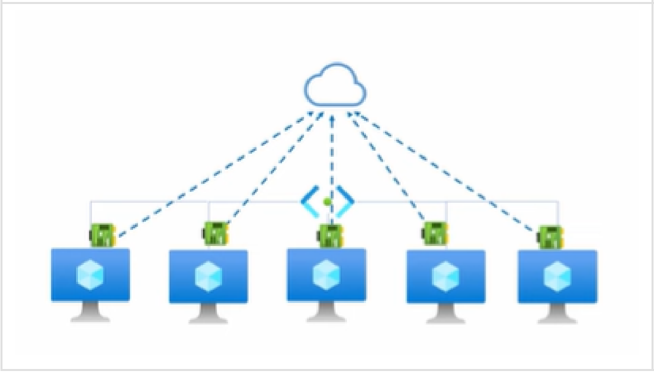
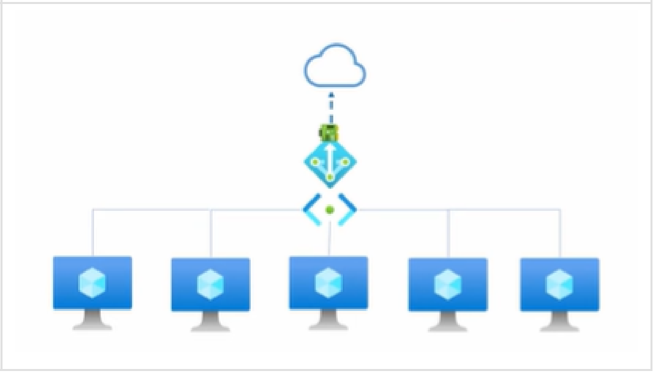
Introduction to NAT

Index

Traditional	NAT
	

- #Static_NAT — permanent one-to-one mapping usually between a public and private IP address. Used for servers which must accept incoming connections.
- #Dynamic_NAT — uses a pool of public addresses which are given out on an as needed first come first served basis. Usually used for internal hosts which need to connect to the Internet but do not accept incoming connections.
- #PAT (Port Address Translation) — allows the same public IP address to be reused.

Network Address Translation is a method of remapping one IP space into another during transit across a routing device, incidentally this is what saved the
world from running out of IP addresses.

We can take an entire company worth of Ip addresses and translate them into one public IP address that goes out one the internet.

Things to Remember

- NAT Gateway can be associated with multiple subnets
- NAT Gateway in a VNet can be associated with multiple subnets
- A subnet cannot be associated with multiple NAT Gateways
- Basic resources not supported for a NAT Gateway subnet
- A NAT gateway can’t be used with IPv6 public IP addresses or prefixes.
- A NAT gateway can’t be used with basic SKU public IP addresses.
- NAT gateway doesn't support ICMP.
- IP fragmentation isn't available for NAT Gateway.

- deploy two VM’s. VM1 with Public IP and another VM2 without a Public IP.
- Open up the VM1 and check your IP, as usual you can get your VM’s public IP.
- Now Open VM2 from VM1 and do the same thing, check your IP. You can still see a Public IP even though you didn’t assigned one.
- This comes from Azure’s #Secure_Network_Address_Translation
- source - https://youtu.be/yghrkFzaYTU?si=vfU4GyO7TTqN5cAc&t=156.
- You need a way to get out to Internet. Azure gives you Network address Translated ip address.

Note

#NAT is associated to the subnet and it replaces azure's default secure Network address translation.

- ! NAT Gateway is an outbound service. So NAT is associated to a single source subnet and destination side is not associated with NAT Gateway.

SNAT Ports

- Each public IP address associated with a NAT gateway provides **64,512** #SNAT-ports
- A single NAT gateway can scale up to 16 public IP address, offering a total of over **1 million** SNAT ports.

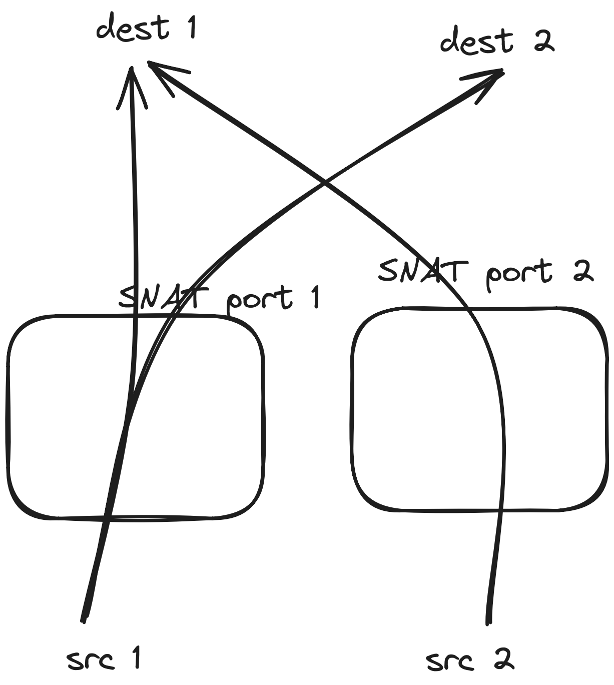
Allocation:

- SNAT ports are **dynamically allocated on-demand** to any virtual machine in the subnet associated with the NAT gateway. This means machines don't need pre-assigned port pools, reducing the risk of exhaustion.
- Ports are selected at random from the available inventory.
- A single port can be used for multiple connections to different destinations, as long as the destination ports are different.

Exhaustion and scaling:

- SNAT #port-exhaustion can occur if there are too many outbound connections for the available ports. However, with the large pool offered by NAT gateway, this is less likely compared to other Azure services like Azure Firewall.
- If needed, you can scale the NAT gateway by adding more public IP addresses, further increasing the available SNAT ports.

- When multiple subnets are attached to the same NAT Gateway, the SNAT port inventory provided by NAT gw is shared across all subnets.
- SNAT ports serve as unique identifiers to distinguish different connection flows from one another.
- ! The **same SNAT port** can be used to connect to **different destination endpoints** at the same time.
- ! **Different SNAT ports** are used to make connections to the **same destination endpoint** in order to distinguish different connection flows from one another.
- SNAT ports being reused to connect to the same destination are placed on a reuse cool down timer before they can be reused.



Availability zones

- A NAT gateway can be created in a specific availability zone or placed in **no zone**. When a NAT gateway is placed in no zone, Azure selects a zone for the NAT gateway to reside in.

1. **Specific Availability Zone:**
   - If you choose a specific availability zone (e.g., Zone 1, Zone 2, or Zone 3), the NAT gateway is explicitly placed in that chosen zone.
   - Placing the NAT gateway in a specific zone ensures high availability within that zone, but it means that if that zone becomes unavailable, the NAT gateway won't be accessible.
2. **No Zone (Regional):**
   - If you choose to place the NAT gateway in no zone, Azure automatically selects an availability zone for the NAT gateway within the region.
   - Placing the NAT gateway in no zone is also known as a regional deployment, and it provides greater resilience against the failure of a single availability zone.
3. **Implications of No Zone Placement:**
   - When a NAT gateway is placed in no zone, Azure's platform makes the decision on which specific availability zone the NAT gateway resides in.
   - This provides automatic redundancy and ensures that if one availability zone experiences an issue, traffic can be routed through the NAT gateway in another zone, maintaining connectivity.
4. **High Availability Considerations:**
   - Placing the NAT gateway in no zone is a good choice when you prioritize high availability and want Azure to handle the selection of the specific zone for redundancy.
   - It is particularly beneficial in scenarios where your workloads need to maintain connectivity even if one availability zone faces disruptions.
5. **Example Scenario:**
   - Suppose you have a multi-tier application deployed across different availability zones for resilience. Placing the NAT gateway in no zone ensures that, in case of a failure in one zone, the NAT gateway seamlessly operates in another zone, preserving connectivity for the application.

In summary, placing a NAT gateway in no zone allows Azure to automatically select an availability zone within the region for redundancy, enhancing high availability for your networking infrastructure. It is a suitable choice when you want Azure to manage the placement of the NAT gateway for optimal resilience across availability zones.

Example SNAT flows for NAT gateway

Many to one SNAT with NAT gateway

virtual machines (10.0.0.1 and 10.2.0.1)
destination IP 23.53.254.142.
NAT gateway public IP address 65.52.1.1

Flow	Source tuple	Source tuple after SNAT	Destination tuple
1	10.0.0.1:4283	65.52.1.1:1234	23.53.254.142:80
2	10.0.0.1:4284	65.52.1.1:1235	23.53.254.142:80
3	10.2.0.1:5768	65.52.1.1:1236	23.53.254.142:80

NAT gateway reuses a SNAT port to connect to a new destination

Flow	Source tuple	Source tuple after SNAT	Destination tuple
4	10.0.0.1:4285	65.52.1.1:1234	26.108.254.155:80

NAT gateway SNAT port cool down for reuse to the same destination

Flow	Source tuple	Source tuple after SNAT	Destination tuple	Packet type connection is closed with	Destination firewall timer for cool down for source port
4	10.0.0.1:4285	65.52.1.1:1234	26.108.254.155:80	TCP FIN	20 seconds

- In a scenario where NAT gateway reuses a SNAT port to make new connections to the same destination endpoint.
- the SNAT port is first placed in a SNAT port reuse phase for **cool down**.
- The SNAT port reuse period helps ensure that SNAT ports aren't reused too quickly when connecting to the same destination.
- beneficial in scenarios where the destination endpoint has a firewall with its own source port timer for cool down in place.
- Flow 4 was connecting to a destination endpoint fronted by a firewall with a 20-second source port cool down timer.
- Connection flow 4 closes with a TCP FIN packet.
- NAT gateway places SNAT port 1234 in cool down for 65 seconds before it can be reused.
- Because port 1234 is in cool down for longer than the firewall source port cool down timer duration of 20 seconds
- connection flow 5 proceeds with reusing SNAT port 1234 without issue.
- source

Flow	Source tuple	Source tuple after SNAT	Destination tuple
5	10.2.0.1:5769	65.52.1.1:1234	26.108.254.155:80

- **!** NAT gateway places SNAT ports under different SNAT port reuse cool down timers depending on how the previous connection closed.

Timer	Description	Value
TCP FIN	After a connection closes by a TCP FIN packet, a 65-second timer is activated that holds down the SNAT port. The SNAT port is available for reuse after the timer ends.	65 seconds
TCP RST	After a connection closes by a TCP RST packet (reset), a 16-second timer is activated that holds down the SNAT port. When the timer ends, the port is available for reuse.	16 seconds
TCP half open	During connection establishment where one connection endpoint is waiting for acknowledgment from the other endpoint, a 30-second timer is activated. If no traffic is detected, the connection closes. Once the connection has closed, the source port is available for reuse to the same destination endpoint.	30 seconds

- For UDP traffic, after a connection closes, the port is in hold down for 65 seconds before it's available for reuse.

TCP reset

In the context of a NAT gateway, a #TCP-reset (RST) packet is a specific response sent by the gateway to abruptly terminate a TCP connection. This can happen for several reasons, and understanding them is crucial for troubleshooting connectivity issues in your network.

Here's a breakdown of TCP resets in NAT gateways:

Causes of TCP resets:

- **Idle timeout:** NAT gateways generally have an idle timeout period. If no data is exchanged on a connection for a set amount of time (typically around 350 seconds), the gateway considers it inactive and drops it. It then sends a reset packet to both endpoints, informing them that the connection is no longer valid.
- **No matching connection flow:** When the NAT gateway receives incoming traffic on a connection that it doesn't recognize (doesn't exist in its state table), it sends a reset packet to inform the sender that the connection doesn't exist. This can happen if the connection timed out and was dropped, or if the original request originated from a different IP address than the expected one.
- **Fragmented TCP packets:** Occasionally, NAT gateways might not handle fragmented TCP packets efficiently, leading to connection termination and a reset packet being sent. This is less common with modern gateways but can still be a factor.
- **Configuration issues:** Misconfigured settings on the NAT gateway or the connected instances can also trigger reset packets due to unexpected behaviour.

Consequences of TCP resets:

- **Connection disruptions:** When a connection receives a reset packet, it's abruptly terminated, potentially disrupting ongoing communication. This can lead to application errors, data loss, and user frustration.
- **Increased network traffic:** Reset packets add additional overhead to the network, particularly if connections drop and re-establish frequently.
- **Troubleshooting challenges:** Identifying the cause of reset packets can be complex, requiring investigation of connection logs, timeouts, and gateway configuration.

Tips for minimizing TCP resets:

- **Configure reasonable idle timeouts:** Adjust the #idle_timeout based on your application needs and expected communication patterns. Longer timeouts reduce resets but introduce potential resource inefficiencies.
- **Use TCP keepalives:** Enable TCP keepalives on your instances to send periodic packets, keeping the connection alive and preventing idle timeouts.
- **Avoid fragmented packets:** Ensure your applications and network settings adhere to best practices for TCP packet size and fragmentation to minimize compatibility issues with the NAT gateway.
- **Monitor and investigate:** Review connection logs and network metrics to identify frequent reset occurrences and diagnose potential causes.

By understanding the role and causes of TCP resets in NAT gateways, you can gain better control over your network connectivity and mitigate their disruptive effects.

TCP idle timeout

- for TCP protocols - 4 minutes to 120 minutes

- for UDP protocols - 4 minutes

- NAT gateway holds the SNAT port until the connection idle times out.
- long idle timeouts will increase the likelihood of SNAT port exhaustion
- Recommended to increase the TCP idle timeout duration to longer than the default time of 4 minutes.

TCP keepalives

- #TCP-keepalives can be used to provide a pattern of refreshing long idle connections and endpoint liveness detection
- source

UDP keepalive

- #UDP-keepalives should be used to ensure that the idle timeout value isn't reached, and that the connection is maintained.
- Unlike TCP connections, a UDP keepalive enabled on one side of the connection only applies to traffic flow in **one direction**.
- UDP keepalives must be **enabled on both sides** of the traffic flow in order to keep the traffic flow alive.

Idle Timeout Timers

#idle_timeout/Timer	Description	Value
#idle_timeout/TCP-idle-timeout	TCP connections can go idle when no data is transmitted between either endpoint for a prolonged period of time. A timer can be configured from 4 minutes (default) to 120 minutes (2 hours) to time out a connection that has gone idle. Traffic on the flow resets the idle timeout timer.	Configurable; 4 minutes (default) - 120 minutes
#idle_timeout/UDP-idle-timeout	UDP connections can go idle when no data is transmitted between either endpoint for a prolonged period of time. UDP idle timeout timers are 4 minutes and are **not configurable**. Traffic on the flow resets the idle timeout timer.	**Not configurable**; 4 minutes

Bandwidth

- Each NAT gateway provides 50 Gbps of throughput (outbound and inbound).

Performance

- support up to 50,000 concurrent connections per IP address to the same destination endpoint.
- Can process 1M packets per second and up to 5M packets per second.