MERN Frontend Deployment Guide - AWS EC2 with Nginx

Prerequisites

- AWS EC2 instance (Ubuntu 20.04 LTS or 22.04 LTS recommended)
- Domain name (optional but recommended)
- SSH access to your EC2 instance

Step 1: EC2 Instance Setup

Launch EC2 Instance

```bash
# Recommended instance type: t3.micro (free tier) or t3.small
# Security Group Rules:
# - SSH (22) - Your IP
# - HTTP (80) - 0.0.0.0/0
# - HTTPS (443) - 0.0.0.0/0
# - Custom TCP (4000) - 0.0.0.0/0 (for backend API)
```

Connect to Instance

```bash
ssh -i your-key.pem ubuntu@your-ec2-public-ip
```

Step 2: Install Dependencies

Update System

```bash
sudo apt update && sudo apt upgrade -y
```

Install Node.js and npm

```bash
# Install Node.js 18.x (LTS)
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs

# Verify installation
node --version
npm --version
```

Install Nginx

```bash
sudo apt install nginx -y
sudo systemctl start nginx
sudo systemctl enable nginx
```

Install PM2 (Process Manager)

```bash
sudo npm install -g pm2
```

Install Git

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

Step 3: Deploy Your Application

Clone Your Repository

```bash
cd /home/ubuntu
git clone https://github.com/your-username/your-repo.git
cd your-repo/frontend
```

Install Dependencies and Build

```bash
# Install dependencies
npm install

# Build for production
npm run build
```

Set up Application Structure

```bash
# Create application directory
sudo mkdir -p /var/www/amazona-frontend
sudo chown -R ubuntu:ubuntu /var/www/amazona-frontend

# Copy build files
cp -r build/* /var/www/amazona-frontend/
```

Step 4: Configure Nginx

Create Nginx Configuration

```bash
sudo nano /etc/nginx/sites-available/amazona-frontend
```

Add Configuration

nginx

```nginx
server {
    listen 80;
    server_name your-domain.com www.your-domain.com;  # Replace with your domain or EC2 IP

    root /var/www/amazona-frontend;
    index index.html;

    # Serve static files
    location / {
        try_files $uri $uri/ /index.html;
    }

    # API Proxy to Backend (Port 4000)
    location /api/ {
        proxy_pass http://localhost:4000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_cache_bypass $http_upgrade;
    }

    # Serve static assets with caching
    location /static/ {
        expires 1y;
        add_header Cache-Control "public, immutable";
    }

    # Handle favicon
    location /favicon.ico {
        access_log off;
        log_not_found off;
    }

    # Security headers
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header Referrer-Policy "no-referrer-when-downgrade" always;
    add_header Content-Security-Policy "default-src 'self' http: https: data: blob: 'unsafe-inl
}
```

Enable Site

```bash
# Enable the site
sudo ln -s /etc/nginx/sites-available/amazona-frontend /etc/nginx/sites-enabled/

# Remove default site
sudo rm /etc/nginx/sites-enabled/default

# Test Nginx configuration
sudo nginx -t

# Restart Nginx
sudo systemctl restart nginx
```

Step 5: Set up Backend (if needed)

If Backend is in Same Repository

```bash
cd /home/ubuntu/your-repo/backend

# Install dependencies
npm install

# Create PM2 ecosystem file
nano ecosystem.config.js
```

PM2 Ecosystem Configuration

```javascript
module.exports = {
  apps: [{
    name: 'amazona-backend',
    script: 'server.js', // or your main backend file
    cwd: '/home/ubuntu/your-repo/backend',
    instances: 1,
    autorestart: true,
    watch: false,
    max_memory_restart: '1G',
    env: {
      NODE_ENV: 'production',
      PORT: 4000
    }
  }]
};
```

Start Backend with PM2

```bash
pm2 start ecosystem.config.js
pm2 save
pm2 startup
```

Step 6: Environment Variables

Create Environment File

```bash
nano /home/ubuntu/your-repo/frontend/.env.production
```

Add Production Variables

```env
REACT_APP_API_URL=https://your-domain.com/api
REACT_APP_PAYPAL_CLIENT_ID=your-paypal-client-id
REACT_APP_GOOGLE_MAPS_API_KEY=your-google-maps-key
```

Rebuild with Environment Variables

```bash
cd /home/ubuntu/your-repo/frontend
npm run build
cp -r build/* /var/www/amazona-frontend/
```

Step 7: SSL Certificate (Recommended)

Install Certbot

```bash
sudo apt install certbot python3-certbot-nginx -y
```

Obtain SSL Certificate

```bash
sudo certbot --nginx -d your-domain.com -d www.your-domain.com
```

Auto-renewal

```bash
sudo crontab -e
# Add this line:
0 12 * * * /usr/bin/certbot renew --quiet
```

Step 8: Monitoring and Logs

Check Application Status

```bash
# Check Nginx status
sudo systemctl status nginx

# Check PM2 processes
pm2 status

# View logs
pm2 logs amazona-backend
sudo tail -f /var/log/nginx/error.log
sudo tail -f /var/log/nginx/access.log
```

Set up Log Rotation

bash

```bash
sudo nano /etc/logrotate.d/amazona
```

text

```text
/var/log/nginx/*.log {
    daily
    missingok
    rotate 52
    compress
    delaycompress
    notifempty
    create 0644 www-data www-data
    postrotate
        if [ -f /var/run/nginx.pid ]; then
            kill -USR1 `cat /var/run/nginx.pid`
        fi
    endscript
}
```

Step 9: Security Hardening

Configure UFW Firewall

bash

```bash
sudo ufw enable
sudo ufw allow ssh
sudo ufw allow 'Nginx Full'
sudo ufw status
```

Regular Updates Script

bash

```bash
nano ~/update_system.sh
```

bash

```bash
#!/bin/bash
sudo apt update && sudo apt upgrade -y
npm update -g
pm2 update
sudo systemctl reload nginx
```

```bash
chmod +x ~/update_system.sh
```

Step 10: Deployment Script

Create Deployment Script

```bash
nano ~/deploy.sh
```

```bash
#!/bin/bash
set -e

echo "🚀 Starting deployment..."

# Navigate to project directory
cd /home/ubuntu/your-repo

# Pull latest changes
git pull origin main

# Frontend deployment
echo "📦 Building frontend..."
cd frontend
npm ci
npm run build
sudo cp -r build/* /var/www/amazona-frontend/

# Backend deployment
echo "🔄 Restarting backend..."
cd ../backend
npm ci
pm2 restart amazona-backend

# Reload Nginx
sudo systemctl reload nginx

echo "✅ Deployment completed successfully!"
```

```bash
chmod +x ~/deploy.sh
```

Troubleshooting

Common Issues

1. **502 Bad Gateway**

```bash
# Check if backend is running
pm2 status
# Check logs
pm2 logs amazona-backend
```

2. **Static Files Not Loading**

```bash
# Check file permissions
sudo chown -R www-data:www-data /var/www/amazona-frontend
sudo chmod -R 755 /var/www/amazona-frontend
```

3. **API Calls Failing**

```bash
# Check proxy configuration in Nginx
sudo nginx -t
# Verify backend is running on port 4000
netstat -tlnp | grep :4000
```

Performance Optimization

1. **Enable Gzip Compression**

```nginx
# Add to nginx.conf
gzip on;
gzip_vary on;
gzip_min_length 1024;
gzip_types text/plain text/css text/xml text/javascript application/javascript application/
```

2. **Enable HTTP/2**

```nginx
# In your server block (requires SSL)
listen 443 ssl http2;
```

Monitoring Dashboard

Set up PM2 Web Interface

```bash
pm2 install pm2-server-monit
```

Access monitoring at: `http://your-domain.com:9615`

This setup provides a production-ready deployment of your MERN frontend application with proper security, monitoring, and scalability considerations.