BỘ THÔNG TIN VÀ TRUYỀN THÔNG

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG CƠ SỞ TẠI TPHCM

KHOA CÔNG NGHỆ THÔNG TIN 2

BÁO CÁO CUỐI KỲ

Chuyên Đề Công Nghệ Phần Mềm

Đề tài: Triển khai website bán điện thoại theo cơ chế CI/CD

Giảng viên hướng dẫn: Lê Thanh Hà

Nhóm sinh viên thực hiện : Trần Ngọc Đăng Khoa – N20DCCN033

TP.HCM, ngày 20 tháng 6 năm 2024

Mục lục

I. Giới thiệu đề tài:

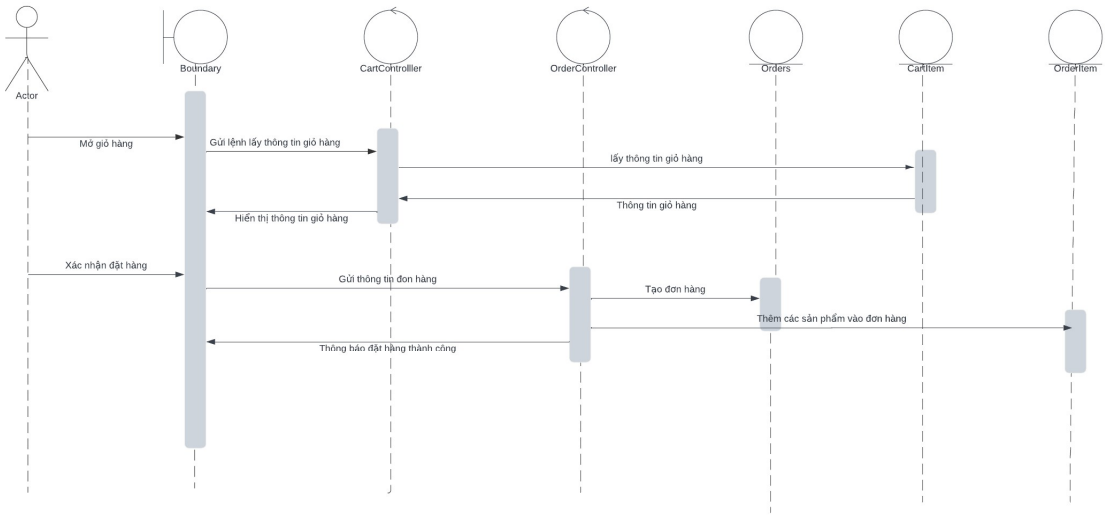
- Trong ngành công nghệ phần mềm, để một phần mềm có thể đến được tay khách hàng cần phải trải một số giai đoạn như: Lên kế hoạch và phân tích yêu cầu – Analysis, Thiết kế phần mềm – Design, Tiến hành coding – Development, Kiểm thử - Testing, Triển khai – Deployment, Bảo trì - Maintenance. Ở đồ án nhỏ này, em sẻ trình bày và thực hiện công đoạn Triển khai – Deployment theo cơ chế CI/CD của việc phát triển phần mềm.

II. Giới thiệu website:

1. Chứng năng của website:

- Xem sản phẩm
- Tìm kiếm sản phẩm
- Đăng kí, đăng nhập, đăng xuất
- Chọn sản phẩm theo option.
- Thêm sản phẩm vào giỏ hàng
- Quản lý giỏ hàng.
- Thanh toán và theo dõi đơn hang.

2. Tuần tự hoạt động chính của website:



3. Công nghệ sử dụng trong website:

- Frontend: sử dụng React js, sass.
- Backend: sử dụng sql server, asp.net core.

III. Thực hiện triển khai website theo phương thức CI/CD:

1. Phương thức CI/CD:

1.1.CI/CD là gì:

CI/CD, viết tắt của tích hợp liên tục và phân phối/triển khai liên tục, nhằm mục đích hợp lý hóa và đẩy nhanh vòng đời phát triển phần mềm.

Tích hợp liên tục (CI) đề cập đến thực tiễn tích hợp các thay đổi mã một cách tự động và thường xuyên vào kho lưu trữ mã nguồn được chia sẻ. Phân phối và/hoặc triển khai liên tục (CD) là một quy trình gồm 2 phần đề cập đến việc tích hợp, thử nghiệm và phân phối các thay đổi mã. Phân phối liên tục dừng việc triển khai sản xuất tự động, trong khi việc triển khai liên tục sẽ tự động phát hành các bản cập nhật vào môi trường sản xuất.

Kết hợp lại với nhau, các phương pháp thực hành được kết nối này thường được gọi là "đường dẫn CI/CD" và được hỗ trợ bởi các nhóm phát triển và vận hành làm việc cùng nhau một cách linh hoạt với phương pháp DevOps hoặc kỹ thuật độ tin cậy của trang web (SRE).

1.2. Tại sao CI/CD lại quan trọng:

CI/CD giúp các tổ chức tránh được lỗi và lỗi mã trong khi duy trì chu kỳ phát triển và cập nhật phần mềm liên tục.

Khi các ứng dụng phát triển lớn hơn, các tính năng của CI/CD có thể giúp giảm độ phức tạp, tăng hiệu quả và hợp lý hóa quy trình làm việc.

Vì CI/CD tự động hóa sự can thiệp thủ công của con người theo truyền thống để lấy mã mới từ cam kết đưa vào sản xuất nên thời gian ngừng hoạt động được giảm thiểu và việc phát hành mã diễn ra nhanh hơn. Và với khả năng tích hợp các bản cập nhật và thay đổi mã nhanh hơn, phản hồi của người dùng có thể được kết hợp thường xuyên và hiệu quả hơn, mang lại kết quả tích cực cho người dùng cuối và tổng thể khách hàng hài lòng hơn.

2. Github Action:

GitHub Actions là nền tảng tích hợp liên tục và phân phối liên tục (CI/CD) cho phép tự động hóa quy trình xây dựng, thử nghiệm và triển khai của mình. Chúng ta có thể tạo quy trình làm việc để xây dựng và kiểm tra mọi yêu cầu kéo tới kho lưu trữ của mình hoặc triển khai các yêu cầu kéo đã hợp nhất vào sản xuất.

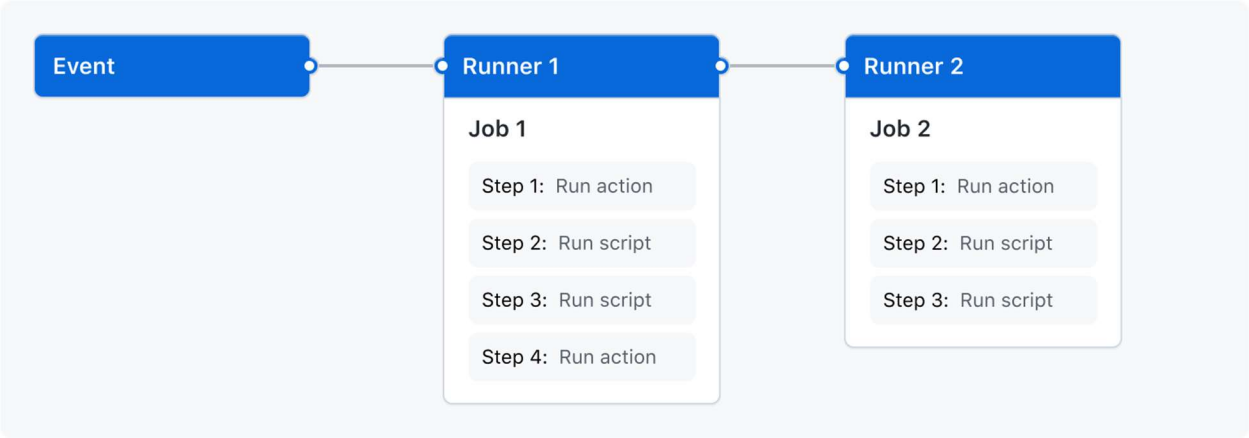
GitHub Actions không chỉ dừng lại ở DevOps và cho phép chạy quy trình công việc khi các sự kiện khác diễn ra trong kho lưu trữ. Ví dụ: có thể chạy quy trình làm việc để tự động thêm nhãn thích hợp bất cứ khi nào ai đó tạo sự cố mới trong kho lưu trữ.

GitHub cung cấp các máy ảo Linux, Windows và macOS để chạy quy trình công việc hoặc chúng ta có thể lưu trữ các trình chạy tự lưu trữ của riêng mình trong trung tâm dữ liệu hoặc cơ sở hạ tầng đám mây của riêng cá nhân.

2.1. Các components của github Action:

Chúng ta có thể định cấu hình quy trình làm việc Tác vụ GitHub để được kích hoạt khi một sự kiện xảy ra trong kho lưu trữ, chẳng hạn như yêu cầu kéo được mở hoặc sự cố đang được tạo. Quy trình công việc chứa một hoặc nhiều công việc có thể chạy theo thứ

tự tuần tự hoặc song song. Mỗi công việc sẽ chạy bên trong trình chạy máy ảo của riêng nó hoặc bên trong một vùng chứa và có một hoặc nhiều bước chạy tập lệnh xác định hoặc chạy một hành động, đây là một tiện ích mở rộng có thể sử dụng lại có thể đơn giản hóa quy trình làm việc.



2.2.Workflow:

Luồng công việc là một quy trình tự động có thể định cấu hình để thực hiện một hoặc nhiều công việc. Quy trình làm việc được xác định bằng tệp YAML được đăng ký trong kho lưu trữ và sẽ chạy khi được kích hoạt bởi một sự kiện hoặc chúng có thể được kích hoạt theo cách thủ công hoặc theo lịch trình đã xác định.

Quy trình công việc được xác định trong thư mục .github/workflows trong kho lưu trữ và kho lưu trữ có thể có nhiều quy trình công việc, mỗi quy trình có thể thực hiện một nhóm tác vụ khác nhau. Ví dụ: có thể có một quy trình công việc để xây dựng và kiểm tra các yêu cầu kéo, một quy trình công việc khác để triển khai ứng dụng của bạn mỗi khi bản phát hành được tạo và một quy trình công việc khác thêm nhãn mỗi khi ai đó mở một vấn đề mới.

2.3.Event:

Sự kiện là một hoạt động cụ thể trong kho lưu trữ kích hoạt quá trình chạy quy trình công việc. Ví dụ: một hoạt động có thể bắt nguồn từ GitHub khi ai đó tạo yêu cầu kéo, mở một vấn đề hoặc đẩy một commit vào kho lưu trữ. Chúng ta cũng có thể kích hoạt quy trình làm việc để chạy theo lịch bằng cách đăng lên API REST hoặc theo cách thủ công.

Để biết danh sách đầy đủ các sự kiện có thể dùng để kích hoạt quy trình làm việc, hãy xem Sự kiện kích hoạt quy trình công việc.

2.4.Job:

Công việc là tập hợp các bước trong quy trình công việc được thực thi trên cùng một nhân viên. Mỗi bước là một tập lệnh shell sẽ được thực thi hoặc một hành động sẽ được chạy. Các bước được thực hiện theo thứ tự và phụ thuộc lẫn nhau. Vì mỗi bước được thực thi trên cùng một trình chạy nên bạn có thể chia sẻ dữ liệu từ bước này sang bước khác. Ví dụ: Chúng ta có thể có một bước xây dựng ứng dụng của mình, sau đó là bước kiểm tra ứng dụng đã được xây dựng.
Chúng ta có thể định cấu hình các phần phụ thuộc của công việc với các công việc khác; theo mặc định, các công việc không có sự phụ thuộc và chạy song song với nhau. Khi một công việc phụ thuộc vào một công việc khác, nó sẽ đợi công việc phụ thuộc hoàn thành trước khi có thể chạy. Ví dụ: Chúng ta có thể có nhiều công việc xây dựng cho các kiến trúc khác nhau không có phần phụ thuộc và một công việc đóng gói phụ thuộc vào những công việc đó. Các công việc xây dựng sẽ chạy song song và khi tất cả chúng hoàn thành thành công, công việc đóng gói sẽ chạy.

2.5.Action:

Hành động là một ứng dụng tùy chỉnh dành cho nền tảng GitHub Action thực hiện một tác vụ phức tạp nhưng thường xuyên lặp lại. Sử dụng một hành động để giúp giảm số lượng mã lặp lại được viết trong tệp quy trình làm việc của mình. Một hành động có thể lấy kho lưu trữ git của bạn từ GitHub, thiết lập chuỗi công cụ chính xác cho môi trường xây dựng hoặc thiết lập xác thực cho nhà cung cấp đám mây.

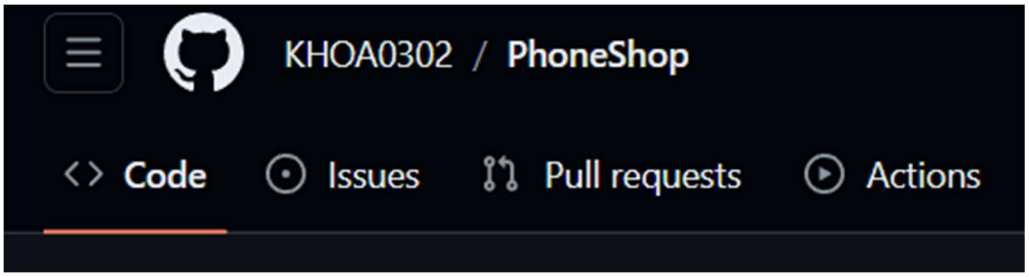
2.6.Runner:

Runer là máy chủ chạy quy trình công việc khi chúng được kích hoạt. Mỗi runner có thể chạy một công việc tại một thời điểm. GitHub cung cấp các trình chạy Ubuntu Linux, Microsoft Windows và macOS để chạy quy trình làm việc; mỗi lần chạy quy trình làm việc sẽ thực thi trong một máy ảo mới được cung cấp. GitHub cũng cung cấp các trình chạy lớn hơn, có sẵn ở các cấu hình lớn hơn.

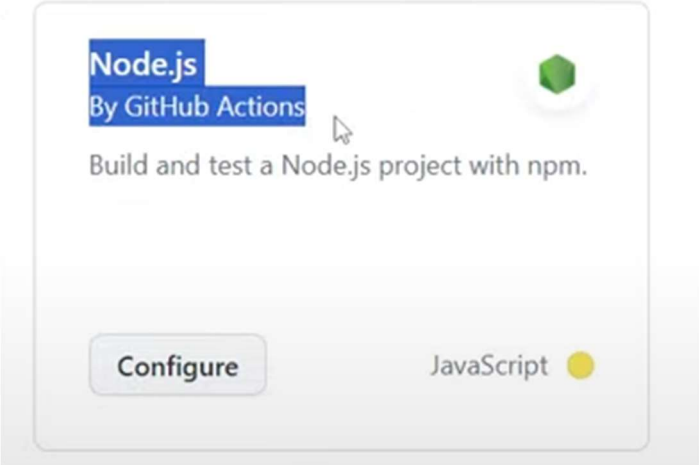
3. Thực hiện triển khai:

3.1.Cấu hình CI/CD trên github Action:

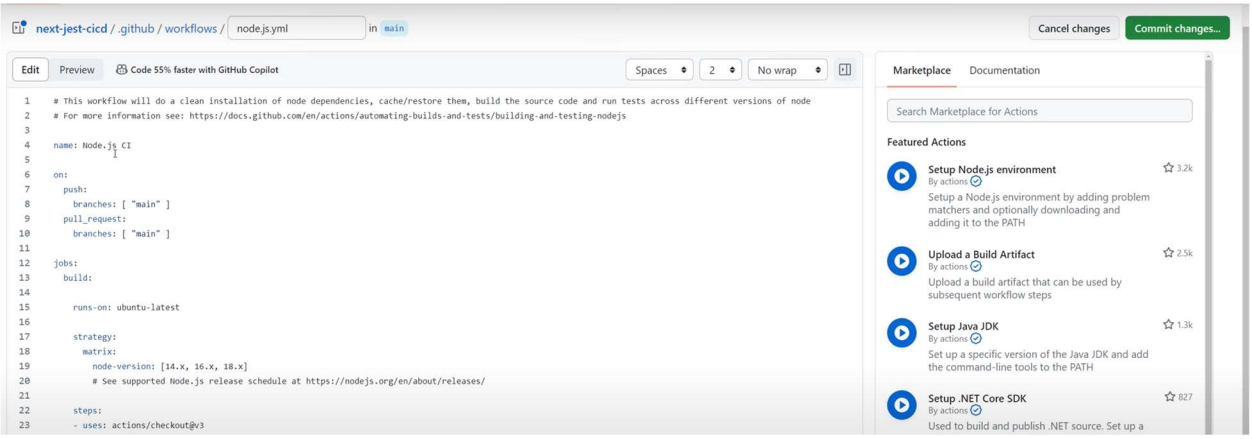
Tạo một repo, đẩy một án Reactjs lên repo vừa tạo trên github.
Vào repo vừa tạo chọn github action:



Tìm kiếm một github action phù hợp với dự án, ở đây em chọn Node.js:

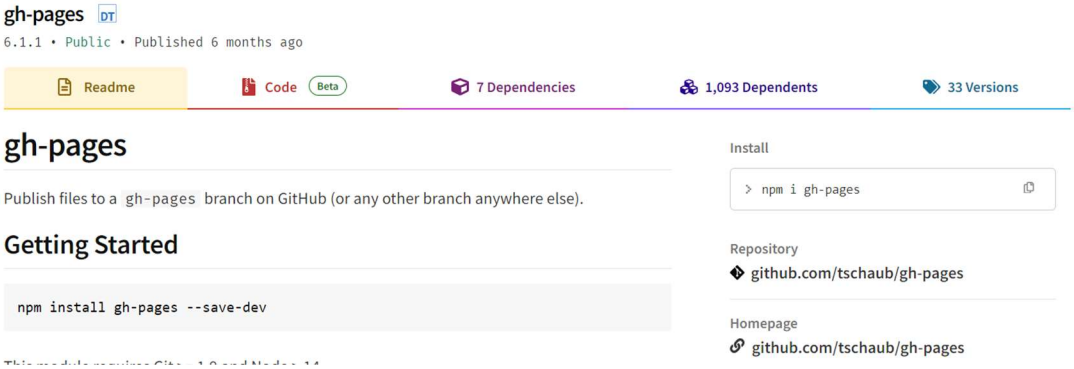


Ở đây Github tạo ra một workflow cho Node.js:



Chuẩn bị điều kiện cần để Deploy web lên môi trường production:

- Tải gh-pages cho dự án.



- Tiến hành config file package.json homepage và deploy.

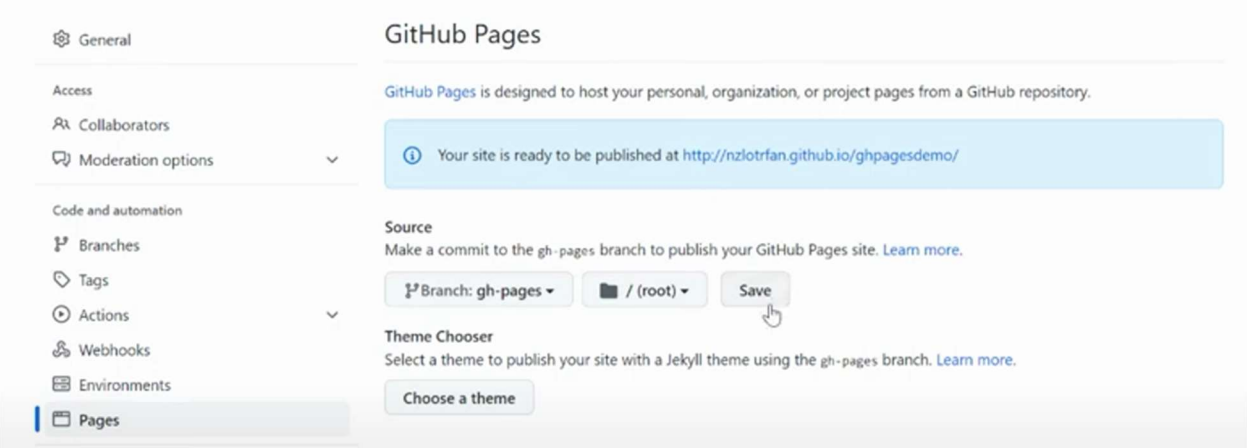
```
{
  "name": "phone_store",
  "version": "0.1.0",
  "private": true,
  "homepage": "http://KHOA0302.github.io/PhoneShop",
```

```
  "scripts": {
    "deploy": "gh-pages -d build",
    "start": "react-app-rewired start",
    "build": "CI=false react-app-rewired build",
    "test": "react-app-rewired test",
    "eject": "react-app-rewired eject"
```

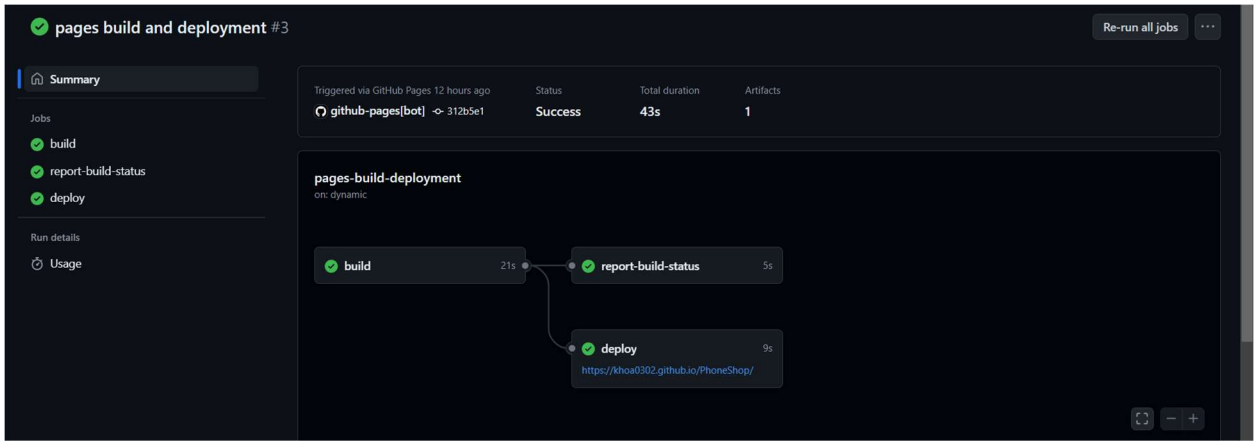
- Tiến hành config file node.js.yml:

```
    - name: Deploy with gh-pages
      run: |
        git remote set-url origin https://git:${GITHUB_TOKEN}@github.com/${GITHUB_REPOSITORY}.git
        npm run deploy -- -u "github-actions-bot <support+actions@github.com>"
      env:
        GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

3.2.Tiến hành deploy website sử dụng CI/CD:



3.3.Workflow của 1 job trên Github Action:



Ở đây khi thực hiện lệnh push trên branch main, event sẽ được kích hoạt và chạy các job đã được thiết kế trong Github Action.

4. Script CI/CD workflow:

4.1. Update:

```yaml
name: Node.js CI

on:
  push:
    branches: ["update"]
  pull_request:
    branches: ["update"]

jobs:
  build:
    runs-on: ubuntu-latest

    strategy:
      matrix:
        node-version: [18.x]
        # See supported Node.js release schedule at https://nodejs.org/en/about/releases/

    steps:
      - uses: actions/checkout@v4
      - name: Use Node.js ${{ matrix.node-version }}
        uses: actions/setup-node@v3
        with:
          node-version: ${{ matrix.node-version }}
          cache: "npm"
      - run: npm ci
```

```
      - run: npm run build --if-present
      - run: npm test
```

31.1. Deployment:

```
name: Deployment

on:
  push:
    branches: ["main"]
  pull_request:
    branches: ["main"]

jobs:
  build:
    runs-on: ubuntu-latest

    strategy:
      matrix:
        node-version: [18.x]
        # See supported Node.js release schedule at https://nodejs.org/en/about/releases/

    steps:
      - uses: actions/checkout@v4
      - name: Use Node.js ${{ matrix.node-version }}
        uses: actions/setup-node@v3
        with:
          node-version: ${{ matrix.node-version }}
          cache: "npm"
      - run: npm ci
      - run: npm run build --if-present
      - run: npm test
      - name: Deploy with gh-pages
        run: |
          git remote set-url origin https://git:${GITHUB_TOKEN}@github.com/${GITHUB_REPOSITORY}.git
          npm run deploy -- -u "github-actions-bot <support+actions@github.com>"
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

Giải thích:

name: Deployment

Dòng này xác định tên của quy trình công việc, trong trường hợp này là "Triển khai".

on: Phần này xác định các sự kiện kích hoạt quy trình làm việc.

push: Quy trình làm việc sẽ được kích hoạt bất cứ khi nào có sự kiện push vào kho lưu trữ.

branch: ["main"]: Điều này chỉ định rằng quy trình làm việc sẽ chỉ được kích hoạt khi đẩy đến nhánh "main".

pull_request: Quy trình làm việc cũng sẽ được kích hoạt cho các sự kiện yêu cầu kéo.

branch: ["main"]: Tương tự như sự kiện đẩy, điều này chỉ định rằng quy trình làm việc sẽ chỉ chạy cho các yêu cầu kéo nhắm mục tiêu vào nhánh "main".

jobs: Phần này xác định các công việc sẽ được thực hiện như một phần của quy trình làm việc. Ở đây, chúng ta có một công việc tên là "build".

build: Phần này xác định chi tiết về công việc "build".

Runs-on: ubuntu-lates: Điều này chỉ định rằng công việc sẽ chạy trên máy ảo Ubuntu có phiên bản mới nhất.

strategy: Phần này xác định chiến lược ma trận để thực hiện công việc. Điều này cho phép chúng tôi chạy công việc với các phiên bản Node.js khác nhau.

matrix: Xác định rằng chúng ta đang sử dụng strategy matrix.

node-version: [18.x]: Điều này chỉ định rằng công việc sẽ được chạy một lần cho mỗi phiên bản trong danh sách được cung cấp, trong trường hợp này chỉ là 18.x (đề cập đến bất kỳ phiên bản Node.js nào bắt đầu bằng 18).

step: Phần này xác định các bước sẽ được thực hiện tuần tự trong công việc "build".

- sử dụng: actions/checkout@v4: Bước này sử dụng hành động/ kiểm tra từ GitHub Actions để kiểm tra mã từ kho lưu trữ.

- Tên: Sử dụng Node.js ${{matrix.node-version }}: Bước này xác định tên cho step (tùy chọn) và sử dụng hành động/thiết lập node để cài đặt một phiên bản Node.js cụ thể ({{ matrix.node-version }}) trên đường chạy.

với: node-version: ${{ matrix.node-version }}: Điều này chỉ định phiên bản Node.js mong muốn sẽ được cài đặt.

cache: "npm": Thao tác này yêu cầu hành động lưu phiên bản Node.js đã tải xuống và các gói npm để có hiệu suất tốt hơn trong những lần chạy tiếp theo.

- run: npm ci: Bước này chạy lệnh npm ci để cài đặt tất cả các phụ thuộc được liệt kê trong tệp pack-lock.json.

- run: npm run build --if-present: Bước này cố gắng chạy tập lệnh npm có tên "build" nếu nó được xác định trong tệp pack.json. Flag --if-present đảm bảo bước này không thất bại nếu tập lệnh không tồn tại.

- run: npm test: Bước này chạy lệnh npm test để thực thi bộ test được xác định trong dự án.

- name: Triển khai với gh-pages: Bước này xác định tên cho bước triển khai và sử dụng tập lệnh tùy chỉnh để triển khai ứng dụng đã xây dựng.

run: |: Điều này xác định một chuỗi nhiều dòng chứa tập lệnh triển khai.

git remote set-url Origin https://git:${GITHUB_TOKEN}@github.com/${GITHUB_REPOSITORY}.git: Dòng này cập nhật URL từ xa cho nhánh "root" bằng cách sử dụng mã thông báo truy cập tạm thời được tạo cho quy trình làm việc (GITHUB_TOKEN).

npm run triển khai -- -u "github-actions-bot <support+actions@github.com>": Dòng này giả định có một tập lệnh có tên "deploy" được xác định trong tệp pack.json. Nó thực thi tập lệnh với các đối số bổ sung:

--: Tách tên tập lệnh khỏi các đối số sau.

-u "github-actions-bot <support+actions@github.com>": Đặt tên người dùng và email cho quá trình triển khai (có thể nhằm mục đích kiểm soát phiên bản).

env: Phần này xác định các biến môi trường được sử dụng trong tập lệnh triển khai.

GITHUB_TOKEN: ${{ secret.GITHUB_TOKEN }}: Thao tác này sẽ đưa bí mật có tên GITHUB_TOKEN từ kho lưu trữ bí mật vào môi trường. Mã thông báo này có thể được sử dụng để xác thực trong quá trình triển khai.