Хранение в K8s. Часть 2

PersistentVolume, PersistentVolumeClaim, StorageClass

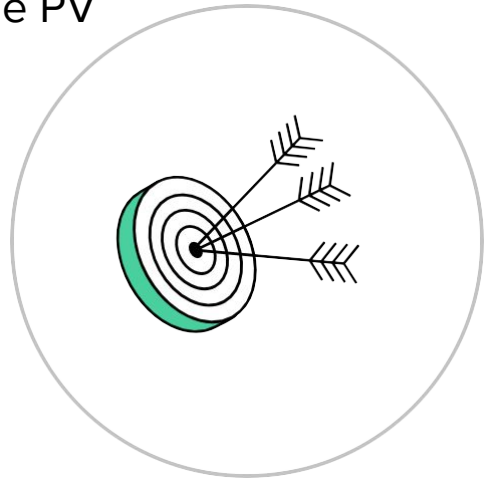
Кирилл Касаткин
DevOps-инженер, Renue

Кирилл Касаткин

DevOps-инженер, Renue

Цели занятия

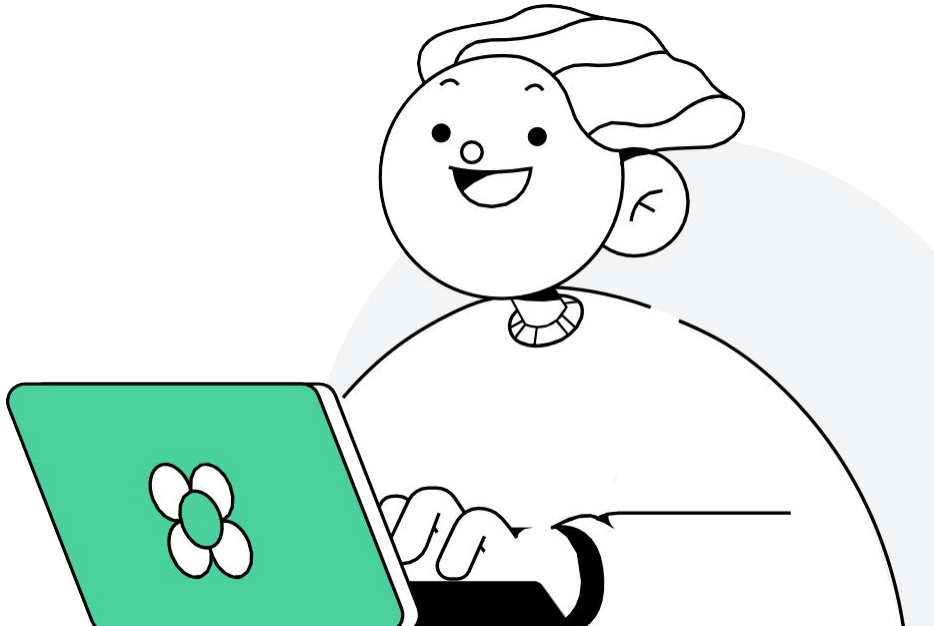
- Узнать:
  - что такое PersistentVolume (PV) и PersistentVolumeClaim (PVC) и чем они отличаются
  - как создаются PV и подключаются к Pod’ам
- Познакомиться с классами хранения
- Понять, каким образом можно автоматизировать создание PV
- Разобрать примеры манифестов объектов K8s



План занятия

1. Типы Volume
2. Persistent Volume
3. Persistent Volume Claim
4. Storage Class
5. Итоги
6. Домашнее задание

*Нажми на нужный раздел для перехода



Вспоминаем прошлое занятие

Вопрос: где будут храниться файлы, определенные volume типа hostPath? Будут ли они удалены после удаления пода?

Вспоминаем прошлое занятие

Вопрос: где будут хранится файлы, определенные volume типа hostPath? Будут ли они удалены после удаления пода?

Ответ: на ноде. Нет, не будут удалены

Типы Volume

1

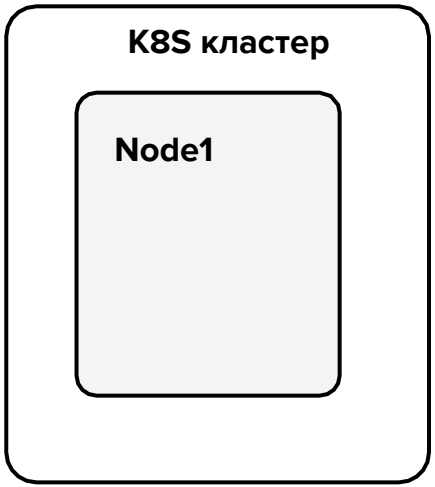
Типы Volume

Существует много типов Volume. Условно их можно разделить на:

- локальные, расположенные непосредственно на ноде, где находится под
- **остальные**, определяемые при помощи PersistentVolume

Типы Volume

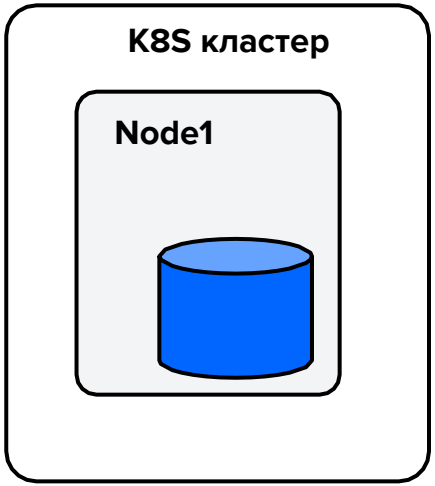
Volume можно рассматривать как абстрактный ресурс кластера, как CPU или RAM, который используется для хранения данных (Storage)



Физически может быть:

Типы Volume

Volume можно рассматривать как абстрактный ресурс кластера, как CPU или RAM, который используется для хранения данных (Storage)

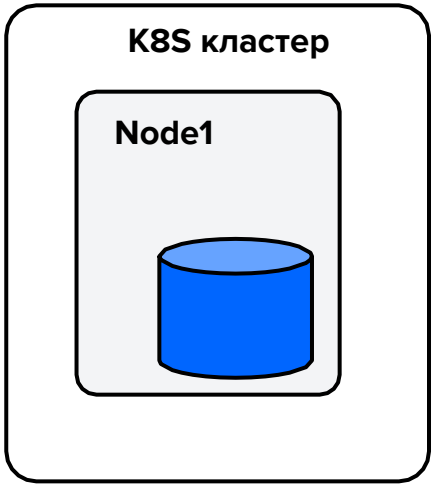


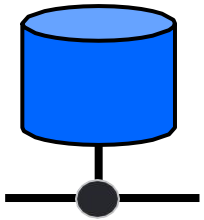
Физически может быть:

- Локальное хранение (например, HDD)

Типы Volume

Volume можно рассматривать как абстрактный ресурс кластера, как CPU или RAM, который используется для хранения данных (Storage)



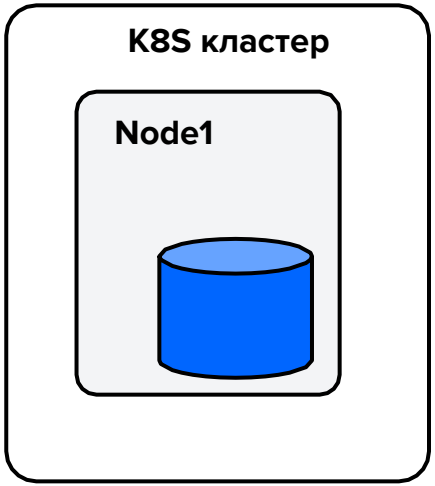


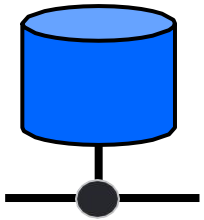
Физически может быть:

- Локальное хранение (например, HDD)
- Сетевое хранение (например, NFS)

Типы Volume

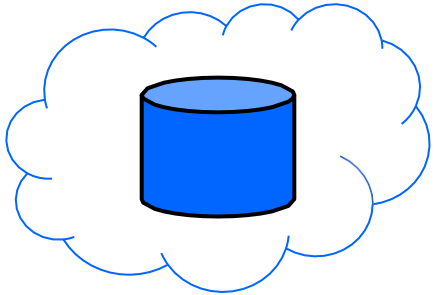
Volume можно рассматривать как абстрактный ресурс кластера, как CPU или RAM, который используется для хранения данных (Storage)



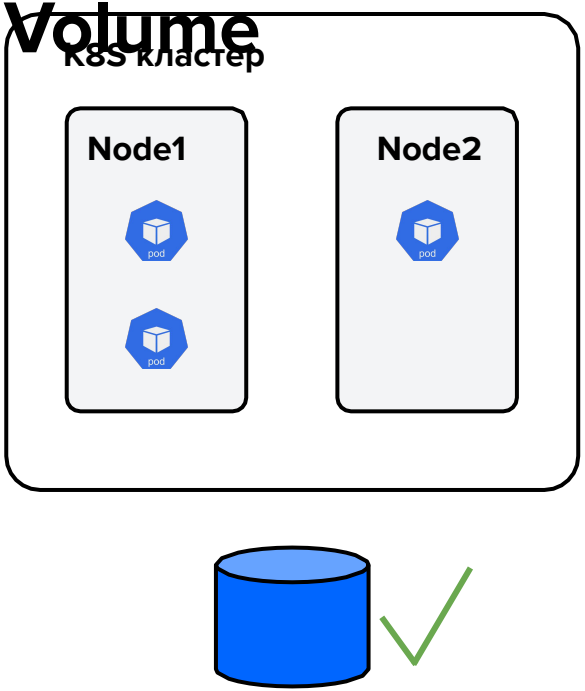


Физически может быть:

- Локальное хранение (например, HDD)
- Сетевое хранение (например, NFS)
- Облачное хранение (например, S3)



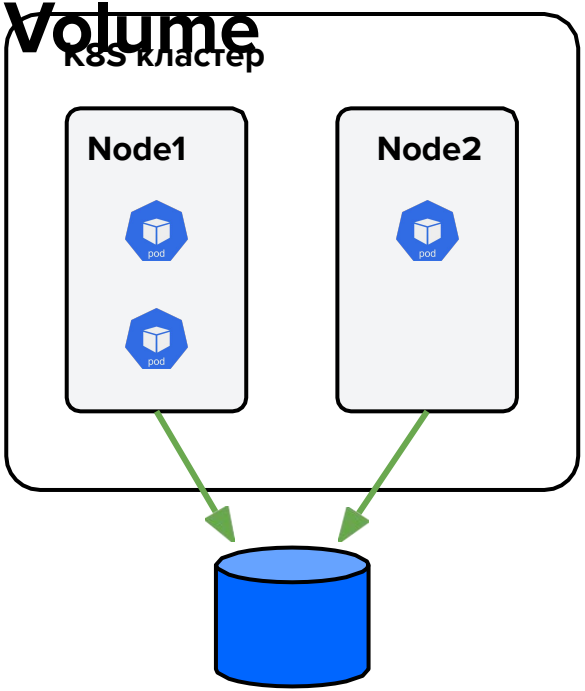
Типы Volume



При этом Volume:

- Не зависит от подов и нод

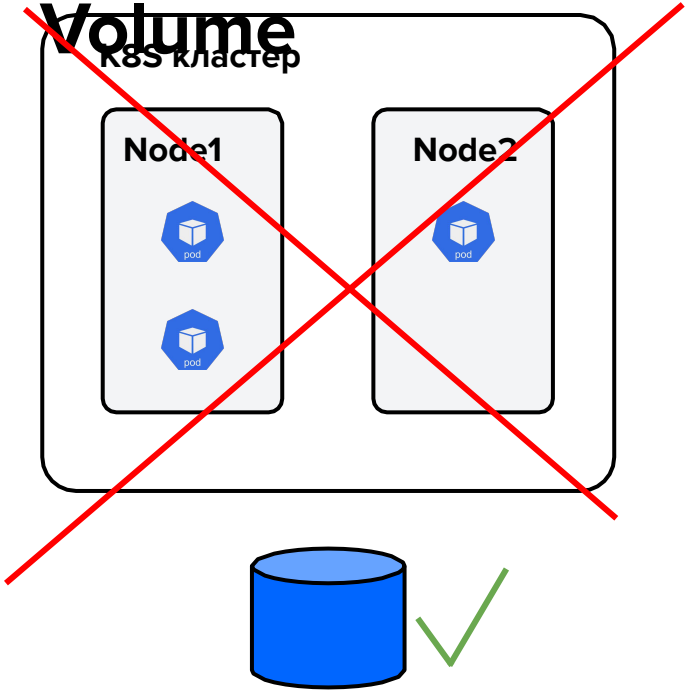
Типы Volume



При этом Volume:

- Не зависит от подов и нод
- Должен быть доступен из всех нод

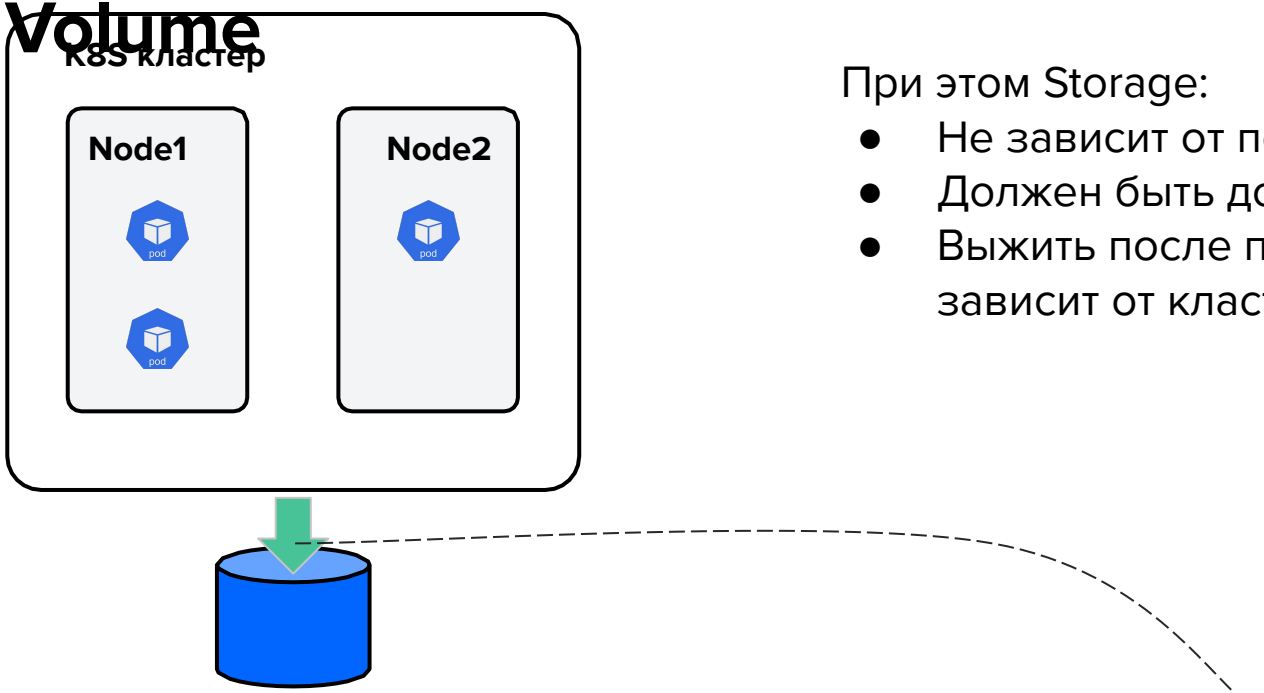
Типы Volume



При этом Volume:

- Не зависит от подов и нод
- Должен быть доступен из всех нод
- Выжить после падения кластера (не зависит от кластера)

Типы Volume

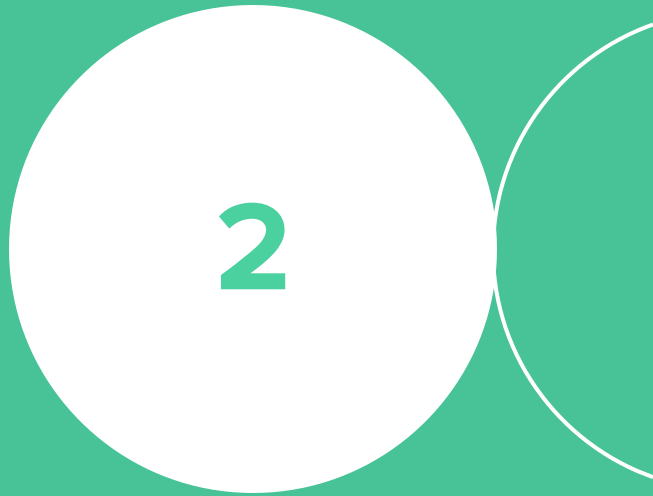


При этом Storage:

- Не зависит от подов и нод
- Должен быть доступен из всех нод
- Выжить после падения кластера (не зависит от кластера)

Кластер K8S обращается к Storage через **Container Storage Interface** (CSI)

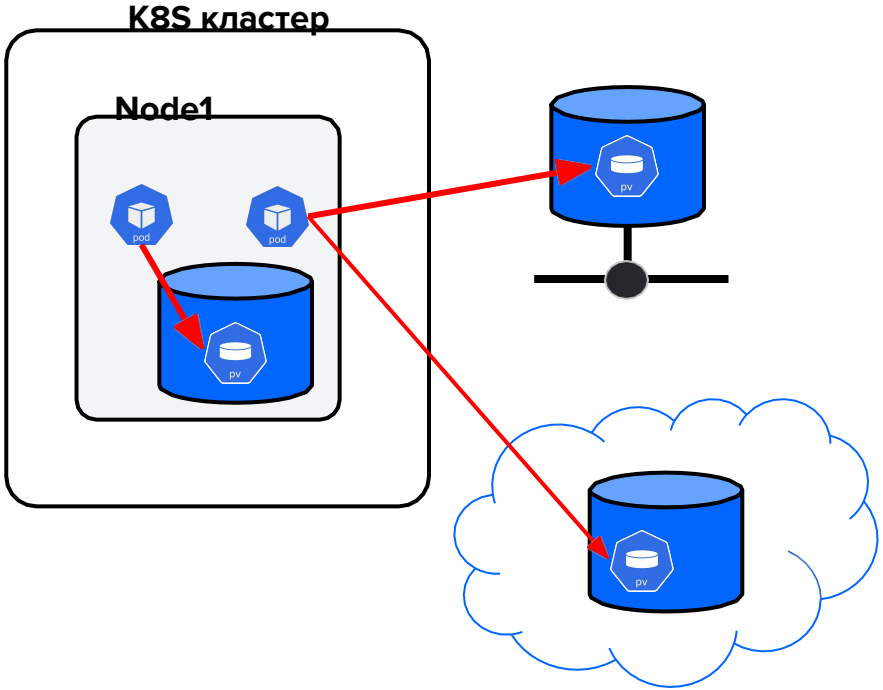
Persistent Volume



Persistent Volume (PV) — объект K8s, который позволяет обращаться с хранилищем как к абстрактным ресурсом

Persistent Volume (PV) —постоянный том

Pod может использовать абстрактный ресурс хранения, а PV выступает как интерфейс или плагин к типу хранения.



Persistent Volume (PV) — постоянный том

Типы PV, которые поддерживает K8S — описаны [здесь]

Types of Persistent Volumes

PersistentVolume types are implemented as plugins. Kubernetes currently supports the following plugins:

- `cephfs` - CephFS volume
- `csi` - Container Storage Interface (CSI)
- `fc` - Fibre Channel (FC) storage
- `hostPath` - HostPath volume (for single node testing only; WILL NOT WORK in a multi-node cluster; consider using `local` volume instead)
- `iscsi` - iSCSI (SCSI over IP) storage
- `local` - local storage devices mounted on nodes.
- `nfs` - Network File System (NFS) storage
- `rbd` - Rados Block Device (RBD) volume

The following types of PersistentVolume are deprecated. This means that support is still available but will be removed in a future Kubernetes release.

- `awsElasticBlockStore` - AWS Elastic Block Store (EBS) (**deprecated** in v1.17)
- `azureDisk` - Azure Disk (**deprecated** in v1.19)
- `azureFile` - Azure File (**deprecated** in v1.21)
- `cinder` - Cinder (OpenStack block storage) (**deprecated** in v1.18)
- `flexVolume` - FlexVolume (**deprecated** in v1.23)
- `gcePersistentDisk` - GCE Persistent Disk (**deprecated** in v1.17)
- `portworxVolume` - Portworx volume (**deprecated** in v1.25)
- `vsphereVolume` - vSphere VMDK volume (**deprecated** in v1.19)

Persistent Volume (PV) — постоянный том

PV, как и другие объекты K8S, определяется манифестом в формате YAML

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfs-volume
spec:
  capacity:
     storage: 5Gi
  volumeMode: Filesystem
  accessModes:
  - ReadWriteOnce
    persistentVolumeReclaimPolicy:
    Recycle storageClassName: slow
  mountOptions:
-       hard
  -   nfsvers=4.0
     nfs:
     path: /path/on/nfs/server
     server: nfs-server-ip-address
```

Пример конфигурации NFS

Persistent Volume (PV) — постоянный том

PV, как и другие объекты K8S, определяется манифестом в формате YAML

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: google-volume
  labels:
    failure-domain.beta.kubernetes.io/zone: us-central1-a  us-central1-b
spec:
  capacity:
    storage: 400Gi
  accessModes:
    - ReadWriteOnce
  gcePersistentDisk:
     pdName: my-data-disk
     fsType: ext4
```

Пример конфигурации Google Cloud

Persistent Volume (PV) — постоянный том

PV, как и другие объекты K8S, определяется манифестом в формате YAML

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: local-volume
spec:
  capacity:
    storage: 100Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  storageClassName: local-storage
  local:
    path: /mnt/disk/ssd1
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - example-node
```

Пример конфигурации локального диска

Persistent Volume (PV) — постоянный том

AccessMode (режимы доступа к томам)

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv1
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /data/pv1
```

- **ReadWriteOnce** — (RWO) volume может быть примонтирован в режиме чтения и записи только к одной ноде
- **ReadOnlyMany** — (ROX) volume может быть примонтирован в режиме только чтения ко множеству нод
- **ReadWriteMany** — (RWX) volume может быть примонтирован в режиме чтения и записи ко множеству нод
- **ReadWriteOncePod** — (RWOP) volume может быть примонтирован в режиме чтения и записи только к одному поду

Persistent Volume (PV) —постоянный том

ReclaimPolicy — определяет как будут использованы ресурсы после удаления PV

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv1
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /data/pv1
  persistentVolumeReclaimPolicy: Retain
```

- **Retain** — после удаления PV ресурсы из внешних провайдеров автоматически не удаляются.
- **Delete** — после удаления PV ресурсы из внешних провайдеров автоматически удаляются (работает только в облачных Storage)
- **Recycle** — [rm-rf] автоматически удаляет ресурсы (устаревшее)

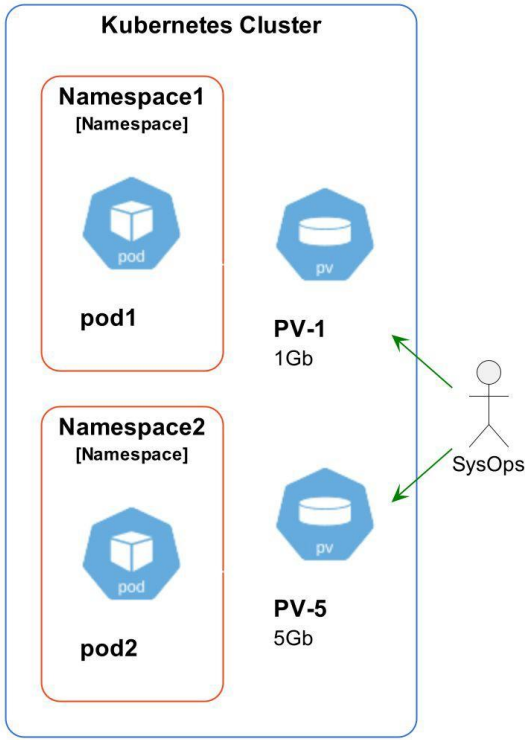
Persistent Volume Claim

3

Persistent Volume Claim (PVC) — запрос (заявка) на выделение тома

Persistent Volume Claim (PVC) — заявка на том

PV не определяется в настройках Pod’а и Namespace

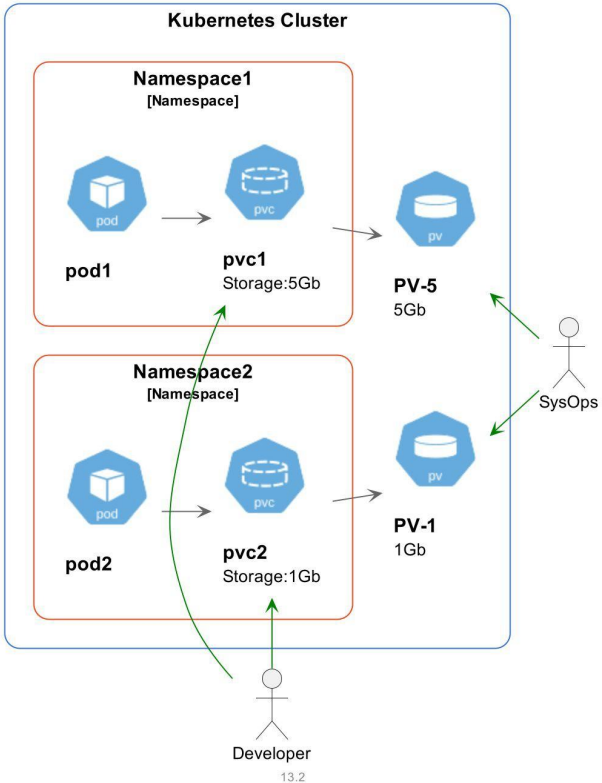


Настройкой хранения занимается Системный администратор:

- Настройка типа хранения (NFS, Cloud, Local)
- Настойка **PV**

Persistent Volume Claim (PVC) — заявка на том

PV не определяется в настройках Pod’a и Namespace



Настройкой хранения занимается Системный администратор:

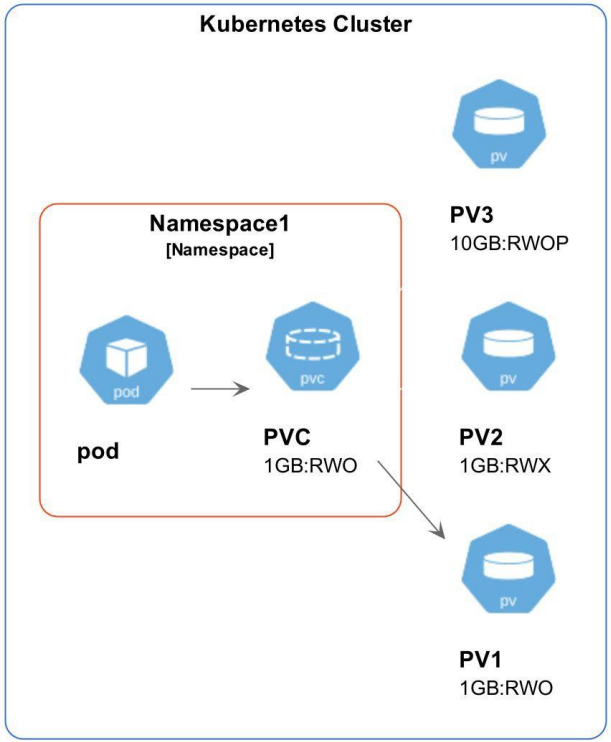
- Настройка типа хранения (NFS, Cloud, Local)
- Настойка **PV**

Настройкой приложения и запроса на хранение занимается Разработчик

- Настройка Pod’a, Namespace
- Настройка **PVC**

Persistent Volume Claim (PVC) — заявка на том

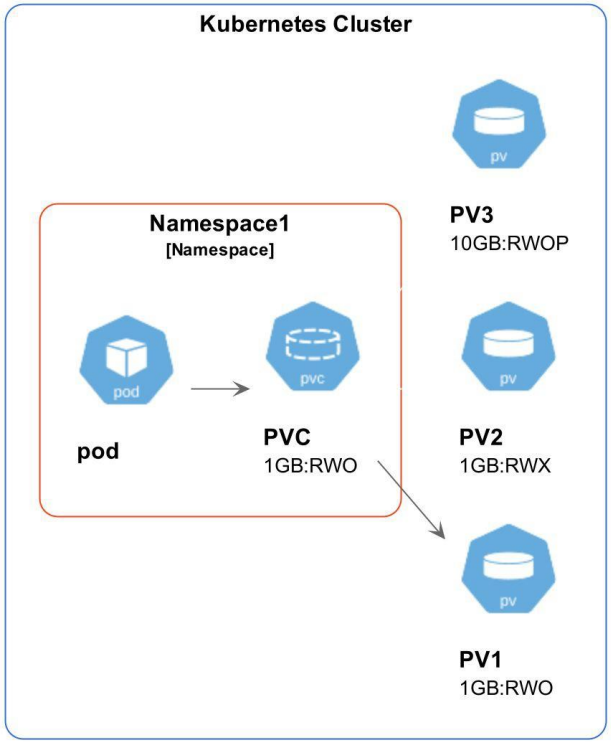
Инициализация PV



- в спецификации Pod’а указывается ссылка на **PVC** (запрос на том)

Persistent Volume Claim (PVC) — заявка на том

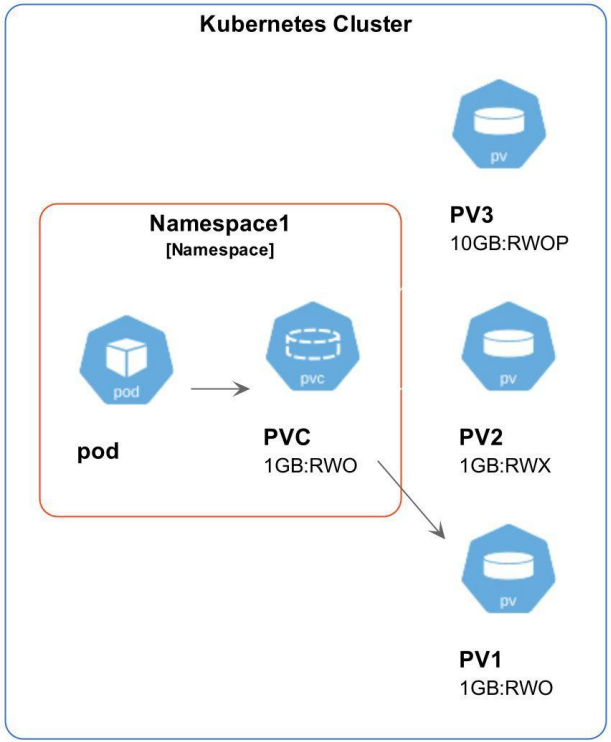
Инициализация PV



- в спецификации Pod’а указывается ссылка на **PVC** (запрос на том)
- в спецификации **PVC** указываются необходимые параметры тома: размер и режим доступа

Persistent Volume Claim (PVC) — заявка на том

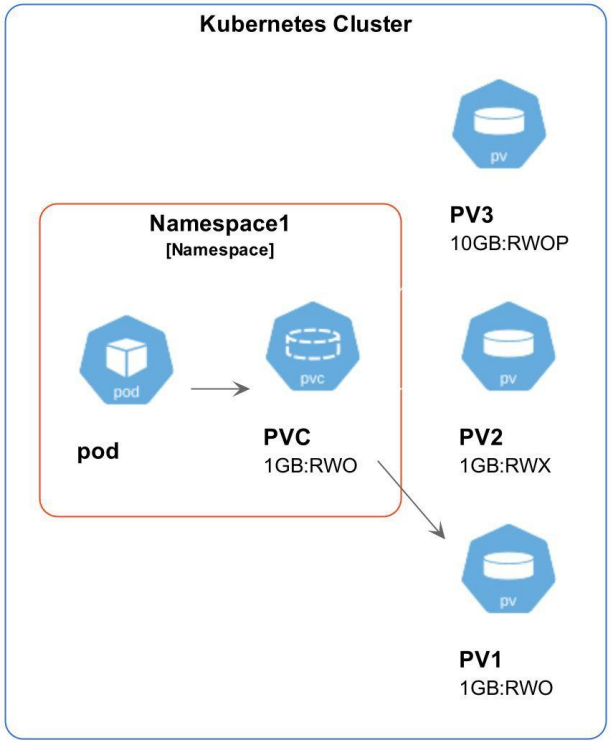
Инициализация PV



- в спецификации Pod’а указывается ссылка на **PVC** (запрос на том)
- в спецификации **PVC** указываются необходимые параметры тома: размер и режим доступа
- при наличии подходящего **PV** происходит связывание объектов **PVC** и **PV** (**bound**)

Persistent Volume Claim (PVC) — заявка на том

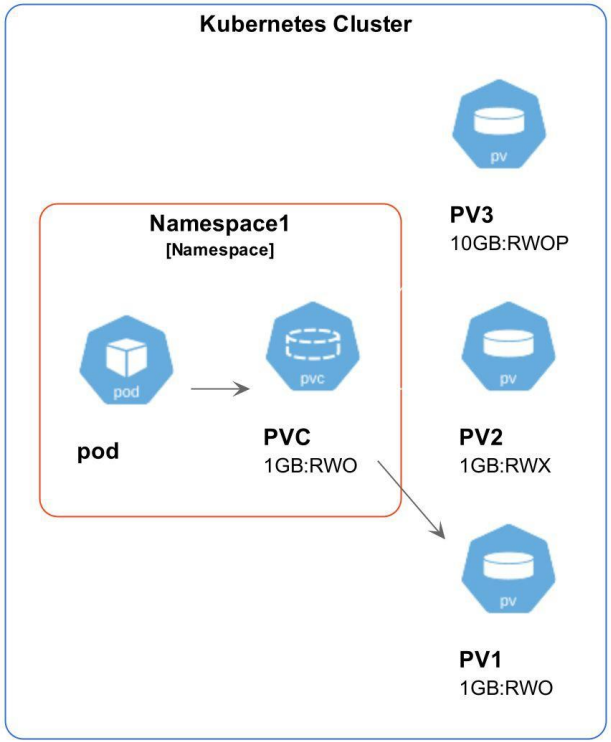
Инициализация PV



- в спецификации Pod’а указывается ссылка на **PVC** (запрос на том)
- в спецификации **PVC** указываются необходимые параметры тома: размер и режим доступа
- при наличии подходящего **PV** происходит связывание объектов **PVC** и **PV** (**bound**)
- Pod запускается и к указанной точке монтируется запрошенный том

Persistent Volume Claim (PVC) — заявка на том

Инициализация PV



- в спецификации Pod’а указывается ссылка на **PVC** (запрос на том)
- в спецификации **PVC** указываются необходимые параметры тома: размер и режим доступа
- при наличии подходящего **PV** происходит связывание объектов **PVC** и **PV** (**bound**)
- Pod запускается и к указанной точке монтируется запрошенный том
- При перезапуске Pod’а связка PVC - PV уже существует и данные сохранены. Новый Pod запускается и к указанной точке монтируется запрошенный том

Persistent Volume Claim (PVC) — заявка на том

PVC определяется в настройках Pod'а и должен быть в том же Namespace

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: app1
  namespace: web
spec:
  containers:
    - name: frontend
      image: nginx
      volumeMounts:
      - mountPath: "/var/www/html"
        name: vol
  volumes:
    - name: vol
      persistentVolumeClaim:
        claimName: pvc-vol
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-vol
  namespace: web
spec:
  storageClassName: manual
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

Persistent Volume Claim (PVC) — заявка на том

Связка (bound) PVC — PV происходит при совпадении характеристик

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-vol
  namespace: web
spec:
  storageClassName: manual
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv1
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /data/pv1
```

StorageClass

StorageClass — объект K8s, который описывает класс хранилища

Storage Class — класс хранилища

SC позволяет администратору определить тип хранилищ, доступных ресурсам кластера k8s.

- name - имя класса
- provisioner - драйвер хранилища
- parameters - параметры

Storage Class — класс хранилища

SC позволяет администратору определить тип хранилищ, доступных ресурсам кластера k8s.

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata: name:
  aws-ebs
provisioner: kubernetes.io/aws-ebs
parameters:
  type: io1
  iopsPerGB: "10"
  fsType: ext4
```

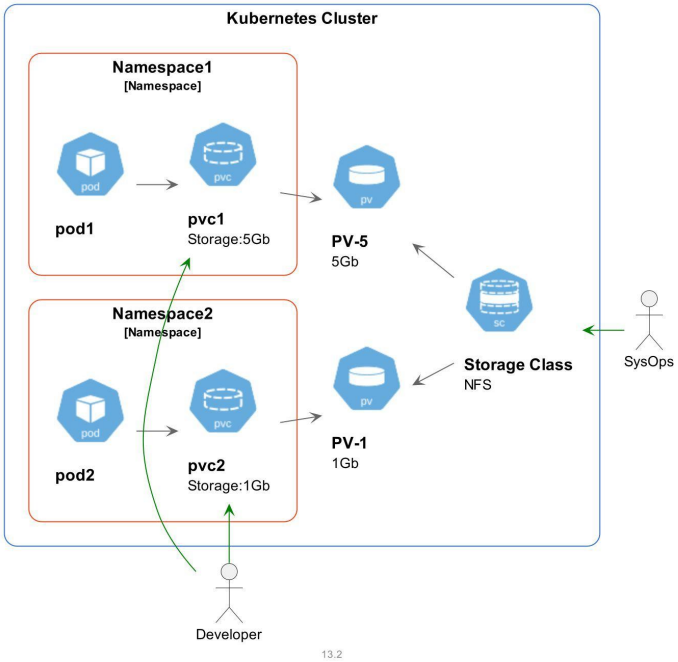
```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: local
provisioner: kubernetes.io/no-provisioner
volumeBindingMode: WaitForFirstConsumer
```

- name - имя класса
- provisioner - драйвер хранилища
- parameters - параметры

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-nfs
provisioner: example.com/external-nfs
parameters:
  server: nfs-server.example.com
  path: /share
  readOnly: "false"
```

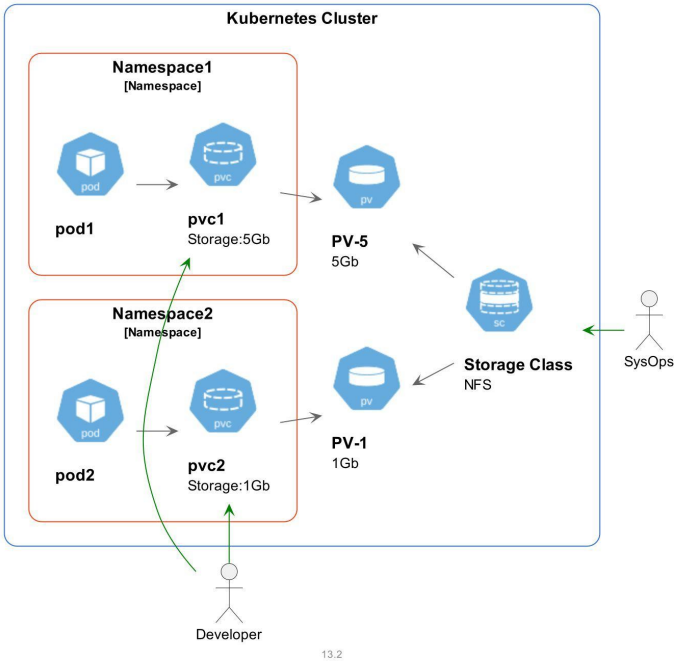
Storage Class — класс хранилища

SC позволяет администратору создать динамическое создание PV. При этом тома выделяются автоматически и эти тома соответствуют тем параметрам, которые требуются.



Storage Class — класс хранилища

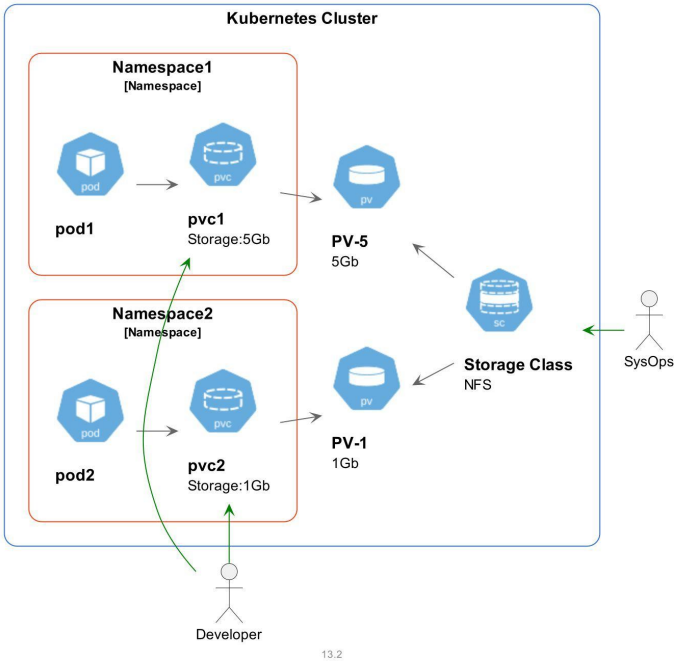
SC позволяет администратору создать динамическое создание PV. При этом тома выделяются автоматически и эти тома соответствуют тем параметрам, которые требуются.



Для динамического выделения томов необходимо подключить один или больше **provisioner**.

Storage Class — класс хранилища

SC позволяет администратору создать динамическое создание PV. При этом тома выделяются автоматически и эти тома соответствуют тем параметрам, которые требуются.



Для динамического выделения томов необходимо подключить один или больше **provisioner**.

Бывают:

- internal - kubernetes.io
- external - внешний

Storage Class — класс хранилища

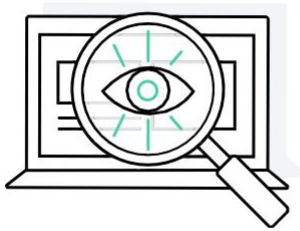
Provisioner SC создает PV автоматически при создании PVC ровно тех характеристик, которые описаны в PVC

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-vol
  namespace: web
spec:
  storageClassName: my-nfs
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-nfs
provisioner: example.com/external-nfs
parameters:
  server: nfs-server.example.com
  path: /share
  readOnly: "false"
```

Демонстрация работы

PV, PVC, SC



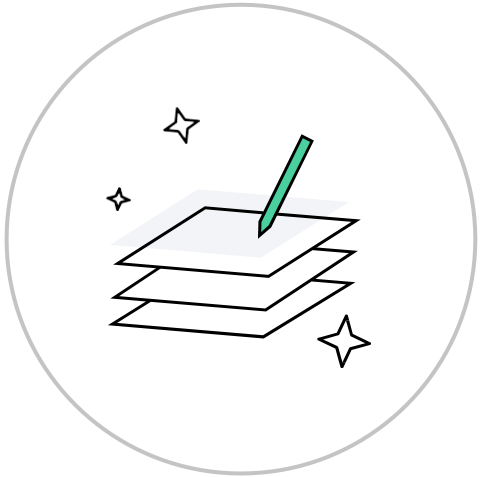
Итоги

1. Узнали, что такое PersistentVolume (PV) и PersistentVolumeClaim (PVC) и чем они отличаются
2. Разобрались с классами хранилищ
3. Поняли, как можно автоматизировать создание PV
4. Рассмотрели примеры манифестов объектов K8s
5. Попробовали подключиться к кластеру и посмотреть в работе объекты, изученные на занятии

Домашнее задание

Давайте посмотрим ваше домашнее задание

1. Вопросы о домашней работе задавайте в чате группы
2. Задачи можно сдавать по частям
3. Зачёт по домашней работе ставят после того, как приняты все задачи



Задавайте вопросы и пишите отзыв о лекции