==================================

Kubernetes Deployment + Service (Best Practice 2025)

==================================

# 1. QUICK ONE-LINERS (perfect for labs, testing, demos)

# Create a Deployment with 3 replicas

kubectl create deployment nginx-deploy --image=nginx:alpine --replicas=3

# Expose it internally (ClusterIP)

kubectl expose deployment nginx-deploy --port=80 --name=nginx-svc

# Expose it externally (NodePort)

kubectl expose deployment nginx-deploy --port=80 --type=NodePort --name=nginx-svc

```
# Expose with LoadBalancer (GKE, EKS,
AKS, DigitalOcean, etc.)
kubectl expose deployment nginx-deploy -
-port=80 --type=LoadBalancer --
name=nginx-lb

# Check everything
kubectl get deploy,rs,pods,svc
kubectl get svc nginx-svc   # note the
NodePort or External IP

# Access (NodePort example)
# http://<any-node-ip>:<node-port>   e.g.,
http://192.168.1.100:31567

# Scale instantly
kubectl scale deployment nginx-deploy --
replicas=10
kubectl scale deployment nginx-deploy --
replicas=2
```

```
#
========================================
================
# BEST PRACTICE: Proper YAML files
(production-ready)
#
========================================
================

# File: deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deploy
  labels:
    app: nginx
spec:
  replicas: 3
```

```
selector:
  matchLabels:
    app: nginx
template:
  metadata:
    labels:
      app: nginx
  spec:
    containers:
    - name: nginx
      image: nginx:alpine
      ports:
      - containerPort: 80
      resources:
        requests:
          memory: "64Mi"
          cpu: "100m"
        limits:
          memory: "128Mi"
          cpu: "200m"

# File: service.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: NodePort              # Use ClusterIP,
NodePort, or LoadBalancer
```

```
# Apply both
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml

# Upgrade image (zero downtime)
kubectl set image deployment/nginx-
deploy nginx=nginx:1.25 --record
kubectl rollout status deployment/nginx-
deploy

# Rollback if something goes wrong
kubectl rollout undo deployment/nginx-
deploy

===================================
ONE-LINE SUPER FAST VERSION (most
people use this)
===================================
kubectl create deployment myapp --
image=nginx:alpine --replicas=4
kubectl expose deployment myapp --
type=NodePort --port=80 --name=myapp-
svc

# Done! Access via the NodePort shown
in:
kubectl get svc myapp-svc

===================================
CLEANUP
===================================

kubectl delete deployment nginx-deploy
kubectl delete service nginx-service
# or
kubectl delete -f deployment.yaml -f
service.yaml
```

==================================

SUMMARY: This is the correct way in 2025

==================================

Deployment = modern, self-healing, rolling updates, rollback

ReplicaSet = created automatically by Deployment

ReplicationController = old/legacy, don't use

Pod alone = no scaling or healing

Use Deployment + Service → always!

Done! You now have a production-grade, scalable, updatable Nginx app running in Kubernetes.