

Comandi Kubernetes essenziali

Tag: Kubernetes

Ultimo aggiornamento: 2026-09-18

Comandi Kubernetes essenziali e controllo salute del cluster

Cheat sheet operativo con esempi e procedura di health check

Ambiente di riferimento: Kubernetes gestito con kubectl/Helm. Eseguire sempre i comandi dal contesto kubeconfig corretto.

1. Contesto e informazioni del cluster

```
kubectl config current-context
kubectl config get-contexts
kubectl cluster-info
kubectl version
kubectl api-resources
```

2. Nodi

```
kubectl get nodes
kubectl get nodes -o wide
kubectl describe node <NODE>
kubectl top nodes
```

Ready indica che il nodo è utilizzabile dal cluster. describe mostra condizioni, taint, capacità, allocazioni ed eventi.

3. Pod

```
kubectl get pods -A
kubectl get pods -A -o wide
kubectl get pods -n <NAMESPACE>
kubectl describe pod <POD> -n <NAMESPACE>
kubectl logs <POD> -n <NAMESPACE>
kubectl logs <POD> -n <NAMESPACE> --previous
kubectl logs -f <POD> -n <NAMESPACE>
kubectl exec -it <POD> -n <NAMESPACE> -- /bin/sh
```

Per pod con più container aggiungere -c <CONTAINER> ai comandi logs/exec.

4. Deployment, ReplicaSet, DaemonSet e StatefulSet

```
kubectl get deployments -A
kubectl get rs -A
kubectl get daemonsets -A
```

```
kubectl get statefulsets -A
kubectl describe deployment <DEPLOYMENT> -n <NAMESPACE>
kubectl rollout status deployment/<DEPLOYMENT> -n <NAMESPACE>
kubectl rollout history deployment/<DEPLOYMENT> -n <NAMESPACE>
kubectl rollout undo deployment/<DEPLOYMENT> -n <NAMESPACE>
```

5. Service, Endpoint e rete

```
kubectl get svc -A
kubectl get endpoints -A
kubectl get endpointslices -A
kubectl describe svc <SERVICE> -n <NAMESPACE>
kubectl get networkpolicy -A
```

Un Service senza endpoint utili non ha backend verso cui inoltrare il traffico; verificare selector e pod Ready.

6. Ingress e Gateway API

```
kubectl get ingress -A
kubectl describe ingress <INGRESS> -n <NAMESPACE>
kubectl get gatewayclass
kubectl get gateway -A
kubectl get httproute -A
kubectl describe gateway <GATEWAY> -n <NAMESPACE>
kubectl describe httproute <HTTPROUTE> -n <NAMESPACE>
```

7. ConfigMap, Secret e ServiceAccount

```
kubectl get configmap -A
kubectl get secret -A
kubectl get serviceaccount -A
kubectl describe configmap <NAME> -n <NAMESPACE>
kubectl get configmap <NAME> -n <NAMESPACE> -o yaml
```

Attenzione: i Secret possono contenere materiale sensibile. Evitare di stamparne o condividere il contenuto se non necessario.

8. Namespace e risorse

```
kubectl get namespaces
kubectl get all -n <NAMESPACE>
kubectl get all -A
kubectl get events -A --sort-by=.lastTimestamp
```

9. Storage

```
kubectl get pv
kubectl get pvc -A
kubectl get storageclass
kubectl describe pvc <PVC> -n <NAMESPACE>
```

10. CRD e risorse custom

```
kubectl get crd
kubectl get crd | grep gateway.networking.k8s.io
kubectl explain <RESOURCE>
kubectl api-resources | grep -i <TERM>
```

11. Applicare e verificare manifest

```
kubectl diff -f manifest.yaml
kubectl apply --dry-run=server -f manifest.yaml
kubectl apply -f manifest.yaml
kubectl get -f manifest.yaml
kubectl delete -f manifest.yaml
```

Per cambiamenti importanti, diff e dry-run server-side permettono di individuare molti problemi prima della modifica reale.

12. Diagnostica

```
kubectl describe pod <POD> -n <NAMESPACE>
kubectl logs <POD> -n <NAMESPACE> --tail=200
kubectl get events -n <NAMESPACE> --sort-by=.lastTimestamp
kubectl get pod <POD> -n <NAMESPACE> -o yaml
kubectl auth can-i <VERB> <RESOURCE> -n <NAMESPACE>
```

13. Health check del cluster Kubernetes

Questa sequenza è pensata come controllo operativo non distruttivo. Non modifica risorse.

13.1 Nodi

```
kubectl get nodes -o wide
kubectl get nodes --no-headers | grep -v ' Ready '
```

Tutti i nodi attesi dovrebbero essere Ready. Approfondire NotReady, SchedulingDisabled non pianificato o condizioni anomale con kubectl describe node.

13.2 Pod non sani

```
kubectl get pods -A
kubectl get pods -A | grep -v Running | grep -v Completed
```

Il secondo comando è un filtro rapido: esaminare Pending, CrashLoopBackOff, ImagePullBackOff, Error, ContainerStatusUnknown e stati simili. Alcuni Job completati sono normali.

13.3 Componenti di sistema

```
kubectl get pods -n kube-system -o wide
kubectl get deployments,daemonsets -n kube-system
kubectl get pods -n kube-system | grep -E 'coredns|kube-proxy|flannel|calico|cilium'
```

Controllare in particolare DNS, CNI e kube-proxy (o equivalente), in base ai componenti effettivamente installati.

13.4 CoreDNS

```
kubectl get pods -n kube-system -l k8s-app=kube-dns -o wide
kubectl get svc -n kube-system kube-dns
kubectl logs -n kube-system -l k8s-app=kube-dns --tail=100
```

I pod CoreDNS devono essere Ready. Errori di pull, crash o mancata risoluzione DNS possono compromettere molte applicazioni.

13.5 Eventi recenti

```
kubectl get events -A --sort-by=.lastTimestamp | tail -50
```

Cercare Failed, BackOff, Unhealthy, FailedScheduling, FailedMount e problemi di rete/storage. Gli eventi vecchi possono riferirsi a incidenti già risolti: correlare sempre timestamp e stato attuale.

13.6 Deployment e DaemonSet

```
kubectl get deployments -A
kubectl get daemonsets -A
kubectl get statefulsets -A
```

READY/AVAILABLE dovrebbero corrispondere alle repliche desiderate, salvo rollout o scaling intenzionali.

13.7 API e risorse base

```
kubectl cluster-info
kubectl get --raw='/readyz?verbose'
kubectl get --raw='/livez?verbose'
```

readyz e livez interrogano gli endpoint di salute dell'API server. Richiedono permessi adeguati.

13.8 Storage

```
kubectl get pvc -A
kubectl get pv
```

PVC in Pending o PV in stato inatteso richiedono verifica di StorageClass, provisioner, mount e capacità.

13.9 Ingress / Gateway / LoadBalancer

```
kubectl get ingress -A
kubectl get svc -A | grep LoadBalancer
kubectl get gatewayclass
kubectl get gateway -A
kubectl get httproute -A
```

Verificare indirizzi assegnati, condizioni Accepted/Programmed per Gateway API e controller effettivamente in esecuzione.

13.10 Controllo risorse

```
kubectl top nodes
kubectl top pods -A --sort-by=cpu
kubectl top pods -A --sort-by=memory
```

Questi comandi richiedono Metrics Server o un provider equivalente. Cercare saturazione CPU/memoria e pod fuori scala rispetto al normale.

14. Checklist 'cluster sano'

- ☐ Tutti i nodi attesi sono Ready.
- ☐ Nessun pod inatteso è in CrashLoopBackOff, ImagePullBackOff, Error o Pending prolungato.
- ☐ CoreDNS è Ready e stabile.
- ☐ CNI/kube-proxy o equivalenti sono operativi sui nodi previsti.
- ☐ Deployment/DaemonSet/StatefulSet hanno repliche coerenti con il desiderato.
- ☐ Non ci sono eventi recenti Failed/Unhealthy non spiegati.
- ☐ API server readyz/livez risponde correttamente.
- ☐ PVC/PV necessari sono Bound e utilizzabili.
- ☐ Ingress/Gateway/LoadBalancer espongono gli indirizzi attesi.
- ☐ CPU e memoria non mostrano saturazioni anomale.

15. Regola operativa per troubleshooting

Quando una risorsa non funziona, partire dallo stato e scendere di livello: get → describe → events → logs → dipendenze (Service/Endpoint, DNS, storage, rete, image pull). Evitare cancellazioni o restart indiscriminati prima di aver raccolto gli elementi diagnostici.