Step-by-Step Guide: Deploying a Next.js Application on an Ubuntu Server with Apache and PM2

**Step 1: SSH into the Server Step1.1:**

* Access your server via SSH using the root user or a user with sufficient privileges.

ssh root@37.9.171.133

## **Step1.1: Create Start Script for PM2**

{

"scripts": {

"start": "next start -p 3000" // Start on port 3000 or any port of your choice

}

}

**Step1.2: No Needed next.config.ts**

import type { NextConfig } from "next";

const nextConfig: NextConfig = {

/\* config options here \*/

experimental: {

cssChunking: true, // default

workerThreads: false,

cpus: 1,

},

staticPageGenerationTimeout: 1000,

};

export default nextConfig;

**Step 2: Update System Packages and Install Dependencies**

* Update the server's package list and install necessary software, including Node.js, npm, Git, and Apache.

sudo apt update

sudo apt install nodejs npm git apache2

**Step 2.5: Install Node & NPM**

sudo apt-get update && apt-get upgrade -y

sudo apt-get install -y curl

sudo curl -sL https://deb.nodesource.com/setup\_24.x | bash

sudo apt-get install -y nodejs

**Step2.5-1: Install Chrome**

sudo apt-get update && apt-get install curl gnupg -y

sudo curl --location --silent https://dl-ssl.google.com/linux/linux\_signing\_key.pub | apt-key add -

sudo sh -c 'echo "deb [arch=amd64] http://dl.google.com/linux/chrome/deb/ stable main" >> /etc/apt/sources.list.d/google.list'

sudo apt-get update

sudo apt-get install google-chrome-stable -y libxss1

sudo rm -rf /var/lib/apt/lists/\*

**Step 3: Create Project Directory**

* Set up the directory where your Next.js application will be hosted.

sudo mkdir -p /var/www/michael9051

sudo chown -R $USER:$USER /var/www/michael9051

**Step 4: Generate an SSH Key (Optional)**

* If you need to access a private Git repository, generate an SSH key to use with Git.

ssh-keygen -t rsa -b 4096 -C "michael9051" -f ~/.ssh/michael9051

* Display the public key and add it to your Git provider:

cat ~/.ssh/michael9051.pub

* Add the key to the repository you wish to clone

**Step 5: Clone the Repository**

* Clone your Next.js project from GitHub or another Git provider.

GIT\_SSH\_COMMAND='ssh -i ~/.ssh/michael9051' git clone git@github.com:Michael9051/nextjs.git /var/www/michael9051

**Step 6: Install Project Dependencies**

* Navigate to the project directory and install the necessary Node.js packages.

cd /var/www/michael9051

npm install

# **Step 6.1: `nmp install` ends with „Killed“**

sudo /bin/dd if=/dev/zero of=/var/swap.1 bs=1M count=1024

sudo /sbin/mkswap /var/swap.1

sudo /sbin/swapon /var/swap.1

**Step 7: Build the Project**

* Build the Next.js project for production.

npm run build

### **Step 7.1: Fix „JavaScript Heap Out of Memory“**

node --max-old-space-size=4096 node\_modules/.bin/vue-cli-service build

**or**

export NODE\_OPTIONS="--max-old-space-size=4096"

# **Step 7.2: Static worker unexpectedly exited with code: null and signal: SIGTERM**

rm -rf .next

rm -rf node\_modules

rm -rf package-lock.json

npm install

npm run build

**Step 8: Install PM2**

* PM2 is a process manager for Node.js that ensures your application remains online. Install it globally.

npm install pm2 -g

pm2 start npm --name "hibreton-frontend" – start

**Step 10: Configure Apache as a Reverse Proxy**

* Apache will serve as a reverse proxy, forwarding requests to the Next.js app running on port 3000.

**Step 10.1: Stop Nginx (if running)**

* If Nginx is running, stop it to avoid port conflicts with Apache.

sudo systemctl stop nginx

**Step 10.2: Create an Apache Configuration File**

* Create a virtual host configuration for your Next.js application.

sudo nano /etc/apache2/sites-available/hibreton-frontend.conf

* Add the following content to the file:

<VirtualHost \*:80>

ServerName 37.9.171.133

ServerAdmin root@magic

DocumentRoot /var/www/michael9051

ProxyPreserveHost On

ProxyPass / http://localhost:3000/

ProxyPassReverse / http://localhost:3000/

ErrorLog ${APACHE\_LOG\_DIR}/michael9051-error.log

CustomLog ${APACHE\_LOG\_DIR}/michael9051-access.log combined

</VirtualHost>

**Step 10.3: Enable the Apache Modules**

* Enable the necessary modules for proxying.

sudo a2enmod rewrite

sudo a2enmod proxy

sudo a2enmod proxy\_http

**Step 10.4: Enable the Site and Disable Default Site**

* Enable your new site and disable the default Apache site.

sudo a2ensite michael9051.conf

sudo a2dissite 000-default.conf

**Step 10.5: Test Apache Configuration**

* Check that the Apache configuration is correct.

sudo apache2ctl configtest

**Step 11: Set Up PM2 to Auto-Start on Reboot**

* Ensure that PM2 starts your application automatically when the server reboots.

pm2 startup systemd

pm2 save

**Step 12: Configure Firewall**

* Allow Apache traffic through the firewall if UFW is active.

sudo ufw allow 'Apache Full'

**Step 13: Access the Application**

* Your Next.js application should now be accessible at http://195.110.58.215.

**Troubleshooting**

**Checking Apache Logs**

* If the application does not load, check the Apache error logs for clues.

sudo tail -f /var/log/apache2/error.log

**Checking the PM2 Process**

* Ensure the Node.js process is running correctly.

pm2 list

* To view logs for the Next.js application:

pm2 logs michael9051

**Restarting or Removing the Application in PM2**

* If you need to restart or remove the application from PM2:

pm2 restart michael9051

pm2 delete michael9051

**Handling Port Conflicts**

* If port 3000 is already in use, find and kill the process using it.

sudo lsof -i :3000

sudo kill -9 <PID>

* Afterward, restart the application with PM2:

pm2 start npm --name "michael9051" -- start

**Next.js Application Not Binding to 0.0.0.0**

* Ensure that your Next.js application is configured to listen on 0.0.0.0 instead of localhost to be accessible externally.

// next.config.js

module.exports = {

server: {

hostname: '0.0.0.0',

port: 3000,

},

};

* With these steps, your Next.js application should be successfully deployed and served using Apache and PM2!