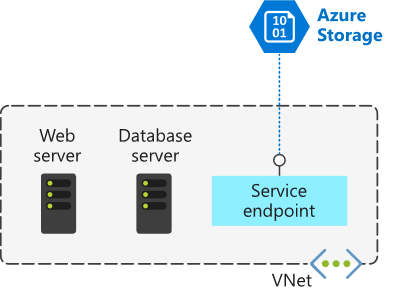
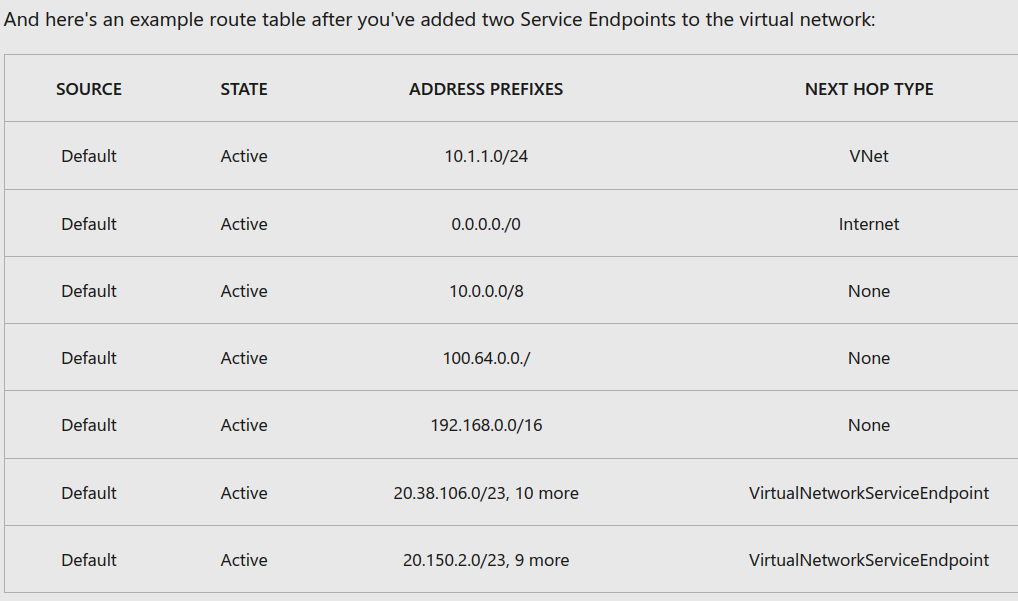
Module 7: Design and implement private access to Azure Services

# Chapter 2: VNET service endpoints

* Service Endpoints extend private address space in Azure by directly connecting to Azure services.
  + They secure Azure resources to only your VNET & traffic remains on the Azure backbone
  + 
    - By **default**, ALL Azure resources have public IP so will be exposed to the Internet (anyone could access your resources).
    - Service Endpoints can connect certain PaaS services directly to private IP in Azure

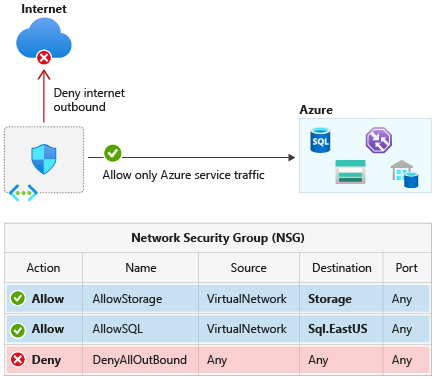
## How to Implement Service Endpoints

* 2 Preliminary Steps:
  + Turn off public access to the service.
  + Add the Service Endpoint to VNET.
* **Example**: *Enable Azure VMs to access service directly via Service Endpoints to your private IP*
  + 
    - All traffic for the service going to Service Endpoint remains internal to Azure.

## Create a Service Endpoint

* **Documentation**: https://learn.microsoft.com/en-us/azure/virtual-network/tutorial-restrict-network-access-to-resources
  + Steps to enable Service Endpoint to Azure Storage:
    - Enable service endpoint on a subnet
    - Use NSG rules to restrict access to Azure Storage
    - Create a service endpoint for Azure Storage

## Configure service tags

* Service Tags group IP addresses from a given Azure service (MS manages IP prefixes in the service tag)
  + Use service tags instead of specific IP addresses in NSG rules to allow/deny traffic for the corresponding service (ex. Tag = API Management)
    - 

# Chapter 3: Define Private Link Service/Private endpoint

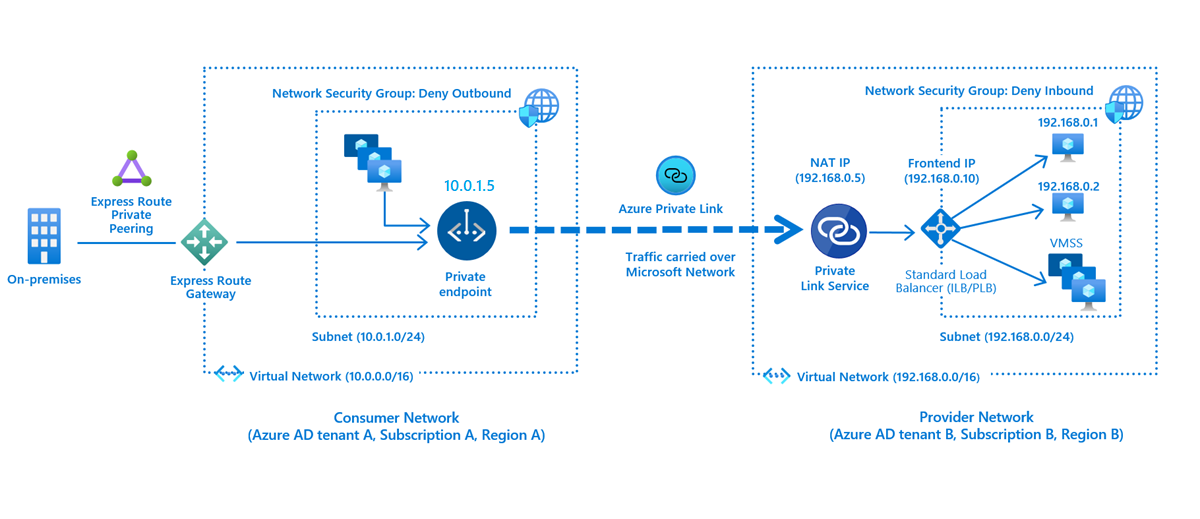
## Azure Private Link (PL)

* Lets you access Azure PaaS Services (**ex**. Azure Storage/SQL DB) over Private Endpoint in VNET.
  + Services normally have a public endpoint/public IP (so connection goes over Internet)
    - This also occurs over:
      * Peered VNETs
      * On-prem to Azure via ExpressRoute
      * Azure VNET connecting to 3rd party Azure resource
  + PL enhances security by replacing a resource's public endpoint w/ a private network interface (PE)
    - This will use the MS Backbone to communicate instead of Internet

## Azure Private Endpoint (PE)

* PEs act as network interface that enables a private connection between your VNET and an Azure service as it replaces the public IP of your resource (private IP)
  + Azure PE differs from Service Endpoint since:
    - Service Endpoint has publicly routable IP address.
    - Private endpoint has private IP in the address space of the VNET where PE is configured.

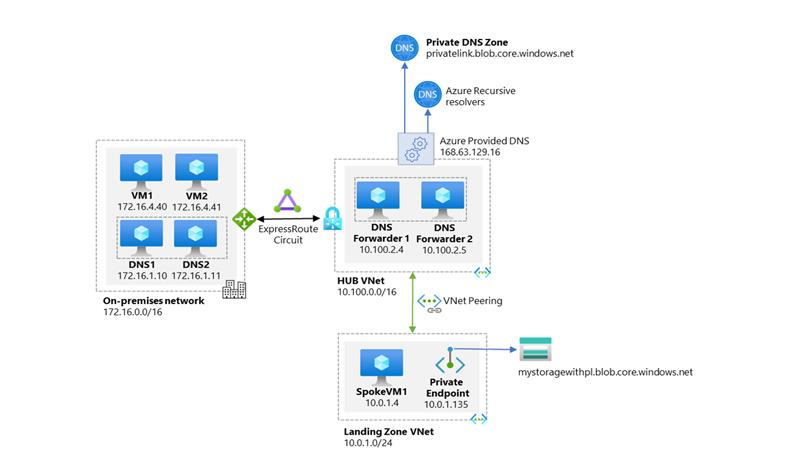
## Azure PL Service

* Private Link Service (PLS) lets you offer Private Link connections to your custom Azure services.
  + Your CX can create a private endpoint inside their VNET and map it to your service.
  + PLS receives connections from 2+ private endpoints; 1 private endpoint connects to 1 PLS.
    - 

## Private Endpoint properties

* Unique name for PE in a RGP.
* PE SUBNET to deploy/allocate private IPs in VNET
* Private Link resource to connect to (**ex**. Key Vault, SA)
  + Sub-resource to connect (**ex**. SA has blob as sub-resource)
* Automatic/manual connection approval method.
  + **Ex**. Azure RBAC permissions to auto-approve PEs.
* Check PE’s connection status (approved state)
* PEs enable connectivity between consumers in same VNET, regionally peered VNETs, globally peered VNETs and on-prem (VPN/ER)
* PE’s NIC is assigned dynamic private IP addresses from subnet and remains unchanged
* PE must be deployed in same region/sub as VNET.
* Private Link resource (**ex**. SA or KV) can be deployed in a diff region as VNET and PE.

# Integrate private endpoint with DNS

* Private DNS zones are typically hosted **centrally** in same Azure sub where the hub VNET is deployed.
  + High-level architecture for enterprise environments with central DNS resolution and where name resolution for Private Link resources is done via Azure Private DNS:
    - 
    - On-prem DNS servers have conditional forwarders set to Private Endpoint public DNS zone forwarder pointing to the DNS forwarders (10.100.2.4 and 10.100.2.5) hosted in hub VNET.
    - The DNS servers 10.100.2.4 and 10.100.2.5 hosted in hub VNET use the Azure-provided DNS resolver (168.63.129.16) as a forwarder.
    - All Azure VNETs have DNS forwarders (10.100.2.4 and 10.100.2.5) configured as the primary and secondary DNS servers.
    - 2 Conditions must be true to allow app teams freedom to create any Azure PaaS resources in their sub:
      * Ensure app teams can only deploy/access Azure PaaS services via Private Endpoints.
      * Ensure whenever Private Endpoints are created, corresponding records are auto-created in the centralized private DNS zone matching the service created.
    - Must remove the DNS record when the Private Endpoint is deleted.

## Azure Private Endpoint DNS configuration

* Configuration options for PE’s DNS settings:
  + Use the host file (testing only) on a VM to override the DNS.
  + Private DNS zone to override DNS resolution for PE.
    - Can be linked to VNET to resolve specific domains.
  + Use DNS forwarder (optional) to override DNS resolution for a Private Link resource.
    - Create a DNS forwarding rule to use a private DNS zone on DNS server hosted in VNET.

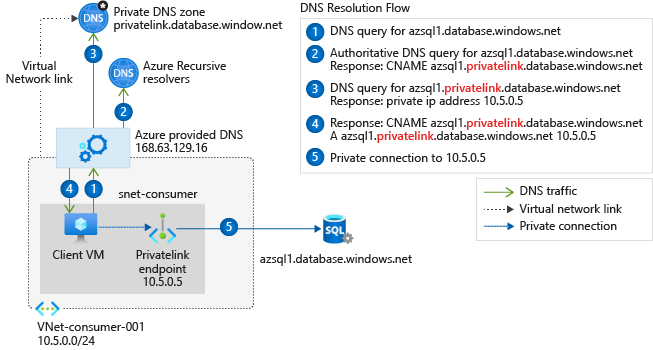
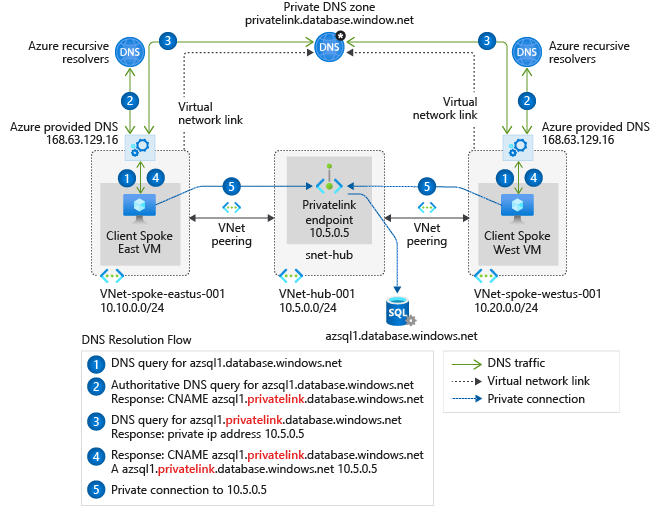
## Significance of IP address 168.63.129.16

* Used to facilitate a communication to Azure platform resources
  + Enables VM Agent to comm. w/ Azure platform (show "Ready" state)
  + Enables comm. w/ DNS virtual server to filter name resolution to resources (**ex**. VM) w/o custom DNS server
  + Enables Azure LB health probes to determine VMs’ health state
  + Enables VMs to obtain a dynamic IP address
  + Enables Guest Agent heartbeat messages for the PaaS role

## DNS configuration scenarios

* Scenarios for DNS resolution integrated:
  + VNET workloads w/o custom DNS server
  + On-prem workloads using a DNS forwarder
  + VNET/On-prem workloads using a DNS forwarder

## Virtual network workloads without custom DNS server

* In this scenario, client queries the PE IP address to the Azure-provided DNS service (168.63.129.16).
  + **This example**: uses the Azure SQL Database (or other service) recommended private DNS zone
    - Requires these resources:
      * Client VNET
      * Private DNS zone (**ex**. privatelink.database.windows.net) w/ type A record
      * PE info (FQDN record name & private IP)
    - 
  + **Other Example**: hub and spoke networking topology. Spoke VNETs share a PE and are linked to the same private DNS zone.
    - 
* Link for info: https://learn.microsoft.com/en-us/training/modules/design-implement-private-access-to-azure-services/4-integrate-private-link-dns

# Chapter 7: Integrate your App Service with Azure virtual networks

## Virtual network integration for Azure services

* Allows for private access to the service from VMs/Compute resources in the VNET
  + Options to do this:
    - Deploying dedicated instances of the service into a VNET.
      * They are privately accessed within VNET/on-premises.
    - Using Private Link to privately access a specific instance of the service from VNET/On-prem.
    - Service Endpoints let you use public endpoints by extending a VNET to the service.

## Configure App Service for regional VNET integration

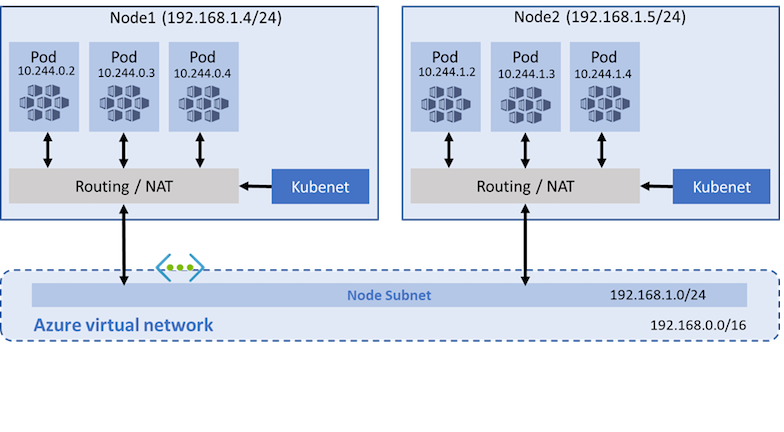
* VNET Integration doesn't enable your apps to be accessed privately. And is used in multi-tenant apps.
  + It gives your apps access to VNET’s resources, but **NOT** inbound to your app
* Azure App Service – 2 ways for VNET Integration:
  + Multi-tenant systems that support pricing plans except Isolated.
  + App Service Environment (ASE), deployed into VNET and supports Isolated pricing plan apps.
    - Doesn’t need VNET integration since already in VNET
* Private site access = app is accessible only from a private network (**ex**. Azure VNET)
  + It’s used only for outbound calls from app into VNET.
    - 2 Variations for integration w/ VNETs:
      * *Regional* VNET Integration
        + Must have a **dedicated subnet** you are integrating with when trying to connect to VNET in same region!
      * *Gateway-required* VNET Integration
        + Must have VNET Gateway provisioned on target VNET when connect to VNET in other regions/classic VNET in the same region
        + Provides access to resources ONLY in target VNET/peered VNETs (**ex**. VPNs).

It doesn't enable access to resources available across ER connections or Service Endpoints.

## Associate the App Service with the virtual network

* Regional VNET Integration enables app to access:
  + Resources in VNET of same region as app.
  + Resources in peered VNETs.
  + Service Endpoint secured services.
  + Resources across Azure ER connections.
  + Resources in VNET you are integrated with.
  + Resources across peered connections (**ex**. Azure ER).

## Configure Azure Kubernetes Service (AKS) for regional VNET integration

* To provide network connectivity, AKS clusters can use:
  + Kubenet (basic networking) --> Only nodes receive an IP address in the VNET’s subnet
  + Azure CNI (advanced networking).
* How AKS nodes receive an IP address in the VNET subnet, but not the pods:
  + 

### Limitations/considerations for kubenet

* Additional hop is required for kubenet (adds minor latency to pod comms).
* RTs and UDRs required for using kubenet (adds complexity)
* Direct pod addressing isn't supported for kubenet.
* Multiple kubenet clusters can't share a subnet (Azure CNI clusters can).
* **Which Network Model is appropriate?**
  + *Use kubenet when*:
    - Limited IP space.
    - Most pod comms are **within** the cluster.
    - No need for advanced AKS features (**ex**. virtual nodes, Azure Network Policy)
  + *Use Azure CNI when:*
    - Available IP space.
    - Most pod comms are to resources **outside** of the cluster.
    - No need to manage UDRs for pod connectivity.
    - Need AKS advanced features