K8S 28: Kubernetes. Resources. Volume

Описание:

Для постоянного хранения данных в Kubernetes, как и в Docker, используются Volume. Но, в отличие от Docker, из-за того, что Kubernetes вместо перезапуска того же контейнера создает новый, данные внутри контейнера теряются.
В отличие от Docker, K8s поддерживает большое количество Volume, некоторые из которых наследуются из Docker, а некоторые специфичны для Kubernetes. Разберем некоторые из них:

- gcePersistentDisk, awsElasticBlockStore, azureDisk — специфичны для соответствующих облачных провайдеров — позволяют подключать блочные диски как Volumes;
- nfs, iscsi — позволяют подключить соответствующие сетевые хранилища по общим протоколам;
- glusterfs, cephfs — подключают созданные блоки хранения в соответствующих кластерных системах хранения;
- csi — позволяет подключать Container Storage Interfaces хранилища (об этом мы поговорим чуть подробнее в следующем задании);
- secret, configMap — приводят содержимое данных ресурсов в формат файлов, где имя ключа — это имя файла, а значение — содержимое файла (в случае секрета — в читаемом формате, а не в base64);
- hostPath — подключает директорию с хоста, где запущен под. Соответственно, до тех пор, пока под запускается на той же ноде, данные сохраняются;
- local — схож с hostPath, но при рестарте пода Kubernetes запускает контейнер на той же ноде, что и раньше, поскольку запоминает, где Volume был создан. Однако при недоступности ноды есть вероятность потерять данные, так как под переедет на другую ноду;
- emptyDir — создает пустую директорию, подключенную к поду на конкретной ноде. То есть данные в этой директории выживают до тех пор, пока не перезапускается под с переездом на другую ноду (при перезапусках контейнера данные сохраняются). У данного типа Volume есть особенность: дополнительно можно указать параметр medium: memory. В таком случае будет создан ram-диск;
- downwardAPI — эти тип Volume, который позволяет монтировать значения метаданных пода внутри контейнера в виде файлов.

hostPath часто используют совместно с каким-то настроенным хранилищем на хосте, к примеру, с подключенным к определенной директории GlusterFS.

Для подключения Volume требуется определить 2 директивы в описании пода (или шаблона пода) — volumes (в общем описании вне контейнера) и volumeMounts (в описании контейнера).
Пример использования Volume на примере emptyDir:

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir: {}
```

Отдельно хотелось бы выделить описание того, как использовать downwardAPI в 2 форматах — как переменные окружения и как Volume:

```
apiVersion: v1
kind: Pod
metadata:
  name: downward
spec:
  containers:
  - name: main
    image: busybox
    command: ["sleep", "9999999"]
    resources:
      requests:
        cpu: 15m
        memory: 100Ki
      limits:
        cpu: 100m
        memory: 4Mi
    env:
    - name: POD_NAME
      valueFrom:
```

```
        fieldRef:
          fieldPath: metadata.name
    - name: POD_NAMESPACE
      valueFrom:
        fieldRef:
          fieldPath: metadata.namespace
    - name: POD_IP
      valueFrom:
        fieldRef:
          fieldPath: status.podIP
    - name: NODE_NAME
      valueFrom:
        fieldRef:
          fieldPath: spec.nodeName
    - name: SERVICE_ACCOUNT
      valueFrom:
        fieldRef:
          fieldPath: spec.serviceAccountName
    - name: CONTAINER_CPU_REQUEST_MILLICORES
      valueFrom:
        resourceFieldRef:
          resource: requests.cpu
          divisor: 1m
    - name: CONTAINER_MEMORY_LIMIT_KIBIBYTES
      valueFrom:
        resourceFieldRef:
          resource: limits.memory
          divisor: 1Ki
---
apiVersion: v1
kind: Pod
metadata:
  name: downward
  labels:
    foo: bar
  annotations:
    key1: value1
    key2: |
      multi
      line
```

```
      value
spec:
  containers:
  - name: main
    image: busybox
    command: ["sleep", "9999999"]
    resources:
      requests:
        cpu: 15m
        memory: 100Ki
      limits:
        cpu: 100m
        memory: 4Mi
    volumeMounts:
    - name: downward
      mountPath: /etc/downward
  volumes:
  - name: downward
    downwardAPI:
      items:
      - path: "podName"
        fieldRef:
          fieldPath: metadata.name
      - path: "podNamespace"
        fieldRef:
          fieldPath: metadata.namespace
      - path: "labels"
        fieldRef:
          fieldPath: metadata.labels
      - path: "annotations"
        fieldRef:
          fieldPath: metadata.annotations
      - path: "containerCpuRequestMilliCores"
        resourceFieldRef:
          containerName: main
          resource: requests.cpu
          divisor: 1m
      - path: "containerMemoryLimitBytes"
        resourceFieldRef:
          containerName: main
```

```
      resource: limits.memory
      divisor: 1
```

В ходе монтирования Volume как директории, содержимое этой директории перезатирается, поэтому, если вам требуется сохранить содержимое директории, но примонтировать конкретный файл из того же ConfigMap, то можно использовать в volumeMount директиву subPath.
Пример использования:

```
apiVersion: v1
kind: Pod
metadata:
  name: example
spec:
    containers:
    - name: test
      image: nginx:stable
      volumeMounts:
      - mountPath: /etc/nginx/sites-enabled/config.conf
        name: site-config
        subPath: config.conf
    volumes:
    - name: site-config
      configMap:
        name: test-config
```

Есть еще один тип Volume, но он настолько большой, что мы поговорим о нем в следующем задании.

Полезные ссылки:

- Volumes (official docs)

Задание:

1. Напишите следующие манифесты:

- configMap с ключами config: config-data-here; another: should-not-be-here и именем rbmk8s27cm;
- secret с парой secret: such-secret-data и именем rbmk8s27secret;
- Deployment с именем rbmk8s27pod, одним контейнером с образом nginx:stable и подключением как volume:

    - emptyDir к директории /mnt/ed,
    - local к директории /mnt/local,
    - secret к директории /mnt/secret/,
    - configMap ключа config к директории /mnt/cm/.
2. Примените данные манифесты в вашем Minikube-кластере и выведите их в кластере через describe (команды и вывод сохраните).
3. Создайте файл thisismyfile в директории для emptyDir и local изнутри контейнера (команды и вывод сохраните).
4. Выведите содержимое всех директорий и файлов в директориях VolumeMount изнутри контейнера (команды и вывод сохраните).
5. Перезапустите под и вновь выведите содержимое всех директорий и файлов в директориях VolumeMount изнутри контейнера - что изменилось? Команды и вывод сохраните.
6. На проверку отправьте манифесты, ответ на вопрос, что изменилось после рестарта, и сохраненные выводы.