**Research Paper: Implementation of a 3-Tier Application Architecture in Azure**

**Abstract**

This paper presents a detailed approach to implementing a 3-tier application architecture in Microsoft Azure, leveraging virtual networks (VNets), subnets, virtual machines (VMs), and Network Security Groups (NSGs). The architecture comprises three tiers: Web, App, and DB, each within separate VNets to ensure isolation and security. This paper covers the step-by-step process of provisioning resources, configuring network and security settings, and adhering to specified conditions to ensure restricted internet access and controlled connectivity between tiers.

**Introduction**

The 3-tier application architecture is a well-established design pattern that separates application concerns into three layers: the presentation layer (Web), the business logic layer (App), and the data access layer (DB). This separation enhances modularity, scalability, and maintainability. Implementing this architecture in a cloud environment such as Microsoft Azure involves careful planning and execution to meet specific requirements, including restricted internet access and controlled inter-tier communication.

**System Design**

**1. Architecture Overview**

The architecture consists of three tiers:

* **Web Tier:** Hosts the presentation layer with an IIS Server (Windows) and an Apache Server (Linux).
* **App Tier:** Hosts the backend API server that handles business logic.
* **DB Tier:** Hosts the SQL Database that stores data.

**2. Network Design**

To ensure isolation and security:

* **Web Tier VNet:** WebTierVNet with subnet WebSubnet.
* **App Tier VNet:** AppTierVNet with subnet AppSubnet.
* **DB Tier VNet:** DBTierVNet with subnet DBSubnet.

**3. Security Requirements**

The following security requirements were specified:

1. **No outbound internet access** for any tier after initial setup.
2. **Inbound internet access** allowed only for Web and App tiers.
3. **No private connectivity** allowed between tiers without necessity.
4. **Individual NSGs** associated with subnets for detailed traffic control.

**Implementation**

**Step 1: Setup Virtual Networks and Subnets**

1. **Creation of VNets and Subnets:**
   * Create WebTierVNet, AppTierVNet, and DBTierVNet with their respective subnets: WebSubnet, AppSubnet, and DBSubnet.

**Step 2: Provision Virtual Machines**

1. **Web Tier:**
   * **Windows IIS Server:** Deployed on a Windows VM in WebSubnet.
   * **Apache Linux Server:** Deployed on a Linux VM in WebSubnet.
2. **App Tier:**
   * **Backend API Server:** Deployed on a VM in AppSubnet.
3. **DB Tier:**
   * **SQL Database Server:** Deployed on a VM in DBSubnet.

**Step 3: Configure Network Security Groups (NSGs)**

1. **NSGs Creation and Association:**
   * **WebSubnetNSG:** Applied to WebSubnet.
   * **AppSubnetNSG:** Applied to AppSubnet.
   * **DBSubnetNSG:** Applied to DBSubnet.
2. **NSG Rules Configuration:**
   * **WebSubnetNSG:**
     + Allow inbound HTTP/HTTPS.
     + Deny all outbound internet traffic.
   * **AppSubnetNSG:**
     + Allow inbound traffic from Web Tier.
     + Deny all outbound internet traffic.
   * **DBSubnetNSG:**
     + Allow inbound traffic from App Tier.
     + Deny all outbound internet traffic.

**Step 4: Configure VNet Peering**

1. **VNet Peering Configuration:**
   * **WebTierVNet to AppTierVNet:** Allowed if necessary.
   * **AppTierVNet to DBTierVNet:** Allowed if necessary.
2. **Peering Settings:**
   * Enabled traffic flow between peered VNets with necessary settings.

**Step 5: Deploy and Configure Applications**

1. **Web Tier:**
   * Deployed IIS website with a button to trigger API calls.
   * Configured Apache server for static content or relevant web application components.
2. **App Tier:**
   * Deployed backend API and verified interaction with the SQL Database.
3. **DB Tier:**
   * Configured SQL Database and verified connectivity from the App Tier.

**Step 6: Restrict Outbound Internet Access**

1. **NSG Rules Verification:**
   * Ensured that all tiers adhered to the no outbound internet access policy after installation.

**Step 7: Testing**

1. **Functionality Testing:**
   * Verified that the IIS server hosted the website correctly.
   * Tested the Apache server for expected operations.
   * Confirmed the button on the website triggered the API call.
   * Validated that the backend API logs data correctly in the SQL Database.

**Step 8: Monitoring and Maintenance**

1. **Setup Monitoring:**
   * Used Azure Monitor and Application Insights for performance and health monitoring.
2. **Security and Configuration Review:**
   * Regularly reviewed NSG rules and VNet peering configurations for compliance with security policies.

**Results**

The implementation met all specified conditions, with successful deployment and configuration of the 3-tier architecture. The security policies were enforced, and the application functions as intended with proper isolation and controlled connectivity.

**Conclusion**

The project demonstrates the effective use of Azure resources to implement a secure and scalable 3-tier application architecture. By adhering to strict network and security requirements, the solution provides a robust framework for deploying complex applications in a cloud environment. Future work could explore automation and optimization strategies to further enhance efficiency and manageability.

**References**

* Microsoft Azure Documentation
* Azure Virtual Network and Subnet Configuration
* Network Security Groups (NSGs) Best Practices
* Azure VNet Peering and Security Configurations

This research paper provides a comprehensive overview of the project, from design to implementation, ensuring that all requirements and conditions are addressed. Let me know if there are any specific details or additional sections you'd like to include!