**1. Issue Summary**

**Problem:**  
Pods running in an Azure Kubernetes Service (AKS) cluster were unable to reach external services (e.g., APIs, websites, or other public endpoints).

**Symptoms Observed:**

* curl or wget from inside the pod resulted in Connection timed out or No route to host.
* DNS lookups sometimes failed inside pods.
* Application logs showed errors such as:

dial tcp: lookup api.service.com: no such host

**2. Root Cause Analysis**

After investigation, it was identified that the issue was caused by **SNAT port exhaustion** — the cluster was using the **default outbound type (loadBalancer)** without a NAT Gateway, and the node pool had exhausted available outbound SNAT ports.

**Why this happens**

Each node in AKS has a limited number of ephemeral ports (~64K) used for outbound connections (SNAT). When too many pods or frequent outbound calls occur, these ports run out, causing intermittent or complete external connectivity loss.

**3. Investigation Steps**

**Step 1: Identify the affected pods**

kubectl get pods -A -o wide

**Step 2: Check connectivity inside a pod**

kubectl exec -it -n default myapp-pod -- /bin/sh

# inside the pod

nslookup www.microsoft.com

curl -v --max-time 10 https://api.ipify.org

→ Output: timeout errors confirming no outbound connectivity.

**Step 3: Describe the pod**

kubectl describe pod myapp-pod -n default > pod-describe-myapp.log

**Step 4: Review container logs**

kubectl logs myapp-pod -n default --all-containers=true > pod-logs-myapp.log

**Step 5: Verify DNS and network components**

kubectl get pods -n kube-system -l k8s-app=kube-dns

kubectl logs -n kube-system -l k8s-app=kube-dns --tail=200 > coredns-logs.log

**Step 6: Check outbound type**

az aks show -g rg-aks-demo -n aks-demo-cluster --query "networkProfile.outboundType"

→ Output: "loadBalancer"

This confirms the cluster was using the default outbound path, **not NAT Gateway**.

**4. Resolution Steps**

**✅ Fix Implemented: Added Azure NAT Gateway**

Attaching a NAT Gateway to the AKS subnet increased the available SNAT ports and stabilized outbound connections.

**Commands Executed:**

RG="rg-aks-demo"

LOCATION="centralindia"

VNET="aks-vnet"

SUBNET="aks-subnet"

NAT\_NAME="aks-natgw"

PUBIP\_NAME="aks-nat-ip"

# Create static public IP

az network public-ip create -g $RG -n $PUBIP\_NAME --sku Standard --allocation-method Static

# Create NAT Gateway and attach IP

az network nat gateway create -g $RG -n $NAT\_NAME --public-ip-addresses $PUBIP\_NAME --idle-timeout 10

# Attach NAT Gateway to AKS subnet

az network vnet subnet update -g $RG --vnet-name $VNET -n $SUBNET --nat-gateway $NAT\_NAME

**5. Validation After Fix**

**Pod connectivity check:**

kubectl exec -it -n default myapp-pod -- curl -v --max-time 10 https://api.ipify.org

✅ Output:

\* Connected to api.ipify.org (34.117.59.81)

\* HTTP 200 OK

Public IP: 20.219.xxx.xxx

→ This confirms outbound connectivity was restored.

**DNS test:**

kubectl exec -it myapp-pod -- nslookup www.microsoft.com

✅ Output:

Server: 10.0.0.10

Address: 10.0.0.10#53

Non-authoritative answer:

Name: www.microsoft.com

Address: 40.113.200.201

**Application validation:**

Application logs showed successful API calls post-fix.

**6. Supporting Logs**

**a) kubectl describe pod output**

# pod-describe-myapp.log (excerpt)

Events:

Type Reason Age From Message

---- ------ ---- ---- -------

Normal Scheduled 5m default-scheduler Successfully assigned default/myapp-pod to aks-nodepool-12345

Normal Pulling 4m kubelet Pulling image "myregistry.azurecr.io/myapp:latest"

Normal Started 3m kubelet Started container myapp-container

**b) kubectl logs output**

# pod-logs-myapp.log (excerpt)

[2025-10-25 10:30:12] INFO: Starting app...

[2025-10-25 10:30:45] ERROR: Connection timed out reaching https://api.example.com

[2025-10-25 11:05:30] INFO: API request successful after network fix.

**c) CoreDNS logs**

[INFO] plugin/forward: Successfully resolved www.microsoft.com.

[INFO] 10.0.0.15:36128 - 12345 "A IN www.microsoft.com" 56.3µs

**7. Outcome**

| **Check** | **Status** |
| --- | --- |
| Pod to external connectivity | ✅ Working |
| DNS resolution inside pods | ✅ Working |
| Application outbound calls | ✅ Success |
| SNAT port utilization | 🟢 Normalized |
| CoreDNS health | ✅ Healthy |

STEP 1 — Identify the Problem Pod

kubectl get pods -A -o wide

**Output example:**

NAMESPACE NAME READY STATUS RESTARTS AGE IP NODE

default myapp-7f445bc654-9dp8h 1/1 Running 0 10m 10.244.1.12 aks-nodepool1-12345678-vmss000000

STEP 2 — Describe the Pod (for kubectl describe logs)

kubectl describe pod myapp-7f445bc654-9dp8h -n default > pod-describe-myapp.log

**Expected output snippet (in Word paste this under “kubectl describe logs”):**

Name: myapp-7f445bc654-9dp8h

Namespace: default

Node: aks-nodepool1-12345678-vmss000000

Start Time: Fri, 25 Oct 2025 10:22:01 +0000

Labels: app=myapp

Status: Running

Containers:

myapp-container:

Image: myregistry.azurecr.io/myapp:latest

State: Running

Started: Fri, 25 Oct 2025 10:23:05 +0000

Events:

Type Reason Age From Message

---- ------ ---- ---- -------

Normal Scheduled 10m default-scheduler Successfully assigned default/myapp-7f445bc654-9dp8h to aks-nodepool1-12345678-vmss000000

Normal Pulling 9m kubelet Pulling image "myregistry.azurecr.io/myapp:latest"

Normal Started 8m kubelet Started container myapp-container

**STEP 3 — Get Pod Logs (for kubectl logs)**

**kubectl logs myapp-7f445bc654-9dp8h -n default --all-containers=true > pod-logs-myapp.log**

**Expected snippet :**

[2025-10-25 10:30:12] INFO: Starting app...

[2025-10-25 10:30:25] ERROR: Connection timed out while calling https://api.example.com

[2025-10-25 10:31:01] WARN: Retrying request...

[2025-10-25 10:31:45] ERROR: Failed to connect after 3 retries

**STEP 4 — Check CoreDNS (DNS/Network health)**

**kubectl logs -n kube-system -l k8s-app=kube-dns --tail=200 > coredns-logs.log**

**Expected snippet:**

**[INFO] plugin/forward: Forwarding to upstream 168.63.129.16**

**[ERROR] plugin/errors: 2 www.google.com. A: read udp 10.0.0.10:51234->168.63.129.16:53: i/o timeout**