Kubernetes Incident Management Runbooks

Overview

This document contains detailed incident management runbooks for common Kubernetes failure modes in both development and production environments. These were selected based on frequency, severity, and their relevance to different layers of the Kubernetes stack (application, infrastructure, and network). Each runbook is designed with DevOps, Site Reliability Engineering (SRE), and Production Support best practices in mind, focusing on early detection, rapid mitigation, root cause identification, and prevention.

Why These Failure Modes Were Chosen

The following failure modes were selected to represent different critical layers of the Kubernetes ecosystem:

1. CrashLoopBackOff – Application layer (development focus): Common during iterative coding and deployment.
2. Node Failure – Infrastructure layer (production focus): Direct impact on service availability and SLA.
3. DNS Resolution Failure – Core networking service (affects both dev and prod): Affects service discovery and external API calls.
4. NetworkPolicy Misconfiguration – Network policy layer (production focus): Causes hard-to-diagnose inter-service failures.

These failures illustrate how incidents span various responsibilities from developer-level fixes to deep infrastructure debugging. Together, they help build a comprehensive incident response capability.

DevOps and SRE Best Practices

- Use automated monitoring with Prometheus, Grafana, and alerting tools (e.g., Alertmanager, PagerDuty).
- Implement readiness and liveness probes to detect failures early.
- Maintain runbooks and incident retrospectives in version-controlled documentation.
- Use Infrastructure as Code (e.g., Helm, Terraform) for reproducibility.
- Apply chaos engineering principles to simulate and prepare for failures.
- Establish clear SLOs, SLIs, and error budgets to guide reliability goals.
- Test and validate NetworkPolicy changes in staging environments before production rollout.

- Ensure redundancy at the node and service level (HA configurations).
- Enforce CI/CD pipeline policies to catch issues (e.g., linting, policy scanning).
- Use central logging (ELK/EFK stacks) and distributed tracing for end-to-end visibility.

CrashLoopBackOff – Development

Failure Mode

Pod is stuck in CrashLoopBackOff state.

Symptoms

- Application is unreachable.
- `kubectl get pods` shows CrashLoopBackOff status.
- High restart count for the pod.

Diagnostics

1. `kubectl describe pod <pod-name>`
2. `kubectl logs <pod-name>`
3. Check deployment configuration and environment variables.

Causes

- Code bug
- Misconfigured environment variable
- Wrong or inaccessible image tag
- App port not exposed properly

Actions

1. Review logs and fix the application crash.
2. Update environment variables or secrets.
3. Validate and push correct image.
4. Rebuild and redeploy.

Remediation

Apply configuration and monitor pod with `kubectl rollout status deployment/<deployment-name>`.

Node Failure – Production

Failure Mode

Kubernetes node becomes unresponsive or fails.

Symptoms

- Pods enter Unknown/NotReady state.
- `kubectl get nodes` shows node as NotReady.
- High latency or outage.

Diagnostics

1. `kubectl describe node <node-name>`
2. Check cloud console for issues.
3. Use SSH, dmesg, journalctl, and top.

Causes

- Kubelet crash
- Resource exhaustion
- Network partition
- Infrastructure failure

Actions

1. Drain or cordon the node.
2. Reschedule pods.
3. Trigger autoscaler or replace node.

Remediation

Uncordon or remove node. Validate pod rescheduling and restore services.

DNS Resolution Failure – Dev & Prod

Failure Mode

Pods cannot resolve DNS names.

Symptoms

- DNS resolution errors in logs.
- `nslookup` or `dig` fails inside pods.

Diagnostics

1. Test DNS with nslookup
2. Check CoreDNS status and logs
3. Review ConfigMap and IPTables

Causes

- CoreDNS crash or overload
- ConfigMap misconfiguration
- NetworkPolicy blocking UDP 53

Actions

1. Restart CoreDNS
2. Fix DNS config
3. Scale CoreDNS or fix policies

Remediation

Verify DNS resolution, monitor with Prometheus, and set up alerts.

NetworkPolicy Misconfiguration – Production

Failure Mode

Services cannot communicate due to NetworkPolicy.

Symptoms

- Connection timeouts or refused errors.
- Breaks in inter-service traffic.

Diagnostics

1. Describe NetworkPolicies
2. Test inter-pod communication
3. Use Netshoot or CNI tooling

Causes

- Missing ingress/egress rules
- Wrong selectors or deny-all by default

Actions

1. Temporarily disable policy
2. Apply allow-all policy
3. Update rules and reapply

Remediation

Test communication, document patterns, simulate/test policy in CI/CD.