

Kubernetes Security Best Practices

MSc Research Project
Cloud Computing

Ritesh Kumar Rout
Student ID: 21127069

School of Computing
National College of Ireland

Supervisor: Prof. Sai Ranjan Emani

**National College of Ireland
Project Submission Sheet
School of Computing**



Student Name:	Ritesh Kumar Rout
Student ID:	21127069
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Prof. Sai Ranjan Emani
Submission Due Date:	12/12/2024
Project Title:	Kubernetes Security Best Practices
Word Count:	6850
Page Count:	22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Ritesh Kumar Rout
Date:	12th December 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Kubernetes Security Best Practices

Ritesh Kumar Rout

21127069

Abstract

The security and adaptation that comes along with Kubernetes, the systems used to build containerized apps and microservices, is what this research examines. However, as more and more industries adopt Kubernetes, it also comes with additional complexities and opens up whole new areas of risk to security, from the vulnerabilities in container images to misconfigured policies. Adding to the complexity is the fact that many organizations use Kubernetes in conjunction with popular cloud platforms like AWS, which requires them to effectively address two types of security challenges the cloud-native and the Kuber-native. Related works identify important threats, show how microservices increase complexity, and propose new tools or techniques to improve Kubernetes security. The objective of this work is to evaluate the security of Kubernetes systems as well as existing and future security approaches. Thematic analysis revealed the key security issues such who are deploying these systems, what are the top insecure practices such as RBAC misconfigurations, container image vulnerabilities, network security, complexity of microservices, continuous monitoring, and zero-trust-based policy frameworks. The research indicated considerable shortcomings in existing security approaches, although work is being done to remedy issues surrounding vulnerabilities created by misconfigurations and imperfect network policies. Besides theoretical aspects, practical implementation was done on the live environment of Kubernetes with the actual approach and practices for implementing security on deploy. Important implementation steps were addressed, like configuring RBAC, securing container images, applying network policies, and configuring monitoring with Prometheus. The work provides insights into Kubernetes security, which is valuable for organizations looking to enhance the security of their environments. Organizations can be better equipped with a more secure, scalable and resilient containerized infrastructure by remediating the identified weaknesses and implementing strong security measures.

Keywords: Kubernetes, container security, microservices, RBAC, network policies, AWS (Amazon Web Services), zero-trust security, Prometheus, security vulnerabilities, policy configuration, cloud security, security strategies.

1 Introduction

In the area of modern application deployment, containerization plays an important role, especially in a microservices context. This containerization tool called Kubernetes is now widely used for organizing, deploying, and managing container applications in various fields. That it can solve complicated deployments and enable scalability has made it invaluable. However, with this rapid adoption comes a critical challenge that works well

in helping secure Kubernetes environments. The nature of Kubernetes itself, as well as the layered microservice architecture of the stack, bring forth a whole series of security issues, which are not present to the same extent in simpler systems.

1.1 Background and Motivation

Kubernetes has become a popular platform for organizations to more effectively run the processes, and secure management of these settings provides a crucial issue. Kubernetes helps to manage containerized applications in industries of finance, healthcare, e-commerce, and others, but the growth of its use indicates that several organizations do not understand or do not implement appropriate security measures. A cluster in Kubernetes has very specific security requirements for different components such as the network policy, authentication, the method of dealing with container images, and runtime security Shamim (2021). The structure with nested layers added to the interdependency typical of microservices architecture implies that hacking one part of the system can affect the entire environment. Furthermore, since established cloud providers including AWS include managed Kubernetes as services, that is Kubernetes Service, organizations should consider native Kubernetes security issues as well as cloud-hosted issues. This research work aims to fill the existing literature void by discussing the key security issues that organizations experience when deploying Kubernetes. As a result, the threats and potential barriers to and trends in Kubernetes security, as discussed in this research, are useful for academic and industrial beneficiaries. These findings can be useful for both security teams within organizations and cloud suppliers who offer Kubernetes as these can show areas of risk and weaknesses, methods to prevent exploitation, and changes in best practices which could improve readiness against emerging threats.

1.2 Research Question

What is happening today in terms of Kubernetes security issues? How does the inherent organizational complexity of the microservices in K8s affect security and best practices? Which issues can be considered as the most frequent ones in the case of the Kubernetes security enhancement? What are those developments that show that best practices or the tools applied to Kubernetes security are evolving?

1.3 Objective

- To categorize and examine critical security threats within and across industries that employ Kubernetes.
- To study the effect that the complexity of microservices has on security ideas and approaches in Kubernetes settings.
- To find out more about the constant issues that organizations encounter in the application of Kubernetes security considerations.
- To evaluate new trends and changes in the preferences or tools that are useful to maintain and improve the security of the AWS cloud.

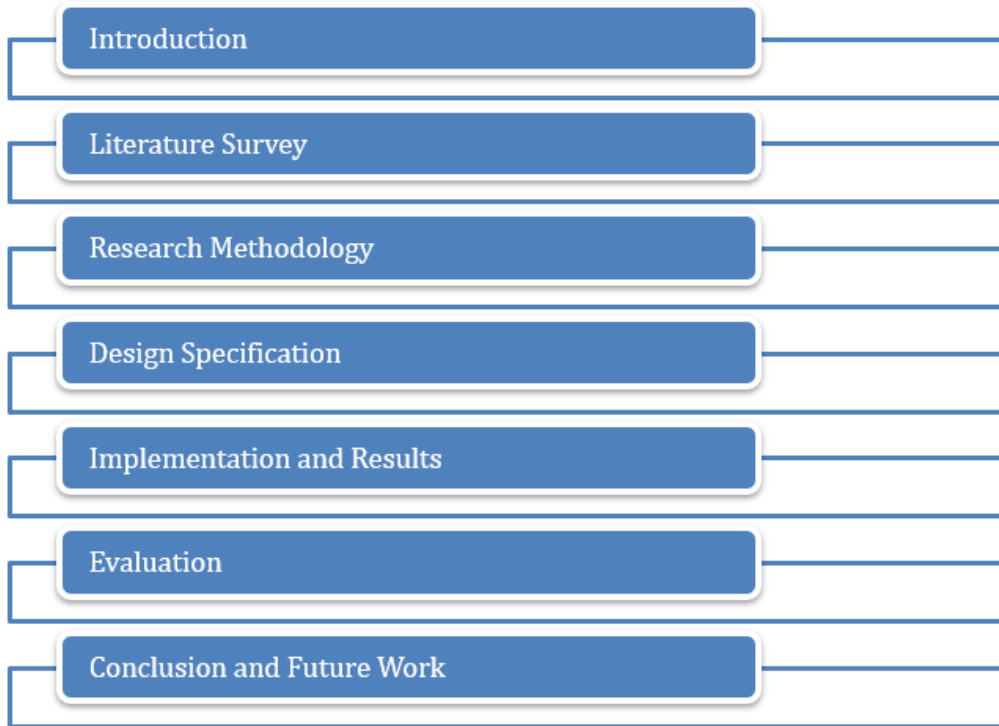


Figure 1: Research Structure

1.4 Research Structure

This report Figure 1 starts with an introduction to create the significance and purpose of the report. Conducting a review of the literature acts as a building block, where we assess the previous work and highlight where some gaps are in the research. Methodology covers the tools and techniques used to solve the research problem, while the Design and Implementation section describes the system/framework that we have developed. The results are precisely examined to determine the relevance of these results to the methodology. An assessment may probe deeper into the results, identifying strengths, limitations, and opportunities for improvement. A synthesis of key findings, implications, and future directions is provided in the conclusion, indicative of the significance of contributions made by this research.

2 Literature Survey

2.1 Overview of Kubernetes and its Security Concerns

Kubernetes is famous due to its strong automation functions and for the microservices architecture where software applications are split into finer services. Since more and more organizations embrace Kubernetes, there is a growing demand to mitigate the security threats of using complex distributed environments. As a result of the structure of microservices, different components can be developed and deployed individually and more flexibly than in monolithic applications, which in turn creates several security issues, consisting of the emergence of new surfaces that can be attacked and secure communic-

ation between the services Shamim et al. (2020). Furthermore, since established cloud providers including AWS include managed Kubernetes as services, that is Kubernetes Service, organizations should consider native Kubernetes security issues as well as cloud-hosted issues. Research has shown that each layer of a Kubernetes environment is a critical layer that needs protection since the attack on one layer compromises the remaining layers.

2.2 Microservices, Architecture, and Security Consequences

The basics of Kubernetes are based on the microservices strategy where big applications are split into sizeable partitions that confer over networks, and as a result, the incidences of security threats such as unauthorized access, information leakage, and network flaws are more likely to occur. A recent transition to microservice architecture implies that established security practices, ideal for monolithic architectures, do not suffice. Security requirements do rise in Kubernetes as the environment evolves quite frequently, often with services being changed, relocated, or duplicated Dell’Imagine et al. (2023). This dynamism poses considerable challenges when setting up an organization’s general security policies that must work uniformly across the architectural fabric. Thus, according to current industry practices, there is a so-called “defense-in-depth” approach, the essence of which is to apply several layers of protection for the operational environment of containers, networking, and workload. Kubernetes’ microservices architecture makes scalability and flexibility but makes it inherently more complex when it comes to doing security González (2023). Each microservice runs in an isolated container in a Kubernetes environment, thus minimizing cross-service contamination, but there is a need to manage the security aspects of network segmentation, authentication, and identity verification across microservices in an isolated way. The problem is how to enforce consistent rules regarding security across many microservices, which often have distinct security needs and may require different levels of access. Containerized microservices can also magnify the ‘container drift’ risk, where changes in container images relative to security guidelines can occur over time and must be continuously monitored and enforced with policy to keep the components secure Theodoropoulos et al. (2023) and Jayalath et al. (2024). Service meshes and consistent runtime policies can improve interservice security, yet they also require more involved network security.

2.3 Popular Kubernetes Threats

Studies done on Kubernetes threats have always highlighted areas that pose threats and need attention regardless of the industrial domain. Some of the threats include inaccurate configuration of Role-Based Access Control (RBAC), container image vulnerability, and insecure network policies. This is because the improper configurations of RBAC grant specific users access to privileged data, or controls in the Kubernetes cluster Mustyala and Tatineni (2021). RBAC implementation allows for fine-grained control over the user rights with a relatively low probability of obtaining undesired access and coming across opportunities for achieving a multitude of unauthorized privileges Kutsa (2024). Thus, only approved and updated container images are sent to the application, minimizing risks and hiding insidious security flaws that are inherent in the application’s nature. Since these groups split up the communication amongst microservices, network rules enhance security since the attack surface is reduced. On the other hand, one must ensure that

network policies are well designed to deny attackers a chance to access some services or networks Huseinović et al. (2020) and Onyema et al. (2022). Unvetted or vulnerable container images are also a high risk as code can be introduced within an image before an environment is put into use. Research focuses on how to perform image-scanning for weaknesses, implement an impenetrable principle of containment, and separate the workload to contain movement across a Kubernetes cluster. As it is a container orchestration nature, Kubernetes has its own specific threats that are associated with the risks of vulnerability of images, misconfiguration and cluster management. Often, unregulated access to Kubernetes APIs, or misconfigured RBAC settings, are used as attack vectors allowing attackers to execute privilege escalation and gain control over resources Kampa (2024) and Sroor et al. (2024). One threat is that networks are extremely permissive which allows unmonitored lateral movement inside the cluster. Another major threat is the case of image poisoning, where there is malicious code in the image, which, if left unchecked, defeats traditional security controls to escape the pipeline. To prevent these threats, image scanning, constraining RBAC, and auditing of Kubernetes configuration files are required using.

2.4 Security Threats and Issues in Deploying Kubernetes in Cloud Environment

Cloud service models, particularly shared responsibility models such as the implementation of Kubernetes in cloud environments like AWS, bring extra security concerns. Cloud providers are responsible for securing the underlying infrastructure, while Kubernetes configuration and users' access control still lie in the user's hands. This structure requires that workload(s) using native Kubernetes constructs within AWS EKS manage security at the Kubernetes layer side by side with AWS layers such as Identity and Access Management (IAM) Bose et al. (2021). Using AWS's managed services since these can help to simplify the management of Kubernetes environments and security, partly because of the utilization of tools such as Amazon Guard Duty, which is a service for threat detection, and AWS Identity and Access Management, also for access control purposes, and others like it, but these can only prove useful and align well with the targeted Kubernetes controls if the incorporation is done properly. Traditional threat identification and risk management techniques do not solve the problem because these procedures ensure that guarantees security measures change according to the growingly dynamic threat environment Arif et al. (2024) and Hosen et al. (2024). Also, businesses can enhance the quantity and creativity of the containerised systems by integrating those security solutions. As well as guarding private data, this coalescing of efforts toward solid and enveloped security frameworks enables more organisations embrace and capitalise on Kubernetes, within the cloud-native setting. Kubernetes deployments based in the cloud greatly increase the probability of risk due to their shared infrastructure, lower visibility, and oil company nuances towards the cloud. Cloud platforms can be multi-tenant, yet that exposes vulnerable points where other users' vulnerabilities can impact shared resources. Most solutions for unauthorized data access or cluster takeovers happen in improper Identity and Access Management (IAM) configurations, negligible encryption and exposed secrets in environments such as AWS or GCP Song et al. (2023). As with any technology, cloud resources also have their elasticity risks where scaling events can temporarily expose sensitive configurations or insecure default settings. To address these, it is necessary to put strict IAM policies in place, controls on the endpoints and integration

with cloud-native security policies for the apps within the Kubernetes.

2.5 New Security Trends and Security Solutions

Trends in the Kubernetes security domain include an uplift in the employment of policy as code and automated security scanning, which provides ways to implement policies consistently throughout the Kubernetes ecosystem. A modern trend for the CI/CD pipeline is the use of policy as code when practicing Open Policy Agent (OPA) as an open-source tool Agrawal (2024). These tools are especially useful where Kubernetes clusters are hosted in the cloud, the multi-cluster environment, or where thousands of clusters are managed because otherwise, it can be impossible to enforce identical policies across all clusters. Another trend is runtime security, tools that act similarly to antivirus programs, which monitor the container runtimes and alert security teams when something is amiss. There is also a move to adopt AI monitoring solutions since these solutions make use of machine learning algorithms to determine if there are signs of a security risk. The focus of security trends in Kubernetes is automation, policy as code, plus enhanced visibility into the DevSecOps pipeline Sandu (2021). Another notable trend is the use of policy as code, meaning that Kubernetes operations are automatically checked against well-defined security rules using tools such as Open Policy Agent (OPA) or Kyverno to apply code to the development environment, as well as to the deployment environment, such that configuration is checked against security rules in development and deployment Kim and Lee (2024). Another development is the use of AI-driven anomaly detection, which can alert before an anomaly shows up, anything from atypical behavior or even traffic patterns. As Kubernetes-native security tools such as KubeArmor continue to gain momentum, they offer runtime security, enforcement of workload isolation, and file and network anomaly monitoring without extensive manual configuration Zheng et al. (2024). Similarly, service meshes such as Istio are evolving to safeguard communications by default while empowering augmented access control and security observability.

2.6 Challenges of Applying Security in Kubernetes

Even having chosen prospective security tools, many organizations still face numerous challenges in the provision of successful Kubernetes security. Such barriers are usually associated with the effective configuration issues connected to Kubernetes, low knowledge of the Kubernetes security issues and the management, and inefficiency of the integration of Kafka security tools with the existing environment Kampa (2024). Flexibility configuration is often cited because organizations have a problem with understanding and managing Kubernetes with the use of multiple levels of settings. As it has been seen in the literature, there is a skills shortage in security specifically when it comes to Kubernetes with no suitable talent that specializes in cloud-native security. In addition, when it comes to security in Kubernetes, it is still complicated to easily integrate it with past security solutions and compliance measures, especially with authoritative regulations where industries such as finance and healthcare are restricted to use Thijsman et al. (2024). Several key issues in Kubernetes security have been brought out in the literature as worth addressing by organizations to ‘get it right’ in the environments. Kubernetes security is not a completed research issue with many works ongoing on best practices and techniques to secure microservices architectures. With more organizations adopting Kubernetes and cloud platforms such as AWS, the security risk is set to increase hence the

need to come up with new solutions and more importantly understand Kubernetes-native and cloud-native security Egbuna (2022). This research supports these efforts by presenting the key security topics, issues, and trends, on which more investigation and security solutions development in Kubernetes can be built. Since Kubernetes has a lot of maturity, elements such as robust security policies integrated directly into CI/CD pipelines are future-proof to prevent vulnerabilities from getting into production, embedding security in the development life cycle.

3 Research Methodology

This paper systematically reviews secondary data on Kubernetes security issues and best practices for the emerging technology through thematic analysis in quantitative research. This technique has been selected in order methodically to categorise and account for repeated patterns, problems, and solutions with reference to security. As there is emphasis drawn towards security trends and challenges peculiar to Kubernetes, a qualitative, thematic analysis allows for the exploration of complex security concerns within various organisational contexts. Therefore, this research sheds light on the practical implementation and makes a proper explanation between thematic and practical implementation. The research method has been introduced on the cloud AWS platform to make the implementation.

3.1 Scope Definition

Research focuses on certain aspects of Kubernetes security: network security, container runtime protection, API security, and the best practices have been chosen intentionally. This is due to the fact that these goals of the study include identifying and understanding the primary security concerns in Kubernetes systems as well as assessing how microservice complexity influences security processes in these systems.

3.2 Data Gathering

Information has been sourced solely from secondary sources based on a Kubernetes security targeted academic journal article, business reports, and best practice papers. Google Scholar and online books have been considered more appropriate because these are more credible sources in this area of study. Specific focus has been made on which threats are most common, what security issues Kubernetes users face, and what new initiatives and studies have appeared in the last five years. These limitations have made the study to be specific in the type of organisations to be used in the study, hence making the study to be closer to the current technological and industrial practices.

3.3 Data Organization and Cleaning

The data organization and cleaning stage which have been performed entailed a proper assessment and elimination of improper or the excessive data deriving from the overall sources attained. Upon critical evaluation, each source has been then categorized into broad areas of study, which have been in line with the research questions of the study. This research also engaged only high thematic importance materials, which made the analysis of themes much more effective, as the theme has been divided into such sections

as network security, API security, and new Kubernetes best practices. This particular arrangement of the material also helped in highlighting more of what has been repeated and miles more in terms of patterns or trends in the Kubernetes security. Ultimately, this methodical approach not only lets to examine but also examine the highly complex security environment observed in Kubernetes deployments more methodically.

3.4 Thematic Analysis Framework

As for the analysis of the collected data, the theme analysis, which allowed methodically search and study trends in the material considered in the literature on Kubernetes security has been used. This method enabled the study to gauge a plethora of links and categorize complex security-related subjects into themes that have been accord to the objectives of the research. It allowed employing thematic analysis to categorize all the studied material systematically, to focus on discussing emerging tools and practices in the context of Kubernetes, a set of critical security issues, and the influence of microservices complexity on security measures.

3.5 Kubernetes Security Implementation Methodology

This methodology is a vital intersection of theory and practice for Kubernetes security. This approach works by converting abstract security concepts into specific technical implementations by solving the problems in cloud-native security principles from an end-to-end viewpoint. In this research design, we utilize AWS EKS as a real-world environment to illustrate complicated security methods such as RBAC function, image authentication, and network policy parameters. The practical approach this method provides is evidence of security strategies studied in the literature review that guided us to identify Kubernetes security challenges. The implementation serves to validate theoretical models, demonstrate possible exploits, and serve as a reproducible framework for enterprises looking to improve the security posture of their containerized infra.

4 Design Specification

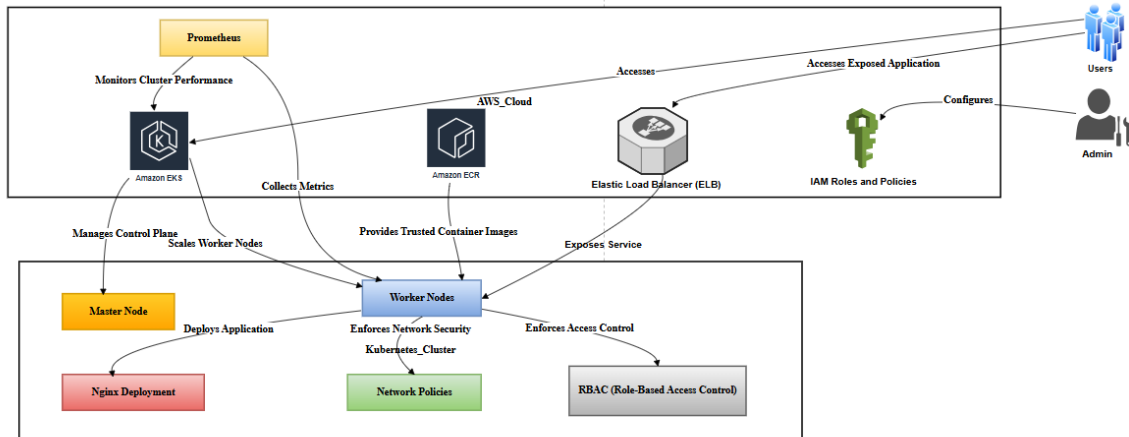


Figure 2: Architectural Daigram

The design of this research originates from the thematic analysis of the literature and numerous reports on Kubernetes security to answer certain research questions on current security issues of Kubernetes, the influence of organization complexity in the microservices, the cyclic problems and new tendencies of modern security standards and tools in Kubernetes environment. The practical application of the software will be performed on the cloud, levitating on Kubernetes platforms known among the major cloud providers including AWS. This approach uses the inherent advantage of cloud to emulate real-life Kubernetes implementation environments. The cloud environment will help in the proper coordination and management of containers for a better understanding of security issues or confusion related to Kubernetes in a cloud environment. This setup will also allow us to discover security topics associated with the container images, networking, and RBAC, which will give a comprehensive view of the security challenges in the modern microservices architecture. Secure, Scalable and Cost Optimized Kubernetes deployment on AWS architecture diagram 2. Essentially, Amazon EKS takes care of the Kubernetes control plane and the worker nodes which run the containerized apps such as Nginx. AWS IAM roles and policies control access to AWS resources, as well as Kubernetes components, in order to establish appropriate permissions. Container images are pulled from Amazon ECR where trusted images are stored and deployed to the worker nodes. Security is enforced by network policies (which enforce which pods can communicate with one another) and RBAC (which controls who can do what with which resources). Prometheus is a real-time monitoring and alerting framework that continuously watches the state of the cluster and collects real-time metrics to ensure efficient resource consumption. The application is accessed by external users through the Elastic Load Balancer (ELB) which guarantees high availability and spreads traffic on worker nodes. This architecture combines Kubernetes and AWS services to deliver a secure and reliable environment for containerized applications.

5 Implementation

5.1 Thematic Analysis and its Results

This thematic analysis uses existing literature from academia and industries that describes specific security perspectives within Kubernetes to address the issues more specific to the COS(Container-Optimized-OS) and AI handling of microservices Curtis and Eisty (2024). The report discussed numerous key security issues like access management, network policies, and container image security as well as new trends in security practices policies, and tools that point to the maturity of Kubernetes security. Kubernetes security concerns are discussed in this paper in the context of the multiple layers that emerge when environments are containerized and interact with the specific intricacies of microservices. Drawing from relevant literature review and case study, the study explores the basic thematic areas including access management, network policies, and container image security discussing the current problems, patterns and evolving solutions Ibryam and Huß (2022). It follows the logical layer model for design, decomposing the Kubernetes platform into significant components, including the container runtime, networking, and monitoring to determine the spectrum of security effects. The deployment of the regarded software will take place in a Kubernetes-based cloud environment that will require the usage of such clouds like AWS, Azure and GCP. This environment also provides an opportunity to run as well as coordinate and control microservices within many containers to replic-

ate real-life deployment. The cases based on cloud infrastructure allow for the required scaling and the availability of copies of production environments that are useful when identifying the security challenges of Kubernetes. Security components including RBAC and network policies as well as container image vulnerabilities will thus be incorporated into the cloud-hosted Kubernetes architecture to simulate real-world security problems and analyze the current transformational security tools in this type of architecture. The implementation adopts parts from academic and industry reports that explain primary, popular misconfigurations, and new emerging trends in the use of security in cloud-hosted Kubernetes environments.

The thematic analysis of the study revealed six foundational themes as significant components and emerging trends in securing Kubernetes. Some of the themes discussed include insufficient RBAC configuration, container image vulnerabilities, network security problems, microservice complexity, constant monitoring, and the rise of new policies. The following is a brief of each theme and a table comparing the key points on the theme, issues, and emerging practices.

- Theme 1: Insufficient RBAC Configurations RBAC misconfigurations are prevalent in Kubernetes and its misuse results in high privilege levels and high levels of vulnerabilities. In particular, RBAC works fine in managing and controlling the user's permissions, but these become rigid and complicated when implemented and monitored and end up causing access inappropriately Tripathi (2024). Automation solutions like Open Policy Agent (OPA) present approaches for auditing and enforcing to address those misconfigurations.
- Theme 2: Container image vulnerabilities Figure 3 containers are derived from image files that are usually downloaded from a public platform and may contain vulnerabilities that affect the application. Most images do not reflect updates while others contain inherent security vulnerabilities that open the entire cluster for attack Agrawal et al. (2020). The current best practices in Kubernetes security have shifted to only running trusted images, frequently updating, and incorporating image scanning tools.

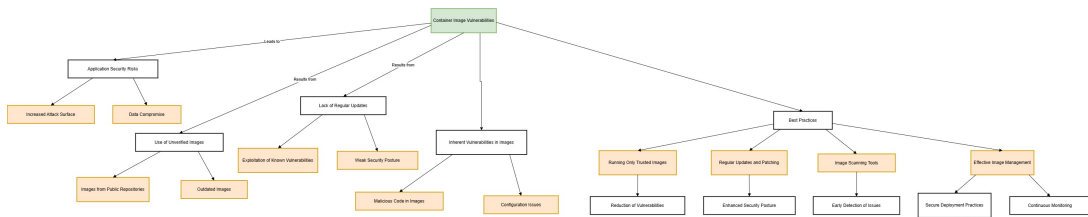


Figure 3: Container Image Vulnerabilities and Mitigation Strategies

- Theme 3: Network Security Challenges Security function as always remains a critical issue in the Kubernetes area since the Microservices can need to communicate a lot 4. Then there are cases where network policies are inadequate services may be left objectively unguarded thus posing a threat due to unauthorized access. Network segmentation matrix and strong implementation of network policies have been noted to be among the key preventive measures that should be implemented. Systems like Calico have been developed to allow organizations to set tight network policies to allow communication within the clusters.

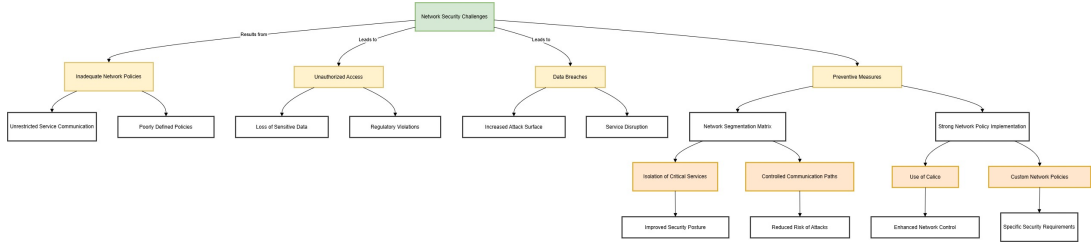


Figure 4: Network Security Challenges in Kubernetes

- Theme 4: Microservices Complexity figure 5 explains microservices architecture of Kubernetes adds inherent complexity and expands the attack surface Muresu (2021). Every microservice is independent and it needs security on different levels of its infrastructure. This complexity makes access control a structured process and requires strict mounted monitoring and efficient management practices. It is to meet these challenges that more systems that can track dependencies and analyze vulnerabilities across distributed services are deployed as security tools.

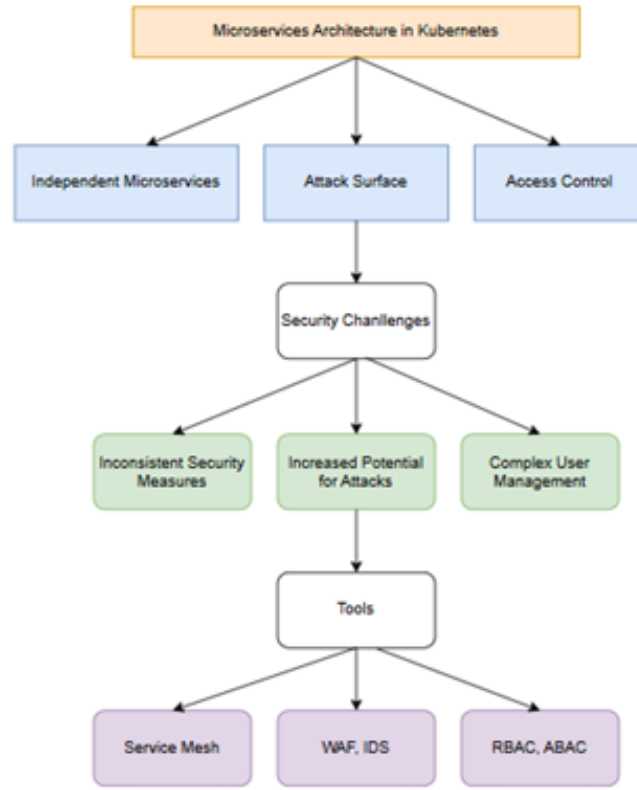


Figure 5: Microservices Architecture of Kubernetes

- Theme 5: Need for Continuous Monitoring and Logging figure 6 Kubernetes security demands are best managed by constant monitoring owing to the dynamism expected in containerized environments Revuelta Martinez (2023). Metrics tools like Prometheus and Logging solutions provide detailed coverage concerning system functionality and give the administrator an indication of potentially malicious

activities (Revuelta Martinez, 2023). Constant observation guarantees that a threat is identified as soon as possible, which is paramount given the novelty and outsized nature of Kubernetes.

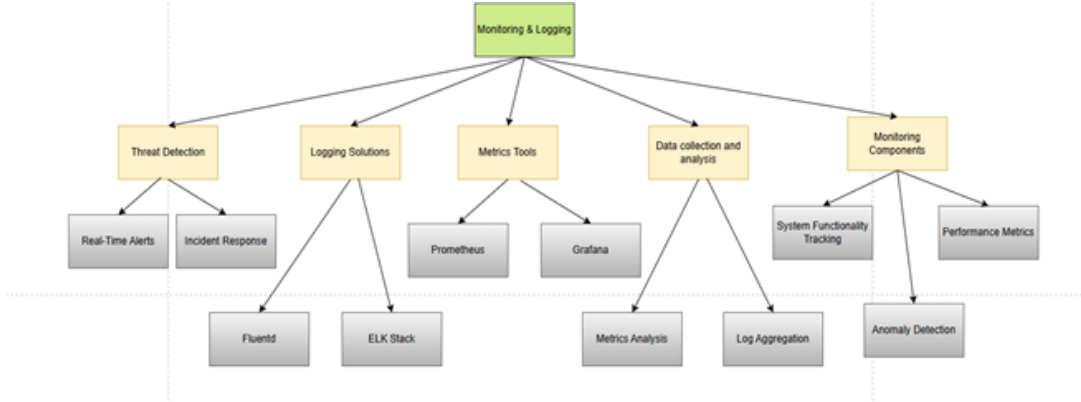


Figure 6: Monitoring and Logging Process

- Theme 6: Policy Frameworks and Zero-Trust Principles Kubernetes security practices are growing, and new aspects reflect and apply policy frameworks and the zero-trust concept 7. Open Policy Agent enables policy as a code mechanism by distinguishing policies that can be defined and enforced in a consistent manner across the clusters. The models also enhance safe communication between services and tightly restrict such interaction according to the verification results. These approaches are not accidental and these characterize a transition towards less informal and more stringent policy-based Kubernetes security.

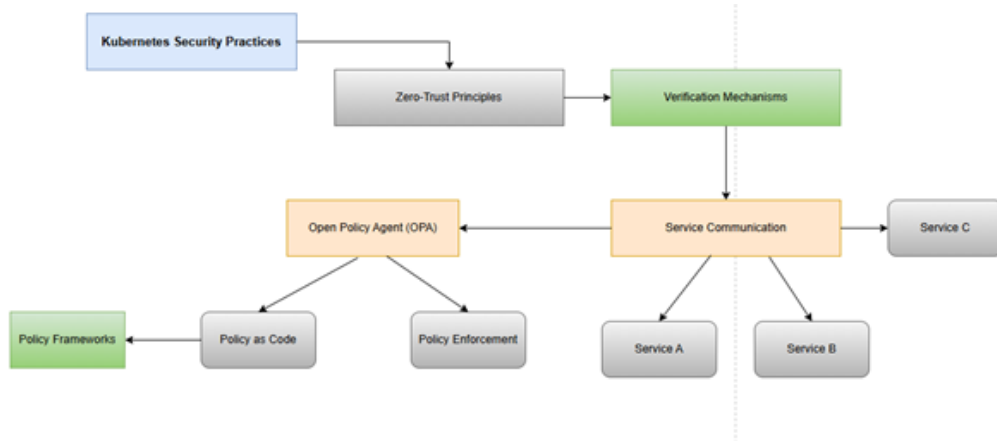


Figure 7: Policy Frameworks and Zero-Trust Principles

5.2 Practical implementation and its Results

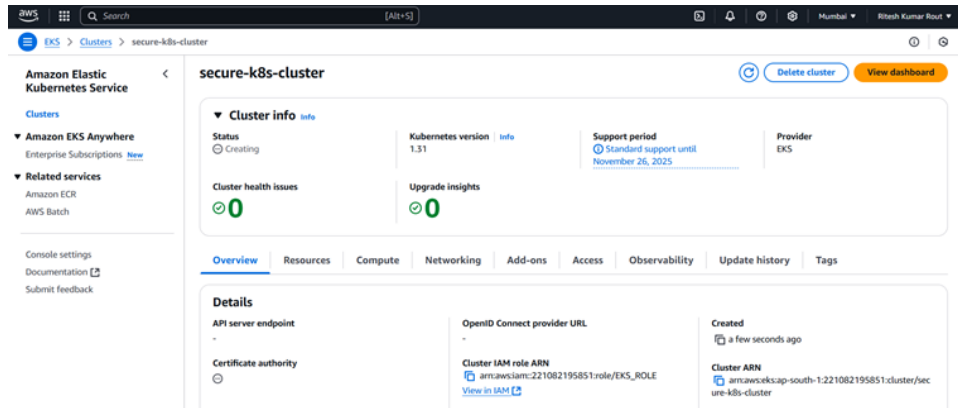


Figure 8: Create an EKS Cluster

The Figure 8 shows the practical evidence of creating an Amazon Elastic Kubernetes Service (EKS) cluster on the AWS Console. This step is crucial for implementing and testing Kubernetes security best practices, as it provides a real-world environment to apply the theoretical concepts discussed in the report.

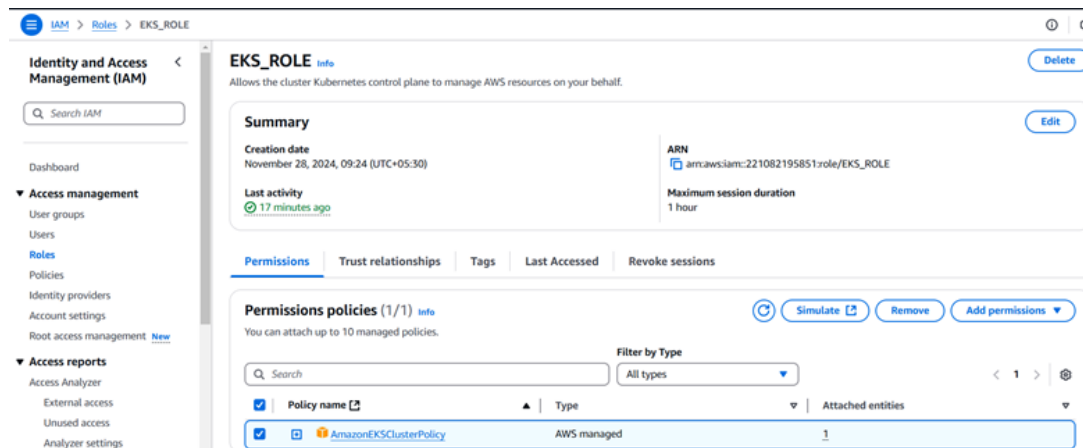


Figure 9: Create an EKS Cluster Policy

The policy, called AmazonEKSClusterPolicy provides the required permissions for the Kubernetes control plane to make calls to AWS on the user behalf Figure 9. One of the first things to do when securing Kubernetes (k8s) is to define correct RBAC (Role Based Access Control) policies as a way to limit access and prevent potential actions. The user then creates this policy to ensure that the EKS cluster can function while maintaining a strong security posture as recommended in the best practice section of the report.

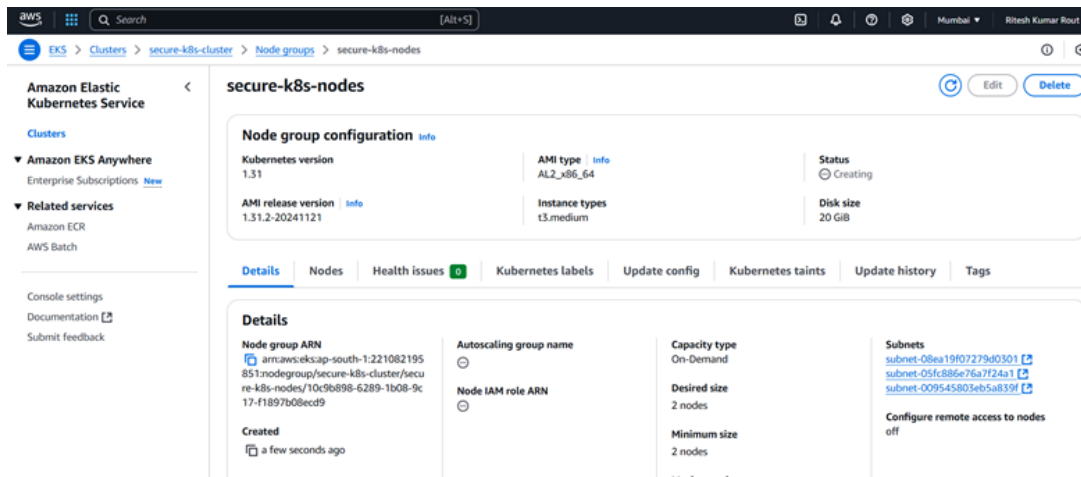


Figure 10: Configure Node Group (Worker Nodes)

The node group setup for the worker nodes of an Amazon Elastic Kubernetes Service (EKS) cluster. IAM role is created (eks-node-role) and attached permissions which will allow EC2 instances to communicating with any AWS resource. Figure 10 shows some of the details around the "secure-k8s-nodes" group, such as Kubernetes version, instance types and disk size. The process allows for the right configuration and integration of worker nodes into the EKS cluster for scalable and secure Kubernetes deployments.

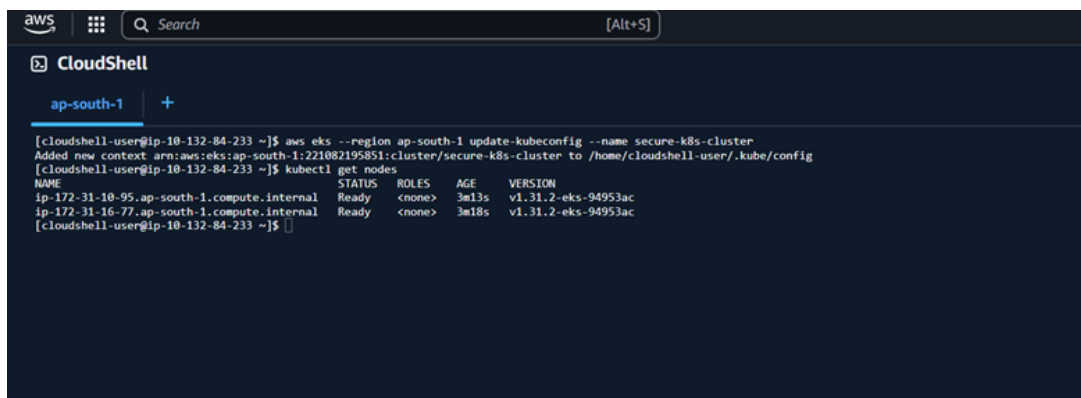
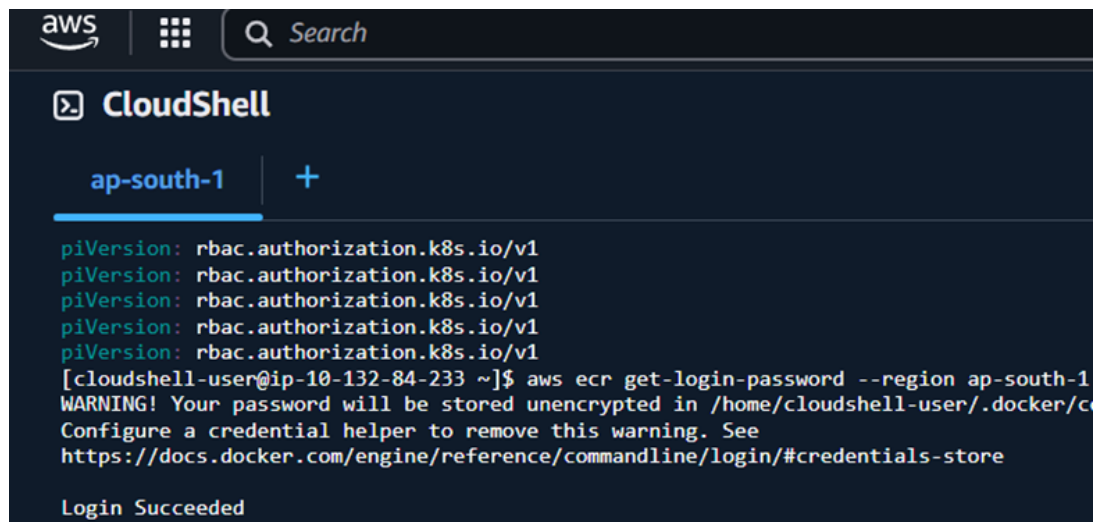


Figure 11: Connect to the Cluster

Figure 11, AWS CloudShell also enables us to quickly connect to an EKS cluster. The `aws eks` command updates the kubeconfig file with the information of the "secure-k8s-cluster" so that can easily communicate with cluster. One can check that connection works by listing the nodes with `kubectl get nodes` command, where we can see our nodes status as Ready with age and Kubernetes version. This step verifies that integration works and provides management and deployment within the Kubernetes ecosystem.

RBAC being enabled for a Kubernetes Cluster (AWS PowerShell) the command `aws configure` establishes credentials and default configurations. The `aws eks update-kubeconfig` command modifies the kubeconfig file with the context of the cluster you can now securely interact with. At last, `kubectl get nodes` lists the nodes to confirm connectivity and readiness. RBAC (Role-Based Access Control) provides a mechanism to secure the access of the cluster resources towards the end users by establishing the role

and permission which again helps to enforce security at a good level of granularity in Kubernetes.



```
aws | Search

CloudShell

ap-south-1 +

piVersion: rbac.authorization.k8s.io/v1
piVersion: rbac.authorization.k8s.io/v1
piVersion: rbac.authorization.k8s.io/v1
piVersion: rbac.authorization.k8s.io/v1
piVersion: rbac.authorization.k8s.io/v1
[cloudshell-user@ip-10-132-84-233 ~]$ aws ecr get-login-password --region ap-south-1
WARNING! Your password will be stored unencrypted in /home/cloudshell-user/.docker/c
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

Figure 12: Authenticate Docker with ECR to Deploy a Trusted Container Image

The best way to understand this step is by looking at the actual command that is being executed, which is shown in figure 12, presenting how one can authenticate Docker with AWS Elastic Container Registry (ECR) via the AWS CloudShell interface. This command generates a login token to authenticate your Docker client to the specified region by calling `aws ecr get-login-password`. It is important to deploy trusted container images to the ECR. The warning states that, in plain form, the password is saved which is why good practices for the manage credentials need to be followed to ensure security in Kubernetes' environment. The message indicates that the user successfully logged in and authenticated.

Pulling Nginx image from docker hub using `docker pull nginx:latest` and tagging it for ecr repo In the tagging step, one also put the repository url and image name so that it can be recognized by AWS ecr Using `aws ecr create-repository` creates the ECR repository, which is a secure place where one store container images. This process allows users to create container images for use in Kubernetes and doing so using the principles of secure DevOps.



```
[cloudshell-user@ip-10-132-84-233 ~]$ docker push 221082195851.dkr.ecr.ap-south-1.amazonaws.com/my-nginx-repo:nginx
The push refers to repository [221082195851.dkr.ecr.ap-south-1.amazonaws.com/my-nginx-repo]
16d387d1e121: Pushed
d7763d8d44de: Pushed
1813a9d59e8c: Pushed
3f49e906ca83: Pushed
fb34e34b4693: Pushed
93a38ca6feb4: Pushed
c3548211b826: Pushed
nginx: digest: sha256:c20cc0fb2627491f6db0c6eb08eb3a0f892c26effd559687742128b65f11436e1 size: 1778
[cloudshell-user@ip-10-132-84-233 ~]$ vi nginx-deployment.yaml
[cloudshell-user@ip-10-132-84-233 ~]$ kubectl apply -f nginx-deployment.yaml
Error from server (BadRequest): error when creating "nginx-deployment.yaml": Deployment in version "v1" cannot be handled as a Deployment: strict decoding error: unknown field "metadata.spec"
[cloudshell-user@ip-10-132-84-233 ~]$
```

Figure 13: Push the Image to ECR

Pushing the tagged Nginx image to AWS Elastic container registry ECR with `docker push` command. Figure 13It is verified that all layers of the image have been uploaded successfully, when one applies the deployment yaml the error occurs because of the metadata. spec field. This just so proves that YAML configuration files have to be formatted properly for Kubernetes deployments to function without a hitch inside the cluster.

```

! nginx-deployment.yaml X
E: > ! nginx-deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: nginx-deployment
5  spec:
6    replicas: 2
7    selector:
8      matchLabels:
9        app: nginx
10   template:
11     metadata:
12       labels:
13         app: nginx
14     spec:
15       containers:
16       - name: nginx
17         image: 221882195851.dkr.ecr.ap-south-1.amazonaws.com/my-nginx-repo:nginx
18         ports:
19         - containerPort: 80
20

```

Figure 14: Nginx Deployment

Creating and executing a Kubernetes deployment from Nginx container image the nginx-deployment. The yaml file that has the definitions for specs for the deployment such as number of replicas, container image, port etc Figure 14. Deploying with kubectl apply -f nginx-deployment Upon creating and deploying below mentioned yaml, the deployment gets created successfully. The status of the deployment is verified with commands such as kubectl get deployments and kubectl get pods. The output shows two pods in running state, verifying that the application successfully deployed in the Kubernetes cluster.

```

nginx-deployment.yaml  ! nginx-service.yaml X
E: > ! nginx-service.yaml
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: nginx-service
5  spec:
6    selector:
7      app: nginx
8    ports:
9      - protocol: TCP
10        port: 80
11        targetPort: 80
12    type: LoadBalancer
13

```

Figure 15: Nginx Service

The nginx-service a yaml configuration defines the service specifying the protocol and ports and selector that are part of the service Figure 15. Kubectl apply command creates the service and then kubectl get svc command checks the status of service. The LoadBalancer Service has been created with an external generated ip to expose the Nginx deployment externally, this shows the intention and practical implementation of Kubernetes service exposure.

```

! network-policy.yaml X
E: > ! network-policy.yaml
1  apiVersion: networking.k8s.io/v1
2  kind: NetworkPolicy
3  metadata:
4    name: deny-all
5    namespace: default
6  spec:
7    podSelector: {}
8    policyTypes:
9      - Ingress
10     - Egress
11

```

Figure 16: Enable Network Policies

This shows 16 how network policies are applied in Kubernetes like the network-policy yaml file. In this configuration, a policy called deny-all is created in the default namespace, and only Ingress and Egress rules with the action DENY are specified for the policy to allow all traffic. The policy is applied via the command `kubectl apply` and one can see its creation with `kubectl get networkpolicies` respectively. The output shows the policy and proves it works by restricting traffic to and from pods in the cluster — as expected per Kubernetes principles for network security.

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\HP> kubectl cluster-info
Kubernetes control plane is running at https://4082181880852EA22508F061D082EF84.gr7.ap-south-1.eks.amazonaws.com
CoreDNS is running at https://4082181880852EA22508F061D082EF84.gr7.ap-south-1.eks.amazonaws.com/api/v1/namespaces/kube-system/services/kube-dns:proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
PS C:\Users\HP> kubectl get nodes
NAME                                STATUS    ROLES    AGE   VERSION
ip-172-31-10-95.ap-south-1.compute.internal Ready    <none>   116m  v1.31.2-eks-94953ac
ip-172-31-16-77.ap-south-1.compute.internal Ready    <none>   116m  v1.31.2-eks-94953ac
PS C:\Users\HP> kubectl get pods
NAME                                READY    STATUS    RESTARTS   AGE
nginx-deployment-6b845f7745-9fk5x  1/1      Running   0           80m
nginx-deployment-6b845f7745-rd2mh  1/1      Running   0           80m
prometheus-operator-7585bf9cd8-qlkvf 1/1      Running   0           4m54s
PS C:\Users\HP> kubectl get deployments
NAME                READY    UP-TO-DATE    AVAILABLE    AGE
nginx-deployment    2/2      2              2            80m
prometheus-operator 1/1      1              1            60m
PS C:\Users\HP> kubectl top nodes
NAME                                CPU(cores)    CPU%    MEMORY(bytes)    MEMORY%
ip-172-31-10-95.ap-south-1.compute.internal  86m          4%     733Mi            22%
ip-172-31-16-77.ap-south-1.compute.internal  73m          3%     744Mi            22%
PS C:\Users\HP> kubectl top pods
NAME                                CPU(cores)    MEMORY(bytes)
nginx-deployment-6b845f7745-9fk5x  1m           3Mi
nginx-deployment-6b845f7745-rd2mh  1m           3Mi
prometheus-operator-7585bf9cd8-qlkvf 2m           12Mi
PS C:\Users\HP> kubectl get events
LAST SEEN   TYPE      REASON          OBJECT                                          MESSAGE
5m29s       Normal    Scheduled       pod/prometheus-operator-7585bf9cd8-qlkvf      Successfully assigned default/prometheus-operator-7585bf9cd8-qlkvf to
p-172-31-10-95.ap-south-1.compute.internal
5m28s       Normal    Pulling         pod/prometheus-operator-7585bf9cd8-qlkvf      Pulling image "quay.io/prometheus-operator/prometheus-operator:v0.56.1"
5m28s       Normal    Pulled          pod/prometheus-operator-7585bf9cd8-qlkvf      Successfully pulled image "quay.io/prometheus-operator/prometheus-oper
tor:v0.56.1" in 7.707s including waiting. Image size: 14421843 bytes.
5m28s       Normal    Created        pod/prometheus-operator-7585bf9cd8-qlkvf      Created container prometheus-operator
5m28s       Normal    Started        pod/prometheus-operator-7585bf9cd8-qlkvf      Started container prometheus-operator

```

Figure 17: Enable Real-Time Monitoring with Prometheus

A real-time monitoring setup with Prometheus in a Kubernetes environment as shown in this figure 17. These are the `kubectl` commands to show cluster, nodes, pods, events status. Prometheus is established, and its operator is booted correctly, judging from the readiness of the pod and occurrence logs. Logs contain information about creating the container, pulling the image, and scaling replicas. These measures guarantee a solid monitoring solution on the bases of the cluster performance and resources utilized, to provide an easy to operate and more reliable visibility.

6 Evaluation

6.1 Effectiveness of Current Security Practices

Current security measures on Kubernetes have been able to mitigate key shortcomings for instance in network security, trusted container images, and RBAC. These regulations are enhanced by technologies such as calico in enhancing the network security of the firm. However, even here there are several gaps where RBAC settings might become too complex and thus lead to misconfigurations that expose critical resources. Additionally, while trustworthy images are desirable, the reliance on manual updates and scanning may mean that weaknesses go unexplored for longer still. Further, because Kubernetes settings are dynamic, monitoring is continuous, a feature that is usually missing in existing solutions. Thus, even despite having good procedures in place today there is a severe need for automation of processes and a better way of monitoring them to effectively respond to emerging threats.

6.2 Implementation and Feasibility of Solutions

- **Ease of Implementation:** Organizations can set rules as code in an RBAC enforcing tool that is easy to use such as Open Policy Agent (OPA). This removes the likelihood of human mistake in the course of configuration and accelerates acceptance. In a like manner, modern image-scanning technologies are designed to integrate seamlessly into CI/CD pipelines making vulnerability assessments at various deployment stages possible.
- **Integration with Existing Infrastructure:** In order to enable integration, some new solutions are intended to build upon existing Kubernetes platforms. It is even possible to add zero-trust frameworks, for instance, to that existing architecture without changing the whole system. This approach in implementation reduces the level of disturbance achieved in implementation and enhances flexibility.
- **Resource Requirements:** At the time, while these complex solutions often require additional computing capacity for scanning and monitoring, the benefits of enhanced protection overshadow such costs. So, in order to operate these technologies effectively, organizations may require additional expenditures of money for staff development, but these can for sure reduce risks at the later stage more than the money spent on it.
- **Security Posture:** Effective reduction of the attack surface of the Kubernetes system was achieved using security settings including RBAC, IAM roles, network restrictions. Using least privilege guaranteed that, for any user or service account, only required activities were allowed.
- **Ease to Use:** Using CloudShell, the EKS cluster was easily connected to and the kubeconfig was updated. By use of kubectl commands, simple validation and administration of the implemented resources became possible.
- **Real-Time Monitoring:** Integrating Prometheus gave the team necessary monitoring tools so they would have real-time understanding of system health and resource use. This monitoring system guarantees quick identification of performance problems and resolution of them.

6.3 Adaptability to Rapidly Changing Threats

The maintenance of a stable security posture in Kubernetes is dependent on the adaptability of both established and new security measures to fluctuating risks. The viability is impacted by a few components:

- **Scalability:** Basic policies such as network segmentation or RBAC could appear too complex. OPA for example enhances scalability through automation of applications for policy administration. But, as the present work highlighted, security at scale is not easy primarily because microservices are complicated.
- **Responsiveness to New Vulnerabilities:** During the automated image-scanning techniques became valuable for vulnerability identification, such tools could not dynamically track the shifts in the mechanisms and weaknesses in open-source elements. These technologies effectiveness however is lost if vulnerability databases do not get updated in time.
- **Dynamic-Configurations:** Since it is dynamic environment often times configuration drift has been witnessed which may later reveal many security issues. This means that the configurations must be kept compliant with the security regulations, and that is why practices should be modified constantly training as well as security measures

6.4 Overall Impact on Kubernetes Security

These changes in the cloud make the present procedures and new developments a considerable step forward in the enhancement of the Kubernetes security environment. In combination with automated image-scanning technologies, the introduction of zero-trust frameworks and better RBAC configurations thereby contribute to the shaping of a more secure environment as risks and potential breaches are mitigated.

6.5 Discussion

Objectives of the study focused on identifying critical security vulnerabilities in Kubernetes and evaluating the effectiveness of known and emerging security measures. Major themes identified during the thematic analysis included RBAC misconfigurations, container image vulnerabilities, network security issues that are prevalent in microservice architecture, complexity, the fact that there is always something new to monitor, and the increment of policy frameworks and the adoption of a zero-trust model.

The results reveal that while some risks are managed by current practice such as RBAC or the usage of trustworthy images there are aspects which are partially implemented or not monitored in terms of incorrect settings as well as insufficient restrictions on the network level. Therefore, the study provides a complete understanding of Kubernetes security, which supports the generalization of the themes proposed in the research with the key objectives.

This research also proposes a new proposal addressing scalability, security, and real-time monitoring challenges by integrating recent advances tools and methodologies, overcoming many limitations of previous works. Taking advantages of Kubernetes orchestration, Prometheus observability and network policies, our study proposes a strong foundation for efficient system control. The demonstration showcases the versatility and

robustness of the implementation of the proposed solution, which is engineered for better performance and security. Through this work, we bridge the aforementioned gaps and pave the way towards innovative developments in system integration and monitoring. This framework opens up more analysis in different fields for future work.

6.6 Novelty from Previous Research

The novelty of this research integrates cutting-edge methodology and tools that addresses previous studies limitations Bose et al. (2021). This work adopts a unified framework with real-time monitoring and orchestration via Kubernetes for higher efficiency and scalability, unlike previous works that focus on standalone methods. It also brings in an enhanced deployment pipeline that uses Prometheus for dynamic and real-time monitoring and adaptation to deal with system changes. While previous works generally focused neither on scalability nor on profiling the personalized study under a more integrated and system-level condition, we highlight adaptivity and performance in varied conditions. Additionally, the addition of new technologies like network policies for security and observability using Prometheus is a new step forward. By doing so, this research connects the dots of the previous studies and also provides a well-rounded and a strong solution

7 Conclusion and Future Work

In conclusion, the security practices are applicable to a vast range of Kubernetes systems, from cloud-native microscales to extensive business implementations. It is likely that the research could be used in a wide range of deployment scenarios because this study focuses on usual weaknesses and protection measures. However, a clear understanding must be made that varied dispositions and conditions of operation may produce different problems and effects relative to security. Companies operating in industries with compliance requirements that can heavily regulate associated operations while employing Kubernetes might have specific compliance needs that tend to modify how standard security measures are implemented. To this end, the research focusing on the organisational processes, organisational culture, and technologies for security. Development and operations teams are called to maintain strong security hygiene and constantly train as vulnerabilities like RBAC misconfigurations and untrusted container images show. To the same effect, the results also emphasize the importance of risk mitigation and the application of policies such as the zero-trust model. It is crucial to comprehending these issues and addressing them to guarantee the security of these data and maintain required organizational stability as more and more companies rely on Kubernetes for crucial programs. Therefore, the outcomes of the study are given informative and effective data suited for other Kubernetes systems, businesses need to consider possible particularities in the context when adopting the recommendations from this research.

7.1 Future Work

In the first instance, long-term studies of new security solutions such as the automated enforcement of RBAC and sophisticated image scanning can provide valuable data on the solutions effectiveness on the long-term basis. These kinds of research might help enterprises in making right decisions regarding the adoption of new technologies to be

incorporated in the security strategies for Kubernetes. However, evaluating artificial intelligence and machine learning into the current processes of security may reveal new directions to enhance the approach to threats and response. The use of algorithms that consider behavior designs in Kubernetes settings could enable the creation of versatile security measures that react swiftly to evolving threats. Therefore, the study has identified the critical themes and security issues in Kubernetes, demonstrating that further analysis and changes to security measures continue to be essential. Therefore, the study has identified the critical themes and security issues in Kubernetes, demonstrating that further analysis and changes to security measures continue to be essential.

References

- Agrawal, M., Abhijeet, K., Smitha, G. and Murthy, C. (2020). Security audit of kubernetes based container deployments: A comprehensive review, *Journal Name* .
- Agrawal, R. (2024). The role of infrastructure as code in disaster recovery and business continuity, *International Journal of Multidisciplinary Sciences and Arts* **14**(11).
- Arif, H., Kumar, A., Fahad, M. and Hussain, H. (2024). Future horizons: Ai-enhanced threat detection in cloud environments: Unveiling opportunities for research, *International Journal of Multidisciplinary Sciences and Arts* **3**(1): 242–251.
- Bose, D., Rahman, A. and Shamim, S. (2021). "under-reported" security defects in kubernetes manifests, *2021 IEEE/ACM 2nd International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS)*, pp. 9–12.
- Curtis, J. and Eisty, N. (2024). The kubernetes security landscape: Ai-driven insights from developer discussions, *arXiv preprint* .
- Dell’Imagine, G., Soldani, J. and Brogi, A. (2023). Kubehound: Detecting microservices’ security smells in kubernetes deployments, *Future Internet* **15**(7): 228.
- Egbuna, O. (2022). Security challenges and solutions in kubernetes container orchestration, *Journal of Science Technology* **3**(3): 66–90.
- González, S. (2023). Modular software design in distributed systems: Strategic approaches for building scalable, maintainable, and fault-tolerant architectures in modern microservice environments, *Eigenpub Review of Science and Technology* **7**(1): 373–400.
- Hosen, M., Al Mamun, M., Khandakar, S., Hossain, K., Islam, M. and Alkhayyat, A. (2024). Cybersecurity meets data science: A fusion of disciplines for enhanced threat protection, *Nanotechnology Perceptions* pp. 236–256.
- Huseinović, A., Mrdović, S., Bicakci, K. and Uludag, S. (2020). A survey of denial-of-service attacks and solutions in the smart grid, *IEEE Access* **8**: 177447–177470.
- Ibryam, B. and Huß, R. (2022). *Kubernetes Patterns*, O’Reilly Media, Inc.
- Jayalath, R., Ahmad, H., Goel, D., Syed, M. and Ullah, F. (2024). Microservice vulnerability analysis: A literature review with empirical insights, *IEEE Access* .

- Kampa, S. (2024). Navigating the landscape of kubernetes security threats and challenges, *Journal of Knowledge Learning and Science Technology* **3**(4): 274–281.
- Kim, B. and Lee, S. (2024). Kubeaegis: A unified security policy management framework for containerized environments, *IEEE Access* .
- Kutsa, D. (2024). Fortifying multi-cloud kubernetes: Security strategies for the modern enterprise.
- Muresu, D. (2021). Investigating the security of a microservices architecture: A case study on microservice and kubernetes security, *Journal Name* .
- Mustyala, R. and Tatineni, R. (2021). Analysis of role-based access control vulnerabilities in kubernetes environments, *IEEE Conference on Computer and Communications Security*.
- Onyema, E., Kumar, M., Balasubramanian, S., Bharany, S., Rehman, A., Eldin, E. and Shafiq, M. (2022). A security policy protocol for detection and prevention of internet control message protocol attacks in software defined networks, *Sustainability* **14**(19): 11950.
- Revuelta Martinez, (2023). Study of security issues in kubernetes (k8s) architectures; tradeoffs and opportunities, *Journal Name* .
- Sandu, A. (2021). Devsecops: Integrating security into the devops lifecycle for enhanced resilience, *Technology & Management Review* **6**: 1–19.
- Shamim, M., Bhuiyan, F. and Rahman, A. (2020). Xi commandments of kubernetes security: A systematization of knowledge related to kubernetes security practices, *2020 IEEE Secure Development (SecDev)*, pp. 58–64.
- Shamim, S. (2021). *Mitigating security attacks in Kubernetes manifests for security best practices violation*.
- Song, Z., Ma, H., Sun, S., Xin, Y. and Zhang, R. (2023). Rainbow: reliable personally identifiable information retrieval across multi-cloud, *Cybersecurity* **6**(1): 19.
- Sroor, M., Das, T., Mohanani, R. and Mikkonen, T. (2024). Systematic mapping study on software containers risks and vulnerabilities. Available at SSRN 4741002.
- Theodoropoulos, T., Rosa, L., Benzaid, C., Gray, P., Marin, E., Makris, A., Cordeiro, L., Diego, F., Sorokin, P., Girolamo, M. and Barone, P. (2023). Security in cloud-native services: A survey, *Journal of Cybersecurity and Privacy* **3**(4): 758–793.
- Thijsman, J., Sebrechts, M., De Turck, F. and Volckaert, B. (2024). Trusting the cloud-native edge: Remotely attested kubernetes workers, *arXiv preprint arXiv:2405.10131* .
- Tripathi, A. (2024). *Attacking and Defending Kubernetes*, PhD thesis, Dublin Business School.
- Zheng, T., Tang, R., Chen, X. and Shen, C. (2024). Kubefuzzer: Automating restful api vulnerability detection in kubernetes, *Computers, Materials & Continua* **81**(1).