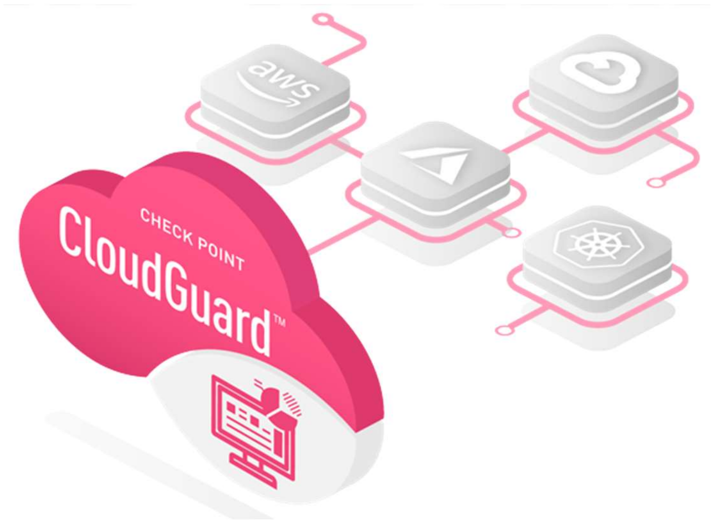
GCP iLB as next hop with Check Point Gateway



Christian Abraham Castillo Porras
Cloud Security Architect - LATAM
November 2020

The iLB from GCP allows you to us a next hop that can redirect packets to the targets of the LB (called backends in GCP).

This configuration is useful if you need a load balancer to serve as a next hop for a default route. When virtual machine (VM) instances in your Virtual Private Cloud (VPC) network send traffic to the internet, the traffic is routed through load-balanced gateway virtual appliances.

Benefits of using your internal TCP/UDP load balancer as a next hop

When the load balancer is a next hop for a static route, you don't need to explicitly configure your clients to send traffic to the load balancer or to each backend VM. You can integrate your backend VMs in a bump-in-the-wire fashion.

Using an internal TCP/UDP load balancer as a next hop for a static route provides the same benefits as Internal TCP/UDP Load Balancing. The load balancer's health check ensures that new connections are routed to healthy backend VMs. By using a managed instance group as a backend, you can configure autoscaling to grow or shrink the set of VMs based on service demand.

Architecture

An internal TCP/UDP load balancer with multiple backend instance groups distributes connections among backend VMs in all of those instance groups. For information about the distribution method and its configuration options, see traffic distribution.

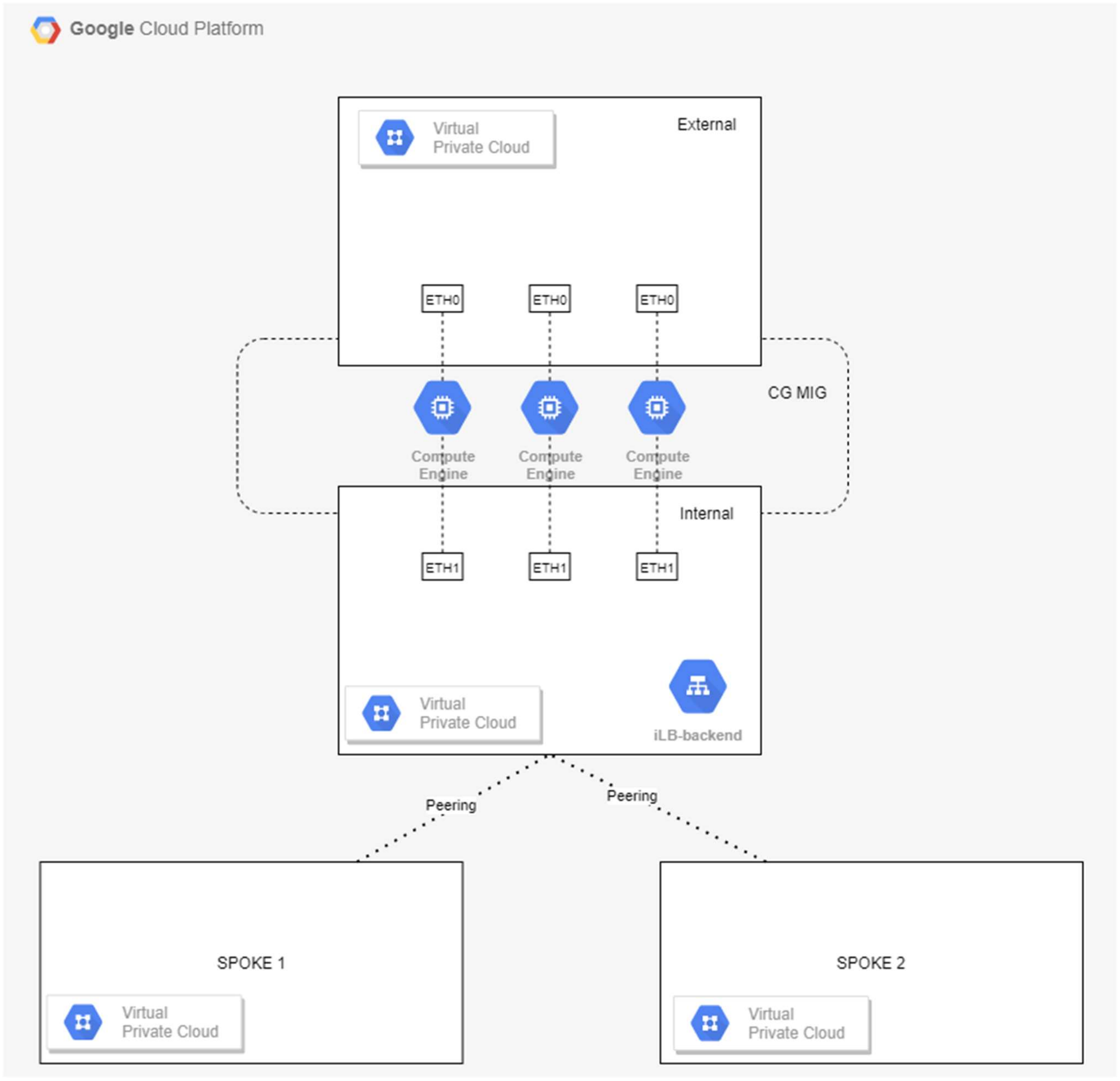
Instances that participate as backend VMs for internal TCP/UDP load balancers must be running the appropriate Linux or Windows guest environment or other processes that provide equivalent functionality. This guest environment must be able to contact the metadata server (`metadata.google.internal`, `169.254.169.254`) to read instance metadata so that it can generate local routes to accept traffic sent to the load balancer's internal IP address.

Regardless of the type of health check that you create, Google Cloud sends health check probes to the IP address of the internal TCP/UDP load balancer's forwarding rule, to the network interface in the VPC selected by the load balancer's backend service. This simulates how load balanced traffic is delivered. Software running on your backend VMs must respond to both load balanced traffic and health check probes sent to the load balancer's IP address. For more information, see Destination for probe packets.

Conclusion

When create an iLB the machine in the backend needs to know how to respond the packets that are sent to the IP Address of the Load Balancer, so the IP address must be on the OS level and the iLB don’t work as proxy and also the Health Checks are sent from a simulated IP address of the LB but need to be responded by the machine running at the backend.

Check Point Cloud Guard IaaS for GCP can be a NAT device and can use the MIG technology to allow the GW to increase the amount of members; this in conjunction with the LB can create an auto scalable solution for egress and east-west traffic protection.

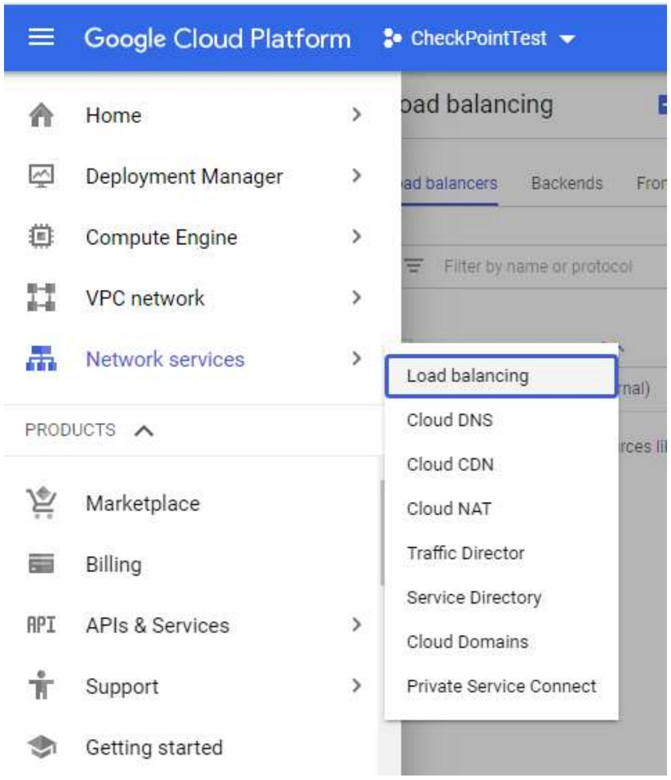


The idea is to have a Hub and Spoke with peering and export default route to the Spokes, this default route is pointing to the iLB as next Hop, redirecting the packets to the Check Point MIG declared as backend.

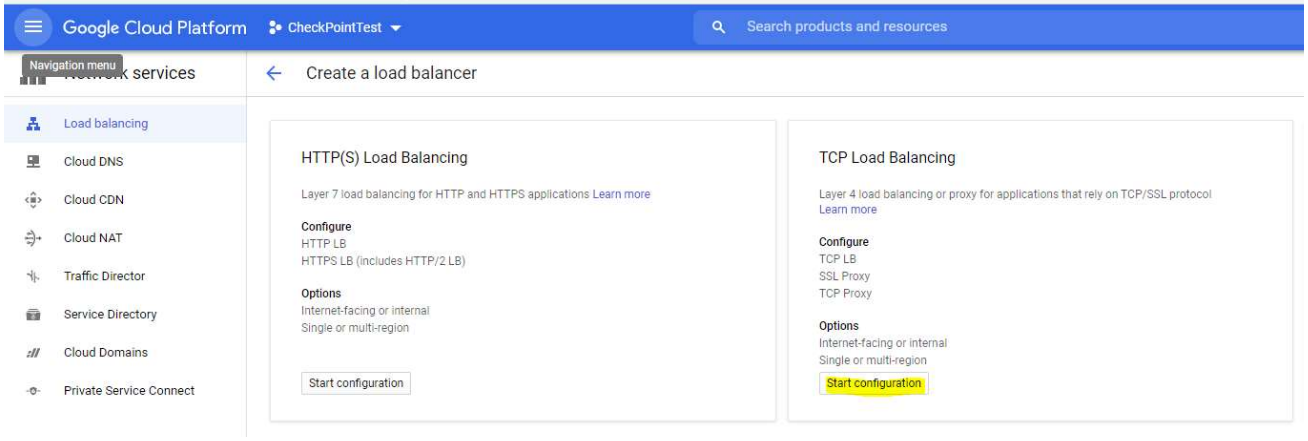
The deployment of the MIG is covered by the Admin Guide here.

Here the steps to create the Load Balancer to allow next hop route.

First, we select Network Services and Load Balancing:



The LB is the kind TCP (when you select to convert it in NextHop it takes TCP/UDP packets)



HTTP(S) Load Balancing

Layer 7 load balancing for HTTP and HTTPS applications Learn more

Configure
HTTP LB
HTTPS LB (includes HTTP/2 LB)

Options
Internet-facing or internal
Single or multi-region

Start configuration

TCP Load Balancing

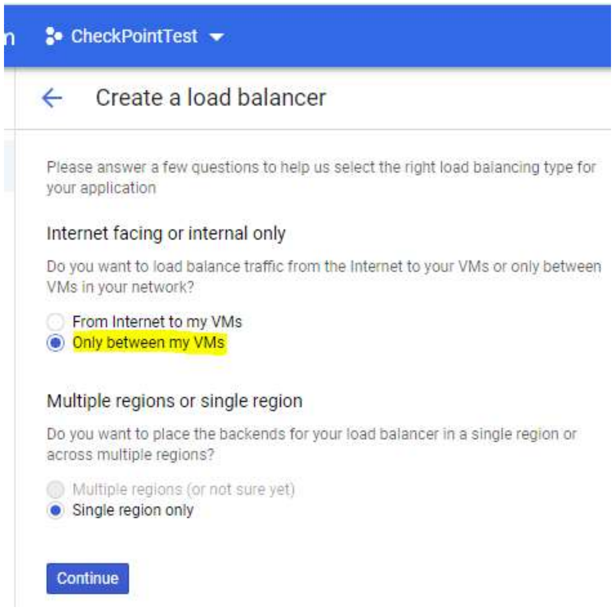
Layer 4 load balancing or proxy for applications that rely on TCP/SSL protocol
Learn more

Configure
TCP LB
SSL Proxy
TCP Proxy

Options
Internet-facing or internal
Single or multi-region

Start configuration

To make it internal, select this way



CheckPointTest

Create a load balancer

Please answer a few questions to help us select the right load balancing type for your application

Internet facing or internal only

Do you want to load balance traffic from the Internet to your VMs or only between VMs in your network?

- From Internet to my VMs
- [✓ Only between my VMs]

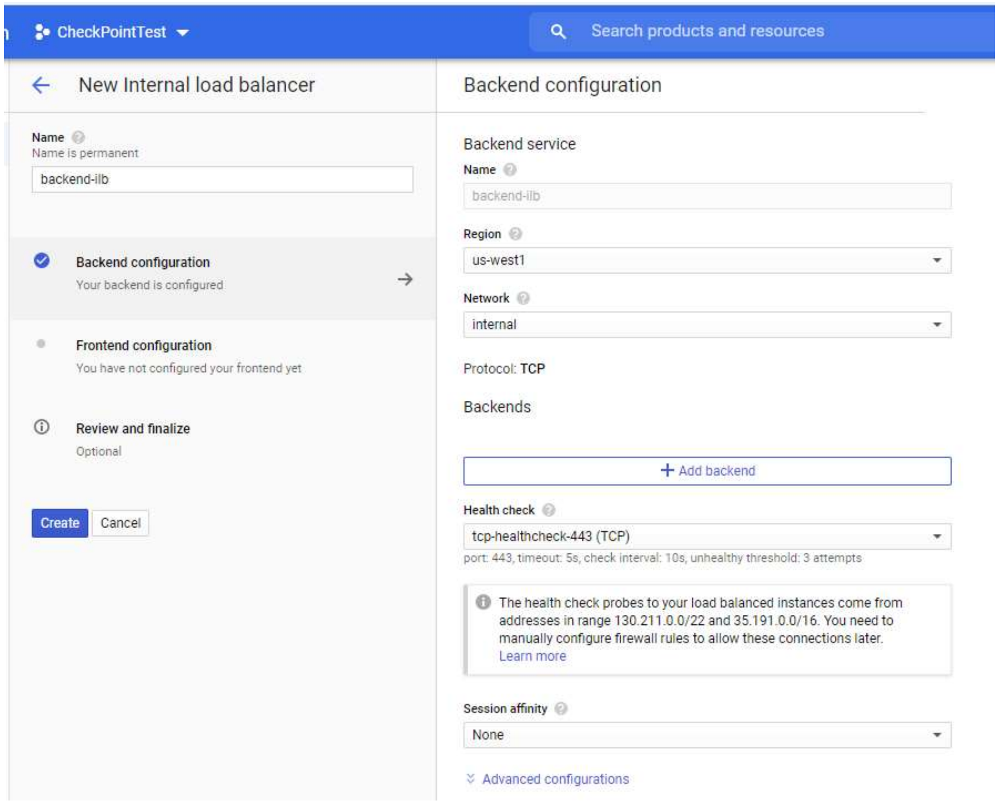
Multiple regions or single region

Do you want to place the backends for your load balancer in a single region or across multiple regions?

- Multiple regions (or not sure yet)
- [✓ Single region only]

Continue

At first phase I will create it without backend, the backend will be the MIG that we will deploy in future.



CheckPointTest | Search products and resources

New Internal load balancer

Name: backend-ilb (Name is permanent)

- Backend configuration — Your backend is configured
- Frontend configuration — You have not configured your frontend yet
- Review and finalize — Optional

Create | Cancel

Backend configuration

Backend service

Name: backend-ilb

Region: us-west1

Network: internal

Protocol: TCP

Backends

+ Add backend

Health check: tcp-healthcheck-443 (TCP)

port: 443, timeout: 5s, check interval: 10s, unhealthy threshold: 3 attempts

The health check probes to your load balanced instances come from addresses in range 130.211.0.0/22 and 35.191.0.0/16. You need to manually configure firewall rules to allow these connections later. Learn more

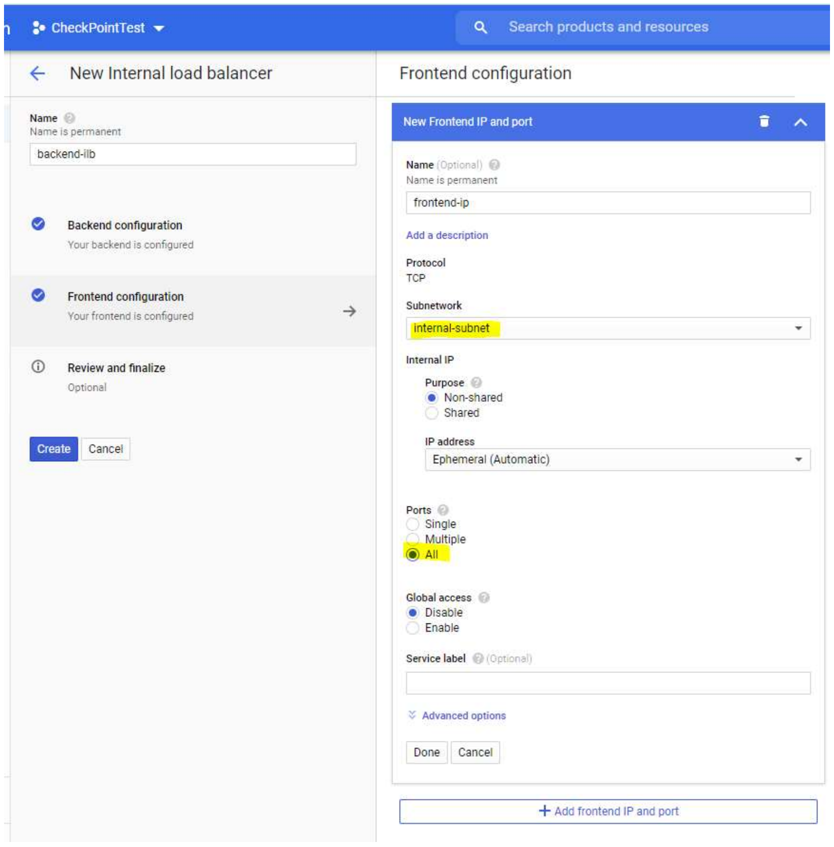
Session affinity: None

Advanced configurations

In addition, you can notice the Health Check is in the port 443 (this only expects to receive a HTTP 200 status), I select this one, because the Check Point GW is GAIA based and we have the WebUI on that port by default, so it will have an answer to the test.

Another thing, the health check do not come from the LB it comes from 130.311.0.0/22 and 35.191.0.0/16.

Since we created the Backend config on the Internal VPC, the frontend will create an IP for the LB, this will select the Subnets on that VPC, your ETH1 of the MIG need to be deployed in that one also.



Name
Name is permanent
backend-ilb

Backend configuration
Your backend is configured

Frontend configuration
Your frontend is configured

Review and finalize
Optional

New Frontend IP and port

Name (Optional)
Name is permanent
frontend-ip

Protocol
TCP

Subnetwork
internal-subnet

Internal IP

Purpose
[✓ Non-shared] Shared

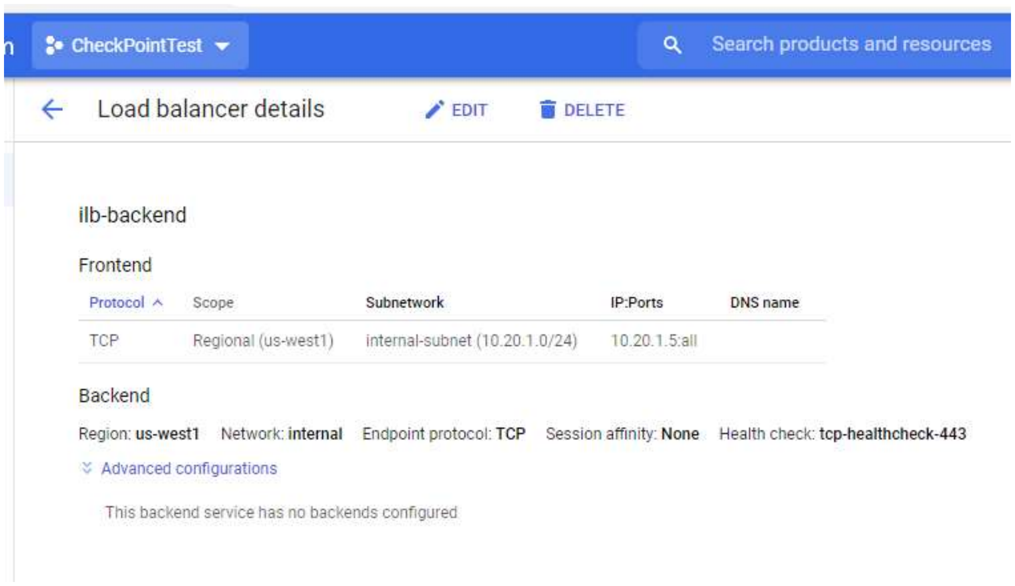
IP address
Ephemeral (Automatic)

Ports
Single Multiple [✓ All]

Global access
[✓ Disable] Enable

Service label (Optional)

To allow the LB be next hop, it needs the Port set to “All”, click on Create and we will have an iLB with internal IP and no backend.



Load balancer details

ilb-backend

Frontend

Protocol	Scope	Subnetwork	IP:Ports	DNS name
TCP	Regional (us-west1)	internal-subnet (10.20.1.0/24)	10.20.1.5:all	

Backend

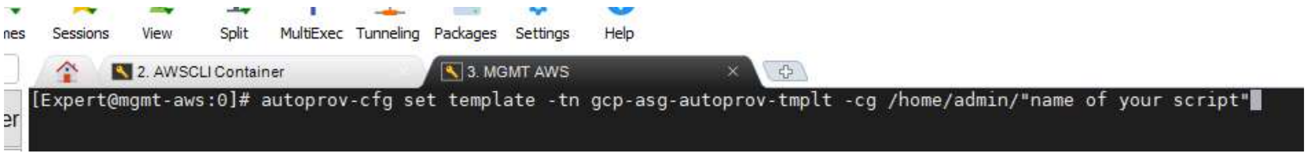
Region: **us-west1** Network: **internal** Endpoint protocol: **TCP** Session affinity: **None** Health check: **tcp-healthcheck-443**

This backend service has no backends configured

Then, we need to set the CME to inject a script into the Check Point Gateway, this script will put the relevant routes to respond the health check probes and the alias with the LB's IP.

I'm using a bash like this one, you need to put the Gateway IP, that is the same IP as the Subnet's Gateway for the ETH1 and the Alias IP that's the same IP address as the LB.

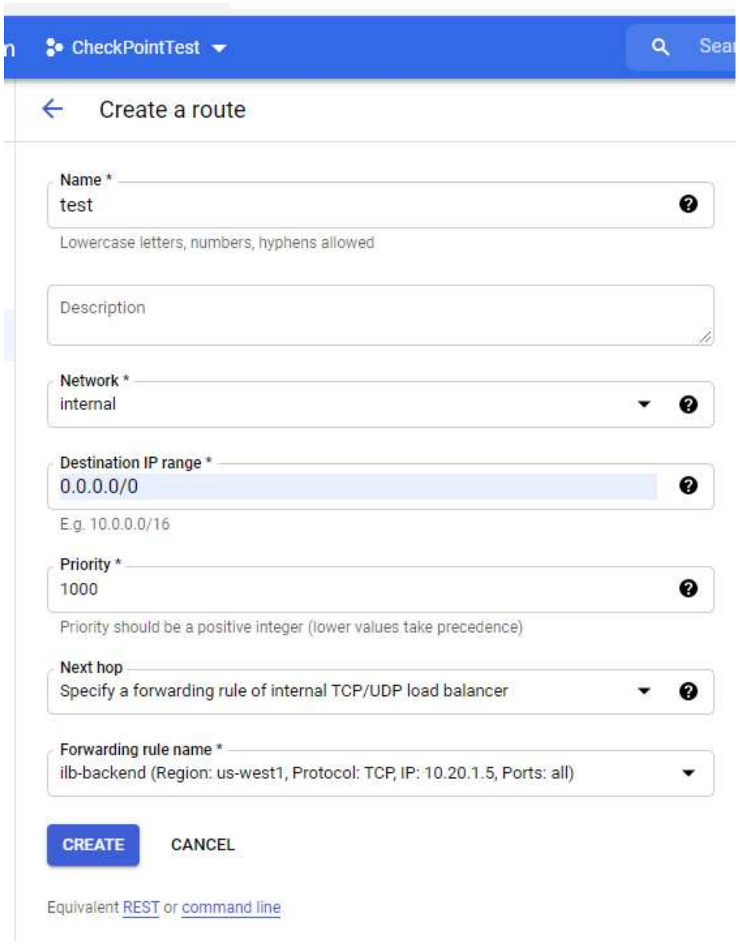
Then you add this script as custom gateway script to the MIG template on the CME, this feature will allow customization to be automated and applied to every new deployed gateway.



```
[Expert@mgmt-aws:0]# autoprov-cfg set template -tn gcp-asg-autoprov-tmplt -cg /home/admin/"name of your script"
```

In this case, my template is named “gcp-asg-autoprov-tmplt”.

With this modification, the Health Checks will respond correctly and now we can set the LB as next hop on a Route Table.



Name *
test
Lowercase letters, numbers, hyphens allowed

Description

Network *
internal

Destination IP range *
0.0.0.0/0
E.g. 10.0.0.0/16

Priority *
1000
Priority should be a positive integer (lower values take precedence)

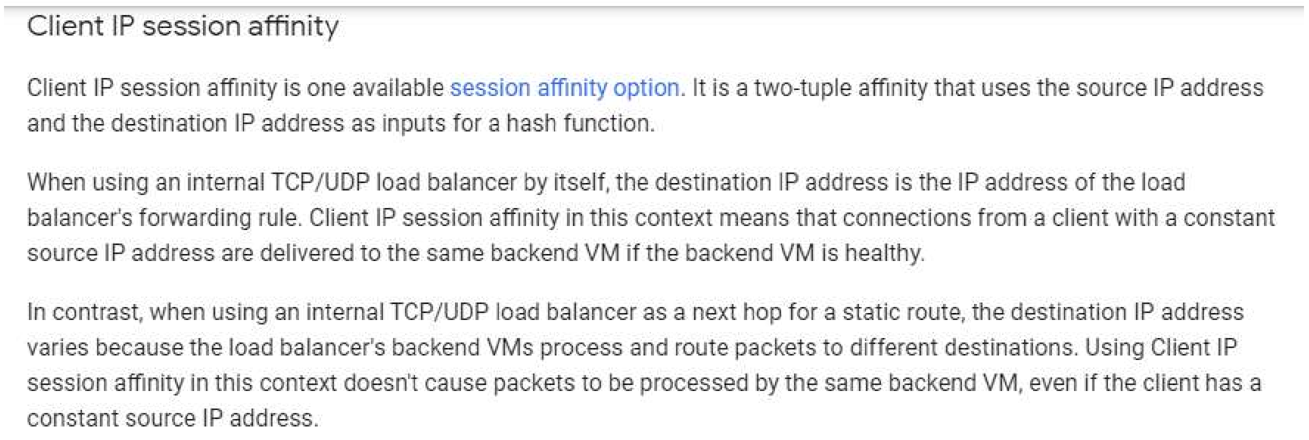
Next hop
Specify a forwarding rule of internal TCP/UDP load balancer

Forwarding rule name *
ilb-backend (Region: us-west1, Protocol: TCP, IP: 10.20.1.5, Ports: all)

Equivalent REST or command line

Now all the E-W and Egress traffic will traverse the Gateway.

One thing to take care is, by the moment of this document GCP have no persistence mechanism on the LB when you use it as Next Hops



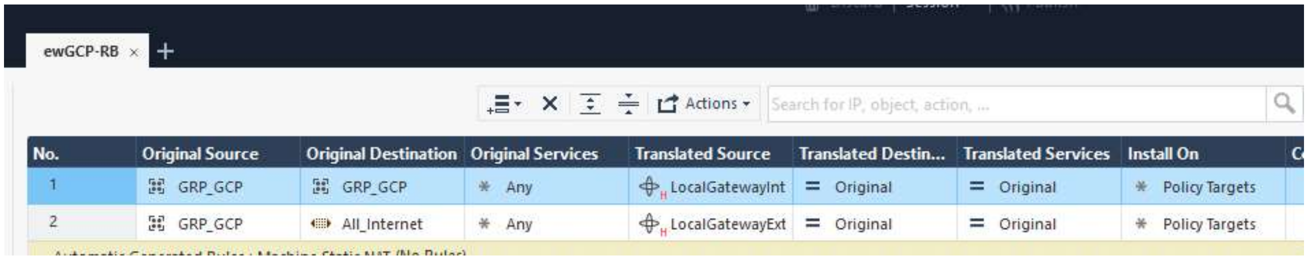
Client IP session affinity

Client IP session affinity is one available session affinity option. It is a two-tuple affinity that uses the source IP address and the destination IP address as inputs for a hash function.

When using an internal TCP/UDP load balancer by itself, the destination IP address is the IP address of the load balancer's forwarding rule. Client IP session affinity in this context means that connections from a client with a constant source IP address are delivered to the same backend VM if the backend VM is healthy.

In contrast, when using an internal TCP/UDP load balancer as a next hop for a static route, the destination IP address varies because the load balancer's backend VMs process and route packets to different destinations. Using Client IP session affinity in this context doesn't cause packets to be processed by the same backend VM, even if the client has a constant source IP address.

Therefore, we need to create a NAT rule to Hide NAT all behind the GW IP, even for East-West traffic.



ewGCP-RB

No.	Original Source	Original Destination	Original Services	Translated Source	Translated Destin...	Translated Services	Install On
1	GRP_GCP	GRP_GCP	Any	LocalGatewayInt	Original	Original	Policy Targets
2	GRP_GCP	All_Internet	Any	LocalGatewayExt	Original	Original	Policy Targets

We can use the Dynamic Objects, since by default all MIG members have the LocalGatewayInternal and LocalGatewayExternal information.

NOTE:

This is a workaround and for the moment is NOT supported, but allows you to use this new feature from GCP and create dynamic load balancing with the MIG.