CSI ASSIGNMENT 7
LOAD BALANCER

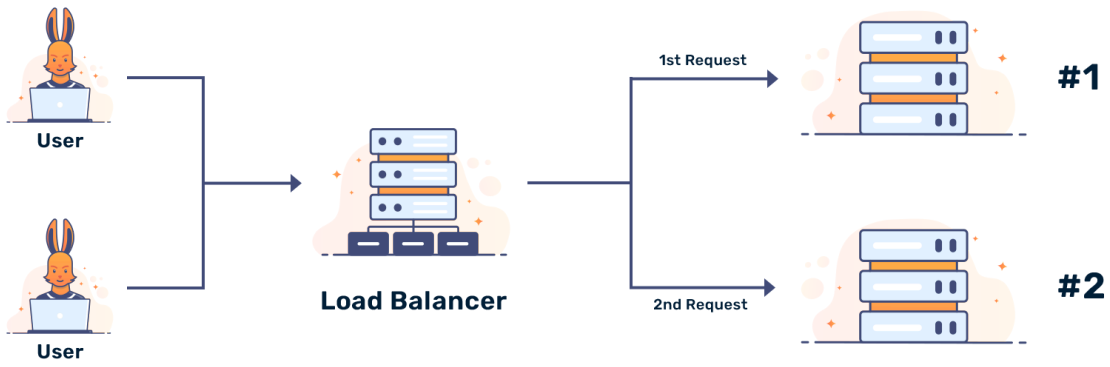
1.What is Azure Load Balancer?

Azure Load Balancer is a service that provides a single point of access to distribute incoming network traffic to multiple VMs. The load balancer automatically distributes incoming traffic across multiple VMs in a manner that provides high availability, performance, and fault tolerance. It provides two main types of load balancing: external load balancing and internal load balancing. External load balancing provides access to VMs from the internet, while internal load balancing provides access to VMs from within the same network.

There are two types of Azure Load Balancer:

1. Public Load Balancer: This type of load balancer is used to distribute traffic coming from the internet to multiple virtual machines in an Azure virtual network. It provides high availability by distributing inbound traffic across multiple VMs and scales resources as demand changes.

2. Internal Load Balancer: This type of load balancer is used to distribute traffic within a virtual network, from one set of VMs to another. It can also balance traffic from an on-premises network to a virtual network in Azure. It is mainly used for providing high availability and scalability for internal resources such as web servers, application servers, and databases.



- **Azure Load Balancer**
    - Layer 4 service that supports both inbound and outbound scenarios, for public (internet-facing) or internal (private) applications.
    - Designed for **high availability, low latency**, and can scale to millions of TCP/UDP flows microsoft.
    - Two SKUs:
        - **Basic**: no availability zone support, fewer features.
        - **Standard**: zone-redundancy, richer health probes, SLA-backed.

- Two types:
  - **External/Public**: exposes frontend via a Public IP for internet traffic.
  - **Internal/Private**: uses a private IP in VNet/subnet; ideal for internal apps or hybrid scenario.

- **Why use it**:
  - Distributes incoming requests uniformly across multiple VMs.
  - Improves fault tolerance via health probes to avoid unhealthy backends.
  - Cost-effective and built-in Azure service—no additional infrastructure needed.

2. Azure Load Balancer Configuration

Here's a high-level overview of the steps involved in configuring Azure Load Balancer:

· Create virtual machines: To use Azure Load Balancer, you first need to create one or more virtual machines (VMs) that will serve as the backend for the load balancer. Ensure that the VMs are properly configured, with the required software and services installed and running.

CSI ASSIGNMENT 7
LOAD BALANCER

1. Create a load balancer: In the Azure portal, navigate to the Load Balancer service, and click on the “Create Load Balancer” button. Fill in the required information, such as the load balancer name, subscription, resource group, and location.

2.Configure the frontend IP address: The frontend IP address is the public IP address that will be used to access the load balancer. You can either assign a new public IP address to the load balancer or use an existing one.

3· Configure the backend pool: The backend pool is a collection of VMs that will receive incoming traffic from the load balancer. You need to add the VMs that you created earlier to the backend pool, and assign them to the load balancer.

4· Create load balancing rules: Load balancing rules define how incoming traffic will be distributed across the backend VMs. You can create load balancing rules for different protocols, such as TCP, UDP, and HTTP/HTTPS.

5· Create probes: Probes are used to monitor the health of the backend VMs. You can create probes for different protocols,

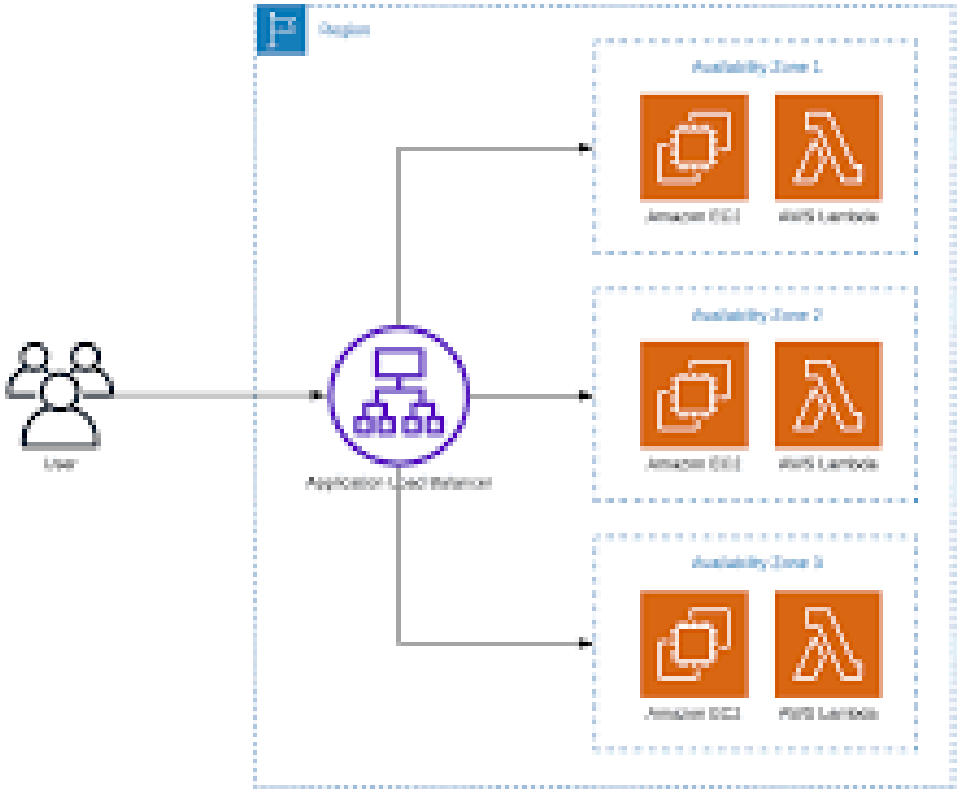
such as TCP, HTTP, and HTTPS, to ensure that the backend VMs are healthy and able to receive incoming traffic.

6· Test the configuration: Once you have completed the configuration, you can test the load balancer to ensure that it's working as expected. You can do this by accessing the frontend IP address from a web browser or other client, and verifying that incoming traffic is being distributed correctly to the backend VMs.

- **Define components**:
    - Resource Group, VNet + subnets (e.g. frontend/backend), Load Balancer (external & internal), Public IP (for external), backend pools, health probes, LB rules, NSGs.
- **Design diagram**: Visualize two VMs behind each LB, with a test/client VM. Include NAT Gateway or outbound config for ILB.
- **Naming & region**: e.g. rg-lb-eastus, use same region for all resources.
- **Security**:
    - NSGs: Allow probe ports (e.g. 80/443), LB rule ports, management RDP/SSH.
    - ILB: restrict NSG to VNet or peered networks only.

3. Create External Load Balancer — Azure Portal

1. **Resource Group** → +Create → name, region.
2. **Public IP** → SKU: Standard (static), TCP/UDP support.
3. **Load Balancer** → type: Public; assign the new Public IP.
4. **Frontend IP config** auto-created.
5. **Backend pool** → +Add: choose VMs (create two Windows or Linux VMs in same VNet).
6. **Health probe**: add TCP or HTTP; specify port (e.g., 80) and probe settings (interval, unhealthy threshold).
7. **Load balancing rule**: protocol TCP, frontend → backend port, backend pool and probe selected.
8. **NSGs**: inbound rule for port 80 (internet source); RDP/SSH to VMs.
9. **Install web server** (IIS/Apache) on each VM with custom page (hostname) to distinguish.
10. **Test**: browse public IP – refresh to see round-robin names.



4. Create Internal Load Balancer — Azure Portal

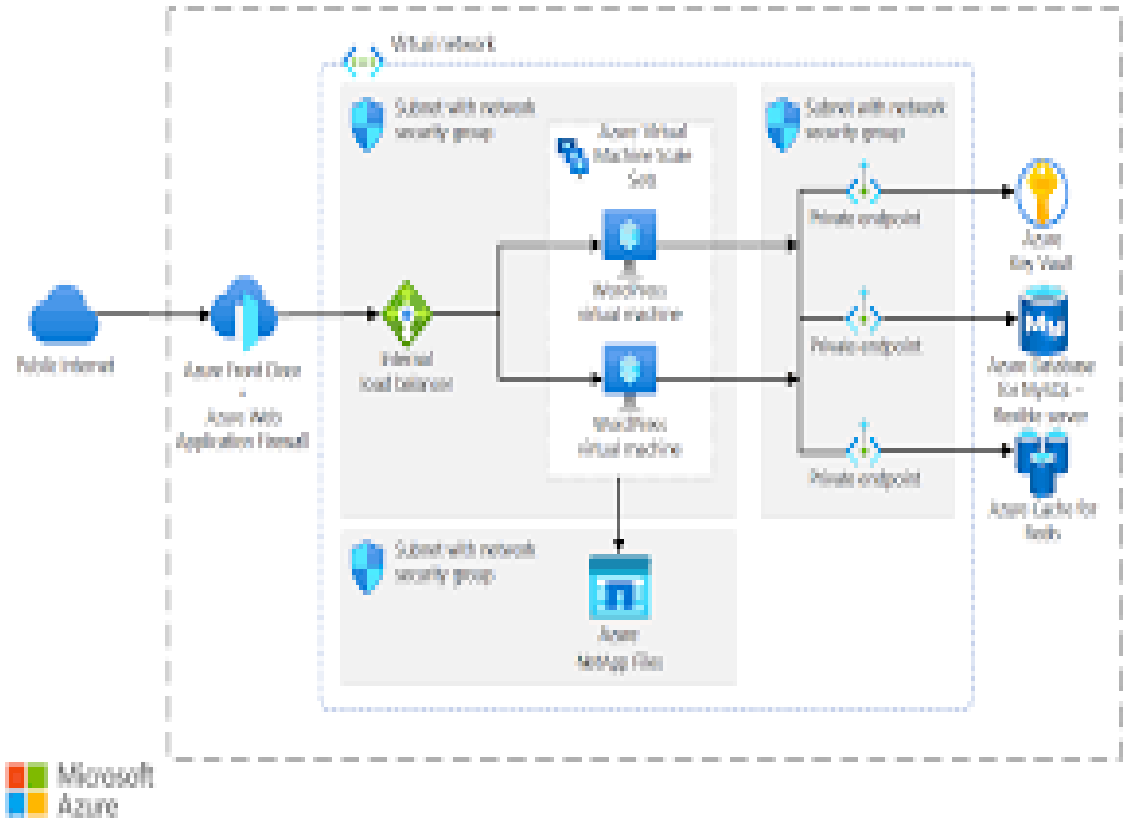
1. As before, but choose **Internal** type.
2. **Frontend IP**: assign static private IP (e.g. 10.0.1.4).
3. **Backend pool**, **probe**, and **LB rule** identical to external LB.
4. **NAT Gateway**: if backend VMs require internet access, attach NAT Gateway to their subnet – follow official quickstart steps.
5. **Test VM**: deploy a separate VM in same subnet/VNet.

6. **Verify**: connect via Bastion or SSH to test VM → browser/curl ILB IP to confirm rotating responses.

5. Verification & Testing

- **Basic tests**: Use curl/wget or browser; note backend response name.
- **Failover simulation**: Stop one VM → refresh; confirm requests still served by healthy instance.
- **Health metrics**: In LB pane, view Metrics, health probe status.
- **Advanced**: perform throughput tests (e.g. Apache Benchmark ab).
- **Monitoring**: Enable diagnostics logs in Azure Monitor; analyze probe failures and LB metrics.

CSI ASSIGNMENT 7
LOAD BALANCER



6. CLI Automation — Azure CLI

Bash code

```
az group create -n rg-lb -l eastus
az network vnet create -g rg-lb -n vnet-lb --address-prefix
10.0.0.0/16 --subnet-name subnet-lb --subnet-prefix 10.0.1.0/24
az network public-ip create -g rg-lb -n pip-ext --sku Standard
--allocation-method Static
az network lb create -g rg-lb -n extLB --sku Standard
--public-ip-address pip-ext
az network lb frontend-ip create -g rg-lb --lb-name extLB -n
extFront
az network lb backend-pool create -g rg-lb --lb-name extLB -n
extBack
```

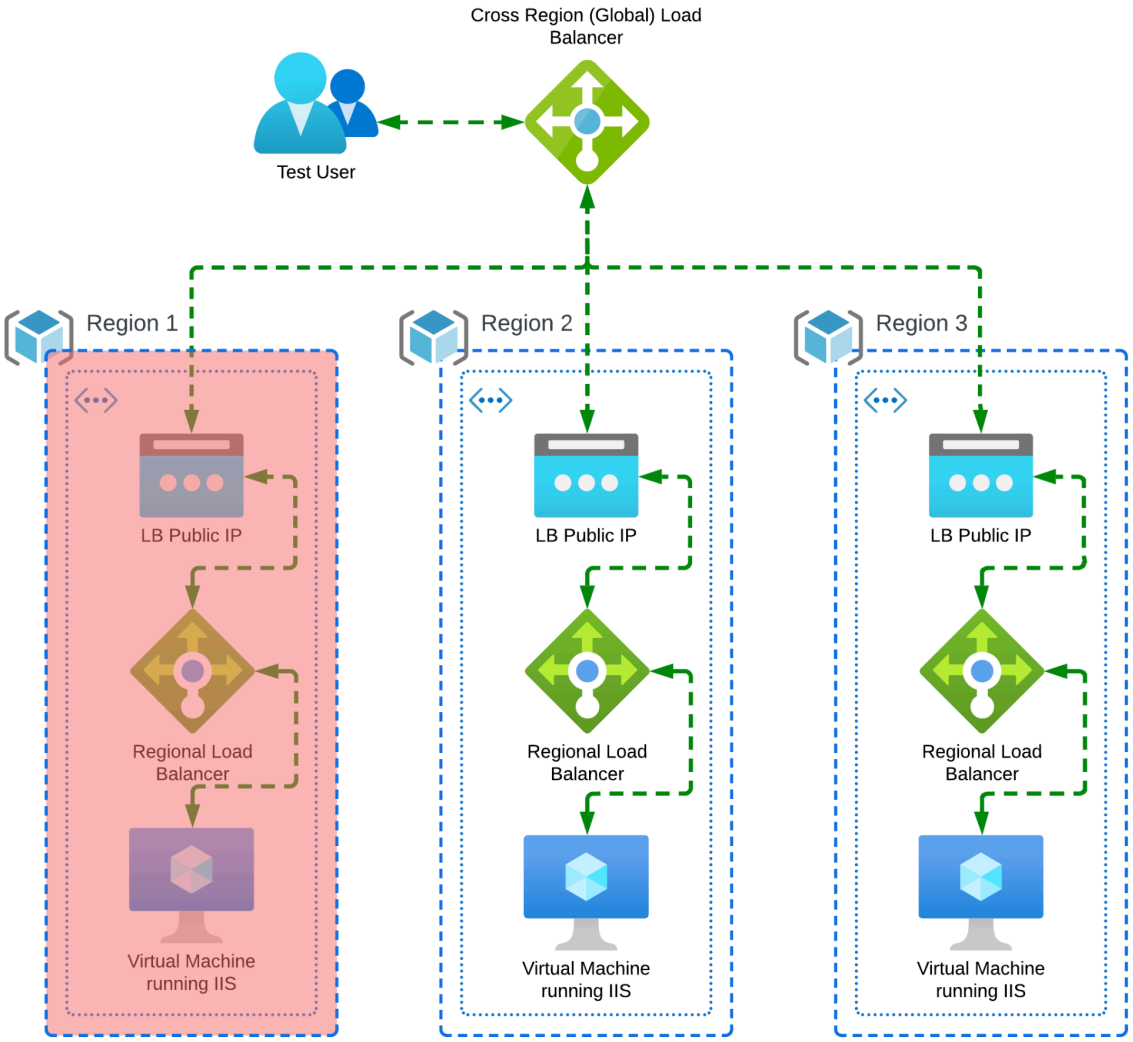
CSI ASSIGNMENT 7
LOAD BALANCER

```
az network lb probe create -g rg-lb --lb-name extLB -n probe80
--protocol Tcp --port 80
az network lb rule create -g rg-lb --lb-name extLB -n lbRule80
--protocol Tcp --frontend-port 80 --backend-port 80
--frontend-ip-name extFront --backend-pool-name extBack
--probe-name probe80
# then create VMs and attach NICs to backend pool…
```

- **ILB CLI** follows quickstart steps with --internal flag (Portal doc)

- Parameterize region, subnet, ports, instance count for automation.

CSI ASSIGNMENT 7
LOAD BALANCER



7. PowerShell Automation

Powershell code

```
$rg = New-AzResourceGroup -Name rg-lb -Location eastus
$vnet = New-AzVirtualNetwork -ResourceGroupName rg-lb -Name
vnet-lb -AddressPrefix 10.0.0.0/16 -Subnet
@(@{Name='subnet-lb';AddressPrefix='10.0.1.0/24'})
$pip = New-AzPublicIpAddress -ResourceGroupName rg-lb -Name
pip-ext -Sku Standard -AllocationMethod Static
$lb = New-AzLoadBalancer -ResourceGroupName rg-lb -Name extLB
-Sku Standard -FrontendIpConfiguration
@{Name='extFront';PublicIpAddress=$pip}
```

```
$lb | Add-AzLoadBalancerBackendAddressPoolConfig -Name extBack |
Add-AzLoadBalancerProbeConfig -Name probe80 -Protocol Tcp -Port
80 -IntervalInSeconds 15 -ProbeCount 2 |
Add-AzLoadBalancerRuleConfig -Name lbRule80 -Protocol Tcp
-FrontendPort 80 -BackendPort 80 -FrontendIpConfiguration
$lb.FrontendIpConfigurations[0] -BackendAddressPool
$lb.BackendAddressPools[0] -Probe $lb.Probes[0] |
Set-AzLoadBalancer
# Add VMs, NICs to the backend pool accordingly
:contentReference[oaicite:46]{index=46}
```

8. Infrastructure as Code

- Use template @2025-05-29 to create VNet, NAT GW, Bastion, ILB, backend VMs.

Highlighted file structure:

```
param adminUsername string
param adminPassword string
resource vnet 'Microsoft.Network/virtualNetworks@2020-11-01' = { ... }
resource natgw 'Microsoft.Network/natGateways@2020-11-01' = { ... }
resource lb 'Microsoft.Network/loadBalancers@2020-11-01' = { ... }
resource vms 'Microsoft.Compute/virtualMachines@[...]' = [for i in range(0,2):
```

- Deploy: az deployment group create -g rg-lb --template-file main.bicep --parameters adminUsername=... adminPassword=...

9. Coexisting External & Internal LBs

- Use-cases:
    - External LB accepts internet requests.

    - Internal LB serves internal-only clients or AKS pods.

- Architecture:
    - Both LBs point to same backend VM NICs.
    - NSGs permit traffic from both sessions.
- Management:
    - Monitor separate metrics, adjust probes accordingly.
    - Ensure private and public frontend IPs don't overlap.

10. Maintenance & Cleanup

- **Scale backends**: Add/remove VM NICs from backend pool via portal/API.
- **Tune health probes**: adjust intervals, thresholds based on app behavior.
- **Upgrade SKU**: Basic → Standard requires redeploy.
- **Clean up**: az group delete -n rg-lb or delete via portal.