THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

Containerized Web Application Deployment Using Kubernetes and Minikube

This project demonstrates the deployment of a containerized Nginx web server using **Kubernetes** on a local development environment powered by **Minikube**.

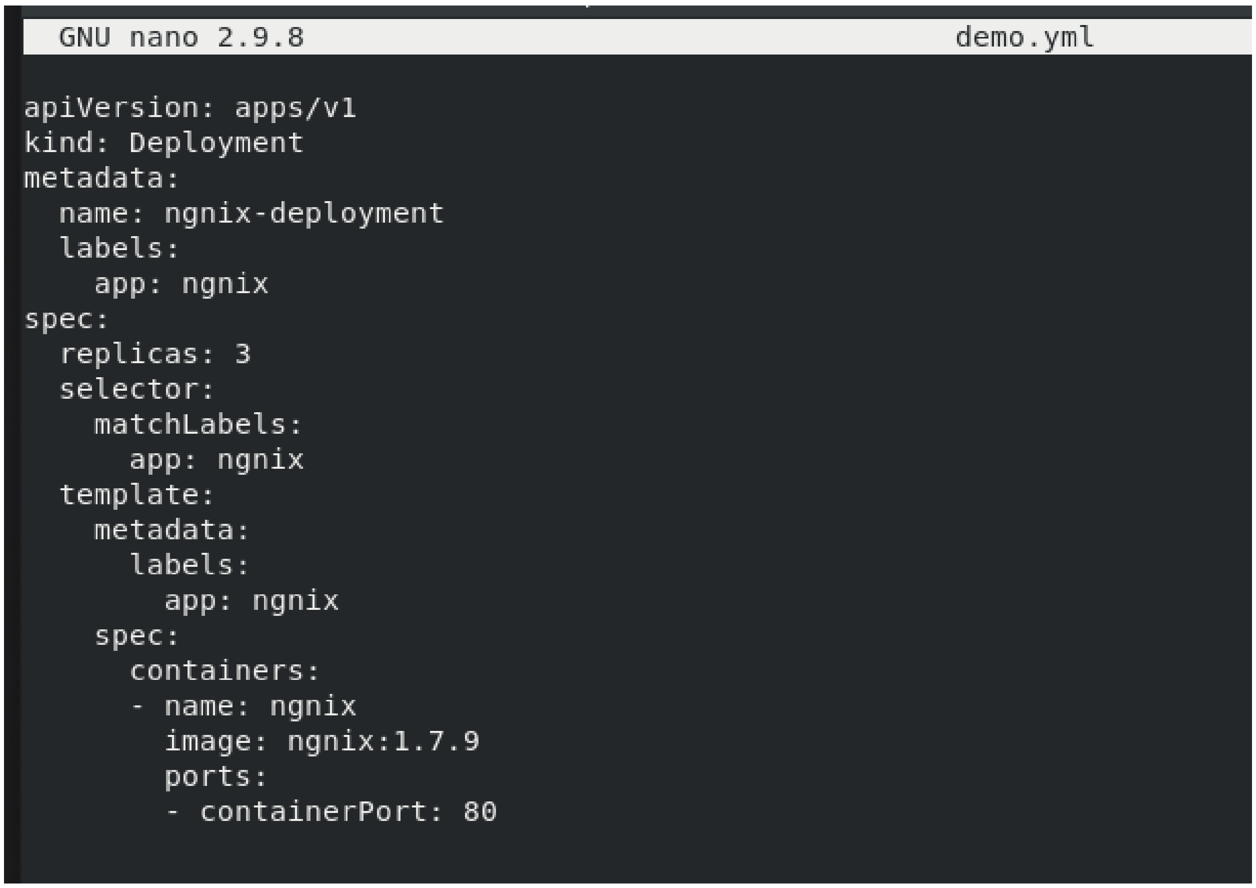
The main objective was to understand and apply core Kubernetes concepts such as creating deployments, managing replicas, exposing services, and observing pod behavior. A custom deployment configuration file (demo.yml) was created to define the structure and behavior of the application within the cluster.

Using kubectl, the deployment was applied, and the Nginx application was exposed externally through a **NodePort** service, allowing access via a web browser. Key tools and commands used include:

- kubectl for applying deployments, exposing services, scaling, and rollout management
- minikube for setting up and managing the local Kubernetes cluster

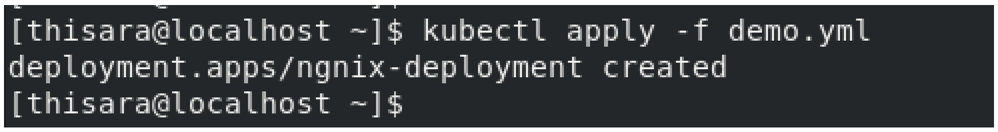
This hands-on project serves as a foundational exercise in container orchestration, service exposure, and workload scaling—critical components in modern cloud-native environments.

1. Create a Deployment



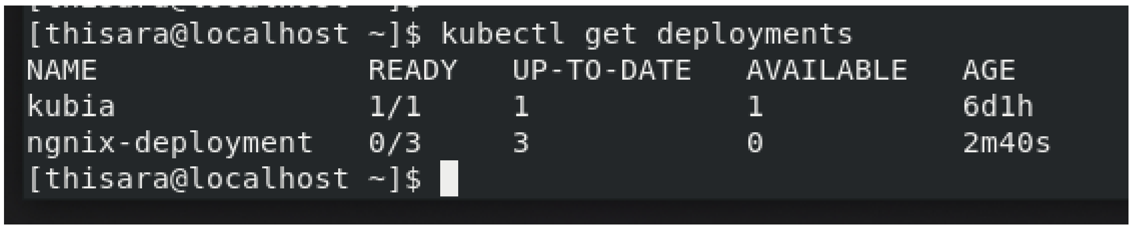
```
  GNU nano 2.9.8                                         demo.yml

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

2. Deploy the deployment file



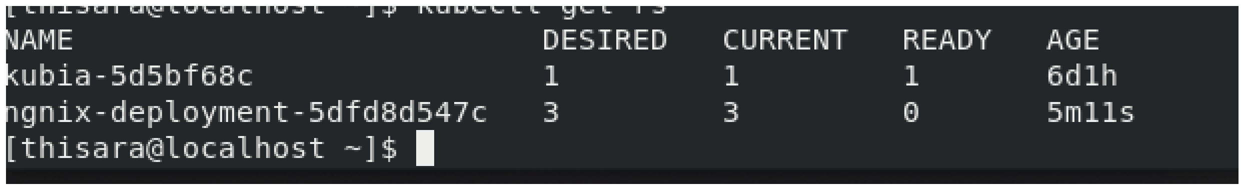
```
[thisara@localhost ~]$ kubectl apply -f demo.yml
deployment.apps/ngnix-deployment created
[thisara@localhost ~]$
```

3. Check the deployments



```
[thisara@localhost ~]$ kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
kubia              1/1     1            1           6d1h
ngnix-deployment   0/3     3            0           2m40s
[thisara@localhost ~]$
```

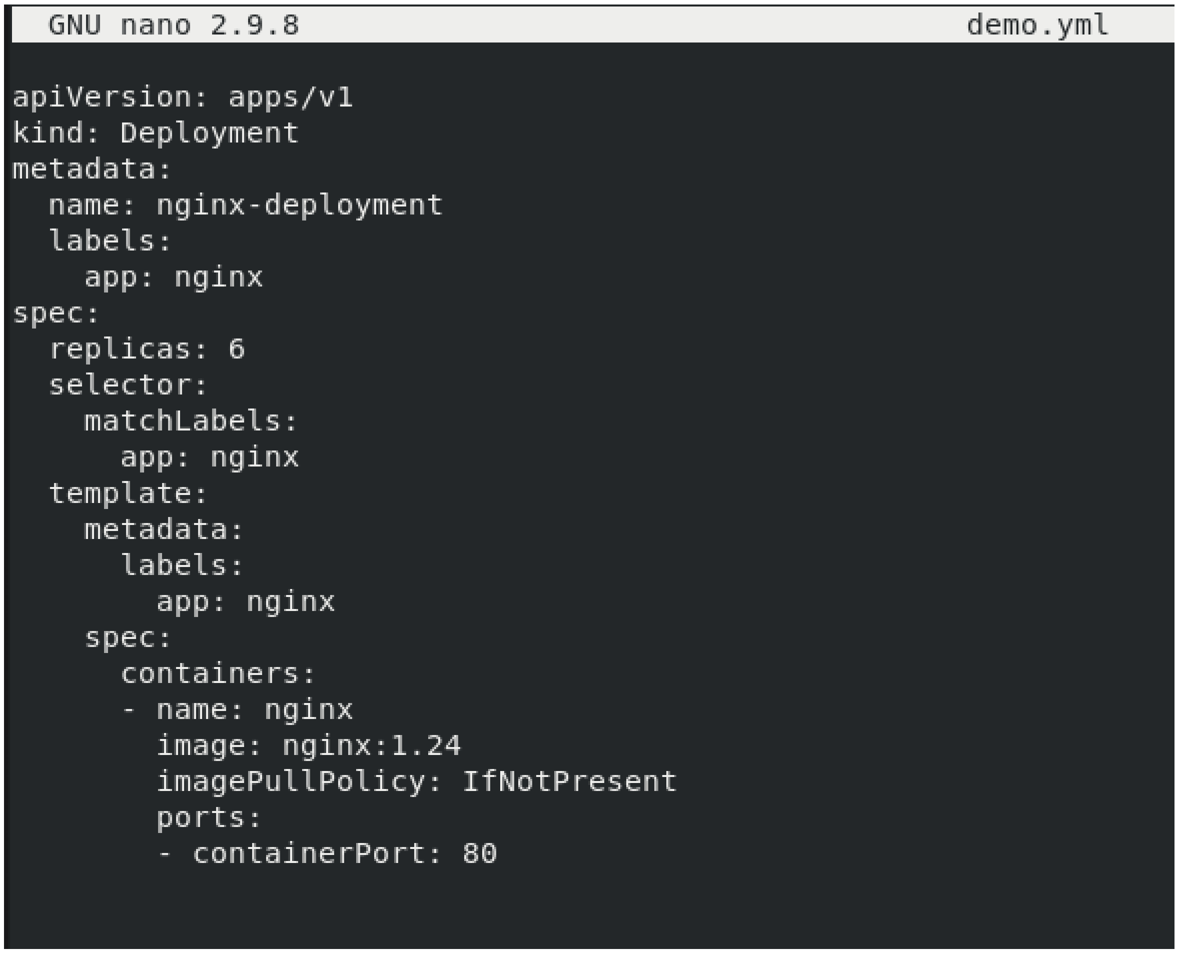
4. Check the number of replicas running



```
[thisara@localhost ~]$ kubectl get rs
NAME                          DESIRED   CURRENT   READY   AGE
kubia-5d5bf68c                1         1         1       6d1h
ngnix-deployment-5dfd8d547c   3         3         0       5m11s
[thisara@localhost ~]$ 
```

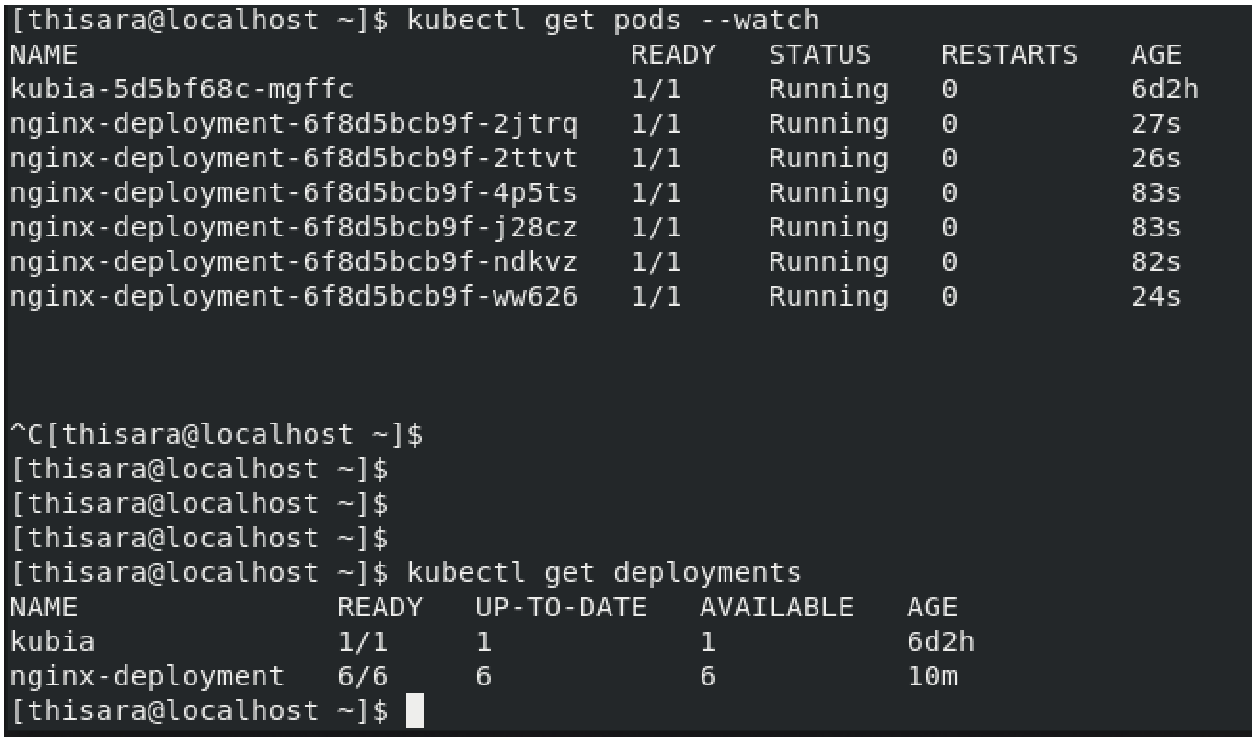
Update Deployments

1. Change the replicas and container image



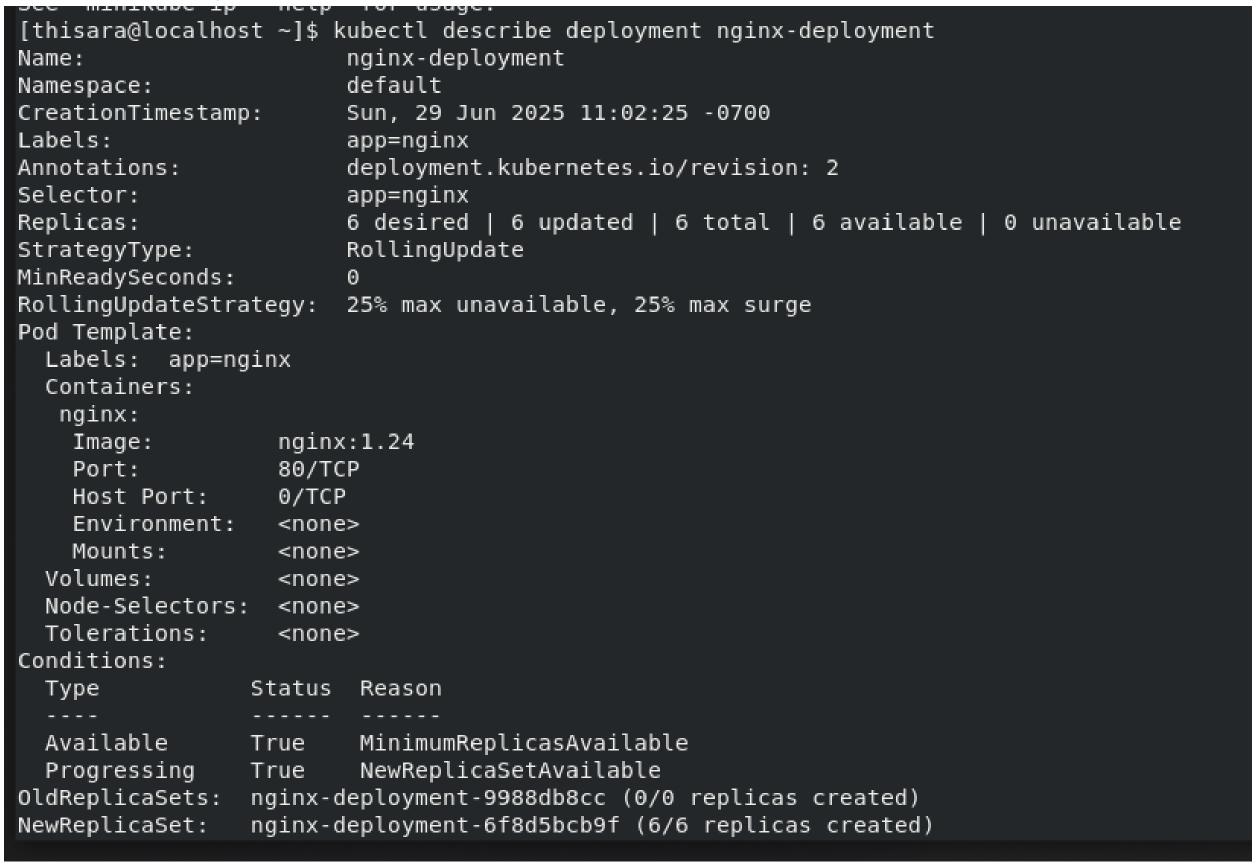
```
  GNU nano 2.9.8                                         demo.yml

apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 6
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.24
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 80
```

2. Check if new changes work or not



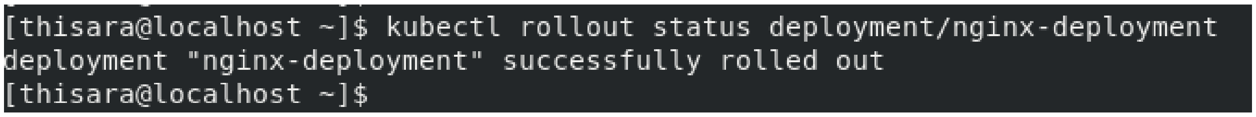
```
[thisara@localhost ~]$ kubectl get pods --watch
NAME                               READY   STATUS    RESTARTS   AGE
kubia-5d5bf68c-mgffc               1/1     Running   0          6d2h
nginx-deployment-6f8d5bcb9f-2jtrq  1/1     Running   0          27s
nginx-deployment-6f8d5bcb9f-2ttvt  1/1     Running   0          26s
nginx-deployment-6f8d5bcb9f-4p5ts  1/1     Running   0          83s
nginx-deployment-6f8d5bcb9f-j28cz  1/1     Running   0          83s
nginx-deployment-6f8d5bcb9f-ndkvz  1/1     Running   0          82s
nginx-deployment-6f8d5bcb9f-ww626  1/1     Running   0          24s



^C[thisara@localhost ~]$
[thisara@localhost ~]$
[thisara@localhost ~]$
[thisara@localhost ~]$
[thisara@localhost ~]$ kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
kubia              1/1     1            1           6d2h
nginx-deployment   6/6     6            6           10m
[thisara@localhost ~]$
```

3. Further checkings (detailed)

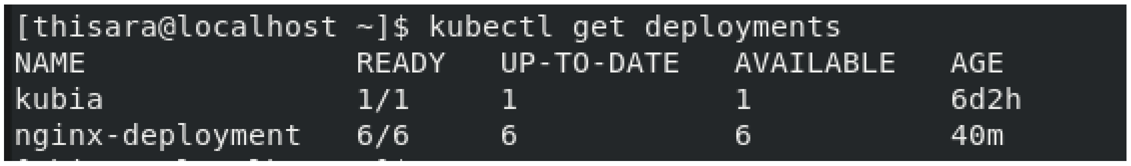


```
[thisara@localhost ~]$ kubectl describe deployment nginx-deployment
Name:                   nginx-deployment
Namespace:              default
CreationTimestamp:      Sun, 29 Jun 2025 11:02:25 -0700
Labels:                 app=nginx
Annotations:            deployment.kubernetes.io/revision: 2
Selector:               app=nginx
Replicas:               6 desired | 6 updated | 6 total | 6 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:         nginx:1.24
    Port:          80/TCP
    Host Port:     0/TCP
    Environment:   <none>
    Mounts:        <none>
  Volumes:         <none>
  Node-Selectors:  <none>
  Tolerations:     <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  nginx-deployment-9988db8cc (0/0 replicas created)
NewReplicaSet:   nginx-deployment-6f8d5bcb9f (6/6 replicas created)
```

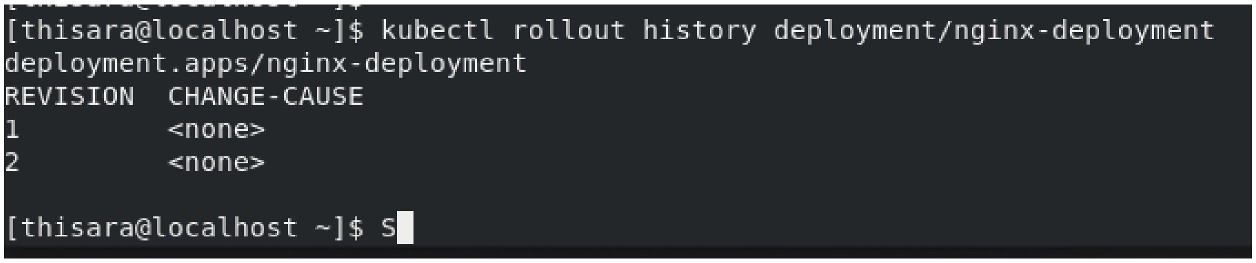
Rollback Deployments



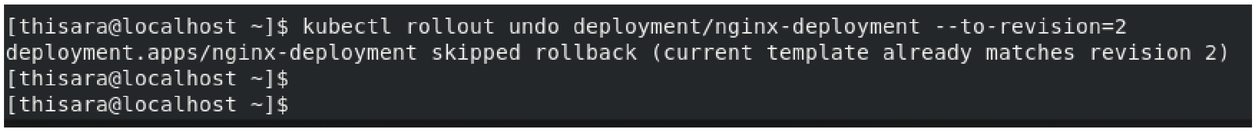
```
[thisara@localhost ~]$ kubectl rollout status deployment/nginx-deployment
deployment "nginx-deployment" successfully rolled out
[thisara@localhost ~]$
```



```
[thisara@localhost ~]$ kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
kubia              1/1     1            1           6d2h
nginx-deployment   6/6     6            6           40m
```

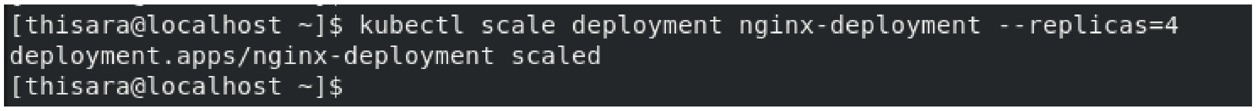

```
[thisara@localhost ~]$ kubectl rollout history deployment/nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         <none>
2         <none>

[thisara@localhost ~]$ S
```

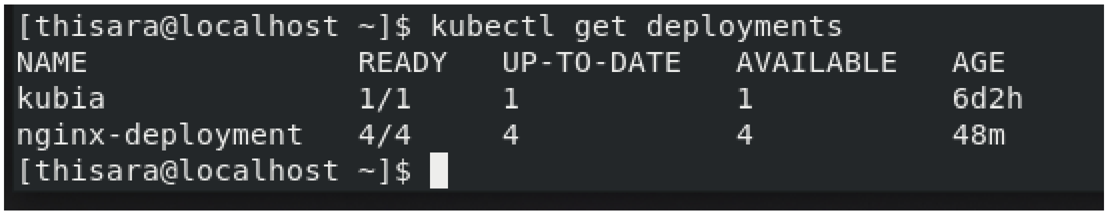
Rolled back to revision 2


```
[thisara@localhost ~]$ kubectl rollout undo deployment/nginx-deployment --to-revision=2
deployment.apps/nginx-deployment skipped rollback (current template already matches revision 2)
[thisara@localhost ~]$
[thisara@localhost ~]$
```

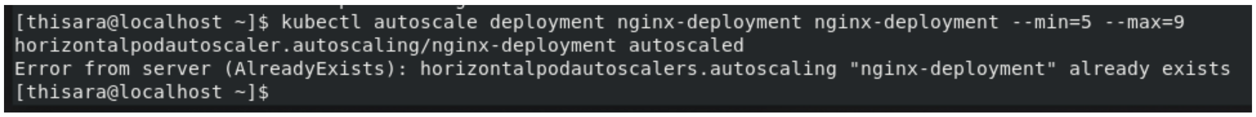
Scale up a Deployment

1. **Scale down to 4 replicas**


```
[thisara@localhost ~]$ kubectl scale deployment nginx-deployment --replicas=4
deployment.apps/nginx-deployment scaled
[thisara@localhost ~]$
```


```
[thisara@localhost ~]$ kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
kubia              1/1     1            1           6d2h
nginx-deployment   4/4     4            4           48m
[thisara@localhost ~]$
```

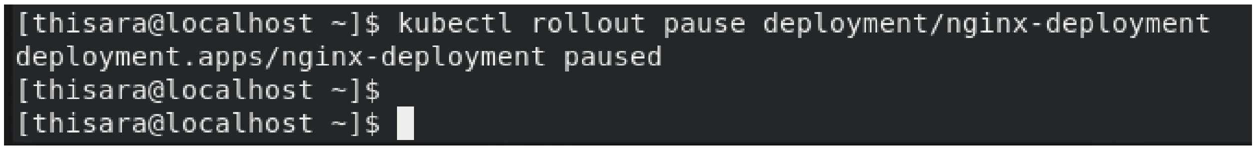
2. **Use Auto Scaling**



```
[thisara@localhost ~]$ kubectl autoscale deployment nginx-deployment nginx-deployment --min=5 --max=9
horizontalpodautoscaler.autoscaling/nginx-deployment autoscaled
Error from server (AlreadyExists): horizontalpodautoscalers.autoscaling "nginx-deployment" already exists
[thisara@localhost ~]$
```

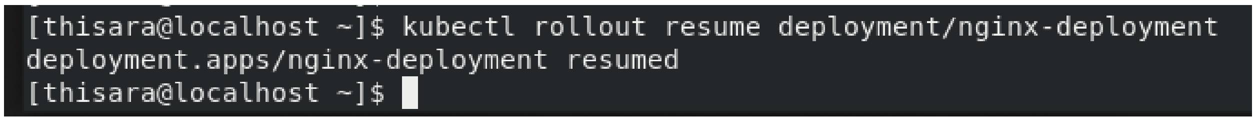
Pause and Resume Deployment

1. **Pause deployment**



```
[thisara@localhost ~]$ kubectl rollout pause deployment/nginx-deployment
deployment.apps/nginx-deployment paused
[thisara@localhost ~]$
[thisara@localhost ~]$
```

2. **Resume Deployment**



```
[thisara@localhost ~]$ kubectl rollout resume deployment/nginx-deployment
deployment.apps/nginx-deployment resumed
[thisara@localhost ~]$
```

Summary of What I Learned

- Gained hands-on experience with Kubernetes fundamentals, including deployments, replica management, and pod lifecycle monitoring using kubectl.
- Learned how to write and apply a Deployment manifest (demo.yml) that defines the application's desired state, including replica count, container image, and exposed ports.
- Practiced using Minikube to simulate a real Kubernetes cluster locally, enabling safe and repeatable testing of cloud-native workflows.
- Successfully exposed the deployed application to the outside world via a NodePort service, and accessed it using the Minikube IP and port in a web browser.
- Understood how to scale applications manually (kubectl scale) and explored autoscaling based on CPU utilization, reflecting real-world elasticity.
- Used advanced rollout controls such as checking deployment history, pausing and resuming rollouts, and undoing changes to manage application updates safely.
- Strengthened problem-solving skills by observing pod status transitions and applying debugging strategies using kubectl describe, pod logs, and cluster insights.
- Overall, this project provided practical exposure to deploying and managing containerized workloads on Kubernetes, forming a strong foundation for working in DevOps environments and with modern infrastructure platforms.