R&D Document: 3-Tier Architecture

1. Title:

Creating a 3-Tier Architecture in AWS with Internet Access Control and Security Rules. (Create three subnets : 1. Web tier 2. App tier 3. DB tier DB Tier should not access any tier(Web & App tier) App tier should access the DB tier and Web tier as well, Web tier should acccess only App tier. Only Web tier is allowed to connect to the internet.Deploy two VM's in each tier(One VM should be Linux & another should be Windows). Configure Apache Server on Linux VM's And IIS Server on Windows.)

2. Introduction:

A 3-tier architecture divides an application into three parts: the Web tier (frontend), the App tier (backend logic), and the DB tier (database). This approach is commonly used in cloud projects to improve system performance, scalability, and security. Each tier can be managed and scaled independently. In this assignment, we are implementing this structure in Amazon Web Services (AWS) using EC2, subnets, and custom security configurations.

3. Objectives:

- Create a Virtual Private Cloud (VPC) for custom networking
- Build 3 subnets for each layer: Web, App, and DB
- Configure internet access only for the Web tier
- Restrict the DB tier from accessing Web or internet
- Allow App tier to talk with both Web and DB
- Launch Linux and Windows virtual machines in each subnet
- Install Apache server on Linux and IIS server on Windows
- Use security groups to restrict unwanted access
- Route traffic properly using route tables and gateways

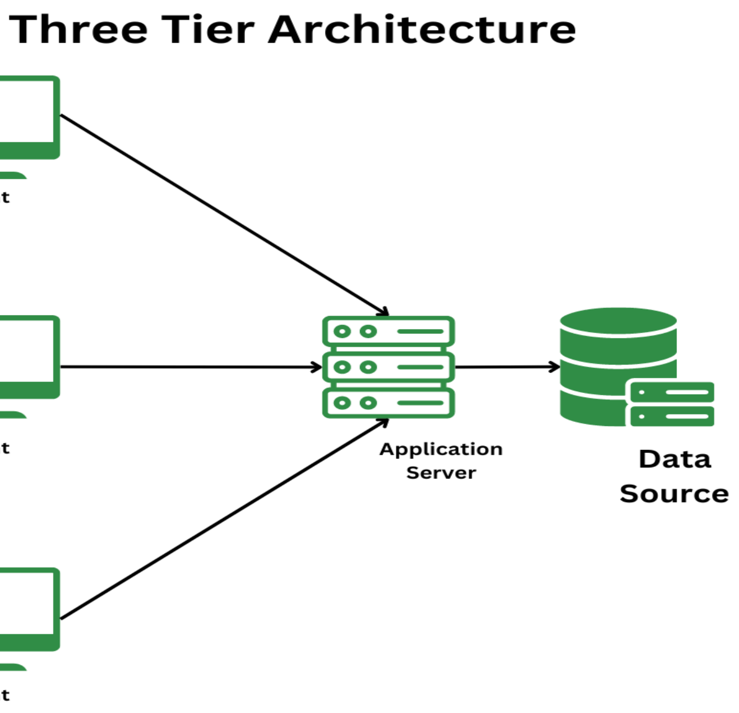
4. Architecture Overview:

- VPC CIDR Block: 10.0.0.0/16 (for internal IP addressing)
- Subnets:
    - ➢ Web Tier Subnet (10.0.1.0/24) – Public
    - ➢ App Tier Subnet (10.0.2.0/24) – Private

- DB Tier Subnet (10.0.3.0/24) – Private and Isolated

- Internet Gateway (IGW): Allows internet access from Web tier
- NAT Gateway: Allows App tier to access the internet for software updates
- Route Tables:
  - Web Route Table: Routes to IGW
  - App Route Table: Routes to NAT Gateway
  - DB Route Table: No route to internet or other tiers
- Security Groups:
  - WebSG: Open for HTTP/HTTPS from outside, allows access to App tier
  - AppSG: Allows inbound from Web tier, outbound to DB tier
  - DBSG: Only accepts inbound from App tier, no outbound allowed

5. Architecture:



6. Step-by-Step AWS Setup:

Step 1: Create a VPC

- Go to VPC dashboard → Click "Create VPC"

- Set name as: awsdemoworkshop
- IPv4 CIDR block: 10.0.0.0/16
- Choose default options → Click "Create VPC"

Step 2: Create Subnets

- Create 3 subnets under the same VPC:
    - WebSubnet → 10.0.1.0/24 → Availability Zone: us-east-1a
    - AppSubnet → 10.0.2.0/24 → us-east-1b
    - DBSubnet → 10.0.3.0/24 → us-east-1c
- WebSubnet is public (enable auto-assign public IP)

Step 3: Create and Attach Internet Gateway (IGW)

- Create a new IGW and name it WebIGW
- Attach it to 3TierVPC

Step 4: Create Route Tables

- Create 3 route tables:
    - WebRT: Add 0.0.0.0/0 → IGW
    - AppRT: Will be updated with NAT gateway route
    - DBRT: Keep it isolated
- Associate:
    - WebSubnet → WebRT
    - AppSubnet → AppRT
    - DBSubnet → DBRT

Step 5: Create NAT Gateway for App Tier

- Go to NAT Gateway → Click "Create NAT Gateway"
- Choose WebSubnet
- Allocate and attach an Elastic IP
- Link NAT Gateway to AppRT by adding route 0.0.0.0/0 → NAT

Step 6: Create Security Groups

- WebSG:
    - Inbound: HTTP (80), HTTPS (443) from Anywhere
    - Outbound: Allow traffic to AppSG
- AppSG:
    - Inbound: Custom TCP from WebSG (ports like 8080 or 5000)
    - Outbound: Allow traffic to DBSG (port 3306 or 1433 depending on DB)
- DBSG:
    - Inbound: Only from AppSG
    - Outbound: Deny all or restrict

Step 7: Launch EC2 Instances in Each Tier

- Launch 6 instances total:
    - WebSubnet:
        - Amazon Linux 2
        - Windows Server 2019 Base
    - AppSubnet:
        - Amazon Linux 2
        - Windows Server 2019 Base
    - DBSubnet:
        - Amazon Linux 2 (optional MySQL/Postgres)
        - Windows Server with SQL Server (optional)
- Make sure:
    - Web tier has public IP enabled
    - App and DB have private IPs only
    - Select correct security groups

Step 8: Install Apache on Linux VMs

sudo yum update -y

```
sudo yum install httpd -y
sudo systemctl start httpd
sudo systemctl enable httpd
sudo bash -c 'echo "<h1>Apache Web/App/DB Tier</h1>" > /var/www/html/index.html'
```

- Open browser → Enter public IP of Web tier → Apache page should load

Step 9: Install IIS on Windows VMs

- Log in to instance → Open Server Manager
- Add Roles and Features → Web Server (IIS)
- After install, open browser → http://localhost
- You should see the default IIS page

Step 10: Testing Communication

- From Web tier:
    - ➢ Use browser or curl to connect to App tier
- From App tier:
    - ➢ Use telnet or curl to connect to DB tier IP and port
- From DB tier:
    - ➢ Try pinging Web or App tier → It should fail (as expected)

7. Conclusion:

This assignment successfully demonstrates how to build a secure and layered network using AWS. We separated each function (web, app, database) into different subnets and controlled how they connect using route tables and security groups. This improves security by reducing exposure to the internet. Only the web layer is public, while app and DB layers are private. This structure is commonly used for hosting modern enterprise-level applications.