





























































**NSD CLOUD DAY09**

1. [案例1：编写service资源文件（一）](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY09/CASE/01/index.html#case1)
2. [案例2：编写service资源文件（二）](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY09/CASE/01/index.html#case2)
3. [案例3：对外发布服务](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY09/CASE/01/index.html#case3)
4. [案例4：configMap练习](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY09/CASE/01/index.html#case4)
5. [案例5：网络存储卷应用案例](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY09/CASE/01/index.html#case5)
6. [案例6：课外实验(微服务web集群实战)](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY09/CASE/01/index.html#case6)

**1 案例1：编写service资源文件（一）**

**1.1 问题**

本案例要求学习service资源文件，具体要求如下：

1. 创建service服务，访问后端apache
2. 测试负载均衡
3. Headless 服务

**1.2 方案**

注意事项：

今日课程所用到的案例需要依赖前面docker课程中制作的镜像，必须将前面课程中的镜像全部导入到私有仓库（192.168.1.100服务器），镜像列表如下：

busybox:latest

myos:v1804

myos:httpd

myos:php-fpm

myos:nginx

**1.3 步骤**

实现此案例需要按照如下步骤进行。

步骤一：创建service资源（在master主机操作）

1）创建2副本的Web服务容器

资源文件已经共享在云盘第四阶段kubernetes/v1.17.6/config/apache-example.yaml，各位同学需要提前下载至自己的master主机。

使用资源文件创建一个多部分的容器环境，默认多副本会自动分配到不同的主机上运行。

1. [root@master ~]# kubectl apply -f apache-example.yaml #创建资源
2. deployment.apps/apache-example created
3. [root@master ~]# kubectl get pod -o wide #查看创建的资源
4. NAME      READY STATUS     RESTARTS AGE IP NODE
5. apache-56d75cb55c-njqwj 1/1 Running     0     1m 10.254.68.2 node-0003
6. apache-56d75cb55c-wf6zt 1/1 Running     0     1m 10.254.39.2 node-0001

2）删除一个pod，观察变化

当发现某一个pod不能使用的时候RS会在其他机器上在创建一个相同的pod，及其对应的容器。

1. [root@master ~]# kubectl get pod -o wide #查看容器运行在哪台主机上
2. NAME      READY STATUS    RESTARTS AGE IP NODE
3. apache-56d75cb55c-ld5jp 1/1      Running    0 5m 10.254.68.3 kube-node3
4. apache-56d75cb55c-wf6zt 1/1     Running    0 7m 10.254.39.2 kube-node1
5. [root@master ~]# kubectl delete pod web-56d75cb55c-ld5jp
6. pod "web-56d75cb55c-ld5jp" deleted
7. #删除其中一个容器（容器名称不能照抄，根据上面查询到的结果删除）
8. [root@master ~]# kubectl get pod -o wide #删除后再次观察，容器被重建了
9. NAME      READY STATUS    RESTARTS AGE IP NODE
10. apache-56d75cb55c-7q9fv 1/1     Running    0 8s 10.254.55.2 kube-node2
11. apache-56d75cb55c-wf6zt 1/1     Running    0 7m 10.254.39.2 kube-node1

3）service

因为容器随时都也被重建，其IP地址也跟着随机变化，我们如何访问容器呢？

service就是解决这一个问题的方法，service会创建一个cluster ip，service总能找到对应的 pod，且 cluster ip保持不变，如果有pod对应多个容器，service 会自动在多个容器间实现负载均衡。

创建service的资源文件已经共享在云盘第四阶段kubernetes/v1.17.6/config/service-example.yaml，各位同学提前下载该资源文件。

1. [root@master ~]# cat service-example.yaml #查看资源文件
2. ---
3. apiVersion: v1
4. kind: Service            # 定义资源类型
5. metadata:
6. name: apache-service    # 服务的名字
7. spec:                # 服务详细信息
8. ports:                # 端口定义
9. - protocol: TCP        # 协议
10. port: 80            # 开放在前端的访问端口
11. targetPort: 80    # 目标主机端口
12. selector:            # 选择为那个 deployment 提供服务（后端）
13. app: myapache        # 通过标签指定后端
14. type: ClusterIP        # 服务类型
15. [root@master ~]# kubectl apply -f service-example.yaml #创建资源
16. [root@master ~]# kubectl get service
17. #查看service资源，查看资源信息可以获得一个随机的Cluster IP（集群IP）

4）验证效果

Service提供的集群IP，仅可以在集群内可以直接访问服务，但集群外无法访问服务。

下面测试访问使用的集群IP是随机的，不能照抄！！！

1. [root@localhost ~]# curl -m 2 http://10.254.146.37/     # 在跳板机上无法访问服务
2. curl: (28) Connection timed out after 2000 milliseconds
3. # 10.254.146.37是service提供的集群IP
4. [root@master ~]# kubectl run test -i -t --image=192.168.1.100:5000/myos:v1804
5. # 启动容器，等会在容器里面测试service提供的集群IP
6. [root@myos /]# curl -m 2 http://10.254.146.37/     # 在集群内使用curl可以访问服务
7. <pre>
8. hello world
9. hello world
10. hello world
11. [root@myos /]#

步骤二：创建handless资源（在master主机操作）

资源文件已经共享在云盘第四阶段kubernetes/v1.17.6/config/service-example.yaml，各位同学提前下载该资源文件，并需要修改该文件的内容。

1. [root@master ~]# vim service-example.yaml # 手工修改该文件
2. ---
3. apiVersion: v1
4. kind: Service
5. metadata:
6. name: apache-service
7. spec:
8. ports:
9. - protocol: TCP
10. port: 80
11. targetPort: 80
12. selector:
13. app: myapache
14. type: ClusterIP
15. clusterIP: None        # 在文件末尾手动添加这一行，设置为 None
16. [root@master ~]# kubectl apply -f service-example.yaml #创建资源
17. [root@myos ~]# dig -t all hl-service.default.svc.cluster.local
18. ... ...
19. ;; ANSWER SECTION:
20. hl-service.default.svc.cluster.local. 30     IN A    10.244.1.41
21. hl-service.default.svc.cluster.local. 30     IN A    10.244.6.34

**2 案例2：编写service资源文件（二）**

**2.1 问题**

本案例要求创建 nodePort 服务，具体要求如下：

1. 创建nodeport服务，并从集群外部测试访问效果

**2.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：创建nodeport服务（在master主机操作）

资源文件已经共享在云盘第四阶段kubernetes/v1.17.6/config/service-example.yaml，各位同学提前下载该资源文件，并需要修改该文件的内容。

1. [root@master ~]# vim service-example.yaml # 手动修改该文件
2. ---
3. apiVersion: v1
4. kind: Service
5. metadata:
6. name: apache-service2
7. spec:
8. ports:
9. - protocol: TCP
10. port: 80
11. targetPort: 80
12. selector:
13. app: myapache
14. type: nodePort         # 在文件末尾修改类型为 nodePort
15. [root@master ~]# kubectl apply -f service-example.yaml #创建资源

一旦将service映射为node节点的端口后，我们就可以在任意节点访问容器服务。

1. [root@master ~]# kubectl get service web-service -o wide
2. NAME     TYPE     CLUSTER-IP EXTERNAL-IP PORT(S) AGE    SELECTOR
3. web-service NodePort 10.254.232.91 <none> 80:32362/TCP 11s    run=apache
4. #查看service资源的IP和端口（注意看端口号）
5. [root@master ~]# curl http://node-0001:32362
6. # node-0001 已经在 /etc/hosts 中配置解析
7. # 在master、node-0001、node-0002等所有节点主机都可以访问容器服务

**3 案例3：对外发布服务**

**3.1 问题**

本案例主要练习ingress控制器，分别实现以下目标：

1. 安装配置ingress控制器
2. 通过ingress向外发布服务

**3.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：安装ingress控制器（在master主机操作操作）

ingress控制器对应的镜像文件已经共享在云盘第四阶段kubernetes/v1.17.6/ingress/ingress-nginx.tar.gz。各位同学需要提前下载。

1) 首先需要将ingress镜像上传至私有镜像仓库（192.168.1.100服务器）

1. [root@master ~]# docker load -i ingress-nginx.tar.gz
2. [root@master ~]# docker tag quay.io/kubernetes-ingress-controller/nginx-ingress-controller 192.168.1.100:5000/nginx-ingress-controller:0.30.0
3. [root@master ~]# docker push 192.168.1.100:5000/nginx-ingress-controller:0.30.0

2) 验证仓库

1. [root@master ~]#curl http://192.168.1.100:5000/v2/nginx-ingress-controller/tags/list
2. {"name":"nginx-ingress-controller","tags":["0.30.0"]}
3. [root@master ~]# kubectl apply -f httpd-example-v1.yaml
4. deployment.apps/web-example configured

3) 创建ingress资源

资源文件已经共享在云盘第四阶段kubernetes/v1.17.6/ingress/目录下，各位同学需要自行下载该资源文件，并需要修改该文件才可以使用。

修改资源文件mandatory.yaml，指定启动ingress容器时应该从私有仓库下载镜像，而默认指定的是官网。

1. [root@master ~]# vim ingress/mandatory.yaml #修改镜像文件image的路径
2. 221: image: 192.168.1.100:5000/nginx-ingress-controller:0.30.0
3. [root@master ~]# kubectl create -f ingress/mandatory.yaml #创建资源
4. namespace/ingress-nginx created
5. [root@master ~]# kubectl -n ingress-nginx get pod #查看资源
6. NAME         READY     STATUS RESTARTS AGE
7. nginx-ingress-controller-fc6766d7-xtsp2     1/1     Running 0     50m

步骤二：通过ingress对外发布容器服务（在master主机操作操作）

1）创建资源，通过ingress映射内部服务

1. [root@master ~]# vim ingress-example.yaml
2. ---
3. apiVersion: extensions/v1beta1
4. kind: Ingress                            # 资源对象类型
5. metadata:
6. name: my-app                        # ingress 资源名称
7. annotations:
8. kubernetes.io/ingress.class: "nginx"
9. spec:
10. backend:                            # 后端服务
11. serviceName: apache-service        # service 名称，需要查看之前实验创建的名称
12. servicePort: 80                    # 服务端口号，是服务的 Port
13. #注意：这里的apache-service是前面实验创建的service的名称
14. #前面service创建的集群IP只能在内部访问，现在通过inpress映射service
15. #用户访问ingress生成的IP就会自动映射到service的集群IP，集群IP再映射到容器IP
16. [root@master ~]# kubectl apply -f ingress-example.yaml

2）验证效果

1. [root@master ~]# kubectl get ingress #查看ingress资源，查看inpressIP地址
2. NAME HOSTS     ADDRESS         PORTS     AGE
3. my-app         \*            192.168.1.31 80            16s
4. ------------------------------------------------------------------------------
5. [root@localhost ~]# curl http://192.168.1.31 #现在，任意主机都可以访问服务
6. <pre>
7. Array
8. (
9. [REMOTE\_ADDR] => 10.244.6.1
10. [REQUEST\_METHOD] => GET
11. [HTTP\_USER\_AGENT] => curl/7.29.0
12. [REQUEST\_URI] => /
13. )
14. php\_host:     apache-example-9d8577cf-lw74h

**4 案例4：configMap练习**

**4.1 问题**

本案例主要做configMap练习，具体要求如下：

1. 通过configMap修改nginx的配置文件
2. 让nginx支持php

**4.2 方案**

ConfigMap是在Pod中映射(文件/目录)的一种方式，允许你将配置文件与镜像文件分离，以使容器化的应用程序具有可移植性。

通过ConfigMap我们可以把真机的目录或文件映射到容器中。

**4.3 步骤**

实现此案例需要按照如下步骤进行。

步骤一：创建configMap资源（在master主机操作）

1）准备一个nginx的配置文件。

启动一个容器，从容器中将nginx.conf文件拷贝到真机。

1. [root@master ~]# kubectl run --generator=run-pod/v1 nginx-pod \
2. --image=192.168.1.100:5000/myos:nginx
3. #启动一个nginx容器
4. [root@master ~]# kubectl cp \
5. nginx-pod:/usr/local/nginx/conf/nginx.conf nginx.conf
6. #将容器中的nginx.conf配置文件拷贝到真机
7. [root@master ~]# vim nginx.conf     # 参考运维课程修改配置，支持动静分离
8. #user nobody;
9. worker\_processes 2;
10. #error\_log logs/error.log;
11. events {
12. worker\_connections 65535;
13. }
14. http {
15. include mime.types;
16. default\_type application/octet-stream;
17. #log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '
18. # '$status $body\_bytes\_sent "$http\_referer" '
19. # '"$http\_user\_agent" "$http\_x\_forwarded\_for"';
20. sendfile on;
21. tcp\_nopush on;
22. keepalive\_timeout 65;
23. #gzip on;
24. server {
25. listen 80;
26. server\_name localhost;
27. location / {
28. root html;
29. index index.html index.htm;
30. }
31. error\_page 404 /404.html;
32. error\_page 500 502 503 504 /50x.html;
33. location = /50x.html {
34. root html;
35. }
36. # pass the PHP scripts to FastCGI server
37. location ~ \.php$ {
38. root html;
39. fastcgi\_pass 127.0.0.1:9000;
40. fastcgi\_index index.php;
41. include fastcgi.conf;
42. }
43. }
44. }

2）定义configMap

configMap可以映射单一文件，也可以映射一个目录。

语法格式：kubectl create configmap 名称 --from-file=文件路径

1. [root@master ~]# kubectl create configmap nginx-conf --from-file=nginx.conf
2. configmap "nginx-conf" created
3. # 创建一个名称为nginx-conf的configMap，对应的是真机的nginx.conf文件
4. [root@master ~]# kubectl get configmap nginx-conf #查看configMap
5. NAME     DATA     AGE
6. nginx-conf     1     10s

3）使用资源文件启动容器调用前面定义的configMap

资源文件已经工作在云盘第四阶段kubernetes/v1.17.6/config/目录下。

1. [root@master ~]# cat configmap-example.yaml             #查看下资源文件
2. #这个资源文件中会启动两个容器，一个是php-fpm，一个是nginx
3. #资源文件中仅给nginx容器映射了一个configmap
4. [root@master ~]# kubectl apply -f configmap-example.yaml #创建资源
5. [root@master ~]# kubectl get pod -o wide                    #查看资源IP
6. NAME         READY     STATUS RESTARTS     AGE     IP
7. myweb-xxx 2/2     Running 0     30s     10.244.1.44

4）测试网页

1. [root@master ~]# curl http://10.244.1.44/info.php #访问测试
2. <pre>
3. ... ...
4. php\_host:     myweb-b75456664-fvbll

**5 案例5：网络存储卷应用案例**

**5.1 问题**

本案例练习练习使用网络存储卷，具体要求如下。

1. 安装 NFS服务
2. 定义 PV，PVC
3. 在K8S集群中挂载NFS存储卷

**5.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：部署NFS服务器（在192.168.1.100主机操作）

1）部署服务器

1. [root@registry ~]# yum install -y nfs-utils
2. [root@registry ~]# mkdir -m 777 /var/webroot
3. [root@registry ~]# vim /etc/exports
4. /var/webroot    \*(rw)
5. [root@registry ~]# systemctl enable --now nfs

2）任意其他主机做客户端测试

1. [root@localhost ~]# yum install -y nfs-utils
2. [root@localhost ~]# showmount -e 192.168.1.100
3. Export list for 192.168.1.100:
4. /var/webroot \*

步骤二：创建PV/PVC资源（在master主机操作）

对应的资源文件在云盘第四阶段kubernetes/v1.17.6/config/目录。

1）创建PV资源

1. [root@master ~]# cat pv-example.yaml #查看学习PV资源文件
2. ---
3. apiVersion: v1
4. kind: PersistentVolume        # 声明持久卷
5. metadata:
6. name: pv-nfs
7. labels:
8. app: web-nfs
9. spec:
10. volumeMode: Filesystem        # 支持Filesystem和Block
11. capacity:
12. storage: 10Gi            # 卷提供的空间
13. accessModes:            # 读写模式 RWO,ROX,RWX
14. - ReadWriteMany            # RWX 模式
15. persistentVolumeReclaimPolicy: Retain    # 卷资源回收方式 Retain,Delete
16. nfs:
17. path: /var/webroot
18. server: 192.168.1.100
19. [root@master ~]# kubectl apply -f pv-example.yaml

2）创建PVC资源（默认情况下PVC与PV会自动匹配容量大小、自动映射）

1. [root@master ~]# cat pvc-example.yaml #查看学习PVC资源文件
2. ---
3. apiVersion: v1
4. kind: PersistentVolumeClaim        # PVC声明
5. metadata:
6. name: pvc-nfs
7. spec:
8. volumeMode: Filesystem        # 支持Filesystem和Block
9. accessModes:            # 读写模式 RWO,ROX,RWX
10. - ReadWriteMany            # RWX 模式
11. resources:
12. requests:
13. storage: 10Gi            # 希望能提供的空间
14. selector:
15. matchLabels:
16. app: web-nfs
17. [root@master ~]# kubectl apply -f pvc-example.yaml

3）创建容器资源，调用PVC

1. [root@master ~]# cat volume-example.yaml #查看学习资源文件
2. ... ...
3. spec:
4. containers:
5. - name: nginx-app
6. image: myos:nginx
7. ports:
8. - containerPort: 80
9. volumeMounts:
10. - mountPath: /var/webroot         # mount的路径
11. name: site-data
12. volumes:
13. - name: site-data
14. persistentVolumeClaim:
15. claimName: pvc-nfs         # mount的资源
16. #使用myos:nginx镜像启动一个名称为nginx-app的容器
17. #并给该容器把固定一个名称为pvc-nfs的PVC存储
18. #将该PVC存储映射到nginx容器的/var/webroot目录
19. [root@master ~]# kubectl apply -f volume-example.yaml

**6 案例6：课外实验(微服务web集群实战)**

**6.1 问题**

本案例结合前面的实验步骤做一个综合的微服务课外练习，具体要求如下。

1. 使用myos:php-fpm 创建后端应用php-app
2. 创建php-service，为后端应用提供内部clusterIP和负载均衡
3. 使用myos:nginx创建应用，并使用php-service解析php文件
4. 创建web-service，发布nginx应用到nodePort
5. 使用Ingress对外发布服务nginx应用

**6.2 步骤**

实现此案例需要参考前面的案例1至案例5的内容自行完成。