GCP Dual-Stack VPN Solution

HA VPN with IPv4 and IPv6 BGP Sessions

Technical Implementation Guide

January 2026

Contents

1 Executive Summary

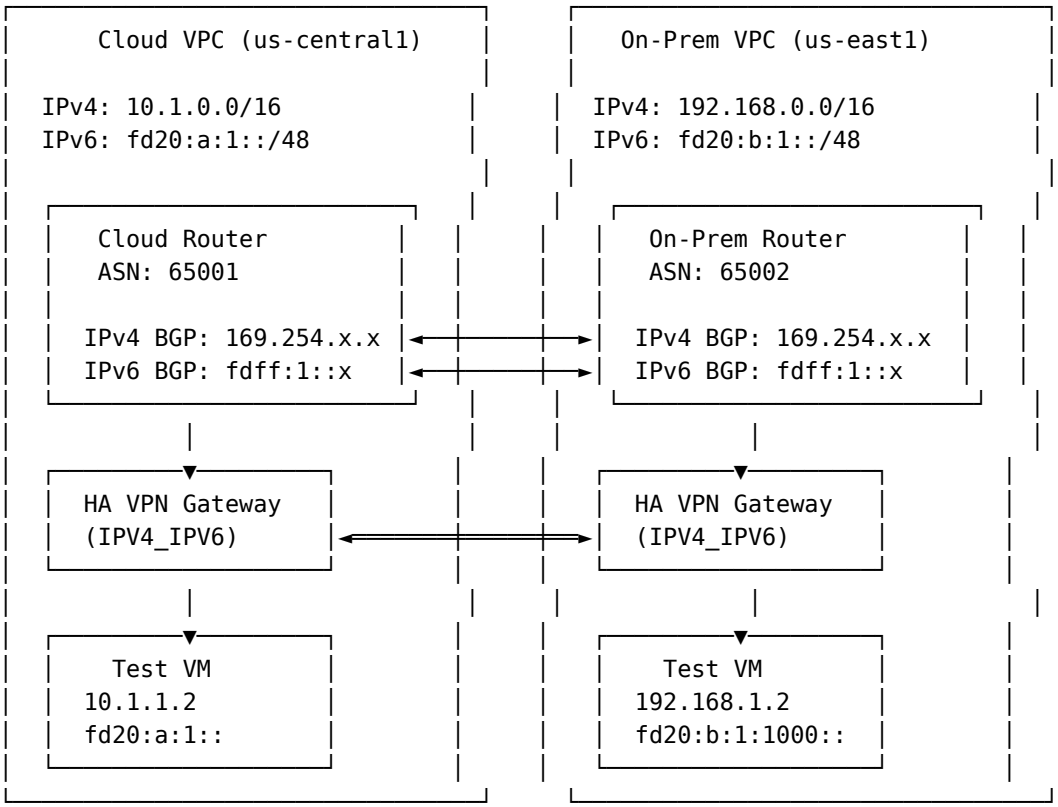
This document describes the implementation of a dual-stack (IPv4 + IPv6) Site-to-Site VPN solution using Google Cloud Platform's HA VPN service. The solution enables full IPv4 and IPv6 connectivity between two VPC networks using BGP dynamic routing.

Capability	**IPv4**	**IPv6**
VPN Tunnels	✓	✓
BGP Sessions	✓	✓
Route Exchange	✓	✓
Cross-VPN Ping	32ms	33ms

Key Achievement: Full dual-stack connectivity with automatic route propagation via dedicated IPv4 and IPv6 BGP sessions.

2 Architecture Overview

2.1 Network Topology



Listing 1: Network Architecture Diagram

2.2 Key Components

Component	**Description**
HA VPN Gateway	Regional resource with `stack_type = "IPV4_IPV6"` for dual-stack tunnel support
Cloud Router	BGP router with separate IPv4 and IPv6 peering sessions
VPN Tunnels	4 tunnels total (2 per direction) for 99.99% SLA
BGP Sessions	8 total: 4 IPv4 sessions + 4 IPv6 sessions

3 Critical Design Decision

3.1 Why Dedicated IPv6 BGP Sessions?

Important: MP-BGP (Multiprotocol BGP) on IPv4 sessions does NOT work for IPv6 route installation when using GCP Cloud Routers on both ends.

Why this is GCP-specific:

GCP Cloud Router does not allow manual manipulation of the BGP next-hop attribute. When MP-BGP advertises IPv6 routes over an IPv4 session, the next-hop remains the IPv4 peering address (e.g., `169.254.x.x`), which cannot be used to forward IPv6 traffic.

With traditional routers (Cisco, Juniper, etc.), you can use MP-BGP with IPv4 transport and manually set the IPv6 next-hop using route-maps or policies. This allows IPv6 routes to be advertised with a valid IPv6 next-hop address, making them installable.

What doesn't work (GCP-to-GCP):

- Setting `enable_ipv6 = true` on IPv4 BGP peers
- IPv6 routes are advertised but NOT installed into VPC routing table
- Routes appear in `bestRoutes` but with IPv4 next-hops (unusable)

What works (GCP-to-GCP):

- Dedicated IPv6 BGP sessions with IPv6 peering addresses
- Using the reserved `fdff:1::/64` ULA range for BGP peering
- /126 subnets for point-to-point links

Alternative (GCP-to-physical router):

- MP-BGP on IPv4 session may work if the remote router can set the IPv6 next-hop
- Requires route-map/policy on the non-GCP side to rewrite next-hop

3.2 BGP Session Configuration

Session	Cloud IP	On-Prem IP	Routes
IPv4 Tunnel 0	`169.254.0.1/30`	`169.254.0.2/30`	IPv4 only
IPv4 Tunnel 1	`169.254.1.1/30`	`169.254.1.2/30`	IPv4 only
IPv6 Tunnel 0	`fdff:1::1/126`	`fdff:1::2/126`	IPv6 only
IPv6 Tunnel 1	`fdff:1::5/126`	`fdff:1::6/126`	IPv6 only

Table 1: BGP Peering Address Allocation

4 Terraform Implementation

4.1 Project Structure

```
terraform-gcp/
├── main.tf              # All resources
├── terraform.tfvars     # Configuration values
└── README.md            # Documentation
```

4.2 Key Resource Definitions

4.2.1 Local Variables

```
locals {
  # VPC Address Ranges
  cloud_ipv4_cidr  = "10.1.0.0/16"
```

```
  cloud_ipv6_cidr  = "fd20:a:1::/48"
  onprem_ipv4_cidr = "192.168.0.0/16"
  onprem_ipv6_cidr = "fd20:b:1::/48"

  # BGP ASNs
  cloud_asn  = 65001
  onprem_asn = 65002

  # IPv6 BGP Peering Addresses (fdff:1::/64 ULA range)
  bgp_v6_cloud_0  = "fdff:1::1/126"
  bgp_v6_onprem_0 = "fdff:1::2/126"
  bgp_v6_cloud_1  = "fdff:1::5/126"
  bgp_v6_onprem_1 = "fdff:1::6/126"
}
```

4.2.2 VPC Networks with IPv6

```
resource "google_compute_network" "cloud" {
  name                     = "vpc-cloud-prod"
  auto_create_subnetworks  = false
  routing_mode             = "GLOBAL"
  enable_ula_internal_ipv6 = true
  internal_ipv6_range      = "fd20:a:1::/48"
}

resource "google_compute_subnetwork" "cloud_workload" {
  name             = "subnet-workload-prod"
  ip_cidr_range    = "10.1.1.0/24"
  region           = "us-central1"
  network          = google_compute_network.cloud.id
  stack_type       = "IPV4_IPV6"
  ipv6_access_type = "INTERNAL"
}
```

4.2.3 HA VPN Gateway (Dual-Stack)

```
resource "google_compute_ha_vpn_gateway" "cloud" {
  name       = "vpngw-cloud-prod"
  region     = "us-central1"
  network    = google_compute_network.cloud.id
  stack_type = "IPV4_IPV6"  # Critical for dual-stack
}
```

4.2.4 IPv4 BGP Session

```
resource "google_compute_router_interface" "cloud_interface_0" {
  name       = "interface-cloud-0"
  router     = google_compute_router.cloud.name
  region     = "us-central1"
  ip_range   = "169.254.0.1/30"
  vpn_tunnel = google_compute_vpn_tunnel.cloud_to_onprem_0.name
}

resource "google_compute_router_peer" "cloud_peer_0" {
  name            = "peer-cloud-to-onprem-0"
  router          = google_compute_router.cloud.name
  region          = "us-central1"
  peer_ip_address = "169.254.0.2"
  peer_asn        = 65002
  interface       = google_compute_router_interface.cloud_interface_0.name
```

```
  # Note: NO enable_ipv6 here - IPv4 only
}
```

4.2.5 IPv6 BGP Session (The Key!)

```
resource "google_compute_router_interface" "cloud_interface_0_v6" {
  name       = "interface-cloud-0-v6"
  router     = google_compute_router.cloud.name
  region     = "us-central1"
  ip_range   = "fdff:1::1/126"  # IPv6 ULA address
  vpn_tunnel = google_compute_vpn_tunnel.cloud_to_onprem_0.name
}

resource "google_compute_router_peer" "cloud_peer_0_v6" {
  name            = "peer-cloud-to-onprem-0-v6"
  router          = google_compute_router.cloud.name
  region          = "us-central1"
  peer_ip_address = "fdff:1::2"  # IPv6 peer address
  peer_asn        = 65002
  interface       = google_compute_router_interface.cloud_interface_0_v6.name
  enable_ipv6     = true  # Enable IPv6 route exchange
}
```

4.2.6 Firewall Rules (Separate IPv4/IPv6)

```
# IPv4 firewall rule
resource "google_compute_firewall" "cloud_allow_internal" {
  name    = "fw-cloud-allow-internal"
  network = google_compute_network.cloud.id
  allow { protocol = "icmp" }
  allow { protocol = "tcp" }
  allow { protocol = "udp" }
  source_ranges = ["10.1.0.0/16", "192.168.0.0/16"]
}

# IPv6 firewall rule (MUST be separate)
resource "google_compute_firewall" "cloud_allow_internal_ipv6" {
  name    = "fw-cloud-allow-internal-ipv6"
  network = google_compute_network.cloud.id
  allow { protocol = "58" }  # ICMPv6
  allow { protocol = "tcp" }
  allow { protocol = "udp" }
  source_ranges = ["fd20:a:1::/48", "fd20:b:1::/48"]
}
```

GCP Limitation: Firewall rules cannot mix IPv4 and IPv6 in the same `source_ranges`. Create separate rules for each address family.

5 Deployment

5.1 Prerequisites

- GCP Project with billing enabled
- Compute Engine API enabled
- Terraform >= 1.5.0
- gcloud CLI authenticated

5.2 Deployment Commands

```
# Initialize Terraform
cd terraform-gcp
terraform init

# Create terraform.tfvars
cat > terraform.tfvars << EOF
project_id        = "your-project-id"
region            = "us-central1"
onprem_region     = "us-east1"
environment       = "prod"
vpn_shared_secret = "YourSecureSharedSecret123!"
EOF

# Plan and apply
terraform plan -out=tfplan
terraform apply tfplan
```

5.3 Deployment Time

Resource	Time
VPC Networks	20 seconds
HA VPN Gateways	15 seconds
VPN Tunnels	15 seconds
BGP Sessions	30 seconds
Total	**3 minutes**

6 Verification

6.1 Check BGP Session Status

```
gcloud compute routers get-status router-cloud-prod \
  --region=us-central1 \
  --format="yaml(result.bgpPeerStatus[].name,result.bgpPeerStatus[].state)"
```

Expected Output:

```
result:
  bgpPeerStatus:
  - name: peer-cloud-to-onprem-0
    state: Established
  - name: peer-cloud-to-onprem-0-v6
    state: Established
  - name: peer-cloud-to-onprem-1
    state: Established
  - name: peer-cloud-to-onprem-1-v6
    state: Established
```

6.2 Verify Route Installation

```
gcloud compute routers get-status router-cloud-prod \
  --region=us-central1 \
  --format="yaml(result.bestRoutes)" | grep -E "destRange|nextHopIp"
```

Expected Output (showing IPv6 routes with IPv6 next-hops):

```
destRange: fd20:b:1::/48
nextHopIp: fdff:1::2        # IPv6 next-hop!
```

```
destRange: fd20:b:1:1000::/64
nextHopIp: fdff:1::2
```

6.3 Test Connectivity

```
# SSH to cloud VM
gcloud compute ssh vm-cloud-test-prod --zone=us-central1-a

# Test IPv4
ping -c 3 192.168.1.2

# Test IPv6
ping6 -c 3 fd20:b:1:1000::
```

Expected Results:

```
PING 192.168.1.2: 3 packets transmitted, 3 received, 0% packet loss
rtt min/avg/max = 32.2/32.5/32.8 ms

PING fd20:b:1:1000::: 3 packets transmitted, 3 received, 0% packet loss
rtt min/avg/max = 32.5/33.3/34.7 ms
```

7 Troubleshooting

7.1 Common Issues

Symptom	Solution
IPv6 BGP session DOWN	Verify /126 subnets don't overlap. Each tunnel needs unique /126 range.
IPv6 routes not installed	Don't use MP-BGP on IPv4 sessions. Create dedicated IPv6 BGP sessions.
Firewall blocking traffic	Create separate IPv4 and IPv6 firewall rules. Can't mix address families.
Ping6 100% packet loss	Check `bestRoutes` for IPv6 next-hops. Must be `fdff:1::x`, not `169.254.x.x`.

7.2 Diagnostic Commands

```
# Check all routes in router
gcloud compute routers get-status router-cloud-prod \
  --region=us-central1 --format=json | jq '.result.bestRoutes[]'

# Check VPC routes
gcloud compute routes list --filter="network:vpc-cloud-prod"

# Check firewall rules
gcloud compute firewall-rules list --filter="network:vpc-cloud-prod"
```

8 Comparison: Azure vs GCP

Capability	Azure VWAN	GCP HA VPN
IPv6 in VPN config	×	✓
IPv6 BGP sessions	×	✓
IPv6 route installation	×	✓
Cross-VPN IPv4	× (VM issue)	✓ 32ms

Cross-VPN IPv6	×	✓ 33ms
Deployment time	30-45 min	3 min

Conclusion: GCP HA VPN provides full dual-stack support when using dedicated IPv6 BGP sessions. Azure Virtual WAN does not support IPv6 over S2S VPN.

9 Cleanup

To destroy all resources:

```
cd terraform-gcp
terraform destroy
```

Document generated from successful GCP dual-stack VPN implementation

Project: dual-stack-vpn-test | Regions: us-central1, us-east1