Cloud Security Framework

AWS Transit Gateway Direct Attach Firewalls Deployment Guide

This document will detail how to install the AWS Transit Gateway Intgration demo. The security framework will use Panhandler as the deployment tool. The following prerequisites are required to be installed on your local computer prior to the installation of the Cloud Security Framework .

Prerequisites:

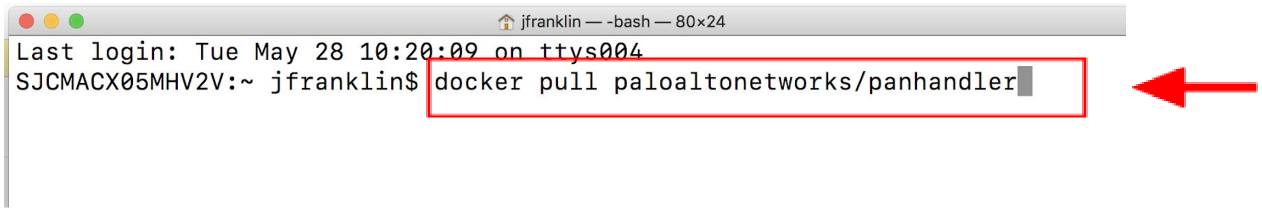
Docker:
Docker will need to be installed to run Panhandler. Here are some links to help with the install:

For Mac:
https://docs.docker.com/docker-for-mac/install/

For Windows:
https://docs.docker.com/docker-for-windows/install/

Panhandler:
Once Docker is installed and running, you will need to pull the panhandler container image from docker hub and build a Panhandler container. From a Mac terminal window or a PC powershell window run the following commands:

docker pull paloaltonetworks/panhandler

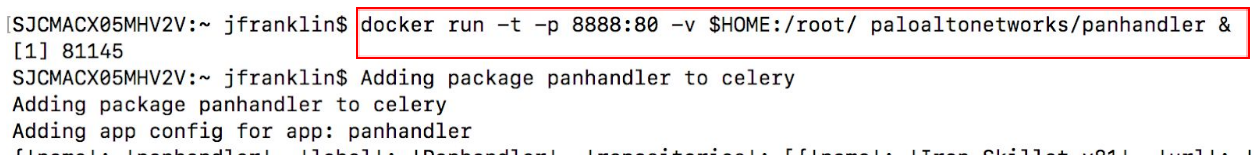


```
Last login: Tue May 28 10:20:09 on ttys004
SJCMACX05MHV2V:~ jfranklin$ docker pull paloaltonetworks/panhandler
```

After panhandler image is downloaded from Docker Hub, start Panhandler using the following commands:

Mac command:
docker run -t -p 8888:80 -v $HOME:/root/ paloaltonetworks/panhandler &

Windows command:
docker run -t -p 8888:80 -v /c/Users/%USERNAME%:/root/ paloaltonetworks/panhandler

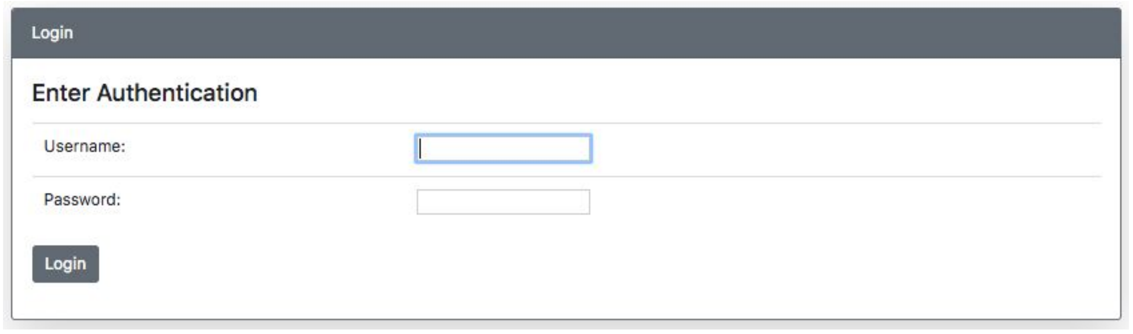


```
[SJCMACX05MHV2V:~ jfranklin$ docker run -t -p 8888:80 -v $HOME:/root/ paloaltonetworks/panhandler &
[1] 81145
SJCMACX05MHV2V:~ jfranklin$ Adding package panhandler to celery
Adding package panhandler to celery
Adding app config for app: panhandler
```

At this point you should have a container running on your machine with Panhandler. To access Panhandler, open a browser and navigate to the following URL:

http://localhost:8888

You should see the following login page.

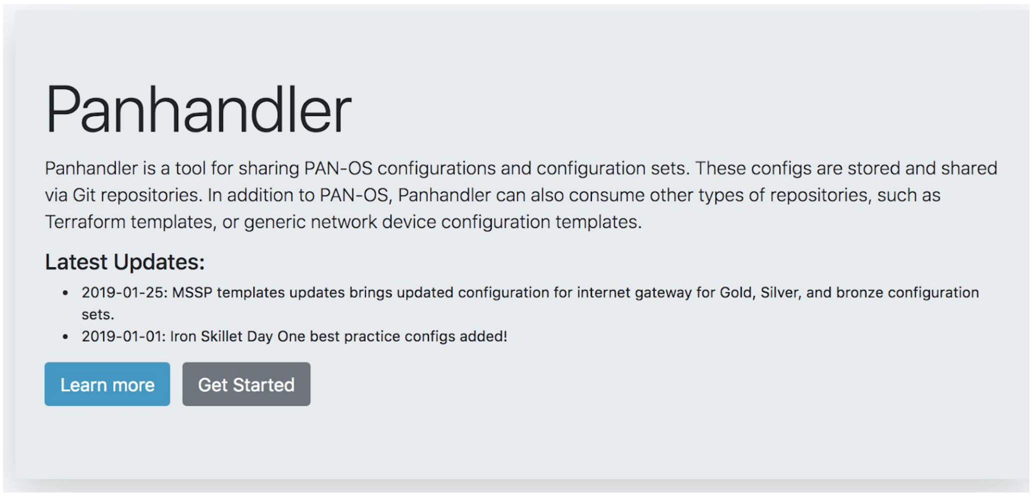


Default login username and password are:

Username: paloalto

Password: panhandler

Once you are logged in, you will get a welcome screen



Panhandler

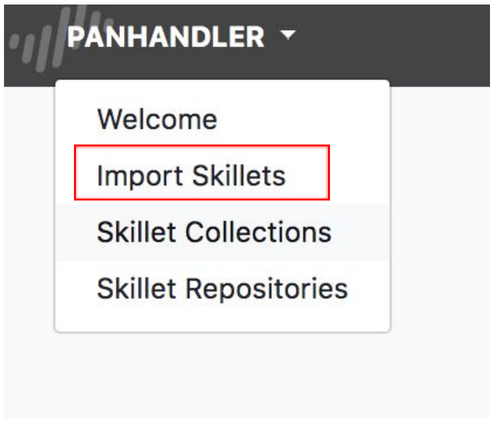
Panhandler is a tool for sharing PAN-OS configurations and configuration sets. These configs are stored and shared via Git repositories. In addition to PAN-OS, Panhandler can also consume other types of repositories, such as Terraform templates, or generic network device configuration templates.

Latest Updates:

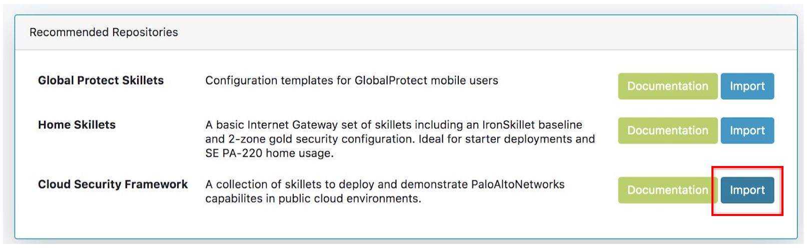
- 2019-01-25: MSSP templates updates brings updated configuration for internet gateway for Gold, Silver, and bronze configuration sets.
- 2019-01-01: Iron Skillet Day One best practice configs added!

Deployment:

In the upper left hand corner, you will see the Panhandler drop down. Click on the drop down and navigate to the "Import Skillets" option.



To import the Cloud Security Framework, you can select the import button from the screen below.

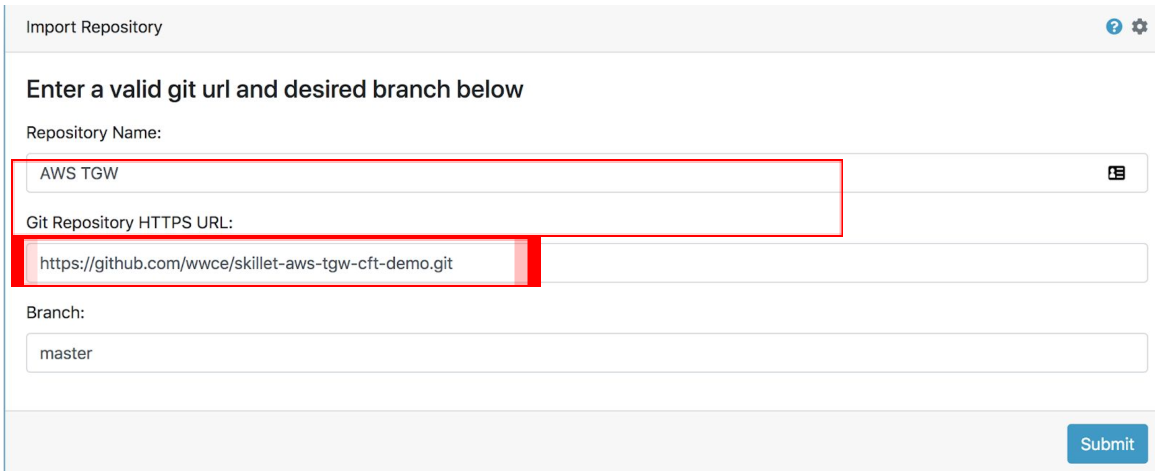


Recommended Repositories

Global Protect Skillets	Configuration templates for GlobalProtect mobile users	Documentation Import
Home Skillets	A basic Internet Gateway set of skillets including an IronSkillet baseline and 2-zone gold security configuration. Ideal for starter deployments and SE PA-220 home usage.	Documentation Import
Cloud Security Framework	A collection of skillets to deploy and demonstrate PaloAltoNetworks capabilites in public cloud environments.	Documentation Import

Or you can manually add the GitHub Repository for the Security framework. Once the Repository Name and Git Repository URL are added, select submit:

GitHub URL: https://github.com/wwce/skillet-aws-tgw-cft-demo


Import Repository

Enter a valid git url and desired branch below

Repository Name:

AWS TGW

Git Repository HTTPS URL:

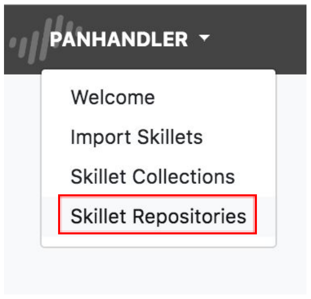
https://github.com/wwce/skillet-aws-tgw-cft-demo.git

Branch:

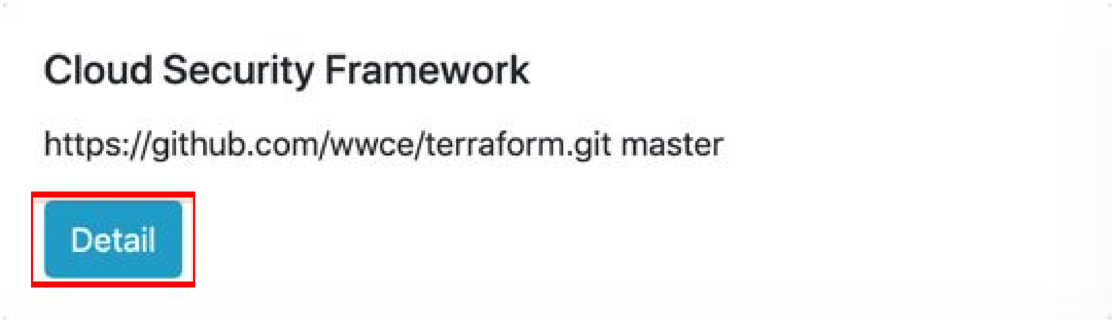
master

Updates

Repositories are periodically updated to include bug fixes or additional functionality. After the initial import, periodically updating the skillets to the latest version is recommended. From the upper left hand corner Panhandler drop down list, select “Skillet Repositories”

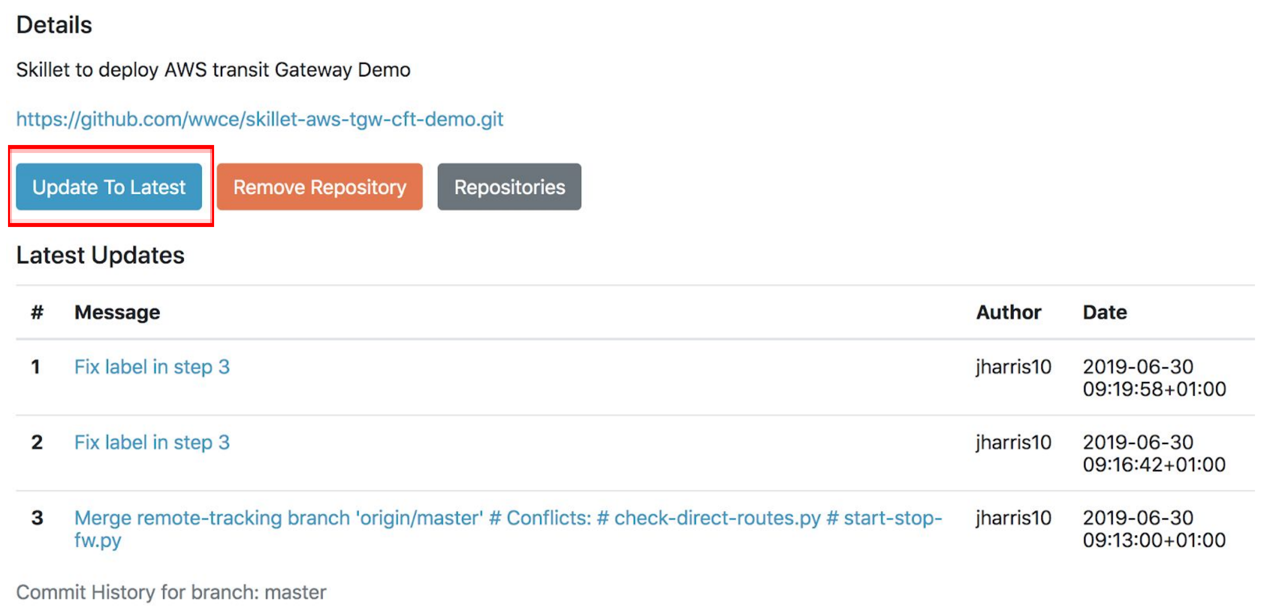


Select Detail under “Cloud Security Framework”:


Cloud Security Framework

https://github.com/wwce/terraform.git master

Select Update to Latest:


Details

Skillet to deploy AWS transit Gateway Demo

https://github.com/wwce/skillet-aws-tgw-cft-demo.git

Latest Updates

#	Message	Author	Date
1	Fix label in step 3	jharris10	2019-06-30 09:19:58+01:00
2	Fix label in step 3	jharris10	2019-06-30 09:16:42+01:00
3	Merge remote-tracking branch 'origin/master' # Conflicts: # check-direct-routes.py # start-stop-fw.py	jharris10	2019-06-30 09:13:00+01:00

Commit History for branch: master

Transit Gateway Introduction

Transit Gateway

The AWS Transit Gateway (TGW) service, enables an organization to easily scale connectivity across thousands of VPCs, AWS accounts and on-premises networks. It simplifies connectivity of multiple VPCs with the ability to connect to a single gateway rather than peering between each of the VPCs. The TGW acts as a hub to control traffic routed between spoke VPCs. In order to fully understand how the TGW works it is important to understand these key concepts:

- Transit Gateway—TGW is a hub which consists of attachments and route tables. The TGW supports VPC attachments as either VPC or VPN attachments.
- Transit Gateway Attachments—TGW supports two types of attachments *VPC* and *VPN*. VPN attachment support dynamic routing and ECMP. VPC attachments only support static routing without ECMP support. You can attach a VPC with both VPC and VPN attachment types concurrently. With the VPC attachment type, TGW deploys a network interface per availability zone within the VPC. VPN attachments are IPSec VPN tunnels to CGWs in the VPC.
- Transit Gateway Route Tables—TGW route tables behave like route domains. Segmentation of the network can be achieved by creating multiple route tables and associating VPCs and VPNs to them. In the TGW reference architecture, we group spoke VPCs to one route table and our security VPCs to another. Within a VPC we use local route tables to route to the TGW. Once in the TGW we use TGW route tables to route to a VPC or VPN attachment.

- Association—A TGW attachment is associated with one TGW route table.
- Route propagation—You can configure static routes within the TGW route table, or you can use the VPC or VPN connection to propagate routes into the TGW routing table. With a VPN connection routes are propagated with BGP and ECMP is supported. With VPC attachment you can't achieve ECMP today.
- Routing limitation—Static routes within a VPC or a TGW route table only allow a single route of the same value pointing to a single next hop. This means you can't configure two default routes 0.0.0.0/0 for example in the same route table to separate next hops. Even though the TGW route tables can support 10,000 routes there is still a BGP prefix limitation of 100.

Deployment Options

There are a two primary deployment options for providing a secure architecture for TGW.

In both options we have multiple spoke VPCs connected via a VPC attachment method, what differs is how we attach the firewalls..

Option 1 Multiple Security VPCs with VPC Attachment

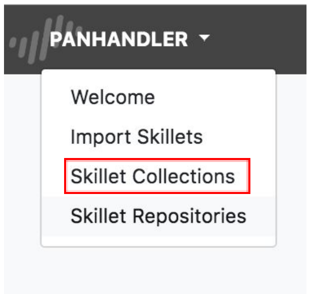
In this option our three security VPCs (FW-In, FW-Out and FW-E-W) are connected to the TGW with a VPC attachment. With the VPC only attachment method for the FW-Out and FW-E-W security VPCs, you are limited to static routing, no ECMP support and the need to reprogram the static routes during an outage to an alternative firewall network interface. This can be done manually or automated through use of AWS CloudWatch, AWS Lambda and an AWS CloudFormation template script for detection and failover of the firewalls, The advantage of VPC attachment is a simple design with no VPN tunnels but the disadvantage is the lack of ECMP and the ability to provide firewall failover with reprogramming the routes either manually or dynamically.

Option 2 Multiple Security VPCs with VPC-VPN Attachment

This approach is recommended over the first option due to the fact that we are able to support ECMP and dynamic failover using dynamic routing. In this option our FW-In VPC uses a VPC attachment type and the FW-Out and FW-E-W are connected via VPN and VPC attachment.

Deployment

Once the repository has been added/updated, you are now ready to deploy the demo framework. From the upper left hand corner Panhandler drop down list, select "Skillet Collections"



You will see the screen below. Select the AWS Transit Gateway Demo

This will take you to the template deployment screen. You will see four options.

The primary purpose of the skillet is to make it easy to create a demo environment that can be used to discuss how the firewalls are inserted into a transit gateway topology. A secondary purpose is to demonstrate how we provide high availability with direct attached firewalls.

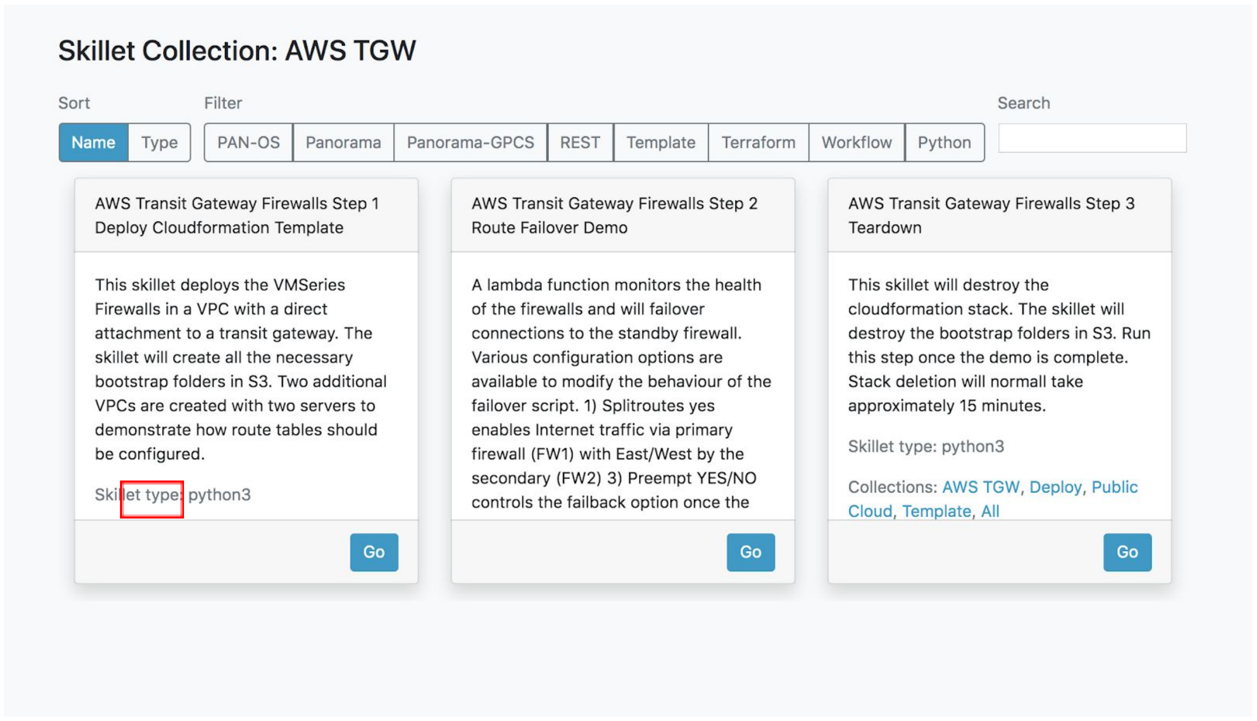
Step 1: This skillet deploys the VMSeries Firewalls in a VPC with a direct attachment to a transit gateway. The skillet will create all the necessary bootstrap folders in S3.Two additional VPCs are created with two servers to demonstrate how route tables should be configured

Step 2: The user is able to start and stop firewalls and modify the behaviour of the lambda function providing HA failover by modifying the lambda functions environment variables.

Step 3: Teardown- Once the demo is complete and you are done with the infrastructure, this step will destroy the demo environment.

__Launch Step 1: (Build of the Infrastructure)__

Click on the Go button on the step 1 deployment template.



Skillet Collection: AWS TGW

Sort: Name | Type

Filter: PAN-OS | Panorama | Panorama-GPCS | REST | Template | Terraform | Workflow | Python

Search

AWS Transit Gateway Firewalls Step 1 Deploy Cloudformation Template

This skillet deploys the VMSeries Firewalls in a VPC with a direct attachment to a transit gateway. The skillet will create all the necessary bootstrap folders in S3. Two additional VPCs are created with two servers to demonstrate how route tables should be configured.

Skillet type: python3

Go

AWS Transit Gateway Firewalls Step 2 Route Failover Demo

A lambda function monitors the health of the firewalls and will failover connections to the standby firewall. Various configuration options are available to modify the behaviour of the failover script. 1) Splitroutes yes enables Internet traffic via primary firewall (FW1) with East/West by the secondary (FW2) 3) Preempt YES/NO controls the failback option once the

Go

AWS Transit Gateway Firewalls Step 3 Teardown

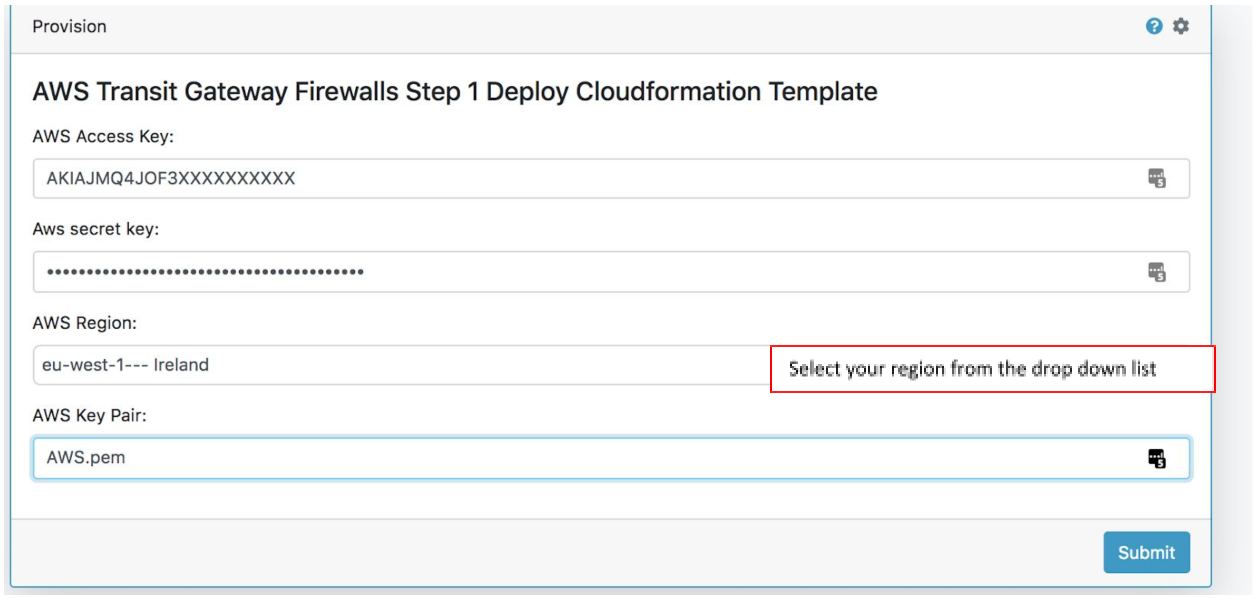
This skillet will destroy the cloudformation stack. The skillet will destroy the bootstrap folders in S3. Run this step once the demo is complete. Stack deletion will normall take approximately 15 minutes.

Skillet type: python3

Collections: AWS TGW, Deploy, Public Cloud, Template, All

Go

You should see the following screen:



Provision

AWS Transit Gateway Firewalls Step 1 Deploy Cloudformation Template

AWS Access Key: AKIAJMQ4JOF3XXXXXXXXXX

Aws secret key: ••

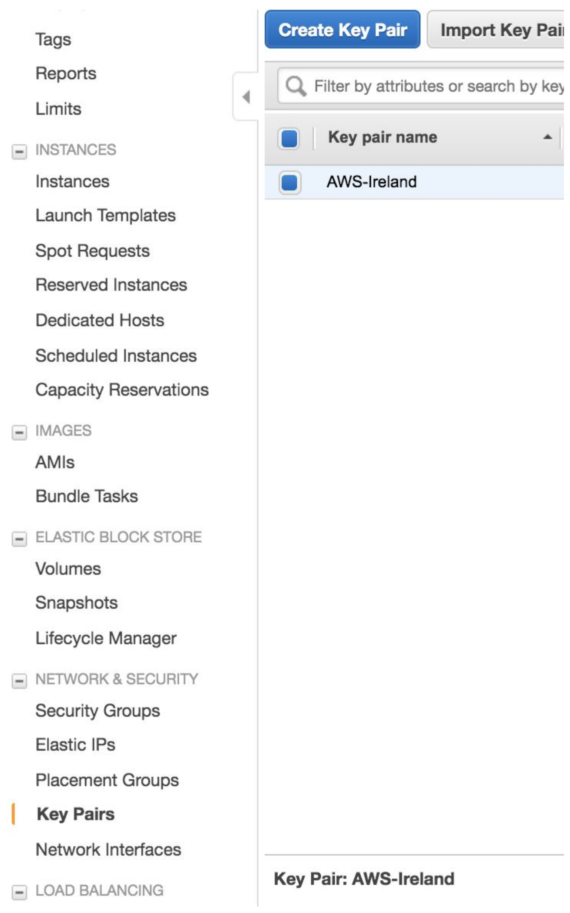
AWS Region: eu-west-1--- Ireland

Select your region from the drop down list

AWS Key Pair: AWS.pem

Submit

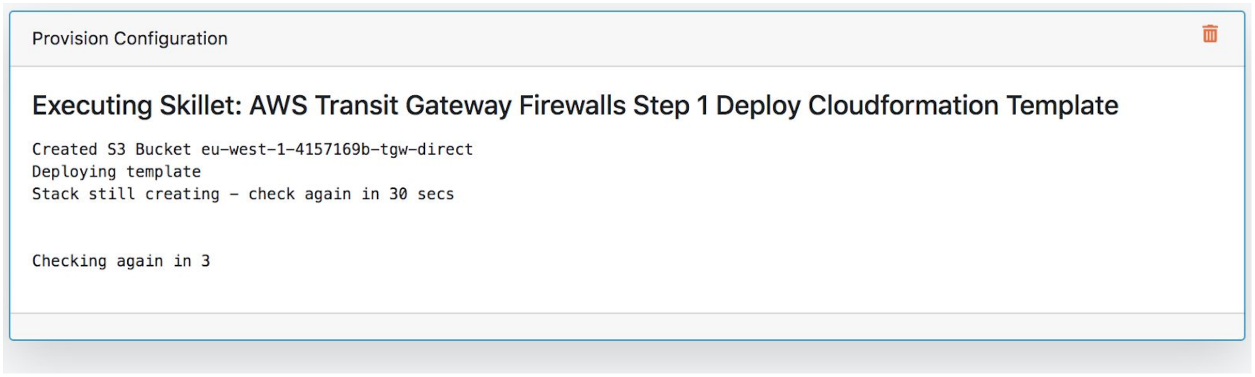
Important Note: If you do not have an SSH key pair already you must first generate the key pair and download it to your local machine



Once the above information is added, select the submit button to begin the installation. The first time the demo build is run, Panhandler will validate the correct version of libraries and tools are loaded into the environment. This will only run the first time you deploy the demo.:

Once the environment verification is completed, (blue continue button will indicate the process is complete) you will see the following:

Select the Continue button from the screen above and it will immediately start the build of the AWS Transit Gateway Demo, as shown below.



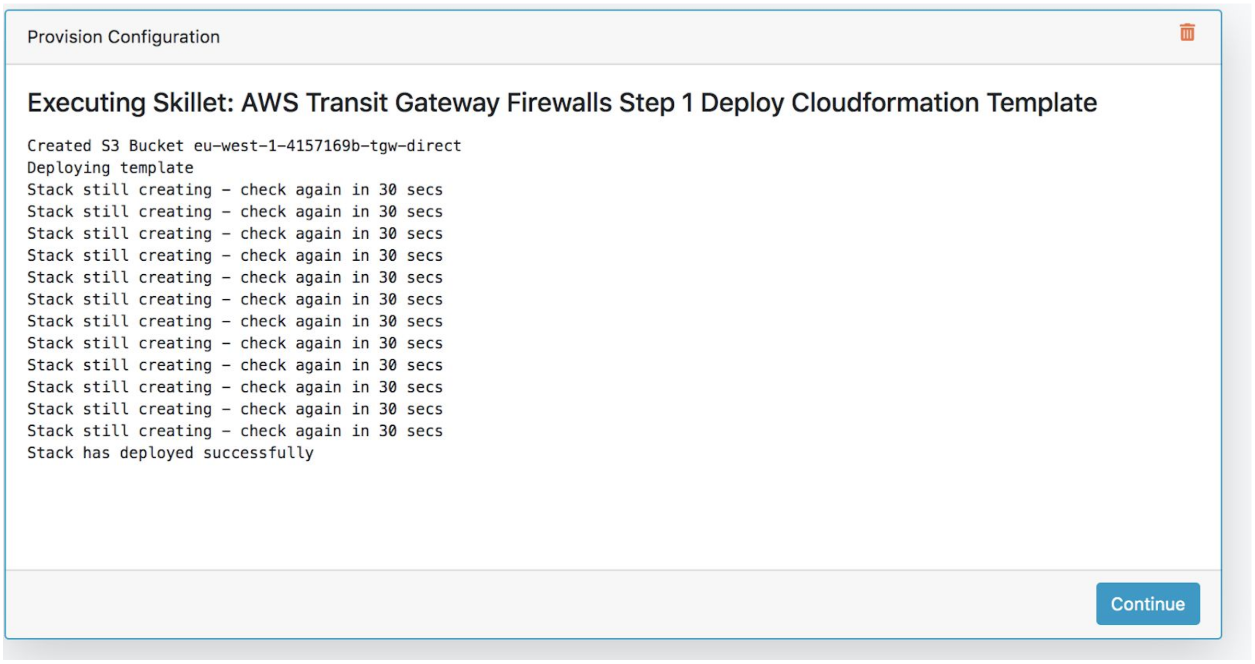
Executing Skillet: AWS Transit Gateway Firewalls Step 1 Deploy Cloudformation Template

```
Created S3 Bucket eu-west-1-4157169b-tgw-direct
Deploying template
Stack still creating - check again in 30 secs


Checking again in 3
```

The build of the architecture will take approximately 10 minutes.

Once the deployment completes you will need to wait an additional 5 minutes while the firewalls initialise and the route tables are updated.

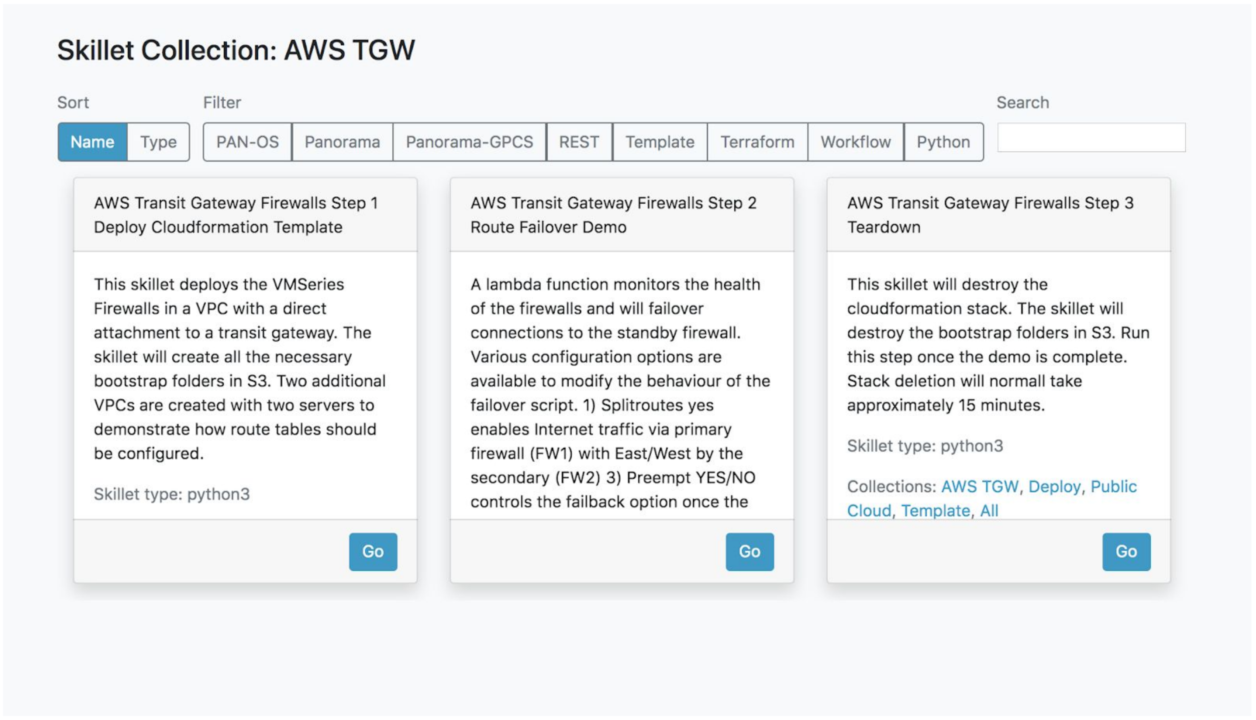
Once the Framework build is complete, you will see the following:



Executing Skillet: AWS Transit Gateway Firewalls Step 1 Deploy Cloudformation Template

```
Created S3 Bucket eu-west-1-4157169b-tgw-direct
Deploying template
Stack still creating - check again in 30 secs
Stack still creating - check again in 30 secs
Stack still creating - check again in 30 secs
Stack still creating - check again in 30 secs
Stack still creating - check again in 30 secs
Stack still creating - check again in 30 secs
Stack still creating - check again in 30 secs
Stack still creating - check again in 30 secs
Stack still creating - check again in 30 secs
Stack still creating - check again in 30 secs
Stack still creating - check again in 30 secs
Stack still creating - check again in 30 secs
Stack has deployed successfully
```

Select Continue from the screen above to take you back to the main AWS Transit Gateway Demo page.


Skillet Collection: AWS TGW

Sort

Filter

Search

AWS Transit Gateway Firewalls Step 1 Deploy Cloudformation Template

This skillet deploys the VMSeries Firewalls in a VPC with a direct attachment to a transit gateway. The skillet will create all the necessary bootstrap folders in S3. Two additional VPCs are created with two servers to demonstrate how route tables should be configured.

Skillet type: python3

AWS Transit Gateway Firewalls Step 2 Route Failover Demo

A lambda function monitors the health of the firewalls and will failover connections to the standby firewall. Various configuration options are available to modify the behaviour of the failover script. 1) Splitroutes yes enables Internet traffic via primary firewall (FW1) with East/West by the secondary (FW2) 3) Preempt YES/NO controls the failback option once the

AWS Transit Gateway Firewalls Step 3 Teardown

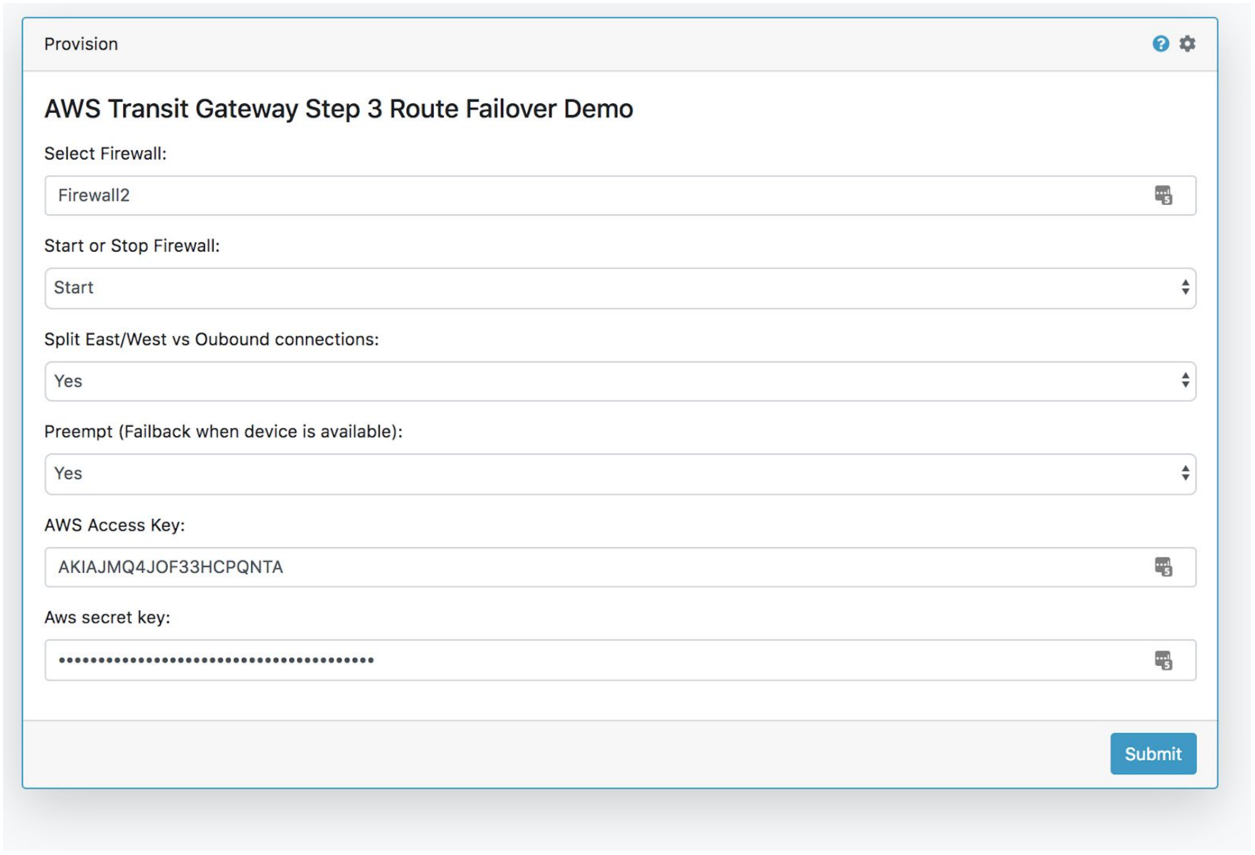
This skillet will destroy the cloudformation stack. The skillet will destroy the bootstrap folders in S3. Run this step once the demo is complete. Stack deletion will normall take approximately 15 minutes.

Skillet type: python3

Collections: AWS TGW, Deploy, Public Cloud, Template, All

Launch Step 2:

Click on the Go button on the step 3 deployment template. This will attempt to modify environment variable that modify the behaviour of the HA function and start /stop the firewalls. You should see the following screen:


Provision

AWS Transit Gateway Step 3 Route Failover Demo

Select Firewall: Firewall2

Start or Stop Firewall: Start

Split East/West vs Oubound connections: Yes

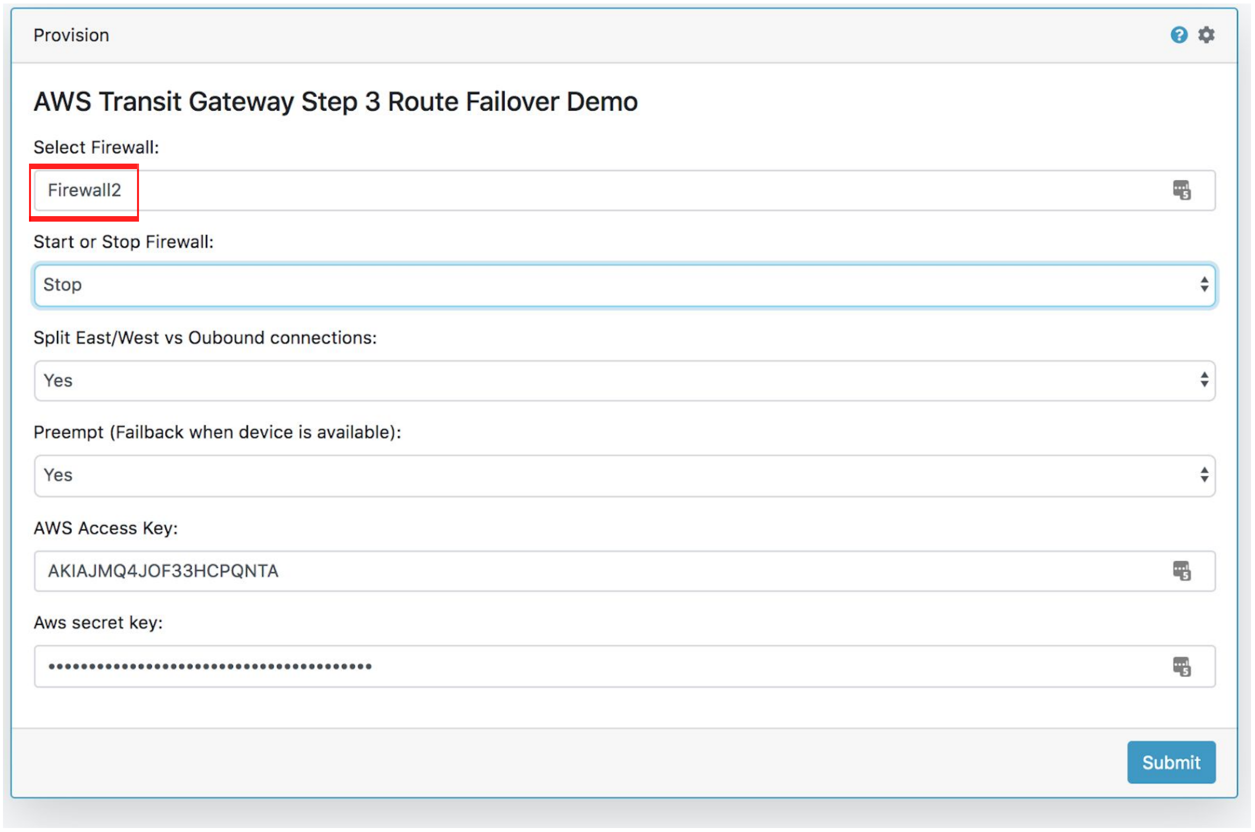
Preempt (Failback when device is available): Yes

AWS Access Key: AKIAJMQ4JOF33HCPQNTA

Aws secret key: ••

Submit

Select one of the firewalls to stop: In this case Firewall 2



Provision

AWS Transit Gateway Step 3 Route Failover Demo

Select Firewall:

Firewall2

Start or Stop Firewall:

Stop

Split East/West vs Oubound connections:

Yes

Preempt (Failback when device is available):

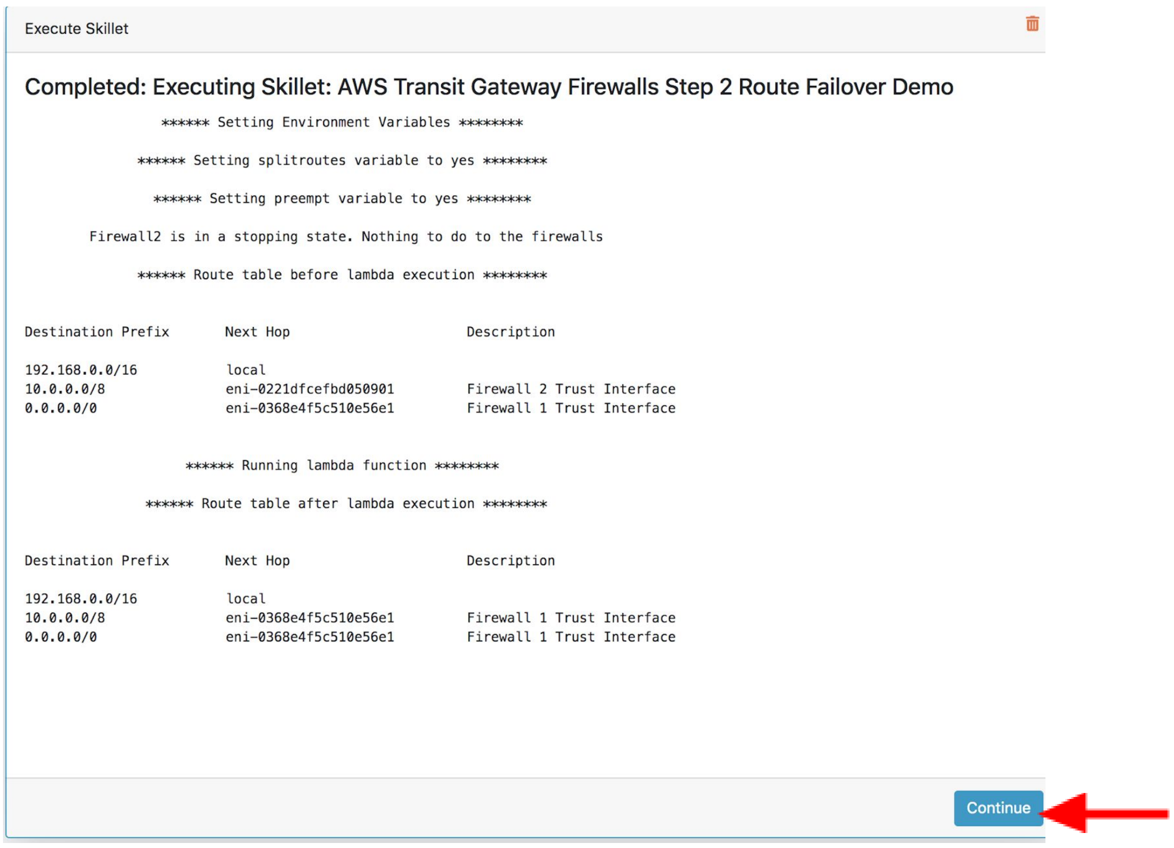
Yes

AWS Access Key:

AKIAJMQ4JOF33HCPQNTA

Aws secret key:

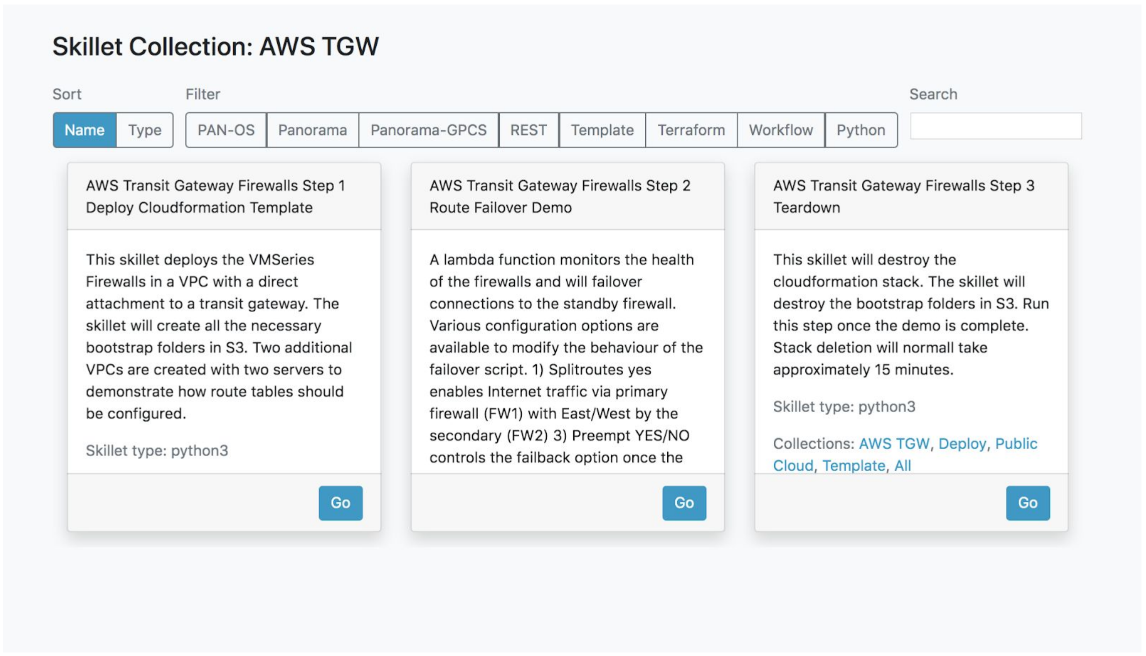
••



Completed: Executing Skillet: AWS Transit Gateway Firewalls Step 2 Route Failover Demo

```
                 ****** Setting Environment Variables ********

              ****** Setting splitroutes variable to yes ********

                ****** Setting preempt variable to yes ********

        Firewall2 is in a stopping state. Nothing to do to the firewalls

              ****** Route table before lambda execution ********


Destination Prefix       Next Hop                     Description

192.168.0.0/16           local
10.0.0.0/8               eni-0221dfcefbd050901        Firewall 2 Trust Interface
0.0.0.0/0                eni-0368e4f5c510e56e1        Firewall 1 Trust Interface


                    ****** Running lambda function ********

               ****** Route table after lambda execution ********


Destination Prefix       Next Hop                     Description

192.168.0.0/16           local
10.0.0.0/8               eni-0368e4f5c510e56e1        Firewall 1 Trust Interface
0.0.0.0/0                eni-0368e4f5c510e56e1        Firewall 1 Trust Interface
```

Select Continue and it will take you back to the main page.


Skillet Collection: AWS TGW

AWS Transit Gateway Firewalls Step 1 Deploy Cloudformation Template

This skillet deploys the VMSeries Firewalls in a VPC with a direct attachment to a transit gateway. The skillet will create all the necessary bootstrap folders in S3. Two additional VPCs are created with two servers to demonstrate how route tables should be configured.

Skillet type: python3

AWS Transit Gateway Firewalls Step 2 Route Failover Demo

A lambda function monitors the health of the firewalls and will failover connections to the standby firewall. Various configuration options are available to modify the behaviour of the failover script. 1) Splitroutes yes enables Internet traffic via primary firewall (FW1) with East/West by the secondary (FW2) 3) Preempt YES/NO controls the failback option once the

AWS Transit Gateway Firewalls Step 3 Teardown

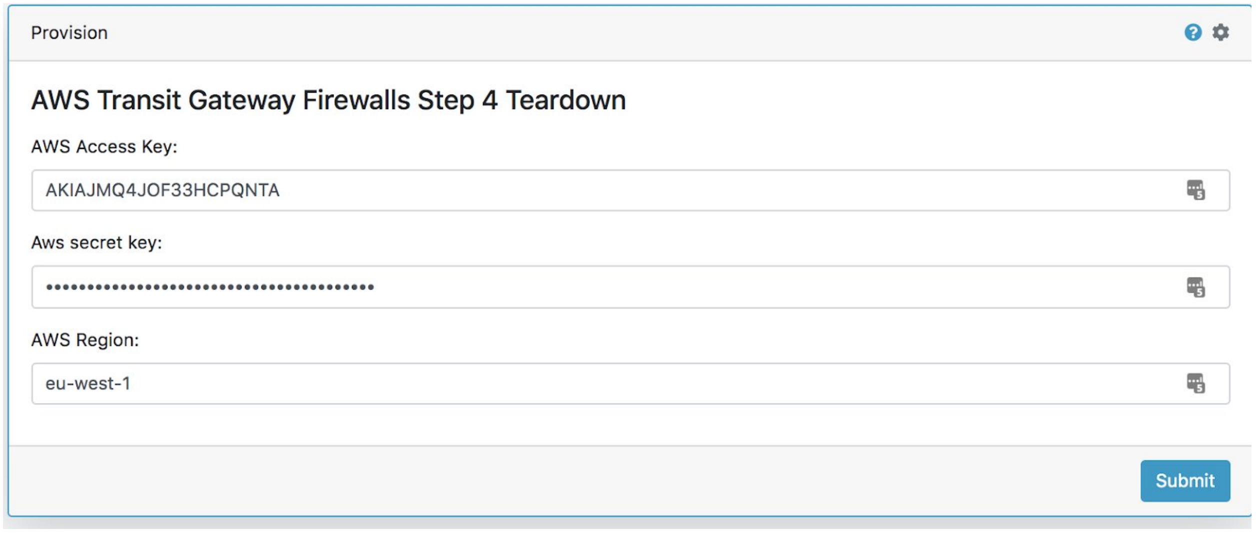
This skillet will destroy the cloudformation stack. The skillet will destroy the bootstrap folders in S3. Run this step once the demo is complete. Stack deletion will normall take approximately 15 minutes.

Skillet type: python3

Collections: AWS TGW, Deploy, Public Cloud, Template, All

Launch Step 3:

Click on the Go button on the step 3 deployment template. This will clean up the environment, destroying everything thus far created. You should see the following screen:



Provision

AWS Transit Gateway Firewalls Step 4 Teardown

AWS Access Key: AKIAJMQ4JOF33HCPQNTA

Aws secret key: ••

AWS Region: eu-west-1

Select Submit. Note that this process will take several minutes to complete. The process is complete when you get the “Continue” button at the bottom of the page.

Provision Configuration

Executing Skillet: AWS Transit Gateway Firewalls Step 4 Teardown

```
Deleting Stack panw-d07979d3tgw-direct
Stack still deleting
Stack still deleting
Stack still deleting
Stack still deleting
Stack still deleting
Stack still deleting
Stack still deleting
Stack still deleting
Stack still deleting
Stack still deleting
Stack still deleting
Stack still deleting
Stack still deleting
Stack still deleting
Stack still deleting
Stack still deleting
Stack has been deleted
Deleted Stack panw-d07979d3tgw-direct
Deleting S3 Bucket eu-west-1-d07979d3-tgw-direct
```