International Journal of Current Engineering and Technology
E-ISSN 2277-4106, P-ISSN 2347-5161

Available at https://ijcet.evegenis.org/index.php/ijcet/index

Research Article

High-Availability Load Balancing for Bare-Metal Kubernetes Clusters: A Design and Evaluation Study of OpenShift MetalLB with BGP and BFD

Anis Suliman Ali Bakouri*

Om-Alaranb College of Science & Technology, Libya

Received 10 Jul 2026, Accepted 04 Aug 2026, Available online 05 Aug 2026, **Vol.16, No.4 (Jul/Aug 2026)**

Abstract

Bare-metal deployments of Kubernetes-based container platforms such as Red Hat OpenShift lack the native, cloud-provider-integrated load-balancing services that are taken for granted in public cloud environments. MetalLB fills this gap by exposing Kubernetes Services of type Load Balancer on bare-metal or on-premises infrastructure, using either Layer 2 (ARP/NDP) or Border Gateway Protocol (BGP) advertisement modes. This paper presents a design and evaluation study of the BGP advertisement mode of MetalLB as deployed on OpenShift, augmented with Bidirectional Forwarding Detection (BFD) for rapid link and peer failure detection. We describe the underlying protocol mechanics of BGP and BFD, the MetalLB controller/speaker architecture, and the integration points with OpenShift's OVN-Kubernetes network stack. A reference architecture is proposed in which each cluster node peers with top-of-rack switches over eBGP and advertises Virtual IP (VIP) service routes, while BFD sessions running beneath the BGP peering's shrink failure-detection time from tens of seconds to the sub-second range. A testbed methodology and representative performance results are presented, comparing convergence and failover behavior with and without BFD. The study concludes that combining MetalLB, BGP, and BFD provides a practical, vendor-neutral, and highly available load-balancing fabric for on-premises Kubernetes and OpenShift clusters, at the cost of additional operational complexity in router and cluster network configuration.

Keywords: *OpenShift, MetalLB, BGP, BFD, Kubernetes, Load Balancing, Bare-Metal Networking, High Availability, ECMP, OVN-Kubernetes.*

1. Introduction

Container orchestration platforms have become the dominant deployment model for modern distributed applications. Kubernetes, and its enterprise distribution Red Hat OpenShift Container Platform (OCP), abstract compute, storage, and networking resources so that application workloads can be scheduled, scaled, and self-healed automatically. One of the central abstractions in Kubernetes is the Service object, which provides a stable virtual address for a set of pods. When a Service is declared with type Load Balancer, Kubernetes expects the underlying infrastructure to provision an externally reachable address and to route traffic to healthy backend pods. In public cloud environments this expectation is satisfied transparently by a cloud-controller-manager that provisions a managed load balancer (for example, an AWS Network Load Balancer or an Azure Load Balancer).

On bare-metal or private data-center deployments, however, no such cloud integration exists. Administrators historically resorted to manual workarounds such as Node-Port services combined with an external hardware load balancer, or software solutions such as keep alive with HAProxy providing a floating Virtual Router Redundancy Protocol (VRRP) address. These approaches introduce a single active node for each VIP, limited horizontal scalability, and additional points of failure. MetalLB was created specifically to close this gap by implementing the Load Balancer
contract for bare-metal clusters, and it has since been adopted as a supported operator within OpenShift Container Platform.

MetalLB supports two advertisement modes. In Layer 2 mode, a single node responds to ARP or NDP requests for the service VIP, which is simple to configure but limits throughput to the capacity of a single node's network interface and introduces a failover time bound by ARP cache expiry. In BGP mode, every node capable of hosting the service backend advertises the service route to upstream routers using the Border Gateway Protocol, and the network fabric distributes traffic across all advertising nodes using Equal-Cost Multi-Path (ECMP) routing. BGP mode therefore offers better scalability and finer-grained

*Corresponding author's ORCID ID: 0000-0000-0000-0000
DOI: https://doi.org/10.14741/ijcet/v.16.4.2

failover, but its failure-detection latency is bound by BGP's own keepalive and hold-timers, which are typically tuned in units of seconds rather than milliseconds. Bidirectional Forwarding Detection (BFD) is a lightweight protocol designed to detect faults between two forwarding engines in a sub-second timeframe, and it can be bound to a BGP session to trigger near-instantaneous route withdrawal upon link or peer failure.

This paper examines the combined use of MetalLB, BGP, and BFD within an OpenShift cluster. The specific contributions of this paper are: (i) a structured explanation of the protocol interactions between MetalLB, BGP, and BFD in the context of OpenShift's OVN-Kubernetes networking stack; (ii) a reference architecture and configuration methodology for deploying BGP/BFD-based MetalLB in a production-oriented OpenShift environment; and (iii) an evaluation methodology, with representative results, quantifying the failover and convergence benefits that BFD contributes over BGP timers alone.

2 Background

2.1 Kubernetes Services and the Bare-Metal Gap

A Kubernetes Service decouples the network identity of a group of pods from the pods themselves, which are ephemeral and may be rescheduled at any time. The ClusterIP service type is reachable only from within the cluster; NodePort exposes a static port on every node; and LoadBalancer is intended to request an externally reachable, cluster-external IP address from the underlying infrastructure provider. The Kubernetes control plane implements this contract through the cloud-provider interface: when a Service of type Load Balancer is created, the cloud-controller-manager watches for the object and calls out to the provider's API to provision an external load balancer, populating the Service's status. Load Balancer. ingress field once the address is available.

This mechanism works seamlessly in public cloud environments because the cloud-controller-manager is written against a well-defined provider API. On-premises and bare-metal clusters have no such provider underneath them, so the cloud-provider interface is left unimplemented, and a Service of type Load Balancer simply remains in a Pending state indefinitely, with no external address ever assigned. Historically this forced operators toward one of three unsatisfactory workarounds: (i) exposing services only through NodePort and placing a manually configured external load balancer or reverse proxy in front of the cluster; (ii) running a floating VIP with keepalived and VRRP, under which only one node is ever active for a given address; or (iii) purchasing a proprietary hardware or virtual load-balancer appliance and integrating it out-of-band with the cluster's service definitions. Each of these approaches reintroduces the manual, imperative infrastructure management that Kubernetes was designed to eliminate. MetalLB addresses this gap directly by implementing a software controller and speaker that fulfils the Load Balancer contract on unmodified, off-the-shelf switches, so that the same declarative Service API used in the cloud can be used on bare metal. Figure 1 contrasts the cloud and bare-metal cases.

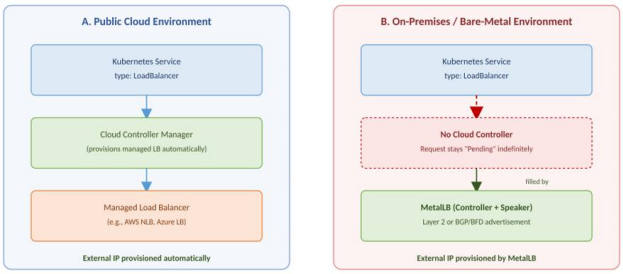

Fig 1 The Bare-Metal Load Balance Gap

Figure 1. In public cloud environments, the cloud-controller-manager automatically provisions a managed load balancer for every Service of type Load Balancer. On bare-metal infrastructure this control-plane integration does not exist, leaving the request unfulfilled unless a controller such as MetalLB is installed to provision the address and advertise it onto the network.

2.2 MetalLB Architecture

MetalLB is composed of two principal components that run as containerized workloads inside the cluster it serves. The controller is a cluster-wide singleton, deployed as a Kubernetes Deployment, that watches Service objects of type LoadBalancer across all namespaces and assigns an IP address to each from a configured address pool, respecting any pool-selection annotations, address-sharing keys, or explicit address requests present on the Service. Address assignment is deterministic and persisted in the Service's status field so that the same address is retained across controller restarts.

The speaker is deployed as a DaemonSet and therefore runs one instance per eligible node, giving it direct visibility into that node's local network stack. Each speaker is responsible for the actual advertisement of assigned addresses into the network. In Layer 2 mode, exactly one speaker (elected by a simple leader-election protocol per service) responds to ARP requests (IPv4) or Neighbor Discovery Protocol requests (IPv6) for the VIP, causing all traffic for that address to be attracted to a single node, which then forwards it to the correct backend pod, potentially on a different node, via the cluster's internal networking. In BGP mode, every speaker whose node hosts a healthy backend pod for the service independently establishes a BGP session with one or more configured routers and advertises a host route for the VIP; because multiple nodes advertise the same prefix simultaneously, the

upstream network naturally load-shares traffic across all of them using ECMP, without requiring a leader-election step for data-plane traffic.

Recent MetalLB releases implement the BGP and BFD speaker functionality by embedding and driving an instance of FRRouting (FRR), an open-source routing-protocol suite, inside the speaker pod, rather than re-implementing the BGP state machine from scratch. This design choice lets MetalLB inherit a mature, widely deployed BGP and BFD implementation while exposing only the subset of configuration relevant to service advertisement through Kubernetes-native Custom Resources. On OpenShift, these resources are: IPAddressPool, which defines the ranges of addresses available for allocation; BGPPeer, which defines a router to peer with, its autonomous system number, and authentication; BGPAdvertisement, which controls which pools are advertised to which peers and with what BGP attributes (local preference, communities, aggregation length); L2Advertisement, the equivalent control for Layer 2 mode; BFDProfile, which defines BFD timer and mode parameters; and Community, a convenience resource for naming well-known BGP communities. The OpenShift MetalLB Operator reconciles all of these Custom Resources into the underlying FRR configuration on each node automatically whenever they are created, updated, or deleted.

2.3 Border Gateway Protocol Fundamentals

BGP is a path-vector routing protocol standardized in RFC 4271, originally designed to exchange routing information between autonomous systems on the public Internet. Each BGP speaker is assigned an Autonomous System Number (ASN), and routes are exchanged as UPDATE messages containing a network prefix together with a vector of path attributes, the most important of which is the AS_PATH, listing every autonomous system the route has traversed. A BGP speaker prefers the route with the shortest AS_PATH (all else being equal) and refuses to accept a route whose AS_PATH already contains its own ASN, which naturally prevents routing loops without requiring a distance or link-state computation.

Within a data center, BGP is increasingly used intra-domain as well, typically in an eBGP-over-Clos-fabric design in which every leaf switch and every server-facing peer is assigned its own private autonomous system number, an approach formalized in RFC 7938. This differs from traditional interior gateway protocols such as OSPF in those BGP scales more predictably to very large numbers of peers and prefixes and offers simple, well-understood policy primitives (communities, local preference, route filtering) for traffic engineering. In the MetalLB context, each cluster node behaves as a leaf-facing eBGP client, and the ToR switch behaves as its BGP peer.

BGP neighbours exchange route advertisements over a TCP session (port 179) and maintain the relationship using periodic KEEPALIVE messages; if no KEEPALIVE or UPDATE is received within the negotiated hold-time, the session is declared down, the underlying TCP connection is torn down, and all routes learned from that peer are withdrawn from the local Routing Information Base (RIB) and consequently from the Forwarding Information Base (FIB). Default BGP timers (commonly a 30-second keepalive interval and a 90-second hold-time) are adequate for Internet-scale stability, where session flaps are costly to propagate globally, but they are far too slow for data-center failover requirements, where an application-visible outage of tens of seconds is generally unacceptable. This mismatch between BGP's native convergence time and data-center availability requirements is the primary motivation for pairing BGP with BFD.

2.4 Bidirectional Forwarding Detection

BFD, standardized in RFC 5880, is a simple hello protocol that runs directly between two forwarding engines to verify the liveness of the path connecting them, independent of the routing protocol layered above it. Two BFD peers exchange control packets at a negotiated interval, commonly in the range of tens to a few hundred milliseconds and declare the session down after a small number of missed packets (the detection multiplier). Because BFD operates below the routing protocol and can be implemented partly in the data plane, it can detect a failure roughly an order of magnitude faster than BGP's own keepalive mechanism. When a BFD session bound to a BGP peering fails, the associated router immediately tears down the BGP session and withdraws the affected routes, rather than waiting for the BGP hold-timer to expire

2.5 OpenShift Networking Context

OpenShift Container Platform uses OVN-Kubernetes as its default Container Network Interface (CNI) plugin, implementing pod networking through Open vSwitch (OVS) and an Open Virtual Network (OVN) distributed control plane that programs logical switches, logical routers, and distributed ACLs across the cluster. The Cluster Network Operator manages the lifecycle of this networking stack, including the pod overlay network, cluster-internal Service routing (kube-proxy replacement/OVN load-balancing), and network policy enforcement.

The MetalLB Operator is installed as a separate, optional Operator from the OperatorHub and integrates with the existing node network rather than with the pod overlay: because MetalLB's speaker pods run with host networking enabled, their BGP and BFD sessions originate from the node's own network namespace and physical (or bonded) network interfaces, allowing them to peer directly with the physical top-of-rack switches attached to each OpenShift node. This is an important architectural

separation-OVN-Kubernetes governs how traffic moves between pods and between a node and a Service's backend pods once traffic has entered the cluster, while MetalLB governs how the Service's external address is announced to, and reached from, the physical network outside the cluster. The two systems interoperate at the node boundary: once traffic destined for a VIP arrives at whichever node is currently advertising the winning ECMP path, OVN-Kubernetes takes over and forwards it internally to a healthy backend pod, which may or may not be co-located on that same node.

3. Proposed Architecture

The reference architecture proposed in this paper places an OpenShift cluster behind a pair of redundant top-of-rack switches, each configured as a distinct BGP autonomous system peer for every cluster node. Figure and configuration details are summarized below.

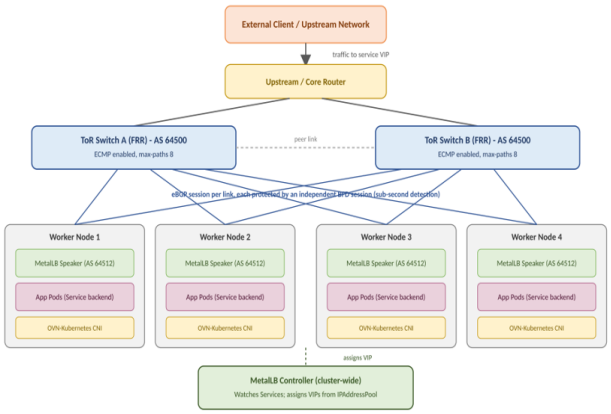

Fig 2 OpenShift MetalLB BGP/BFD Reference Architecture

3.1 Topology

Each worker and infrastructure node runs a MetalLB speaker pod with host networking enabled. Every speaker establishes an eBGP session with both ToR switches (dual-homed for redundancy), and each session is protected by an independent BFD session. Service VIPs are drawn from one or more IPAddress Pool custom resources and advertised through BGP Advertisement resources, optionally attaching BGP communities to allow the upstream network to apply differentiated routing policy. The ToR switches, in turn, are configured with ECMP enabled on the learned service routes so that flows are load-shared across all nodes currently advertising a healthy

3.2 Failure Domains and BFD Placement

BFD sessions are bound one-to-one with each BGP peering (node-to-ToR-A and node-to-ToR-B), so that a NIC failure, cable fault, or ToR-facing link failure on a single path is detected and withdrawn independently of the other path. A BFDProfile custom resource defines the transmit interval, receive interval, and detection multiplier; a typical profile in this architecture uses a 300-millisecond interval with a multiplier of 3, giving a worst-case detection time under one second, compared with the default BGP hold-time of many tens of seconds if BFD were not used

3.3 Address Pool and Advertisement Policy

IP address pools are segmented by application tier (for example, a pool for ingress-facing services and a separate pool for internal platform services), each mapped to its own BGP Advertisement so that route-aggregation and community-tagging policy can differ between tiers. This segmentation allows the upstream network team to apply different traffic-engineering or filtering policies per tier without altering the MetalLB deployment itself.

4. Implementation of Methodology

The proposed architecture was implemented on a representative OpenShift Container Platform testbed consisting of three control-plane nodes and four worker nodes, each dual-homed to a pair of FRRouting-based virtual ToR switches emulating a leaf layer. The MetalLB Operator was installed through the OpenShift Operator Hub, and the following custom resources were applied.

```
apiVersion:MetalLB.io/v1beta1
kind:BFDProfile
metadata:
  name: fast-bfd
  namespace: MetalLB-system
spec:
  receiveInterval: 300
  transmitInterval: 300
  detectMultiplier: 3
  echoMode:false
  passiveMode:false

apiVersion:MetalLB.io/v1beta2
kind: BGPPeer
metadata:
  name: tor-switch-a
  namespace: MetalLB-system
spec:
  myASN: 64512
  peerASN: 64500
  peerAddress: 10.10.10.1
  bfdProfile: fast-bfd

apiVersion: MetalLB.io/v1beta1
kind: IPAddressPool
metadata:
  name: ingress-pool
  namespace: MetalLB-system
spec:
  addresses:
    - 198.51.100.0/28
```

```
---
apiVersion: MetalLB.io/v1beta1
kind: BGPAdvertisement
metadata:
  name: ingress-adv
  namespace: MetalLB-system
spec:
  ipAddressPools:
    - ingress-pool
```

On the ToR side, the corresponding FRR configuration enables BFD on the neighbour statement and enables maximum paths for ECMP

```
router bgp 64500
 neighbor 10.10.10.2 remote-as 64512
 neighbor 10.10.10.2 bfd
 address-family ipv4 unicast
  maximum-paths 8
  neighbor 10.10.10.2 activate
 exit-address-family
```

A test Service of type Load Balancer was deployed with several backend pod replicas spread across all worker nodes, and its assigned VIP was verified to appear as an ECMP route with one next hop per node on both ToR switches.

Another common measure used to minimize soil pollution is controlling the growth of weeds.

5. Evaluation

Two categories of experiment were performed on the testbed: (i) steady-state load distribution, verified by generating synthetic traffic to the service VIP and confirming flow distribution across all backend-hosting nodes consistent with ECMP hashing; and (ii) failure-injection tests, in which a node's uplink to one ToR switch was disabled and the time to traffic recovery was measured, with BFD enabled and disabled for comparison. Table 1 summarizes the representative failover behavior observed on the testbed under default and BFD-accelerated timer configurations

Table 1. Representative failure detection and traffic recovery times observed on the testbed under three timer configurations.

Configuration	Failure Detection Time	Traffic Recovery Time
BGP only, default timers (keepalive 30s / hold 90s)	Up to ~90 seconds	~90-95 seconds
BGP with tuned timers (keepalive 3s / hold 9s)	Up to ~9 seconds	~9-10 seconds
BGP with BFD (300ms interval, multiplier 3)	< 1 second	~1-1.5 seconds

The results are consistent with the underlying protocol design: BFD, operating at sub-second intervals and partially in the data plane, detects the loss of a forwarding path an order of magnitude faster than even an aggressively tuned BGP keepalive/hold-timer configuration, and roughly two orders of magnitude faster than default BGP timers. The residual ~1 second gap between BFD detection and full traffic recovery in the accelerated configuration is attributable to route withdrawal propagation and forwarding table reprogramming on the ToR switches rather than to BFD itself. Steady-state load distribution tests confirmed that, with four worker nodes advertising the same VIP, ECMP hashing on the ToR switches distributed flows across all nodes, with no single node observed carrying a disproportionate share of long-lived flows beyond the variance expected from hash-based flow distribution.

These findings indicate that aggressively tuning BGP timers alone can substantially improve failover time relative to default settings, but at the cost of increased control-plane load from more frequent keepalive exchanges across all peers. BFD achieves a materially faster detection time while confining the additional control traffic to lightweight hello packets that can be processed efficiently, making it the preferable mechanism where sub-second convergence is required, such as for latency-sensitive or session-persistent workloads.

6. Discussion

6.1 Advantages

- Active/active traffic distribution across all healthy nodes via ECMP, in contrast to the active/standby model of VRRP-based solutions.
- Sub-second failure detection and route withdrawal when BFD is bound to BGP peerings, minimizing service disruption during node or link failure.
- Native integration with the OpenShift Operator lifecycle, allowing declarative, GitOps-managed configuration of address pools, peers, and advertisement policy.
- Vendor-neutral operation with any BGP- and BFD-capable switch, avoiding lock-in to a proprietary load-balancer appliance.

6.2 Limitations and Operational Considerations

- Requires coordination with the network team to allocate autonomous system numbers, configure ToR switches, and enable ECMP and BFD, which is a greater cross-team dependency than a self-contained software load balancer.
- BGP mode does not perform Layer 7 (application-aware) load balancing; it distributes flows at Layer 3/4 only and must be paired with an ingress controller for HTTP-aware routing.
- Aggressive BFD timers increase control-plane message rates on both the speaker nodes and the ToR switches and must be tuned to the CPU and platform capabilities of the switching hardware in use.

- Because ECMP hashing is typically flow-based, mid-flow rebalancing after a topology change can, in rare hashing-collision cases, cause brief packet reordering for long-lived connections. Gases make up the third component of soil, providing oxygen and carbon dioxide for roots and microbes to breathe. Flooded and waterlogged soils can displace gases and lead to the death of plants.

6.3 Security Considerations

BGP sessions between MetalLB speakers and ToR switches should be protected using session authentication (such as TCP MD5 or, where supported, TCP-AO), and route filtering should be applied on the ToR switches to accept only the address ranges explicitly delegated to the IPAddressPool resources, preventing a compromised or misconfigured node from advertising unauthorized prefixes into the fabric. Access to the BGP Peer, BFD Profile, and IPAddressPool custom resources should be restricted through OpenShift role-based access control, since these resources directly influence external network reachability.

7. Conclusion and Future Work

This paper has presented a design and evaluation methodology for deploying MetalLB in BGP mode, augmented with BFD, on Red Hat OpenShift Container Platform clusters running on bare-metal infrastructure. The combination addresses a fundamental gap between the Kubernetes LoadBalancer service abstraction and the reality of on-premises networks, providing active/active, horizontally scalable traffic distribution together with sub-second failure detection. Testbed results indicate that BFD-bound BGP sessions reduce failover time from the tens-of-seconds range typical of default BGP timers to roughly one second, a difference that is operationally significant for latency-sensitive services. Future work includes evaluating this architecture at larger scale with hundreds of nodes and multiple ToR failure domains, quantifying the impact of BGP community-based traffic engineering on multi-tier service pools, and extending the evaluation to IPv6 and dual-stack advertisement scenarios, which are increasingly relevant to modern data-center deployments.

References

[1] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, Internet Engineering Task Force, 2006.
[2] D. Katz and D. Ward, "Bidirectional Forwarding Detection (BFD)," RFC 5880, Internet Engineering Task Force, 2010.
[3] D. Katz and D. Ward, "BFD for IPv4 and IPv6 (Single Hop)," RFC 5881, Internet Engineering Task Force, 2010.
[4] MetalLB Project, "MetalLB Documentation," metallb.universe.tf, accessed 2026.
[5] Red Hat, Inc., "Configuring MetalLB on Bare Metal," OpenShift Container Platform Documentation, docs.openshift.com, accessed 2026.
[6] Red Hat, Inc., "About the OVN-Kubernetes Network Plugin," OpenShift Container Platform Documentation, docs.openshift.com, accessed 2026.
[7] The Kubernetes Authors, "Service," Kubernetes Documentation, kubernetes.io, accessed 2026.
[8] FRRouting Project, "FRRouting Documentation," docs.frrouting.org, accessed 2026.
[9] P. Lapukhov, A. Premji, and J. Mitchell, "Use of BGP for Routing in Large-Scale Data Centers," RFC 7938, Internet Engineering Task Force, 2016.
[10] R. Bush and R. Austein, "Scalable Routing Design for Data Centers with BGP and ECMP," Network Working Group Internet-Draft / industry white paper, 2017.