Deploying VM Series in a Transit Gateway with Panorama

http://www.paloaltonetworks.com

Table of Contents

Version History

Version number	Comments
1.0	Initial GitHub check-in

1. About

This document will explain how to deploy an AWS Transit Gateway with the VM-Series Firewall on AWS, automate the connection to Panorama, and automatically obtain a BYOL license with an auth code. A Transit Gateway 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.

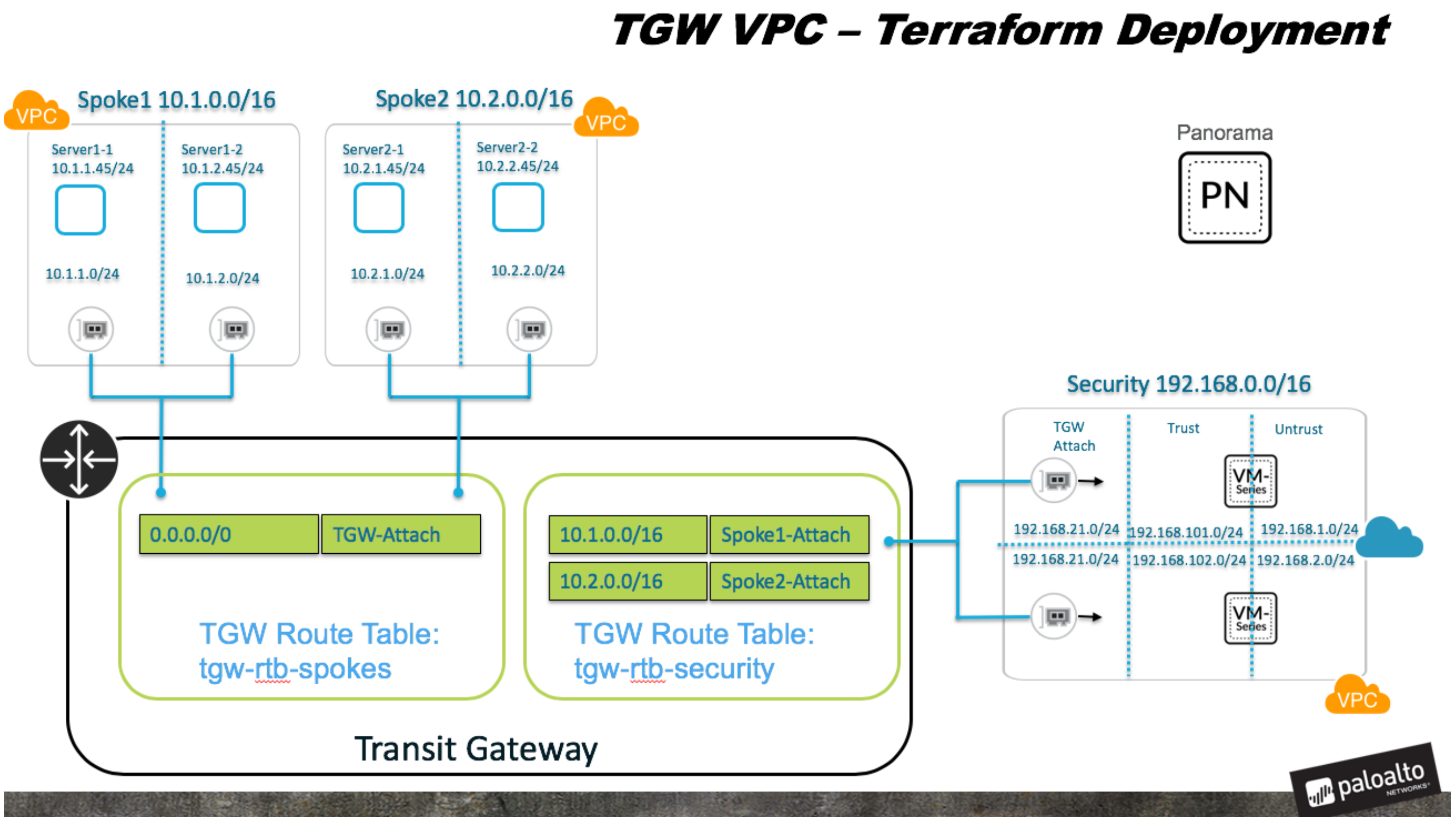
AWS Transit Gateway is a service that allows customers to connect their Amazon Virtual Private Clouds (VPCs) and their on-premises networks to a single gateway. As you grow the number of workloads running on AWS, you need to be able to scale your networks across multiple accounts and Amazon VPCs to keep 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 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:



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 onto your local machine: https://github.com/djspears/TGW-Panorama-BYOL
3. Deployed Panorama. This does not need to be in AWS but must be accessible to the VM firewalls after instantiation. https://docs.paloaltonetworks.com/panorama/8-1/panorama-admin/set-up-panorama/set-up-the-panorama-virtual-appliance/install-the-panorama-virtual-appliance/install-panorama-in-aws
4. Generate a Panorama VM-Auth Key: https://docs.paloaltonetworks.com/vm-series/8-1/vm-series-deployment/bootstrap-the-vm-series-firewall/generate-the-vm-auth-key-on-panorama
5. AWS credentials are setup for terraform to use. Here are to guides that may assist with this:

   https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-credentials.html

   https://www.terraform.io/docs/providers/aws/index.html
6. If BYOL licensing is being used then the authcode must be registered with the support portal. The following link provides instructions on how to do this: https://docs.paloaltonetworks.com/vm-series/8-1/vm-series-deployment/license-the-vm-series-firewall/register-the-vm-series-firewall/register-the-vm-series-firewall-with-auth-code.html

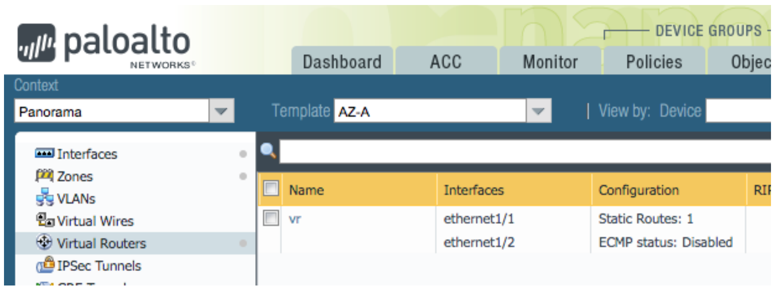
5. Panorama Connection Methods

There are a couple of ways to configure bootstrapping and ultimately connect to Panorama. The firewall can be bootstrapped and obtain its configuration from Panorama by defining the Device Group and/or Template in the init-cfg.txt file used during bootstrapping. This is probably the safest approach which does not require ongoing maintenance of the boostrap.xml file.

It is also possible to have the firewall bootstrap using the boostrap.xml file and not connect automatically to a Device Group/Template. The device group and/or template could be assigned later. This would require manual intervention to address any conflicts.

There are a few other things to keep in mind when setting up the automated connection to Panorama:

- If the firewall is going to attach to a Device Group and/or a Template during the bootstrap process and a bootstrap.xml files is also being used, any conflicts between the boostrap.xml and the Device Group/Template configuration will cause the automatic connection and configuration to be unsuccessful.
- IMPORTANT: If a firewall is going to automatically connect to a template, the Virtual Router definition in Panorama has to be a name different than default. If this not done, then the bootstrapped firewall will not successfully connect to the template. Here is a screen shot of a modified Virtual Router in Panorama:
-



Name	Interfaces	Configuration	RI
vr	ethernet1/1 ethernet1/2	Static Routes: 1 ECMP status: Disabled	

6. Panorama Configuration

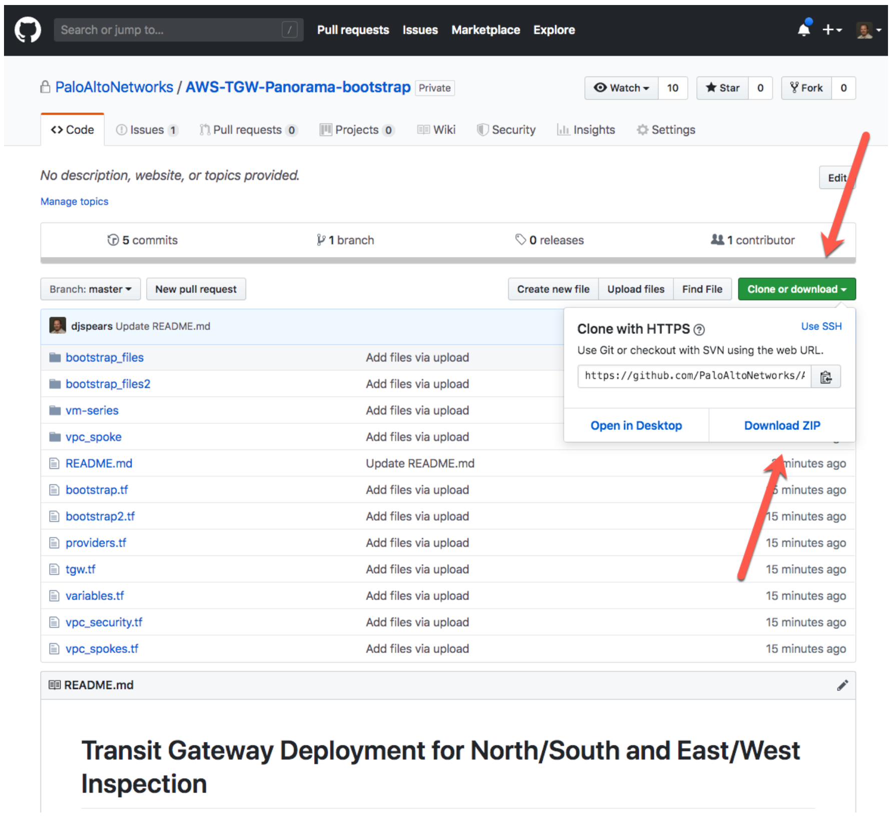
This document is not going to cover the deployment and initial configuration of Panorama. This guide assumes that has been done and that the panorama instance is accessible by the VM-Series during the bootstrap process. Here is a deployment guide that provides guidance on this:
https://www.paloaltonetworks.com/resources/guides/panorama-on-aws-deployment-guide

If there desire is to import a firewall configuration into Panorama so it can be used for other deployments the process to perform that import can be found here:
https://knowledgebase.paloaltonetworks.com/KCSArticleDetail?id=kA10g000000ClZSCA0

NOTE: This is the process that was used to import the configurations into Panorama that is being used in this guide. The initial bootstrap process used the bootstrap.xml files from the Github repo and connected to panorama and then the firewall configurations where imported into the Device Group/ Template Stack that were subsequently reused for new deployments.

7. Download the files from Github

In the Github repo click on the Clone or Download button to download the contents of the repository. There are multiple ways to get the files downloaded this is just one.



No description, website, or topics provided.

Clone with HTTPS
Use Git or checkout with SVN using the web URL.
https://github.com/PaloAltoNetworks/

djspears Update README.md

bootstrap_files	Add files via upload	
bootstrap_files2	Add files via upload	
vm-series	Add files via upload	
vpc_spoke	Add files via upload	
README.md	Update README.md	minutes ago
bootstrap.tf	Add files via upload	5 minutes ago
bootstrap2.tf	Add files via upload	15 minutes ago
providers.tf	Add files via upload	15 minutes ago
tgw.tf	Add files via upload	15 minutes ago
variables.tf	Add files via upload	15 minutes ago
vpc_security.tf	Add files via upload	15 minutes ago
vpc_spokes.tf	Add files via upload	15 minutes ago

README.md

Transit Gateway Deployment for North/South and East/West Inspection

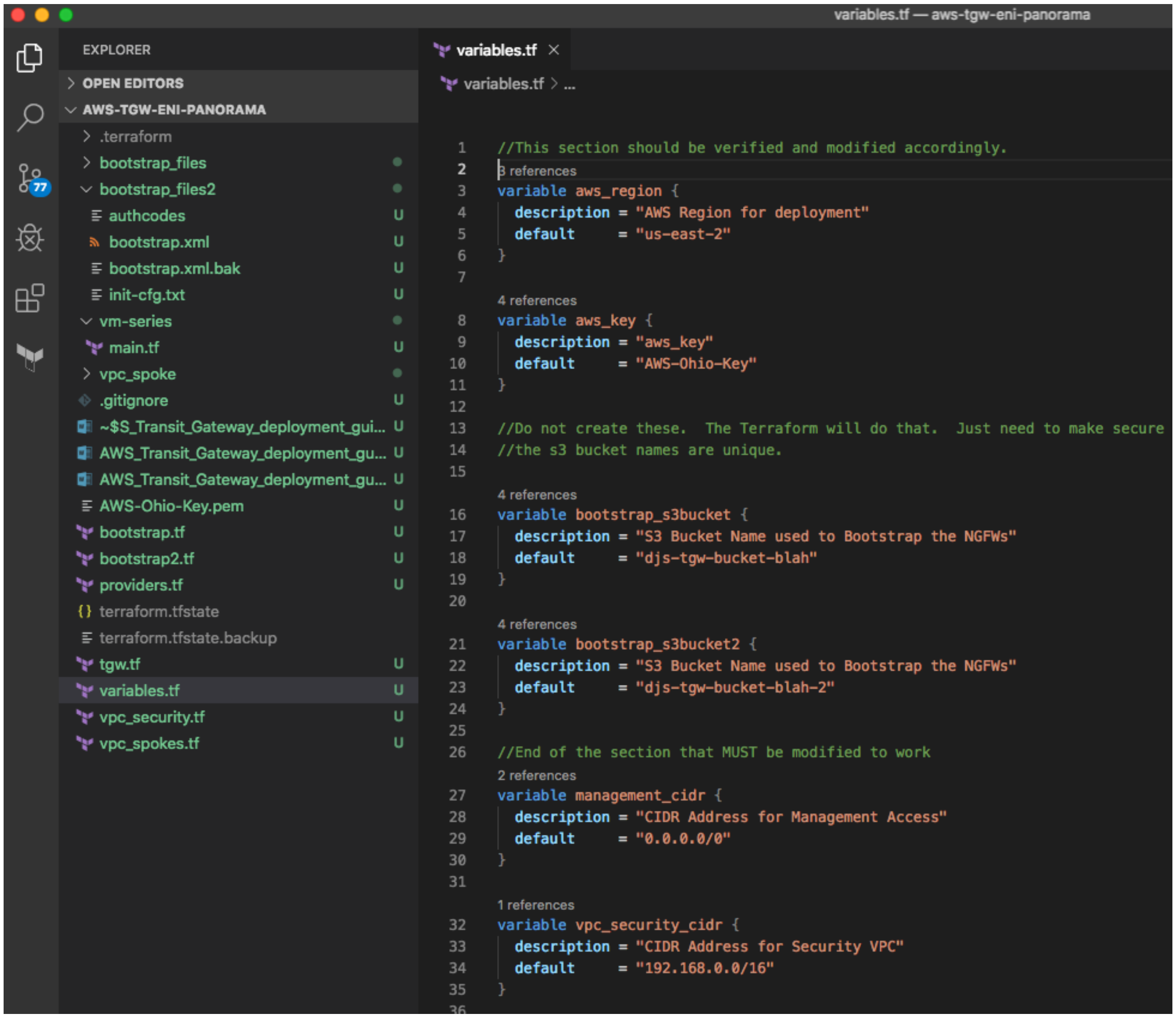
Save the files to a known directory and unzip.

9. Modify the deployment files

Leverage a file editor to open the following files and modify as needed.

Variables.tf file

The variables.tf file has many options that can be customized for this deployment. There are a couple that need to be changed. I have put these at the top of the file:

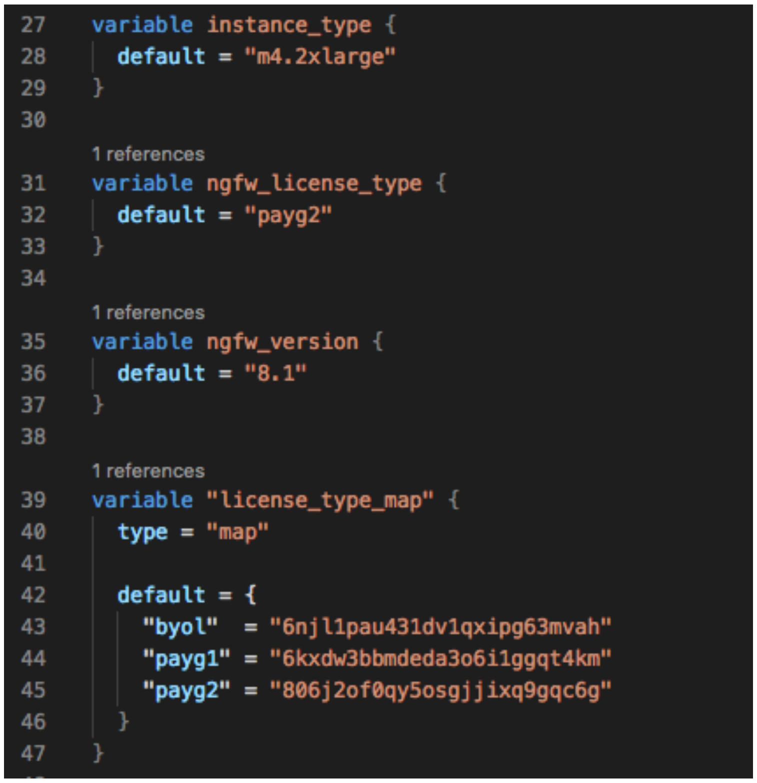


```
//This section should be verified and modified accordingly.
3 references
variable aws_region {
  description = "AWS Region for deployment"
  default     = "us-east-2"
}

4 references
variable aws_key {
  description = "aws_key"
  default     = "AWS-Ohio-Key"
}

//Do not create these.  The Terraform will do that.  Just need to make secure
//the s3 bucket names are unique.

4 references
variable bootstrap_s3bucket {
  description = "S3 Bucket Name used to Bootstrap the NGFWs"
  default     = "djs-tgw-bucket-blah"
}

4 references
variable bootstrap_s3bucket2 {
  description = "S3 Bucket Name used to Bootstrap the NGFWs"
  default     = "djs-tgw-bucket-blah-2"
}

//End of the section that MUST be modified to work
2 references
variable management_cidr {
  description = "CIDR Address for Management Access"
  default     = "0.0.0.0/0"
}

1 references
variable vpc_security_cidr {
  description = "CIDR Address for Security VPC"
  default     = "192.168.0.0/16"
}
```

Modify the aws_region, aws_key, bootstrap_s3bucket, and bootstrap_s3bucket2. The bootstrap s3 bucket will get created via the terraform deployment and files inserted. The S3 bucket names must be unique to AWS so these must get changed.

VM Series Firewall deployment file

There is a main.tf file located in the vm-series directory. This file can be modified to deploy a different type/version of the vm series firewall. Starting at line 27 the file looks like this:

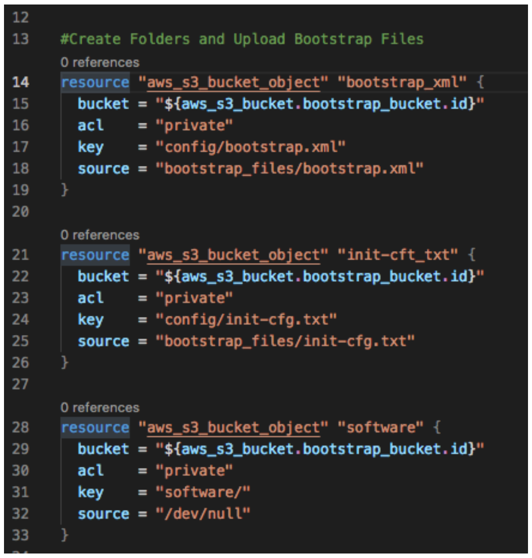


```
27   variable instance_type {
28     default = "m4.2xlarge"
29   }
30
     1 references
31   variable ngfw_license_type {
32     default = "payg2"
33   }
34
     1 references
35   variable ngfw_version {
36     default = "8.1"
37   }
38
     1 references
39   variable "license_type_map" {
40     type = "map"
41
42     default = {
43       "byol"  = "6njl1pau431dv1qxipg63mvah"
44       "payg1" = "6kxdw3bbmdeda3o6i1ggqt4km"
45       "payg2" = "806j2of0qy5osgjjixq9gqc6g"
46     }
47   }
```

The default configuration deploys a PayGo2 license running version 8.1. If BYOL is going to be used then modify the license type to byol.

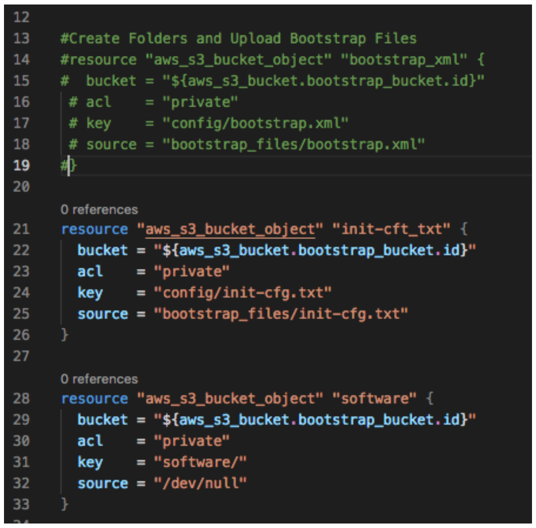
Bootstrap files

This template deploys 2 firewalls and also creates 2 S3 bucket each with the files needed to bootstrap the firewall. There is a bootstrap.tf and bootstrap2.tf that create the S3 buckets. By default a bootstrap.xml file is put into the S3 buckets. If the desire is to have the firewall get it's initial configuration from Panorama then these files need to be modified. The portion of this file that copies the bootstrap.xml to the S3 bucket starts at line 14 and looks like the following:



```
12
13   #Create Folders and Upload Bootstrap Files
     0 references
14   resource "aws_s3_bucket_object" "bootstrap_xml" {
15     bucket = "${aws_s3_bucket.bootstrap_bucket.id}"
16     acl    = "private"
17     key    = "config/bootstrap.xml"
18     source = "bootstrap_files/bootstrap.xml"
19   }
20
     0 references
21   resource "aws_s3_bucket_object" "init-cft_txt" {
22     bucket = "${aws_s3_bucket.bootstrap_bucket.id}"
23     acl    = "private"
24     key    = "config/init-cfg.txt"
25     source = "bootstrap_files/init-cfg.txt"
26   }
27
     0 references
28   resource "aws_s3_bucket_object" "software" {
29     bucket = "${aws_s3_bucket.bootstrap_bucket.id}"
30     acl    = "private"
31     key    = "software/"
32     source = "/dev/null"
33   }
```

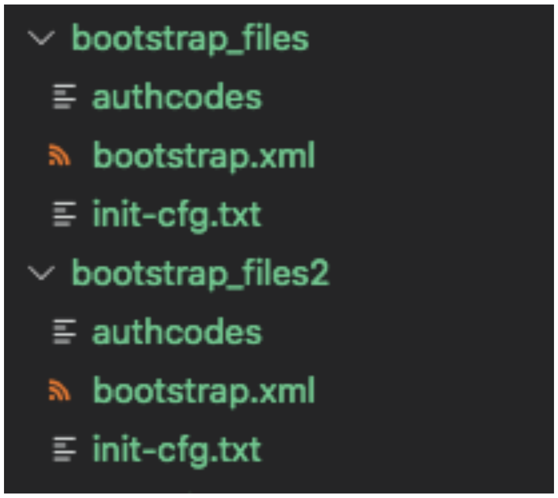
If there is a desire to bootstrap without the boostrap.xml file modify the bootstrap.tf and bootstrap2.tf comment out the section that copies the bootstrap.xml file to the s3 bucket. The files should look like the following:



```
12
13   #Create Folders and Upload Bootstrap Files
14   #resource "aws_s3_bucket_object" "bootstrap_xml" {
15   #  bucket = "${aws_s3_bucket.bootstrap_bucket.id}"
16    # acl    = "private"
17    # key    = "config/bootstrap.xml"
18    # source = "bootstrap_files/bootstrap.xml"
19   #}
20
     0 references
21   resource "aws_s3_bucket_object" "init-cft_txt" {
22     bucket = "${aws_s3_bucket.bootstrap_bucket.id}"
23     acl    = "private"
24     key    = "config/init-cfg.txt"
25     source = "bootstrap_files/init-cfg.txt"
26   }
27
     0 references
28   resource "aws_s3_bucket_object" "software" {
29     bucket = "${aws_s3_bucket.bootstrap_bucket.id}"
30     acl    = "private"
31     key    = "software/"
32     source = "/dev/null"
33   }
```

NOTE: If this is being deployed from a Windows workstation, there was a Terraform bug that would produce an error when trying to create a s3 folder without any files in it.

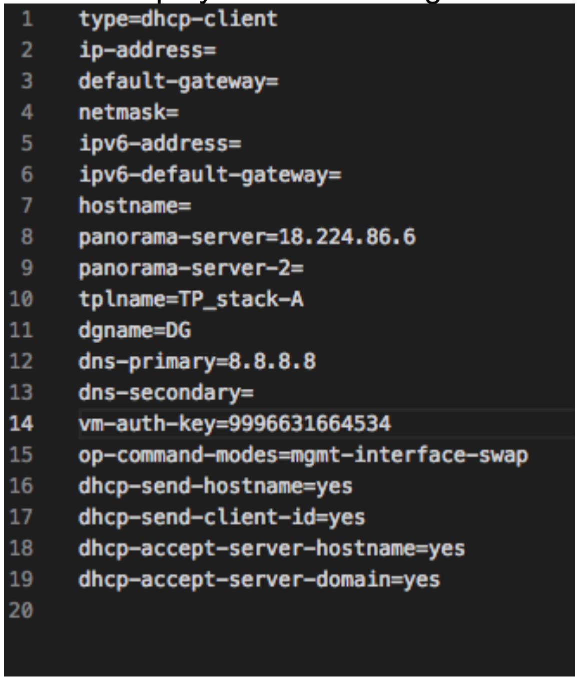
The files used to bootstrap the vm-series firewall are located in the bootstrap_files and bootstrap_files2 directory. This are the files that are located in them:



```
bootstrap_files
    authcodes
    bootstrap.xml
    init-cfg.txt
bootstrap_files2
    authcodes
    bootstrap.xml
    init-cfg.txt
```

The init-cfg.txt will have some items that will need to be modified. More information about the items that are supported in this file are located here: https://docs.paloaltonetworks.com/vm-series/8-1/vm-series-deployment/bootstrap-the-vm-series-firewall/create-the-init-cfgtxt-file/sample-init-cfgtxt-file.html

For this deployment the init-cfg.txt file looks like this:



```
type=dhcp-client
ip-address=
default-gateway=
netmask=
ipv6-address=
ipv6-default-gateway=
hostname=
panorama-server=18.224.86.6
panorama-server-2=
tplname=TP_stack-A
dgname=DG
dns-primary=8.8.8.8
dns-secondary=
vm-auth-key=9996631664534
op-command-modes=mgmt-interface-swap
dhcp-send-hostname=yes
dhcp-send-client-id=yes
dhcp-accept-server-hostname=yes
dhcp-accept-server-domain=yes
```

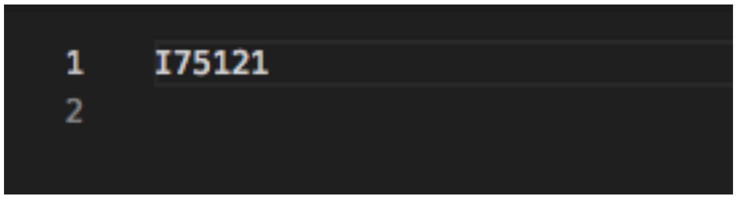
The items that must be updated/modified are:

Panorama-server: put in the correct IP of the Panorama server used in this deployment.
vm-auth-key: This is a auth key created on Panorama as part of the prerequisite work on Page 6 of this guide. https://docs.paloaltonetworks.com/vm-series/8-1/vm-series-deployment/bootstrap-the-vm-series-firewall/generate-the-vm-auth-key-on-panorama

tplname: This is the Template Stack configured within Panorama.
Dgname: This is the Device Group name create with in Panorama.
Note: for the deployment in this lab, there was a single device group that had the security policies and then a unique template for each firewall. This is because each firewall is in a different AZ and needed unique routes.

Be sure to modify both init-cfg.txt files!

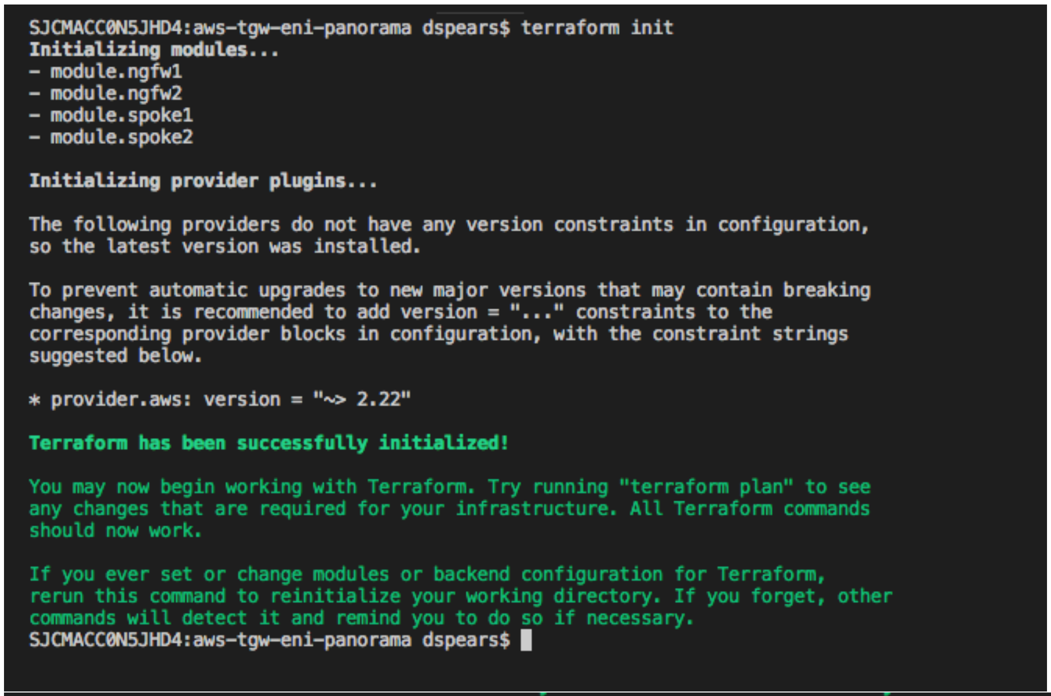
Finally, if BYOL is being used then an authcode must be provided to automate the licensing process during the bootstrap process. This is located in the authcodes file. The updated authcodes file should look like this:



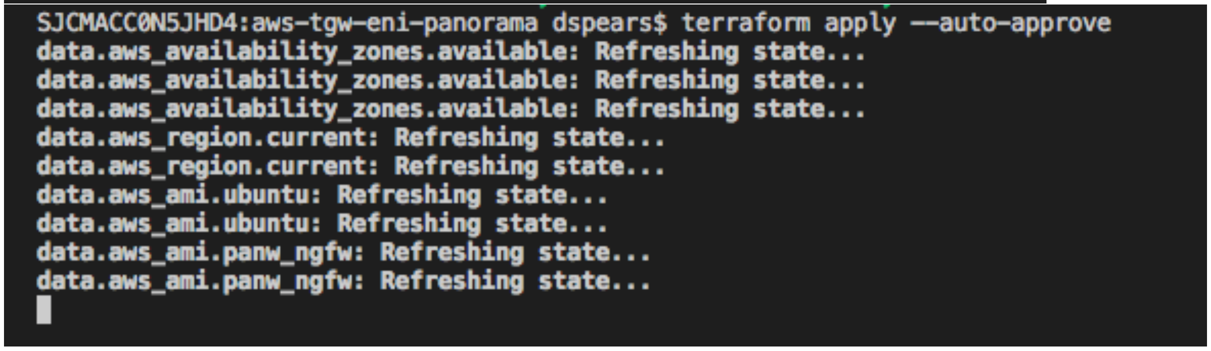
```
I75121
```

10. Deploying the template

To deploy the template open a command prompt and cd to the directory with the updated files and perform an **Terraform init** and **Terraform apply - - auto-approve**:

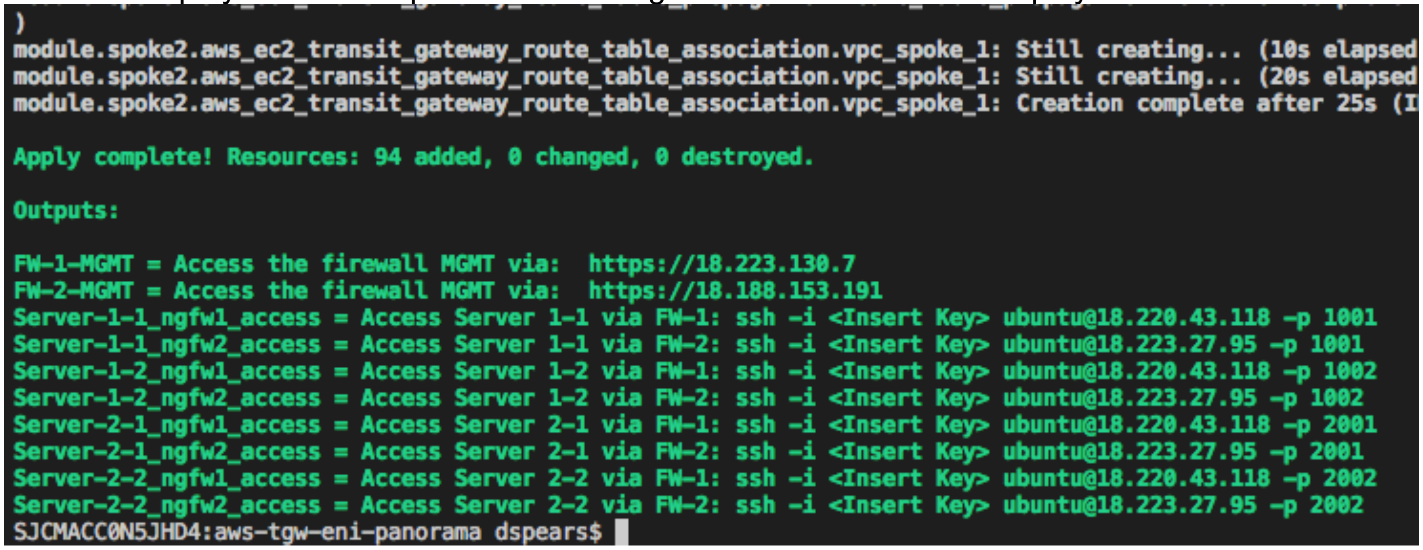


```
SJCMACC0N5JHD4:aws-tgw-eni-panorama dspears$ terraform init
Initializing modules...
- module.ngfw1
- module.ngfw2
- module.spoke1
- module.spoke2

Initializing provider plugins...

The following providers do not have any version constraints in configuration,
so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.

* provider.aws: version = "~> 2.22"

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
SJCMACC0N5JHD4:aws-tgw-eni-panorama dspears$
```



```
SJCMACC0N5JHD4:aws-tgw-eni-panorama dspears$ terraform apply --auto-approve
data.aws_availability_zones.available: Refreshing state...
data.aws_availability_zones.available: Refreshing state...
data.aws_availability_zones.available: Refreshing state...
data.aws_region.current: Refreshing state...
data.aws_region.current: Refreshing state...
data.aws_ami.ubuntu: Refreshing state...
data.aws_ami.ubuntu: Refreshing state...
data.aws_ami.panw_ngfw: Refreshing state...
data.aws_ami.panw_ngfw: Refreshing state...
```

Once the deployment is complete the following information should be displayed:

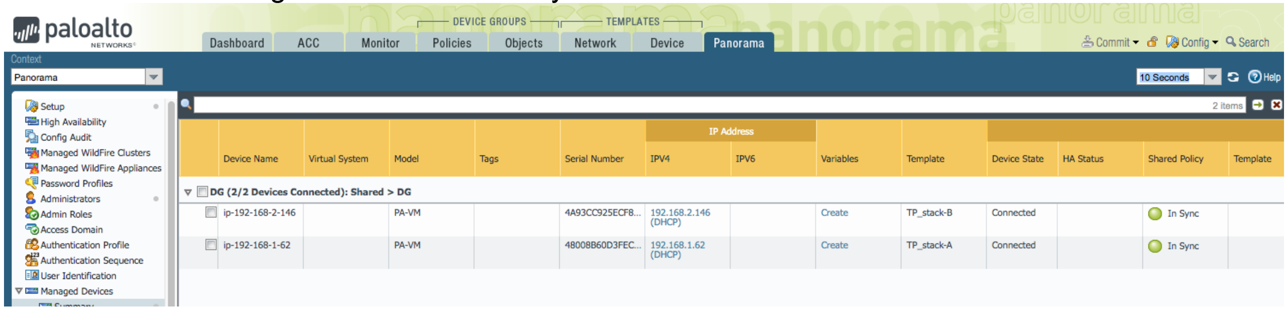


```
)
module.spoke2.aws_ec2_transit_gateway_route_table_association.vpc_spoke_1: Still creating... (10s elapsed
module.spoke2.aws_ec2_transit_gateway_route_table_association.vpc_spoke_1: Still creating... (20s elapsed
module.spoke2.aws_ec2_transit_gateway_route_table_association.vpc_spoke_1: Creation complete after 25s (I

Apply complete! Resources: 94 added, 0 changed, 0 destroyed.

Outputs:

FW-1-MGMT = Access the firewall MGMT via:  https://18.223.130.7
FW-2-MGMT = Access the firewall MGMT via:  https://18.188.153.191
Server-1-1_ngfw1_access = Access Server 1-1 via FW-1: ssh -i <Insert Key> ubuntu@18.220.43.118 -p 1001
Server-1-1_ngfw2_access = Access Server 1-1 via FW-2: ssh -i <Insert Key> ubuntu@18.223.27.95 -p 1001
Server-1-2_ngfw1_access = Access Server 1-2 via FW-1: ssh -i <Insert Key> ubuntu@18.220.43.118 -p 1002
Server-1-2_ngfw2_access = Access Server 1-2 via FW-2: ssh -i <Insert Key> ubuntu@18.223.27.95 -p 1002
Server-2-1_ngfw1_access = Access Server 2-1 via FW-1: ssh -i <Insert Key> ubuntu@18.220.43.118 -p 2001
Server-2-1_ngfw2_access = Access Server 2-1 via FW-2: ssh -i <Insert Key> ubuntu@18.223.27.95 -p 2001
Server-2-2_ngfw1_access = Access Server 2-2 via FW-1: ssh -i <Insert Key> ubuntu@18.220.43.118 -p 2002
Server-2-2_ngfw2_access = Access Server 2-2 via FW-2: ssh -i <Insert Key> ubuntu@18.223.27.95 -p 2002
SJCMACC0N5JHD4:aws-tgw-eni-panorama dspears$
```

The information here will provide info on how to connect to the various systems that were deployed.

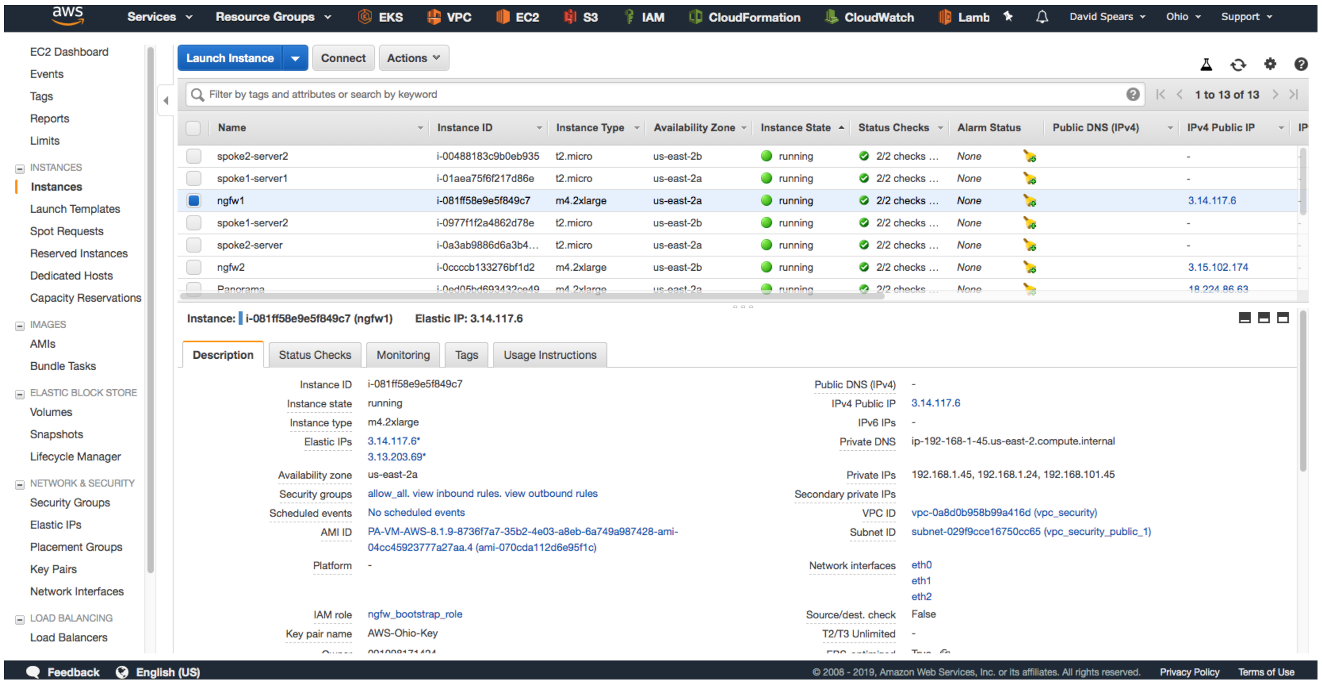
11. Verifying the deployment

The full deployment will take a while especially if BYOL is being used as there are a few reboots needed. Looking at Panorama will show when the firewalls are up and have been updated. Looking at the Panorama tab/Managed Devices/Summary should look like this:



	Device Name	Virtual System	Model	Tags	Serial Number	IP Address IPV4	IP Address IPV6	Variables	Template	Device State	HA Status	Shared Policy	Template
DG (2/2 Devices Connected): Shared > DG													
☐	ip-192-168-2-146		PA-VM		4A93CC925ECF8...	192.168.2.146 (DHCP)		Create	TP_stack-B	Connected		● In Sync	
☐	ip-192-168-1-62		PA-VM		48008B60D3FEC...	192.168.1.62 (DHCP)		Create	TP_stack-A	Connected		● In Sync	

Navigate to the AWS console. Looking at the EC2 running instances will provide a list of running instances:

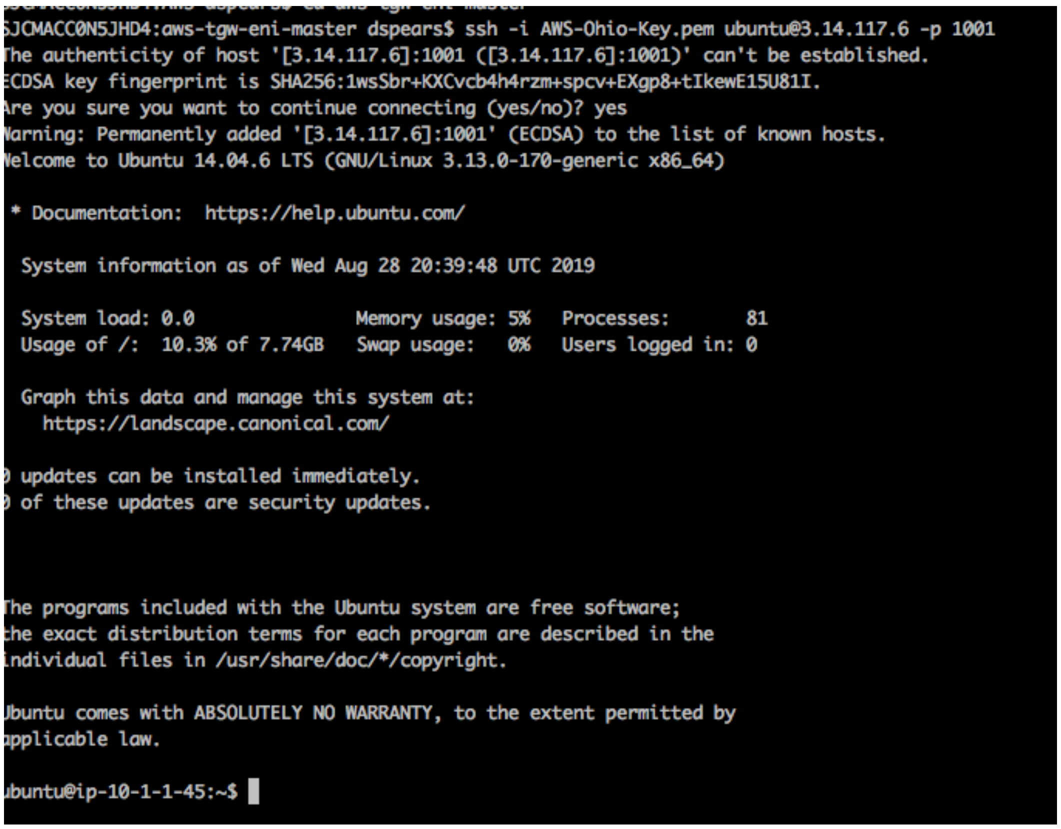


Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP
spoke2-server2	i-00488183c9b0eb935	t2.micro	us-east-2b	running	2/2 checks ...	None		-
spoke1-server1	i-01aea75f6f217d86e	t2.micro	us-east-2a	running	2/2 checks ...	None		-
ngfw1	i-081ff58e9e5f849c7	m4.2xlarge	us-east-2a	running	2/2 checks ...	None		3.14.117.6
spoke1-server2	i-0977f1f2a4862d78e	t2.micro	us-east-2b	running	2/2 checks ...	None		-
spoke2-server	i-0a3ab9886d6a3b4...	t2.micro	us-east-2a	running	2/2 checks ...	None		-
ngfw2	i-0ccccb133276bf1d2	m4.2xlarge	us-east-2b	running	2/2 checks ...	None		3.15.102.174
Panorama	i-0ed05bd693432ce49	m4.2xlarge	us-east-2a	running	2/2 checks ...	None		18.224.86.63

Instance: i-081ff58e9e5f849c7 (ngfw1) Elastic IP: 3.14.117.6

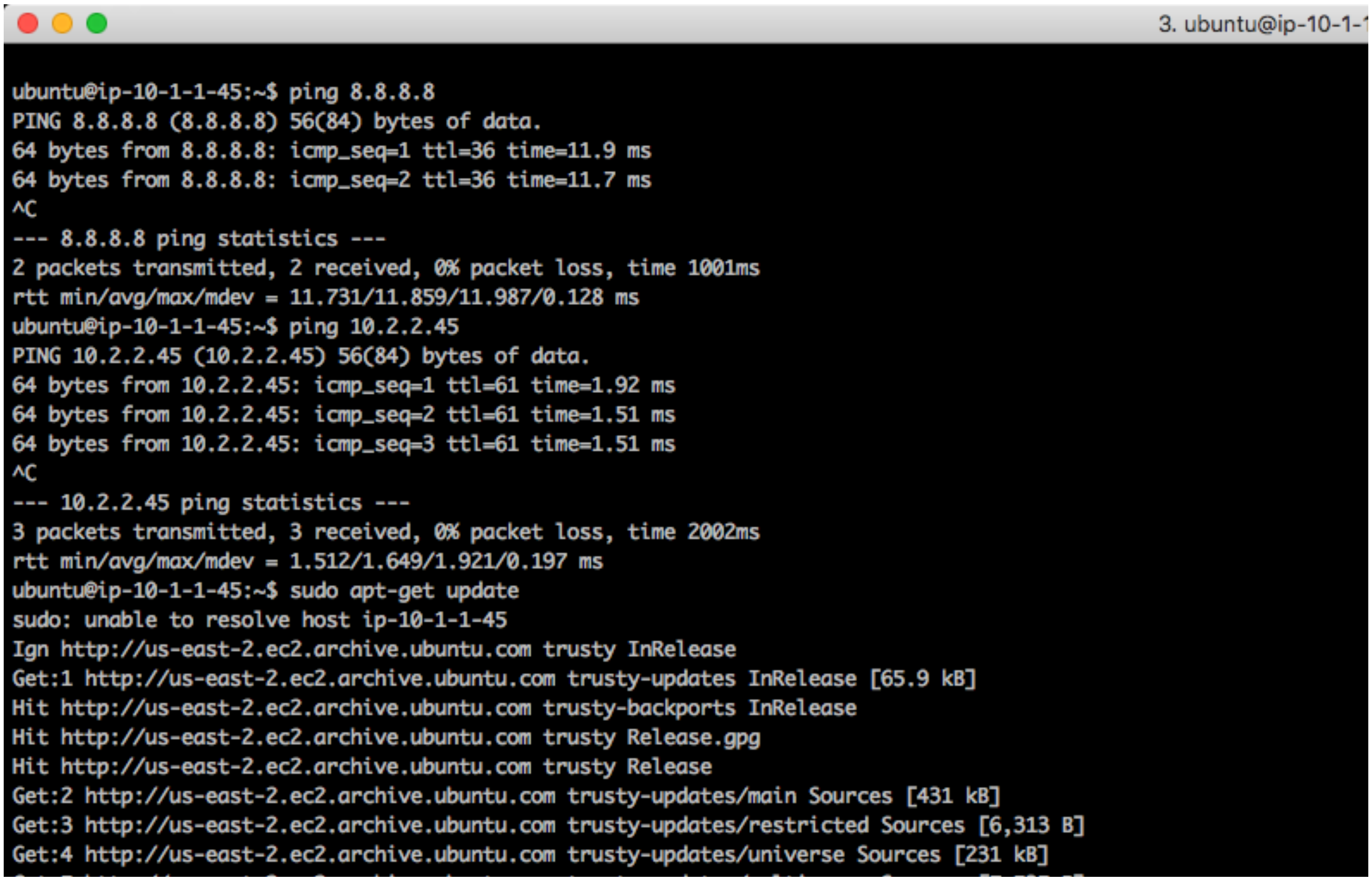
Instance ID	i-081ff58e9e5f849c7	Public DNS (IPv4)	-
Instance state	running	IPv4 Public IP	3.14.117.6
Instance type	m4.2xlarge	IPv6 IPs	-
Elastic IPs	3.14.117.6* 3.13.203.69*	Private DNS	ip-192-168-1-45.us-east-2.compute.internal
Availability zone	us-east-2a	Private IPs	192.168.1.45, 192.168.1.24, 192.168.101.45
Security groups	allow_all. view inbound rules. view outbound rules	Secondary private IPs	
Scheduled events	No scheduled events	VPC ID	vpc-0a8d0b958b99a416d (vpc_security)
AMI ID	PA-VM-AWS-8.1.9-8736f7a7-35b2-4e03-a8eb-6a749a987428-ami-04cc45923777a27aa.4 (ami-070cda112d6e95f1c)	Subnet ID	subnet-029f9cce16750cc65 (vpc_security_public_1)
Platform	-	Network interfaces	eth0 eth1 eth2
IAM role	ngfw_bootstrap_role	Source/dest. check	False
Key pair name	AWS-Ohio-Key	T2/T3 Unlimited	-

Using the information provide in the Terraform output you should be able to log into one of the VM remotely:



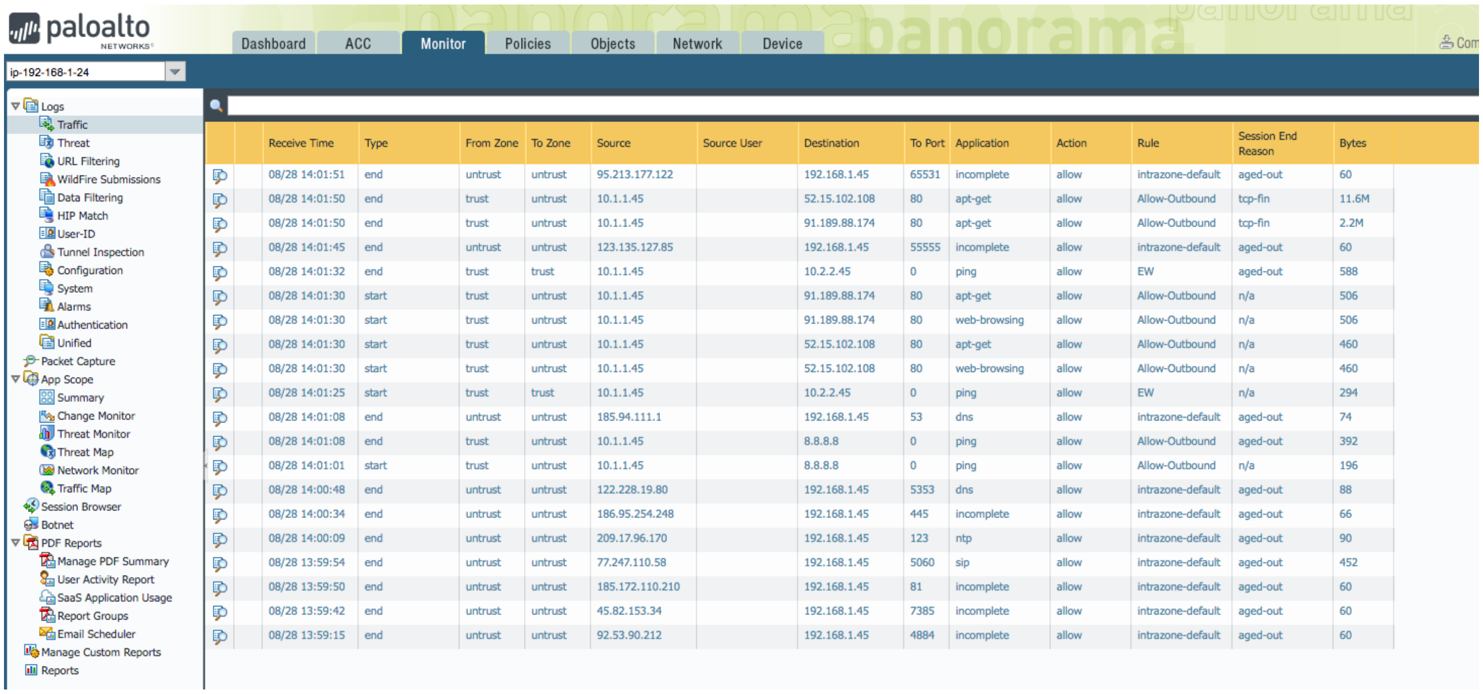
```
5JCMACC0N5JHD4:aws-tgw-eni-master dspears$ ssh -i AWS-Ohio-Key.pem ubuntu@3.14.117.6 -p 1001
The authenticity of host '[3.14.117.6]:1001 ([3.14.117.6]:1001)' can't be established.
ECDSA key fingerprint is SHA256:1wsSbr+KXCvcb4h4rzm+spcv+EXgp8+tIkewE15U81I.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[3.14.117.6]:1001' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-170-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Wed Aug 28 20:39:48 UTC 2019

  System load: 0.0          Memory usage: 5%   Processes:       81
  Usage of /:  10.3% of 7.74GB   Swap usage:   0%   Users logged in: 0

  Graph this data and manage this system at:
    https://landscape.canonical.com/

0 updates can be installed immediately.
0 of these updates are security updates.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

ubuntu@ip-10-1-1-45:~$
```

From here it is possible to perform some connectivity testing with PINGs and other things such as software updates:


3. ubuntu@ip-10-1-1

```
ubuntu@ip-10-1-1-45:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=36 time=11.9 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=36 time=11.7 ms
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 11.731/11.859/11.987/0.128 ms
ubuntu@ip-10-1-1-45:~$ ping 10.2.2.45
PING 10.2.2.45 (10.2.2.45) 56(84) bytes of data.
64 bytes from 10.2.2.45: icmp_seq=1 ttl=61 time=1.92 ms
64 bytes from 10.2.2.45: icmp_seq=2 ttl=61 time=1.51 ms
64 bytes from 10.2.2.45: icmp_seq=3 ttl=61 time=1.51 ms
^C
--- 10.2.2.45 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.512/1.649/1.921/0.197 ms
ubuntu@ip-10-1-1-45:~$ sudo apt-get update
sudo: unable to resolve host ip-10-1-1-45
Ign http://us-east-2.ec2.archive.ubuntu.com trusty InRelease
Get:1 http://us-east-2.ec2.archive.ubuntu.com trusty-updates InRelease [65.9 kB]
Hit http://us-east-2.ec2.archive.ubuntu.com trusty-backports InRelease
Hit http://us-east-2.ec2.archive.ubuntu.com trusty Release.gpg
Hit http://us-east-2.ec2.archive.ubuntu.com trusty Release
Get:2 http://us-east-2.ec2.archive.ubuntu.com trusty-updates/main Sources [431 kB]
Get:3 http://us-east-2.ec2.archive.ubuntu.com trusty-updates/restricted Sources [6,313 B]
Get:4 http://us-east-2.ec2.archive.ubuntu.com trusty-updates/universe Sources [231 kB]
```

It is also then possible to leverage Panorama and open up the logs on the firewall and see the traffic:



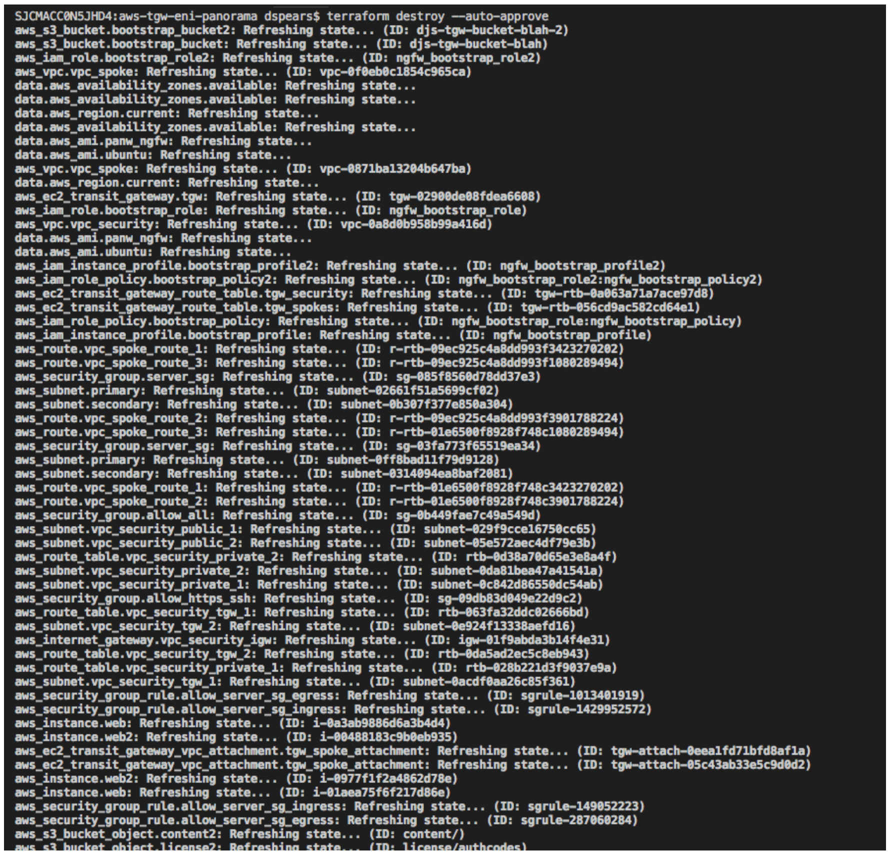
Receive Time	Type	From Zone	To Zone	Source	Source User	Destination	To Port	Application	Action	Rule	Session End Reason	Bytes
08/28 14:01:51	end	untrust	untrust	95.213.177.122		192.168.1.45	65531	incomplete	allow	intrazone-default	aged-out	60
08/28 14:01:50	end	trust	untrust	10.1.1.45		52.15.102.108	80	apt-get	allow	Allow-Outbound	tcp-fin	11.6M
08/28 14:01:50	end	trust	untrust	10.1.1.45		91.189.88.174	80	apt-get	allow	Allow-Outbound	tcp-fin	2.2M
08/28 14:01:45	end	untrust	untrust	123.135.127.85		192.168.1.45	55555	incomplete	allow	intrazone-default	aged-out	60
08/28 14:01:32	end	trust	trust	10.1.1.45		10.2.2.45	0	ping	allow	EW	aged-out	588
08/28 14:01:30	start	trust	untrust	10.1.1.45		91.189.88.174	80	apt-get	allow	Allow-Outbound	n/a	506
08/28 14:01:30	start	trust	untrust	10.1.1.45		91.189.88.174	80	web-browsing	allow	Allow-Outbound	n/a	506
08/28 14:01:30	start	trust	untrust	10.1.1.45		52.15.102.108	80	apt-get	allow	Allow-Outbound	n/a	460
08/28 14:01:30	start	trust	untrust	10.1.1.45		52.15.102.108	80	web-browsing	allow	Allow-Outbound	n/a	460
08/28 14:01:25	start	trust	trust	10.1.1.45		10.2.2.45	0	ping	allow	EW	n/a	294
08/28 14:01:08	end	untrust	untrust	185.94.111.1		192.168.1.45	53	dns	allow	intrazone-default	aged-out	74
08/28 14:01:08	end	trust	untrust	10.1.1.45		8.8.8.8	0	ping	allow	Allow-Outbound	aged-out	392
08/28 14:01:01	start	trust	untrust	10.1.1.45		8.8.8.8	0	ping	allow	Allow-Outbound	n/a	196
08/28 14:00:48	end	untrust	untrust	122.228.19.80		192.168.1.45	5353	dns	allow	intrazone-default	aged-out	88
08/28 14:00:34	end	untrust	untrust	186.95.254.248		192.168.1.45	445	incomplete	allow	intrazone-default	aged-out	66
08/28 14:00:09	end	untrust	untrust	209.17.96.170		192.168.1.45	123	ntp	allow	intrazone-default	aged-out	90
08/28 13:59:54	end	untrust	untrust	77.247.110.58		192.168.1.45	5060	sip	allow	intrazone-default	aged-out	452
08/28 13:59:50	end	untrust	untrust	185.172.110.210		192.168.1.45	81	incomplete	allow	intrazone-default	aged-out	60
08/28 13:59:42	end	untrust	untrust	45.82.153.34		192.168.1.45	7385	incomplete	allow	intrazone-default	aged-out	60
08/28 13:59:15	end	untrust	untrust	92.53.90.212		192.168.1.45	4884	incomplete	allow	intrazone-default	aged-out	60

It is also possible to log into the firewalls directly using the URLs/IPs from the output of the terraform deployment. The firewall username/passwords are:

```
Username: admin
Password: Pal0Alt0@123
```

12. Cleanup

The environment can be deleted by executing **Terraform destroy - -auto-approve**



```
SJCMACC0N5JHD4:aws-tgw-eni-panorama dspears$ terraform destroy --auto-approve
aws_s3_bucket.bootstrap_bucket2: Refreshing state... (ID: djs-tgw-bucket-blah-2)
aws_s3_bucket.bootstrap_bucket: Refreshing state... (ID: djs-tgw-bucket-blah)
aws_iam_role.bootstrap_role2: Refreshing state... (ID: ngfw_bootstrap_role2)
aws_vpc.vpc_spoke: Refreshing state... (ID: vpc-0f0eb0c1854c965ca)
data.aws_availability_zones.available: Refreshing state...
data.aws_availability_zones.available: Refreshing state...
data.aws_region.current: Refreshing state...
data.aws_availability_zones.available: Refreshing state...
data.aws_ami.panw_ngfw: Refreshing state...
data.aws_ami.ubuntu: Refreshing state...
aws_vpc.vpc_spoke: Refreshing state... (ID: vpc-0871ba13204b647ba)
data.aws_region.current: Refreshing state...
aws_ec2_transit_gateway.tgw: Refreshing state... (ID: tgw-02900de08fdea6608)
aws_iam_role.bootstrap_role: Refreshing state... (ID: ngfw_bootstrap_role)
aws_vpc.vpc_security: Refreshing state... (ID: vpc-0a8d0b958b99a416d)
data.aws_ami.panw_ngfw: Refreshing state...
data.aws_ami.ubuntu: Refreshing state...
aws_iam_instance_profile.bootstrap_profile2: Refreshing state... (ID: ngfw_bootstrap_profile2)
aws_iam_role_policy.bootstrap_policy2: Refreshing state... (ID: ngfw_bootstrap_role2:ngfw_bootstrap_policy2)
aws_ec2_transit_gateway_route_table.tgw_security: Refreshing state... (ID: tgw-rtb-0a063a71a7ace97d8)
aws_ec2_transit_gateway_route_table.tgw_spokes: Refreshing state... (ID: tgw-rtb-056cd9ac582cd64e1)
aws_iam_role_policy.bootstrap_policy: Refreshing state... (ID: ngfw_bootstrap_role:ngfw_bootstrap_policy)
aws_iam_instance_profile.bootstrap_profile: Refreshing state... (ID: ngfw_bootstrap_profile)
aws_route.vpc_spoke_route_1: Refreshing state... (ID: r-rtb-09ec925c4a8dd993f3423270202)
aws_route.vpc_spoke_route_3: Refreshing state... (ID: r-rtb-09ec925c4a8dd993f1080289494)
aws_security_group.server_sg: Refreshing state... (ID: sg-085f8560d78dd37e3)
aws_subnet.primary: Refreshing state... (ID: subnet-02661f51a5699cf02)
aws_subnet.secondary: Refreshing state... (ID: subnet-0b307f377e850a304)
aws_route.vpc_spoke_route_2: Refreshing state... (ID: r-rtb-09ec925c4a8dd993f3901788224)
aws_route.vpc_spoke_route_3: Refreshing state... (ID: r-rtb-01e6500f8928f748c1080289494)
aws_security_group.server_sg: Refreshing state... (ID: sg-03fa773f65519ea34)
aws_subnet.primary: Refreshing state... (ID: subnet-0ff8bad11f79d9128)
aws_subnet.secondary: Refreshing state... (ID: subnet-0314094ea8baf2081)
aws_route.vpc_spoke_route_1: Refreshing state... (ID: r-rtb-01e6500f8928f748c3423270202)
aws_route.vpc_spoke_route_2: Refreshing state... (ID: r-rtb-01e6500f8928f748c3901788224)
aws_security_group.allow_all: Refreshing state... (ID: sg-0b449fae7c49a549d)
aws_subnet.vpc_security_public_1: Refreshing state... (ID: subnet-029f9cce16750cc65)
aws_subnet.vpc_security_public_2: Refreshing state... (ID: subnet-05e572aec4df79e3b)
aws_route_table.vpc_security_private_2: Refreshing state... (ID: rtb-0d38a70d65e3e8a4f)
aws_subnet.vpc_security_private_2: Refreshing state... (ID: subnet-0da81bea47a41541a)
aws_subnet.vpc_security_private_1: Refreshing state... (ID: subnet-0c842d86550dc54ab)
aws_security_group.allow_https_ssh: Refreshing state... (ID: sg-09db83d049e22d9c2)
aws_route_table.vpc_security_tgw_1: Refreshing state... (ID: rtb-063fa32ddc02666bd)
aws_subnet.vpc_security_tgw_2: Refreshing state... (ID: subnet-0e924f13338aefd16)
aws_internet_gateway.vpc_security_igw: Refreshing state... (ID: igw-01f9abda3b14f4e31)
aws_route_table.vpc_security_tgw_2: Refreshing state... (ID: rtb-0da5ad2ec5c8eb943)
aws_route_table.vpc_security_private_1: Refreshing state... (ID: rtb-028b221d3f9037e9a)
aws_subnet.vpc_security_tgw_1: Refreshing state... (ID: subnet-0acdf0aa26c85f361)
aws_security_group_rule.allow_server_sg_egress: Refreshing state... (ID: sgrule-1013401919)
aws_security_group_rule.allow_server_sg_ingress: Refreshing state... (ID: sgrule-1429952572)
aws_instance.web: Refreshing state... (ID: i-0a3ab9886d6a3b4d4)
aws_instance.web2: Refreshing state... (ID: i-00488183c9b0eb935)
aws_ec2_transit_gateway_vpc_attachment.tgw_spoke_attachment: Refreshing state... (ID: tgw-attach-0eea1fd71bfd8af1a)
aws_ec2_transit_gateway_vpc_attachment.tgw_spoke_attachment: Refreshing state... (ID: tgw-attach-05c43ab33e5c9d0d2)
aws_instance.web2: Refreshing state... (ID: i-0977f1f2a4862d78e)
aws_instance.web: Refreshing state... (ID: i-01aea75f6f217d86e)
aws_security_group_rule.allow_server_sg_ingress: Refreshing state... (ID: sgrule-149052223)
aws_security_group_rule.allow_server_sg_egress: Refreshing state... (ID: sgrule-287060284)
aws_s3_bucket_object.content2: Refreshing state... (ID: content/)
aws_s3_bucket_object.license2: Refreshing state... (ID: license/authcodes)
```