Lab 3 : Kubernetes Deployments

Nom du module : PaaS Environment **Année Universitaire** : 2021/2022 **Public cible** : 4 ArcTIC	**Enseignants** : ➢ Soumaya Mbarek ➢ Hamdi Gabsi

Goals :

In this lab, we will have a closer look to Kubernetes Deployments. We will explore, how Kubernetes Deployments work, and which features in terms of the rollout/rollback they offer to the Kubernetes administrator.

I. Step 0 : Access the Kubernetes Playground

We start by accessing the Katacode Kubernetes Playground.

II. Step 1: Create a Deployment

Let us create a Deployment using the following YAML file :

```yaml
# frontend-minimalistic-deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deploy
spec:
  replicas: 3
  selector:
    matchLabels:
      tier: my-matching-label
  template:
    metadata:
      labels:
        tier: my-matching-label
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
```

```
Terminal Host 1 +
controlplane $ kubectl create -f deploy.yaml
deployment.apps/frontend-deploy created
controlplane $ kubectl get deploy
NAME              READY   UP-TO-DATE   AVAILABLE   AGE
frontend-deploy   3/3     3            3           2m52s
controlplane $ kubectl get rs
NAME                         DESIRED   CURRENT   READY   AGE
frontend-deploy-6bfbb9b589   3         3         3       3m20s
controlplane $
```

Lab 3 : Kubernetes Deployments

III. Step 2: Compare ReplicaSets and Deployments

What is the difference between deployment and ReplicaSet? To research on this topic, let us export the deployment and the ReplicaSet to YAML files and compare them

Note *:* "The default output format for all kubectl commands is the human readable plain-text format. To output details to your terminal window in a specific format, you can add either the -o or --output flags to a supported kubectl command."

```
Terminal Host 1 +
controlplane $ kubectl get deploy -o yaml > dep.yaml
controlplane $ kubectl get rs -o yaml > rs.yaml
controlplane $ diff dep.yaml rs.yaml
```

```
strategy:
  rollingUpdate:
    maxSurge: 25%
    maxUnavailable: 25%
  type: RollingUpdate
```

IV. Step 3: Scale-up the Number of Replicas

Let us scale up the number of replicas. First, we check the ReplicaSet, before we scale up to 5 replicas and then we print out the status of the ReplicaSet again.

```
Terminal Host 1 +
controlplane $ kubectl get rs
NAME                          DESIRED   CURRENT   READY   AGE
frontend-deploy-6bfbb9b589    3         3         3       25m
controlplane $ kubectl scale deploy frontend-deploy  --replicas 5
deployment.apps/frontend-deploy scaled
controlplane $ kubectl get rs
NAME                          DESIRED   CURRENT   READY   AGE
frontend-deploy-6bfbb9b589    5         5         5       26m
```

V. Step 4: Change Image

In this step, we will compare, what happens, if an Image of the POD template is changed for a ReplicaSet vs a Deployment. Even though a Deployment is creating a ReplicaSet, we will see, that there is a major difference.

a. Step 4.1: Change Image of a ReplicaSet

First, let's delete the object "deploy frontend-deploy". Second, we create a ReplicaSet object, before we change the image version from v3 to v2 with kubectl edit. We will see, that

Lab 3 : Kubernetes Deployments

the PODs are not changed by this. For updating the image, you need to delete a POD manually, so a new POD is created from the updated template.

```yaml
# frontend-minimalistic-rs.yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend-rs
spec:
  replicas: 3
  selector:
    matchLabels:
      tier: my-matching-label-rs
  template:
    metadata:
      labels:
        tier: my-matching-label-rs
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
```

Let us check the image version of the created PODs:

```
controlplane $ kubectl describe pods | grep image
  Normal  Pulled     68s   kubelet, node01    Container image "gcr.io/google_sam
ples/gb-frontend:v3" already present on machine
  Normal  Pulled     68s   kubelet, node01    Container image "gcr.io/google_sam
ples/gb-frontend:v3" already present on machine
  Normal  Pulled     68s   kubelet, node01    Container image "gcr.io/google_sam
ples/gb-frontend:v3" already present on machine
```

We now update the image version (V3 → V2).

```
Terminal Host 1
spec:
  replicas: 3
  selector:
    matchLabels:
      tier: my-matching-label-rs
  template:
    metadata:
      creationTimestamp: null
      labels:
        tier: my-matching-label-rs
    spec:
      containers:
      - image: gcr.io/google_samples/gb-frontend:v2
        imagePullPolicy: IfNotPresent
        name: php-redis
        resources: {}
```

Lab 3 : Kubernetes Deployments

We check POD Image Versions.

```
controlplane $ kubectl describe pods | grep image
  Normal   Pulled            110s                kubelet, node01    Container i
mage "gcr.io/google_samples/gb-frontend:v3" already present on machine
  Normal   Pulled            108s                kubelet, node01    Container i
mage "gcr.io/google_samples/gb-frontend:v3" already present on machine
  Normal   Pulled            110s                kubelet, node01    Container i
mage "gcr.io/google_samples/gb-frontend:v3" already present on machine
```

Different from what we might have expected, the POD image versions did not change. However, we can force it to change by deleting a POD. A new POD with the updated image is created automatically:

```
controlplane $ kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
frontend-rs-7v6hg   1/1     Running   0          4m21s
frontend-rs-mr48z   1/1     Running   0          4m21s
frontend-rs-msqcb   1/1     Running   0          4m21s
controlplane $ kubectl delete pod frontend-rs-7v6hg
pod "frontend-rs-7v6hg" deleted
```

```
controlplane $ kubectl describe pods | grep image
  Normal  Pulling    71s   kubelet, node01    Pulling image "gcr.io/google_sampl
es/gb-frontend:v2"
  Normal  Pulled     48s   kubelet, node01    Successfully pulled image "gcr.io/
google_samples/gb-frontend:v2"
  Normal   Pulled            5m49s               kubelet, node01    Container im
age "gcr.io/google_samples/gb-frontend:v3" already present on machine
  Normal   Pulled            5m51s               kubelet, node01    Container im
age "gcr.io/google_samples/gb-frontend:v3" already present on machine
```

Here, we can see, that the image of the PODs is updated only if the POD has been deleted. More precisely, the image of a POD cannot be changed, and changing the POD template of a ReplicaSet does not change the existing PODs in any way.

Let us now delete the ReplicaSet, so it does not confuse us later on.

```
controlplane $ kubectl delete rs frontend-rs
replicaset.apps "frontend-rs" deleted
```

VI. Step 4.2: Change Image of a Deployment

We first re-create the deployment of step 1.

Let us check the image version of the created PODs:

Lab 3 : Kubernetes Deployments

```
controlplane $ kubectl describe pods | grep image
  Normal   Pulled            18s                kubelet, node01    Container ima
ge "gcr.io/google_samples/gb-frontend:v3" already present on machine
  Normal   Pulled            22s                kubelet, node01    Container ima
ge "gcr.io/google_samples/gb-frontend:v3" already present on machine
  Normal   Pulled            17s                kubelet, node01    Container ima
ge "gcr.io/google_samples/gb-frontend:v3" already present on machine
```

a. Step 4.2.1 Update Image

We now update the image version of the Deployment the same way, as we have previously done with the ReplicaSet ;

Terminal Host 1 +

```
        tier: my-matching-label
    spec:
      containers:
      - image: gcr.io/google_samples/gb-frontend:v2
        imagePullPolicy: IfNotPresent
        name: php-redis
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
```

We observe the ReplicaSets right after having changed the image version:

```
NAME                          DESIRED   CURRENT   READY   AGE
frontend-deploy-6489864594    3         3         3       114s
frontend-deploy-6bfbb9b589    0         0         0       22m
```

```
controlplane $ kubectl describe  pods | grep image
  Normal  Pulled     4m4s  kubelet, node01    Container image "gcr.io/google_sam
ples/gb-frontend:v2" already present on machine
  Normal  Pulling    4m28s  kubelet, node01    Pulling image "gcr.io/google_samp
les/gb-frontend:v2"
  Normal  Pulled     4m8s   kubelet, node01    Successfully pulled image "gcr.io
/google_samples/gb-frontend:v2"
  Normal  Pulled     4m2s  kubelet, node01    Container image "gcr.io/google_sam
ples/gb-frontend:v2" already present on machine
```

Lab 3 : Kubernetes Deployments

Note : We see a difference between ReplicaSets and Deployments: while ReplicaSets do not update the PODs, if we change an immutable parameter of the POD template, the Deployment does so.

First, the Deployment automatically creates a new ReplicaSet. Then, it gradually spins up PODs in the new ReplicaSet, while scaling down the old ReplicaSet. In the end, only PODs with the new template parameters exist.

VII. Step 5: Rollout, Rollback

Now we will explore another feature of the Deployments: Rollout and Rollback. We start by looking at the rollout history:

```
controlplane $ kubectl rollout history deployment frontend-deploy
deployment.apps/frontend-deploy
REVISION  CHANGE-CAUSE
1         <none>
2         <none>
```

The first revision was the one, which we initially created, using Image version v3, while the second revision has Image version v2.

a. Step 5.1: Rollout Undo

We have rolled back to the original version v3.

```
controlplane $ kubectl rollout undo deployment frontend-deploy
deployment.apps/frontend-deploy rolled back
controlplane $ kubectl describe pods | grep image
  Normal  Pulled     17s   kubelet, node01    Container image "gcr.io/google_sam
ples/gb-frontend:v3" already present on machine
  Normal  Pulled     15s   kubelet, node01    Container image "gcr.io/google_sam
ples/gb-frontend:v3" already present on machine
  Normal  Pulled     19s   kubelet, node01    Container image "gcr.io/google_sam
ples/gb-frontend:v3" already present on machine
```

Note : per default, only the last two revisons are kept. This can be changed with the optional parameter ".spec.*revisionHistoryLimit*".