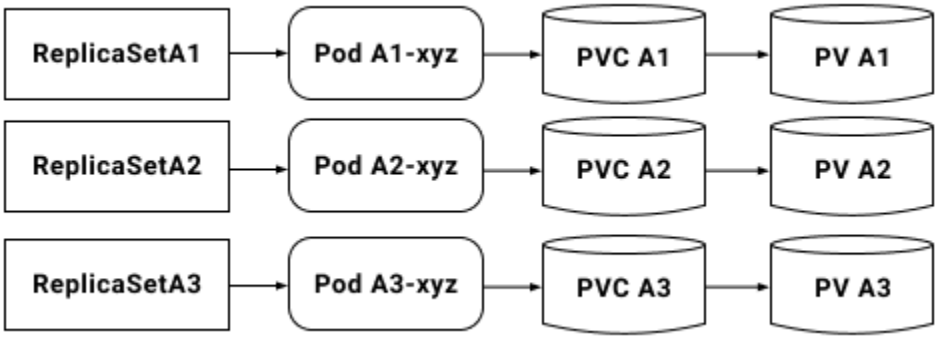
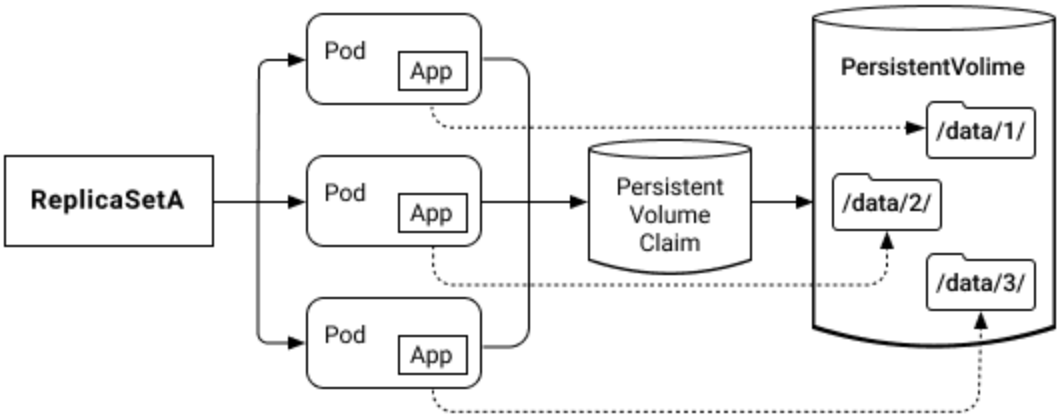
KUB 21: Запуск Stateful приложений

Описание:

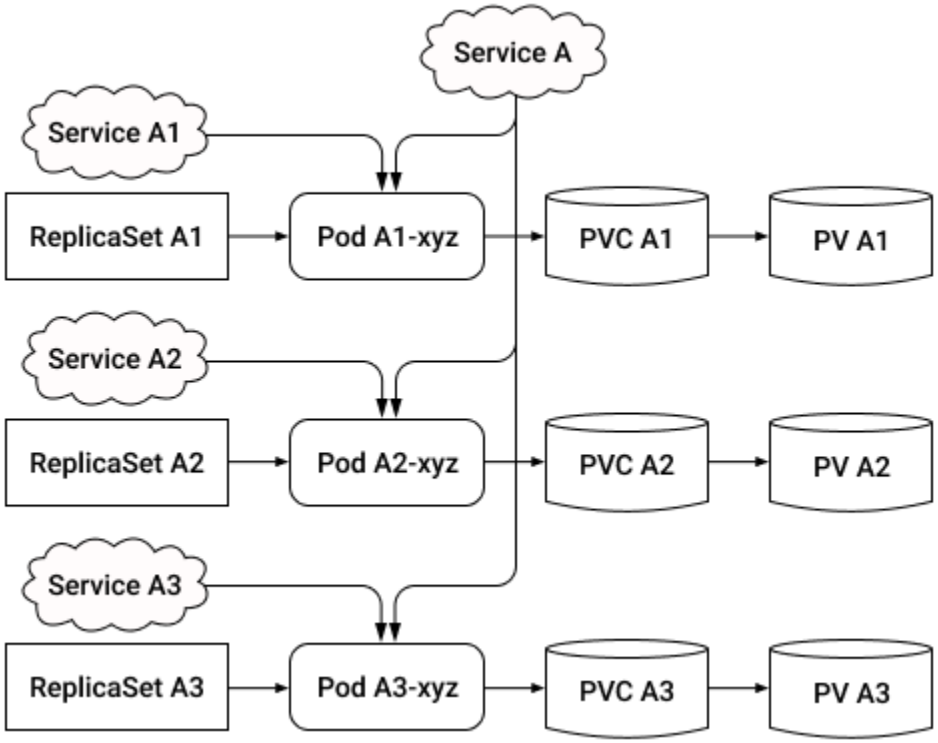
Этот контроллер служит для запуска приложений с состоянием (stateful). Правильный пример — kafka или база данных. Всякому контейнеру необходимо знать имена соседних узлов и хранить свои данные. Такого поведения непросто добиться с `ReplicaSet`. Если в описании `ReplicaSet` есть `PersistentVolumeClaim`, то все pods будут запрашивать один PVC. К тому же при перезапуске pod пропадет его IP address и hostname.

Само собой, можно обойтись `ReplicaSet`, но посмотрите, сколько объектов будет нужно поддерживать и следить за их состоянием.

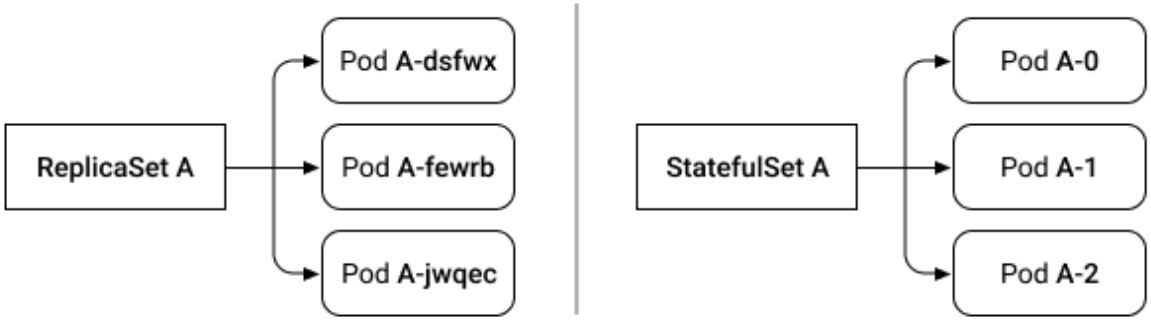




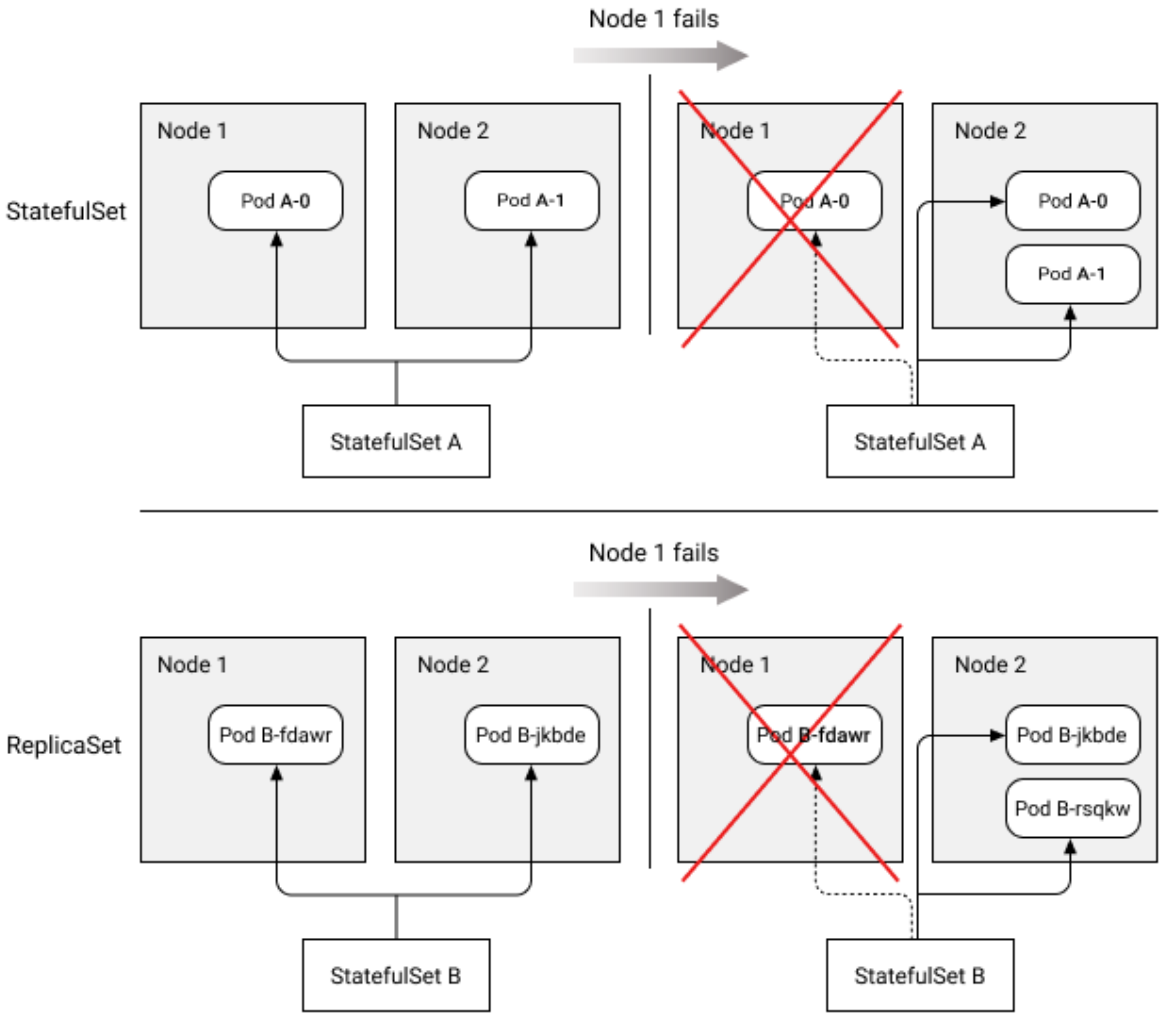
Также придется сделать `service` для каждого `ReplicaSet` отдельно и общий для всех.



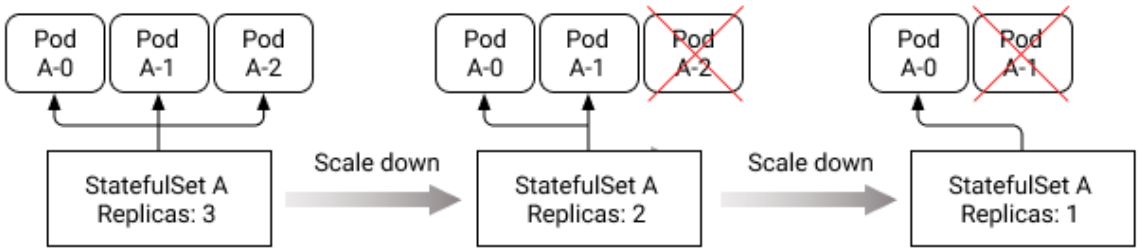
Чтобы не испытывать страдания при запуске подобных приложений, сделали контроллер `StatefulSet`.



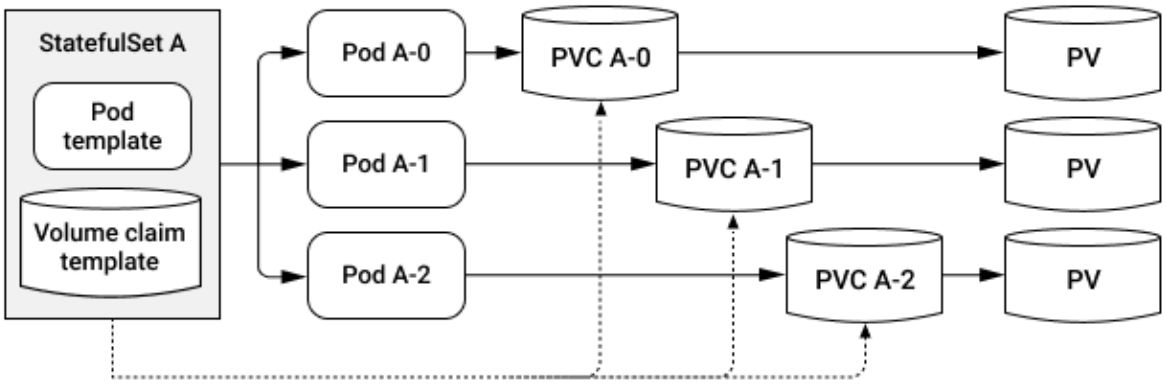
Каждый `pod` имеет имя, но только поды `StatefulSet` доступны по DNS и имеют предсказуемое имя.



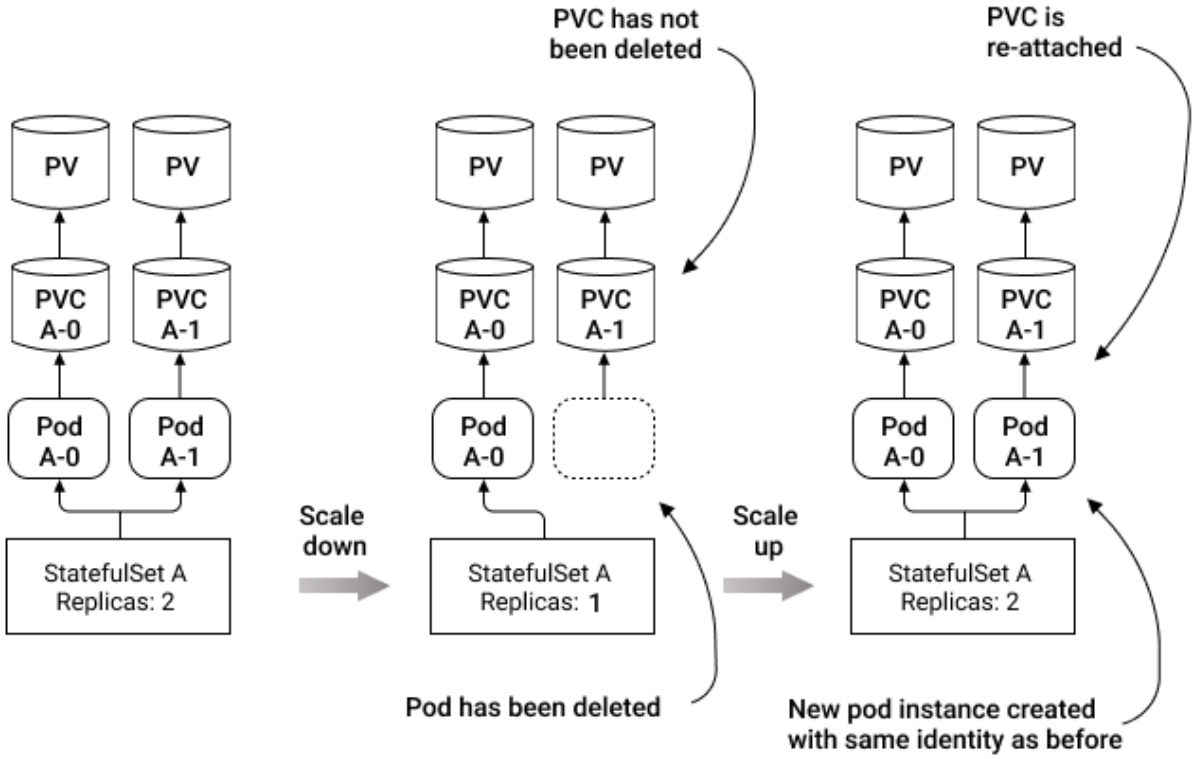
Имя сохраняется при перезапуске. Немного по-иному работает масштабирование `StatefulSet`. В отличие от `ReplicaSet`, оно происходит по порядку. `ReplicaSet` просто запускает дополнительные копии.



Точно также создаются отдельные `PVC` для каждого `pod` из `replicaset`.



Но при масштабировании вниз `PVC` не удаляется.



Пример описания `StatefulSet`:

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: nginx-ss
spec:
  serviceName: nginx-ss
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
```

```
        ports:
        - name: http
          containerPort: 8080
        volumeMounts:
        - name: data
          mountPath: /var/data
  volumeClaimTemplates:
  - metadata:
      name: data
    spec:
      storageClassName: yc-network-ssd
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 2Gi
```

Заметьте, дополнительно описывается `volumeClaimTemplates`. Посмотрите описание созданного `pod`, попробуйте удалить его, изменить количество реплик и понаблюдать за созданием/удалением `pod`.

Параметр `spec.serviceName` особенный. При запуске `statefulset` он незаменим. Создает запись в DNS с SRV-записями по всем pod из такого `statefulset`. `StatefulSet` при падении узла кластера ведет себя иначе, чем `replicaset`.

Предположим, что на трех узлах кластера запущено три `pod` из `statefulset`. Что произойдет при падении узла с нулевым pod из `statefulset`? При просмотре `kubectl get sts` и `kubectl get po` мы увидим `status=unknown` у этого `pod`. Почему он не перезапустился? Дело в том, что kubelet удаляет поды только по изменению состояния `etcd` и сообщениям с мастера.

Если попробовать удалить `pod` руками `kubectl delete po sts-pod-0`, то ничего не изменится. Он будет удален только после перезапуска узла кластера. Либо можно принудительно удалить pod: `--force --grace-period=0`. В этом случае `pod` будет перезапущен на другом узле кластера.

Как правило, в связке с `StatefulSet` используется `Headless Service`, что позволяет на уровне приложения заниматься обнаружением экземпляров сервисов при помощи DNS-запросов (мы обсуждали этот тип Service ранее).

Полезные ссылки:

- StatfulSets (official docs)
- StatefulSet Basics tutorial (official docs)

Задание:

1. Настройте nfs provisioner по аналогии с предыдущим заданием (не обязательно следовать таким же именам - главное чтобы pv успешно создавались в кластере).
2. Создайте statefulset с именем `kafka-sts` в namespace `default`, который будет запускать 3 пода kafka и создавать для них отдельные pvc `kafka-pvc` на 1 Гб каждый.
3. Отправьте задание на проверку.