Kubernetes Cheat Sheet 2025

For DevOps & SRE Engineers

Kubectl Basics & Setup

Install kubectl

```
# Linux
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

Check Version & Cluster Info

```
kubectl version --client
kubectl cluster-info
```

Context & Configuration

```
kubectl config get-contexts                 # List all contexts
kubectl config use-context <context-name>   # Switch context
kubectl config current-context              # Show current context
```

Namespace Management

Create a Namespace

kubectl create namespace staging

List Namespaces

kubectl get namespaces

Set Default Namespace

kubectl config set-context --current --namespace=staging

Delete Namespace

kubectl delete namespace staging

Pods

Create a Pod

```yaml
# pod.yml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
  - name: nginx
    image: nginx:1.25
    ports:
    - containerPort: 80
```

Pod Commands

```
kubectl get pods
kubectl describe pod nginx-pod
kubectl logs nginx-pod
kubectl delete pod nginx-pod
```

Deployments

Create a Deployment

```yaml
# deployment.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.25
        ports:
        - containerPort: 80
```

kubectl apply -f deployment.yml

Deployment Commands

```bash
kubectl get deployments
kubectl rollout status deployment/nginx-deployment
kubectl scale deployment nginx-deployment --replicas=5
```

Services

Create a Service

```yaml
# service.yml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```

kubectl apply -f service.yml

Service Types

- ClusterIP (default)
- NodePort
- LoadBalancer
- ExternalName

<u>Commands</u>

kubectl get services
kubectl describe service nginx-service

ConfigMaps & Secrets

Create a ConfigMap

kubectl create configmap app-config --from-literal=DB_HOST=mysql-db

Use ConfigMap in Pod

```yaml
env:
  - name: DB_HOST
    valueFrom:
      configMapKeyRef:
        name: app-config
        key: DB_HOST
```

Create a Secret

kubectl create secret generic db-secret --from-literal=password=mysecret123

Use Secret in Pod

```yaml
env:
  - name: DB_PASSWORD
    valueFrom:
      secretKeyRef:
        name: db-secret
        key: password
```

Volumes & Persistent Storage

PersistentVolumeClaim (PVC)

```yaml
# pvc.yml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
```

kubectl apply -f pvc.yml

Mount PVC in Pod

```yaml
volumes:
  - name: storage
    persistentVolumeClaim:
      claimName: my-pvc
containers:
  - volumeMounts:
    - mountPath: "/data"
      name: storage
```

StatefulSets

StatefulSet Example

```yaml
# statefulset.yml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  serviceName: "nginx"
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.25
        ports:
        - containerPort: 80
```

kubectl apply -f statefulset.yml

DaemonSets & Jobs

DaemonSet

```yaml
# daemonset.yml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd
spec:
  selector:
    matchLabels:
      name: fluentd
  template:
    metadata:
      labels:
        name: fluentd
    spec:
      containers:
      - name: fluentd
        image: fluentd:latest
```

Job

```yaml
# job.yml
apiVersion: batch/v1
kind: Job
metadata:
  name: hello-job
spec:
  template:
    spec:
      containers:
      - name: hello
        image: busybox
        command: ["echo", "Hello Kubernetes!"]
      restartPolicy: Never
```

Networking & Ingress

Ingress Example

```yaml
# ingress.yml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
spec:
  rules:
  - host: myapp.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: nginx-service
            port:
              number: 80
```

kubectl apply -f ingress.yml

Network Policies

```yaml
# network-policy.yml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-all
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  - Egress
```

Resource Management

Resource Limits

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

Check Resource Usage

kubectl top pods
kubectl top nodes

Debugging & Logs

Logs

kubectl logs <pod-name>
kubectl logs -f <pod-name> # Follow logs
kubectl logs --previous <pod-name> # Previous container

Exec into Pod

kubectl exec -it <pod-name> -- /bin/bash

Describe Resources

kubectl describe pod <pod-name>
kubectl describe node <node-name>

Rollouts & Rollbacks

Rollout Status

kubectl rollout status deployment/nginx-deployment

Rollback

kubectl rollout undo deployment/nginx-deployment

History

kubectl rollout history deployment/nginx-deployment

Helm Basics

Install Helm

curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

Helm Commands

```
helm repo add bitnami https://charts.bitnami.com/bitnami
helm install my-nginx bitnami/nginx
helm list
helm uninstall my-nginx
```

Security (RBAC & ServiceAccounts)

ServiceAccount

```
# serviceaccount.yml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: my-serviceaccount
```

Role & RoleBinding

```yaml
# role.yml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "list"]
---
# rolebinding.yml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods
subjects:
- kind: ServiceAccount
  name: my-serviceaccount
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

Monitoring & Observability

Metrics Server

kubectl top nodes
kubectl top pods

Prometheus & Grafana

helm install prometheus prometheus-community/prometheus
helm install grafana grafana/grafana

Custom Resources (CRDs) & Operators

CustomResourceDefinition (CRD)

```yaml
# crd.yml
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: myresources.example.com
spec:
  group: example.com
  versions:
    - name: v1
      served: true
      storage: true
  scope: Namespaced
  names:
    plural: myresources
    singular: myresource
    kind: MyResource
```

Troubleshooting Commands

<u>Common Issues</u>

Check events
kubectl get events --sort-by=.metadata.creationTimestamp

Check node status
kubectl get nodes -o wide

Check pod issues
kubectl describe pod <pod-name>

Check service endpoints
kubectl get endpoints <service-name>

Best Practices & Tips

Best Practices

- Use namespaces for isolation
- Set resource limits
- Use liveness and readiness probes
- Avoid using latest tag for images
- Use Helm for package management

Useful Aliases

```
alias k='kubectl'
alias kgp='kubectl get pods'
alias kgs='kubectl get services'
alias kd='kubectl describe'
```

Documentation & Help

```
kubectl explain pod.spec.containers
kubectl --help
```