Janarthanan Kugathasan

**StrongSwan HA Cluster in AWS with Private IPs – Terraform Provisioning Guidelines (V2)**

Contents

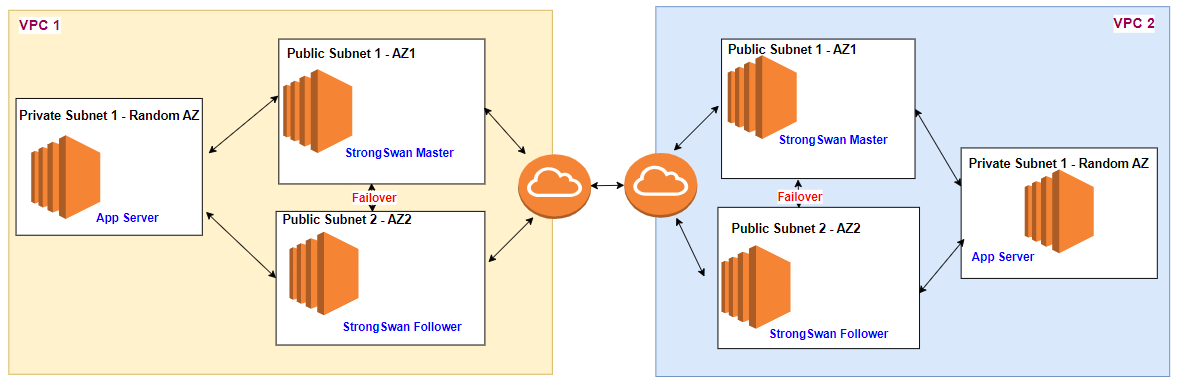
[**Architecture** 2](#_Toc147671291)

[**Resources Created** 2](#_Toc147671292)

[**Method** 3](#_Toc147671293)

[**How Failover between StrongSwan takes place?** 4](#_Toc147671294)

# **Architecture**



# **Resources Created**

There are 2 Terraform modules provided.

1. VPCSite
   1. This module is used to create 1VPC with 2 Public Subnets, 1 Private Subnet
   2. Corresponding Route Tables (Public RT and Private RT)
2. EC2
   1. This module is used to create 2 EC2 Ubuntu based StrongSwan Servers (1 in each subnet – Public Subnet1, Public Subnet2)
   2. 1 EC2 in Private Subnet (Just for validating the IPsec VPN connectivity)
   3. Security group for EC2 (Allowing SSH/ICMP)
   4. Route Table to attach Strong Swan server as gateway to the other site network
   5. IAM instance profile for StrongSwan Servers (EC2) to allow them to update the route table and to run SSM document

**Note**- VPC Peering is created to enable/simulate connectivity between different VPC

# **Method**

1. Fill the main.tf file
   1. Fill the provider section with the desired region and the IAM instance profile that you have created locally
   2. Since we need to create 2 VPCs for our testing, we have to call “**VPCSite**” and **“EC2**” module twice to create the required resources. Fill the parameters accordingly except in yellow highlighted
   3. Leave the resources “**openssl\_random\_password**” and “**aws\_vpc\_peering\_connection**” as it is

|  |  |  |
| --- | --- | --- |
| **Module Name** | **Variable Name** | **Purpose** |
| **VPCSite** | source | Relative location of VPCSite Module |
| **VPCSite** | site\_name | VPC Site Name |
| **VPCSite** | availability\_zone1 | AZ Name in that region |
| **VPCSite** | availability\_zone2 | AZ Name in that region |
| **VPCSite** | VPC\_CIDR | VPC CIDR |
| **VPCSite** | Public\_CIDR1 | Public CIDR |
| **VPCSite** | Public\_CIDR2 | Public CIDR |
| **VPCSite** | Private\_CIDR1 | Private CIDR |
| **EC2** | source | Relative location of EC2 Module |
| **EC2** | site\_name | Will be passed from previous module |
| **EC2** | EC2\_Size | EC2 Size |
| **EC2** | AMI\_ID | AMI ID (Must be Ubuntu) on that region |
| **EC2** | VPC\_Id | Will be passed from previous module |
| **EC2** | Master\_private\_ip | Master Private IP of EC2 of primary site |
| **EC2** | Follower\_private\_ip | Follower Private IP of EC2 of primary site |
| **EC2** | Pre\_Shared\_Key | Will be passed from previous module |
| **EC2** | Public\_SubnetID1 | Will be passed from previous module |
| **EC2** | Public\_SubnetID2 | Will be passed from previous module |
| **EC2** | Private\_SubnetID1 | Will be passed from previous module |
| **EC2** | Primary\_cidr | Will be passed from previous module |
| **EC2** | Secondary\_cidr | Will be passed from previous module |
| **EC2** | Secondary\_Master\_private\_ip | Master Private IP of EC2 of a secondary site |
| **EC2** | Secondary\_Follower\_private\_ip | Follower Private IP of EC2 of a secondary site |
| **EC2** | Secondary\_Site\_name | Will be passed from previous module |
| **EC2** | Peering\_ID | Will be passed from previous module |

1. Run “**terraform init**”
2. Run “**terraform plan**” -> To check what all resources will be created
3. Run “**terraform apply -auto-approve**” to create all required resources
   1. It will take 2-3 minutes to create all resources
   2. But userdata script in EC2 instance will run for about 10-15 mins, **so its advisable to start your testing after 20 mins.**
   3. You can check the logs of userdata script using “sudo tail -f var/log/cloud-init-output.log”
   4. SSH is only possible via EC2-Connect method
4. Run “**terraform destroy -auto-approve**” to delete all resources. It will take 3-5 mins to finish deleting the resources

# **How Failover between StrongSwan takes place?**

1. Keepalive daemon is configured to monitor the status of IPsec service
2. When IPsec service is down or EC2 is down, then the failover server automatically promoted to Master state
3. During Master state following actions are performed,
   1. SSM document is triggered to run on another site EC2 instances to update their IPSec config files and restart the IPSec service
   2. Private Route Table and Public Route Table will be updated with current instance id for routing