

Hybrid Cloud Setup Using VPN

1. Project Overview

Project Name: Hybrid Cloud Setup Using VPN

Services Used: AWS Site-to-Site VPN, VPC, EC2, Route Tables, Virtual Private Gateway, Customer Gateway

Goal: Securely connect an on-premises *simulated environment* (VirtualBox VM) with AWS VPC using Site-to-Site VPN.

Skills Gained: Hybrid networking, routing, VPN concepts, secure connectivity, hybrid DNS basics.

2. Architecture Overview (Conceptual)

- On-Premises Network (VirtualBox VM)
 - Example CIDR: 192.168.1.0/24
 - Acts as Customer Network
 - AWS Cloud
 - VPC CIDR: 10.0.0.0/16
 - Public subnet with EC2 instance
 - Site-to-Site VPN tunnel connecting both networks
-

3. Prerequisites

On-Premises (Local System)

- VirtualBox installed
- Linux VM (Ubuntu preferred)
- VM Network Adapter: **Bridged or Host-only**
- Static private IP (example: 192.168.1.10)

AWS Account Requirements

- AWS account
 - IAM user with VPC full access
 - Region selected (example: **ap-south-1**)
-

4. Step-by-Step AWS Console Setup

Step 1: Create a VPC

1. Go to **AWS Console** → **VPC**
2. Click **Create VPC**
3. Select **VPC only**
4. Configure:
 - Name: Hybrid-VPC
 - IPv4 CIDR: 10.0.0.0/16
5. Click **Create VPC**

vpc-04a82266b105ca4fb / Hybrid-VPC

Details

VPC ID vpc-04a82266b105ca4fb	State Available	Block Public Access Off	DNS hostnames Disabled
DNS resolution Enabled	Tenancy default	DHCP option set dept-022f8ada1dc09847f	Main route table rtb-02a92bae57e23b09e
Main network ACL acl-0229497f3d488645	Default VPC No	IPv4 CIDR 10.0.0.0/16	IPv6 pool -
IPv6 CIDR (Network border group) -	Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups -	Owner ID 162308490504
Encryption control ID -	Encryption control mode -		

Step 2: Create a Public Subnet

1. VPC Dashboard → Subnets → Create subnet
2. Select VPC: Hybrid-VPC
3. Configure:
 - Subnet name: Public-Subnet
 - Availability Zone: ap-south-1a
 - CIDR: 10.0.1.0/24
4. Click **Create subnet**

Subnets (1) info

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
Public-Subnet	subnet-0b2afbf9611561fcf	Available	vpc-04a82266b105ca4fb Hybr...	Off	10.0.1.0/24

Step 3: Create Internet Gateway

1. VPC → Internet Gateways → Create
2. Name: Hybrid-IGW
3. Click **Create internet gateway**
4. Attach to VPC:
 - Actions → Attach → Select Hybrid-VPC

igw-05bb7971542a91173 / Hybrid-IGW

Details

Internet gateway ID igw-05bb7971542a91173	State Attached	VPC ID vpc-04a82266b105ca4fb Hybrid-VPC	Owner 162308490504
--	-------------------	--	-----------------------

Tags (1)

Key Name	Value Hybrid-IGW
-------------	---------------------

Step 4: Create Route Table

1. VPC → **Route Tables** → Create
2. Name: Public-RT
3. VPC: Hybrid-VPC
4. Edit Routes:
 - Destination: 0.0.0.0/0
 - Target: Internet Gateway (Hybrid-IGW)
5. Associate with Public-Subnet

Step 5: Launch EC2 Instance (AWS Side)

1. Go to **EC2** → **Launch Instance**
2. Choose AMI: Amazon Linux 2
3. Instance type: t2.micro
4. Network:
 - VPC: Hybrid-VPC
 - Subnet: Public-Subnet
 - Auto-assign public IP: Enabled
5. Security Group:
 - Allow SSH (22)
 - Allow ICMP (Ping)
6. Launch instance

Step 6: Create Customer Gateway (On-Prem Representation)

1. VPC → **Customer Gateways** → Create
2. Configure:
 - Name: OnPrem-CGW
 - Routing: Static
 - IP Address: **Public IP of your local internet (ISP)**
3. Click **Create customer gateway**

The screenshot shows the AWS VPC Customer Gateways page. A success message at the top states: "You successfully created cgw-0a9304c53e2e3b4ab / OnPrem-CGW." Below this is a table titled "Customer gateways (1/1) info". The table has columns: Name, Customer gateway ID, State, BGP ASN, IP address, and Type. One row is listed: "OnPrem-CGW" with ID "cgw-0a9304c53e2e3b4ab", State "Available", BGP ASN "65000", IP address "152.57.123.122", and Type "ipsec.1". At the bottom, there's a detailed view for "Customer gateway cgw-0a9304c53e2e3b4ab / OnPrem-CGW" with tabs for "Details" and "Tags". The "Details" tab shows fields like Customer gateway ID, State, BGP ASN, Type, Certificate ARN, Device, and IP address.

Step 7: Create Virtual Private Gateway

1. VPC → **Virtual Private Gateways** → Create
2. Name: Hybrid-VGW
3. ASN: Default
4. Create and Attach:
 - Attach to Hybrid-VPC

The screenshot shows the AWS VPC Virtual Private Gateways page. A success message at the top states: "You successfully attached vgw-005678abbdcbc2dfc / Hybrid-VGW to vpc-03edd1d9daf10bdd." Below this is a table titled "Virtual private gateways (1) info". The table has columns: Name, Virtual private gateway ID, State, VPC attachment state, and Type. One row is listed: "Hybrid-VGW" with ID "vgw-005678abbdcbc2dfc", State "Available", VPC attachment state "Attached", and Type "ipsec.1". At the bottom, there's a "Select a virtual private gateway" section.

Step 8: Create Site-to-Site VPN Connection

1. VPC → **Site-to-Site VPN Connections** → Create
2. Configure:
 - Name: Hybrid-VPN
 - Target Gateway Type: Virtual Private Gateway
 - Virtual Private Gateway: Hybrid-VGW
 - Customer Gateway: Existing (OnPrem-CGW)
 - Routing Options: Static

- Static IP Prefix: 192.168.1.0/24

3. Create VPN Connection

The screenshot shows the AWS VPC Dashboard with the 'VPN connections' section selected. A single VPN connection, 'Hybrid-VPN', is listed. The connection details are as follows:

Name	VPN ID	State	Virtual private gateway	Transit gateway
Hybrid-VPN	vpn-019b5ee50a48fe371	Available	vgw-005678abbdcbc2dfc	-

Below the table, the 'VPN connection vpn-019b5ee50a48fe371 / Hybrid-VPN' is expanded. The 'Details' tab is selected, showing the following configuration:

VPN ID	vpn-019b5ee50a48fe371	State	Available	Virtual private gateway	vgw-005678abbdcbc2dfc	Customer gateway	cgw-0c3beb657a461e87
Transit gateway	-	Customer gateway address	157.50.198.84	Type	ipsec.1	Category	VPN

Step 9: Download VPN Configuration

1. Select VPN connection
 2. Click **Download configuration**
 3. Vendor: Generic
 4. Save the configuration file

VPN connection configuration details:

VPN Connection Configuration

AWS utilizes unique identifiers to manipulate the configuration of a VPN Connection. Each VPN Connection is assigned a VPN Connection Identifier and is associated with two other identifiers, namely the Customer Gateway Identifier and the Virtual Private Gateway Identifier.

Your VPN Connection ID : vpn-019b5ee50a48fe371
Your Virtual Private Gateway ID : vgw-005678abbdcbc2dfc
Your Customer Gateway ID : cgw-0c3bebb657a461e87

Subnets: A VPN Connection consists of a pair of IPSec tunnel security associations (SAs).
Routes: It is important that both tunnel security associations be configured.

Internet: Egress-IP IPsec Tunnel #1

gateway: #1: Internet Key Exchange Configuration

DHCP options: Configure the IKE SA as follows:
Please note, these sample configurations are for the minimum requirement of AES128 SHA1 and DH Group 2

Elastic IP Manager: NAT gateways

Peering connections: VPN ID: vpn-019b5ee50a48fe371, Include sample type: Enable, Transit gateway: -

Route servers:

Step 10: Update VPC Route Table for VPN

1. VPC → Route Tables → Select Public-RT
 2. Edit routes:

- Destination: 192.168.1.0/24
- Target: Virtual Private Gateway (Hybrid-VGW)

3. Save routes

The screenshot shows the AWS VPC Route Tables console. The top navigation bar includes 'VPC' > 'Route tables' > 'rtb-02fb8df9f0ccdde23'. A green success message box at the top right says 'Updated routes for rtb-02fb8df9f0ccdde23 / Public-RT successfully' with a 'Details' link. The left sidebar has sections for 'Virtual private cloud' (Your VPCs, Subnets, Route tables selected), 'Internet gateways', 'Egress-only internet gateways', 'DHCP option sets', 'Elastic IPs', 'Managed prefix lists', 'NAT gateways', 'Peering connections', and 'Route servers'. The main content area shows the 'Routes' tab for the selected route table. It displays three routes in a table:

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	igw-0f9c5356e0cac9e65	Active	No	Create Route
10.0.0.0/16	local	Active	No	Create Route Table
192.168.1.0/24	igw-0f9c5356e0cac9e65	Active	No	Create Route

Step 11: Verify VPN Status

Purpose

To check whether the **Site-to-Site VPN tunnel** between AWS and on-premises network is **established and working**.

Console Steps

1. Login to AWS Management Console
2. Go to Services → VPC
3. In the left menu, click **Site-to-Site VPN Connections**
4. Select the VPN connection named **Hybrid-VPN**
5. Scroll down to the **Tunnel details** section

What You Will See

AWS creates **2 VPN tunnels** for high availability:

Tunnel	Status
Tunnel 1	UP / DOWN
Tunnel 2	UP / DOWN

Status Meaning

✓UP

- VPN tunnel is successfully established
- AWS can reach the on-premises gateway
- Routing and encryption are working

- Hybrid connectivity is **active**

XDOWN

- VPN tunnel is **not established**
- AWS cannot reach the on-premises VPN device
- Common reasons:
 - On-prem VPN software not configured
 - Wrong public IP in Customer Gateway
 - ISP blocks IPsec traffic
 - Tunnel not initiated yet

The screenshot shows the AWS VPC Dashboard under the 'VPN connections' section. A specific connection named 'Tunnel 2' is selected. The connection details are as follows:

Setting	Value
Tunnel ID	Tunnel 2
Local IP Range	3.7.116.190
Remote IP Range	169.254.108.132/30
Status	Down
Last Update	December 15, 2025

Tunnel 1 options

Setting	Value
Phase 1 encryption algorithms	AES128, AES256, AES128-GCM-16, AES256-GCM-16
Phase 2 encryption algorithms	AES128, AES256, AES128-GCM-16, AES256-GCM-16
IKE version	ikev1, ikev2
Rekey margin time	270
CloudWatch log group for tunnel VPN log	-
CloudWatch log group for tunnel BGP log	-
Phase 1 integrity algorithms	SHA1, SHA2-256, SHA2-384, SHA2-512
Phase 2 integrity algorithms	SHA1, SHA2-256, SHA2-384, SHA2-512
Rekey fuzz	100
Replay window size	1024
Output format for tunnel VPN log	-
Output format for tunnel BGP log	-
Phase 1 DH group numbers	2, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24
Phase 2 DH group numbers	2, 5, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24
DDP timeout	30
DDP timeout action	clear
Tunnel endpoint lifecycle control	Off
Phase 1 lifetime	28800
Phase 2 lifetime	3600
Startup action	add
Tunnel VPN log	Disabled
Tunnel BGP log	Disabled

12. Verification & Testing

From On-Prem VM

```
ping 10.0.0.10
```

```
ssh ec2-user@10.0.0.10
```

From EC2 Instance

```
ping 192.168.1.10
```

Successful ping confirms hybrid connectivity.

```

unbound-anchor-1.17.1-1.amzn2023.0.10.x86_64      unbound-libs-1.17.1-1.amzn2023.0.10.x86_64

Complete!
[ec2-user@ip-10-0-1-89 ~]$ ipsec --version
Libreswan 4.12
[ec2-user@ip-10-0-1-89 ~]$ sudo systemctl enable ipsec
sudo systemctl start ipsec
sudo systemctl status ipsec
Created symlink /etc/systemd/system/multi-user.target.wants/ipsec.service → /usr/lib/systemd/system/ipsec.service.
● ipsec.service - Internet Key Exchange (IKE) Protocol Daemon for IPsec
   Loaded: loaded (/usr/lib/systemd/system/ipsec.service; enabled; preset: disabled)
     Active: active (running) since Mon 2025-12-15 14:49:58 UTC; 78ms ago
       Docs: man:ipsec(8)
             man:pluto(8)
             man:ipsec.conf(5)
   Process: 28241 ExecStartPre=/usr/libexec/ipsecd/addconn --config /etc/ipsec.conf --checkconfig (code=exited, status=0/SUCCESS)
   Process: 28242 ExecStartPre=/usr/libexec/ipsecd/_stackmanager start (code=exited, status=0/SUCCESS)
   Process: 28625 ExecStartPre=/usr/sbin/ipsecd --checknss (code=exited, status=0/SUCCESS)
   Process: 28630 ExecStartPre=/usr/sbin/ipsecd --checknflog (code=exited, status=0/SUCCESS)
 Main PID: 28641 (pluto)
   Status: "Startup completed."
     Tasks: 2 (limit: 1120)
    Memory: 7.8M
      CPU: 504ms
     CGroup: /system.slice/ipsec.service
             └─28641 /usr/libexec/ipsecd/pluto --leak-detective --config /etc/ipsec.conf --nofork

Dec 15 14:49:58 ip-10-0-1-89.ap-south-1.compute.internal pluto[28641]: listening for IKE messages
Dec 15 14:49:58 ip-10-0-1-89.ap-south-1.compute.internal pluto[28641]: Kernel supports NIC esp-hw-offload
Dec 15 14:49:58 ip-10-0-1-89.ap-south-1.compute.internal pluto[28641]: adding UDP interface enX0 10.0.1.89:500

```

13. Hybrid DNS

- On-prem DNS resolves local names
- AWS Route 53 (optional) resolves cloud resources
- Use conditional forwarding for hybrid DNS

14. Security Best Practices

- Restrict Security Group rules
- Use strong VPN pre-shared keys
- Enable CloudWatch logs for VPN
- Rotate keys periodically

15. Project Outcome

- ✓ Secure hybrid connectivity established
- ✓ On-prem and AWS networks communicate privately
- ✓ Real-world enterprise hybrid cloud setup simulated

16. Conclusion

This project demonstrates a **real-world Hybrid Cloud architecture** using AWS Site-to-Site VPN. It is widely used in enterprises to extend on-prem infrastructure securely into AWS without exposing services to the public internet.