



Deploying a Transit Gateway within a Transit VPC on AWS

<http://www.paloaltonetworks.com>

Table of Contents

Version History	3
1. About	4
2. Topology	4
3. Support Policy	6
4. Prerequisites	6
5. Create S3 Buckets for Security VPC	6
6. Deploy the Transit Gateway Stack	9
9. When everything works	13
10. Accessing the Firewall	18
11. Cleanup	20

Version History

Version number	Comments
1.0	Initial GitHub check-in

1. About

This document will explain how to deploy an AWS Transit Gateway as part of a Transit VPC with the VM-Series on AWS. A Transit VPC uses a hub and spoke architecture that allows security teams to centralize secure connectivity for VPC-to-VPC, VPC to corporate and VPC to the internet communications.

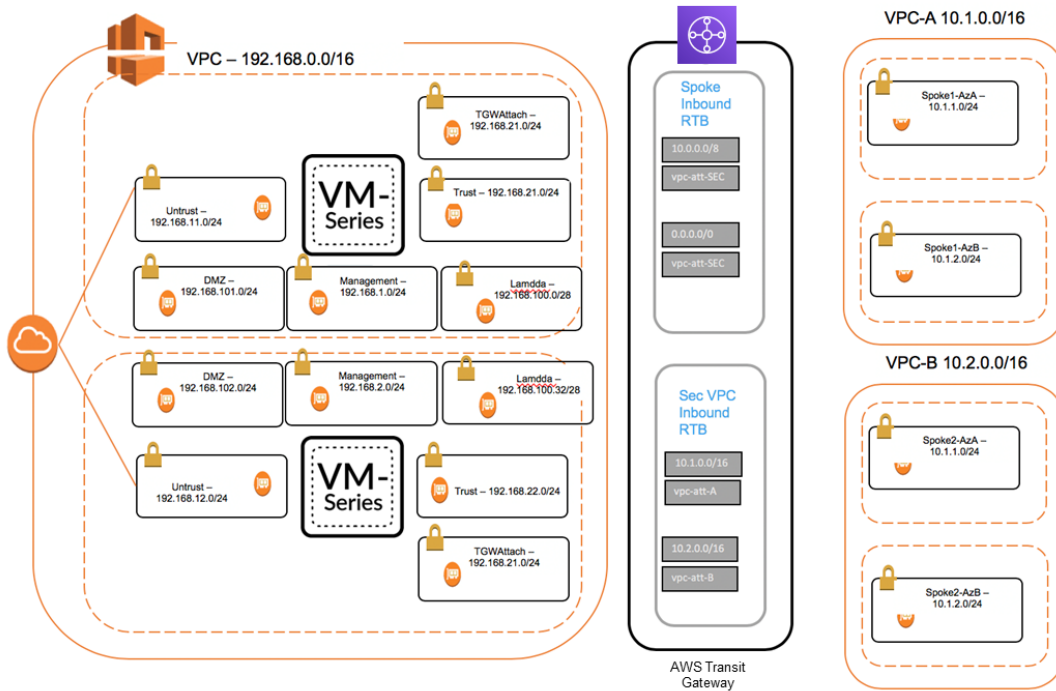
AWS Transit Gateway is a service that enables customers to connect their Amazon Virtual Private Clouds (VPCs) and their on-premises networks to a single gateway. As you grow the number of workloads running on AWS, you need to be able to scale your networks across multiple accounts and Amazon VPCs to keep up with the growth. Today, you can connect pairs of Amazon VPCs using peering. However, managing point-to-point connectivity across many Amazon VPCs, without the ability to centrally manage the connectivity policies, can be operationally costly and cumbersome. For on-premises connectivity, you need to attach your AWS VPN to each individual Amazon VPC. This solution can be time consuming to build and hard to manage when the number of VPCs grows into the hundreds.

With an AWS Transit Gateway, you only have to create and manage a single connection from the central gateway in to each VPC, on-premises data center, or remote office across your network. Transit Gateway acts as a hub that controls how traffic is routed among all the connected networks which act like spokes. This hub and spoke model simplifies management and reduces operational costs because each network only connects to the Transit Gateway and not to every other network. Any new VPC is simply connected to the Transit Gateway and is then automatically available to every other network that is connected to the Transit Gateway. This ease of connectivity makes it easy to scale your network as you grow.

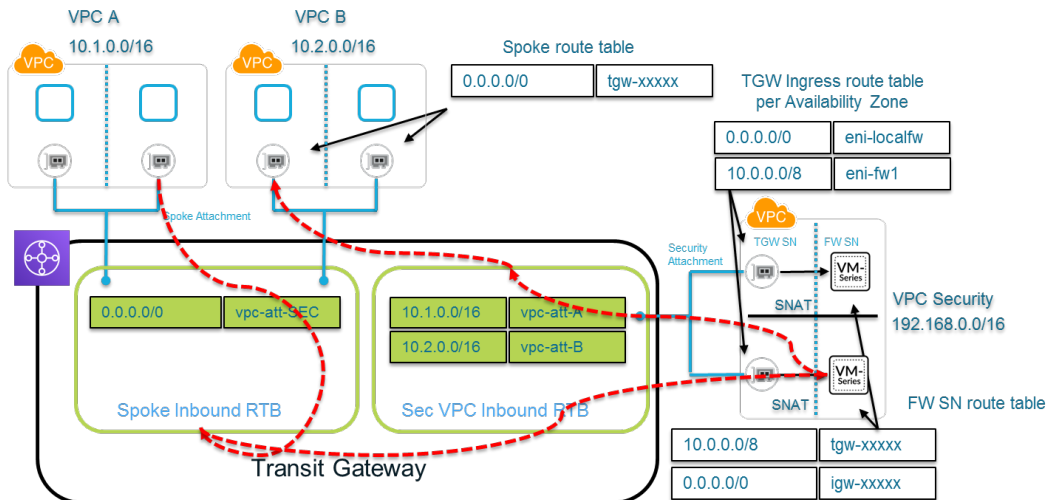
More information on the AWS Transit Gateway can be found here: <https://aws.amazon.com/transit-gateway/>

2. Topology

The topology below displays the VPCs and subnets that will be deployed. The Security or Firewall hub will provide security inspection and connectivity while the Transit Gateway will facilitate communications for the application VPCs. This topology is deployed in a single account:



VPC INSERTION



3. Support Policy

This solution is released under an as-is, best effort, support policy. These scripts should be seen as community supported and Palo Alto Networks will contribute our expertise as and when possible. We do not provide technical support or help in using or troubleshooting the components of the project through our normal support options such as Palo Alto Networks support teams, or ASC (Authorized Support Centers) partners and backline support options. The underlying product used (the VM-Series firewall) by the scripts or templates are still supported, but the support is only for the product functionality and not for help in deploying or using the template or script itself.

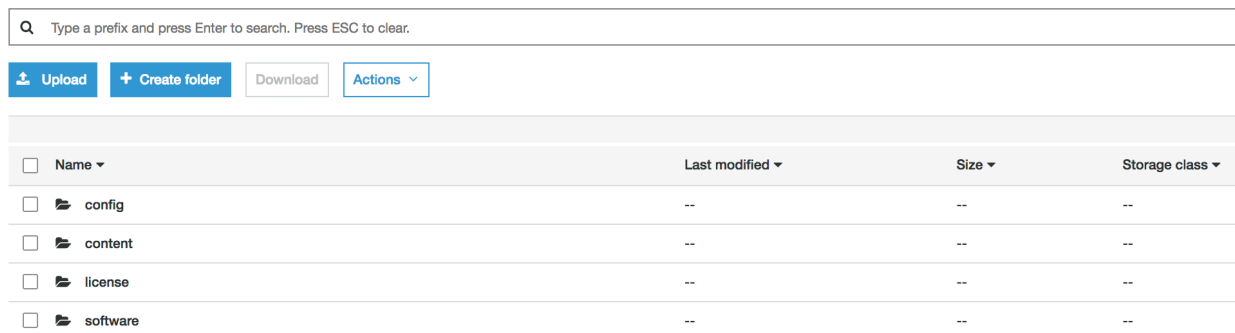
4. Prerequisites

Here are the prerequisites required to successfully launch this template:

1. AWS account
2. Clone or download the files from the following GitHub repository on to your local machine:
<https://github.com/jharris10/transitgateway>

5. Create S3 Buckets for Security VPC

In the AWS S3 console, create an S3 bucket with config, content, license and software folders.



The screenshot shows the AWS S3 console interface. At the top, there is a search bar with the text "Type a prefix and press Enter to search. Press ESC to clear." Below the search bar, there are four buttons: "Upload", "Create folder", "Download", and "Actions" with a dropdown arrow. Below the buttons, there is a table with four columns: "Name", "Last modified", "Size", and "Storage class". The table contains four rows, each representing a folder: "config", "content", "license", and "software". Each row has a checkbox on the left and "--" in the other three columns.

<input type="checkbox"/> Name ▾	Last modified ▾	Size ▾	Storage class ▾
<input type="checkbox"/> config	--	--	--
<input type="checkbox"/> content	--	--	--
<input type="checkbox"/> license	--	--	--
<input type="checkbox"/> software	--	--	--

In the config folder add the VM-Series bootstrap.xml and init-cfg.txt files from the cloned repositories /bootstrap folder

jharris10 / transitgateway

Unw

<> Code

! Issues 0

🔗 Pull requests 0

📁 Projects 0

📖 Wiki

📊 Insights

⚙️ S

Branch: master ▾
transitgateway / bootstrap /

Create

panwce added bootstrap directory

..

bootstrap.xml
added bootstrap directory

init-cfg.txt
added bootstrap directory

Either create another S3 bucket or use the existing bucket and add the Lambda.zip and TransitGatewayRouteMonitorLambda.zip files from the Lambda directory in the cloned repository into this bucket.

Amazon S3 > jrh-tgw-boot

Overview

Properties

Permissions

Management

Type a prefix and press Enter to search. Press ESC to clear.

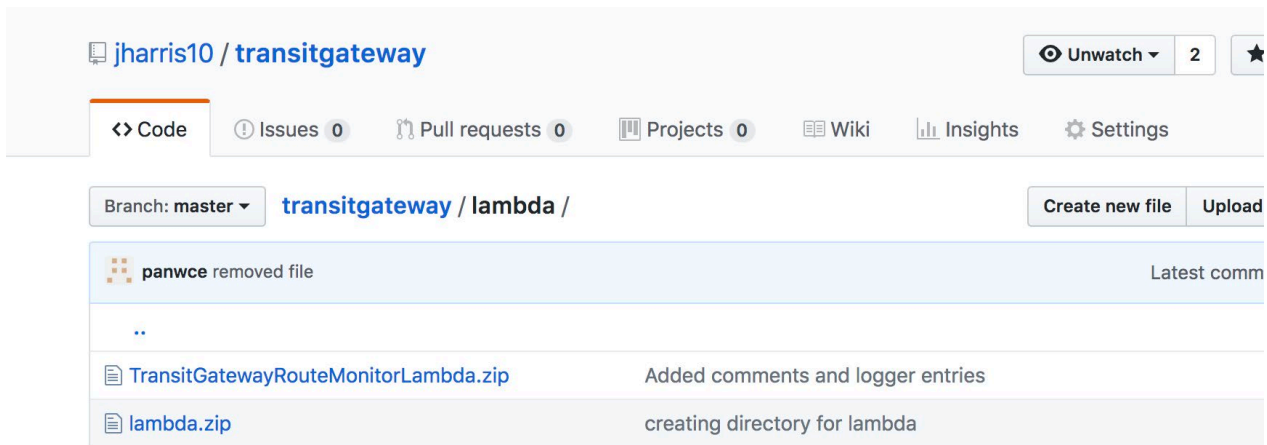
Upload

Create folder

Download

Actions ▾

<input type="checkbox"/> Name ▾	Last modified ▾	Size ▾	Storage class
<input type="checkbox"/> config	--	--	--
<input type="checkbox"/> content	--	--	--
<input type="checkbox"/> license	--	--	--
<input type="checkbox"/> software	--	--	--
<input type="checkbox"/> TransitGatewayRouteMonitorLambda.zip	Jan 23, 2019 11:40:29 PM GMT+0200	2.2 KB	Standard
<input type="checkbox"/> lambda.zip	Jan 8, 2019 9:31:18 PM GMT+0200	7.7 MB	Standard



Note: The buckets need to be in the same region in which you will deploy the Transit Gateway template.

6. Deploy the Transit Gateway Stack

In the AWS CloudFormation console create a new stack and select the Transit-Gateway-Demo-v0.5.yaml template and fill in the parameters

Stacks > Create stack

Create stack

Prerequisite - Prepare template

Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.


☒ Template is ready
 ☐ Use a sample template
 ☐ Create template in Designer


Specify template
A template is a JSON or YAML file that describes your stack's resources and properties.

Template source
Selecting a template generates an Amazon S3 URL where it will be stored.

☐ Amazon S3 URL
 ☒ Upload a template file

Upload a template file

Choose file  Transit-Gateway-Demo-v0.5.yaml
JSON or YAML formatted file

S3 URL: <https://s3-external-1.amazonaws.com/cf-templates-1d9wa1n0rrklv-us-east-1/2019028eo7-Transit-Gateway-Demo-v0.5.yaml> [View in Designer](#) 

Cancel **Next**

1. Route Monitor Configuration

- The first section configures the behaviour of the route failover Lambda function.
- RouteFailover – Setting this to true will return the route table to the original configuration where firewall 1 is used for internet connections and firewall 2 is used for east/west connections.
- SplitRoutes – Setting this value to false will result in a single firewall handling both internet connections and east/west connections

Parameters
Parameters are defined in your template and allow you to input custom values when you create or update a stack.

Route Monitor Configuration

RouteFailover
Restore original route table entries when device recovers

false ▼

splitroutes
Share routing across both firewalls FW1 for Internet FW2 for east/west

true ▼

Sumamry Route for spoke VPCs the template assumes a 10.0.0.0/8 block
Summary route for spoke vpcs for example 10.0.0.0/8

10.0.0.0/8

1. Security VPC Subnet Configuration

The template assumes that the default subnet values are used. It may be necessary to modify the existing bootstrap.xml file in the bootstrap folder if the network configuration changes.

Security VPC Subnet Configuration

VpcAZs

Select 2 AZs

CIDR Block for the VPC

Enter the VPC CIDR that you want to use

VPC Name

Name of the newly created Security VPC

Management Subnet CIDR Block

Management subnet comma-delimited list of CIDR blocks

LambdaSubnetipBlocks

Management subnet comma-delimited list of CIDR blocks

Trust Subnet CIDR Block

Trust subnet comma-delimited list of CIDR blocks

Untrust Subnet CIDR Block

Untrust subnet comma-delimited list of CIDR blocks

NAT Gateway Subnet CIDR Block

AWS NAT Gateway Comma-delimited list of CIDR blocks

Transit Gateway Subnet Attachment Block

AWS NAT Gateway Comma-delimited list of CIDR blocks

2. Lambda Configuration

Enter the name of the Lambda zip files you uploaded previously and the name of the S3 bucket they are stored in. Note: The S3 bucket must be in the same region as the stack deployment.

Lambda Configuration

LambdaZipFile

Lambda code zip filename which is stored in above mentioned Required parameters LambdaFunctionsBucketName

RouteMonitorLambdaZipFile

Lambda code zip filename which is stored in above mentioned Required parameters LambdaFunctionsBucketName

LambdaFunctionsBucketName

Existing S3 bucket name which contains the Lambda funtions zip file

Bootstrap Configuration

S3 bucket containing the bootstrap folders and files

Enter the name S3 Bucket Name containing the Bootstrap files

3. Other Parameters

Complete the remaining fields and Click through to kick of stack creation

Other parameters

FWInstanceType
Enter the instance type and size for the VM-Series firewall

FWLicenseType
Enter the license type for the Firewall

KeyName
Amazon EC2 Key Pair

NatInstanceType
Instance type to use for NAT

SSHLocation
Restrict SSH & HTTPS access to the Web Servers (by default can be accessed from anywhere)

WebInstanceType
WebServer EC2 instance type

Cancel
Previous
Next

Click through to kick of stack creation.

You should see a stack create complete when the transit VPC account has been successfully initialized.

CloudFormation > Stacks > TGW-Example: Stack details

TGW-Example

Actions

Stack info | **Events** | Resources | Outputs | Parameters | Template

Events

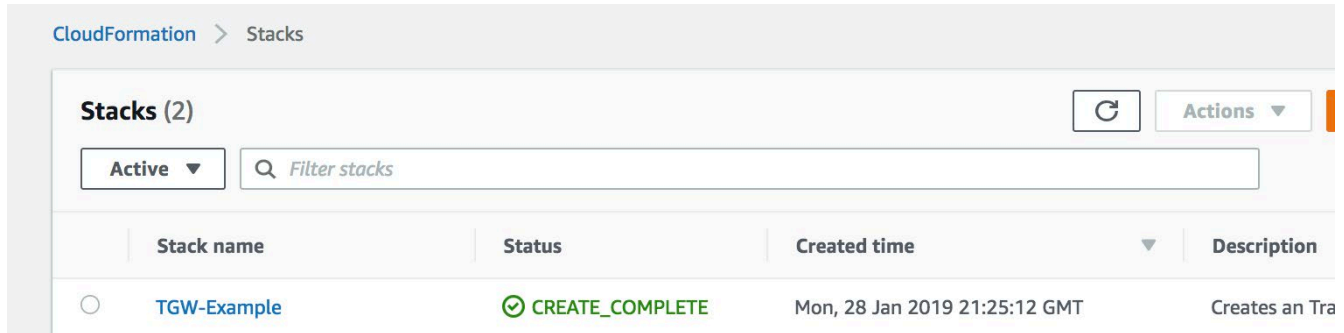
Timestamp	Logical ID	Status	Status reason
28 Jan 2019 21:29:31	TGW-Example	CREATE_COMPLETE	-
28 Jan 2019 21:29:28	TransitGatewayRouteMoni torLambda	CREATE_COMPLETE	-
28 Jan 2019 21:29:27	TransitGatewayRouteMoni		Resource creation Initiated

Stacks (2)

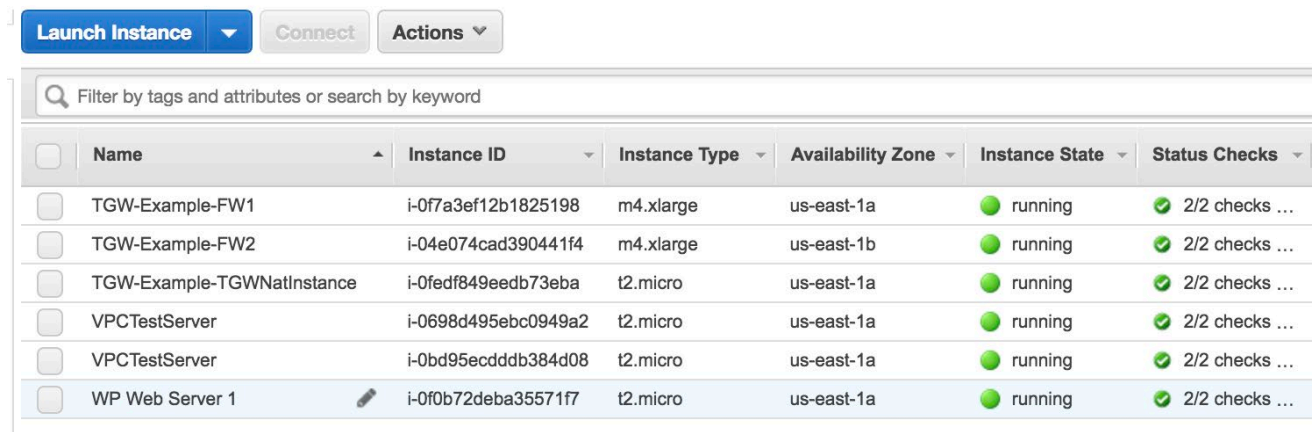
	Stack name	Status	Created time	Description
	TGW-Example	CREATE_COMPLETE	Mon, 28 Jan 2019 21:25:12 GMT	Creates an Tra

9. When everything works

.You should see a “create_complete” status if the template had deployed correctly



In the screenshot below, you can see a pair of VM-Series firewalls, depicted as TGW-Example FW1 and FW2:



Verify the Setup

Lambda Functions

The TransitGatewayInitialiseLambda function updates the VPC subnet route tables with the next hop of the transit gateway. We use Lambda to create these routes as at this time we cannot accomplish this with the CloudFormation Template. The function also starts a lambda step function that runs some post deployment configuration tasks on the firewalls. In this case its sets a static route for the spoke VPCs and updates and address object with the firewalls untrust IP assigned by AWS.

[Log Groups](#) > [aws/lambda/TGW-ExampleGatewayInitialiseLambda-18YRTMTN5S3P](#) > 2019/01/28/[[\\$LATEST](#)]253fc43a0b1d4ebe975b71b4d40511fb

[Expand all](#) ☒ Row ☐ Text

Routes Added to VPC Route Tables

all 2019-01-27 [21:28]

TC +00:00)	Message
28	No older events found at the moment. Retry
	START RequestId: 22ef9091-5576-4d77-bdd3-44007478cfb9 Version: \$LATEST
	[INFO] 2019-01-28T21:28:56.154Z 22ef9091-5576-4d77-bdd3-44007478cfb9 Got event {'RequestType': 'Create', 'ServiceToken': 'arn:aws:lambda:us-east-1:106808901910:function:TGW-E'}.
	[INFO] 2019-01-28T21:28:56.495Z 22ef9091-5576-4d77-bdd3-44007478cfb9 Found credentials in environment variables.
	[INFO] 2019-01-28T21:28:57.665Z 22ef9091-5576-4d77-bdd3-44007478cfb9 Got response to add_route_tgw_nh {'Return': True, 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'Headers': {'x-amzn-trace-id': 'Root=1-5c4e975b-1b4d40511fb'}, 'MaxConnectionsPerHost': 100, 'RequestContentLength': 100, 'RestAPIID': '...', 'Service': 'Route53', 'Status': 200, 'Version': '2015-12-01'}}.
	[INFO] 2019-01-28T21:28:57.666Z 22ef9091-5576-4d77-bdd3-44007478cfb9 Got response to route update on VPC0 {'Return': True, 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'Headers': {'x-amzn-trace-id': 'Root=1-5c4e975b-1b4d40511fb'}, 'MaxConnectionsPerHost': 100, 'RequestContentLength': 100, 'RestAPIID': '...', 'Service': 'EC2', 'Status': 200, 'Version': '2016-04-01'}}.
	[INFO] 2019-01-28T21:28:58.772Z 22ef9091-5576-4d77-bdd3-44007478cfb9 Got response to add_route_tgw_nh {'Return': True, 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'Headers': {'x-amzn-trace-id': 'Root=1-5c4e975b-1b4d40511fb'}, 'MaxConnectionsPerHost': 100, 'RequestContentLength': 100, 'RestAPIID': '...', 'Service': 'Route53', 'Status': 200, 'Version': '2015-12-01'}}.
	[INFO] 2019-01-28T21:28:58.792Z 22ef9091-5576-4d77-bdd3-44007478cfb9 Got response to route update on VPC1 {'Return': True, 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'Headers': {'x-amzn-trace-id': 'Root=1-5c4e975b-1b4d40511fb'}, 'MaxConnectionsPerHost': 100, 'RequestContentLength': 100, 'RestAPIID': '...', 'Service': 'EC2', 'Status': 200, 'Version': '2016-04-01'}}.
	[INFO] 2019-01-28T21:28:58.523Z 22ef9091-5576-4d77-bdd3-44007478cfb9 Got response to add_route_tgw_nh {'Return': True, 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'Headers': {'x-amzn-trace-id': 'Root=1-5c4e975b-1b4d40511fb'}, 'MaxConnectionsPerHost': 100, 'RequestContentLength': 100, 'RestAPIID': '...', 'Service': 'Route53', 'Status': 200, 'Version': '2015-12-01'}}.
	[INFO] 2019-01-28T21:28:58.524Z 22ef9091-5576-4d77-bdd3-44007478cfb9 Got response to route update on SecVPC {'Return': True, 'ResponseMetadata': {'RequestId': '...', 'HTTPStatusCode': 200, 'Headers': {'x-amzn-trace-id': 'Root=1-5c4e975b-1b4d40511fb'}, 'MaxConnectionsPerHost': 100, 'RequestContentLength': 100, 'RestAPIID': '...', 'Service': 'EC2', 'Status': 200, 'Version': '2016-04-01'}}.
	https://cloudformation-custom-resource-response-useast1.s3.amazonaws.com/arn%3Aaws%3Acloudformation%3Aus-east-1%3A106808901653%3Astack-TGW-E
	Response body:
	{'Status': 'SUCCESS', 'Reason': 'See the details in CloudWatch Log Stream: 2019/01/28/[LATEST]253fc43a0b1d4ebe975b71b4d40511fb', 'PhysicalResourceId': '...'}
	[INFO] 2019-01-28T21:28:58.528Z 22ef9091-5576-4d77-bdd3-44007478cfb9 Starting new HTTPS connection (1): cloudformation-custom-resource-response-useast1
	Status code: OK
	[INFO] 2019-01-28T21:28:58.748Z 22ef9091-5576-4d77-bdd3-44007478cfb9 Got response to cfnSend None
	END RequestId: 22ef9091-5576-4d77-bdd3-44007478cfb9
	REPORT RequestId: 22ef9091-5576-4d77-bdd3-44007478cfb9 Duration: 2594.92 ms Billed Duration: 2600 ms Memory Size: 128 MB Max Memory Used: 43 MB
	No newer events found at the moment. Retry .

<input type="text" value="search : TGW-Example-rt-ToTGW"/>	<input type="button" value="Add filter"/>				
<input type="checkbox"/>	Name	Route Table ID	Explicitly Associated with	Main	VPC ID
<input checked="" type="checkbox"/>	TGW-Example-rt-ToTGW	rtb-0a013098dfd201ee8	2 subnets	No	vpc-09f9d7f2

```

22:07 [INFO] 2019-02-05T11:22:07.995Z c8f6cb32-f1c5-4a01-ac47-50e39af41154 Got response to add_route_tgw_nh {'Return': True, 'ResponseMetadata': {'
22:07 [INFO] 2019-02-05T11:22:07.996Z c8f6cb32-f1c5-4a01-ac47-50e39af41154 Got response to route update on SecVPC {'Return': True, 'ResponseMetadata': {'
22:08 [INFO] 2019-02-05T11:22:08.387Z c8f6cb32-f1c5-4a01-ac47-50e39af41154 StateMachine is not Running, hence starting StepFunction
22:08 [INFO] 2019-02-05T11:22:08.457Z c8f6cb32-f1c5-4a01-ac47-50e39af41154 Calling start state function None
22:08 https://cloudformation-custom-resource-response-euwest1.s3-eu-west-1.amazonaws.com/am%3Aaws%3Acloudformation%3Aeu-west-1%3A1068085
22:08 Response body:

```

Page 14

The step function runs the InitialiseFWLambda function

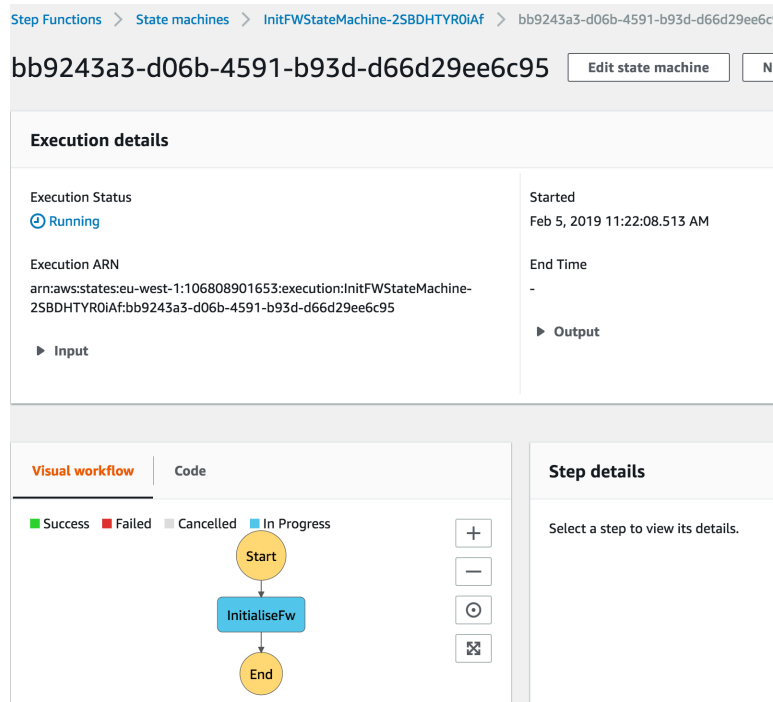


Figure 4 Step Function Status

When the step function completes you will see the status update

Step Functions > State machines > InitFWStateMachine-2SBDHTYR0iAf

InitFWStateMachine-2SBDHTYR0iAf

Details

ARN
arn:aws:states:eu-west-1:106808901653:stateMachine:InitFWStateMachine-2SBDHTYR0iAf

IAM role ARN
arn:aws:iam::106808901653:role/StateMachineExecutionRole-Tuesday2

Creation date
Feb 5, 2019 11:21:52.261 AM

Executions | Definition | Tags

Executions (2)

Search for executions

Name	Status	Started
bb9243a3-d06b-4591-b93d-d66d29ee6c95	Succeeded	Feb 5, 2019 11:22:08.513 AM
4b8377b9-6326-4f4b-9e7f-ae0a09cdd19a	Succeeded	Feb 5, 2019 11:22:08.375 AM

Figure 5 Successful Step Function

CloudWatch > Log Groups > /aws/lambda/Tuesday2-InitialiseFwLambda-67T5L8YATU05 > 2019/02/05/[\$LATEST]1a9bb2272e52484d8a725b10ac6919d3

Expand all | Row | Text

Filter events

Valid Response to "show chassis status" API call

Time (UTC +00:00)	Message
2019-02-05	No older events found at the moment. Retry.
11:33:08	[INFO] 2019-02-05T11:33:08.669Z Found credentials in environment variables.
11:33:08	START RequestId: 8cef7e3f-828d-4173-aa9c-832b39a75d71 Version: \$LATEST
11:33:08	[INFO] 2019-02-05T11:33:08.722Z 8cef7e3f-828d-4173-aa9c-832b39a75d71 Querying for subnet
11:33:08	[INFO] 2019-02-05T11:33:08.746Z 8cef7e3f-828d-4173-aa9c-832b39a75d71 Starting new HTTPS connection (1): ec2.eu-west-1.amazonaws.com
11:33:08	[INFO] 2019-02-05T11:33:08.914Z 8cef7e3f-828d-4173-aa9c-832b39a75d71 Querying for subnet
11:33:08	[INFO] 2019-02-05T11:33:08.969Z 8cef7e3f-828d-4173-aa9c-832b39a75d71 Got Event {}
11:33:08	[INFO] 2019-02-05T11:33:08.986Z 8cef7e3f-828d-4173-aa9c-832b39a75d71 [INFO]: Sending command: <urlib.request.Request object at 0x7efd72dc3eb8>
11:33:09	[INFO] 2019-02-05T11:33:09.259Z 8cef7e3f-828d-4173-aa9c-832b39a75d71 [INFO]: FW is up!
11:33:09	[INFO] 2019-02-05T11:33:09.266Z 8cef7e3f-828d-4173-aa9c-832b39a75d71 [RESPONSE]: b'<response status="success"><result><![CDATA[yes]]></result></response>'
11:33:09	[INFO] 2019-02-05T11:33:09.266Z 8cef7e3f-828d-4173-aa9c-832b39a75d71 [INFO]: FW is ready for configure
11:33:09	[INFO] 2019-02-05T11:33:09.266Z 8cef7e3f-828d-4173-aa9c-832b39a75d71 [INFO]: FW is up
11:33:09	[INFO] 2019-02-05T11:33:09.666Z 8cef7e3f-828d-4173-aa9c-832b39a75d71 [INFO]: Sending command: <urlib.request.Request object at 0x7efd72dd7a58>
11:33:10	[INFO] 2019-02-05T11:33:10.786Z 8cef7e3f-828d-4173-aa9c-832b39a75d71 [INFO]: FW is up!
11:33:10	[INFO] 2019-02-05T11:33:10.786Z 8cef7e3f-828d-4173-aa9c-832b39a75d71 [RESPONSE]: b'<response status="success"><result><![CDATA[yes]]></result></response>'
11:33:10	[INFO] 2019-02-05T11:33:10.786Z 8cef7e3f-828d-4173-aa9c-832b39a75d71 [INFO]: FW is ready for configure
11:33:10	[INFO] 2019-02-05T11:33:10.786Z 8cef7e3f-828d-4173-aa9c-832b39a75d71 [INFO]: FW is up
11:33:11	END RequestId: 8cef7e3f-828d-4173-aa9c-832b39a75d71
11:33:11	REPORT RequestId: 8cef7e3f-828d-4173-aa9c-832b39a75d71 Duration: 2584.27 ms Billed Duration: 2600 ms Memory Size: 128 MB Max Memory Used: 41 MB

Figure 6 Output from InitialiseFwLambda function

Once the stack build has completed go to the output tab where you will find relevant information regarding IP address allocations

Stack ID: **stack-2**

Stack info | Events | Resources | **Outputs** | Parameters | Template

Outputs (7)

🔍 Search outputs

Key ▲	Value ▼	Description
Fw1MgmtIP	192.168.1.137	Firewall 1 Untrust Interface Public IP
Fw1PublicIP	54.76.185.112	Firewall 1 Untrust Interface Public IP
Fw2MgmtIP	192.168.2.221	Firewall 2 Untrust Interface Public IP
Fw2PublicIP	34.255.165.210	Firewall 1 Untrust Interface Public IP
KeyName	AWS-Ireland	Key Pair you have selected for SSH
NATInstancePublicIp	52.16.157.178	NAT Instance Public IP
VPCID	vpc-04588e88f41edc152	VPC ID

Figure 7 Stack Output with IP Address Allocation

10. Accessing the Firewall

In order to access the firewall's web UI or the CLI, it is recommended that you use the NAT instance that has been deployed in the Transit VPC. In order to do that you will need to setup an SSH tunnel from your localhost to the remote NAT instance.

For Web UI:

```
$ssh -i <AWS SSH key> -l ec2-user <public IP address of NAT instance> -L 4000:<private IP address of fw eth0>:443 -nNtv
```

You can then point your browser to <https://localhost:4000> For CLI:

```
$ssh -i <AWS SSH key> -l ec2-user <public IP address of NAT instance> -L 4000:<private IP address of fw eth0>:22 -nNtv
```

You can now ssh to localhost:4001 using the panadmin/Pal0Alt0123! credentials.

```
$ssh admin@localhost -p 4001
```

NOTE: You can use any port of you choosing other than 4000

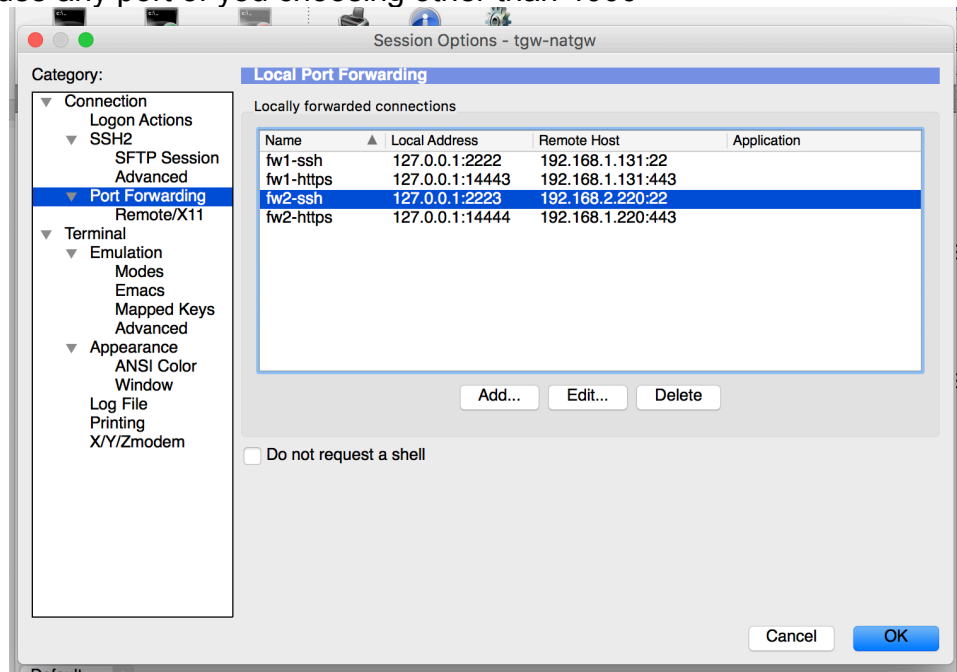


Figure 8 SecureCRT with Port Forwarding

11. Testing the Deployment

1) Login to the firewalls via a browser.

Verify that the Lambda function has updated the firewall with a route to the spoke VPCs and updated the untrust interface address object.

Client	Command	Result	Configuration Path	Full Path	Before Change	After Change	Seq
Web	edit	Succeed...	vsys vsys1 address Fw-Untrust-Int ip-netmask	/config/devices/entry[... Untrust-Int']/ip-netmask	ip-netmask 192.168.11.97;	ip-netmask 192.168.11.97;	4
Web	edit	Succeed...	vsys vsys1 address Fw-Untrust-Int ip-netmask	/config/devices/entry[... Untrust-Int']/ip-netmask	ip-netmask 192.168.11.203;	ip-netmask 192.168.11.97;	3
Web	set	Succeed...	network virtual-router default routing-table ip static-route vnets	/config/devices/entry[... router/entry[@name='... table/ip/static-route/entry[@name='v...		vnets { destination 10.0.0.0/8; interface ethernet1/2; nexthop {	2
Web	set	Succeed...	network virtual-router default routing-table ip static-route vnets	/config/devices/entry[... router/entry[@name='... table/ip/static-route/entry[@name='v...		routing-table { ip { static-route { vnets { destination 10.0.0.0	1

Figure 9 Configuration log showing firewall updates from Lambda

Figure 10 Firewall Changes From Lambda API calls

2) Test Connectivity

The firewall is configured with a NAT rule for ssh access to the server and Web Access to the test server on nonstandard ports.

The test url is <http://<fwpublicip>/index.php>

The firewall public ip can be obtained from the stack output

CloudFormation > Stacks > Wednesday4: Stack details			
Wednesday4			
Stack Info	Events	Resources	Outputs
Outputs (7)			
Search outputs			
Key	Value	Description	
Fw1MgmtIP	192.168.1.19	Firewall 1 Untrust Interface Public IP	
Fw1PublicIP	192.168.1.19	Firewall 1 Untrust Interface Public IP	
Fw2MgmtIP	192.168.2.212	Firewall 2 Untrust Interface Public IP	
Fw2PublicIP	192.168.2.212	Firewall 2 Untrust Interface Public IP	
KeyName	AWS-Ireland	Key Pair you have selected for SSH	
NATInstancePublicIp	192.168.1.19	NAT Instance Public IP	
VPCID	vpc-081a2b4bac3e2e2e2	VPC ID	

Figure 11 Stack output

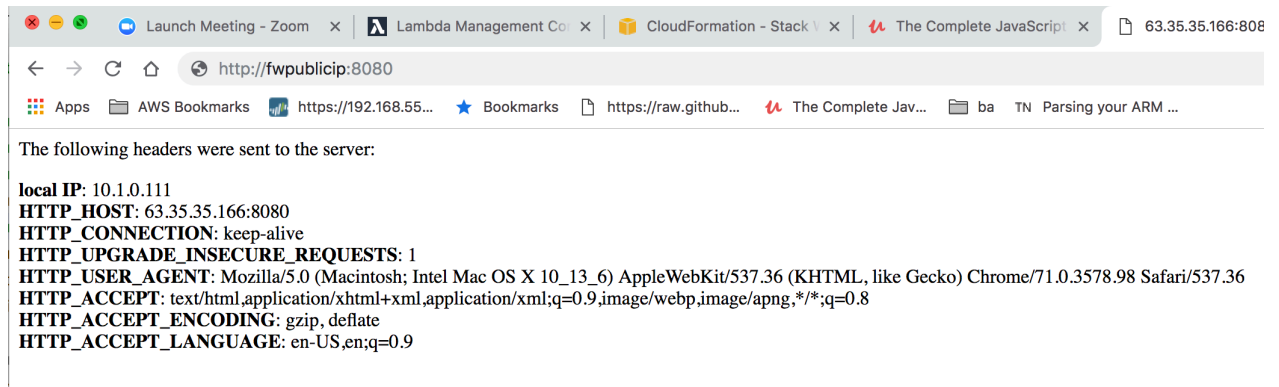


Figure 12 Server test page

12. Cleanup

You can clean up the setup by deleting the stack deployed. You may have to manually delete some resources that were created by Lambda functions.