Create deployment with nginx: latest image

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 4
  strategy:
    type: RollingUpdate

  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

The kubectl diff command in Kubernetes is used to preview changes between your local configuration files and the current live state of resources in a cluster—without applying those changes.

Example: The above Deployment had “replicas 4” and “image nginx: latest”

But will create another deployment with same name, with different replicas and image name.

Example: create with “replicas 6” and image name “nginx-gsk” check with below deployment code.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 6
  strategy:
    type: RollingUpdate

  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:gsk
        ports:
        - containerPort: 80
```

Check with diff command

Syntax: kubectl diff -f filename.yaml

Example:

kubectl diff -f test-dep.yaml

OUT-PUT:

```
    app: nginx
  name: nginx-deployment
@@ -15,7 +15,7 @@
  uid: e48fad21-586e-4b79-a5f2-25bcab051a2c
spec:
  progressDeadlineSeconds: 600
-  replicas: 4
+  replicas: 6
  revisionHistoryLimit: 10
  selector:
    matchLabels:
@@ -32,7 +32,7 @@
        app: nginx
    spec:
      containers:
-      - image: nginx:1.27.0
+      - image: nginx:gsk
        imagePullPolicy: IfNotPresent
        name: nginx
        ports:
```

Observe in above out put “images” and “replicas”

Rollout history:

The kubectl rollout history command in Kubernetes is used to **view the revision history** of a resource like a Deployment, DaemonSet, or StatefulSet. It helps you track changes over time and is especially useful when you want to **audit updates** or **roll back** to a previous version.

Example :

Syntax:

kubectl rollout history <resource_type>/<resource_name>

kubectl rollout history deployment nginx-deployment

Output:

```
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         <none>
3         <none>
4         <none>
```

If you want to see the details of a specific revision:

Syntax: kubectl rollout history deployment/deployment-name--revision=number

Example:

kubectl rollout history deployment nginx-deployment --revision=4

Out-put:

```
deployment.apps/nginx-deployment with revision #4
Pod Template:
  Labels:       app=nginx
        pod-template-hash=5bddfff64f
  Containers:
   nginx:
    Image:      nginx:1.27.0
    Port:       80/TCP
    Host Port:  0/TCP
    Environment:        <none>
    Mounts:     <none>
  Volumes:      <none>
  Node-Selectors:       <none>
  Tolerations:  <none>
```

Rollout undo

The kubectl rollout undo command in Kubernetes is used to **roll back a Deployment (or other rollout-capable resources like DaemonSets or StatefulSets)** to a previous revision. This is especially useful when a recent update causes issues and you need to quickly restore a stable version.

What It Does

- Reverts a Deployment to its **previous revision** or a **specific revision**.
- Helps recover from broken updates or failed rollouts.
- Works with resources that support rollout history (e.g., Deployments).

Syntax:

kubectl rollout undo deployment/deployment-name

Example:

kubectl rollout undo deployment nginx-deployment