This page provides an overview of multiple network interfaces in a virtual machine (VM) instance, including how they work and sample configurations. For information about creating configurations that use multiple interfaces, see Creating multiple network interfaces (/vpc/docs/create-use-multiple-interfaces).

Google Cloud Virtual Private Cloud (VPC) networks are by default isolated private networking domains. Networks have a global scope (/compute/docs/regions-zones/global-regional-zonal-resources) and contain regional subnets. VM instances within a VPC network can communicate among themselves via internal IP addresses as long as firewall rules permit. However, no internal IP address communication is allowed *between* networks, unless you set up mechanisms such as VPC Network Peering (/vpc/docs/vpc-peering) or Cloud VPN (/vpn/docs).

Every instance in a VPC network has a default network interface. You can create additional network interfaces attached to your VMs. Multiple network interfaces enable you to create configurations in which an instance connects directly to several VPC networks. Each of the interfaces must have an internal IP address, and each interface can also have an external IP address. Each instance can have up to 8 interfaces, depending on the instance's type. For more information, see Maximum number of interfaces (/vpc/docs/create-use-multiple-interfaces#max-interfaces).

Typically, you might require multiple interfaces if you wish to configure an instance as a network appliance that does load balancing, Intrusion Detection and Prevention (IDS/IPS), Web Application Firewall (WAF), or WAN optimization between networks. Multiple network interfaces are also useful when applications running in an instance require traffic separation, such as separation of data plane traffic from management plane traffic.

Use multiple network interfaces when an individual instance needs access to more than one VPC network, but you don't want to connect both networks directly.

- **Network and security function**: Multiple network interfaces enable virtualized network appliance functions such as load balancers, network address translation (NAT) servers, and proxy servers that are configured with multiple network interfaces. See Example 1: Networking and security virtual appliances (#config-appliances) for more details.

- **Perimeter and DMZ isolation**: An important best practice in tiered networking architectures is to isolate public-facing services from an internal network and its services. Use multiple network interfaces to create configurations where there are separate network interfaces on the instance, one of them accepting public-facing traffic and another handling back-end private traffic that has more restrictive access controls.

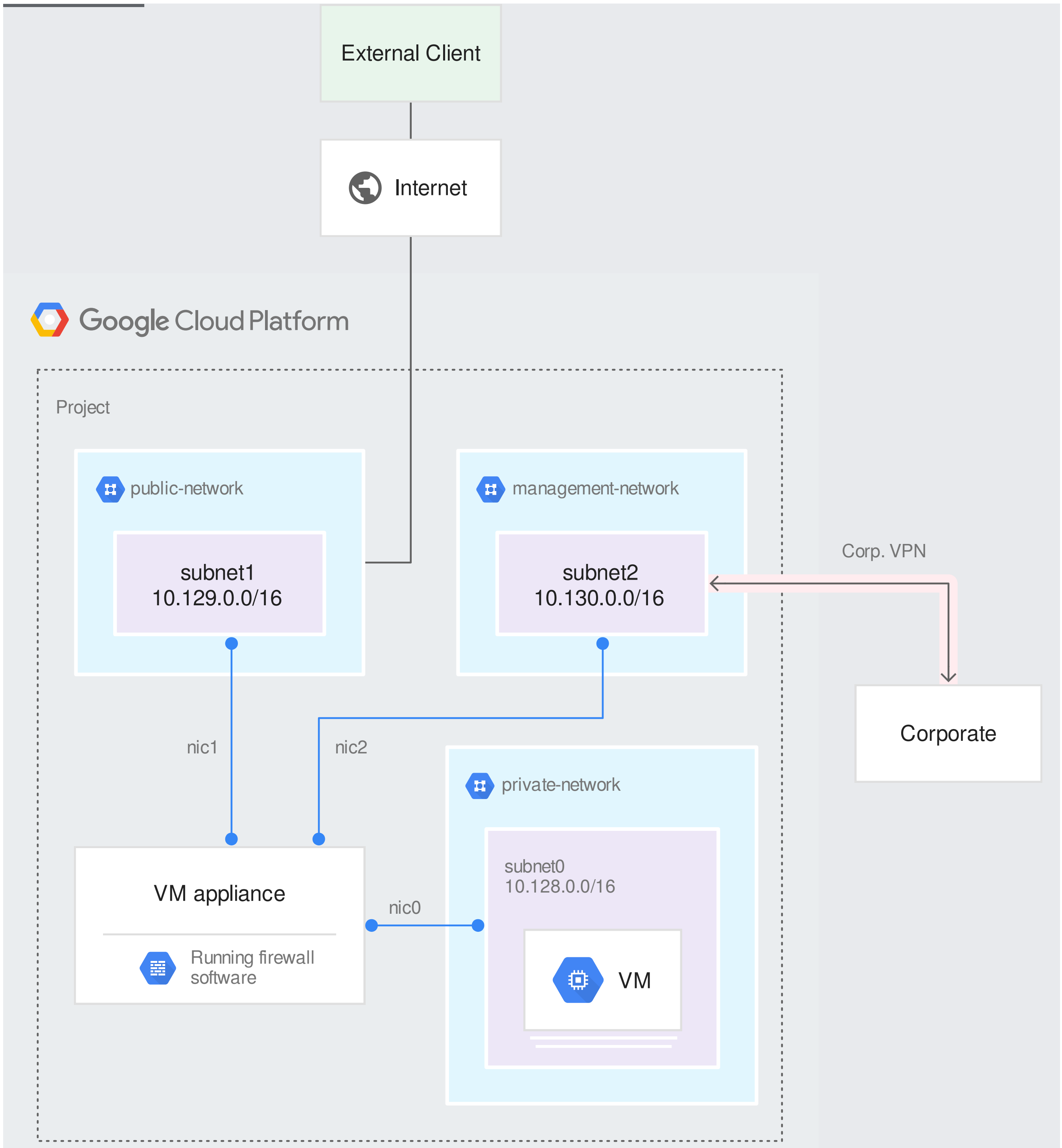
  Any resources that can be reached from the Internet should be separated from your internal network and its services. This drastically limits the scope and damage that a security breach can cause. For example, you can place a second network interface on each web server that connects to a mid-tier network where an application server resides. The application server can also be dual-homed to a back-end network where the database server resides. Each dual-homed instance receives and processes requests on the front end, initiates a connection to the back end, and then sends requests to the servers on the back-end network.

  By configuring separate interfaces, one public-facing and another private- facing, you can apply separate firewall rules and access controls to each interface separately and enforce security functions in communications from the public to private domain. For more information, see Example 2: Using third-party appliances in a Shared VPC network scenario (#third-party).

This section examines several common examples of how to use multiple network interfaces.

Networking and security virtual appliances, such as WAF, security application- level firewalls, and WAN accelerators, are usually configured with multiple virtual interfaces. Each of the multiple interfaces is configured with its own private IP address and optionally with its own public IP address.

The following figure describes a typical setup. In this specific case, you configure a virtual network appliance on the path from public to private connectivity. In this way, traffic can only reach a private VPC network from a public external client through an application-level virtualized firewall enforcement point. This application-level firewall is enforced on top of virtual machines.



(/vpc/images/multinic/multinic1.svg)

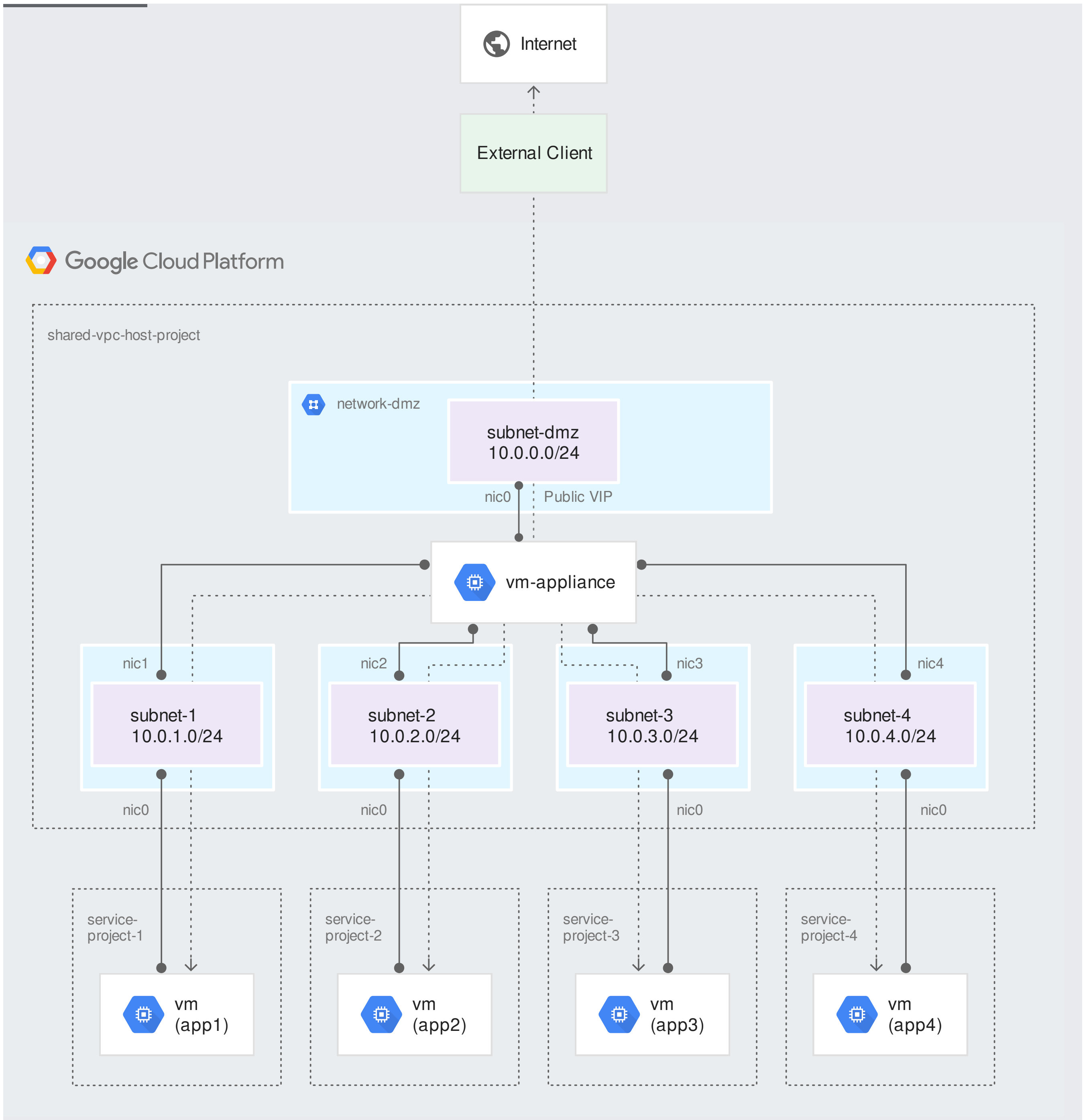
Use case 1: Provisioning and configuring instances with multiple interfaces (click to enlarge)

The following assumes that `subnet0`, `subnet1`, and `subnet2` already exist, with non-overlapping ranges. To configure VM-appliance in this example, use the following command.

This command creates an instance with three network interfaces:

- `nic0` is attached to subnet0 and has no public IP address
- `nic1` is attached to subnet1 and has an ephemeral public IP address
- `nic2` is attached to subnet2 and has no public IP address

This setup is useful when you want to share a single set of centralized third-party appliances for workloads or applications that are hosted in different projects. In the example shown below, there are four distinct applications, App1, App2, App3 and App4, that are hosted in different service projects. You need them to be protected for all Internet ingress and you need egress traffic to be inspected and filtered in a third-party appliance that is centrally located in the Shared VPC host project.



(/vpc/images/multinic/multinic3.svg)
Use case 2: Shared VPC example with a third-party appliance (click to enlarge)

To create the VM and network interfaces in this example, use the following commands.

To create VM-appliance:

This creates an instance with five network interfaces:

- `nic0` is attached to subnet-dmz, which is part of network-dmz, with a static address 'reserved-address'
- `nic1` is attached to subnet-1, which is part of network-1, with no public IP
- `nic2` is attached to subnet-2, which is part of network-2, with no public IP
- `nic3` is attached to subnet-3, which is part of network-3, with no public IP
- `nic4` is attached to subnet-4, which is part of network-4, with no public IP

Shared VPC (/vpc/docs/shared-vpc) enables you to share VPC networks across projects in your Cloud Organization.

Shared VPC allows you to create instances associated with a Shared VPC network that is hosted in a centralized Shared VPC host project. See Provisioning Shared VPC (/vpc/docs/provisioning-shared-vpc) for full information on configuring Shared VPC networks.

When you create instances with multiple network interfaces, your instances or instance templates can have certain interfaces attached to subnets local to the project, while other interfaces can be attached to Shared VPC networks.

To create instances with one or more interfaces associated with Shared VPC networks, you must have the `compute.networkUser` role in the Shared VPC host project.

When an internal DNS query is made with the instance hostname, it resolves to the primary interface (`nic0`) of the instance. If the `nic0` interface of the instance belongs to a VPC network different from the VPC network of the instance issuing the internal DNS query, the query will fail.

Private Compute Engine DNS records will not be generated per interface.

In a default multiple interface configuration, the OS is configured to use DHCP. The DHCP and ARP behavior of each of the multiple interfaces is the same as the DHCP and ARP in an instance with a single interface.

In a multiple interface instance that uses DHCP, every interface gets a route for the subnet that it is in. In addition, the instance gets a single default route that is associated with the primary interface `eth0`. Unless manually configured otherwise, any traffic leaving an instance for any destination other than a directly connected subnet will leave the instance via the default route on `eth0`.

In this sample, the primary interface `eth0` gets the default route (`default via 10.138.0.1 dev eth0`), and both interfaces `eth0` and `eth1` get routes for their respective subnets.

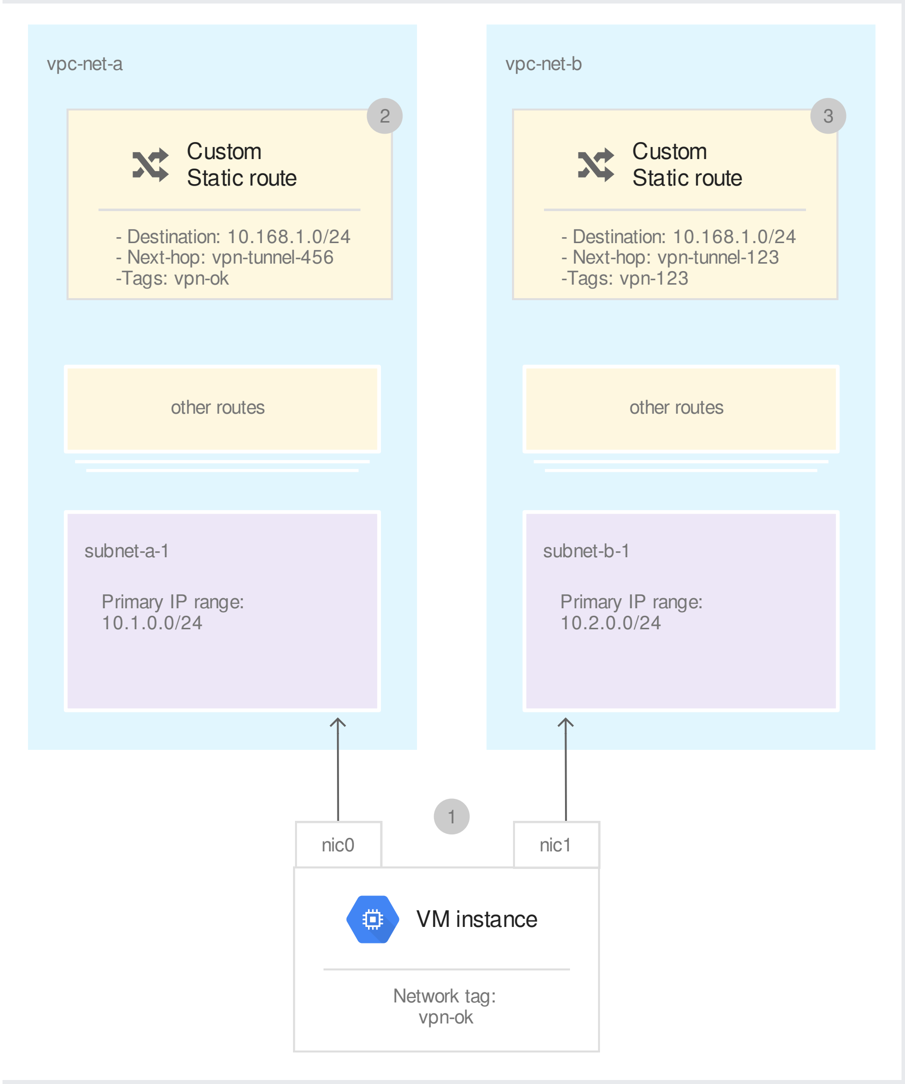
For more information, read Configuring Policy Routing (/vpc/docs/create-use-multiple-interfaces#configuring_policy_routing).

When a VM instance has multiple interfaces and a network tag (/vpc/docs/add-remove-network-tags), the network tag might not impact all of the VM's interfaces. A VM's network tag impacts an interface if the interface is in a VPC network that contains a custom static route (/vpc/docs/routes#static_routes) with a matching tag.

For example:

1. A VM has two interfaces: `nic0` and `nic1`. The `nic0` interface is in `vpc-net-a`. The `nic1` interface is in `vpc-net-b`. The VM has a network tag called `vpn-ok`. The tag is an attribute on the instance, not on a specific interface.
2. The `vpc-net-a` network has a custom static route with a tag called `vpn-ok`.
3. The `vpc-net-b` network has a custom static route with a tag called `vpn-123`.

These numbered steps correspond to the number callouts in the following diagram:



(/vpc/images/network-tags-multiple-interfaces.svg)
Custom Static Routes and Mltiple Network Interfaces (click to enlarge)

In the case of the `vpc-net-a` network, because it has a route with a tag in common with the VM, the VM's `vpn-ok` tag applies to the VM's `nic0` interface in `vpc-net-a`. In contrast, because the `vpc-net-b` doesn't have a static route with the `vpn-ok` tag, the VM's `vpn-ok` network tag is ignored on the VM's `nic1` interface.

If you choose to use tags with routes, note that tags are applied at the instance level and, as such, tags apply to all interfaces of a virtual machine instance. If this is not desireable, you set up your configuration so that only certain tags are used in routes in a given VPC network, effectively ensuring those tags only apply to the interfaces associated with the specific VPC network.

Except for Internal TCP/UDP Load Balancing (/load-balancing/docs/internal/), all Google Cloud load balancers only distribute traffic to the first interface (`nic0`) of a backend instance.

Each VPC network has its own set of firewall rules. If an instance's interface is in a particular VPC network, that network's firewall rules apply to that interface.

For example, suppose a VM instance has two interfaces:

- `nic0` in VPC network `network-1`
- `nic1` in VPC network `network-2`

Firewall rules that you create for the `network-1` network apply to `nic0`. Firewall rules that you create for the `network-2` network apply to `nic1`.

See to the firewall rules overview (/vpc/docs/firewalls) for more information.

- Ingress firewall rules can use either network tags or service accounts to identify sources, targets (destinations), or both.
- Egress firewall rules can use either network tags or service accounts to identify targets (sources).

See source and target filtering by service account (/vpc/docs/firewalls#serviceaccounts) for more information.

Network tags and service accounts identify *instances*, not specific interfaces. Keep in mind that firewall rules are associated with a single VPC network and each interface of a multi-NIC instance must be in a unique VPC network.

The following example demonstrates how you can effectively use source tags for ingress `allow` firewall rules. The `vm1` instance has two network interfaces:

- `nic0` in `network-1`
- `nic1` in `network-2`

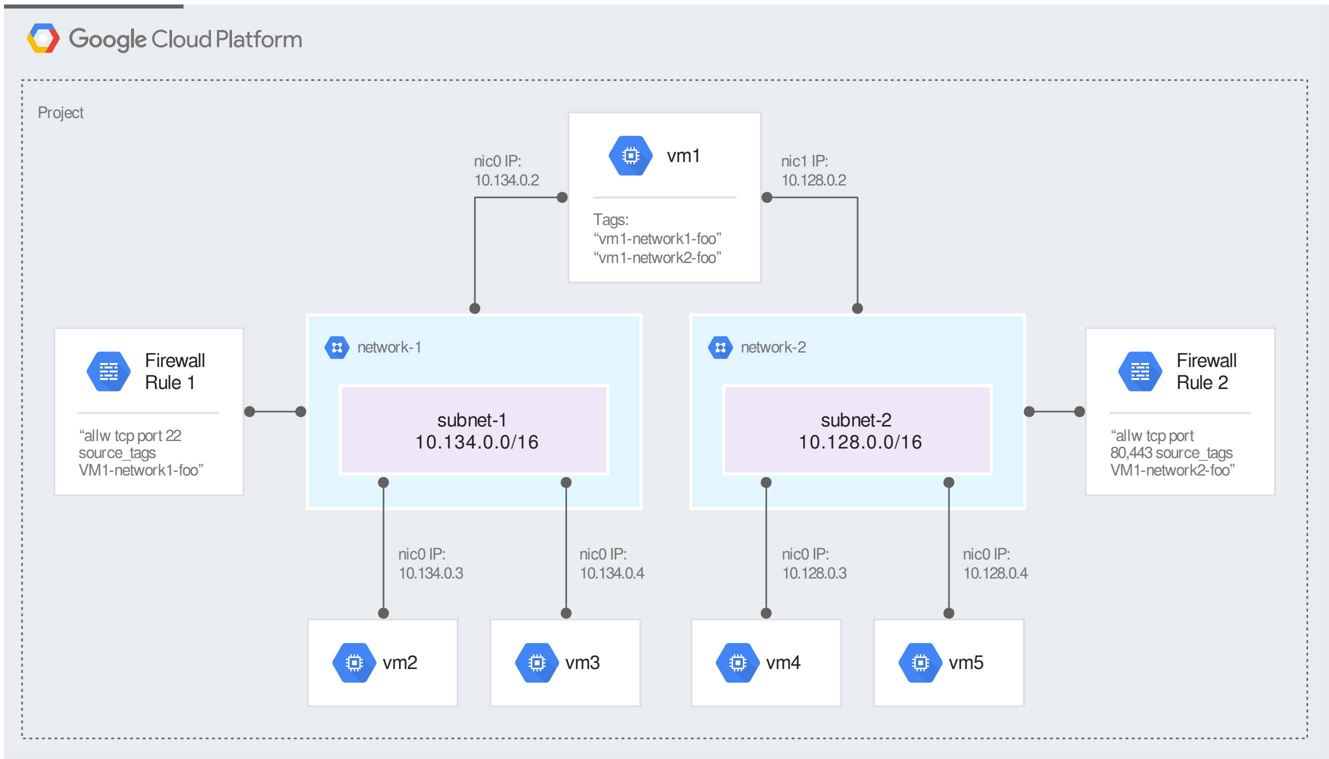
Suppose you need to allow the following traffic from `vm1`:

- SSH traffic from `vm1` to any instance in `network-1`
- HTTP and HTTPS traffic from `vm1` to any instance in `network-2`

To accomplish this, you can do the following:

1. Assign two network tags (/vpc/docs/add-remove-network-tags#adding_new_tags_to_vm_instances) to `vm1`: `vm1-network1` and `vm1-network2`
2. Create (/vpc/docs/using-firewalls#creating_firewall_rules) an ingress `allow` firewall rule in `network-1` with the following components (/vpc/docs/firewalls#gcp_firewall_rule_summary_table) to allow SSH traffic from `vm1` to all VMs in `network-1`:
    - Action: `allow`
    - Direction: `ingress`
    - Sources: VMs with tag `vm1-network1`
    - Targets: All instances in the VPC network
    - Protocols and ports: `tcp:22`
3. Create an ingress allow firewall rule in `network-2` with the following components to allow HTTP and HTTPS traffic from `vm1` to all VMs in `network-2`:
    - Action: `allow`
    - Direction: `ingress`
    - Sources: VMs with tag `vm1-network2`
    - Targets: All instances in the VPC network
    - Protocols and ports: `tcp:80,443`

The following diagram illustrates this firewall configuration example:



(/vpc/images/multinic/multinic5.svg)

Firewall rules (click to enlarge)

- Read Creating instances with multiple network interfaces (/vpc/docs/create-use-multiple-interfaces).