# 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 connecting to an on-premises simulation environment using LibreSwan IPsec and FRR BGP routing.

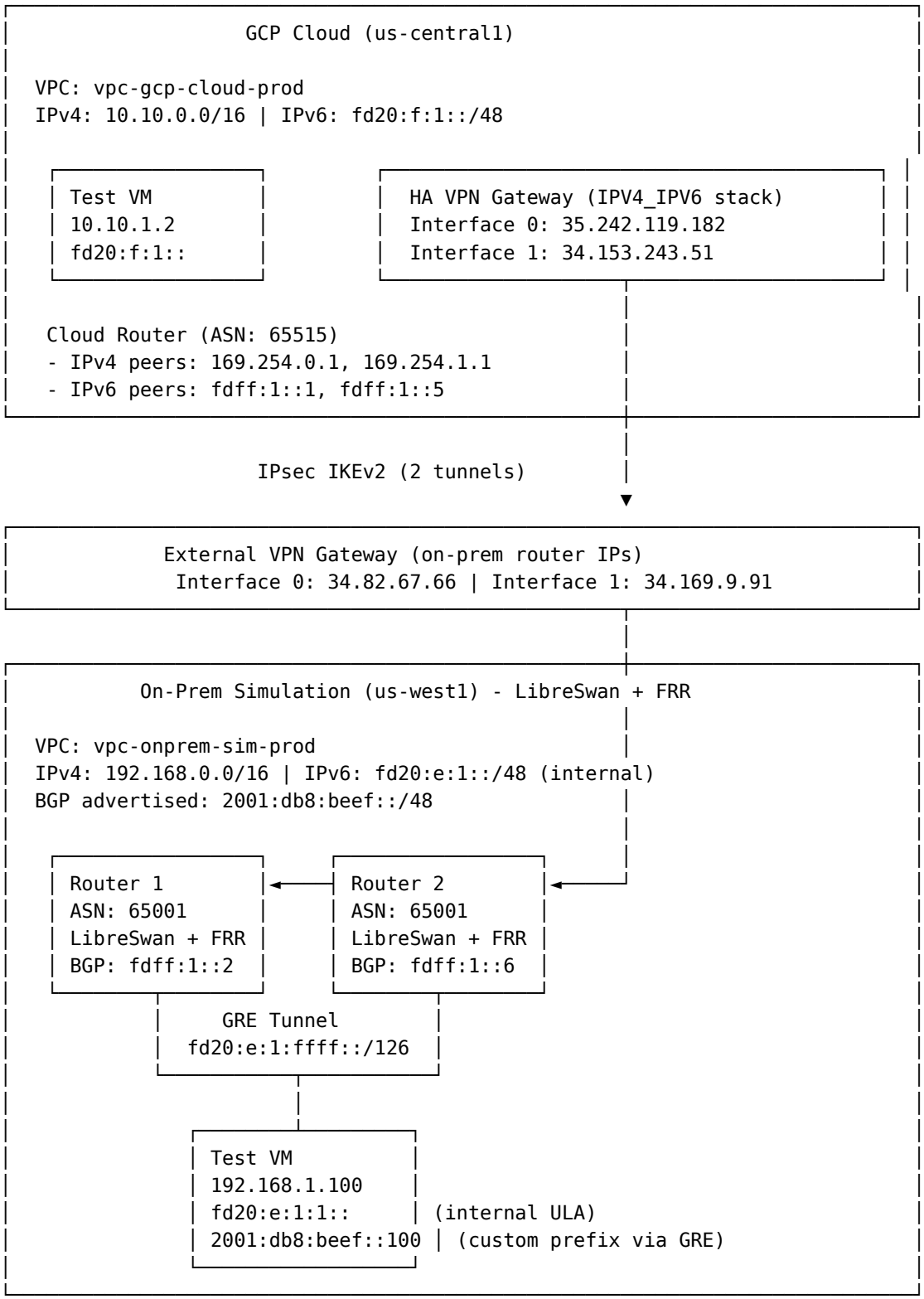
| **Capability** | **IPv4** | **IPv6** |
|---|---|---|
| VPN Tunnels | ✓ | ✓ |
| BGP Sessions | ✓ | ✓ |
| Route Exchange | ✓ | ✓ |
| Cross-VPN Ping | 38ms | 33-37ms |
| Custom Prefix Advertisement | ✓ | ✓ |

**Key Achievements:**

- Full dual-stack connectivity with automatic route propagation via dedicated IPv4 and IPv6 BGP sessions
- On-prem simulation using LibreSwan + FRR successfully peers with GCP Cloud Router
- Custom IPv6 prefix advertisement (2001:db8:beef::/48) via BGP

# 2 Architecture Overview

## 2.1 Network Topology




Listing 1: Network Architecture with On-Prem Simulation

## 2.2 Key Components

| Component | Description |
|---|---|

| | |
|---|---|
| HA VPN Gateway | Regional resource with `stack_type = "IPV4_IPV6"` for dual-stack tunnel support |
| External VPN Gateway | Represents on-prem router public IPs with `redundancy_type = "TWO_IPS_REDUNDANCY"` |
| Cloud Router | BGP router with separate IPv4 and IPv6 peering sessions (ASN 65515) |
| LibreSwan | IPsec VPN daemon on on-prem routers with VTI interfaces |
| FRR | Free Range Routing daemon for BGP on on-prem routers (ASN 65001) |
| GRE Tunnel | Carries traffic for custom-advertised prefixes not in VPC range |

### 2.3 IPv6 Addressing Strategy

| Component | IPv6 Range | Purpose |
|---|---|---|
| GCP Cloud VPC | fd20:f:1::/48 | ULA for GCP internal |
| On-prem VPC (internal) | fd20:e:1::/48 | ULA for on-prem internal |
| On-prem BGP advertisement | 2001:db8:beef::/48 | Custom prefix advertised via BGP |
| BGP peering | fdff:1::/64 | Link-local for BGP sessions |

## 3 Critical Design Decisions

### 3.1 Why Dedicated IPv6 BGP Sessions?

**Important:** MP-BGP (Multiprotocol BGP) on IPv4 sessions does NOT properly install IPv6 routes in GCP Cloud Router.

**What doesn't work:**

- 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 for IPv6 forwarding)

**What works:**

- 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

### 3.2 IPv6 Address Range Validity

**Critical Finding:** The Linux kernel will NOT route to IPv6 addresses in unallocated ranges. `dead:beef::/48` does NOT work because it's outside valid IPv6 address space.

**Valid IPv6 ranges for routing:**

| Range | Type | Routable |
|---|---|---|
| 2000::/3 | Global Unicast (starts with 2 or 3) | ✓ Yes |

| | | |
|---|---|---|
| fc00::/7 | Unique Local (starts with fc or fd) | ✓ Yes |
| dead::/16 | Unallocated (starts with d) | × No |
| face::/16 | Unallocated | × No |

**Why `dead:beef::/48` fails:**

- `dead` in hex = `1101 1110 1010 1101` in binary
- First 3 bits are `110`, not `001` (required for global unicast)
- Linux kernel refuses to use routes for these addresses
- Kernel falls back to default gateway instead of specific route

**Working alternative:**

- Use `2001:db8:beef::/48` instead (documentation range, valid global unicast)
- Or use `fdbe:ef00::/48` (ULA range, looks like "beef")

### 3.3 GRE Tunnel for Custom Prefixes

Since custom-advertised prefixes (like 2001:db8:beef::/48) are not part of the on-prem VPC's allocated range, a GRE tunnel is needed to route traffic to the test VM:

```
Router 1 (fd20:e:1:ffff::1/126) ◄──── GRE Tunnel ────► Test VM (fd20:e:1:ffff::2/126)
                                                                  |
                                                       2001:db8:beef::100/48
```

The GRE tunnel uses the VPC's internal ULA range for transport, and the test VM has the custom prefix configured on the tunnel interface.

### 3.4 BGP Session Configuration

| Session | GCP 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

```
cloud_provi/
├── terraform-gcp-external/      # GCP Cloud Infrastructure
│   ├── main.tf                  # VPC, subnets, firewall
│   ├── vpn.tf                   # HA VPN Gateway, tunnels
│   ├── bgp.tf                   # Cloud Router, BGP peers
│   ├── test-vm.tf               # Test VM
│   ├── variables.tf             # Input variables
│   └── outputs.tf               # VPN config for on-prem
│
├── terraform-onprem-sim/        # On-Prem Simulation
│   ├── main.tf                  # VPC, routers, test VM
│   ├── cloud-init-router.yaml   # LibreSwan + FRR bootstrap
│   └── scripts/
│       └── configure-gcp-vpn.sh # GCP VPN config script
```

## 4.2 Key Resource Definitions

### 4.2.1 GCP VPC with IPv6

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

resource "google_compute_subnetwork" "cloud_workload" {
  name             = "subnet-workload-prod"
  ip_cidr_range    = "10.10.1.0/24"
  region           = "us-central1"
  network          = google_compute_network.cloud.id
  stack_type       = "IPV4_IPV6"
  ipv6_access_type = "INTERNAL"
}
```

### 4.2.2 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
}

resource "google_compute_external_vpn_gateway" "onprem" {
  name            = "extgw-onprem-prod"
  redundancy_type = "TWO_IPS_REDUNDANCY"

  interface {
    id         = 0
    ip_address = var.onprem_router_1_ip  # Router 1 public IP
  }
  interface {
    id         = 1
    ip_address = var.onprem_router_2_ip  # Router 2 public IP
  }
}
```

### 4.2.3 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"  # On-prem router IPv6
  peer_asn        = 65001
```

```
  interface       = google_compute_router_interface.cloud_interface_0_v6.name
  enable_ipv6     = true
}
```

### 4.2.4 On-Prem Router Configuration (LibreSwan)

```
# /etc/ipsec.d/gcp-tunnel-r1.conf
conn gcp-tunnel-r1
    authby=secret
    auto=start
    left=%defaultroute
    leftid=34.82.67.66
    right=35.242.119.182
    type=tunnel
    ikev2=yes
    ike=aes256-sha256-modp2048
    esp=aes256-sha256
    ikelifetime=36000s
    salifetime=10800s
    leftsubnet=0.0.0.0/0
    rightsubnet=0.0.0.0/0
    mark=200/0xffffffff
    vti-interface=vti20
    vti-routing=no
```

### 4.2.5 On-Prem Router Configuration (FRR BGP)

```
router bgp 65001
  bgp router-id 192.168.0.10
  no bgp ebgp-requires-policy
  no bgp network import-check

  ! IPv4 neighbor
  neighbor 169.254.0.1 remote-as 65515
  neighbor 169.254.0.1 ebgp-multihop 2

  ! IPv6 neighbor (dedicated session)
  neighbor fdff:1::1 remote-as 65515
  neighbor fdff:1::1 ebgp-multihop 2

  address-family ipv4 unicast
    network 192.168.0.0/16
    neighbor 169.254.0.1 activate
  exit-address-family

  address-family ipv6 unicast
    network fd20:e:1::/48
    network 2001:db8:beef::/48
    neighbor fdff:1::1 activate
  exit-address-family
exit
```

### 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 = ["10.10.0.0/16", "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:e:1::/48", "fd20:f:1::/48", "2001:db8:beef::/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 Steps

```
# Step 1: Deploy On-Prem Simulation
cd terraform-onprem-sim
terraform init
terraform apply -var="project_id=your-project-id"

# Note the router public IPs from output

# Step 2: Deploy GCP Cloud Infrastructure
cd ../terraform-gcp-external
terraform apply \
  -var="project_id=your-project-id" \
  -var="onprem_router_1_ip=<router-1-ip>" \
  -var="onprem_router_2_ip=<router-2-ip>"

# Step 3: Configure On-Prem Routers
# Save config from terraform output
terraform output -raw onprem_router_1_json_config > /tmp/r1.json

# SSH to Router 1 and configure
gcloud compute ssh vm-router-1-prod --zone=us-west1-a
sudo /usr/local/bin/configure-gcp-vpn.sh --config /tmp/r1.json --router-id 1

# Repeat for Router 2
```

## 5.3 Deployment Time

| Resource | Time |
|---|---|
| On-Prem VPC + Routers | 2 minutes |
| GCP VPC + VPN Gateway | 1 minute |

| | |
|---|---|
| VPN Tunnels + BGP | 30 seconds |
| Router Configuration | 1 minute |
| **Total** | **5 minutes** |

# 6 Verification

## 6.1 Check IPsec Tunnel Status (On-Prem)

```
gcloud compute ssh vm-router-1-prod --zone=us-west1-a
sudo ipsec status
```

**Expected Output:**

```
gcp-tunnel-r1[1]: ESTABLISHED 1 hour ago
gcp-tunnel-r1{1}:  INSTALLED, TUNNEL
```

## 6.2 Check BGP Session Status

```
# On-prem router
sudo vtysh -c "show bgp summary"

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

**Expected Output:**

```
result:
  bgpPeerStatus:
  - name: peer-to-onprem-0-v4
    state: Established
  - name: peer-to-onprem-0-v6
    state: Established
  - name: peer-to-onprem-1-v4
    state: Established
  - name: peer-to-onprem-1-v6
    state: Established
```

## 6.3 Verify Route Installation

```
gcloud compute routers get-status router-cloud-prod-us-central1 \
  --region=us-central1 \
  --format=json | jq '.result.bestRoutesForRouter[] |
    select(.destRange | contains("2001:db8") or contains("fd20:e"))'
```

**Expected Output:**

```
{
  "destRange": "2001:db8:beef::/48",
  "nextHopIp": "fdff:1::2",
  "routeStatus": "ACTIVE",
  "routeType": "BGP"
}
{
  "destRange": "fd20:e:1::/48",
  "nextHopIp": "fdff:1::2",
  "routeStatus": "ACTIVE",
  "routeType": "BGP"
}
```

### 6.4 Test Connectivity

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

# Test IPv4 to on-prem
ping -c 3 192.168.1.100

# Test IPv6 to on-prem ULA
ping6 -c 3 fd20:e:1:2001:0:1::

# Test IPv6 to custom-advertised prefix
ping6 -c 3 2001:db8:beef::100
```

**Expected Results:**

```
PING 192.168.1.100: 3 packets, 0% loss, rtt avg 38ms
PING fd20:e:1:2001:0:1::: 3 packets, 0% loss, rtt avg 37ms
PING 2001:db8:beef::100: 3 packets, 0% loss, rtt avg 33ms
```

## 7 Troubleshooting

### 7.1 Common Issues

| Symptom | Solution |
|---|---|
| IPv6 BGP session DOWN | Verify /126 subnets don't overlap. Check VTI interface has IPv6 address. |
| 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. |
| Ping6 to custom prefix fails | Ensure prefix is in valid range (2000::/3 or fd00::/8). Use GRE tunnel. |
| `dead:beef::` not routable | Invalid IPv6 range. Use `2001:db8:beef::` instead. |
| LibreSwan crash with dual-stack | Create separate IPv4 and IPv6 IPsec connections, not combined. |

### 7.2 LibreSwan Dual-Stack Issue

**LibreSwan 3.x Limitation:** Using `leftsubnet=0.0.0.0/0,::/0` (combined IPv4+IPv6 traffic selectors) can cause daemon crashes.

**Workaround:** Create separate IPsec connections for IPv4 and IPv6, both using the same VTI interface and mark:

```
# IPv4 connection
conn gcp-tunnel-r1-v4
    leftsubnet=0.0.0.0/0
    rightsubnet=0.0.0.0/0
    mark=200/0xffffffff
    vti-interface=vti20

# IPv6 connection (same VTI)
conn gcp-tunnel-r1-v6
    leftsubnet=::/0
    rightsubnet=::/0
```

```
    mark=200/0xffffffff
    vti-interface=vti20
```

### 7.3 Diagnostic Commands

```
# On-prem router
sudo ipsec status                    # IPsec tunnel status
sudo vtysh -c "show bgp summary"     # BGP peer status
sudo vtysh -c "show bgp ipv6"        # IPv6 routes
ip -6 route show                     # Kernel IPv6 routes
ip -6 route get 2001:db8:beef::100   # Test route lookup

# GCP
gcloud compute routers get-status <router> --region=<region>
gcloud compute vpn-tunnels describe <tunnel> --region=<region>
gcloud compute routes list --filter="network:<vpc>"
```

## 8 Cloud Provider Comparison

| Capability | Azure VWAN | Azure VPN GW | GCP HA VPN |
|---|---|---|---|
| IPv6 VPN config | × | Preview | ✓ GA |
| IPv6 BGP sessions | × | Preview | ✓ GA |
| IPv6 route learning | × | Preview | ✓ GA |
| Cross-VPN IPv4 | ✓ | ✓ 69ms | ✓ 38ms |
| Cross-VPN IPv6 | × | Untested | ✓ 33ms |
| Deployment time | 30-45 min | 30-45 min | 3-5 min |
| Custom IPv6 prefix via BGP | × | × | ✓ |

**Conclusion:** GCP HA VPN provides full dual-stack support with the fastest deployment time. Azure requires preview enrollment for IPv6 VPN support.

## 9 Summary

**Working Configuration:**

- GCP HA VPN with `stack_type = "IPV4_IPV6"`
- Dedicated IPv6 BGP sessions (not MP-BGP on IPv4)
- LibreSwan on-prem with separate IPv4/IPv6 IPsec connections
- FRR BGP with dual address families
- GRE tunnel for routing custom-advertised prefixes

**Key Findings:**

1. MP-BGP over IPv4 sessions does not install IPv6 routes in GCP
2. `dead:beef::/48` fails because it's in unallocated IPv6 space
3. Use valid prefixes: `2001:db8::/32` (documentation) or `fd00::/8` (ULA)
4. LibreSwan requires separate connections for IPv4 and IPv6 traffic selectors

## 10 Cleanup

To destroy all resources:

```
# Destroy GCP cloud infrastructure
cd terraform-gcp-external
terraform destroy

# Destroy on-prem simulation
```

```
cd ../terraform-onprem-sim
terraform destroy
```

---

*Document generated from successful GCP dual-stack VPN implementation*

Project: dual-stack-vpn-test | Regions: us-central1 (GCP), us-west1 (On-Prem Sim)

Test Results: IPv4 38ms, IPv6 ULA 37ms, IPv6 Custom Prefix 33ms | 0% packet loss