# k8s中操作Deployment示例

本示例包含以下内容：

* [创建Deployment](#create-deployment)
* [更新Deployment](#rollout-deployment)
* [回滚Deployment](#rollback-deployment)
* [Deployment扩容和缩容](#scaling-deployment)
* [Deployment的暂停和恢复](#pausing-resuming-deployment)
* [用Service暴露Deployment](#expose)

## 创建Deployment

创建nginx-deployment.yaml文件，声明deployment的各项参数，文件内容如下：

apiVersion: apps/v1

kind: Deployment

metadata:

#指定deployment的名称

name: nginx-deployment

labels:

app: nginx

spec:

#期望创建3个nginx实例（3个pod）

replicas: 3

selector:

#选择label：app=nginx的pod来创建实例

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.7.9

ports:

- containerPort: 80

该文件表示要创建3个pod，每个pod运行nginx1.7.9镜像。  
执行创建

$ kubectl create -f nginx-deployment.yaml

$ deployment.apps "nginx-deployment" created

创建成功，查看deployment和pod的运行状况：

$ kubectl get deploy

NAME DESIRED CURRENT UP-TO-DATE AVAILABLE AGE

nginx-deployment 3 3 3 3 57s

$ kubectl get po -o wide

NAME READY STATUS RESTARTS AGE IP NODE

nginx-deployment-75675f5897-jgl27 1/1 Running 0 12s 172.17.0.5 minikube

nginx-deployment-75675f5897-rrnsk 1/1 Running 0 12s 172.17.0.7 minikube

nginx-deployment-75675f5897-txlkv 1/1 Running 0 12s 172.17.0.6 minikube

可以看到deployment已经创建成功，并创建了3个pod，IP地址分别是172.17.0.5，172.17.0.6，172.17.0.7。  
分别对三个IP地址使用curl命令，均能返回nginx的欢迎页内容，表示创建成功。

$ curl http://172.17.0.5

<!DOCTYPE html>

<html>

<head>

<title>Welcome to nginx!</title>

<style>

body {

width: 35em;

margin: 0 auto;

font-family: Tahoma, Verdana, Arial, sans-serif;

}

</style>

</head>

<body>

<h1>Welcome to nginx!</h1>

<p>If you see this page, the nginx web server is successfully installed and

working. Further configuration is required.</p>

<p>For online documentation and support please refer to

<a href="http://nginx.org/">nginx.org</a>.<br/>

Commercial support is available at

<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>

</body>

</html>

## 更新Deployment

我们将nginx:1.7.9版本更新到nginx:1.9.1。  
手动设置升级的nginx版本：

$ kubectl set image deployment/nginx-deployment nginx=nginx:1.9.1 --record

deployment.apps "nginx-deployment" image updated

查看rollout，pod和rs状态：

$ kubectl rollout status deploy nginx-deployment

Waiting for rollout to finish: 2 out of 3 new replicas have been updated...

$ kubectl get po

NAME READY STATUS RESTARTS AGE

nginx-deployment-75675f5897-rrnsk 1/1 Terminating 0 1m

nginx-deployment-75675f5897-txlkv 0/1 Terminating 0 1m

nginx-deployment-c4747d96c-bf97v 1/1 Running 0 4s

nginx-deployment-c4747d96c-ctpl2 1/1 Running 0 5s

nginx-deployment-c4747d96c-rthg5 1/1 Running 0 6s

$ get rs

NAME DESIRED CURRENT READY AGE

nginx-deployment-75675f5897 0 0 0 1m

nginx-deployment-c4747d96c 3 3 3 9s

可以看到旧的pod正在被终止，新的pod被创建出来，这是滚动升级的过程。  
等升级完成再来看看rollout和pod的状态：

$ kubectl rollout status deploy nginx-deployment

deployment "nginx-deployment" successfully rolled out

$ kubectl get po

NAME READY STATUS RESTARTS AGE

nginx-deployment-c4747d96c-bf97v 1/1 Running 0 4m

nginx-deployment-c4747d96c-ctpl2 1/1 Running 0 4m

nginx-deployment-c4747d96c-rthg5 1/1 Running 0 4m

可以看到已经升级完成了，依旧是3个pod。  
我们可以通过describe命令查看deployment的升级过程：

$ kubectl describe deployments

Name: nginx-deployment

Namespace: default

CreationTimestamp: Mon, 31 Dec 2018 16:52:13 +0800

Labels: app=nginx

Annotations: deployment.kubernetes.io/revision=2

kubernetes.io/change-cause=kubectl set image deployment/nginx-deployment nginx=nginx:1.9.1 --record=true

Selector: app=nginx

Replicas: 3 desired | 3 updated | 3 total | 3 available | 0 unavailable

StrategyType: RollingUpdate

MinReadySeconds: 0

RollingUpdateStrategy: 25% max unavailable, 25% max surge

Pod Template:

Labels: app=nginx

Containers:

nginx:

Image: nginx:1.9.1

Port: 80/TCP

Host Port: 0/TCP

Environment: <none>

Mounts: <none>

Volumes: <none>

Conditions:

Type Status Reason

---- ------ ------

Available True MinimumReplicasAvailable

Progressing True NewReplicaSetAvailable

OldReplicaSets: <none>

NewReplicaSet: nginx-deployment-c4747d96c (3/3 replicas created)

Events:

Type Reason Age From Message

---- ------ ---- ---- -------

Normal ScalingReplicaSet 12m deployment-controller Scaled up replica set nginx-deployment-75675f5897 to 3

Normal ScalingReplicaSet 10m deployment-controller Scaled up replica set nginx-deployment-c4747d96c to 1

Normal ScalingReplicaSet 10m deployment-controller Scaled down replica set nginx-deployment-75675f5897 to 2

Normal ScalingReplicaSet 10m deployment-controller Scaled up replica set nginx-deployment-c4747d96c to 2

Normal ScalingReplicaSet 10m deployment-controller Scaled down replica set nginx-deployment-75675f5897 to 1

Normal ScalingReplicaSet 10m deployment-controller Scaled up replica set nginx-deployment-c4747d96c to 3

Normal ScalingReplicaSet 10m deployment-controller Scaled down replica set nginx-deployment-75675f5897 to 0

从Event.Message这一列可以看出，滚动升级的过程就是把新的deployment的ReplicaSet设置从0慢慢上升到期望，这里也就是3，同时把旧deployment的ReplicaSet设置从3慢慢降到0。  
根据RollingUpdateStrategy的设置，保证在滚动升级中可用pod的数量。

## 回滚Deployment

当更新失败，或者是更新版本不稳定，k8s也可以方便的回滚到指定的历史版本。  
比如，当我们更新的时候，把1.9.1误输成了1.91

$ kubectl set image deployment/nginx-deployment nginx=nginx:1.91 --record

deployment.apps "nginx-deployment" image updated

查看deploy，pod和rollout状态：

$ kubectl get deploy

NAME DESIRED CURRENT UP-TO-DATE AVAILABLE AGE

nginx-deployment 3 4 1 3 45m

$ kubectl get po

NAME READY STATUS RESTARTS AGE

nginx-deployment-595696685f-q8zrr 0/1 ImagePullBackOff 0 29s

nginx-deployment-c4747d96c-bf97v 1/1 Running 0 44m

nginx-deployment-c4747d96c-ctpl2 1/1 Running 0 44m

nginx-deployment-c4747d96c-rthg5 1/1 Running 0 44m

$ kubectl rollout status deploy nginx-deployment

Waiting for rollout to finish: 1 out of 3 new replicas have been updated...

命令kubectl rollout status deploy nginx-deployment会阻塞等待deployment更新完成，由于我们输入了错误的nginx版本，无法更新完成，这条命令会一直阻塞，所以我们先CTRL+C终止执行。  
UP-TO-DATE=1表示创建了一个用于更新的Pod。  
从kubectl get po的结果我们得知有一个pod创建不成功。  
我们再通过describe命令看一下升级不成功的pod发生了什么：

$kubectl describe po nginx-deployment-595696685f-q8zrr

Name: nginx-deployment-595696685f-q8zrr

Namespace: default

Node: minikube/192.168.0.62

Start Time: Mon, 31 Dec 2018 17:37:07 +0800

Labels: app=nginx

pod-template-hash=1512522419

Annotations: <none>

Status: Pending

IP: 172.17.0.7

Controlled By: ReplicaSet/nginx-deployment-595696685f

Containers:

nginx:

Container ID:

Image: nginx:1.91

Image ID:

Port: 80/TCP

Host Port: 0/TCP

State: Waiting

Reason: ImagePullBackOff

Ready: False

Restart Count: 0

Environment: <none>

Mounts:

/var/run/secrets/kubernetes.io/serviceaccount from default-token-q244m (ro)

Conditions:

Type Status

Initialized True

Ready False

PodScheduled True

Volumes:

default-token-q244m:

Type: Secret (a volume populated by a Secret)

SecretName: default-token-q244m

Optional: false

QoS Class: BestEffort

Node-Selectors: <none>

Tolerations: node.kubernetes.io/not-ready:NoExecute for 300s

node.kubernetes.io/unreachable:NoExecute for 300s

Events:

Type Reason Age From Message

---- ------ ---- ---- -------

Normal Scheduled 12m default-scheduler Successfully assigned nginx-deployment-595696685f-q8zrr to minikube

Normal SuccessfulMountVolume 12m kubelet, minikube MountVolume.SetUp succeeded for volume "default-token-q244m"

Normal Pulling 10m (x4 over 12m) kubelet, minikube pulling image "nginx:1.91"

Warning Failed 10m (x4 over 12m) kubelet, minikube Failed to pull image "nginx:1.91": rpc error: code = Unknown desc = Error response from daemon: manifest for nginx:1.91 not found

Warning Failed 10m (x4 over 12m) kubelet, minikube Error: ErrImagePull

Normal BackOff 9m (x6 over 11m) kubelet, minikube Back-off pulling image "nginx:1.91"

Warning Failed 1m (x40 over 11m) kubelet, minikube Error: ImagePullBackOff

可以看到，由于镜像nginx:1.91是不存在的，所以pod报错：Failed to pull image "nginx:1.91":xxxx，导致升级不成功。  
通过以下命令查看升级历史记录，就跟svn，git差不多。

$ kubectl rollout history deploy nginx-deployment

deployments "nginx-deployment"

REVISION CHANGE-CAUSE

1 <none>

2 kubectl set image deployment/nginx-deployment nginx=nginx:1.9.1 --record=true

3 kubectl set image deployment/nginx-deployment nginx=nginx:1.91 --record=true

由于我们之前升级加了--record参数，可以看到升级当时所使用命令。  
使用rollout history xxx --revision=n可以查看明细：

$ kubectl rollout history deploy nginx-deployment --revision=2

deployments "nginx-deployment" with revision #2

Pod Template:

Labels: app=nginx

pod-template-hash=703038527

Annotations: kubernetes.io/change-cause=kubectl set image deployment/nginx-deployment nginx=nginx:1.9.1 --record=true

Containers:

nginx:

Image: nginx:1.9.1

Port: 80/TCP

Host Port: 0/TCP

Environment: <none>

Mounts: <none>

Volumes: <none>

用undo操作回滚：

$ kubectl rollout undo deploy nginx-deployment

deployment.apps "nginx-deployment"

或者使用--to-revision来回滚到指定版本：

$ kubectl rollout undo deploy nginx-deployment --to-revision=2

deployment.apps "nginx-deployment"

看一下rollout,pod,rs的状态：

$ get deployment

NAME DESIRED CURRENT UP-TO-DATE AVAILABLE AGE

nginx-deployment 3 3 3 3 1h

$ kubectl get pod

NAME READY STATUS RESTARTS AGE

nginx-deployment-c4747d96c-bf97v 1/1 Running 0 1h

nginx-deployment-c4747d96c-ctpl2 1/1 Running 0 1h

nginx-deployment-c4747d96c-rthg5 1/1 Running 0 1h

$ kubectl get rs

NAME DESIRED CURRENT READY AGE

nginx-deployment-595696685f 0 0 0 1h

nginx-deployment-75675f5897 0 0 0 1h

nginx-deployment-c4747d96c 3 3 3 1h

$ kubectl rollout status deploy nginx-deployment

deployment "nginx-deployment" successfully rolled out

回滚成功。  
describe查看一下deployment：

$ kubectl describe deploy nginx-deployment

Name: nginx-deployment

Namespace: default

CreationTimestamp: Mon, 31 Dec 2018 16:52:13 +0800

Labels: app=nginx

Annotations: deployment.kubernetes.io/revision=4

kubernetes.io/change-cause=kubectl set image deployment/nginx-deployment nginx=nginx:1.9.1 --record=true

Selector: app=nginx

Replicas: 3 desired | 3 updated | 3 total | 3 available | 0 unavailable

StrategyType: RollingUpdate

MinReadySeconds: 0

RollingUpdateStrategy: 25% max unavailable, 25% max surge

Pod Template:

Labels: app=nginx

Containers:

nginx:

Image: nginx:1.9.1

Port: 80/TCP

Host Port: 0/TCP

Environment: <none>

Mounts: <none>

Volumes: <none>

Conditions:

Type Status Reason

---- ------ ------

Available True MinimumReplicasAvailable

Progressing True NewReplicaSetAvailable

OldReplicaSets: <none>

NewReplicaSet: nginx-deployment-c4747d96c (3/3 replicas created)

Events:

Type Reason Age From Message

---- ------ ---- ---- -------

Normal DeploymentRollback 4m deployment-controller Rolled back deployment "nginx-deployment" to revision 2

Normal ScalingReplicaSet 4m deployment-controller Scaled down replica set nginx-deployment-595696685f to 0

从Events.Message中得知，deployment-controller将版本回滚到revision 2了。

## Deployment扩容和缩容

Deployment扩容缩容的本质其实就是改变ReplicaSet的数量来控制Pod的数量，增加就是扩容，缩小就是缩容。

### 扩容

假设我们要将nginx从3个pod扩展到10个pod：

$ kubectl scale deploy nginx-deployment --replicas=10

deployment.extensions "nginx-deployment" scaled

查看po,rs,deployment状态：

$ kubectl get po

NAME READY STATUS RESTARTS AGE

nginx-deployment-c4747d96c-2zw9w 0/1 ContainerCreating 0 2s

nginx-deployment-c4747d96c-bf97v 1/1 Running 0 2h

nginx-deployment-c4747d96c-ctpl2 1/1 Running 0 2h

nginx-deployment-c4747d96c-fr6b6 0/1 ContainerCreating 0 2s

nginx-deployment-c4747d96c-grgr6 0/1 ContainerCreating 0 2s

nginx-deployment-c4747d96c-hh9tk 0/1 ContainerCreating 0 2s

nginx-deployment-c4747d96c-pdz4q 0/1 ContainerCreating 0 2s

nginx-deployment-c4747d96c-rthg5 1/1 Running 0 2h

nginx-deployment-c4747d96c-rvfs7 0/1 ContainerCreating 0 2s

nginx-deployment-c4747d96c-s5mrc 0/1 ContainerCreating 0 2s

$ kubectl get rs

NAME DESIRED CURRENT READY AGE

nginx-deployment-595696685f 0 0 0 1h

nginx-deployment-75675f5897 0 0 0 2h

nginx-deployment-c4747d96c 10 10 7 2h

$ kubectl get deploy

NAME DESIRED CURRENT UP-TO-DATE AVAILABLE AGE

nginx-deployment 10 10 10 10 2h

可以看到，pod已经从3个变成10个了，这里pod的状态还在启动中，启动完成之后10个pod都将会是Running状态。

### 缩容

假设我们要将nginx从10个pod减少为3个：

$ kubectl scale deploy nginx-deployment --replicas=3

deployment.extensions "nginx-deployment" scaled

$ kubectl get deploy

NAME DESIRED CURRENT UP-TO-DATE AVAILABLE AGE

nginx-deployment 3 3 3 3 2h

$ kubectl get rs

NAME DESIRED CURRENT READY AGE

nginx-deployment-595696685f 0 0 0 1h

nginx-deployment-75675f5897 0 0 0 2h

nginx-deployment-c4747d96c 3 3 3 2h

$ kubeget po

NAME READY STATUS RESTARTS AGE

nginx-deployment-c4747d96c-2zw9w 1/1 Terminating 0 3m

nginx-deployment-c4747d96c-bf97v 1/1 Running 0 2h

nginx-deployment-c4747d96c-ctpl2 1/1 Running 0 2h

nginx-deployment-c4747d96c-hh9tk 1/1 Terminating 0 3m

nginx-deployment-c4747d96c-pdz4q 0/1 Terminating 0 3m

nginx-deployment-c4747d96c-rthg5 1/1 Running 0 2h

可以看到pod的数量在减少，当缩容成功之后，会只剩下3个pod。

手动扩容缩绒还不能体现k8s的精髓之处，k8s还能够根据节点的负载情况，自动扩容缩容，这样才能实现自动化，智能化。自动扩容的内容不在本篇中的讨论范围内。大概可能或许假以时日会写一篇关于自动扩容的例子。

## Deployment的暂停和恢复

在上述的例子中，当我们输入更新指令之后，滚动更新就立即开始了，但有的时候，我们还需要修改其他一些配置来保证新版本的正确运行，这时就需要先暂停滚动更新，等其他配置工作完成之后，再开始滚动升级。  
我们删掉当前deployment，用初始文件重新部署一个:

$ kubectl delete deploy nginx-deployment

deployment.extensions "nginx-deployment" deleted

$ kubectl create -f nginx-deployment.yaml

deployment.apps "nginx-deployment" created

暂停滚动更新：

$ kubectl rollout pause deploy nginx-deployment

deployment.apps "nginx-deployment" paused

我们将当前的1.9.1版本“升级”成1.7.9：

$ kubectl set image deploy nginx-deployment nginx=nginx:1.7.9

deployment.apps "nginx-deployment" image updated

查看rollout history

$ kubectl rollout history deploy nginx-deployment

deployments "nginx-deployment"

REVISION CHANGE-CAUSE

1 <none>

我们发现由于我们使用rollout pause暂停了滚动更新，该deployment并没有更新。  
继续修改配置，完成更新操作：

$ kubectl set resources deploy nginx-deployment -c=nginx --limits=cpu=200m,memory=512Mi

deployment.apps "nginx-deployment" resource requirements updated

$ kubectl rollout resume deploy nginx-deployment

deployment.apps/nginx-deployment resumed

配置完之后查看pod状态：

$ kubectl get pod

NAME READY STATUS RESTARTS AGE

nginx-deployment-6d95f89565-7wgrb 0/1 Pending 0 2m

nginx-deployment-75675f5897-9pqz5 1/1 Running 0 18m

nginx-deployment-75675f5897-mdclm 1/1 Running 0 18m

nginx-deployment-75675f5897-vf4lw 1/1 Running 0 18m

发现第一个Pod一直处于Pending状态，这剧本好像不太对…  
有了前面describe查看deployment升级失败的例子，这里我们举一反三一下：

$ kubectl describe pod nginx-deployment-6d95f89565-7wgrb

Name: nginx-deployment-6d95f89565-7wgrb

Namespace: default

Node: <none>

Labels: app=nginx

pod-template-hash=2851945121

Annotations: <none>

Status: Pending

IP:

Controlled By: ReplicaSet/nginx-deployment-6d95f89565

Containers:

nginx:

Image: nginx:1.7.9

Port: 80/TCP

Host Port: 0/TCP

Limits:

cpu: 200m

memory: 512Mi

Requests:

cpu: 200m

memory: 512Mi

Environment: <none>

Mounts:

/var/run/secrets/kubernetes.io/serviceaccount from default-token-q244m (ro)

Conditions:

Type Status

PodScheduled False

Volumes:

default-token-q244m:

Type: Secret (a volume populated by a Secret)

SecretName: default-token-q244m

Optional: false

QoS Class: Guaranteed

Node-Selectors: <none>

Tolerations: node.kubernetes.io/not-ready:NoExecute for 300s

node.kubernetes.io/unreachable:NoExecute for 300s

Events:

Type Reason Age From Message

---- ------ ---- ---- -------

Warning FailedScheduling 4s (x12 over 2m) default-scheduler 0/1 nodes are available: 1 Insufficient cpu.

看到Events知道了，是我的服务器太渣了，分配不起200m的cpu资源，那就分配少一些，重新分配一下：

$ kubectl set resources deploy nginx-deployment -c=nginx --limits=cpu=20m,memory=256Mi

deployment.apps "nginx-deployment" resource requirements updated

等一阵子等pod启动完毕：

$ kubectl get po

NAME READY STATUS RESTARTS AGE

nginx-deployment-6b58c6db49-n9jmf 1/1 Running 0 20s

nginx-deployment-6b58c6db49-rbwxw 1/1 Running 0 22s

nginx-deployment-6b58c6db49-rklg2 1/1 Running 0 17s

现在能正常更新完成了。

## 用Service暴露Deployment

到目前为止，上述deployment所生产的Pod都只能被Node内部访问，而且而且每次pod扩容，缩容或者是升级之后，IP地址均可能改变。我们用service将deployment暴露出去，让k8s外部的应用也可以访问：

$ kubectl expose deploy nginx-deployment --type=NodePort --name=nginx --port=80

service "nginx" exposed

使用get svc查看服务：

$ kubectl get svc

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 21h

nginx NodePort 10.111.219.247 <none> 80:31302/TCP 13s

这时候外部应用就可以使用31302这个端口来访问内部的包含nginx的pod了。