# J. BRADLEY MORGAN

(770) 596-2238 | Acworth, GA | [jmorgan67@gatech.edu](mailto:jmorgan67@gatech.edu) | [www.linkedin.com/in/brad-morgan60](http://www.linkedin.com/in/brad-morgan60)

**AWS PROJECTS & ARCHITECTURES**

**Highly Available Dynamic Site-to-Site VPN**

This project came from an advanced demo series from Adrian Cantrill’s Solutions Architect – Professional course. I was able to complete this demo on my own after working through it multiple times.

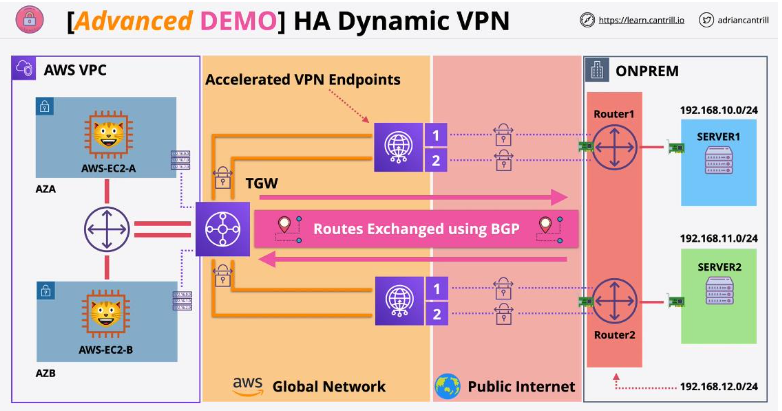
Deployed an AWS VPC and a simulated on-premises environment using a CloudFormation template for each environment. The goal of this project was to connect both environments dynamically while ensuring high availability. The initial AWS environment had two subnets, two EC2 instances, a Transit Gateway (TGW), VPC Attachment, and a default route pointing at the TGW. The initial on-premises environment had one public subnet and two private subnets. The public subnet contained two Ubuntu, one StrongSwan, and Free VPN Endpoints.

For the on-premises environment, two Customer Gateways (CGWs) were created so that if one connection failed, the entire hybrid connection would remain highly available. Traffic flowing through the CGWs would connect to virtual servers created by the CloudFormation template.

During the creation of the VPN, acceleration was enabled. Since a Virtual Private Gateway (VGW) does not support acceleration, a TGW had to be used. Two VPN attachments were configured for the TGW. These attachments were Accelerated VPN Endpoints. The Accelerated VPN Endpoints function over the AWS Global Network and connect to the CGWs located in the on-premises environment. Two VPN configurations were downloaded to be able to exchange routes later on.

Next, the on-premises Ubuntu and StrongSwan routers were configured to create IPSec tunnels to AWS, each going to a different AWS Endpoint. Before this part of the demo began, I had to be sure that the VPN connections were in an “Available” state.

BGP sessions were then enabled on all of the tunnels. BGP runs inside of the tunnels. FRR needed to be installed to enable BGP. Once the IP addresses were configured from the Outside and Inside IP addresses of the VPN tunnels, data would be able to flow to and from both environments (routes exchanged). Enabling BGP for both VPNs took about 15 minutes for each tunnel.



**Lambda Persistent Shared File System Using EFS**

This demo came from Adrian Cantrill’s Solutions Architect – Professional certification course. The goal of this demo was to solidify knowledge on serverless and event-driven architectures. I went through this demo multiple times and was able to perform it by myself.

In this demo, a Lambda function runs in a VPC and extends the storage available within the Lambda runtime environment using Elastic File System (EFS). The demo began with a one-click deployment of a CloudFormation template which was created from the course creator, Adrian Cantrill. The VPC contains two public subnets and two private subnets. The public subnets communicate with the Internet Gateway (IGW) and thus connects to the public internet. NAT Gateways are stored in the public subnets which creates the ability of the private subnets to connect to the public internet securely. EFS access points are stored in the private subnets as well as the one Lambda function.

Event Bridge is used to invoke the Lambda function which is configured to download new cat pictures on the EC2 instances every two minutes. Event Bridge creates an event-driven architecture which performs the invocation of the Lambda function on my behalf, resulting in new cat pictures every two minutes. Because of EFS mount points, the result of the web page from the DNS IPv4 Address was the same on both EC2 instances.

I learned a lot from this demo lesson and gained practical experience on using Lambda, EFS, and Event Bridge. It was a lot of fun!

