Project Report

On

Automated Security and Compliance Monitoring for Kubernetes Microservices

Submitted in fulfillment for the award of **Post Graduate Diploma in IT Infrastructure System & Security (PG-DITISS)** from CDAC ACTS (Pune)

Guided By:

Mr. Nikhil Kumar

Presented By:

Atharva Sharad Walurkar	PRN: 240840123010
Eshwar Kishor Satale	PRN: 240840123016
Ketan Govind Rehpade	PRN: 240840123024
Shubham Prakash Potawade	PRN: 240840123051
Om Santosh Vikhe	PRN: 240840123057

CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that

Atharva Walurkar	PRN: 240840123010
Eshwar Satale	PRN: 240840123016
Ketan Rehpade	PRN: 240840123024
Shubham Potawade	PRN: 240840123051
Om Vikhe	PRN: 240840123057

Have successfully completed their project on

"Automated Security and Compliance Monitoring for Kubernetes Microservices"

Under the guidance of

Mr. Nikhil Kumar

Project Guide

Project Supervisor

ACKNOWLEDGEMENT

This project "Automated Security and Compliance Monitoring for kubernetes Deployments" was a great learning experience for us and we are submitting this work to Advanced Computing Training School (CDAC ACTS). We all are very glad to mention the name of Mr. Nikhil Kumar for his valuable guidance to work on this project. His guidance and support helped us to overcome various obstacles and intricacies during the course of project work

We are highly grateful to Mrs. Namrata Ailawar (Joint Director (ACTS training Centre), C-DAC, for her guidance and support whenever necessary while doing this course Post Graduate Diploma in IT Infrastructure, System & Security (PG-DITISS) through C-DAC ACTS, Pune.

Our most heartfelt thanks go to Miss. Sajida Shaikh (Course Coordinator, PG-DITISS) who gave all the required support and kind Co-ordination to provide all the necessities like required hardware, internet facility and extra Lab-hours to complete the project and throughout the course up to the last day here in C-DAC ACTS, Pune.

Sincerely,

Atharva Walurkar

Eshwar Satale

Ketan Rehpade

Shubham Potawade

Om Vikhe

ABSTRACT

In an era where containerized applications are becoming the backbone of modern IT infrastructure, ensuring the security and reliability of Kubernetes clusters is paramount. This project focuses on establishing a secure and efficient Kubernetes cluster, leveraging kind for local deployment and expanding it with a worker node for distributed workloads. ArgoCD ensuring that the latest application version is pulled from Git and deployed in a secure and compliant manner

To enhance observability and security, Prometheus and Grafana are integrated for real-time monitoring, while Kubescape and Kube-bench are utilized for vulnerability scanning and security benchmarking. The system is further hardened by implementing automated scripts that identify and remediate vulnerabilities, ensuring compliance with industry security standards. Additionally, a microservice is deployed on the worker node, while the master node handles monitoring and security analysis.

The primary objectives of this project are to improve cluster security, enhance operational visibility, and automate vulnerability management to minimize human intervention. By implementing this approach, organizations can achieve a resilient, self-healing Kubernetes environment, capable of detecting and mitigating security risks in real time. This project serves as a comprehensive guide for securing Kubernetes clusters.

TABLE OF CONTENTS

CERTIFICATE	II
ACKNOWLEDGEMENT	III
ABSTRACT	IV
INDEX	V

Sr. No.	TITLE	PAGE
1	INTRODUCTION	1
2	PREREQUISITES	2
3	DATA FLOW DIAGRAM	4
4	LAB SETUP STEPS	5
5	WORKING	7
6	IMPLEMENTATION & OUTPUT	14
7	CONCLUSION	31
8	REFERENCES	32

INTRODUCTION

In modern cloud-native environments, ensuring security, compliance, and monitoring for microservices-based applications is crucial. This project focuses on automating the deployment, security scanning, and monitoring of a Kubernetes-based microservices architecture using a combination of open-source tools.

A Kubernetes cluster was deployed using kind (Kubernetes-in-Docker) to host microservices, providing a local, containerized environment for development and testing. ArgoCD was integrated for GitOps-based deployment, ensuring that application updates from a Git repository are automatically synchronized and deployed in the cluster. Helm was leveraged for package management, simplifying the installation and management of various tools.

For real-time monitoring and observability, Prometheus and Grafana were deployed. Prometheus collects and aggregates metrics from cluster components and microservices, while Grafana provides visualization through interactive dashboards.

To maintain security and compliance, multiple tools were implemented:

- Kyverno enforces Kubernetes security policies.
- Kube-Bench performs CIS benchmarking to assess security configurations.
- Kubescape scans the cluster for misconfigurations and compliance violations.

A custom script was developed and scheduled using a cron job to automate security scans at regular intervals. The script runs Kubescape, generates a security report in PDF format, and sends it via email to ensure continuous compliance monitoring without manual intervention.

This project provides a robust and automated approach to managing Kubernetes microservices, integrating deployment automation, security enforcement, and monitoring into a streamlined workflow.

PREREQUISITES

Before deploying the microservices-based application in the Kubernetes environment, the following prerequisites must be met:

1. System Requirements

- Operating System: Ubuntu 20.04 or later (Recommended)
- CPU: Minimum 4 cores
- RAM: At least 4GB (Recommended: 8GB for smooth operation)
- Storage: At least 20GB of free disk space.

2. Software Dependencies

The following tools and frameworks must be installed before setting up the Kubernetes cluster:

- Docker: Required for running Kubernetes in Docker (kind)
- kind (Kubernetes-in-Docker): To create a lightweight Kubernetes cluster locally
- kubectl: Command-line tool for interacting with the Kubernetes cluster
- Helm: Package manager for deploying Prometheus, Grafana, and security tools
- ArgoCD: For automated GitOps-based deployment from a Git repository
- cron: To schedule periodic security scans using Kubescape.

3. Network & Access Requirements:

The following tools are needed for observability, policy enforcement, and security scanning:

- Prometheus: Used for real-time metrics collection and monitoring.
- Grafana: Provides visualization dashboards for Prometheus metrics.
- Kyverno: Kubernetes-native policy engine for security and compliance.
- Kube-Bench: Assesses the Kubernetes cluster's compliance with CIS security benchmarks.
- Kubescape: Performs security posture assessments for Kubernetes configurations.

4. Networking and Access Requirements

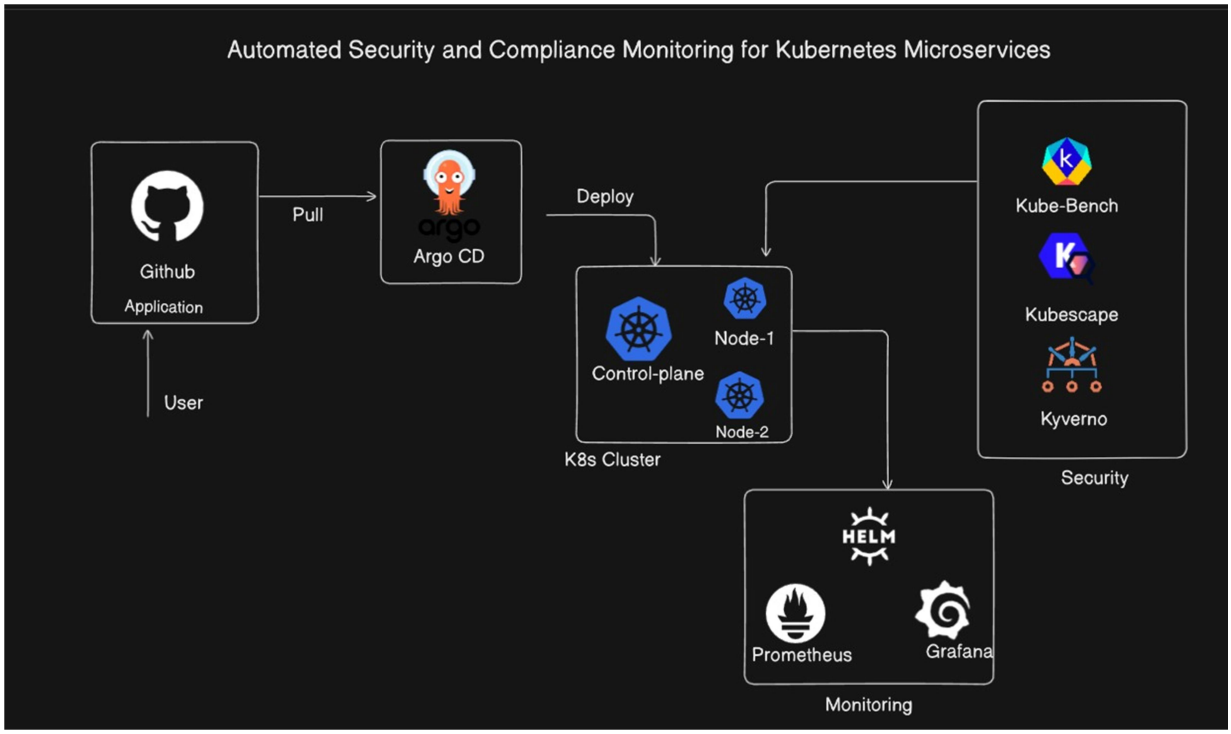
Network Access: Ensure that the system has internet access to pull container images and dependencies.

Port Availability:
6443 (Kubernetes API server)
9090 (Prometheus)
3000 (Grafana)

5. Basic Knowledge Requirements

- Containerization (Docker, Kubernetes)
- Linux Command Line & System Administration
- Basic Networking Concepts (TCP/IP, DNS, Firewalls)
- YAML Configuration Files (for Kubernetes manifests and Helm charts)

DATAFLOW DIAGRAM



LAB SETUP AND STEPS

Lab Environment

The project is deployed on a local Kubernetes cluster using kind (Kubernetes-in-Docker). The environment consists of multiple tools for deployment, monitoring, and security.

Cluster Setup: Kind-based Kubernetes cluster with a control plane and two worker nodes.

Deployment Management: ArgoCD automates the deployment of microservices by continuously syncing with a Git repository.

Package Management: Helm is used to install and manage tools like Prometheus, Grafana, Kubescape, and Kyverno.

Security and Compliance: Tools like Kube-Bench, Kubescape, and Kyverno ensure security policies and CIS compliance checks.

Monitoring & Visualization: Prometheus and Grafana provide real-time monitoring and visualization of system metrics.

Step-by-Step Implementation

1. Setting Up the Kubernetes Cluster
Install kind to create a local Kubernetes cluster.
Define a multi-node cluster configuration file and create the cluster using kind.

2. Configuring ArgoCD for GitOps Deployment
Install ArgoCD on the cluster and expose its UI for managing deployments.
Connect ArgoCD to the GitHub repository to automatically sync application updates.

3. Deploying the Application
Define Kubernetes manifests for microservices in GitHub.
Configure ArgoCD to pull the latest application version and deploy it automatically to the cluster.

4. Installing Monitoring Tools
Helm is used to deploy Prometheus and Grafana on the cluster.

Prometheus scrapes metrics from the microservices, and Grafana visualizes them with custom dashboards.

5. Implementing Security & Compliance Monitoring
Kyverno enforces security policies on workloads deployed in the cluster.
Kubescape scans for misconfigurations and generates reports.
Kube-Bench runs security benchmark checks to ensure compliance with Kubernetes security best practices.

6. Automating Security Scans.
Triggers a Kubescape scan after each deployment.
The scan results are stored and sent via email as a PDF report.

WORKING

The project is built around a **Kubernetes cluster** deployed using **kind (Kubernetes-in-Docker)** to host a **microservices-based application**. The deployment process is automated using **ArgoCD**, while **Helm** is utilized for managing installations of monitoring and security tools like **Prometheus, Grafana, Kube-Bench, Kubescape, and Kyverno**. The entire setup ensures **continuous security, compliance, and monitoring** of the deployed microservices.

1. Kubernetes

Kubernetes (K8s) is the foundation of this project, providing container orchestration capabilities to efficiently deploy, scale, and manage microservices. It abstracts away infrastructure complexities, allowing applications to be deployed in an automated, resilient, and scalable manner.

In this project:

- A multi-node Kubernetes cluster is created using kind, simulating a production-like environment.
- The cluster consists of a control-plane node (which manages scheduling, networking, and orchestration) and worker nodes (where application containers run).
- Applications and security tools are deployed as Kubernetes pods, managed by Deployments and Services for load balancing and availability.

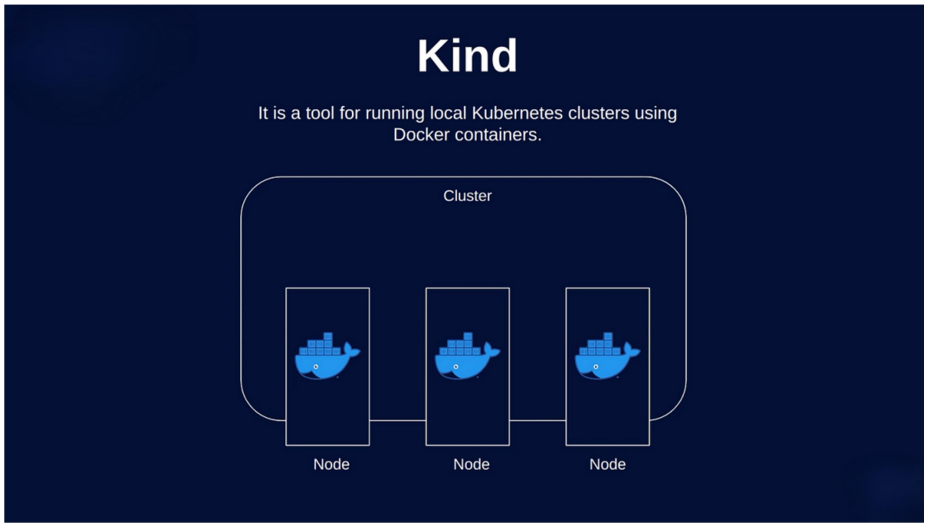
2. Docker

Docker plays a key role in containerizing the application and the security/monitoring tools deployed within Kubernetes. Instead of running applications as traditional processes, Docker packages them into lightweight, portable containers that can be easily deployed across different environments.

In this setup:

- The microservices and security tools are containerized using Docker images.
- These containers are pulled from a registry (e.g., Docker Hub or a private GitHub Container Registry) and deployed on Kubernetes.
- This allows rapid scaling, consistent environments, and dependency isolation for each application component.

3. kind (Kubernetes-in-Docker)



kind is used to create a Kubernetes cluster inside Docker.

- kind runs Kubernetes inside Docker containers, eliminating the need for a full-fledged cloud environment.
- The cluster consists of a control-plane node that manages Kubernetes operations and worker nodes that run the application workloads.
- It provides a simple yet effective way to test Kubernetes deployments in a self-contained, local environment before moving to production.

4. Helm (Package Manager for Kubernetes)

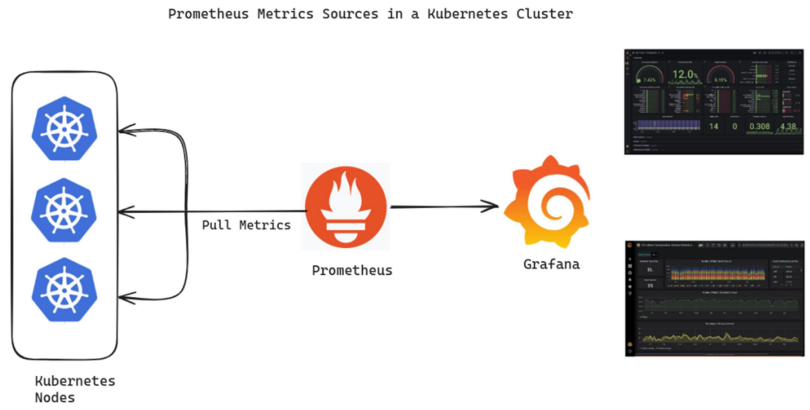
Helm is used in this project to simplify the installation and management of complex Kubernetes applications. Instead of manually configuring each component, Helm allows applications to be installed, upgraded, and version-controlled using Helm charts.

Here, Helm is used to deploy:

- Prometheus & Grafana for monitoring
- Kubescape, Kyverno, and Kube-Bench for security and compliance

By using Helm, these tools can be installed with minimal effort, ensuring consistency and easy upgrades.

5. Prometheus & Grafana (Monitoring and Visualization)



Monitoring is critical to ensure application health, detect performance bottlenecks, and prevent system failures. This project integrates Prometheus and Grafana for real-time monitoring and visualization.

- Prometheus:
  - Collects metrics from Kubernetes nodes, applications, and security tools.
  - Uses a time-series database to store and analyze performance data.
  - Helps track CPU usage, memory consumption, network traffic, and more.
- Grafana:
  - Provides interactive dashboards to visualize metrics collected by Prometheus.
  - Displays insights on system health, application performance, and security posture.
  - Alerts can be configured to notify users of potential system issues.

These monitoring tools ensure that system administrators can track application behavior and take necessary actions to optimize performance and troubleshoot issues proactively.

6. ArgoCD (GitOps-Based Deployment Automation)

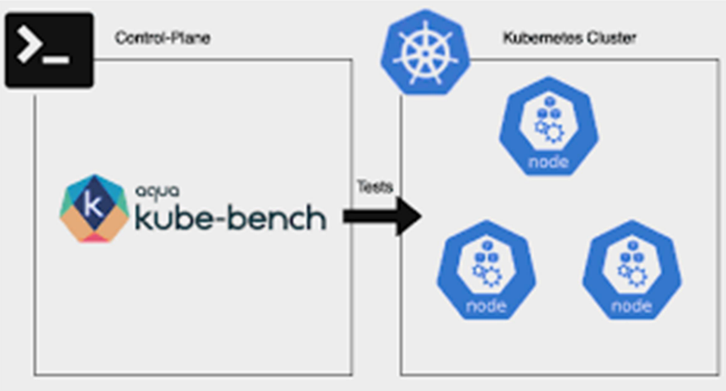
ArgoCD is a continuous deployment tool that ensures Kubernetes applications are always in sync with their defined configurations in a Git repository. Instead of manually applying changes, ArgoCD automates deployment updates whenever new changes are committed to Git.

In this setup:

- The application manifests (YAML files defining Kubernetes deployments, services, and configurations) are stored in a Git repository.
- A script runs at scheduled intervals using a cron job, pulling the latest updates from Git.
- The updated manifests are automatically deployed to the Kubernetes cluster.

This GitOps-based workflow ensures that all deployments remain consistent, reproducible, and version-controlled.

7. Kube-Bench (CIS Security Benchmarking for Kubernetes)



Kube-Bench is a security auditing tool that verifies whether the Kubernetes cluster follows best practices defined by the CIS (Center for Internet Security) benchmarks. It performs:

- Security checks on Kubernetes configurations, authentication settings, and cluster policies.
- Audits on control-plane and worker node security settings.
- Reports on potential vulnerabilities and provides remediation recommendations.

By running Kube-Bench, we ensure that the Kubernetes environment is secure and compliant with industry standards.

8. Kubescape (Kubernetes Misconfiguration Scanner)

Kubescape is an automated security scanning tool that analyzes Kubernetes configurations for:

- Misconfigurations that could lead to security vulnerabilities.
- Exposures to known attack patterns and compliance risks.
- Access control weaknesses, such as overly permissive roles.

A cron job is configured to run Kubescape scans at scheduled intervals, automatically generating PDF reports with detailed security insights.

9. Kyverno (Kubernetes Policy Enforcement Engine)

Kyverno is a policy management tool used to enforce security rules and compliance requirements within Kubernetes. Instead of modifying application configurations manually, Kyverno ensures that:

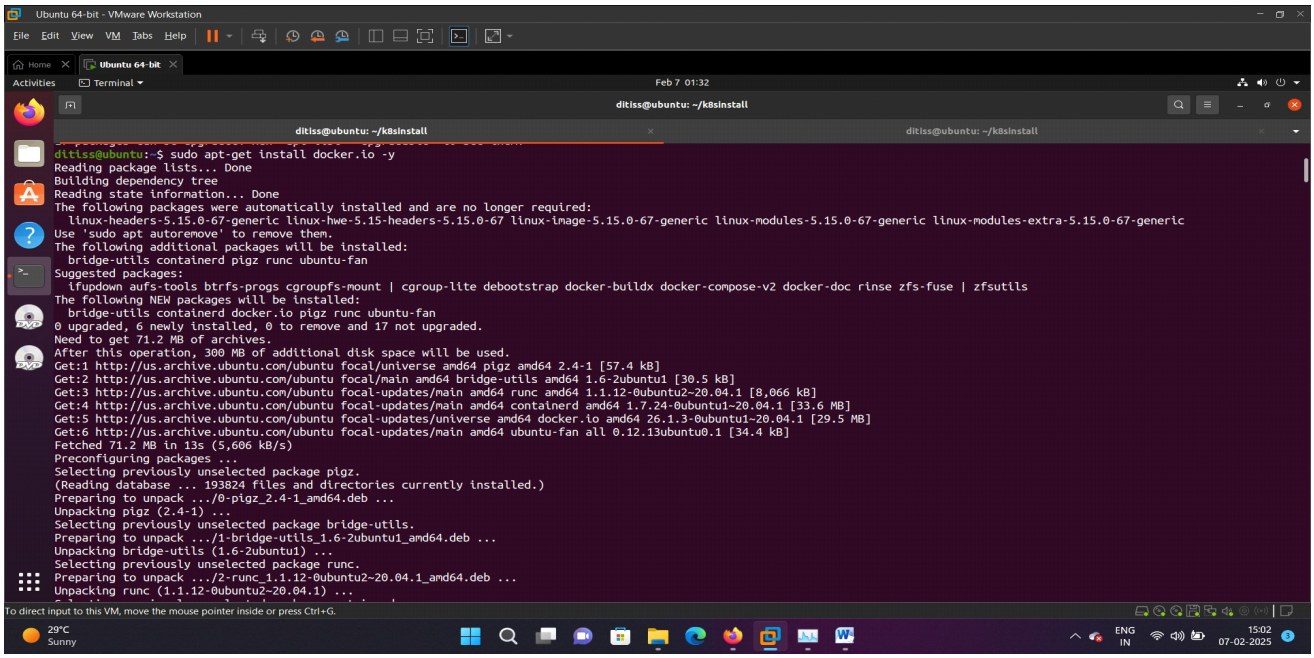
- Only secure and validated deployments are allowed.
- Specific security policies (e.g., restricting privileged access, enforcing network policies) are enforced.

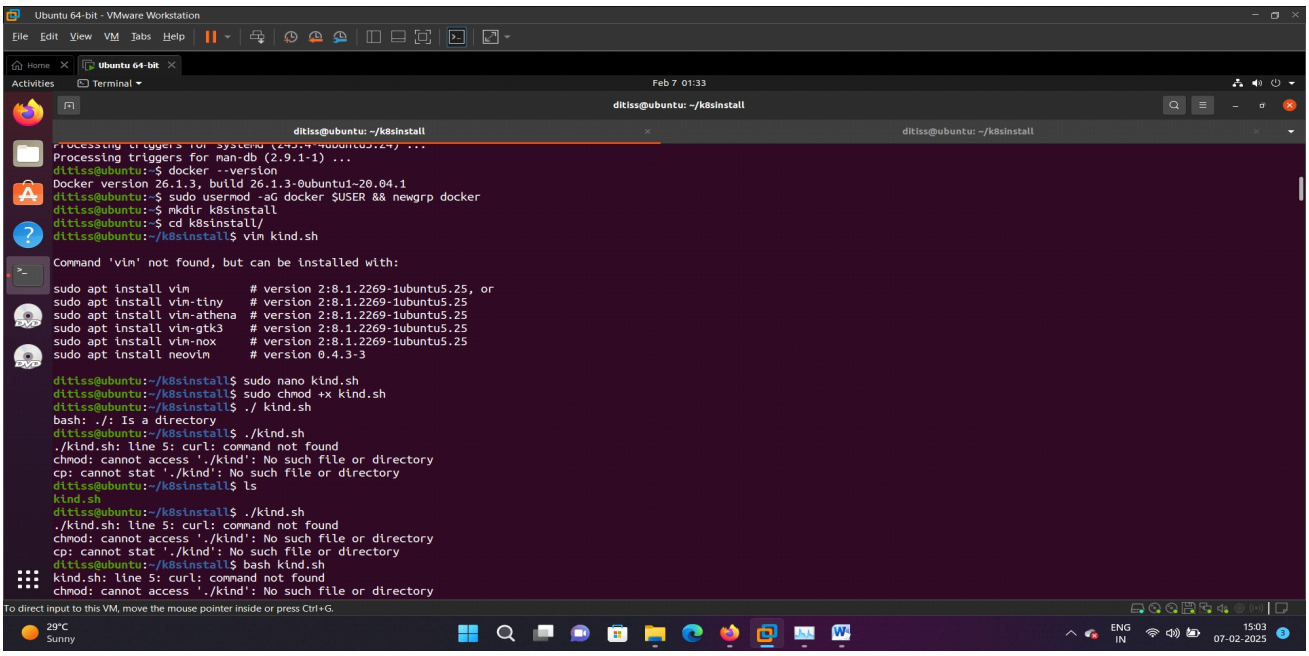
- Any non-compliant workloads are automatically blocked.

By integrating Kyverno, the cluster remains secure and compliant with defined security policies.

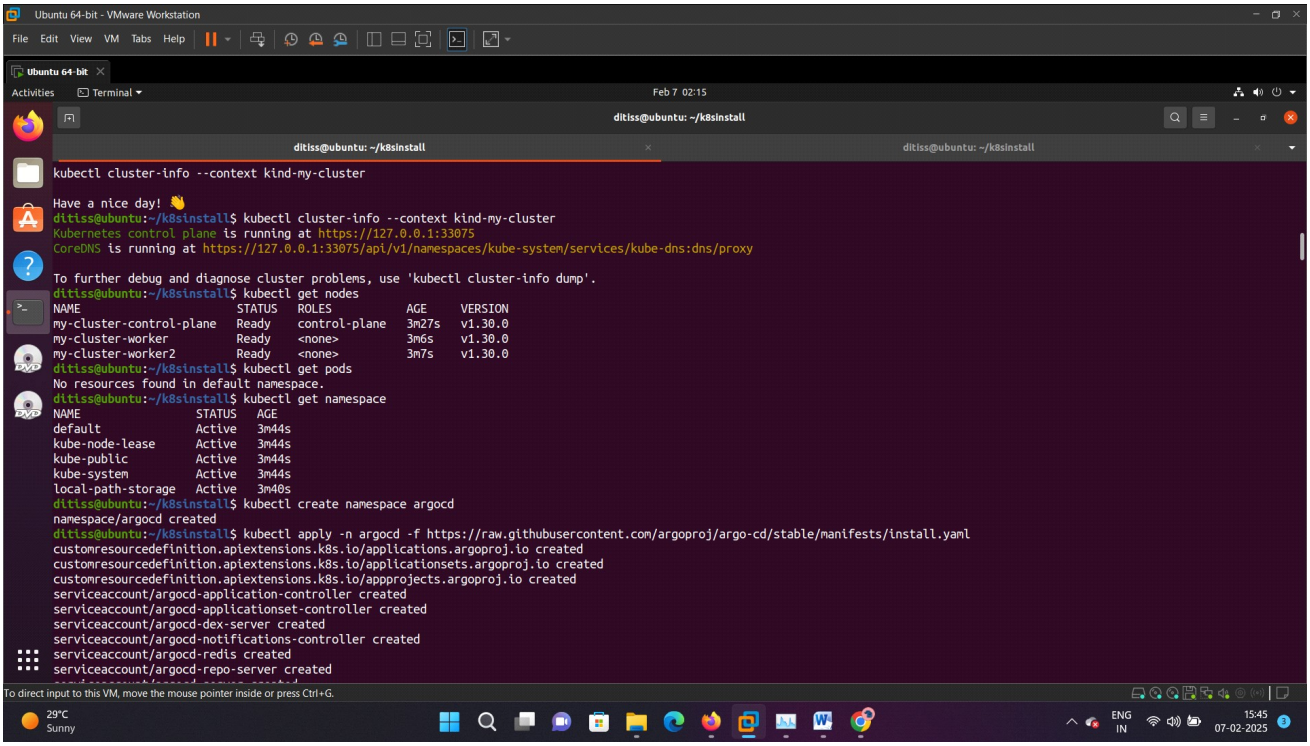
IMPLEMENTATION & OUTPUTS

Docker installation and kind installation

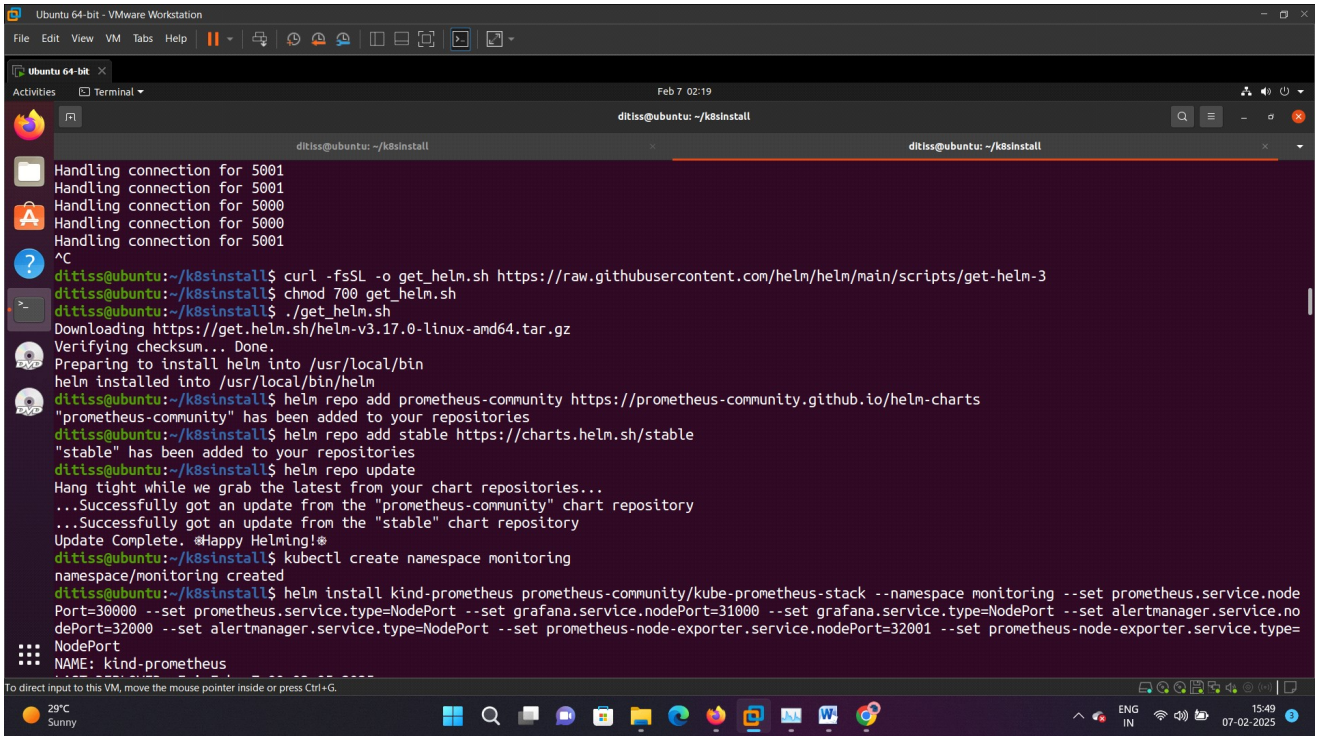


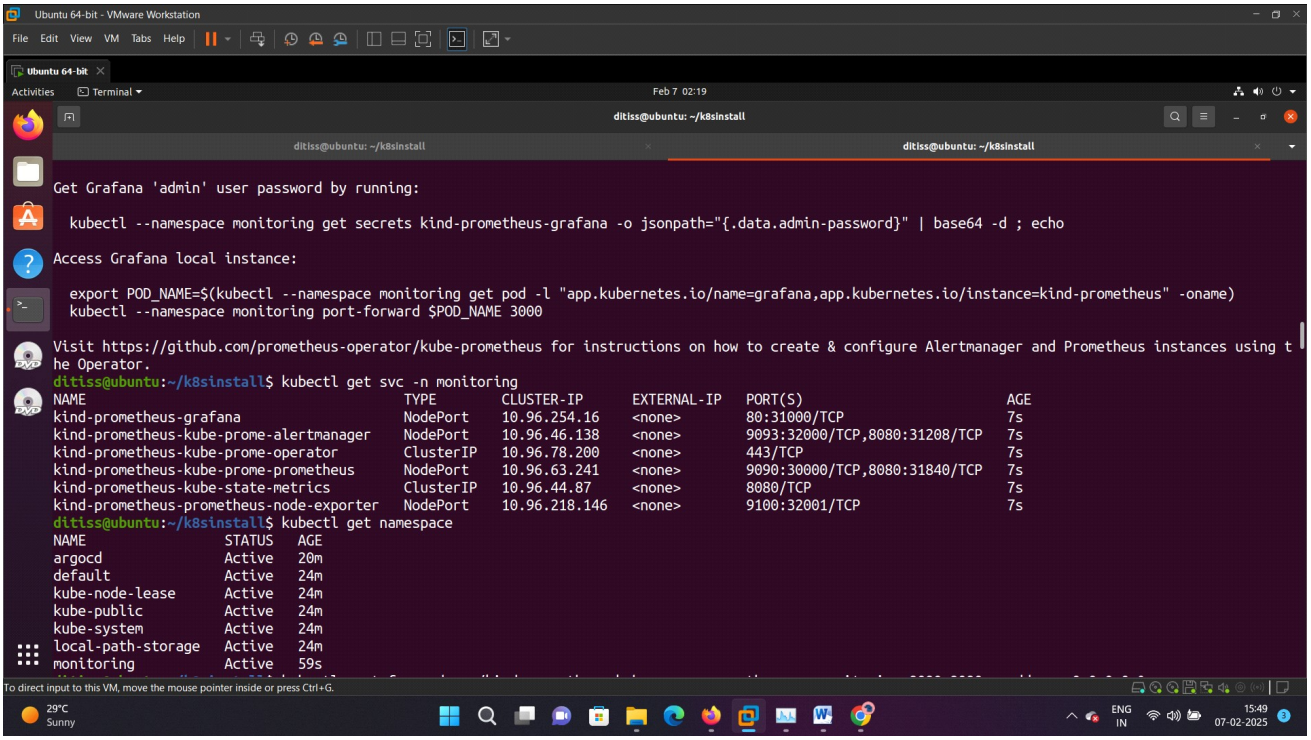


Creating cluster via kind of three nodes which include 1 control plane and 2 worker nodes.
Installation of ArgoCD in namespace ArgoCD

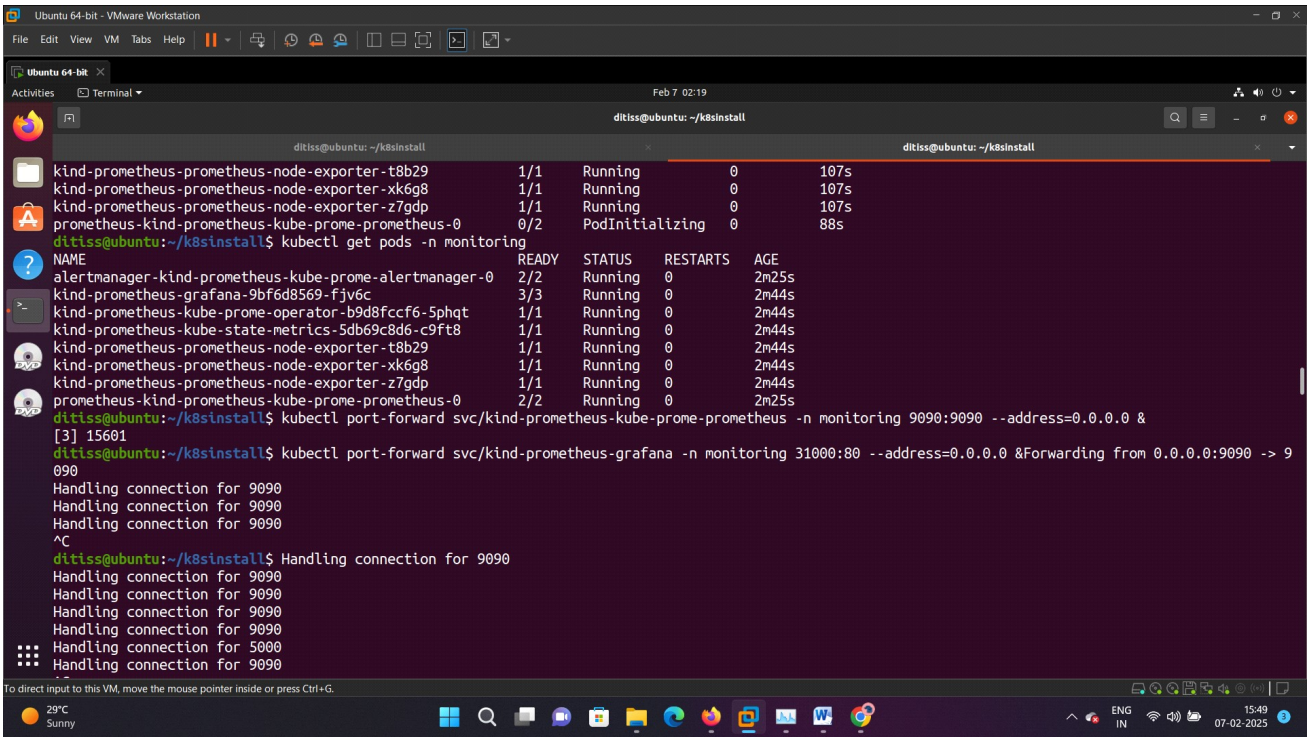


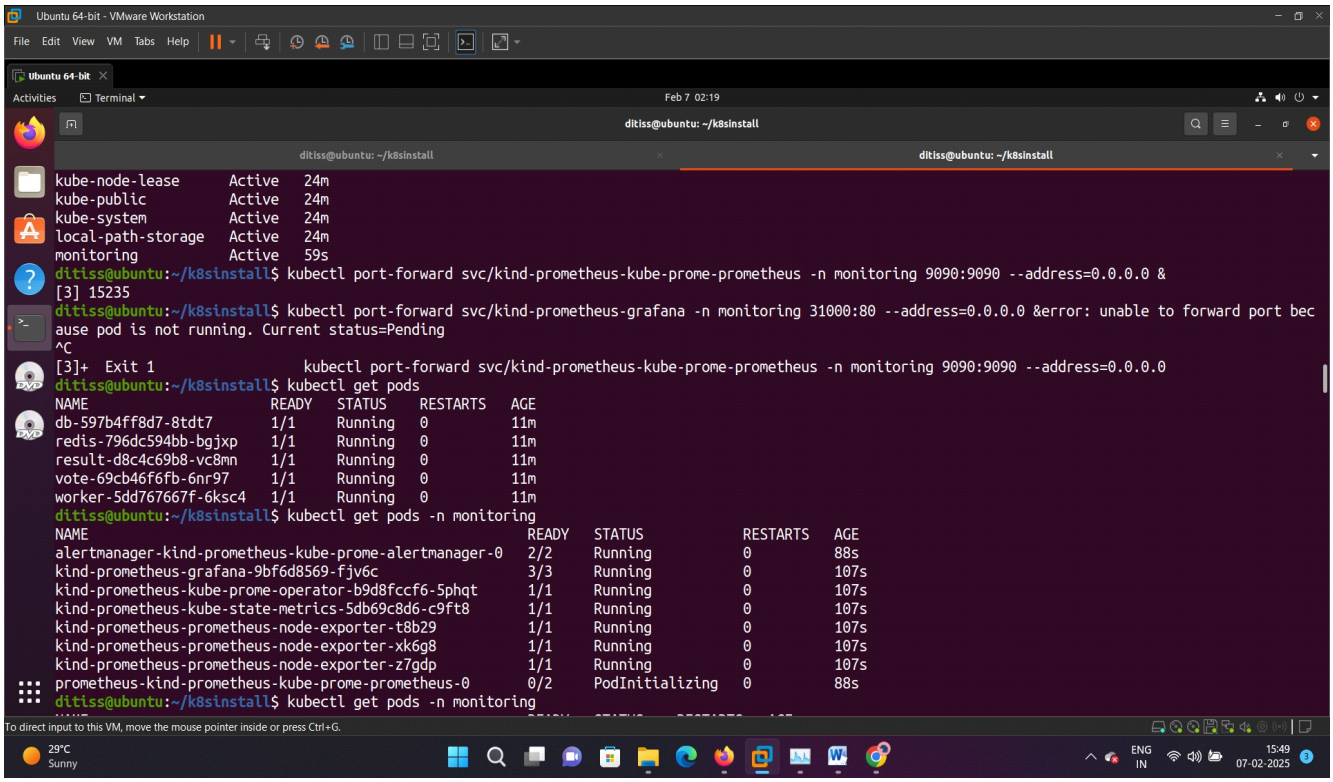
HELM installation and
Prometheus and Grafana installation using helm in namespace monitoring



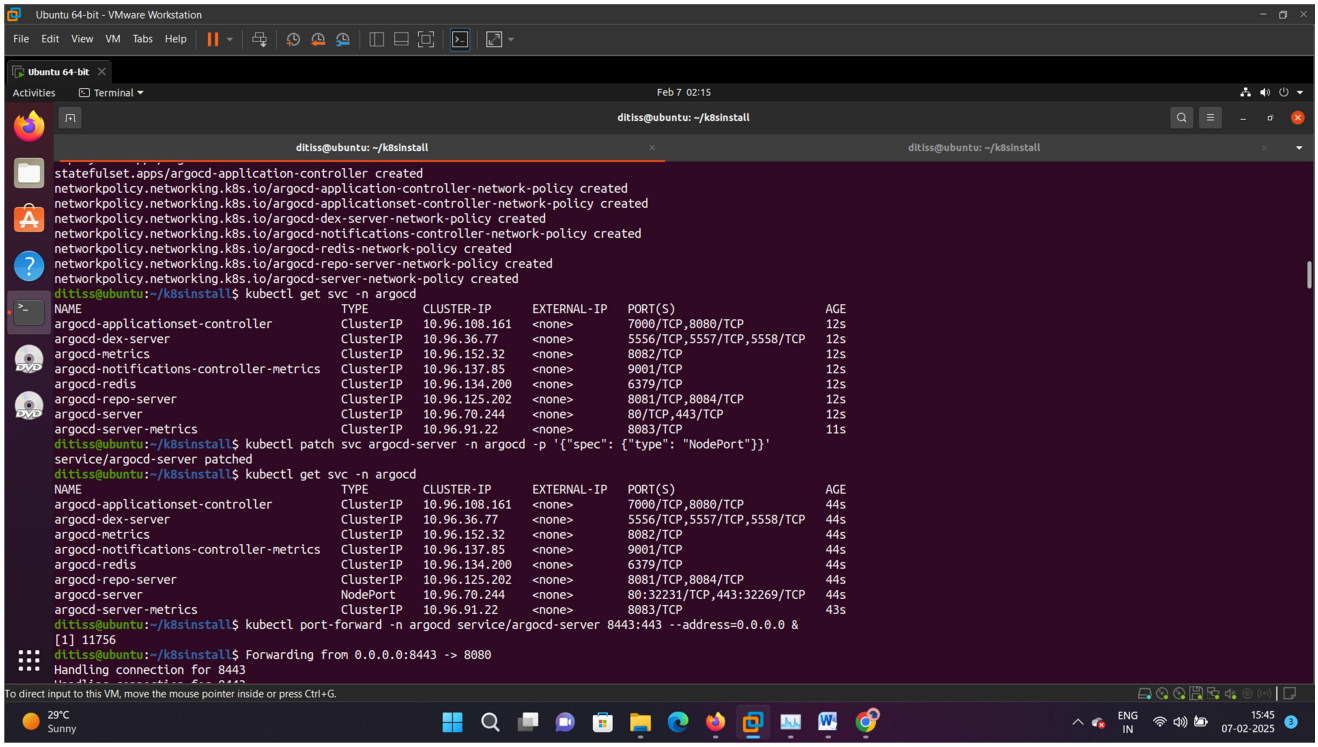


Enable port forwarding for Prometheus=9090 and Grafana=31000

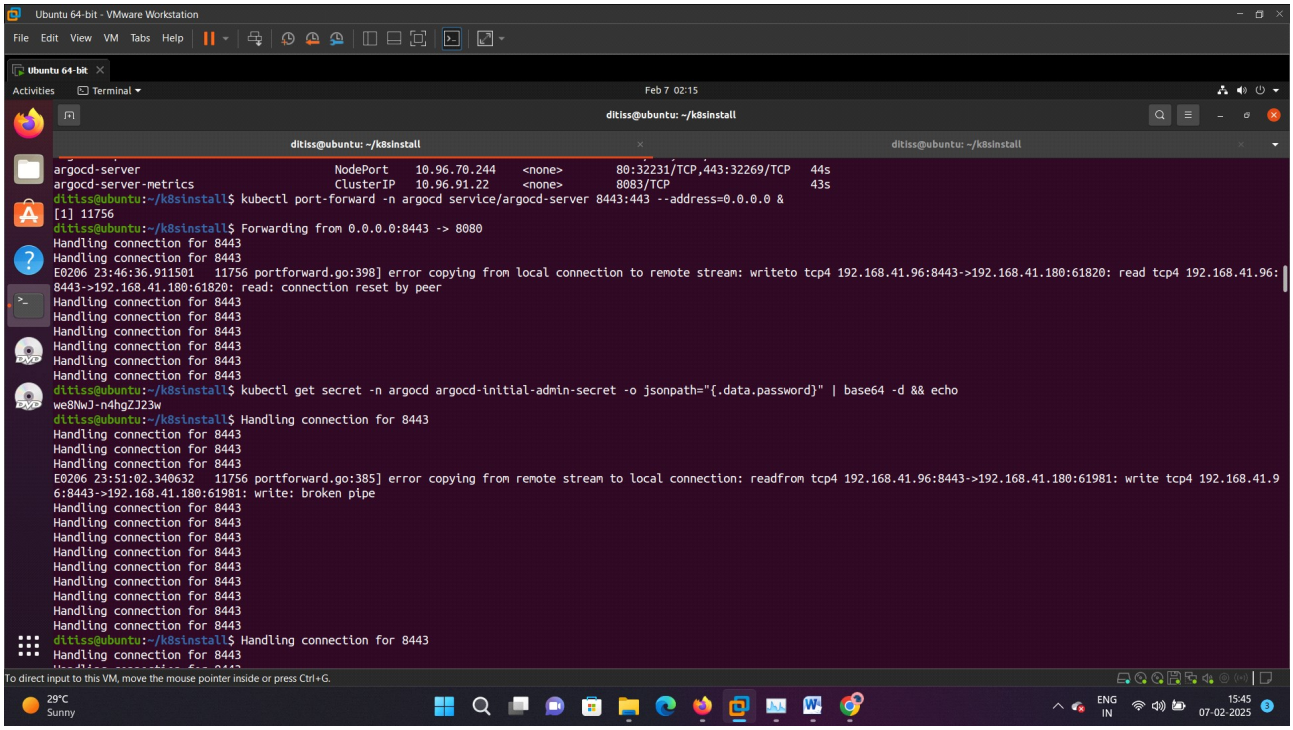




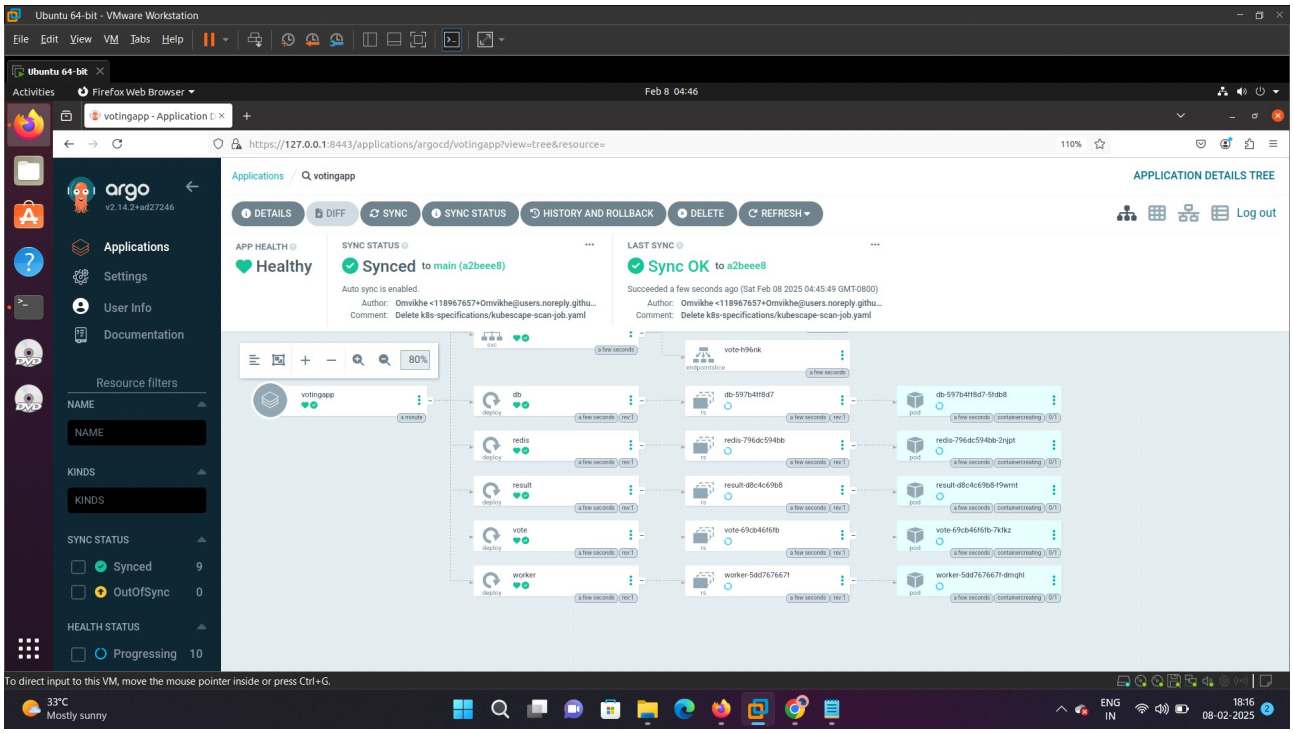
ArgoCD port forwarding=8443



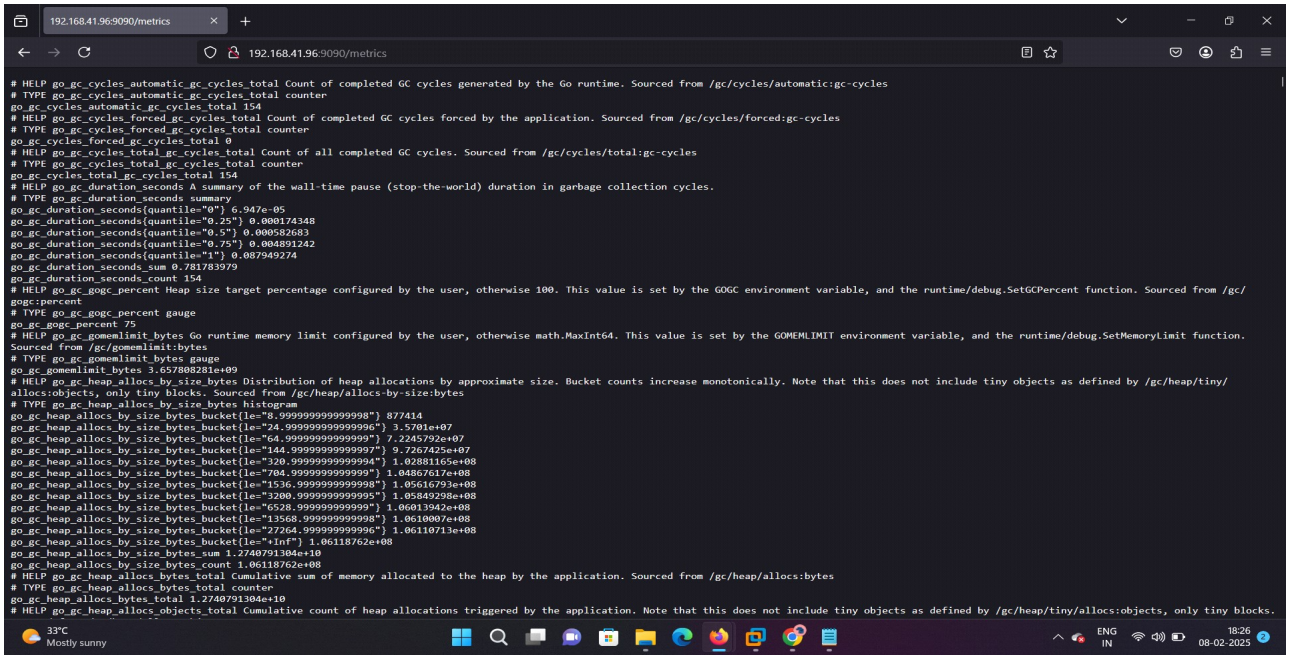
Credentials for ArgoCD login



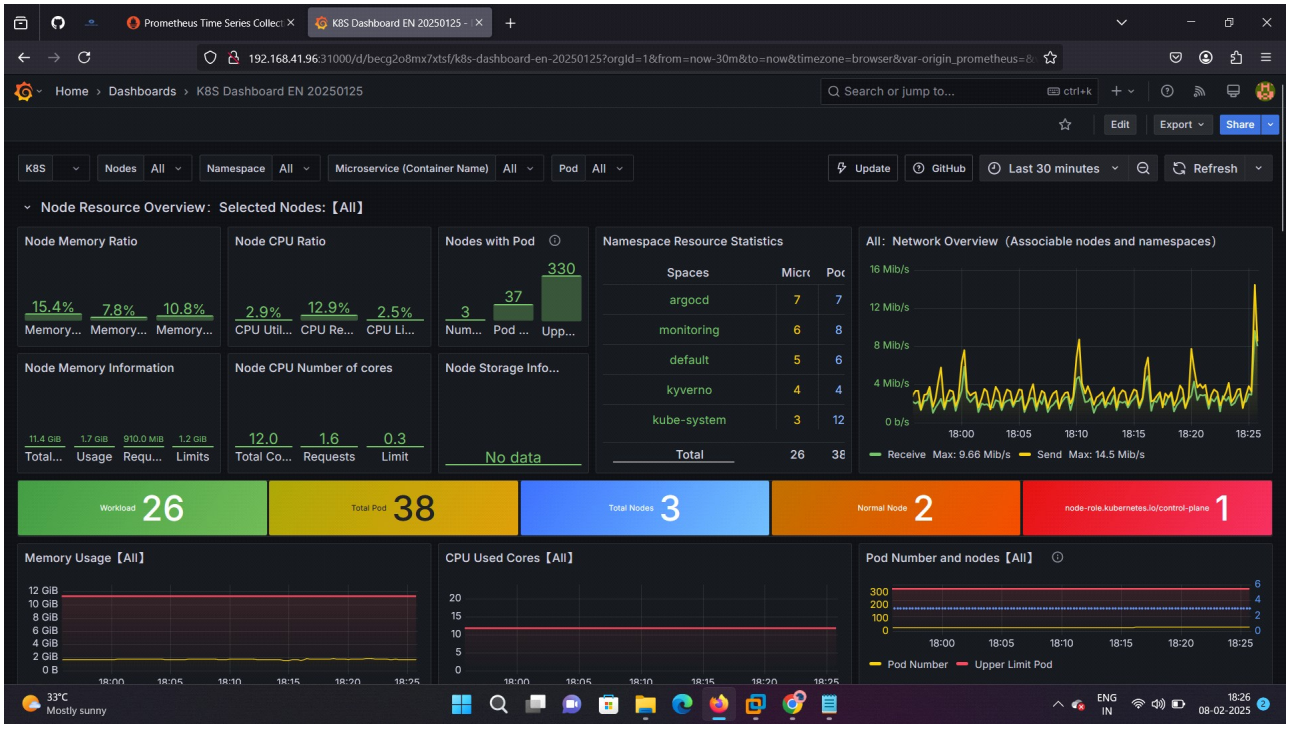
Voting application deployed

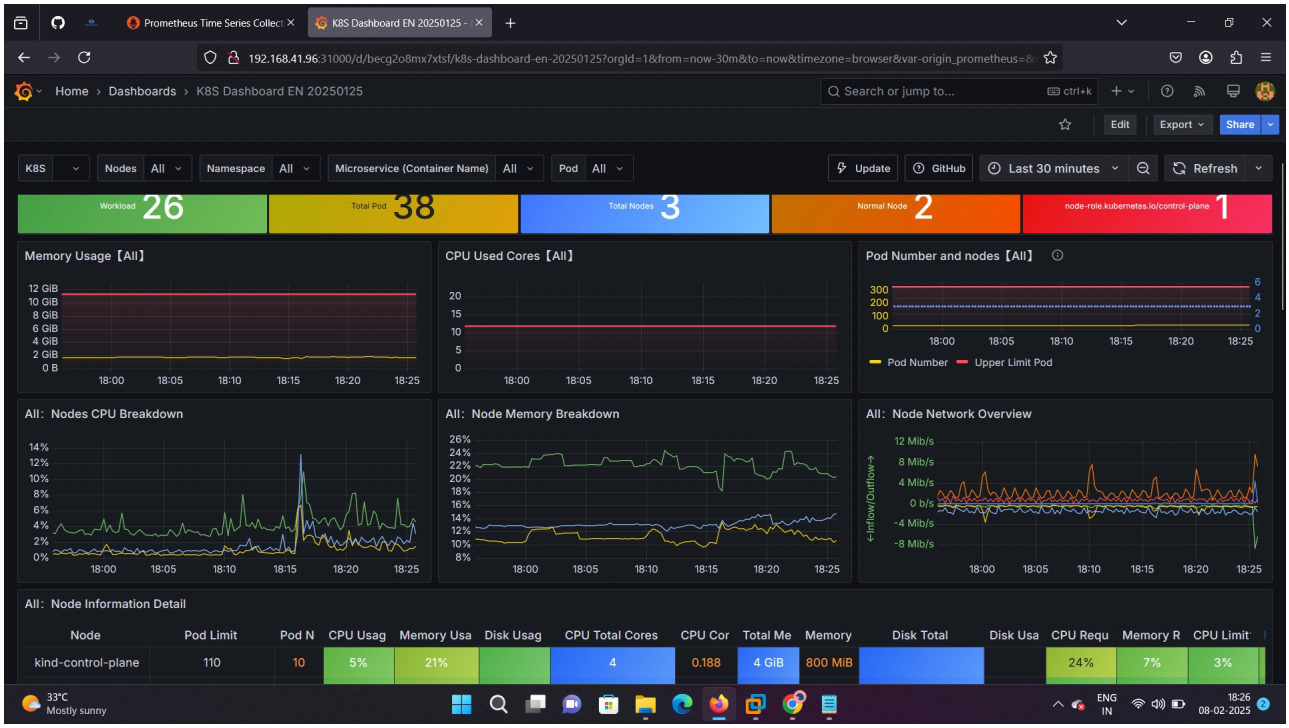


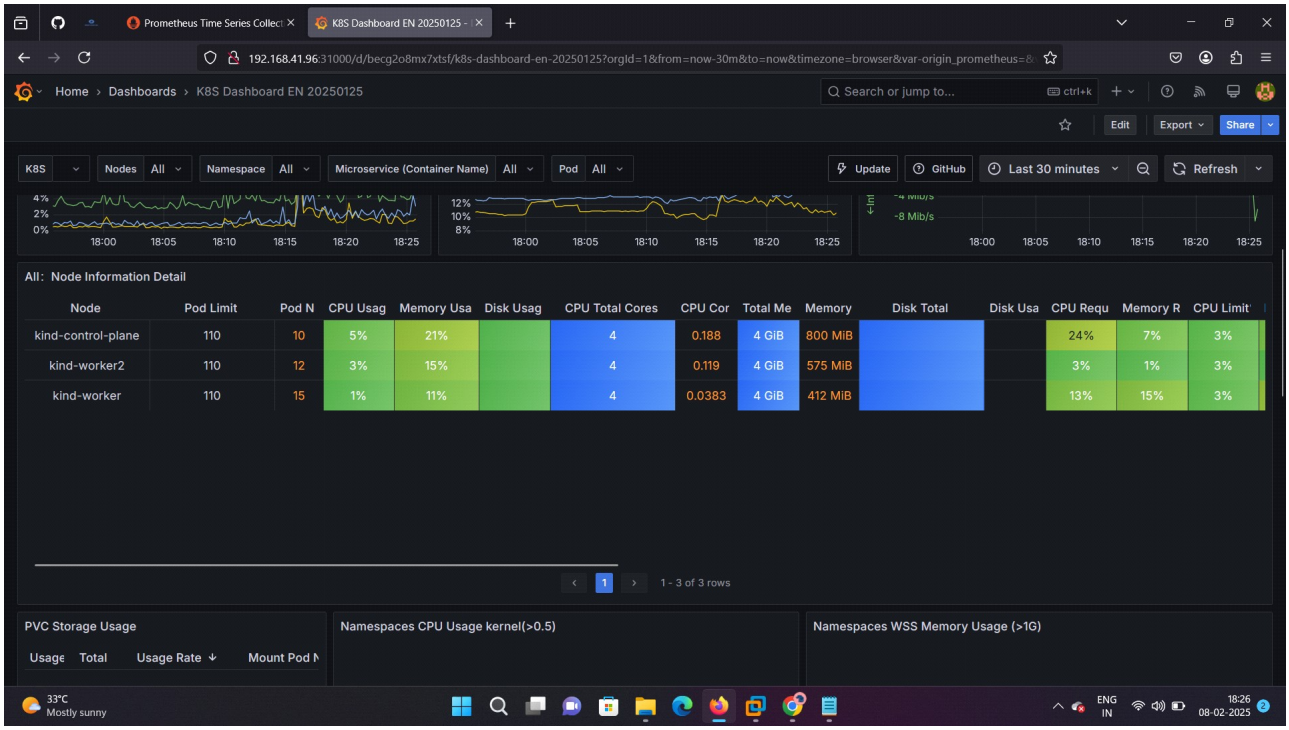
Prometheus metrics of an deployment

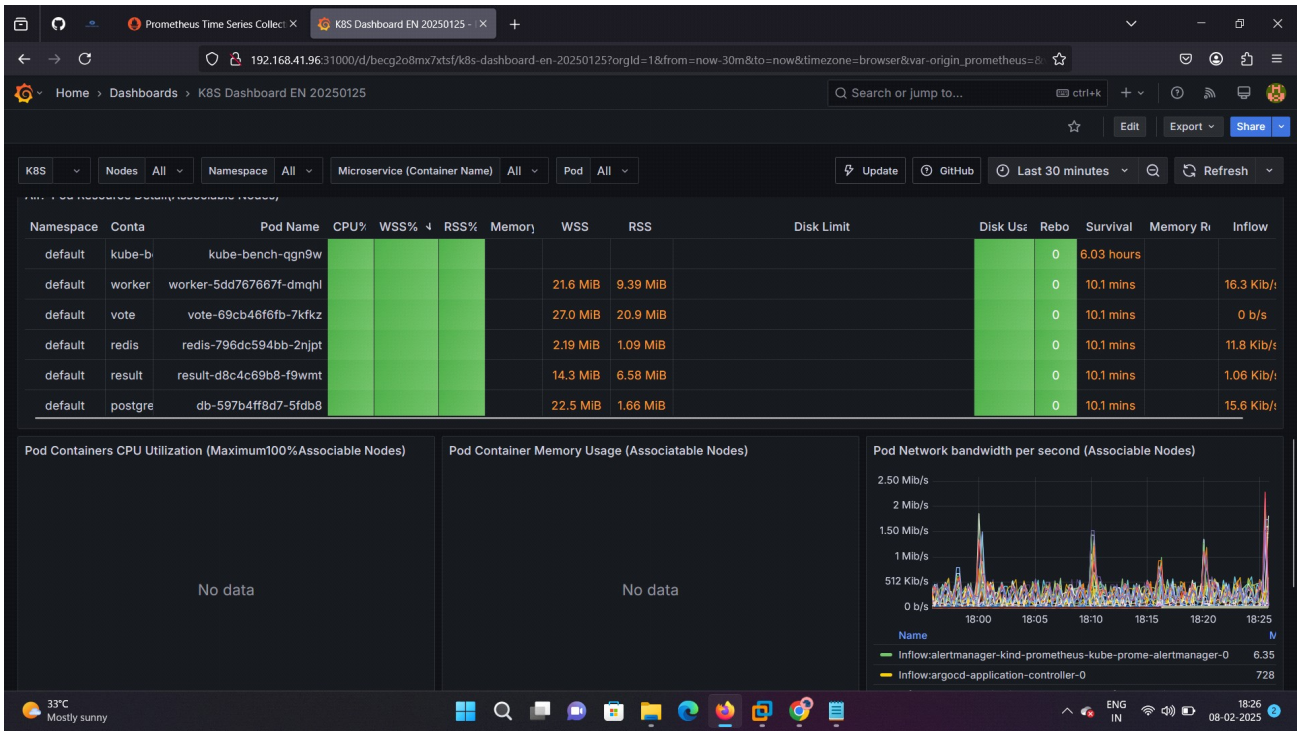


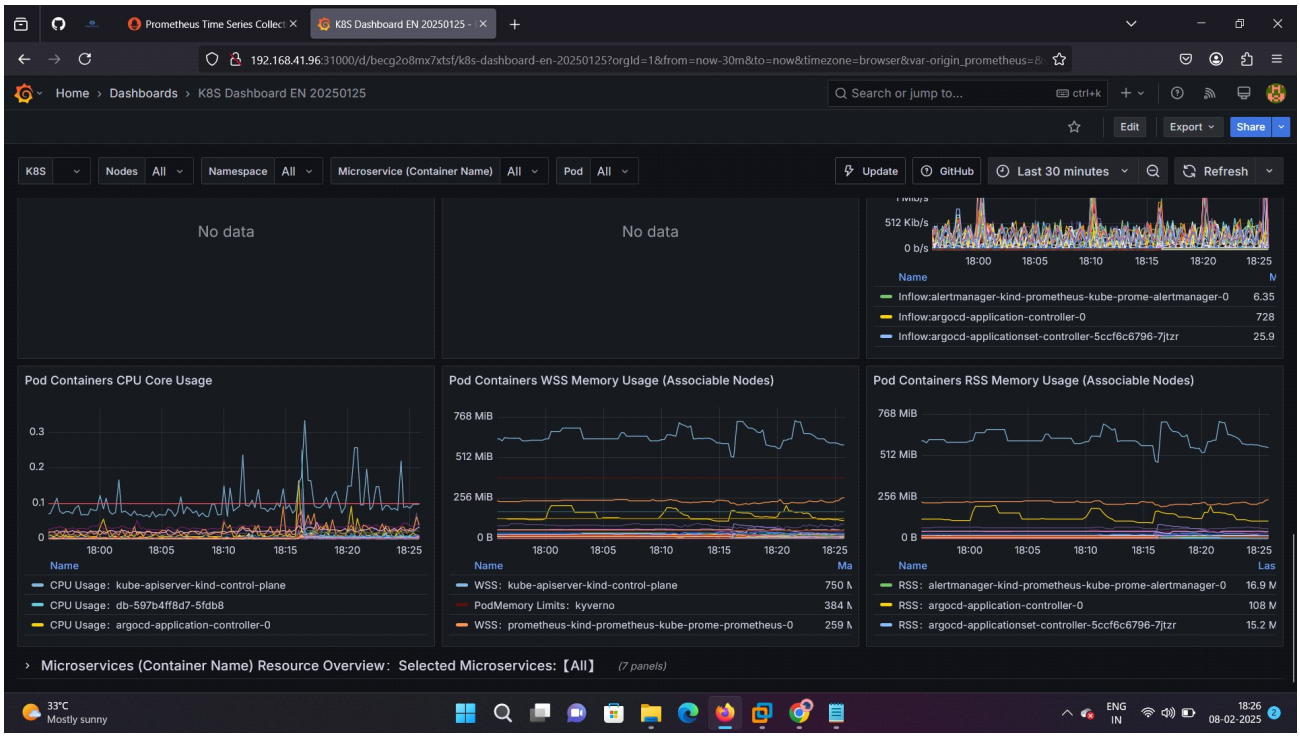
Metrics visualization via Grafana



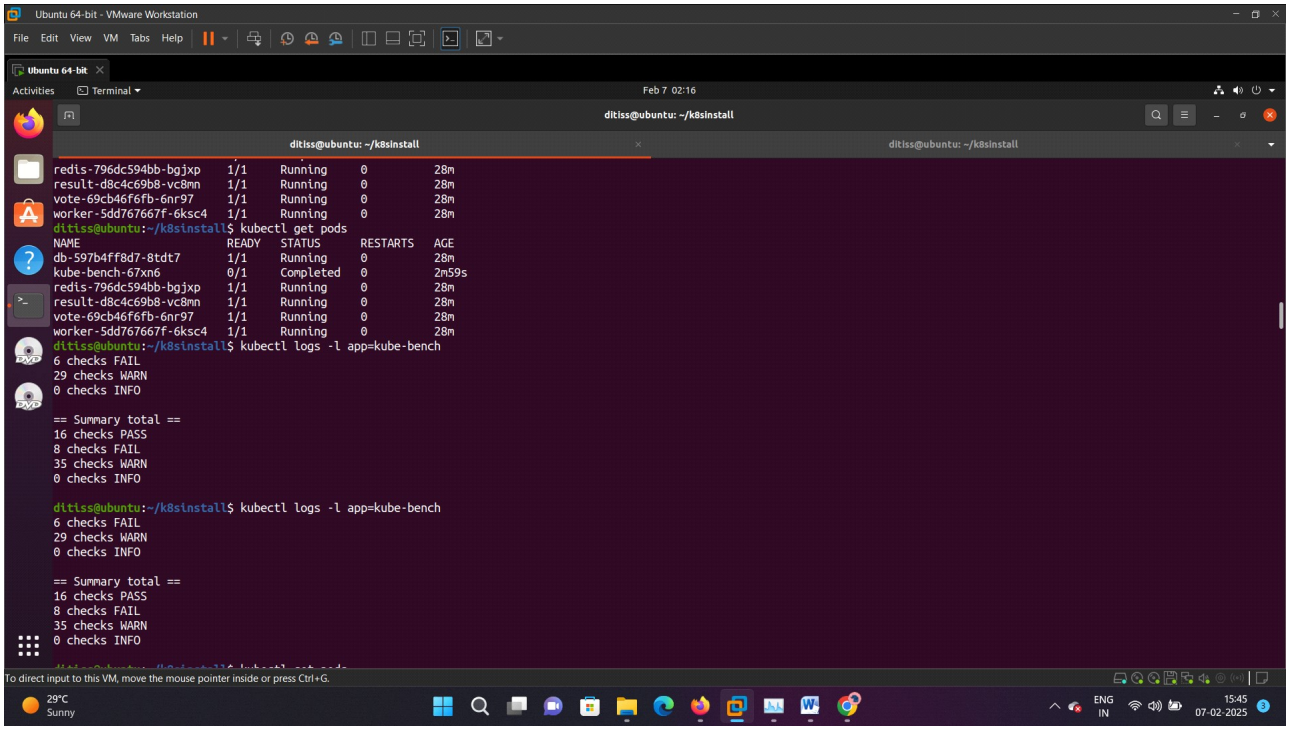

Prometheus Time Series Collect
K8S Dashboard EN 20250125 -
192.168.41.96:31000/d/becg2o8mx7xtsf/k8s-dashboard-en-20250125?orgId=1&from=now-30m&to=now&timezone=browser&var-origin_prometheus=&
Home › Dashboards › K8S Dashboard EN 20250125
Search or jump to...
Edit
Export
Share
K8S
Nodes All
Namespace All
Microservice (Container Name) All
Pod All
Update
GitHub
Last 30 minutes
Refresh
Workload 26
Total Pod 38
Total Nodes 3
Normal Node 2
node-role.kubernetes.io/control-plane 1
Memory Usage【All】
CPU Used Cores【All】
Pod Number and nodes【All】
Pod Number
Upper Limit Pod
All: Nodes CPU Breakdown
All: Node Memory Breakdown
All: Node Network Overview
All: Node Information Detail
Node
Pod Limit
Pod N
CPU Usag
Memory Usa
Disk Usag
CPU Total Cores
CPU Cor
Total Me
Memory
Disk Total
Disk Usa
CPU Requ
Memory R
CPU Limit
kind-control-plane
110
10
5%
21%
4
0.188
4 GiB
800 MiB
24%
7%
3%
33°C
Mostly sunny
ENG
IN
18:26
08-02-2025

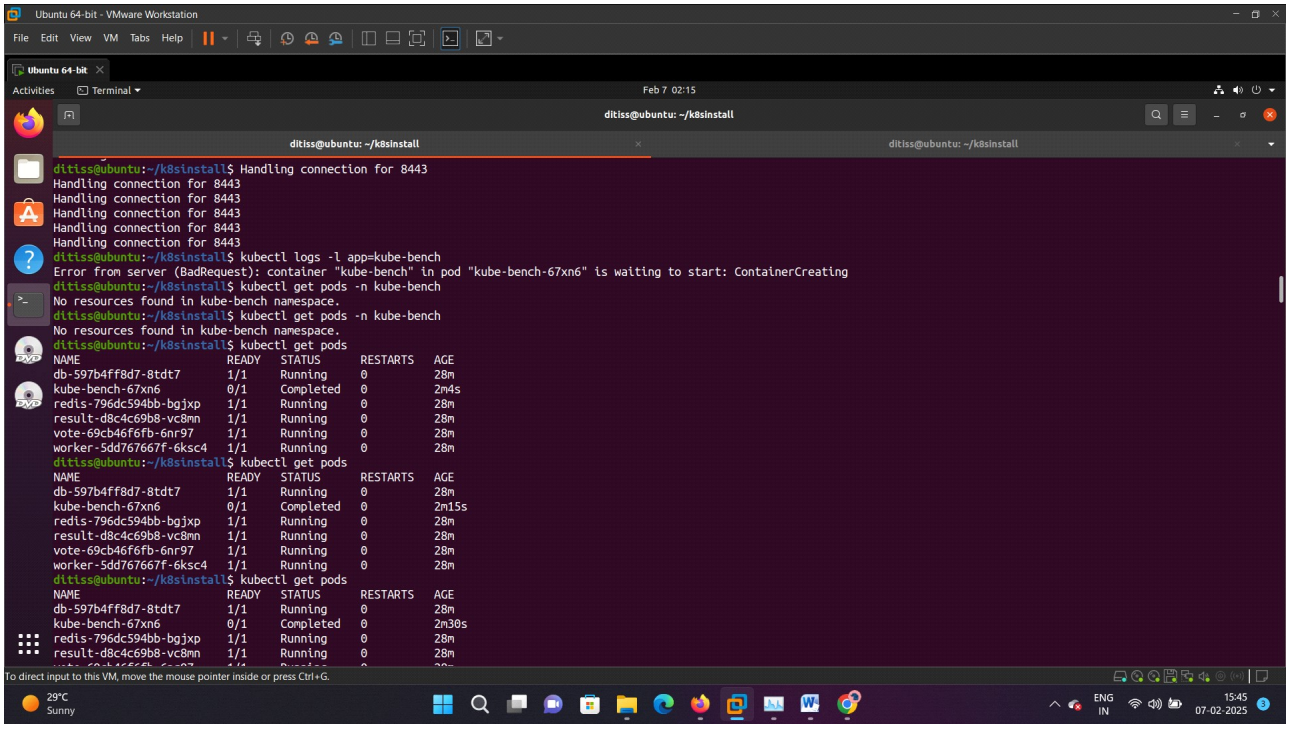

Prometheus Time Series Collect
K8S Dashboard EN 20250125 -
192.168.41.96:31000/d/becg2o8mx7xtsf/k8s-dashboard-en-20250125?orgId=1&from=now-30m&to=now&timezone=browser&var-origin_prometheus=&
Home › Dashboards › K8S Dashboard EN 20250125
Search or jump to...
Edit
Export
Share
K8S
Nodes All
Namespace All
Microservice (Container Name) All
Pod All
Update
GitHub
Last 30 minutes
Refresh
All: Node Information Detail
Node
Pod Limit
Pod N
CPU Usag
Memory Usa
Disk Usag
CPU Total Cores
CPU Cor
Total Me
Memory
Disk Total
Disk Usa
CPU Requ
Memory R
CPU Limit
kind-control-plane
110
10
5%
21%
4
0.188
4 GiB
800 MiB
24%
7%
3%
kind-worker2
110
12
3%
15%
4
0.119
4 GiB
575 MiB
3%
1%
3%
kind-worker
110
15
1%
11%
4
0.0383
4 GiB
412 MiB
13%
15%
3%
1 - 3 of 3 rows
PVC Storage Usage
Usage
Total
Usage Rate
Mount Pod
Namespaces CPU Usage kernel(>0.5)
Namespaces WSS Memory Usage (>1G)
33°C
Mostly sunny
ENG
IN
18:26
08-02-2025


Prometheus Time Series Collect
K8S Dashboard EN 20250125 -
192.168.41.96:31000/d/becg2o8mx7xtsf/k8s-dashboard-en-20250125?orgId=1&from=now-30m&to=now&timezone=browser&var-origin_prometheus=&
Home › Dashboards › K8S Dashboard EN 20250125
Search or jump to...
Edit
Export
Share
K8S
Nodes All
Namespace All
Microservice (Container Name) All
Pod All
Update
GitHub
Last 30 minutes
Refresh
Namespace Conta Pod Name CPU% WSS% RSS% Memory WSS RSS Disk Limit Disk Usa Rebo Survival Memory R Inflow
default kube-b kube-bench-qgn9w 0 6.03 hours
default worker worker-5dd767667f-dmqhl 21.6 MiB 9.39 MiB 0 10.1 mins 16.3 Kib/s
default vote vote-69cb46f6fb-7kfkz 27.0 MiB 20.9 MiB 0 10.1 mins 0 b/s
default redis redis-796dc594bb-2njpt 2.19 MiB 1.09 MiB 0 10.1 mins 11.8 Kib/s
default result result-d8c4c69b8-f9wmt 14.3 MiB 6.58 MiB 0 10.1 mins 1.06 Kib/s
default postgre db-597b4ff8d7-5fdb8 22.5 MiB 1.66 MiB 0 10.1 mins 15.6 Kib/s
Pod Containers CPU Utilization (Maximum100%Associable Nodes)
No data
Pod Container Memory Usage (Associatable Nodes)
No data
Pod Network bandwidth per second (Associable Nodes)
2.50 Mib/s
2 Mib/s
1.50 Mib/s
1 Mib/s
512 Kib/s
0 b/s
18:00 18:05 18:10 18:15 18:20 18:25
Name
Inflow:alertmanager-kind-prometheus-kube-prome-alertmanager-0 6.35
Inflow:argocd-application-controller-0 728
33°C
Mostly sunny
ENG
IN
18:26
08-02-2025

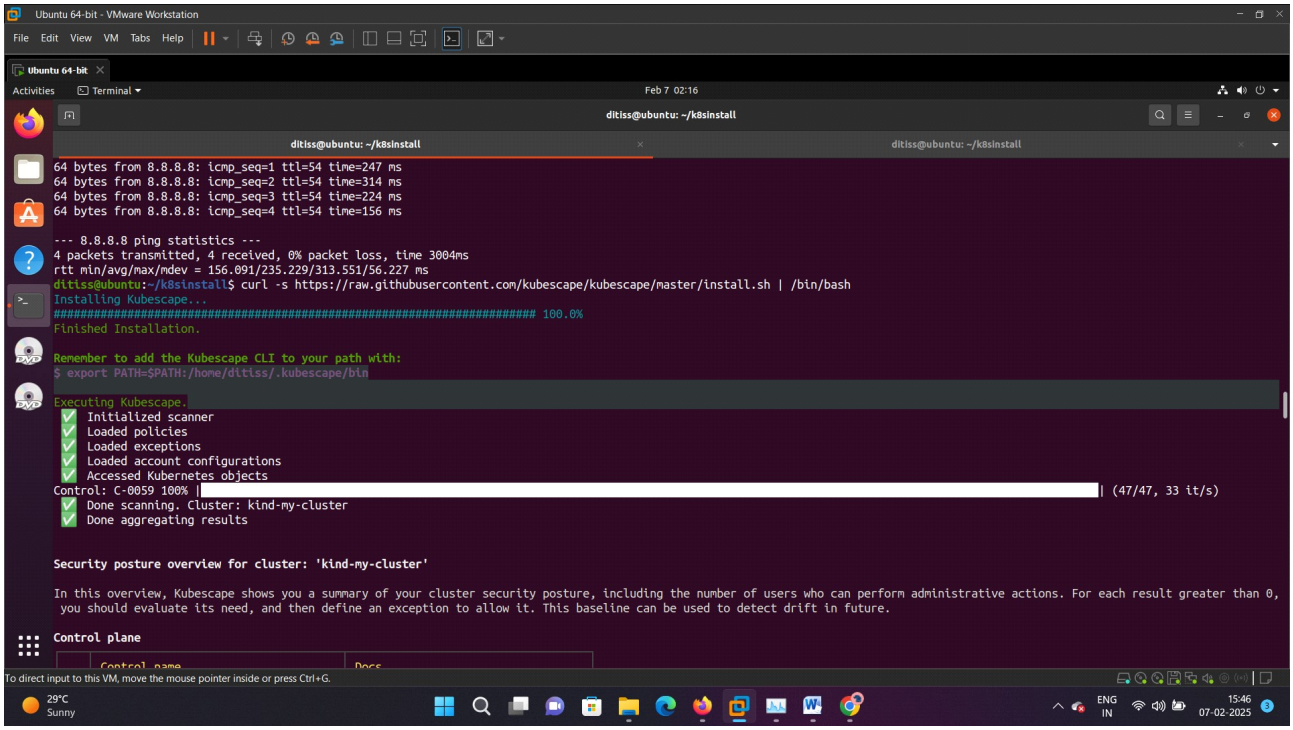

Prometheus Time Series Collect
K8S Dashboard EN 20250125 -
192.168.41.96:31000/d/becg2o8mx7xtsf/k8s-dashboard-en-20250125?orgId=1&from=now-30m&to=now&timezone=browser&var-origin_prometheus=&
Home › Dashboards › K8S Dashboard EN 20250125
Search or jump to...
Edit
Export
Share
K8S
Nodes All
Namespace All
Microservice (Container Name) All
Pod All
Update
GitHub
Last 30 minutes
Refresh
No data
No data
512 Kib/s
0 b/s
18:00 18:05 18:10 18:15 18:20 18:25
Name
Inflow:alertmanager-kind-prometheus-kube-prome-alertmanager-0 6.35
Inflow:argocd-application-controller-0 728
Inflow:argocd-applicationset-controller-5ccf6c6796-7jtzr 25.9
Pod Containers CPU Core Usage
0.3
0.2
0.1
0
18:00 18:05 18:10 18:15 18:20 18:25
Name
CPU Usage: kube-apiserver-kind-control-plane
CPU Usage: db-597b4ff8d7-5fdb8
CPU Usage: argocd-application-controller-0
Pod Containers WSS Memory Usage (Associable Nodes)
768 MiB
512 MiB
256 MiB
0 B
18:00 18:05 18:10 18:15 18:20 18:25
Name
WSS: kube-apiserver-kind-control-plane 750 M
PodMemory Limits: kyverno 384 M
WSS: prometheus-kind-prometheus-kube-prome-prometheus-0 259 M
Pod Containers RSS Memory Usage (Associable Nodes)
768 MiB
512 MiB
256 MiB
0 B
18:00 18:05 18:10 18:15 18:20 18:25
Name
RSS: alertmanager-kind-prometheus-kube-prome-alertmanager-0 16.9 M
RSS: argocd-application-controller-0 108 M
RSS: argocd-applicationset-controller-5ccf6c6796-7jtzr 15.2 M
Microservices (Container Name) Resource Overview: Selected Microservices:【All】 (7 panels)
33°C
Mostly sunny
ENG
IN
18:26
08-02-2025

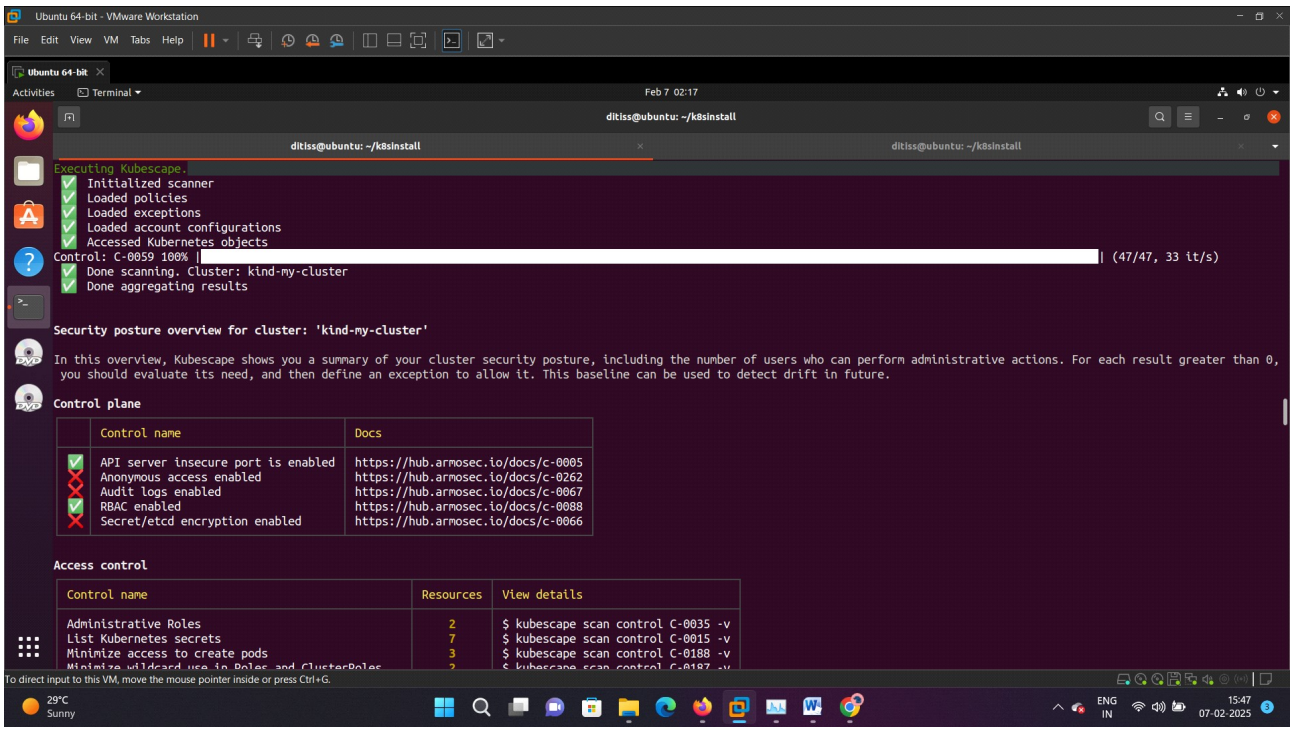
Kube-bench scan report

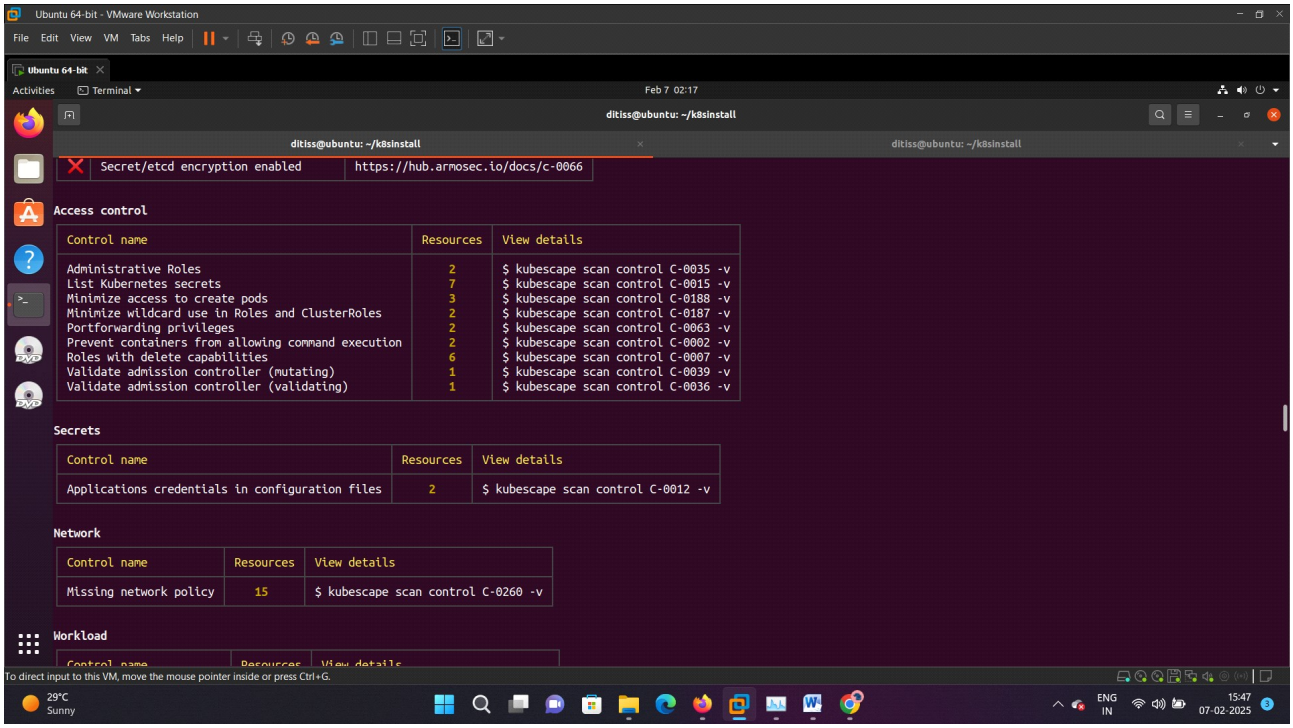


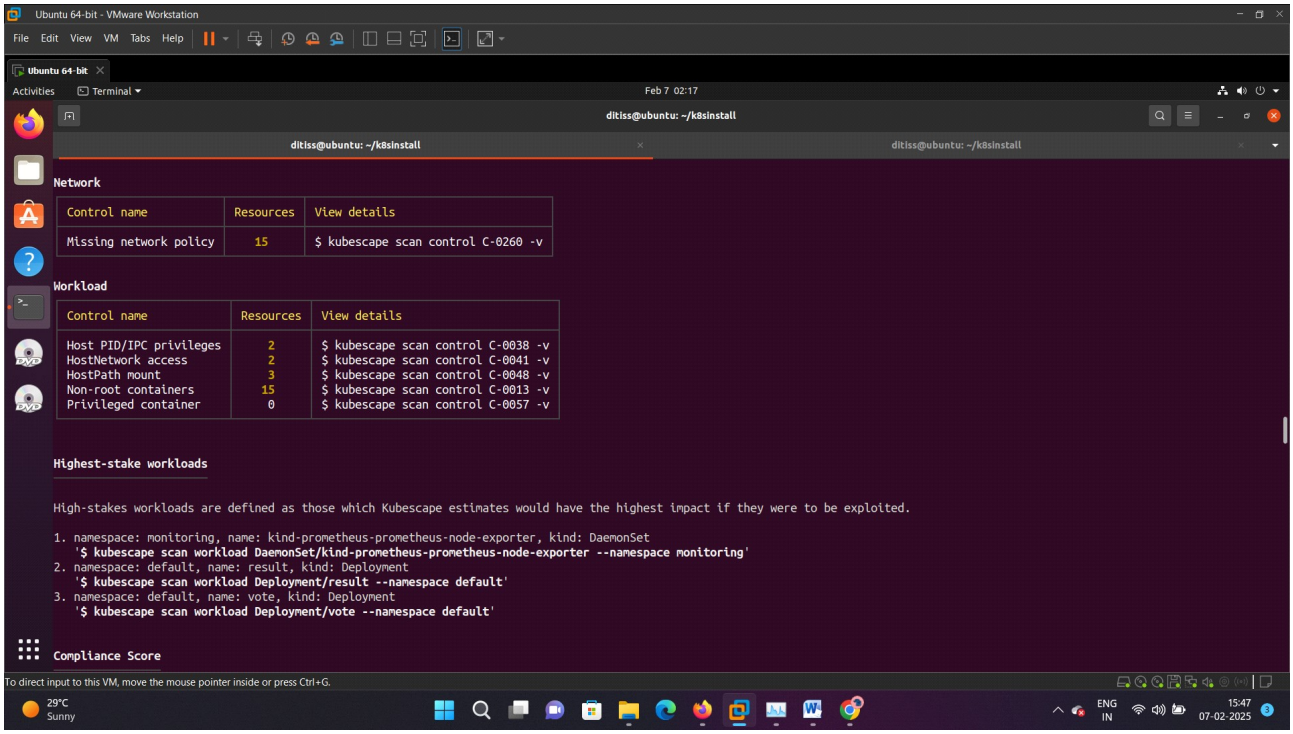


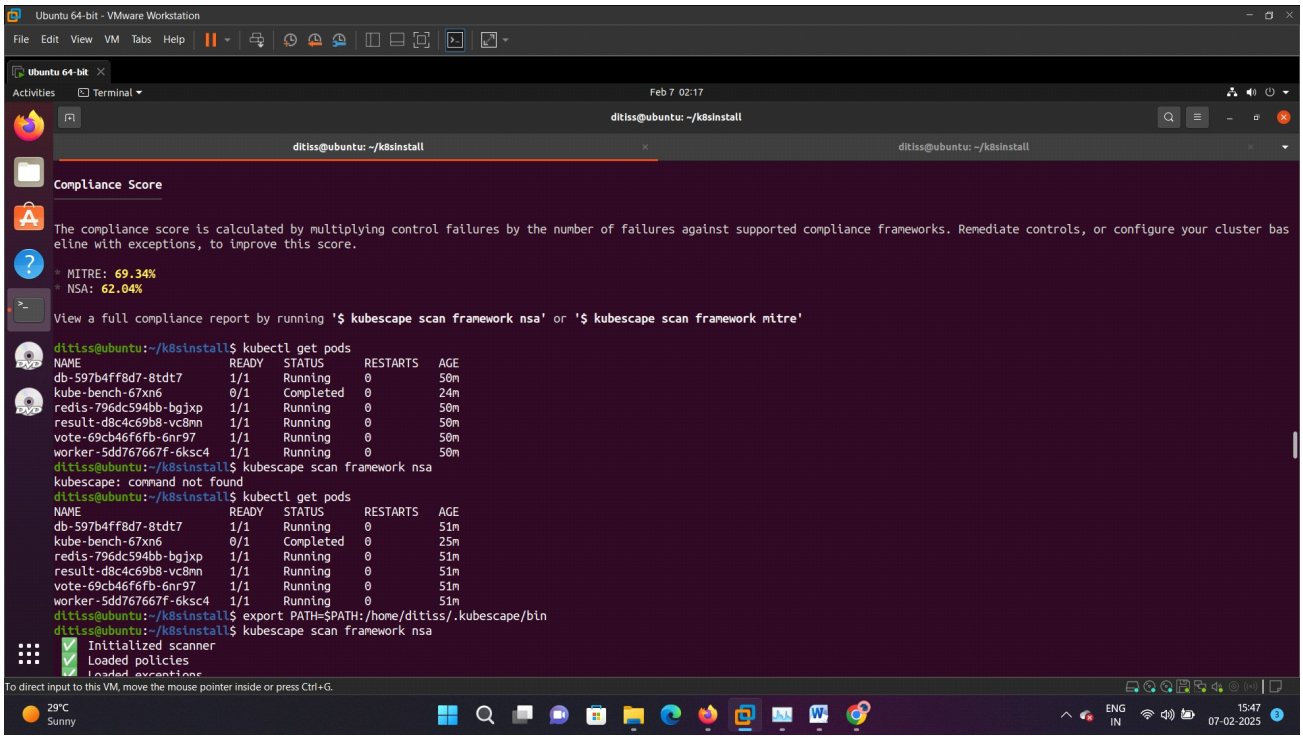
Kubescape installation and report of kubescape scan



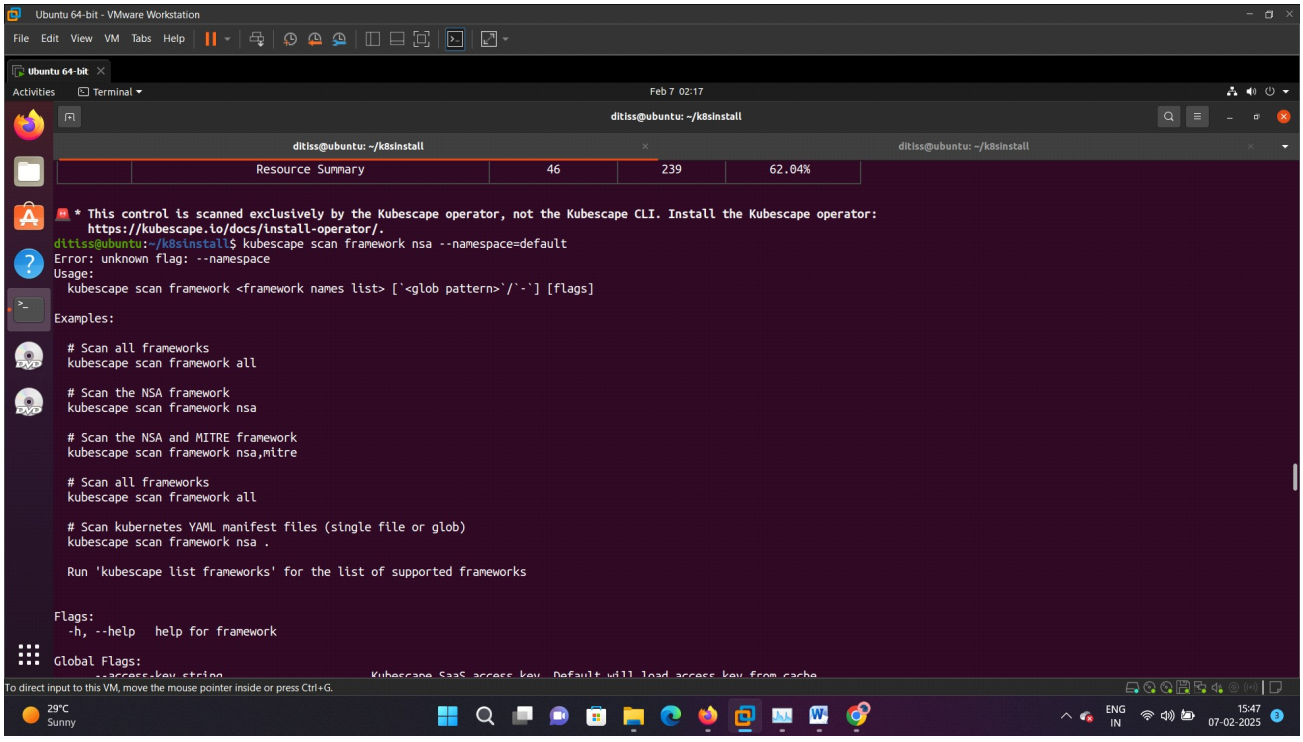


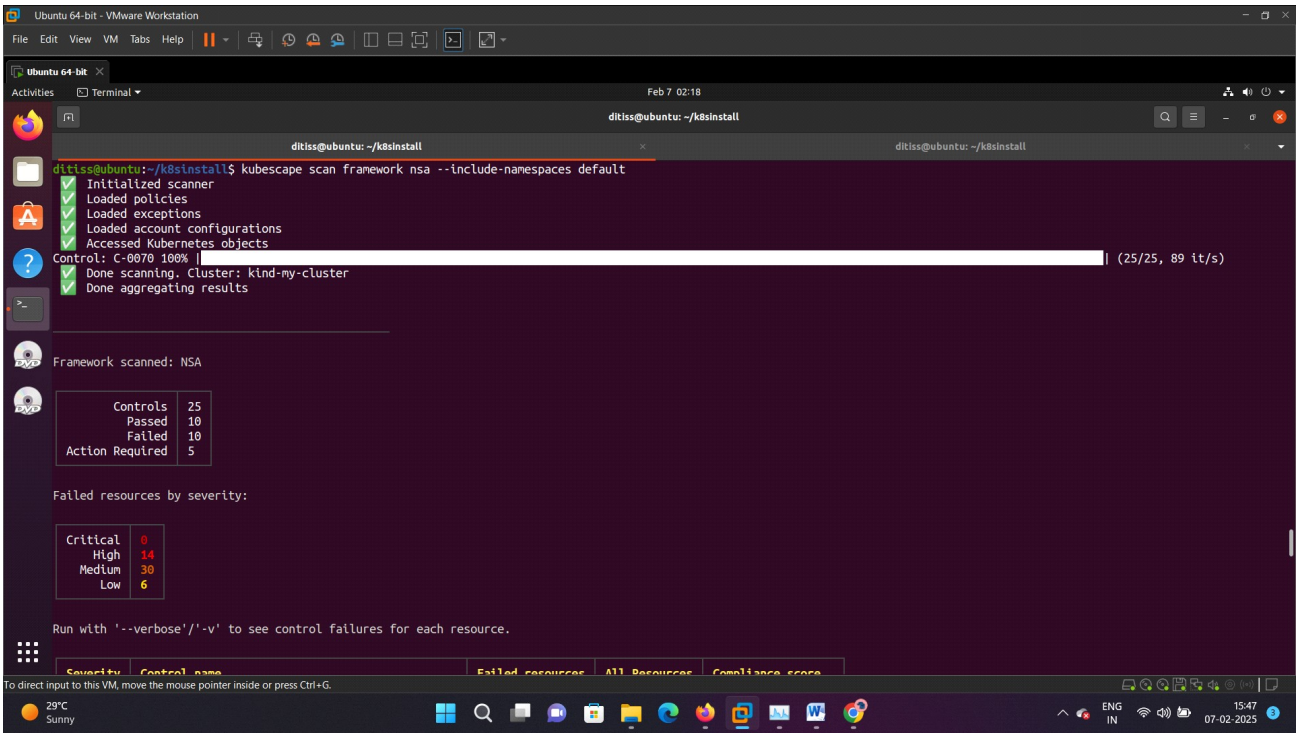

Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help
Ubuntu 64-bit
Activities Terminal Feb 7 02:17
ditiss@ubuntu: ~/k8sinstall
ditiss@ubuntu: ~/k8sinstall
ditiss@ubuntu: ~/k8sinstall
Secret/etcd encryption enabled https://hub.armosec.io/docs/c-0066
Access control
Control name Resources View details
Administrative Roles 2 $ kubescape scan control C-0035 -v
List Kubernetes secrets 7 $ kubescape scan control C-0015 -v
Minimize access to create pods 3 $ kubescape scan control C-0188 -v
Minimize wildcard use in Roles and ClusterRoles 2 $ kubescape scan control C-0187 -v
Portforwarding privileges 2 $ kubescape scan control C-0063 -v
Prevent containers from allowing command execution 2 $ kubescape scan control C-0002 -v
Roles with delete capabilities 6 $ kubescape scan control C-0007 -v
Validate admission controller (mutating) 1 $ kubescape scan control C-0039 -v
Validate admission controller (validating) 1 $ kubescape scan control C-0036 -v
Secrets
Control name Resources View details
Applications credentials in configuration files 2 $ kubescape scan control C-0012 -v
Network
Control name Resources View details
Missing network policy 15 $ kubescape scan control C-0260 -v
Workload
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
29°C Sunny
ENG IN
15:47 07-02-2025

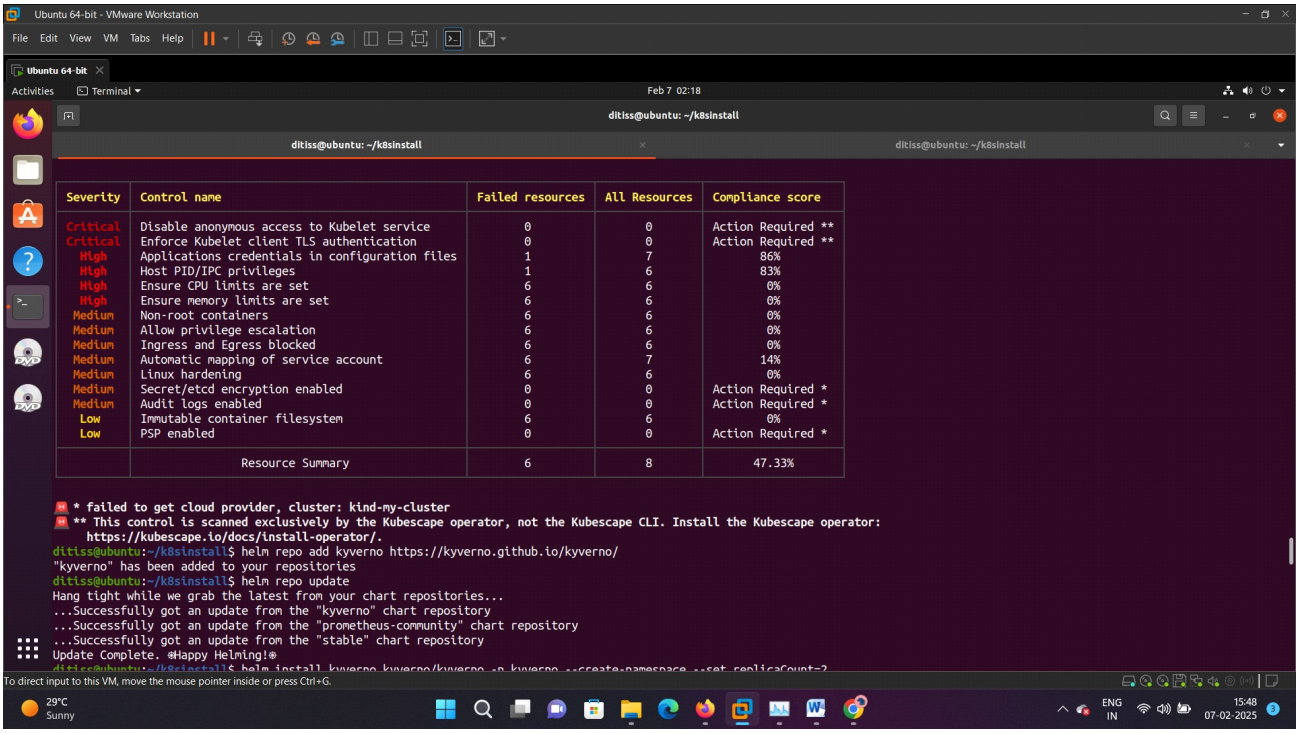

Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help
Ubuntu 64-bit
Activities Terminal Feb 7 02:17
ditiss@ubuntu: ~/k8sinstall
ditiss@ubuntu: ~/k8sinstall
ditiss@ubuntu: ~/k8sinstall
Network
Control name Resources View details
Missing network policy 15 $ kubescape scan control C-0260 -v
Workload
Control name Resources View details
Host PID/IPC privileges 2 $ kubescape scan control C-0038 -v
HostNetwork access 2 $ kubescape scan control C-0041 -v
HostPath mount 3 $ kubescape scan control C-0048 -v
Non-root containers 15 $ kubescape scan control C-0013 -v
Privileged container 0 $ kubescape scan control C-0057 -v
Highest-stake workloads
High-stakes workloads are defined as those which Kubescape estimates would have the highest impact if they were to be exploited.
1. namespace: monitoring, name: kind-prometheus-prometheus-node-exporter, kind: DaemonSet
'$ kubescape scan workload DaemonSet/kind-prometheus-prometheus-node-exporter --namespace monitoring'
2. namespace: default, name: result, kind: Deployment
'$ kubescape scan workload Deployment/result --namespace default'
3. namespace: default, name: vote, kind: Deployment
'$ kubescape scan workload Deployment/vote --namespace default'
Compliance Score
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
29°C Sunny
ENG IN
15:47 07-02-2025



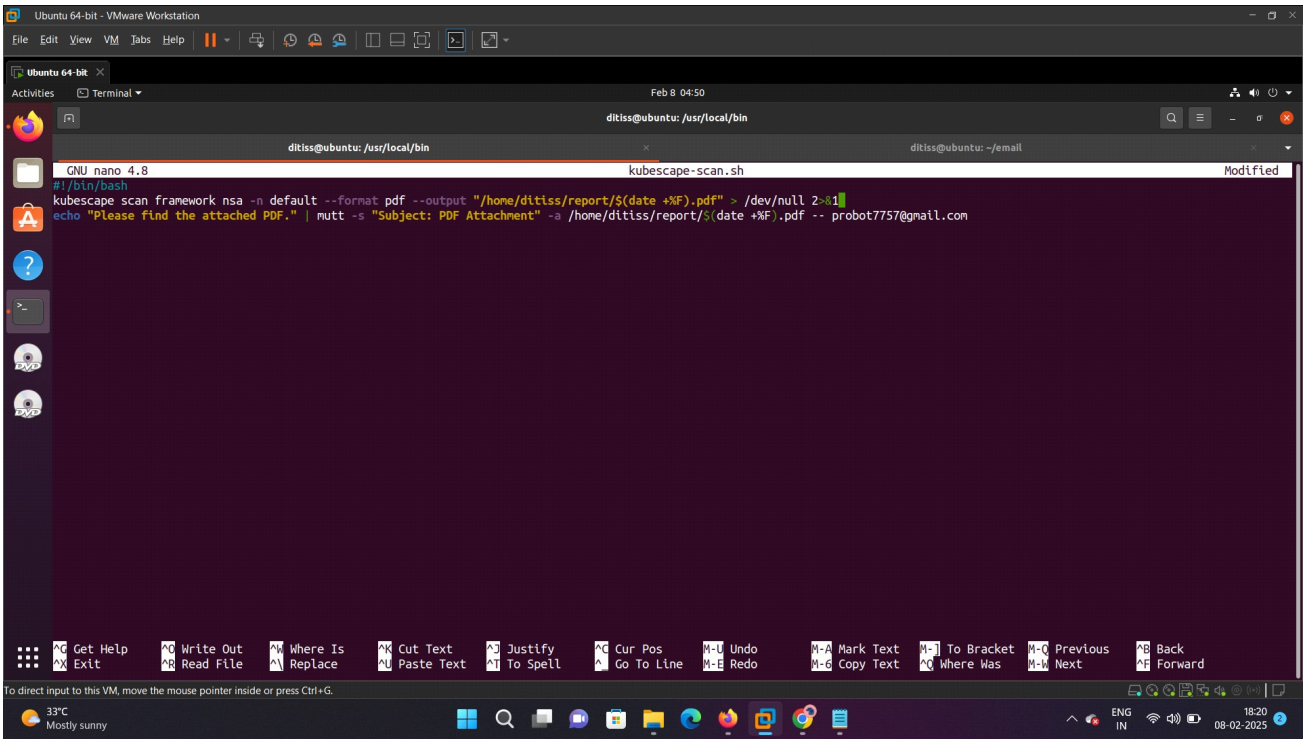
Kubescape scan report of default namespace where app is deployed against nsa framework



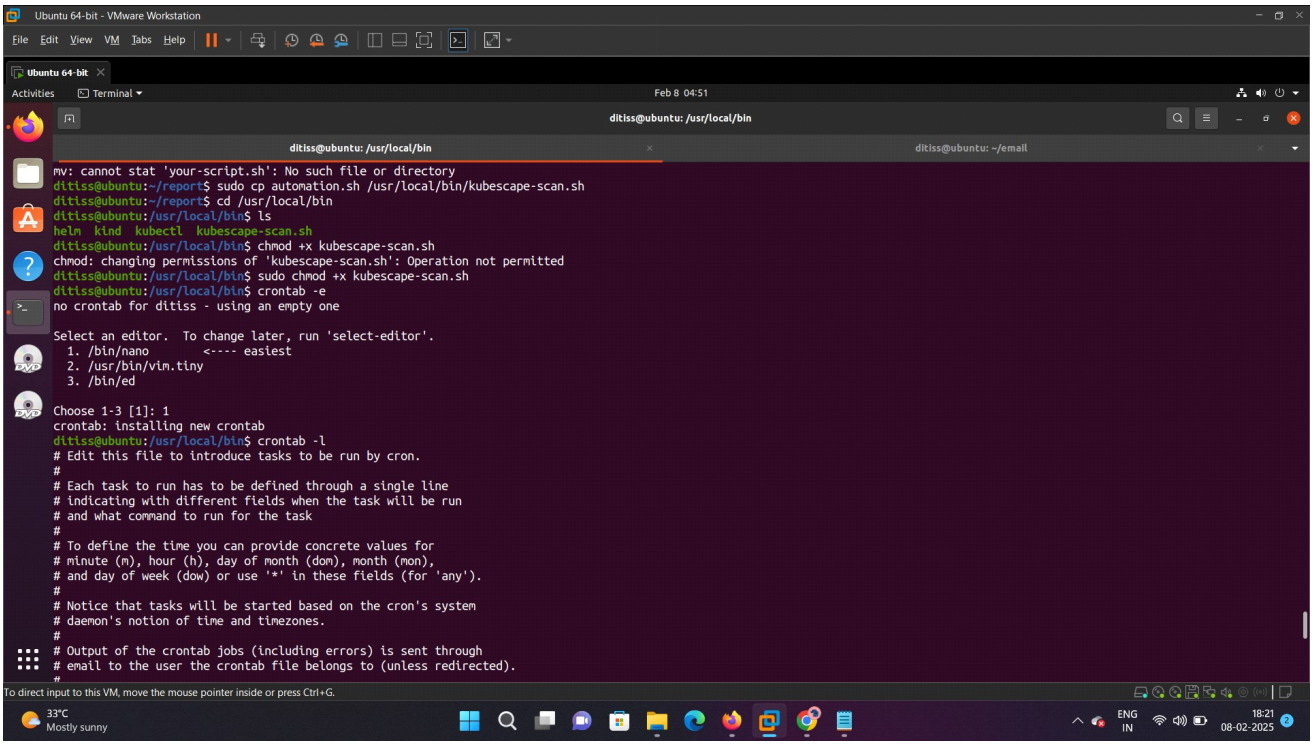

Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help
Ubuntu 64-bit
Activities Terminal Feb 7 02:18
ditiss@ubuntu: ~/k8sinstall
ditiss@ubuntu: ~/k8sinstall
ditiss@ubuntu: ~/k8sinstall
ditiss@ubuntu:~/k8sinstall$ kubescape scan framework nsa --include-namespaces default
Initialized scanner
Loaded policies
Loaded exceptions
Loaded account configurations
Accessed Kubernetes objects
Control: C-0070 100% | (25/25, 89 it/s)
Done scanning. Cluster: kind-my-cluster
Done aggregating results
Framework scanned: NSA
Controls 25
Passed 10
Failed 10
Action Required 5
Failed resources by severity:
Critical 0
High 14
Medium 30
Low 6
Run with '--verbose'/'-v' to see control failures for each resource.
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
29°C Sunny
ENG IN
15:47 07-02-2025

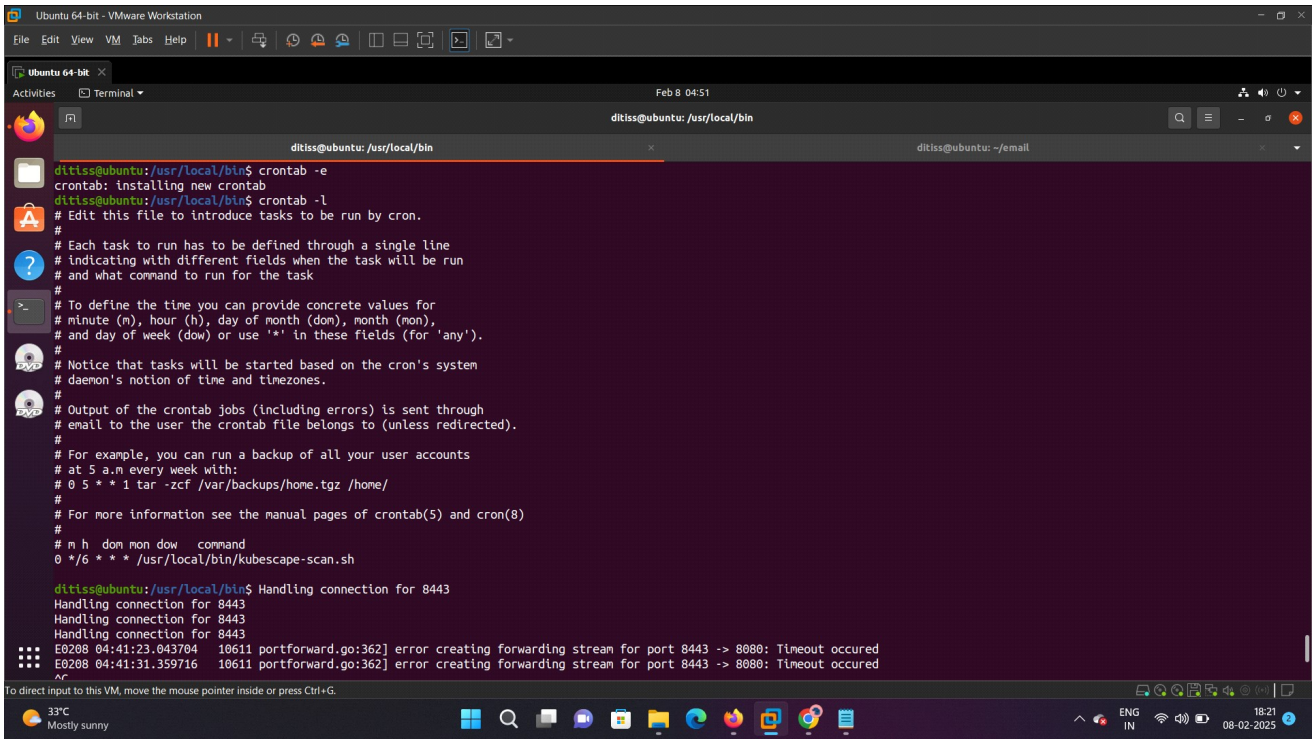

Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help
Ubuntu 64-bit
Activities Terminal Feb 7 02:18
ditiss@ubuntu: ~/k8sinstall
ditiss@ubuntu: ~/k8sinstall
Severity Control name Failed resources All Resources Compliance score
Critical Disable anonymous access to Kubelet service 0 0 Action Required **
Critical Enforce Kubelet client TLS authentication 0 0 Action Required **
High Applications credentials in configuration files 1 7 86%
High Host PID/IPC privileges 1 6 83%
High Ensure CPU limits are set 6 6 0%
High Ensure memory limits are set 6 6 0%
Medium Non-root containers 6 6 0%
Medium Allow privilege escalation 6 6 0%
Medium Ingress and Egress blocked 6 6 0%
Medium Automatic mapping of service account 6 7 14%
Medium Linux hardening 6 6 0%
Medium Secret/etcd encryption enabled 0 0 Action Required *
Medium Audit logs enabled 0 0 Action Required *
Low Immutable container filesystem 6 6 0%
Low PSP enabled 0 0 Action Required *
Resource Summary 6 8 47.33%
* failed to get cloud provider, cluster: kind-my-cluster
** This control is scanned exclusively by the Kubescape operator, not the Kubescape CLI. Install the Kubescape operator:
https://kubescape.io/docs/install-operator/.
ditiss@ubuntu:~/k8sinstall$ helm repo add kyverno https://kyverno.github.io/kyverno/
"kyverno" has been added to your repositories
ditiss@ubuntu:~/k8sinstall$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "kyverno" chart repository
...Successfully got an update from the "prometheus-community" chart repository
...Successfully got an update from the "stable" chart repository
Update Complete. ⎈Happy Helming!⎈
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
29°C Sunny
ENG IN
15:48 07-02-2025

Script to automate report generation and send to email

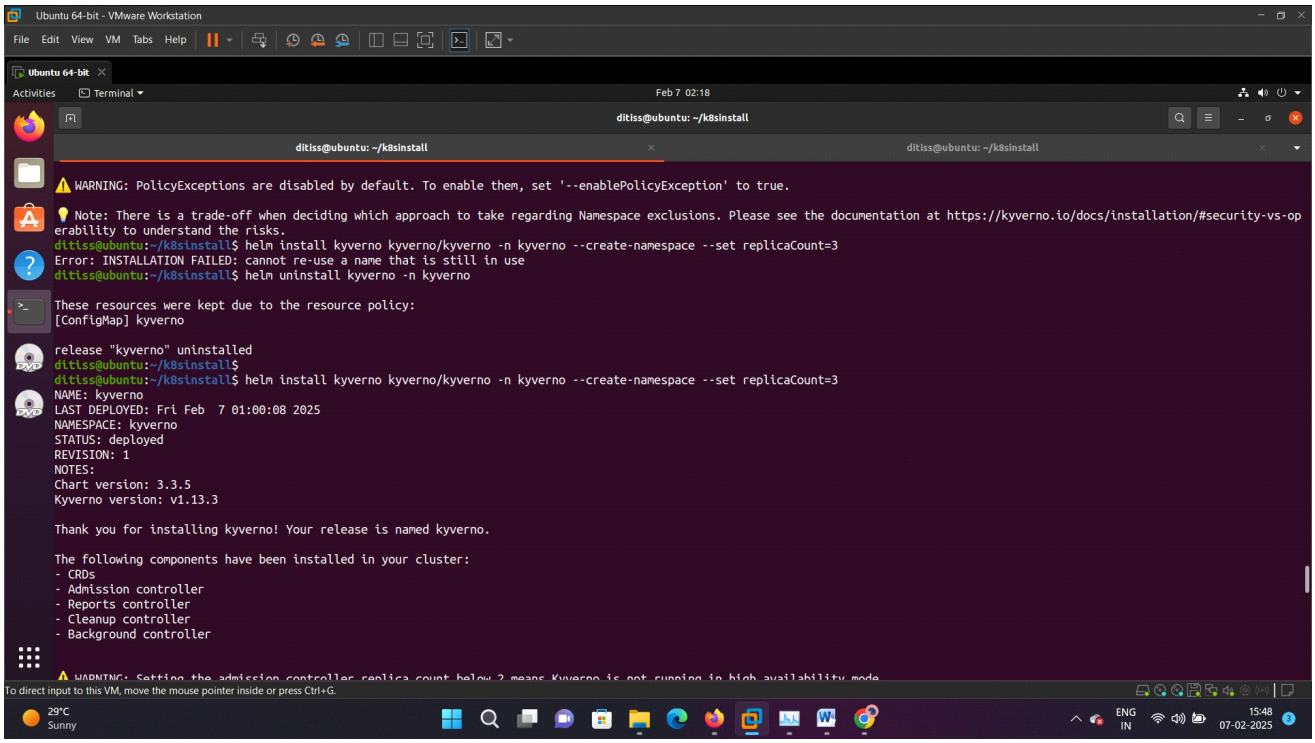


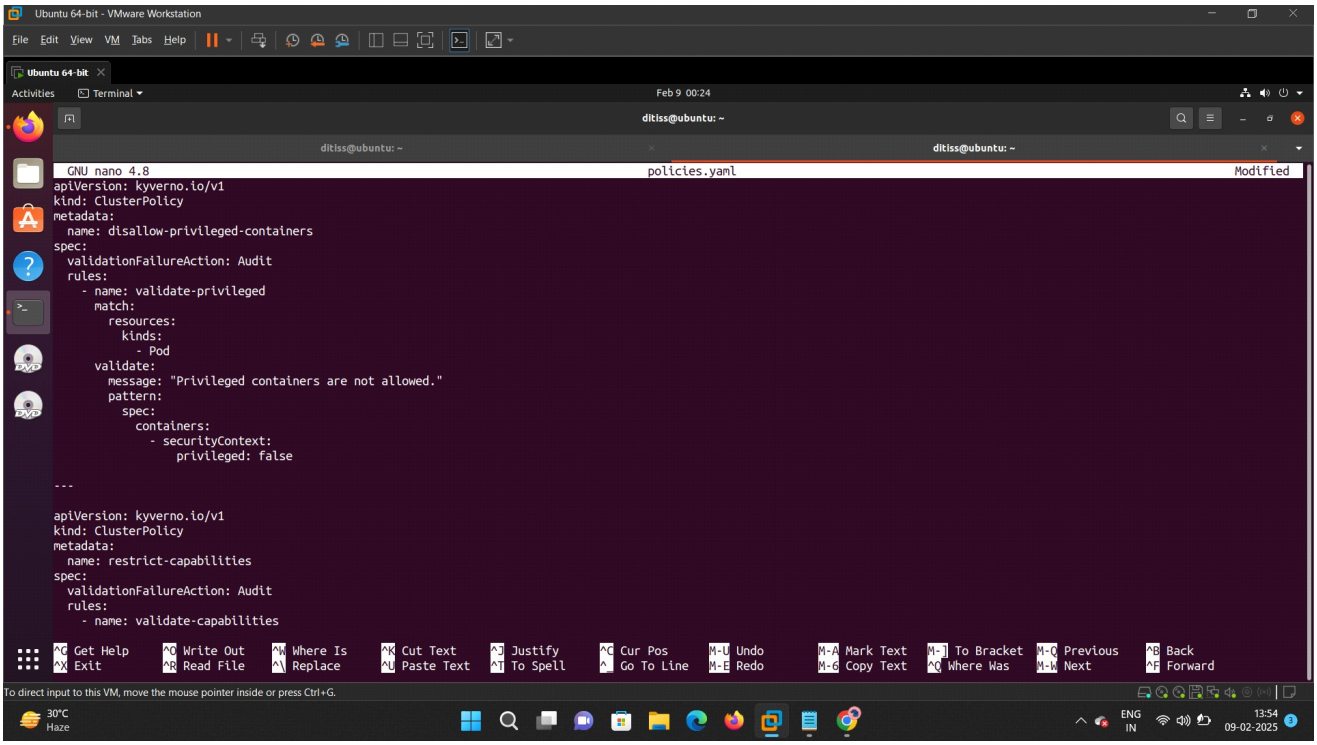
Automate execution/job via crontab it will execute in every 6 hr and send email to mentioned email address

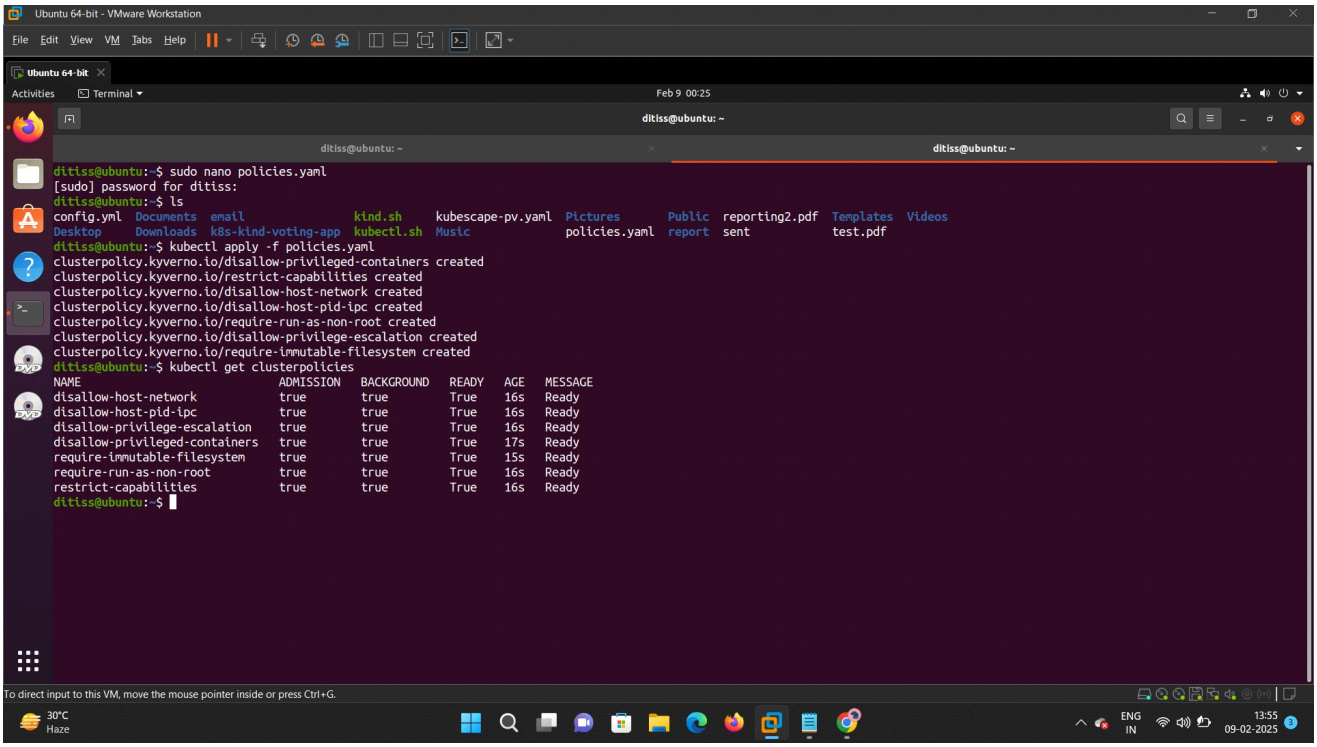


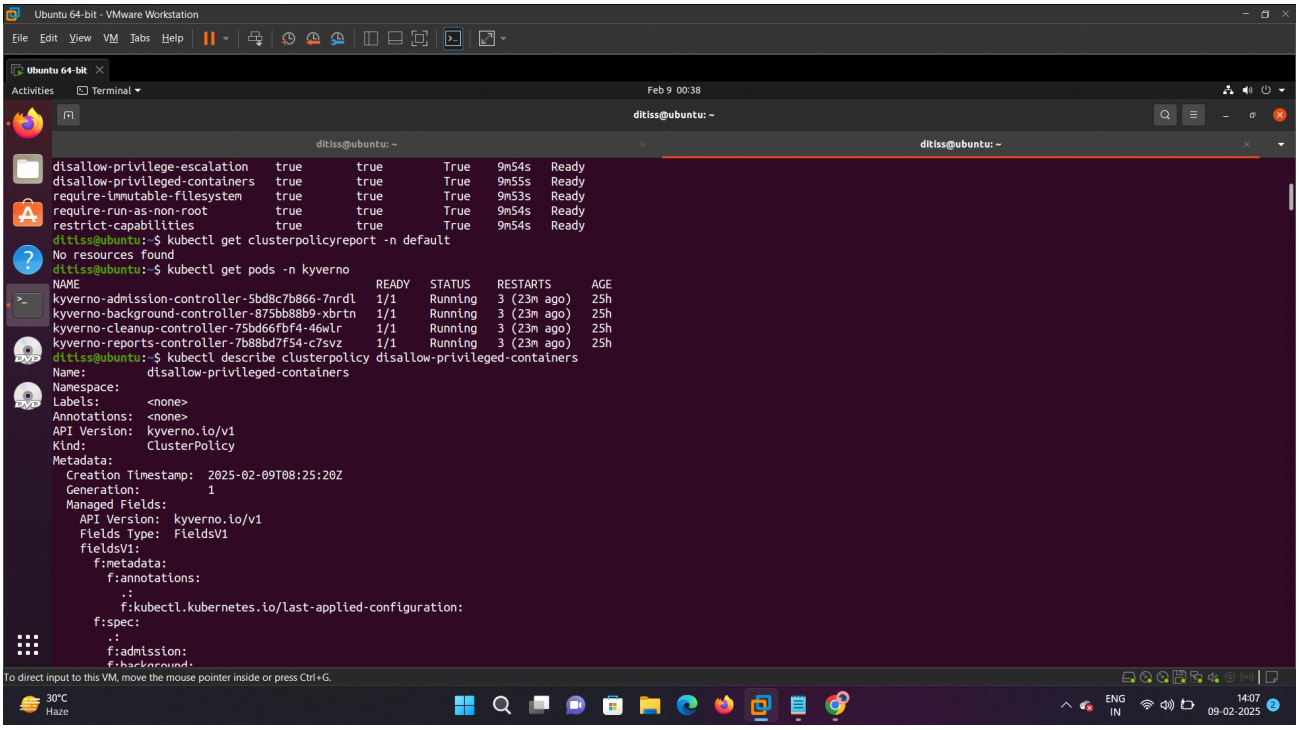


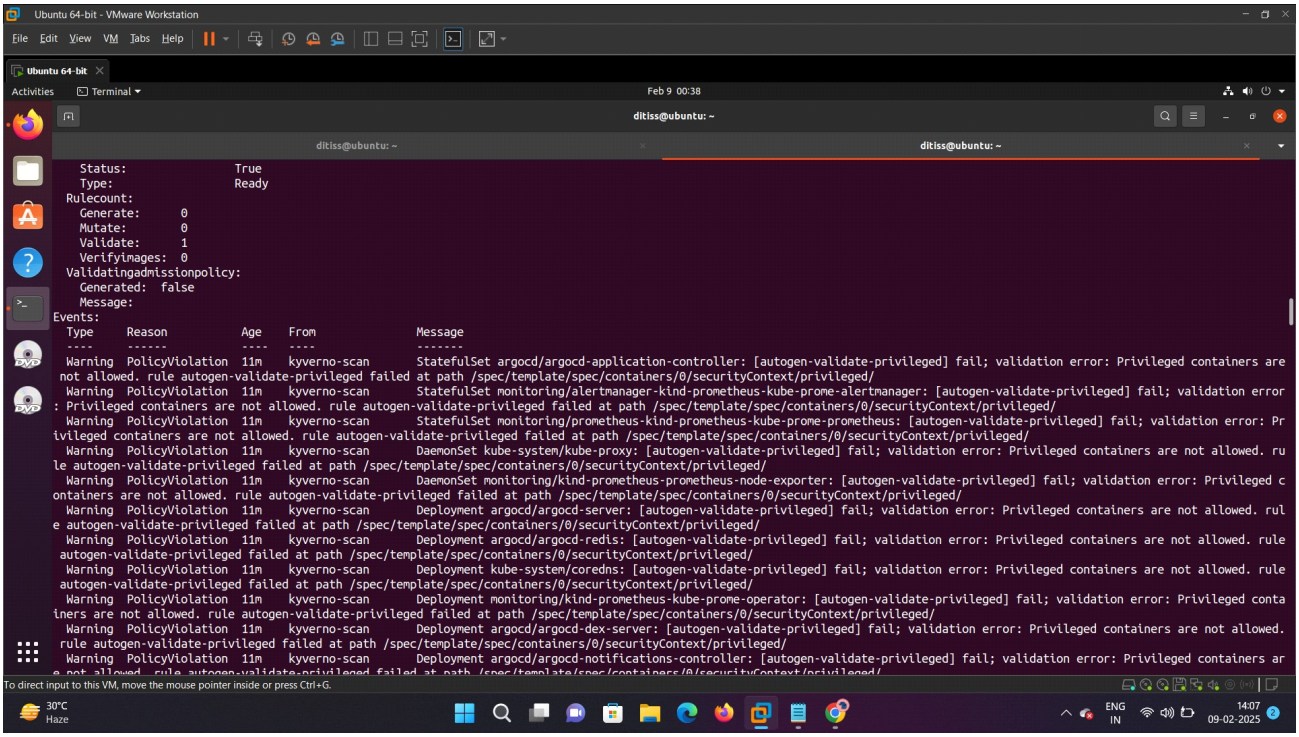
Kyverno installation via helm




Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help
Ubuntu 64-bit
Activities Terminal Feb 9 00:24
ditiss@ubuntu: ~
ditiss@ubuntu: ~
ditiss@ubuntu: ~
GNU nano 4.8 policies.yaml Modified
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
name: disallow-privileged-containers
spec:
validationFailureAction: Audit
rules:
- name: validate-privileged
match:
resources:
kinds:
- Pod
validate:
message: "Privileged containers are not allowed."
pattern:
spec:
containers:
- securityContext:
privileged: false
---
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
name: restrict-capabilities
spec:
validationFailureAction: Audit
rules:
- name: validate-capabilities
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo M-A Mark Text M-] To Bracket M-Q Previous ^B Back
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo M-6 Copy Text ^Q Where Was M-W Next ^F Forward
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
30°C Haze
ENG IN 13:54 09-02-2025


Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help
Ubuntu 64-bit
Activities Terminal Feb 9 00:25
ditiss@ubuntu: ~
ditiss@ubuntu: ~
ditiss@ubuntu: ~
ditiss@ubuntu:~$ sudo nano policies.yaml
[sudo] password for ditiss:
ditiss@ubuntu:~$ ls
config.yml Documents email kind.sh kubescape-pv.yaml Pictures Public reporting2.pdf Templates Videos
Desktop Downloads k8s-kind-voting-app kubectl.sh Music policies.yaml report sent test.pdf
ditiss@ubuntu:~$ kubectl apply -f policies.yaml
clusterpolicy.kyverno.io/disallow-privileged-containers created
clusterpolicy.kyverno.io/restrict-capabilities created
clusterpolicy.kyverno.io/disallow-host-network created
clusterpolicy.kyverno.io/disallow-host-pid-ipc created
clusterpolicy.kyverno.io/require-run-as-non-root created
clusterpolicy.kyverno.io/disallow-privilege-escalation created
clusterpolicy.kyverno.io/require-immutable-filesystem created
ditiss@ubuntu:~$ kubectl get clusterpolicies
NAME ADMISSION BACKGROUND READY AGE MESSAGE
disallow-host-network true true True 16s Ready
disallow-host-pid-ipc true true True 16s Ready
disallow-privilege-escalation true true True 16s Ready
disallow-privileged-containers true true True 17s Ready
require-immutable-filesystem true true True 15s Ready
require-run-as-non-root true true True 16s Ready
restrict-capabilities true true True 16s Ready
ditiss@ubuntu:~$
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
30°C Haze
ENG IN 13:55 09-02-2025


Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help
Ubuntu 64-bit
Activities Terminal Feb 9 00:38
ditiss@ubuntu: ~
disallow-privilege-escalation true true True 9m54s Ready
disallow-privileged-containers true true True 9m55s Ready
require-immutable-filesystem true true True 9m53s Ready
require-run-as-non-root true true True 9m54s Ready
restrict-capabilities true true True 9m54s Ready
ditiss@ubuntu:~$ kubectl get clusterpolicyreport -n default
No resources found
ditiss@ubuntu:~$ kubectl get pods -n kyverno
NAME READY STATUS RESTARTS AGE
kyverno-admission-controller-5bd8c7b866-7nrdl 1/1 Running 3 (23m ago) 25h
kyverno-background-controller-875bb88b9-xbrtn 1/1 Running 3 (23m ago) 25h
kyverno-cleanup-controller-75bd66fbf4-46wlr 1/1 Running 3 (23m ago) 25h
kyverno-reports-controller-7b88bd7f54-c7svz 1/1 Running 3 (23m ago) 25h
ditiss@ubuntu:~$ kubectl describe clusterpolicy disallow-privileged-containers
Name: disallow-privileged-containers
Namespace:
Labels: <none>
Annotations: <none>
API Version: kyverno.io/v1
Kind: ClusterPolicy
Metadata:
Creation Timestamp: 2025-02-09T08:25:20Z
Generation: 1
Managed Fields:
API Version: kyverno.io/v1
Fields Type: FieldsV1
fieldsV1:
f:metadata:
f:annotations:
.:
f:kubectl.kubernetes.io/last-applied-configuration:
f:spec:
.:
f:admission:
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
30°C Haze
ENG IN 14:07 09-02-2025


Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help
Ubuntu 64-bit
Activities Terminal Feb 9 00:38
ditiss@ubuntu: ~
Status: True
Type: Ready
Rulecount:
Generate: 0
Mutate: 0
Validate: 1
Verifyimages: 0
Validatingadmissionpolicy:
Generated: false
Message:
Events:
Type Reason Age From Message
Warning PolicyViolation 11m kyverno-scan StatefulSet argocd/argocd-application-controller: [autogen-validate-privileged] fail; validation error: Privileged containers are not allowed. rule autogen-validate-privileged failed at path /spec/template/spec/containers/0/securityContext/privileged/
Warning PolicyViolation 11m kyverno-scan StatefulSet monitoring/alertmanager-kind-prometheus-kube-prome-alertmanager: [autogen-validate-privileged] fail; validation error: Privileged containers are not allowed. rule autogen-validate-privileged failed at path /spec/template/spec/containers/0/securityContext/privileged/
Warning PolicyViolation 11m kyverno-scan StatefulSet monitoring/prometheus-kind-prometheus-kube-prome-prometheus: [autogen-validate-privileged] fail; validation error: Privileged containers are not allowed. rule autogen-validate-privileged failed at path /spec/template/spec/containers/0/securityContext/privileged/
Warning PolicyViolation 11m kyverno-scan DaemonSet kube-system/kube-proxy: [autogen-validate-privileged] fail; validation error: Privileged containers are not allowed. rule autogen-validate-privileged failed at path /spec/template/spec/containers/0/securityContext/privileged/
Warning PolicyViolation 11m kyverno-scan DaemonSet monitoring/kind-prometheus-prometheus-node-exporter: [autogen-validate-privileged] fail; validation error: Privileged containers are not allowed. rule autogen-validate-privileged failed at path /spec/template/spec/containers/0/securityContext/privileged/
Warning PolicyViolation 11m kyverno-scan Deployment argocd/argocd-server: [autogen-validate-privileged] fail; validation error: Privileged containers are not allowed. rule autogen-validate-privileged failed at path /spec/template/spec/containers/0/securityContext/privileged/
Warning PolicyViolation 11m kyverno-scan Deployment argocd/argocd-redis: [autogen-validate-privileged] fail; validation error: Privileged containers are not allowed. rule autogen-validate-privileged failed at path /spec/template/spec/containers/0/securityContext/privileged/
Warning PolicyViolation 11m kyverno-scan Deployment kube-system/coredns: [autogen-validate-privileged] fail; validation error: Privileged containers are not allowed. rule autogen-validate-privileged failed at path /spec/template/spec/containers/0/securityContext/privileged/
Warning PolicyViolation 11m kyverno-scan Deployment monitoring/kind-prometheus-kube-prome-operator: [autogen-validate-privileged] fail; validation error: Privileged containers are not allowed. rule autogen-validate-privileged failed at path /spec/template/spec/containers/0/securityContext/privileged/
Warning PolicyViolation 11m kyverno-scan Deployment argocd/argocd-dex-server: [autogen-validate-privileged] fail; validation error: Privileged containers are not allowed. rule autogen-validate-privileged failed at path /spec/template/spec/containers/0/securityContext/privileged/
Warning PolicyViolation 11m kyverno-scan Deployment argocd/argocd-notifications-controller: [autogen-validate-privileged] fail; validation error: Privileged containers are
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
30°C Haze
ENG IN 14:07 09-02-2025

Conclusion

This project successfully demonstrates the deployment, monitoring, and security enforcement of a microservices-based application within a Kubernetes cluster. By leveraging Kind for cluster setup, Docker for containerization, and Helm for package management, the application was efficiently deployed and managed. The integration of Prometheus and Grafana enabled real-time monitoring and visualization of system metrics, ensuring operational transparency.

Security was a key focus of this project, with Kyverno enforcing policy-based security compliance, Kube-Bench conducting CIS Kubernetes benchmark audits, and Kubescape identifying misconfigurations and vulnerabilities. Additionally, a cron-job based automation script was implemented to perform a periodic scan and send the scanned report to the mentioned email address, ArgoCD ensuring that the latest application version is pulled from Git and deployed in a secure and compliant manner.

This setup provides a scalable, secure, and automated environment for running microservices while maintaining compliance with industry security standards. Future enhancements could include extending the security policies, integrating additional automation tools, and exploring multi-cluster deployments to further optimize the infrastructure.

REFERENCES

1) https://kubernetes.io/docs/

2) https://prometheus.io/docs/introduction/overview/

3) https://grafana.com/docs

4) https://hub.armosec.io/docs

5) https://github.com/aquasecurity/kube-bench

6) https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository

7) https://github.com/LondheShubham153/k8s-kind-voting-app

8) https://argo-cd.readthedocs.io/en/stable/