**VOLUMES**

Том привязан к жизненному циклу модуля и будет существовать только до тех пор, пока модуль существует, но в зависимости от типа тома файлы тома могут оставаться нетронутыми даже после того, как модуль и том исчезнут, а затем могут быть смонтированы в новый том

**Типы томов:**

* emptyDir – простой пустой каталог, используемый для хранения временных данных;
* hostPath – используется для монтирования каталогов из файловой системы рабочего узла в модуль;
* gitRepo – том, инициализируемый в ходе проверки содержимого репозитория Git;
* nfs – общий ресурс NFS, монтируемый в модуле
* gcePersistentDisk тома в облачных хранилищах (AWS GCE и тд)
* configMap, secret, downwardAPI – специальные типы томов, используемые для предоставления модулю определенных ресурсов Kubernetes и кластерной информации;
* persistentVolumeClaim – способ использовать заранее или динамически резервируемое постоянное хранилище.

1. **emptyDir:**

**gitRepo и emptyDir это, в сущности, выделенный каталог, создаваемый специально для модуля и используемый исключительно модулем, который содержит этот том. При удалении модуля том и его содержимое удаляются**

apiVersion: v1

kind: ReplicationController

metadata:

name: replicacontrol

spec:

replicas: 2

selector:

app: fort

template:

metadata:

labels:

app: fort

spec:

containers:

- image: x3m5555/fortune:ver1 #первый контейнер с генератором текста который копируется в /var/htdocs/index.html

name: html-generator

volumeMounts: #том под названием html монтируется в /var/htdocs в контейнере

- name: html

mountPath: /var/htdocs

- image: nginx:alpine #второй контейнер, просто веб сервер, который открывает страницу созданную генератором 1го

name: web-server

volumeMounts: #тот же том с именем html монтируется в /usr/share/nginx/html второго контейнера

- name: html

mountPath: /usr/share/nginx/html

readOnly: true

ports:

- containerPort: 80

protocol: TCP

volumes: #единственный том emptyDir под названием html монтируется в два контейнера

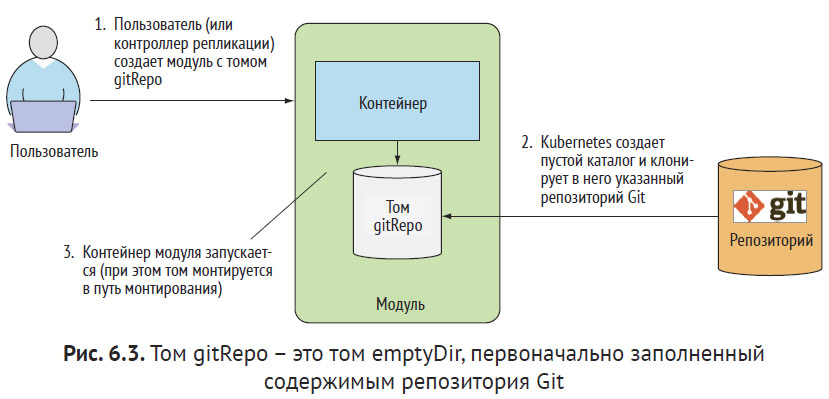
- name: html

emptyDir: {}

1. **gitRepo**

Том gitRepo – это, в сущности, том emptyDir, который заполняется путем клонирования репозитория Git и извлечения конкретной версии данных при запуске модуля (но до создания его контейнеров)

После создания тома он не синхронизируется с репозиторием на который ссылается, но всякий раз когда под создается он извлекает последнюю версию из Git



volumes:

- name: html

gitRepo: #создает том репо

repository: <https://github.com/andgit7/proxy-HPV.git> #клон репо Git

revision: main #будет извлечена ветка main

directory: . #репо клонируется в корневой каталог, если не указать клонируется в подкаталог proxy-HPV

[**https://hub.docker.com/r/jlowin/git-sync**](https://hub.docker.com/r/jlowin/git-sync) **контейнер с автоматической синхронизацией (пример)**

spec:

containers:

- image: nginx:alpine

name: web-server

volumeMounts:

- name: html

mountPath: /usr/share/nginx/html

readOnly: true

ports:

- containerPort: 80

protocol: TCP

- image: jlowin/git-sync #контейнер с синхронизацией

name: git-sync-volume

volumeMounts:

- name: html

mountPath: /git

env: #переменные

- name: GIT\_SYNC\_REPO #путь до репо

value: <https://github.com/andgit7/proxy-HPV.git>

- name: GIT\_SYNC\_DEST #путь до папки в контейнере

value: /git

- name: GIT\_SYNC\_WAIT #интервал проверки обновлений на GitHub

value: "10"

- name: GIT\_SYNC\_BRANCH #имя бранча

value: main

volumes:

- name: html

emptyDir: {}

1. **hostPath**

Первый тип постоянного хранилища том hostPath указывает на определенный файл или каталог в файловой системе узла. Поды, *работающие на одном узле* и использующие один и тот же путь в томе hostPath, видят одни и те же файлы. Поды с других узлов не видят том

1. **PersistentVolume**

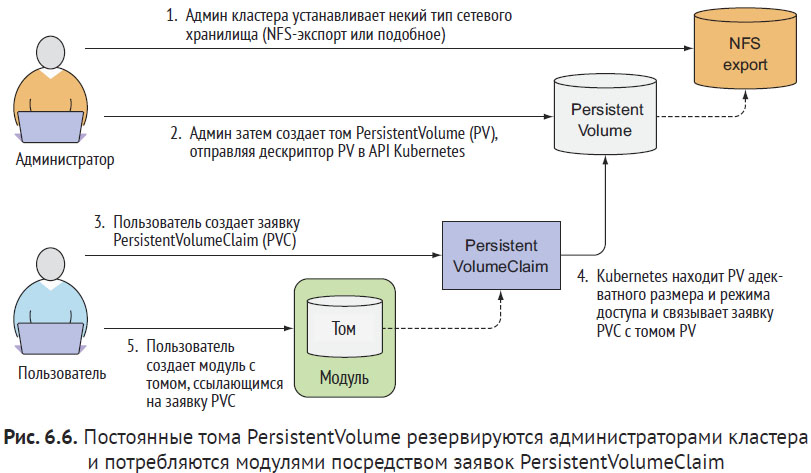
Постоянный том PersistentVolume и заявка на него PersistentVolumeClaim

PersistentVolume - через сервер API K8S создается том с определенным объемом и режимами доступа, хранится на одном из нодов

PersistentVolumeClaim - когда нужен постоянный том в поде надо создать манифест с заявкой в API K8S, где указывается минимальный объем и режим доступа.

Заявка затем используется как один из томов в поде

Том становится недоступным пока не буде высвобожден путем удаления заявки PersistentVolumeClaim

****

***Манифест создания PersistentVolume:***

apiVersion: v1

kind: PersistentVolume

metadata:

name: nginx-pv

spec:

capacity:

storage: 50M #объем

accessModes: #режимы доступа

- ReadWriteOnce

- ReadOnlyMany

persistentVolumeReclaimPolicy: Retain #после высвобождения зявки том не удаляется

hostPath:

path: /tmp/nginx #место хранения тома на ноде

***Манифест создания PersistentVolumeClaim:***

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: nginx-pvc #по этому имени сошлется под

spec:

resources:

requests:

storage: 20M #запрос хранилища объемом

accessModes: #режим доступа (тут чтение/запись для одного клиента)

- ReadWriteOnce

storageClassName: "" #Указание пустой строки в качестве имени класса хранилища обеспечивает привязку заявки PVC к предварительно #зарезервированному ресурсу PV вместо динамического резервирования нового ресурса

***Режимы доступа:***

RWO – ReadWriteOnce – только один узел может монтировать том для чте-

ния и записи;

RWX – ReadWriteMany – несколько узлов могут монтировать том как для

чтения, так и для записи.

ROX – ReadOnlyMany – несколько узлов могут монтировать том для чте-

ния;

kubectl get pv #просмотр томов PersistentVolume

kubectl get pvc #просмотр заявок PersistentVolumeClaim

***Манифест создания пода с PersistentVolume:***

apiVersion: v1

kind: Pod

metadata:

name: nginx-v

labels:

app: nginx-vol

spec:

containers:

- image: nginx

name: nginx

volumeMounts:

- name: nginx-data

mountPath: /usr/share/nginx/html/ #место монтирования тома

ports:

- containerPort: 80

protocol: TCP

volumes:

- name: nginx-data

persistentVolumeClaim: #ссылка на PersistentVolumeCliam

claimName: nginx-pvc #по имени в томе пода

***после удаления пода и заявки том не может быть привязан к новой заявке, Поскольку этот том вы уже использовали, он может содержать данные и не должен быть привязан к совершенно новой заявке без предоставления администратору кластера возможности его очистить.***

Политики рекламации томов:

*Retain* - не удаляется, но нельзя повторно использовать вновь созданной заявкой (только удалив pv, pvc, pod и заново все создав)

*Recycle* - удаляет содержимое тома, но делает его доступным для повторной заявки, можно использовать разными заявками и подами

*Delete* - удаляет хранилище

**Ресурс StorageClass** указывает, какой поставщик (provisioner) должен использоваться для создания ресурса PersistentVolume, когда заявка PersistentVolumeClaim запрашивает этот ресурс StorageClass. Параметры, определенные в определении StorageClass, передаются поставщику и специфичны для каждого плагина поставщика.