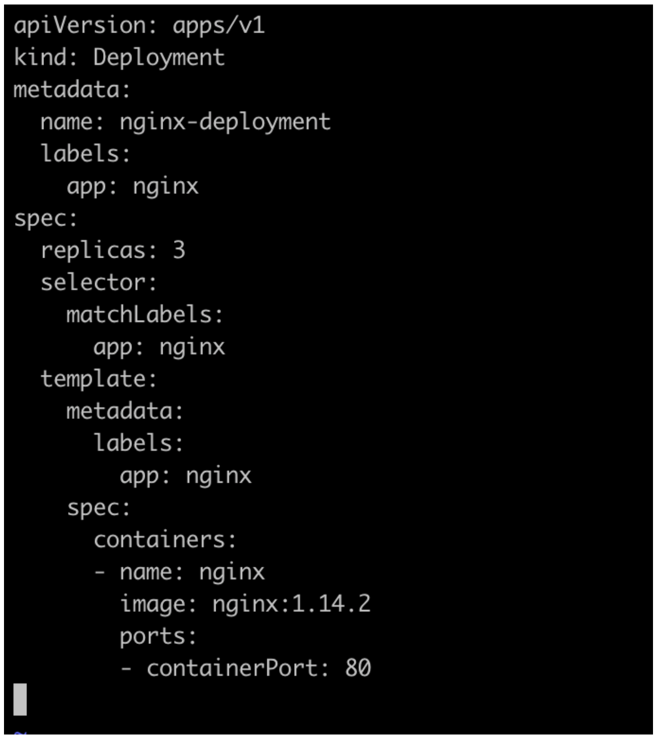
Kubernetes Deployment & Replica Sets

Kubernetes Deployment:
Kubernetes has the capability of auto-healing and autoscaling capability and a pod alone cannot achieve this. Therefore, it is ideal to use Kubernetes deployment. In other words, we should always deploy our application as a deployment but not as a pod directly. When we create a deployment resource, it will create a replica set (a Kubernetes controller) and will create pods.

Replica Set:
Replica set is a Kubernetes controller that implements the auto healing feature of the pod. When you create a deployment, a replica set is automatically created which is responsible for tracking controller behaviour in Kubernetes. Controllers in general, always ensure that the desired state is always present on the actual cluster. Desired state and the actual must be same.

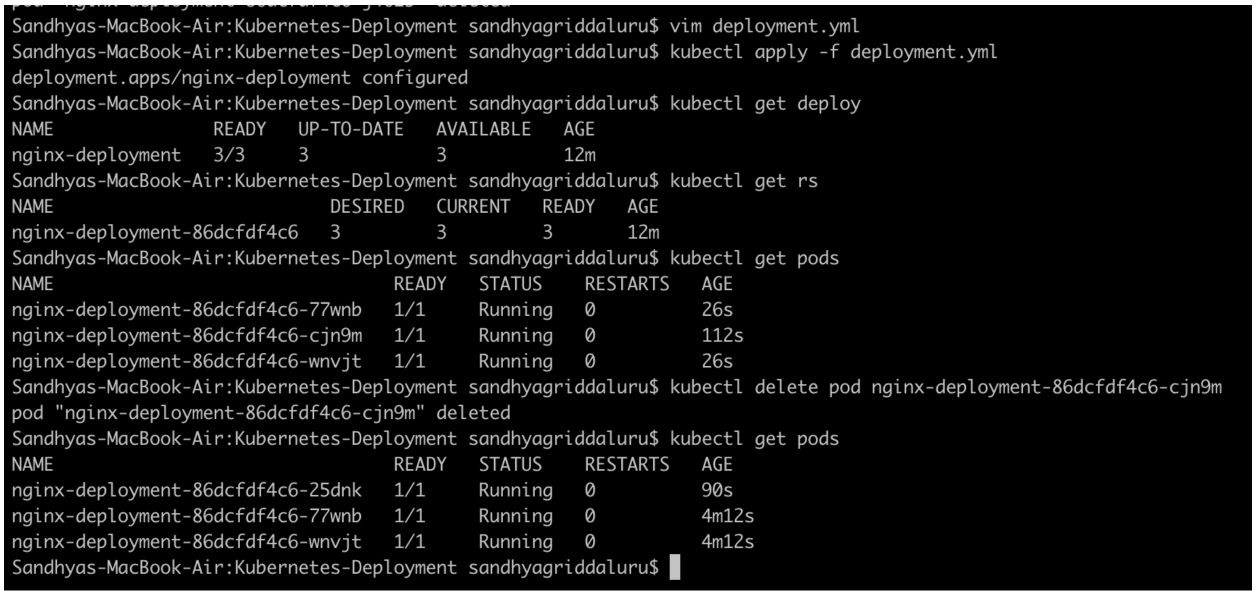
Steps:

1. Create a deployment file



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

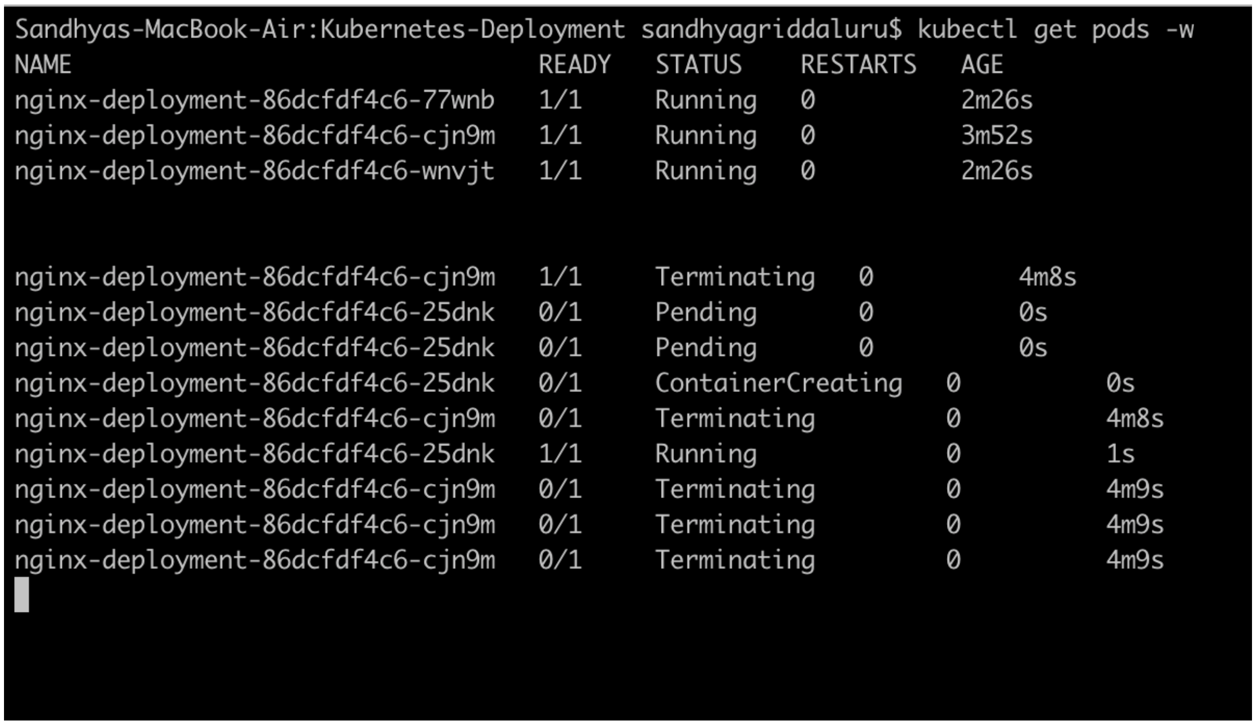
2. Start the deployment using **“kubectl apply -f deployment_filename.yml”**



```
Sandhyas-MacBook-Air:Kubernetes-Deployment sandhyagriddaluru$ vim deployment.yml
Sandhyas-MacBook-Air:Kubernetes-Deployment sandhyagriddaluru$ kubectl apply -f deployment.yml
deployment.apps/nginx-deployment configured
Sandhyas-MacBook-Air:Kubernetes-Deployment sandhyagriddaluru$ kubectl get deploy
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3/3     3            3           12m
Sandhyas-MacBook-Air:Kubernetes-Deployment sandhyagriddaluru$ kubectl get rs
NAME                          DESIRED   CURRENT   READY   AGE
nginx-deployment-86dcfdf4c6   3         3         3       12m
Sandhyas-MacBook-Air:Kubernetes-Deployment sandhyagriddaluru$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-86dcfdf4c6-77wnb   1/1     Running   0          26s
nginx-deployment-86dcfdf4c6-cjn9m   1/1     Running   0          112s
nginx-deployment-86dcfdf4c6-wnvjt   1/1     Running   0          26s
Sandhyas-MacBook-Air:Kubernetes-Deployment sandhyagriddaluru$ kubectl delete pod nginx-deployment-86dcfdf4c6-cjn9m
pod "nginx-deployment-86dcfdf4c6-cjn9m" deleted
Sandhyas-MacBook-Air:Kubernetes-Deployment sandhyagriddaluru$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-86dcfdf4c6-25dnk   1/1     Running   0          90s
nginx-deployment-86dcfdf4c6-77wnb   1/1     Running   0          4m12s
nginx-deployment-86dcfdf4c6-wnvjt   1/1     Running   0          4m12s
Sandhyas-MacBook-Air:Kubernetes-Deployment sandhyagriddaluru$
```

Delete a pod and notice that the pod is automatically created again because we have set our desired state as 3 in our deployment file for replicas.

To watch the activity of the pods use the command **“kubectl get pods -w”**



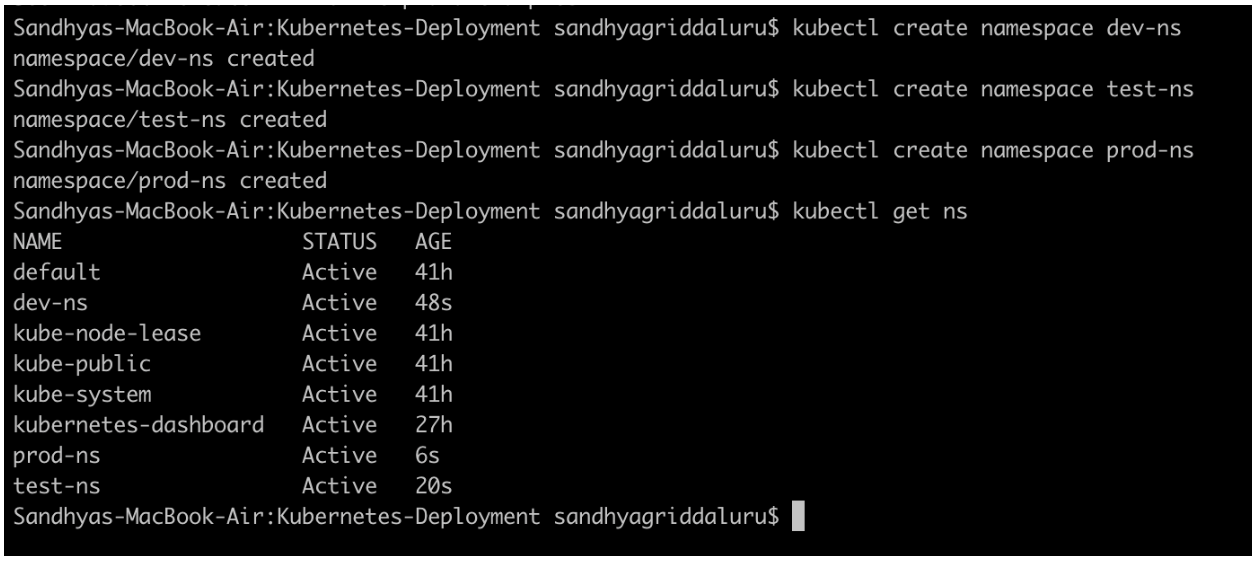
```
Sandhyas-MacBook-Air:Kubernetes-Deployment sandhyagriddaluru$ kubectl get pods -w
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-86dcfdf4c6-77wnb   1/1     Running   0          2m26s
nginx-deployment-86dcfdf4c6-cjn9m   1/1     Running   0          3m52s
nginx-deployment-86dcfdf4c6-wnvjt   1/1     Running   0          2m26s


nginx-deployment-86dcfdf4c6-cjn9m   1/1     Terminating   0          4m8s
nginx-deployment-86dcfdf4c6-25dnk   0/1     Pending       0          0s
nginx-deployment-86dcfdf4c6-25dnk   0/1     Pending       0          0s
nginx-deployment-86dcfdf4c6-25dnk   0/1     ContainerCreating   0          0s
nginx-deployment-86dcfdf4c6-cjn9m   0/1     Terminating         0          4m8s
nginx-deployment-86dcfdf4c6-25dnk   1/1     Running             0          1s
nginx-deployment-86dcfdf4c6-cjn9m   0/1     Terminating         0          4m9s
nginx-deployment-86dcfdf4c6-cjn9m   0/1     Terminating         0          4m9s
nginx-deployment-86dcfdf4c6-cjn9m   0/1     Terminating         0          4m9s
```

Namespaces:

Before we create pod, we create a namespace which is a virtual folder. It is used to place objects. It's a way to organize cluster into sub clusters.

"kubectl get ns" is used to list all the available namespaces.

"kubectl create namespace new_space" creates new namespace of name 'new_space'.



```
Sandhyas-MacBook-Air:Kubernetes-Deployment sandhyagriddaluru$ kubectl create namespace dev-ns
namespace/dev-ns created
Sandhyas-MacBook-Air:Kubernetes-Deployment sandhyagriddaluru$ kubectl create namespace test-ns
namespace/test-ns created
Sandhyas-MacBook-Air:Kubernetes-Deployment sandhyagriddaluru$ kubectl create namespace prod-ns
namespace/prod-ns created
Sandhyas-MacBook-Air:Kubernetes-Deployment sandhyagriddaluru$ kubectl get ns
NAME                   STATUS   AGE
default                Active   41h
dev-ns                 Active   48s
kube-node-lease        Active   41h
kube-public            Active   41h
kube-system            Active   41h
kubernetes-dashboard   Active   27h
prod-ns                Active   6s
test-ns                Active   20s
Sandhyas-MacBook-Air:Kubernetes-Deployment sandhyagriddaluru$
```

Difference between Replication Controller and Replica Set:

While both Replication Controllers and Replica Sets are used to manage replica pods, Replica Sets provide more features and flexibility, making them the preferred choice for managing replication in modern Kubernetes deployments. Deployments are even more powerful and are frequently preferred for managing production applications.

- Replication controllers only uses the equality-based selector. This means that the selector can only match labels using equality operators (=, ==).
- In addition to the equality-based selector, Replica set supports the set-based selector. This enables more complex label matching to be performed using set-based operators (in, notin, exists, doesnotexist).