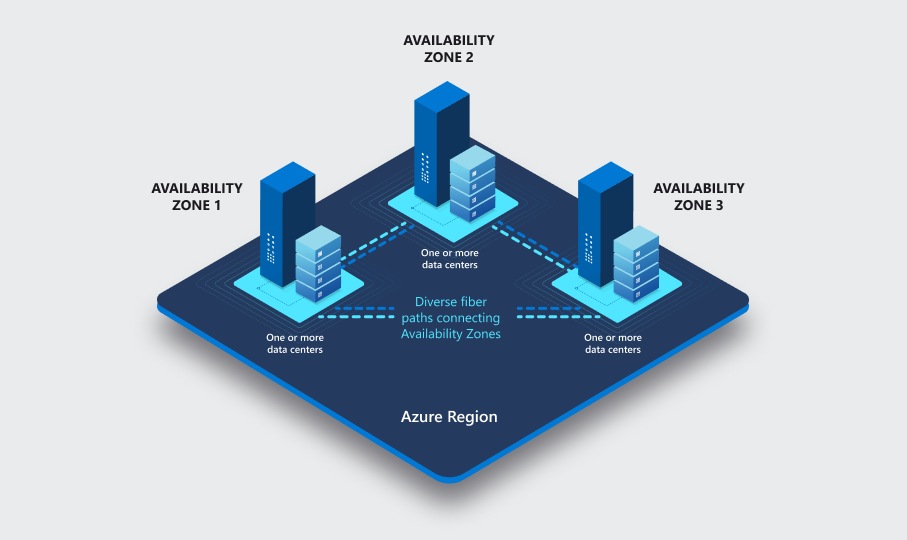
**1) Availability Zones in Azure**

* **Definition:** Availability Zones are physically separate datacentre’s within an Azure region, designed to enhance high availability and fault tolerance.
* **Key Features:**
  + Independent power, cooling, and networking for each zone.
  + Protection against datacenter-level failures.
* **Use Cases:**
  + High-availability applications.
  + Disaster recovery and business continuity.
  + Distributed workloads for improved reliability.
* **Example Services:**
  + Virtual Machines (VMs) can be distributed across zones.
  + Zone-redundant storage ensures data durability.
  + Load balancers can distribute traffic across zones.

By understanding these concepts and carefully planning resources, you can design scalable, reliable, and efficient cloud solutions.



**2) Availability Set**

An **Availability Set** is a logical grouping of Virtual Machines (VMs) designed to provide redundancy and high availability. By deploying your VMs into an availability set, you ensure that your application remains operational during planned maintenance or unplanned hardware failures.

**Practice Steps:**

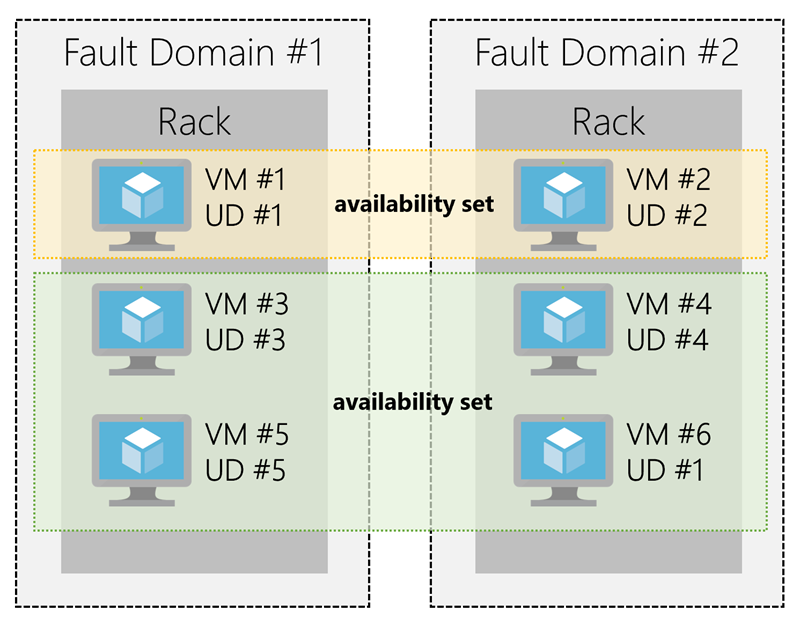
1. Navigate to the Azure portal.
2. Create a new Availability Set:
   * Go to **Create a resource** > **Compute** > **Availability Set**.
   * Specify the resource group, name, and region.
   * Configure fault domains and update domains (discussed below).
3. Deploy multiple VMs into this availability set:
   * While creating each VM, select the availability set you created.

**3) Fault Domain**

A **Fault Domain** represents a grouping of hardware within a data center that shares a common power source and network switch. Fault domains help isolate VMs to protect against hardware failures.

**Practice Steps:**

1. During the Availability Set creation:
   * Set the number of fault domains (typically 2 or 3).
   * Azure will automatically distribute your VMs across these fault domains.
2. Validate fault domain allocation:
   * After deploying VMs, navigate to the availability set.
   * Check the **Fault Domain** column under the **Instances** tab.

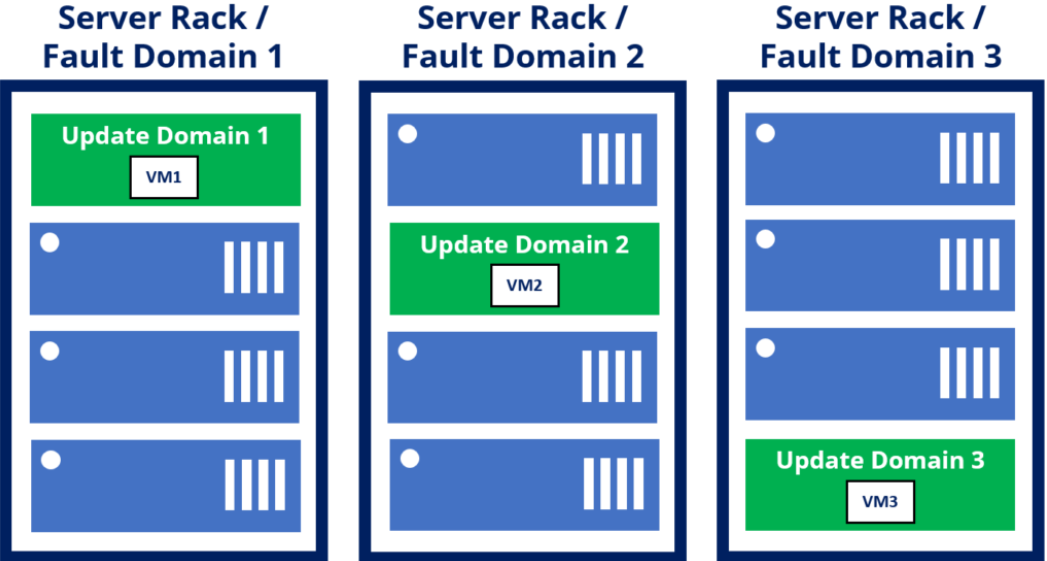


**4) Update Domain**

An **Update Domain** defines a group of VMs and associated hardware that can be updated or rebooted simultaneously during planned maintenance events. Azure ensures updates are rolled out domain by domain to maintain application uptime.

**Practice Steps:**

1. During the Availability Set creation:
   * Set the number of update domains (default is 5).
2. Validate update domain allocation:
   * After deploying VMs, navigate to the availability set.
   * Check the **Update Domain** column under the **Instances** tab.
3. Simulate an update:
   * Use Azure tools to simulate maintenance events or observe behavior during Azure-scheduled updates.



**Summary of Key Concepts**

* **Availability Set**: Logical grouping of VMs to provide high availability.
* **Fault Domain**: Ensures VMs are distributed across separate physical hardware to prevent single points of failure.
* **Update Domain**: Staggers updates to maintain application availability during maintenance events.

**5) Azure Load Balancer and Load Balancing Algorithms**

This document provides a comprehensive practice guide for setting up and exploring Azure Load Balancer and different load-balancing algorithms without exposing public IPs or ports. The aim is to create a secure and private environment for testing and learning.

**Prerequisites**

* Azure account with required permissions.
* Basic knowledge of virtual networks (VNets), virtual machines (VMs), and Azure resources.
* Azure CLI or Azure PowerShell installed (optional for automation).

**Lab Setup Overview**

1. **Create a Virtual Network**
   * Define a private VNet with appropriate subnets for backend resources.
   * Ensure the VNet is isolated without a public-facing endpoint.
2. **Provision Backend Virtual Machines**
   * Create at least two VMs within the VNet for simulating backend services.
   * Configure internal-only communication by assigning private IP addresses.
3. **Set Up Azure Load Balancer**
   * Deploy an Azure Load Balancer in the same VNet.
   * Use the private IP address for internal load balancing.
4. **Configure Load Balancer Rules**
   * Define load-balancing rules based on internal ports.
   * Associate the backend pool with VMs.
5. **Experiment with Load-Balancing Algorithms**
   * Understand and apply different algorithms like Round Robin, Least Connections, and Hash-based distribution.
6. **Validate Traffic Distribution**
   * Use network testing tools (e.g., curl, ApacheBench) to simulate and analyze traffic.

**Step-by-Step Instructions**

**Step 1: Create a Virtual Network**

1. **Navigate to Azure Portal** and create a new Virtual Network.
2. Define the VNet name, address space, and subnet configurations.
3. Ensure no public IP is associated with the VNet or subnets.

**Step 2: Provision Backend Virtual Machines**

1. In the Azure portal, create two or more VMs within the VNet.
2. Select private IP configuration for the network interface.
3. Install a lightweight web server (e.g., Nginx or Apache) on each VM to serve as backend services.

**Step 3: Deploy an Azure Load Balancer**

1. Go to the Azure Portal and search for "Load Balancer."
2. Choose "Internal Load Balancer" during the setup process.
3. Assign a private IP from the VNet.

**Step 4: Configure Load Balancer Backend Pool**

1. Add the backend VMs to the backend pool.
2. Ensure all VMs use the same internal port for communication (e.g., port 80 or 443).

**Step 5: Define Load-Balancing Rules**

1. Navigate to the "Rules" section of the Load Balancer.
2. Create a load-balancing rule for traffic distribution within the private network.
3. Configure the health probe to monitor VM availability.

**Step 6: Experiment with Load-Balancing Algorithms**

Azure Load Balancer supports the following algorithms:

1. **Round Robin**: Distributes requests evenly across all VMs in the backend pool.
   * Test by sending multiple requests and observing even distribution.
2. **Least Connections**: Routes requests to the VM with the fewest active connections.
   * Simulate active connections and analyze request handling.
3. **Source IP Hash**: Uses the client’s IP to determine the backend server.
   * Test by generating requests from different client IPs.

To switch between algorithms, update the load balancer settings in the Azure Portal.

**Step 7: Validate Traffic Distribution**

1. Log into one of the backend VMs and observe incoming requests.
2. Use network tools to simulate multiple requests:
3. curl http://<load-balancer-private-ip>

ab -n 100 -c 10 http://<load-balancer-private-ip>/

1. Analyze logs and ensure the algorithm works as expected.

**Clean Up Resources**

After completing the practice, clean up resources to avoid unnecessary costs:

1. Delete the Load Balancer.
2. Remove the VMs and VNet.
3. Verify that all dependent resources are deleted.

**Conclusion**

This guide provides a secure and private setup to practice using Azure Load Balancer and experimenting with load-balancing algorithms. By following these steps, you will gain hands-on experience with configuring and validating load-balancing solutions in Azure.

Note- No public IP is needed, no port needed when we are using load balancer concept for VM.