RIFERIMENTO RAPIDO KUBERNETES

kubectl, pod, deployment, servizi, config, debug

Basi kubectl

Info Cluster

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

Comandi Essenziali

`kubectl get <resource>`	Elenca risorse
`kubectl describe <resource> <name>`	Info dettagliate sulla risorsa
`kubectl create -f file.yaml`	Crea risorsa da file
`kubectl apply -f file.yaml`	Crea o aggiorna risorsa
`kubectl delete -f file.yaml`	Elimina risorsa da file
`kubectl edit <resource> <name>`	Modifica risorsa in-place
`kubectl api-resources`	Elenca tutti i tipi di risorsa

Formati di Output

`-o wide`	Colonne extra (IP, nodo)
`-o yaml`	Output YAML completo
`-o json`	Output JSON completo
`-o jsonpath='{.spec}'`	Estrae campi specifici
`--sort-by=.metadata.name`	Ordina output per campo

Pod

Operazioni sui Pod

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

YAML Pod

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

Valori di Stato Pod

`Running`	Tutti i container avviati
`Pending`	In attesa di scheduling o pull immagine
`CrashLoopBackOff`	Il container continua a crashare e riavviarsi
`ImagePullBackOff`	Impossibile scaricare l'immagine container
`Completed`	Eseguito fino al completamento (Job)

Deployment

YAML 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
```

Comandi Deployment

`kubectl get deploy`	Elenca deployment
`kubectl scale deploy web --replicas=5`	Scala le repliche
`kubectl set image deploy/web web=nginx:1.28`	Aggiorna immagine (rolling)
`kubectl rollout status deploy/web`	Monitora l'avanzamento del rollout
`kubectl rollout undo deploy/web`	Rollback alla revisione precedente
`kubectl rollout history deploy/web`	Visualizza cronologia revisioni

Servizi

Tipi di Servizio

`ClusterIP`	Solo interno (default)
`NodePort`	Espone su ogni IP del nodo a porta statica
`LoadBalancer`	Load balancer esterno (cloud)
`ExternalName`	Alias DNS a servizio esterno

YAML Servizio

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

Esposizione Rapida

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

ConfigMap e 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
```

Usare nei Pod

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

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

Comandi

`kubectl get cm`	Elenca ConfigMap
`kubectl get secret`	Elenca Secret
`kubectl describe cm app-cfg`	Mostra dati ConfigMap
`kubectl get secret db-creds -o yaml`	Mostra Secret (codificato base64)

Namespace

Comandi Namespace

`kubectl get ns`	Elenca namespace
`kubectl create ns staging`	Crea namespace
`kubectl delete ns staging`	Elimina namespace e tutte le risorse
`kubectl get pods -n staging`	Elenca pod nel namespace
`kubectl get pods -A`	Elenca pod in tutti i namespace

Imposta Namespace Default

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

Volumi

PersistentVolumeClaim

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

Montare nel Pod

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

Tipi di Volume

`emptyDir`	Dir temporanea, eliminata con il pod
`hostPath`	Monta percorso del filesystem host
`persistentVolumeClaim`	Storage persistente (PVC)
`configMap`	Monta ConfigMap come file
`secret`	Monta Secret come file

Ingress

YAML 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 }
```

Note Ingress

`Ingress Controller`	Richiesto (nginx-ingress, traefik, ecc.)
`pathType: Prefix`	Corrisponde al prefisso URL
`pathType: Exact`	Corrisponde al percorso URL esatto
`TLS`	Aggiunge sezione `tls:` con nome secret

Debug

Comandi Diagnostici

`kubectl logs <pod>`	stdout/stderr del container
`kubectl logs <pod> -c <ctr>`	Log di un container specifico
`kubectl logs <pod> --previous`	Log dal container crashato
`kubectl describe pod <pod>`	Eventi, condizioni, stato
`kubectl exec -it <pod> -- sh`	Shell nel container
`kubectl port-forward <pod> 8080:80`	Inoltro porta locale al pod
`kubectl top pods`	Utilizzo CPU/memoria (metrics-server)
`kubectl get events --sort-by=.lastTimestamp`	Timeline eventi del cluster

Pod di Debug

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

Pattern Comuni

Label e Selettori

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

Limiti di Risorse

```
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 }
```

Ricette Rapide

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