Networking for DevOps

Fundamentals to Azure Cloud Networking

Complete Study Guide

OSI Model | TCP/UDP | IP & CIDR | DNS | HTTP | Ports | Azure Networking

DevOps Engineer Job Preparation

TABLE OF CONTENTS

PART 1
NETWORKING FUNDAMENTALS

1. OSI Model (Open Systems Interconnection)

1.1 What is the OSI Model?

The OSI model is a conceptual framework that standardizes how different network protocols and systems communicate. It divides networking into 7 layers, each with specific responsibilities. Think of it as a blueprint for how data travels from your application to the network and back.

Why DevOps Engineers Need to Know This: When you're troubleshooting why a container can't talk to a database, or why your load balancer isn't working, understanding OSI layers helps you quickly identify where the problem is — application layer? Network layer? Transport layer?

The 7 Layers (Remember: Please Do Not Throw Sausage Pizza Away)

Layer	Name	Protocols	Responsibility	Examples
7	Application	HTTP, HTTPS, FTP, DNS, SMTP, SSH	What users interact with directly	Web browsers, email clients, REST APIs
6	Presentation	SSL/TLS, encryption, data formatting	Translates data formats, encryption	JPEG, PNG, encryption of HTTPS
5	Session	NetBIOS, RPC, session management	Establishes, maintains, terminates connections	Login sessions, database connections
4	Transport	TCP, UDP	Reliable/unreliable data transfer, ports	Port 443 for HTTPS, Port 22 for SSH
3	Network	IP, ICMP, routing	Logical addressing, routing between networks	Routers, IP addresses (192.168.1.1)
2	Data Link	Ethernet, Wi-Fi, MAC addresses	Physical addressing within local network	Switches, MAC addresses, VLANs
1	Physical	Cables, Wi-Fi signals, NICs	Actual physical connection	Ethernet cables, fiber optics, radio waves

Real-World DevOps Example

When you type 'https://api.company.com/users' in your browser and hit enter:

Layer 7 (Application): Browser creates HTTP GET request

Layer 6 (Presentation): SSL/TLS encrypts the request

Layer 5 (Session): Establishes session with the server

Layer 4 (Transport): TCP breaks data into segments, adds port 443

Layer 3 (Network): IP adds source/destination IP addresses for routing

Layer 2 (Data Link): Ethernet adds MAC addresses for local network

Layer 1 (Physical): Data converted to electrical signals on the wire

The server receives it in reverse order (1→7) and sends response back the same way.

Troubleshooting by Layer (DevOps Application)

Layer	Symptom	What to Check
Layer 7	Application not responding	Check: app logs, HTTP status codes, API timeouts
Layer 4	Connection refused	Check: is port open? Firewall blocking? Service running?
Layer 3	Cannot ping server	Check: IP address correct? Routing tables? VPN connected?
Layer 2	No network access	Check: Ethernet cable plugged in? WiFi connected? MAC address filtered?
Layer 1	Physical connection issue	Check: Cable damaged? Network card working? Port disabled?

📌 **NOTE:** For DevOps, focus on Layers 3, 4, and 7. These are where 90% of issues occur: IP/routing problems (L3), port/firewall issues (L4), and application errors (L7).

2. TCP vs UDP

2.1 Understanding Transport Layer Protocols

TCP and UDP are both Layer 4 (Transport) protocols. They both move data between applications on different machines, but they work very differently. Choosing the wrong one can break your application or waste resources.

TCP (Transmission Control Protocol)

TCP is connection-oriented and reliable. It guarantees that data arrives in order and without errors. Think of it like a phone call — you establish connection, talk, and hang up.

How TCP Works

Three-Way Handshake (Connection Establishment):

- Client sends SYN (synchronize) to server
- Server responds with SYN-ACK (acknowledge)
- Client sends ACK back
- Connection established — now data can flow

Data Transfer with Acknowledgment:

- Every packet sent must be acknowledged by receiver
- If acknowledgment not received, packet is retransmitted
- Packets are numbered, so receiver can reorder them if needed

Four-Way Handshake (Connection Termination):

- Both sides send FIN (finish) and ACK to close gracefully

TCP Features

Reliable — guaranteed delivery, packets retransmitted if lost

Ordered — data arrives in the same order it was sent

Error-checked — checksum detects corrupted packets

Flow control — sender doesn't overwhelm receiver

Congestion control — adjusts sending rate based on network conditions

UDP (User Datagram Protocol)

UDP is connectionless and unreliable (but fast!). It sends data without establishing a connection or confirming delivery. Think of it like sending postcards — you drop them in the mail and hope they arrive.

How UDP Works

No connection setup — just send packets immediately

No acknowledgment — sender doesn't know if packet arrived

No guarantee of order — packets may arrive out of sequence

No retransmission — lost packets are lost forever

▸ **UDP Features**

Fast — no handshake overhead, no waiting for acknowledgments

Lightweight — smaller packet headers (8 bytes vs TCP's 20 bytes)

No connection state — server doesn't track connections, scales better

Broadcast/Multicast support — send one packet to multiple receivers

TCP vs UDP — When to Use Which

Feature	TCP	UDP
Connection	Connection-oriented (3-way handshake)	Connectionless (no handshake)
Reliability	Guaranteed delivery, retransmits lost packets	No guarantee, packets can be lost
Ordering	Data arrives in order	Packets may arrive out of order
Speed	Slower (acknowledgment overhead)	Faster (no overhead)
Error Checking	Extensive	Basic checksum only
Use Cases	HTTP, HTTPS, FTP, SSH, databases, email	DNS, video streaming, VoIP, gaming, DHCP
When to Use	When you MUST have reliable delivery	When speed matters more than reliability

Real DevOps Examples

▸ **Example 1: Why DNS Uses UDP**

DNS queries are tiny (usually <512 bytes) and need to be fast. If a DNS packet is lost, the client just resends the query — much simpler than maintaining TCP connections for millions of DNS queries per second. However, for large DNS responses (like zone transfers), TCP is used.

▸ **Example 2: Why Databases Use TCP**

When your application queries a database, you absolutely need every byte of data. If even one character is missing from a SQL query result, the application will break. TCP's guaranteed delivery is essential. PostgreSQL, MySQL, MongoDB all use TCP.

▸ **Example 3: Why Live Video Streaming Uses UDP**

In a live stream or video call, if a packet is lost, retransmitting it 3 seconds later is useless — the moment has passed. Better to skip that frame and continue. UDP's speed matters more than perfection. Netflix, Zoom, Twitch all use UDP-based protocols.

> **TIP:** In interviews, if asked 'Should I use TCP or UDP?', think: 'Can I tolerate data loss?' If no → TCP. If speed is critical and loss is acceptable → UDP.

3. IP Addressing & CIDR

3.1 IP Addresses

An IP address uniquely identifies a device on a network. IPv4 uses 32-bit addresses (4 numbers from 0-255). IPv6 uses 128-bit addresses (we'll focus on IPv4 as it's still dominant).

IPv4 Address Structure

```
192.168.1.100
```

This is four octets (8 bits each) separated by dots. Each octet ranges from 0 to 255.

```
Binary: 11000000.10101000.00000001.01100100

Decimal: 192     .168     .1      .100
```

Public vs Private IP Addresses

Type	Visibility	Characteristics	Examples
Public IP	Routable on the internet	Assigned by ISP, globally unique	8.8.8.8, 40.112.72.205
Private IP	Only within local network	NOT routable on internet, reused in different networks	10.x.x.x, 172.16-31.x.x, 192.168.x.x

Private IP Ranges (RFC 1918)

```
10.0.0.0 to 10.255.255.255     (10.0.0.0/8)     — 16 million addresses

172.16.0.0 to 172.31.255.255   (172.16.0.0/12)  — 1 million addresses

192.168.0.0 to 192.168.255.255 (192.168.0.0/16) — 65,536 addresses
```

> 📌 **NOTE:** Your home router likely uses 192.168.1.x. Azure VNets commonly use 10.0.0.0/16 or 172.16.0.0/16.

Special IP Addresses

Address	Purpose
127.0.0.1	Localhost / Loopback — always refers to current machine

0.0.0.0	All interfaces / default route in routing tables
255.255.255.255	Broadcast address — send to all devices on local network
169.254.x.x	APIPA (Automatic Private IP) — self-assigned when DHCP fails

3.2 CIDR (Classless Inter-Domain Routing)

CIDR notation (e.g., 192.168.1.0/24) defines an IP range using a network prefix. It replaced the old 'Class A/B/C' system and is essential for subnetting in cloud environments.

Understanding CIDR Notation

```
192.168.1.0/24
```

192.168.1.0 = network address (first address in range)

/24 = subnet mask — first 24 bits are network, last 8 bits are hosts

Subnet mask: 255.255.255.0 (binary: 11111111.11111111.11111111.00000000)

Number of addresses = 2^(32-24) = 2^8 = 256 addresses

Usable addresses = 256 - 2 = 254 (first is network, last is broadcast)

Address range: 192.168.1.0 to 192.168.1.255

Common CIDR Blocks

CIDR	Subnet Mask	Total IPs	Usable Hosts	Typical Use
/32	255.255.255.255	1	1 (single host)	Firewall rule for specific IP
/31	255.255.255.254	2	2 (point-to-point)	Router-to-router links
/30	255.255.255.252	4	2	Small point-to-point connections
/29	255.255.255.248	8	6	Very small subnet
/28	255.255.255.240	16	14	Small office network
/27	255.255.255.224	32	30	Small subnet
/26	255.255.255.192	64	62	Medium subnet
/25	255.255.255.128	128	126	Medium-large subnet
/24	255.255.255.0	256	254	Standard subnet (254 hosts)
/23	255.255.254.0	512	510	Larger subnet
/22	255.255.252.0	1,024	1,022	Large subnet
/21	255.255.248.0	2,048	2,046	Very large subnet
/20	255.255.240.0	4,096	4,094	Huge subnet
/16	255.255.0.0	65,536	65,534	Azure VNet default size
/8	255.0.0.0	16,777,216	16,777,214	Entire Class A network

CIDR Calculation Examples

- **Example 1: 10.0.0.0/16**

  Network: 10.0.0.0

  Subnet mask: 255.255.0.0

  First 16 bits locked, last 16 bits available for hosts

  Total addresses: 2^16 = 65,536

  Range: 10.0.0.0 to 10.0.255.255

  Usable: 10.0.0.1 to 10.0.255.254 (65,534 hosts)

  Common for: Azure VNet address space

- **Example 2: 172.16.5.0/24**

  Network: 172.16.5.0

  Subnet mask: 255.255.255.0

  Range: 172.16.5.0 to 172.16.5.255

  Usable: 172.16.5.1 to 172.16.5.254 (254 hosts)

  Common for: Azure subnet within VNet

Subnetting in Azure (Real Example)

You have Azure VNet with 10.0.0.0/16. You want to create multiple subnets:

```
VNet:               10.0.0.0/16          (10.0.0.0 to 10.0.255.255)


Subnet 1 (Web):     10.0.1.0/24          (10.0.1.0 to 10.0.1.255) —
254 IPs

Subnet 2 (App):     10.0.2.0/24          (10.0.2.0 to 10.0.2.255) —
254 IPs

Subnet 3 (DB):      10.0.3.0/24          (10.0.3.0 to 10.0.3.255) —
254 IPs

Subnet 4 (Gateway):10.0.4.0/27           (10.0.4.0 to 10.0.4.31) —
30 IPs
```

📌 **NOTE:** Azure reserves 5 IP addresses in each subnet (first 4 and last 1). So a /24 subnet gives you 254 - 5 = 249 usable IPs.

💡 **TIP:** For interviews, memorize: /16 ≈ 65k IPs, /24 = 256 IPs, /32 = single IP.

4. DNS (Domain Name System)

4.1 What is DNS?

DNS is the internet's phone book. It translates human-readable domain names (google.com) into machine-readable IP addresses (142.250.185.46). Without DNS, you'd have to memorize IP addresses for every website.

How DNS Works (Step-by-Step)

1. User types 'www.company.com' in browser
2. Browser checks local cache — is IP already known?
3. If not cached, query sent to recursive DNS resolver (usually ISP's DNS or 8.8.8.8)
4. Resolver checks its cache. If not found, queries root DNS servers
5. Root server responds: 'For .com domains, ask the .com TLD servers'
6. Resolver queries .com TLD server: 'Where is company.com?'
7. TLD responds: 'Ask the authoritative name server for company.com'
8. Resolver queries authoritative server for company.com
9. Authoritative server responds with IP: 40.112.72.205
10. Resolver caches the result and returns IP to browser
11. Browser connects to 40.112.72.205

All this happens in milliseconds!

DNS Record Types

Record Type	Purpose	Example
A (Address)	Maps domain to IPv4 address	www.company.com → 40.112.72.205
AAAA (Quad-A)	Maps domain to IPv6 address	www.company.com → 2001:db8::1
CNAME (Canonical Name)	Maps domain to another domain (alias)	blog.company.com → company.github.io
MX (Mail Exchange)	Specifies mail servers for domain	company.com → mail.google.com (priority 10)
TXT (Text)	Stores text data for various purposes	SPF records, domain verification, DKIM
NS (Name Server)	Specifies authoritative name servers	company.com → ns1.azure-dns.com
PTR (Pointer)	Reverse DNS — maps IP to domain name	40.112.72.205 → server.company.com

SRV (Service)	Specifies location of services	_sip._tcp.company.com → sipserver.com:5060
SOA (Start of Authority)	Administrative info about zone	Primary name server, admin email, TTL

Real DevOps Examples

▸ Example 1: A Record for Web Server

```
# DNS Zone: company.com
www       A    40.112.72.205    # Web server IP
api       A    40.112.73.120    # API server IP
@         A    40.112.72.205    # Root domain points to web server
```

▸ Example 2: CNAME for Subdomains

```
# Point multiple subdomains to same Azure App Service
www       CNAME   myapp.azurewebsites.net
app       CNAME   myapp.azurewebsites.net
staging   CNAME   myapp-staging.azurewebsites.net
```

▸ Example 3: MX Records for Email

```
# Route email to Microsoft 365
@  MX  10  company-com.mail.protection.outlook.com
@  MX  20  backup-mx.company.com
```

Lower priority number = preferred mail server

▸ Example 4: TXT for Domain Verification

```
# Prove you own the domain to Azure
@  TXT  MS=ms12345678


# SPF record to prevent email spoofing
@  TXT  v=spf1 include:_spf.microsoft.com ~all
```

DNS TTL (Time To Live)

TTL tells DNS resolvers how long to cache the record (in seconds).

TTL	Use Case	Example
3600 (1 hour)	Standard for stable records	Production servers that rarely change

300 (5 minutes)	For records that might change	During migration or testing
60 (1 minute)	For active DNS changes	Blue-green deployments, failover testing
86400 (24 hours)	For very stable records	Name servers, rarely-changing infrastructure

📌 **NOTE:** Before making DNS changes in production, lower TTL to 300 (5 min) a day in advance. After change, raise it back to 3600 to reduce query load.

Public DNS Resolvers (Memorize These)

```
Google Public DNS:       8.8.8.8 and 8.8.4.4
Cloudflare DNS:          1.1.1.1 and 1.0.0.1
Quad9:                   9.9.9.9
Azure DNS:               168.63.129.16 (internal Azure VNet DNS)
```

💡 **TIP:** If DNS is not resolving, test with: nslookup google.com 8.8.8.8

5. HTTP Status Codes

5.1 Understanding HTTP Status Codes

HTTP status codes are 3-digit numbers returned by web servers to indicate the result of a request. As a DevOps engineer, you'll see these in logs constantly and need to troubleshoot based on them.

Status Code Categories

Range	Category	Meaning	Who's Responsible
1xx	Informational	Request received, continuing process	Rarely seen in practice
2xx	Success	Request successfully received, understood, and accepted	Everything worked
3xx	Redirection	Further action needed to complete request	Resource moved
4xx	Client Error	Client sent a bad request	User's fault
5xx	Server Error	Server failed to fulfill valid request	Server's fault

Most Important Status Codes (Memorize These)

▸ 2xx — Success

Code	Meaning	Example
200 OK	Request succeeded. Most common success code.	GET /api/users → returns user list
201 Created	Resource was successfully created.	POST /api/users → new user created
202 Accepted	Request accepted for processing but not complete yet.	Async job submitted
204 No Content	Success but no content to return.	DELETE /api/users/123 → success, nothing to return

▸ 3xx — Redirection

Code	Meaning	Example
301 Moved Permanently	Resource permanently moved to new URL. Update bookmarks!	http://site.com → https://site.com
302 Found	Temporary redirect. Original URL still valid.	Maintenance redirect
304 Not Modified	Cached version is still valid. Don't download again.	Browser cache is fresh
307 Temporary Redirect	Like 302 but preserves HTTP method (POST stays POST).	Temporary failover

▸ 4xx — Client Errors

Code	Meaning	Example
400 Bad Request	Invalid request syntax, malformed JSON, missing fields.	POST with invalid JSON body
401 Unauthorized	Authentication required. User not logged in.	Accessing /admin without token
403 Forbidden	Authenticated but not authorized. You don't have permission.	Regular user accessing admin API
404 Not Found	Resource doesn't exist. Wrong URL or resource deleted.	GET /api/users/999999
405 Method Not Allowed	HTTP method not supported on this endpoint.	POST to read-only endpoint
408 Request Timeout	Server timed out waiting for request.	Slow client connection
429 Too Many Requests	Rate limit exceeded. Too many requests too fast.	API throttling

▸ 5xx — Server Errors

Code	Meaning	Example
500 Internal Server Error	Generic server error. Something crashed.	Unhandled exception in code
502 Bad Gateway	Upstream server returned invalid response. Gateway/proxy can't reach app.	App server down, load balancer can't connect
503 Service Unavailable	Server overloaded or down for maintenance.	Deployment in progress, server restarting

504 Gateway Timeout	Upstream server didn't respond in time. Slow backend.	Database query took 60 seconds, timeout at 30

Real DevOps Scenarios

▸ **Scenario 1: 502 Bad Gateway During Deployment**

You deploy a new version. Users immediately get 502 errors.

Cause: Application hasn't started yet, or crashed on startup. Load balancer forwards requests but app isn't listening.

Fix: Check app logs. Is app running? Port correct? Health check endpoint responding?

▸ **Scenario 2: 504 Gateway Timeout on API**

API works fine usually, but sometimes returns 504.

Cause: Database query is slow or times out. Application Gateway timeout (default 30 seconds) is reached.

Fix: Optimize slow queries. Add database indexes. Increase Gateway timeout if queries legitimately take longer.

▸ **Scenario 3: 429 Too Many Requests**

Mobile app users complain they can't log in during peak hours.

Cause: API rate limiting kicked in. Too many requests from same IP/user.

Fix: Increase rate limits if legitimate traffic. Add caching to reduce backend calls. Implement exponential backoff in client.

> **TIP:** In interviews, if asked 'What's the difference between 401 and 403?', answer: '401 means NOT logged in (no credentials or invalid credentials). 403 means logged in but FORBIDDEN (not authorized for this resource).'

6. Important Port Numbers

6.1 Well-Known Ports (0-1023)

Well-known ports are standardized and used by common services. You MUST memorize these for DevOps interviews and daily troubleshooting.

Essential Ports (Memorize These!)

Port	Service	Description	Common Use
20/21	FTP	File Transfer Protocol (data/control)	Legacy file transfer
22	SSH	Secure Shell — remote server access	ssh user@server
23	Telnet	Unencrypted remote access (NEVER use!)	Replaced by SSH
25	SMTP	Email sending (Simple Mail Transfer Protocol)	Mail servers
53	DNS	Domain Name System (UDP primary, TCP for zone transfers)	DNS queries
80	HTTP	Unencrypted web traffic	http://website.com
110	POP3	Email retrieval (Post Office Protocol)	Older email protocol
143	IMAP	Email retrieval (better than POP3)	Modern email
443	HTTPS	Encrypted web traffic (HTTP over TLS)	https://website.com
445	SMB	Windows file sharing (Server Message Block)	File shares
3306	MySQL	MySQL database default port	Database connections
3389	RDP	Remote Desktop Protocol (Windows)	Remote into Windows VM
5432	PostgreSQL	PostgreSQL database	Database connections
6379	Redis	Redis cache and message broker	Caching layer
8080	HTTP-Alt	Alternative HTTP (Tomcat, Jenkins, test servers)	Dev/test environments
27017	MongoDB	MongoDB NoSQL database	Database connections

Kubernetes & Container Ports

Port(s)	Service	Description	Usage
2379/2380	etcd	Kubernetes cluster state database	K8s control plane
6443	Kubernetes API	API server secure port	kubectl commands
10250	kubelet	Kubelet API (on each node)	Node management
30000-32767	NodePort	K8s NodePort service range	External access to services

Azure-Specific Ports

Port	Service	Usage	Azure Context
3389	RDP	Remote Desktop to Windows VMs	Windows VM access
22	SSH	SSH to Linux VMs	Linux VM access
1433	SQL Server	Azure SQL Database	Database connections
5671/5672	AMQP	Azure Service Bus, Event Hubs	Message queuing

DevOps Tools Ports

Port	Tool	Purpose	DevOps Use
8080	Jenkins	Jenkins web UI and API	CI/CD server
9090	Prometheus	Prometheus metrics server	Monitoring
3000	Grafana	Grafana dashboards	Visualization
9200/9300	Elasticsearch	Elasticsearch HTTP API / cluster communication	Log aggregation
5601	Kibana	Kibana web UI	Log visualization
2375/2376	Docker	Docker daemon API (unsecured/TLS)	Container management

Security Note on Ports

⚠ **WARNING:** NEVER expose ports like 22 (SSH), 3389 (RDP), 3306 (MySQL) directly to the internet. Use VPN, bastion hosts, or Azure Bastion instead. Bots constantly scan for these open ports.

Real DevOps Example: Firewall Rule

```
# Azure NSG rule to allow HTTPS from internet
Priority: 100
Source: Internet (or specific IP)
Destination: Any
Port: 443
Protocol: TCP
Action: Allow

# Allow SSH only from corporate VPN
Priority: 200
Source: 203.0.113.0/24  (corporate network)
Destination: Any
Port: 22
Protocol: TCP
Action: Allow
```

TIP: To test if a port is open: telnet server-ip 443 or nc -zv server-ip 443

PART 2
AZURE CLOUD NETWORKING

7. Azure Virtual Network (VNet)

7.1 What is Azure VNet?

Azure Virtual Network (VNet) is your private network in the cloud. It's logically isolated from other VNets and the internet. Think of it as your own data center network in Azure — you define the IP address space, create subnets, and control traffic flow.

Key Concepts

Isolation — VNet is private and isolated by default. Resources in different VNets cannot communicate unless explicitly connected.

Address Space — Define your own IP ranges (e.g., 10.0.0.0/16) using private IPs from RFC 1918.

Subnets — Divide VNet into smaller segments for organization and security.

Region-Specific — VNet exists in one Azure region, but can connect to VNets in other regions via peering.

Creating a VNet (Conceptual Steps)

Choose address space (e.g., 10.0.0.0/16 gives 65,536 IPs)

Create subnets within that address space:

- Frontend subnet: 10.0.1.0/24 (254 usable IPs)
- Backend subnet: 10.0.2.0/24
- Database subnet: 10.0.3.0/24

Deploy resources (VMs, App Services, AKS) into appropriate subnets

Configure Network Security Groups (NSGs) for firewall rules

Real DevOps Example: Multi-Tier Application VNet

```
VNet Name: ProductionVNet

Address Space: 10.1.0.0/16


Subnets:

  - WebTier:       10.1.1.0/24    (Web servers, load balancer)

  - AppTier:       10.1.2.0/24    (Application servers, APIs)

  - DataTier:      10.1.3.0/24    (Databases, cache)

  - GatewaySubnet: 10.1.4.0/27   (VPN Gateway — special subnet name
required)

  - AzureBastionSubnet: 10.1.5.0/27  (Azure Bastion — special name
required)
```

📌 **NOTE:** GatewaySubnet and AzureBastionSubnet are special subnet names required by Azure. Don't rename them.

7.2 Subnets

Subnets divide a VNet into smaller networks. Each subnet is a range of IP addresses within the VNet address space.

Why Use Subnets?

Security Segmentation — Separate web, app, and database tiers with different security rules

Organization — Group resources by function, environment, or team

NSG Application — Apply different firewall rules to different subnets

Routing Control — Route traffic differently based on subnet

Azure Reserved IPs in Subnets

Azure reserves 5 IP addresses in EVERY subnet:

IP Address	Purpose
x.x.x.0	Network address (not usable)
x.x.x.1	Reserved by Azure for default gateway
x.x.x.2	Reserved by Azure for DNS
x.x.x.3	Reserved by Azure for DNS
x.x.x.255	Broadcast address (not usable)

So a /24 subnet (256 IPs) has only 251 usable IPs, not 256.

Subnet Sizing Guidelines

CIDR	Total IPs	Usable (minus Azure reserved 5)	Typical Use
/28 (16 IPs)	11 usable	Small subnet for gateways, bastion	
/27 (32 IPs)	27 usable	GatewaySubnet, BastionSubnet minimum size	
/26 (64 IPs)	59 usable	Small application tier	
/25 (128 IPs)	123 usable	Medium subnet	
/24 (256 IPs)	251 usable	Standard subnet — good for most cases	
/23 (512 IPs)	507 usable	Large subnet for scale-out apps	
/22 (1024 IPs)	1019 usable	Very large subnet or AKS clusters	

TIP: For AKS clusters, plan for growth. A /24 subnet supports ~30 nodes. Use /22 or /20 for production AKS.

8. Network Security Groups (NSG)

8.1 What is an NSG?

Network Security Group is Azure's distributed firewall. It filters network traffic to and from Azure resources based on rules you define. Think of it as iptables or firewalld in Linux, but managed by Azure.

How NSG Works

Rule-Based Filtering — Define allow/deny rules based on source, destination, port, protocol

Priority-Based Evaluation — Rules processed in priority order (100-4096, lower number = higher priority)

Stateful — If you allow inbound on port 443, return traffic is automatically allowed

Applied to Subnet or NIC — Can attach NSG to subnet (affects all resources) or individual network interface (affects one VM)

NSG Rule Structure

Component	Description
Priority	100-4096 (lower = higher priority). Process in order until match found.
Name	Descriptive name for the rule (e.g., 'AllowHTTPS')
Port	Single port (443), range (8000-9000), or * (all)
Protocol	TCP, UDP, ICMP, Any
Source	IP address, CIDR range, Service Tag (Internet, VirtualNetwork, AzureLoadBalancer)
Destination	IP address, CIDR range, Service Tag
Action	Allow or Deny
Direction	Inbound (traffic coming TO resource) or Outbound (traffic leaving FROM resource)

Example NSG Rules

▸ Web Server NSG (Applied to WebTier Subnet)

```
Priority | Name              | Port     | Protocol | Source      |
Destination | Action | Direction

100      | AllowHTTPS        | 443      | TCP      | Internet    |
Any         | Allow  | Inbound

110      | AllowHTTP         | 80       | TCP      | Internet    |
Any         | Allow  | Inbound

200      | AllowSSHFromVPN   | 22       | TCP      | 203.0.113.0/24
| Any      | Allow  | Inbound

300      | DenyAllInbound    | *        | *        | *           | *
| Deny   | Inbound
```

▸ Database Subnet NSG

```
Priority | Name              | Port     | Protocol | Source
| Destination | Action | Direction

100      | AllowFromAppTier | 5432     | TCP      | 10.1.2.0/24
| Any         | Allow  | Inbound

200      | DenyAllElse       | *        | *        | *
| *           | Deny   | Inbound
```

This ensures database only accepts connections from the App subnet, nowhere else.

Default NSG Rules (Cannot Delete)

Every NSG has default rules at the bottom (priority 65000+). You can't delete them, but your custom rules override them.

Default Rule	Priority	Action	Meaning
AllowVNetInbound	65000	Allow	Traffic within VNet allowed by default
AllowAzureLoadBalancerInbound	65001	Allow	Azure Load Balancer health probes

DenyAllInbound	65500	Deny	Deny everything else (unless your rules allow it)

Service Tags (Simplified Source/Destination)

Instead of specifying IP ranges, use service tags:

Service Tag	What It Represents
Internet	Any public internet IP address
VirtualNetwork	All IPs within the VNet and connected VNets
AzureLoadBalancer	Azure's load balancer infrastructure
Storage	Azure Storage service IP ranges
Sql	Azure SQL Database IP ranges
AzureCloud	All Azure datacenter IPs

NSG Best Practices

Least Privilege — Start with deny-all, then allow only what's needed

Use Service Tags — Easier than maintaining IP lists

Document Rules — Use descriptive names like 'AllowHTTPSFromCDN' not 'Rule1'

Subnet-Level NSGs — Apply to subnet for all resources, override with NIC-level NSG if needed

Audit Regularly — Review NSG rules quarterly, remove unused rules

⚠ **WARNING:** Common mistake: Opening port 22 or 3389 to 0.0.0.0/0 (entire internet). This is scanned and attacked within minutes. Use Azure Bastion or VPN instead.

9. Load Balancer vs Application Gateway

9.1 When to Use Which?

Azure has TWO main load balancing services. Choosing the wrong one is a common mistake. Here's how to decide:

Feature	Azure Load Balancer	Application Gateway
OSI Layer	Layer 4 (Transport)	Layer 7 (Application)
Traffic Type	Any TCP/UDP traffic	HTTP/HTTPS only
Routing	Based on IP address and port	Based on URL path, hostname, headers
SSL Termination	No (passes through)	Yes (offloads SSL from backend)

Web Application Firewall	No	Yes (optional WAF add-on)
WebSocket Support	Yes (generic TCP)	Yes (HTTP/1.1 upgrade)
Typical Use Case	Non-HTTP apps, VM load balancing	Web applications, API gateways, microservices
Cost	Lower	Higher
Example	Load balance database connections, RDP, SSH	Route /api to API servers, /images to static server

9.2 Azure Load Balancer (Layer 4)

How It Works

Load Balancer distributes traffic based on IP address and port. It doesn't understand HTTP — it just forwards TCP/UDP packets.

Client connects to Load Balancer public IP

Load Balancer picks a backend VM based on algorithm (hash, round-robin)

Traffic forwarded to VM private IP

VM responds directly to client (or through LB depending on type)

Load Balancer Types

Public Load Balancer — Has public IP. Routes internet traffic to VMs in backend pool.

Internal Load Balancer — Has private IP. Routes traffic within VNet (e.g., app tier → database cluster).

Load Balancer Components

Component	Purpose
Frontend IP	The IP address clients connect to (public or private)
Backend Pool	VMs or VMSS instances that receive traffic
Health Probe	Checks if backend is healthy (TCP, HTTP). Unhealthy backends removed from pool.
Load Balancing Rule	Port mapping: frontend port → backend port, distribution algorithm
Inbound NAT Rule	Forward specific port to specific VM (e.g., SSH to individual VMs)

Real Example: Database Load Balancer

```
Scenario: 3-node PostgreSQL cluster with read replicas


Internal Load Balancer:
  Frontend IP:       10.1.3.100 (private IP in DB subnet)
```

```
  Backend Pool:      db-vm-1, db-vm-2, db-vm-3
  Health Probe:      TCP port 5432 every 5 seconds
  LB Rule:           10.1.3.100:5432 → Backend:5432 (distribute
connections)

Application servers connect to 10.1.3.100:5432
Load Balancer distributes to healthy database nodes
```

9.3 Application Gateway (Layer 7)

How It Works

Application Gateway understands HTTP. It can route based on URL paths, hostnames, headers. It terminates SSL and can inspect/modify HTTP requests.

Key Features

URL-Based Routing — Send /api/* to API servers, /images/* to static file servers

Host-Based Routing — Route app.company.com and api.company.com to different backend pools

SSL Termination — Handle SSL encryption/decryption at gateway, send plain HTTP to backends

Web Application Firewall (WAF) — Protect against OWASP Top 10 (SQL injection, XSS, etc.)

Session Affinity — Route same user to same backend (sticky sessions)

Autoscaling — Scale out/in based on traffic

Redirection — HTTP to HTTPS redirect built-in

Application Gateway Components

Component	Purpose
Frontend IP	Public and/or private IP where clients connect
Listener	Listens on specific port (80, 443) and hostname
Backend Pool	VMs, VMSS, App Services, or IP addresses
HTTP Settings	Port, protocol, cookie-based affinity, timeout
Routing Rule	Links listener → backend pool with path-based or host-based routing
Health Probe	Custom HTTP/HTTPS probe to check backend health
SSL Certificate	Upload cert for HTTPS termination
WAF Policy	Optional firewall rules to block malicious requests

Real Example: Microservices Application Gateway

```
Frontend: appgateway.company.com (Public IP)

Routing Rules:
  / (root)            → Frontend Pool (React app VMs)
  /api/*              → API Pool (Node.js VMs)
  /auth/*             → Auth Pool (Authentication service)
  /static/*           → Storage Account (Azure Blob with CDN)

SSL Certificate: *.company.com
WAF: Enabled with OWASP 3.2 ruleset
Health Probe: GET /health every 30 seconds
```

When to Use Application Gateway

Web Applications — Multiple websites or services on same gateway

Microservices — Route different paths to different backend services

SSL Offload — Want to handle SSL at gateway, not at each backend

WAF Protection — Need to block SQL injection, XSS attacks

Cookie-Based Affinity — Stateful app needs same user → same server

When to Use Load Balancer

Non-HTTP Traffic — RDP, SSH, database connections, custom protocols

Simplicity & Cost — Basic TCP/UDP load balancing is all you need

Low Latency — Layer 4 has less overhead than Layer 7

Large Scale — Millions of connections, Load Balancer handles higher throughput

📌 **NOTE:** You can use BOTH together! Application Gateway for web traffic → Load Balancer for backend database cluster.

10. VNet Peering

10.1 What is VNet Peering?

VNet Peering connects two Azure VNets so they can communicate privately as if they're on the same network. Traffic stays on Microsoft's backbone network — never goes over public internet.

Types of VNet Peering

VNet Peering (Regional) — Connect VNets in the same Azure region

Global VNet Peering — Connect VNets across different Azure regions

How VNet Peering Works

Create peering connection from VNet A to VNet B

Create reverse peering from VNet B to VNet A (peering is not automatic bi-directional!)

Resources in VNet A can now reach resources in VNet B using private IPs

No gateway needed, no encryption (but traffic never leaves Microsoft network)

VNet Peering Properties

Feature	Details
Transitive	No. A↔B and B↔C does NOT mean A↔C. Must explicitly peer A↔C.
Bandwidth	No bandwidth limit — uses Azure backbone
Latency	Very low — direct connection
Address Space Overlap	NOT allowed. VNets must have non-overlapping IP ranges.
Cross-Region	Yes (Global VNet Peering)
Cross-Subscription	Yes (with proper permissions)
Cost	Ingress free, egress charged per GB (small cost)

Real Example: Hub-and-Spoke Topology

```
Hub VNet (10.0.0.0/16):  Shared services (DNS, AD, NVA)
  |
  +-- Spoke 1 (10.1.0.0/16): Production workloads
  |
  +-- Spoke 2 (10.2.0.0/16): Development workloads
```

```
|
+-- Spoke 3 (10.3.0.0/16): Test workloads
```

Peering Configuration:

Hub ↔ Spoke 1 (bidirectional peering)

Hub ↔ Spoke 2 (bidirectional peering)

Hub ↔ Spoke 3 (bidirectional peering)

Important: Spoke 1 cannot directly talk to Spoke 2. Traffic must go through Hub (use Network Virtual Appliance in Hub to route between spokes).

Common Use Cases

Multi-Region Apps — VNet in East US ↔ VNet in West Europe for DR/HA

Hub-Spoke Architecture — Centralized services in hub, isolated workloads in spokes

Cross-Team Collaboration — Development team VNet ↔ QA team VNet

Hybrid Cloud — On-prem connects to hub VNet, which peers to all spoke VNets

⚠ **WARNING:** VNet address spaces cannot overlap! If VNet A is 10.0.0.0/16, VNet B cannot use 10.0.0.0/16 or any overlapping range.

11. Private Endpoints

11.1 What is a Private Endpoint?

Private Endpoint brings Azure PaaS services (Storage, SQL, Key Vault, etc.) into your VNet with a private IP address. Instead of accessing these services over the public internet, you access them via private IP within your VNet.

The Problem It Solves

Normally, Azure Storage is accessed via public endpoint: myaccount.blob.core.windows.net (public IP). This means:

Traffic goes over internet (even if within Azure)

Public endpoint is exposed (even with firewall rules)

Difficult to lock down for highly secure environments

Private Endpoint solves this:

Storage gets a private IP (e.g., 10.1.3.50) in your VNet

Access via private IP, traffic stays within VNet

Can disable public access entirely

How Private Endpoint Works

Create Private Endpoint in your VNet

Choose Azure service (Storage Account, SQL Database, Key Vault, etc.)

Private IP assigned from your VNet subnet

Private DNS zone created automatically (or you manage manually)

DNS resolution: myaccount.blob.core.windows.net → 10.1.3.50 (private IP)

Applications use same DNS name but connect via private IP

Supported Azure Services (Partial List)

Azure Storage (Blob, File, Queue, Table)

Azure SQL Database, Azure Database for PostgreSQL/MySQL

Azure Cosmos DB

Azure Key Vault

Azure Container Registry

Azure App Service, Azure Functions

Azure Event Hubs, Service Bus

Azure Cognitive Services

Real Example: Secure Database Access

```
Before Private Endpoint:

  App in VNet connects to: mydb.database.windows.net

  Resolves to: 40.112.72.205 (public IP)

  Traffic: VNet → Internet → Azure SQL (even though both in
Azure!)

  Security: SQL firewall must allow VNet or specific IPs
```

```
After Private Endpoint:
  Private Endpoint created in VNet subnet 10.1.3.0/24
  Private IP assigned: 10.1.3.75
  Private DNS: mydb.database.windows.net → 10.1.3.75
  App connects to mydb.database.windows.net
  Resolves to 10.1.3.75 (private IP)
  Traffic: App → Private IP (stays in VNet)
  Security: Public access disabled, only accessible from VNet
```

Private Endpoint vs Service Endpoint

Feature	Private Endpoint	Service Endpoint
Traffic Path	Traffic stays within VNet	Traffic goes to public endpoint but via Azure backbone
IP Address	Service gets private IP in your VNet	Service keeps public IP
DNS	Private DNS zone resolves to private IP	Resolves to public IP (with source IP filtering)
Security	Can disable public access entirely	Public endpoint still exists (firewall rules required)
Cost	Charged per endpoint + data processed	Free (but limited to certain services)
Use Case	Maximum security, regulatory requirements	Simple VNet access, cost-sensitive

TIP: For production security: use Private Endpoint. For dev/test or cost savings: Service Endpoint is acceptable.

12. Azure Bastion

12.1 What is Azure Bastion?

Azure Bastion is a fully managed service that provides secure RDP/SSH access to your VMs without exposing them to the public internet. No need for public IPs on VMs, no need for VPN. Access VMs directly from Azure Portal over SSL.

The Problem It Solves

Traditional VM access methods and their problems:

Method	Issues	Recommendation
Public IP on VM + NSG	VM exposed to internet. Constant brute-force SSH/RDP attacks. Security risk.	✗ Don't do this in production
Jump Box / Bastion Host	Dedicated VM with public IP for access. Still exposed, must maintain and patch.	⚠ Better but still risky
VPN Gateway	Secure but requires client configuration, VPN software, certificates.	✓✓ Good but complex
Azure Bastion	Fully managed, no public IPs on VMs, access via browser, no client software.	✓✓ Best for most scenarios

How Azure Bastion Works

Deploy Azure Bastion in your VNet (requires dedicated subnet named 'AzureBastionSubnet')

Bastion gets a public IP (but your VMs don't!)

In Azure Portal, click 'Connect' on a VM → Choose 'Bastion'

Enter VM credentials (username/password or SSH key)

Browser opens HTML5-based RDP or SSH session

Connection: Browser → Azure Bastion (HTTPS 443) → VM (RDP 3389 or SSH 22) over private IP

Azure Bastion Features

No Public IP on VMs — VMs remain fully private

No Client Software — Access from any browser, any device

Seamless Portal Integration — Click 'Connect' in Azure Portal, start session

SSL Protection — All traffic encrypted over HTTPS

Automatic Patching — Microsoft manages Bastion updates

No NSG Rules Needed on Bastion Subnet — Bastion handles required connectivity

Azure Bastion Requirements

Dedicated Subnet — Must be named 'AzureBastionSubnet', minimum /27 (32 IPs)

Standard Public IP — Bastion needs a Standard SKU public IP (but VMs don't!)

VNet Integration — Must be in same VNet as VMs you want to access

Real Example: Secure VM Access Architecture

```
VNet: ProductionVNet (10.1.0.0/16)

Subnets:
  - AzureBastionSubnet: 10.1.0.0/27   (for Bastion service)
  - AppServers:         10.1.1.0/24   (VMs with NO public IPs)
  - DatabaseServers:    10.1.2.0/24   (VMs with NO public IPs)

Azure Bastion:
  Name: ProductionBastion
  Subnet: AzureBastionSubnet
  Public IP: bastion-pip (only Bastion has this, VMs don't)

Access Flow:
  Admin → Azure Portal → Click VM 'Connect' → Select Bastion
  → Browser opens SSH/RDP session
  → Traffic: Browser → Bastion (443) → VM (22/3389 private IP)
```

Bastion SKUs

SKU	Scale	Features	Use Case
Basic	Up to 20 concurrent sessions	RDP/SSH access	Good for small teams
Standard	Up to 50 concurrent sessions	RDP/SSH + file upload/download, shareable link	Production workloads

Cost Considerations

Azure Bastion is charged hourly plus data transfer. It's more expensive than a self-managed jump box VM, but far more secure and zero-maintenance.

Option	Approx. Cost	Trade-off
Azure Bastion	$140/month + data	Fully managed, secure, no maintenance
Jump Box VM	$30-70/month	Must maintain, patch, secure, monitor

> **TIP:** For production environments with compliance requirements (HIPAA, PCI-DSS), Azure Bastion is often mandatory — VMs cannot have public IPs.

⚠ **WARNING:** Common mistake: Creating AzureBastionSubnet that's too small. Use /27 or /26 minimum. /28 will fail deployment.

PART 3
INTERVIEW QUESTIONS & ANSWERS

13. Interview Questions with Detailed Answers

13.1 Networking Fundamentals Questions

[BASIC] Explain the OSI model and which layers are most relevant for DevOps.

Answer: OSI model has 7 layers. For DevOps, the most important are: Layer 7 (Application) — where HTTP/HTTPS issues occur, troubleshooting API errors, status codes. Layer 4 (Transport) — TCP/UDP, port numbers, firewall rules, load balancing. Layer 3 (Network) — IP addressing, routing, VPN issues, VNet peering. When troubleshooting connectivity, I work from Layer 7 down: Is the app responding? Is the port open? Can I ping the IP? Is the cable plugged in?

[BASIC] What is the difference between TCP and UDP? Give real examples of when to use each.

Answer: TCP is connection-oriented and reliable — guarantees delivery and order. Use for: HTTP, databases, SSH, email — anything where data loss is unacceptable. UDP is connectionless and fast — no guarantee of delivery. Use for: DNS queries, video streaming, VoIP, gaming — where speed matters more than perfect reliability. Example: Database connections use TCP because missing one byte corrupts the entire query. Live video uses UDP because retransmitting a lost frame 3 seconds later is useless.

[INTERMEDIATE] Explain CIDR notation. What does 10.0.0.0/16 mean? How many usable IPs does it provide?

Answer: CIDR (Classless Inter-Domain Routing) defines a network range. 10.0.0.0/16 means: First 16 bits are network (10.0), last 16 bits are for hosts. Total IPs = 2^(32-16) = 65,536. Usable IPs = 65,534 (first is network address, last is broadcast). In Azure, also subtract 5 reserved IPs per subnet. This notation is used everywhere in cloud: VNet address spaces, subnet ranges, firewall rules, routing tables.

[INTERMEDIATE] You see HTTP 502 Bad Gateway error. What does it mean and how do you troubleshoot?

Answer: 502 means the gateway/proxy received invalid response from upstream server. Common causes: (1) Application server is down or crashed — check if process is running. (2) Application is starting up but not ready yet — wait and retry. (3) Application responding on wrong port — verify app listens on port gateway expects. (4) Health check failing — check health endpoint returns 200. Troubleshooting steps: Check app logs for errors. Verify app process is running (systemctl status). Test app directly on backend (curl localhost:8080/health). Check gateway/load balancer health probe status.

[ADVANCED] Explain the difference between 401 Unauthorized and 403 Forbidden. Why are they often confused?

Answer: 401 Unauthorized means authentication failed or missing — you haven't proven who you are. User is anonymous or credentials are invalid. Client should send credentials (login). 403 Forbidden means you're authenticated (logged in) but not authorized for this resource — you don't have permission. Even with correct credentials, you're not allowed. They're confused because both prevent access, but 401 is 'who are you?' and 403 is 'I know who you are, but you can't do that'. Example: Accessing /admin without login → 401. Accessing /admin as regular user (not admin) → 403.

[ADVANCED] Design a subnet architecture for a 3-tier web application in Azure. Explain your choices.

Answer: I'd use a /16 VNet with multiple /24 subnets: VNet: 10.0.0.0/16. Subnets: (1) AzureBastionSubnet: 10.0.0.0/27 — for secure VM access, requires /27 minimum. (2) AppGatewaySubnet: 10.0.1.0/24 — Application Gateway, needs /24 for autoscaling. (3) WebTier: 10.0.10.0/24 — frontend VMs behind App Gateway. (4) AppTier: 10.0.20.0/24 — API/business logic servers. (5) DataTier: 10.0.30.0/24 — databases with strictest NSG rules. (6) GatewaySubnet: 10.0.100.0/27 — VPN Gateway for on-prem connectivity. NSG rules: Web tier allows 80/443 from internet via App Gateway. App tier allows app port only from Web tier. Data tier allows DB port only from App tier. Bastion subnet only allows 443 inbound.

13.2 Azure Networking Questions

[BASIC] What is Azure VNet and why do we need it?

Answer: Azure Virtual Network (VNet) is your private network in the cloud. It's logically isolated from other VNets and the internet. You need it to: (1) Deploy VMs, AKS, and other services securely. (2) Control network traffic with subnets and NSGs. (3) Connect to on-premises network via VPN or ExpressRoute. (4) Isolate workloads by environment (prod/dev/test) or team. Without VNet, resources would be on public internet directly, which is insecure. VNet gives you control over IP addressing, DNS,

routing, and security.

[BASIC] What is an NSG and how does it work?

Answer: Network Security Group (NSG) is Azure's firewall. It controls inbound and outbound traffic based on rules. Each rule has: priority (100-4096, lower = higher priority), source/destination (IP or service tag), port, protocol, and allow/deny action. NSG evaluates rules in priority order — first match wins. Can attach NSG to subnet (affects all resources in subnet) or network interface (affects individual VM). It's stateful — if you allow inbound 443, return traffic is automatic. Best practice: apply NSG to subnet for broad control, use NIC-level NSG for exceptions.

[INTERMEDIATE] What's the difference between Azure Load Balancer and Application Gateway?

Answer: Azure Load Balancer is Layer 4 (TCP/UDP) — routes based on IP and port. Doesn't understand HTTP. Fast, simple, low cost. Use for: database load balancing, non-HTTP apps, VM load balancing. Application Gateway is Layer 7 (HTTP/HTTPS only) — routes based on URL path, hostname, headers. Terminates SSL, includes WAF, supports cookie-based affinity. Use for: web apps, microservices, API gateways, anything HTTP. Example: Load Balancer for PostgreSQL cluster (TCP 5432). Application Gateway for routing /api to API servers and /app to frontend.

[INTERMEDIATE] Explain VNet Peering. Is it transitive?

Answer: VNet Peering connects two VNets privately over Microsoft's backbone. Resources communicate using private IPs. Benefits: no gateway needed, low latency, works across regions. Important: Peering is NOT transitive. If VNet A peers with B, and B peers with C, A cannot reach C directly — must explicitly peer A↔C. Address spaces cannot overlap — can't peer 10.0.0.0/16 with 10.0.0.0/16. Use cases: hub-and-spoke topology (hub has shared services, spokes are isolated workloads), multi-region DR, cross-team VNet access.

[ADVANCED] What is a Private Endpoint and when would you use it instead of Service Endpoint?

Answer: Private Endpoint brings Azure PaaS services (Storage, SQL, Key Vault) into your VNet with a private IP. Traffic never leaves VNet. Can disable public access entirely. Use when: maximum security required (compliance like HIPAA, PCI-DSS), want to block all public internet access, regulatory requirement for private connectivity. Service Endpoint is simpler/cheaper — traffic goes to public endpoint but via Azure backbone with source IP filtering. Still has public endpoint. Use when: basic VNet access is enough, cost is concern, security requirements are moderate. Key difference: Private Endpoint = service IN your VNet. Service Endpoint = VNet traffic TO public endpoint.

[ADVANCED] Design a secure network for VMs that require RDP/SSH access without exposing them to internet.

Answer: Best practice: Azure Bastion. Architecture: (1) Create dedicated

AzureBastionSubnet (/27 minimum) in VNet. (2) Deploy Azure Bastion service (gets public IP, but VMs don't). (3) VMs in private subnets with NO public IPs. (4) NSG on VM subnets allows RDP/SSH only from AzureBastionSubnet range. (5) Access: Portal → VM → Connect → Bastion → browser-based RDP/SSH over HTTPS. Benefits: VMs never exposed to internet (no brute-force attacks), no VPN client needed, access from any browser, fully managed (no patching), audit logs in Azure. Alternative for dev/test: self-managed jump box, but Bastion is production-grade.

13.3 Scenario-Based Questions

💼 SCENARIO: Application Can't Connect to Database

Your web app in Azure suddenly can't connect to Azure SQL Database. It worked yesterday. Database is up and responding. How do you troubleshoot?

Expected Approach: Systematic approach: (1) Check app error logs — what's the exact error? Connection timeout? Auth failure? (2) Verify connection string — any recent changes? Is password correct? (3) Check Azure SQL firewall rules — is app's IP/VNet allowed? Check if 'Allow Azure Services' is enabled. (4) If using Private Endpoint — verify Private DNS resolution. Run nslookup from app server — does SQL FQDN resolve to private IP? (5) Check NSG rules — if database in VM, is port 1433 allowed from app subnet? (6) Test connectivity directly — from app VM: 'telnet sqlserver.database.windows.net 1433' or 'psping'. (7) Check SQL logs for blocked connections. (8) Verify service health in Azure Portal — any outages? Most common causes: NSG rule changed, SQL firewall rule removed, Private Endpoint DNS issue, expired password.

💼 SCENARIO: High Latency Between Azure Regions

Your app in East US communicates with database in West Europe. Users complain of slow response times. How do you improve latency?

Expected Approach: Options to reduce cross-region latency: (1) Use Global VNet Peering instead of public internet — traffic on Microsoft backbone, much lower latency. (2) Deploy read replicas closer to users — Azure SQL active geo-replication, CosmosDB multi-region. Users read from nearest region. (3) Implement caching — Azure Cache for Redis in each region to avoid cross-region database calls. (4) Use CDN for static content — Azure Front Door or CDN caches content at edge, reduces backend calls. (5) Consider Azure Traffic Manager or Front Door for intelligent routing — route users to nearest healthy endpoint. (6) Evaluate if cross-region is necessary — can you replicate data asynchronously instead of synchronous calls? (7) If using ExpressRoute, verify BGP routing is optimal. (8) Monitor with Network Watcher — identify actual bottleneck (network vs application). Measure current latency before and after changes — use Connection Monitor.

SCENARIO: Intermittent Connection Timeouts

Users report your website times out intermittently — not always, maybe 1 in 20 requests. How do you diagnose this?

Expected Approach: Intermittent issues are hardest to debug. Approach: (1) Check Application Gateway / Load Balancer metrics — are backends failing health checks? Intermittent failures = likely one backend is unhealthy. (2) Review health probe logs — which backend is timing out? (3) Check that backend's logs during timeout windows — memory spikes? CPU at 100%? (4) Monitor NSG flow logs — are connections being blocked or dropped? (5) Check connection limits — is a backend hitting max connections? (6) Review autoscaling settings — are backends scaling fast enough during traffic spikes? (7) Check for network-level issues — Connection Monitor can detect packet loss. (8) If using App Service — check HTTP queue length metric, may need to scale out. (9) Database timeout? Check SQL connection pool exhaustion. (10) Enable verbose logging temporarily to catch next occurrence. Key: correlate timestamp of user complaint with backend metrics/logs to identify failing component.

SCENARIO: Cost Optimization for Networking

Your Azure bill shows high networking costs. What are common causes and how do you reduce them?

Expected Approach: Common high-cost scenarios: (1) VNet peering data transfer — especially Global VNet Peering across regions. Consider consolidating resources in fewer regions or using Private Endpoint instead of peering. (2) Public IP addresses — charged for Standard Public IPs. Audit unused IPs and delete. Use Azure Bastion instead of public IPs on each VM. (3) Application Gateway idle time — if low traffic, consider switching to Load Balancer (cheaper) or use App Gateway v2 autoscaling to scale to zero. (4) VPN Gateway always-on — if only needed occasionally, use Point-to-Site instead of Site-to-Site. (5) ExpressRoute unused bandwidth — right-size circuit, don't overprovision. (6) Outbound data transfer — largest cost driver. Reduce by: using CDN, caching aggressively, enabling compression, keeping data in same region. (7) NAT Gateway data processing — if very high traffic, verify it's needed. (8) Use Azure Cost Management to identify top networking resources by cost. Typical findings: 40% of networking cost is avoidable (unused IPs, over-provisioned gateways).

PART 4
HANDS-ON PRACTICE & PROJECTS

14. Practice Assignments & Projects

14.1 Networking Fundamentals — Practice

EASY Level

Calculate usable IPs for these CIDR blocks: /24, /26, /28, /30

Identify which of these are public vs private IPs: 192.168.1.1, 8.8.8.8, 10.0.0.1, 172.217.14.206

Look up DNS records for google.com using nslookup — identify A, AAAA, MX, TXT records

Test connectivity to a website on port 443 using telnet or nc

Use traceroute/tracert to see network path to a remote server

MEDIUM Level

Given VNet 10.0.0.0/16, divide it into 4 equal subnets. Calculate range and usable IPs for each.

Design CIDR plan for an application with: 500 web servers, 100 app servers, 10 database servers. What subnet sizes needed?

Create a DNS zone file with A, CNAME, MX records for a fictional company

Document OSI layers involved when you browse https://website.com — what happens at each layer?

Set up a simple HTTP server and test different HTTP methods (GET, POST, PUT, DELETE) — observe status codes

DIFFICULT Level

Design complete IP addressing scheme for multi-region deployment: US (East, West), Europe (North), Asia (Southeast). No overlap, room for growth.

Implement split-horizon DNS: internal clients get private IPs, external clients get

public IPs for same hostname

Troubleshoot a connectivity issue: Given symptoms and network diagram, identify where problem is (Layer 3? Layer 4? Layer 7?)

Calculate optimal MTU for VPN tunnel considering overhead from IPsec, explain impact of fragmentation

Design DNS architecture with high availability: primary/secondary servers, zone transfers, DNSSEC

14.2 Azure Networking — Practice (Use Free Tier)

EASY Level

Create a VNet with address space 10.0.0.0/16 in Azure Portal

Add three subnets: Frontend (10.0.1.0/24), Backend (10.0.2.0/24), Database (10.0.3.0/24)

Create an NSG with rules allowing HTTP (80) and HTTPS (443) from internet

Deploy a simple VM with no public IP, then add a public IP later

Test NSG rules by trying to SSH/RDP with rule disabled vs enabled

MEDIUM Level

Create hub-and-spoke topology: 1 hub VNet, 2 spoke VNets, configure peering

Deploy Internal Load Balancer with 2 backend VMs, configure health probe

Set up Azure Bastion and access a VM without public IP

Create Private Endpoint for Storage Account, verify DNS resolution to private IP

Configure Application Gateway with path-based routing: /api → backend1, /app → backend2

DIFFICULT Level

Implement complete hub-spoke with Network Virtual Appliance (NVA) for spoke-to-spoke traffic

Configure User-Defined Routes (UDR) to force all internet traffic through firewall/NVA

Set up VPN Gateway with Point-to-Site connection, test connectivity from on-prem

Deploy multi-region VNets with Global VNet Peering, implement Traffic Manager for failover

Configure WAF on Application Gateway with OWASP rules, test SQL injection protection

14.3 Mini Project Ideas

Project 1: Secure 3-Tier Application Architecture

Goal — Deploy WordPress with web, app, and database tiers in Azure

Steps:

Create VNet (10.0.0.0/16) with 4 subnets: Bastion, Web, App, DB

Deploy Azure Bastion for secure access

Create NSG for each tier: Web allows 80/443, App allows app port only from Web, DB allows 3306 only from App

Deploy 2 web server VMs in Web subnet behind Load Balancer

Deploy 1 app server VM in App subnet

Deploy Azure Database for MySQL in DB subnet with Private Endpoint

Configure Application Gateway with WAF in front of web tier

Test: access WordPress via App Gateway public IP, verify no direct access to backend VMs

Document network diagram showing traffic flow

Project 2: Multi-Region DR Setup

Goal — Deploy application in two regions with failover

Steps:

Create VNet in East US and West Europe

Configure Global VNet Peering between them

Deploy identical app stack in both regions

Set up Azure Database replication (geo-replication) primary → secondary

Configure Azure Traffic Manager with priority routing (East US primary)

Test failover: shut down East US resources, verify Traffic Manager routes to West Europe

Measure failover time and document RTO/RPO

Project 3: Secure Hybrid Cloud Connectivity

Goal — Connect on-premises network to Azure securely

Steps:

- Create VNet with GatewaySubnet (/27 minimum)
- Deploy VPN Gateway in VNet
- Simulate on-premises: create separate VNet to act as 'on-prem' with VPN Gateway
- Configure VNet-to-VNet VPN connection
- Set up routing: resources in each VNet can reach other via private IPs
- Test connectivity: VM in VNet A pings VM in VNet B using private IP
- Configure NSG to allow specific traffic between 'on-prem' and cloud
- Document: connection diagram, IP addressing, routing tables

Project 4: Network Monitoring & Troubleshooting Lab

Goal — Set up monitoring and practice troubleshooting

Steps:

- Deploy VNet with 3 VMs: web, app, database
- Enable NSG Flow Logs on all NSGs
- Configure Network Watcher: Connection Monitor, packet capture
- Intentionally create issues: block port in NSG, break routing, misconfigure DNS
- Use Network Watcher to diagnose each issue:
  - IP Flow Verify: test if traffic allowed/denied
  - Next Hop: verify routing paths
  - Connection Troubleshoot: check end-to-end connectivity
- Document each issue and how you diagnosed it

PART 5
COMMON MISTAKES & TROUBLESHOOTING

15. Common Networking Mistakes

15.1 IP Addressing & Subnetting Mistakes

Mistake 1: Overlapping VNet Address Spaces

Problem: Trying to peer VNet A (10.0.0.0/16) with VNet B (10.0.0.0/16).

Error: Peering fails because address spaces overlap.

Fix: Use non-overlapping ranges. VNet A: 10.0.0.0/16, VNet B: 10.1.0.0/16, VNet C: 10.2.0.0/16.

> ⚠ **WARNING:** Plan IP addressing from the start! Changing VNet address space after deployment is extremely difficult.

Mistake 2: Subnet Too Small

Problem: Created /28 subnet (16 IPs) for AKS cluster or Application Gateway.

Error: Deployment fails — not enough IPs. Azure reserves 5, AKS needs minimum 30+ for scaling.

Fix: Use /24 (251 usable) as default subnet size. For AKS production: /22 or /20.

> 💡 **TIP:** Always plan for growth. Better to have unused IPs than run out later.

Mistake 3: Forgetting Azure's Reserved IPs

Problem: Created /27 subnet (32 IPs), deployed 27 VMs, then deployment fails.

Cause: Azure reserves first 4 and last 1 IP in every subnet. /27 has 32 - 5 = 27 usable.

Fix: Account for 5 reserved IPs when calculating capacity.

15.2 NSG (Firewall) Mistakes

Mistake 1: Wrong Rule Priority

Problem: Created DenyAll rule with priority 100, then AllowHTTPS at 200. HTTPS blocked.

Cause: Lower priority number = higher priority. Rule 100 evaluated first and denies.

Fix: Allow rules should have lower numbers (100-1000), Deny rules higher (4000+).

Mistake 2: Allowing 0.0.0.0/0 on SSH/RDP

Problem: NSG allows port 22 from 0.0.0.0/0 (entire internet).

Security Risk: VM will be scanned and attacked within minutes. Brute-force attempts constantly.

Fix: Use Azure Bastion. OR allow SSH only from specific IPs (corporate VPN range). NEVER allow 0.0.0.0/0 on 22, 3389, 3306.

Mistake 3: Forgetting Inbound vs Outbound

Problem: Created inbound rule allowing 443, but app can't make HTTPS calls to internet.

Cause: Outbound rules are separate. Default allows outbound, but custom NSG might have denied it.

Fix: Check both inbound AND outbound rules. Applications need outbound for API calls, updates, etc.

15.3 Load Balancer / Application Gateway Mistakes

Mistake 1: Using Layer 4 LB for HTTP Routing

Problem: Trying to route /api and /app to different backends using Load Balancer.

Cause: Load Balancer is Layer 4 — doesn't understand HTTP URLs.

Fix: Use Application Gateway for HTTP routing, Load Balancer only for simple port-based.

Mistake 2: Health Probe Misconfiguration

Problem: Application Gateway marks all backends as unhealthy, no traffic flows.

Cause: Health probe endpoint wrong (/health vs /healthz), or app doesn't respond with 200.

Fix: Verify health probe path exists and returns 200 OK. Test with curl from VM.

Mistake 3: Backend Pool Empty

Problem: Created Application Gateway but no traffic reaches backends.

Cause: Forgot to add VMs/VMSS to backend pool.

Fix: Always verify backend pool has targets and they're healthy in monitoring tab.

15.4 DNS Mistakes

Mistake 1: Forgetting to Update DNS After IP Change

Problem: Changed VM IP or deployed new load balancer, but users still get old IP.

Cause: DNS cached at client, recursive resolver, and browser.

Fix: Wait for TTL to expire OR proactively lower TTL to 300 (5 min) before change, make change, then raise TTL back.

Mistake 2: Incorrect DNS for Private Endpoint

Problem: Created Private Endpoint but app still connects to public IP.

Cause: Private DNS zone not linked to VNet, or DNS record missing.

Fix: Verify: nslookup myaccount.blob.core.windows.net resolves to private IP (10.x), not public. Link Private DNS zone to VNet.

Mistake 3: Split-Brain DNS Issues

Problem: Internal users can't resolve external domain, or external users get internal IPs.

Cause: Misconfigured split-horizon DNS — same domain in public and private zones.

Fix: Use different subdomains: internal.company.com (private), www.company.com (public).

15.5 Troubleshooting Flowchart

Network Connectivity Issue — Systematic Approach

Can you ping the IP? → NO? → Layer 3 routing problem

Check routing tables (UDR)

Check VNet peering status

Check VPN/ExpressRoute connection

Can you ping the IP? → YES → Is the port open?

Test with: telnet server-ip 443 or nc -zv server-ip 443

NO? → NSG blocking or service not listening

YES? → Layer 7 application issue

Check NSG Rules

Is port allowed in NSG?

Check both subnet NSG and NIC NSG

Verify priority order — deny rule before allow?

Check Service

Is service running? (systemctl status or equivalent)

Is service listening on correct port? (netstat -tuln)

Check service logs for errors

Check Load Balancer / Application Gateway

Are backends healthy?

Is health probe configured correctly?

Check backend pool has members

DNS Issues?

Does nslookup resolve to correct IP?

Is Private DNS zone linked?

Check /etc/hosts for overrides

PART 6
QUICK REVISION GUIDE

16. Quick Reference Cheat Sheets

16.1 OSI Model Quick Reference

Layer	Protocols	Focus
7 - Application	HTTP, DNS, SSH	What user interacts with
6 - Presentation	SSL/TLS	Encryption, formatting
5 - Session	Session mgmt	Connection establishment
4 - Transport	TCP, UDP	Ports, reliability
3 - Network	IP, routing	IP addressing, routing
2 - Data Link	Ethernet, MAC	Local network, switches
1 - Physical	Cables	Physical connection

16.2 Port Numbers — Memorize These

Port	Service	Port	Service
22	SSH	443	HTTPS
80	HTTP	3306	MySQL
53	DNS	5432	PostgreSQL
3389	RDP	6379	Redis
25	SMTP	27017	MongoDB

16.3 HTTP Status Codes Quick Ref

Code	Meaning
200 OK	Success
201 Created	Resource created
301 Moved Permanently	Permanent redirect
400 Bad Request	Invalid request
401 Unauthorized	Not authenticated
403 Forbidden	Not authorized
404 Not Found	Resource doesn't exist
500 Internal Server Error	Server crashed
502 Bad Gateway	Upstream server down
503 Service Unavailable	Server overloaded
504 Gateway Timeout	Upstream server timeout

16.4 CIDR Quick Reference

CIDR	Total	CIDR	Total
/32	1 IP	/24	256 IPs (254 usable)
/31	2 IPs	/23	512 IPs
/30	4 IPs	/22	1,024 IPs
/29	8 IPs	/21	2,048 IPs
/28	16 IPs	/20	4,096 IPs
/27	32 IPs	/16	65,536 IPs
/26	64 IPs	/8	16,777,216 IPs

16.5 Azure Networking Quick Decisions

VNet Sizing

VNet: Use /16 (65k IPs) as standard

Subnet: Use /24 (254 IPs) as default

AKS: Use /22 or /20 for production

Bastion: Use /27 minimum

Load Balancing Decision Tree

HTTP traffic? → Use Application Gateway

Non-HTTP traffic (DB, RDP, SSH)? → Use Load Balancer

Need URL routing or WAF? → Use Application Gateway

Simple TCP load balancing? → Use Load Balancer

Secure VM Access

Best: Azure Bastion (no public IPs, browser-based)

Alternative: VPN Gateway (requires client setup)

Never: Public IP on every VM with NSG (security risk)

16.6 Pre-Interview Checklist

Review these the night before interview:

- [ ] Explain OSI model layers 3, 4, 7 and their relevance to DevOps
- [ ] Describe TCP vs UDP — when to use each, give examples

Calculate usable IPs from CIDR: /24, /27, /16

Explain difference between 401, 403, 502, 503, 504

Memorize ports: 22, 80, 443, 3306, 5432, 6379, 3389

Describe Azure VNet, subnets, NSG, how they relate

Explain Load Balancer vs Application Gateway

Describe VNet Peering — is it transitive?

What is Private Endpoint and when to use it?

How does Azure Bastion work? Why use it?

END OF NETWORKING STUDY GUIDE

Master the fundamentals. Understand Azure services. Practice hands-on. Success will follow!