Create Kubernetes Deployment

Objective: Create a nginx deployment, use kubectl to list information about the deployment and update the deployment.

Preparation: You need to have a Kubernetes cluster, and the kubectl command-line tool must be configured to communicate with your cluster. If you do not already have a cluster, you can create one by using Minikube. Also, navigate to the appropriate lab folder and confirm the below mentioned files are present.

Outcome: Once complete the participant will have experience in creating deployments in YAML files, lauching them with kubectl and then updating them.

Data Files: `deployment.yaml, deployment-scale.yaml, deployment-update.yaml`

Step 1. Creating and exploring an nginx deployment

You can run an application by creating a Kubernetes Deployment object, and you can describe a Deployment in a YAML file. For example, this YAML file describes a Deployment that runs the nginx:1.7.9 Docker image:

deployment.yaml

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2 # tells deployment to run 2 pods matching the t
emplate
  template: # create pods using pod definition in this templa
te
    metadata:
      # unlike pod-nginx.yaml, the name is not included in th
e meta data as a unique name is
      # generated from the deployment name
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Create a Deployment based on the YAML file:

```
$ kubectl create -f <path-to-file>
# Or, online: http://k8s.io/docs/tutorials/stateless-applicat
ion/deployment.yaml
```

> *NOTE: You are creating a file from the above mentioned "deployment.yaml" example. Simply create the file and then move to create it in #1.*

2. Display information about the Deployment:

```
$ kubectl describe deployment nginx-deployment
```

Output:

```
 user@computer:~/kubernetes.github.io$ kubectl describe deplo
yment nginx-deployment
 Name:     nginx-deployment
 Namespace:    default
 CreationTimestamp:  Tue, 30 Aug 2016 18:11:37 -0700
 Labels:     app=nginx
 Selector:   app=nginx
 Replicas:   2 updated | 2 total | 2 available | 0 unavailabl
e
 StrategyType:   RollingUpdate
 MinReadySeconds:  0
 RollingUpdateStrategy:  1 max unavailable, 1 max surge
 OldReplicaSets:   <none>
 NewReplicaSet:    nginx-deployment-1771418926 (2/2 replicas
created)
 No events.
```

3. List the pods created by the deployment label:

```
$ kubectl get pods -l app=nginx
```

Output:

```
  NAME                                READY     STATUS    RES
TARTS   AGE
  nginx-deployment-1771418926-7o5ns   1/1       Running   0
        16h
  nginx-deployment-1771418926-r18az   1/1       Running   0
        16h
```

4. Display information about a pod:

```
$ kubectl describe pod <pod-name>
```

Where `<pod-name>` is the name of one of your pods.

Step 2. Updating the deployment

You can update the deployment by applying a new YAML file. This YAML file specifies that the deployment should be updated to use nginx 1.8.

deployment-update.yaml

```
apiVersion: extensions/v1beta1
```

```
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.8 // Update the version of nginx from
1.7.9 to 1.8
        ports:
        - containerPort: 80
```

Apply the new YAML file:

```
$ kubectl apply -f <path-to-file>
```

2. Watch the deployment create pods with new names and delete the old pods:

```
$ kubectl get pods -l app=nginx
```

Step 3. Scaling the application by increasing the replica count

You can increase the number of pods in your Deployment by applying a new YAML file. This YAML file sets `replicas` to 4, which specifies that the Deployment should have four pods:

deployment-scale.yaml

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 4   // Update the replicas from 2 to 4 (Delete th
is comment)
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.8
        ports:
        - containerPort: 80
```

1. Apply the new YAML file:

```
$ kubectl apply -f <path-to-file>
// Or, online: http://k8s.io/docs/tutorials/stateless-application/deployment-scale.yaml
```

2. Verify that the Deployment has four pods:

```
$ kubectl get pods -l app=nginx
```

The output is similar to this:

```
  NAME                                READY     STATUS    RESTARTS   AGE
  nginx-deployment-148880595-4zdqq    1/1       Running   0          25s
  nginx-deployment-148880595-6zgi1    1/1       Running   0          25s
  nginx-deployment-148880595-fxcez    1/1       Running   0          2m
  nginx-deployment-148880595-rwovn    1/1       Running   0          2m
```

Step 4. Deleting a deployment

1. Delete the deployment, by name:

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

Conclusion

Now, you know how to create deployments and update them using kubeclt, and this is just the tip of the iceberg. Remember, a Deployment provides declarative updates for Pods and ReplicaSets (the next-generation ReplicationController). You only need to describe the desired state in a Deployment object, and the Deployment controller will change the actual state to the desired state at a controlled rate for you. You can define Deployments to create new ReplicaSets, or remove existing Deployments and adopt all of their resources with new Deployments.