# ☸ Comprehensive Kubernetes CLI Reference

A complete guide to `kubectl` command-line interface (CLI) commands organized by functionality. This reference covers everything from cluster management to advanced debugging and troubleshooting.

---

## ⚙ 1. Installation & Setup

### Install kubectl

#### macOS:

```bash
brew install kubectl
```

#### Linux:

```bash
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
```

#### Windows:

```bash
choco install kubernetes-cli
# or use Windows package manager
```

### Verify Installation

- `kubectl version` — Show client and server Kubernetes version
- `kubectl version --client` — Show only client version
- `kubectl version --short` — Compact version output
- `kubectl version --output=json` — JSON format version output

### Configuration & Contexts

- `kubectl config view` — Display current kubeconfig

- `kubectl config view --flatten` — Show merged kubeconfig as single file
- `kubectl config view --raw` — Show raw kubeconfig (unredacted)
- `kubectl config current-context` — Display currently active context
- `kubectl config get-contexts` — List all available contexts
- `kubectl config use-context <context-name>` — Switch to different context
- `kubectl config set-context <context> --namespace=<ns>` — Set default namespace for context
- `kubectl config set-cluster <name> --server=<url>` — Configure cluster
- `kubectl config set-credentials <name> --token=<token>` — Configure user credentials
- `kubectl cluster-info` — Display cluster information and endpoints
- `kubectl cluster-info dump` — Export cluster diagnostics
- `kubectl cluster-info dump --output-directory=./cluster-dump` — Save diagnostics to directory

---

## 🏢 2. Cluster Management

### Cluster Information

- `kubectl get nodes` — List all nodes in cluster
- `kubectl get nodes -o wide` — Nodes with additional details (IP, OS, kernel)
- `kubectl get nodes --show-labels` — Show node labels
- `kubectl get nodes --selector=<label-selector>` — Filter nodes by labels
- `kubectl describe node <node-name>` — Get detailed node information
- `kubectl get nodes --no-headers` — List nodes without headers
- `kubectl get nodes -o jsonpath='{.items[*].metadata.name}'` — Extract node names

### Node Management

- `kubectl cordon <node-name>` — Mark node as unschedulable (no new pods)
- `kubectl uncordon <node-name>` — Mark node as schedulable again
- `kubectl drain <node-name>` — Evict all pods from node for maintenance
- `kubectl drain <node-name> --ignore-daemonsets --delete-emptydir-data` — Drain with forced options
- `kubectl drain <node-name> --force --grace-period=0` — Force immediate drain

- `kubectl delete node <node-name>` — Remove node from cluster
- `kubectl top nodes` — Display node resource usage (CPU/memory)
- `kubectl top nodes --containers` — Node usage with container breakdown
- `kubectl label nodes <node> <key>=<value>` — Add label to node
- `kubectl label nodes <node> <key>=<value> --overwrite` — Update node label
- `kubectl annotate nodes <node> <key>=<value>` — Add annotation to node
- `kubectl taint nodes <node> <key>=<value>:NoSchedule` — Add taint to node
- `kubectl taint nodes <node> <key>:NoSchedule-` — Remove taint from node

### Cluster Health & Diagnostics

- `kubectl get componentstatuses` — Check status of cluster components
- `kubectl get endpoints` — List available API endpoints
- `kubectl get events --all-namespaces` — View cluster-wide events
- `kubectl get events --field-selector involvedObject.kind=Pod` — Events for specific resource type
- `kubectl describe cluster` — Get cluster description
- `kubectl get storageclass` — List storage classes available

---

## 📦 3. Resource Management (Core Concepts)

### Creating Resources

- `kubectl create deployment <name> --image=<image>` — Create deployment
- `kubectl create service clusterip <name> --tcp=<port>:<port>` — Create ClusterIP service
- `kubectl create configmap <name> --from-literal=<key>=<value>` — Create ConfigMap
- `kubectl create configmap <name> --from-file=<path>` — Create ConfigMap from file
- `kubectl create secret generic <name> --from-literal=<key>=<value>` — Create secret
- `kubectl create secret docker-registry <name> --docker-server=<url>` — Create docker registry secret
- `kubectl create namespace <name>` — Create namespace
- `kubectl create pvc <name> --size=<size>` — Create PersistentVolumeClaim

### Applying Resources from Files

- `kubectl apply -f <file.yaml>` — Apply configuration from file
- `kubectl apply -f <directory>` — Apply all YAML files in directory
- `kubectl apply -f <url>` — Apply configuration from URL
- `kubectl apply -k <directory>` — Apply configuration using Kustomization
- `kubectl apply -f - < <file.yaml>` — Apply from stdin

## Getting Resources

- `kubectl get pods` — List pods in current namespace
- `kubectl get pods --all-namespaces` — List pods in all namespaces
- `kubectl get pods -n <namespace>` — List pods in specific namespace
- `kubectl get pods -o wide` — Pods with IP, node, and status details
- `kubectl get pods -o json` — Output in JSON format
- `kubectl get pods -o yaml` — Output in YAML format
- `kubectl get pods --field-selector=status.phase=Running` — Filter by field
- `kubectl get pods --selector=<label-key>=<label-value>` — Filter by labels
- `kubectl get pods --sort-by=.metadata.creationTimestamp` — Sort results
- `kubectl get all` — List all resources in namespace
- `kubectl get all -n <namespace>` — All resources in specific namespace

## Resource Types & Aliases

### Common resource types:

- `pod` / `pods` / `po` — Pod containers
- `deployment` / `deployments` / `deploy` — Deployments
- `statefulset` / `statefulsets` / `sts` — StatefulSet
- `daemonset` / `daemonsets` / `ds` — DaemonSet
- `job` / `jobs` — Job
- `cronjob` / `cronjobs` / `cj` — CronJob
- `service` / `services` / `svc` — Service
- `ingress` / `ingresses` / `ing` — Ingress

- `configmap` / `configmaps` / `cm` — ConfigMap
- `secret` / `secrets` — Secret
- `persistentvolume` / `persistentvolumes` / `pv` — PersistentVolume
- `persistentvolumeclaim` / `persistentvolumeclaims` / `pvc` — PersistentVolumeClaim
- `storageclass` / `storageclasses` / `sc` — StorageClass
- `namespace` / `namespaces` / `ns` — Namespace
- `node` / `nodes` — Node
- `roleBinding` / `rolebindings` / `rb` — RoleBinding
- `clusterrole` / `clusterroles` — ClusterRole
- `event` / `events` / `ev` — Event

---

## 🔍 4. Inspection & Debugging

### Describing Resources

- `kubectl describe pod <pod-name>` — Get detailed pod information
- `kubectl describe deployment <deployment-name>` — Get deployment details
- `kubectl describe service <service-name>` — Get service details
- `kubectl describe pvc <pvc-name>` — Get PersistentVolumeClaim details
- `kubectl describe node <node-name>` — Get node details and conditions
- `kubectl describe <resource-type> <resource-name>` — Generic describe command

### Pod Debugging

- `kubectl logs <pod-name>` — View pod logs
- `kubectl logs <pod-name> -c <container-name>` — View specific container logs
- `kubectl logs <pod-name> --all-containers` — View all container logs
- `kubectl logs <pod-name> --previous` — View previous crashed pod logs
- `kubectl logs <pod-name> -f` — Follow live logs
- `kubectl logs <pod-name> --tail=50` — Show last 50 log lines
- `kubectl logs <pod-name> --timestamps` — Show logs with timestamps

- `kubectl logs <pod-name> --since=1h` — Logs from last hour
- `kubectl logs <deployment-name>` — Logs from deployment pods
- `kubectl logs -l <label>=<value>` — Logs from pods with labels

### Executing Commands in Pods

- `kubectl exec -it <pod-name> -- /bin/bash` — Interactive shell in pod
- `kubectl exec -it <pod-name> -c <container> -- /bin/sh` — Shell in specific container
- `kubectl exec <pod-name> -- <command>` — Execute single command
- `kubectl exec <pod-name> -- env` — View pod environment variables
- `kubectl exec <pod-name> -- ls -la /app` — Execute commands with arguments

### Pod Interaction

- `kubectl attach <pod-name>` — Attach to running pod
- `kubectl port-forward <pod-name> 8080:80` — Forward local port to pod
- `kubectl port-forward svc/<service-name> 8080:80` — Forward port via service
- `kubectl port-forward deployment/<deployment-name> 8080:80` — Forward via deployment
- `kubectl cp <pod-name>:/path/to/file ./local/file` — Copy file from pod
- `kubectl cp ./local/file <pod-name>:/path/to/file` — Copy file to pod

### Event & Status Information

- `kubectl get events` — List cluster events
- `kubectl get events -n <namespace>` — Events in namespace
- `kubectl get events --field-selector involvedObject.name=<pod-name>` — Events for specific pod
- `kubectl describe pod <pod-name> | grep -A 20 "Events"` — Pod events section

### Resource Metrics & Health

- `kubectl top pod` — Display pod resource usage (requires metrics-server)
- `kubectl top pod --all-namespaces` — Pod usage across all namespaces
- `kubectl top pod -n <namespace>` — Pod usage in namespace
- `kubectl top node` — Display node resource usage
- `kubectl get resourcequota` — View resource quotas

- `kubectl describe resourcequota <quota-name>` — Quota details

---

## 🏷️ 5. Labels, Selectors & Annotations

### Working with Labels

- `kubectl label pod <pod-name> <key>=<value>` — Add label to pod
- `kubectl label pod <pod-name> <key>=<value> --overwrite` — Update existing label
- `kubectl label pods -l <key>=<value> <new-key>=<new-value>` — Label multiple pods
- `kubectl label all -l <key>=<value> <new-key>=<new-value>` — Label all resources matching selector
- `kubectl label <resource-type> <resource-name> <key>-` — Remove label

### Viewing Labels

- `kubectl get pods --show-labels` — Display labels for all pods
- `kubectl get pods -L <label-key>` — Show specific label column
- `kubectl get pods -L <key1>,<key2>` — Show multiple label columns

### Label Selectors

- `kubectl get pods -l <key>=<value>` — Resources matching exact label
- `kubectl get pods -l <key>!=<value>` — Resources not matching label
- `kubectl get pods -l <key> in (value1,value2)` — Multiple label values
- `kubectl get pods -l <key> notin (value1)` — Labels not in set
- `kubectl get pods -l <key>` — Resources with label key (any value)
- `kubectl get pods -l '!<key>'` — Resources without label key

### Annotations

- `kubectl annotate pod <pod-name> <key>=<value>` — Add annotation
- `kubectl annotate <resource-type> <resource-name> <key>=<value> --overwrite` — Update annotation
- `kubectl annotate pod <pod-name> <key>-` — Remove annotation
- `kubectl describe <resource-type> <resource-name> | grep Annotations` — View annotations

---

## 📝 6. Configuration Management

### ConfigMaps

- `kubectl create configmap <name> --from-literal=<key>=<value>` — Create ConfigMap with key-value pairs
- `kubectl create configmap <name> --from-file=<path>` — Create from file
- `kubectl create configmap <name> --from-file=<dir>` — Create from directory
- `kubectl get configmap` — List ConfigMaps
- `kubectl get configmap <name> -o yaml` — View ConfigMap in YAML
- `kubectl describe configmap <name>` — Get ConfigMap details
- `kubectl edit configmap <name>` — Edit ConfigMap
- `kubectl delete configmap <name>` — Delete ConfigMap

### Secrets

- `kubectl create secret generic <name> --from-literal=<key>=<value>` — Create generic secret
- `kubectl create secret docker-registry <name> --docker-server=<url> --docker-username=<u> --docker-password=<p>` — Create registry secret
- `kubectl create secret tls <name> --cert=<path> --key=<path>` — Create TLS secret
- `kubectl get secret` — List secrets
- `kubectl get secret <name> -o yaml` — View secret in YAML
- `kubectl describe secret <name>` — Get secret details
- `kubectl edit secret <name>` — Edit secret
- `kubectl delete secret <name>` — Delete secret
- `kubectl get secret <name> -o jsonpath='{.data.<key>}' | base64 --decode` — Decode secret value

---

## 🚀 7. Deployment Management

### Deployment Operations

- `kubectl create deployment <name> --image=<image>` — Create deployment
- `kubectl create deployment <name> --image=<image> --replicas=3` — Create with replicas
- `kubectl get deployment` — List deployments

- `kubectl get deployment -o wide` — Deployments with detailed info
- `kubectl describe deployment <name>` — Get deployment details
- `kubectl edit deployment <name>` — Edit deployment
- `kubectl scale deployment <name> --replicas=3` — Scale deployment
- `kubectl autoscale deployment <name> --min=1 --max=10 --cpu-percent=80` — Enable autoscaling
- `kubectl delete deployment <name>` — Delete deployment

### Rollout Management

- `kubectl rollout status deployment/<name>` — Check rollout status
- `kubectl rollout history deployment/<name>` — View rollout history
- `kubectl rollout history deployment/<name> --revision=2` — Details of specific revision
- `kubectl rollout undo deployment/<name>` — Rollback to previous version
- `kubectl rollout undo deployment/<name> --to-revision=2` — Rollback to specific revision
- `kubectl rollout restart deployment/<name>` — Restart all pods in deployment
- `kubectl set image deployment/<name> <container>=<image>` — Update container image
- `kubectl set image deployment/<name> <container>=<image> --record` — Update with rollout record
- `kubectl patch deployment <name> -p '{"spec":{"replicas":5}}'` — Patch deployment

### Deployment Updates

- `kubectl set env deployment/<name> <key>=<value>` — Set environment variables
- `kubectl set resources deployment/<name> --limits=cpu=100m,memory=128Mi` — Set resource limits
- `kubectl set resources deployment/<name> --requests=cpu=50m,memory=64Mi` — Set resource requests

---

## 🎯 8. StatefulSets, DaemonSets & Jobs

### StatefulSet Operations

- `kubectl create statefulset <name> --image=<image>` — Create StatefulSet
- `kubectl get statefulset` — List StatefulSets
- `kubectl describe statefulset <name>` — Get StatefulSet details
- `kubectl scale statefulset <name> --replicas=3` — Scale StatefulSet

- `kubectl delete statefulset <name>` — Delete StatefulSet
- `kubectl delete statefulset <name> --cascade=orphan` — Delete without removing pods

### DaemonSet Operations

- `kubectl get daemonset` — List DaemonSets
- `kubectl describe daemonset <name>` — Get DaemonSet details
- `kubectl edit daemonset <name>` — Edit DaemonSet
- `kubectl delete daemonset <name>` — Delete DaemonSet

### Job & CronJob Operations

- `kubectl create job <name> --image=<image>` — Create job
- `kubectl get jobs` — List jobs
- `kubectl describe job <name>` — Get job details
- `kubectl logs job/<name>` — View job logs
- `kubectl delete job <name>` — Delete job
- `kubectl get job <name> -o jsonpath='{.items[0].status}'` — Get job status
- `kubectl get cronjob` — List CronJobs
- `kubectl create cronjob <name> --image=<image> --schedule="0 * * * *"` — Create CronJob
- `kubectl delete cronjob <name>` — Delete CronJob

---

## 🔌 9. Service & Networking

### Service Management

- `kubectl create service clusterip <name> --tcp=<port>:<port>` — Create ClusterIP service
- `kubectl create service nodeport <name> --tcp=<port>:<port>` — Create NodePort service
- `kubectl create service loadbalancer <name> --tcp=<port>:<port>` — Create LoadBalancer service
- `kubectl get service` — List services
- `kubectl get service -o wide` — Services with endpoint details
- `kubectl describe service <name>` — Get service details
- `kubectl edit service <name>` — Edit service

- `kubectl delete service <name>` — Delete service
- `kubectl get endpoints <service-name>` — View service endpoints (backend pods)

### Service Discovery & DNS

- `kubectl get dns` — List DNS services
- `kubectl get service <name> -o jsonpath='{.spec.clusterIP}'` — Get cluster IP
- `kubectl get service <name> -o jsonpath='{.status.loadBalancer.ingress[0].ip}'` — Get LoadBalancer IP
- `kubectl get svc <name> -o jsonpath='{.spec.ports[*].nodePort}'` — Get NodePort

### Ingress Management

- `kubectl get ingress` — List ingresses
- `kubectl describe ingress <name>` — Get ingress details
- `kubectl edit ingress <name>` — Edit ingress
- `kubectl delete ingress <name>` — Delete ingress
- `kubectl get ingress --all-namespaces` — Ingresses across all namespaces

---

## 💾 10. Storage Management

### PersistentVolume (PV) Operations

- `kubectl get pv` — List all persistent volumes
- `kubectl describe pv <name>` — Get PV details
- `kubectl get pv --show-labels` — PVs with labels
- `kubectl patch pv <name> -p '{"spec":{"claimRef":null}}'` — Unbind PV from PVC

### PersistentVolumeClaim (PVC) Operations

- `kubectl get pvc` — List PVCs
- `kubectl get pvc -n <namespace>` — PVCs in specific namespace
- `kubectl describe pvc <name>` — Get PVC details
- `kubectl delete pvc <name>` — Delete PVC
- `kubectl get pvc <name> -o jsonpath='{.spec.volumeName}'` — Get bound PV name

### StorageClass Operations

- `kubectl get storageclass` — List storage classes
- `kubectl describe storageclass <name>` — Get storage class details
- `kubectl get storageclass -o yaml` — Storage classes in YAML

---

## 👥 11. RBAC (Role-Based Access Control)

### Role & RoleBinding

- `kubectl create role <name> --verb=<verb> --resource=<resource>` — Create role
- `kubectl get role` — List roles
- `kubectl describe role <name>` — Get role details
- `kubectl edit role <name>` — Edit role
- `kubectl delete role <name>` — Delete role
- `kubectl create rolebinding <name> --clusterrole=<role> --serviceaccount=<ns>:<sa>` — Create RoleBinding
- `kubectl get rolebinding` — List RoleBindings
- `kubectl describe rolebinding <name>` — Get RoleBinding details

### ClusterRole & ClusterRoleBinding

- `kubectl get clusterrole` — List cluster roles
- `kubectl describe clusterrole <name>` — Get cluster role details
- `kubectl get clusterrolebinding` — List cluster role bindings
- `kubectl describe clusterrolebinding <name>` — Get cluster role binding details
- `kubectl create clusterrole <name> --verb=<verb> --resource=<resource>` — Create cluster role
- `kubectl create clusterrolebinding <name> --clusterrole=<role> --serviceaccount=<ns>:<sa>` — Create cluster role binding

### ServiceAccount

- `kubectl create serviceaccount <name>` — Create service account
- `kubectl get serviceaccount` — List service accounts
- `kubectl describe serviceaccount <name>` — Get service account details

- `kubectl delete serviceaccount <name>` — Delete service account

### RBAC Authorization Testing

- `kubectl auth can-i <verb> <resource>` — Test if user can perform action
- `kubectl auth can-i create pods` — Check if current user can create pods
- `kubectl auth can-i delete deployments --as=<user>` — Check specific user permissions
- `kubectl auth can-i '*' '*'` — Check all permissions

---

##  12. Namespace Management

### Namespace Operations

- `kubectl create namespace <name>` — Create namespace
- `kubectl get namespace` — List namespaces
- `kubectl get ns` — Abbreviated namespace listing
- `kubectl describe namespace <name>` — Get namespace details
- `kubectl edit namespace <name>` — Edit namespace
- `kubectl delete namespace <name>` — Delete namespace
- `kubectl config set-context --current --namespace=<namespace>` — Set default namespace

### Namespace Resource Quotas

- `kubectl create resourcequota <name> --hard=pods=10,memory=1Gi` — Create resource quota
- `kubectl get resourcequota` — List resource quotas
- `kubectl describe resourcequota <name>` — Get quota details
- `kubectl delete resourcequota <name>` — Delete quota

### Network Policies

- `kubectl get networkpolicy` — List network policies
- `kubectl describe networkpolicy <name>` — Get policy details
- `kubectl delete networkpolicy <name>` — Delete policy

---

## 🖍️ 13. Editing & Patching Resources

### Editing Resources

- `kubectl edit <resource-type> <resource-name>` — Open resource in default editor
- `kubectl edit pod <name>` — Edit pod definition
- `kubectl edit deployment <name>` — Edit deployment
- `kubectl set env <resource-type>/<name> <key>=<value>` — Set environment variables

### Patching Resources

- `kubectl patch pod <name> -p '{"spec":{"activeDeadlineSeconds":100}}'` — Apply JSON patch
- `kubectl patch deployment <name> -p '{"spec":{"replicas":3}}'` — Patch with JSON
- `kubectl patch service <name> -p '{"spec":{"type":"LoadBalancer"}}'` — Change service type
- `kubectl patch <resource> <name> --type merge -p '{"spec":{"image":"nginx:1.20"}}'` — Merge patch
- `kubectl patch <resource> <name> --type strategic-merge-patch -p <patch>` — Strategic merge

### kubectl Apply & Replace

- `kubectl apply -f <file.yaml>` — Apply declarative configuration
- `kubectl apply -f - < <file>` — Apply from stdin
- `kubectl replace -f <file.yaml>` — Replace entire resource
- `kubectl replace --force -f <file.yaml>` — Force replace (delete then create)

---

## 🗑️ 14. Deletion & Cleanup

### Deleting Resources

- `kubectl delete pod <name>` — Delete pod
- `kubectl delete pod <name> --grace-period=0 --force` — Force delete pod
- `kubectl delete deployment <name>` — Delete deployment
- `kubectl delete -f <file.yaml>` — Delete resources from file
- `kubectl delete <resource-type> -l <label-key>=<label-value>` — Delete by label
- `kubectl delete --all <resource-type>` — Delete all resources of type

- `kubectl delete --all <resource-type> -n <namespace>` — Delete all in namespace

### Cleanup Strategies

- `kubectl delete pod --field-selector=status.phase=Failed` — Delete failed pods
- `kubectl delete pod --field-selector=status.phase=Succeeded` — Delete succeeded pods
- `kubectl delete pod --older-than=24h` — Delete old pods (if supported)

---

## 🔐 15. Security & Authentication

### Certificate Management

- `kubectl create secret tls <name> --cert=<path> --key=<path>` — Create TLS secret
- `kubectl get secret <name> -o jsonpath='{.data.tls\.crt}'` — Get certificate
- `kubectl describe secret <name>` — View certificate details

### RBAC & Authorization

- `kubectl create clusterrolebinding <name> --clusterrole=cluster-admin --serviceaccount=<ns>:<sa>` — Grant admin to service account
- `kubectl create rolebinding <name> --role=<role> --serviceaccount=<ns>:<sa> -n <ns>` — Bind role to service account

### Authentication & Context

- `kubectl config use-context <context>` — Switch context (user/cluster/namespace)
- `kubectl config get-contexts` — List available contexts
- `kubectl config delete-context <context>` — Remove context
- `kubectl whoami` — Display current user (requires auth provider support)

---

## 📊 16. Output Formatting & Queries

### Output Formats

- `kubectl get pods -o json` — JSON output
- `kubectl get pods -o yaml` — YAML output
- `kubectl get pods -o wide` — Wide output (more columns)

- `kubectl get pods -o name` — Only resource names
- `kubectl get pods -o custom-columns=<columns>` — Custom columns
- `kubectl get pods -o jsonpath='<path>'` — JSONPath query
- `kubectl get pods --no-headers` — Omit headers
- `kubectl get pods --sort-by=<field>` — Sort output

### JSONPath Queries

- `kubectl get pod <name> -o jsonpath='{.metadata.name}'` — Get pod name
- `kubectl get pods -o jsonpath='{.items[*].metadata.name}'` — All pod names
- `kubectl get pods -o jsonpath='{.items[*].status.podIP}'` — All pod IPs
- `kubectl get nodes -o jsonpath='{.items[*].status.nodeInfo.kernelVersion}'` — Node kernel versions
- `kubectl get pods -o jsonpath='{.items[?(@.status.phase=="Running")].metadata.name}'` — Running pods only

### Advanced Filtering

- `kubectl get pods --field-selector=status.phase=Running` — Filter by field
- `kubectl get pods --field-selector=metadata.namespace=default` — Filter by namespace field
- `kubectl get pods --field-selector=spec.nodeName=<node>` — Pods on specific node

---

## 🔧 17. Advanced Operations

### Rolling Updates & Blue-Green Deployments

- `kubectl set image deployment/<name> <container>=<image>:<tag>` — Update image
- `kubectl rollout status deployment/<name>` — Monitor rollout
- `kubectl rollout undo deployment/<name>` — Rollback deployment
- `kubectl set image deployment/<name> <container>=<image> --record` — Update with record

### Scaling & Autoscaling

- `kubectl scale deployment <name> --replicas=5` — Manual scaling
- `kubectl autoscale deployment <name> --min=2 --max=10 --cpu-percent=50` — Enable HPA
- `kubectl get hpa` — List horizontal pod autoscalers
- `kubectl describe hpa <name>` — Get HPA details

### Multi-Container Pod Operations

- `kubectl logs <pod-name> -c <container>` — Logs from specific container
- `kubectl exec -it <pod-name> -c <container> -- /bin/bash` — Shell in specific container
- `kubectl top pod <pod-name> --containers` — Resource usage per container

### Pod Disruption Budgets

- `kubectl get poddisruptionbudget` — List PDBs
- `kubectl describe pdb <name>` — Get PDB details
- `kubectl create pdb <name> --selector=<label> --min-available=1` — Create PDB

### Custom Resource Definitions (CRDs)

- `kubectl get crd` — List custom resource definitions
- `kubectl describe crd <name>` — Get CRD details
- `kubectl get <custom-resource-type>` — List custom resources
- `kubectl apply -f <crd-file.yaml>` — Install CRD

---

## 🐛 18. Troubleshooting & Diagnostics

### Pod Troubleshooting

- `kubectl describe pod <name>` — Complete pod information and events
- `kubectl logs <name>` — View pod logs
- `kubectl logs <name> --previous` — View crashed pod logs
- `kubectl exec -it <name> -- /bin/bash` — Interactive debugging
- `kubectl get pod <name> -o yaml` — Full pod definition
- `kubectl get events` — Cluster events

### Common Issues

**Pod stuck in Pending:**

```bash
```

```
kubectl describe pod <name>
# Check: resource limits, node availability, persistent volume binding
```

**Pod stuck in CrashLoopBackOff:**

```
bash

kubectl logs <pod-name> --previous
# View initialization and application logs
```

**Pod stuck in ImagePullBackOff:**

```
bash

kubectl describe pod <name>
# Check: image name, registry credentials, image availability
```

### Debugging Tools

- `kubectl run -it --rm debug --image=busybox -- /bin/sh` — Run debug container
- `kubectl run -it --rm debug --image=ubuntu -- /bin/bash` — Run Ubuntu debug container
- `kubectl debug node/<node-name> -it --image=ubuntu` — Debug node issues
- `kubectl debug <pod-name> -it --image=ubuntu` — Debug pod (creates ephemeral container)

### Network Debugging

- `kubectl run tmp-shell --rm -it --image=praqma/network-multitool -- sh` — Network debugging pod
- `kubectl exec -it <pod> -- wget -O- http://<service>:<port>` — Test service connectivity
- `kubectl exec -it <pod> -- nslookup <service-name>` — DNS resolution test

---

## 🔄 19. Plugin & Extension Management

### kubectl Plugins

- `kubectl plugin list` — List installed plugins
- `kubectl plugin <name>` — Execute plugin
- `kubectl krew list` — List installed packages (requires krew)
- `kubectl krew install <plugin>` — Install plugin via krew

- `kubectl krew upgrade` — Upgrade all plugins

### Popular Plugins

- `kubectl ctx` — Quickly switch contexts
- `kubectl ns` — Quickly switch namespaces
- `kubectl tail` — Tail logs from multiple resources
- `kubectl debug` — Debug pods and nodes
- `kubectl tree` — Show resource tree relationships
- `kubectl cost` — Analyze resource costs

---

## 📋 20. Common Workflows & Examples

### Deployment Workflow

#### Create and deploy:

```bash
kubectl create deployment myapp --image=myapp:1.0 --replicas=3
kubectl expose deployment myapp --type=LoadBalancer --port=80 --target-port=8080
kubectl get deployment,service
```

#### Update and rollback:

```bash
kubectl set image deployment/myapp myapp=myapp:2.0 --record
kubectl rollout status deployment/myapp
kubectl rollout undo deployment/myapp
```

### Debugging Workflow

#### Troubleshoot failing pod:

```bash
```

```
kubectl describe pod <pod-name>
kubectl logs <pod-name> --all-containers --previous
kubectl exec -it <pod-name> -- /bin/sh
kubectl get events --field-selector involvedObject.name=<pod-name>
```

### Scale and monitor:

```bash
kubectl scale deployment myapp --replicas=5
kubectl rollout status deployment/myapp
kubectl top pods -n default
```

## Multi-Namespace Management

### Work across namespaces:

```bash
kubectl get pods --all-namespaces
kubectl get pods -n production
kubectl config set-context --current --namespace=production
kubectl create namespace staging
```

### Copy resources between namespaces:

```bash
kubectl get configmap myconfig -o yaml | kubectl apply -n other-ns -f -
kubectl get secret mysecret -o yaml | kubectl apply -n other-ns -f -
```

## Configuration Management Workflow

### Manage configurations:

```bash
kubectl create configmap app-config --from-literal=DATABASE_URL=postgres://db:5432
kubectl create secret generic app-secrets --from-literal=API_KEY=secret123
kubectl get configmap,secret
kubectl describe configmap app-config
```

### Update configurations:

```bash
kubectl edit configmap app-config
kubectl rollout restart deployment/myapp  # Restart pods to pick up changes
```

## Storage Workflow

### Create and manage storage:

```bash
kubectl create pvc my-pvc --size=10Gi --storage-class=fast
kubectl get pvc
kubectl describe pvc my-pvc
kubectl get pv  # Check backing persistent volume
```

## Network Policy Workflow

### Restrict traffic between pods:

```bash
kubectl get networkpolicy
kubectl apply -f network-policy.yaml
kubectl describe networkpolicy <policy-name>
```

## RBAC Setup Workflow

### Create service account with permissions:

```bash
kubectl create serviceaccount myapp
kubectl create role myapp-role --verb=get,list --resource=pods
kubectl create rolebinding myapp-binding --role=myapp-role --serviceaccount=default:myapp
kubectl auth can-i get pods --as=system:serviceaccount:default:myapp
```

## Health Check & Monitoring

### Monitor cluster health:

```bash
```

```
kubectl get nodes
kubectl describe node <node-name>
kubectl top nodes
kubectl get events --all-namespaces --sort-by='.lastTimestamp'
```

**Check resource usage:**

```
bash

kubectl top pods
kubectl top pods --all-namespaces
kubectl describe resourcequota
```

---

## 🎯 21. Best Practices & Tips

### Command Structure Best Practices

**Use long-form flags for clarity:**

```
bash

# Good
kubectl get pods --all-namespaces --selector=app=myapp

# Works but less clear
kubectl get pods -A -l app=myapp
```

**Always specify namespace explicitly:**

```
bash

# Recommended
kubectl get pods -n production
kubectl apply -f deployment.yaml -n production

# Not recommended - relies on default context
kubectl get pods
```

**Use labels for organization:**

```
bash
```

```
kubectl label deployment myapp tier=frontend
kubectl get deployments -l tier=frontend
```

## Efficiency Tips

### Chain commands for quick operations:

```bash
# Get pod IP quickly
kubectl get pod <n> -o jsonpath='{.status.podIP}'

# Get all pod names
kubectl get pods -o jsonpath='{.items[*].metadata.name}'

# Get pods by node
kubectl get pods -o wide --sort-by='.spec.nodeName'
```

### Use context switching efficiently:

```bash
kubectl config get-contexts
kubectl config use-context production
kubectl config current-context
```

### Bulk operations with selectors:

```bash
# Delete all pods with label
kubectl delete pods -l app=myapp

# Scale all deployments
kubectl scale deployment -l app=myapp --replicas=3
```

## Debugging Best Practices

### Gather complete diagnostics:

```bash
```

```
kubectl describe pod <name>
kubectl logs <name>
kubectl get events
kubectl exec -it <name> -- env
```

**Use verbose logging:**

```bash
kubectl get pods -v=8  # Max verbosity
kubectl get pods -v=2  # Debug level
```

**Test connectivity systematically:**

```bash
# From pod to service
kubectl exec <pod> -- wget -O- http://<service>:<port>

# DNS resolution
kubectl exec <pod> -- nslookup <service-name>

# Port checking
kubectl exec <pod> -- netstat -tln | grep <port>
```

---

## 📚 22. Kubectl Configuration Files

### kubeconfig Structure

**Standard kubeconfig locations:**

- `~/.kube/config` — Default kubeconfig on Unix-like systems
- `%USERPROFILE%\.kube\config` — Windows default
- `$KUBECONFIG` — Custom path via environment variable

**kubeconfig components:**

- `clusters` — Kubernetes cluster connection details
- `contexts` — Combinations of cluster, namespace, and user
- `users` — Authentication credentials

- `current-context` — Active context

### Managing Multiple Clusters

#### Merge kubeconfigs:

```bash
export KUBECONFIG=~/.kube/config:~/.kube/prod-config:~/.kube/staging-config
kubectl config view --flatten > ~/.kube/merged-config
```

#### Switch between clusters:

```bash
kubectl config get-contexts
kubectl config use-context production-cluster
kubectl config current-context
```

#### Set context defaults:

```bash
kubectl config set-context production --cluster=prod-cluster --namespace=production
kubectl config set-context --current --namespace=staging
```

---

## 🚀 23. Performance & Optimization

### Query Optimization

#### Efficient resource retrieval:

```bash
# Get only names (fastest)
kubectl get pods -o name

# Avoid describe when you only need specific fields
kubectl get pod <n> -o jsonpath='{.status.phase}'

# Use field selectors instead of filtering in post-processing
kubectl get pods --field-selector=status.phase=Running
```

**Bulk operations:**

```bash
# Better than individual kubectl calls in loops
kubectl delete pods -l app=myapp

# Get all resources at once
kubectl get all -n production
```

### Limiting Query Scope

**Always specify namespace when possible:**

```bash
# Slow - queries all namespaces
kubectl get pods

# Fast - single namespace
kubectl get pods -n production
```

**Use selectors to narrow results:**

```bash
# More efficient than getting all pods
kubectl get pods -l tier=backend
```

---

## 🔗 24. Integration with Other Tools

### JSON & YAML Processing

**Query with jq (JSON processor):**

```bash
kubectl get pods -o json | jq '.items[].metadata.name'
kubectl get svc -o json | jq '.items[] | select(.spec.type=="LoadBalancer")'
```

**Modify YAML with tools:**

```bash
kubectl get deployment myapp -o yaml | yq '.spec.replicas = 5' | kubectl apply -f -
```

## Scripting & Automation

### Create monitoring script:

```bash
#!/bin/bash
while true; do
  kubectl top nodes
  kubectl top pods
  sleep 5
done
```

### Batch pod deletion:

```bash
#!/bin/bash
kubectl delete pods --field-selector=status.phase=Failed --all-namespaces
kubectl delete pods --field-selector=status.phase=Succeeded --all-namespaces
```

## CI/CD Integration

### Apply manifests in pipeline:

```bash
kubectl apply -f manifests/ --record
kubectl rollout status deployment/myapp
kubectl rollout undo deployment/myapp --to-revision=1  # If deployment fails
```

### Validate manifests before deployment:

```bash
kubectl apply -f manifests/ --dry-run=client
kubectl apply -f manifests/ --dry-run=server
```

---

# ⚠️ 25. Common Mistakes & Solutions

## Mistake 1: Wrong Namespace

**Problem:**

```bash
kubectl get pods
# Nothing found, but you know pods exist
```

**Solution:**

```bash
kubectl get pods --all-namespaces
kubectl config set-context --current --namespace=production
kubectl get pods
```

## Mistake 2: Resource Quota Exceeded

**Problem:**

```bash
# Pod fails to create: "Pod failed quota validation"
```

**Solution:**

```bash
kubectl describe resourcequota
kubectl describe namespace production
# Request additional resources or delete unused pods
```

## Mistake 3: Image Pull Errors

**Problem:**

```bash
# Pod stuck in ImagePullBackOff
```

**Solution:**

```bash
kubectl describe pod <pod-name>
# Check image name, registry credentials
kubectl create secret docker-registry regcred \
  --docker-server=registry.example.com \
  --docker-username=user \
  --docker-password=pass
```

## Mistake 4: Incorrect Port Mapping

### Problem:

```bash
# Service won't respond
```

### Solution:

```bash
kubectl describe service <svc-name>
# Check: port, targetPort, selector match pod labels
kubectl get endpoints <svc-name>
# Verify endpoints (backend pods) are listed
```

## Mistake 5: Persistent Volume Not Binding

### Problem:

```bash
# PVC stuck in Pending
```

### Solution:

```bash
kubectl describe pvc <pvc-name>
# Check: storage class exists, PV matches requirements
kubectl get pv,pvc
```

---

# 📖 26. Resource Limits & Quotas Quick Reference

## Setting Resource Limits

### In pod specification:

```yaml
resources:
  requests:
    memory: "64Mi"
    cpu: "250m"
  limits:
    memory: "128Mi"
    cpu: "500m"
```

### Via kubectl:

```bash
kubectl set resources deployment myapp --requests=cpu=100m,memory=128Mi --limits=cpu=200m,memory=256Mi
```

### Viewing resource usage:

```bash
kubectl top nodes
kubectl top pods
kubectl top pod --containers
```

## Namespace Quotas

### Create quota:

```bash
kubectl create quota myquota --hard=pods=10,cpu=5,memory=5Gi
```

### View quotas:

```bash
kubectl describe resourcequota myquota
kubectl get resourcequota
```

## 🎓 27. Learning Resources & Commands Cheat Sheet

### Quick Command Reference

| Task | Command |
|---|---|
| List resources | `kubectl get <resource-type>` |
| Get details | `kubectl describe <resource> <name>` |
| View logs | `kubectl logs <pod>` |
| Execute command | `kubectl exec -it <pod> -- <cmd>` |
| Port forward | `kubectl port-forward <pod> 8080:80` |
| Create resource | `kubectl create <resource> <name>` |
| Edit resource | `kubectl edit <resource> <name>` |
| Delete resource | `kubectl delete <resource> <name>` |
| Apply manifest | `kubectl apply -f <file>` |
| Rollout status | `kubectl rollout status deploy/<name>` |
| Scale deployment | `kubectl scale deploy <name> --replicas=3` |
| Check RBAC | `kubectl auth can-i <verb> <resource>` |
| Get metrics | `kubectl top <resource>` |

### Help & Documentation

- `kubectl --help` — Overview of all commands
- `kubectl <command> --help` — Help for specific command
- `kubectl explain pod` — Explain pod resource structure
- `kubectl explain pod.spec` — Explain specific resource fields
- `kubectl api-resources` — List all available API resources
- `kubectl api-versions` — List API versions supported by cluster

### Official Documentation

- **Official Kubernetes Documentation** — https://kubernetes.io/docs/
- **kubectl Reference** — https://kubernetes.io/docs/reference/kubectl/
- **Kubernetes API Reference** — https://kubernetes.io/docs/reference/generated/kubernetes-api/

- **kubectl Cheat Sheet** — https://kubernetes.io/docs/reference/kubectl/cheatsheet/
- **Kubectl Book** — https://kubectl.docs.kubernetes.io/

---

## 🔑 28. Environment Variables & Shortcuts

### Useful Environment Variables

```bash
# Set default namespace
export KUBECONFIG=~/.kube/config

# Increase command timeout
export KUBECTL_COMMAND_FLAGS="--request-timeout=30s"

# Enable shell completion
source <(kubectl completion bash)
source <(kubectl completion zsh)

# Set default output format
export KUBE_OUTPUT=json
```

### Bash Aliases for Efficiency

```bash
# Add to ~/.bashrc or ~/.zshrc
alias k=kubectl
alias kgp='kubectl get pods'
alias kgd='kubectl get deployment'
alias kgs='kubectl get service'
alias kl='kubectl logs'
alias kex='kubectl exec -it'
alias kd='kubectl describe'
alias ka='kubectl apply -f'
alias kdel='kubectl delete'
alias kgn='kubectl get nodes'
```

### kubectl Completion

#### Enable bash completion:

bash

```
kubectl completion bash | sudo tee /etc/bash_completion.d/kubectl
```

**Enable zsh completion:**

bash

```
kubectl completion zsh | sudo tee /etc/zsh/site-functions/_kubectl
```

---

## 🎯 29. Advanced kubectl Tricks

### Custom Columns for Better Visibility

**Pod overview with custom columns:**

bash

```
kubectl get pods -o custom-columns=NAME:.metadata.name,STATUS:.status.phase,NODE:.spec.nodeName,IP:.status.podIP
```

**Deployment summary:**

bash

```
kubectl get deployment -o custom-columns=NAME:.metadata.name,REPLICAS:.spec.replicas,READY:.status.readyReplicas
```

### Advanced JSONPath Queries

**Complex filtering:**

bash

```
# Pods with requests but no limits
kubectl get pods -o jsonpath='{.items[?(@.spec.containers[*].resources.requests)].metadata.name}'

# Pods using specific image
kubectl get pods -o jsonpath='{.items[?(@.spec.containers[*].image=="nginx:latest")].metadata.name}'

# Failed and pending pods
kubectl get pods -o jsonpath='{.items[?(@.status.phase!="Running")].metadata.name}'
```

### Bulk Updates Without Files

#### Update all deployments:

```bash
kubectl get deployment --all-namespaces -o json | jq '.items[] | .metadata.name' | xargs -I {} kubectl set image deployment/{}
```

#### Add labels to all pods:

```bash
kubectl get pods --all-namespaces -o json | kubectl label --all -f - new-label=value
```

### Efficient Monitoring Loop

#### Watch pod status in real-time:

```bash
watch kubectl get pods
watch kubectl top pods
watch 'kubectl get pods | grep -v Running'
```

---

## 📊 30. Metrics, Monitoring & Observability

### Pod & Node Metrics

#### View resource usage:

```bash
kubectl top nodes
kubectl top nodes --containers=true
kubectl top pods
kubectl top pods --all-namespaces
kubectl top pods -l app=myapp
```

### Events & Logging

#### Comprehensive event viewing:

```bash
kubectl get events --all-namespaces
kubectl get events -n default --sort-by='.lastTimestamp'
kubectl get events --field-selector involvedObject.kind=Pod
kubectl get events --field-selector type=Warning
```

**Log streaming from multiple pods:**

```bash
kubectl logs -l app=myapp --all-containers -f
kubectl logs <pod> -c <container> -f
kubectl logs <pod> --tail=100 -f
```

### Cluster Health Diagnostics

**Comprehensive cluster check:**

```bash
kubectl cluster-info
kubectl cluster-info dump --output-directory=./cluster-dump
kubectl get componentstatuses
kubectl get nodes --show-labels
kubectl get storageclasses
```

---

## 🎓 Conclusion

Kubectl is an incredibly powerful tool with extensive capabilities for managing Kubernetes clusters. Key takeaways:

1. **Master the basics first** — `get`, `describe`, `logs`, `exec`
2. **Use labels and selectors** — Organize and query resources efficiently
3. **Understand contexts and namespaces** — Avoid mistakes with multiple clusters
4. **Leverage formatting options** — Get exactly the information you need
5. **Automate with scripts** — Use shell scripting for repetitive tasks
6. **Stay organized** — Use meaningful names, labels, and namespaces
7. **Monitor continuously** — Use `top`, `events`, and logs for observability

8. **Practice troubleshooting** — Use `describe`, `logs`, and `exec` systematically
9. **Keep security in mind** — Use RBAC, secrets, and proper authentication
10. **Learn from documentation** — kubectl has extensive built-in help

With these commands and best practices, you'll be able to manage, deploy, debug, and maintain Kubernetes clusters effectively and efficiently.