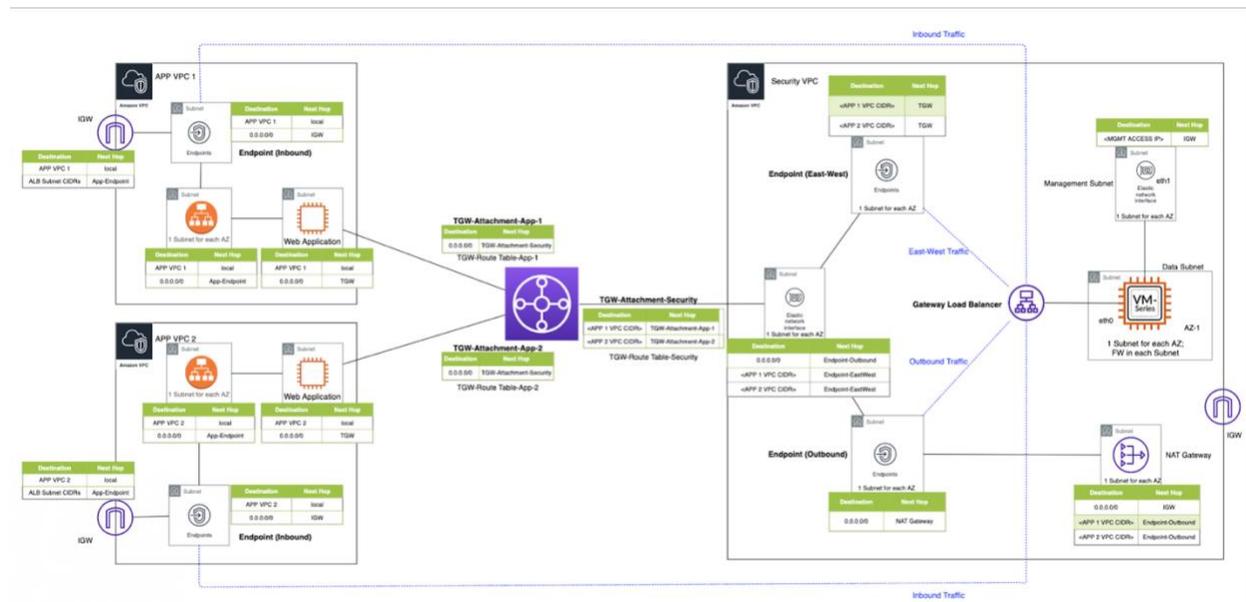


Introduction

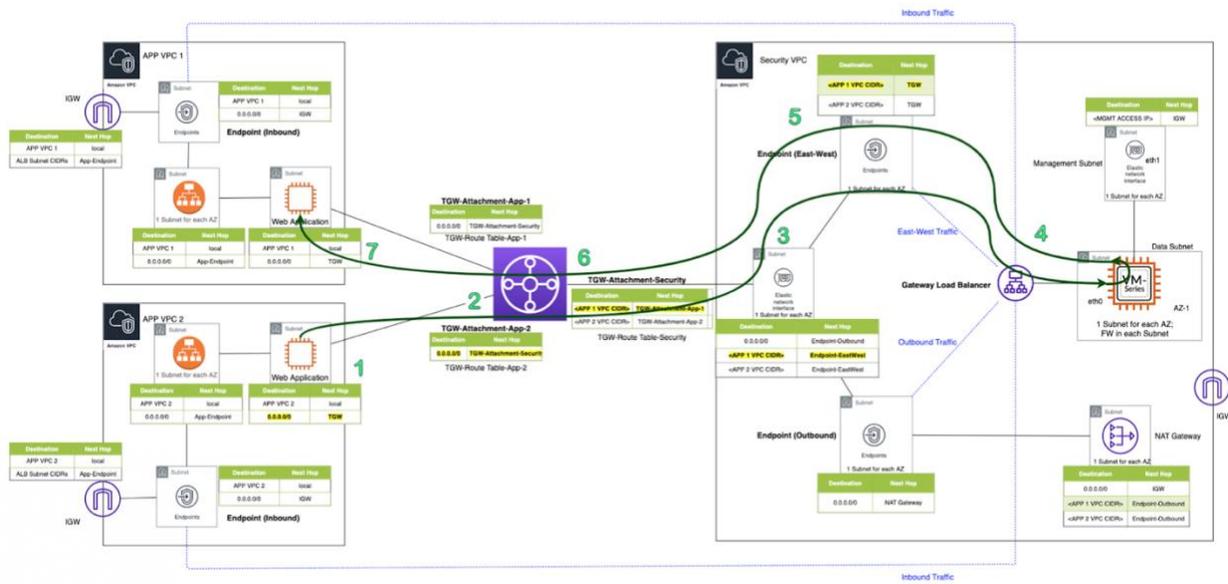
With the introduction of the Gateway Load Balancer (GWLB) in mid-November 2020, AWS provided its customers with any port, load-balancing router. Prior to that, Azure and GCP were the only public clouds that had such a construct. Customers use these to provide a security layer that is scalable, resilient, and adaptable. In the AWS implementation, endpoints are an integral part of the solution but are not a new concept in AWS. They connect elastic network interfaces (ENIs) to targets (e.g. GWLB) via "worm holes" in the fabric and have been used with network load balancers (NLBs) for some time. These worm holes in the fabric bypass the usual routing constructs and can perform result in some difficulty when troubleshooting. In this blog post, we will trace the flow of a request originating from a client in one VPC (network 10.102.0.0/16) to a server in another VPC (network 10.101.0.0/16). The infrastructure was deployed using the following TerraForm template:

<https://github.com/wwce/terraform/tree/master/aws/GWLB-Demo>

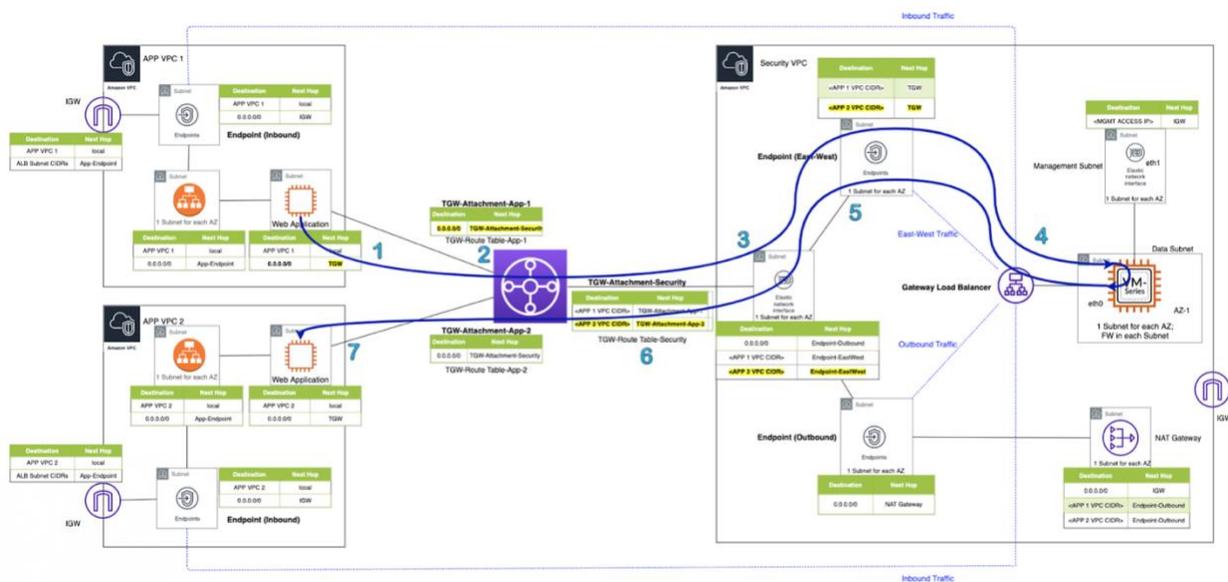
and follows current best practices regarding architecture:



This architecture also supports inbound and outbound traffic flows, which are treated separately in other blog posts. Today, we will focus on the following request flow:



And the corresponding response flow:



Note that AWS assigns unique resource identifiers to each resource in the environment. Examples include `tgw-attach-0b86ac38ab82dff9` or `subnet-0e1119f6fc333ea6d`. Every resource created is assigned one of these unique identifiers. This means that although the template creates the environment using identical resources, the individual resource identifiers will be different.

N.B. - Routes to the 104.219.136.0/21 and 107.64.0.0/10 subnets pointing to the internet gateway (IGW) in the APP VPCs are the

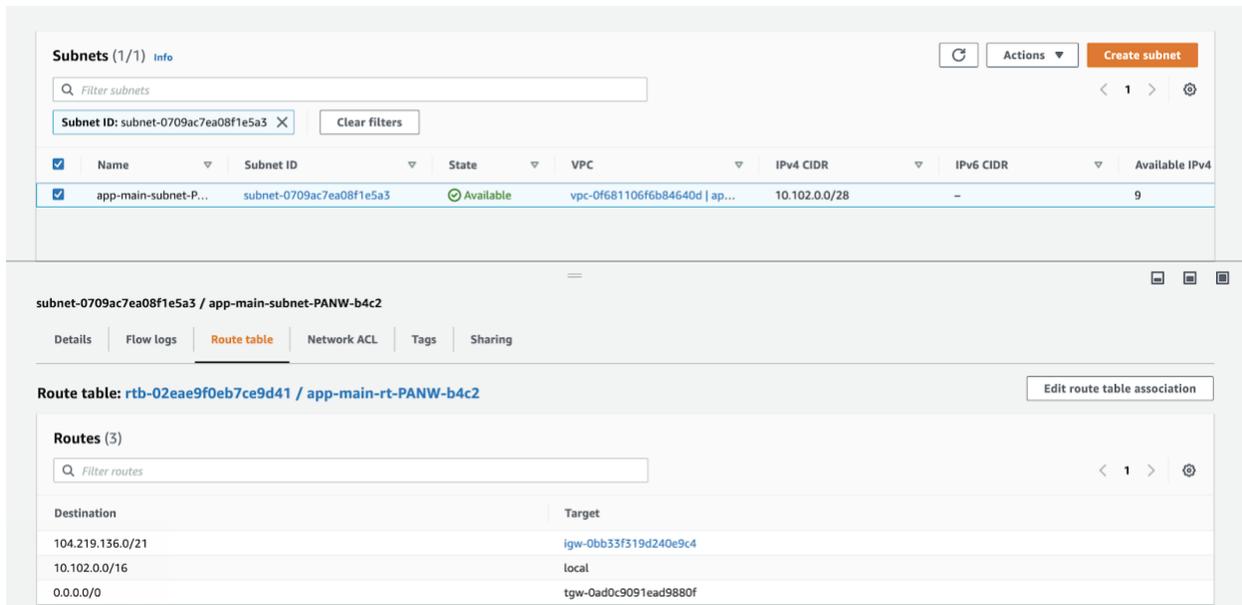
author's primary/secondary ISP subnets and were added post-deployment to facilitate direct access to the hosts in the VPCs for troubleshooting. They do not exist in the publicly-available templates and can be ignored.

Request Step 1 - Can We Talk?

The process begins when a user/process on a host (IP 10.102.0.5) in APP VPC 2 needs to connect to a host (IP 10.101.0.4) in APP VPC 1. Looking at the EC2 instance, we can see the IP address as well as the subnet membership:

The screenshot displays the AWS Management Console interface for an EC2 instance. At the top, there's a search bar with 'app-PANW-b4c2' entered. Below the search bar, a table lists instance details for 'app-PANW-b4c2'. The instance is in a 'Running' state, using a 't2.micro' instance type, and is located in the 'us-east-1c' availability zone. The public IPv4 address is '3.208.253.71'. Below the table, the 'Instance: i-049eb3bcd0d6951f (app-PANW-b4c2)' details are shown. The 'Instance summary' section includes: Instance ID (i-049eb3bcd0d6951f), Instance state (Running), Instance type (t2.micro), Public IPv4 address (3.208.253.71), Elastic IP addresses (3.208.253.71), IAM Role (-), Private IPv4 addresses (10.102.0.5), Private IPv4 DNS (ip-10-102-0-5.ec2.internal), VPC ID (vpc-0f681106f6b84640d), and Subnet ID (subnet-0709ac7ea08f1e5a3).

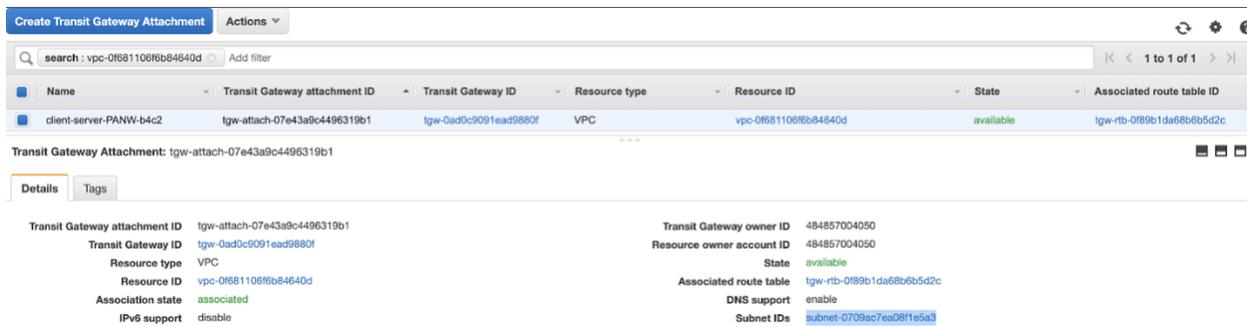
The subnet route table has a default route pointing to the transit gateway (TGW) as the next hop.



The requester does an internal route lookup and puts the packet out on the wire (local subnet) which has a default route via the TGW.

Request Step 2 - Transit Gateway (TGW)

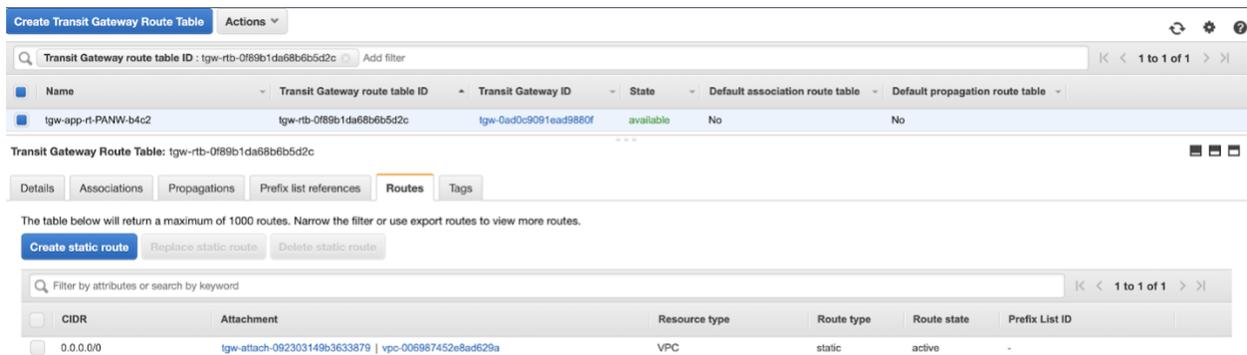
The TGW is connected to the VPC via a Transit Gateway Attachment. To see this association, we navigate to the Transit Gateway Attachment list and filter on the VPC hosting the requester:



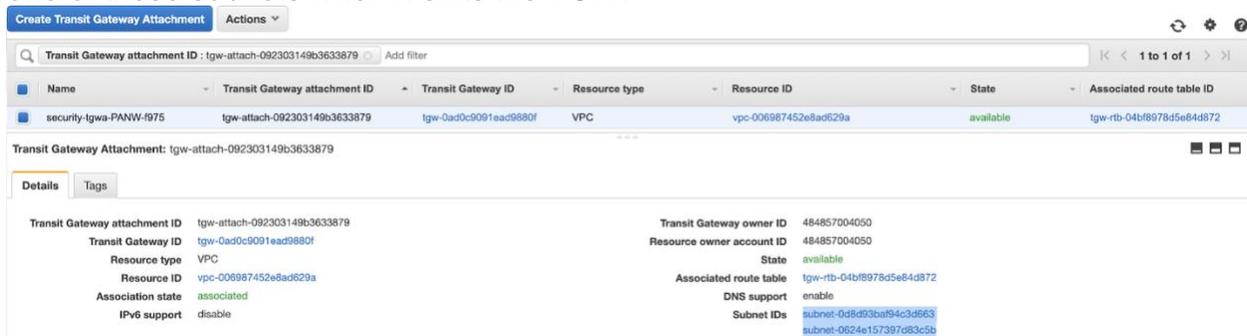
Note that when creating a TGW Attachment, a subnet must be specified and traffic can only be routed to a TGW Attachment in the same availability zone (AZ). In this case, the TGW Attachment and the host exist on the same subnet (and hence same AZ).

Routing within the TGW is handled via route tables associated with the TGW attachment. In the above picture, we can see that the route table associated with the TGW Attachment is tgw-rtb-0f89b1da68b6b5d2c. Clicking on the link to the route table

and inspecting the routes, we can see that the default route points to yet another attachment:



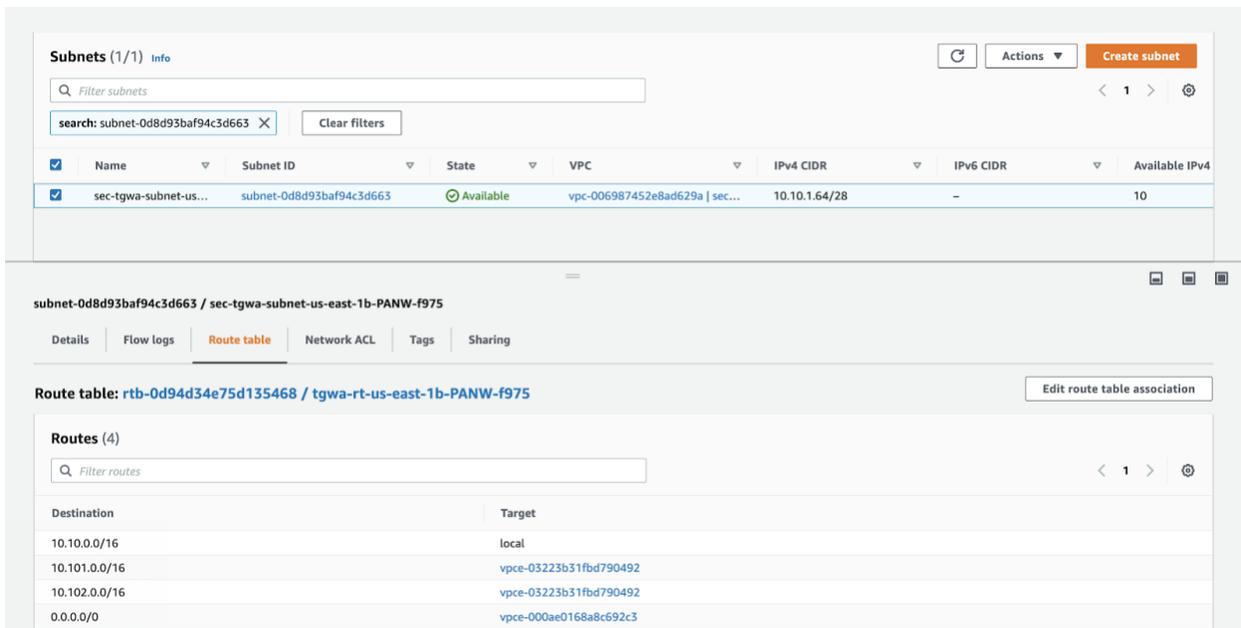
Following the rabbit a little further down the hole, we find that the attachment is associated with two different subnets. Traffic from the requester gets dropped off into one of these subnets when it exits the TGW.



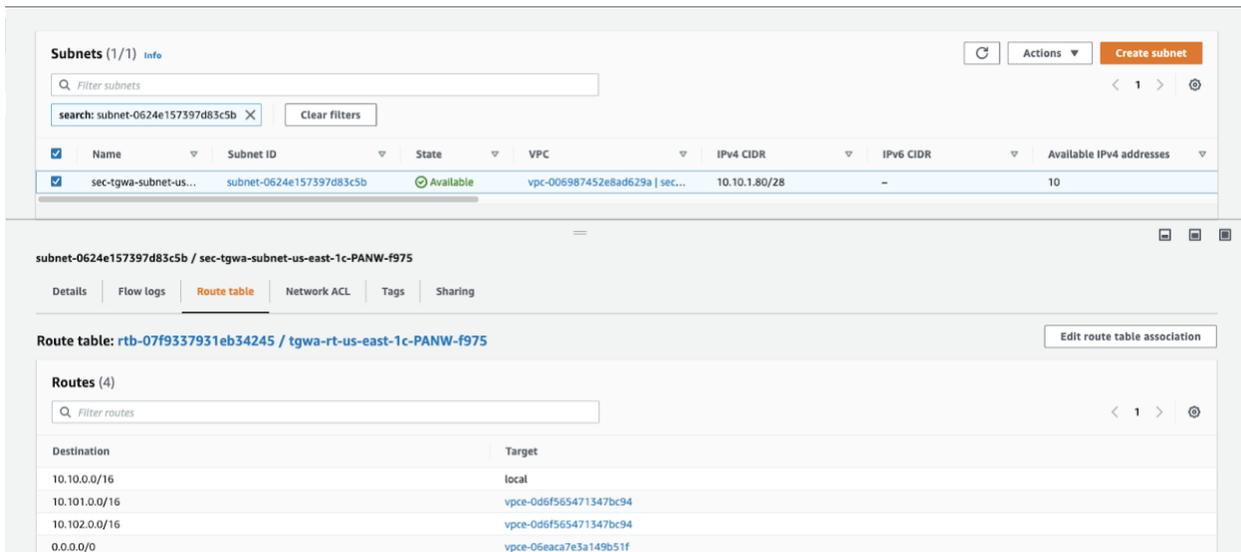
N.B. - The TGW has the ability to load balance across as well as ensure traffic symmetry. More information on traffic symmetry can be found here:

<https://docs.aws.amazon.com/vpc/latest/tgw/transit-gateway-appliance-scenario.html>

If we look at the route table of one of the subnets, we can see that the traffic is directed to a GWLB endpoint:



The route table associated with the other subnet looks similar (note that the endpoint ID is different):



In this case, the traffic is sent to the same Endpoint irrespective of whether we are attempting to reach APP VPC 1 or APP VPC 2.

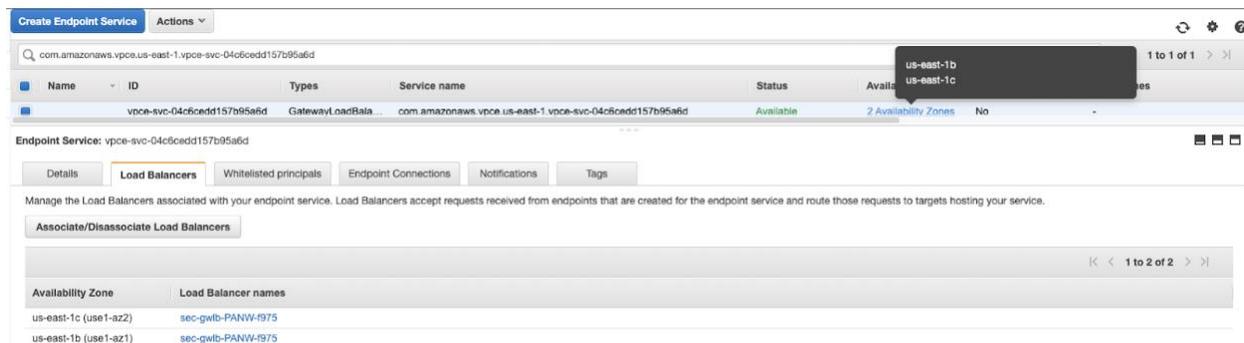
Request Step 3 - The GWLB Endpoint

Recall that endpoints are ENIs that provide direct access to services within the VPC. They are AZ-specific constructs and are instantiated in every AZ where service access is required. The Endpoint is connected to the GWLB via an Endpoint Service.

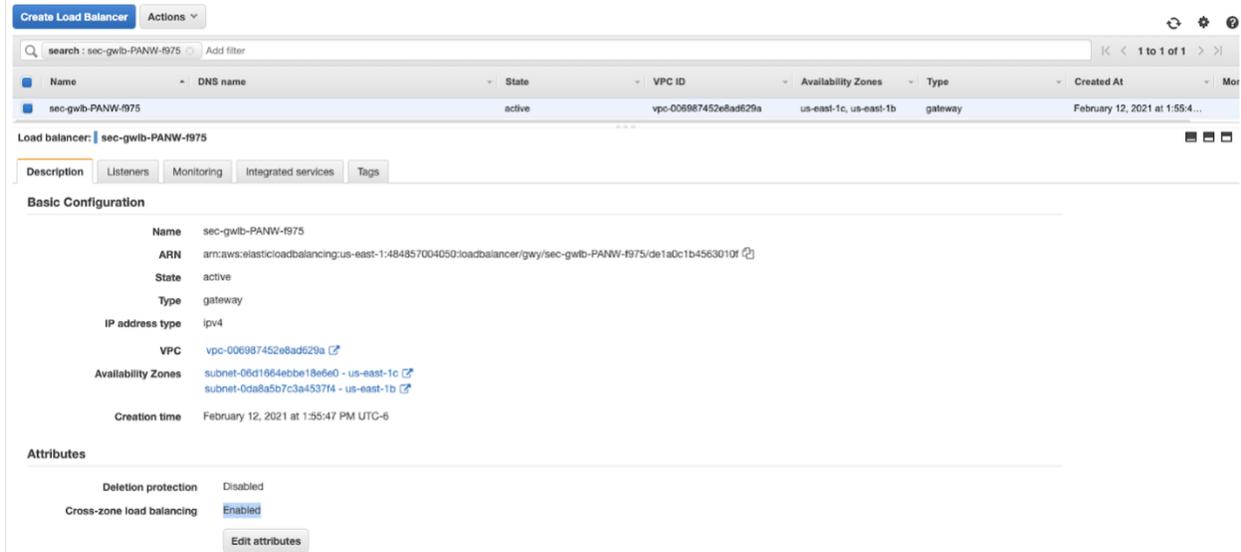
Clicking on one of the targets associated with either APP VPC 1 (10.101.0.0/16) or APP VPC 2 (10.102.0.0/16), we can see additional information about the endpoint, including the associated Endpoint Service:



If we then look at Endpoint Services, we can see that this service is associated with a multi-AZ load balancer (also note that the Endpoint Service is associated with multiple AZs):



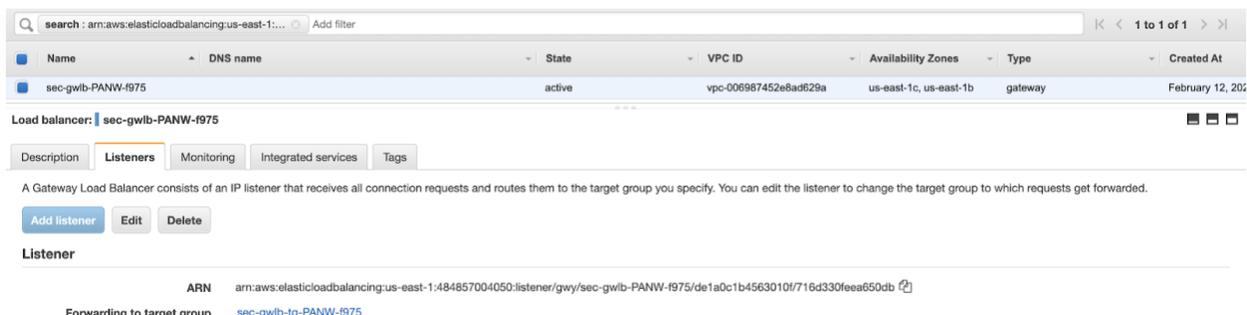
Clicking on the loadbalancer, we can see more detailed information:



Pro Tip: If it has not already been done, "Cross-zone load balancing" should be enabled in the attributes. This ensures that the GWLB can use any backend pool member in any availability zone and facilitates resiliency.

Request Step 4 - The Firewalls

The GWLB uses Generic Network Virtualization Encapsulation (GENEVE) to create an overlay network between the load balancer and the firewalls. At present, this overlay network is not connected to the firewalls virtual router, which improves packet handling efficiency but requires that all traffic ingress/egress the FW via the GENEVE tunnel. Under the hood, the GWLB is a souped-up NLB and the configuration is very similar. Once the traffic reaches the GWLB, it is distributed amongst the available backend pool members. Looking at the listeners for the GWLB, we see one of the first differences between the GWLB and a standard NLB:



The GWLB is an any port load balancer and consequently no port(s) are specified/required. All TCP/UDP traffic is load balanced to the associated target group.

Selecting the target group, we see that it is comprised of the FW in the security VPC:

The screenshot shows the AWS Target Group console for 'sec-gwlb-tg-PANW-f975'. The 'Basic configuration' section includes:

Target type: Instance	Protocol: Port: GENEVE: 6081	VPC: vpc-006987452e8ad629a	Load balancer: sec-gwlb-PANW-f975
-----------------------	------------------------------	----------------------------	-----------------------------------

The 'Registered targets (2)' table lists the following instances:

Instance ID	Name	Port	Zone	Status	Status details
i-0e6c62c3020a82cee	FW-us-east-1c-PANW-f975	6081	us-east-1c	healthy	
i-0bc983c8ae20ace5a	FW-us-east-1b-PANW-f975	6081	us-east-1b	healthy	

The FW are targeted by instance ID, which ensures source IP preservation but requires that the management and first data plane interface be swapped.

Selecting one of the targets, we can see the firewall details:

The screenshot shows the AWS Instance console for 'Instance: i-0e6c62c3020a82cee (FW-us-east-1c-PANW-f975)'. The 'Instance summary' section includes:

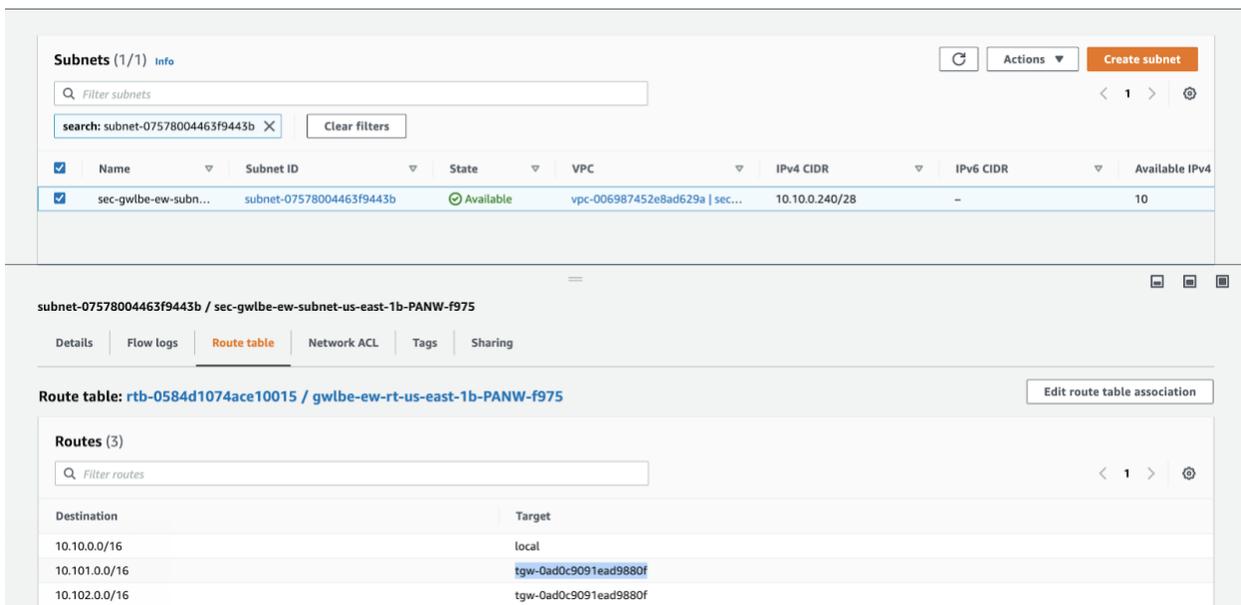
Instance ID: i-0e6c62c3020a82cee (FW-us-east-1c-PANW-f975)	Public IPv4 address: -	Private IPv4 addresses: 10.10.0.28, 10.10.0.100
Instance state: Running	Public IPv4 DNS: -	Private IPv4 DNS: ip-10-10-0-100.ec2.internal
Instance type: m5.xlarge	Elastic IP addresses: 52.7.218.8 (fw-mgmt-eip-us-east-1c-PANW-f975) [Public IP]	VPC ID: vpc-006987452e8ad629a (sec-vpc-PANW-f975)
AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations. Learn more	IAM Role: iam-role-PANW-f975	Subnet ID: subnet-06d1664ebbe18e6e0 (sec-data-subnet-us-east-1c-PANW-f975)

Request Step 5 - Return to the GWLB Endpoint

The permitted request is returned to the GWLB via the GENEVE tunnel and then back to the endpoint. Recall that the ID of the endpoint is vpce-03223b31fd790492. If we take a closer look at that endpoint, we can determine the subnet that it resides in:

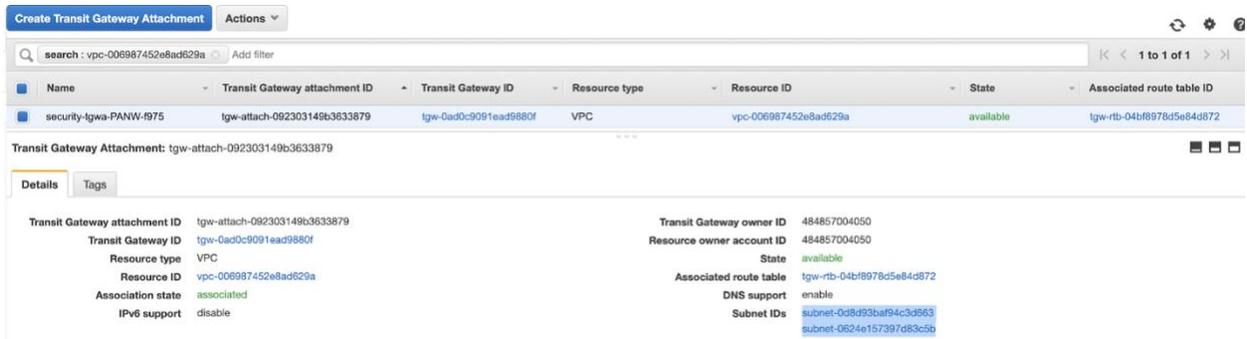


The subnet route table has the next hop to the destination as the TGW:



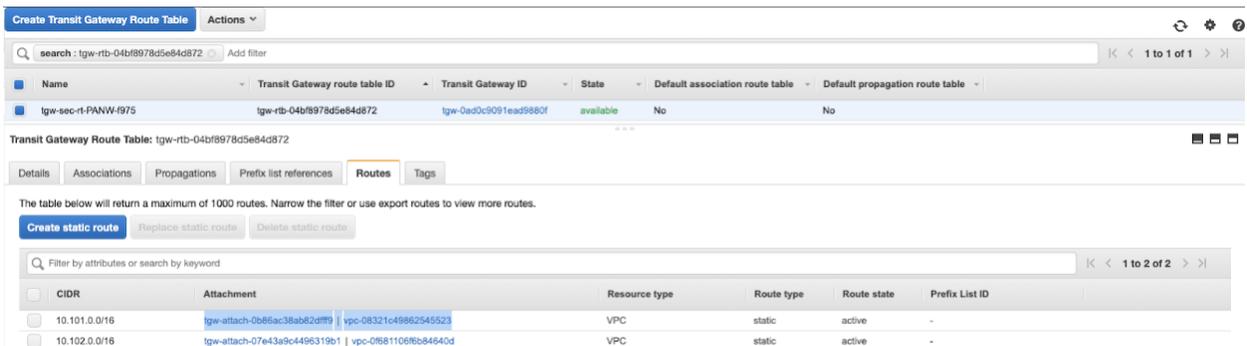
Request Step 6 - Return to the TGW

The TGW is connected to the VPC at the subnet level via a Transit Gateway Attachment. To see this association, we navigate to the Transit Gateway Attachment list in the VPC section of the GUI and filter on the security VPC (vpc-006987452e8ad629a in this example):

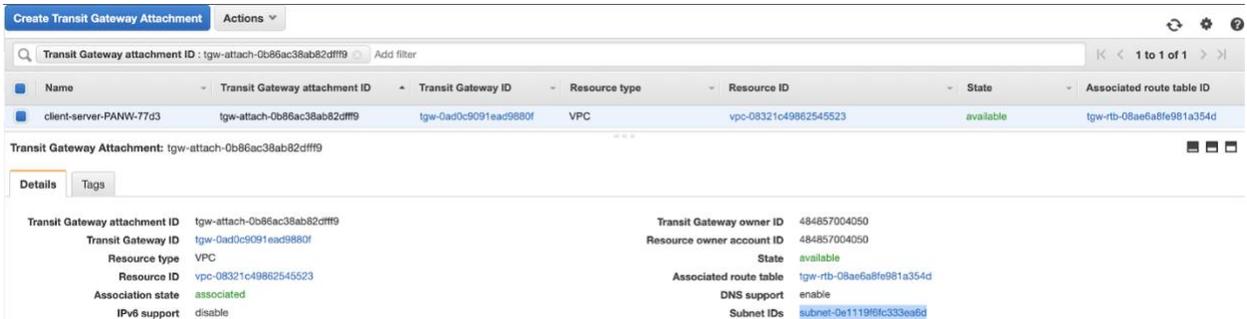


The TGW attachment and the endpoint are both in the same AZ so the packets can be routed as required.

Recall that routing within the TGW is handled via route tables associated with the TGW attachment. In the above picture, we can see that the route table associated with the TGW attachment is tgw-rtb-04bf8978d5e84d872. Clicking on the link to the route table and inspecting the routes, we can see that the route to the target subnet (10.101.0.0/16) points to another attachment (tgw-attach-0b86ac38ab82dff9):



Following this rabbit a little further down the rabbit hole, we find that the attachment is associated with a single subnet. Traffic exiting the TGW gets dropped off into this subnet.



Request Step 7 - The Eagle Has Landed

The request exits the TGW and gets dropped off into the subnet. Inspection of the subnet route table reveals that any traffic destined for the VPC network is delivered directly to the target:

The screenshot shows the AWS console interface for Subnets and Route tables. The top section displays a list of subnets with the following data:

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	Available IPv4
app-main-subnet-P...	subnet-0e1119f6fc333ea6d	Available	vpc-08321c49862545523 ap...	10.101.0.0/28	-	9

The bottom section shows the route table for the selected subnet: **Route table: rtb-08cda79410c2378c9 / app-main-rt-PANW-77d3**. It lists three routes:

Destination	Target
104.219.136.0/21	igw-01c49479188ae080e
10.101.0.0/16	local
0.0.0.0/0	tgw-0ad0c9091ead9880f

Inspection of the target host reveals that it resides on the destination network. This tells us that the traffic exiting the TGW is delivered directly to the target.

The screenshot shows the AWS console interface for an EC2 Instance. The top section displays a list of instances with the following data:

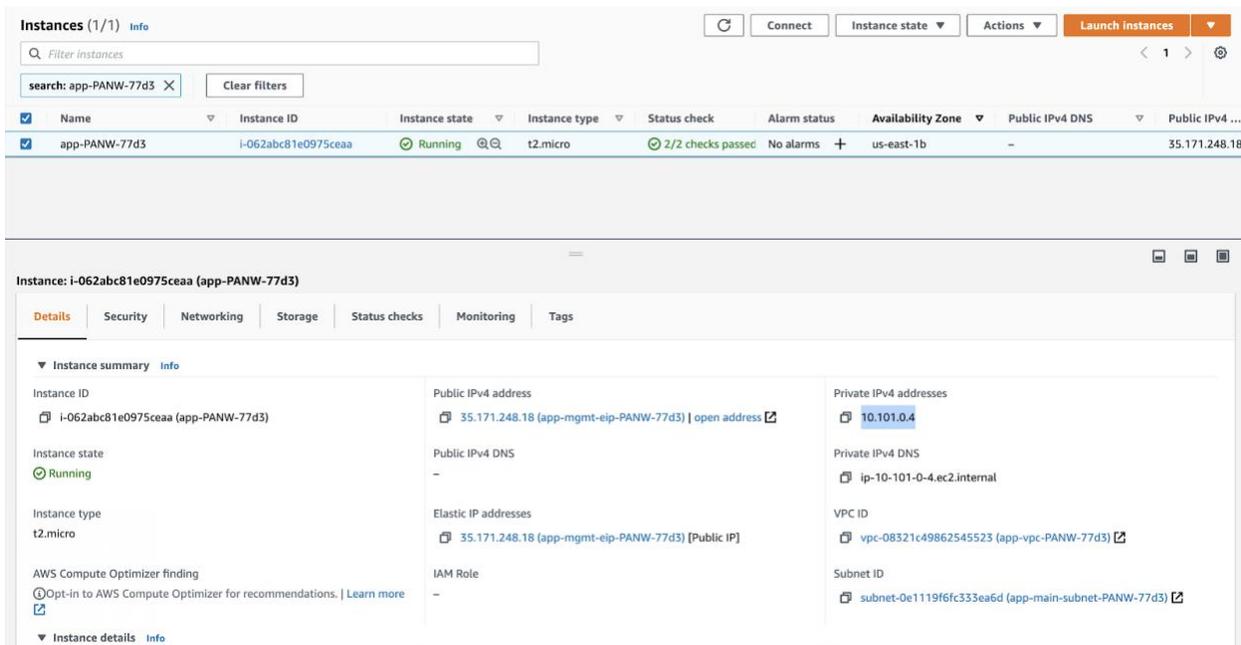
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
app-PANW-77d3	i-062abc81e0975ceaa	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b	-	35.171.248.18

The bottom section shows the details for the instance **i-062abc81e0975ceaa (app-PANW-77d3)**. Key details include:

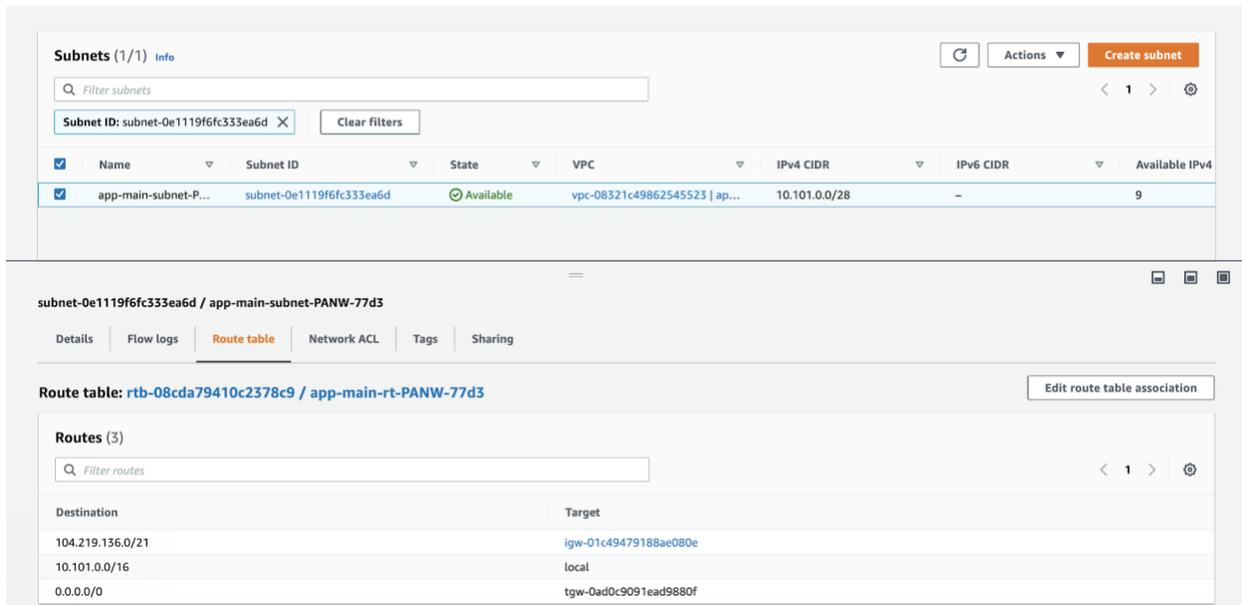
- Instance ID:** i-062abc81e0975ceaa (app-PANW-77d3)
- Instance state:** Running
- Instance type:** t2.micro
- Public IPv4 address:** 35.171.248.18 (app-mgmt-eip-PANW-77d3) | open address
- Private IPv4 addresses:** 10.101.0.4
- Subnet ID:** subnet-0e1119f6fc333ea6d (app-main-subnet-PANW-77d3)

Response Step 1 - The Destination Designs to Respond

We start with the target host and note the assigned IP address (10.101.0.4) as well as subnet membership:



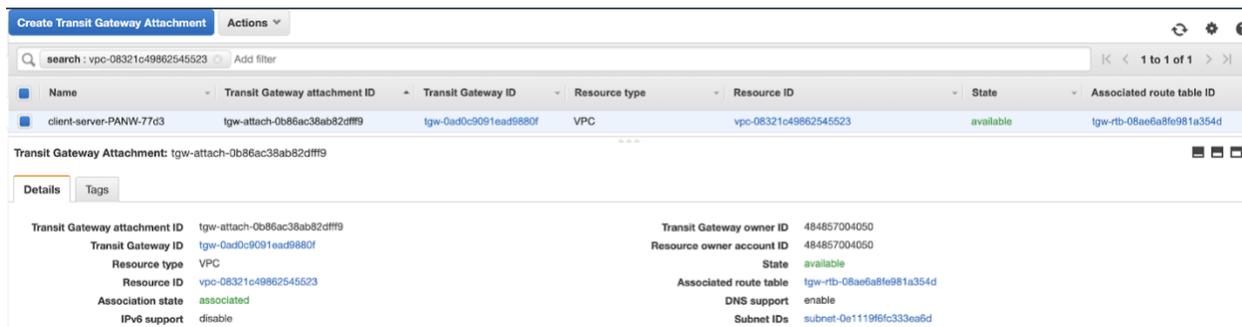
Clicking on the subnet and looking at the associated routing table, we see that the default route for the subnet points to the transit gateway (TGW) as the next hop.



Assuming a listener on the requested port, the responder does an internal route lookup and puts the packet out on the wire (local subnet) which has a default route via the TGW.

Response Step 2 - Transit Gateway (TGW)

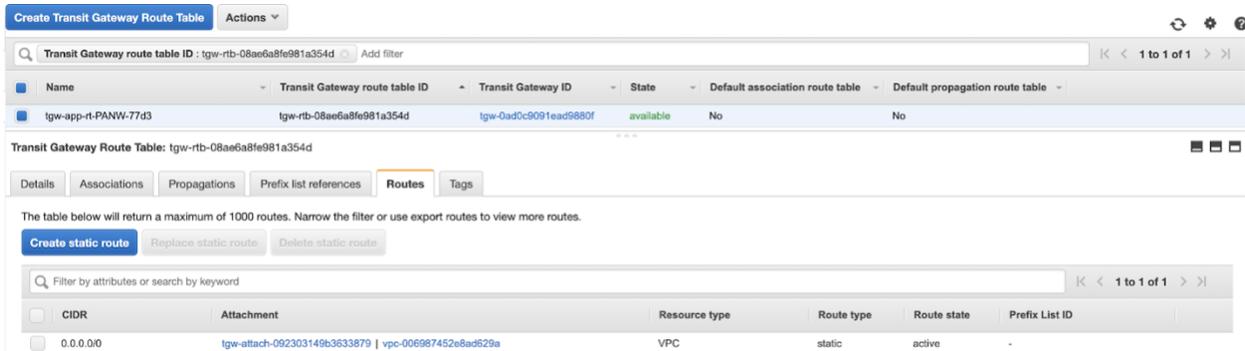
The TGW is connected to the VPC via a Transit Gateway Attachment. To see this association, we navigate to the Transit Gateway Attachment list and filter on the VPC hosting the target (vpc-08321c49862545523):



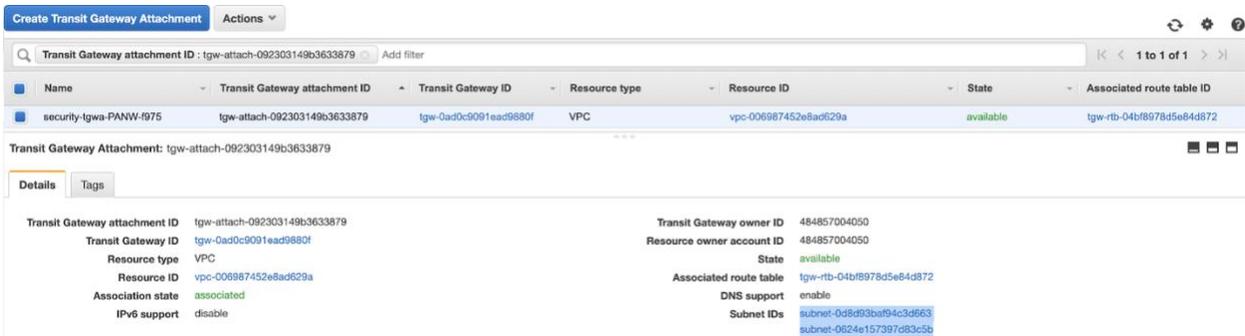
Note that when creating a TGW Attachment, a subnet must be specified and traffic can only be routed to a TGW Attachment in the same availability zone (AZ). In this case, the TGW Attachment and the host exist on the same subnet (and hence same AZ).

Routing within the TGW is handled via route tables associated with the TGW attachment. In the above picture, we can see that the route table associated with the

TGW attachment is tgw-rtb-08ae6a8fe981a354d. Clicking on the link to the route table and inspecting the routes, we can see that the default route points to a different attachment:



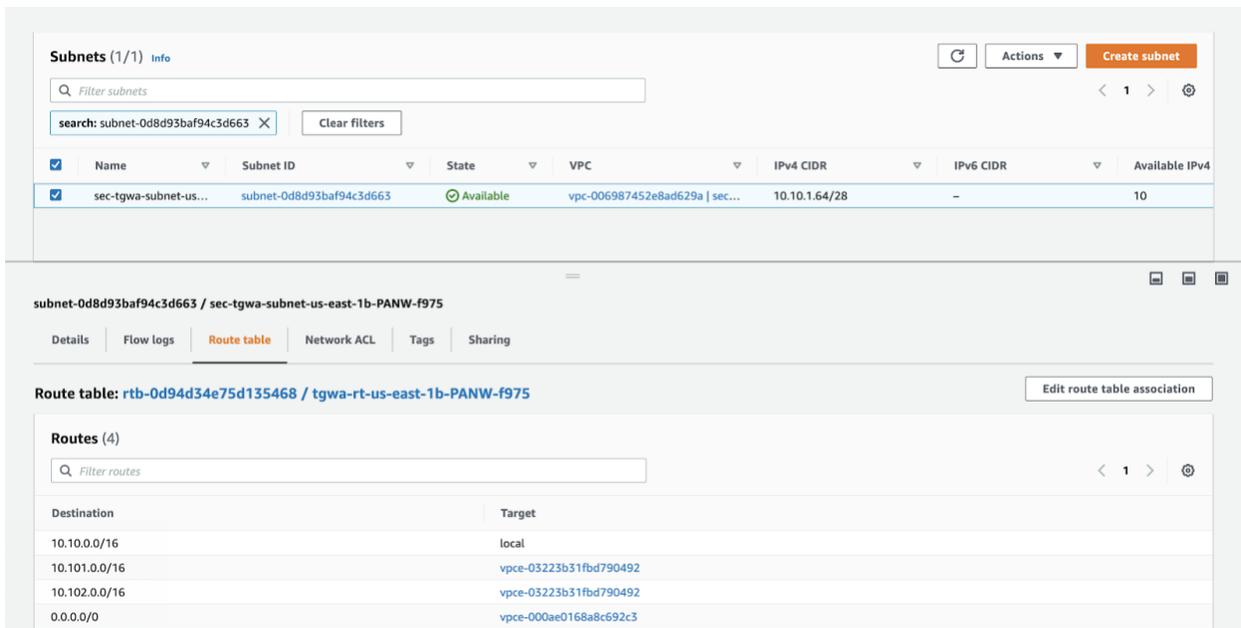
Following the rabbit a little further down the hole, we find that the attachment is associated with two different subnets. Traffic from the requester gets dropped off into one of these subnets when it exits the TGW.



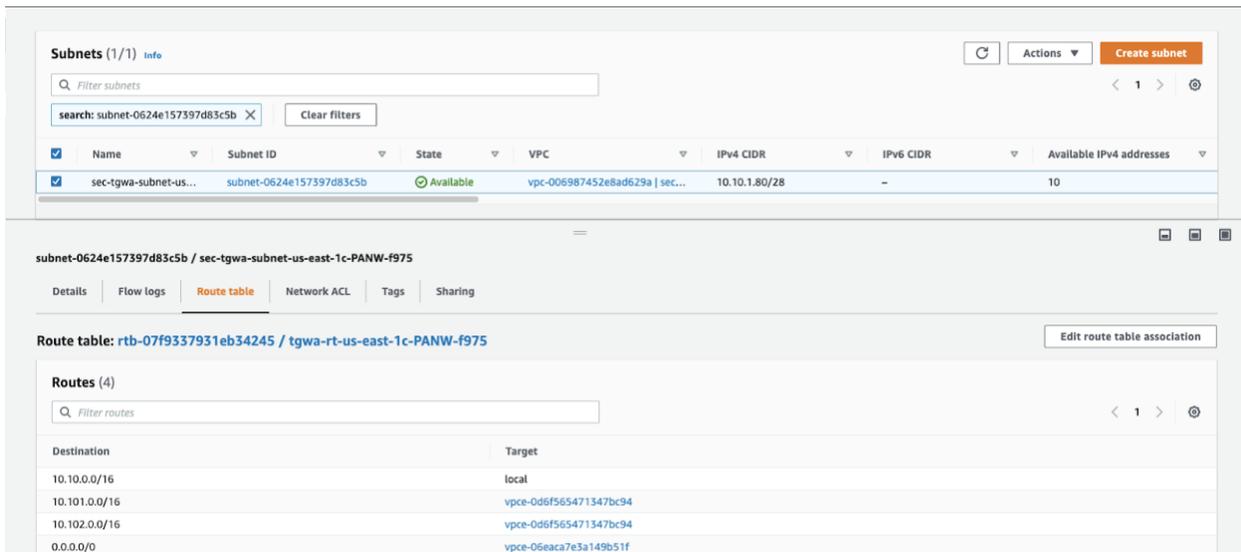
N.B. - The TGW has the ability to load balance across as well as ensure traffic symmetry. More information on traffic symmetry can be found here:

<https://docs.aws.amazon.com/vpc/latest/tgw/transit-gateway-appliance-scenario.html>

If we look at the route table of one of the subnets, we can see that the traffic is directed to a GWLB endpoint:



The route table associated with the other subnet looks similar (note that the Endpoint ID is different):



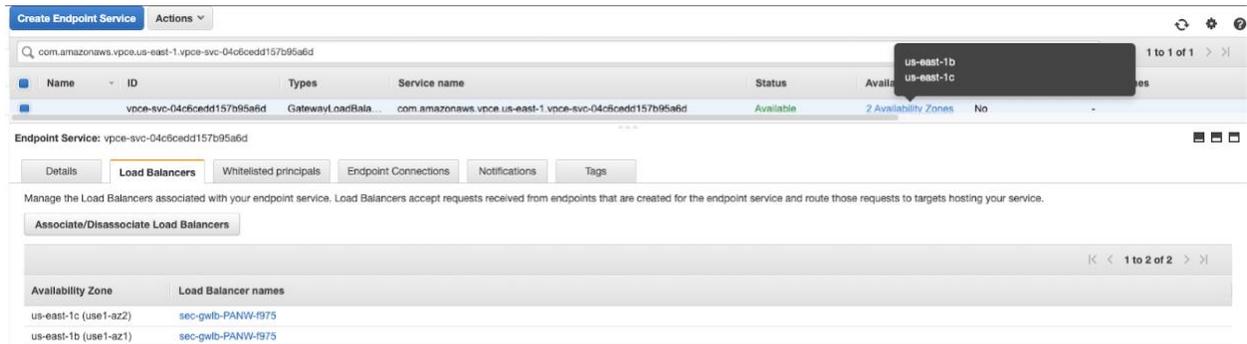
Response Step 3 - The GWLB Endpoint

Recall that endpoints are ENIs that provide direct access to services within the VPC. ENIs are AZ-specific constructs and are instantiated in every AZ where service access is required. An Endpoint is connected to the GWLB via an Endpoint Service. In this case, the traffic is sent to the same Endpoint irrespective of whether we are attempting to reach APP VPC 1 or APP VPC 2. To see more information about this connection,

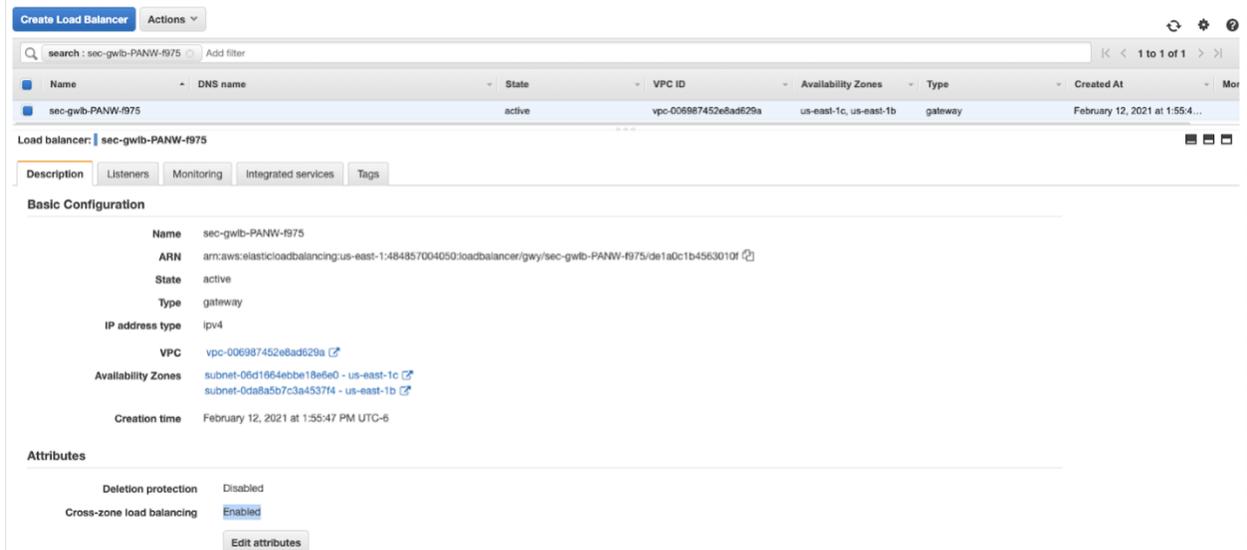
click on the target associated with either APP VPC 1 (10.101.0.0/16) or APP VPC 2 (10.102.0.0/16). The subsequent page shows additional information about the endpoint, including the associated Endpoint Service:



If we then look at Endpoint Services, we can see that this service is associated with a multi-AZ load balancer (also note that the Endpoint Service is associated with multiple AZs):



Clicking on the loadbalancer, we can see more detailed information:



Pro Tip: If it has not already been done, "Cross-zone load balancing" should be enabled in the attributes. This ensures that the GWLB can use any backend pool member in any availability zone and facilitates resiliency.

Response Step 4 - The Firewalls

As mentioned earlier, there is no port associated with the listener on the GWLB. All TCP/UDP traffic is load balanced to the associated target group.

The screenshot shows the AWS Management Console interface for a Gateway Load Balancer. The main heading is "sec-gwlb-PANW-f975". Below this, there are tabs for "Description", "Listeners", "Monitoring", "Integrated services", and "Tags". The "Listeners" tab is active, showing a single listener configuration. The listener is named "sec-gwlb-PANW-f975" and is in an "active" state. It is associated with VPC ID "vpc-006987452e8ad629a" and Availability Zones "us-east-1c, us-east-1b". The listener type is "gateway" and was created on "February 12, 2020". Below the listener details, there are buttons for "Add listener", "Edit", and "Delete". The "Forwarding to target group" is set to "sec-gwlb-tg-PANW-f975".

Selecting the target group, we see that it is comprised of the FW in the security VPC:

The screenshot shows the AWS Management Console interface for a Target Group. The main heading is "sec-gwlb-tg-PANW-f975". Below this, there are tabs for "Group details", "Targets", "Monitoring", and "Tags". The "Targets" tab is active, showing a list of registered targets. The target group is named "sec-gwlb-tg-PANW-f975" and is in an "active" state. It is associated with VPC ID "vpc-006987452e8ad629a" and Availability Zones "us-east-1c, us-east-1b". The target group type is "gateway" and was created on "February 12, 2020". Below the target group details, there are buttons for "Deregister" and "Register targets". The "Registered targets (2)" section shows a table with the following data:

Instance ID	Name	Port	Zone	Status	Status details
i-0e6c62c3020a82cee	FW-us-east-1c-PANW-f975	6081	us-east-1c	healthy	
i-0bc983c8ae20ace5a	FW-us-east-1b-PANW-f975	6081	us-east-1b	healthy	

The FW are targeted by instance ID, which ensures source IP preservation but requires that the management and first data plane interface be swapped.

Selecting one of the targets, we can see the firewall details:

The screenshot shows the AWS Management Console interface for an EC2 instance. At the top, there's a header for 'Instances (1/1)' with a search bar and various action buttons like 'Connect', 'Instance state', 'Actions', and 'Launch instances'. Below this is a table with one instance listed: 'FW-us-east-1c-PANW-f975' with ID 'i-0e6c62c3020a82cee', state 'Running', type 'm5.xlarge', and '2/2 checks passed'. The main content area shows the 'Instance summary' for 'i-0e6c62c3020a82cee (FW-us-east-1c-PANW-f975)'. It includes details such as Instance ID, Instance state (Running), Instance type (m5.xlarge), Public IPv4 address (none), Public IPv4 DNS (none), Elastic IP addresses (52.7.218.8), IAM Role (iam-role-PANW-f975), Private IPv4 addresses (10.10.0.28, 10.10.0.100), Private IPv4 DNS (ip-10-10-0-100.ec2.internal), VPC ID (vpc-006987452e8ad629a), and Subnet ID (subnet-06d1664ebbe18e6e0).

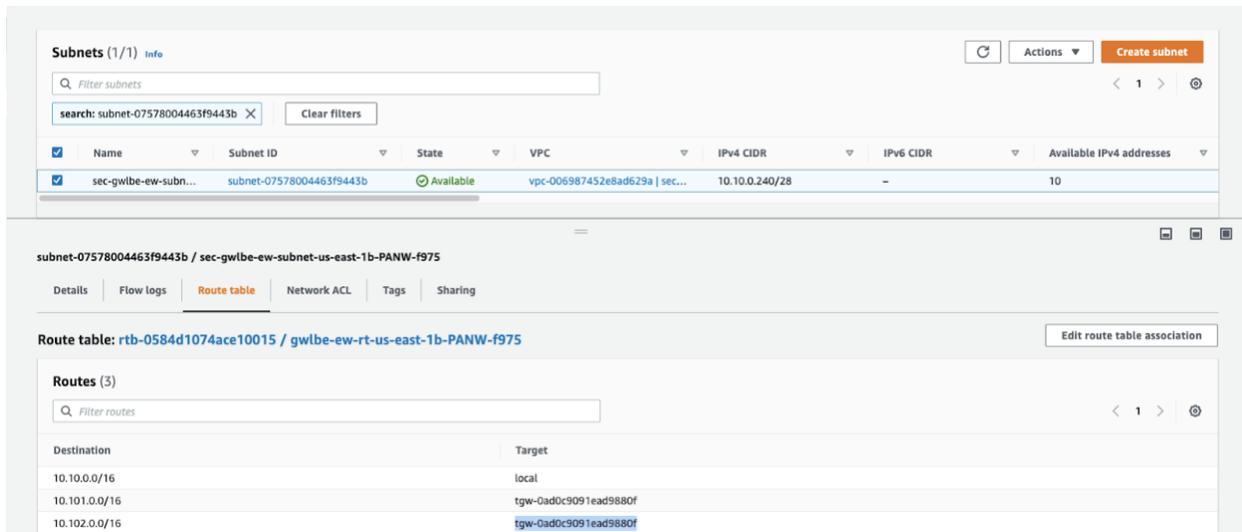
Response Step 5 - Return to the GWLB Endpoint

The permitted request is returned to the GWLB via the GENEVE tunnel and then back to the endpoint. Recall that the ID of the endpoint is vpce-03223b31fd790492. If we take a closer look at that endpoint, we can determine the subnet that it resides in:

The screenshot shows the AWS Management Console interface for a specific endpoint. At the top, there's a header for 'Endpoint ID : vpce-03223b31fd790492' with a search bar and 'Add filter' button. Below this is a table with one endpoint listed: 'vpce-03223b31fd790492' with VPC ID 'vpc-006987452e8ad629a', Service name 'com.amazonaws.vpce.us-east-1.vpce-svc-04c6cedd157b95a6d', Endpoint type 'GatewayLoadBala...', and Status 'available'. The main content area shows the 'Subnets' tab for the endpoint 'vpce-03223b31fd790492'. It includes a 'Manage Subnets' button and a table with the following data:

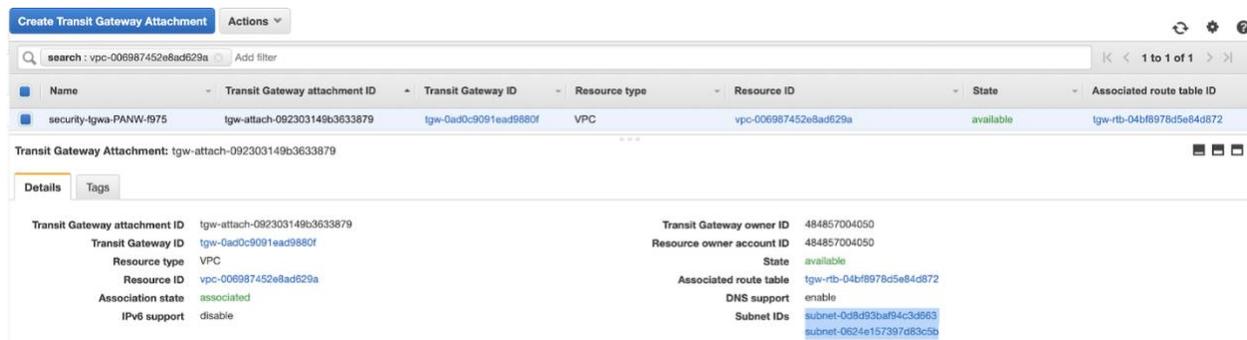
Subnet ID	Availability Zone	IPv4 Addresses	IPv6 Addresses	Network Interface ID	Outpost ID
subnet-07578004463f9443b	us-east-1b (use1-az1)	10.10.0.252	-	eni-03fae688ac27a4185	-

The subnet route table points has the next hop to the destination as the TGW:



Response Step 6 - Return to the TGW

The TGW is connected to the VPC at the subnet level via a Transit Gateway Attachment. To see this association, we navigate to the Transit Gateway Attachment list in the VPC section of the GUI and filter on the security VPC (vpc-006987452e8ad629a in this example):



The TGW attachment and the endpoint are both on the same subnet (same AZ).

Recall that routing within the TGW is handled via route tables associated with the TGW attachment. In the above picture, we can see that the route table associated with the TGW attachment is tgw-rtb-04bf8978d5e84d872. Clicking on the link to the route table and inspecting the routes, we can see that the route to the requester subnet (10.102.0.0/16) points to another attachment (tgw-attach-07e43a9c4496319b1):

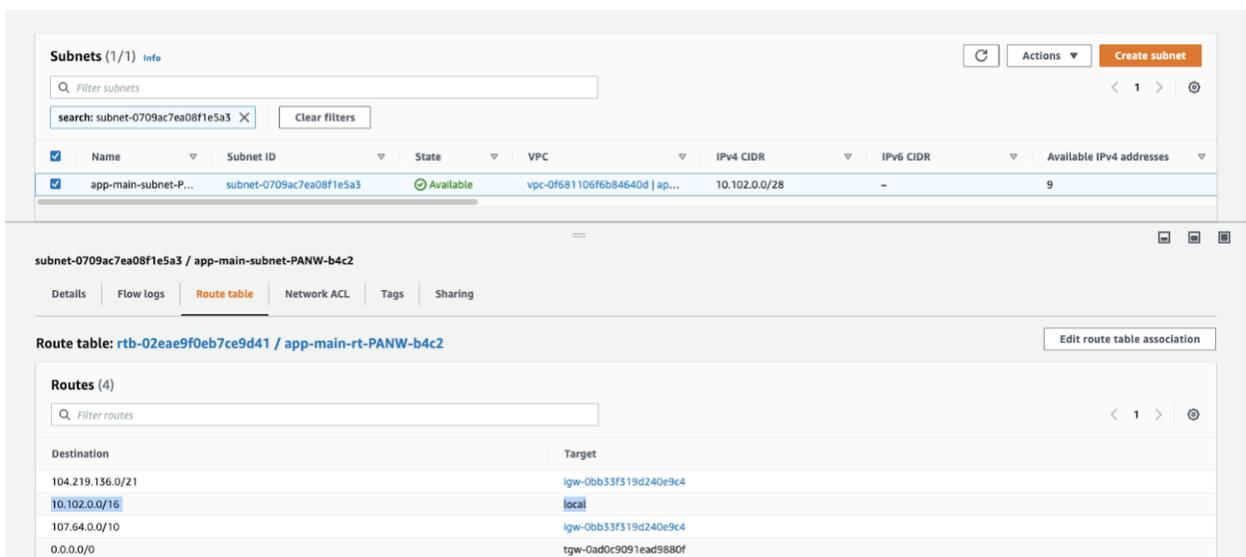


Following this rabbit a little further down the hole, we find that the attachment is associated with a single subnet. Traffic exiting the TGW gets dropped off into this subnet.



Response Step 7 - At Last

Inspection of the subnet route table reveals that any traffic destined for the network is delivered locally:



Inspection of the target host reveals that it resides on the destination network. This tells us that the traffic exiting the TGW is delivered directly to the target.

The screenshot shows the AWS Management Console interface for an EC2 instance named 'app-PANW-b4c2'. The instance is in a 'Running' state. Key details include:

- Instance ID:** i-049eb3bcd0d6951f
- Instance type:** t2.micro
- Status check:** 2/2 checks passed
- Public IPv4 address:** 3.208.253.71
- Private IPv4 address:** 10.102.0.5
- VPC ID:** vpc-0f681106f6b84640d
- Subnet ID:** subnet-0709ac7ea08f1e5a3

Et voilà:

```

ubuntu@admin-appliance: ~ (com.docker.cli)
ubuntu@ip-10-102-0-5:~$ ssh
ssh 10.101.0.4
ubuntu@10.101.0.4's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1037-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Mon Feb 15 22:02:05 UTC 2021

System load:  0.0          Processes:      103
Usage of /:   20.1% of 7.69GB   Users logged in:  0
Memory usage: 20%          IP address for eth0: 10.101.0.4
Swap usage:   0%

 * Introducing self-healing high availability clusters in MicroK8s.
   Simple, hardened, Kubernetes for production, from RaspberryPi to DC.

   https://microk8s.io/high-availability

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
   https://ubuntu.com/livepatch

1 package can be updated.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

New release '20.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Feb 15 22:01:05 2021 from 104.219.139.193
ubuntu@ip-10-101-0-4:~$
    
```

Looking at the FW logs, we can see both original source and original destination:

Q (addr in 10.101.0/16) and (app eq ssh)

	GENERATE TIME	TYPE	FROM ZONE	TO ZONE	SOURCE	DESTINATI...	SOURCE USER	NAT APPLIED	NAT SOURCE IP	NAT DEST IP	TO PORT	APPLICATI...	ACTION	RULE	SESSION END REASON	BYTES
	02/15 22:01:55	start	Trust	Trust	10.102.0.5	10.101.0.4		no			22	ssh	allow	Allowed-traffic	n/a	321
	02/15 19:19:27	end	Trust	Trust	10.102.0.5	10.101.0.4		no			22	ssh	allow	Allowed-traffic	tcp-fin	5.0k
	02/15 19:19:09	start	Trust	Trust	10.102.0.5	10.101.0.4		no			22	ssh	allow	Allowed-traffic	n/a	321