Presentation Title

Subtitle or Company Name

Author Name

January 2026

Networking Fundamentals: OSI & TCP/IP Models

Day 1 Session - MLOps Foundation Series | Table of Contents

- **01** Why Networking Matters for MLOps
- **02** The OSI Model (7 Layers)
- **03** The TCP/IP Model (5 Layers)
- **04** OSI vs TCP/IP Comparison
- **05** Deep Dive: Each Layer
- **06** Common Protocols
- **07** Practical Examples
- **08** Networking Commands
- **09** Troubleshooting Guide
- **10** Resources

LEARNING OBJECTIVE

By the end of this session, participants will be able to map networking errors to specific layers of the OSI model, enabling faster troubleshooting of cloud-based machine learning pipelines and efficient infrastructure design.

Source: MLOps Foundation Series Curriculum

Why Networking Matters for MLOps / The OSI Model (7 Layers)

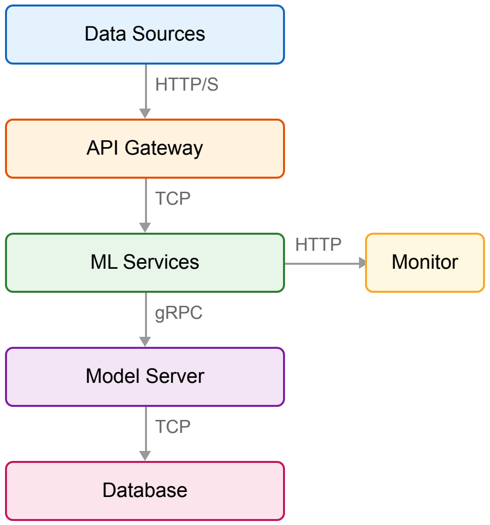
Foundational networking concepts for robust model deployment, security, and debugging

Networking in MLOps Context

Why it matters:

- **Model Deployment:** APIs, load balancers, service meshes
- **Data Pipelines:** Secure transfer between services
- **Infrastructure:** VPCs, subnets, security groups
- **Debugging:** Identifying connectivity bottlenecks
- **Security:** Firewalls, encryption, access control

MLOps Network Architecture



Simplified request flow in production environment

The OSI Model (Open Systems Interconnection)

Layer	Function	Data Unit	Key Devices	Protocols / Examples
7. Application Software Interaction	User interface & services	Data	Firewalls (App Level)	HTTP, FTP, SMTP, DNS
6. Presentation Translation / Encrypt	Data formatting & encryption	Data	-	SSL/TLS, JPEG, ASCII
5. Session Sync & Ports	Connection management	Data	-	NetBIOS, RPC
4. Transport End-to-End Conn.	End-to-end delivery	Segment	Load Balancers	TCP, UDP
3. Network Packets & Routing	Routing & addressing	Packet	Routers	IP, ICMP, ARP
2. Data Link Frames & MAC	Node-to-node delivery	Frame	Switches, Bridges	Ethernet, PPP, MAC
1. Physical Cables & Signals	Physical transmission	Bits	Hubs, NICs	USB, DSL, Bluetooth

🧠 Memory Tricks

↓ 7-1: "**A**ll **P**eople **S**eem **T**o **N**eed **D**ata **P**rocessing"
↑ 1-7: "**P**lease **D**o **N**ot **T**hrow **S**ausage **P**izza **A**way"

KEY INSIGHT

For MLOps, **Layer 4 (Transport)** and **Layer 7 (Application)** are critical. Misconfiguration here often causes the most common production issues (e.g., API timeouts, port conflicts, or serialization errors).

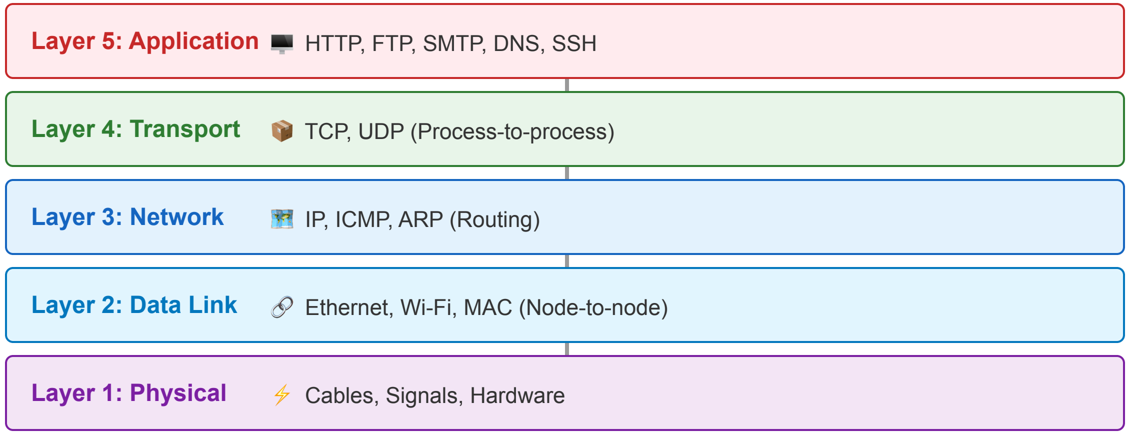
Source: Open Systems Interconnection (OSI) model standards; Cloud Native Computing Foundation (CNCF) MLOps best practices

The TCP/IP Model (5 Layers) vs. OSI Model

Comparison of the practical Internet protocol suite against the theoretical OSI reference model

TCP/IP 5-Layer Implementation

The TCP/IP model is the practical foundation of the modern Internet. While originally defined with 4 layers, the 5-layer model is commonly taught to distinguish physical hardware from data link protocols.



Layer 5: Application 🖥 HTTP, FTP, SMTP, DNS, SSH

Layer 4: Transport 📦 TCP, UDP (Process-to-process)

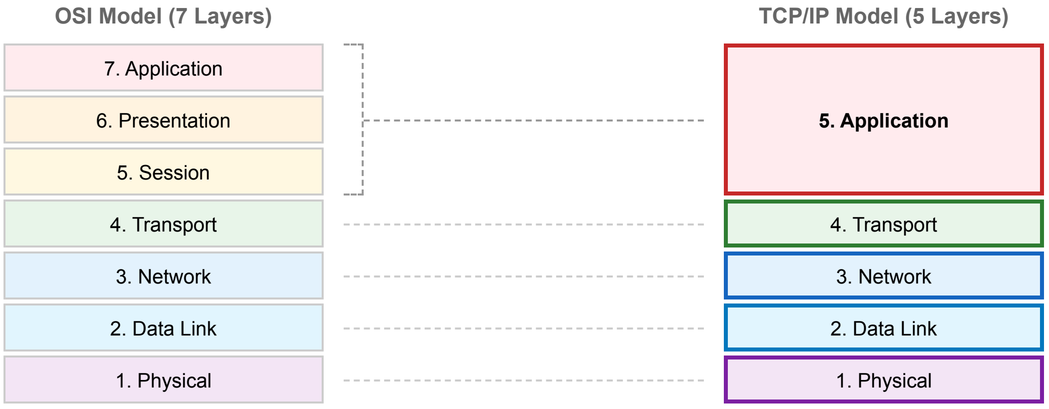
Layer 3: Network 🗺 IP, ICMP, ARP (Routing)

Layer 2: Data Link 🔗 Ethernet, Wi-Fi, MAC (Node-to-node)

Layer 1: Physical ⚡ Cables, Signals, Hardware

L#	Name	Function	Key Protocols
5	**Application**	User services	HTTP, DNS, SMTP
4	**Transport**	Process delivery	TCP, UDP
3	**Network**	Host routing	IP, ICMP
2	**Data Link**	Node delivery	Ethernet, Wi-Fi
1	**Physical**	Raw signals	Cables, Fiber

OSI vs. TCP/IP Comparison



Key Differences

Aspect	OSI Model	TCP/IP Model
Development	ISO standard (Theoretical)	DoD/DARPA (Practical)
Approach	Protocol-independent reference	Protocol-specific solution
Layers	7 separate layers	5 (condenses upper layers)
Session/Pres	Distinct layers	Combined into Application
Primary Use	Education & Troubleshooting	Real-world Internet

WHY BOTH MODELS MATTER

Use OSI Model For:

- Learning networking concepts
- Troubleshooting specific layer issues
- Vendor-neutral communication

Use TCP/IP Model For:

- Actual network implementation
- Understanding Internet protocols
- Cloud & Infrastructure operations

Source: Network Engineering Standards; ISO/IEC 7498-1; RFC 1122

Deep Dive: Network Layers & Protocols

Detailed breakdown of functions, key concepts, and technologies across the OSI model layers

Layer 1: Physical Layer ⚡

Function: Transmits raw bits (0s/1s) over physical media. Defines electrical/optical specs, voltage, and connectors.

Key Concepts:

- **Bandwidth:** Capacity (Mbps, Gbps)
- **Latency:** Travel time

Technologies:

Copper (Cat6) | Fiber Optic | Wi-Fi (802.11ax) | Bluetooth

Layer 2: Data Link Layer 🔗

Function: Node-to-node transfer on the same network. Handles error detection and physical addressing.

MAC Address (Hardware ID)

`AA:BB:CC:DD:EE:FF (48-bit)`

Key Concepts:

- **Frame:** Data unit at this layer
- **ARP:** Maps IP to MAC addresses

Devices & Protocols:

Switches, Bridges, Ethernet (802.3), PPP

Layer 3: Network Layer 🗺️

Function: Logical addressing (IP), routing packets between networks, and path determination.
Protocols: IP (IPv4/IPv6), ICMP (Ping), OSPF.

IPv4 Address Classes & Private Ranges:

Class	Range / Private	Purpose
A	1.0.0.0 - 126.x (Priv: 10.0.0.0/8)	Large Orgs
B	128.0.0.0 - 191.x (Priv: 172.16.0.0/12)	Medium
C	192.0.0.0 - 223.x (Priv: 192.168.0.0/16)	Small/Home

Layer 4: Transport Layer 📦

Function: End-to-end communication, reliability, flow control, and port addressing.

TCP
Reliable, Ordered, Connection-oriented (Web, Email)

UDP
Fast, No guarantee, Connectionless (Video, DNS)

Common Ports:

22: SSH | **53:** DNS | **80:** HTTP | **443:** HTTPS | **3306:** MySQL

Layers 5 & 6: Session & Presentation (OSI)

Layer 5 (Session): Manages sessions/connections.
Examples: RPC, SQL Sessions, NetBIOS

Layer 6 (Presentation): Translation, encryption, compression.
Examples: SSL/TLS, JPEG, ASCII, MPEG

**In TCP/IP, these are handled by Layer 7 or Layer 4.*

Layer 7: Application Layer 🖥️

Function: User interface to network services; application-specific protocols.

Protocol	Port	Use Case
HTTP/S	80/443	Web Browsing
SMTP	25	Sending Email
FTP	21	File Transfer
DHCP	67/68	Assigning IPs

KEY INSIGHT

Encapsulation drives modularity: Each layer relies solely on the service of the layer below it, allowing hardware (L1/L2) and software (L7) to evolve independently without breaking the stack.

Source: Network Fundamentals Documentation; TCP/IP Guide; OSI Model Standards

Common Protocols / Practical Examples

Detailed view of DNS resolution, HTTP/S handshakes, data encapsulation, and real-world API flows

DNS Resolution Process (Full Recursive Lookup)

1. User types 'google.com' → Checks Local Cache (Miss) → Queries Local DNS

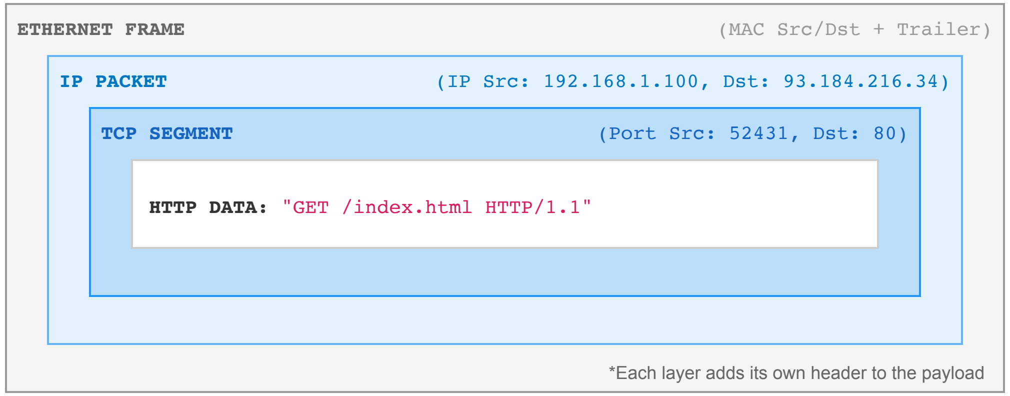
2. Local DNS queries **Root Server (.)** → Returns .com TLD Server IP

3. Local DNS queries **TLD Server (.com)** → Returns Google Auth Server IP

4. Local DNS queries **Authoritative Server** → Returns 142.250.185.78

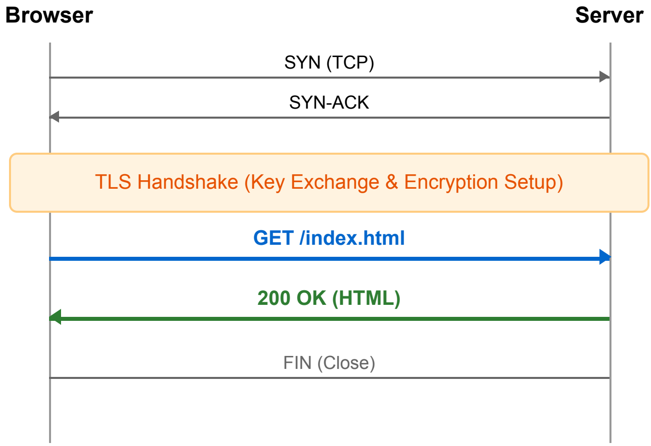
5. Result: Local DNS caches IP & returns to User → Browser connects

Data Encapsulation (The "Envelope" Model)

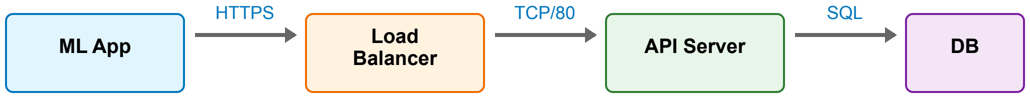

ETHERNET FRAME
(MAC Src/Dst + Trailer)
IP PACKET
(IP Src: 192.168.1.100, Dst: 93.184.216.34)
TCP SEGMENT
(Port Src: 52431, Dst: 80)
HTTP DATA: "GET /index.html HTTP/1.1"

*Each layer adds its own header to the payload

HTTP/HTTPS Request Flow (Sequence)



Real-World Example: ML API Request Flow



Data is de-encapsulated up to L7 at LB for inspection, then re-encapsulated to API

KEY INSIGHT

Understanding protocol flows (Layer 4 vs Layer 7) is critical for performance tuning (e.g., minimizing TLS handshakes) and debugging connectivity issues across microservices.

Source: Network Protocol Standards (IETF RFC 791, 793, 2616); Internal Architecture Docs

Networking Commands / Layer 1 - Physical

OSI LAYER 1

Essential toolset for validating hardware interfaces, link status, and signal integrity

COMMAND	PRIMARY FUNCTION	KEY METRICS & OUTPUT	DIAGNOSTIC USE CASE
`ip link show` Modern replacement for deprecated *ifconfig*	Displays the state of all network interfaces on the host. Provides status flags indicating if the interface is administratively up and physically connected.	• **State:** UP / DOWN / UNKNOWN • **MAC Address:** link/ether • **MTU:** Maximum Transmission Unit • **Queue:** qdisc state	**First-step verification:** Checking if the network cable is plugged in ("LOWER_UP") and if the interface is enabled by the OS.
`ethtool eth0` Requires root/sudo for changing settings	Queries and controls the network driver and hardware settings. Essential for diagnosing physical link negotiation issues.	• **Speed:** 100Mb/s, 1000Mb/s • **Duplex:** Full / Half • **Auto-negotiation:** on / off • **Link detected:** yes / no	**Hardware debugging:** Identifying speed mismatches (e.g., switch port is 1Gbps but server negotiated 100Mbps) or duplex collisions.
`iwconfig` Wireless equivalent of ifconfig	Displays parameters specific to wireless interfaces. Used to verify connection to Access Points (APs) and signal health.	• **SSID:** Network Name • **Frequency:** 2.4GHz / 5GHz • **Bit Rate:** Current transmission rate • **Signal Level:** dBm (Strength)	**Signal validation:** Troubleshooting poor performance due to low signal strength (-80dBm or lower) or verifying correct AP association.

STRATEGIC INSIGHT: THE "LAYER 1 FIRST" APPROACH

Network troubleshooting should always follow a bottom-up methodology. **Over 70% of network "downtime" stems from physical layer issues** such as disconnected cables, port speed mismatches, or signal interference.

Validating connectivity with `ip link` and `ethtool` before analyzing routing tables or firewall rules prevents wasted engineering cycles on higher-layer diagnostics when the physical path is broken.

Source: Linux System Administrator's Guide; IEEE 802.3 Ethernet Standards; McKinsey Digital Infrastructure Best Practices

Layer 2 - Data Link / Layer 3 - Network

Essential diagnostic commands for physical addressing and logical routing

Layer 2: Data Link Diagnostics

ARP Cache & Neighbor Table

`arp -a`	Legacy command to show ARP cache (MAC addresses)
`ip neighbor show`	Modern equivalent; displays neighbor objects

Use to verify if the device can resolve local MAC addresses for IP communication.

Bridge Configuration

`bridge link show`	Show bridge link status and ports

Essential for diagnosing virtual interfaces and software bridges (common in Docker/VMs).

Layer 3: Network Diagnostics

Interface Configuration

`ip addr show`	Display IP addresses and property info

Routing Logic

`ip route show`	Show kernel routing table

Reachability & Path Analysis

`ping 8.8.8.8`	Test basic end-to-end connectivity (ICMP)
`traceroute google.com`	Map the packet path hop-by-hop
`mtr google.com`	Real-time traceroute with packet loss stats

SO WHAT?

Network troubleshooting must follow the OSI model bottom-up. Always verify **Layer 2** (local physical connectivity and ARP resolution) before diagnosing **Layer 3** (IP routing and internet reachability). If `ip neighbor` fails, `ping` will never succeed.

Source: Linux Networking Documentation; Network Administrator Guide

Layer 4 - Transport

Essential toolset for socket monitoring, connectivity verification, and security auditing

1. Host-Based Socket Monitoring

```
netstat -tuln
```

- Traditional command to display **T**CP/**U**DP **L**istening ports **N**umerically.
- universally available but slower on systems with many connections.
- *Usage: Verifying if a service is actually running and binding to a port.*

```
ss -tuln
```

- Modern replacement for netstat; dumps socket stats directly from kernel.
- significantly faster and provides more detailed TCP state information.
- *Usage: Preferred tool for high-performance diagnostics on modern Linux.*

2. Network Connectivity & Discovery

```
nc -zv host 80
```

- Netcat (nc) in **Z**ero-I/O mode (scan) with **V**erbose output.
- "Swiss Army Knife" for reading/writing data across network connections.
- *Usage: Quick connectivity checks to specific ports (firewall validation).*

```
nmap -p 1-1000 host
```

- Network Mapper; scans a range of ports on a remote host.
- Identifies open ports, service versions, and OS fingerprints.
- *Usage: Security auditing and network inventory discovery.*

Tool Capabilities Summary

Command	Category	Function	Key Benefit
`netstat`	Monitoring	Lists active connections and listening ports	Legacy compatibility
`ss`	Monitoring	Investigates sockets via kernel API	Speed & efficiency
`nc`	Testing	Establishes raw TCP/UDP connections	Simple connectivity validation
`nmap`	Security	Systematic port scanning and reconnaissance	Comprehensive discovery

STRATEGIC INSIGHT

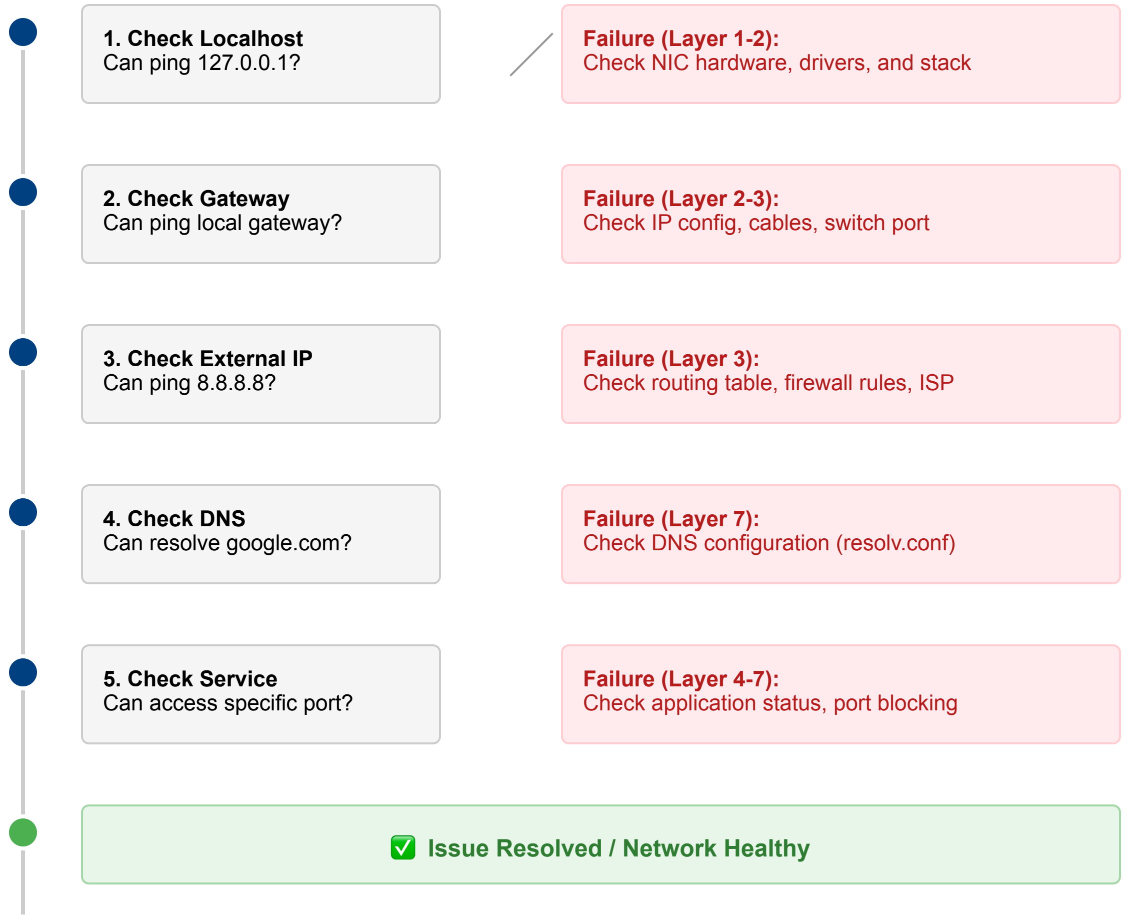
Distinguishing between **service availability** (checked via `ss/netstat` on localhost) and **network reachability** (checked via `nc/nmap` remotely) is crucial for isolating root causes. If a service is listening locally but unreachable via netcat, the issue lies in the network (firewall/routing), not the application.

Source: Linux Network Administrator's Guide; Enterprise Security Best Practices

Layer 7 - Application / Network Diagnostic Script

Systematic troubleshooting workflow and essential command-line diagnostics

Troubleshooting Logic Flow



1. Check Localhost
Can ping 127.0.0.1?

Failure (Layer 1-2):
Check NIC hardware, drivers, and stack

2. Check Gateway
Can ping local gateway?

Failure (Layer 2-3):
Check IP config, cables, switch port

3. Check External IP
Can ping 8.8.8.8?

Failure (Layer 3):
Check routing table, firewall rules, ISP

4. Check DNS
Can resolve google.com?

Failure (Layer 7):
Check DNS configuration (resolv.conf)

5. Check Service
Can access specific port?

Failure (Layer 4-7):
Check application status, port blocking

✅ **Issue Resolved / Network Healthy**

Diagnostic Toolset

Command	Function / Output Details
`curl -v [url]`	Verbose HTTP request; shows handshake, headers, SSL
`dig [domain]`	Detailed DNS lookup with query time and record authority
`nslookup [domain]`	Standard DNS query; interactive mode available
`host [domain]`	Simplified DNS lookup; returns IP address directly
`wget [url]`	Retrieves content/files; validates download capability

Automation Script Header

```
#!/bin/bash

# Initialize diagnostic sequence
echo "🔍 Network Diagnostics"
echo "======================"

# [Full logic from left column implements here]
```

KEY INSIGHT

Following a strict OSI-layered approach—starting from physical connectivity (Ping) up to application logic (Layer 7)—eliminates assumptions and drastically reduces troubleshooting time for complex connectivity issues.

Source: Network Administration Best Practices; DevOps Playbook

Check interfaces / Check gateway

Validation of network layer configuration and routing parameters to ensure connectivity

1. Network Interface Inspection

Retrieves active network interfaces and their assigned IP addresses. The command filters output to show only interface identifiers and inet addresses.

```
# Display header and filter IP address information
echo ""
echo "📡 Network Interfaces:"
ip addr show | grep -E "^[0-9]+:|inet "
```

2. Default Gateway Verification

Identifies the default route in the kernel routing table. This confirms the host knows where to send traffic destined for external networks.

```
# Check for default route entry
echo ""
echo "🚪 Default Gateway:"
ip route | grep default
```

KEY TAKEAWAY

Foundation of Connectivity: Verifying a valid IP assignment and an active default gateway is the primary step in network diagnostics. Without these, the host cannot communicate with the control plane or external APIs, rendering higher-level checks irrelevant.

Source: Network Configuration & Troubleshooting Guide; Linux IP Route Documentation

Test Local Connectivity / Test Gateway Connectivity

Automated validation sequence for network stack and upstream routing

Script Logic & Execution Flow

```
echo ""
echo "🏠 Local Connectivity (localhost):"
ping -c 1 127.0.0.1 > /dev/null && echo "✅ OK" || echo "❌ Failed"

# Extract default gateway IP
gateway=$(ip route | grep default | awk '{print $3}')

if [ -n "$gateway" ]; then
    echo ""
    echo "🔗 Gateway Connectivity ($gateway):"
    ping -c 1 $gateway > /dev/null && echo "✅ OK" || echo "❌ Failed"
fi
```

Functional Analysis & Validation Steps

1. Stack Integrity Verification

- Targets **localhost (127.0.0.1)** to verify the network adapter driver and TCP/IP stack functionality.
- Independent of physical cabling or external network status.

2. Dynamic Route Discovery

- Automatically identifies the next-hop router using `ip route`.
- Adapts to changing network environments (DHCP/Static) without hardcoded IPs.

3. Conditional Execution

- Implements logic check `if [ -n "$gateway" ]` to prevent execution errors if no route exists.
- Ensures script robustness in disconnected (air-gapped) environments.

4. User Feedback Mechanism

- Suppresses raw ping output (`> /dev/null`) for cleaner logs.
- Uses standard exit codes (`&&` / `||`) to print clear Status Indicators (✅ /❌).

KEY INSIGHT: ISOLATION STRATEGY

This two-step verification protocol enables rapid troubleshooting by isolating **local configuration issues** from **upstream network failures**. If the localhost check fails, the issue is internal (OS/Driver); if only the gateway check fails, the issue is external (Router/Switch).

Source: Network Validation Script Repository

Test external connectivity

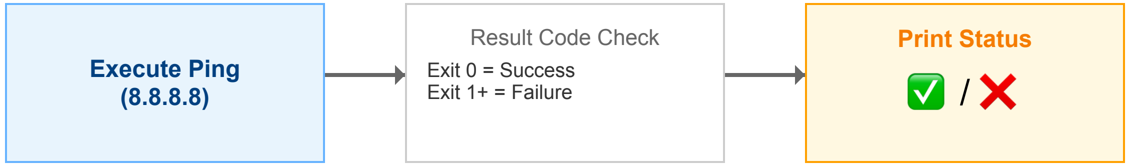
Validation logic for ensuring outbound network reachability via command line interface

Validation Strategy

Objective

- **Verify Outbound Routing:** Ensures the local environment can route traffic through the network gateway to the public internet.
- **Target Selection:** Uses Google Public DNS (`8.8.8.8`) as a highly available, standard endpoint for connectivity testing.
- **Minimal Footprint:** Sends a single ICMP packet to minimize network traffic and execution time during automated checks.

Execution Flow



Script Implementation

```
# Print header and execute check
echo ""
echo "🌐 External Connectivity (8.8.8.8):"
ping -c 1 8.8.8.8 > /dev/null && echo "✅ OK" || echo "❌ Failed"
```

Syntax Breakdown

Component	Function
`ping -c 1`	Send exactly one packet (count=1) to define success quickly
`> /dev/null`	Suppress standard technical output to keep logs clean
`&& echo "..."`	Conditional AND: Executes only if ping returns exit code 0 (Success)
`|| echo "..."`	Conditional OR: Executes only if ping returns non-zero (Failure)

KEY INSIGHT

This binary check provides an immediate "Go/No-Go" status for the network layer. A failure here ("❌ Failed") is a blocking issue that indicates the host cannot reach the internet, requiring immediate investigation into firewall rules, NAT gateway status, or route table configuration before proceeding with application deployment.

Source: System Administration Best Practices; GNU Coreutils Documentation

Network Diagnostic Verification: DNS & Ports

Automated validation of external name resolution and internal service listener status

01. DNS Resolution Test

Verifies the system's ability to resolve external domain names using the `host` command. This check confirms upstream DNS connectivity.

```
host google.com > /dev/null 2>&1
```

Target: google.com (Connectivity Benchmark)

02. Listening Port Analysis

Audits active network sockets using `ss` (Socket Statistics). This identifies which services are listening for incoming connections.

```
ss -tuln | head -10
```

Flags: TCP (-t), UDP (-u), Listening (-l), Numeric (-n)

Execution Log Output

```
# Executing diagnostics...

📖 DNS Resolution (google.com):
✅ OK

🚀 Listening Ports:
Netid  State   Recv-Q  Send-Q   Local Address:Port    Peer Address:Port
tcp    LISTEN  0       128      0.0.0.0:22            0.0.0.0:*
... [truncating output for display]

======================
```

DIAGNOSTIC ASSESSMENT

Successful execution of these checks confirms the server has valid external reachability (DNS) and provides visibility into the active attack surface (Open Ports), prerequisites for a secure and functional deployment.

Source: System Administration / DevOps Standard Operating Procedures

Troubleshooting Guide & Resource Library

Layered diagnostic framework, symptom reference, and curated learning materials for network engineers

Common Issues by OSI Layer

Layers 1-2 (Physical/Data)

- Cable disconnected / damaged
- NIC hardware failure
- Wrong VLAN assignment
- Duplex mismatch

Layer 3 (Network)

- Wrong IP configuration
- Subnet mask mismatch
- Missing/Incorrect gateway
- Routing table errors

Layer 4 (Transport)

- Port blocked (Security Group)
- Firewall rules dropping packets
- Service/Process not running
- Connection timeout

Layer 7 (Application)

- DNS resolution failure
- SSL/TLS Certificate expiry
- Application misconfiguration
- Authentication errors

Quick Troubleshooting Reference

Symptom	Layer	Check
No link light	1	Cable, port, NIC
Can't reach local devices	2	MAC address, switch, VLAN
Can't reach other networks	3	IP address, subnet, gateway, routing
Connection refused	4	Port, firewall, service status
Timeout on specific service	4-7	Service running, firewall, DNS
SSL/TLS errors	7	Certificate, date/time, cipher suite
DNS not resolving	7	DNS server, /etc/resolv.conf

DIAGNOSTIC STRATEGY

Adopting a systematic "bottom-up" approach (Layer 1 to 7) eliminates physical and network configuration issues before debugging complex application logic.

Recommended Resources & Tools

📖 Official Documentation

- RFC 1122 - Internet Host Requirements
- RFC 791 - Internet Protocol
- RFC 793 - Transmission Control Protocol

🎓 Learning Resources

- GeeksforGeeks - OSI vs TCP/IP Model
- Network Notes - OSI Model Explained
- freeCodeCamp - TCP/IP Layers
- JMU - Five Layer Model

🔧 Interactive Tools

- Subnet Calculator
- Wireshark (Packet Analyzer)
- Nmap (Network Scanner)
- Postman (API Testing)

📚 Recommended Books

- *Computer Networking* (Kurose & Ross)
- *TCP/IP Illustrated* (Stevens)
- *Network Warrior* (Donahue)

🎥 Video Resources

- Network Direction (YouTube)
- Professor Messer - Network+
- Practical Networking (YouTube)

☁ Cloud Networking

- AWS VPC Documentation
- GCP Networking Overview
- Azure Networking Documentation

Source: IETF RFC Database; Cloud Provider Documentation (AWS, GCP, Azure); Standard Networking Curriculum

Quick Reference Card

MLOps Foundation Series • Network Models & Ports

OSI Model (7 Layers)

Layer 7: Application
End-user processes (HTTP, FTP, DNS)

Layer 6: Presentation
Data formatting & encryption (SSL, JPEG)

Layer 5: Session
Interhost communication (Sessions)

Layer 4: Transport
End-to-end delivery (TCP, UDP) [Ports]

Layer 3: Network
Path determination (IP) [IP Addr]

Layer 2: Data Link
Physical addressing (Ethernet) [MAC]

Layer 1: Physical
Media, signal, binary transmission

TCP/IP Model (Mapped)

Layer 5: Application

Combines OSI Application, Presentation, and Session layers.

- Protocols: HTTP, FTP, SSH, DNS, SMTP
- Focus: Data generation and encoding

Layer 4: Transport
TCP (Reliable), UDP (Fast/Stateless)

Layer 3: Network
IP Addressing & Packet Routing

Layer 2: Data Link
Frame handling & MAC Addressing

Layer 1: Physical
Hardware transmission media

Essential Ports Dictionary

PORT	SERVICE / PROTOCOL
22	SSH (Secure Shell)
25	SMTP (Email)
53	DNS (Domain Name System)
80	HTTP (Web Traffic)
443	HTTPS (Secure Web)
3306	MySQL Database
5000	Flask App (Default)
5432	PostgreSQL Database
6379	Redis Cache
8080	HTTP Alternative / Proxy
9090	Prometheus Metrics
27017	MongoDB

PRACTICAL IMPLICATION: MAPPING LAYERS TO INFRASTRUCTURE

While the OSI model provides the theoretical framework for network isolation, MLOps engineers primarily operate at the **TCP/IP Transport Layer (Layer 4)** when configuring Security Groups and Firewalls (managing Ports). Understanding the correlation between specific services (e.g., PostgreSQL) and their default ports (5432) is critical for debugging connectivity in distributed cloud environments (VPCs).

Source: MLOps Foundation Series | Last Updated: January 2026 | Next Session: Cloud Networking with AWS VPCs

Thank You