

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)

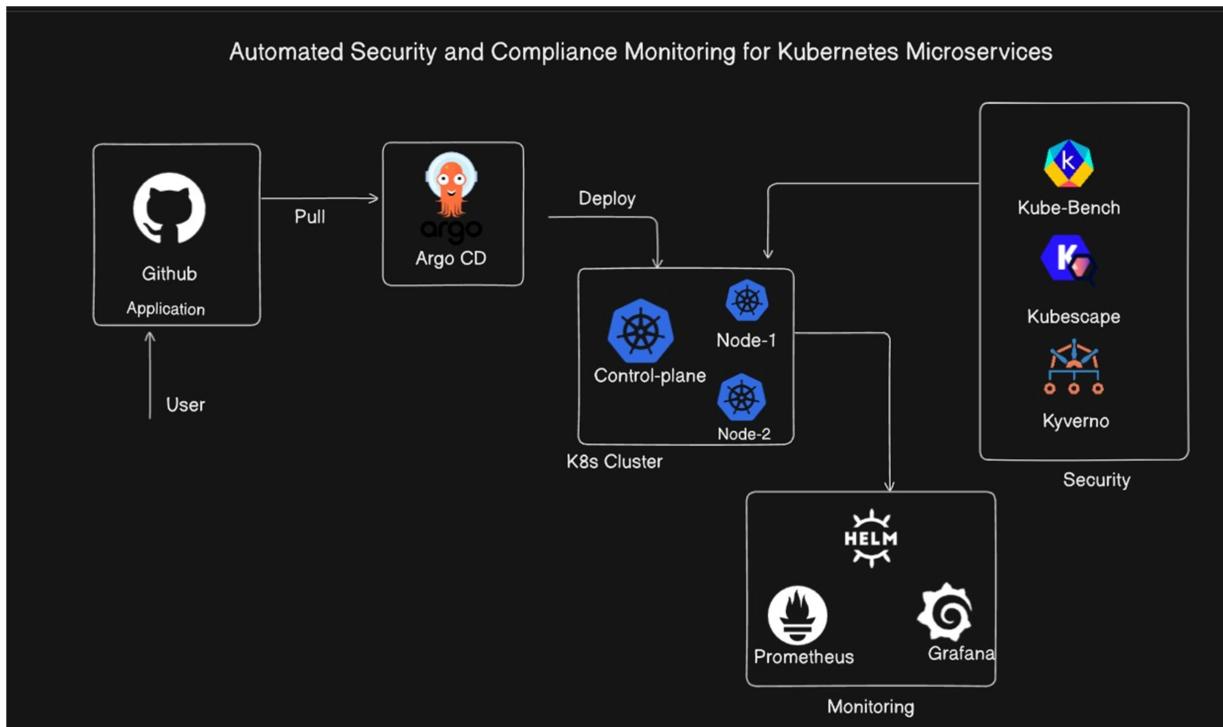
9090 (Prometheus)

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

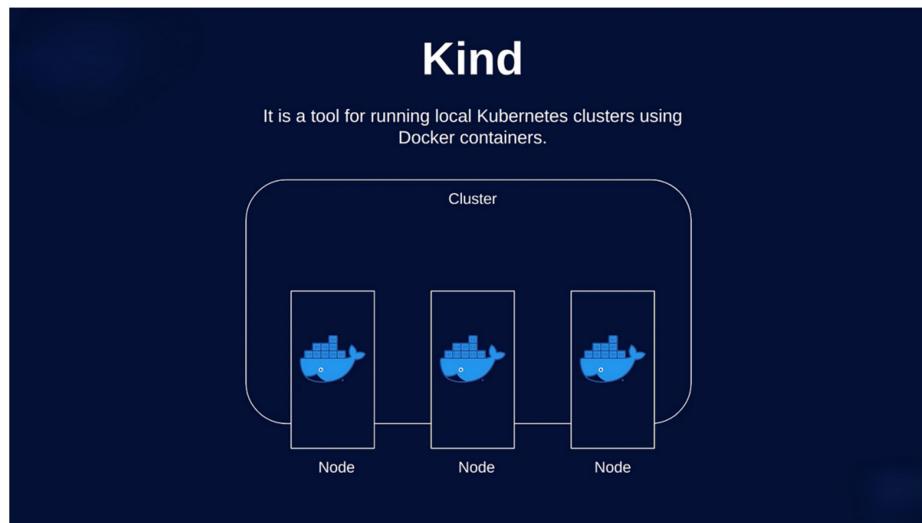


Docker plays a key role in containerizing the application and the security/monitoring tools deployed within Kubernetes. Instead of running applications as traditional processes, Docker packages them into lightweight, portable containers that can be easily deployed across different environments.

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.

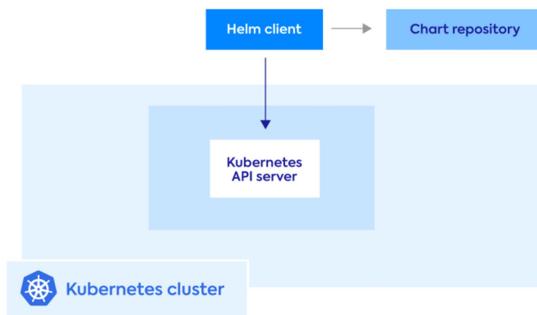
3. kind (Kubernetes-in-Docker)



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)



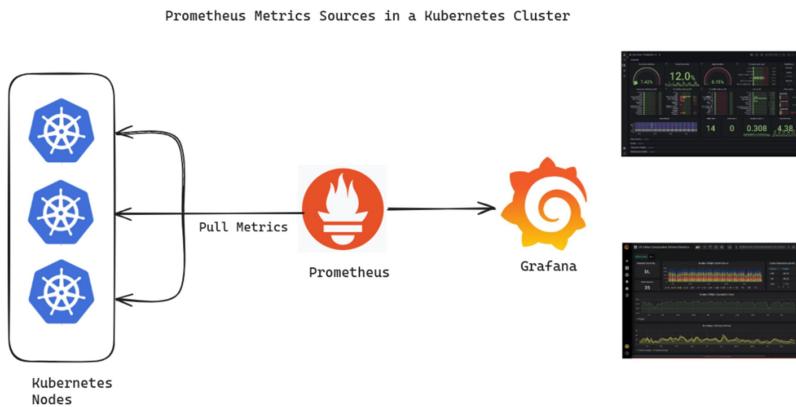
Helm is used in this project to simplify the installation and management of complex Kubernetes applications. Instead of manually configuring each component, Helm allows applications to be installed, upgraded, and version-controlled using Helm charts.

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)



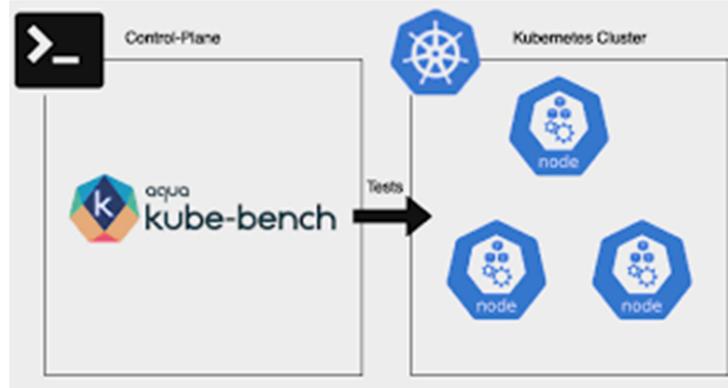
ArgoCD is a continuous deployment tool that ensures Kubernetes applications are always in sync with their defined configurations in a Git repository. Instead of manually applying changes, ArgoCD automates deployment updates whenever new changes are committed to Git.

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:

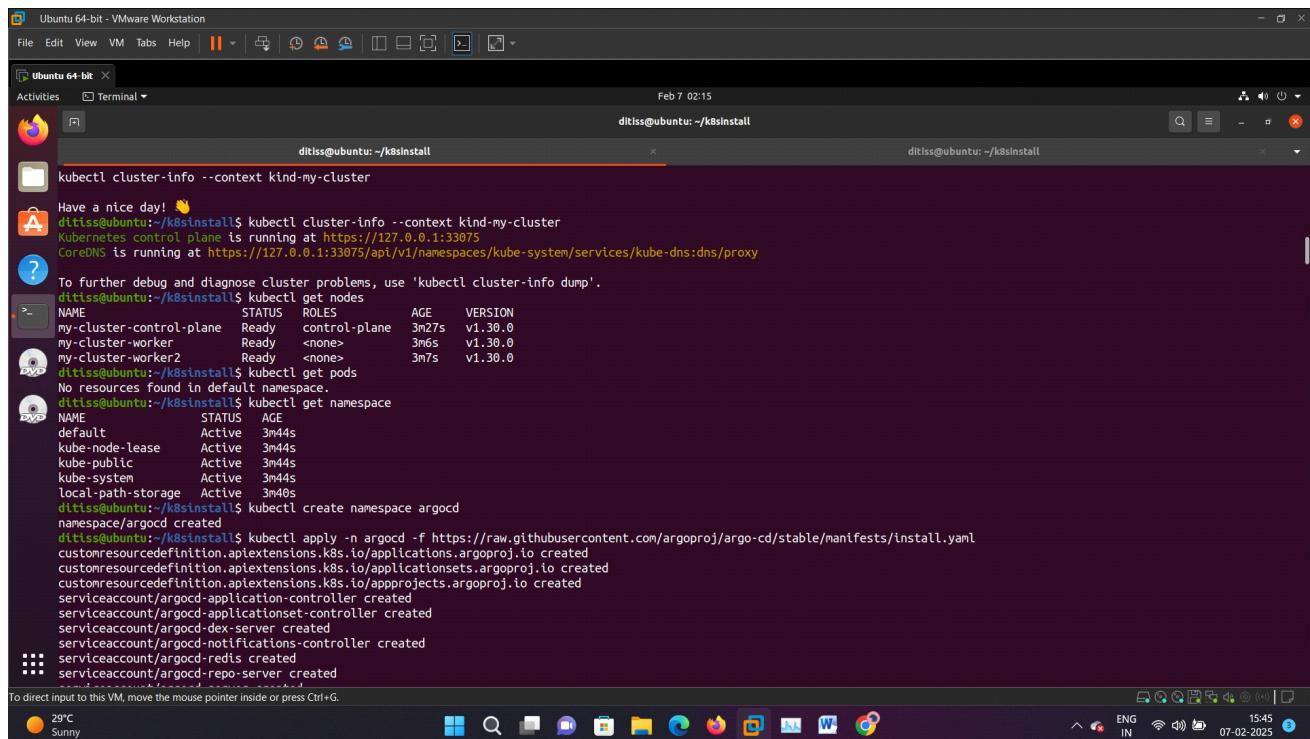
- Only secure and validated deployments are allowed.
- Specific security policies (e.g., restricting privileged access, enforcing network policies) are enforced.
- Any non-compliant workloads are automatically blocked.

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



```

Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help || Activities Terminal Feb 7 02:15 ditiss@ubuntu: ~/k8sinstall

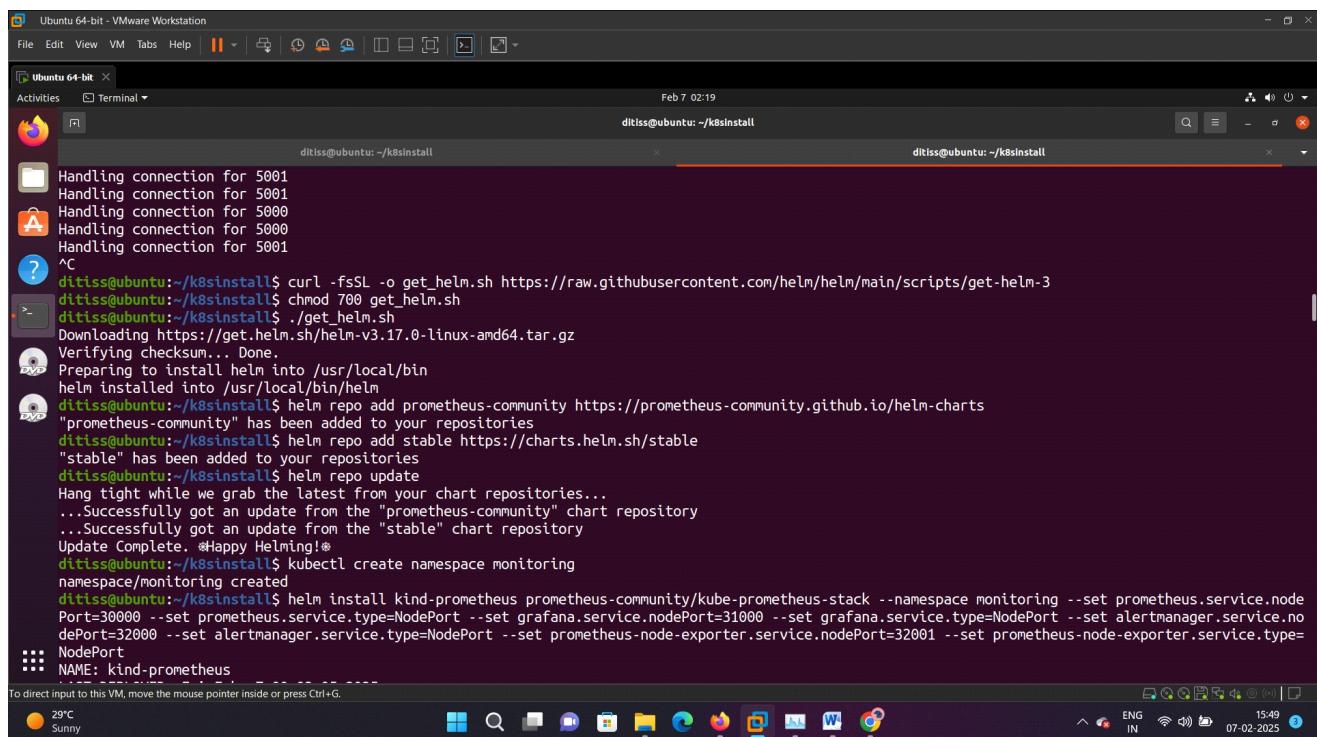
ditiss@ubuntu:~/k8sinstall$ kubectl cluster-info --context kind-my-cluster
Have a nice day! 🌻
Kubernetes control plane is running at https://127.0.0.1:3075
CoreDNS is running at https://127.0.0.1:3075/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
ditiss@ubuntu:~/k8sinstall$ kubectl get nodes
NAME STATUS ROLES AGE VERSION
my-cluster-control-plane Ready control-plane 3m27s v1.30.0
my-cluster-worker Ready <none> 3m6s v1.30.0
my-cluster-worker2 Ready <none> 3m7s v1.30.0
ditiss@ubuntu:~/k8sinstall$ kubectl get pods
No resources found in default namespace.
ditiss@ubuntu:~/k8sinstall$ kubectl get namespaces
NAME STATUS AGE
default Active 3m44s
kube-node-lease Active 3m44s
kube-public Active 3m44s
kube-system Active 3m44s
local-path-storage Active 3m45s
ditiss@ubuntu:~/k8sinstall$ kubectl create namespace argocd
namespace/argocd created
ditiss@ubuntu:~/k8sinstall$ kubectl apply -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml
customresourcedefinition.apiextensions.k8s.io/applicationsets.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/applicationsets.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/applications.argoproj.io created
serviceaccount/argocd-application-controller created
serviceaccount/argocd-application-set-controller created
serviceaccount/argocd-dex-server created
serviceaccount/argocd-notifications-controller created
serviceaccount/argocd-redis created
serviceaccount/argocd-repo-server created
ditiss@ubuntu:~/k8sinstall$ ^C
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

29°C Sunny ENG IN 15:49 07-02-2025

```

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



```

Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help || Activities Terminal Feb 7 02:19 ditiss@ubuntu: ~/k8sinstall

ditiss@ubuntu:~/k8sinstall$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
ditiss@ubuntu:~/k8sinstall$ chmod 700 get_helm.sh
ditiss@ubuntu:~/k8sinstall$ ./get_helm.sh
Downloading https://get.helm.sh/helm-v3.17.0-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
ditiss@ubuntu:~/k8sinstall$ helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
"prometheus-community" has been added to your repositories
ditiss@ubuntu:~/k8sinstall$ helm repo add stable https://charts.helm.sh/stable
"stable" 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 "prometheus-community" chart repository
...Successfully got an update from the "stable" chart repository
Update Complete. *Happy Helm-ing!*
ditiss@ubuntu:~/k8sinstall$ kubectl create namespace monitoring
namespace/monitoring created
ditiss@ubuntu:~/k8sinstall$ helm install kind-prometheus prometheus/prometheus-community/kube-prometheus-stack --namespace monitoring --set prometheus.service.nodePort=30000 --set prometheus.service.type=NodePort --set grafana.service.nodePort=31000 --set grafana.service.type=NodePort --set alertmanager.service.nodePort=32000 --set alertmanager.service.type=NodePort --set prometheus-node-exporter.service.nodePort=32001 --set prometheus-node-exporter.service.type=NodePort
ditiss@ubuntu:~/k8sinstall$ ^C
NAME: kind-prometheus
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

29°C Sunny ENG IN 15:49 07-02-2025

```

Enable port forwarding for Prometheus=9090 and Grafana=31000

Ubuntu 64-bit - VMware Workstation

File Edit View VM Tabs Help || Activities Terminal

Ubuntu 64-bit

Activities Terminal

ditiss@ubuntu: ~/k8sinstall

Feb 7 02:19

ditiss@ubuntu: ~/k8sinstall

kind-prometheus-node-exporter-t8b29 1/1 Running 0 107s
kind-prometheus-node-exporter-xk6g8 1/1 Running 0 107s
kind-prometheus-node-exporter-z7gdp 1/1 Running 0 107s
prometheus-kind-prometheus-kube-prime-prometheus-0 0/2 PodInitializing 0 88s

ditiss@ubuntu:~/k8sinstall\$ kubectl get pods -n monitoring

NAME	READY	STATUS	RESTARTS	AGE
alertmanager-kind-prometheus-kube-prime-alertmanager-0	2/2	Running	0	2m25s
kind-prometheus-grafana-9bf6d8569-fjv6c	3/3	Running	0	2m44s
kind-prometheus-kube-prime-operator-b9d8fccf6-5phqt	1/1	Running	0	2m44s
kind-prometheus-kube-state-metrics-5db69c8d6-c9ft8	1/1	Running	0	2m44s
kind-prometheus-node-exporter-t8b29	1/1	Running	0	2m44s
kind-prometheus-node-exporter-xk6g8	1/1	Running	0	2m44s
kind-prometheus-node-exporter-z7gdp	1/1	Running	0	2m44s
prometheus-kind-prometheus-kube-prime-prometheus-0	2/2	Running	0	2m25s

ditiss@ubuntu:~/k8sinstall\$ kubectl port-forward svc/kind-prometheus-kube-prime-prometheus -n monitoring 9090:9090 --address=0.0.0.0 & [3] 15601

ditiss@ubuntu:~/k8sinstall\$ kubectl port-forward svc/kind-prometheus-grafana -n monitoring 31000:80 --address=0.0.0.0 &Forwarding from 0.0.0.0:9090 -> 9090

Handling connection for 9090
Handling connection for 9090
Handling connection for 9090
^C

ditiss@ubuntu:~/k8sinstall\$ Handling connection for 9090
Handling connection for 5000
Handling connection for 9090

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

29°C Sunny ENG IN 15:49 07-02-2025

Ubuntu 64-bit - VMware Workstation

```
ditiss@ubuntu: ~/k8sinstall
Activities Terminal Feb 7 02:19
ditiss@ubuntu: ~/k8sinstall
ditiss@ubuntu: ~/k8sinstall
```

```
ditiss@ubuntu:~/k8sinstall$ kubectl port-forward svc/kind-prometheus-kube-prome-prometheus -n monitoring 9090:9090 --address=0.0.0.0 &
[3] 15235
ditiss@ubuntu:~/k8sinstall$ kubectl port-forward svc/kind-prometheus-grafana -n monitoring 31000:80 --address=0.0.0.0 &error: unable to forward port because pod is not running. Current status=Pending
^C
[3]+ Exit 1 kubectl port-forward svc/kind-prometheus-kube-prome-prometheus -n monitoring 9090:9090 --address=0.0.0.0
ditiss@ubuntu:~/k8sinstall$ kubectl get pods
NAME READY STATUS RESTARTS AGE
db-597b4ff8d7-8tdt7 1/1 Running 0 11m
redis-796dc594bb-bgjxp 1/1 Running 0 11m
result-d8c4c69b8-vc8mn 1/1 Running 0 11m
vote-69cb46f6fb-6nr97 1/1 Running 0 11m
worker-5dd767667f-6ksc4 1/1 Running 0 11m
ditiss@ubuntu:~/k8sinstall$ kubectl get pods -n monitoring
NAME READY STATUS RESTARTS AGE
alertmanager-kind-prometheus-kube-prome-alertmanager-0 2/2 Running 0 88s
kind-prometheus-grafana-9bf6d8569-fjv6c 3/3 Running 0 107s
kind-prometheus-kube-prome-operator-b9d8fccf6-5phqt 1/1 Running 0 107s
kind-prometheus-kube-state-metrics-5db69c8d6-c9ft8 1/1 Running 0 107s
kind-prometheus-prometheus-node-exporter-t8b29 1/1 Running 0 107s
kind-prometheus-prometheus-node-exporter-xk6g8 1/1 Running 0 107s
kind-prometheus-prometheus-node-exporter-z7gdp 1/1 Running 0 107s
prometheus-kind-prometheus-kube-prome-prometheus-0 0/2 PodInitializing 0 88s
ditiss@ubuntu:~/k8sinstall$ kubectl get pods -n monitoring
```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

29°C Sunny ENG IN 15:49 07-02-2025

ArgoCD port forwarding=8443

Ubuntu 64-bit - VMware Workstation

```
ditiss@ubuntu: ~/k8sinstall
Activities Terminal Feb 7 02:15
ditiss@ubuntu: ~/k8sinstall
ditiss@ubuntu: ~/k8sinstall
```

```
ditiss@ubuntu:~/k8sinstall$ argocd app create argocd
statefulset.apps/argocd-application-controller created
networkpolicy.networking.k8s.io/argocd-application-controller-network-policy created
networkpolicy.networking.k8s.io/argocd-applicationset-controller-network-policy created
networkpolicy.networking.k8s.io/argocd-dex-server-network-policy created
networkpolicy.networking.k8s.io/argocd-notifications-controller-network-policy created
networkpolicy.networking.k8s.io/argocd-redis-network-policy created
networkpolicy.networking.k8s.io/argocd-repo-server-network-policy created
networkpolicy.networking.k8s.io/argocd-server-network-policy created
ditiss@ubuntu:~/k8sinstall$ kubectl get svc -n argocd
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
argocd-applicationset-controller ClusterIP 10.96.108.161 <none> 7000/TCP,8080/TCP 12s
argocd-dex-server ClusterIP 10.96.36.77 <none> 5556/TCP,5557/TCP,5558/TCP 12s
argocd-metrics ClusterIP 10.96.152.32 <none> 8082/TCP 12s
argocd-notifications-controller-metrics ClusterIP 10.96.137.85 <none> 9001/TCP 12s
argocd-redis ClusterIP 10.96.134.200 <none> 6379/TCP 12s
argocd-repo-server ClusterIP 10.96.125.202 <none> 8081/TCP,8084/TCP 12s
argocd-server ClusterIP 10.96.70.244 <none> 88/TCP,443/TCP 12s
argocd-server-metrics ClusterIP 10.96.91.22 <none> 8083/TCP 11s
ditiss@ubuntu:~/k8sinstall$ kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "NodePort"}}'
service/argocd-server patched
ditiss@ubuntu:~/k8sinstall$ kubectl get svc -n argocd
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
argocd-applicationset-controller ClusterIP 10.96.108.161 <none> 7000/TCP,8080/TCP 44s
argocd-dex-server ClusterIP 10.96.36.77 <none> 5556/TCP,5557/TCP,5558/TCP 44s
argocd-metrics ClusterIP 10.96.152.32 <none> 8082/TCP 44s
argocd-notifications-controller-metrics ClusterIP 10.96.137.85 <none> 9001/TCP 44s
argocd-redis ClusterIP 10.96.134.200 <none> 6379/TCP 44s
argocd-repo-server ClusterIP 10.96.125.202 <none> 8081/TCP,8084/TCP 44s
argocd-server NodePort 10.96.70.244 <none> 88-32231/TCP,443-32269/TCP 44s
argocd-server-metrics ClusterIP 10.96.91.22 <none> 8083/TCP 43s
ditiss@ubuntu:~/k8sinstall$ kubectl port-forward -n argocd service/argocd-server 8443:443 --address=0.0.0.0 &
[1] 11756
ditiss@ubuntu:~/k8sinstall$ Forwarding from 0.0.0.0:8443 -> 8080
Handling connection for 8443
```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

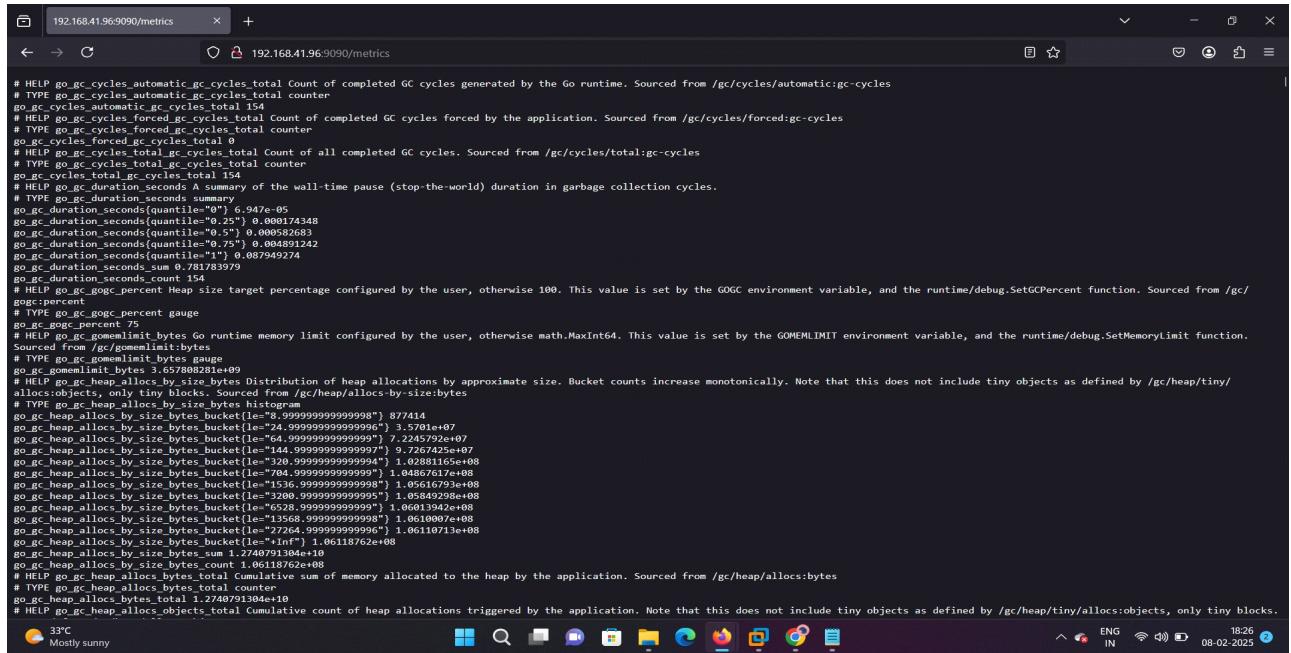
29°C Sunny ENG IN 15:45 07-02-2025

Credentials for ArgoCD login

Voting application deployed

A screenshot of the Argo UI interface on an Ubuntu 64-bit VM. The left sidebar shows the Argo application dashboard with sections for Applications, Settings, User Info, Documentation, and Resource filters. The main area displays the sync status of a 'votingapp' deployment. The 'SYNC STATUS' section shows 'Synced' to 'main (a2bee8)' with an auto-sync enabled status. The 'LAST SYNC' section shows a successful sync to 'a2bee8' at 04:45:49 on Feb 8, 2024. A detailed timeline diagram illustrates the sync process across various Kubernetes components like db, redis, result, vote, and worker deployments.

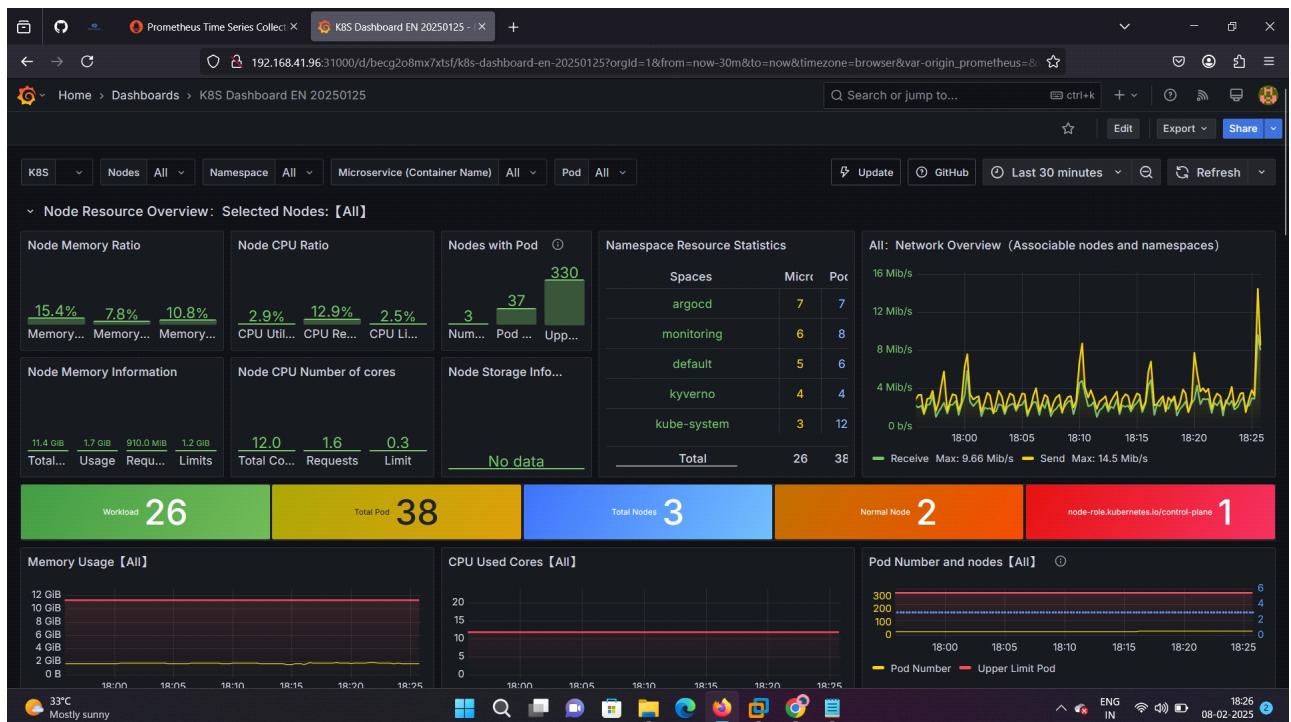
Prometheus metrics of an deployment

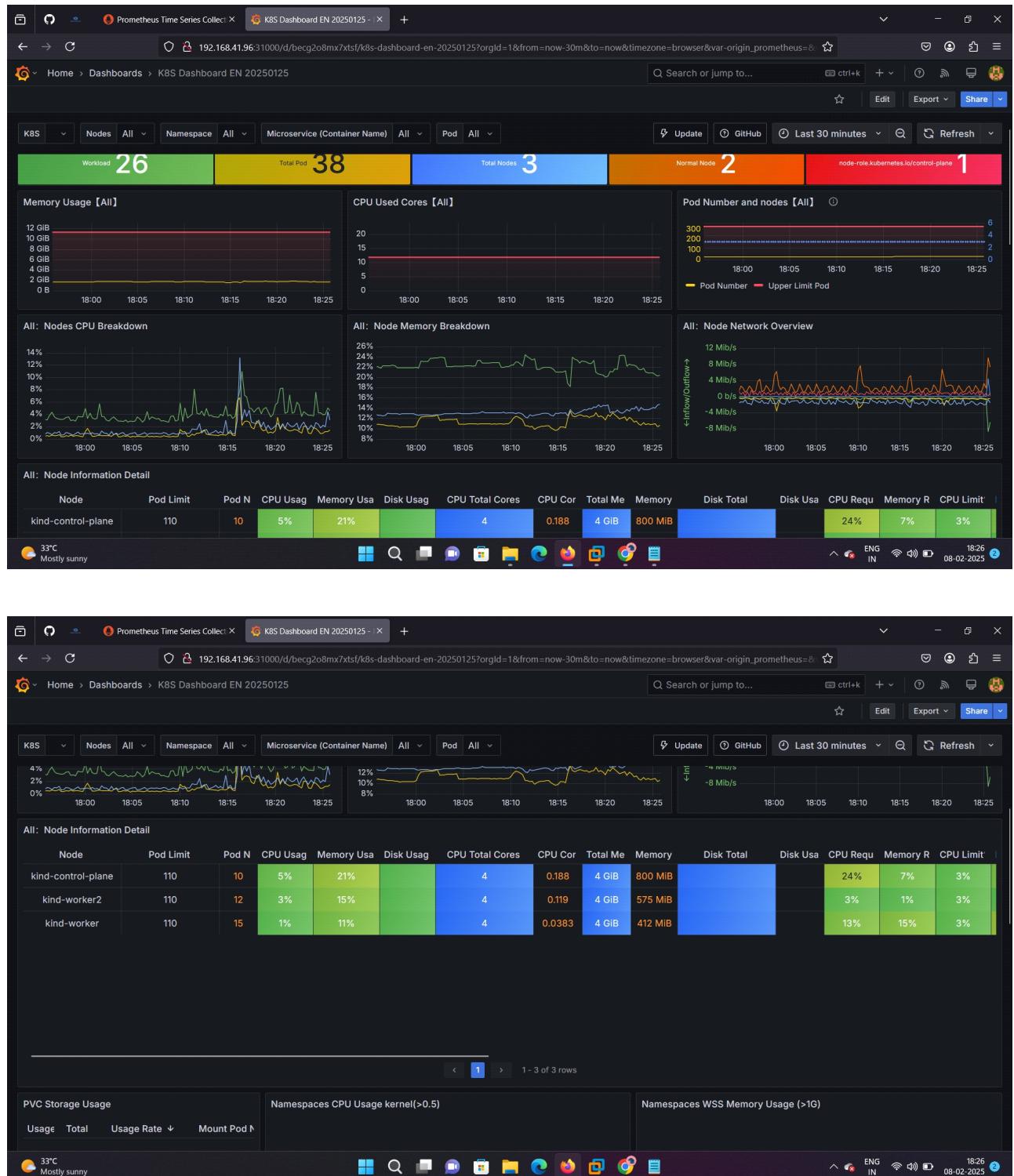


```
# HELP go_gc_cycles_automatic_gc_cycles_total Count of completed GC cycles generated by the Go runtime. Sourced from /gc/cycles/automatic:gc-cycles
# TYPE go_gc_cycles_automatic_gc_cycles_total 154
go_gc_cycles_automatic_gc_cycles_total 154
# HELP go_gc_cycles_forced_gc_cycles_total Count of completed GC cycles forced by the application. Sourced from /gc/cycles/forced:gc-cycles
# TYPE go_gc_cycles_forced_gc_cycles_total 1
go_gc_cycles_forced_gc_cycles_total 1
# HELP go_gc_cycles_total_gc_cycles_total Count of all completed GC cycles. Sourced from /gc/cycles/total:gc-cycles
# TYPE go_gc_cycles_total_gc_cycles_total 154
go_gc_cycles_total_gc_cycles_total 154
# HELP go_gc_duration_seconds A summary of the wall-time pause (stop-the-world) duration in garbage collection cycles.
# TYPE go_gc_duration_seconds 0.947e-05
go_gc_duration_seconds{quantile="0.25"} 0.000174348
go_gc_duration_seconds{quantile="0.5"} 0.000582683
go_gc_duration_seconds{quantile="0.75"} 0.004891242
go_gc_duration_seconds{quantile="1"} 0.087949274
go_gc_duration_seconds{sum="0.000174348",count="154"} 0.000174348
go_gc_duration_seconds_count 154
# HELP go_gc_gogo_percent Heap size target percentage configured by the user, otherwise 100. This value is set by the GOGC environment variable, and the runtime/debug.SetGCPercent function. Sourced from /gc/gogo:percent
# TYPE go_gc_gogo_percent gauge
go_gc_gogo_percent 100
# HELP go_gc_gomemlimit_bytes Go runtime memory limit configured by the user, otherwise math.MaxInt64. This value is set by the GOMEMLIMIT environment variable, and the runtime/debug.SetMemoryLimit function. Sourced from /gc/gomemlimit:bytes
# TYPE go_gc_gomemlimit_bytes gauge
go_gc_gomemlimit_bytes 3.657880281e+09
# HELP go_gc_heap_allocs_by_size_bytes Distribution of heap allocations by approximate size. Bucket counts increase monotonically. Note that this does not include tiny objects as defined by /gc/heap/tiny/alllocs:bytes
# TYPE go_gc_heap_allocs_by_size_bytes histogram
go_gc_heap_allocs_by_size_bytes_bucket{le="8.99999999999998"} 877414
go_gc_heap_allocs_by_size_bytes_bucket{le="24.99999999999996"} 3.5701e+07
go_gc_heap_allocs_by_size_bytes_bucket{le="72.99999999999994"} 7.2245792e+07
go_gc_heap_allocs_by_size_bytes_bucket{le="144.99999999999992"} 1.0601396e+08
go_gc_heap_allocs_by_size_bytes_bucket{le="216.9999999999999"} 1.02881165e+08
go_gc_heap_allocs_by_size_bytes_bucket{le="288.9999999999997"} 1.04867617e+08
go_gc_heap_allocs_by_size_bytes_bucket{le="360.9999999999995"} 1.05616793e+08
go_gc_heap_allocs_by_size_bytes_bucket{le="432.9999999999993"} 1.05849298e+08
go_gc_heap_allocs_by_size_bytes_bucket{le="504.9999999999991"} 1.0601396e+08
go_gc_heap_allocs_by_size_bytes_bucket{le="576.9999999999989"} 1.0601396e+08
go_gc_heap_allocs_by_size_bytes_bucket{le="648.9999999999987"} 1.0601396e+08
go_gc_heap_allocs_by_size_bytes_bucket{le="720.9999999999985"} 1.0601396e+08
go_gc_heap_allocs_by_size_bytes_bucket{le="792.9999999999983"} 1.0601396e+08
go_gc_heap_allocs_by_size_bytes_bucket{le="864.9999999999981"} 1.0601396e+08
go_gc_heap_allocs_by_size_bytes_bucket{le="936.9999999999979"} 1.0601396e+08
go_gc_heap_allocs_by_size_bytes_bucket{le="1008.9999999999977"} 1.0601396e+08
go_gc_heap_allocs_by_size_bytes_bucket{le="1080.9999999999975"} 1.0601396e+08
go_gc_heap_allocs_by_size_bytes_bucket{le="1152.9999999999973"} 1.0601396e+08
go_gc_heap_allocs_by_size_bytes_bucket{le="1224.9999999999971"} 1.0601396e+08
go_gc_heap_allocs_by_size_bytes_bucket{le="1296.9999999999969"} 1.0601396e+08
go_gc_heap_allocs_by_size_bytes_sum 1.2740791304e+10
go_gc_heap_allocs_by_size_bytes_total 1.2740791304e+10
# HELP go_gc_heap_allocs_bytes_total Cumulative sum of memory allocated to the heap by the application. Sourced from /gc/heap/allocs:bytes
# TYPE go_gc_heap_allocs_bytes_total counter
go_gc_heap_allocs_bytes_total 1.2740791304e+10
# HELP go_gc_heap_allocs_objects_total Cumulative count of heap allocations triggered by the application. Note that this does not include tiny objects as defined by /gc/heap/tiny/alllocs:objects, only tiny blocks.
# TYPE go_gc_heap_allocs_objects_total counter
go_gc_heap_allocs_objects_total 1.2740791304e+10

```

Metrics visualization via Grafana





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... Q

Nodes All Namespace All Microservice (Container Name) All Pod All

Last 30 minutes

Update GitHub Last 30 minutes Refresh Share

Namespace	Cont	Pod Name	CPU%	WSS%	RSS%	Memory	WSS	RSS	Disk Limit	Disk Use	Rebo	Survival	Memory R	Inflow
default	kube-b	kube-bench-qgn9w									0	6.03 hours		
default	worker	worker-5dd767667f-dmqlh					21.6 MiB	9.39 MiB			0	10.1 mins	16.3 KiB/s	
default	vote	vote-69cb46f6fb-7kfz					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	

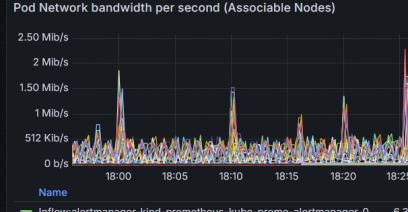
Pod Containers CPU Utilization (Maximum 100% Associateable Nodes)

No data

Pod Container Memory Usage (Associateable Nodes)

No data

Pod Network bandwidth per second (Associateable Nodes)



33°C Mostly sunny

ENG IN 08-02-2025

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... Q

Nodes All Namespace All Microservice (Container Name) All Pod All

Last 30 minutes

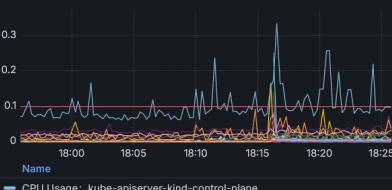
Update GitHub Last 30 minutes Refresh Share

Name	Value
Inflow:alertmanager-kind-prometheus-kube-prime-alertmanager-0	6.35
Inflow:argocd-application-controller-0	728

No data

No data

Pod Containers CPU Core Usage



Microservices (Container Name) Resource Overview: Selected Microservices: [All] (7 panels)

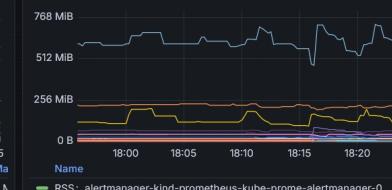
33°C Mostly sunny

ENG IN 08-02-2025

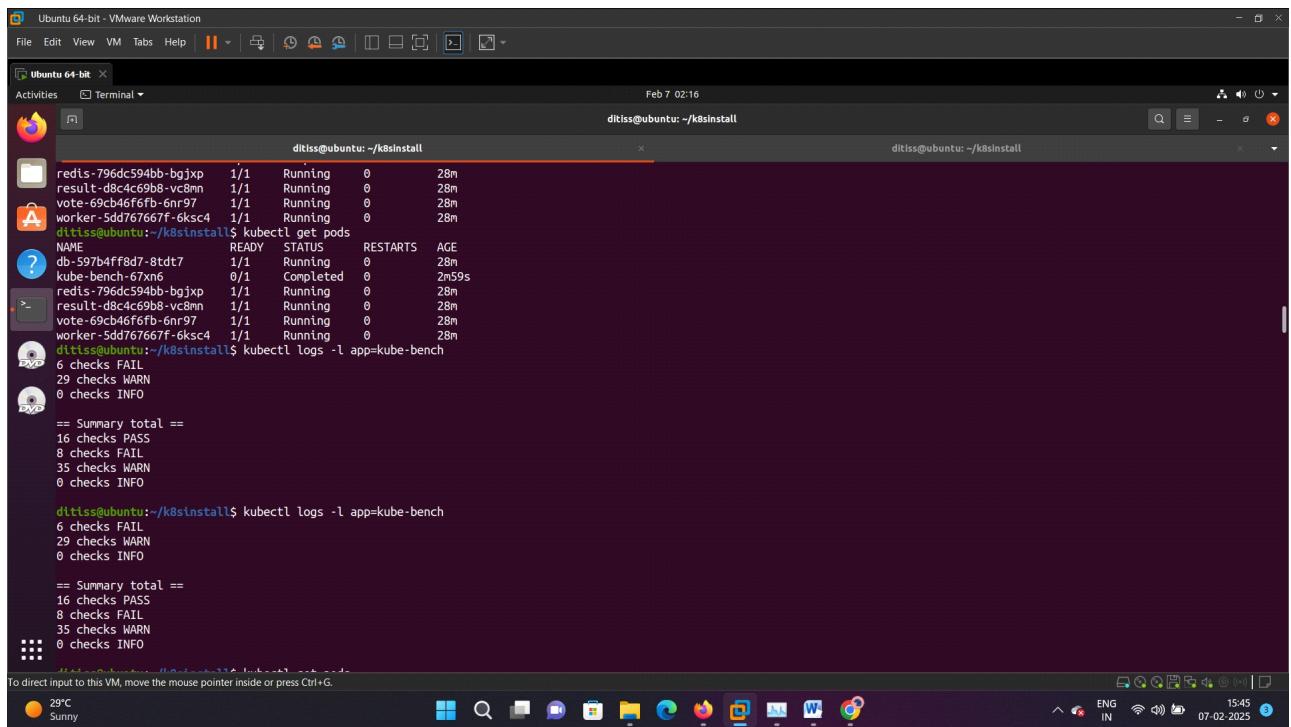
Pod Containers WSS Memory Usage (Associateable Nodes)



Pod Containers RSS Memory Usage (Associateable Nodes)



Kube-bench scan report

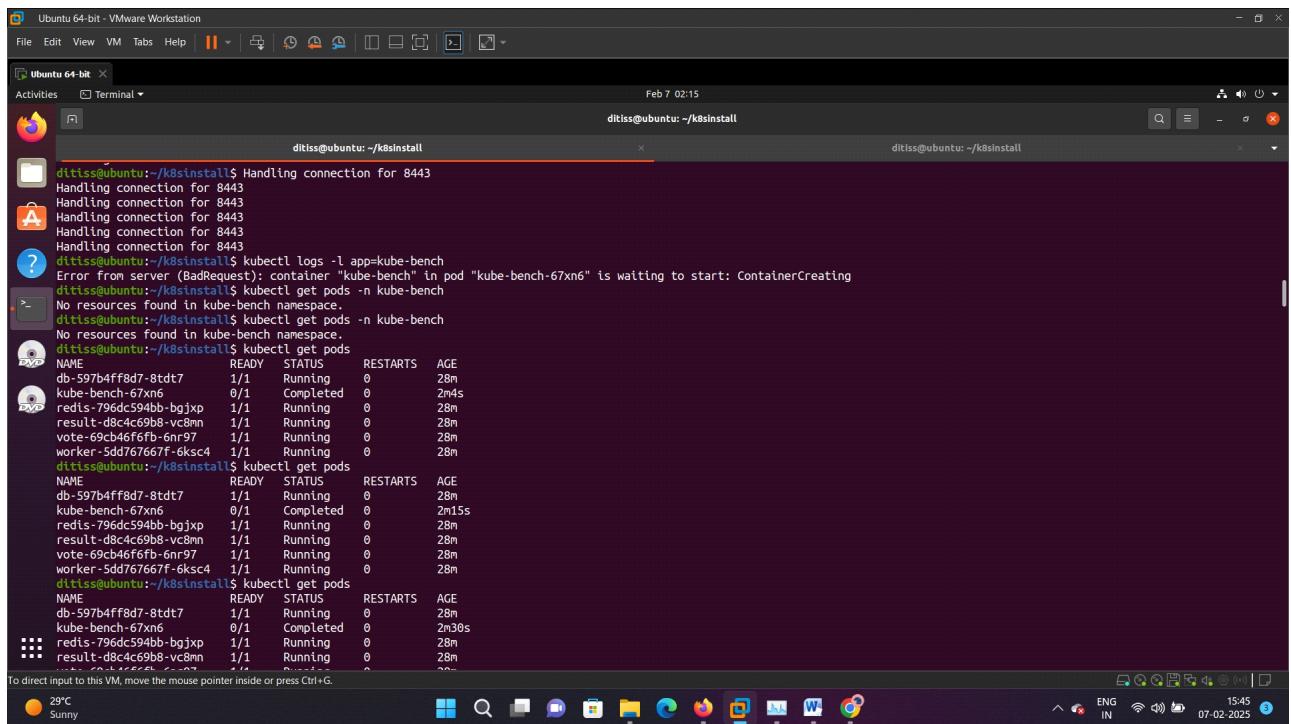


```

ditiss@ubuntu:~/k8sinstall$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
redis-796dc594bb-bgjxp  1/1     Running   0          28m
result-d9c4cc69b8-vc8mn  1/1     Running   0          28m
vote-69c946f6fb-6nr97   1/1     Running   0          28m
worker-5dd767667f-6ksc4  1/1     Running   0          28m
ditiss@ubuntu:~/k8sinstall$ kubectl logs -l app=kube-bench
6 checks FAIL
29 checks WARN
0 checks INFO
== Summary total ==
16 checks PASS
8 checks FAIL
35 checks WARN
0 checks INFO

ditiss@ubuntu:~/k8sinstall$ kubectl logs -l app=kube-bench
6 checks FAIL
29 checks WARN
0 checks INFO
== Summary total ==
16 checks PASS
8 checks FAIL
35 checks WARN
0 checks INFO

```

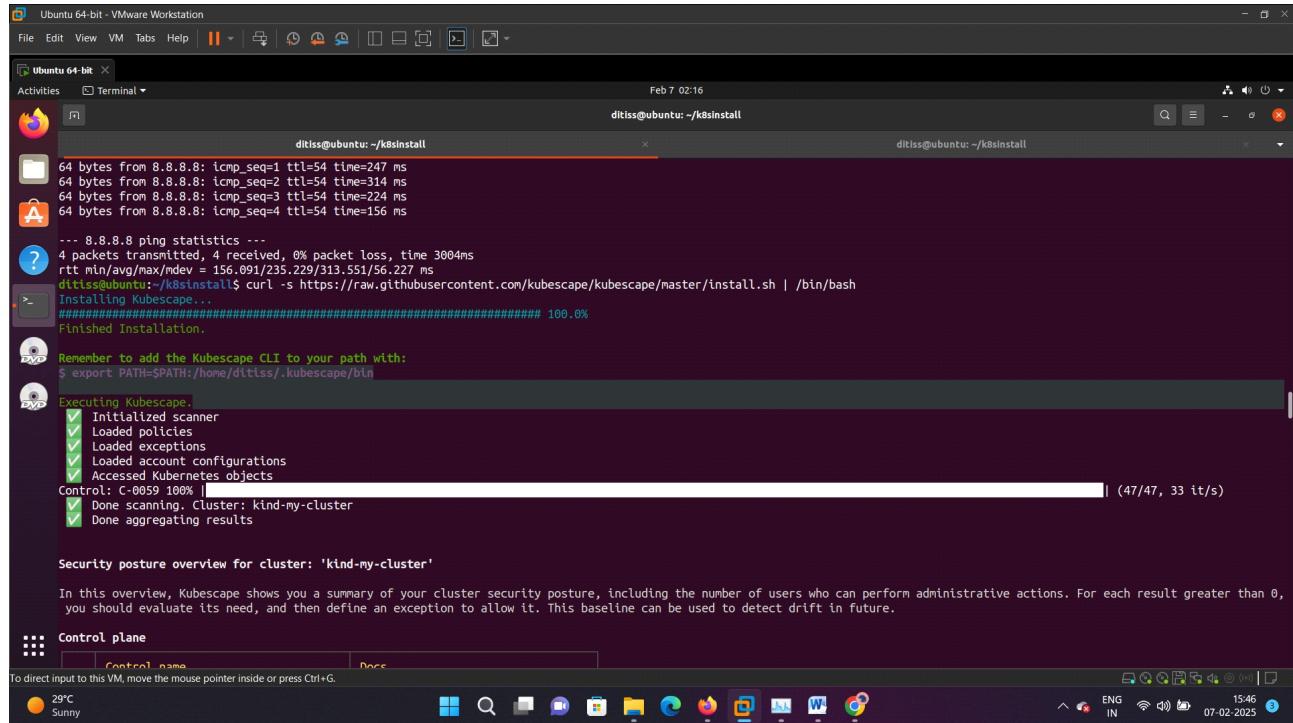


```

ditiss@ubuntu:~/k8sinstall$ Handling connection for 8443
ditiss@ubuntu:~/k8sinstall$ kubectl logs -l app=kube-bench
Error from server (BadRequest): container "kube-bench" in pod "kube-bench-67xn6" is waiting to start: ContainerCreating
ditiss@ubuntu:~/k8sinstall$ kubectl get pods -n kube-bench
No resources found in kube-bench namespace.
ditiss@ubuntu:~/k8sinstall$ kubectl get pods -n kube-bench
No resources found in kube-bench namespace.
ditiss@ubuntu:~/k8sinstall$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
db-597b4ff8d7-8tdt7  1/1     Running   0          28m
kube-bench-67xn6    0/1     Completed  0          2m4s
redis-796dc594bb-bgjxp  1/1     Running   0          28m
result-d9c4cc69b8-vc8mn  1/1     Running   0          28m
vote-69c946f6fb-6nr97  1/1     Running   0          28m
worker-5dd767667f-6ksc4  1/1     Running   0          28m
ditiss@ubuntu:~/k8sinstall$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
db-597b4ff8d7-8tdt7  1/1     Running   0          28m
kube-bench-67xn6    0/1     Completed  0          2m15s
redis-796dc594bb-bgjxp  1/1     Running   0          28m
result-d9c4cc69b8-vc8mn  1/1     Running   0          28m
vote-69c946f6fb-6nr97  1/1     Running   0          28m
worker-5dd767667f-6ksc4  1/1     Running   0          28m
ditiss@ubuntu:~/k8sinstall$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
db-597b4ff8d7-8tdt7  1/1     Running   0          28m
kube-bench-67xn6    0/1     Completed  0          2m30s
redis-796dc594bb-bgjxp  1/1     Running   0          28m
result-d9c4cc69b8-vc8mn  1/1     Running   0          28m
vote-69c946f6fb-6nr97  1/1     Running   0          28m
worker-5dd767667f-6ksc4  1/1     Running   0          28m

```

Kubescape installation and report of kubescape scan



Ubuntu 64-bit - VMware Workstation

File Edit View VM Tabs Help Feb 7 02:16 ditliss@ubuntu: ~/k8sinstall

Activities Terminal ditliss@ubuntu: ~/k8sinstall

```
ditliss@ubuntu: ~/k8sinstall
64 bytes from 8.8.8.8: icmp_seq=1 ttl=54 time=247 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=54 time=314 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=54 time=224 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=54 time=156 ms
...
8.8.8.8 ping statistics ...
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 156.091/235.229/313.551/56.227 ms
ditliss@ubuntu:~/k8sinstall$ curl -s https://raw.githubusercontent.com/kubescape/kubescape/master/install.sh | /bin/bash
Installing Kubescape...
#####
Finished Installation.

Remember to add the Kubescape CLI to your path with:
$ export PATH=$PATH:/home/ditliss/.kubescape/bin

Executing Kubescape.
[✓] Initialized scanner
[✓] Loaded policies
[✓] Loaded exceptions
[✓] Loaded account configurations
[✓] Accessed Kubernetes objects
Control: C-0059 100% [██████████] (47/47, 33 it/s)
[✓] Done scanning. Cluster: kind-my-cluster
[✓] Done aggregating results

Security posture overview for cluster: 'kind-my-cluster'

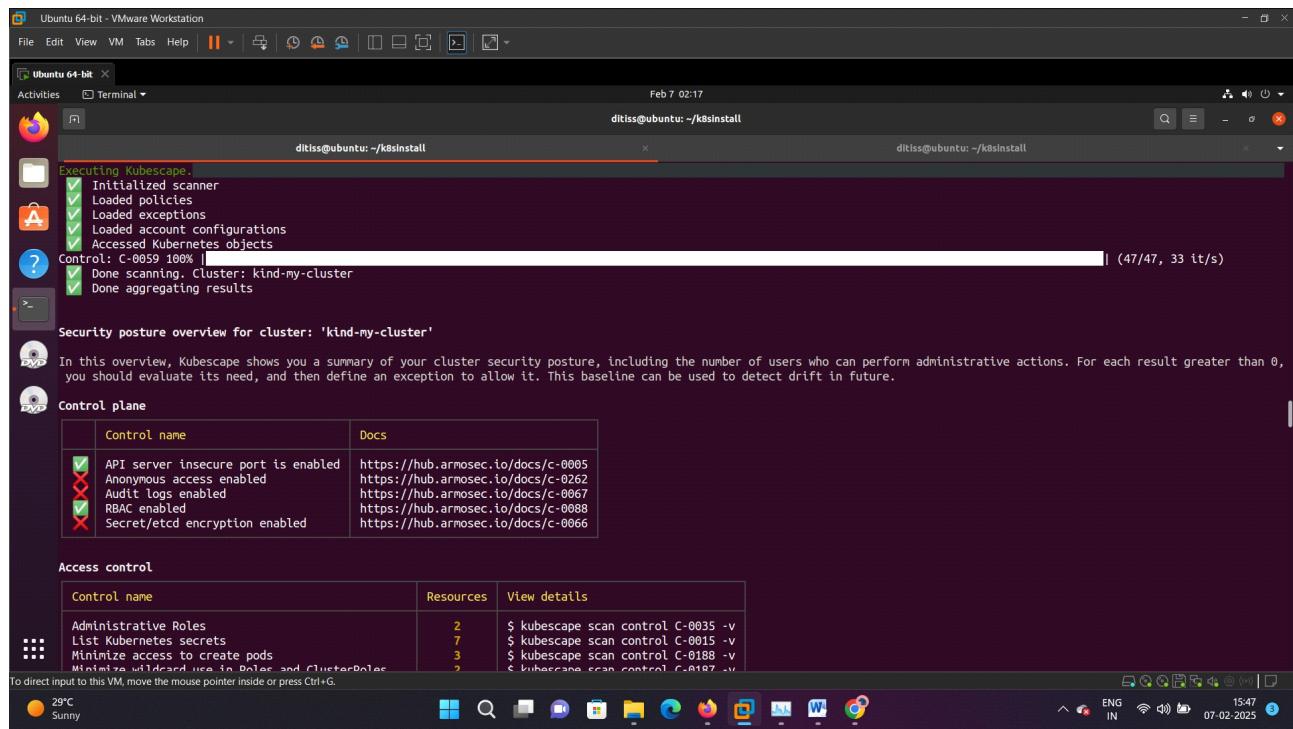
In this overview, Kubescape shows you a summary of your cluster security posture, including the number of users who can perform administrative actions. For each result greater than 0, you should evaluate its need, and then define an exception to allow it. This baseline can be used to detect drift in future.
```

Control plane

Control name	Docs
API server insecure port is enabled	https://hub.armosec.io/docs/c-0005
Anonymous access enabled	https://hub.armosec.io/docs/c-0262
Audit logs enabled	https://hub.armosec.io/docs/c-0067
RBAC enabled	https://hub.armosec.io/docs/c-0088
Secret/etcd encryption enabled	https://hub.armosec.io/docs/c-0066

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

29°C Sunny ENG IN 15:46 07-02-2025



Ubuntu 64-bit - VMware Workstation

File Edit View VM Tabs Help Feb 7 02:17 ditliss@ubuntu: ~/k8sinstall

Activities Terminal ditliss@ubuntu: ~/k8sinstall

```
ditliss@ubuntu: ~/k8sinstall
Executing Kubescape.
[✓] Initialized scanner
[✓] Loaded policies
[✓] Loaded exceptions
[✓] Loaded account configurations
[✓] Accessed Kubernetes objects
Control: C-0059 100% [██████████] (47/47, 33 it/s)
[✓] Done scanning. Cluster: kind-my-cluster
[✓] Done aggregating results

Security posture overview for cluster: 'kind-my-cluster'

In this overview, Kubescape shows you a summary of your cluster security posture, including the number of users who can perform administrative actions. For each result greater than 0, you should evaluate its need, and then define an exception to allow it. This baseline can be used to detect drift in future.

Control plane
```

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

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 | Feb 7 02:17

Activities Terminal ditiss@ubuntu: ~/k8sinstall

Secrets 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

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

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 | Feb 7 02:17

Activities Terminal 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

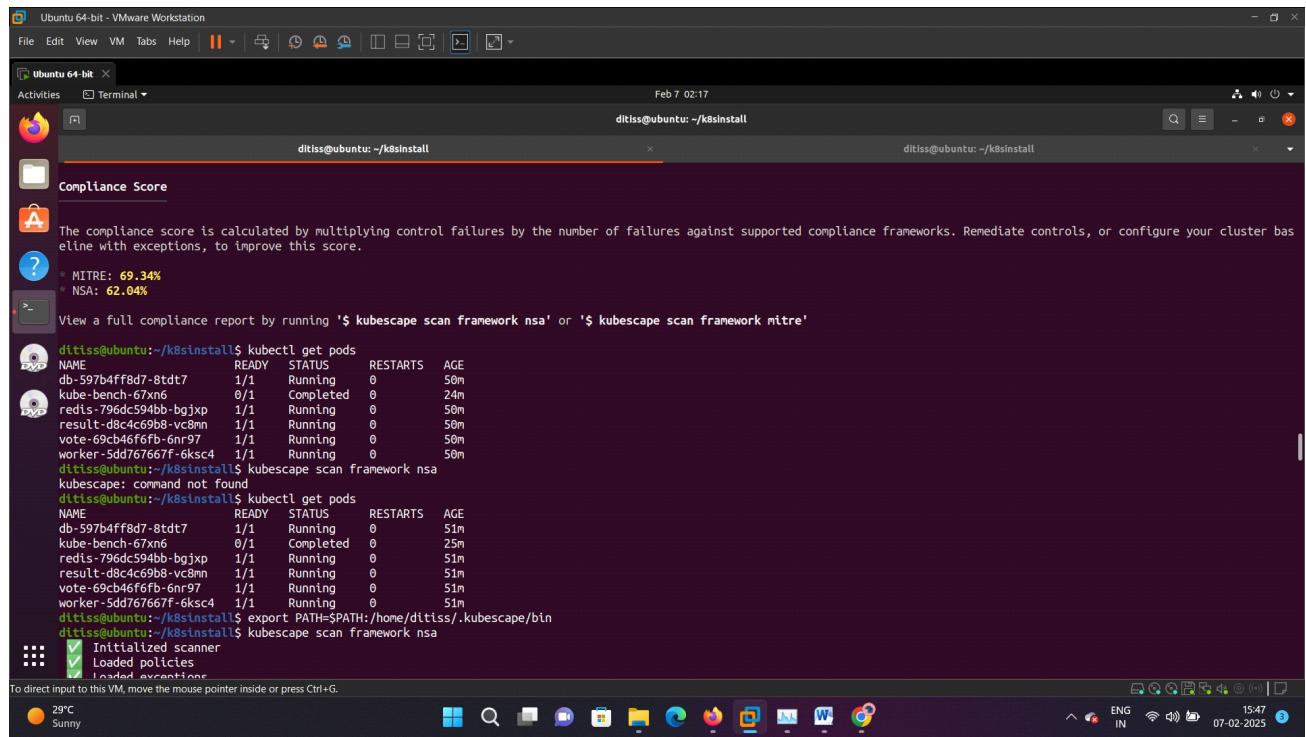
High-stakes workloads are defined as those which Kubescape estimates would have the highest impact if they were to be exploited.

- namespace: monitoring, name: kind-prometheus-prometheus-node-exporter, kind: DaemonSet
'\$ kubescape scan workload DaemonSet/kind-prometheus-prometheus-node-exporter --namespace monitoring'
- namespace: default, name: result, kind: Deployment
'\$ kubescape scan workload Deployment/result --namespace default'
- 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



```

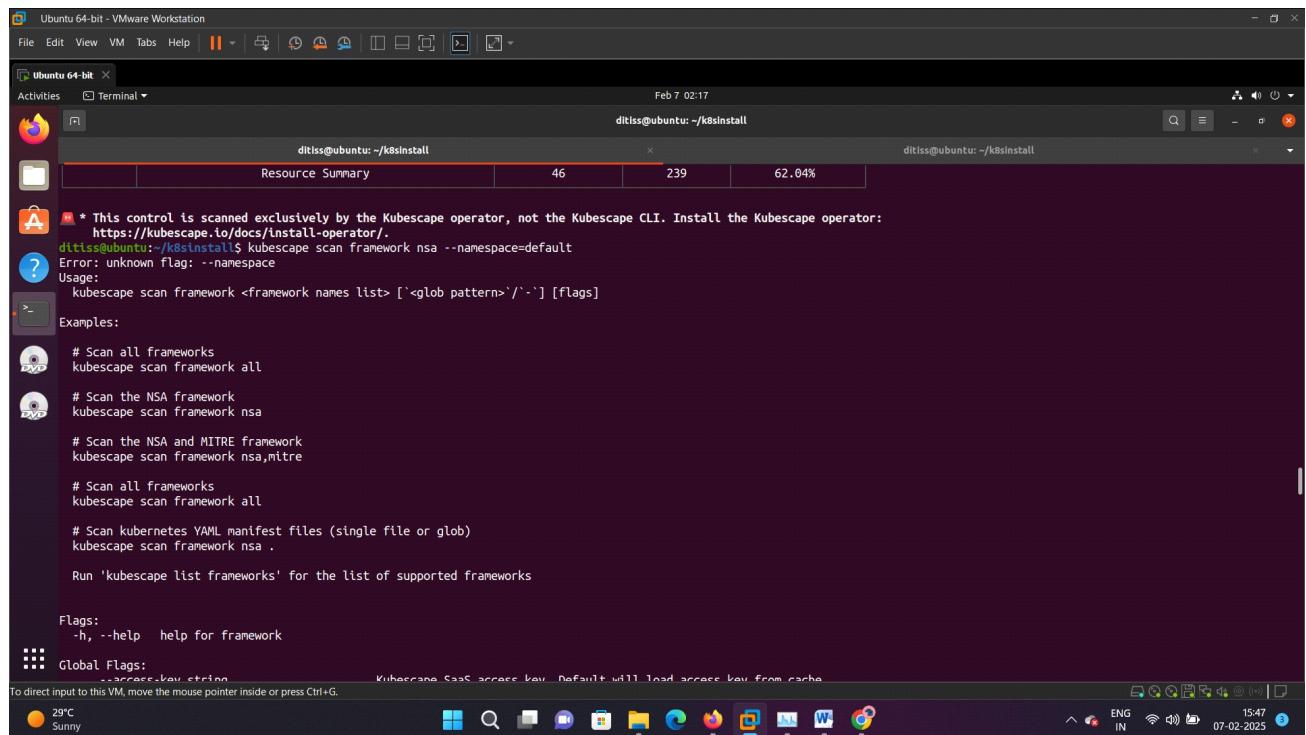
Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help | Terminal | Activities | Feb 7 02:17
ditiss@ubuntu: ~/k8sinstall

Compliance Score
[A] Compliance score is calculated by multiplying control failures by the number of failures against supported compliance frameworks. Remediate controls, or configure your cluster baseline with exceptions, to improve this score.
? * MITRE: 69.34%
* NSA: 62.04%
[>] View a full compliance report by running '$ kubescape scan framework nsa' or '$ kubescape scan framework mitre'

ditiss@ubuntu:~/k8sinstall$ kubectl get pods
NAME READY STATUS RESTARTS AGE
db-597b4ff8d7-8tdt7 1/1 Running 0 50m
kube-bench-67x6 0/1 Completed 0 24m
redis-796dc594bb-bgjxp 1/1 Running 0 50m
result-db8c4cc6908-vc8mn 1/1 Running 0 50m
vote-69cb46f6fb-6nr97 1/1 Running 0 50m
worker-5dd767667f-6ks4c 1/1 Running 0 50m
ditiss@ubuntu:~/k8sinstall$ kubescape scan framework nsa
kubescape: command not found
ditiss@ubuntu:~/k8sinstall$ kubectl get pods
NAME READY STATUS RESTARTS AGE
db-597b4ff8d7-8tdt7 1/1 Running 0 51m
kube-bench-67x6 0/1 Completed 0 25m
redis-796dc594bb-bgjxp 1/1 Running 0 51m
result-db8c4cc6908-vc8mn 1/1 Running 0 51m
vote-69cb46f6fb-6nr97 1/1 Running 0 51m
worker-5dd767667f-6ks4c 1/1 Running 0 51m
ditiss@ubuntu:~/k8sinstall$ export PATH=$PATH:/home/ditiss/.kubescape/bin
ditiss@ubuntu:~/k8sinstall$ kubescape scan framework nsa
[>] Initialized scanner
[>] Loaded policies
[>] Loaded exceptions
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

```

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



```

Ubuntu 64-bit - VMware Workstation
File Edit View VM Tabs Help | Terminal | Activities | Feb 7 02:17
ditiss@ubuntu: ~/k8sinstall

Resource Summary | 46 | 239 | 62.04%
ditiss@ubuntu: ~/k8sinstall

[A] * 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$ kubescape scan framework nsa --namespace=default
Error: unknown flag: --namespace
Usage:
  kubescape scan framework <framework names list> [<glob pattern>] [flags]
Examples:
  # Scan all frameworks
  kubescape scan framework all
  # Scan the NSA framework
  kubescape scan framework nsa
  # Scan the NSA and MITRE framework
  kubescape scan framework nsa,mitre
  # Scan all frameworks
  kubescape scan framework all
  # Scan kubernetes YAML manifest files (single file or glob)
  kubescape scan framework nsa .
Run 'kubescape list frameworks' for the list of supported frameworks

Flags:
  -h, --help    help for framework
[>] Global Flags:
  --access-key string   Kubescape SaaS access key. Default will load access key from cache
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

```

Ubuntu 64-bit - VMware Workstation

File Edit View VM Tabs Help | 

Activities Terminal  Feb 7 02:18 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	Passed	Failed	Action Required
25	10	10	5

Failed resources by severity:

Critical	High	Medium	Low
0	14	30	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 | 

Activities Terminal  Feb 7 02:18 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 log enabled	0	0	Action Required *
Low	Immutable enabled	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 Helm-ing!
ditiss@ubuntu:~/k8sinstall$ helm install kyverno kyverno/kyverno --create-namespaces --set replicasCount=2
```

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

A screenshot of a Linux desktop environment, likely Ubuntu, running in a VMware Workstation window. The desktop has a dark theme with icons for various applications like a browser, file manager, and system tools. A terminal window is open in the foreground, showing a shell script being run. The script uses 'kubescape' to scan a 'nsa' framework, generates a PDF report, and then sends it via email as an attachment. The terminal shows the command: '#!/bin/bash kubescape scan framework nsa -n default --format pdf --output "/home/ditiss/report/\$(date +%F).pdf" > /dev/null 2>&1 echo "Please find the attached PDF." | mutt -s "Subject: PDF Attachment" -a /home/ditiss/report/\$(date +%F).pdf -- probot7757@gmail.com'. The desktop also shows a date and time indicator (Feb 8 04:50) and a system tray with network, battery, and volume icons.

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

Ubuntu 64-bit - VMware Workstation

File Edit View VM Tabs Help || Activities Terminal Feb 8 04:51 ditiss@ubuntu: /usr/local/bin

```
ditiss@ubuntu: /usr/local/bin
mv: cannot stat 'your-script.sh': No such file or directory
ditiss@ubuntu:~/report$ sudo cp automation.sh /usr/local/bin/kubescape-scan.sh
ditiss@ubuntu:~/report$ cd /usr/local/bin
ditiss@ubuntu:~/usr/local/bin$ ls
helm kubectl kubescape-scan.sh
ditiss@ubuntu:/usr/local/bin$ chmod +x kubescape-scan.sh
chmod: changing permissions of 'kubescape-scan.sh': Operation not permitted
ditiss@ubuntu:/usr/local/bin$ sudo chmod +x kubescape-scan.sh
ditiss@ubuntu:/usr/local/bin$ crontab -e
no crontab for ditiss - using an empty one

Select an editor. To change later, run 'select-editor'.
 1. /bin/nano   <---- easiest
 2. /usr/bin/vim.tiny
 3. /bin/ed

ditiss@ubuntu:~/usr/local/bin$ Choose 1-3 [1]: 1
crontab: installing new crontab
ditiss@ubuntu:~/usr/local/bin$ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezone.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

Ubuntu 64-bit - VMware Workstation

File Edit View VM Tabs Help || Activities Terminal Feb 8 04:51 ditiss@ubuntu: /usr/local/bin

Ubuntu 64-bit x ditiss@ubuntu: ~/.email

```
ditiss@ubuntu:~/.email$ crontab -e
crontab: installing new crontab
ditiss@ubuntu:~/.email$ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m. every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
0 */6 * * * /usr/local/bin/kubescape-scan.sh

ditiss@ubuntu:~/.email$ Handling connection for 8443
Handling connection for 8443
Handling connection for 8443
Handling connection for 8443
E0208 04:41:23.043784 10611 portforward.go:362] error creating forwarding stream for port 8443 -> 8080: Timeout occurred
E0208 04:41:31.359716 10611 portforward.go:362] error creating forwarding stream for port 8443 -> 8080: Timeout occurred
E0208 04:41:31.359716 10611 portforward.go:362] error creating forwarding stream for port 8443 -> 8080: Timeout occurred
E0208 04:41:31.359716 10611 portforward.go:362] error creating forwarding stream for port 8443 -> 8080: Timeout occurred

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
```

Kyverno installation via helm

Ubuntu 64-bit - VMware Workstation

File Edit View VM Tabs Help || Activities Terminal Feb 7 02:18 ditiss@ubuntu: ~/k8sinstall

Ubuntu 64-bit x ditiss@ubuntu: ~/k8sinstall ditiss@ubuntu: ~/k8sinstall

WARNING: PolicyExceptions are disabled by default. To enable them, set '--enablePolicyException' to true.

Note: There is a trade-off when deciding which approach to take regarding Namespace exclusions. Please see the documentation at <https://kyverno.io/docs/installation/#security-vs-op-ability-to-understand-the-risks>.

```
ditiss@ubuntu:~/k8sinstall$ helm install kyverno kyverno/kyverno --create-namespace --set replicaCount=3
Error: INSTALLATION FAILED: cannot re-use a name that is still in use
ditiss@ubuntu:~/k8sinstall$ helm uninstall kyverno -n kyverno
```

These resources were kept due to the resource policy:
[ConfigMap] kyverno

release "kyverno" uninstalled
ditiss@ubuntu:~/k8sinstall\$ helm install kyverno kyverno/kyverno --create-namespace --set replicaCount=3
NAME: kyverno
LAST DEPLOYED: Fri Feb 7 01:00:08 2025
NAMESPACE: kyverno
STATUS: deployed
REVISION: 1
NOTES:
Chart version: 3.3.5
Kyverno version: v1.13.3

Thank you for installing kyverno! Your release is named kyverno.

The following components have been installed in your cluster:

- CRDs
- Admission controller
- Reports controller
- Cleanup controller
- Background controller

WARNING: Setting the admission controller replicas count below 2 means Kyverno is not running in high availability mode.

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

29°C Sunny ENG IN 1548 07-02-2025

Ubuntu 64-bit - VMware Workstation

File Edit View VM Tabs Help || Activities Terminal Feb 9 00:24 ditliss@ubuntu: ~

```
GNU nano 4.8
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
```
```

ditliss@ubuntu: ~ Modified

Get Help Write Out Where Is Cut Text Justify Cur Pos Undo Mark Text To Bracket Previous Back Exit Read File Replace Paste Text To Spell Go To Line Undo Redo Copy Text Where Was Next Forward

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

The screenshot shows a dual-terminal session on an Ubuntu 64-bit VM. The left terminal displays the following command sequence:

```
ditiss@ubuntu:~$ sudo nano policies.yaml
[sudo] password for ditiss:
ditiss@ubuntu:~$ ls
config.yaml Documents email kind.sh kubescape-pv.yaml Pictures Public reporting2.pdf Templates Videos
Desktop Downloads k8s-kind-voting-app kubeclt.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
```

The right terminal shows the output of the last command:

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

Ubuntu 64-bit - VMware Workstation

File Edit View VM Tabs Help || 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 1m 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 1m kyverno-scan StatefulSet monitoring/alertmanager-kind-prometheus-kube-prime-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 1m kyverno-scan StatefulSet monitoring/prometheus-kind-prometheus-kube-prime-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 1m 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 1m 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 1m 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 1m 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 1m 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 1m kyverno-scan Deployment monitoring/kind-prometheus-kube-prime-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 1m 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 1m kyverno-scan Deployment argocd/argocd-notifications-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/
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
```

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/>