**1️⃣ Verify VPC Peering Connection Status**

**AWS Console Path**:  
AWS Console → VPC → Peering Connections → Select Peering → Check Status

**What to look for**:

* **Active** ✅ — Good to go.
* **Pending Acceptance** ⚠ — Must accept from other VPC.
* **Failed** ❌ — Delete & recreate.

**Real Use Case**:  
In one project, we created a peering from **App VPC** in Account A to **DB VPC** in Account B. We forgot to accept in Account B. The peering stayed in *Pending Acceptance*, so all packets dropped.

**Interview Tip**: Emphasize that this is the **first check** before even touching routes or security.

**2️⃣ Check Route Tables**

**AWS Console Path**:  
AWS Console → VPC → Route Tables → Select Table for Subnet → Routes tab

**What to look for**:

* In **VPC-A**: Destination = CIDR of VPC-B, Target = pcx-XXXXXX
* In **VPC-B**: Destination = CIDR of VPC-A, Target = pcx-XXXXXX

**Real Use Case**:  
We had VPC-A route to VPC-B, but VPC-B had no return route. TCP handshake failed because SYN-ACK never made it back. Adding the reverse route fixed it instantly.

**Interview Tip**: Mention “routing must be **bi-directional** for peering to work”.

**3️⃣ Check Security Groups**

**AWS Console Path**:  
AWS Console → EC2 → Instances → Select Instance → Security tab → Security groups → Inbound & Outbound rules

**What to look for**:

* DB SG inbound: Allow TCP 3306 from CIDR of VPC-A
* App SG outbound: Allow TCP 3306 to CIDR of VPC-B

**Real Use Case**:  
A MySQL connection failed even though routes were correct — inbound SG allowed 3306 but outbound SG on DB instance blocked responses. Fixed by adding an outbound rule for ephemeral ports.

**Interview Tip**: Say “SGs are **stateful**, so return traffic is allowed automatically — unless outbound is restricted”.

**4️⃣ Check Network ACLs**

**AWS Console Path**:  
AWS Console → VPC → Network ACLs → Select NACL → Inbound & Outbound Rules

**What to look for**:

* Inbound: Allow 3306 & ephemeral ports 1024–65535 from peer VPC CIDR.
* Outbound: Same rules reversed.

**Real Use Case**:  
NACL blocked DB responses on ephemeral ports — ping worked but MySQL connection failed. Once we added ephemeral ports, the connection succeeded.

**Interview Tip**: Stress that “NACLs are **stateless**, both inbound and outbound rules must be explicitly set”.

**5️⃣ Check for Overlapping CIDR Ranges**

**AWS Console Path**:  
AWS Console → VPC → Your VPCs → Check IPv4 CIDR

**What to look for**:

* VPC-A CIDR ≠ VPC-B CIDR
* No overlap (e.g., both can’t be 10.0.0.0/16)

**Real Use Case**:  
Two VPCs had 10.0.0.0/16 → peering created fine, but AWS ignored routes because of ambiguity. Changed one VPC to 10.1.0.0/16 and routes started working.

**Interview Tip**: Say “Peering works only when IP ranges are unique — AWS won’t route overlapping networks”.

**6️⃣ Enable DNS Resolution**

**AWS Console Path**:  
AWS Console → VPC → Your VPCs → Select VPC → Actions → Edit DNS Resolution / DNS Hostnames

**What to look for**:

* DNS resolution: **Enabled**
* DNS hostnames: **Enabled** (if using private DNS)

**Real Use Case**:  
Application tried db.internal but got **NXDOMAIN**. IP worked fine — confirmed DNS over peering wasn’t enabled. Enabling it fixed hostname resolution.

**Interview Tip**: Use the phrase “Without DNS over peering, you must use IPs — which is brittle”.

**7️⃣ Test Connectivity (Step-by-Step)**

**AWS Console Path**:  
AWS Console → EC2 → Connect → Session Manager / SSH

**Tests**:

# Layer 3: Network

ping 172.31.2.20

# Layer 4: TCP

telnet 172.31.2.20 3306

# Layer 7: Application

mysql -h 172.31.2.20 -u root -p

**Real Use Case**:  
Ping worked (routing ok), telnet failed → SG blocked MySQL port. Adding inbound SG rule fixed it.

**Interview Tip**: Mention “test progressively from network to app layer to isolate the failure”.

**8️⃣ Cross-Region Peering Limits**

**AWS Console Path**:  
AWS Console → VPC → Peering Connections → Select Peering → Details tab

**What to look for**:

* CIDR-based SG rules (no SG references across regions)
* Awareness of data transfer costs

**Real Use Case**:  
App in ap-south-1 → DB in us-east-1. SG-ID reference failed because cross-region peering requires CIDR rules. Changed SG inbound to use CIDR — worked instantly.

**Interview Tip**: Say “Cross-region peering behaves differently — can’t reference SG IDs”.

✅ **If you explain like this in interviews — with AWS console paths + what to check + real example — you’ll sound hands-on and experienced.**

If you want, I can **create a visual AWS console screenshot-based flow** showing exactly where to click for each of these steps. That would make your interview answer look like you’ve done this in production.

Do you want me to prepare that visual?

**Final Troubleshooting Checklist**

1. ✅ Peering status = Active
2. ✅ Routes in both VPCs point to the peer
3. ✅ SGs allow correct CIDRs & ports
4. ✅ NACLs allow both inbound/outbound
5. ✅ No overlapping CIDRs
6. ✅ DNS resolution enabled if using hostnames
7. ✅ Connectivity tested at multiple layers
8. ✅ Aware of region-specific limits

