**Configure and deploy below application on Linux server (3 tier application)**

**1. Setting Up MongoDB (Database Layer)**

**Update the package list**

#sudo apt update

**Install MongoDB**

#sudo apt-get install -y mongodb-org

**Start and enable MongoDB service:**

#sudo systemctl start mongod

#sudo systemctl enable mongod

**Check if MongoDB is running correctly by status**

#sudo systemctl status mongod

**Connect to the MongoDB shell**

#mongosh

**2. Setting Up the Backend (Node.js Application)**

**Install Node.js and npm (Node package manager)**

sudo apt install -y nodejs

**Cloning the Application Source**

#git clone https://github.com/BL-AniketChile/NodeJs-API.git

**Open .env file**

#nano .env

**Add the following environment variable in .env file**

MONGODB\_URL=mongodb://localhost:27017/demo

PORT=3000

**Creating .service file in /etc/systemd/system**

#cd /etc/systemd/system

#ls

#nano nodeapp.service

**Add the following content in nodeapp.service file**

[Unit]  
Description=Node.js Application Service  
After=network.target  
  
[Service]  
User=ubuntu  
Group=ubuntu  
WorkingDirectory=/home/ubuntu/NodeJs-API  
ExecStart=/usr/bin/node server.js  
Restart=always  
RestartSec=10  
  
[Install]  
WantedBy=multi-user.target

**Starting the node application as a system service**

#sudo systemctl start nodeapp

#sudo systemctl enable nodeapp

#sudo systemctl status nodeapp

#cd NodeJs-API

**We can also use a process manager like pm2 to keep the app running in the background**

#sudo npm install pm2 -g

#pm2 start server.js

#node server.js

#sudo systemctl status nodeapp.service (It should be active: running)

**2**. **Setting Up the Frontend (Apache Web Server)**

**Install Apache web server**

#sudo apt update

#sudo apt install apache2

**Start and enable Apache**

#sudo systemctl start apache2

#sudo systemctl enable apache2

**Configure Apache as a Reverse Proxy (Enable proxy modules)**

* The Apache web server will act as a reverse proxy to route frontend traffic to Node.js backend application.
* Apache web server will forwards the user requests to backend Node application

#sudo a2enmod proxy

#sudo a2enmod proxy\_http

**Configure the Apache virtual host**

* Create a new configuration file nodeapp.conf in /etc/apache2/sites-available/ directory

#sudo nano /etc/apache2/sites-available/nodeapp.conf

**Add the following proxy settings inside the <VirtualHost \*:80> block:**

<VirtualHost \*:80>

ServerName default

ProxyRequests Off

ProxyPass / http://localhost:3000/

ProxyPassReverse / http://localhost:3000/

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

</VirtualHost>

**Note:**

* We have to use ‘default’ as a server name instead of IP address of server in nodeapp.conf file.
* If we use IP address as Server name, we need to change every time after restarting the Ec2 instance, because Public IP address of instance will change everytime

**After changing the nodeapp.conf file, we need to restart the apache**

#sudo systemctl restart apache2

**We need to add a Rule for HTTP (Port 80)** **in security group (Inbound Rules) of current Ec2 instance**

* **Type**: HTTP
* **Protocol**: TCP
* **Port Range**: 80
* **Source**: My IP (This will allow access only from your **current IP address**)

http://Public IP address of current Ec2 instance/route

* This setup allows Apache to forward requests from the specified port (80) to Node.js application running on port 3000, enabling smooth integration of your frontend and backend
* Port 80 is default port of http

<http://3.109.32.53/hello_world>