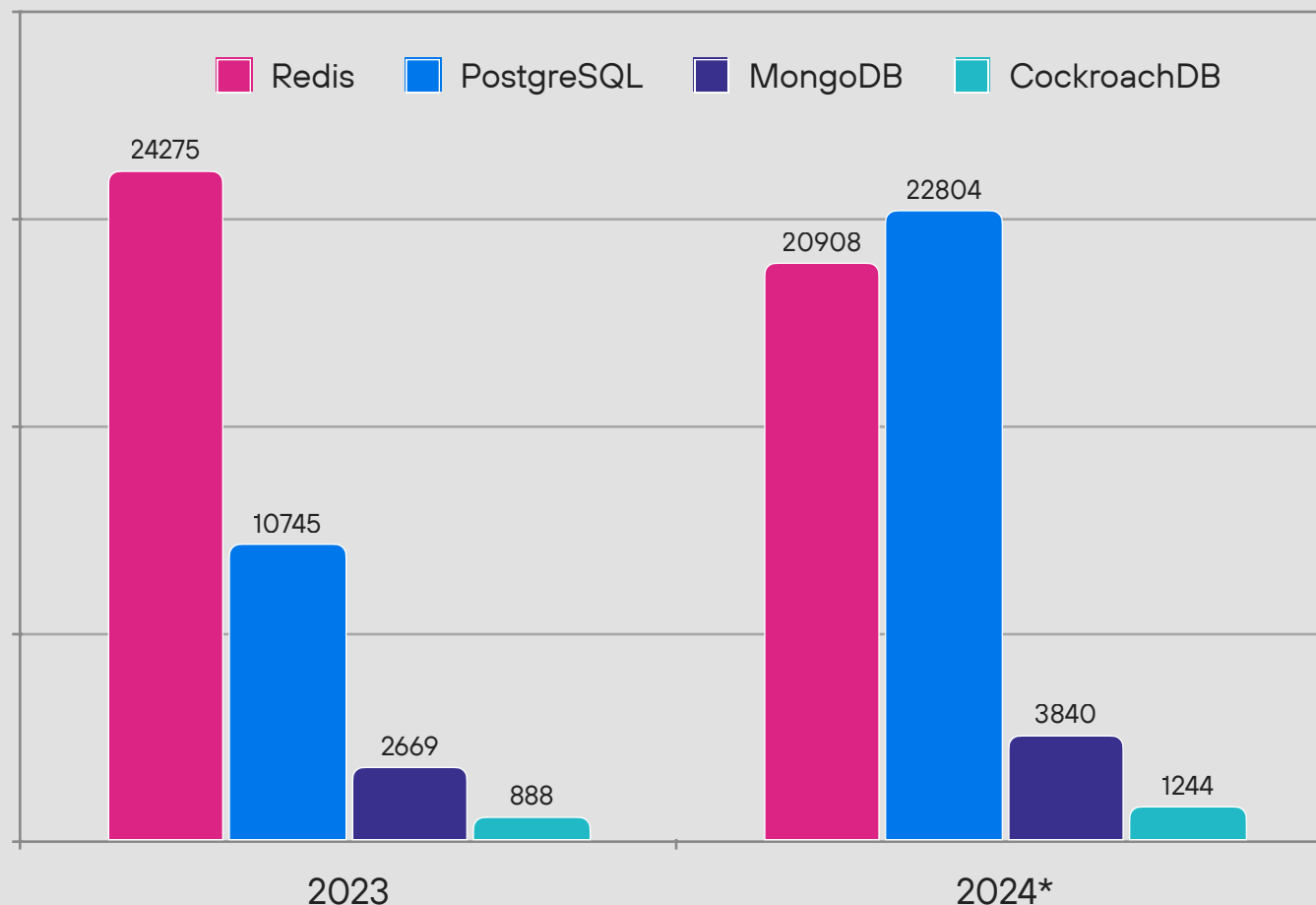
DBaaS — это

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



Статистика платформы *DBaaS*

Количество создаваемых хранилищ

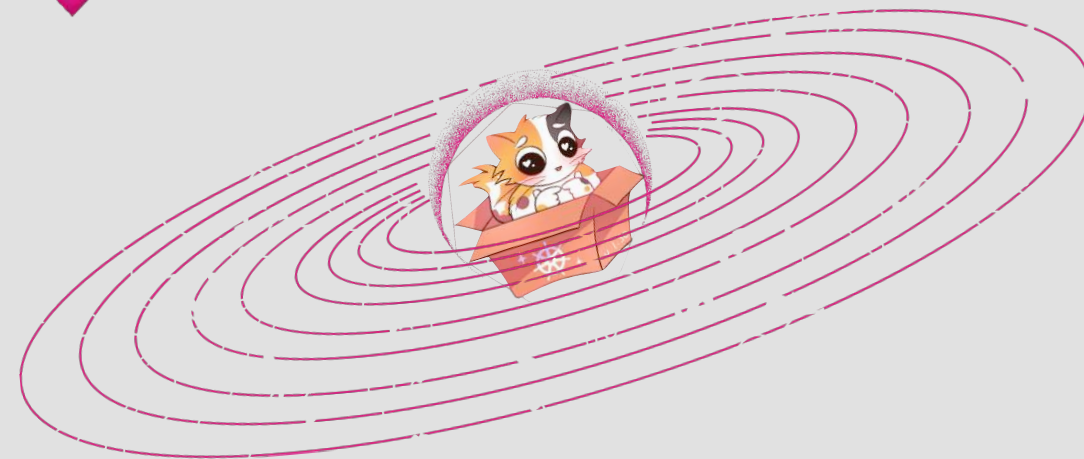


**Сокращено время
создания новой БД**
С 3-7 дней до 2-5 минут.

* Использована линейная экстраполяция
на основе данных первого квартала 2024.

DBaaS + *Kubernetes* = ❤️

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



Автоматизация запуска приложений

Простота обслуживания железа

Наличие готовых решений в opensource

Независимость от окружения

Больше про DBaaS в Авито



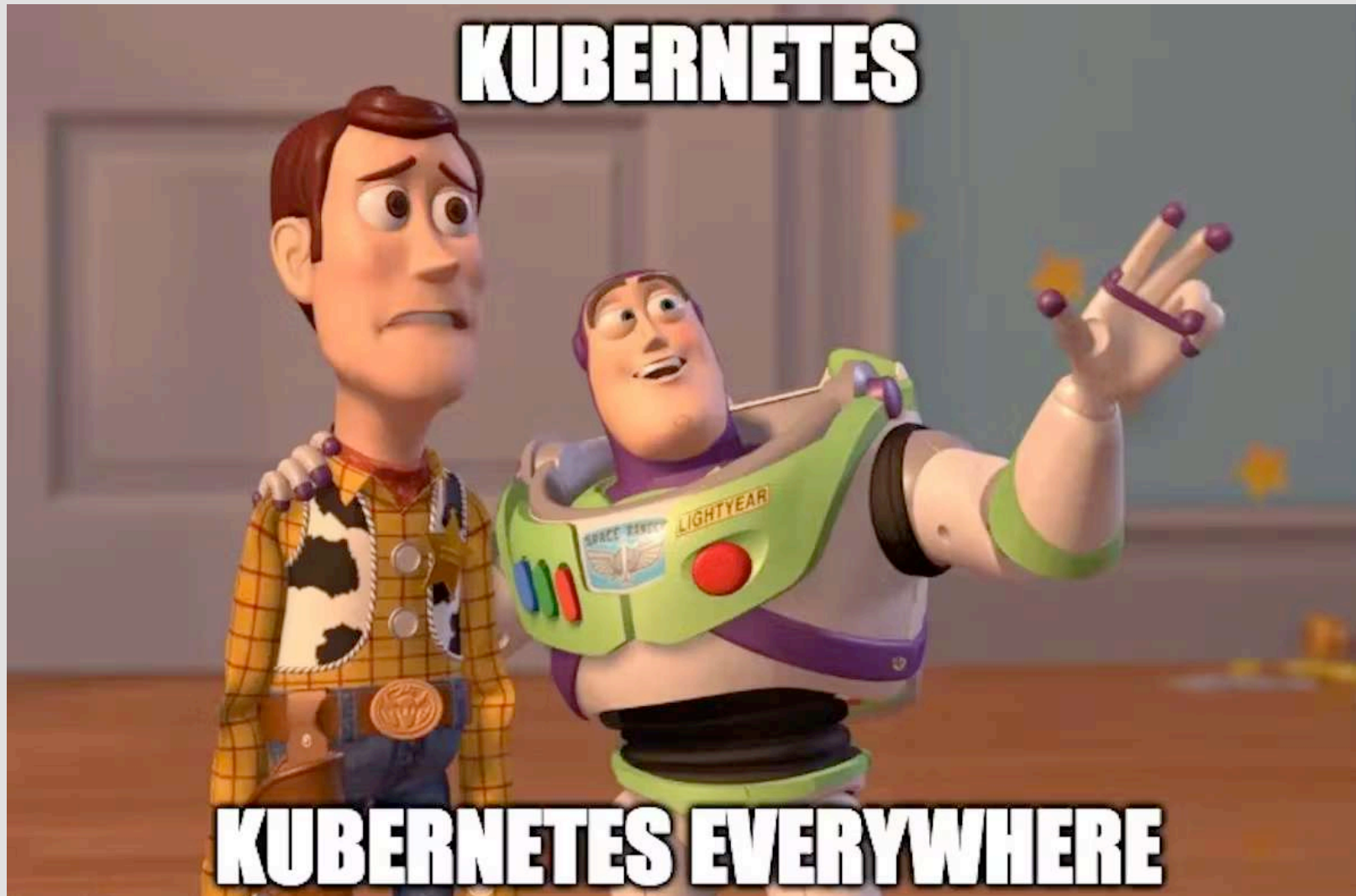
<https://clc.to/kchUxA>

Avito Database Meetup #1

- Что такое платформа DBaaS в Авито и зачем она нужна.
- Работа со Stateful-приложениями в Kubernetes.
- Паттерны управления базами данных в multi-cluster Kubernetes-среде.
- Особенности адаптации классических баз данных в платформу.

Kubernetes

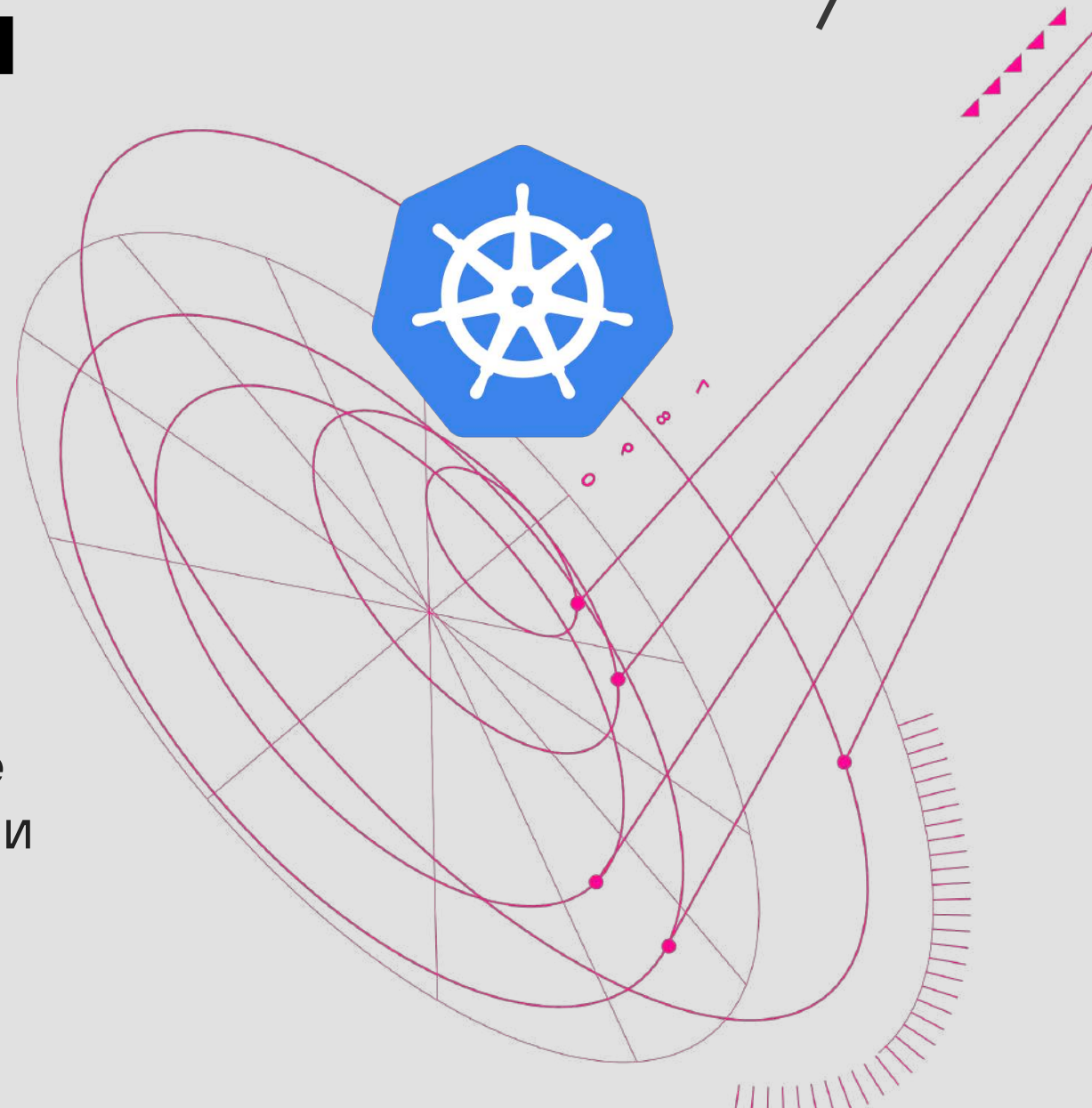
Что такое Kubernetes? Как хранить данные персистентно в приложениях, развёрнутых в Kubernetes?



Да кто такой ЭТОТ ваш *Kubernetes*?

Kubernetes (K8s) — это открытое программное обеспечение для автоматизации развёртывания, масштабирования и управления контейнеризированными приложениями.

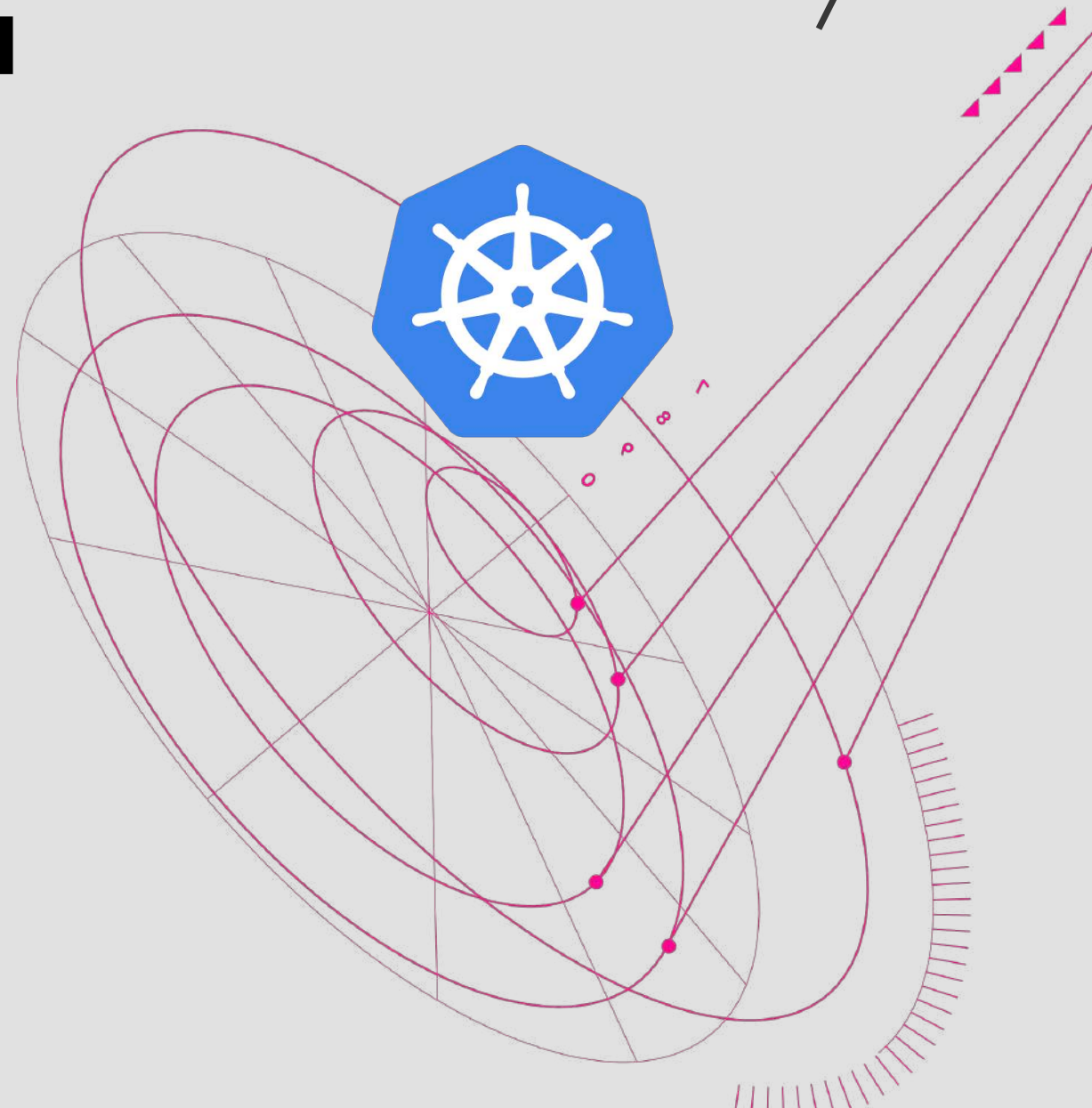
Kubernetes группирует контейнеры, составляющие приложение, в логические единицы для более простого управления и обнаружения. © kubernetes.io



Да кто такой ЭТОТ ваш *Kubernetes*?



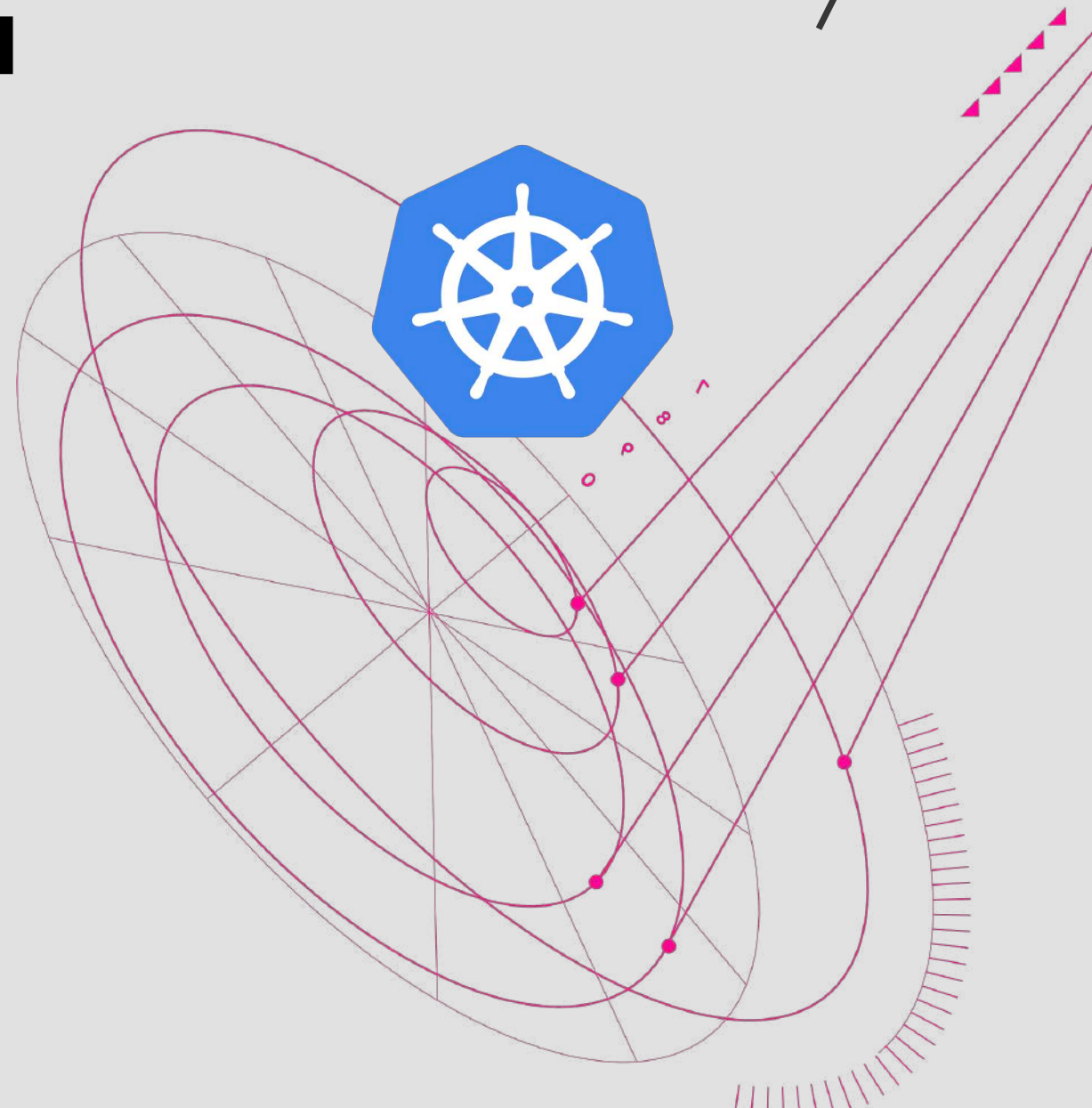
Предполагаю, что вы знаете базовые концепции Kubernetes, такие как Pod, Deployment, etc.



Да кто такой ЭТОТ ваш *Kubernetes*?

! Предполагаю, что вы знаете базовые концепции Kubernetes, такие как Pod, Deployment, etc.

! В данном разделе мы рассмотрим PersistentVolume, PersistentVolumeClaim, Dynamic Provisioning, CSI и DaemonSet.



PersistentVolume

Ресурс, описывающий том персистентного хранилища: например, область диска на ноде.

PersistentVolume

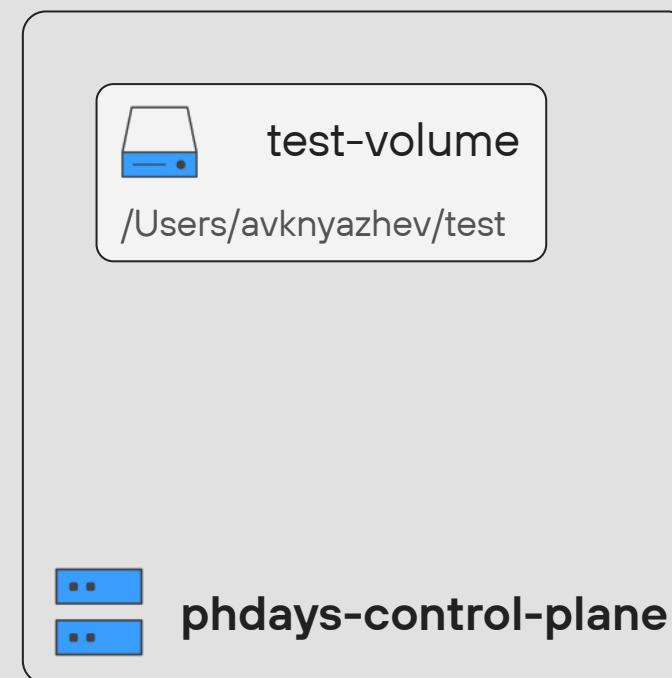
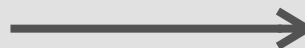
Ресурс, описывающий том персистентного хранилища: например, область диска на ноде.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: test-volume
spec:
  capacity:
    storage: 1G
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  storageClassName: manual
  hostPath:
    path: "/Users/avknyazhev/test"
```

PersistentVolume

Ресурс, описывающий том персистентного хранилища: например, область диска на ноде.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: test-volume
spec:
  capacity:
    storage: 1G
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  storageClassName: manual
  hostPath:
    path: "/Users/avknyazhev/test"
```




PersistentVolume

Ресурс, описывающий том персистентного хранилища: например, область диска на ноде.

```
$ kubectl get persistentvolumes
```

| NAME | CAPACITY | ACCESS MODES | RECLAIM POLICY | STATUS | AGE |
|-------------|----------|--------------|----------------|-----------|-----|
| test-volume | 1G | RWO | Retain | Available | 33s |



test-volume

/Users/avknyazhev/test



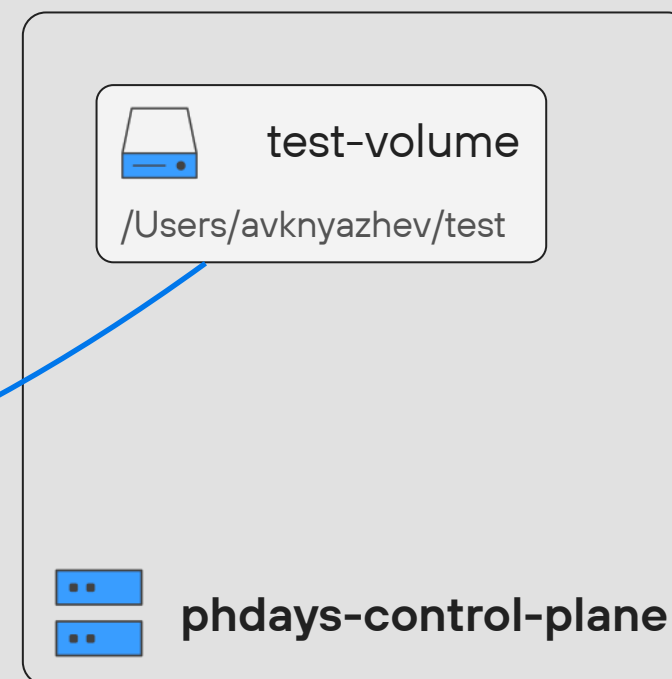
phdays-control-plane

PersistentVolume

Ресурс, описывающий том персистентного хранилища: например, область диска на ноде.

```
$ kubectl get persistentvolumes
```

| NAME | CAPACITY | ACCESS MODES | RECLAIM POLICY | STATUS | AGE |
|-------------|----------|--------------|----------------|-----------|-----|
| test-volume | 1G | RWO | Retain | Available | 33s |



Ok And?



© tenor.com

PersistentVolumeClaim

Ресурс, описывающий пользовательский запрос на хранилище.

PersistentVolumeClaim

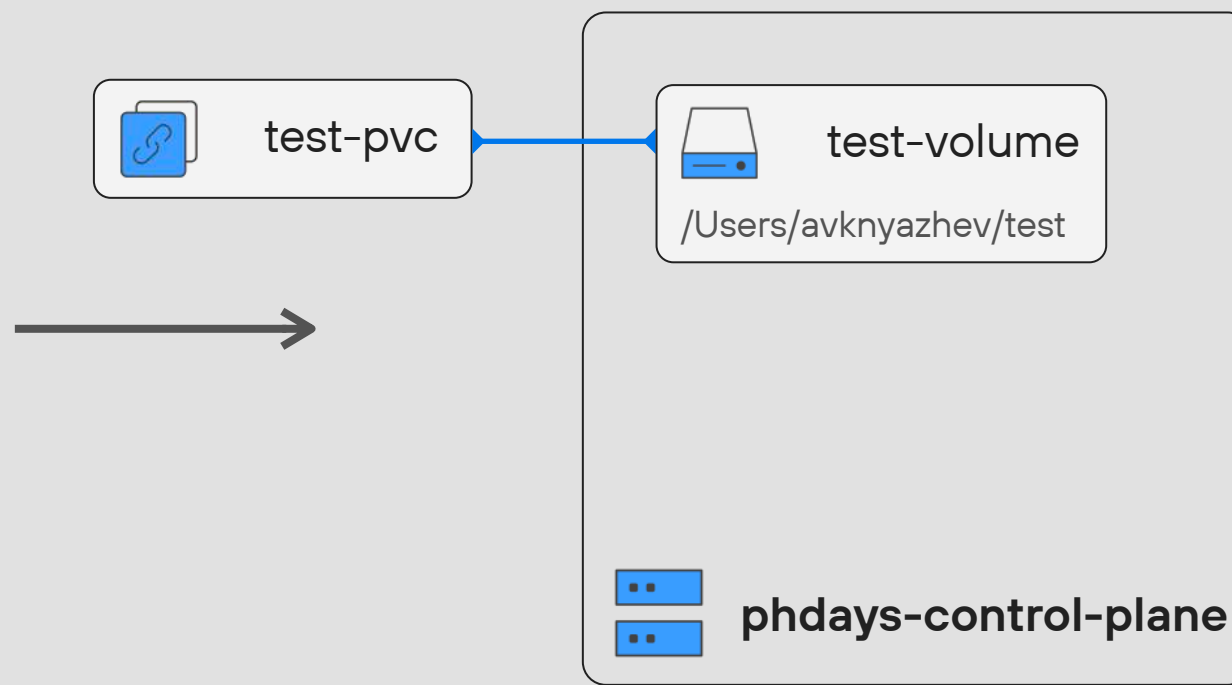
Ресурс, описывающий пользовательский запрос на хранилище.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: test-pvc
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 1G
  storageClassName: manual
```

PersistentVolumeClaim

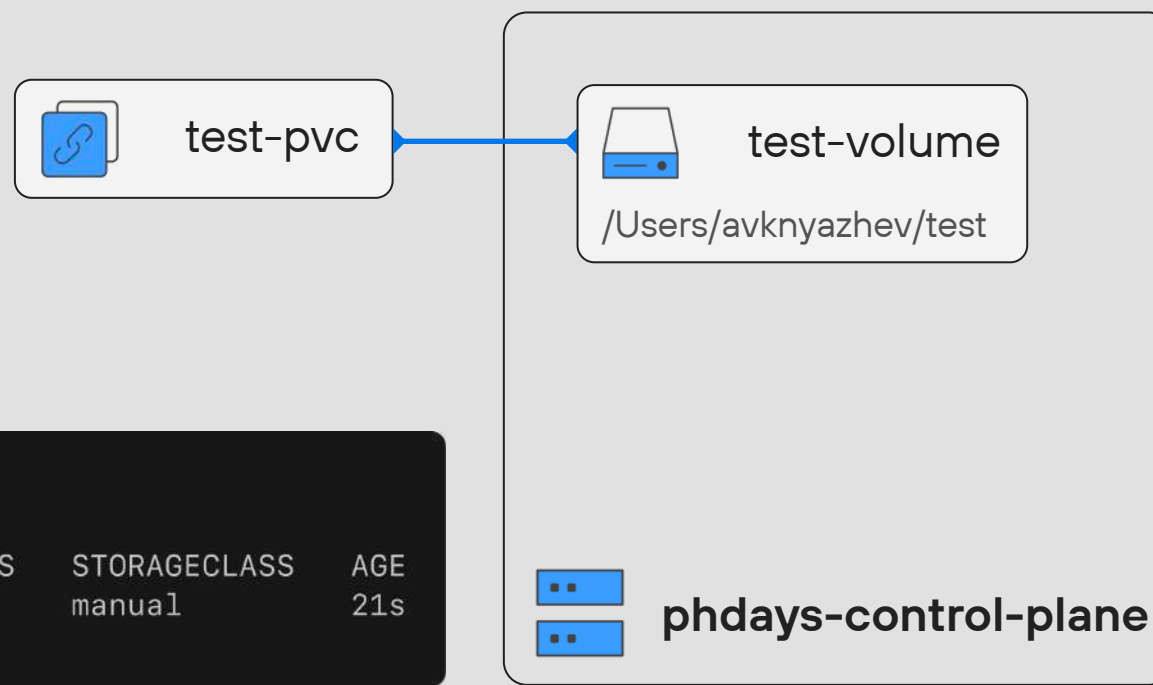
Ресурс, описывающий пользовательский запрос на хранилище.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: test-pvc
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 1G
  storageClassName: manual
```



PersistentVolumeClaim

Ресурс, описывающий пользовательский запрос на хранилище.



```
$ kubectl get persistentvolumeclaims
```

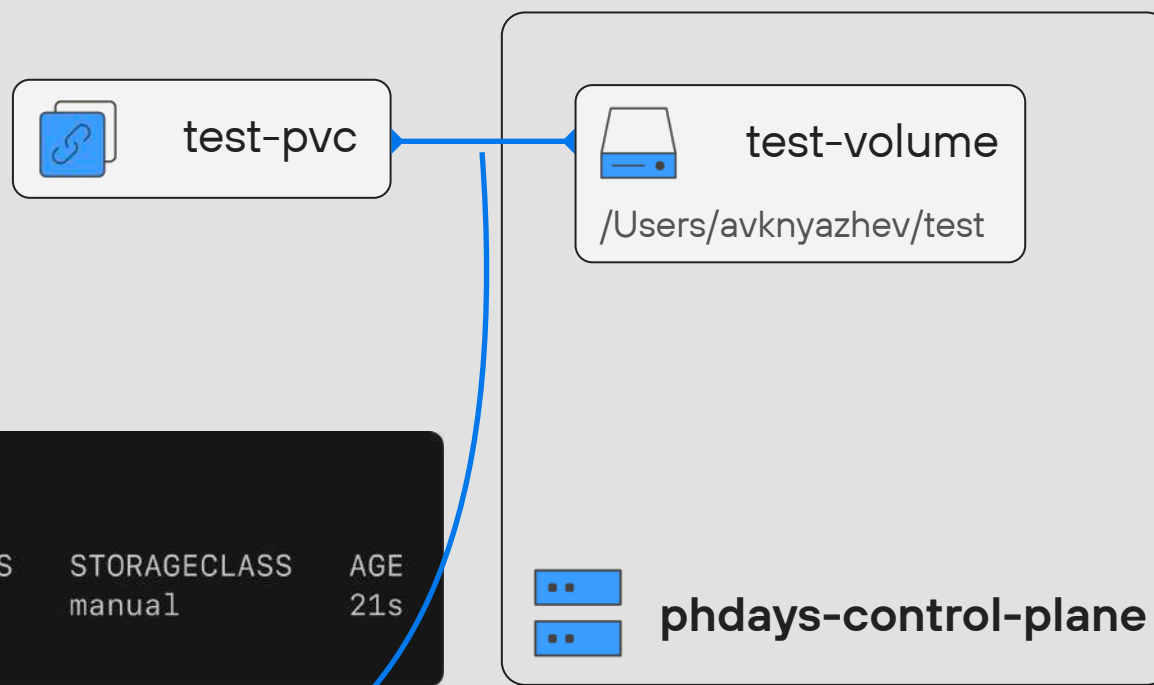
| NAME | STATUS | VOLUME | CAPACITY | ACCESS MODES | STORAGECLASS | AGE |
|----------|--------|-------------|----------|--------------|--------------|-----|
| test-pvc | Bound | test-volume | 1G | RWO | manual | 21s |

PersistentVolumeClaim

Ресурс, описывающий пользовательский запрос на хранилище.

```
$ kubectl get persistentvolumeclaims
```

| NAME | STATUS | VOLUME | CAPACITY | ACCESS MODES | STORAGECLASS | AGE |
|----------|--------|-------------|----------|--------------|--------------|-----|
| test-pvc | Bound | test-volume | 1G | RWO | manual | 21s |

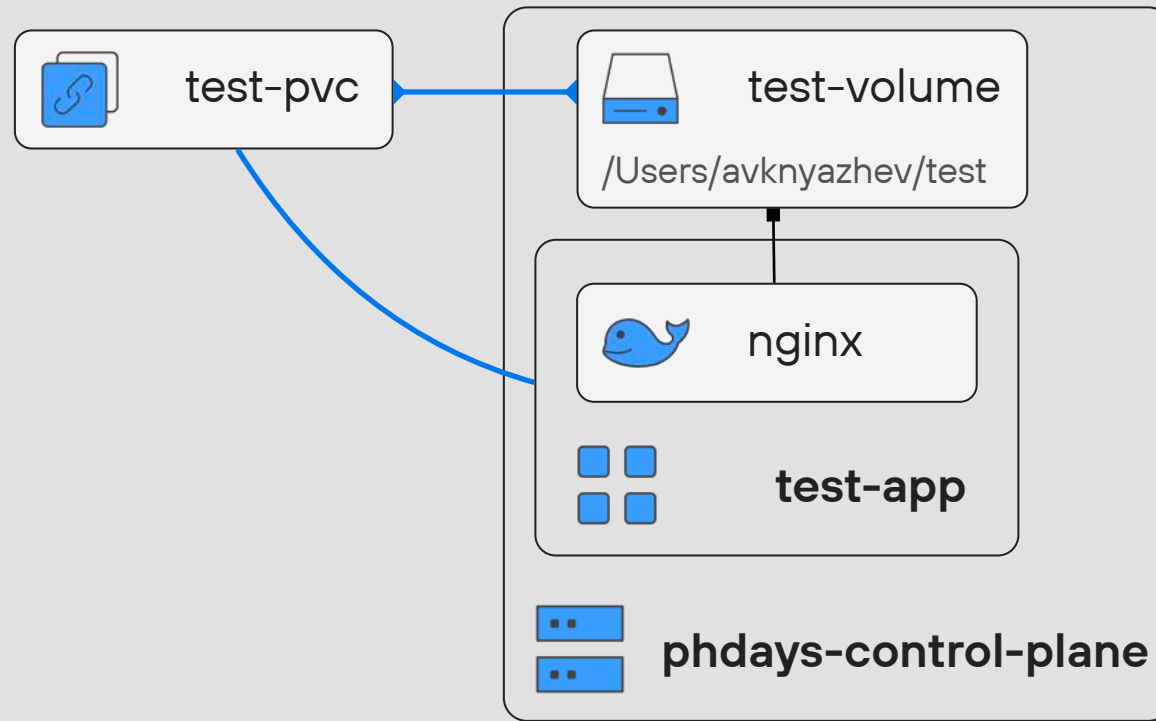


Вернёмся к Pod-у

```
apiVersion: v1
kind: Pod
metadata:
  name: test-app
spec:
  containers:
    - name: nginx
      image: nginx:1.14.2
      volumeMounts:
        - mountPath: "/var/www/html"
          name: static
  volumes:
    - name: static
      persistentVolumeClaim:
        claimName: test-pvc
```

Вернёмся к Pod-у

```
apiVersion: v1
kind: Pod
metadata:
  name: test-app
spec:
  containers:
    - name: nginx
      image: nginx:1.14.2
      volumeMounts:
        - mountPath: "/var/www/html"
          name: static
  volumes:
    - name: static
      persistentVolumeClaim:
        claimName: test-pvc
```



Static provisioning

Описанная схема представляет собой static provisioning томов — PersistentVolumes выделяются вручную администратором кластера.

Static provisioning

Описанная схема представляет собой static provisioning томов — PersistentVolumes выделяются вручную администратором кластера.

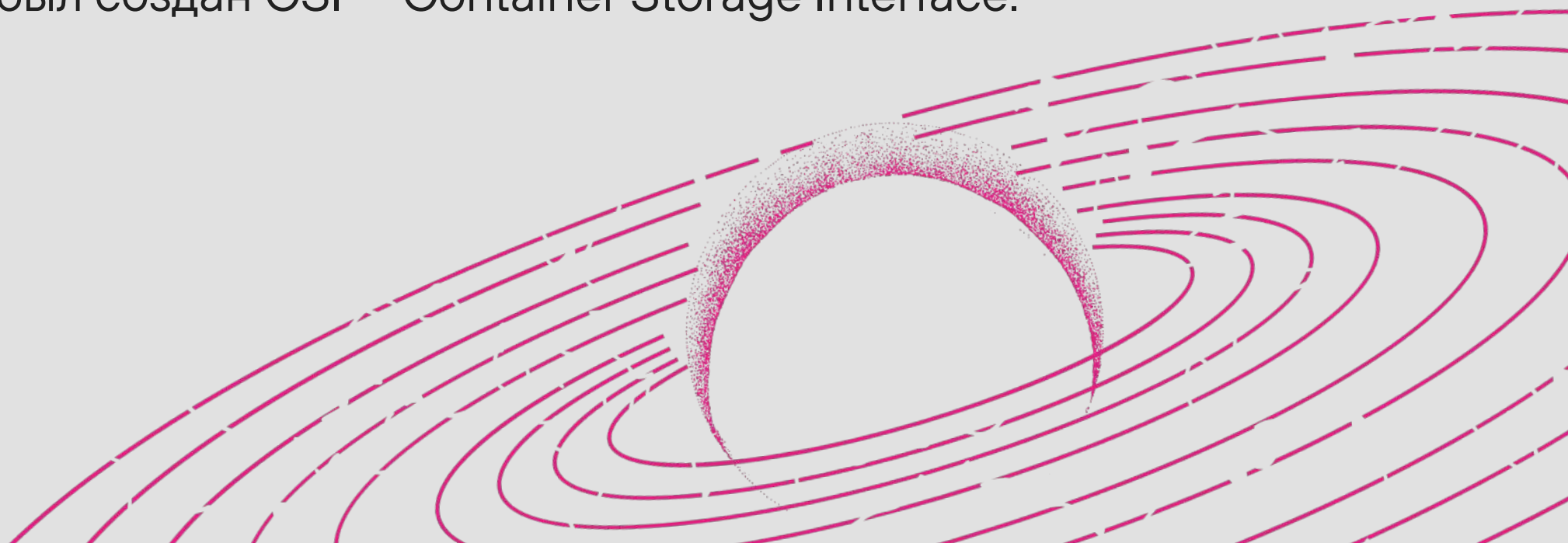


© meme-arsenal.ru

Dynamic provisioning. CSI

k8s позволяет создавать PersistentVolumes динамически. В этом случае PV создаётся автоматикой на основе PersistentVolumeClaim.

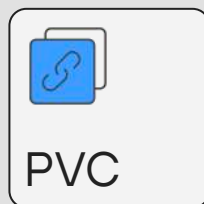
Чтобы не держать в кодовой базе функционал выделения томов для разных провайдеров, был создан CSI — Container Storage Interface.



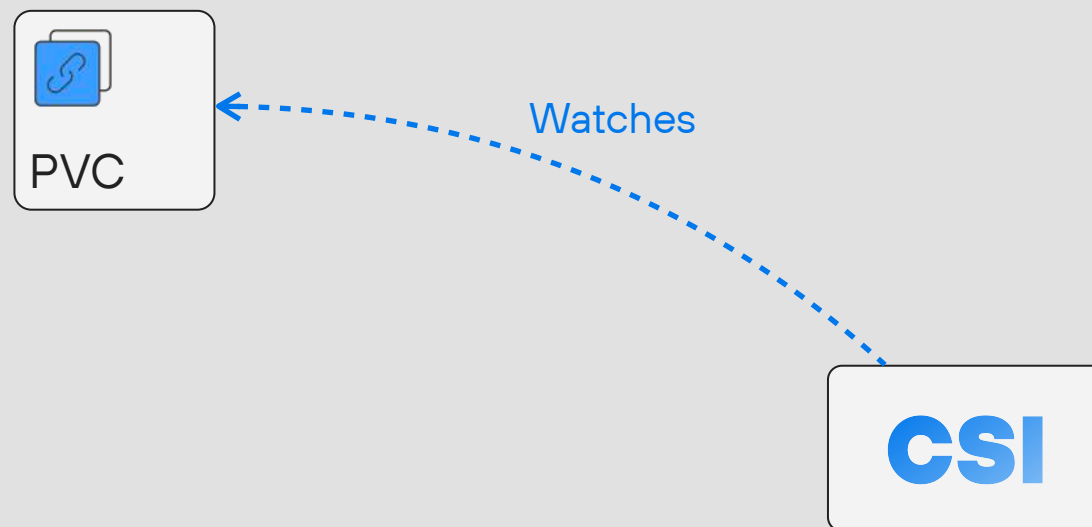
Dynamic provisioning. CSI



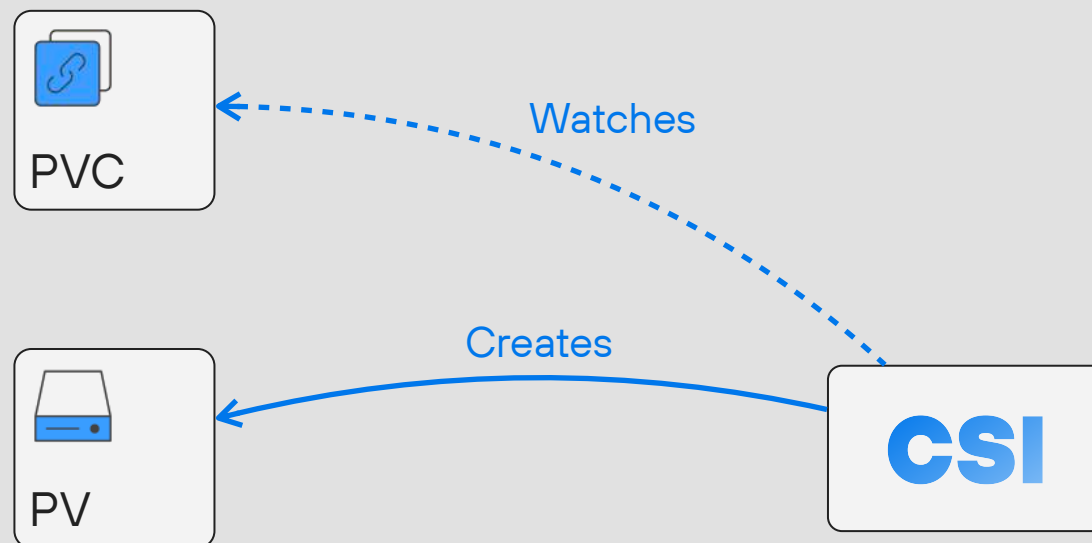
Dynamic provisioning. CSI



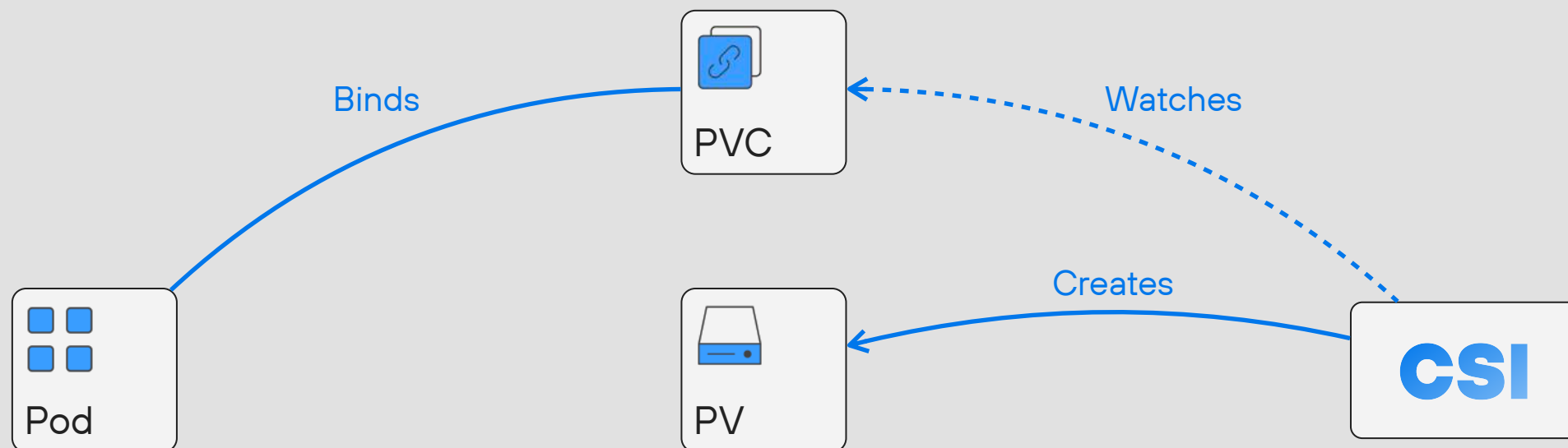
Dynamic provisioning. CSI



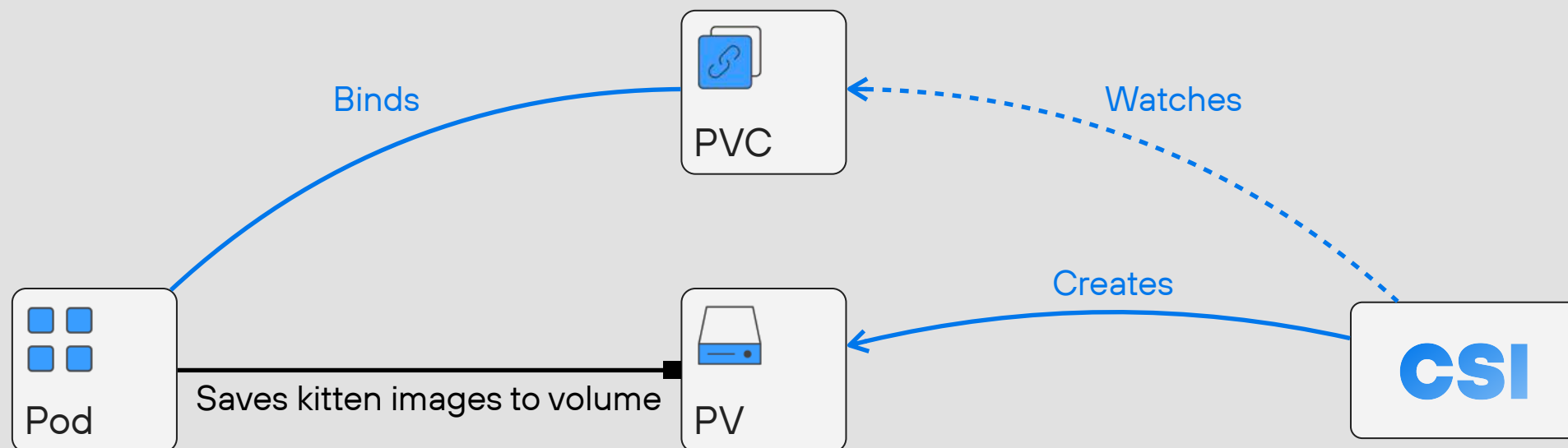
Dynamic provisioning. CSI



Dynamic provisioning. CSI



Dynamic provisioning. CSI



TopoLVM

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

- / Создаёт PV с использованием LVM (Logical Volume Manager).
- / Выделяет диск на воркер-ноде.
- / Позволяет абстрагироваться от физической организации хранения данных.
- / Поддерживает dynamic provisioning и volume expansion без даунтайма.
- / Изолирует Volumes на уровне блочных устройств.
- / Умеет в интеграцию с планировщиком Kubernetes, учитывая объем свободного диска.

DaemonSet

Абстракция, описывающая приложение, копии которого нужно запустить на каждой ноде кластера.

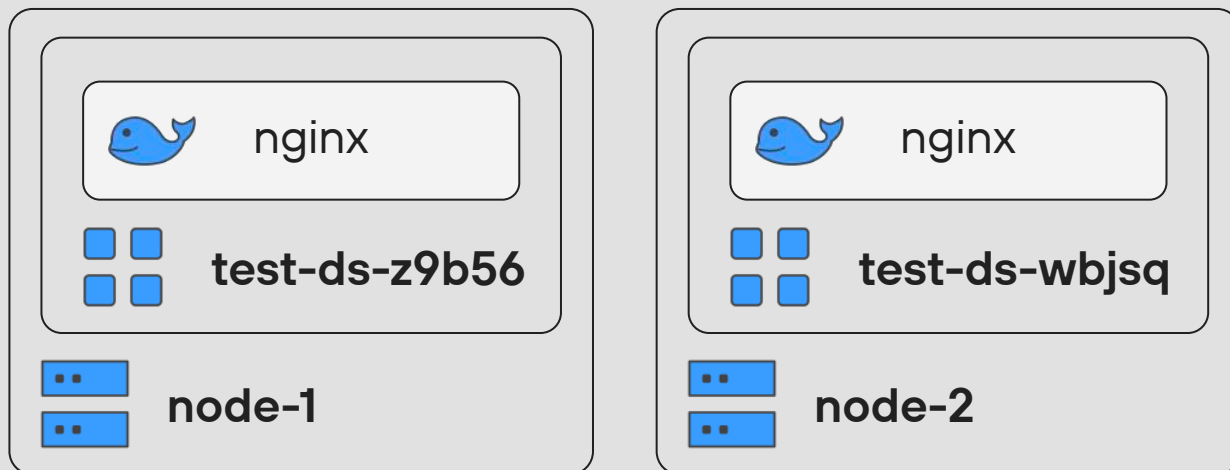
DaemonSet

Абстракция, описывающая приложение, копии которого нужно запустить на каждой ноде кластера.

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: test-ds
  labels:
    app: test
spec:
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
```

DaemonSet

Абстракция, описывающая приложение, копии которого нужно запустить на каждой ноде кластера.

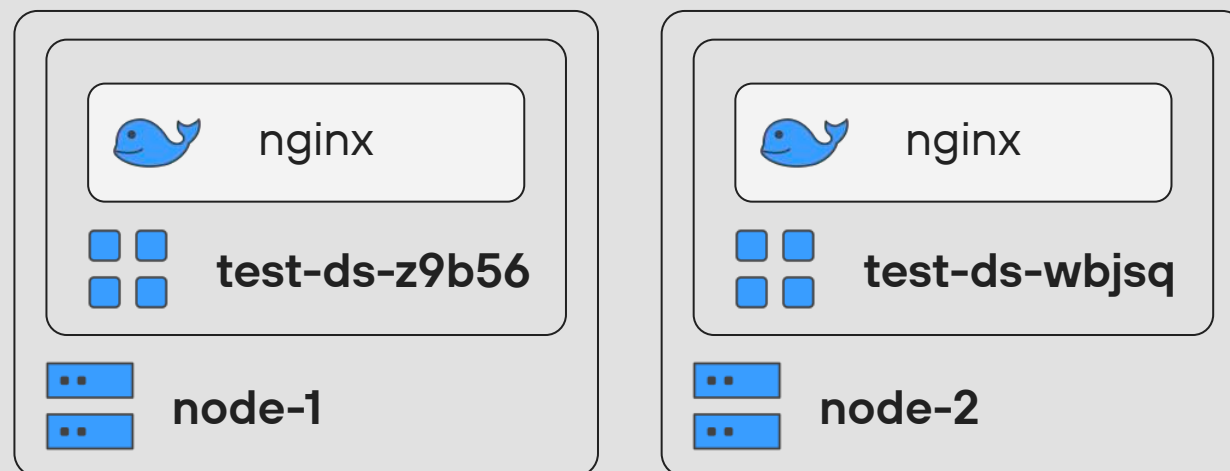


```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: test-ds
  labels:
    app: test
spec:
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
```

DaemonSet

```
$ kubectl get pods
```

| NAME | READY | STATUS | RESTARTS | AGE |
|---------------|-------|---------|----------|-----|
| test-ds-wbjsq | 1/1 | Running | 0 | 30s |
| test-ds-z9b56 | 1/1 | Running | 0 | 30s |



```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: test-ds
  labels:
    app: test
spec:
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
```

03

phd 2X pt



Предпосылки

Зачем ограничивать дисковый ввод-вывод в Kubernetes?

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.



А что *Kubernetes*?

Kubernetes «из коробки» умеет в ограничения CPU и Memory. Поддержки ограничения дискового ввода-вывода и сети пока нет. 🙄

В нашем случае, отсутствие ограничений дискового ввода-вывода препятствует production-эксплуатации PostgreSQL в Kubernetes.

/ В Kubernetes 1.29 добавили alpha-версию VolumeAttributeClass API. С помощью него можно объявлять классы хранилищ, определяя такие характеристики, как IOPS и Bandwidth. Но данное API находится в alpha-версии. Кроме того, механизм требует поддержки на стороне провайдера CSI, не сильно предназначен для назначения индивидуальных лимитов под каждый запущенный экземпляр приложения и ограниченно поддерживает изменение ограничений.

04

phd 2x pt

Реализация

Как мы в итоге реализовали ограничение дискового ввода-вывода?





© youtube.com

+



+



Нужно больше операторов!

Для ограничения дискового ввода-вывода мы разработали оператор. Он запущен как DaemonSet и применяет ограничения на своей ноде.

ЮВА

Нужно больше операторов!

Для ограничения дискового ввода-вывода мы разработали оператор. Он запущен как DaemonSet и применяет ограничения на своей ноде.

I O V A

Нужно больше операторов!

Для ограничения дискового ввода-вывода мы разработали оператор. Он запущен как DaemonSet и применяет ограничения на своей ноде.

Input/**O**utput **B**andwidth **A**djuster

Операторы в K8s

- / Представляют собой программные расширения к K8s, не изменяющие его код.
- / Реализованы в виде отдельных приложений.
- / Реализуют такие принципы работы k8s, как control loop.
- / Выполняют роль контроллеров для custom resources.

/ Контроллер отслеживает ресурс K8s, в поле spec которого содержится желаемое состояние (desired state, in English) объекта. На основании желаемого состояния объекта контроллер должен привести фактическое состояние как можно ближе к желаемому.

Операторы в k8s

Функция *Reconcile* реализует *control loop*,
вызывается на любом изменении
отслеживаемого ресурса.

```
func (r *Reconciler) Reconcile(  
    ctx context.Context,  
    req ctrl.Request,  
) (ctrl.Result, error) {  
    var limit dbaasv1alpha1.IOLimit  
    err := r.k8s.Get(ctx, req.NamespacedName, &limit)  
    switch {  
    case apierrors.IsNotFound(err):  
        return ctrl.Result{}, nil  
    case err != nil:  
        return ctrl.Result{}, err  
    }  
  
    // Do some cool logic here.  
  
    return ctrl.Result{}, nil  
}
```

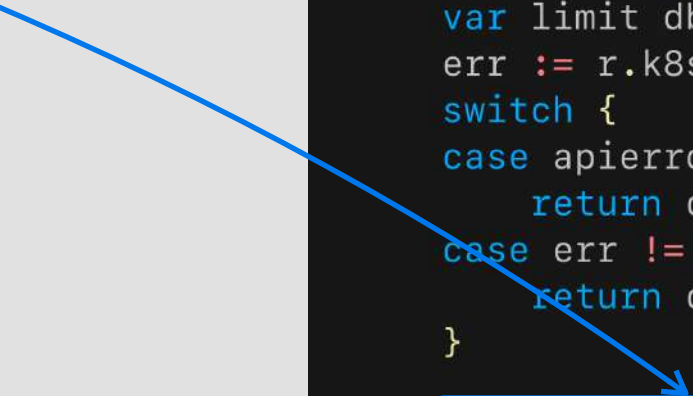
Операторы в k8s

Получение желаемого состояния объекта.

```
func (r *Reconciler) Reconcile(  
    ctx context.Context,  
    req ctrl.Request,  
) (ctrl.Result, error) {  
    var limit dbaasv1alpha1.IOLimit  
    err := r.k8s.Get(ctx, req.NamespacedName, &limit)  
    switch {  
    case apierrors.IsNotFound(err):  
        return ctrl.Result{}, nil  
    case err != nil:  
        return ctrl.Result{}, err  
    }  
  
    // Do some cool logic here.  
  
    return ctrl.Result{}, nil  
}
```

Операторы в k8s

Крутая логика по приведению желаемого к действительному,



```
func (r *Reconciler) Reconcile(  
    ctx context.Context,  
    req ctrl.Request,  
) (ctrl.Result, error) {  
    var limit dbaasv1alpha1.IOLimit  
    err := r.k8s.Get(ctx, req.NamespacedName, &limit)  
    switch {  
    case apierrors.IsNotFound(err):  
        return ctrl.Result{}, nil  
    case err != nil:  
        return ctrl.Result{}, err  
    }  
    // Do some cool logic here.  
    return ctrl.Result{}, nil  
}
```

Применение лимитов

Для применения лимитов используется механизм cgroup v2.

- / В отличие от первой версии, IO контроллер cgroup v2 может ограничивать буферизованный ввод-вывод.

Путь к интерфейсной папке контрольной группы контейнера имеет следующий вид:

```
/sys/fs/cgroup/kubepods.slice/kubepods-pod<POD UID>.slice/cri-containerd-<CONTAINER ID>.scope
```



<https://clc.to/dx2HTw>

Применение лимитов

Внутри папки контрольной группы содержатся интерфейсные файлы контроллеров, нас интересуют интерфейсные файлы IO-контроллера.

```
$ ls -l | grep io
-rw-r--r-- 1 root root 0 Apr 16 13:12 cpuset.cpus.partition
-rw-r--r-- 1 root root 0 Apr 16 13:12 io.max
-rw-r--r-- 1 root root 0 Apr 16 13:12 io.pressure
-r--r--r-- 1 root root 0 Apr 16 13:12 io.stat
-rw-r--r-- 1 root root 0 Apr 16 13:12 io.weight
```

Применение лимитов

Внутри папки контрольной группы содержатся интерфейсные файлы контроллеров, нас интересуют интерфейсные файлы IO-контроллера.

```
$ ls -l | grep io
-rw-r--r-- 1 root root 0 Apr 16 13:12 cpuset.cpus.partition
-rw-r--r-- 1 root root 0 Apr 16 13:12 io.max
-rw-r--r-- 1 root root 0 Apr 16 13:12 io.pressure
-r--r--r-- 1 root root 0 Apr 16 13:12 io.stat
-rw-r--r-- 1 root root 0 Apr 16 13:12 io.weight
```

Сюда записываются ограничения дискового ввода-вывода

Применение лимитов

Внутри папки контрольной группы содержатся интерфейсные файлы контроллеров, нас интересуют интерфейсные файлы IO-контроллера.

```
$ ls -l | grep io
-rw-r--r-- 1 root root 0 Apr 16 13:12 cpuset.cpus.partition
-rw-r--r-- 1 root root 0 Apr 16 13:12 io.max
-rw-r--r-- 1 root root 0 Apr 16 13:12 io.pressure
-r--r--r-- 1 root root 0 Apr 16 13:12 io.stat
-rw-r--r-- 1 root root 0 Apr 16 13:12 io.weight
```



Сюда записываются ограничения дискового ввода-вывода

```
$ cat io.max
8:16 rbps=2097152 wbps=max riops=max wiops=120
```


IOLimit

Ограничения на уровне хранилища объявляются с помощью ресурса IOLimit.

```
apiVersion: dbaas.dbaas.avito.ru/v1alpha1
kind: IOLimit
metadata:
  name: ioba-test-limit
spec:
  storageName: db1
  replicaSetName: rs001
  containers:
    - name: postgresql
      volumes:
        - name: data
          reads:
            iops: 300
            bandwidth: 100MBps
          writes:
            iops: 300
            bandwidth: 20MBps
```

IOLimit

Ограничения на уровне хранилища объявляются с помощью ресурса IOLimit.

```
apiVersion: dbaas.dbaas.avito.ru/v1alpha1
kind: IOLimit
metadata:
  name: ioba-test-limit
spec:
  storageName: db1
  replicaSetName: rs001
  containers:
    - name: postgresql
      volumes:
        - name: data
          reads:
            iops: 300
            bandwidth: 100MBps
          writes:
            iops: 300
            bandwidth: 20MBps
```

Ограничения объявляются на уровне набора реплик хранилища.

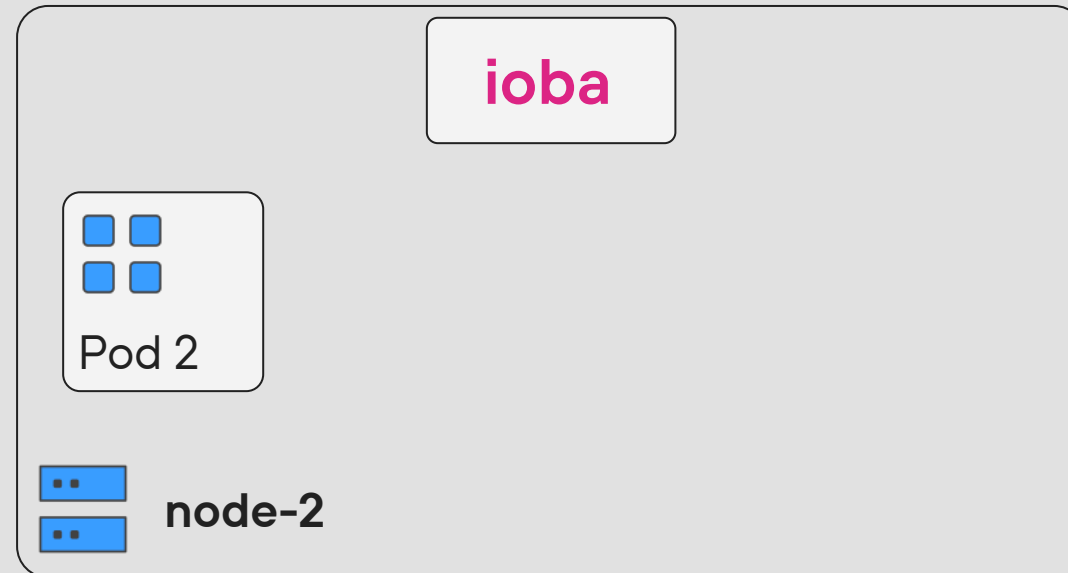
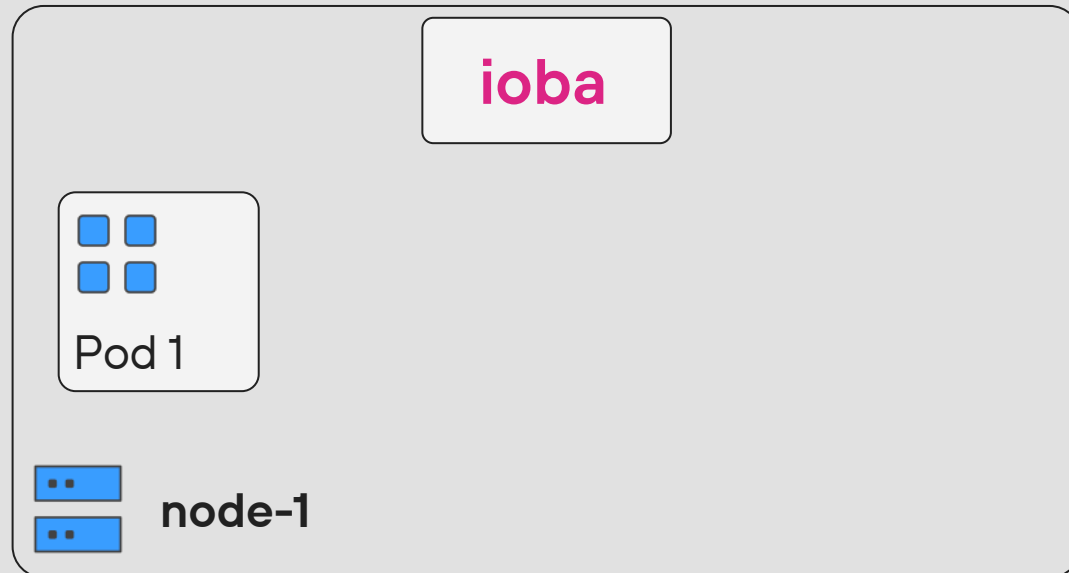
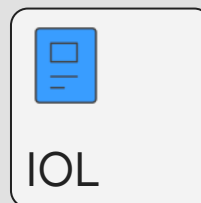
IOLimit

Ограничения на уровне хранилища объявляются с помощью ресурса IOLimit.

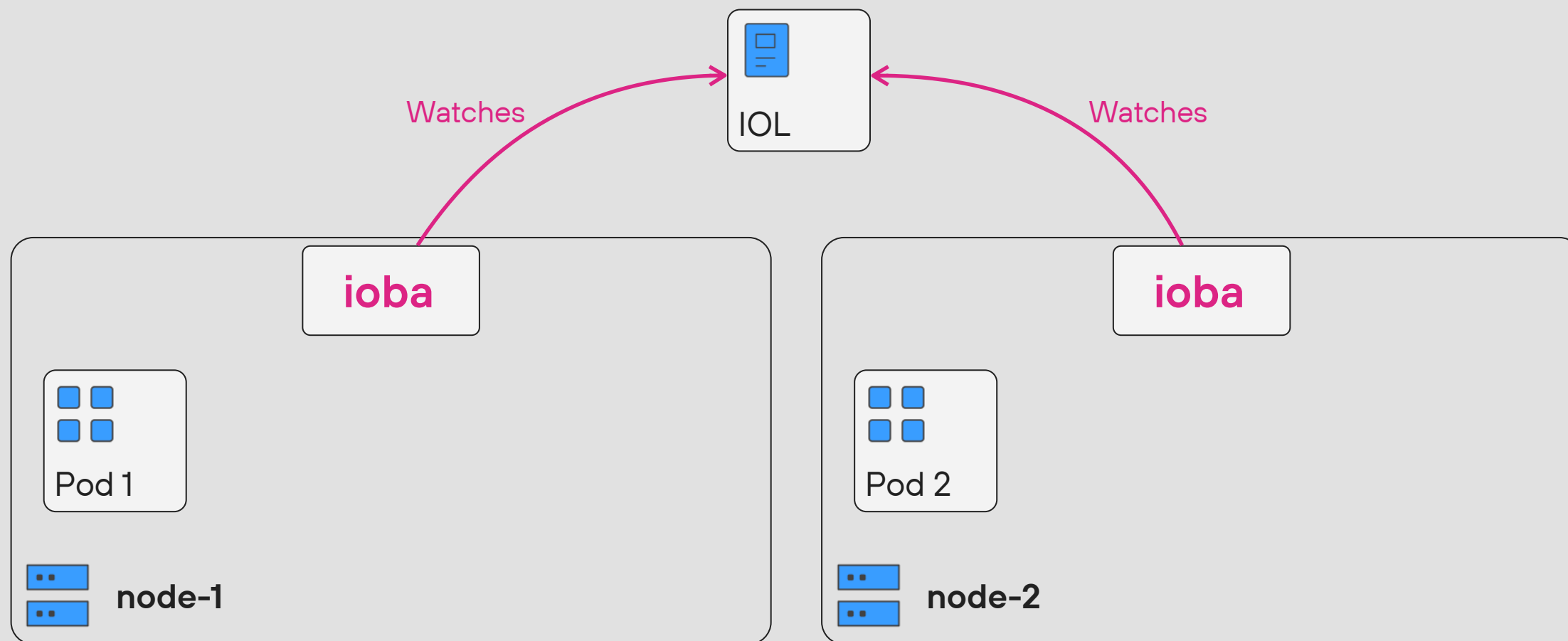
```
apiVersion: dbaas.dbaas.avito.ru/v1alpha1
kind: IOLimit
metadata:
  name: ioba-test-limit
spec:
  storageName: db1
  replicaSetName: rs001
  containers:
    - name: postgresql
      volumes:
        - name: data
          reads:
            iops: 300
            bandwidth: 100MBps
          writes:
            iops: 300
            bandwidth: 20MBps
```

Можно указать разные ограничения для каждого тома каждого контейнера внутри пода хранилища.

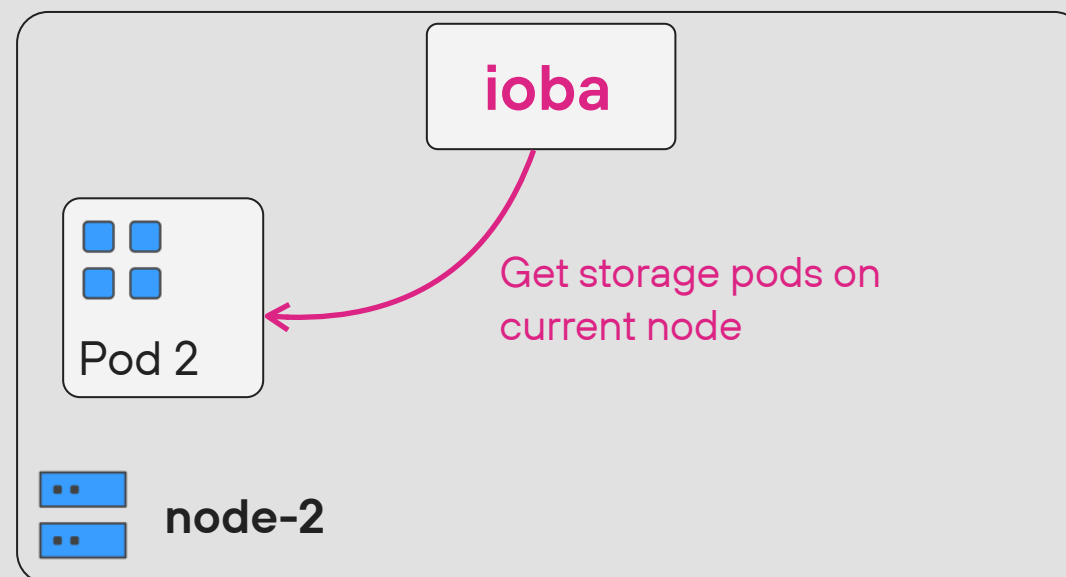
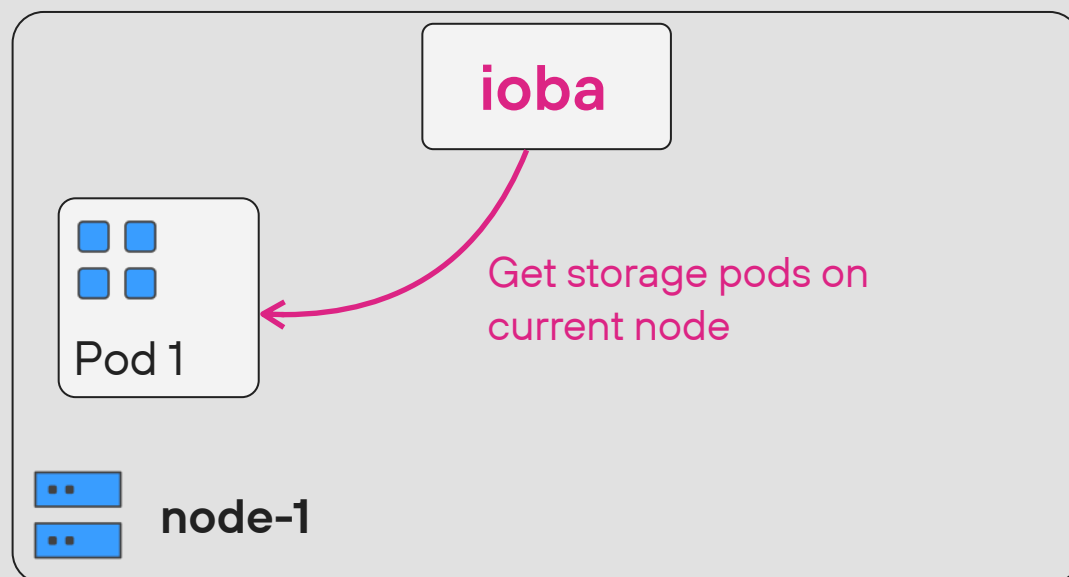
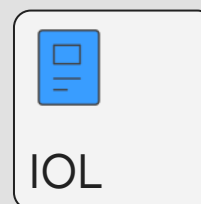
IOLimit



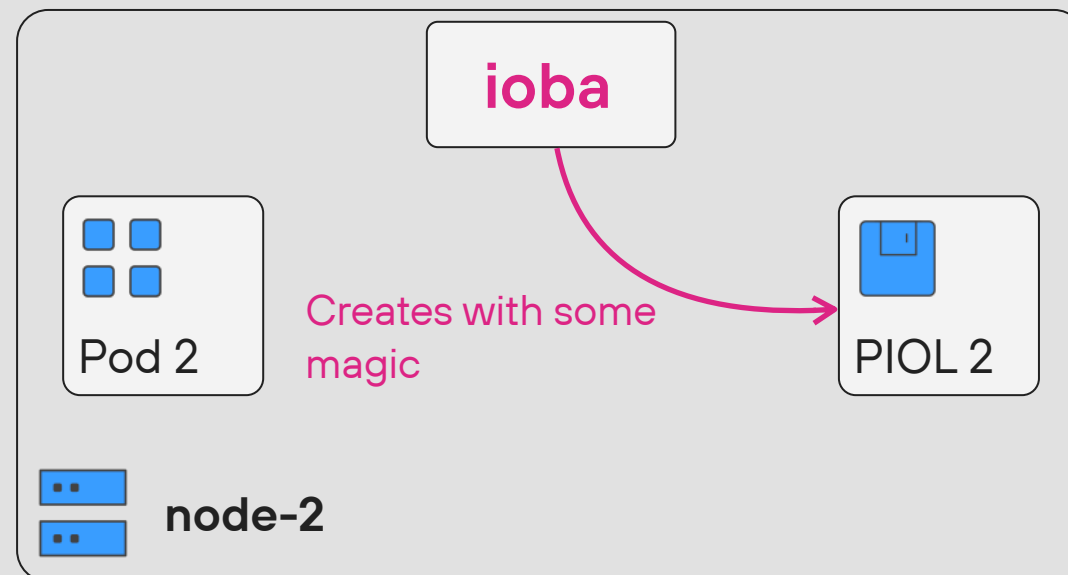
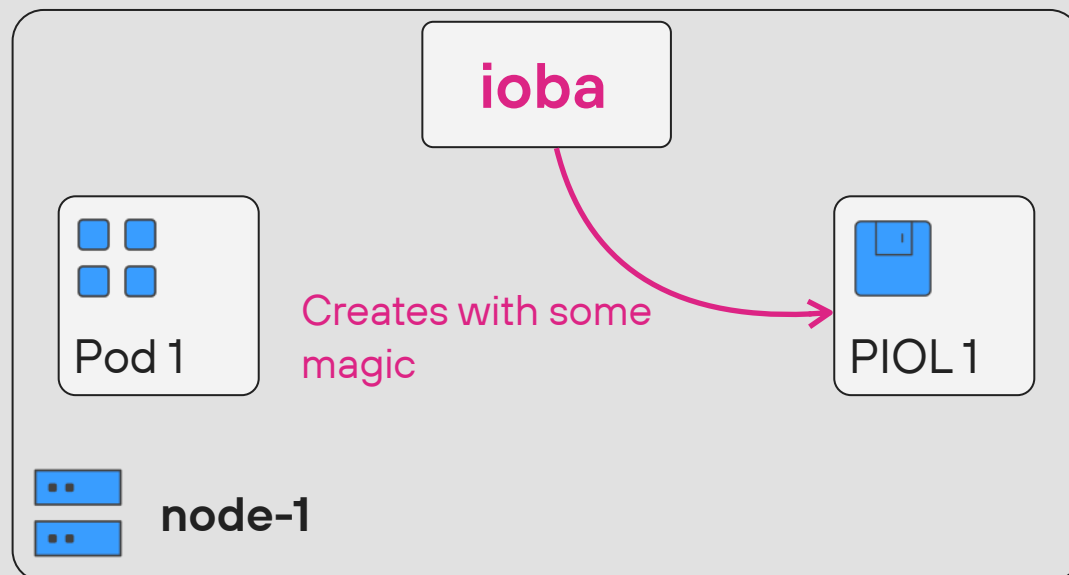
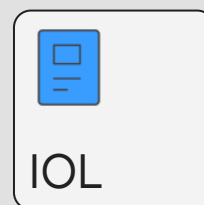
IOLimit



IOLimit



IOLimit



PodIOLimit

Ограничения на уровне пода объявляются с помощью ресурса PodIOLimit. Он содержит более низкоуровневую информацию, необходимую для применения лимитов.

```
apiVersion: dbaas.dbaas.avito.ru/v1alpha1
kind: PodIOLimit
metadata:
  labels:
    dbaas.dbaas.avito.ru/node-name: ioba-worker2
  name: ioba-test-limit-ioba-test-0
spec:
  limits:
    - containerID: bb6deaaa978e99a26e53bead0d39c4d31b7f2432863105ea4479cab4283dcb96
      containerName: postgresql
      deviceNumbers: "254:1"
      readBandwidth: "100000000"
      readIOPS: "300"
      volumeName: data
      writeBandwidth: "20000000"
      writeIOPS: "300"
  nodeName: ioba-worker2
  podName: ioba-test-0
  podUID: 9b06e0ab-fd1f-49b9-8581-d4eed47b282a
  replicaSetName: rs001
  storageName: db1
```


Как получить PodIOLimit?

Для получения информации, необходимой для применения лимитов, необходимо получить следующие данные:

- / ID контейнера
- / Номера монтируемого устройства
- / UID пода

```
spec:
  limits:
    - containerID: bb6deaaa9...
      containerName: postgresql
      deviceNumbers: "254:1"
      readBandwidth: "100000000"
      readIOPS: "300"
      volumeName: data
      writeBandwidth: "200000000"
      writeIOPS: "300"
      podUID: 9b06e0ab-fd1f-49b9-8581-d4eed47b282a
    ...
```

Получение ID контейнера

ID контейнера вытаскивается из ресурса Pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: ioba-test-0
spec:
  ...
status:
  containerStatuses:
  - containerID: containerd://bb6deaaa978e99a26e53bead0d39c4d31b7f2432863105ea4479cab4283dcb96
    name: postgresql
    ready: true
    restartCount: 2
    started: true
  ...
```


Получение ID контейнера

ID контейнера вытаскивается из ресурса Pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: ioba-test-0
spec:
  ...
status:
  containerStatuses:
    - containerID: containerd://bb6deaaa978e99a26e53bead0d39c4d31b7f2432863105ea4479cab4283dcb96
      name: postgresql
      ready: true
      restartCount: 2
      started: true
    ...
```

Получение ID контейнера

ID контейнера вытаскивается из ресурса Pod.



```
apiVersion: v1
kind: Pod
metadata:
  name: ioba-test-0
spec:
  ...
status:
  containerStatuses:
  - containerID: containerd://bb6deaaa978e99a26e53bead0d39c4d31b7f2432863105ea4479cab4283dcb96
    name: postgresql
    ready: true
    restartCount: 2
    started: true
  ...
```

Получение ID контейнера

ID контейнера вытаскивается из ресурса Pod.



```
apiVersion: v1
kind: Pod
metadata:
  name: ioba-test-0
spec:
  ...
status:
  containerStatuses:
    - containerID: containerd://bb6deaaa978e99a26e53bead0d39c4d31b7f2432863105ea4479cab4283dcb96
      name: postgresql
      ready: true
      restartCount: 2
      started: true
    ...
```

Получение ID контейнера

ID контейнера вытаскивается из ресурса Pod.



```
apiVersion: v1
kind: Pod
metadata:
  name: ioba-test-0
spec:
  ...
status:
  containerStatuses:
  - containerID: containerd://bb6deaaa978e99a26e53bead0d39c4d31b7f2432863105ea4479cab4283dcb96
    name: postgresql
    ready: true
    restartCount: 2
    started: true
  ...
```

Получение ID контейнера

ID контейнера вытаскивается из ресурса Pod.



```
apiVersion: v1
kind: Pod
metadata:
  name: ioba-test-0
spec:
  ...
status:
  containerStatuses:
    - containerID: containerd://bb6deaaa978e99a26e53bead0d39c4d31b7f2432863105ea4479cab4283dcb96
      name: postgresql
      ready: true
      restartCount: 2
      started: true
  ...
```


Получение ID контейнера

ID контейнера вытаскивается из ресурса Pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: ioba-test-0
spec:
  ...
status:
  containerStatuses:
    - containerID: containerd://bb6deaaa978e99a26e53bead0d39c4d31b7f2432863105ea4479cab4283dcb96
      name: postgresql
      ready: true
      restartCount: 2
      started: true
    ...
```



Получение UID Pod-a

UID Pod-a вытаскивается из ресурса Pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: ioba-test-0
  namespace: default
  uid: 9b06e0ab-fd1f-49b9-8581-d4eed47b282a
spec: ...
```

Получение UID Pod-a

UID Pod-a вытаскивается из ресурса Pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: ioba-test-0
  namespace: default
  uid: 9b06e0ab-fd1f-49b9-8581-d4eed47b282a
spec: ...
```

Получение номеров устройства

Номера примонтированного устройства можно посмотреть в файловой системе /proc.

```
$ cat /proc/<pid>/mountinfo
1979 1978 0:585 / /proc rw,nosuid,nodev,noexec,relatime - proc proc rw
1980 1978 0:586 / /dev rw,nosuid - tmpfs tmpfs rw,size=65536k,mode=755
1981 1980 0:587 / /dev/pts rw,nosuid,noexec,relatime - devpts devpts rw,gid=5,mode=620,ptmxmode=666
1982 1980 0:577 / /dev/mqueue rw,nosuid,nodev,noexec,relatime - mqueue mqueue rw
1989 1978 0:581 / /sys ro,nosuid,nodev,noexec,relatime - sysfs sysfs ro
1991 1978 254:1 /docker/volumes/.../csi-hostpath-data/424f63b8-eb78-11ee-b165-de6392de7398
/test rw,relatime - ext4 /dev/vda1 rw,discard
```

Получение номеров устройства

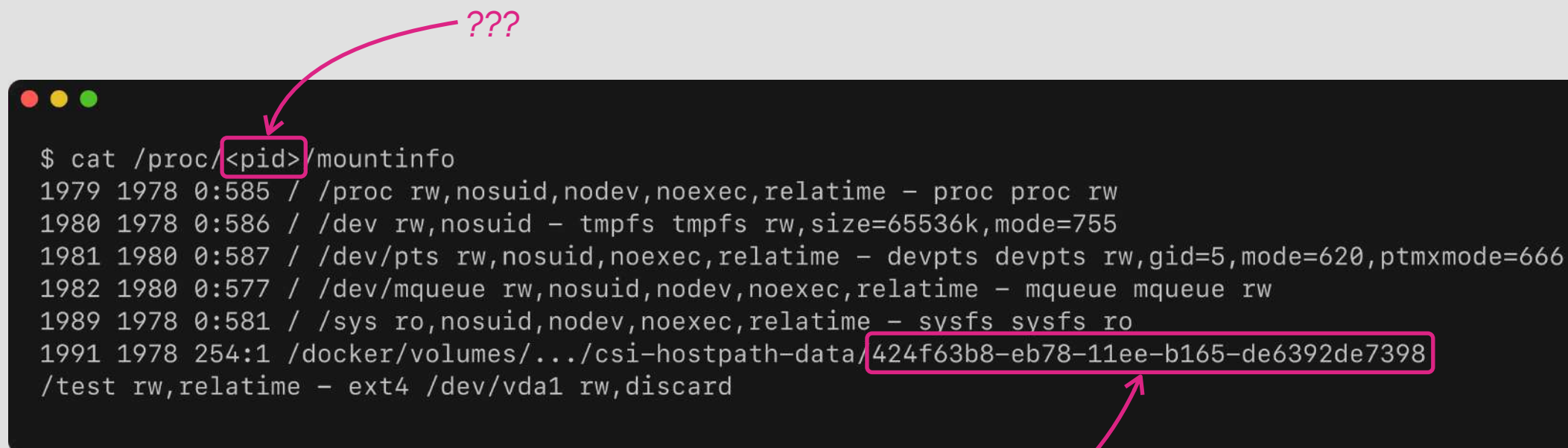
Номера примонтированного устройства можно посмотреть в файловой системе /proc.

```
$ cat /proc/<pid>/mountinfo
1979 1978 0:585 / /proc rw,nosuid,nodev,noexec,relatime - proc proc rw
1980 1978 0:586 / /dev rw,nosuid - tmpfs tmpfs rw,size=65536k,mode=755
1981 1980 0:587 / /dev/pts rw,nosuid,noexec,relatime - devpts devpts rw,gid=5,mode=620,ptmxmode=666
1982 1980 0:577 / /dev/mqueue rw,nosuid,nodev,noexec,relatime - mqueue mqueue rw
1989 1978 0:581 / /sys ro,nosuid,nodev,noexec,relatime - sysfs sysfs ro
1991 1978 254:1 /docker/volumes/.../csi-hostpath-data/424f63b8-eb78-11ee-b165-de6392de7398
/test rw,relatime - ext4 /dev/vda1 rw,discard
```



Получение номеров устройства

Номера примонтированного устройства можно посмотреть в файловой системе /proc.



```
$ cat /proc/<pid>/mountinfo
1979 1978 0:585 / /proc rw,nosuid,nodev,noexec,relatime - proc proc rw
1980 1978 0:586 / /dev rw,nosuid - tmpfs tmpfs rw,size=65536k,mode=755
1981 1980 0:587 / /dev/pts rw,nosuid,noexec,relatime - devpts devpts rw,gid=5,mode=620,ptmxmode=666
1982 1980 0:577 / /dev/mqueue rw,nosuid,nodev,noexec,relatime - mqueue mqueue rw
1989 1978 0:581 / /sys ro,nosuid,nodev,noexec,relatime - sysfs sysfs ro
1991 1978 254:1 /docker/volumes/.../csi-hostpath-data/424f63b8-eb78-11ee-b165-de6392de7398
/test rw,relatime - ext4 /dev/vda1 rw,discard
```

???

Получение номеров устройства

Номера примонтированного устройства можно посмотреть в файловой системе /proc.

Хостовой PID основного процесса контейнера.

```
$ cat /proc/<pid>/mountinfo
1979 1978 0:585 / /proc rw,nosuid,nodev,noexec,relatime - proc proc rw
1980 1978 0:586 / /dev rw,nosuid - tmpfs tmpfs rw,size=65536k,mode=755
1981 1980 0:587 / /dev/pts rw,nosuid,noexec,relatime - devpts devpts rw,gid=5,mode=620,ptmxmode=666
1982 1980 0:577 / /dev/mqueue rw,nosuid,nodev,noexec,relatime - mqueue mqueue rw
1989 1978 0:581 / /sys ro,nosuid,nodev,noexec,relatime - sysfs sysfs ro
1991 1978 254:1 /docker/volumes/.../csi-hostpath-data/424f63b8-eb78-11ee-b165-de6392de7398
/test rw,relatime - ext4 /dev/vda1 rw,discard
```

volumeHandle — поле из спецификации PV.

Получение volumeHandle

volumeHandle содержится в спецификации PersistentVolume.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pvc-423228a1-0a93-4134-9a32-63b867a41984
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 1Gi
  csi:
    driver: hostpath.csi.k8s.io
    volumeHandle: 1b6536c4-fcb3-11ee-92c6-4ed0e5726fd8
  storageClassName: hostpath-storageclass
  volumeMode: Filesystem
status:
  phase: Bound
```

Получение volumeHandle

volumeHandle содержится в спецификации PersistentVolume.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pvc-423228a1-0a93-4134-9a32-63b867a41984
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 1Gi
  csi:
    driver: hostpath.csi.k8s.io
    volumeHandle: 1b6536c4-fcb3-11ee-92c6-4ed0e5726fd8
  storageClassName: hostpath-storageclass
  volumeMode: Filesystem
status:
  phase: Bound
```

Получение PID

Так как Kubernetes абстрагирует пользователя от внутренней логики запуска приложений, PID в хостовой системе нельзя узнать, используя абстракции K8s.

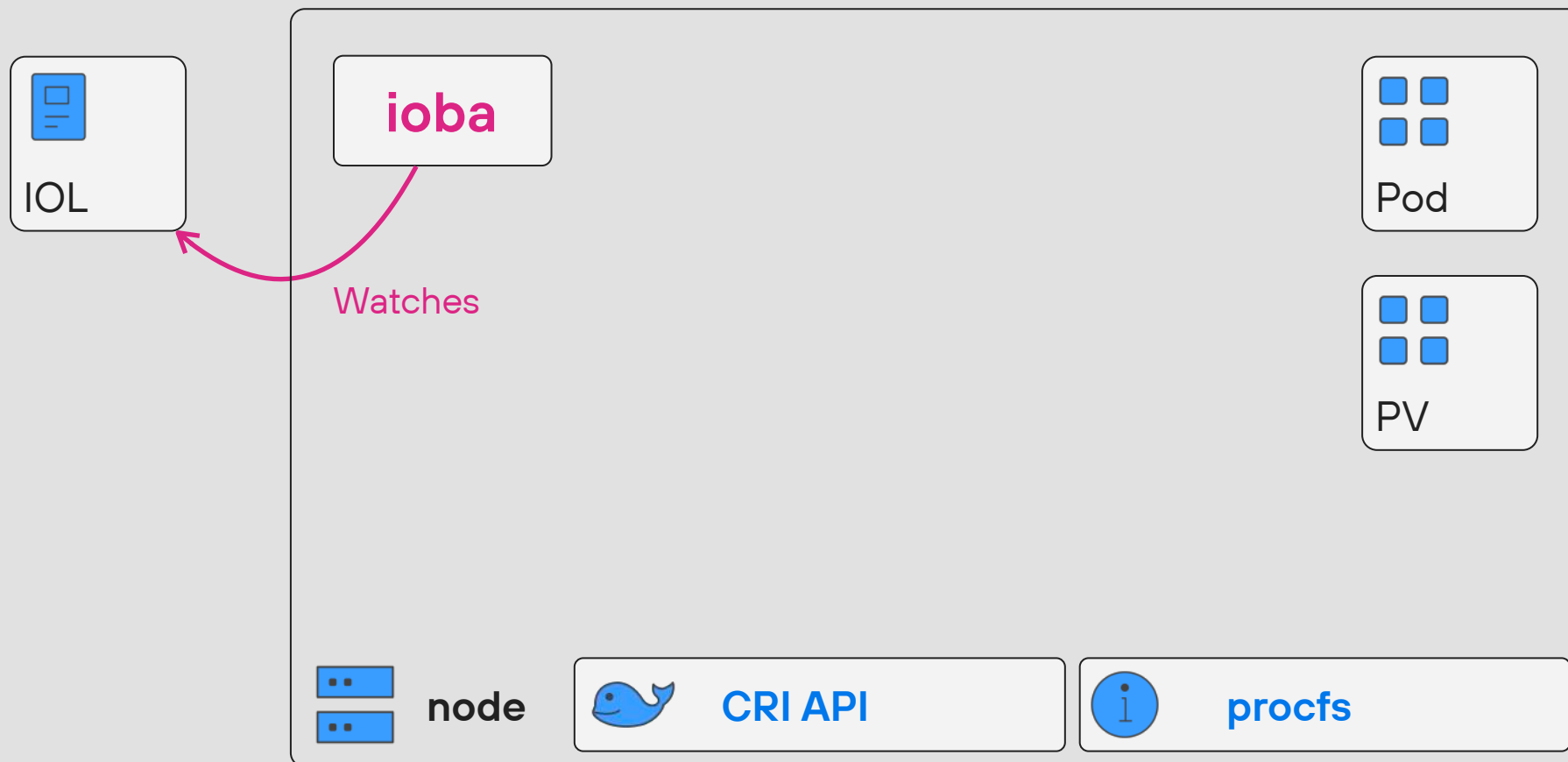
Получение PID

Так как Kubernetes абстрагирует пользователя от внутренней логики запуска приложений, PID в хостовой системе нельзя узнать, используя абстракции K8s.

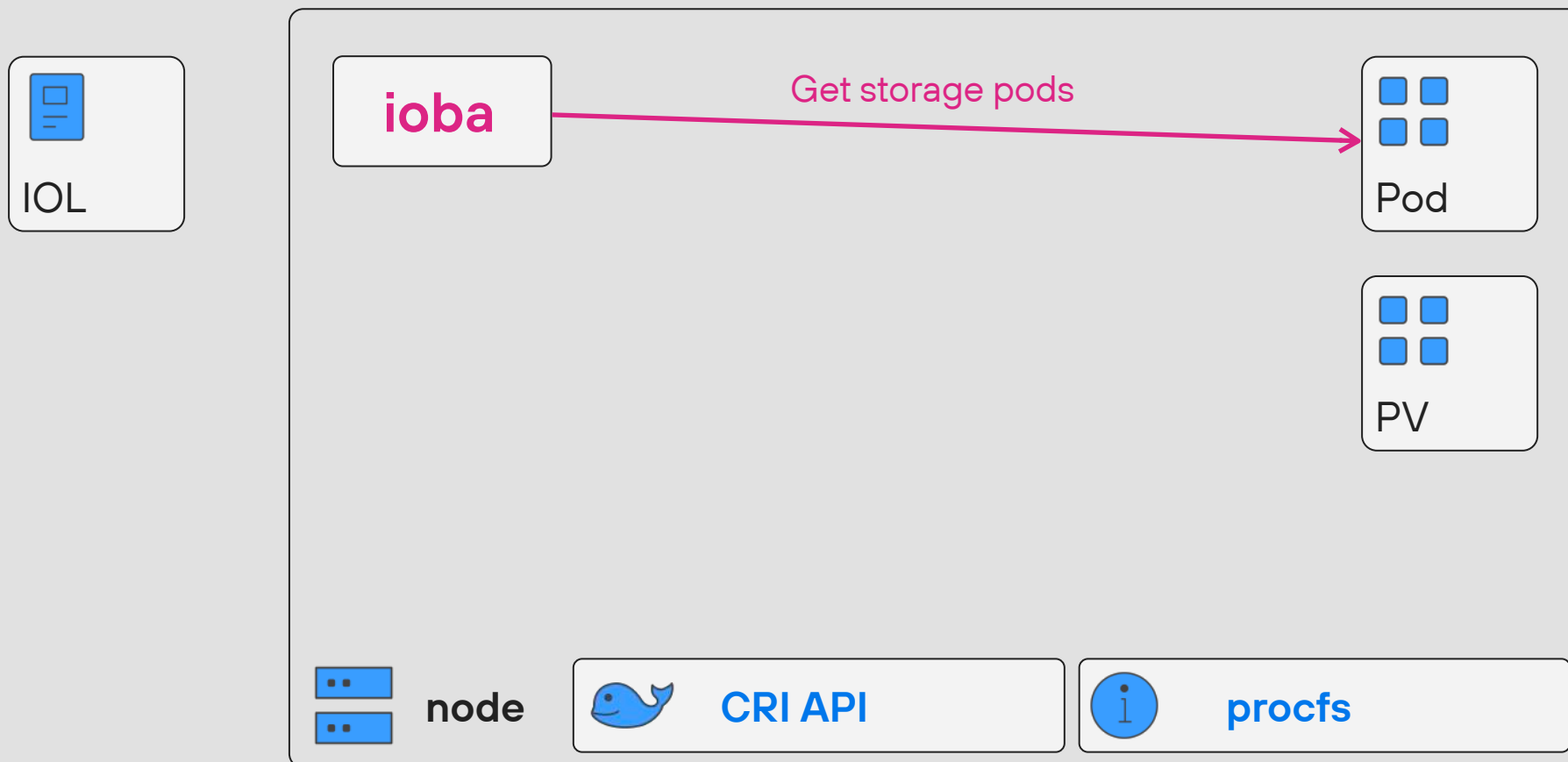
Поэтому... мы ломаем абстракции и узнаем PID напрямую из CRI!

```
service RuntimeService {  
    // ContainerStatus returns status of the container. If the container is not  
    // present, returns an error.  
    rpc ContainerStatus(ContainerStatusRequest) returns (ContainerStatusResponse) {}  
}
```

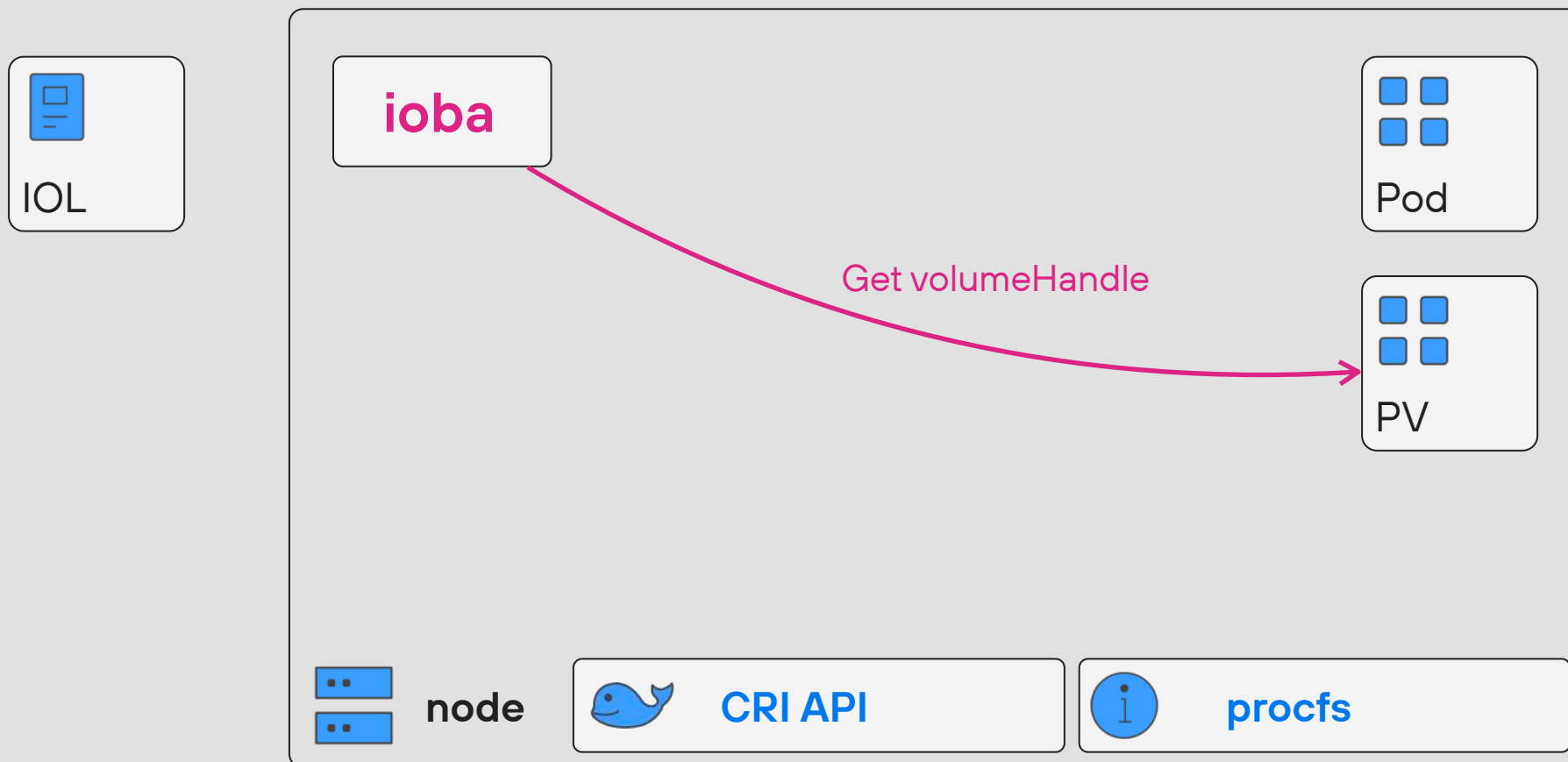
Итоговая схема



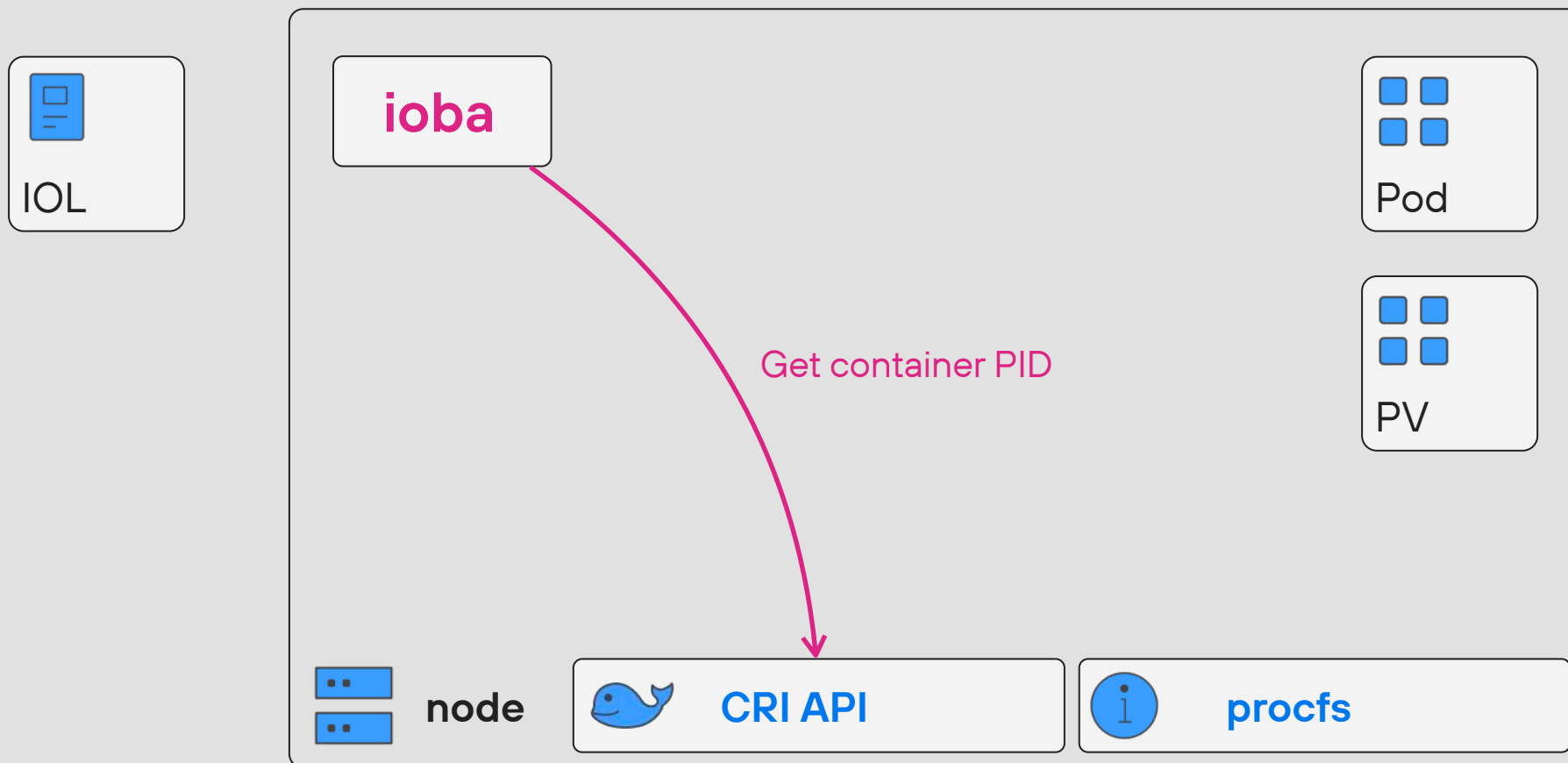
Итоговая схема



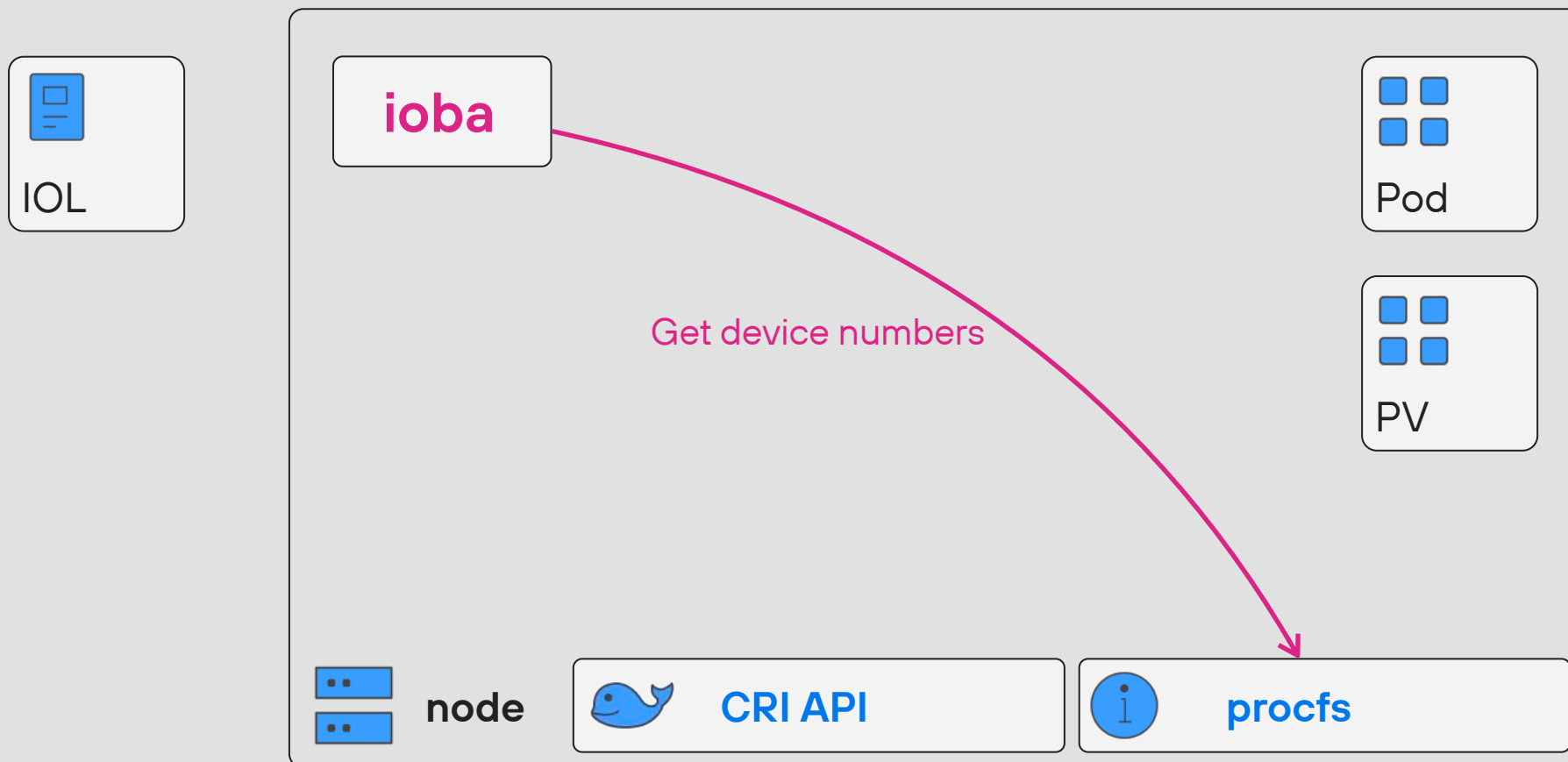
Итоговая схема



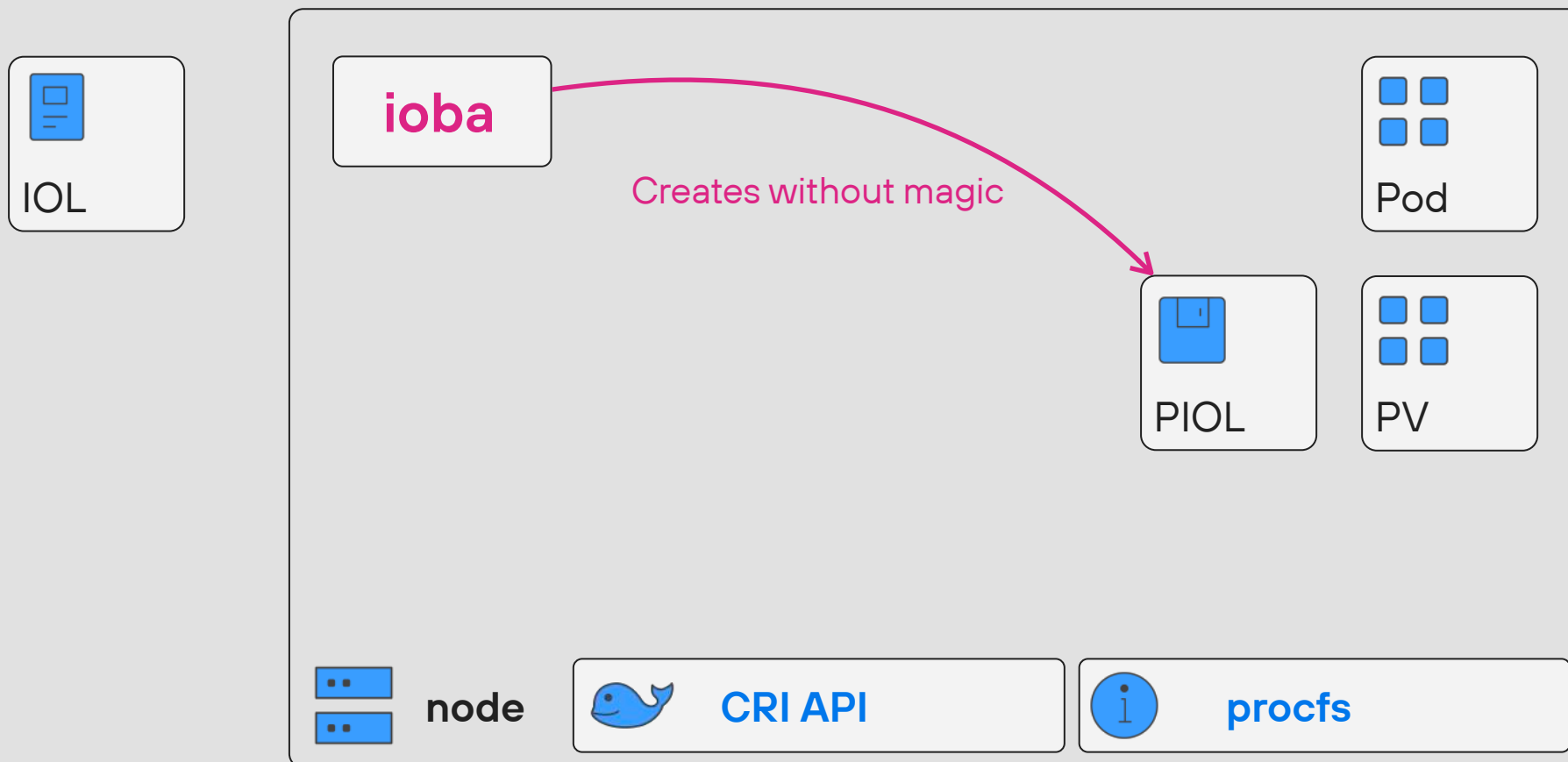
Итоговая схема



Итоговая схема



Итоговая схема





Планирование в k8s

Планировщик K8s определяет ноду, на которой будет запущен любой создаваемый под. Он учитывает запросы пода на ресурсы (requests) и доступные ресурсы ноды.



node-1

Free CPU=300m
Free Mem=2G

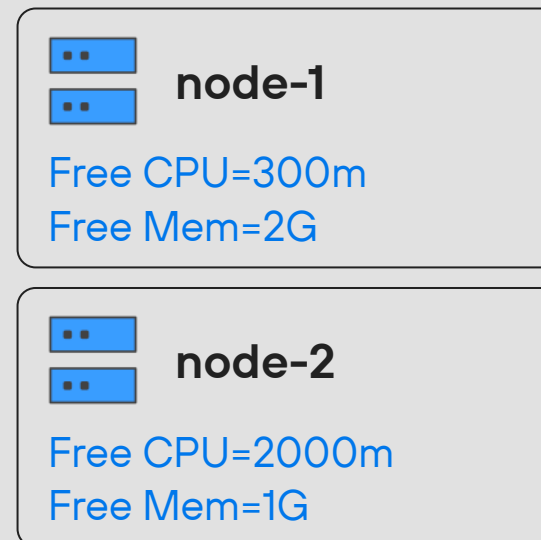


node-2

Free CPU=2000m
Free Mem=1G

Планирование в k8s

Планировщик K8s определяет ноду, на которой будет запущен любой создаваемый под. Он учитывает запросы пода на ресурсы (requests) и доступные ресурсы ноды.



Планирование в k8s

Планировщик K8s определяет ноду, на которой будет запущен любой создаваемый под. Он учитывает запросы пода на ресурсы (requests) и доступные ресурсы ноды.



node-1

Free CPU=300m

Free Mem=2G



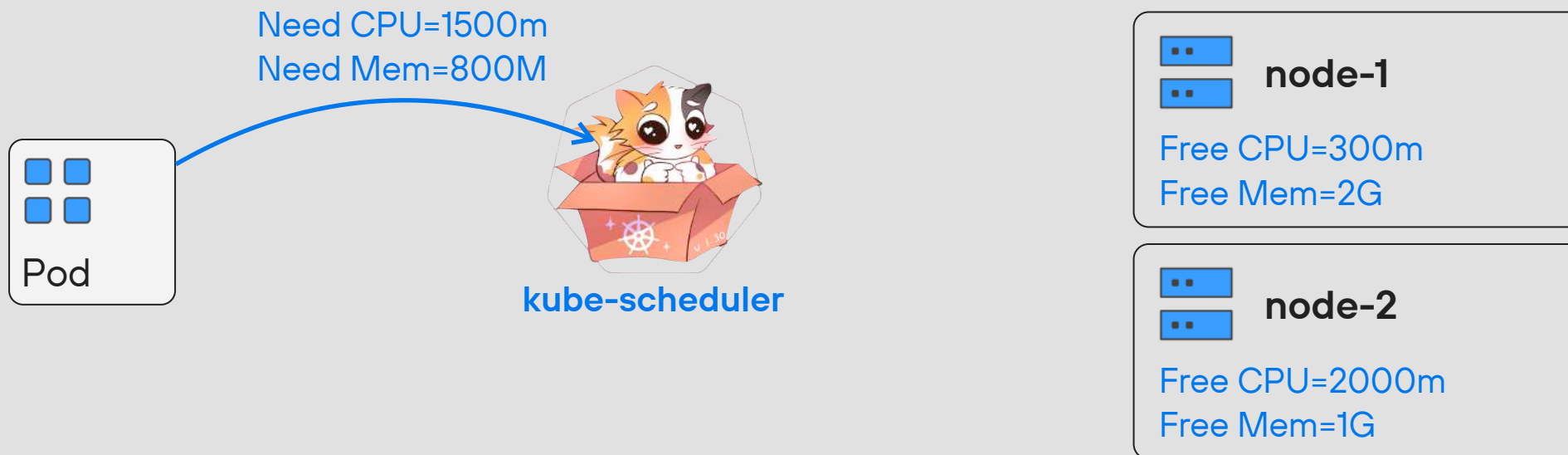
node-2

Free CPU=2000m

Free Mem=1G

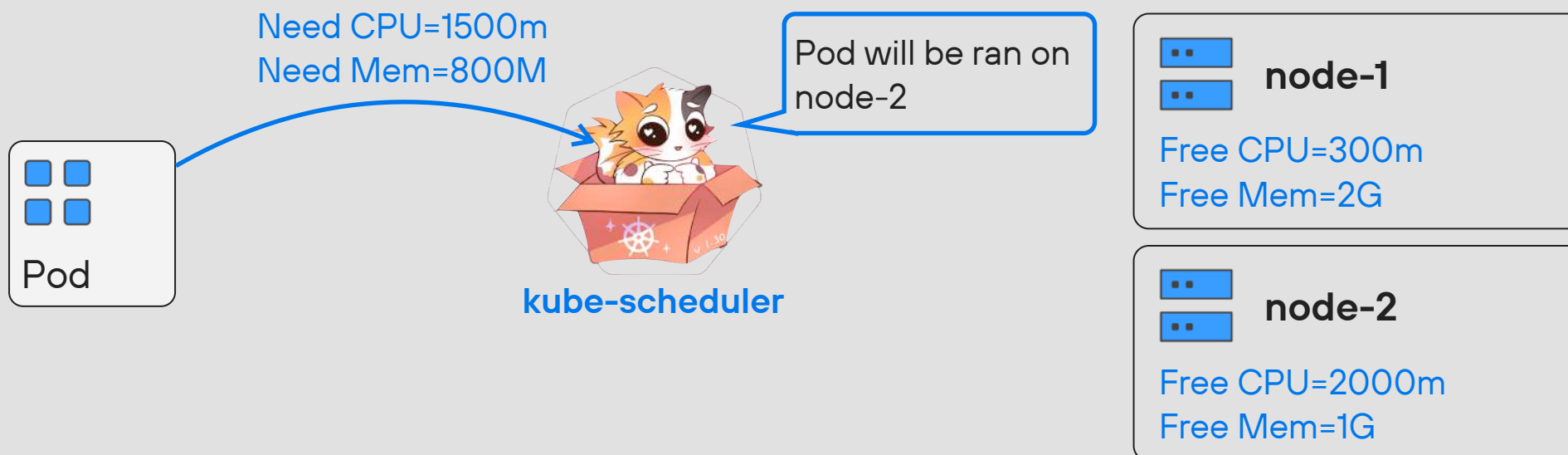
Планирование в k8s

Планировщик K8s определяет ноду, на которой будет запущен любой создаваемый под. Он учитывает запросы пода на ресурсы (requests) и доступные ресурсы ноды.



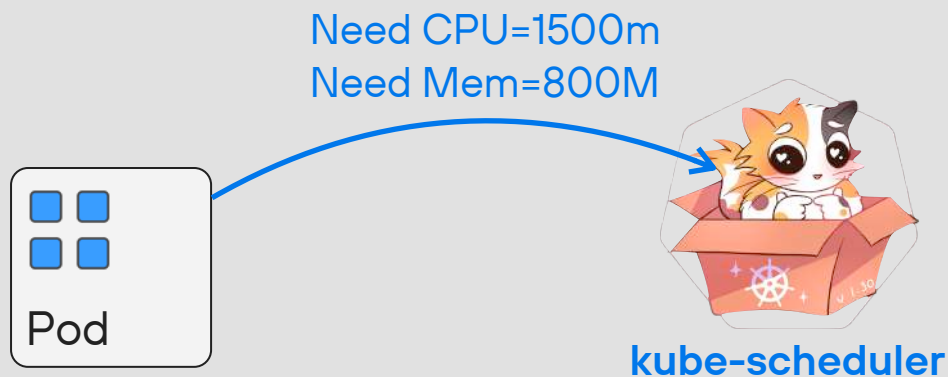
Планирование в k8s

Планировщик K8s определяет ноду, на которой будет запущен любой создаваемый под. Он учитывает запросы пода на ресурсы (requests) и доступные ресурсы ноды.



Планирование в k8s

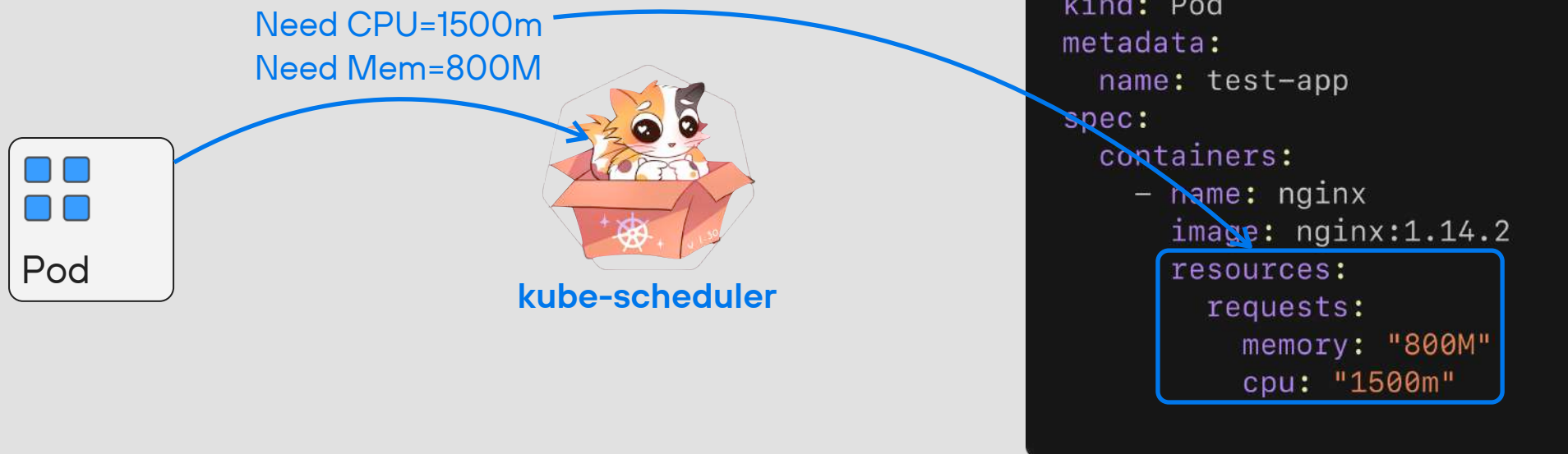
Планировщик K8s определяет ноду, на которой будет запущен любой создаваемый под. Он учитывает запросы пода на ресурсы (requests) и доступные ресурсы ноды.



```
apiVersion: v1
kind: Pod
metadata:
  name: test-app
spec:
  containers:
    - name: nginx
      image: nginx:1.14.2
      resources:
        requests:
          memory: "800M"
          cpu: "1500m"
```

Планирование в k8s

Планировщик K8s определяет ноду, на которой будет запущен любой создаваемый под. Он учитывает запросы пода на ресурсы (requests) и доступные ресурсы ноды.



Планирование в k8s

Kubernetes позволяет создавать свои типы ресурсов, которые также будут учитываться планировщиком. Для этого нужно заполнить ёмкость ноды по этому ресурсу, и начать указывать его в блоке requests.

Планирование в k8s

Kubernetes позволяет создавать свои типы ресурсов, которые также будут учитываться планировщиком. Для этого нужно заполнить ёмкость ноды по этому ресурсу, и начать указывать его в блоке requests.

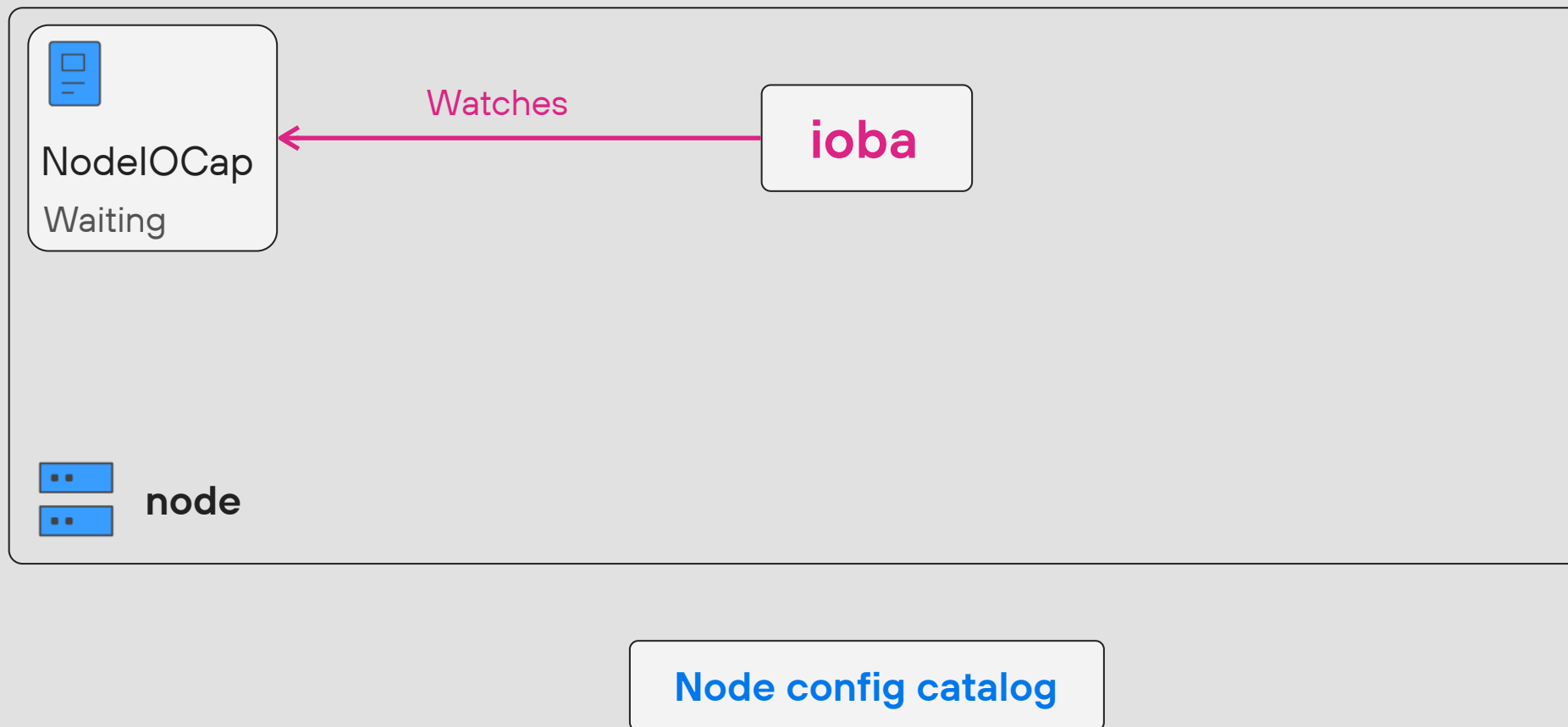
```
apiVersion: v1
kind: Pod
metadata:
  name: crdb-test
spec:
  containers:
    - name: cockroachdb
      image: cockroachdb/cockroach:v23.2.3
      resources:
        requests:
          memory: "2G"
          cpu: "1500m"
          ioba.dbaas.avito.ru/disk_read_iops: "300"
          ioba.dbaas.avito.ru/disk_write_iops: "100"
          ioba.dbaas.avito.ru/disk_read_bandwidth_mbps: "100"
          ioba.dbaas.avito.ru/disk_write_bandwidth_mbps: "100"
```

Как задавать capacity ноды?

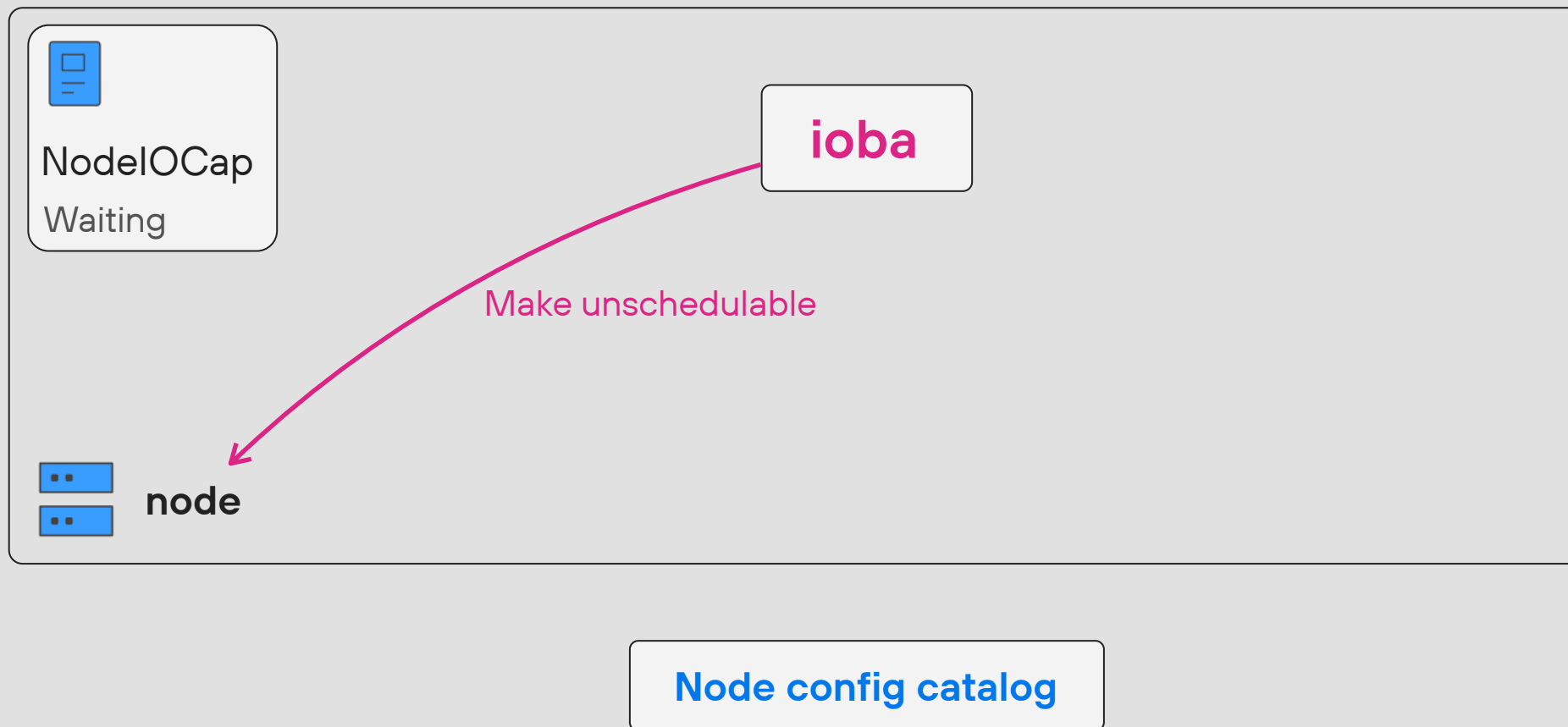
При появлении новой ноды в кластере, ioba создаёт ресурс NodeIOCapacity. Для указанной ноды он содержит ёмкость по ресурсам дискового ввода-вывода и текущее состояние. Изначально ресурс находится в состоянии Waiting.

```
apiVersion: ioba.dbaas.avito.ru/v1alpha1
kind: NodeIOCapacity
metadata:
  name: some-node-capacity
  labels:
    node-name: some-node
spec:
  nodeName: some-node
status:
  state: Waiting
```

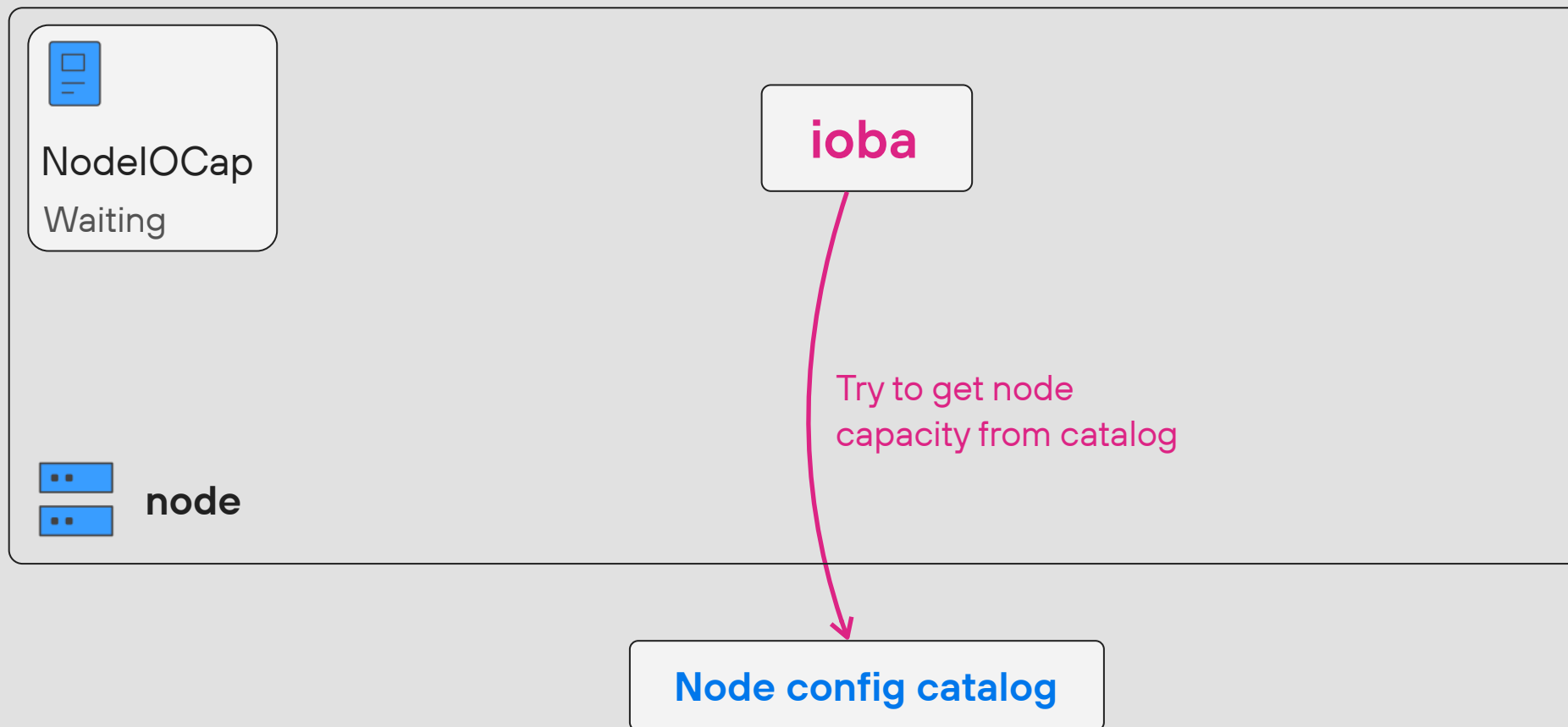
Как задавать capacity ноды?



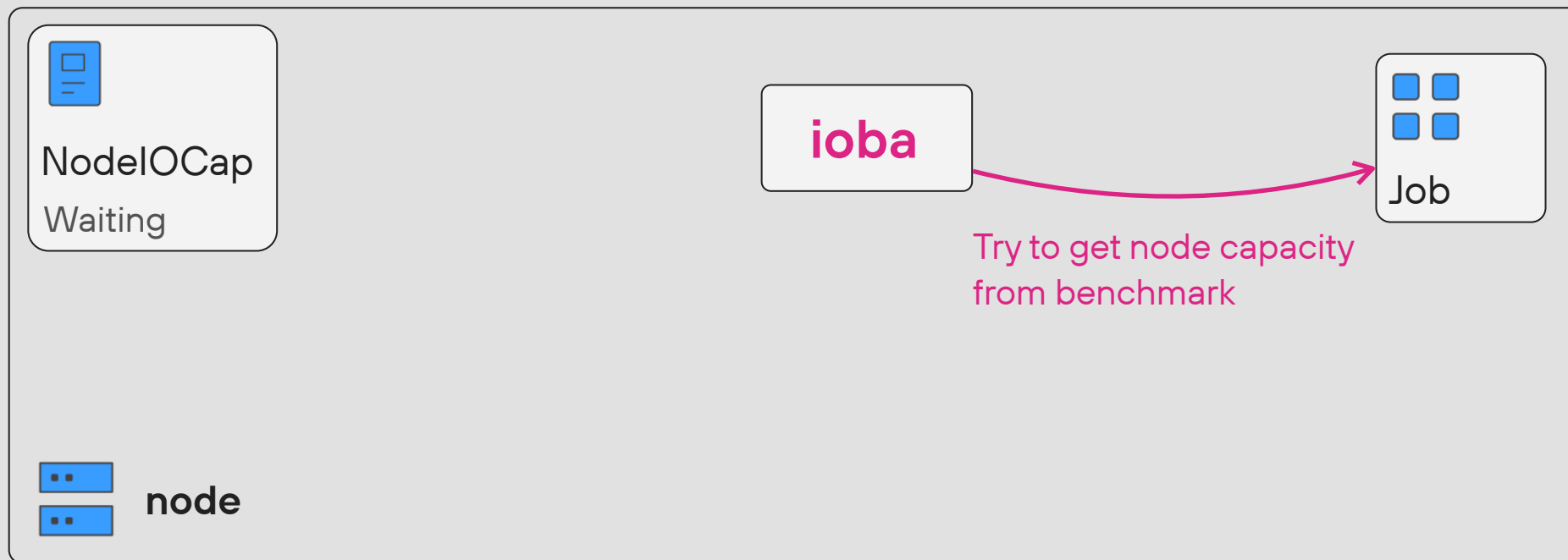
Как задавать capacity ноды?



Как задавать capacity ноды?

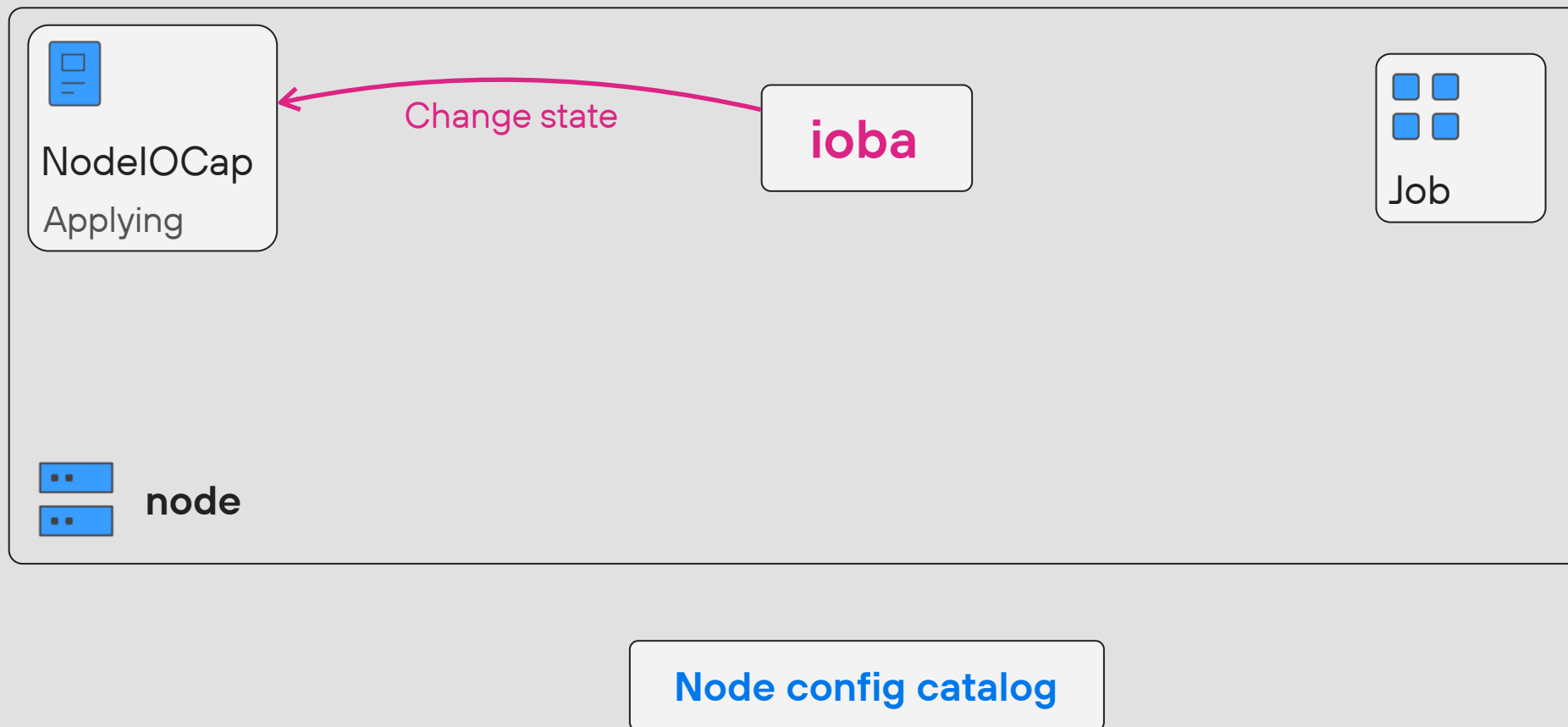


Как задавать capacity ноды?

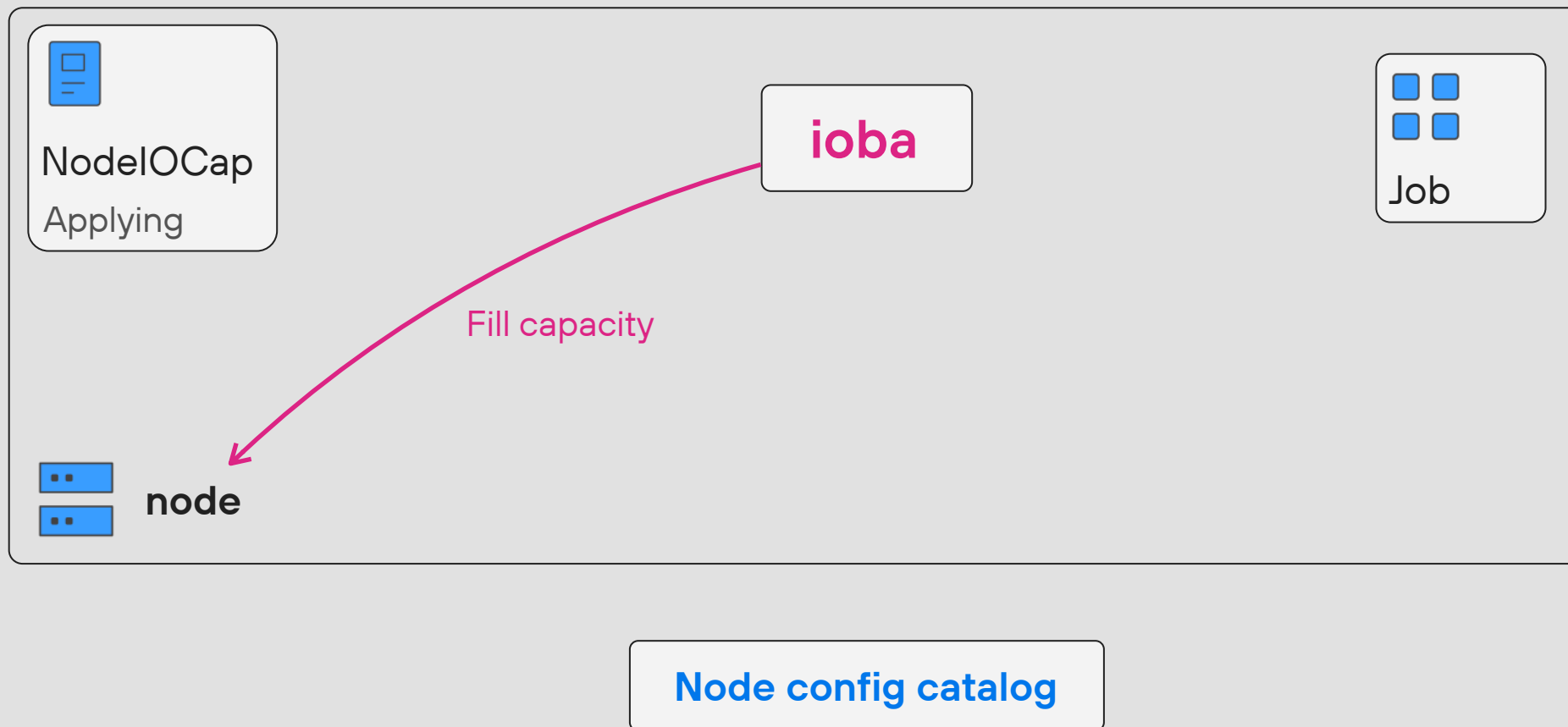


Node config catalog

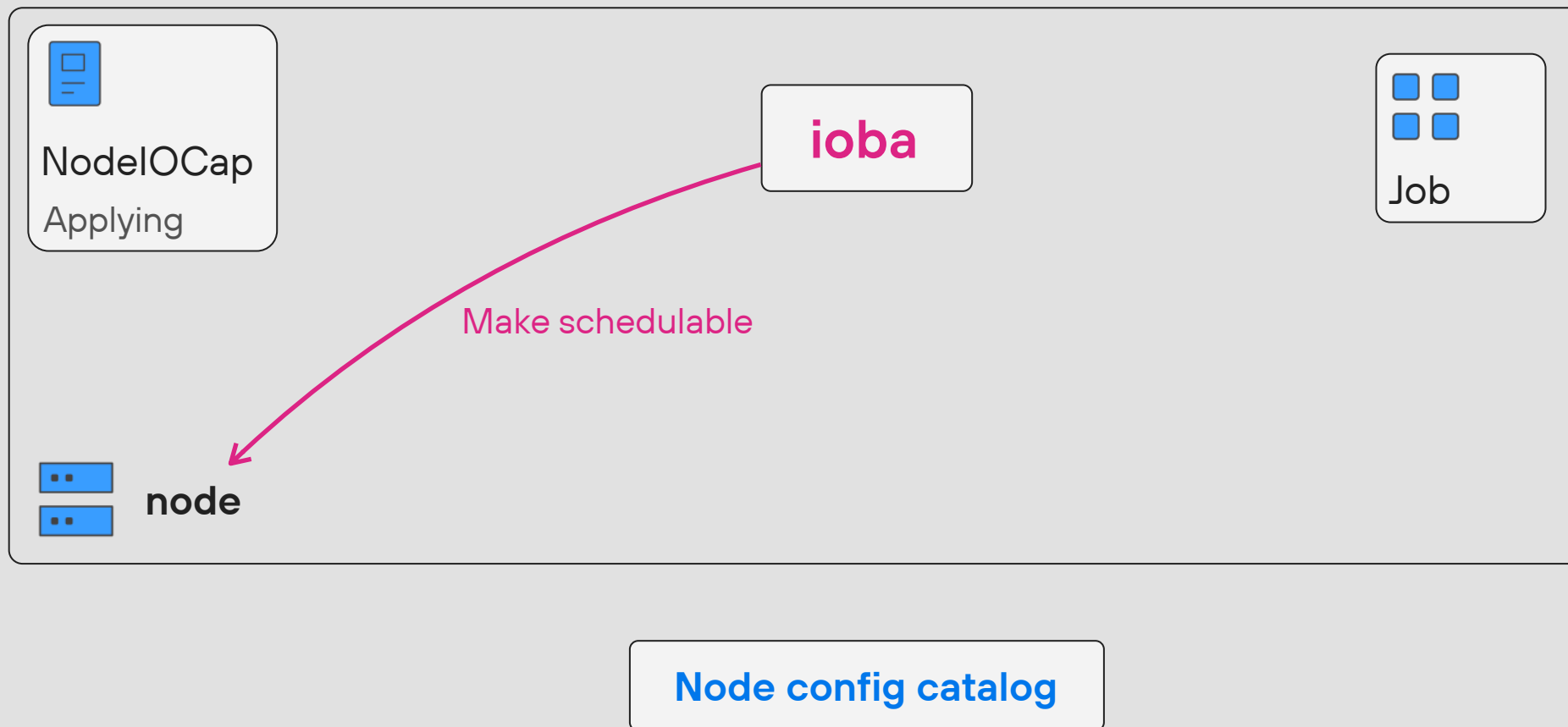
Как задавать capacity ноды?



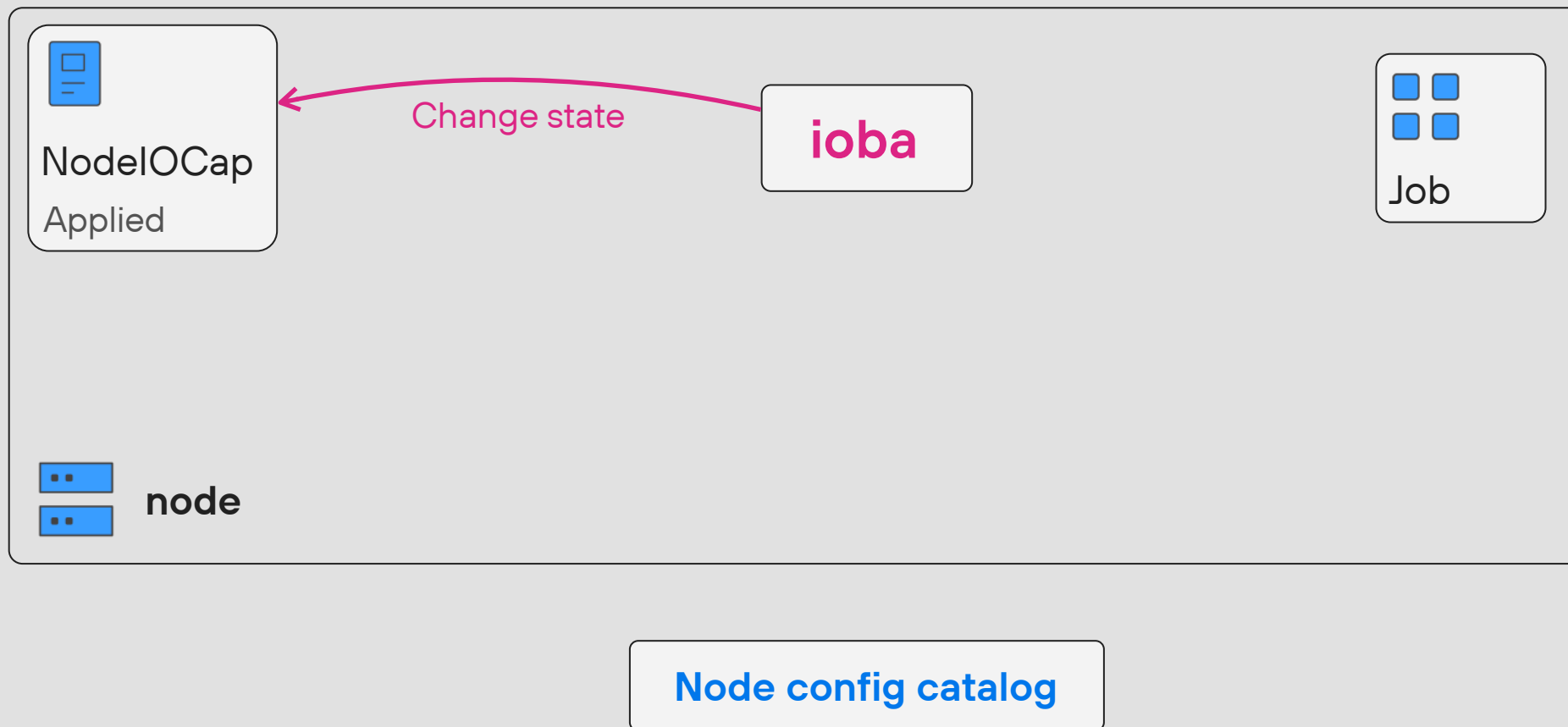
Как задавать capacity ноды?



Как задавать capacity ноды?

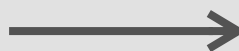


Как задавать capacity ноды?



Как задавать capacity ноды?

```
apiVersion: ioba.dbaas.avito.ru/v1alpha1
kind: NodeIOCapacity
metadata:
  name: some-node-capacity
  labels:
    node-name: some-node
spec:
  nodeName: some-node
status:
  state: Waiting
```



```
apiVersion: ioba.dbaas.avito.ru/v1alpha1
kind: NodeIOCapacity
metadata:
  name: some-node-capacity
  labels:
    node-name: some-node
spec:
  nodeName: some-node
  reads:
    iops: 1000000
    bandwidthMBps: 5000000000
  writes:
    iops: 1000000
    bandwidthMBps: 5000000000
status:
  conditions:
    ...
  state: Applied
```


Выводы

- / В Kubernetes есть отличные примитивы для персистентного хранения данных.
- / В новых версиях Kubernetes есть зачатки механизма ограничения дискового ввода-вывода.
- / Kubernetes подходит для эксплуатации тысяч инстансов баз данных.
- / Если не хотите дожидаться полноценного внедрения функционала ограничения дискового ввода-вывода в Kubernetes, или он не удовлетворяет вашим требованиям, можете сделать как мы — это не страшно ;)

Спасибо!

phd 2X 

Задавайте вопросы

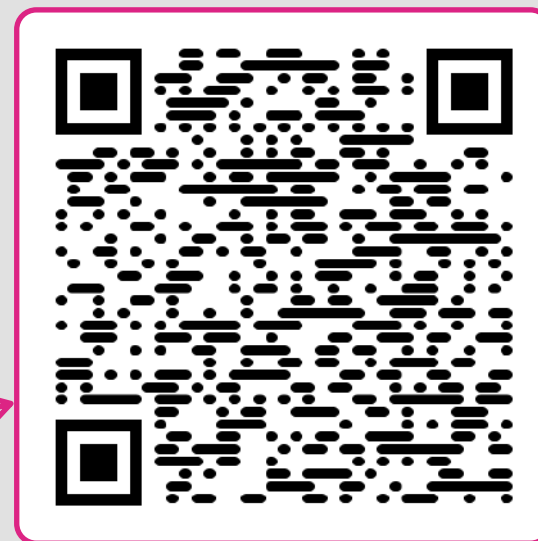


avknyazhev@avito.ru



@muhomorfus

Слайды



QR-код для оценки доклада

phd 2 Positive Hack
Days Fest
от positive technologies

