[1. 创建Pod使用Volume](https://note.youdao.com/ynoteshare1/iframe.html#3080-1576327266106)

[2. PV and PVC](https://note.youdao.com/ynoteshare1/iframe.html#2022-1576327290307)

[2.1 PV](https://note.youdao.com/ynoteshare1/iframe.html#7148-1576327279799)

[2.2 PVC](https://note.youdao.com/ynoteshare1/iframe.html#2682-1576327279801)

[3. StorageClass](https://note.youdao.com/ynoteshare1/iframe.html#2319-1584120050566)

[3.1 NFS环境准备](https://note.youdao.com/ynoteshare1/iframe.html#7289-1584120139842)

[3.2 StorageClass插件部署](https://note.youdao.com/ynoteshare1/iframe.html#7882-1584120050566)

[3.3 测试](https://note.youdao.com/ynoteshare1/iframe.html#2326-1584120050566)

[2.ConfigMap](https://note.youdao.com/ynoteshare1/iframe.html#3741-1584120105435)

[2.1 环境变量使用](https://note.youdao.com/ynoteshare1/iframe.html#3694-1584120105435)

[2.2 Volume挂载使用](https://note.youdao.com/ynoteshare1/iframe.html#1985-1584120105435)

[3.Secret](https://note.youdao.com/ynoteshare1/iframe.html#8153-1584120105435)

[3.1 环境变量使用](https://note.youdao.com/ynoteshare1/iframe.html#0022-1584120105435)

[3.2 Volume挂载](https://note.youdao.com/ynoteshare1/iframe.html#7377-1584120105435)

[3.3 secret docker-registory](https://note.youdao.com/ynoteshare1/iframe.html#1055-1584120105435)

[3.3.1 部署Harbor](https://note.youdao.com/ynoteshare1/iframe.html#2799-1584120105435)

[3.3.2 创建secret](https://note.youdao.com/ynoteshare1/iframe.html#9283-1584120105435)

[3.3.3 创建Pod，调用imagePullSecrets](https://note.youdao.com/ynoteshare1/iframe.html#8921-1584120105435)

[4. emptyDir](https://note.youdao.com/ynoteshare1/iframe.html#5290-1576327279804)

[4.1 emptyDir](https://note.youdao.com/ynoteshare1/iframe.html#1828-1576327361083)

[4.2 emptyDir + init-containers](https://note.youdao.com/ynoteshare1/iframe.html#5135-1576327279805)

**1. 创建Pod使用Volume**

<https://kubernetes.io/docs/concepts/storage/volumes/>

cat << EOF > test-volume.yaml

apiVersion: v1

kind: Pod

metadata:

name: test-pd

spec:

containers:

- image: nginx

imagePullPolicy: IfNotPresent

name: test-container

volumeMounts:

- mountPath: /test-pd

name: test-volume

volumes:

- name: test-volume

hostPath:

path: /data

EOF

**2. PV and PVC**

**2.1 PV**

<https://kubernetes.io/docs/concepts/storage/persistent-volumes/>

cat << EOF > pv.yaml

kind: PersistentVolume

apiVersion: v1

metadata:

name: task-pv-volume

labels:

type: local

spec:

storageClassName: manual //该名称将用于将PersistentVolumeClaim请求绑定到此，一对一

capacity:

storage: 10Gi

accessModes:

- ReadWriteOnce

hostPath:

path: "/storage/pv1"

EOF

kubectl apply -f pv.yaml

[root@master01 ~]# kubectl get pv

NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE

task-pv-volume 10Gi RWO Retain Available manual 7s

述具体的PV功能。

访问模式:

(RWO) ReadWriteOnce – the volume can be mounted as read-write by a single node (单node的读写)

(ROM) ReadOnlyMany – the volume can be mounted read-only by many nodes (多node的只读)

(RWM) ReadWriteMany – the volume can be mounted as read-write by many nodes (多node的读写)

pv可以设置三种回收策略：保留（Retain），回收（Recycle）和删除（Delete）。

- 保留（Retain）：允许人工处理保留的数据。（默认）

- 回收（Recycle）：将执行清除操作，之后可以被新的pvc使用。

- 删除（Delete）：将删除pv和外部关联的存储资源，需要插件支持。

PV卷阶段状态：

Available – 资源尚未被claim使用

Bound – 卷已经被绑定到claim了

Released – claim被删除，卷处于释放状态，但未被集群回收。

Failed – 卷自动回收失败

**2.2 PVC**

cat << EOF > pvc.yaml

kind: PersistentVolumeClaim

apiVersion: v1

metadata:

name: task-pv-claim

spec:

storageClassName: manual

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 3Gi

EOF

kubectl apply -f pvc.yaml

[root@master01 ~]# kubectl get pv

NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE

task-pv-volume 10Gi RWO Retain Bound default/task-pv-claim manual 2m17s

[root@master01 ~]# kubectl get pvc

NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE

task-pv-claim Bound task-pv-volume 10Gi RWO manual 9s

########## 创建一个pod使用pvc

cat << EOF > pod-pv.yaml

kind: Pod

apiVersion: v1

metadata:

name: task-pv-pod

spec:

volumes:

- name: task-pv-storage

persistentVolumeClaim:

claimName: task-pv-claim

containers:

- name: task-pv-container

image: nginx

imagePullPolicy: IfNotPresent

ports:

- containerPort: 80

name: "http-server"

volumeMounts:

- mountPath: "/usr/share/nginx/html"

name: task-pv-storage

EOF

kubectl apply -f pod-pv.yaml

### 在master节点操作

[root@master01 pv1]# kubectl exec -it task-pv-pod bash

root@task-pv-pod:/# cd /usr/share/nginx/html/

root@task-pv-pod:/usr/share/nginx/html# ls

root@task-pv-pod:/usr/share/nginx/html# touch index.html

root@task-pv-pod:/usr/share/nginx/html# echo 11 > index.html

root@task-pv-pod:/usr/share/nginx/html# exit

exit

[root@master01 pv1]# curl 192.168.1.41

11

### pod运行在node01，所以要去node01节点查看hostpath

[root@node01 ~]# cd /storage/

[root@node01 storage]# ls

pv1

[root@node01 storage]# cd pv1/

[root@node01 pv1]# ls

index.html

[root@node01 pv1]#

**3. StorageClass**

**3.1 NFS环境准备**

# 安装nfs server

yum -y install nfs-utils

# 启动服务，并设置为开机自启

systemctl enable --now nfs

# 创建共享目录

mkdir /storage

# 编辑nfs配置文件

vim /etc/exports

/storage \*(rw,sync,no\_root\_squash)

# 重启服务

systemctl restart nfs

# kubernetes集群计算节点部署

yum -y install nfs-utils

# 在计算节点测试

mkdir /test

mount.nfs 172.17.224.182:/storage /test

touch /test/123

**3.2 StorageClass插件部署**

# 下载系统插件:

yum -y install git

<https://github.com/kubernetes-incubator/external-storage>

git clone <https://github.com/kubernetes-incubator/external-storage.git>

# 修改yaml信息

cd /root/external-storage/nfs-client/deploy

vim deployment.yaml

env:

- name: PROVISIONER\_NAME

value: fuseim.pri/ifs

- name: NFS\_SERVER

value: 172.31.53.155

- name: NFS\_PATH

value: /storage

volumes:

- name: nfs-client-root

nfs:

server: 172.31.53.155

path: /storage

# 部署插件

kubectl apply -f rbac.yaml

kubectl apply -f deployment.yaml

kubectl apply -f class.yaml

**3.3 测试**

##### 测试一：创建pvc后自动创建pv并bound

cat << EOF > pvc-nfs.yaml

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: nginx-test

spec:

accessModes:

- ReadWriteMany

storageClassName: managed-nfs-storage

resources:

requests:

storage: 1Gi

EOF

##### 测试二：创建Pod，自动创建pvc与pv

cat << EOF > statefulset-pvc-nfs.yaml

apiVersion: apps/v1 kind: StatefulSet metadata: name: web spec: selector: matchLabels: app: nginx # has to match .spec.template.metadata.labels serviceName: "nginx" replicas: 3 # by default is 1 template: metadata: labels: app: nginx # has to match .spec.selector.matchLabels spec: terminationGracePeriodSeconds: 10 containers: - name: nginx image: nginx ports: - containerPort: 80 name: web volumeMounts: - name: www mountPath: /usr/share/nginx/html volumeClaimTemplates: - metadata: name: www spec: accessModes: [ "ReadWriteMany" ] storageClassName: "managed-nfs-storage" resources: requests: storage: 1Gi

EOF

##### 测试三：将nfs的storageclass设置为默认，创建Pod不指定storageclass，申请pvc的资源是否成功

# 设置managed-nfs-storage为默认

kubectl patch storageclass managed-nfs-storage -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'

# 测试，编写yaml文件不指定storageclass

cat <<EOF> statefulset2.yaml

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: web

spec:

selector:

matchLabels:

app: nginx # has to match .spec.template.metadata.labels

serviceName: "nginx"

replicas: 2 # by default is 1

template:

metadata:

labels:

app: nginx # has to match .spec.selector.matchLabels

spec:

terminationGracePeriodSeconds: 10

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

name: web

volumeMounts:

- name: html

mountPath: /usr/share/nginx/html

volumeClaimTemplates:

- metadata:

name: html

spec:

accessModes: [ "ReadWriteOnce" ]

resources:

requests:

storage: 1Gi

EOF

kubectl apply -f statefulset2.yaml

**2.ConfigMap**

**2.1 环境变量使用**

<https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/>

# 通过yaml文件创建env configmaps

cat << EOF > configmap.yaml

apiVersion: v1

kind: ConfigMap

metadata:

name: test-config

data:

username: damon

password: redhat

EOF

### pod使用configmaps的env环境变量

cat << EOF > config-pod-env1.yaml

apiVersion: v1

kind: Pod

metadata:

name: test-configmap-env-pod

spec:

containers:

- name: test-container

image: radial/busyboxplus

imagePullPolicy: IfNotPresent

command: [ "/bin/sh", "-c", "sleep 1000000" ]

envFrom:

- configMapRef:

name: test-config

EOF

### pod命令行使用comfigmaps的env环境变量

cat << EOF > config-pod-env2.yaml

apiVersion: v1

kind: Pod

metadata:

name: test-configmap-command-env-pod

spec:

containers:

- name: test-container

image: radial/busyboxplus

imagePullPolicy: IfNotPresent

command: [ "/bin/sh", "-c", "echo \$(MYSQLUSER) \$(MYSQLPASSWD); sleep 1000000" ]

env:

- name: MYSQLUSER

valueFrom:

configMapKeyRef:

name: test-config

key: username

- name: MYSQLPASSWD

valueFrom:

configMapKeyRef:

name: test-config

key: password

EOF

**2.2 Volume挂载使用**

# 创建配置文件的configmap

echo 123 > index.html

kubectl create configmap web-config --from-file=index.html

# pod使用volume挂载

cat << EOF > test-configmap-volume-pod.yaml

apiVersion: v1

kind: Pod

metadata:

name: test-configmap-volume-pod

spec:

volumes:

- name: config-volume

configMap:

name: web-config

containers:

- name: test-container

image: nginx

imagePullPolicy: IfNotPresent

volumeMounts:

- name: config-volume

mountPath: /usr/share/nginx/html

EOF

# subPath使用

cat << EOF > test-configmap-subpath.yaml

apiVersion: v1

kind: Pod

metadata:

name: test-configmap-volume-pod

spec:

volumes:

- name: config-volume

configMap:

name: web-config

containers:

- name: test-container

image: nginx

imagePullPolicy: IfNotPresent

volumeMounts:

- name: config-volume

mountPath: /usr/share/nginx/html/index.html

subPath: index.html

EOF

**3.Secret**

**3.1 环境变量使用**

##### 手动加密

echo -n 'admin' | base64

YWRtaW4=

echo -n 'redhat' | base64

cmVkaGF0

##### 解密

echo 'YWRtaW4=' | base64 --decode

#返回结果：admin

echo 'cmVkaGF0' | base64 --decode

#返回结果：redhat

###### 创建secret的yaml

cat << EOF > secret-env.yaml

apiVersion: v1

kind: Secret

metadata:

name: mysecret-env

type: Opaque

data:

username: YWRtaW4=

password: cmVkaGF0

EOF

##### pod env使用secret

cat << EOF > secret-pod-env1.yaml

apiVersion: v1 kind: Pod metadata: name: envfrom-secret spec: containers: - name: envars-test-container image: nginx envFrom: - secretRef: name: test-secret

EOF

cat << EOF > secret-pod-env2.yaml

apiVersion: v1

kind: Pod

metadata:

name: pod-env-secret

spec:

containers:

- name: mycontainer

image: radial/busyboxplus

imagePullPolicy: IfNotPresent

command: [ "/bin/sh", "-c", "echo \$(SECRET\_USERNAME) \$(SECRET\_PASSWORD); sleep 1000000" ]

env:

- name: SECRET\_USERNAME

valueFrom:

secretKeyRef:

name: mysecret-env

key: username

- name: SECRET\_PASSWORD

valueFrom:

secretKeyRef:

name: mysecret-env

key: password

EOF

**3.2 Volume挂载**

##### 创建配置文件的secret

kubectl create secret generic web-secret --from-file=index.html

##### volume挂在secret

cat << EOF > pod-volume-secret.yaml

apiVersion: v1

kind: Pod

metadata:

name: pod-volume-secret

spec:

containers:

- name: pod-volume-secret

image: nginx

imagePullPolicy: IfNotPresent

volumeMounts:

- name: test-web

mountPath: "/usr/share/nginx/html"

readOnly: true

volumes:

- name: test-web

secret:

secretName: web-secret

EOF

**3.3 secret docker-registory**

**3.3.1 部署Harbor**

# 下载离线安装包

wget https://github.com/goharbor/harbor/releases/download/v1.10.0/harbor-offline-installer-v1.10.0.tgz

# 下载docker-compose

wget https://docs.rancher.cn/download/compose/v1.25.4-docker-compose-Linux-x86\_64

chmod +x v1.25.4-docker-compose-Linux-x86\_64 && mv v1.25.4-docker-compose-Linux-x86\_64 /usr/local/bin/docker-compose

# 修改docker daemon.json ，添加安全私有镜像仓库

"insecure-registries": ["172.31.53.128"],

systemctl restart docker

# 安装harbor之前需要在harbor安装目录下修改harbor.yml 文件

./install.sh

# 登陆web创建用户，设定密码，创建项目

user: damon

password: [Damon@123](mailto:Damon@123)

# docker login 私有仓库

docker login <http://172.31.53.128/>

# 上传image到damon用户的私有仓库中

docker tag

docker push

**3.3.2 创建secret**

kubectl create secret docker-registry harbor-secret --docker-server=172.27.141.61 --docker-username=damon --docker-password=Damon@123 --docker-email=damon@qq.com

**3.3.3 创建Pod，调用imagePullSecrets**

cat << EOF > harbor-sc.yaml

apiVersion: v1

kind: Pod

metadata:

name: nginx

spec:

containers:

- name: u-demo

image: 172.31.53.96:18080/test/nginx:latest

imagePullSecrets:

- name: harbor-secret

EOF

kubectl create -f harbor-sc.yaml

**4. emptyDir**

**4.1 emptyDir**

<https://kubernetes.io/docs/tasks/configure-pod-container/configure-volume-storage/>

cat << EOF > emptydir.yaml

apiVersion: v1

kind: Pod

metadata:

name: emptydir-pod

labels:

app: myapp

spec:

volumes:

- name: storage

emptyDir: {}

containers:

- name: myapp1

image: radial/busyboxplus

imagePullPolicy: IfNotPresent

volumeMounts:

- name: storage

mountPath: /storage

command: ['sh', '-c', 'sleep 3600000']

- name: myapp2

image: radial/busyboxplus

imagePullPolicy: IfNotPresent

volumeMounts:

- name: storage

mountPath: /storage

command: ['sh', '-c', 'sleep 10000000']

EOF

kubectl apply -f emptydir.yaml

[root@master01 ~]# kubectl get pod

NAME READY STATUS RESTARTS AGE

emptydir-pod 2/2 Running 0 2m21s

[root@master01 ~]# kubectl exec -it emptydir-pod -c myapp1 sh

/ # cd /s

sh: cd: can't cd to /s

/ # cd /storage/

/storage # ls

/storage # touch 123

/storage # ls

123

/storage # exit

[root@master01 ~]# kubectl exec -it emptydir-pod -c myapp2 sh

/ # ls /storage/

123

**4.2 emptyDir + init-containers**

cat << EOF > initcontainers.yaml

apiVersion: v1

kind: Pod

metadata:

name: myapp-pod

labels:

app: myapp

spec:

volumes:

- name: storage

emptyDir: {}

containers:

- name: myapp-containers

image: radial/busyboxplus

imagePullPolicy: IfNotPresent

volumeMounts:

- name: storage

mountPath: /storage

command: ['sh', '-c', 'if [ -f /storage/testfile ] ; then sleep 3600000 ; fi']

initContainers:

- name: init-containers

image: radial/busyboxplus

imagePullPolicy: IfNotPresent

volumeMounts:

- name: storage

mountPath: /storage

command: ['sh', '-c', 'touch /storage/testfile && sleep 10']

EOF

kubectl apply -f initcontainers.yaml

[root@master01 ~]# kubectl get pod

NAME READY STATUS RESTARTS AGE

myapp-pod 0/1 Init:0/1 0 8s

[root@master01 ~]# kubectl get pod

NAME READY STATUS RESTARTS AGE

myapp-pod 1/1 Running 0 17s

[root@master01 ~]# kubectl exec -it myapp-pod -- ls /storage/

testfile