Creating a Deployment

Creating a Deployment isn't that different from creating a ReplicationController. A Deployment is also composed of a label selector, a desired replica count, and a pod template. In addition to that, it also contains a field, which specifies a deployment strategy that defines how an update should be performed when the Deployment resource is modified.

```
cat << EOF > kubia-deployment-v1.yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: kubia
spec:
  replicas: 3
  template:
    metadata:
      name: kubia
      labels:
        app: kubia
    spec:
      containers:
      - image: luksa/kubia:v1
        name: nodejs
EOF
```

Because the ReplicationController from before was managing a specific version of the pods, you called it `kubia-v1`. At a given point in time, the Deployment can have multiple pod versions running under its wing, so its name shouldn't reference the app version.

Be sure to include the `--record` command-line option when creating it. This records the command in the revision history, which will be useful later.

```
kubectl create -f kubia-deployment-v1.yaml --record
```

```
deployment.apps "kubia" created
```

You can use the usual `kubectl get deployment` and the `kubectl describe deployment` commands to see details of the Deployment, but let me point you to an additional command, which is made specifically for checking a Deployment's status:

```
kubectl rollout status deployment kubia
```

```
Waiting for rollout to finish: 2 of 3 updated replicas are available
...
deployment "kubia" successfully rolled out
```

```
kubectl get po
```

```
NAME                     READY     STATUS    RESTARTS   AGE
kubia-6fc8ff6566-5lk9g   1/1       Running   0          1m
kubia-6fc8ff6566-7g8tg   1/1       Running   0          1m
kubia-6fc8ff6566-bm5p6   1/1       Running   0          1m
```

ReplicationController to create pods, their names were composed of the name of the controller plus a randomly generated string (for example, `kubia-v1-m33mv`). The three pods created by the Deployment include an additional numeric value in the middle of their names.

The number corresponds to the hashed value of the pod template in the Deployment and the ReplicaSet managing these pods. As we said earlier, a Deployment doesn't manage pods directly. Instead, it creates ReplicaSets and leaves the managing to them, so let's look at the ReplicaSet created by your Deployment

```
kubectl get replicasets
```

```
NAME               DESIRED   CURRENT   READY     AGE
kubia-6fc8ff6566   3         3         3         2m
```

The ReplicaSet's name also contains the hash value of its pod template. As you'll see later, a Deployment creates multiple ReplicaSets—one for each version of the pod template. Using the hash value of the pod template like this allows the Deployment to always use the same (possibly existing) ReplicaSet for a given version of the pod template.

Expose `kubia` deployment

```
kubectl expose deployment/kubia --type="NodePort" --port 8080
```

```
service "kubia" exposed
```

```
kubectl get svc
```

```
NAME         TYPE         CLUSTER-IP    EXTERNAL-IP   PORT(S)
  AGE
kubia        NodePort     10.7.247.76   <none>        8080:32229/TCP
  44s
```

Get External IP addresses of nodes

```
kubectl get nodes -o jsonpath='{ $.items[*].status.addresses[?(@.typ
e=="ExternalIP")].address }'
```

```
35.232.43.157 35.226.116.253 104.154.32.120
```

Curl one of the IPs with NodePort port

```
while true; do curl 35.232.43.157:32229; done
```

```
This is v1 running in pod kubia-6fc8ff6566-bm5p6
This is v1 running in pod kubia-6fc8ff6566-7g8tg
This is v1 running in pod kubia-6fc8ff6566-5lk9g
This is v1 running in pod kubia-6fc8ff6566-5lk9g
This is v1 running in pod kubia-6fc8ff6566-5lk9g
This is v1 running in pod kubia-6fc8ff6566-bm5p6
This is v1 running in pod kubia-6fc8ff6566-5lk9g
```

```
^C
```