**Recipe 2 – Networking for DevOps**

This recipe uses the **What – How – Why** format to guide you through essential networking concepts every DevOps engineer must master:

* TCP/IP & Ports
* DNS
* HTTP/HTTPS
* VPN
* Load Balancers
* Firewalls
* Protocols (ICMP, TCP, UDP)
* Subnetting (CIDR, Masks)

**🔸 What (Introduction & Theory – Beginner Friendly)**

Think of **networking** as the nervous system of DevOps. Whether your code runs on Kubernetes, in AWS/Azure/GCP, or inside a container, it must **talk to other systems**. If Linux is the engine, then networking is the **road system that connects all engines together**.

Let’s break it down:

**1. TCP/IP & Ports**

* TCP/IP is the foundation of the internet.
* **IP Address** = like a house address (where a machine lives). Example: 192.168.1.10.
* **Port** = like a door in that house (which service is listening). Example:
  + Port **22** → SSH
  + Port **80** → HTTP
  + Port **443** → HTTPS
* Without IPs and ports, services cannot communicate.

**2. DNS (Domain Name System)**

* Humans like names (openai.com), but machines like numbers (104.18.12.123).
* DNS translates names into IP addresses (like a phonebook).
* Example: typing google.com triggers a DNS lookup → gives back an IP.

**3. HTTP/HTTPS (Web Protocols)**

* **HTTP** = rules for communication between browsers and servers.
* Example: GET /index.html asks for a web page.
* **HTTPS** = HTTP + SSL/TLS encryption → secure communication.
* In DevOps, APIs, web apps, CI/CD dashboards all run on HTTP/HTTPS.

**4. VPN (Virtual Private Network)**

* Creates a secure tunnel between your system and a private network.
* Example: Connecting to a company’s AWS VPC from your laptop securely.
* Encrypts all traffic so outsiders cannot spy.

**5. Load Balancers**

* Imagine one cashier vs. 10 customers. That’s a bottleneck.
* Load balancers distribute traffic across multiple servers.
* Types:
  + **L4 (Transport level – TCP/UDP)** → balances based on IP/port.
  + **L7 (Application level – HTTP)** → balances based on content (e.g., /api vs /login).

**6. Firewalls**

* A **security guard** that allows or denies traffic.
* Works with **rules**:
  + Example: allow port 22 (for SSH).
  + Example: deny port 23 (Telnet, insecure).

**7. Protocols**

* **ICMP** → used by ping (test connectivity).
* **TCP** → reliable, connection-based (used in web, ssh).
* **UDP** → fast but unreliable (used in DNS, video streaming).

**8. Subnetting (CIDR Notation)**

* Large networks are broken into smaller networks (**subnets**).
* Example: 192.168.1.0/24 → means 256 IPs (from 192.168.1.0 to 192.168.1.255).
* **CIDR (Classless Inter-Domain Routing)** notation:
  + /24 → 255.255.255.0 mask.
  + /16 → bigger subnet, 65k IPs.

👉 Without subnetting, cloud VPCs and Kubernetes clusters cannot scale securely.

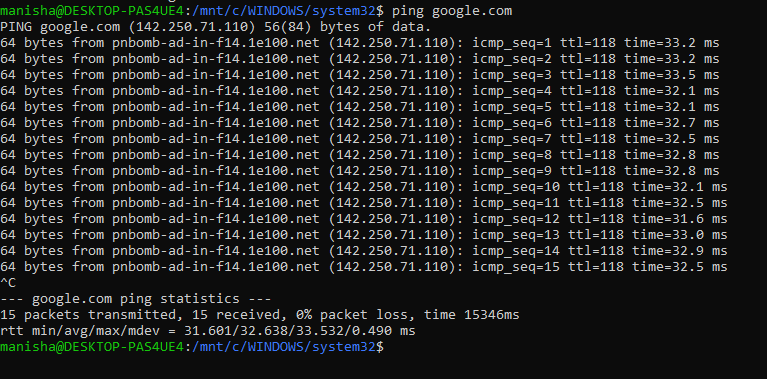
**🔸 How (Step-by-Step with Beginner Explanations)**

Let’s practice networking with some real commands:

**Part 1: Basic Connectivity**

**Step 1: Ping a website**

ping google.com

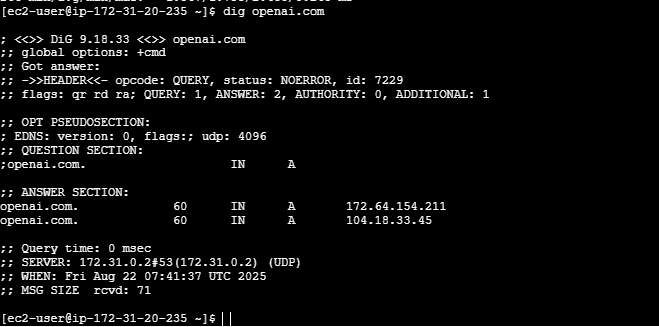


* Uses **ICMP** packets to check connectivity.
* Press Ctrl+C to stop.

**Part 2: DNS Lookup**

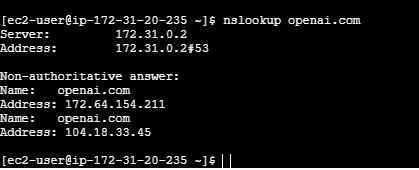
**Step 2: Resolve domain to IP**

dig openai.com



* Shows which DNS server answered, and what IP address was returned.
* Alternative:

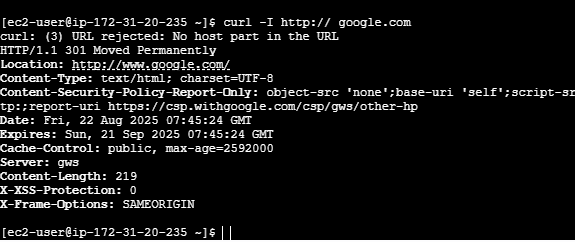
nslookup openai.com



**Part 3: HTTP/HTTPS Check**

**Step 3: Send a web request**

curl -I <https://example.com>

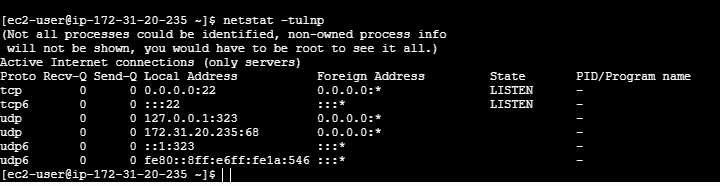


* -I = fetch only headers.
* Shows response: 200 OK = success, 404 Not Found = missing page.

**Part 4: Open Ports**

**Step 4: List listening ports**

netstat –tulnp



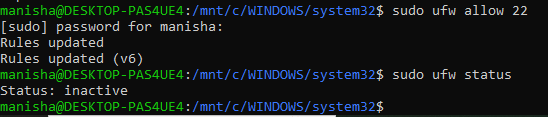
* t = TCP, u = UDP, l = listening, n = show numbers, p = process.
* Example output:
  + tcp 0 0 0.0.0.0:22 sshd → SSH is listening on port 22.

**Part 5: Firewall**

**Step 5: Configure firewall (Ubuntu example)**

sudo ufw allow 22

sudo ufw status

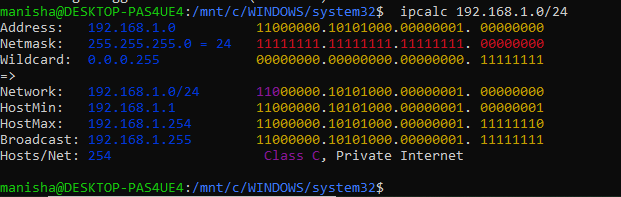


* Opens port 22 (SSH).
* status shows which ports are allowed/denied.

**Part 6: Subnetting**

**Step 6: Subnet calculation**

ipcalc 192.168.1.0/24



* Shows network range, broadcast, usable IPs.
* Great for understanding VPC setup in AWS/Azure.

**🔸 Why (Practical Thinking & Reasoning)**

Now let’s see why these networking concepts are **critical in DevOps**:

1. **Why TCP/IP & Ports?**  
   ✔️ Because Kubernetes pods, Docker containers, and servers all talk over IP:Port. If you don’t understand ports, you won’t know why your app isn’t reachable.
2. **Why DNS?**  
   ✔️ Every service uses DNS (APIs, S3 buckets, GitHub). If DNS fails, the internet feels “down.” Debugging starts with dig or nslookup.
3. **Why HTTP/HTTPS?**  
   ✔️ CI/CD dashboards (Jenkins, GitHub Actions, ArgoCD) all run on web interfaces. Knowing HTTP status codes = faster debugging.
4. **Why VPN?**  
   ✔️ To access private servers or cloud environments securely. Without VPNs, companies expose themselves to attacks.
5. **Why Load Balancers?**  
   ✔️ No website or API scales without load balancing. Kubernetes Ingress, AWS ELB, NGINX all are load balancers in disguise.
6. **Why Firewalls?**  
   ✔️ Security first. If you don’t open the right ports, apps won’t work. If you open too many, you risk a hack.
7. **Why Protocols?**  
   ✔️ When you ping, you use ICMP. When you curl, you use TCP. When DNS queries resolve, they use UDP. Debugging = knowing which protocol is in play.
8. **Why Subnetting?**  
   ✔️ Cloud networks (VPCs, VNets, subnets) are based on CIDR. If you misconfigure, services won’t talk to each other → broken pipelines.

👉 In short: Networking is the **lifeline of DevOps**. Without it, CI/CD fails, Kubernetes fails, and cloud infra breaks. Master these basics, and you’ll debug faster than 80% of beginners.