Here’s a structured breakdown to help you **successfully instantiate Azure Private Link Service (PLS)** — *not to be confused with Azure Private Link* — for **exposing on-prem assets (like services or APIs) to customers over private connectivity**:

### **✅ Benefits of Azure Private Link Service**

Azure Private Link Service allows you to expose services running **behind a standard load balancer in your Azure VNet** to customers via **private endpoints in their own VNets**, even if your service originates from **on-prem via hybrid architectures**.

* **Private Access for Customers**  
   Customers connect to your service using private IP addresses over Azure backbone — no public internet exposure.
* **Enhanced Security Posture**  
   Reduces attack surface and supports granular NSG and NSX policies.
* **Controlled Exposure**  
   You control who can connect via approved private endpoints, reducing the risk of unauthorized access.
* **Hybrid Scenario Support**  
   Supports NVA or ExpressRoute-connected backends, so on-prem services can be made available via Azure PLS.
* **Scalable Multi-Tenant Design**  
   Ideal for ISVs or partners exposing multi-tenant services securely to multiple customers.

### **⚠️ Constraints**

Key limitations you must consider when deploying PLS:

* **Standard Load Balancer Required**  
   PLS only works with a Standard SKU Load Balancer (no Basic SKU support).
* **TCP/UDP Only**  
   Only supports TCP and UDP protocols — **no HTTP-layer routing (Layer 7)**.
* **No Direct On-Prem Hosting**  
   The service itself must be hosted in an Azure VNet (e.g., as an NVA or proxy to on-prem), not purely on-prem.
* **Inbound Connectivity Only**  
   Customers can initiate traffic, but not vice versa — your service can’t initiate outbound traffic to customer VNet.
* **Manual DNS Integration**  
   Customer must manually configure DNS or use Private DNS Zones — it’s not automatic.

### **🔍 Considerations for Successful Setup**

* **On-Prem Integration via ExpressRoute/VPN**  
   If the actual service is on-prem, integrate using **Azure VPN Gateway or ExpressRoute** to route traffic from Azure Load Balancer to the on-prem service.
* **NVA or Proxy in Azure**  
   For security and routing, deploy a **Network Virtual Appliance (NVA)** or **Azure Firewall** in a spoke VNet to proxy requests to on-prem assets.
* **Subnet Planning**  
   The PLS must be fronted by a **Standard Load Balancer with a dedicated subnet**, and the backend pool should point to reachable IPs (e.g., NVA or on-prem via gateway).
* **IP Whitelisting and NSGs**  
   Restrict access to approved customer VNets using **Private Endpoint connection approval** and **NSG/UDR rules**.
* **Monitoring and Auditing**  
   Integrate with **Azure Monitor, Network Watcher**, and **Diagnostic Logs** for visibility into PLS operations.
* **Cost Implications**  
   Private Link Services and Standard Load Balancer have **per-hour and per-GB egress costs** — evaluate based on expected usage.

### **🔗 Dependencies and Required Resources**

|  |  |
| --- | --- |
| **Resource/Dependency** | **Description** |
| **Azure VNet (Hub/Spoke)** | Required to host the Standard Load Balancer and proxy or NVA if on-prem is involved |
| **Standard Load Balancer (Internal)** | Mandatory to front-end your Private Link Service |
| **Backend Pool (e.g., NVA, VM, App)** | Must include reachable targets (e.g., on-prem routed via VPN Gateway or ExpressRoute) |
| **Private Link Service** | Configured with the Load Balancer frontend and backend |
| **NSG/UDR** | To restrict or route traffic between the PLS, backend, and customer endpoints |
| **ExpressRoute/VPN Gateway** | Needed if exposing on-prem resources via Azure PLS |
| **Private Endpoint (Customer Side)** | Customers will create a Private Endpoint connection to your PLS |
| **Private DNS Zone (Customer Side)** | For internal DNS resolution of the service exposed via PLS |

Let me know if you'd like this in a **Visio-style diagram**, **Terraform template**, or **step-by-step deployment guide**.

##### **You said:**

yes

###### **ChatGPT said:**

Great — since you'd like a **visual diagram**, a **Terraform template**, and a **step-by-step deployment guide**, here’s the full breakdown:

## **📊 Architecture Diagram: Azure Private Link Service for On-Prem Exposure**

Here’s a description of the architecture (you can visualize or diagram this in Visio, draw.io, or Lucidchart):

java

CopyEdit

Customer VNet  
└── Private Endpoint ─────┐  
 │  
 ▼  
 Azure Private Link Service (PLS)  
 ▲  
 Standard Load Balancer (Internal - TCP/UDP Only)  
 └── Backend Pool (e.g., Azure NVA, Azure VM)  
 |  
 Azure VPN Gateway / ExpressRoute  
 |  
 On-Prem Service (e.g., API, DB)

## **🛠️ Terraform Template Snippet (PLS & Dependencies)**

Assumes you already have the VPN/ExpressRoute & NVA in place.

hcl

CopyEdit

resource "azurerm\_resource\_group" "example" {  
 name = "rg-private-link-service"  
 location = "eastus"  
}  
  
resource "azurerm\_virtual\_network" "example" {  
 name = "vnet-pls"  
 address\_space = ["10.10.0.0/16"]  
 location = azurerm\_resource\_group.example.location  
 resource\_group\_name = azurerm\_resource\_group.example.name  
}  
  
resource "azurerm\_subnet" "lb" {  
 name = "subnet-lb"  
 resource\_group\_name = azurerm\_resource\_group.example.name  
 virtual\_network\_name = azurerm\_virtual\_network.example.name  
 address\_prefixes = ["10.10.1.0/24"]  
 enforce\_private\_link\_service\_network\_policies = true  
}  
  
resource "azurerm\_lb" "internal" {  
 name = "internal-lb"  
 location = azurerm\_resource\_group.example.location  
 resource\_group\_name = azurerm\_resource\_group.example.name  
 sku = "Standard"  
 frontend\_ip\_configuration {  
 name = "internal-fe"  
 subnet\_id = azurerm\_subnet.lb.id  
 private\_ip\_address\_allocation = "Dynamic"  
 }  
}  
  
resource "azurerm\_private\_link\_service" "pls" {  
 name = "pls-onprem"  
 location = azurerm\_resource\_group.example.location  
 resource\_group\_name = azurerm\_resource\_group.example.name  
 load\_balancer\_frontend\_ip\_configuration\_ids = [  
 azurerm\_lb.internal.frontend\_ip\_configuration[0].id  
 ]  
 ip\_configuration {  
 name = "pls-config"  
 private\_ip\_address = "10.10.1.100"  
 subnet\_id = azurerm\_subnet.lb.id  
 }  
 visibility {  
 subscription\_ids = ["<customer-subscription-id>"]  
 }  
 auto\_approval {  
 subscription\_ids = ["<customer-subscription-id>"]  
 }  
}

## **🧭 Step-by-Step Deployment Guide**

### **1. Prepare Your On-Prem Connection**

* Ensure ExpressRoute or VPN Gateway is set up.
* On-prem target (e.g., API, app, DB) must be routable from Azure.

### **2. Deploy VNet + Subnet for Load Balancer**

* Create a VNet (hub or spoke) and subnet for the Standard Load Balancer.
* Enable enforce\_private\_link\_service\_network\_policies = true.

### **3. Deploy NVA or Jump Host (Optional)**

* If routing traffic to on-prem, deploy an Azure VM or NVA in a backend subnet.

### **4. Create Internal Standard Load Balancer**

* Configure backend pool with the NVA or VM IP.
* Add a health probe and TCP load balancing rule.

### **5. Create Azure Private Link Service**

* Associate with LB frontend.
* Define auto-approval rules for customer subscriptions.
* Set up visibility scope (subscription, tenant, or all).

### **6. Customer: Create Private Endpoint**

* Customers create Private Endpoint to your PLS resource ID.
* DNS setup (manual or using Private DNS Zones) is required.

### **7. Test and Monitor**

* Validate traffic flow: Customer VNet → PLS → Load Balancer → On-Prem.
* Use NSG, UDR, and diagnostic logs to troubleshoot and monitor.

## **🧠 Optional Enhancements**

* **TLS Termination**: Use App Gateway + PLS if TLS/HTTPS required.
* **Rate Limiting/Inspection**: Insert Azure Firewall or 3rd-party NVA.
* **Auditing**: Enable diagnostic logs for PLS, NSGs, and Load Balancer.

NAT VMs and Load Balancers serve different but complementary roles in Azure Private Link Service architectures:

## **NAT VMs (Network Address Translation VMs)**

NAT VMs are needed when you have resources in private subnets that require outbound internet access but shouldn't be directly accessible from the internet. They act as a gateway for outbound traffic while maintaining security:

* **Outbound internet access**: Private resources can reach external services (updates, APIs, etc.) through the NAT VM
* **Security boundary**: Prevents direct inbound connections to private resources
* **Centralized control**: Single point for managing and monitoring outbound traffic
* **IP address management**: Provides a consistent public IP for outbound connections

## **Load Balancers in Private Link Service**

Load Balancers are essential components of Azure Private Link Service for several reasons:

* **Traffic distribution**: Distributes incoming private endpoint connections across multiple backend instances
* **High availability**: Ensures service continuity if backend instances fail
* **Scalability**: Handles increased load by spreading traffic across multiple targets
* **Health monitoring**: Continuously checks backend health and routes traffic only to healthy instances
* **Single entry point**: Provides a unified interface for the Private Link Service

## **Why Both Together?**

In a typical Private Link Service setup:

1. **Private Link Service** connects to a Standard Load Balancer (required)
2. **Load Balancer** distributes traffic to backend VMs in private subnets
3. **NAT VMs** provide outbound internet access for those same backend VMs when needed

The Load Balancer handles inbound private connectivity, while NAT VMs handle outbound internet connectivity. This separation of concerns provides both secure private access through Private Link and controlled outbound access through NAT, creating a robust and secure architecture.

Without the Load Balancer, you couldn't create the Private Link Service. Without NAT VMs, your private backend resources would be isolated from external services they might need to access.