# Comprehensive Guide: Deploying Your React App to GitHub Pages

This guide provides step-by-step instructions for deploying the Random Quote Generator, a React/TypeScript application built with Vite, to GitHub Pages. We will cover both automated deployment using GitHub Actions and a manual deployment process.

## 1. Introduction

GitHub Pages offers a robust and free platform for hosting static websites directly from your GitHub repository. This guide will walk you through the entire process, from setting up your local project and GitHub repository to configuring a seamless, automated deployment pipeline.

The Random Quote Generator is a feature-rich application built with a modern frontend stack, and this document details the specific configurations required to get it live on the web.

**Application Features:**
- **Framework**: React 18.3 + TypeScript
- **Build Tool**: Vite
- **Styling**: TailwindCSS
- **Key Functionality**: Random quote display, category filtering, favorites management, social sharing, and dark/light mode.

## 2. Prerequisites

Before you begin, ensure you have the following installed and configured:
- **Node.js and npm**: Required for managing project dependencies and running scripts.
- **Git**: For version control and pushing your code to GitHub.

- **A GitHub Account**: To create repositories and use GitHub Pages.
- **A Code Editor**: Such as Visual Studio Code.

# 3. Project & Repository Setup

A specific project structure and configuration are necessary for a successful deployment.

## 3.1. File Structure

Your project should be organized as follows. The key files for deployment are the `vite.config.ts`, `package.json`, and the `.github/workflows/deploy.yml` file.

```
random-quote-generator/
├── .github/
│   └── workflows/
│       └── deploy.yml  # GitHub Actions workflow for automated
deployment
├── dist/               # Build output directory (generated)
├── public/             # Static assets
├── src/                # Application source code
├── package.json        # Project dependencies and scripts
├── vite.config.ts      # Vite configuration with base path
└── README.md
```

## 3.2. `package.json` Configuration

Your `package.json` must include `gh-pages` for manual deployment and scripts for building and deploying the application.

```
{
  "name": "random-quote-generator",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "tsc && vite build",
    "predeploy": "npm run build",
    "deploy": "gh-pages -d dist",
    "lint": "eslint . --ext ts,tsx --report-unused-disable-
directives --max-warnings 0",
    "preview": "vite preview"
  },
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    // ... other dependencies
  },
  "devDependencies": {
    "@types/react": "^18.2.66",
    "@types/react-dom": "^18.2.22",
    "@typescript-eslint/eslint-plugin": "^7.2.0",
    "@typescript-eslint/parser": "^7.2.0",
    "@vitejs/plugin-react": "^4.2.1",
    "eslint": "^8.57.0",
    "eslint-plugin-react-hooks": "^4.6.0",
    "eslint-plugin-react-refresh": "^0.4.6",
    "gh-pages": "^6.1.1",
    "typescript": "^5.2.2",
    "vite": "^5.2.0"
  }
}
```

- **`predeploy`**: This script runs automatically before the `deploy` script, ensuring your application is built before deployment.
- **`deploy`**: This script, powered by `gh-pages`, pushes the contents of the `dist` directory to a special `gh-pages` branch on your repository.

## 3.3. `vite.config.ts` Configuration

For GitHub Pages, you must set the `base` path in your `vite.config.ts` to your repository's name.

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [react()],
  base: '/random-quote-generator/', // <-- Set to your repository
name
})
```

**Important:** The `base` path must be `/` followed by your repository name, followed by a trailing `/`.

## 3.4. Initialize and Push to GitHub

1. **Initialize Git**: In your project's root directory, run:
   `bash git init git add . git commit -m "Initial commit"`
2. **Create a GitHub Repository**: Go to GitHub, create a new repository named `random-quote-generator`.

3. **Push to GitHub**: Link your local repository to the remote one and push your code. Replace `<Your-Username>` with your GitHub username.
   `bash git remote add origin https://github.com/<Your-Username>/random-quote-generator.git git branch -M main git push -u origin main`

# 4. Automated Deployment with GitHub Actions

GitHub Actions automates the build and deployment process whenever you push code to your `main` branch.

## 4.1. The Workflow File

Create a file at `.github/workflows/deploy.yml` with the following content:

```yaml
name: Deploy to GitHub Pages

on:
  push:
    branches:
      - main # Deploys on pushes to the main branch

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout 🛎️
        uses: actions/checkout@v3

      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18' # Use a specific Node.js version

      - name: Install Dependencies 📦
        run: npm install

      - name: Build 🏗️
        run: npm run build

      - name: Deploy 🚀
        uses: peaceiris/actions-gh-pages@v3
        with:
          github_token: ${{ secrets.GITHUB_TOKEN }}
          publish_dir: ./dist
```

### 4.2. How It Works

- `on: push: branches: - main`: This triggers the workflow on every push to the `main` branch.
- `jobs: build-and-deploy`: Defines the sequence of tasks.
  - `actions/checkout@v3`: Checks out your repository's code.
  - `actions/setup-node@v3`: Sets up the specified Node.js version.
  - `npm install`: Installs all project dependencies.
  - `npm run build`: Builds the React application for production. The output is placed in the `dist` directory.
  - `peaceiris/actions-gh-pages@v3`: This action takes the built application from the `dist` directory and deploys it to the `gh-pages` branch of your repository.

### 4.3. Triggering the Deployment

Commit the `.github/workflows/deploy.yml` file and push it to your `main` branch. This will trigger the first automated deployment. You can monitor the progress in the "Actions" tab of your GitHub repository.

## 5. Manual Deployment

If you prefer not to use GitHub Actions, you can deploy your application manually from your local machine.

1. **Build the Project**: First, ensure your project is built with the latest changes.
   `bash npm run build`
2. **Deploy to GitHub Pages**: Run the deploy script.
   `bash npm run deploy`
   This command will push the `dist` folder to the `gh-pages` branch on your GitHub repository.

# 6. Configuring Your Deployed Site

After either the automated or manual deployment completes:

1. **Go to Repository Settings**: Navigate to your repository on GitHub and click on the "Settings" tab.
2. **Select Pages**: In the left sidebar, click on "Pages".
3. **Configure Source Branch**: Under "Build and deployment", set the "Source" to **Deploy from a branch**. Then, select the `gh-pages` branch and the `/(root)` folder.
4. **Save Changes**: Click "Save".

GitHub will provide you with the URL for your live site, which will be in the format `https://<Your-Username>.github.io/random-quote-generator/`.

## 6.1. Custom Domain (Optional)

You can configure a custom domain for your GitHub Pages site in the same "Pages" settings section. You will need to add a `CNAME` file to your `public` directory and configure your domain's DNS records.

# 7. Troubleshooting Common Issues

- **Blank Page After Deployment**: This is often caused by an incorrect `base` path in `vite.config.ts`. Double-check that it matches your repository name exactly (e.g., `/my-repo/`).
- **404 Errors on Sub-pages (for multi-page apps)**: GitHub Pages works best with single-page applications (SPAs). For routing, you may need to use a hash-based router (`HashRouter` in React Router) or a workaround to handle client-side routing.
- **GitHub Actions Failures**: Check the logs in the "Actions" tab of your repository for detailed error messages. Common issues include missing dependencies or build errors.

- **CSS or Images Not Loading**: This is also frequently a `base` path issue. Ensure all asset paths are relative.

# 8. Conclusion

You have now successfully deployed your Random Quote Generator application to GitHub Pages. By following this guide, you can leverage GitHub's powerful and free hosting for your static web projects, with an automated deployment pipeline that makes updating your site as simple as pushing a commit.

---

This guide was generated by the MiniMax Agent.