AWS Dual-Stack VPN with Transit Gateway

Site-to-Site VPN Configuration Guide

January 2026

Contents

1 Executive Summary

This document describes the implementation of a dual-stack (IPv4 + IPv6) Site-to-Site VPN between AWS and an on-premises network using AWS Transit Gateway. The solution was validated using a simulated on-premises environment hosted in Google Cloud Platform with FreeSwan (LibreSwan) for IPsec and FRR for BGP routing.

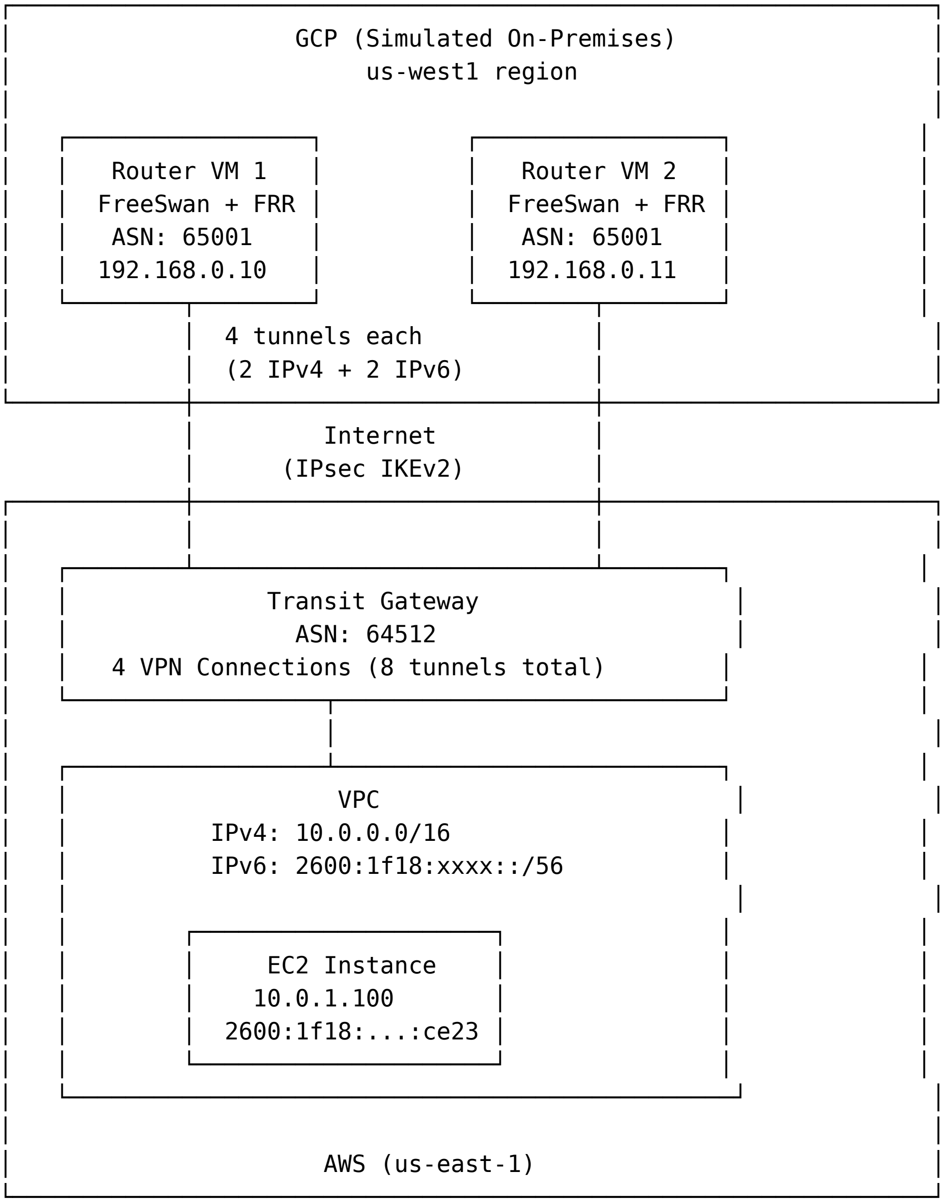
Key Finding: AWS Site-to-Site VPN does not support true dual-stack in a single tunnel. Separate VPN connections are required for IPv4 and IPv6 traffic.

1.1 Test Results Summary

Test	Router 1	Router 2	Latency	Status
IPv4 Connectivity	Pass	Pass	63-64ms	✓
IPv6 Connectivity	Pass	Pass	62-65ms	✓
BGP IPv4 Sessions	Established	Established	N/A	✓
BGP IPv6 Sessions	Established	Established	N/A	✓
IPsec Tunnels (IPv4)	2 UP	2 UP	N/A	✓
IPsec Tunnels (IPv6)	2 UP	2 UP	N/A	✓

2 Architecture Overview

2.1 Network Topology



2.2 IP Addressing

Network	IPv4 CIDR	IPv6 CIDR
AWS VPC	10.0.0.0/16	2600:1f18:xxxx::/56 (Amazon-assigned)
AWS Workload Subnet	10.0.1.0/24	/64 from VPC
On-Prem Network	192.168.0.0/16	fd20:c:1::/48 (ULA)

On-Prem Router Subnet	192.168.0.0/24	fd20:c:1::/64
VPN Inside (IPv4)	169.254.x.x/30	N/A
VPN Inside (IPv6)	N/A	fdxx:xxxx::/126 (per tunnel)

3 AWS Dual-Stack VPN Requirements

3.1 Critical Requirement: Separate VPN Connections

AWS Site-to-Site VPN uses the `tunnel_inside_ip_version` parameter to determine traffic selectors:

Setting	Traffic Selector	Traffic Allowed
`tunnel_inside_ip_version = "ipv4"`	`0.0.0.0/0`	IPv4 only
`tunnel_inside_ip_version = "ipv6"`	`::/0`	IPv6 only

Important: A single VPN connection cannot carry both IPv4 and IPv6 traffic. You must create separate VPN connections for each address family.

3.2 Required Components

For full dual-stack with redundancy, the following resources are needed:

Resource	Count	Purpose
Transit Gateway	1	Required for IPv6 VPN support
Customer Gateways	2	One per on-prem router
VPN Connections (IPv4)	2	IPv4 traffic (one per CGW)
VPN Connections (IPv6)	2	IPv6 traffic (one per CGW)
IPsec Tunnels	8	2 tunnels per VPN connection
BGP Sessions (IPv4)	4	Over 169.254.x.x addresses
BGP Sessions (IPv6)	4	Over fdxx:: addresses

3.3 Transit Gateway Configuration

The Transit Gateway VPC attachment **must** have IPv6 support enabled for IPv6 route propagation:

```
resource "aws_ec2_transit_gateway_vpc_attachment" "main" {
  subnet_ids         = [aws_subnet.tgw.id]
  transit_gateway_id = aws_ec2_transit_gateway.main.id
  vpc_id             = aws_vpc.main.id
  ipv6_support       = "enable"  # CRITICAL for IPv6 routes
}
```

Without `ipv6_support = "enable"`, the VPC IPv6 CIDR will not be propagated to the Transit Gateway route table.

4 Terraform Configuration

4.1 VPN Connection Resources

4.1.1 IPv6 VPN Connection

```
resource "aws_vpn_connection" "router_1_ipv6" {
  customer_gateway_id = aws_customer_gateway.router_1.id
  transit_gateway_id  = aws_ec2_transit_gateway.main.id
  type                = "ipsec.1"

  static_routes_only       = false
  tunnel_inside_ip_version = "ipv6"  # IPv6 traffic selectors
  enable_acceleration      = false

  # IKEv2 with AES256-SHA256
  tunnel1_ike_versions                 = ["ikev2"]
  tunnel1_phase1_encryption_algorithms = ["AES256"]
  tunnel1_phase1_integrity_algorithms  = ["SHA2-256"]
  tunnel1_phase1_dh_group_numbers      = [14]
  tunnel1_phase2_encryption_algorithms = ["AES256"]
  tunnel1_phase2_integrity_algorithms  = ["SHA2-256"]
  tunnel1_phase2_dh_group_numbers      = [14]

  # Same for tunnel2...

  tags = { Name = "vpn-router-1-ipv6" }
}
```

4.1.2 IPv4 VPN Connection

```
resource "aws_vpn_connection" "router_1_ipv4" {
  customer_gateway_id = aws_customer_gateway.router_1.id
  transit_gateway_id  = aws_ec2_transit_gateway.main.id
  type                = "ipsec.1"

  static_routes_only       = false
  tunnel_inside_ip_version = "ipv4"  # IPv4 traffic selectors
  enable_acceleration      = false

  # Same IKE/IPsec parameters as IPv6...

  tags = { Name = "vpn-router-1-ipv4" }
}
```

4.2 Route Table Configuration

```
resource "aws_route_table" "main" {
  vpc_id = aws_vpc.main.id

  # On-prem IPv4 via TGW
  route {
    cidr_block         = "192.168.0.0/16"
    transit_gateway_id = aws_ec2_transit_gateway.main.id
  }

  # On-prem IPv6 via TGW
  route {
    ipv6_cidr_block    = "fd20:c:1::/48"
    transit_gateway_id = aws_ec2_transit_gateway.main.id
  }

  # VPN tunnel inside addresses (ULA) via TGW
  route {
    ipv6_cidr_block    = "fd00::/8"
    transit_gateway_id = aws_ec2_transit_gateway.main.id
  }
}
```

5 On-Premises Router Configuration

5.1 IPsec Configuration (LibreSwan)

5.1.1 IPv6 Tunnel Configuration

```
# /etc/ipsec.d/aws-r1-v6-tun1.conf
conn aws-r1-v6-tun1
    authby=secret
    auto=start
    left=%defaultroute
    leftid=<router-public-ip>
    right=<aws-tunnel-outside-ip>
    type=tunnel
    ikelifetime=8h
    keylife=1h
    phase2alg=aes256-sha256
    ike=aes256-sha256-modp2048
    keyingtries=%forever
    # IPv6 traffic selectors (REQUIRED for IPv6 VPN)
    leftsubnet=::/0
    rightsubnet=::/0
    mark=100/0xffffffff
    vti-interface=vti1
    vti-routing=no
    leftvti=169.254.x.x/30
    dpddelay=10
    dpdtimeout=30
    dpdaction=restart_by_peer
```

5.1.2 IPv4 Tunnel Configuration

```
# /etc/ipsec.d/aws-r1-v4-tun1.conf
conn aws-r1-v4-tun1
    authby=secret
    auto=start
    left=%defaultroute
    leftid=<router-public-ip>
    right=<aws-tunnel-outside-ip>
    type=tunnel
    ikelifetime=8h
    keylife=1h
    phase2alg=aes256-sha256
    ike=aes256-sha256-modp2048
    keyingtries=%forever
    # IPv4 traffic selectors
    leftsubnet=0.0.0.0/0
    rightsubnet=0.0.0.0/0
    mark=200/0xffffffff
    vti-interface=vti3
    vti-routing=no
    leftvti=169.254.y.y/30
    dpddelay=10
    dpdtimeout=30
    dpdaction=restart_by_peer
```

5.2 VTI Interface Setup

Each tunnel requires a VTI (Virtual Tunnel Interface):

```
# IPv6 tunnel VTI (needs both IPv4 and IPv6 addresses)
ip tunnel add vti1 local 192.168.0.10 remote <aws-outside-ip> mode vti key 100
ip addr add 169.254.x.x/30 dev vti1           # IPv4 inside address
ip -6 addr add fdxx:xxxx::2/126 dev vti1      # IPv6 inside address
ip -6 addr add fd20:c:1::1/128 dev vti1       # On-prem source address
ip link set vti1 up mtu 1419

# IPv4 tunnel VTI
ip tunnel add vti3 local 192.168.0.10 remote <aws-outside-ip> mode vti key 200
ip addr add 169.254.y.y/30 dev vti3
ip link set vti3 up mtu 1419

# Disable reverse path filtering
sysctl -w net.ipv4.conf.vti1.disable_policy=1
sysctl -w net.ipv4.conf.vti1.rp_filter=0
```

5.3 BGP Configuration (FRR)

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

  # IPv4 BGP neighbors (over IPv4 tunnels)
  neighbor 169.254.11.61 remote-as 64512
  neighbor 169.254.11.61 description aws-tgw-v4-tun1
  neighbor 169.254.11.61 ebgp-multihop 255
  neighbor 169.254.11.61 update-source 169.254.11.62

  neighbor 169.254.148.57 remote-as 64512
  neighbor 169.254.148.57 description aws-tgw-v4-tun2
  neighbor 169.254.148.57 ebgp-multihop 255
  neighbor 169.254.148.57 update-source 169.254.148.58

  # IPv6 BGP neighbors (over IPv6 tunnels)
  neighbor fd0a:8229:3c32:9e:73c3:ca48:e7b7:b181 remote-as 64512
  neighbor fd0a:8229:3c32:9e:73c3:ca48:e7b7:b181 description aws-tgw-v6-tun1

  neighbor fd56:7823:e336:1529:6e0:4b0b:1a4:da49 remote-as 64512
  neighbor fd56:7823:e336:1529:6e0:4b0b:1a4:da49 description aws-tgw-v6-tun2

  # IPv4 address family
  address-family ipv4 unicast
    network 192.168.0.0/16
    neighbor 169.254.11.61 activate
    neighbor 169.254.148.57 activate
  exit-address-family

  # IPv6 address family
  address-family ipv6 unicast
    network fd20:c:1::/48
    neighbor fd0a:8229:3c32:9e:73c3:ca48:e7b7:b181 activate
    neighbor fd0a:8229:3c32:9e:73c3:ca48:e7b7:b181 next-hop-self
    neighbor fd56:7823:e336:1529:6e0:4b0b:1a4:da49 activate
    neighbor fd56:7823:e336:1529:6e0:4b0b:1a4:da49 next-hop-self
  exit-address-family
```

6 Verification Commands

6.1 AWS Side

```
# Check VPN connection status
aws ec2 describe-vpn-connections \
  --query 'VpnConnections[].{ID:VpnConnectionId,State:State,Tunnels:VgwTelemetry}'

# Check Transit Gateway route table
aws ec2 search-transit-gateway-routes \
  --transit-gateway-route-table-id <tgw-rtb-id> \
  --filters "Name=type,Values=propagated"
```

6.2 On-Premises Router

```
# IPsec tunnel status
sudo ipsec status | grep ESTABLISHED

# BGP session status
sudo vtysh -c "show bgp summary"

# IPv4 routes learned from AWS
sudo vtysh -c "show bgp ipv4 unicast"

# IPv6 routes learned from AWS
sudo vtysh -c "show bgp ipv6 unicast"

# Test connectivity
ping -I 192.168.0.10 10.0.1.100      # IPv4
ping6 -I fd20:c:1::1 <aws-ipv6>      # IPv6
```

7 Troubleshooting

7.1 Common Issues

7.1.1 IPsec Tunnel Not Establishing

Symptom: Tunnel stays in `STATE_PARENT_I2` state

Possible Causes:

- Incorrect traffic selectors (IPv4 vs IPv6 mismatch)
- Pre-shared key mismatch
- Firewall blocking UDP 500/4500 or ESP

Solution:

- Verify `leftsubnet` matches VPN connection type
- Check `/etc/ipsec.secrets` format
- Verify security groups allow IKE and IPsec

7.1.2 BGP Session Not Establishing

Symptom: BGP neighbor shows `Idle` or `Active` state

Possible Causes:

- Wrong source address for BGP
- Route to BGP peer missing
- TCP port 179 blocked

Solution:

- Add `update-source` with VTI inside address
- Verify VTI interface is UP
- Test ping to BGP peer IP

7.1.3 IPv6 Ping Fails

Symptom: IPv6 ping times out

Possible Causes:

- Using wrong source address
- Return route missing in AWS
- Security group missing ICMPv6 rule

Solution:

- Use `-I <on-prem-ipv6>` with ping6
- Add `fd00::/8` route to TGW
- Allow ICMPv6 from `fd00::/8` in security group

8 Cloud Provider Comparison

Capability	**Azure VWAN**	**GCP HA VPN**	**AWS TGW VPN**
True dual-stack single tunnel	No	Yes*	No
IPv6 VPN support	No	Yes	Yes
Separate IPv4/IPv6 connections	N/A	No	Required
IPv6 BGP sessions	No	Yes	Yes
Cross-VPN IPv4	Failed	32ms	63ms
Cross-VPN IPv6	Failed	33ms	63ms
Deployment time	30-45 min	~3 min	~5 min

*GCP requires dedicated IPv6 BGP sessions for proper route installation

9 Conclusion

AWS Transit Gateway VPN supports dual-stack connectivity but requires careful configuration:

1. **Separate VPN connections** for IPv4 and IPv6 traffic
2. **Transit Gateway** is mandatory (VGW does not support IPv6)
3. `ipv6_support = "enable"` on TGW VPC attachment
4. **Proper source addresses** when testing (use on-prem IPs, not VTI IPs)

The solution provides reliable dual-stack connectivity with ~63ms latency between GCP (us-west1) and AWS (us-east-1), with full BGP route exchange and redundancy across multiple tunnels.

10 Appendix: File Locations

File	**Purpose**
`terraform-aws/main.tf`	AWS infrastructure (TGW, VPN, VPC)
`terraform-aws/outputs.tf`	VPN tunnel configurations
`terraform-onprem-sim/main.tf`	GCP on-prem simulation
`terraform-onprem-sim/cloud-init-router.yaml`	Router VM bootstrap
`r1-config.json`, `r2-config.json`	IPv6 tunnel configs
`r1-ipv4-config.json`, `r2-ipv4-config.json`	IPv4 tunnel configs