Kubernetes - CKAD (Deployments)

MSc. Carlos Avendaño Arango

Resources

Github Repository

https://github.com/carloselpapa10/kubernetes.git

Roadmap

- Deployments
  - Use cases
  - Sample
  - Basic commands
  - Rolling back
  - Scaling
  - Pausing and resuming
- Exercises

Deployments

A deployment is a type of controller that provides declarative updates for **Pods** and **ReplicaSets**.

You describe a desired state in a Deployment, and the Deployment controller changes the actual state to the desired state at a controlled rate. You can define Deployments to create new ReplicaSets, or to remove existing Deployments and adopt all their resources with new Deployments.

https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

Deployments - Use Cases

- Create a Deployment to rollout a ReplicaSet.
- Declare the new state of the Pods.
- Rollback to an early Deployment revision.
- Scale up the Deployment to facilitate more load.
- Pause the Deployment.
- Use the status of the Deployment.
- Clean up older ReplicaSets.

Deployments - Sample

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
    template:
      metadata:
        labels:
          app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

```
$ kubectl run nginx-deployment --image=nginx:1.7.9 --replicas=3 --labels=app=nginx --port=80
--dry-run -o yaml
```

NOTE: You MUST specify an appropriate selector and Pod template labels in a Deployment (in this case, app: nginx).

Do not overlap labels and selectors with other controllers (including other Deployment and StateFul Sets). Kubernetes doesn't stop you from overlapping, and if multiple controllers have overlapping selectors those controllers might conflict and behave unexpectedly.

On the other side, *do not change the generated label called "pod-template-hash" in the created Pods in order to avoid overlapping.*

Deployments - Basic commands

Create a Deployment

```
$ kubectl apply -f lab-deployments/nginx-deployment.yaml --record=true
```

To see the Deployment rollout status

```
$ kubectl rollout status deployment/nginx-deployment
```

To get the replica set (rs)

```
$ kubectl get rs
```

NOTE: You may specify the --record flag to write the command executed in the resource annotation-kubernetes-io/change-cause. It is useful for future introspection. For example, too see the commands executed in each Deployment version.

Deployments - Basic commands

Updating a Deployment (image)

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

To see the Deployment rollout status

```
$ kubectl rollout status deployment/nginx-deployment
```

NOTE

- Deployment ensures that only a certain number of Pods are down while they are being updated. By default, it ensures that at least 25% of the desired number of Pods are up (25% max unavailable).

- Deployment also ensures that only a certain number of Pods are created above the desired number of Pods. By default, it ensures that at most 25% of the desired number of Pods are up (25% max surge).

Deployments - Rolling Back

"When you rollback to an early version, only Deployment Pod Template part is rolled back".

```
$ kubectl rollout undo deployment/nginx-deployment
```

Note: You can set .spec.revisionHistoryLimit field in a Deployment to specify how many old ReplicaSets for this Deployment you want to retain.

To see the rollout history (*revision* and *change-cause*)

```
$ kubectl rollout history deployment/nginx-deployment
```

Rollback to a specific version

```
$ kubectl rollout undo deployment/nginx-deployment --to-revision=2
```

To see the details of each history

```
$ kubectl rollout history deployment/nginx-deployment --revision=2
```

NOTE

- The Deployment controller stops the bad rollout automatically and stops scaling up the new ReplicaSet. This depends on the rollingUpdate parameter (maxUnavailable) that you have specified. Kubernetes by default sets the value to 25%.

Deployments - Scaling

```
$ kubectl scale deployment/nginx-deployment --replicas=10
```

Assuming **Horizontal Pod Autoscaling** is enabled in your cluster, you can setup an autoscaler for your Deployment and choose the *minimum* and *maximum* number of Pods you want to run based on the CPU utilization of your existing Pods.

```
$ kubectl autoscale deployment/nginx-deployment --min=10 --max=15 --cpu-percent=80
```

NOTE
RollingUpdate Deployments support running multiple versions of an application at the same time. When you or an autoscaler scales a RollingUpdate Deployment that is in the middle of a rollout (either in progress or paused), the Deployment balances the additional replicas in the existing active ReplicaSets (ReplicaSets with Pods) in order to mitigate risk.

Deployments - Pausing and Resuming

You can **pause** a deployment before triggering one or more updates and then **resume** it. This allows you to apply multiple fixes in between pausing and resuming without triggering unnecessary rollouts.

```
$ kubectl rollout pause deployment/nginx-deployment

$ kubectl set image deployment/nginx-deployment nginx=nginx:1.9.1
$ kubectl set resources deployment/nginx-deployment -c=nginx --limits=cpu=200m,memory=512Mi

$ kubectl rollout resume deployment/nginx-deployment
```

NOTE
You can not rollback a paused Deployment until you resume it.

Exercises

Exercise 1

Create a deployment with image nginx:1.7.8, called nginx, having 2 replicas, defining port 80 as the port that this container exposes (don't create a service for this deployment)

Solution

```
(Option 1)$ kubectl run nginx --image=nginx:1.7.8 --replicas=2 --port=80
(Option 2)
$ kubectl create deployment nginx --image=nginx:1.7.8 --dry-run -o yaml >
lab-deployments/exercise-1.yaml
vim lab-deployments/exercise-1.yaml (add port and set replicas to 2)
$ kubectl apply -f lab-deployments/exercise-1.yaml
```

```
replicas: 2
template:
  spec:
    containers:
      name: nginx
      image: nginx:1.7.8
      ports:
      - containerPort: 80
```

Exercise 2

View the YAML of this deployment

Solution

```
$ kubectl get deploy nginx -o yaml --export
```

Exercise 3

View the YAML of the replica set that was created by this deployment

Solution

```
$ kubectl get rs -o yaml --export
```

Exercise 4

Get the YAML for one of the pods

Solution

```
$ kubectl get po nginx-5c8784fd6d-cx2c4 -o yaml --export
```

Exercise 5

Check how the deployment rollout is going

Solution

```
$ kubectl rollout status deployment/nginx
```

Exercise 6

Update the nginx image to nginx:1.7.9

Solution

```
$ kubectl set image deployment/nginx nginx=nginx:1.7.9
```

Exercise 7

Check the rollout history and confirm that the replicas are OK

Solution

```
$ kubectl rollout history deployment nginx
$ kubectl get deploy nginx
$ kubectl kubectl get rs
```

Exercise 8

Undo the latest rollout and verify that new pods have the old image (nginx:1.7.8)

Solution

```
$ kubectl rollout undo deploy nginx
$ kubectl get pod
$ kubectl describe pod nginx-5c8784fd6d-jjg78
Search the value of the field ‘image’ into the Containers object.
```

Exercise 9

Do an on purpose update of the deployment with a wrong image nginx:1.91

Solution

```
$ kubectl set image deploy nginx nginx=nginx:1.91
```

Exercise 10

Verify that something's wrong with the rollout

Solution

```
$ kubectl rollout status deploy nginx
$ kubectl get po
You should see ‘ImagePullBackOff’ in one of the Pods.
```

Exercise 11

Return the deployment to the second revision (number 2) and verify the image is nginx:1.7.9

Solution

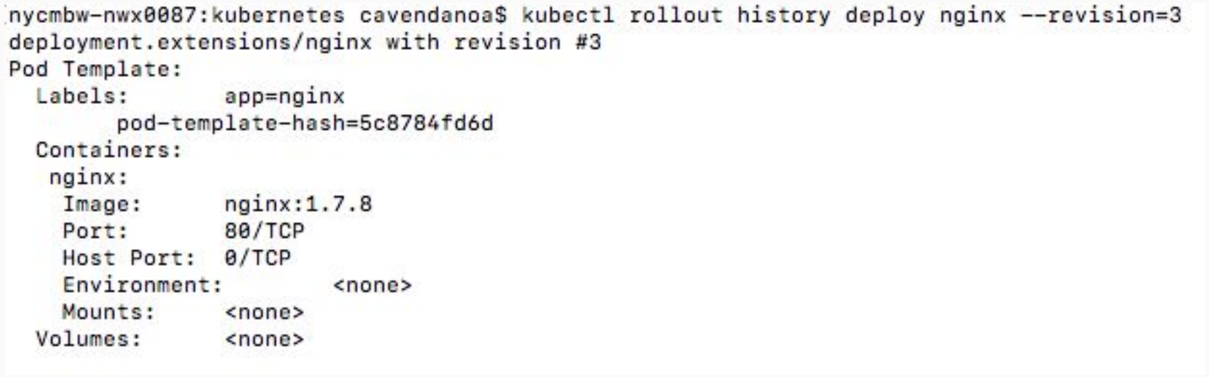
```
$ kubectl rollout history deploy nginx
$ kubectl rollout undo deploy nginx --to-revision=2
$ kubectl get po
$ kubectl describe po nginx-5754944d6c-77zdb | grep -i Image
You should see 'nginx:1.7.9'.
$ kubectl rollout status deploy nginx
You should see 'deployment "nginx" successfully rolled out'.
```

Exercise 12

Check the details of the third revision (number 3)

Solution

```
$ kubectl rollout history deploy nginx --revision=3
```



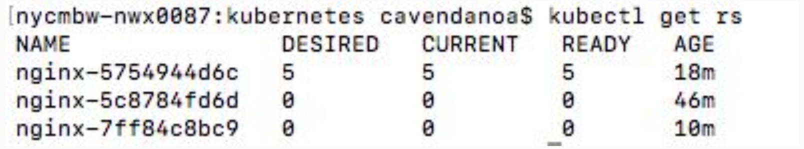
```
nycmbw-nwx0087:kubernetes cavendanoa$ kubectl rollout history deploy nginx --revision=3
deployment.extensions/nginx with revision #3
Pod Template:
  Labels:       app=nginx
        pod-template-hash=5c8784fd6d
  Containers:
   nginx:
    Image:      nginx:1.7.8
    Port:       80/TCP
    Host Port:  0/TCP
    Environment:        <none>
    Mounts:     <none>
  Volumes:      <none>
```

Exercise 13

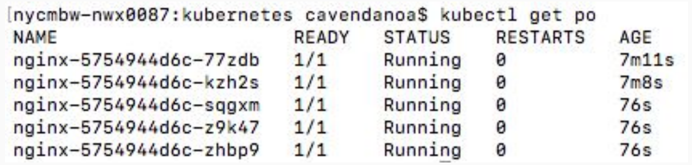
Scale the deployment to 5 replicas

Solution

```
$ kubectl scale deploy nginx --replicas=5
$ kubectl get rs
$ kubectl get po
```



```
[nycmbw-nwx0087:kubernetes cavendanoa$ kubectl get rs
NAME               DESIRED   CURRENT   READY   AGE
nginx-5754944d6c   5         5         5       18m
nginx-5c8784fd6d   0         0         0       46m
nginx-7ff84c8bc9   0         0         0       10m
```



```
[nycmbw-nwx0087:kubernetes cavendanoa$ kubectl get po
NAME                     READY   STATUS    RESTARTS   AGE
nginx-5754944d6c-77zdb   1/1     Running   0          7m11s
nginx-5754944d6c-kzh2s   1/1     Running   0          7m8s
nginx-5754944d6c-sqgxm   1/1     Running   0          76s
nginx-5754944d6c-z9k47   1/1     Running   0          76s
nginx-5754944d6c-zhbp9   1/1     Running   0          76s
```

Exercise 14

Autoscale the deployment, pods between 5 and 10, targeting CPU utilization at 80%

Solution

```
$ kubectl autoscale deploy nginx --min=5 --max=10 --cpu-percent=80
You see ‘HorizontalPodAutoscaler.autoscaling/nginx autoscaled’
```

Exercise 15

Pause the rollout of the deployment

Solution

```
$ kubectl rollout pause deploy nginx
You see 'deployment.extensions/nginx paused'
```

Exercise 16

Update the image to nginx:1.9.1 and check that there's nothing going on, since we paused the rollout

Solution

```
$ kubectl set image deploy nginx nginx=nginx:1.9.1
$ kubectl rollout history deploy nginx
No new revision
```

Exercise 17

Resume the rollout and check that the nginx:1.9.1 image has been applied

Solution

```
$ kubectl rollout resume deploy nginx
$ kubectl get po
$ kubectl describe po nginx-7448597cd5-9579t | grep -i Image:
You should see ‘Image: nginx:1.9.1’ in the output
```

Exercise 18

Resume the rollout and check that the nginx:1.9.1 image has been applied

Solution

```
$ kubectl rollout resume deploy nginx
$ kubectl get po
$ kubectl describe po nginx-7448597cd5-9579t | grep -i Image:
You should see ‘Image: nginx:1.9.1’ in the output
```

Exercise 19

Delete the deployment and the horizontal pod autoscaler you created

Solution

```
$ kubectl delete deploy nginx
$ kubectl delete hpa nginx
```