Kubernetes Troubleshooting

Pods • Nodes • Common Failures • Events/Quotas • Ephemeral Debug • Networking • Storage

Audience: Advanced / Intermediate (beginner-accessible)

Focus: Systematic diagnosis, mental models, and repeatable playbooks

Agenda

- Debugging Pods (describe/logs/exec)
- Debugging Nodes (health, taints & conditions)
- Common Pod Failures (CrashLoopBackOff, ImagePullBackOff)
- Events & Resource Quotas
- Recreating Failures (ephemeral containers)
- Troubleshooting Networking Issues
- Troubleshooting Storage Issues

1) Debugging Pods

First 5 Minutes: Describe → Logs → Exec

> Triage checklist (namespace-aware: add `-n <ns>`)

```
kubectl get pods -o wide
kubectl describe pod <pod>
kubectl logs <pod> -c <container>
kubectl logs <pod> -c <container> --previous
kubectl exec -it <pod> -c <container> -- sh
```

- **Look for (describe):** Scheduled, Pulled, Created, Started, probe failures, OOMKilled, permissions, mount errors.
- **Log focus:** Last 200–1000 lines, error patterns, startup sequences, dependency timeouts.
- **If no shell:** Use ephemeral containers (see section 5).

2) Debugging Nodes

Node Health, Taints, Conditions

```
kubectl get nodes -o wide
kubectl describe node <node>
kubectl get nodes --show-labels
kubectl get node <node> -o jsonpath='{.spec.taints}'
```

- **Key conditions:** Ready, DiskPressure, MemoryPressure, PIDPressure, NetworkUnavailable.
- **Kubelet status:** Check logs on node (journal) for CRI/CNI/CSI errors.
- **Taints:** NoSchedule/NoExecute may block pods—verify tolerations on workloads.

```
kubectl get pod -A -o wide --field-selector spec.nodeName=<node>
```

Scheduling & Capacity Clues

3) Common Pod Failures

CrashLoopBackOff

- **Meaning:** Container exits repeatedly (non-zero or immediate exit).
- **Causes:** Bad config/env, missing dependency, port conflicts, liveness killing it, insufficient resources.

Playbook:

```
kubectl logs --previous
# Confirm probes/args/env, ConfigMap/Secret mounted?
# Check OOM (reason: OOMKilled)
# Start without liveness probe or with sleep entrypoint to debug
kubectl logs <pod> -c <ctr> --previous
kubectl describe pod <pod> | awk '/State|Last State|Reason|Exit Code|Restart Count/'
```

ImagePullBackOff / ErrImagePull

- **Meaning:** Image cannot be pulled

4) Events & Resource Quotas

Events as a Timeline

```
kubectl get events --sort-by=.lastTimestamp -A
kubectl describe pod <pod> | sed -n '/Events/,$p'
```

- **Interpretation:** Look for throttled pulls, failed mounts, admission denials, policy violations, requeues.

Quotas & Limits

```
kubectl get resourcequota -n <ns>
kubectl describe resourcequota -n <ns> <rq>
kubectl get limitrange -n <ns>
```

- **Symptoms:** Creates denied (exceeded quota), pods pending due to missing requests/limits.

5) Recreating Failures

Ephemeral Containers for Debug

- **Use when:** Container lacks shell/tools, or crashes immediately.

Add ephemeral container:

```
kubectl debug -n <ns> -it <pod> --image=busybox:stable --target=<container> --share-processes
kubectl -n <ns> debug pod/<pod> -c debugger --image=ghcr.io/distroless.dev/bash
```

- **Inspect from inside pod netns:** Verify files, env, DNS, ports, permissions.
- **Alternative:** `kubectl debug node/<node> --image=...` to chroot into node for daemon issues.

Controlled Reproduction

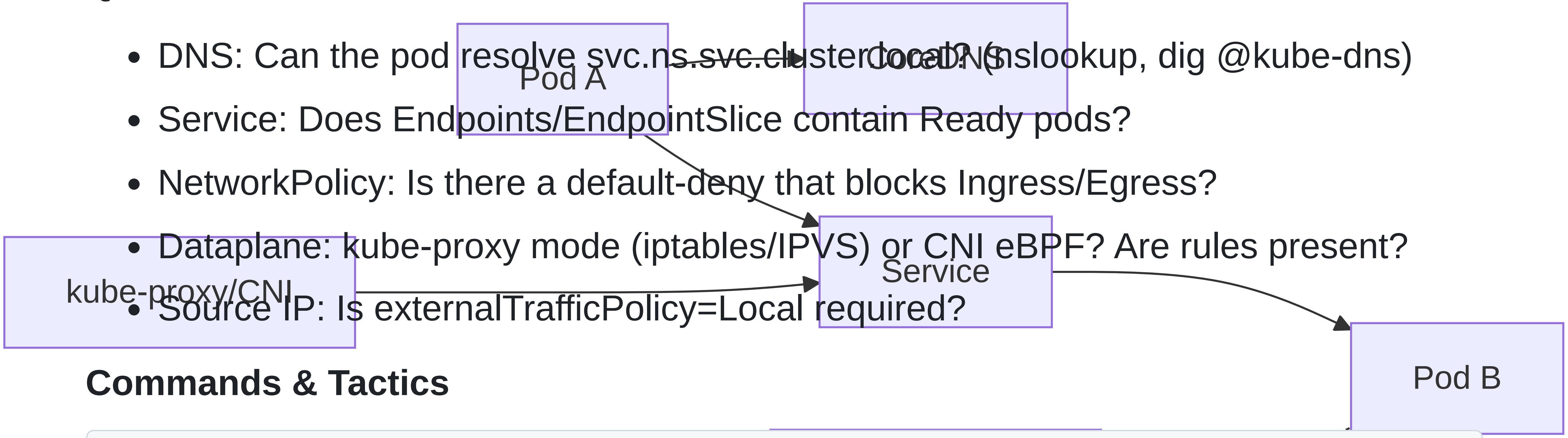
- Adjust command/args to add sleep before real entrypoint.
- Scale replicas to 1; disable HPA; pin to a node (nodeSelector) that shows issue

6) Troubleshooting Networking Issues

Model & Checklist

Questions to answer:

- DNS: Can the pod resolve svc.ns.svc.cluster.local? (nslookup, dig @kube-dns)
- Service: Does Endpoints/EndpointSlice contain Ready pods?
- NetworkPolicy: Is there a default-deny that blocks Ingress/Egress?
- Dataplane: kube-proxy mode (iptables/IPVS) or CNI eBPF? Are rules present?
- Source IP: Is externalTrafficPolicy=Local required?



Commands & Tactics

```
# DNS from within PodA
kubectl exec -it <podA> -- sh -c 'getent hosts svc.ns.svc.cluster.local || nslookup svc.ns'

# Service/Endpoints
kubectl get svc <svc> -n <ns> -o wide
kubectl get endpointslices -n <ns> | grep <svc>
kubectl describe endpointslice <name> -n <ns>

# NetworkPolicy effective?
kubectl get netpol -n <ns>
```

7) Troubleshooting Storage Issues

PV/PVC Binding & Mounts

- **Symptoms & Causes:**
    - PVC Pending: No matching StorageClass, quota exceeded, topology constraints (zone).
    - Mount failures: Permission/SELinux denials, fs type mismatch, busy mount points.
    - ReadOnly filesystem: Volume marked RO, or underlying claim policies.

```
kubectl get pvc -n <ns>
kubectl describe pvc/<pvc> -n <ns>
kubectl describe pod/<pod> -n <ns> | sed -n '/Volumes:/,/Events/p'
kubectl get pv | grep <pvc-name>
```

- **Dynamic provisioning:** Check StorageClass params and WaitForFirstConsumer

Playbook: End-to-End Triage (Summary)

- Identify scope: Single pod, deployment, node, or namespace-wide?
- Describe & Events: `kubectl describe` → events timeline.
- Logs: Current and --previous; check crash signatures.
- Resources: Requests/limits, `kubectl top`, OOM/throttle clues.
- Network: DNS → Service → Endpoints → NetPol → proxy rules.
- Storage: PVC/PV/SC status → mounts → permissions/SELinux.
- Node: Conditions/taints, kubelet logs, CNI/CSI health.
- Reproduce: Ephemeral container, delay entrypoint, pin placement.
- Document fix: Root cause, remediation, guardrail (policy/alert/test).

Reference Commands (Cheat Sheet)

Pods

```
kubectl get pod -A -o wide
kubectl describe pod <pod> -n <ns>
kubectl logs <pod> -n <ns> --all-containers --tail=500
kubectl exec -it <pod> -n <ns> -- sh
```

Nodes

```
kubectl get nodes -o wide
kubectl describe node <node>
kubectl top node && kubectl top pod -A
```

Services & DNS

```
kubectl get svc,endpointslices -n <ns>
kubectl -n <ns> exec -it <pod> -- sh -c 'nslookup <svc>; getent hosts <svc>'
```

Best Practices & Guardrails

- Design for debuggability: Structured logs, /metrics, lightweight /healthz.
- Shift-left policies: Block known-bad patterns (e.g., missing requests, :latest).
- Observability: Correlate logs/metrics/traces; add deployment annotations to dashboards.
- Runbooks: Keep issue-specific SOPs in repo; practice incident drills.