Lab - Deployment Object in Kubernetes

08:58 PM

Kubernetes Deployment Use Cases

It's important to understand why you'd want to use a Kuberenetes Deployment in the first place. Some of these use cases include:

- **Ensuring availability of a workload:** Deployments specify how many copies of a particular workload should always be running, so if a workload dies, Kubernetes will automatically restart it, ensuring that the workload is always available.
- **Scaling workloads:** Kubernetes makes it easy to change how many replicas a Deployment should maintain, making it straightforward increase or decrease the number of copies running at any given time. It even offers autoscaling!
- **Managing the state of an application:** Deployments can be paused, edited, and rolled back, so you can make changes with a minimum of fuss.
- **Easily exposing a workload outside the cluster:** It might not sound like much, but being able to create a service that connects your application with the outside world with a single command is more than a little convenient.

Now let's look at actually building Deployments.

Failed deployment issue

- Inssucffient premission
- Image pull

Kubectl Rollout status deployment nameof deplyment

Kubectl Rollout history deployment nameof deplyment

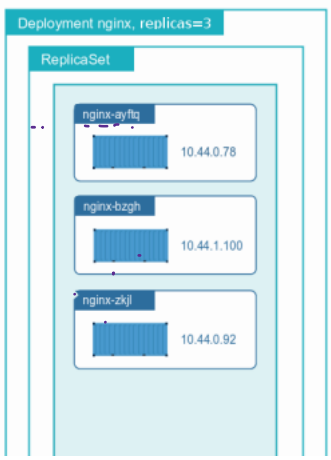
Deployments

- A Deployment provides declarative updates for Pods and ReplicaSets
- Deployment creates ReplicaSet, which creates the Pods
- Updating a deployment creates new ReplicaSet and updates the revision of the deployment.
- During update pods from the initial RS are scaled down, while pods from the new RS are scaled up.
- Rollback to an earlier revision, will update the revision of Deployment
- The --record flag of kubectl allows us to record current command in the annotations of the resources being created or updated
- Strategy – how to replace the old pods
  - Rolling update (default): maxUnavailable, maxSurge
  - Recreate

Working with Deployments

- Creating a deployment
  - kubectl run ghost --image=ghost --record
  - kubectl create -f dep1.yaml --record
  - dep1.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
```



```
ports:
- containerPort: 80
```

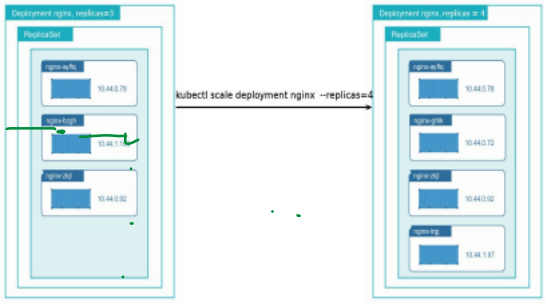
Working with Deployments (cont)

Check the status

- kubectl get deployment nginx [--watch]
- kubectl get deployment nginx -o yaml
- kubectl describe deployment nginx

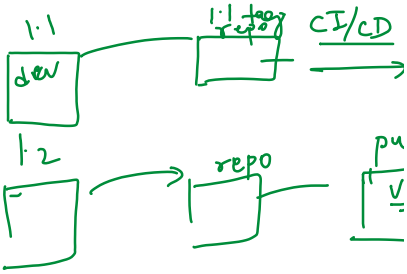
Scale a deployment

- kubectl scale deployment nginx --replicas=4



Working with Deployments (cont)

- Update a deployment
  - kubectl set image deployment/nginx nginx=nginx:1.7.9 --all=true
  - kubectl edit deployment nginx
- Check the status of a rollout
  - kubectl rollout status deployment nginx
  - kubectl rollout history deployment nginx
- Undo a rollout
  - kubectl rollout undo deployment/nginx [--to-revision=2]
- Pause and resume a deployment – allows multiple changes
  - kubectl rollout pause deployment/nginx
  - kubectl rollout resume deployment/nginx



Kubecelt get deploy

descrot nameofdep

****

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: mydeployments
spec:
  replicas: 2
  selector:
   matchLabels:
    name: deployment
  template:
    metadata:
      name: testpod
      labels:
        name: deployment
    spec:
     containers:
       - name: c00
         image: ubuntu
         command: ["/bin/bash", "-c", "while true; do echo Hello world; sleep 5; done"]
```