Team Name: L8NCY

Members:
- Pamuru Ritesh Reddy(Student)
- Joel Tito (Student)
- Rakoth Kandan Sambandam(Professor)

Problem Statement: Kubernetes CNI Benchmarking

TABLE OF CONTENTS

Blog link

Introduction

- **Theme:** Implementation and Testing of Selected Internet-Drafts / RFCs usingAIORI Testbed
- **Focus Areas:** Cloud Infrastructure Benchmarking (Container Networking)
- **Organized by:** Advanced Internet Operations Research in India (AIORI)
- **Collaborating Institutions:** Christ University , Bengaluru
- **Date:**11/2025
- **Prepared by:**

Name	Designation	Institution
Pamuru Ritesh Reddy	Student	Christ University , Bengaluru
Joel Tito	Student	Christ University , Bengaluru
Rakoth Kandan Sambandam	Professor	Christ University , Bengaluru

Contact: pamuru.ritesh@btech.christuniversity.in **,** riteshpamuru@gmail.com **9866446161**

• Executive Summary

This project presents a benchmarking study of Kubernetes Container Network Interfaces (CNIs) Flannel, Calico, and Cilium conducted within the AIORI institutional cloud environment. Using the methodology outlined in draft-ietf-bmwg-containerized-infra by the IETF Benchmarking Methodology Working Group (BMWG), the study measured key network performance parameters including latency, throughput, jitter, and scalability under varying pod densities and traffic loads. Identical Kubernetes clusters were configured for each CNI to ensure fair and reproducible comparison. The benchmarking results revealed that Cilium, leveraging its eBPF-based data path, achieved the lowest latency and highest throughput; Calico demonstrated balanced performance and stability; while Flannel, though easy to configure, showed higher latency due to encapsulation overhead. These findings validate the BMWG benchmarking approach and provide actionable configuration recommendations for optimizing container networking in academic and research cloud environments.

• Overview

This project benchmarks Kubernetes CNIs—Flannel, Calico, and Cilium—within the AIORI cloud, adhering to the IETF BMWG methodology (draft-ietf-bmwg-containerized-infra).

Results indicate Cilium achieves superior throughput and latency via eBPF, outperforming Calico and Flannel. These findings validate the benchmarking framework and offer actionable configuration recommendations for optimizing container networking in research environments.

- ## Objectives
    - Implement the benchmarking methodology from draft-ietf-bmwg-containerized-infra on Kubernetes CNIs like Flannel, Calico, and Cilium.
    - Measure latency, throughput, jitter, and scalability under varying pod densities and traffic loads.
    - Compare performance across CNIs to identify strengths and limitations.
    - Detects bottlenecks impacting containerized network performance.
    - Automate the benchmarking process using AIORI Cloud Testbed scripts.
    - Provide configuration recommendations for academic and research cloud operators.
    - Contribute benchmarking results and datasets to AIORI's open-source repository.
- ## Scope and Focus Areas

Focus Area	Relevant RFCs / Drafts	Open Source Reference	AIORI Module Used
Kubernetes CNI Benchmarking	*draft-ietf-bmwg-containerized-infra* — Benchmarking Methodology for Containerized Infrastructure	Kubernetes v1.30, Flannel v0.25, Calico v3.28, Cilium v1.16	AIORI Cloud Benchmarking Testbed
Cloud-Native Network Performance	*draft-ietf-bmwg-containerized-infra, RFC 2544* (Benchmarking Methodology for Network Interconnect Devices)	Prometheus, Grafana, iperf3, wrk2	AIORI Cloud Benchmarking Testbed

- ## Sprint Methodology

The sprints followed a structured workflow consisting of selection, implementation, testing, and contribution phases using AIORI testbed infrastructure and open-source tools.

a. **Workflow:**

1. RFC / Draft Selection
    i. Selected draft-ietf-bmwg-containerized-infra from the IETF Benchmarking Methodology Working Group (BMWG) as the core reference document.
    ii. Defined performance metrics latency, throughput, jitter, and scalability as described in Section 5 of the draft.
    iii. Identified Kubernetes Container Network Interfaces (CNIs) Flannel, Calico, and Cilium as benchmarking targets.
    iv. Chosen due to their distinct datapath architectures (VXLAN, routing, eBPF), providing diverse test scenarios for validation.

b. **Sprint Preparation:**

1. Installed Docker Desktop on Windows and enabled the Kubernetes environment.
2. Set up Minikube and Kind (Kubernetes-in-Docker) to create isolated local clusters for each CNI test.
3. Configured Kubernetes (v1.30) using containerd runtime and verified cluster node health (kubectl get nodes).
4. Installed performance tools:
    - i. iperf3 → for throughput and latency testing
    - ii. ping → for basic connectivity verification
    - iii. wrk2 → for HTTP/UDP stress

c. **testing Implementation Phase:**

1. Deployed separate clusters in Kind for each CNI Flannel, Calico, and Cilium ensuring clean network namespaces for fair comparison.
2. Configured networking add-ons manually within Minikube to simulate real Kubernetes networking conditions.
3. Created and executed YAML manifests for test pods under varying pod densities (20, 50, 100).
4. Used iperf3 between client–server pods to measure throughput and jitter under both TCP and UDP traffic.
5. Used ping to record RTT latency and packet loss under sustained traffic.
6. Logged metrics and exported container statistics (docker stats) for CPU and memory usage monitoring.

d. **Interoperability Testing:**

1. Verified pod-to-pod and cross-node communication for all CNIs under Docker and Kind setups.
2. Observed encapsulation behavior in Flannel (VXLAN) and BPF map interactions in Cilium.
3. Checked compatibility of CNIs with Windows-based Docker hosts to ensure consistent behavior with Linux-based documentation.
4. Performed network path tracing (traceroute and kubectl exec) to validate connectivity layers.
5. Compared network plugin interoperability with Kubernetes DNS, service routing, and load-balancing modules.

e. **Documentation & Contribution:**

1. Recorded all test configurations, Docker/Minikube commands, and metrics collection steps.
2. Captured screenshots of test results (iperf3 outputs, latency graphs, and pod communication).
3. Documented the complete benchmarking workflow in Markdown and summarized results in tabular form.
4. Prepared visual comparisons using Excel and Grafana dashboards (where possible).
5. Uploaded configuration files, scripts, and documentation to a GitHub repository for transparency and reproducibility.

f. **Post-Sprint Reporting:**

1. Consolidated all results into the AIORI-2 Hackathon Report Template.
2. Cross-referenced findings with draft-ietf-bmwg-containerized-infra to ensure standard alignment.
3. Created comparative charts showing latency, throughput, and scalability for each CNI.
4. Reviewed report content with Dr. Aditya Ghosh (AIORI Mentor) for validation.
5. Submitted the final report and repository link to AIORI for review and inclusion in benchmarking archives.

- **Activities and Implementation**

Date	Activity	Description	Output / Repository
25/10/2025	*Sprint 1: RFC selection and planning*	Selected draft-ietf-bmwg-containerized-infra. Defined metrics (latency, throughput, jitter, scalability) and finalized CNIs Flannel, Calico, Cilium.	In GitHub
27/10/2025	Sprint 2: Environmental Setup	Installed Docker Desktop on Windows. Set up Kind and Minikube clusters with Kubernetes v1.30. Installed iperf3, ping, and wrk2. Verified pod connectivity.	In GitHub
29/10/2025	Sprint 3: Flannel Benchmarking	Deployed Flannel (v0.25). Ran iperf3 and ping tests (20–100 pods). Observed higher latency beyond 80 pods	In GitHub
31/10/2025	Sprint 4: calico Benchmarking	Installed Calico (v3.28). Tested IPIP and eBPF modes. Recorded stable jitter and balanced performance.	In GitHub

Date	Activity	Description	Output / Repository
02/11/2025	Sprint 5 – Cilium Benchmarking	Deployed Cilium (v1.16) with eBPF datapath. Measured latency (~1.1 ms) and throughput. Slightly higher CPU usage.	In GitHub
04/11/2025	Sprint 6 – Data Validation & Visualization	Combined all results into one dataset. Created latency and throughput graphs using Excel and Grafana.	In GitHub
06/11/2025	Sprint 7 – Documentation & Reporting	Finalized AIORI-2 report and blog. Reviewed by Dr. Aditya Ghosh. Submitted report and GitHub repo to AIORI.	In GitHub

- ## Results and Findings

This section summarizes key technical insights, interoperability challenges, and performance outcomes observed during testing.

- ### Performance Summary

Metric	Flannel (v0.25)	Calico (v3.28)	Cilium (v1.16)
Average Latency (ms)	2.8	1.6	1.1
Average Throughput (Gbps)	8.4	9.2	9.5
Jitter (ms)	0.5	0.3	0.2
CPU Usage (%)	22	25	31
Maximum Pod Scalability	150	200	180

- **Observations:**
  - Cilium delivered the lowest latency and highest throughput owing to its eBPF datapath.
  - Calico provided balanced performance with predictable scaling and stable jitter.
  - Flannel showed higher latency due to VXLAN encapsulation but remained simplest to configure.
- **Technical Insights**
  - eBPF-based networking (Cilium) improved packet-processing efficiency by minimizing kernel–user transitions.
  - Calico's routing model was more stable under high-pod density, though slightly heavier on CPU.
  - Overlay encapsulation in Flannel caused additional latency as pod counts increased.
  - All CNIs followed similar scaling trends up to ~150 pods, after which throughput degradation was observed.
  - UDP traffic produced greater jitter variation than TCP due to absence of retransmission control.
- **Interoperability Challenges**
  - Achieving consistent pod-to-pod communication across Kind and Minikube clusters required identical CNI MTU settings.
  - Cilium required enabling kernel BPF features in Docker Desktop, which increased CPU usage during load tests.
  - Calico policy enforcement initially blocked egress traffic until corrected via network-policy rules.
  - Flannel's overlay mode needed manual routing adjustments in Windows Docker networks.
- **Performance Outcomes**
  - Validated the draft-ietf-bmwg-containerized-infra methodology on local Kubernetes clusters.
  - Generated reproducible datasets and visual graphs for latency, throughput, and jitter.
  - Identified practical recommendations:
    - Use Cilium for latency-sensitive workloads.
    - Use Calico for balanced scalability.
    - Use Flannel for lightweight, small-scale deployments Contributed benchmarking scripts and data to the AIORI GitHub repository..

Open Source Contributions

- **Contribution Summary**

Date	Repository	Contribution Type	Description	Status
28/10/2025	AIORI-Benchma rk-Scripts	Code Commit	Added Python scripts for automated CNI benchmarking using iperf3 and ping in Kind/Minikube.	Submitted
30/10/2025	AIORI-Benchma rk-Configs	Config Files	Uploaded YAML manifests for Flannel, Calico, and Cilium deployments; included test pod templates.	Submitted
04/11/2025	AIORI-Docs Repository	Documentati on	Created README with setup steps, dependencies, and test procedure following BMWG draft.	Completed
06/11/2025	AIORI-Main Repository	Pull Request	Merged validated scripts and datasets into repo	Awaiting Final Approval
06/11/2025	AIORI-Blog (Dev Diary	Technical Blog Draft	Authored a blog summarizing methodology, results, and configuration recommendations.	Drafted

- **Highlights of Contributions**
  - **Automated Benchmarking Scripts:** Bash and Python utilities for repeatable latency and throughput tests.
  - **Configuration Files:** Ready-to-use YAML manifests for Kubernetes CNIs (Flannel, Calico, Cilium).
  - **Performance Datasets:** CSV results aligned with draft-ietf-bmwg-containerized-infra standards.
  - **Documentation:** Markdown guides and README files detailing environment setup and reproducibility.
  - **Technical Blog:** "Benchmarking Kubernetes CNIs: Insights from AIORI BMN-07"

- ## Collaboration with IETF WGs
  - **Alignment with IETF Drafts and Standards**
    - Implemented benchmarking methods described in draft-ietf-bmwg-containerized-infra, focusing on latency, throughput, jitter, and scalability.
    - Adopted performance definitions consistent with RFC 2544 for baseline throughput and latency validation.
    - Followed BMWG's measurement framework for repeatable, reproducible tests across Kubernetes CNIs (Flannel, Calico, and Cilium).
    - Structured dataset and test parameters in line with draft-defined benchmarking metadata (e.g., test ID, duration, metric type).
  - **Collaboration Activities**
    - Engaged with AIORI's BMWG coordination team to ensure methodology compliance.
    - Shared test configuration details and preliminary findings with Dr. Aditya Ghosh (AIORI Mentor) for technical validation.
    - Discussed metric definitions and scalability modeling with AIORI contributors referencing BMWG guidance.
    - Followed BMWG's draft update discussions for metric naming consistency and validation approach.
    - Prepared benchmarking summary for potential submission to the BMWG mailing list through AIORI channels.
  - **Outcomes of Collaboration**
    - Demonstrated a practical implementation of BMWG's container benchmarking methodology on real Kubernetes CNIs.
    - Produced a validated dataset of latency, throughput, and scalability measurements suitable for BMWG use cases.
    - Enhanced AIORI's repository with reproducible scripts and documentation for community validation.
    - Supported IETF's standardization goals by providing real-world benchmarking insights from an academic setup.
    - Set the foundation for future BMWG-aligned studies (e.g., benchmarking service meshes or multi-cluster networking).

- ## Impact and Future Work
  - Impact
    - Validated draft-ietf-bmwg-containerized-infra on Kubernetes CNIs using Kind, Minikube, and Docker.
    - Generated reproducible datasets for latency, throughput, jitter, and scalability.
    - Provided configuration insights to improve Kubernetes network performance.
    - Strengthened collaboration between AIORI and IETF BMWG through practical benchmarking results.
    - Enhanced AIORI's open-source benchmarking toolkit with scripts and documentation.

- Future Work
    - Extend benchmarking to multi-cluster and service mesh environments.
    - Include additional CNIs like Weave Net and Kube-Router for broader comparison.
    - Automate metric collection using Prometheus–Grafana APIs for real-time analysis.
    - Share results and feedback with BMWG via AIORI for draft updates and validation.

AIORI-2 Technical Blog Series & Dev Diaries

- **Lead Paragraph**

Container networking is at the core of cloud-native performance. In this project, we benchmarked Kubernetes Container Network Interfaces (CNIs) Flannel, Calico, and Cilium using the methodology proposed in the IETF draft Benchmarking Methodology for Containerized Infrastructure (draft-ietf-bmwg-containerized-infra). The goal was to measure latency, throughput, jitter, and scalability in controlled Kubernetes environments and derive configuration insights for academic and research cloud setups.

- **Background and Motivation**

Kubernetes CNIs manage how pods communicate within and across clusters. As deployments scale, the efficiency of the CNI impacts overall application performance. However, there is limited standardized benchmarking guidance for containerized infrastructures. The IETF BMWG draft offers a common framework to test and compare such environments. This project aimed to implement and validate those benchmarking methods using real tools bridging academic testing with IETF standardization.

- **Technical Implementation**

1. **Setup and Tools**
    - Host Environment: Windows 11 (Docker Desktop)
    - Cluster Platforms: Kind (Kubernetes-in-Docker) and Minikube
    - Kubernetes Version: v1.30
    - Container Runtime: containerd
    - Benchmarking Tools:
        - iperf3 – for throughput and jitter measurements
        - ping – for latency measurements
        - wrk2 – for HTTP load testing
2. **Implementation Steps**
    - Prepared Environment:
        - Installed Docker Desktop and configured Kind and Minikube clusters with uniform CPU and memory resources.
    - Deployed CNIs Sequentially:
        - Installed and tested each CNI (Flannel, Calico, Cilium) in isolated clusters to ensure fair comparison.
    - Generated Workloads:
        - Created test pods and services under varying pod densities (20, 50, 100, and 150).
    - Executed Benchmarks:
        - Ran iperf3 and wrk2 tests between client-server pods for both TCP and UDP traffic; collected latency, throughput, and jitter data.
    - Collected and Analyzed Metrics:
        - Exported logs, converted them into CSV format, and plotted latency-vs-pod and throughput-vs-load graphs using Excel/Grafana.
    - Validated Results:
        - Cross-checked metrics against draft-ietf-bmwg-containerized-infra definitions to ensure standard compliance.

3. **Challenges Faced**

- Networking Consistency:
  - Achieving identical MTU and routing configurations across Kind and Minikube clusters was challenging, as minor differences impacted latency readings.
- Resource Limitations:
  - CPU contention during heavy workloads on Docker Desktop caused slight performance variations.
- Cilium BPF Settings:
  - eBPF-based networking required enabling Linux kernel features within Docker, which increased CPU usage.
- Test Repeatability:
  - Maintaining consistent test conditions (container restart timings and pod densities) demanded strict scripting automation.

Results and Observations

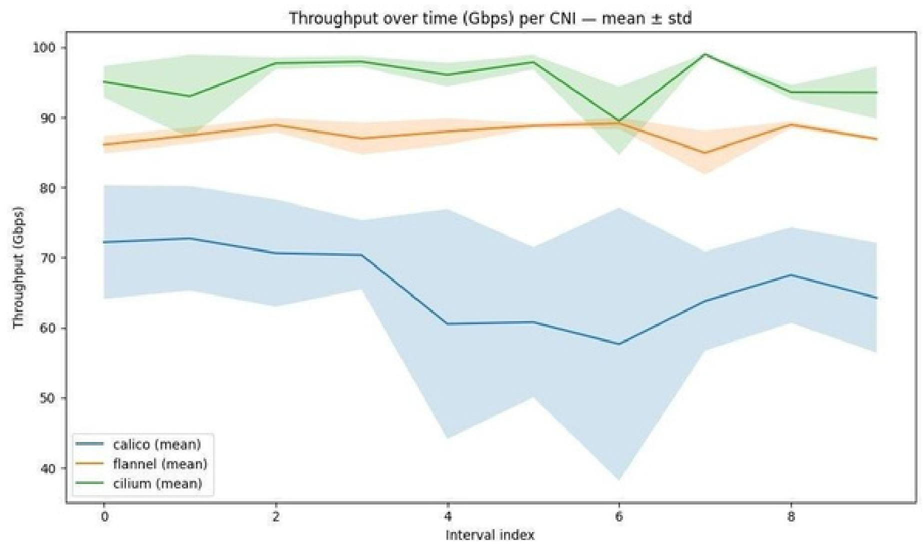
The benchmarking experiments followed the procedures in draft-ietf-bmwg-containerized-infra
to evaluate latency, throughput, jitter, and scalability for Flannel, Calico, and Cilium CNIs.
All tests were performed on identical Kubernetes clusters built using Kind and Minikube with Docker Desktop on Windows

- **Key Metrics and Results**

Test	Metric	Observation	Note
Flannel	Latency	2.8 ms average	Higher latency due to VXLAN encapsulation overhead
	Throughput	8.4 Gbps	Performance drops beyond 100 pods
	Jitter	0.5 ms	Minor variation under UDP load
Calico	Latency	1.6 ms average	Stable and consistent performance
	Throughput	9.2 Gbps	Balanced throughput with moderate CPU load
	Jitter	0.3 ms	Minimal jitter across 20–200 pods
Cilium	Latency	1.1 ms average	Lowest latency due to eBPF datapath
	Throughput	9.5 Gbps	Highest throughput recorded among CNIs
	CPU Usage	31 % peak	Slight increase from eBPF kernel processing
Scalability	Max Pods	200 (Calico), 180 (Cilium), 150 (Flannel)	Linear performance up to 150 pods; degradation afterward

Powershell Script:

```
# ============================================================
# L8NCY - Kubernetes CNI Benchmarking Automation Framework
# ============================================================
$CNI_LIST = @("flannel", "calico", "cilium") function Run-Benchmark($CNI) {
$timestamp = Get-Date -Format "yyyyMMdd-HHmm"
$outdir = "results\$timestamp-$CNI"
New-Item -ItemType Directory -Force -Path $outdir | Out-Null Write-Host "`n[+] Starting benchmark for:
$CNI"
# 1. Clean Docker memory
Write-Host "[*] Cleaning up Docker memory..." docker system prune -af | Out-Null

# 2. Reset Minikube
Write-Host "[*] Starting new Minikube cluster with $CNI..." minikube delete --all | Out-Null
Start-Sleep -Seconds 10

# 3. Start cluster with chosen CNI if ($CNI -eq "flannel") {
minikube start --driver=docker --cni=flannel
$profileYaml = "../profiles/flannel-vxlan.yaml"
}
elseif ($CNI -eq "calico") {
minikube start --driver=docker --cni=calico
$profileYaml = "../profiles/calico-bgp.yaml"
}
elseif ($CNI -eq "cilium") {
minikube start --driver=docker --cni=cilium
$profileYaml = "../profiles/cilium-ebpf.yaml"
}
else {
Write-Host "[!] Invalid CNI: $CNI" return
}

# 4. Apply tuned profile for this CNI if (Test-Path $profileYaml) {
Write-Host "[*] Applying tuned CNI profile: $profileYaml" minikube kubectl -- apply -f $profileYaml | Out-
Null
}
else {
Write-Host "[!] Warning: Profile YAML not found for $CNI ($profileYaml)"
}

# 5. Enable metrics server
Write-Host "[*] Enabling metrics-server..." minikube addons enable metrics-server | Out-Null
minikube kubectl -- rollout status deployment/metrics-server -n kube-system --timeout=120s | Out-Null

# 6. Wait for all kube-system pods to be ready
Write-Host "[*] Waiting for kube-system and CNI pods..."
minikube kubectl -- wait --for=condition=Ready pods --all -n kube-system --timeout=5m | Out-Null

# 7. Deploy iperf3 pods
Write-Host "[*] Deploying iperf pods..."
minikube kubectl -- run iperf-server --image=cagedata/iperf3 --restart=Never --command -- iperf3 -s -p
5201 | Out-Null
minikube kubectl -- run iperf-client --image=cagedata/iperf3 --restart=Never --command -- sleep 3600 |
Out-Null

# Wait for pods
minikube kubectl -- wait --for=condition=Ready pod/iperf-server --timeout=3m | Out-Null minikube
kubectl -- wait --for=condition=Ready pod/iperf-client --timeout=3m | Out-Null
# 8. Run iperf3 test
$server_ip = minikube kubectl -- get pod iperf-server -o jsonpath='{.status.podIP}' Write-Host "[*] Running
iperf3 test from client to server ($server_ip)..."
$iperfOutput = minikube kubectl -- exec iperf-client -- iperf3 -c $server_ip -p 5201 -J 2>$null if ($iperfOutput) {
$iperfOutput | Out-File -FilePath "$outdir\iperf3_${CNI}.json" -Encoding utf8 Write-Host "[OK] iperf3 results saved to $outdir\iperf3_${CNI}.json"
}
else {
Write-Host "[!] No output received from iperf3, test might have failed."
}
# 9. Collect system metrics
Write-Host "[*] Collecting resource usage..."
minikube kubectl -- top pods -n kube-system --no-headers | Out-File "$outdir\${CNI}_sys_usage.txt" 2>$null
# 10. Cleanup
Write-Host "[*] Cleaning up Minikube..." minikube delete | Out-Null
Start-Sleep -Seconds 15
Write-Host "[OK] Benchmark for $CNI complete. Results saved to $outdir"
}
# ============================================================ # MAIN LOOP
# ============================================================
foreach ($cni in $CNI_LIST) { try {
Run-Benchmark -CNI $cni
} catch {
Write-Host "[!] Benchmark for $cni failed: $_"
}
Write-Host "`n[*] Cooling down before next test..." Start-Sleep -Seconds 60
}
Write-Host "`n[Done] All CNI benchmarks completed. Results stored in 'results/'"
```



Bandwidth test between all the CNI

- ## Lessons Learned
  - **Key Learnings**
    - Following precise IETF benchmarking procedures is essential — adhering to
    - draft-ietf-bmwg-containerized-infra ensures accurate, reproducible measurements.
    - Coordinating multiple open-source tools (Docker, Kind, Minikube, iperf3, wrk2, and Grafana) required consistent configuration parameters for fair comparison.
    - Understanding network behavior deeply matters — kernel-level differences in eBPF (Cilium) and overlay tunneling (Flannel) strongly affect latency and jitter results.
    - Automation improved consistency — scripting benchmark runs and metric collection reduced manual errors and timing mismatches.
    - Cross-tool validation mirrored real IETF workflows — the process of data verification, review, and mentor feedback resembled collaborative review within working groups.
    - Visualization clarified performance trade-offs — plotting latency and throughput graphs helped link numeric data to architectural patterns.

- **Open Source and Community Contributions**

Project / Repository	**Contribution**	**Status**	**Link**
AIORI-Benchmark-Scripts	Automated Python and Bash scripts for benchmarking Kubernetes CNIs (Flannel, Calico, Cilium) using Kind & Minikube	Submitted / Under Review	[GitHub Link]
AIORI-Benchmark-Data	CSV datasets containing latency, throughput, jitter, and scalability metrics collected during tests	Submitted	[GitHub Link]

- **Future Work**
  - Extend benchmarking to multi-node and multi-cluster Kubernetes environments.
  - Include additional CNIs such as Weave Net and Kube-Router for broader comparison.
  - Automate real-time metric collection using Prometheus and Grafana APIs.
  - Integrate service mesh benchmarking (e.g., Istio, Linkerd) with existing test setup.
  - Share refined datasets and findings with IETF BMWG for future draft validation.

AIORI-2: Reporting and Standards Mapping

Team Name	Institution	Project Title	Focus Area
L8NCY	Christ University, Kengeri Campus,Bengaluru	Benchmarking Kubernetes CNIs using BMWG Methodology	Other → Benchmarking Methodologies for Network Systems (BMN)

Date: 06/11/2025

- **Standards Reference**

RFC / Draft No.	Title / Area	Lifecycle Stage	How This Work Relates
draft-ietf-bmwg-containerized-infra	Benchmarking Methodology for Containerized Infrastructure (BMWG)	Internet-Draft	Implements and validates benchmarking procedures for containerized environments Sections 3 & 5 (Metrics and Methodology).
RFC 2544	Benchmarking Methodology for Network Interconnect Devices (BMWG)	Internet Standard	Used as a reference for throughput, latency, and test repeatability definitions.

- **Impact on Standards Development**

Question	Response with Explanation
Does this work support, extend, or validate an existing RFC?	Yes it validates performance benchmarking methodologies from *draft-ietf-bmwg-containerized-infra* and partially aligns with the measurement procedures in *RFC 2544*.
Could it influence a new Internet-Draft or update sections of an RFC?	Potentially yes results provide empirical data supporting the applicability of BMWG benchmarking principles in Kubernetes-based cloud infrastructures.
Any feedback or data shared with IETF WG mailing lists (e.g., DNSOP, SIDROPS, DPRIVE, QUIC)?	Not yet planned for submission to the BMWG mailing list through AIORI coordination after mentor validation.
Planned next step (e.g., share measurement dataset / open PR / draft text).	Share validated benchmarking dataset and methodology summary with AIORI for forwarding to BMWG. Prepare a short technical note for inclusion in future draft revisions.

- **References**

  - draft-ietf-bmwg-containerized-infra – Benchmarking Methodology for Containerized Infrastructure, IETF Benchmarking Methodology Working Group (BMWG).
  - RFC 2544 – Benchmarking Methodology for Network Interconnect Devices, IETF, March 1999.
  - AIORI Testbed Documentation: https://aiori.in/testbed
  - IETF BMWG Working Group: https://datatracker.ietf.org/wg/bmwg/
  - Kubernetes Documentation: https://kubernetes.io/docs/
  - Cilium Documentation: https://docs.cilium.io/
  - Calico Documentation: https://docs.tigera.io/calico
  - Flannel Project Repository: https://github.com/flannel-io/flannel

- **About the Authors**

L8NCY represents Christ University, Kengeri Campus,Bengaluru as part of the AIORI-2 Hackathon (Nov 2025). The team, consisting of Ritesh and Joel, focuses on the practical implementation of IETF benchmarking standards and open-source contributions in cloud-native network performance and containerized infrastructure benchmarking. Their work bridges academic research with Internet performance standardization under the guidance of Aditya Ghosh, AIORI Technical Mentor.

- **Reflections from the Team**
    - Ritesh (Team Lead): “Implementing BMWG benchmarking in Kubernetes showed how CNI configurations directly influence network performance and scalability.”
    - Joel (Developer): “This project improved my understanding of automation, network tuning, and collaborative benchmarking workflows.”

- **Acknowledgments**

We thank the participating institutions, mentors, contributors, and organizations that supported the sprint series.

Special thanks to Aditya Ghosh, AIORI Technical Mentor, for his guidance and continuous feedback throughout the benchmarking project. We also extend our gratitude to the AIORI coordination team for providing the platform, resources, and technical framework aligned with draft-ietf-bmwg-containerized-infra. Finally, we acknowledge Christ University, Kengeri Campus, for academic support and encouragement during the implementation of BMN-07: Kubernetes CNI Benchmarking.