Full-Stack Deployment & Maintenance Playbook

This guide provides a comprehensive, step-by-step process for deploying and maintaining a full-stack application on an Ubuntu VPS.

Application Architecture:

- **Server:** Ubuntu 22.04 VPS
- **Web Server:** Nginx
- **Process Manager:** PM2
- **Security:** Let's Encrypt (SSL/HTTPS)
- **Domains:**
  - `https://your-domain.com` - Static Landing Page
  - `https://app.your-domain.com` - React Frontend
  - `https://api.your-domain.com` - Node.js Backend

Part 1: Initial Server Setup & Deployment

This section covers the entire process from a fresh server to a fully deployed application.

Step 1: DNS Configuration

(Done in your Domain Registrar's control panel)

Point your domains to your server's IP address by creating these 'A' records.

Type	Name/Host	Value/Points To
A	`@`	`YOUR_SERVER_IP`
A	`app`	`YOUR_SERVER_IP`
A	`api`	`YOUR_SERVER_IP`

> **Note:** If a record for `@` already exists, **Edit** it instead of adding a new one.

Step 2: Server Preparation

Connect to your server and install all essential software.

```
# Connect to your server
ssh root@YOUR_SERVER_IP

# Update all system packages
sudo apt update && sudo apt upgrade -y

# Install Nginx
sudo apt-get install -y nginx

# Install Node.js v18 and npm
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs

# Install PM2 (Process Manager) globally
sudo npm install pm2 -g

# Install Git
sudo apt-get install git -y
```

Step 3: Deploy Application Code

Create directories and upload/clone your project files.

```
# 1. Create directories on the server
sudo mkdir -p /var/www/landing-page /var/www/frontend /var/www/backend

# 2. Deploy Backend (via Git Clone)
cd /var/www/backend
# The dot '.' at the end is crucial
git clone https://github.com/your-username/your-repo.git .

# 3. Deploy Frontend (via SCP from local machine)
# On your LOCAL machine, navigate to your React project folder
npm run build
scp -r ./build/* root@YOUR_SERVER_IP:/var/www/frontend/

# 4. Deploy Landing Page (via SCP from local machine)
# On your LOCAL machine, navigate to your landing page folder
scp -r ./* root@YOUR_SERVER_IP:/var/www/landing-page/
```

Step 4: Configure & Start the Backend

Set up the backend environment and run it with PM2.

```
# 1. Navigate to the backend directory on the server
cd /var/www/backend

# 2. Create the environment file
# IMPORTANT: Never commit your .env file to Git!
nano .env
```

Paste your secrets into the `.env` file:

```
DATABASE_URL="your_mongodb_atlas_connection_string"
PORT=5500
JWT_SECRET="your_secret_key"


# 3. Install dependencies
npm install

# 4. Start the application with PM2
pm2 start index.js --name "backend-api"
```

Step 5: Configure Nginx

Create a separate configuration file for each domain.

- **API Backend (`api.your-domain.com`):**
  `sudo nano /etc/nginx/sites-available/api.your-domain.com`

```
server {
    listen 80;
    server_name api.your-domain.com;
    location / {
        # Ensure the port matches your backend's .env file
        proxy_pass http://localhost:5500;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

- **React Frontend (`app.your-domain.com`):**
  `sudo nano /etc/nginx/sites-available/app.your-domain.com`

```
server {
    listen 80;
    server_name app.your-domain.com;
    root /var/www/frontend;
    index index.html;
    location / {
        try_files $uri /index.html;
    }
}
```

- **Landing Page (`your-domain.com`):**
  `sudo nano /etc/nginx/sites-available/your-domain.com`

```
server {
    listen 80;
    server_name your-domain.com www.your-domain.com;
    root /var/www/landing-page;
    index index.html;
    location / {
        try_files $uri $uri/ =404;
    }
}
```

Step 6: Enable Nginx Sites & Secure with SSL

Activate the configurations and add HTTPS.

```
# 1. Enable all three sites by creating symbolic links
sudo ln -s /etc/nginx/sites-available/api.your-domain.com /etc/nginx/site
sudo ln -s /etc/nginx/sites-available/app.your-domain.com /etc/nginx/site
sudo ln -s /etc/nginx/sites-available/your-domain.com /etc/nginx/sites-er

# 2. Test Nginx configuration for errors
sudo nginx -t

# 3. Restart Nginx to apply changes
sudo systemctl restart nginx
```

```
# 4. Whitelist Server IP in MongoDB Atlas
# Go to your Atlas Cluster > Network Access > Add IP Address > Add your s

# 5. Install Certbot and get SSL certificates
sudo apt-get install -y certbot python3-certbot-nginx
sudo certbot --nginx
```

> Follow the Certbot prompts. When asked to select domains, press **Enter** to select all. Choose the **Redirect** option to enforce HTTPS.

Part 2: Updating Your Live Application

This section covers the streamlined process for deploying code changes.

Updating the Backend

Use these steps after pushing new code to your backend's Git repository.

```
# 1. Connect to your server and navigate to the backend directory
ssh root@YOUR_SERVER_IP
cd /var/www/backend

# 2. Pull the latest code from your main branch
git pull origin main

# 3. Install any new dependencies (important!)
npm install

# 4. Restart the application with PM2 for zero-downtime update
pm2 restart backend-api

# 5. Check logs to ensure a successful start
pm2 logs backend-api
```

Updating the Frontend

This process starts on your local machine.

1. **On Your Local Machine:**
    - Make sure you have the latest code (`git pull`).

- Create a new production build:

```
npm run build
```

2. **On Your Server:**

- Clear the old frontend files:

```
sudo rm -rf /var/www/frontend/*
```

3. **On Your Local Machine:**

- Upload the new build files:

```
# Run from your React project directory
scp -r ./build/* root@YOUR_SERVER_IP:/var/www/frontend/
```

4. **Verify in Browser:**

- Open your site `https://app.your-domain.com`.
- Perform a **hard refresh** (`Cmd+Shift+R` or `Ctrl+Shift+R`) to clear the browser cache.