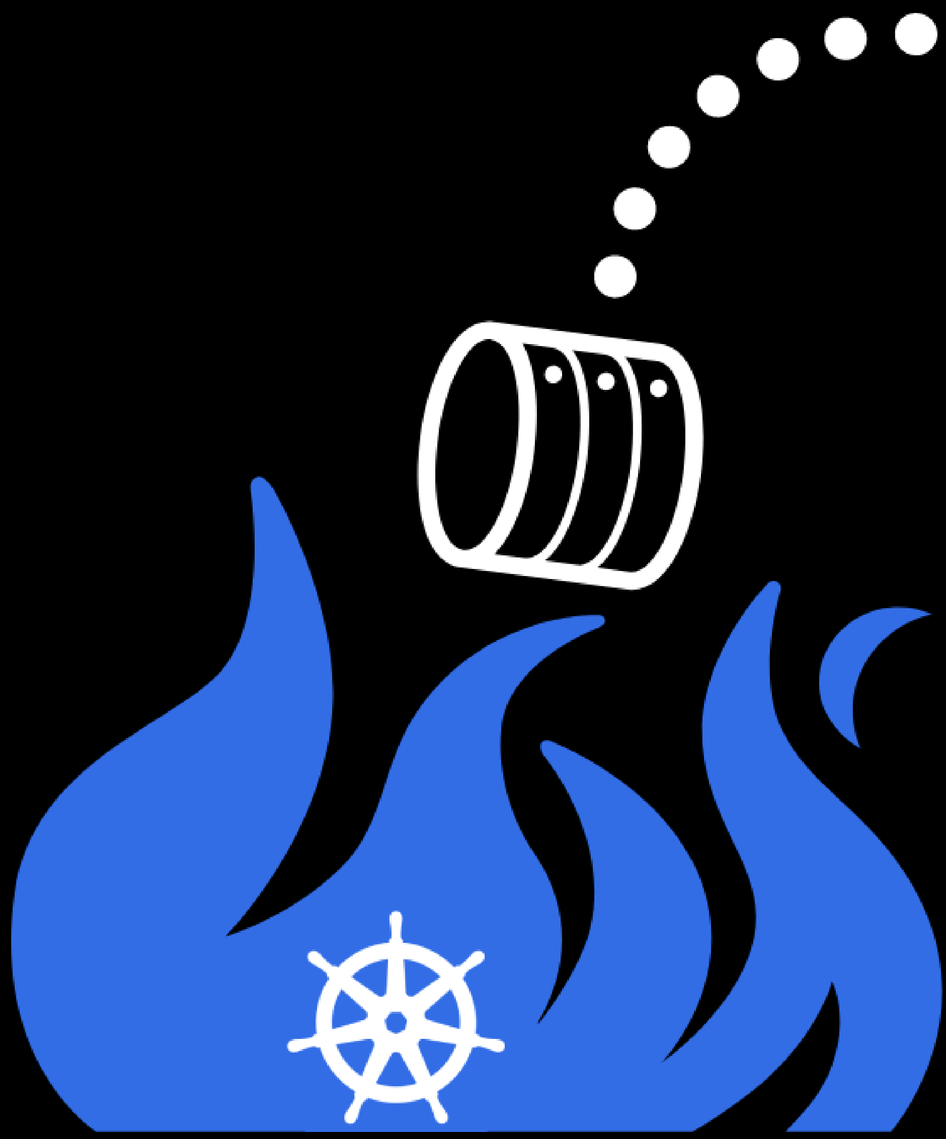
avito.tech

Москва

2024

Stateful-приложения в Kubernetes (DB edition)



Игорь Конев

Старший инженер DBaaS

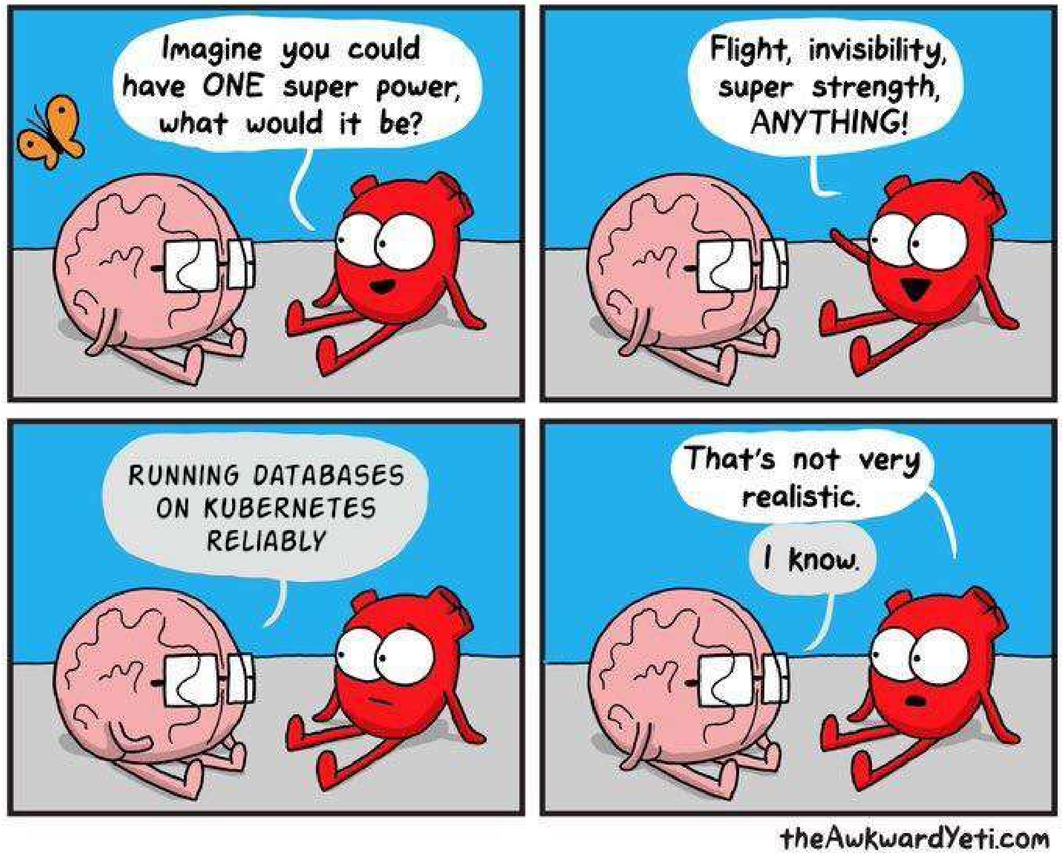
Игорь Конев

Старший инженер DBaaS

Работаю в Авито и команде DBaaS с 2021 года.
Сейчас придумываю и разрабатываю платформу STaaS.
Тыкаю Kubernetes палкой с 2019 года.
Рад ответить на вопросы tg: @igor_sde

If you deploy databases in Kubernetes, we recommend following these 4 extra release steps:

@memenetes

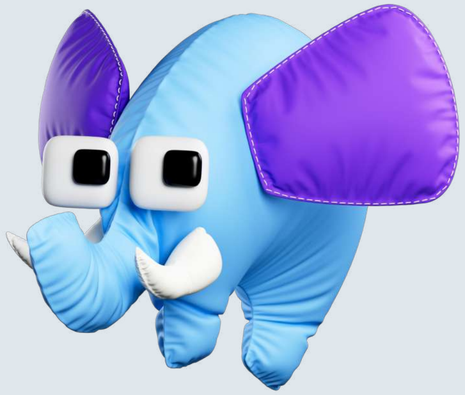


Twitter: @memenetes

Базы в Kubernetes — это не страшно (почти)

Подложив достаточно соломы, можно автоматизировать значительную часть работы по администрированию БД за счёт контейнеров и их оркестрации.

А это позволит ещё быстрее расти и масштабировать бизнес.



Дисклеймер:
Предполагаю, что вы знаете базовые концепции Kubernetes (Pod, Deployment). Иначе не вместимся в формат, сорян(

Абстракции Kubernetes для Stateful-приложений

01/03

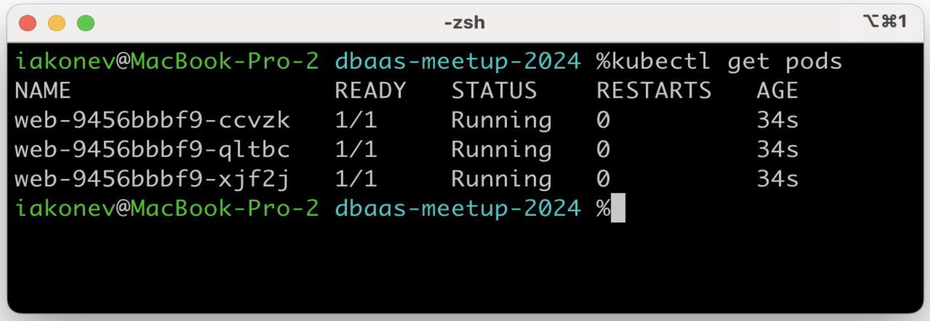
StatefulSet. Сравнение с Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 3
  template:
    spec:
      containers:
      - name: nginx
        image: registry.k8s.io/nginx-slim:0.8
        ports:
        - containerPort: 80
          name: web
```



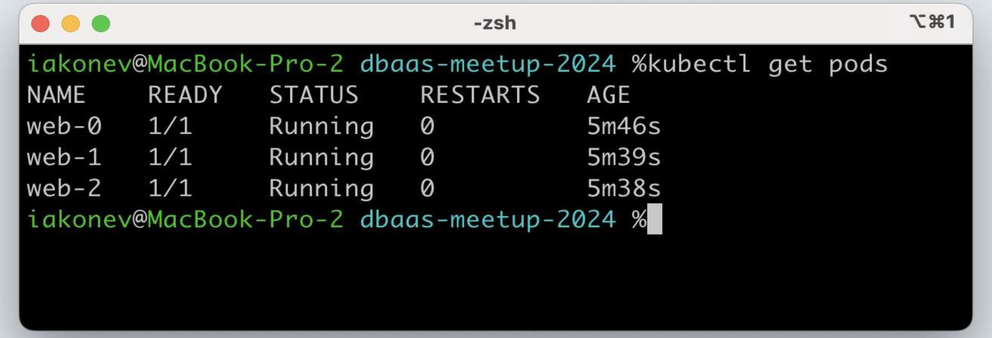
```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  replicas: 3
  template:
    spec:
      containers:
      - name: nginx
        image: registry.k8s.io/nginx-slim:0.8
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: www
          mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
  - metadata:
      name: www
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: "my-storage-class"
      resources:
        requests:
          storage: 1Gi
```

Deployment



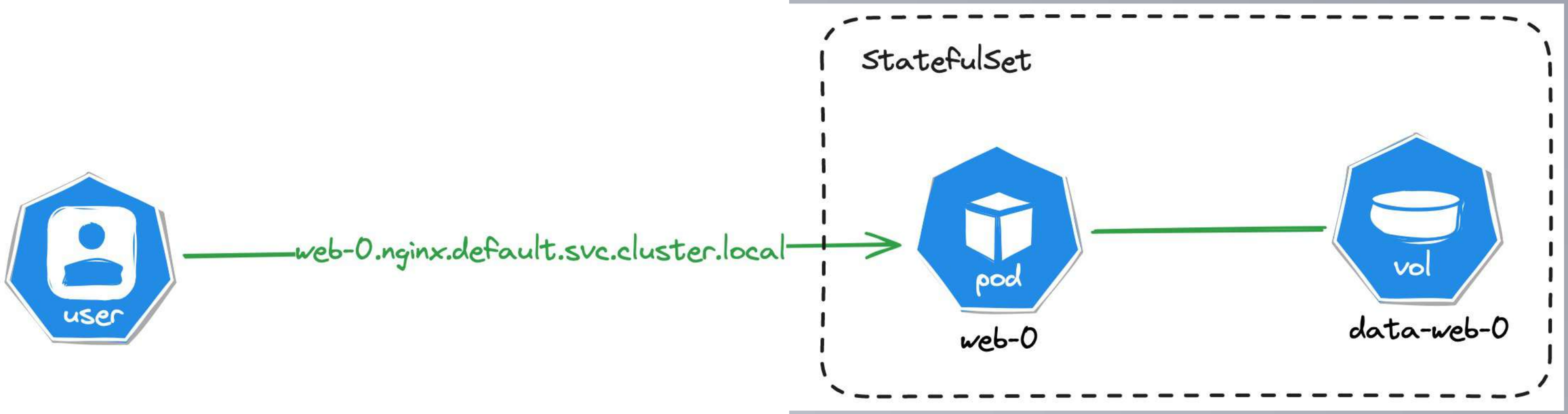
- Все поды разворачиваются одновременно.
- Имена подов случайны и формируются по шаблону:
  <имя deployment>-<rs hash>-<pod hash>.

StatefulSet



- По умолчанию разворачивает по одному поду последовательно.
- Имена подов персистентны и формируются по шаблону:
  <имя statefulset>-<порядковый номер>.

Доступ в Pod персистентного приложения



Использование стабильных сетевых идентификаторов

Поскольку Pod в рамках StatefulSet имеет стабильное имя, пользователь может подключаться к нему, не боясь потери данных в случае рестарта.

StatefulSet. Сравнение с Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 3
  template:
    spec:
      containers:
      - name: nginx
        image: registry.k8s.io/nginx-slim:0.8
        ports:
        - containerPort: 80
          name: web
```



```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  replicas: 3
  template:
    spec:
      containers:
      - name: nginx
        image: registry.k8s.io/nginx-slim:0.8
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: www
          mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
  - metadata:
      name: www
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: "my-storage-class"
      resources:
        requests:
          storage: 1Gi
```

StatefulSet. Сравнение с Deployment

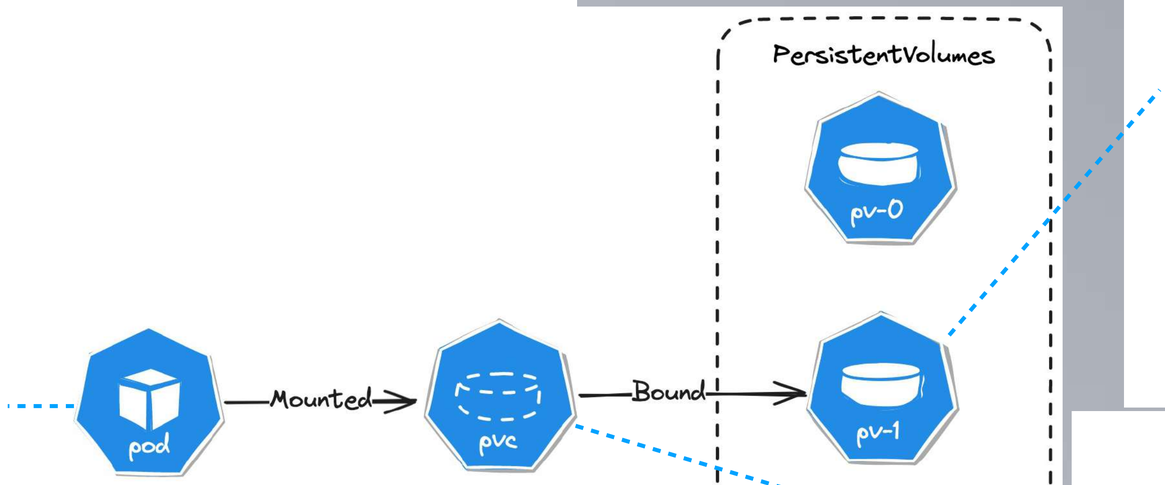
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 3
  template:
    spec:
      containers:
      - name: nginx
        image: registry.k8s.io/nginx-slim:0.8
        ports:
        - containerPort: 80
          name: web
```



```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  replicas: 3
  template:
    spec:
      containers:
      - name: nginx
        image: registry.k8s.io/nginx-slim:0.8
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: www
          mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
  - metadata:
      name: www
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: "my-storage-class"
      resources:
        requests:
          storage: 1Gi
```

Статический provisioning томов в Kubernetes

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-app
spec:
  containers:
  - name: app
    # ...
    volumeMounts:
    - name: persistent-storage
      mountPath: /data
  volumes:
  - name: persistent-storage
    persistentVolumeClaim:
      claimName: pvc-0
```

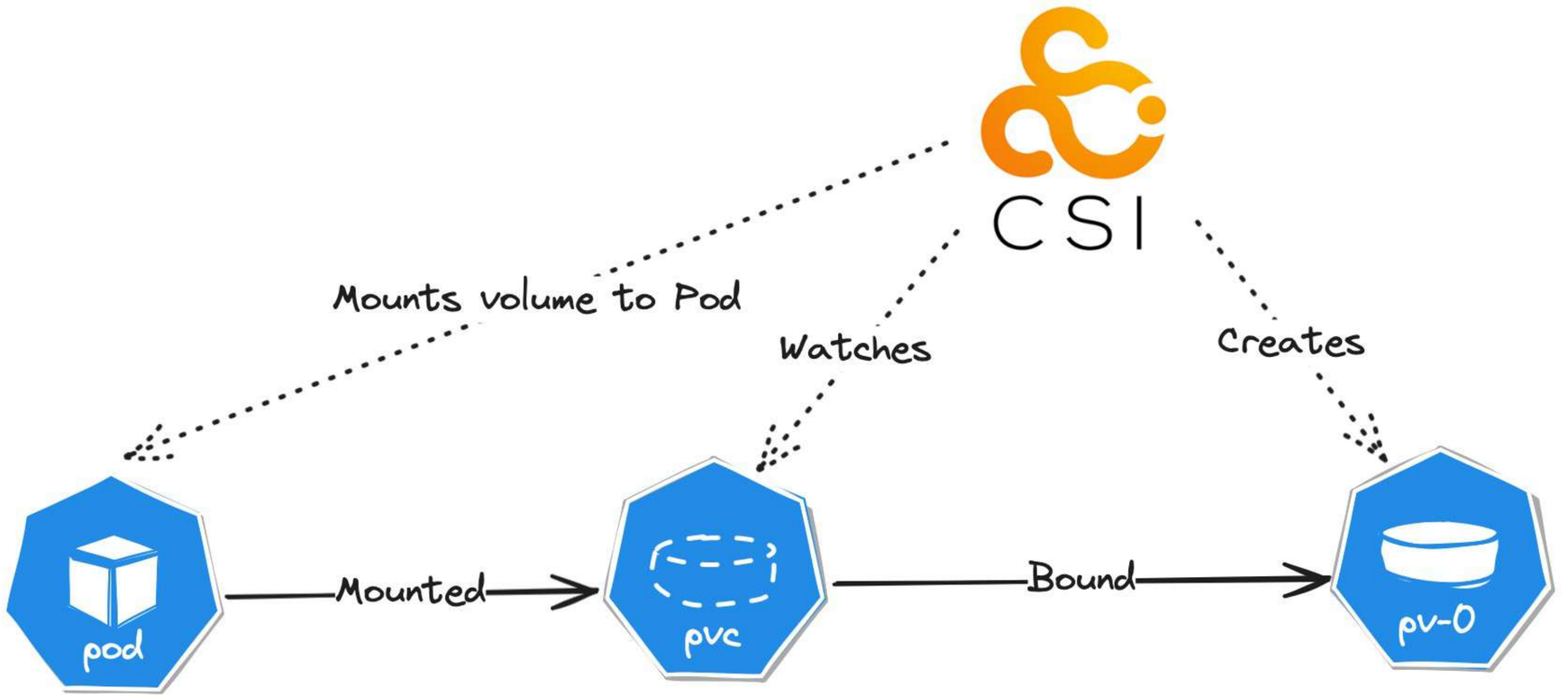



```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-1
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  # local/iSCSI/hostPath...
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-0
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

О чем картинка

Динамический provisioning томов в Kubernetes



О чем картинка

StatefulSet. Сравнение с Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 3
  template:
    spec:
      containers:
      - name: nginx
        image: registry.k8s.io/nginx-slim:0.8
        ports:
        - containerPort: 80
          name: web
```



```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  replicas: 3
  template:
    spec:
      containers:
      - name: nginx
        image: registry.k8s.io/nginx-slim:0.8
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: www
          mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
  - metadata:
      name: www
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: "my-storage-class"
      resources:
        requests:
          storage: 1Gi
```

Что из этого мы используем

02 / 03

StatefulSet с базой данных

Все БД в DBaaS деплоятся как StatefulSet.

Рассмотрим типичное наполнение манифеста на примере Redis:

replicas — чаще всего это 1 из-за требования к нескольким дата-центрам, подробнее в следующем докладе.

initContainers — инициализация: доставка первичных секретов, задание ограничений сети и т.д.

containers — сама база и её сайдкары: HA-агенты, экспортеры метрик и т.п.

resources — эффективная утилизация кластера.

volumes — помимо PV, нам нужны ConfigMap и emptyDir.

volumeClaimTemplates — PVC для хранения данных.

TopoLVM: почему мы выбрали этот CSI

01

Минимум вложений при переезде с LXC

02

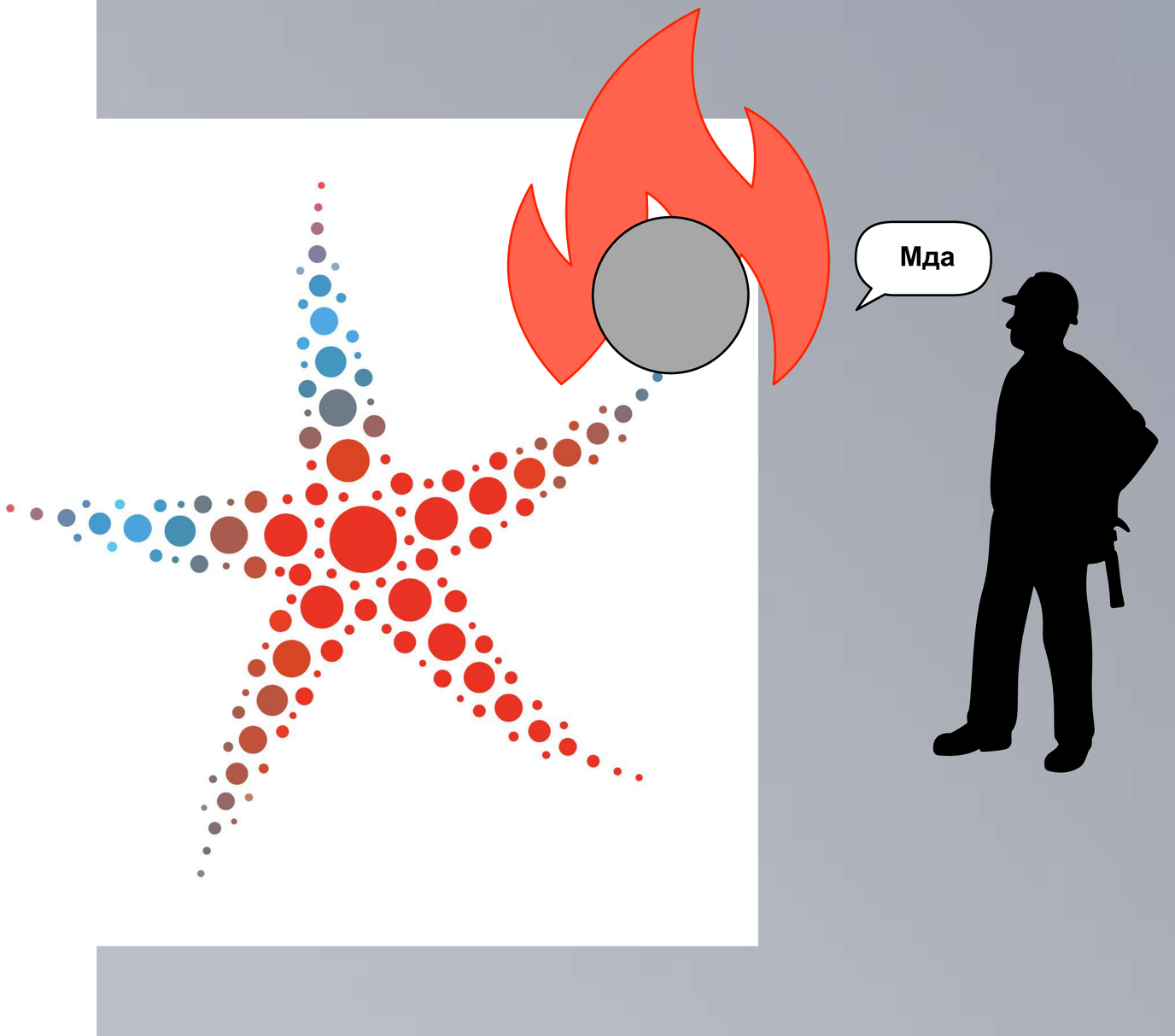
Изоляция на уровне блочных девайсов

03

Dynamic Provisioning + Volume Expansion без даунтайма

04

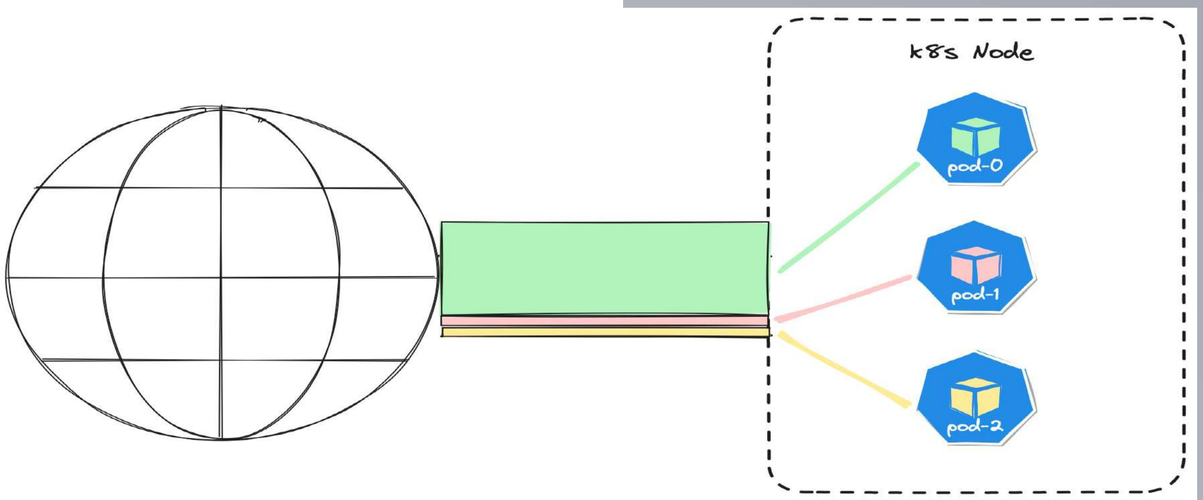
Интеграция с шедулером и учёт свободного места на нодах



Наши велосипеды

03/03

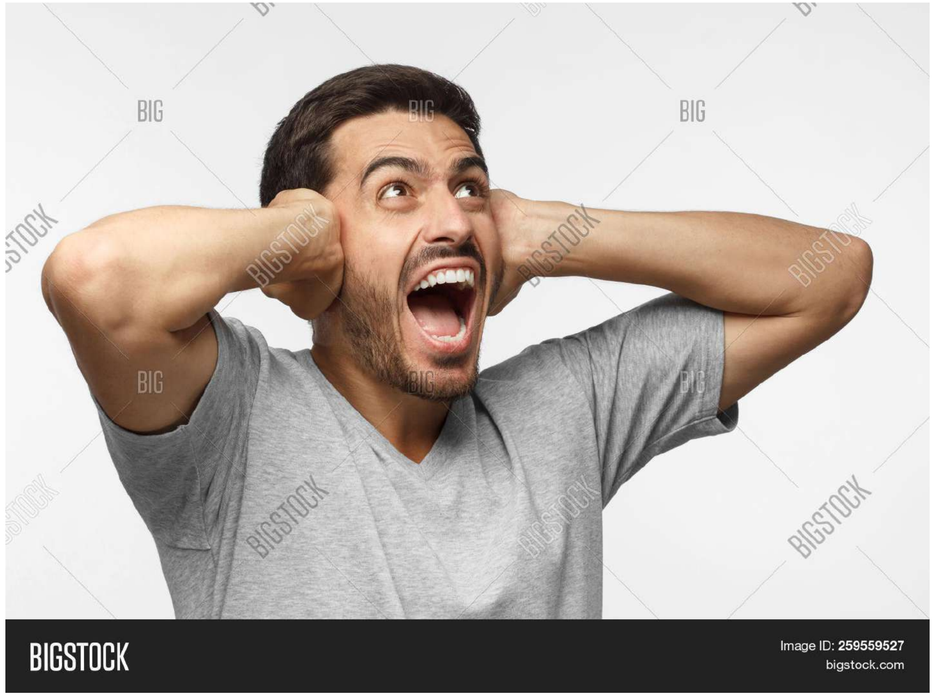
Проблема шумных соседей



Развёрнутые на одном узле поды неизбежно разделяют ресурсы узла

Проблемы возникают, когда некоторые поды начинают чрезмерно потреблять ресурс, занимая всю полосу пропускания. Это негативно сказывается на «соседях».

Проблема шумных соседей



Использование стабильных сетевых идентификаторов

Проблема шумных соседей. Дисковое I/O

ALLO IOBA ETO TI?

В коммунальных кластерах DBaaS базы делят диски между собой.

Для равномерной утилизации их пропускной способности мы придумали своё решение.

ioba работает на основе механизма ядра cgroups v2 (control groups) и запускается как DaemonSet в DBaaS-кластерах.

ioba отслеживает, когда Pod задеплоился на ноду, получает из конфига желаемые ограничения I/O и устанавливает их, используя cgroups.

Ограничения задаются в виде Custom Resource IOLimit, где описывается, какие дисковые лимиты имеют контейнеры в томах.

```yaml
apiVersion: dbaas.dbaas.avito.ru/v1alpha1
kind: IOLimit
metadata:
   name: disk-io-limit
spec:
   storageName: redis1
   replicaSetName: rs001
   containers:
      - name: db
        volumes:
           - name: data
             reads:
               iops: 200
               bandwidth: 200M
             writes:
               iops: 200
               bandwidth: 200M
```

Volume Expansion

Сейчас в StatefulSet нет нативной поддержки расширения томов. Идут активные споры**.**

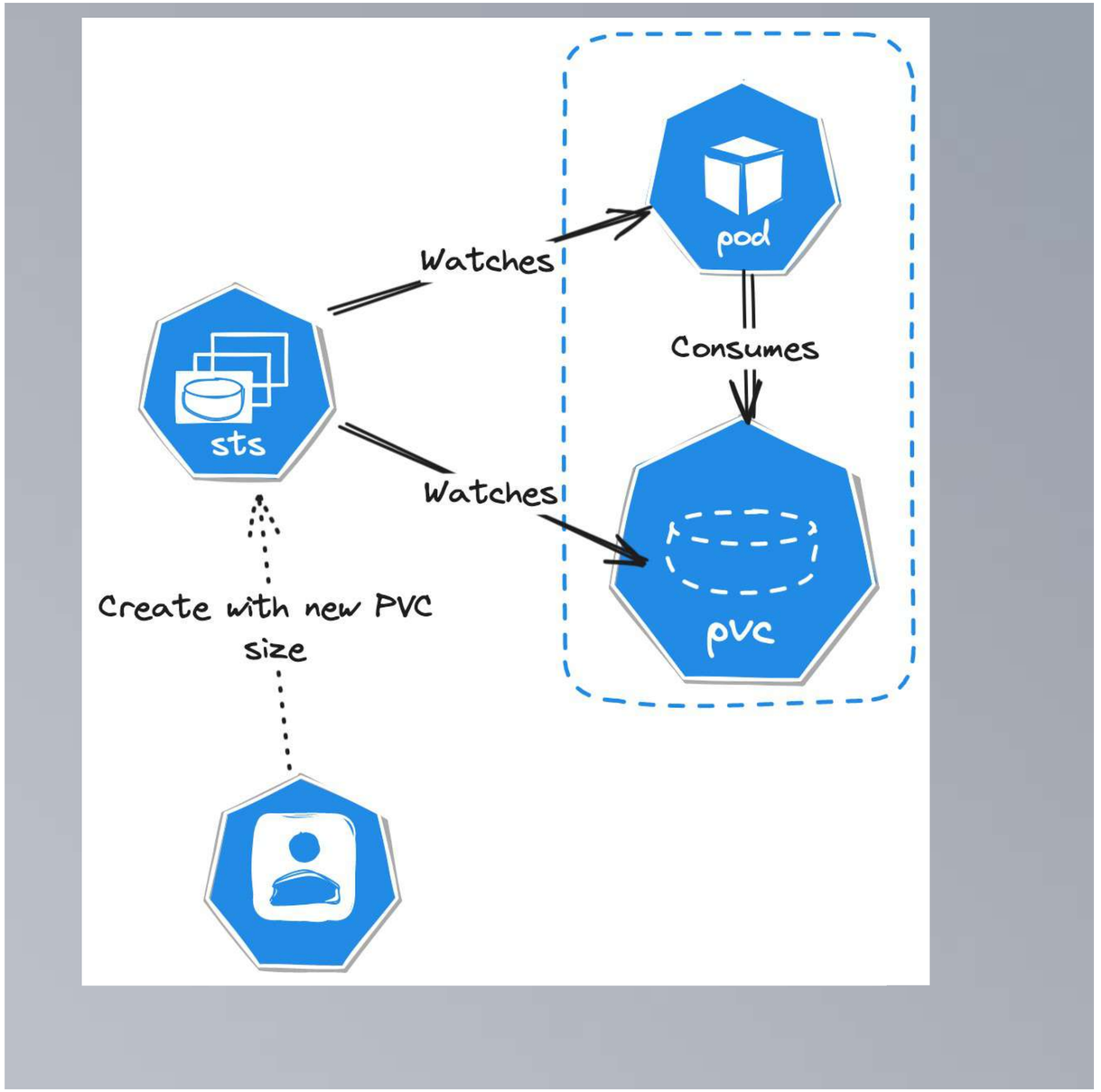
Но нам надо как-то с этим жить, поскольку место в базах имеет свойство заканчиваться.

При попытке обновить размер томов в манифесте StatefulSet, возникнет ошибка.

Применяем известный в сообществе хак с изменением размера PersistentVolumeClaim и пересозданием StatefulSet.

Для расширения томов CSI должен поддерживать фичу *allowVolumeExpansion.*

Удаляем StatefulSet с флагом *--cascade=orphan*, чтобы не трогать поды.



Дешедулинг

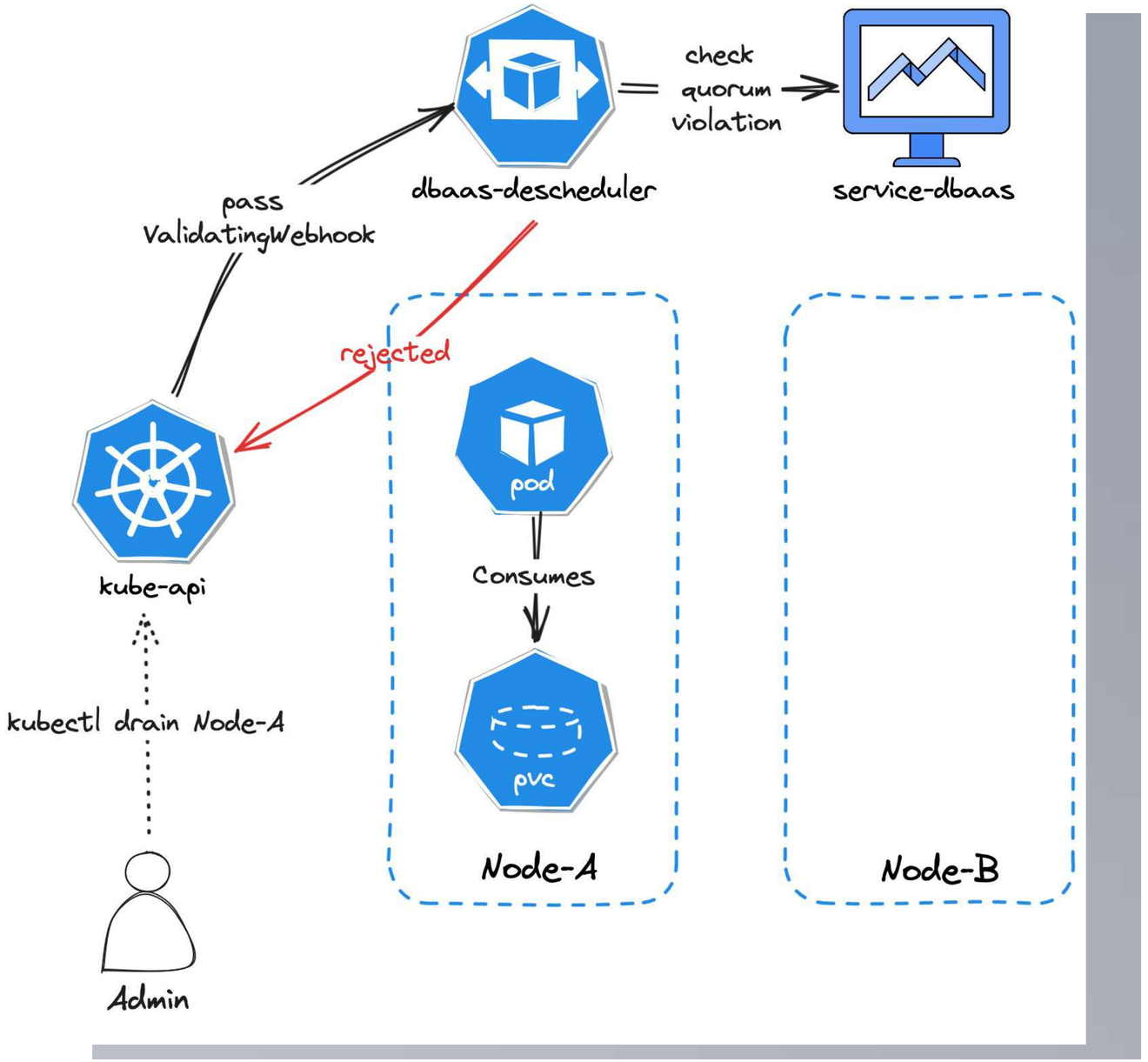
Нужно уметь выселять поды с определённой ноды в кластере. Например, для обслуживания.

В DBaaS это не так просто, поскольку БД используют локальные диски для хранения данных.

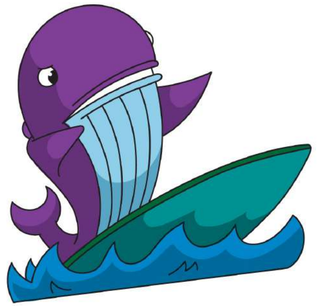
Придумали свой механизм *dbaas-descheduler*, который интегрируется с *kubectl drain* и *Eviction API.*

При попытке эвакуации ноды срабатывает *ValidationWebhook,* который заводит задачи на выселение подов БД с ноды.

dbaas-descheduler расселяет БД с сохранением гарантий платформы. PV пересоздаются на других нодах и наливаются за счёт репликации.



Заключение



Kubernetes ещё есть, куда развиваться в плане поддержки Stateful-приложений.

Но нам удалось построить на нём DBaaS и автоматизировать жизненный цикл БД в Авито.

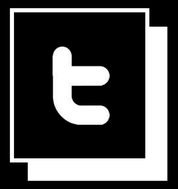
Останавливаться на достигнутом не планируем, задач ещё непаханое поле!

А где же Операторы?

avito.tech Москва | 2024

Игорь Конев

Е6-инженер DBaaS



Запрещенограм: igor_sde
LinkedIn: linkedin.com/in/igor-konev

Telegram: @igor_sde

Представьте, каких чудес можно достичь, автоматизируя процессы, которые кажутся непригодными для этого. Автоматизация — это ключ к масштабированию и обеспечению надёжности!(с) Джейсон Стэтхэм