Assignment 2

Tasks:

1. You are tasked with creating a simple pod in your Kubernetes cluster. The pod should run a container using the `busybox` image.

2. Change the image name from busybox to nginx, also check that pod is running well.

3. Create a ReplicaSet named "app-replicaset" managing three replicas of an application pod using the `nginx:v1` image.

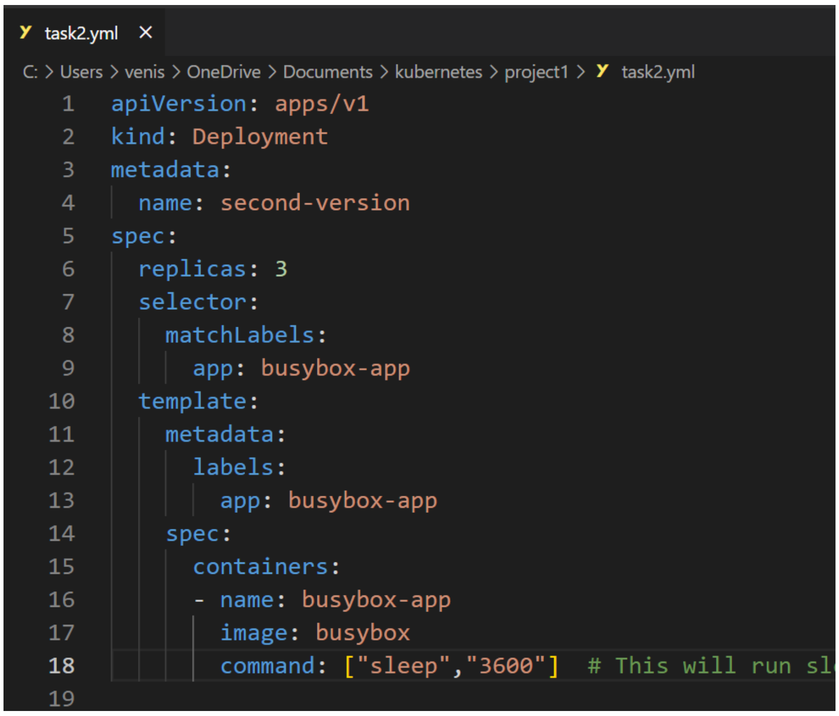
4. Create a Deployment named "app-deployment" managing four replicas of an application pod using the `nginx:alpine` image.

5. Explain how to automatically roll back to the previous version using the "app-deployment."

6. Describe the differences between a ClusterIP service and a LoadBalancer service, providing a use case for each.

1. Here we have written an .yml file which is of type Deployment and it is using the container image as “busybox” and its name is “mycontainer”.

   Here commad: [“sleep”,”3600”] will ensures that sleep command will run for 1 hour and because of that pod will be keep on running state.

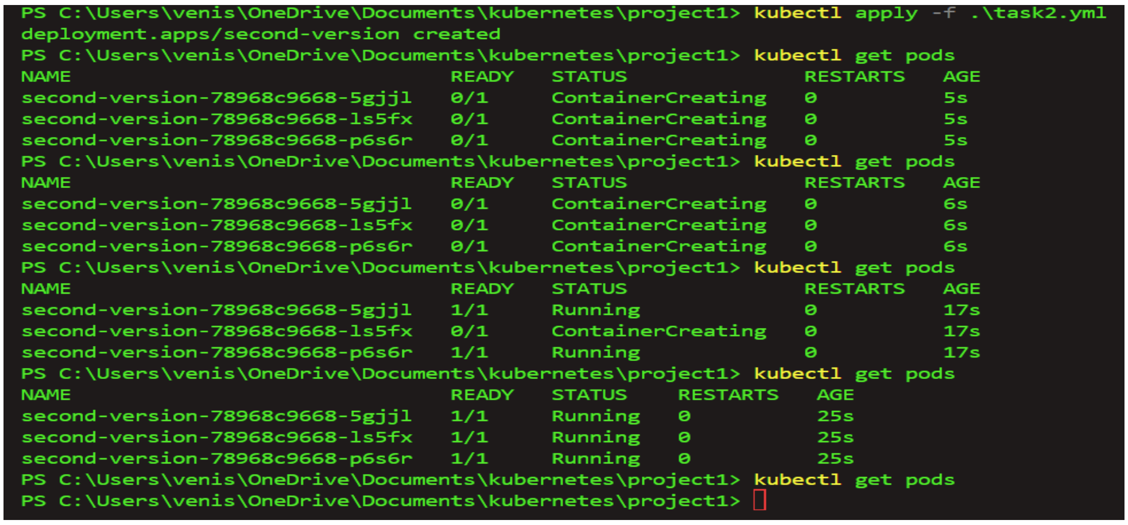


task2.yml ×

C: > Users > venis > OneDrive > Documents > kubernetes > project1 > task2.yml

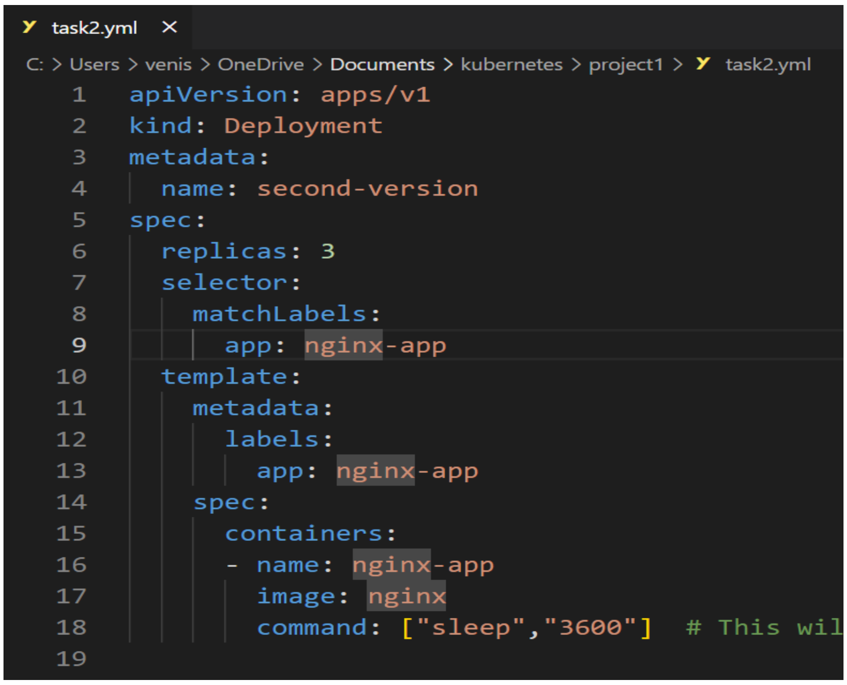
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: second-version
spec:
  replicas: 3
  selector:
    matchLabels:
      app: busybox-app
  template:
    metadata:
      labels:
        app: busybox-app
    spec:
      containers:
      - name: busybox-app
        image: busybox
        command: ["sleep","3600"]  # This will run sl

```

- Now lets apply this file using command kubectl apply -f filename
  Here we can see that when we try to access pods using command kubectl get pods Container is created and they are running.



```
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl apply -f .\task2.yml
deployment.apps/second-version created
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl get pods
NAME                              READY   STATUS              RESTARTS   AGE
second-version-78968c9668-5gjjl   0/1     ContainerCreating   0          5s
second-version-78968c9668-ls5fx   0/1     ContainerCreating   0          5s
second-version-78968c9668-p6s6r   0/1     ContainerCreating   0          5s
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl get pods
NAME                              READY   STATUS              RESTARTS   AGE
second-version-78968c9668-5gjjl   0/1     ContainerCreating   0          6s
second-version-78968c9668-ls5fx   0/1     ContainerCreating   0          6s
second-version-78968c9668-p6s6r   0/1     ContainerCreating   0          6s
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl get pods
NAME                              READY   STATUS              RESTARTS   AGE
second-version-78968c9668-5gjjl   1/1     Running             0          17s
second-version-78968c9668-ls5fx   0/1     ContainerCreating   0          17s
second-version-78968c9668-p6s6r   1/1     Running             0          17s
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
second-version-78968c9668-5gjjl   1/1     Running   0          25s
second-version-78968c9668-ls5fx   1/1     Running   0          25s
second-version-78968c9668-p6s6r   1/1     Running   0          25s
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl get pods
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1>
```

2. Now change the container image from busybox to nginx and run command kubectl apply -f filename and here we can see the changes after getting pods.

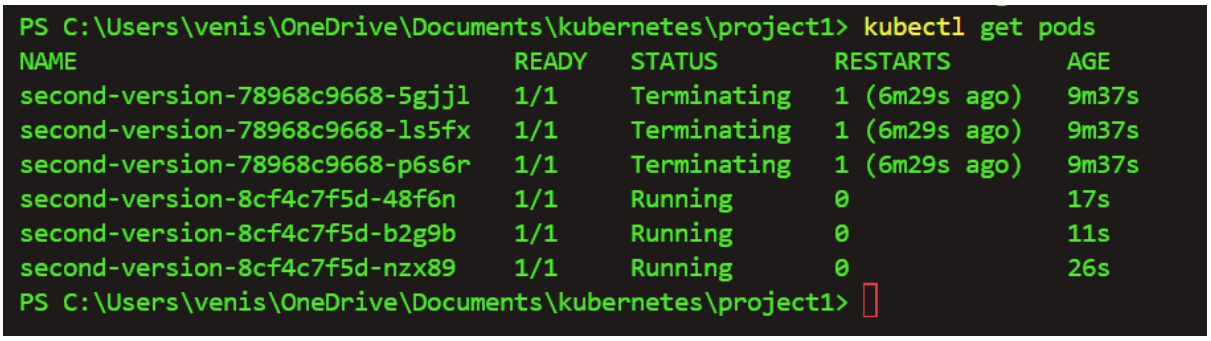


task2.yml ×

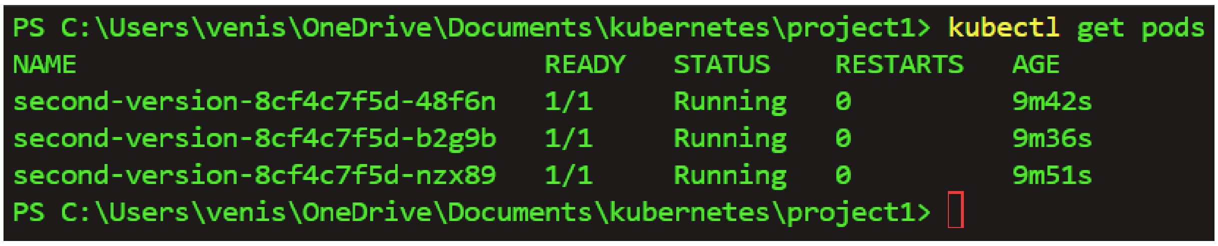
C: > Users > venis > OneDrive > Documents > kubernetes > project1 > task2.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: second-version
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx-app
  template:
    metadata:
      labels:
        app: nginx-app
    spec:
      containers:
      - name: nginx-app
        image: nginx
        command: ["sleep","3600"]  # This wil

```

- Here we can see that old pods are terminating and new pods are starting and Eventually only pods which are running on nginx image are running.

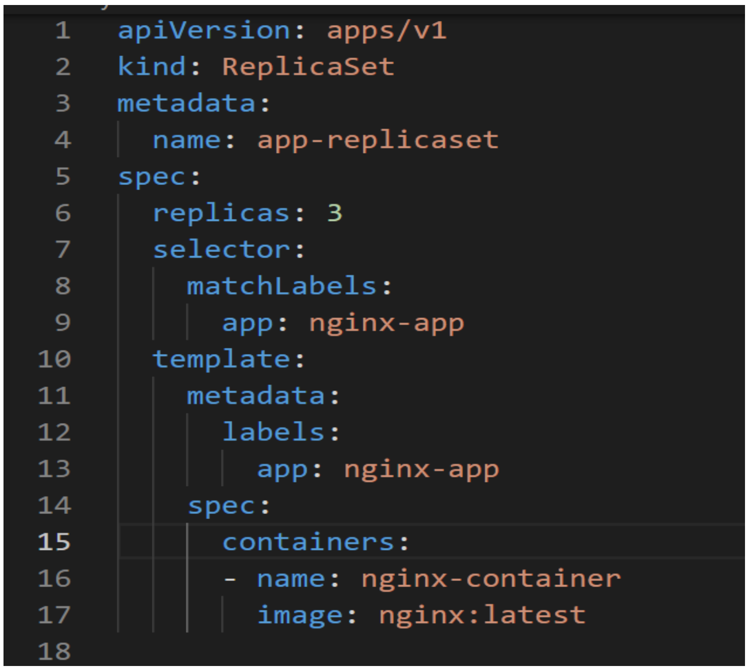


```
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl get pods
NAME                               READY   STATUS        RESTARTS        AGE
second-version-78968c9668-5gjjl    1/1     Terminating   1 (6m29s ago)   9m37s
second-version-78968c9668-ls5fx    1/1     Terminating   1 (6m29s ago)   9m37s
second-version-78968c9668-p6s6r    1/1     Terminating   1 (6m29s ago)   9m37s
second-version-8cf4c7f5d-48f6n     1/1     Running       0               17s
second-version-8cf4c7f5d-b2g9b     1/1     Running       0               11s
second-version-8cf4c7f5d-nzx89     1/1     Running       0               26s
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1>
```



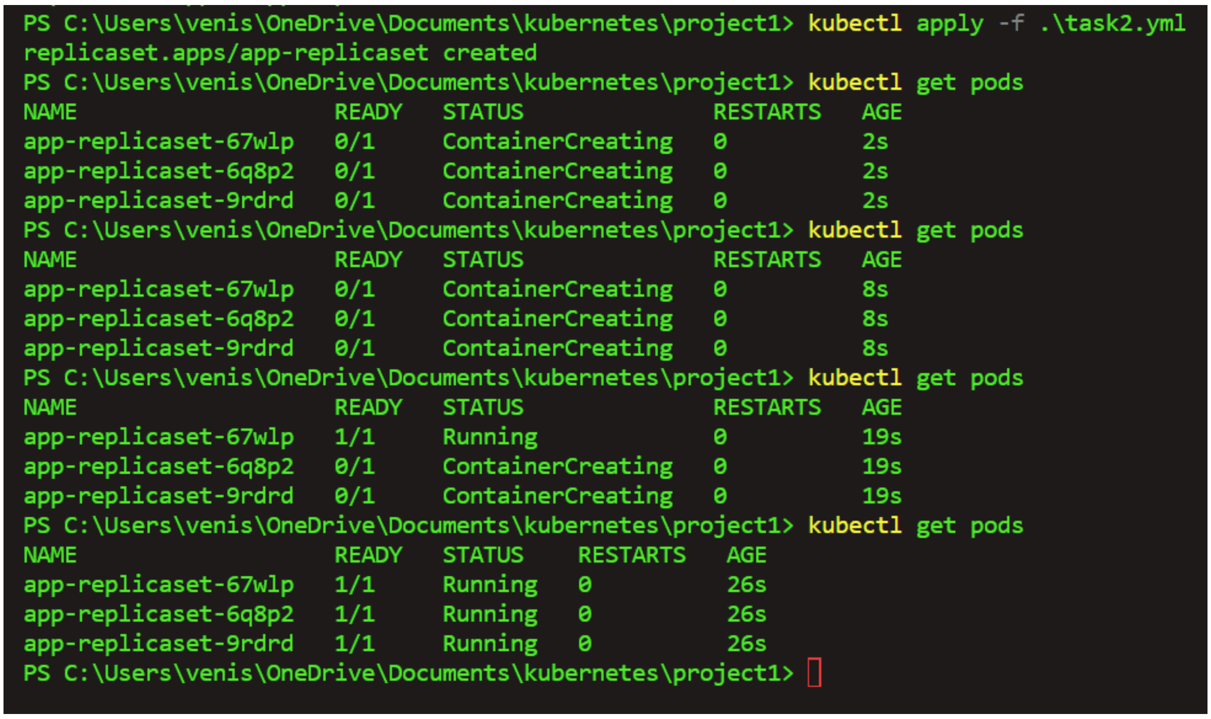
```
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl get pods
NAME                             READY   STATUS    RESTARTS   AGE
second-version-8cf4c7f5d-48f6n   1/1     Running   0          9m42s
second-version-8cf4c7f5d-b2g9b   1/1     Running   0          9m36s
second-version-8cf4c7f5d-nzx89   1/1     Running   0          9m51s
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1>
```

3. Now lets create a replicaset named “app-replicaset” which will manage 3 replicas of an application, here is out .yml file.



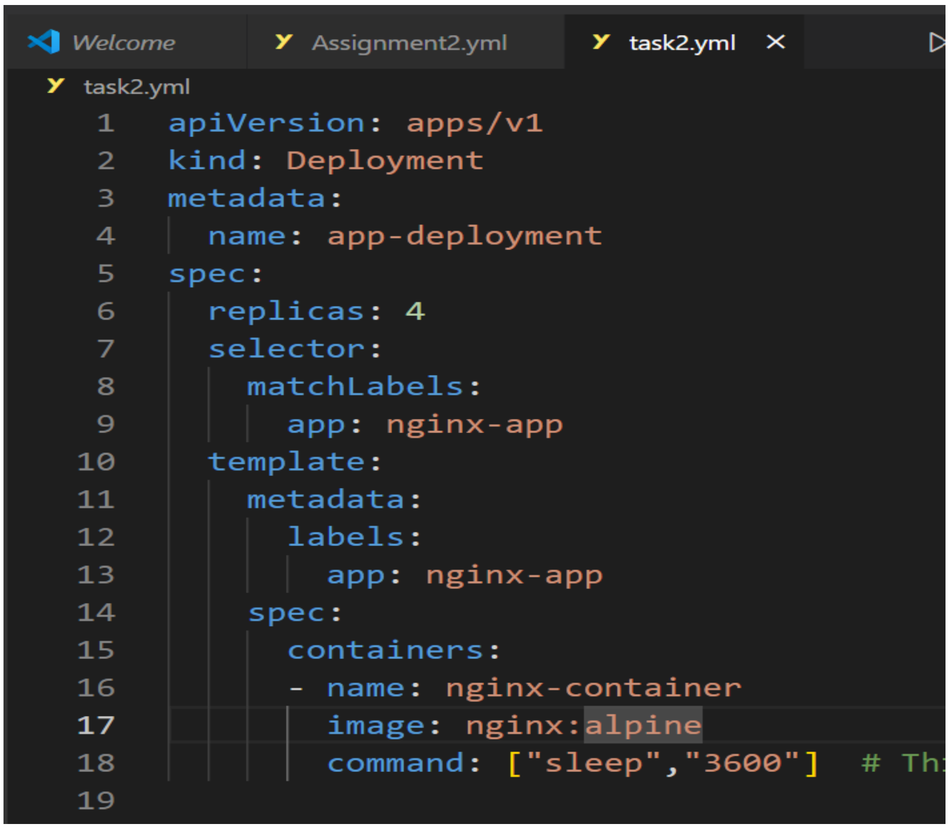
```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: app-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx-app
  template:
    metadata:
      labels:
        app: nginx-app
    spec:
      containers:
      - name: nginx-container
        image: nginx:latest

```

- Here we can see that 3 pods are running.



```
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl apply -f .\task2.yml
replicaset.apps/app-replicaset created
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl get pods
NAME                   READY   STATUS              RESTARTS   AGE
app-replicaset-67wlp   0/1     ContainerCreating   0          2s
app-replicaset-6q8p2   0/1     ContainerCreating   0          2s
app-replicaset-9rdrd   0/1     ContainerCreating   0          2s
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl get pods
NAME                   READY   STATUS              RESTARTS   AGE
app-replicaset-67wlp   0/1     ContainerCreating   0          8s
app-replicaset-6q8p2   0/1     ContainerCreating   0          8s
app-replicaset-9rdrd   0/1     ContainerCreating   0          8s
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl get pods
NAME                   READY   STATUS              RESTARTS   AGE
app-replicaset-67wlp   1/1     Running             0          19s
app-replicaset-6q8p2   0/1     ContainerCreating   0          19s
app-replicaset-9rdrd   0/1     ContainerCreating   0          19s
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl get pods
NAME                   READY   STATUS    RESTARTS   AGE
app-replicaset-67wlp   1/1     Running   0          26s
app-replicaset-6q8p2   1/1     Running   0          26s
app-replicaset-9rdrd   1/1     Running   0          26s
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> 
```

4. Now creating an Deployment named "app-deployment" which manages 4 replicas using nginx:alpine.

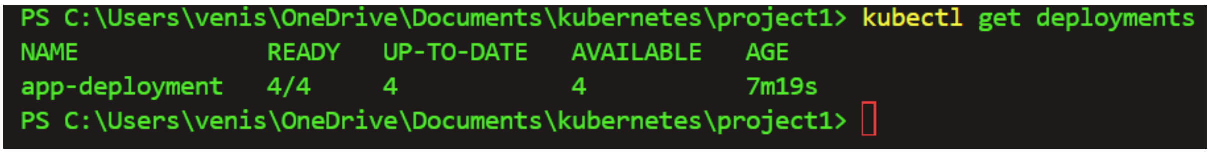


Welcome | Assignment2.yml | task2.yml

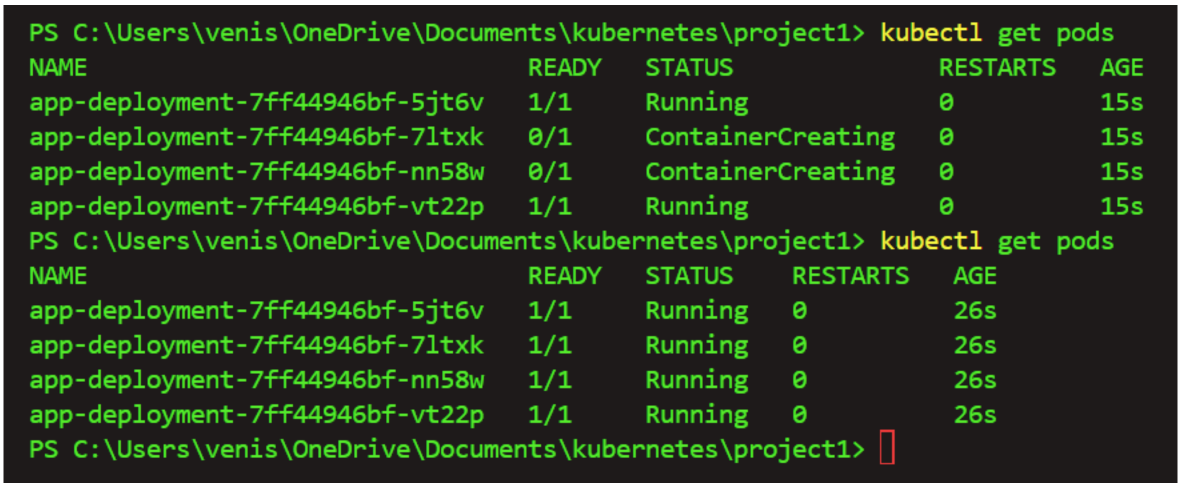
task2.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app-deployment
spec:
  replicas: 4
  selector:
    matchLabels:
      app: nginx-app
  template:
    metadata:
      labels:
        app: nginx-app
    spec:
      containers:
      - name: nginx-container
        image: nginx:alpine
        command: ["sleep","3600"]  # Th

```

- Here we can see that deployment has been created and 4 pods are also running.

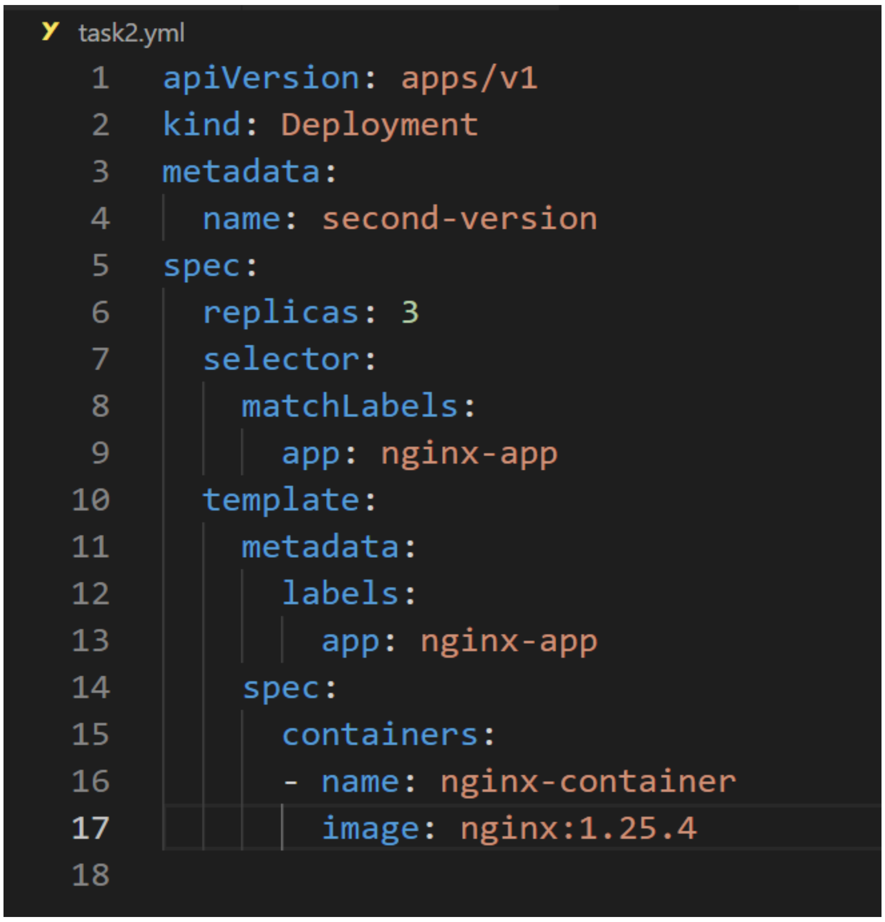


```
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl get deployments
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
app-deployment   4/4     4            4           7m19s
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1>
```



```
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl get pods
NAME                              READY   STATUS              RESTARTS   AGE
app-deployment-7ff44946bf-5jt6v   1/1     Running             0          15s
app-deployment-7ff44946bf-7ltxk   0/1     ContainerCreating   0          15s
app-deployment-7ff44946bf-nn58w   0/1     ContainerCreating   0          15s
app-deployment-7ff44946bf-vt22p   1/1     Running             0          15s
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
app-deployment-7ff44946bf-5jt6v   1/1     Running   0          26s
app-deployment-7ff44946bf-7ltxk   1/1     Running   0          26s
app-deployment-7ff44946bf-nn58w   1/1     Running   0          26s
app-deployment-7ff44946bf-vt22p   1/1     Running   0          26s
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1>
```

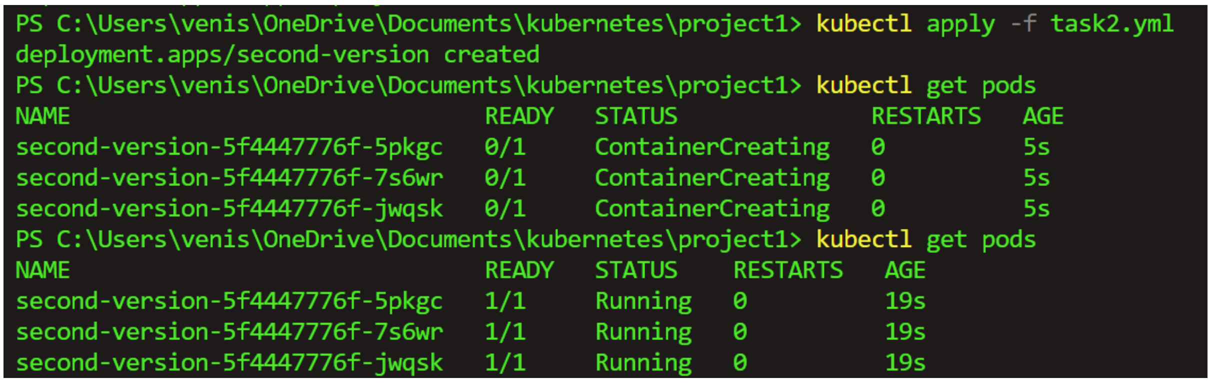
5. Here we have created deployment which uses nginx:1.25.4 which is the latest version of nginx.



task2.yml

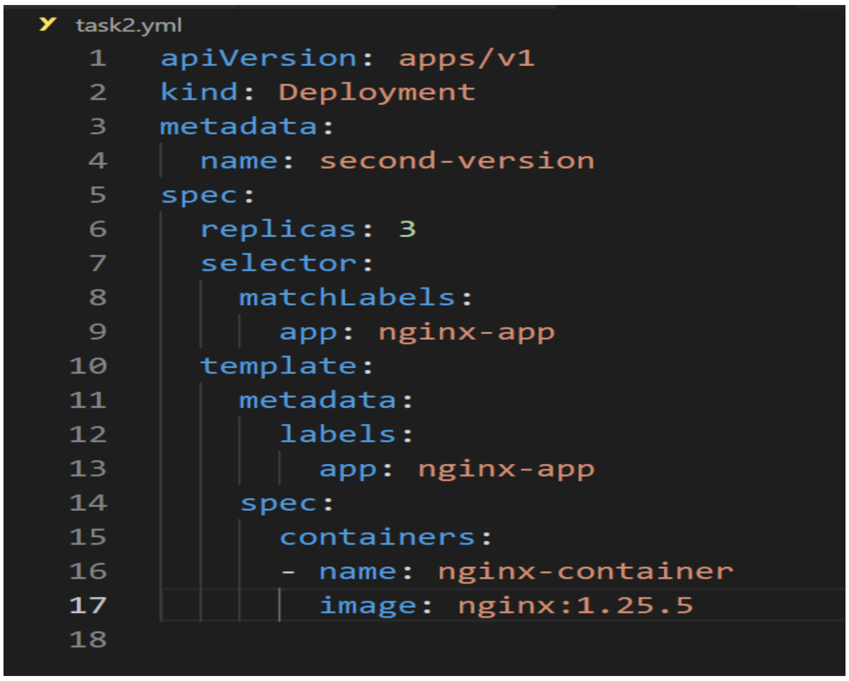
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: second-version
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx-app
  template:
    metadata:
      labels:
        app: nginx-app
    spec:
      containers:
      - name: nginx-container
        image: nginx:1.25.4

```

- Here we can see that 3 pods are running.



```
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl apply -f task2.yml
deployment.apps/second-version created
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl get pods
NAME                             READY   STATUS              RESTARTS   AGE
second-version-5f4447776f-5pkgc  0/1     ContainerCreating   0          5s
second-version-5f4447776f-7s6wr  0/1     ContainerCreating   0          5s
second-version-5f4447776f-jwqsk  0/1     ContainerCreating   0          5s
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl get pods
NAME                             READY   STATUS    RESTARTS   AGE
second-version-5f4447776f-5pkgc  1/1     Running   0          19s
second-version-5f4447776f-7s6wr  1/1     Running   0          19s
second-version-5f4447776f-jwqsk  1/1     Running   0          19s
```

- Now we are changing version of nginx image which is not available for now.

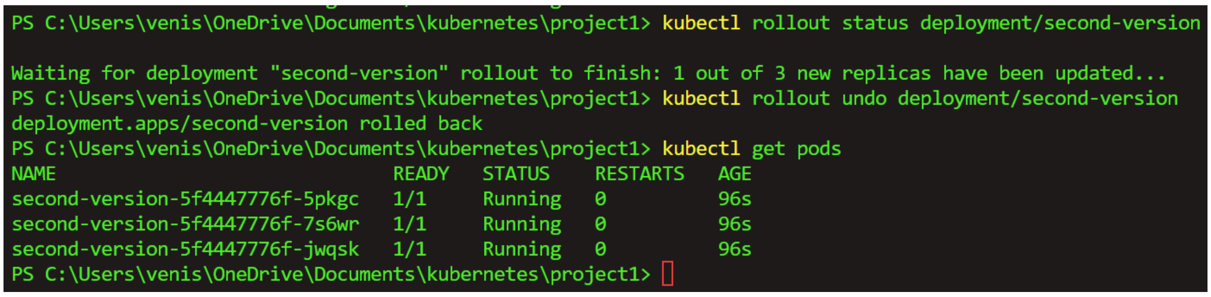


task2.yml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: second-version
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx-app
  template:
    metadata:
      labels:
        app: nginx-app
    spec:
      containers:
      - name: nginx-container
        image: nginx:1.25.5

```

- Now if we apply the changes, we can see that new pod is creating and it's giving me ErrImagePull error. Now run the command Kubectl rollout status and here 1 replicas has been updated. Now run following command to rollback to previous version of deployment.

  Kubectl rollout undo deployment/deployment_name



```
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl rollout status deployment/second-version

Waiting for deployment "second-version" rollout to finish: 1 out of 3 new replicas have been updated...
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl rollout undo deployment/second-version
deployment.apps/second-version rolled back
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> kubectl get pods
NAME                             READY   STATUS    RESTARTS   AGE
second-version-5f4447776f-5pkgc  1/1     Running   0          96s
second-version-5f4447776f-7s6wr  1/1     Running   0          96s
second-version-5f4447776f-jwqsk  1/1     Running   0          96s
PS C:\Users\venis\OneDrive\Documents\kubernetes\project1> 
```

6. ClusterIp Service Vs LoadBalancer Service

1. ClusterIP Service:

- ➢ It exposes the pods within the same kubernetes cluster to different objects. It assigns an internal IP address which can be accessible only through the same cluster.

Use Case:

- ➢ This type of services mostly used for internal communication between other parts of the same Application within the same kubernetes cluster. For example, if we have microservices architecture and in that one service needs to communicate with other service then we can expose that service using Cluster IP service.

2. LoadBalancer Service:

- ➢ LoadBalancer service exposes a set of pods to the external world outside the kubernetes cluster. It provides an external load balancer that distributes the incoming traffic among the pods evenly.

Use Case:

- ➢ This type of service mostly used for external communication to internet or external clients. For example, If our application needs to handle incoming HTTP or HTTPS traffic from users over the internet, then we can make application as a LoadBalancer service. It ensures that traffic is evenly distributed among the pods and also provides high scalability and availability.

Real-Life Example: E-Commerce Application

- ➢ Suppose we have an E-commerce application which have services like Product service, Order service, User service, Frontend service so now lets see how both services can be useful in this scenario.

1. ClusterIP Service:

- ➢ We can use clusterIP service to expose product, order and user services to each other in the same kubernetes cluster. These services will communicate with each other without being directly accessible from outside of a cluster and this provides isolation and security.
- For Example: Order service will communicate with product service for checking availability of particular product.

2. LoadBalancer Service:

- ➢ we can use LoadBalancer service, you make the e-commerce application accessible to users on the internet while providing high availability and scalability through load balancing.
- ➢ For Example: Loadbalancer distributes incoming requests among the pods running the frontend service, ensuring that application remains responsive and scalable.