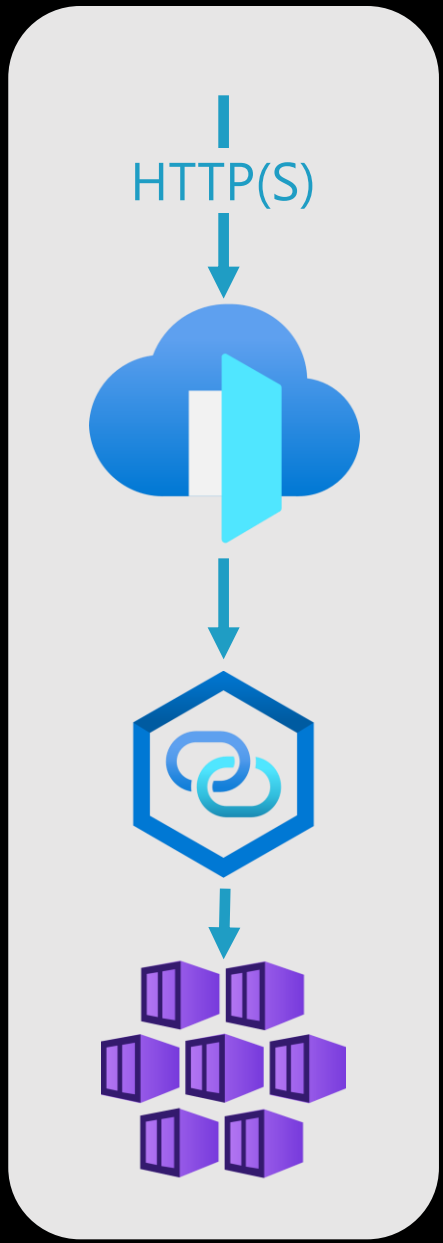
Exposing AKS apps with Front Door and PLS



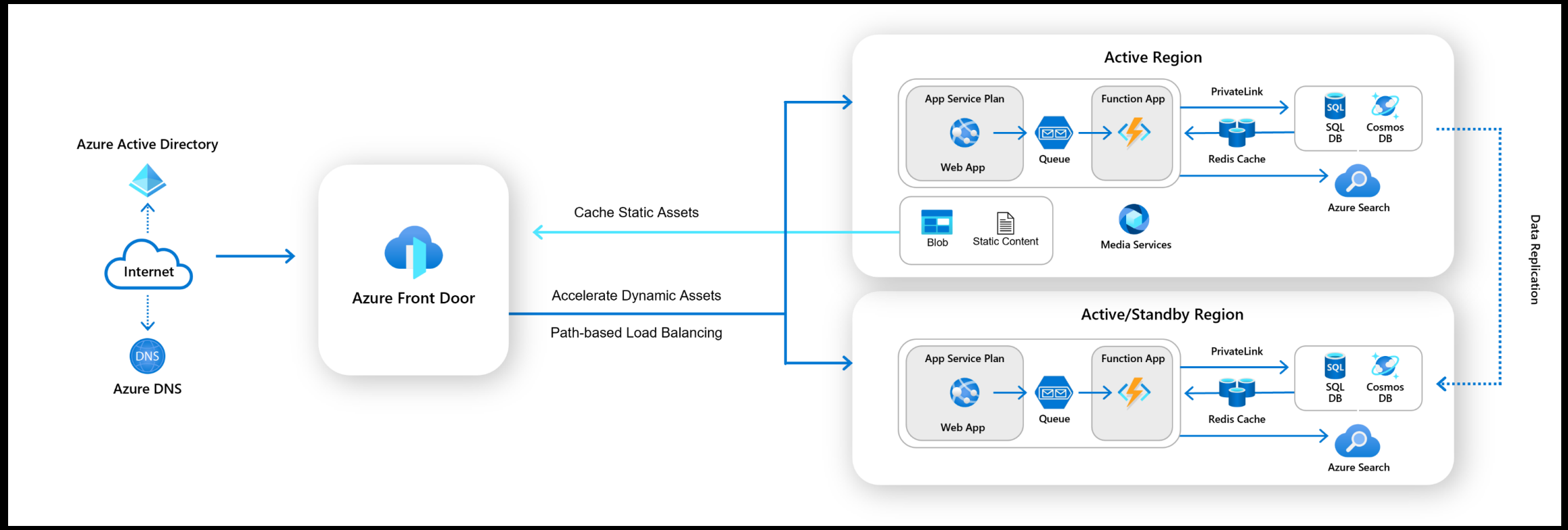
Front Door vs Azure load balancing services

- Global Load Balancer
- Layer 7 (HTTP/S)
- TLS offloading
- CDN
- WAF

	Application Gateway	Front Door	Load Balancer	Traffic Manager
Supported protocols	HTTP, HTTPS, HTTP2	HTTP, HTTPS, HTTP2	TCP, UDP	Any
Private load balancing	✓		✓	
Global load balancing		✓	✓	✓
Supported environments	Azure, non-Azure cloud, on prem	Azure, non-Azure cloud, on prem	Azure	Azure, non-Azure cloud, on prem
Host and path based load balancing	✓	✓		
TLS offloading	✓	✓		
Site acceleration		✓		
Security	WAF, NSG	WAF	NSG	

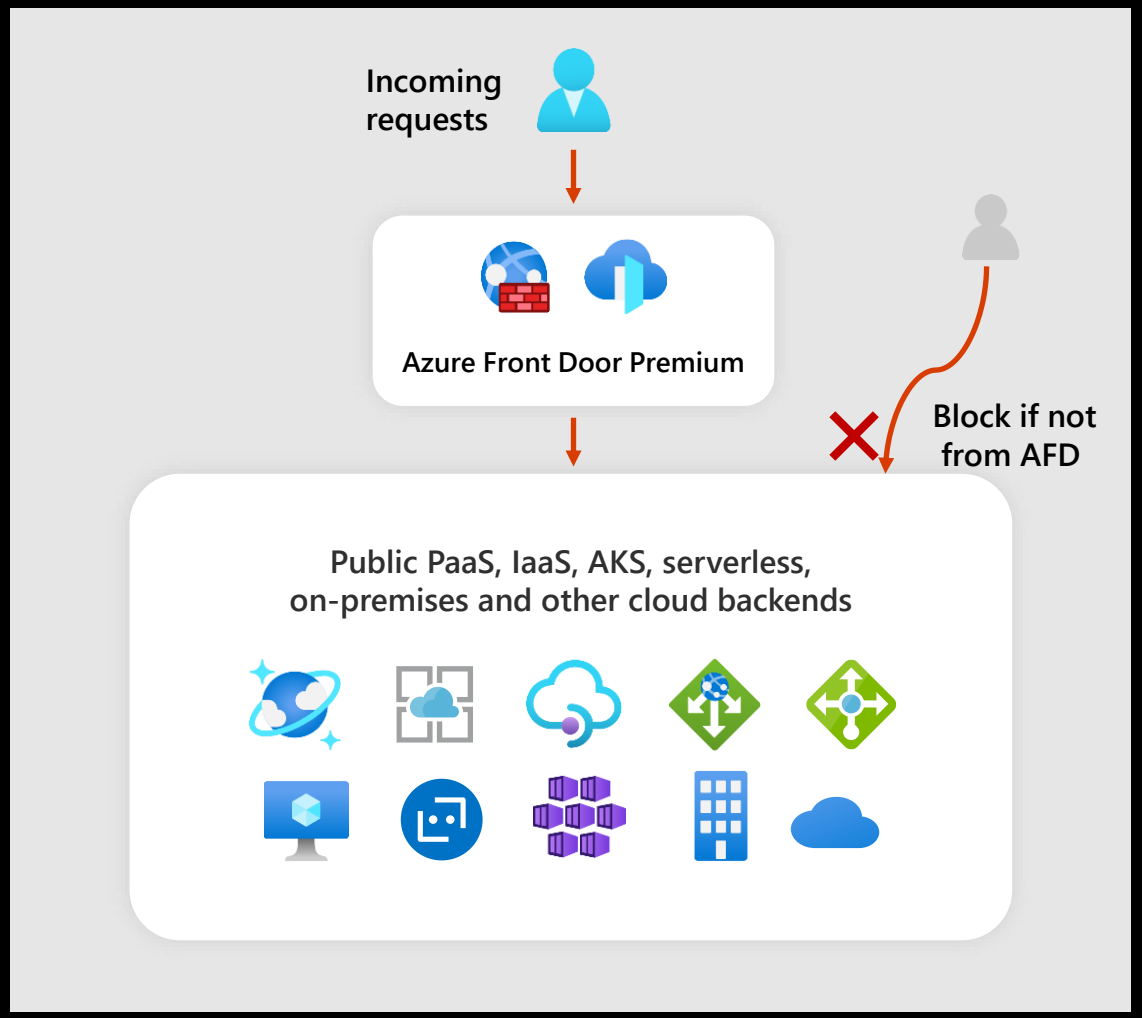
Front Door routes traffic to multiple regions

Front Door can load balance the traffic to applications deployed in multiple Azure regions or on-prem. This is useful for architectures using the following patterns: Active/Active, Active/Passive, Blue/Green and global applications.

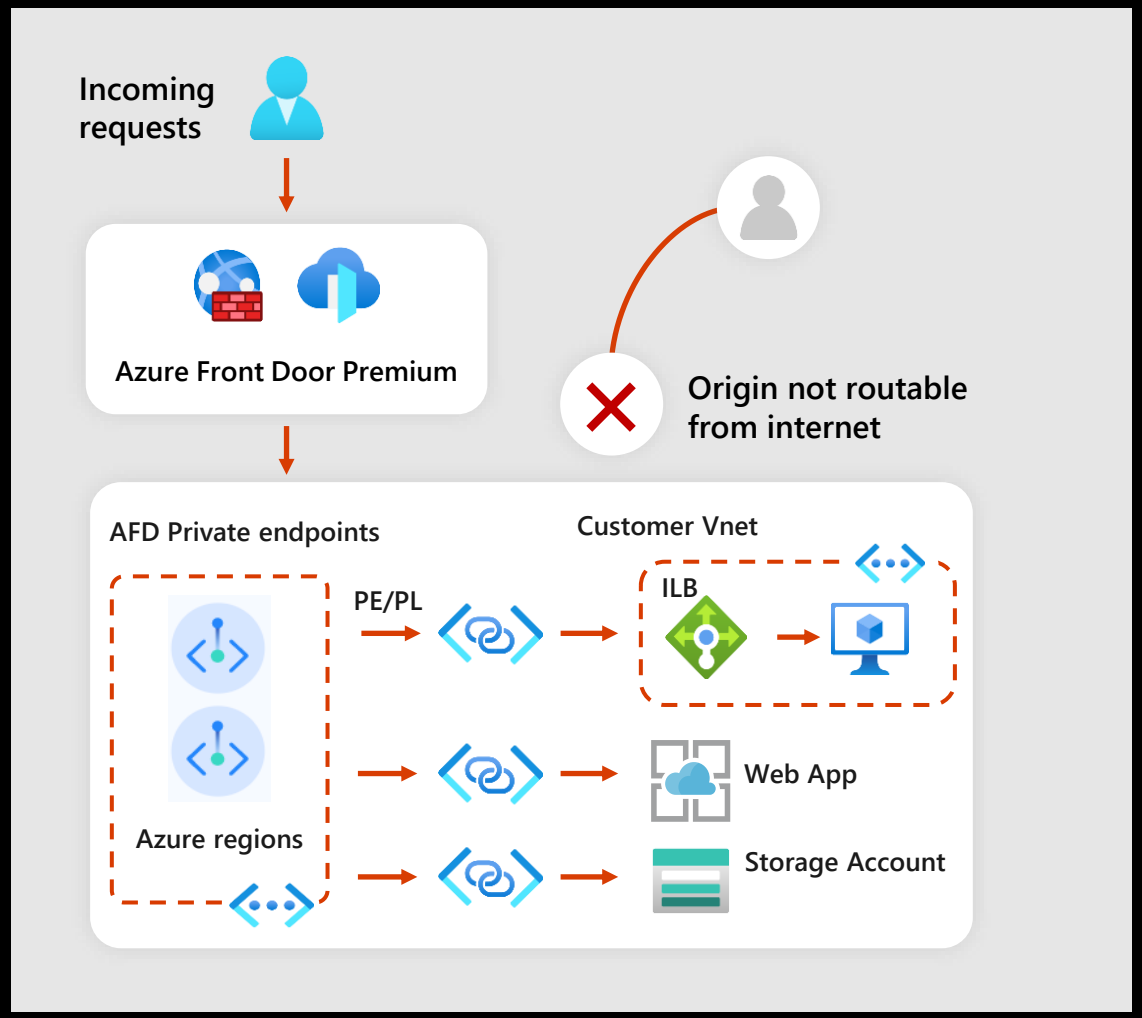


Front Door for public and private services

Public origins via IP and header restriction



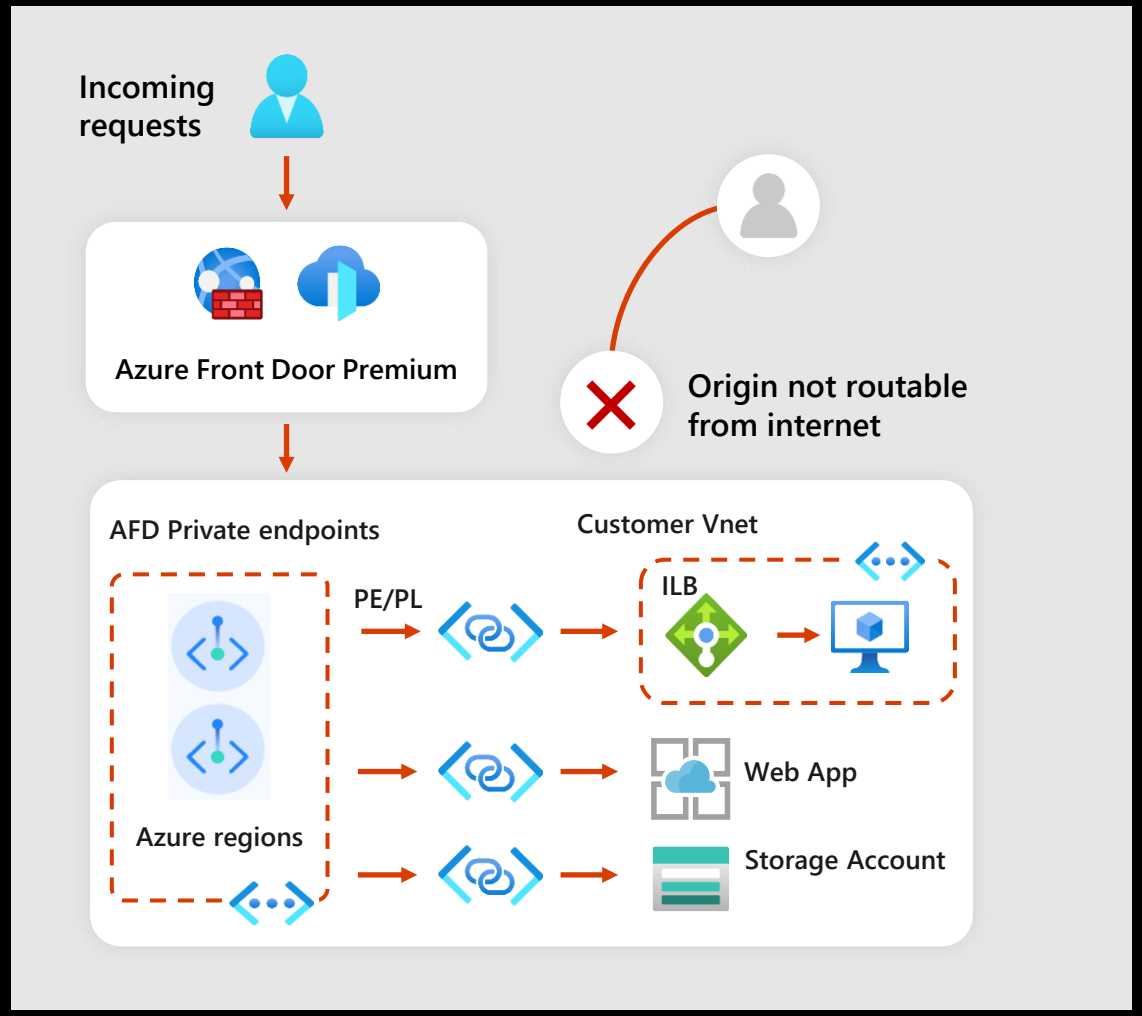
Private origins in Azure via private link service



Front Door and Private Link Service (PLS)

- Azure Front Door Premium can connect to a backend application via Azure Private Link Service (PLS).
- The backend system can be hosted in a virtual network or hosted as a PaaS service such as Azure Web App or Azure Storage.
- Private Link removes the need for your origin to be accessed publicly.
- When you enable Private Link to your origin in Azure Front Door Premium, Front Door creates a private endpoint on your behalf from an Azure Front Door managed regional private network.
- You need to explicitly approve Front Door private endpoint request at the origin.
- If you deploy a private origin using Front Door Premium and the Private Link Service (PLS), TLS/SSL offload is fully supported.

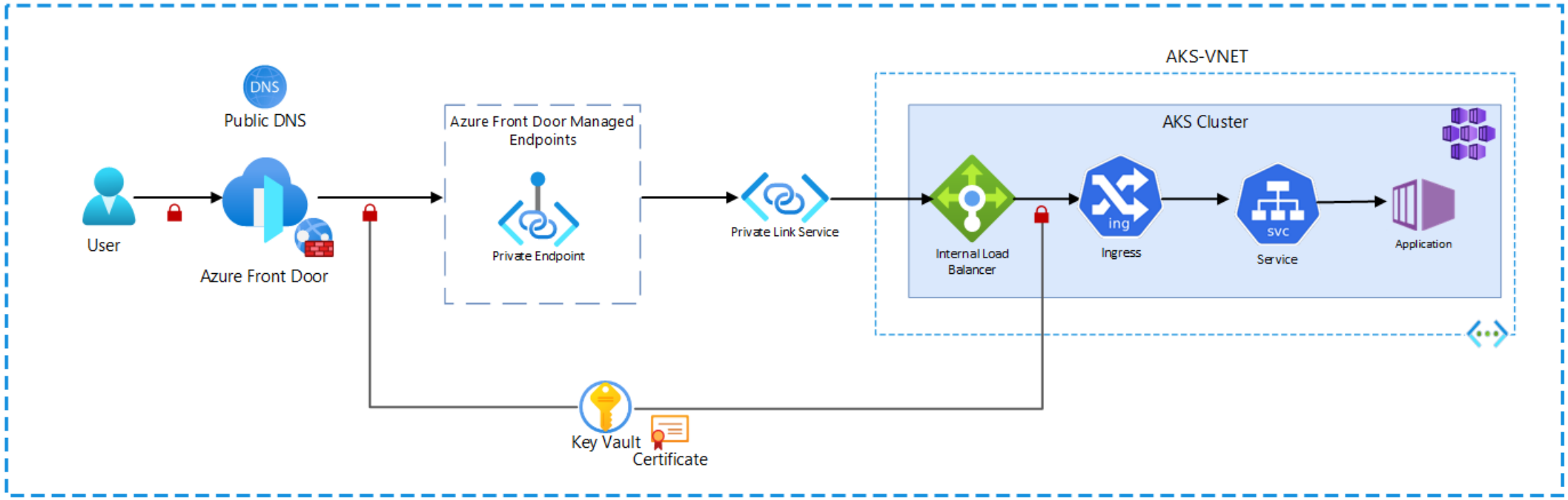
Private origins in Azure via private link service



Exposing AKS apps using Front Door and PLS

AKS creates an internal Load Balancer and Private Link Service through Kubernetes service annotations inside cluster VNET.

Then, Front Door can create a Private Endpoint in its own managed VNET to connect to the PLS of the cluster.



Service annotations to enable ILB & PLS

```yaml
apiVersion: v1
kind: Service
metadata:
  name: webapp-internal-service-pls
  namespace: webapp
  annotations:
    service.beta.kubernetes.io/azure-load-balancer-internal: "true"
    service.beta.kubernetes.io/azure-load-balancer-ipv4: 10.10.0.25
    service.beta.kubernetes.io/azure-pls-create: "true"
    service.beta.kubernetes.io/azure-pls-name: "pls-aks-service"
    service.beta.kubernetes.io/azure-pls-ip-configuration-subnet: "snet-aks"
    service.beta.kubernetes.io/azure-pls-ip-configuration-ip-address-count: "1"
    service.beta.kubernetes.io/azure-pls-proxy-protocol: "false"
    service.beta.kubernetes.io/azure-pls-visibility: "*"
    service.beta.kubernetes.io/azure-pls-auto-approval: "<subscription ID>"
spec:
  type: LoadBalancer
  selector:
    app: webapp
  ports:
  - port: 80
    targetPort: 80
```

Nginx Ingress annotations to enable ILB & PLS

```yaml
apiVersion: approuting.kubernetes.azure.com/v1alpha1
kind: NginxIngressController
metadata:
  name: nginx-internal-static-pls
spec:
  ingressClassName: nginx-internal-static-pls
  controllerNamePrefix: nginx-internal-static-pls
  loadBalancerAnnotations:
    service.beta.kubernetes.io/azure-load-balancer-internal: "true“
    service.beta.kubernetes.io/azure-load-balancer-ipv4: 10.10.0.30
    service.beta.kubernetes.io/azure-pls-create: "true"
    service.beta.kubernetes.io/azure-pls-name: "pls-aks-ingress"
    service.beta.kubernetes.io/azure-pls-ip-configuration-subnet: "snet-aks“
    service.beta.kubernetes.io/azure-pls-ip-configuration-ip-address-count: "1"
    service.beta.kubernetes.io/azure-pls-proxy-protocol: "false"
    service.beta.kubernetes.io/azure-pls-visibility: "*"
    service.beta.kubernetes.io/azure-pls-auto-approval: "<subscription ID>"
```

More configuration options for ingress

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: frontdoor-ingress
  annotations:
  kubernetes.io/ingress.class: nginx
  nginx.ingress.kubernetes.io/enable-modsecurity: "true"
  nginx.ingress.kubernetes.io/modsecurity-snippet: |
    SecRuleEngine On
    SecRule &REQUEST_HEADERS:X-Azure-FDID \"@eq
0\"  \"log,deny,id:106,status:403,msg:\'Front Door ID not present\'\"
    SecRule REQUEST_HEADERS:X-Azure-FDID \"@rx ^(?!xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx).*$\"  \"log,deny,id:107,status:403,msg:\'Wrong Front Door ID\'\"
spec:
  #section omitted on purpose
```

learn.microsoft.com/en-us/azure/frontdoor/origin-security