Bavithran
@bavicnative

Kubernetes

A DevSecOps Playbook

Kubernetes in Production – The 10 Non-Negotiables

Before You Go Live...

"We shipped to production and... the pods were running, but nothing worked."

Too many teams treat Kubernetes like a dev toy, until their first 3 AM pager.

Before you press deploy in prod, here are 10 rules you never break if you care about uptime, sanity, and security.

This isn't a checklist for the faint-hearted. It's built from scars, outages, and real-world chaos.

Bavithran
@bavicnative

1. Liveness & Readiness Probes

My pod is alive. But is it ready?

Imagine this: You deploy. The load balancer starts sending traffic. Your app is booting. Users see errors. Management panics.

The fix? Kubernetes gives you **livenessProbe** and **readinessProbe** for a reason. Don’t skip them.

Real Implementation:

```
livenessProbe:
 httpGet:
  path: /healthz
  port: 8080
readinessProbe:
 httpGet:
  path: /ready
  port: 8080
```

Golden Rules:

- Your app might run but still not be ready. That’s the point.
- Without readiness probes, Kubernetes thinks your pod is fine — even when it's not.

Without liveness probes, zombie pods stay alive, silently failing.

Production Rule: Never ship a deployment without both probes. Ever.

Bavithran
@bavicnative

2. Resource Requests & Limits

"It worked on my cluster - until it ate all the memory."
Kubernetes nodes are shared playgrounds. Without boundaries, a single container can choke the system.

Real Implementation:

```
resources:
  requests:
    cpu: "250m"
    memory: "512Mi"
  limits:
    cpu: "500m"
    memory: "1Gi"
```

Golden Rules:

- Requests are promises. Limits are hard stops.
- Always set both. No exceptions.
- Use performance profiling to tune them — not guesswork.

Production Rule: No container goes to prod without defined resource limits.

Bavithran
@bavicnative

3. Horizontal Pod Autoscaler (HPA)

"Traffic spiked. The app crashed. Monitoring said it was fine."
Manual scaling is a myth. You won’t be online every second.

Real Implementation:

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
spec:
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 60
```

Golden Rules:

- Enable metrics-server
- Use CPU for simple apps, custom metrics for precision

Production Rule: Autoscale what matters. Always.

4. Network Policies

"A frontend in one namespace talked to a database in another — unencrypted."
By default, Kubernetes allows all traffic. You must explicitly lock it down.

Real Implementation:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
spec:
 podSelector:
  matchLabels:
   role: backend
 ingress:
 - from:
  - podSelector:
    matchLabels:
     role: frontend
```

Golden Rules:

- Default deny, then explicitly allow
- Define policies per namespace or microservice

Production Rule: No public east-west traffic. Ever.

5. Pod Disruption Budgets (PDB)

"Our upgrade caused a complete service outage — even with three replicas."
PDBs prevent voluntary disruptions from killing all your pods.

Real Implementation:

```
apiVersion: policy/v1
kind: PodDisruptionBudget
spec:
  minAvailable: 2
  selector:
    matchLabels:
      app: myapp
```

Golden Rules:

- Use with StatefulSets and Deployments
- Ensure PDB aligns with HPA and replica count

Production Rule: If uptime matters, use PDBs.

Bavithran
@bavicnative

6. Security Contexts & Pod Policies

"The container was running as root. A CVE exploited it."

Least privilege is not optional — it’s required.

Real Implementation:

```
securityContext:
  runAsNonRoot: true
  allowPrivilegeEscalation: false
```

Golden Rules:

- Drop root wherever possible
- Apply PodSecurityAdmission or OPA Gatekeeper
- Restrict capabilities

Production Rule: Treat every pod like it’s exposed to the internet.

Bavithran
@bavicnative

7. Logging & Monitoring

"The app failed silently. We had no logs."

If you can't observe it, you can't fix it.

Golden Rules:

- Centralize logs using Fluentd, Loki, or ELK
- Use Prometheus + Grafana for metrics
- Set actionable alerts with Alertmanager
- Trace using OpenTelemetry or Jaeger

Production Rule: Observability isn't optional. It's your lifeline.

8. Image Provenance & Vulnerability Scanning

"We pulled an image from a public registry. It had a crypto miner."
Images must be trusted, verified, and scanned continuously.

Golden Rules:

- Use signed images (Cosign, Notary)
- Scan every build (Trivy, Clair)
- Host images in private registries (Harbor, ACR, ECR)

Production Rule: Trust no image you didn't build or scan yourself.

9. Secrets Management

"Our S3 credentials were in plaintext inside a ConfigMap."
Kubernetes Secrets are base64-encoded — not encrypted by default.

Golden Rules:

- Use external secrets (Vault, SOPS, Azure Key Vault)
- Enable envelope encryption on etcd
- Rotate secrets regularly

Production Rule: Secrets belong in secret stores. Not in YAML files.

10. Rolling Updates & Deployment Strategies

"We deployed a new version. It broke everything. No rollback was in place."

Modern deployments should be gradual, observable, and reversible.

Golden Rules:

- Use RollingUpdate for default strategy
- For critical changes, use Blue-Green or Canary (Argo Rollouts, Flagger)
- Monitor metrics during rollout
- Automate rollback if errors spike

Production Rule: Every rollout must have a rollback plan.

Found this useful?

Follow

Let's connect

💬 Found it useful? Drop your thoughts below and share it with your fellow DevOps engineers!