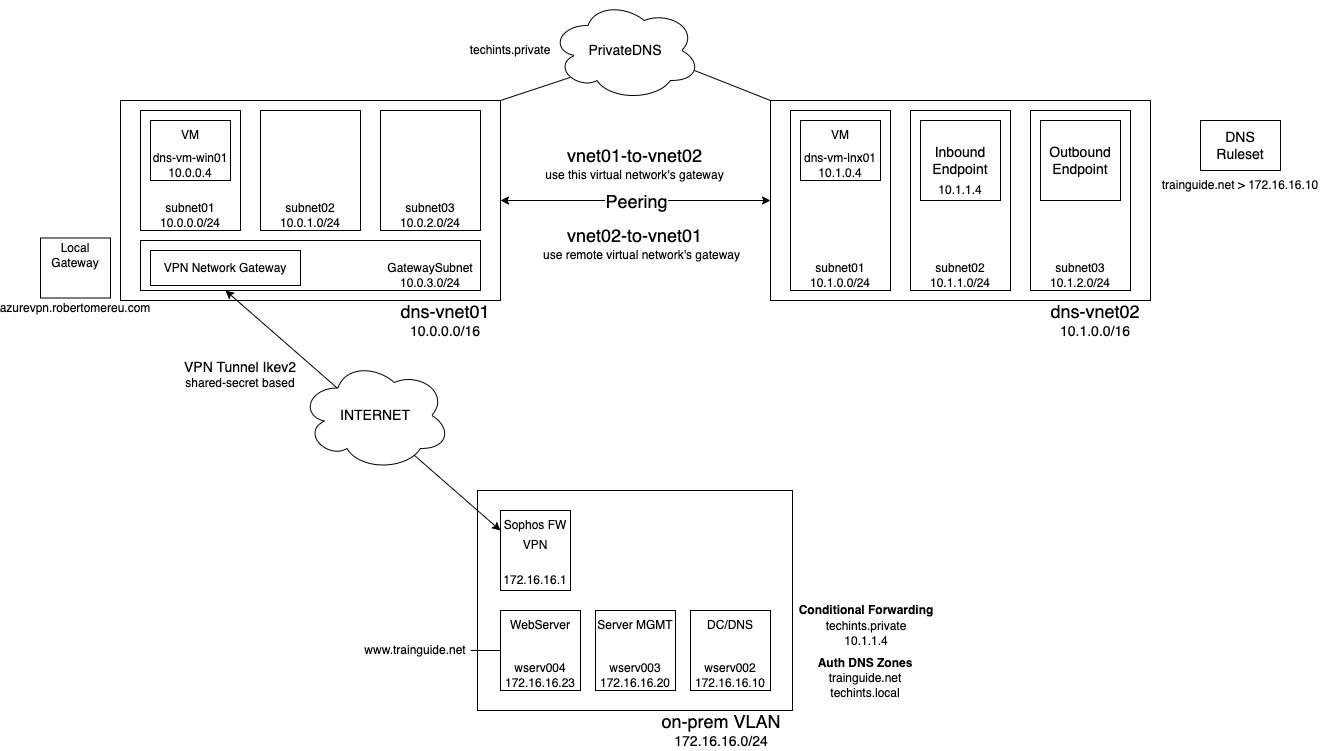
# AZ-104T00A – Administer Network Traffic

Good day everyone. We are going to discuss Network traffic with Network Routing and Endpoints, Load Balancer, Application Gateway and Network Watcher. Before proceeding, I’d like to complete the demo regarding VPN and DNS since we haven’t showed till the end last time:

## Configure Network Routing and Endpoints

Administrators use network routes to control the flow of traffic through a network. Azure virtual networking provides capabilities to help you customize your network routes, establish service endpoints, and access private links.

Review System Routes

Azure uses system routes to direct network traffic between virtual machines, on-premises networks, and the internet. Information about the system routes is recorded in a route table. Azure VMs that are in the same VNet can communicate automatically with each other and with the Internet without any explicit configuration changes, even when they are in different subnets. This is also the case for communication from the VMs to your on-premises network when a hybrid connection from Azure to your datacenter has been established. This ease of setup is made possible by what is known as system routes, which define how IP traffic flows in Azure VNets. The following are the default system routes that Azure will use and provide for you: Within the same subnet, from one subnet to another within a VNet, VMs to the Internet, A VNet to another VNet through a VPN gateway, a VNet to another VNet through VNet peering, a VNet to your on-premises network through a VPN gateway or ExpressRoute, or Virtual Network Service Endpoint.

Identify User-Defined Routes

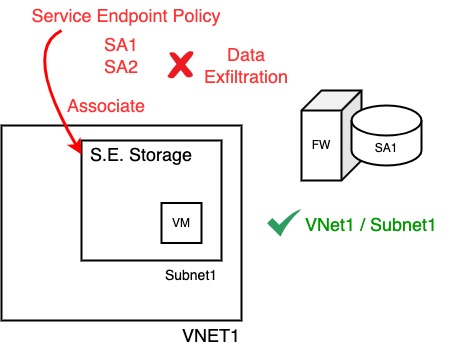
There are some use cases where you will want to configure the routing of packets differently from what is provided by the default system routes. One of these scenarios is when you want to send traffic through a network virtual appliance, such as a third-party Load Balancer, firewall, or router deployed into your VNet from the Azure Marketplace. To make this possible, you must create what are known as user-defined routes (UDRs). The UDR is implemented by creating a route table resource. Within the route table, a number of routes are configured. Each route specifies the destination IP range (in CIDR notation) and the next hop IP address. A variety of different types of next hop are supported: **Virtual Appliance.** A virtual machine running a network application such as a load-balancer or firewall. With this next hop type, you also specify the IP address of the appliance, which can be a virtual machine or internal load-balancer for high-availability virtual appliances. **Virtual Network Gateway.** Used to route traffic to a VPN Gateway (but not an ExpressRoute Gateway, which uses BGP for custom routes). Because there can be only one VPN Gateway associated with a VNet, you are not prompted to specify the actual gateway resource. **Virtual Network.** Used to route traffic within the Virtual Network. **Internet.** Used to route a specific IP address or prefix to the Internet. **None.** Used to drop all traffic send to a given IP address or prefix. This route table is then associated with one or more subnets. Traffic originating in the subnet whose destination matches the destination IP range of a route table rule will instead be routed to the corresponding next hop IP address. The service running at this IP address is responsible for all onward routing. You can have multiple route tables, and the same route table can be associated to one or more subnets. Each subnet can only be associated to a single route table. All VMs in a subnet use the route table associated to that subnet. Do not apply a route table to a subnet if the route table contains a rule with a next hop address within that subnet. To do so could create a routing loop. For this reason, virtual network appliances should be deployed to dedicated subnets, separate from the resources that route through that appliance.

Demonstration – Custom Routing Tables

Create a Route Table on a subnet already present.

Determine Service Endpoint Uses

We already talk about network security groups. Network security groups can control from the VNet what I can talk to, what types of service with the service tags remember would let me say, hey I can't talk to storage in this region and what I can accept from. But what about on the service itself? Maybe on the service on the storage account or the MySQL database? I only want to be able to talk to a particular thing in a particular subnet and as is don't let me do that. So NSG is a great, they're focused on the traffic into and out of the VNet, it's focused on the VNet side.

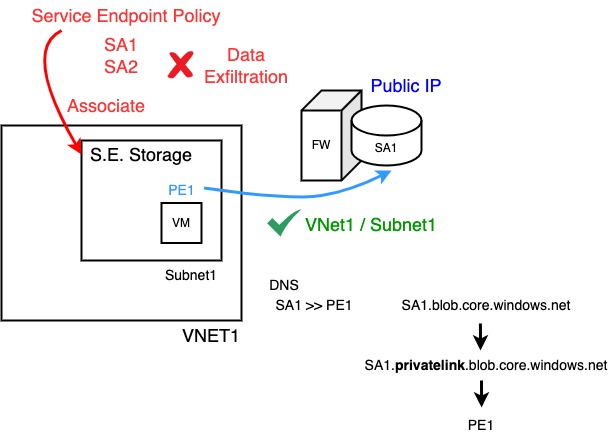
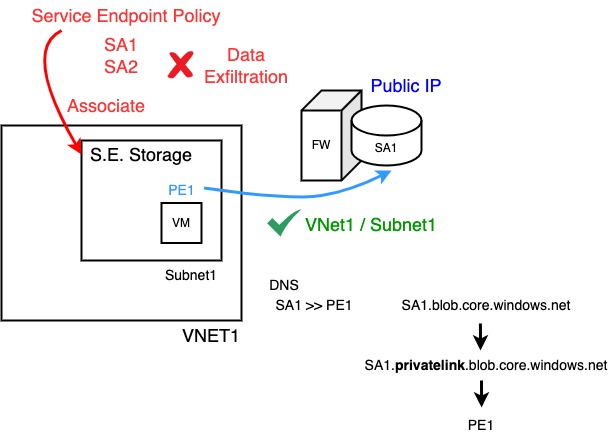


Service endpoints are a mechanism to integrate Azure PaaS services into your virtual network and access them through a Microsoft Azure backbone network instead of over the Internet. Today, Azure service traffic from a virtual network uses public IP addresses as source IP addresses. With service endpoints, service traffic switches to use virtual network private addresses as the source IP addresses when accessing the Azure service from a virtual network. This switch allows you to access the services without the need for reserved public IP addresses that are typically used in IP firewalls. We can say that Service endpoints prevent the exposure of data and services to Internet.

So, we have my Vnet1 and we have some service. It's just storage account. It doesn't matter, these are very similar so storage account one. We can start saying that most of the Azure PaaS services have their own firewall. I can do rules in terms of IP addresses and many other things to lock down. So, let's think through what this looks like. As we said, most of these services have a native firewall. And it can lock down and it can restrict things, so it could restrict things based on IP addresses and other things. But I want to be able to say hey, I've got this subnet. Subnet 1 and what I want is to only be able to things that live in this subnet to be allowed to talk to this storage account. So, on this firewall I want to say only let this subnet. I can't do it based on the IP range because remember this is all private IPS. It has no clue what some internal IP range is. I have to be able to identify the subnet. So, what I'm going to do is I can add a service endpoint. So, I'm going to add a service endpoint for storage. It now makes this subnet known to storage services and it's going to be a slightly faster route. Then, what I can say is on the firewall configuration, I'm going to say yes, allow in Vnet1 / Subnet1 but no other things, and we can see that. So, looking at the portal, in one of my virtual networks, I don't actually have to do this in advance. If I was just to. Go and create one of these rules it will say, hey, this subnet doesn't have the service endpoint, should I create it? But let's say we're ahead of ourselves. We can see service endpoints. And we can see I have added it for storage. That's super easy. We have the drop down. I could select other services that I want to enable service endpoints for. Then on an instance of that particular service it has to be the same region. So now we pick this one. And I'll go to its networking. I want to enable it from selected virtual networks and I'm going to say add an existing virtual network. I select the virtual network. And notice there's my subnet. And it doesn't say service endpoint required. These other ones do, so didn't add it, it would go and add it for me. But I could now just check that box. With this, the firewall on this storage account would only allow things in that particular subnet to talk to it. So that's a really nice capability that I can do there.

But the challenge still is that virtual network could still talk to others, could still do data exfiltration. So, the other thing I can create, is a service endpoint policy, and the service endpoint policy will basically say, hey, you're allowed to talk to storage account one, you're allowed to storage account two, and then I associate it with a subnet. Things in the subnet can only talk to storage account one and only storage account two they couldn't talk to storage account three. So, what this does is this stops data exfiltration. That's the whole point. I'm limiting. I can't bring it from storage account one and then copy the storage account 10, my evil storage account. I can't do that anymore by adding in that service endpoint policy.

Identify Private Link uses

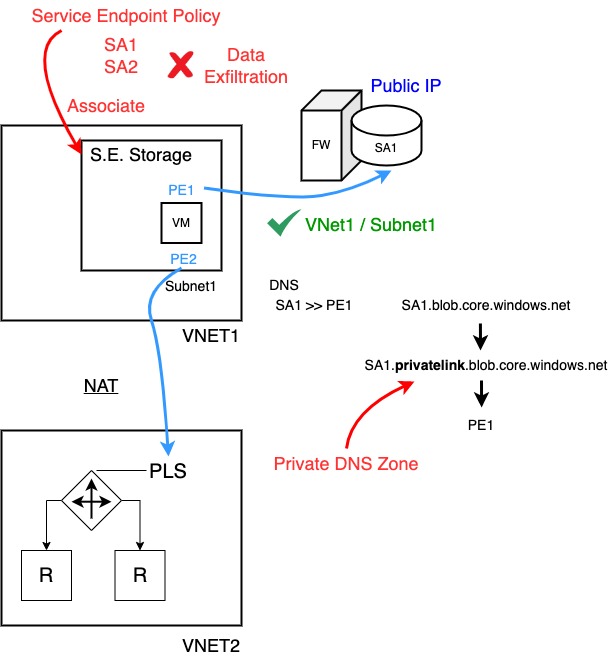
And we have private link. Actually, have a public IP address. This connection is actually going via some Internet based public IP. So, this is actually got this firewall is fronting a public IP. Now when I'm on an Azure resource, it doesn't actually bounce out to the Internet and come back, but it's still a public facing IP. It's still there on that resource and maybe I don't want that. So, what private link does is, I want to lock it down so now I get a privacy endpoint in my virtual network, in my particular subnet it's just an Ip address that represents that service. Anything can interact with that IP address. So, I could be on a connected network like on premises. As it has a path like ExpressRoute private peering or site to site VPN, it could use that IP address. Because it is instance level it also helps with data exfiltration because now, I could say hey, I only have private endpoints to these two storage accounts and my network security group could block. The service tag for storage. Because maybe the private endpoint is not in the IP range of that, so then I can still go and get to them. This also can solve the problem. If I want to use a service from a different virtual network, I don't have to peer them anymore. So, let's look at this.

I can use service endpoints and I can use private endpoints in the same subnet if I wanted to. But now the scenario is, I'm now going to add Private endpoint, it's just an IP address. But it now represents a specific instance of storage, it's storage account one. Because this now exists, if I wanted to, I could actually turn off the public IP. I don't want it no one can use the public IP. I can only use the private endpoint.

That is some other things that have to happen here. There’re some changes to DNS. So, DNS basically if I want to talk to storage account one, actually you need to talk to this private endpoint IP address. Now that's simplifying it massively. What actually happens is when I talk to BLOB, there's a storageaccount1.blob.core.windows.net. When I turn on private link it now becomes an alias to storage account one.privatelink.blob.core.windows.net which then that resolves to private endpoint one. So that's the magic that actually makes all of these things really work together. So that's a private endpoint, and I've got one of these in the environment so we can see it right here.

I've created a private endpoint for BLOB. On this particular storage account. So, my private endpoint, if we look at it. All the DNS configuration for me is integrated with my standard zones and we can even see it's created this privatelink.blob.core.windows.net so ordinarily on this storage account. Just come and look. We look at the endpoints. Storageaccountname.blob.core.windows.net now if we looked at that VM I have so this is in the VNet that's using that if I do nslookup on that storage account name what you can see it's now alias to a private link version of the zone and it resolves not to a public Ip but to that private IP address of the record. So now when anyone talked to BLOB, it's going to go talk to 10.0.1.4 which is the IP address. If we look at private link, so there's different ways I can actually interact with this, but if you look at private link it's a nice view. And my endpoints there 10.0.1.4 and that's just a regular Ip. If I was to go and look at my virtual network. And I looked at connected devices. It's just a NIC. So, anything connected to that could actually use. Now could be on premises could go and use that private endpoint. So that's fantastic for Microsoft PaaS services, storage accounts, Postgres, Cosmos DB.

To recap, Azure Private Link provides private connectivity from a virtual network to Azure platform as a service (PaaS), customer-owned, or Microsoft partner services. It simplifies the network architecture and secures the connection between endpoints in Azure by eliminating data exposure to the public internet. Azure Private Link keeps all traffic on the Microsoft global network. There's no public internet access. It’s global and there are no regional restrictions. You can connect privately to services running in other Azure regions. Services delivered on Azure can be brought into your private virtual network by mapping your network to a private endpoint. Private Link can privately deliver your own services in your customer's virtual networks. All traffic to the service can be routed through the private endpoint. No gateways, NAT devices, Azure ExpressRoute or VPN connections, or public IP addresses are required.

**Private Link Service**

What about if I had my own virtual network? And I have my own resources, could be VMs or AKS node pools. They're sitting behind a load balancer. And I want to be able to use it from this VNet. Now I could peer them. But what about if maybe they got this overlapping IP ranges? I can't peer them, I just don't wanna peer them. I'm a SaaS provider. I want to be able to provide this. I don't want to peer my VNet with the customer. Well, I can add a private link service. And then I can just add a private endpoint that goes to my private link service.

What private link does is network address translation anyway, so it abstracts my IP range to whatever this actually is, so I can do my own services as well. This is really powerful. This is replacing service endpoint most of the time in all honesty. People tend to use this now. There's a cost, there's a cost for creating a private endpoint, there's a cost for the traffic over it. But it gives me a lot richer set of functionalities when I compare it to for example the service endpoints.

## Configure Azure Load Balancer

Last time we talked about Virtual Network and vNIC with their associated IP Addresses. All these IP addresses have been within the virtual network. They're a private IP space. There is no special DMZ or Internet facing subnet in Azure. Some clouds have the concept of a subnet has been marked as external facing and is not the case in Azure. There is no subnet where everything just gets a public IP.

By default, Azure provides an outbound SNAT, Source Network Address Translation based on ports, enabling resources to access the Internet and receive responses.

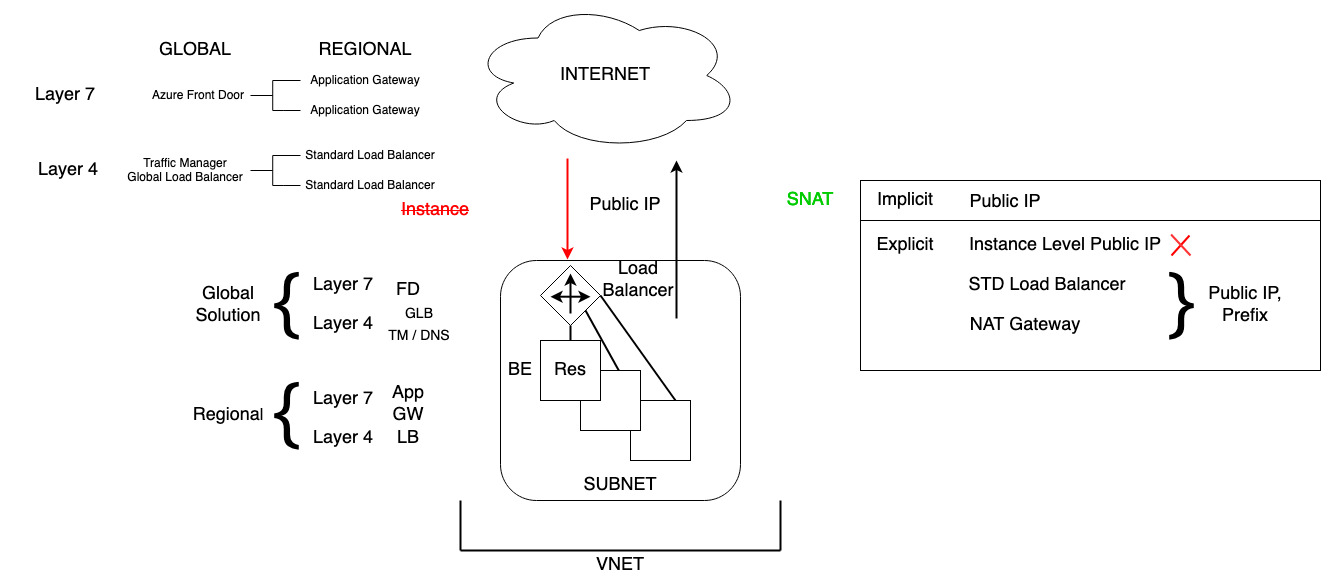
I have a virtual network, a vnet. Now I have a certain subnet. Within that subnet I have some resources. Could be many resources whatever it is and then floating out in wherever. Is the Internet. Things inside a virtual network just by default can talk to the Internet and they'll get a stateful response back. It won't let anything in, but we will let the response come back in. Now this is gonna work, so this is going to be a layout. So it's using SNAT. Remember that the Internet is just a whole set of networks that are connected and can route. They know how to get to something, or at least what's a hint for the next hop to go to, to be able to get to something. The Internet is just a bunch of connecting networks.

Well, because they're connected, they have to better route, which means they have to be valid IP address ranges. But we already said our virtual network is probably not, it's this RFC 1918. It's non routable. I need something at the edge that is a valid Public IP and what SNAT does is, when you private IP address make a request, it can specify a source port. And if you think about the port number, I think it's about 64,000 usable ports for anyone IP address. So, I'm going to map Port 4000 for example, to this particular request from VM1, Port 4001 I'm going to map to this request from VM2 and so on. So, one IP address can handle a huge number of contiguous requests from different clients. So, there's that is provided.

In different ways there's an implicit. So, an implicit is just there's some public IP created that you don't really see or know about. If I just have a VM, I try and talk to the Internet, it will do this implicit method. It will go and allocate a public IP and it will use it to provide this SNAT service when you go out to the Internet and let the responses come back.

I can do also explicit. This is where I'm actually defining something now. One way is I actually have an instance level. The public IP. This is saying to the VM, in your IP configuration on the Virtual Nic, you also have a public IP. Now the guest OS still doesn't really know what public IP, it has nothing to do with it. It's the Azure fabric. When response comes into this public IP, it will just send it to the NIC of the virtual machine. The VM doesn't know it has that public IP. But in this case, if I have an instance level public IP, I'll use that to do my outbound Internet.

Another option is using a standard load balancer. I'm saying standard. It would also work with basic or I can use something called a Nat gateway. So, this is a specific service designed to provide outbound. For all of these I give them a public IP or a public IP prefix where a prefix is a set of contiguous IP addresses. Maybe I need multiple addresses, I need them to be one after each other, so I can use a prefix. The benefit here of intent like Nat Gateway, it's designed purely to do NAT outbound. I can define IP addresses; it will dynamically use as many public IP's as it needed. As it fills up those 64,000 ports per IP, it will go and grab another one, depending on the amount of workload coming in.

Almost never do I want an instance level public IP. I want to use one of these to control it. And of course, the benefit here, this implicit one, I have no idea what this public IP address is going to be. It's just something. Whereas with both the standard load balancer and a NAT gateway I'm defining the public IP so that when if there were things behind these services that try and access some other thing on the cloud, well I know what IP the request is going to be coming from.

So, what about if I want to offer something to the Internet? I actually maybe I'm running a web server. I could give you an instance level public IP. We know that sucks. We don't wanna do that. More likely I'm going to put it behind a load balancer or an app gateway. Maybe there's some other network virtual appliance? I don't want to use the Azure native functionality. Maybe I'm using a NGINX or something.

There are other solutions to do that but saying it's gonna offer a service on the public IP and then distribute it to multiple backends. So, if the topic before was about getting outbound to the Internet, well now what I want is from the Internet to be able to talk to my service and so I have to have a public IP. It can't talk to my private IP from my VNet. Again, I could do instance level. I do not want to do. So, what's the problem with doing an instance level? It goes directly to a particular resource, so firstly, I've got no opportunity to add any kind of protection in front of it. The only protection is going to blindly send everything. Things like distributed denial of service, there's native protection at the VNet to stop huge scale distributed denial of service types attacks. Everything just gets passed straight through. It would be nicer to have something to be more specific about what it's going to pass through and when and maybe even add like web application firewalls to do some checking. Hey am I trying to do a SQL injection attack? But also, it's going to one resource, it's that resource is down for maintenance. I've only got the scale of 1 resource. I probably want to have multiple back-end resources so that I can distribute the workload and what I really want to think about owning this public IP, is some kind of load balancer.

Load balancing is one of the crucial requirements of a network design. The term load balancing refers to the distribution of workloads across multiple computing resources. Load balancing aims to optimize resource use, maximize throughput, minimize response time, and avoid overloading any single resource. It can also improve availability by sharing a workload across redundant computing resources.

Now remember that load balancer might be providing the outbound as well. There's a standard external load balancer and then what I do is I have a back end set that would have all these members in it when it gets to request because the public IP is bound to the front-end configuration of the load balancer and this is the back-end configuration.

It's doing health probing and only if you're healthy, we'll actually send you stuff and requests. There were different types of load balancer, so remember those layers again. So, layer four would be just the regular load balancer. Understand TCP, UDP. I have a certain amount of stickiness. Stickiness is well, if a request comes in, do I always send it to the same back-end member. So, there's five items by default, source IP, destination IP, source port, destination port, protocol. If all five of those things match, I'll go to the same instance. Maybe the port changes? Maybe the protocol might change.

I can do things where I don’t care about the ports, it just cares about the IPs and the protocol. I can also do things where it doesn't care about even the protocol. In this case the ports can change, the protocol can change, but as long as the source IP and destination IP at the same, it would go to the same back-end instance.

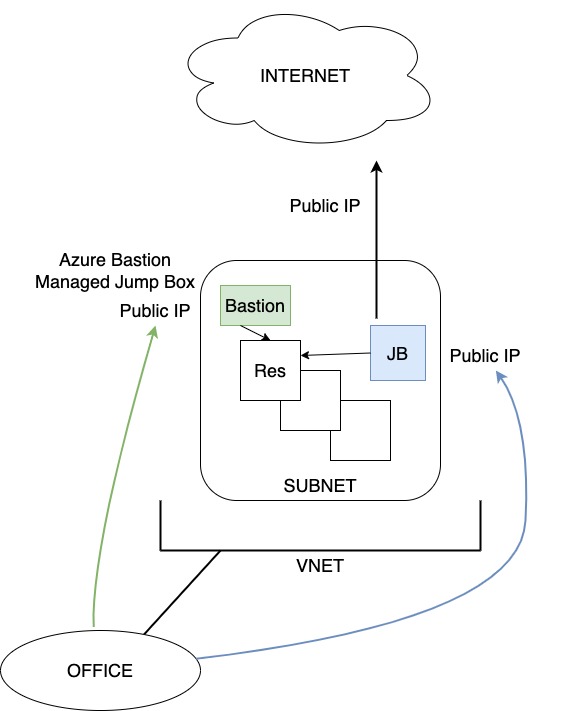
**Application Gateway**

But it doesn't understand a higher-level things like HTTP or HTTPS, it can't inspect the traffic. Cookie based affinity. Well, that's what an Application Gateway does. So, the other option is App Gateway. It understands layer seven, it can provide those additional features.

So, these solutions here are regional. They're providing and they've got to be careful when I say that. The service itself is in a region now as we saw App Gateway could technically point to things anyway it wanted to. But the App Gateway instance lives within a particular Azure region. But then there are global solutions. At layer seven, that would be Azure Front Door. At layer four there's actually the global load balancer now. But there's also DNS, so there's traffic manager which is DNS based so really would work across any type of service you wanted to.

So maybe I've got an Azure front door for example. So, this is my global. And then what it would actually point to are regional solutions. So, since Azure front door is layer 7, so HTTP, HTTPS, I would have at Gateway instances at the regional level. To then go and talk to some web service, could be containers, it could be Azure app service, it could be VMS, but that would be how I would make that distribution so a global service regional.

If instead these were not at gateways, if they were just standard load balancers, I probably wouldn't use Azure front door. Maybe I'd use traffic manager and it's DNS based to distribute between them or I could use the global load balancer which is also layer 4 so we have different architectures depending on what are the services we're actually using.

**Bastion**

So that's how I can talk to the Internet, offer things to the Internet, that that's how those things work. OK. Do not enable RDP or SSH or Winrm to the Internet. It's a great way to test how good your passwords are, because they're going to get hacked near instantly. If I have to offer it out to the Internet at minimum, use things like defender just in time. What it will do is it will lock down the rules that enable communication to it so that it can only come from your IP address for maybe 2 hours.

So, let's say this is my office. And in my office, I'm sitting there on my machine. The best-case scenario is I've got a link between my network and Azure. Because then I can just RDP or SSH to the private IP. I don't have to go via the Internet at all. But if I don't have this link, if I have to go via the Internet. Then, there are different things I could do.

I could use things like Azure Bastion. Azure Bastion is a managed jump box. A jump box I'm assuming is a familiar concept. The idea would be if I manually did it, I would create a VM and I'll call it jump box and that jump box VM would have a public IP. So, what I would do is I would first RDP to the jump box. I'm now sitting on this machine and then I could RDP to the private IP address. So, I'm jumping via that box to my target resource, but I have to be super careful about locking this down and protecting it. Azure Bastion lives in a certain subnet, but it can work for peered networks as well where I through the portal or with the native tools. Connect via the bastion which has a public IP and then it acts as that managed jump box service.

Choose a Load Balancer Solution

Azure provides various load balancing services that you can use to distribute your workloads across multiple computing resources - Application Gateway, Front Door, Load Balancer, and Traffic Manager.

**Application Gateway** provides application delivery controller (ADC) as a service, offering various Layer 7 load-balancing capabilities. Use it to optimize web farm productivity by offloading CPU-intensive SSL termination to the gateway. Optimize delivery from application server farms while increasing application security with web application firewall. It’s one of the regional load-balancing services (or private). It means that load-balancing services distribute traffic within virtual networks across virtual machines (VMs) or zonal and zone-redundant service endpoints within a region. You can think of them as systems that load balance between VMs, containers, or clusters within a region in a virtual network.

**Front Door** is an application delivery network that provides global load balancing and site acceleration service for web applications. It offers Layer 7 capabilities for your application like SSL offload, path-based routing, fast failover, caching, etc. to improve performance and high-availability of your applications. It’s one of the Global load-balancing services. They distribute traffic across regional backends, clouds, or hybrid on-premises services. These services route end-user traffic to the closest available backend. They also react to changes in service reliability or performance, in order to maximize availability and performance. You can think of them as systems that load balance between application stamps, endpoints, or scale-units hosted across different regions/geographies.

**Azure Load Balancer** is a fully managed load-balancing service, which is used to distribute inbound traffic across a pool of back-end servers running in an Azure virtual network. It can receive traffic on either Internet-facing or Intranet-facing endpoints and supports both UDP and TCP traffic. Azure Load Balancer operates at the transport layer (OSI layer 4) to route inbound and outbound connections at the packet level. It does not terminate TCP connections, and thus, it does not have visibility into application-level constructs. For example, it cannot support SSL offloading, URL path-based routing, or cookie-based session affinity. It provides low latency and high throughput, scaling to millions of network flows. It also supports automatic failover between back-end servers based on health probes and enables high availability applications. It’s a regional load-balancing service.

**Traffic Manager** is a DNS-based traffic load balancer that enables you to distribute traffic optimally to services across global Azure regions, while providing high availability and responsiveness. Because Traffic Manager is a DNS-based load-balancing service, it load-balances only at the domain level. For that reason, it can't fail over as quickly as Front Door, because of common challenges around DNS caching and systems not honoring DNS TTLs. It’s a global load-balancing service.

When selecting the load-balancing options, here are some factors that are considered when you select the **Help me choose** default tab in Azure load balancing:

* **Traffic type**. Is it a web (HTTP/HTTPS) application? Is it public facing or a private application?
* **Global versus. regional**. Do you need to load balance VMs or containers within a virtual network, or load balance scale unit/deployments across regions, or both?
* **Availability**. What is the service SLA
* **Cost**. See Azure Pricing. In addition to the cost of the service itself, consider the operations cost for managing a solution built on that service.
* **Features and limits**. What are the overall limitations of each service?

Implement a Public Load Balancer

Azure Load Balancer supports two modes: Public Load Balancer or Internal Load Balancer (or private). In each case, the frontend IP configuration defines the endpoint upon which the Load Balancer receives incoming traffic. The public load balancer is used to load-balance traffic for Internet-facing applications. The frontend IP configuration references a separate public IP address resource, which is used to receive inbound traffic. It’s also important to note that public load balancers can provide outbound connections to the Internet for VMs that are located inside your Azure virtual network. To make this happen, what public load balancers do is translate the private IP addresses of the virtual machines to public IPs. This allows them to communicate externally even though the VMs themselves have no public IP addresses. **Possible scenario**: Consider a scenario where internet traffic attempts to reach virtual machines in a web tier subnet that implements a public load balancer. Internet clients send webpage requests to the public IP address of a web app on TCP port 80. Azure Load Balancer intercepts the traffic and distributes the requests across the virtual machines in the load-balanced set according to the defined load-balancing rules.

Implement an Internal Load Balancer

The internal Load Balancer is used to load-balance traffic for Intranet-facing applications, or between application tiers. The frontend IP configuration references a subnet, and an IP address from that subnet is allocated using either dynamic or static assignment to the Load Balancer. **Possible scenario**: Suppose you have an Azure SQL Database tier subnet with several virtual machines, and you implement an internal load balancer. Database requests need to be distributed to the backend. The internal load balancer receives the database requests and uses the load-balancing rules to determine how to distribute the requests to the back-end SQL servers. The SQL servers respond on port 1443

Determine Load Balancer SKUs

Azure Load Balancer has 3 types of SKUs – Basic, Gateway and Standard. Each SKU is catered towards a specific scenario and has differences in scale, features, and pricing. If the load-balancer uses the standard pricing tier, then the public IP address must also use the standard pricing tier. Standard tier Load Balancers support both zone specific and zone redundant deployment options. The choice of deployment option is taken from the associated public IP address, rather than being explicitly in the Load Balancer properties.

Create Backend Pools

There are actually several pieces that make up a load balancer. You have the Frontend IP Configuration, the Backend Pool, Health Probes, and Load Balancing Rules. You also have High Availability Ports, Inbound NAT Rules, and Outbound Rules. The Frontend IP of a load balancer is the point of contact for clients. It can be a private IP address or a public IP address, depending on the type of load balancer. When someone needs to access an application that is load balanced, that person would access it through the Frontend IP. Load balancers can even have multiple Frontend IP addresses assigned to them. The Backend Pool is really just a collection of VMs or VM instances within a scale set that is configured to service the incoming requests to a load balancer. When a request for an application comes in on the Frontend IP, the load balancer sends the request to the backend pool. The load balancer will even automatically reconfigure itself whenever you add or remove instances from the backend pool. This ensures that the load balancer never sends traffic to an instance that has been removed. Health probes determine the status of the instances that are configured in the backend pool. They determine whether or not a specific instance is healthy and if it can receive traffic. When a health probe that you configure during load balancer set up stops responding, the load balancer will stop sending connections to the unhealthy instance.

Create Load Balancer Rules

Load balancing rules determine how inbound traffic gets distributed across the backend pool instances. A typical configuration for a load balanced Web server would include a load-balancing rule for port 80 traffic, or HTTP, that routes traffic from the front-end IP back to port 80 on the backend instances. When you configure a load balancer rule with 'protocol - all and port - 0', what you are doing is configuring high-availability ports. What this rule does is allow you to use a single rule to load balance all TCP flows and UDP flows that hit all ports of an internal standard load balancer. You would typically leverage this feature if you need to load balance a large number of ports.

Configure Session Persistence

Session persistence specifies how to handle traffic from a client. By default, successive requests from a client are handled by any virtual machine in your pool. You can modify the session persistence behavior as follows: **None (default)** – Any virtual machine can handle the request. **Client IP** - Successive requests from the same client IP address are handled by the same virtual machine. **Client IP and protocol** - Successive requests from the same client IP address and protocol combination are handled by the same virtual machine.

## Configure Azure Application Gateway

Azure application Gateway is another load balancer offering that is used to load balance web traffic to web applications. While the Azure load balancer offering operates at the transport layer of the OSI model, and routes traffic based on source IP and port to a destination IP and port, the Azure application Gateway uses additional HTTP attributes to make routing decisions. For example, when a request hits the application Gateway, the gateway can make routing decisions based on the URI path or based on host headers.

Implement Application Gateway

Let’s assume, for example, that we have a website that hosts images and video. You could use application Gateway to route incoming traffic to the correct set of servers based on the URL. Incoming requests for a URL that includes /images can be routed to the servers, or backend pool, that hosts your images. Incoming requests for a URL that includes /video can be routed to the pool that hosts your videos. The pool that you route traffic to can consist of VM’s, a VM scale set, or even on-prem servers. This is known as application layer routing and it happens at layer 7 of the OSI model. The Azure application Gateway offers several features that allow you to load balance and manage traffic to web applications. Some key features include SSL termination, autoscaling, web application firewall, and zone redundancy. SSL termination allows you to offload encryption and decryption to the gateway itself, rather than forcing the backend servers to handle it. The application Gateway consists of front-end IP addresses, listeners, request routing rules, HTTP, settings, backend pools, and health probes.

Setup Application Gateway Components

As is the case with load balancers, an application gateway has a front-end IP address associated with it. The front-end IP address can be public, private, or both. An application Gateway can support one public IP address or one private IP address. Listeners are used to check for incoming connection requests to the gateway. When you configure a listener, you can specify a protocol, a port, a hostname, and an IP address. If an incoming request matches what has been configured in the listener, the listener accepts the request. The gateway will then route the request to whatever backend members are configured in the rule. Speaking of rules, request routing rules are used to determine how to route traffic on the listeners. The request routing rule is essentially the glue that binds the listener, the backend pool, and the backend HTTP settings. What the rule does is pretty simple. It takes a request that has been accepted by the listener and forwards that request to the backend. The request routing rule defines which backend server pool that requests should be sent to. The HTTP settings include ports and protocols that are used to determine whether or not the traffic between the application Gateway and the backend servers is encrypted or not. HTTP settings also determine whether or not cookie-based session affinity should be used to keep a specific user session on the same server. The backend pools that I have mentioned can consist of NICs, VM scale sets, public IP addresses, internal IP addresses, FQDNs, and multitenant backends like App Service. Backend pools are essentially that - pools of resources that the gateway can direct traffic to. Lastly, the health probes are used to monitor the health of the resources contained within the backend pool. When an unhealthy resource is detected by the health probe, that resource is removed from service. The health probe will then continue monitoring the unhealthy resource. If it becomes healthy again, the resource is added back to the pool automatically.

## Configure Network Watcher

Network Watcher provides a central hub for a wide range of network monitoring and diagnostic tools. These tools are valuable across a wide range of network troubleshooting scenarios, and also provide access to other tools such as the Network Performance Monitor and Connection Monitor.

Describe Network Watcher Features

Network Watcher is enabled as a single instance per Azure region. It is not deployed like a conventional Azure resource, although it does appear as a resource in a resource group. Any subscription containing a virtual network resource will automatically have Network Watcher enabled. Otherwise, it can be enabled via the Azure portal, under **All Services** > **Network Watcher**. Also, you can see the Network Watcher status per region. Network Watcher can also be deployed via the command line (using the New-AzNetworkWatcher cmdlet or the az network watcher configure commands), which unlike the Azure portal, provides control over the resource group used. Some of the Network Watcher tools require the Network Watcher VM extension to be installed on the VM being monitored. This extension is available for both Windows and Linux VMs. It is installed automatically when using Network Watcher via the Azure portal. Here you can find some of the tools used in Network Watcher. **IP Flow Verify:** The IP Flow Verify tool provides a quick and easy way to test whether a given network flow will be allowed into or out of an Azure virtual machine. It will report whether the requested traffic is allowed or blocked, and in the latter case, which NSG rule is blocking the flow. It is a useful tool for verifying that NSGs are correctly configured. It works by simulating the requested packet flow through the NSGs applied to the VM. For this reason, the VM must be in a running state. **Next Hop**: The Next Hop tool provides a useful way to understand how a VM’s outbound traffic is being directed. For a given outbound flow, it shows the next hop IP address and type and the route table ID of any user-defined route in effect. Possible next hop types are Internet, VirtualAppliance, VirtualNetworkGateway, VirtualNetwork, VirtualNetworkPeering, VirtualNetworkServiceEndpoint, None (this is used for user-defined routes). **VPN Troubleshoot:** The VPN Troubleshoot feature in Network Watcher provides automated diagnostics of Azure VPN gateways and connections. The results provide a detailed report on gateway health and connection health, providing accurate pointers regarding common issues that might occur when enabling informed remediations. VPN Troubleshoot only supports route-based VPN gateways (not policy-based gateways or ExpressRoute gateways). It supports both IPsec Site-to-Site VPNs and VNet-to-VNet connections; it does not support ExpressRoute connections or Point-to-Site connections. During the troubleshooting process, logs are written to a storage account. This account must be created before starting the troubleshooting process. **Connection Troubleshoot:** Connection Troubleshoot is a Network Watcher feature designed to allow you to test the connectivity between an Azure VM or an App Gateway and another endpoint—either another Azure VM, or an arbitrary Internet or Intranet endpoint. This diagnostic tool can identify a range of problems, including guest VM issues, such as guest firewall configuration, low memory or high CPU, Azure configuration issues such as Network Security Groups blocking traffic, or routing issues diverting traffic. It can also diagnose other network issues, such as DNS failures. **Network Topology:** The Network Topology view in Network watcher provides a diagrammatic view of the resources in your virtual network. It is not a diagnostic or alerting tool. It is a quick and easy way to review your network resources and manually check for misconfiguration. A limitation of the tool is that it only shows the topology within a single virtual network. All common network resource types are supported, although for application gateways, only the backend pool connected to the network interface is shown.