
Bavithran
@bavicnative

🔐 Kubernetes Security Checklist

10 Common Vulnerabilities & How to Prevent Them – A DevSecOps Playbook

Bavithran
@bavicnative

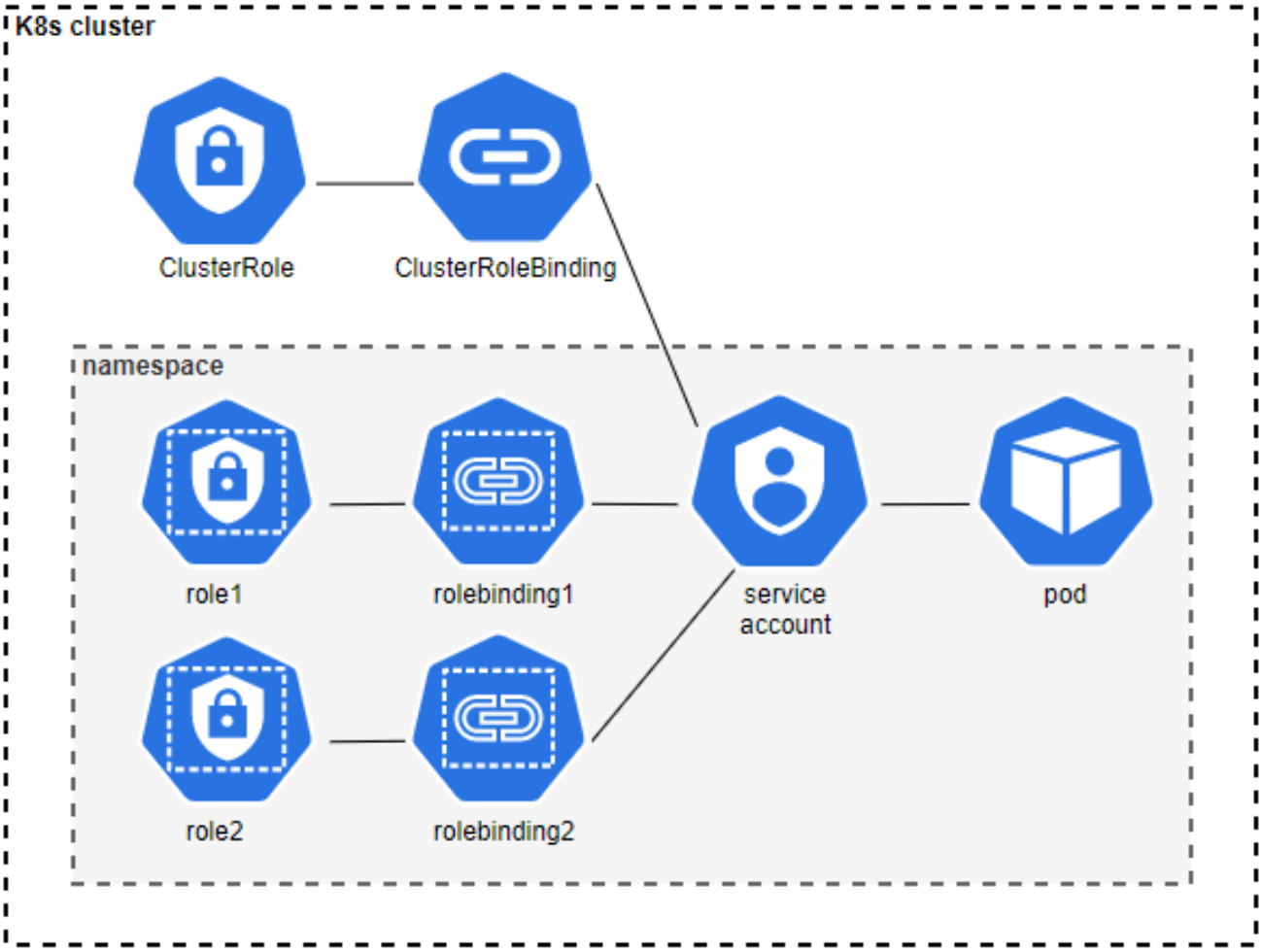
1.Overly Permissive RBAC Roles

Issue: Users or service accounts have cluster-wide access they don't need.

Risk: Accidental or malicious changes across the cluster.

Fix Checklist:

1. Use **Role** instead of **ClusterRole** wherever possible.
2. Apply least privilege principle.
3. Regularly audit RBAC policies:
4. **kubectl get clusterrolebindings --all-namespaces**




Bavithran
@bavicnative

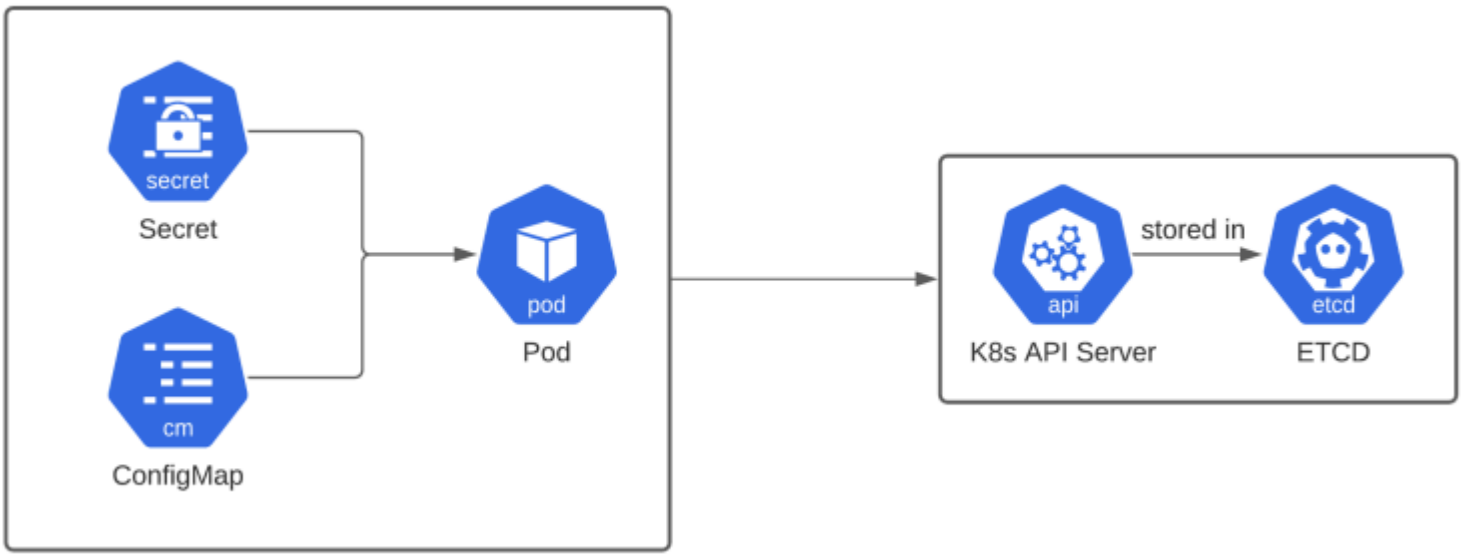
2. Secrets Stored as Plain Text

Issue: Kubernetes Secrets are base64-encoded, not encrypted.

Risk: Anyone with access can decode them.

Fix Checklist:

1. Enable encryption at rest using KMS (Key Management Service).
2. Use external secret managers like HashiCorp Vault, AWS Secrets Manager, or Azure Key Vault.
3. Avoid committing secrets to Git repos.




Bavithran
@bavicnative

3. Containers Running as Root

Issue: Pods run with root privileges.

Risk: Exploited containers can access host system.

Fix Checklist:

1. Set **securityContext.runAsNonRoot: true** in pod specs.
2. Use non-root base images (e.g., **distroless, alpine**).
3. Enforce with PodSecurityPolicy or OPA/Gatekeeper.


Bavithran
@bavicnative

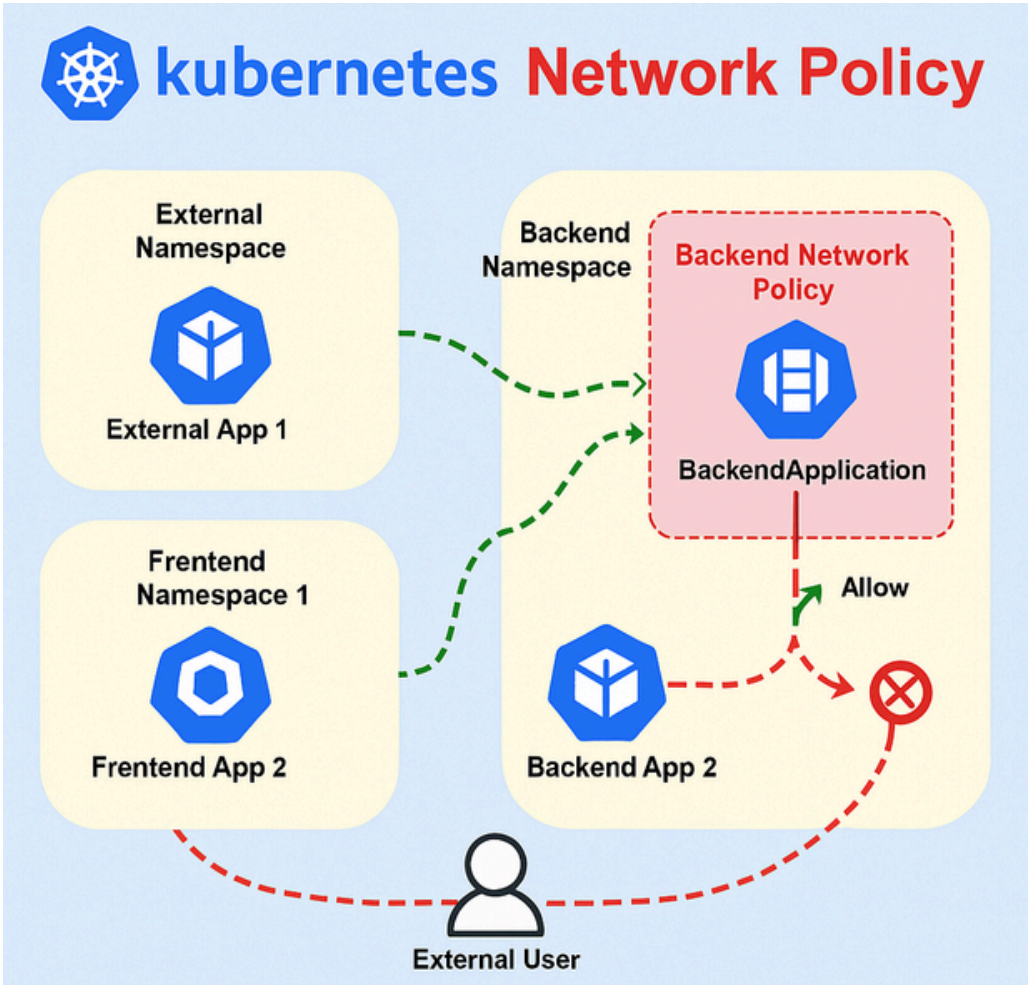
4. Missing Network Policies

Issue: All pods can talk to each other by default.

Risk: No traffic isolation = lateral movement in case of compromise.

Fix Checklist:

1. Define NetworkPolicy to restrict pod-to-pod communication.
2. Deny all by default, allow selectively.
3. Use labels to control traffic.



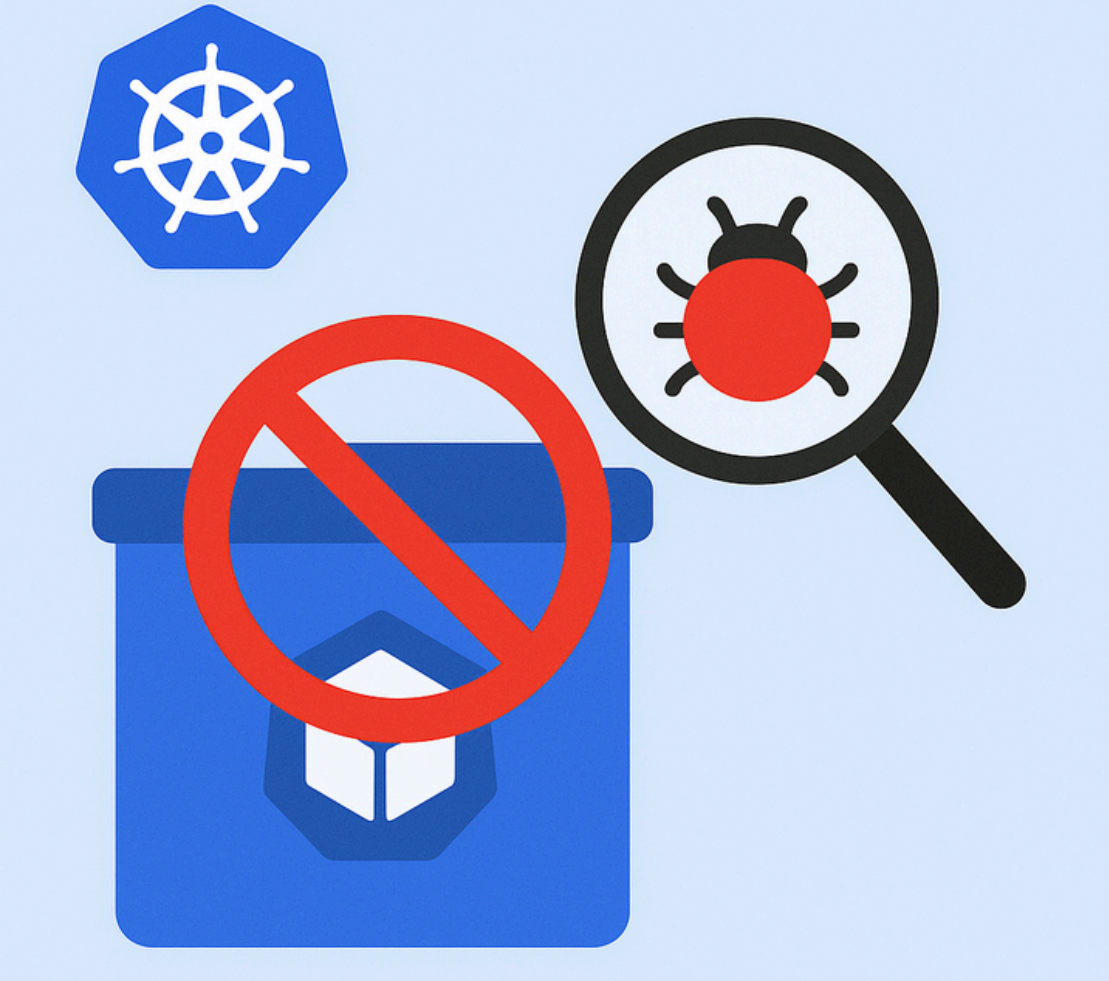

Bavithran
@bavicnative

5. Lack of Image Scanning

Issue: Vulnerable container images get deployed.
Risk: Known CVEs (Common Vulnerabilities and Exposures) exploited.

Fix Checklist:

1. Integrate image scanners like **Trivy, Aqua, Anchore, or Clair**.
2. Scan in CI/CD before pushing to registry.
3. Use signed images and trusted registries.




Bavithran
@bavicnative

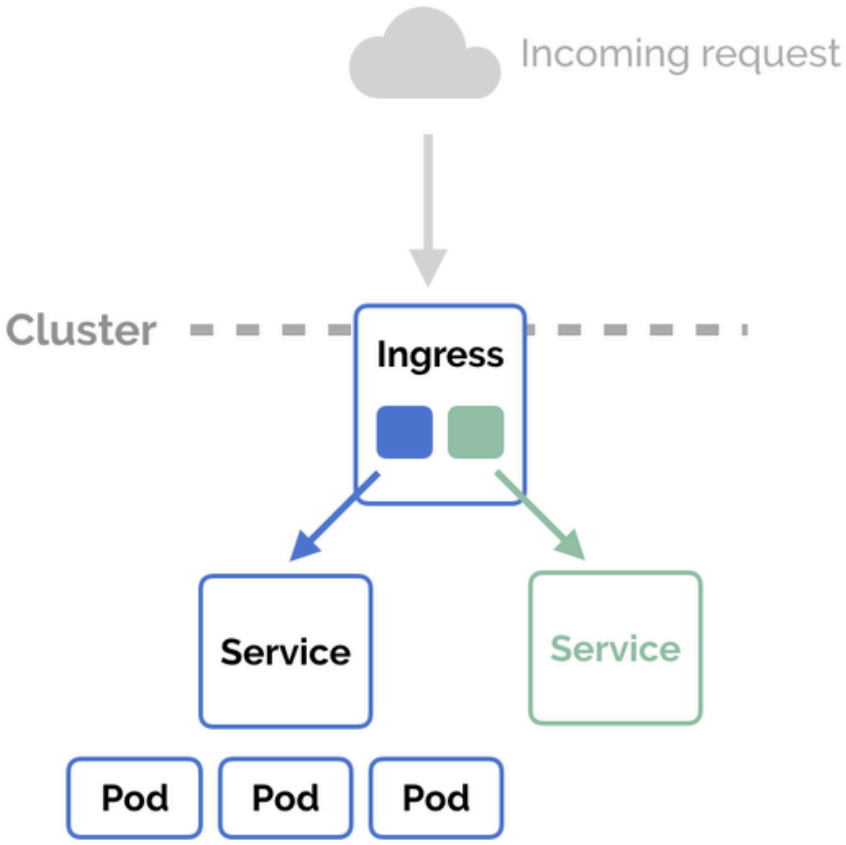
6. Uncontrolled Ingress Access

Issue: Public access to services that should be internal.

Risk: Exposes sensitive services to the internet.

Fix Checklist:

1. Use Ingress annotations for access control.
2. Apply WAF (Web Application Firewall) or API gateway with rate limiting.
3. Restrict paths and hosts via **Ingress** rules.




Bavithran
@bavicnative

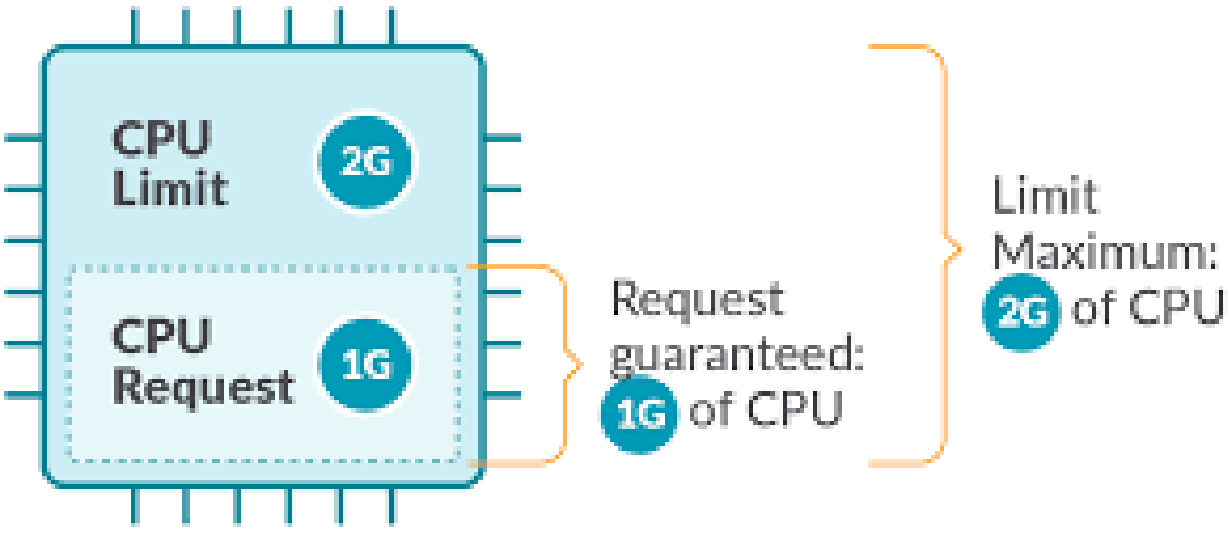
7. No Resource Limits Defined

Issue: Pods can consume unlimited CPU/memory.

Risk: Resource starvation, DoS inside cluster.

Fix Checklist:

1. Define **resources.requests** and **resources.limits** for every pod/container.
2. Set LimitRange and ResourceQuota at namespace level.




Bavithran
@bavicnative

8. Insecure API Server Access

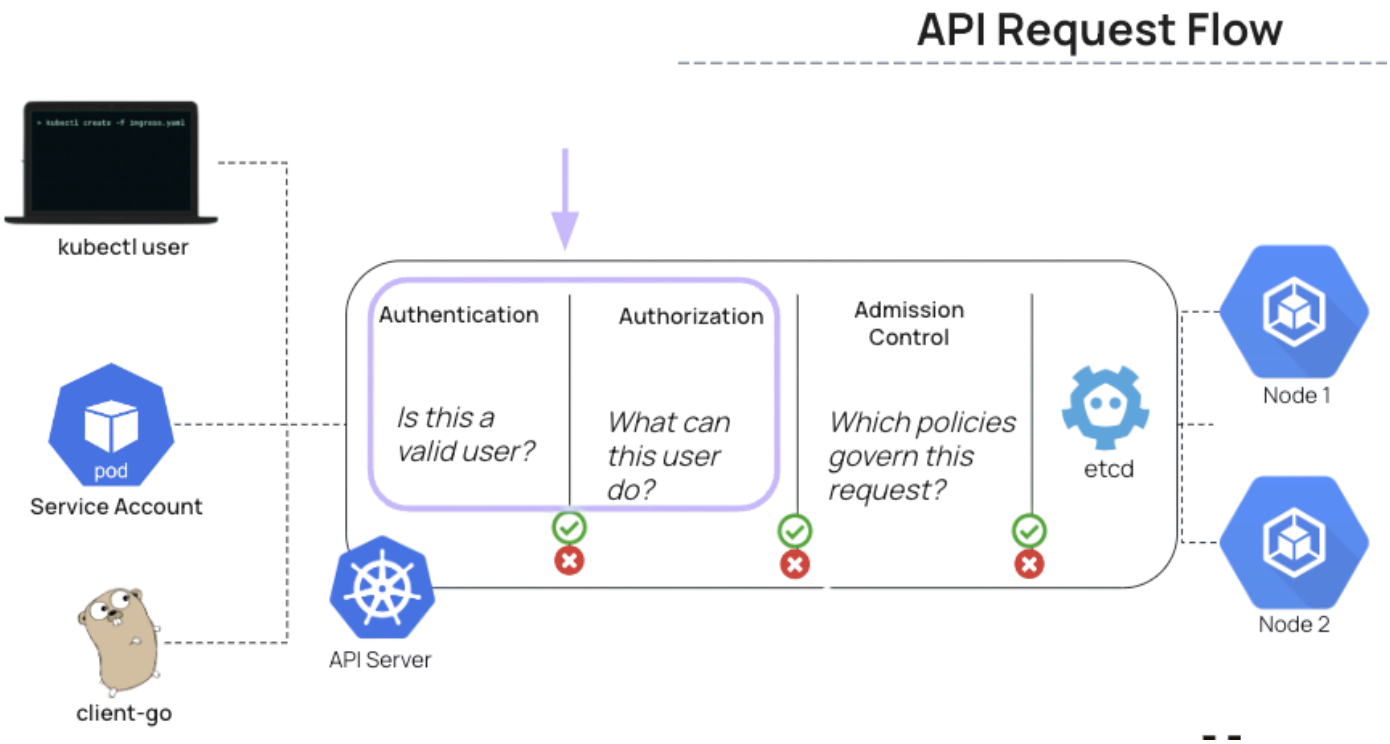
Issue: API server open to the internet or lacks strong authentication.

Risk: Unauthorized access, data exfiltration.

Fix Checklist:

1. Restrict access via firewall/security groups.
2. Enable RBAC + OIDC or client certs for authentication.
3. Disable anonymous access:

--anonymous-auth=false




Bavithran
@bavicnative

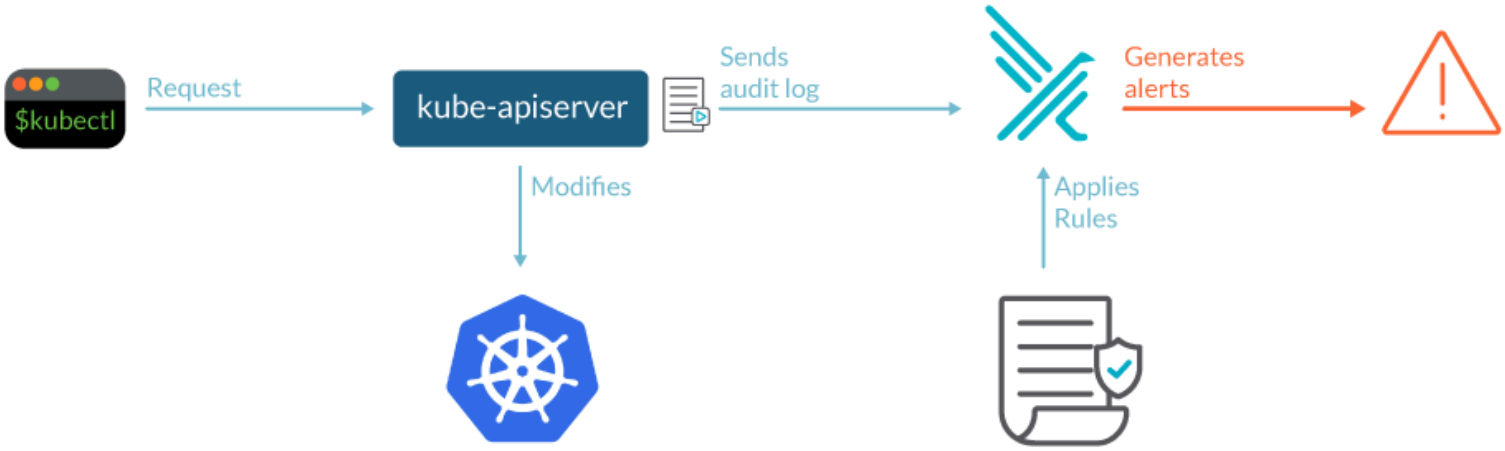
9. Unmonitored Audit Logs

Issue: No visibility into who did what.

Risk: Difficult to trace incidents or respond.

Fix Checklist:

1. Enable Kubernetes audit logging **(--audit-log-path)**.
2. Use tools like Falco, ELK, or Loki to monitor and alert.
3. Set up retention and backup of logs.




Bavithran
@bavicnative

10. Lack of Regular Security Updates

Issue: Outdated K8s versions, CVEs unpatched.

Risk: Attackers exploit known vulnerabilities.

Fix Checklist:

1. Follow Kubernetes release cycles and changelogs.
2. Automate patching via cluster management tools (e.g., EKS, AKS, GKE).
3. Regularly update node OS and containers.

Found this useful?

Follow

Let's connect

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