**Challenge**

**Create Solution diagram** with set of proven practices for running virtual machines (VMs) for an application with a N-tier architecture. 3-tier web app include following tiers.

* **Web tier.** Handles incoming HTTP requests. Responses are returned through this tier.
* **Business tier.** Implements business processes and other functional logic for the system.
* **Data tier.** Provides persistent data storage.

Architecture Diagram should include following components.

* **Availability Sets.** Create an Availability Set for each tier, and provision at least two VMs in each tier. This approach is required to reach the availability SLA for VMs.
* **Subnets.** Create a separate subnet for each tier. Specify the address range and subnet mask using CIDR notation.
* **Load balancers.** Use an Internet-facing load balancer to distribute incoming Internet traffic to the web tier, and an internal load balancer to distribute network traffic from the web tier to the business tier.
* **Jumpbox**. A jumpbox, also called a bastion host, is a VM on the network that administrators use to connect to the other VMs. The jumpbox has an NSG that allows remote traffic only from whitelisted public IP addresses. The NSG should permit remote desktop (RDP) traffic if the jumpbox is a Windows VM, or secure shell (SSH) requests if the jumpbox is a Linux VM.
* **Monitoring**. Monitoring software such as Nagios, Zabbix, or Icinga can give your insight into response time, VM uptime, and the overall health of your system. Install the monitoring software on a VM that is placed in a separate management subnet.
* **NSGs**. Use network security groups (NSGs) to restrict network traffic within the VNet. For example, in the 3-tier architecture shown here, the data tier does not accept traffic from the web front end, only from the business tier and the management subnet.
* **Key Vault**. Use Azure Key Vault to manage encryption keys, for encrypting data at rest.

**Recommendations**

**VNet / Subnets**

* When you create the VNet, allocate enough address space for the subnets you will need. Specify the address range and subnet mask using CIDR notation. Use an address space that falls within the standard private IP address blocks, which are 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16.
* Do not pick an address range that overlaps with your on-premises network, in case you need to set up a gateway between the VNet and your on-premises network later. Once you create the VNet, you cannot change the address range.
* Design subnets with functionality and security requirements in mind. All VMs within the same tier or role should go into the same subnet, which can be a security boundary. Specify the address space for the subnet in CIDR notation.

**Load balancers**

* The external load balancer distributes Internet traffic to the web tier. Create a public IP address for this load balancer.
* The internal load balancer distributes network traffic from the web tier to the business tier. To give this load balancer a private IP address, create a frontend IP configuration and associate it with the subnet for the business tier.

**Jumpbox**

* Place the jumpbox in the same VNet as the other VMs, but in a separate management subnet.
* Create a public IP address for the jumpbox.
* Use a small VM size for the jumpbox, such as Standard A1.
* Configure the NSGs for the web tier, business tier, and database tier subnets to allow administrative (RDP/SSH) traffic to pass through from the management subnet.
* To secure the jumpbox, create an NSG and apply it to the jumpbox subnet. Add an NSG rule that allows RDP or SSH connections only from a whitelisted set of public IP addresses.

The NSG can be attached either to the subnet or to the jumpbox NIC. In this case, we recommend attaching it to the NIC, so RDP/SSH traffic is permitted only to the jumpbox, even if you add other VMs to the same subnet.

**Availability considerations**

* Put each tier or VM role into a separate availability set. Don't put VMs from different tiers into the same availability set.
* At the data tier, having multiple VMs does not automatically translate into a highly available database. For a relational database, you will typically need to use replication and failover to achieve high availability. The business tier will connect to a primary database, and if that VM goes down, the application fails over to a secondary database, either manually or automatically.

**Security considerations**

* Encrypt data at rest. Use Azure Key Vault to manage the database encryption keys. Key Vault can store encryption keys in hardware security modules (HSMs).
* Do not allow RDP/SSH access from the public Internet to the VMs that run the application workload. Instead, all RDP/SSH access to these VMs must come through the jumpbox. An administrator logs into the jumpbox, and then logs into the other VM from the jumpbox. The jumpbox allows RDP/SSH traffic from the Internet, but only from known, whitelisted IP addresses.
* Use NSG rules to restrict traffic between tiers. For example, in the 3-tier architecture the web tier does not communicate directly with the data tier. To enforce this, the data tier should block incoming traffic from the web tier subnet.

**Scalability considerations**

The load balancers distribute network traffic to the web and business tiers. Scale horizontally by adding new VM instances. Note that you can scale the web and business tiers independently, based on load. To reduce possible complications caused by the need to maintain client affinity, the VMs in the web tier should be stateless. The VMs hosting the business logic should also be stateless.

**Add reliability to an N-tier architecture on Azure.**

* A network virtual appliance for greater network security.
* SQL Server AlwaysOn Availability Groups for high availability in the data tier.
* **Network virtual appliance (NVA)**. A VM running software that performs network security functionality. Typical features provided by an NVA include:
  + Firewall.
  + Traffic optimization, such as WAN optimization.
  + Packet inspection.
  + SSL offloading.
  + Layer 7 load balancing.
  + Logging and reporting.
* **SQL Server AlwaysOn Availability Group**. Provides high availability at the data tier, by enabling replication and failover.

The application consists of two services, labelled A and B. For example, they might be web apps or web APIs. Client requests are routed either to service A or to service B, depending on the content of the request (for example, the URL path). Service A writes to a SQL database. Service B sends data to an external service, such as Redis cache or a message queue, which is outside the scope of this problem statement.

These general characteristics imply some high-level requirements for the system:

* Intelligent load balancing, to route requests based on URLs or message content. (Layer-7 load balancing.)
* Logging and monitoring of network traffic.
* Network packet inspection.
* Multiple storage technologies might be used.