Networking & Content Delivery

Traffic management with AWS Global Accelerator

by Tino Tran | on 29 MAR 2019 | in AWS Global Accelerator, Networking & Content Delivery | Permalink | Share

As customers migrate a growing number of critical workloads to AWS, they have requested more capabilities when they deploy applications across multiple Regions. Critical workloads, such as dynamic API delivery, gaming, and video/voice over IP require higher levels of availability and performance through the use of multi-Region architectures. This demand brings several challenges you must consider, from planning for data consistency to determining how to manage deployments. Of these challenges, one of the critical decisions is determining how to distribute and manage traffic across multiple Regions.

As a Edge architect who has designed multiple solutions for global distributed end users, I can assure you that managing traffic across multiple software stacks is not a rudimentary exercise. Use cases such as A/B testing, global load balancing of traffic, and failover require special attention to detail.   This is why we've repeatedly heard from our customers that they need a managed solution to help simplify these use cases for them. To learn more about edge networking with AWS click here.

AWS Global Accelerator to the rescue

With the launch of AWS Global Accelerator, we can now help you simplify the way traffic is routed across your applications in multiple AWS Regions. Global Accelerator provides several key features to help you distribute your traffic while improving the availability and performance of your application.

Static Anycast IPv4 addresses – When you deploy an accelerator, you are provided with two IPv4 static addresses that can be used as the interface to your application. Each IP address is served from edge locations containing network zones, each zone has infrastructure that is isolated from the infrastructure in the other zones, to provide redundancy. This allows client applications to retry if one IP address become unavailable due to network disruptions. With static IP addresses, you can now make changes to application infrastructure behind the scenes without having to update DNS records or your client applications. In addition to this, we've seen customers use these IP addresses for a number of use cases, such as whitelisting IP addresses where only certain IP addresses are allowed for egress traffic from their company networks. We've also seen customers use IP addresses to zero-rate traffic for applications that don't charge their customers for requests made to specific services.

AWS global network and performance routing – When a request is made to a Global Accelerator IP address, it enters the AWS network through an edge location close to the end user before being routed to the the most optimal AWS Region over the reliable AWS global network.   AWS provides a congestion-free global network to direct TCP or UDP internet traffic from your users to your applications running on Network Load Balancers, Application Load Balancers, and Elastic IP addresses in AWS Regions. This eliminates the inconsistency of routing traffic over the public internet, which can improve performance for latency-sensitive applications such as APIs and video/voice over IP use cases.

Global traffic dials and Regional endpoint weights – When Global Accelerator is provisioned in front of multiple AWS Regions, by default requests are routed to the Region closest to your end users to provide the most optimal performance. To adjust how the traffic is routed, Global Accelerator provides traffic dials to shift traffic between the Regions configured behind your accelerator.   These traffic dials come in handy when you need to redirect traffic from a Region that has exceeded its capacity, take a Region out of service for maintenance, or gradually ramp up traffic for a newly added Region. For even finer grained control of Regions containing multiple endpoints, you can use Global Accelerator's endpoint weights to balance traffic across Network Load Balancers, Application Load Balancers, or Elastic IPs. Weighted endpoints can help you roll out new application changes when performing A/B testing or blue/green deployments.

How it all works

To understand how Global Accelerator works, let's first review a few of the key terms used when deploying an accelerator:

Accelerator – When you deploy Global Accelerator, you create a configuration for deployment known as an accelerator.

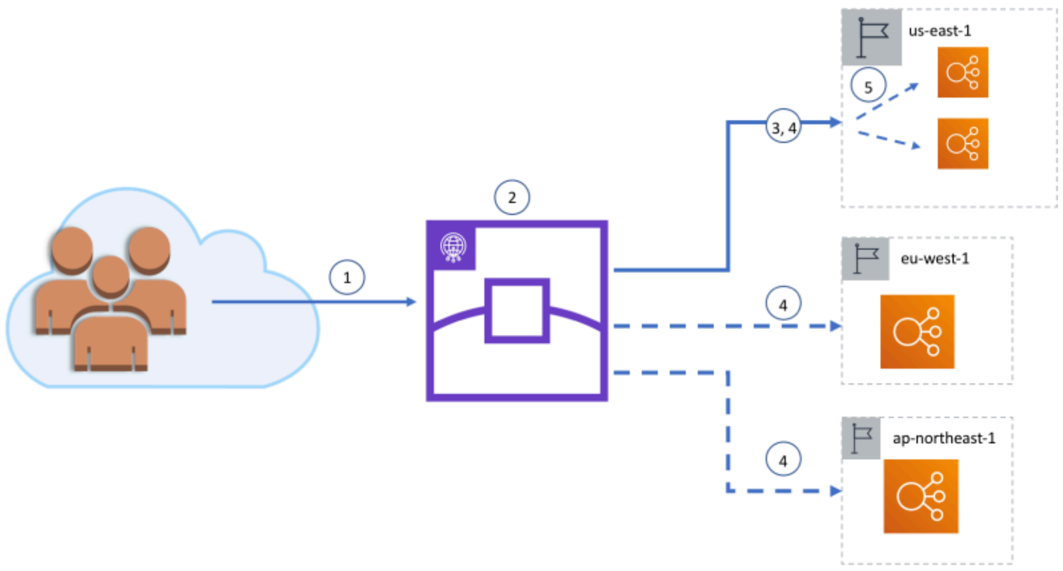
Listener – When you configure an accelerator, you must configure at least one listener to process inbound connections from clients to the accelerator, based on the protocol and port.

Endpoint Group – Endpoint groups define the different AWS Regions where your application is deployed.

Traffic Dial – Configured on an endpoint group to dictate the percentage of traffic an endpoint group can receive.

Endpoint – An endpoint is an Elastic IP address, Network Load Balancer, or Application Load Balancer that is associated with an endpoint group.

Endpoint Weight – This is configured on the endpoint to determine how requests are distributed among endpoint groups with multiple endpoints.

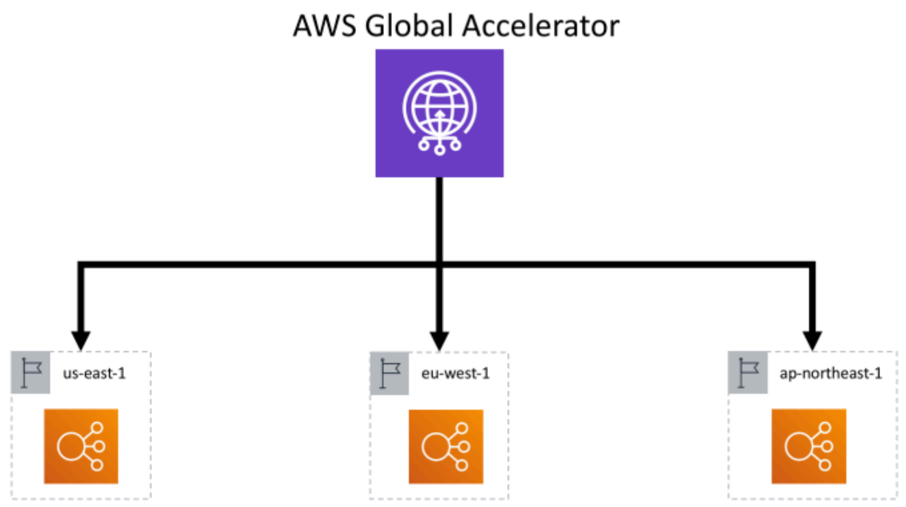


Behind the scenes

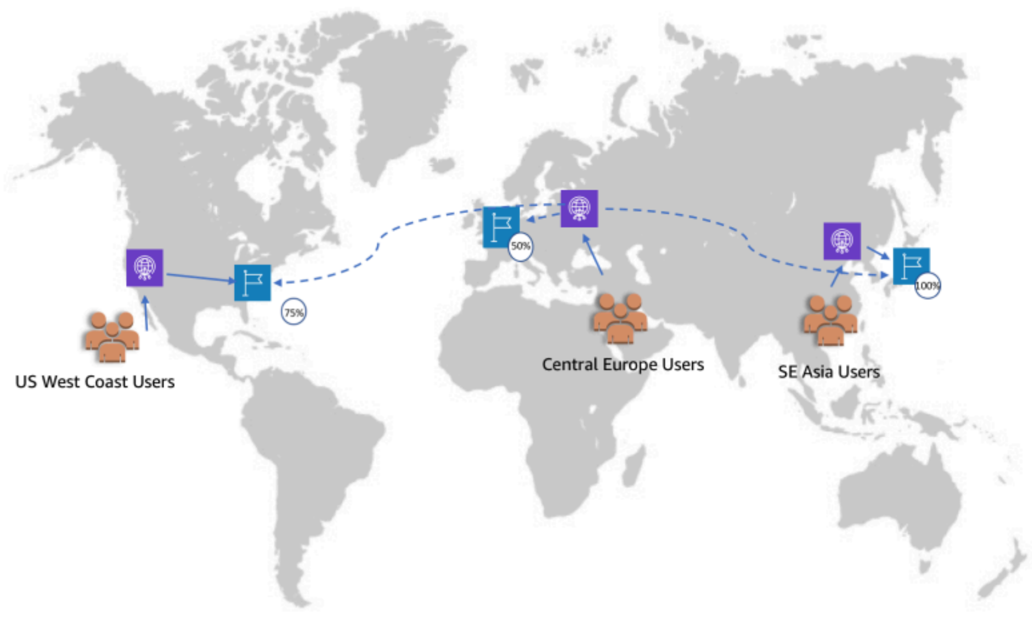
1. When a request is made to an accelerator static IP address, the request is first routed to a nearby Global Accelerator edge location over the public internet via the Anycast BGP protocol.
2. The accelerator accepts the request if there is a listener configured that matches the protocol and port, then determines the most optimal endpoint group based on:
    1. Geographic proximity to the edge location.
    2. Traffic dial settings
    3. Health of the endpoints in the endpoint group. **Note:** Global Accelerator constantly checks the health of endpoints configured under an endpoint group to identify if an endpoint is healthy.
3. If the endpoint group closest to the edge location has the traffic dial configured to 100 percent and the endpoints in the Region are passing health checks, the request is forwarded over the AWS global network.
4. If the endpoint group closest to the Edge location has the traffic dial configured to less than 100 percent, the configured percentage of requests received by the edge location is sent to the closest endpoint group, and the remaining requests are distributed to other endpoint groups weighted by geographic proximity and the traffic dials settings. In all cases, endpoint groups must have healthy endpoints to receive requests.
5. For endpoint groups with multiple endpoints, Global Accelerator spreads the traffic across the endpoints using a 5 tuple hash based on protocol, source IP address, destination IP address, source port, and destination port. If endpoint weights are configured, Global Accelerator sends traffic to an endpoint based on the weight that you assign to it as a proportion of the total weight for all endpoints in the group. **Note**: Global Accelerator also provides client affinity capabilities for stateful application use cases.

Let’s see it in action

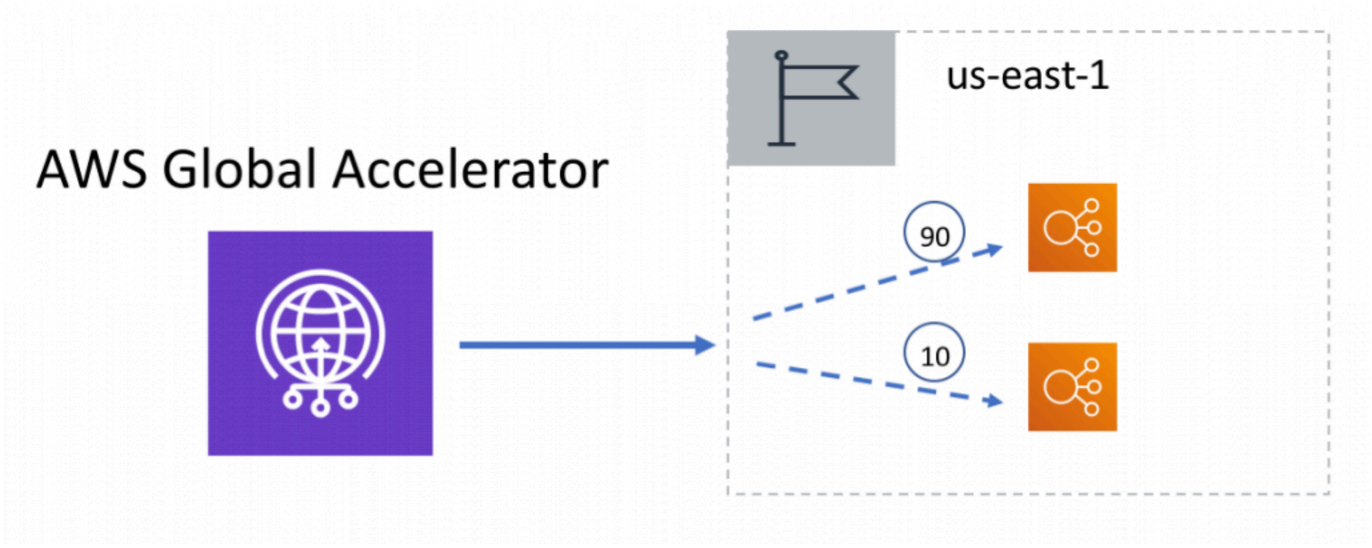
In this blog post, I’ll show you how to use these features to distribute traffic across multiple AWS Regions and across endpoints within an AWS Region. To demonstrate this, I will deploy a Global Accelerator in front of redundant applications fronted by AWS Application Load Balancer in three different AWS Regions, as shown in the following diagram.



Then I'll use the Global Accelerator traffic dials to shift traffic between the Regions. In this example, I'll show you how to send half of Europe's normal traffic to the North America and Asia Regions.



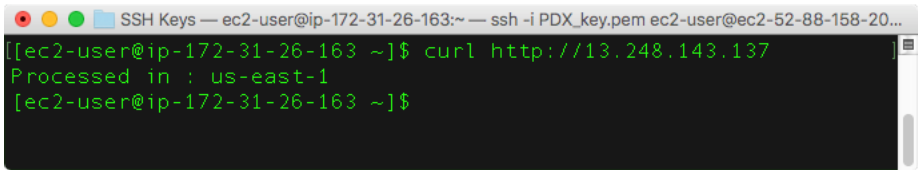
Finally, I'll show you how to add a new endpoint within an existing Region and gradually shift traffic from the existing endpoint over to the new endpoint.



Initial setup

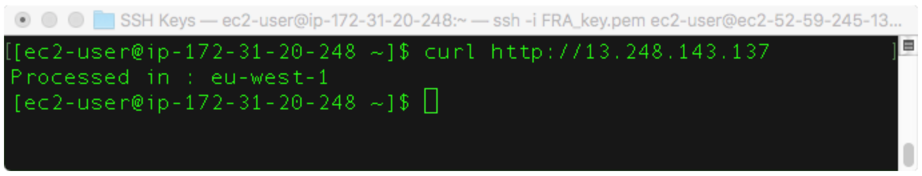
To demonstrate how traffic dials work, I've deployed an HTTP application using an AWS Application Load Balancer backed by Amazon EC2 in 3 Regions across the North America, Europe, and Asia. For each Region, the application simply returns an HTTP response with a text body describing the Region that serviced the request.

US-EAST-1



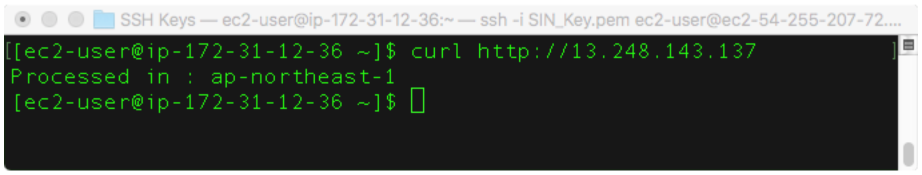
```
SSH Keys — ec2-user@ip-172-31-26-163:~ — ssh -i PDX_key.pem ec2-user@ec2-52-88-158-20...
[[ec2-user@ip-172-31-26-163 ~]$ curl http://13.248.143.137
Processed in : us-east-1
[ec2-user@ip-172-31-26-163 ~]$
```

EU-WEST-1



```
SSH Keys — ec2-user@ip-172-31-20-248:~ — ssh -i FRA_key.pem ec2-user@ec2-52-59-245-13...
[[ec2-user@ip-172-31-20-248 ~]$ curl http://13.248.143.137
Processed in : eu-west-1
[ec2-user@ip-172-31-20-248 ~]$
```

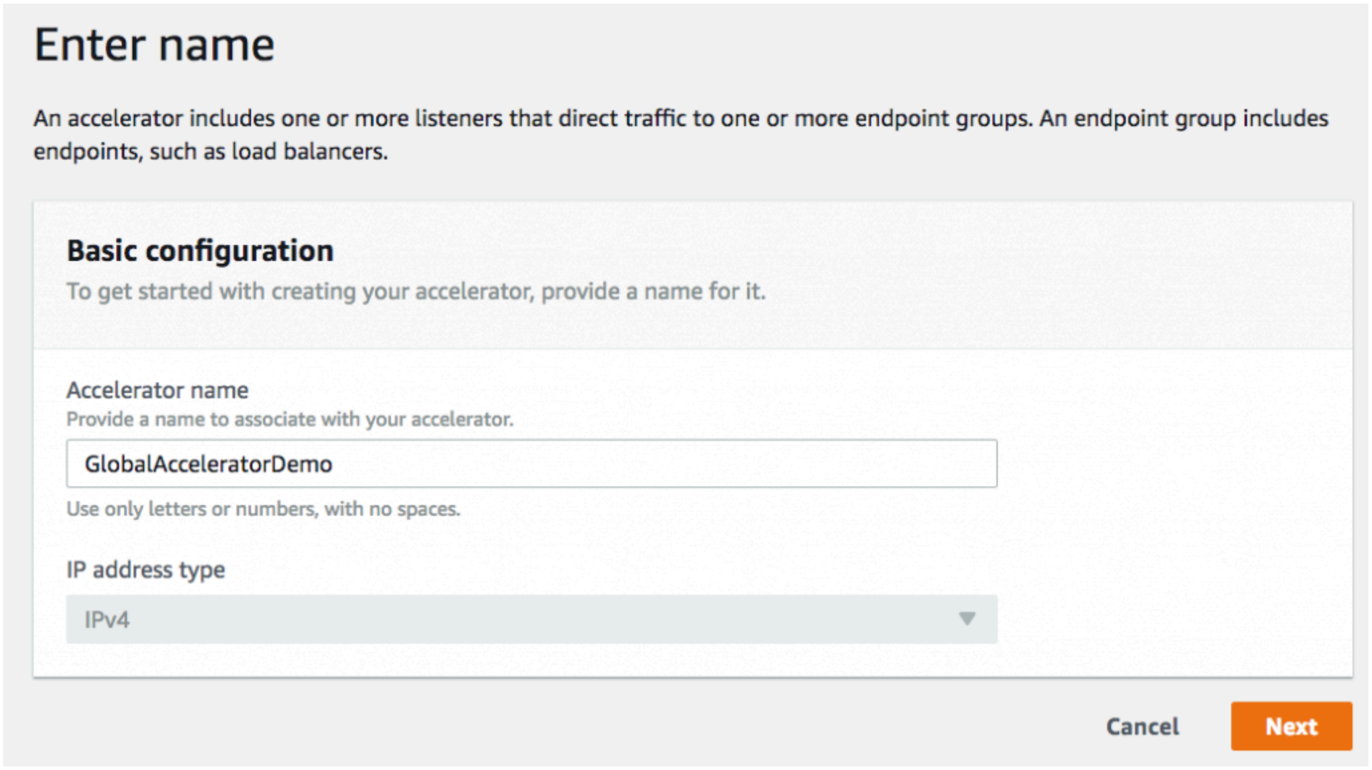
AP-NORTHEAST-1



```
SSH Keys — ec2-user@ip-172-31-12-36:~ — ssh -i SIN_Key.pem ec2-user@ec2-54-255-207-72....
[[ec2-user@ip-172-31-12-36 ~]$ curl http://13.248.143.137
Processed in : ap-northeast-1
[ec2-user@ip-172-31-12-36 ~]$
```

I've then created an Accelerator in front of the three Regions using the following steps:

1. In the AWS Global Accelerator console, choose **Create an Accelerator.**
2. Provide a name (such as GlobalAcceleratorDemo) for the accelerator, then choose **Next**.



Enter name

An accelerator includes one or more listeners that direct traffic to one or more endpoint groups. An endpoint group includes endpoints, such as load balancers.

Basic configuration

To get started with creating your accelerator, provide a name for it.

Accelerator name

Provide a name to associate with your accelerator.

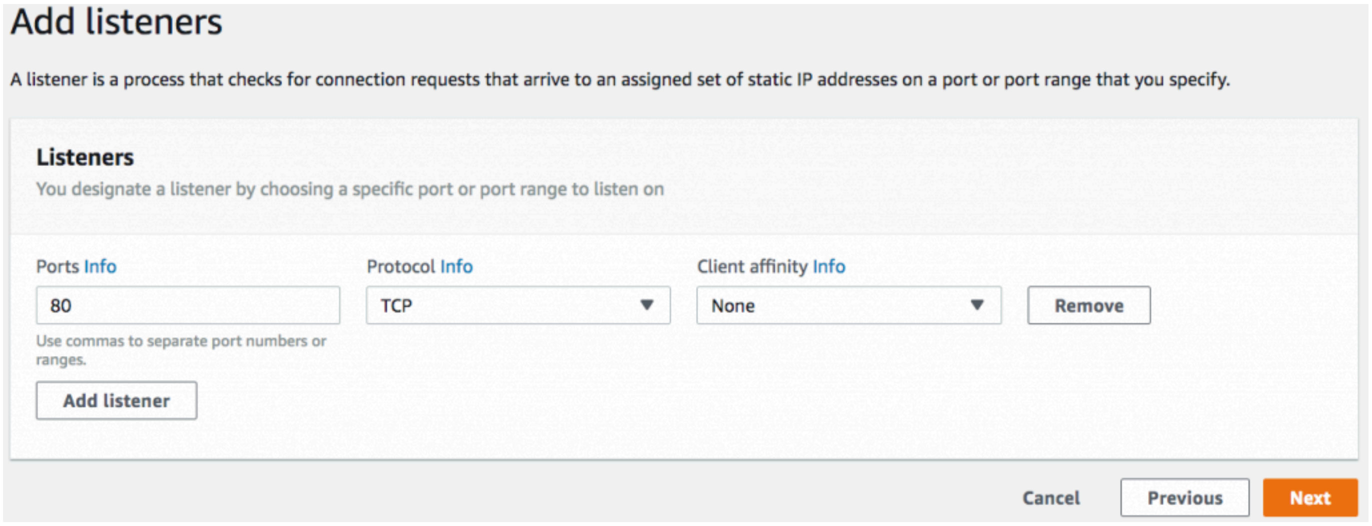
GlobalAcceleratorDemo

Use only letters or numbers, with no spaces.

IP address type

IPv4

3. Add a TCP listener on port 80, then choose **Next**. We are serving HTTP traffic, so port 80 is a logical choice.



Add listeners

A listener is a process that checks for connection requests that arrive to an assigned set of static IP addresses on a port or port range that you specify.

Listeners

You designate a listener by choosing a specific port or port range to listen on

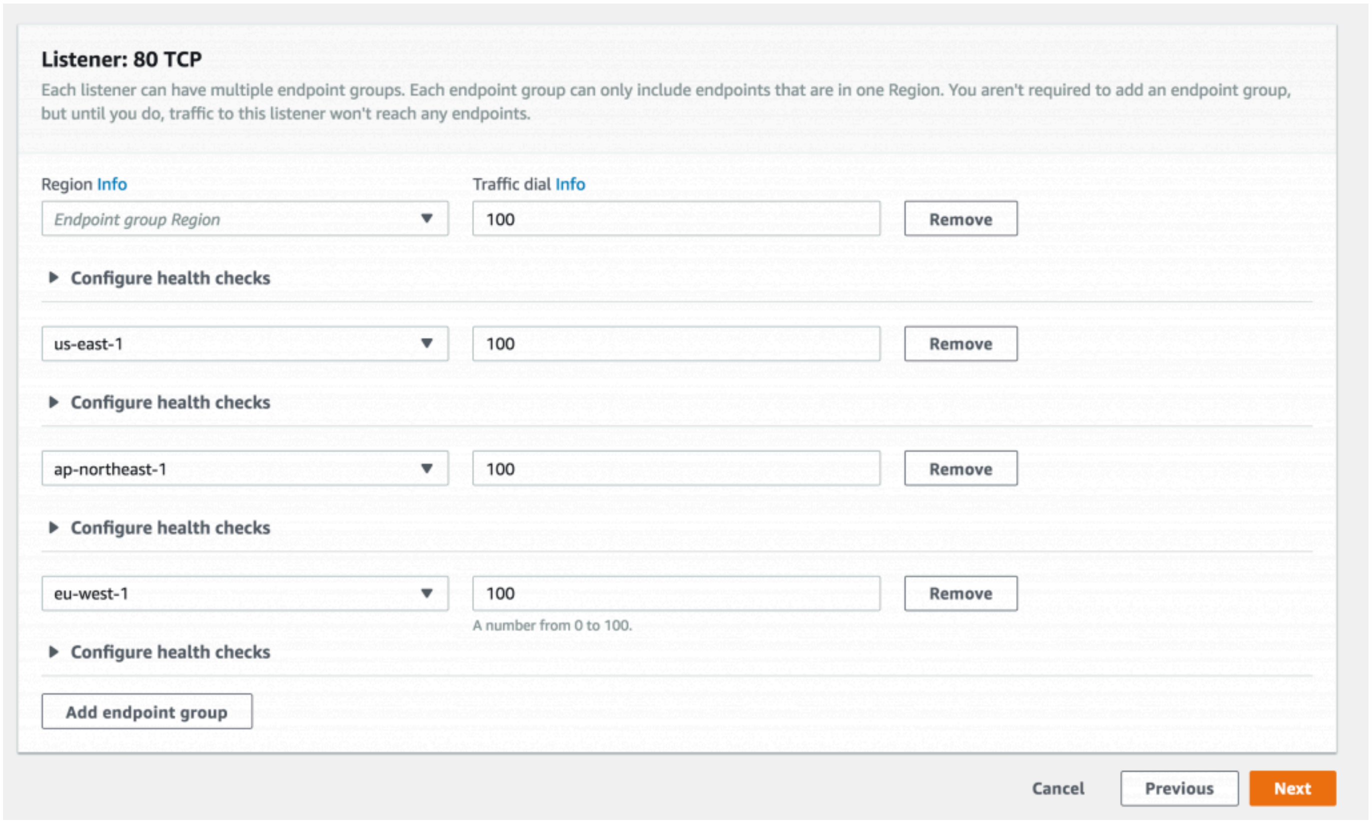
Ports Info: 80

Use commas to separate port numbers or ranges.

Protocol Info: TCP

Client affinity Info: None

4. Add an endpoint group for each Region leaving the **traffic dials** set to 100%, and then choose **Next**.



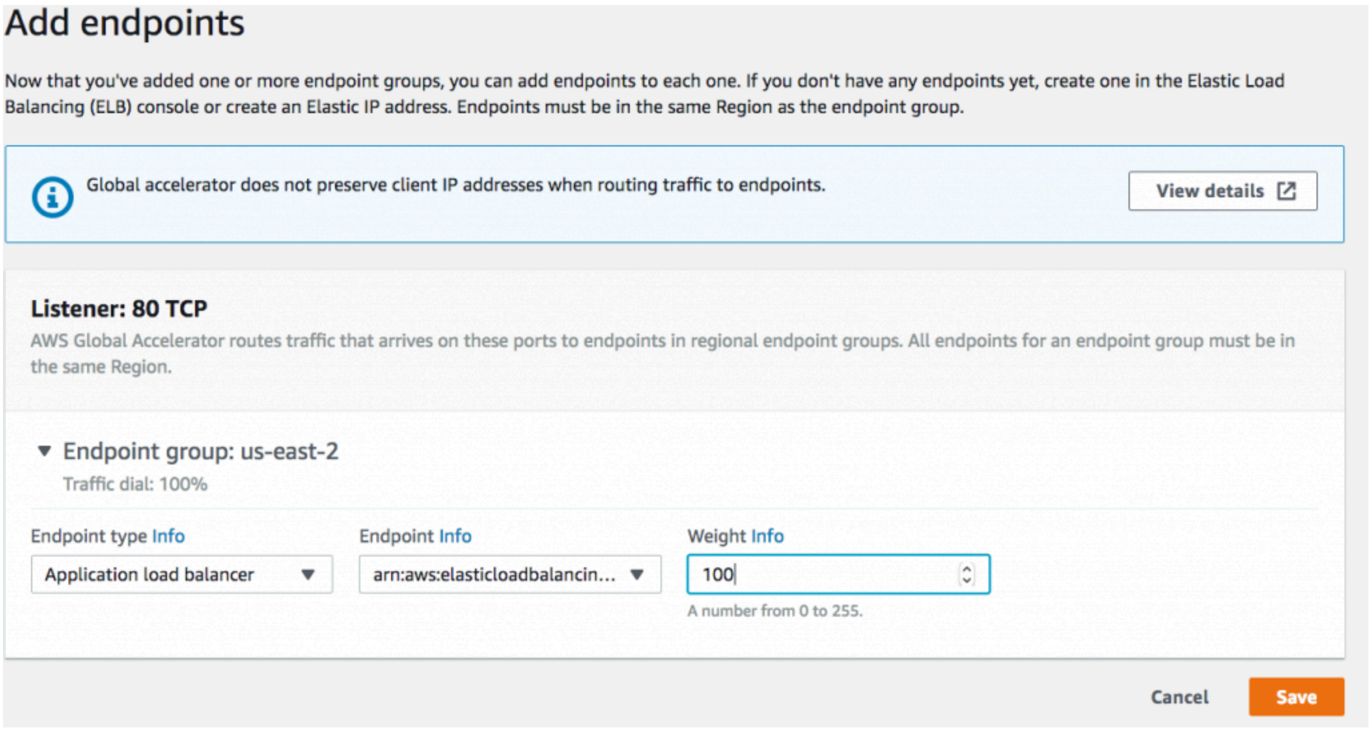
Listener: 80 TCP

Each listener can have multiple endpoint groups. Each endpoint group can only include endpoints that are in one Region. You aren't required to add an endpoint group, but until you do, traffic to this listener won't reach any endpoints.

Region Info	Traffic dial Info	
Endpoint group Region	100	Remove
us-east-1	100	Remove
ap-northeast-1	100	Remove
eu-west-1	100	Remove

A number from 0 to 100.

5. Choose **Create Accelerator**.
6. Select the newly created accelerator from the list, then select the listener.
7. For each endpoint group (repeat 3x), add the Application Load Balancer endpoint.



Add endpoints

Now that you've added one or more endpoint groups, you can add endpoints to each one. If you don't have any endpoints yet, create one in the Elastic Load Balancing (ELB) console or create an Elastic IP address. Endpoints must be in the same Region as the endpoint group.

Global accelerator does not preserve client IP addresses when routing traffic to endpoints.

Listener: 80 TCP

AWS Global Accelerator routes traffic that arrives on these ports to endpoints in regional endpoint groups. All endpoints for an endpoint group must be in the same Region.

Endpoint group: us-east-2

Traffic dial: 100%

Endpoint type Info	Endpoint Info	Weight Info
Application load balancer	arn:aws:elasticloadbalancin...	100

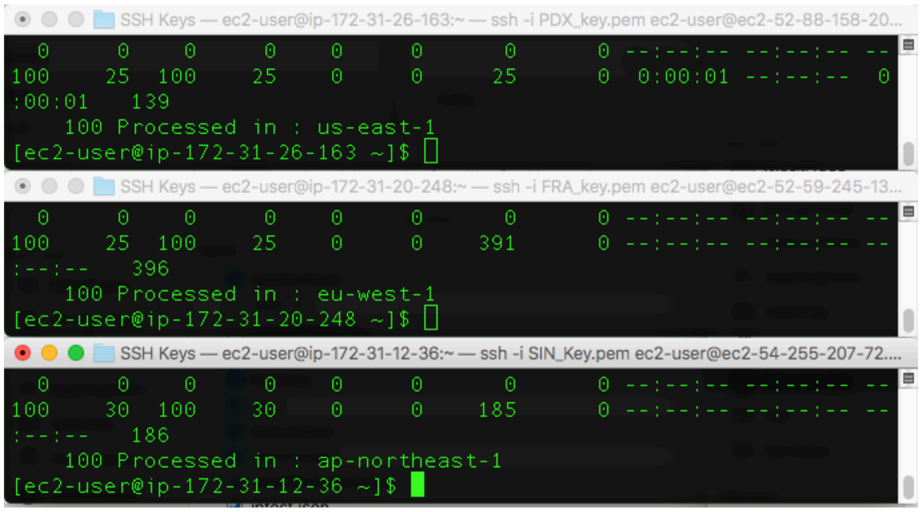
A number from 0 to 255.

Testing traffic dials

To simulate clients from different geographic locations, I've launched an Amazon EC2 instance in 3 AWS Regions (us-east-1, eu-central-1, ap-northeast-1) that are nearby each endpoint group configured. From each of the instances, I can execute curl against one of the static IP addresses of the accelerator. For this exercise, I'll execute the following command from each Linux EC2 instance to call one of the static IP addresses provided by the accelerator 100 times and output a count of where each request was processed.

NOTE: Testing Global Accelerator from within different AWS Regions is useful for testing how clients from different geographies will be routed. However, it won't demonstrate the performance benefits of using Global Accelerator for production workloads with clients from the public internet because each AWS Region is already connected via the AWS global network.

```bash
for ((i=0;i<100;i++)); do  curl http://13.248.143.137/ >> output.txt; done; cat output.txt | sort | uniq -c ; rm output.txt;
```



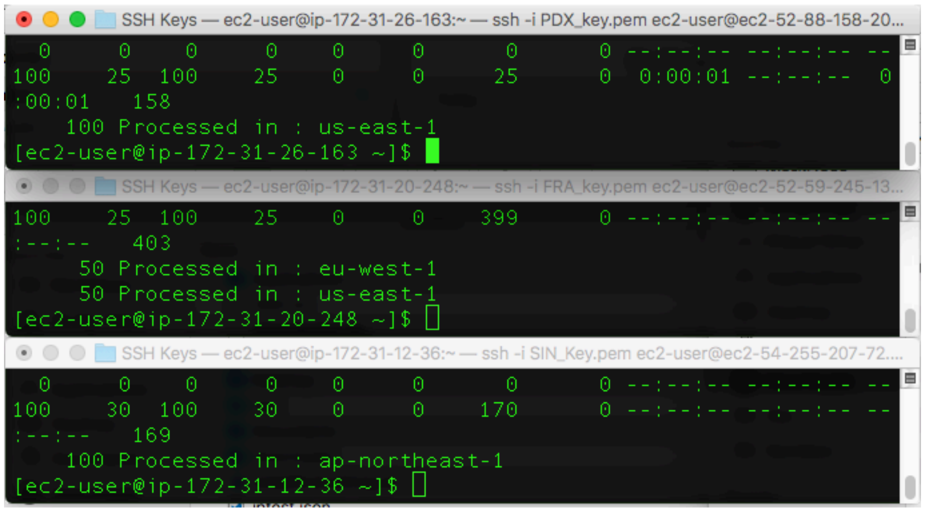
```
SSH Keys — ec2-user@ip-172-31-26-163:~ — ssh -i PDX_key.pem ec2-user@ec2-52-88-158-20...
  0      0     0      0     0      0      0       0 --:--:-- --:--:-- --
100     25   100     25     0      0     25       0  0:00:01 --:--:--   0
:00:01    139
    100 Processed in : us-east-1
[ec2-user@ip-172-31-26-163 ~]$ 

SSH Keys — ec2-user@ip-172-31-20-248:~ — ssh -i FRA_key.pem ec2-user@ec2-52-59-245-13...
  0      0     0      0     0      0      0       0 --:--:-- --:--:-- --
100     25   100     25     0      0    391       0 --:--:-- --:--:-- --
:--:--    396
    100 Processed in : eu-west-1
[ec2-user@ip-172-31-20-248 ~]$ 

SSH Keys — ec2-user@ip-172-31-12-36:~ — ssh -i SIN_Key.pem ec2-user@ec2-54-255-207-72....
  0      0     0      0     0      0      0       0 --:--:-- --:--:-- --
100     30   100     30     0      0    185       0 --:--:-- --:--:-- --
:--:--    186
    100 Processed in : ap-northeast-1
[ec2-user@ip-172-31-12-36 ~]$ 
```

Notice that with the traffic dials currently set to 100 percent, executing the command from EC2 in each of our test Regions resulted in all 100 requests being served from the nearby Region with a configured Global Accelerator endpoint group.

Now, let's pretend that the endpoint in eu-west-1 needed to go into maintenance mode and is operating at reduced capacity. To direct half of the usual traffic away from this endpoint group, I updated the traffic dial for us-east-1 to 50 percent. Configuration changes should take place within a couple minute on average. Running the same curl loop again from all Regions, I get the following results:



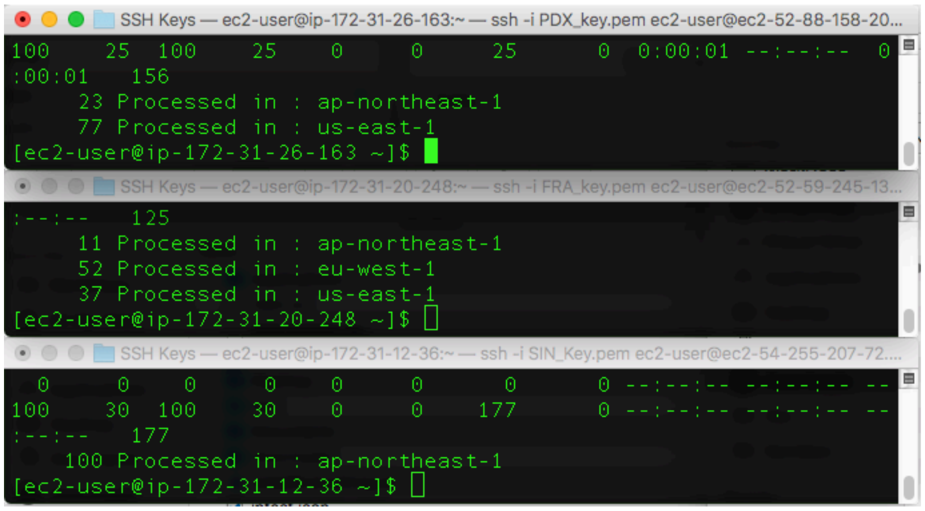
```
SSH Keys — ec2-user@ip-172-31-26-163:~ — ssh -i PDX_key.pem ec2-user@ec2-52-88-158-20...
  0      0     0      0     0      0      0       0 --:--:-- --:--:-- --
100     25   100     25     0      0     25       0  0:00:01 --:--:--   0
:00:01    158
    100 Processed in : us-east-1
[ec2-user@ip-172-31-26-163 ~]$ 

SSH Keys — ec2-user@ip-172-31-20-248:~ — ssh -i FRA_key.pem ec2-user@ec2-52-59-245-13...
100     25   100     25     0      0    399       0 --:--:-- --:--:-- --
:--:--    403
     50 Processed in : eu-west-1
     50 Processed in : us-east-1
[ec2-user@ip-172-31-20-248 ~]$ 

SSH Keys — ec2-user@ip-172-31-12-36:~ — ssh -i SIN_Key.pem ec2-user@ec2-54-255-207-72....
  0      0     0      0     0      0      0       0 --:--:-- --:--:-- --
100     30   100     30     0      0    170       0 --:--:-- --:--:-- --
:--:--    169
    100 Processed in : ap-northeast-1
[ec2-user@ip-172-31-12-36 ~]$ 
```

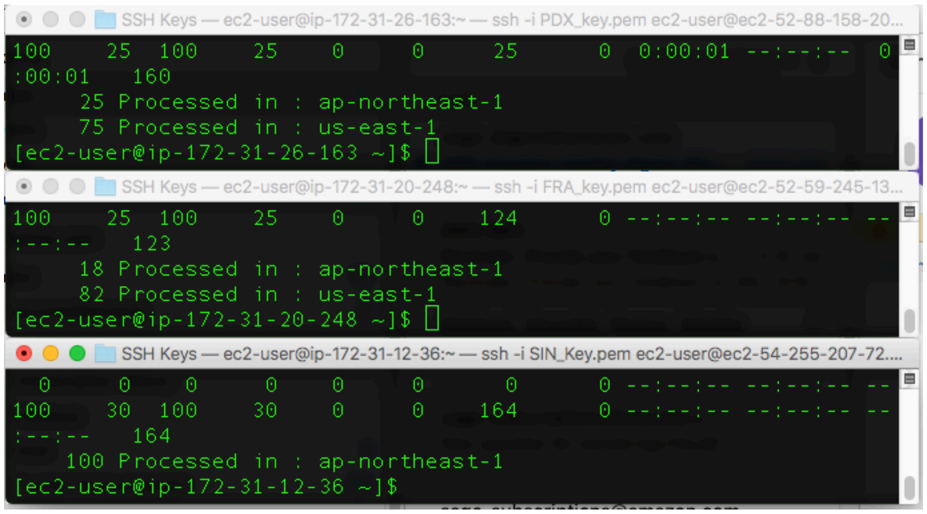
For the requests originating from Frankfurt, I now see that 50 percent of them are served from eu-west-1 and the other 50 percent are now served out of us-east-1 (the next closest Region). All requests originating from us-east-1 and ap-northeast-1 both continue to be served from their immediate Regions.

Next, to reduce some of the additional load on us-east-1, I can turn down the traffic dial for that endpoint group to 75 percent and re-run the the same test.


```
SSH Keys — ec2-user@ip-172-31-26-163:~ — ssh -i PDX_key.pem ec2-user@ec2-52-88-158-20...
100    25  100    25    0     0     25      0  0:00:01 --:--:--  0
:00:01   156
     23 Processed in : ap-northeast-1
     77 Processed in : us-east-1
[ec2-user@ip-172-31-26-163 ~]$
SSH Keys — ec2-user@ip-172-31-20-248:~ — ssh -i FRA_key.pem ec2-user@ec2-52-59-245-13...
:--:--   125
     11 Processed in : ap-northeast-1
     52 Processed in : eu-west-1
     37 Processed in : us-east-1
[ec2-user@ip-172-31-20-248 ~]$
SSH Keys — ec2-user@ip-172-31-12-36:~ — ssh -i SIN_Key.pem ec2-user@ec2-54-255-207-72...
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --
100    30  100    30    0     0    177      0 --:--:-- --:--:-- --
:--:--   177
    100 Processed in : ap-northeast-1
[ec2-user@ip-172-31-12-36 ~]$
```

For the requests originating from Oregon, I can now see that 77 requests were served out of us-east-1 (approximately 75 percent), 23 requests are served out of ap-northeast-1 (the next closest Region). Requests from Frankfurt are now divided among 3 Regions: 11 from ap-northeast-1, 37 from us-east-1, and 52 from eu-west-1. Lastly, all requests from Singapore continue to be served from ap-northeast-1. Notice that traffic from Frankfurt was served from three Regions because the traffic dials for the primary and secondary preferred Regions were configured with traffic dials set to less than 100 percent.

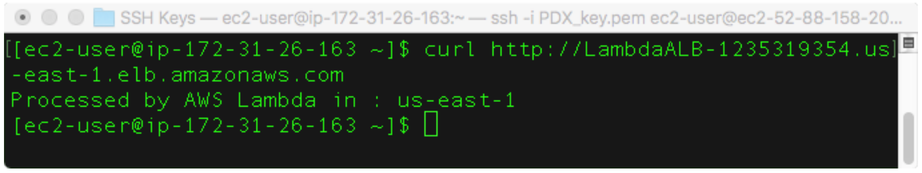
For the last test, let's pretend that eu-west-1 needs to be taken completely out of service and set the traffic dial to 0 percent.


```
SSH Keys — ec2-user@ip-172-31-26-163:~ — ssh -i PDX_key.pem ec2-user@ec2-52-88-158-20...
100    25  100    25    0     0     25      0  0:00:01 --:--:--  0
:00:01   160
     25 Processed in : ap-northeast-1
     75 Processed in : us-east-1
[ec2-user@ip-172-31-26-163 ~]$
SSH Keys — ec2-user@ip-172-31-20-248:~ — ssh -i FRA_key.pem ec2-user@ec2-52-59-245-13...
100    25  100    25    0     0    124      0 --:--:-- --:--:-- --
:--:--   123
     18 Processed in : ap-northeast-1
     82 Processed in : us-east-1
[ec2-user@ip-172-31-20-248 ~]$
SSH Keys — ec2-user@ip-172-31-12-36:~ — ssh -i SIN_Key.pem ec2-user@ec2-54-255-207-72...
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --
100    30  100    30    0     0    164      0 --:--:-- --:--:-- --
:--:--   164
    100 Processed in : ap-northeast-1
[ec2-user@ip-172-31-12-36 ~]$
```

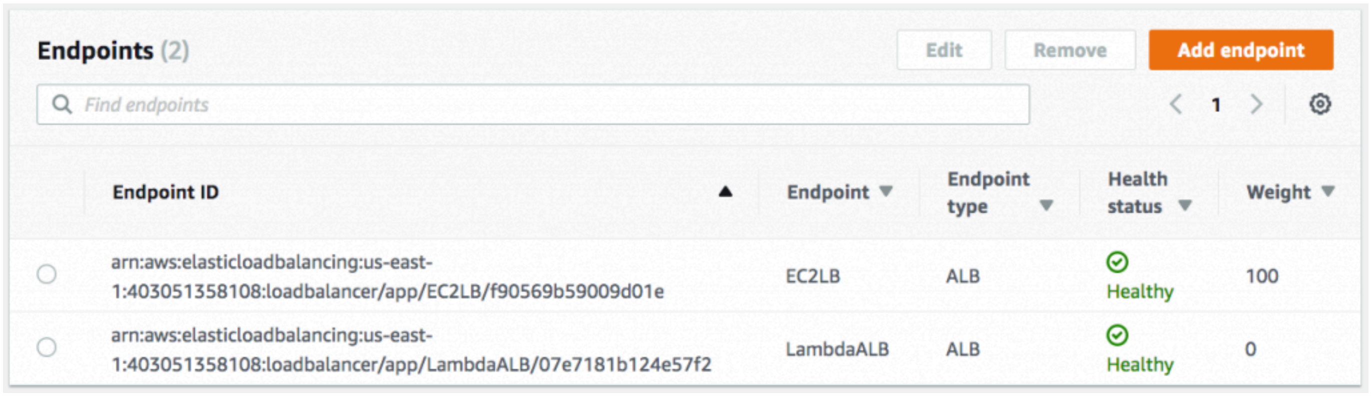
With the traffic dial for us-east-1 left at 75 percent, the requests are now split between ap-northeast-1 and us-east-1 with eu-west-1 receiving no requests. Notice that the traffic is routed independently from each of the Regions, strictly based on the traffic dial configuration of the endpoint group.

Testing endpoint weights

To demonstrate how to migrate traffic to a new version of an application, I'll use the Global Accelerator endpoint weights to add a new serverless endpoint backed by AWS Lambda to us-east-1. In this scenario I'll do a simple A/B test before migrating 100 percent of the traffic. To distinguish between the two endpoints, testing the new endpoint directly returns a different message.

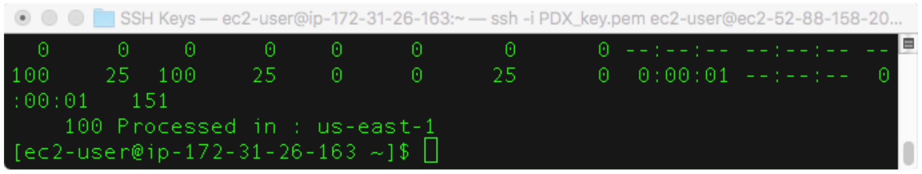

```
SSH Keys — ec2-user@ip-172-31-26-163:~ — ssh -i PDX_key.pem ec2-user@ec2-52-88-158-20...
[[ec2-user@ip-172-31-26-163 ~]$ curl http://LambdaALB-1235319354.us]
-east-1.elb.amazonaws.com
Processed by AWS Lambda in : us-east-1
[ec2-user@ip-172-31-26-163 ~]$
```

To start, I will reset all of the traffic dials to 100 percent and add the new Application Load Balancer endpoint to the us-east-1 endpoint group with a starting weight of zero.



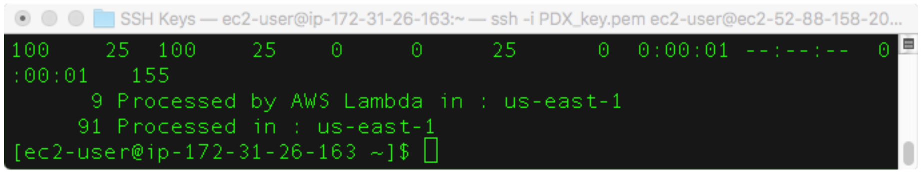
Endpoint ID	Endpoint	Endpoint type	Health status	Weight
arn:aws:elasticloadbalancing:us-east-1:403051358108:loadbalancer/app/EC2LB/f90569b59009d01e	EC2LB	ALB	✓ Healthy	100
arn:aws:elasticloadbalancing:us-east-1:403051358108:loadbalancer/app/LambdaALB/07e7181b124e57f2	LambdaALB	ALB	✓ Healthy	0

With an endpoint weight of zero for the new serverless endpoint, re-running the curl command from Oregon shows that requests are still only served from the previous endpoint in us-east-1.


SSH Keys — ec2-user@ip-172-31-26-163:~ — ssh -i PDX_key.pem ec2-user@ec2-52-88-158-20...

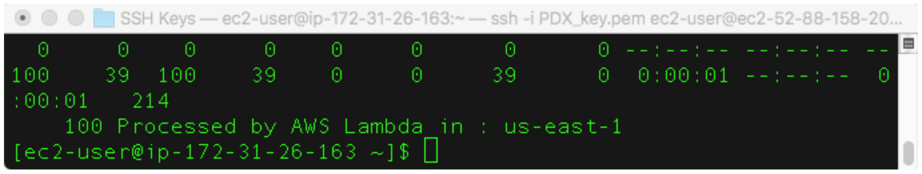
```
  0      0    0      0    0      0       0      0 --:--:-- --:--:-- --
100     25  100     25    0      0      25      0  0:00:01 --:--:--  0
:00:01   151
    100 Processed in : us-east-1
[ec2-user@ip-172-31-26-163 ~]$ 
```

To send some sample traffic to the new endpoint, I'll update the weights to 90 for the EC2 endpoint and 10 for the AWS Lambda endpoint that will distribute 10 percent of North America's traffic to the new endpoint. Then I'll re-run the test.


SSH Keys — ec2-user@ip-172-31-26-163:~ — ssh -i PDX_key.pem ec2-user@ec2-52-88-158-20...

```
100     25  100     25    0      0      25      0  0:00:01 --:--:--  0
:00:01   155
      9 Processed by AWS Lambda in : us-east-1
     91 Processed in : us-east-1
[ec2-user@ip-172-31-26-163 ~]$ 
```

With approximately 10 percent of the traffic from North American clients hitting the new endpoint, I'm now able to verify that my new application changes are behaving as expected before shifting 100 percent of North America's traffic to the new endpoint. To complete the migration, I'll now increase the endpoint weight to 100 and reduce the weight of the old endpoint to 0. Now when I re-run the test, I can see that 100 percent of the traffic from North American clients is now using the serverless endpoint.


SSH Keys — ec2-user@ip-172-31-26-163:~ — ssh -i PDX_key.pem ec2-user@ec2-52-88-158-20...

```
  0      0    0      0    0      0       0      0 --:--:-- --:--:-- --
100     39  100     39    0      0      39      0  0:00:01 --:--:--  0
:00:01   214
    100 Processed by AWS Lambda in : us-east-1
[ec2-user@ip-172-31-26-163 ~]$ 
```

Conclusion

By following this blog post, you can quickly shift traffic between the AWS Regions and Regional endpoints configured behind your Global Accelerator with just a few updates. This demonstrates how Global Accelerator can be a useful tool in any operator's runbook for moving traffic away from Regions under maintenance, reducing load from Regions under load, ramping up traffic to newly added Regions, performing blue-green deployments, and performing A/B testing. Beyond traffic dials and endpoint groups, Global Accelerator also provides many other features such as failover, client affinity, health checking, and DDOS protection. To learn more about all of the Global Accelerator features visit our website.

{ ...} Blog: Using AWS Client VPN to securely access AWS and on-premises resources

Learn about AWS VPN services

Watch re:Invent 2019: Connectivity to AWS and hybrid AWS network architectures

TAGS: AWS Global Accelerator