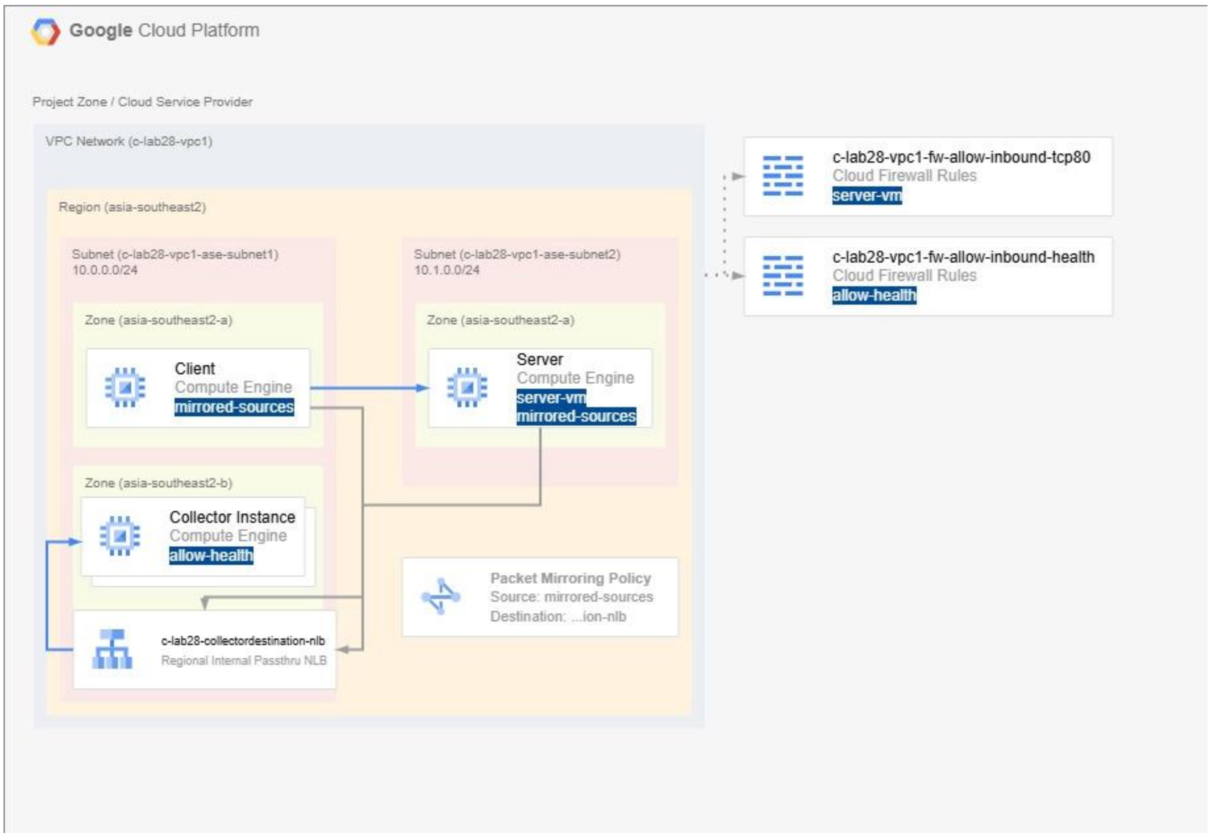
PT Christian Implementation 1 Documentation

Goal

Monitor Traffic between Client and Server VMs

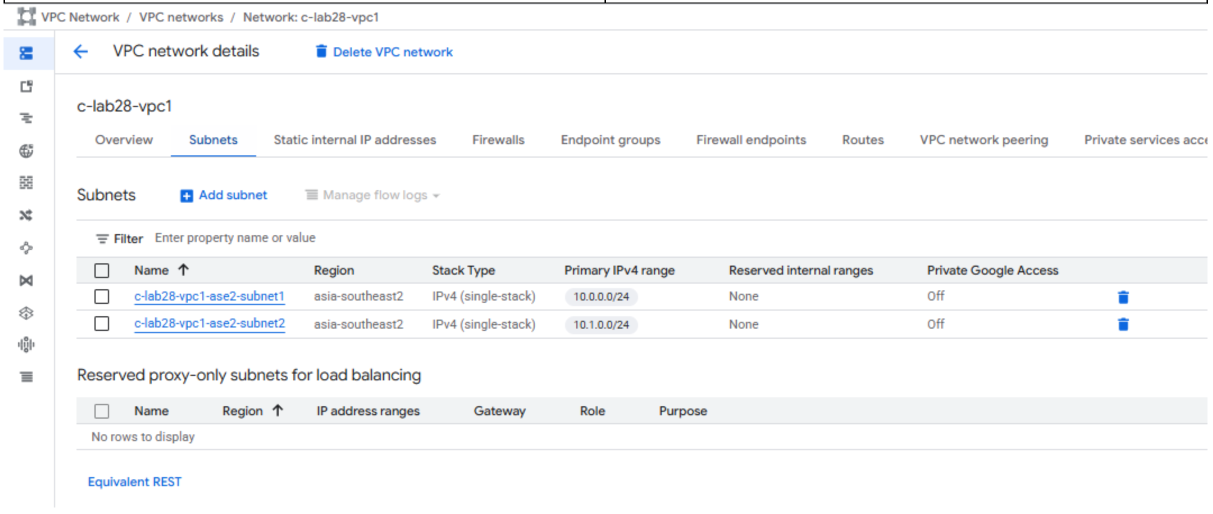

Google Cloud Platform
Project Zone / Cloud Service Provider
VPC Network (c-lab28-vpc1)
Region (asia-southeast2)
Subnet (c-lab28-vpc1-ase-subnet1) 10.0.0.0/24
Subnet (c-lab28-vpc1-ase-subnet2) 10.1.0.0/24
Zone (asia-southeast2-a)
Client Compute Engine mirrored-sources
Zone (asia-southeast2-a)
Server Compute Engine server-vm mirrored-sources
Zone (asia-southeast2-b)
Collector Instance Compute Engine allow-health
c-lab28-collectordestination-nlb Regional Internal Passthru NLB
Packet Mirroring Policy Source: mirrored-sources Destination: ...ion-nlb
c-lab28-vpc1-fw-allow-inbound-tcp80 Cloud Firewall Rules server-vm
c-lab28-vpc1-fw-allow-inbound-health Cloud Firewall Rules allow-health

Step 0: Provision Environment

Created VPC Network

VPC Network name	c-lab28-vpc1
MTU	1460
Subnet creation mode	Custom
Subnetworks	c-lab28-vpc1-ase2-subnet1 Region: asia-southeast2 IPv4 Range: 10.0.0.0/24 VPC Flow Logs: Disabled Hybrid Subnets: Disabled Private Google Access: Disabled

	c-lab28-vpc1-ase2-subnet2 Region; asia-southeast2 IPv4 Range: 10.1.0.0/24 VPC Flow Logs: Disabled Hybrid Subnets: Disabled Private Google Access: Disabled
VPC Firewall Rules	c-lab28-vpc1-allow-custom c-lab28-vpc1-allow-ssh
Dynamic Routing mode	Regional


VPC Network / VPC networks / Network: c-lab28-vpc1

VPC network details   Delete VPC network

c-lab28-vpc1

Overview | Subnets | Static internal IP addresses | Firewalls | Endpoint groups | Firewall endpoints | Routes | VPC network peering | Private services acc

Subnets   Add subnet   Manage flow logs

Filter Enter property name or value

Name ↑	Region	Stack Type	Primary IPv4 range	Reserved internal ranges	Private Google Access
c-lab28-vpc1-ase2-subnet1	asia-southeast2	IPv4 (single-stack)	10.0.0.0/24	None	Off
c-lab28-vpc1-ase2-subnet2	asia-southeast2	IPv4 (single-stack)	10.1.0.0/24	None	Off

Reserved proxy-only subnets for load balancing

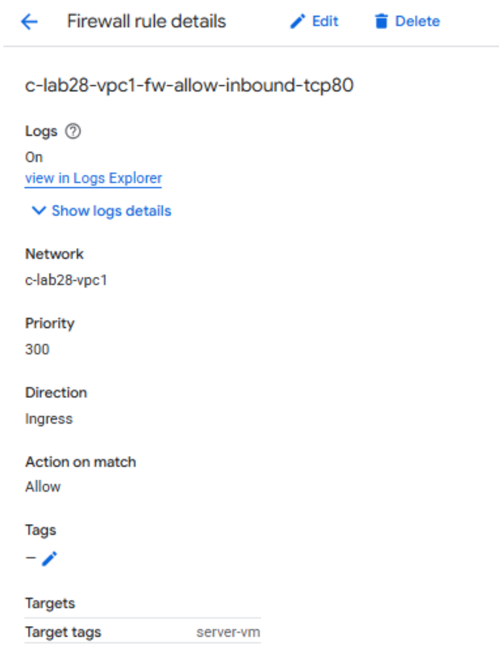
Name	Region ↑	IP address ranges	Gateway	Role	Purpose

No rows to display

Equivalent REST

Created VPC Firewall Rule

VPC Firewall Rule name	c-lab28-vpc1-fw-allow-inbound-tcp80
VPC Network	c-lab28-vpc1
Logs	On
Priority	300
Direction	Ingress
Action	Allow
Targets	Specified Target Tags
Target Tags	server-vm
Source Filter	IPv4 Ranges
Source IPv4 Ranges	0.0.0.0/0
Protocols and Ports	TCP:80



c-lab28-vpc1-fw-allow-inbound-tcp80

Logs
On
view in Logs Explorer
Show logs details

Network
c-lab28-vpc1

Priority
300

Direction
Ingress

Action on match
Allow

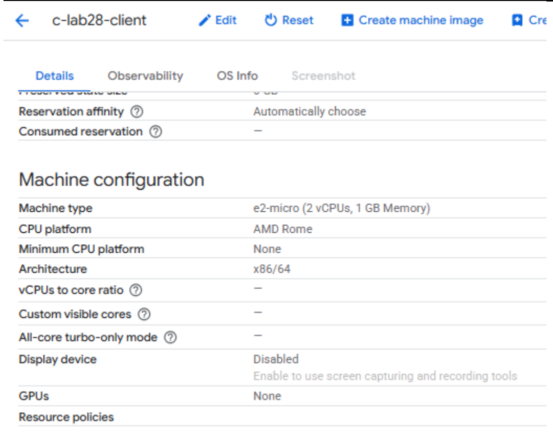
Tags
–

Targets

Target tags	server-vm

Created Virtual Machine (Client)

Virtual Machine name	c-lab28-client
Zone	asia-southeast2-a
Machine type	e2-micro
Boot Disk OS	Debian GNU/Linux 12 (bookworm)
Boot Disk type	Balancer persistent disk
Boot Disk size	10 GB
Network Tags	client-vm
VPC Network	c-lab28-vpc1
Subnetwork	c-lab28-vpc1-ase2-subnet1
Primary Internal IPv4 Address	Ephemeral
External IPv4 Address	Ephemeral



Reservation affinity	Automatically choose
Consumed reservation	–

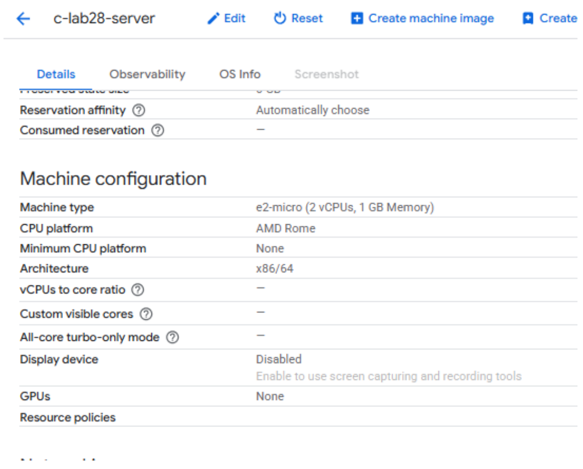
Machine configuration

Machine type	e2-micro (2 vCPUs, 1 GB Memory)
CPU platform	AMD Rome
Minimum CPU platform	None
Architecture	x86/64
vCPUs to core ratio	–
Custom visible cores	–
All-core turbo-only mode	–
Display device	Disabled Enable to use screen capturing and recording tools
GPUs	None
Resource policies	

Created Virtual Machine (Server)

Virtual Machine name	c-lab28-server
Zone	asia-southeast2-a

Machine type	e2-micro
Boot Disk OS	Debian GNU/Linux 12 (bookworm)
Boot Disk type	Balancer persistent disk
Boot Disk size	10 GB
Network Tags	server-vm
VPC Network	c-lab28-vpc1
Subnetwork	c-lab28-vpc1-ase2-subnet2
Primary Internal IPv4 Address	Ephemeral
External IPv4 Address	Ephemeral



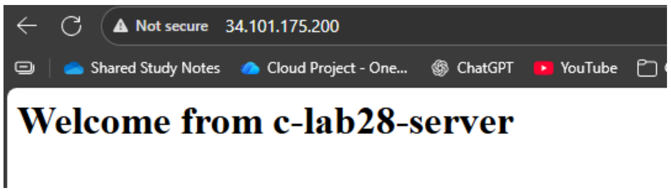
Reservation affinity	Automatically choose
Consumed reservation	–

Machine configuration

Machine type	e2-micro (2 vCPUs, 1 GB Memory)
CPU platform	AMD Rome
Minimum CPU platform	None
Architecture	x86/64
vCPUs to core ratio	–
Custom visible cores	–
All-core turbo-only mode	–
Display device	Disabled Enable to use screen capturing and recording tools
GPUs	None
Resource policies	

Configured Server VM

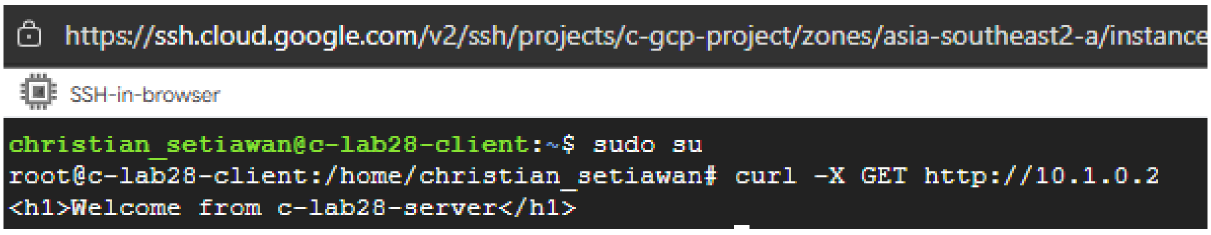
```
apt update
apt install apache2
echo “<h1>Welcome from $(hostname)</h1>” > /var/www/html/index.html
```



Welcome from c-lab28-server

Test #1 – Test Connectivity between Client and Server

```
curl -X GET http://10.1.0.2
```


https://ssh.cloud.google.com/v2/ssh/projects/c-gcp-project/zones/asia-southeast2-a/instance
SSH-in-browser

```
christian_setiawan@c-lab28-client:~$ sudo su
root@c-lab28-client:/home/christian_setiawan# curl -X GET http://10.1.0.2
<h1>Welcome from c-lab28-server</h1>
```

Connectivity established **successfully**!

Step 1: Create the Collector Destination

Theory

Packet Mirroring requires an instance group of **Collector Instances**.

Source: https://docs.cloud.google.com/vpc/docs/using-packet-mirroring#collector_instances

To enable Packet Mirroring, you must have an internal passthrough Network Load Balancer that can serve as a packet mirroring collector.

The **internal passthrough Network Load Balancer** must meet the following requirements:

- The internal passthrough Network Load Balancer's forwarding rule must have **Packet Mirroring** enabled when the rule is created. This status cannot be changed after the rule is created. You can use this forwarding rule to collect both IPv4 and IPv6 traffic.
- The internal passthrough Network Load Balancer is in the same region as the instances that you're mirroring.
- The internal passthrough Network Load Balancer's backend service must use a session affinity of NONE (5-tuple hash).
- The internal passthrough Network Load Balancer's backend service must have backend subsetting disabled.

If your **collector instances** are not set up to respond to the health check that you've configured with your backend service, the health check can fail. Packets can still be mirrored in this case.

Source: https://docs.cloud.google.com/vpc/docs/using-packet-mirroring#collector_instances

To prepare your VPC network for Packet Mirroring traffic, do the following:

- Ensure that **collector instances** in the load balancer's instance group can receive traffic from **mirrored instances** or from the IPv4 and IPv6 address ranges of mirrored instances.

    For example, to let collector instances receive IPv4 traffic from any VM, create a firewall rule with a source IPv4 address range of 0.0.0.0/0. To let collector instances receive IPv6 traffic from any VM, create a firewall rule with a source IPv6 address range of ::/0. To prevent internet traffic from reaching the collector instances, assign only internal IPv4 and IPv6 addresses to them.

- Ensure that **collector instances** can receive traffic from the Google Cloud health checking systems.

  For example, for IPv4 traffic, create a firewall rule that allows traffic to the collector instances from the IPv4 address ranges of 130.211.0.0/22 and 35.191.0.0/16. For IPv6 traffic, create a firewall rule that allows traffic to the collector instances from the IPv6 address range of 2600:2d00:1:b029::/64.

- If you want to test Packet Mirroring by manually sending egress traffic from one or more **mirrored instances**, create a firewall rule that allows SSH traffic to those instances.

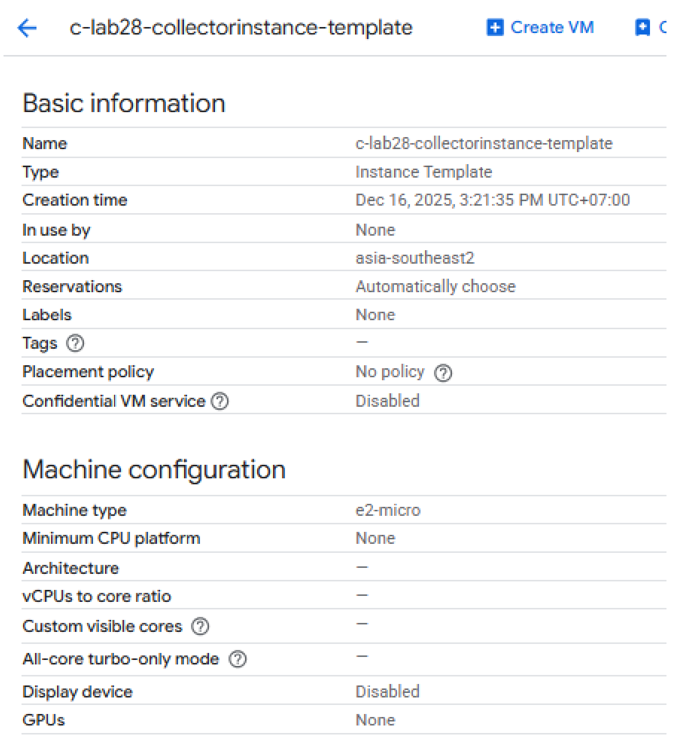
  For example, to allow SSH connections to your mirrored instances from all IPv4 and IPv6 addresses, allow ingress TCP traffic to port 22 from any source IPv4 and IPv6 address. If you want to only allow SSH connections that are initiated from a certain IPv4 or IPv6 address range, specify that IPv4 or IPv6 address range as a source range for the firewall rule. For more information about testing your internal passthrough Network Load Balancer, see Test load balancing.

Source: https://docs.cloud.google.com/vpc/docs/using-packet-mirroring#firewall_rules

Implementation

Created Instance Template

Instance Template name	c-lab28-collectorinstance-template
Location	Regional
Region	asia-southeast2
Machine Type	e2-micro
Boot Disk OS	Debian GNU/Linux 12 (bookworm)
Boot Disk type	Balanced persistent disk
Boot Disk size	10 GB
Network Tags	allow-health
VPC Network	c-lab28-vpc1
Subnetwork	c-lab28-vpc1-ase2-subnet1
External IPv4 Address	None



c-lab28-collectorinstance-template

Basic information

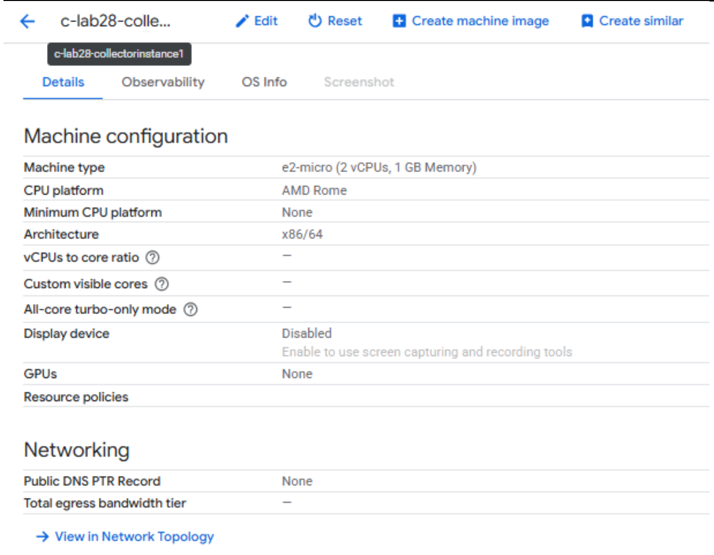
Name	c-lab28-collectorinstance-template
Type	Instance Template
Creation time	Dec 16, 2025, 3:21:35 PM UTC+07:00
In use by	None
Location	asia-southeast2
Reservations	Automatically choose
Labels	None
Tags	–
Placement policy	No policy
Confidential VM service	Disabled

Machine configuration

Machine type	e2-micro
Minimum CPU platform	None
Architecture	–
vCPUs to core ratio	–
Custom visible cores	–
All-core turbo-only mode	–
Display device	Disabled
GPUs	None

Created Virtual Machine (from the Instance Template)

Instance Template name	c-lab28-collectorinstance-template
Zone	asia-southeast2-a
Machine Type	e2-micro
Boot Disk OS	Debian GNU/Linux 12 (bookworm)
Boot Disk type	Balanced persistent disk
Boot Disk size	10 GB
Network Tags	allow-health
VPC Network	c-lab28-vpc1
Subnetwork	c-lab28-vpc1-ase2-subnet1
External IPv4 Address	None



c-lab28-colle...

c-lab28-collectorinstance1

Machine configuration

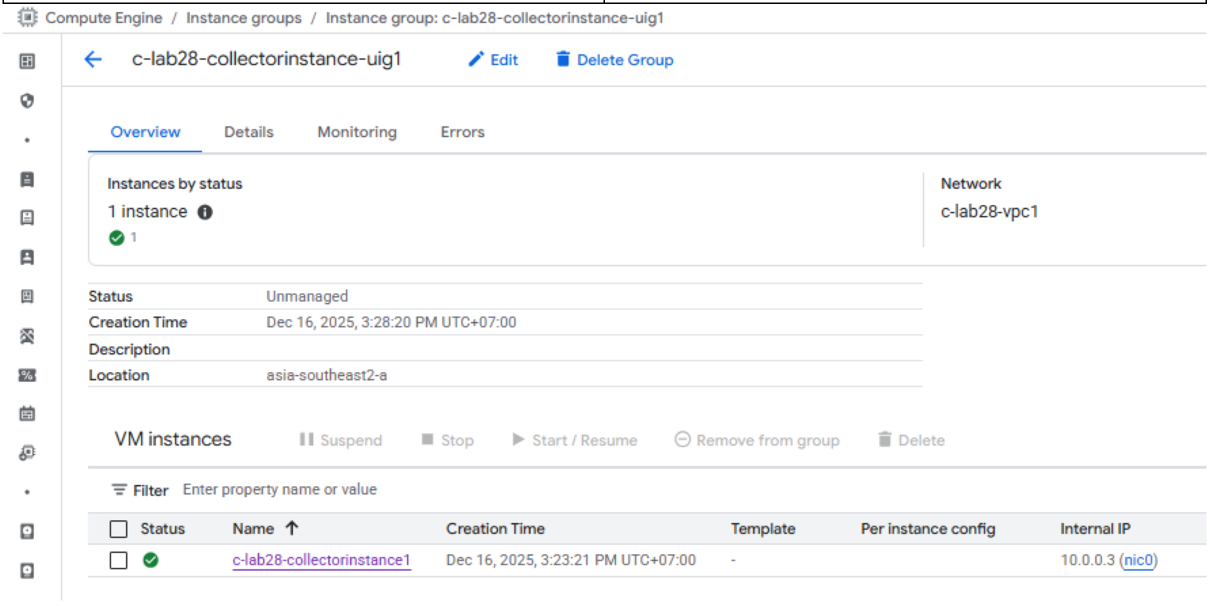
Machine type	e2-micro (2 vCPUs, 1 GB Memory)
CPU platform	AMD Rome
Minimum CPU platform	None
Architecture	x86/64
vCPUs to core ratio	–
Custom visible cores	–
All-core turbo-only mode	–
Display device	Disabled Enable to use screen capturing and recording tools
GPUs	None
Resource policies	

Networking

Public DNS PTR Record	None
Total egress bandwidth tier	–

Created Unmanaged Instance Group

Unmanaged Instance Group name	c-lab28-collectorinstance-uig1
Zone	asia-southeast2-a
Instances	c-lab28-collectorinstance1


Compute Engine / Instance groups / Instance group: c-lab28-collectorinstance-uig1

c-lab28-collectorinstance-uig1

Instances by status
1 instance
1

Network
c-lab28-vpc1

Status	Unmanaged
Creation Time	Dec 16, 2025, 3:28:20 PM UTC+07:00
Description	
Location	asia-southeast2-a

VM instances

Status	Name	Creation Time	Template	Per instance config	Internal IP
✓	c-lab28-collectorinstance1	Dec 16, 2025, 3:23:21 PM UTC+07:00	-		10.0.0.3 (nic0)

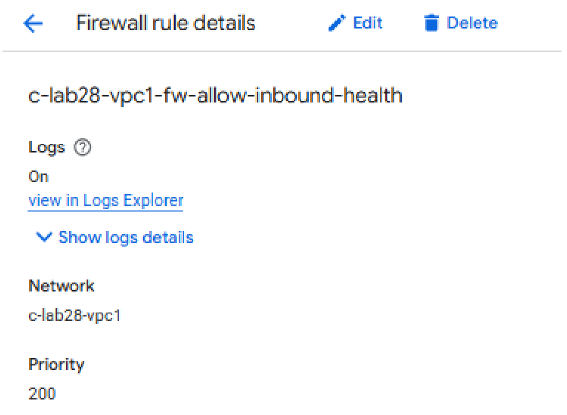
Created Internal Regional Passthrough Network Load Balancer

Load Balancer name	c-lab28-collectordestination-nlb
Region	asia-southeast2
VPC Network	c-lab28-vpc1
Backend Type	Instance Group
Protocol	TCP
Health Check	c-lab28-collectordestination-nlb Protocol: TCP Port: 80 Check Interval: 5 seconds Timeout: 5 seconds Healthy Threshold: 2 Unhealthy Threshold: 2
Backends	Instance Group: c-lab28-collectorinstance-uig1
Session Affinity	None
Frontend IP and Port	c-lab28-collectordestination-nlb-fe1 Protocol: TCP IP Version: IPv4 Subnetwork: c-lab28-vpc1-ase2-subnet1 Internal IP Purpose: Non-shared

	IP Address: Ephemeral (Automatic) Ports: All Global Access: Disable Enable for Packet Mirroring: Enabled

Created VPC Firewall Rule

VPC Firewall Rule name	c-lab28-vpc1-fw-allow-inbound-health
VPC Network	c-lab28-vpc1
Logs	On
Priority	200
Direction	Ingress
Action	Allow
Targets	Specified Target Tags
Target Tags	allow-health
Source Filter	IPv4 Ranges
Source IPv4 Ranges	130.211.0.0/22 35.191.0.0/16 35.235.240.0/20
Protocols and Ports	TCP:80



c-lab28-vpc1-fw-allow-inbound-health

Logs
On
view in Logs Explorer
Show logs details

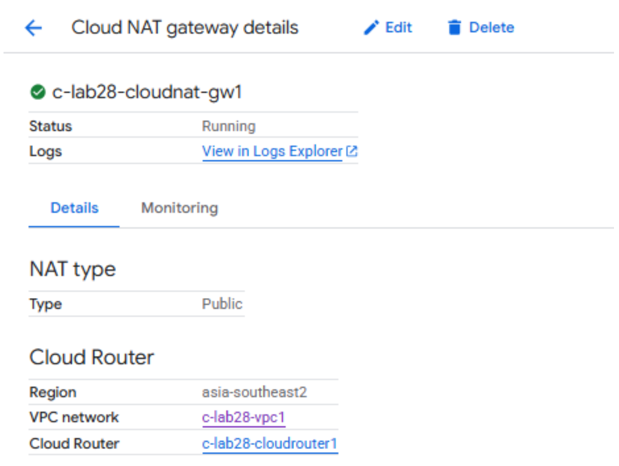
Network
c-lab28-vpc1

Priority
200

Configured Collector Instance to reply to Health Check

Created Cloud NAT Gateway

Cloud NAT Gateway name	c-lab28-cloudnat-gw1
Cloud Router name	c-lab28-cloudrouter1
NAT Type	Public


Cloud NAT gateway details
Edit
Delete

c-lab28-cloudnat-gw1

Status	Running
Logs	View in Logs Explorer

Details
Monitoring

NAT type

Type	Public

Cloud Router

Region	asia-southeast2
VPC network	c-lab28-vpc1
Cloud Router	c-lab28-cloudrouter1

Run a Web Server that listens at Port 80

```
apt update
apt install apache2
```

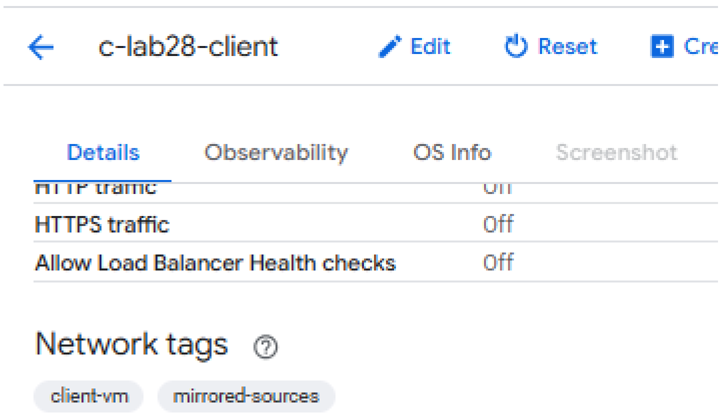
Step 2: Create the Packet Mirroring Policy

Created Packet Mirroring Policy

Packet Mirroring Policy name	c-lab28-packetmirroring-policy1
Region	asia-southeast2
Priority	1000
Enabled	Enabled
Source and Destination	On the same VPC Network
Mirrored Sources	Network Tag: mirrored-sources
Internal Passthrough NLB	c-lab28-collectordestination-nlb-fe1
Traffic to Mirror	Mirror all IPv4 traffic

Updated VMs

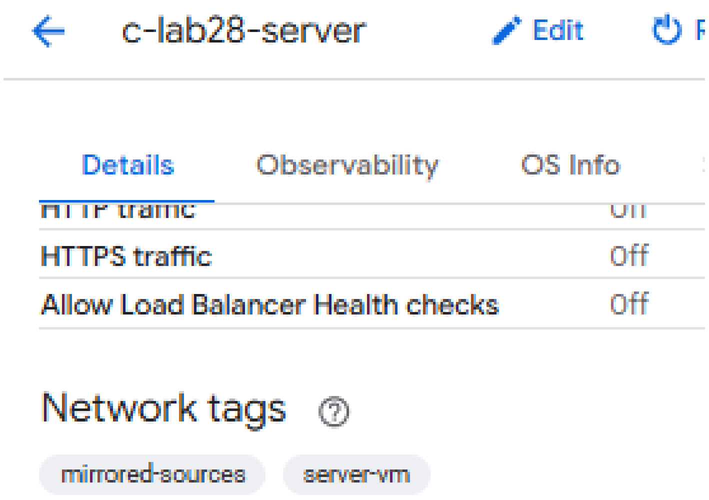
Added **Network Tag “mirrored-sources”** to Client VM and Server VM


c-lab28-client

HTTPS traffic	Off
Allow Load Balancer Health checks	Off

Network tags

client-vm mirrored-sources


c-lab28-server

HTTPS traffic	Off
Allow Load Balancer Health checks	Off

Network tags

mirrored-sources server-vm

Step 3: Verify the Packet Mirroring Policy is working

Theory

To verify that your collector instances are correctly receiving mirrored traffic, you can use **tcpdump**.

1. Connect to a collector instance.
2. If the **tcpdump** command is not available, install it.
3. Identify your network interface:

```
ip address
```

   In the list of network interfaces, find the name that is associated with your collector instance's primary internal IPv4 address—for example, ens4.

4. Start analyzing packets:

```
sudo tcpdump -i INTERFACE_NAME -f "host IP_ADDRESS"
```

Replace the following:

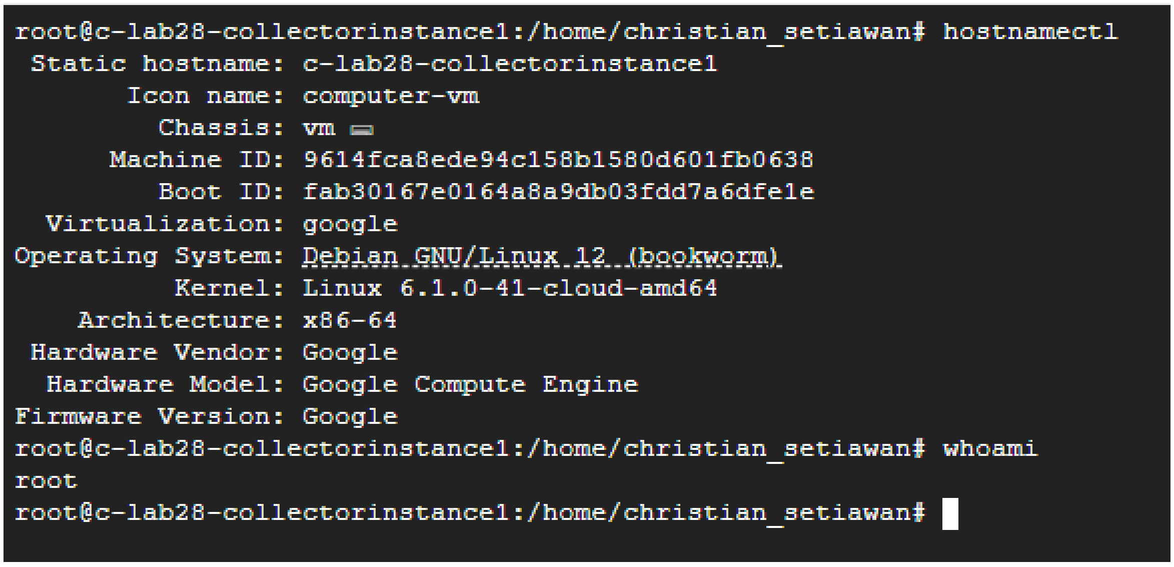
- *INTERFACE_NAME*: the interface name that you identified in step 3.
- *IP_ADDRESS*: the IPv4 address of a mirrored source VM.

5. To run the test, send traffic from the mirrored source VM—for example, by sending an ICMP ping. In the output of tcpdump, verify that you can see the expected traffic.

Source: https://docs.cloud.google.com/vpc/docs/using-packet-mirroring#verify

Implementation

Log In to **Mirrored Source** Instance

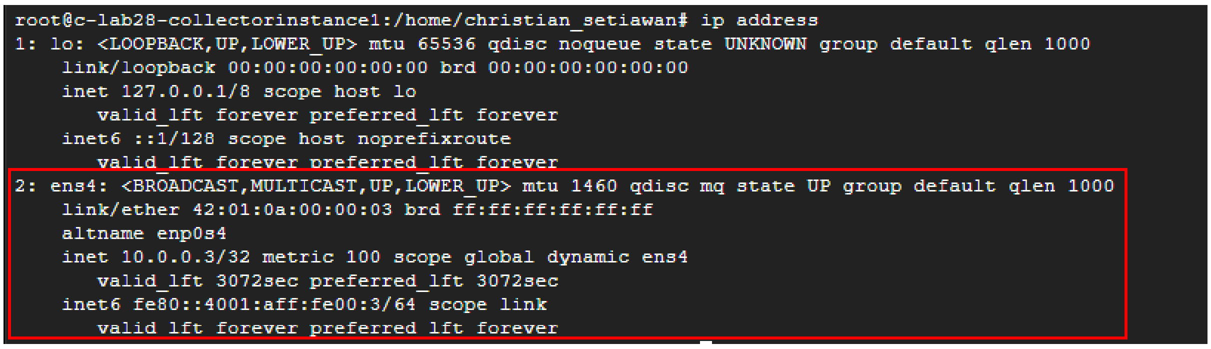


```
root@c-lab28-collectorinstance1:/home/christian_setiawan# hostnamectl
 Static hostname: c-lab28-collectorinstance1
       Icon name: computer-vm
         Chassis: vm 🖴
      Machine ID: 9614fca8ede94c158b1580d601fb0638
         Boot ID: fab30167e0164a8a9db03fdd7a6dfe1e
  Virtualization: google
Operating System: Debian GNU/Linux 12 (bookworm)
          Kernel: Linux 6.1.0-41-cloud-amd64
    Architecture: x86-64
 Hardware Vendor: Google
  Hardware Model: Google Compute Engine
Firmware Version: Google
root@c-lab28-collectorinstance1:/home/christian_setiawan# whoami
root
root@c-lab28-collectorinstance1:/home/christian_setiawan#
```

Executed the following command

Identify the Network Interface:

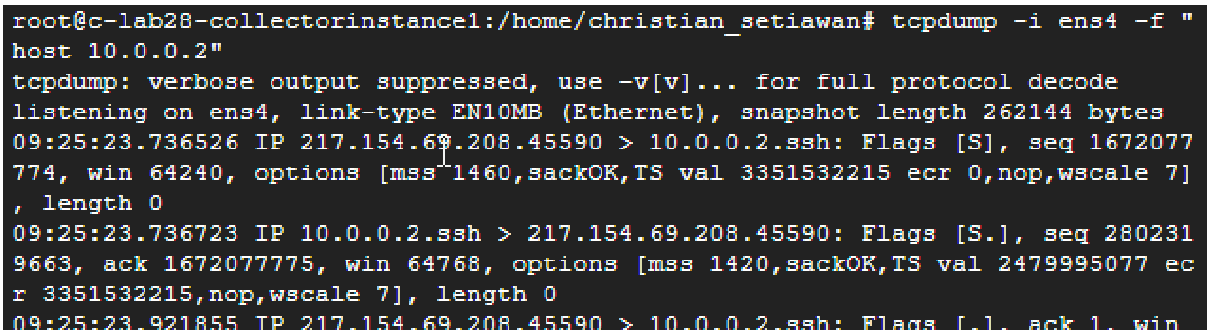
```
ip address
```



```
root@c-lab28-collectorinstance1:/home/christian_setiawan# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1460 qdisc mq state UP group default qlen 1000
    link/ether 42:01:0a:00:00:03 brd ff:ff:ff:ff:ff:ff
    altname enp0s4
    inet 10.0.0.3/32 metric 100 scope global dynamic ens4
       valid_lft 3072sec preferred_lft 3072sec
    inet6 fe80::4001:aff:fe00:3/64 scope link
       valid_lft forever preferred_lft forever
```

In our case, its **ens4**

```
tcpdump -i ens4 -f "host 10.0.0.2"
```



```
root@c-lab28-collectorinstance1:/home/christian_setiawan# tcpdump -i ens4 -f "
host 10.0.0.2"
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on ens4, link-type EN10MB (Ethernet), snapshot length 262144 bytes
09:25:23.736526 IP 217.154.69.208.45590 > 10.0.0.2.ssh: Flags [S], seq 1672077
774, win 64240, options [mss 1460,sackOK,TS val 3351532215 ecr 0,nop,wscale 7]
, length 0
09:25:23.736723 IP 10.0.0.2.ssh > 217.154.69.208.45590: Flags [S.], seq 280231
9663, ack 1672077775, win 64768, options [mss 1420,sackOK,TS val 2479995077 ec
r 3351532215,nop,wscale 7], length 0
09:25:23.921855 IP 217.154.69.208.45590 > 10.0.0.2.ssh: Flags [.], ack 1, win
```

Example of Traffic Captured from Client to Server by Client VM

```
09:23:49.883130 IP 10.0.0.2.35322 > 10.1.0.2.http: Flags [.], ack 266, win 509, options
[nop,nop,TS val 3218440765 ecr 950978835], length 0
```

Packet Mirroring works!

Example of Traffic Captured from Client to Server by Server VM

```
09:26:35.579655 IP 10.0.0.2.40086 > 10.1.0.2.http: Flags [.], ack 265, win 509, options
[nop,nop,TS val 3218606462 ecr 951144531], length 0
```

Packet Mirroring works!

How to read:

- 09:26:35.579655
  Time the Packet was captured

- 10.0.0.2.40086 > 10.1.0.2.http
  Source: 10.0.0.2:40086
  Destination: 10.1.0.2:http

- Flags [.]
  ACK only (no SYN, no FIN, no data)

- ack 265
  The Client acknowledges it receives 265 bytes from the Server

- win 509
  The Client's TCP receive window is 509 bytes at the moment

- [nop,nop,TS val 3218606462 ecr 951144531]
  This shows the TCP Timestamps:
    - TS val : sender's timestamp

    - TS ecr : echoed timestamp from the peer

    User for RTT measurement and PAWS protection

- length 0
  This shows the Payload Length (in this case it’s 0 because its just a TCP ACK)

<u>Summary</u>:

“This packet is a regular TCP ACK from a client at acknowledging data from a server at on port 80. It contains no HTTP data, just TCP-level housekeeping.”

Source: Copilot

DONE