Kubernetes kubectl Cheat Sheet

Kubernetes kubectl Cheat Sheet

A practical quick-reference guide for kubectl commands organized by category. Keep this handy for daily Kubernetes operations.

Cluster Info

Get an overview of your cluster and node status.

```
# Display cluster endpoint and services
kubectl cluster-info

# List all nodes in the cluster
kubectl get nodes

# Show detailed node information
kubectl describe node worker-01

# View node resource usage (requires metrics-server)
kubectl top nodes

# Get node with labels
kubectl get nodes --show-labels

# Filter nodes by label
kubectl get nodes -l node-role.kubernetes.io/control-plane
```

Namespaces

Namespaces provide logical isolation within a cluster.

```
# List all namespaces
kubectl get namespaces

# Create a new namespace
kubectl create namespace staging

# Delete a namespace (and all resources in it)
kubectl delete namespace old-app
```

```
# Set default namespace for current context
kubectl config set-context --current --namespace=staging

# List resources in a specific namespace
kubectl get pods -n production

# List resources across all namespaces
kubectl get pods --all-namespaces
kubectl get pods -A
```

Pods

Pods are the smallest deployable units in Kubernetes.

```
# List pods in current namespace
kubectl get pods

# List pods with more details
kubectl get pods -o wide

# List pods in a specific namespace
kubectl get pods -n production

# List pods across all namespaces
kubectl get pods -A

# Show detailed pod information
kubectl describe pod nginx-deployment-6b7f675859-abc12

# View pod logs
kubectl logs nginx-pod

# Stream logs in real-time
kubectl logs -f nginx-pod

# View logs from a specific container in multi-container pod
kubectl logs nginx-pod -c sidecar

# View logs from previous container instance (after restart)
kubectl logs nginx-pod --previous

# Execute command inside a pod
kubectl exec nginx-pod -- ls /app

# Get an interactive shell inside a pod
kubectl exec -it nginx-pod -- /bin/bash

# Execute command in a specific container
kubectl exec nginx-pod -c main-container -- cat /etc/config/app.conf

# Port forward local traffic to a pod
kubectl port-forward nginx-pod 8080:80

# Port forward to a specific namespace
kubectl port-forward nginx-pod 8080:80 -n production
```

```
# Copy file from local to pod
kubectl cp ./local-file.txt nginx-pod:/remote/path/file.txt

# Copy file from pod to local
kubectl cp nginx-pod:/remote/path/file.txt ./local-file.txt

# Copy from specific container in pod
kubectl cp nginx-pod:/app/config.yaml ./config.yaml -c sidecar

# Delete a pod
kubectl delete pod nginx-pod

# Delete pod immediately (bypass graceful termination)
kubectl delete pod nginx-pod --force --grace-period=0

# Delete all pods in a namespace
kubectl delete pods --all -n staging
```

Deployments

Deployments manage stateless applications with ReplicaSets.

```
# Create deployment from image
kubectl create deployment nginx --image=nginx:1.21

# Create deployment with replicas
kubectl create deployment nginx --image=nginx:1.21 --replicas=3

# List deployments
kubectl get deployments

# Show detailed deployment info
kubectl describe deployment nginx

# Scale deployment
kubectl scale deployment nginx --replicas=5

# Autoscale deployment (requires metrics-server)
kubectl autoscale deployment nginx --min=2 --max=10 --cpu-percent=80

# View rollout status
kubectl rollout status deployment/nginx

# View rollout history
kubectl rollout history deployment/nginx

# Rollback to previous version
kubectl rollout undo deployment/nginx

# Rollback to specific revision
kubectl rollout undo deployment/nginx --to-revision=2

# Update container image
kubectl set image deployment/nginx nginx=nginx:1.22

# Set environment variable
kubectl set env deployment/nginx APP_ENV=production
```

```
# Edit deployment (opens in editor)
kubectl edit deployment nginx

# Delete deployment
kubectl delete deployment nginx
```

Services

Services expose applications running in pods.

```
# List services
kubectl get services

# Show detailed service info
kubectl describe service nginx-service

# Expose deployment as a service (ClusterIP)
kubectl expose deployment nginx --port=80 --target-port=80

# Expose as NodePort
kubectl expose deployment nginx --type=NodePort --port=80

# Expose as LoadBalancer (for cloud providers)
kubectl expose deployment nginx --type=LoadBalancer --port=80

# Create service from YAML
kubectl apply -f service.yaml

# Delete service
kubectl delete service nginx-service

# Get service cluster IP
kubectl get service nginx-service -o jsonpath='{.spec.clusterIP}'
```

ConfigMaps

ConfigMaps store non-sensitive configuration data.

```
# Create configmap from literal values
kubectl create configmap app-config --from-literal=API_KEY=abc123 --from-literal=DB_HOST=db.e

# Create configmap from file
kubectl create configmap app-config --from-file=config.properties

# Create configmap from directory
kubectl create configmap app-config --from-file=./config/

# Create configmap from env file
kubectl create configmap app-config --from-env-file=app.env

# List configmaps
kubectl get configmaps

# Show detailed configmap info
```

```
kubectl describe configmap app-config

# Get configmap data as YAML
kubectl get configmap app-config -o yaml

# Edit configmap
kubectl edit configmap app-config

# Delete configmap
kubectl delete configmap app-config
```

Secrets

Secrets store sensitive data encoded in base64.

```
# Create secret from literal values
kubectl create secret generic db-credentials --from-literal=username=admin --from-literal=pas

# Create secret from file
kubectl create secret generic tls-cert --from-file=tls.crt=./cert.pem --from-file=tls.key=./k

# Create docker registry secret
kubectl create secret docker-registry regcred --docker-server=registry.example.com --docker-u

# Create TLS secret from files
kubectl create secret tls my-tls-secret --cert=path/to/cert.crt --key=path/to/cert.key

# List secrets
kubectl get secrets

# Show detailed secret info (data is base64 encoded)
kubectl describe secret db-credentials

# View decoded secret data
kubectl get secret db-credentials -o jsonpath='{.data.password}' | base64 --decode

# Edit secret
kubectl edit secret db-credentials

# Delete secret
kubectl delete secret db-credentials
```

Ingress

Ingress exposes HTTP/HTTPS routes from outside the cluster.

```
# List ingress resources
kubectl get ingress

# List ingress in all namespaces
kubectl get ingress -A

# Show detailed ingress info
kubectl describe ingress my-ingress
```

```
# Create ingress from YAML
kubectl apply -f ingress.yaml

# Get ingress load balancer IP
kubectl get ingress my-ingress -o jsonpath='{.status.loadBalancer.ingress[0].ip}'

# Delete ingress
kubectl delete ingress my-ingress
```

Persistent Volumes

Persistent storage for stateful applications.

```
# List persistent volume claims
kubectl get pvc

# List persistent volumes
kubectl get pv

# Show detailed PVC info
kubectl describe pvc my-pvc

# Show detailed PV info
kubectl describe pv pvc-abc123

# Create PVC from YAML
kubectl apply -f pvc.yaml

# Delete PVC
kubectl delete pvc my-pvc

# View storage classes
kubectl get storageclass

# Get PVC bound volume
kubectl get pvc my-pvc -o jsonpath='{.spec.volumeName}'
```

RBAC

Role-Based Access Control for cluster security.

```
# List roles in namespace
kubectl get roles

# List role bindings in namespace
kubectl get rolebindings

# List cluster roles (cluster-wide)
kubectl get clusterroles

# List cluster role bindings
kubectl get clusterrolebindings

# Show detailed role info
kubectl describe role pod-reader -n production
```

```
# Create role from YAML
kubectl apply -f role.yaml

# Create role binding
kubectl create rolebinding pod-reader-binding --role=pod-reader --serviceaccount=production:d

# Create cluster role binding for user
kubectl create clusterrolebinding admin-binding --clusterrole=admin --user=john@example.com

# List service accounts
kubectl get serviceaccounts

# Create service account
kubectl create serviceaccount my-app

# Delete role binding
kubectl delete rolebinding pod-reader-binding -n production
```

Context and Config

Manage multiple clusters and configurations.

```
# View current config
kubectl config view

# List all contexts
kubectl config get-contexts

# Show current context
kubectl config current-context

# Switch context
kubectl config use-context production-cluster

# Set context with namespace
kubectl config set-context dev --cluster=dev-cluster --namespace=development --user=dev-user

# Delete context
kubectl config delete-context old-cluster

# List clusters in config
kubectl config get-clusters

# List users in config
kubectl config get-users

# Set credentials
kubectl config set-credentials john --username=john --password=secret

# Set cluster entry
kubectl config set-cluster my-cluster --server=https://my-cluster.example.com --insecure-skip
```

Debugging

Troubleshoot issues in your cluster.

```
# Describe any resource
kubectl describe pod problematic-pod

# View pod events
kubectl get events --field-selector involvedObject.name=problematic-pod

# View all events in namespace sorted by time
kubectl get events --sort-by='.lastTimestamp'

# View events across all namespaces
kubectl get events -A

# Check pod resource usage
kubectl top pod

# Check pod resource usage in specific namespace
kubectl top pod -n production

# Check container logs with timestamps
kubectl logs nginx-pod --timestamps

# View logs from last hour
kubectl logs nginx-pod --since=1h

# View last 100 lines of logs
kubectl logs nginx-pod --tail=100

# Run a debug container
kubectl debug nginx-pod -it --image=busybox

# Run debug container on node
kubectl debug node/worker-01 -it --image=ubuntu

# Get pod resource specifications
kubectl get pod nginx-pod -o yaml

# Check pod status conditions
kubectl get pod nginx-pod -o jsonpath='{.status.conditions[*].message}'

# Check why a pod is not scheduling
kubectl describe pod nginx-pod | grep -A 10 Events
```

Resource Management

Create, update, and delete Kubernetes resources.

```
# Apply configuration from file
kubectl apply -f deployment.yaml

# Apply all manifests in directory
kubectl apply -f ./k8s/

# Apply from URL
kubectl apply -f https://raw.githubusercontent.com/org/repo/main/deployment.yaml
```

```
# Delete resource from file
kubectl delete -f deployment.yaml

# Delete resources by label
kubectl delete pods -l app=old-app

# Delete all resources in file
kubectl delete -f ./k8s/

# Edit a resource live
kubectl edit deployment nginx

# Patch a resource
kubectl patch deployment nginx -p '{"spec":{"replicas":3}}'

# Patch with strategic merge
kubectl patch deployment nginx --type='strategic' -p '{"spec":{"template":{"spec":{"container

# Replace resource (complete replacement)
kubectl replace -f deployment.yaml

# Force replace (delete and recreate)
kubectl replace --force -f deployment.yaml

# Wait for resource to be ready
kubectl wait --for=condition=available --timeout=600s deployment/nginx

# Wait for pod to be deleted
kubectl wait --for=delete pod/temporary-pod --timeout=60s
```

Output Formats

Control output format for better readability or scripting.

```
# YAML output
kubectl get pod nginx-pod -o yaml

# JSON output
kubectl get pod nginx-pod -o json

# Wide output (more columns)
kubectl get pods -o wide

# Custom columns
kubectl get pods -o custom-columns=NAME:.metadata.name,IMAGE:.spec.containers[0].image,NODE:.

# Custom columns from file
kubectl get pods -o custom-columns-file=columns.txt

# Sort by field
kubectl get pods --sort-by=.metadata.creationTimestamp

# Show labels
kubectl get pods --show-labels

# Label columns
```

```
kubectl get pods -L app,environment

# JSONPath output
kubectl get pods -o jsonpath='{.items[*].metadata.name}'

# JSONPath with newlines
kubectl get pods -o jsonpath='{range .items[*]}{.metadata.name}{"\n"}{end}'

# Get specific field
kubectl get pod nginx-pod -o jsonpath='{.spec.containers[0].image}'

# Name only
kubectl get pods -o name

# No headers
kubectl get pods --no-headers

# Combine formats
kubectl get pods -o wide --no-headers | awk '{print $1}'
```

Quick Reference Flags

Common flags used across multiple commands.

Flag	Description
`-n, --namespace`	Target namespace
`-A, --all-namespaces`	All namespaces
`-o, --output`	Output format (yaml, json, wide, name)
`-f, --filename`	Resource file or directory
`-l, --selector`	Label selector
`--dry-run`	Simulate without applying (client, server)
`--force`	Force operation
`--grace-period`	Grace period before force delete
`--timeout`	Operation timeout
`-w, --watch`	Watch for changes
`--show-labels`	Show labels in output
`--no-headers`	Omit column headers

Common Workflows

Quick Deployment Check

```
kubectl get deploy,po,svc -l app=myapp
```

Debug Failing Pod

```
kubectl describe pod failing-pod | grep -A 20 Events
kubectl logs failing-pod --previous
kubectl get events --field-selector involvedObject.name=failing-pod
```

Resource Cleanup

```
kubectl delete pods,svc -l app=old-app
kubectl delete pvc --all -n old-namespace
kubectl delete namespace old-namespace
```

Quick Pod Shell Access

```
kubectl exec -it $(kubectl get pod -l app=myapp -o jsonpath='{.items[0].metadata.name}') -- /
```

Copy Config from Pod

```
kubectl cp $(kubectl get pod -l app=myapp -o jsonpath='{.items[0].metadata.name}'):/etc/confi
```

Next Steps

- Helm CLI Cheat Sheet — Kubernetes package manager commands
- Docker CLI Cheat Sheet — Build and manage container images
- Terraform Cheat Sheet — Infrastructure as code reference
- Kubernetes GitOps with ArgoCD — Declarative continuous deployment