CKAD Practice Solutions – Part 1 (Q1–Q60)

Concise kubectl commands and minimal YAML where needed. Adapt namespaces/images to your cluster.

Q1. Create a pod nginx-pod running nginx:1.19; restart on failure.

```
kubectl run nginx-pod --image=nginx:1.19 --restart=OnFailure
```

Q2. Logs of backend-pod in dev; last 20 lines.

```
kubectl logs -n dev backend-pod --tail=20
```

Q3. Expose deployment myapp on 8080 as myapp-svc (ClusterIP).

```
kubectl expose deploy myapp --port=8080 --target-port=8080 --name=myapp-svc --
type=ClusterIP
```

Q4. Create ConfigMap app-config with APP_ENV=production, APP_DEBUG=false.

```
kubectl create configmap app-config --from-literal=APP_ENV=production --from-
literal=APP_DEBUG=false
```

Q5. Mount ConfigMap as env vars in a pod.

```
cat <<'YAML' | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata: {name: cm-env-pod}
spec:
  containers:
  - name: app
    image: nginx
    envFrom:
    - configMapRef: {name: app-config}
YAML
```

Q6. Create Secret db-secret with username=admin, password=MyP@ssw0rd.

```
kubectl create secret generic db-secret --from-literal=username=admin --from-
literal=password='MyP@ssw0rd'
```

Q7. Use db-secret as env in deployment api-deploy.

```
kubectl set env deployment/api-deploy --from=secret/db-secret
```

Q8. Create pod sidecar-pod with app nginx and sidecar tailing nginx access log.

```
cat <<'YAML' | kubectl apply -f -
apiVersion: v1
kind: Pod
```

```
metadata: {name: sidecar-pod}
spec:
  volumes:
  - name: logs
    emptyDir: {}
  containers:
  - name: app
    image: nginx
    volumeMounts:
    - {name: logs, mountPath: /var/log/nginx}
  - name: sidecar
    image: busybox
    command: ['sh','-c','tail -F /var/log/nginx/access.log']
    volumeMounts:
    - {name: logs, mountPath: /var/log/nginx}
YAML
```

Q9. Share a volume for logs between both containers (see Q8).

```
Already shown in Q8 via emptyDir volume 'logs' and two volumeMounts.
```

Q10. Check CPU/memory usage of pods in default.

```
kubectl top pods -n default
```

Q11. Describe orders-pod and filter only events.

```
kubectl describe pod orders-pod | sed -n '/Events:/,$p'
```

Q12. Troubleshoot CrashLoopBackOff.

```
kubectl logs <pod> -p
kubectl describe pod <pod>
kubectl get events --sort-by=.lastTimestamp
kubectl exec -it <pod> -- /bin/sh (if possible)
kubectl get pod <pod> -o yaml | grep -A5 -e image -e command
```

Q13. Create deployment frontend with 3 replicas nginx:1.20.

```
kubectl create deploy frontend --image=nginx:1.20 --replicas=3
```

Q14. Rolling update frontend to nginx:1.21.

```
kubectl set image deploy/frontend nginx=nginx:1.21 && kubectl rollout status
deploy/frontend
```

Q15. Rollback frontend to previous version.

```
kubectl rollout undo deploy/frontend
```

Q16. Create job batch-job echo Hello CKAD once.

```
kubectl create job batch-job --image=busybox -- /bin/sh -c 'echo Hello CKAD'
```

Q17. Create cronjob daily-job run 1 AM daily printing Backup started.

```
kubectl create cronjob daily-job --image=busybox --schedule='0 1 * * *' -- /bin/sh
-c 'echo Backup started'
```

Q18. Expose frontend as NodePort 30080.

```
kubectl expose deploy frontend --type=NodePort --port=80 --target-port=80 --
name=frontend-np --
overrides='{"spec":{"ports":[{"port":80,"targetPort":80,"nodePort":30080}]}}'
```

Q19. Create Ingress routing /app1→app1-svc:8080, /app2→app2-svc:9090.

```
cat <<'YAML' | kubectl apply -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata: {name: apps-ing}
spec:
  rules:
  - http:
      paths:
      - path: /app1
        pathType: Prefix
        backend: {service: {name: app1-svc, port: {number: 8080}}}
      - path: /app2
        pathType: Prefix
        backend: {service: {name: app2-svc, port: {number: 9090}}}
YAML
```

Q20. Verify DNS of backend-svc inside a pod.

```
kubectl exec -it <pod> -- nslookup backend-svc
```

Q21. Create PV 1Gi hostPath /mnt/data.

```
cat <<'YAML' | kubectl apply -f -
apiVersion: v1
kind: PersistentVolume
metadata: {name: pv1}
spec:
  capacity: {storage: 1Gi}
  accessModes: [ReadWriteOnce]
  hostPath: {path: /mnt/data}
  persistentVolumeReclaimPolicy: Retain
YAML
```

Q22. Create PVC app-pvc 500Mi.

```
cat <<'YAML' | kubectl apply -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata: {name: app-pvc}
spec:
  accessModes: [ReadWriteOnce]
  resources: {requests: {storage: 500Mi}}
YAML
```

Q23. Mount PVC into db-pod at /data/db.

```
cat <<'YAML' | kubectl apply -f -
apiVersion: v1
kind: Pod
```

```
metadata: {name: db-pod}
spec:
  volumes:
  - name: data
    persistentVolumeClaim: {claimName: app-pvc}
  containers:
  - name: db
    image: busybox
    command: ['sh','-c','sleep 3600']
    volumeMounts:
    - {name: data, mountPath: /data/db}
YAML
```

Q24. Create ServiceAccount deploy-sa.

```
kubectl create sa deploy-sa
```

Q25. Create pod sa-pod using deploy-sa.

```
cat <<'YAML' | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata: {name: sa-pod}
spec:
  serviceAccountName: deploy-sa
  containers:
  - name: c
    image: nginx
YAML
```

Q26. Create Role pod-reader in dev allowing get,list,watch on pods.

```
kubectl -n dev apply -f - <<'YAML'
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata: {name: pod-reader}
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get","list","watch"]
YAML
```

Q27. Bind Role pod-reader to user developer.

```
kubectl -n dev apply -f - <<'YAML'
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata: {name: pod-reader-binding}
subjects:
- kind: User
  name: developer
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: pod-reader
YAML
```

Q28. Pause and resume rollout of frontend.

```
kubectl rollout pause deploy/frontend
kubectl rollout resume deploy/frontend
```

Q29. Drain node worker-1 safely.

```
kubectl drain worker-1 --ignore-daemonsets --delete-emptydir-data
```

Q30. Scale frontend to 5 replicas.

```
kubectl scale deploy/frontend --replicas=5
```

Q31. Pod multiport-pod exposing 80 and 443.

```
kubectl run multiport-pod --image=nginx --port=80 --expose=false --
overrides='{"apiVersion":"v1","kind":"Pod","spec":{"containers":[{"name":"nginx","
image":"nginx","ports":[{"containerPort":80},{"containerPort":443}]}]}}' -o yaml -
-dry-run=client | kubectl apply -f -
```

Q32. busybox-pod sleep 3600.

```
kubectl run busybox-pod --image=busybox -- /bin/sh -c 'sleep 3600'
```

Q33. List all pods across namespaces with nodes.

```
kubectl get pods -A -o wide
```

Q34. ConfigMap game-config from game.properties.

```
kubectl create configmap game-config --from-file=game.properties
```

Q35. Mount game-config at /etc/game.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: game-pod}
spec:
  volumes:
  - name: game
    configMap: {name: game-config}
  containers:
  - name: c
    image: busybox
    command: ['sh','-c','sleep 3600']
    volumeMounts:
    - {name: game, mountPath: /etc/game}
YAML
```

Q36. Encode mysecuretoken and store in Secret api-token.

```
echo -n 'mysecuretoken' | base64
kubectl create secret generic api-token --from-literal=token=mysecuretoken
```

Q37. Inject api-token as files under /etc/secrets.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: secret-pod}
spec:
  volumes:
  - name: s
    secret: {secretName: api-token}
  containers:
  - name: c
    image: busybox
    command: ['sh','-c','sleep 3600']
    volumeMounts:
    - {name: s, mountPath: /etc/secrets, readOnly: true}
YAML
```

Q38. logging-pod with app+logger copying logs every 30s.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: logging-pod}
spec:
  volumes:
  - name: logs
    emptyDir: {}
  containers:
  - name: app
    image: nginx
    volumeMounts: [{name: logs, mountPath: /var/log/nginx}]
  - name: logger
    image: busybox
    command: ['sh','-c','while true; do cp /var/log/nginx/* /logs/; sleep 30;
done']
    volumeMounts: [{name: logs, mountPath: /logs}]
YAML
```

Q39. Use emptyDir to share data (see Q38).

```
Shown in Q38 via emptyDir volume 'logs'.
```

Q40. Get YAML of web-pod and save to /tmp/web.yaml.

```
kubectl get pod web-pod -o yaml > /tmp/web.yaml
```

Q41. Show last restart reason of cache-pod.

```
kubectl get pod cache-pod -o
jsonpath='{.status.containerStatuses[0].lastState.terminated.reason}{"\n"}'
```

Q42. Watch real-time logs of deployment api-deploy.

```
kubectl logs deploy/api-deploy -f --all-containers
```

Q43. Create ReplicaSet redis-rs with 2 replicas redis:6.0.

```
kubectl apply -f - <<'YAML'
apiVersion: apps/v1
kind: ReplicaSet
metadata: {name: redis-rs}
spec:
  replicas: 2
  selector: {matchLabels: {app: redis}}
  template:
    metadata: {labels: {app: redis}}
    spec:
      containers:
      - name: redis
        image: redis:6.0
YAML
```

Q44. Scale ReplicaSet redis-rs to 4.

```
kubectl scale rs/redis-rs --replicas=4
```

Q45. Convert ReplicaSet to Deployment redis-deploy.

```
kubectl create deploy redis-deploy --image=redis:6.0 --dry-run=client -o yaml |
kubectl apply -f - ; kubectl delete rs redis-rs
```

Q46. Job math-job runs expr 3 + 7.

```
kubectl create job math-job --image=busybox -- /bin/sh -c 'expr 3 + 7'
```

Q47. CronJob weekly-report midnight Sunday.

```
kubectl create cronjob weekly-report --image=busybox --schedule='0 0 * * 0' --
/bin/sh -c 'echo report'
```

Q48. ClusterIP redis-svc for redis-deploy on 6379.

```
kubectl expose deploy redis-deploy --port=6379 --name=redis-svc --type=ClusterIP
```

Q49. Headless Service for StatefulSet zk.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Service
metadata: {name: zk}
spec:
  clusterIP: None
  selector: {app: zk}
  ports: [{port: 2181}]
YAML
```

Q50. Verify service endpoints of redis-svc.

```
kubectl get endpoints redis-svc -o wide
```

Q51. PV 2Gi storageClass manual hostPath /data/pv.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
```

```
kind: PersistentVolume
metadata: {name: pv-manual}
spec:
  capacity: {storage: 2Gi}
  accessModes: [ReadWriteOnce]
  storageClassName: manual
  hostPath: {path: /data/pv}
  persistentVolumeReclaimPolicy: Retain
YAML
```

Q52. PVC redis-pvc 1Gi storageClass manual.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: PersistentVolumeClaim
metadata: {name: redis-pvc}
spec:
  accessModes: [ReadWriteOnce]
  storageClassName: manual
  resources: {requests: {storage: 1Gi}}
YAML
```

Q53. StatefulSet redis with 2 replicas using PVC redis-pvc (template).

```
kubectl apply -f - <<'YAML'
apiVersion: apps/v1
kind: StatefulSet
metadata: {name: redis}
spec:
  serviceName: redis
  replicas: 2
  selector: {matchLabels: {app: redis}}
  template:
    metadata: {labels: {app: redis}}
    spec:
      containers:
      - name: redis
        image: redis:6.0
        volumeMounts: [{name: data, mountPath: /data}]
  volumeClaimTemplates:
  - metadata: {name: data}
    spec:
      accessModes: [ReadWriteOnce]
      resources: {requests: {storage: 1Gi}}
      storageClassName: manual
YAML
```

Q54. Create PodSecurityPolicy allowing privileged (deprecated; use PSP API).

```
kubectl apply -f - <<'YAML'
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata: {name: privileged-psp}
spec:
  privileged: true
  runAsUser: {rule: RunAsAny}
  seLinux: {rule: RunAsAny}
  fsGroup: {rule: RunAsAny}
  supplementalGroups: {rule: RunAsAny}
  volumes: ["*"]
YAML
```

Q55. NetworkPolicy deny-all ingress in dev.

```
kubectl -n dev apply -f - <<'YAML'
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata: {name: deny-all}
spec:
  podSelector: {}
  policyTypes: ["Ingress"]
YAML
```

Q56. Allow ingress from pods with label role=frontend.

```
kubectl -n dev apply -f - <<'YAML'
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata: {name: allow-frontend-to-all}
spec:
  podSelector: {}
  policyTypes: ["Ingress"]
  ingress:
  - from:
    - podSelector: {matchLabels: {role: frontend}}
YAML
```

Q57. ClusterRole ns-admin full access to ConfigMaps and Secrets.

```
kubectl apply -f - <<'YAML'
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata: {name: ns-admin}
rules:
- apiGroups: [""]
  resources: ["configmaps","secrets"]
  verbs: ["*"]
YAML
```

Q58. Bind ns-admin to user ops-team.

```
kubectl apply -f - <<'YAML'
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata: {name: ns-admin-binding}
subjects:
- kind: User
  name: ops-team
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: ns-admin
YAML
```

Q59. Rolling restart redis-deploy.

```
kubectl rollout restart deploy/redis-deploy
```

Q60. Delete temp-pod without new replica.

```
kubectl delete pod temp-pod --force --grace-period=0  # If managed, scale
rs/deploy first to prevent recreation
```

CKAD Practice Solutions – Part 2 (Q61–Q120)

Concise kubectl commands and minimal YAML. Adjust namespaces, labels, and storage classes to your cluster.

Q61. Init container echoes before nginx starts.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: init-pod}
spec:
  initContainers:
  - name: init
    image: busybox
    command: ['sh','-c','echo Init Done']
  containers:
  - name: web
    image: nginx
YAML
```

Q62. Mount host /var/log at /logs.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: hostpath-pod}
spec:
  volumes:
  - name: hostlogs
    hostPath: {path: /var/log}
  containers:
  - name: c
    image: busybox
    command: ['sh','-c','sleep 3600']
    volumeMounts: [{name: hostlogs, mountPath: /logs}]
YAML
```

Q63. Schedule pod on worker-2 via nodeSelector.

```
kubectl label node worker-2 disk=ssd  # example label (if needed)
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: node-pod}
spec:
  nodeSelector: {kubernetes.io/hostname: worker-2}
  containers:
  - name: c
    image: nginx
YAML
```

Q64. ConfigMap feature-config and mount at /etc/config/feature.

```
kubectl create configmap feature-config --from-literal=FEATURE_X=true
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: feature-pod}
spec:
  volumes:
  - name: cfg
    configMap: {name: feature-config}
  containers:
  - name: c
    image: busybox
    command: ['sh','-c','sleep 3600']
    volumeMounts: [{name: cfg, mountPath: /etc/config/feature}]
YAML
```

Q65. Downward API: inject pod name to POD_NAME.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: downward-name}
spec:
  containers:
  - name: c
    image: busybox
    command: ['sh','-c','env; sleep 3600']
    env:
    - name: POD_NAME
      valueFrom:
        fieldRef: {fieldPath: metadata.name}
YAML
```

Q66. Expose container's CPU limit as env CPU_LIMIT via Downward API.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: cpu-limit-downward}
```

```
spec:
  containers:
  - name: app
    image: busybox
    command: ['sh','-c','echo $CPU_LIMIT; sleep 3600']
    resources:
      limits: {cpu: "500m"}
    env:
    - name: CPU_LIMIT
      valueFrom:
        resourceFieldRef:
          containerName: app
          resource: limits.cpu
YAML
```

Q67. TLS secret tls-secret from tls.crt/tls.key and mount at /etc/tls.

```
kubectl create secret tls tls-secret --cert=tls.crt --key=tls.key
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: tls-pod}
spec:
  volumes:
  - name: tls
    secret: {secretName: tls-secret}
  containers:
  - name: web
    image: nginx
    volumeMounts: [{name: tls, mountPath: /etc/tls, readOnly: true}]
YAML
```

Q68. Sidecar adapter transforms logs to shared volume.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: adapter-pod}
spec:
  volumes:
  - name: shared
    emptyDir: {}
  containers:
  - name: api
    image: myapi:v1
    volumeMounts: [{name: shared, mountPath: /adapter}]
  - name: adapter
    image: busybox
    command: ['sh','-c','while true; do cat /adapter/app.log | sed s/error/ERR/g >
/adapter/app.transformed; sleep 5; done']
    volumeMounts: [{name: shared, mountPath: /adapter}]
YAML
```

Q69. Add livenessProbe on /healthz:8080 every 5s.

```
kubectl patch pod adapter-pod --type='merge' -p '
spec:
  containers:
  - name: api
    livenessProbe:
      httpGet: {path: /healthz, port: 8080}
      periodSeconds: 5
'
```

Q70. Add readinessProbe /ready with initialDelay 10s.

```
kubectl patch pod adapter-pod --type='merge' -p '
spec:
  containers:
  - name: api
    readinessProbe:
      httpGet: {path: /ready, port: 8080}
      initialDelaySeconds: 10
'
```

Q71. Find pods in dev restarted >1 time.

```
kubectl get pods -n dev --no-headers | awk '$4 > 1 {print $1, $4}'
```

Q72. Show events in test sorted by time.

```
kubectl get events -n test --sort-by=.lastTimestamp  # or: --sort-
by=.metadata.creationTimestamp on newer clusters
```

Q73. Save logs from pods with label tier=backend to file.

```
kubectl logs -l tier=backend --all-containers --tail=-1 > /tmp/backend-logs.txt
```

Q74. Deployment blue-app (nginx:1.19, 2 replicas).

```
kubectl create deploy blue-app --image=nginx:1.19 --replicas=2
```

Q75. Deployment green-app (nginx:1.20, 2 replicas).

```
kubectl create deploy green-app --image=nginx:1.20 --replicas=2
```

Q76. Blue-green switch: point service app-svc to green-app.

```
kubectl patch svc app-svc -p '{"spec":{"selector":{"app":"green-app"}}}'
```

Q77. DaemonSet log-agent on all nodes (fluentd).

```
kubectl apply -f - <<'YAML'
apiVersion: apps/v1
kind: DaemonSet
metadata: {name: log-agent}
spec:
  selector: {matchLabels: {app: log-agent}}
  template:
    metadata: {labels: {app: log-agent}}
    spec:
      containers:
```

```
      - name: fluentd
        image: fluentd:latest
YAML
```

Q78. Job parallel-job with 3 completions, max 2 parallel.

```
kubectl create job parallel-job --image=busybox --parallel=2 --completions=3 --
/bin/sh -c 'echo hi && sleep 2'
```

Q79. Headless service for StatefulSet db on 5432.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Service
metadata: {name: db}
spec:
  clusterIP: None
  selector: {app: db}
  ports: [{port: 5432, targetPort: 5432}]
YAML
```

Q80. Ingress TLS using tls-secret; /api → api-svc:8080.

```
kubectl apply -f - <<'YAML'
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata: {name: api-ing}
spec:
  tls:
  - hosts: [example.com]
    secretName: tls-secret
  rules:
  - host: example.com
    http:
      paths:
      - path: /api
        pathType: Prefix
        backend: {service: {name: api-svc, port: {number: 8080}}}
YAML
```

Q81. Restrict api-svc to only be reachable within namespace dev.

```
# Use a NetworkPolicy to only allow same-namespace traffic.
# This selects pods behind api-svc by label (app=api) and only allows ingress from
namespace 'dev'.
kubectl -n dev apply -f - <<'YAML'
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata: {name: api-namespace-only}
spec:
  podSelector: {matchLabels: {app: api}}
  policyTypes: ["Ingress"]
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: dev
YAML
```

Q82. StatefulSet mysql with 3 replicas and per-pod PVC at /var/lib/mysql.

```
kubectl apply -f - <<'YAML'
apiVersion: apps/v1
kind: StatefulSet
metadata: {name: mysql}
spec:
  serviceName: mysql
  replicas: 3
  selector: {matchLabels: {app: mysql}}
  template:
    metadata: {labels: {app: mysql}}
    spec:
      containers:
      - name: mysql
        image: mysql:8
        env:
        - name: MYSQL_ALLOW_EMPTY_PASSWORD
          value: "yes"
        volumeMounts: [{name: data, mountPath: /var/lib/mysql}]
  volumeClaimTemplates:
  - metadata: {name: data}
    spec:
      accessModes: [ReadWriteOnce]
      resources: {requests: {storage: 5Gi}}
YAML
```

Q83. Expand PVC mysql-pvc from 5Gi to 10Gi and pick up change.

```
# StorageClass must allow volume expansion. Then:
kubectl patch pvc mysql-pvc -p
'{"spec":{"resources":{"requests":{"storage":"10Gi"}}}}'
# For some filesystems, restart the pod/statefulset pod if needed:
kubectl delete pod <pod-using-mysql-pvc> --grace-period=0 --force  # if safe
# Or rollout restart the workload:
kubectl rollout restart statefulset/mysql
```

Q84. Backup data from PVC app-pvc using tar to /backup.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: pvc-backup}
spec:
  restartPolicy: Never
  volumes:
  - name: src
    persistentVolumeClaim: {claimName: app-pvc}
  - name: out
    emptyDir: {}
  containers:
  - name: bb
    image: busybox
    command: ['sh','-c','tar czf /backup/app.tgz -C /data .']
    volumeMounts:
    - {name: src, mountPath: /data}
    - {name: out, mountPath: /backup}
YAML
# Retrieve archive from the pod:
kubectl cp pvc-backup:/backup/app.tgz ./app.tgz
```

Q85. Pod restricted-pod with runAsNonRoot: true.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: restricted-pod}
spec:
  containers:
  - name: web
    image: nginx
    securityContext:
      runAsNonRoot: true
YAML
```

Q86. PodSecurityContext: all containers run as user 1001.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: runas-1001}
spec:
  securityContext:
    runAsUser: 1001
    runAsGroup: 1001
    fsGroup: 1001
  containers:
  - name: c
    image: nginx
YAML
```

Q87. Role secret-reader (get secrets) in prod; bind to user auditor.

```
kubectl -n prod apply -f - <<'YAML'
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata: {name: secret-reader}
rules:
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["get"]
YAML
kubectl -n prod apply -f - <<'YAML'
apiVersion: rbac.authorization.k8s.io/v1
```

```
kind: RoleBinding
metadata: {name: secret-reader-bind}
subjects:
- kind: User
  name: auditor
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: secret-reader
YAML
```

Q88. Deny all egress from finance except DNS (UDP 53 to kube-dns).

```
# Allow only DNS to kube-dns in kube-system; otherwise deny all egress.
kubectl -n finance apply -f - <<'YAML'
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata: {name: allow-dns-deny-rest}
spec:
  podSelector: {}
  policyTypes: ["Egress"]
  egress:
  - to:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: kube-system
      podSelector:
        matchLabels:
          k8s-app: kube-dns
    ports:
    - protocol: UDP
      port: 53
YAML
```

Q89. Canary: frontend-canary 1 replica alongside frontend; route 20% via Ingress.

```
# Create canary Deployment
kubectl create deploy frontend-canary --image=nginx --replicas=1
# Assuming NGINX Ingress Controller; use canary annotations.
kubectl apply -f - <<'YAML'
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: frontend
  annotations:
    nginx.ingress.kubernetes.io/canary: "true"
    nginx.ingress.kubernetes.io/canary-weight: "20"
spec:
  rules:
  - host: example.com
    http:
      paths:
      - path: /
```

```
        pathType: Prefix
        backend: {service: {name: frontend-canary, port: {number: 80}}}
YAML
# Base (non-canary) Ingress should point to the main frontend service with 80%
traffic.
```

Q90. Evict all pods from worker-3 immediately (no graceful drain).

```
kubectl drain worker-3 --ignore-daemonsets --delete-emptydir-data --force --grace-
period=0
```

Q91. PriorityClass high-priority and pod nginx-priority that uses it.

```
kubectl apply -f - <<'YAML'
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata: {name: high-priority}
value: 100000
preemptionPolicy: PreemptLowerPriority
globalDefault: false
description: High priority for critical pods
YAML
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: nginx-priority}
spec:
  priorityClassName: high-priority
  containers:
  - name: web
    image: nginx
YAML
```

Q92. Pod tolerates node taint key=dedicated:NoSchedule.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: toleration-pod}
spec:
  tolerations:
  - key: "dedicated"
    operator: "Exists"
    effect: "NoSchedule"
  containers:
  - name: c
    image: nginx
YAML
```

Q93. Force-delete terminating stuck-pod.

```
kubectl delete pod stuck-pod --grace-period=0 --force
```

Q94. List pods NOT Running in prod.

```
kubectl get pods -n prod --field-selector=status.phase!=Running
```

Q95. Run interactive busybox pod debugger.

```
kubectl run -it debugger --image=busybox --restart=Never -- sh
```

Q96. ConfigMap from .env and inject into web-deploy.

```
kubectl create configmap app-env --from-env-file=.env
kubectl set env deploy/web-deploy --from=configmap/app-env
```

Q97. Secret db-conn and mount as env var in backend-pod.

```
kubectl create secret generic db-conn --from-
literal=connection='jdbc:mysql://mysql:3306/db'
kubectl set env pod/backend-pod --from=secret/db-conn
```

Q98. Downward API: expose namespace and pod IP as env vars.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: downward-pod}
spec:
  containers:
  - name: c
    image: busybox
    command: ['sh','-c','env; sleep 3600']
    env:
    - name: MY_NAMESPACE
      valueFrom: {fieldRef: {fieldPath: metadata.namespace}}
    - name: POD_IP
      valueFrom: {fieldRef: {fieldPath: status.podIP}}
YAML
```

Q99. Patch web-deploy to image nginx:1.23 without editing YAML.

```
kubectl set image deploy/web-deploy nginx=nginx:1.23
```

Q100. Generate YAML for quick-deploy (2 replicas) without creating.

```
kubectl create deploy quick-deploy --image=nginx --replicas=2 -o yaml --dry-
run=client
```

Q101. Multi-container monitoring-pod with emptyDir.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: monitoring-pod}
spec:
  volumes:
  - name: shared
    emptyDir: {}
  containers:
  - name: app
    image: nginx
    volumeMounts: [{name: shared, mountPath: /var/log/app}]
  - name: metrics
    image: busybox
    command: ['sh','-c','while true; do wget -qO- http://localhost/status || true; sleep 15; done']
    volumeMounts: [{name: shared, mountPath: /var/log/app}]
YAML
```

Q102. Sidecar log-shipper forwarding /var/log/app externally.

```
# Example using netcat; replace HOST and PORT with your endpoint.
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: shipper-pod}
spec:
  volumes:
  - name: logs
    emptyDir: {}
  containers:
  - name: app
    image: nginx
    volumeMounts: [{name: logs, mountPath: /var/log/app}]
  - name: log-shipper
    image: busybox
    command: ['sh','-c','tail -F /var/log/app/app.log | nc HOST PORT']
    volumeMounts: [{name: logs, mountPath: /var/log/app}]
YAML
```

Q103. postStart echo 'Container started'.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: poststart-pod}
spec:
  containers:
  - name: c
    image: busybox
    command: ['sh','-c','sleep 3600']
    lifecycle:
      postStart:
        exec: {command: ['sh','-c','echo Container started']}
YAML
```

Q104. preStop sleep 10.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: prestop-pod}
spec:
  terminationGracePeriodSeconds: 30
  containers:
  - name: c
    image: busybox
    lifecycle:
      preStop:
        exec: {command: ['sh','-c','sleep 10']}
YAML
```

Q105. initContainer downloads config before app starts.

```
kubectl apply -f - <<'YAML'
apiVersion: v1
kind: Pod
metadata: {name: init-download}
spec:
  volumes:
  - name: cfg
    emptyDir: {}
  initContainers:
  - name: fetch
    image: busybox
    command: ['sh','-c','wget -O /cfg/app.conf https://example.com/app.conf ||
echo key=val >/cfg/app.conf']
    volumeMounts: [{name: cfg, mountPath: /cfg}]
  containers:
  - name: app
    image: busybox
    command: ['sh','-c','cat /etc/app/app.conf; sleep 3600']
    volumeMounts: [{name: cfg, mountPath: /etc/app}]
YAML
```

Q106. Find pods using >100Mi memory in dev.

```
kubectl top pods -n dev | awk '$3 ~ /Mi$/ {gsub("Mi","",$3); if ($3>100) print $1,
$3"Mi"}'
```

Q107. Debug ImagePullBackOff steps.

```
kubectl describe pod <pod> | sed -n '/Events:/,$p'
kubectl get pod <pod> -o yaml | grep -A2 -e image -e imagePullSecrets
kubectl get secret -n <ns>
kubectl get nodes -o wide  # check network/registry access
kubectl set image pod/<pod> <container>=<valid-image>:<tag>  # or fix
imagePullSecrets
```

Q108. Export pod metrics to a file (JSON; convert to YAML if desired).

```
# Save metrics for namespace dev to JSON:
kubectl get --raw /apis/metrics.k8s.io/v1beta1/namespaces/dev/pods > /tmp/dev-pod-
metrics.json
# Optional (requires yq installed on your node): yq -P < /tmp/dev-pod-metrics.json
> /tmp/dev-pod-metrics.yaml
```

Q109. Find events related to ReplicaSet redis-rs.

```
kubectl get events --all-namespaces --field-selector
involvedObject.kind=ReplicaSet,involvedObject.name=redis-rs --sort-
by=.lastTimestamp
```

Q110. Watch logs from pods with app=frontend.

```
kubectl logs -l app=frontend -f --all-containers
```

Q111. Create Deployment vote-app with 3 replicas (image vote:v1).

```
kubectl create deploy vote-app --image=vote:v1 --replicas=3
```

Q112. Rolling update with maxUnavailable=1, maxSurge=2.

```
kubectl apply -f - <<'YAML'
apiVersion: apps/v1
kind: Deployment
metadata: {name: vote-app}
spec:
  replicas: 3
  selector: {matchLabels: {app: vote-app}}
  strategy:
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 2
  template:
    metadata: {labels: {app: vote-app}}
    spec:
      containers:
      - name: app
        image: vote:v2
YAML
```

Q113. Pause a deployment update then resume.

```
kubectl rollout pause deploy/vote-app
# validate state, then
kubectl rollout resume deploy/vote-app
```

Q114. Job multi-task with 5 completions in parallel.

```
kubectl create job multi-task --image=busybox --parallel=5 --completions=5 --
/bin/sh -c 'echo task; sleep 2'
```

Q115. CronJob hourly-job every hour.

```
kubectl create cronjob hourly-job --image=busybox --schedule='0 * * * *' --
/bin/sh -c 'date && echo hourly run'
```

Q116. Service vote-svc exposing vote-app on port 5000.

```
kubectl expose deploy vote-app --port=5000 --target-port=5000 --name=vote-svc --
type=ClusterIP
```

Q117. Restrict vote-svc to only pods with role=frontend.

```
# NetworkPolicy selecting pods behind vote-app (label app=vote-app),
# allowing ingress only from pods labeled role=frontend.
kubectl apply -f - <<'YAML'
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata: {name: vote-allow-frontend}
```

```
spec:
  podSelector: {matchLabels: {app: vote-app}}
  policyTypes: ["Ingress"]
  ingress:
  - from:
    - podSelector: {matchLabels: {role: frontend}}
YAML
```

Q118. Ingress /vote → vote-svc:5000 with TLS.

```
kubectl apply -f - <<'YAML'
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata: {name: vote-ing}
spec:
  tls:
  - hosts: [example.com]
    secretName: tls-secret
  rules:
  - host: example.com
    http:
      paths:
      - path: /vote
        pathType: Prefix
        backend: {service: {name: vote-svc, port: {number: 5000}}}
YAML
```

Q119. Inspect Endpoints created by vote-svc.

```
kubectl get endpoints endpointslice -l kubernetes.io/service-name=vote-svc -o wide
```

Q120. Delete all Services in namespace test without deleting deployments.

```
kubectl delete svc --all -n test
```

📄 CKAD Practice Questions with Solutions – Set 5 (Q121–Q160)

121–125: Core Concepts

Q121. Create a pod `gpu-pod` that requests a GPU resource (`nvidia.com/gpu:1`).
☐ **Solution:**

```
apiVersion: v1
kind: Pod
metadata:
  name: gpu-pod
spec:
  containers:
  - name: gpu-container
```

```
    image: nvidia/cuda:11.0-base
    resources:
      limits:
        nvidia.com/gpu: 1
```

Q122. Create a pod `scratch-pod` using the `alpine` image that runs `sleep 3600`.
□ `kubectl run scratch-pod --image=alpine -- sleep 3600`

Q123. List all pods in `test` namespace with their node placement.
□ `kubectl get pods -n test -o wide`

Q124. Delete all evicted pods in namespace `default`.
□ `kubectl get pods | grep Evicted | awk '{print $1}' | xargs kubectl delete pod`

Q125. Create a namespace `payments` and run a pod `pay-pod` in it.
□

```
kubectl create ns payments
kubectl run pay-pod --image=nginx -n payments
```

126–130: Configuration

Q126. Create a ConfigMap `redis-config` from literal `maxmemory=2mb`.
□ `kubectl create cm redis-config --from-literal=maxmemory=2mb`

Q127. Create a Secret `tls-secret` with `tls.crt` and `tls.key` files.
□ `kubectl create secret tls tls-secret --cert=tls.crt --key=tls.key`

Q128. Inject ConfigMap `redis-config` as environment variables into pod `redis-pod`.
□

```
envFrom:
- configMapRef:
    name: redis-config
```

Q129. Mount secret `db-secret` at `/etc/db` inside `backend-pod`.
□

```
volumeMounts:
- name: db-vol
  mountPath: /etc/db
volumes:
- name: db-vol
  secret:
    secretName: db-secret
```

Q130. Update the image of `vote-deploy` to `vote:v2` without downtime.
□ `kubectl set image deploy/vote-deploy vote=vote:v2`

131–135: Multi-Container Pods

Q131. Create a pod `sidecar-logger` with:

- app container: `nginx`
- sidecar container: `busybox` tailing `/var/log/nginx/access.log`

□ Use `emptyDir` volume, mount to both.

Q132. Add readiness probe checking `/ready` on port 8080.
□

```
readinessProbe:
  httpGet:
    path: /ready
    port: 8080
  initialDelaySeconds: 5
  periodSeconds: 10
```

Q133. Add liveness probe executing `curl http://localhost:8080/health`.
□

```
livenessProbe:
  exec:
    command: ["curl", "http://localhost:8080/health"]
```

Q134. Add `initContainer` to `app-pod` that waits for `db-svc:3306`.
□ `command: ["sh", "-c", "until nc -z db-svc 3306; do sleep 2; done"]`

Q135. Create pod with ephemeral volume shared between two containers.
□ Use `emptyDir` volume.

136–140: Jobs & CronJobs

Q136. Create a Job `batch-job` that runs `echo Hello CKAD`.
□ `kubectl create job batch-job --image=busybox -- echo "Hello CKAD"`

Q137. Run a Job `retry-job` that retries up to 3 times.
□

```
backoffLimit: 3
```

Q138. Run a Job `parallel-job` with 5 completions, 2 parallel.
□

```
completions: 5
parallelism: 2
```

Q139. Create a CronJob `midnight-job` running daily at midnight.
□ `schedule: "0 0 * * *"`

Q140. Suspend a CronJob `hourly-job`.
□ `kubectl patch cronjob hourly-job -p '{"spec":{"suspend":true}}'`

141–145: Services & Networking

Q141. Expose `vote-app` as `NodePort` on port 31000.
□ `kubectl expose deploy vote-app --port=80 --type=NodePort --name=vote-svc --target-port=5000`

Q142. Create a headless service `db-svc` for StatefulSet `mysql`.
□

```
clusterIP: None
```

Q143. Inspect endpoints of `vote-svc`.
□ `kubectl get endpoints vote-svc`

Q144. Apply a network policy allowing only pods with `role=frontend` to access `backend`.
□ YAML with `podSelector` + `from` labels.

Q145. Deny all ingress traffic to `db` pods by default.
□ NetworkPolicy with empty ingress list.

146–150: Security

Q146. Run a pod `nonroot-pod` with user ID 1000.
□

```
securityContext:
  runAsUser: 1000
```

Q147. Prevent privilege escalation in a container.
□

```
securityContext:
  allowPrivilegeEscalation: false
```

Q148. Create a ServiceAccount `build-sa` and use it in pod.
□

```
serviceAccountName: build-sa
```

Q149. Bind role `pod-reader` to SA `build-sa` in `dev` namespace.
□
```
kubectl create rolebinding rb1 --role=pod-reader --
serviceaccount=dev:build-sa -n dev
```

Q150. Restrict pod `secure-pod` to read-only root filesystem.
□

```
securityContext:
  readOnlyRootFilesystem: true
```

151–160: Troubleshooting & Advanced

Q151. Find pods consuming >200m CPU in namespace `prod`.
□ `kubectl top pod -n prod --containers`

Q152. Pod stuck in `CrashLoopBackOff`. Debug logs & describe it.
□ `kubectl logs pod-name` & `kubectl describe pod pod-name`

Q153. Find the image used by all pods in namespace `test`.
□ `kubectl get pods -n test -o jsonpath='{..image}'`

Q154. Scale deployment `vote-app` to zero replicas.
□ `kubectl scale deploy vote-app --replicas=0`

Q155. Roll back `vote-app` to previous version.
□ `kubectl rollout undo deploy vote-app`

Q156. Find events related to pod `pay-pod`.
□ `kubectl describe pod pay-pod`

Q157. Capture logs of container `sidecar` from pod `multi-container`.
□ `kubectl logs multi-container -c sidecar`

Q158. Run a temporary busybox pod for debugging DNS.
□ `kubectl run dns-test --image=busybox:1.28 -it -- nslookup kubernetes`

Q159. Delete all completed Jobs in namespace `dev`.
□ `kubectl delete jobs --all -n dev --field-selector=status.successful=1`

Q160. Save current state of deployment `vote-app` to `vote-backup.yaml`.
□ `kubectl get deploy vote-app -o yaml > vote-backup.yaml`

Core Concepts (161–168)

Q161. Create a pod `alpine-echo` that prints “hi” and exits.
□ **Solution:**

```
kubectl run alpine-echo --image=alpine --restart=Never -- /bin/sh -c 'echo
hi'
```

Q162. Create a deployment `api` with 4 replicas of `nginx:1.25`.
□

```
kubectl create deploy api --image=nginx:1.25 --replicas=4
```

Q163. Show which node each `default` pod is on.
□

```
kubectl get po -o wide
```

Q164. Delete all pods in namespace `scratch` **without** deleting other objects.
□

```
kubectl delete pod --all -n scratch
```

Q165. Print only pod names in `dev`.
□

```
kubectl get po -n dev -o name
```

Q166. Generate YAML (don't create) for a deployment `tiny` with 1 replica `busybox` running `sleep 3600`.
□

```
kubectl create deploy tiny --image=busybox -o yaml --dry-run=client --
/bin/sh -c 'sleep 3600'
```

Q167. Force delete a pod `hung` immediately.
□

```
kubectl delete pod hung --grace-period=0 --force
```

Q168. Create a namespace `qa` and set it as default context.
□

```
kubectl create ns qa
kubectl config set-context --current --namespace=qa
```

Config & Secrets (169–176)

Q169. Create ConfigMap `app-kv` with `ENV=prod` and `LOG_LEVEL=info`.
□

```
kubectl create cm app-kv --from-literal=ENV=prod --from-
literal=LOG_LEVEL=info
```

Q170. Mount `app-kv` as environment variables in pod `cm-env`.
□

```
apiVersion: v1
kind: Pod
metadata: {name: cm-env}
spec:
  containers:
  - name: c
    image: busybox
    command: ["sh","-c","env; sleep 3600"]
    envFrom:
    - configMapRef: {name: app-kv}
```

Q171. Create secret `db-pass` with key `password=Sup3rS3cret`.
□

```
kubectl create secret generic db-pass --from-literal=password=Sup3rS3cret
```

Q172. Inject `db-pass` as a file under `/etc/creds` in pod `cred-pod`.
□

```
apiVersion: v1
kind: Pod
metadata: {name: cred-pod}
spec:
  volumes:
  - name: s
    secret: {secretName: db-pass}
  containers:
  - name: c
    image: busybox
    command: ["sh","-c","ls -l /etc/creds; sleep 3600"]
    volumeMounts:
    - {name: s, mountPath: /etc/creds, readOnly: true}
```

Q173. Add env `POD_NAME` and `NODE_NAME` via Downward API in pod `down-env`.
□

```
apiVersion: v1
kind: Pod
metadata: {name: down-env}
spec:
  containers:
  - name: c
    image: busybox
    command: ["sh","-c","echo $POD_NAME@$NODE_NAME; sleep 3600"]
    env:
    - name: POD_NAME
      valueFrom: {fieldRef: {fieldPath: metadata.name}}
    - name: NODE_NAME
      valueFrom: {fieldRef: {fieldPath: spec.nodeName}}
```

Q174. Patch deployment `api` to add env `FEATURE_X=true` to container `nginx`.
□

```
kubectl patch deploy/api --type='json' -p='[

{"op":"add","path":"/spec/template/spec/containers/0/env","value":[{"name":
"FEATURE_X","value":"true"}]}
]'
```

Q175. Create ConfigMap `env-file` from file `.env`.
□

```
kubectl create cm env-file --from-env-file=.env
```

Q176. Update deployment `api` to consume `env-file` ConfigMap as env vars.
□

```
kubectl set env deploy/api --from=configmap/env-file
```

Multi-Container, Probes & Lifecycle (177–184)

Q177. Create pod `sidecar-app` with `web:nginx` and `log:busybox` tailing `/var/log/nginx/access.log`.
□

```
apiVersion: v1
kind: Pod
metadata: {name: sidecar-app}
spec:
  volumes: [{name: logs, emptyDir: {}}]
  containers:
  - name: web
    image: nginx
    volumeMounts: [{name: logs, mountPath: /var/log/nginx}]
  - name: log
    image: busybox
    command: ["sh","-c","tail -F /var/log/nginx/access.log"]
    volumeMounts: [{name: logs, mountPath: /var/log/nginx}]
```

Q178. Add HTTP liveness probe on `/healthz` port 8080 to deployment `api`.
□

```
kubectl patch deploy/api --type='merge' -p '
spec:
  template:
    spec:
      containers:
      - name: nginx
        livenessProbe:
          httpGet: {path: /healthz, port: 8080}
          initialDelaySeconds: 5
          periodSeconds: 10
'
```

Q179. Add TCP readiness probe on port 5432 to pod `db-client` container `c`.
□

```
kubectl patch pod db-client --type='json' -p='[
 {"op":"add","path":"/spec/containers/0/readinessProbe","value":{
   "tcpSocket":{"port":5432},"initialDelaySeconds":5,"periodSeconds":5
 }}
]'
```

Q180. Add `postStart` hook that echoes “started” for pod `hook-pod`.
□

```
apiVersion: v1
kind: Pod
metadata: {name: hook-pod}
spec:
  containers:
  - name: c
    image: busybox
    command: ["sh","-c","sleep 3600"]
    lifecycle:
      postStart:
        exec: {command: ["sh","-c","echo started"]}
```

Q181. Add `preStop` hook that sleeps 5s for deployment `api`.
□

```
kubectl patch deploy/api --type='merge' -p '
spec:
  template:
    spec:
      terminationGracePeriodSeconds: 30
      containers:
      - name: nginx
        lifecycle:
          preStop:
            exec: {command: ["sh","-c","sleep 5"]}
'
```

Q182. Add init container to `api` pods waiting for `redis:6379`.
□

```
kubectl patch deploy/api --type='merge' -p '
spec:
  template:
    spec:
      initContainers:
      - name: wait-redis
        image: busybox
        command: ["sh","-c","until nc -z redis 6379; do sleep 2; done"]
'
```

Q183. Share data between two containers via `emptyDir` in pod `share`.
□

```
apiVersion: v1
kind: Pod
metadata: {name: share}
spec:
  volumes: [{name: data, emptyDir: {}}]
  containers:
  - name: w
    image: busybox
    command: ["sh","-c","echo hi > /data/x; sleep 3600"]
    volumeMounts: [{name: data, mountPath: /data}]
  - name: r
    image: busybox
    command: ["sh","-c","cat /data/x; sleep 3600"]
    volumeMounts: [{name: data, mountPath: /data}]
```

Q184. Add resource requests/limits (200m CPU, 256Mi mem) to `api` container.
□

```
kubectl set resources deploy/api -c nginx --requests=cpu=200m,memory=256Mi
--limits=cpu=200m,memory=256Mi
```

Jobs & CronJobs (185–190)

Q185. Create Job `wordcount` printing word count of a string.
□

```
kubectl create job wordcount --image=busybox -- /bin/sh -c "echo 'one two
two' | tr ' ' '\n' | sort | uniq -c"
```

Q186. Create Job `retry3` with `backoffLimit: 3`.
□

```
apiVersion: batch/v1
kind: Job
metadata: {name: retry3}
spec:
  backoffLimit: 3
  template:
    spec:
      restartPolicy: Never
      containers:
      - name: c
        image: busybox
        command: ["sh","-c","exit 1"]
```

Q187. Create parallel Job `multi` with `completions: 6`, `parallelism: 3`.
□

```
kubectl create job multi --image=busybox --parallel=3 --completions=6 --
/bin/sh -c 'date; sleep 2'
```

Q188. Create CronJob `hourly-clean` running at minute 15 every hour.
□

```
kubectl create cronjob hourly-clean --image=busybox --schedule='15 * * * *'
-- /bin/sh -c 'echo clean; date'
```

Q189. Suspend CronJob `hourly-clean`.
□

```
kubectl patch cronjob hourly-clean -p '{"spec":{"suspend":true}}'
```

Q190. Create CronJob `weekday-9am` running Mon–Fri at 09:00.
□

```
kubectl create cronjob weekday-9am --image=busybox --schedule='0 9 * * 1-5'
-- /bin/sh -c 'echo hello'
```

Services, Ingress & NetworkPolicy (191–196)

Q191. Expose `api` as ClusterIP `api-svc` on port 80 → targetPort 80.
□

```
kubectl expose deploy api --name=api-svc --port=80 --target-port=80 --
type=ClusterIP
```

Q192. Create NodePort service `api-np` on nodePort `30088`.
□

```
apiVersion: v1
kind: Service
metadata: {name: api-np}
spec:
  type: NodePort
  selector: {app: api}
  ports:
  - port: 80
    targetPort: 80
    nodePort: 30088
```

Q193. Create Ingress `api-ing` routing `/api` to `api-svc:80`.
□

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata: {name: api-ing}
spec:
  rules:
  - http:
      paths:
      - path: /api
        pathType: Prefix
        backend:
          service: {name: api-svc, port: {number: 80}}
```

Q194. Create NetworkPolicy `deny-all` in `dev` blocking all ingress to pods.
□

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata: {name: deny-all, namespace: dev}
spec:
  podSelector: {}
  policyTypes: ["Ingress"]
```

Q195. Allow ingress to label `app=api` only from pods with label `role=frontend`.
□

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata: {name: api-allow-frontend}
spec:
  podSelector: {matchLabels: {app: api}}
  policyTypes: ["Ingress"]
  ingress:
  - from:
    - podSelector: {matchLabels: {role: frontend}}
```

Q196. List endpoints and EndpointSlices for `api-svc`.
□

```
kubectl get endpoints api-svc -o wide
kubectl get endpointslice -l kubernetes.io/service-name=api-svc -o wide
```

Storage (197–198)

Q197. Create PVC `app-pvc` requesting 2Gi RWO. Mount it at `/data` in pod `store`.
□

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata: {name: app-pvc}
spec:
  accessModes: [ReadWriteOnce]
  resources: {requests: {storage: 2Gi}}
---
apiVersion: v1
kind: Pod
metadata: {name: store}
spec:
  volumes: [{name: v, persistentVolumeClaim: {claimName: app-pvc}}]
  containers:
  - name: c
    image: busybox
    command: ["sh","-c","touch /data/ok; sleep 3600"]
    volumeMounts: [{name: v, mountPath: /data}]
```

Q198. Expand PVC `app-pvc` to 5Gi (assume storage class allows expansion).
□

```
kubectl patch pvc app-pvc -p
'{"spec":{"resources":{"requests":{"storage":"5Gi"}}}}'
```

Security, RBAC & Scheduling (199–200)

Q199. Create Role `cm-reader` in `qa` allowing `get,list,watch` on configmaps; bind to ServiceAccount `reader`.
□

```
kubectl -n qa create role cm-reader --verb=get --verb=list --verb=watch --
resource=configmaps
kubectl -n qa create rolebinding cm-reader-bind --role=cm-reader --
serviceaccount=qa:reader
```

Q200. Schedule a pod `gpu-only` to nodes labeled `gpu=true` using node affinity (required).
□

```
apiVersion: v1
kind: Pod
metadata: {name: gpu-only}
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: gpu
            operator: In
            values: ["true"]
  containers:
  - name: c
    image: nvidia/cuda:11.0-base
    command: ["sh","-c","nvidia-smi || true; sleep 3600"]
```

If you'd like, I can package **Q161–Q200** into a **.docx** (questions + solutions) like the earlier sets.