Isovalent Private Networks for AI Inference on Cisco Unified Edge



© 2026 Cisco and/or its affiliates. All rights reserved.

Contents

Executive summary **3**

Introduction **4**

Lab implementation 5

Cisco Unified Edge 5

Red Hat OpenShift Virtualization 6

Red Hat AI Inference Server 6

Isovalent Enterprise Platform 6

How Private Networks works 7

Workload network configuration 8

The inference path inside the cluster 9

The external path to the system of record 10

Security policy 10

Key findings 12

Conclusion 13

References 13

Authors 13

© 2026 Cisco and/or its affiliates. All rights reserved.

Executive summary

Organizations may need to add a containerized AI inference service at an edge site while keeping an existing application in a Virtual Machine (VM). The inference service must run locally when decisions are time-sensitive or when data must remain at that site. The VM must communicate directly with the inference pod without exposing that path to external networks.

Separate VM and container networking models make this harder to manage. Teams must maintain duplicate controls and separate views of the same application traffic. A change in the VM's IP address can also affect firewall rules, database access entries, and audit records. The platform therefore needs an isolated VM-to-pod path, stable workload IP addresses, controlled access to external systems, and shared traffic visibility.

Cisco® Unified Edge provides the edge platform for this mixed environment. Red Hat OpenShift Virtualization brings VMs and containers into the same cluster. Isovalent® Networking for Virtualization (INV) provides isolated networks for both VMs and containers through its Private Networks feature. Network policy controls traffic, and Hubble provides shared visibility across both workload types. This combination fits edge locations that need containerized AI services and application continuity at the same time.

This paper presents a lab-tested example of that combination. It is a pattern to evaluate, not a Cisco Validated Design or production recommendation.

Table 1. Scope

Scope	Items
Validated in this lab	Private VM-to-pod connectivity, source address preservation, selective external database access, and shared flow visibility
Not validated	Scalability, failover, upgrades, performance, encryption, privacy and compliance, AI security design, and fleet operations

© 2026 Cisco and/or its affiliates. All rights reserved.

Introduction

Edge applications take many forms. In one common pattern, existing business logic stays in a VM while new services run in containers. The VM may run on the same cluster as the containers, often as a result of migration from an existing hypervisor. Databases and other systems of record may remain on existing infrastructure because their location, ownership, or lifecycle differs from the new application service.

This lab uses AI inference to demonstrate that application mix. The inference service runs as a container on an OpenShift node with GPU acceleration, close to the application that consumes it. The application remains in a VM on the same cluster and continues to use an established database outside OpenShift. The application follows this sequence:

1. It reads an input record from PostgreSQL.

2. It sends a request to the inference service and receives the response generated by the AI model.

3. It writes the result back to PostgreSQL.

The read-infer-write order matters when the system of record is already established and only the inference service is new. Insurance claims handling is a good example. A claim handler's application pulls up a claim, sends the supporting documents to the inference service, and gets back the key facts, such as what was damaged, who was involved, and whether anyone was hurt. The application then saves that summary back to the claims system, which remains the system of record.

This leads to two things. The claim data stays where it is, so the application talks to the claims system the same way it always has. The AI service is the only new piece, and it sits right next to the application on the same cluster, so nothing outside needs to reach it. The application therefore has two very different paths, one that leaves the cluster and one that never does.

Table 2. Two data paths

Flow	Destination	Requirement
AI inference	Inference service in the cluster	Direct connectivity within OpenShift
Data access	PostgreSQL in the edge network	IP connectivity across the cluster boundary

© 2026 Cisco and/or its affiliates. All rights reserved.

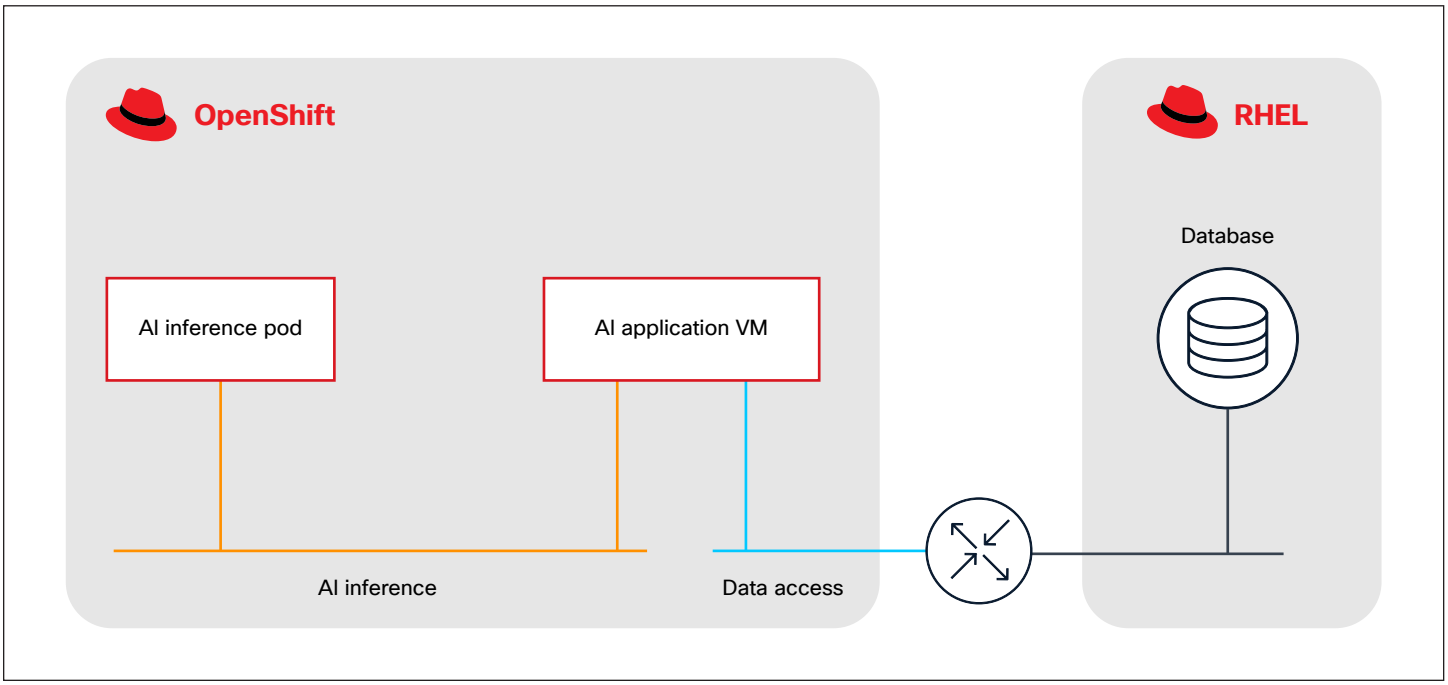


Figure 1. Logical topology

Lab implementation

Cisco Unified Edge

The Cisco UCS® XE9305 Chassis is a 3-Rack-Unit (3RU) hardware platform designed for edge computing environments. These environments have less space and less on-site IT support than data centers. The chassis combines compute, storage, networking, and security in one modular system, with five slots for Cisco UCS XE130c M8 Compute Nodes. Each node supports a PCIe accelerator slot compatible with an NVIDIA L4 GPU.

Two Edge Chassis Management Controllers (eCMCs) connect the chassis to the upstream network and carry both management and data traffic for every node over shared uplinks. Because each node reaches the network through internal chassis paths rather than its own external cable, a site needs far fewer upstream switch ports, and fewer cable runs also make ongoing maintenance simpler.

Cisco Intersight® gives operations teams one place to manage Cisco Unified Edge chassis and compute nodes. It provides a common view of inventory and health, while profiles and policies support repeatable configurations. After a new system is claimed, zero-touch provisioning can configure its hardware and install the operating system. Fleet management extends this control across distributed edge sites. Teams can also schedule configuration updates across the fleet, which reduces repetitive work at remote sites.

The lab runs on one Cisco UCS XE9305 Chassis, and its uplinks connect to the Meraki®-based edge network through VLAN trunking interfaces. Three Cisco UCS XE130c M8 Compute Nodes in the chassis form an OpenShift compact cluster. One of them has an NVIDIA L4 GPU for inference.

© 2026 Cisco and/or its affiliates. All rights reserved.

Red Hat OpenShift Virtualization

Red Hat OpenShift Virtualization extends the container platform so that virtual machines and containers can run side by side. This capability is delivered through KubeVirt, an open-source project that enables OpenShift to treat virtual machines as native Kubernetes resources, managed with the same tools and workflows used for containers.

Red Hat AI Inference Server

Red Hat AI Inference Server provides an OpenAI-compatible endpoint delivered as a container with direct access to a GPU. It is built on vLLM, an open-source inference engine optimized for high-performance model serving. The application running within the VM interacts with the endpoint using standard OpenAI chat completion requests and responses.

In this lab, the platform serves RedHatAI/Meta-Llama-3.1-8B-Instruct-FP8-dynamic, an FP8-quantized version of Meta-Llama-3.1-8B-Instruct that has been validated by Red Hat. Quantization reduces the amount of VRAM required to run the model, allowing it to fit within the memory capacity of the NVIDIA L4 GPU on the compute node.

Isovalent Enterprise Platform

Isovalent Enterprise Platform provides networking, security, and observability capabilities. Isovalent Networking for Kubernetes and Isovalent Networking for Virtualization (INV) are part of Isovalent Enterprise Platform.

Isovalent Networking for Kubernetes is based on Cilium, an open-source networking project hosted by the Cloud Native Computing Foundation. Its data plane runs inside the Linux kernel using eBPF, which reduces CPU overhead and latency compared with other solutions that depend on iptables and userspace processing. It segments workloads by IP address, and also by identity, which comes from labels on the workload. In this lab it serves as the cluster's Container Network Interface (CNI) plugin, the component that connects every pod to the network.

Isovalent Networking for Virtualization (INV) extends that networking to VM workloads, and Private Networks is one of its features. The lab uses Private Networks to segment the network, Isovalent Network Policy to control the database path, and Hubble, the observability component of Isovalent Enterprise, to record the traffic.

Table 3. Hardware and software list

Component	Role in this lab	Version used in the lab
Cisco UCS XE9305 Chassis	Edge infrastructure platform	Infrastructure Bundle 6.0(1.251006)
Cisco UCS XE130c M8 Compute Node	Compute hardware, with NVIDIA L4 GPU	Server Bundle 6.0(1.251026)
Red Hat OpenShift Container Platform	Cluster platform for containers and VMs	4.20.24
Red Hat OpenShift Virtualization	Platform for the existing application VM	4.20.21
Red Hat AI Inference Server	Serving platform for the RedHatAI/Meta-Llama-3.1-8B-Instruct-FP8-dynamic model	3.4.0
Isovalent Enterprise Platform	Container Network Interface (CNI) and Private Networks	1.19.5

© 2026 Cisco and/or its affiliates. All rights reserved.

How Private Networks works

Private Networks is a network segmentation feature introduced in Isovalent Enterprise Platform 1.19. Each Private Network has its own addresses and routes and is isolated from the others, similar to a Virtual Private Cloud (VPC). Workloads in the same Private Network can communicate, while workloads in different Private Networks remain isolated from each other.

By default, a Private Network has no connection outside the cluster. If access to external resources is required, local-access provides a routed Layer-3 exit through a dedicated physical interface or VLAN subinterface on the worker nodes.

The lab uses one Private Network for each of the VM's two destinations. The ai-inference network carries traffic to the inference service and stays inside the cluster. The ai-workloads network carries traffic across the cluster boundary to the database in the edge network.

A single Private Network could carry both paths. However, local-access operates at the Private Network level, not at the workload level. Once enabled, every attached workload shares the routed exit, including the inference pod. Isolation from the edge network would then depend on the policy being present and configured correctly. The two-network design limits the external exit to ai-workloads. The ai-inference network has no external path, even if a policy is absent or configured incorrectly.

Table 4. Private Network details

Private Network	Subnet	Attached workloads
ai-workloads	10.131.8.0/24	Application VM (primary interface, 10.131.8.11)
ai-inference	172.16.8.0/24	Application VM (secondary interface, 172.16.8.11) and inference pod (primary interface, 172.16.8.10)

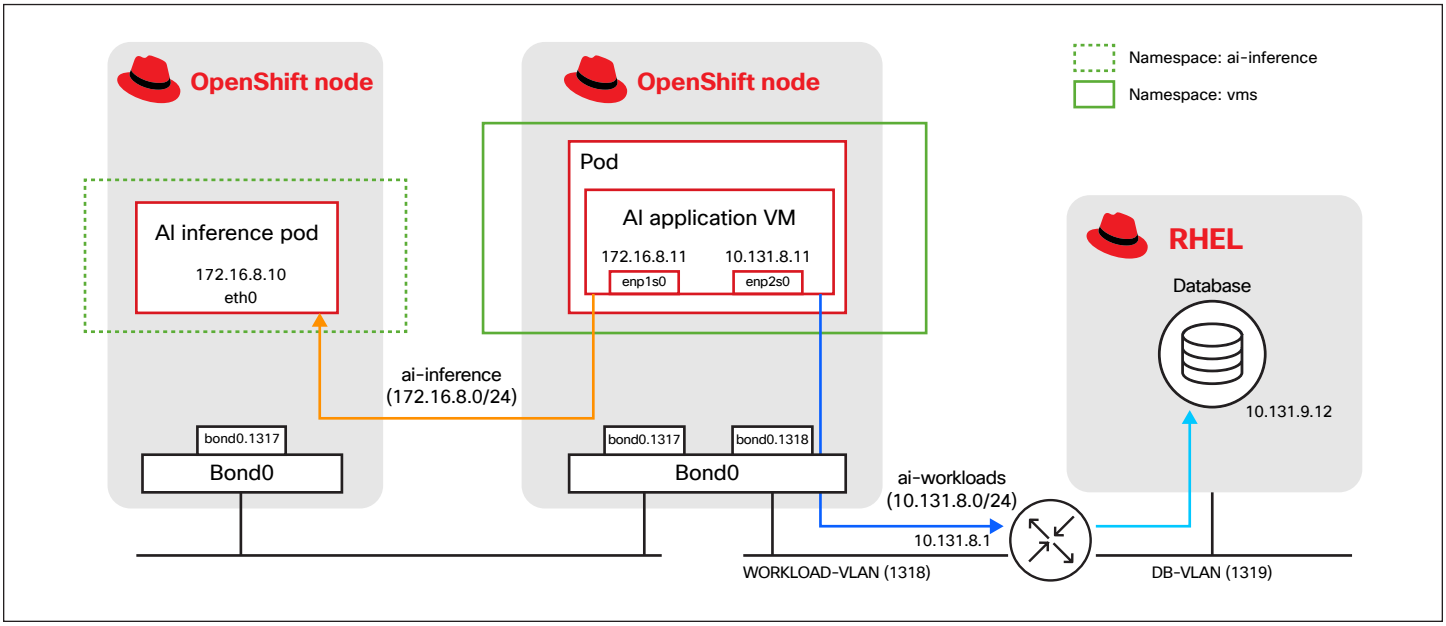


Figure 2. Private Network topology

© 2026 Cisco and/or its affiliates. All rights reserved.

As the topology shows, the VM has an interface on each Private Network, while the inference pod has an interface only on ai-inference. Each Private Network is defined by a ClusterwidePrivateNetwork resource. The namespace selector for ai-inference includes the ai-inference and vms namespaces, so one isolated network spans two namespaces.

Note: The bond0.1317 interface belongs to the OpenShift machine network, which every node uses.

For the Private Network ai-workloads to reach the edge network, it also needs a path out of the cluster through a dedicated interface or VLAN subinterface. The path comes from a different resource, a PrivateNetworkNodeAttachment. In this lab, a PrivateNetworkNodeAttachment binds the Private Network ai-workloads to the VLAN 1318 on the selected OpenShift worker nodes. INV creates and owns bond0.1318 on the eligible node where the VM runs. That interface is where the cluster stops and the edge network starts.

Reaching PostgreSQL also requires the network team to provide:

- VLAN 1318 trunked to every eligible node where the VM may run
- An upstream gateway
- A firewall policy permitting TCP port 5432 from 10.131.8.11/32 to the PostgreSQL server, with return traffic allowed
- A route for 10.131.8.0/24 back toward the cluster

Workload network configuration

A workload joins a Private Network through Kubernetes annotations. The network-attachment annotation names its primary attachment and IP address. If the workload needs another network, secondary-network-attachments lists the secondary attachments.

The inference pod joins only ai-inference, so one annotation is all it needs:

```
# Inference pod
privnet.isovalent.com/network-attachment: '{"network":"ai-inference","ipv4":"172.16.8.10"}'
```

The VM is different because it joins both Private Networks: ai-workloads is the VM's primary attachment, and ai-inference is its secondary attachment. Neither interface uses the cluster's default pod network, and the VM is multi-homed on Private Networks alone. With INV, both interfaces get their connection straight from the Private Networks that they join.

```
privnet.isovalent.com/network-attachment: |-
  {"network":"ai-workloads","ipv4":"10.131.8.11"}
privnet.isovalent.com/secondary-network-attachments: |-
  [{"network":"ai-inference","ipv4":"172.16.8.11"}]
```

The annotations tell INV which networks and IP addresses the VM should use. The VM specification then connects each attachment to a virtual interface. Inside the guest, the inference interface has a route for 172.16.8.0/24 only, so every other destination, including the database, leaves through the ai-workloads interface.

© 2026 Cisco and/or its affiliates. All rights reserved.

The inference path inside the cluster

The application running inside the VM communicates with the inference service at its Private Network address, 172.16.8.10:8000. In the lab, it sent an OpenAI-compatible chat completion request to that address and received the answer generated by the AI model:

```
curl -sS http://172.16.8.10:8000/v1/chat/completions \
  -H 'Content-Type: application/json' \
  -d '{"model":"RedHatAI/Meta-Llama-3.1-8B-Instruct-FP8-dynamic",
       "messages":[{"role":"user","content":"Summarize this claim note in one sentence: Rear bumper damaged;
no injuries reported; vehicle remains drivable."}],
       "max_tokens":64,
       "temperature":0.7,
       "stream":false}' \
  | jq -r '.choices[0].message.content'

A vehicle's rear bumper was damaged in an incident, but fortunately, no one was hurt and the vehicle is
still operational.
```

Hubble, Cilium's observability component, provides visibility into Private Network traffic. Hubble Timescape stores the flow data and provides a graphical interface for viewing it over a selected time range. Figure 3 shows the inference traffic between the VM and the ai-inference pod, including the IP addresses, ports, and forwarding decisions.

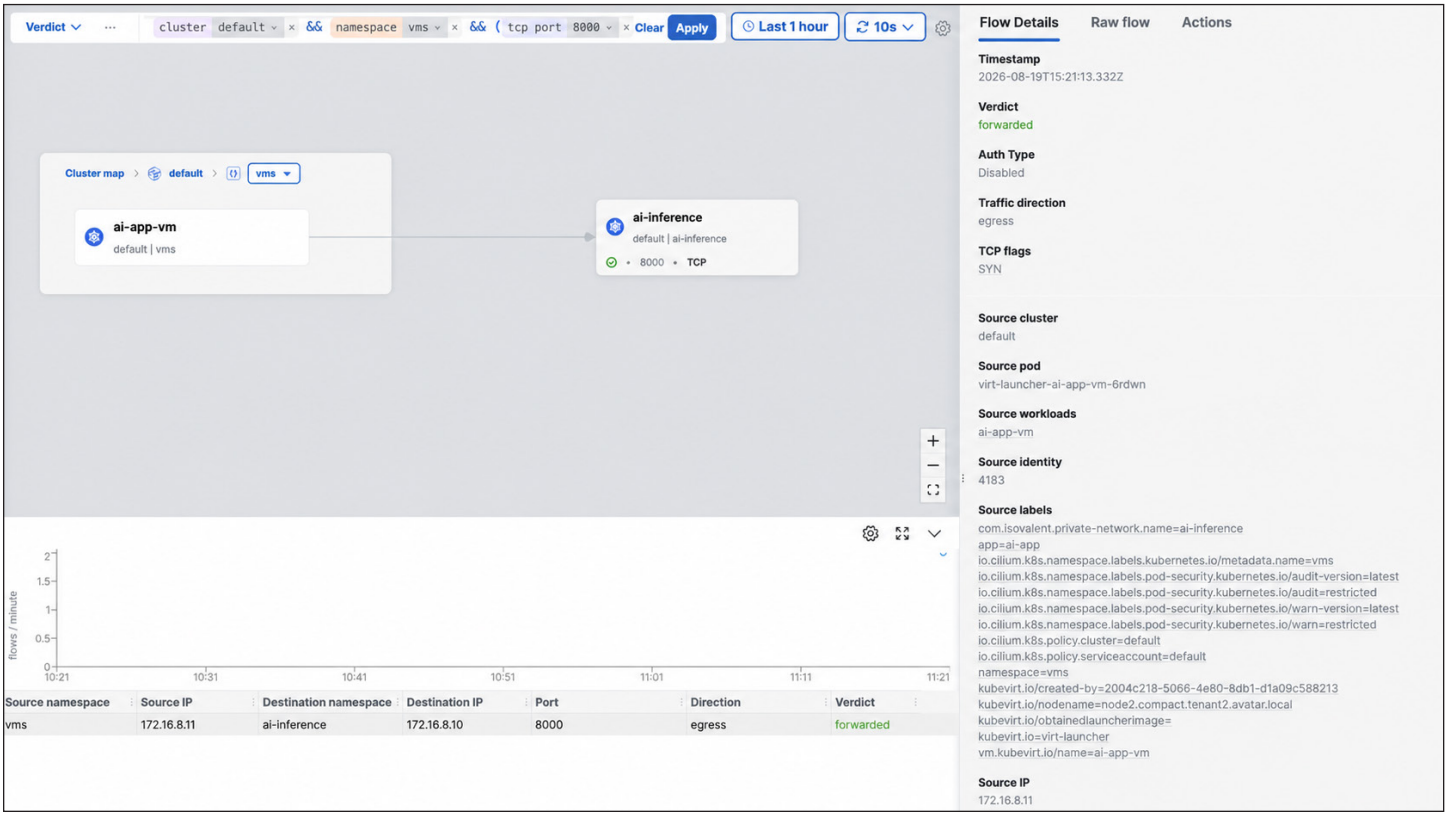


Figure 3. AI inference traffic in Hubble Timescape

© 2026 Cisco and/or its affiliates. All rights reserved.

The external path to the system of record

The next test focused on the second path, from the VM to PostgreSQL outside the cluster. The application opened a database session and asked PostgreSQL to report the client and server addresses:

```
psql "host=10.131.9.12 port=5432 dbname=<database> user=<user> connect_timeout=5" \
  -W -v ON_ERROR_STOP=1 \
  -c "SELECT inet_client_addr() AS client_ip, inet_server_addr() AS server_ip;"

  client_ip  |  server_ip
-------------+-------------
 10.131.8.11 | 10.131.9.12
(1 row)
```

PostgreSQL saw 10.131.8.11, the VM's ai-workloads address, instead of an OpenShift node address. This result shows that local-access preserved the VM's source address across the cluster boundary. Because the VM kept its own address, existing firewall rules, database access entries, and audit records can continue to identify the application.

Security policy

Isovalent Network Policy extends the open-source Cilium Network Policy model with additional enterprise capabilities. It uses IsovalentNetworkPolicy, a Kubernetes resource that tells Cilium which traffic to allow or deny. For a Private Network using local-access, the policy can match an external destination by IP address, transport protocol, and destination port before it leaves the cluster. In this lab, the policy explicitly denied egress traffic to a second, unapproved PostgreSQL server at 10.131.9.13. The PostgreSQL server at 10.131.9.12 remained reachable, even though both addresses are in the same subnet.

In addition to the destination IP address and port, the policy must also identify which of the VM's two network attachments the rule applies to. Both attachments inherit the same Kubernetes label k8s:vm.kubevirt.io/name:ai-app-vm from the VM definition, so that the label alone cannot distinguish the database path from the inference path. INV also gives each attachment a Private Network identity label: cni:com.isovalent.private-network.name. The policy combines two labels to select the VM's database-facing endpoint:

- k8s:vm.kubevirt.io/name: ai-app-vm
- cni:com.isovalent.private-network.name: ai-workloads

© 2026 Cisco and/or its affiliates. All rights reserved.

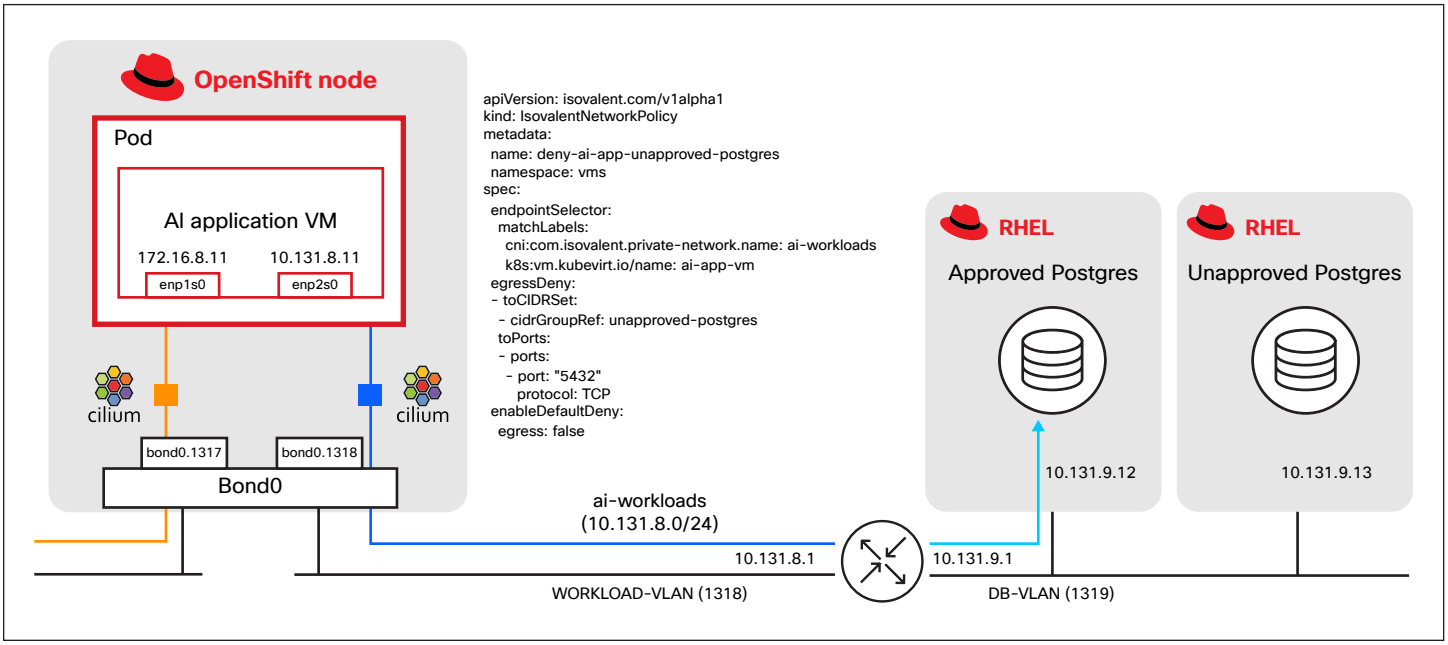


Figure 4. Policy enforcement on the database path

Hubble recorded traffic to both database addresses at the VM's ai-workloads endpoint. Each record included the source and destination IP addresses and the policy verdict, so the flow and the enforcement decision appeared together. Traffic to 10.131.9.12 was forwarded, and traffic to 10.131.9.13 was dropped at the endpoint before it reached the local-access interface.



© 2026 Cisco and/or its affiliates. All rights reserved.

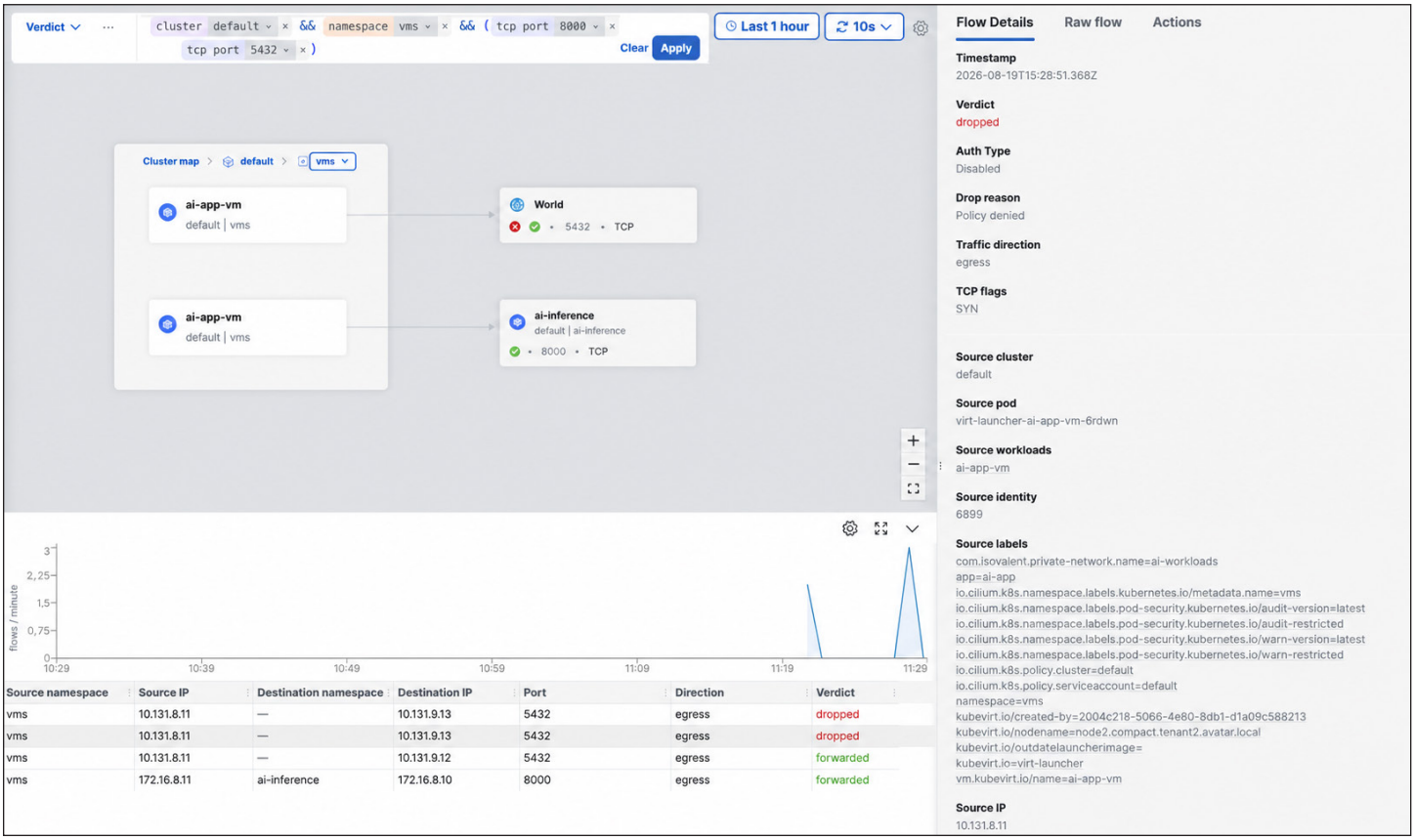


Figure 5. Dropped traffic to the unapproved database

Key findings

This lab deployed Red Hat OpenShift Virtualization with Isovalent Networking for Virtualization on Cisco Unified Edge. The lab used a Cisco UCS XE9305 Chassis with Cisco UCS XE130c M8 Compute Nodes. The lab showed four results:

- Using Isovalent Private Networks, the VM reached the inference service at its Private Network address, and the inference traffic stayed inside the cluster.
- The VM's address was preserved across the cluster boundary instead of being translated to the OpenShift node address.
- Isovalent Network Policy denied traffic to the unapproved database before it left the cluster, while the approved database remained reachable.
- Hubble recorded flows for both paths and policy verdicts for the database path in one place.

© 2026 Cisco and/or its affiliates. All rights reserved.

Conclusion

Bringing VMs and containers onto one platform solves only part of the problem. Their traffic also needs a common networking and policy model. In this lab, Isovalent Networking for Virtualization provided isolated workload paths, stable IP addresses, controlled external access, and shared visibility. This pattern can reduce duplicate controls and separate traffic views as teams add containerized AI services to existing VM applications. The lab provides a starting point for evaluating that approach, with production design and validation left to each site.

References

- Cisco Unified Edge with Red Hat Design Guide
- Isovalent Networking for Virtualization: Enterprise-Grade Network Segmentation and Multi-Tenancy for VMs in Kubernetes
- Isovalent Networking for Virtualization: Migrate VMs Without Breaking the Network
- Isovalent Enterprise Platform Documentation: Isolate Workloads with Private Networks
- Isovalent Enterprise Platform Documentation: Create Multi-Homed Workloads with Multus
- Isovalent Enterprise Platform Documentation: Connect to Network Fabric with local-access
- Cilium Network Policy
- Inspecting Network Flows with the Hubble CLI
- OpenShift Container Platform 4.20: Virtualization Networking
- OpenShift Container Platform 4.20: Use Cases for a Secondary Network

Authors

Shixiong Shang: Technical Marketing Engineer, Cisco

© 2026 Cisco and/or its affiliates. All rights reserved. Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: www.cisco.com/go/trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)