Got it! Below is a **Next.js** specific documentation for **Day 6: Deployment Preparation and Staging Environment Setup**.

**Marketplace Builder Hackathon 2025**

**Day 6: Deployment Preparation and Staging Environment Setup**

This documentation provides a step-by-step guide to preparing and deploying your **Next.js** application for production, setting up a staging environment, and automating deployment with **CI/CD** (Continuous Integration/Continuous Deployment). It covers everything you need to ensure your app is ready for deployment and easily tested in a staging environment.

**Table of Contents**

1. [Overview](#overview)
2. [Prerequisites](#prerequisites)
3. [Step 1: Preparing Your Next.js App for Deployment](#step-1-preparing-your-nextjs-app-for-de)
4. [Step 2: Setting Up a Staging Environment](#step-2-setting-up-a-staging-environment)
5. [Step 3: Continuous Deployment (CD)](#step-3-continuous-deployment-cd)
6. [Step 4: Deployment to Vercel, Netlify, and GitHub Pages](#step-4-deployment-to-vercel-netlify-and)
7. [Step 5: Troubleshooting and Common Issues](#step-5-troubleshooting-and-common-issue)
8. [Conclusion](#conclusion)

**Overview**

In **Day 6** of the **Marketplace Builder Hackathon**, you will focus on preparing your **Next.js** application for deployment and setting up a **staging environment** for testing. You will also automate deployment with Continuous Deployment (CD) and learn how to deploy your app to platforms like **Vercel**, **Netlify**, and **GitHub Pages**.

**Prerequisites**

Before starting this process, make sure you have the following:

* A working **Next.js** app (refer to the previous days for app setup).
* Basic knowledge of Git and GitHub.
* Familiarity with **CI/CD** concepts.
* Accounts with **Vercel**, **Netlify**, or **GitHub** for deployment.

**Step 1: Preparing Your Next.js App for Deployment**

To prepare your Next.js app for deployment, follow these steps:

**1.1. Install Dependencies**

Make sure all the necessary dependencies are installed in your project:

bash

CopyEdit

npm install

**1.2. Optimize Your App**

**1.2.1. Image Optimization**

Next.js has built-in image optimization. Use the <Image> component for image rendering:

jsx

CopyEdit

import Image from 'next/image';

function MyImageComponent() {

return <Image src="/images/sample.jpg" alt="Sample" width={500} height={300} />;

}

**1.2.2. Static Site Generation (SSG)**

Use getStaticProps and getStaticPaths to pre-render pages at build time, improving performance:

jsx

CopyEdit

export async function getStaticProps() {

const res = await fetch('https://api.example.com');

const data = await res.json();

return { props: { data } };

}

**1.2.3. Dynamic Imports**

For large components, use dynamic imports to split bundles and reduce the initial load time:

jsx

CopyEdit

import dynamic from 'next/dynamic';

const DynamicComponent = dynamic(() => import('../components/DynamicComponent'));

function HomePage() {

return <DynamicComponent />;

}

**1.2.4. Analyze Your Build**

Use the next-bundle-analyzer to check for unnecessary large files:

bash

CopyEdit

npm install @next/bundle-analyzer

Then add the following to your next.config.js:

js

CopyEdit

const withBundleAnalyzer = require('@next/bundle-analyzer')({

enabled: process.env.ANALYZE === 'true',

});

module.exports = withBundleAnalyzer({});

**Step 2: Setting Up a Staging Environment**

**2.1. Create a Staging Branch**

To create a separate staging environment, start by creating a staging branch:

bash

CopyEdit

git checkout -b staging

Push it to GitHub:

bash

CopyEdit

git push origin staging

**2.2. Set Up Environment Variables for Staging**

Your staging environment might require different settings than production (e.g., API keys). Create a .env.staging file for staging-specific environment variables:

env

CopyEdit

NEXT\_PUBLIC\_API\_URL=https://staging-api.example.com

NEXT\_PUBLIC\_GOOGLE\_API\_KEY=staging\_google\_api\_key

Make sure to update your staging branch with this .env.staging file.

**Step 3: Continuous Deployment (CD)**

**3.1. Integrate GitHub with Deployment Platforms**

**3.1.1. Vercel**

* Sign in to [Vercel](https://vercel.com/).
* Connect your GitHub repository to Vercel.
* Vercel will automatically deploy the app to production when you push to the main branch and to staging when you push to the staging branch.

**3.1.2. Netlify**

* Sign in to [Netlify](https://www.netlify.com/).
* Link your GitHub repository.
* Set up the build command (npm run build) and the publish directory (out for static exports).
* Configure automatic deployments to Netlify from specific branches (e.g., main for production, staging for testing).

**3.1.3. GitHub Pages**

For static sites:

* Install the GitHub Pages package:

bash

CopyEdit

npm install gh-pages --save-dev

* Add deployment scripts in package.json:

json

CopyEdit

"scripts": {

"predeploy": "npm run build",

"deploy": "gh-pages -d build"

}

* Run the deploy command:

bash

CopyEdit

npm run deploy

**Step 4: Deployment to Vercel, Netlify, and GitHub Pages**

**4.1. Vercel Deployment**

Vercel is the preferred platform for **Next.js** apps. Deploy by connecting your GitHub repository to Vercel, and it will automatically deploy on every push.

**4.2. Netlify Deployment**

Netlify can also host Next.js apps. Use the **static export** feature or configure **Next.js functions** for server-side code.

**4.3. GitHub Pages Deployment**

If you’re deploying a static app, GitHub Pages is a good choice:

* Use the gh-pages package to deploy.
* Configure your repository settings to use the gh-pages branch.

**Step 5: Troubleshooting and Common Issues**

**5.1. Build Errors**

* Make sure all dependencies are correctly installed.
* Check for missing environment variables.

**5.2. Deployment Failures**

* Ensure that you have set up the build and publish directory correctly in your deployment platform.
* Check build logs for errors.

**5.3. Staging Environment Issues**

* Make sure the environment variables in .env.staging are set correctly.
* Ensure the correct branch is deployed.