

AI-Assisted Multi-Cluster Kubernetes Governance: A Secure, Resilient, and Policy-Driven Framework for Large-Scale Cloud Infrastructure Management

¹Bhanu Kiran Kumar Muggalla

¹Platform Engineer, DevOps, Kubernetes/OpenShift, Cloud Infrastructure, and AI for Infrastructure Automation

*Computer Information Systems, University of Central Missouri, USA
Email: bhanukiran71@gmail.com*

Abstract

The adoption of Kubernetes across hybrid-cloud, multi-cloud, and edge environments has shifted infrastructure management from isolated clusters to distributed multi-cluster systems. This transition improves capacity and fault isolation but creates governance problems involving fragmented visibility, policy drift, inconsistent access controls, security misconfiguration, and complex recovery decisions. This study develops an AI-assisted governance framework for secure, resilient, and policy-driven management of multi-cluster Kubernetes infrastructure. The work is positioned as a conceptual design-science study. It derives architectural requirements from a structured narrative review, threat analysis, and comparison of existing orchestration, security, policy-as-code, AIOps, and controller-assurance approaches. The proposed framework integrates a central governance plane, distributed policy enforcement, cross-cluster telemetry, anomaly and failure-risk analysis, compliance assessment, workload-resilience controls, human approval for high-risk actions, and auditable rollback. A reproducible validation protocol is specified for future implementation, including baseline definitions, data partitions, model configuration, threat and failure injection, repeated trials, statistical analysis, privacy controls, and reporting requirements. Published Kubernetes-security experiments are used only as external benchmark evidence and are not presented as results of the proposed framework. The analysis shows that the principal contribution is functional integration: deterministic policies remain authoritative, AI prioritizes uncertain operational evidence, and human oversight constrains consequential remediation. The framework provides a practical reference architecture for organizations that require consistent governance across heterogeneous Kubernetes estates while preserving local enforcement and administrative accountability.

Keywords: Kubernetes; multi-cluster governance; artificial intelligence; policy-as-code; cloud security; infrastructure resilience

1. Introduction

1.1 Background of the Study

Kubernetes has become a major platform for deploying and managing containerized applications. Its declarative control model reflects lessons from large-scale systems such as Borg, where schedulers and controllers continuously reconcile actual infrastructure with a desired state (Burns et al., 2016; Verma et al., 2015). This model supports automated placement, scaling, service discovery, and recovery, but it also makes configuration quality and controller behaviour central governance concerns.

Large organizations increasingly operate several clusters across private data centres, public clouds, geographical regions, and edge sites. Multi-cluster deployment can provide regulatory separation, capacity expansion, provider independence, and fault containment. It also fragments identities, policies, monitoring data, network controls, and administrative responsibilities. Research on Kubernetes scheduling and container orchestration identifies scalability, workload placement, interoperability, and resource efficiency as continuing problems (Carrión, 2022; Rodriguez & Buyya, 2019; Zhong et al., 2022). Geo-distributed studies additionally show that network delay, partitions, and static controller settings can destabilize federated services (Tamiru et al., 2020).

Governance coordinates these distributed environments through cluster registration, identity management, policy distribution, compliance monitoring, workload placement, incident response, and audit. Existing multi-cluster platforms improve orchestration and mobility, including mck8s, liquid computing, and cross-cluster scaling frameworks (Dumitru-Guzu et al., 2024; Iorio et al., 2023; Tamiru et al., 2021). Monitoring systems such as SLATE and Acala improve distributed visibility (Carcassi et al., 2020; Huang & Pierre, 2023). These advances are important, but orchestration and observability alone do not provide an integrated security and compliance model.

Artificial intelligence for IT operations, commonly described as AIOps, can analyse logs, metrics, traces, and alerts at a scale that is difficult to handle manually. Its applications include anomaly detection, failure prediction, incident classification, and root-cause support (Dang et al., 2019; Notaro et al., 2021). Machine learning has also been applied to workload scheduling, resource management, and autoscaling (Mao et al., 2019; Qiu et al., 2020, 2023; Rossi et al., 2019; Rządca et al., 2020). In governance, however, AI should not replace deterministic policy or administrative accountability. Its appropriate role is to prioritize uncertain evidence, estimate risk, and recommend actions within explicit policy and human-approval boundaries.

1.2 Problem Statement

Multi-cluster governance remains fragmented. Policies may differ across clusters because of separate administrators, cloud-provider controls, application requirements, and deployment pipelines. Insecure manifests, excessive privileges, exposed interfaces, vulnerable images, and weak network rules can therefore be reproduced across several environments. Empirical research has found recurring security misconfigurations in public Kubernetes manifests, including absent resource limits and unsafe security settings (Rahman et al., 2023; Shamim, 2021). Container isolation also does not remove risks affecting images, runtimes, hosts, registries, and orchestrators (Combe et al., 2016; Souppaya et al., 2017; Sultan et al., 2019).

Operational evidence is commonly distributed among separate monitoring platforms. A security team may identify suspicious activity in one cluster without recognizing related events elsewhere. Static rules are valuable for known violations but may not detect emerging runtime behaviour. Manual triage further delays containment and recovery, while unrestricted automation can create outages when evidence is incomplete. The research problem is therefore how to combine consistent policy enforcement, cross-cluster observability, intelligent risk analysis, resilient recovery, and accountable human control within one governance architecture.

1.3 Aim, Objectives, and Research Questions

The study aims to design and analytically assess an AI-assisted framework for secure, resilient, and policy-driven multi-cluster Kubernetes governance. Its objectives are to develop a centralized governance architecture with local enforcement, integrate explainable anomaly and failure-risk analysis, support continuous compliance, coordinate resilient workload recovery, and specify a reproducible experimental protocol for later validation.

The study addresses four questions:

1. How can AI improve governance across multiple Kubernetes clusters?
2. How can policy-as-code support consistent security and compliance?
3. Through what mechanisms could AI-assisted remediation reduce detection and recovery time?
4. What performance and operational costs must be measured when the framework is implemented?

1.4 Research Contributions

The study contributes an integrated reference architecture, a separation-of-authority model in which policy remains deterministic while AI provides risk-ranked recommendations, a human-in-the-loop response process with rollback and audit, and a validation protocol that defines baselines, data requirements,

statistical tests, privacy controls, and reproducibility artefacts. It does not claim that the framework has already achieved empirical performance improvements.

2. Literature Review

2.1 Kubernetes Governance and Multi-Cluster Management

Kubernetes separates the control plane from worker nodes. Controllers observe resources through the application programming interface and reconcile them toward declared configurations. Namespaces, role-based access control, service accounts, quotas, admission control, labels, and network policies provide governance primitives, but their effectiveness depends on consistent implementation. Scheduling studies show that placement decisions must balance resource utilization, latency, availability, and application constraints (Carrión, 2022; Zhong et al., 2022).

Multi-cluster management extends these responsibilities across administrative and network boundaries. mck8s coordinates placement and scaling across geo-distributed clusters, while liquid computing creates dynamic Kubernetes topologies across otherwise fragmented resources (Iorio et al., 2023; Tamiru et al., 2021). Cross-cluster scaling has also been studied in cloud-native 5G deployments (Dumitru-Guzu et al., 2024). Nevertheless, distributed control introduces synchronization delays, partial failure, and divergent configuration. Tamiru et al. (2020) demonstrate that communication delays and static federation settings can produce instability, which supports the need for local policy enforcement and failure-tolerant governance.

2.2 Security and Policy-as-Code

Kubernetes security risks arise from misconfigured manifests, excessive permissions, vulnerable images, compromised credentials, exposed services, unsafe host access, and malicious runtime behaviour. Rahman et al. (2023) identified 1,051 misconfigurations in 2,039 manifests, illustrating the scale of configuration risk. Security guidance therefore emphasizes image provenance, least privilege, host protection, runtime controls, network restriction, and continuous audit (Shamim et al., 2020; Souppaya et al., 2017).

Policy-as-code represents organizational requirements as version-controlled, testable rules. Admission controls can reject privileged containers, unapproved image registries, missing resource limits, or invalid labels before deployment. Open Policy Agent and Gatekeeper support parameterized policies, audit, and Kubernetes admission control, while GitOps provides review and distribution mechanisms (Open Policy Agent, n.d.). Automated inter-service policy generation also demonstrates how communication requirements can be converted into fine-grained access controls (Li et al., 2021). Static policies remain

necessary but insufficient because runtime attacks and novel behaviour may satisfy manifest rules while still creating risk.

2.3 AI-Assisted Operations and Security

AIOps research applies statistical and machine-learning methods to operational data. DeepLog detects anomalous log sequences, LogRobust addresses evolving log data, and MicroRCA and Seer use telemetry to support root-cause localization and performance diagnosis (Du et al., 2017; Gan et al., 2019; Wu et al., 2020; Zhang et al., 2019). KubAnomaly applies neural networks to container system calls and runtime events, while recent Kubernetes research combines multi-class threat detection with adaptive deception (Aly et al., 2025; Tien et al., 2019). These studies support the feasibility of learned detection, but they also show that performance depends on datasets, features, attack classes, and operating conditions.

AI has also been used for resource orchestration. FIRM, AWARE, Autopilot, reinforcement-learning scalers, and learned schedulers predict demand or select resource actions (Mao et al., 2019; Qiu et al., 2020, 2023; Rossi et al., 2019; Rządca et al., 2020). Their contribution is mainly performance-oriented. Security governance requires additional controls because a mistaken placement or remediation decision can affect confidentiality, regulatory compliance, or service availability.

2.4 Resilience and Controller Assurance

Controllers can contain safety and liveness defects even when their intended policies are correct. Sieve tests controller reliability, Acto performs end-to-end testing of operator correctness, Kivi model-checks controller behaviour, and Anvil verifies eventual stable reconciliation (Gu et al., 2023; Liu et al., 2024; Sun et al., 2022, 2024). These works indicate that governance automation must include validation, bounded authority, and rollback. AI-generated recommendations should therefore pass deterministic policy checks and recovery verification before an incident is closed.

2.5 Related Approaches and Research Gap

Table 1 compares representative approaches using dimensions selected from the study requirements: governance scope, AI support, policy enforcement, security monitoring, resilience, and accountable remediation. The dimensions were chosen because they correspond directly to the functions required by the research questions, rather than serving as a numerical quality ranking.

Table 1 *Comparative analysis of representative Kubernetes governance approaches*

Study or approach	Primary scope	AI support	Policy enforcement	Security monitoring	Resilience or assurance	Main limitation for this study
-------------------	---------------	------------	--------------------	---------------------	-------------------------	--------------------------------

Kubernetes and Borg foundations (Burns et al., 2016; Verma et al., 2015)	Cluster control and scheduling	No direct governance AI	Declarative state	Operational	Reconciliation	No integrated cross-cluster compliance layer
mck8s and liquid computing (Iorio et al., 2023; Tamiru et al., 2021)	Geo-distributed orchestration	Limited	Orchestration policies	Limited	Workload mobility	Security analytics and accountable remediation are not central
SLATE and Acala (Carcassi et al., 2020; Huang & Pierre, 2023)	Distributed monitoring	No	No direct enforcement	Strong observability	Indirect	Visibility without integrated policy response
NIST and manifest-security studies (Rahman et al., 2023; Souppaya et al., 2017)	Container and configuration security	No	Guidance and validation	Configuration-focused	Limited	Limited predictive resilience
Sieve, Acto, Kivi, and Anvil (Gu et al., 2023; Liu et al., 2024; Sun et al., 2022, 2024)	Controller testing and verification	No	Correctness properties	Reliability-focused	Strong assurance	Primarily development-time assurance
AWARE, FIRM, and Autopilot (Qiu et al., 2020, 2023; Rzedca et al., 2020)	Resource management	Yes	Limited	Performance-focused	Predictive scaling	Does not integrate security compliance and policy governance

The literature offers strong components but limited integration of cross-cluster telemetry, AI-assisted security analysis, policy-as-code, compliance scoring, predictive resilience, explainable recommendations, and human-controlled remediation. The proposed framework addresses this integration gap without claiming that architectural integration alone proves superior empirical performance.

3. Proposed AI-Assisted Governance Framework

3.1 Design Principles

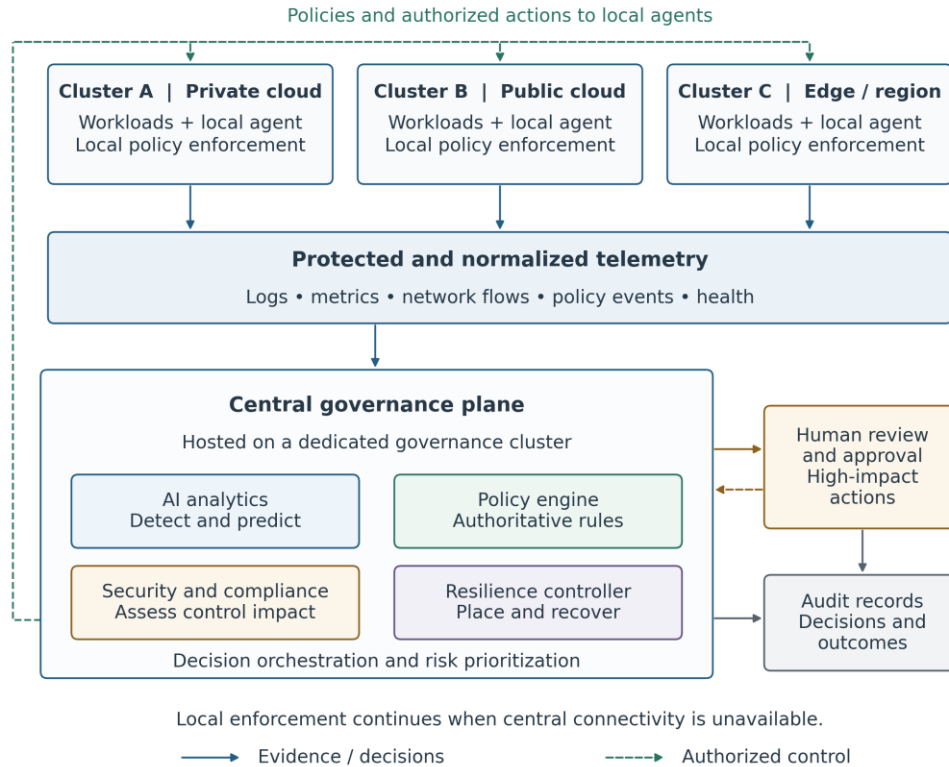
The framework follows seven principles. Central visibility provides a unified view of cluster health, workload state, policy violations, and security events. Distributed enforcement allows each cluster to continue applying policy during disconnection from the central governance plane. Zero-trust control requires authenticated identities, least privilege, encrypted communication, and continuous authorization. Explainable AI associates every recommendation with contributing features, confidence, affected resources, and policy context. Human oversight governs high-impact actions. Automated recovery handles only bounded, pre-authorized responses. Interoperability relies on Kubernetes APIs, portable telemetry, and version-controlled policies.

3.2 Architecture

Each participating cluster runs a local governance agent for telemetry collection, policy evaluation, secure communication, and execution of approved actions. The central governance plane maintains cluster registration, policy versions, cross-cluster context, risk analysis, approval workflows, and audit records. This design avoids making local enforcement dependent on continuous central connectivity. The central governance plane denotes the logical coordination services. The governance cluster denotes the dedicated Kubernetes cluster that hosts these services; the terms do not describe the same architectural layer.

The observability layer collects audit logs, application logs, network flows, resource metrics, policy decisions, runtime alerts, and health signals. The AI analytics engine normalizes events, extracts features, estimates anomaly and failure probabilities, and produces an evidence-backed recommendation. The policy engine remains authoritative for admission, configuration, identity, network, and data-location rules. A compliance module maps violations to organizational controls. The resilience controller evaluates health, capacity, dependency, security posture, and placement constraints before recommending replication, relocation, or failover.

Figure 1 Architecture of the proposed AI-assisted multi-cluster Kubernetes governance framework



3.3 Observability and Data Processing

Audit logs record API identities, actions, objects, outcomes, and timestamps. Network telemetry identifies unexpected communication or exfiltration paths. Metrics describe saturation, restarts, scheduling failure, control-plane latency, and node readiness. Policy and runtime engines supply structured violations and security events. Records are time-synchronized, deduplicated, assigned provenance, and transformed into a common schema. Sensitive identifiers are pseudonymized before model training unless operational investigation requires controlled re-identification.

Feature groups include failed authentication rate, privilege changes, policy-violation frequency, unusual destinations, deviation from workload resource baselines, restart bursts, node-health changes, and correlated events across clusters. The design supports both supervised classifiers and unsupervised detectors because labeled attacks may be scarce. Model output never directly overrides an explicit deny policy.

3.4 Policy, Compliance, and Risk Decisions

Policies are reviewed, tested, signed, and distributed through a GitOps repository. Admission policies evaluate proposed resources, while periodic audit detects drift in running resources. Exceptions require an

owner, justification, scope, and expiration date. Table 2 links representative governance areas to evidence, deterministic rules, AI support, and bounded responses.

Table 2 *Governance policies, risk indicators, and bounded responses*

Governance area	Risk indicator	Deterministic policy	AI support	Bounded response
Identity and access	Repeated failures or unusual administrative action	Least privilege and approved role bindings	Identity-behaviour anomaly score	Suspend the session or request investigation
Workload security	Privileged container or host access	Reject prohibited security contexts	Configuration-risk prioritization	Block admission and explain the violation
Image security	Unapproved registry or critical vulnerability	Require signed images from approved registries	Image-risk prioritization	Quarantine pending replacement
Network security	Unexpected external or lateral traffic	Enforce approved communication paths	Network anomaly detection	Isolate the workload subject to risk policy
Resource governance	Unusual CPU or memory growth	Require requests, limits, and quotas	Demand and saturation prediction	Recommend scaling or redistribution
Compliance	Missing encryption, labels, or audit controls	Apply mandatory control profiles	Compliance scoring and drift correlation	Block deployment or create remediation work
Cluster health	Node failure trend or rising control-plane latency	Maintain availability thresholds	Failure-risk estimation	Prepare replication or failover
Data location	Placement outside an approved region	Restrict location through labels and admission rules	Placement-risk assessment	Recommend compliant relocation with approval

For transparent decision-making, the proposed framework combines five normalized inputs into the risk-priority index R :

$$R = 0.35p + 0.20s + 0.15c + 0.15q + 0.15i$$

Here, R is the overall risk-priority index; p is the calibrated probability of an anomalous or unsafe event; s is deterministic event severity; c is the criticality of affected workloads; q is the affected-cluster scope; and i is the impact on applicable compliance controls. Every input lies in $[0, 1]$. The weights assign 35% to p , 20% to s , and 15% each to c , q , and i . They are non-negative and sum to 1, so R also lies in $[0, 1]$. R is a prioritization index, not a calibrated probability of harm.

For the reference design, s , c , and i use the fixed mapping none = 0, low = 0.25, moderate = 0.50, high = 0.75, and critical = 1.00. These categories are assigned respectively from a versioned rule-severity

catalogue, a workload-criticality register, and a compliance-control impact catalogue. Where several rules, workloads, or controls are affected, the highest applicable category is used. Unknown is treated as missing, not as none. Scope is $q = a/K$, where a is the number of affected workload clusters and K is the number of registered workload clusters in the declared governance scope; the governance host cluster is excluded. K must be positive.

If any required input is missing, invalid, or outside its defined range, the numerical score is withheld and the event is labelled “insufficient evidence”. The system records the unavailable inputs and requests human triage while continuing all deterministic policy enforcement. Missing risk components are neither set to zero nor compensated for by redistributing weights. This decision-level rule is distinct from the feature imputation specified in Section 4.3.

Classification uses the unrounded score: low risk when $R < 0.40$; medium risk when $0.40 \leq R < 0.70$; and high risk when $R \geq 0.70$. Medium-risk events generate recommendations for review. Only predefined, authorized, reversible low-risk actions may execute automatically. The input mappings, weights, and thresholds are proposed architectural defaults to be tested and calibrated during implementation, not experimentally optimized values.

For a worked example, let $p = 0.80$, $s = 0.75$, $c = 1.00$, $i = 0.50$, and let two of three workload clusters be affected, giving $q = 2/3$. Then $R = 0.35(0.80) + 0.20(0.75) + 0.15(1.00) + 0.15(2/3) + 0.15(0.50) = 0.755$. The event is therefore high risk. This calculation illustrates the rule and is not an experimental result.

Let $D = 1$ when a versioned deterministic policy identifies a critical violation, and $D = 0$ otherwise. If $D = 1$, the final risk class is high regardless of R , including when R cannot be computed. The system preserves the original score or missing-score status and records the overriding rule separately. A mandatory deny rule blocks the prohibited request independently of AI. The override does not authorize consequential remediation: high-impact actions still require human approval, even if the numerical score is low.

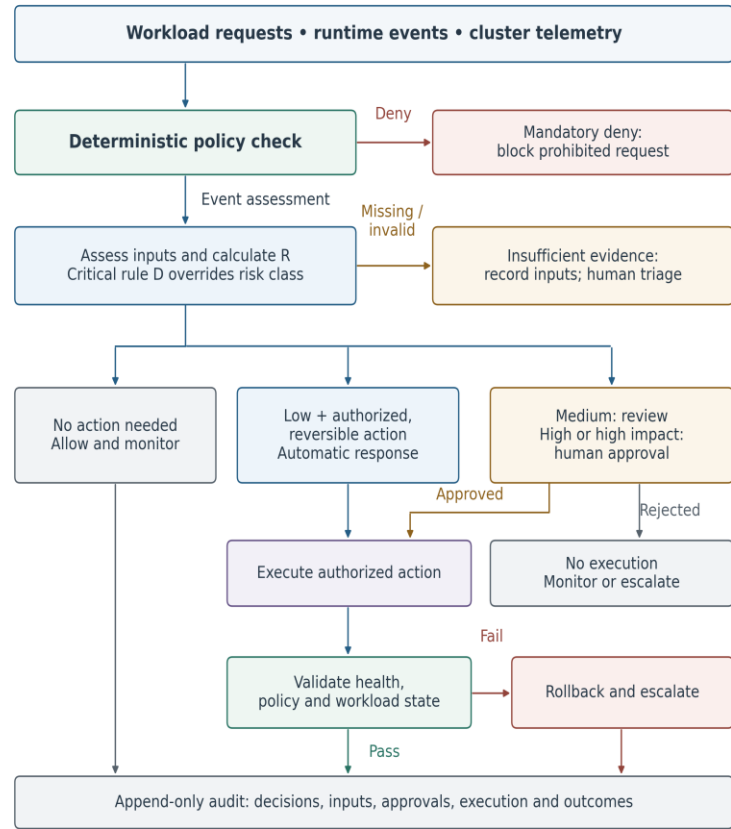
3.5 Resilience and Human-in-the-Loop Response

The resilience controller evaluates capacity, latency, storage state, dependency health, security posture, and geographical constraints. It may recommend replication, rescheduling, traffic redirection, or restoration. Low-risk actions can be automated only when they are predefined, reversible, and authorized, such as adding a missing label or restarting an unhealthy noncritical pod. High-risk actions, including revoking administrative credentials, isolating a production cluster, moving regulated data, deleting workloads, or initiating regional failover, require an authorized human decision.

The approval interface presents evidence, confidence, affected resources, expected effects, policy constraints, and rollback conditions. Before execution, the system records the current configuration and

relevant workload state. Post-action validation checks service health, policy compliance, data availability, and dependency communication. Failure triggers rollback and escalation. Every detection, recommendation, approval, rejection, execution, validation, and rollback is retained in an append-only audit record.

Figure 2 *AI-assisted policy-enforcement and incident-response workflow*



Note. Deterministic deny rules remain authoritative. Only authorized, reversible low-risk actions may execute automatically. Medium-risk recommendations require review, and high-impact actions require human approval regardless of score. Missing evidence does not disable deterministic enforcement or a critical-rule override. All branches, including denied requests and human triage, are audited.

4. Research Methodology and Validation Protocol

4.1 Study Design and Methodological Status

The study uses design-science research to construct and analytically assess a governance artefact. Activities include problem identification, requirement derivation, architecture design, threat analysis, comparison with related approaches, and specification of an empirical validation protocol. No prototype, original

telemetry dataset, controlled multi-cluster experiment, or framework-specific performance result is claimed in this article. Published measurements are used only to establish external benchmarks and design targets.

4.2 Literature Selection

A structured narrative search was conducted across ACM Digital Library, IEEE Xplore, USENIX proceedings, SpringerLink, Wiley Online Library, NIST publications, and Google Scholar. Search combinations included “Kubernetes multi-cluster governance,” “Kubernetes security misconfiguration,” “policy-as-code admission control,” “AIOps anomaly detection,” “Kubernetes controller verification,” “container autoscaling,” and “geo-distributed Kubernetes.” English-language peer-reviewed studies published from 2015 through March 2025 were prioritized. Authoritative standards and official living documentation were considered separately; access dates for living documentation are reported in the reference list. Studies were included when they addressed at least one framework requirement and provided identifiable methods or technical architecture. Duplicates, marketing material, nontechnical commentary, and sources without clear relevance were excluded. Thirty-five scholarly and standards sources plus the official Open Policy Agent documentation were retained. The process supports a reasoned design synthesis but is not presented as a systematic review or exhaustive meta-analysis.

4.3 Reference AI Implementation

The validation protocol fixes an initial implementation so that later experiments can be replicated. Events are grouped into 60-second windows advanced every 10 seconds. Numeric missing values are imputed using training-set medians, categorical fields are one-hot encoded, and continuous variables are standardized using training statistics. Experimental runs, rather than individual windows, are assigned to training, validation, and test partitions in a 70:15:15 ratio to prevent leakage.

The reference classifier is XGBoost with a binary logistic objective, 300 trees, maximum depth 6, learning rate 0.05, subsample 0.80, column sample 0.80, L2 regularization 1.0, and random seed 42. Class imbalance is handled through a training-derived positive-class weight; validation and test sets remain unchanged. The decision threshold is chosen on validation data to maximize F1 subject to a false-positive rate no greater than 5%. Probability calibration uses isotonic regression fitted only on validation data. The final report must identify library versions, hardware, training time, selected threshold, tuning range, feature definitions, and feature importance. This is a proposed reference configuration, not a trained model reported by the present study.

4.4 Proposed Experimental Environment and Baselines

The validation design uses one dedicated governance cluster to host the central governance plane and three workload clusters representing private-cloud, public-cloud, and edge conditions. All software versions, manifests, hardware specifications, network conditions, and workload rates must be frozen before testing and published with the results.

Table 3 *Proposed validation environment and reporting requirements*

Component	Reference configuration	Required reporting
Governance cluster (central governance plane host)	Three nodes, each at least 4 vCPU, 16 GB RAM, and 100 GB storage	Kubernetes distribution/version, OS, runtime, hardware, storage
Workload clusters A and B	Four nodes per cluster for web/API and transactional services	Requests per second, resource requests, replicas, dataset size
Edge cluster C	Three constrained nodes for batch and analytical workloads	Architecture, CPU/RAM limits, intermittent-link profile
Policy and security	Gatekeeper or Kyverno, Falco, and Cilium	Exact versions, policies, rule sets, enforcement mode
Observability and GitOps	Prometheus, Grafana, audit logging, and a GitOps controller	Retention, scrape interval, reconciliation interval, clock synchronization
Network	Configurable inter-cluster latency, bandwidth, loss, and partitions	Measured round-trip time, bandwidth, loss, and impairment procedure
Threat injection	Test identities, privileged-pod attempts, controlled drift, stress workload, and blocked network flows	Script, start/end markers, expected label, cleanup procedure
Failure injection	Kubelet/node shutdown and isolated control-plane interruption	Exact command, affected component, duration, restoration method
Dataset report	Windowed telemetry separated by run and cluster	Total records, duration, normal/anomalous counts, class distribution, missingness, and split counts

Manual governance receives the same dashboards and raw alerts as the other approaches, but a trained administrator follows a fixed runbook without automated classification. Rule-based governance receives the same telemetry and uses predefined admission, Falco, network, and GitOps rules without a learned model. AI-assisted governance adds the reference classifier, risk score, and bounded response workflow. Scenario order is randomized, and all three approaches use identical workloads and event injections.

Each of seven scenarios is repeated 30 times per governance approach after five warm-up runs. Primary outcomes are detection time, policy-decision latency, recovery time, recovery success, availability, CPU, memory, network overhead, throughput, precision, recall, F1, and false-positive rate. Results are reported using medians, interquartile ranges, means where appropriate, and 95% bootstrap confidence intervals. Repeated-measures comparisons use the Friedman test, followed by paired Wilcoxon tests with Holm

correction. Effect size is reported using Cliff’s delta. Detection outcomes are compared with paired contingency analysis. These procedures are specified for future validation and no resulting statistics are reported here.

4.5 Data Governance and Reproducibility

Validation should use synthetic identities, artificial workloads, and isolated test clusters. Direct identifiers in audit and network records are replaced with rotating pseudonyms before model development. Raw telemetry is encrypted in transit and at rest, restricted through role-based access control, and retained only for the approved test period. A provenance manifest records cluster, sensor, policy version, scenario, and transformation history for every dataset partition.

A reproducibility repository should contain infrastructure-as-code, Kubernetes manifests, policy definitions, workload generators, threat and failure scripts, feature code, random seeds, model configuration, data dictionaries, and analysis notebooks. If raw events cannot be released, the repository should provide an anonymized derived dataset or a deterministic generation procedure.

5. Analytical Assessment and Published Benchmark Evidence

5.1 Requirements Traceability

The framework was assessed by tracing each governance requirement to an architectural mechanism and a future empirical test. This confirms internal coverage but does not prove operational superiority.

Table 4 *Requirements traceability and validation status*

Requirement	Framework mechanism	Present evidence	Future empirical measure
Cross-cluster visibility	Common telemetry schema and central correlation	Architectural traceability and prior monitoring research	Event completeness, ingestion delay, and cross-cluster correlation accuracy
Policy consistency	Versioned central policy with local admission and audit	Policy workflow and OPA/GitOps capabilities	Enforcement success, deployment latency, and drift duration
Threat prioritization	Calibrated anomaly probability and transparent risk score	Published anomaly-detection evidence	Precision, recall, F1, false-positive rate, and detection time
Resilient recovery	Health assessment, placement constraints, validation, and rollback	Design analysis and controller-assurance literature	Relocation time, recovery time, success rate, and availability
Accountable automation	Risk thresholds, human approval, explanation, and audit	Workflow inspection	Approval time, override rate, rollback rate, and explanation usefulness

Operational feasibility	Distributed agents and bounded central services	Architecture analysis	CPU, memory, network, throughput, and policy latency
-------------------------	---	-----------------------	--

5.2 Published AI Detection Benchmarks

Table 5 summarizes measurements reported by two peer-reviewed Kubernetes-security studies. Tien et al. (2019) analysed system calls and container behaviour, whereas Aly et al. (2025) used network-flow features for multi-class detection. Their datasets and experimental conditions differ, so the values are not a controlled comparison and are not results of the proposed framework. The four system-call/container-behaviour rows are drawn from Table 8 of the former study (complex dataset with all-event features), and the three Naive Bayes configurations are drawn from Table 2 of the latter.

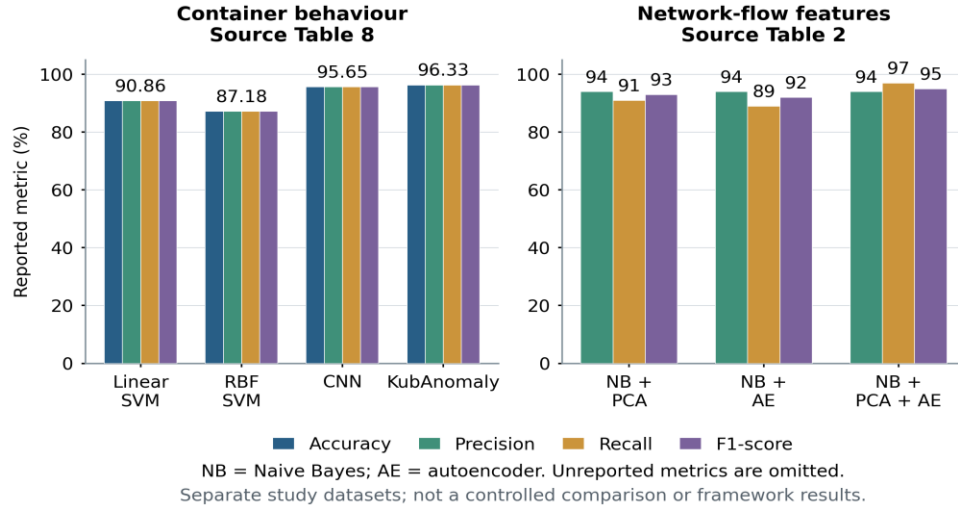
Table 5 *Published AI model performance in Kubernetes-security experiments*

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	False-positive rate (%)
Linear SVM (Tien et al., 2019)	90.86	90.86	90.86	90.86	NR
RBF SVM (Tien et al., 2019)	87.18	87.18	87.18	87.18	NR
Convolutional neural network (Tien et al., 2019)	95.65	95.65	95.65	95.65	NR
KubAnomaly neural network (Tien et al., 2019)	96.33	96.33	96.33	96.33	NR
Naive Bayes with PCA (Aly et al., 2025)	NR	94.00	91.00	93.00	NR
Naive Bayes with autoencoder (Aly et al., 2025)	NR	94.00	89.00	92.00	NR
Naive Bayes with PCA and autoencoder (Aly et al., 2025)	NR	94.00	97.00	95.00	NR

Note. NR means not reported in the cited source table, not zero. Detection accuracy of up to 91% and an average false-positive rate of 5.5% are reported by Aly et al. (2025) outside the configuration table; these contextual summaries are not shown as configuration-specific entries. Precision, recall, and F1-score in the latter study are converted from proportions to percentages. The earlier study reports AUC, which is not a false-positive rate.

KubAnomaly reported stronger aggregate classification than the tested SVM configurations, but its attack-level performance varied. This supports per-class reporting rather than reliance on overall accuracy. Aly et al. (2025) reported high recall for the combined PCA and autoencoder configuration, illustrating the potential security value of feature transformation. The reported 5.5% average false-positive rate also shows why high-impact actions should not be triggered solely by model output.

Figure 3 Published Kubernetes AI detection benchmarks from Tien et al. (2019) and Aly et al. (2025). The panels correspond to source Tables 8 and 2, respectively; unreported configuration metrics are omitted.



5.3 Answers to the Research Questions

RQ1: How can AI improve multi-cluster governance? AI can correlate telemetry across clusters, identify deviations that static rules do not express, estimate failure risk, group related alerts, and prioritize incidents. Its contribution is analytical assistance, not independent authority. Empirical improvement remains to be tested using the protocol in Section 4.

RQ2: How can policy-as-code support consistency? A signed, version-controlled policy source can distribute common requirements while local admission controllers and audit engines enforce them during disconnection. GitOps reconciliation and time-limited exceptions make drift visible and accountable. Consistency must later be measured through enforcement success, propagation latency, and drift duration.

RQ3: Through what mechanisms could AI-assisted remediation reduce detection and recovery time? The proposed mechanisms are continuous event analysis, failure-risk estimation, pre-authorized reversible actions, prepared workload-placement decisions, and verified recovery with rollback. Monitoring and prediction could shorten detection, while prepared actions could shorten response and restoration. Policy checks, human approval, and state transfer may introduce additional delays. These are architectural mechanisms and testable expectations; the present conceptual study establishes neither a numerical reduction nor a causal effect. Their effectiveness must be evaluated using the protocol in Section 4.

RQ4: What performance and operational costs must be measured when the framework is implemented? Telemetry transport, policy evaluation, feature processing, inference, and audit storage will

consume CPU, memory, network capacity, and admission latency. Distributed enforcement may reduce central bottlenecks, but no magnitude can be claimed before implementation. The protocol therefore measures overhead under identical baseline workloads and reports uncertainty.

6. Discussion

6.1 Interpretation and Security Implications

The analytical assessment indicates that the framework’s value lies in coordinating capabilities that are often deployed separately. Policy-as-code provides deterministic prevention and drift detection. AI supports uncertain runtime analysis and cross-cluster correlation. Resilience controls connect detection to placement and recovery. Human approval prevents a probabilistic model from exercising unrestricted authority. This separation addresses a central tension in AIOps: faster decisions are useful only when their operational and security consequences remain bounded.

Continuous telemetry could improve the identification of privilege escalation, abnormal API use, suspicious network communication, and unexpected resource demand. Admission control could reduce configuration errors before deployment, while audit detects later drift. These controls align with container-security recommendations concerning least privilege, configuration management, runtime monitoring, and auditability (Souppaya et al., 2017). Zero-trust enforcement is supported through verified identities, restricted communication, short-lived authorization, and context-sensitive policy decisions.

The design also introduces privacy and integrity obligations. Audit logs and network flows may contain identities, service relationships, and business-sensitive metadata. A governance platform that centralizes this information becomes a high-value target. Pseudonymization, data minimization, retention limits, provenance, encryption, and restricted access are therefore part of the security architecture rather than optional operational practices.

6.2 Operational and Management Implications

The central governance plane could reduce the effort required to inspect clusters individually and could make policy ownership, exceptions, and incident status easier to track. Local enforcement preserves basic control when wide-area connectivity fails. The human-in-the-loop process allows routine, reversible actions to be automated while administrators retain control of changes that could affect production availability, regulated data, or privileged access.

These benefits involve trade-offs. Telemetry and inference introduce processing and network cost. Stricter admission policies may delay deployments or reject legitimate exceptions. Human approval reduces automation risk but may slow urgent response. The framework therefore treats approval time, override

frequency, rollback frequency, explanation usefulness, and resource overhead as first-class evaluation measures.

6.3 Comparison with Previous Studies

KubAnomaly and the framework of Aly et al. (2025) show that learned methods can detect Kubernetes-related threats, but they focus primarily on detection and response within their experimental settings. The proposed architecture extends the scope to policy distribution, compliance mapping, cross-cluster placement, audit, and approval. mck8s and liquid computing address multi-cluster orchestration but do not make AI-assisted security governance their principal function (Iorio et al., 2023; Tamiru et al., 2021). Acala and SLATE improve visibility without providing the full decision and remediation chain (Carcassi et al., 2020; Huang & Pierre, 2023).

Controller-assurance research adds an important constraint. Acto, Kivi, Sieve, and Anvil demonstrate that controllers can fail to reach or preserve desired states (Gu et al., 2023; Liu et al., 2024; Sun et al., 2022, 2024). Consequently, an AI recommendation cannot be considered successful merely because an action was issued. Recovery must be verified against health, policy, dependency, and data checks, with rollback when those conditions fail.

6.4 Practical Applications

Financial institutions could apply the framework to access control, regional placement, transaction-service continuity, and audit evidence. Healthcare organizations could use policy profiles to protect clinical systems while restricting telemetry containing sensitive information. Telecommunications providers could coordinate service placement and recovery across edge and regional clusters. Government agencies could standardize controls across sovereign, private, and public cloud environments. Large digital platforms could use cross-cluster risk correlation and bounded remediation to govern geographically distributed services. In every case, production adoption requires domain-specific policy, threat modelling, and independent validation.

7. Limitations and Future Research

This article presents a conceptual design-science artefact, not an experimentally validated system. No original telemetry dataset was collected, no model was trained, and no causal performance comparison was conducted. The architecture therefore cannot substantiate numerical claims about detection accuracy, policy consistency, recovery time, availability, or overhead. Published benchmark values are included only to demonstrate the range and reporting practices of related Kubernetes-security studies.

The literature synthesis is structured but narrative. It does not provide the protocol, dual screening, or exhaustive coverage expected of a systematic review. The architecture may also behave differently across cloud providers, Kubernetes distributions, edge devices, network conditions, and regulatory environments. Training data may contain incomplete labels or fail to represent new attacks. False positives could interrupt legitimate workloads, while human approval could delay urgent recovery. Centralized telemetry raises privacy, confidentiality, provenance, and retention concerns.

Future work should implement the reference testbed and publish repeated measurements, confidence intervals, effect sizes, confusion matrices, and per-class results. External validation should include different cloud providers and larger cluster estates. Federated learning could reduce the need to centralize raw telemetry. Reinforcement learning may support constrained placement after safety testing. Large language models could assist policy drafting and explanation, but generated rules would require formal validation and human review. Further research should examine privacy-preserving analytics, explainable remediation, controller verification, policy-conflict detection, adversarial robustness, and resource-constrained edge and Internet of Things clusters.

8. Conclusion

Distributed Kubernetes estates create governance problems that isolated cluster administration cannot address consistently. Policies can drift, telemetry can remain fragmented, runtime threats can escape static configuration checks, and recovery decisions can cross security, availability, and regulatory boundaries. The research problem is therefore not simply how to automate more actions, but how to coordinate policy, intelligence, resilience, and accountability across heterogeneous clusters.

This study develops an AI-assisted governance framework that combines a central governance plane with local enforcement, common telemetry, deterministic policy-as-code, anomaly and failure-risk analysis, compliance assessment, resilience controls, human approval, rollback, and audit. Its principal contribution is the separation of responsibilities. Explicit policy remains authoritative; AI analyses uncertain evidence and ranks risk; the resilience controller proposes bounded actions; and authorized administrators retain control of consequential remediation.

The article is intentionally positioned as a conceptual design-science study. It does not present illustrative values as measured findings. Published Kubernetes-security results are treated only as external benchmarks, and questions concerning detection speed, recovery, policy consistency, availability, and overhead remain hypotheses for empirical testing. To support that next stage, the study specifies an initial model configuration, data partitions, threat and failure scenarios, equivalent baselines, repeated trials, statistical procedures, privacy controls, and reproducibility materials.

The framework offers practical value as a reference architecture for financial services, healthcare, telecommunications, government infrastructure, and large digital platforms. These environments require consistent governance without removing local cluster autonomy or human accountability. The proposed design provides a defensible basis for implementation and evaluation, but its operational benefits must be demonstrated through transparent experiments before performance or effectiveness claims are made.

References

- Aly, A., Hamad, A. M., Al-Qutt, M., & Fayez, M. (2025). Real-time multi-class threat detection and adaptive deception in Kubernetes environments. *Scientific Reports*, 15, Article 8924. <https://doi.org/10.1038/s41598-025-91606-8>
- Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016). Borg, Omega, and Kubernetes. *Communications of the ACM*, 59(5), 50–57. <https://doi.org/10.1145/2890784>
- Carcassi, G., Breen, J., Bryant, L., Gardner, R. W., McKee, S., & Weaver, C. (2020). SLATE: Monitoring distributed Kubernetes clusters. In *Practice and Experience in Advanced Research Computing 2020: Catch the Wave* (pp. 19–25). Association for Computing Machinery.
- Carrión, C. (2022). Kubernetes scheduling: Taxonomy, ongoing issues and challenges. *ACM Computing Surveys*, 55(7), Article 138. <https://doi.org/10.1145/3539606>
- Combe, T., Martin, A., & Di Pietro, R. (2016). To Docker or not to Docker: A security perspective. *IEEE Cloud Computing*, 3(5), 54–62. <https://doi.org/10.1109/MCC.2016.100>
- Dang, Y., Lin, Q., & Huang, P. (2019). AIOps: Real-world challenges and research innovations. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings* (pp. 4–5). IEEE. <https://doi.org/10.1109/ICSE-Companion.2019.00023>
- Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1285–1298). Association for Computing Machinery. <https://doi.org/10.1145/3133956.3134015>
- Dumitru-Guzu, O.-M., Vlădeanu, C., & Kooij, R. (2024). A novel framework for cross-cluster scaling in cloud-native 5G NextGen core. *Future Internet*, 16(9), Article 325. <https://doi.org/10.3390/fi16090325>
- Gan, Y., Zhang, Y., Hu, K., Cheng, D., He, Y., Pancholi, M., & Delimitrou, C. (2019). Seer: Leveraging big data to navigate the complexity of performance debugging in cloud microservices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages*

- and Operating Systems* (pp. 19–33). Association for Computing Machinery. <https://doi.org/10.1145/3297858.3304004>
- Gu, J. T., Sun, X., Zhang, W., Jiang, Y., Wang, C., Vaziri, M., Legunsen, O., & Xu, T. (2023). Acto: Automatic end-to-end testing for operation correctness of cloud system management. In *Proceedings of the 29th Symposium on Operating Systems Principles* (pp. 96–112). Association for Computing Machinery. <https://doi.org/10.1145/3600006.3613161>
- Huang, C.-K., & Pierre, G. (2023). Acala: Aggregate monitoring for geo-distributed cluster federations. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing* (pp. 156–164). Association for Computing Machinery. <https://doi.org/10.1145/3555776.3577716>
- Iorio, M., Risso, F., Palesandro, A., Camiciotti, L., & Manzalini, A. (2023). Computing without borders: The way towards liquid computing. *IEEE Transactions on Cloud Computing*, 11(3), 2820–2838. <https://doi.org/10.1109/TCC.2022.3229163>
- Li, X., Chen, Y., Lin, Z., Wang, X., & Chen, J. H. (2021). Automatic policy generation for inter-service access control of microservices. In *30th USENIX Security Symposium* (pp. 3971–3988). USENIX Association. <https://www.usenix.org/conference/usenixsecurity21/presentation/li-xing>
- Liu, B., Lim, G., Beckett, R., & Godfrey, P. B. (2024). Kivi: Verification for cluster management. In *2024 USENIX Annual Technical Conference* (pp. 509–527). USENIX Association. <https://www.usenix.org/conference/atc24/presentation/liu-bingzhe>
- Mao, H., Schwarzkopf, M., Venkatakrishnan, S. B., Meng, Z., & Alizadeh, M. (2019). Learning scheduling algorithms for data processing clusters. In *Proceedings of the ACM Special Interest Group on Data Communication* (pp. 270–288). Association for Computing Machinery. <https://doi.org/10.1145/3341302.3342080>
- Notaro, P., Cardoso, J., & Gerndt, M. (2021). A survey of AIOps methods for failure management. *ACM Transactions on Intelligent Systems and Technology*, 12(6), Article 81. <https://doi.org/10.1145/3483424>
- Open Policy Agent. (n.d.). *OPA for Kubernetes admission control*. Retrieved August 21, 2026, from <https://www.openpolicyagent.org/docs/kubernetes>
- Qiu, H., Banerjee, S. S., Jha, S., Kalbarczyk, Z. T., & Iyer, R. K. (2020). FIRM: An intelligent fine-grained resource management framework for SLO-oriented microservices. In *14th USENIX Symposium on Operating Systems Design and Implementation* (pp. 805–825). USENIX Association. <https://www.usenix.org/conference/osdi20/presentation/qiu>

- Qiu, H., Mao, W., Wang, C., Franke, H., Youssef, A., Kalbarczyk, Z. T., & Iyer, R. K. (2023). AWARE: Automate workload autoscaling with reinforcement learning in production cloud systems. In *2023 USENIX Annual Technical Conference* (pp. 387–402). USENIX Association. <https://www.usenix.org/conference/atc23/presentation/qiu-haoran>
- Rahman, A., Shamim, S. I., Bose, D. B., & Pandita, R. (2023). Security misconfigurations in open source Kubernetes manifests: An empirical study. *ACM Transactions on Software Engineering and Methodology*, 32(4), Article 99. <https://doi.org/10.1145/3579639>
- Rodriguez, M. A., & Buyya, R. (2019). Container-based cluster orchestration systems: A taxonomy and future directions. *Software: Practice and Experience*, 49(5), 698–719. <https://doi.org/10.1002/spe.2660>
- Rossi, F., Nardelli, M., & Cardellini, V. (2019). Horizontal and vertical scaling of container-based applications using reinforcement learning. In *2019 IEEE 12th International Conference on Cloud Computing* (pp. 329–338). IEEE.
- Rzadca, K., Findeisen, P., Swiderski, J., Zych, P., Broniek, P., Kusmierk, J., Nowak, P., Strack, B., Witusowski, P., Hand, S., & Wilkes, J. (2020). Autopilot: Workload autoscaling at Google. In *Proceedings of the Fifteenth European Conference on Computer Systems* (Article 16). Association for Computing Machinery. <https://doi.org/10.1145/3342195.3387524>
- Shamim, S. I. (2021). Mitigating security attacks in Kubernetes manifests for security best practices violation. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 1689–1690). Association for Computing Machinery.
- Shamim, S. I., Bhuiyan, F. A., & Rahman, A. (2020). XI commandments of Kubernetes security: A systematization of knowledge related to Kubernetes security practices. In *2020 IEEE Secure Development Conference* (pp. 58–64). IEEE.
- Souppaya, M., Morello, J., & Scarfone, K. (2017). *Application container security guide* (NIST Special Publication 800-190). National Institute of Standards and Technology. <https://doi.org/10.6028/NIST.SP.800-190>
- Sultan, S., Ahmad, I., & Dimitriou, T. (2019). Container security: Issues, challenges, and the road ahead. *IEEE Access*, 7, 52976–52996. <https://doi.org/10.1109/ACCESS.2019.2911732>
- Sun, X., Luo, W., Gu, J. T., Ganesan, A., Alagappan, R., Gasch, M., Vaziri, M., Legunsen, O., Padon, O., & Xu, T. (2022). Automatic reliability testing for cluster management controllers. In *16th USENIX*

- Symposium on Operating Systems Design and Implementation* (pp. 143–159). USENIX Association. <https://www.usenix.org/conference/osdi22/presentation/sun>
- Sun, X., Ma, W., Gu, J. T., Ma, Z., Chajed, T., Howell, J., Lattuada, A., Padon, O., Suresh, L., Szekeres, A., & Xu, T. (2024). Anvil: Verifying liveness of cluster management controllers. In *18th USENIX Symposium on Operating Systems Design and Implementation* (pp. 649–666). USENIX Association. <https://www.usenix.org/conference/osdi24/presentation/sun>
- Tamiru, M. A., Pierre, G., Tordsson, J., & Elmroth, E. (2020). Instability in geo-distributed Kubernetes federation: Causes and mitigation. In *2020 28th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems* (pp. 1–8). IEEE. <https://doi.org/10.1109/MASCOTS50786.2020.9285934>
- Tamiru, M. A., Pierre, G., Tordsson, J., & Elmroth, E. (2021). mck8s: An orchestration platform for geo-distributed multi-cluster environments. In *2021 International Conference on Computer Communications and Networks* (pp. 1–10). IEEE. <https://doi.org/10.1109/ICCCN52240.2021.9522318>
- Tien, C.-W., Huang, T.-Y., Tien, C.-W., Huang, T.-C., & Kuo, S.-Y. (2019). KubAnomaly: Anomaly detection for the Docker orchestration platform with neural network approaches. *Engineering Reports*, 1(5), Article e12080. <https://doi.org/10.1002/eng2.12080>
- Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., & Wilkes, J. (2015). Large-scale cluster management at Google with Borg. In *Proceedings of the Tenth European Conference on Computer Systems* (Article 18). Association for Computing Machinery. <https://doi.org/10.1145/2741948.2741964>
- Wu, L., Tordsson, J., Elmroth, E., & Kao, O. (2020). MicroRCA: Root cause localization of performance issues in microservices. In *2020 IEEE/IFIP Network Operations and Management Symposium* (pp. 1–9). IEEE. <https://doi.org/10.1109/NOMS47738.2020.9110353>
- Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., Xie, C., Yang, X., Cheng, Q., Li, Z., Chen, J., He, X., Yao, R., Lou, J.-G., Chintalapati, M., Shen, F., & Zhang, D. (2019). Robust log-based anomaly detection on unstable log data. In *Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 807–817). Association for Computing Machinery. <https://doi.org/10.1145/3338906.3338931>
- Zhong, Z., Xu, M., Rodriguez, M. A., Xu, C., & Buyya, R. (2022). Machine learning-based orchestration of containers: A taxonomy and future directions. *ACM Computing Surveys*, 54(10s), Article 217. <https://doi.org/10.1145/3510415>