100 REAL-WORLD KUBECTL OUTPUTS AND WHAT THEY REALLY MEAN

	Command	What It Tells You
01	kubectl get pods	→ See status, node, and age — is it CrashLoopBackOff? Pending? That's your first clue.
02	kubectl describe pod <pod>	→ Check Events at the bottom: probe failures, OOMKills, pull errors — it's all here.
03	kubectl logs <pod>	→ Your app's voice. Crashes, exceptions, stack traces — this is where they speak.
04	kubectl logs -p <pod>	→ Previous container logs. Great for finding out what caused the last crash.
05	kubectl exec -it <pod> -- sh	→ Go inside. Check file paths, DNS, environment — find config issues fast.
06	kubectl get events --sort-by=.metadata.creationTimestamp	→ Full timeline. What failed first? When? Tracks the chaos.
07	kubectl top pod	→ Shows real CPU & memory usage. Is something eating 2Gi when it only asked for 200Mi?
08	kubectl get endpoints <svc>	→ No endpoints = no pods = your service is blind. Fix that deployment.
09	kubectl get svc	→ Misconfigured ports? Wrong type? Exposed to internet when it shouldn't be?
10	kubectl get deploy	→ Desired vs Current. Is it scaled down to 0 or stuck updating?

	Command	What It Tells You
11	kubectl rollout status deploy <name>	→ Is it rolling out cleanly or stuck? This will tell you.
12	kubectl get rs	→ Old ReplicaSets hanging around? Might be hogging resources.
13	kubectl get nodes	→ Check status, roles, taints. Any nodes in NotReady? That's a red flag.
14	kubectl describe node <node>	→ Resource pressure, taints, conditions. Hidden node issues show up here.
15	kubectl get pvc	→ Are your Persistent Volumes bound? Pending PVCs = stuck pods.
16	kubectl get pv	→ Reclaim policies, storage class issues. Leaking storage? This shows it.
17	kubectl describe svc <svc>	→ IPs, selectors, target ports. Find mismatches that break networking.
18	kubectl get configmap	→ Your app depends on this. If it's missing or misconfigured, nothing works.
19	kubectl get secret	→ Secrets not mounted or misnamed? Leads to crash-on-start.
20	kubectl get hpa	→ Is HPA even scaling? Min/Max/Current replica clues live here.
21	kubectl get ingress	→ No rules or wrong host = HTTP 404. This is where you check.
22	kubectl describe ingress <name>	→ Shows backend mappings. Use this for debugging strange 502s.
23	kubectl port-forward	→ Temporarily access a service locally for quick testing.
24	kubectl cp <pod>:<path> ./	→ Copy logs, files or data out of a pod for offline analysis.
25	kubectl auth can-i create pods	→ Quick check for RBAC. Useful during permission errors.

	Command	What It Tells You
26	kubectl get cronjobs	→ Are jobs stuck? Misfiring? Crashing on every run?
27	kubectl get jobs	→ Succeeded, failed, active — helps debug batch workloads.
28	kubectl describe job <name>	→ Shows pod status, completions, failures.
29	kubectl get all	→ One-shot view of pods, services, deployments. Great for context.
30	kubectl label pod <pod> key=value	→ Add labels to group or filter pods in real time.
31	kubectl get pod -o yaml	→ Full YAML view — image, env, mounts. Great for deep debugging.
32	kubectl rollout undo deploy <name>	→ Roll back a failed release in seconds.
33	kubectl get daemonsets	→ Are agents (e.g., logs, metrics) running on every node?
34	kubectl get networkpolicies	→ Blocks in traffic? Policies could be rejecting your connections.
35	kubectl get poddisruptionbudgets	→ If this is misconfigured, your app won't scale down.
36	kubectl describe hpa <name>	→ See real metrics and scaling decisions behind the scenes.
37	kubectl get lease	→ Important for HA controllers like leader elections.
38	kubectl describe configmap <name>	→ Spot missing keys or bad values before they break your app.
39	kubectl api-resources	→ Lists all types available in the cluster. Often missed!
40	kubectl get po --field-selector=status.phase=Pending	→ Filters pods stuck in Pending — often a scheduling issue.

	Command	What It Tells You
41	kubectl get events -A	→ Cluster-wide view of everything going wrong.
42	kubectl taint nodes	→ Confirms why pods aren't scheduling. Taints can block deployments.
43	kubectl cordon <node>	→ Safely drain a node without new pods getting scheduled.
44	kubectl drain <node>	→ Evict pods from a node before maintenance. Important for HA.
45	kubectl edit deploy <name>	→ Live edit YAML — risky but fast fix.
46	kubectl patch	→ Targeted updates to resources without full redeploy.
47	kubectl version --short	→ Spot version mismatches between server and client.
48	kubectl get sc	→ Check StorageClasses — defaults, parameters, reclaim policies.
49	kubectl top node	→ Which node is overloaded? Fix noisy neighbors.
50	kubectl get csr	→ Shows pending cert signing requests — critical for TLS bootstraps.
51	kubectl delete pod <name>	→ Crash fix 101 — force a restart.
52	kubectl describe replicaset <name>	→ Useful for debugging rollouts and misaligned scaling.
53	kubectl explain pod.spec.containers	→ Know what each field does. This is built-in documentation.
54	kubectl api-versions	→ Deprecated APIs? Plan your upgrades right.
55	kubectl diff -f <yaml>	→ See what will change before applying a YAML.

	Command	What It Tells You
56	kubectl apply -f <yaml> --dry-run=client	→ Test YAML without impacting the cluster.
57	kubectl get po -l app=<name>	→ Filter pods by labels — goldmine for debugging multi-pod apps.
58	kubectl set image	→ Deploy new version of your app with 1 command.
59	kubectl scale deploy <name> --replicas=0	→ Instantly bring down all pods — for emergencies or restarts.
60	kubectl scale deploy <name> --replicas=5	→ Scale up/down manually — great for testing HPA fallbacks.
61	kubectl get namespaces	→ Check for unnecessary namespaces. Too many = chaos and sprawl.
62	kubectl get po -n <namespace>	→ You're likely debugging the wrong namespace. This shows the real pods.
63	kubectl describe po -n <ns>	→ See events and conditions specific to that namespace.
64	kubectl get events --field-selector=type=Warning	→ Filters out the noise. Focus on actual red flags.
65	kubectl get ingress -A	→ Cluster-wide overview. Are rules clashing? Any 404-prone gaps?
66	kubectl get svc -A	→ Unused services? Port conflicts? Easy to spot in this view.
67	kubectl get cm -A	→ Too many ConfigMaps across namespaces = bad hygiene.
68	kubectl get secret -A	→ Secrets in plaintext? Missing from expected namespaces?
69	kubectl get sa -A	→ ServiceAccounts tied to wrong permissions? This reveals misuses.
70	kubectl describe sa <name>	→ Which secrets/tokens is this ServiceAccount using?

	Command	What It Tells You
71	kubectl get role -A	→ Who's got what power in which namespace? Useful for RBAC audits.
72	kubectl get rolebinding -A	→ Check if dangerous roles are being assigned casually.
73	kubectl get clusterrole	→ Spot over-permissive roles (like cluster-admin clones).
74	kubectl get clusterrolebinding	→ Your RBAC attack surface. Check if default users are too privileged.
75	kubectl describe rolebinding <name>	→ Maps users/groups to roles. Audit this monthly.
76	kubectl get mutatingwebhookconfiguration	→ Admission controllers like Kyverno/Falco show up here.
77	kubectl get validatingwebhookconfiguration	→ These can block deployments. Broken ones = stuck workloads.
78	kubectl get lease -A	→ Identify controllers holding leader elections (HA apps).
79	kubectl get csr -A	→ Debug client certificate bootstraps (nodes, users, etc).
80	kubectl get po -w	→ Watch pods live. See restarts, terminations, transitions in real time.
81	kubectl logs -f <pod>	→ Follow logs live — debug crashes, probe fails, latency.
82	kubectl get deploy -w	→ Watch rollout live. Useful during set image or apply.
83	kubectl get hpa -w	→ Watch autoscaling in real time based on metrics.
84	kubectl run debug --image =busybox -- sleep 3600	→ Drop in a temporary debugging pod — no deployment needed.
85	kubectl get po -o jsonpath='{.items[*].status.containerStatuses[*].restartCount}'	→ Shows restart counts across all pods. Spot instability fast.

	Command	What It Tells You
86	kubectl get po -o json | jq '.items[].status.containerStatuses[].lastState'	→ Previous states: why your containers restarted.
87	kubectl get po -o wide	→ Shows node, IP, and restarts. Helps match symptoms to infra.
88	kubectl get po --sort-by=.status.startTime	→ Know which pod is oldest — useful in rollouts and HPA.
89	kubectl get deploy -o=jsonpath='{.items[*].spec.template.spec.containers[*].image}'	→ What images are you running? Avoids zombie versions.
90	kubectl config view	→ Validate current context, user, clusters — especially on shared machines.
91	kubectl config current-context	→ Stop debugging the wrong cluster.
92	kubectl config use-context <name>	→ Switch between dev/stage/prod clusters safely.
93	kubectl delete -f <file.yaml>	→ Undo an apply by deleting all defined resources.
94	kubectl logs <pod> -c <container>	→ In multi-container pods, this is how you target the right one.
95	kubectl get po -l app!=<name>	→ Exclude certain apps. Focus on noisy neighbors or infra-only pods.
96	kubectl get rs -o=jsonpath='{.items[*].spec.replicas}'	→ Replica mismatches? Spot autoscaling leftovers.
97	kubectl describe cronjob <name>	→ Check schedule, concurrency policy, and last/next run.
98	kubectl get serviceaccounts -o=jsonpath='{.items[*].spec.template.spec.containers[*].image}'	→ Inspect token exposure through SA secrets.
99	kubectl get nodes -o jsonpath='{.items[*].status.conditions}'	→ Node health at a glance. Spot DiskPressure, NetworkUnavailable, etc.
100	kubectl get po -l tier!=prod	→ Target non-prod pods only. Great for cleanup, testing, validation.