Hands-on Lab: Update Components and Redeploy the Website

Estimated time needed: **30** minutes

Introduction

You recently deployed the Medical Appointment Booking website. The CEO has now asked you to increase traffic to the website. You can increase the website traffic by making increasing the website ranking on search engines, for example, Google. You can add keywords for search engine optimization (SEO), such as doctor appointment, remote doctor access, and so on. In this lab, you will add meta tags for keywords, content, and description for SEO. You will then redeploy the website.

Objectives

After completing this lab, you will be able to:

- Add SEO keywords for website components
- Commit the changes in your GitHub repository
- Redeploy the website in the Skills Network lab environment
- (Optionally) Simulate an automated continuous integration and continuous deployment (CI/CD) pipeline using GitHub actions

Prerequisites

- You should have completed the prerequisite courses, especially the **Developing Front-End Apps with React** and **Intermediate Web and Front-End Development** courses.
- You must have completed the following labs:
    - Design Website Layouts
    - Create a GitHub Repository for Your Project
    - Build Static Website Layouts
    - Set Up the React Environment
    - Build the Appointment Booking Component
    - Integrate Existing Functionality
    - Build the Notification Component
    - Build the Reviews Component
    - Build the Profile Component
    - Build and Deploy the Website

Exercise 1: Add keywords for SEO

1. In case you exited and are re-entering the Skills Network lab environment:
    1. Open a new terminal.
    2. Clone your React project's GitHub repository.
2. Add the required keywords for SEO in the `index.html` file in your React project.

    In the HTML `Head` section of a web page, you can use the meta tag to specify keywords related to the page content. For example:

    ```
    <meta name="keywords" content="example, website, products, showcase, demonstration">
    ```

MongoDB | Code Engine | index.html

medical_app > public > index.html

```
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <meta name="keywords" content="example, website, products, showcase, demonstration">
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
```

2. **Take a screenshot** of the section in `index.html` file where you added the seo tags. Then, save the image as **seo.png**.

Exercise 2: Redeploy the application

1. Open a new terminal and execute the `npm install` command from the root directory.
2. Run `npm run build` in the `root` folder, which creates a `build` folder inside your server folder.
3. Make sure that mongoDB server is active and you have pasted the generated password in db.js.

   Refer to the **Set up database connectivity exercise** in the **Set Up the React Project with Backend Connectivity** hands-on lab.

4. Export your Skills Network Lab namespace in the `root` folder and print it on the console using the MY_NAMESPACE variable and the `echo` command.

   ```
   MY_NAMESPACE=$(ibmcloud cr namespaces | grep sn-labs-)
   echo $MY_NAMESPACE
   ```

5. To deploy the application to the IBM Cloud Image Registry (ICR), you will:

   1. Build a container image from your source code using the Dockerfile in your current directory using the `docker build` command.
   2. After building the image, push it to the ICR using the `docker push` command.
   3. Finally, run the image from the ICR using the `docker run` command, specifying the port mappings for your application.

   Note: Refer to the **Deploy the website exercise** in the **Build and Deploy the Website** hands-on lab for details.

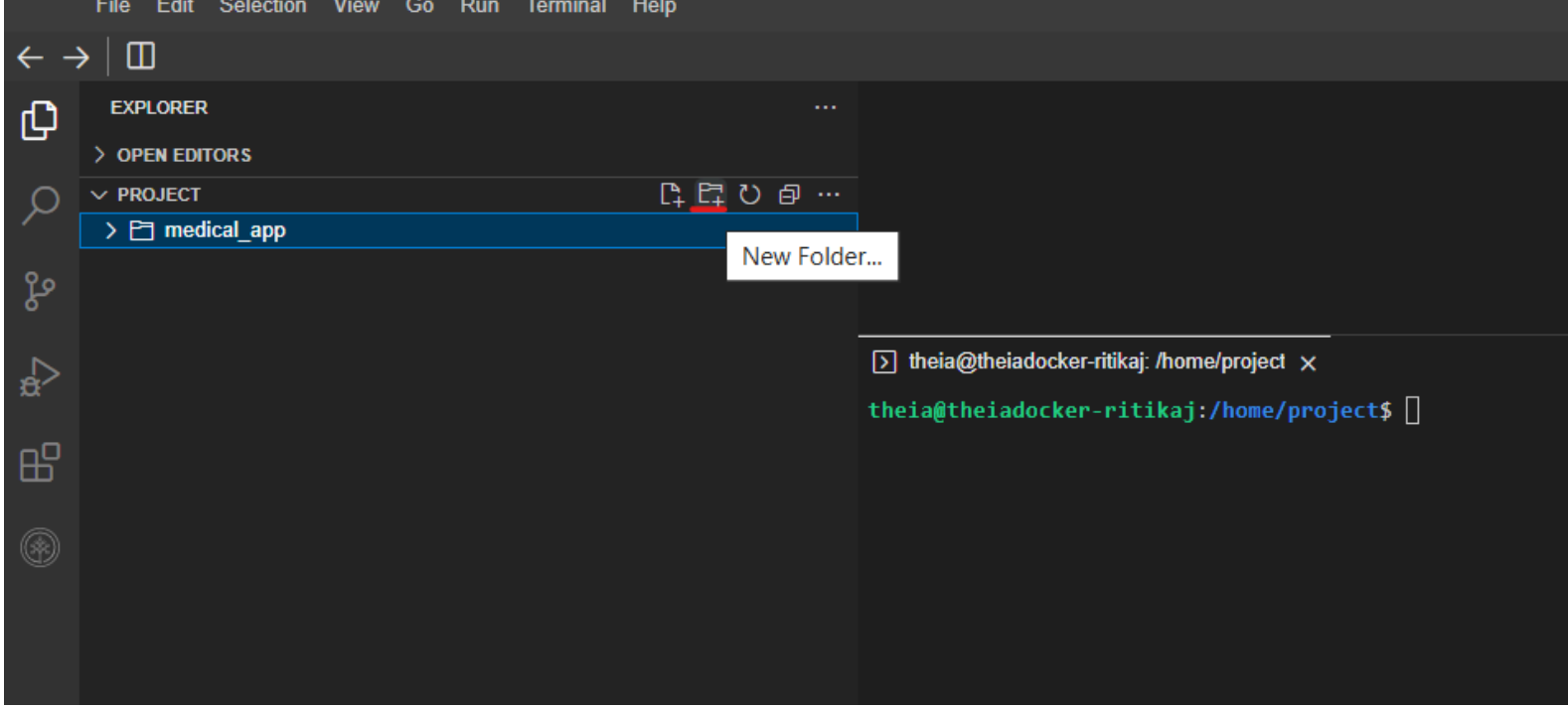
Exercise 3: Commit the changes in your GitHub repository

1. Perform `git config --global` to enter user name and email, `git add`, `git commit`, and `git push` commands to update changes into your React project's GitHub repository for proper code management.

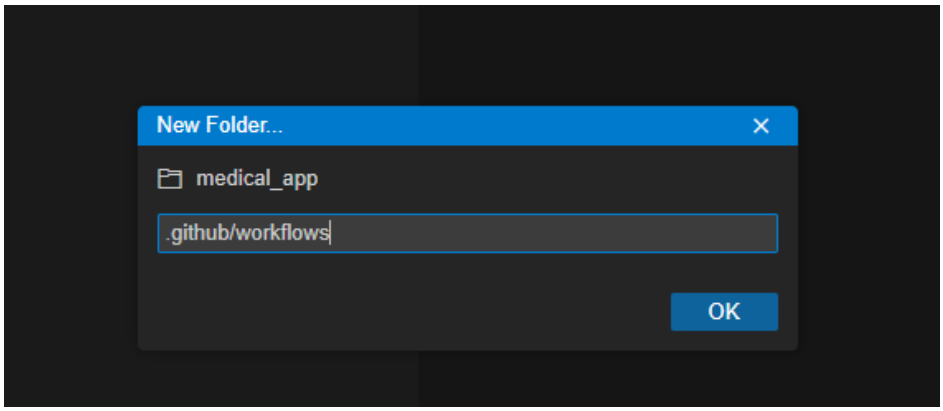
   Refer to the **Capstone Project Reference: Git Commands** reading for details about Git command syntax and use relevant to the project.

[Optional] Exercise 4: Simulate a CI/CD pipeline using GitHub actions

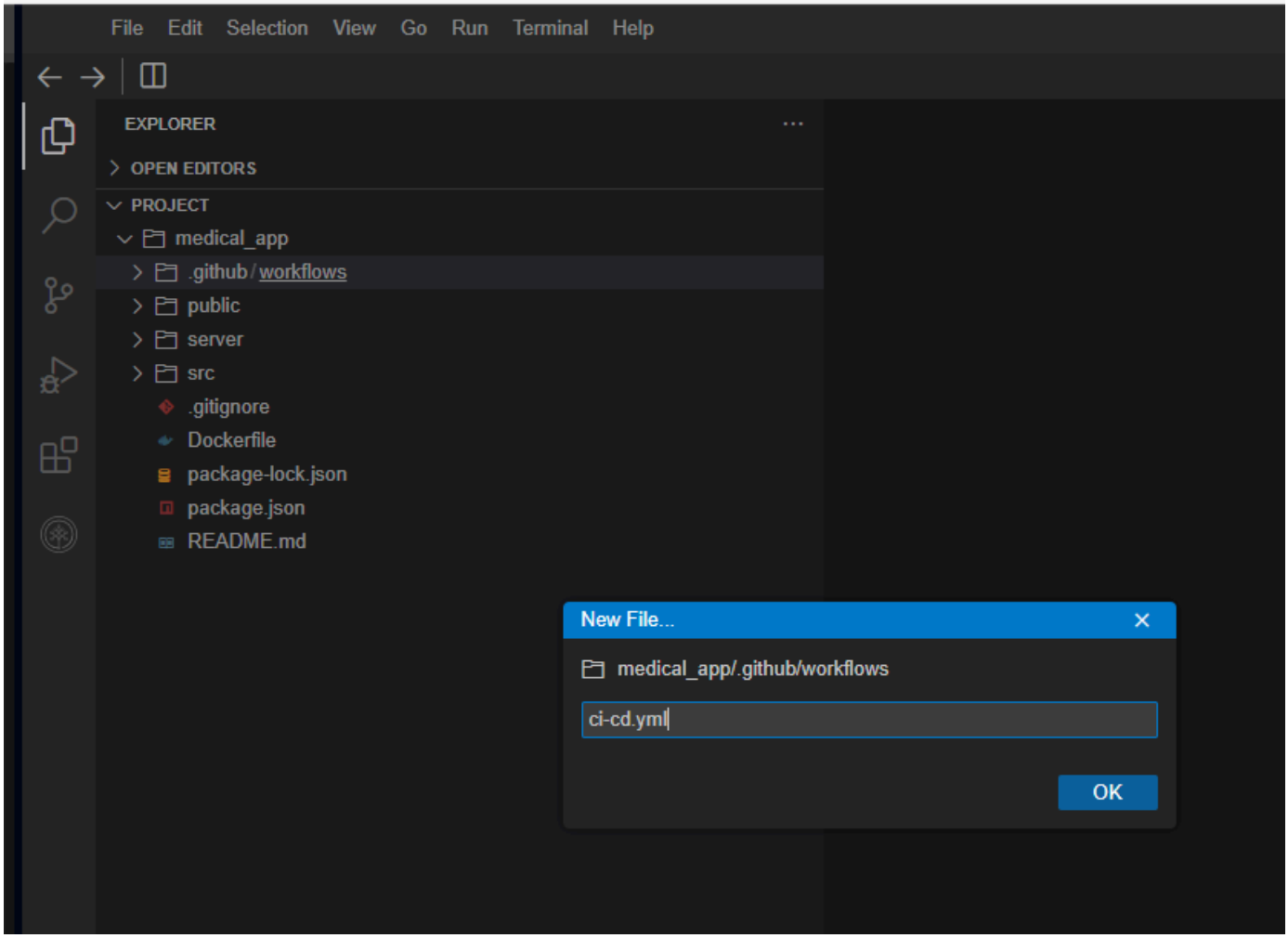
1. Navigate to your `root` directory and click the `New Folder` icon.



2. Provide a meaningful name to the folder, such as `.github/workflows`.



3. Create a `ci-cd.yml` file in the `.github/workflows` folder.



4. Paste the yaml code in the `ci-cd.yml` file.

   Note: If the name of your repository branch is `master`, then replace `main` with `master` in the yaml file.

```
# Name of the workflow
name: 'CI/CD'
# Trigger the workflow on push events to the main branch or master branch
on:
  push:
    branches:
      - main
# Define the jobs to be run in the workflow
jobs:
  # Job to build the application
  build:
    # Run the job on the latest version of Ubuntu
    runs-on: ubuntu-latest
    # Set the environment variable CI to false
    env:
      CI: false
    # Define the steps to be run in the job
    steps:
    # Checkout the code from the repository
    - name: Checkout code
      uses: actions/checkout@v2
    # Setup Node.js with version 14
    - name: Setup Node.js
      uses: actions/setup-node@v2
      with:
        node-version: '14'
    # Install the dependencies for the application
    - name: Install dependencies
      run: npm install
    # Build the application
    - name: Build
      run: npm run build
```

▶ Click here for **code details**.

5. When you modify any file within your code and push the updates to GitHub, the workflow will function as an automated CI/CD process. Any changes you make to your code will automatically trigger the CI/CD process.

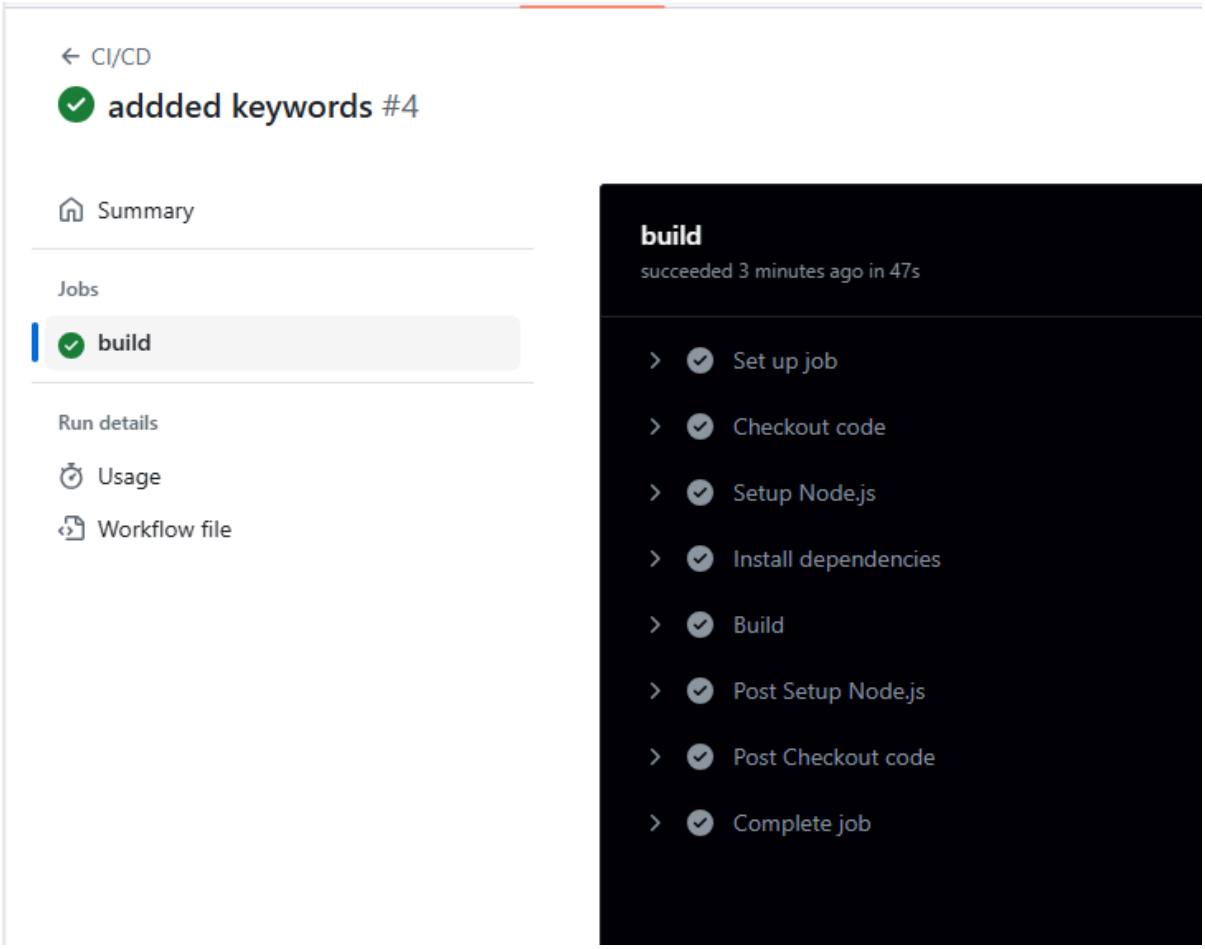
```
theia@theiadocker-ritikaj:/home/project/medical_app2$ git config --global user.email [redacted]
theia@theiadocker-ritikaj:/home/project/medical_app2$ git config --global user.name [redacted]
theia@theiadocker-ritikaj:/home/project/medical_app2$ git add .
theia@theiadocker-ritikaj:/home/project/medical_app2$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   public/index.html
        modified:   server/db.js

theia@theiadocker-ritikaj:/home/project/medical_app2$ git commit -m "addded keywords"
[main 8bb7797] addded keywords
 2 files changed, 2 insertions(+), 2 deletions(-)
theia@theiadocker-ritikaj:/home/project/medical_app2$ git push
Username for 'https://github.com': [redacted]
Password for 'https://[redacted]@github.com':
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 490 bytes | 490.00 KiB/s, done.
Total 6 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), completed with 5 local objects.
To https://github.com/[redacted]/medical_app2.git
   58422e8..8bb7797  main -> main
```

6. In your repository, open the `Actions` tab.

7. Notice that the commit has automatically started the workflow, and after some time, the workflow will be successful, indicating a green tick.



8. After the workflow in your GitHub Actions configuration has been completed, you must rebuild and redeploy your application. The steps include:

    1. Open a new terminal in your lab environment and navigate to the `root` folder.
    2. Use the Docker commands `docker build` and `docker run` to rebuild and redeploy your application.

9. Click **Web Application** to launch the application.

10. You should see the changes on the website.

Web Application

Note: If the web application does not launch when you click the **Web Application** button, you can manually launch it by clicking **Skills Network Toolbox –> Launch Application**. When prompted, specify the port number as **8080**.

Screenshot checklist

You should have taken the following screenshot as part of this lab:

- *seo.png*

Exercise 5: Upload screenshots to the GitHub repository

1. In case you exited and are re-entering the Skills Network lab environment:

    1. Open a new terminal.
    2. Clone your React project's GitHub repository.

2. Use **Explorer** to create a directory named **screenshots** under the `root` folder of the project.

3. Add the following images that you took in the previous labs to the **screenshots** directory.

    - appt_search_design.png
    - appt_doccard_design.png
    - reviews_design.png
    - navbar_layout.png
    - signup_form_layout.png
    - login_form_layout.png
    - react_navbar_folder_struct.png
    - react_navbar_output.png
    - react_signup_folder_struct.png
    - react_login_folder_struct.png
    - react_signup_output.png
    - react_login_output.png
    - docsearch.png
    - doctor_card.png
    - apptform.png
    - appttest.png
    - notification_output.png
    - notification_test.png
    - rating_selector.png
    - review_form_style.png
    - test_rev.png
    - profilecard.png
    - integration.png
    - profileform.png
    - profile.png
    - reportlayout.png
    - report.png
    - style.png
    - test_profile.png
    - test_success.png
    - launch.png

▶ Click here for a **hint**.

7. Perform `git add`, `git commit`, and `git push` commands to update changes into your React project' GitHub repository for proper code management.

    Refer to the **Capstone Project Reference: Git Commands** reading for details about Git command syntax and use relevant to the project.

Note about data management and persistence

it is crucial to follow a few essential steps to ensure the proper management and persistence of your data in a GitHub repository:

- **Regular updates:** Whenever you make changes or add new components to your project, adding, committing, and pushing the updates to your GitHub repository is essential. This ensures that your latest work is safely stored and accessible to collaborators.

- **Session persistence:** During an active session, your data remains accessible. However, it's important to note that if your session expires or you log out, you must clone the repository again to resume work.

- **Ignoring node modules:** When pushing data to GitHub, it's best practice to exclude the node modules folder from both your server and client directories. This folder contains external dependencies and can be large, making the repository heavy and slowing down the process. Adding the folder to the .gitignore file prevents it from being pushed to the repository, keeping your commits cleaner and more focused.

By adhering to these guidelines, you can maintain a well-organized and efficient GitHub repository, ensuring your work is securely stored and easily accessible to you and your collaborators.

Author(s)

Richa Arora

Other Contributor(s)

Ritika Joshi