**3 tier arc:-**

***To implement a \*\*3-tier architecture\*\* in Azure Cloud, you can follow these steps:***

**### 1. \*\*Define Your Architecture\*\***

- Identify the components of each tier:

- \*\*Presentation Tier\*\*: Web server (e.g., Azure App Service, Azure Static Web Apps).

- \*\*Application Tier\*\*: Backend services (e.g., Azure Virtual Machines, Azure App Service, Azure Kubernetes Service).

- \*\*Data Tier\*\*: Database (e.g., Azure SQL Database, Azure Cosmos DB).

**### 2. \*\*Set Up Azure Resources\*\***

- \*\*Create Resource Groups\*\*: Organize resources related to your application.

- \*\*Provision Virtual Machines or App Services\*\*:

- For the \*\*Web Tier\*\*, create an Azure App Service or Virtual Machine.

- For the \*\*Business Logic Tier\*\*, create another Azure App Service or VM.

- For the \*\*Data Tier\*\*, set up a database service like Azure SQL Database.

- \*\*Networking\*\*:

- Set up a \*\*Virtual Network\*\* (VNet) to host the resources.

- Create subnets for each tier to isolate traffic.

**### 3. \*\*Configure Security\*\***

- \*\*Network Security Groups (NSGs)\*\*: Define rules to restrict access between tiers (e.g., only allow traffic from the web tier to the application tier).

- \*\*Application Gateway / Web Application Firewall (WAF)\*\*: Place this in front of the web tier for enhanced security and traffic management.

**### 4. \*\*Deploy Applications\*\***

- \*\*Deploy Frontend\*\*: Upload your web application to the Azure App Service or configure your web server on the VM.

- \*\*Deploy Backend\*\*: Deploy your application logic to the application tier using services like Azure App Service, or configure your business logic on a VM.

- \*\*Database Configuration\*\*: Set up your database schema, connection strings, and ensure proper access configurations.

**### 5. \*\*Connect the Tiers\*\***

- Ensure that your application tier can communicate with your data tier. Set up connection strings in the application tier to point to the database.

- Implement API endpoints in the application tier that the web tier can call.

### 6. \*\*Testing and Monitoring\*\*

- \*\*Testing\*\*: Test the application flow from the presentation tier to the application tier and to the data tier.

- \*\*Monitoring\*\*: Set up Azure Monitor and Application Insights to monitor the health and performance of your application.

### 7. \*\*Scale and Optimize\*\*

- \*\*Autoscaling\*\*: Set up autoscaling for your App Services or VMs based on traffic.

- \*\*Caching\*\*: Consider using Azure Cache for Redis to reduce latency and improve performance for frequently accessed data.

### 8. \*\*Deployment Automation\*\*

- Use \*\*Azure DevOps\*\* or \*\*GitHub Actions\*\* to automate the deployment process, ensuring consistent and repeatable deployments.

### 9. \*\*Backup and Disaster Recovery\*\*

- Implement backup strategies for your data tier and configure recovery plans to handle outages.

Following this flow will help you set up a robust 3-tier architecture in Azure Cloud, ensuring proper separation of concerns, scalability, and security.

***It sounds like you're setting up a \*\*multi-tier architecture\*\* in Azure with both public and private subnets. Here's how it would typically work:***

### \*\*1. Network Setup:\*\*

- \*\*Public Subnets (2)\*\*:

- These subnets will have internet access, which means the virtual machines (VMs) in these subnets will have public IP addresses.

- \*\*Example Use Case\*\*: VMs hosting your \*\*web servers\*\* are placed here, allowing users to connect to them via the internet.

- \*\*Private Subnets (2)\*\*:

- These subnets won't have direct internet access, and VMs in these subnets will only communicate with other VMs within the virtual network (VNet).

- \*\*Example Use Case\*\*: VMs hosting your \*\*application layer\*\* (app server) and \*\*database\*\* will be placed here. Only the VMs in the public subnets (like your web servers) can communicate with them.

### \*\*2. Connectivity Flow\*\*:

- \*\*Web VMs in Public Subnet\*\*:

- These VMs have \*\*public IPs\*\* and can be accessed by users via the internet.

- They will connect to the \*\*application VMs\*\* in the private subnet via \*\*private IP addresses\*\*.

- \*\*Application VMs in Private Subnet\*\*:

- These VMs do not have public IPs and are not directly accessible from the internet.

- They can be accessed only by the \*\*web VMs\*\* from the public subnet, ensuring a secure flow of traffic from the public web tier to the private app tier.

- \*\*Database VMs in Private Subnet\*\*:

- Similarly, your \*\*database\*\* VMs are also in the private subnet and can only be accessed by the \*\*application VMs\*\*, which ensures additional security for your database layer.

---

### \*\*Challenges and Key Considerations\*\*:

- \*\*NAT Gateway for Private VMs\*\*:

- If your \*\*application\*\* or \*\*database\*\* VMs need to access the internet (for updates, etc.), you'll need to set up a \*\*NAT Gateway\*\* or use a \*\*jump box\*\* VM (bastion) for controlled access.

- \*\*Network Security Groups (NSGs)\*\*:

- For each subnet, you'll want to use \*\*NSGs\*\* to control inbound and outbound traffic.

- Public Subnets: Open \*\*HTTP/HTTPS\*\* for web access and allow traffic to flow to the private subnets.

- Private Subnets: Restrict access to only allow traffic from the \*\*public subnet\*\* and specific ports (like HTTP for app servers, SQL for databases).

- \*\*Routing\*\*:

- You'll need to configure routing so that traffic from the \*\*web servers\*\* (public subnet) can reach the \*\*app servers\*\* and \*\*databases\*\* (private subnets).

---

### \*\*Example Flow\*\*:

1. \*\*Users\*\* -> access \*\*web servers\*\* (Public Subnet, via Public IP).

2. \*\*Web servers\*\* -> connect to \*\*app servers\*\* (Private Subnet, via Private IP).

3. \*\*App servers\*\* -> connect to \*\*database servers\*\* (Private Subnet, via Private IP).

This ensures that:

- The web tier is publicly accessible.

- The app and database tiers are protected and only accessible from within the private network.

Let me know if you need help with any specific configuration!

***It seems like you have created four VMs, and all of them have both \*\*public and private IP addresses\*\*. Since you are setting up a combination of \*\*public\*\* and \*\*private\*\* subnets, some of the VMs should only have \*\*private IP addresses\*\* (for the application and database VMs) and should not be exposed to the internet via public IP addresses.***

Here’s how to correct this setup:

### \*\*Steps to Adjust Your VM Setup\*\*:

#### \*\*1. Remove Public IP from VMs in Private Subnets:\*\*

- \*\*Why\*\*: VMs in the \*\*private subnets\*\* (application and database tiers) should not have public IPs for security reasons, as they don’t need direct internet access.

- \*\*How to Remove Public IP\*\*:

1. Go to the \*\*Azure Portal\*\*.

2. Navigate to the \*\*Virtual Machines\*\* section.

3. Select the VM that’s supposed to be in a \*\*private subnet\*\* (like your app or database VM).

4. Under \*\*Networking\*\*, find the \*\*public IP\*\* attached to the VM.

5. \*\*Disassociate\*\* the public IP or delete it if you don't need it.

This will keep the VM within the private subnet and allow it to communicate only internally (via private IP).

#### \*\*2. Keep Public IPs Only for VMs in Public Subnets:\*\*

- \*\*Why\*\*: VMs in \*\*public subnets\*\* (like the web servers) need to be accessible from the internet and, therefore, should have public IP addresses.

Make sure that only the \*\*web servers\*\* have public IPs.

#### \*\*3. Ensure Proper NSG (Network Security Group) Rules\*\*:

- For the VMs in public subnets (with public IPs):

- Allow \*\*inbound HTTP/HTTPS\*\* traffic (for web access).

- Restrict traffic to allow only \*\*outbound connections\*\* to the private subnet for connecting to the app and database servers.

- For the VMs in private subnets (with private IPs):

- Ensure that inbound connections are only allowed from the public subnet (web tier).

- Block all public inbound traffic.

#### \*\*4. Optional: Use a NAT Gateway for Internet Access (Private Subnets)\*\*:

- If you need the VMs in the private subnets to access the internet for updates (but without exposing them to the public), you can configure a \*\*NAT Gateway\*\*.

This allows VMs in private subnets to initiate outbound traffic to the internet without having public IP addresses.

---

### \*\*Example Configuration\*\*:

1. \*\*Web VM\*\* in \*\*Public Subnet\*\*:

- Has both \*\*private IP\*\* (internal communication) and \*\*public IP\*\* (internet-facing access).

- Users access the web application using the \*\*public IP\*\* or domain name.

2. \*\*App VM\*\* in \*\*Private Subnet\*\*:

- Has only a \*\*private IP\*\* (for internal communication).

- Only accessible by the \*\*web VM\*\* in the public subnet.

3. \*\*Database VM\*\* in \*\*Private Subnet\*\*:

- Has only a \*\*private IP\*\*.

- Only accessible by the \*\*app VM\*\*.

---

### \*\*Final Checklist\*\*:

- Web VMs: \*\*Public and Private IP\*\* (public subnet).

- Application and Database VMs: \*\*Private IP only\*\* (private subnet).

- Use \*\*NSGs\*\* to limit communication between tiers.

- If necessary, add a \*\*NAT Gateway\*\* for private VMs to access the internet without public exposure.

Let me know if you need help adjusting these settings or setting up the NAT Gateway!

***To remove the \*\*public IP address\*\* from a Virtual Machine (VM) in Azure, follow these steps:***

### \*\*Step-by-Step Guide to Remove Public IP from a VM\*\*:

#### \*\*1. Go to the Azure Portal:\*\*

- Open your browser and navigate to the [Azure Portal](https://portal.azure.com).

#### \*\*2. Navigate to the Virtual Machine:\*\*

- In the left-hand menu, select \*\*Virtual Machines\*\*.

- From the list of VMs, select the \*\*VM\*\* from which you want to remove the public IP address.

#### \*\*3. Go to Networking Settings:\*\*

- Once the VM is selected, on the VM's \*\*Overview\*\* page, click on \*\*Networking\*\* from the left-side menu (under \*\*Settings\*\*).

#### \*\*4. Locate the Public IP:\*\*

- In the \*\*Networking\*\* tab, you'll see the \*\*Network Interface\*\* that’s attached to the VM.

- Under the \*\*Public IP Addresses\*\* section, you will see the currently assigned \*\*Public IP\*\*.

#### \*\*5. Disassociate the Public IP:\*\*

- Click on the \*\*Public IP\*\* address link to open the public IP configuration.

- At the top, you’ll see an option to \*\*Disassociate\*\* the public IP.

- Click \*\*Disassociate\*\* to remove the public IP from the network interface.

#### \*\*6. Confirm the Changes:\*\*

- Once you've disassociated the public IP, the VM will no longer have a direct public IP address.

- Click \*\*Save\*\* to apply the changes.

#### \*\*7. Verify the VM:\*\*

- Go back to the \*\*Networking\*\* tab of the VM to verify that no public IP is listed.

- You can also check the \*\*Overview\*\* page of the VM to ensure that the public IP is no longer associated.

### \*\*Optional - Delete the Public IP Resource (if no longer needed)\*\*:

- If you no longer need the public IP at all (for example, if it's not going to be used by any other resources), you can delete it:

- Go to \*\*Public IP Addresses\*\* in the Azure Portal.

- Find the \*\*public IP\*\* you disassociated and delete it to free up resources.

---

### \*\*Notes\*\*:

- After removing the public IP, the VM will only be accessible using its \*\*private IP\*\* (for internal communication within the Virtual Network).

- Ensure that any necessary inbound/outbound connections are configured appropriately, especially if you use \*\*Bastion\*\*, \*\*VPN\*\*, or \*\*NAT Gateway\*\* for private VMs.

Let me know if you need help with any specific steps or additional configuration!

*Once you’ve properly configured your infrastructure with \*\*2 public subnets\*\* and \*\*2 private subnets\*\*, and ensured that the \*\*correct VMs\*\* are placed in the right subnets (public VMs in public subnets, private VMs in private subnets), your infrastructure should be almost ready. However, there are a few additional key elements to check to ensure everything is fully operational:*

### \*\*Checklist to Ensure Your Infrastructure is Ready\*\*:

#### \*\*1. Subnet and VM Configuration:\*\*

- \*\*2 Public VMs\*\* (in the public subnets) should have \*\*public and private IPs\*\* for external access and internal communication.

- \*\*2 Private VMs\*\* (in the private subnets) should have \*\*only private IPs\*\* for internal communication.

#### \*\*2. Network Security Groups (NSGs):\*\*

- \*\*Public VMs (Web Tier)\*\*:

- NSG rules should allow \*\*inbound HTTP/HTTPS (port 80/443)\*\* for web traffic.

- Allow \*\*SSH (port 22)\*\* or \*\*RDP (port 3389)\*\* access from specific IPs (e.g., your office or home network) for management.

- NSG rules should allow outbound traffic to the \*\*private VMs\*\* in the private subnet (application and database VMs).

- \*\*Private VMs (Application and Database Tier)\*\*:

- NSG rules should only allow \*\*inbound traffic from the public VMs\*\* (to allow communication between the web and application/database tiers).

- Block all other inbound traffic from the internet.

- Allow outbound traffic for updates or internal communication (use \*\*NAT Gateway\*\* for internet access, if required).

#### \*\*3. Route Tables (if needed):\*\*

- \*\*Public Subnets\*\*:

- The default route should allow traffic to the internet (this is usually handled automatically by Azure for public subnets).

- \*\*Private Subnets\*\*:

- Ensure routing rules are configured to allow communication with the \*\*public subnets\*\* for internal communication (web tier to app tier, app tier to database).

- Optionally, use a \*\*NAT Gateway\*\* to enable outbound internet access from the private VMs without exposing them directly to the internet.

#### \*\*4. Communication Between the Tiers:\*\*

- Ensure that the \*\*Web Tier (public VMs)\*\* can communicate with the \*\*App Tier\*\* (private VMs) and the \*\*Database Tier\*\* (private VMs).

- Check that the web application can reach the app server, and the app server can reach the database.

#### \*\*5. Load Balancer (if applicable):\*\*

- If you are using multiple VMs in the web tier or app tier, consider placing them behind a \*\*Load Balancer\*\* to distribute traffic evenly.

- \*\*Public Load Balancer\*\* for the web tier to handle traffic from users.

- \*\*Internal Load Balancer\*\* for the app tier if you have multiple app VMs.

#### \*\*6. DNS (if applicable):\*\*

- Configure a \*\*DNS name\*\* or \*\*custom domain\*\* for your web application (if needed), so users can access it using a domain name rather than the public IP.

#### \*\*7. Monitoring & Alerts:\*\*

- Set up \*\*monitoring\*\* for your VMs and services using \*\*Azure Monitor\*\* to track performance, uptime, and potential issues.

- Configure \*\*alerts\*\* for critical events such as high CPU usage, low memory, or VM downtime.

#### \*\*8. Testing Your Infrastructure:\*\*

- \*\*Access the Public VMs (Web Tier)\*\*: Test accessing the web VMs from a browser using the public IP or domain name to ensure they are reachable.

- \*\*Internal Communication\*\*: Test that the web VMs can connect to the app VMs, and app VMs can connect to the database.

- \*\*No Direct Access to Private VMs\*\*: Verify that the private VMs (app and database) are \*\*not directly accessible from the internet\*\*.

---

### \*\*Summary of Steps to Confirm Readiness\*\*:

1. \*\*Correct Subnet Assignment\*\*: Ensure 2 VMs are in public subnets and 2 in private subnets.

2. \*\*NSG Rules\*\*: Apply proper network security rules to control access and communication.

3. \*\*Private IP Only for Private Subnets\*\*: Private VMs should only have private IP addresses.

4. \*\*Load Balancer & DNS (optional)\*\*: If scaling, use a load balancer; configure DNS if needed.

5. \*\*Test Connectivity\*\*: Verify public access to the web VMs and internal communication with app and database tiers.

6. \*\*Monitor\*\*: Set up monitoring to keep track of performance and alerts for issues.

Once you've verified all the above steps, your infrastructure should be ready for production or further testing.

Let me know if you need help with any specific part of the setup!

**From public VM to access private VM :**

Azure Bastion: Best for securely accessing private VMs directly from the Azure Portal without public IP.

Jump/Bastion Host: Use one of the public VMs to connect to private VMs via their private IP.

VPN: Set up a VPN to connect your local machine to the Azure Virtual Network, then use private IPs to access the private VMs.

NAT Gateway: For outbound-only internet access from private VMs.

***Congratulations on completing your project! Here's an overview of the flow of your project, highlighting how the request flows and important terms involved:***

### 1. \*\*Client (Browser)\*\*:

- A user initiates a request by visiting the public IP or domain name of the project, e.g., `http://13.70.65.137/`.

- The request (HTTP GET) is sent to the public IP of \*\*VM2\*\*, which is running the \*\*Nginx web server\*\*.

### 2. \*\*Nginx (Reverse Proxy)\*\*:

- \*\*Nginx\*\* is acting as a \*\*reverse proxy\*\* on \*\*VM2\*\*. When it receives the request, it checks its configuration.

- The `server\_name` and `location` blocks in the Nginx configuration determine how to handle the request.

- For static files like CSS or images, Nginx directly serves them from the `/static/` directory on \*\*VM2\*\*.

- For dynamic content (e.g., handling user interactions), Nginx forwards the request to \*\*VM3\*\*.

### 3. \*\*Proxy Pass to Django App (VM3)\*\*:

- The Nginx configuration has a `proxy\_pass` directive, which forwards the dynamic requests to \*\*VM3\*\* at `http://10.0.3.4:8000`, where the \*\*Django application\*\* is running.

- Nginx also forwards additional headers like the real IP address and the protocol being used (HTTP or HTTPS).

### 4. \*\*Django Application (VM3)\*\*:

- \*\*Django\*\* processes the request based on the defined routes (URLs) and views. For example, it might fetch data from the database, process forms, or render templates.

- After processing, Django returns the generated HTML response (or JSON, depending on the request type) to Nginx.

### 5. \*\*MySQL Database (VM4)\*\*:

- If the Django application requires data, it interacts with \*\*MySQL\*\* hosted on \*\*VM4\*\* (with private IP `10.0.4.4`).

- Django connects to MySQL using the private network. It uses SQL queries to retrieve or update the data.

- After getting the necessary data, Django processes it and sends the final response back to Nginx.

### 6. \*\*Response to Client (Browser)\*\*:

- Once Nginx receives the response from Django, it sends the final HTML or content back to the user's browser.

- The browser renders the page, and the user can interact with the application.

---

### Key Terms:

- \*\*Nginx\*\*: A web server that can also act as a reverse proxy, forwarding client requests to backend servers like Django.

- \*\*Reverse Proxy\*\*: A server (Nginx) that forwards requests to another server (Django) and then sends the response back to the client.

- \*\*Proxy Pass\*\*: A directive in Nginx that forwards requests to another server.

- \*\*Django\*\*: A Python web framework that handles the backend logic of your application.

- \*\*Static Files\*\*: Non-dynamic content like CSS, JavaScript, or images served by Nginx.

- \*\*Gunicorn\*\*: A WSGI HTTP server for running Python web applications like Django. It's often used with Nginx.

- \*\*MySQL\*\*: The relational database where Django stores application data.

### Flow Summary:

1. \*\*Browser\*\* ➔ \*\*Nginx\*\* (Public VM2) ➔ \*\*Django\*\* (Private VM3) ➔ \*\*MySQL\*\* (Private VM4) ➔ \*\*Django\*\* ➔ \*\*Nginx\*\* ➔ \*\*Browser\*\*.

This flow ensures separation of concerns: Nginx handles web traffic, Django handles application logic, and MySQL manages the database.

Let me know if you'd like to dive into any part of this flow further!