Server Setup Documentation

This document outlines the step-by-step process for setting up a server with Jenkins, backend services, Nginx reverse proxy, and secure SSH access.

1. JDK Installation

Steps :

1. Run Following command in Terminal

```
sudo apt-get update
sudo apt install openjdk-21-jdk -y
```

2. Jenkins Installation & Configuration

Description:

Jenkins is installed as a CI/CD tool to automate the build and deployment process. It is configured to run on port 8081 with a custom home directory.

Steps:

1. Add Jenkins repository and install dependencies:

```
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo"deb[signed-by=/usr/share/keyrings/jenkins-keyring.asc]
https://pkg.jenkins.io/debian-stable binary/" |
sudo tee /etc/apt/sources.list.d/jenkins.list
```

2. Install Jenkins:

```
sudo apt-get install jenkins
```

3. Configure Jenkins service:

```
sudo systemctl enable jenkins
sudo systemctl start jenkins
```

4. Modify Jenkins home directory and port:

Edit the Jenkins service file:

```
sudo nano /usr/lib/systemd/system/jenkins.service
```

5. Update the following lines:

```
Environment="JENKINS_HOME=/home/azureuser/backend/jenkins-home"
Environment="JENKINS_PORT=8081"
```

6. Reload and restart Jenkins:

```
sudo systemctl daemon-reload
sudo systemctl restart jenkins
```

7. Open the firewall for Jenkins:

```
sudo ufw allow 8081/tcp
sudo ufw reload
```

3. Backend Services Setup

Description:

Multiple backend services (e.g., service-registry, users, notifications) are deployed as systemd services. Each service runs as a Spring Boot application managed by systemd.

Steps:

1. Create a systemd service file for each backend service:
   `sudo nano /etc/systemd/system/backend-service-registry.service`

2. Example service file content:

```
[Unit]
Description=Backend Service Registry
After=network.target

[Service]
User=azureuser
WorkingDirectory=/var/lib/jenkins/backend/backend-service-registry
ExecStart=/usr/bin/java -jar app.jar
Restart=always

[Install]
WantedBy=multi-user.target
```

3. Reload and enable the service:

```
sudo systemctl daemon-reload
sudo systemctl enable backend-service-registry
sudo systemctl start backend-service-registry
```

4. Set proper permissions for the service directory:

```
sudo mkdir -p /var/lib/jenkins/backend/backend-service-registry
sudo chown -R azureuser:azureuser /var/lib/jenkins/backend
sudo chmod -R 755 /var/lib/jenkins/backend
```

5. Repeat the above steps for other services (e.g., backend-users, backend-notification).

4. Nginx Reverse Proxy Setup

Description:

Nginx is configured as a reverse proxy to route incoming requests to the appropriate backend services based on the API path.

Steps:

1. Install Nginx:

```
sudo apt install nginx -y
```

2. Configure Nginx as a reverse proxy:

Edit the default Nginx configuration file:

```
sudo nano /etc/nginx/sites-available/default
```

3. Add proxy configurations:

```
server {
        listen 80;
        server_name yourdomain.com;

        location / {
           root /var/www/react-app/user-mgmt/;
           try_files $uri /index.html;
        }
```

```
        location /api/registry {
            proxy_pass http://localhost:7001;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
        }

        location /api/users {
            proxy_pass http://localhost:7002;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
        }

        location /api/notifications {
            proxy_pass http://localhost:7003;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
        }
}
```

4. Test and restart Nginx:

```
sudo nginx -t
sudo systemctl restart nginx
sudo ufw allow 80/tcp
```

5. SSH Configuration

Description:

SSH access is secured by disabling password authentication and enabling key-based authentication.

Steps:

1. Configure SSH access:

```
mkdir -p ~/.ssh
```

```
chmod 700 ~/.ssh
nano ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
```

2. Modify SSH configuration:

   a. Edit the SSH config file:

   ```
   sudo nano /etc/ssh/sshd_config
   ```

   b. Ensure the following settings:

   ```
   PermitRootLogin no
   PasswordAuthentication no
   PubkeyAuthentication yes
   ```

3. Restart the SSH service:

```
sudo systemctl restart ssh
```

6. Maintenance & Monitoring

Description:

Regular monitoring and maintenance tasks are performed to ensure the server and services are running smoothly.

Steps:

1. Check running services:

```
sudo systemctl list-units --type=service
```

2. Monitor logs:

```
sudo journalctl -u jenkins -f --no-pager
sudo journalctl -u backend-service-registry -f
```

7. Additional Components

Description:

Additional tools like Maven, Node.js, and Git are installed to support the development and deployment process.

Steps:

1. Install Maven:

```
sudo apt install maven -y
```

2. Install Node.js:

```
sudo apt install -y ca-certificates curl gnupg
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://deb.nodesource.com/gpgkey/nodesource-repo.gpg.key
|sudo gpg --dearmor -o /etc/apt/keyrings/nodesource.gpg
NODE_MAJOR=20
echo "deb [signed-by=/etc/apt/keyrings/nodesource.gpg]
https://deb.nodesource.com/node_$NODE_MAJOR.x nodistro main" |
sudo tee /etc/apt/sources.list.d/nodesource.list
sudo apt update && sudo apt install -y nodejs gcc g++ make
```

3. Install Git:

```
sudo apt install git -y
```

8. Setup Frontend

Description:

The frontend application is deployed under /var/www/frontend, but you can replace this directory as needed.

Steps:

1. Create directory if not exist:

```
sudo mkdir -p /var/www/frontend/reacy-app/user-mgmt/
```

2. Set the correct permissions (so NGINX can access it):

```
sudo chown -R www-data:www-data /var/www/frontend
```