















































































**NSD CLOUD DAY08**

1. [案例1：kubectl基本命令](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY08/CASE/01/index.html#case1)
2. [案例2：资源对象文件](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY08/CASE/01/index.html#case2)
3. [案例3：集群扩容更新与回滚](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY08/CASE/01/index.html#case3)
4. [案例4：创建DaemonSet资源文件](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY08/CASE/01/index.html#case4)
5. [案例5：job和cronjob控制器的创建资源文件](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/CLOUD/DAY08/CASE/01/index.html#case5)

**1 案例1：kubectl基本命令**

**1.1 问题**

本案例要求学习使用kubectl基本命令来管理kubernetes，具体要求如下：

1. 查询 节点 信息（node）
2. 启动容器（run）
3. 查询资源信息（deployment）
4. 查询容器信息（pod）
5. 进入容器（exec，attach）
6. 删除资源（delete）

**1.2 方案**

Kubectl是用于控制Kubernetes集群的命令行工具。

Kubectl的语法格式如下:

kubectl [command] [TYPE] [NAME] [flags]

command: 子命令，如 create，get，describe，delete

type: 资源类型，可以表示为单数，复数或缩写形式

name: 资源的名称，如果省略，则显示所有资源信息

flags: 指定可选标志，或附加的参数

**1.3 步骤**

实现此案例需要按照如下步骤进行。

步骤一：kubectl命令练习（在master主机操作）

1）run创建容器资源。

语法格式：kubectl run 资源名称 -i -t --image=私有仓库镜像名称:标签

1. [root@master ~]# kubectl run haha -i -t --image=192.168.1.100:5000/myos:v1804
2. kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kubectl run --generator=run-pod/v1 or kubectl create instead.
3. If you don't see a command prompt, try pressing enter.
4. [root@haha-8bbd48d7b-wcnkd /]# #注意：这里已经进入了容器
5. [root@haha-8bbd48d7b-wcnkd /]# ifconfig
6. eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1450
7. inet 10.244.7.2 netmask 255.255.255.0 broadcast 0.0.0.0

2)get查询资源

语法格式：

kubectl get node 查询节点状态

kubectl get deployment 查询资源名称

kubectl get pod 查询pod容器资源

kubectl get pod -o wide 查询主机信息

kubectl get pod -o yaml 显示资源文件Yaml格式

1. [root@master ~]# kubectl get nodes
2. NAME     STATUS     ROLES     AGE     VERSION
3. master         Ready     master     12h     v1.17.6
4. node-0001         Ready    <none>     16h     v1.17.6
5. node-0002     Ready     <none>     16h     v1.17.6
6. node-0003     Ready     <none>     16h     v1.17.6
7. [root@master ~]# kubectl get deployment.apps
8. NAME         READY     UP-TO-DATE AVAILABLE AGE
9. haha         1/1     1      1      10m
10. [root@master ~]# kubectl get pod
11. NAME    READY    STATUS    RESTARTS      AGE
12. haha-xxxx    1/1    Running     1 14m
13. [root@master ~]# kubectl get pod -o wide
14. NAME    READY    STATUS    RESTARTS     AGE    IP     NODE ... ...
15. haha-xxxx    1/1    Running     1 14m    10.244.7.4     node-0003 ... ...

3）exec进入一个正在运行的容器中

语法格式：

kubectl exec -it 容器id 执行的命令

1. [root@kube-master ~]# kubectl get pod
2. NAME READY     STATUS    RESTARTS     AGE
3. haha-8bbd48d7b-wcnkd     1/1     Running     0     31m
4. #查看一个容器资源的名称
5. [root@kube-master ~]# kubectl exec -it haha-8bbd48d7b-wcnkd /bin/bash
6. # 进入该容器中
7. [root@haha-8bbd48d7b-wcnkd /]#

4）查看资源的详细信息（主要用于排错）

语法格式：

kubectl describe 资源类型 资源名称

1. [root@kube-master ~]# kubectl describe deployment haha
2. # 查看deployment资源的详细信息
3. Name: haha
4. Namespace:     default
5. CreationTimestamp:    Wed, 04 Mar 2020 15:50:24 +0800
6. ... ...
7. [root@kube-master ~]# kubectl describe pod haha-8bbd48d7b-wcnkd
8. # 查看容器的详细信息
9. Name:     haha-8bbd48d7b-wcnkd
10. Namespace:      default
11. Node:     kube-node1/192.168.1.11
12. Start Time:     Wed, 04 Mar 2020 15:50:24 +0800
13. Labels:     pod-template-hash=466804836
14. ... ...

5）查看 console 终端的输出信息

1. [root@kube-master ~]# kubectl attach haha-8bbd48d7b-wcnkd -c haha -i -t
2. If you don't see a command prompt, try pressing enter.
3. # 先通过attach进入一个容器
4. [root@haha-8bbd48d7b-wcnkd /]# echo hello world #在容器中echo输出信息
5. hello world
6. [root@haha-8bbd48d7b-wcnkd /]# Ctrl-p + Ctrl-q #按快捷键退出
7. [root@kube-master ~]#
8. [root@kube-master ~]# kubectl logs haha-8bbd48d7b-wcnkd
9. [root@haha-8bbd48d7b-wcnkd /]# echo hello world
10. hello world

6）delete删除资源

语法格式：

kubectl delete 资源类型 资源名称

1. [root@kube-master ~]# kubectl delete pod haha-xxxxxxxx
2. pod "haha-8bbd48d7b-wcnkd" deleted
3. # 删除pod，注意：这里的pod名称不能照抄！！！！
4. [root@kube-master ~]# kubectl get pod -o wide # 容器被删除后会自动重建
5. NAME READY STATUS RESTARTS AGE IP      NODE
6. haha-xxxxxxxx 1/1 Running 0     3s     10.254.9.2 kube-node2
7. haha-xxxxxxxx 1/1 Terminating 2     1h     10.254.39.2 kube-node1
8. [root@kube-master ~]# kubectl get deployment #查看deployment资源
9. NAME DESIRED CURRENT UP-TO-DATE AVAILABLE AGE
10. haha 1 1 1 1 1h
11. [root@kube-master ~]# kubectl delete deployment haha
12. deployment.extensions "haha" deleted
13. # 删除deployment资源，容器彻底消失

**2 案例2：资源对象文件**

**2.1 问题**

本案例要求熟悉资源文件的语法，具体要求如下：

1. 写一个 deployment 的资源文件
2. 启动一个基本系统(myos.yaml)
3. 启动一个 apache 服务(myweb.yaml)
4. 熟悉 kubectl 查询资源信息（deployment）
5. 熟悉 kubectl 查询容器信息（pod）
6. 熟悉 kubectl 进入容器（exec，attach）
7. 熟悉 kubectl 删除资源（delete）

**2.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：查看现有资源的资源对象文件（在master主机操作）

语法格式：

kubectl get 资源对象 资源名称 -o 格式（json|yaml）

1. [root@kube-master ~]# kubectl get deployment apache -o yaml
2. apiVersion: extensions/v1beta1
3. kind: Deployment
4. metadata:
5. annotations:
6. ... ...
7. # 导出deployment资源的资源对象文件，depoyment资源的名称为apache，导出yaml格式

步骤二：编写资源对象文件（在master主机操作）

注意：所有资源文件都在云盘第四阶段kubernetes/v1.17.6/config/目录有案例！！！

1）Pod资源文件

1. [root@master ~]# vim pod-example.yaml
2. #学习云盘中Pod资源模板的内容
3. [root@master ~]# kubectl apply -f pod-example.yaml
4. #创建Pod资源

2）Deployment资源文件

1. [root@master ~]# vim deployment-example.yaml
2. #学习云盘中deployment资源模板的内容
3. [root@master ~]# kubectl apply -f deployment-example.yaml
4. #创建deployment资源

3)备注

为了建立控制器和 pod 间的关联，kubernetes 先给每个 pod 打上一个标签（Label），然后再给相应的位置定义标签选择器（Label Selector），引用这些标签，资源文件的效果如下：

1. ... ...
2. selector:      #声明标签选择器
3. app: nginx     #为服务的后端选择标签
4. ... ...
5. metadata:
6. labels:     #声明标签
7. app: nginx     #定义标签名字(上下标签必须一致)
8. ... ...

4）如何使用资源文件

使用资源文件管理对象，语法格式：kubectl (apply|create|delete) -f 资源文件

* create 创建资源对象
* apply 声明更新资源对象
* delete 删除资源对象

1. [root@kube-master ~]# kubectl apply -f deployment-example.yaml
2. deployment.extensions "deployment-example" created
3. [root@kube-master ~]# kubectl get deployment deployment-example
4. NAME         DESIRED    CURRENT UP-TO-DATE AVAILABLE    AGE
5. deployment-example 1    1     1     0         19s
6. [root@kube-master ~]# kubectl delete -f deployment-example.yaml
7. deployment.extensions " deployment-example " deleted

**3 案例3：集群扩容更新与回滚**

**3.1 问题**

本案例主要练习集群扩容更新与回滚，分别实现以下目标：

1. 创建一个 myweb.yml 使用 apache 启动
2. 练习集群扩容
3. 更新：
4. 从 Apache 滚动更新到 nginx服务
5. 回滚 nginx 到 apache 服务
6. 验证

**3.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：创建一个单节点的Web容器（在master主机操作操作）

资源文件在云盘的第四阶段kubernetes/v1.17.6/config/目录有案例。

各位同学需要提前下载素材文件并传到master主机。

1. [root@master ~]# kubectl apply -f httpd-example-v1.yaml
2. deployment.apps/web-example configured

步骤二：集群扩容（在master主机操作操作）

我们可以使用命令行调整容器副本数量，也可以通过修改资源文件、更新资源的方式调整容器副本数量。

！！！推荐使用资源文件的方式！！！

方法一(不推荐):

1. [root@master ~]# kubectl scale deployment.apps httpd-example --replicas=3

方法二：

1. [root@master ~]# kubectl edit deployments.apps httpd-example
2. # 编辑 replicas 值，修改为自己需要的副本数量

步骤三：修改控制器更新策略（在master主机操作操作）

资源文件在云盘的第四阶段kubernetes/v1.17.6/config/目录有案例。

各位同学需要提前下载素材文件并传到master主机。

1. [root@master ~]# cat http-example-v2.yaml #学习云盘中提供的资源文件
2. spec:
3. selector:
4. matchLabels:
5. app: myapp
6. replicas: 3
7. revisionHistoryLimit: 10        # 保留10个历史版本
8. strategy:                 # 更新策略
9. rollingUpdate:            # 滚动更新
10. maxSurge: 25%            # 最多超过 25% 的副本数量
11. maxUnavailable: 25%        # 最多有 25% 的副本不可用
12. type: RollingUpdate     # 更新方式 [Recreate,RollingUpdate]
13. [root@master ~]# kubectl apply -f http-example-v2.yaml #更新资源

步骤三：更新与回滚（在master主机操作操作）

查看历史版本，回滚

语法格式：

kubectl rollout history deployment httpd-example

kubectl rollout undo deployment httpd-example --to-revision=1

1. [root@master ~]# kubectl apply -f httpd-example-v2.yaml
2. deployment.apps/myweb configured
3. [root@master ~]# kubectl rollout history deployment httpd-example
4. deployment.apps/myweb
5. REVISION CHANGE-CAUSE
6. 1 <none>
7. 2 <none>
8. [root@master ~]# kubectl rollout undo deployment httpd-example --to-revision=1
9. deployment.apps/myweb rolled back

**4 案例4：创建DaemonSet资源文件**

**4.1 问题**

本案例要求创建DaemonSet资源文件，具体要求如下：

1. 设置污点策略
2. 通过标签选择器在指定节点启动容器

**4.2 步骤**

实现此案例需要按照如下步骤进行。

步骤一：创建daemonset资源（在master主机操作）

1）查看、学习daemonset资源文件（仅查看即可）。

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

1. [root@master ~]# cat daemonset-example.yaml
2. ---
3. apiVersion: apps/v1
4. kind: DaemonSet        # 声明 DaemonSet 资源类型
5. metadata:
6. name: daemonset-example    # DaemonSet资源名称
7. namespace: default        # POD所在的命名空间
8. spec:
9. selector:
10. matchLabels:
11. app: myweb
12. template:
13. ... ...             # 以下配置与 Deployment 完全相同

2）创建Daemonset资源

资源文件在云盘第四阶段kubernetes/v1.17.6/config/目录下

1. [root@master ~]# kubectl apply -f daemonset-example.yaml

步骤二：设置污点策略（在master主机操作）

1）查看污点标签

污点标签包括：

* NoSchedule 不会被调度
* PreferNoSchedule 尽量不调度
* NoExecute 驱逐节点

1. [root@master ~]# kubectl describe node master
2. ... ...
3. Taints: node-role.kubernetes.io/master:NoSchedule
4. ... ...

2）设置、删除污点标签

注意：可以自定义添加修改标签

注意：不要修改所有系统默认自带的标签

1. [root@master ~]# kubectl taint node node-0001 key=value:NoSchedule
2. # 给node-0001节点设置污点标签，设置NoSchedule标签
3. [root@master ~]# kubectl taint node node-0001 key-
4. # 删除node-0001的污点标签

步骤二：设置污点策略（在master主机操作）

nodeSelector是节点选择约束的最简单推荐形式。

我们可以给节点打上标签，根据标签来选择需要的节点

1. 查看标签的命令：
2. [root@master ~]# kubectl get node --show-labels
3. 设置标签的命令
4. [root@master ~]# kubectl label nodes <node-name> <label-key>=<label-value>
5. 删除标签的命令
6. [root@master ~]# kubectl label nodes <node-name> <label-key>-

根据标签选择

注意：如果添加了标签选择，但无法找到对应的标签，POD将一直处于 Pending 状态，直到标签可用。

nodeName 标签也可以指定node节点名称，不推荐使用

可以给任意案例添加选择器，如：deployment-example.yaml

1. spec:            # POD.SPEC 标签
2. nodeSelector:        # 标签选择
3. disktype: ssd        # node 标签
4. ----------------------------------------------------------------------------
5. spec:            # POD.SPEC 标签
6. nodeName: node-0001    # 直接指定node主机名称，简单粗暴

**5 案例5：job和cronjob控制器的创建资源文件**

**5.1 问题**

本案例练习书写job和cronjob控制器的资源文件。

1. 熟悉job控制器的资源文件
2. 熟悉cronjob控制器的资源文件

**5.2 步骤**

实现此案例需要按照如下步骤进行。

注意：资源文件在云盘第四阶段kubernetes/v1.17.6/config/目录下。

各位同学需要提前将该目录下的素材下载到master主机。

步骤一：创建job计划任务控制器（在master主机操作）

job任务是单任务

1. [root@master ~]# vim job-example.yaml
2. ---
3. apiVersion: batch/v1        # 当前格式版本
4. kind: Job             # 当前资源类型
5. metadata:
6. name: pi
7. spec:
8. template:
9. spec:
10. containers:
11. - name: pi
12. image: 192.168.1.100:5000/myos:v1804
13. command: ["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"] # 命令
14. restartPolicy: OnFailure    # 重启策略，只支持[OnFailure,Never]
15. [root@master ~]# kubectl apply -f job-example.yaml

步骤二：创建cronjob计划任务控制器（在master主机操作）

cronjob任务的本质是多次创建job（周期性计划任务）

1. [root@master ~]# vim cronjob-example.yaml
2. ---
3. apiVersion: batch/v1beta1
4. kind: CronJob
5. metadata:
6. name: cronjob-pi
7. spec:
8. schedule: "\*/1 \* \* \* \*"
9. jobTemplate:
10. spec:
11. template:
12. spec: containers:
13. - name: pi
14. image: 192.168.1.100:5000/myos:v1804
15. command: ["perl", "-Mbignum=bpi", "-wle", "print bpi(500)"]
16. restartPolicy: OnFailure
17. [root@master ~]# kubectl apply -f cronjob-example.yaml

提示：需要等待 1 分钟才能看到变化。