KUBERNETES 快速参考

kubectl、Pod、Deployment、Service、配置、调试

kubectl基础

集群信息

```
kubectl cluster-info
kubectl get nodes
kubectl config current-context
kubectl config use-context my-cluster
```

核心命令

`kubectl get <resource>`	列出资源
`kubectl describe <resource> <name>`	查看资源详情
`kubectl create -f file.yaml`	从文件创建资源
`kubectl apply -f file.yaml`	创建或更新资源
`kubectl delete -f file.yaml`	从文件删除资源
`kubectl edit <resource> <name>`	就地编辑资源
`kubectl api-resources`	列出所有资源类型

输出格式

`-o wide`	额外列（IP、节点）
`-o yaml`	完整 YAML 输出
`-o json`	完整 JSON 输出
`-o jsonpath='{.spec}'`	提取指定字段
`--sort-by=.metadata.name`	按字段排序输出

Pod

Pod 操作

```
kubectl get pods
kubectl get pods -A                 # all namespaces
kubectl run nginx --image=nginx     # quick pod
kubectl delete pod nginx
```

Pod YAML

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp
  labels: { app: myapp }
spec:
  containers:
    - name: app
      image: nginx:1.27
      ports:
        - containerPort: 80
```

Pod 状态值

`Running`	所有容器已启动
`Pending`	等待调度或镜像拉取
`CrashLoopBackOff`	容器持续崩溃并重启
`ImagePullBackOff`	无法拉取容器镜像
`Completed`	已运行完成（Job）

Deployment

Deployment YAML

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 3
  selector:
    matchLabels: { app: web }
  template:
    metadata:
      labels: { app: web }
    spec:
      containers:
        - name: web
          image: nginx:1.27
          ports:
            - containerPort: 80
```

Deployment 命令

`kubectl get deploy`	列出 Deployment
`kubectl scale deploy web --replicas=5`	调整副本数
`kubectl set image deploy/web web=nginx:1.28`	滚动更新镜像
`kubectl rollout status deploy/web`	查看发布进度
`kubectl rollout undo deploy/web`	回滚到上一个版本
`kubectl rollout history deploy/web`	查看版本历史

Service

Service 类型

`ClusterIP`	仅集群内部访问（默认）
`NodePort`	在每个节点 IP 上开放静态端口
`LoadBalancer`	外部负载均衡器（云环境）
`ExternalName`	指向外部服务的 DNS 别名

Service YAML

```
apiVersion: v1
kind: Service
metadata:
  name: web-svc
spec:
  type: ClusterIP
  selector: { app: web }
  ports:
    - port: 80
      targetPort: 80
```

快速暴露

```
kubectl expose deploy web --port=80 --type=ClusterIP
kubectl expose deploy web --port=80 --type=NodePort
kubectl get svc
```

ConfigMap 与 Secret

ConfigMap

```
kubectl create configmap app-cfg \
  --from-literal=DB_HOST=db.example.com \
  --from-file=config.ini
```

Secret

```
kubectl create secret generic db-creds \
  --from-literal=username=admin \
  --from-literal=password=s3cret
```

在 Pod 中使用

```
# As environment variables
envFrom:
  - configMapRef: { name: app-cfg }
  - secretRef:    { name: db-creds }

# As volume mount
volumes:
  - name: cfg
    configMap: { name: app-cfg }
```

相关命令

`kubectl get cm`	列出 ConfigMap
`kubectl get secret`	列出 Secret
`kubectl describe cm app-cfg`	查看 ConfigMap 数据
`kubectl get secret db-creds -o yaml`	查看 Secret（base64 编码）

命名空间

命名空间命令

`kubectl get ns`	列出命名空间
`kubectl create ns staging`	创建命名空间
`kubectl delete ns staging`	删除命名空间及其所有资源
`kubectl get pods -n staging`	列出命名空间中的 Pod
`kubectl get pods -A`	列出所有命名空间的 Pod

设置默认命名空间

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

存储卷

PersistentVolumeClaim

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: data-pvc
spec:
  accessModes: [ReadWriteOnce]
  resources:
    requests: { storage: 10Gi }
```

挂载到 Pod

```
volumes:
  - name: data
    persistentVolumeClaim:
      claimName: data-pvc
containers:
  - volumeMounts:
      - name: data
        mountPath: /app/data
```

卷类型

`emptyDir`	临时目录，Pod 删除后消失
`hostPath`	挂载宿主机文件系统路径
`persistentVolumeClaim`	持久存储（PVC）
`configMap`	将 ConfigMap 挂载为文件
`secret`	将 Secret 挂载为文件

Ingress

Ingress YAML

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: web-ingress
spec:
  rules:
    - host: app.example.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: web-svc
                port: { number: 80 }
```

Ingress 说明

`Ingress Controller`	必须安装（nginx-ingress、traefik 等）
`pathType: Prefix`	匹配 URL 前缀
`pathType: Exact`	精确匹配 URL 路径
`TLS`	添加 `tls:` 节并指定 Secret 名称

调试

诊断命令

`kubectl logs <pod>`	容器标准输出/标准错误
`kubectl logs <pod> -c <ctr>`	指定容器的日志
`kubectl logs <pod> --previous`	已崩溃容器的日志
`kubectl describe pod <pod>`	事件、条件、状态
`kubectl exec -it <pod> -- sh`	进入容器 Shell
`kubectl port-forward <pod> 8080:80`	将本地端口转发到 Pod
`kubectl top pods`	CPU/内存使用（需 metrics-server）
`kubectl get events --sort-by=.lastTimestamp`	集群事件时间线

调试 Pod

```
kubectl run debug --rm -it --image=busybox -- sh
# or attach ephemeral container
kubectl debug -it <pod> --image=busybox
```

常见模式

标签与选择器

```
kubectl get pods -l app=web
kubectl get pods -l 'env in (prod,staging)'
kubectl label pod myapp env=prod
```

资源限制

```
resources:
  requests: { cpu: 100m, memory: 128Mi }
  limits:   { cpu: 500m, memory: 256Mi }
```

存活探针与就绪探针

```
livenessProbe:
  httpGet: { path: /healthz, port: 8080 }
  initialDelaySeconds: 5
  periodSeconds: 10
readinessProbe:
  httpGet: { path: /ready, port: 8080 }
```

快速技巧

`Dry run`	`kubectl apply -f file.yaml --dry-run=client`
`Generate YAML`	`kubectl create deploy web --image=nginx --dry-run=client -o yaml`
`Watch`	`kubectl get pods -w`
`Copy files`	`kubectl cp file.txt pod:/tmp/`
`Restart deploy`	`kubectl rollout restart deploy/web`