**HYBRID CLOUD SETUP USING AWS VPN**

## **1. Project Overview**

**Hybrid Cloud** is a fully functional hybrid cloud architecture project built entirely on **AWS Free Tier**. It connects an **on-premises network** simulated using an EC2 instance to a **cloud network (AWS VPC)** using a **Site-to-Site VPN tunnel** configured with **strongSwan** (open-source IPSec VPN). The goal is to simulate a real-world enterprise-grade hybrid environment without any physical hardware.

## **2. Purpose of the Project**

The primary purpose of this project is to:

* Demonstrate secure connectivity between on-prem and cloud networks
* Simulate enterprise hybrid infrastructure without cost
* Learn AWS VPN services and IPSec tunneling using strongSwan
* Provide hands-on experience with networking, security, and remote access in cloud computing.

## **3. Problems Before the Project**

## No secure communication between local and cloud-hosted applications

## No simulation of hybrid infrastructure for learning or testing

## Lack of cost-effective, real-world infrastructure in cloud learning paths

## Difficulty understanding VPN/IPSec configuration in cloud environments.

## **4. Problems Solved with This Project**

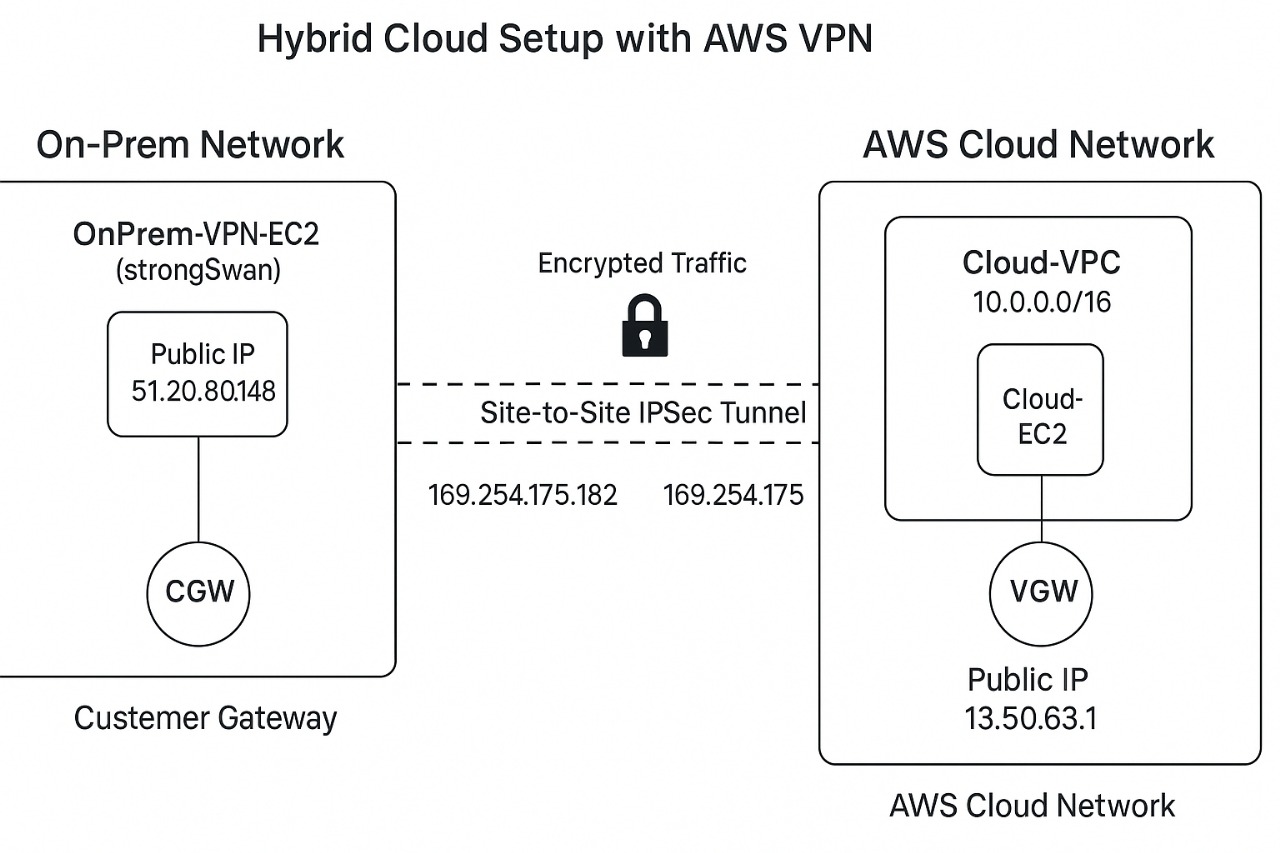
## Established secure encrypted tunnel using IPSec/IKEv1 between cloud and on-prem

## Enabled private-to-private IP communication

## Removed dependency on physical VPN appliances

## Simulated enterprise-level hybrid connectivity using only Free Tier resources.

## **5. Architecture**



## **6. Configuration Steps with Purpose**

### **Phase 1: AWS VPC Setup (Cloud Side)**

#### a. **VPC Creation (Cloud-VPC)**

* **CIDR**: 10.0.0.0/16
* **Purpose**: Logical container for cloud network

#### **b. Subnet Creation (Cloud-Subnet)**

* **CIDR**: 10.0.1.0/24
* **Purpose**: Allocate a range for cloud EC2s

#### **c. Internet Gateway (IGW)**

* Attached to VPC to allow internet access

#### **d. Route Table Update**

* Added route to 0.0.0.0/0 via IGW
* Added route to 192.168.1.0/24 via **Virtual Private Gateway (VGW)** after VPN setup

#### **e. Security Group**

* Allowed SSH (port 22) and ICMP (ping) from custom IPs / 0.0.0.0/0

### **Phase 2: On-Prem VPC Simulation**

#### **a. VPC Creation (OnPrem-VPC)**

* **CIDR**: 192.168.0.0/16

#### **b. Subnet (OnPrem-Subnet)**

* **CIDR**: 192.168.1.0/24
* **Purpose**: Simulated local data center range

#### **c. EC2 Instance: OnPrem-VPN-EC2**

* OS: Amazon Linux (strongSwan manually installed)
* Purpose: Simulated on-prem gateway

#### **d. Security Group:**

* Allow **UDP 500, 4500** (IPSec ports)
* Allow **ICMP** and **SSH** from trusted IPs

### **Phase 3: VPN Setup**

#### **a. Virtual Private Gateway (VGW)**

* Attached to **Cloud-VPC**

#### **b. Customer Gateway (CGW)**

* Dynamic public IP from OnPrem EC2
* **Type**: Generic (no Elastic IP to stay in Free Tier)

#### **c. Site-to-Site VPN Connection**

* Uses CGW and VGW
* **IKE version**: IKEv1
* Downloaded Generic Config for **strongSwan**

## **Phase 3.1: strongSwan Setup and Tunneling**

### **Packages Installed on Amazon Linux 2023 OnPrem EC2:**

Sudo dnf groupinstall -y "Development Tools"  
sudo dnf install -y gcc gmp-devel libcurl-devel libcap-ng-devel libselinux-devel pam-devel flex bison openssl-devel wget iproute

### **Download and install strongSwan from Source:**

wget https://download.strongswan.org/strongswan-5.9.11.tar.gz  
tar xzf strongswan-5.9.11.tar.gz  
cd strongswan-5.9.11  
./configure --prefix=/usr --sysconfdir=/etc --enable-systemd --disable-gmp --enable-openssl  
make  
sudo make install

### **Create strongSwan Config Files:**

#### **/etc/ipsec.conf:**

config setup  
 charondebug="ike 2, knl 2, cfg 2, net 2"  
 uniqueids=no  
  
conn aws-vpn  
 auto=start  
 keyexchange=ikev1  
 type=tunnel  
 authby=psk  
  
 left=%defaultroute  
 leftid=51.20.80.148  
 leftsubnet=192.168.1.0/24  
 leftsourceip=169.254.175.182  
  
 right=13.50.63.1  
 rightid=13.50.63.1  
 rightsubnet=10.0.0.0/16  
 rightsourceip=169.254.175.181  
  
 ike=aes128-sha1-modp1024  
 esp=aes128-sha1  
 dpdaction=restart  
 dpddelay=10s  
 dpdtimeout=30s  
 ikelifetime=28800s  
 lifetime=3600s  
 keyingtries=%forever

**/etc/ipsec.secrets:**

51.20.80.148 13.50.63.1 : PSK "PF7Kc7Vah5vtREx2ukjMtXx115mvEaYt"

### **Start strongSwan Service:**

sudo ipsec restart

### **Manually Bring Up the Tunnel:**

sudo ipsec up aws-vpn

### **Verify Tunnel Status:**

sudo ipsec statusall

Look for INSTALLED and ESTABLISHED status to confirm tunnel is active.

### **Phase 4: Routing and Testing**

#### **a. Route Table Update in AWS:**

* In Cloud-VPC Route Table, added static route:
  + Destination: 192.168.1.0/24
  + Target: Virtual Private Gateway

#### **b. Ping Test (Cloud EC2 > OnPrem EC2):**

ping 192.168.1.41

Successful response indicates full hybrid VPN tunnel functionality.

## **7. IP Address Summary**

| Component | IP Address | Notes |
| --- | --- | --- |
| Cloud VPC CIDR | 10.0.0.0/16 | AWS side |
| Cloud Subnet CIDR | 10.0.1.0/24 | AWS EC2 lives here |
| Cloud EC2 Private IP | 10.0.1.x | Example: 10.0.1.100 |
| OnPrem VPC CIDR | 192.168.0.0/16 | Local simulation |
| OnPrem Subnet CIDR | 192.168.1.0/24 | Local subnet for VPN EC2 |
| OnPrem EC2 Private IP | 192.168.1.41 | Static or assigned IP |
| OnPrem EC2 Public IP | 51.20.80.148 | Dynamic CGW IP |
| VGW Public IP | 13.50.63.1 | AWS VPN side |
| Tunnel Inside (CGW) | 169.254.175.182 | VPN tunnel interface |
| Tunnel Inside (VGW) | 169.254.175.181 | VPN tunnel interface |

## **8. Conclusion**

This project successfully delivers a **production-simulated hybrid cloud setup** using AWS VPN and

an on-premises simulation. It solves the challenge of learning hybrid connectivity without expensive

hardware or third-party tools. Using only Free Tier resources and native Linux tools, it provides hands

on experience in VPN, routing, network design, and secure cloud communication.

**Hybrid Cloud** is not just a learning project — it’s a foundation for building scalable, secure, real

world enterprise architectures on AWS.

**Next Steps / Advancements (Optional):**

* Add second tunnel for failover
* Monitor VPN with CloudWatch + SNS
* Convert this setup into Terraform-based IaC
* Use AWS Transit Gateway for multi-site simulation