Static Analysis of Kubernetes Object Definitions Using Kube-Score: Enhancing Security and Resilience

Amar Gurajapu
AT&T
https://orcid.org/0009-0002-9038-2200

[1]*Date of Receiving: 22 December 2025; Date of Acceptance: 27 January 2026; Resubmitted after correction: 29th January 2026; Date of Publication: 02 February 2026*

Abstract

Deploying applications using Kubernetes has become a standard practice in modern cloud-native environments. However, these deployments are frequently hindered by misconfigurations, security vulnerabilities, and operational instability, all of which can significantly affect overall system reliability and security. This paper investigates the application of kube-score, an open-source static code analysis tool specifically designed for evaluating the quality and robustness of Kubernetes object definitions. We systematically examine the key features of kube-score, highlighting its ability to detect configuration errors and provide actionable, context-aware recommendations that enhance both the security posture and operational resilience of Kubernetes workloads. In addition, we review alternative approaches to Kubernetes configuration analysis, positioning kube-score in terms of usability, integration flexibility, and effectiveness. The paper further discusses deployment models for kube-score, including its use as a containerized service within CI/CD pipelines and as a standalone plugin for local development. Through a detailed case study, we demonstrate how integrating kube-score into existing DevOps workflows enables teams to identify and remediate potential issues early in the development lifecycle, thereby reducing risk and promoting best practices. Our findings underscore the practical benefits of automated static analysis in supporting secure, stable, and efficient Kubernetes operations.

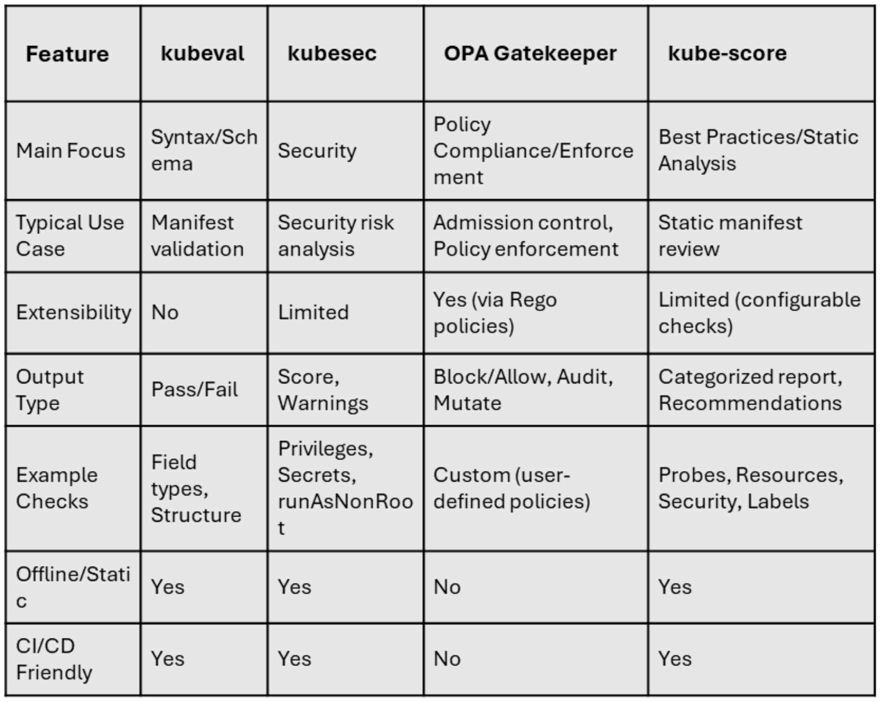
Index Terms – *Kubernetes; kube-score; Static Analysis; Security; Resilience; DevOps; Cloud-Native*

1. Introduction

With the evolution of modern architecture, Kubernetes has become the de facto standard for container orchestration in cloud-native environments. However, the complexity of configuring Kubernetes manifests can introduce subtle errors and security vulnerabilities. Ensuring best practices are followed is critical for maintaining reliable and secure workloads. This paper explores the role of kube-score, a static analysis tool that reviews Kubernetes object definitions and provides targeted recommendations for improvement.

Previous research and tooling in the Kubernetes ecosystem include tools like kubeval for manifest validation, kubesec for security scoring, and OPA Gatekeeper for policy enforcement. While these tools offer valuable checks, kube-score distinguishes itself by focusing on human-readable recommendations and best practices, targeting both security and operational robustness

[1] *How to cite the article:* Gurajapu A.; (February 2026); Static Analysis of Kubernetes Object Definitions Using Kube-Score: Enhancing Security and Resilience; *International Journal of Universal Science and Engineering;* Vol 13, 1-6



Feature	kubeval	kubesec	OPA Gatekeeper	kube-score
Main Focus	Syntax/Schema	Security	Policy Compliance/Enforcement	Best Practices/Static Analysis
Typical Use Case	Manifest validation	Security risk analysis	Admission control, Policy enforcement	Static manifest review
Extensibility	No	Limited	Yes (via Rego policies)	Limited (configurable checks)
Output Type	Pass/Fail	Score, Warnings	Block/Allow, Audit, Mutate	Categorized report, Recommendations
Example Checks	Field types, Structure	Privileges, Secrets, runAsNonRoot	Custom (user-defined policies)	Probes, Resources, Security, Labels
Offline/Static	Yes	Yes	No	Yes
CI/CD Friendly	Yes	Yes	No	Yes

(kubeval, 2025), (kubesec, 2019), (Gatekeeper, 2024), (kube-score, 2025)

TABLE 1. KUBE-SCORE AND OTHER TOOLS

kube-score is an open-source utility that statically analyzes Kubernetes manifests (YAML files) without requiring access to a live cluster. It inspects resources like Deployments, Services, and Ingresses, providing a comprehensive report on potential issues and improvement suggestions.

How kube-score Works

- Parses Kubernetes object definitions.
- Runs a series of checks, such as:
    - Are resource requests and limits set?
    - Are containers running as non-root?
    - Are probes (liveness/readiness) defined?
    - Are labels and annotations used appropriately?
- Outputs a scored report categorizing findings by severity.

Example Output

```
Deployment my-app
  [WARNING] Container should not run as root
  [INFO] Missing readiness probe
  [CRITICAL] No resource limits defined
```

2. Methodology

The research applied a qualitative descriptive approach, structured around the development and validation of the kubernetes manifest tailored for modern cloud architecture.

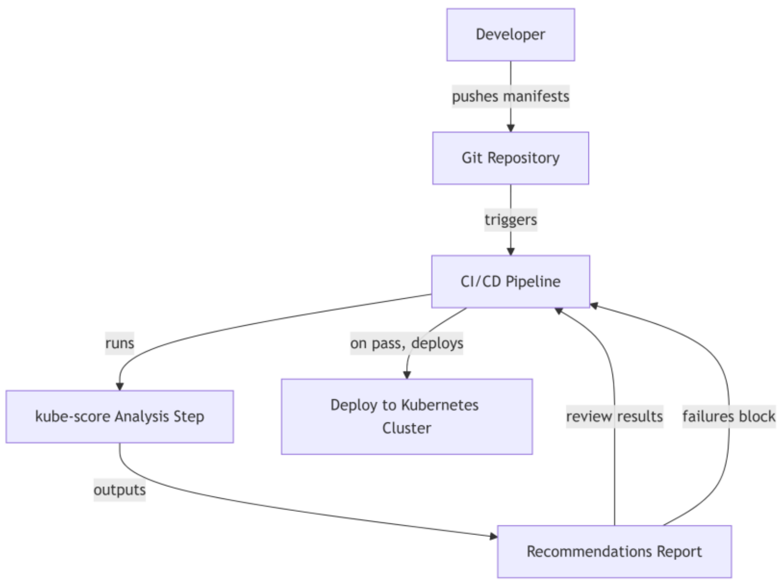


FIGURE 1. INTEGRATION INTO CI/CD WORKFLOW

The methodology comprised of:

Preparation
Gathered YAML files from the application repository.

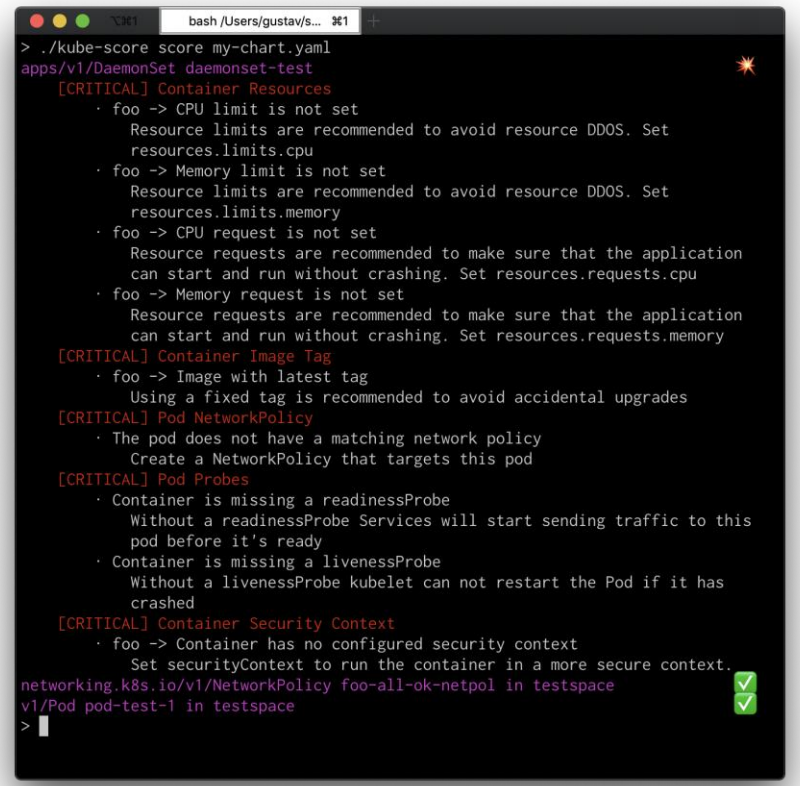
Analysis
Ran kube-score webservice hosted as a containerized platform with each our manifest.

Interpretation
Reviewed the recommendations and categorized them
by type (security, reliability, etc.).

Remediation
Applied fixes based on kube-score output and re-ran the analysis to validate improvements.

3. Result and Discussion

These instructions Kubernetes have become the de facto standard for container orchestration in cloud-native environments. However, the complexity of configuring Kubernetes manifests can introduce subtle errors and security vulnerabilities. Ensuring best practices are followed is critical for maintaining reliable and secure system.



```
> ./kube-score score my-chart.yaml
apps/v1/DaemonSet daemonset-test                                            💥
    [CRITICAL] Container Resources
        · foo -> CPU limit is not set
          Resource limits are recommended to avoid resource DDOS. Set
          resources.limits.cpu
        · foo -> Memory limit is not set
          Resource limits are recommended to avoid resource DDOS. Set
          resources.limits.memory
        · foo -> CPU request is not set
          Resource requests are recommended to make sure that the application
          can start and run without crashing. Set resources.requests.cpu
        · foo -> Memory request is not set
          Resource requests are recommended to make sure that the application
          can start and run without crashing. Set resources.requests.memory
    [CRITICAL] Container Image Tag
        · foo -> Image with latest tag
          Using a fixed tag is recommended to avoid accidental upgrades
    [CRITICAL] Pod NetworkPolicy
        · The pod does not have a matching network policy
          Create a NetworkPolicy that targets this pod
    [CRITICAL] Pod Probes
        · Container is missing a readinessProbe
          Without a readinessProbe Services will start sending traffic to this
          pod before it's ready
        · Container is missing a livenessProbe
          Without a livenessProbe kubelet can not restart the Pod if it has
          crashed
    [CRITICAL] Container Security Context
        · foo -> Container has no configured security context
          Set securityContext to run the container in a more secure context.
networking.k8s.io/v1/NetworkPolicy foo-all-ok-netpol in testspace           ✅
v1/Pod pod-test-1 in testspace                                              ✅
> 
```

Source: (zegl, 2025)

FIGURE 2. SAMPLE OUTPUT FROM THE TOOL

Initial Assessment

The initial analysis revealed several issues:

- Missing resource limits - All Deployments lacked resource limits, increasing the risk of resource contention.
- Containers running as root - Default configurations allowed containers to run with root privileges.
- No health checks - Liveness and readiness probes were missing, making failures harder to detect.

Improvements

Based on kube-score recommendations, the following changes were implemented:

- Defined CPU and memory limits for all containers.
- Set the security context to run containers as non-root users.
- Added liveness and readiness probes critical services.

Results

After remediation, a follow-up run of kube-score indicated all critical and warning issues were resolved. The tool's actionable feedback improved the application's security posture and operational resilience.

kube-score proved effective for identifying both security-related misconfigurations and general best practice deviations. While it does not replace runtime security tools or policy enforcement frameworks, it serves as a valuable component in a defense-in-depth strategy. Integrating kube-score into CI/CD pipelines ensures continuous compliance with Kubernetes best practices.

Limitations

- Static analysis cannot catch run-time specific issues.
- Custom checks may be needed for organization-specific policies.

4. Conclusion

Static analysis using kube-score is a practical approach to proactively improving the security and reliability of Kubernetes deployments. Future work includes extending checks for custom resources, automating remediation suggestions, and integrating with broader policy management solutions.

The evolution of container security reflects a fundamental shift from perimeter-based approaches to comprehensive, lifecycle-oriented security strategies for dynamic, distributed environments. As evidenced throughout the article, effective container security requires coordinated implementation of controls across multiple layers: build-time vulnerability scanning, runtime behavioral monitoring, network segmentation, access control, and compliance automation. Organizations that successfully integrate these controls into their development and operational workflows achieve both security and agility benefits (Potla, 2025). This paper specifically emphasizes strengthening build-time security.

This is a tool that you should use to combat security vulnerabilities on a daily basis. Mitigation of the risk will avoid unnecessary risk and defects which might have impact on cost, schedule and quality. Kube-score will surely improve your productivity and workflow as it audits, secures and validates files you pass to it, and bundles multiple useful scans. Kube-score can be configured to run as part of your CI/CD pipeline meaning that anything that is misconfigured or gets flagged due to not matching a particular security framework can halt your deployment and allow developers to fix the issue before it becomes a problem.

5. Acknowledgements

Thanks to Swapna Anumolu and Vasavi Yeka for support with prototyping phase. Special thanks to Tony Hansen for reviewing the paper and providing valuable input.

References

Gurajapu, A. (2026a). *Best practices for monitoring Kubernetes clusters: Reliability and minimise operational overhead*. World Journal of Advanced Engineering Technology and Sciences, *18*(1), 7–15. **https://doi.org/10.30574/wjaets.2026.18.1.0002**

Gurajapu, A. (2026b). *Leveraging artificial intelligence to bridge execution gaps in SAFe®-scaled agile based programs*. World Journal of Advanced Engineering Technology and Sciences, *18*(1), 1–6. **https://doi.org/10.30574/wjaets.2026.18.1.1585**

Gurajapu, A. (2026c). *Orchestrating adaptive resilience and continuity restoration in cloud-native environments* (with A. Agarwal). *International Journal of Inventions in Engineering & Science Technology, 12*(1), 1–6. **https://doi.org/10.37648/ijiest.v12i01.001**

Gurajapu, A. (2026d). *Secure runtime encryption of critical source-code functions for IP protection*. *World Journal of Advanced Research and Reviews, 29*(1), 734–737. **https://doi.org/10.30574/wjarr.2026.29.1.0079**

Gurajapu, A. (2026e). *Shift-left security validation of containers via Kubernetes admission webhook*. *Frontiers in Computer Science and Artificial Intelligence, 5*(2), 63–68. **https://doi.org/10.32996/jcsts.2026.5.1.6**

Gurajapu, A. (2025a, December). *Static analysis of Kubernetes object definitions using kube-score: Enhancing security and resilience*. *European Journal of Information Technologies and Computer Science*. **https://www.researchgate.net/publication/386984115** (Preprint; no formal DOI)

Gurajapu, A. (2025b). *Swap Kubernetes secrets without application disruption: Comparative study and eBPF-powered kernel interception framework*. *World Journal of Advanced Engineering Technology and Sciences, 18*(1), 66–70. **https://doi.org/10.30574/wjaets.2026.18.1.0005**

Gurajapu, A. (n.d.). *Towards a futuristic security roadmap: Advanced strategies*. *Journal of Computer Science and Technology Studies, 8*(1), 31–39. **https://doi.org/10.32996/jcsts.2025.8.1.2**

Introduction | Gatekeeper. (2024). GitHub. **https://open-policy-agent.github.io/gatekeeper/website/docs/**

kubesec.io. (2019). **https://kubesec.io/**

kube-score: Kubernetes object analysis with recommendations for improved reliability and security. (2025). **https://kube-score.com/**

Potla, S. (2025). The evolution of container security in Kubernetes environments. *World Journal of Advanced Research and Reviews, 26*(2), 2361. **https://doi.org/10.30574/wjarr.2025.26.2.1741**

zegl. (2025, April 28). *kube-score* [GitHub repository]. GitHub. **https://github.com/zegl/kube-score**

About Author

The key author works for AT&T and has extensive work experience on DevOps, SDN, Cloud Computing, Artificial Intelligence, Cybersecurity which aligns with the citation.

Amar Gurajapu is Principal Member of Technical Staff at AT&T. Amar has 25 years of experience in Telecom Software Engineering. He is leading Cyber Security and Key Digital initiatives for key Network systems portfolio aligned with AT&T organization goals