# Kubernetes Lab Manual: Deployments, Scaling, Resource Limits & CronJob (20 min)

**Goal:** In ~20 minutes, you will deploy and scale an nginx application, add resource requests/limits, and run a CronJob that prints the date every minute.

## 0) Prerequisites & Dependencies

* **Kubernetes cluster** (any): kind, Minikube, Docker Desktop, k3s, or a managed cluster.
* **kubectl** installed and configured to talk to the cluster.
* **Cluster access:** you can create namespaces and workloads.
* **Optional (for resource usage):** metrics-server installed (kubectl top pods). Not required for this lab.

**Verify connectivity**

*kubectl version --short  
kubectl get nodes*

Expect at least one **Ready** node.

## 1) Create a dedicated namespace

Create a clean workspace for this lab.

*kubectl create namespace lab-wm  
kubectl config set-context --current --namespace=lab-wm*

Tip: If you prefer not to change context, append -n lab-wm to each command.

## 2) Create a Deployment (nginx) with 3 replicas

We’ll use a lightweight image and expose port 80. Save the following as **deployment-nginx.yaml**.

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: nginx-deploy  
spec:  
 replicas: 3  
 selector:  
 matchLabels:  
 app: nginx  
 template:  
 metadata:  
 labels:  
 app: nginx  
 spec:  
 containers:  
 - name: nginx  
 image: nginx:1.25-alpine  
 ports:  
 - containerPort: 80

Apply and verify:

kubectl apply -f deployment-nginx.yaml  
kubectl get deploy,pods -o wide  
kubectl rollout status deployment/nginx-deploy

You should see **3/3** replicas available.

**(Optional) Create a ClusterIP Service** to test quickly within the cluster. Save as **svc-nginx.yaml** and apply.

apiVersion: v1  
kind: Service  
metadata:  
 name: nginx-svc  
spec:  
 selector:  
 app: nginx  
 ports:  
 - port: 80  
 targetPort: 80

kubectl apply -f svc-nginx.yaml  
kubectl get svc

## 3) Scale the Deployment to 5 replicas

Scale imperatively and confirm:

kubectl scale deployment nginx-deploy --replicas=5  
kubectl get deploy,pods  
kubectl describe deploy nginx-deploy | sed -n '/Replicas:/p'

Expected: Deployment shows **5** desired and **5 available** once rollout completes.

## 4) Add resource requests & limits

Two common approaches are shown below. Use either **A** (patch) or **B** (re-apply YAML).

### A) Patch existing Deployment

kubectl set resources deployment/nginx-deploy \  
 --containers=nginx \  
 --requests=cpu=100m,memory=128Mi \  
 --limits=cpu=250m,memory=256Mi  
  
kubectl rollout status deployment/nginx-deploy

Verify on a pod:

POD=$(kubectl get pods -l app=nginx -o jsonpath='{.items[0].metadata.name}')  
kubectl get pod "$POD" -o jsonpath='{.spec.containers[0].resources}' && echo

### B) Redeploy with resources in YAML

Update **deployment-nginx.yaml** with the resources block and re-apply.

apiVersion: apps/v1  
kind: Deployment  
metadata:  
 name: nginx-deploy  
spec:  
 replicas: 5 # keep current scale  
 selector:  
 matchLabels:  
 app: nginx  
 template:  
 metadata:  
 labels:  
 app: nginx  
 spec:  
 containers:  
 - name: nginx  
 image: nginx:1.25-alpine  
 ports:  
 - containerPort: 80  
 resources:  
 requests:  
 cpu: 100m  
 memory: 128Mi  
 limits:  
 cpu: 250m  
 memory: 256Mi

kubectl apply -f deployment-nginx.yaml  
kubectl rollout status deployment/nginx-deploy

## 5) Run a CronJob (date command every minute)

We’ll run a tiny container that prints the date each minute. Save as **cronjob-date.yaml**.

apiVersion: batch/v1  
kind: CronJob  
metadata:  
 name: print-date  
spec:  
 schedule: "\* \* \* \* \*" # every minute  
 concurrencyPolicy: Forbid  
 successfulJobsHistoryLimit: 3  
 failedJobsHistoryLimit: 1  
 jobTemplate:  
 spec:  
 ttlSecondsAfterFinished: 120 # auto-cleanup finished Jobs after 2 minutes (if feature is enabled)  
 template:  
 spec:  
 restartPolicy: OnFailure  
 containers:  
 - name: clock  
 image: busybox:1.36  
 command: ["/bin/sh","-c","date; echo Hello from Kubernetes CronJob"]

Apply and observe executions:

kubectl apply -f cronjob-date.yaml  
kubectl get cronjob  
  
# Wait 60–90 seconds for the first run  
kubectl get jobs --watch

When a Job appears, check its Pod and logs:

kubectl get pods -l job-name=$(kubectl get jobs -o jsonpath='{.items[-1:].0.metadata.name}')  
POD=$(kubectl get pods -l job-name=$(kubectl get jobs -o jsonpath='{.items[-1:].0.metadata.name}') -o jsonpath='{.items[0].metadata.name}')  
kubectl logs "$POD"

You should see a timestamp and the greeting message.

## 6) Quick Validation Checklist

* **Deployment created** with nginx and initially **3 replicas**.
* **Scaled** to **5 replicas**; rollout completed successfully.
* **Resource requests/limits** applied to nginx container.
* **CronJob** created; Jobs running every minute and Pods show date output in logs.

## 7) Troubleshooting Tips

* **ImagePullBackOff:** Check network and image tag; try nginx:stable-alpine or pull-able tag.
* **Pods Pending:** Node resource pressure or scheduling constraints. Reduce requests/limits or free node resources.
* **CrashLoopBackOff:** Check kubectl logs and kubectl describe pod. Verify container command and image.
* **No Jobs appearing:** Ensure the CronJob schedule is correct and controller is running. Confirm cluster time (CronJobs follow controller time).
* **Cannot change namespace:** Use -n lab-wm on commands instead of changing context.

## 8) Cleanup (optional)

kubectl delete -f cronjob-date.yaml  
kubectl delete -f svc-nginx.yaml # if created  
kubectl delete -f deployment-nginx.yaml  
kubectl delete namespace lab-wm

## 9) Instructor Notes (optional)

* Time-box: ~5 min Deployment, ~3 min Scaling, ~5 min Resources, ~5–7 min CronJob & verification.
* Common checks: kubectl rollout status, kubectl describe for events, and kubectl logs for the CronJob.
* If using Minikube: minikube kubectl -- get nodes or set up shell alias; minikube service nginx-svc can open a browser test.

**End of Lab ✅**