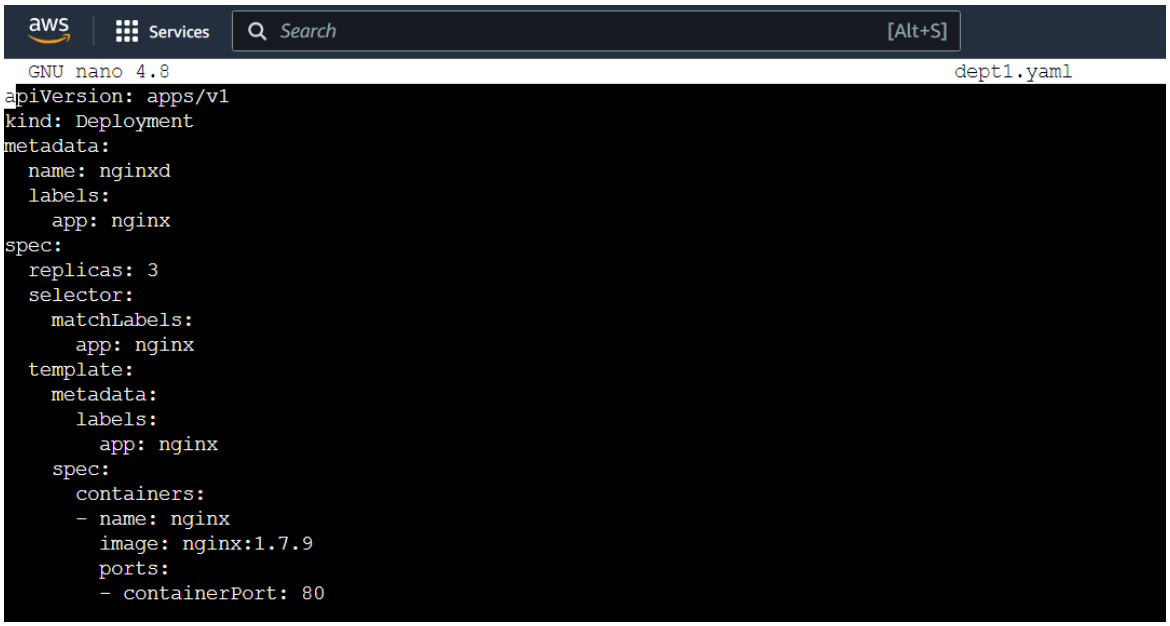
Creating a Deployment

Operation 1: Write a yaml file for the Deployment that you are creating

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginxd
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Go to command line and create a yaml file and paste the above created specs.

nano <filename>.yaml



Once done hit Ctrl+s and then Ctrl+x to save & exit

Operation 2: next thing to do is create the deployment yaml file.

kubectl create –f <filename>

```
ubuntu@ip-172-31-90-123:~$ nano dept1.yaml
ubuntu@ip-172-31-90-123:~$ kubectl create -f dept1.yaml
deployment.apps/nginxd created
ubuntu@ip-172-31-90-123:~$
```

Check if pods are created or not

kubectl get pods

```
ubuntu@ip-172-31-90-123:~$ nano dept1.yaml
ubuntu@ip-172-31-90-123:~$ kubectl create -f dept1.yaml
deployment.apps/nginxd created
ubuntu@ip-172-31-90-123:~$ kubectl get pods
NAME                                READY   STATUS        RESTARTS   AGE
nginx-deployment-86dcfdf4c6-vqldx   1/1     Terminating   0          45h
nginxd-9d6cbcc65-gpz76              1/1     Running       0          3m1s
nginxd-9d6cbcc65-nlrbl              1/1     Running       0          3m1s
nginxd-9d6cbcc65-svnps              1/1     Running       0          3m1s
ubuntu@ip-172-31-90-123:~$
```

Operation 3: You can have a look at the deployment, as well as the the replica set and the pods the deployment looks after

kubectl get deploy

```
ubuntu@ip-172-31-90-123:~$ kubectl get deploy
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
nginxd    3/3     3            3           6m15s
ubuntu@ip-172-31-90-123:~$
```

Or

kubectl get deployment

```
ubuntu@ip-172-31-90-123:~$ kubectl get deployment
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
nginxd    3/3     3            3           6m53s
ubuntu@ip-172-31-90-123:~$
```

Operation 4: now if I want to check replica sets

kubectl get rs

```
ubuntu@ip-172-31-90-123:~$ kubectl get rs
NAME                  DESIRED   CURRENT   READY   AGE
nginxd-9d6cbcc65      3         3         3       9m5s
ubuntu@ip-172-31-90-123:~$
```

Operation 5: now scale your deployments using the command below

kubectl scale --replicas=<no of replica you want> deployments <deployment name>

```
ubuntu@ip-172-31-90-123:~$ kubectl scale --replicas=4 deployments nginxd
deployment.apps/nginxd scaled
ubuntu@ip-172-31-90-123:~$
```

Now let's check our pods

kubectl get pods

```
ubuntu@ip-172-31-90-123:~$ kubectl get pods
NAME                                READY   STATUS        RESTARTS   AGE
nginx-deployment-86dcfdf4c6-vqldx   1/1     Terminating   0          45h
nginxd-9d6cbcc65-9htpk              1/1     Running       0          114s
nginxd-9d6cbcc65-gpz76              1/1     Running       0          16m
nginxd-9d6cbcc65-nlrbl              1/1     Running       0          16m
nginxd-9d6cbcc65-svnps              1/1     Running       0          16m
ubuntu@ip-172-31-90-123:~$
```

Operation 6: change the image of the deployment your deployments using the command below

kubectl set image depolyments/<deployment name> <container name> = <image name>

```
ubuntu@ip-172-31-90-123:~$ kubectl set image deployments/nginxd nginx=centos:07
deployment.apps/nginxd image updated
ubuntu@ip-172-31-90-123:~$
```

To check the image use kubectl describe deployment command

```
  Labels:   app=nginx
  Containers:
   nginx:
    Image:          centos:07
    Port:           80/TCP
    Host Port:      0/TCP
    Environment:    <none>
    Mounts:         <none>
  Volumes:          <none>
```

Operation 7: Now will check how rolling update works

kubectl rollout status deployments/<deployment name>

```
ubuntu@ip-172-31-90-123:~$ kubectl rollout status deployments/nginxd
Waiting for deployment "nginxd" rollout to finish: 2 out of 4 new replicas have been updated...
```

Command to check the status and the history the rollout that has been done –

kubectl rollout history deployment/<deployment name>

```
ubuntu@ip-172-31-90-123:~$ kubectl rollout history deployment/nginxd
deployment.apps/nginxd
REVISION  CHANGE-CAUSE
1         <none>
2         <none>

ubuntu@ip-172-31-90-123:~$
```

1) First we changed the no. replica

2) Second, we have changed the image name

Operation 8: now let's suppose the image centos:07 is not working properly now, and we wants to retrieve back our older image, for that just write the below command-

kubectl rollout undo deployment/,deployment name> --to-revision=<Revision no>

```
ubuntu@ip-172-31-90-123:~$ kubectl rollout undo deployment/nginxd --to-revision=1
deployment.apps/nginxd rolled back
ubuntu@ip-172-31-90-123:~$
```

To check-

kubectl describe deployment <deployment name>

```
Labels:  app=nginx
Containers:
 nginx:
  Image:        nginx:1.7.9
  Port:         80/TCP
  Host Port:    0/TCP
  Environment:  <none>
  Mounts:       <none>
Volumes:        <none>
```

And our rollout status is also done

```
ubuntu@ip-172-31-90-123:~$ kubectl rollout status deployments/nginxd
deployment "nginxd" successfully rolled out
ubuntu@ip-172-31-90-123:~$
```