Referência Rápida de Kubernetes

kubectl, pods, deployments, services, configs, depuração

Básico do kubectl

Informações do Cluster

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

Comandos Essenciais

`kubectl get <resource>`	Listar recursos
`kubectl describe <resource> <name>`	Informações detalhadas do recurso
`kubectl create -f file.yaml`	Criar recurso a partir de arquivo
`kubectl apply -f file.yaml`	Criar ou atualizar recurso
`kubectl delete -f file.yaml`	Excluir recurso a partir de arquivo
`kubectl edit <resource> <name>`	Editar recurso no local
`kubectl api-resources`	Listar todos os tipos de recurso

Formatos de Saída

`-o wide`	Colunas extras (IP, nó)
`-o yaml`	Saída YAML completa
`-o json`	Saída JSON completa
`-o jsonpath='{.spec}'`	Extrair campos específicos
`--sort-by=.metadata.name`	Ordenar saída por campo

Pods

Operações com Pods

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

YAML de Pod

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

Valores de Status de Pod

`Running`	Todos os containers iniciados
`Pending`	Aguardando agendamento ou pull da imagem
`CrashLoopBackOff`	Container fica quebrando e reiniciando
`ImagePullBackOff`	Não é possível fazer pull da imagem do container
`Completed`	Executou até a conclusão (Jobs)

Deployments

YAML de Deployment

```
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
```

Comandos de Deployment

`kubectl get deploy`	Listar deployments
`kubectl scale deploy web --replicas=5`	Escalar réplicas
`kubectl set image deploy/web web=nginx:1.28`	Atualizar imagem (rolling)
`kubectl rollout status deploy/web`	Monitorar progresso do rollout
`kubectl rollout undo deploy/web`	Reverter para revisão anterior
`kubectl rollout history deploy/web`	Ver histórico de revisões

Services

Tipos de Service

`ClusterIP`	Somente interno (padrão)
`NodePort`	Expõe no IP de cada nó em porta estática
`LoadBalancer`	Load balancer externo (nuvem)
`ExternalName`	Alias DNS para serviço externo

YAML de Service

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

Expose Rápido

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

ConfigMaps e Secrets

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
```

Usando em Pods

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

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

Comandos

`kubectl get cm`	Listar ConfigMaps
`kubectl get secret`	Listar Secrets
`kubectl describe cm app-cfg`	Exibir dados do ConfigMap
`kubectl get secret db-creds -o yaml`	Exibir Secret (codificado em base64)

Namespaces

Comandos de Namespace

`kubectl get ns`	Listar namespaces
`kubectl create ns staging`	Criar namespace
`kubectl delete ns staging`	Excluir namespace e todos os recursos
`kubectl get pods -n staging`	Listar pods no namespace
`kubectl get pods -A`	Listar pods em todos os namespaces

Definir Namespace Padrão

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

Volumes

PersistentVolumeClaim

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

Montar em Pod

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

Tipos de Volume

`emptyDir`	Diretório temporário, excluído com o pod
`hostPath`	Monta caminho do sistema de arquivos do host
`persistentVolumeClaim`	Armazenamento persistente (PVC)
`configMap`	Monta ConfigMap como arquivos
`secret`	Monta Secret como arquivos

Referência Rápida de Kubernetes

Ingress

YAML de Ingress

```
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 }
```

Notas sobre Ingress

`Ingress Controller`	Necessário (nginx-ingress, traefik, etc.)
`pathType: Prefix`	Corresponde ao prefixo da URL
`pathType: Exact`	Corresponde ao caminho exato da URL
`TLS`	Adicione seção **`tls:`** com nome do secret

Depuração

Comandos de Diagnóstico

`kubectl logs <pod>`	stdout/stderr do container
`kubectl logs <pod> -c <ctr>`	Logs de container específico
`kubectl logs <pod> --previous`	Logs de container que falhou
`kubectl describe pod <pod>`	Eventos, condições, status
`kubectl exec -it <pod> -- sh`	Shell no container
`kubectl port-forward <pod> 8080:80`	Encaminhar porta local para o pod
`kubectl top pods`	Uso de CPU/ memória (metrics-server)
`kubectl get events --sort-by=.lastTimestamp`	Linha do tempo de eventos do cluster

Pod de Depuração

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

Padrões Comuns

Labels e Seletores

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

Limites de Recursos

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

Liveness e Readiness

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

Receitas Rápidas

`Dry run`	**`kubectl apply -f file.yaml --dry-run=client`**
`Gerar YAML`	**`kubectl create deploy web --image=nginx --dry-run=client -o yaml`**
`Watch`	**`kubectl get pods -w`**
`Copiar arquivos`	**`kubectl cp file.txt pod:/tmp/`**
`Reiniciar deploy`	**`kubectl rollout restart deploy/web`**