**Cloud Networking Basics (How Traffic Flows in the Cloud)**

When deploying an application in the cloud, **networking is crucial** for ensuring secure and efficient data flow.

**🖥️ 1. Virtual Private Cloud (VPC)**

* A **VPC is like a private data center in the cloud** where you host your servers, databases, and applications.
* **A VPC (Virtual Private Cloud) is a logically isolated private network inside a cloud provider (AWS, GCP, Azure). It allows you to create and manage subnets, firewall rules, routing, and networking to control how your resources communicate internally and externally.**
* You define **subnets** inside the VPC to separate different components (e.g., one for backend, one for the database). Public subnet and private subnet.

**IT Examples:**

* You have a **web application** hosted on Google Cloud. You create a **VPC** to keep internal communication secure.
* Inside this VPC, you create **two subnets**:
  1. **Public Subnet** (for web servers exposed to the internet)
  2. **Private Subnet** (for databases and backend services)

✅ **Example**:

* Your Java backend runs on **Compute Engine inside a private VPC**, ensuring it’s not exposed to the public internet

✅ **Key Benefits:**  
✔️ Secure and isolated network  
✔️ Full control over traffic routing  
✔️ Can connect to on-prem data centers

**Alternative: Using Load Balancer for Scalability**

Instead of directly exposing your VM, you can place it behind a **Google Cloud HTTP Load Balancer**, which:

* Distributes traffic to multiple VMs (if needed).
* Offers **global** high availability.
* Provides **auto-scaling**.
* Supports **SSL termination** for HTTPS.

**Internal Cloud Communication (Within Google Cloud)**

Cloud services communicate with each other internally in different ways.

**A. Private Communication within Google Cloud (No Public Internet)**

* Google Cloud **VPC (Virtual Private Cloud)** provides a private **internal network**.
* Services within the same **VPC** can communicate using **private IPs**.
* This is **secure** and does not expose services to the public internet.

✅ **Example: Compute Engine → Cloud SQL (MySQL/PostgreSQL)**

* If your application on Compute Engine needs to talk to **Cloud SQL**, you can use:
  + **Private IP** → Connect directly using internal networking.
  + **Cloud SQL Proxy** → Securely connect without exposing database credentials.

**B. Microservices Communication**

If your application consists of **multiple services**, they can communicate using:

1. **Internal Load Balancer** → Used for private service-to-service communication.
2. **Service Networking (VPC Peering)** → Connects multiple VPCs securely.

**Example: Frontend React App → Backend FastAPI on Compute Engine**

* Frontend (React) calls API hosted on FastAPI running on Compute Engine.
* API exposes an internal endpoint like http://backend-service.internal.
* A **Private Load Balancer** routes traffic internally without internet exposure.

**🛡️ 2. Cloud Firewall (Security Rules)**

**Firewalls** define what traffic can enter or leave your VPC. A **firewall** controls **which traffic is allowed or blocked** in your cloud network.

* You set **allow/deny rules** to **protect your application from unauthorized access**.

✅ **Example**:

* A firewall rule allows **only HTTP and HTTPS traffic** from the internet to your **Load Balancer**, preventing unauthorized SSH access.

**Real-World Scenario: Security Gate**

* An office **security gate** allows **authorized employees** but **blocks outsiders**.
* Similarly, **firewall rules** allow, or block traffic based on **IP addresses, ports, or protocols**.

**Example Use Case**

* A company wants to **allow** only internal users to access their **database**.
* They configure **firewall rules**:
  + Allow **only VPC traffic** on **port 5432 (PostgreSQL)**.
  + Deny **all external traffic**.

**🌐 3. Cloud Load Balancer (ILB & External LB)**

* **Internal Load Balancer (ILB)** distributes traffic between services inside the VPC.
* **External Load Balancer** handles internet-facing traffic and routes it to backend instances.

✅ **Example**:

* A Java app runs on multiple **Compute Engine VMs** behind an **external load balancer**, ensuring high availability.

**🔗 4. Cloud DNS (Domain Resolution)**

* **Cloud DNS maps human-readable domain names (e.g.,** [**www.mysite.com**](http://www.mysite.com)**) to IP addresses**.

 A company hosts a web app on **Compute Engine** (34.120.50.1).

✅ **Example**:

* Instead of users typing an IP (34.123.45.67), they type www.mysite.com, and **Cloud DNS** resolves it to the correct IP.

✅ **Key Benefits:**  
✔️ Protects private resources  
✔️ No need for public IPs  
✔️ Secure outbound internet access

**Hybrid Communication (On-Prem to Cloud)**

If you have **on-premises servers** and want them to connect to your cloud services, you can use:

* **Cloud VPN** → Secure tunnel between on-premises and GCP.
* **Interconnect** → High-speed dedicated fiber connection.
* **Cloud NAT** → Allows private VMs to access the internet without exposing their IPs.

**🚀 4. Cloud NAT (Secure Internet Access)**

* **Cloud NAT allows VMs inside a private VPC to access the internet securely** (e.g., fetching software updates, calling APIs) **without exposing their private IPs**.

✅ **Example**:

* A **Compute Engine VM inside a private subnet** needs to access a **payment gateway API**, so it uses **Cloud NAT** to securely communicate with external services.

**🔌 Cloud VPN & Interconnect**

* **Cloud VPN** securely connects an on-premises data center to the cloud.

✅ **Example**:

* A bank wants to connect its **on-premise PostgreSQL database** to a **GCP Compute Engine backend** securely using **Cloud VPN**.

**Cloud VPN** creates a **secure tunnel** between your **on-premises data center** and **Google Cloud/AWS** over the **public internet** using **IPsec encryption**.

 A bank has a **local data center** running a **customer database**.

 They want to **migrate** applications to Google Cloud but still need the **on-prem database**.

 They set up **Cloud VPN** to securely connect **on-prem data** with **GCP applications**.

✅ **Key Benefits:**  
✔️ Secure encrypted traffic  
✔️ Extends on-prem network into the cloud  
✔️ Easy to set up

**What is Interconnect?**

**Cloud Interconnect** provides a **dedicated** high-speed private connection between your **on-prem data center** and **Google Cloud/AWS**, bypassing the public internet.

**Real-World Scenario: Private Fiber Connection**

* Imagine your company has offices in **two cities**.
* Instead of using the **public internet**, you set up a **private fiber line** for **faster and more secure** data transfer.
* Similarly, **Interconnect** provides a **direct** private link between your **on-prem and cloud VPC**.

**🔗 Cloud Interconnect Use Case**

**Scenario: On-Prem ERP System Needs to Access GCP Databases**

**Company:** A **retail chain** with a **centralized ERP system** running **on-prem** needs to connect to a **Cloud SQL PostgreSQL** instance in GCP.

* The ERP needs **low-latency, high-speed connectivity**.
* A **VPN over public internet is too slow & unreliable**.
* **Solution:** Use **Cloud Interconnect** (Dedicated or Partner) to create a **direct private link** between on-prem and GCP.

🔹 **Traffic Flow:**  
1️⃣ **On-prem ERP System → Cloud Interconnect → GCP Cloud SQL**  
2️⃣ **Database queries execute faster (~1-2 ms latency)**  
3️⃣ **Secure, high-speed connection without public internet risks**

✅ **Key Benefits:**  
✔️ High-speed private connection  
✔️ More reliable than Cloud VPN  
✔️ Avoids public internet latency

**When to Use What?**

* **Use Cloud NAT** → When private VMs need **outbound internet access** (APIs, updates, etc.).
* **Use Cloud Interconnect** → When **on-prem needs a private, high-speed connection** to GCP.

**How Does Your Application on GCP Compute Engine Connect to the Internet?**

When you deploy an application on a Google Cloud Compute Engine (VM), there are multiple components involved to make it accessible from a browser.

**Step-by-Step Flow**

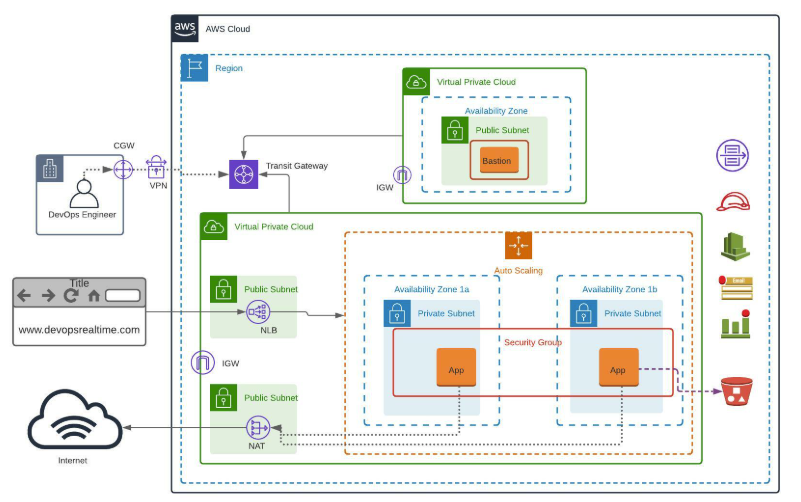
1. **User opens a browser** → Enters your application URL (e.g., https://myapp.example.com).
2. **DNS Resolution** → The browser queries **Google Cloud DNS** (or another DNS provider) to resolve the domain name into an **IP address**.
3. **Request Reaches GCP Network** → The request is routed through the **Google Cloud Load Balancer** or directly to your **Compute Engine VM** (if using a static IP).
4. **Firewall Rules Allow Traffic** → Your VM **must** have an **open firewall rule** to allow HTTP (port 80) or HTTPS (port 443) traffic.
5. **Application Processes the Request** → Your backend (e.g., Python FastAPI, Node.js, or Java) handles the request and sends a response.
6. **Response Sent Back to User** → The response travels back over the internet to the user’s browser.

**Example Setup for External Access**

✅ **Public IP Address** → Attach a static external IP to your VM.  
✅ **Firewall Rules** → Allow ingress traffic on ports **80 (HTTP)** or **443 (HTTPS)**.  
✅ **DNS Mapping** → Use Google Cloud DNS to map a domain name to the VM's IP.  
✅ **SSL Certificate** → Use **Google-managed SSL** for secure access.

A diagram of a computer server

AI-generated content may be incorrect.



A diagram of a computer

AI-generated content may be incorrect.