# 1. ¿Qué es una red (LAN, WAN e Internet)?

Una **red informática** es un conjunto de dispositivos (ordenadores, servidores, móviles, etc.) interconectados para compartir información y recursos. Cada equipo en la red tiene una **dirección IP** única, como su "domicilio" digital[1][2]. Las redes se clasifican por su alcance:

- **LAN (Local Area Network)**: red de área local, limitada a un espacio pequeño (casa, oficina, aula). Permite compartir recursos cercanos (impresoras, archivos) con baja latencia[3].
- **WAN (Wide Area Network)**: red de área amplia, conecta múltiples LAN geográficamente dispersas (por ejemplo, sedes de una empresa en diferentes ciudades)[4][3].
- **Internet**: la WAN más grande del mundo –la "red de redes"– que interconecta todas las demás redes globalmente[4][5]. Por ejemplo, tu portátil en una LAN hogareña llega a Internet mediante el router de tu proveedor de servicios (ISP), y una instancia de AWS en una VPC puede llegar a Internet a través de una **Internet Gateway** (puerta de enlace de Internet) en su VPC[6][7].

**Ejemplo:** Imagina que tu casa es una LAN: tus dispositivos (PC, móvil, impresora) se conectan al router de casa. Ese router se enlaza al ISP (WAN). Internet conecta tu ISP con AWS y cualquier otro servidor en el mundo. Así, cuando tu PC accede a Google, envía datos por la LAN, luego por la WAN del ISP, y finalmente por la WAN global de Internet hasta los servidores de Google. Sin conexión de red (ni local ni global) no podría intercambiar datos.

# 2. Modelo OSI y dispositivos básicos

El **modelo OSI** es un marco conceptual que estandariza las funciones de red en **7 capas** jerárquicas[8]. Cada capa realiza tareas específicas para enviar datos de un punto a otro. Las capas OSI (de abajo hacia arriba) son:
1. **Capa 1 – Física:** transmite bits por medios físicos (cables, fibra, inalámbrico). Equipos: *hubs*, repetidores, cables, tarjetas de red físicas.
2. **Capa 2 – Enlace de datos:** organiza los bits en tramas y gestiona las direcciones MAC. Equipos: *switches*, puentes (bridges). Un switch opera en capa 2; envía tramas solo al puerto correcto según la dirección MAC.
3. **Capa 3 – Red:** enruta paquetes entre redes usando direcciones IP. Equipos: *routers*, gateways. Por ejemplo, un router conecta tu LAN doméstica con Internet y enruta paquetes basados en la IP de destino.
4. **Capa 4 – Transporte:** segmenta los datos y garantiza la comunicación fi able (o no) de extremo a extremo. Protocolos: **TCP** (orientado a conexión, entrega garantizada en orden) y **UDP** (sin conexión, rápido pero sin garantía).
5. **Capa 5 – Sesión:** administra sesiones y diálogos entre aplicaciones (inicia, controla y cierra conexiones lógicas).
6. **Capa 6 – Presentación:** formatea los datos (cifrado, compresión, codificación de caracteres)

para la capa superior.
7. **Capa 7 – Aplicación:** interfaz con el usuario y las aplicaciones (HTTP, SMTP, SSH, etc.).

> *"El modelo OSI contiene siete capas que se apilan (conceptualmente) de abajo a arriba. Las capas OSI son: física, enlace de datos, red, transporte, sesión, presentación y aplicación."*[8]

**Ejemplo de mapeo de dispositivos:** Un **hub** simplemente repite la señal eléctrica a todos sus puertos (capa 1), un **switch** reenvía tramas según MAC (capa 2), y un **router** reenvía paquetes según IP (capa 3). Esto permite aislar dominios de colisión (hub) y dominios de broadcast/red (switch) y conectar redes diferentes (router).

**Ejercicio:** Relaciona cada dispositivo con su capa OSI: - *Hub:* Capa 1 (física) – repite bits.
- *Switch:* Capa 2 (enlace) – filtra tramas por dirección MAC.
- *Router:* Capa 3 (red) – dirige paquetes entre redes IP.
- *Servidor web (proceso Apache):* Capa 7 (aplicación) – atiende peticiones HTTP (por ejemplo en el puerto 80).

# 3. Direcciones IP

Una **dirección IP** identifica de forma única a un dispositivo en la red[2][9]. En IPv4, se representa como cuatro números decimales separados por puntos (por ejemplo, **192.168.1.100**), cada uno entre 0 y 255. Cada dirección IP tiene dos partes: la **parte de red** (identifica la red a la que pertenece) y la **parte de host** (identifica el dispositivo en esa red)[10][9].

Analogía postal: la IP es como tu dirección de casa. Cada paquete de datos incluye la IP de origen y destino (como el remitente y destinatario en una carta) para que llegue al lugar correcto[10]. Si tu instancia AWS tiene IP 18.205.4.100 y te conectas por SSH, tu cliente envía datos a esa IP; los routers intermedios encaminan el paquete basándose en ella hasta llegar a AWS[9][10].

**IPs públicas vs. privadas:** Las **IPs públicas** son únicas globalmente y enrutable en Internet. Permiten que dispositivos externos (como tu portátil desde casa) lleguen directamente a ese equipo. Las **IPs privadas** solo son válidas dentro de una red local (LAN) y no enrutan en Internet; son definidas por el RFC1918[11]. Ejemplos de rangos privados comunes son 10.0.0.0/8, 172.16.0.0/12 y 192.168.0.0/16[11] (por ejemplo, 192.168.0.x en la mayoría de routers hogareños).

- **Ejemplo:** Una PC en tu LAN hogareña puede tener IP **192.168.1.10** (privada) y el router puede tener IP pública **20.35.50.10**. Al navegar, tu tráfico sale con la IP pública del router (gracias al NAT) para alcanzar Internet, pero en tu LAN se comunica como 192.168.1.10.

En AWS, cada instancia recibe siempre una **IP privada interna** (por ejemplo 10.0.1.25) para comunicarse dentro de su VPC. Si la instancia está en una subred pública configurada para ello, AWS también le asigna una **IP pública** (por ejemplo 54.85.13.20) para que sea accesible desde Internet[7][5]. La IP privada se mantiene mientras la instancia exista; la IP pública "saliente" de por sí puede cambiar al detener/iniciar la instancia (a menos que sea una *Elastic IP* fija)[7][12].

Si una instancia no tiene IP pública (por diseño en una subred privada), **nadie en Internet** puede conectarse directamente a ella.

**Ejercicio:** Supón una instancia EC2 con IP privada *10.0.2.15* en una VPC. Indica si es accesible desde Internet: - Si no tiene IP pública ni NAT, *no* es accesible desde Internet.
- Si le asignas una IP pública, *sí* será accesible (siempre que su subred tenga una Internet Gateway y el firewall lo permita)[7][12].

# 4. Máscara de subred y subnetting

Una **subred** es una porción más pequeña de una red mayor[13]. Se usa para organizar dispositivos, aislar tráfico y enrutarlo eficientemente. Al crear subredes, los routers conocen rangos específicos por cada segmento, evitando rutas innecesarias[13].

**Máscara de subred:** Define qué parte de la IP corresponde a la red y qué parte a los hosts. Por ejemplo, la notación **/24** equivale a la máscara 255.255.255.0 y dice que los primeros 24 bits (3 octetos) son la red. Con 192.168.1.0/24, la subred abarca las IP 192.168.1.0 a 192.168.1.255, de las cuales 192.168.1.1–192.168.1.254 son hosts útiles. Otra subred podría ser 192.168.2.0/24, etc.

En binario, 255.255.255.0 = 11111111.11111111.11111111.00000000. Los 1s marcan los bits de red, los 0s los de host. Cuantos más bits sean 1 (por ejemplo /26 = 255.255.255.192 = 11111111.11111111.11111111.11000000), más pequeño es el rango de hosts. En /26 cabe 64 direcciones (62 hosts útiles + 1 red + 1 broadcast).

**Ejercicio (Subnetting básico):**
- Parte la red 192.168.1.0/24 en subredes /26 (máscara 255.255.255.192): ¿Qué subredes obtienes?
- Las subredes son: 192.168.1.0/26, 192.168.1.64/26, 192.168.1.128/26, 192.168.1.192/26.
- Cada /26 tiene 64 direcciones: e.g. para 192.168.1.64/26 el rango de hosts es 192.168.1.65–192.168.1.126 (192.168.1.64 es red, 192.168.1.127 broadcast).

- **Solución de ejemplo:** Dada la IP **192.168.1.130/26**, determine la red, broadcast y rango de hosts:
- La máscara /26 (255.255.255.192) indica subredes de 64 direcciones. 130 cae en la subred **192.168.1.128/26** (que va de .128 a .191).
- Dirección de red: **192.168.1.128**. Broadcast: **192.168.1.191**. Hosts: 192.168.1.129–192.168.1.190.

**Ejemplo ASCII (división de subredes):**

```
Red original: 10.10.0.0/24
Se divide en dos /25 (128 direcciones cada una):
  - 10.10.0.0   /25 (hosts 10.10.0.1 – 10.10.0.126)
  - 10.10.0.128 /25 (hosts 10.10.0.129 – 10.10.0.254)
O en cuatro /26 (64 direcciones cada una):
  - 10.10.0.0   /26 (hosts .1 – .62)
```

```
- 10.10.0.64  /26 (hosts .65 – .126)
- 10.10.0.128 /26 (hosts .129 – .190)
- 10.10.0.192 /26 (hosts .193 – .254)
```

En **AWS VPC**, al crear la red virtual (VPC) defines un rango grande (por ejemplo 10.0.0.0/16). Dentro, subdivides en *subnets* más pequeñas (por ejemplo 10.0.1.0/24 en AZ A pública, 10.0.2.0/24 en AZ A privada, etc.). AWS requiere asignar cada subred a una Zona de Disponibilidad. En la práctica, suele haber subredes públicas (con ruta a Internet Gateway) y privadas (sin acceso directo externo)[14][12]. Mantener bases de datos en subredes privadas sin IP pública es buena práctica: no las ve Internet, solo acceden internamente (o vía VPN).

# 5. NAT e Internet en AWS

Los dispositivos con **IP privadas** necesitan alguna forma de salir a Internet. En redes hogareñas, el router hace **NAT (Network Address Translation)**: sustituye la IP privada de tu PC por su IP pública al enviar tráfico[15][16]. Esto permite que varios dispositivos compartan una sola IP pública (y los puertos en NAT mantengan las sesiones). El NAT **impide conexiones entrantes no solicitadas**; funciona como un portero: deja salir a la LAN respuestas a peticiones, pero no deja entrar conexiones externas sin invitación.

En AWS, el concepto es similar. Una instancia **sin IP pública** en una subred privada *no puede* acceder directamente a Internet (no tiene puerta de enlace pública). Para darle salida, colocas un **NAT Gateway** (o NAT Instance) en una subred pública. Luego, en la tabla de rutas de la subred privada defines la ruta 0.0.0.0/0 apuntando al NAT Gateway. Al hacerlo, las instancias privadas pueden iniciar conexiones salientes: el NAT Gateway sustituye sus IP privadas por la suya pública (y reenvía las respuestas)[15][17]. De esta forma, sí pueden actualizar paquetes, acceder a APIs, etc., pero siguen **no accesibles desde Internet** (nadie puede iniciar sesión directamente en ellas, ya que no tienen IP pública ni ruta entrante).

Por otra parte, las instancias en **subred pública** reciben una ruta directa a una **Internet Gateway (IGW)** asociada a la VPC[6][7]. El IGW es un componente escalable que permite tráfico bidireccional entre la VPC e Internet. Requiere que la instancia tenga IP pública (o Elastic IP) y que su tabla de rutas envíe 0.0.0.0/0 al IGW[6][7]. Con esto, la instancia puede *entrar y salir* de Internet (según las reglas de firewall). Sin un IGW no hay acceso externo en la VPC, ni saliente (aparte de VPC a VPC)[6][18].

**Resumen rápido:**
- **NAT Gateway:** las instancias privadas *salen* a Internet (inbound bloqueado)[15].
- **IGW + IP pública:** instancias en subred pública *entran y salen* a Internet (según reglas)[7][17].

**Ejercicio:** En tu VPC tienes:
- Subred A (pública) con IGW, y Subred B (privada) con ruta al NAT.
- Instancia Web en A con IP pública **3.85.24.117**.
- Instancia BD en B sin IP pública.

¿Puede la instancia Web navegar por Internet? ¿Puede la BD hacerlo?
- *Web (subred pública):* Sí, porque está en subred pública, con IGW y IP pública.

- *BD (subred privada):* Sí puede navegar *saliente* gracias al NAT Gateway configurado (recibe las respuestas), pero *no es accesible* desde Internet (no tiene IP pública ni IGW directo).

Si te olvidas de crear el NAT y lanzas una instancia en la subred privada, de fábrica **no podrá hacer ping a 8.8.8.8 ni apt-get update**, pues no hay ruta saliente. Este es un problema típico: el estudiante ve "no hay Internet" en la VM privada porque falta el NAT o IGW. La frase "mi instancia no tiene salida a Internet" suele significar eso: está aislada, sin IGW/NAT correctamente configurado.

# 6. DHCP (IP dinámica)

**DHCP** (Dynamic Host Configuration Protocol) es el protocolo que asigna automáticamente direcciones IP y otros parámetros de red a los dispositivos al conectarse[19]. Funciona como un "conserje" de la red: cuando un equipo nuevo entra, el servidor DHCP le da una IP libre, máscara de subred, puerta de enlace predeterminada y servidores DNS[19]. Esto evita tener que configurar manualmente cada IP.

En AWS, el servicio de VPC incluye un servidor DHCP interno. Al lanzar una instancia, AWS le asigna automáticamente una IP privada del rango de la subred (y opcionalmente una IP pública) [19]. Esa IP privada "alquilada" permanece con la instancia mientras esté en ejecución. Si la apagas y enciendes, AWS suele conservarla; si la terminas, vuelve al pool. Las IP públicas dinámicas (no-Elastics) también se gestionan por DHCP: al reiniciar la instancia, puede obtener una IP pública distinta (de ahí la recomendación de usar Elastic IP para una dirección pública fija).

> *"DHCP garantiza que a todos los dispositivos se les asigne una dirección IP [y otros parámetros de red][19]."*

**Ejemplo práctico:** Te conectas al WiFi de casa y ves que tu móvil tiene IP 192.168.1.101. No la configuraste manualmente: fue DHCP del router. Si desconectas el móvil y lo conectas al rato, quizás ahora sea 192.168.1.105 (el DHCP le dio una distinta).

En entornos locales, si un dispositivo no obtiene DHCP, suele autoconfigurarse con IP de rango 169.254.x.x (APIPA) indicando fallo del servidor DHCP. En AWS esto no ocurre porque el DHCP siempre funciona; si por alguna razón no ves IP en la instancia, es señal de otro problema.

**Nota:** Además de IP, el DHCP suministra máscara, *gateway* y *DNS*. Así, el equipo ya sabe por dónde salir de su subred (gateway) y cómo resolver nombres sin configurar nada extra. En una VPC, el gateway suele ser el ".1" de la subred y el DNS interno de AWS es otra IP en el rango. Si tu instancia ve DNS, podrá resolver nombres en Internet (p. ej. ubuntu.com). Si falla DNS, no navegarás por nombres aunque sí por IP.

# 7. Puertos y protocolos (TCP vs UDP)

Cada dispositivo en red tiene puertos lógicos (0–65535) para identificar servicios diferentes[20]. Piensa en la IP como la dirección de un edificio y el puerto como el número de departamento. Así, a una misma IP pueden "vivir" varios servicios escuchando en distintos puertos.

- **Ejemplos de puertos comunes:**
- Puerto 22 (SSH, acceso remoto seguro).
- Puerto 80 (HTTP, web no cifrada).
- Puerto 443 (HTTPS, web cifrada).
- Puerto 3389 (RDP, escritorio remoto Windows).
- Puerto 3306 (MySQL), 5432 (PostgreSQL), 25 (SMTP correo), etc.

> *"El TCP y el UDP identifican los terminales de comunicación con números de puerto. Un número de puerto es análogo al nombre del destinatario en una carta[20]."*

En la práctica, para nuestros laboratorios los puertos más importantes son TCP/22 para SSH, TCP/80 y 443 para servidores web. Si intentas acceder al servicio equivocado (por ejemplo, conectar por SSH pero sólo abriste HTTP en el firewall), la "puerta" está cerrada y no habrá respuesta.

**TCP vs UDP:**
- **TCP (Transmission Control Protocol):** orientado a conexión, fiable. Establece una conexión, envía datos y verifica cada fragmento (acknowledgement). Reenvía lost data y asegura entrega ordenada[21]. Por ejemplo, SSH, HTTPS, FTP, SMTP usan TCP porque deben llegar intactos.
- **UDP (User Datagram Protocol):** sin conexión, sin garantías. Envía datagramas sin confirmar recepción[22]. Es más rápido y eficiente cuando se toleran pérdidas: streaming de video, VoIP, DNS queries (generalmente UDP/53), juegos en línea, etc.

> *"El TCP... garantiza una entrega fiable en orden del flujo de bytes... El UDP... la entrega no está garantizada[21]."*

Por eso suele decirse: TCP = **fiabilidad** (simula un canal seguro como una llamada telefónica); UDP = **velocidad** (como emitir una señal sin confirmar, tipo vídeo en directo). En los SG (grupos de seguridad) de AWS especificas si la regla es TCP o UDP. Por defecto, la mayoría de los servicios comunes iniciales (SSH, HTTP) usan TCP.

**Ejercicio (Puertos):** Enumera el servicio típico de cada puerto:
- 22: **SSH** (consola remota Linux)[23].
- 80: **HTTP** (web sin cifrar).
- 443: **HTTPS** (web segura)[24].
- 3389: **RDP** (Windows remoto)[24].
- 53: **DNS** (resuelve nombres, UDP).

Si intentas `ssh -p 22` a una instancia pero en su SG no has abierto el puerto 22, la conexión fallará (esperará hasta timeout). Siempre revisa en el SG que el puerto del servicio (22 para SSH, 80/443 para web, 3389 para RDP, etc.) esté permitido para tu IP origen.

# 8. Firewall: SG de AWS y firewall local

Un **firewall** filtra tráfico de red según reglas (IP de origen, destino, puerto, protocolo). Actúa como guardia: decide qué paquetes dejar pasar[25]. En AWS usamos principalmente **Security Groups (SG)**, que son firewalls virtuales **stateful** a nivel de instancia[26][25]. Cada instancia

EC2 tiene uno o más SG asociados. Las reglas del SG definen qué tráfico **entrante** se permite hacia la instancia y qué tráfico **saliente** se permite fuera.

- **Reglas entrantes:** Por defecto un nuevo SG *no permite nada entrante*. Por ejemplo, si no agregas una regla SSH, ningún paquete TCP/22 entrante será aceptado, aunque la instancia tenga IP pública. Debes añadir por ejemplo "TCP puerto 22 desde tu IP/32" para abrir la sesión SSH, o "TCP 80/443 desde 0.0.0.0/0" para un servidor web público.
- **Reglas salientes:** Por defecto un SG permite **todo el tráfico saliente** (0.0.0.0/0, todas protocolos)[26]. Esto significa que la instancia puede iniciar conexiones a cualquier parte a menos que lo restrinjas.

> *"Un firewall es un dispositivo de seguridad de red diseñado para monitorear, filtrar y controlar el tráfico entrante y saliente basado en reglas[25]."*

**Stateful:** Los SG son stateful, es decir, permiten automáticamente el retorno de tráfico establecido. Si permites entrada por el puerto 443, por ejemplo, la respuesta sale aunque no hayas puesto regla de salida específica. Y viceversa: si permites que la instancia inicie (outbound) por cierto puerto, el tráfico de respuesta entrante estará permitido aunque no esté en las reglas inbound.

Además del SG por instancia, AWS ofrece **NACLs (Network ACLs)** a nivel de subred, pero en los laboratorios sencillos suelen dejarse en "permitir todo" para no complicar. El SG es suficiente.

**Firewall local:** Dentro de la instancia también puede correr un firewall (iptables, UFW, Windows Firewall). Este es **independiente del SG**. Ejemplo: podrías abrir el puerto 80 en el SG pero dentro de Linux habilitar iptables que bloquee el puerto 80; entonces el servicio web no responderá, aunque el SG lo permita. En general, en entornos educativos solemos confiar en los SG de AWS y relajar el firewall interno o desactivarlo, pero es bueno saber que existe esa segunda capa.

**Ejemplo práctico:** Instalas un servidor web en Linux y quieres que sea accesible desde Internet. Pasos de revisión:
1. La instancia debe tener IP pública (o Elastic IP) y estar en subred con IGW[7].
2. El SG debe permitir entrada TCP/80 (HTTP) o 443 (HTTPS) desde el origen deseado (0.0.0.0/0 si es público)[25].
3. En la instancia, el servidor debe estar activo escuchando en ese puerto (p.ej. Apache corriendo en el 80) y el firewall interno (si lo hay) debe permitir el puerto.
4. Si tras todo esto sigues sin ver respuesta, prueba hacer *ping* (ICMP) a la IP pública para ver si responde. Si el ping falla, podría ser simplemente que no abriste ICMP en el SG (por defecto no se permite ICMP). Pero si `curl http://IP` o `telnet IP 80` no responde, puede ser que el servidor esté detenido o que el firewall local bloquee.

Como analogía: el SG es la muralla exterior del castillo; si no abres la puerta (regla) nadie entra a la sala (instancia). El firewall local es la puerta interior; incluso con la muralla abierta, si la puerta interior está cerrada, no pasarás.

# 9. DNS (Sistema de Nombres de Dominio)

El **DNS** es la "guía telefónica de Internet"[27]. Dado que las IP son difíciles de recordar, DNS traduce nombres de dominio legibles (ej. *google.com*, *midominio.es*) a direcciones IP numéricas[27]. Cuando escribes un nombre en el navegador, tu equipo pregunta al **servidor DNS** cuál es la IP correspondiente. El DNS es jerárquico: un resolver local consulta a los root, luego a los servidores de nombres autoritativos, etc., pero todo ocurre automáticamente.

> *"El sistema de nombres de dominio (DNS) es el directorio telefónico de Internet... traduce los nombres de dominio a direcciones IP para que los navegadores puedan cargar recursos de Internet[27]."*

En la nube esto sigue igual. AWS asigna nombres DNS internos a las instancias (ej. `ip-10-0-1-25.ec2.internal`) que apuntan a sus IP privadas, y también nombres públicos (ej. `ec2-3-85-24-117.compute-1.amazonaws.com`) para las IP públicas. Si tu instancia no pudiera resolver DNS, no podría, por ejemplo, actualizar paquetes con `apt-get update` usando nombres de repositorios; sólo funcionaría si usas las IP fijas de esos repositorios. En cada VPC AWS hay un resolver DNS interno (por defecto en la IP .2 de la red) que las instancias usan para resolver tanto nombres privados como públicos, siempre que tengan salida a Internet.

**Ejemplo:** Montas un bucket S3 o una base de datos RDS. AWS te da una URL DNS (p.ej. `miapp.s3.amazonaws.com` o `mi-rds-instance.abc123.us-east-1.rds.amazonaws.com`) que debes usar en tus aplicaciones. Gracias al DNS, tu instancia puede comunicarse con ellos sin saber su IP interna. A la inversa, podrías registrar tu propio dominio (por ejemplo `www.midominio.com`) en Route 53 y apuntarlo a la IP pública de tu instancia, de modo que los usuarios usen el nombre, no la IP.

# 10. Acceso remoto (SSH/RDP)

Para administrar tus instancias necesitas conectarte remotamente. En Linux se usa **SSH (Secure Shell)** por defecto. SSH es un protocolo seguro de capa de aplicación que abre una consola en otra máquina vía red. Tu PC ejecuta un cliente SSH; la instancia AWS corre un servicio SSH (sshd) que "escucha" típicamente en el puerto 22. Al conectarte, se crea un canal cifrado donde ingresas comandos.

**Requisitos para SSH a una instancia AWS Linux:**

1. **Dirección accesible:** La instancia necesita IP pública (o un nombre público que la resuelva) si vienes desde Internet. Si sólo tiene IP privada, solo será alcanzable desde la misma VPC (por ejemplo, vía VPN o través de un bastion host público). En entornos educativos, generalmente damos IP pública a las instancias para que los alumnos puedan acceder directamente.

2. **Regla del firewall (SG):** Debes permitir el puerto TCP/22 en el Security Group de la instancia, desde tu IP o red local. Si omites esto, la conexión SSH se rehusará (timeout). Un SG típico solo permite SSH entrante desde tu IP (ej. `192.0.2.50/32`).

3. **Clave SSH válida:** AWS usa autenticación por clave pública/privada. Al lanzar la instancia, AWS te pide un par de claves (key pair). Descargas la *clave privada* (archivo `.pem`). La clave pública correspondiente se inyecta en la instancia. Para conectar, usas tu clave privada: e.g. `ssh -i mi_clave.pem ec2-user@18.85.24.117`. AWS te indica el nombre de usuario por defecto según la AMI (por ejemplo, `ec2-user` para Amazon Linux, `ubuntu` para Ubuntu, `centos` para CentOS, `admin` o `ec2-user` para RHEL, etc.). **Es crucial proteger el .pem:** dales permiso 400 (`chmod 400 mi_clave.pem`) y no lo compartas. Sin la clave privada correcta (o usando el usuario equivocado), obtendrás un error de "Permission denied (publickey)".

4. **Cliente SSH en tu equipo:** Hoy día la mayoría de sistemas operativos trae un cliente SSH. En Windows 10+ puedes usar PowerShell o la app terminal con `ssh`. También están herramientas gráficas como PuTTY (requiere convertir .pem a .ppk) o Termius.

5. **Conectividad de red:** Asegúrate de que nada bloquee el puerto 22 saliente desde tu red local. Algunas redes corporativas bloquean SSH; en ese caso no podrás alcanzar la instancia ni aunque todo en AWS esté bien configurado.

**Ejemplo de conexión SSH:**
- Tienes una instancia **18.85.24.117** en la VPC predeterminada.
- Al crearla seleccionas la clave `alumno1.pem`.
- En su Security Group agregas una regla entrante: **TCP/22** desde tu IP pública (p.ej. `203.0.113.5/32`).
- En tu PC (203.0.113.5) ejecutas:

```
ssh -i alumno1.pem ec2-user@18.85.24.117
```

Aceptas la huella de la clave (yes) y ya estarás en la instancia con el prompt de shell.

**Posibles problemas y diagnósticos:**
- *Timeout / No route to host:* Verifica que la instancia esté `running`, tenga IP pública, la subred tenga IGW (o si es privada, VPN), y que tu red no bloquee SSH.
- *Permission denied (timeout rápido):* Seguramente el SG no permitía tu IP en el puerto 22.
- *Permission denied (publickey):* Has contactado al puerto 22, pero la clave o usuario es incorrecto.
- *Host not found:* Falló la resolución DNS (si usaste nombre en vez de IP). Prueba con la IP directamente.
- *Conexión cerrada inmediatamente:* Podría ser el firewall interno de la instancia (iptables/UFW) bloqueando SSH, o que sshd está caído (menos común en AMIs estándar).

Para **Windows**, en AWS se usa **RDP (Remote Desktop)**. RDP funciona sobre TCP/3389. En lugar de clave pública, AWS obtiene una contraseña inicial. Necesitas habilitar el puerto 3389 en el SG y luego usar la IP pública (o nombre) con un cliente de Escritorio Remoto.

**Consejo:** Guárdate bien la clave privada. Si la pierdes no hay "reset" fácil; tendrías que crear otra instancia o usar mecanismos avanzados (por ejemplo, lanzar otra instancia en la misma VPC, montar el disco de la instancia original y extraer una clave nueva). Mejor prevenir.

# 11. VPN (Redes privadas virtuales)

Una **VPN (Virtual Private Network)** crea un **túnel cifrado** entre dos redes o entre tu equipo y una red remota, usando Internet como medio. En la práctica hace que una máquina remota parezca estar conectada directamente a tu red local[28].

**Tipos comunes de VPN:**
- **Site-to-Site (oficina ↔ AWS):** Conecta una red local (on-premises) con una VPC en AWS. Cada extremo tiene un gateway VPN que cifra el tráfico por Internet. Así las máquinas en la oficina (ej. red 192.168.10.0/24) y las instancias en AWS (p.ej. 10.0.0.0/16) se comunican como si estuvieran en la misma red extendida. AWS ofrece *AWS Site-to-Site VPN* para esto.
- **Client VPN (teletrabajo):** Un usuario individual (un portátil) se conecta mediante cliente VPN a la VPC, obteniendo virtualmente una IP de esa red. Por ejemplo, instalas OpenVPN o AWS Client VPN, te conectas desde casa, y tu portátil ahora puede acceder a las instancias con IPs privadas (como 10.0.1.25) como si estuvieras dentro de la VPC. Todo el tráfico se cifra.

> *"Una VPN crea una conexión de red privada entre dispositivos a través de Internet. Se utiliza para transmitir datos de forma segura a través de redes públicas[28]."*

**Usos típicos en cloud:**
- **Extender la LAN al cloud:** Conecta la oficina de una empresa con la VPC. Por ejemplo, puede haber una base de datos en la oficina y aplicaciones en AWS que la usan por la VPN, como si todo estuviera en la misma LAN.
- **Acceso remoto seguro:** Tú, estudiante, desde casa conectas tu PC a la VPC por VPN. Así obtienes una IP interna y puedes llegar a instancias privadas (que no tienen IP pública) por SSH o RDP sin exponerlas a Internet.
- **Privacidad en redes públicas:** Algunas VPN (como NordVPN) se usan para cifrar tu tráfico en cafeterías o para «camuflar» tu IP, pero esto ya sale del alcance de los labs AWS.

El funcionamiento básico de una VPN (p.ej. IPSec) es encapsular y cifrar tus paquetes. Imagina que metes tus datos en un sobre en tu PC, el sobre viaja por Internet (inseguro) cifrado, y sólo lo abren en el gateway del otro extremo. El resultado es una conexión segura **y privada** sobre la red pública. En AWS se emplea mucho para entornos híbridos.

**Ejemplo básico de VPN site-to-site:** Tu empresa ACME tiene 192.168.1.0/24 en la oficina y 10.0.0.0/16 en AWS. Configuran un túnel IPSec entre el firewall local (p.ej. Cisco) y el Virtual Private Gateway de la VPC. Luego agregan en las rutas de la VPC que todo el tráfico a 192.168.1.0/24 vaya por la VPN, y en el router local que 10.0.0.0/16 vaya por el túnel. Desde cualquier PC en la oficina podrás hacer ping a una instancia en AWS por su IP privada, y viceversa. Todo va cifrado por Internet.

**Nota para labs:** Si tienes una instancia privada y te dicen "conéctate vía VPN", significa que no tiene IP pública. Deberás conectarte al VPN primero para acceder a ella. Si la VPN falla (túnel caído), simplemente las dos redes no se verán hasta restaurarla. Configurar VPN es avanzado, pero entender que existe y para qué sirve es suficiente en nivelación.

# 12. Diagnóstico básico de red (ping, traceroute)

Cuando algo no funciona en la red (p.ej. “mi instancia no responde”), unas herramientas básicas te ayudan a identificar dónde está el problema:

- **Ping:** Envía un paquete ICMP “echo request” y espera el “echo reply”. Si obtienes respuesta, significa que hay **conectividad IP** entre tu origen y destino y mide latencia round-trip. Si no responde (timeout), puede deberse a varias causas: la instancia está apagada, la ruta no existe, o simplemente ICMP está bloqueado. Ten en cuenta que **por defecto el SG de AWS suele bloquear ICMP entrante**, así que un ping fallido desde fuera no siempre indica fallo—puede ser que ICMP no esté permitido en el SG. Dentro de la VPC puedes habilitar ICMP en los SG para hacer pruebas internas (ping entre instancias).

- **Traceroute (tracert en Windows):** Muestra la ruta completa que toman los paquetes hasta el destino, listando cada “salto” (router) intermedio. Te ayuda a ver *dónde* se corta la comunicación. Por ejemplo, un traceroute desde tu PC a la IP pública de la instancia podría mostrar saltos por tu ISP, luego por Amazon, y finalmente quedar en *“” en el último salto (lo que sugiere que el último router o la instancia no respondió). Si el traceroute muere en un paso remoto, puede indicar fallo en esa parte de la red o un bloqueo. Internamente, un traceroute desde la instancia hacia tu IP revelará si el tráfico llega hasta el router de la VPC, Internet, etc. Nota: muchos routers intermedios (especialmente en la red de AWS o de Google) no responden a traceroute, así que verás “”* aunque el tráfico sí pase.

**Pasos de diagnóstico típicos (ejemplo “no puedo SSH”):**
1. **Ping a la IP pública de la instancia:** Si responde, sabes que la instancia está encendida y que el firewall (ICMP) lo permitió. Si no responde, intenta desde dentro de la VPC u otra ruta para aislar.
2. **Telnet/traceroute al puerto SSH (22):** En tu PC `telnet IP 22` o `nc -vz IP 22`. Si no se conecta, probablemente el SG o NACL lo bloquea.
3. **Traceroute:** Desde tu PC: `traceroute IP_instancia`. Observa hasta dónde llega. Desde la instancia: `traceroute <tu_IP>` o a internet (8.8.8.8).
4. **Revisa configuración AWS:** Confirma en la consola EC2 que la instancia corre, tiene IP pública (o Elastic IP), que su subred tiene ruta a IGW si es pública, o a NAT si necesita salir a Internet. En la pestaña de Seguridad verifica reglas Inbound/Outbound del SG.
5. **Logs/Firewall local:** Si todo AWS parece correcto, verifica el firewall interno y los servicios en la instancia (p.ej. que SSH esté activo y listening).

**Ejemplo de uso de ping/traceroute:**
- Tu instancia *no responde SSH*. Haces `ping IP`. Timeout. Posible causa: SG bloquee ICMP o instancia apagada. Luego haces `ssh -v -i clave.pem ec2-user@IP`. Observas que alcanza el servidor pero da *Connection refused* o *timeout*. Si es “refused” puede indicar SSH caído o puerto cerrado. Si es timeout, SG o ruta. Un `traceroute IP` puede mostrar que llega hasta Amazon y luego nada (indica que el problema es más interno).

En resumen, ante problemas: **IP correcta + puertos + firewall + rutas/DNS**. Ping y traceroute son tus aliados iniciales.

# 13. Conclusión

En esta unidad hemos repasado los fundamentos de redes esenciales para trabajar con cloud (AWS). Vimos qué es una red (LAN/WAN/Internet), cómo identificamos equipos con **IP** (pública/privada), cómo segmentamos redes con **subnets** y máscaras, y cómo administramos la conectividad a Internet usando **NAT**, **Internet Gateway** y DHCP[11][7][19]. Aprendimos el modelo OSI y dónde encajan dispositivos como routers y switches[8]. También cubrimos **puertos y protocolos** (TCP vs UDP, ejemplos de puerto 22/80/443)[20][24]. Vimos el papel de los **firewalls** en AWS (Security Groups) y local, y la importancia de **DNS** para traducir nombres a IP[27]. Revisamos cómo conectarnos remotamente con **SSH/RDP** (requisitos: IP, firewall abierto, clave) y qué hacer si "no conecta". Por último, introdujimos las **VPN** como túneles seguros de red[28].

**Glosario rápido:** LAN (red local), WAN (red amplia), IP pública/privada, subnet (/mask), NAT, Internet Gateway, DHCP, puerto TCP/UDP, firewall/SG, DNS, SSH, VPN. Con esto ya puedes entender términos como "lanzar instancia en subred pública", "asociar Elastic IP", "abrir puerto 22 en SG", "mi instancia no tiene salida a Internet", etc.

**Consejo final:** Al diagnosticar problemas de conexión, recuerda revisar siempre los cuatro pilares: **IP → Puertos → Firewall → Rutas/DNS**. Ej.: "Ping no responde" (¿IGW/NAT? ¿SG ICMP?), "SSH timeout" (¿SG22 abierto? ¿IP correcta?), "curl falla" (¿DNS resuelve? ¿firewall local?), etc. Conociendo estos conceptos y ejercicios prácticos, estarás preparado para avanzar en tus laboratorios AWS sin que la "red" te sorprenda.

¡Éxitos y a practicar!

---


[1] Red de computadoras - Wikipedia, la enciclopedia libre

https://es.wikipedia.org/wiki/Red_de_computadoras

[2] ¿Qué es una dirección IP y qué significa?

https://latam.kaspersky.com/resource-center/definitions/what-is-an-ip-address?srsltid=AfmBOoo4jRFDDfhZKjdIdRubmUK_UhK135C1Re9GdqNrJJZVnj0LZAJR

[3] [4] Red LAN - Concepto, tipos, topologías y qué es Internet

https://concepto.de/red-lan/

[5] [12] [17] Enable internet access for a VPC using an internet gateway - Amazon Virtual Private Cloud

https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Internet_Gateway.html

[6] [7] [14] [18] Activación del acceso a Internet de una VPC con una puerta de enlace de Internet - Amazon Virtual Private Cloud

https://docs.aws.amazon.com/es_es/vpc/latest/userguide/VPC_Internet_Gateway.html

[8] ¿Qué es el modelo OSI? - Definición, capas y más | Proofpoint ES

https://www.proofpoint.com/es/threat-reference/osi-model

[9] ¿Qué es una dirección IP? ¿Cómo funciona? ¿Cómo localizarlo? | Fortinet

https://www.fortinet.com/lat/resources/cyberglossary/what-is-ip-address

[10] [13] ¿Qué es una subred? | Cómo funciona una subred | Cloudflare

https://www.cloudflare.com/es-es/learning/network-layer/what-is-a-subnet/

[11] Red privada - Wikipedia, la enciclopedia libre

https://es.wikipedia.org/wiki/Red_privada

[15] NAT: qué es y cómo funciona | Cibersafety

https://cibersafety.com/que-es-nat-funcionamiento-redes/

[16] ¿Qué es NAT (Network Address Translation y para qué sirve? - Fastweb - Expertos en Conectividad

http://www.fastweb.com.mx/index.php/ablog/que-es-nat-network-address-translation-y-para-que-sirve

[19] ¿Qué es DHCP? ¿Cómo funciona DHCP? ¿Por qué es importante? | Fortinet

https://www.fortinet.com/lat/resources/cyberglossary/dynamic-host-configuration-protocol-dhcp

[20] [21] [22] [23] [24] ¿Qué son los puertos? | ¿Cómo funcionan los puertos? | Akamai

https://www.akamai.com/es/glossary/what-are-ports

[25] ¿Qué es un firewall? Definición y tipos de firewall | Fortinet

https://www.fortinet.com/lat/resources/cyberglossary/firewall

[26] Grupos de seguridad predeterminados para las VPC - Amazon Virtual Private Cloud

https://docs.aws.amazon.com/es_es/vpc/latest/userguide/default-security-group.html

[27] ¿Qué es un DNS? | Cómo elegir el mejor DNS | Cloudflare

https://www.cloudflare.com/es-es/learning/dns/what-is-dns/

[28] ¿Qué es una VPN? - Explicación de las redes privadas virtuales - AWS

https://aws.amazon.com/es/what-is/vpn/