Understand Kubernetes Object and Create Nginx Deployment

Kubernetes Objects

*T*hese are persistent entities in the Kubernetes system. Kubernetes uses these entities to represent the state of your cluster.

A Kubernetes object is a “record of intent”—once you create the object, the Kubernetes system will constantly work to ensure that object exists.

By creating an object, you’re effectively telling the Kubernetes system what you want your cluster’s workload to look like; this is your cluster’s *desired state*.

Kubernetes object includes two nested object fields that govern the object’s configuration: the object `spec` and the object `status`

Deploying workloads using YAML files

We will create the resource configurations required to deploy your applications in Kubernetes

Make sure you have a Kubernetes cluster ready and kubectl configured to manage the cluster resources.

Agenda:

- Creating a Deployment
- Verifying a Deployment
- Editing a Deployment
- Rolling back a Deployment
- Deleting a Deployment

Kubernetes: Creating a Deployment

A Deployment is an object that actually represents your application which is running on the cluster

We will create a Deployment using a yaml file where we will have configurations defined for our application like images, port, pods etc.

So let's create a sample yaml file for the nginx deployment. You can also refer the git repo for this yaml file

```
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Now, lets describe each tags in more detail

`apiversion`—Version of Kubernetes API

`kind`—Kind of object like Deployment/Service you want to create

`metadata`—to uniquely identify your object with tag `name`

`spec`—Desired state of your object by providing below details:

1. `.spec.selector` how the Deployment finds which Pods to manage
2. `.spec.replicas` how many pods

For more details on deployment refer their official documentation

Now Let's create a deployment object using the yaml file

`– -record` This flag is used to write the command executed in the resource annotation `kubernetes.io/change-cause`.

```
kubectl apply -f nginx.yaml --record=true
```

```
C:\Users\khand\Desktop\extra\Study\devops4solutions\gitcode\kubernetes-sample-deployment>kubectl apply -f nginx.yaml
deployment.apps/nginx-deployment created

C:\Users\khand\Desktop\extra\Study\devops4solutions\gitcode\kubernetes-sample-deployment>
```

Now if you run the rollout history you will see the message

```
kubectl rollout history deployment.v1.apps/nginx-deployment
```

```
C:\Users\khand\Desktop\extra\Study\devops4solutions\gitcode\kubernetes-sample-deployment>kubectl rollout history deployment.v1.apps/nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         kubectl apply --filename=nginx.yaml --record=true
```

Verify the Deployment

Let's check the status of the deployment

```
kubectl get deployment
```

```
C:\Users\khand\Desktop\extra\Study\devops4solutions\gitcode\kubernetes-sample-deployment>kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2/2     2            2           2m19s

C:\Users\khand\Desktop\extra\Study\devops4solutions\gitcode\kubernetes-sample-deployment>
```

Check the deployment status

```
kubectl rollout status deployment nginx-deployment
```

```
C:\Users\khand\Desktop\extra\Study\devops4solutions\gitcode\kubernetes-sample-deployment>kubectl rollout status deployment nginx-deployment
deployment "nginx-deployment" successfully rolled out
```

Check the replica sets and pods which got deployed as part of this deployment

Replica Set—maintains a stable set of replica Pods running at any given time.

```
kubectl get rs,pods
```

```
C:\Users\khand\Desktop\extra\Study\devops4solutions\gitcode\kubernetes-sample-deployment>kubectl get rs,pods
NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-574b87c764   2         2         2       42m

NAME                                    READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-574b87c764-nrznh   1/1     Running   0          42m
pod/nginx-deployment-574b87c764-t6nvj   1/1     Running   0          51s
```

Edit the Deployment

Firstly, let's add the annotation which will provide the description of your change

`CHANGE-CAUSE` is copied from the Deployment annotation `kubernetes.io/change-cause` to its revisions upon creation. You can specify the`CHANGE-CAUSE` message by:

```
kubectl annotate deployment nginx-deployment kubernetes.io/change-cause="initial depl
oyment"

kubectl rollout history deployment.v1.apps/nginx-deploymen
```

```
C:\Users\khand\Desktop\extra\Study\devops4solutions\gitcode\kubernetes-sample-deployment>
C:\Users\khand\Desktop\extra\Study\devops4solutions\gitcode\kubernetes-sample-deployment>kubectl annotate deployment nginx-deployment k
1 deployment with image 1.14"
deployment.apps/nginx-deployment annotated

C:\Users\khand\Desktop\extra\Study\devops4solutions\gitcode\kubernetes-sample-deployment>kubectl rollout history deployment.v1.apps/ngi
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         initial deployment with image 1.14
```

Now we will try to edit the deployment and scale it to 3

```
kubectl edit deployment nginx-deployment
```

Now check the rolling Status, you will see only one revision because scaling the Deployment, do not create a Deployment revision. Again, edit the deployment and change the image to the some other version you will see the new revision gets created.

```
kubectl rollout history deployment.v1.apps/nginx-deployment
```

```
C:\Users\khand\Desktop\extra\Study\devops4solutions\gitcode\kubernetes-sample-deployment>kubectl rollout history deployment.v1
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         initial deployment with image 1.14
```

Describe Deployment

```
kubectl get deployment
```

Rolling Back Deployment

By default, all of the Deployment's rollout history is kept in the system so that you can rollback anytime you want.

A Deployment's revision is created when a Deployment's rollout is triggered.

If you update the labels or container images of the template then only a new Deployment revision gets created.

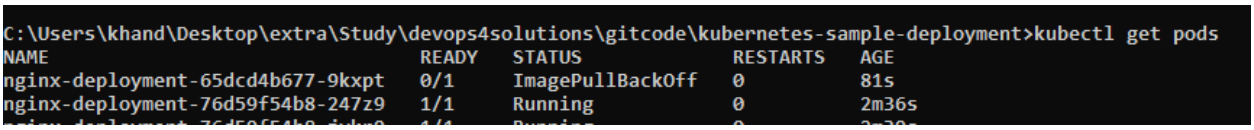
Other updates, such as scaling the Deployment, do not create a Deployment revision.

Therefore, when you do the rollback to the previous version then only pod template gets rolled back but not the pod replica. If you have 3 pods running then after rollback also 3 pods will be running.

Now, let's use the different image of nginx which is not available in dockerhub

using below command

```
kubectl set image deployment/nginx-deployment nginx=nginx:1.16.1.2 --record
```



To fix the above issue you need to revert back to the previous version

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

Deleting a deployment

```
kubectl delete deployment nginx-deployment
```

Congratulations, we have successfully explore Kubernetes Objects , learn about how to create a nginx deployment using yaml file.