e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:10/October-2024 Impact Factor- 8.187 www.irjmets.com

OPTIMIZING IPV4 UTILIZATION FOR EGRESS-ONLY OPERATIONS IN GOOGLE CLOUD PLATFORM DURING THE TRANSITION TO IPV6

Tom Jose Kalapura[*1]

[*1]Walmart Inc, Sunnyvale CA, USA.

DOI : https://www.doi.org/10.56726/IRJMETS61936

ABSTRACT

This article provides an in-depth analysis of the Google Cloud Platform Routable VPC extension to the non-Routable VPC. Google Cloud's Virtual Private Cloud (VPC) is a virtualized network designed to support cloud-based services and resources. It offers network connectivity, isolation, security, subnets, firewall protection, and routing. This journal examines the architecture of private VPCs, addressing the challenges posed by the IPv4 shortage.

Keywords: Google Shared Networking, Cloud Computing, Scalability, GCP Networking Limits.

I. INTRODUCTION

Routable IPv4 space is rapidly diminishing with the adoption of newer technology stacks. The most effective solution to this problem is transitioning to IPv6. However, due to various constraints and dependencies on other platforms, the migration from IPv4 to IPv6 takes more time than initially planned. This issue is particularly problematic for companies aiming to scale up and adopt new technologies. The transition to IPv6 is essential for ensuring future growth and technological advancement, but it requires careful planning and coordination across multiple systems and platforms. As organizations continue to expand, the pressure to move to IPv6 becomes increasingly critical, underscoring the need for a strategic and phased approach to this transition.

This document explores how to leverage IPv4 space more efficiently, particularly within Google Cloud Platform (GCP), when the requirement is solely for egress operations without the need for ingress. This is typical for many ephemeral workloads that need to read data from PaaS products like Storage, SQL and Bigquery, process that data, and then send the output to another PaaS product. By optimizing the use of IPv4 addresses in such scenarios, organizations can manage their limited IPv4 space more effectively while continuing to support critical operations and workloads. The strategies discussed will help in maximizing the utilization of existing IPv4 resources, thereby easing the transition period to IPv6 and ensuring smoother operational continuity. Additionally, it will address the challenges of managing IPv4 scarcity in the interim, allowing businesses to maintain operational efficiency and readiness for future technological shifts.

Each Cloud provider have each component level hard limits. This solution can be used for solving the constraint with Private RFC1918 as well as Peering Group limits around things like Instance Count, Routes, Subnetworks, etc. This model serves to isolate these workloads in private non-peered VPCs that use RFC5735 Class E Address space to remove several of these constraints.

II. METHODOLOGY

1) Details about each Component in the diagram

- **Shared-vpc-host-project**: A GCP project designated to host all the shared VPC's.
- **VPC Networks (Shared)**:

This project hosts three routable VPC networks, named as follows. These VPCs are peered with each other, allowing communication between networks. They use private **RFC1918**IP ranges, shared across the organization.

- Shared-vpc-network-regionA - SubnetA(10.1.0.0/24)
- Shared-vpc-network-regionB - SubnetB(10.2.0.0/24)
- Shared-vpc-network-regionC - SubnetC(10.3.0.0/24)
- **Private-vpc-network-project**: A GCP project designated to host all the private VPC's.

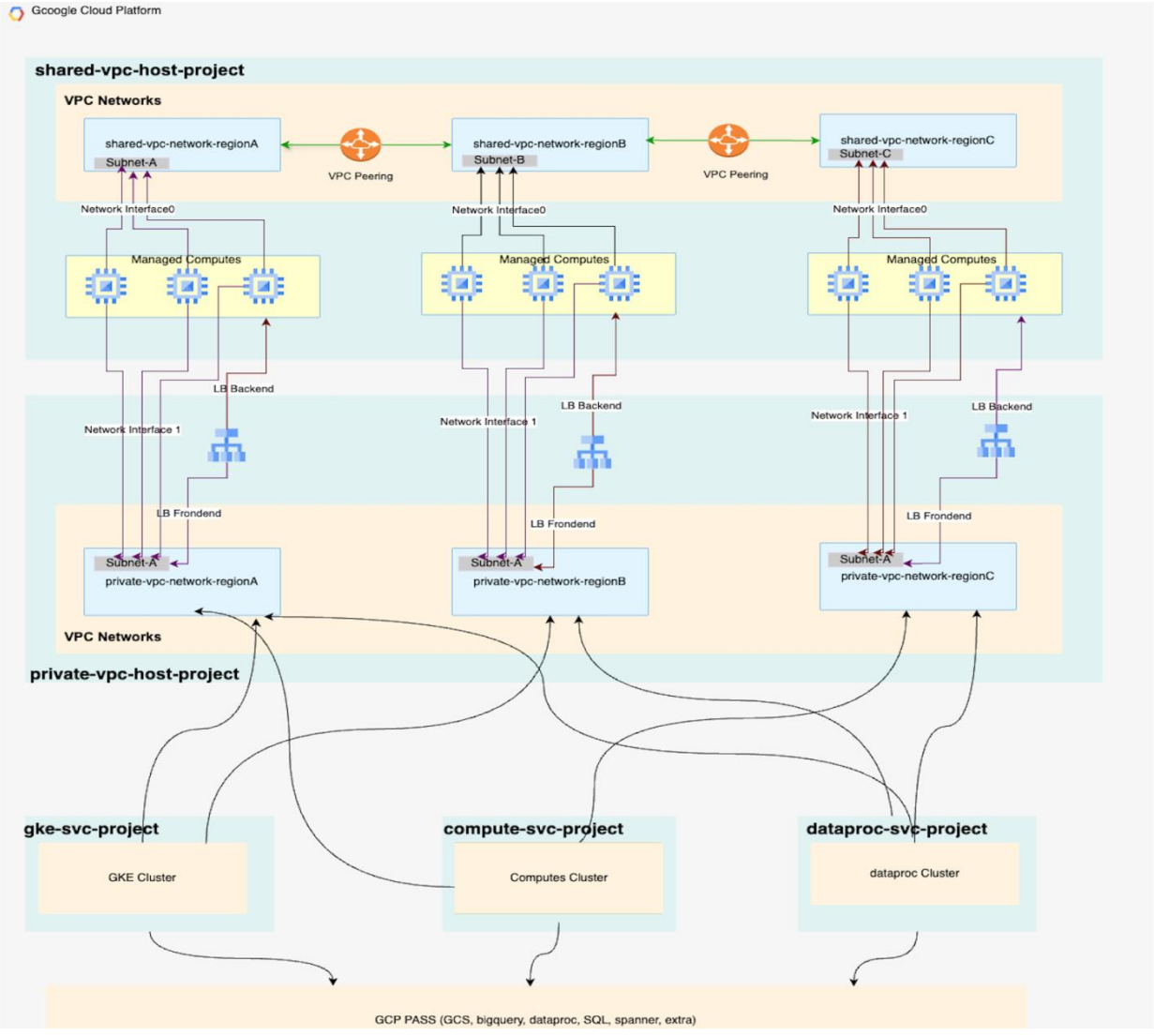
e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:10/October-2024 Impact Factor- 8.187 www.irjmets.com

- **VPC Networks (Private)**:

This project hosts three non-routable VPC networks. These VPCs are not and use **RFC5735** Class E addresses (e.g., 250.0.0.0/8), which are separate and repeatable across each private VPC's.

- private-vpc-network-regionA - SubnetA(250.0.0.0/8)
- private-vpc-network-regionB - SubnetA(250.0.0.0/8)
- private-vpc-network-regionC - SubnetA(250.0.0.0/8)
- **Managed Instance Group & Load Balancer**:
- There will be a **Managed Instance Group** with compute instances based on the egress requirements. Each instance will have two network interfaces: one from the shared VPC network and another from the private VPC network. Additionally, instances will use IP table routes to switch traffic between interfaces.
- **A Load Balancer** will be deployed with an IP address from the non-routable VPC subnet on the frontend and the Managed Instance Group on the backend.
- **VPC Firewall**: Configure the VPC firewall to:
- Allow IAP tunnels for SSH access to compute instances.
- Permit egress to the subnet range.
- Allow ingress for the Load Balancer health check ranges.
- **<service>-svc-project:** GCP project to host services like GKE, compute, dataproc, extra

Architecture Diagram



e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:10/October-2024 Impact Factor- 8.187 www.irjmets.com

- **VPC Routing for RFC1918 Ranges, restricted and fallback:**
  - 10.0.0.0 - 10.255.255.255 (10/8 prefix)
  - 172.16.0.0 - 172.31.255.255 (172.16/12 prefix)
  - 192.168.0.0 - 192.168.255.255 (192.168/16 prefix)
  - Routing for restricted PaaS endpoint ranges. (199.36.153.4/29,)
  - Set fallback routing to 0.0.0.0/0.
- **IP Table Routing**:

Managed instances should have IP table routes configured to route traffic for:

  - 35.191.0.0/16
  - 130.211.0.0/22
- **Health Checks**:

Properly configure health checks to auto repair instances in case of any health events.

2) GCP Shared and Private VPC Setup Sample Code

This Terraform code defines the setup of a private VPC network and related resources within Google Cloud, focusing on private service access (PSA), managed instance groups (MIGs), load balancing, and firewall rules.

a) VPC and Subnetwork Creation

- The google_compute_network resource creates a Virtual Private Cloud (VPC) named vpcnet-private-svc-access-usw1 with regional routing mode and disabled subnet auto-creation. This VPC is hosted in the project private-vpc-network-regionA.
- The google_compute_subnetwork resource defines a private subnetwork (private-vpc-subnet) in region1, with a custom CIDR range 250.0.0.0/8, ensuring private Google access is enabled.

```
resource "google_compute_network" "vpcnet-private-svc-access-usw1" {
      name                    = "private-vpc-network-regionA"
      auto_create_subnetworks  = false
      project                 = "private-vpc-host-project"
      routing_mode            = "REGIONAL"
}

resource "google_compute_subnetwork" "private-vpc-subnet" {
      project                 = google_compute_network.vpcnet-private-svc-access-usw1.project
      name                    = "private-vpc-subnet"
      ip_cidr_range           = "250.0.0.0/8"
      region                  = var.region1
      network                 = google_compute_network.vpcnet-private-svc-access-usw1.name
      private_ip_google_access  = true
}
```

b) IAM Binding for Subnetwork

- The IAM binding (google_compute_subnetwork_iam_binding) allows users in the group1@domain.com group the role of compute. Network User to interact with the private subnetwork created in the previous step.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:10/October-2024 Impact Factor- 8.187 www.irjmets.com

```
resource "google_compute_subnetwork_iam_binding" "private-
vpc-subnet-iam-networkUser" {
      project                  = google_compute_network.vpcnet-
private-svc-access-usw1.project
      region                   =
google_compute_subnetwork.private-vpc-subnet.region
      subnetwork               =
google_compute_subnetwork.private-vpc-subnet.name
      role                     = "roles/compute.networkUser"
      members                    = [
"group:group1@domain.com" ]
}
```

c) Private Service Access Configuration

- The google_compute_global_address resource reserves an internal IP address (252.0.0.0) to be used for VPC peering. This allows private services to communicate over this connection.

```
resource "google_compute_global_address" "private-vpc-psa" {
      name                     = "private-vpc-psa"
      address_type               = "INTERNAL"
      purpose                  = "VPC_PEERING"
      network                  =
google_compute_subnetwork.private-vpc-subnet.network
      address                  = "252.0.0.0"
      prefix_length            = 8
}
```

d) Compute Instance Template

- The module "private-instance_template" creates a Compute Engine instance template for the private VPC, hosted in regionA. This template uses a startup script (startup.sh) and a predefined machine type, image, and disk configuration. The template supports dual networking with interfaces in both the private and shared VPCs.
- Key components explained:

1. Networking: It specifies a network and subnetwork, including additional subnetwork configurations for improved connectivity. The primary interface connects to a shared VPC, which is used for standard network communication across different projects.The can_ip_forward parameter allows instances to route traffic, often used for networking functions like NAT gateways.
2. Image and Disks: The module allows specifying custom images, families, and projects. Disk properties, including size, type, and auto-delete behavior, can be customized to suit different workload requirements.
3. Startup Script: The metadata block includes a startup script path, which is useful for bootstrapping VMs with required software and configurations.
4. Additional Networks: This section adds further network configurations, enabling the VM to attach to multiple networks. This secondary interface connects to a private VPC.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:10/October-2024 Impact Factor- 8.187 www.irjmets.com

```
module "private-instance_template" {
      source                = "terraform-google-
modules/vm/google//modules/private-instance_template"
      name_prefix           = "private-vpc-network-regionA-
tmp"
      project_id            =
google_compute_subnetwork.private-vpc-subnet.project
      machine_type                = var.machine_type
      metadata              = {
                                  "startup-script" =
file("${path.module}/startup.sh")
                                  }

      /* network */
      network               =
"https://www.googleapis.com/compute/v1/projects/shared-
vpc-host-project/global/networks/shared-vpc-network-
regionA"
      subnetwork            =
"https://www.googleapis.com/compute/v1/projects/shared-
vpc-host-project/regions/us-central1/subnetworks/subnet-
01"
      subnetwork_project    = "shared-vpc-host-project"
      can_ip_forward        = true

      /* image */
      source_image          = var.source_image
      source_image_family   = var.source_image_family
      source_image_project  = var.source_image_project
```

e) Managed Instance Group (MIG)

- The module "private-managed-instance-group" defines a Managed Instance Group (MIG) that deploys multiple instances based on the template. It ensures high availability with auto-scaling and health checks.
- Auto-scaling parameters are defined, and policies ensure the MIG can scale up or down based on load, CPU usage, or custom metrics.

```
module "private-managed-instance-group" {
    source                      = "terraform-google-modules/vm/google//modules/mig"
    project_id                  = google_compute_subnetwork.private-vpc-subnet.project
    hostname                    = "private-vpc-network-regionA-mig"
    region                      = google_compute_subnetwork.private-vpc-subnet.region
    private-instance_template   = module.private-instance_template.self_link
    target_size                 = var.target_size

    /* health check */
    health_check                = var.health_check

    /* autoscaler */
    autoscaler_name             = "private-vpc-network-regionA-psa-autoscaler"
    autoscaling_enabled         = var.autoscaling_enabled
    max_replicas                = var.max_replicas
    min_replicas                = var.min_replicas
    cooldown_period             = var.cooldown_period
    autoscaling_cpu             = var.autoscaling_cpu
    autoscaling_metric          = var.autoscaling_metric
    autoscaling_lb              = var.autoscaling_lb
    autoscaling_scale_in_control  = var.autoscaling_scale_in_control

    /* Rolling update */
    update_policy                   = var.update_policy
}
```

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:10/October-2024 Impact Factor- 8.187 www.irjmets.com

f) Load Balancer Configuration

- The google_compute_region_backend_service resource sets up a regional load balancer backend to distribute traffic across the MIG, using health checks and client IP-based session affinity.
- The google_compute_forwarding_rule creates a frontend for the load balancer that directs traffic to the backend using an internal TCP load balancer. This resource links the forwarding rule to the subnet and load balancer.

```
resource "google_compute_region_backend_service" "private-load-lb-backend" {
        depends_on                      = [module.private-managed-instance-group]
        project                         = google_compute_subnetwork.private-vpc-subnet.project
        name                            = "priv-svc-access-usc1"
        region                          = google_compute_subnetwork.private-vpc-subnet.region
        protocol                        = "TCP"
        timeout_sec                     = 30
        connection_draining_timeout_sec = 300
        session_affinity                = "CLIENT_IP" #"CLIENT_IP_PORT_PROTO"
        health_checks                   = module.private-managed-instance-group.health_check_self_links
        network                         = google_compute_subnetwork.private-vpc-subnet.network
        backend {
            group                           = module.private-managed-instance-group.instance_group
            balancing_mode                  = "CONNECTION"
        }
}

resource "google_compute_forwarding_rule" "private-load-lb-connector" {
        depends_on            = [module.private-managed-instance-group]
        project               = google_compute_subnetwork.private-vpc-subnet.project
        name                  = "priv-svc-access-usc1-frontend"
        region                = google_compute_subnetwork.private-vpc-subnet.region
        network               = google_compute_subnetwork.private-vpc-subnet.network
        subnetwork            = google_compute_subnetwork.private-vpc-subnet.id
        load_balancing_scheme = "INTERNAL"
        backend_service       = google_compute_region_backend_service.private-load-lb-backend.self_link
        ip_protocol           = "TCP"
        all_ports             = true
}
```

g) Routing Setup

- The module "vpc-router"defines routing rules that handle traffic both inside and outside the VPC. Routes are configured to handle traffic to:
  - External IP addresses via a fallback default route (0.0.0.0/0).
  - Private ranges (172.16.0.0/12, 10.0.0.0/8, and 192.168.0.0/16) through the internal load balancer (iLB).
  - Restricted endpoints (199.36.153.4/30 and 199.36.153.8/30) via the internet.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:10/October-2024 Impact Factor- 8.187 www.irjmets.com

```
module "vpc-router" {
    source              = "terraform-google-
modules/network/google//modules/routes"
    project_id          = google_compute_subnetwork.private-
vpc-subnet.project
    network_name        = google_compute_subnetwork.private-
vpc-subnet.network

    routes = [
        {
            name                = "tf-managed-internet-
        fallback-0-0-0-0-usc1"
            destination_range   = "0.0.0.0/0"
            priority            = 0
            next_hop_ilb        =
        google_compute_forwarding_rule.private-load-lb-
        connector.ip_address
        },
        {
            name                = "tf-managed-priv-svc-
        access-172-16-0-0-usc1"
            destination_range   = "172.16.0.0/12"
            priority            = 0
            next_hop_ilb    =
        google_compute_forwarding_rule.private-load-lb-
        connector.ip_address
        },
        {
            name                = "tf-managed-priv-svc-
        access-10-0-0-0-usc1"
```

h) Firewall Rules

- The firewall rules (module "firewall_rules") control the ingress and egress traffic for the private VPC network:
  - priv-svc-allow-iap-tunnel-rdp-ssh-usw1: Allows SSH (port 22) and RDP (port 3389) connections via the IAP tunnel.
  - priv-svc-allow-psa-egress-usw1: Allows outbound egress traffic to the PSA CIDR range 240.0.0.0/4.
  - priv-svc-allow-ilb-health-probes-usw1: Allows health checks from Google Cloud's internal load balancer ranges (130.211.0.0/22, 35.191.0.0/16).
  - priv-svc-allow-psa-ingress-usw1: Allows inbound traffic from the PSA range and internal network (240.0.0.0/4).

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:10/October-2024 Impact Factor- 8.187 www.irjmets.com

```
module "firewall_rules" {
    source              = "terraform-google-
    modules/network/google//modules/firewall-rules"
    project_id          = google_compute_subnetwork.private-
vpc-subnet-usw1.project
    network_name        = google_compute_subnetwork.private-
vpc-subnet-usw1.network

    rules = [
    {
        name            = "priv-svc-allow-iap-tunnel-rdp-ssh-
usw1"
        direction       = "INGRESS"
        priority        = 1000
        ranges          = ["35.235.240.0/20"]
        allow = [{
            protocol = "tcp"
            ports    = ["22", "3389"]
        }]
    },
    {
        name            = "priv-svc-allow-psa-egress-usw1"
        direction       = "EGRESS"
        priority        = 1000
        ranges          = ["240.0.0.0/4"]
        allow = [{
            protocol = "all"
        }]
    },
    {
```

9. Bash Script Explanation: Dual NIC Setup, IP Forwarding, and Routing Configuration

- This script configures a Google Compute Engine instance with dual network interfaces for routing and NAT (Network Address Translation). It enables IP forwarding, sets up iptables rules, and configures routing tables for specific network traffic handling, such as health check probes and Private Service Access (PSA) routes.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:10/October-2024 Impact Factor- 8.187 www.irjmets.com

- Script Breakdown:

1. Update & Install Net Tools:

- apt update updates the package list on the system.
- apt -y install net-tools installs net-tools, which provides network interface utilities.

2. Fetching Primary Interface Details:

- The script retrieves the gateway and IP address of the primary (first) network interface from GCP's instance metadata using curl commands.
- It then identifies the interface name associated with the primary IP address using the ip addr show command.

3. Fetching Secondary Interface Details:

- Similarly, it retrieves the gateway, IP address, and subnet mask for the secondary network interface (second NIC).
- It identifies the interface name for this NIC, similar to the primary interface.

4. Validation of Dual NIC Setup:

- If either of the interfaces isn't detected, the script exits with an error message, ensuring a dual NIC setup is present.

5. IP Forwarding Configuration:

- IP forwarding is enabled in the kernel by writing 1 to /proc/sys/net/ipv4/ip_forward.
- This setting is made persistent across reboots by adding net.ipv4.ip_forward=1 to the /etc/sysctl.d/20-natgw.conf file.

6. Installing iptables-persistent:

- iptables-persistent is installed to save the iptables rules across system reboots.
- Pre-configured responses (debconf-set-selections) ensure the installation happens without user interaction.

7. NAT Setup on Primary Interface:

- A POSTROUTING rule is added to the iptables to masquerade (hide) outbound traffic going out of the primary network interface (NIC 1).

8. IP Forwarding on Secondary Interface:

- A FORWARD rule is added to accept traffic coming into the secondary network interface (NIC 2), allowing the traffic to be forwarded.

9. Saving iptables Configuration:

- The current iptables configuration is saved using iptables-save > /etc/iptables/rules.v4, ensuring that it persists across reboots.

10. Source-Based Policy Routing for Health Checks:

- A new routing table (rt-nic1) is added to /etc/iproute2/rt_tables for the secondary interface.
- A rule is added to route traffic from the secondary interface's gateway/subnet through the rt-nic1 routing table.
- This ensures health check replies are sent over the correct NIC instead of the default gateway.

11. Adding Routes for iLB (Internal Load Balancer) Health Checks:

- Routes are set for health check traffic (ranges 35.191.0.0/16 and 130.211.0.0/22) through the secondary interface's gateway.
- Script Breakdown:

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:10/October-2024 Impact Factor- 8.187 www.irjmets.com

```
apt update
apt -y install net-tools
# Primary Interface Details
INTERFACE1_GATEWAY="$(curl --silent -H 'Metadata-
Flavor:Google'
http://metadata.google.internal/computeMetadata/v1/instanc
e/network-interfaces/0/gateway)"
INTERFACE1_IPV4="$(curl --silent -H 'Metadata-
Flavor:Google'
http://metadata.google.internal/computeMetadata/v1/instanc
e/network-interfaces/0/ip)"
INTERFACE1=$(ip addr show | grep "$INTERFACE1_IPV4" |
awk '{ print $NF }')

# Secondary Interface Details
INTERFACE2_GATEWAY="$(curl --silent -H 'Metadata-
Flavor:Google'
http://metadata.google.internal/computeMetadata/v1/instanc
e/network-interfaces/1/gateway)"
INTERFACE2_IPV4="$(curl --silent -H 'Metadata-
Flavor:Google'
http://metadata.google.internal/computeMetadata/v1/instanc
e/network-interfaces/1/ip)"
INTERFACE2_MASK="$(curl --silent -H 'Metadata-
Flavor:Google'
http://metadata.google.internal/computeMetadata/v1/instanc
e/network-interfaces/1/subnetmask)"
INTERFACE2=$(ip addr show | grep "$INTERFACE2_IPV4" |
awk '{ print $NF }')
if [[ -z "$INTERFACE1" ]] || [[ -z "$INTERFACE2" ]];
```

3) Key Features

a. VPC Network: A custom VPC (vpcnet-private-svc-access-usw1) with private service access and regional routing.

b. Subnetwork: A private subnetwork (private-vpc-subnet) in region1, configured for private IP access.

c. IAM Roles: Configured IAM binding for network access via compute.networkUser role.

d. Private Service Access: Setup of internal IP ranges for service-to-service communication via VPC peering.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:10/October-2024 Impact Factor- 8.187 www.irjmets.com

e. Managed Instance Group: Auto-scaled managed instances with dual NIC interfaces (for private and shared VPCs).

f. Load Balancing: An internal TCP load balancer distributes traffic to the MIG.

g. Routing: Routing for private ranges, PSC endpoints, and fallback to the internet.

h. Firewall Rules: Specific ingress and egress firewall rules to control traffic between the private VPC and external resources.

i. This setup is optimized for securely handling internal traffic between Google Cloud services and the external world via managed instances, while enforcing strict networking, routing, and security policies.

III. CONCLUSION

In conclusion, the approach outlined in this document offers an effective solution for managing the diminishing availability of IPv4 addresses during the transition to IPv6, particularly in Google Cloud Platform (GCP). By isolating workloads in non-peered VPCs using RFC5735 Class E address space and focusing on egress-only operations, this strategy helps overcome IPv4 limitations such as instance counts, routes, and subnetworks. This method allows organizations to continue critical operations while maximizing their limited IPv4 resources, ensuring smoother continuity and scalability as they gradually move towards full IPv6 adoption.

This solution is particularly advantageous for ephemeral workloads that only require outbound connectivity to Platform-as-a-Service (PaaS) products such as BigQuery, Cloud Storage, and Cloud SQL. By eliminating ingress requirements, businesses can optimize the efficiency of their remaining IPv4 space. Additionally, this model mitigates constraints on private RFC1918 space and peering group limits, helping to alleviate the burden of scaling under existing IPv4 limitations. The adoption of this approach allows companies to maintain operational effectiveness while managing IPv4 exhaustion, all while positioning themselves for future growth through a phased and strategic migration to IPv6.

In essence, this document provides organizations with a practical interim solution that balances the need for current operational scalability with the long-term goal of IPv6 migration. This proactive approach not only supports day-to-day operations but also ensures readiness for future technological shifts without compromising performance or availability.

IV. REFERENCE

[1] M. Tatipamula, P. Grossetete, and H. Esaki, “IPv6 integration and coexistence strategies for next-generation networks,” IEEE Commun. Mag., vol. 42, no. 1, pp. 88 – 96, jan 2004.

[2] D. Waddington and F. Chang, “Realizing the transition to IPv6,” IEEE Commun. Mag., vol. 40, no. 6, pp. 138 –147, jun 2002.

[3] A. Durand, “Deploying IPv6,” IEEE Internet Computing, vol. 5, no. 1, pp. 79–81, jan/feb 2001.

[4] J. J. Amoss and D. Minoli, Handbook of IPv4 to IPv6 Transition: Methodologies for Institutional and Corporate Networks. Auerbach Publications, 2007.

[5] M. Mackay and C. Edwards, “A Managed IPv6 Transitioning Architecture for Large Network Deployments,” IEEE Internet Computing, vol. 13, no. 4, pp. 42 –51, july-aug. 2009.

[6] Silva Hagen, IPv6 Essentials, 2nd Edition ed. O’Reilly Media, 2006.

[7] R. Hinden and S. Deering, “IP Version 6 Addressing Architecture,” 2006, IETF RFC 4291.